Semi-Supervised Fingerprinting of Protocol Messages

Jérôme François, Humberto Abdelnur, Radu State, Olivier Festor

Semi-Supervised Fingerprinting of Protocol Messages

Jérôme François, Humberto Abdelnur, Radu State, Olivier Festor

Abstract This paper addresses the fingerprinting of network devices using semi-supervised clustering. Semi-supervised clustering is a new technique that uses known and labeled data in order to assist a clustering process. We propose two different fingerprinting approaches. The first one is using behavioral features that are induced from a protocol state machine. The second one is relying on the underlying parse trees of messages. Both approaches are passive. We provide a performance analysis on the SIP protocol. Important application domains of our work consist in network intrusion detection and security assessment.

1 Introduction

Assuming a protocol, fingerprinting a device aims to retrieve the names and versions of software or hardware equipment (also named the device type). From a security point of view, knowing the device type may help to design a powerful attack but the network administrator is also able to evaluate precisely the risks in order to apply necessary. Device fingerprinting may help to identify attack tools or to enforce the host identity verification. Whereas a device often announces its type thanks to a specific field in the messages (the user-agent field), it can be easily faked. Previous fingerprinting approaches show that remote identification is possible due to deviations in the implementation of a given protocol but most of them are limited to manually constructed signature. Therefore, our main motivation is to automatically observe protocol deviations in order to establish signatures without human and net-

Jérôme François (this work was finalized at the Interdisciplinary Centre for Security, Reliability and Trust - University of Luxembourg, e-mail: jerome.francois@uni.lu), Humberto Abdelnur, Olivier Festor
INRIA Nancy-Grand Est, France, e-mail: firstname.lastname@inria.fr

Radu State
University of Luxembourg, e-mail: radu.state@uni.lu

work interactions (passive fingerprinting) for any protocol (generic approach). The only assumption is to capture and label few traffic from each device type to fingerprint. We propose first a fingerprinting scheme that can learn distinctive patterns in the state machine of a particular implementation. We consider also a second fingerprinting approach, where syntactic information from parse trees is leveraged to identify a given device or protocol stack. Thus, this paper highlights the benefit of using recent classification techniques for security purposes.

Our paper is structured as follows: section 2 describes the semi-supervised clustering approach. We continue in section 3 with the state machine fingerprinting. Section 4 addresses the syntactic one. Section 5 describes relevant prior work and section 6 concludes the paper and highlights future works.

2 Semi-Supervised Clustering

2.1 Overview

Semi-supervised learning approaches focus on data sets with only a small amount of labeled data and a lot of unlabeled data samples. Semi-supervised clustering was introduced in [19]. The main idea in semi-structured clustering also known as *label propagation* algorithm is to construct a fully connected graph with some nodes with labels (class name of the node). The edges between two nodes have weights associated depending on the distances between nodes. Iteratively, all the labels are propagated to unlabeled regions in the fully connected graph. Nodes that are not labeled, will iteratively estimate the probabilities f belonging to each class. At the end of the iterations, an unlabeled node will be allocated to the most probable class.

2.2 Formal Definition

We have a set composed of labeled data (x_1,y_1),...,(x_l,y_l) and unlabeled data (x_{l+1},y_{l+1}),...,(x_{l+u},y_{l+u}) with $l \ll u$, where $Y_L = \{y_1,...,y_l\}$ are the class labels of the labeled data and $Y_U = \{y_l+1,...y_l+u\}$ unobserved yet. It is assumed that the number of classes C is known and that all classes are in the labeled data samples [19]. Let X=$\{x_1,...,x_{l+u}\}$ be the different data items x_i. We want to estimate the class labels of the unlabeled samples Y_U from the data items X and their class labels Y_L. This is done using a distance function. If two cluster tuples (x_i,y_i) and (x_j,y_j) have to be compared (y_i respectively y_j represent the class labels), the distance d_{ij} measures the difference between the two data items x_i and x_j. The performance of the semi-clustering algorithm is dependent on this distance function which is defined in next sections. The labeled and unlabeled data samples are represented in a fully connected graph, where the edge between node i, j is weighted. The edge

weight w_{ij} is given by the following expression:

$$w_{ij} = e^{-\frac{d_{ij}^2}{\sigma^2}} \tag{1}$$

As in [19] we define a $(l+u) \times (l+u)$ transition matrix T, where T_{ij} gives the probability to jump from node i to j.

