Preventing information leakage in NDN with name and flow filters

Daishi Kondo

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Preventing information leakage in NDN with name and flow filters

(Prévenir la fuite d'information dans les NDN grâce aux filtres de noms et de flux)

THÈSE

présentée et soutenue publiquement le 17 déc, 2018

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Daishi Kondo

Composition du jury

Rapporteurs :	Miki Yamamoto	Professor at Kansai University
	Toru Hasegawa	Professor at Osaka University
Examinateurs :	Giovanna Carofiglio	Distinguished Engineer / Senior Director at Cisco Systems
	Houda Labiod	Professor at Telecom ParisTech
	Isabelle Chrisment	Professor at University of Lorraine
	Sylvain Contassot-Vivier	Professor at University of Lorraine
Invités :	Hideki Tode	Professor at Osaka Prefecture University
	Tohru Asami	CEO at ATR
Encadrants :	Olivier Perrin	Professor at University of Lorraine
	Thomas Silverston	Associate Professor at Shibaura Institute of Technology

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Acknowledgments

Firstly, I would like to express my deep gratitude to my supervisor Olivier Perrin. He has always been supporting and encouraging me during my PhD study. I would also like to thank co-advisor Thomas Silverston for giving me the opportunity to study in France. I will never forget about the experience, and it will inspire my future life. Besides my supervisor and co-advisor, I would like to express my sincere gratitude to Tohru Asami from Advanced Telecommunications Research Institute International and Hideki Tode from Osaka Prefecture University. I really appreciate that we had deep research discussions continually during my PhD study though we had very long distance between France and Japan. Their guidance helped me perform my research and write this thesis, and I cannot imagine that I could have finished my PhD study without them.

My sincere thanks also go to my thesis jury: Miki Yamamoto, Toru Hasegawa, Giovanna Carofigilio, Houda Labiod, Isabelle Chrisment, and Sylvain Contassot-Vivier. Their insightful comments widened my research from the various perspectives.

I keep my special thanks for the members from the RESIST team for a lovely working atmosphere. Especially, I would like to express my appreciation to Isabelle Chrisment and Thibault Cholez. Through my difficult times, their sincere support motivated me to work hard.

It was a great pleasure to meet the members from the DOCTOR project: Bertrand Mathieu, Laurent Morel, Edgardo Montes de Oca, Wissam Mallouli, and Guillaume Doyen. It was the first time for me to carry out such a big project as one of the members. This experience will help me create another big project by myself in the near future.

I am grateful to my supporters for their encouragement to go abroad to study: Osamu Dairiki and Isao Nakaguchi from NS Solutions Corporation, Kazuo Iwano from Mitsubishi Chemical Holdings Corporation, and Hiroshi Takayanagi, Kazuya Makita, Takeshi Shimizu, Satoshi Hirose, Naotaka Hashimoto, Eduardo Gonzalez, and Chihiro Nakamoto from Japan Business Systems, Inc. Without their help and suggestions, I could not have decided to perform my PhD study in France.

I would like to give special thanks to my closest friends: Abdulqawi Saif, Anastasiia Tsukanova, Benjamin Elizalde, Charles Dumenil, Eleonora Carocci, Elian Aubry, Eric Biagioli, Evangelia Tsiontsiou, Giulia De Santis, Guilherme Alves, Haichuan Song, Hatice Calik, Hiroaki Ishikawa, Hoang Long Mai, Ioannis Kokkinis, Iordan Iordanov, Jonás Martínez, Khadija Musayeva, Kohei Yamamoto, Lei Mo, Magdalena Krzaczkowska, Maria Kouroutzi, Marianna Grigoriou, Marjan Bozorg, Maxime Compastié, Meihui Gao, Meiko Fujita, Messaoud Aouadj, Moe Matsuki, Mohamed Eissa, Narumi Oba, Natsuki Inoue, Paolo Pigato, Sény Diatta, Shuguo Zhuo, Shumpei Iwao, Shuo Jin, Takeshi Nagasu, Tan Nguyen, Tatiana Makhalova, Tayeb Oulad Kouider, Tomomi Shibuta, Vassilis Vassiliades, Vitalis Ntombrougidis, Wazen Shbair, Xavier Marchal, and Zia Alborzi.

Finally, I would like to thank my family for their encouragement over many years: Hiroshi, Kaoru, and Ayaka.

To The Memory of My Father
Hiroshi Kondo
May 20th, 1955 – November 5th, 2018

Abstract

In recent years, Named Data Networking (NDN) has emerged as one of the most promising future networking architectures. To be adopted at Internet scale, NDN needs to resolve the inherent issues of the current Internet. Since information leakage from an enterprise is one of the big issues even in the Internet and it is very crucial to assess the risk before replacing the Internet with NDN completely, this thesis investigates whether a new security threat causing the information leakage can happen in NDN. Assuming that (i) a computer is located in the enterprise network that is based on an NDN architecture, (ii) the computer has already been compromised by suspicious media such as a malicious email, and (iii) the company installs a firewall connected to the NDN-based future Internet, this thesis focuses on a situation that the compromised computer (i.e., malware) attempts to send leaked data to the outside attacker.

NDN is basically a "pull"-based architecture and there are only two kinds of packets: Interest and Data, which are a request and a response packet, respectively. In order to retrieve content, a consumer first sends the Interest to NDN network and then obtains the corresponding Data from the producer or the intermediate NDN node. In other words, they cannot send a Data unless they receive the Interest packet. Therefore, as one of the naive methods to mitigate information leakage through a Data, an enterprise network firewall can carefully inspect a Data to publish, and produce it instead of the inside employee in the network (i.e., a whitelist). In this case, all the publicly-accessible content is on the firewall.

However, the firewall cannot manage a naming policy on the outside content and NDN forwarding nodes do not verify whether the name really exists. That causes a risk of information leakage through an Interest by malware's hiding information such as customer information in the Interest name and sending it toward the outside attacker. The malware can pretend to access outside content, so that it is quite difficult for the firewall to detect the information leakage attack. This thesis argues that the information leakage attack through an Interest in NDN should be one of the essential security attacks at protocol level and it is important to develop the detection method of this attack.

The contributions of this thesis are fivefold. Firstly, this thesis proposes an information leakage attack through a Data and through an Interest in NDN. This thesis investigates the one through an Interest deeply, and, as a more advanced attack for the attacker to hide the malicious activity, this thesis proposes a steganography-embedded Interest name to perform information leakage efficiently. To the best of author's knowledge, this is the first research about the information leakage attack in NDN.

Secondly, in order to address the information leakage attack, this thesis proposes an NDN firewall which monitors and processes the NDN traffic coming from the consumers with the whitelist and blacklist. To design the firewall, this thesis focuses on two requirements: (i) designing an NDN firewall independent from NDN Forwarding Daemon (NFD), which decides how to forward an Interest, and (ii) performing a fast lookup of the names or name prefixes in the whitelist and blacklist. By utilizing a cuckoo filter, which is a probabilistic filter, the proposed NDN firewall provides Interest packet filtering based on the names or name prefixes in the lists that can be updated on the fly. While satisfying the requirements and providing the functions, the firewall implementation achieves high performance. Specifically, the throughput degradation with the firewall is only from 0.912% to 2.34%, which will be acceptable in an enterprise network.

Thirdly, this thesis proposes an NDN name filter to classify a name in the Interest as legitimate or not. Since NDN has not been deployed on a large scale, a dataset about NDN traffic does not exist. Assuming that it is highly possible for the future NDN naming policy to become one naturally evolved from the current Uniform Resource Locator (URL) naming policy, this thesis utilizes content names based on URLs collected by a web crawler. By using search engine information and applying the name dataset to an isolation forest, this thesis builds NDN name filters. This thesis evaluates the performances of the name filters and shows that the proposed name filters can drastically choke the information leakage throughput per Interest and malware has to send 137 times more Interest packets to leak information than without using the filters.

The name filter can, indeed, reduce the throughput per Interest, but to ameliorate the speed of this attack, malware can send numerous Interests within a short period of time. Moreover, the malware can even exploit an Interest with an explicit payload in the name (like an HTTP POST message in the Internet), which is out of scope in the proposed name filter and can increase the information leakage throughput by adopting a longer payload. That is the limitation of the name filter. To take traffic flow to the NDN firewall from the consumer into account, fourthly, this thesis proposes an NDN flow monitored at an NDN firewall. This thesis first introduces the concept of NDN flow and specifies it strictly, which has not yet been standardized in NDN research. Then, this thesis proposes a method to generate an NDN flow dataset analogically derived from the HTTP flow dataset in the current Internet because there is no dataset about NDN traffic.

Fifthly, in order to deal with the drawbacks of the NDN name filter, this thesis proposes an NDN flow filter to classify a flow as legitimate or not. By applying the obtained NDN flow dataset to a Support Vector Machine (SVM), this thesis builds an NDN flow filter against the information leakage attack, and the performance evaluation shows that the information leakage throughput choked by the flow filter is from $1.32 \cdot 10^{-2}$ to $6.47 \cdot 10^{-2}$ times that choked only by the name filter. Thus, the flow filter complements the name filter and greatly chokes the information leakage throughput.

Keywords: Named Data Networking, Information leakage, Firewall, Name filter, Flow filter

Résumé

Au cours des dernières années, les réseaux de données nommées (Named-Data Networking – NDN) sont devenus l'une des architectures réseau les plus prometteuses. Cependant, pour être adopté à l'échelle d'Internet, NDN doit résoudre les problèmes inhérents à l'Internet actuel. Étant donné que la fuite dinformations au-delà des frontières d'une entreprise est l'un des gros problèmes, même sur Internet, et quil est important dévaluer le risque encouru avant de remplacer Internet par des réseaux de type NDN, cette thèse examine la robustesse potentielle de NDN face à une menace permettant la fuite dinformations. En supposant (i) qu'un ordinateur situé sur un réseau d'entreprise basé sur une architecture NDN, (ii) que l'ordinateur a déjà été compromis par un vecteur d'attaque suspect tel qu'un courrier électronique malveillant, et (iii) que la société installe un pare-feu connecté au NDN, cette thèse se concentre sur la situation dans laquelle l'ordinateur infecté (par exemple par un logiciel malveillant) tente de divulguer des données à un attaquant externe à l'entreprise.

Une architecture NDN est une architecture basée sur le "pull" et il n'y a que deux types de paquets: les paquets intérêts et les paquets données, qui permettent respectivement d'initier une demande et de recevoir une réponse. Pour pouvoir récupérer un contenu, un consommateur envoie d'abord l'intérêt au réseau NDN, puis obtient les données correspondantes du producteur ou d'un nud intermédiaire du réseau NDN. En d'autres termes, un nud ne peut pas envoyer de données à moins de recevoir un paquet d'intérêt. Par conséquent, une des méthodes naïves pour limiter la fuite dinformations via une donnée consiste à installer un pare-feu sur le réseau dentreprise qui pourra inspecter soigneusement la donnée à publier et qui pourra éventuellement la produire à l'extérieur du périmètre du réseau de l'entreprise (cela consiste donc à établir une liste blanche). Dans ce cas, tout le contenu accessible au public se trouve sur le pare-feu.

Toutefois, le pare-feu ne peut pas gérer de stratégie basée sur les contenu externes et les nuds de transfert NDN ne vérifient pas si un nom existe réellement. Cela entraîne un risque de fuite d'informations via la production d'un intérêt qui dissimulant des informations telles que des données confidentielles du client dans le nom de l'intérêt et en les envoyant à un attaquant externe. Le logiciel malveillant peut prétendre à accéder à du contenu extérieur, de sorte qu'il est assez difficile pour le pare-feu de détecter l'attaque et la fuite d'informations. Cette thèse soutient que ce type d'attaque (fuite d'informations via un intérêt dans NDN) devrait être l'une des attaques principales au niveau du protocole et qu'il est important de développer la méthode de détection de cette attaque, et les contre-mesures associées.

Les contributions de cette thèse sont ainsi au nombre de cinq. Premièrement, cette thèse propose un modèle d'attaque par fuite d'informations via une donnée et via un intérêt dans le réseau NDN. Cette thèse étudie en détail ce mécanisme et, en tant qu'attaque avancée permettant à lattaquant de cacher son activité malveillante, elle propose un mécanisme basé sur la stéganographie pour le nommage dintérêt afin de pouvoir effectuer efficacement une fuite dinformations. À la connaissance de l'auteur, il s'agit de la première recherche sur l'attaque par fuite d'informations dans le NDN.

Deuxièmement, afin de remédier à l'attaque par fuite d'informations, cette thèse propose un pare-feu NDN qui surveille et traite le trafic NDN provenant des consommateurs basé sur une liste blanche et une liste noire. Pour concevoir le pare-feu, cette thèse s'articule autour de deux impératifs: (i) concevoir un pare-feu NDN indépendant du démon de transfert NDN (NDN Forwarding Daemon – NFD), qui décide comment transférer un intérêt, et (ii) effectuer une recherche rapide des noms ou des préfixes de noms dans la liste blanche et la liste noire. En utilisant un filtre probabiliste (Cuckoo filter), le pare-feu NDN proposé fournit un filtrage des paquets dintérêt pouvant être mis à jour à la volée et basé sur les noms ou les préfixes de noms

disponibles. Une mise en uvre effective du pare-feu est disponible et elle atteint des performances élevées. Plus précisément, la dégradation du débit avec le pare-feu nest que de 0.912% à 2.34%, ce qui sera acceptable dans un réseau dentreprise.

Troisièmement, cette thèse propose un filtre de nom NDN pour classifier un nom comme légitime ou anormal dans un paquet Intérêt. Comme peu de réseaux NDN ont été à ce jour déployés à grande échelle, il n'existe pas encore de jeux de données sur le trafic NDN. En supposant que la future politique de nommage NDN puisse être semblable à celle qui a été développée pour l'Internet actuel – Uniform Resource Locator (URL) – ou à une évolution de cette dernière, cette thèse utilise des noms de contenu basés sur les URL collectées par un robot d'exploration Web. En utilisant les informations disponibles et en appliquant à l'ensemble de données de nommage une forêt d'isolation, cette thèse propose des filtres de nom NDN. Une évaluation des performances des filtres de nom est présentée et celle-ci montre que les filtres proposés peuvent réduire considérablement le débit de fuite dinformations par intérêt et que les logiciels malveillants doivent envoyer 137 fois plus de paquets dintérêts pour faire fuiter les informations de fuite que lorsque l'on n'utilise pas de filtres.

De plus, le filtre de noms peut réduire le débit par Intérêt, mais pour améliorer la vitesse de cette attaque, les logiciels malveillants peuvent envoyer de nombreux intérêts en très peu de temps. Le logiciel malveillant peut même exploiter un Intérêt avec un contenu explicite dans le nom (comme peut le faire une requête HTTP POST), ce qui rend inefficace le filtre de nom proposé et ce qui peut augmenter le débit de fuite d'informations. Cette situation représente une limitation du filtre de nom. Pour prendre en compte le trafic vers le pare-feu NDN du consommateur, cette thèse propose un flux NDN surveillé sur un pare-feu NDN. Dans un premier temps, nous présentons le concept de flux NDN et nous le spécifions précisément. Nous proposons ensuite une méthode pour générer un jeu de données de flux NDN dérivé analogiquement du jeu de données de flux HTTP provenant d'Internet car il nexiste aucun jeu de données sur le trafic NDN.

Enfin, afin de traiter les inconvénients du filtre de noms NDN, la thèse propose un filtre de flux NDN permettant de classer un flux comme légitime ou non. Sur la base du jeu de données de flux NDN généré précédemment, nous proposons un filtre de flux NDN contre l'attaque par fuite d'informations. En appliquant sur le jeu de données obtenu une machine à vecteurs de support (SVM), nous avons créé un filtre qui réduit le risque de fuite d'informations, et l'évaluation des performances réalisée montre que le débit de fuite d'informations passant par le filtre de flux est compris entre $1.32 \cdot 10^{-2}$ à $6.47 \cdot 10^{-2}$ fois le débit passant par le filtre de nom uniquement. Nous avons ainsi proposé une architecture efficace qui repose sur un filtre de flux complèté par un filtre de nom, avec comme résultat un risque réduit de fuite dinformations.

Mots-clés: Named Data Networking, Fuite d'informations, Pare-feu, Filtre de nom, Filtre de flux

Contents

List of Figures

List of Tables

Glossary

NDN : Named Data Networking
NFD : NDN Forwarding Daemon
URL : Uniform Resource Locator
SVM : Support Vector Machine
HTTP : Hypertext Transfer Protocol
ICN : Information Centric Networking
HTTPS : HTTP Secure
SSL : Secure Sockets Layer
TLS : Transport Layer Security
IFA : Interest Flooding Attack
CPA : Content Poisoning Attack
DGA : Domain Generation Algorithm
TCP : Transmission Control Protocol
IP : Internet Protocol
URI : Uniform Resource Identifier
FQDN : Fully Qualified Domain Name
mcTLS : multi-context TLS
DPI : Deep Packet Inspection
C&C : Command-and-Control
FIB : Forwarding Information Base
PIT : Pending Interest Table
CS : Content Store
FTP : File Transfer Protocol
RBNs : Routable Backward Names
BN : Bayesian Network
NPGA : Name Prefix Generation Algorithm
CCN : Content Centric Networking
JSON : JavaScript Object Notation
GPL : General Public License
PoC : Proof of Concept
VNF : Virtual Network Function
NFV : Network Function Virtualization
TLDs : Top-Level Domains
gTLDs : generic Top-Level Domains
ccTLDs : country code Top-Level Domains
CDF : Cumulative Distribution Function
SEO : Search Engine Optimization
NLP : Natural Language Processing

RNNs : Recurrent Neural Networks
APT : Advanced Persistent Threat

Chapter 1

General Introduction

Contents

1.1 Background: From the Internet to NDN

The Internet has been widely used to provide a lot of applications for a long time, and the attractive applications are gradually undergoing change. In particular, the web is one of the killer applications, which is a host-to-host oriented application. In recent years, video service is becoming more popular, and according to "The Zettabyte Era: Trends and Analysis" [1], by 2021, 82 % of the Internet traffic volume will be video traffic. The video is located in the server and can be retrieved by the Hypertext Transfer Protocol (HTTP) [2] request message including the Uniform Resource Locator (URL) [3] to designate what content to be downloaded. This kind of the application can be said as a host-to-content oriented application, which means that the application cares about *what content is being requested* rather than *where the content is*.

The Internet does not fit the trend of the current applications. For example, in order to retrieve desired content such as video, a client sends the request packet of the application to the server whose IP address is specified in the destination IP address field of the packet. In this case, the corresponding response packet should be returned from the server. However, if the other servers near the client have the exactly same content, they could also reply to the request packet. Thus, this content retrieval process might not be optimal for a host-to-content oriented application. That is a limitation of the Internet for the current content access model.

To adapt the current use case of the Internet, Information Centric Networking (ICN) [4] has been proposed, which is a new networking architecture performing a shift from a host-to-host to a host-to-content communication paradigm. An ICN request packet includes the content name, not the address of the content producer (in the Internet, the IP address of the content server), and the packet goes toward the nearby content producer. As for the content producer, not only the actual content publisher but also the ICN node having the content cache can reply to this request. That means that the architecture focuses on *what content is being requested* rather than *where the content is*, and therefore, ICN is more suitable for host-to-content oriented applications

such as video service. As one of the ICN architectures, this thesis adopts Named Data Networking (NDN) [5] since currently a lot of researchers pay attention to NDN and attempt to deploy it for future Internet.

Moreover, NDN adopts a security-by-design approach and practically defends against several security attacks existing in the Internet. At the beginning, the Internet was not designed to endure security problems such as spoofing and tampering, and therefore, they happened in HTTP. Against them, currently, HTTP Secure (HTTPS) [6] is widely used in the Internet and it utilizes Secure Sockets Layer (SSL) [7]/Transport Layer Security (TLS) [8], which are security protocols. The HTTPS can solve such security problems, but as one of the side effects of the HTTPS, exploiting the host-to-host encryption method by the TLS, it invalidates in-network caching (e.g., proxy) and then degrades the content delivery efficiency. In order to eliminate these problems, an NDN packet includes the signature, which solves the above security problems and activates the in-network caching. In general, such a new technology can mitigate conventional security threats, but at the same time, the technology unfortunately creates new ones. Indeed, NDN can address spoofing and tampering, but simultaneously NDN creates several new security threats such as Interest Flooding Attack (IFA) and Content Poisoning Attack (CPA). These are two well-known security threats and the countermeasures against them have been proposed [9, 10].

1.2 Problem Statement: Information Leakage in NDN

Currently, in the business world, information leakage through a targeted attack bothers a lot of companies, which drastically impacts on their benefit and profitability [11]. For example, in 2013, the retail chain "Target" suffered from 46% drop in profits after the attack and spent more than $100 million for the sake of the system upgrade to prevent another attack [12]. In order to perform the information leakage, an attacker first sends an employee in the targeted company, for example, an email including malware, which looks like apparently a legitimate email, and makes the employee open the email to infect the computer with the malware. After that, the malware creates a communication channel to the outside attacker. Then, the attacker can remotely control the malware and finally steal confidential information from the company. Considering that the volume of the traffic encrypted by the HTTPS has been increasing, it is becoming more difficult for the company firewall to inspect and detect the malicious channel, which is also one of the side effects of the HTTPS. As this security threat results from accessing such suspicious media, one of the countermeasures would be to provide employees with cybersecurity education [13]. It is, however, almost impossible to eliminate all human errors completely. Thus, it is very critical to investigate how to prevent information leakage after a computer has been infected by malware, assuming that it is difficult for the computer to avoid the infection.

Since information leakage is one of the big issues even in the Internet and it is absolutely crucial to assess the risk before replacing the Internet with NDN completely, this thesis investigates whether a new security threat causing the information leakage happens in NDN. Assuming that (i) a computer is located in the enterprise network that is based on an NDN architecture, (ii) the computer has already been compromised by suspicious media such as a malicious email, and (iii) the company installs a firewall connected to the NDN-based future Internet, this thesis focuses on a situation that the compromised computer (i.e., malware) attempts to send leaked data to the outside attacker. NDN is basically a "pull"-based architecture and there are only two kinds of packets: *Interest* and *Data*, which are a request and a response packet, respectively. In order to retrieve content, a consumer first sends the Interest to NDN network and then obtains the corresponding Data from the producer or the intermediate NDN node. In other words, they

cannot send a Data unless they receive the Interest. Therefore, as one of the naive methods to mitigate information leakage through a Data, an enterprise network firewall can carefully inspect a Data to publish, and produce it instead of the inside employee in the network (i.e., a whitelist). In this case, all the publicly-accessible content is on the firewall as the originals or their caches. However, the firewall cannot manage a naming policy on the outside content and NDN forwarding nodes do not verify whether the name really exists. That causes a risk of information leakage through an Interest since the malware can hide information such as customer information in the Interest name and send it toward the outside attacker. The malware can pretend to access outside content, so that it is quite difficult for the firewall to detect the information leakage attack. This thesis argues that the information leakage attack through an Interest in NDN should be one of the essential security attacks at protocol level and it is important to develop the detection method of this attack.

1.3 Thesis Statement

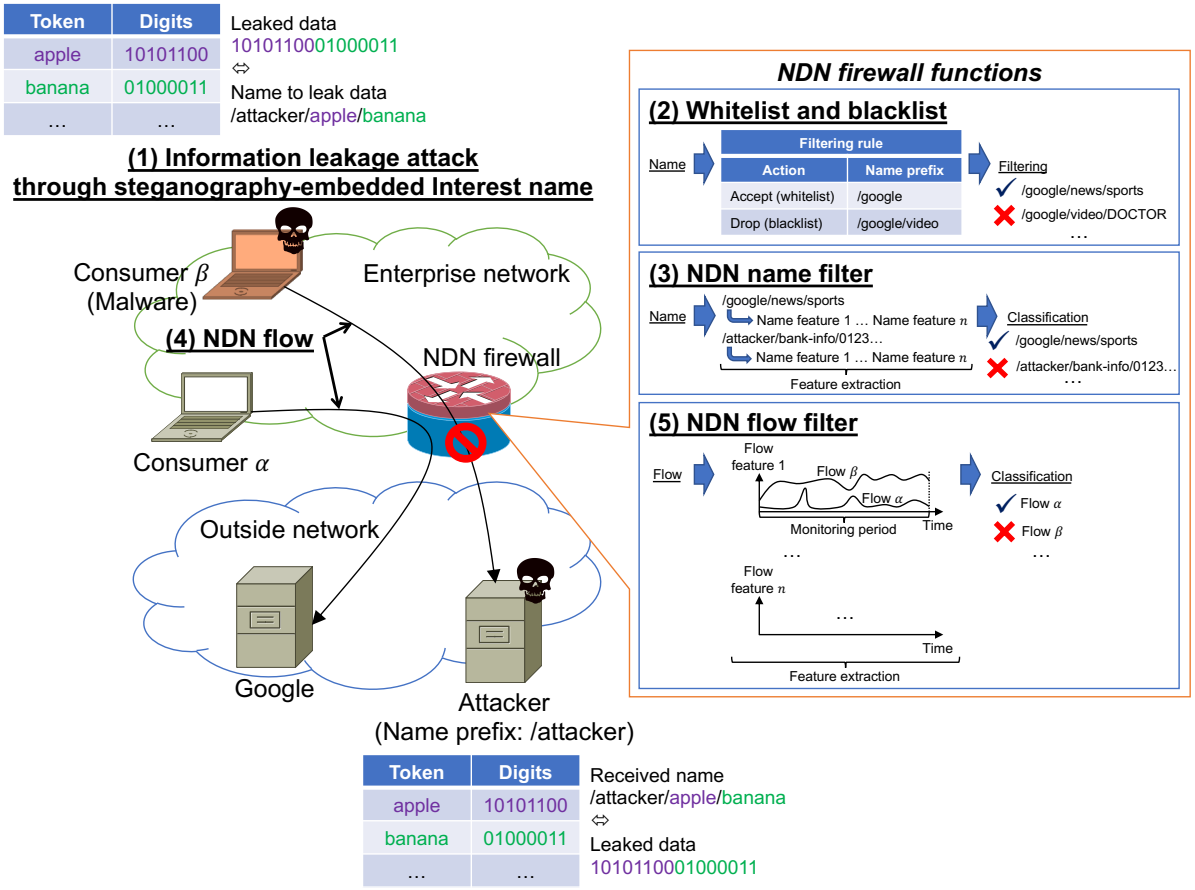


Figure 1.1: High level idea of five contributions.

The contributions of this thesis are fivefold (Fig. 1). Firstly, ***this thesis proposes an information leakage attack through a Data and through an Interest in NDN***. This thesis investigates the one through an Interest deeply, and, as a more advanced attack for the

attacker to hide the malicious activity, this thesis proposes a steganography-embedded Interest name to perform information leakage efficiently. For example, assuming that the attacker and malware share the same table for steganography, which includes each token (strings) and the corresponding digits, and the malware knows the attacker's name prefix "`/attacker`", which is routable from the malware, the malware converts the leaked data "1010110001000011" to the Interest name "`/attacker/apple/banana`" using the table of steganography and sends the attacker the Interest including the name to leak data. Then, the attacker decrypts the received name using the shared table and obtains the leaked data "1010110001000011". To the best of author's knowledge, this is the first research about the information leakage attack in NDN.

Secondly, in order to address the information leakage attack, ***this thesis proposes an NDN firewall which monitors and processes the NDN traffic coming from the consumers with the whitelist and blacklist***. To design the firewall, this thesis focuses on two requirements: (i) designing an NDN firewall independent from NDN Forwarding Daemon (NFD) [14], which decides how to forward an Interest, and (ii) performing a fast lookup of the names or name prefixes in the whitelist and blacklist. By utilizing a cuckoo filter [15], which is one of the probabilistic filters, the proposed NDN firewall provides Interest packet filtering based on the names or name prefixes in the lists that can be updated on the fly. For example, when the enterprise network operator lets the consumer utilize all of services of Google except the video service, in order to perform such access control, the operator can append rules "accept `/google`" and "drop `/google/video`" into the NDN firewall, and therefore, "`/google/news/sports`" should be accepted while "`/google/video/DOCTOR`" should be dropped. While satisfying the requirements and providing the functions, the firewall implementation achieves high performance. Specifically, the throughput degradation with the firewall is only from 0.912% to 2.34%, which will be acceptable in an enterprise network.

Thirdly, ***this thesis proposes an NDN name filter to classify a name in the Interest as legitimate or not***. Since NDN has not been deployed on a large scale, a dataset about NDN traffic does not exist. Assuming that it is highly possible for the future NDN naming policy to become one naturally evolved from the current URL naming policy, this thesis utilizes content names based on URLs collected by a web crawler to build and evaluate the proposed name filter. For example, by extracting several name features from the names, the proposed name filter judges the name "`/google/news/sports`" as legitimate and the name "`/attacker/bank-info/0123...`" as anomalous. By using search engine information and applying the name dataset to an isolation forest [16], this thesis builds NDN name filters. This thesis evaluates the performances of the name filters and shows that the proposed name filters can drastically choke the information leakage throughput per Interest, and therefore, malware has to send 137 times more Interest packets to leak information than without using the filters.

The name filter can, indeed, reduce the throughput per Interest, but to ameliorate the speed of this attack, malware can send numerous Interests within a short period of time. Moreover, the malware even can exploit an Interest with an explicit payload in the name (like an HTTP POST message in the Internet), which is out of scope in the proposed name filter and can increase the information leakage throughput by adopting a longer payload. That is the limitation of the name filter. To take traffic flow to the NDN firewall from the consumer into account, fourthly, ***this thesis proposes an NDN flow monitored at an NDN firewall***. At first, this thesis introduces the concept of NDN flow and specifies it strictly, which has not yet been standardized in NDN research [17]. Then, this thesis proposes a method to generate an NDN flow dataset analogically derived from the HTTP flow dataset in the current Internet because there is no dataset about NDN traffic.

Fifthly, in order to deal with the drawbacks of the NDN name filter, ***this thesis proposes an***

NDN flow filter to classify a flow as legitimate or not. Based on the generated NDN flow dataset, this thesis builds an NDN flow filter against the information leakage attack. For example, by monitoring the NDN flow during the specific period and extracting several flow features, the proposed flow filter identifies the flow α as legitimate and the flow β as anomalous. By applying the obtained dataset to a Support Vector Machine (SVM) [18], the performance evaluation shows that the information leakage throughput choked by the flow filter is from $1.32 \cdot 10^{-2}$ to $6.47 \cdot 10^{-2}$ times that choked only by the name filter. Thus, the flow filter complements the name filter and greatly chokes the information leakage throughput.

Here is the summary of the contributions. This thesis first proposes and investigates an information leakage attack through a Data and through an Interest in NDN, and argues that the information leakage attack through an Interest in NDN should be one of the essential security attacks at protocol level and it is important to develop the detection method of this attack (the first contribution). This thesis implements the firewall (the second contribution), which monitors and processes NDN traffic with the whitelist and blacklist, and, by a simulation using NDN names obtained from the URLs, this thesis evaluates the effectiveness of the name filter against the attack (the third contribution). Then, to address the drawbacks of the name filter, this thesis proposes an NDN flow monitored at an NDN firewall (the fourth contribution), and by a simulation using generated NDN flow dataset derived from the HTTP flow dataset, this thesis confirms the efficacy of the flow filter against the attack (the fifth contribution). The proposed filters can be easily installed in the firewall to deal with future real NDN traffic.

1.4 Thesis Organization

The rest of this thesis is organized as follows.

Chapter 2 explains the related works about this thesis and finally argues that an information leakage attack through an Interest name in NDN should be one of the essential security attacks at protocol level such as Interest flooding attack and content poisoning attack. In this thesis, it is the main motivation to investigate and solve the attack.

Chapter 3 investigates several information leakage attack models in NDN deeply. In order to assess the risk of the attack, this thesis focuses on the models.

Chapter 4 proposes an NDN firewall which monitors and processes NDN traffic coming from the consumers with the whitelist and blacklist. This is the first step to address the information leakage attack in this thesis.

Chapter 5 proposes an NDN name filter to classify a name in the Interest as legitimate or not. This name filter is one of the applications in the firewall.

Chapter 6 proposes an NDN flow monitored at an NDN firewall. This thesis shows a method to generate the corresponding NDN flow dataset from an HTTP flow dataset and utilizes the created dataset to build a more sophisticated filter complementing the NDN name filter in the next chapter.

Chapter 7 proposes an NDN flow filter to classify a flow as legitimate or not. This flow filter deals with the drawbacks of the name filter and the flow filter is also one of the applications in the firewall.

Chapter 8 summarizes this thesis and shows some perspectives of this research area.

Chapter 2

Related Work and Motivation

Contents

2.1 Introduction

The Internet has been widely used to provide a lot of applications for a long time. In particular, the web is one of the killer applications, which is a host-to-host oriented application, and in recent years, video service on the web is becoming more popular (e.g., YouTube, Netflix, etc.). The video is located in the server and can be retrieved by the HTTP [2] request message including the URL [3] to designate what content to be downloaded. This kind of application can be said as

being a host-to-content oriented application, which means that the application cares about *what content is being requested* rather than *where the content is*, and therefore, the Internet does not fit the trend of the current applications. For example, in order to retrieve desired content such as video, a client sends the request packet of the application to the server whose IP address is specified in the destination IP address field of the packet. In this case, the corresponding response packet should be returned from the server. However, if the other servers near the client have the exactly same content, they could also reply to the request packet. Thus, this content retrieval process might not be optimal for a host-to-content oriented application. That is a limitation of the Internet for the current content access model. On a security side, at the beginning, the Internet was not designed to endure security problems such as spoofing and tampering, and therefore, they happened in HTTP. Against them, currently, HTTPS [6] is widely used in the Internet and it utilizes SSL [7]/TLS [8], which are security protocols. The HTTPS can solve such security problems, but as one of the side effects of the HTTPS, exploiting the host-to-host encryption method by the TLS, it invalidates in-network caching (e.g., proxy) and then degrades the content delivery efficiency. Moreover, the URL and domain name, which is a part of the URL, are unfortunately used for an attacker to perform several attacks (i.e., name vulnerability).

In response to the new requirements, ICN [4] is a new networking architecture proposal performing a shift from a host-to-host (i.e., Internet) to a host-to-content communication paradigm. An ICN request packet includes the content name, not the address of the content producer (in the Internet, the IP address of the content server), and the packet goes toward the nearby content producer. As for the content producer, not only the actual content publisher but also the ICN node having the content cache can reply to this request. That means that the architecture focuses on *what content is being requested* rather than *where the content is*, and therefore, ICN is more suitable for massive content diffusion such as video delivery. As one of the ICN architectures, this thesis adopts NDN [5] since currently a lot of researchers pay attention to NDN and attempt to deploy it for future Internet. On the security side, NDN adopts a security-by-design approach and practically defends against several security attacks in the Internet. In order to eliminate these attacks such as spoofing and tampering, an NDN packet includes the signature, which solves the above security problems and activates the in-network caching. In general, new technology can mitigate conventional security threats, but at the same time, the technology unfortunately creates new ones. Indeed, NDN can address spoofing and tampering, but at the same time, NDN creates several new security threats. Since one of the key features in NDN is a name, this thesis surveys security threats exploiting name vulnerability in the Internet and discusses whether they can be solved in NDN. Finally, this thesis argues that an information leakage attack through an Interest name will inherit one through DNS tunneling [19], which is one of the Internet security threats, and it will be one of the new essential security attacks at NDN protocol level.

The rest of this chapter proceeds as follows. Section 2.2 explains an overview of the Internet. Section 2.3 shows Internet security threats related to HTTPS while Section 2.4 surveys three Internet security threats exploiting name vulnerability: *malicious URL* [20], *Domain Generation Algorithm (DGA)* [21], and *DNS tunneling*. Section 2.5 describes an overview of NDN and explains the naming policy under the assumption in this thesis. Section 2.6 introduces two well-known new security threats in NDN: *Interest Flooding Attack (IFA)* [9] and *Content Poisoning Attack (CPA)* [10]. Section 2.7 discusses whether Internet security threats exploiting name vulnerability can be solved in NDN, and concludes that an information leakage attack through an Interest name will occur in NDN. Section 2.8 shows an overview of machine learning which is one of the promising solutions to detect several security attacks. Finally, Section 2.9 summarizes this chapter.

2.2 Overview of the Internet

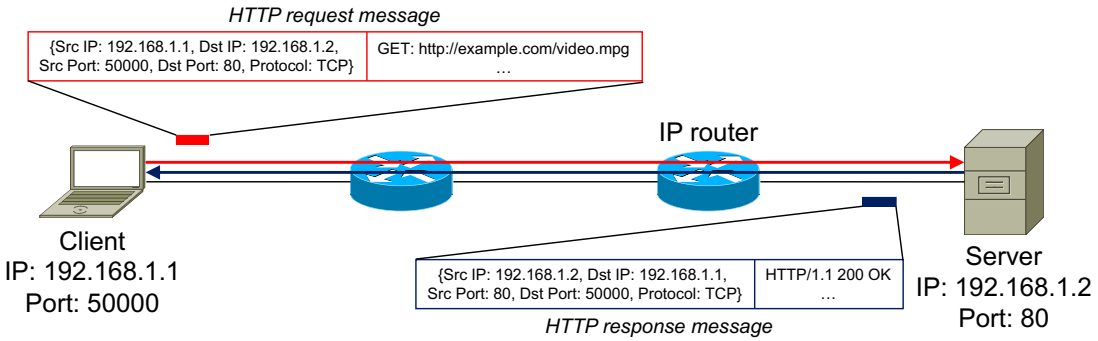


Figure 2.1: HTTP-based content access in the Internet.

The current Internet installs TCP/IP protocol suite, which is a set of communication protocols. Transmission Control Protocol (TCP) [22] is placed on the transport layer and performs accurate delivery of packets, and Internet Protocol (IP) [23], which is the primary protocol in the Internet, is placed on the Internet layer and defines an IP address to transfer Internet packets. TCP/IP creates host-to-host communication channels between hosts, so that the Internet relies on a host-to-host communication paradigm.

The top of the TCP/IP suite is an application layer, and applications such as the web are in the layer. In the Internet, the web is one of the killer applications, and in recent years, video service is becoming more popular. Content such as a video is named by a URL defined by RFC 1808 [3], which is a subset of a Uniform Resource Identifier (URI) defined by RFC 3986 [24]. RFC 1808 defines a URL as `<scheme>://<net_loc>/<path>;<params>?<query>#<fragment>` and the `<net_loc>` part indicates a host generating the URL (e.g., a Fully Qualified Domain Name (FQDN)). A user or an organization owning this host creates URLs by following its own naming policy and they should be unique in the world. According to a report from Google [25], in 2008 there were 1 trillion unique URLs in the Internet, which indicates that a lot of URLs have already been published.

In Fig. 2.1, the video with the name of "`http://example.com/video.mpg`" is located in the server whose IP address is 192.168.1.2. In order to retrieve the video, the client whose IP address is 192.168.1.1 sends the HTTP GET request message including the URL to the server, and the server replies to it by the corresponding HTTP response message. Finally, the response message is sent to the client, and the client obtains the video. This kind of the application cares about *what content is being requested* (e.g., `http://example.com/video.mpg`) rather than *where the content is* (e.g., 192.168.1.2), and it can be said as a host-to-content oriented application.

Thus, the Internet (TCP/IP) does not fit the trend of the current application. For example, as shown in Fig. 2.1, in order to retrieve the desired content, the client whose IP address is 192.168.1.1 sends the GET request packet of the application to the server whose IP address 192.168.1.2 is specified in the destination IP address field of the packet. In this case, the corresponding response packet should be returned from the server. However, the other servers near the client could reply to the request packet if they have the exactly same content. Thus, the Internet might not be optimal for such a host-to-content oriented application, and that is a limitation of the Internet for the current content access model. To provide the host-to-content oriented application more efficiently, one of the essential requirements is to perform a shift from

a host-to-host to a host-to-content communication paradigm.

2.3 Internet Security Threats

On the security side, at the beginning, the Internet was not designed to endure security problems such as spoofing (impersonation as a trustworthy source by an attacker) and tampering (intentional modification of data to cause damage by an attacker), so that the Internet addresses these problems with the application layer. In order to mitigate these problems, currently, HTTPS [6] is widely used. Unlike the standard HTTP that transmits and receives messages in plaintext, HTTPS performs some functions such as authentication of servers, encryption, and falsification detection of communication contents using SSL [7]/TLS [8] protocol, and therefore, this can prevent attacks such as spoofing, man-in-the-middle attacks, and eavesdropping.

Fig. 2.2 describes an overview of SSL protocol. The record layer receives all messages from the ChangeCipherSpec protocol, the Alert protocol, the Handshake protocol, and applications such as HTTP, encapsulates and passes them to a transport layer protocol such as TCP. The ChangeCipherSpec protocol sends the ChangeCipherSpec message to explicitly indicate that the encryption should be activated now. The Alert protocol signals an error to the connected host. The Handshake protocol is used to make a secure channel.

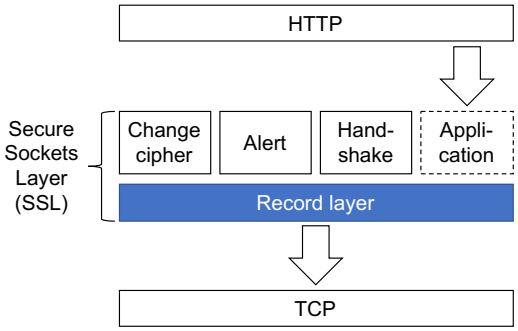


Figure 2.2: Overview of SSL protocol [26].

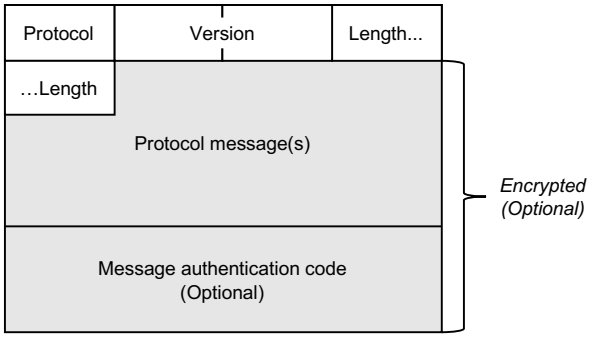


Figure 2.3: Packet format of record layer [26].

Fig. 2.3 explains a packet format of the record layer that is composed of a record layer header and a record layer message. The record layer header consists of five bytes: protocol, version, and length, which indicate the kind of protocol (e.g., ChangeCipherSpec protocol) creating the record layer message, the version of used SSL, and the length of the record layer message, respectively. The record layer message includes the protocol messages and optionally contains the message authentication code. Moreover, the record layer message can be optionally encrypted (e.g., from HTTP to HTTPS).

However, as one of the side effects of the HTTPS, by adopting a host-to-host encryption method and making an encrypted communication channel between a client and the server, it invalidates in-network caching (e.g., proxy) since the encrypted traffic is not cacheable (Fig. 2.4), so that this degrades the content delivery efficiency. Moreover, an attacker exploits new security holes and carries out several security attacks. For example, considering that the volume of the encrypted traffic has been increasing, as shown in Fig. 2.5, it is becoming more difficult for the enterprise firewall to inspect and detect the malicious communication channel between the malware and attacker since the traffic is encrypted, which is also one of the side effects of the HTTPS.

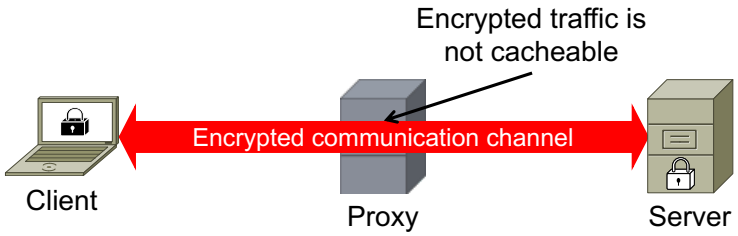


Figure 2.4: Invalidated in-network caching.

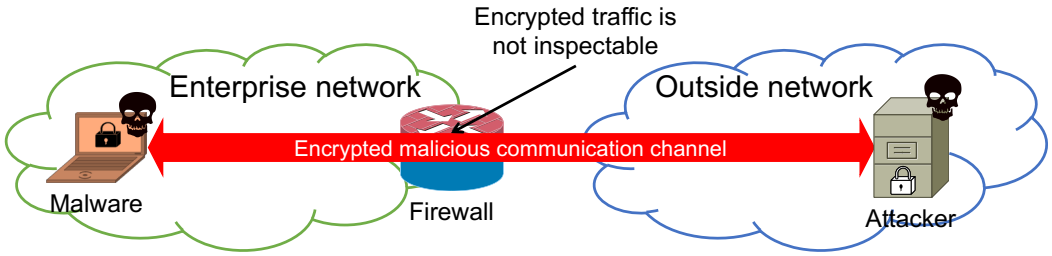


Figure 2.5: Invalidated traffic monitoring.

