ICN PATTA: ICN Privacy Attack Through Traffic Analysis

Enkeleda Bardhi*, Mauro Conti†, Riccardo Lazzeretti* and Eleonora Losiouk†
*Department of Computer, Control and Management Engineering, Sapienza University of Rome
Email: {bardhi, lazzeretti}@diag.uniroma1.it
†Department of Mathematics, University of Padua
Email: {conti, elosiouk}@math.unipd.it

***Abstract*—PATTA is the first privacy attack based on network traffic analysis in Information-Centric Networking. PATTA aims to automatically identify the category of requested content by sniffing the communication towards the first hop router. PATTA applies text processing and machine learning techniques to content names in content-oriented architectures. We evaluate PATTA in a simulated network, achieving an accuracy in determining a real-time content category equal to 96%.**

***Index Terms*—Information-Centric Networking, Named Data Networking, network traffic analysis, user privacy attack.**

I. Introduction

Network traffic analysis is the process of monitoring the packets exchanged in a communication to identify possible anomalies [17]. Its application has been investigated by researchers in the TCP/IP protocol with the purpose of violating user's privacy by retrieving information. Despite their purpose, all its existing applications rely on a set of features contained in network packets or in the exchanged traffic. For example, the identification of a visited website occurs through Machine Learning (ML) classifiers trained over a set of network traffic information, such as the destination IP address [10]; the activities performed by a user on a mobile device can be determined by supervised ML classifiers applied on multiple network data flows [11], [18]. The selection of the most suitable algorithm in order to construct the classifier's model depends on different criteria, such as size of the available dataset, classification speed, nature of the dataset. Some possible and widely used classifiers are: Support Vector Machine (SVM), Artificial Neural Networks (ANN), Naïve Bayes (NB), Multinomial Naïve Bayes (MNB), k-Nearest Neighbors (kNN). Some countermeasures against traffic analysis, such as The Onion Router (TOR) [2] or traffic morphing [20], have been proposed. Nevertheless, Dyer et al. [12] showed these countermeasures still fail when coarse-grained side-channels attacks are applied. Furthermore, Cai et al. [8] demonstrated that a website fingerprinting attack has 50% probability of success while using HTTPOS [13] and randomized pipelining over TOR. Panchenko et al. [16] also proposed a fingerprinting attack even under a TOR anonymity system. To this end, Panchenko et al. [15] showed that simple features extracted from TOR traffic statistics and a careful selection of the training set can allow the attacker to achieve really good fingerprinting results even in Internet scale.

Even if widely studied in existing Internet architecture, to the best of our knowledge, so far nobody has investigated its feasibility in Information-Centric Networking (ICN), which motivated us to research this further. ICN, born to overcome the limitations of the current Internet (e.g., the availability of IP addresses, the increasing number of network devices, the lack of security defence mechanisms), encounters Named Data Networking (NDN) [21] architecture as the most successful one. Despite the benefits, researchers have already identified several threats targeting the user privacy [5], [6], [9], among which some concern the NDN naming schema [14]: content names are visible to the network and they contain semantic information about the content itself, inhibiting privacy practices. In this scenario, an attacker can map the content requested by the victim to its category, by only observing the content names specified in the interest packet. Furthermore, the attacker gets rid of the problem of plural names used to refer to the same content by using automatic approaches that allow her to define content's category. In this paper, we propose the first Privacy Attack Through Traffic Analysis (PATTA) in the NDN architecture. PATTA relies on both text processing techniques and ML approaches to infer the category of the content that a victim requests. In particular, PATTA involves a set of supervised ML classifiers trained over the most significant segments of a predefined set of content names, extracted through a Term Frequency-Inverse Document Frequency (TF-IDF) approach. Thereafter, an attacker sniffs the communication between the victim and the first hop router and makes use of the trained classifiers to classify categories of requested contents, achieving an accuracy up to 96%. The contributions of this paper are: (i) design and implementation of PATTA; (ii) evaluation and proofness of the feasibility of PATTA in a simulated scenario; (iii) release of the dataset of PATTA[1].

