Risk analysis of information-leakage through interest packets in NDN

Daishi Kondo, Thomas Silverston, Hideki Tode, Tohru Asami, Olivier Perrin

▶ **To cite this version:**

Daishi Kondo, Thomas Silverston, Hideki Tode, Tohru Asami, Olivier Perrin. Risk analysis of information-leakage through interest packets in NDN. INFOCOM WKSHPS 2017 - IEEE International Conference on Computer Communications, May 2017, Atlanta, United States. hal-01946257

HAL Id: hal-01946257
https://hal.science/hal-01946257

Submitted on 5 Dec 2018

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

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

Risk Analysis of Information-Leakage Through Interest Packets in NDN

Daishi Kondo*†, Thomas Silverston‡, Hideki Tode§, Tohru Asami¶ and Olivier Perrin*†
*University of Lorraine, LORIA (CNRS UMR 7503), France
†Inria Nancy - Grand Est, France
‡NICT, National Institute of Information and Communications Technology, Tokyo, Japan
§Graduate School of Engineering, Osaka Prefecture University, Japan
¶Graduate School of Information Science and Technology, The University of Tokyo, Japan
Email: {daishi.kondo, olivier.perrin}@loria.fr,
thomas@nict.go.jp, tode@cs.osakafu-u.ac.jp, asami@akg.t.u-tokyo.ac.jp

***Abstract*—Information-leakage is one of the most important security issues in the current Internet. In Named-Data Networking (NDN), Interest names introduce novel vulnerabilities that can be exploited. By setting up a malware, Interest names can be used to encode critical information (*steganography embedded*) and to leak information out of the network by generating anomalous Interest traffic. This security threat based on Interest names does not exist in IP network, and it is essential to solve this issue to secure the NDN architecture. This paper performs risk analysis of information-leakage in NDN. We first describe vulnerabilities with Interest names and, as countermeasures, we propose a name-based filter using search engine information, and another filter using one-class Support Vector Machine (SVM). We collected URLs from the data repository provided by Common Crawl and we evaluate the performances of our per-packet filters. We show that our filters can choke drastically the throughput of information-leakage, which makes it easier to detect anomalous Interest traffic. It is therefore possible to mitigate information-leakage in NDN network and it is a strong incentive for future deployment of this architecture at the Internet scale.**

I. Introduction

Information-leakage is one of the main security threats for companies in the Internet and it is mostly the result of Targeted Attacks as reported in several security reports [1] or newspapers [2]. Targeted Attacks usually come from a lack of vigilance from corporate network users, where an attacker succeeds to access the network by setting up a malware (e.g., via emails). The attacker can then access internal resources and leak crucial information (e.g., customer list, bank information). Targeted Attacks can have an important impact on the company business and damage its reputation (e.g., Sony, Target, etc.). As Named-Data Networking (NDN) [3] is an emerging architecture proposed for the future Internet, it is essential to investigate potential security threats with information-leakage.

Thus, the main motivation of this paper is to perform risk analysis of information-leakage in NDN for a company network. In this context, this paper will therefore discuss proper security attacks and protection models. Differently from IP address field in IP network, the length of names in NDN is much longer and variable. In addition, NDN forwarding nodes do not verify if names really exist or not. Thus, attackers can exploit these vulnerabilities and perform information-leakage using Interest packets with anomalous content names [4]. Preventing information-leakage in NDN network is essential to secure its deployment.

This paper shows that a properly configured NDN enterprise network can greatly reduce information-leakage. We first present vulnerabilities with Interest packets and we show that Interest names can be exploited to encode information (e.g., *steganography embedded*) and to leak information out of the network by generating anomalous Interest traffic. Then, as countermeasures, we propose a name-based per-packet filter using search engine information, by which Interest names are considered legitimate if highly ranked. For the case that this filter cannot predict if unreferenced names are legitimate or anomalous, we propose another filter using one-class Support Vector Machine (SVM).

We evaluate the performances of our filters by collecting URLs from the data repository provided by Common Crawl [5]. We show that our filters can choke drastically the throughput of information-leakage, which makes it easier to detect anomalous Interest traffic. Malwares have to send 264 times more Interest packets to leak information than without using our filters, and it reduces drastically the speed of this threat. Then, network administrators can detect these incidents and react accordingly to prevent information-leakage. By using our filter, NDN can therefore be more resilient to information-leakage and it is a strong incentive for future deployment of this architecture.