$$T_{ij} = P(i \to j) = \frac{w_{ij}}{\sum_{k=1}^{l+u} w_{jk}} \tag{2}$$

We define a $(l+u) \times C$ label matrix Y, where a row reflects the label probability distribution of a node. The element Y_{ic} is the probability that element Y_i belongs to the class c. Initially this probabilities are initialized with $1/C$ for the unlabeled items. The label propagation algorithm by [19] has three different steps:

1. Propagate $Y^{t+1} \leftarrow T * Y^t$: all nodes propagate their labels, Y^{t+1} denotes the matrix Y at the iteration $t+1$
2. Row normalization of Y^{t+1}: this maintains a probability distribution.
3. Clamping of labeled data: Y_{ic}=1 if item Y_i had an initial label of c. This step assures that initial labels are maintained.

The previous steps are repeated from *step* 1 to 3, until Y converges which is always the case as proved in [19]. In a multi-class prediction problem with C classes, [5], a $C \times C$ contingency or confusion matrix $Z = z_{ij}$ is used. z_{ij} is the number of times a sample belonging to class i, is put in class j. The overall evaluation parameter is the quality Q_{total}, which is the value of all correct predictions made:

$$Q_{total} = \frac{\sum_i z_{ii}}{N} \quad where\ N = \sum_{ij} z_{ij}$$

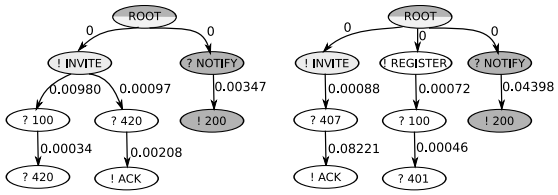
3 Fingerprinting Protocol State Machines

3.1 Behavioral Tree

Since the theoretical details of protocol state machine fingerprinting are given in [11], this section gives a general overview. Each device type is represented by multiple behavioral trees as for instance in figure 1(a). The nodes correspond to the emitted (prefixed by **!**) or received (prefixed by **?**) messages and are only symbolized by their types. Thus, each path of the tree is an observed sequence of messages (a session) for the current device with average delays put as edge labels.

For comparing such trees, a kernel function considers similar paths (*sim_paths*) (same sequence of nodes from the root) and computes a time based difference:

$$K(t_i, t_j) = \sum_{p \in sim\_paths} \sum_{edge \in p} e^{-\alpha |delay(edge,t_1) - delay(edge,t_2)|} \tag{3}$$



(a) Twinkle 1.10 (soft-phone) (b) Cisco 7940 (hardphone)

Fig. 1: Sessions tree examples. Two shared paths are grey colored

3.2 Performance Evaluation and Experimental Datasets

For validating our approach, SIP [15] was chosen due to its popularity for VoIP applications and to many related security problems as for example [3].

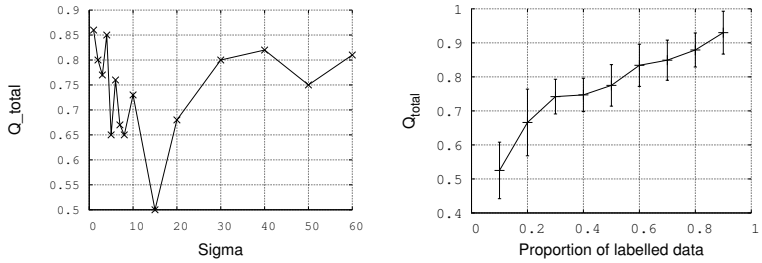
The first dataset denoted as `testbed dataset` was generated using both softphones like `Twinkle` and hardphones from different brands (Cisco, Linksys, Snom or Thomson) connected on a local testbed. The `operator dataset` refers to network traces provided by a real VoIP operator. 160MB of data were randomly extracted for our evaluation. The main difference with `testbed dataset` is that devices are connected through Internet entailing greater noise and longer delays as highlighted in table 1. Major characteristics are summarized in Table 1. INVITE messages correspond to call initiations. Whereas the `operator dataset` contains more messages, few INVITE messages are present. It reflects a realistic operator traffic where users have to periodically send REGISTER messages for maintaining the matching between the global user identifiers (SIP AOR) and the IP addresses. This type of sessions contain few messages which implies a small number of messages per session. These facts contribute to test our systems with different configurations.