In order to address these side effects, several researchers investigate how to handle encrypted traffic. Naylor et al. [27] propose multi-context TLS (mcTLS) which extends TLS to introduce trusted middleboxes. This approach obtains explicit consent from clients and content servers to control the encrypted traffic, and then can realize several middlebox functions such as an enterprise firewall. In order to perform Deep Packet Inspection (DPI) directly on the encrypted traffic, Sherry et al. [28] propose Blindbox, and, from the encrypted traffic, the Blindbox searches specific tokens, which should be a trigger to detect, for example, malicious activities caused by malware. In order to extract the specific tokens, they also develop a new searchable encryption scheme, and, by applying the scheme to rules from organizations such as McAfee and tokens from the clients who tokenize the traffic, the Blindbox can perform pattern matching. Without decrypting HTTPS traffic, Shbair et al. [29] identify the applications in the HTTPS traffic by extracting the statistical features and applying them to a machine learning technique. Still, the research about how to handle encrypted traffic is ongoing.

Thus, it is needed to solve conventional security attacks, to infer new attacks, to assess the risks, and to propose the countermeasures against them in advance.

2.4 Name Vulnerability in the Internet

As mentioned previously, a content name is associated with the URL. The URL and a domain name, which is a part of the URL, can unfortunately be used by an attacker to perform several threats (i.e., name vulnerability). Internet security threats exploiting name vulnerability can be classified into three types: *malicious URL*, *Domain Generation Algorithm (DGA)*, and *DNS tunneling*.

2.4.1 Malicious URL

Malicious URLs induce people to be infected by malware (i.e., drive-by download attack) or to access phishing/spam webpages. For example, an attacker prepares "`http://www.paypal.com.`*(hash_value)*`.org/`" as a phishing URL. Some users might be attracted by token "paypal" and access the phishing URL, but the main domain is "*(hash_value)*`.org`", which is the attacker's one. Therefore, the users might obtain unexpected content. Fetterly et al. [30] investigate URL properties to distinguish spam URLs and they conclude that though the properties of the spam URLs except the host names are not useful information to identify spam webpages, the host names are likely to have a lot of characters, dots, dashes, and digits and these features should be indicative of spam URLs. In order to detect malicious URLs such as phishing, Ma et al. [20] utilize several features of the URLs such as the length of host name and a binary feature for each token extracted with several delimiters in the host name and the path URL (i.e., bag-of-words) while Blum et al. [31] take a similar approach. Akiyama et al. [32] assume that an unknown malicious URL is located in the structural neighborhood of a known malicious URL created by the same attacker (e.g., when "`http://www1.example.com/exploit1.php`" is a known malicious URL, "`http://www1.example.com/exploit2.php`" is a potentially malicious one), and propose a method to improve blacklisting. Performing a lexical URL analysis, Khonji et al. [33] report that large subsets of URL tokens are likely to reappear in future URLs and the overlap between tokens in phishing and legitimate URLs is quite small, which can be useful knowledge to build a classifier to distinguish the phishing URLs. Lin et al. [34] obtain lexical and descriptive features from URLs and detect malicious URLs using an online learning mechanism, which fits with the very short lifetime characteristic of the malicious URLs.

2.4.2 DGA

In order to create a communication channel between malware and the Command-and-Control (C&C) server, initially, malware included the hard-coded IP address of the C&C server. However, by reverse-engineering the malware, discovering the IP address, and blacklisting it, this approach was invalidated. Against the reverse-engineering, Domain Generation Algorithm (DGA) is used to algorithmically generate a lot of domain names shared between the malware and C&C server using the same seed. For example, the shared DGA produces 1,000 domain names: "`attacker0.com`", ..., "`attacker999.com`". Once the C&C server registers one of them and receives the DNS query from the malware, the malware can obtain the IP address of the C&C server and finally they build the covert channel using any kinds of protocols such as DNS, HTTPS, etc., if the use of them are not restricted by, for example, a firewall. Moreover, constantly changing the used domain name makes blacklisting such malicious domain ineffective. Yadav et al. [35] investigate frequencies of alphanumeric characters as well as bigrams in legitimate and anomalous domain names. Antonakakis et al. [21] focus on the fact that bots using the same DGA would generate similar NXDOMAIN traffic, which indicates that there are no IP addresses for the requested domain names, and then utilize n-gram features, entropy-based features, and structural domain features to cluster the anomalous domain names. Thomas et al. [36] also utilize NXDOMAIN traffic to find such anomalous domain names. Grill et al. [37] use only NetFlow/IPFIX statistics, expecting that the number of DNS queries created by malware during a small time interval should be more than the number of actually accessed unique IP addresses. Plohmann et al. [38] perform a comprehensive measurement study of DGAs.

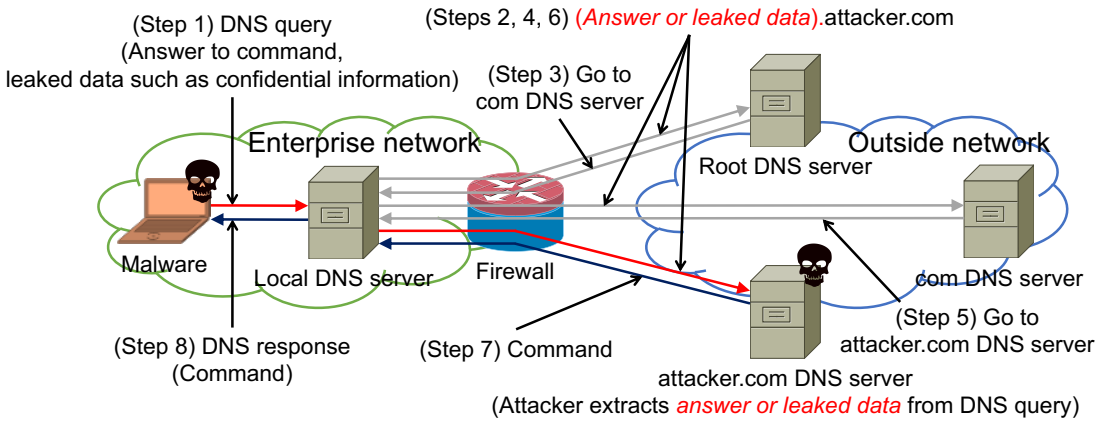


Figure 2.6: Overview of DNS tunneling.

2.4.3 DNS Tunneling

After making a channel between malware and the attacker in some way such as DGA, by exploiting domain names in DNS queries and the corresponding DNS responses, they bypass a firewall and perform tunneling of data and commands. This threat is called DNS tunneling. Fig. 2.6 describes an overview of DNS tunneling by explaining how to perform an information leakage attack through DNS tunneling as one of the use cases. For example, the malware and attacker have already shared a domain name "`attacker.com`" for their channel, and, to obtain a command to search the confidential information in the enterprise network from the attacker, the malware generates "(*get_ command*)`.attacker.com`" and sends it as the DNS query to the local DNS server in the enterprise network (Step 1). Since this DNS query is generated by the malware for the purpose of collecting the commands, the corresponding DNS response is not cached in the local DNS server. Thus, following a conventional process to resolve a domain name, the local DNS server iteratively asks the root (Steps 2 and 3), the "`com`" (Steps 4 and 5), and the "`attacker.com`" DNS server to resolve the name (Step 6). Then, the "`attacker.com`" DNS server obtains the information "(*get_ command*)" and replies to the DNS query with the corresponding DNS response, which includes the command (Step 7), and the response is finally forwarded to the malware via the local DNS server (Step 8). After repeating a process of sending the answer to the command and obtaining a new command, at the end the malware steals the confidential information and leaks it to the outside attacker in the same manner (i.e., exploiting domain names to leak the information in DNS queries and the DNS responses). Leijenhorst et al. [39] show that DNS tunneling can achieve up to 110 KB/s in throughput with DNScat [40], which is one of the DNS tunneling applications, but it adds huge traffic overhead. Merlo et al. [41] compare several DNS tunneling tools in terms of throughput, RTT, and overhead. In order to detect DNS tunneling, some countermeasures have been proposed. Born et al. [42] and Qi et al. [43] analyze character frequencies of domain names and detect DNS tunneling. Differently, Farnham [44] investigates not only such a payload analysis but also a traffic analysis such as analyzing count and frequency of queries. Ellens et al. [45] also perform a traffic analysis using a flow defined in RFC 3917 [46] and report that appropriate metrics to detect the tunneling are, for example, bytes per flow or the number of flows over time. Kara et al. [47] focus on DNS TXT record and detect DNS tunneling activities with this record. Aiello et al. [48] analyze simple statistical properties such as inter-arrival time of packets and packet size and apply them to ma-

chine learning techniques. These countermeasures, however, are only effective to detect attacks generated by some specific tools such as DNScat or some malware such as Morto worm [49], which does not mean that the countermeasures can eliminate the threat of the attack essentially. Xu et al. [19] conclude that DNS-based botnet C&C channel, which is based on DNS tunneling, is "feasible, powerful, and difficult to detect and block".

Takeaway

Sections 2.4.1, 2.4.2, and 2.4.3 showed typical Internet security threats exploiting name vulnerability. This vulnerability is derived from a URL and domain name in the URL. Still, these threats are serious problems and the related researches are ongoing. In the next section, this thesis introduces NDN and discusses whether the name vulnerability in the Internet can be inherited even in NDN since a name, which is similar to the URL and domain name, is one of the key features in NDN.

2.5 Overview of NDN

In order to perform a shift from a host-to-host to a host-to-content communication paradigm and adapt the current Internet use cases, NDN has been proposed as a clean-slate architecture for future Internet. In NDN, a request packet from the consumer includes a desired content name, not an address of the content producer, and therefore, the corresponding response packet is returned from the content producer or the intermediate NDN node having the cached response packet to the consumer (i.e., host-to-content). This section introduces an overview of NDN in terms of a packet format and an architecture, and explains the naming policy under the assumption in this thesis.

2.5.1 Packet Format

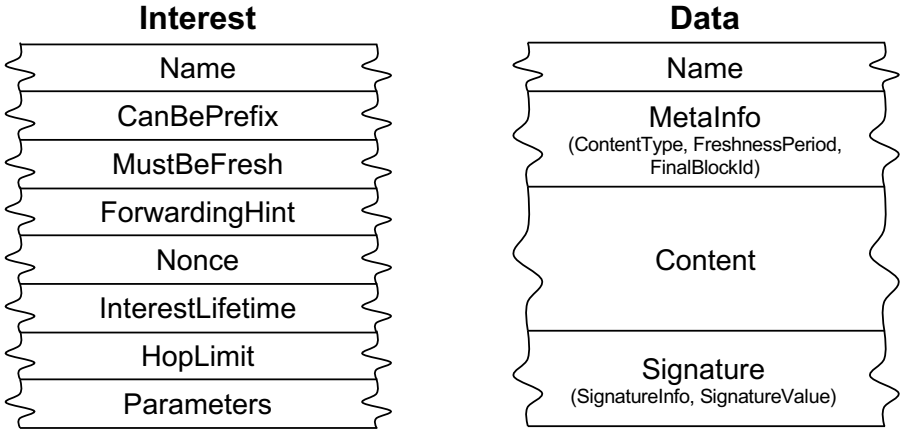


Figure 2.7: NDN packet format about Interest and Data [50].

NDN is a "pull"-based architecture and there are only two kinds of packets: *Interest* and *Data*, which are a request and a response packet, respectively. Thus, by sending an Interest to NDN network, the consumer retrieves the corresponding Data. Fig. 2.7 indicates the current NDN packet format about an Interest and a Data [50]. Unlike the Internet packets shown in Fig. 2.1,

the Interest does not have the consumer identification (in the Internet, the source IP address), and therefore, NDN strengthens consumer anonymity by default. The Interest is composed of Name, CanBePrefix, MustBeFresh, ForwardingHint, Nonce, InterestLifetime, HopLimit, and Parameters. All elements except the Name (i.e., a content name), which will be explained in Section 2.5.3, are optional. However, as for the Nonce, which is used to detect looping Interests, it is required when an Interest is forwarded over the network links. When the CanBePrefix appears in the Interest, the Name in the Interest is the name prefix, exact, or full name of the corresponding Data. Otherwise, the Name is either the exact or full name of the Data. In the case that the consumer wants to retrieve a fresh Data, the Interest can contain the MustBeFresh. The ForwardingHint includes a list of name delegations. The InterestLifetime expresses the time remaining before the Interest times out while the HopLimit shows the number of hops the Interest is allowed to be forwarded. The Parameters can transfer arbitrary data which parameterizes the request for Data.

On the other hand, the Data consists of Name, MetaInfo, Content, and Signature. The MetaInfo and the Content are optional in the Data while the Name and the Signature are necessary. The Name in the Data corresponds to the one in the Interest. The MetaInfo includes ContentType, FreshnessPeriod, and FinalBlockId. The ContentType indicates the type of the content such as a public key, and the FreshnessPeriod indicates how long an NDN node should wait after the arrival of the Data before marking it non-fresh. The FinalBlockId can express the last explicit name component of the final block. The Content is the data in the Data. The Signature is composed of SignatureInfo, which describes the signature algorithm and information to retrieve the public key, and SignatureValue, which is the value of the signature.

2.5.2 Architecture

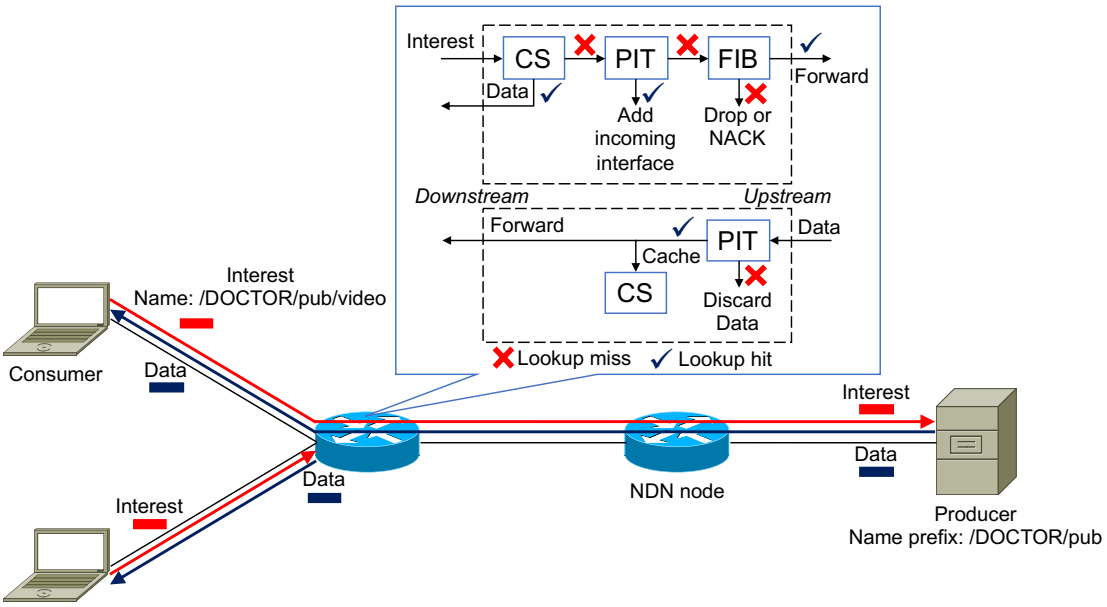


Figure 2.8: Overview of NDN architecture [5].

Fig. 2.8 describes an overview of an NDN architecture. An NDN node consists of three components: *Forwarding Information Base (FIB)*, *Pending Interest Table (PIT)*, and *Content Store (CS)*. FIB has routing information to forward Interests. PIT keeps all pending Interests and

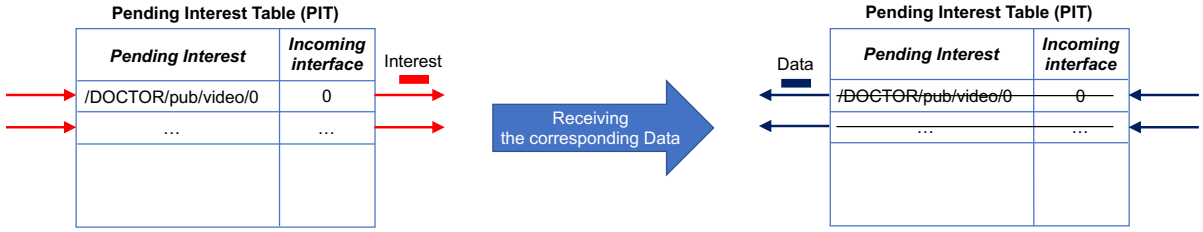


Figure 2.9: Pending Interest Table (PIT).

the arrival interfaces of the Interests into entries and each entry is removed when the matching Data is received or a timeout occurs (Fig. 2.9). CS is used for caching Data packets with the caching replacement policy. When a producer publishes content, the producer first has to advertise the name prefix (e.g., `/DOCTOR/pub`) for name routing to the NDN network. Then, in order to retrieve content, a consumer specifies the desired content name including the producer's name prefix (e.g., `/DOCTOR/pub/video`) in the Interest, and sends it to the next NDN node. After receiving the Interest, the NDN node confirms whether the corresponding Data is cached in the CS. If the Data is cached, the NDN node sends it to the consumer from the CS. Otherwise, the NDN node checks if the information about the Interest such as the name has already been in the PIT. If it has been registered in the PIT, that means that the same Interest has already come to the PIT and the corresponding Data has not arrived yet, and therefore, the NDN node just adds the incoming interface of the Interest to the PIT and does not forward the Interest to the next NDN node. If not, the NDN node registers the name and incoming interface of the Interest in the PIT, and, after that, the NDN node forwards the Interest to the next NDN node following the FIB. This process should be repeated until the Interest discovers the corresponding Data, and the Data can be returned from the content producer or the intermediate NDN node having the cached Data. When the Data is sent to the consumer, the Data follows the reverse path of the Interest, which can be configured by the PITs of the passed NDN nodes. While delivering the Data, each of the NDN nodes caches the Data based on the caching algorithm installed in the CS. Finally, the consumer receives the Data including the content producer signature, and then the consumer verifies the signature. Thus, by the data-centric security, NDN resolves spoofing and tampering, which are the conventional security problems in the Internet.

2.5.3 Naming Policy

The NDN naming policy [51] refers to a URI, which means that an NDN name has hierarchical structure. The naming policy specifies `ndn` in `<scheme>` part but ignores `<authority>` part, which corresponds to `<net_loc>` part in a URL. In order to request a specific Data, the NDN name in the Interest may include the SHA-256 digest of the corresponding entire Data packet as the last name component while the full name of every Data includes the digest. Moreover, according to "NDN Technical Memo: Naming Conventions" [52], the name can carry the segmenting, versioning, time stamping, and sequencing.

In the Internet, there is a URL similar to the NDN name based on a URI. As mentioned in Section 2.2, a lot of URLs have already been published, so that this thesis believes that people are accustomed to the current URLs. Considering a high affinity between a lot of already published URLs and the people who are familiar with the URLs, this thesis assumes that it is highly possible for the future NDN naming policy to become one naturally evolved from the current

URL naming policy. Indeed, Schnurrenberger [53] also discusses content names based on URL datasets. Moreover, in the case that the NDN naming policy is extended from the current URL naming policy, it is very easy to translate the current numerous content names distributed in the Internet to the corresponding NDN names. Thus, this thesis predicts that the naming policy of NDN will be based on that of the URL.

As mentioned previously, in the NDN naming policy, `<authority>` part is ignored. On the other hand, this thesis considers that an authority described in `<net_loc>` part can define the name uniquely, so that this idea is different from the NDN naming policy. However, this thesis assumes that the same discussion can be applied to the naming policy except `<authority>` part. Fig. 2.10 shows a URL-based NDN name which this thesis adopts. `<path>` part is constructed by several name components. This thesis omits `<params>` part because mainly this part is an option in File Transfer Protocol (FTP).

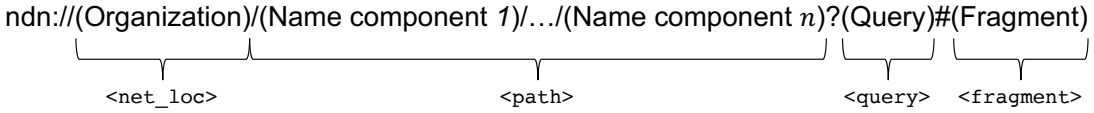


Figure 2.10: URL-based NDN name.

2.6 NDN Security Threats

This thesis previously showed that several kinds of threats happen in the Internet. In general, new technology can mitigate conventional security threats, but at the same time, the technology unfortunately creates new ones. For example, NDN can address spoofing and tampering, which are security problems in the Internet, by producer's signature, but *Interest Flooding Attack (IFA)* and *Content Poisoning Attack (CPA)* are well-known new security threats in NDN. This section introduces these new security threats in NDN.

2.6.1 Interest Flooding Attack

PIT is stateful since it keeps all pending Interests and the arrival interfaces of the Interests into entries (Fig. 2.9). Thus, an attacker can exhaust the PIT easily using a number of malicious Interests which do not retrieve the corresponding Data packets at all because there are no Data packets satisfying the Interests. This attack is called *Interest Flooding Attack (IFA)* and it is one of the well-known NDN security threats in an NDN research community. Against the IFA, Dai et al. [54] propose an Interest traceback as a countermeasure. In this countermeasure, after detecting the attack in some way (e.g., preparing a threshold for PIT size), the spoofed Data packets corresponding to the malicious Interests are generated by a traceback router, and then the Data packets should be sent to the Interest originator, which is the attacker, since the Data packets follow the same path of the Interest packets thanks to the PITs. However, when even one PIT on the path is overflowed and therefore the spoofed Data cannot be forwarded to the attacker, the countermeasure does not work well. Afanasyev et al. [9] utilize two features of NDN: routers managing state about the forwarded Interests, and Data traffic following the reverse path of the Interest traffic, and propose several countermeasures. One of the proposals can exclude malicious consumers while preventing legitimate consumers from service degradation. Compagno et al. [55] propose countermeasures relying on both local metrics (e.g., PIT usage in each router)

and collaborative techniques (i.e., collaboration between nearby routers). Unlike these proposed countermeasures, against the IFA, Ghali et al. [56] propose a stateless architecture without PITs and the architecture is based on Routable Backward Names (RBNs), which is a routable prefix similar to a source IP address and is used to send content back to the requesting consumers. Indeed, the IFA can be mitigated by eliminating the PITs, but at the same time the architecture looses some advantages such as consumer anonymity.

Even if a network operator has a method to detect the source of the IFA and exclude the attacker as soon as possible before the malicious Interests diffuse in the network, an Interest does not have the consumer identification as explained in Section 2.5.1. Therefore, the attacker can easily escape from tracking, and the attack and detection might be a cat-and-mouse game.

2.6.2 Content Poisoning Attack

In order to avoid spoofing and tampering, a Data packet has the producer's signature. Indeed, a consumer receiving a forged Data can exclude it by verifying the signature, but the signature verification overload is generally quite heavy. Thus, it is normally optional for intermediate NDN nodes to check the signature, and finally, the CSes in the intermediate nodes are polluted, which is called *Content Poisoning Attack (CPA)*. Ghali et al. [10] propose a content ranking algorithm to detect and mitigate the CPA, which is based on consumer actions indicating if the consumer retrieves a forged content. As one of the drawbacks in the proposal, it is possible for a malicious consumer to negatively effect the ranking algorithm by sending wrong feedback. Kim et al. [57] doubt that Data packets cached in CS which might never be utilized should be verified, and propose a scheme where a signature verification is performed for only Data packets when they are actually served from the CS. However, a malicious consumer first requests and retrieves a forged Data, which makes the CS cache the Data, and then, by malicious consumer's sending the Interest again, finally the node is forced to verify the Data in the CS. DiBenedetto et al. [58] develop evasion strategies which assist a forwarding strategy in NFD, which decides how to forward an Interest, and the evasion strategies avoid forwarding an Interest to malicious content. Mai et al. [59] propose a security monitoring plane which extracts several metrics from NDN nodes (e.g., the number of incoming and outgoing Interests, the number of PIT entries), and using Bayesian Network (BN) [60], they perform an anomaly detection constructed by these metrics against the CPA. Since the proposed monitoring plane is generalized, the monitoring plane might be effective to detect not only CPA but also IFA.

The CPA and detection might also be a cat-and-mouse game since the attacker does not have to generate the signature using the attacker's private key, and therefore, the forged Data does not have the attacker identification.

2.7 Name Vulnerability in NDN

One of the key features in NDN is a name, and this key feature is unfortunately exploited to perform IFA. In other words, this IFA uses name vulnerability. As explained in Section 2.4, in the Internet, there is also name vulnerability. From the knowledge about the name vulnerability, one research question is "***Is name vulnerability in the Internet inherited even in NDN?***"

This section discusses whether the Internet security threats exploiting name vulnerability can be solved in NDN. Table 2.1 summarizes a taxonomy of name vulnerability in NDN inheriting the one in the Internet. Malicious URL, DGA, and DNS tunneling will turn into ***malicious name***, ***Name Prefix Generation Algorithm (NPGA)***, and ***name tunneling***, respectively.

2.7.1 Malicious Name

Malicious names will be exploited to perform the same attacks as the ones in the Internet (e.g., drive-by download attack). The malicious names in NDN will be detected by several countermeasures in the Internet introduced in Section 2.4.1 since even in NDN the malicious names must attract users to access them, so that the names might also be constructed by specific tokens used in the Internet (e.g., "paypal" for phishing).

2.7.2 NPGA

Like DGA, Name Prefix Generation Algorithm (NPGA) will also be used to establish a C&C communication channel between malware and the attacker. The attacker algorithmically generates name prefixes using the shared NPGA (e.g., 1,000 name prefixes: "`/attacker0`", ..., "`/attacker999`") and selects one of the name prefixes, which must be advertised to the NDN network and becomes routable from the malware. When sending an Interest with the advertised name prefix and receiving the Data from the attacker, the malware can confirm that the one-way channel to send Interests and receive the corresponding Data packets has been built successfully. As for reusability of the remedies in the Internet against the NPGA, though, for example, a frequency analysis should be available in both of the Internet and NDN, available properties in DNS are different from ones in NDN, so that the proposed remedies cannot be reused to detect the NPGA. For instance, according to RFC 1035 [61], the number of characters in a domain name has to be 255 characters or less and also the one in each label is restricted to 63 characters or less, while the length of an Interest name and the one of the name prefix in NDN are variable. In addition, patterns of DNS traffic, which resolves a domain name, should be different from ones of NDN traffic, which retrieves content. Thus, this thesis argues that there is a need to propose new countermeasures against the NPGA.

2.7.3 Name Tunneling

Like DNS tunneling, name tunneling will be utilized to perform tunneling of data and commands by Interest names in the Interests and the corresponding Data packets. In terms of throughput per name, name tunneling achieves higher throughput than DNS tunneling thanks to variable length of the Interest name. Proposing new countermeasures against the name tunneling should be needed with the same reason shown in Section 2.7.2.

Table 2.1: Taxonomy of name vulnerability in NDN inheriting one in the Internet

Feature	Malicious URL	DGA	DNS tunneling
Substitution	Malicious name	Name Prefix Generation Algorithm (NPGA)	Name tunneling
Potential attack	Drive-by download attack, spam/phishing	Establishing of C&C communication channel	Tunneling of data and commands
Reusability of remedies in the Internet	Yes	Partially, yes, but **available properties in DNS are different from the ones in NDN**	

2.7.4 Has Name Vulnerability Inherited from the Internet to NDN Been Discussed in NDN Research Area?

As for an Interest name, there are three kinds of the name: a human-readable name, a non-human-readable name, and a combination of a human-readable and non-human-readable name. NDN team encourages a use of human-readable clear-text strings [52]. On the other hand, in order not to use a public key needed for data authentication, Baugher et al. [62] propose a self-verifying name, which is a hash value (i.e., non-human readable) of the corresponding read-only Data, and the name is retrieved from Catalog associating a hash name with the real name, which is optionally signed by a trusted party. Ghali et al. [63] apply a cryptographic hash function to (usually human-readable) application-layer names in order to translate them into network-layer names, which provides some benefits for FIB, PIT, and CS such as faster lookup of the elements stored in them. Moreover, in order to realize some use cases such as evading censorship [64, 65, 66], it is needed to apply an encryption method to an Interest name. These name researches focus on performance improvement or name privacy issues, but they do not mention name vulnerability inherited from the Internet to NDN at all.

In order to mitigate the threats derived from the name vulnerability, one of the solutions is to build a filter to check NDN traffic, which can be installed in an NDN firewall. Goergen et al. [67] propose a firewall for Content Centric Networking (CCN) [68], whose architecture is quite similar to NDN. According to the investigation of this thesis, their research is the first one describing firewall use in CCN, but the authors do not mention the name vulnerability at all, and moreover, the firewall source code is not available.

Takeaway

Sections 2.7.1, 2.7.2, and 2.7.3 showed new NDN security threats exploiting name vulnerability, which are inherited from the Internet, and Section 2.7.4 showed that there are no researches to point out the name vulnerability. As one of the attacks with the name vulnerability, an information leakage attack through name tunneling can happen by imitating one through DNS tunneling (see Section 2.4.3). Since information leakage is one of the big issues even in the Internet and it is absolutely crucial to assess the risk before replacing the Internet with NDN completely, this thesis investigates the details of the attack in NDN deeply, proposes some countermeasures, and assesses the risk of the attack. To address the information leakage problems in NDN comprehensively, the next chapter explains two attack models: an information leakage attack through a Data and one through an Interest, which equals one through name tunneling. To the best of author's knowledge, this work is the first one to research the attack, and this thesis argues that the attack should be one of the essential security attacks at protocol level such as IFA and CPA.

2.8 Overview of Machine Learning

In order to address the information leakage attack, as one of the promising approaches, this thesis adopts machine learning techniques. Recently, the machine learning techniques has attracted not only research world but also business world since the techniques provide several benefits such as reducing operational cost thanks to automation of work. There are many applications for machine learning: image recognition, stock price prediction, anomaly detection, etc. In network research area, the machine learning is also adopted to detect several security attacks, and indeed, previously introduced research works exploit several machine learning techniques [59, 48, 20, 31, 33, 34, 35, 21, 29].

Typically, there are two broad machine learning categories.

- Supervised learning

  The objective of the supervised learning is to learn a rule which associates inputs with the outputs. After making the rule, the supervised learning predicts the new outputs with the maximum accuracy for given new inputs. Fig. 2.11 shows an example of classification. There are two kinds of inputs: one labeled as +1 and the other labeled as -1. From these labeled inputs, a classification algorithm learns the rule to classify new inputs as +1 or -1. The widely used supervised learning algorithms are a Support Vector Machine (SVM) [18], a random forest [69], etc. The SVM and random forest are used to classify inputs to one category, and one of the applications using them is spam filtering.

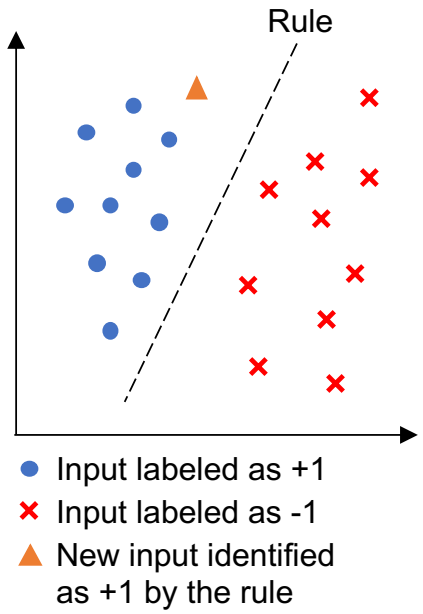


Figure 2.11: Example of classification.

- Unsupervised learning

  The objective of the unsupervised learning is to learn structure in unlabeled inputs. After understanding the structure, the unsupervised learning can tell the user the hidden patterns in the inputs. Fig. 2.12 shows an example of clustering. A clustering algorithm learns the structure of the unlabeled inputs and categorizes them into three clusters. The widely used unsupervised learning algorithms are isomap [70], a one-class SVM [71], an isolation forest [16], etc. The isomap is a nonlinear dimensionality reduction method, and therefore, it is useful to visualize high-dimensional data points as low-dimensional ones. The one-class SVM and isolation forest are useful to perform anomaly detection in the case that there are not enough malicious samples, and these algorithms classify inputs to the inliers (observations explained by underlying probability density function) and outliers (observations deviating too much from the inliers).

One of the advantages of the supervised learning is that the results from it are more accurate than those from the unsupervised learning since the dataset to make a model is labeled. However, labeling the dataset itself is one of the challenging works, and therefore, it is one of the

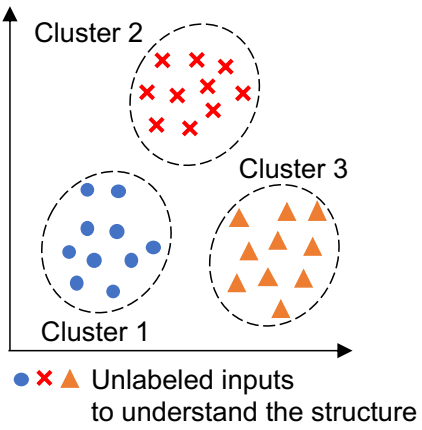


Figure 2.12: Example of clustering.

disadvantages of the supervised learning. On the other hand, the unsupervised learning does not need labeled dataset and preparing unlabeled dataset is easier, but this generates less accuracy results.

Since NDN has not been deployed on a large scale, there is neither a legitimate NDN traffic dataset nor a malicious one causing an information leakage attack through an Interest. In order to perform a risk analysis of the attack, this thesis utilizes a dataset from the current Internet and converts it to a synthetic dataset. Then, to build some countermeasures against the attack, in Chapters 5, 6, and 7, this thesis applies some supervised and unsupervised learning techniques to the created dataset and analyzes the risk of the attack using the countermeasures.

2.9 Summary

This chapter first explained an overview of the Internet and clarified the problems of the host-to-host communication paradigm and three Internet security threats exploited name vulnerability: malicious URL, DGA, and DNS tunneling. Next, this chapter described an overview of NDN, which performs a shift from a host-to-host to a host-to-content communication paradigm and resolves the problems in the current Internet, and the naming policy. Considering a high affinity between a lot of already published URLs and the people who are familiar with the URLs in the Internet, this thesis assumes a URL-based NDN name. Then, this chapter introduced two well-known new security threats in NDN: IFA and CPA. Moreover, since one of the key features in NDN is a name, this chapter discussed whether Internet security threats exploiting name vulnerability (i.e., malicious URL, DGA, and DNS tunneling) can be solved in NDN. This chapter concluded that the above three threats will turn into malicious name, NPGA, and name tunneling, respectively, and this chapter argued that an information leakage attack through name tunneling (i.e., an information leakage attack through an Interest name) is one of the essential security attacks at protocol level. Finally, this chapter showed an overview of machine learning and this thesis exploits some machine learning techniques to perform a risk analysis of the information leakage attack.

The next chapter describes information leakage attack models in NDN deeply. Then, this thesis proposes an NDN firewall and two countermeasures against the information leakage attack: an NDN name filter and an NDN flow filter.

Chapter 3

Information Leakage Attack Models in NDN

Contents

3.1 Introduction

Chapter 2 concluded that an information leakage attack through an Interest name will occur in NDN. Assuming that a computer inside an enterprise network, which is based on an NDN architecture and installs a firewall connected to the NDN-based future Internet, has already been compromised by suspicious media such as a targeted email, this chapter focuses on a situation that the compromised computer (i.e., malware) attempts to send leaked data to the outside attacker. To address information leakage problems in NDN comprehensively, this chapter investigates two attack models: an information leakage attack through a Data and one through an Interest. To the best of author's knowledge, this is the first research about the information leakage attacks in NDN.

The rest of this chapter proceeds as follows. Section 3.2 explains information leakage attacks in NDN. Though mainly this thesis discusses an information leakage attack through an Interest, this chapter also explains an information leakage attack through a Data, which possibly happens but can be solved easily (see Section 5.3.1). Section 3.3 discusses an information leakage attack through available elements other than a name in the Interest, and finally, Section 3.4 summarizes this chapter.

3.2 Information Leakage Attacks in NDN

This section first introduces an information leakage attack through a Data, and then one through an Interest. Moreover, as for the information leakage attack through an Interest, this section

shows a more sophisticated attack to exploit an Interest name than just adding leaked data into a name: generating a steganography-embedded Interest name, which looks legitimate at first glance (in fact, a malicious one to leak data).

3.2.1 Information Leakage Attack through Data

After collecting confidential information from the enterprise network, the malware can create Data packets of the information and associate them with the names which the outside attacker knows. In this case, in order for the attacker to retrieve the Data packets using the corresponding Interests, the name prefix used in the names has to be routable from the attacker to the malware.

For example, Fig. 3.1 assumes that two name prefixes "`/DOCTOR/priv`" and "`/DOCTOR/pub`", which are created by the enterprise, are utilized to share the private and public content, respectively. When the employees perform private content sharing between them inside the enterprise network, the name prefix "`/DOCTOR/priv`" should be used. If an Interest with the name prefix is sent to the enterprise network from the outside network, based on the name prefix "`/DOCTOR/priv`", the firewall drops the Interest (i.e., a blacklist). The name prefix "`/DOCTOR/pub`" is used to publish the public content such as a web page, but the malware can also exploit the name prefix for information leakage as follows. In advance, the attacker and malware share the same name components which should be appended to the name prefix "`/DOCTOR/pub`", and then the attacker sends an Interest including the name "`/DOCTOR/pub/`*(the shared name components)*" (Step 1) and retrieves the Data created by the malware, which should also include the corresponding Interest name (Step 2). Goergen et al. [67] propose a firewall to prevent some kinds of content (e.g., documents about .doc, .pdf, etc.) from being shared externally. However, they do not discuss the attack caused by the malware at all, so that their firewall is not enough against such a more sophisticated attack.

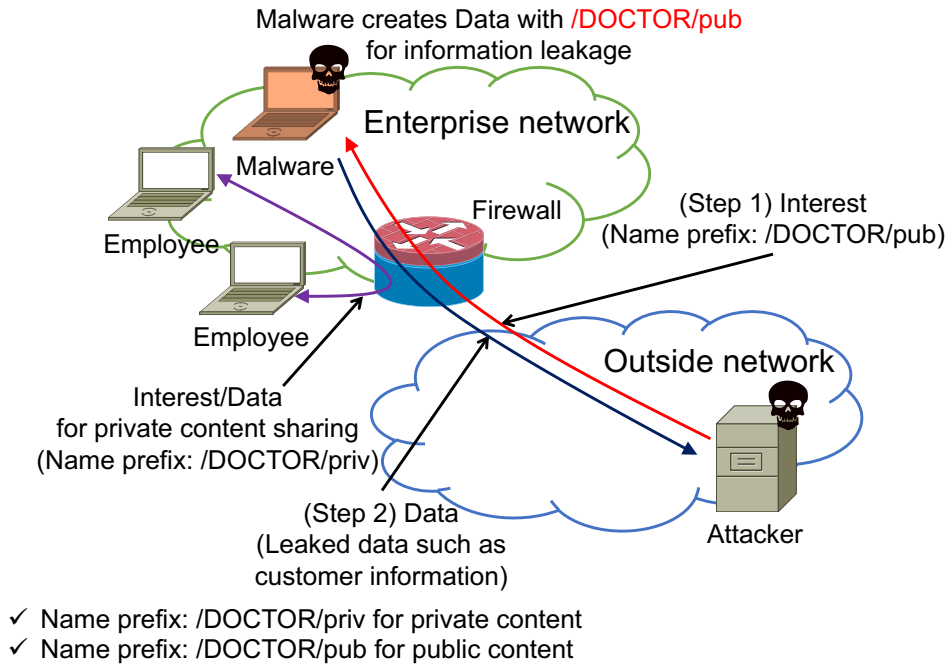


Figure 3.1: Information leakage attack through Data.

3.2.2 Information Leakage Attack through Interest

Assuming that attacker's name prefix is routable from the malware, by hiding (or just adding) leaked data into Interests and sending them out, from the enterprise network, the malware can leak the confidential information to the attacker. Especially, when the malware exploits the content names in the Interests to hide the data, it is very difficult for the company firewall to detect the information leakage since the Interests seem to attempt to retrieve outside producer's Data packets associated with the names. This thesis focuses only on an information leakage attack through an Interest name, and as for exploiting the other available elements in the Interest such as Nonce [50] to perform the information leakage attack, Section 3.3 discusses that briefly.

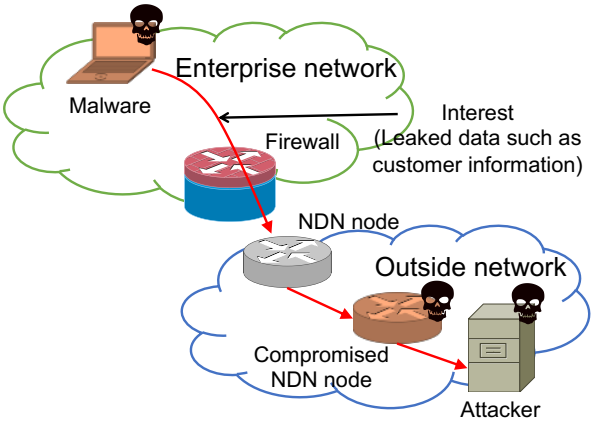


Figure 3.2: One-way Interest.

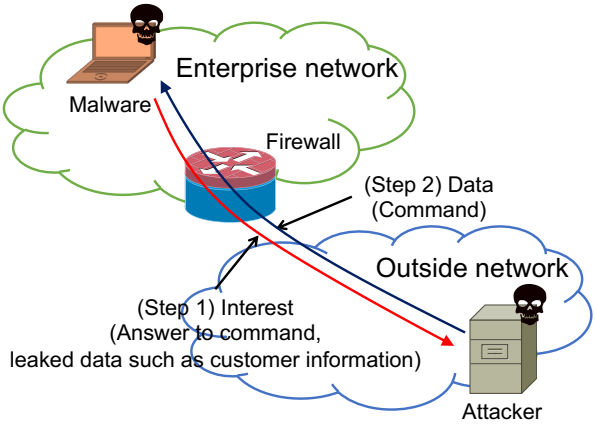


Figure 3.3: Interest/Data.

Table 3.1: Taxonomy of information leakage attack through Interest

Feature	One-way Interest	Interest/Data
Malware remote control	No	Yes
Retransmission	No	Yes
Attacker anonymity	Yes	No†
Erasure coding	Yes	No
PIT overflow	Yes	No

† Yes, for some cases (exploiting bots, etc.).

There are two possible methods to perform an information leakage attack through an Interest: a method exploiting one-way Interest (***one-way Interest method***, Fig. 3.2) and one exploiting an Interest with the corresponding Data (***Interest/Data method***, Fig. 3.3). The features of these methods are summarized in Table 3.1.

As for the one-way Interest method (Fig. 3.2), the malware transmits the leaked data out of the enterprise network by sending the malicious Interests to the attacker. The malware has to know the name prefixes toward the compromised NDN nodes (e.g., a Wi-Fi AP installed by the attacker) in order to forward the Interests to the attacker. These name prefixes do not need to be the attacker's ones, which means that the malware can abuse legitimate name prefixes to send malicious Interests toward the compromised nodes and the attacker can extract the data from them. Therefore, using the legitimate name prefixes might help the malware

hide the activity. In this method, the attacker does not reply with any Data packets to the malware, so that the attacker cannot have a fine-tuned control of the malware. Therefore, in the case that some Interests have been lost, the attacker cannot request for the retransmission of the missing information. That can happen when the firewall drops the Interests or the PIT is overflowed. In order to deal with the dropped Interests, the attacker might use erasure coding such as LT codes [72] and Raptor codes [73]. Although this method is not the most efficient to leak information, the main advantage of the method is to preserve attacker anonymity since the attacker just receives malicious Interests and does not reply to them with any Data packets including the attacker's signature.