The reminder of this paper is organized as follows. Section II describes attack model, while Section III presents our countermeasures. Section IV provides details on our data set, and Section V evaluates the performances of the attack and countermeasures. Section VI surveys related work about information-leakage in the Internet. Section VII finally concludes the paper and presents future perspectives.

II. Attack by Steganography-Embedded Interest

A. Information-Leakage through Interest Packets in NDN

NDN is basically a "Pull"-based architecture, in which a user can reply with a Data packet only if it has received the Interest packet for the content name. Thus, information-leakage through Data packets can be prevented with a properly configured firewall [4]. However, if a computer is compromised by an attacker's malware, it is therefore possible for the malware to use this computer to encode confidential information into the names of Interest packets (i.e., *steganography embedded*) and send these anomalous Interest packets out of the network toward the attacker. In this paper, we consider the case where an enterprise network is based on NDN architecture and connected to NDN Future Internet.

TABLE I: Taxonomy of information-leakage through Interest packets

Features	TYPE I one-way Interest	TYPE II Interest and 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 (bots, etc.)

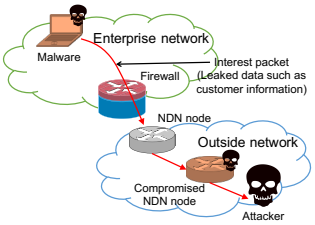

Fig. 1: Type I: One-way Interest.

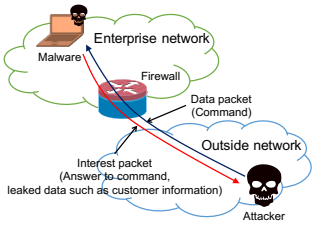

Fig. 2: Type II: Interest/Data.

There are two possible methods to perform information-leakage through Interest packets, referred as "TYPE I" (one-way Interest packets) and "TYPE II" (Interest packets with their corresponding Data packets). These methods are summarized on Table I and illustrated on Figs. 1 and 2.

For TYPE I method (Fig. 1), a malware transmits leaked information out of the enterprise network by sending Interest packets to the attacker. The malware has to know the name prefixes toward compromised NDN nodes (e.g., Wi-Fi AP installed by the attacker) in order to forward Interests to the attacker. In this method, the attacker does not reply with any Data packets to the malware and cannot have a fine-tuned control of the malware. Therefore, in the case that some Interest packets have been lost, the attacker cannot request for a retransmission of missing information. This can happen if a firewall drops packets, or when a PIT is overflowed. The attacker may use erasure coding such as LT codes [6] and Raptor codes [7] to deal with dropped Interest packets. Although this method is not the most efficient for leaked information, the main advantage of the method is to preserve the attacker anonymity because only anomalous Interests packets are received and the attacker does not reply with any Data packets with its own signature.

For TYPE II method (Fig. 2), an attacker controls explicitly the malware and can communicate with it by using Interest and Data packets. Thus, the attacker can remotely control the malware assuming the name prefix of the attacker is routable from the malware. In the case of a packet drop, the attacker can request a retransmission with Data packets and there is no need to use erasure coding. This method is more efficient than the TYPE I method, but once the attack is detected, it is possible for the attacker to be tracked as his signature was included into Data messages. The attacker, however, can control bots remotely and avoid being tracked.

Note that in this paper we consider only Name element of the Interest packet format for information-leakage and we keep for future work other elements such as Selectors or Nonce [8].

B. Encoding Information into Names with Steganography

(a) `<scheme>://<net_loc>/<path>;<params>?<query>#<fragment>`

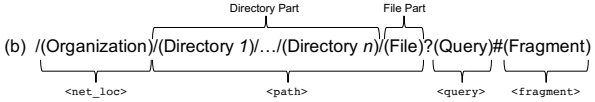

Fig. 3: (a) URL [10]; (b) content name naturally extending URL [4].

Mason et al. [9] proposed English Shellcode to encrypt and hide information, which transforms the Shellcode into the one similar to English prose. We show a similar design of encoder/decoder to include some information in names of Interest packets, assuming:

i) the names in NDN will follow similar naming scheme as those of URLs in the current Internet and described by the RFC 1808 [10] (Fig. 3 (a));
ii) the Interest name prefix of the attacker including <net_loc> part is routable from the malware.

In this paper, we set <net_loc> part and the name of application as "attacker.(TLD)" and "/info-leak", respectively. For example, the name from <scheme> part to "/info-leak" should be "ndn://attacker.com/info-leak", assuming an attacker creates anomalous names which belong to "com" domain.

