Digital Forensics in VoIP networks

Jérôme François, Radu State, Thomas Engel, Olivier Festor

Digital Forensics in VoIP networks

Jérôme François [#1], Radu State [*2], Thomas Engel [#*3], Olivier Festor [+4]

[#] *Interdisciplinary Center for Security, Reliability and Trust, University of Luxembourg*
6 rue Richard Coudenhove-Kalergi, L-1359, Luxembourg, Luxembourg
[1] jerome.francois@uni.lu
[3] thomas.engel@uni.lu

[*] *SECAN-LAB, University of Luxembourg*
6 rue Richard Coudenhove-Kalergi, L-1359, Luxembourg, Luxembourg
[2] radu.state@uni.lu

[+] *INRIA Nancy - Grand Est Research Center*
CS 20101 - 54603 Villers les Nancy Cedex, France
[4] olivier.festor@inria.fr

***Abstract*—With VoIP being deployed on large scale, forensic analysis of captured VoIP traffic is of major practical interest. In this paper, we present a new fingerprinting approach that identifies the types of devices (name, version, brand, series) in captured VoIP traffic. We focus only on the signaling plane and discard voice related data. Although we consider only one signaling protocol for the illustration, our tool relies on structural information trees and can easily be adapted to any protocol of that has a known syntax. We have integrated our tool within the well known tshark application in order to provide an easy to use support for forensic analysts.**

I. INTRODUCTION

Facing a high increase in both the number and variety of attacks, the research in digital forensics needs to address a larger scope of problems and challenges. It has been shown in [1], [2] that novel attacks can be difficult to counter and major companies like Google are also affected [3]. For such attacks, forensics is highly important for preventing future attacks, analyzing its impact, locating the responsible and recovering the system back to a safe state. Such examples show also that finding evidences is not an obvious task and requires to analyze as fast as possible a huge volume of data [1], [4]. Although many of the current tools require manual operations (e.g. [5]), the need for automatic forensic tools appeared in the last years. Individual events related to malicious activities are aggregated in [1] into an evidence graph for highlighting dependencies. Due to the large volume of network traffic, solutions like [6] only exploit network flows for profiling host behaviors. However, [7] highlights the use of full packet captures. Even if storing high volumes of such captures is possible, valuable subsets of captures have to be selected. We therefore propose a new device fingerprinting tool which aims at automatically inferring the type of a device (name, version, brand, series). Such a tool is really helpful for investigation and locating attack related device types (attack tools or devices known as vulnerable) or for detecting non authorized devices on the network. Furthermore, our tool relies only on the message structure and discards private information from captured messages since investigations can be faced with privacy issues [6]. We have targeted the specific case of VoIP traffic due to its popularity and the large spectrum of VoIP threats [8]. for several reasons. The typical use case for our work is as follows. VoIP Signalization data has been captured and has to be analyzed in order to detect if a rogue device (for instance a personal computer) has been injecting traffic. Although, a specific field in the SIP messages identifies the user-agent of a given device, this field can be easily spoofed. We have addressed this issue by developing our tool that identifies the device types by only analyzing the parsing structures.

The paper is structured as follows: the tool is described in section II and the corresponding anti-fingerprinting in section III. Experiments are detailed in section IV. Section V addresses fingerprinting prior works. Conclusion is given in section VI.

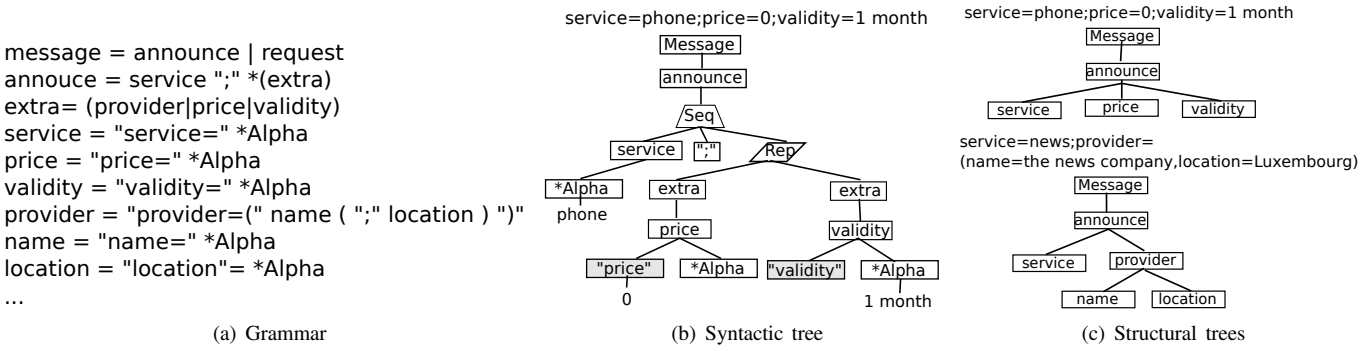
WIFS'2010, December 12-15, 2010, Seattle, WA, USA.
978-1-4244-4652-0/09/$25.00