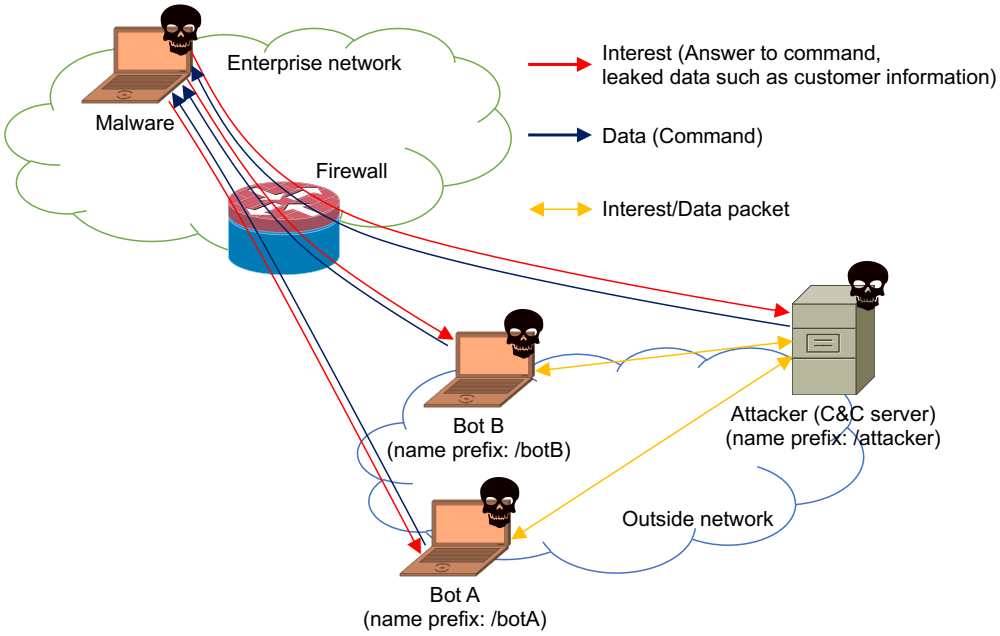


Figure 3.4: 1-to-1 and 1-to-N attack model (i.e., malware to attacker and to N bots).

As for the Interest/Data method (Fig. 3.3), assuming that attacker's name prefix is routable from the malware, the attacker explicitly controls the malware by utilizing Interest (Step 1) and the Data packets (Step 2). Thus, the attacker can communicate with the malware. In the case of a packet drop, the attacker can request for the retransmissions of the missing Interest packets by exploiting the Data packets, so that there is no need to utilize erasure coding. This method is more efficient than the one-way Interest method, but once the attack is detected, it is possible for the attacker to be tracked as the attacker's signature has been included in the Data packets. The attacker, however, can control the bots remotely and avoid being tracked. This thesis names the attack model for the malware to directly communicate with the attacker a ***1-to-1 attack model*** (i.e., malware to the attacker). Moreover, this thesis names the attack model for the malware to indirectly communicate with the attacker, which should be a C&C server, through the bots, a ***1-to-N attack model*** (i.e., malware to the N bots). Fig. 3.4 illustrates these two attack models. In the 1-to-N attack model, the malware should know the bots' name prefixes (e.g., `/botA`, `/botB`) and they should be routable from the malware.

Generating a ***steganography-embedded Interest name*** pretending a legitimate one (in fact, a malicious one to leak data) should be a more sophisticated attack to exploit an Interest name than just adding leaked data into a name. As a similar use case of steganography, Mason et

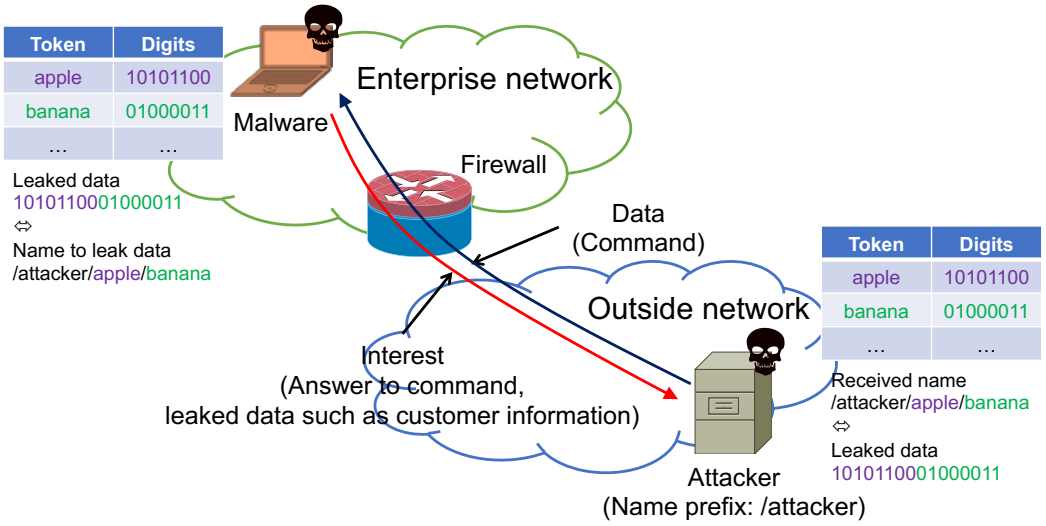


Figure 3.5: Information leakage attack through steganography-embedded Interest name.

al. [74] propose English Shellcode to encrypt and hide information, which transforms Shellcode into the one similar to English prose. To create the steganography-embedded Interest name, as shown in Fig. 3.5, the attacker and malware have to share the same table for steganography, which includes each token (strings) and the corresponding digits. Note that, in order to describe how to generate and utilize the steganography embedded Interest name, Fig. 3.5 adopts the Interest/Data method, but the description is also available in the one-way Interest method. Assuming that the malware knows the attacker's name prefix "`/attacker`" and the name prefix is routable from the malware, the malware converts the leaked data "1010110001000011" to the Interest name "`/attacker/apple/banana`" using the table of steganography. After malware's sending the Interest including the name to leak data, the attacker decrypts the name using the shared table and obtains the leaked data "1010110001000011".

Fig. 3.6 illustrates a general framework to transmit leaked data from malware to the attacker. First, the malware compresses the collected data, for example, in a zip format. For the one-way Interest method, the malware further encodes the compressed data with erasure encoding. To bypass a firewall and perform information leakage through an Interest name, the malware further encodes the output data with token encoding for steganography. Then, it creates and adds a malicious name to leak the data into an Interest.

3.3 Discussion

In the current specification about an Interest, which Section 2.5.1 introduced, in the Interest, available elements other than a name are CanBePrefix, MustBeFresh, ForwardingHint, Nonce, InterestLifetime, HopLimit, and Parameters. All elements except the name are optional, but as for the Nonce, it is required when an Interest is forwarded over the network links. Thus, this thesis assumes that for malware to send malicious Interests to the attacker, the malware has to create them including at least a name and a Nonce.

In the elements, the CanBePrefix and the MustBeFresh cannot have the value. The InterestLifetime and the HopLimit can include only a non-negative integer and a 1 byte unsigned integer as the value, respectively, so that this restriction makes it hard for the malware to utilize

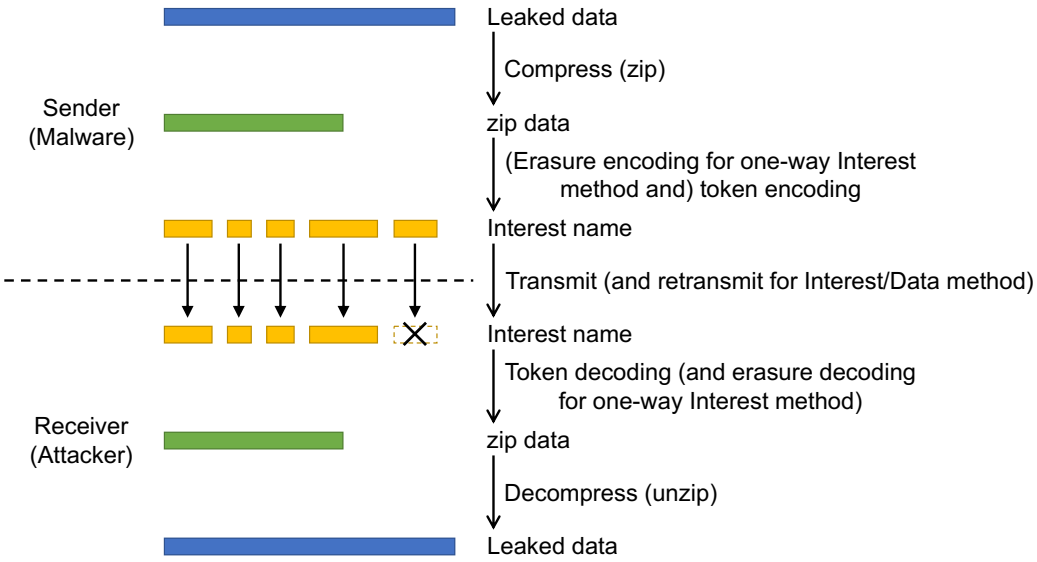


Figure 3.6: Transmission of leaked data from malware to attacker.

these elements to hide the data. On the other hand, the name, ForwardingHint, Nonce, and Parameters can be exploited to perform an information leakage attack through an Interest since except the Nonce (randomly-generated 4 byte string), they are of variable length. However, except the name, they are optional, though the Nonce should be needed in practice, as mentioned previously. Therefore, this thesis does not take the ForwardingHint and the Parameters into account to perform an information leakage attack through an Interest.

As for the Nonce, this thesis utilizes it to express a sequence number which is needed for the retransmissions of some missing Interests in the Interest/Data method. Basically, the Nonce should be randomly generated, so that the malware and attacker have to produce random numbers corresponding to the sequential numbers and share them in advance. Otherwise, their malicious activity might be detected because of the Nonce which is a sequential number.

As Section 2.5.3 mentioned, the name may include the SHA-256 digest of the corresponding entire Data packet as the last name component, segmenting, versioning, time stamping, and sequencing, which are optional. The size of the SHA-256 digest is 32 bytes, and therefore, it could be useful to leak the data. However, in the Interest/Data method, after receiving the SHA-256 digest from the malware, the attacker has to create and send the Data whose SHA-256 digest corresponds to the one appended in the Interest. To generate the Data, the attacker has to break one of the properties of the SHA-256: second pre-image resistance, so that it should be extremely difficult to produce the corresponding Data. Thus, the SHA-256 digest cannot be used to leak the data. As for the segmenting, versioning, time stamping, and sequencing, like the InterestLifetime and the HopLimit, they have also a restriction (e.g., a sequential number), and therefore, they are not useful for the attack.

3.4 Summary

This chapter explained an information leakage attack through a Data and through an Interest. As for the one through an Interest, this chapter introduced a one-way Interest method and an Interest/Data method, and in order to compensate for one drawback of the Interest/Data method

(i.e., attacker anonymity), this chapter presented a 1-to-N attack model, which exploits bots. Moreover, this chapter showed a more sophisticated attack to exploit an Interest name than just adding leaked data into a name: generating a steganography-embedded Interest name, which is apparently a legitimate one (in fact, a malicious one to leak data), and described a general framework to transmit leaked data from malware to the attacker. Finally, this chapter discussed an information leakage attack through available elements other than a name in the Interest. From the next chapter, in order to assess the risk of the information leakage, this thesis focuses on an Interest/Data method with a 1-to-1 and a 1-to-N attack model since the method is more efficient than a one-way Interest method, and this thesis assumes that a malicious Interest is composed of the name and the Nonce, and the name does not include the SHA-256 digest, segmenting, versioning, time stamping, and sequencing, according to the discussion in Section 3.3.

Chapter 4

NDN Firewall

Contents

4.1 Introduction

As shown in Section 2.5, NDN adopts a security-by-design approach based on signatures. Producer's signature in the Data packet addresses spoofing and tampering. However, there are still several security attacks such as IFA, information leakage attack (see Chapter 3), and CPA. Since NDN is a "pull"-based architecture, a trigger of these attacks is a malicious Interest. This means that shutting such an Interest out based on some detection methods can eliminate these security threats (i.e., there is no smoke without fire). Moreover, from an enterprise perspective, it is usually prohibited for the employees to access to, for example, social network services or video services by the working regulations in terms of labor productivity as well as information leakage prevention. Thus, the enterprise network operator might want to manage the content access policy from inside as well as from outside of the enterprise network. In order to adapt to the above use cases, this thesis designs and implements an NDN firewall. Note that, assuming that the detection methods against the NDN security threats exist, this thesis proposes firewall functions (e.g., packet filtering) to mitigate them based on the whitelist as well as the blacklist where the detection methods append the rules on demand, with the goal to make the firewall the key countermeasure against traffic issues. To the best of author's knowledge, there are only

a few research works about an ICN firewall [67], but on a deprecated CCN version and with no source code released.

In order to design an NDN firewall, this thesis focuses on two requirements. Firstly, the current NDN Forwarding Daemon (NFD) [14] does not support appending filtering rules to perform Interest packet filtering. Indeed, the NFD can be modified to install a filtering function, but that may cause some performance degradation of the NFD, whose computing resources should be solely dedicated to forwarding packets, and moreover, this function is not mandatory for all routers. Thus, *(i) designing an NDN firewall independent from the NFD* is required. Secondly, unlike IP addresses in the Internet, names or name prefixes have variable length in NDN. Assuming that an NDN firewall performs a filtering function based on the names or name prefixes, looking for them in the whitelist and blacklist is a serious bottleneck that can degrade the performances. To solve this problem, *(ii) performing a fast lookup of the names or name prefixes in the lists* is required.

To satisfy the two above requirements, the proposed NDN firewall is completely decoupled from NFD and installs a cuckoo filter [15] which is one of the probabilistic filters. The NDN firewall performs Interest packet filtering based on the names or name prefixes in the lists that can be updated on the fly. The throughput degradation with the firewall is only from 0.912% to 2.34%, which will be acceptable in an enterprise network.

The remainder of this chapter is organized as follows. Section 4.2 describes a taxonomy of an IP and an NDN firewall, a firewall use case, and an architecture and implementation of the proposed NDN firewall. Section 4.3 explains the configuration of the firewall. Section 4.4 presents experiments of the firewall, and Section 4.5 discusses the current firewall implementation. Finally, Section 4.6 summarizes this chapter.

4.2 Proposal of NDN Firewall

4.2.1 Taxonomy of IP and NDN Firewall

In the Internet, a communication channel is based on an end-to-end principle. In the recent years, following this principle, users rely on an end-to-end encryption such as TLS, which makes it difficult for network operators to apply their content access policies to the encrypted traffic once the connection is established. A communication paradigm of NDN is different from that of the Internet and an NDN Interest name can always be evaluated by middleboxes. Table 4.1 presents a taxonomy of an IP firewall and an NDN firewall. Although the end-to-end encryption can be applied even in NDN, NDN loses its advantages such as cache [75] in this case. Thus, this thesis does not discuss the end-to-end encryption in NDN. Unlike the Internet, NDN has no consumer identification corresponding to the client IP address, and performs consumer anonymization by default. However, in the case that a consumer connects to an NDN firewall directly, for example, in an enterprise network, the consumer can be tracked by the consumer MAC. The signature in the signed Interest [76] can also identify the consumer, but it is optional. As for the content name, in the Internet, as explained in Section 2.3, a URL is transferred via the HTTPS channel, so the URL is encrypted. Therefore, the encrypted URL is incomprehensible from the IP firewall while, in NDN, the name is readable and understandable.

4.2.2 Use Case

Fig. 4.1 shows two use cases of the NDN firewall against different attacks. Roughly, the attacks can be classified into two types: an attack from outside and one from inside. Fig. 4.1 assumes

Table 4.1: Taxonomy of IP and NDN firewall

Feature	IP firewall	NDN firewall
Client/consumer identification	Client IP, client certificate†	Consumer MAC††, signature†††
Server/producer identification	Server IP, server certificate†	Name prefix, signature
Content name	Encrypted URL	Name

† Optional (TLS handshake).
†† In the case that a consumer connects to an NDN firewall directly, for example, in an enterprise network.
††† Optional (signed Interest).

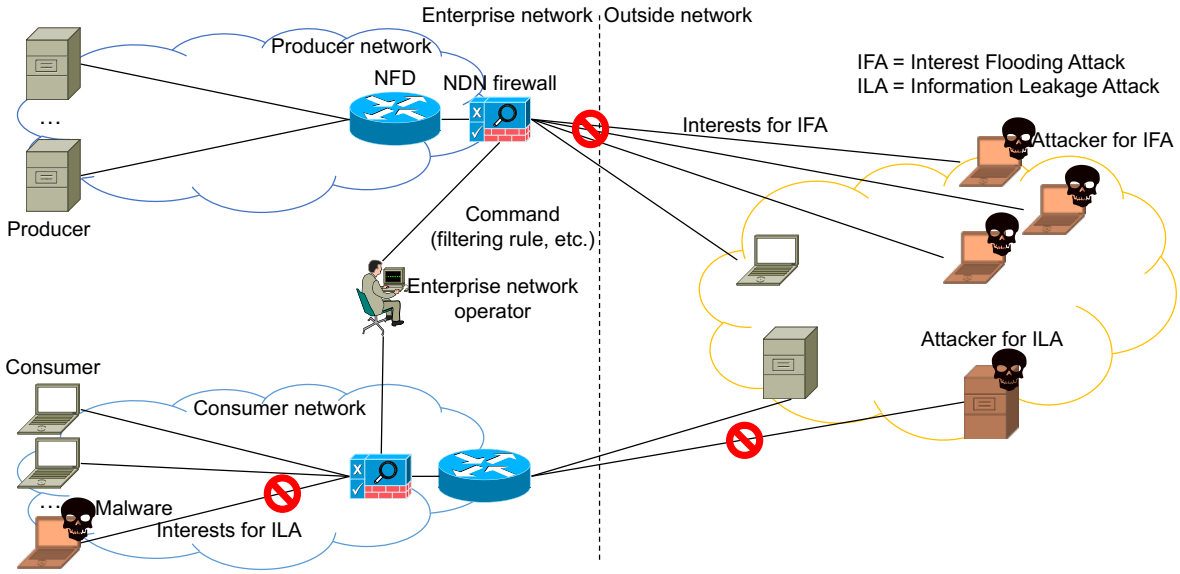


Figure 4.1: High level idea of NDN firewall use case.

that the enterprise network has the NDN firewall to protect itself from these attacks. As for an attack from outside, one of the attacks is IFA. In order to protect the network against the IFA, the NDN firewall (located in the producer network in Fig. 4.1) can drop the malicious Interests using the whitelist before forwarding them to the network. As for an attack from inside, Chapter 2 pointed out vulnerability of an Interest name which is used to perform an information leakage attack explained in Chapter 3. Once the information leakage pattern is detected, the firewall (located in the consumer network in Fig. 4.1) can filter the malicious outgoing Interests.

Moreover, an NDN firewall is useful for not only preventing these attacks but also performing access control with the whitelist and blacklist (Fig. 4.2). For example, when an enterprise network operator lets the consumer utilize all of services of Google except the video service, in order to perform such access control, the operator can append rules "accept `/google`" and "drop `/google/video`" into the NDN firewall. In addition, when the operator does not basically accept an email service of Yahoo but lets the consumer receive the email just from Alice, the operator can build a spam filter as access control by appending rules "drop `/yahoo/mail`" and "accept `/yahoo/mail/alice`" into the firewall. Thus, applying longest name prefix matching to

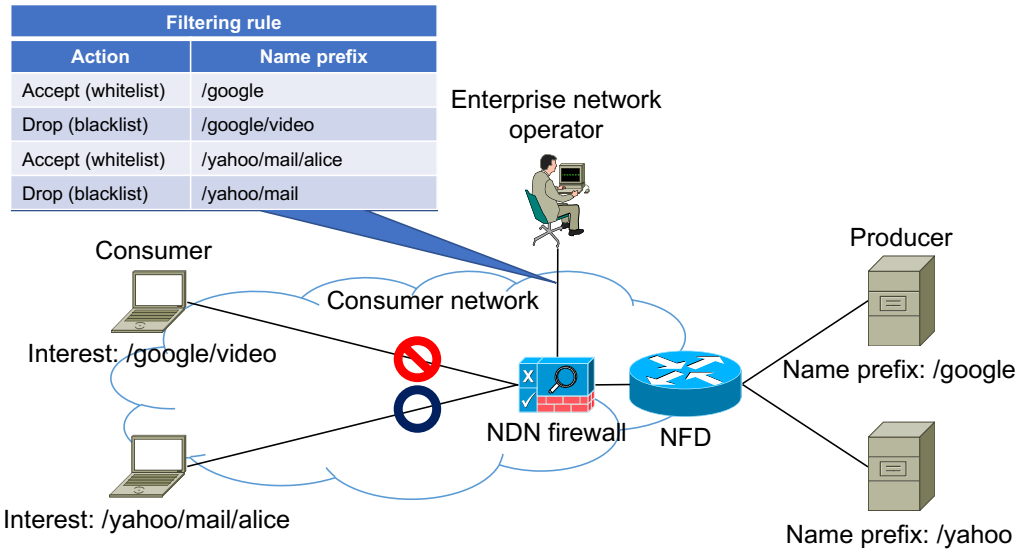


Figure 4.2: Access control with whitelist and blacklist.

the whitelist and blacklist, the NDN firewall can enforce finely tuned rules. When no matching occurs, the NDN firewall performs a default policy: "accept" or "drop" packets by default.

4.2.3 Architecture and Implementation

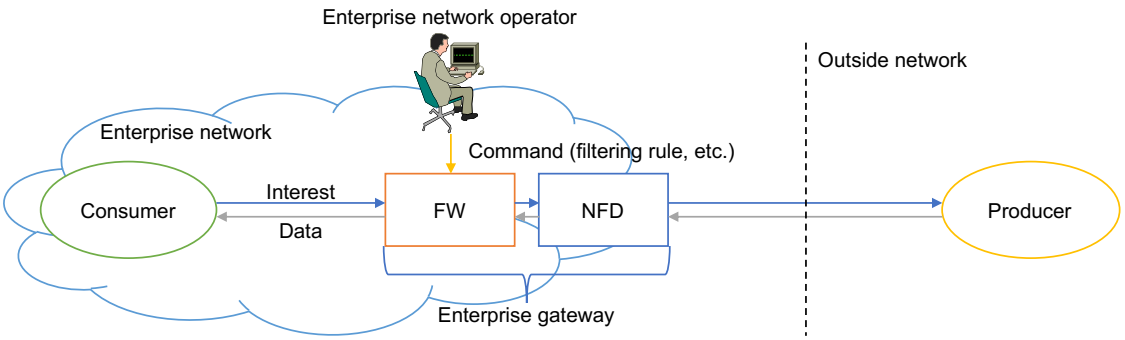


Figure 4.3: Communication channels to NDN firewall.

To satisfy the first requirement introduced in Section 4.1, as shown in Fig. 4.3, the proposed NDN firewall is decoupled from NFD and there is at least one NFD between the firewall and the producer. The consumer's Interest can be sent to the firewall directly or via several NFD nodes. In addition, the NDN firewall is designed to extract Interest packets from only consumer side and Data packets from only producer side, and therefore, as shown in Fig. 4.1, a firewall instance is needed for each side of the network to defend the corresponding side (i.e., consumer-side or producer-side). The main reason behind this design is that features of an attack from outside are different from those of an attack from inside, so that, in terms of NDN firewall usage, it is better to place the firewalls separately because objectives of prevention methods against each attack are different, and so are the rules to be applied.

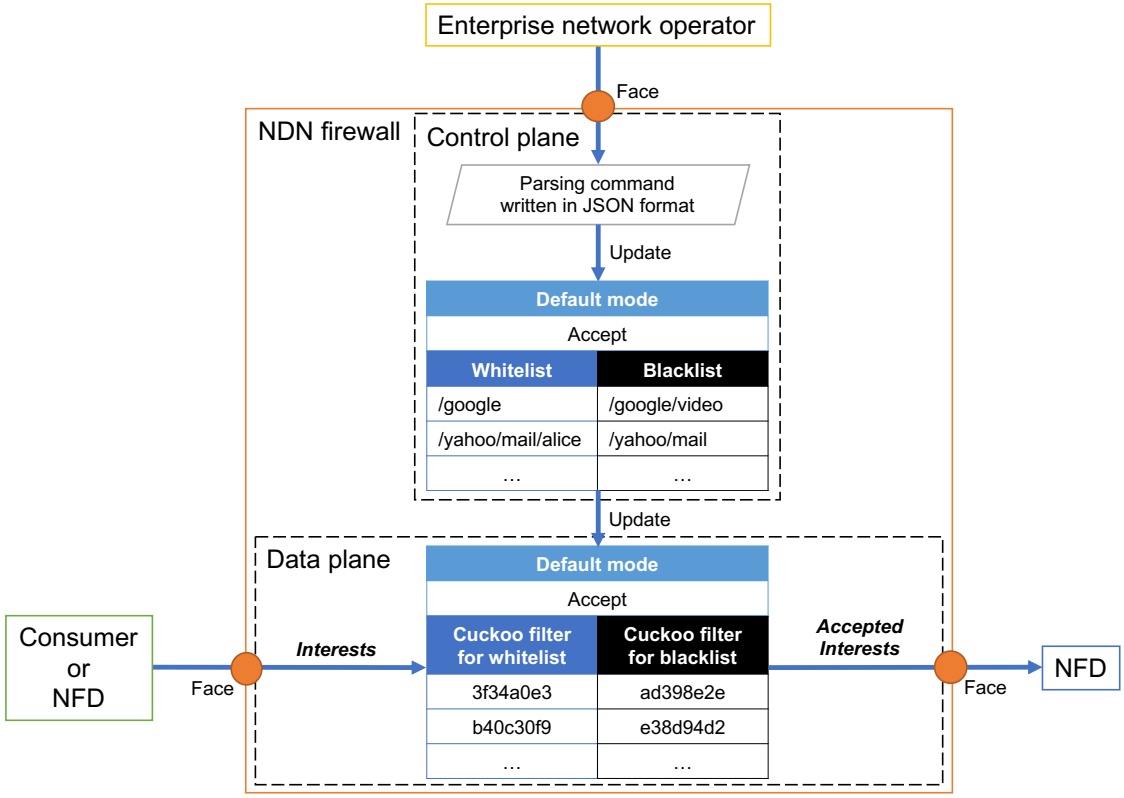


Figure 4.4: Interest packet processing in NDN firewall.

As mentioned previously, looking up names or name prefixes can be a serious bottleneck for the performance such as throughput. To satisfy the second requirement introduced in Section 4.1 and ameliorate the performance, this thesis utilizes cuckoo filter [15] which is one of the probabilistic filters such as bloom filter [77]. While achieving higher performance than the bloom filter, the cuckoo filter permits deletion of items, which the bloom filter does not. In an NDN firewall, it is possible for the operator to append and delete rules of the firewall often, so that this thesis adopts the cuckoo filter. However, in compensation for the high performance, the cuckoo filter might face false positive, and this point is discussed in Section 4.5.

Fig. 4.4 describes the details of the proposed NDN firewall, and especially focuses on how to deal with the Interests sent by the consumer. The NDN firewall first obtains the ingress traffic from the consumer or NFD and extracts the Interests from the stream. Then, the firewall checks if the names or name prefixes of these Interests are listed in the whitelist or blacklist. In this process, based on a slash character ("/") as a delimiter, the firewall has to obtain some name prefixes (and also the name in some cases) in each of the Interest names. In the current implementation, the number of name prefixes which should be extracted from a name is at most $\min(N_{/}, \max(\max(N^{w}{}_{/}), \max(N^{b}{}_{/})))$, where $N_{/}$ is the number of slash characters in the name and $\max(N^{w}{}_{/})$ and $\max(N^{b}{}_{/})$ are the maximum number of slash characters in all the name prefixes in the whitelist and the one in the blacklist, respectively. For example, when the firewall receives a name "`/yahoo/mail/alice/DOCTOR`" and "`/yahoo/mail/alice`" and "`/yahoo/mail`" are inserted into the whitelist and into the blacklist, respectively, the number of extracted name prefixes is at most 3 ($\min(4, \max(3, 2)) = 3$), and in order to perform longest name prefix matching, one by one, the firewall obtains the name prefix from right to left and checks whether

it is in either the whitelist or the blacklist (i.e., "`/yahoo/mail/alice`", "`/yahoo/mail`", and "`/yahoo`" from "`/yahoo/mail/alice/DOCTOR`"). Then, the firewall confirms that "`/yahoo/mail/alice`" is in the whitelist and accepts the Interest. If the names or name prefixes are listed in neither the whitelist nor blacklist, the firewall checks the default mode. If the mode is "accept", the Interests are accepted, and if the mode is "drop", they are dropped. After extracting the accepted Interests, the firewall generates the egress traffic with the Interests and forwards them to the connected NFD. Finally, the accepted Interests retrieve the Data packets corresponding to them.

As shown in Fig. 4.4, from the control plane, the proposed NDN firewall reads online commands written in JSON format to update the mode and the lists dynamically. From operator's point of view, it is also needed to check the state of the mode and the current lists in the firewall. Since the cuckoo filter has the names or name prefixes kept as hash values, the firewall must have a whitelist and blacklist that maintain the names or name prefixes as plain-text, which correspond to these hash values.

4.3 NDN Firewall Management

This section explains management of the NDN firewall. An NDN firewall launch command is used once in order to activate the NDN firewall. On the other hand, after the activation, an NDN firewall online command is available to update rules on the fly.

4.3.1 NDN Firewall Launch Command

The proposed NDN firewall program is called `ndnfirewall`, and it can be run in the following way:

```
ndnfirewall [-m mode] [-w #_of_items] [-b #_of_items] [-lp local_port_#]
[-lpc local_port_#_for_command] [-ra remote_address] [-rp remote_port_#]
[-h help]
```

where:

- `-m` specifies the firewall default mode: accept or drop.
- `-w` configures the capacity of total items in the whitelist.
- `-b` configures the capacity of total items in the blacklist.
- `-lp` indicates the interface of the firewall (the local port number), which should be used by a consumer or NFD in order to connect to the firewall.
- `-lpc` indicates the interface of the firewall (the local port number), which should be used to insert the NDN firewall online command.
- `-ra` indicates the interface of the remote NFD (the remote IP address), which should be used by the NDN firewall in order to connect to the remote NFD.
- `-rp` indicates the interface of the remote NFD (the remote port number), which should be used by the NDN firewall in order to connect to the remote NFD.
- `-h` explains the NDN firewall usage.

As for the firewall mode, it can be changed in real time using the NDN firewall online command written in Section 4.3.2. As for the other parameters, they cannot be changed after being run.

4.3.2 NDN Firewall Online Command

The proposed NDN firewall online command is written in JavaScript Object Notation (JSON) format. This subsection shows one simplified example of the online command below.

```
{
  "get": {
    "mode": [],
    "rules": [
      "white",
      "black"
    ]
  },
  "post": {
    "mode": [
      "accept",
      "drop"
    ],
    "append-accept": [
      "/example1",
      "/example2"
    ],
    "append-drop": [
      "/example3",
      "/example4"
    ],
    "delete-accept": [
      "/example1",
      "/example2"
    ],
    "delete-drop": [
      "/example3",
      "/example4"
    ]
  }
}
```

The online command has roughly two kinds of name/value pairs whose names are `get` and `post`. The value of `get` is one object which can support two kinds of pairs whose names are `mode` and `rules`. To get the current mode, the value of `mode` has to be an empty array, and then an NDN firewall returns either of a mode which basically accepts all packets or a mode which basically drops all packets. The value of `rules` has to be an array including `white` or `black`, and after receiving this pair, the NDN firewall returns the rules which have already been in the whitelist or the blacklist.

The value of `post` is also one object which can support five kinds of pairs whose names are `mode`, `append-accept`, `append-drop`, `delete-accept`, and `delete-drop`. The value of `mode` for `post` has to be an array including `accept` or `drop`, and after receiving the pair, the NDN firewall changes the current mode to the specified one. Each value of `append-accept`, `append-drop`, `delete-accept`, and `delete-drop` also has to be an array including name prefixes, and after receiving each of the pairs, the NDN firewall appends or deletes rules which accept or drop

Interests based on name prefixes in the whitelist or the blacklist. If the online command is syntactically wrong, the NDN firewall rejects it.

4.4 Experiments

This section evaluates the proposed NDN firewall in terms of throughput. For the sake of reproducibility, the code is made available for the community under GNU General Public License (GPL) at [78].

4.4.1 Experimental Setup

To perform the Proof of Concept (PoC), this thesis utilizes IP network to transport NDN packets using TCP and the NDN firewall command using UDP. More precisely, this thesis uses TCP for the NDN Data plane and UDP for management operations between the enterprise network operator and the NDN firewall. The NDN firewall is written in C++ and utilizes ndn-cxx version 0.6.1 [79]. This experiment is conducted by a machine with Intel Xeon processor W3565 (4c@3.20GHz) and 8GB RAM, which operates Ubuntu 16.04.

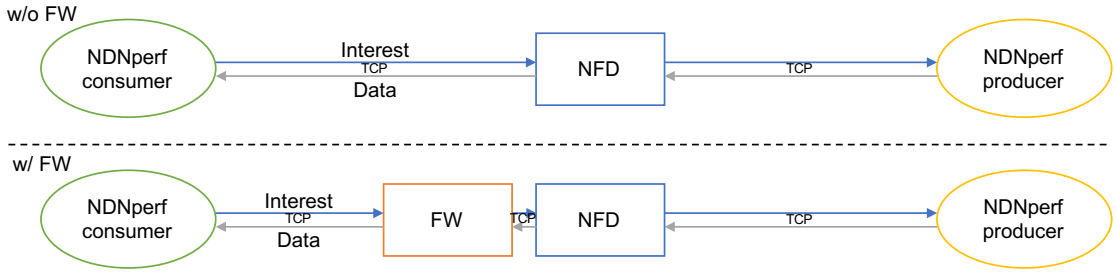


Figure 4.5: Two topologies (*w/o FW* and *w/ FW*).

To perform throughput evaluation, this thesis instantiates two topologies shown in Fig. 4.5: a topology *w/o FW* and one *w/ FW*. Using these topologies, this thesis investigates how much throughput is degraded by adding the proposed firewall. As for the cuckoo filter, this thesis sets the fingerprint size for each item in the whitelist and the blacklist to 32 bits (to reduce false positive rate, longer fingerprints, which is a bit string derived from the item using a hash function, are required (see [15])) and the capacity of total items in the whitelist as well as the blacklist is configured to 2,000,000 (i.e., the whole capacity is 4,000,000). As for the default mode of the firewall, this thesis sets the mode to "accept".

This thesis creates names and each $N_{/}$ is larger than the $\max(\max({N^w}_{/}), \max({N^b}_{/}))$, so that, according to Section 4.2.3, the number of name prefixes extracted by the firewall is at most the $\max(\max({N^w}_{/}), \max({N^b}_{/}))$. The firewall knows the $\max({N^w}_{/})$ and $\max({N^b}_{/})$, and therefore, this thesis can evaluate the firewall performance using the fixed number of extracted name prefixes. Assuming that statistics of NDN names can be predicted from those of the current URLs in the Internet, 95th percentile of $N_{/}$ is 5 according to the URL statistical analysis shown in Section 5.4.2. Considering the above value, this thesis prepares four scenarios for the experiments (Fig. 4.6); in scenario n ($n = 1, 2, 3, 4$), adding 1,000,000 name prefixes where the $\max({N^w}_{/})$ is n and 1,000,000 name prefixes where the $\max({N^b}_{/})$ is n into each of the whitelist and the blacklist (i.e., the number of total name prefixes used in each scenario is 2,000,000). In this case, when one name is in neither the whitelist nor blacklist, in order to inspect the name,

the firewall checks these lists 2, 4, 6, and 8 times in each scenario, respectively. Note that since the cuckoo filter is a probabilistic filter, the number of name prefixes registered in the lists does not affect on name lookup performance (i.e., the computation complexity is $O(1)$).

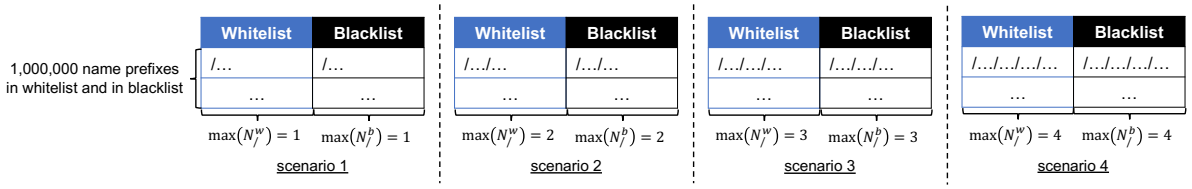


Figure 4.6: Four scenarios for experiments.

To produce NDN traffic, this thesis uses NDNperf [80]. As for the setting of the NDNperf consumer, the Interest packet window size is set to 64 packets and the name is composed of 5 name components separated by 5 slashes (Fig. 4.7). The first 4 name components are fixed, while the 5th component is an incremental sequence number, and the NDNperf consumer generates the name whose name prefixes are in neither the whitelist nor blacklist. As for the setting of the NDNperf producer, the number of the threads to sign Data packets using a SHA256 hash is set to 2 and the payload size of the Data is set to 8192 bytes.

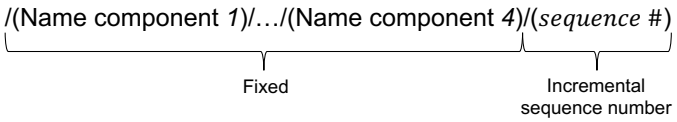


Figure 4.7: Name generated by NDNperf consumer.

This thesis measures the throughput at the NDNperf consumer, and in order to evaluate the throughput, this thesis ignores a certain period of time from the start time of running the NDNperf consumer and producer (40 sec, in the experiments) as the transient state. After that, this thesis performs the measurement for 40 sec. The NDNperf consumer reports the throughput in every 2 sec.

4.4.2 Results

Fig. 4.8 shows the averaged throughput and the standard deviation. Compared to the case that the measurement is performed in the topology *w/o FW*, the throughput degradation by adding the firewall is only from 0.912% to 2.34% in each scenario, which will be acceptable in an enterprise network. In addition, the number of times for the firewall to check the lists has no significant impact on the measured performances despite the slight decrease seen in scenario 1 being within the error margin of real experiments.

4.5 Discussion

The proposed NDN firewall applies the cuckoo filter to the whitelist and blacklist of the NDN firewall, so that the firewall might face false positive while performing longest name prefix matching. In order to inspect one Interest name, several name lookup processes in both of

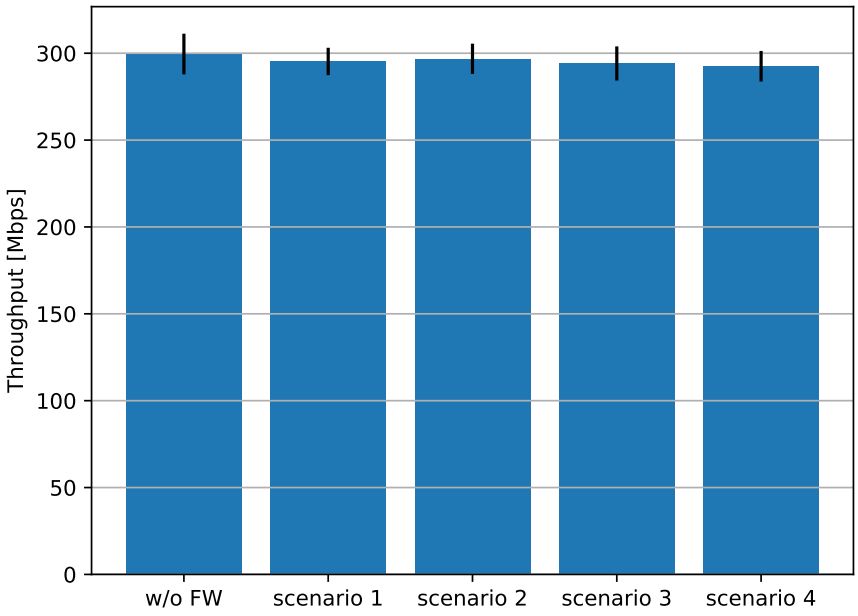


Figure 4.8: Throughput at NDNperf consumer.

the lists should be needed, which might increase the false positive rate. As Section 4.2.3 mentioned, the number of name prefixes which should be extracted from a name is at most $\min(N_{/}, \max(\max(N^{w}{}_{/}), \max(N^{b}{}_{/})))$. In other words, the number of times N_c to check the lists is:

$$N_c \leq \begin{cases} 2 \times N_{/}, & \text{if both } \max(N^{w}{}_{/}) \text{ and } \max(N^{b}{}_{/}) \geq N_{/} \\ \max(N^{w}{}_{/}) + N_{/}, & \text{if } \max(N^{b}{}_{/}) \geq N_{/} > \max(N^{w}{}_{/}) \\ N_{/} + \max(N^{b}{}_{/}), & \text{if } \max(N^{w}{}_{/}) \geq N_{/} > \max(N^{b}{}_{/}) \\ \max(N^{w}{}_{/}) + \max(N^{b}{}_{/}). & \text{if both } \max(N^{w}{}_{/}) \text{ and } \max(N^{b}{}_{/}) < N_{/} \end{cases}$$

When the $N_{/}$ is very large, the upper limit of N_c can be reduced by setting a small value to $\max(N^{w}{}_{/})$ and $\max(N^{b}{}_{/})$. Finally, the false positive rate to check one name is $1 - (1 - \epsilon)^{N_c}$, where ϵ is the false positive rate which the cuckoo filter encounters when it checks one item. According to the URL statistical analysis shown in Section 5.4.2, this thesis believes that the N_c will not be very large (in the experiments, the N_c is at most 8 in scenario 4), so that the false positive rate to inspect a name will not become large. When smaller values of the false positive rate is needed, the firewall can set longer fingerprints for each item. As for the relationship between the false positive rate and the fingerprint size, see [15].

The current firewall implementation does not support functions to process a Data packet, but the firewall can easily implement them. One of the practical use cases about Data packet processing is to verify producer's signature in the Data. To mitigate CPA, the simplest countermeasure is to apply signature verification to every Data. However, the signature verification overhead is generally quite heavy, so that it is normally optional for intermediate NDN nodes to check the signature. Instead, the firewall can verify the signature only when needed. The firewall can be easily extended to support the certificate or the public key upload using a new

online command and verify the signature with the public key.

Finally, let this thesis revisit two requirements: *(i) designing an NDN firewall independent from NFD* and *(ii) performing a fast lookup of the names or name prefixes in the whitelist as well as the blacklist.*

Although Goergen et al. [67] modify CCN daemon directly to implement firewall functions, this thesis designs an NDN firewall independent from NFD, so that the NDN firewall can work as a Virtual Network Function (VNF). Exploiting Network Function Virtualization (NFV) orchestrator, a firewall user can easily allocate and control it according to the demand. In addition, when the orchestrator fully monitors the network traffic, it might build sophisticated detection methods against NDN security threats and send appropriate rules to the firewall in order to mitigate the threats.

The current NDN firewall relies on a cuckoo filter to check the names or name prefixes in the whitelist and blacklist quickly. To find the names or name prefixes from the lists, longest name prefix matching should be carried out since the NDN firewall supports access control such as a spam filter. Performing the fast name lookup for the longest name prefix matching is a challenging work to improve FIB performance, and there are a lot of research efforts [81, 82, 83, 84, 85, 86, 87, 88]. The proposed fast name lookup should be useful also in the NDN firewall, so that this thesis does not pursue proposing new scheme of the fast name lookup, which is out of scope in this thesis.

4.6 Summary

This chapter designed an NDN firewall to mitigate NDN security threats based on a whitelist as well as a blacklist where external detection modules append the rules in real time. The proposed NDN firewall performed Interest packet filtering based on the names or name prefixes in the lists and dynamic NDN firewall configuration by the NDN firewall online command. While providing these functions, the firewall showed high performance. According to the experiments, the throughput degradation with the firewall was only from 0.912% to 2.34%, which will be acceptable in an enterprise network.

From the next chapter, this thesis proposes a name and a flow filter against an information leakage attack through an Interest, which can be installed in the developed firewall.

Chapter 5

NDN Name Filter

Contents

5.1 Introduction

Chapter 3 investigated an information leakage attack through a Data and through an Interest, deeply. Chapter 5 and Chapter 7 propose some countermeasures against them, and especially, this chapter focuses on an NDN name filter.

