



HAL
open science

DNS and semantic analysis for phishing detection

Samuel Marchal

► **To cite this version:**

Samuel Marchal. DNS and semantic analysis for phishing detection. Other [cs.OH]. Université de Lorraine, 2015. English. NNT : 2015LORR0058 . tel-01751692

HAL Id: tel-01751692

<https://hal.univ-lorraine.fr/tel-01751692>

Submitted on 29 Mar 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

Contact : ddoc-theses-contact@univ-lorraine.fr

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4

Code de la Propriété Intellectuelle. articles L 335.2- L 335.10

http://www.cfcopies.com/V2/leg/leg_droi.php

<http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm>

Analyse du DNS et Analyse Sémantique pour la Détection de l'Hameçonnage (DNS and Semantic Analysis for Phishing Detection)

THÈSE

présentée et soutenue publiquement le 22 Juin 2015

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Samuel MARCHAL

Composition du jury

<i>Rapporteurs :</i>	Prof. Dr. Eric FILIOL	ESEIA
	Prof. Dr. Eric TOTEL	Supélec Rennes
<i>Examineurs :</i>	Prof. Dr. Ulrich SORGER	Université du Luxembourg
	Prof. Dr. Thomas ENGEL	Université du Luxembourg
	Prof. Dr. Olivier FESTOR	TELECOM Nancy - Université de Lorraine
	Prof. Dr. Claude GODART	Université de Lorraine
<i>Invités :</i>	Dr. Habil. Radu STATE	Interdisciplinary Centre for Security, Reliability and Trust
	Dr. Vijay GURBANI	Bell Laboratories

Mis en page avec la classe thesul.

Remerciements

My first thanks go to the reviewers of this document and to the jury members who accepted to evaluate it. I thank them for the time they spent to read it, for the interest they showed to my work and for the constructive reviews and comment I got out of their evaluation. These helped me to improve this manuscript, to identify some improvements that can be brought to this work and new research perspectives that can be explored.

I faithfully thank my two co-supervisors, Thomas Engel and Olivier Festor for welcoming me in their team at SnT and LORIA during the four years of my Ph.D.. They both provided me a very good support and wise advices while I was doing my research activities. I thank them for their listening, their help and the constructive feedback they gave me. Their supervision has been a key element for the achievement of this Ph.D.

I also want to thank Radu State and Jérôme François. I met Radu Sate while he was my professor at TELECOM Nancy. I discovered and started to do research activities under his supervision. He gave me the taste and the motivation to do research by sharing his work and passion with me. I thanks him for the opportunity he gave me to work with him, for his support and the help he provided me during the past four years. I thank Jérôme François for the supervision and the help he provided me when I started my Ph.D. He put me on the right track from the beginning and we collaborated on many research activities afterwards. It has been a pleasure to work with both of them and they were of great help to produce the results presented in this document.

I want to thank all the people from the SecanLab team (SnT) and the MADYNES team (LORIA). Working within these teams is a great environment to carry research, exchange ideas and produce high quality work. It has been a pleasure for me to work in both during my Ph.D. and I am happy to have spend time there. I address a special thanks to my office mates for the good working environment they provided me. I thank as well people from LORIA and SnT, I interacted with a lot of different people along these years and I am glad that some of them became good friends.

I thank CETREL, the industrial partner for my Ph.D. and more specifically Sam Gabbai and Jean-Yves Decker. It has been a pleasure to work with them and to carry research activities to solve concrete problems. Sam and Jean-Yves have always been of great help and I thank them for the precious time they gave me and their availability during our collaboration.

Finally, I address a special thank to my family and in first place my parents who supported me all along my studies. I thank as well my friends with who I spent good times out of office which helped me to relax and work more efficiently.

To you all, thank you.

*À mes parents,
ma famille.*

Contents

General Introduction	1
1 Context	1
2 Issues and Challenges in Phishing	4
3 Organization of Contributions	5
Part I State of The Art and Background	7
Chapter 1 Phishing and Protection Techniques	9
1.1 Phishing: an Online Con Game	10
1.1.1 Definition and History	10
1.1.2 Phishing Vectors	12
1.1.3 Economic Impact and Evolution	14
1.1.4 Challenges to Fight Phishing	15
1.2 Phishing Prevention Techniques	16
1.2.1 Strong Authentication Schemes	17
1.2.2 Security Toolbars	18
1.2.3 Blacklists	19
1.3 Phishing Detection Techniques	20
1.3.1 Phishing Emails Detection	21
1.3.2 Web Page Content Analysis	22
1.3.3 URL Analysis	23
Chapter 2 Domain Name System Monitoring	29
2.1 The Domain Name System	30
2.1.1 Organization and Implementation	30
2.1.2 DNS Usage	33
2.1.3 DNS Misuses and Security Issues	35
2.2 DNS Monitoring	39
2.2.1 DNS Monitoring Strategies	39

2.2.2	Performance Evaluation and Anomaly Detection	41
2.2.3	Malicious Activity Detection	43

Part II Phishing Domain Names and URLs Detection 47

Chapter 3 Large Scale Passive DNS Monitoring for Identifying Malicious Domains 49

3.1	Passive DNS Monitoring Architecture	50
3.1.1	DNS Data Gathering	50
3.1.2	Distributed Storage and Processing System	53
3.2	Data Mining in DNS Space	54
3.2.1	DNS Features Extraction	54
3.2.2	Domain Names Clustering	56
3.3	Experimental Evaluation	58
3.3.1	Dataset	58
3.3.2	Feature Analysis	59
3.3.3	K-means Clustering Evaluation	61

Chapter 4 Phishing Domain Name Identification Based on Word Relatedness 67

4.1	Phishing URL Obfuscation	68
4.1.1	Obfuscation Techniques	69
4.1.2	Obfuscation Words Semantic	70
4.2	Semantic Analysis of Domain Names	71
4.2.1	Word Extraction	72
4.2.2	Word Relatedness Computation	73
4.2.3	Similarity Metrics	74
4.3	Domain Sets Comparison	76
4.3.1	Dataset	76
4.3.2	Similarity Metrics Evaluation	77
4.3.3	Domains Set Size and Composition	80

Chapter 5 Semantic Based Phishing URLs Rating 85

5.1	Intra-URL Relatedness Analysis	86
5.1.1	URL Word Extraction	87
5.1.2	Shortcomings of Word Relatedness Evaluation Tools	87
5.1.3	Search Engine Query Data	89

5.1.4	Feature Computation	90
5.2	Implementation	92
5.2.1	Distributed Word Relatedness Inference	92
5.2.2	Bloom Filter for Features Computation	93
5.3	Phishing URL Detection	95
5.3.1	Dataset	95
5.3.2	Features Analysis	96
5.3.3	URL Classification	98
5.3.4	URL Rating	101
 Part III Semantic Based Phishing Domain Names Prediction		103
 Chapter 6 Semantic DNS Probing		105
6.1	Smart DNS Probing	106
6.1.1	Hostnames Composition Schemes	106
6.1.2	System Overview	108
6.1.3	Smart DNS Brute Forcer	108
6.2	Semantic Discovery of Subdomains	110
6.2.1	Similar Names	110
6.2.2	Incremental Discovery	112
6.2.3	Splitter	112
6.3	DNS Probing Evaluation	113
6.3.1	Methodology	113
6.3.2	Exploration Parameters	114
6.3.3	Performance Evaluation	116
 Chapter 7 Proactive Discovery of Phishing Domain Names		123
7.1	Modeling a Phisher’s Language	124
7.1.1	Domain Names Features	125
7.1.2	Domain Names Generation Model	126
7.2	Domain Names Features Evaluation	129
7.2.1	Dataset	129
7.2.2	Features Analysis	130
7.3	Phishing Domain Names Generation	133
7.3.1	Types of Generated Domains	134
7.3.2	Efficiency and Steadiness of Generation	136
7.3.3	Predictability and Strategy	138

General Conclusion	143
1 Summary of Contributions	143
2 Research Perspectives	145
List of Figures	149
List of Tables	151
Bibliography	153

General Introduction

1 Context

The power of persuasion has been used for thousands of years to convince people to do things dictated by a leader employing persuasion. This ancestral art is used by politicians, salesmen or lawyers for instance, in order to spread ideas, to sell products or convince a jury, respectively. Even though these examples are legal practices, one may find the ratio of power unfair between people mastering this technique and their gullible victims. The power of persuasion has also been used to perpetrate other activities considered as illegal such as swindling. In a swindle, a crook uses his skills to abuse people credulity in order to make them do actions for his own benefit. This can consist in lending money without warranty, provide services or products without paying, give advance payment for fake sales, etc. These practices have been used by unscrupulous people for centuries in order to make easy money. These tricks were initially performed using direct interaction with victims through convincing speeches. However, time changes and the way to perpetrate swindles as well as their targets changes. Nowadays, other means of communications than direct talk are available through electronic communications like phone calls, emails, instant messaging, etc. Moreover, the direct getting of money is not necessarily the first objective of modern swindling and the acquisition of others valuable immaterial things, like data that can be sold or used to steal money, became more common.

Phishing is an example of modern swindles that targets electronic communications users such as phones and computers users. The same objectives are aimed by e-crooks, who are named phishers, namely to persuade their victims to perform some actions using electronic communications means. Phishers use their power of persuasion to tailor convincing socially engineered emails or websites to manipulate their victims. They use carefully chosen words and sentences to establish a trust atmosphere with their victims in order to push them to perform some actions. Rather than targeting the direct stealing of money or delivery of products for free, phishing mostly aims to steal the victim's confidential electronic data that has become valuable.

The Internet has made it easy to use services that in the past required a more intimate contact between the people conducting the transaction. Some general services such as news providers, education services or science libraries are now available on the Internet. Personalized services such as payment services, banking management services or retail services are also proposed. These personalized services are sensitive because usually dealing with money management and user's confidential information. Hence, the access to these services is valuable in order to steal the information and/or the money stored. For instance, gaining enough personal information about a victim can be used to impersonate him through identity theft. A stolen identity can be used to pose as a person in others swindles in order to hide and protect the identity of the real crook, or to access personal electronic services in order to act in his name. This represents actually the main goal sought by phishers: to steal the required information in order to access sensitive services.

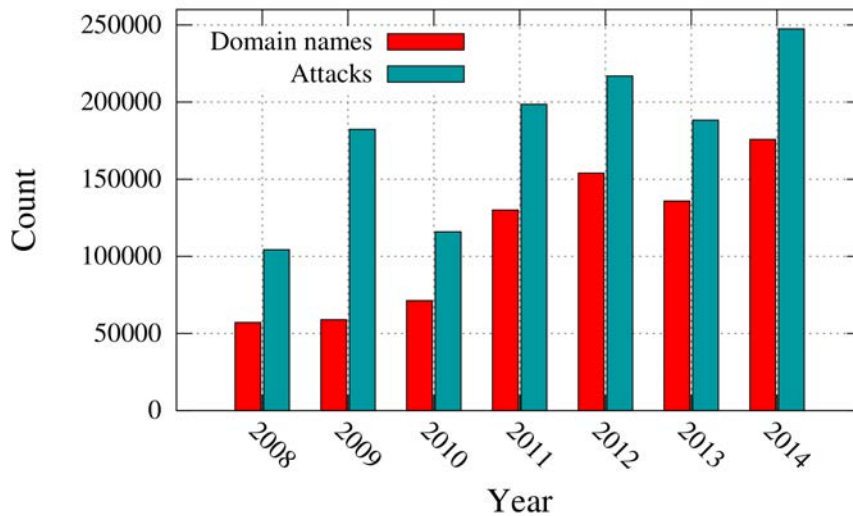


Figure 1: Phishing attacks and phishing domain names recorded every year (source:APWG)

Phishing appeared almost 20 years ago and its first victims were ISP users from which phishers tried to steal the account access information using spoofed emails alleging having been sent by administrators. Phishing attacks usually target users of a given sensitive service related to a brand. Phishers lure the brand clients by alleging to be some brand's representatives in order to ask information related to their usage of the service. This information mostly consists in credentials for a given website or credit card numbers. Several vectors are used for phishing while the mostly used are emails and websites that mimic the ones of legitimate services and alleged to be related to them. Despite this diversity, a common point of many vectors is the use of link misdirecting victims to phishing contents. The use of obfuscated URLs and domain names is widespread in phishing attacks and the use of malicious domain names as a support for attacks is increasing as depicted in Figure 1, showing the relevancy to identify URLs and domain names to fight phishing. This figure shows the evolution of the number of phishing attacks and phishing domain names in use every year between 2008 and 2014. We can see that the count of registered phishing attacks fluctuates between 100,000 and 250,000 globally along the period. However, we can see a regular increase in the count of domain names used as a support for phishing attacks starting from around 50,000 in 2008 and reaching almost 170,000 in 2014.

Over the years, phishing activities dramatically increased in terms of attacks and number of targeted brands [apw04, AR14]. This augmentation of phishing attacks is depicted by an ever increasing financial damage that reached US \$5.9 billion in 2013 [rsa14]. This increase is ongoing since phishing appeared and according to the current trend, this progression will continue. We identified four main reasons explaining this increase and the installation of phishing as a continual threat:

- The first is the increasing of the phishing attack surface. Over the years, the number of Internet users, being potential victims, increased to reach about three billions currently, compared to only half a billion in 2001 [int15]. The same observation can be done for the number of Internet connected devices that is estimated to tens of billions and this number will double by 2020 [cis]. This raises the number of physical phishing vectors that are not limited anymore to desktops or phones but include as well laptops, smartphones or tablets. Finally more and more services that can be targeted by phishing attacks are available on

the Internet, as highlighted by the raise of online websites, reaching almost one billion [net15]. Hence, many new potential victims, physical vectors and targets become available letting space for new kind of phishing attacks to be perpetrated.

- The second reason is the variety of phishing attacks used to perpetrate phishing. Regular phone calls, sms, emails or websites are examples of communication technologies used to perform phishing. Protecting against this variety of vectors is difficult and existing phishing prevention and detection techniques only cope with few of them. Detection techniques for phishing emails [FST07] or phishing websites [MKK08, CDM10, CSDM14] exist for instance, but their application is limited to few attacks compared to the tens that exist. Hence, a global protection implies the use of several independent techniques as we can see today with email filtering, web browser warnings and website authentication techniques that are jointly used to protect against phishing. However, some phishing attacks still succeed to bypass this cumulated protections in order that phishing impact is still progressing.
- The third reason is the increasing number of phishers and attacks perpetrated. The former is explained by the fact that phishing is an easy to perpetrate task requiring low technical skills. The main effort to build phishing attacks is invested in the social engineering tricks used [HCNK⁺14]. This can easily be performed by technically unqualified crooks thanks to the availability of ready-to-use phishing kits [CKV08] and the availability of cheap infrastructures to deploy the attacks. The increase of attacks performed is explained by the decrease of gain per attack forcing phishers to launch more campaigns to keep a constant revenue from their crime [HF08]. Phishing can be qualified as the cybercrime equivalent of pickpocketing since many people are perpetrated it for low revenue. Hence, targeted countermeasures against specific phishers do not cope with this cybercrime since many other phishers would still continue their activities.
- The fourth and predominant reason is the lack of user awareness about the risk associated with electronic communications and the value of the information stored on their several websites accounts. Most people do not understand and are not concerned about the impact of credential stealing, credit card number stealing or identity theft [pon14]. This lack of concerns does not motivate them to protect their data from stealing. Security is a secondary purpose for most users and their limited technical knowledge does not allow them to enhance the security level of their electronic communications [WT99]. New users of modern electronic communications means are gullible and easy targets for phishers who can easily lure them. This widespread unawareness is the main reason of the efficiency of phishing attacks.

Phishing is an ever growing activity that became of major concerns. Many factors explain its expansion and the raise of its financial damage to reach several billions of dollars every year. The variety of phishing attacks, the augmentation of potential victims and physical vectors, the ease to perpetrate this modern swindle and the widespread unawareness of victims make it a troublesome cybercrime activity. Beside its financial impact, phishing raise as well concerns regarding the use of electronic communications means to communicate. People see personal information stealing and misuse as a very-likely-to-occur event in their life [pon14]. This perception of phishing as a fatality and not as a problem that can be prevented leads to erode the trust among electronic communications users. A direct risk of this lose of trust is the decreasing usage of electronic means such as emails as way of communication [HF08]. This renders the fight against phishing paramount to preserve the widespread usage of this useful technology.

2 Issues and Challenges in Phishing

For more than ten years now, many solutions ranging from hardened authentication methods to techniques for identifying phishing websites have been developed to fight phishing. However, the ever increasing number of phishing attacks performed and the monetary damage caused by phishing shows that there is still room for improvement in order to develop techniques that will be able to reverse this increasing trend. The fight against phishing is a challenging task and to develop efficient protection method, one must consider several factors:

- The main challenge in tailoring efficient phishing protection techniques is that phishing cannot be treated as other security issues. Malware infection or network intrusions for instance, rely on the exploitation by an attacker of technical security breaches that are the result of flaws in the implementation of programs or network protocols. However, phishing targets the most vulnerable part of any system: the user. Phishing mostly relies on the use of social engineering tricks and the technical sophistication of phishing attacks is low [HCNK⁺14]. Hence, the technical analysis of flaws exploited by phishing attacks and the adoption of technical countermeasures is not efficient to cope with the problem. Actually, phishing exploits one flaw of current electronic communications: the lack of authentication between users. While several strong authentication techniques exist, these are not mandatory and not understood by most users. Most people are unable to authenticate the identity of the entity they communicate with. Phishing protection techniques must help people to assess the legitimacy of the entity they are communicating with in an easy manner, in order to avoid the impersonation of legitimate entities by crooks.
- A second challenge lies on the difficulty to identify phishes. Since phishers mimic legitimate entities behaviour by sending emails or creating websites that copy the original ones, the differentiation between phishes and legitimate communication is difficult. Many features are common to legitimate communications and phishes and only few differ. The identification of these discriminating features is the main challenge to build reliable phishing protection techniques. This reliability is paramount in order to prevent illegitimate communications while allowing legitimate. On these features depend the adoption and the usage of a protection techniques by users, since users are globally not motivated to use protection techniques [DT05] and ignore them when these are not reliable [ECH08].
- The third challenge is to develop techniques that can cope with the several phishing vectors. Phishing detection techniques usually focus on some categories of phishing attacks like fake websites identification or phishing emails detection. Other techniques are even more limited and target only some specific phishing attacks like browsers windows spoofing [YS02, DT05] or tabnabbing [DRNDJ13]. The development of too specific phishing protection techniques does not provide a good protection against the large range of phishing attacks. To cope with this, the accumulation of case-specific phishing protection technique is needed to provide a wide protection coverage. To operate, these cumulated techniques require a large computation time, introducing thus a delay in the identification of phishes. A long delay can impact the usability of a protection technique if it aims a real-time usage.
- A last challenge lies on one characteristic of phishing attacks namely their short lifetime. Phishing attacks last on average less than one day and often only few hours [apw14]. Even though, this lifetime is short, the financial damage of these attacks is high. To limit this damage, protection techniques must fast identify phishes. This requirement raises issues

about the usage of off-line phishing detection methods in this context. Efficient phishing protection techniques must focus more on on-the-fly identification of phishes to limit the impact of an attack. However, such methods must be used in a context of current usage of electronic communication means such as exchange of instant messages, or web surfing. Hence, the proposed method must not impact users' experience and must not introduce large delay that would prevent their usage.

3 Organization of Contributions

Seeing the characteristics required by an efficient phishing protection method in term of speed, coverage, reliability and ease of use, we propose in this manuscript new techniques that can cover these requirements. We exploit the fact that phishing attacks are a kind of modern swindle. Phishers are crooks employing their persuasion power to convince their victims to act for their benefit. They employ carefully chosen words in their communications to establish a trust atmosphere and delude victims. Based on this fact we propose to analyse the meaning and semantic of words used by phishers in order to detect messages produced by them. To cover a large set of phishing attacks, we analyse the semantic of URL and domain names. These resource locators are used in a large range of phishing attacks to misdirect users to malicious contents. The identification of phishing URLs leads to cope with several phishing vectors and that is why it is currently used as phishing protection method in reactive URL blacklists [goob, mic]. However, to cope with the slow process of crowd verification used by blacklist, we rather perform a real-time analysis of URLs and exploit the semantic of words embedded in them. Observing the increasing usage of malicious domain names to perform phishing attacks, as presented in Figure 1, we focus as well the semantic analysis on domain names and explore the possibility of predicting domain names used for phishing by analysing phishing domains composition and semantic.

This document is structured around two main research directions related to phishing URLs and domain names detection and phishing domains prediction:

Part I: State of the Art and Background. This part gives the necessary background to position the contributions provided in this document according to the working context of phishing and domain name analysis. Chapter 1 defines the concept of phishing attacks and presents some of the most used phishing vectors. We provide an overview of the phishing nefarious impact and list the requirements to develop efficient phishing protection methods. The existing techniques developed to cope with phishing are presented and we identify their weaknesses and their ability to meet the formulated requirements. Chapter 2 presents the organization and functioning of the Domain Name System. An overview of the different usage of DNS monitoring techniques are presented and we argue about the relevancy of using DNS monitoring to identify phishes.

Part II: Phishing Domain Names and URLs Detection. This part presents the first contributions of this document in developing techniques to identify domain names and URLs used in phishing attacks. Chapter 3 introduces a domain name clustering technique based on passively captured DNS data. The method is able to group domain names according to their activity and to discriminate phishing from legitimate domain names. This is further used in Chapter 4 as a pre-process to group domain names. Chapter 4 introduces a technique to infer the legitimacy or maliciousness of a set of domain names using semantic analysis. Metrics quantifying the semantic similarity between two sets of words are introduced and used to compare words extracted from legitimate and phishing domain names. These metrics allow to differentiate

phishing from legitimate sets of domain names. Chapter 5 introduces a URL phishing detection technique relying on the analysis of intra-URL relatedness. Search engine query data is used to quantify the relatedness between the registered domain name of a URL and the remaining of it. It is showed that legitimate URLs present more intra-relatedness than phishing URLs. The proposed technique relying on a machine learning algorithm is able to identify phishing URLs with an accuracy of 95% and a process time of less than a second thanks to a distributed processing architecture.

Part III: Semantic Based Phishing Domain Names Prediction. This part explores the possibility to predict domain names that will be used by phishers. The predictable character of domain names is explored in Chapter 6. We present a technique relying on the finding of semantically related words in order to discover the different subdomains of a domain name. Based on a set of known subdomains this techniques is able to discover new subdomains and outperforms existing state of the art techniques showing the validity of using semantically related words to predict domain names. A similar technique is used in Chapter 7 to generate a predictive phishing blacklist. A domain name generator relying on a Markov Chain model using semantic extension is introduced. Learning from a set of existing phishing domains the generator is able to produce domain names that will be used for phishing activities and this even long time before these are used. This work shows that phishing domain names follow specific composition schemes and use words restricted to a limited vocabulary such that these are predictable.

The dissertation concludes that lexical and semantic analysis performed on domain names and URLs is relevant to build phishing protection methods. This analysis combined with other data sources such as DNS information shows good results in the identification and prevention of phishes. It meets three essential requirements for a phishing protection that are speed, coverage and reliability.

Part I

State of The Art and Background

Chapter 1

Phishing and Protection Techniques

Contents

1.1 Phishing: an Online Con Game	10
1.1.1 Definition and History	10
1.1.2 Phishing Vectors	12
1.1.3 Economic Impact and Evolution	14
1.1.4 Challenges to Fight Phishing	15
1.2 Phishing Prevention Techniques	16
1.2.1 Strong Authentication Schemes	17
1.2.2 Security Toolbars	18
1.2.3 Blacklists	19
1.3 Phishing Detection Techniques	20
1.3.1 Phishing Emails Detection	21
1.3.2 Web Page Content Analysis	22
1.3.3 URL Analysis	23

Introduction

The increasing usage of e-services (e.g. e-banking and e-commerce) during the last decades saw the emergence of new threats associated to these services. The valuable information handled by these services attracted miscreants seeking to steal this data and use it for lucrative purposes. One example of such cybercrime activities is phishing. The first appearance of this term dates back to 1996 and refers to the attack perpetrated against America Online (AOL) where scammers posing as AOL employees sent messages to ask customers for confidential information. Although this was the first recorded occurrence of a phishing attack, phishing became commonly known by ordinary people only ten years later. Now, twenty years after its appearance, phishing has become one of the most lucrative cybercrime activities causing billions of dollars of loss every year [gar07, str10, rsa14]. Although several techniques have been developed to cope with phishing during previous years, its economic impact is still increasing over time [rsa14]. Methods to perpetrate phishing evolve at the same pace as protection techniques, making it a continual threat.

Phishing is a criminal mechanism employing technical subterfuges and social engineering to abuse the credulity of uninformed users. The technique usually consists in masquerading as a trustworthy entity in order to convince an individual to perform an action that he would only do if asked by the impersonated entity. In most cases, this action consists in providing credentials information for e-services access, providing credit card information, downloading and installing malware, etc. The fight against phishing is difficult because phishing targets the most vulnerable part of the system: the user. As described in [DT05], in phishing both system designers and attackers battle in the user interface space to guide (or misguide) the users. Hence, the problem cannot be tackled as a traditional system or network security issue but must heavily consider the human factor. Most phishing attacks can be detected by experienced users but for basic Internet users, security is a secondary purpose and they are not motivated and skilled enough to properly identify phishes. Phishing protection methods must consider the human factor and more precisely the limited skills of users and the "unmotivated user" property [WT99]. It is shown in [SHK⁺10] that half of unexperienced users fall for phish and that even after being trained almost a third of studied users are still tricked by phishing attacks. Thus, developing efficient protection techniques is challenging and raise several requirements like ease of use, speed or performance.

Some automated and easy to use methods have been proposed to protect users from phishing. Phishing email filtering techniques [FST07, RW12, AKS14], security toolbars [CLTM04, GPGL11] and Web browser phishing warnings [goob, mic] are examples of such techniques. These methods can be classified in two categories being phishing prevention methods, helping to prevent exposition to phishing by enforcing authentication for instance, and phishing detection methods, which analyse a given email or web page in order to assess its legitimacy. The scope of some techniques is however limited to specific phishing attacks and the identification delay is important. The variety of means used to perform phishing and the short lifetime of phishing attacks make these solutions often inefficient or easy to bypass, requiring new solutions to be proposed. Despite more than ten years of fight against phishing, its nefarious impact is still growing.

We start in this chapter by defining in Section 1.1 what phishing is and present the means used to perpetrate this task as well as an overview of the economic impact and evolution of phishing activities. Based on the observations, we define the requirements to meet for efficient phishing protection. Section 1.2 presents three techniques to prevent phishing attacks and give some examples of implemented solutions. Methods for phishing detection are presented in Section 1.3 and we identify the strengths and weaknesses of each state of the art technique.

1.1 Phishing: an Online Con Game

Phishing has been a continual threat present for almost 20 years. The range of malicious activities and attacks categorized as phishing is wide and some have few similarities with each others. We first give a definition of phishing including the different aspects of the activities it includes and provide a short history of phishing. Then, we present some phishing attacks and vectors leveraging technical subterfuges and social engineering. We provide an analysis of the phishing evolution in term of economic impacts, attacks performed and techniques used over the years. Finally we present the several challenges to develop efficient phishing protection solutions.

1.1.1 Definition and History

The term phishing, which was also referred to Web spoofing at the beginning, comes from the fact that scammers used spoofed emails and websites as baits to lure users. Hence, *phishing*

refers to *fish*ing. The "f" was replaced by "ph" to refer to one of the earliest form of hacking against telephone networks and was called *phone phreaking*. Hence, "ph" became a common hacking replacement pattern for the character "f".

Accurately defining phishing is difficult and available definitions provided by the literature are inconsistent. According to PhishTank [phi]: "*Phishing is a fraudulent attempt, usually made through email, to steal your personal information*". Similarly, Webster dictionary defines phishing as "*a scam by which an email user is duped into revealing personal or confidential information which the scammer can use illicitly*". The website *phishing.org* describes phishing as a "*Process where a targeted individual is contacted by email or telephone by someone posing as a legitimate institution to lure the individual into providing sensitive information such as banking information, credit card details, and passwords*". In [JJJM07], a more general definition states that "*Phishing is a form of deception in which an attacker attempts to fraudulently acquire sensitive information from a victim by impersonating a trustworthy entity*".

These definitions are however too restrictive since phishing is not limited to stealing personal information. The delivery of malware or trojan horses through phishing emails claiming for software updates is a common phishing practice. This does not steal any personal data. Moreover, the given definitions state that the attacker is acting on behalf of a third party, which is not always true. Some phishing attacks consist in alleging to deliver safe contents while these actually deliver malware, although phishers are not masquerading as a known trustworthy entity. As a result we select the definition from [KIJ13], which covers the several aspects of phishing attacks: "*Phishing is a type of computer attack that communicates socially engineered messages to humans via electronic communication channels to persuade to perform certain actions for the attacker's benefit*".

The first appearance of the term *phishing* occurred in early 1996 [Oll05]. Following the measures taken by America Online in late 1995 to cope with the opening of AOL accounts using fake credit card numbers, AOL crackers resorted to phishing for legitimate accounts. Masquerading as AOL staff members, phishers started to send instant messages to legitimate AOL users asking for "*verification of account information*" or "*confirmation of billing information*" in order to steal their passwords. Lots of users fell for the ruse and AOL started to include warnings to prevent users from revealing their personal information through such methods. The main use of these stolen accounts was to send spamming emails. In 2001, after targeting Internet service providers, phishers started to focus their attention on more profitable Internet actors like e-payment and e-commerce services. A first attack against E-gold service, which was a digital gold currency, was unsuccessful. However, in late 2003 phishers maliciously registered dozens of domain names looking like belonging to eBay and PayPal. Shortly after, they used worms to lure PayPal customers with spoofed emails. These emails contained redirection links to fake websites and asking them to update their credit card information. This introduced the most used phishing attack where mass mailing of spoofed emails embedding links direct to fake websites. By 2004, phishing has been recognized as an industrialized part of the economy of crime and the study of this underground economy started.

Since phishing appears, the original technique to perform it with emails and websites spoofing remains the most widely used nowadays. Even though several techniques, more elaborated and complex, have been developed, the easiest and most used means targets the unawareness of unsavvy Internet users is still the most efficient. The technical sophistication of phishing attacks is fairly low compared to the sophistication of the social engineering used [HCNK⁺14]. Hence, the fight against phishing still mostly consists in detecting and preventing connections to fake websites and detecting illegitimate emails.

1.1.2 Phishing Vectors

The vague definition describing phishing as "*a type of computer attack that communicates socially engineered messages to humans via electronic communication channels to persuade to perform certain actions for the attacker's benefit*" is the result of the several techniques used to perpetrate this activity. Phishing attacks rely on combined technical subterfuges and social engineering tricks. Techniques to perform phishing can be categorised in three categories being web-based phishing, communication-based phishing and malware-based phishing. In addition these can be divided in two other categories as either these perform identity spoofing or not. Figure 1.1 presents several phishing vectors according to this classification. In addition it provides the kind of intended actions associated with these vectors *i.e.* information stealing or malware delivery. It shows as well the possible redirection between the several phishing vectors. The majority of perpetrated phishing attacks belong to web-based and communication-based phishing using mostly identity spoofing. We describe some widespread techniques of these categories causing the largest financial loss [rsa14] and being the ones principally targeted by phishing detection [ZHC07, FST07, MKK08, MSSV09b].

- **Emails:** Emails alleging to be sent by a trustworthy entity ask receivers to perform some actions. The email purpose is generally the update of personal information such as website passwords or credit card information, the confirmation of a transaction requiring personal information, etc. This was the first phishing vector used against AOL users to steal their account information. E-services companies quickly warned users against providing personal information within emails. This raised users awareness such that most users did not fall for this trick anymore. Hence, phishers started to ask for software update alleging for security issues and actually delivered malware. Email embedding URLs and links directing to fake websites were used instead of directly asking for information in email replies. Links directing to drive-by download are also inserted in emails. A drive-by download is the unintended download and installation of a software, usually a malware.

The email sender address is often forged (or spoofed) to pretend to be sent from a trustworthy entity. Moreover, phishing emails usually include company logos and security indicators to make users feel confident with the legitimacy of the email. Phishing emails can target specific entities or people but these are mostly sent to thousands or millions of users as part of an email spamming campaign. Instant messaging services are used the same way as emails, sending link directing to fake websites. These represents the two most common communication-based phishing techniques with identity spoofing.

- **Fake websites:** Websites having a similar and sometimes identical look and feel as popular websites. These have no relation with the website being spoofed and are managed by a phisher. Websites being copied are mostly related to banking activities, payment, retail and software brand. People trying to log into a fake website, thinking it is legitimate, enter their usernames and passwords, which are stolen by phishers. After login, users are sometimes redirected to the real websites making the phish invisible. Some additional information can be asked, like credit card information for fake payment service websites. Phishing websites related to software providers deliver malware as updates or propose charged updates. Internet users are directed to these fake websites via embedded link in emails, instant messages or in other web pages. These links are in the form of Uniform Resource Locator (URL) based on domain names that are not the ones of the targeted brand. Three techniques are used to build a phishing URL leading to fake websites. These are either domain hijacking,

DNS spoofing or registration of free domain names for malicious purposes, which is mostly used.

- **Low-cost product websites:** Legitimate websites that propose products at low cost or download of movies and games for free. When users pay a low amount for products through these websites, the credit card information is stolen and no product is shipped. Downloaded games or movies usually contain malware, crimeware or spyware that will be installed on users' computers.
- **Content injection:** A phisher changes a part of the content of a legitimate webpage. This modified part can either embed forms where users can enter personal information or include link directing to a fake website. Content injection can be done using cross-site scripting.
- **Spoofed browser windows:** This technique is similar to the content injection except that the web page is not modified. An illegitimate browser window is placed on top of or next to a legitimate window. This illegitimate browser window spoofs a password dialogue window usually. While visiting a website users can enter username and passwords in the prompted spoofed window instead in the legitimate fields and the phisher gains this information.
- **Tabnabbing:** An attack that employs scripts to rewrite the content of a web page of average interest with the content of popular websites. This exploits the ability of modern web pages to rewrite their content long time after these are loaded. Unattended open tabs in a browser can thus change their content to adopt the look and feel of popular websites while these are not used. When users return to the web page, which has changed in the meantime, they may not remember it was not the web page they visited first and log into it giving their credential and password to the phisher.
- **Web proxy:** A phished user on a shopping website is redirected to a server controlled by the phisher. The server acts as a proxy between the user and the shopping website delivering the legitimate content to the user. Once the user has finished his shopping and wants to make the purchase, the proxy serves up a modified web page that drives the user through a checkout process designed to extract personal information.
- **Fake antivirus (AV):** A malware being installed on a computer alleged to be the free version of an antivirus software. The fake AV claims that the computer is infected with several malware and an upgrade of the antivirus to the full version is needed to remove the viruses. The fake AV often blocks all the functionalities of the computer and the only possible action is to make the upgrade. The upgrade costs money and requires credit card information to be installed. Phishers take the payment and steal the payment information. Fake antiviruses are usually installed through drive-by download.

These represent the principal means used to perform phishing. Some other phishing scams such as VoIP phishing (or *Vishing*) or SMS phishing (or SMiShing) can also be cited. However, prevention techniques against these computer-based phishing attacks are different from the previous phishing techniques that are mainly web-based phishing attacks. Some malware products such as key loggers, trojan horses or fake antiviruses are considered as well as part of phishing attacks. The prevention and detection of these attacks rely though more on malware analysis than on phishing detection. Moreover, these malware applications are mostly delivered to users and installed using phishing vectors dedicated to malware delivery as presented in Figure 1.1. Phishing emails are the primary vector of malware spreading [HCNK⁺14]. Hence, the identification of other phishing vectors is efficient to cope with malware-based phishing.

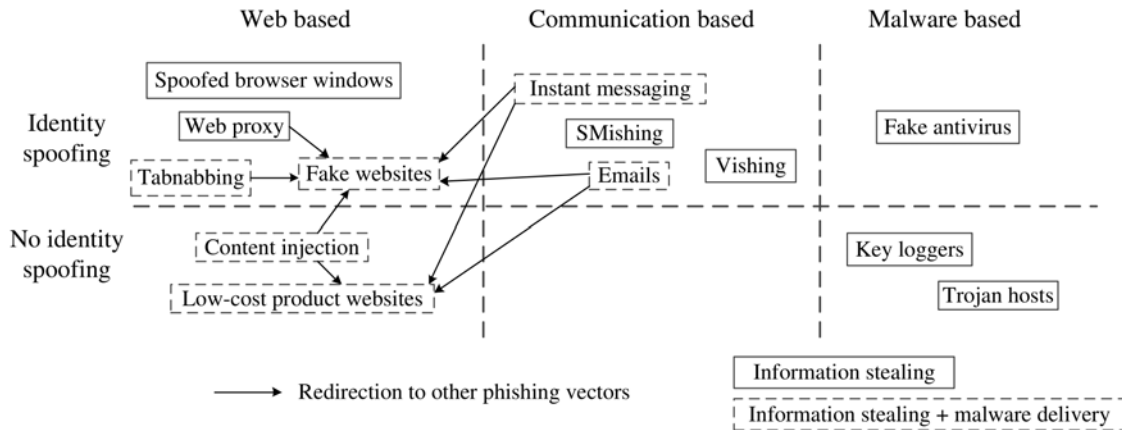


Figure 1.1: Phishing vectors classification

1.1.3 Economic Impact and Evolution

A first study on the phishing impact [apw04], made by the *Anti-Phishing Working Group*, reported some hundreds phishing attacks per month at the end of 2003. Back then, the most targeted industry sector was financial services and only 24 brands were targeted by phishing.

In ten years, phishing activities have grown above expectations despite the effort engaged in the fight to cope with it. The latest report [rsa14] states that around 60,000 phishing attacks are observed every month. This increased more than twofold since 2010 where around 200,000 phishing attacks were reported for the whole year. These phishing attacks are mainly web-based attacks since more than 40,000 unique phishing websites are detected every months and around 60,000 unique phishing emails are reported every month in 2014 [apw14]. This proves that phishing emails and fake websites are still mostly used in current phishing attacks. Many domain names are registered to direct users to phishing websites since around 10,000 maliciously registered domain names were used for phishing every month in 2014 [apw14]. This shows an increase compared to 2012 where only 7,712 maliciously registered domains were used during the whole first semester [AR14]. Alternative techniques to direct users to fake websites such as domain hijacking or DNS cache poisoning are cumbersome tasks that require technical skills. Registering free domain names for malicious purposes is a much easier task for phishers explaining the increasing trend in the use of maliciously registered domain names.

As the Web was growing, the count of brand targeted by phishing raised. More than ten industry sectors are targeted by phishing nowadays. These mostly consisted in ISPs (AOL) at the beginning and started to target financial institutions few years later. Now, social network, online gaming, government or retail websites are targeted as well. As depicted in Figure 1.2, the most targeted industry sector is payment services (e.g. PayPal, Visa, MasterCard, etc.) that represents almost 40% of the phishing targets. ISP and financial institutions are still targeted but in lower proportion, 8.42% and respectively 20.2%. Compared to the initial 24 brands targeted by phishing in 2003, around 350 different brands are currently targeted [apw14]. This renders the protection of brands with dedicated solutions [eba] difficult since one would have to install and use too much add-ons to stay safe.

The raise of phishing targets caused as well the raise of financial damage. No accurate estimation is available to quantify the financial loss due to phishing. However, in 2007 a Gartner

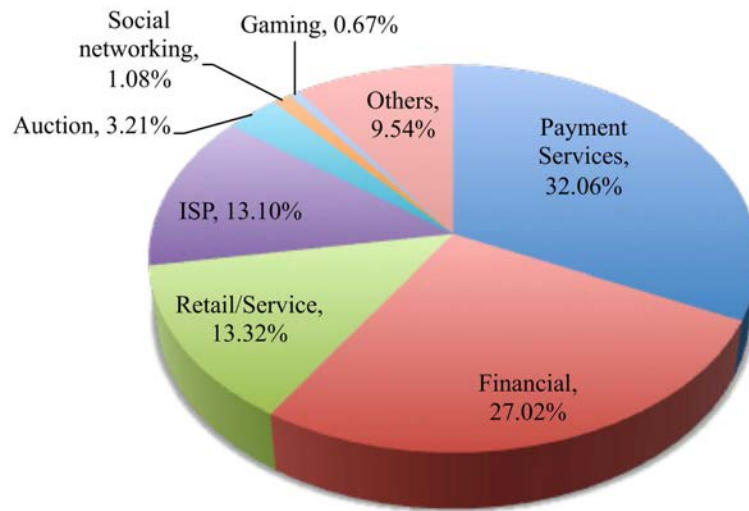


Figure 1.2: Most targeted industry sectors for the 3rd quarter 2014 (source: APWG)

survey [gar07] estimated a direct financial loss of 3.2 billion dollars over the year due to phishing. In 2010, a report [str10] about identity theft mainly performed with phishing attacks presented an estimation of 54 billion dollars loss as consequence of this theft. Lately, in 2013, the estimated direct loss over the year due to phishing was of 5.9 billion dollars reaching a record [rsa14]. As described in [AR14] the first day of a phishing attack is the most lucrative and thus fast take down of phishing websites is paramount to limit the financial damage. Between 2010 and 2014, the average uptime of phishing websites has been divided by two, dropping from 72 hours to 32 hours. More significant is the median uptime of phishing websites that decreased from 15 hours in 2010 to less than 9 hours in 2014. The development of phishing detection and prevention techniques reduced the delay of fake websites take down, thus reducing the financial loss due to each phishing attack. While causing important loss, phishing is a low reward activity for phishers [HF08]. Hence, the number of phishing attacks constantly increases to increase the gain, having for consequence the increase of victims and the raise of the global financial loss. The availability of easy-to-deploy phishing kits [CKV08] that target both users and victims makes performing phishing attacks an easy task and explain the raise of perpetrated attacks and the increase of phisher's count. Phishing is the cybercrime equivalent of pickpocketing. It is easily perpetrated with low technical skills and low cost to the attacker thanks to phishing kits and cheap criminal hosting services.

1.1.4 Challenges to Fight Phishing

The observation of phishing evolution shows that there is still room for improvement in techniques to cope with this cybercrime. Despite the progress made for fast take down of phishing websites, the global damage of phishing is still increasing due to the increase of attacks. Besides the financial damage caused by phishing attacks, phishing causes an erosion of trust among Internet users and start to destroy emails as a way of communicating. Hence, the development of new means to deal with this problem is an ongoing activity to reverse the loss growth trend.

The variety of phishing vectors and the ease to perform phishing attacks makes the fight against it very challenging. We identified six requirements to develop efficient protection techniques. These consider the characteristics of phishing namely the variety of vectors used, the

speed of phishing attacks and the human factor. The requirements are the followings:

- **Speed:** Since phishing attacks make large monetary damage in few time and especially during the first hours of a phishing attack, the identification of a phish must be fast to limit the nefarious effects. If we consider phishing detection methods, these are involved while users are surfing the Web or consulting emails, thus a detection method must not impact the quality of the user's experience by introducing large delay.
- **Coverage:** Phishing defences must be able to prevent against as many vectors as possible and a perfect phishing protection method would be able to deal with the several techniques presented in Section 1.1.2 to provide the best protection.
- **Information required:** The best phishing detection method must be easy to implement and must not rely on several information. Some phishing detection methods rely on several kind of information that is not necessarily available, which limit their applicability. In addition the retrieval of this information is often time consuming, which impacts the speed of the methods.
- **Reliability:** Phishing protection methods must protect from the most phishing attacks, as described in the coverage requirement. However, these must identify as low as possible legitimate communications as phishing attacks. Too many false alarms can affect badly the user experience. In addition, it can cause the loss of confidence in the protection technique impacting the user consideration.
- **Ease of use:** Methods must be easy to use and to understand by users. Most users and especially phishing victims have few technical knowledge and few knowledge of the way phishing attacks are performed, explaining why they are trapped. Hence, phishing protection must consider this parameter and be tailored to be easily used.
- **Actual usage:** This requirement is a consequence of the previous one and evaluate if the protection method can be actually implemented and used by users or if it is just ignored or too complicated to implement. This mostly evaluates the ability of a given method to cope with the unmotivated user property.

Several methods have been develop to cope with phishing and have been proved efficient in identifying attacks such that the time of phishing attacks globally decreased. Despite this efficacy, phishing is still an ongoing problem and these did not succeed to get rid of this threat. To understand the cause of this unsuccess, we analyse to which extent state of the art phishing protection methods meet the introduced requirements.

1.2 Phishing Prevention Techniques

For more than ten years the research activities to develop efficient techniques to cope with phishing have been very active. The solutions proposed can be divided in two main categories according to their goals that are phishing prevention and phishing detection. The former methods were the first to be introduced in order to prevent any connections to phishing websites for users. These techniques were mainly focused on three fields that are the development of strong authentication protocols, security toolbars to raise user awareness about phishing danger and the implementation of blacklists.

1.2.1 Strong Authentication Schemes

While accessing banking services, retail services or social network on the Internet for instance, one is expected to enter some personal information (*e.g.* username, password, etc.) to prove his identity to the web service and access his account. This is needed since the account associated to such service contains some other more sensitive information about the user that must be protected from access of others. This is the process of authentication required by any Internet service dealing with personal data. This process has been implemented with some flaws letting unauthorized users accessing this information. The use of weak passwords and their reuse by unsavvy users make the authentication process subject to attacks. Moreover, the reverse process, consisting in authenticating a website to a user, has been proven weak. Users cannot reliably identify the entity they communicate with, letting space for phishing attacks to occur.

Hence, some solutions have been proposed to fight phishing by developing strong authentication processes allowing both parties (client and server) to prove their identity without having to reveal sensitive information during the handshake.

The first proposed solutions consisted of browser extensions aiming to help users distinguish content provided by legitimate websites from content provided by illegitimate entities. This is to avoid that users enter sensitive information in illegitimate fields and to prevent thus credentials stealing. In [YS02], a browser extension giving different colors to window boundaries is introduced. Two kinds of windows are depicted to users according to the nature of the window content. It allows to distinguish browser provided status from web server provided contents. Easily distinguishable windows help users to detect malicious content provided by web servers and prevent them for providing sensitive information to illegitimate entities. In [DT05], a similar solution of tuned browser windows is proposed to authenticate a web server. It uses unique abstract images to display a dedicated password window for users to enter their credentials and log in to a given website. The look of the window is different for every user and transaction and generated based on a shared password between the user and the server. Hence, users can easily see if the displayed password window is a spoofed window since such window would not have the expected look for a given legitimate website.

Methods for improving servers authentication have been proposed like in [TH09]. DNS TXT records are used to store the legitimate entity's certificate and authenticate to the client. A client plugin and a server plugin are used. The client plugin authenticates the web server by validating the certificate stored in the DNS TXT record. Then both plugins realize a mutual authentication using a one-time password.

One bad habit of Internet users is to use similar weak passwords for different websites. These can be easily guessed by phishers and one stolen password can give access to several accounts. To cope with this bad habit some techniques to strengthen and differentiate passwords have been proposed [RJM⁺05, YS06, GLLA07]. For instance, Ross *et al.* introduce a browser extension named PwdHash [RJM⁺05]. This extension transparently produces different password for each site a user wants to sign in based on a single password. It relies on hash functions that generates a password from the unique user password and some data associated with the website that cannot be spoofed by a phisher: the domain name. It strengthens web password authentication and prevents to provide user's password to phishers in fake websites since fake websites have a different domain name than the one they spoof.

Other techniques to strengthen the authentication process proposed the use of a second factor authentication other than a password. Nikiforakis *et al.* introduce the concept of Past Activity Tests (PACTs) to authenticate on a website [NMAM09]. A question related to past actions of the user while he last connected is asked to log in. Phishers using stolen credentials are unable

to reply to this kind of questions since it is the first time they try to log in and have denied access. The use of an additional trusted device like a cellphone for authentication is introduced in [PKP06]. A robust authentication technique relying on javascript is presented in [BJKP14]. This relies on a token delivered with a secure channel that is stored on the web browser and further used for every authentication.

On the one hand, the latest presented methods having for aim the strengthening of users authentication are efficient to avoid passwords stealing and the reuse of them by phishers. Some of the techniques proposing a second factor authentication have been adopted by sensitive services such as e-banking services to enforce the security. On the other hand, methods for strengthening server authentication while having good foundations did not improve the situation. Contrarily to strong user authentication that is imposed by e-services, strong server authentication is up to Internet users. Since most Internet users are unaware of the phishing dangers and since security is a secondary purpose, these optional methods have not been adopted. These security solutions [YS02, DT05] are not mandatory, difficult to understand and globally add constraints to users for their primary purpose of surfing the Web. Other techniques [TH09] being transparently used still have some flaws like being vulnerable to DNS cache poisoning attacks. Even though some of these techniques would be widely implemented, most users do not understand the provided security indicators proving authenticity of the entities they communicate with [HJ08]. This limits the applicability of server authentication techniques.

1.2.2 Security Toolbars

To cope with the unmotivated user property to use security enhancement solutions [WT99] and the incapacity for users to efficiently exploit security indicators provided to them by web browsers, some browser extensions also called security toolbar were developed. Efficient indicators provided by web browsers can be used by mature users to easily infer the legitimacy of a website. One of these indicators is the address bar showing the URL of the consulted web page. One can easily see whether the domain name of the web page is the expected one. Another indicator is the use of a secure connection depicted by the use of the HTTPS protocol in the address bar. Finally, for the authentication of the web server one communicates with, the verification of the use of Transport Layer Security (TLS) certificates with a padlock icon is fast. Further investigation about the issuer of the certificate can be done by a simple click. All these indicators provide the necessary information to authenticate a website. However, a small part of users use them and that is why browser extensions or security toolbars were developed to ease the access to this information and give some additional information to users.

The Netcraft toolbar [net] is an example of commercial anti-phishing toolbar providing a risk rating for a visited web page in the form of an easily understandable colour code (green = safe, red = unsafe). SpoofGuard security toolbar [CLTM04] provides the same kind of information with colour code. In addition, the Netcraft toolbar displays the time since the web page is monitored, a popularity rank for the web page and the country where the web site is hosted. Figure 1.3 shows the Netcraft toolbar with the website information it displays.

Several techniques are used to infer the likelihood that a web page is a phish and display this information. SpoofGuard security toolbar [CLTM04] is another web browser plugin relying on stateless web page evaluation to provide a spoof index of a website to users. Trustbar [HJ08] provides some TLS certificate derived indicators and others users customized indicators to depict the likelihood that a website is a phish or not. A user study shows that indicators provided by Trustbar are more efficient in raising user awareness about danger than basic browsers security indicators such as padlocks or HTTPS indicators. Gastellier *et al.* propose another security tool-

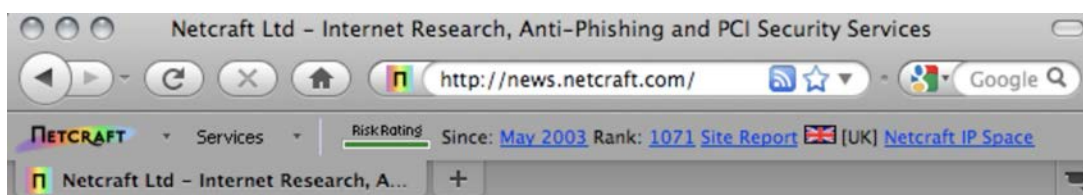


Figure 1.3: Example of security toolbar (Netcraft toolbar)

bar depicting a phishing score for a website that is computed based on 20 heuristics [GPGL11]. It additionally explicitly displays if the website is phishing or legitimate and gives the country hosting the website as well as the IP address of the server.

A slightly different approach is proposed in [WML06] with Web Wallet, which is a browser side bar, to prevent phishing attacks and password stealing. Users can enter their sensitive information in the side bar instead of in the web page directly. The application then checks if the web site is legitimate and if it can receive the sensitive information. If it is the case, the data is filled in the web page form automatically by the Web Wallet. Otherwise a warning is raised to users and the information is not transmitted. This application significantly decreases the spoof rate from 63% to 7%.

A limitation of these techniques is that users do not use the toolbars after installing them and especially for Web Wallet where a user study shows that people tend to type password directly in the web page rather than in the dedicated window. In addition the Web Wallet window can be spoofed and many users fell for this trick. Some studies [WMG06, ECH08] show that the global efficiency of phishing toolbar is bad since even if simplified information is provided, users do not understand phishing attacks and the displayed information. Moreover, it is highlighted in [ECH08] that passive phishing warning implemented in security toolbars are mostly ignored by people while active warnings blocking access are more considered and lead to prevent almost 80% of phishing attacks. Finally, the methods implemented in these toolbars for detecting phishing websites are not efficient since a study of ten popular toolbars [ZECH07] shows that the best performer reaches only 90% identification while having a false positive rate of 42%. This causes a lot of false warnings for legitimate websites explaining why phishing toolbars warnings are often ignored.

1.2.3 Blacklists

Due to low deployment and low usage of methods to enforce web server authentication and to the relative inefficacy of phishing toolbars to detect phishing sites with their high false positives rate, blacklists of phishing websites have been built. These blacklists are composed of the URLs and domain names pointing to phishing websites and can be used for several purposes such as filtering phishing emails or preventing connections to phishing websites. Usually a blacklist is manually composed based on crowd verification on community websites such as PhishTank [phi]. Suspicious URLs are submitted by users to the blacklist websites and checked by several voluntary users before being confirmed as phishes and added to the blacklist that is made freely available. This is the most widespread phishing detection technique and it is implemented in several browsers through Google Safe Browsing [goob] or Microsoft Smart Screen [mic] for instance. These two solutions have been proved quite efficient in identification of phishes, detecting almost 90% of the phishing URLs submitted to in a study [LMKK07].

Due to this implementation based on crowd verification, the strength of phishing blacklists is

its expected close to zero rate of legitimate websites identified as phishing. However, an admitted limitation of phishing blacklists is their delay to include phishing URLs in the list. It is shown in [SCH⁺09] that blacklists are not efficient during the early life of phishing websites. Blacklists only detect 20% of newly appeared phishing websites while this rate reaches between 47% and 83% after 12 hours. Hence, Liu *et al.* proposed a technique to improve human verification of blacklist entries [LXP⁺11]. Computational techniques including a weight voting metric and a technique to cluster similar phishes together before being presented to users are used. An evaluation of this technique shows that no legitimate websites were added to the blacklist and that the median time for adding a URL to the blacklist is set down to 0.7 hour. Another solution [LLCT14] proposes to automatically update blacklists based on a set of already known phishing URLs. The initial set of URLs is used as seeds to extend the blacklist by tracking redirection of these links to other URLs and extend the blacklist.

Similar methods to build predictive blacklists of potentially malicious URLs based on already known phishing URLs were introduced in [XYA⁺08, PKKG10]. In [XYA⁺08], Xie *et al.* generate signature for malicious URLs using regular expressions. URLs related to the same spam or phishing campaign are grouped together to further extract a campaign specific URL signature being a regular expression. Phishnet [PKKG10] similarly clusters URLs based on their shared common domain names, IP addresses or directory structures, and extract a regular expression from each cluster. The variable part of regular expressions is exploited to generate new URLs instead of only comparing suspicious URLs to extracted patterns as it is done in [XYA⁺08]. Though this method is more proactive than the previous one since it generates new URLs likely to be involved in a phishing campaign, both are only able to discover malicious URLs related to already blacklisted URLs that are likely to belong to the same phishing attack. These techniques are not able to disclose URLs of new phishing campaigns, which limits their efficacy.

Some network specific blacklists have been proposed [ZPU08, SLM10] to provide a tailor made protection to specific network according to their activities. A centralized log sharing architecture is used in [ZPU08], to gather and process alert logs of contributors. The relevancy of each recorded attack is ranked according to every blacklist consumer to evaluate how related an attack is to a victim. A page rank algorithm is used to estimate if a domain is likely to conduct an intrusion against a particular network and thus includes it in the specific blacklist. The produced network specific blacklists provide high attacker hit rate and good new attacker prediction.

Based on crowd verification, blacklists should overcome the weaknesses of automatic identification techniques namely the high false positive rate as observed for security toolbars, while introducing a delay for identifying a URL as phish. However, the evaluation of some blacklists [SBJ08] shows that these still wrongly identify a large count of legitimate URLs as phishes while missing a significant part of phishing URLs. The latter is expected since phishes must be reported before being added to a blacklist and it is expected that users making reports miss some phishing websites. To cope with the delay for adding entries in blacklist and to cover a larger range of phishing attacks, automatic phishing identification techniques have been developed.

1.3 Phishing Detection Techniques

Phishing prevention methods consisting in strengthening authentication are conceptually good but not used by phishing victims. The same statement holds for security toolbars providing useful information that is not considered by web surfers. Phishing blacklists are rather efficient to identify phishes, although their scope is limited to their content and their content is not up-to-

date. To cope with these limitations, phishing detection techniques being able to identify in real time submitted phishes have been introduced. These analyse different features to give a decision about the likelihood that a website, a mail or a link is a phish. Phishing detection techniques target the main phishing vectors and we will present the three categories of phishing detection techniques namely phishing emails detection, phishing websites detection based on web page content analysis and phishing URLs detection.

1.3.1 Phishing Emails Detection

Emails were the primary used vector of phishing, phishers used socially engineered text to convince people to do actions such as replying to emails and giving some personal information, going to a website or installing malware. Emails are often the cause of the visit of a phishing website using a link directing to it, because phishing websites are not likely to be indexed in search engine and thus can difficultly be found this way. Hence, the detection of phishing emails can prevent many nefarious consequences of phishing. Techniques have been developed to identify phishing emails as soon as these are received by a mail server and automatically classify them as spam before these reach users mailboxes, protecting users from this threat. However, some early phishing emails protection methods tried to educate users in order to identify by themselves phishing emails. Kumaraguru *et al.* introduced a people training method embedded in email client to help detecting phishing emails [KRA⁺07]. Fake phishing emails are intently and regularly sent by the application to users in order to test them. If users fall for the trick of these fake phishing emails, they are warned that this was a phishing email in order to raise their awareness. This technique is more efficient than regular passive training techniques or email security notices to raise people awareness about phishing emails, according to the authors.

Automated techniques were proposed as well to detect phishing emails and filter them. A popular approach relying on machine learning is proposed in [FST07]. Several features are extracted from emails such as predominant characteristics present in phishing emails while these are not in legitimate emails. Some of these features are the presence of "Here" links in emails that are used to misdirect users to fake websites. Similarly, they look for HTML encoded emails, the presence of embedded non-matching URLs *i.e.* URLs written in the email that are linked to a link that does not match the displayed URL. They search for the presence of javascript, the number of links in the email, the number of domain names, the presence of IP based URL and other URL based features, etc. This technique presents a detection rate of 96% and a global accuracy of 99.5% in the correct identification of emails. The same features set is used in [BCP⁺08] and complemented by latent topic model features. Latent topic model features can be seen as a cluster of words that appear together in the same email and the authors expect that words used together in phishing emails are different than those used in legitimate emails. Bergholz *et al.* use dynamic Markov chains to build two models, one of phishing emails and another of legitimate emails to further identify unlabelled emails as belonging to one of the two categories. This approach significantly improves the results obtained in [FST07]. Cook *et al.* [CGD08] introduce 11 rules that check the content of emails and mainly the embedded URLs to identify phishes. A clustering technique using 47 features extracted from emails is used to build several email profiles in [HA13]. These profiles include phishing emails, and models are generated from clusters to identify the profile of unknown emails.

The textual content of phishing emails can also be analysed to identify phishes since phishers put a large effort in the writing of their socially engineered messages [HCNK⁺14]. Natural Language Processing techniques and machine learning techniques are used in [RW12] to detect phishing emails. It uses Probabilistic Latent Semantic Analysis (PLSA) on the text of an email

to build a topic model of it. All words and labels present in an email are extracted to build the term-document-frequency (TDF) corresponding to the email. Then PLSA is used to match the TDF matrix of terms representing the email to a smaller matrix representing the topic of the email. Based on the extracted topic, they can conclude if the email is a phishing email or not using a machine learning algorithm (AdaBoost) fed with the features extracted using PLSA. One similar recent approach [AKS14] specifically targets the identification of phishing emails that neither deliver any malicious content nor contain any link but try to extract information with email replies. Aggarwal *et al.* perform a textual analysis of these emails to detect the mention of victim's name, monetary incentive and reply requests using natural language processing methods.

As observed for the presented methods, phishing email detection covers only one vector of phishing: emails, and victims can be directed to phishing websites with other means like links present in legitimate websites for instance. Moreover, some of the presented detection techniques only cover a subpart of phishing emails *e.g.* email embedding links or email asking for replies. This makes these approaches very specific and not able to deal with the variety that presents phishing attacks. Hence, other propositions to complement these approaches were made in order to detect directly phishing websites.

1.3.2 Web Page Content Analysis

Phishing websites usually try to adopt the look and feel of the popular websites they spoof. Hence, they often exhibit visual similarities with other websites or at least exhibit specific characteristics that researchers tried to identify. Web page content analysis was used to perform this task and identify phishing websites on the fly.

Some techniques focused on the analysis of visual similarities between phishing websites and the legitimate websites being spoofed [MKK08, CDM10, CSDM14]. Medvet *et al.* [MKK08] propose a technique to extract a signature of a web page depicting its visual composition. The signature considers the text contained in the page, its style, color, font family and size as presented in the leaf text nodes of the HTML Document Object Model (DOM) tree. It considers as well the images embedded in the page, their position, size and color histograms. The last component of the signature corresponds to the global viewport of the web page rendered by the user agent. Signatures of legitimate and phishing web pages are compared and if the similarity is too high, an unknown web page is considered as phishing. The idea of visual similarity comparison is good for spoofing website detection, although the authors do not propose any solution to previously identify the potentially spoofed website in order to find the comparison basis for the presumed phishing web page. Another work [CSDM14] relies on the same basis of visual similarity comparison to provide a targeted protection for a limited number of websites. Every original web page to protect from phishing is cached in the system and when an unknown web page is requested by the user, the rendered image is captured as an image file. The Normalized Compression Distance (NCD [CV05]) is computed between the cached web page images and the requested web page. A set of features is obtained and submitted to a classification algorithm to determine if the web page is a phishing page. This technique addresses the weakness of [MKK08] since it proposes a set of web pages to compare suspected phishes to. However, this shows that visual similarity analysis of web pages is limited to the protection of a reduced set of predefined websites.