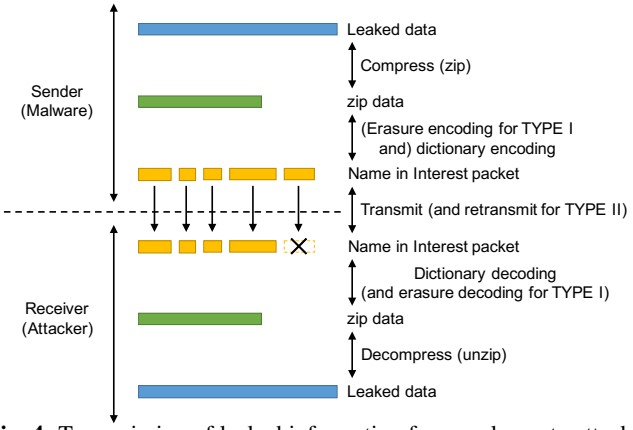

Fig. 4: Transmission of leaked information from malware to attacker.

Fig. 4 shows a general framework to transmit leaked data from a malware to the attacker. First, the data to be leaked is compressed (Zip). For TYPE I method, the malware further encodes the compressed data with erasure encoding. To bypass a firewall and realize information-leakage with names, the malware further encodes the output data by our dictionary encoding for *steganography*. Then, it creates anomalous names to leak data. To perform dictionary encoding, we prepare a table with each dictionary word and its corresponding digits. The dictionary words used in this paper will be introduced in the performance evaluation part (Section V).

III. Countermeasures with Name-Based Filters

This paper focuses on information-leakage from compromised computers in an enterprise network. We therefore assume a regular user behavior to access outside content:

i) users can access remote content, whose names are found by search engines, or by accessing links toward the content;
ii) content access policy is managed by network administrators and they can prohibit accessing unwanted content names. Network administrators can also explicitly define White List with names that can be accessed.

A. Name-Based Filter Using Search Engine Information

In the attack by *steganography embedded* Interest packets, anomalous names are created in order to leak information from corporate network (e.g., /mydomain/leak/name1/bank1/name2/bank2), and are not

common names to be requested by users with search engines (e.g., /named-data.net/doc/ndn-tlv/).

Assuming that search engine services will still exist in future NDN network, search engines could serve to help detecting legitimate names. Thus, we propose first a name-based filter using search engine information. For instance, when our filter receives an Interest name, it performs a request to a search engine. If the name is indexed by the search engine, the filter considers the Interest packet as legitimate; otherwise the filter drops the packet. This first approach is easy to implement but can also be considered as being naive regarding today's Internet use.

Indeed, according to [11], search engines index only 4% of all content (referred as Surface Web), and the remaining 96% are not indexed by search engine (referred as Deep Web). In addition, the number of Internet users using search engine decreases, as its amount was 55% in 2014, to only 49% in 2015 [12]. One of the reasons for this decrease is the mobile era, and the fact that it is difficult for users to search content with small-size mobile screen, and they prefer accessing content via other methods such as social networks. From company perspectives, it is usually prohibited for company users, 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 described in this Section, the enterprise network administrator will have to define policy to allow users access remote content; search engine or authorized links toward content.

As most of content come from Deep Web (96% as stated before), many of them should still be accessible by company users. Thus, a huge amount of content is not indexed by search engines such as newly generated ones or password-protected ones. A simple name-based filter using search engine information will therefore not be accurate enough to decide the legitimates names to be accessed. Indeed, this filter cannot be aware of most content names that are not indexed. Thus, there is need to add further information to improve the efficiency of name-based filter. To overcome this proposal, we go one step beyond and propose a more sophisticated name-based filter using one-class SVM.

B. Name-Based Filter Using One-Class SVM

In order to filter out anomalous Interest packers used to leak information from a corporate network, we propose a name-based filter using one-class SVM [13]. One-class SVM is a very useful model to perform an anomaly detection in the case that there are not enough anomalous samples. It relies on unsupervised learning techniques that are commonly used with data mining.

Regarding NDN architecture, as it is not deployed, there are currently not anomalous traffic nor content names available. Thus, we rely on URLs that are commonly used on the Internet.

We thus consider NDN names as being URLs and we study the URLs properties in the next Section. Based on these characteristics, we will be able to describe feature vector and parameters for our name-based filter using one-class SVM (Section V-A1). Attack scenarios (*steganography embedded*) with generated anomalous names will therefore be considered, and our countermeasures to detect if names are legitimate or anomalous will be evaluated.