II. Background

A. NDN Overview

In NDN, the basic communication model is changed with respect to the current Internet architecture. The communication entities that produce data are called *producers*, while the ones that request data are called *consumers*. Instead of addressing

[1]https://gitlab.com/bardhienkeleda/icn-patta-dataset.git

hosts in order to deliver packets to the receiver, in NDN the consumers request the content by using unique application-layer names. Content can be requested by consumers issuing an interest request packet that carries the content name, while producers respond to the interest request by using data packets. The communication follows a *pull* approach: a data packet reaches the consumer only upon receiving a request for that content.

B. Text Processing Techniques

TF-IDF [22] is a widely used Natural Language Processing (NLP) technique for text classification. It measures the importance of one (or a set) of document(s) by returning a score, which increases according to how many times a word is present in a text and decreases with the frequency of times it is found in the complete set of documents. There are other text processing techniques such as Bag of Words (BoW) [19] and Google Word2Vec [4]. The former is a basic model used in NLP since it produces a vector containing word occurrences and discarding the order. Instead, the latter places the words into feature space and their location is determined by their meaning.

III. Assumptions

A. System Model

We consider a generic NDN network with a set of consumers, all connected to the first hop router, and a producer. In particular, we assume that the entities participating in the network do not rely on privacy preserving techniques. Consumer issues content requests following Zipf distribution [7] with a frequency of 1 request/second. Additionally, we assume that `Name` field in the interest request packet is not encrypted.

B. Threat Model

For the attack scenario we consider a victim issuing content interest requests and an attacker sniffing the outgoing victim's traffic between the victim and the first router. While sniffing, the attacker extracts the significant features of the sniffed network packets and uses automatic classification tools to infer the victim's requested content. The attacker aims to discover the content's category that the victim is requesting among a set of sensitive content categories. Privacy of a single user can be disrupted by the attacker being able to deduce the category of the content that the user is requesting, among which also *sensitive* content. Similarly, an attacker can use this attack to disrupt the privacy of a group of users. The attacker is able to eavesdrop the network traffic passing through the wireless link connecting the victim or victims with the first hop router. Also, she does not need to have knowledge about the rest of the network, i.e., routers and producers.

IV. PATTA Attack overview

PATTA exploits one of the crucial characteristics of NDN paradigm, i.e., the self-explanatory nature of content names. For example, a name containing a component entitled *video* is expected to refer to a video content. PATTA is flexible to multiple content names associated to the same content by producers. PATTA achieves this by using ML in conjunction with text processing. PATTA encompasses two main phases: *classifiers' setup* and *real-time classification*. In the former, the attacker collects victim's requests for (sensitive) contents. Then, she goes through a preprocessing phase. During the preprocessing, she initially filters the collected dataset to keep only names of the target contents. Then, she further transforms the format of collected traffic from TCP/IP into NDN. Finally, she labels the dataset and splits it into training, testing and flexibility sets. On the training set, she first performs a feature extraction procedure through two text processing techniques, to identify the most significant segments in content interest names. Then, she uses the identified features to train ML classifiers. At the end of the training phase, the attacker tests the obtained models on the testing set. The latter phase starts with a network traffic capturing during which the attacker sniffs the outgoing traffic between the victim and the first hop router. The real-time network traffic requires a preprocessing, i.e., to consider only the interest requests without taking into account the responses from the producer and thereafter prepares a set of features, namely, a set of the most important interest name segments. These features are obtained using the same text processing techniques used in the *classifiers' setup phase*. Afterwards, the attacker uses the obtained features of observed traffic together with the already trained ML classifiers to classify the victim's real-time interest requests.