More general techniques of web page content analysis have been proposed and the most famous example is CANTINA [ZHC07]. CANTINA extracts the terms included in a web page and apply an information retrieval algorithm (TF-IDF [SM83]) to generate a lexical signature of the web page. The signature consists in five terms that provide the fingerprint of a web page.

These five terms are later fed into a search engine such as Google to look whether the web page from which the lexical signature is extracted appear in the top n search results. If it does not, it is considered as a phishing page. It assumes that search engines index the majority of legitimate websites and that legitimate sites are ranked higher than phishing ones. This system present 97% accuracy in correct identification and 6% of legitimate web pages are identified as phishing. This approach was extended with two modules in [XHRC11] to improve its efficiency. One module considers that several ready-to-use phishing kits are used to generate phishing web pages. Suspicious web pages are compared to web pages from known phishing attacks and if there is enough similarity between them, it is considered as phish. The second module detects login forms in web pages by analysing the HTML Document Object Model and discards from analysis pages that do not contain login forms since phishing attacks mostly contain these in the aim to steal credentials from users. Similar web page content analysis techniques use different criterias to identify phishing pages. Xiang and Hong [XH09] use information retrieval techniques to identify inconsistencies between the identity a phishing web page has, and the identity of the web page it tries to imitate. A multi-criteria approach [MTM14] identifies the use of javascript, pop-up windows, the web traffic generated by a website, etc. From these observations, it creates features that are fed to an artificial neural network that adapts to phishing characteristics evolution and provides an adaptive strategy for phishing web pages detection.

Finally, a dedicated phishing detection technique [DRNDJ13] targets the detection of tabnabbing by detecting changes in the tabs of a browser window and highlights web page zones that changed while the user was inactive.

Globally, visual similarity analysis [MKK08, CDM10, CSDM14] has a limited action scope since it needs a reference web page to be applied. However, CANTINA and similar techniques have been proved quite efficient in identifying phishing web pages reaching usually more than 90% accuracy and do not need any reference for comparison. An undisclosed parameter of these techniques is the time these take to analyse a web page and render the decision. Since these are real time identification techniques, these must provide fast analysis to not impact the web surfing of users. In addition, these web page content based analysis methods are limited to the identification of phishing websites. Hence some more general techniques that can be applied to phishing emails, web page or drive-by download have been developed to identify phishing URLs.

1.3.3 URL Analysis

Several phishing attacks leverage links to misdirect users to websites or drive-by downloads. These links are Uniform Resource Locators (URLs) that are embedded in emails or legitimate web pages usually. Since URLs bring information to users about what resource one tries to consult, phishers use obfuscation techniques to make phishing URLs look trustworthy. URLs typically contain DNS information *i.e.* the domain name and hostname of the server, the path of directories and files indicating the location of the consulted resource on the server and some PHP information as well. Most users actually read URLs but are not knowledgeable about the information they contain or the DNS hierarchy. Phishers use this lack of knowledge to mix terms of legitimate URLs in order to create phishing URLs and fool users. Since URLs are used in several phishing vectors, their identification in real time can mitigate numerous phishing attacks. This real time identification is as well an alternative to phishing URL blacklists that does not require any update delay.

Using the characteristics that phishers use some tricks to obfuscate URLs, several solutions propose to perform a lexical analysis of URLs to identify phishes [BWSW10, LMF11, KIJ11, BSHA14]. A study on the anatomy of phishing URLs regarding characters frequency [MG08]

shows that phishing domains use few vowels and few different characters. Some additional observations are that long URLs with short domain names is a characteristic of phishing and phishing domain names often embed the targeted brand. Moreover phishers often use URL shortening services. Hence, several techniques [BWSW10, LMF11, KIJ11] rely on the extraction of lexical features from URLs. These features are for instance the different tokens that compose a URL as well as their relative position in the URL, the count of level domains, the length in characters of the URL, the presence of some keywords. These features are then subjected to either batch (*e.g.* SVM, Naive Bayes, Logistic Regression) or on-line machine learning techniques (*e.g.* Adaptive Regularization Of Weight: AROW) to build a model of phishing and legitimate URLs that is further used to classify unknown URLs. The advantage of on-line or semi-supervised machine learning techniques over batch machine learning techniques is that these can learn from mislabelled data or noisy dataset and keep a high accuracy ($> 90\%$). One strength of URL lexical features is that their extraction does not impact the page loading latency [LMF11] while some techniques including additional features can impact this latency.

More elaborated techniques [GPCR07, MSSV09a, MSSV09b, GSMyG⁺11, FM14] use additional information that can be extracted from URLs such as host based features. This is for instance WHOIS information for the domain name, the IP prefix of the matching IP address of the domain name or the Autonomous System (AS) number [MSSV09b]. Other techniques [GPCR07] use the page rank of a given web page as given by Google and classify URLs according to several obfuscation categories. Extracted features are used, as previously, to feed machine learning algorithms and learn a model of phishing URLs. Some systems present very good results with identification of almost 99% of URLs correctly [MSSV09b]. To cope with scalability issues of dealing with high dimensional feature vectors as presented in [MSSV09a], Feroz and Mengel propose to use distributed machine learning solutions with Mahoot [FM14].

Protection methods	<i>Fake websites</i>	<i>Low-cost product websites</i>	<i>Tabnabbing</i>	<i>Content injection</i>	<i>Spoofed browser windows</i>	<i>Web proxy</i>	<i>Instant messaging</i>	<i>Emails</i>
Authentication schemes	✓		✓		✓			
Security toolbars	✓	✓	✓					
Blacklists	✓	✓	✓			✓	✓	✓
Emails detection								✓
Web page content analysis	✓		✓	✓				
URL analysis	✓	✓	✓			✓	✓	✓

Table 1.1: Phishing vectors targeted by phishing protection methods

Finally, some hybrid techniques [WRN10, TGM⁺11] combining URL based features and web page content analysis have been introduced. Monarch [TGM⁺11] is a real-time URL filter leveraging lexical properties of URLs and page content properties such as dynamically loaded content or javascript events. This technique present an average analysis time of 5.54 seconds per

URL and an accuracy of 91%.

As presented, URL analysis provide high phishing identification rate with most techniques having over 90% accuracy. In addition, this technique mostly requires few information since only URLs and no additional content is needed for most techniques [BWSW10, LMF11, KIJ11]. The analysis of URLs can also be applied to several phishing attacks including emails, fake websites, malware delivery, etc. One issue of this technique is however that most work do not indicate the time taken to analyse URLs and the only one providing this information [TGM⁺11] present high latency over five seconds. This delay can be problematic if it has to be used in real-time while users are surfing.

To summarize the different phishing protection approaches presented in Section 1.2 and 1.3, Table 1.1 presents which solutions can cope with the phishing vectors presented in Section 1.1.2. Malware-based phishing attacks are not considered since none of the presented methods can cope with this threat and as said before, this relies more on malware detection. Vishing and SMishing are not considered for the same former reason. Table 1.1 depicts the coverage of each solution and we can see that blacklists and URL analysis are the solutions having the highest coverage while phishing emails detection can only cope with one vector.

Table 1.2 evaluates each approach according to their compliance with the requirements introduced in Section 1.1.4. The six requirements are presented in different columns and four levels of compliance can be given to each;

- --: the requirement is not satisfied
- -: the requirement is satisfied in some points
- +: the requirement is almost satisfied
- ++: the requirement is fully satisfied

URL analysis is separated in two categories to differentiate approaches using only lexical features and techniques using additional features such as host-based features. We can see that URL lexical analysis is the technique satisfying the most the requirements. In addition, to cope with several phishing vectors as observed in Table 1.1, it overtops blacklists by providing faster processing and by requiring less information. It is worth noting that phishing emails detection and blacklists are currently the most used techniques while not satisfying all the requirements. These are complementary techniques offering wide coverage, being easy to use and being highly reliable.

Conclusion

Phishing is a recurrent threat that leverages many means to be perpetrated, using technical subterfuges and social engineering. Phishing is a security problem that does not target a specific breach in systems or networks but it targets the weaknesses of users, making it a difficult problem to solve. Despite its 20 years of existence and the work made to raise people awareness, it is a continuous problem that causes an increasing financial damage over the years.

Many techniques have been developed to cope with phishing during the last ten years. The first proposition consisting in strengthening the authentication between Internet users and web servers failed due to few concerns and knowledge of many users. For the same reason, security toolbars providing useful information regarding the legitimacy of a website are not adopted since

Protection methods	Speed	Coverage	Information required	Reliability	Ease of use	Actual usage
Authentication schemes	++	-	+	++	-	--
Security toolbars	+	-	-	--	+	-
Blacklists	--	++	-	++	++	++
Emails detection	+	--	+	++	++	++
Web page content analysis	-	--	+	+	+	--
URL analysis (all features)	-	++	-	+	++	+
URL lexical analysis	+	++	++	-	++	+

Table 1.2: The extent to which phishing protection methods meet requirements

these imply a supplementary task to verify the legitimacy of a website. This was expected since most current browsers already provide all the necessary information to authenticate a website in the form of the address bar, the padlock and TLS certificate indicators. Another weakness of security toolbars is their low efficacy in identifying phishing websites. Hence, from the first developed methods, the only currently used is phishing blacklists since these are based on a human verification process that avoids false alarms due to legitimate websites identified as phishing. Blacklists are however slowly updated and they miss a lot of newly appeared phishing websites or unreported websites.

Automated identification techniques do not have this problem of update delay since they identify phishes on the fly. The main solutions are applied to the identification of phishing emails, phishing websites and phishing URLs. We have seen that phishing emails and phishing websites identification is specific since many techniques target a specific kind of phishing emails or web sites visually similar to a reference website for instance. We are as well not informed of the delay each technique produces. While it is not problematic for phishing emails identification since emails are consulted a while after these are received, delay is an important parameter for the identification of web pages while surfing. Techniques for identification of phishing websites rely on many features extracted from the web page and the extraction time can be an issue. URL analysis seems to be the most promising state of the art technique since this is the one having the less limitations, especially for these relying only on lexical analysis. Lexical analysis requires only to extract labels from the URL and to analyse its composition, which does not produce any delay. Finally, since URLs are used in several phishing attacks, this protection method is the one having the widest scope, allowing to provide the best protection. One limitation is lexical analysis based methods only have lower accuracy than techniques using all the range of possible features. Moreover, the learned identification models are based on static heuristics related to the use of certain labels or finding a specific count of level domain, etc. This limits the adaptability of the approach in some contexts, as for instance the use of new labels in phishing URLs related to a newly targeted brand.

Despite having some drawbacks, the application of these techniques improved the speed to

take down phishing websites, thus limiting the economic impact of each phishing attack. However, the ease to perform phishing attacks and the raise of both criminals perpetrating phishing and potential victims, since more and more people have access to the Web, makes the financial loss due to phishing growing. Hence, there is a need to develop new solutions to detect phishes until Internet users starts to understand the danger of phishing. These detection solutions must be fast, have a wide scope, be easy to understand and to use for users, be reliable and be adaptive. Using lexical analysis methods as a basis for further improvements, especially in accuracy, seems to be the most promising approach.

Chapter 2

Domain Name System Monitoring

Contents

2.1	The Domain Name System	30
2.1.1	Organization and Implementation	30
2.1.2	DNS Usage	33
2.1.3	DNS Misuses and Security Issues	35
2.2	DNS Monitoring	39
2.2.1	DNS Monitoring Strategies	39
2.2.2	Performance Evaluation and Anomaly Detection	41
2.2.3	Malicious Activity Detection	43

Introduction

The Domain Name System (DNS) is a key component of today's Internet operation. It ensures the finding of any resource on the Internet by just knowing an easy to remember domain name. Its main function is to map and translate domain names to IP addresses and this functionality is used by almost every Internet connected device nowadays. Hence, the monitoring of DNS traffic has been early used as an indicator for evaluating Internet activities. It is used to evaluate website popularity [RMTP08, ale], quality of service [LSZ02, PAS⁺04], network performance [HHW⁺10], etc.

The access to Internet resources with a translation mechanism to IP addresses provides great benefit to legitimate Web services and enhances their availability by allowing to perform load balancing between several servers and fast recovery in case of server failure. These same assets have been also used by miscreants to maintain and develop their malicious activities. The use of specific DNS features harden the process of malicious server tracking rendering their take down difficult. Of major interest is DNS fluxing networks, which are used by organised phishers [ssa08] to hide the real location of malicious servers hosting phishing websites behind an ever-changing network of machines. These machines are usually compromised users' hosts being part of a botnet. Fluxing increases drastically the resistance of phishing to countermeasures.

DNS fluxing presents some characteristics from the Domain Name System point of view that are distinctive from a normal usage of the DNS. Hence, the monitoring of DNS information has been used recently in order to detect malicious activities. A lot of solutions were developed to propose techniques to capture, extract and analyse the relevant DNS information for

identifying malicious activities [CLLK07, APD⁺10, CTD⁺12]. Most of the work was focused on specific domain names activities identification such as domain names involved in botnets communications, fluxing domain names and the differentiation from Content Delivery Network (CDN) domain names [HGRF08, PCDL09, APD⁺10, BN12]. Many techniques, being too specific, missed the diversity of domain names activities and had weaknesses for differentiating malicious and legitimate activities. Also, most techniques of DNS monitoring for malicious activities detection only use DNS specific features that are found in DNS packets. Few researches [YRRR10, APN⁺12, BG10a] are focused on the lexical analysis of domain names and none of them considers the semantic dimension of domain names. However, domain names are composed of domains and host names that are meaningful and the analysis of this characteristic let envision new perspectives for extending the features set available to detect malicious domain names.

In this chapter, we start by describing the organization and functioning of the Domain Name System in Section 2.1. We provide a detailed presentation of DNS usages, DNS misuses to enhance malicious activities and an analysis of security issues and threats related to the DNS. We present in Section 2.2 some state of the art work on DNS traffic monitoring and analysis, give their applications and identify their shortcomings.

2.1 The Domain Name System

The Domain Name System [Moc87a, Moc87b, MD88] is a service having for main role to establish a mapping between a domain name and an IP address. More generally, the DNS provides a way to link any information to a domain name. For consistency and availability of the information the domain name system is implemented in a distributive hierarchical manner. It is used by many services to enhance the availability of resources over the Internet and is used for legitimate and malicious purposes. In this section we present the functioning of the DNS and how it is leveraged to enhance legitimate services and to strengthen malicious activities. We also present some threats specific to the DNS and security issues in its conception.

2.1.1 Organization and Implementation

The DNS is composed of three major components that present the organization of its information, the distribution and delegation for authority on the information it provides and finally the way to access this information for users. These three components are the domain name space, the name servers and the resolvers.

The Domain Name Space

The Domain Name Space is a tree structure to represent domain names and the information associated to. Each node and leaf of the tree has a label and names a set of information that can be extracted with queries. The labels of nodes are only composed of letters, digits, and hyphens and have a maximum length of 63 characters. Labels are not case-sensitive. The domain name of a node is the list of labels on the path from the node to the root of the tree. The root of the domain name space is the "." node. Figure 2.1 represents a subtree of the domain name space and the path from the node *snt* to the root is highlighted in red to deduce the domain name *snt.uni.lu*. Every node in the tree is called a level domain. Nodes at the base of the tree are called first level domains or Top Level Domains (TLD) *e.g. lu* for *snt.uni.lu*. Then, as we go down in the hierarchy, nodes are called second level domains (2LD) *e.g. uni*, third level domains (3LD) *e.g. snt* and so on. A domain name is thus a list of level domains separated

by dots. In the domain name space every node of the tree reports to its parent node, meaning that *snt.uni.lu.* reports to the domain name *uni.lu.* and we say that *snt* is a subdomain of *uni.lu.* Understanding this hierarchy and dependency between domain names is paramount to be protected from obfuscation URL techniques used by phishers and to prevent from being trapped with obfuscation lures.

The querying of DNS information is made using DNS queries that must specify the domain name of a node and the type of information requested. The type of information corresponds to a Resource Record (RR) type. Several Resource Record types exist to provide specific matching information for a domain name, some of the mostly used RRs with their associated information are:

- A records: IPv4 address
- AAAA records: IPv6 address
- CNAME records: domain name alias
- MX records: domain name of the mail server
- NS records: domain name of the authoritative name server
- TXT records: any text
- PTR records: reverse DNS record giving the domain name of an IP address represented in a domain name manner (*e.g. 78.56.4.123.in-addr.arpa.*)

To give an example of a DNS Resource Record, here is a DNS A RR with the information it contains: `snt.uni.lu. 22000 IN A 158.64.76.208`

This record contains the queried domain name (`snt.uni.lu.`), the duration of validity of the record in seconds also called Time To Live and abbreviated TTL (22000), the type of Resource Record (A) and the value of the record, for A Resource Records it is an IPv4 address (158.64.76.208).

The Name Servers

The Name Servers are the repositories of the DNS tree information. The storage of the DNS information is distributed and delegated to several servers being divided into sections called zones. There are basically two kinds of name servers being authoritative name server responsible for maintaining and providing the information for a given zone and recursive name servers responsible for resolving Fully Qualified Domain Names (FQDN) using an iterative process. A FQDN is a domain name that contains a hostname and a domain. For instance in the FQDN *snt.uni.lu*: *snt* is the hostname and *uni.lu* is the domain.

An authoritative name server delegated to a given DNS zone has authority for a complete subset of the domain name space. It is responsible to deliver the correct information for this subset *i.e.* valid and up-to-date Resource Records. There are 13 authoritative root servers responsible for the DNS root zone ("."). These root servers have the complete information about the root zone and know where to find the nodes that are directly under their authorities namely the Top Level Domains nodes. Root servers know the authoritative name servers for the several TLDs and can give the information to access them when requested. The same statement holds for accessing the information of nodes being under the TLD nodes, this information can be accessed by requesting the authoritative name server for TLDs. There is a hierarchical delegation of the

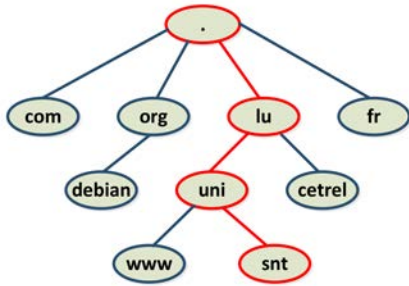


Figure 2.1: Domain Name Space hierarchy and path to the node *snt*

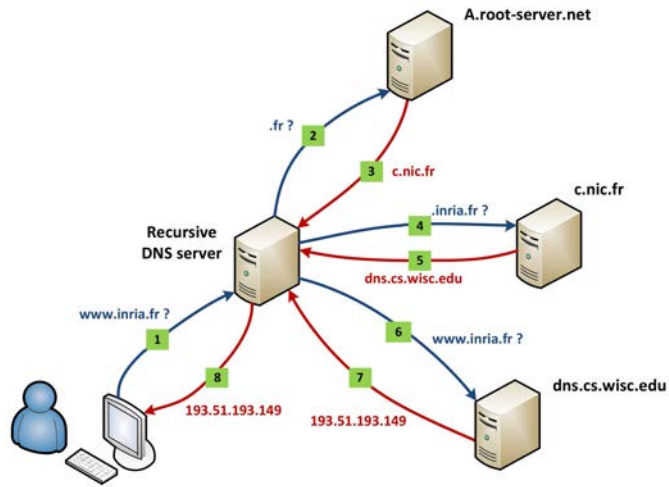


Figure 2.2: DNS resolution of the domain name *www.inria.fr*

DNS information implying a recursive process to resolve a domain name by querying the several name servers authoritative for the different zones.

The recursive name servers (RDNS) are typically not authoritative for any zone and just have a role to resolve Fully Qualified Domain Names requested by clients since these support recursion. To perform this resolution, RDNS servers maintain a cache of Resource Records for frequently requested domain names. These Resource Records have a validity duration (TTL) and can only be provided during this period. If a DNS request for a domain name is not cached or if its validity expired, the recursive DNS server must resolve the domain name with an iterative resolution process.

The Resolvers

The Resolvers are programs implemented on client machines to extract the user's requested DNS information from name servers. Resolvers are able to access at least one name server in order to answer a DNS query directly. Resolvers have usually access to recursive name servers that do the iterative resolution process for them. These recursive name servers are usually operated by Internet Service Providers (ISP) and Open DNS servers are also available. Recursive name servers operate the resolution process by querying iteratively the different level domains of a domain name to resolve it, if it is not cached. If the requested Resource Record for a domain name is cached, the recursive name server reply with this cached record. Figure 2.2 depicts a typical process of iterative domain names resolution for the domain name *www.inria.fr* assuming that all name servers have an empty cache. The user wants to access the resource located with the domain name *www.inria.fr*. Hence, the resolver present in his machine knows a DNS recursive server to which it addresses the DNS query with requested recursion for the A Resource Record of the domain name *www.inria.fr* (1). Then, assuming that the RDNS server has an empty cache, it first queries a known root server to get the name server authoritative for the domain name *fr*. *i.e.* it performs an NS DNS request. Once it gets the DNS replies (3), it contacts the authoritative name server of the domain name *fr*. to get the address of the authoritative

name server of the domain name *inria.fr*. (4). When it gets the reply (5) it can finally request the needed information by performing a A DNS request for *www.inria.fr* (6) to the appropriate authoritative server and forward the information to the client resolver (8).

This gives an overview of the design and the implementation of the Domain Name System. Besides its primary function to map domain names to information, the DNS, thanks to its robust conception, has been leveraged to enhance several Internet based services.

2.1.2 DNS Usage

Domain names are unique in the domain name space. If one wants to use a domain name and provide some information related to this node of the domain name space, one must register a free domain name to a domain name registrar, adding a new node in the tree. Registering a domain name is equivalent to owning a domain name and ensures that the registrant is the one that controls the information provided by the domain name in the DNS. Maliciously registered domain names corresponds to domain names registered to be used for malicious purposes as for phishing.

Basic Usage

Domain names provide an easy to remember way to locate resources on the Internet because the different nodes of the domain name space are labels that are usually meaningful words allowing to categorise the information. For instance the Top Level Domain *.com* is meant to be used for locating *commercial* and for-profit business resources and services, *.org* is meant to be used by non-profit *organization* and *.edu* targets *educational* institutions. The same observation holds for lower level domains where for instance *uni.lu* is the domain name of the University (*uni*) of Luxembourg (*lu*) or *www.inria.fr* is the domain name of the webserver (*www*) of the National Research Institute INRIA (*inria*) that is located in France (*fr*). Hence, besides providing a way to store and deliver the information, the DNS allows to name in a meaningful manner the information it stores and to access it using these meaningful names as unique resource.

The mostly used service in DNS is to translate domain names to IP addresses in order to physically locate a resource on a network using A RR (IPv4) or AAAA RR (IPv6). The reverse process to find the domain name corresponding to an IP address is done using PTR DNS requests to specific name servers. IP addresses must be written in reversed order in front of the domain name *in-addr.arpa* for IPv4 and *ip6.arpa* for IPv6. The DNS also gives the domain name of authoritative name servers for a given domain name with NS RR. Besides this usage, the DNS is used to locate the mail servers of a given domain name through MX resource records. Each MX RR for the same domain name has an additional feature to inform about priority of mail server usage.

By providing an abstraction to locate physical resources, the DNS offers several advantages to enhance the management of Internet resources. For instance several Resource Records of the same type can be associated to a single domain name, this is called round-robin DNS (RRDNS). This has been used to store redundant information on several servers for highly requested services. Popular Internet websites manage their domain names and provide several A Resource Records for the same domain name pointing to different IP addresses. This provides a set of web servers providing the same content and user's machine can freely choose between the several records, one web server to request the content. This strategy avoids to have a single point of failure by providing redundancy. Some services specialized in the delivery of content over the Internet

exploit even more the DNS to provide highly available resources. These are called Content Delivery Networks.

Content Delivery Network

A Content Delivery Network (CDN) [PB07] is an Internet wide distributed system of servers deployed in multiple data centres, which provides content delivery services. CDNs provides Internet contents with high availability and high performances to users by taking advantages of the abstraction of resource location provided by the DNS.

When requested for a given content with a DNS request, Content Delivery Networks deliver tailor made DNS Resource Records taking into account several parameters such as the location of the requester, the location of the data center where the content is available or the load of the servers part of the network providing the content. DNS requests to CDNs are for domain name of the form *user-content-info.cdn.com* where *user-content-info* contains some user specific and content requested coded information in order to define which servers of the CDN are the best to quickly deliver the content. The IP addresses of the selected content delivery servers are included as A RR in the DNS reply sent to the requesting user. Resource Records provided by CDNs have a low Time To Live to direct users to different servers over time and to distribute the load among the several servers. In addition the adoption of low TTL facilitates the recovery in case a delivery server is down since a new DNS request must be done when the TTL expires and a new delivery server can be selected.

From a DNS point of view, CDN domain names have some distinguishable characteristics:

- RRs with low TTL (from 30 seconds to 10 minutes on average).
- Ever changing A RR for the same domain name with IP addresses belonging to a fixed set.
- Several subdomains, often algorithmically generated, for the same domain name (corresponding to the several contents provided and user served).

Many Internet content providers like Apple or Facebook pay CDN operators to deliver their content. CDNs services provide high availability and high performances in content delivery. In addition, these reduce the cost of their clients since clients do not need to own servers to host and deliver their content themselves, thus reducing their infrastructure costs.

Other Applications

The Domain Name System is also used for other use cases than simply mapping information to domain names. It is for instance used to prevent email spoofing through the Sender Policy Framework (SPF) [Kit14, Kuc14]. SPF is an email sender validation system used to confirm that an email coming from a given domain name was actually sent by an authorized server. In SPF, a domain name uses DNS TXT RRs to publish a list of authorized hosts that can send emails with a sender address containing the given domain name. Hence, mail receivers can make a TXT DNS request for the domain name contained in the email sender address and confirm whether the email is sent by an authorized mail server to confirm its legitimacy or not. This is actually a phishing prevention technique to prevent being trapped by email spoofing.

DNS is also used as a means to distribute IP addresses and domain names blacklists [JS04]. Rather than storing a hard coded version of publicly available IP and domain blacklists, the legitimacy of both can be checked making DNS requests. The process is similar to the consultation of

PTR Resource Records. Having a domain name *blacklist.org* providing an IP or a domain blacklist, to determine dynamically the legitimacy of a domain name is done making a A DNS request for *blacklist.org* appended to the reversed IP address or domain name to check. For instance to check *12.34.56.7* and *www.phishing.com*, the following A DNS requests *7.56.34.12.blacklist.org*, and respectively, *com.phishing.www.blacklist.org* are made. The value of the A RR contained in the reply gives the status of the requested resource *e.g.* *127.0.0.1* the resource is blacklisted, *127.0.0.2* the resource is not in the blacklist.

The domain name system is used for many additional purposes beyond domain names translation and provides enhanced availability of resources as well as some security applications for preventing email spoofing or detecting malicious hosts. However, it has been also used to enhance malicious activities.

2.1.3 DNS Misuses and Security Issues

When a service provides benefits to legitimate services, the same benefits can apply to malicious activities as well. This statement holds for the DNS. While enhancing content delivery and increasing availability of legitimate contents, it is also used to ensure the availability of malicious contents on the Internet. As a popular and common Internet service and network protocol, the DNS is used to convey and hide other kind of network traffic. In addition, the old conception of this service (1987) did not consider security breaches that have been exploited for several malicious purposes.

DNS Tunnelling

The DNS is a key element of network communications that is used in a first step before initiating any other sort of communications using protocols like HTTP, SIP, SSH, etc. Due to this paramount role, the DNS protocol is one of the few protocols that can be allowed by firewalls. Hence, DNS has been used to encapsulate several kind of network traffic that are mostly filtered. For instance DNS tunnelling has been used for backdoor communications where infected machines open a backdoor on the port 53 to receive messages that are not dropped by firewall or detected by Intrusions Detection Systems (IDS). SSH traffic is usually encapsulated in the case of backdoor communications and an attacker can control the host he compromised using this covert channel.

Another widely used example is to encapsulate HTTP traffic in DNS packets. Wireless Access Points (APs) are available everywhere nowadays and credentials are required to connect to these APs. Some access points present in airports or hotels for instance are public, and allow users to freely connect. Users are then redirected to a web page where they can usually either enter credentials or pay for full Internet access. These kinds of open access point block HTTP traffic to the Internet but let DNS traffic goes through so that users can be redirected to the access point web page asking for credentials or payment. Users can exploit this free access to DNS communications to encapsulate HTTP traffic in DNS packets and have Internet access through public access points for free. Dan Kaminsky developed and proposed such a kit to create a DNS tunnel [Kam04]. Basically the only requirement is to register a domain name *example.tunnel.com* and to deploy a name server authoritative for this domain name. Any DNS requests made for the domain name *example.tunnel.com* are addressed to the controlled name server. By encapsulating HTTP GET requests in DNS TXT queries, the controlled DNS server receive them and can make the requests and return the HTTP replies content in DNS TXT replies. Hence, users can surf

the Internet through the public access point without paying any fees. HTTP GET requests are encoded in the subdomains of the queried domain names like in *http.get.example.tunnel.com*. The requested content is encoded in the response field of the TXT RR contained in the DNS reply.

DNS Fluxing

DNS fluxing [ssa08] is an activity used to enhance the availability of malicious resources and contents by hiding the real location of a given resources behind a network of ever changing machines. The hidden resource is typically a server delivering malware, a phishing website or the command and control server of a botnet (C&C).

A botnet is a network of compromised users' machines (bots) being controlled by a botmaster. The machines part of the botnet have been infected by malware letting a botmaster control them, these are called zombies. The zombies of a botnet are controlled by the botmaster through a command and control server also called a mothership. Botmasters do not know which machines are part of the botnets, and only bots know how to contact the mothership and to ask for instructions to execute. Hence, the mothership is the only means to control the bots and the takedown of it causes to loose the control of the botnet. Hence, several techniques have been developed to keep the mothership up. Similarly to protecting the mothership of a botnet, the hiding of web servers hosting phishing websites to ensure the longevity of a phishing attack is paramount. DNS fluxing is used to perform this activity and to counteract IP blacklisting and easy malicious server takedown.

DNS fluxing can be of three sorts being IP fluxing, which has two categories fast flux and double flux, and domain fluxing. Fast flux or single flux is the most common example of fluxing and the easiest to perform. Fast flux consists in frequently changing the A RR of a registered domain name in order that the domain name is always associated to different IP addresses. In double flux, there is a second level of dynamics added by changing the name servers authoritative for the domain name. The same process as for A RR is applied to NS RR and authoritative name servers for a single domain name change over time over a set of machines.

To perform IP fluxing, one needs to have a set of Internet connected hosts over control, as illustrated in Figure 2.3 for a user consulting a phishing website hosted on a double flux network. The controlled hosts usually consists in bots of a botnet (zombies in Figure 2.3), to which the phisher can associate the domain name of a phishing website in A RRs. The set of controlled hosts appearing in A RR represent the only machines that know where is actually hosted the phishing website (the mothership in Figure 2.3) and can forward the requests these receive to it, acting as proxies. Figure 2.3 show the process of a user visiting a phishing website (*double.flux.com*) registered by the phisher. Steps 1–7 depict the domain name resolution. Step 5 is characteristic to double fluxing in which some zombies act as authoritative name servers. These bots are forwarding DNS requests to their mothership. The mothership provides several A RR for *double.flux.com* that are frequently changing over time. In single flux, a single regular authoritative name server is used to provide the ever-changing A RR. Once the user got the DNS reply, it contacts a zombie home PC controlled by the phisher that redirects the packets to the real server hosting the fake website (8–10). While single fluxing hides the hosting of the phishing website, double fluxing hides as well the location of the authoritative name server providing the changing Resource Records. Both techniques are used for phishing web servers hiding and botnet command and control communication. In C&C communication the user's machine requests instructions to execute to its mothership. Rather than having an IP address to contact the mothership, it has a domain name and perform the resolution process to contact the

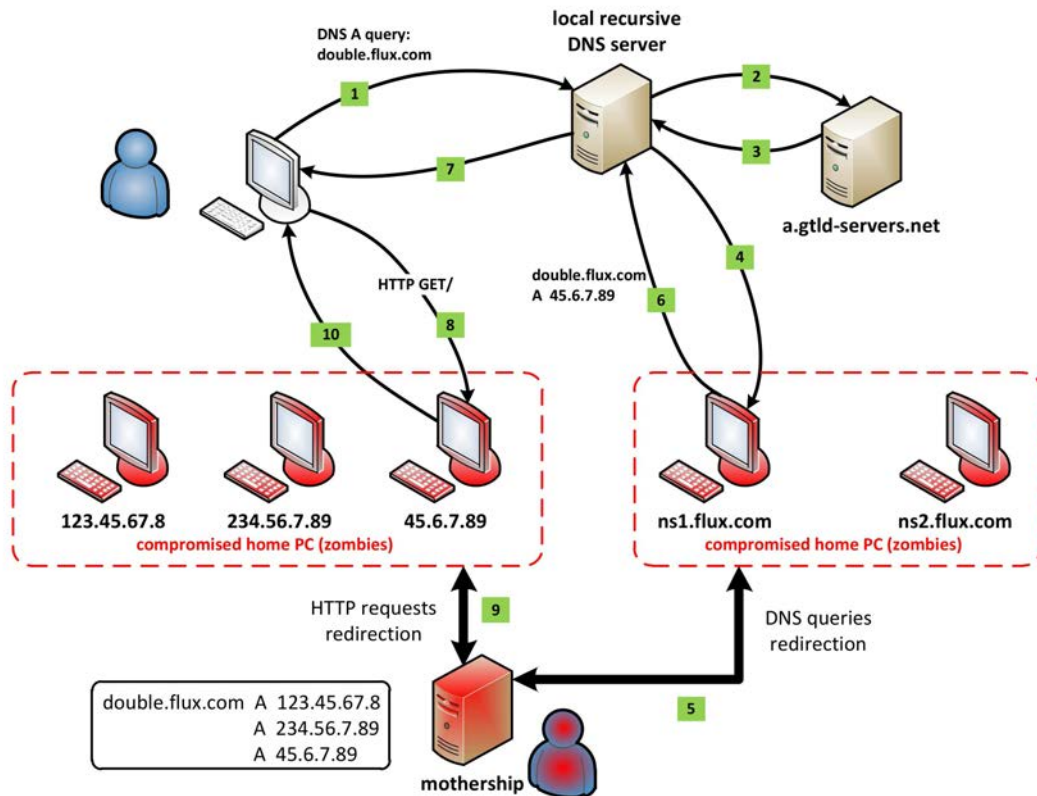


Figure 2.3: Phishing web site hosting using a double flux network

mothership through proxy hosts being some selected bots used to act this way. DNS fluxing is resistant to IP address blacklisting since the set of host serving as proxy is changing frequently. Hence, processes for domain names blacklisting have been set up to counteract IP fluxing and block the resolution of domain names used for IP fluxing also called fluxing domain names.

To cope with this countermeasure, domain fluxing is used, essentially for botnet command and control communications, to avoid domain names blacklisting. Bots of the botnet have a generation algorithm that generates several domain names every day [Ayc12] and only few are actually registered by the botmaster and used for C&C. These domain names change often over time and slow domain name blacklisting is inefficient to prevent malicious connections.

From a DNS point of view, IP fluxing domain names have characteristics [HGRF08] close to CDN domain names but with some differences:

- RRs with low TTL (from 30 seconds to 10 minutes on average).
- Ever-changing A RR for the same domain name with IP addresses belonging to an evolving set.
- Ever-changing NS RR for the same domain name with IP addresses belonging to an evolving set (only for double flux).
- IP addresses are scattered across several networks

Security Issues

The DNS is exposed to some security issues because it was designed to be scalable and distributed with no security concerns. Its conception mainly lacks a technique for authenticating Resource Records provided by name servers to resolvers. This led to develop attacks to provide fake Resource Records to DNS users by changing the Resource Records for a given domain name. The main application is to modify A Resource Records in order to associate the IP address of a controlled server with a domain name that an attacker did not register. This technique is widely used in phishing to direct users to phishing websites. The modification of delivered resource records can be done using several techniques [AA04]. Intercepting packets with a man-in-the-middle attack, an attacker being on the shared network can deliver fake DNS replies to a given DNS query made by a resolver. If the attacker is on another network he can perform DNS query prediction combined with DNS packet ID guessing to deliver a forged DNS reply to a resolver. Name chaining attack consists in including additional Resource Record information, that is not related to the initial query, in DNS packets. This information will though be considered by the victim receiving the reply and further used to replace existing stored records.

These techniques are used in DNS cache poisoning attacks also called DNS spoofing. DNS cache poisoning [TVBP10] consists in injecting fake Resource Records in the cache of DNS resolvers and name servers serving several users. By introducing a record mapping a targeted domain name to an IP address leading to a controlled server, the traffic is diverted to this server. Internet users being served by the poisoned DNS recursive server and requesting the changed Resource Records are directed to the attacker's server when querying a spoofed domain name. This attack is used in phishing to direct users to fake websites while using a valid domain name.

To cope with authentication issues in DNS, DNSSEC [AAL⁺05] an extension to DNS, has been proposed. DNSSEC adds a means to sign and authenticate the Resource Records included in DNS replies using public key cryptography schemes and a trusted third party. By its conception DNNSEC copes with the above mentioned issues used for DNS cache poisoning. However DNSSEC is not widely adopted yet and it does not solve all security problems and let the DNS vulnerable to domain hijacking for instance. Domain hijacking is the stealing of a domain name by changing its registration information with the domain registrar. Domain hijackers usually impersonate the original domain owner with stolen personal information and ask the registrar either to change the registration information or to transfer the domain name to another registrar. The hijacked domain names then belong to the attacker who can map any information to this domain name, having the same nefarious consequences as for DNS cache poisoning. Another flaw introduced by DNSSEC is to ease zone enumeration [AAL⁺05]. Zone enumeration consists in the process of discovering all the names of a zone, which is used by attackers to map network hosts of a given zone. Moreover, DNNSEC does not prevent DNS denial of service (DoS) attacks and can even enhance DoS [vRDSP14].

Reflexive Denial of Service (RDoS) over DNS is a widely used technique to deny the service of a targeted DNS server. It consists in using bots sending several DNS queries, with a spoofed source IP address being the one of the targeted DNS server, to open recursive DNS server. The requests are made for a domain name registered by the attacker and large amount of data is associated with this domain names in Resource Records. Open recursive servers reply to the targeted name server with large DNS replies making the server overwhelmed with the quantity of data it has to treat such that it is not able to treat legitimate requests.

The DNS has an essential role in today's network communications that exceeds the expected usage planned at the time of its conception. Its role of domain names to IP addresses mapping is

paramount but its applications to powerful content delivery or blacklist distribution are necessary as well. Its wide adoption by Internet miscreants to enhance their attacks and the misuse of its protocol for hidden communications render its study worth for detecting malicious activities. Hence, several methods have been proposed to monitor the DNS traffic for several applications related to its usage.

2.2 DNS Monitoring

As a core service of the Internet that is widely used by Internet users, the DNS traffic has been monitored and analysed for several purposes. The first intend was the evaluation of the quality of services provided by the DNS and the detection of anomalies in the generated traffic [WFBc04, CWFC08, ADL⁺10]. Then, as a widely used service it has been used for additional purposes like network performance evaluation [HHW⁺10] or lightweight websites popularity ranking [RMTP08]. As the use of DNS and implication in malicious activities increased, it has finally been used to study and detect malicious activities in order to counteract them. We start by presenting the several strategies of DNS monitoring and then present the applications for performance evaluation and anomaly detection. Finally we present state of the art techniques of DNS monitoring applied to malicious activities detection and more precisely to detect phishing hosting through malicious infrastructure and phishing domain names.

2.2.1 DNS Monitoring Strategies

Two kinds of DNS monitoring can be performed. The first type is active monitoring that has been mainly used for performance evaluation since one knows what he wants to evaluate using DNS. Active DNS probing can only be targeted and it consists in requesting a specific record to a specific name server. Some applications are finding the delay of response between two given machines [HHW⁺10] or the evaluation of the DNS resolution process from a given location [LSZ02]. Active probing is also used for DNS specific threat detection like cache poisoning [ADL⁺10].

However, the most widespread technique relies on passive DNS monitoring. Passive DNS monitoring is rather used for anomaly detection in DNS traffic and misconfiguration disclosure [WSS11, CWFC08], since this kind of behaviour is unexpected. It is also widely used in malicious activities detection like malicious domain names detection [APN⁺12, YRRR12], botnets detection [VSB09, CL12] or flux networks detection [HGRF08, PCDL09]. Finding malicious domain names, botnets or fluxing networks is not targeted. This detection operates in the wild without previous knowledge about the malicious infrastructures to discover. The way to detect them is to perform a large scale monitoring of the DNS traffic in order to intercept communications from infected bots or phished users to flux networks. Another important point for malicious activity detection is that passive monitoring does not allow an attacker to detect the monitoring since he is not probed. Active probing from a single machine is detectable by an attacker and allow him to take countermeasures to avoid the disclosure of his activity.

Passive DNS monitoring can be performed at different levels of the DNS hierarchy and we present them in Figure 2.4. This figure presents four hierarchical levels of name servers with from top to bottom, one root name server (RootNS), two TLD name servers (TLDNS), some authoritative name servers for two-level-domains (AuthNS) and some recursive DNS servers (RDNS) serving different network of machines. Five different pasive DNS monitoring locations are presented in red and numbered (1–5):

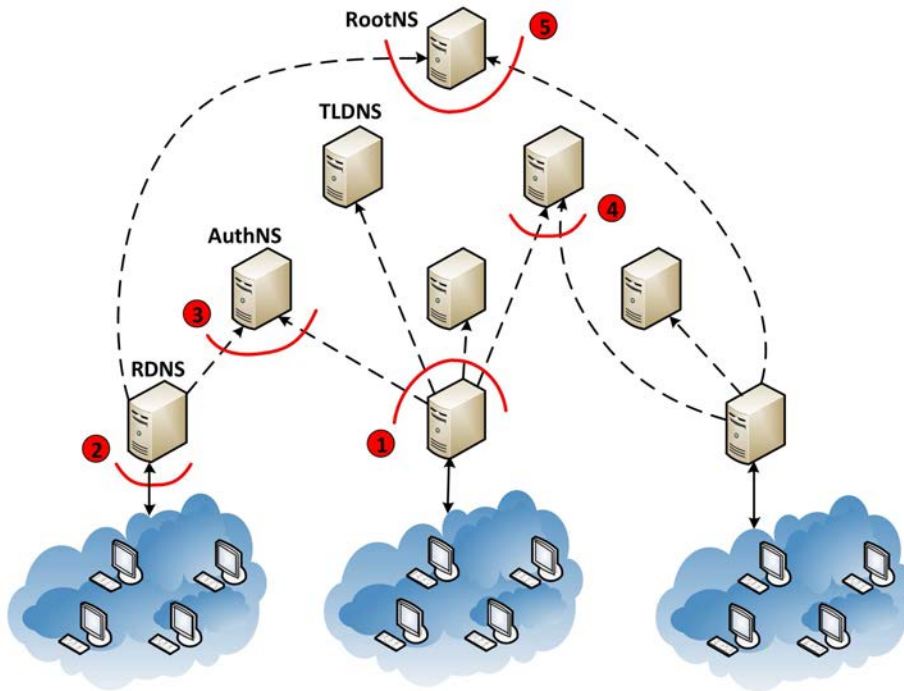


Figure 2.4: Different DNS probing locations

1. Recursive DNS servers level: This technique allows to gather all requested domain names from a given set of served hosts without redundancy of DNS requests/replies. This technique is interesting for dedicated network protection for instance. In [Wei05], Weimer introduces this technique called *passive DNS replication* which consists in the capture of DNS packets at the recursive DNS server (RDNS) level, between the recursive server and authoritative name servers. He shows the advantages of this technique compared to other DNS monitoring techniques in term of keeping user's privacy and avoiding redundancy of data. Weimer describes an architecture that implements such DNS monitoring technique and presents several application for the gathered data such as malware containment, phishing prevention, botnet mitigation, etc. A community project led by the ISC Security Exchange Information [isc] propose the sharing of passively gathered RDNS data through their pDNS project. Contributors provide records from their monitored RDNS servers and have access to a worldwide database of passive DNS records. This database has been used in several work related to malicious activity detection [APD⁺10, BKKB11].
2. Below RDNS servers: A monitoring technique that provides a fine grained view of individual users' querying behaviour. This kind of monitoring has applications for case specific anomaly detection such as finding misconfigured hosts or botnet compromised hosts [CLK07]. This technique as though some law-compliance issues since this does not preserve users anonymity as the latter does.
3. Authoritative name servers level: This monitors DNS packets between a given name server authoritative for some domain names and the recursive DNS servers. This technique preserves anonymity and is applied to monitor the requesting behaviour of users for a given

domain name. This has applications to detect the hosting of malicious domain names by DNS operators [APL⁺11] or the evaluation of CDN quality of service [PAS⁺04] for instance.

4. Top level domain authoritative name servers: This monitors DNS packets between a given name server authoritative for a TLD and the recursive DNS servers. An architecture for this DNS monitoring technique is introduced in [DTMV12] and applied to monitor the *.it* ccTLD. The captured traffic is minded to analyse the trends, the evolution and interests of Internet user of a country. The technique has lower efficiency than HTTP traffic monitoring but DNS traffic monitoring is law-compliant. Some given applications are name server scanning detection, identifying sources of spam by identifying DNS requests for non-existing domain names and DNS traffic statistics [XXXB09].
5. Root name servers level: This monitors DNS packets between a given root name server and the recursive DNS servers. The applications of this technique is quite limited and relates to quantity and quality of DNS traffic evaluation [CWFC08]. The study leads to find global misconfiguration in the implementation of different DNS server distributions [BCN01] for instance.

The different passive and active DNS monitoring techniques have several purposes for security monitoring and network management. Some of these are specifically applied to the DNS but have as well other Internet-related applications.

2.2.2 Performance Evaluation and Anomaly Detection

The main effort put in DNS monitoring at the beginning was to analyse DNS performances, misconfigurations and detection of threats to the service. We describe some of these applications and give more general approaches with the kind of monitoring these rely on.

DNS Specific Applications

Some of the early studies of DNS traffic targeted the root name server level [DOK92, BCN01, CWFC08] to disclose misconfiguration in the DNS. Danzig *et al.* [DOK92] focused the analysis on the pollution on wide-area network traffic due to errors in DNS server implementation (caching, retransmission and selection of alternative servers). They conclude that DNS consumes 20 times more wide-area bandwidth than necessary due to misconfiguration using root server level DNS monitoring. Castro *et al.* [CWFC08] draw the same conclusion showing that invalid DNS traffic represent around 98% of the total traffic. Their analysis consisted of the largest scale simultaneous collection of passive DNS traffic ever gathered: one day of DNS traffic from eight root servers. They analyse both the evolution of DNS traffic in quantity and requests count per client. Broido *et al.* [BNC03] study the DNS traffic generated by DNS updates and disclose misconfiguration in Microsoft name servers sending on the Internet updates for private IP addresses.

DNS caching strategies play a major role in order to reduce the unnecessary DNS traffic as well as the load of name servers. In [WFBc04], a passive monitoring at the recursive DNS servers level for queries targeting root and TLD name servers is performed. This studies the effects of DNS caching strategies on the queries made for upper level servers to determine the overload generated by several DNS server implementation such as BIND, DJBDNS or Windows. This study leads to identify the strengths and weaknesses of each server implementation in distributing the query load to upper level DNS servers. With a similar monitoring technique, Jung *et al.* [JSBM02] study the impact of TTL values and cache sharing on DNS caching efficiency.

The active probing of DNS server is performed in [LSZ02] to compare the performance of domain name resolution from different locations in the Internet for a set of domain names. The probing shows consistency of the provided information but a response time that varies a lot depending on locations of the requesters. A proposition to equitably choose name servers placement improves the response time.

Finally, DNS specific applications have been made to detect cache poisoning and DDoS attacks with DNS queries visualization tool [RKG06]. Automated techniques that do not require human interventions are introduced in [WHLY06, ADL⁺10]. For instance, Antonakakis *et al.* [ADL⁺10] introduce Anax, a large-scale, scalable centralized DNS protection system that detects poisoned records in the cache of name servers. The technique relies on the active probing of 300,000 open DNS recursive servers making different queries for a set of domain names that are likely to be targeted by cache poisoning. It extracts a set of features from the DNS responses that are subjected to a machine learning algorithm. The system identifies change in cached DNS records and deduces cache poisoning mainly through the change in IP addresses associated to a given domain name.

Network Traffic Measurement

Some other applications of DNS monitoring are dedicated to the evaluation of performances of some services relying on the DNS like CDNs or blacklists delivery services that were presented in Section 2.1.2. For instance, a study of DNS logs from Akamai name servers in [PAS⁺04] show some limitations of DNS-based network traffic management as it is used in CDNs. The DNS does not offer quick enough response to deal efficiently with link failures and performance degradation according to the study. However, Huang *et al.* [HHW⁺10] propose an efficient method to measure network performances using DNS queries. A DNS reflection method is used to compute the distance between two given machines and more precisely between an end-user and a data center providing a given content. This technique is used for global traffic management to select the closest CDN server to deliver a content. With the increasing use of CDNs to deliver contents, it is difficult to distinguish between content owners and content hosters. The passive monitoring of DNS traffic [BMM⁺12] allows to tag traffic flows with their associated domain names in order to understand who is the owner and the hoster of contents available in the Internet. Hence, one can deduce which CDNs deliver a given resource or identify the different contents delivered by a given CDN. The monitoring of DNS traffic can evaluate as well which service generate which proportion of the traffic. For instance, the part of DNS traffic attributed to CDN activities can be evaluated. The proportion of DNS traffic overload attributed to the consultation of DNS blacklists (DNSBL) is evaluated [JS04] by monitoring DNS requests sent from a given monitored network.

DNS monitoring is also used for evaluating the popularity of websites or events. For instance, by monitoring TLD name servers, Wang *et al.* [WSS11] detect global anomaly in network traffic that are characteristics to some news events. Their anomaly detection scheme relies on a clustering analysis of the covariance change of DNS query traffic from different TLD servers of the same TLD. The assessment of the technique shows it is able to detect a specific event happening in China (5.19 events) by monitoring DNS traffic to the *.cn* TLD name servers. Rajab *et al.* [RMTP08] use DNS cache probing to infer the density of clients accessing a given network service. Studying the evolution of cached Resource Records for a given domain name in local recursive DNS servers leads to derive an estimation of the number of clients accessing this service. DNS cache probing can thus estimates the popularity rank of websites with a less invasive method than the popular Alexa service [ale]. Another application of this technique is the estimation of

botnet infected hosts.

2.2.3 Malicious Activity Detection

Besides the evaluation of network performances, quality of DNS service or cache poisoning detection, DNS monitoring has been used to detect malicious activities leveraging this service. The main effort has been put in tailoring methods to detect malicious flux network and botnets related activities. Some other work however targeted the identification of general purpose malicious domain names relying either on DNS specific information or on lexical analysis of domain names.

Fluxing Networks and Botnets

The most spread malicious activity relying on DNS is flux networks since these networks are the support of many malicious activities including phishing. Holz *et al.* [HGRF08] were the first to operate passive DNS monitoring in order to identify the characteristics of domain names used in flux networks. They defined several heuristics that we presented in Section 2.1.3, in order to differentiate fluxing domain names from legitimate domains. The conclusion is that dynamic in DNS is often associated with malicious activities. However, a contradictory study [BN12] examines to what extent a legitimate activity show DNS stability. Analysing DNS traces from a large operator network with several customers, Berger *et al.* define stability metrics based on IP address association with domain names and geographic dispersion of IP addresses. This analysis concludes that half of the DNS responses analysed are actually related to stable domain names. However, the other half includes DNS responses for popular websites such as *www.google.com* or *www.facebook.com* or CDNs that are considered as highly dynamic.

Hence, some more elaborated techniques relying on several features have been developed to detect fluxing. These mostly rely on the analysis of passively gathered DNS traffic at RDNS level [PCDL09, APD⁺10, BKKB11] but other techniques [CTD⁺12] take as well advantage of active DNS probing in order to differentiate the activities in which different flux networks are used (phishing, spam, botnet C&C, malware networks). Antonakakis *et al.* [APD⁺10] perform fluxing domain detection by building an on-line reputation system for domain names. 41 features are computed from gathered DNS data among network-based, zone-based and evidence-based features. Evidence-based features refer to IP blacklists and honeypot data. Zone-based features consist in lexical analysis (character + n-gram analysis) and TLD analysis of domain names. Network-based features, being the largest set of features, represent IP addresses to domain name mapping, the different BGP prefixes of IP addresses, their country, their AS number, the registrars of the different IP sets, etc. Based on these features, several models corresponding to categories of domain names are built. A two stage clustering (network-based then zone-based) is applied to group domain names according to their different activities. Then, unknown domain names can be rate as malicious or legitimate using a proposed reputation system, which is a statistical classifier fed with the extracted features.

Other techniques targeted more specifically the identification of botnet command and control architecture using DNS monitoring [CLLK07, VSB09, CL12]. Based on already known malicious domain names, Villamarin *et al.* [VSB09] identify a pool of bot infected hosts. Then, by monitoring the DNS traffic below RDNS servers and comparing DNS traffic similarities with previously identified hosts, they discover other bots that query similar domains. Using the same DNS monitoring technique, Choi *et al.* [CLLK07] study the querying behaviour of individual hosts of a given network. Queried domain names are grouped regarding the different hosts that request

them. Analysing the evolution of pools of hosts that request the same domain names over time, botnet domains can be identified. The authors show that the hosts requesting botnet domains do not vary much over time compare to legitimate domain names. A metric to quantify the requesting host pool similarity is introduced.

Globally the technique to identify flux networks and botnets using DNS analysis have been proved efficient, identifying more than 99% of fluxing domain names [PCDL09]. However, these techniques require previous knowledge of already known fluxing domain names, since these rely on classification algorithm requiring training on ground truth data [PCDL09, APD⁺10, BKKB11]. Moreover, these techniques require large amount of DNS replies coming from different RDNS servers in several locations, for the same domain names in order to compute relevant features to feed classification algorithms. Some work [APD⁺10, BKKB11] were performed with historical data from ISC SEI [isc] and the applicability of these methods to smaller DNS dataset representing shorter capture durations is questionable. Hence, the time taken by these methods to identify flux networks is likely to be too long for phishing detection applications, since phishing attacks have a short lifetime. Finally, the last presented techniques for bot infected host detection present privacy concerns since both methods rely on DNS traffic monitoring below RDNS servers.

Malicious Domain Names

Some faster techniques, requiring fewer DNS data to extract relevant information, target the identification of malicious domain names. These techniques actually rely on the analysis of freshly registered domain names by monitoring the early DNS querying behaviour for these domain names. Hao *et al.* [HFP11] passively monitor DNS requests for an authoritative Top Level Domain server. They study both spatial and temporal DNS lookup patterns for newly registered domain names across multiple networks in order to early identify malicious domain names. By analysing DNS lookup patterns coupled with registration information about domain names, they identify differences between malicious (phishing, spamming, malware, scam, botnet domains) and legitimate domains. The detection technique uses Jaccard index computation over the set of /24 networks that request a domain names from a day to another or over a period of few days. The study of the early DNS querying behaviour for newly registered domain names shows that malicious domain names become highly popular and requested few time after their registration. This technique can allow DNS operators to identify malicious domains that are under their authority using only their DNS data and without requiring data from other networks as proposed with Kopsis [APL⁺11]. Kopsis use the same features coupled with requesters' information and IP address reputation to identify malicious domain names hosted on upper level name servers (AuthNS and TLDNS). Reactive registrar interventions by fast unregistering malicious domain names when these are disclosed is efficient in the fight against malicious domain names [LLF⁺11], rendering these automated identification techniques useful. Another alternative for the use of these malicious domain identification techniques is to build proactive blacklists of malicious domain names. Felegyhazi *et al.* [FKP10] use domain zones information about domain name registration to identify suspicious domain names and add them to a predictive domain blacklist. Based on known bad domain names, they compare the registration date, registrars, etc. with other domain names to find similarities.

Studying the registration of domain names and their early life right after is a sound solution to quickly find malicious domain names, although it can only be applied by domain registrars. Some domain registrars are heavily involved in the fight against malicious domain names registration while other do not take any countermeasure [LLF⁺11]. The consequence of using these techniques

is just to switch the problem from some registrars to others and it does not cope with malicious domain names registration.

Hence, other techniques requiring less DNS information and mostly focusing on lexical analysis of domain names were introduced [BG10b, YRRR10, YRRR12, APN⁺12]. Two main applications are targeted by lexical analysis in the state of the art, namely the detection of DNS tunnels and algorithmically generated domain names. Born *et al.* [BG10a] analyse bigram frequencies in domain names and show that DNS tunnels traffic has almost evenly distributed character frequencies while normal DNS traffic follow word language distribution. The conclusive remarks of this study are used to build fingerprints of legitimate and tunnelling DNS traffic that are used to assess the legitimacy of unknown traffic[BG10b].

Antonakakis *et al.* [APN⁺12] identify botnet involved hosts and C&C servers only using lexical analysis of domain names. The detection targets botnet using DGA-based (domain generation algorithm) malware [Ayc12]. Domain names included in DNS NXDomain responses passively captured below a RDNS server are clustered based on statistical features (n-gram, entropy, domain length, etc.). Then, using a Hidden Markov Model, models representing the composition of domain names are extracted from each cluster that is supposed to represent a single domain generation algorithm. The models are then applied to DNS traffic in order to identify domain names supposedly generated by one of the identified DGA. This technique has a limitation since models are built using a Hidden Markov Model that is not the real DGA. Domain generation algorithms usually use a seed such as the current date, hence the Markov Model is not necessarily suited to identify the malicious domain names. Yadav *et al.* [YRRR12] use statistical measures such as K-L distance, Levenshtein Edit distance and Jaccard measure to study the distribution of alpha-numeric characters in domain names. They can discover algorithmically generated domain names using this technique and were able to find domain names used by the Conficker botnet and to disclose a new unknown botnet during their experiment.

Malicious activities	Monitoring	Location	DNS features	Lexical analysis
Flux networks	Active/Passive	1	✓	
Domain fluxing	Passive	1 2		✓
Botnets	Active/Passive	1 2	✓	✓
Malicious domain names	Active/Passive	3 4 5	✓	✓
Cache poisoning	Active	2	✓	
DNS tunnelling	Passive	1 2	✓	✓

Table 2.1: DNS related malicious activities and the techniques to detect them

Techniques for lexical analysis of domain names are interesting since these do not require heavy deployment of large scale DNS monitoring infrastructures to gather a lot of DNS data. The few examples of existing techniques focused on the identification of DNS tunnelling or algorithmically generated domain names relying on the fact that these have different characters distribution. These domain names are however not used in phishing since meaningful domain names are mostly used in phishing and these would follow the alpha-numerical distribution of legitimate domain names. Hence, new techniques of lexical domain names analysis must be developed to cope specifically with phishing.

Table 2.1 summarizes some DNS related malicious activities and the techniques used in state of the art work to detect these activities. It shows if DNS monitoring can be used and if it is

active probing or passive monitoring as well as the monitoring/probing locations that are relevant according to Figure 2.4. It shows as well if DNS features and lexical analysis of domain names are relevant to detect each of these malicious activities.

Conclusion

In this section we presented the implementation and functioning of a core Internet service namely the Domain Name System. This service provides a paramount service to map domain names to IP addresses but give as well a meaningful way to locate resources on the Internet. In addition, it is used to enhance the availability and the delivery of contents on the Internet through Content Delivery Networks and has others security applications with blacklist delivery and SPF. Unfortunately, this service is also used to vectoring malicious activities by providing a way to hide the real infrastructure of malicious networks namely botnets and phishing servers. The development of flux networks hardens the fight against botnets and phishing since the actual hosting of malicious contents is difficult to disclose. However, since current malicious activities leverage the DNS, the monitoring of its traffic is worth to identify such activities.

The monitoring and analysis of the DNS traffic has been used for several purposes ranging from evaluation of network performances to threats detection. Several monitoring locations can be chosen according to their purposes, the scalability sought and privacy concerns. Some researchers proposed solutions to collect and analyse the DNS traffic in order to identify malicious flux networks being the support of botnets and phishing activities. Many proposed solutions that rely on the accumulation of large amounts of DNS traffic to operate. This implies high latency in the identification of malicious domain names, which is not an issue for botnet detection, since botnets are long time operating malicious infrastructure. However, the application to phishing detection is limited due to the short lifetime of phishing attacks. Other techniques have a limited scope since these can only be operated by specific Internet actors because these rely on DNS traffic monitoring at upper levels of the DNS hierarchy. Finally some good perspectives were observed from the lexical analysis of malicious domain names. Proposed solutions are fast to operate since these do not require large amount of DNS data. However, the applications of these techniques are limited to DNS tunnels identification and algorithmically generated domain names. Besides the fact that some legitimate services like CDNs use algorithmically generated domain names, phishing domain names are usually not and are composed of meaningful words that are involved in the social engineering process.

Hence, the lexical study of domain names coupled with few DNS data may show promises for phishing domain names identification. However, more elaborated techniques must be developed to differentiate between phishing and legitimate domain names since both are not algorithmically generated. One solution to operate a careful analysis of phishing domain names is to study the words composing them and their meanings to observe if some specific words or semantic fields are used differently in legitimate and phishing domain names.