	#devices	#messages	#INVITE	#sessions	Avg #msgs/session	Avg delay (sec)
Testbed	26	18066	3183	2686	6.73	1.53
Operator	40	96033	1861	30006	3.20	7.32

Table 1: Experimental datasets statistics

We use the `testbed dataset` for assessing the accuracy of our system with different σ parameter values in order to tune it before applying the fingerprinting technique on the `operator dataset`. All experiments are run multiple times and the accuracy is measured as the average of Q_{total}. The corresponding standard deviation is displayed as an error bar on graphs.

Small values of σ will tend to bring together clusters that lie far apart. Large values will move apart clusters that lie close. We have extensively tested a range of



(a) Impact of the σ (sigma) parameter (b) Impact of the proportion of labeled data

Fig. 2: State machine fingerprinting

potential values. While many choices of the σ parameter provide very good results (around 80%), one range shows very poor results as shown in figure 2(a). We have investigated this case and concluded that many clusters were colluded for this range.

Obviously, the overall quality measure is steadily increasing with respect to the proportion of the labeled data as shown in figure 2(b). For instance, when only 30% of the data is labeled, the overall quality is more than 70%. In order to obtain 90% , the system requires around 80% of labeled data. Considering the `operator dataset`, 82% of devices were identified thanks to 40% of labeled data.

4 Fingerprinting With Syntactic Information

4.1 Syntactic Trees

The syntactic fingerprinting was introduced in [10]. The key idea is that device/stack specific features can be revealed due to the programming choices taken by the software developers. In fact, a protocol is generally defined by an Augmented Backus-Naur Form (ABNF) [7] grammar. Then, each message can be represented as successive derivations of rules. A toy example is illustrated in figures 3 and 4.

For applying the semi supervised algorithm, a distance metric is necessary for comparing two trees. Because computing usual distance is time consuming, new distances with polynomial complexity were introduced in [17]. For comparing two syntactic trees, the similarity between isomorphic subtrees is computed by calculating the similarity between each node thanks to table 2 and by assuming a zero similarity if the ancestor nodes are different [10]. Assuming *max_sim*, the maximum similarity among all isomorphisms, and $|T|$, the number of nodes in the tree T, the distance between two trees, T_1 and T_2, is:

$$d(T_1,T_2) = max(|T_1|,|T_2|) - max\_sim \tag{4}$$

```
Message =  Request  SP *Header SP 0*1Body
Request = Invite / Notify / Cancel
Invite = "INVITE"
Cancel = "CANCEL"
Notify = "NOTIFY"
Header = Accept / Date / Call-id / User-Agent
Body = *Alpha
Alpha = %x41-5A / %x61-7A        ; A-Z / a-z
HCOLON =  *SP ":" *SP
SP =  %x20                       ; space
Accept = "Accept" HCOLON *Alpha "."
Date = "Date" HCOLON *Alpha "."
Call-Id = "Call-Id" HCOLON *Alpha "."
User-Agent = "user-Agent" HCOLON *Alpha "."
```