V. Classifiers Setup

A. Traffic Collection

To obtain our dataset, we collect a TCP/IP traffic and afterwards transform into the NDN format. Due to the similarity of NDN content names with URLs, we leverage the website scraping technique to collect URLs contained in a set of websites belonging to the following categories: Adults, Arts & Entertainment, Education, Faith & Beliefs, News and Technology. In order to choose the websites for each category, we follow some websites classification lists. For example, for News category, we scrape the top 20 journalism brands according to Forbes. In particular, we pick 300 websites and design an automated Python script to visit them. For each website, the script visits its homepage and all the links it contains. We expect that the number of links visited for each page varies depending on the considered category. For example, for News category, we visit 21 websites and obtain 2608 URLs. For the Faith and Beliefs category, even though we pick a higher number of websites to be scraped, i.e., 35 websites, we obtain a smaller number of instances, i.e., 1153 instances. The full number of selected websites and obtained instances for each category is represented in Table I. We use an Ubuntu 20.04 TLS machine to run the automated script, Mozilla 81.0 as a browser for traffic generation and Wireshark [3] as traffic sniffing tool.

B. Dataset Preprocessing

For preprocessing task, we filter HTTP and HTTPS traffic using Wireshark. This procedure is performed for each generated traffic file in a Comma-Separated Values (CSV) format. All these files are merged into the final dataset. Furthermore, we filter out the traffic of those websites that do not contain links in their homepage. Then, for each website homepage and for all links contained in it, we assign the category name as its label. Next, we proceed with the transformation of traffic from HTTP requests to NDN interests. Each HTTP/HTTPS request for a certain webpage becomes an NDN interest having the same structure. In particular *http, htpps* words are transformed into *ndn* and other special characters such as *?* are transformed into */*. At the end of this step, for each category, we randomly pick one domain for each category and create the flexibility set. The remaining domains are merged into a unique dataset that is moreover split in 70% for training and 30% for testing.

C. Feature Selection

For each NDN content name in the dataset, TF-IDF creates a BoW, while the corpus is the union of all bags of words. Each word of the corpus gets a TF-IDF score. During the creation of both BoW and corpus, we perform text cleaning to remove special characters, insignificant words, and very frequent-small length words (e.g., ndn, www, com), which could disrupt the classifier's performance. For the feature selection task, along with TF-IDF, we make use of *N-grams*, a text representation mainly used in auto completion sentence and auto spell check systems. The use of *N-grams* during TF-IDF calculation allows to keep as much useful information as possible from the tree-like structure of NDN content names, maintaining also the sequential order of name components. We use sequences of terms one component long (i.e., unigrams) and two component long (i.e., bigrams) or a combination of both unigrams and bigrams. Finally, we discard words which are shorter than three, random words without a meaning in English dictionary and words with TF-IDF score smaller than the threshold chosen after different experimentation with different threshold values.

D. ML Classifiers Training

To train the ML classifiers, we only exploit *supervised learning* algorithms due to their good performance on classification tasks and the labeled dataset. All ML models are trained using the features selected from the previous step. We train 18 different models, varying the number of features and the configuration of *N-grams*: 1-grams, 2-grams or both 1-grams and 2-grams. Then, we proceed with a two-step testing of the trained classifiers: firstly, using instances belonging to the same website domains used during training and secondly, using instances belonging to website domains different from the ones used during training. By doing so, we aim to evaluate the generalization capabilities of the trained classifiers. Table II shows the accuracy obtained on the testing set for 18 classifiers. The *accuracy* score of the model increases with respect to the increasing number of features for the three models, i.e., Linear-Support Vector Machine (L-SVM), MNB and SVM. We additionally computed *f1-score* for all the classifiers and the results showed that it is aligned with accuracy score. Therefore, for evaluation we stick only with accuracy score. Next, we calculated the accuracy of the classifiers on the flexibility set and Table III shows the results. The achievement of high accuracy scores for classifiers, such as MNB, SVM and L-SVM, even in unseen domains during training demonstrates the capability of our classifiers to correctly categorize with high probability new content based on word's importance. At the end of this step, we select the best performing classifiers in terms of accuracy, in both testing and flexibility sets.

TABLE I: Number of domains and instances for each category.