IV. URLs Dataset

NDN is an architecture for Future Internet and it has not been deployed at large-scale. This is not an operated network and there is no data set that is representative of regular use of this network. Thus, we consider names in NDN network will be based on URLs as stated before. This section presents therefore our URLs data set and describes in details the characteristics of the data set.

In order to infer the properties of names commonly used in the Internet, we collected URLs from the data repository provided by Common Crawl [5]. At first, we obtained the crawl archive for February 2016, which holds more than 1.73 billion URLs and we extracted unique URLs belonging to 7 Top-Level Domains (TLDs): "com", "net", "org" and "info" as gTLDs (generic Top-Level Domain), and "jp", "fr", and "uk" as ccTLDs (country code Top-Level Domain). In this data set, the number of URLs vary largely for each TLD (million to hundred of millions). Thus, in order to obtain the same amount of URLs for each TLDs, we extract randomly 1 million URLs for each TLDs. Thus, in this paper, we rely on 7 millions URLs and analyze their common characteristics in the following (1 million for each TLDs).

TABLE II: URL Attributes and computed percentiles

Attributes	Percentiles		
	90%	95%	99%
Path Length (L_P)	81	98	147
Query Length (L_Q)	108	171	236
Directory Length (L_D)	19	34	72
File Name Length (L_{FN})	47	72	106
Number of "/" in Path ($N_/$)	4	5	7
Number of "=" in Query ($N_=$)	4	6	13
Number of "&" in Query ($N_\&$)	3	5	13
Avg. Alphabet Characters in Path			
Avg. Other Characters in Path			
Avg. Alphabet Characters in Query			
Avg. Other Characters in Query			

1) URL Attributes: Similarly to our previous paper [4] , we extract from URLs several attributes to describe the URLs properties (see Fig. 3 (b)) that are summarized in Table II. All the URLs in our data set provided by Common Crawl did not necessarily have <fragment> part as defined in RFC 1808. Thus, we did not consider these URLs for our analysis. For some other URLs, there is no "file name" attributes because default pages are often omitted such as index.html. In such a case, we set the length of file name to 0. In addition, for each TLD, we compute the average frequencies of alphabet characters and other printable characters of the URLs path and query parts. We referred to other printable characters in URLs as in RFC 3986 [17].

2) URL Statistics for Each TLD: The Cumulative Distribution Function (CDF) of the attributes is also computed and we show on Table II the 90/95/99-percentiles of these attributes. As these CDFs and 95th percentiles are consistent to those computed for other data set in our previous work [4], we argue that these distributions are stable for any huge URL data set repository. Thus, the properties we infer from URLs can therefore be generalized to any other URLs, and to future NDN names.

TABLE III: Cosine similarity of averaged frequencies of alphabets in Path and Query compared to typical English Text

TLD	com	net	org	info	jp	fr	uk
Path	0.970	0.957	0.960	0.968	0.976	0.975	0.975
Query	0.930	0.889	0.936	0.928	0.922	0.944	0.947

3) Frequencies of Characters: As Born et al. [23] and Qi et al. [24] analyse character frequencies in normal domain names, we compute the averaged frequencies of alphabets and the other printable characters in Path and Query respectively. For each TLD, the frequencies of alphabet characters are consistent, and also are similar to the frequencies of alphabet characters in typical English Text as defined in [18]. We then compute the cosine similarities of averaged frequencies of alphabet characters in Path and Query compared to the frequencies of alphabet characters in typical English Text as shown in Table III. From these results, for all TLDs, the cosine similarities metric for Path and Query parts is larger than 0.957 and 0.889 respectively. In addition, the frequencies of the other printable characters are also similar. Specific characters such as '-', '_', '.', and '%' were also more frequently used than any other.

Thus, our URL data set exhibits high similarity with typical English Text and this is the reason why we choose WordNet [19] as our dictionary coding to create anomalous names instead of segments from the URL dataset (Section V-B2).

V. Performance Evaluation

In this section, we evaluate the performances of the *steganography embedded* Interest packets attacks and show the throughput that can be achieved to bypass filters and leak information from the network. We separated the 7 TLDs data set presented in Section IV into two distinct sets: a *Training set* and a *Testing set*. Each training set contains 800,000 URLs, and it is used for the SVM to learn the classification rules. The remaining 200,000 URLs are used to test our filters.