As introduced in Section 2.5, NDN is a "pull"-based architecture, so that a producer cannot send the Data unless the producer receives the corresponding Interest from the consumer. Thus, as for the attack through a Data shown in Section 3.2.1, an enterprise firewall can easily solve it by carefully inspecting a Data to publish and producing the Data instead of the inside employee in the enterprise network, which means that the firewall can control the Interests from the outside network using the whitelist-based name filter. As for the attack through an Interest shown in Section 3.2.2, however, the firewall cannot manage a naming policy on the outside content and NDN forwarding nodes do not verify if the name really exists or not. That causes a risk of information leakage through an Interest since the malware can hide information such as customer information in the Interest name and send it toward the outside attacker. The malware can pretend to access outside content, so that it is quite difficult for the firewall to detect the information leakage attack.

Against the information leakage attack through an Interest, this thesis first proposes a name filter using search engine information. When the filter receives an Interest, it performs a request to a search engine in order to check whether the Interest name is indexed by the search engine or not. This filter judges the indexed name as legitimate since the malicious name created by the malware is not published anywhere and therefore cannot be indexed. However, this filter cannot predict if the unindexed names are legitimate or not because the unindexed names are not always malicious (e.g., unindexed names such as newly generated or password-protected content ones should be legitimate). To overcome the limitation of the filter, this thesis proposes a name filter using an isolation forest introduced in Section 2.8 since there are not enough malicious samples.

This thesis evaluates the performances of the proposed name filters by collecting URLs from the data repository provided by Common Crawl [89]. This thesis shows that the proposed name filters can drastically choke the information leakage throughput and malware has to send 137 times more Interest packets to leak information than without using the filters.

The remainder of this chapter is organized as follows. Section 5.2 explains properties of malware generating malicious NDN names to perform an information leakage attack through an Interest. Against the attack, Section 5.3 proposes a name filter using search engine information and using an isolation forest while this section also proposes a simple name filter against an information leakage attack through a Data. Section 5.4 describes the NDN name dataset corresponding to the collected URL dataset and its statistics. Using the dataset, Section 5.5 evaluates the proposed name filters. Section 5.6 discusses this research and Section 5.7 summarizes this chapter.

5.2 Properties of Malware Generating Malicious NDN Names

When malware creates a malicious name to leak data, there is a naive idea to ameliorate the information leakage throughput: *adding the data into the name as much as possible*. However, the properties of the generated name might not be similar to the ones of a legitimate name. For example, when the leaked data is composed of hexadecimal digits, the name including the leaked data as much as possible might be much longer than the legitimate name and the frequencies of letters and the other printable characters in the malicious name could be different from the ones in the legitimate name. Therefore, possibly it is quite easy to detect the name as malicious.

As Section 3.2.2 mentioned, malware can create a steganography-embedded Interest name as a more sophisticated attack. In this attack, each token corresponds to the digits, and after attacker's gathering several tokens from the malware, they can be decoded and then become the leaked data. Thus, to improve the throughput, an essential key is to *add the tokens into the name as many as possible*. To do so, the malware can exploit the path and the query in the name shown in Fig. 2.10. As for the fragment, in the Internet, the fragment is resolved by a browser and is not sent to the server by the browser [90], so that the fragment cannot be used to add the tokens. When adding the tokens, the malware has to concatenate them with delimiters in order to let the attacker decrypt the steganography-embedded Interest name safely (e.g., assuming that "`/attacker.com/aaa`" is a steganography-embedded Interest name and the token table includes "a", "aa", and "aaa", the leaked data cannot be decrypted uniquely). As for the delimiters between the tokens, this thesis considers a slash character ("/") in path and an equal and an ampersand character ("=", "&") in query. Since the tokens are used, the frequencies of letters and the other printable characters in the path and the query of the created name might follow the ones of a language used in the tokens (e.g., English).

A naive information leakage throughput improvement method by malware will adopt four

properties in Table 5.1. By adding the tokens into the name as many as possible, the number of "/" in path and "=" in query increases (*Property 1*), and therefore, the length of path and query becomes large (*Property 2*). In addition, to improve the throughput, the length of token, which corresponds to the leaked data, should be small (*Property 3*), and by exploiting the tokens, the frequencies of letters and the other printable characters in path and query follows the ones of a language used in the tokens (*Property 4*). The number of equal characters is related to the number of ampersand ones since they are used to construct and concatenate key-value pairs (e.g., key1=value1&key2=value2), and therefore, this thesis does not consider the number of ampersand characters as one of the *Properties*.

Table 5.1: Properties of malware generating steganography-embedded Interest name

Property 1	The number of "/" in path and "=" in query will be large.
Property 2	The length of path and query will be large.
Property 3	The length of token will be small.
Property 4	The frequencies of letters and the other printable characters in path and query will follow ones of a language used in the tokens.

5.3 Proposal of NDN Name Filter

This thesis proposes several NDN name filters against an information leakage attack through a Data and through an Interest introduced in Chapter 3. The proposed name filters are installed in the enterprise firewall.

5.3.1 Name Filter against Information Leakage Attack through Data

As a "pull"-based architecture, even though malware sends the Data toward the firewall before receiving the corresponding Interest, the firewall cannot forward it to the outside network since there is no PIT entry for the Data. Thanks to the big advantage in NDN, an information leakage attack through a Data can be prevented easily by making a name filter based on a whitelist. When the firewall builds the name filter, the firewall has to follow the policies below.

- *Policy 1*: The firewall has to carefully inspect content and decide whether it can be published to the outside network or not.
- *Policy 2*: The firewall has to accept only Interests including whitelisted names from the outside network, otherwise drop the other Interests from the outside network.
- *Policy 3*: The firewall has to publish all the content accepted to be public to the outside network instead of the inside employees.

In this case, all the publicly-accessible content is on the firewall, and therefore, when the employees attempt to publish content to the outside network, they need to ask the firewall about that beforehand. Once the firewall follows these policies and creates the whitelist-based name filter, the name filter can eliminate the information leakage attack through a Data.

5.3.2 Name Filter against Information Leakage Attack through Interest

Considering regular employee behaviour to access outside content from the enterprise network, this thesis first makes the assumptions below.

- An employee accesses an outside content name via a search engine.
- The employee also contacts an outside content name directly which is not indexed by the search engine (e.g., password-protected content).
- By managing a content access policy, the firewall can prohibit accessing unwanted content names and also explicitly define a whitelist with names that can be accessed.

Name Filter Using Search Engine Information

Assuming that a search engine will still exist even in future Internet, the search engine might serve to detect a legitimate name since a malicious name to leak the information from the enterprise network is not indexed by the search engine (the malicious name is created by malware, and therefore, the name is not published anywhere). Thus, this thesis proposes a name filter using search engine information. When the filter receives an Interest, it performs a request to a search engine in order to check whether the Interest name is indexed by the search engine or not. If the name is found by the search engine, the filter considers the Interest as legitimate. It is quite easy to implement this filter, but, as will be explained next, there is a limitation by considering today's Internet use.

According to [91], search engines index only 4% of all content (referred as Surface Web), and the remaining 96% are not indexed by the search engines (referred as Deep Web). In addition, the number of Internet users using search engines decreases, as its rate was 55% in 2014 but only 49% in 2015 [92]. One of the factors about this decrease is the mobile era. Since it is difficult for users to search content with a small-size mobile screen, they prefer accessing content via the other methods such as social networks. However, from enterprise perspectives, it is usually prohibited for employees, who work with company-supplied personal computers, to use social networks by their working regulations in terms of labor productivity as well as information leakage prevention. Following the conventions, the firewall in the enterprise will have to define a content access policy to allow the employees to access outside content using a search engine or an authorized link toward the content.

As most of content come from Deep Web, many of them should be still accessible from employees. For example, accessing content unreachable from a search engine such as newly generated or password-protected content should be legitimate behaviour. Therefore, a naive name filter using search engine information will not be enough accurate to decide if a name is legitimate or not. Indeed, this filter cannot be aware of most content names that are not indexed. Thus, there is a need to add further information to improve the efficiency of the name filter.

Name Filter Using Isolation Forest

To overcome the limitation, this thesis proposes a name filter using an isolation forest since there are not enough malicious samples. Regarding NDN architecture, as it has not been deployed yet, there is neither a legitimate NDN name dataset nor a malicious one. Thus, assuming that an NDN name will be based on URL as Section 2.5.3 stated and the legitimate NDN name dataset is derived from the URL dataset, this thesis extracts 122 features from each of the NDN names

and builds the filter (Table 5.2). This thesis also extracts the frequencies of characters in the NDN name, and for this, this thesis uses the function F that is defined here.

Let $\mathcal{S}$ be the set of all printable characters according to RFC 3986 [24]. Let $\mathcal{L} \subset \mathcal{S}$ be the set of letters (i.e., a ... z, A ... Z), and let $\mathcal{C} = \mathcal{S} \setminus \mathcal{L}$ be the set of the other printable characters that are not letters (i.e., 0123456789~!@#$%&*()-_=+;:',./?[]). This thesis denotes with $\mathcal{N}$ the set of all possible NDN names constructed with characters in $\mathcal{S}$.

This thesis defines the function

$$F(\cdot,\cdot) : \mathcal{N} \times \mathcal{S} \to \mathbb{R},$$
$$F(n,c) = \frac{\#\text{ of occurrences of } c \text{ in } n}{\text{size of } n},$$

which takes as input an NDN name n from $\mathcal{N}$ and a character c from $\mathcal{S}$, and returns the frequency of c in n.

Table 5.2: Features extracted from NDN name

Notation	Feature variable
$N_{/}$	# of "/" in path
$N_{=}$	# of "=" in query
L_{Path}	Length of path
L_{Query}	Length of query
$F(Path, \ell)$	Frequency of the letter $\ell \in \mathcal{L}$ in path (26 dimensions)
$F(Path, c)$	Frequency of the character $c \in \mathcal{C}$ in path (33 dimensions)
$F(Query, \ell)$	Frequency of the letter $\ell \in \mathcal{L}$ in query (26 dimensions)
$F(Query, c)$	Frequency of the character $c \in \mathcal{C}$ in query (33 dimensions)

5.4 NDN Name Dataset and its Statistics

This section presents the NDN name dataset derived from the URL dataset and describes the statistics of the dataset.

5.4.1 NDN Name Dataset

In order to infer properties of names commonly used in the Internet, this thesis collects URLs from a data repository provided by Common Crawl [89]. This thesis first obtains the crawl archive for February 2016, which holds more than 1.73 billion URLs and this thesis extracts unique URLs belonging to eight Top-Level Domains (TLDs): "com", "net", "org" and "info" as generic Top-Level Domains (gTLDs), and "jp", "fr", "uk", and "de" as country code Top-Level Domains (ccTLDs). In this dataset, the number of URLs varies largely for each TLD (million to billion), and therefore, in order to obtain the same number of URLs for each TLD, this thesis extracts two million URLs for each TLD randomly. In other words, this thesis relies on 16 million URLs.

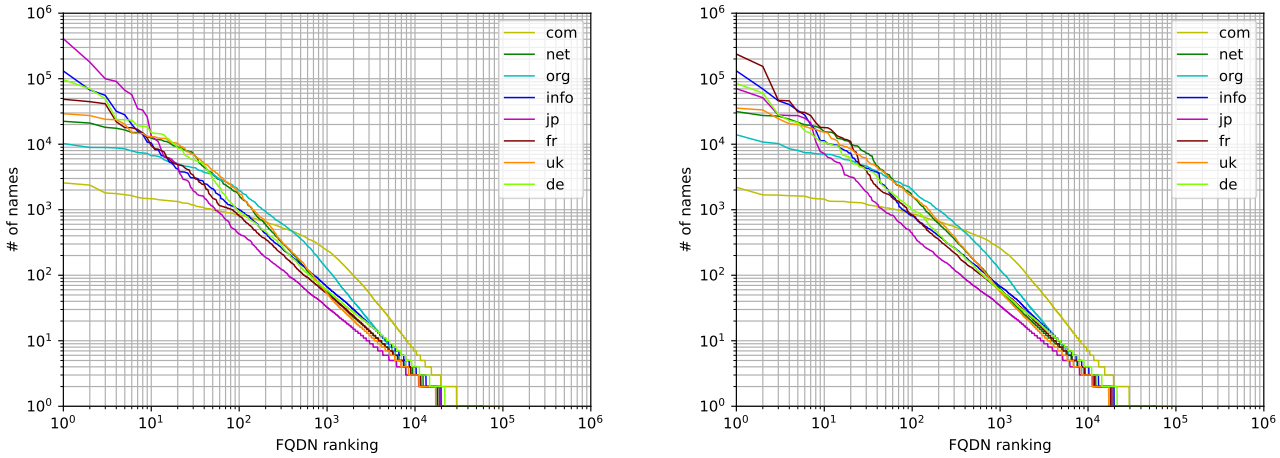
Then, assuming that a protector and an attacker collect a name dataset by each own crawler and these two obtained datasets in which FQDNs are not overlapped at all, this thesis divides the dataset including 16 million URLs into two datasets that include randomly chosen unique

FQDNs: *one for the protector to make a name filter* (see Section 5.5.1) and *the other for the attacker to create a malicious name* (see Section 5.5.1).

Table 5.3 summarizes the name dataset for the protector and for the attacker. Fig. 5.1 shows FQDN ranking versus the number of names for each dataset. As for the definition of the FQDN ranking, an FQDN is ranked by the number of unique names that the FQDN holds in each TLD dataset. Fig. 5.1 indicates that two datasets, i.e., the protector and the attacker dataset, which are not overlapped at all but have similar relationships between FQDN ranking and the number of names.

Table 5.3: Summary of name datasets

	Protector		Attacker	
	# of FQDNs	# of NDN names	# of FQDNs	# of NDN names
com	100,660	986,530	100,294	1,013,470
net	56,342	947,613	56,348	1,052,387
org	48,538	997,139	48,382	1,002,861
info	58,199	969,734	58,141	1,030,266
jp	58,900	1,383,230	58,847	616,770
fr	57,057	751,649	57,074	1,248,351
uk	53,459	1,008,608	53,457	991,392
de	65,109	1,091,233	65,036	908,767



(a) Protector's name dataset.

(b) Attacker's name dataset.

Figure 5.1: FQDN ranking vs. the number of names.

5.4.2 Statistics

This thesis analyzes nine name attributes in the protector dataset and in the attacker name dataset:

1. (the number of slash characters ("/") in path),

2. (the number of equal characters ("=") in query),

3. (the length of path),

4. (the length of name component),

5. (the length of query),

6. (average frequencies of letters in path),

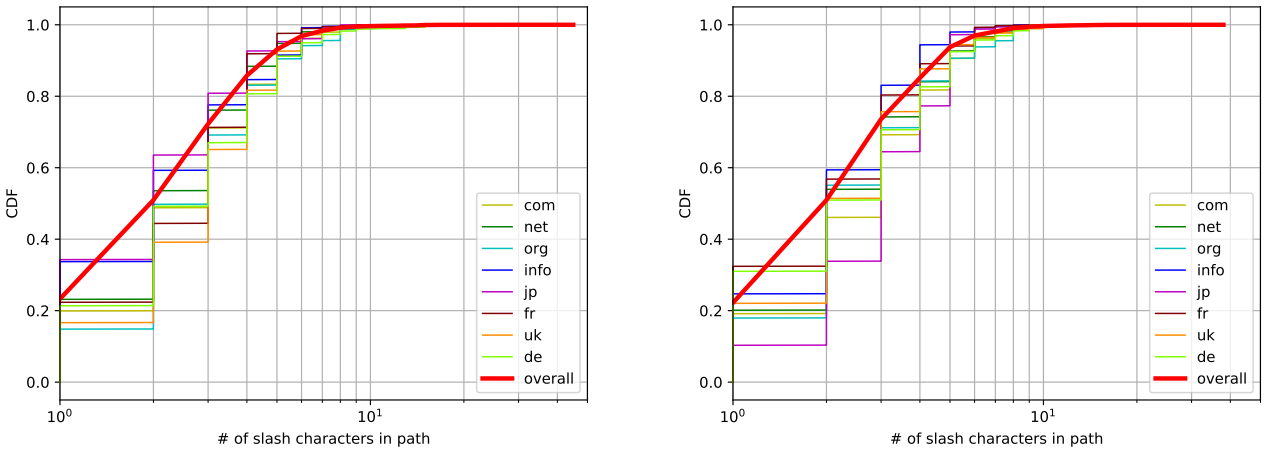
7. (average frequencies of the other printable characters in path),

8. (average frequencies of letters in query), and

9. (average frequencies of the other printable characters in query).

As for some names, no name component exists because some default pages such as index.html are often omitted (e.g., `http://www.doctor-project.org/`). In such a case, this thesis sets the length of name component to zero. To obtain the average frequencies of letters and the other printable characters in path and in query, this thesis first calculates their frequencies from each name, respectively. Then, this thesis sums up the obtained frequencies and averages them within each TLD.

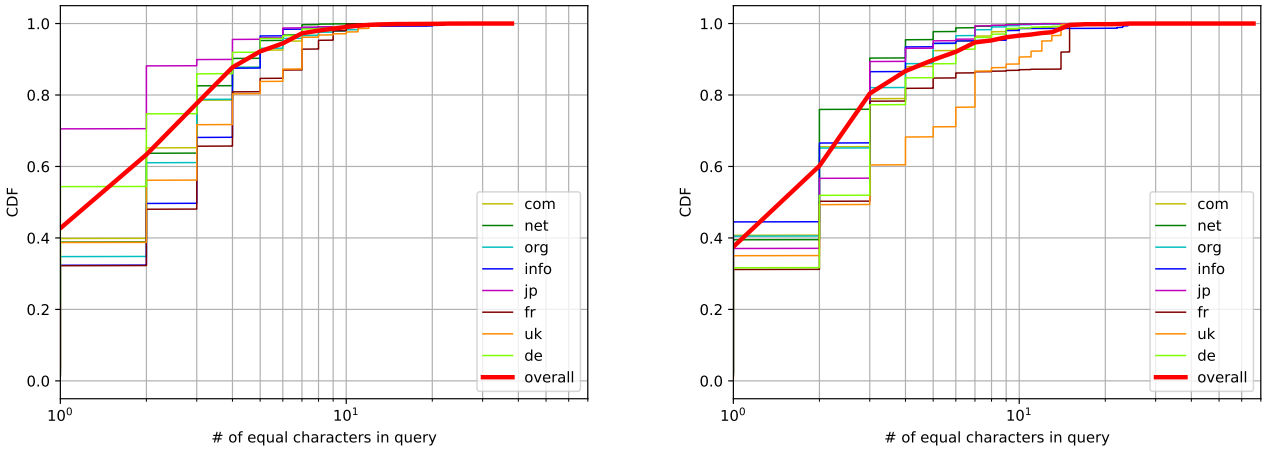
Figs. 5.2, 5.3, 5.4, 5.5, and 5.6 show the Cumulative Distribution Function (CDF) of attribute 1, 2, 3, 4, and 5, respectively, for each TLD. After summing up each CDF for eight TLDs, this thesis divides the sum by eight to obtain the average CDF (i.e., "overall" in legend). In any figures, the CDF of "overall" in the protector's name dataset is similar to the one in the attacker's name dataset while the CDF of each of the TLDs in the protector's name dataset is slightly different from the one in the attacker's name dataset. Table 5.4 shows the computed percentiles of five attributes.



(a) Protector's name dataset.

(b) Attacker's name dataset.

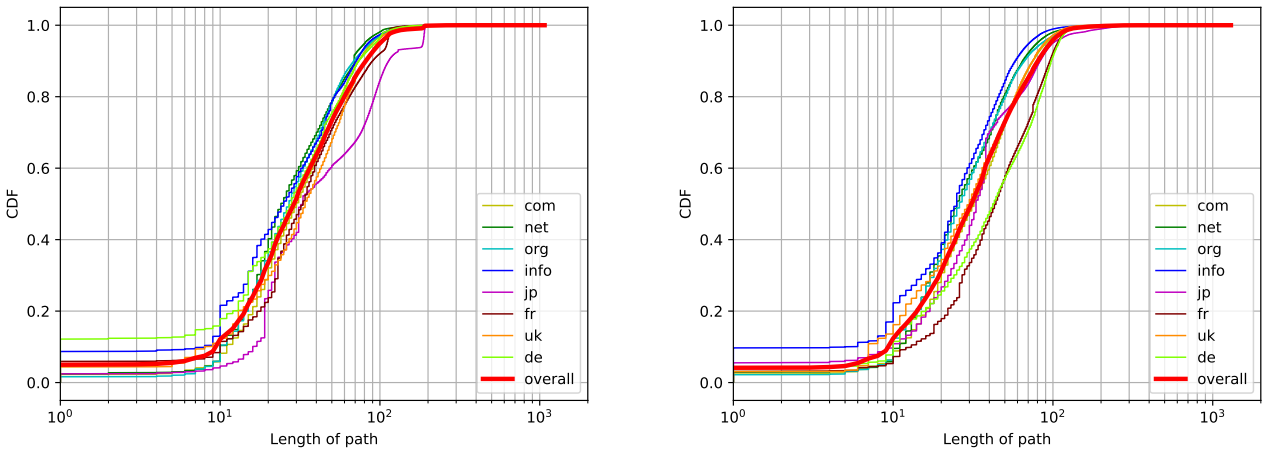
Figure 5.2: CDF of **number of slash characters in path**.



(a) Protector's name dataset.

(b) Attacker's name dataset.

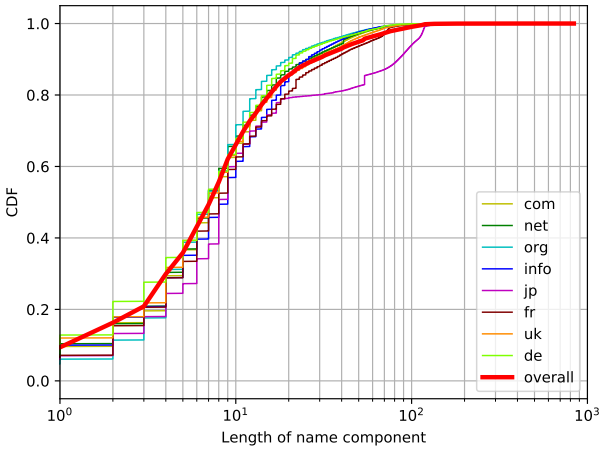
Figure 5.3: CDF of **number of equal characters in query**.



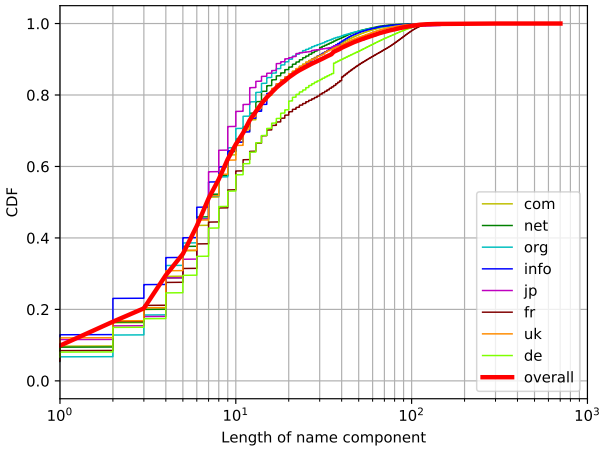
(a) Protector's name dataset.

(b) Attacker's name dataset.

Figure 5.4: CDF of **length of path**.

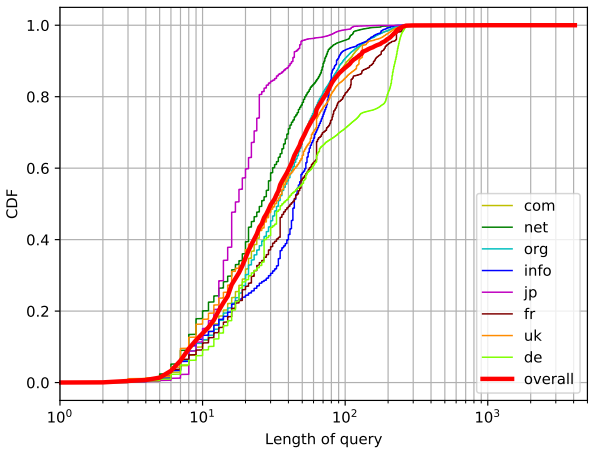


(a) Protector's name dataset.

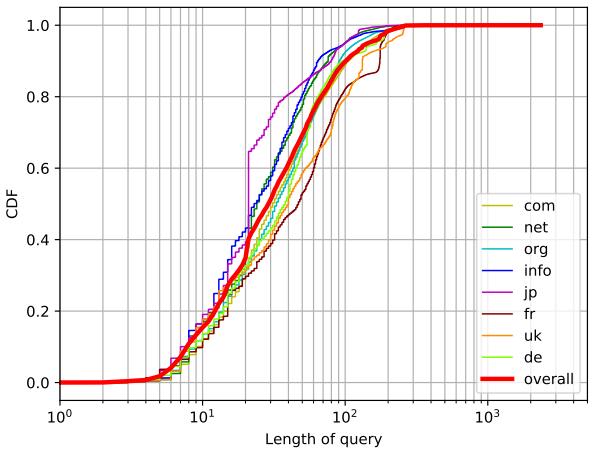


(b) Attacker's name dataset.

Figure 5.5: CDF of **length of name component**.



(a) Protector's name dataset.



(b) Attacker's name dataset.

Figure 5.6: CDF of **length of query**.

Table 5.4: Computed percentiles

(a) gTLDs (com, net, org, info)

Attributes	com						net						org						info					
	Protector			Attacker			Protector			Attacker			Protector			Attacker			Protector			Attacker		
	90%	95%	99%	90%	95%	99%	90%	95%	99%	90%	95%	99%	90%	95%	99%	90%	95%	99%	90%	95%	99%	90%	95%	99%
# of "/" in path	4	5	8	4	5	8	4	5	7	4	5	7	4	6	8	4	6	8	4	5	5	3	4	5
# of "=" in query	4	5	10	4	5	12	3	4	6	2	3	6	4	5	10	4	5	7	4	4	8	3	5	22
Length of path	73	88	127	74	89	135	68	80	109	64	79	124	68	86	128	65	84	130	70	84	120	57	70	101
Length of name component	26	40	69	26	41	69	25	40	71	23	35	62	19	32	66	20	32	66	25	37	64	27	39	64
Length of query	103	134	217	105	142	209	71	87	136	75	99	161	96	134	209	90	118	205	86	130	193	63	99	222

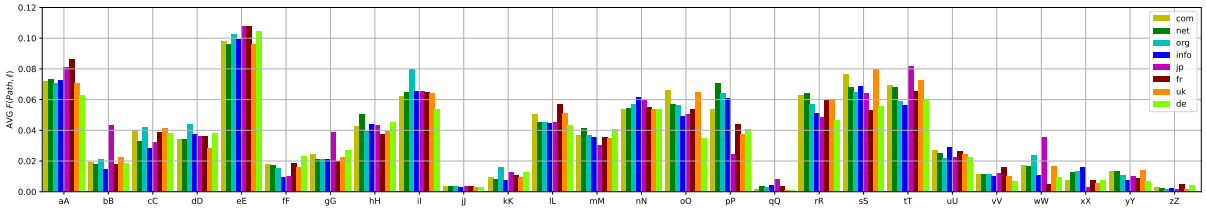
(b) ccTLDs (jp, fr, uk, de)

Attributes	jp						fr						uk						de					
	Protector			Attacker			Protector			Attacker			Protector			Attacker			Protector			Attacker		
	90%	95%	99%	90%	95%	99%	90%	95%	99%	90%	95%	99%	90%	95%	99%	90%	95%	99%	90%	95%	99%	90%	95%	99%
# of "/" in path	3	4	6	4	4	6	3	4	6	4	5	5	4	5	7	4	5	9	4	6	12	4	5	8
# of "=" in query	3	3	7	3	4	6	6	7	9	13	14	14	6	6	11	9	11	13	3	4	8	5	6	11
Length of path	112	185	190	82	99	200	89	110	115	95	106	126	80	95	126	74	88	117	73	92	121	97	108	129
Length of name component	83	104	117	18	48	99	35	57	77	55	82	108	29	44	73	27	43	76	20	33	67	39	64	101
Length of query	42	48	108	79	99	133	160	203	250	174	179	212	124	140	225	131	195	258	224	238	260	95	165	227

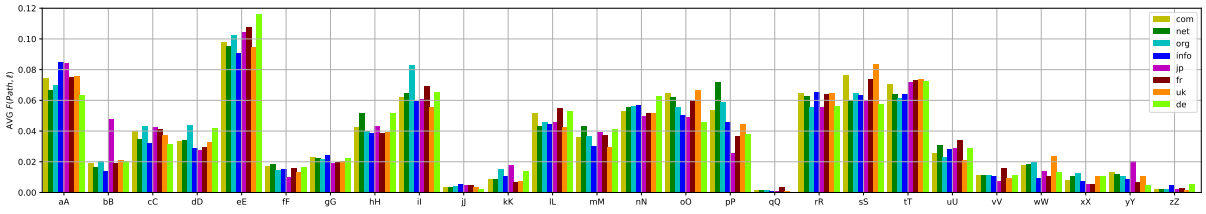
(c) Average (overall)

Attributes	overall					
	Protector			Attacker		
	90%	95%	99%	90%	95%	99%
# of "/" in path	4	5	7	4	5	7
# of "=" in query	4	6	9	5	7	14
Length of path	81	100	164	80	96	129
Length of name component	28	50	97	30	48	91
Length of query	112	178	238	100	144	221

Figs. 5.7, 5.8, 5.9, and 5.10 show average frequencies of letters and of the other printable characters in path and those in query. In any figures, the average frequencies of letters and of the other printable characters from each TLD in the protector's and attacker's name dataset are similar. Moreover, by calculating cosine similarities[1] of average frequencies of letters and of the other printable characters in path and those in query between the protector's and attacker's name dataset, most of the average frequencies of letters and of the other printable characters among the TLDs in the protector's name dataset and in the attacker's name dataset resemble each other (Table 5.5). Specific characters such as "-", "_", ".", and "%" are more often used than the others.

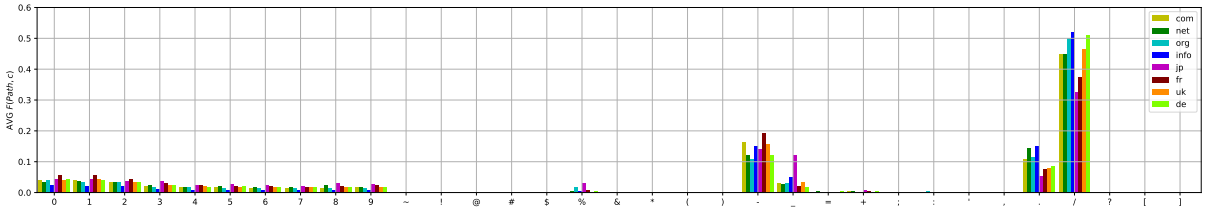


(a) Protector's name dataset.

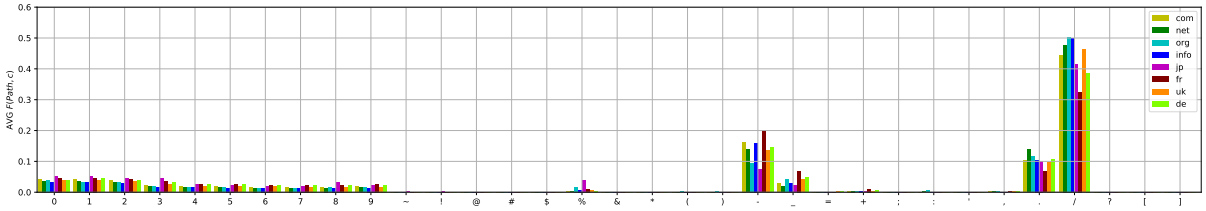


(b) Attacker's name dataset.

Figure 5.7: Average frequencies of **letters in path**.



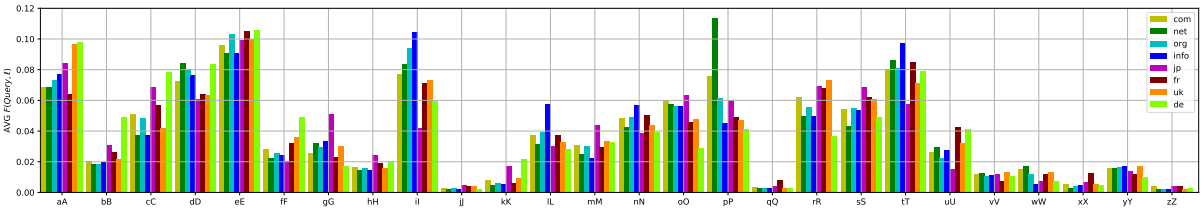
(a) Protector's name dataset.



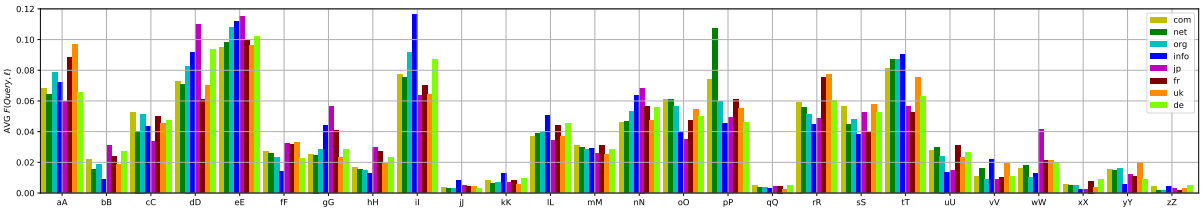
(b) Attacker's name dataset.

Figure 5.8: Average frequencies of **the other printable characters in path**.

[1] $CosineSimilarity = \mathbf{A} \cdot \mathbf{B} / \|\mathbf{A}\| \|\mathbf{B}\|$.



(a) Protector's name dataset.



(b) Attacker's name dataset.

Figure 5.9: Average frequencies of **letters in query**.

Table 5.5: Cosine similarities of average frequencies of letters and of the other printable characters between protector's and attacker's name dataset

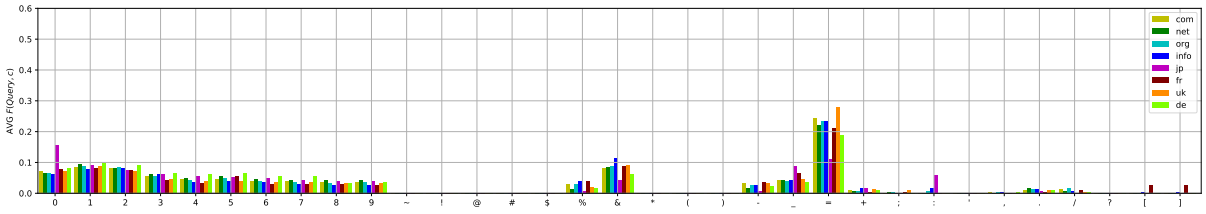
(a) Path

TLD	com	net	org	info	jp	fr	uk	de
Letter	1.00	0.998	0.999	0.989	0.985	0.992	0.996	0.994
Other	1.00	0.999	0.999	0.995	0.937	0.990	0.998	0.983

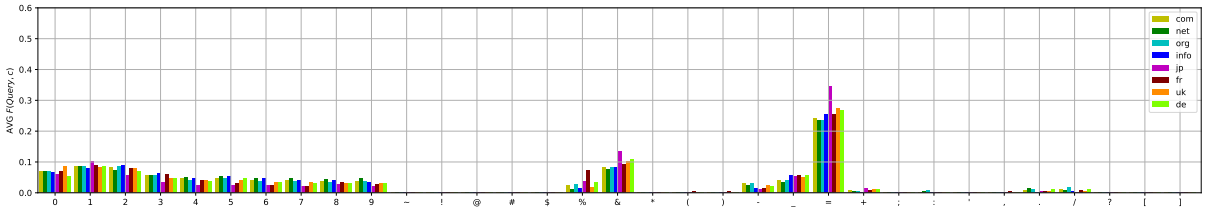
(b) Query

TLD	com	net	org	info	jp	fr	uk	de
Letter	1.00	0.995	0.998	0.981	0.922	0.973	0.994	0.945
Other	1.00	0.997	0.999	0.982	0.705	0.975	0.997	0.943

As explained in the following, the properties of these attributes can be discussed by taking Search Engine Optimization (SEO) into account. SEO is an optimization to improve the rank of the web site in search engines such as Google and let this web site appear in the top level of retrieval results by search engines. One of the methods to improve the rank is to modify the structure of URL. The document about SEO that Google provides [93] describes that URL should be one which is composed of the information easily understood by a crawler constructing a search engine as well as a user seeing the web site. Then, it is convenient for the crawler to obtain information easily and for the user to cite a link, and therefore, this operation improves the rank. Moreover, Moz, which is one of the SEO companies, reports the policies to modify the structure of URL [94]. Concretely, these policies are "to make URL human-readable", "to add the keywords", "to shorten the length", "to make URL suiting for the title of web page", "to remove the stop words such as "and", "or", and "but" considering readability and the length of URL", "to avoid utilizing the unsafe characters written in RFC 1738 [95]", "to decrease the number of name components", "to avoid utilizing the hash value", "to use "-" or "_" to separate words", "to



(a) Protector's name dataset.



(b) Attacker's name dataset.

Figure 5.10: Average frequencies of **the other printable characters in query**.

avoid utilizing the keywords repeatedly", and so on. Therefore, possibly the current URLs are following the above rules to improve the rank, and consequently SEO affects the properties of these attributes.

5.5 Experiments

This section evaluates the performances of the proposed NDN name filters against steganography-embedded Interest names.

5.5.1 Experimental Setup

In general, security exploits can be distinguished into two different types: *a1*) an attacker does not know the countermeasure by the protector and *a2*) an attacker knows the countermeasure but not its parameters. Similarly, countermeasures can be also categorized into two different types: *p1*) a protector does not know the attack method by attacker and *p2*) a protector knows the attack method but not its parameters. This thesis considers a scenario where the protector knows the attack method (i.e., an information leakage attack through an Interest) and prepares the protection method (i.e., *p2*). In order to study the upper limit of the information leakage throughput in the case *p2*, this thesis analyses the risk of the attack in the case that the attacker knows the countermeasure but not its parameters (i.e., *a2*), and this case is of benefit to the attacker.

Protector

Fig. 5.11 explains a flow to check an NDN Interest name using two name filters proposed in Section 5.3.2. The name should be checked by at first the name filter using the search engine information, and when the name is dropped by the first filter, the name filter using the isolation forest, which ic the second filter, inspects the name. Assuming that the name filter using the search engine information does not work effectively since malicious names do not exist in

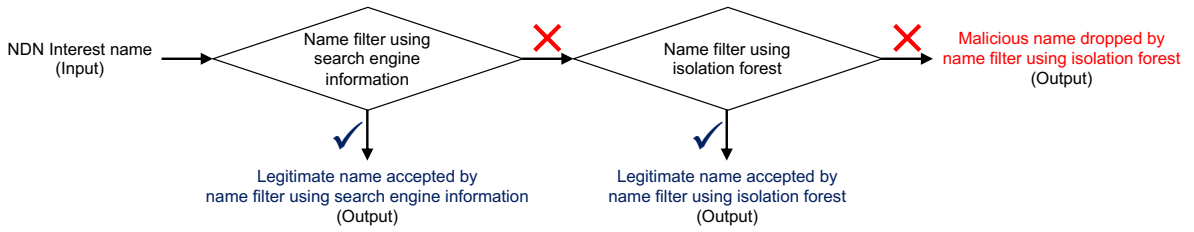


Figure 5.11: Flow to check NDN Interest name using two name filters.

the search engine, this section evaluates the name filter using the isolation forest. As for the evaluation about the name filter using the search engine information, Section 5.6 discusses it.

To build the name filter using the isolation forest, as the training dataset, this thesis uses the legitimate NDN name dataset for the protector, which is summarized in Table 5.3. The dataset is based on eight name datasets separated by eight TLDs. According to the statistics about an NDN name shown in Section 5.4.2, the statistics obtained from each of the datasets are slightly different from each other, and therefore, this thesis prepares eight name filters using the isolation forest. For example, when an Interest carries the name including "com" domain, the name should be checked by the filter derived from the "com" dataset. In addition, to utilize the isolation forest, as a parameter of the protector side, this thesis needs to configure the proportion of outliers R_O^P in the dataset, which is one of the parameters of the isolation forest[2]. This thesis prepares six outlier proportions: 0.01, 0.05, 0.1, 0.2, 0.3, and 0.4. Since the name dataset is considered as legitimate, the outlier proportion can be the false positive rate, which is defined as the ratio of the number of legitimate names identified as malicious to the total number of legitimate names.

Attacker

As mentioned previously, in order to analyze the risk of the information leakage attack, this thesis considers the case that the attacker knows the countermeasure of the protector but not its parameters, which is of benefit to the attacker. Thus, to create malicious names with steganography to leak data, this thesis assumes that the attacker builds his or her own isolation forest-based filter using the NDN name dataset for the attacker, which is shown in Table 5.3, in the same way as the protector's one.

This thesis prepares three PDF files (Y.4001/F.748.2, Y.4412/F.747.8, Y.4413/F.748.5) from latest ITU-T recommendations [99] as leaked data. These PDF files include a variety of text and figures, which are common in technological documents. Specifically, in Y.4001/F.748.2, Y.4412/F.747.8 and Y.4413/F.748.5, there are 6, 4 and 9 figures and 18, 22 and 24 pages, respectively. Then, this thesis compresses and converts these files into a single ZIP file (3.4 MB). Hereafter, this thesis considers how an attacker obtains this file.

Fig. 5.12 describes a flow to create malicious names with steganography to leak data, following the properties summarized in Table 5.1. To ameliorate the information leakage throughput, as Section 5.2 mentioned, an essential key is to *add the tokens into the name as many as possible*. Thus, the main principles to create malicious names are

(i) to extract NDN names classified as the inliers by the isolation forest whose proportion of outliers is set to R_O^A (i.e., 0.01, 0.05, 0.1, 0.2, 0.3, or 0.4) as a parameter of the attacker

[2]This thesis uses scikit-learn [98] with default values for the remaining parameters.

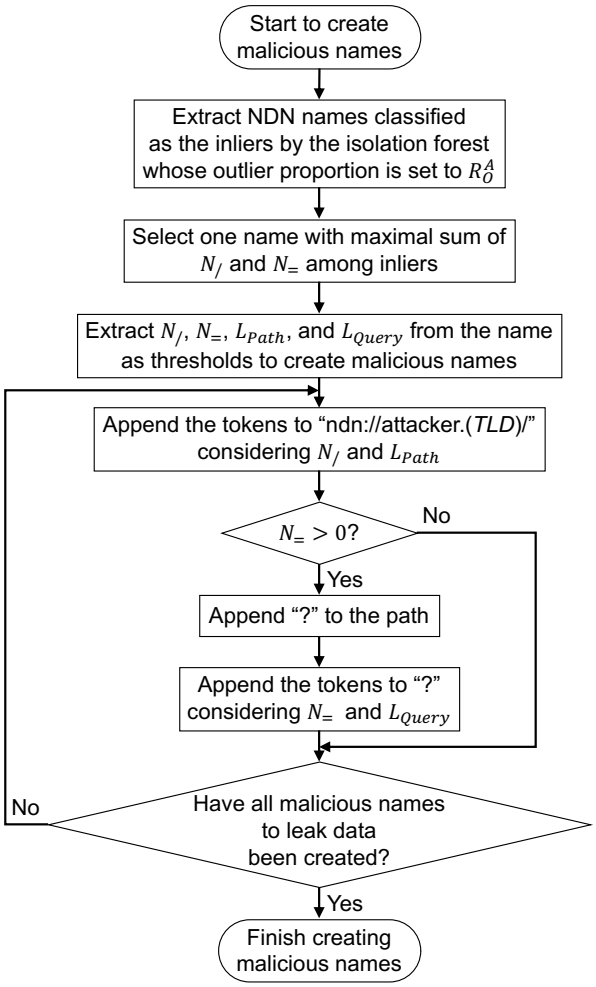


Figure 5.12: Flow to create malicious names.

side;

(ii) to select one name with maximal sum of $N_/$ and $N_=$ among inliers;

(iii) to extract $N_/$, $N_=$, L_{Path}, and L_{Query} from the name as the thresholds to create the malicious names; and

(iv) to create the malicious names with steganography considering the thresholds.