Fig. 3: Grammar

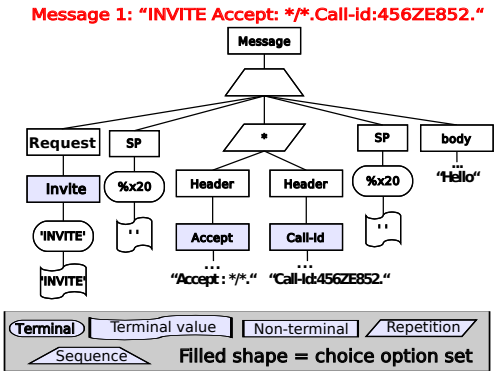


Fig. 4: Syntactic tree

	non-terminal	repetition	sequence	others
non-terminal	1	0	0	0
repetition	0	1	1	0
sequence	0	1	1	0
others	0	0	0	0

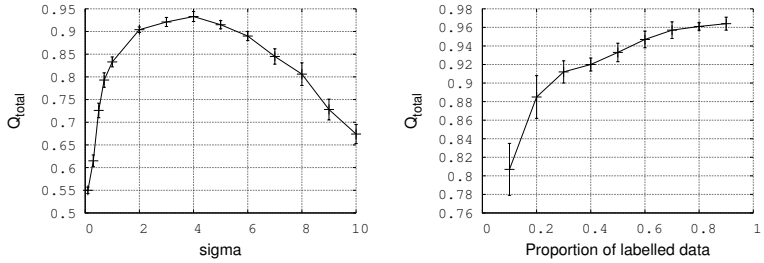
Table 2: Syntactic node similarity

Device Name	#mesg	height			#nodes		
		Max	Min	Avg	Max	Min	Avg
Asterisk_v1.4.21	1081	28	23	25	2517	883	1284
Cisco-7940_v8.9	168	25	23	24	2784	812	1352
Thomson2030_v1.59	164	28	23	24	2576	793	1391
Twinkle_v1.1	195	25	23	23	2457	805	1299
Linksys_v5.1.8	195	28	23	25	2783	852	1248
SJPhone_v1.65	288	30	23	24	2330	951	1133

Table 3: Testbed dataset – Tree statistics

4.2 Evaluation

Table 3 summarizes the characteristics of syntactic trees of the most represented device types in the `testbed dataset`. The trees are generally huge with a height around 30 and often contain more than 800 nodes due to the SIP grammar (more than 500 lines). Although, the behavioral fingerprinting results vary well due to σ as shown in figure 2(a), figure 5(a) highlights clearly a maximal value and so an easy distinguishable best configuration. Figure 5(b) illustrates the impact of the proportion of the labeled data. We observe that even a very small percentage of labeled data can be used to obtain good results. For instance, when the known labeled data is about ten percent of the whole data quantity, we can expect the quality value to be in the 81s percents. With 40%, the overall quality is 0.92. Assuming the `operator dataset`, this value reaches 68%. This limited result is due to the creation of groups including several devices which are not distinguishable. In fact, some devices use probably the same or close protocol stacks (same series, same brands) which does not entail enough differences in syntactic structure.



(a) Impact of the σ (sigma) parameter (b) Impact of the proportion of labeled data

Fig. 5: Syntactic fingerprinting

5 Related Work

Network and service fingerprinting is a common task in security and network assessment. The key assumption is that subtle differences due to development divergences can be traced [9]. Passive fingerprinting monitors traffic without any interaction as for example [2] which uses a set of signatures to identify the operating systems. In contrast, active fingerprinting generates specific requests directed to a device and monitors the responses as for instance [14] (operating system and service versioning). A related work is [6], which describes a mechanism to automatically explore and select the right requests to send. The authors in [13] give an overview of techniques used for determining the different classes of traffic (Web, P2P, Chat..) whereas [12] focus on the identification on the flow types. Our work is different and complementary since its goal is to determine precisely the implementation. This kind of methods was explored in [1] for determining the web server version by observing the value or the order of some headers. Determining the version of a SIP equipment could be based on the bad randomness value of the Call-id field [16]. As argued in the introduction, changing these fields is very easy in order to counter fingerprinting. SIP fingerprinting is also addressed in [18] with other fields protocol and an active probing technique. Furthermore, such techniques are only able to identify devices for which signatures were manually constructed and keeping a signature database up-to-date is difficult to the huge variety and upgrade of devices. A related work is presented in [4] which needs a grid of 10 computers during several days for an equivalent dataset as the `testbed dataset`. We have used the constructs introduced in our previous works[10, 11] and argued to use a semi supervised learning which uses a small quantity of labeled data. Hence, the accuracy results are similar once the percentage of messages used for training is sufficient but semi supervised fingerprinting is clearly better for small volume of training messages (10%). Especially, it provides better results (around 8%) whit the `operator dataset`. Hence, semi-supervised fingerprinting is robust even the variety of devices is very high. Whereas our previous based on Support Vector Machines [8] require various parameters to define, σ is the only one to tune for the semi supervised learning.