Part II

Phishing Domain Names and URLs Detection

Chapter 3

Large Scale Passive DNS Monitoring for Identifying Malicious Domains

Contents

3.1	Passive DNS Monitoring Architecture	50
3.1.1	DNS Data Gathering	50
3.1.2	Distributed Storage and Processing System	53
3.2	Data Mining in DNS Space	54
3.2.1	DNS Features Extraction	54
3.2.2	Domain Names Clustering	56
3.3	Experimental Evaluation	58
3.3.1	Dataset	58
3.3.2	Feature Analysis	59
3.3.3	K-means Clustering Evaluation	61

Introduction

We have seen in Chapter 1 that an important vector of phishing attacks is rogue link, *i.e.* malicious URL, leading to fake websites, drive-by download, etc. Hence, the identification of these phishing links is an efficient way to cope with phishing attacks. An early detection of these phishing URLs is able to prevent users from an unexpected connection leading to phishing. We focus in this part of the dissertation on the identification of malicious URLs namely the ones used by phishers to target Internet users. As the main component of a URL is its domain name, we mostly focus the analysis on this part of the URL. Moreover the domain name is the part from the URL providing the largest amount of information because it is directly linked with the Domain Name System (DNS). This chapter is dedicated to the identification of malicious domain names through the analysis of DNS information.

Presented in Chapter 2, the DNS is one key component for the correct operation of the Internet. Several threats specific to the DNS exist: malicious domains hosting phishing sites or malware, covert channel communications over DNS, cache poisoning, client side attacks, etc. One activity of major interest related to DNS misuses is hosting malicious phishing sites. This malicious hosting mainly relies on two activities previously described in Section 2.1.3 namely fast-flux and double-flux. Fluxing domain names are characterized by specific DNS features. Some

features are the high count of IP addresses associated to a single domain name or the low *Time to Live* value for its Resource Records (*RR*). These features can only be computed from aggregated DNS data corresponding to a single domain name. Hence, a continuous monitoring of DNS activity is required and single DNS packet inspection is not sufficient. Long time period DNS monitoring requires an appropriate scalable data storage system. The exploitation of stored DNS data for mining and analysis requires as well the appropriate distributed techniques to retrieve the data.

We introduce in this chapter a method to identify fluxing domain names through passive DNS analysis. We propose the design and implementation of a passive DNS monitoring architecture leveraging distributed data storage and processing techniques to deal with large quantity of DNS data. We introduce nine features, extracted from DNS data, relevant for fluxing domain names identification. These features are used in an automated clustering method to capture relevant groups of functional different domains and especially malicious and legitimate domains. The contributions presented in this chapter were mainly published in [MFW⁺12, ME12].

This chapter is organized as follows: we start in Section 3.1 by presenting the requirements and implementation of a distributed passive DNS monitoring architecture. This is implemented to capture DNS packets. Section 3.2 describes the feature set we extracted from gathered DNS packets as well as the machine learning technique used to identify domain names activity. Finally, Section 3.3 presents results from experiments we made on two unlabelled DNS datasets.

3.1 Passive DNS Monitoring Architecture

In this section, the architecture and the design of the passive DNS monitoring solution are presented. This architecture has to comply with several major requirements in order to provide flexibility. We designed this architecture to have several purposes. These are not limited to the application presented in this chapter namely the clustering of domain names. From an operational point of view, this architecture is able to retrieve both online and offline DNS packet captures. Thus, it can be used as an online monitoring tool, but can also be applied to offline incident handling. For instance, in case of analysing a compromised network, the system is able to retrospectively mine DNS traffic, detect and report suspicious activities. One critical requirement for such an architecture is the large quantity of data that has to be processed and stored. Our experience in deploying a passive DNS monitoring tool showed that for a regional backbone network, the daily quantity of data can easily reach one GigaByte per day. Since tracking malicious domain names over a monthly basis can lead to dealing with data quantities in the hundreds of GigaBytes, we leverage distributed storage and retrieval solutions. First attempts to use relational database system was not scalable. Existing DNS monitoring approaches leverage efficient key-value storage systems (see *Cassandra* [LM10], or *Redis* [Ler10]). In our design we leverage the popular *Hadoop* framework [Whi09] in order to distribute both the computation and the data storage.

3.1.1 DNS Data Gathering

The core architecture of the DNS packets gathering system is illustrated in Figure 3.1. It is composed of three main components and is based on the architecture proposed by Florian Weimer in [Wei05]. The first component is a passive DNS sensor (Figure 3.1 *DNS passive monitoring*) that is a simple packet capturer filtering DNS related traffic. The choice to perform passive monitoring rather than active monitoring relies on two reasons:

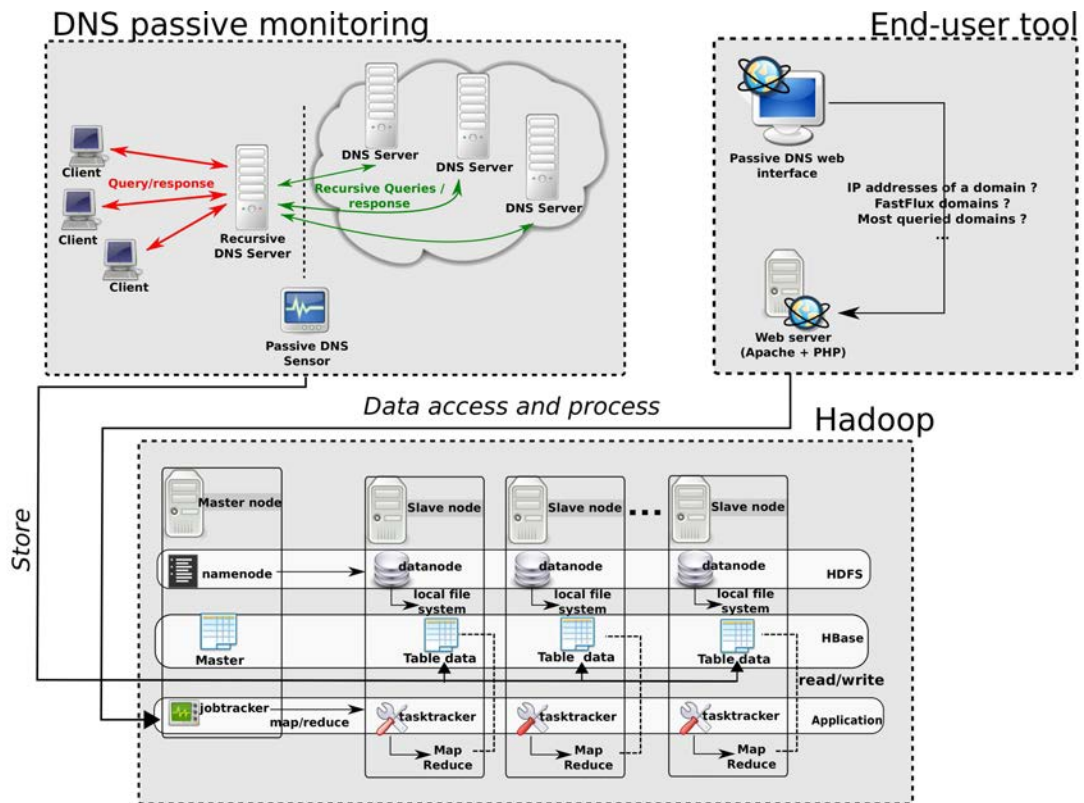


Figure 3.1: Passive DNS Monitoring Architecture

- **First**, to monitor passively a given recursive DNS server leads to get all DNS requests and replies of the clients it serves. Hence, it captures DNS data from domain names actually requested by Internet users and allow to protect these users by identifying their real threats.
- **Second**, this kind of monitoring is totally unnoticeable as it consists on a script running on the DNS sever and dumping packets received on a given interface. This prevents our monitoring to be discovered by miscreant owning malicious domains due to several DNS requests coming from the same machine. This identification would lead malicious domains owner to stop announcing a domain name being probed before getting enough information to infer its maliciousness.

The DNS sensor is placed between the recursive DNS server and the upstream DNS servers as depicted in Figure 3.1. Placing probes at this level meets two essential requirements:

- **It keeps users' anonymity:** Internet users while surfing on the Internet make DNS requests to their delegated recursive DNS server. Once the recursive DNS server gets a DNS request, it either makes the recursive DNS resolution or sends directly the DNS reply if the DNS record is cached. As a result, all DNS requests at this point are made by the recursive DNS server. No original DNS request is captured. This preserves anonymity.
- **It avoids to capture redundant DNS replies:** Thanks to the caching strategies of recursive DNS servers, DNS requests are only made when new information is needed *i.e.* either when a new domain name — not cached — is requested or when the *Time to Live* of

the Resource Record of a domain name cached expired. Hence, only DNS replies containing *new* information are captured.

The role of these probes is to listen for DNS replies, extract the data and feed the retrieved information into the distributed storage system. The information extracted from DNS packets and to be stored consists in:

- Resource Record type (A, AAAA, MX, NS, etc.)
- Response Code (NOERROR, NXDOMAIN, etc.)
- Question (*e.g.* the requested domain name)
- Response (value associated with the domain name — IP address (A Resource Record), mail server name (MX Resource Record), etc.)
- Authoritative Answer flag
- Time to Live (*TTL*) of the Resource Record
- Timestamp first seen
- Timestamp last seen
- Count of replies seen (between first and last seen)

This information is contained in DNS replies and has been presented in Section 2.1.1. Extracting this information provides an aggregated view of a domain name activity. It shows for instance how long a Resource Record is valid *i.e.* one record observed in several replies during a long period of time (first seen / last seen) denotes stability for a domain name. Whereas domain names with several different records depicts high variability. Hence, this information is suited to identify fluxing domain names, since one main feature of fluxing is variability.

This information extraction seems simple in theory, but in practice this tends to be difficult. Many DNS replies are not well structured and many reply messages have been observed to be erroneous. For instance, we observed large quantities of A Resource Record types that were returned to *127.0.0.1*. Therefore, a tedious case by case analysis has to be performed before storage. Another unexpected issue consists in letter capitalization in domain names. A question that arises is, whether domain names must be normalized to small capitals or if there may be large capitals too. The answer seems obvious as the DNS is not case sensitive, but we have observed that some important actors in the Internet, *e.g.* *Google*, play with variations within the same name, with both, small caps and large caps. For instance, for *google.com*, respective PTR records can have both kinds of caps. Some observed examples are *GoOgle.com*, *gOOgle.com*, *gOoGle.com*, etc. The assumption regarding this behaviour is that Google somehow uses this trick to limit the impact of cache poisoning, in case of badly implemented DNS cache server. Others assumptions are to infer the origin of the DNS replies, *e.g.* geolocalization of a datacenter, or to encode some data back from the original query of the user.

Stored data can be analyzed either by a human operator using the *End-user tool* depicted in Figure 3.1, which is a Web based interface or the data can be mined automatically by a data mining application thanks to the distributed data storage and processing system represented as *Hadoop* in Figure 3.1.

3.1.2 Distributed Storage and Processing System

Most of the activities related to DNS security monitoring require small processes running over a very large database. For instance, looking for the IP addresses corresponding to a domain name is an easy task, but leads to mine a huge volume of data. Thus, the paradigm shifts from a highly computational to a data-intensive problem to be solved.

MapReduce [DG04, LD10] is a design pattern for data-intensive problems. For achieving the same task as in a centralized approach, the design of the algorithm must be rethought. Mapping is applied onto each piece of distributed data and so, on each machine. Basically, data is processed to extract required features, with one feature being the key. Later, these key features are used to aggregate results, because all outputs of the mappers with the same key are sent to a unique reducer in charge of producing the final result.

For example, if each input line is a domain name and a matching IP address (*A* RR), then each mapper can emit the IP address with the domain name as key. Thanks to a shuffle phase, the reducer can collect every IP addresses corresponding to a domain name. This trivial example shows that, even if the required data (all entries with the same domain name) is not located on a single machine, finally it will be aggregated into the same reducer. The input of the reduce function is an intermediate key with a list of all intermediate values generated for this key by all mappers. Therefore, the reducer can generate the aggregated result for each key passed as argument of the reduce function.

The general architecture of a *Hadoop* [Whi09] cluster is depicted in Figure 3.1 (*Hadoop*). There are slave machines that are responsible for storing and processing the data in a distributed way and they are synchronized through a master. For storage, the *Hadoop Distributed File System* (HDFS) is composed of the *namenode* that only stores the file system structure, whereas data blocks are managed by *datanodes* with a configurable redundancy. For scalability purpose, accessing data is directly done through *datanodes*. The *namenode* only indicates where the data is, but it does not provide a proxy access to the data. *MapReduce* jobs are coordinated by the *jobtracker*, which is responsible of assigning the map and reduce tasks to the different *tasktrackers*. Then, these access their own local data for running the map. The reduce stage needs that the mappers directly output their results to a reducer that is determined by computing a hash of the key of the output map.

Regarding a trivial example of getting all the IP addresses associated to a given domain name, the data needs to be read for each request to execute. With a relational database, there is only one read and therefore retrieving the information is easier since it is structured. In a similar way, we leverage *HBase* [hba], an open-source implementation of a structured and highly scalable storage model that holds the benefits from the HDFS by providing a distributed and structured persistent storage to our system. As shown in Figure 3.1, there are master and slaves nodes. Similar to HDFS, the slaves run a *RegionServer* daemon responsible for storing subparts of tables locally, although the master tracks only the metadata changes. Even if not exclusively designed for working with *MapReduce*, *HBase* provides an interface to the *MapReduce* job. Thus, the mapper can read data in the table and the reducer can write data into the table. The table in *HBase* may be sparse since the data structure is flexible. Without loss of generality, it is a column-oriented approach, where each column is in fact a column family, which can contain any number of columns. For instance, for the DNS data, blacklisted domain names are stored as a column family with inner columns representing the different blacklists. Hence, depending on a given domain name, the number of columns may vary. We briefly reviewed the use of *HBase* by omitting some details for the sake of clarity. The interested reader is referred to [hba] for detailed explanations.

This section described the implementation of our distributed data storage and processing system with *Hadoop*. While running experiments, we observed significant latency during the initialization of the Map operation, but on very large volumes of data, this latency becomes insignificant. Once able to gather DNS data thanks to a scalable DNS passive monitoring architecture, we want to extract features therefrom that can highlight malicious domain names activity.

3.2 Data Mining in DNS Space

Some security abuses found in DNS can be identified through DNS data analysis. In this section we focus on the identification of two of them that are highly related to phishing. These are fast-flux and double-flux domain names. At a first glance, a simple fast-flux detection method could consist in monitoring DNS replies for large sets of different records associated with one single domain name. However, this method does not work when faced with large server farms or content delivery networks (CDN) like *Akamai* [aka] or *CloudFront* [clo]. Typical CDNs achieve the same redundancy as fast flux overlay networks, using similar techniques. The only subtle difference consists in the lifetime of a given domain name. Malicious domains have short spanned life times where CDNs domains do not.

We first propose a set of features to identify these abuses and explain why these are relevant. Then we present the clustering method we apply on this feature set to perform unsupervised learning and identify group of domain names having similar activity.

3.2.1 DNS Features Extraction

For identifying domain names activity we define different analysis parameters in order to model the DNS information. Therefore, we introduce nine features relevant for identifying fluxing domains (fast flux / double flux) according to the characteristics described in Section 2.1.3. These features are computed from the DNS information extracted from DNS packets, this was presented in Section 3.1.1. Having a DNS reply for a domain name $4ld.3ld.2ld.tld$. These features are extracted from the Fully Qualified Domain Name (FQDN) observed ($4ld.3ld.2ld.tld$) but also from lower level domain names $3ld.2ld.tld$ and $2ld.tld$. Top level domains are not considered because several entities register domain names under the same tld . Hence, no tld is likely to be a fluxing domain name. The features extracted from a domain name $domain$ are the following and their purposes are presented:

- *IPCount*: the count of IPv4 addresses associated to $domain$. We search for *A* and *CNAME* records from gathered Resource Records (*RR*) and count all IPv4 addresses associated to $domain$. This features is typical for detecting flux networks or content delivery networks as these present in theory higher IP counts compare to other kind of domains.
- S_{ip1} : an entropy-based index of IP address scattering. All IPv4 addresses associated to $domain$ are represented in binary form. Then, for all 32-bit positions of the IP address, we compute the Shannon entropy [Sha48] that considers the two conditions: the bit is 0 or the bit is 1. To recall, the Shannon entropy for a variable X is defined in Equation (3.1) with $p(x)$ being the probability that X is in state x and $p(x)\log_2(p(x))$ is set to 0 if $p(x) = 0$. After these computations, we sum the 32 entropy values to obtain an index $S_{ip1} \in [0; 32]$, where 32 represents the maximum scattering. The complete process to compute S_{ip1} is

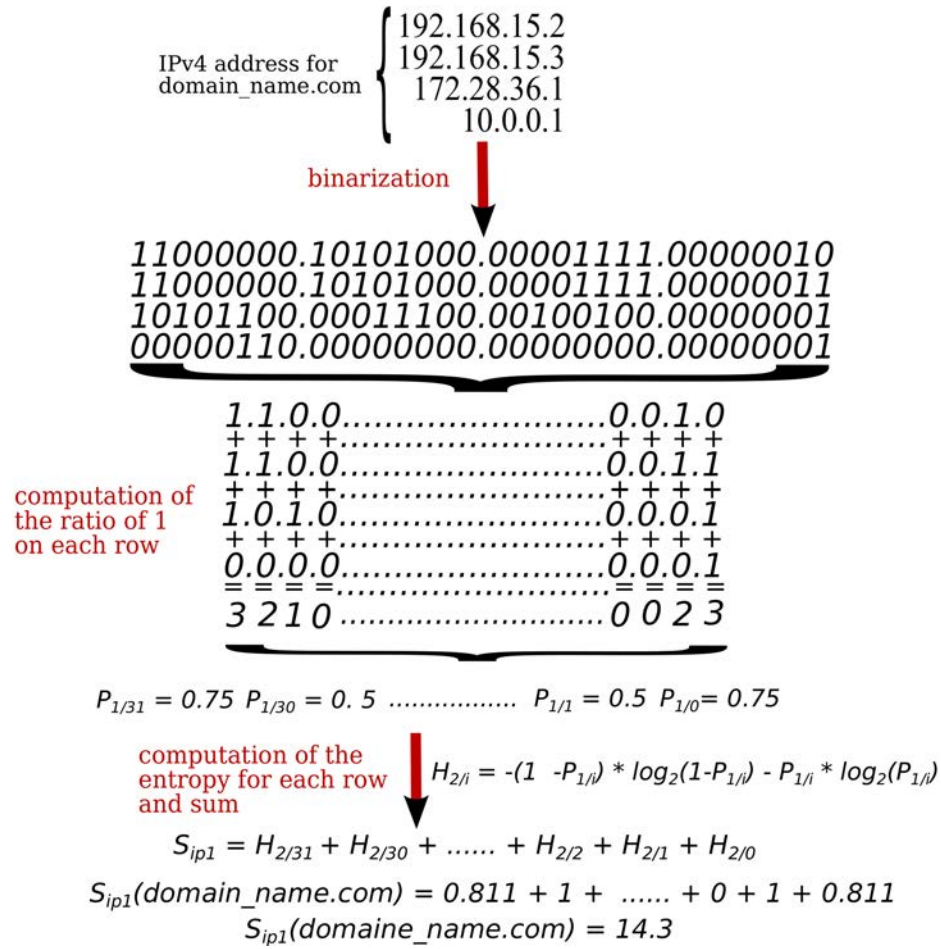


Figure 3.2: Computation steps for S_{ip1}

depicted in Figure 3.2. This index highlights the scattering of IP addresses associated to a given domain name through several subnets on the Internet. It discriminates CDNs from flux network as for a given probing place we expect CDNs to select servers geographically close while fluxing does not.

$$H(X) = - \sum_x p(x) \times \log_2(p(x)) \tag{3.1}$$

- S_{ip2} : another IP address scattering index. Considering an IPv4 address as $B_1.B_2.B_3.B_4$ for the four bytes that compose it. Having a set of IPv4 addresses associated to *domain*, we count the number of different B_1 , B_2 and B_3 we get among the set. These counts are defined respectively as *bytescount*₁, *bytescount*₂ and *bytescount*₃. To compute S_{ip2} as defined in Equation (3.2) we sum these three metrics after giving respectively the weight 100, 10 and 1 to each metric. Considering the set of four IP addresses in Figure 3.2 we obtain $S_{ip2} = 3 \times 100 + 3 \times 10 + 3 = 333$ The advantage of S_{ip2} compared to S_{ip1} is that it considers two functions, which are not used in the first index. Here, the differences in the positions of bytes in an IPv4 addresses are weighted, since intuitively the first byte in an IP address (B_1) has a higher relevancy in term of scattering than the last byte (B_4). Second, it also considers and counts the number of different IP addresses in a data set.

Regarding the entropy, if we have two addresses $0.0.0.0$ and $255.255.255.255$, S_{ip1} will be at maximum value, *i.e.* 32. However, if we have 20 different IP addresses for instance, the index would be lower, even if we can observe that the scattering becomes more important. With S_{ip2} , the more different IP addresses we get, the higher is S_{ip2} . S_{ip2} is not bounded on any range of values.

$$S_{ip2}(\text{domain}) = 100 \times \text{bytescount}_1 + 10 \times \text{bytescount}_2 + \text{bytescount}_3 \quad (3.2)$$

- *TTL*: the mean *TTL* value from all *A* Resource Records associated to *domain*. Low *TTL* values are usually a relevant indicator of flux networks and CDNs.
- *ReqCount*: the total count of DNS replies made over the observation period for *domain*. Variations in the distribution of this variable might indicate that an anomaly occurred. Typical phishing sites are highly requested during a short period of time, which differs a lot from the rest of the time.
- *TimeUp*: the period in days (*lastseen* – *firstseen*) during which we can observe DNS replies for *domain*. If there is only one observed request, this feature is set to 0. It has been observed that legitimate domain names have longer time spans, while malicious domain names exhibit daily/hourly lifetimes [AR14].
- *ReqRate*: the ratio of replies per time period for *domain*. This feature is a combination of the two previous features, which provides the number of requests on a per hour basis. This statistical feature captures the average usage pattern/frequency for *domain*.
- *SubDom*: the count of subdomains of *domain*. We count in our data the number of domain names matching the pattern *prefix.domain*. CDNs are known to use a lot of algorithmically generated subdomains. These are used once, thus a high count of subdomains can be observed for CDNs.
- *ServCount*: the count of authoritative servers for a domain name. For this feature, high values are an indicator of double flux networks.

We defined these nine features since these are sufficient to represent the DNS characteristics of different domain names activities. Table 3.1 show the several features with their expected values according to different domain names activities. We define five types of domain names, namely fluxing domains, CDN domains, user tracking domains, popular domains and low popular domains. We can see in this table that the several domain name types have different expected values showing that the feature set should be relevant and sufficient to distinguish these activities.

3.2.2 Domain Names Clustering

The extraction of the feature set previously presented is aimed at identifying domain names activity *i.e.* fluxing, double-fluxing or legitimate activities. Hence once these features extracted from a dataset, we want to subject them to a machine learning algorithm in order to infer domain names activity. Machine learning algorithms are divided in two main categories namely supervised learning and unsupervised learning — recently a third category emerged and consists in semi-supervised learning. Basically the difference between these two first categories is the data subjected to them.

Features	Fluxing	CDN	User Tracking	Popular	Low popular
@IP count	++	++	+	+	--
IP scattering	++	+	+	-	--
TTL value	--	--	--	-	++
DNS Requests	-	++	+	++	-
Uptime	-	--	-	++	+
Request rate	-	++	+	+	--
Subdomains	-	++	++	+	--
Servers	++	+	+	+	--

Table 3.1: Expected feature values depending on domain names activity

In supervised learning, the data feeding the classification algorithm is labelled. In other words, to use supervised methods, one needs a dataset of instances categorized in predefined classes. Based on this, the algorithm infers a classification model that can further be applied to unknown instances in order to determine their class. One drawback of this method is that it can only identify what it learned *i.e.* whatever the instance subjected to is, it is put in one class even if it should not belong to. This method has already been applied in several works related to passive DNS analysis but with a different feature set [PCDL09, ADL⁺10, BKKB11].

Unsupervised learning, in the opposite, works with unlabelled data. The algorithm tries to find hidden structure in the data based only on the features characterizing each instance. One important category of unsupervised learning algorithm is clustering. Clustering consists in splitting a dataset in different groups containing instances close in terms of feature values. The advantage of this technique is that no previous knowledge about the instances one wants to cluster is needed.

Even though we built a features set to identify certain kind of domain activities, the gathering technique for our DNS data does not enable to know the class of a domain a priori *i.e.* our dataset is unlabelled. In addition, building a static classification model with n classes at time t does not ensure that unknown instances analysed at time $t + 1$ will belong to one of these n classes. One characteristic of malicious domains is their short lifetime [AR14] and Internet miscreants constantly develop new subterfuges [DTH06, AR14] to cope with new detection techniques. Hence, the chosen data mining approach is based on a clustering task and the k-means algorithm [HW79] is chosen to perform it.

The k-means algorithm is a classical clustering algorithm that is commonly used in data mining [Jai10]. The aim of k-means is to divide instances into k clusters. More formally this means, given a set of instances (x_1, \dots, x_n) , with each instance being a d -dimensional vector (in our case $d = 9$), k-means tries to optimally divide the n instances into k clusters $S = \{S_1, \dots, S_k\}$ by minimizing the within-cluster sum of square (WCSS) defined in Equation (3.3) where $\|x_i - c_i\|$ represents the distance from an entity point $x_i \in S_i$ to its cluster centroid c_i . The number of clusters k has to be set in advance and ($k \leq n$). For a detailed description of the k-means algorithm we refer the reader to [HW79].

$$\sum_{i=1}^k \sum_{x \in S_i} \|x_i - c_i\|^2 \quad (3.3)$$

We introduced in this section nine features to infer domain name activities and presented a clustering algorithm to subject the feature set to. This feature set and the machine learning technique are used on passively captured DNS packets to assess their relevancy.

3.3 Experimental Evaluation

This section introduces two datasets from which features are extracted. We highlight the relevancy of the feature set chosen by analysing it on the test datasets as a first step. Then the experimental outcomes of the clustering phase are presented.

3.3.1 Dataset

We aim to test our method against two datasets presenting different characteristics to evaluate its applicability. We have used two datasets that are different with respect to geographical location, volume, duration and access networks. Both datasets were captured according to the technique presented in Section 3.1.1. The first dataset originates from the INRIA Nancy Research Labs, which represents a medium sized campus network in France. The second dataset originates from a regional Internet Service Provider located in Luxembourg. Detailed information about these datasets is presented in Table 3.2 in terms of captures duration, count of DNS replies captured and size of the dataset. For the remaining of this section the datasets will be named according to the country they come from: dataset Luxembourg and dataset France.

Country	Luxembourg	France
Duration	1 hour	48 days
# DNS replies	10 M	70,095
Size of Dataset	270 MB	22.8 MB

Table 3.2: Passive DNS capture statistics for dataset Luxembourg and dataset France

Additional statistics on the differences between these two datasets are summarized in Table 3.3. We looked at the characteristics of the DNS information for a set of hosts and domains depending on the capture location. We selected some popular and well represented domain names like: Akamai (*akadns.net*), Facebook (*facebook.com*), Apple (*apple.com*), and Google (*google.com*). Table 3.3 presents the *IPCCount* of some subdomains of these and their respective *SubDom* feature. It is worth noting that the *IPCCount* of only one subdomain example is given for both Akamai and Apple, as Apple uses Akamai [aka] to deliver some of its contents.

Even though the two datasets were captured in close locations, some differences are noticeable. The domain name *apple.com* has similar statistics in both dataset, however we can see a five fold difference in the *IPCCount* of *csi.l.google.com* from dataset Luxembourg to dataset France, the same observation holds for *star.facebook.com*. This trend tends to indicate that big Internet actors perform load balancing on their servers and dedicate more servers to bigger countries (*e.g.* France) than they do for small ones (*e.g.* Luxembourg). This explains the difference in *IPCCount* for same domain names in different countries. Since the *IPCCount* is an important feature for

flux network detection, we must be careful to cluster data coming from the same location and not to mix data from different locations. This would influence domain names feature values and lead to misleading clustering or classification results. This fact is another argument to prefer unsupervised learning to supervised learning in this case study. It proves that a classification model learned in a given location would not necessarily be relevant in another location.

It is worth noting as well that *akadns.net* has a nine fold difference in *SubDom* from dataset France to dataset Luxembourg. This is natural since Akamai is a major Content Delivery Network that hosts services which are much more represented in the largest dataset. Some estimates [HWLR08] of the Akamai network ranged it at about 50,000 hosts world wide. Although a passive DNS analysis only reveals geographically closed, from the passive DNS probes location, Akamai hosts, this data is relevant to estimate the local load balancing and service availability of the Akamai CDN.

Features	Luxembourg	France
<i>IPCount</i> for <i>csi.l.google.com</i>	74	404
<i>IPCount</i> for <i>star.facebook.com</i>	8	39
<i>IPCount</i> for <i>x.apple.com.akadns.net</i>	24	24
<i>SubDom</i> for <i>akadns.net</i>	1,137	135
<i>SubDom</i> for <i>google.com</i>	306	444
<i>SubDom</i> for <i>facebook.com</i>	174	66
<i>SubDom</i> for <i>apple.com</i>	134	156

Table 3.3: *IPCount* and *SubDom* features for some domain names

3.3.2 Feature Analysis

We merged the France and Luxembourg dataset to form a joined dataset. We filtered and retained only the different FQDNs that have been requested and for which at least one reply with the RCode *NOERROR* was received. The nine features are extracted for each domain of this filtered set. We analyse the values of these features among the instances of the merged dataset to have an idea of these having discriminative values.

Table 3.4 shows for each feature the minimum (Min), maximum (Max), median (Median), fifth percentile (5%) and ninety-fifth percentile (95%) values for each of the nine features. In addition the mean (Mean) and standard deviation (SD) values are given at the end of each row. To have a better idea of some feature values repartition, histograms giving the ratio of instances per feature value are depicted in Figure 3.3. Moreover the density curve is drawn (blue) as well as the fifth percentile (5%) and ninety-fifth percentile (95%) values (red).

At first glance on Figure 3.3 we see that most of the elements (around 50%) of the dataset are concentrated on a single value: one IP address per domain explaining IP scattering index S_{ip1} and S_{ip2} of 0 for most domain names. Most domain names have only one request and two authoritative servers, etc. Taking the features *SubDom* and *ReqRate* in Table 3.4 almost 95% of the domains have been requested only once ($ReqRate = 0$) and more than 95% of the domain names have no subdomain. However, the maximum values of these features are: $Max(ReqRate) = 9,620 req/hour$ and $Max(SubDom) = 194$. This shows that most domain names of the dataset are similar from a DNS point of view but nonetheless few elements of the set are outliers compared to the rest. This is actually a good hint that our feature set is able to detect domain names having *abnormal* behaviour.

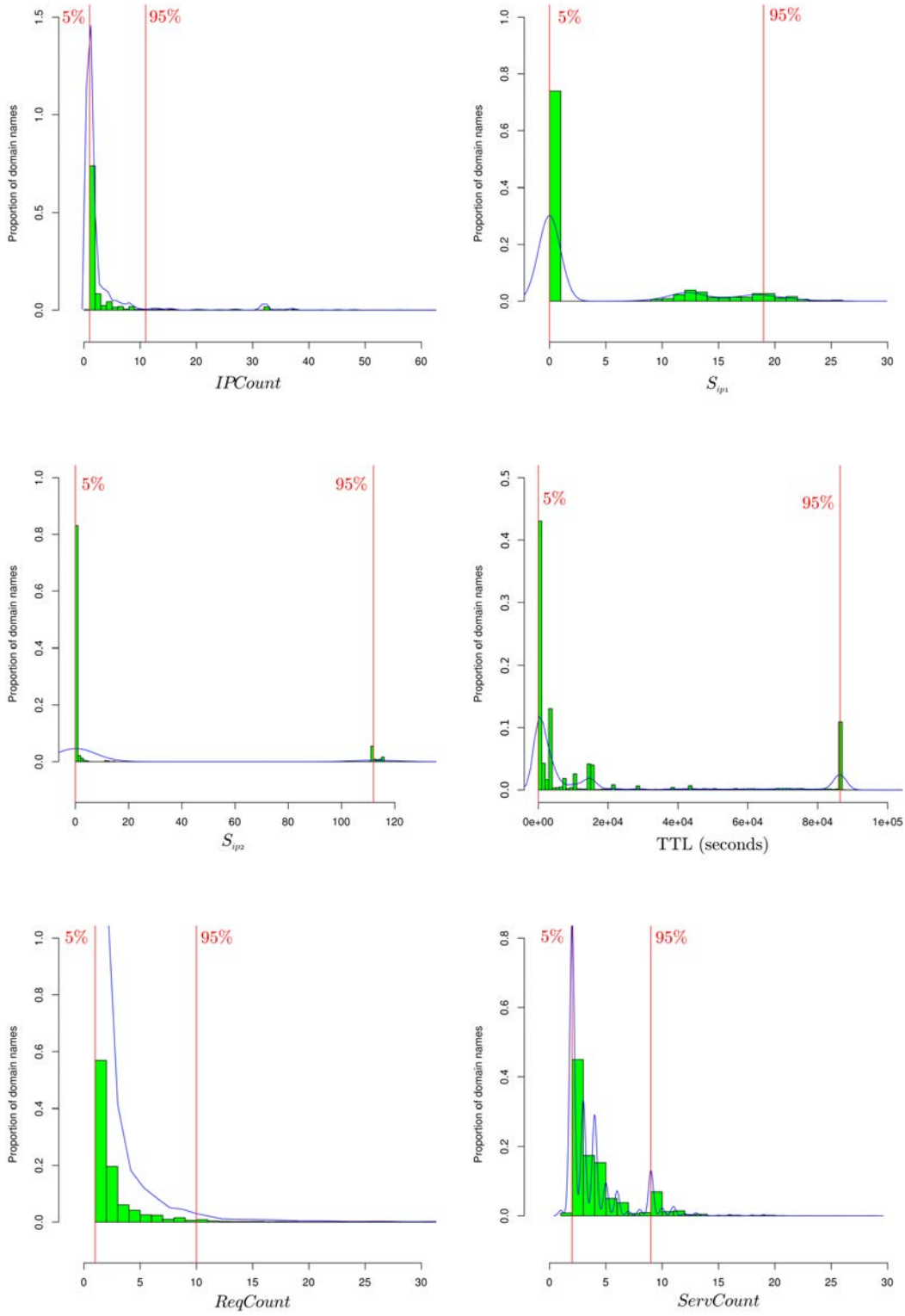


Figure 3.3: Features statistics

Features	Min	5%	Median	95%	Max	Mean	SD
<i>IPCount</i>	0	1	1	11	389	3.12	9.66
<i>S_{ip1}</i>	0	0	0	19	27	4.037	7.089
<i>S_{ip2}</i>	0	0	0	111	789	17.226	54.011
<i>TTL</i>	0	17	3,573	86,400	604,800	20,580	41,612
<i>ReqCount</i>	1	1	1	10	591	3.793	14.51
<i>TimeUp</i>	0	0	0	27	48	4.698	9.737
<i>ReqRate</i>	0	0	0	0.117	9,620.496	12.074	27.018
<i>SubDom</i>	0	0	0	0	194	0.164	3.277
<i>ServCount</i>	1	2	3	9	29	3.767	2.675

Table 3.4: Feature values repartition

From Figure 3.3 we observe that most domain names have few associated IP addresses. However a small spike in 32 indicates that around 3% of the domain names of the dataset have exactly 32 IP addresses associated with. This explains why we have a significant part of S_{ip1} values between 10 and 22. The spike at value 111 for S_{ip2} corresponds to domain names being associated with two IP addresses having no byte in common *i.e.* two IP addresses having no common prefix.

The biggest part of the domain names have a low TTL . Previously we explained that a low TTL is the characteristic of a domain belonging to a CDN or belonging to a flux network. However most of the domain names are normally not in these categories. There are two explanations for having so many domain names with a low TTL . The first one is that one single CDN can use a lot of different domain names and subdomains. For instance one domain name of Akamai has more than 1,000 subdomains while regular domain names have few in general. Hence, lots of query/reply for FQDNs related to CDNs are captured during one hour explaining a high representation in the dataset. The second reason is that domain names with low TTL do not stay long in recursive DNS servers' cache (the time of the TTL). Hence, since we took only a subset of one hour passive DNS capture for one dataset, the chance to see domain names with low TTL is higher than the one to capture replies for domain names having TTL greater than one hour. The spike we can see for 10% of domain names having a TTL around 85,000 seconds corresponds actually to a TTL of 86,400 seconds that is equal to one day. Many domain names from low to medium popularity do not require load balancing and have a stable infrastructure that does not require frequent update of DNS information. The TTL of Resource Records for these domain names is often set to one day.

3.3.3 K-means Clustering Evaluation

To assess the relevancy of the feature set, we use in this section the k-means clustering algorithm that has been previously presented. Experiments were performed using the open-source machine learning framework Weka [HEH⁺09] on the France dataset only, to avoid bias in results because of different locations as explained in Section 3.3.1. This tool is known for its large library of supervised and unsupervised machine learning algorithms. As a first step for determining the optimal number of clusters k in which we should split the dataset we use previous processing. Two techniques are widely used to determine the number of clusters to build with clustering techniques.

One uses graphical analysis of dendrogram in order to choose the optimal split. This method

is based on hierarchical clustering technique that builds a tree showing all the different split possible to divide a set in k subsets. This method is highly exhaustive and shows all the options to split a set. However, hierarchical clustering techniques have high complexity and are limited to small dataset due to processing capacity limitation.

As a result, the solution chosen is based on the minimization of the within-cluster sum of square (WCSS) presented earlier in Equation (3.3) as part of the k-means algorithm. The computation of this metric for a given cluster consists in adding the distances from each element of the set to cluster to its cluster centroid. A low value of WCSS corresponds to a good assignation of elements to clusters. Hence, by minimizing this value, one ensures that elements having close features values are clustered together and that these elements have high similarity between them. We built clusters out of the global dataset using the k-means algorithm for $k \in \{2; 15\}$ and computed the sum of the within-cluster sum of square from each clusters formed during the process. The code corresponding to this experiment is the following:

```

1  _____ WCSS computation algorithm _____
2
3  for n = 2 to 15:
4
5      // clusters initialisation: centroids c[i] and cluster assignation A[j]
6      D' = D
7      for i = 1 to n:
8          j = rand(|D'|)
9          c[i] = D'[j]
10         D' = D' - {c[i]}
11
12     for i = 1 to |D|:
13         A[i] = argmin(j= 1 to n) { ||D[i] - c[j]||^2 )
14
15     change = true
16     while change: // recalculate clusters until convergence
17         for i = 1 to n: // recompute centroids
18             mean,count = 0
19             for j = 1 to |D|:
20                 if A[j] == i:
21                     mean += D[j]
22                     count ++
23             c[i] = mean / count
24
25     change = false
26     for i = 1 to |D|: // reassign instances to clusters
27         a = argmin(j= 1 to n) { ||D[i] - c[j]||^2 }
28         if a != A[i]:
29             A[i] = a
30             change = true
31
32     for i = 1 to |D|: // compute the WCSS for the n clusters formed
33         WCSS[n] += ||D[i] - c[A[i]]||^2
34
35 return argmin(i= 2 to 15) { WCSS[i] } // return the optimal cluster count

```

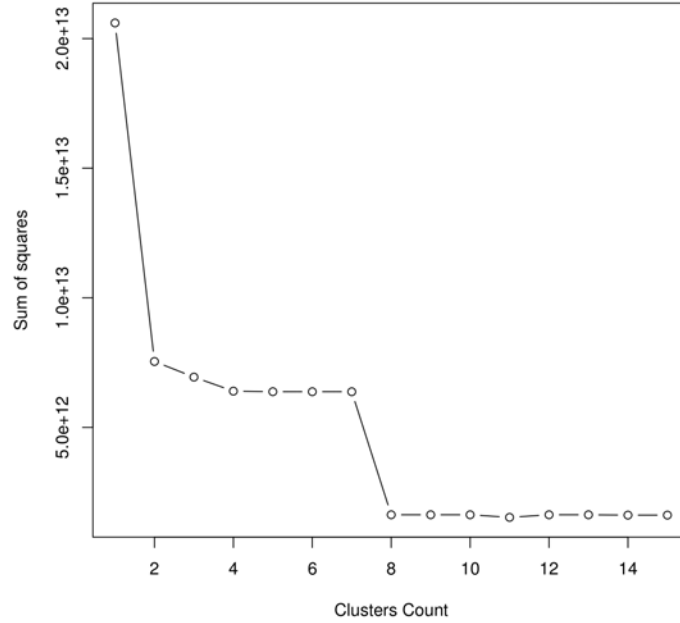


Figure 3.4: WCSS values for 2-15 clusters obtained with k-means

This code shows the computation of the k-means on a feature vectors set D composed of domain names with their nine extracted DNS features previously presented. For each of the n clusters to form, it selects an initial centroid being a element randomly picked from D . Centroid are stored in $[c]$. Elements of D are then assigned to the cluster having the closest defined centroid. This assignment is stored in $[A]$. Then, an iterative process updates the cluster by recomputing the centroid and reassigning element to the closest recomputing centroid. Once the algorithm converge – no element is reassigned to other clusters after recomputing the centroids – it computes the WCSS being the sum of the distance between each element of D and its centroids. Finally, the algorithm return the optimal cluster count.

The value of within-cluster sum of squares for every $k \in \{2; 15\}$ is depicted in Figure 3.4. Two events are noticeable in this graph. The first is the large decrease in WCSS values while passing from one set to two clusters. The second is the smaller, but still important, decrease in WCSS values while adding one more cluster from seven to eight. This graph shows that building from two to seven clusters does not improve the cluster composition, so it is from eight to fifteen. Hence, the optimal number of clusters to split our set are two and eight. We favor $k = 8$ value since we consider that domain names activities can be classified in more than two categories.

During experiments we analysed the relations between the obtained clusters and the different features. Figure 3.5 depicts values for $TimeUP$, S_{ip1} , $ServCount$ and TTL for elements of the eight clusters. Each small cross represents one element of each cluster in x axis and its associated value for the feature in y axis.

Three clusters, cluster 3, 6 and 7 are associated to high values of S_{ip1} feature and very similar low values for TTL especially for cluster 6. The S_{ip1} values are more than two orders of magnitude higher compared to values of the remaining clusters. However, these three cluster have different $TimeUp$ values. While looking at the cluster components, we have observed

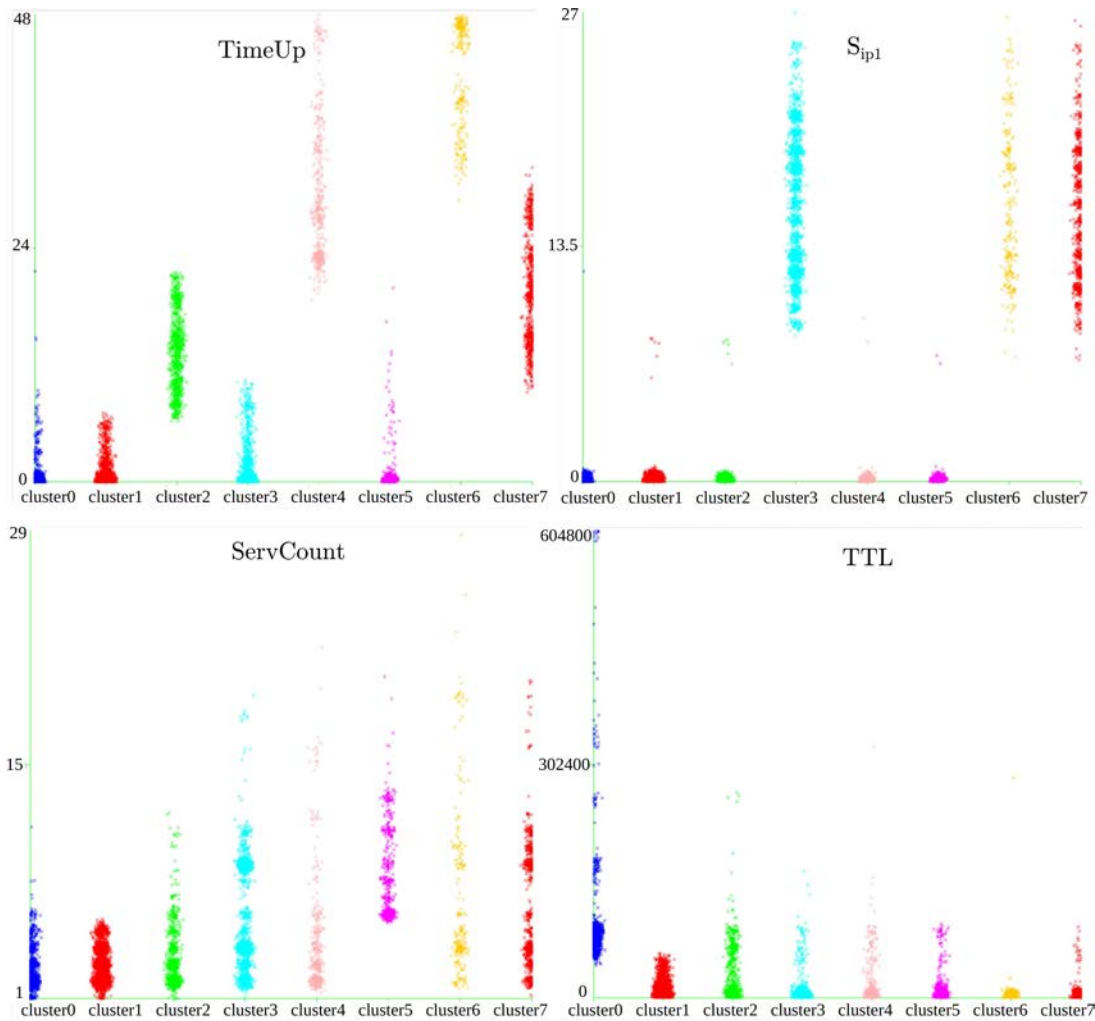


Figure 3.5: Feature values according to cluster number

that cluster 6, having the highest *TimeUp* value, contains very popular domains like *google.com*, *facebook.com*, *skype.com*, etc. This is an expected conclusion that highly popular domains provide high availability (low *TTL*), perform load balancing over servers scattered all over the world (high S_{ip1}) and have requests observed all along the observation period ($TimeUp \approx 48$ days). Cluster 7 with lower *TimeUp* is grouping domain names that perform user tracking services, like for instance *doubleclick.net*, *tradedoubler.com*, *quantcast.com*, etc. Cluster 3 is particular. While clusters 6 and 7 have both dispersion metrics in the high ranges (S_{ip1} , S_{ip2}), cluster 3 groups domain names characterized by high S_{ip1} values and lower S_{ip2} values compared to clusters 6 and 7. A manual analysis of this cluster revealed that most domain names are related to CDNs as for example Akamai [aka] or CloudFront [clo]. Typical domain names in this cluster are composed of an algorithmically generated suffix followed by a CDN domain name such as *fbcdn-video-a.akamaihd.net* or *d19n4gh4cmsbnt.cloudfront.net*. This explains the short life time (*TimeUp*) compared to the two previous clusters, as these FQDNs are composed of an algorithmically generated subdomains that may be used only once. This result is consistent with the operation of CDNs, where DNS replies are based on geographical locations of the requesting client. In our case, we have observed high values in the number of *A* records per domain name, but most

returned values were discriminated by the S_{ip2} metric. The several IP addresses actually belong to networks with a common IP prefix.

Cluster 5 regroups domain names with high count of authoritative servers (*ServCount*) and high count of DNS replies per time interval (*ReqRate*). These domain names are very popular with very high ranks according to Alexa's website ranking [ale]. Typical domain names in this cluster are *apple.com*, *amazon.fr* or *adobe.com*. Finally cluster 0 with high *TTL* values groups lowly popular domains according to Alexa's ranking. These were occasionally consulted by users of the network we monitored.

We identified 80 malicious domain names in the dataset. These were identified thanks to regularly updated domain blacklists that can be freely downloaded from the Internet. These consist in Malware Domain List [malb], DNS-BH Malware Domains Blocklist [mala] and Zeus blocklist [zeu]. These 80 domain names were not enough to form their own cluster, however these were grouped in cluster 7 with thousands of legitimate ones. As we expected these malicious domain names have features like low *TTL*, high IP scattering and were mainly fluxing domains. Examples of such domains are:

- 00007.ru: used for hosting malicious code.
- 000.bbexe.cn: used for phishing site and rogue login script.
- 01.finni.in: used for malicious hosting.
- 010608.myftp.biz: used to host infected pdf files that exploit an Adobe Acrobat Reader vulnerability.

These experiments show that extracting the nine DNS features previously described from two datasets and follow-up analysis leads to reveal differences between domain names and between locations where the data is gathered from. The feature analysis highlighted major trends in domain names feature values, while revealing as well outliers domain names having values different from the majority. In a last step applying the k-means clustering to form eight clusters out of one dataset showed the ability to group domain names according to their activities. Content Delivery Network domains were group together, as were user tracking services. Low popularity domain names were separated from high popularity domains such as *google.com* or *facebook.com*. While the ability of the method to form a single cluster containing all malicious domain names was not proved, the 80 malicious domain names we identified were clustered along other legitimate in a single cluster. This was mainly due to the low quantity of such domain names in our dataset preventing from forming a separated cluster.

Conclusion

As a core service of the Internet, the DNS carries a huge amount of information that is extremely rich to security monitoring. To exploit this information in an efficient and automated way, we presented a passive DNS monitoring solution that leverages both an advanced distributed processing deployment pattern and a relevant data mining algorithm. Using this system, we analysed two different datasets of passively collected DNS traces by applying an automated clustering algorithm, where the different clusters represent different types of DNS traffic activities. We showed that the proposed DNS feature set is relevant for discriminating domain names activities and more precisely malicious from legitimate. We were able to identify fluxing domain names being involved in phishing activities and delivery of malicious content as well as other activities such as content delivery or user tracking.

Several work regarding passive DNS analysis have been made previously using centralized storage system and processing [Wei05, ZBW07] to identify malicious domain names. But to the best of our knowledge we are the first to propose the use of Map-reduce like algorithms for this goal in [ME12]. In addition, where related work [ADL⁺10, BKKB11, PCDL09] mainly use supervised machine learning for this purpose, we proposed a system relying on clustering techniques with the k-means algorithm that do not require previous knowledge about DNS data. This system was fed with state of the art features and new features introduced in this chapter that highlight IP scattering.

This technique while being efficient at first glance has some drawback. We have seen that probes location is an important factor that impacts values of feature we extract from a DNS replies. Information for the same domain name coming from different probes is different. Hence, to get general features for a given domain name, passive DNS probes must be deployed all around the world as does the ISC Security Exchange Information [isc] through their pDNS project. Without this global deployment, results would be biased by users' local interest in some websites. Another requirement to render trustworthy decisions on the activity of a domain name is the need to get several DNS replies for the same domain name in order to get accurate characterising features. For instance *IPCount*, S_{ip1} , S_{ip2} or *TimeUp* are features that are accurately computed after a long observation period and several DNS replies. This introduces a delay in the identification of malicious domain names. Finally, we have seen that the dataset to analyse must be balanced. If the set of malicious domain names is too small, chances are high that these will be put in an existing cluster rather than form their own new one as seen during experiments.

To cope with some of the passive DNS analysis limitations and provide a technique to label unknown clusters, we propose to analyse the composition of domain names in order to identify phishing ones.

Chapter 4

Phishing Domain Name Identification Based on Word Relatedness

Contents

4.1 Phishing URL Obfuscation	68
4.1.1 Obfuscation Techniques	69
4.1.2 Obfuscation Words Semantic	70
4.2 Semantic Analysis of Domain Names	71
4.2.1 Word Extraction	72
4.2.2 Word Relatedness Computation	73
4.2.3 Similarity Metrics	74
4.3 Domain Sets Comparison	76
4.3.1 Dataset	76
4.3.2 Similarity Metrics Evaluation	77
4.3.3 Domains Set Size and Composition	80

Introduction

The use of passive DNS probes to gather DNS related features is an efficient way to categorize domain names activity. Using passive DNS data, we were able to group malicious domain names, especially those performing phishing. This malicious activity has specific features that can be revealed by analysing DNS packets fields. However, we have seen in the previous chapter that the deployment of two DNS probes gives different data that may lead to biased results. Hence, large scale deployment is required to obtain relevant features. Another drawback of this method is the need for several DNS replies for the same domain names in order to compute some features paramount to identify fluxing domains (*e.g. IPCount, ServCount*). Moreover phishing attacks do not always rely on fluxing networks, the use of this technique is the sign of phishing campaigns launched by criminal organization owning or renting botnets to support fluxing. Other phishing campaigns having smaller scope do not rely on such malicious infrastructures. Finally, the method proposed in Chapter 3 groups domain names having similar activities. Nevertheless, the clusters formed are not labelled and a manual analysis is required to identify clusters' activity, *e.g.* malicious or legitimate.

Passive DNS analysis has also some assets that we previously presented and being interesting for phishing detection. It is a passive detection method being undetectable. DNS requests are made before any connection to the potential malicious domain name is established and all requests of a given network goes through its delegated recursive DNS server. This asset is interesting if we seek to protect a specific network as domain names actually requested can be analysed. In addition, the amount of information captured through passive DNS analysis is quite limited compared with the whole network traffic data, since DNS traffic is only a subset. Since the passive capture does not store information about request originators, it is as well privacy friendly. Privacy is an issue in security monitoring and network forensic [MMN08, AKM⁺11].

Hence, this chapter focuses on the analysis of passive DNS traffic. We propose a method to automatically analyse passive DNS data for tracking phishing related domains. In complement to the previous chapter, our analysis relies on the semantic of domain names. Phishing mainly relies on social engineering lures as we described in Chapter 1. Domain names used to perform phishing, and being embedded in spoofed email for instance, are obfuscated using specific words. These words are carefully chosen by phishers to make Internet users feel safe, by using attractive words, including and combining brand names or specific keywords such as *secure* or *protection* [GPCR07].

We propose a technique to identify malicious group of domain names by quantifying semantic relatedness between set of words extracted therefrom. Domain names are taken from passively gathered DNS replies and grouped according to common features. Then meaningful words are extracted from domain names and three metrics to compute semantic relatedness between set of words are introduced. These are tested on ground truth data to identify sets of malicious domains. This method only relying on lexical and semantic analysis cope with the need for large scale deployment of DNS probes and long time observation period. It only uses domain names and no additional DNS features are needed. In addition this provides a technique to label unknown clusters of domain names. This complements the clustering method introduced in Chapter 3 by identifying clusters formed as malicious or legitimate. The contributions of this chapter were partly published in [MFSE12b].

This chapter is structured as follows: we start by presenting the URL obfuscation techniques and words used in phishing URLs in Section 4.1. Then, Section 4.2 introduces the process of word extraction from domain names and the three metrics to quantify semantic similarity between two sets of words. Section 4.3 presents the evaluation of the metrics on a large test set of malicious and legitimate domain names and applications to phishing domains detection are proposed.

4.1 Phishing URL Obfuscation

Phishers usually try to lure their victims into clicking on rogue URLs pointing to phishing sites or drive-by downloads. From observations made on a substantial set of blacklisted URLs from the community website PhishTank [phi], it was noticed that phishing URLs are often obfuscated by embedding different terms making them very long and including several meaningful words. This is performed to delude Internet users who are not aware of DNS hierarchy and seeing keywords at any level of a URL make them trust the rogue link. We first present the different URL obfuscation techniques before analysing the semantic fields of words embedded by phishers in URLs.

4.1.1 Obfuscation Techniques

Different URL obfuscation techniques are used with the aim of hiding the real host, more particularly the registered domain, the only part of the URL that cannot be freely defined. If one wants to use a domain name *mydomain.tld* and derive several URLs from it: *url1.mydomain.tld*, *url2.mydomain.tld/file*, he has first to register the domain name *mydomain.tld* at a domain registrar, ensuring that it cannot be registered by anybody else. Once the domain name *mydomain.tld* is registered at a domain registrar it is added to a central registry database and nobody can register this domain name again. This guarantees that only the entity having registered a domain name can use it. Assuming a phisher wants to trap PayPal users, he must use a *domain.tld* other than *paypal.com*, since this domain name is already registered by PayPal Inc. The phisher must register a domain name *mydomain.tld* and try to deceive people by blending labels such as *paypal* into the rest of the URL: *login.mydomain.tld/paypal*. The goal is to hide the real registered domain of the URL: *mydomain.tld*, which is not related to PayPal Inc.

A registered domain consists of two parts: a *main level domain* and a *public suffix*. A *public suffix* is a domain name suffix under which an Internet user can register a name. It can be just a Top Level Domain like *.com*, *.lu* or a combination of level domains like *.co.uk* or *.blogspot.com*. The *public suffix* will be abbreviated *ps*. A *main level domain* is the level domain preceding a *public suffix* and will be abbreviated *mld* for the remainder of this document. A registered domain is then: *mld.ps*. For instance in *www.paypal.com/login*: *com* is the *ps* and *paypal* is the *mld*.

The different obfuscation techniques consist in blending either the original domain name or phishing keywords into the remaining part of the URL. These keywords are usually the targeted brand, related services of the brand and other attractive words such as *secure*, *login*, *protect*, etc.

Obf. Type	Example
Type I	http://school497.ru/222/www.paypal.com/29370274276105805/ http://paypal.com.eu.compte.client.update.condst.com.br/
Type II	http://www.quadrodefertas.com.br/www1.paypal-com/encrypted/ssl218 http://sezopoztos.com/paypalitlogin/us/webscr.html?cmd=_login-run
Type III	http://cgi-3.paypal-secure.de/info2/verikredit.html http://paypal-shopping.co.il/
Type IV	http://69.72.130.98/janaseva/https.paypal.com/uk/onepagepaypal.htm ftp://212.13.144.72/SERVICE/PayPal.com/security/alert/paypal.com
Type V	http://tiny.cc/clientID00858JD8 http://goo.gl/HQx5g

Table 4.1: Example of obfuscated URLs for the domain name *paypal.com*

Assume a URL formed of a hostname with different level domain (*ld*), a path (*path*) and a query (*key=value*): *http://5ld.4ld.3ld.mld.ps/path1/path2/path3?key1=value1&key2=value2*. The obfuscation consists in blending keywords into the path, the query and the lower level domain of the hostname (*5ld.4ld.3ld*). In the following we present the most used URL obfuscation techniques [GPCR07], with examples given in Table 4.1 for the domain name *paypal.com*:

- **Type I: URL obfuscation with other domain:** In this case, the *mld.ps* is a real domain name, usually registered by the phisher, while the original website being phished is part of the path, the query or the upper level domain.

- **Type II: URL obfuscation with keywords:** Again the *mld.ps* is a real domain name, and the brand being phished and related words are part of the path, the query or upper level domain.
- **Type III: Typosquatting domains or long domains:** the *mld.ps* of the URL is the domain being phished but misspelled, with letters or words missing or added, or the domain is pronounced the same way as the original but written differently. The targeted brand can also be combined with other words to create an unregistered domain.
- **Type IV: URL obfuscation with IP address:** the URL's hostname is replaced by an IP address and the brand being phished is part of the path or the query.
- **Type V: Obfuscation with URL shortener:** A URL shortening service is used to hide the name of the real host. Such URLs are not meaningful and are mainly used in phishing attacks targeting services that use this kind of short URL, like Twitter.

This chapter focuses on the identification of the obfuscated URLs of type 1, 2 and 3. Since the method we propose is based on data extracted from DNS packets passively captured, only domain names are analysed. When requesting content from a URL on the Internet only the domain name is the object of the DNS request. Thus, only obfuscations operating at the lower domain levels of the domain names and typosquatting are considered.