	Adult	**Arts & Entert.**	**Education**	**Faith & Beliefs**	**Health**	**News**	**Technology**
Number of websites	63	51	56	35	32	21	33
Number of instances	5586	3333	3495	1153	1346	2608	2105

TABLE II: Classifier accuracy over the testing set.

Classifier's Model	**1-grams 893 features**	**1-grams 1785 features**	**2-grams 460 features**	**2-grams 917 features**	**(1, 2)-grams 1350 features**	**(1, 2)-grams 2700 features**
L-SVM	$91,92\%^{\dagger}$	$93,34\%^{\dagger}$	$60,75\%^{\dagger}$	$66,20\%^{\dagger}$	$92,30\%^{\dagger}$	$93,43\%^{*}\dagger$
MNB	88,76%	90,10%	56,82%	63,05%	89,05%	$90,16\%^{*}$
SVM	91,84%	$93,20\%^{*}$	60,64%	66,07%	91,89%	93,09%

$*$ Best performing model of the line † Best performing model of the column

TABLE III: Classifier accuracy over the flexibility set.

Classifier's Model	**1-grams 893 features**	**1-grams 1785 features**	**2-grams 460 features**	**2-grams 917 features**	**(1, 2)-grams 1350 features**	**(1, 2)-grams 2700 features**
L-SVM	45,84%	52,05%	$40,27\%^{\dagger}$	39,46%	49,19%	$54,70\%^{*}$
MNB	$62,04\%^{\dagger}$	$66,24\%^{\dagger}$	39,49%	$41,71\%^{\dagger}$	$62,20\%^{\dagger}$	$66,74\%^{*}\dagger$
SVM	49,70%	57,76%	39,85%	39,14%	54,54%	$61,02\%^{*}$

$*$ Best performing model of the line † Best performing model of the column

VI. Real-Time Classification

Here, we present the results of the *real-time classification* phase during which we use mini-NDN [1]. For the experiments

we consider three different scenarios, one for each of the best performing classifiers: SVM, MNB and L-SVM. For each of the considered scenarios, we then consider two configurations: only 1-grams used with 1785 features and both (1, 2)-grams with 2700 features. We exclude 2-grams due to their results. For each scenario, we run five simulations lasting 600 seconds each, while considering one or five victims. Each victim connected to the first hop router issues 1 request/second following a Zipf distribution with $\alpha = 0.95$ and picks the requests from the obtained union of testing and flexibility set. Table IV shows the results of classification while both one or five victims are connected to the first hop router. In the former case, we achieve good results for all three classifiers and in particular for SVM classifier which achieves up to 96% of accuracy. For the latter case instead, similarly to what might happen in a real scenario, due to a higher number of requests for content issued by a group of victims, the attacker might not capture the whole traffic generated by victims. According to our simulations, with 5 victims requesting content following Zipf distribution, where each of them issues 1 request/second and with five simulations for each configuration of classifiers, the attacker captures 90% of the victims' traffic. Even in this case, with 91% accuracy the attacker is successful in inferring the contents requested by the victims. Due to the results we showed before and also in Table IV, we conclude that, by carefully training the classifiers, the attacker is able to achieve really good classification results.

TABLE IV: Classifier's accuracy tested on 1-grams with 1785 features and (1,2)-grams with 2700 features for 1 and 5 victims.

Classifier Model	**1-grams 1785 features**	**(1, 2)-grams 2700 features**
L-SVM	**V1**: 87,34%	**V1**: 91,74%
	V5: 84,53%	**V5**: 88,00%
MNB	**V1**: 92,16%	**V1**: 92,65%
	V5: 89,01%	**V5**: 88,81%
SVM	**V1**: 89,33%	**V1**: 96,19%
	V5: 80,99%	**V5**: 91,14%