In general, security exploits can be distinguished into two different types: a1) an attacker does not know the countermeasure by the protector; a2) an attacker knows the countermeasure but not its parameters. Similarly, for countermeasures: p1) a protector does not know the attack method by attacker; p2) a protector knows the attack method but not its parameters. In this paper, we consider the scenario where the protector knows the attack method (i.e., *steganography embedded* Interest packets) and prepares the protection method (i.e., p2). In order to study the upper limit of information-leakage risk for the case p2, this paper analyses the risk in the case that the attacker knows the countermeasure but not its parameters (i.e., a2); this case is of benefit to the attacker.

A. Protector

1) Name-Based Filter Using One-Class SVM: We propose a name-based filter based on one-class SVM as presented in Section III-B. We derive from the URLs properties 125 features for the one-class SVM method as follows: seven URLs attributes from Table II (Path Length, Query Length, etc.), 26 alphabets characters in Path and in Query, 33 other printable characters in Path and in Query.

For our SVM filter, we choose the "radial basis kernel" parameter because it is the most adapted to our data set [14]. With this kernel parameter, we made the models fit to the training set configuring the parameter ν, which is the upper bound of the training errors, as 0.01, 0.05, 0.1, 0.2, 0.3 and 0.4, and also the parameter γ, which is kernel coefficient, as one divided by the number of dimensions in the feature vector (i.e., 1/125), which is the default value in scikit-learn [15]. On the one hand, if γ is very large, it can lead to over-fitting. On the other hand, if γ is very small, the model can be similar to linear model. It means that the training and testing errors depend on the value of γ.

Table IV shows the training error and testing error for each value of ν. In all cases, the differences between training and testing errors are quite small, and the training errors are close to the ν parameter. As training and testing exhibit the same low errors, we can generalize the results and the false positive rate will be close to ν value. Note that false positive rate is defined as the ratio of the number of legitimate content names identified as anomalous divided by the total number of legitimate content names.

B. Attacker

1) Leaked Data: In order to leak the data, we prepared three Pdf files (Y.4001/F.748.2, Y.4412/F.747.8, Y.4413/F.748.5) from latest ITU-T recommendations [16]. 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, we compressed and converted these files into a single Zip file (3.4 MBytes). Hereafter, we consider how an attacker obtains this file. In our experiment, the Zip file is directly encoded into the names in Interest packets by dictionary encoding without erasure encoding.

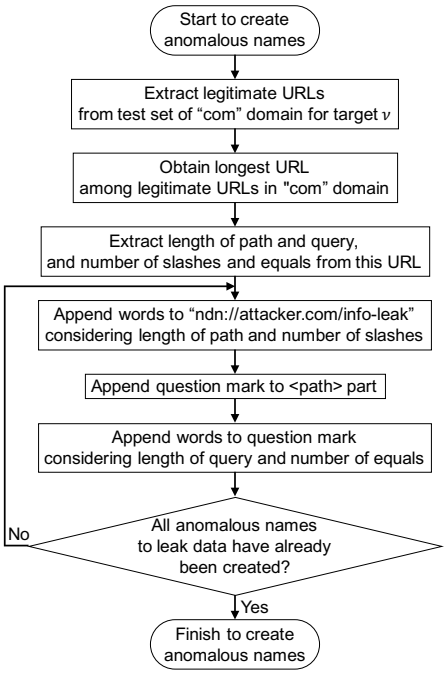

Fig. 5: Flow to create anomalous names to leak data in the "com" domain (Explanation about ν is shown in Section V-A1).

2) Anomalous Name Creation by Attacker: Fig. 5 shows the detailed flow to create anomalous names with the dictionary coding (i.e., *steganography*) in "com" TLD. As an attacker wants to add as many meaningful words as possible into each name to increase the throughput of the leaked information, the main principles to create anomalous names are (i) to choose the legitimate URL whose length is long enough; (ii) then to extract the name features (such as Path Length); and (iii) to make the anomalous names similar to the selected URL.

TABLE IV: Training error and testing error

TLD	$\nu = 0.01, \gamma = 1/125$		$\nu = 0.05, \gamma = 1/125$		$\nu = 0.1, \gamma = 1/125$		$\nu = 0.2, \gamma = 1/125$		$\nu = 0.3, \gamma = 1/125$		$\nu = 0.4, \gamma = 1/125$	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing	Training	Testing	Training	Testing
com	0.0209	0.0228	0.0603	0.0606	0.110	0.111	0.199	0.200	0.308	0.309	0.396	0.397
net	0.0101	0.0112	0.0498	0.0503	0.113	0.114	0.212	0.213	0.296	0.296	0.393	0.392
org	0.0103	0.0117	0.0499	0.0512	0.0996	0.100	0.206	0.207	0.297	0.298	0.394	0.395
info	0.0102	0.0109	0.0498	0.0509	0.0993	0.101	0.198	0.199	0.295	0.295	0.393	0.393
jp	0.0101	0.0106	0.0795	0.0789	0.0990	0.0987	0.197	0.195	0.294	0.293	0.420	0.418
fr	0.0378	0.0386	0.0498	0.0501	0.0995	0.0986	0.199	0.197	0.297	0.296	0.423	0.422
uk	0.0103	0.0120	0.0685	0.0684	0.118	0.117	0.199	0.198	0.316	0.315	0.415	0.414