4.1.2 Obfuscation Words Semantic

Domain names used for phishing have a global trend namely that these are composed of several meaningful words. While looking at words embedded in several phishing domain names we can observe that these tend to be the same or at least to be related. Some examples of phishing domain names coming from PhishTank [phi] blacklist are:

- *myapple-login.com.daflonpneus.com.br*
- *https-paypal-update-your-account-paypal-credit-card-payent.restaurantekosherclub.com*
- *update.information.verfying.paypal.com.etiquetasonline.com.br*
- *paypal-iden-6e8rg5-easyway-config.ccmnow.com*
- *wellsfargo.online.com.nakshathira.com*
- *newzealand-onlinescuredadobe.aluminiosperuanos.com*

Looking at these few examples, we see that these are based on *mld.ps* that have no relationship with the targeted brand. However, the words blended in lower level of the domain names consist in the targeted brand and other words that belong to limited semantic fields. The semantic fields present can be classified as follows:

- account access: *login*, *account* and *iden*
- IT maintenance: *update* and *config*
- security: *secure* and *verifying*
- web services: *paypal*, *apple*, *adobe*, *wellsfargo* and *online*

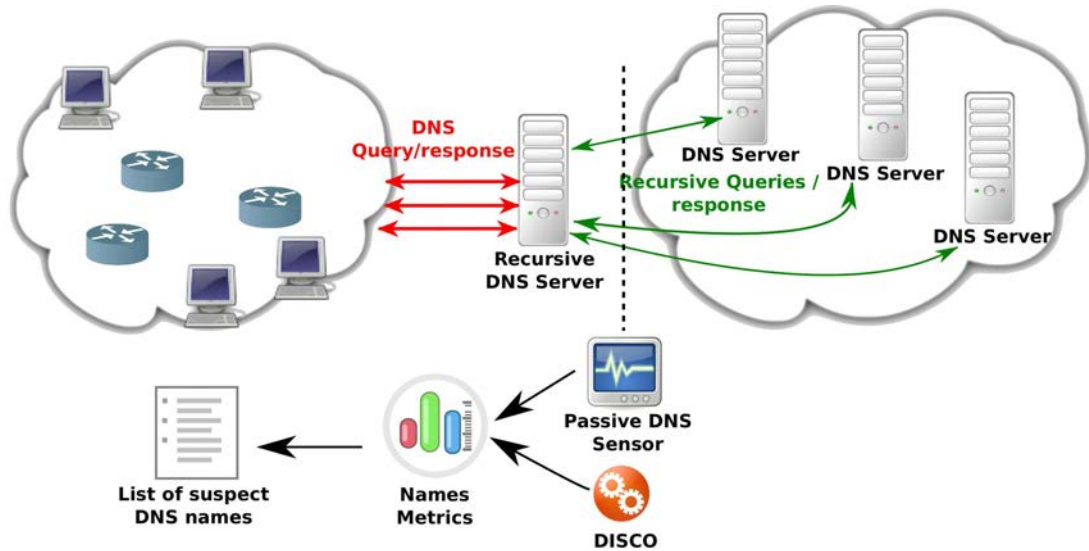


Figure 4.1: Malicious domain set identification: architecture overview

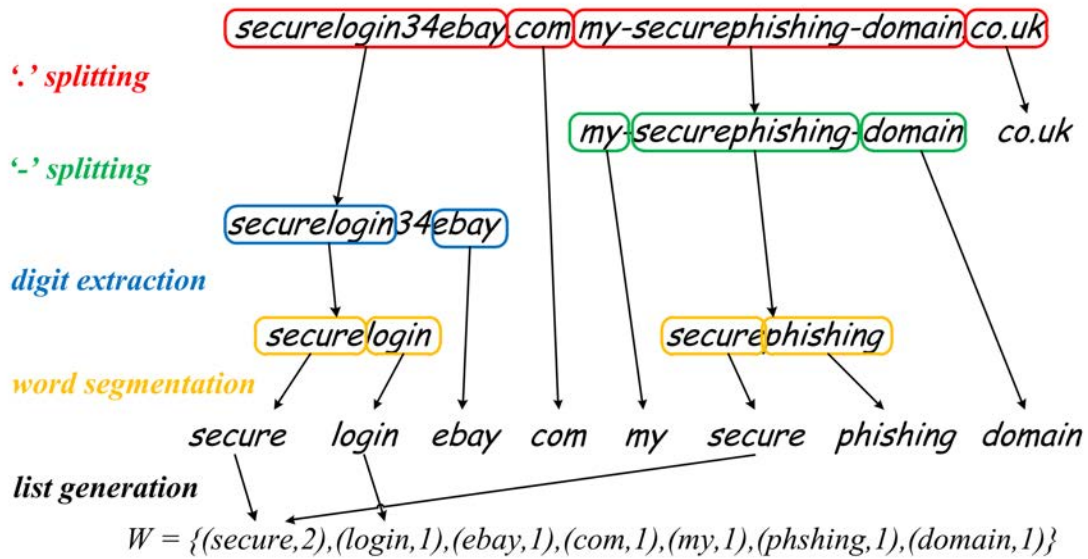
These four semantic fields identified from a little subset of phishing URLs are actually recurrent ones because they contain the words that Internet users expect to see while clicking on a link. These fields are also directly related to the data and the information that phishers want to steal: account credentials, credit card number, etc.

The last category (web services) contains famous brands of online services widely used on the Internet (payment services, online banking, Apple). These targeted services often deal with payment or finance and need login information to access personal account this is why some words related to account access are contained in the first category (account access). The second category (IT maintenance) contains words related to maintenance. This can be used either to ask for credential or credit card information update and thus steal the current credentials or to pretend an update of software version is needed and push users to download malware. Inexperienced users mainly accept the updates without doubting the link is not trustworthy. As a result, the words present in the third category (security) are used. Some keywords related to security are used to emphasize the 'legitimacy' of the URL. Even if few examples are listed, these semantic fields are widely used and have been observed in hundreds of URLs during analysis. Hence, we extract words from domain names in order to identify if these match the semantic fields phishers tend to use as these are limited.

Domain names concerning this analysis are gathered the same way as in previous chapter through a passive DNS sensor as depicted in Figure 4.1. This time, only one passive DNS probe is deployed behind the recursive DNS server delegated to a selected network. Domain names are extracted from DNS replies before being analysed by a module leveraging semantic metrics to render decision on the maliciousness of a set of domains.

4.2 Semantic Analysis of Domain Names

Unlike standard texts, domain names are composed of few words and therefore deriving a global semantic is hard. To make it easier, grouping multiple domain names to increase the count of words before determining the overall semantic is possible. It is also compatible with phishing because a given phishing campaign often uses several domain names and subdomains at the

Figure 4.2: Word extraction for `securelogin34ebay.com.my-securephishing-domain.co.uk`

same time [XYA⁺08]. Several phishing campaigns are launched from the same botnets or same Autonomous System (AS) known to host malware [SKG12].

This domain grouping can be performed using the clustering technique presented in previous chapter. This is possible since we showed that we were able to isolate phishing domains in a single cluster. Other state of the art technique such as [PB08] can be used as an alternative. The goal is to analyse if domain names from two different set types, legitimate and malicious, are composed of words that share semantic similarities or disclose semantic differences. Thus, there are several steps to perform this operation: (1) an extraction of the words that compose a domain name, (2) find a metric characterizing semantic relatedness between two words and, (3) introduce metrics to give a score of similarity between two sets composed of several domain names.

4.2.1 Word Extraction

The first requirement is to split a domain name in order to extract all words that compose it. This is highlighted in Figure 4.2 for the domain name `securelogin34ebay.com.my-securephishing-domain.co.uk`. This is not a real domain name but it is proposed to depict the splitting process.

Domain names are first split by level domain according to the separating dots '.', which are basic separators between level domains in the DNS. The *public suffix* is excluded based on the list from Public Suffix List [pub], as this is a part of the domain that is not defined by phishers but it is constrained by the registrar where the domain is registered. Hence, this part does not include any words that can bring semantic value to the analysis. Since hyphens '-' are allowed in domain names, a second split is done accordingly. Furthermore, digits are removed and considered also as separating characters. This leads to have remaining parts composed of letters only ([a-z] as the DNS is not case sensitive), which can still be composed of several words like `securelogin` or `securephishing` in the example of Figure 4.2.

The segmentation of alphabetical parts leverages a technique from [SH09] that consists in successively dividing a label in two parts. Every time, the likelihood of the split is computed

until finding the combination that gives the maximum score. Hence, assuming a label l , for each position $i \in [1; \text{len}(l)]$, l is divided in two parts and the probability $P(l, i)$ given in Equation (4.1) is computed. In this equation $\text{pre}(l, i)$ returns the substring of l composed of the first i characters and $\text{post}(l, i)$ is the remaining part. $P_{\text{word}}(w)$ returns the probability of having the word w , equivalent to its frequency in a database of text samples. This process is applied to all newly split parts $\text{pre}(l, i)$ and $\text{post}(l, i)$ as long as $\exists i \in [1; \text{len}(l) - 1]$ such as $P(l, i) \geq P_{\text{word}}(l)$. At the end we get all the words composing a given label like *secure* and *login* for *securelogin*.

$$P(l, i) = P_{\text{word}}(\text{pre}(l, i)) \times P_{\text{word}}(\text{post}(l, i)) \quad (4.1)$$

Finally, the occurrences of each word are counted and stored in a set W by couples (*word*, *occurrence*) as shown in Figure 4.2. The obtained set is:
 $W = \{(\text{secure}, 2), (\text{login}, 1), (\text{ebay}, 1), (\text{com}, 1), (\text{my}, 1), (\text{phishing}, 1), (\text{domain}, 1)\}$.

The count of occurrences for each word is operated for further use in semantic metrics where words appearing several times in domain names can be associated to a larger coefficient in metrics computation in order to impact the results accordingly. The process of word extraction is performed on every domain name of a set to analyse. The set W is updated for each processed domain name.

4.2.2 Word Relatedness Computation

Computing a similarity score between two sets of words is not straightforward. First, defining word relatedness is tricky as this can describe several relationships between words. The relationship to identify is semantic relatedness or similarity *i.e.* words sharing the same semantic field as for instance *mars* and *venus*, which are two planets. This kind of relationship is partially defined by identifying hypernymy which links general synsets to specific ones like *planet* to *earth*. By identifying all words having a common hypernym one can deduce a semantic field as for *earth*, *mars* and *venus* are *planets*. Meronymy, which is a part-whole relation, synonymy or antonymy are other relationships to deduce semantic fields. As claimed by Kilgarriff [Kil03], these usual notions imply a manual analysis to establish relationships between words.

WordNet [Mil95] is an example of a lexical database containing a collection of English language words that can provide related words to a given word. The relationships in WordNet are defined manually and are limited to a subset of the English vocabulary. This limits its applicability and its extension to further language or semantic fields. Hence, applications of WordNet to words extracted from domain names are limited because Internet vocabulary includes several languages and words that are not present in usual dictionaries.

This is why automatic method to approximate these notions have been developed [CH90, LD97, CV07]. These usually define the distributional similarity and relatedness of words based on occurrence and co-occurrence count in text sample. DISCO (extracting DIStributionally related words using CO-occurrences) [Kol08, Kol09] is an example of computing automatically relatedness score between two words. DISCO considers the distance between two words within a window by defining $\|w, r, w'\|$, the count of occurrence of the word w' , r words after the word w , where $r \in \{-3; 3\} \setminus \{0\}$. Table 4.2 shows an example of the computation of $\|w, r, w'\|$ for two sample pieces of text centered on *services*. A sliding window of length four words is applied to text samples in order to compute the mutual information of every word. The Mutual Information between two words w and w' , $I(w, r, w')$, has been introduced by Hindle [Hin90] and is defined in Equation (4.2).

$$I(w, r, w') = \log \frac{(\|w, r, w'\| - 0.95) \times \|*, r, *\|}{\|w, r, *\| \times \|*, r, w'\|} \quad (4.2)$$

The word similarity metric considered, $sim(w_1, w_2)$, is based on the Mutual Information $I(w, r, w')$ and was defined by Lin [Lin98]. For two words w_1 and w_2 and considering $T(w)$ as all the pairs $(r, w') \mid I(w, r, w') > 0$, $sim(w_1, w_2)$ is given in Equation (4.3).

$$sim(w_1, w_2) = \frac{\sum_{(r,w) \in T(w_1) \cap T(w_2)} I(w_1, r, w) + I(w_2, r, w)}{\sum_{(r,w) \in T(w_1)} I(w_1, r, w) + \sum_{(r,w) \in T(w_2)} I(w_2, r, w)} \quad (4.3)$$

The algorithm of DISCO is based on the similarity metric $sim(w_1, w_2)$. DISCO can be fed with any text sample and apply this metric to disclose word relatedness. DISCO was subjected to four word corpora consisting in the full content of Wikipedia in English, Spanish, German and French. Given a word w_1 , DISCO can either give a similarity score with another word w_2 or return $Disco(w_1, n)$, consisting in the n most related words to w_1 . These words w_i are ordered by their decreasing respective similarity score $sim(w_1, w_i)$. Using DISCO on words extracted from domain names we can compute the similarity pairwise between words or return the most related words. However this does not provide a technique to compute semantic relatedness between two set of words.

position	-3	-2	-1	0	+1	+2	+3
sample1	a	client	uses	services	of	the	platform
sample2	the	platform	provides	services	to	the	client

$\ services, -2, platform\ =1$	$\ services, -3, the\ =1$
$\ services, -2, client\ =1$	$\ services, -3, a\ =1$
$\ services, -1, uses\ =1$	$\ services, 1, of\ =1$
$\ services, -1, provides\ =1$	$\ services, 1, to\ =1$
$\ services, 3, platform\ =1$	$\ services, 2, the\ =2$
$\ services, 3, client\ =1$	

Table 4.2: Example of co-occurrence count (2 windows centered on *services*)

4.2.3 Similarity Metrics

Given a set of p domain names $D = \{d_1, \dots, d_p\}$, words are extracted from each domain name following the technique given in Section 4.2.1. These form a set of n couples. Each couple is composed of one word w_i with its corresponding number of occurrences o_{w_i} in all the set of domain names: $W_D = \{(w_1, o_{w_1}), \dots, (w_n, o_{w_n})\}$. $distword_{w_i, W_D}$ is defined in Equation (4.4) as the frequency of a word w_i in W_D .

$$distword_{w_i, W_D} = \frac{o_{w_i}}{\sum_{j \in \{1, n\}} o_{w_j}} \quad (4.4)$$

Following this formula, three metrics are introduced to quantify the semantic similarity between two sets of domain names, A and B . All these metrics are based on $sim(w_1, w_2)$ given in Equation (4.3). These present different computational complexities and different strengths that will be assessed in next section.

The first metric, $Sim_1(A, B)$ defined in Equation (4.5), considers all the words $w_A \in W_A$ and $w_B \in W_B$ and compares them pairwise using $sim(w_1, w_2)$. The sum of pairwise score is done

to provide a score of similarity between the two sets A and B . The higher the score, the more similarity there is. Since this performs a pairwise comparison, the computation complexity of this score is $O(n^2)$.

$$Sim_1(A, B) = \sum_{w_A \in W_A} \sum_{w_B \in W_B} sim(w_A, w_B) \quad (4.5)$$

The second metric is similar to the first one except that it considers the number of occurrences of the words into each dataset A and B . Logically, this is done such that $sim(w_1, w_2)$ obtained from words appearing more frequently has a bigger impact on the final similarity score than $sim(w_1, w_2)$ computed from words that appear few times. Therefore, when computing $Sim_2(A, B)$ in Equation (4.6), $sim(w_A, w_B)$ is multiplied by the frequency of w_A and w_B in their respective dataset $distword_{w_X, W_X}$ to weigh each similarity. $Sim_2(A, B)$ has as well a complexity in $O(n^2)$.

$$Sim_2(A, B) = \sum_{w_A \in W_A} \sum_{w_B \in W_B} sim(w_A, w_B) \times distword_{w_A, W_A} \times distword_{w_B, W_B} \quad (4.6)$$

Preliminary experiments showed that computing $sim(w_A, w_B)$ is time consuming. Pairwise comparisons in Equations (4.5) and (4.6) are not efficient due to their complexity in $O(n^2)$. However, retrieving the top n most related words of w using $Disco(w, n)$ requires approximatively the same amount of time than computing $sim(w_A, w_B)$. Thus, we consider that results of $sim(w_A, w_B)$ between words having few relatedness is negligible and we propose to include this result only for the n most related words. Hence, we take the n most related words of each word of the set W_A before searching them into the set W_B . Equation (4.7) describes this process.

$$Sim'_3(A, B) = \sum_{w \in W_A} \sum_{w' \in Disco(w, n)} sim(w, w') \times distword_{w', W_B} \quad (4.7)$$

As highlighted in Equation (4.7), the score is weighted by the frequency of word $distword_{w', W_B}$. Words w' not present in W_B have $distword_{w', W_B} = 0$ and thus are not considered for computation. There is no pairwise comparison anymore and the complexity drops to $O(n)$ as the length of $Disco(w, n)$ does not depend on the number of words in W_A . However $Sim'_3(A, B)$ is not symmetric. Thus, we define the symmetric metric $Sim_3(A, B)$ in Equation (4.8) combining two computations of $Sim'_3(A, B)$ and $Sim'_3(B, A)$, and having a complexity in $O(n)$ as well. We acknowledge that $Sim_3(A, B)$ is an approximation of $Sim_2(A, B)$ but it has a far lower complexity and is more suited to comparison of big sets of domain names.

$$Sim_3(A, B) = Sim'_3(A, B) + Sim'_3(B, A) \quad (4.8)$$

Three metrics giving a score of semantic relatedness between two sets of domain names Sim_1 , Sim_2 and Sim_3 are defined. The computation complexity of these metrics relies on the number of words w that are extracted from a set D composed of n domains. A domain name can be composed of several words, but considering that several words appear in different domain names of the same set we can reasonably approximate $n \simeq w$. Hence, assuming that we compare two domain sets of n elements each, the complexity of Sim_1 and Sim_2 is $O(n^2)$. Since Sim_3 does not make pairwise comparisons of all words, its complexity is $O(n)$. Even though Sim_3 has a lower complexity, the next section assesses if it is able to have comparable accuracy to Sim_1 and Sim_2 .

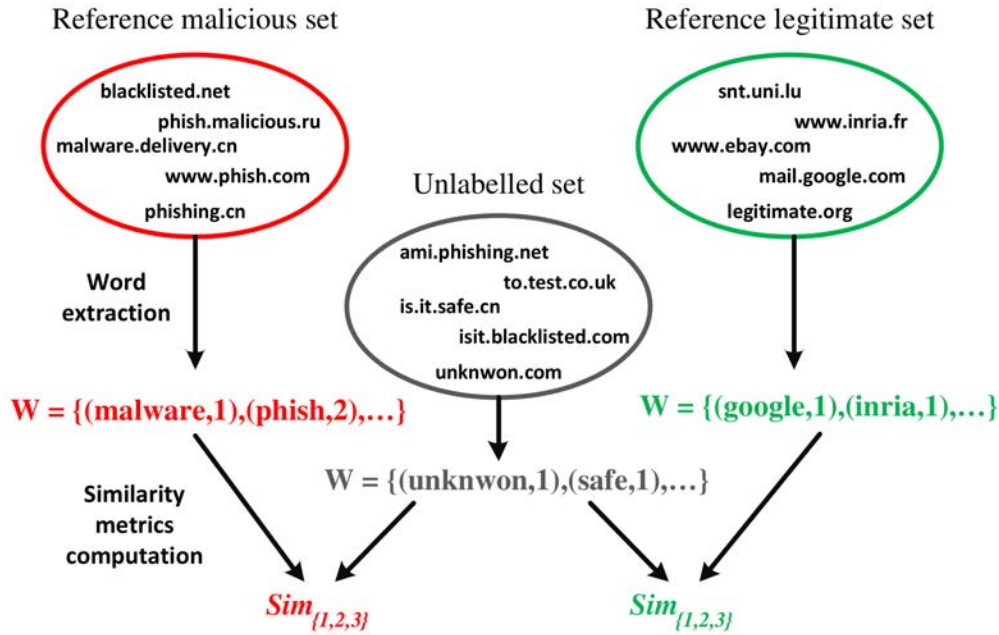


Figure 4.3: Unlabelled domain set identification process

4.3 Domain Sets Comparison

To assess the relevance of the introduced metrics and show that semantic relatedness evaluation leads to discriminate malicious from legitimate domain names, we consider two datasets of malicious and legitimate domain names on which we compute the metrics. Then, we analyse the results of these experiments and compare the metrics to prove the efficiency of this semantic approach to discriminating malicious domain names from legitimate domain names. The proposed metrics can be used to label unlabelled sets of domain names as malicious or legitimate by computing them with a reference malicious and respectively legitimate set as depicted in Figure 4.3. The comparison of the obtained similarity score enable to render a decision on the maliciousness of the unlabelled set.

4.3.1 Dataset

To test our method, two sets of domain names are formed: a legitimate set, containing non-malicious domain names and a malicious set, containing domain names confirmed malicious.

Malicious Dataset

To create a large dataset including a variety of malicious domain names, three freely downloadable blacklists are used. These have been selected because each of them proposes an historic of blacklisted domain names related to several types of malicious activities. The details of these three sources are as follows:

- **PhishTank** [phi]: PhishTank is a collaborative project to which people can submit phishing emails and websites. Suspected phishing URLs are further checked by several people

before being confirmed as malicious and added to a blacklist. 5,521 phishing URLs were downloaded for our experiments.

- **DNS-BH** [mala]: DNS-Black-Hole maintains an up-to-date list of domain names known to support malware and spyware diffusion. A list of 17,035 malicious domain names is available.
- **MDL** [malb]: as PhishTank does, Malware Domain List is based on a community approach and construct a blacklist from proposed inputs from contributors. This list contains 82,480 URL entries.

Following the extraction of the distinct domain names from the 105,036 URLs and the deletion of duplicated entries, 66,633 distinct domain names remain to form the malicious dataset.

Legitimate Dataset

To balance, *i.e.* to have same count of malicious and legitimate instances, the malicious dataset with legitimate domain names, two sources of non-malicious domain names are chosen. These provide again variety and faithfully represent normal domain names in a realistic manner. These sources are

- **Alexa** [ale]: Alexa is a company that collects browsing behaviour information in order to report statistics about Web traffic and websites ranking. It provides a ranking for the top *1,000,000 sites*. 50,000 domain names out of the top 200,000 have been picked for our experiments.
- **Passive DNS** from a Luxembourg Internet Service Provider: this dataset was obtained using a passive DNS infrastructure as depicted in Figure 4.1 in collaboration with a Luxembourgish ISP. It is composed of 16,633 DNS names after having deleted duplicate entries from Alexa and known malicious domain names (checked with the previously described blacklists).

Finally, 66,633 entries are contained in the legitimate dataset. Hence, the two datasets, *legitimate* and *malicious* are of equal sizes. We agree that a 1/1 ratio between legitimate and malicious domains does not faithfully reflect real world repartition. However, this repartition is needed to confirm semantic differences between legitimate domain names and malicious domain names.

4.3.2 Similarity Metrics Evaluation

To assess the relevance of the semantic approach in identifying set of malicious domain names, we show that words composing malicious domain names belong to different semantic fields than those composing legitimate domain names even if malicious domain names use patterns or brands to mimic the composition of popular URLs.

To have an extensive evaluation, the initial malicious and legitimate datasets defined in Section 4.3.1 are both split in five subsets of equal sizes (13,326 domain names). These ten subsets called mal- i , $i \in \{1; 5\}$ for malicious domain names, and leg- i , $i \in \{1; 5\}$ for legitimate domain names, are compared by computing similarity scores. The following comparisons are done:

- Sets composed of malicious domain names vs. sets composed of malicious domain names (*mal/mal*)
- Sets composed of malicious domain names vs. sets composed of legitimate domain names (*mal/leg*)
- Sets composed of legitimate domain names vs. sets composed of legitimate domain names (*leg/leg*)

In order to keep only meaningful relevant words for semantic relationship analysis, words composed of at least four characters are considered during extraction, others are discarded. It avoids considering generic words or articles such as *the*, *of*, *www*, etc. that would be present in both sets and bias the comparison.

Table 4.3 shows the values of Sim_1 computed pairwise between all the malicious and legitimate subsets. The lowest Sim_1 is, the less similarity there is between the two compared sets. A gray shaded key is used for improving the readability and three main areas are easily separable: *mal/leg*, *leg/leg*, *mal/mal*. Therefore, the first metric, Sim_1 is helpful for distinguishing malicious and legitimate domain names. The lowest similarity values are logically obtained when the types of subsets are different (*mal/leg*). The similarity scores are globally below the value 20 for these sets, this can be seen by the darkest area in Table 4.3. Comparing two legitimate sets, similarity values around 25 are reached (bottom left in Table 4.3) highlighting a 25% increase in similarity between *leg/leg* compared to *mal/leg* sets. Malicious datasets exhibit even higher semantic similarity with a maximum score of 31.

	leg-5	leg-4	leg-3	leg-2	leg-1	mal-5	mal-4	mal-3	mal-2
mal-1	19.3	19.3	20.1	18.7	20.1	29.7	30.0	30.3	31.0
mal-2	19.4	19.3	20.2	18.8	20.2	29.4	29.6	30.0	
mal-3	19.2	19.2	19.9	18.5	19.9	28.6	28.9		
mal-4	18.5	18.4	19.2	17.9	19.1	28.4			
mal-5	18.3	18.3	19.0	17.8	19.0				
leg-1	25.7	25.6	26.1	25.1					
leg-2	24.5	24.4	25.0						
leg-3	25.5	25.5							
leg-4	24.8								

Table 4.3: Values of Sim_1 computed between malicious and legitimate subsets

Analysing these results brought on Sim_1 we can draw some conclusions. This gives a first insight that malicious sets of domains can be discriminate from legitimate sets using semantic analysis of words embedded in them. Comparing two sets of different types (*mal/leg*) presents the lowest similarity. In addition the comparison of the different subsets provides almost the same scores: $Sim_1(mal, leg) \approx 20$, $Sim_1(leg, leg) \approx 25$ and $Sim_1(mal, mal) \approx 30$ assessing the stability of Sim_1 . We can deduce that the semantic fields of words used in malicious domain names have a smaller scope than legitimate ones because these tend to be more similar ($Sim_1(mal, mal) \approx 30$). This confirms that attackers target specific services when luring users

since they use a limited vocabulary. Finally Sim_1 is well suited to identifying unknown sets of domain names by comparing them to a reference set composed of malicious instances. The score between malicious subsets (mal/mal) is 50% higher than the score obtained while comparing malicious subsets to legitimate subsets (mal/leg). The first proposed metric, being quite basic, gives good preliminary results in malicious domain names identification.

	leg-5	leg-4	leg-3	leg-2	leg-1	mal-5	mal-4	mal-3	mal-2
mal-1	1.034	1.069	1.033	0.989	1.010	1.366	1.332	1.385	1.332
mal-2	1.033	1.053	1.035	0.994	1.013	1.369	1.298	1.365	
mal-3	1.085	1.137	1.089	1.046	1.058	1.388	1.347		
mal-4	1.013	1.041	1.002	0.964	0.964	1.370			
mal-5	1.010	1.051	1.002	0.972	0.972				
leg-1	1.475	1.481	1.489	1.455					
leg-2	1.455	1.452	1.507						
leg-3	1.508	1.525							
leg-4	1.472								

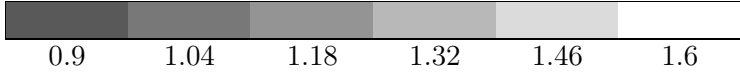



Table 4.4: Values of $Sim_2 \times 10^3$ computed between malicious and legitimate subsets

Table 4.4 shows the values for computing Sim_2 with the same protocol as for Sim_1 . The scores are multiplied by 1,000 for readability purposes. This metric has the same characteristic as Sim_1 while comparing malicious to legitimate subsets. This gives the lowest similarity score ($Sim_2(mal, leg) \times 10^3 \approx 1$). However, this metric tends to show that words embedded in legitimate domains ($Sim_2(leg, leg) \times 10^3 \approx 1.5$) have higher semantic relationship than malicious ones ($Sim_2(leg, leg) \times 10^3 \approx 1.3$). The difference with Sim_1 is that the count of word occurrences is considered. Therefore, even though the semantic scope of words is larger for legitimate domain names (lower Sim_1), some of these are often used explaining a higher value for Sim_2 . Hence, Sim_2 is also a good alternative to distinguish malicious and legitimate domain names. However, an unknown set of domains must rather be compared to a reference legitimate set since the score between legitimate subsets (leg/leg) is 50% higher than the one obtained while comparing malicious subsets to legitimate subsets (mal/leg). The largest difference must be chosen to provide the best accuracy.

Finally, Table 4.5 contains the value of Sim_3 computed between the subsets. After preliminary tests, n was set to 100 in Equation (4.7) for $Disco(w, n)$. This means that the 100 most similar words to w are considered for computation. Sim_3 shows similar scores for mal/mal and leg/leg comparisons ($Sim_3 \approx 0.95$). These scores are approximatively 30% higher than the scores obtained from malicious to legitimate sets comparison ($Sim_3(mal, leg) \approx 0.75$). This metric, which is an approximation of Sim_2 with lower computational complexity ($O(n)$) is able to differentiate malicious domain names from legitimate ones. However, the speed gain is made at the price of decreasing the gap of similarity values between legitimate and malicious sets, dropping from 50% difference for Sim_1 and Sim_2 to 30% for Sim_3 . In addition, almost no difference is noticed in the comparison of legitimate sets (leg/leg) and malicious sets (mal/mal) since $Sim_3(mal, mal) \approx Sim_3(leg, leg)$. This sets a trade-off between speed and efficiency, Sim_1

	leg-5	leg-4	leg-3	leg-2	leg-1	mal-5	mal-4	mal-3	mal-2
mal-1	0.776	0.795	0.793	0.789	0.785	0.955	0.962	0.965	0.975
mal-2	0.782	0.800	0.798	0.797	0.797	0.965	0.968	0.973	
mal-3	0.772	0.796	0.793	0.788	0.784	0.951	0.962		
mal-4	0.783	0.804	0.804	0.800	0.796	0.953			
mal-5	0.769	0.785	0.784	0.782	0.772				
leg-1	0.946	0.948	0.952	0.938					
leg-2	0.915	0.924	0.922						
leg-3	0.936	0.934							
leg-4	0.935								


Table 4.5: Values of Sim_3 computed between malicious and legitimate subsets

and Sim_2 are slower but more accurate than Sim_3 .

4.3.3 Domains Set Size and Composition

These experiments show that the three metrics Sim_1 , Sim_2 and Sim_3 lead to efficiently discriminate malicious from legitimate domain sets. An application of this result is to consider one of these metric and, given a set of labelled malicious or legitimate domain names, identify if an unknown set is either malicious or not by comparing both as depicted in Figure 4.3. Ideally, the tested subset should contain a unique domain name, meaning that we are able to identify the maliciousness of a single unknown domain name extracted from a DNS packet. However, as explained in Section 4.2, a single domain name is composed of few words. It is hard to derive a relevant semantic metric from one domain name and thus domain names must be grouped using some state of the art techniques (IP subnet, autonomous systems, registration address, etc). The clustering technique introduced in Chapter 3 is an alternative solution as well. The initialisation to form clusters of domain names requires to wait for several DNS replies including new domain names before having enough domain names in a group to label it as malicious or legitimate. However, once several initials groups of domain names are formed, if a newly captured domain can fit in an existing group, its identification is fast as it takes the label of the group it fits in. The initialisation process induces a delay in the identification of the first captured phishing domain names. Determining the minimum size of the test domain set is thus paramount to provide the fastest decision and minimize this delay.

Even if the differences in values are proportionally lower for Sim_1 and Sim_2 , Sim_3 is more computationally efficient ($O(n)$) as mentioned in Section 4.2.3. Sim_3 gives approximatively the same result taking a malicious set or a legitimate set as reference model. Hence, Sim_3 is computed between the subset *mal-1*, being the model, and two other subsets, from *mal-2* (*mal*) and *leg-5* (*leg*). We increase the count of domain names in each subset and compute Sim_3 to determine what is the minimum set size. Figure 4.4 depicts the evolution of the value of Sim_3 according to the count of domain names in each subset (*mal* and *leg*) compared to *mal-1*.

We can see that for small size of subsets, from one to five domain names, the curves representing the legitimate and the malicious subset are mingled. From 10 to 60, these are still very

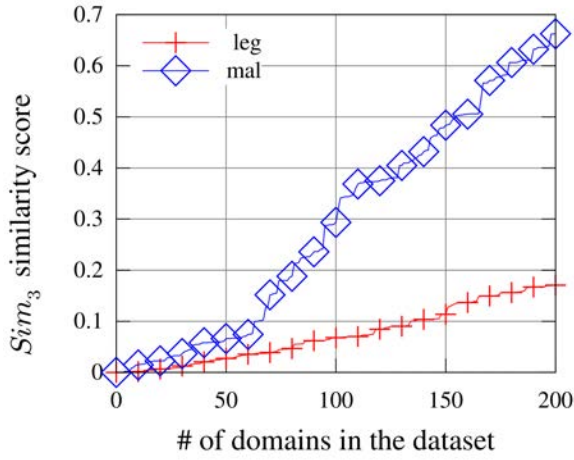


Figure 4.4: Similarity score Sim_3 depending on the count of domain names in the set

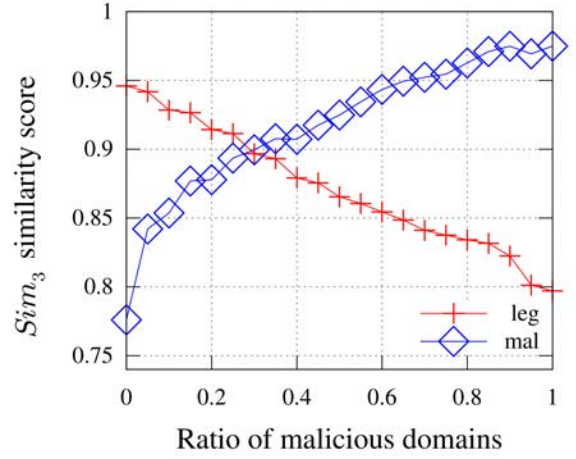


Figure 4.5: Similarity score Sim_3 depending on the proportion of malicious domain names in the set

close but a clear difference is observed and the difference between scores is two fold ($Sim_3(mal-1, mal) \approx 2 \times Sim_3(mal-1, leg)$). Then for more than 60 domain names, a real gap is observed and the difference between the malicious and the legitimate set can be identified with high confidence. Hence, malicious domain names can be identified using a simple threshold value for set composed of more than 10 domain names. However, the larger the unknown set is the more information it contains and the easier is the identification as depicted in Figure 4.4 for sets of more than 60 domain names.

The experiments carried out considered the best case scenario where we are able to form sets of domain names composed only of one kind of domain names, *i.e.* malicious or legitimate. However, forming sets composed only of one kind of domain names is not straightforward. Moreover, sets composition cannot be known in advance since the technique we introduce is applied to unknown set of domains. Hence, we analyse the similarity metric score obtained by comparing a mixed set of domain names to a legitimate and respectively a malicious set of domain names. We took two sets as base models: *mal-1* and *leg-1*. Then, we took the two subsets used in previous experiments *mal-2* and *leg-5* to form a new set of mixed malicious and legitimate domain names. We varied the ratio of malicious domain names in this set from 0% to 100% and computed Sim_3 between this set and *mal-1*, respectively *leg-1*. Figure 4.5 depicts this experiments by showing the similarity score Sim_3 of the mixed domains set when compared to a malicious (mal) and a legitimate (leg) set. The ratio of malicious domain names in the set (coming from *mal-2*) is depicted on the x-axis and increase by 5% from 0% to 100%. 0% corresponds to *leg-5* being compared to *mal-1* and *leg-1* and 100% to *mal-2* respectively.

When compared to the malicious set, the mixed subset gets its Sim_3 score increasing fast as the ratio of malicious domain names increases. With 0% malicious domain names $Sim_3 = 0.776$ but with 5% malicious domain names Sim_3 jumps to 0.842 and with more than 30% malicious domain names in the set, Sim_3 goes over 0.9. This shows that adding few malicious domain names to a set of legitimate domain names impacts heavily the results of semantic similarity comparison. Referring to Table 4.5, we can see that Sim_3 scores between legitimate and malicious subsets are almost always below 0.8 but comparing a malicious set to a set containing 5% of malicious domain names we got $Sim_3 = 0.842$. This similarity score is neither high enough to conclude

that the set is malicious nor it is low enough to conclude that it is legitimate. The conclusion we can draw out of this score is that we got a set of mixed malicious and legitimate domain names.

This property is interesting since it shows that mixed set of domain names can be easily identified as their similarity scores with a labelled legitimate set and a malicious set are not significant enough to mislabel these. This complements domain names clustering of Chapter 3. We saw that malicious domain names can be mixed with legitimate domain names in one cluster when assessing the clustering technique. This happens when few malicious domain names are present in a dataset and therefore cannot form a single cluster. As a result, we had several clusters of legitimate domain names and one containing few malicious domain names along with several legitimate domain names. Using Sim_3 to label the unknown sets of domain names, we can identify set composed only of legitimate domain names and also set composed of mixed malicious and legitimate domain names since high variations in similarity score is observed. Having Sim_3 scores that do not show enough similarity or dissimilarity, reveals mixed set of domain names requiring further analysis to separate legitimate domain names from malicious domain names.

Hence, we showed that we are able to identify both sets composed of malicious domain names and sets composed of legitimate domain names. These sets must be composed of more than 10 domain names to be classified. Finally, sets composed of mixed malicious and legitimate domain names can be discovered since these exhibit similarity scores that does not enable to be classified in either category. In addition, the similarity metrics complement the DNS based clustering technique introduced in Chapter 3 by providing a method to label unknown clusters of domain names.

Conclusion

To complement and deal with some shortcomings of the passive DNS analysis method presented in Chapter 3, we introduced a new technique that can be used to identify clusters of phishing domain names. While the first scheme proposed analysis of DNS packets features, the new one relies on the semantic analysis of domain names gathered using passive DNS monitoring.

We showed that phishing URLs leverage several obfuscation techniques and a global trend is to include in phishing domain names words belonging to limited semantic fields. Hence, we propose to identify these fields and prove that these are different than the ones used in legitimate domain names. Words embedded in domain names are extracted and approaches combining state of the art semantic and natural language processing paradigms are introduced to quantify semantic similarity between set of words and by transition set of domain names. We showed through experiments on ground truth data that the three metrics identify semantic differences between legitimate and malicious domain names. A malicious domain name detection technique based on semantic comparison of unlabelled domain names to a reference set of domain names is proposed and proved efficient.

By design this method copes with the need of large scale passive DNS probes deployment of the method presented in Chapter 3. This method needs only one DNS probe and is suited to local network targeted protection. This also reduces the delay in malicious domain names identification as it does not require several DNS replies for the same domain name to get relevant features. However, differently from the previous technique this one needs a learning stage on ground truth data. This is a drawback because phishing techniques evolve and semantic models of malicious domain names may evolve as well. In addition, the semantic approach cannot identify the maliciousness of single domain names since one domain name does not carry enough information to extract a semantic model. Hence, a prior clustering of domain names must be

performed according to state of the art techniques or through the clustering technique proposed in Section 3 making both methods complementary. We demonstrated that the minimum size of a domain set to analyse is 10 and high confidence is obtained with set composed of more than 60 members. The shortcoming of the semantic approach is that a delay is introduced, due to the prior clustering. This delay only affects though the initialisation process, when the first domain clusters are formed. Afterwards, if captured domain names fit in existing labelled clusters, their identification is straightforward, enabling real-time malicious domains identification. To reduce this delay, the cluster initialisation process using DNS data gathered from several probes can be a solution. However, we have seen in Chapter 3 that geographical dispersion of DNS probes can bias the clustering results. An alternative solution would be to find a way to perform the semantic analysis on single domain names.

A technique to identify single phishing domain names or URL without learning stage would have the benefits of both methods without their weaknesses. This could identify in real time phishing domain names as these are requested by Internet users and protect them from phishing threats. Such an approach is proposed in the next chapter.

Chapter 5

Semantic Based Phishing URLs Rating

Contents

5.1 Intra-URL Relatedness Analysis	86
5.1.1 URL Word Extraction	87
5.1.2 Shortcomings of Word Relatedness Evaluation Tools	87
5.1.3 Search Engine Query Data	89
5.1.4 Feature Computation	90
5.2 Implementation	92
5.2.1 Distributed Word Relatedness Inference	92
5.2.2 Bloom Filter for Features Computation	93
5.3 Phishing URL Detection	95
5.3.1 Dataset	95
5.3.2 Features Analysis	96
5.3.3 URL Classification	98
5.3.4 URL Rating	101

Introduction

Detection methods presented in Chapter 3 and 4 targeted specifically malicious and phishing domain names. Both were proved efficient in identification of phishing domain names and these are complementary as Chapter 3 presents a domain names clustering technique that is used in Chapter 4 as previous processing to quantify the semantic relatedness of unknown sets of domain names with malicious/legitimate sets of domain names. The weakness of the former is the delay of the identification process because clustering based on DNS features requires a lot of DNS data. The semantic analysis needs prior domain names grouping inducing a delay to initialize the method, which is then able to work in real-time. While reducing the amount of information needed with the semantic analysis, as this relies on lexical analysis of domain names and no additional DNS features are required, this method needs sets of known malicious or legitimate domain names as ground truth to operate. Nevertheless, the idea to rely only on domain names to identify phishing is interesting. Passive DNS analysis was used in Chapter 4 for clustering of domain names based on DNS features. However, if a method able to identify single phishing domain names is developed, this can be applied in filtering of phishing emails

as part of a spam detection process at a mail server or in detection of phishing URLs at Web proxies. Besides, browser add-ons can leverage it to identify phishing domain names. An issue to perform such single domain names identification is that some domain names are very short and few meaning can be extracted from them as discussed in Chapter 4. Moreover, existing manual or automated techniques for computing word relatedness are limited to the vocabulary contained in the text samples these are fed with.

To cope with these two issues and perform identification of a single phishing entity, we introduce in this chapter a method for identifying phishing URLs. To extend the analysis and to have more meaningful words to analyse with semantic techniques we propose to use full URLs and not only domain names. We showed in Section 4.1.1 that several obfuscation techniques are applied to the full URL and not only to the domain name, analysing URLs enlarges the detection scope. To analyse the extracted words we use search engine query data rather than existing methods of word relatedness computation to infer semantic similarity between words. We demonstrate that search engine query data is more suited to analyse URL vocabulary than state of the art techniques. The underlying method targets identification of phishing URLs that are based on registered domains (malicious or not) that are not related to their targeted brand. From observations of phishing URLs, we claim that there are few relationships between the registered domain and the rest of the URL because the *mld.ps*, *i.e.* the registered domain, can not be freely defined as presented in Section 4.1.1. However, the words that compose the rest of the URL (lower level domain, path, query) often have many interrelationships and are related to few semantic fields identified in Section 4.1.2. Based on this, we define the concept of *intra-URL relatedness*, which quantifies the semantic similarity between the *mld.ps* and the words composing the rest of the URL. We extract 12 features related to intra-URL relatedness from a single URL which are input to machine learning algorithms to identify phishing URLs. Finally, a *phishingness* score is computed for every single URL, rating the level of maliciousness/legitimacy of the URL. The contributions of this chapter were published in [MFSE14a, MFSE14b].

The rest of this chapter is structured as follows: we start in Section 5.1 by introducing the concept of intra-URL relatedness and describe the method to infer it using search engine query data by computing several features from a URL. Section 5.2 presents the implementation of the feature computation process using streaming analytics tools and Bloom filters for performance. Section 5.3 presents a detailed evaluation of the feature set and its ability to detect phishing URLs on a ground truth dataset of 96,018 URLs.

5.1 Intra-URL Relatedness Analysis

In this section we introduce the core concept of intra-URL relatedness, which is the quantification of the relatedness among the words composing the different parts of a URL and more precisely between the registered domain (*mld.ps*) and the rest of the URL. Reminding the five URL obfuscation techniques used in phishing and presented in Section 4.1.1, this method leads to identify Types I, II, III and IV. The common feature of these obfuscated URLs is that the brand and some related terms are included in the path, the query and low level domains. These terms are related as these have relationships with the targeted brand and have no obvious relation with the *mld.ps* used for phishing. This is the opposite of what happens for a legitimate URL, where all the parts of the URL are related. We first present the word extraction process applied to URLs and then identify the limitations of existing methods for computing word relatedness. We come with a new technique that relies on search engine query data for computing intra-URL relatedness. Finally we introduce a set of 12 features relevant for phishing URLs identification.

5.1.1 URL Word Extraction

The examples of obfuscated phishing URLs from Type I to IV highlight a global characteristic in URL obfuscation, namely that there is no relation between the *mld.ps* and the rest of the URL. To reveal this, we split the URL in the two parts that are presumed to have no relationship: extract the *mld.ps* and separate it from the rest. As the *ps* may be composed of multiple level domains, we use Public Suffix List [pub] to identify it and then retrieve the immediately preceding level domain as the *mld*. For the rest of the URL, a split according to non-alpha-numeric characters is first performed. From extracted parts composed of several words, a dictionary-based word splitter [SH09] is used as previously presented in Section 4.2.1 and Figure 4.2.

Based on this splitting two sets are composed: one, called RD_{url} (for Registered Domain), consists of just two elements: $RD_{url} = \{mld, mld.ps\}$. The other, REM_{url} (for REMaining part), is composed of all extracted words from the URL except *mld.ps*.

As an example, given http://sezopoztos.com/paypalitlogin/us/webscr.html?cmd=_login-run, the following sets are extracted:

- $RD_{url} = \{sezopoztos, sezopoztos.com\}$
- $REM_{url} = \{paypal, it, login, us, web, src, html, cmd, login, run\}$

The *mld.ps* is not split like the other part to keep the *mld* unmodified, which can be composed of several words. Assume a Type III obfuscated URL such as <http://cgi-3.paypal-secure.de/info2/verikredit.html>. With a basic split the word *paypal* would be an element of $RD_{phish} = \{paypal, secure, de\}$.

If <http://cgi-3.paypal.de/info2/verikredit.html> is a real PayPal URL, we have $RD_{legit} = \{paypal, de\}$ and $RD_{legit} \cap RD_{phish} = \{paypal, de\}$. It does not point out the difference between the two URLs (legitimate and phishing). However with the proposed decomposition of *mld.ps* we have the two lists $RD_{phish} = \{paypal-secure, paypal-secure.de\}$ and $RD_{legit} = \{paypal, paypal.de\}$ giving $RD_{legit} \cap RD_{phish} = \emptyset$. Hence our proposed decomposition emphasizes the difference between the two domain names.

Once the two sets are built, the next step is to evaluate the relatedness of their components. It is tempting to compute word similarity or word relatedness with existing tools such as Disco [Kol08, Kol09]. However this tool, even if efficient in most cases and especially in Chapter 4, it is not necessarily suited to intra-URL relatedness computation.

5.1.2 Shortcomings of Word Relatedness Evaluation Tools

WordNet [Mil95] has already been presented as a lexical database containing a collection of English language words. These words are manually linked according to semantic or lexical relations including synonymy, antonymy, entailment, etc. The limitation of this tool is that it is only based on English vocabulary that is likely to appear in an English dictionary, whereas Internet vocabulary includes several different languages and many words that are not contained in dictionaries.

Automated techniques and measures have also been developed to evaluate word relatedness. Latent Semantic Analysis (LSA) proposed by Landauer and Dumais [LD97] or Pointwise Mutual Information (PMI), introduced by Church and Hanks [CH90], then used by Turney [Tur01] based on statistical data from search engine results for the queried words, are examples of these techniques. The Normalized Google Distance (NGD [CV07]) computes the semantic similarity between two words by querying the Google search engine for these words and counts the Web pages where these appear together and individually. Disco [Kol08, Kol09] relies on mutual

Brand	<i>mld</i>	<i>mld.ps</i>
JPMorgan Chase	jpmorganchase	jpmorganchase.com
TAM Airline	tam	tam.com.br
Visa	visa	visa.com
Windows Live	live	live.com
Poste Italiane	poste	poste.it
Co-operative Bank	co-operativebank	co-operativebank.co.uk
Wells Fargo	wellsfargo	wellsfargo.com
Blizzard	blizzard	blizzard.com

Table 5.1: Subset of most phishing targeted brands with *mld* & *mld.ps*

Tool	#<i>mld</i>	%<i>mld</i>	#<i>mld.ps</i>	%<i>mld.ps</i>
WordNet	20	21.3%	0	0%
Disco	23	24.5%	0	0%
Yahoo Clues	87	92.6%	68	72.3%
Google Trends	92	97.9%	76	80.9%
Total	94	-	94	-

Table 5.2: Count of labels matching at least one related word for 4 tools

information evaluation between two words based on corpora as presented in Section 4.2.2. These tools are not suited to evaluate word similarity for URL vocabulary as the corpus of words their relatedness scoring relies on does not contain domain names and most part of abbreviated words found in URLs.

To prove the limitations of these existing tools, we tested whether two of them are able to find related words for a set of labels. WordNet and Disco are chosen since these are the only usable through an API. The testing set consisted of the RD_{url} extracted from a set of 94 URLs from the most often targeted brands. The dataset is built up with data from PhishTank [phi]. Phishing URLs present in PhishTank blacklist are categorised according to the brand these target. 94 brands and associated URLs are present in this list. A subset of this test set is given in Table 5.1 and the result of the test for each tool is given in the two first rows of Table 5.2. The counts of *mld* and *mld.ps* for which the tested tools can give at least one related word are given in absolute value and percentage terms.

Neither WordNet nor Disco performs well on this test set. These only provide related words for less than 25% of *mld* and never match any *mld.ps*, although the brands and tested domain names are well known. In addition for the *mlds* that match a result, it is usually for a brand that is also a meaningful word such as *live* or *visa*. The results of some matching requests are words related to the verb *live*, and not to the online service from Microsoft. The same happens for *visa*, which, in dictionaries and corpora, does not refer to the credit card company. The test proves that current word relatedness tools are not suited to the measure of intra-URL relatedness.

While creating a dedicated corpus to be used with existing methods would be helpful but challenging, word relatedness can be dynamically inferred from search engine query data.

5.1.3 Search Engine Query Data

To perform the evaluation of intra-URL relatedness, we use search engine query data. The reason is that URL obfuscation is a social engineering lure. Phishing URLs target a brand, so clever phishers blend within them the brand and words that Internet users associate with the brand, such as a provided service: *payment* for PayPal. All brands targeted by phishing are popular and provide services on the Internet. People generally use search engines to access these services. When one makes a search, he types some keywords that are typically the brand or the domain name and the service needed like *paypal payment* or *hsbc.com on-line banking*. These words associations reflect the cognitive process of users searching for PayPal or HSBC. Consequently, such words are the ones phishers tend to blend into URLs to trap PayPal and HSBC customers [AR14].

Hence, mining search engine query data for measuring word relatedness is relevant in a phishing context. To achieve this goal, we use search engine query data from two top-ranked search engines: Google and Yahoo. Both offer services, that, given a term, provide some insights on requesting trends concerning it. These services are respectively *Google Trends* [goc] and *Yahoo Clues* [yah].

Google Trends is a public facility of Google that shows the relative interest of Google users over time in a term. A term t is defined by Google as a set of words w . $\{paypal\}$ and $\{paypal, login, secure\}$ are two examples of terms. We will use the same definition in the context of this dissertation. Google Trends depicts the geographic interest for this term and provides related terms according to users' related searches. These related terms were requested by users after they first searched for the given term and matched a restriction. A restriction is the fact that a user does not find the website she is looking for during his first search. Hence, he performs a second search. Google Trends provides the top ten related searches over time as well as the ten rising related searches namely those on which interest has increased recently. One gathers up to twenty related terms for one given term using Google Trends.

Yahoo Clues provides the same kind of services as Google Trends except that it is based on Yahoo search query data. In addition it offers an analysis of characteristics of people searching for the term (gender, age) and it offers an insight into the search flows, the terms requested just before (5 terms) and after (5 terms) a term. Like Google Trends it also provides a set of related searches.

The results of these tools rely on the popularity of requested terms. Both tools provide search insights only if the volume of recorded data is significant enough. Hence, the more popular a term is, the more related terms we can obtain. Combining both sources can provide up to forty related terms to one given term. A result for the queried term $\{paypal\}$ for both tools Google Trends and Yahoo Clues is given in Table 5.3. The ability of these tools to find related words for targeted *mld* and *mld.ps* is highlighted in Table 5.2. Both tools were tested on the same set of terms used for WordNet and Disco and the results are shown in rows 3 and 4. These perform better, with Google Trends being the best at finding related words. However both provide match results for more than 90% *mld* and 70% *mld.ps*, much more than usual similarity evaluation tools tested earlier.

Having a URL url and the extracted sets RD_{url} and REM_{url} , Google Trends and Yahoo Clues are automatically requested for each element of the two sets. We define $Term_w$, as the set of terms resulting from the requests of the word w in both Google Trends (related & rising) and Yahoo Clues (related & requests). A subset of $Term_{paypal}$ is given in Table 5.3 with $Term_{paypal} = \{\{paypal, account\}, \{paypal, login\}, \{paypal, credit, card\}, \dots\}$. We define four sets of words built from a URL url : $REL_{rd}(url)$, $REL_{rem}(url)$, $AS_{rd}(url)$ and $AS_{rem}(url)$.

Google related	Google rising	Yahoo related	Yahoo requests
{paypal, account}	{amazon, paypal}	{bill, me, later}	{paypal, login}
{paypal, login}	{paypal, fees}	{netspend}	{paypal.com}
{paypal, credit, card}	{ebay, uk}	{suntrust}	{paypal, buyer, credit}
{paypal, email}	{paypal, login}	{regions}	{paypal, customer, service}

Table 5.3: Example of term results from Google Trends and Yahoo Clues for {paypal}

$REL_{set}(url)$ consists of all the words *related* to the words of *set* *i.e.*, words included in terms that are results of requests for elements of *set*. Here *set* is either RD_{url} or REM_{url} . The formulas for these sets are given in Equations (5.1) and (5.2).

$$REL_{rd}(url) = \{w \in t \mid t \in Term_{w'}, w' \in RD_{url}\} \quad (5.1)$$

$$REL_{rem}(url) = \{w \in t \mid t \in Term_{w'}, w' \in REM_{url}\} \quad (5.2)$$

$AS_{set}(url)$ is the set of words that are *associated* with the words of *set*, *i.e.* the words that appear in a common single term. Assuming a term t composed of three words $\{w_1, w_2, w_3\}$, there is a symmetric *association* relationship between w_1 and w_2 , w_1 and w_3 , w_2 and w_3 . The two sets $AS_{rd}(url)$ for RD_{url} and $AS_{rem}(url)$ for REM_{url} are defined in Equations (5.3) and (5.4) respectively.

$$AS_{rd}(url) = \{w \in t \mid \exists w' \in RD_{url}, w' \in t, w' \neq w\} \quad (5.3)$$

$$AS_{rem}(url) = \{w \in t \mid \exists w' \in REM_{url}, w' \in t, w' \neq w\} \quad (5.4)$$

These four sets are extracted to quantify the relationship between and inside each set RD_{url} and REM_{url} . *Associated* and *related* terms highlight different level of semantic relatedness. One set consists in words directly used with the requested term (*associated*) which is a stronger relationship than words appearing in related searches (*related*).

Assume the URL http://sezopoztos.com/paypalitlogin/us/webscr.html?cmd=_login-run, Figure 5.1 presents the full process from word extraction to $AS_{rem}(url)$ and $REL_{rem}(url)$ composition based on a subset of $Term_{paypal}$. We obtain:

$$AS_{rem}(url) = \{amazon, fees, login\}$$

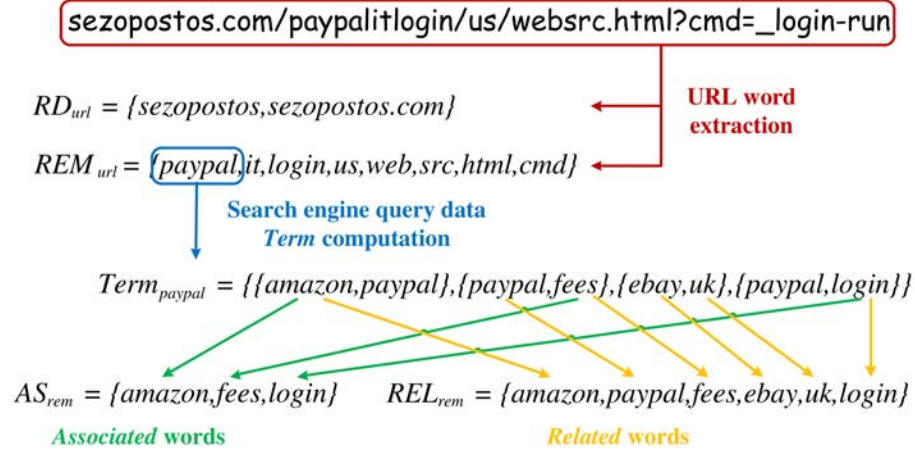
$$REL_{rem}(url) = \{amazon, paypal, fees, ebay, uk, login\}$$

5.1.4 Feature Computation

Based on the sets defined in Section 5.1.3, we introduce 12 features characterising intra-URL relatedness and URL popularity. The popularity criteria is based on the search count for components of a URL: registered domain, *mld*, etc. These features are described in Table 5.4.

The features 1-6 define intra-URL relatedness by computing the Jaccard index. This is done pairwise between the four sets defined in Section 5.1.3: $REL_{rd}(url)$, $REL_{rem}(url)$, $AS_{rd}(url)$ and $AS_{rem}(url)$. The Jaccard index is a long-established metric used to calculate similarity and diversity between two sets A and B . The closer $J(A, B)$ is to 1 the more similar are A and B . It is defined in Equation (5.5).

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \in [0; 1] \quad (5.5)$$

Figure 5.1: Word extraction for *securelogin34ebay.com.my-securephishing-domain.co.uk*

Features	Description
1 $J_{RR} = \frac{ REL_{rd}(url) \cap REL_{rem}(url) }{ REL_{rd}(url) \cup REL_{rem}(url) }$	Jaccard index b/w $REL_{rd}(url)$ and $REL_{rem}(url)$
2 $J_{RA} = \frac{ REL_{rd}(url) \cap AS_{rem}(url) }{ REL_{rd}(url) \cup AS_{rem}(url) }$	Jaccard index b/w $REL_{rd}(url)$ and $AS_{rem}(url)$
3 $J_{AA} = \frac{ AS_{rd}(url) \cap AS_{rem}(url) }{ AS_{rd}(url) \cup AS_{rem}(url) }$	Jaccard index b/w $AS_{rd}(url)$ and $AS_{rem}(url)$
4 $J_{AR} = \frac{ AS_{rd}(url) \cap REL_{rem}(url) }{ AS_{rd}(url) \cup REL_{rem}(url) }$	Jaccard index b/w $AS_{rd}(url)$ and $REL_{rem}(url)$
5 $J_{ARrd} = \frac{ AS_{rd}(url) \cap REL_{rd}(url) }{ AS_{rd}(url) \cup REL_{rd}(url) }$	Jaccard index b/w $AS_{rd}(url)$ and $REL_{rd}(url)$
6 $J_{ARrem} = \frac{ AS_{rem}(url) \cap REL_{rem}(url) }{ AS_{rem}(url) \cup REL_{rem}(url) }$	Jaccard index b/w $AS_{rem}(url)$ and $REL_{rem}(url)$
7 $card_{rem} = REM_{url} $	count of words in REM_{url}
8 $ratio_{Arem} = \frac{ AS_{rem}(url) }{ REM_{url} }$	ratio of associated words for words in REM_{url}
9 $ratio_{Rrem} = \frac{ REL_{rem}(url) }{ REM_{url} }$	ratio of related words for words in REM_{url}
10 $mld_{res} = \begin{cases} 0 & \text{if } Term_{mld} = 0 \\ 1 & \text{else} \end{cases}$	whether there is search engine results or not for the mld of the URL
11 $mld.ps_{res} = \begin{cases} 0 & \text{if } Term_{mld.ps} = 0 \\ 1 & \text{else} \end{cases}$	whether there is search engine results or not for the $mld.ps$ of the URL
12 $ranking$	Alexa ranking for $mld.ps$

Table 5.4: Intra-URL relatedness features description

These six features quantify the relatedness between the two parts of the URL ($mld.ps$ and the rest) through J_{RR} , J_{RA} , J_{AA} and J_{AR} , as these compute Jaccard indexes between sets extracted from different parts (RD_{url} and REM_{url}). These also measure the relatedness between the words contained in each part (registered domain / remaining part of the URL) with J_{ARrd} and J_{ARrem} , since these features are computed from sets extracted within the same part of a URL. Words embedded in the remaining part of phishing URLs tends to have lots of inter-relationships.

Features 7-12 reflect the popularity of a URL and its components with the count of words that compose it ($card_{rem}$) and the count of related and associated words found in search engine query data based on these words with $ratio_{Arem}$ and $ratio_{Rrem}$. These two features are weighted

by $card_{rem}$. Features $mld.ps_{res}$ and mld_{res} represent the popularity of the registered domain by giving boolean values describing whether the $mld.ps$ and mld match results while queried in Google Trends and Yahoo Clues. The final feature (*ranking*) is the ranking of the $mld.ps$ according to the Alexa [ale] website ranking list. If a particular $mld.ps$ is not in the list, the value 10,000,000 is considered.

Features 10, 11 and 12 can be considered as relying on the reputation of a domain name and not on the intra-URL relatedness. Even if features 10 and 11 are new — *ranking* has been used already in state of the art work — we compare in Section 5.3.3 classification results with and without these three features to assess the relevancy of intra-URL relatedness features.

5.2 Implementation

The computation of intra-URL relatedness features requires access to search engine query data. In its current implementation, access to search engine query data is provided through sequential HTTP requests to Google Trends and Yahoo Clues. Four requests are needed for every word we extract from URLs. Set operations are performed on results from these requests to compute the 12 features. Sequential HTTP requests and operations on large sets of words induce a delay in feature computation and for this reason we use distributed streaming analytics tools along with space-efficient data structures to reduce it.

We present in this section the detailed description of the implementation of the features computation process described in Section 5.1.

5.2.1 Distributed Word Relatedness Inference

The bottleneck for features computation is the sequential network communication overhead with the Google and Yahoo servers. However, this bottleneck can be easily removed by leveraging existing Big Data architectures for streaming analytics. For our case, the most relevant architecture is the Storm project [sto], which inherently allows to distribute parallel computations over Storm topologies. Because of the transactional support, we have chosen to use the Trident topologies. Within such a topology, nodes perform a processing logic. Nodes are connected with links that indicate how the data is processed. Data is sent within a *stream* and Storm can distribute the computation along a sequence of nodes. There are two types of nodes: *spouts* and *bolts*. *Spouts* represent the source of data. For our architecture, the *spout* (URL-Spout), depicted in Figure 5.2, is a URL extraction component that extracts URLs.

Each individual URL is tokenized into different words that is sent to several *bolts* (URL-Bolt). This is done using one of the stream grouping method. Several such methods exist and their working depend on how a *spout* decides to split the output to the connected *bolts*. The *Field-grouping* method is used in a first step. This method sends the several words extracted from URLs to different bolts. This ensures that a word occurring in several URLs will be always sent to the same URL-Bolt and therefore a caching strategy can avoid repeating the same requests. Each URL-Bolt will furthermore replicate the input to four Sem-Bolts using the *All-grouping* method. The *All-grouping* method is the replication to all attached *bolts*. Each Sem-Bolt connects to the Google and Yahoo servers and retrieve the list of semantically equivalent words as depicted in Figure 5.2 through the Storm implementation. Finally, the intersection among these needs to be performed. This is done by the Intersection-Bolt using the method further described in Section 5.2.2. Using such an implementation leads to reduce the communication overhead to the single round trip time between our platform and the Google/Yahoo servers instead of sequentially making the needed HTTP requests.