Table 5.6 indicates the thresholds of $N_/$, $N_=$, L_{Path}, and L_{Query} extracted from each TLD name dataset using the isolation forest. Moreover, to generate the malicious names with steganography, the attacker has to prepare a table with each token and its corresponding digits. The table is configured as follows; using "/", "=", and "&" as delimiters, this thesis collects the tokens from the path and from the query in the inliers identified by the isolation forest, respectively, and assigns four hexadecimal digits (i.e., two bytes information) to each token in the table.

As for a kind of tokens, Kondo et al. [96] utilize the dictionary words from WordNet [97] as tokens. This thesis, however, uses tokens collected from the real NDN names, or the NDN names converted from the real URLs used in the current Internet. The number of needed

tokens is $16^4 = 65,536$, and from the attacker's NDN name dataset shown in Table 5.3, this thesis extracts frequently used tokens in the inliers preferentially. Occasionally, this thesis could find some tokens whose length are quite large (e.g., percent-encoding [24]), and in terms of the information leakage throughput, smaller length is preferable. Thus, when there are several tokens of whom the frequency of appearance are the same in the inliers, this thesis takes the tokens in the ascending order of length while this thesis ignores the token whose length is larger than 90 percentiles of the length of name component shown in Table 5.4. Table 5.7 shows the number of tokens collected from the path and from the query in the inliers. As for the tokens from the path, in any cases, the number of tokens is 65,536. However, in some cases, the number of tokens from the query does not reach 65,536. In this case, each lack of the token is covered by the corresponding four hexadecimal digits. As a comparison to the steganography-embedded Interest name (hereafter, this thesis calls it *Token*), this thesis prepares malicious names where, instead of the token collected from the real NDN names, the corresponding four hexadecimal digits are directly used as the tokens (hereafter, this thesis calls it *Hex*).

The tokens from the path and from the query are inserted in the path and in the query in the malicious names, respectively, and especially, in the query, this thesis puts reserved keys such as "key1" before the equal characters (e.g., key1=*token1*&key2=*token2*) to identify the order of tokens. The reason of this convention is as follows. When the malware puts the tokens as the keys and the values in the query (e.g., *token1*=*token2*&*token3*=*token4*), the protector's name filter can shuffle the order of the key-value pairs, which does not have impact on legitimate content retrieval processes, but bothers the attacker since he or she cannot successfully decode the malicious names unless he or she decrypts the tokens in the query considering the order of them. Finally, the features of the created malicious names should be similar to the ones of the name selected in (ii).

Table 5.6: Thresholds of $N_{/}$, $N_{=}$, L_{Path}, and L_{Query} to create malicious names

TLD	$R_O^A = 0.01$				$R_O^A = 0.05$				$R_O^A = 0.1$				$R_O^A = 0.2$				$R_O^A = 0.3$				$R_O^A = 0.4$			
	$N_{/}$	$N_{=}$	L_{Path}	L_{Query}	$N_{/}$	$N_{=}$	L_{Path}	L_{Query}	$N_{/}$	$N_{=}$	L_{Path}	L_{Query}	$N_{/}$	$N_{=}$	L_{Path}	L_{Query}	$N_{/}$	$N_{=}$	L_{Path}	L_{Query}	$N_{/}$	$N_{=}$	L_{Path}	L_{Query}
com	32	0	260	0	32	0	260	0	32	0	260	0	32	0	260	0	32	0	260	0	32	0	260	0
net	38	0	216	0	38	0	216	0	38	0	216	0	38	0	216	0	38	0	216	0	21	0	222	0
org	1	66	10	1381	1	66	10	1381	3	41	25	726	2	27	24	248	2	27	24	248	2	25	24	231
info	3	24	26	257	3	24	26	257	3	24	26	257	2	25	26	249	17	0	107	0	17	0	107	0
jp	2	29	27	244	2	29	27	244	27	0	214	0	21	0	112	0	21	0	112	0	21	0	112	0
fr	1	61	1	2352	1	61	1	2352	1	61	1	2352	3	26	32	234	18	0	118	0	18	0	118	0
uk	2	32	20	175	2	32	20	175	2	32	20	175	3	22	18	169	3	22	18	169	3	22	18	169
de	3	25	23	237	3	25	23	237	3	25	23	237	24	0	173	0	24	0	173	0	24	0	173	0

Table 5.7: Number of tokens collected from path and from query in inliers

TLD	$R_O^A = 0.01$		$R_O^A = 0.05$		$R_O^A = 0.1$		$R_O^A = 0.2$		$R_O^A = 0.3$		$R_O^A = 0.4$	
	Path	Query	Path	Query	Path	Query	Path	Query	Path	Query	Path	Query
com	65,536	65,536	65,536	65,536	65,536	65,536	65,536	65,536	65,536	65,536	65,536	61,967
net	65,536	65,536	65,536	65,536	65,536	65,536	65,536	65,536	65,536	65,536	65,536	31,737
org	65,536	65,536	65,536	65,536	65,536	65,536	65,536	65,536	65,536	65,536	65,536	42,258
info	65,536	65,536	65,536	65,536	65,536	65,536	65,536	65,536	65,536	48,855	65,536	7,647
jp	65,536	32,159	65,536	22,930	65,536	16,111	65,536	149	65,536	32	65,536	16
fr	65,536	65,536	65,536	65,536	65,536	65,536	65,536	33,788	65,536	89	65,536	8
uk	65,536	65,536	65,536	65,536	65,536	65,536	65,536	65,536	65,536	51,745	65,536	19,711
de	65,536	65,536	65,536	65,536	65,536	65,536	65,536	47,449	65,536	214	65,536	15

5.5.2 Results

Tables 5.8, 5.9, and 5.10 show the performances of each of the NDN name filters (R_O^P = 0.01, 0.2, 0.4) against the malicious names to leak data (R_O^A = 0.01, 0.05, 0.1, 0.2, 0.3, 0.4) in terms of **the true positive rate, which is defined as the ratio of the number of malicious names identified as malicious to the total number of generated malicious names**, and the information leakage throughput per Interest, whose unit is **bytes/Interest**. When, for example, a malicious name "`ndn://attacker.com/`*token1*`/`*token2*`/`*token3*`?key1=`*token4*`&key2=`*token5*" bypasses the filter successfully, the name can convey 10 bytes of leaked data to the attacker, and **the information leakage throughput can be calculated by the total amount of leaked data volume obtained from the malicious names successfully bypassing the filter divided by the total number of the created malicious names**. In these tables, the averaged value marked as blue boldface indicates maximum true positive rate or minimum information leakage throughput (i.e., the best performance from the protector's point of view) while the averaged value marked as red boldface indicates minimum true positive rate or maximum information leakage throughput (i.e., the best performance from the attacker's point of view). As for the performances of each of the NDN name filters where R_O^P is set to 0.05, 0.1, or 0.3, this thesis shows them in Appendix A since this thesis confirms that every time the value R_O^P increases, the true positive rate tends to increase and the information leakage throughput tends to decrease.

From Tables 5.8a, 5.9a, and 5.10a, in any configurations about R_O^A, every true positive rate for *Token* is less than the one for *Hex* since, by using the tokens actually obtained from the attacker's name datasets, the frequencies of letters and the other printable characters in the path and in the query of *Token* become similar to the ones of the inliers in the protector's name datasets. Thus, a steganography-embedded Interest name can circumvent a protector more efficiently than a naive name such as one just including raw data. Moreover, the larger the R_O^P is, the higher, in any configurations about R_O^A, every true positive rate becomes, since, by increasing the R_O^P, the name filter is built by the name dataset excluding more outliers though the compensation for higher true positive rate is to obtain higher false positive rate. Against the higher true positive rate, the attacker can create the malicious names to bypass the filter by increasing the R_O^A.

From Tables 5.8b, 5.9b, and 5.10b, when the true positive rate for *Hex* is the same as the one for *Token*, the information leakage throughput using *Hex* is larger than the one using *Token* since *Hex* does not encode the data with steganography which causes throughput degradation in the case of *Token*. However, as mentioned previously, *Hex* can be detected as anomalous more easily than *Token*, so that the information leakage throughput using *Token* becomes larger than the one using *Hex* when the R_O^P becomes larger. Assuming that the protector adopts the name filter whose R_O^P is 0.4, the average of the information leakage throughput for all the domains is up to 22.6 bytes/Interest.

As shown in Table 5.6, there are three kinds of a malicious name: one composed of only path, of only query, and of both of path and query. According to Tables 5.8, 5.9, and 5.10, regardless of the kinds of a TLD, the generated malicious names which consist of only path can bypass the name filters more easily than the other kinds of a malicious name. That might indicate that, for malware, it is preferable to exploit only path to create malicious names leaking data and hide the activities.

Table 5.8: Performances of NDN name filter against malicious names ($R_O^P = 0.01$)

(a) True positive rate

TLD	$R_O^A = 0.01$		$R_O^A = 0.05$		$R_O^A = 0.1$		$R_O^A = 0.2$		$R_O^A = 0.3$		$R_O^A = 0.4$	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
net	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
org	0.593	0.131	0.593	0.133	0.972	0.411	0.890	0.379	0.890	0.383	0.868	0.403
info	1.00	0.863	1.00	0.851	1.00	0.815	0.999	0.773	0.00	0.00	0.00	0.00
jp	0.994	0.880	0.994	0.864	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
fr	0.00	0.00	0.00	0.00	0.00	0.00	0.821	0.193	0.00	0.00	0.00	0.00
uk	0.620	0.249	0.620	0.254	0.620	0.247	0.731	0.215	0.731	0.200	0.731	0.208
de	0.446	0.218	0.446	0.213	0.446	0.225	0.00	0.00	0.00	0.00	0.00	0.00
AVG	**0.457**	0.293	**0.457**	0.289	0.380	0.212	0.430	0.195	0.203	**0.0729**	0.200	0.0764

(b) Information leakage throughput per Interest [bytes/Interest]

TLD	$R_O^A = 0.01$		$R_O^A = 0.05$		$R_O^A = 0.1$		$R_O^A = 0.2$		$R_O^A = 0.3$		$R_O^A = 0.4$	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	64.0	51.2	64.0	51.2	64.0	51.3	64.0	50.8	64.0	49.4	64.0	46.0
net	76.0	44.9	76.0	43.7	76.0	43.0	76.0	44.9	76.0	46.7	42.0	41.9
org	54.5	116	54.5	116	2.43	50.8	5.49	26.4	5.49	26.4	6.07	23.8
info	0.00429	5.77	0.00429	6.30	0.00429	7.91	0.0708	9.78	34.0	20.8	34.0	21.2
jp	0.310	5.11	0.310	6.11	54.0	52.3	42.0	28.5	42.0	29.2	42.0	30.3
fr	124	124	124	124	124	124	8.93	21.8	36.0	17.9	36.0	19.0
uk	13.7	19.2	13.7	19.4	13.7	20.1	10.2	21.1	10.2	20.9	10.2	25.4
de	27.7	26.7	27.7	27.6	27.7	26.7	48.0	23.7	48.0	24.8	48.0	26.1
AVG	45.0	49.1	45.0	**49.3**	45.2	47.0	31.8	**28.4**	39.5	29.5	35.3	29.2

Table 5.9: Performances of NDN name filter against malicious names ($R_O^P = 0.2$)

(a) True positive rate

TLD	$R_O^A = 0.01$		$R_O^A = 0.05$		$R_O^A = 0.1$		$R_O^A = 0.2$		$R_O^A = 0.3$		$R_O^A = 0.4$	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	0.0154	0.00	0.0154	0.00	0.0154	1.50e-05	0.0154	0.00	0.0154	0.00	0.0154	0.00
net	0.000155	0.00	0.000155	0.00	0.000155	0.00	0.000155	0.00	0.000155	0.00	9.81e-05	1.22e-05
org	1.00	0.928	1.00	0.922	1.00	0.988	1.00	0.989	1.00	0.990	1.00	0.989
info	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.0395	1.82e-05	0.0395	1.86e-05
jp	1.00	1.00	1.00	1.00	0.00585	0.00	0.0104	0.00	0.0104	0.00	0.0104	8.86e-06
fr	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.996	0.00	0.00	0.00	0.00
uk	0.999	0.855	0.999	0.851	0.999	0.844	1.00	0.798	1.00	0.792	1.00	0.825
de	1.00	0.875	1.00	0.870	1.00	0.877	2.80e-05	0.00	2.80e-05	0.00	2.80e-05	0.00
AVG	**0.752**	0.707	**0.752**	0.705	0.628	0.589	0.503	0.473	0.258	**0.223**	0.258	0.227

(b) Information leakage throughput per Interest [bytes/Interest]

TLD	$R_O^A = 0.01$		$R_O^A = 0.05$		$R_O^A = 0.1$		$R_O^A = 0.2$		$R_O^A = 0.3$		$R_O^A = 0.4$	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	63.0	51.2	63.0	51.2	63.0	51.3	63.0	50.8	63.0	49.4	63.0	46.0
net	76.0	44.9	76.0	43.7	76.0	43.0	76.0	44.9	76.0	46.7	42.0	41.9
org	0.00524	9.71	0.00524	10.5	0.00	1.05	0.00	0.477	0.00	0.430	0.00	0.419
info	3.15e-05	7.38e-05	3.15e-05	0.00	3.15e-05	0.00	0.00	0.00	32.7	20.8	32.7	21.2
jp	0.00	0.00	0.00	0.00	53.7	52.3	41.6	28.5	41.6	29.2	41.6	30.3
fr	0.00442	0.00	0.00442	0.00	0.00442	0.00	0.00	0.103	36.0	17.9	36.0	19.0
uk	0.0180	3.62	0.0180	3.77	0.0180	4.04	0.00590	5.27	0.00590	5.25	0.00590	5.48
de	0.00657	4.23	0.00657	4.48	0.00657	4.13	48.0	23.7	48.0	24.8	48.0	26.1
AVG	17.4	**14.2**	17.4	**14.2**	24.1	19.5	28.6	19.2	**37.2**	24.3	32.9	23.8

Table 5.10: Performances of NDN name filter against malicious names ($R_O^P = 0.4$)

(a) True positive rate

TLD	$R_O^A = 0.01$		$R_O^A = 0.05$		$R_O^A = 0.1$		$R_O^A = 0.2$		$R_O^A = 0.3$		$R_O^A = 0.4$	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	0.698	1.49e-05	0.698	2.99e-05	0.698	1.50e-05	0.698	0.00	0.698	0.00	0.698	8.06e-05
net	0.694	2.62e-05	0.694	0.00	0.694	1.26e-05	0.694	5.25e-05	0.694	1.36e-05	0.377	0.000183
org	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
info	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.985	0.0888	0.985	0.0871
jp	1.00	1.00	1.00	1.00	0.999	0.114	0.994	0.144	0.994	0.135	0.994	0.149
fr	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.997	0.0741	0.997	0.0507
uk	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.999	1.00	1.00
de	1.00	1.00	1.00	1.00	1.00	1.00	0.839	0.0107	0.839	0.0115	0.839	0.0092
AVG	**0.924**	0.750	**0.924**	0.750	**0.924**	0.639	0.903	0.519	0.901	0.289	0.861	**0.287**

(b) Information leakage throughput per Interest [bytes/Interest]

TLD	$R_O^A = 0.01$		$R_O^A = 0.05$		$R_O^A = 0.1$		$R_O^A = 0.2$		$R_O^A = 0.3$		$R_O^A = 0.4$	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	19.3	51.2	19.3	51.2	19.3	51.3	19.3	50.8	19.3	49.4	19.3	46.0
net	23.3	44.9	23.3	43.7	23.3	43.0	23.3	44.9	23.3	46.7	26.2	41.9
org	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
info	3.15e-05	0.00	3.15e-05	0.00	3.15e-05	0.00	0.00	0.00	0.524	18.7	0.524	19.2
jp	0.00	0.00	0.00	0.00	0.0673	46.3	0.246	24.2	0.246	25.0	0.246	25.6
fr	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.110	16.4	0.110	17.9
uk	0.00	0.00487	0.00	0.00456	0.00	0.00252	0.00	0.00473	0.00	0.0229	0.00	0.00216
de	0.00	0.00	0.00	0.00	0.00	0.00	7.75	23.5	7.75	24.6	7.75	26.0
AVG	**5.33**	12.0	**5.33**	11.9	**5.33**	17.6	6.32	17.9	6.40	**22.6**	6.77	22.1

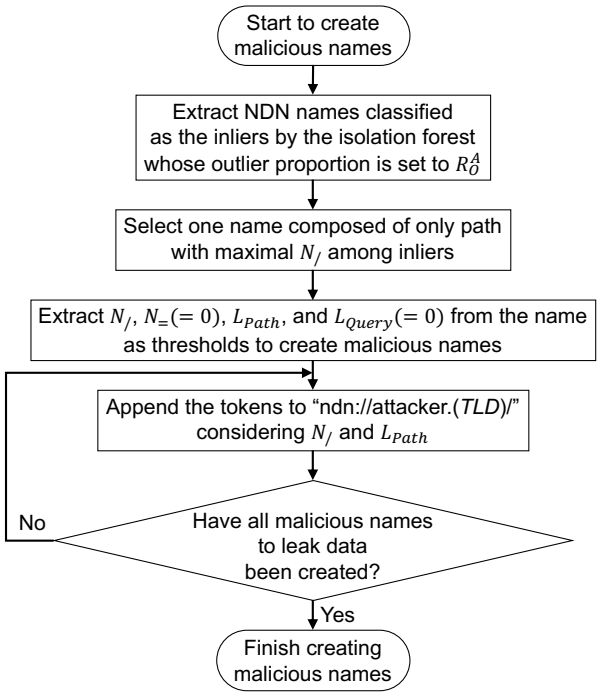


Figure 5.13: Flow to create malicious names exploiting only path.

Table 5.11: Thresholds of $N_/$ and L_{Path} to create malicious names exploiting only path

TLD	$R_O^A = 0.01$		$R_O^A = 0.05$		$R_O^A = 0.1$		$R_O^A = 0.2$		$R_O^A = 0.3$		$R_O^A = 0.4$	
	$N_/$	L_{Path}	$N_/$	L_{Path}	$N_/$	L_{Path}	$N_/$	L_{Path}	$N_/$	L_{Path}	$N_/$	L_{Path}
com	32	260	32	260	32	260	32	260	32	260	32	260
net	38	216	38	216	38	216	38	216	38	216	21	222
org	21	151	21	151	21	151	21	151	21	151	21	151
info	17	107	17	107	17	107	17	107	17	107	17	107
jp	27	214	27	214	27	214	21	112	21	112	21	112
fr	18	118	18	118	18	118	18	118	18	118	18	118
uk	24	189	24	189	24	189	24	189	24	189	24	189
de	24	173	24	173	24	173	24	173	24	173	24	173

To verify the above hypothesis that path is useful for malware to circumvent an NDN name filter and send leaked data to the outside attacker, this thesis also creates malicious names exploiting only path (Fig. 5.13). Table 5.11 shows thresholds of $N_/$ and L_{Path} to create the malicious names. Tables 5.12, 5.13, and 5.14 show the performances of each of the NDN name filters ($R_O^P = 0.01$, 0.2, 0.4) against the malicious names exploiting only path ($R_O^A = 0.01$, 0.05, 0.1, 0.2, 0.3, 0.4) in terms of the true positive rate and the information leakage throughput per Interest. As for the performances of each of the NDN name filters where R_O^P is set to 0.05, 0.1, or 0.3, this thesis shows them in Appendix A. As expected, the generated malicious names can bypass the name filters, and therefore, the true positive rate decreases and the information leakage throughput increases. Assuming that the protector adopts the name filter whose R_O^P is 0.4, the average of the information leakage throughput for all the domains is up to 32.1 bytes/Interest.

In terms of true positive rate and information leakage throughput, the above method to create malicious names using the tokens obtained from the NDN names in the real Internet is more efficient to leak data than the method proposed in [96], which uses the dictionary words from WordNet. As for the method of this thesis, the frequencies of letters and the other printable characters in the path become similar to the ones of the inliers in the protector's name datasets. In addition, this thesis selects one name with maximal sum of $N_{/}$ and $N_{=}$ among inliers while [96] obtains the longest name, so that the tokens can be added much more.

When the proposed filter is not used, the attacker can fill the Interest name with the leaked data in hexadecimal digits. Selecting the longest name in the attacker's name datasets, which the attacker can add 2,352 hexadecimal digits, the information leakage throughput reaches 1.18 Kbytes/Interest, which is maximum since 1 byte of the leaked data is mapped into 2 hexadecimal digits. Thus, by using the proposed filter, the malware has to send 36.8 times (=1.18 K/32.1) more Interests to the attacker than without using the filter. Moreover, theoretically, in NDN, much longer name is acceptable. Since a Data packet has to include the corresponding name in the Interest, the length of the name should be less than the maximum size of a Data packet (8800 bytes by default [100]). Thus, assuming that the maximum length of the name can consist of 8800 digits (i.e., 4.40 Kbytes/Interest), by using the proposed filter, the malware has to send 137 times (=4.40 K/32.1) more Interests to the attacker than without using the filter.

Table 5.12: Performances of NDN name filter against malicious names exploiting only path (R_O^P = 0.01)

(a) True positive rate

TLD	R_O^A = 0.01		R_O^A = 0.05		R_O^A = 0.1		R_O^A = 0.2		R_O^A = 0.3		R_O^A = 0.4	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
net	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
org	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
info	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
jp	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
fr	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
uk	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
de	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AVG	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**

(b) Information leakage throughput per Interest [bytes/Interest]

TLD	R_O^A = 0.01		R_O^A = 0.05		R_O^A = 0.1		R_O^A = 0.2		R_O^A = 0.3		R_O^A = 0.4	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	64.0	51.2	64.0	51.2	64.0	51.3	64.0	50.8	64.0	49.4	64.0	46.0
net	76.0	44.9	76.0	43.7	76.0	43.0	76.0	44.9	76.0	46.7	42.0	41.9
org	42.0	30.8	42.0	30.6	42.0	31.4	42.0	32.4	42.0	33.5	42.0	34.5
info	34.0	19.9	34.0	19.3	34.0	18.7	34.0	20.1	34.0	20.8	34.0	21.2
jp	54.0	52.1	54.0	52.2	54.0	52.3	42.0	28.5	42.0	29.2	42.0	30.3
fr	36.0	14.7	36.0	13.8	36.0	13.6	36.0	15.6	36.0	17.9	36.0	19.0
uk	48.0	31.5	48.0	32.3	48.0	33.3	48.0	34.8	48.0	36.2	48.0	38.1
de	48.0	20.7	48.0	21.5	48.0	22.4	48.0	23.7	48.0	24.8	48.0	26.1
AVG	**50.3**	33.2	**50.3**	33.1	**50.3**	33.3	48.8	**31.4**	48.8	32.3	44.5	32.1

Table 5.13: Performances of NDN name filter against malicious names exploiting only path ($R_O^P = 0.2$)

(a) True positive rate

TLD	$R_O^A = 0.01$		$R_O^A = 0.05$		$R_O^A = 0.1$		$R_O^A = 0.2$		$R_O^A = 0.3$		$R_O^A = 0.4$	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	0.0154	0.00	0.0154	0.00	0.0154	1.50e-05	0.0154	0.00	0.0154	0.00	0.0154	0.00
net	0.000155	0.00	0.000155	0.00	0.000155	0.00	0.000155	0.00	0.000155	0.00	9.81e-05	1.22e-05
org	0.000540	0.00	0.000540	0.00	0.000540	0.00	0.000540	0.00	0.000540	0.00	0.000540	0.00
info	0.0395	6.98e-05	0.0395	3.95e-05	0.0395	1.64e-05	0.0395	1.17e-05	0.0395	1.82e-05	0.0395	1.86e-05
jp	0.00585	0.00	0.00585	0.00	0.00585	0.00	0.0104	0.00	0.0104	0.00	0.0104	8.86e-06
fr	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
uk	1.40e-05	0.00	1.40e-05	0.00	1.40e-05	0.00	1.40e-05	0.00	1.40e-05	0.00	1.40e-05	1.11e-05
de	2.80e-05	0.00	2.80e-05	0.00	2.80e-05	0.00	2.80e-05	0.00	2.80e-05	0.00	2.80e-05	0.00
AVG	7.69e-03	8.73e-06	7.69e-03	4.94e-06	7.69e-03	3.93e-06	**8.25e-03**	**1.46e-06**	**8.25e-03**	2.28e-06	**8.25e-03**	6.35e-06

(b) Information leakage throughput per Interest [bytes/Interest]

TLD	$R_O^A = 0.01$		$R_O^A = 0.05$		$R_O^A = 0.1$		$R_O^A = 0.2$		$R_O^A = 0.3$		$R_O^A = 0.4$	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	63.0	51.2	63.0	51.2	63.0	51.3	63.0	50.8	63.0	49.4	63.0	46.0
net	76.0	44.9	76.0	43.7	76.0	43.0	76.0	44.9	76.0	46.7	42.0	41.9
org	42.0	30.8	42.0	30.6	42.0	31.4	42.0	32.4	42.0	33.5	42.0	34.5
info	32.7	19.9	32.7	19.3	32.7	18.7	32.7	20.1	32.7	20.8	32.7	21.2
jp	53.7	52.1	53.7	52.2	53.7	52.3	41.6	28.5	41.6	29.2	41.6	30.3
fr	36.0	14.7	36.0	13.8	36.0	13.6	36.0	15.6	36.0	17.9	36.0	19.0
uk	48.0	31.5	48.0	32.3	48.0	33.3	48.0	34.8	48.0	36.2	48.0	38.1
de	48.0	20.7	48.0	21.5	48.0	22.4	48.0	23.7	48.0	24.8	48.0	26.1
AVG	**49.9**	33.2	**49.9**	33.1	**49.9**	33.3	48.4	**31.4**	48.4	32.3	44.2	32.1

Table 5.14: Performances of NDN name filter against malicious names exploiting only path (R_O^P = 0.4)

(a) True positive rate

TLD	R_O^A = 0.01		R_O^A = 0.05		R_O^A = 0.1		R_O^A = 0.2		R_O^A = 0.3		R_O^A = 0.4	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	0.698	1.49e-05	0.698	2.99e-05	0.698	1.50e-05	0.698	0.00	0.698	0.00	0.698	8.06e-05
net	0.694	2.62e-05	0.694	0.00	0.694	1.26e-05	0.694	5.25e-05	0.694	1.36e-05	0.377	0.000183
org	0.160	9.90e-05	0.160	8.04e-05	0.160	9.16e-05	0.160	9.47e-05	0.160	5.87e-05	0.160	7.05e-05
info	0.985	0.170	0.985	0.122	0.985	0.0765	0.985	0.0775	0.985	0.0888	0.985	0.0871
jp	0.999	0.108	0.999	0.112	0.999	0.114	0.994	0.144	0.994	0.135	0.994	0.149
fr	0.997	0.107	0.997	0.0748	0.997	0.0641	0.997	0.0729	0.997	0.0741	0.997	0.0507
uk	0.351	0.000147	0.351	0.000208	0.351	7.79e-05	0.351	0.000101	0.351	0.000127	0.351	0.000167
de	0.839	0.00896	0.839	0.00891	0.839	0.00904	0.839	0.0107	0.839	0.0115	0.839	0.00920
AVG	**0.715**	0.0493	**0.715**	0.0398	**0.715**	**0.0330**	**0.715**	0.0382	**0.715**	0.0387	0.675	0.0371

(b) Information leakage throughput per Interest [bytes/Interest]

TLD	R_O^A = 0.01		R_O^A = 0.05		R_O^A = 0.1		R_O^A = 0.2		R_O^A = 0.3		R_O^A = 0.4	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	19.3	51.2	19.3	51.2	19.3	51.3	19.3	50.8	19.3	49.4	19.3	46.0
net	23.3	44.9	23.3	43.7	23.3	43.0	23.3	44.9	23.3	46.7	26.2	41.9
org	35.3	30.8	35.3	30.6	35.3	31.4	35.3	32.4	35.3	33.5	35.3	34.5
info	0.524	16.3	0.524	16.8	0.524	17.1	0.524	18.3	0.524	18.7	0.524	19.2
jp	0.0673	46.4	0.0673	46.3	0.0673	46.3	0.246	24.2	0.246	25.0	0.246	25.6
fr	0.110	12.9	0.110	12.6	0.110	12.5	0.110	14.3	0.110	16.4	0.110	17.9
uk	31.1	31.5	31.1	32.3	31.1	33.3	31.1	34.8	31.1	36.2	31.1	38.1
de	7.75	20.6	7.75	21.4	7.75	22.2	7.75	23.5	7.75	24.6	7.75	26.0
AVG	**14.7**	31.8	**14.7**	31.9	**14.7**	**32.1**	**14.7**	30.4	**14.7**	31.3	15.1	31.2

5.6 Discussion

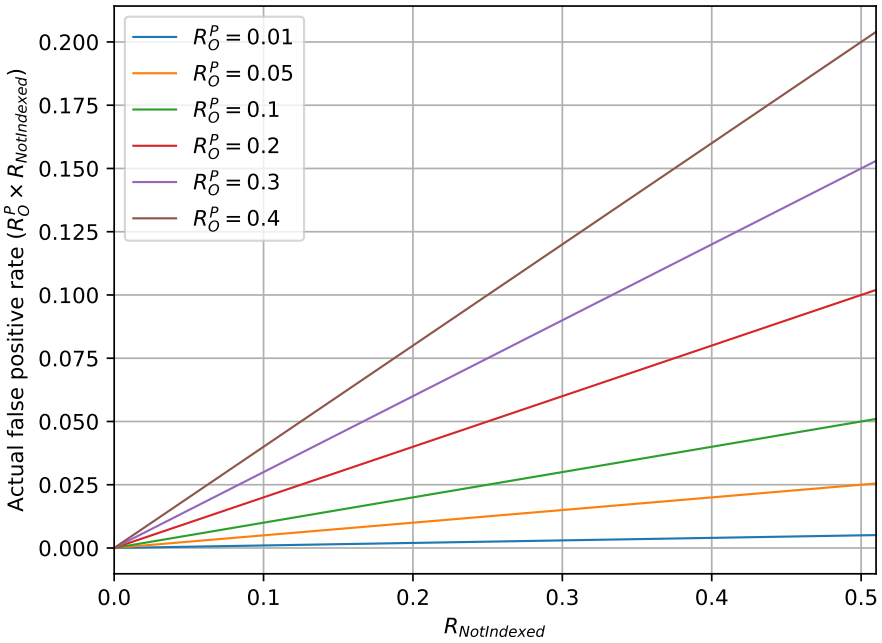


Figure 5.14: Actual false positive rate by using two name filters.

The false positive rate R_O^P introduced in Section 5.5 is caused by the name filter using the isolation forest. Actual false positive rate should be computed after checking Interest names with the name filter using the search engine information and using the isolation forest. Thus, the actual false positive rate is $R_O^P \times R_{NotIndexed}$, where $R_{NotIndexed}$ is the probability that legitimate users access real Deep Web content, which is not indexed by the search engine. Fig. 5.14 shows that the actual false positive rate for each R_O^P depends on $R_{NotIndexed}$, which will be very small if the enterprise network is properly managed. Therefore, the proposed filters can keep the false positive rate quite low thanks to the name filer using the search engine information.

The proposed algorithms to create malicious names shown in Figs. 5.12 and 5.13 might be non-optimal to increase the information leakage throughput. These algorithms concatenate several tokens using a slash character ("/") in path and an equal and an ampersand character ("=", "&") in query as delimiters. In the case that L_{Path} or L_{Query} is quite long, which means that there is large space to append the tokens to path or query as many as possible, by concatenating several tokens using some delimiters such as "-" and "_" and adding the concatenated tokens to the malicious names (e.g., `/apple_banana`), the malware could improve the information leakage throughput. However, as introduced in Section 5.4.2, in terms of SEO, "-" and "_" should be used to separate words, and therefore, the concatenated words should become a meaningful sentence following the grammar of the used language (e.g., in `https://moz.com/blog/15-seo-best-practices-for-structuring-urls`, `15-seo-best-practices-for-structuring-urls` should be "15 seo best practices for structuring urls" and that follows English grammar). As for the sequence of the tokens, it depends on data to leak, and therefore, if the malware just connects the tokens with such delimiters, the connected tokens would neither mean anything nor

follow some language grammar. The generated awkward name could be detected as anomalous by some Natural Language Processing (NLP) techniques [101]. Thus, it is extremely difficult for the malware to simply concatenate the tokens using some delimiters and improve the information leakage throughput.

In the Internet, in order to detect malicious URLs introduced in Section 2.4.1, some countermeasures rely on token-based filtering based on the fact that specific tokens are used in the malicious URLs. However, this kind of token-based filtering is not efficient to mitigate the information leakage attack through an Interest. For example, phishing URLs have to attract users using attractive tokens such as "paypal" and make them click the URLs. On the other hand, since the purpose of generating the name to perform the information leakage attack is not cheating users but leaking data, the name does not need to include such tokens. Thus, the malware can generate the name to leak data without any restrictions and the protector cannot filter out the name by using tokens.

In general, in the Internet, a host generating the URL (e.g., an FQDN) follows its own naming policy to effectively identify each Web page from the others using a small set of keywords. This may reduce the number of tokens following the name prefix. If one name prefix creates many names by appending many different types of tokens, that name prefix and the generated names may be detected as anomalous. Thus, to protect from an information leakage attack, the name filter has to know the statistical distribution of the number of possible tokens appearing after each slash character ("/") based on the many Web pages in the Internet. Once the name prefix is blacklisted, the malware and the attacker must exploit the other name prefixes that are shared between them by, for example, NPGA explained in Section 2.7).

As a precondition, to utilize the proposed NDN name filter, a policy to let an enterprise firewall inspect traffic from the employees carefully should be needed. For example, as introduced in Section 2.7, it is possible for a consumer to obfuscate the Interest name in some cases such as avoiding censorship. Like dropping *Hex*, the proposed name filter would also drop such an obfuscated name even if the name is legitimate and is not exploited to perform the information leakage attack. Moreover, as for an enterprise, in terms of risk hedge, the enterprise should manage any kinds of activities of the employees in order to not only detect the malicious Interest names created by malware but also avoid incidents such as malware infection caused by, for example, drive-by download attack and employee's leaking information to the outside intentionally.

The proposed name filters can drastically choke the information leakage throughput per Interest. Against the filters, however, to ameliorate the speed of this attack, malware can send numerous Interests within a short period of time. Moreover, the malware even can exploit an Interest with an explicit payload in the name (like an HTTP POST message in the Internet), which is out of scope in the name filters, and can increase the information leakage throughput by adopting a longer payload. These malicious activities can be detected by an NDN flow filter, which this thesis introduces from the next chapter.

5.7 Summary

This chapter proposed some countermeasures against an information leakage attack through a Data and through an Interest. Against the information leakage attack through a Data, this chapter proposed a whitelist-based name filter which is a naive but effective solution thanks to an NDN "pull"-based architecture. Against the information leakage attack through an Interest, this chapter proposed a name filter using search engine information and using an isolation forest. These two name filters can choke drastically the information leakage throughput and malware

has to send 137 times more Interest packets to leak information than without using the filters. Against the filters, however, to ameliorate the speed of this attack, malware can send numerous Interests within a short period of time. From the next chapter, this thesis defines an NDN flow and proposes an NDN flow filter to overcome the limitation of the NDN name filters.

Chapter 6

NDN Flow Monitored at Firewall

Contents

6.1 Introduction

In Chapter 5, as a first step in preventing the information leakage attack, this thesis proposed an NDN name filter derived from content names based on URLs collected by a web crawler, and discussed the information leakage throughput per Interest. To ameliorate the speed of this attack, malware can send numerous Interests within a short period of time. Moreover, the malware can even exploit an Interest with an explicit payload in the name (like an HTTP POST message in the Internet), which was out of scope in the name filter and can increase the information leakage throughput by adopting a longer payload. In order to detect the above malicious behaviour of the malware, this thesis proposes an NDN flow filter from the next chapter.

In this chapter, this thesis first newly defines the concept of NDN flow, which has not yet been standardized in NDN research [17]. Then, this thesis presents the conversion method of an HTTP flow dataset into the corresponding NDN flow dataset and generates the NDN flow dataset derived from the Waikato HTTP flow dataset [102] because there is no dataset about NDN traffic. Finally, this thesis shows the statistics of the NDN flow dataset.

The remainder of this chapter is organized as follows. Section 6.2 introduces how to generate the corresponding NDN flow dataset from an HTTP flow dataset. Section 6.3 shows the generated NDN flow dataset and its statistics. Finally, Section 6.4 summarizes this chapter.

6.2 Conversion of HTTP to NDN Flow

By assuming that applications over the current HTTP are run over NDN, this section shows how to generate the corresponding NDN flow dataset from an HTTP flow dataset.

In the Internet, a flow can be defined by 5-tuples (i.e., source/destination IP address, source/destination port number, and protocol type) (Fig. 6.1). Although many applications can generate the Internet flows, this thesis focuses on HTTP flows since HTTP is a name-centric protocol that is widely used to retrieve content named by URL [103]. Indeed, as HTTP has high affinity with NDN, some research efforts are derived from HTTP/NDN ([104, 105]). Yuan et al. [106] defined NDN flow using a name, and the flow was maintained by the expiration time and incoming face. The IRTF ICN Research Group also discussed name-based flow [107]. Fig. 6.2 shows the name-based NDN flow. Hereafter, this thesis discusses the NDN flow by assuming that the firewall shown in Fig. 6.2 is a monitoring point.

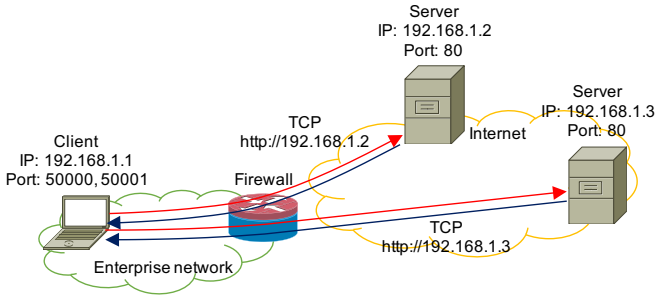


Figure 6.1: HTTP flow.

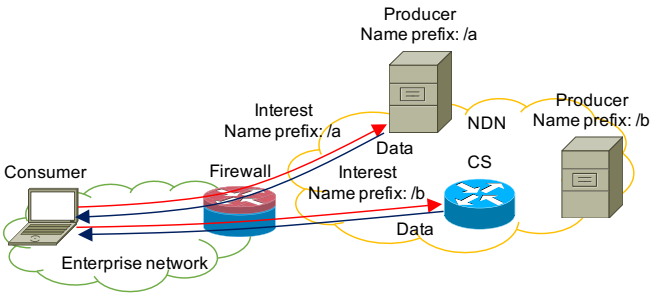


Figure 6.2: NDN flow.

Fig. 6.3 illustrates the proposed conversion scheme from an HTTP flow using 5-tuples into an NDN flow. For the NDN flow, this thesis follows a name-based flow idea discussed in [106] and [107], and utilize a name prefix to make the NDN flow. Note that [106] and [107] do not show a scheme to convert an HTTP flow into an NDN flow. In the HTTP flow, the client (192.168.1.1) first obtains the HTML file (file.html) from the server whose IP address is 192.168.1.2, and parses the file in order to access the URLs in the HTML file. Then, the client generates two flows to 192.168.1.2 and 192.168.1.3, where each request message obtains its corresponding response message. When this thesis converts the HTTP to NDN flow, this thesis associates the destination IP address with the name prefix. In performing the conversion, this thesis has to pay attention to the maximum size of a Data packet in NDN. Basically, in NDN one Interest retrieves one Data packet. When content size is larger than the maximum size of a Data packet (8800 bytes by default [100]), the content should be segmented into several Data packets, and multiple Interest packets corresponding to the Data packets can be generated. A manifest [108] is proposed, which includes a list of Data segment names (e.g., `/b/video.mpg/0`, `/b/video.mpg/1`) and meta-information of the Data packets. This thesis assumes that when a consumer accesses content, the consumer first obtains the manifest of the content, and the manifest can be collected by one Interest. To distinguish between an Interest to retrieve a manifest and an Interest specifying the segment name described in the manifest, this thesis names them ***manifest Interest*** and ***sequenced Interest***, respectively. In Fig. 6.3, the consumer first retrieves the manifest of the HTML file (file.html) from the producer whose name prefix is "`/a`". Sending the sequenced Interests listed in the manifest, the consumer then obtains and parses file.html, and continues to perform manifest-based content collection for image.jpg and video.mpg. These series of Interest packets make up the Interest flow to obtain the target object as well as the corresponding Data

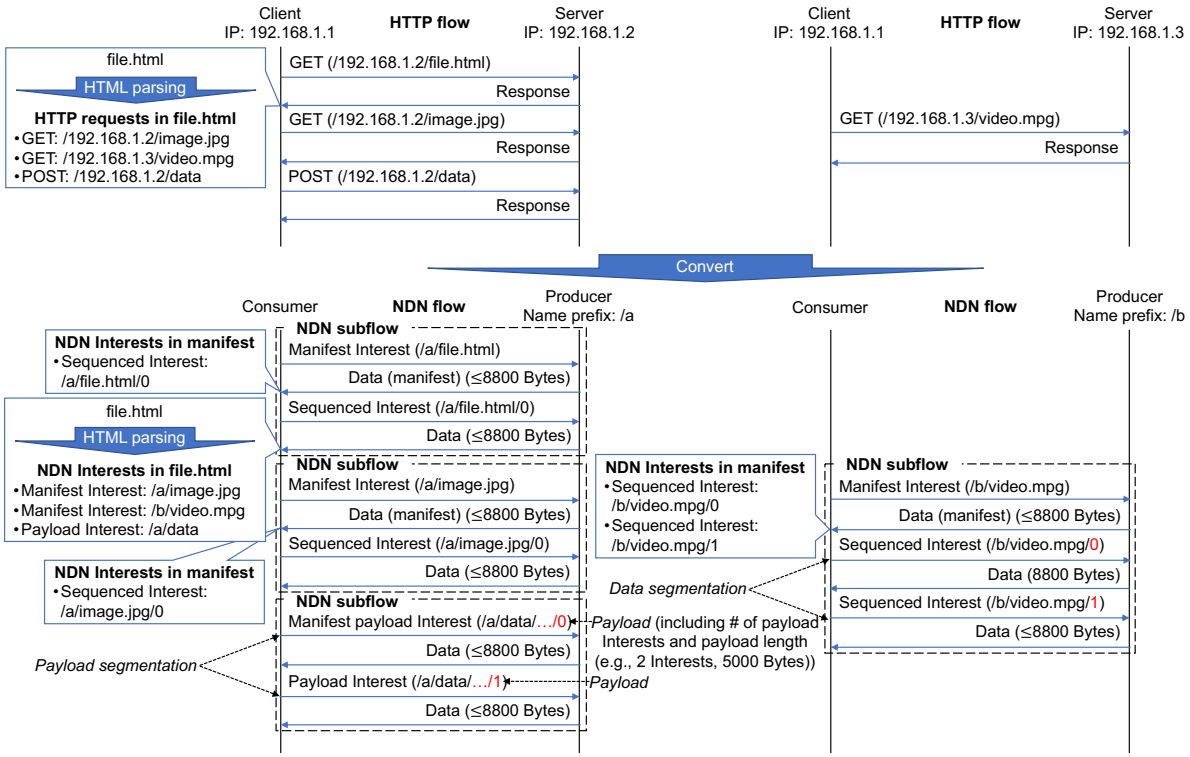


Figure 6.3: Conversion of HTTP to NDN flow.

flow. This thesis calls the flow to obtain each object as an ***NDN subflow***. Thus, a set of NDN subflows for each producer constitutes an NDN flow.

When an Interest appends a payload into the name like "`/a/data`" (e.g., Signed Interest [76] for the current implementation), the length of the content name may become longer than the maximum Data packet size. In this case, the Interest has to be divided into the multiple Interests whose name lengths leave some space for content in the Data, since the Data packet has to include the corresponding name. This thesis calls an Interest with a payload in the name a ***payload Interest***. To let the producer know how many successive Interests will be received and the payload length of these Interests, the first payload Interest should insert the number of successive Interests and the payload length in its payload. This thesis names the first payload Interest as a ***manifest payload Interest***. To the best of author's knowledge, a protocol to split an Interest that includes a large payload into several successive Interests, considering a maximum packet size of a Data, has not been discussed in detail in NDN research.