TABLE V: Threshold of each attribute in anomalous names

TLD	$\nu = 0.01$				$\nu = 0.05$				$\nu = 0.1$				$\nu = 0.2$				$\nu = 0.3$				$\nu = 0.4$			
	L_P	L_Q	$N_/$	$N_=$	L_P	L_Q	$N_/$	$N_=$	L_P	L_Q	$N_/$	$N_=$	L_P	L_Q	$N_/$	$N_=$	L_P	L_Q	$N_/$	$N_=$	L_P	L_Q	$N_/$	$N_=$
com	10	773	2	15	19	391	3	7	19	230	3	7	16	171	2	6	15	141	2	3	15	112	2	4
net	32	496	4	4	212	0	12	0	19	178	2	7	16	118	2	6	22	80	2	2	20	76	2	3
org	22	334	2	3	16	248	2	4	15	212	2	6	16	151	2	6	17	109	2	5	17	98	2	4
info	206	121	5	5	26	249	2	24	26	245	2	24	46	67	3	4	14	86	2	5	16	80	2	1
jp	342	0	3	0	226	0	3	0	191	0	4	0	191	0	4	0	191	0	4	0	190	0	4	0
fr	286	0	10	0	10	261	1	10	10	259	1	10	10	229	1	9	26	189	3	14	26	178	3	14
uk	15	270	2	3	11	263	1	14	11	263	1	14	11	260	1	14	11	258	1	14	12	132	2	7

In order to prepare a table with each dictionary words and its corresponding digits (*steganography*), we extracted the words from WordNet [19]. WordNet 3.1 counts 147,478 words. In terms of the transmission rate, it is better to select shorter words. Therefore, we sort the 147,478 unique words in ascending order of their length, we extract the top $16^4 = 65,536$ words, and we assign 4 hexadecimal digits to each word in our conversion table.

We assume that the attacker prepares a data set independently from the protector and builds its one-class SVM filter by using this dataset and creates information-leakage packets with anomalous content names. For each TLD, Table V shows the threshold value of each attribute for building anomalous names with flow from Fig. 5. These thresholds are presented for ν value from 0.01 to 0.4. L_P, L_Q, $N_/$ and $N_=$ show the threshold of the Path Length, the threshold of the query length, the number of '/' and the number of '=', respectively.

C. Per-Packet Throughput of Information-Leakage

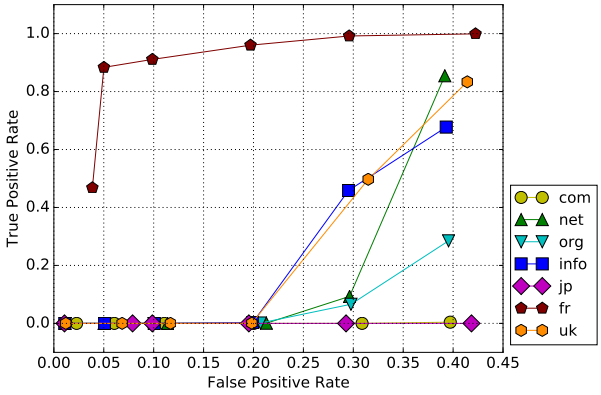

Fig. 6: ROC curve for dictionary encoding with $\nu = 0.4$.

Fig. 6 shows the ROC curves when the names by dictionary encoding designed for the thresholds at $\nu = 0.4$ are processed by the filters designed for $\nu = 0.01$ to 0.4. While the True Positive Rate (TPR), which is defined as the ratio of the number of anomalous content names identified as anomalous divided by the total number of anomalous content names, all false positive rates are the testing errors for the filters using one-class SVM shown in Table IV. Fig. 6 shows that all the TPR are almost 0 for $\nu < 0.2$ except for the "fr" domain. The reason why TPR for "fr" domain is high is that $N_= = 14$ at $\nu = 0.4$, and it is a much larger value than the one computed in Table II (95th percentile, $N_= = 6$).