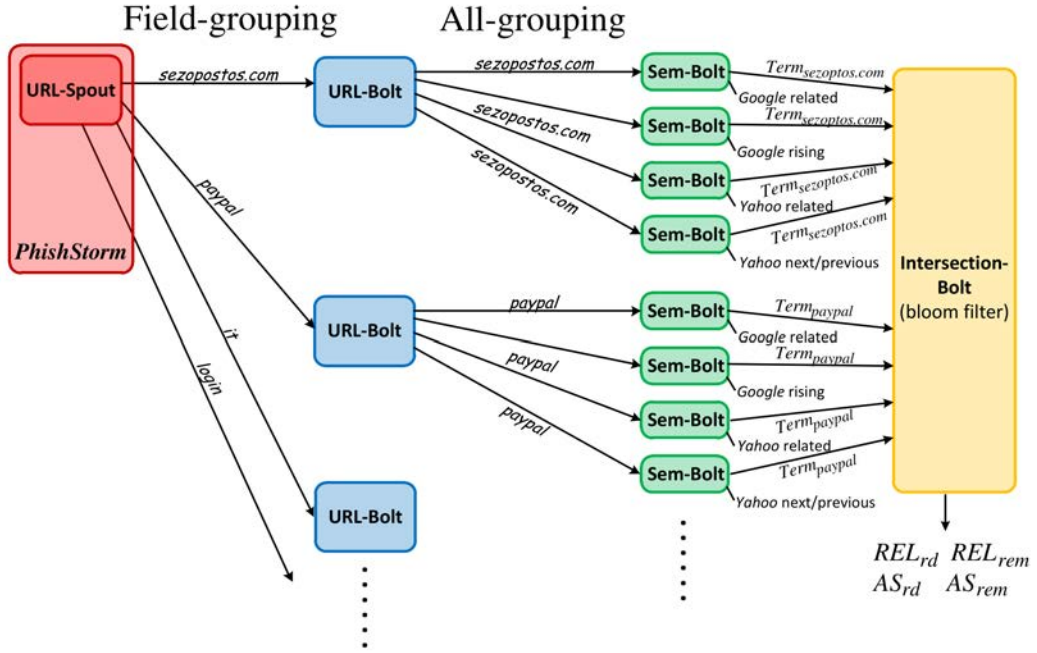


Figure 5.2: Distributed URL processing with Storm topology

Since the Storm topology is defined at compile time, good estimates for the count of URL-Bolts are needed. For this purpose, the histogram depicted in Figure 5.3, showing the distribution of individual words per URL, is used. This histogram depicts the proportion of URLs being composed of n words. This statistic is obtained by applying the word extraction process introduced in Section 5.1.1 on the URL dataset presented later in Section 5.3.1. We can see that most URLs are composed of two to five words. This means that on average the system needs to request the search engine query data for two to five words plus the *mld* and *mld.ps*. Hence to optimize our architecture we set the number of URL-Bolts to seven ($5+2$), in order that an ‘average’ URL can be processed in one round.

This method to gather search engine query data leverages a feature computation technique based on set operations that are space-efficient and computationally-efficient.

5.2.2 Bloom Filter for Features Computation

Features presented in Section 5.1.4 rely mostly on set operations, as for Jaccard Index computation, which requires union, intersection and counting elements operations. Moreover, $REL_{rd}(url)$, $REL_{rem}(url)$, $AS_{rd}(url)$ and $AS_{rem}(url)$, which are the based sets for the features computation, require intersection operation between the several $Term_w$ sets result from querying Google and Yahoo. As a result we implement all the word sets previously defined with an efficient data structure: the Bloom filter [Blo70].

Bloom filters are statistical data structures relying on several hash functions to represent sets of elements. This data structure is represented as a bit array and is subjected to false positives for element lookup, *i.e.* an element identified as being in the set is not necessarily in the set, but an element identified as not being in the set is surely not in the set. Nevertheless, Bloom filters

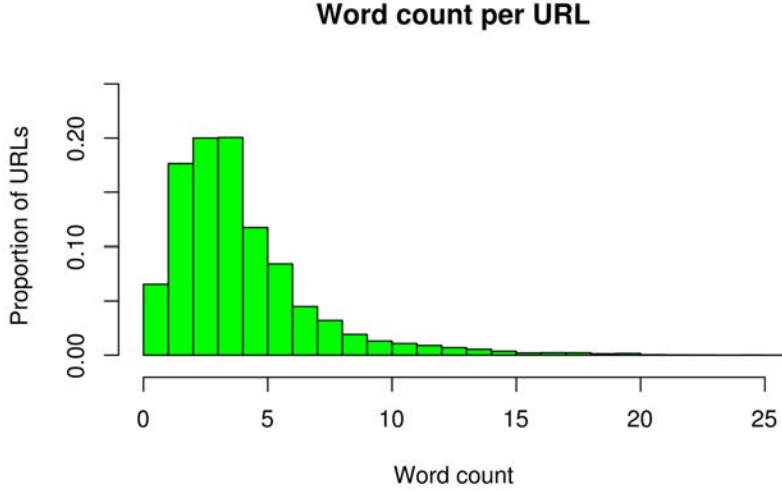


Figure 5.3: Repartition of the count of embedded words per URL

have the strength to be space-efficient and to have constant complexity for lookup and adding elements. This complexity does not depend on the number of elements the Bloom filter contains and is $O(k)$ with k being the number of hash functions. A Bloom filter can be described with two parameters being its size m in bits, which shall be set in advance, and its number of different hash functions k . Assume a Bloom filter containing n elements, the probability of false positive for testing the fact that an element is in the set is expressed in Equation (5.6) with P_{error} .

$$P_{error} = \left(1 - \left[1 - \frac{1}{m} \right]^{kn} \right)^k \tag{5.6}$$

Computing the three needed operations, *i.e.* union, intersection and elements count are also subject to error and to some requirements. To perform set intersection and union with Bloom filters the requirement is that both Bloom filters have the same size (m) and that they share the same hash functions [MNPS07]. The union of two sets consists in a bitwise OR between the two Bloom filters and the intersection of a bitwise AND. For two Bloom filters BF_1 and BF_2 , we have:

- $P_{error}(BF_1 \cup BF_2) < P_{error}(BF_1) + P_{error}(BF_2)$
- $P_{error}(BF_1 \cap BF_2) < \min(P_{error}(BF_1), P_{error}(BF_2))$.

The element count operation of a Bloom filter containing n elements can be approximated by n^* based on the count of bit X set to 1 as shown in Equation (5.7). Hence, Bloom filters offer the required operations to compute the features of intra-URL relatedness in a space-efficient and computationally-efficient way, while introducing approximation.

$$n^* = - \frac{m * \ln \left(1 - \frac{X}{m} \right)}{k} \tag{5.7}$$

As described before the parameters of Bloom filters (size m in bit and number of hash functions k) must be defined in advance. To keep accurate feature values we set the false positive rate P_{error} to 0.0001 (0.01 %) for $REL_{rd}(url)$, $REL_{rem}(url)$, $AS_{rd}(url)$ and $AS_{rem}(url)$. Hence, we can deduce m and k by determining the maximum number of elements that will count these sets. To determine this count we refer to Figure 5.3 showing the word count per URL. We can see that URLs embed from 0 to around 20 words. To have a safety margin we set the maximum word count per URL to 25. Hence, every word embedded in a URL is requested to search engine query data tools to gather up to 40 terms per single word. These terms are composed on average of three words. This gives the maximum number of elements contained in each set: $n = 25 * 40 * 3 = 3000$. The optimal number of bits per element for a Bloom filter is given in Equation (5.8) and for $P_{error} = 0.0001$ we got $bit/element \approx 19$.

$$bit/element = \frac{1}{\ln(2)} * \log\left(\frac{1}{P_{error}}\right) \quad (5.8)$$

$$k = \frac{m}{n} * \ln(2) \quad (5.9)$$

Thus, we have $m = 19 * 3000 = 57000$ bits and we can deduce the number of hash functions to use according to Equation (5.9). We get $k = 13$ and each set $REL_{rd}(url)$, $REL_{rem}(url)$, $AS_{rd}(url)$ and $AS_{rem}(url)$ is set up as a Bloom filter of size 57,000 bits and 13 hash functions. Since these have all same size and same hash functions, these can be compared using union and intersection operations to compute features of intra-URL relatedness.

Having described the technique and the implementation of intra-URL relatedness features computation, we build a ground truth dataset of URLs to test their efficacy through a machine learning algorithm.

5.3 Phishing URL Detection

To assess the ability of the proposed feature set to be used in supervised classification for identifying phishing URLs, we build a test dataset. Unlike for Chapter 4 the dataset used for experiments must be composed of URLs and not of domain names. Hence, a new URLs test set is built and it consists of two sets: one of these is a malicious dataset, the *phishing dataset*; the other is the *legitimate dataset*. Features are extracted from each set and results are compared to show the relevancy of the feature set for discriminating phishing from legitimate URLs. The test set is then used in a machine learning application to detect phishing URLs.

5.3.1 Dataset

Phishing Dataset

We used PhishTank [phi] to build a phishing dataset. PhishTank provides lists of valid and active phishing URLs through its blacklist that is daily updated.

We downloaded this list on a daily basis between October 11th and November 10th, 2012 and built a phishing ground truth dataset of 53,089 unique URLs. URLs consisting only of *mld.ps*, *www.mld.ps* or IP addresses without path or query were discarded because it is impossible to compute the intra-URL relatedness for such URLs, as $REM_{url} = \emptyset$. In addition we partially addressed the identification of such phishing domains in Chapter 4. After this selection we had 48,009 extended phishing URLs in the *phishing dataset* meaning less than 10% phishing URLs discarded.

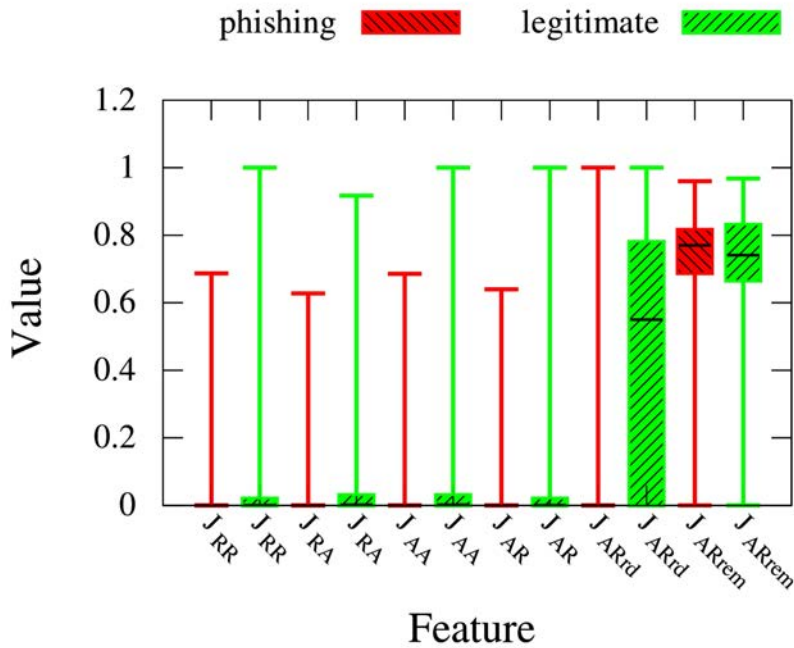


Figure 5.4: Box-and-whisker diagram for Jaccard based features (min/max)

Legitimate Dataset

To provide additional learning instances for legitimate URLs, we selected URLs from the Open Directory Project: DMOZ [dmo]. DMOZ is a directory of the Web containing more than two million URLs classified in several categories. We selected URLs within four categories that are, according to our knowledge of phishing, the most targeted categories: Business, Computers, Games and Shopping. We first discarded URLs consisting only of *mld.ps* or *www.mld.ps*, as for the *phishing dataset*. Then a uniform random selection was made on the rest to keep 48,009 legitimate URLs.

We constructed a balanced dataset (half phishing/half legitimate) of ground truth data composed of 96,018 URLs. We acknowledge that a half/half ratio for phishing and legitimate URLs does not reflect real world repartition. However this dataset is used in Section 5.3.3 to assess the efficiency of the search engine query data and the features extracted therefrom, in distinguishing phishing from legitimate URLs through ten-fold cross-validation. As presented in [FS10, FM14], imbalanced dataset in cross-validation provides misleading results.

5.3.2 Features Analysis

The 12 features described in Section 5.1.4 were extracted from each dataset. The box-and-whisker diagram of Figure 5.4 compares the median, 1st quartile, 3rd quartile, minimum and maximum values for each Jaccard-based feature according to the set it is extracted from. The same statistical values are given in Table 5.5 for the six remaining features as these features are either not defined on $[0, 1]$ or are binary values.

There is no significant difference between the legitimate and phishing datasets for features J_{RR} , J_{RA} , J_{AA} and J_{AR} with equal minimum, 1st quartile and median. The values of these

features are slightly more scattered for the legitimate dataset, reaching higher values. The 3rd quartile is around 0.02 for legitimate URLs whereas it is 0 for the phishing dataset, while the maximum reaches 1 for legitimate, but it does not exceed 0.7 for the phishing dataset. This validates our assumption that the similarity between *mld.ps* and the rest of the URL is more important in legitimate URLs than in phishing ones. In addition we can identify J_{ARrd} as a good discriminative feature a priori. Despite having the same range of values for both kinds of URLs [0, 1] the median and 3rd quartile are orders of magnitude higher for legitimate URLs (0.55/0.78) compared to phishing ones (0/0). Analysing $card_{rem}$ in Table 5.5, we conclude that phishing URLs blend more words within them than legitimate ones. Legitimate *mld.ps* and *mld* logically match more results than phishing ones (mld_{res} , $mld.ps_{res}$) when searched for in Google Trends and Yahoo Clues. Finally the ranking of legitimate domain names is higher than phishing ones since 31,767 legitimate domain names are ranked in the Alexa top one million websites against only 8,081 phishing domain names.

Features	Legitimate dataset					Phishing dataset				
	min	1 st Q	med	3 rd Q	max	min	1 st Q	med	3 rd Q	max
$card_{rem}$	0	1	2	4	20	0	3	5	9	58
$ratio_{Arem}$	0	29.462	93	181.5	6097	0	55.412	114.1	172.22	3122
$ratio_{Rrem}$	0	33.833	92	179	5507	0	59.789	113.5	169.75	2826.5
mld_{res}	0	0	1	1	1	0	0	0	0	1
$mld.ps_{res}$	0	0	0	1	1	0	0	0	0	1
$ranking$	1	6655	82260	10e7	10e7	1	10e7	10e7	10e7	10e7

Table 5.5: Statistical values of features extracted from legitimate and phishing datasets

Nevertheless it is worth noting that some phishing *mld.ps* have a high ranking, namely 1. This top-ranked domain is *google.com*. The reason why such top ranked *mld.ps* is present in the phishing set is that the Google Docs [gooa] facility is used as a support for phishing. Phishers steal personal data by creating in it on-line forms that victims are asked to fill out. The use of well-ranked domain names as a basis for phishing URLs proves that reputation based features can not be sufficient to identify phishing URLs. However, intra-URL relatedness can fill the niche by identifying semantic differences inside these URLs.

To further evaluate the impact of each feature on the classification process, the Information Gain is computed. Assuming S , a set of instances having n features (x_1, x_2, \dots, x_n) and a label l (*phishing/legitimate*), the Information Gain $IG(S|i)$ evaluates the likelihood of deducing l for elements of S given the feature $i \in \{1, n\}$. It is defined in Equation (5.10), based on the entropy of the dataset S : $H(S)$ and the average conditional entropy of S given the feature i : $H(S|i)$.

$$IG(S|i) = H(S) - H(S|i) \quad (5.10)$$

The information gain for the 12 features is given in Table 5.6, ordered by descending value. The higher the information gain value, the more discriminative the feature. All the features bring information for classification, $ranking$ being the most significant feature mainly due to the fact that most phishing domain names have a very poor rank, *i.e.* only 8,081 domain names used for phishing are ranked against 31,767 legitimate. It is followed by J_{ARrd} which contains a large amount of information as shown in Figure 5.4. J_{ARrem} is a significant feature despite no clear difference being brought out in Figure 5.4.

Feature	IG	Feature	IG
<i>ranking</i>	0.388	<i>mld_{res}</i>	0.178
<i>J_{ARrd}</i>	0.286	<i>J_{RA}</i>	0.133
<i>card_{rem}</i>	0.273	<i>J_{RR}</i>	0.125
<i>ratio_{Arem}</i>	0.217	<i>J_{AA}</i>	0.123
<i>J_{ARrem}</i>	0.208	<i>J_{AR}</i>	0.12
<i>ratio_{Rrem}</i>	0.208	<i>mld.ps_{res}</i>	0.07

Table 5.6: Information Gain values for the 12 features

Having established that the feature set is relevant in distinguishing phishing from legitimate URLs, we further assess its efficiency within a machine learning framework.

5.3.3 URL Classification

To automatically detect phishing URLs, we use supervised classification techniques. We build a feature vector matrix from the dataset presented in Section 5.3.1. Each feature vector is composed of 12 elements, namely the 12 features introduced in Section 5.1.4. The predicted variable is 0 for a legitimate URL and 1 for a phishing URL.

This gives a matrix of 96,018 feature vectors representing the 96,018 URLs of the testing dataset. Since there is a wide range of supervised classification algorithms, we assessed our dataset according to several classifiers using Weka [HEH⁺09]. Seven classifiers were tested covering tree-based (Random Tree, Random Forest, C4.5, LMT), rule-based (PART, JRip) and function-based (SVM). The classification was made without parameters tuning through a ten-fold cross-validation as a first step to select the most promising approach. Results for accuracy, true positives and true negatives are given in Figure 5.5 for each classifier. To give additional information about confidence interval of classification results for these classifiers, Table 5.7 shows the median, 5th percentile, 95th percentile and standard deviation (*SD*) values for the *Accuracy* of each classifier out of 100 runs. For sake of clarity we define for URLs:

- Phishing URLs classified as phishing: true positives (*TP*) and $TP_{rate} = \frac{TP}{TP+FN}$
- Legitimate URLs classified as phishing: false positives (*FP*) and $FP_{rate} = \frac{FP}{TN+FP}$
- Legitimate URLs classified as legitimate: true negatives (*TN*) and $TN_{rate} = \frac{TN}{TN+FP}$
- Phishing URLs classified as legitimate: false negatives (*FN*) and $FN_{rate} = \frac{FN}{TP+FN}$
- Phishing and legitimate URLs well-classified: $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$

Among the tested classifiers, SVM yields the worst accuracy (86.31%) while being efficient in identifying legitimate URLs (93.1%). Rule-based classifiers have approximately the same performance, around 90%, with disproportionate true positives and true negatives. The best performers are tree-based classifiers, with Random Forest, correctly classifying 95.22% of URLs, being the best. In addition, Table 5.7 shows that the top performer classifiers give accurate results, having a standard deviation around 0.25% over 100 runs.

Hence, Random Forest is selected for classification. Random Forest [Bre01] is a classification method that creates a multitude of decision trees during training. During prediction, it outputs

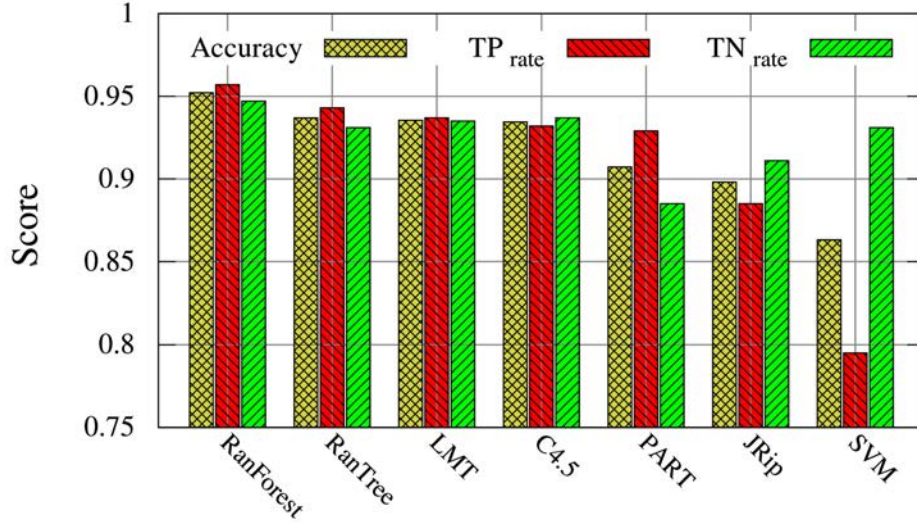


Figure 5.5: Phishing classification results for seven classifiers

Classifier	5 th perc.	median	95 th perc.	SD
RanForest	94.68%	95.22%	95.41%	0.23
RanTree	93.35%	93.81%	94.1%	0.23
LMT	93.10%	93.55%	94.01%	0.27
C4.5	93.01%	93.45%	93.87%	0.26
PART	89.84%	90.62%	91.49%	0.47
JRip	88.32%	89.66%	90.48%	0.66
SVM	85.23%	86.31%	87.53%	0.62

Table 5.7: Confidence interval for classification results

a *hard decision* for the class of an instance as the class that has been predicted by most of the individual trees. However a *soft prediction* can also be deduced from the combination of results given by individual trees. This *soft prediction* is bounded on $[0, 1]$ and gives a confidence score for the prediction. It is then compared to a discrimination threshold to deliver the *hard decision*. Assume two classes 0 and 1, and a default discrimination threshold is fixed to 0.5. If the *soft prediction* is below this threshold, then the *hard decision* for an instance is class 0, otherwise it is class 1. By varying this threshold we can vary true positive and false positive rates.

We tuned the parameters of Random Forest training in order to achieve better classification. After varying the number of trees to be generated during training from 10 to 200, it was set to 100, a value giving good results while keeping fast training of the classifier. The ROC (Receiver Operating Characteristic) curve describing the classification results for the tuned classifier in true positive rate and false positive rate is illustrated in Figure 5.6. The ROC curve corresponds to the variation of true positives and false positives while varying the discrimination threshold from 0 to 1. To minimize the number of legitimate URLs classified as phishing (false positives) we adjust the discrimination threshold from 0.49 (the value giving the best accuracy) to 0.76. This reduces the accuracy from 95.66% to 94.91% but also decreases the FP_{rate} from 4.13% to 1.44%.

Class	Class. as phish.	Class. as leg.	Precision	F-measure	Accuracy
Phishing	91.27% (TP)	8.73% (FN)	98.44%	94.72%	94.91%
Legitimate	1.44% (FP)	98.56% (TN)			

Table 5.8: Detailed classification results for Random Forest (threshold = 0.76)

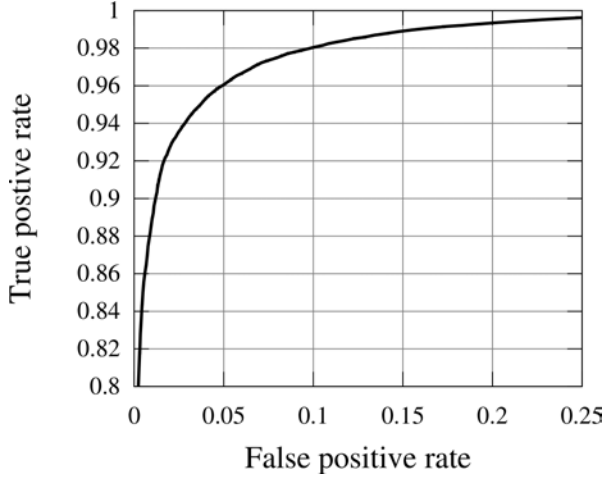


Figure 5.6: ROC curve for Random Forest classification

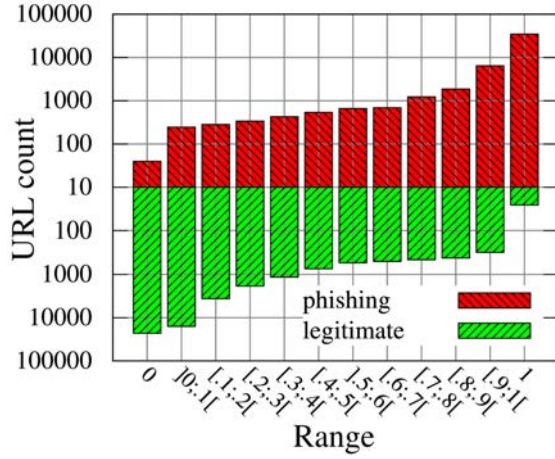


Figure 5.7: Phishing and legitimate URL partition according to rating ranges

The detailed classification metrics for the Random Forest algorithm with a 0.76 discrimination threshold are given in Table 5.8. The two first columns represent the rate of well-classified and misclassified instances for each class: TP_{rate} , FP_{rate} , FN_{rate} and TN_{rate} . The *Precision* corresponds to the ratio of phishing URLs classified as phishing with respect to the total URLs classified, as described in Equation (5.11). The *F-measure* is defined in Equation (5.12) with $Recall = TP_{rate}$.

$$Precision = \frac{TP}{TP + FP} \quad (5.11)$$

$$F-measure = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (5.12)$$

To show the relevancy of intra-URL relatedness features, we classified with different features the set of URLs. Using only features 1-9 for classification yields an accuracy of 93.48% whereas using reputation based features 10-12 only yields 83.97%. While having the best Information Gain as shown in Table 5.6, feature 12 (*ranking*) and other reputation based features are not sufficient to distinguish between phishing and non-phishing URLs alone. However, we show that the introduced feature set yields good results in doing this task. In addition, combining the new features with reputation based features can lead to an improvement in the classification accuracy making this work complementary to the state of the art.

Even though this technique, which gives a *hard decision* for URL class, is proved efficient, correctly classifying 94.91% of URLs with only 1.44% of legitimate URLs classified as malicious, we further leverage machine learning to build a reputation system.

5.3.4 URL Rating

The *soft prediction* value provided by Random Forest is defined on the range $[0; 1]$. In the previous section a discrimination threshold was fixed to give a *hard decision* on the phishingness degree of a URL. However *soft prediction* values are not necessarily uniformly distributed over the range $[0; 1]$ and some sub-ranges of values may be more suitable to providing a highly reliable decision on the phishingness of a URL. Hence, we analysed the *soft prediction* distribution regarding phishing or legitimate URLs. The *soft prediction* range of value $[0; 1]$ is divided in 12 sub-ranges, two being the exact value 0 and 1 and the ten remaining being ranges of length 0.1: $]0; 0.1[$, $[0.1; 0.2[$, ... , $[0.9; 1[$. The *soft prediction* provided by the tuned Random Forest was computed for all 96,018 URLs of the dataset through a ten-fold cross-validation. We counted the URLs having a score belonging to each sub-range. Figure 5.7 depicts this count according to the set (phishing at the top / legitimate at the bottom) the URLs come from. The 12 different sub-ranges are on the x-axis and the URL count is on the y-axis in a log scale centered on 10 and increasing in both directions for each class (phishing up/legitimate down).

We can observe that most URLs are grouped in each extremity of the range and mostly in the sub-ranges $]0; 0.1[$, $[0.9; 1[$ and 1, which contain a total of 80,630 URLs out of 96,018. In addition the middle values of *soft prediction* have few of either kind of URLs, usually less than 1,000. This confirms that the *soft prediction* is not uniformly distributed over its range of definition. Considering the two extreme values, very few phishing URLs (40) have the score 0, whereas 22,863 legitimate URLs do. The same happens for a *soft prediction* of 1 where 34,790 phishing URLs have this score against only 26 legitimate. Given that 0 corresponds to legitimate and 1 to phishing, we are able to classify 60.11% of the dataset (57,719 URLs) with an accuracy of 99.89%. URLs getting a *soft prediction* of 0 or 1 are very likely to be either legitimate or phishing URLs respectively. This proves that some ranges of *soft prediction* values are more suited to making a reliable prediction. If we extend the analysis to the range $[0; 0.1]$, it contains 38,741 legitimate URLs and only 288 phishing ones. The range $[0.9; 1]$ is composed of 41,260 phishing URLs and 341 legitimate URLs. Considering these two sub-ranges, these contain 83.97% of the testing dataset and their components are correctly identified as legitimate or phishing with an accuracy of 99.22%.

The *soft prediction* can be used as a confidence score for a URL. The closer it is to 1, the higher the risk; the closer to 0 the safer the URL. This score can provide a confidence rate to URLs/links for browsing uses. We have demonstrated that such a rating system is reliable in 99.22% of the cases for most of the URLs (83.96%).

While performing our experiments, we timed the process from labels extraction, requesting search engines, features computation to classification decision. With the implementation presented in Section 5.2 leveraging Storm for HTTP requests and Bloom filters for set operations the processing time for the set of 96,018 URLs is 20.6 hours. This gives an average processing time of 0.77 seconds per URL, which is acceptable for many phishing URLs detection applications.

Conclusion

This chapter introduces an efficient phishing URL detection system relying on lexical analysis of URLs. The approach is based on the intra-URL relatedness. This relatedness reflects the relationship among the words blended into a URL and particularly into the part of the URL that can be freely defined and the registered domain. We leverage search engine query data in order to extract 12 features from a URL characterizing its intra relatedness and its popularity. The introduced features are proved relevant in supervised classification on a ground truth dataset of

96,018 phishing and legitimate URLs. The experiment yielded a classification accuracy of 94.91% with a low false positive rate of 1.44%. This experiment was extended to introduce a URL rating system to dynamically compute a confidence rate for URLs. The confidence rate computation on the URL test set is able to correctly identify 99.22% of the legitimate and phishing URLs for 83.97% of the URLs. The implementation of this system relying on real-time streaming analytics architecture and Bloom filters is proved fast to identify phishing URLs (< 0.8 second).

This technique copes with limitations previously observed in Chapter 3 where a large amount of data (DNS data) is needed to identify malicious domain names. Since intra-URL relatedness can analyse and rate single URLs, it copes with the limitations of the technique presented in Chapter 4 where previous clustering of domain names has to be performed before lexical analysis to identify phishing domain names. Computation of intra-URL relatedness provides a real-time phishing detection technique relying only on single URL lexical and semantic analysis. This can be implemented on any end user machine due to on-line availability of search engine query data through tools such as Google Trends and Yahoo Clues.

However, this technique has some limitations: it is not applicable to all types of obfuscated URLs. URLs composed of only a malicious domain name, URLs based on shortening services or URLs algorithmically generated can bypass the detection technique. This kind of URLs and malicious domains are however mostly used in botnet communication (C&C) or spamming activities [YRRR12]. Such activities do not rely on a social engineering process as phishing does. The main part of URLs used for phishing are meaningful and composed of many terms [AR14], this is why our technique is relevant in a phishing context. Another limitation of the implementation is that data publicly available through Google Trends and Yahoo Clues is limited. For each requested term only the ten related most popular terms are returned by these tools. Related terms, which are less requested by search engine users do not appear in results while being relevant for intra-URL relatedness computing. For the same reason, some unpopular terms blended in URLs do not match any results. The reason is that Google and Yahoo do not provide data that is not representative enough, *i.e.* for terms that are not requested enough by their users. These facts limit the accuracy of intra-URL relatedness computing and is one of the reason why additional features such as *ranking* are included in the feature set. A full access to Web search logs would highly improve the relevancy of intra relatedness metrics and, as a result, the classification performance. Despite this limited access to data, the results presented in this chapter provide strong hints regarding the relevancy of using search engine query data for phishing detection.

In this part we presented three techniques to identify phishing URLs and domain names. We first used DNS data analysis before exploring lexical and semantic analysis. Along the three previous chapters we improved the different phishing detection techniques to reduce the delay in identification and improve the accuracy of phishing detection. Some interesting conclusions arose while observing the composition of phishing domain names and phishing URLs, namely that these have a predictable nature in term of words and semantic fields that are used in their composition. As a result, in order to improve the efficacy of techniques to cope with phishing we further explore the way to switch from phishing detection techniques to phishing prediction techniques by leveraging the predictable character of phishing domain names composition.

Part III

Semantic Based Phishing Domain Names Prediction

Chapter 6

Semantic DNS Probing

Contents

6.1 Smart DNS Probing	106
6.1.1 Hostnames Composition Schemes	106
6.1.2 System Overview	108
6.1.3 Smart DNS Brute Forcer	108
6.2 Semantic Discovery of Subdomains	110
6.2.1 Similar Names	110
6.2.2 Incremental Discovery	112
6.2.3 Splitter	112
6.3 DNS Probing Evaluation	113
6.3.1 Methodology	113
6.3.2 Exploration Parameters	114
6.3.3 Performance Evaluation	116

Introduction

The Domain Name System is critical for the operation of the Internet since it is mainly used for locating hosts based on human readable names. Service availability is improved by dynamic reallocation to other machines without changing the DNS name. However, as seen in Chapter 2, this mechanism is also employed by attackers to improve the robustness and the efficiency of their attacks [PCDL09]. Phishing attacks leverage maliciously registered domain names. Previous chapters of this document introduced new techniques to detect phishing domain names and URLs. These chapters disclosed as well the semantic characteristics of phishing domain names and URLs. Hence, we will leverage these characteristics to explore new methods to predict domain names used for malicious activities.

As a first step to analyse the predictable character of the DNS we focus on DNS probing, *i.e.* guessing domain names that are registered and used in the Internet. In our case, DNS probing does not consist in the discovery of Fully Qualified Domain Names (FQDNs) but rather consists in discovering the several subdomains of a given domain name. This provides an alternative to IP address scanning to discover hosts of an organization. IP address scanning is tedious and easily detectable. However, since DNS requests are made to intermediate DNS servers, this

alternative is not detectable. Moreover, Kamra *et al.* show in [KFMK05] that DNS scanning allows to identify potentially vulnerable IPv6 addresses faster than with classical IP scanning. This is due to the large address space of IPv6 and DNS scanning is used in [KFMK05] to ease the spread of worms in the IPv6 Internet. DNS probing is also used by attackers to discover the networking organization, as well as potential vulnerable hosts of a given network. Usual methods of DNS probing [dns, fie] rely on dictionaries composed of widely used hostnames such as common services: FTP (File Transfer Protocol) or SSH (Secure Shell). Then DNS probing of a domain name *example.com* consists in requesting domain names such as *ftp.example.com*. However, these existing dictionary based techniques are limited to the words contained in their dictionary. A proper DNS configuration can easily prevent a network to be probed by these techniques. In addition, these probing techniques cannot be used to predict malicious domain names due to their limited scope and flexibility.

DNS has a role to provide a semantic meaning to something that does not have a meaning by associating domain names to IP addresses. Moreover, observations of subdomains naming schemes show that human based domain names usually follow semantic rules. Hence, we introduce in this chapter a DNS probing technique relying on semantic analysis of domain names. This includes the word semantic as well as the numerical semantic (series of numbers) of domain names. A first DNS probing solution based on natural language processing technique is presented to provide an initial set of subdomains for a given domain name. Based on small subsets of already known subdomains we introduce three extension strategies relying on semantic to extend the set of known subdomains of a given domain. This approach improves results provided by state of the art techniques and is complementary to them. We assess the relevancy and efficacy of semantic DNS probing on a set of 24 well-known domain names, discovering hundreds of subdomains for them. The contributions of this chapter were published in [MFWE12].

The rest of this chapter is organized as follows: We first motivate our work by presenting hostnames composition schemes before giving a global overview of the semantic DNS probing technique we develop, in Section 6.1. In Section 6.2, we introduce the three semantic modules developed to probe domain names. We assess semantic DNS probing in Section 6.3, through several experiments evaluating improvements over three well-known state of the art DNS probing tools.

6.1 Smart DNS Probing

DNS probing is the discovery of the several subdomains *ld* of a given registered domain *mld.ps*. To confirm the validity/existence of *ld*, one makes a DNS request for the domain name *ld.mld.ps* and analyses the response flag of the DNS reply. If the response flag is *NOERROR*, the label *ld* is a subdomain of *mld.ps*, if it is *NXDOMAIN*, *ld* is not a subdomain of *mld.ps*. In this section, we present our global approach to perform automated DNS probing based on observations of the domain names composition. We present a technique to discover an initial list of subdomains on which a semantic extension can be applied to quickly discover new subdomains.

6.1.1 Hostnames Composition Schemes

Accessing machines on a network is made using DNS since remembering tens of IP addresses corresponding to several machines is infeasible. Remembering names like *ftp* or *mail* for accessing FTP and respectively mail servers with domain names such as *ftp.mynetwork.com* is easier than remembering an IP address such as *10.3.6.133* for instance. Machine names (hostnames) on a network are given by network administrators and they are the people who mainly use these

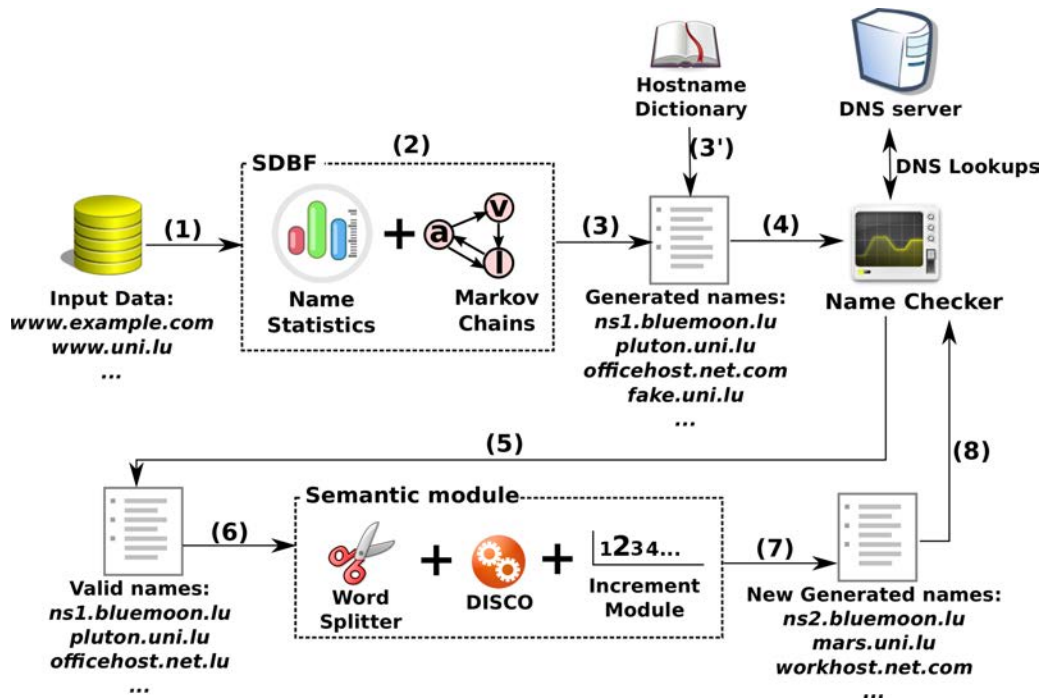


Figure 6.1: Semantic DNS probing system overview

domain names. Hence, the naming of network machines follow different schemes that we have observed on passive DNS traces and specific networks we have access to. Some of the naming schemes observed include:

- Location of a machine in a building: *room201-pc1*, *office102-server*, etc.
- Service provided by a machine: *www*, *ns*, *ftp*, *smtp*, etc.
- Abstract relationship with provided service: *hermes* or *mercury* (messenger of gods) for mail servers for instance.
- Network administrator interests:
 - Movie/sitcom characters: *bart*, *marge*, *homer*, *barney*, etc.
 - Planets: *mars*, *venus*, *jupiter*, etc.
 - Animals: *dog*, *cat*, *mouse*, etc.
 - Cities: *roma*, *milan*, *paris*, *london*, etc.
 - Etc.

The listed naming schemes highlight semantic similarities between hostnames that can be infer with semantic metrics. By observing these lists, knowing some initial elements, it is easy to discover new ones. The discovery of subdomains of a domain name consists in the discovery of these hostnames. Hence, to develop a method to automatically discover related words to existing subdomains is interesting for DNS probing. Such a technique can reduce the number of DNS requests performed by dictionary based DNS scanning tool such as DNSenum [dns] or Fierce [fie]. It also allows to probe subdomains that are not contained in basic dictionaries.

6.1.2 System Overview

To discover the most subdomains ld of a domain name $mld.ps$, we introduce an approach to cope with the limitations of dictionary based DNS scanning techniques. The global overview of the subdomain generation system is given in Figure 6.1. The two main steps of the subdomains generation are:

1. The generation of an initial set of subdomains of a domain name that can be based on dictionary (3') or on an automated generation technique called SDBF (3). This initial set of subdomains is checked making DNS requests (4) to confirm domain names existence in order to build a set of valid subdomains (5).
2. The initial set of valid subdomains is submitted to the semantic extension module (6) that generates new labels likely to be other subdomains (7). These are further checked making DNS requests (8) to be added to the set of initial valid names (5). Steps (5) to (8) can be repeated several times to extend the set of known subdomains based on newly discovered ones.

The generation of an initial list of subdomains for a given domain name can be done using dictionary based techniques (3') such as DNSenum [dns] or Fierce [fie]. However, these lists of subdomains already contain many related labels limited to few semantic fields, but many semantic fields that are not usually used for naming hostnames are not contained in these lists. Hence, using a flexible subdomain generation technique to provide a first set of existing subdomains is more interesting.

6.1.3 Smart DNS Brute Forcer

SDBF (Smart DNS Brute Forcer) is a DNS probing technique introduced by Wagner *et al.* in [WFS⁺12]. This technique consists in a learning stage of domain name composition based on a set of existing domain names using natural language processing techniques. Then, based on the learned model, SDBF can generate several domain names and subdomains depending on some parameters. This is an interesting technique to build an initial set of varied existing subdomains for the semantic DNS probing system.

SDBF Features

The main features of SDBF are based on linguistic parameters learned from a set of domain names. Having a set of existing domain names $N = \{n_1, \dots, n_n\}$ as input list – (1) in Figure 6.1 – we extract sets of labels/characters characterizing domain names composition:

- $L_i = \{l_1, \dots, l_o\}$: the set of labels used as level domains at level domain $i \in \{1, \dots, m\}$.
- $C_i = \{c_1, \dots, c_p\}$: the set of individual characters used at level domain $i \in \{1, \dots, m\}$.
- $G_{i,x} = \{x_1, \dots, x_q\}$: the set of characters following character $x \in C_i$ in labels at level domain $i \in \{1, \dots, m\}$.

The statistical features characterizing domain names composition include:

- $\#dlen_j$: the count of domain names $n \in N$ composed of j level domains.

- $\#ldlen_{i,j}$: the count of different labels $l \in L_i$ composed of j characters and present at the i^{th} level domain of a domain name.
- $\#ldfirstchar_{i,j}$: the count of labels $l \in L_i$ present at the i^{th} level domain and starting with the character $j \in C_i$.
- $\#ngram_{i,j,k}$: the count of times that a character $j \in C_i$ is succeeded by $k \in G_{i,j}$ in a label $l \in L_i$ present at the i^{th} level domain of a domain.

These features are transformed into distributions as follows – (2) in Figure 6.1 – to be further used in the generation model:

- the distribution of domain lengths $j \in \{1, \dots, m\}$ (in level domains):

$$distdlen(X = j) = \frac{\#dlen_j}{\sum_{k=1}^m \#dlen_k} \quad (6.1)$$

- the distribution of label lengths $j \in \{1, \dots, m\}$ (in characters) at a given level domain l :

$$distldlen_l(X = j) = \frac{\#ldlen_{l,j}}{\sum_{k=1}^m \#ldlen_{l,k}} \quad (6.2)$$

- the distribution of first characters $j \in C_l$ for labels at a given level domain l :

$$distldfirstchar_l(X = j) = \frac{\#ldfirstchar_{l,j}}{\sum_{k \in C_l} \#ldfirstchar_{l,k}} \quad (6.3)$$

- the distribution of characters $j \in G_{l,c}$ following the character $c \in C_l$ at level domain l :

$$ngram_{l,c}(X = j) = \frac{\#ngram_{l,c,j}}{\sum_{k \in G_{l,c}} \#ngram_{l,c,k}} \quad (6.4)$$

These four distributions summarize the composition characteristics of domain names. These are used to generate new domain names following the composition rules of existing domains.

Name generation

Once the system is trained, SDBF can generate new names to probe. It first defines how many level domains the domain will have. To achieve this, a random number following the distribution of $distdlen$ is generated. As SDBF is designed to be highly customizable, this value can also be set by the user. The same process is applied to determine the length of labels in characters for each level domain l to generate: $distldlen_l$. Again, this value can be fixed instead of using the distribution metric. Finally, for a label with a given length k , the first character is generated following the distribution $distldfirstchar_l$, and the remaining $k - 1$ characters are generated by applying the characters transitions given by $ngram_{l,c}$.

Because it is common usage to scan a domain name, SDBF allows to set fixed part of a domain name in order to generate only the required level domains of a domain name. For example, the objective may be to discover several domain names providing *www* services with the rule *www.*.**. Another rule can be to discover different name servers in Luxembourg with the rule *ns.*.lu*. Finally, the goal can be to discover all subdomains/hostnames of a given domain

names such as *uni.lu* with the rule **.uni.lu*. This is the main application of SDBF and the one we use it for. This corresponds also to the dictionary approaches which are fed with a set of widely used hostnames (www, ns, ftp, smtp, etc.) and just concatenate each hostname with a domain name.

Once names are generated, (3) in Figure 6.1, their existence is checked by the name checker (4), which makes DNS requests. This formally corresponds to a function $valid(D)$ returning the valid domain names of a set D , *i.e.* those that exist.

Because SDBF is designed to generate the most likely domain names, some of them may be generated several times as the generator does not have memory of already generated domain names. Therefore, the generator leverages a Bloom filter [Blo70] in order to discard already generated subdomains. When probing a domain name, the user may require to generate millions of subdomains. Thus, having a scalable structure like a Bloom filter is necessary to avoid to probe twice the same domain name with DNS requests. Although a name cannot be probed twice, some domain names not being probed can be discarded. This is due to the false positive rate introduced by using Bloom filters as presented in Section 5.2.2.

A major advantage of Bloom filters is that they can be easily distributed among different machines. Therefore, SDBF can be run in parallel, probing from multiple locations without probing the same domain name multiple times. Moreover, bias due to language specificities might be discarded since participating machines could be located in different countries, having each their own local database of features. In addition, large enterprise networks deploy multiple authoritative servers for reliability reasons. By doing parallel and iterative queries to these servers from multiple locations within the network, differences can reveal configuration errors.

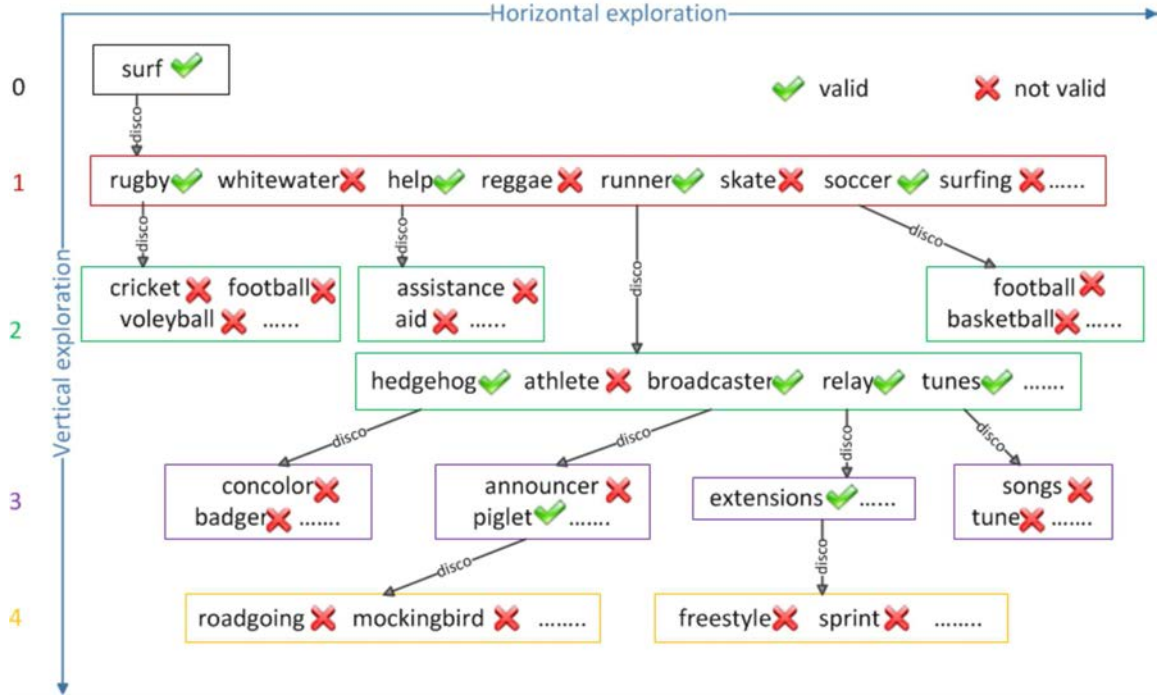
6.2 Semantic Discovery of Subdomains

As illustrated in Figure 6.1, the semantic module takes as input a set of subdomains, for which the validity has been checked (5). The goal is to extend this set of discovered subdomains by analyzing individual labels. There are three modules, the similar names module (DISCO) and the incremental module that can be used individually or combined together, whereas the splitter module is an optional preprocessing step.

6.2.1 Similar Names

The first semantic extension aims to discover names that are similar and related. These are distinct notions [BH06]. Similarity refers to words having a close meaning (for example, notebook and laptop). Semantic relatedness refers to words sharing the same semantic field like mars and venus, which are different planets. As claimed by Kilgariff *et al.* [Kil03], these usual notions imply a manual analysis to establish relationships between words. This limits the applicability and the extension to other languages or semantic fields. Hence, as done in Chapter 4, we use DISCO [Kol08, Kol09], which is based on an efficient and accurate method for approximating automatically these two notions within one metric.

Assume a probed domain name *mld.ps* for which we already found an existing subdomain being a label l , *i.e.* the domain $l.mld.ps$ exists. The objective is to find semantically similar labels l' that may be subdomains of *mld.ps*. The semantic exploration has two dimensions. The first one is the horizontal exploration, which is limited by lim_h . This consists in the selection of the lim_h most similar words to l according to DISCO. These lim_h similar words are the ones that will be tested as potential subdomains of *mld.ps*. These words correspond to $Disco(l, lim_h)$ defined in Section 4.2.2. This result forms a new set of labels $Expl_H(l, lim_h) = Disco(l, lim_h)$

Figure 6.2: Horizontal and vertical exploration for *surf.apple.com*

which is tested by the name checker (Figure 6.1) by concatenating each element with *mld.ps*. Each label of $Expl_H(l, lim_h)$ being confirmed as a subdomain of *mld.ps* is added to the set $Valid(Expl_H(l, lim_h))$.

The second exploration dimension is the vertical dimension. It looks for additional similar names starting from already found subdomains, *i.e.* the names that will be requested in DISCO with $Disco(l, lim_h)$ to find additional candidates. The limit of the vertical exploration is set by lim_v and consists in the number of times the previous process is performed with the newly discovered valid subdomains contained in $Valid(Expl_H(l, lim_h))$. This is defined in Equation (6.5).

$$Expl_V(l, lim_v) = \begin{cases} \emptyset & \text{if } lim_v = 0 \\ \bigcup_{l' \in Expl_H(l, lim_h)} Valid(Expl_H(l', lim_H)) & \text{if } lim_v = 1 \\ \bigcup_{l' \in Expl_V(l, lim_v-1)} Valid(Expl_H(l', lim_H)) & \text{otherwise} \end{cases} \quad (6.5)$$

In order to reduce the search space, only validated labels are considered for further extensions, as noticed by the use of *Valid* in Equation (6.5). The vertical exploration stops when it reaches lim_v recursions or when no new correct labels are found. So, lim_v does not need necessarily to be manually set.

The vertical exploration is actually a recursive process, which is highlighted in Figure 6.1 by the loop (5)-(6)-(7)-(8). Figure 6.2 represents a subset of a real probing campaign with $lim_h = 50$ and $lim_v = 5$. The starting label is *surf*, which is a subdomain of *apple.com*. The horizontal exploration reveals unsuccessful (*surfing*, *skate*) and successful (*rugby*, *soccer*...) labels *i.e.* labels not being, and respectively being subdomains of *apple.com*. Then, the vertical exploration entails a horizontal extension for each of the successful labels by repeating the process of horizontal

exploration. Here the vertical exploration is operated for four recursions. The fifth recursion set with $lim_v = 5$ is not operated since no labels of the fourth recursion are subdomains of *apple.com*.

6.2.2 Incremental Discovery

In some cases, machines and services are replicated or are named according to office/room location in a company. This is represented by hostnames such as *pc1-room102*, *mail2*, etc. Assuming that one of these names has been discovered, the others can be generated by finding out the numerical components of a level domain. New potential subdomains can be generated using the two following heuristics:

- Heuristic $H_{increment,1}$, tests all possible numerical values (including \emptyset) for each individual digit of a known subdomain. This limits the exploration to numbers of the same or smaller power of ten. For example, numbers from 0 to 9 and from 00 to 99 are tested for a known subdomain like *pc13*.
- Heuristic $H_{increment,2}$, tests n randomly generated values. The random generation process first fixes the range of the generated value by fixing the number of digits i according to the geometric distribution $f(X = i) = (1 - \alpha)^i \times \alpha$. This distribution favors smaller number to bigger ones. Then a uniform random function generates a number composed of i digits.

Both heuristics can be combined, the first one to test closest numerical values and the second to extend the exploration to less probable values.

6.2.3 Splitter

Labels of domain names can be composed of several words like *linuxserver* or *linux-server*. Applying DISCO on such names does not provide any results since it works only for single words. Therefore, the labels have to be divided automatically in advance to extract every word. Using a list of separating characters, as for instance “.” or “_”, works to split some labels. However, it is too restrictive and we refer, as in previous chapters, to the word segmentation method described in [SH09]. This process can extract every meaningful word blended in any labels. DISCO can then be applied on each element of the label to create new related labels. For instance, for *linux-server*: *apple-server*, *windows-server* or *apple-terminal* can be generated using this process.

Additionally, the splitter module can also discover the incremental part of a domain name. A label like *computer23* is split as *computer* and *23* according to the splitter. These two extracted labels consist in the fixed part and respectively the part to be explored by the incremental discovery process. This split process also reveals non numerical increments, as observed in our database for some subdomains such as *servera* or *serverb*. The splitter separates *server* from the incremental characters a and b . This can further be incremented using ASCII codes.

The similar names module, the incremental discovery module and the splitter module explore the different methods to find related subdomains based on existing ones. If a domain name follows one of the composition scheme described in Section 6.1.1, the introduced modules cover the range of techniques to probe it. We show in the next section that several popular domain names follow these composition schemes by assessing the efficacy of semantic DNS probing.

6.3 DNS Probing Evaluation

In this section, we assess the efficacy of semantic based DNS probing. We compare the introduced technique to three existing DNS probing techniques on a set of well-known domain names. The three components of the semantic module are used separately and combined to test the efficacy of each one. We also analyse the overhead of the semantic DNS probing compared to existing solutions to prove that in most cases it discovers more subdomains with fewer operations.

6.3.1 Methodology

Protocol

Assume a domain name *mld.ps*, the evaluation methodology consists in using several DNS probing tools to generate potential subdomains and test their existence making DNS requests. Dictionary based techniques probe domain names by iterating over a set of labels $\{l_1, \dots, l_n\}$, to form the hostname $l_i.mld.ps$. For the evaluation, three state of the art tools are considered: Fierce [fie], DNSenum [dns] and SDBF [WFS⁺12]. Both Fierce and DNSenum are dictionary based probing techniques included in Backtrack, a Linux distribution designed for digital forensics and penetration testing. The hostnames dictionary from Fierce includes 1,895 words, whereas the one from DNSenum includes 266,930 entries to test. SDBF, being a flexible domain name generation tool, was configured the same way as Fierce and DNSenum. SDBF generates subdomains of a domain names *mld.ps* by fixing the generation rule **.mld.ps* meaning that it generates domain names of length three level domains with the two last being already fixed. Since SDBF is able to generate as many domain names as needed, we limited the number of generated domain names to 266,930, which is the number of elements in DNSenum dictionary. A detailed analysis of these three tools performance is described in [WFS⁺12]. The main result is that SDBF and Fierce provide the best results, but all of them are complementary, generating domain names that the others do not find.

Having three sets of already known subdomains of a domain name generated by each tool, new ones are generated and probed using the semantic module and one of the following strategies:

- Similar names (DISCO)
- Similar names (DISCO) + Splitter
- Similar names (DISCO) + Incremental discovery
- Similar names (DISCO) + Splitter + Incremental discovery

Except if mentioned, the last description including all the semantic extension is the technique applied in experiments. The original databases provided with the semantic tools [Kol08, SH09] are used to train the similar names module and the splitter module.

Dataset

The domain names used to test the DNS probing comes from the top 50 websites ranked by Alexa [ale]. However, only 19 domain names out of these 50 were considered for two reasons: First, similar domain names with different TLDs have been discarded, since these have often similar subdomains and this biases the evaluation. For instance, *google* has twelve domain names with different TLDs in the top 50 Alexa. Evaluating the semantic exploration on these domain names gives approximatively the same results and the good results we got by probing *google.com*

would increase the average performance of semantic DNS probing. Second, we discarded domain names performing DNS wildcarding. DNS wildcarding consists, for an authoritative DNS server responsible for a domain name d , in always replying positively (NOERROR) to any DNS request for a domain name $l.d$. Hence, any label l seems like being subdomains of d while it is actually not. We inferred wildcarding domain names by sending several DNS requests for algorithmically generated subdomains that are unlikely to be registered. If we got only NOERROR DNS replies, the domain name was considered as wildcarding domain.

We also added to this set of 19 initials domain names a set of five popular luxembourgish domain names including the one of our university (*uni.lu*) which is probed from inside the network. These 24 domain names are presented in Table 6.1.

Metrics

For the evaluation, we define $Init_i(dom)$ where $i \in \{SDBF, DNSenum, Fierce\}$, the initial set of discovered subdomains for a domain name dom . We also define the set of all discovered subdomains $Init_{overall}(dom)$ of dom in Equation (6.6).

$$Init_{overall}(dom) = Init_{SDBF}(dom) \cup Init_{DNSenum}(dom) \cup Init_{Fierce}(dom) \quad (6.6)$$

We define New_i where $i \in \{SDBF, DNSenum, Fierce, overall\}$ as the set of newly discovered subdomains using the semantic module. New_i corresponds to the valid newly generated subdomains based on the initial set $Init_i$. Assuming $|S|$ as the cardinality of a set S , the improvement brought by the semantic module corresponds to the ratio of newly discovered subdomains regarding the count of initial subdomains already known. It is defined in Equation (6.7). A significant value of $\%Imp_i$ shows that semantic DNS probing is able to find new subdomains that previous methods are not able to find. It shows as well the degree of complementarity between methods. $\%Imp_i$ is the main evaluation metric used in the following experiments

$$\%Imp_i = \frac{|New_i|}{|Init_i|}, i \in \{SDBF, DNSenum, Fierce, overall\} \quad (6.7)$$

6.3.2 Exploration Parameters

The exploration parameters lim_h and lim_v define the count of labels that are generated by the semantic module. The count of generated labels has an impact on the count of subdomains we can discover and also on the time taken for the DNS probing since for every generated label we perform a DNS request. Hence, we first evaluate what are the optimal values for these two parameters.

Horizontal search:

The horizontal search, limited by lim_h , consists in the count of relative words we get from DISCO. Setting lim_h limits the exploration to the top lim_h most similar words, as presented in Section 6.2. On the one hand, we can assume that the more words we test, the more subdomains we can find. On the other hand, each DNS request is time consuming and this may lead to the detection of the DNS probe. Hence, we analyse in Figure 6.3 the percentage of newly discovered subdomains depending on $1 \leq lim_h \leq 60$. The plotted metric, $ImpH_{i,h}$, represents the proportion of newly discovered subdomains when $lim_h = h$ compared to $lim_h - 1$, *i.e.* the improvement brought by probing one more subdomain. Having $\%Imp_{i,h}$, the value of $\%Imp_i$ when $lim_h = h$, we define $ImpH_{i,h}$ in Equation (6.8).

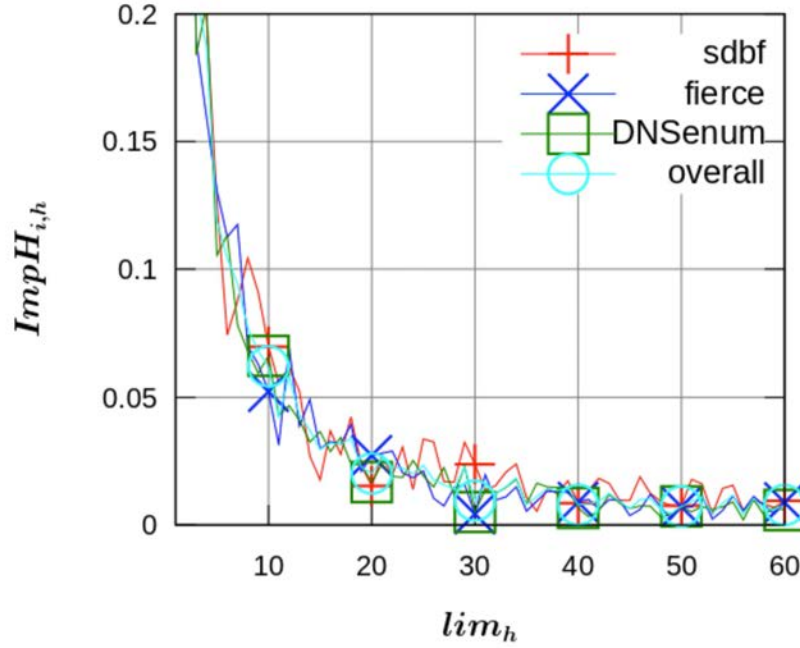


Figure 6.3: Ratio of newly generated valid subdomains depending on lim_h .

$$ImpH_{i,h} = \begin{cases} \%Imp_{i,h} & \text{if } h = 1 \\ \%Imp_{i,h} - \%Imp_{i,h-1} & \text{otherwise} \end{cases} \quad (6.8)$$

$ImpH_{i,h}$ is computed as an average improvement over all the 24 domain names of our test set. It is computed based on the subdomain sets produced by SDBF, Fierce, DNSenum and the union of these (overall). Figure 6.3 shows that the improvement decreases exponentially from $lim_h = 0$ to $lim_h = 60$. We can see that from $lim_h = 0$ to $lim_h = 10$ we have an improvement bigger than 5% for every tool. Moreover we can see that $ImpH_{i,h}$ has similar values for the different tested tools since the curves are very closed. From $lim_h = 10$ to $lim_h = 40$, $ImpH_{i,h}$ is still significant while decreasing progressively but for $lim_h \geq 40$, the improvement remains constant and is around 1% per new word tested. Hence, we conclude that having a horizontal exploration limit (lim_h) higher than 40 words does not significantly improve the results. That is why we set lim_h to 40 for the rest of the experiments. However, if we need a deep domain investigation, increasing lim_h leads to discover new names, as the improvement curves are still positive in Figure 6.3.