Table 6.1 presents an analogy between HTTP and NDN flow features monitored at the firewall in Fig. 6.2. According to the assumption about tracking each consumer, a source and a destination address in the HTTP flow should be a MAC/IP address of the consumer and a name prefix in the NDN Interest flow, respectively. When the consumer utilizes a signed Interest, the Interest includes the signature which has to be verified by the consumer public key, so that the signature in the signed Interest should also correspond to the source address in the HTTP flow. Likewise, the name prefix and producer signature of the Data in the Data flow should be the source address, and the MAC/IP address of the consumer in the Data flow should correspond to the destination address. Basically, a requested URL, GET/POST request message,

Table 6.1: Analogy between HTTP and NDN flow features

HTTP flow	NDN flow at firewall	
	Interest flow	Data flow
Source address	MAC/IP†address (and signature††)	Name prefix and signature
Destination address	Name prefix	MAC/IP†address
Requested URL	Name	Name
GET request message	Manifest Interest	N/A
POST request message	Manifest payload Interest and payload Interests	N/A
GET/POST response message	N/A	Data
HTTP response timeout	Expiration time	N/A
Segmentation (done by TCP)	Manifest payload Interest	Manifest
Session time (derived from TCP)	# of all Interest packets	# of all Data packets

† In the case of NDN over IP.
†† Signed Interest (optional).

GET/POST response message, and HTTP response timeout in the HTTP flow correspond to a content name, an Interest packet, a Data packet, and expiration time of an Interest in the NDN flow, respectively. For the GET request message, the message is a trigger to retrieve content and corresponds to a manifest Interest in NDN. The POST request message corresponds to a manifest payload Interest and the subsequent payload Interests. NDN does not have a transport layer; hence, segmentation done by TCP has to be realized by an NDN application. Thus, a manifest payload Interest and a manifest, which are operated by the NDN application, will compensate for this lack using payload Interests and sequenced Interests, respectively. In the HTTP flow, session time can be measured between SYN and the last packet of the HTTP flow. In contrast, in NDN a consumer does not make a session for a specific producer. The above NDN flow generation algorithm divides the HTTP session into several NDN subflows, each of which corresponds to getting or posting a specific object. The duration of an Interest subflow correlates with the number of Interest packets; thus, it is approximately linear to the time spent by the corresponding TCP session for the GET/POST request. The sum of these Interest packets can be seen to have a linear relation to the whole TCP session of the HTTP.

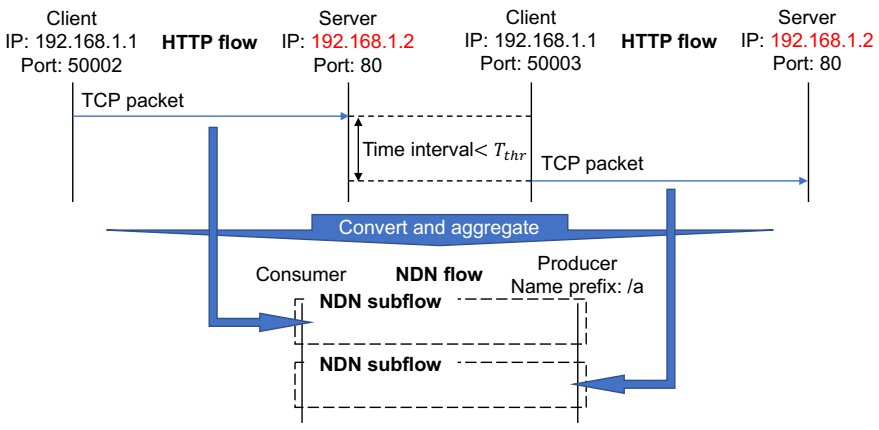


Figure 6.4: HTTP flow aggregation into NDN flow.

As for an HTTP flow, using different source port numbers, one client can create multiple HTTP flows to the same server at the same time. But in NDN, there is neither a source nor

a destination port number such that HTTP flows can be aggregated into one NDN flow, as shown in Fig. 6.4. The proposed NDN flow definition can better reflect the load of a specific producer. In order to decide how many HTTP flows should be aggregated, this thesis has to define a threshold for the packet time interval T_{thr} between HTTP flows. If the time interval is less than T_{thr}, then the flows can be combined.

Note that NDN installs CS (i.e., cache), which can distribute content instead of the content producer, so that the NDN flow is not an end-to-end flow like the HTTP flow (Fig. 6.1) but a hop-by-hop flow (Fig. 6.2), which might be terminated at intermediate NDN node's CS. Indeed, HTTP can also utilize cache such as a cache server (e.g., proxy) and browser cache. Even if the cache server is used, the destination IP address in the HTTP flow is that of the cache server such that an end-to-end flow can be still generated. When this thesis can obtain an HTTP flow dataset monitored at the firewall to which every consumer is directly connected via the underlay protocol such as an Ethernet or IP network, this thesis can generate the corresponding NDN flow dataset using the HTTP flow dataset. Moreover, the firewall can distinguish each consumer with a MAC/IP address as shown in Table 6.1.

Ongoing research in NDN have focused on several functions such as congestion control [109], which should be done by TCP in the Internet, but their standardization has yet to be achieved. Thus, in this study, this thesis assumes that NDN installs TCP-like functions.

6.3 NDN Flow Dataset and its Statistics Generated from HTTP Flow Dataset in the Current Internet

Here, following Section 6.2, this thesis shows how to obtain the corresponding NDN flow dataset from existing traffic traces.

6.3.1 HTTP Flow Dataset

Why Waikato Traces?

This thesis utilized Waikato University traffic traces in Waikato VIII (duration: Apr/7/2011–Nov/5/2011) [10
In these traces, the IP addresses were anonymized by Crypto-PAn AES encryption, which is a prefix-preserving anonymization method. The encryption key was changed once a week, and therefore this thesis can track each consumer for one week at the most. This thesis focuses on three traces[1], each of which is a one-day trace. These traces were captured from

- Wednesday, June 8, 12:00:01 (local time) to Thursday, June 9, 12:00:00 (local time),
- Monday, June 13, 12:00:01 (local time) to Tuesday, June 14, 12:00:00 (local time),
- and Wednesday, July 13, 12:00:01 (local time) to Thursday, July 14, 12:00:00 (local time).

[1]RFC 7230 [2] officially supports HTTP pipelining in HTTP/1.1 traffic, which allows a browser to send multiple request messages without waiting for the corresponding response messages. However, according to "HTTP Pipelining - Big in Mobile" [110], most major browsers did not support HTTP pipelining (or deactivated it by default). Only Opera for desktop and mobile, and Android for mobile employed HTTP pipelining in 2011. StatCounter [111] statistics on browser usage share in June 2011 showed that the share of Opera in desktop was 1.74% while the share of Opera and Android in mobile was 40.1%. As for the traffic volume of desktop and mobile in June 2011, 93.5% of all traffic came from desktop and 6.53% was mobile traffic. Thus, the ratio of HTTP traffic using HTTP pipelining to all the HTTP traffic in the Waikato HTTP traces this thesis utilized should be approximately 4.25% ($= 93.5 \times 0.0174 + 6.53 \times 0.401$). Accordingly, this thesis assumed that the HTTP traffic using HTTP pipelining was negligible and the traces consisted of HTTP traffic without HTTP pipelining.

There are two reasons to exploit the traces.

1. *Less HTTPS Traffic, More HTTP Traffic:* Recently, HTTPS traffic volume has drastically increased such that the application payload is encrypted, so that the request message information, such as GET or POST, cannot be extracted at all. However, the traces this thesis exploited were created in 2011; in fact, for outgoing flow from the university, the ratios of HTTPS traffic flow (destination port number: 443) to the whole Web traffic flow (HTTP (destination port number: 80 or 8080) and HTTPS traffic flow) in the three traces used are 15.03%, 10.17%, and 8.91%, respectively. Thus, in order to create an NDN flow dataset, this thesis can obtain much more information from these traces than the latest traces. According to [112], Facebook and YouTube started to support HTTPS in April 2013 and January 2014, respectively, which means that the traffic about such popular content providers in the latest trace cannot be analyzed. Thus, the latest trace does not always describe a content access model of user well.

2. *Availability of All Traffic Generated by Users:* The capture point of the traces was located between Waikato University's network and the Internet. This means that all the traffic generated by users in the university passed through the capture point. On the other hand, even if ISP traffic is captured at one point, it is possible for some traffic not to go through the point. In summary, the university traffic can imitate a hop-by-hop flow between a consumer and a capture point such as an enterprise network firewall.

HTTP Flow Extraction

First, using pkt2flow [113], this thesis extracted the HTTP flows from the traces based on 5-tuples. In order to make a flow filter, this thesis specified the IP address including the IP prefix of Waikato University, 80 or 8080, and TCP as a source IP address, a destination port number, and a protocol type, respectively. Then, this thesis set 30 min (default value in pkt2flow) as the flow timeout and ignored the flows that did not include the TCP SYN packet.

Usually, in an open dataset, an application payload is removed completely for the purpose of user privacy. In the Waikato HTTP traces, however, **each application payload is truncated to the first 4 bytes so that this thesis can understand the kind of request message such as GET or POST** (Fig. 6.5). Note that even though the dataset provided by Waikato University does not include requested URLs, this does not affect the modeling of some metrics in NDN such as the number of Interest packets within one subflow, which depends on the response message size. For the HTTP request messages in the extracted flows, this thesis focused on only GET and POST since this thesis converted these two messages into NDN subflows. In cases when other request messages such as HEAD existed in the HTTP flows, this thesis neglected the flows that included these request messages.

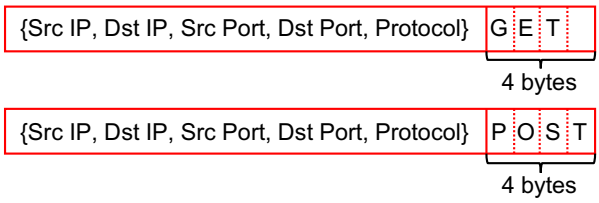


Figure 6.5: Truncated payload.

Table 6.2 shows a summary of the HTTP flow datasets. With regard to the number of Waikato IP addresses, it might not be equal to the number of clients in Waikato University because of network address translation. However, this thesis assumes that the number of Waikato IP addresses corresponds to the number of clients.

Table 6.2: Summary of HTTP flow datasets

	Jun/8–9	Jun/13–14	Jul/13–14
# of Waikato IP addresses	767	760	908
# of HTTP flows by only GETs	1,125,679	1,125,362	1,813,429
# of HTTP flows by only POSTs	246,233	322,384	229,393
# of HTTP flows by GETs/POSTs	28,550	31,553	10,932

6.3.2 NDN Flow Dataset

Table 6.3 shows a summary of the NDN flow datasets after the conversion of the HTTP flow datasets described in Section 6.3.1. Here, this thesis classifies the NDN consumers into legitimate and anomalous sets using the NDN flow datasets derived from the HTTP flow datasets in order to build an NDN flow filter which will be introduced in Chapter 7. First, in order to identify the consumers as legitimate or anomalous, as shown in Fig. 7.1, this thesis converted the 24-h flow datasets obtained from consumers into the corresponding feature vectors described in Table 7.2 and applied the isolation forest (the details of the process will be described in Section 7.3). To do so, this thesis configured the appropriate outlier proportion, which is one of the parameters of the isolation forest[2]. For two outlier proportions, 0.005 and 0.01, this thesis visualized the inliers and outliers using isomap in two dimensions[2] (Figs. 6.6a and 6.6b). With the outlier proportion set to 0.005 (Fig. 6.6a), the isolation forest separates the outliers more clearly from the inliers near the coordinate origin than when the outlier proportion was set to 0.01 (Fig. 6.6b). Thus, this thesis set the proportion to 0.005. Table 6.4 summarizes the obtained NDN flow datasets generated by legitimate and anomalous consumers after labeling flows from inliers and those from outliers as legitimate and anomalous, respectively.

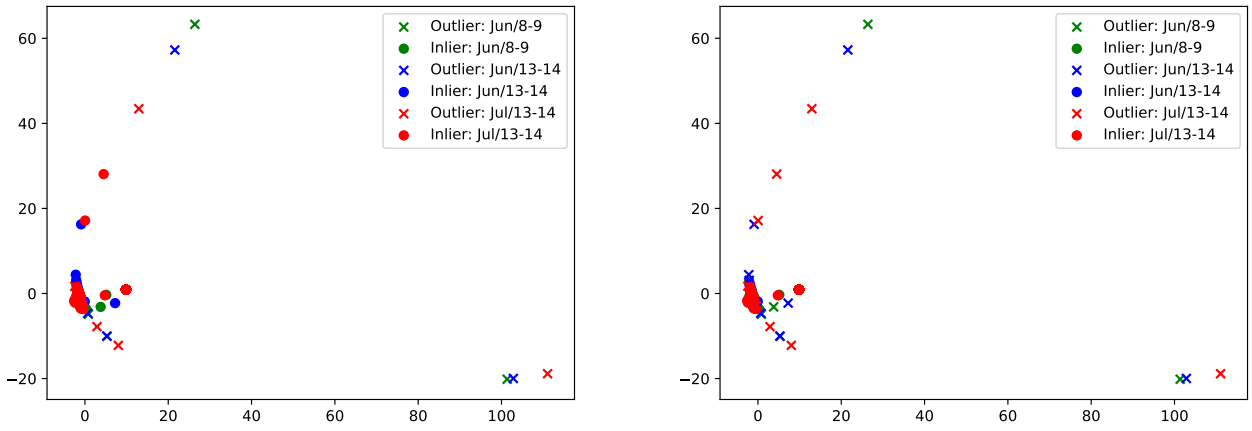
Table 6.3: Summary of NDN flow datasets

	Jun/8–9	Jun/13–14	Jul/13–14
# of Waikato consumers	767	760	908
# of NDN subflows by manifest Interest	3,167,397	3,183,382	2,841,950
# of NDN subflows by manifest payload Interest	359,083	443,360	277,641

6.3.3 Statistics

Figs. 6.7 and 6.8 describe CDFs of the number of sequenced Interests in one NDN subflow generated by a manifest Interest and those of data size of payload Interests in one NDN subflow generated by a manifest payload Interest on the Jun/8–9, Jun/13–14, and Jul/13–14 dataset,

[2]This thesis uses scikit-learn [98] with default values for the remaining parameters.



(a) 0.005 outlier proportion.

(b) 0.01 outlier proportion.

Figure 6.6: Visualization by isomap.

Table 6.4: Summary of legitimate and anomalous NDN flow datasets

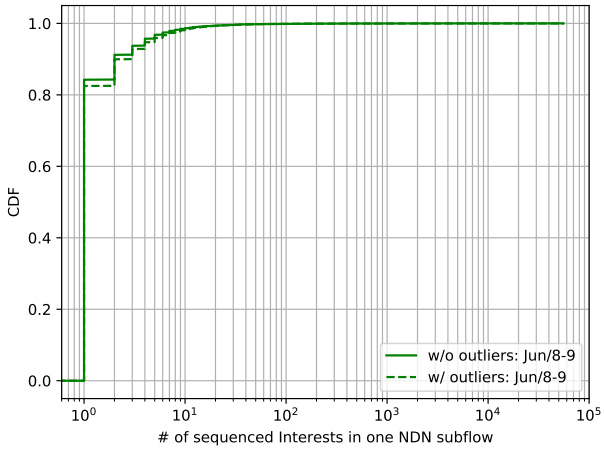
	Jun/8–9		Jun/13–14		Jul/13–14	
	Legitimate	Anomalous	Legitimate	Anomalous	Legitimate	Anomalous
# of Waikato consumers	763	4	756	4	903	5
# of NDN subflows by manifest Interest	1,055,045	2,112,352	911,194	2,272,188	871,669	1,970,281
# of NDN subflows by manifest payload Interest	48,916	310,167	36,061	407,299	44,533	233,108

respectively. According to Figs. 6.7a, 6.7b, and 6.7c, the presence or absence of the outliers does not have impact on the statistic, and also the change of the datasets (i.e., date) does not affect it, since retrieving content from the producer does not depend on whether the consumer is an inlier or an outlier. The outliers have impact on the number of NDN subflows generated by a manifest Interest as shown in Table 6.4.

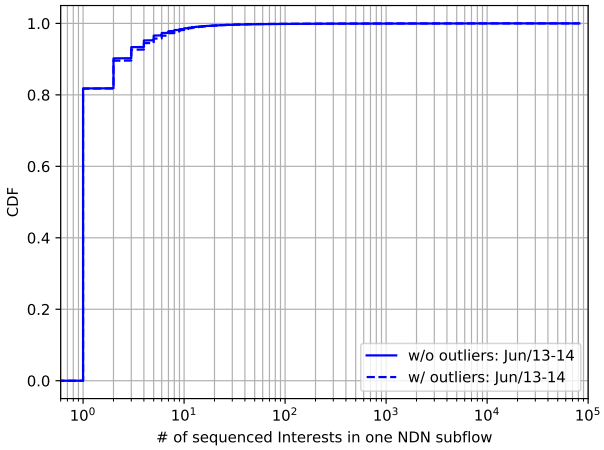
On the other hand, according to Figs. 6.8a, 6.8b, and 6.8c, the presence or absence of the outliers greatly have impact on the statistic while the change of the datasets also affects it. For example, near the point where the data size is around 300 bytes, the CDFs including the outliers show steepness, and that might be caused by a program often sending the data whose size is fixed. Moreover, the data size depends on the consumer (e.g., uploading content from the consumer), and that has impact on the CDFs.

From Figs. 6.7a, 6.7b, and 6.7c, around 82% of the NDN subflows include only one sequenced Interest. Assuming that the data size of a payload Interest is less than 8800 bytes, Figs. 6.8a, 6.8b, and 6.8c show that most of the NDN subflows consist of only one payload Interest.

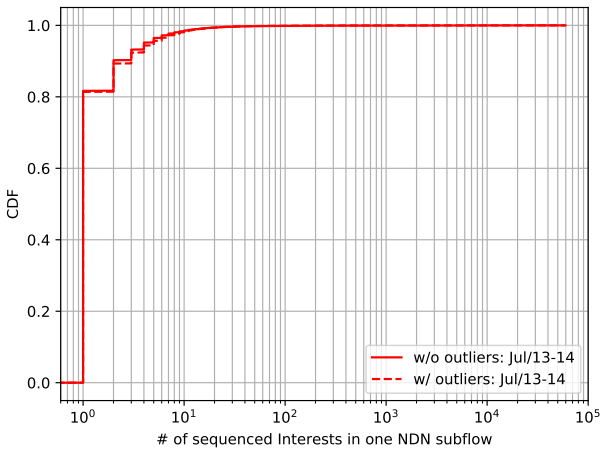
Fig. 6.9 shows the CDFs of time interval of both a manifest Interest and a manifest payload Interest (Manifest Interest + Manifest payload Interest), only a manifest Interest (Manifest Interest), and only a manifest payload Interest (Manifest payload Interest) on the Jun/8–9, Jun/13–14, and Jul/13–14 dataset. Note that Fig. 6.9 does not take account of the HTTP flow aggregation into the NDN flow in Fig. 6.4, and Fig. 6.9 is introduced by obtaining the NDN flows corresponding to each of the HTTP flows and calculating the time interval in each of the NDN flows. According to Figs. 6.9a, 6.9b, and 6.9c, regardless of the presence or absence of the outliers,



(a) Jun/8–9.

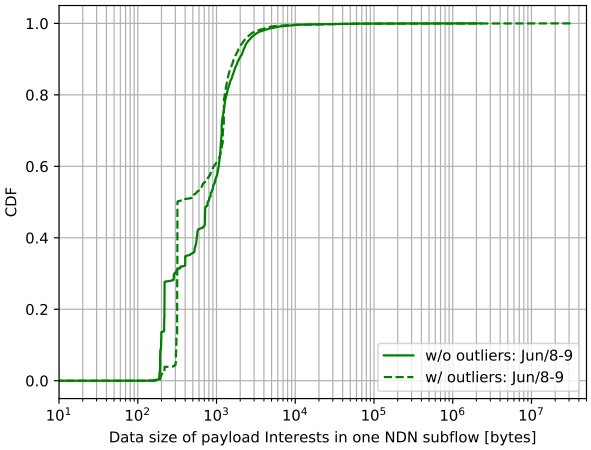


(b) Jun/13–14.

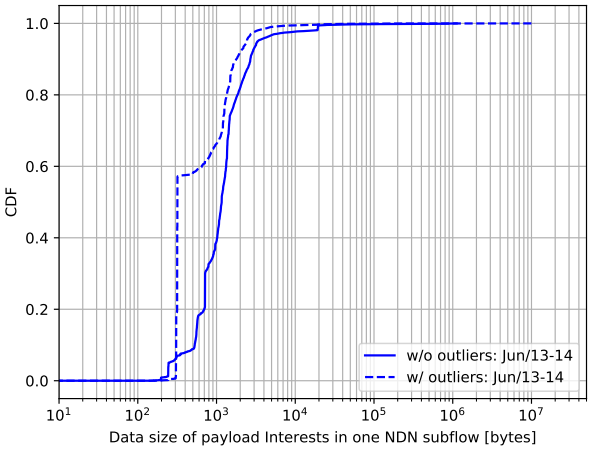


(c) Jul/13–14.

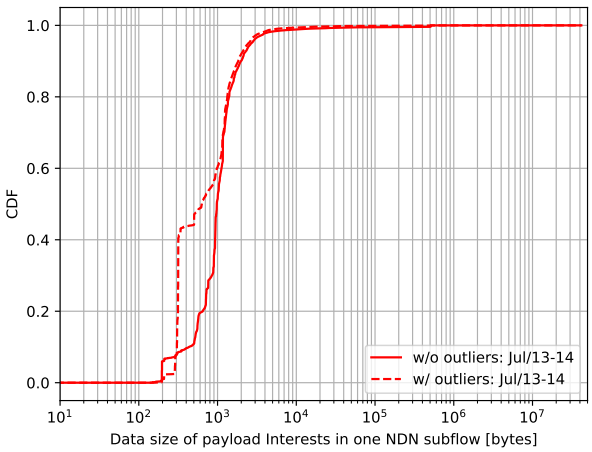
Figure 6.7: CDF of number of sequenced Interests in one NDN subflow.



(a) Jun/8–9.

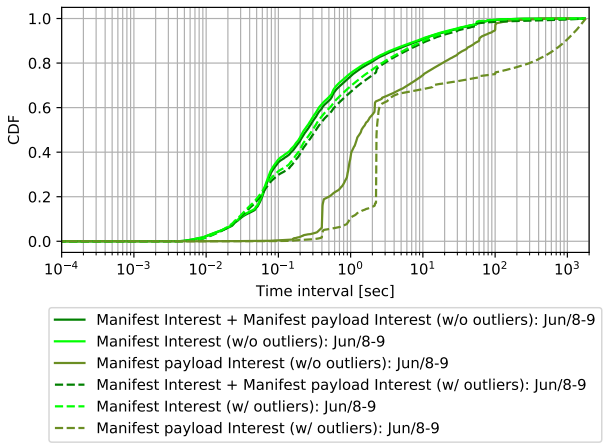


(b) Jun/13–14.

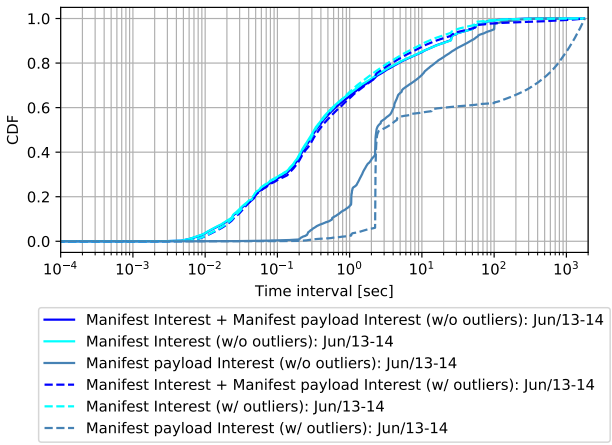


(c) Jul/13–14.

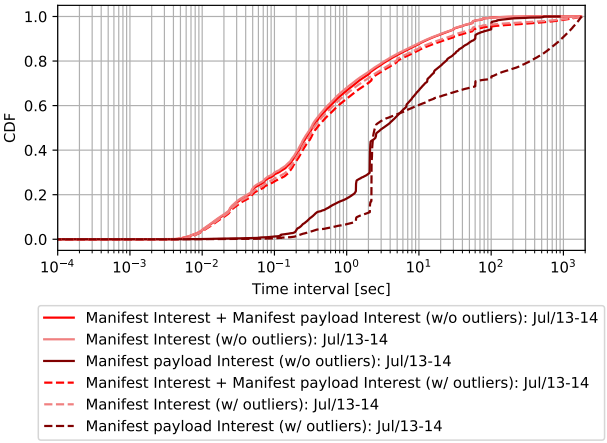
Figure 6.8: CDF of data size of payload Interests in one NDN subflow [bytes].



(a) Jun/8–9.



(b) Jun/13–14.



(c) Jul/13–14.

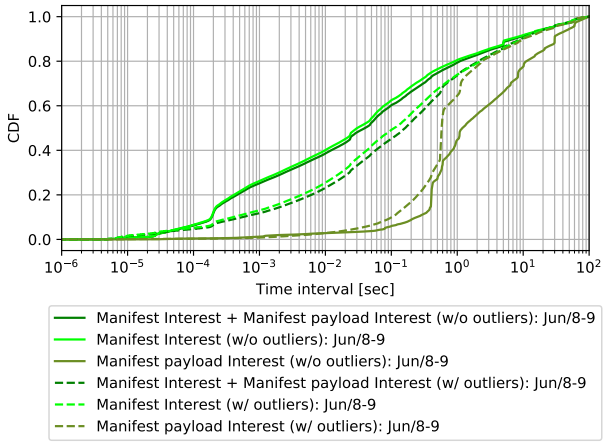
Figure 6.9: CDF of time interval [sec]. In Fig. 6.9b, the graph "Manifest Interest + Manifest payload Interest (w/o outliers): Jun/13–14" almost completely overlaps with the graph "Manifest Interest (w/o outliers): Jun/13–14".

the CDFs of time interval of both a manifest Interest and a manifest payload Interest are quite similar to the ones of only a manifest Interest. It is because the number of samples about time interval of a manifest Interest is around from 23 to 41 times larger than that of a manifest payload Interest. As for the CDFs of time interval of only a manifest payload Interest, which include the outliers, they indicate steepness around near the point where the time interval is around 2.25 sec, and that comes from the consumer which continuously sends a manifest payload Interest to the specific producer per around 2.25 sec. Moreover, regardless of the presence or absence of the outliers, the time interval of a manifest Interest is shorter than the one of a manifest payload Interest.

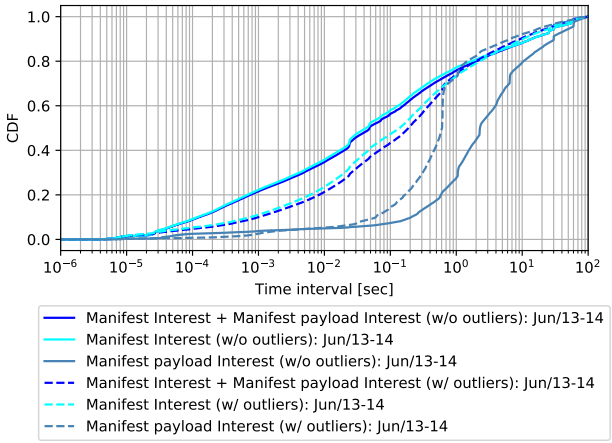
Here, this thesis considers the HTTP flow aggregation into the NDN flow in Fig. 6.4. From Fig. 6.9, around from 95 to 99% of the time intervals of a manifest Interest and a manifest payload Interest are less than 100 sec, and this thesis sets 100 sec to T_{thr}. Then, Fig. 6.10 can be obtained by applying the HTTP flow aggregation to Fig. 6.9. According to Figs. 6.10a, 6.10b, and 6.10c, by the aggregation, the time interval becomes shorter than the one shown in Fig. 6.9.

6.4 Summary

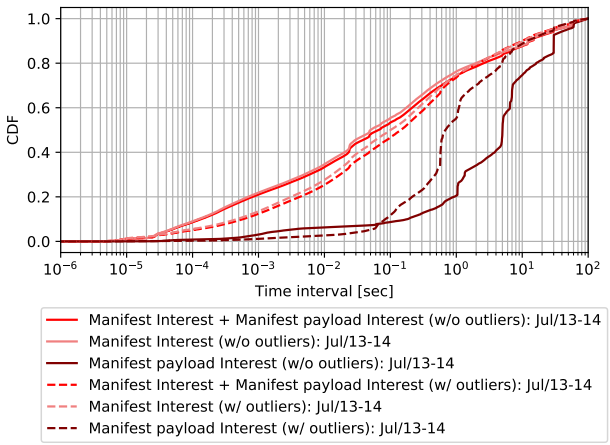
This chapter first introduced the concept of NDN flow and proposed how to derive the corresponding NDN flows from HTTP traces. Then, this chapter showed the statistics of the generated NDN flow datasets. From the next chapter, this thesis proposes an NDN flow filter and evaluates the performances of the proposed flow filter using the generated datasets.



(a) Jun/8–9.



(b) Jun/13–14.



(c) Jul/13–14.

Figure 6.10: CDF of time interval considering HTTP flow aggregation to NDN flow [sec].

Chapter 7

NDN Flow Filter

Contents

7.1 Introduction

In this chapter, this thesis proposes an NDN flow filter against information leakage through an Interest, and evaluates its performance using the NDN flow datasets shown in Chapter 6. Unlike the previously proposed name filter, the flow filter utilizes the features of an NDN flow but not those of an NDN name (descriptive features such as name length) from the constraints of the underlying HTTP dataset.

To this end, this thesis first investigates properties of malware generating malicious NDN flows causing information leakage. Then, using the inliers and outliers in the NDN flow datasets, this thesis builds an NDN flow filter based on an SVM for outlier detection. Since neither legitimate nor malicious NDN flow datasets are available, this thesis assumes that the malicious NDN flows tend to be categorized as outliers to achieve higher information leakage throughput. Hereafter, this thesis refers to the inliers and outliers as legitimate flows and anomalous flows, respectively. This thesis shows that such anomalous flows are effectively detected by the proposed filter. This forces malware to pretend a consumer generating a legitimate NDN flow for successful information leakage. The experiments show that, in this case, the information leakage throughput choked by the flow filter is from $1.32 \cdot 10^{-2}$ to $6.47 \cdot 10^{-2}$ times that choked only by the name filter, and this thesis also assesses the information leakage throughput which can be drastically choked by complementing the name filter.

The remainder of this chapter is organized as follows. Section 7.2 describes several properties of malware generating malicious NDN flows which cause information leakage. Against the

information leakage attack, Section 7.3 proposes an NDN flow filter which addresses the limitations of the proposed NDN name filter. Using the datasets introduced in Chapter 6, Section 7.4 evaluates the proposed flow filter. Section 7.5 discusses this research and Section 7.6 summarizes this chapter.

7.2 Properties of Malware Generating Malicious NDN Flows

To retrieve content, a consumer first obtains the manifest of the content using the manifest Interest. Since the names of successive sequenced Interests generated by each manifest do not have any letters other than their sequence numbers, the malware can exploit only the manifest Interest as a steganography-embedded Interest. This manifest Interest could be inspected by the proposed NDN name filter proposed in Chapter 5. The name filter can drastically choke the information leakage throughput per Interest by up to 32.1 bytes/Interest forcing the malware to send 137 times more Interests to leak information than without the filter. Thus, in order to obtain a high throughput, the malware has to rapidly send manifest Interests for information leakage to the attacker or the bots. Once the payload Interest is exploited, malware can ameliorate the information leakage throughput since the name filter cannot be applied to these Interests. Thus, within a specified period T, as the firewall monitors each consumer traffic, malware will adopt the four properties listed in Table 7.1 as a naive information leakage throughput improvement method. When *Properties 1* and *2* are satisfied by malware, this thesis calls it ***manifest Interest-based malware***. On the other hand, when *Properties 3* and *4* are satisfied, this thesis calls the malware ***payload Interest-based malware***. When the malware shows *Properties 1, 2, 3,* and *4*, this thesis calls it ***manifest and payload Interest-based malware***. These properties will be referred to define a feature vector in Section 7.3 and analyzed in Section 7.5.

Table 7.1: Properties of malware generating malicious NDN flows

Property 1	The number of manifest Interests will be large.
Property 2	The number of sequenced Interests per manifest Interest will be small.
Property 3	The number of manifest payload Interests will be large.
Property 4	The size of data sent by payload Interests will be large.

7.3 Proposal of NDN Flow Filter

In order to protect against attack, this thesis needs a filter (installed in the firewall) that determines whether an NDN flow is legitimate or malicious. Instead of manually programming such a filter, this thesis aims to build it using machine learning.

One of the standard procedures is to use a classification algorithm that builds a model using legitimate and malicious activities, which then predicts whether an activity is legitimate or malicious. In the thesis's case, this thesis does not have any actual normal or malicious data. Thus, based on monitoring *long-term* activity (24 h), this thesis classified consumers as either normal (inliers) or anomalous consumers (outliers). Using the same one-day dataset, this thesis made the firewall respond to the judgement quickly by monitoring each consumer's *short-term* activity (over the last few minutes). This means that the firewall cannot wait until the end of the day to decide whether certain activities are normal or not. Note, as mentioned previously, this

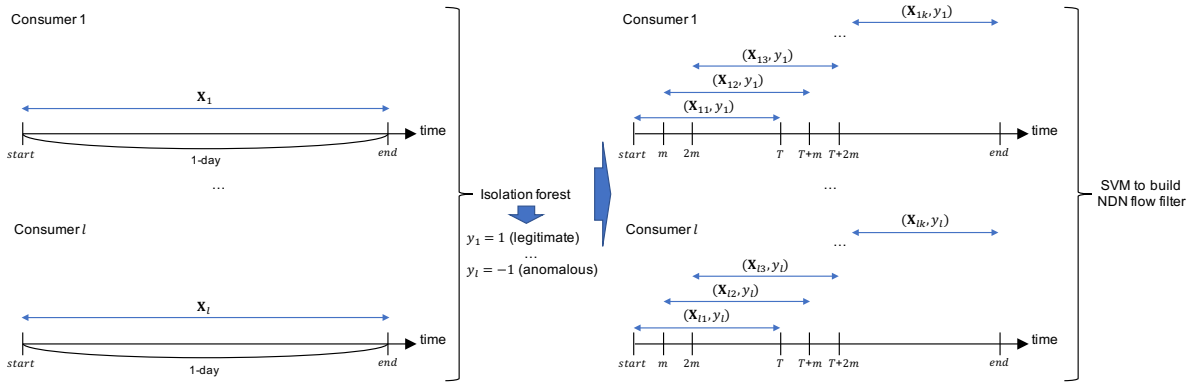


Figure 7.1: Dataset creation to build NDN flow filter.

thesis does not have any legitimate and malicious data, but has only normal and anomalous data. Thus, this thesis tries to build a flow filter for anomaly detection, which can be used for detection of malicious activities. Hereafter, this thesis uses the terms "legitimate" and "anomalous" instead of "legitimate" and "malicious."

In order to build a filter, given the above restrictions, this thesis performed the following steps (see Fig. 7.1):

(1) this thesis split an NDN flow dataset into l consumer datasets;

(2) this thesis extracted a feature vector $\mathbf{x}_i$ (described below and Table 7.2) for each consumer i based on its long-term activity (24 h) and created a dataset $D_C = \{\mathbf{x}_1, \ldots, \mathbf{x}_l\}$;

(3) this thesis utilized an isolation forest [16], a state-of-the-art anomaly detection algorithm, to mark the consumers as either legitimate ($y_i = 1$) or anomalous ($y_i = -1$);

(4) this thesis split the dataset of each consumer i into k window datasets of size T (short-term) shifted by an offset m, and extracted a feature vector $\mathbf{x}_{ij}$ (described below and Table 7.2) for all $j = 1, \ldots, k$ window datasets;

(5) this thesis created a labeled dataset for classification by associating each $\mathbf{x}_{ij}$ with its corresponding label y_i, calculated at step (3), $D_W = \{(\mathbf{x}_{ij}, y_i)\}_{i=1:l, j=1:k}$.

(6) this thesis used D_W to train an SVM classifier [18] for anomaly detection, which can construct an optimal hyperplane that separates legitimate from anomalous activities.

Table 7.2 indicates features extracted from a flow window of the NDN consumer. The first feature distinguishes two attack models (i.e., 1-to-1and 1-to-N attack model) shown in Section 3.2.2. The rest of features are derived from Table 7.1.

7.4 Experiments

7.4.1 Experimental Setup

Window Dataset

To build the SVM-based NDN flow filter described in Section 7.3, as shown in Fig. 7.1, this thesis needs to obtain the window datasets from the NDN flow datasets given by Section 6.3.2.

Table 7.2: Features extracted from flow window of NDN consumer

Notation	Feature variable
$N_{Producer}$	Total # of accessed producers
$N_{ManifestInterest}$	Total # of manifest Interests
$N_{SequencedInterest}$	Total # of sequenced Interests
$N_{ManifestPayloadInterest}$	Total # of manifest payload Interests
$S_{Payload}$	Total payload size of payload Interests
$\sigma_{N_{ManifestInterest}}$	STDEV of # of manifest Interests per producer
$\sigma_{N_{SequencedInterest}}$	STDEV of # of sequenced Interests per producer
$\sigma_{N_{ManifestPayloadInterest}}$	STDEV of # of manifest payload Interests per producer
$\sigma_{S_{Payload}}$	STDEV of payload size of payload Interests per producer

Table 7.3: Parameter settings for window T and offset m

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7
T [sec]	60	120	240	480	120	240	480
m [sec]	12	24	48	96	12	12	12

Table 7.3 shows the parameter settings for window T and offset m to extract the NDN window datasets. This thesis increased T exponentially starting at 60 s from Case 1 to Case 4, and starting at 120 s from Case 5 to Case 7. From Case 1 to Case 4, this thesis set each m as 0.2 multiplied by the corresponding T. From Case 5 to Case 7, this thesis fixed m at 12 s. It is possible for consumers not to generate any traffic in a specified period T; in this case, this thesis considers that the corresponding window dataset does not exist.

Training, Validation, and Test Sets

This thesis built seven NDN flow filters (each corresponding to Cases 1 to 7 described above) based on 70% of the Jun/8–9 dataset as the training and validation sets. To do so, this thesis performed feature scaling and a grid search on the hyperparameters[1] using five-fold cross-validation. Finally, this thesis evaluated the performance using 30% of the Jun/8–9 dataset as the test set.

In addition, this thesis used 100% of the Jun/13–14 and Jul/13–14 datasets as the test sets. This is done to evaluate the vulnerability of the training process, since this thesis expects that other days will have a different distribution, which should be reflected as worse results in the classification performance of the trained filters.

7.4.2 Results

Performance of NDN Flow Filter against Anomalous NDN Flows

[1]Kernel: [RBF], γ: [10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}], C: [10^0, 10^1, 10^2, 10^3, 10^4]

Table 7.4: Performance of NDN flow filter (based on Jun/8–9 dataset) in terms of precision (P), recall (R), and F_1 score on test sets

		Case 1			Case 2			Case 3			Case 4			Case 5			Case 6			Case 7		
		P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Jun/8–9	Legitimate	0.97	0.99	0.98	0.98	1.00	0.99	0.99	1.00	0.99	0.99	1.00	0.99	0.98	1.00	0.99	0.99	1.00	0.99	0.99	1.00	1.00
	Anomalous	0.93	0.69	0.79	0.94	0.73	0.82	0.96	0.76	0.85	0.96	0.79	0.86	0.94	0.74	0.83	0.98	0.78	0.87	0.99	0.80	0.89
Jun/13–14	Legitimate	0.94	0.99	0.96	0.96	0.99	0.98	0.97	0.99	0.98	0.98	1.00	0.99	0.96	0.99	0.97	0.97	0.99	0.98	0.97	1.00	0.98
	Anomalous	0.84	0.43	0.57	0.86	0.49	0.63	0.86	0.58	0.69	0.86	0.61	0.71	0.86	0.48	0.62	0.87	0.57	0.69	0.86	0.38	0.53
Jul/13–14	Legitimate	0.94	0.99	0.96	0.95	0.99	0.97	0.96	0.99	0.98	0.97	1.00	0.98	0.95	0.99	0.97	0.96	0.99	0.98	0.97	1.00	0.98
	Anomalous	0.84	0.42	0.56	0.86	0.43	0.57	0.82	0.43	0.56	0.85	0.43	0.57	0.85	0.42	0.57	0.84	0.43	0.57	0.79	0.30	0.44

Table 7.4 shows the performance of the seven filters in terms of precision[2](P), recall[3](R), and F_1 score[4] on all test sets. The results show that the legitimate class can easily be classified from the *short-term* windows of all trained filters (Cases 1 to 7), since all metrics (P, R, F_1) have a high value (≥ 0.94) on all test sets. A perfect score of 1.0 is not achieved since it is possible for a legitimate consumer to generate anomalous flows (which would be classified as anomalous).

On the other hand, the results of the anomalous class on the test set of Jun/8–9 (the day from which the training and validation sets were generated) show the highest P for Case 7 (0.99), the highest R for Case 7 (0.80), and the highest F_1 for Case 7 (0.89), and indicate that all metrics (P, R, F_1) increase as the window size increases (from Cases 1 to 4 and Cases 5 to 7). It is worth noticing that all metrics of the anomalous class have lower value than those of the legitimate class; this is expected since (1) in the dataset, there are only very few points for the anomalous class, compared to the legitimate one, and results in a suboptimal decision boundary; and (2) the anomalous consumers do not only generate anomalous flows, but create a lot of legitimate traffic as well.

The test sets of days Jun/13–14 and Jul/13–14 show a different story. In particular, P ranges from 0.84 to 0.87 for Jun/13–14, and from 0.79 to 0.86 for Jul/13–14; R ranges from 0.38 to 0.61 for Jun/13–14, and from 0.30 to 0.43 for Jul/13–14; F_1 score ranges from 0.53 to 0.71 for Jun/13–14, and from 0.44 to 0.57 for Jul/13–14. The reduction in performance compared to Jun/8–9 is more pronounced with the date difference (i.e., the results are worse for Jul/13–14 which is 1 month later compared to Jun/13–14 which is 5 days later).

Information Leakage Throughput of Malware Mimicking Legitimate

The information leakage throughput can be calculated by the following formula: $R_{Leaked} = 8 \times (S_{LeakedData} \times N_{ManifestInterest} + S_{Payload})/T$ bps[5], where the $S_{LeakedData}$ indicates the leaked data size by a manifest Interest, and the $N_{ManifestInterest}$ and $S_{Payload}$ are given in Table 7.2. To evaluate the throughput, this thesis assumes that the firewall installs not only the flow filter but also the name filter to inspect a manifest Interest. As this thesis introduced in Chapter 5, against a steganography-embedded Interest, the previously proposed name filter can choke throughput per manifest Interest by up to 32.1 bytes/Interest, and therefore, this thesis sets the $S_{LeakedData}$ as 32.1 bytes. Picking up a window dataset with the maximum throughput R_{Leaked} out of the whole dataset, this thesis will get the corresponding $N_{ManifestInterest}$ and $S_{Payload}$. This dataset is categorized as legitimate by the name filter, while the feature vector with these $N_{ManifestInterest}$ and $S_{Payload}$ is detected as anomalous by the flow filter. Thus, this thesis assumes that malware is clever and tries to mimic a legitimate consumer, and then maximizes R_{Leaked} in this restriction. In the following discussion, this thesis maximizes R_{Leaked} by benchmarking one window dataset from the training and validation dataset, which is identified as legitimate by the flow filter.