If our filter is not used, the attacker can fill the URL with the leaked data in hexadecimal digits. Selecting the longest URL in our data sets (4,127 characters excluding FQDN), the per-packet throughput of information-leakage reaches 2.06 KBytes/Interest_packet, which is maximum since 1 Byte of leaked data is mapped into 2 hexadecimal digits.

If our filter is used, as shown in Section V-B2, the attacker has to encode two Bytes of leaked data into each word used in the name template (e.g., ndn://attacker.com/info-leak/word1?key1=word2&key2=word3&key3=word4&key4=word5). Thus, as an example, our created name template conveys at most 10 Bytes of leaked data because all the words in the template may not be used due to the threshold restrictions in Table V. When ν for the one-class SVM filter is set to less than 0.2, our SVM filter can start filtering out some of these packets. For each name template, regarding previous example in Section V-B1 with 3.4 MB of leaked data, one can compute the per-packet throughput by summing up all the received leaked data bytes and dividing this sum by all the sent packets. The average of per-packet throughput for all the domains except "fr" gives the information-leakage throughput in our data set, which is up to 7.79 Bytes/Interest_packet. We do not include "fr" domain as its TPR was high compared with other domains (Fig. 6). Thus, by using our filter, the malware has to send 264 times more Interest packets to the attacker than without using the filter (2.06 $KB/7.79\ B$).

D. Discussion

According to Fig. 6, we find that an attacker can perform information-leakage through Interest packets with two methods. The first one is to design anomalous names for ν parameter much larger than the one of the targeted firewall. The second is to choose the best URL seed for this larger ν. The best seed URL to create anomalous names must be the longest URL with $N_/$ and $N_=$ less than their 95th percentile, $L_P >$ (Average word length) $\cdot N_/$, and $L_Q >$ (Average word length + key length) $\cdot N_=$.

Note that the false positive rates shown in Fig. 6 are computed only for the filter using one-class SVM. Actual false positive rate should be computed after checking content names with the name-based filter using search engine information and one-class SVM. Thus, the actual false positive rate is $\nu \times R_{NotIndexed}$. $R_{NotIndexed}$ is the probability that legitimate

users access real Deep Web content, which are not indexed by search engine. Fig. 7 shows the actual false positive rate for each ν depends on $R_{NotIndexed}$, which will be very small if the enterprise network is properly managed. Therefore, we can keep low false positive rate and high performance for our filter.

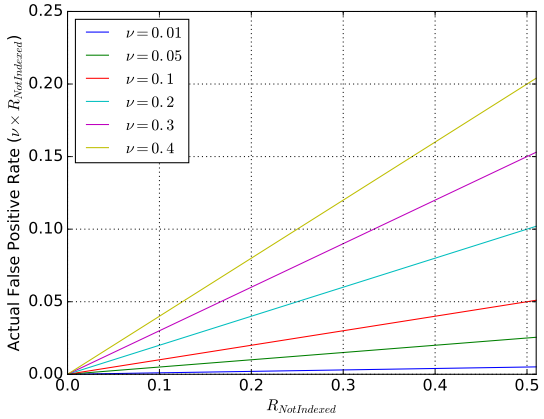

Fig. 7: Actual false positive rate.

Although it is possible for information to be leaked from NDN network through Interest packets, our proposed filter chokes off per-packet throughput of information-leakage. Therefore, the malware has to send a huge amount of Interest packets to the attacker to leak information and the throughput of this threat is drastically reduced. Thanks to our filter, it will be easy to detect anomalous Interest traffic by a per-flow filter based on traffic analysis at the subsequent stage. Network administrator can also perform naming policy control in NDN and reduces risks of information-leakage.

VI. Related Work

In this section, we present information-leakage through DNS Tunneling in the Internet, whose schema is similar to information-leakage through Interest packets in NDN.

In the Internet, it is possible to bypass a firewall and realize tunneling of data and commands by exploiting DNS queries and responses. This threat is called DNS tunneling. When an attacker carries out DNS tunneling, the attacker exploits Fully Qualified Domain Name (FQDN) or DNS record such as TXT record; the attacker can therefore access important information. Leijenhorst et al. [20] show that DNS tunneling can achieve 110 KBps with DNScat [21], but it adds huge traffic overhead. Merlo et al. [22] compare some DNS tunneling tools in terms of throughput, RTT, and overhead. In order to detect DNS tunneling, some countermeasures have been proposed. Born et al. [23] and Qi et al. [24] analyze domain names character frequencies and detect DNS tunneling domain names. Differently, Farnham [25] investigates not only payload but also traffic analysis (e.g., frequency analysis for domain name). Kara et al. [26] focus on DNS TXT record and detect DNS tunneling activities with this record. But these countermeasures are only effective to detect attacks generated by Morto worm [27] or other specific tools. Xu et al. [28] concludes that DNS-based botnet C&C channel, which is based on DNS tunneling, is "feasible, powerful, and difficult to detect and block".