Vertical search:

Since the semantic probing method is based on previously discovered subdomains, we can re-launch it over newly discovered subdomains given by the last recursion performed. This number of performed probes is called the vertical depth and is limited by lim_v . The recursion process can stop before this limit if none of the newly generated names is valid as described in Section 6.2. We ran the DNS probing of the 24 domain names without fixing lim_v and observed that no more than five recursions were performed before the probing process stops automatically. Hence, the conclusion is that no name generated during the fifth recursion is valid. To test the optimal

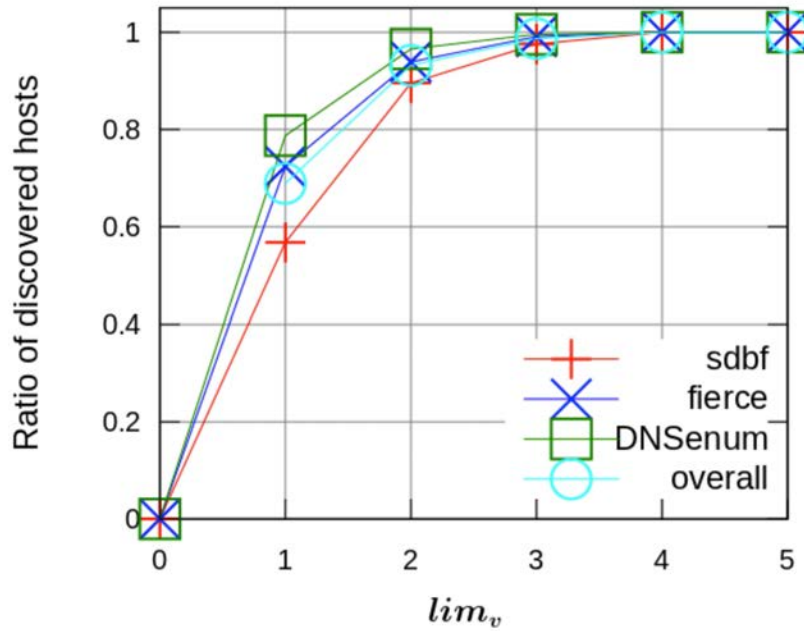


Figure 6.4: Cumulative frequency graph of discovered subdomains depending on lim_v .

vertical depth value, we compared the total count of valid names generated after five recursions to the total count obtained after n recursions, *i.e.* $0 \leq n = lim_v \leq 5$. Figure 6.4 represents the cumulative frequency graphs of valid names generated according to the number of recursions performed. This is given in ratio of the total count of discovered subdomains for $lim_v = 5$. We observed that between 55% and 80% of the subdomains are found during the first recursion. This score depends on the tool providing the initial set of valid subdomains. More than 95% of the valid subdomains are generated before the fourth recursion for the three tools. Hence, we can reasonably limit the probe to three recursions by setting $lim_v = 3$. This gives good results while limiting DNS requests performed.

From the experiments, we have observed that the horizontal and the vertical depth are linked. We raised the horizontal depth to 200 and observed that almost all the new valid subdomains were discovered during the first recursion. However, it is worth noting that choosing a high horizontal depth value leads to perform more useless DNS requests as words less likely to be related with the original subdomains are tested. As observed in Figure 6.3, we have better success rate for low values of lim_h .

6.3.3 Performance Evaluation

Efficacy and improvement

Having defined the optimal values for initial parameters, we evaluate the semantic DNS probing. We set $lim_h = 40$ and $lim_v = 3$. We ran the DNS probing of the 24 domain names presented in Section 6.3.1 based on the initial sets of subdomains obtained from SDBF, Fierce and DNSenum. Figure 6.5 and Figure 6.6 shows the result of this probe. Domain names are presented on the

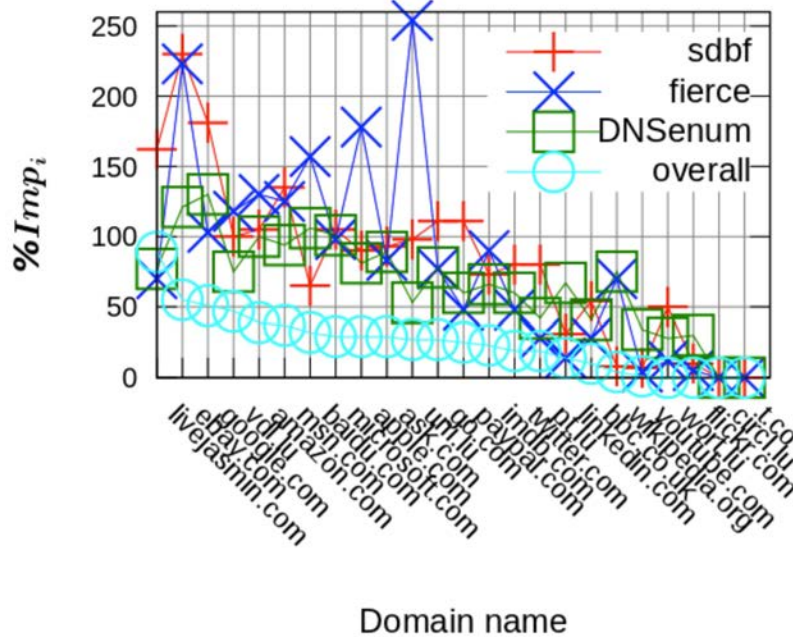


Figure 6.5: Percentage ($\%Imp_i$) of newly discovered subdomains for each domain names

x-axis. The improvement $\%Imp_i$ for each tool and each domain name is plotted on the y-axis in Figure 6.5. Figure 6.6 shows the count of newly discovered subdomains. Regarding the individual improvements, in many cases the count of discovered subdomains for each domain name is doubled or more ($\%Imp_i > 100\%$). For instance, with the original dataset from SDBF, the count of valid subdomains from *go.com*, *msn.com* or *google.com*, is increased by more than 100%. Moreover for *ebay.com*, we reach an improvement higher than 200% for both SDBF and Fierce. Similar results can be observed for DNSenum. For *ebay.com*, *baidu.com* and *msn.com*, the semantic exploration discovers hundreds of new subdomains that others solutions are not able to find. Table 6.1 presenting the detailed results of this experiment, shows that the average improvement for each tool is between 84% (for SDBF) and 102% (for Fierce). Hence, semantic DNS probing provides a relevant complementary solution to these three tools, being able to double in many cases the count of known subdomains.

Looking at improvement provided by semantic DNS probing to the *overall* set of subdomains generated by the three tools, results are a bit less significant. For four domain names (*wort.lu*, *flickr.com*, *circl.lu*, *t.co*) semantic exploration does not improve the results of the three combined tools. However, for *livejasmin.com*, *ebay.com* and *google.com*, the improvement over the three combined tool is higher than 50%. The average improvement over the three tools is 30% ($\%Imp_{overall} = 30$) proving that semantic DNS probing provides a solution to discover new subdomains that existing solutions are unable to find.

This proves the efficiency of semantic DNS probing since the most common subdomains have already been discovered by one of the initial tools (SDBF, Fierce or DNSenum). From a domain name such as *mars.pt.lu*: *merkur.pt.lu* and *jupiter.pt.lu* have been found. From *kangaroo.apple.com*: we discovered *camel.apple.com*, *porcupine.apple.com* and *piglet.apple.com*. Our first assumption deduced from observations that subdomains are attributed by human and thus,

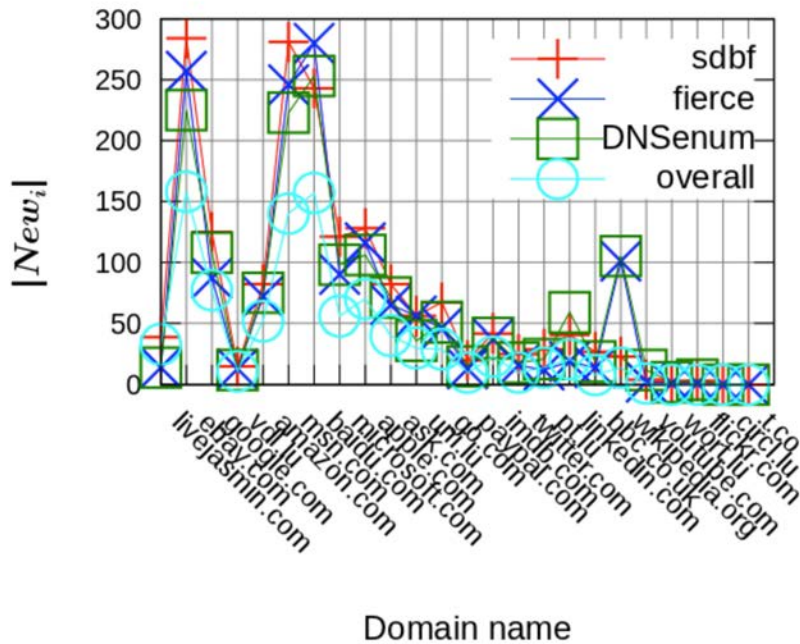


Figure 6.6: Count ($|New_i|$) of newly discovered subdomains for each domain names

a semantic relation exists between these, proves correct. Network administrators use semantic relations to link different machines. These domain names, based on planets or animals for example confirm it.

Strategy

As introduced in Section 6.3.1, different strategies are tested by combining the similar names module (SN), the splitter and the incremental discovery module (ID). Figure 6.7 shows the efficiency of several strategies initialized with SDBF. We see that the similar names module, being the base of each strategy evaluated, leads to discover the main part of new subdomains. The plots of other strategies mainly coincide with the one from the similar names module used alone. The second observation is that the splitter module provides very few improvement to similar names, being almost always merged with its plot. Finally, the incremental discovery brings some good results, especially for the domain names *livejasmin.com* and *linkedin.com*. Using this module and from the known subdomains *news10.livejasmin.com*, we discovered 31 new subdomains *newsX* with $X \in \{1; 9\} \cup \{11; 32\}$.

These results show that the strategy has to be carefully chosen. For fast probing of many domain names, only the similar names module should be used. However, if the objective is to exhaustively probe one domain, all modules must be combined, since each of them improve the results.

Overhead

The overhead in this context is defined as the count of DNS requests performed ($\#probes$). As previously mentioned, SDBF and DNSenum require more than 250,000 DNS probes per domain

Domain names	SDBF			Fierce			DNSenum			Overall		
	$ Init $	$ New $	$\%Imp$	$ Init $	$ New $	$\%Imp$	$ Init $	$ New $	$\%Imp$	$ Init $	$ New $	$\%Imp$
livejasmin.com	24	39	162	20	14	70	18	14	77	37	33	89
ebay.com	123	284	230	115	257	223	185	225	121	284	158	55
google.com	69	125	181	84	87	103	83	108	130	149	77	51
vdl.lu	15	15	100	11	13	118	16	12	75	23	11	47
amazon.com	78	82	105	55	72	130	75	75	100	132	52	39
msn.com	207	281	135	196	246	125	236	223	94	372	140	37
baidu.com	369	243	65	178	280	157	238	253	106	478	157	32
microsoft.com	115	121	105	91	90	98	97	98	101	189	56	29
apple.com	141	128	90	65	116	178	130	106	81	241	70	29
ask.com	88	82	93	78	65	83	79	71	89	135	40	29
uni.lu	57	56	98	22	56	254	67	36	53	110	30	27
go.com	60	67	111	59	46	77	65	51	78	104	29	27
paypal.com	18	20	111	27	13	48	25	15	60	39	10	25
imdb.com	57	42	73	41	37	90	57	38	66	98	23	23
twitter.com	31	25	80	33	16	48	30	18	60	51	10	19
pt.lu	36	29	80	42	12	28	49	21	42	71	14	19
linkedin.com	131	41	31	130	19	14	88	59	67	147	21	14
bbc.co.uk	50	27	54	49	14	28	46	19	41	79	8	10
wikipedia.org	282	23	8	143	101	70	139	105	75	296	14	4
youtube.com	55	4	7	49	2	4	35	12	34	62	1	1
wort.lu	4	2	50	8	1	12	7	2	28	13	0	0
flickr.com	29	3	10	17	1	5	13	4	30	40	0	0
circl.lu	17	0	0	6	0	0	9	0	0	19	0	0
t.co	1	0	0	1	0	0	1	0	0	1	0	0
all domains	2057	1739	84	1520	1558	102	1788	1565	87	3170	954	30

Table 6.1: Probing results for 24 domain names

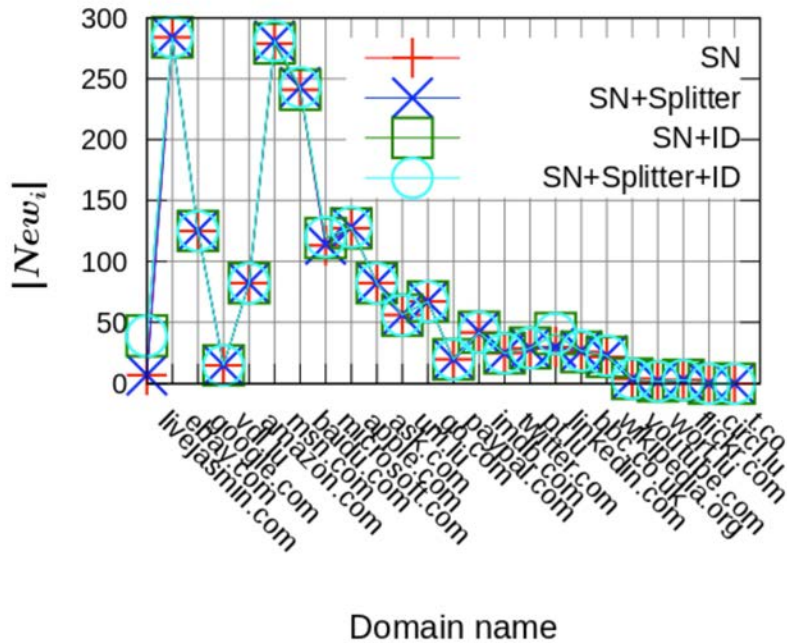


Figure 6.7: Count of newly discovered subdomains regarding the strategy used.

names to produce their results. Figure 6.8 depicts the count of DNS requests made to probe each domain name with semantic DNS probing. We can observe that our method always needs to

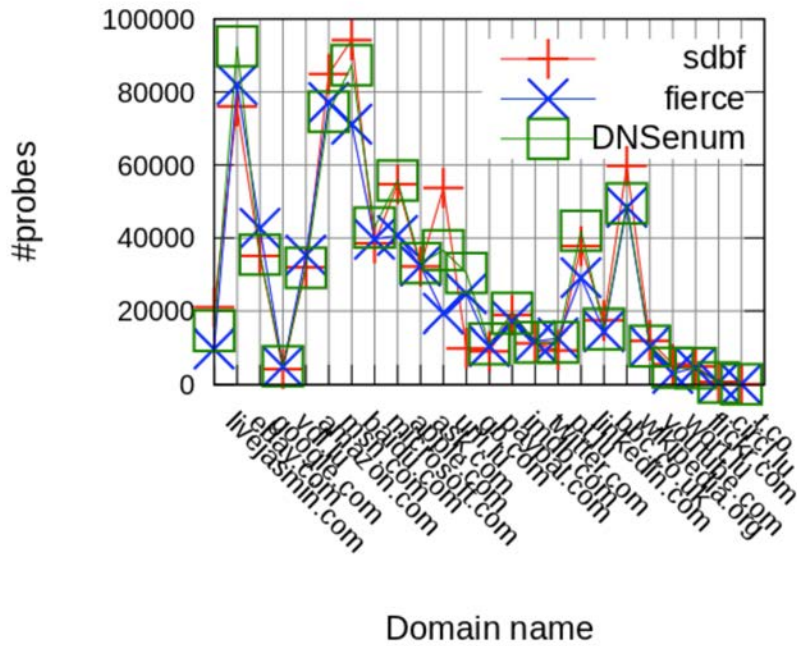


Figure 6.8: Count of DNS requests made per domain name to probe.

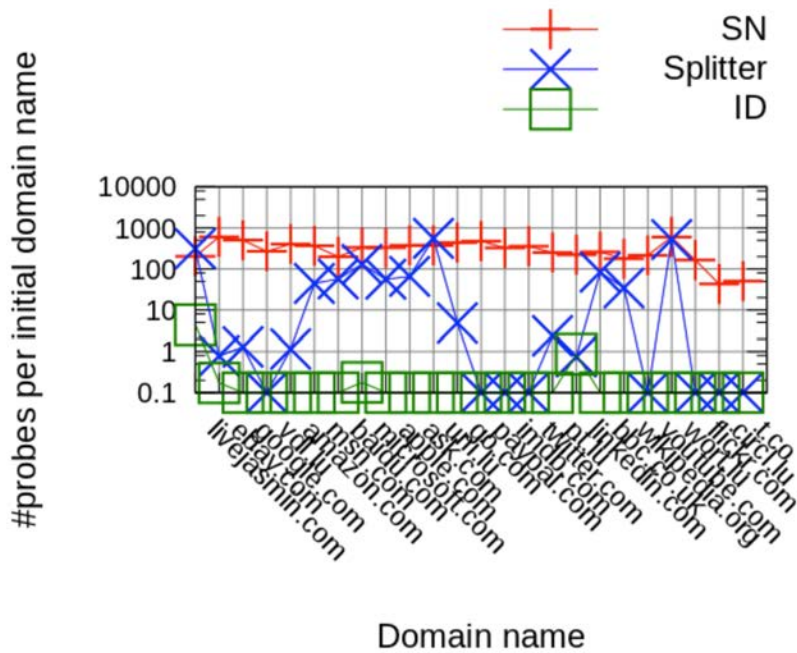


Figure 6.9: Count of DNS request made per subdomain in the initial dataset.

perform less than 100,000 DNS requests. The DNS requests count of the semantic DNS probing is based on the set of names provided by prior tools. The highest count of probes are made

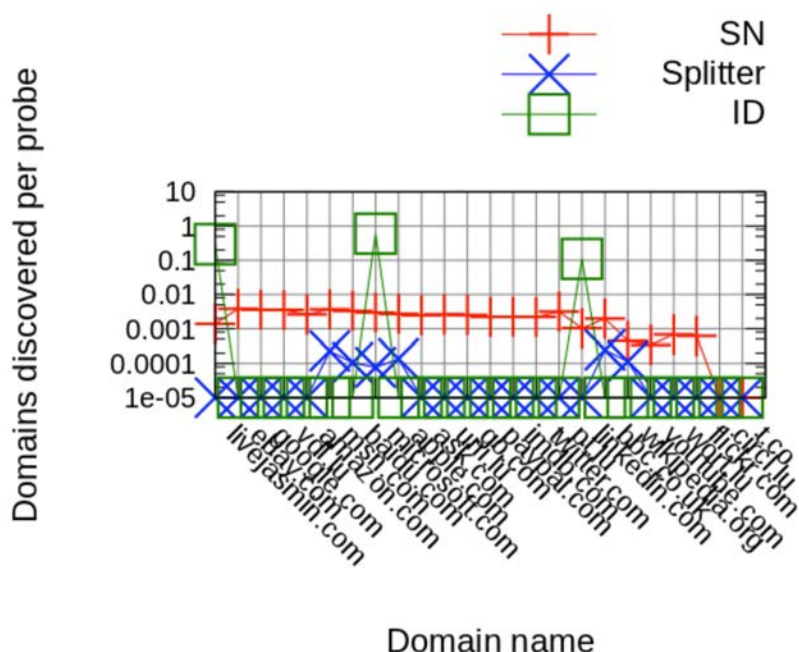


Figure 6.10: Ratio of newly discovered subdomain per probe.

for the largest initial datasets (*ebay.com*, *msn.com*, *baidu.com*). However, half of the probing campaigns requires less than 20,000 DNS requests to complete.

Figure 6.9 shows the count of DNS requests per initial subdomains depending on the semantic module used (SN, Splitter, ID). Figure 6.10 shows the average count of subdomains discovered by DNS request depending on the semantic module used. This depicts the ratio of valid subdomains generated by the semantic modules out of the total count of generated names. In Figure 6.9 we observe that the similar names module engenders a quite steady ratio of requests per initial name (between 200 and 500 requests). The efficiency of this module, as we can see in Figure 6.10, is also steady, it discovers around one domain name for 200 probes (0.005 subdomain/probe). Other modules engender less DNS requests than the previous one, as we can see in Figure 6.9 but are also less efficient in discovering valid subdomains. It is worth noting that the incremental discovery module discovers new subdomains in very few probes when used. Figure 6.10 shows that subdomains are discovered for three domain names in less than 10 DNS requests per valid name found. This highlights that the incremental discovery module is very efficient but only in specific cases.

These results show that semantic DNS probing requires less requests than previous DNS probing methods (more than four times compare to SDBF or DNSenum). It can discover approximately the same count of new subdomains as observed in Section 6.3.3. As a basis the similar names module should be used since it provides the steadiest and the best results. Nevertheless, the efficiency of the other modules (splitter and incremental discovery) depends on the targeted domain name and these are useful in specific cases.

Conclusion

To switch from phishing domain names and URL detection to prediction of domain names used in phishing activities, we explored in this chapter the predictable character of domain names. Some work already disclosed this characteristic by providing efficient DNS probing solutions to discover subdomains of a domain name. These state of the art techniques rely though on basic implementations that need many probes and these are not adaptable with their fixed dictionary. Hence, to complement this work we introduced a new DNS probing technique that captures and exploits the meaning of domain names.

We argued that, since the DNS has a role to provide a meaning to IP addresses, semantic analysis can lead to predict domain names composition. Based on the fact that domain names are given by people, we assumed that subdomains of a given domain name share common semantic fields. Hence, we introduced three modules able to generate new labels likely to be subdomains of certain domain name. These semantic modules exploit the composition of domain names by extracting meaningful labels in order to find related words to these labels. This method was assessed against 24 popular domain names and compared to three existing DNS probing techniques. We tested the ability of semantic DNS probing to improve the results delivered by existing solutions and showed that this method is at least as efficient as state of the art techniques while producing a lowest overhead.

This shows that semantic analysis of domain names is able to extract domain names composition schemes. By carefully using the extracted information we can build models able to predict domain names that are likely to be used. The results of experiments carried out in this chapter assessed that domain names are predictable since we were able to discover subdomains of popular domain names. Although the task of predicting subdomains is easier than full domain names prediction, this gives serious hints on the applicability of semantic techniques to predict full domain names.

Chapter 7

Proactive Discovery of Phishing Domain Names

Contents

7.1	Modeling a Phisher’s Language	124
7.1.1	Domain Names Features	125
7.1.2	Domain Names Generation Model	126
7.2	Domain Names Features Evaluation	129
7.2.1	Dataset	129
7.2.2	Features Analysis	130
7.3	Phishing Domain Names Generation	133
7.3.1	Types of Generated Domains	134
7.3.2	Efficiency and Steadiness of Generation	136
7.3.3	Predictability and Strategy	138
1	Summary of Contributions	143
2	Research Perspectives	145

Introduction

Current approaches to cope with phishing are reactive techniques. They are mostly implemented as blacklists or with real-time identification methods relying on machine learning. The shortcoming of URL blacklist is the delay between the time when a phishing website is set online and the time when its URL is blacklisted. This delay can reach several days. Knowing that the average uptime of phishing attacks is around 32 hours and the median uptime is only nine hours [AR14], reactive blacklisting is not suited to cope with phishing. Due to this short lifetime, in most cases phishing websites are included in blacklists when they are not active anymore. Regarding real-time identification methods, we introduced in Chapter 5 such a technique that is shown efficient. Other examples of such techniques are Google Safe Browsing and Microsoft Smart Screen being integrated in Google Chrome and respectively Internet Explorer client web browser. These are efficient phishing prevention technique as shown in [LMKK07]. In addition, these identification techniques operate in real-time on previously unseen URLs as shown in Chapter 5. However, while blacklists can be made publicly available and thus easily integrated to any web client or

mail client, automated detection techniques rely on complex algorithms that cannot be easily integrated in any application. Blacklist checking only requires a lookup in a list. Automated detection techniques require the extraction and the processing of data, which consumes memory and processing resources and need as well some software products to be installed. These are often proprietary solutions dedicated to specific software or applications, which limits their widespread usage contrarily to blacklists.

Hence, we introduce in this chapter a technique to build a phishing domain names blacklist that does not rely on the protracted process of users report and verification. Based on the finding made in Chapter 6 and related to the predictability of domain names, we develop a technique to generate domain names likely to be used for phishing. We focus on the generation of registered domain names *mld.ps* that are registered by phishers. The reason is that phishers maliciously register few domain names that are used as a basis in several phishing URLs and several phishing campaigns [PKKG10]. Moreover, some phishing attacks involve URLs containing unique number in order to track targeted victims. The only common point between these unique URLs remains their domain name. Hence, identifying one phishing domain name can lead to discard hundreds of phishing URLs. This makes domain name blacklisting more efficient than URL blacklisting, which must make a perfect match between entries. We have shown in Chapter 4 that phishing domain names present similarities, especially in their semantic composition. These similar characteristics can be identified and used to automatically generate phishing domain names.

Leveraging natural language processing methods, we build a proactive domain name monitoring scheme that generates a list of potential domain names to track in order to identify new phishing activities. The creation of the list uses phishing domain name features to build a natural language model using Markov chains combined with semantic associations. This allows to generate domain names already used or that will be used in phishing attacks and integrate these in a proactive blacklist. This copes with the delay usually needed to add an entry in a reactive blacklist, making this technique suited to phishing and its short lifetime websites. We evaluate and compare the introduced features using malicious and legitimate datasets before testing the ability of the approach to proactively discover new phishing domain names. The contributions of this chapter were published in [MFSE12a].

The rest of this chapter is organized as follows: we start in Section 7.1 by introducing the domain name generation technique based on features extracted from existing domain names and a natural language model. In Section 7.2, we present a phishing domain names and a legitimate domain names dataset and compare domain names composition regarding several lexical features. We assess the proactive generation of phishing domain names in Section 7.3.

7.1 Modeling a Phisher’s Language

Phishers are human who use common words to compose their phishing domain names. They use names similar to legitimate domain names, append some other words that come from a specific vocabulary and leverage some domain specific knowledge and expertise. Phishing domain names follow some composition patterns characterizing their usage. Hence, we argue that the monitoring and the analysis of existing domain names to extract these patterns allow to emulate the composition process used by phishers in order to generate potential phishing domain names. The domain names considered in this chapter are maliciously registered domain names. These are composed of two parts, the main level domain (*mld*) and the public suffix (*ps*). The registered domain *mld.ps* has been defined in Section 4.1. These maliciously registered domain names

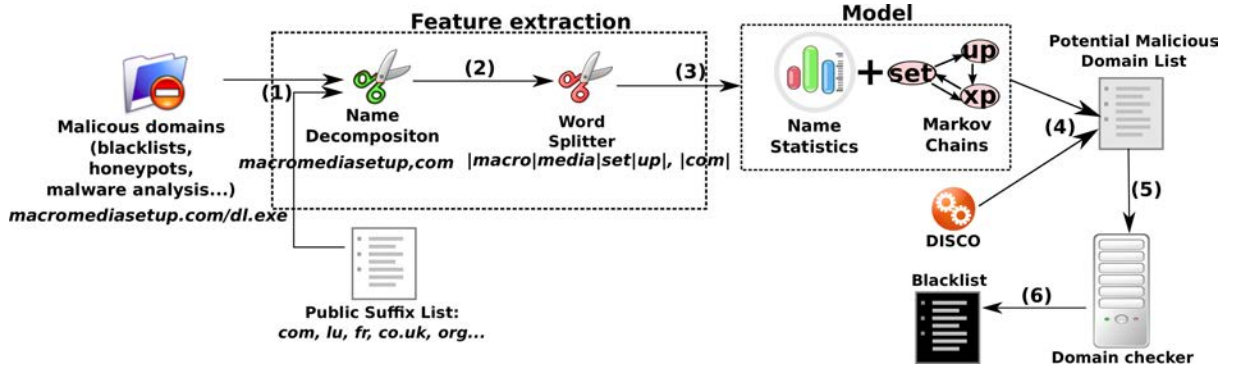


Figure 7.1: Overview of the phishing domain names generation and validation process

are the ones targeted by our work. As presented in Section 4.1 with the different obfuscation techniques, phishing domain names are often composed of appended meaningful words. We introduce a method to analyse and build a model of words composing domain names in order to generate new ones.

An overview of the domain names generation system is illustrated in Figure 7.1. The input of the system is a set of domain names from which we extract features. This set can be composed of any kind of domain names having similarity in their composition. In our case, since we want to generate domain names likely to be used for phishing, this is a set of blacklisted phishing domain names. Based on this set, the first step (1) decomposes the domain name and extracts the two parts for which we build a model: the *mld* and the *ps*. Each of these two parts is divided into words (2) and composition features are extracted. For the public suffix, a simple split according to dots is sufficient. For the main level domain, a word segmentation technique [SH09] is used in order to extract every meaningful word. The extracted features are then used to build a model (3) by computing statistics. The length distribution of the main level domain in words is one of these features for instance. This model is combined (4) with semantic extensions to enlarge the generation scope. Generated domain names can be currently involved, have been involved or will be involved in phishing activities. These must be tracked on a daily basis for instance, in order to detect new phishing websites. The domain names are checked by the domain checker (5) to confirm their existence and their malicious activities. This checking can be made using one of the following technique: signature-based approach, honeypots, manual analysis, etc. Once existence and maliciousness confirmed, checked domain names are added to a blacklist (6) that can be integrated to various DNS based intrusion prevention systems.

7.1.1 Domain Names Features

We extract features representing the composition of malicious domain names. The trend to embed several words in domain names characterizes phishing. Hence, we analyse and identify which words are used, how many words compose a domain name, which words follow a given one in phishing domain names. We introduce features representing this composition scheme in order to build a model of domain names composition. This word composition model mainly applies to the main level domain. Having a set of *mlds* $M = \{mld_1, mld_2, \dots, mld_n\}$ extracted from a set of domain names $D = \{d_1, d_2, \dots, d_n\}$ and assuming $W = \{w_1, w_2, \dots, w_m\}$ the set of all words extracted from $mld \in M$, we define four statistical features extracted from M :

- $\#wlen_n$ the count of main level domains $mld \in M$ composed of n words.

- $\#mld_w$ the count of main level domains $mld \in M$ containing the word $w \in W$.
- $\#fisrtword_w$ the count of main level domains $mld \in M$ starting with the word $w \in W$.
- $\#biwords_{w_1, w_2}$ the count of main level domains $mld \in M$ containing the word w_1 followed by w_2 with $(w_1, w_2) \in W^2$.

We compute from these statistical features the corresponding distributions to be used in a domain name generation model:

- the distribution of the main level domain length $n \in \mathbb{N}$ (in words):

$$distwlen_n = \frac{\#wlen_n}{\sum_{i \in \mathbb{N}} \#wlen_i} \quad (7.1)$$

- the distribution of main level domains $mld \in M$ containing the word $w \in W$:

$$distmld_w = \frac{\#mld_w}{\sum_{i \in W} \#mld_i} \quad (7.2)$$

- the distribution of main level domains $mld \in M$ starting with the word $w \in W$:

$$distfirstword_w = \frac{\#fisrtword_w}{\sum_{i \in W} \#fisrtword_i} \quad (7.3)$$

- the distribution of main level domains $mld \in M$ containing the consecutive words $w_1 \in W$ followed by $w_2 \in W$:

$$distbiwords_{w_1, w_2} = \frac{\#biwords_{w_1, w_2}}{\sum_{i \in W} \#biwords_{w_1, i}} \quad (7.4)$$

For the set of public suffixes $P = \{ps_1, ps_2, \dots, ps_n\}$ extracted from the domain names $D = \{d_1, d_2, \dots, d_n\}$, we define two simple features:

- $\#d_{ps}$ the count of domain names $d \in D$ having the public suffix $ps \in P$.
- the distribution of domain names $d \in D$ having the public suffix $ps \in P$:

$$distd_{ps} = \frac{\#d_{ps}}{\sum_{i \in P} \#d_i} \quad (7.5)$$

Main level domains composition is detailed using several features and distributions since their composition is complex and this is the part defined by phishers. Public suffixes do not offer flexibility in their composition. Phishers just have to pick one under which they can register their phishing domain names. However, we represent the usage of different public suffixes with different probabilities since some public suffixes are used more in malicious activities than in others [AR14]. This is explained by public suffix popularity and by some registrars being more careful than others when authorizing domain names registration.

7.1.2 Domain Names Generation Model

The generator designed for domain name generation is based on an n-gram model and a Markov chain. Coming from natural language processing, n-grams [MS99] are successive *grams* sequences of length $n \in \mathbb{N}$, extracted from a string. For example, an n-gram with $n = 2$ is called a *bigram*, an n-gram with $n = 3$ is a *trigram*. These *grams* are usually characters. Such a model is used with characters in SDBF to generate subdomains as presented in Section 6.1.3. However, in this chapter, the considered *grams* are words. N-gram analysis provides a way to study string composition by gathering *grams* transitions with these sequences. To study transitions from one word to the following one without considering previous context, bigrams are sufficient. Hence, in the context of this study considering words, we especially focus on bigrams of words that are called *biwords*.

Considering the following label *macromediasetup*, it is composed of four words being extracted during the name decomposition process: *macro*, *media*, *set*, *up*. Three biwords are extracted from this label and these are: *macromedia*, *mediaset*, *setup*. The probability of having these biwords in a set of domain names has been defined in Equation (7.4). This probability represents in fact the probability of having one word following a given one. This gives a model for domain names composition. These probabilities of transitions with the words extracted from a set of domain names are used to build a Markov chain model representing their composition.

A Markov chain [Mar71] is a mathematical system defined as a set of states $S = \{s_1, s_2, \dots, s_r\}$ and possible transitions between these states. A Markov chain undergoes transitions from one state to another. Future states only depends on the current state, no previous state is considered when computing a transition. Each possible transition between two states can be taken with a transition probability. Shannon was the first to propose a Markov model of natural language in [Sha48] to approximate the statistical structure of a piece of text. We use a Markov model the same way to represent the domain name structure. The states of the Markov chains are defined as the words $w \in W$ and the probability of transition between two words/states w_1 and w_2 is given by $distbiwords_{w_1, w_2}$. In order to generate new names, the Markov chain is completed with additional transitions that have never been observed. This technique is called additive smoothing or Laplace smoothing. For each state s , a small probability (0.05) is assigned for transitions to all the words $w \in W$ for which s does not have any transition yet. This probability is shared between the words according to the distribution $distmld_w$ given in Equation (7.2). The same method is applied for the states s that do not have any existing transitions. In this case, their transitions follow the exact probability given by the distribution $distmld_w$. The initial state of the Markov chain, *i.e.* the one from which the transition process can start is randomly selected for every domain name generation using $distfirstword_w$ in Equation (7.3). The count of transitions made before the process ends and the domain name is generated is defined by $distwlen_n$ in Equation (7.1). Given these two last parameters by applying n steps from a word w in the Markov chain, a label is generated as the *mld* of a domain name.

A part of a created Markov chain is given in Table 7.1 for some transitions and the associated probabilities, starting from the word/state *pay*. Figure 7.2 depicts a small Markov chain model for generating domain names. This is composed of seven states, the only initial state is *secure*. We can see in black the original transitions with their associated probabilities learned from the learning set. Green transitions depict the transitions obtained using Laplace smoothing on the state *secure*.

The words composing the main level domain of different malicious domain names often belong

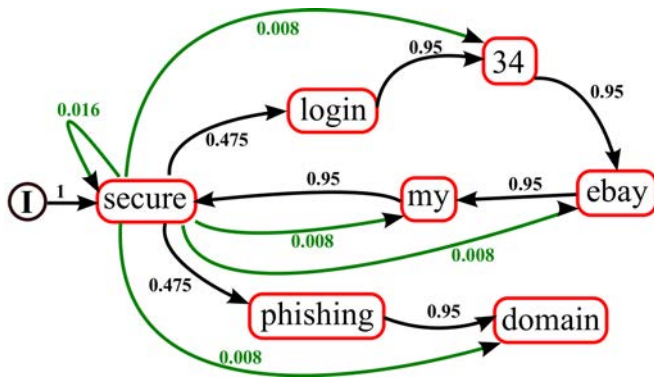


Figure 7.2: Markov chain model example

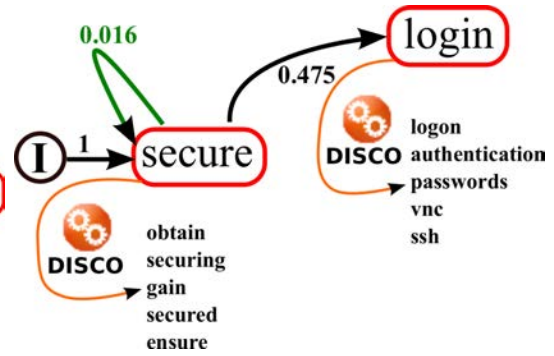


Figure 7.3: Markov chain semantic extension

Transition	per	z	for	secure	bucks	bill	process	pay	account	soft	page	...
Probability	0.13	0.1	0.06	0.06	0.06	0.03	0.03	0.03	0.03	0.03	0.03	...

Table 7.1: Example of Markov chain transitions for the state *pay*

to one or more shared semantic fields. Given some malicious domain names such as *xpantivirus-local.com*, *xpantivirusplaneta.com*, *xpantivirusmundo.com* and *xpantivirusterra.com*, it clearly appears that these are related. Applying the word extraction process on these domain names, the words *xp*, *anti* and *virus* are extracted from all of them. Moreover, four words *local*, *planeta*, *mundo* and *terra* are extracted from each of them. These four words are closely related, particularly the three last ones. Given one of these domain names, the remaining three can be found easily by finding related words to *planeta* or *mundo*. However, even if this intuitive conclusion is obvious for human, it is more complicated to implement in an automated system.

To improve our generation engine based on the Markov chain model, we use once again DISCO. To each state of the Markov chain, we take the associated label l and compute $Disco(l, 5)$ returning the five words $\{w_1, w_2, w_3, w_4, w_5\}$ most related to l . Then, we compute $sim(l, w_i)$ for each of these words, which gives the similarity score between l and w_i . $Sim(l, w_i)$ has been defined in Equation (4.3). We include this result in the Markov chain by providing alternative word choices in each state. Once a transition to a state s containing the label l is chosen there are two solutions:

1. The label l is selected to be part of the generated domain name with a probability 0.5.
2. One of the words w_i from $Disco(l, 5)$ is selected to be part of the generated domain name with a probability $0.5 \times \frac{sim(l, w_i)}{\sum_{k=1}^5 sim(l, w_k)}$.

This semantic extension is depicted in Figure 7.3 for two states of the Markov chain model from Figure 7.2. Including this semantic extension allows to generate domain names including words that have not been observed in the domain names learning set. This increases the flexibility of the approach and does not limit the malicious domain name discovery to the learning set.

A complete example of label generation is illustrated in Figure 7.4 for a main level domain. The length of the label in words is first computed (1), then the first word that starts the label is selected (2). The Markov chain is applied for the remaining words to generate (3). For each word at the step (2) and (3), DISCO is applied to generate other words. To complete the process, a

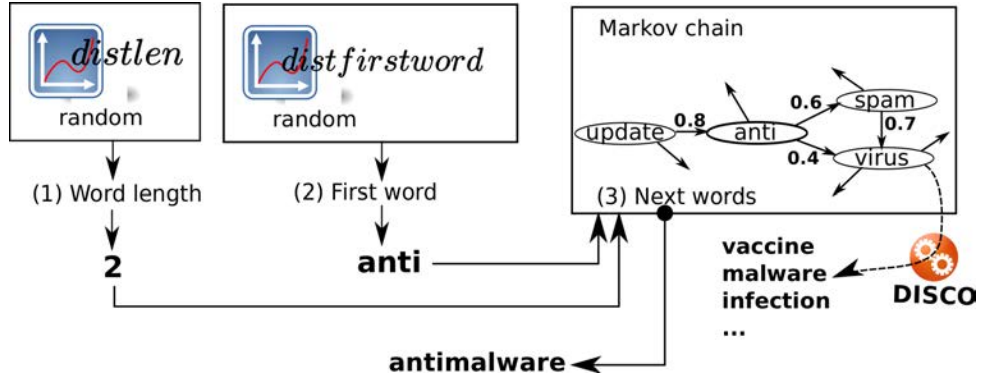


Figure 7.4: Main level domain generator

public suffix is appended to the generated main level domain. This is randomly selected following the distribution $dist_{ps}$ defined in Equation (7.5).

7.2 Domain Names Features Evaluation

For assessing our approach, two datasets are selected. The first one is a malicious dataset composed of domain names from which maliciousness is confirmed. The second dataset is a legitimate dataset containing non-malicious domain names. In a first step, these are used to show that the features introduced in Section 7.1.1 allow to discriminate phishing domain names from legitimate ones. In a second step, the malicious dataset is used to train the phishing domain name generator to assess its efficiency.

7.2.1 Dataset

To compose the dataset of malicious domain names, three freely downloadable blacklists are used. These blacklists are the same that were previously used in Chapter 4. These have been selected because each of them proposes an historical list of blacklisted domain names ordered by their discovery date. This is an essential dataset requirement in order to test the predictability of the approach. Each blacklist has collected malicious domain names during at least three years.

- **PhishTank** [phi]: The downloaded historical blacklist contained 3,738 phishing URLs.
- **DNS-BH** [mala]: A list of 17,031 malicious domain names was available.
- **MDL** [malb]: This blacklist contained 80,828 URL entries.

DNS-BH and MDL are not only dedicated to phishing, but also to malware diffusion. These two lists have been chosen because as described in [AR14], diffusion of malware designed for data-stealing and particularly crimeware represents a large part of phishing activities. This various dataset allows also to strengthen the validation of the approach. Following the extraction of the distinct domain names from the 101,597 URLs and the deletion of duplicated entries between the three lists, the final dataset contains 51,322 different registered domains $mld.ps$. Out of these 51,322 domain names, 39,980 have their mld divisible in at least two parts. This proves that the majority of phishing domain names have a main level domain composed of several words.

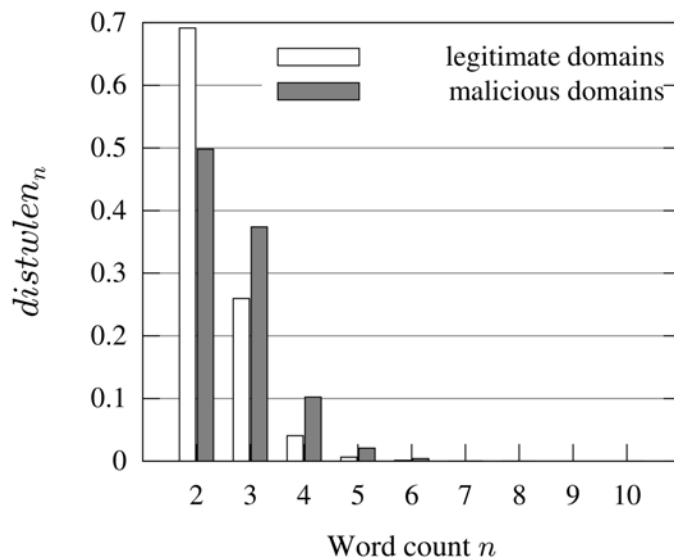


Figure 7.5: $distwlen_n \mid n \in \{2; 10\}$ for malicious and legitimate domain names

To complement this malicious dataset we add legitimate domain names. The legitimate dataset is selected to show that even if malicious domain names use some brands included in the URLs of popular websites in order to mimic them, these still disclose differences. Two sources are chosen to compose this legitimate dataset. These are the same used in Chapter 4.

- **Alexa** [ale]: From Alexa's top *1,000,000 websites* list, 40,000 domain names are randomly picked in the top 200,000 domain names.
- **Passive DNS** from a Luxembourg ISP: To diversify this dataset and in order to have the same amount of domain names in each dataset (legitimate/malicious), we complement it with 11,322 domain names extracted from DNS responses. DNS responses were passively gathered from DNS recursive servers. We ensure that these domain names are not present in the initial dataset from Alexa.

The legitimate dataset contains 51,322 entries. 38,712 names have their main level domain *mld* divisible in at least two parts. Hence, we have two datasets: a *legitimate* and a *malicious* dataset of equal size.

7.2.2 Features Analysis

The metrics and statistical parameters extracted from each dataset are compared to demonstrate that the introduced features are different when extracted from malicious or legitimate domain names. This is done to show that the domain name generator will be able to generate phishing domain names when it learns a model from phishing domains. By showing that statistical features being the base of the generator are different between malicious and legitimate domain names, it proves the model can mimic the composition of each kind of domain names and that the model will be different for each kind of domain names.

We first analyse the count of words n that composes the main level domain of malicious and legitimate domain names: $\#wlen_n$. In this experiment only domain names that can be split in

at least two parts are considered. The malicious dataset contains 39,980 such domain names and the legitimate dataset 38,712. Figure 7.5 shows the distribution of the ratio of *mlds* that are composed from 2 to 10 words ($distwlen_n \mid n \in \{2; 10\}$) in the legitimate dataset and in the malicious dataset.

We can see in Figure 7.5 that 69% of legitimate *mlds* are composed of two words whereas only 50% of malicious are. For all upper values of n , the ratio of malicious domain names is higher than the legitimate one. This shows that malicious *mlds* tend to be composed of more words than legitimate *mlds*. It confirms that phishers use several words in their maliciously registered domain names as an obfuscation technique. This makes $\#wlen_n$ a discriminative features for phishing and legitimate domain names.

We further examine the composition of legitimate and malicious domain names. We analyse the similarities of words used in main level domains (*mld*) and public suffixes (*ps*) of each dataset. Two probabilistic distributions are extracted from the domain names of the two datasets:

- the different public suffixes of $d \in D$: $\forall ps \in P, P_1(ps) = distd_{ps}$.
- the different words that compose the main level domains of $d \in D$: $\forall w \in W, P_2(w) = distmld_w$.

To compare these probabilistic distributions and infer similarities and dissimilarities we use the Hellinger Distance. The Hellinger Distance is a metric used to quantify the similarity (or dissimilarity) between two probabilistic distributions P and Q . The Hellinger Distance is symmetric and bounded on $[0; 1]$. A score of 1 corresponds to a total dissimilarity ($P \cap Q = \emptyset$) and a score of 0 means that P and Q have the same probabilistic distribution. The Hellinger Distance is defined in continuous space in Equation (7.6). The equivalent function in discrete space distribution is given in Equation (7.7) and is the one that is considered to compute the similarity between the distributions P_1 and P_2 .

$$H^2(P, Q) = \frac{1}{2} \int \left(\sqrt{\frac{dP}{d\lambda}} - \sqrt{\frac{dQ}{d\lambda}} \right)^2 d\lambda \quad (7.6)$$

$$H^2(P, Q) = \frac{1}{2} \sum_{x \in P \cup Q} \left(\sqrt{P(x)} - \sqrt{Q(x)} \right)^2 \quad (7.7)$$

This metric is preferred to other more usual metrics such as the Jaccard Index or the KL-divergence. The Jaccard Index only considers the presence or not of an element in two datasets but does not consider the probability associated to each element. The KL-divergence metric is a non-symmetric measure and an unbounded function ($[0; +\infty]$). In addition, the KL-divergence requires that Q includes at least the same elements of P , *i.e.* $\forall i, P(i) > 0 \Rightarrow Q(i) > 0$. This constraint may not be satisfied with our datasets and the Hellinger Distance can consider disjoint probabilistic distributions. This makes the Hellinger Distance more suited to perform probabilistic distributions comparison.

To compare legitimate and malicious domain names, rather than only compare P_1 and P_2 extracted from the whole malicious and the whole legitimate dataset, we split these in five subsets of equal size. This allows not only to show the dissimilarity between malicious and legitimate domain names but also to show similarities between malicious domain names and similarities between malicious domain names. The malicious dataset and the legitimate dataset are randomly split in five smaller subsets, *mal-x* and respectively *leg-x* $\mid x \in \{1; 5\}$, of equal size (10,264 domain names).

	leg-5	leg-4	leg-3	leg-2	leg-1	mal-5	mal-4	mal-3	mal-2
mal-1	0.133	0.136	0.133	0.129	0.134	0.014	0.012	0.013	0.014
mal-2	0.134	0.140	0.135	0.131	0.135	0.014	0.012	0.013	
mal-3	0.135	0.139	0.134	0.131	0.136	0.013	0.013		
mal-4	0.130	0.136	0.131	0.127	0.132	0.013			
mal-5	0.134	0.138	0.132	0.129	0.134				
leg-1	0.017	0.017	0.018	0.019					
leg-2	0.018	0.020	0.018						
leg-3	0.016	0.019							
leg-4	0.017								

Table 7.2: Hellinger Distance for ps (leg=legitimate, mal=malicious)

Table 7.2 shows the Hellinger Distance computed pairwise between all 10 subsets for the public suffix (ps) distributions: $P_1(ps)$. Hellinger Distance values are represented in the cells of the table and a gray shaded key highlights these scores. The darker the cell is, the more similar are the probabilistic distributions extracted from each compared subset, *i.e.* the darkest cells represent a distance close to 0. Globally, the public suffixes are similar for domain names extracted from all subsets, since the distance does not exceed the score 0.15 ($0 < H(P, Q) < 0.15$). A clear difference is although present in $H(P, Q)$ when P and Q are picked from the same dataset (leg/leg or mal/mal), where we get a score $H(P, Q) \approx 0.015$. However, when subsets of domain names of different kinds are compared (leg/mal) the distance shows almost a ten-fold increase: $H(P, Q) \approx 0.130$. This shows that malicious domain names use similar public suffixes that are different from those used by legitimate domain names.

Table 7.3 considers the distribution of words in main level domains (mld): $P_2(w)$. These scores are depicted the same way as in Table 7.2 except that the gray color scale has a different range. Here, the distributions are more scattered and show higher distances ($0.4 < H(P, Q) < 0.6$) meaning less similarities. However, the difference is more important between subsets created from distinct datasets (mal/leg) where the distance reaches values around 0.56. When comparing legitimate subsets we got Hellinger Distance values lower than 0.5 globally. Moreover, we can see that malicious main level domains tend to be very similar in their composition and words used. These domain names have the lowest distance ($H(P, Q) \approx 0.44$) meaning that their probabilistic distributions are close and most likely contain many common words belonging to a limited vocabulary. This result highlight once again that malicious and legitimate domain names are different. Seeing that malicious domain names show the most similarities is a good thing to generate a composition model that can generalize and faithfully mimic these characteristics.

As a last feature analysis experiment, we extract the Markov chain model out of each full dataset, legitimate and malicious. Table 7.4 provides the statistics of the Markov chain model of mld composition extracted from each dataset. The count of initial states is given by $Card(V) \mid \forall w \in V, \#fisrtword_w > 0$, these are the entry point of the Markov chain. The total count of states in the Markov chain corresponds to $Card(W) \mid \forall w \in W, \#words_w > 0$. The count of transitions before implementation of Laplace smoothing is given by $Card(U^2) \mid \forall (w_1, w_2) \in$

	leg-5	leg-4	leg-3	leg-2	leg-1	mal-5	mal-4	mal-3	mal-2
mal-1	0.564	0.571	0.561	0.566	0.565	0.446	0.439	0.443	0.438
mal-2	0.565	0.569	0.566	0.571	0.565	0.445	0.447	0.446	
mal-3	0.561	0.566	0.563	0.569	0.564	0.448	0.444		
mal-4	0.563	0.567	0.558	0.564	0.561	0.447			
mal-5	0.564	0.565	0.554	0.555	0.558				
leg-1	0.501	0.494	0.490	0.493					
leg-2	0.493	0.497	0.496						
leg-3	0.490	0.491							
leg-4	0.489								

Table 7.3: Hellinger Distance for words in *mlds* (leg=legitimate, mal=malicious)

$U^2, \#biwords_{w_1, w_2} > 0$. This table strengthens the assertion that words present in malicious main level domains are more related together than those present in legitimate main level domains. As we can see, the Hellinger Distance is lower between malicious subsets compared to legitimate subsets despite the higher count of words (states) in the Markov chain created from the malicious dataset.

These experiments show that a generation model built from phishing domain names is likely to generate domain names having characteristics specific to phishing domain names. This proactive generation tool for maliciously registered domain would have a limited impact regarding legitimate domain names generation since these have different characteristics.

Metrics	<i>Legitimate</i>	<i>Malicious</i>
# initial states	14079	14234
# states	23257	26987
# transitions	48609	56286

Table 7.4: Markov chain statistics for main level domain model

7.3 Phishing Domain Names Generation

The dataset chosen for the rest of the experiments is the whole malicious dataset introduced in Section 7.2.1. This dataset is split in two subsets and depending on the experiment performed, the domain names selection technique to compose the subsets and the count of domain names in each subset vary. One of these subsets is called the training set, from which the features described in Section 7.1.1 are extracted in order to build the domain names generation model. Based on it, new domain names are generated and their maliciousness is confirmed if these belong to the second subset, called the testing set.

The term *probing campaign* is defined as the generation of one million unique *mld.ps* with the domain names generator. These are checked in term of existence and maliciousness. A domain name is considered as existing if it is actually reachable over the Internet, *i.e.* it is mapped to

an IP address. For each generated domain name, a DNS A request is performed and according to the DNS response status, the domain name is considered as existing (status = *NOERROR*) or non-existing (status = *NXDOMAIN*). Maliciousness is confirmed if the domain name belong to the testing set, which is composed of blacklisted domain names.

7.3.1 Types of Generated Domains

The first part of the experiment analyses the existence and the type of generated domain names. Five probing campaigns were run using a generation model trained on 10% of the malicious dataset. These 10% domain names were randomly picked in the dataset. We analysed the type of domain names that were generated out of these five millions domain names. Figure 7.6 is an histogram depicting the results of this experiment. Each of the probing campaigns is represented on the x-axis. Each of the five drawbars of the histogram represents the number of existing domain names generated, *i.e.* these having a DNS reply with the status *NOERROR*. Existing domain names are further divided in three categories being *wildcarding*, *domain for sale* and *unknown*.

We can see that between 80,000 and 110,000 existing domain names were generated during each probing campaign. Out of these, the majority represents wildcarded domain names (white rectangles in Figure 7.6). Domain wildcarding is a technique that consists in associating an IP address to all possible subdomains of a domain name by registering a domain name such as **.yahoo.com*. As a result all DNS queries sent for a domain name containing the suffix *yahoo.com* are answered with a *NOERROR* status DNS response containing always the same IP address. This technique is useful to tolerate Internet users typing mistakes, or misspelling of subdomains without any consequence. However, some public suffixes such as *.ws*, *.tk* or *.us.com* apply also wildcarding. As a result these public suffixes have been identified in order to discard all generated *mld.ps* that contain one of them. These domain names get DNS replies with the good status while not being necessarily registered and used. We can see that these domain names represent between 75% and 85% (from 60,000 to 90,000) of the existing domain names discovered.

The remaining part is composed of two categories. First, some domain names are registered but lead to websites of domain names resellers such as GoDaddy or Future Media Architect. A lot of meaningful domain names belong to this category, around 4,000 per campaign. Some examples of such domain names are *freecolours.com* or *westeuropa.com*. Regarding a probing campaign, the IP addresses obtained through DNS responses are stored and sorted by their count of occurrences. The IP addresses having more than 50 occurrences are manually checked to see if these are either related to real hosting or domain selling. Around 50 IP addresses and ranges have been identified as leading to domain name resellers. These domain names are also discarded in our study, as these are not really used while being registered. In addition, these are not likely to be malicious domain names.

Second, the black part of the histogram in Figure 7.6 represents the domain names that are unknown and have to be checked to confirm if these are related to phishing or not. As highlighted in Figure 7.6, the remaining potential malicious domain names registered represent only between 15,000 and 20,000 domains out of one million of generated ones. These may be considered as few domain names (only 2%) but since our method is a proactive one, all non-registered domain names generated can potentially be used in future phishing campaigns. These are not existing domain names yet. However, phishing websites lifetime is short. Domain names are maliciously registered, used few hours before being taking down and not being accessible anymore. Thus the existence of generated domain names must be checked regularly to discover new malicious domain names.

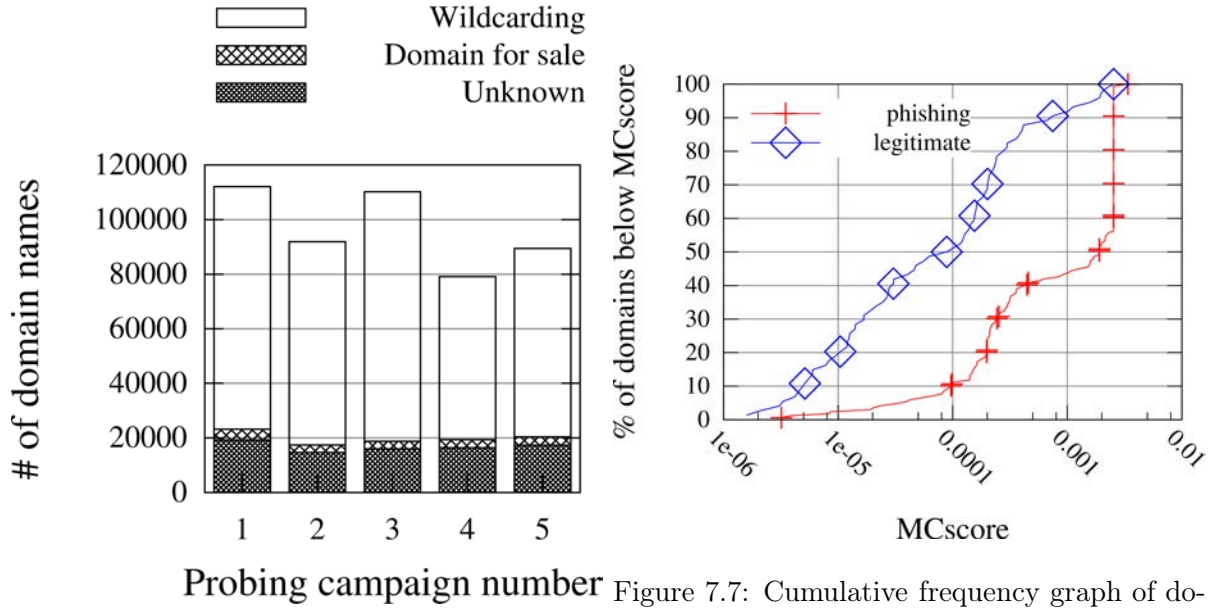


Figure 7.6: Proportion and type of existing generated domain names.

Figure 7.7: Cumulative frequency graph of domain names depending on their $MCscore$.

The reduction process presented to discard non-existing domain names, wildcarding domains and domains for sale is automated and reduce the overhead of the maliciousness checking process.

Some domain names of the unknown part were identified as malicious and legitimate. This was made using the remaining 90% of the malicious dataset and the legitimate dataset. Out of the five probing campaigns, we had around 500 generated domain names confirmed as malicious and around 200 generated domain names that were considered as legitimate per campaign. This portion of legitimate domain names generated is considered as false positives *i.e.* domain names generated as malicious while being legitimate. Out of one million domain names generated this gives a 0.02% false positive rate for the generator. This score is quite low but to reduce this number of incorrectly generated malicious domain names, we computed a score for each of these labelled generated domain names. This score, called the $MCscore$, measures the similitude with the underlying training dataset, which has been used for building the model. Assuming a registered domain name $w_1w_2\dots w_n.ps$ where w_i is the i^{th} word composing the name. w_i may have been generated using DISCO from an original word observed w'_i . The $MCscore$ is computed using the probability of the transitions made in the Markov chain during the generation process. This is described in Equation (7.8). The first word probability is multiplied by each probability of crossed transition in the Markov chain. If some parts are found using DISCO, the similarity score given by $sim(w_1, w'_1)$ is used. If the word corresponding to the state is used without requiring DISCO, we have $sim(w_i, w_i) = 1$.

$$MCscore = distfirstword_{w'_1} \times sim(w_1, w'_1) \times \prod_{i=1}^n distbiwords_{w_i, w'_{i+1}} \times sim(w_{i+1}, w'_{i+1}) \quad (7.8)$$

Figure 7.7 represents the cumulative frequency graph of domain names (in %) having a $MCscore$ lower than $x \in [1e^{-6}, 0.01]$. Phishing domain names are represented with the red curve and legitimate with the blue curve. We can see that globally phishing related domain names have a higher $MCscore$ than legitimate ones. Even if a high count of domain names are

labeled as unknown and some of these are legitimate, it is easy to discard a large part of these in order to keep a set containing a main part of malicious domain names. If we consider as malicious only the generated domain names having a *MCscore* higher than 0.001, then 93% of the legitimate domain names are discarded while 57% of the malicious domain names are kept according to Figure 7.7. This shows that using the most probable transitions in the Markov chain leads to generate domain names most likely to be used for phishing activities. Using less probable paths in the Markov chain leads to generate malicious but also legitimate domain names currently in use. The *MCscore* can be used to avoid the use of a maliciousness domain checking technique or to reduce its workload by testing only generated domain names having the lowest scores.