Table 7.5: Number of bots to perform maximum information leakage throughput

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7
# of bots	5	12	25	40	12	25	38

[2]The number of true positives over the number of true positives plus the number of false positives.
[3]The number of true positives over the number of true positives plus the number of false negatives.
[4]$F_1 = 2 \times (P \times R)/(P + R)$.
[5]1 byte = 8 bits

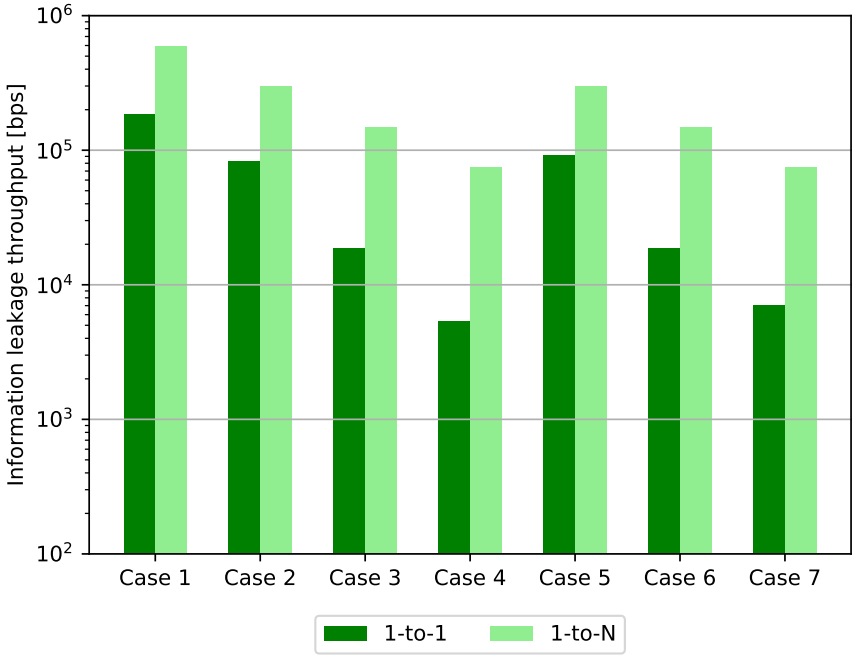


Figure 7.2: Maximum information leakage throughput in 1-to-1 and 1-to-N attack models.

To evaluate the 1-to-1 attack model, this thesis should pick one legitimate window dataset, $N_{Producer}$ of which is 1. Likewise, to evaluate the 1-to-N attack model, this thesis should pick one legitimate window dataset with $N_{Producer} \geq 2$. Fig. 7.2 shows the information leakage throughput in the 1-to-1 and 1-to-N attack models, which is calculated using the Jun/8–9 training and validation window dataset. The number of bots required to achieve maximum information leakage throughput is between 5 and 40 (Table 7.5). By exploiting several bots, the 1-to-N attack model achieves higher information leakage throughput than the 1-to-1 attack model. Moreover, in the 1-to-N attack model, the Data packet that the malware receives includes the bot's signature; hence, in terms of attacker anonymity, the 1-to-N attack model would be more useful. In summary, from attacker's point of view, the 1-to-N model is preferable, but as window T increases (i.e., from Cases 1 to 4 and from Cases 5 to 7), the number of needed bots increases to maximize the throughput.

Table 7.6: Number of bots to perform maximum information leakage throughput under banning payload Interests

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7
# of bots	2	3	4	51	3	4	54

Compared to a manifest Interest using steganography, a payload Interest might leak much more information easily since the name filter cannot inspect it and the payload does not have to be encoded. Thus, banning payload Interests might be a simple and effective solution against the information leakage attack, which means that only pulling content is accepted. Fig. 7.3

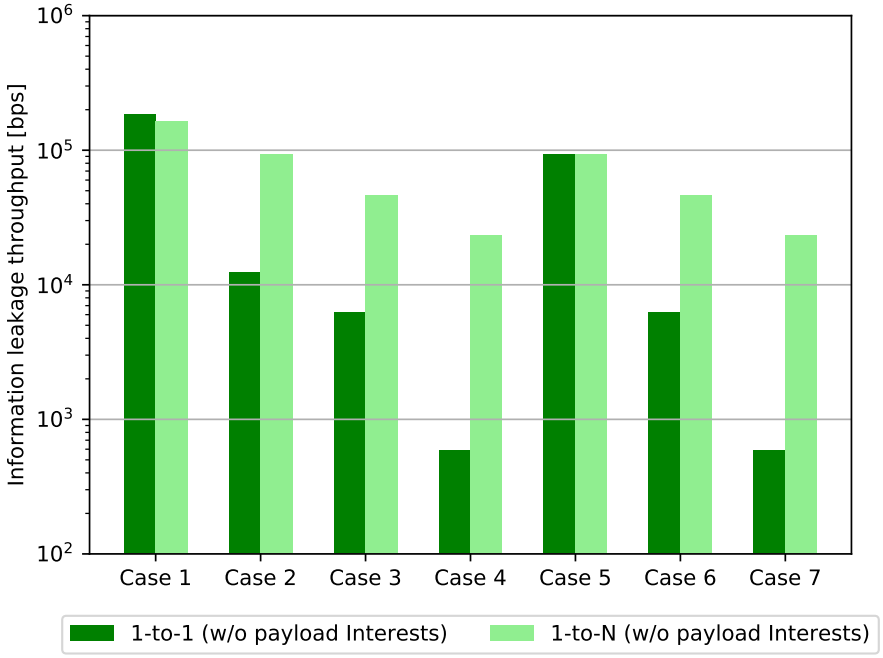


Figure 7.3: Maximum information leakage throughput in 1-to-1 and 1-to-N attack models under banning payload Interests.

shows the information leakage throughput in the 1-to-1 and 1-to-N attack models under banning payload Interests, which is calculated using the Jun/8–9 training and validation window dataset. To obtain the maximum information leakage throughput, this thesis picks one legitimate window dataset which maximizes R_{Leaked} while replacing $S_{Payload}$ to zero. The number of bots required to achieve maximum information leakage throughput is between 2 and 54 (Table 7.6). In any cases except Case 1, the 1-to-N attack model can achieve higher information leakage throughput than the 1-to-1 attack model, so the discussion is much the same as the one about Fig. 7.2.

Using the Jun/8–9 window dataset, this thesis also calculates the maximum information leakage throughput for three filters: the name and flow filter, the name and flow filter under banning payload Interests, and the only name filter (Fig. 7.4). In all cases, the throughput choked by the name and flow filter and by the name and flow filter under banning payload Interests are significantly lower than the throughput choked only by the name filter. This shows that the flow filter effectively solves the main problem of the name filter where the malware sends a lot of manifest and payload Interests in a short period of time. The throughputs after applying the name and flow filter to the traffic are $6.47 \cdot 10^{-2}$, $6.44 \cdot 10^{-2}$, $6.39 \cdot 10^{-2}$, $6.28 \cdot 10^{-2}$, $6.44 \cdot 10^{-2}$, $6.37 \cdot 10^{-2}$, and $4.24 \cdot 10^{-2}$ times those after applying only the name filter to the traffic in Cases 1 to 7, respectively, while the throughputs choked by the name and flow filter under banning payload Interests are $2.02 \cdot 10^{-2}$, $2.01 \cdot 10^{-2}$, $1.99 \cdot 10^{-2}$, $1.96 \cdot 10^{-2}$, $2.01 \cdot 10^{-2}$, $1.98 \cdot 10^{-2}$, and $1.32 \cdot 10^{-2}$ times those choked only by the name filter in Cases 1 to 7, respectively. After applying the name and flow filter to the traffic, the throughput can be reduced to 74.4 Kbps, and moreover, the throughput choked by the name and flow filter under banning payload Interests can be reduced to 23.2 Kbps.

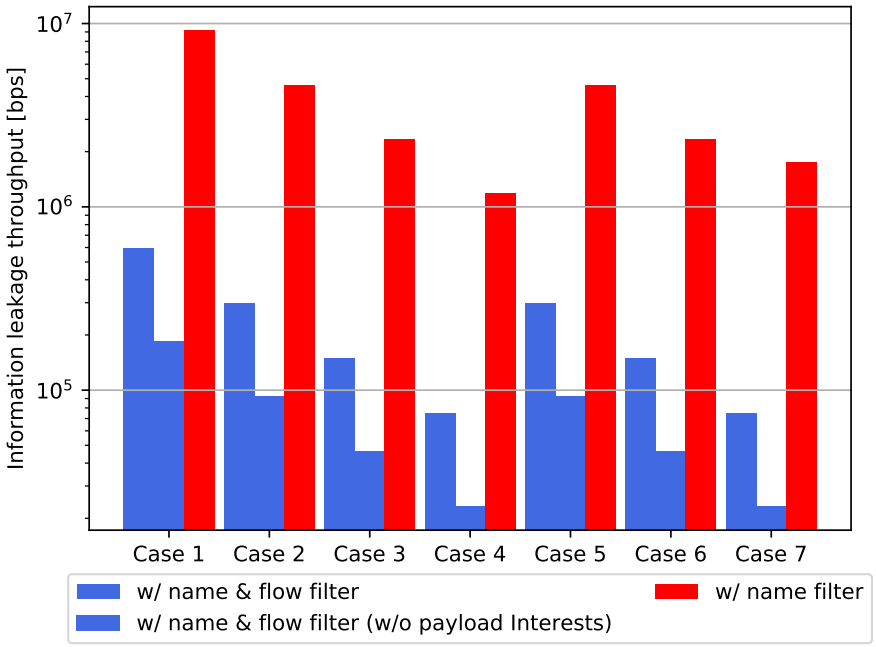


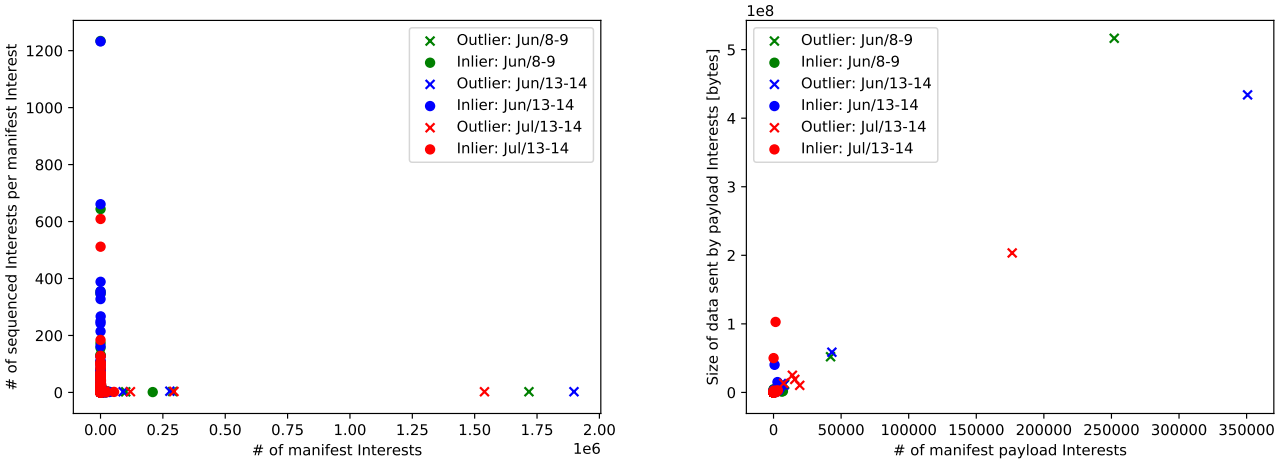
Figure 7.4: Maximum information leakage throughput choked by name and flow filter, by name and flow filter under banning payload Interests, and by only name filter.

7.5 Discussion

The results of Table 7.4 show that the filters are no longer effective 5 days (Jun/13–14) and 1 month (Jul/13–14) after the date (Jun/8–9) when the training and validation sets to build the filters were captured. It is also striking that the results of Case 7 at these later dates become worse than the ones of Case 6. This indicates that the distribution of anomalous data is different at these later dates, or in other words, the anomalous consumers have different features than the ones of the training day. It also highlights the need for training the filters periodically with up-to-date information. In particular, in this paper, this thesis intentionally used a one-shot training process, rather than a continual one, to showcase this vulnerability. On a deployed system, the filter should be periodically replaced (e.g., every couple of days) with a new one that is trained using a dataset which contains both the past activities as well as the most recent ones.

Following a similar line of thought, the SVM models this thesis uses have two limitations. First, their non-parametric nature makes them intractable to use with huge datasets since their computational complexity scales with the number of data points. Second, they do not have a mechanism for remembering information that they have seen in the past. A more robust and flexible filter could be made by employing types of parametric machine learning models that integrate temporal information, such as recurrent neural networks (RNNs) [114]. Such models could in principle remember past information and output at a given time step the probability that the activity is anomalous or legitimate, rather than deterministically perform a binary classification as the SVM models.

It is worth noting that in general, the extracted outliers might not always correspond to malware. For example, one outlier might be a crawler program whose behavior is totally different



(a) Two metrics about *Properties 1* and *2*.

(b) Two metrics about *Properties 3* and *4*.

Figure 7.5: Four metrics of outliers and inliers.

from that of a real user. To assess the appropriateness of the assumption that the malicious NDN flows tend to be categorized as outliers to achieve higher information leakage throughput, this thesis provides Figs. 7.5a and 7.5b to show four metrics of the outliers and inliers about *Properties 1, 2, 3,* and *4* that this thesis described in Section 7.2. From the results, three types of malware which this thesis defined in Section 7.2 can be approximated by the outliers.

In Table 7.3, the minimum value of T is 60 s, which means that it takes 60 s to detect the malware. Since the malware can generate and send out a lot of Interests within 60 s, it might leak all information before detected. One of the naive countermeasures is to limit the number of Interest packets sent within T. However, it is possible for this kind of activity to be legitimate (e.g., live streaming which might generate many Interests continuously). In this case, in order to avoid misdetection, acceptable name prefixes of such an application should be manually inserted into a whitelist of the firewall.

7.6 Summary

This chapter assessed the risk of information leakage through NDN Interest packets using the proposed NDN flow filter. First, this chapter presented properties of malware leaking information. Then, this chapter designed an SVM-based NDN flow filter and estimated that the information leakage throughput choked by the flow filter was from $1.32 \cdot 10^{-2}$ to $6.47 \cdot 10^{-2}$ times those choked only by the name filter, and the throughput choked by the name and flow filter under banning payload Interests could be reduced to 23.2 Kbps. Thus, the filter can drastically reduce the impact of this security threat for NDN.

Chapter 8

General Conclusion

Contents

The Internet has been widely used to provide a lot of applications for a long time on a host-to-host communication paradigm. However, considering the current trend of applications such as video service, which can be said as a host-to-content oriented application, the conventional host-to-host communication paradigm might not be optimal for the use case of the current Internet. To provide such an application more efficiently, one of the essential requirements is to perform a shift from a host-to-host to a host-to-content communication paradigm. On the security side, at the beginning, the Internet was not designed to endure security problems such as spoofing and tampering, so that the Internet must address these problems with the application layer such as SSL/TLS protocol. Indeed, by utilizing such a protocol, HTTPS (i.e., HTTP over SSL/TLS) can eliminate the problems, but adopting a host-to-host encryption method, it invalidates in-network caching (e.g., proxy) and then degrades the content delivery efficiency. Moreover, thanks to the encryption method, an attacker and the malware can establish the encrypted communication channel and avoid being detected by a firewall. As one of the attack scenarios, they can exploit the hidden channel to perform information leakage from an enterprise, which is one of the big issues in the Internet. Thus, designing a secure network architecture is also an emerging requirement.

In recent years, NDN has proposed as one of the most promising future networking architectures performing a shift from a host-to-host to a host-to-content communication paradigm. NDN adopts a security-by-design approach and practically defends against several security attacks in the Internet. For example, binding an NDN packet with the signature can eliminate spoofing and tampering while activating the in-network caching. NDN can satisfy some requirements, but, there is no discussion about a malicious communication channel causing information leakage in NDN. It is very crucial to assess such a security risk before replacing the Internet with NDN completely.

This thesis investigates whether a new security threat causing the information leakage can happen in NDN. Assuming that (i) a computer is located in the enterprise network that is based on an NDN architecture, (ii) the computer has already been compromised by suspicious media such as a malicious email, and (iii) the company installs a firewall connected to the NDN-based Future Internet, this thesis focuses on a situation that the compromised computer (i.e., malware) attempts to send leaked data to the outside attacker.

NDN is basically a "pull"-based architecture and there are only two kinds of packets: Interest and Data, which are a request and a response packet, respectively. In order to retrieve content, a consumer first sends the Interest to NDN network and then obtains the corresponding Data from the producer or the intermediate NDN node. In other words, they cannot send a Data unless they receive the Interest. Therefore, as one of the naive methods to mitigate information leakage through a Data, an enterprise firewall can carefully inspect a Data to publish, and produce it instead of the inside employee in the network (i.e., a whitelist).

However, the firewall cannot manage a naming policy on the outside content and NDN forwarding nodes do not verify if the name really exists. That causes a risk of information leakage through an Interest by malware's hiding information such as customer information in the Interest name and sending it toward the outside attacker. The malware can pretend to access outside content, so that it is quite difficult for the firewall to detect the information leakage attack. This thesis argues that the information leakage attack through an Interest in NDN should be one of the essential security attacks at protocol level and it is important to develop the detection method of this attack.

8.1 Achievements

The contributions of this thesis were fivefold. First, this thesis proposed an information leakage attack through a Data and through an Interest in NDN. This thesis investigated the attack deeply, and, as a more advanced attack for the attacker to hide the malicious activity, this thesis proposed a steganography-embedded Interest name to perform information leakage efficiently. To the best of author's knowledge, this is the first research about the information leakage attack in NDN.

Secondly, in order to address the information leakage attack, this thesis proposed an NDN firewall which monitors and processes the NDN traffic coming from the consumers with the whitelist and blacklist. To design the firewall, this thesis focused on two requirements: (i) designing an NDN firewall independent from NFD, which decides how to forward an Interest, and (ii) performing a fast lookup of the names or name prefixes in the whitelist and blacklist. By utilizing a cuckoo filter, which is a probabilistic filter, the proposed NDN firewall provided Interest packet filtering based on the names or name prefixes in the lists that can be updated on the fly. While satisfying the requirements and providing the functions, the firewall implementation achieved high performance. Concretely, the throughput degradation with the firewall was only from 0.912% to 2.34%. This thesis believes that this will be acceptable in an enterprise network and the source code is available on github [78].

Thirdly, this thesis proposed an NDN name filter to classify a name in the Interest as legitimate or anomalous. Since NDN has not been deployed on a large scale, a dataset about NDN traffic does not exist. Assuming that it is highly possible for the future NDN naming policy to become one naturally evolved from the current URL naming policy, this thesis utilized content names based on URLs collected by a web crawler. By using search engine information and applying the name dataset to an isolation forest, this thesis built NDN name filters. This thesis evaluated the performances of the name filters and showed that the proposed name filters could drastically choke the information leakage throughput and malware had to send 137 times more Interest packets to leak information than without using the filters hardening the possible attack. This is the first approach against information leakage through an Interest in NDN.

The name filter could, indeed, degrade the throughput per Interest, but to ameliorate the speed of this attack, malware can send numerous Interests within a short period of time. More-

over, the malware even can exploit an Interest with an explicit payload in the name (like an HTTP POST message in the Internet), which was out of scope in the proposed name filter and can increase the information leakage throughput by adopting a longer payload. That is the limitation of the name filter. To take traffic flow to the NDN firewall from the consumer into account, fourthly, this thesis proposed an NDN flow monitored at an NDN firewall. This thesis first introduced the concept of NDN flow and specified it strictly, which has not yet been standardized in NDN research. Then, this thesis proposed a method to generate an NDN flow dataset analogically derived from the HTTP flow dataset in the current Internet because there is no dataset about NDN traffic. The obtained statistics should be useful not only for a risk analysis of information leakage, which this thesis focuses on, but also simulation parameters for content access model.

Fifthly, in order to deal with the drawbacks of the NDN name filter, this thesis proposed an NDN flow filter to classify a flow as legitimate or not. Based on the generated NDN flow dataset, this thesis built an NDN flow filter against the information leakage attack. By applying the obtained dataset to an SVM, this thesis built an NDN flow filter against the information leakage attack, and the performance evaluation showed that the information leakage throughput choked by the flow filter was from $1.32 \cdot 10^{-2}$ to $6.47 \cdot 10^{-2}$ times that choked only by the name filter, and the throughput choked by the name and flow filter under banning payload Interests could be reduced to 23.2 Kbps. Thus, the flow filter complemented the name filter and greatly choked the information leakage throughput.

8.2 Future Work

The information leakage throughput choked by the name and flow filter can be reduced to 23.2 Kbps. In other words, by mimicking legitimate user's behaviour, it is possible for malware to leak data very slowly from an enterprise. In the case that the malware can continue to hide in the enterprise and the attacker does not care about the low throughput, the attacker can retrieve any kinds of a large file after waiting for a long time. Indeed, this kind of the case happens in the Internet and this attack is called Advanced Persistent Threat (APT) [115], which is a type of targeted attack. One of the features of the APT is "low and slow" approach to avoid the detection. Therefore, considering not only targeted attack but also APT, besides the proposed name and flow filter, several countermeasures have to be proposed. As future perspectives in this research area, this thesis shows several promising countermeasures below.

- Developing anti-malware software

  Usually, in an enterprise, in order for the employees to avoid malware infection, every computer installs anti-malware software. Even if the computer is infected by malware, once the information about the malware is added in the database of the anti-malware software, the malware can be excluded from the computer. However, after the attack has succeeded, using the covert channel, the attacker sends subspecific malware, and therefore, the attacker can bypass the firewall continuously. Especially, in the APT, malware is customized to perform the attack, so that it is extremely difficult for the firewall to detect it. Thus, this solution might not be so effective.

- Filtering Interest names based on whitelist

  An enterprise firewall can manage a content access policy in the enterprise by defining acceptable name prefixes or names and appending them into the whitelist. Especially, manipulating payload Interests should be effective against information leakage since payload

Interests can leak much more information than manifest Interests. Moreover, to perform some cases such as sending form data, one of the methods is to utilize payload Interests. Thus, the enterprise can mitigate the risk of the information leakage attack by whitelist-based filtering, and the overhead of the whitelist-based filtering is not heavy (see Chapter 4). However, the flexibility of the content access of the employees is restricted.

- Applying a time-series analysis to an NDN flow filter

  The NDN flow filter proposed in Chapter 7 defines a flow window and judges whether one flow window is legitimate sequentially. In other words, the filter does not have a mechanism for remembering information that the filter has seen in the past. One of the other promising approaches is to exploit a time-series analysis performed by, for example, RNNs. The RNNs have been designed to handle temporal data and it will be useful to detect anomalous flow.

- Analyzing cache hit and miss

  After collecting data to leak, malware generates malicious Interests and then sends them to the outside attacker. Thus, the corresponding Data packets are not cached in any NDN nodes. When the malware sends only the malicious Interests out, the successive cache misses from the infected computer should occur. This solution can also be applied to detect DNS tunneling, and to the best of author's knowledge, there is no research exploiting a feature of DNS cache hit and miss. Thus, the solution might be useful in both NDN and the Internet.

Appendix A

Performances of NDN name filters ($R_O^P = 0.05, 0.1, 0.3$)

Tables A.1, A.2, and A.3 show the performances of each of the NDN name filters ($R_O^P = 0.05$, 0.1, 0.3) against the malicious names to leak data ($R_O^A = 0.01$, 0.05, 0.1, 0.2, 0.3, 0.4) in terms of the true positive rate and the information leakage throughput per Interest, while Tables A.4, A.5, and A.6 show the corresponding performances against the malicious names exploiting only path. The discussion is much the same as Section 5.5.2 with the exception of the case to create malicious names including "fr" as the TLD. For these names, when R_O^A is set to 0.01, 0.05, or 0.1 and R_O^P to defend against the malicious names is set to 0.05 or 0.1, the true positive rate for *Token* is higher than the one for *Hex*. This is because the created malicious names consists of only query according to Table 5.6. From Figs. 5.8 and 5.10, as for "fr", the average frequency of percentage character ("%") in query is larger than the one in path, and therefore, percent-encoding might be used in query more than in path. Thus, in such a case, the true positive rate for *Hex* might be lower.

Table A.1: Performances of NDN name filter against malicious names ($R_O^P = 0.05$)

(a) True positive rate

TLD	$R_O^A = 0.01$		$R_O^A = 0.05$		$R_O^A = 0.1$		$R_O^A = 0.2$		$R_O^A = 0.3$		$R_O^A = 0.4$	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
net	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
org	0.975	0.512	0.975	0.505	1.00	0.747	0.997	0.743	0.997	0.751	0.996	0.769
info	1.00	0.995	1.00	0.995	1.00	0.993	1.00	0.985	0.00	0.00	0.00	0.00
jp	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
fr	0.00	0.00541	0.00	0.000891	0.00	0.00303	0.989	0.559	0.00	0.00	0.00	0.00
uk	0.946	0.555	0.946	0.556	0.946	0.544	0.977	0.490	0.977	0.469	0.977	0.489
de	0.937	0.606	0.937	0.598	0.937	0.613	0.00	0.00	0.00	0.00	0.00	0.00
AVG	**0.607**	0.459	**0.607**	0.457	0.485	0.363	0.495	0.347	0.247	**0.153**	0.247	0.157

(b) Information leakage throughput per Interest [bytes/Interest]

TLD	$R_O^A = 0.01$		$R_O^A = 0.05$		$R_O^A = 0.1$		$R_O^A = 0.2$		$R_O^A = 0.3$		$R_O^A = 0.4$	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	64.0	51.2	64.0	51.2	64.0	51.3	64.0	50.8	64.0	49.4	64.0	46.0
net	76.0	44.9	76.0	43.7	76.0	43.0	76.0	44.9	76.0	46.7	42.0	41.9
org	3.39	65.5	3.39	66.3	0.0204	21.7	0.142	10.9	0.142	10.6	0.168	9.20
info	0.000883	0.200	0.000883	0.223	0.000883	0.332	0.00146	0.647	34.0	20.8	34.0	21.2
jp	0.00	0.000571	0.00	0.00263	54.0	52.3	42.0	28.5	42.0	29.2	42.0	30.3
fr	124	123	124	124	124	124	0.530	11.8	36.0	17.9	36.0	19.0
uk	1.96	11.3	1.96	11.5	1.96	12.1	0.858	13.5	0.858	13.7	0.858	16.2
de	3.13	13.4	3.13	14.0	3.13	13.2	48.0	23.7	48.0	24.8	48.0	26.1
AVG	34.1	38.7	34.1	38.9	**40.4**	39.7	28.9	**23.1**	37.6	26.6	33.4	26.2

Table A.2: Performances of NDN name filter against malicious names ($R_O^P = 0.1$)

(a) True positive rate

TLD	$R_O^A = 0.01$		$R_O^A = 0.05$		$R_O^A = 0.1$		$R_O^A = 0.2$		$R_O^A = 0.3$		$R_O^A = 0.4$	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
net	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
org	0.999	0.623	0.999	0.619	1.00	0.898	1.00	0.901	1.00	0.907	1.00	0.911
info	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00
jp	1.00	1.00	1.00	1.00	0.00	0.00	2.45e-05	0.00	2.45e-05	0.00	2.45e-05	0.00
fr	0.178	0.399	0.178	0.343	0.178	0.687	0.999	0.789	0.00	0.00	0.00	0.00
uk	0.988	0.706	0.988	0.703	0.988	0.693	0.997	0.642	0.997	0.627	0.997	0.656
de	0.990	0.752	0.990	0.743	0.990	0.752	0.00	0.00	0.00	0.00	0.00	0.00
AVG	**0.644**	0.560	**0.644**	0.551	0.520	0.504	0.500	0.417	0.250	**0.192**	0.250	0.196

(b) Information leakage throughput per Interest [bytes/Interest]

TLD	$R_O^A = 0.01$		$R_O^A = 0.05$		$R_O^A = 0.1$		$R_O^A = 0.2$		$R_O^A = 0.3$		$R_O^A = 0.4$	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	64.0	51.2	64.0	51.2	64.0	51.3	64.0	50.8	64.0	49.4	64.0	46.0
net	76.0	44.9	76.0	43.7	76.0	43.0	76.0	44.9	76.0	46.7	42.0	41.9
org	0.178	50.6	0.178	51.1	0.00	8.81	0.00438	4.19	0.00438	3.96	0.00618	3.54
info	3.15e-05	0.00219	3.15e-05	0.00354	3.15e-05	0.00718	0.00	0.0146	34.0	20.8	34.0	21.2
jp	0.00	0.00	0.00	0.00	54.0	52.3	42.0	28.5	42.0	29.2	42.0	30.3
fr	102	74.5	102	81.4	102	38.9	0.038	5.69	36.0	17.9	36.0	19.0
uk	0.425	7.43	0.425	7.60	0.425	8.07	0.128	9.41	0.128	9.57	0.128	10.9
de	0.477	8.38	0.477	8.87	0.477	8.37	48.0	23.7	48.0	24.8	48.0	26.1
AVG	30.4	29.6	30.4	30.5	37.1	26.3	28.8	**20.9**	**37.5**	25.3	33.3	24.9

Table A.3: Performances of NDN name filter against malicious names ($R_O^P = 0.3$)

(a) True positive rate

TLD	$R_O^A = 0.01$		$R_O^A = 0.05$		$R_O^A = 0.1$		$R_O^A = 0.2$		$R_O^A = 0.3$		$R_O^A = 0.4$	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	0.219	0.00	0.219	0.00	0.219	1.50e-05	0.219	0.00	0.219	0.00	0.219	5.38e-05
net	0.0291	0.00	0.0291	0.00	0.0291	0.00	0.0291	0.00	0.0291	1.36e-05	0.0110	1.22e-05
org	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
info	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.415	0.000693	0.415	0.000552
jp	1.00	1.00	1.00	1.00	0.451	0.000122	0.418	0.000683	0.418	0.000435	0.418	0.000868
fr	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.780	0.00147	0.780	0.000598
uk	1.00	0.957	1.00	0.956	1.00	0.951	1.00	0.912	1.00	0.901	1.00	0.923
de	1.00	1.00	1.00	1.00	1.00	1.00	0.135	0.000208	0.135	0.000334	0.135	0.000107
AVG	**0.781**	0.745	**0.781**	0.745	0.712	0.619	0.600	0.489	0.500	**0.238**	0.497	0.241

(b) Information leakage throughput per Interest [bytes/Interest]

TLD	$R_O^A = 0.01$		$R_O^A = 0.05$		$R_O^A = 0.1$		$R_O^A = 0.2$		$R_O^A = 0.3$		$R_O^A = 0.4$	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	50.0	51.2	50.0	51.2	50.0	51.3	50.0	50.8	50.0	49.4	50.0	46.0
net	73.8	44.9	73.8	43.7	73.8	43.0	73.8	44.9	73.8	46.7	41.5	41.9
org	0.00	0.00522	0.00	0.00	0.00	0.0284	0.00	0.00948	0.00	0.0101	0.00	0.0136
info	3.15e-05	7.38e-05	3.15e-05	0.00	3.15e-05	0.00	0.00	0.00	19.9	20.8	19.9	21.2
jp	0.00	0.00	0.00	0.00	29.6	52.3	24.4	28.5	24.4	29.2	24.4	30.3
fr	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.91	17.9	7.91	19.0
uk	0.000378	1.06	0.000378	1.09	0.000378	1.24	0.00	2.27	0.00	2.47	0.00	2.39
de	0.00	0.0102	0.00	0.013	0.00	0.00574	41.5	23.7	41.5	24.8	41.5	26.1
AVG	15.5	12.1	15.5	**12.0**	19.2	18.5	23.7	18.8	**27.2**	23.9	23.2	23.4

Table A.4: Performances of NDN name filter against malicious names exploiting only path (R_O^P = 0.05)

(a) True positive rate

TLD	R_O^A = 0.01		R_O^A = 0.05		R_O^A = 0.1		R_O^A = 0.2		R_O^A = 0.3		R_O^A = 0.4	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
net	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
org	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
info	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
jp	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
fr	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
uk	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
de	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AVG	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**

(b) Information leakage throughput per Interest [bytes/Interest]

TLD	R_O^A = 0.01		R_O^A = 0.05		R_O^A = 0.1		R_O^A = 0.2		R_O^A = 0.3		R_O^A = 0.4	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	64.0	51.2	64.0	51.2	64.0	51.3	64.0	50.8	64.0	49.4	64.0	46.0
net	76.0	44.9	76.0	43.7	76.0	43.0	76.0	44.9	76.0	46.7	42.0	41.9
org	42.0	30.8	42.0	30.6	42.0	31.4	42.0	32.4	42.0	33.5	42.0	34.5
info	34.0	19.9	34.0	19.3	34.0	18.7	34.0	20.1	34.0	20.8	34.0	21.2
jp	54.0	52.1	54.0	52.2	54.0	52.3	42.0	28.5	42.0	29.2	42.0	30.3
fr	36.0	14.7	36.0	13.8	36.0	13.6	36.0	15.6	36.0	17.9	36.0	19.0
uk	48.0	31.5	48.0	32.3	48.0	33.3	48.0	34.8	48.0	36.2	48.0	38.1
de	48.0	20.7	48.0	21.5	48.0	22.4	48.0	23.7	48.0	24.8	48.0	26.1
AVG	**50.3**	33.2	**50.3**	33.1	**50.3**	33.3	48.8	**31.4**	48.8	32.3	44.5	32.1

Table A.5: Performances of NDN name filter against malicious names exploiting only path (R_O^P = 0.1)

(a) True positive rate

TLD	R_O^A = 0.01		R_O^A = 0.05		R_O^A = 0.1		R_O^A = 0.2		R_O^A = 0.3		R_O^A = 0.4	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
net	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
org	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
info	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
jp	0.00	0.00	0.00	0.00	0.00	0.00	2.45e-05	0.00	2.45e-05	0.00	2.45e-05	0.00
fr	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
uk	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
de	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AVG	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**3.06e-06**	**0.00**	**3.06e-06**	**0.00**	**3.06e-06**	**0.00**

(b) Information leakage throughput per Interest [bytes/Interest]

TLD	R_O^A = 0.01		R_O^A = 0.05		R_O^A = 0.1		R_O^A = 0.2		R_O^A = 0.3		R_O^A = 0.4	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	64.0	51.2	64.0	51.2	64.0	51.3	64.0	50.8	64.0	49.4	64.0	46.0
net	76.0	44.9	76.0	43.7	76.0	43.0	76.0	44.9	76.0	46.7	42.0	41.9
org	42.0	30.8	42.0	30.6	42.0	31.4	42.0	32.4	42.0	33.5	42.0	34.5
info	34.0	19.9	34.0	19.3	34.0	18.7	34.0	20.1	34.0	20.8	34.0	21.2
jp	54.0	52.1	54.0	52.2	54.0	52.3	42.0	28.5	42.0	29.2	42.0	30.3
fr	36.0	14.7	36.0	13.8	36.0	13.6	36.0	15.6	36.0	17.9	36.0	19.0
uk	48.0	31.5	48.0	32.3	48.0	33.3	48.0	34.8	48.0	36.2	48.0	38.1
de	48.0	20.7	48.0	21.5	48.0	22.4	48.0	23.7	48.0	24.8	48.0	26.1
AVG	**50.3**	33.2	**50.3**	33.1	**50.3**	33.3	48.8	**31.4**	48.8	32.3	44.5	32.1

Table A.6: Performances of NDN name filter against malicious names exploiting only path (R_O^P = 0.3)

(a) True positive rate

TLD	R_O^A = 0.01		R_O^A = 0.05		R_O^A = 0.1		R_O^A = 0.2		R_O^A = 0.3		R_O^A = 0.4	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	0.219	0.00	0.219	0.00	0.219	1.50e-05	0.219	0.00	0.219	0.00	0.219	5.38e-05
net	0.0291	0.00	0.0291	0.00	0.0291	0.00	0.0291	0.00	0.0291	1.36e-05	0.0110	1.22e-05
org	0.0199	0.00	0.0199	0.00	0.0199	0.00	0.0199	0.00	0.0199	0.00	0.0199	0.00
info	0.415	0.00207	0.415	0.00106	0.415	0.000563	0.415	0.000592	0.415	0.000693	0.415	0.000552
jp	0.451	0.000167	0.451	9.15e-05	0.451	0.000122	0.418	0.000683	0.418	0.000435	0.418	0.000868
fr	0.780	0.00395	0.780	0.00191	0.780	0.00149	0.780	0.00166	0.780	0.00147	0.780	0.000598
uk	0.0112	0.00	0.0112	0.00	0.0112	0.00	0.0112	0.00	0.0112	1.06e-05	0.0112	1.11e-05
de	0.135	9.68e-05	0.135	0.000201	0.135	0.000150	0.135	0.000208	0.135	0.000334	0.135	0.000107
AVG	**0.258**	0.000785	**0.258**	0.000408	**0.258**	0.000293	0.253	0.000393	0.253	0.000370	0.251	**0.000275**

(b) Information leakage throughput per Interest [bytes/Interest]

TLD	R_O^A = 0.01		R_O^A = 0.05		R_O^A = 0.1		R_O^A = 0.2		R_O^A = 0.3		R_O^A = 0.4	
	Hex	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*	*Hex*	*Token*
com	50.0	51.2	50.0	51.2	50.0	51.3	50.0	50.8	50.0	49.4	50.0	46.0
net	73.8	44.9	73.8	43.7	73.8	43.0	73.8	44.9	73.8	46.7	41.5	41.9
org	41.2	30.8	41.2	30.6	41.2	31.4	41.2	32.4	41.2	33.5	41.2	34.5
info	19.9	19.9	19.9	19.3	19.9	18.7	19.9	20.1	19.9	20.8	19.9	21.2
jp	29.6	52.1	29.6	52.2	29.6	52.3	24.4	28.5	24.4	29.2	24.4	30.3
fr	7.91	14.6	7.91	13.8	7.91	13.6	7.91	15.6	7.91	17.9	7.91	19.0
uk	47.5	31.5	47.5	32.3	47.5	33.3	47.5	34.8	47.5	36.2	47.5	38.1
de	41.5	20.7	41.5	21.5	41.5	22.4	41.5	23.7	41.5	24.8	41.5	26.1
AVG	**38.9**	33.2	**38.9**	33.1	**38.9**	33.3	38.3	**31.4**	38.3	32.3	34.2	32.1

Résumé de la thèse en français

1 Introduction générale

1.1 De Internet aux NDN

Depuis de nombreuses années, Internet a été largement utilisé pour fournir un nombre croissant d'applications, et de nouvelles applications voient le jour. Le Web, une des applications les plus connues et utilisées, est une application orientée hôte à hôte. Au cours des dernières années, les services vidéo sont de plus en plus populaires et, selon "The Zettabyte Era: Trends and Analysis" [1], d'ici 2021, 82% du trafic Internet sera du trafic vidéo (en volume). Dans ce genre d'application, la vidéo se trouve généralement sur le serveur et peut être récupérée en initiant une requête HTTP (Hypertext Transfer Protocol) [2] sur un URL (Uniform Resource Locator) [3] qui désigne le contenu à télécharger. Ce nouveau type d'application peut être défini comme une application orientée hôte à contenu, en cela que l'application se préoccupe plutôt du *contenu demandé* que de la *localisation du contenu*.

Internet n'est donc pas nécessairement adapté à ce fonctionnement des applications actuelles. Dans le détail, pour récupérer un contenu souhaité tel que la vidéo, un client envoie une requête de l'application au serveur dont l'adresse IP est spécifiée dans le champ d'adresse IP de destination du paquet. Dans ce cas, le paquet de réponse correspondant doit être renvoyé par le serveur. Toutefois, si les autres serveurs proches du client ont exactement le même contenu, ils peuvent également répondre au paquet de demande. Ainsi, ce processus de récupération de contenu peut ne pas être optimal pour une application orientée hôte à contenu. C'est une des limitations d'Internet pour le modèle d'accès aux contenus.

Pour adapter Intervent à ce cas d'utilisation, une nouvelle architecture de réseau a été proposée, le Information Centric Networking (ICN) [4]. Elle permet de passer d'un paradigme de communication hôte à hôte à un paradigme de communication hôte-contenu. Dans cette architecture, un paquet de demande ICN comprend le nom du contenu, et non l'adresse du producteur de contenu (sur Internet, l'adresse IP du serveur de contenu), et le paquet est envoyé à tous les producteurs de contenu à proximité. Ainsi, comme pour le producteur de contenu, non seulement l'éditeur de contenu, mais également le nud ICN disposant en cache du contenu peuvent répondre à cette demande. Cela signifie que larchitecture se concentre sur *le type de contenu demandé* plutôt que sur *le lieu où le contenu se trouve*. Par conséquent, ICN convient mieux aux applications orientées hôte à contenu, telles que les services vidéo présentés précédemment. Cette thèse, qui fait s'intégre dans la catégorie des architectures ICN, adopte le Named Data Networking (NDN) [5], car de nombreux chercheurs s'intéressent aux réseaux de type NDN et tentent de le déployer comme pouvant devenir le futur réseau Internet.

Un autre intérêt des réseaux de type NDN est qu'ils adoptent une approche de sécurité dès la conception, permettant ainsi de lutter contre plusieurs attaques existantes sur Internet. À l'origine, Internet n'a pas été spécifiquement conçu pour supporter des problèmes de sécurité

tels que l'usurpation d'identité et la falsification. Ces problèmes sont arrivés avec l'avénement de l'utilisation de HTTP. Actuellement, HTTPS (HTTP Secure) [6] est largement utilisé sur Internet et se base sur le couple SSL (Secure Sockets Layer) [7]/TLS(Transport Layer Security) [8], qui sont des protocoles de sécurité. HTTPS peut résoudre certains problèmes de sécurité, mais, en exploitant la méthode de chiffrement hôte à hôte avec TLS, il invalide la possiblité de mettre en cache (par exemple, via un proxy) et cela dégrade ensuite l'efficacité de la diffusion du contenu. Pour éliminer ces problèmes, un paquet NDN inclut la signature, ce qui résout les problèmes de sécurité précédents et permet la mise en cache de contenus sur le réseau. En règle générale, lorsqu'une nouvelle technologie permet d'atténuer les menaces de sécurité existantes, elle en crée malheureusement de nouvelles. Par exemple, NDN peut lutter contre l'usurpation d'identité et la falsification, mais simultanément, NDN crée plusieurs nouvelles menaces pour la sécurité, telles que les attaques IFA (Interest Flooding Attack) et les attaques CPA (Content Poisoning Attack). Ce sont deux menaces sont bien connues et des contre-mesures ont été proposées dans [9, 10].

1.2 La fuite d'informations dans les NDN

À lheure actuelle, les fuites dinformations provoquées par une attaque ciblée posent de graves problèmes à de nombreuses entreprises, ce qui a un impact considérable sur leurs bénéfices et leur rentabilité [11]. Par exemple, en 2013, la chaîne de distribution "Target" a subi une chute de 46% de ses bénéfices après l'attaque et a dépensé plus de 100 millions de dollars pour la mise à niveau du système afin d'empêcher une autre attaque [12]. Afin de réussir la fuite d'informations, un attaquant contacte d'abord un employé de la société ciblée, par exemple grâce à un message mail contenant un malware. Ce message ressemble à un email légitime, et l'employé ouvre celui-ci, ce qui permet d'infecter son poste avec le malware. Le logiciel malveillant peut alors créer un canal de communication avec l'attaquant externe. Lattaquant peut ainsi contrôler le logiciel malveillant à distance et finalement voler des informations confidentielles. Si on considère que le volume du trafic HTTPS est important, il est alors difficile pour le pare-feu de la société dinspecter et de détecter le canal malveillant. Étant donné que cette menace pour la sécurité résulte de l'accès à des médias suspects, l'une des contre-mesures serait de fournir aux employés une éducation en matière de cybersécurité [13]. Il est cependant presque impossible d'éliminer complètement toutes les erreurs humaines. Il devient donc essentiel dexaminer tous les moyens possibles pour empêcher les fuites dinformations après linfection dun logiciel malveillant par un ordinateur, en supposant donc quil est difficile pour lordinateur déviter linfection.