VII. Conclusion

This paper performs risk analysis of information-leakage in NDN architecture. At first, this paper presents two possible attacks to perform information-leakage through Interest packets based on content names utilizing *steganography*. Then, this paper proposes a name-based per-packet filter using search engine information and another per-packet filter using one-class SVM. We collected URLs from the data repository provided by Common Crawl and evaluate the performances of our filters. Our per-packet filters can reduce drastically the speed of this threat, so that malwares have to send 264 times more Interest packets to leak information. Then, it is possible to mitigate information-leakage in NDN network, and thus the NDN architecture will be more resilient to information-leakage.

As next step, we will propose per-flow filters against possible counter-attacks for our per-packet filters.

Acknowledgment

This work is supported by DOCTOR Project, funded by French National Research Agency (ANR-14-CE28-0001).

References

[1] IT Security Risks Survey 2014: http://media.kaspersky.com/en/IT_Security_Risks_Survey_2014_Global_report.pdf
[2] http://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/the-impact-of-targeted-attacks
[3] L. Zhang, et al., "Named Data Networking," *ACM SIGCOMM CCR*, vol. 44, no. 3, pp. 66–73, Jul. 2014.
[4] D. Kondo, T. Silverston, H. Tode, T. Asami, and O. Perrin, "Name Anomaly Detection for ICN," in *IEEE LANMAN* 2016.
[5] Common Crawl, http://commoncrawl.org/
[6] M. Luby, "LT Codes," *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, vol. 0, 2002.
[7] A. Shokrollahi, "Raptor Codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.
[8] NDN Packet Format Specification, http://named-data.net/doc/ndn-tlv/
[9] J. Mason, S. Small, F. Monrose, and G. MacManus, "English Shellcode," in *CCS 2009*.
[10] RFC 1808, https://tools.ietf.org/html/rfc1808
[11] "Under The Ocean of the Internet - The Deep Web," https://www.sans.org/reading-room/whitepapers/covert/ocean-internet-deep-web-37012
[12] http://uk.businessinsider.com/the-number-of-people-using-search-engines-is-in-decline-2015-9?r=US&IR=T
[13] B. Schölkopf, et al., "Estimating the Support of a High-Dimensional Distribution," *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, Jul. 2001.
[14] C. Hsu, C. Chang, and C. Lin, "A Practical Guide to Support Vector Classification," 2003.
[15] F. Pedregosa, et al. "Scikit-learn: Machine Learning in Python," *Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[16] ITU-T http://www.itu.int/en/ITU-T/publications/Pages/latest.aspx
[17] RFC 3986, https://tools.ietf.org/html/rfc3986
[18] Frequency analysis, https://en.wikipedia.org/wiki/Frequency_analysis
[19] G. A. Miller, "WordNet: A Lexical Database for English," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, Nov. 1995.
[20] T. V. Leijenhorst, K. Chin, and D. Lowe, "On the Viability and Performance of DNS Tunneling," in *ICITA* 2008.
[21] DNScat, http://tadek.pietraszek.org/projects/DNScat/
[22] A. Merlo, G. Papaleo, S. Veneziano, and M. Aiello, "A Comparative Performance Evaluation of DNS Tunneling Tools," in *CISIS* 2011.
[23] K. Born, and D. Gustafson, "Detecting DNS Tunnels Using Character Frequency Analysis," http://arxiv.org/pdf/1004.4358v1.pdf
[24] C. Qi, X. Chen, C. Xu, J. Shi, and P. Liu, "A Bigram based Real Time DNS Tunnel Detection Approach," in *ITQM* 2013.
[25] G. Farnham, "Detecting DNS Tunneling," https://www.sans.org/reading-room/whitepapers/dns/detecting-dns-tunneling-34152
[26] A. M. Kara, et al., "Detection of Malicious Payload Distribution Channels in DNS," in *IEEE ICC* 2014.
[27] Morto Worm Sets a (DNS) Record, http://www.symantec.com/connect/blogs/morto-worm-sets-dns-record
[28] 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.