7.3.2 Efficiency and Steadiness of Generation

This section assesses the variation of the efficiency of the malicious domain names discovery regarding the count of domain names used to learn the model. The malicious dataset is split in two part for this experiment: a training and a testing set. The training set is used to train the generator by building the Markov Chain model from domain names contained in it. The testing set is used to test if the domain names generated using the learned Markov model are actually malicious or not. The generated domain names are checked against the part of the malicious set that was not used to build the model, *i.e.* the testing set, and confirmed as malicious if they are part of this. This assessment seeks to evaluate the count of know phishing domain names the training set must contain in order to build a model that can faithfully represent the composition of phishing domain names. Five probing campaigns were performed with a ratio that varies from 10% training/90% testing (10/90) to 90% training/10% testing (90/10). The training and testing subsets were randomly made up. Figure 7.8 shows the count of malicious domain names generated per campaign depending on the total count of generated names from 0 to 1,000,000. The five curves represent the five splitting ratio used with the proportion of the training set first and then the proportion of testing set in percentage: *training/testing*.

Figure 7.8 shows that the best results are given using a 30% training/70% testing split. This splitting ratio leads to discover 508 new phishing domain names out of one million generated. We can see that a training set composed of only 10% of the phishing domains is capable to generate more than 370 new domain names confirmed as malicious. However, bigger training sets to learn the Markov model do not provide better results since fewer malicious domain names are generated with a training set representing more than 50% of the malicious set. This can be explained by the fact that the testing set is smaller, thus fewer generated domain names can be confirmed as malicious according to this set. This shows as well that the Markov model needs few instances of phishing domain names to learn a generator faithfully representing the composition of domain names since few domain names in the training set are needed to generate a lot of new ones. It is worth noting that with a smaller training set, more domain names are discovered faster. The curve representing 10% training/90% testing grows fast, and after only 100,000 generations more than half (217 domain names) of the total count of phishing domain names generated are found. However, only this ratio seems to reach its limit in the count of discovered phishing domain names after one million probes since the curve tends to a horizontal asymptote. This asymptote is due to the low number of domain names in the training set, which explains that all the tracks with the higher probability transitions have already been followed in the Markov chain. It shows some limit of this model to generate a lot of different new domain names. For others split ratio, following the curve's trend, if more probes are performed, a reasonable assumption is that more malicious domain names can be discovered.

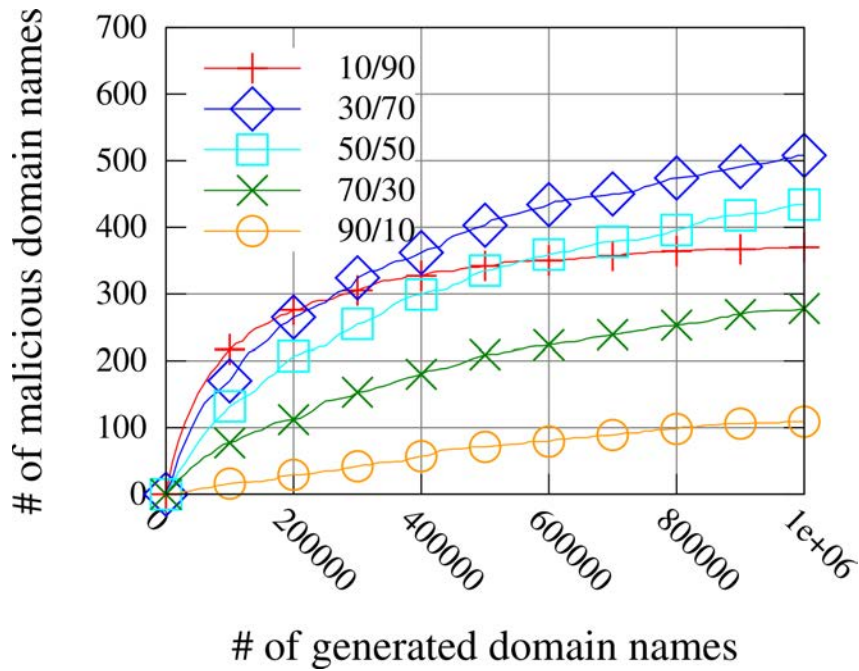


Figure 7.8: Count of malicious domain names generated depending on the total count of generated domain names and on the variation of training/testing split ratio

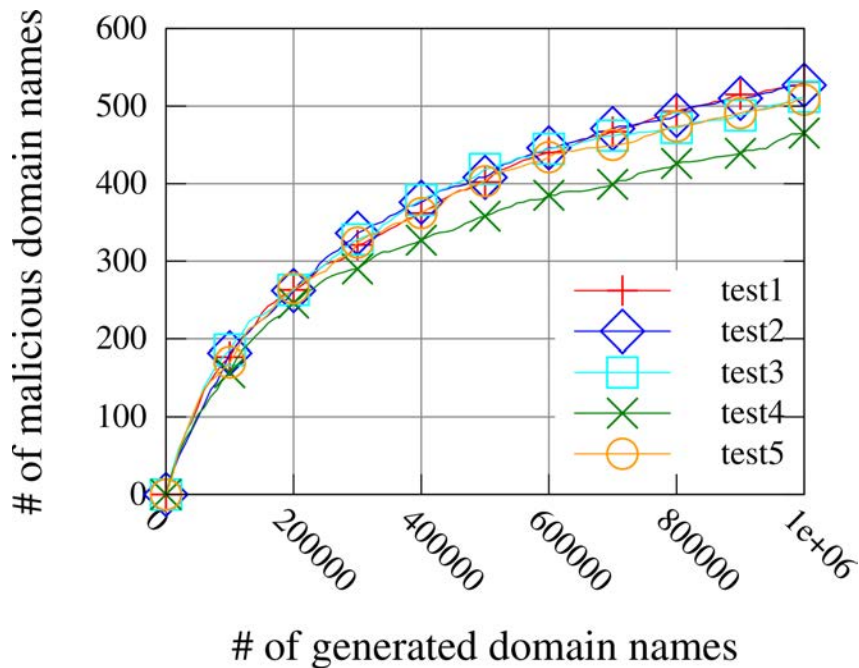


Figure 7.9: Count of malicious domain names generated depending on the total count of generated domain names: five probing campaigns with 30% training/70% testing split ratio

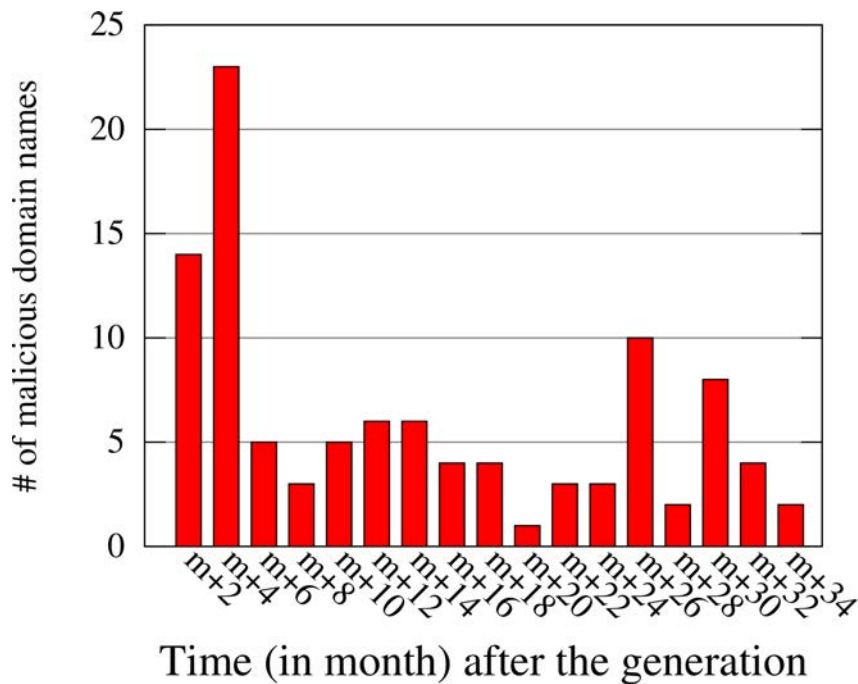


Figure 7.10: Distribution of malicious domain names discovered regarding the time they are blacklisted

To assess the regularity and steadiness of the generation process depending on the composition of the learning set we performed five similar probing campaigns with different learning set. We selected the best split ratio: 30% training/70% testing to perform this experiment. For each campaign we randomly made up different training sets and a testing sets from the malicious set. Figure 7.9 depicts the results of this experiment with the count of malicious domain names generated according to the total count of generated domain names. The five curves depicts the five campaigns with different 30% training/70% testing sets. Observations are similar for every campaigns, which lead to discover around 500 phishing domain names. Moreover, half of the discovered phishing domain names are generated during the first 200,000 generations, highlighting the ability of the system to generate the most likely malicious domain names in priority before being discarded for next generations. This shows that the learning of the composition of domain names is not dependant on the elements contained in the training set and in the testing set.

7.3.3 Predictability and Strategy

This experiment evaluates the time between the date when a malicious domain name can be generated using the generator and the date it is actually used for phishing and blacklisted. The training set is composed of the 10% oldest blacklisted domain names and the remaining 90% belong to the testing set. The testing set represent 34 months of blacklisted domain names and the training set 4 months. We chose this repartition to evaluate both the ability of the method to learn a general generation model from a limited labelled domain set and its ability to generate domain names a long time before they are used. Hence, we needed a small learning set and a testing set covering a long period of time. Figure 7.10 depicts the result of this experiments. All generated domain names were generated based on a malicious domain names set available

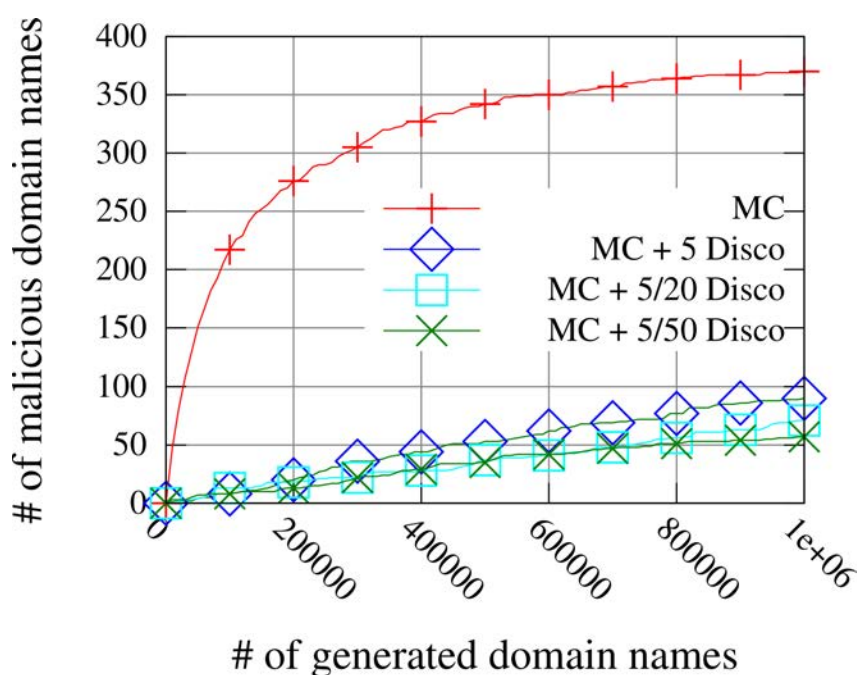


Figure 7.11: Count of discovered malicious domain names depending on the adopted strategy

at $t = 0$. The histogram of Figure 7.10 shows the count of generated domain names that were used for phishing and blacklisted at $t + x$, *i.e.* x months after generation. We see that a large quantity of generated malicious domain names are used during a period of four months after their generation, 14 in the two following months and 23 more in the next two months. This shows that domain names composition follows fashion schemes because more generated malicious domain names are used few time after the ones that are used to train the model. However, it is worth noting that such domain names continue to be used years after these are generated showing that even old datasets can be used to generate current phishing domain names.

We have described in Section 7.1.2 the two core building blocks for generating domain names: the Markov chain model and the semantic extension module. The impact of each module on the generation is assessed according to four strategies described below:

- MC: the Markov chain model without semantic extension is used.
- MC + 5 DISCO: the Markov chain model and for each state of the Markov chain the five most related words ($Disco(w, 5)$) are tested. This scenario is the basic implementation described in Section 7.1.2.
- MC + 5/20 DISCO: the Markov chain model and for each state of the Markov chain five words randomly picked among the twenty most related words ($Disco(w, 20)$) are tested.
- MC + 5/50 DISCO: the Markov chain model and for each state of the Markov chain five words randomly picked among the fifty most related words ($Disco(w, 50)$) are tested.

The objective of this experiment is to identify the best tradeoff between the success rates in discovering rogue domain names depending on the computational effort. Figure 7.11 shows the number of generated malicious domain names depending on the count of probes performed over

a probing campaign. The same training set is used to build the generation model for the four different probing campaigns. It clearly comes out that the Markov chain model alone yields the best results in term of malicious domain names discovered with a total of 370. DISCO strategies are able to generate only between 57 and 90 malicious domain names over these campaigns depending on the technique. However, it is worth noting that between 79% and 85% of these generated domain names are unique, *i.e.* none of the other strategies are able to find them. Even if DISCO strategies are less efficient than the basic Markov chain in term of confirmed malicious domain names per generated domain names, these strategies lead to discover malicious domain names that the basic model is not able to generate. If a global probing is targeted all the part of the generation module must be used in order to discover the maximum of phishing related domain names. However, the Markov chain model is sufficient to find out domain names with minimum generation processes.

Conclusion

After exploring the solutions for phishing domain names identification in real-time in Part II, we explored in this part proactive methods. Chapter 5 presented an efficient technique for identifying phishing URLs in real-time only based on relatedness analysis of single URLs. We explored in Chapter 6 the possibility to predict domain names using semantic properties of domain names naming schemes. This was used in DNS probing applications and showed better results than sate of the art techniques of DNS probing. Based on these first findings assessing the relevancy of using natural language model and semantic analysis to represent domain names, we explored in this chapter the way to apply it to phishing prevention. One main characteristic of phishing campaigns and phishing websites is their short lifetime, making short reaction time to cope with them paramount. Hence, rather than looking for reactive techniques or real-time techniques to prevent phishing, we introduced a proactive phishing prevention technique.

This relies on the structural and lexical analysis of phishing domain names composition. We introduced relevant features capturing the structural composition of domain names and showed that these features have different values when extracted from phishing or legitimate domain names. We showed that words composing phishing domain names belong to a limited vocabulary that is different from the ones used in legitimate domain names. This led to build a natural language model of phishing domain names. This model relies on the introduced features and leverages a Markov chain model and an n-gram analysis. This model is extended with a semantic module and is used as a phishing domain names generation technique to generate domain names likely to be used by phishers in future phishing campaigns. We studied the efficiency of this generation technique based using a set of malicious domain names to build the generation model. The experiments showed that the model is able to generate numerous phishing domain names that are actually used in malicious activities after their generation. Some legitimate domain names were generated as well but a score computed during the generation process in the Markov chain allows to discard most of them. This provides an interesting approach to cope with phishing attacks by preventing connection to malicious link before these are actually used.

The fight against continual threats such as phishing used to rely on reactive techniques. However, the techniques that have been develop to cope with this problem during the preceding decades did not succeed to stem this threat. Phishers just developed new tricks and new subterfuges to bypass the developed protection techniques. Hence, to efficiently treat this problem there is a need to think faster than miscreant and predict in advance the new means they will use to perpetrate their malicious activities. Such a solution is proposed in this chapter with a

proactive blacklisting method able to predict domain names that will be used by phishers. This technique is not foolproof since the malicious domain names generator generates some existing legitimate domain names. Moreover, the main part of the generated domain names is not used and will probably never be. Nevertheless, proactive prevention techniques are likely to be the methods to definitely get rid of this continual threat.

General Conclusion

To cope with the ever growing phishing activities and their consequences, this document described several challenges that researchers have to address in order to reverse this increasing trend by developing phishing protection solutions. These include the need to develop fast detection techniques that can cope with short phishing attacks lifetime and that can be integrated in real-time detection system without introducing delay. A second challenge is to develop phishing protection methods with a large scope in order to deal with most phishing vectors. A third challenge is the reliability of phishing detection techniques, since phishes tend to mimic legitimate contents it is difficult to develop techniques that can reliably identify phishes without misidentifying legitimate contents. The last challenge relies less on technical aspects but more on the usability of developed solutions in order that unsavvy users can understand and easily use the developed protection techniques.

This document does not address all these challenges in order to provide a bulletproof protection against the whole range of phishing attacks. However, it provides some relevant contributions to the fight against phishing, by introducing the use of semantic analysis and word relatedness computation to identify phishing URLs. Even though the semantic analysis was already used for phishing detection, this technique was applied only to content containing a lot of information namely emails and webpages. We introduced in this document a technique to extract the semantic context of domain names and URLs, which are locators containing limited information. Lexical and semantic analysis of URLs has the advantage to rely only on information contained in these entities, meaning that this applies to any phishing attack leveraging URLs. It allows to cover a wide range of phishing attacks since URL are widely used in several phishing attacks. Moreover, this analysis has been proved fast such that it should not impact user experience and is able to detect in real-time phishes. Finally, for some applications including a URL recommendation system, this technique presents high reliability this techniques highlight high reliability that let envision real-world deployment.

The contribution of this document is to introduce the use of lexical and semantic analysis for two applications:

- Phishing domain names and URLs detection
- Semantic based phishing domain names prediction

1 Summary of Contributions

Phishing Domain Names and URLs Detection

We presented two efficient methods to detect phishing domain names and phishing URLs respectively. The first proposed method focused on the identification of phishing and more generally malicious domain names. Since the information contained in single domain names can be limited

for short domain names, we first proposed a technique to cluster domain names according to their activity. This technique computes features from DNS information that was extracted from passively monitored DNS replies. New features including IP scattering measures were introduced in order to depict the dispersion of IP addresses mapped to a single domain name. As shown, these features are highly relevant for distinguishing fluxing domain names from legitimate domain names. Studies showed that flux networks are highly used as support for phishing attacks making their identification relevant to detect phishing attacks. Applying the k-means clustering algorithm on the set of features led to form several clusters of domain names. Using manual analysis, we identified the activities of domain names included in each cluster showing that the DNS-based features we defined were relevant for distinguishing domain names usage including the difference between malicious and legitimate domain names.

In order to automate the labelling process of domain clusters that are formed using DNS-based features, we used the property that domain names are meaningful. Since, malicious domains and fluxing domains are widely used for phishing activities, we explored one characteristic of phishing domain names and URLs namely to be obfuscated. Phishers use their manipulation skills to create domain names and URLs that will lure users by carefully choosing the words that are embedded in their URLs. We presented a sequential technique to extract all the meaningful words embedded in a domain name in order to form set of words from several domain names. We introduced several metrics to compute the semantic relatedness between two sets of words in order to show that words embedded in legitimate domain names belong to different semantic fields than words used in phishing domain names. The evaluation of the introduced semantic similarity metrics showed that our assumptions were correct and that legitimate domain names are different from phishing domain names from a semantic composition point of view. The defined metrics can be used to identify sets of phishing domain names in a computationally efficient manner by comparing unknown set of domains to a labelled legitimate and malicious domains set.

The last contribution of this part was a method using semantic relatedness evaluation of the components that compose a single URL to infer its likelihood to be a phish. From the insights brought by Chapters 4 about the use of specific words and semantic fields in phishing URLs and domain names, the concept of intra-URL relatedness was defined. The computation of features quantifying this relatedness led to identify single phishing URLs with high accuracy. This technique relies only on lexical analysis of URL and the extraction of few features ensuring its rapidity since it infers the phishingness of a URL in less than a second. This technique can operate in real-time meeting the requirement of speed for a phishing detection techniques. It focuses on the identification of phishing URLs meeting as well the requirement of coverage, since a large range of phishing attacks including phishing emails, fake websites, link injection in web pages or drive-by download is covered. Finally the relevancy of the proposed features is assessed by the reliability of the machine learning based phish identification we built based on them. The usage of this in a URL reputation system can reach over 99% accuracy for most URLs.

Semantic Based Phishing Domain Names Prediction

Since time is a paramount parameter in phishing detection and to go further in the early identification of phishes we explored the possibility to predict phishing domain names used by phishers. The applications of such technique are to prevent the malicious registration of domain names or to build predictive blacklists for instance. Witnessing the fact that domain names are meaningful entities, which are composed by humans, we showed that their composition follow semantic similarity patterns that can be learned. For memorisation purposes several subdomains of a single

domain names are semantically related with relationship such as synonymy, antonymy, entailment or hyponymy. Hence, by inferring these relationships using semantic relatedness computation, we proved that the subdomains of a domain name are predictable and presented a method for DNS probing that outperforms the related work relying on predefined dictionary of labels.

The findings of this first research work were further used as a basis to build domain names that are likely to be used for phishing. Studying the composition of maliciously registered domain names, we observed that these are composed of several meaningful words and modeled this composition with a Markov Chain Model. We further showed that words used in legitimate and phishing domain names belong to different semantic fields since these show low semantic relatedness when compared. Exploring this statement, we conclude that using a model of composition extracted from phishing domain names we can generate new domain names having the characteristics of existing phishing domains. The experiments performed with the built phishing domain names generator assessed that it is able to produce domain names that will be used for phishing months or years before these are actually registered and used. This produces as well a limited number of legitimate domain names showing some shortcomings and confirming that phishing domain names are difficult to spot since these mimic the composition of legitimate ones. However, the contributions presented in this document showed that semantic analysis of domain names is a relevant solution to discriminate phishing from legitimate domain names.

Using semantic analysis of the composition of domain names and URLs to protect from phishing provides solutions for three of the four introduced requirements that are speed, reliability and coverage. The presented solutions provide either low latency, for phishing URLs rating, or no latency at all, for predictive methods based on domain names generation models. These are reliable especially for real-time techniques of URL identification reaching >99% accuracy and present large coverage.

2 Research Perspectives

One requirement that is not addressed by the contributions presented in this manuscript is the concrete usability of semantic analysis results by unsavvy users. Even though, it was theoretically assessed efficient, no real world experiment with a sample of test users was carried out. Moreover, the considered test sets for assessment were limited in size and the scalability of the methods was not assessed. Hence, the results presented in this document raise some research perspectives for future work.

Combination of technical solutions with user intervention.

Even though some work [ECH08] already studied the efficiency of different warning solutions to prevent phishing and their capacity to be understood and considered by users. The efficiency of the introduced technique to be used by users and its efficiency depending on the implementation mode as automated blocking system or recommendation system for instance, must be studied. In a similar manner the relevancy of the potential phishing related displayed information on raising user awareness and their ability to teach user must be studied. Automated phishing detection techniques have an essential role in the fight against phishing but due to their subjection to false positives, these must be coupled with user intervention. The fight against phishing can only be won if we make progress in both the development of efficient protection techniques and in raising user awareness. Moreover, these two fields must not be separated in different studies but must

be jointly examined in order to evaluate the impact of each on the other in order to tailor easily usable phishing protection techniques.

Extension with other semantic analysis techniques.

This document proposed the use of lexical and semantic analysis of URLs to detect phishes. Some state of the art techniques of semantic analysis based on Mutual Information computation [Hin90] were used with DISCO [Kol08, Kol09]. Another technique based on the mining of search engine query data was introduced in Chapter 5 to cope with the a priori inefficiency of existing solutions to address the needed task. However, it would be worth applying techniques such as TF-IDF [SM83] or LSA [LD97] to the domain cluster comparison performed in Chapter 4. The former was successfully used in CANTINA [ZHC07] to detect phishing web pages and can possibly give good results if applied to domain names. Additionally, a study of word occurrences and frequencies can be performed for phishing and legitimate domain names composition in order to see if the *phishing* language defined is different from the language usually used to compose legitimate domain names. Other features extracted from domain names and URLs and introduced in state of the art work [BWSW10, LMF11, BSHA14] should be as well integrated to improve the results presented in this document. Specifically, the machine learning based phishing rating system introduced in Chapter 5 can be improved using these features.

Usage of more refined machine learning and natural language processing techniques.

Even though in Chapter 5, we presented the relevancy of the introduced features set according to several supervised machine learning algorithm, the same exhaustive study was not applied to every proposed technique. The clustering method proposed in Chapter 3 relies on a basic Euclidean distance and other distance to group the domain names should be tested such as the Manhattan distance or the Minkowski distance for instance. Other techniques than centroid-based clustering should be tested as well, like distribution-based clustering and density-based clustering. These techniques may be more amenable to group domain names according to their related activities and to separate malicious from legitimate domain names. Similarly, the technique used to model the composition of phishing domain names in Chapter 7 is a basic Markov Chain model. This modelling can be improved by using more more refined models such as a Hidden Markov Model (HMM) or by modelling the phisher's language as a formal grammar.

Correlation of other data sources with semantic relatedness features.

While the main focus of this document is the use of semantic analysis to identify phishes, we used as well DNS information to complement this technique when needed. The correlation of multiple data sources can be worth to improve the accuracy of semantic based phishing protection techniques. Other work already used host-based information, domain name reputation information or leveraged existing blacklist to improve phishes detection accuracy. We explored as well this track by proposing to combine information extracted from URLs to DNS information, honeypot information and IP flow records in order to detect malicious communications [MJSE14]. Such data correlation system can have a wider action scope than just phishing detection but can detect as well network intrusions or malware delivery by monitoring other network communications than HTTP or DNS traffic. This raises as well other challenges through the treatment of the large amount of information to correlate and the use of distributed computing and storage system seems mandatory.

Scalability of phishing protection.

We have seen that the proposed methods of phishing protections rely on data mining and machine learning strategies. During experiments, to assess their relevancy, the introduced methods were applied to small sample dataset containing from tens to hundreds of thousands URLs or domain names. However, with almost one billion online webpages currently, we must consider the scalability issues of the methods to be implemented in real-world scenarios. We already leveraged distributed data storage and processing solutions for DNS data analysis with Hadoop. Similarly, the inference of intra-URL relatedness was performed with streaming analytics solution *e.g.* Storm. However, we did not proceed to large scale evaluation with high usage of these services dealing with millions of instances as it is supposed to be used. In such scenario, even though the used solutions are supposed to scale, some issues can be raised and the conception of new adapted distributed processing and storage solutions may be needed to cope with the workload that may face the system in order to stay usable in real-time conditions. The best way to assess this scalability capability is to deploy for real usage the solutions proposed in this document. It would provide as well results related to the real efficiency of the proposed methods to cope with phishing attacks.

Transversal security applications for semantic relatedness analysis.

This work explores the use of semantic relatedness evaluation and composition of URLs and domain names. The use of semantic analysis of contents is relevant in a phishing context since this is a swindle where phishers use text manipulation to lure their victims. While it was already used to detect phishing webpages and emails, we introduced in this document applications to phishing domain names and URLs. A promising research topic would be to applied similar techniques of word composition and semantic relatedness analysis to other contents like phishing related malware products such as fake antivirus that use partly social engineering to trap users in buying fake upgrade. Others fields that use meaningful word representation to identify resources such Content Centric Networking and Named Data Networking can leverage semantic analysis to identify malicious contents as it is done for phishing URLs and domain names.

List of Figures

1	Phishing attacks and phishing domain names recorded every year (source:APWG)	2
1.1	Phishing vectors classification	14
1.2	Most targeted industry sectors for the 3 rd quarter 2014 (source: APWG)	15
1.3	Example of security toolbar (Netcraft toolbar)	19
2.1	Domain Name Space hierarchy and path to the node <i>snt</i>	32
2.2	DNS resolution of the domain name <i>www.inria.fr</i>	32
2.3	Phishing web site hosting using a double flux network	37
2.4	Different DNS probing locations	40
3.1	Passive DNS Monitoring Architecture	51
3.2	Computation steps for S_{ip1}	55
3.3	Features statistics	60
3.4	WCSS values for 2-15 clusters obtained with k-means	63
3.5	Feature values according to cluster number	64
4.1	Malicious domain set identification: architecture overview	71
4.2	Word extraction for <i>securelogin34ebay.com.my-securephishing-domain.co.uk</i>	72
4.3	Unlabelled domain set identification process	76
4.4	Similarity score Sim_3 depending on the count of domain names in the set	81
4.5	Similarity score Sim_3 depending on the proportion of malicious domain names in the set	81
5.1	Word extraction for <i>securelogin34ebay.com.my-securephishing-domain.co.uk</i>	91
5.2	Distributed URL processing with Storm topology	93
5.3	Repartition of the count of embedded words per URL	94
5.4	Box-and-whisker diagram for Jaccard based features (min/max)	96
5.5	Phishing classification results for seven classifiers	99
5.6	ROC curve for Random Forest classification	100
5.7	Phishing and legitimate URL partition according to rating ranges	100
6.1	Semantic DNS probing system overview	107
6.2	Horizontal and vertical exploration for <i>surf.apple.com</i>	111
6.3	Ratio of newly generated valid subdomains depending on lim_h .	115
6.4	Cumulative frequency graph of discovered subdomains depending on lim_v .	116
6.5	Percentage ($\%Imp_i$) of newly discovered subdomains for each domain names	117
6.6	Count ($ New_i $) of newly discovered subdomains for each domain names	118
6.7	Count of newly discovered subdomains regarding the strategy used.	119
6.8	Count of DNS requests made per domain name to probe.	120

6.9	Count of DNS request made per subdomain in the initial dataset.	120
6.10	Ratio of newly discovered subdomain per probe.	121
7.1	Overview of the phishing domain names generation and validation process	125
7.2	Markov chain model example	128
7.3	Markov chain semantic extension	128
7.4	Main level domain generator	129
7.5	$distwlen_n \mid n \in \{2; 10\}$ for malicious and legitimate domain names	130
7.6	Proportion and type of existing generated domain names.	134
7.7	Cumulative frequency graph of domain names depending on their <i>MCscore</i>	134
7.8	Count of malicious domain names generated depending on the total count of generated domain names and	
7.9	Count of malicious domain names generated depending on the total count of generated domain names: f	
7.10	Distribution of malicious domain names discovered regarding the time they are blacklisted	138
7.11	Count of discovered malicious domain names depending on the adopted strategy	139

List of Tables

1.1	Phishing vectors targeted by phishing protection methods	24
1.2	The extent to which phishing protection methods meet requirements	26
2.1	DNS related malicious activities and the techniques to detect them	45
3.1	Expected feature values depending on domain names activity	57
3.2	Passive DNS capture statistics for dataset Luxembourg and dataset France . . .	58
3.3	<i>IPCCount</i> and <i>SubDom</i> features for some domain names	59
3.4	Feature values repartition	61
4.1	Example of obfuscated URLs for the domain name <i>paypal.com</i>	69
4.2	Example of co-occurrence count (2 windows centered on <i>services</i>)	74
4.3	Values of Sim_1 computed between malicious and legitimate subsets	78
4.4	Values of $Sim_2 \times 10^3$ computed between malicious and legitimate subsets	79
4.5	Values of Sim_3 computed between malicious and legitimate subsets	80
5.1	Subset of most phishing targeted brands with <i>mld</i> & <i>mld.ps</i>	88
5.2	Count of labels matching at least one related word for 4 tools	88
5.3	Example of term results from Google Trends and Yahoo Clues for $\{paypal\}$. . .	90
5.4	Intra-URL relatedness features description	91
5.5	Statistical values of features extracted from legitimate and phishing datasets . . .	97
5.6	Information Gain values for the 12 features	98
5.7	Confidence interval for classification results	99
5.8	Detailed classification results for Random Forest (threshold = 0.76)	100
6.1	Probing results for 24 domain names	119
7.1	Example of Markov chain transitions for the state <i>pay</i>	127
7.2	Hellinger Distance for <i>ps</i> (leg=legitimate, mal=malicious)	132
7.3	Hellinger Distance for words in <i>mlds</i> (leg=legitimate, mal=malicious)	133
7.4	Markov chain statistics for main level domain model	133

Bibliography

- [AA04] Derek Atkins and Rob Austein. Rfc 3833: Threat analysis of the domain name system (dns), 2004.
- [AAL⁺05] Roy Arends, Rob Austein, Matt Larson, Dan Massey, and Scott Rose. Rfc 4033: Dns security introduction and requirement, 2005.
- [ADL⁺10] Manos Antonakakis, David Dagon, Xiapu Luo, Roberto Perdisci, Wenke Lee, and Justin Bellmor. A centralized monitoring infrastructure for improving DNS security. In *Proceedings of the 13th Symposium on Recent Advances in Intrusion Detection*, Lecture Notes in Computer Science, pages 18–37. Springer Berlin Heidelberg, 2010.
- [aka] Akamai Content Delivery Network. <http://www.akamai.com/> (last visited on 2015-04-21).
- [AKM⁺11] Mikhail Afanasyev, Tadayoshi Kohno, Justin Ma, Nick Murphy, Stefan Savage, Alex C. Snoeren, and Geoffrey M. Voelker. Privacy-preserving network forensics. *Communications of the ACM*, 54(5):78–87, 2011.
- [AKS14] Shivam Aggarwal, Vishal Kumar, and S. D. Sudarsan. Identification and detection of phishing emails using natural language processing techniques. In *Proceedings of the 7th International Conference on Security of Information and Networks*, SIN '14, pages 217:217–217:222. ACM, 2014.
- [ale] Alexa WebSites ranking. <http://www.alexa.com/> (last visited on 2015-04-21).
- [APD⁺10] Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. Building a dynamic reputation system for dns. In *Proceedings of the 19th USENIX Security Symposium*, SEC '10, pages 18–18. USENIX Association, 2010.
- [APL⁺11] Manos Antonakakis, Roberto Perdisci, Wenke Lee, Nikolaos Vasiloglou, II, and David Dagon. Detecting malware domains at the upper dns hierarchy. In *Proceedings of the 20th USENIX Security Symposium*, SEC '11, pages 1–16. USENIX Association, 2011.
- [APN⁺12] Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. From throw-away traffic to bots: Detecting the rise of dga-based malware. In *Proceedings of the 21st USENIX Conference on Security Symposium*, SEC '12, pages 24–24. USENIX Association, 2012.

- [apw04] Phishing Activity Trends Report: January, 2004. Technical Report January, 2004, APWG, 2004.
- [apw14] Phishing Activity Trends Report: 2nd Quarter 2014. Technical Report 2Q2014, APWG, 2014.
- [AR14] Greg Aarin and Rod Rasmussen. Global Phishing Survey 1H2014: Trends and Domain Name Use. Technical Report 1H2014, APWG, 2014.
- [Ayc12] John Aycok. What’s in a name. . . generator? *Journal in Computer Virology*, 8(1-2):53–60, 2012.
- [BCN01] Nevil Brownlee, Kc Claffy, and Evi Nemeth. Dns measurements at a root server. In *Proceedings of the Global Telecommunications Conference*, GLOBECOM ’01, pages 1672–1676. IEEE, 2001.
- [BCP⁺08] Andre Bergholz, Jeong Ho Chang, Gerhard Paaß, Frank Reichartz, and Siehyun Strobel. Improved phishing detection using model-based features. In *Proceedings of the 5th Conference on Email and Anti-Spam*, CEAS ’08, pages 1–10, 2008.
- [BG10a] Kenton Born and David Gustafson. Detecting dns tunnels using character frequency analysis. *arXiv preprint*, 2010.
- [BG10b] Kenton Born and David Gustafson. Ngviz: Detecting dns tunnels through n-gram visualization and quantitative analysis. In *Proceedings of the Annual Workshop on Cyber Security and Information Intelligence Research*, CSIIRW ’10, pages 47:1–47:4. ACM, 2010.
- [BH06] Alexander Budanitsky and Graeme Hirst. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47, 2006.
- [BJKP14] Bastian Braun, Martin Johns, Johannes Koestler, and Joachim Posegga. Phish-safe: Leveraging modern javascript api’s for transparent and robust protection. In *Proceedings of the 4th ACM Conference on Data and Application Security and Privacy*, CODASPY ’14, pages 61–72. ACM, 2014.
- [BKKB11] Leyla Bilge, Engin Kirda, Christopher Kruegel, and Marco Balduzzi. EXPOSURE: Finding malicious domains using passive DNS analysis. In *Proceedings of the 18th Annual Network and Distributed System Security Symposium*, NDSS ’11. Internet Society, 2011.
- [Blo70] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.
- [BMM⁺12] Ignacio N. Bermudez, Marco Mellia, Maurizio M. Munafò, Ram Keralapura, and Antonio Nucci. Dns to the rescue: Discerning content and services in a tangled web. In *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, IMC ’12, pages 413–426. ACM, 2012.
- [BN12] Andreas Berger and Eduard Natale. Assessing the real-world dynamics of dns. In *Proceedings of the 4th International Conference on Traffic Monitoring and Analysis*, TMA ’12, pages 1–14. Springer-Verlag, 2012.

-
- [BNC03] Andre Broido, Evi Nemeth, and Kc Claffy. Spectroscopy of dns update traffic. In *Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '03, pages 320–321. ACM, 2003.
- [Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [BSHA14] Phoebe A. Barraclough, Graham Sexton, Alamgir Hossain, and Nauman Aslam. Intelligent phishing detection parameter framework for e-banking transactions based on neuro-fuzzy. In *Proceedings of the Science and Information Conference*, pages 545–555. IEEE, 2014.
- [BWSW10] Aaron Blum, Brad Wardman, Thamar Solorio, and Gary Warner. Lexical feature based phishing url detection using online learning. In *Proceedings of the 3rd ACM Workshop on Artificial Intelligence and Security*, AISEC '10, pages 54–60. ACM, 2010.
- [CDM10] Teh-Chung Chen, Scott Dick, and James Miller. Detecting visually similar web pages: Application to phishing detection. *ACM Transactions on Internet Technology*, 10(2):5:1–5:38, 2010.
- [CGD08] Debra L. Cook, Vijay K. Gurbani, and Michael Daniluk. Phishwish: A stateless phishing filter using minimal rules. In *Proceedings of the Conference on Financial Cryptography and Data Security*, Lecture Notes in Computer Science, pages 182–186. Springer Berlin Heidelberg, 2008.
- [CH90] Kenneth W. Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.
- [cis] The Internet of Things. <http://share.cisco.com/internet-of-things.html> (last visited on 2015-04-21).
- [CKV08] Marco Cova, Christopher Kruegel, and Giovanni Vigna. There is no free phish: An analysis of "free" and live phishing kits. In *Proceedings of the USENIX Workshop on Offensive Technologies*, WOOT '08. USENIX Association, 2008.
- [CL12] Hyunsang Choi and Heejo Lee. Identifying botnets by capturing group activities in {DNS} traffic. *Computer Networks*, 56(1):20–33, 2012.
- [CLLK07] Hyunsang Choi, Hanwoo Lee, Heejo Lee, and Hyogon Kim. Botnet detection by monitoring group activities in dns traffic. In *Proceedings of the 7th IEEE International Conference on Computer and Information Technology*, CIT '07, pages 715–720. IEEE, 2007.
- [clo] Amazon CloudFront CDN. <http://aws.amazon.com/cloudfront/> (last visited on 2015-04-21).
- [CLTM04] Neil Chou, Robert Ledesma, Yuka Teraguchi, and John C. Mitchell. Client-side defense against web-based identity theft. In *Proceedings of the 11th Annual Network and Distributed System Security Symposium*, NDSS '04, pages 1–16, 2004.

- [CSDM14] Teh-Chung Chen, Torin Stepan, Scott Dick, and James Miller. An anti-phishing system employing diffused information. *ACM Transactions on Information and System Security*, 16(4):16:1–16:31, 2014.
- [CTD⁺12] Alper Caglayan, Mike Toothaker, Dan Drapeau, Dustin Burke, and Gerry Eaton. Behavioral analysis of botnets for threat intelligence. *Information Systems and e-Business Management*, 10(4):491–519, 2012.
- [CV05] Rudi L. Cilibrasi and Paul M. B. Vitanyi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005.
- [CV07] Rudi L. Cilibrasi and Paul M. B. Vitanyi. The google similarity distance. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):370–383, 2007.
- [CWFC08] Sebastian Castro, Duane Wessels, Marina Fomenkov, and Kimberly Claffy. A day at the root of the internet. *SIGCOMM Computer Communication Review*, 38(5):41–46, 2008.
- [DG04] Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. In *Proceedings of the 6th Symposium on Operating Systems Design & Implementation (OSDI)*, pages 137–150. USENIX Association, 2004.
- [dmoz] DMOZ - the Open Directory Project. <http://www.dmoz.org/> (last visited on 2015-04-21).
- [dns] DNSenum. <https://code.google.com/p/dnsenum/> (last visited on 2015-04-21).
- [DOK92] Peter B. Danzig, Katia Obraczka, and Anant Kumar. An analysis of wide-area name server traffic: A study of the internet domain name system. In *Proceedings of the ACM SIGCOMM Conference on Data Communication*, SIGCOMM '92, pages 281–292. ACM, 1992.
- [DRNDJ13] Philippe De Ryck, Nick Nikiforakis, Lieven Desmet, and Wouter Joosen. Tabshots: Client-side detection of tabnabbing attacks. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ASIACCS '13, pages 447–456. ACM, 2013.
- [DT05] Rachna Dhamija and J. Doug Tygar. The battle against phishing: Dynamic security skins. In *Proceedings of the 2005 Symposium on Usable Privacy and Security*, SOUPS '05, pages 77–88. ACM, 2005.
- [DTH06] Rachna Dhamija, J. Doug Tygar, and Marti Hearst. Why phishing works. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, CHI '06, pages 581–590. ACM, 2006.
- [DTMV12] Luca Deri, Lorenzo Luconi Trombacchi, Maurizio Martinelli, and Daniele Vannozi. Towards a passive dns monitoring system. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 629–630. ACM, 2012.
- [eba] eBay Toolbar. <http://pages.ebay.com.au/securitycentre/ebay-toolbar.html> (last visited on 2015-04-21).

-
- [ECH08] Serge Egelman, Lorrie F. Cranor, and Jason Hong. You've been warned: An empirical study of the effectiveness of web browser phishing warnings. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1065–1074. ACM, 2008.
- [fie] Fierce Domain Scan - RSnake. <http://ha.ckers.org/fierce/> (last visited on 2015-04-21).
- [FKP10] Mark Felegyhazi, Christian Kreibich, and Vern Paxson. On the potential of proactive domain blacklisting. In *Proceedings of the 3rd USENIX Conference on Large-scale Exploits and Emergent Threats*, LEET '10, pages 6–6. USENIX Association, 2010.
- [FM14] Mohammed Nazim Feroz and Susan Mengel. Examination of data, rule generation and detection of phishing url using online logistic regression. In *Proceedings of the IEEE Conference on Big Data*, BigData '14, pages 241–250. IEEE, 2014.
- [FS10] George Forman and Martin Scholz. Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement. *SIGKDD Exploration Newsletter*, 12(1):49–57, 2010.
- [FST07] Ian Fette, Norman Sadeh, and Anthony Tomasic. Learning to detect phishing emails. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 649–656. ACM, 2007.
- [gar07] Gartner survey shows phishing attacks escalated in 2007. Technical report, Gartner Research, 2007.
- [GLLA07] Mohamed G. Gouda, Alex X. Liu, Lok M. Leung, and Mohamed A. Alam. SPP: An anti-phishing single password protocol. *Computer Networks*, 51(13):3715–3726, 2007.
- [gooa] Google Docs. <https://docs.google.com/> (last visited on 2015-04-21).
- [goob] Google Safe Browsing. <https://developers.google.com/safe-browsing/> (last visited on 2015-04-21).
- [gooc] Google Trends. <http://www.google.com/trends/> (last visited on 2015-04-21).
- [GPCR07] Sujata Garera, Niels Provos, Monica Chew, and Aviel D Rubin. A framework for detection and measurement of phishing attacks. In *Proceedings of the 2007 ACM Workshop On Recurring Malcode*, WORM '07, pages 1–8. ACM, 2007.
- [GPGL11] Sophie Gastellier-Prevost, Gustavo Gonzalez Granadillo, and Maryline Laurent. Decisive heuristics to differentiate legitimate from phishing sites. In *Proceedings of the conference on network and information systems security*, SAR-SSI '11, 2011.
- [GSMyG⁺11] Binod Gyawali, Thamar Solorio, Manuel Montes-y Gómez, Bradley Wardman, and Gary Warner. Evaluating a semisupervised approach to phishing url identification in a realistic scenario. In *Proceedings of the 8th Conference on Email and Anti-Spam*, CEAS '11, pages 176–183, 2011.

- [HA13] Isredza R. A. Hamid and Jemal H. Abawajy. Profiling phishing email based on clustering approach. In *Proceedings of the 12th Conference on Trust, Security and Privacy in Computing and Communications*, TrustCom '13, pages 628–635, 2013.
- [hba] Apache HBase. <http://hbase.apache.org/> (last visited on 2015-04-21).
- [HCNK⁺14] Seth Hardy, Masashi Crete-Nishihata, Katharine Kleemola, Adam Senft, Byron Sonne, Greg Wiseman, Phillipa Gill, and Ronald J. Deibert. Targeted threat index: Characterizing and quantifying politically-motivated targeted malware. In *Proceedings of the 23rd USENIX Security Symposium*, SEC '14, pages 527–541. USENIX Association, 2014.
- [HEH⁺09] Mark Hall, Frank Eibe, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.
- [HF08] Cormac Herley and Dinei Florêncio. A profitless endeavor: Phishing as tragedy of the commons. In *Proceedings of the 2008 Workshop on New Security Paradigms*, NSPW '08, pages 59–70. ACM, 2008.
- [HFP11] Shuang Hao, Nick Feamster, and Ramakant Pandrangi. Monitoring the initial dns behavior of malicious domains. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 269–278. ACM, 2011.
- [HGRF08] Thorsten Holz, Christian Gorecki, Konrad Rieck, and Felix C Freiling. Measuring and detecting fast-flux service networks. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium*, NDSS '08, pages 125–134. Internet Society, 2008.
- [HHW⁺10] Cheng Huang, Nick Holt, Y. Angela Wang, Albert Greenberg, Jin Li, and Keith W. Ross. A DNS reflection method for global traffic management. In *Proceedings of the USENIX annual technical conference*, USENIXATC '10, pages 1–6. USENIX Association, 2010.
- [Hin90] Donald Hindle. Noun classification from predicate-argument structures. In *Proceedings of the 28th Annual Meeting on Association for Computational Linguistics*, ACL'90, pages 268–275. Association for Computational Linguistics, 1990.
- [HJ08] Amir Herzberg and Ahmad Jbara. Security and identification indicators for browsers against spoofing and phishing attacks. *ACM Transactions on Internet Technology*, 8(4):16:1–16:36, 2008.
- [HW79] John A. Hartigan and Manchek A. Wong. Algorithm AS 136: A k-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.
- [HWLR08] Cheng Huang, Angela Wang, Jin Li, and Keith W. Ross. Measuring and evaluating large-scale CDNs (Paper withdrawn at Mirosoft's request). In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, IMC '08, pages 15–29. ACM, 2008.

-
- [int15] Number of worldwide internet users from 2000 to 2014. <http://www.statista.com/statistics/273018/number-of-internet-users-worldwide/> (last visited on 2015-04-21), 2015.
- [isc] Internet Systems Consortium - Security Information Exchange. <http://www.isc.org/> (last visited on 2015-04-21).
- [Jai10] Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.
- [JJJM07] Tom N. Jagatic, Nathaniel A. Johnson, Markus Jakobsson, and Filippo Menczer. Social phishing. *Communications of the ACM*, 50(10):94–100, 2007.
- [JS04] Jaeyeon Jung and Emil Sit. An empirical study of spam traffic and the use of dns black lists. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, IMC '04, pages 370–375. ACM, 2004.
- [JSBM02] Jaeyeon Jung, Emil Sit, Hari Balakrishnan, and Robert Morris. Dns performance and the effectiveness of caching. *IEEE/ACM Transactions on Networking*, 10(5):589–603, 2002.
- [Kam04] Dan Kaminsky. Ozymandns: a dns tunnel. <http://en.cship.org/wiki/OzymanDNS> (last visited on 2015-04-21), 2004.
- [KFMK05] Abhinav Kamra, Hanhua Feng, Vishal Misra, and Angelos D. Keromytis. The effect of DNS delays on worm propagation in an IPv6 Internet. In *Proceedings of IEEE Infocom*, INFOCOM '05, pages 2405–2414. IEEE, 2005.
- [KIJ11] Mahmoud Khonji, Youssef Iraqi, and Andrew Jones. Lexical url analysis for discriminating phishing and legitimate websites. In *Proceedings of the 8th Conference on Email and Anti-Spam*, CEAS '11, pages 109–115, 2011.
- [KIJ13] Mahmoud Khonji, Youssef Iraqi, and Andrew Jones. Phishing detection: A literature survey. *IEEE Communications Surveys & Tutorials*, 15(4):2091–2121, 2013.
- [Kil03] Adam Kilgariff. Thesauruses for natural language processing. In *Proceedings of the International Conference on Natural Language Processing and Knowledge Engineering*, pages 5–13. IEEE, 2003.
- [Kit14] Scott Kitterman. Rfc 7208: Sender policy framework (spf) for authorizing use of domains in email, version 1, 2014.
- [Kol08] Peter Kolb. DISCO: A Multilingual Database of Distributionally Similar Words. In *Proceedings of KONVENS 2008 – Ergänzungsband: Textressourcen und lexikalisches Wissen*, pages 37–44, 2008.
- [Kol09] Peter Kolb. Experiments on the difference between semantic similarity and relatedness. In *Proceedings of the 17th Nordic Conference of Computational Linguistics*, NODALIDA '09, pages 81–88. Northern European Association for Language Technology, 2009.

- [KRA⁺07] Ponnurangam Kumaraguru, Yong Rhee, Alessandro Acquisti, Lorrie Faith Cranor, Jason Hong, and Elizabeth Nunge. Protecting people from phishing: The design and evaluation of an embedded training email system. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 905–914. ACM, 2007.
- [Kuc14] Murray Kucherawy. Rfc 7372: Email authentication status codes, 2014.
- [LD97] Thomas K. Landauer and Susan T. Dumais. A solution to plato’s problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211–240, 1997.
- [LD10] Jimmy Lin and Chris Dyer. *Data-Intensive Text Processing with MapReduce (Synthesis Lectures on Human Language Technologies)*. Morgan and Claypool Publishers, 2010.
- [Ler10] Reuven M. Lerner. At the forge: Redis. *Linux Journal*, 2010.
- [Lin98] Dekang Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th International Conference on Computational Linguistics*, COLING '98, pages 768–774. Association for Computational Linguistics, 1998.
- [LLCT14] Lung-Hao Lee, Kuei-Ching Lee, Hsin-Hsi Chen, and Yuen-Hsien Tseng. Poster: Proactive blacklist update for anti-phishing. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 1448–1450. ACM, 2014.
- [LLF⁺11] He Liu, Kirill Levchenko, Márk Félégyházi, Christian Kreibich, Gregor Maier, Geoffrey M Voelker, and Stefan Savage. On the effects of registrarlevel intervention. In *Proceedings of the 4th Usenix Workshop on Large-Scale Exploits and Emergent Threats*, LEET '11. USENIX Association, 2011.
- [LM10] Avinash Lakshman and Prashant Malik. Cassandra: A decentralized structured storage system. *ACM SIGOPS Operating System Review*, 44(2):35–40, 2010.
- [LMF11] Anh Le, Athina Markopoulou, and Michalis Faloutsos. PhishDef: URL names say it all. In *Proceedings of IEEE Infocom*, INFOCOM '11, pages 191–195. IEEE, 2011.
- [LMKK07] Christian Ludl, Sean McAllister, Engin Kirda, and Christopher Kruegel. On the effectiveness of techniques to detect phishing sites. In *Proceedings of Detection of Intrusions and Malware, and Vulnerability Assessment*, DIMVA '07, pages 20–39. Springer, 2007.
- [LSZ02] Richard Liston, Sridhar Srinivasan, and Ellen Zegura. Diversity in dns performance measures. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement*, IMW '02, pages 19–31. ACM, 2002.
- [LXP⁺11] Gang Liu, Guang Xiang, Bryan A. Pendleton, Jason I. Hong, and Wenyin Liu. Smartening the crowds: Computational techniques for improving human verification to fight phishing scams. In *Proceedings of the Seventh Symposium on Usable Privacy and Security*, SOUPS '11, pages 8:1–8:13. ACM, 2011.

-
- [mala] DNS-BH - Malware Domain Blocklist. <http://www.malwaredomains.com> (last visited on 2015-04-21).
- [malb] Malware Domain List. <http://www.malwaredomainlist.com> (last visited on 2015-04-21).
- [Mar71] Andrey Markov. Extension of the Limit Theorems of Probability Theory to a Sum of Variables Connected in a Chain. In *Dynamic Probabilistic Systems (Volume I: Markov Models)*, chapter Appendix B, pages 552–577. John Wiley & Sons, Inc., 1971.
- [MD88] Paul Mockapetris and K. J. Dunlap. Development of the domain name system. In *Proceedings of the ACM SIGCOMM Conference on Data Communication*, SIGCOMM '88, pages 123–133. ACM, 1988.
- [ME12] Samuel Marchal and Thomas Engel. Large scale DNS analysis. In *Proceedings of the 6th IFIP International Conference on Autonomous Infrastructure, Management, and Security, and Vulnerability Assessment*, AIMS '12, pages 151–154. Springer-Verlag, 2012.
- [MFSE12a] Samuel Marchal, Jérôme François, Radu State, and Thomas Engel. Proactive discovery of phishing related domain names. In *Research in Attacks, Intrusions, and Defenses*, RAID '12, pages 190–209. Springer-Verlag, 2012.
- [MFSE12b] Samuel Marchal, Jérôme François, Radu State, and Thomas Engel. Semantic based DNS forensics. In *Proceedings of the International Workshop on Information Forensics and Security*, WIFS '12, pages 91–96. IEEE, 2012.
- [MFSE14a] Samuel Marchal, Jérôme François, Radu State, and Thomas Engel. PhishScore: hacking phishers' minds. In *Proceedings of the 10th International Conference on Network and Service Management*, CNSM '14, pages 46–54, 2014.
- [MFSE14b] Samuel Marchal, Jérôme François, Radu State, and Thomas Engel. PhishStorm: Detecting phishing with streaming analytics. *IEEE Transactions on Network and Service Management*, 11(4):458–471, December 2014.
- [MFW⁺12] Samuel Marchal, Jérôme François, Cynthia Wagner, Radu State, Alexandre Du-launoy, Thomas Engel, and Olivier Festor. DNSSM: A large scale passive DNS security monitoring framework. In *Proceedings of the Network Operations and Management Symposium*, NOMS '12, pages 988–993. IEEE, 2012.
- [MFWE12] Samuel Marchal, Jérôme François, Cynthia Wagner, and Thomas Engel. Semantic exploration of DNS. In *Proceedings of NETWORKING 2012*, pages 370–384. Springer-Verlag, 2012.
- [MG08] D. Kevin McGrath and Minaxi Gupta. Behind phishing: An examination of phisher modi operandi. In *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, LEET '08. USENIX Association, 2008.
- [mic] Microsoft SmartScreen Filter. <http://windows.microsoft.com/en-us/internet-explorer/products/ie-9/features/smartscreen-filter> (last visited on 2015-04-21).

- [Mil95] George A. Miller. WordNet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
- [MJSE14] Samuel Marchal, Xiuyan Jiang, Radu State, and Thomas Engel. A big data architecture for large scale security monitoring. In *Proceedings of the IEEE International Congress on Big Data*, BigData Congress'14, pages 56–63. IEEE, 2014.
- [MKK08] Eric Medvet, Engin Kirda, and Christopher Kruegel. Visual-similarity-based phishing detection. In *Proceedings of the 4th International Conference on Security and Privacy in Communication Networks*, SecureComm '08, pages 22:1–22:6. ACM, 2008.
- [MMN08] John McHugh, Ron McLeod, and Vagishwari Nagaonkar. Passive network forensics: behavioural classification of network hosts based on connection patterns. *ACM SIGOPS Operating Systems Review*, 42(3):99–111, 2008.
- [MNPS07] Loizos Michael, Wolfgang Nejdl, Odysseas Papapetrou, and Wolf Siberski. Improving distributed join efficiency with extended bloom filter operations. In *Proceedings of the 21st International Conference on Advanced Information Networking and Applications*, pages 187–194, 2007.
- [Moc87a] Paul Mockapetris. Rfc 1034: Domain names - concepts and facilities, 1987.
- [Moc87b] Paul Mockapetris. Rfc 1035: Domain names - implementation and specification, 1987.
- [MS99] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, 1999.
- [MSSV09a] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Beyond blacklists: Learning to detect malicious web sites from suspicious urls. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 1245–1254. ACM, 2009.
- [MSSV09b] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Identifying suspicious urls: An application of large-scale online learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 681–688. ACM, 2009.
- [MTM14] Rami M. Mohammad, Fadi A. Thabtah, and Lee McCluskey. Predicting phishing websites based on self-structuring neural network. *Neural Computing and Applications*, 25(2):443–458, 2014.
- [net] Netcraft Toolbar. <http://toolbar.netcraft.com/> (last visited on 2015-04-21).
- [net15] Web Server Survey. Technical Report Januray 2015, Netcraft, 2015.
- [NMAM09] Nikos Nikiforakis, Andreas Makridakis, Elias Athanasopoulos, and Evangelos P. Markatos. Alice, what did you do last time? fighting phishing using past activity tests. In *Proceedings of the 3rd European Conference on Computer Network Defense*, pages 107–117. Springer Verlag, 2009.

-
- [Oll05] Gunter Ollmann. The phishing guide - understanding & preventing phishing attacks. Technical report, Next Generation Security Software Ltd., 2005.
- [PAS⁺04] Jeffrey Pang, Aditya Akella, Anees Shaikh, Balachander Krishnamurthy, and Srinivasan Seshan. On the responsiveness of dns-based network control. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, IMC '04, pages 21–26. ACM, 2004.
- [PB07] Al-Mukaddim K. Pathan and Rajkumar Buyya. A taxonomy and survey of content delivery networks. Technical report, Grid Computing and Distributed Systems Laboratory - University of Melbourne, 2007.
- [PB08] David Plonka and Paul Barford. Context-aware clustering of DNS query traffic. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, IMC '08, pages 217–230. ACM, 2008.
- [PCDL09] Roberto Perdisci, Iginio Corona, David Dagon, and Wenke Lee. Detecting malicious flux service networks through passive analysis of recursive DNS traces. In *Proceedings of the Annual Computer Security Applications Conference*, ACSAC '09, pages 311–320. IEEE, 2009.
- [phi] PhishTank: Out of the Net, into the Tank. <http://www.phishtank.com/> (last visited on 2015-04-21).
- [PKKG10] Pawan Prakash, Manish Kumar, Ramana R. Kompella, and Minaxi Gupta. Phish-Net: Predictive blacklisting to detect phishing attacks. In *Proceedings of IEEE Infocom*, INFOCOM '10, pages 1–5. IEEE, 2010.
- [PKP06] Bryan Parno, Cynthia Kuo, and Adrian Perrig. Phoolproof phishing prevention. In *Proceedings of Financial Cryptography and Data Security*, Lecture Notes in Computer Science, pages 1–19. Springer Berlin Heidelberg, 2006.
- [pon14] Consumers' perception about privacy & security: Do they still care ? Technical Report October 2014, Ponemon Institute, 2014.
- [pub] Public Suffix List. <https://publicsuffix.org/list/> (last visited on 2015-04-21).
- [RJM⁺05] Blake Ross, Collin Jackson, Mike Miyake, Dan Boneh, and John C. Mitchell. Stronger password authentication using browser extensions. In *Proceedings of the 14th USENIX Security Symposium*, SEC '05, pages 527–541. USENIX Association, 2005.
- [RKG06] Pin Ren, John Kristoff, and Bruce Gooch. Visualizing dns traffic. In *Proceedings of the 3rd International Workshop on Visualization for Computer Security*, VizSEC '06, pages 23–30. ACM, 2006.
- [RMTP08] Moheeb Abu Rajab, Fabian Monrose, Andreas Terzis, and Niels Provos. Peeking through the cloud: Dns-based estimation and its applications. In *Proceedings of the International Conference on Applied Cryptography and Network Security*, pages 21–38. Springer Berlin Heidelberg, 2008.
- [rsa14] RSA Fraud Report - 2013 a year in review. Technical Report January 2014, EMC² - RSA, 2014.