6 Conclusion

In this paper, we have addressed the problem of fingerprinting devices and/or implementation stacks. Our approach is based on semi-supervised clustering of time enhanced state machine induced features. We have also looked at the syntactic information that is contained in messages. We have obtained results that are promising taking into account the small quantity of labeled data. While most supervised fingerprinting algorithms use about four fifths of the data to train the system and only twenty percent for testing, our approach achieves good results when only few labeled data items are available. We will look at other protocols — for instance wireless protocols — and assess the operational applicability in this scenario.

References

1. Httpprint. http://www.net-square.com/httprint/httprint_paper.html
2. P0f. http://lcamtuf.coredump.cx/p0f.shtml
3. Abdelnur, H., Avanesov, T., Rusinowitch, M., State, R.: Abusing SIP Authentication. In: Information Assurance and Security (2008)
4. Abdelnur, H., State, R., Festor, O.: Advanced Network Fingerprinting. In: Recent Advances in Intrusion Detection (2008)
5. Baldi, P., Brunak, S., Chauvin, Y., Andersen, C.A., Nielsen, H.: Assessing the accuracy of prediction algorithms for classification: an overview. Bioinformatics **16**(5), 412–24 (2000)
6. Caballero, J., Venkataraman, S., Poosankam, P., Kang, M.G., Song, D., Blum, A.: FiG: Automatic Fingerprint Generation. In: Distributed System Security Conference (2007)
7. Crocker, D.H., Overell, P.: Augmented BNF for Syntax Specifications: ABNF (1997)
8. Debnath, R., Takahide, N., Takahashi, H.: A decision based one-against-one method for multi-class support vector machine. Pattern Anal. Appl. **7**(2), 164–175 (2004)
9. Douglas Comer and John C. Lin: Probing TCP Implementations. In: USENIX Summer, pp. 245–255 (1994)
10. François, J., Abdelnur, H., State, R., Festor, O.: Advanced Fingerprinting For Inventory Management. Research Report RR-7044, INRIA (2009)
11. François, J., Abdelnur, H., State, R., Festor, O.: Behavioral and Temporal Fingerprinting. Research Report RR-6995, INRIA (2009)
12. Haffner, P., Sen, S., Spatscheck, O., Wang, D.: ACAS: automated construction of application signatures. In: MineNet. ACM (2005)
13. Kim, H., Claffy, K., Fomenkov, M., Barman, D., Faloutsos, M., Lee, K.: Internet traffic classification demystified: myths, caveats, and the best practices. In: CoNEXT. ACM (2008)
14. Lyon, G.F.: Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning. Insecure, USA (2009)
15. Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., Schooler, E.: SIP: Session Initiation Protocol (2002)
16. Scholz, H.: SIP Stack Fingerprinting and Stack Difference Attacks. Black Hat Briefings (2006)
17. Torsello, A., Hidovic-Rowe, D., Pelillo, M.: Polynomial-time metrics for attributed trees. IEEE Transactions on Pattern Analysis and Machine Intelligence **27**(7) (2005)
18. Yan, H., Sripanidkulchai, K., Zhang, H., yin Shae, Z., Saha, D.: Incorporating Active Fingerprinting into SPIT Prevention Systems. Third Annual VoIP Security Workshop (2006)
19. Zhu, X., Ghahramani, Z.: Learning from labeled and unlabeled data with label propagation. Tech. rep. (2002). URL http://www.gatsby.ucl.ac.uk/~zoubin/papers/CMU-CALD-02-107.ps.gz