V1: One victim **V5:** Five victims

VII. Conclusions

In this paper, we proposed and evaluated PATTA, a novel privacy attack in NDN. PATTA makes use of a promising approach since we investigated a text processing technique such as TF-IDF that allows us to correlate NDN content name segments with the category content being requested. By relying on this technique, the category recognition capabilities of our attack are robust also in case of possible changes in the content names and in the presence of totally new content names. In our work, we first trained a set of supervised learning classifiers based on the relevance of segments composing the content name and we achieved recognition results up to 96%. Subsequently, we transferred our findings into a full NDN simulated environment in order to prove the feasibility of our attack. According to our empirical results, the attacker is able to analyse in real time the generated traffic by the victim and achieves 96% when only 1 victim is considered or 91% with 5 victims. Future work shall further investigate the feasibility of attacks in the case of a higher number of victims connected to the first hop router. Furthermore, we also leave room for implementation of the attack in a real scenario.

References

[1] minindn. [Online]. Available: https://github.com/named-data/mini-ndn
[2] Tor website. [Online]. Available: https://www.torproject.org
[3] Wireshark website. [Online]. Available: https://www.wireshark.org
[4] Word2vec code. [Online]. Available: https://code.google.com/archive/p/word2vec/
[5] E. G. AbdAllah, H. S. Hassanein, and M. Zulkernine, "A survey of security attacks in information-centric networking," *IEEE Commun. Surv. Tutor.*, vol. 17, no. 3, pp. 1441–1454, 2015.
[6] G. Acs, M. Conti, P. Gasti, C. Ghali, and G. Tsudik, "Cache privacy in named-data networking," in *ICDCS*. IEEE, 2013, pp. 41–51.
[7] L. A. Adamic and B. A. Huberman, "Zipf's law and the internet." *Glottometrics*, vol. 3, no. 1, pp. 143–150, 2002.
[8] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 605–616.
[9] A. Chaabane, E. De Cristofaro, M. A. Kaafar, and E. Uzun, "Privacy in content-oriented networking: Threats and countermeasures," *ACM SIGCOMM COMP. COM.*, vol. 43, no. 3, pp. 25–33, 2013.
[10] M. Conti, Q. Q. Li, A. Maragno, and R. Spolaor, "The dark side (-channel) of mobile devices: A survey on network traffic analysis," *IEEE Commun. Surv. Tutor.*, vol. 20, no. 4, pp. 2658–2713, 2018.
[11] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, "Can't you hear me knocking: Identification of user actions on android apps via traffic analysis." Association for Computing Machinery, 2015.
[12] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail," in *2012 IEEE Symp. Secur. Priv.* IEEE, 2012, pp. 332–346.
[13] X. Luo, P. Zhou, E. W. Chan, W. Lee, R. K. Chang, R. Perdisci *et al.*, "Httpos: Sealing information leaks with browser-side obfuscation of encrypted flows." in *NDSS*, vol. 11, 2011.
[14] E. Mannes and C. Maziero, "Naming content on the network layer: a security analysis of the information-centric network model," *ACM Comput. Surv.*, vol. 52, no. 3, pp. 1–28, 2019.
[15] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle, "Website fingerprinting at internet scale." in *NDSS*, 2016.
[16] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*, 2011, pp. 103–114.
[17] J.-F. Raymond, "Traffic analysis: Protocols, attacks, design issues, and open problems," in *Designing Privacy Enhancing Technologies*. Springer, 2001, pp. 10–29.
[18] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic," in *IEEE Eur. Symp. Secur. Priv. EuroS P*, 2016, pp. 439–454.
[19] H. M. Wallach, "Topic modeling: beyond bag-of-words," in *Proceedings of the 23rd ICML*, 2006, pp. 977–984.
[20] C. V. Wright, S. E. Coull, and F. Monrose, "Traffic morphing: An efficient defense against statistical traffic analysis." in *NDSS*, vol. 9. Citeseer, 2009.
[21] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.
[22] W. Zhang, T. Yoshida, and X. Tang, "Tfidf, lsi and multi-word in information retrieval and text categorization," in *Conf. Proc. - IEEE Int. Conf. Syst. Man Cybern.* IEEE, 2008, pp. 108–113.