- [RW12] Venkatesh Ramanathan and Harry Wechsler. phishgillnet-phishing detection using probabilistic latent semantic analysis. *EURASIP Journal on Information Security*, pages 1–22, 2012.
- [SBJ08] Sushant Sinha, Michael Bailey, and Farnam Jahanian. Shades of grey: On the effectiveness of reputation-based "blacklists". In *Proceedings of the 3rd International Conference on Malicious and Unwanted Software, MALWARE '2008*, pages 57–64. IEEE, 2008.
- [SCH⁺09] Steve Sheng, Lorrie F. Cranor, Jason Hong, Brad Wardman, Gary Warner, and Chengshan Zhang. An empirical analysis of phishing blacklists. In *Proceedings of the 6th Conference on Email and Anti-Spam, CEAS '09*, pages 1–10, 2009.
- [SH09] Toby Segaran and Jeff Hammerbacher. *Beautiful Data: The Stories Behind Elegant Data Solutions*, chapter 14. O'Reilly Media, 2009.
- [Sha48] Claude E. Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27:379–423,623–656, 1948.
- [SHK⁺10] Steve Sheng, Mandy Holbrook, Ponnurangam Kumaraguru, Lorrie Faith Cranor, and Julie Downs. Who falls for phish?: A demographic analysis of phishing susceptibility and effectiveness of interventions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '10*, pages 373–382. ACM, 2010.
- [SKG12] Craig A. Shue, Andrew J. Kalafut, and Minaxi Gupta. Abnormally Malicious Autonomous Systems and Their Internet Connectivity. *IEEE/ACM Transactions on Networking*, 20(1):220–230, 2012.
- [SLM10] Fabio Soldo, Anh Le, and Athina Markopoulou. Predictive blacklisting as an implicit recommendation system. In *Proceedings of IEEE Infocom, INFOCOM '10*, pages 1–9. IEEE, 2010.
- [SM83] Gerard Salton and Michael J. McGill. *Introduction to modern information retrieval*. McGraw-Hill, New York, 1983.
- [ssa08] SSAC Advisory on Fast Flux Hosting and DNS. Technical Report SAC 025, ICANN Security and Stability Advisory Committee, 2008.
- [sto] Storm - Distributed and fault-tolerant realtime computation. <https://storm.apache.org/> (last visited on 2015-04-21).
- [str10] 2010 Identity Fraud Survey Report. Technical report, Javelin Strategy & Research, 2010.
- [TGM⁺11] Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, and Dawn Song. Design and evaluation of a real-time url spam filtering service. In *Proceedings of the IEEE Symposium on Security and Privacy, S&P '11*, pages 447–462. IEEE, 2011.
- [TH09] Hicham Tout and William Hafner. Phishpin: An identity-based anti-phishing approach. In *Proceedings of the International Conference on Computational Science and Engineering*, pages 347–352. IEEE Computer Society, 2009.

-
- [Tur01] Peter D. Turney. Mining the Web for Synonyms: PMI-IR Versus LSA on TOEFL. In *Proceedings of the 12th European Conference on Machine Learning, EMCL '01*, pages 491–502. Springer-Verlag, 2001.
- [TVBP10] Jonathan T. Trostle, Bill Van Besien, and Ashish Pujari. Protecting against dns cache poisoning attacks. In *Proceedings of the 6th IEEE Workshop on Secure Network Protocols, NPSec '10*, pages 25–30. IEEE, 2010.
- [vRDSP14] Roland van Rijswijk-Deij, Anna Sperotto, and Aiko Pras. Dnssec and its potential for ddos attacks: A comprehensive measurement study. In *Proceedings of the 2014 Conference on Internet Measurement Conference, IMC '14*, pages 449–460. ACM, 2014.
- [VSB09] Ricardo Villamarín-Salomón and José Carlos Brustoloni. Bayesian bot detection based on dns traffic similarity. In *Proceedings of the 2009 ACM Symposium on Applied Computing, SAC '09*, pages 2035–2041. ACM, 2009.
- [Wei05] Florian Weimer. Passive DNS Replication. In *Proceedings of the 17th annual FIRST Conference*, pages 1–13, 2005.
- [WFBc04] Duane Wessels, Marina Fomenkov, Nevil Brownlee, and kc claffy. Measurements and laboratory simulations of the upper dns hierarchy. In *Proceedings of the International Conference on Passive and Active Network Measurement, PAM '04*, pages 147–157. Springer Berlin Heidelberg, 2004.
- [WFS⁺12] Cynthia Wagner, Jérôme François, Radu State, Thomas Engel, Gérard Wagener, and Alexandre Dulaunoy. SDBF: Smart DNS brute-forcer. In *Proceedings of the Network Operations and Management Symposium, NOMS '12*, pages 1001–1007. IEEE, 2012.
- [Whi09] Tom White. *Hadoop: The Definitive Guide*. O'Reilly Media, 2009.
- [WHLY06] Yao Wang, Ming-zeng Hu, Bin Li, and Bo-ru Yan. Tracking anomalous behaviors of name servers by mining dns traffic. In *Proceedings of the Conference on Frontiers of High Performance Computing and Networking*, pages 351–357. Springer Berlin Heidelberg, 2006.
- [WMG06] Min Wu, Robert C. Miller, and Simson L. Garfinkel. Do security toolbars actually prevent phishing attacks? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '06*, pages 601–610. ACM, 2006.
- [WML06] Min Wu, Robert C. Miller, and Greg Little. Web wallet: Preventing phishing attacks by revealing user intentions. In *Proceedings of the Second Symposium on Usable Privacy and Security, SOUPS '06*, pages 102–113. ACM, 2006.
- [WRN10] Colin Whittaker, Brian Ryner, and Marria Nazif. Large-scale automatic classification of phishing pages. In *Proceedings of the 17th Annual Network and Distributed System Security Symposium, NDSS '10*. Internet Society, 2010.
- [WSS11] Zheng Wang and Tseng Shian-Shyong. Anomaly detection of domain name system (dns) query traffic at top level domain servers. *Scientific Research and Essays*, 6(18):3858–3872, 2011.

- [WT99] Alma Whitten and J. Doug Tygar. Why johnny can't encrypt: A usability evaluation of pgp 5.0. In *Proceedings of the 8th USENIX Security Symposium*, SEC '99, pages 169–184. USENIX Association, 1999.
- [XH09] Guang Xiang and Jason I. Hong. A hybrid phish detection approach by identity discovery and keywords retrieval. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 571–580. ACM, 2009.
- [XHRC11] Guang Xiang, Jason I. Hong, Carolyn P. Rose, and Lorrie F. Cranor. Cantina+: A feature-rich machine learning framework for detecting phishing web sites. *ACM Transactions on Information and System Security*, 14(2):21:1–21:28, 2011.
- [XXXB09] Yuchi Xuebiao, Wang Xin, Li Xiaodong, and Yan Baoping. Dns measurements at the .cn tld servers. In *Proceedings of International Conference on Fuzzy Systems and Knowledge Discovery*, FSKD '09, pages 540–545. IEEE, 2009.
- [XYA⁺08] Yinglian Xie, Fang Yu, Kannan Achan, Rina Panigrahy, Geoff Hulten, and Ivan Osipkov. Spamming botnets: Signatures and characteristics. In *Proceedings of the ACM SIGCOMM Conference on Data Communication*, SIGCOMM '08, pages 171–182. ACM, 2008.
- [yah] Yahoo Clues. <http://clues.yahoo.com/analysis> (last visited on 2013-10-24).
- [YRRR10] Sandeep Yadav, Ashwath Kumar Krishna Reddy, A.L. Narasimha Reddy, and Supranamaya Ranjan. Detecting algorithmically generated malicious domain names. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, pages 48–61. ACM, 2010.
- [YRRR12] Sandeep Yadav, Ashwath Kumar Krishna Reddy, A.L. Narasimha Reddy, and Supranamaya Ranjan. Detecting algorithmically generated domain-flux attacks with dns traffic analysis. *IEEE/ACM Transactions on Networking*, 20(5):1663–1677, 2012.
- [YS02] Zishuang Ye and Sean Smith. Trusted paths for browsers. In *Proceedings of the 11th USENIX Security Symposium*, SEC '02, pages 1–17. USENIX Association, 2002.
- [YS06] Ka-Ping Yee and Kragen Sitaker. Passpet: Convenient password management and phishing protection. In *Proceedings of the Second Symposium on Usable Privacy and Security*, SOUPS '06, pages 32–43. ACM, 2006.
- [ZBW07] Bojan Zdrnja, Nevil Brownlee, and Duane Wessels. Passive monitoring of DNS anomalies. In *Proceedings of the 4th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, DIMVA '07, pages 129–139. Springer-Verlag, 2007.
- [ZECH07] Yue Zhang, Serge Egelman, Lorrie Cranor, and Jason Hong. Phinding phish: Evaluating anti-phishing tools. In *Proceedings of the 14th Annual Network and Distributed System Security Symposium*, NDSS '07, pages 1–16, 2007.
- [zeu] ZeuS Tracker - ZeuS blocklist. <http://zeustracker.abuse.ch> (last visited on 2015-04-21).

-
- [ZHC07] Yue Zhang, Jason I. Hong, and Lorrie F. Cranor. Cantina: A content-based approach to detecting phishing web sites. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 639–648. ACM, 2007.
- [ZPU08] Jian Zhang, Phillip A. Porras, and Johannes Ullrich. Highly predictive blacklisting. In *Proceedings of the 17th USENIX Security Symposium*, SEC '08, pages 107–122. USENIX Association, 2008.

Résumé

L'hameçonnage, ou *phishing* en anglais, est un type d'escroqueries modernes qui cible les utilisateurs de moyens électroniques de communications tels que les téléphones ou les ordinateurs. Les escrocs qui perpétuent ces activités sont nommés hameçonneurs ou *phishers*. Leur objectif est de persuader leurs victimes d'effectuer certaines actions pour leur propre profit. Les hameçonneurs utilisent leur pouvoir de persuasion pour créer des messages électroniques ou des sites Internet capables de leurrer et de manipuler leurs naïves victimes. Ils utilisent des mots et des phrases soigneusement choisis pour établir une atmosphère de confiance avec leurs victimes afin de les pousser à effectuer certaines actions. Bien que les anciennes méthodes d'escroqueries visait directement le vol d'argent ou l'obtention de services à titre gracieux, la principale cible de l'hameçonnage est le vol de données électroniques confidentielles appartenant à ses victimes, car ce type de données a pris de la valeur récemment.

La croissance d'Internet a facilité l'usage de services qui n'étaient réalisés auparavant que par le biais de contacts physiques directs. Certains services basiques tels que la presse écrite ou télévisée, des enseignements en ligne ou des bibliothèques scientifiques sont maintenant disponibles sur Internet. D'autres services plus personnalisés tels que les services de paiement en ligne, de gestion bancaire ou de vente par correspondance sont également accessibles. Ces services personnalisés sont sensibles car ils sont généralement relatifs à de la gestion d'argent et d'informations confidentielles. Par conséquent, l'obtention d'accès à ces services est précieuse afin de voler l'information et l'argent qui y sont stockés. Par exemple, obtenir suffisamment de renseignements à propos d'une personne peut conduire à une usurpation d'identité. Une identité usurpée peut être utilisée pour agir dans d'autres escroqueries en nom et place de cette personne afin de cacher et protéger l'identité du véritable escroc. Cette même identité usurpée peut être aussi utilisée pour accéder à des services Internet en prétendant être le détenteur d'un compte pour ces services. Cela représente en fait le principal objectif recherché par les hameçonneurs : voler les informations requises pour accéder à des services Internet sensibles.

L'hameçonnage est apparu il y a près de 20 ans et ses premières victimes furent des utilisateurs de fournisseur d'accès Internet dont les hameçonneurs essayaient de voler les informations d'accès aux comptes par le biais d'emails usurpant l'identité d'administrateurs. Les attaques d'hameçonnage ciblent généralement les utilisateurs de services sensibles donnés et relatifs à des sociétés données. Les hameçonneurs attirent les clients de ces sociétés en prétendant en être les représentants afin de demander des informations relatives à l'utilisation de leurs services. Ces informations volées représentent principalement des données d'accès à des sites Internet ou des numéros de carte bancaire. Les attaques d'hameçonnage utilisent beaucoup de vecteurs différents mais dans une proportion prédominante elles s'appuient sur des courriels et des faux sites Internet qui imitent ceux des services légitimes qu'elles visent. Un point commun à de nombreuses attaques est en fait l'utilisation de lien Internet dirigeant les victimes vers des contenus malveillants. L'utilisation d'URLs offusquées et contenant des noms de domaine malveillants pour représenter ces liens Internet est très répandue et a fortement augmenté récemment.

De même, au fil des années, les activités d'hameçonnage ont considérablement augmenté en termes d'attaques perpétrées et de nombre de sociétés ciblées. Cette augmentation des attaques d'hameçonnage est soulignée par un préjudice financier grandissant et qui a atteint 5,9 milliards de dollars en 2013 [rsa14]. Il existe quatre principales raisons à cette augmentation :

1. L'augmentation de la surface d'attaque : au fil des années, le nombre d'utilisateurs d'Internet étant de potentielles victimes, a augmenté pour atteindre environ trois milliards aujourd'hui, comparativement à seulement un demi-milliard en 2001. Le même constat est fait pour le nombre d'appareils connectés à Internet qui est estimé à des dizaines de milliards, et ce nombre devrait doubler d'ici 2020. Cela augmente le nombre de vecteurs physiques d'hameçonnage qui ne sont plus limités aux ordinateurs de bureau ou aux téléphones, mais incluent également des ordinateurs portables, des smartphones ou des tablettes. Enfin de plus en plus de services Internet sont disponibles et peuvent être ciblés par les attaques d'hameçonnage comme le souligne le nombre de près d'un milliard de sites Internet en ligne de nos jours. Ainsi, de nombreuses nouvelles victimes potentielles, vecteurs physiques et services sont disponibles pour les hameçonneurs et laissent la possibilité pour de nouvelles attaques d'être perpétrées.
2. La variété des attaques : les appels téléphoniques, les SMS, les courriels ou les sites Internet sont des exemples de technologies de communication utilisées pour perpétrer l'hameçonnage. Protéger les utilisateurs contre cette variété d'attaques est difficile et les techniques de prévention et de détection existantes ne concernent que certaines d'entre elles. De nombreuses techniques de détection de courriels d'hameçonnage ou de sites Internet d'hameçonnage existent, mais leur champ d'application est limité à quelques attaques bien que des dizaines d'autres existent. Par conséquent, dans le contexte actuel, une protection globale contre l'hameçonnage implique l'utilisation de plusieurs techniques indépendantes. De nos jours, le filtrage du courrier électronique, les avertissements des navigateurs Internet et les techniques d'authentification de sites Internet sont utilisés conjointement pour protéger contre l'hameçonnage. Cependant, certaines attaques parviennent encore à contourner cette protection incrémentale de façon telle, que les conséquences de l'hameçonnage sont grandissantes.
3. Le nombre croissant d'hameçonneurs et d'attaques perpétrées : le premier phénomène s'explique par le fait que l'hameçonnage est une activité facile à perpétrer et nécessitant peu de compétences techniques. Le principal effort, pour construire une attaque, est investi dans les astuces d'ingénierie sociale utilisées. De plus, une attaque d'hameçonnage peut être facilement perpétrée par des escrocs techniquement non qualifiés grâce à la disponibilité à la fois de kits d'hameçonnage prêts à l'emploi et d'infrastructures à faible coût pour déployer les attaques. L'augmentation du nombre d'attaques perpétrées s'explique par la diminution du revenu par attaque, forçant les hameçonneurs à lancer plus de campagnes afin de maintenir un revenu constant de leurs activités délictueuses. L'hameçonnage est un délit perpétré par beaucoup d'individus et pour de bas revenus. Par conséquent, les contre-mesures ciblées contre les hameçonneurs sont inefficaces, vu leur nombre, pour réduire cette activité.
4. Le manque de prise de conscience des utilisateurs : cette raison est la principale et concerne le manque de prise de conscience face au risque associé aux communications électroniques et la valeur des informations stockées sur les sites Internet. La plupart des personnes ne sont pas préoccupées par l'impact du vol de leur mot de passe, de leur numéro de carte bancaire ou l'usurpation de leur identité. Ce manque de préoccupations ne les motive pas à protéger leurs données du vol. La sécurité est un objectif secondaire pour la plupart des utilisateurs et leur connaissance technique limitée ne leur permet pas d'améliorer le niveau de sécurité de leurs communications électroniques. Les nouveaux crédules utilisateurs de moyens modernes de communications électroniques sont des cibles faciles pour les hameçon-

neurs qui peuvent facilement les piéger. Cette ignorance généralisée est la principale raison de l'efficacité des attaques d'hameçonnage.

L'hameçonnage est une activité qui ne cesse de croître. De nombreux facteurs expliquent son expansion et l'augmentation de son préjudice financier pour atteindre plusieurs milliards de dollars chaque année [gar07, str10, rsa14]. La variété des attaques d'hameçonnage, l'augmentation des victimes potentielles et des vecteurs physiques, la facilité de commettre cette escroquerie moderne et l'ignorance généralisée des victimes en font une activité préoccupante. Outre son impact financier, l'hameçonnage fait naître des préoccupations concernant l'utilisation des moyens de communications électroniques. Les utilisateurs de ces technologies voient le vol d'informations personnelles et leur utilisation abusive comme un événement très susceptible de se produire. Cette perception de l'hameçonnage comme une fatalité et non comme un problème qui peut être résolu conduit à éroder la confiance présente entre les utilisateurs de moyens de communications électroniques. Un risque direct de cette perte de confiance est l'abandon de technologies tels que les courriels comme moyen de communication. Cela rend la lutte contre l'hameçonnage primordiale pour préserver l'utilisation généralisée de cette technologie.

Pour faire face à cette épineux problème, nous proposons dans ce document de nouvelles techniques pour combattre l'hameçonnage. Ces techniques s'appuient sur nos observations des attaques d'hameçonnage dans l'optique de développer une solution permettant de lutter efficacement contre. Les techniques proposées s'appuient sur l'analyse des noms de domaines et URLs utilisés pour perpétrer des attaques d'hameçonnage et prend en compte le fait que ces noms de domaines sont créés par des humains pour piéger d'autres humains. Nous analysons à la fois la composition sémantique des noms de domaine et les caractéristiques techniques relatives aux noms de domaine pour proposer différentes solutions d'identification rapide d'attaques d'hameçonnage.

Dans une première partie de ce document nous décrivons ce qu'est l'hameçonnage, les techniques utilisées pour le perpétrer et les moyens développés pour le combattre. Nous identifions également les exigences auxquelles doit répondre une technique de détection d'hameçonnage pour être efficace. Dans un second temps, nous présentons le principe de l'analyse et de la surveillance des noms de domaine et du système qui gère ces domaines : le *Système de Noms de Domaine* (DNS). Ensuite, nous passons à la présentation des contributions avec une première méthode de groupement de noms de domaine en fonction de leurs activités. Une méthode automatique d'identification de noms de domaine impliqués dans des activités d'hameçonnage est présentée. Cette méthode repose sur l'inférence de similarité sémantique entre les noms de domaine. Ensuite, nous présentons une méthode d'identification et d'évaluation d'URLs d'hameçonnage. Finalement, les deux dernières contributions de ce document se focalisent sur la prédiction des noms de domaines qui seront utilisés pour des activités d'hameçonnage futures et présentent des générateurs de noms de domaines reposant sur des modèles sémantiques.

L'Hameçonnage et ses Techniques de Protection

L'hameçonnage est une menace persistante qui s'appuie sur de nombreux moyens pour être perpétrée, en utilisant des subterfuges techniques et de l'ingénierie sociale. L'hameçonnage n'est pas un problème de sécurité qui vise une brèche dans l'implémentation d'un système ou d'un réseau, mais il se focalise sur les faiblesses de leurs utilisateurs, ce qui en fait un problème difficile à résoudre. Malgré 20 ans d'existence et les forces engagées pour sensibiliser les utilisateurs de moyens électronique de communication, l'hameçonnage est un problème qui provoque un préjudice financier grandissant au fil du temps. Beaucoup d'attaques différentes sont utilisées

pour perpétrer l'hameçonnage. Elles incluent les courriels qui prétendent être envoyés par des entités légitimes, les faux sites Internet qui imitent des sites légitimes, les sites Internet vendant des produits à bas coûts ou encore des faux programmes d'antivirus. Cette diversité rend son combat difficile car ces différents vecteurs requièrent de multiples techniques pour être détectés. De plus, les attaques d'hameçonnage sont très rapides et durent généralement moins de 12 heures mais parviennent néanmoins à causer de lourdes pertes financières dans ce court laps de temps.

A la vue des caractéristiques des attaques d'hameçonnage, nous définissons quatre exigences primordiales que doivent présenter une technique de protection contre l'hameçonnage afin d'être efficace:

- **Vitesse** : vu que les attaques d'hameçonnage font de gros dégâts monétaire en peu de temps et surtout pendant les premières heures d'une attaque, l'identification d'une attaque doit être rapide pour limiter ses effets néfastes. De plus, si elle doit être utilisée dans un contexte tel que la navigation Internet, une méthode de protection ne doit pas impacter cette activité en introduisant de la latence.
- **Universalité** : une méthode de protection doit être en mesure de protéger contre un maximum de type d'attaques.
- **Fiabilité** : une méthode de protection doit être capable de détecter un maximum d'attaques d'hameçonnage. Toutefois, celle-ci doit également être fiable et ne pas identifier comme de l'hameçonnage une communication légitime.
- **Facilité d'utilisation** : une méthode de protection doit être facile à utiliser et à comprendre par les utilisateurs. La plupart des utilisateurs et en particulier les victimes d'hameçonnage n'ont que des connaissances techniques limitées et peu de connaissance de la façon dont les attaques d'hameçonnage sont effectuées. Ainsi, cette méthode doit tenir compte de ce paramètre et être adaptée pour être facilement utilisable.

De nombreuses techniques ont été développées pour combattre l'hameçonnage au cours des dix dernières années. Les premières propositions consistaient dans le renforcement des techniques d'authentification entre les internautes et les sites Internet qu'ils consultaient. Cependant ces techniques ont été inefficaces car trop compliquées à utiliser par les utilisateurs inexpérimentés. Pour la même raison, les barres de sécurité qui s'ajoutent sur les navigateurs Internet n'ont pas été adoptées car elles ajoutaient des contraintes aux utilisateurs pour vérifier la légitimité d'un site Internet. Ces barres de sécurité proposent généralement des informations à propos d'un site Internet visité et que l'utilisateur doit interpréter afin de confirmer sans authenticité. Une faiblesse des barres de sécurité est leur faible efficacité dans l'identification des sites d'hameçonnage, les rendant très peu fiables. Une dernière méthode de protection qui fut développée contre l'hameçonnage fut les listes noires de noms de domaines et d'URLs. Ces listes sont constituées grâce aux rapports et à la vérification par des utilisateurs de sites Internet suspects. Ces listes noires peuvent être alors intégrées à des navigateurs Internet et à des clients de messagerie électronique pour filtrer les sites et courriels malveillants. Leur processus, reposant sur une vérification humaine, rend les listes noires très fiables. Elles sont cependant lentement mises à jour et n'incluent pas beaucoup de sites d'hameçonnage nouvellement apparus car non reportés par les utilisateurs.

Pour faire face à ce problème de rapidité, des techniques d'identification automatisées d'attaques d'hameçonnage ont été développées. Elles n'ont pas ce problème de retard de mise à jour car elles identifient les attaques à la volée. Les principales solutions proposées sont appliquées

à l'identification des courriels d'hameçonnage, les sites Internet d'hameçonnage et les URLs impliquées dans des attaques d'hameçonnage. L'identification des courriels d'hameçonnage et des sites Internet d'hameçonnage est très spécifique puisqu'elle se focalise sur des types restreints d'attaques, limitant leur universalité d'application. Dans une autre perspective, le temps pour identifier les attaques reste obscur et beaucoup de travaux réalisés ne publient pas leur performance en terme de rapidité ou présentent de fortes latences supérieures à cinq secondes et qui se montreraient problématique pour une usage en temps réel. Par exemple, les techniques pour l'identification des sites Internet d'hameçonnage s'appuient sur l'extraction de nombreuses caractéristiques des pages Internet, ce qui prend du temps.

Une dernière méthode existante pour identifier les attaques d'hameçonnage consiste en l'analyse des URLs. Cette technique utilise uniquement le contenu d'une URL et en réalise une analyse lexicale, ce qui a l'avantage d'être rapide à réaliser. De plus, les URLs sont utilisés dans beaucoup d'attaques d'hameçonnage faisant de cette méthode de protection celle ayant la portée la plus large. Une limitation des méthodes existantes basées sur l'analyse lexicale est qu'elles ont une précision moindre que les techniques utilisant plus de paramètres comme le contenu d'une page Internet ou des informations relatives au serveur hébergeant un site Internet. En outre, les modèles d'identification appris par ces techniques reposent sur des heuristiques statiques liées à l'utilisation de certains mots dans une URL ou à trouver un nombre spécifique de caractères. Cela limite la capacité d'adaptation de ces approches dans certains contextes, comme par exemple un changement de tendance dans la composition des URLs d'hameçonnage ou l'attaque de nouveaux services par les hameçonneurs.

L'analyse des techniques de protection contre l'hameçonnage montre qu'il y a un besoin de développer de nouvelles méthodes car aucune d'entre-elles ne remplit les quatre critères présentés afin d'obtenir une technique de protection efficace. De nouvelles solutions doivent être proposées et ces solutions de détection doivent être rapides, avoir un large champ d'application, être facile à comprendre et à utiliser pour les utilisateurs et être fiable. Utiliser des méthodes d'analyse lexicales comme base pour de nouvelles améliorations, en particulier dans la fiabilité, semble être l'approche la plus prometteuse car ces solutions sont déjà rapides et ont un large champs d'application, réunissant deux critères primordiaux.

Surveillance du Système de Noms de Domaine

Le *Système de Noms de Domaine* (DNS) est un élément clé du fonctionnement d'Internet. Il assure la découverte des ressources sur Internet en mémorisant simplement un nom de domaine facile à retenir. Sa fonction principale est de traduire des noms de domaine en adresses IP. En fait il permet d'associer n'importe quel type d'information à un nom de domaine. L'accès aux ressources Internet avec un mécanisme de traduction d'adresses IP fournit un grand avantage pour les services Internet en renforçant leur disponibilité et permettant d'effectuer de l'équilibrage de charge entre plusieurs serveurs. Ceci permet également une récupération rapide en cas de panne d'un serveur en changeant simplement dans un enregistrement DNS l'adresse IP du serveur défectueux par celle d'un serveur en état de fonctionnement. Le DNS est utilisé pour améliorer la disponibilité et la diffusion des contenus sur Internet à travers les réseaux de diffusion de contenu (*CDNs* ou *Content Delivery Networks*). Il a également d'autres applications comme la diffusion de listes noires par exemple et est donc utilisé pour de nombreuses applications qui vont au delà de sa conception et de son rôle initial.

Malheureusement, ce service est également utilisé pour renforcer les activités malveillantes en fournissant un moyen de cacher la véritable infrastructure des réseaux malveillants tels que les

réseaux de zombies ou *botnets* ou les serveurs hébergeant des sites d'hameçonnage par exemple. Ce processus est réalisé grâce à l'utilisation de réseaux de flux ou *flux networks*. Le *fluxing* consiste à faire fréquemment changer les adresses IP associées avec un même nom de domaine. Ainsi, un même contenu sera toujours représenté par le même nom de domaine mais sa localisation physique changera constamment. Ce processus est utilisé pour durcir le processus de lutte contre la diffusion de contenus malveillants sur Internet comme les sites d'hameçonnage. Cependant, effectuer cet usage du système de nom de domaine présente certaines caractéristiques qui peuvent être identifiées en analysant le trafic DNS.

Le suivi et l'analyse du trafic DNS a été utilisé à plusieurs fins, allant de l'évaluation des performances du réseau à la détection des menaces. Plusieurs emplacements de surveillance peuvent être choisis en fonction des fins souhaitées, la mise à l'échelle requise et le respect de la vie privée des utilisateurs. Certains chercheurs ont proposé des solutions pour recueillir et analyser le trafic DNS afin d'identifier les réseaux de flux malveillants étant le support de réseaux de zombies et des activités d'hameçonnage. Des caractéristiques sont extraites des paquets DNS afin de représenter la nature des noms de domaines observés en utilisant des techniques de classification. Ces solutions proposées s'appuient sur l'accumulation de grandes quantités de trafic DNS. Cela implique une latence élevée dans l'identification des noms de domaine malveillants, qui n'est pas problématique pour la détection des réseaux de zombies car ces derniers opèrent sur de longues périodes de temps allant de quelques mois à plusieurs années. Cependant, l'application à la détection d'hameçonnage est limitée en raison de la courte durée de vie des attaques d'hameçonnage. Certaines techniques ont une portée limitée puisqu'elles ne peuvent être exploitées que par certains acteurs d'Internet car elles s'appuient sur la surveillance du trafic DNS à des niveaux supérieurs de la hiérarchie DNS. Quelques travaux portant sur l'analyse lexicale des noms de domaine montrent de bonnes perspectives dans l'identification des noms de domaines malveillants. Les solutions d'analyse lexicale proposées sont rapides puisqu'elles ne nécessitent pas de grande quantité de données DNS. Toutefois, les applications de ces techniques sont limitées à la détection de tunnels DNS ou à la détection de noms de domaine générés algorithmiquement. Outre le fait que certains services légitimes comme les CDNs utilisent des algorithmes pour générer leurs noms de domaine, les noms de domaine d'hameçonnage ne le sont généralement pas, limitant donc l'application de ces techniques au problème de l'hameçonnage.

Par conséquent, l'étude lexicale de noms de domaine couplé avec des données DNS peut se montrer prometteuse pour l'identification des noms de domaine d'hameçonnage. Cependant, des techniques plus élaborées doivent être développées afin de différencier les noms de domaine d'hameçonnage des noms de domaine légitimes. Une solution pourrait résider dans l'étude des mots qui composent les noms de domaines et de leurs significations afin d'observer si certains mots ou champs sémantiques sont utilisés différemment dans les noms de domaine légitimes et dans ceux d'hameçonnage.

Surveillance à Grande Echelle du Système de Noms de Domaine pour Identifier les Noms de Domaine Malveillants

Etant un service de base d'Internet, le système de noms de domaine recèle une quantité d'informations extrêmement riche pour la surveillance de la sécurité d'un réseau. Une activité d'intérêt majeur liée aux abus d'utilisation du système de noms de domaine est la diffusion de contenus malveillants comme des sites Internet d'hameçonnage. L'hébergement de ces contenus s'appuie principalement sur une activité décrite précédemment à savoir le changement fréquent des adresses IP associés à un même nom de domaine ou *fluxing*. Ces noms de domaine sont

représentés par des caractéristiques spécifiques au niveau du DNS. Certaines de ces caractéristiques sont le nombre élevé d'adresses IP associées à un nom de domaine unique ou le faible temps de vie des enregistrements DNS associés à ces noms de domaines. Ces caractéristiques ne peuvent cependant être calculées qu'à partir de données DNS agrégées pour un nom de domaine unique. Par conséquent, une surveillance continue du trafic DNS est nécessaire pour inférer ce genre d'activité et une inspection au cas par cas des paquets DNS n'est pas suffisante. La surveillance sur une longue période de temps du trafic DNS peut générer de grande quantité de données qu'il convient de stocker dans un système approprié. De même, l'exploitation et l'analyse des données stockées nécessite également des techniques appropriées.

Nous proposons donc une architecture pour surveiller le trafic DNS et le capturer. Cette architecture présente les moyens de capturer le trafic DNS passivement et de le stocker de manière distribuée pour des raisons de mise à l'échelle. La méthode de capture proposée a les avantages de préserver l'anonymat des utilisateurs émettant des requêtes DNS et d'éviter la redondance des paquets DNS capturés, favorisant ainsi un gain d'espace de stockage. Le système de stockage s'appuie sur des bases de données distribuées et chaque nom de domaine observé est stocké avec neuf caractéristiques le représentant et permettant d'inférer le type d'activité auquel il est lié, et notamment une activité légitime ou malveillante. Ces caractéristiques sont entre autres:

- le nombre d'adresses IP qui lui sont associées,
- la dispersion de ces adresses sur différents réseaux,
- la période de validité moyenne de ses enregistrements,
- le nombre de requêtes observées pour ce nom de domaine,
- la période de temps sur laquelle il a été observé,
- le nombre de sous-domaines qu'il détient,
- le nombre de serveurs étant autoritaires pour délivrer de l'information à propos de ce nom de domaine.

Ces caractéristiques sont choisies car elles permettent d'inférer la dynamique, la longévité, l'importance et la stabilité d'un nom de domaine. Ces critères sont déterminants pour distinguer les noms de domaine effectuant du fluxing de ceux ayant une activité légitime. Les données capturées sont exploitées par un système distribué de traitement de données étant Hadoop. Nous appliquons une méthode de regroupement ou *clustering* sur ces données afin de grouper ensemble les noms de domaine ayant une activité similaire grâce à l'algorithme des K-means.

Cette méthode de capture fut appliquée dans deux réseaux de différente nature permettant ainsi de recueillir deux ensembles de données différents de traces DNS. En appliquant l'algorithme de regroupement des K-means sur ces jeux de données, nous avons été capable de former sept groupes de noms de domaine présentant différentes caractéristiques. Après analyse manuelle de certains éléments de ces groupes, nous avons constaté que certains groupes contenaient uniquement des noms de domaine étant très populaires et un autre uniquement des noms de domaine avec une faible popularité. Les noms de domaine utilisés dans la diffusion de contenu (CDN) étaient rassemblés dans un même groupe, de même que les noms de domaines suivant les comportements des utilisateurs et finalement les noms de domaines impliqués dans les réseaux de flux étaient également rassemblés ensemble. Le résultat de cette expérience tend à prouver que la capture et l'analyse de données DNS est adaptée à l'identification des noms de domaines impliqués dans des activités malveillantes.

Plusieurs travaux concernant l'analyse passive de données DNS ont été précédemment effectués. Cependant, nous sommes les premiers à proposer l'utilisation d'un système de traitement de données distribué pour des raisons de mise à l'échelle qui sont dues à la grande quantité de données à traiter. En outre, les travaux existants se focalisent sur l'identification des noms de domaine malveillants en utilisant généralement des méthodes de classification alors que nous utilisons une méthode de regroupement qui ne nécessitent aucune connaissance préalable sur la nature des noms de domaine à traiter. Nous proposons également de nouvelles caractéristiques permettant d'identifier les noms de domaine malveillant à partir de données DNS comme la dispersion des adresses IP associées à un nom de domaine.

Cette technique, tout en étant efficace à première vue, a quelques inconvénients. L'emplacement des sondes de capture est un facteur important qui influe sur les valeurs des caractéristiques que l'on extrait des paquets DNS pour un même nom de domaine. Par conséquent, pour obtenir des caractéristiques représentatives d'un nom de domaine donné, plusieurs sondes DNS doivent être déployées à travers Internet. Sans ce déploiement à grande échelle, les résultats du regroupement peuvent être faussés par un intérêt local de certains utilisateurs pour des noms de domaine donnés. Une autre exigence de ce genre de méthode est l'accumulation de plusieurs paquets DNS concernant un même nom de domaine pour obtenir des caractéristiques représentatives. Cela introduit une latence dans l'identification des noms de domaine malveillants. Enfin, l'ensemble des données à analyser doit être équilibré. Si le nombre de noms de domaine malveillants est trop faible, il y a de fortes chances que ces derniers ne soient pas suffisamment représentatifs pour former un groupe indépendant de noms de domaine malveillants et qu'ils soient mélangés avec des noms de domaine légitimes.

Identification des Noms de Domaine d'Hameçonnage en Utilisant leur Parenté Sémantique

Pour compléter et faire face à certaines lacunes de la méthode passive d'analyse de données DNS présentée précédemment, nous introduisons une nouvelle technique pour identifier les noms de domaine d'hameçonnage. Alors que la première proposition repose sur des caractéristiques extraites des paquets DNS, cette nouvelle méthode s'appuie sur l'analyse sémantique des noms de domaine et permet d'identifier automatiquement des groupes de noms de domaine inconnu comme étant malveillants ou non. Cette méthode repose sur le fait que les hameçonneurs enregistrent des noms de domaine utilisant un vocabulaire spécifique pour piéger leur victime grâce à des liens offusqués. Nous proposons donc d'identifier ce vocabulaire utilisé dans les nom de domaine malveillants et de prouver qu'il est différent de celui utilisé dans les noms de domaine légitimes.

Pour ce faire, nous extrayons les mots composant les noms de domaine et créons une liste de couple contenant les mots extraits et leur fréquence d'apparition composant ainsi une distribution probabiliste des mots composants un nom de domaine. Comme certains nom de domaine peuvent être courts nous extrayons ces distributions probabilistes d'un groupe de noms de domaine préalablement formés car ayant des similitudes. La méthode de regroupement précédemment proposée et reposant sur de l'analyse de données DNS peut être utilisée à cette fin. Une fois une distribution probabiliste des mots composant des noms de domaine obtenue, nous proposons de quantifier les similarités sémantiques entre deux distributions. Nous cherchons à démontrer par ce biais que les mots composant des noms de domaine malveillants et les mots composants des noms de domaine légitimes appartiennent à des champs sémantiques différents et qu'identifier cette différence peut conduire à leur distinction. Nous proposons donc trois métriques basées sur des méthodes de l'état de l'art et permettant de quantifier la similarité sémantique entre deux

distributions probabilistes de mots. Par transition, vu que ces distributions sont extraites de groupes de noms de domaine, ces métriques permettent de quantifier la similarité sémantique entre deux groupes de noms de domaine.

Afin de tester la validité de cette approche, nous calculons les trois métriques introduites entre différents ensembles de noms de domaine malveillants et légitimes. Cinq ensembles de noms de domaine malveillants et cinq ensemble de noms de domaine légitimes sont comparés deux à deux en utilisant les trois métriques. La conclusion est la même pour l'évaluation des différentes métriques :

- les groupes de noms de domaine légitimes montrent une grande similarité entre eux,
- les groupes de noms de domaine malveillants montrent une grande similarité entre eux,
- les groupes de noms de domaine légitimes lorsqu'ils sont comparés aux groupes de noms de domaine malveillants montrent une faible similarité étant de l'ordre de 20% à 30% inférieure aux deux précédents cas de figure.

Ces résultats montrent effectivement que les mots composant les noms de domaine malveillants sont différents des mots composant les noms de domaine légitimes. Cette différence peut être exploitée en utilisant les métriques proposées pour identifier des groupes de noms de domaine comme malveillants ou légitimes. En comparant successivement un groupe de noms de domaine inconnu à un groupe de référence de nom de domaine malveillants et à un groupe de référence de noms de domaine légitimes, et en prenant en compte la plus grande similarité obtenue, nous pouvons en conclure la nature du groupe inconnu. Nous montrons empiriquement qu'un groupe de noms de domaine doit au moins comporter dix éléments pour être analysé et pour obtenir un résultat concluant.

De par sa conception cette méthode ne nécessite pas de déploiement à grande échelle de sondes DNS contrairement à la précédente méthode. Cependant, il faut une phase d'apprentissage sur des données labélisées afin d'avoir des groupes de référence pour effectuer une comparaison. Ceci est un inconvénient car les techniques d'hameçonnage évoluent très vite et les modèles sémantiques des noms de domaine malveillants peuvent évoluer également. En outre, l'approche sémantique présentée ne peut s'appliquer qu'à des groupes de noms de domaine et non à des noms de domaine individuels. Par conséquent, un regroupement préalable des noms de domaine doit être réalisé, ce qui introduit un délai, tout du moins au moment de l'initialisation de cette méthode afin de former des groupes initiaux de noms de domaine. Cependant, après cette étape, de nouveaux noms de domaine capturés peuvent être inclus dans les groupes déjà formés pouvant ainsi proposer une détection en temps réel des noms de domaine malveillants.

Une technique pour identifier individuellement les noms de domaine ou les URLs d'hameçonnage sans phase d'apprentissage aurait les avantages des deux précédentes méthodes sans en avoir leurs faiblesses. Une telle méthode pourrait identifier en temps réel les noms de domaine ou les URLs d'hameçonnage sans délai engendré par une phase d'initialisation.

Evaluation des URLs d'Hameçonnage en Utilisant des Relations Sémantiques

Pour répondre aux limitations des méthodes précédemment présentées et parvenir à identifier individuellement des attaques d'hameçonnage sans délai, nous proposons une méthode pour identifier en temps réel les URLs d'hameçonnage. Pour étendre l'analyse sémantique proposée précédemment et obtenir plus de mots ayant du sens à analyser, nous proposons de focaliser cette

analyse sur des URLs complètes et non plus seulement des noms de domaine. Cette méthode repose sur une observation que nous avons faite à propos des URLs d’hameçonnage. Lors de leur composition, la plus grande partie de ces URLs peut être définie de façon libre exceptée une partie qui est le nom de domaine car ce dernier doit être enregistré auprès d’une entité d’enregistrement qui s’assure qu’un nom de domaine n’est enregistré que par une seule personne et que seule cette personne a donc le contrôle des ressources qui sont associées à ce nom de domaine. Par conséquent, un hameçonneur doit utiliser d’autres noms de domaine que celui de la société ou du site Internet qu’il attaque car celui-ci est déjà enregistré. Cependant, pour leurrer ses victimes il va faire en sorte que le reste de l’URL qu’il compose, ressemble le plus possible à l’originale du site Internet qu’il attaque. Nous déduisons de cette observation que les différents mots composant les URLs légitimes sont tous apparentés alors que les noms de domaine des URLs d’hameçonnage vont présenter des différences. Nous nommons ce concept de parenté entre les mots composant une URL la *parenté intra URL*.

Quantifier cette parenté entre les éléments composant une URL est similaire à l’inférence de similarité sémantique entre des mots ou des groupes de mots que nous avons précédemment présentée. Cependant, les composants des URLs sont spécifiques et beaucoup d’entre eux n’apparaissent pas dans des dictionnaires empêchant l’utilisation de méthodes existantes pour inférer les liens de parenté. C’est pourquoi nous utilisons les données de requêtes faites aux moteurs de recherches pour évaluer les liens de parenté entre des mots composant les URLs. Les requêtes faites aux moteurs de recherches contiennent généralement des associations de mots, services et sites Internet correspondant aux attentes des utilisateurs. Ces associations de mots dénotent une parenté entre ces mots dans l’esprit des utilisateur et c’est exactement ce que les hameçonneurs cherchent à identifier et copier lorsqu’ils composent leurs URLs. En utilisant ces données de moteurs de recherches qui sont librement accessibles pour des moteurs de recherche tels que Yahoo et Google, nous faisons des requêtes pour les différents mots inclus dans une URL afin de créer des caractéristiques dénotant la parenté intra URL. Douze caractéristiques sont calculées grâce à cette technique. L’extraction de ces paramètres à partir d’un jeu de données d’URLs légitimes et d’un autre jeu de données d’URLs d’hameçonnage nous permet d’utiliser un algorithme d’apprentissage automatique afin de construire deux modèles représentant les caractéristiques que présentent les URLs d’hameçonnage et les URLs légitime en terme de parenté intra URL. Ces modèles peuvent être ensuite utilisés pour identifier des URLs inconnues comme des URLs d’hameçonnage ou comme des URLs légitimes. Une extension que nous proposons permet également d’utiliser cet algorithme d’apprentissage automatique pour donner une note dénotant le niveau de malveillance d’une URL.

Cette technique a été testée sur un ensemble de 96,018 URLs composé pour moitié d’URLs légitimes et pour l’autre moitié d’URLs d’hameçonnage. Les expériences montrent que le jeu de 12 caractéristiques révèle des différences entre les URLs légitimes et celles d’hameçonnage prouvant que notre concept de parenté intra URL permet de distinguer l’hameçonnage des activités légitimes. L’utilisation de notre algorithme d’apprentissage automatique sur le jeu de caractéristiques permet de classifier correctement 94,91% des URLs du jeu de test avec un taux de faux positifs de 1,44%. Le calcul, sur le jeu de test, de la note représentant le niveau de malveillance d’une URL permet d’identifier correctement 83,97% des URLs du jeu de test avec une précisions supérieur à 99%. La mise en œuvre de ce système en s’appuyant sur des techniques distribuées d’analyse en temps réel prouve être rapide en étant capable d’analyser une URL en moins d’une seconde. Cette technique réunit trois des quatre critères que nous avons définis pour développer une méthode de détection d’hameçonnage efficace : elle est rapide, fiable et couvre un large champs d’attaque car elle s’applique aux URLs qui sont employées dans nombre d’attaques d’hameçonnage.

Les trois techniques précédemment présentées pour identifier les URLs et les noms de domaine d’hameçonnage sont efficaces et les améliorations apportées de façon incrémentale permettent de présenter une solution répondant à trois des quatre critères définis. Des conclusions intéressantes ont été présentées en observant la composition de noms de domaine et URLs d’hameçonnage, à savoir que ceux-ci ont un caractère prévisible en terme de mots et de champs sémantiques utilisés dans leur composition. En conséquence, afin d’améliorer l’efficacité des techniques de protection contre l’hameçonnage, nous proposons d’explorer les moyens de passer d’une détection des attaques à une prévision des attaques en s’appuyant sur le caractère prédictible de la composition des noms de domaine d’hameçonnage.

Sondage du Système de Noms de Domaine Reposant sur les Relations Sémantiques

Pour passer de la détection à la prédiction des noms de domaine utilisés pour perpétrer de l’hameçonnage, nous explorons dans cette partie le caractère prévisible des noms de domaine. Certains travaux ont déjà démontré cette caractéristique en fournissant des méthodes efficaces pour découvrir les différents sous-domaines d’un nom de domaine grâce à des dictionnaires de mots couramment utilisés à cet effet. Ces méthodes de sondage du DNS sont cependant basiques et peu adaptables. Ainsi, pour compléter ces méthodes, nous proposons une nouvelle technique de sondage du DNS afin de découvrir les sous-domaines d’un nom de domaine en s’appuyant sur les similarités sémantiques que présentent les sous-domaines d’un même nom de domaine. En effet, les différents sous-domaines d’un même nom de domaine sont fréquemment donnés par une même personne et pour des raisons de mémorisation ces noms sont souvent apparentés. Nous supposons donc qu’en disposant d’un jeu initial de sous-domaines d’un même nom de domaine, nous pouvons découvrir de nouveau nom de domaine en testant des mots sémantiquement apparentés.

Nous présentons trois modules capables de générer de nouveau mots susceptibles d’être des sous-domaines d’un nom de domaine donné et ce, en utilisant une liste de sous-domaines déjà connus. Ces modules sémantiques analysent la composition des noms de domaine en extrayant les mots les composant. Un premier module cherche les mots apparentés aux sous-domaines existants. Un second module essaie de découper les sous-domaines en plusieurs mots ayant du sens afin de former de nouveaux sous-domaines en combinant des mots apparentés. Le dernier module identifie si les noms de domaine sont composés d’une partie numérique et tente d’incrémenter et décrémenter cette composante afin de découvrir de nouveaux sous-domaines. Les trois modules sont combinés afin de générer des mots susceptibles d’être des sous-domaines.

Pour tester la validité de cette approche nous avons sélectionné 24 noms de domaine populaires que nous avons sondés en utilisant trois méthodes existantes afin de créer un jeu initial de sous-domaines sur lequel notre méthode peut être appliquée. Pour chaque mot généré par notre méthode, une requête DNS est faite afin de vérifier si le potentiel sous-domaine en est effectivement un ou non. Les tests réalisés montrent que sur la base des jeux initiaux de sous-domaines, notre méthode est capable de découvrir entre 84% et 102% de nouveaux sous-domaines, signifiant qu’elle est capable de doubler le nombre de sous-domaines connu en moyenne. Ceci montre que cette méthode est complémentaire avec les solutions existantes. De plus, en fusionnant les jeux initiaux produits par les trois différents outils de sondage de noms de domaine de l’état de l’art, le sondage sémantique améliore encore de 30% le nombre de sous-domaines connus en moyenne.

Cela montre que l’analyse de parenté sémantique dans les noms de domaine est capable d’extraire le modèle de composition de ces noms. En utilisant soigneusement les informations

extraites nous pouvons construire des modèles capables de prédire les noms de domaine qui sont susceptibles d'être utilisés. Le résultat des expériences réalisées montre que les noms de domaine sont prévisibles puisque nous avons pu découvrir les sous-domaines de noms de domaine populaires. Bien que la tâche de prédire des sous-domaines est plus facile que la prédiction des noms de domaine complets, ceci donne des indices sur l'applicabilité des techniques d'inférence de parenté sémantique pour prédire les noms de domaine complets.

Découverte Proactive des Noms de Domaines d'Hameçonnage

Après avoir abordé les solutions pour l'identification des noms de domaine et des URLs d'hameçonnage en temps réel, nous venons d'aborder l'applicabilité de l'analyse sémantique pour prédire les noms de domaine légitimes en usage. Cette technique fut appliquée à du sondage du DNS et a présenté de meilleurs résultats que les techniques proposées dans l'état de l'art. Sur la base de ces premières conclusions, nous proposons d'explorer les manières d'appliquer les modèles de langage naturel et l'analyse sémantique à la prévention de l'hameçonnage. Une caractéristique principale des campagnes d'hameçonnage est leur courte durée de vie, rendant une réaction rapide primordiale afin d'y faire face. Ainsi, plutôt que d'utiliser des techniques réactives ou d'identification en temps réel pour combattre l'hameçonnage, nous présentons une technique prédictive à cette fin. Cette méthode se présente sous la forme d'une liste noire prédictive composée de noms de domaine qui sont susceptibles d'être utilisés pour de l'hameçonnage. Elle présente les avantages des listes noires en étant facilement intégrable dans un client de messagerie électronique ou un navigateur Internet tout en supprimant l'inconvénient de latence dans la mise à jour.

La composition de cette liste noire repose sur l'analyse structurelle et lexicale de la composition des noms de domaine d'hameçonnage existants. Nous présentons des caractéristiques pertinentes pour capturer la composition structurelle des noms de domaine. Ces caractéristiques sont le nombre de mots composant un niveau de nom de domaine, les mots utilisés dans les noms de domaine, les TLDs utilisés et les transitions entre les différents mots. Nous montrons que ces caractéristiques ont des valeurs différentes lorsqu'elles sont extraites de noms de domaine d'hameçonnage ou de noms de domaine légitimes. Les mots qui composent les noms de domaine d'hameçonnage appartiennent à un vocabulaire réduit qui est différent de celui utilisé dans les noms de domaine légitimes. Ces caractéristiques permettent donc de construire un modèle reposant sur le langage naturel pour les noms de domaine d'hameçonnage. Ce modèle repose sur un modèle de chaîne de Markov qui permet de générer de nouveaux noms de domaine suivant les règles de composition qu'il a appris. Il est étendu avec un module sémantique afin d'accroître la variété des noms de domaine générés.

Nous avons étudié l'efficacité de cette technique en utilisant un ensemble de noms de domaine d'hameçonnage afin de construire le modèle de génération. Les expériences ont montré que le modèle appris est capable de générer de nombreux noms de domaine d'hameçonnage qui sont effectivement utilisés dans des activités malveillantes après leur génération. Certains noms de domaine légitimes sont également générés, mais dans de faibles proportions et un score calculé pendant le processus de génération de la chaîne de Markov permet d'identifier la plupart d'entre eux. Ceci fournit une approche intéressante pour faire face aux attaques d'hameçonnage en empêchant la connexion à une ressource malveillante avant qu'elle ne soit effectivement disponible.

La lutte contre les menaces persistantes telles que l'hameçonnage s'appuyait jusqu'alors sur des techniques réactives. Cependant, les techniques qui ont été développées pour combattre ce problème n'ont pas réussi à enrayer cette menace. Les hameçonneurs développent continue-

ment de nouvelles attaques et nouveaux subterfuges pour contourner les techniques de protection mises en œuvre. Ainsi, pour traiter efficacement ce problème, il est nécessaire de réfléchir plus vite que les hameçonneurs et de prévoir à l'avance les nouveaux moyens qu'ils utiliseront pour perpétrer leurs activités malveillantes. Une telle solution est proposée avec cette méthode de composition de liste noire prédictive qui est en mesure de prédire les noms de domaine qui seront utilisés dans de futures attaques d'hameçonnage. Cette technique n'est pas infaillible car elle génère une part non négligeable de noms de domaine légitimes. En outre, la plus grande partie des noms de domaine générés n'est pas encore utilisée et ne le sera probablement jamais. Néanmoins, les techniques de prévention proactives sont susceptibles d'être les méthodes permettant de se débarrasser de la menace persistante qu'est l'hameçonnage.

Pour combattre les activités d'hameçonnage et leurs néfastes conséquences qui ne cessent de croître, cette thèse présente plusieurs défis qu'il convient de relever afin d'inverser cette tendance, en développant des solutions efficaces pour protéger les utilisateurs de moyens de communications électroniques de l'hameçonnage. Il y a d'abord une nécessité de développer des techniques de détection rapides et capables de faire face aux attaques d'hameçonnage ayant une courte durée de vie. Ces techniques doivent pouvoir être intégrées dans un système de détection en temps réel qui ne détériorerait pas la qualité d'utilisation des applications auxquelles elles seraient intégrées comme des navigateurs Internet par exemple. Un deuxième défi consiste à développer des méthodes de protection anti-hameçonnage avec une grande portée afin de faire face à la majorité des attaques d'hameçonnage. Un troisième défi est de développer des techniques de détection d'hameçonnage fiables, car les contenus d'hameçonnage ont tendance à imiter des contenus légitimes rendant leur identification compliquée. Le dernier défi résidait moins dans des aspects techniques, mais plus sur la convivialité d'utilisation des solutions développées afin que les utilisateurs inexpérimentés puissent facilement comprendre et utiliser ces techniques.

Cette thèse ne traite pas tous ces défis afin de fournir une protection infaillible contre l'ensemble des attaques d'hameçonnage. Cependant, elle fournit quelques contributions pertinentes qui améliorent le combat contre l'hameçonnage, en introduisant l'utilisation de l'analyse de parenté sémantique et de la composition des noms de domaine et des URLs pour identifier les attaques d'hameçonnage. Bien que l'analyse de parenté sémantique ait déjà été utilisée pour la détection d'attaque d'hameçonnage dans le passé, elle fut appliquée uniquement à des contenus contenant beaucoup d'informations à savoir des courriels et des pages Internet. Nous avons présenté dans ce document différentes techniques pour extraire le contexte sémantique des noms de domaine et des URL, qui sont des localisateurs contenant peu d'informations. L'analyse lexicale et sémantique des URLs a l'avantage de s'appuyer uniquement sur des informations contenues dans ces entités, ce qui signifie que cette méthode s'applique à toutes attaques d'hameçonnage utilisant des URLs. Ceci permet de couvrir un large éventail d'attaques d'hameçonnage car les URLs sont utilisées dans un grand nombre d'entre elles. En outre, cette technique s'est montrée rapide d'exécution de telle façon qu'elle ne détériorerait pas l'expérience des utilisateurs en étant capable de détecter des attaques en temps réel. Enfin, pour certaines applications comme un système de recommandation d'URLs par exemple, cette technique présente une grande fiabilité laissant envisager un déploiement dans le monde réel.

Une exigence qui n'est pas abordée par les contributions présentées dans cette thèse est la facilité d'utilisation des résultats obtenus par des utilisateurs inexpérimentés. Bien que ces techniques se soient montrées théoriquement efficaces, aucune étude de leur utilisabilité n'a été effectuée avec un groupe d'utilisateurs. En outre, les ensembles de test considérés pour les évaluations étaient de taille limitée ne permettant pas d'évaluer la capacité de mise à l'échelle de nos solutions bien qu'elles soient conçues sur des modèles distribués devant permettre ce passage

à l'échelle. De même, nous n'avons pas effectué de réel déploiement de ces techniques dans le monde réel. Par conséquent, les résultats présentés dans cette thèse soulèvent des nouvelles perspectives pour des travaux de recherche futurs.

Abstract

Phishing is a kind of modern swindles that targets electronic communications users and aims to persuade them to perform actions for a another's benefit. Miscreants performing this activity are named phishers and employ their power of persuasion to tailor socially engineered messages able to deceive their gullible victims. A popular example of phishing activities is the stealing of web services account login information or credit card information using fake websites or spoofed emails. However, several means are used to perform phishing attacks and several goals are sought, which harden the fight against phishing. Despite the forces engaged to get rid of this threat, phishing remains a concerning problem since the financial damage it causes is increasing overtime. Moreover, the perceived fatality about being a victim of phishing erodes the trust among users and threaten the use of electronic means as way of communicating. Existing solutions to cope with phishing attacks are not adapted to their short lifetime and the variety of means used to perform them, making them inefficient. Crowd verified blacklists, emails content analysis techniques or web page content analysis techniques did not succeed to reverse the increasing trend presented by phishing consequences. None of these solutions present the essential requirements that must meet a phishing protection technique to be efficient and which are speed, coverage, reliability and usability.

Stating that phishing attacks rely mostly on social engineering and that most phishing vectors leverage directing links represented by domain names and URLs, we introduce new solutions to cope with phishing. These solutions rely on the lexical and semantic analysis of the composition of domain names and URLs. Both of these resource pointers are created and obfuscated by phishers to trap their victims. Hence, we demonstrate in this document that phishing domain names and URLs present similarities in their lexical and semantic composition that are different form legitimate domain names and URLs composition. We use this characteristic to build models representing the composition of phishing URLs and domain names using machine learning techniques and natural language processing models. The built models are used for several applications such as the identification of phishing domain names and phishing URLs, the rating of phishing URLs and the prediction of domain names used in phishing attacks. All the introduced techniques are assessed on ground truth data and show their efficiency by meeting speed, coverage and reliability requirements. This document shows that the use of lexical and semantic analysis can be applied to domain names and URLs and that this application is relevant to detect phishing attacks.

Keywords: phishing detection, DNS monitoring, semantic analysis, URL lexical analysis, Internet security, machine learning

L'hameçonnage est une escroquerie moderne qui cible les utilisateurs de communications électroniques et vise à les convaincre de réaliser des actions pour le bénéfice d'un individu nommé hameçonneur. Les hameçonneurs emploient leur pouvoir de persuasion pour formuler des messages capables de tromper leurs crédules victimes. Un exemple populaire d'hameçonnage est le vol d'information relative à des comptes de sites internet ou de numéro de carte de crédit en utilisant de faux sites internet ou des courriels falsifiés. Cependant, beaucoup de techniques sont utilisées pour effectuer des attaques d'hameçonnage et beaucoup d'objectifs sont recherchés, rendant difficile la lutte contre l'hameçonnage. Malgré les forces engagées pour se débarrasser de cette menace, l'hameçonnage reste un problème important si l'on considère le préjudice financier grandissant qu'il provoque. Les solutions existantes pour combattre les attaques d'hameçonnage ne sont pas adaptées à leur courte durée d'exécution et à la variété des moyens utilisés pour les réaliser, les rendant inefficaces. Les listes noires, l'analyse du contenu des courriels ou des pages internet sont tant de techniques qui ne sont pas parvenues à inverser la tendance. Aucune de ces solutions ne présente les exigences essentielles auxquelles doivent répondre une technique de protection efficace contre l'hameçonnage et qui sont la vitesse, l'universalité, la fiabilité et la facilité d'utilisation.

Constatant que les attaques d'hameçonnage s'appuient essentiellement sur de l'ingénierie sociale et que la plupart des attaques d'hameçonnage utilisent des liens représentés par des noms de domaine et des URLs, nous proposons de nouvelles solutions pour combattre l'hameçonnage. Ces solutions reposent sur une analyse lexicale et sémantique de la composition des noms de domaine et des URLs. Ces deux pointeurs de ressources sont créés et offusqués par les hameçonneurs pour piéger leurs victimes. Ainsi, nous démontrons dans cette thèse que les noms de domaine et les URLs utilisés dans des attaques d'hameçonnage présentent des similitudes dans leur composition lexicale et sémantique, et que celles-ci sont différentes des caractéristiques présentées par les noms de domaine et les URL légitimes. Nous utilisons ces caractéristiques pour construire des modèles représentant la composition des URLs et des noms de domaine d'hameçonnage en utilisant des techniques d'apprentissage automatique et des méthodes de traitement du langage naturel. Les modèles construits sont utilisés pour des applications telles que l'identification de noms de domaine et des URLs d'hameçonnage, la notation des URLs d'hameçonnage et la prédiction des noms de domaine utilisés dans les attaques d'hameçonnage. Les techniques proposées sont évaluées sur des données réelles et elles montrent leur efficacité en répondant aux exigences de vitesse, d'universalité et de fiabilité. Cette thèse démontre que l'utilisation de l'analyse lexicale et sémantique peut être appliquée aux noms de domaine et aux URLs et que cette utilisation est pertinente pour détecter les attaques d'hameçonnage.

Phishing is a kind of modern swindles that targets electronic communications users and aims to persuade them to perform actions for a another's benefit. Miscreants performing this activity are named phishers and employ their power of persuasion to tailor socially engineered messages able to deceive their gullible victims. A popular example of phishing activities is the stealing of web services account login information or credit card information using fake websites or spoofed emails. However, several means are used to perform phishing attacks and several goals are sought, which harden the fight against phishing. Despite the forces engaged to get rid of this threat, phishing remains a concerning problem since the financial damage it causes is increasing overtime. Moreover, the perceived fatality about being a victim of phishing erodes the trust among users and threaten the use of electronic means as way of communicating. Existing solutions to cope with phishing attacks are not adapted to their short lifetime and the variety of means used to perform them, making them inefficient. Crowd verified blacklists, emails content analysis techniques or web page content analysis techniques did not succeed to reverse the increasing trend presented by phishing consequences. None of these solutions present the essential requirements that must meet a phishing protection technique to be efficient and which are speed, coverage, reliability and usability.

Stating that phishing attacks rely mostly on social engineering and that most phishing vectors leverage directing links represented by domain names and URLs, we introduce new solutions to cope with phishing. These solutions rely on the lexical and semantic analysis of the composition of domain names and URLs. Both of these resource pointers are created and obfuscated by phishers to trap their victims. Hence, we demonstrate in this document that phishing domain names and URLs present similarities in their lexical and semantic composition that are different from legitimate domain names and URLs composition. We use this characteristic to build models representing the composition of phishing URLs and domain names using machine learning techniques and natural language processing models. The built models are used for several applications such as the identification of phishing domain names and phishing URLs, the rating of phishing URLs and the prediction of domain names used in phishing attacks. All the introduced techniques are assessed on ground truth data and show their efficiency by meeting speed, coverage and reliability requirements. This document shows that the use of lexical and semantic analysis can be applied to domain names and URLs and that this application is relevant to detect phishing attacks.