Étant donné que la fuite dinformations est donc un problème important de sécurité, il est absolument essentiel dévaluer le risque concernant ce type d'attaque sur NDN avant de remplacer complètement Internet par NDN. C'est un des premiers objectifs de cette thèse d'examiner si une nouvelle menace à la sécurité causant la fuite dinformations se produit dans NDN. En supposant que (i) un ordinateur de l'entreprise se trouve sur le réseau basé sur une architecture NDN, (ii) l'ordinateur a déjà été compromis par un support suspect tel qu'un courrier électronique malveillant, et (iii) la société a installé un pare-feu au-dessus de NDN, la thèse évalue la situation dans laquelle lordinateur infecté tente denvoyer des données à lattaquant externe. À la base, les architectures NDN sont fondamentalement des architectures de type ńpull-based architectureż et il nexiste que deux types de paquets: *Interest* et *Data*, qui sont respectivement un paquet de demande et un paquet de réponse. Pour récupérer le contenu, un consommateur envoie d'abord un paquet d'intérêt au réseau NDN, et il obtient les données correspondantes à partir du producteur ou du nud NDN intermédiaire. En d'autres termes, ces nuds ne peuvent pas envoyer de données à moins de recevoir la demande d'intérêt. Par conséquent, une des méthodes naïves

pour limiter la fuite dinformations serait de proposer un pare-feu qui inspecte soigneusement les données à publier et à produire. Dans ce cas, tout le contenu accessible au public se trouve sur le pare-feu, que ce soit les originaux ou les caches. Toutefois, le pare-feu ne gère pas de stratégie de nommage sur le contenu externe et les nuds NDN intermédiaires ne vérifient pas si le nom de la ressource existe réellement. Cela entraîne un risque de fuite d'informations via un paquet d'intérêt, car le logiciel malveillant peut masquer des informations telles que les informations du client dans le nom de l'intérêt et les envoyer à l'attaquant externe. Le logiciel malveillant peut alors prétendre accéder à du contenu extérieur, de sorte qu'il est assez difficile pour le pare-feu de détecter l'attaque. Cette thèse soutient que l'attaque par fuite d'informations via un paquet d'intérêt dans NDN pourrait devenir l'une des attaques de sécurité importante au niveau du protocole et qu'il est donc important de développer la méthode de détection de cette attaque.

1.3 Énoncé de la thèse

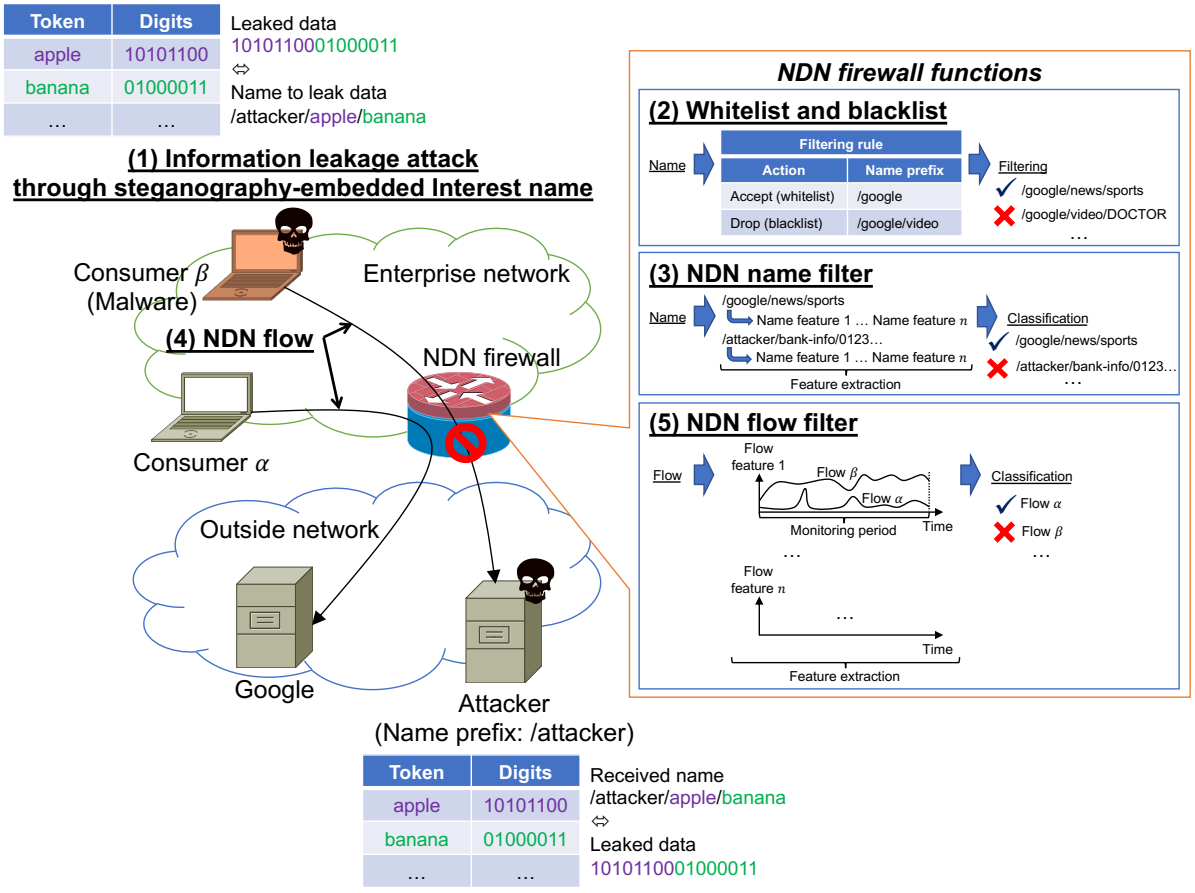


Figure 1: Description des cinq contributions.

Les contributions de cette thèse sont au nombre de cinq (Fig. 1). Tout d'abord, ***cette thèse propose la modélisation d'une attaque par fuite d'informations via une donnée et un paquet d'intérêt pour NDN***. Cette thèse étudie en détail l'attaque qui consiste à utiliser un paquet d'intérêt et permettant à lattaquant de cacher son activité malveillante, et elle propose une

méthode utilisant la stéganographie pour effectuer de manière efficace la fuite dinformations. Par exemple, en supposant que l'attaquant et le logiciel malveillant partagent la même table pour la stéganographie, et que le programme malveillant connaît le préfixe du nom de l'attaquant "`/attacker`", le logiciel malveillant peut convertir les données divulguées "1010110001000011" en nom d'intérêt "`/attacker/apple/banana`" à l'aide du tableau de stéganographie. Lattaquant eput ensuite décrypter le nom reçu à laide de la table partagée et obtient les données divulguées "1010110001000011". À la connaissance de l'auteur, il s'agit de la première recherche sur l'attaque par fuite d'informations dans les réseaux de type NDN.

Deuxièmement, afin de remédier à lattaque par fuite dinformations précédente, ***cette thèse propose un pare-feu NDN qui surveille et traite le trafic NDN provenant des consommateurs disposant d'une liste blanche et d'une liste noire***. Pour concevoir le pare-feu, la thèse propose une solution qui s'articule autour de deux impératifs: (i) concevoir un pare-feu NDN indépendant du démon de transfert NDN (NFD – NDN Forwarding Daemon) [14], qui décide comment transférer un intérêt, et (ii) effectuer une recherche rapide des noms ou des préfixes de nom dans la liste blanche et la liste noire. En utilisant un filtre Cuckoo filter [15], qui est un filtre probabiliste, le pare-feu NDN proposé fournit un filtrage des paquets dintérêt basé sur les noms ou les préfixes de nom des listes pouvant être mises à jour à la volée. Par exemple, lorsque l'opérateur de réseau de l'entreprise autorise le consommateur à utiliser tous les services de Google, à l'exception du service vidéo, il peut ajouter les règles "accept `/google`" et "drop `/google/video`" dans le pare-feu NDN, et ainsi, "`/google/news/sports`" doit être accepté alors que "`/google/video/DOCTOR`" doit être rejeté. La mise en uvre du pare-feu proposé atteint des performances élevées. Plus précisément, la dégradation du débit avec le pare-feu actif nest que de 0.912 à 2.34%, ce qui sera acceptable dans un réseau dentreprise.

Troisièmement, ***cette thèse propose un filtre de nom NDN pour classifier un nom dans le paquet dintérêt comme légitime ou non***. Comme les réseaux de type NDN n'ont pas encore été déployés à grande échelle, il n'existe pas de jeu de données sur le trafic NDN. En supposant qu'il existe une forte probabilité que la future politique de nommage dans NDN soit basée sur la stratégie de nommage des URL actuellement utilisée, cette thèse utilise des noms de contenu basés sur les URL collectées par un robot d'exploration Web pour générer et évaluer le filtre de nom proposé. Par exemple, en extrayant plusieurs noms, le filtre de noms proposé juge le nom "`/google/news/sports`" comme légitime et le nom "`/attacker/bank-info/0123...`" comme anormal. En utilisant les informations disponibles et en appliquant à l'ensemble de données un algorithme non supervisé de type Isolation forest [16] pour isoler les cas problématiques, cette thèse crée des filtres de nom NDN. Nous avons évalué les performances des filtres de nom et montré que les filtres proposés peuvent considérablement gêner le débit de fuite d'informations par intérêt. Par exemple, les logiciels malveillants doivent envoyer 137 fois plus de paquets d'intérêt pour faire fuiter les informations que dans un contexte non filtré.

Le filtre de noms peut donc réduire le débit par intérêt, mais pour améliorer leur attaque, les logiciels malveillants peuvent envoyer de nombreux intérêts en très peu de temps. Le malware peut également exploiter un intérêt avec un contenu spécifique dans le nom (comme peut le faire un message HTTP POST sur Internet), ce qui met en défaut le filtre de nom proposé précédemment et peut augmenter le débit de fuite d'informations. C'est une des limitations du filtre de nom. Pour prendre en compte le flux de trafic entre le pare-feu NDN et le consommateur, ***la thèse propose comme quatrième contribution un pare-feu NDN qui surveille les flux NDN***. Dans un premier temps, nous introduisons le concept de flux NDN et nous le définissons de manière stricte, ce qui na pas encore été fait dans le domaine des NDN [17]. Ensuite, nous proposons une méthode pour générer un jeu de données de flux NDN dérivé d'un jeu de données de flux HTTP puisqu'il nexiste à l'heure actuelle aucun jeu de données sur le

trafic NDN.

Cinquièmement, pour remédier aux inconvénients du filtre de noms NDN, ***cette thèse propose un filtre de flux NDN permettant de classer un flux comme légitime ou non***. Sur la base du jeu de données de flux NDN généré, nous avons créé un filtre de flux NDN contre l'attaque par fuite d'informations. Par exemple, en surveillant le flux NDN sur une période définie et en extrayant plusieurs entités de flux, le filtre de flux proposé est capable d'identifier le flux α comme légitime et le flux β comme anormal. En appliquant le jeu de données obtenu à une machine à vecteurs de support (SVM) [18], l'évaluation des performances montre que le débit de fuite d'informations obstrué par le filtre de flux est compris entre $1.32 \cdot 10^{-2}$ à $6.47 \cdot 10^{-2}$ par rapport au débit basé uniquement sur le filtre de noms. Ainsi, le filtre de flux complète de manière intéressante le filtre de nom et réduit considérablement le débit de fuite dinformations.

1.4 Organisation de la thèse

La suite de cette thèse est organisée comme suit.

Chapter 2 explique les travaux liés à cette thèse et soutient enfin qu'une attaque par fuite d'informations via un nom d'intérêt pourrait être l'une des attaques de sécurité principales au niveau du protocole, telles que les attaques par inondation d'intérêts et par empoisonnement du contenu.

Chapter 3 étudie en profondeur plusieurs modèles dattaque par fuite dinformations dans les réseaux de type NDN. Afin d'évaluer le risque de l'attaque, cette thèse se concentre sur les modèles.

Chapter 4 propose un pare-feu NDN qui surveille et traite le trafic NDN provenant des consommateurs avec gestion de liste blanche et liste noire. C'est la première étape pour aborder l'attaque par fuite d'informations dans cette thèse.

Chapter 5 propose un filtre de nom NDN pour classifier un nom présent dans l'intérêt comme légitime ou non. Ce filtre de nom est lune des applications du pare-feu.

Chapter 6 propose un flux NDN surveillé sur un pare-feu NDN. Cette thèse montre une méthode pour générer le jeu de données de flux NDN correspondant à partir d'un jeu de données de flux HTTP et utilise le jeu de données créé pour créer un filtre plus sophistiqué complétant le filtre de nom NDN du chapitre précédent.

Chapter 7 propose un filtre de flux NDN pour classifier un flux comme légitime ou non. Ce filtre de flux traite les inconvénients du filtre de nom et le filtre de flux fait également partie des applications du pare-feu.

Chapter 8 résume cette thèse et montre les perspectives de ce domaine de recherche.

2 Conclusion générale

Internet a été largement utilisé pour fournir de nombreuses applications basées sur un paradigme de communication hôte à hôte. Toutefois, compte tenu de la tendance actuelle des applications telles que les services de vidéo à la demande, que lon peut qualifier dapplications orientées hôte à contenu, le paradigme de communication classique hôte à hôte pourrait ne pas être optimal. Pour fournir que ce genre applications soient plus eficaces, lune des possibilités serait de passer dun paradigme de communication hôte à hôte à un paradigme hôte vers contenu. De plus, d'un point de vue sécurité, Internet n'a pas été conçu pour supporter des problèmes de sécurité tels que l'usurpation d'identité et des falsifications, de sorte qu'il faut résoudre ces problèmes au niveau de la couche application, avec par exemple la prise en charge du protocole SSL/TLS. En utilisant un tel protocole, HTTPS peut éliminer les problèmes. Cependant, en adoptant une

méthode de chiffrement hôte à hôte, il invalide la possibilité de mettre en cache les contenus et dégradant ainsi l'efficacité de livraison de ces contenus. De plus, en utilisant la méthode de chiffrement, un attaquant peut établir un canal de communication chiffré et éviter ainsi dêtre détecté par un pare-feu. Parmi les scénarios d'attaque potentiels, un attaquant peut exploiter le canal caché pour générer des fuites d'informations, ce qui constitue l'un des gros problèmes d'Internet. Ainsi, la conception dune architecture de réseau sécurisée est également une exigence importante.

Ces dernières années, les réseaux de type NDN ont été proposés comme lune des architectures de réseau les plus prometteuses, passant du paradigme de communication hôte à hôte au paradigme hôte vers contenu. De plus, NDN adopte une approche de sécurité dès la conception et permet ainsi de lutter contre plusieurs attaques de sécurité existantes sur Internet. Même si NDN peut satisfaire à certaines exigences de sécurité, il reste des attaques possibles, et une attaque particulière repose sur un canal de communication malveillant causant une fuite dinformations dans le NDN. Il est primordial d'évaluer un tel risque de sécurité avant de vouloir remplacer complètement Internet par NDN.

Cette thèse examine si une nouvelle menace de sécurité causant la fuite d'informations peut se produire dans le NDN. En supposant que (i) un ordinateur de l'entreprise se trouve sur le réseau basé sur une architecture NDN, (ii) l'ordinateur a déjà été compromis par un support suspect tel qu'un courrier électronique malveillant, et (iii) la société a installé un pare-feu au-dessus de NDN, la thèse évalue la situation dans laquelle lordinateur infecté tente denvoyer des données à lattaquant externe.

À la base, les architectures NDN sont fondamentalement des architectures de type ńpull-based architectureż et il nexiste que deux types de paquets: *Interest* et *Data*, qui sont respectivement un paquet de demande et un paquet de réponse. Pour récupérer le contenu, un consommateur envoie d'abord un paquet d'intérêt au réseau NDN, et il obtient les données correspondantes à partir du producteur ou du nud NDN intermédiaire. En d'autres termes, ces nuds ne peuvent pas envoyer de données à moins de recevoir la demande d'intérêt. Par conséquent, une des méthodes naïves pour limiter la fuite dinformations serait de proposer un pare-feu qui inspecte soigneusement les données à publier et à produire.

Cependant, le pare-feu ne peut pas gérer de stratégie de nommage sur le contenu externe et les nuds de transfert NDN ne vérifient pas si le nom existe réellement. Cela entraîne un risque de fuite d'informations via un paquet d'intérêt, en dissimulant des informations telles que les informations du client dans le nom de l'intérêt et en les envoyant à l'attaquant externe. Le logiciel malveillant peut prétendre accéder à du contenu extérieur, de sorte qu'il est assez difficile pour le pare-feu de détecter l'attaque. Cette thèse soutient que l'attaque par fuite d'informations via un intérêt dans NDN devrait être l'une des attaques de sécurité essentielles au niveau du protocole et qu'il est important de développer la méthode de détection de cette attaque.

2.1 Contributions

Les cinq contributions de cette thèse sont les suivantes.

Premièrement, nous avons avons proposé la modélisation d'une attaque par fuite d'informations via une donnée et un paquet d'intérêt pour NDN. Elle détaille l'attaque qui consiste à utiliser un paquet d'intérêt et permettant à lattaquant de cacher son activité malveillante, et elle a proposé une méthode utilisant la stéganographie pour effectuer de manière efficace la fuite dinformations. À la connaissance de l'auteur, il s'agit de la première recherche sur l'attaque par fuite d'informations dans les réseaux de type NDN.

Deuxièmement, afin de remédier à lattaque par fuite dinformations précédente, nous pro-

posons un pare-feu NDN qui surveille et traite le trafic NDN provenant des consommateurs disposant d'une liste blanche et d'une liste noire. Pour concevoir le pare-feu, la thèse propose une solution qui s'articule autour de deux impératifs: (i) concevoir un pare-feu NDN indépendant du démon de transfert NDN (NFD – NDN Forwarding Daemon), qui décide comment transférer un intérêt, et (ii) effectuer une recherche rapide des noms ou des préfixes de nom dans la liste blanche et la liste noire. En utilisant un filtre Cuckoo filter, qui est un filtre probabiliste, le pare-feu NDN proposé fournit un filtrage des paquets dintérêt basé sur les noms ou les préfixes de nom des listes pouvant être mises à jour à la volée. La mise en uvre du pare-feu proposé atteint des performances élevées (le code source est disponible sur github [78]). L'évaluation des performances a montré que la dégradation du débit avec le pare-feu actif nest que de 0.912 à 2.34%, ce qui est acceptable dans un réseau dentreprise.

Troisièmement, nous avons élaboré un filtre de nom NDN pour classifier un nom dans le paquet dintérêt comme légitime ou non. Comme les réseaux de type NDN n'ont pas encore été déployés à grande échelle, il n'existe pas de jeu de données sur le trafic NDN. En supposant qu'il existe une forte probabilité que la future politique de nommage dans NDN soit basée sur la stratégie de nommage des URL actuellement utilisée, cette thèse utilise des noms de contenu basés sur les URL collectées par un robot d'exploration Web pour générer et évaluer le filtre de nom proposé. En utilisant les informations disponibles et en appliquant à l'ensemble de données un algorithme non supervisé de type Isolation forest pour isoler les cas problématiques, nous avons créé des filtres de nom NDN. Nous avons évalué les performances des filtres de nom et montré que les filtres proposés peuvent considérablement gêner le débit de fuite d'informations par intérêt. Par exemple, les logiciels malveillants doivent envoyer 137 fois plus de paquets d'intérêt pour faire fuiter les informations que dans un contexte non filtré. C'est la première approche contre la fuite d'informations via un paquet d'intérêt dans NDN.

Un filtre de noms pourrait donc réduire le débit par intérêt, mais pour améliorer leur attaque, les logiciels malveillants pourraient alors envoyer de nombreux intérêts en très peu de temps. Le malware peut également exploiter un intérêt avec un contenu spécifique dans le nom (comme peut le faire un message HTTP POST sur Internet), ce qui met en défaut le filtre de nom proposé précédemment et peut augmenter le débit de fuite d'informations. C'est une des limitations du filtre de nom. Pour prendre en compte le flux de trafic entre le pare-feu NDN et le consommateur, nous avons proposé comme quatrième contribution un pare-feu NDN qui surveille les flux NDN. Dans un premier temps, nous avons défini le concept de flux NDN, ce qui na pas encore été fait dans le domaine des NDN. Ensuite, nous avons défini une méthode pour générer un jeu de données de flux NDN dérivé d'un jeu de données de flux HTTP puisqu'il nexiste à l'heure actuelle aucun jeu de données sur le trafic NDN. Les statistiques obtenues pourront être utiles non seulement pour une analyse des risques de fuite dinformations sur laquelle porte cette thèse, mais également pour comme paramètres de simulation d'un modèle daccès au contenu.

Cinquièmement, pour remédier aux inconvénients du filtre de noms NDN, nous avons présenté un filtre de flux NDN permettant de classer un flux comme légitime ou non. Sur la base du jeu de données de flux NDN généré, nous avons créé un filtre de flux NDN contre l'attaque par fuite d'informations. En appliquant le jeu de données obtenu à une machine à vecteurs de support (SVM), l'évaluation des performances montre que le débit de fuite d'informations obstrué par le filtre de flux est compris entre $1.32 \cdot 10^{-2}$ à $6.47 \cdot 10^{-2}$ par rapport au débit basé uniquement sur le filtre de noms. Ainsi, le filtre de flux complète de manière intéressante le filtre de nom et réduit considérablement le débit de fuite dinformations.

2.2 Perspectives

Le débit de fuite d'informations entravé par le filtre de nom et le filtre de flux peut être réduit à 23.2 Kbps. Cela signifie qu'en imitant le comportement d'un utilisateur légitime, il est possible que les logiciels malveillants fassent fuiter les données d'une entreprise très lentement, pour imiter le comportement normal. Dans le cas où le logiciel malveillant pourrait être présent mais non découvert dans l'entreprise et que l'attaquant ne se préoccuperait pas du faible débit, il pourrait alors récupérer n'importe quel type de fichier volumineux après une longue attente. En effet, ce genre de cas se produit sur Internet et cette attaque s'appelle APT (Advanced Persistent Threat) [115], qui est un type d'attaque ciblée. Lune des caractéristiques de lAPT est lapproche ńfaible et lenteż pour éviter la détection. Par conséquent, compte tenu non seulement de l'attaque ciblée, mais également de l'APT, outre le nom et le filtre de flux proposés, plusieurs contre-mesures devraient être proposées. En tant que perspectives futures dans ce domaine de recherche, nous envisageons plusieurs contre-mesures prometteuses ci-dessous.

- Développer un logiciel anti-malware

  Habituellement, dans une entreprise, chaque ordinateur installe un logiciel anti-malware afin d'éviter toute infection par un logiciel malveillant. Même si l'ordinateur est infecté par un logiciel malveillant, une fois que les informations sur le logiciel malveillant sont ajoutées à la base de données du logiciel anti-malware, le logiciel malveillant peut être exclu de l'ordinateur. Toutefois, une fois l'attaque réussie, l'attaquant peut envoyer envoie un sous-programme malveillant à l'aide d'un canal caché, et peut donc ainsi contourner le pare-feu en permanence. En particulier, dans une attaque de type Advanced Persistent Threat, les logiciels malveillants sont personnalisés pour exécuter l'attaque, de sorte qu'il est extrêmement difficile pour le pare-feu de les détecter. Ainsi, cette solution pourrait ne pas être aussi efficace.

- Filtrage des noms d'intérêts en fonction de la liste blanche

  Un pare-feu d'entreprise peut gérer une stratégie d'accès au contenu dans l'entreprise en définissant des préfixes de nom acceptables ou des noms, puis en les ajoutant à la liste blanche. En particulier, la manipulation des contenus des paquets d'intérêts doit être efficace contre les fuites dinformations, car les contenus des paquets d'intérêts peuvent contenir beaucoup plus dinformations que les déclarations des paquets d'intérêts. Ainsi, l'entreprise peut atténuer le risque d'attaque par fuite de données par un filtrage basé sur une liste blanche et la surcharge du filtrage basé sur une liste blanche n'est pas importante (voir Chapter 4). Cependant, la flexibilité de l'accès au contenu des employés est limitée.

- Application d'une analyse chronologique à un filtre de flux NDN

  Le filtre de flux NDN proposé dans Chapter 7 définit une fenêtre de flux et détermine si une fenêtre de flux est légitime séquentiellement. En d'autres termes, le filtre ne dispose pas d'un mécanisme permettant de mémoriser les informations qu'il a vues par le passé. Une des autres approches prometteuses consiste à exploiter une analyse de séries chronologiques réalisée, par exemple, par des RNN (Recurrent Neural Network). Les RNN ont été conçus pour traiter les données temporelles et ils seraient utiles pour pouvoir détecter un flux anormal.

- Analyse des occurrences dans la mémoire cache

  Après avoir collecté des données, le logiciel malveillant génère des intérêts malveillants, puis les envoie à l'attaquant externe. Ainsi, les paquets de données correspondants ne sont

pas mis en cache dans les nuds NDN. Lorsque le logiciel malveillant envoie uniquement les intérêts malveillants, les échecs de cache successifs de l'ordinateur infecté doivent se produire. Cette solution peut également être utilisée pour détecter le tunnel DNS, et à la connaissance de l'auteur, aucune recherche n'exploite une fonctionnalité de hit and miss du cache DNS. Ainsi, la solution pourrait être utile à la fois sur NDN et sur Internet.

List of Publications

Journal

- G. Doyen, T. Cholez, W. Mallouli, B. Mathieu, H. L. Mai, X. Marchal, D. Kondo, M. Aouadj, A. Ploix, E. M. de Oca, and O. Festor, "An Orchestrated NDN Virtual Infrastructure transporting Web Traffic: Design, Implementation and First Experiments with Real End-Users," *IEEE Communication Magazine* (under review).

- D. Kondo, T. Silverston, V. Vassiliades, H. Tode, and T. Asami, "Name Filter: A Countermeasure against Information Leakage Attacks in Named Data Networking," *IEEE Access*, vol. 6, pp. 65151–65170, Oct. 2018.

International Conferences

- H. L. Mai, M. Aouadj, G. Doyen, D. Kondo, X. Marchal, T. Cholez, E. M. de Oca, and W. Mallouli, "Implementation of Content Poisoning Attack Detection and Reaction in Virtualized NDN Networks," in *ICIN* 2018.

- D. Kondo, T. Silverston, H. Tode, T. Asami, and O. Perrin, "Risk Analysis of Information-Leakage through Interest Packets in NDN," in *IEEE INFOCOM Workshop (NOM)* 2017.

- D. Kondo, T. Silverston, H. Tode, T. Asami, and O. Perrin, "Name Anomaly Detection for ICN," in *IEEE LANMAN* 2016.

- D. Kondo, H. Lee, and A. Nakao, "Content Piece Rarity Aware In-network Caching for BitTorrent," in *IEEE GLOBECOM* 2015.

- D. Kondo, Y. Hirota, A. Fujimoto, H. Tode, and K. Murakami, "P2P Live Streaming System for Multi-view Video with Fast Switching," in *Networks* 2014.

- D. Kondo, A. Fujimoto, Y. Hirota, H. Tode, and K. Murakami, "P2P Live Streaming Distribution System with Fast Content Switching and Category-Based Recommendation Function," in *3PGCIC Workshop (SMDMS)* 2012.

National Conferences (in Japan)

- H. Tode, D. Kondo, T. Silverston, and T. Asami, "Filter Design against Information-Leakage via NDN Interest Packets," in *IEICE General Conference* 2018.

- D. Kondo, H. Tode, and T. Asami, "Generation of NDN Flow Dataset from Waikato HTTP Flow Dataset and its Statistics," in *Technical Report of IEICE* 2018. **(Encouragement Talk)**

- D. Kondo, T. Silverston, H. Tode, T. Asami, and O. Perrin, "Per-Packet Throughput of Information-Leakage through Content Names in NDN Interest Packets," in *Technical Report of IEICE* 2017.
- D. Kondo, T. Silverston, H. Tode, T. Asami, and O. Perrin, "User-Driven Privacy Management Framework for Content Access," in *Technical Report of IEICE* 2017.
- D. Kondo, T. Silverston, H. Tode, and T. Asami, "NDN Information Leakage Prevention based on Statistical Information of URL Data Sampled from the Internet," *IEICE Technical Committee on ICN* 2016.
- D. Kondo, T. Silverston, H. Tode, and T. Asami, "Requirement Specifications for ICN Firewall," in *IEICE General Conference* 2016.
- D. Kondo, T. Silverston, H. Tode, and T. Asami, "On Names in ICN from URL Data Sampled in the Internet," in *IEICE General Conference* 2016.
- D. Kondo, T. Silverston, H. Tode, and T. Asami, "Design of Interest Packet Filtering for Information Leakage Prevention based on URL Data Sampled from the Internet," in *Technical Report of IEICE* 2016.
- D. Kondo, and A. Nakao, "Caching Algorithm using Information of Rare Pieces for BitTorrent," in *Technical Report of IEICE* 2015. **(Encouragement Talk)**
- D. Kondo, and A. Nakao, "In-network Detection of Rare Content Pieces for BitTorrent Swarm," in *Technical Report of IEICE* 2014.
- D. Kondo, A. Fujimoto, Y. Hirota, H. Tode, and K. Murakami, "P2P Live Streaming System for Multi-View Video with Fast Switching," in *Technical Report of IEICE* 2013.
- D. Kondo, A. Fujimoto, Y. Hirota, H. Tode, and K. Murakami, "P2P Live Streaming System with Fast Channel Switching and Category-based Recommendation Functions," in *the IEICE General Conference* 2012.

Bibliography

[1] The Zettabyte Era: Trends and Analysis, https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.pdf

[2] RFC 7230, https://tools.ietf.org/html/rfc7230

[3] RFC 1808, https://tools.ietf.org/html/rfc1808

[4] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A Survey of Information-Centric Networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, Jul. 2012.

[5] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, k. claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking," *ACM SIGCOMM CCR*, vol. 44, no. 3, pp. 66–73, Jul. 2014.

[6] RFC 2818, https://tools.ietf.org/html/rfc2818

[7] RFC 6101, https://tools.ietf.org/html/rfc6101

[8] RFC 5246, https://tools.ietf.org/html/rfc5246

[9] A. Afanasyev, P. Mahadevan, I. Moiseenko, E. Uzun, and L. Zhang, "Interest Flooding Attack and Countermeasures in Named Data Networking," in *IFIP Networking* 2013.

[10] C. Ghali, G. Tsudik, and E. Uzun, "Needle in a Haystack: Mitigating Content Poisoning in Named-Data Networking," in *SENT* 2014.

[11] IT Security Risks Survey 2014, https://media.kaspersky.com/en/IT_Security_Risks_Survey_2014_Global_report.pdf

[12] Understanding Targeted Attacks: The Impact of Targeted Attacks, https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/the-impact-of-targeted-attacks

[13] R. Beuran, C. Pham, D. Tang, K. Chinen, Y. Tan, and Y. Shinoda, "CyTrONE: An Integrated Cybersecurity Training Framework," in *ICISSP* 2017.

[14] NFD Developer's Guide, https://named-data.net/wp-content/uploads/2016/10/ndn-0021-7-nfd-developer-guide.pdf

[15] B. Fan, D. G. Andersen, M. Kaminsky, and M. D. Mitzenmacher, "Cuckoo Filter: Practically Better Than Bloom," in *ACM CoNEXT* 2014.

[16] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation Forest," in *IEEE ICDM* 2008.

[17] Information-Centric Networking Research Group (ICNRG), `https://trac.ietf.org/trac/irtf/wiki/icnrg`

[18] C. Cortes and V. Vapnik, "Support-Vector Networks," in *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[19] K. Xu, P. Butler, S. Saha, and D. Yao, "DNS for Massive-Scale Command and Control," *IEEE TDSC*, vol. 10, no. 3, pp. 143–153, 2013.

[20] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs," in *ACM SIGKDD* 2009.

[21] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon, "From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware," in *USENIX Security* 2012.

[22] RFC 793, `https://tools.ietf.org/html/rfc793`

[23] RFC 791, `https://tools.ietf.org/html/rfc791`

[24] RFC 3986, `https://tools.ietf.org/html/rfc3986`

[25] We knew the web was big... (Google Official Blog), `https://googleblog.blogspot.jp/2008/07/we-knew-web-was-big.html`

[26] S. A. Thomas, "SSL and TLS Essentials: Securing the Web," *John Wiley & Sons, Inc.*, 2000.

[27] D. Naylor, K. Schomp, M. Varvello, I. Leontiadis, J. Blackburn, D. R. López, K. Papagiannaki, P. R. Rodriguez, and P. Steenkiste, "Multi-Context TLS (mcTLS): Enabling Secure In-Network Functionality in TLS," in *ACM SIGCOMM* 2015.

[28] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, "BlindBox: Deep Packet Inspection over Encrypted Traffic," in *ACM SIGCOMM* 2015.

[29] W. M. Shbair, T. Cholez, J. Francois, and I. Chrisment, "A Multi-Level Framework to Identify HTTPS Services," in *IEEE/IFIP NOMS* 2016.

[30] D. Fetterly, M. Manasse, and M. Najork, "Spam, Damn Spam, and Statistics: Using Statistical Analysis to Locate Spam Web Pages," in *WebDB* 2004.

[31] A. Blum, B. Wardman, T. Solorio, and G. Warner, "Lexical Feature Based Phishing URL Detection Using Online Learning," in *AISec* 2010.

[32] M. Akiyama, T. Yagi, and M. Itoh, "Searching Structural Neighborhood of Malicious URLs to Improve Blacklisting," in *SAINT* 2011.

[33] M. Khonji, Y. Iraqi, and A. Jones, "Lexical URL Analysis for Discriminating Phishing and Legitimate Websites," in *CEAS* 2011.

[34] M. S. Lin, C. Y. Chiu, Y. J. Lee, and H. K. Pao, "Malicious URL Filtering – A Big Data Application," in *IEEE Big Data* 2013.

[35] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, "Detecting Algorithmically Generated Malicious Domain Names," in *IMC* 2010.

[36] M. Thomas, and A. Mohaisen, "Kindred Domains: Detecting and Clustering Botnet Domains Using DNS Traffic," in *SIMPLEX* 2014.

[37] M. Grill, I. Nikolaev, V. Valeros, and M. Rehak, "Detecting DGA Malware Using NetFlow," in *IFIP/IEEE IM WORKSHOP (CogMan)* 2015.

[38] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, and E. Gerhards-Padilla, "A Comprehensive Measurement Study of Domain Generating Malware," in *USENIX Security* 2016.

[39] T. V. Leijenhorst, K. Chin, and D. Lowe, "On the Viability and Performance of DNS Tunneling," in *ICITA* 2008.

[40] DNScat, `http://tadek.pietraszek.org/projects/DNScat/`

[41] A. Merlo, G. Papaleo, S. Veneziano, and M. Aiello, "A Comparative Performance Evaluation of DNS Tunneling Tools," in *CISIS* 2011.

[42] K. Born, and D. Gustafson, "Detecting DNS Tunnels Using Character Frequency Analysis," `https://arxiv.org/pdf/1004.4358v1.pdf`

[43] C. Qi, X. Chen, C. Xu, J. Shi, and P. Liu, "A Bigram Based Real Time DNS Tunnel Detection Approach," in *ITQM* 2013.

[44] G. Farnham, "Detecting DNS Tunneling," `https://www.sans.org/reading-room/whitepapers/dns/detecting-dns-tunneling-34152`

[45] W. Ellens, P. Żuraniewski, A. Sperotto, H. Schotanus, M. Mandjes, and E. Meeuwissen, "Flow-Based Detection of DNS Tunnels," in *AIMS* 2013.

[46] RFC 3917, `https://tools.ietf.org/html/rfc3917`

[47] A. M. Kara, H. Binsalleeh, M. Mannan, A. Youssef, and M. Debbabi, "Detection of Malicious Payload Distribution Channels in DNS," in *IEEE ICC* 2014

[48] M. Aiello, M. Mongelli, and G. Papaleo, "DNS Tunneling Detection through Statistical Fingerprints of Protocol Messages and Machine Learning," *Int. J. Commun. Syst.*, vol. 28, no. 14, pp. 1987–2002, Jul. 2014.

[49] Morto Worm Sets a (DNS) Record, `https://www.symantec.com/connect/blogs/morto-worm-sets-dns-record`

[50] NDN Packet Format Specification 0.3 documentation, `https://named-data.net/doc/NDN-packet-spec/current/`

[51] NDN Packet Format Specification (Name), `https://named-data.net/doc/NDN-packet-spec/current/name.html`

[52] NDN Technical Memo: Naming Conventions, `https://named-data.net/wp-content/uploads/2014/08/ndn-tr-22-ndn-memo-naming-conventions.pdf`

[53] U. Schnurrenberger, "Comparing Apples to Apples in ICN," in *IEEE CCNC WORKSHOP (FI4D)* 2017.

[54] H. Dai, Y. Wang, J. Fan, and B. Liu, "Mitigate DDoS Attacks in NDN by Interest Traceback," in *IEEE INFOCOM WORKSHOP (NOMEN)* 2013.

[55] A. Compagno, M. Conti, P. Gasti, and G. Tsudik, "Poseidon: Mitigating Interest Flooding DDoS Attacks in Named Data Networking," in *IEEE LCN* 2013.

[56] C. Ghali, G. Tsudik, E. Uzun, and C. A. Wood, "Closing the Floodgate with Stateless Content-Centric Networking," in *ICCCN* 2017.

[57] D. Kim, S. Nam, J. Bi, and I. Yeom, "Efficient Content Verification in Named Data Networking," in *ACM ICN* 2015.

[58] S. DiBenedetto, and C. Papadopoulos, "Mitigating Poisoned Content with Forwarding Strategy," in *IEEE INFOCOM WORKSHOP (NOM)* 2016.

[59] H. L. Mai, T. Nguyen, G. Doyen, R. Cogranne, W. Mallouli, E. M. de Oca, and O. Festor, "Towards a Security Monitoring Plane for Named Data Networking and its Application against Content Poisoning Attack," in *IEEE/IFIP NOMS* 2018.

[60] T. D. Nielsen and F. V. Jensen, "Bayesian Networks and Decision Graphs," *Springer Science & Business Media*, 2009.

[61] RFC 1035, `https://tools.ietf.org/html/rfc1035`

[62] M. Baugher, B. Davie, A. Narayanan, and D. Oran, "Self-Verifying Names for Read-Only Named Data," in *IEEE INFOCOM WORKSHOP (NOMEN)* 2012.

[63] C. Ghali, G. Tsudik, and C. A. Wood, "Network Names in Content-Centric Networking," in *ACM ICN* 2016.

[64] S. DiBenedetto, P. Gasti, G. Tsudik, and E. Uzun, "ANDaNA: Anonymous Named Data Networking Application," in *NDSS* 2012.

[65] R. Tourani, S. Misra, J. Kliewer, S. Ortegel, and T. Mick, "Catch Me If You Can: A Practical Framework to Evade Censorship in Information-Centric Networks," in *ACM ICN* 2015.

[66] J. Kurihara, K. Yokota, and A. Tagami, "A Consumer-Driven Access Control Approach to Censorship Circumvention in Content-Centric Networking," in *ACM ICN* 2016.

[67] D. Goergen, T. Cholez, J. Francois, and T. Engel, "A Semantic Firewall for Content-Centric Networking," in *IFIP/IEEE IM Mini-Conference* 2013.

[68] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking Named Content," in *ACM CoNEXT* 2009.

[69] L. Breiman, "Random Forests," in *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[70] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec. 2000.

[71] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the Support of a High-Dimensional Distribution," *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, Jul. 2001.

[72] M. Luby, "LT Codes," in *IEEE FOCS* 2002.

[73] A. Shokrollahi, "Raptor Codes," *IEEE TIT*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.

[74] J. Mason, S. Small, F. Monrose, and G. MacManus, "English Shellcode," in *CCS 2009*.

[75] C. Ghali, G. Tsudik, and C. A. Wood, "(The Futility of) Data Privacy in Content-Centric Networking," in *WPES* 2016.

[76] Signed Interest, `http://named-data.net/doc/ndn-cxx/current/specs/signed-interest.html`

[77] B. H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.

[78] NDN Firewall, `https://github.com/daishi-kondo/ndnfirewall`

[79] ndn-cxx: NDN C++ library with eXperimental eXtensions, `https://github.com/named-data/ndn-cxx`

[80] NDNperf, `https://github.com/Kanemochi/ndnperf`

[81] Y. Wang, K. He, H. Dai, W. Meng, J. Jiang, B. Liu, and Y. Chen, "Scalable Name Lookup in NDN Using Effective Name Component Encoding," in *IEEE ICDCS* 2012.

[82] H. Yuan, T. Song, and P. Crowley, "Scalable NDN Forwarding: Concepts, Issues and Principles," in *ICCCN* 2012.

[83] Y. Wang, T. Pan, Z. Mi, H. Dai, X. Guo, T. Zhang, B. Liu, and Q. Dong, "NameFilter: Achieving Fast Name Lookup with Low Memory Cost via Applying Two-Stage Bloom Filters," in *IEEE INFOCOM* 2013.

[84] W. Quan, C. Xu, J. Guan, H. Zhang, and L. A. Grieco, "Scalable Name Lookup with Adaptive Prefix Bloom Filter for Named Data Networking," *IEEE Communications Letters* 2014.

[85] Y. Wang, B. Xu, D. Tai, J. Lu, T. Zhang, H. Dai, B. Zhang, and B. Liu, "Fast Name Lookup for Named Data Networking," in *IEEE IWQoS* 2014.

[86] D. Perino, M. Varvello, L. Linguaglossa, R. Laufer, and R. Boislaigue, "Caesar: A Content Router for High-Speed Forwarding on Content Names," in *ANCS* 2014.

[87] H. Yuan and P. Crowley, "Reliably Scalable Name Prefix Lookup," in *ANCS* 2015.

[88] J. Takemasa, Y. Koizumi, and T. Hasegawa, "Toward an Ideal NDN Router on a Commercial Off-The-Shelf Computer," in *ACM ICN* 2017.

[89] Common Crawl, `http://commoncrawl.org/`

[90] URL Fragments and Redirects, `https://blogs.msdn.microsoft.com/ieinternals/2011/05/16/url-fragments-and-redirects/`

[91] B. Hawkins, "Under The Ocean of the Internet - The Deep Web," `https://www.sans.org/reading-room/whitepapers/covert/ocean-internet-deep-web-37012`

[92] Google's search business might not be as water-tight as people think it is, https://www.businessinsider.com.au/the-number-of-people-using-search-engines-is-in-decline-2015-9

[93] Search Engine Optimization (SEO) Starter Guide, https://support.google.com/webmasters/answer/7451184

[94] 15 SEO Best Practices for Structuring URLs, https://moz.com/blog/15-seo-best-practices-for-structuring-urls

[95] RFC 1738, https://tools.ietf.org/html/rfc1738

[96] D. Kondo, T. Silverston, H. Tode, T. Asami, and O. Perrin, "Risk Analysis of Information-Leakage through Interest Packets in NDN," in *IEEE INFOCOM WORKSHOP (NOM)* 2017.

[97] G. A. Miller, "WordNet: A Lexical Database for English," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, Nov. 1995.

[98] F. Pedregosa, et al. "Scikit-learn: Machine Learning in Python," *Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[99] ITU-T, https://www.itu.int/en/ITU-T/publications/Pages/latest.aspx

[100] ndn-cxx (tlv.hpp), https://github.com/named-data/ndn-cxx/blob/master/src/encoding/tlv.hpp

[101] C. D. Manning and H. Schütze, "Foundations of Statistical Natural Language Processing," *The MIT Press*, 1999.

[102] WITS: Waikato VIII, https://wand.net.nz/wits/waikato/8/

[103] L. Popa, A. Ghodsi, and I. Stoica, "HTTP As the Narrow Waist of the Future Internet," in *HotNets* 2010.

[104] H. Yuan and P. Crowley, "Experimental Evaluation of Content Distribution with NDN and HTTP," in *IEEE INFOCOM* 2013

[105] X. Marchal, M. E. Aoun, B. Mathieu, W. Mallouli, T. Cholez, G. Doyen, P. Truong, A. Ploix, and E. M. de Oca, "A Virtualized and Monitored NDN Infrastructure Featuring a NDN/HTTP Gateway," in *ACM ICN* 2016.

[106] H. Yuan and P. Crowley, "Scalable Pending Interest Table Design: From Principles to Practice," in *IEEE INFOCOM* 2014.

[107] Flow Classification in Information Centric Networking, https://tools.ietf.org/html/draft-moiseenko-icnrg-flowclass-02

[108] Fetching Content in Named Data Networking with Embedded Manifests, https://named-data.net/wp-content/uploads/2014/09/ndn-tr-25-manifest-embedding.pdf

[109] K. Schneider, C. Yi, B. Zhang, and L. Zhang, "A Practical Congestion Control Scheme for Named Data Networking," in *ACM ICN* 2016.

[110] HTTP Pipelining - Big in Mobile, http://www.guypo.com/http-pipelining-big-in-mobile/

[111] StatCounter, http://gs.statcounter.com/

[112] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, M. Munafò, K. Papagiannaki, and P. Steenkiste, "The Cost of the "S" in HTTPS," in *ACM CoNEXT* 2014.

[113] pkt2flow, https://github.com/caesar0301/pkt2flow

[114] S. Haykin, "Neural Networks and Learning Machines," *Pearson*, Upper Saddle River, NJ, USA, 2008.

[115] Advanced Persistent Threats: A Symantec Perspective, https://www.symantec.com/content/en/us/enterprise/white_papers/b-advanced_persistent_threats_WP_21215957.en-us.pdf