II. MESSAGE STRUCTURE FINGERPRINTING

Our previous works [9], [10] introduced different ways for performing device fingerprinting. Although [10] assumes only the knowledge of message types, [9] considers the syntactic structure of the message and so the knowledge of the grammar protocol. The main conclusion of these works is that syntactic information is more helpful for identifying the type of remote devices but the main bottleneck of this approach is to handle huge syntactic trees (like those generated by a syntax parser) containing more than 800 nodes. Hence constructing and comparing such trees is a time consuming task limiting its application to cases where small volumes of data have to be processed. Our structural fingerprinting approach leverages the advantage of the syntactic fingerprinting and avoid this main issue.

A. Information structural representation

Similar to the syntactic fingerprinting, the structural fingerprinting relies on how a message is constructed and how it



(a) Grammar (b) Syntactic tree (c) Structural trees

Fig. 1. Toy example

is parsed. However, structural fingerprinting does not need to analyze the entire syntactic tree of a message but the structure of the provided information. In fact, the method does not consider the information value (content) although most of the fingerprinting relies on specific values in the message. To illustrate the case, we consider a toy example in figure 1. Figure 1(a) shows a partial ABNF [11] grammar of a protocol whose goal is to request or announce a service. When a host announces a service, the service which is announced is the essential information. The provider, the validity period and the price are optional (in brackets). Repetitions are signaled by a star (*). Figure 1(b) shows a message and its syntactic tree. Details of the construction is given in our previous work [9]. In brief, a node is created for each derived rule with its components as children nodes. For the repetition, a special node named *Rep* is used whereas sequences are indicated by *Seq*. Some details are omitted especially for strings represented by **Alpha* which is also a tree. However, the syntactic tree in 1(b) is clearly bigger than the first one in figure 1(c) which was constructed for the same message based on the information structure which can be viewed as the different fields or parts of a message. In fact, this tree contains not all syntactic information but only semantics about the information provided. The second tree represents another message and the goal of structural fingerprinting is to compare such trees to identify the sender. Devices are not forced to send exactly the same information thanks to optional rules and this has a direct impact of the information structure. This is due to liberty or ambiguities in the grammar or implementation errors.

B. Classification

For each message, its structural representation is constructed and the goal is to assign a type of device to each one. We assume a learning set L of N labeled samples $L = \{(s_1, t_1) \ldots, (s_N, t_N)\}$ where t_i is the device type corresponding to the structural representation s_i. Then for each t_j in a testing set T of M structural representations ($T = \{s'_1, \ldots, s'_M\}$), the goal is to assign a device type thanks to a classifier Ψ defined using the learning set.

We leveraged support vector machine (SVM) techniques especially multi-class classification since they show good accuracy with a limited overhead in various domain [12]. The kernel function needed for SVM is derived from a distance between two trees. It is the maximal cardinality of the subtree isomorphisms rooted in the original root. More theoretical details about this method can be found in our previous work [9] or in the original paper defining polynomial time distance complexity [13]. From a practical side, a path to a node p is the sequence of nodes from the root node to p. Computing the similarity between two trees is equivalent to retrieve P_1 and P_2, the sets of all possible paths for each tree and to compute the number of shared paths between these sets. This can be view as the intersection $I_{12} = P_1 \cap P_2$ even if the set intersection and set definition is not mathematical definitions because such sets can contain several times the same path. The complexity of this algorithm is $O(nm)$ where n and m are the number of nodes in the trees to compare (details in [9]).

C. Implementation

Protocols are usually defined by RFCs. Most forensic tools include support for existing protocols. For instance, wireshark supports many protocols and can be easily extended to new ones [14]. Hence, our implementation relies on the structural representation of messages provided by wireshark. Figure 2 illustrates how wireshark analyzes a message and there is clearly a hierarchical organization of information. The shaded parts represent which is extracted by our fingerprinting scheme in order to discard any private information like for example the *username*. We have decided to leverage the wireshark/tshark software in order to extend them and thus provide at the end a tool that many analysts are already familiar with.

More precisely, our approach takes benefits of tshark's (wireshark command line version) ability to generate the xml file representing the messages of a pcap file. We modified a little bit the program in order to filter out all information content and to only keep structure of a protocol given as parameter. Figure 3 describes the general architecture of our tool which is available at http://wiki.uni.lu/secan-lab/docs/psf.tar.gz

D. Privacy preserving

Our method only relies on structural information and discards all private information (the content of the message). Hence, structural fingerprinting can be qualified as privacy preserving. Thus, an open collaborative repository from various

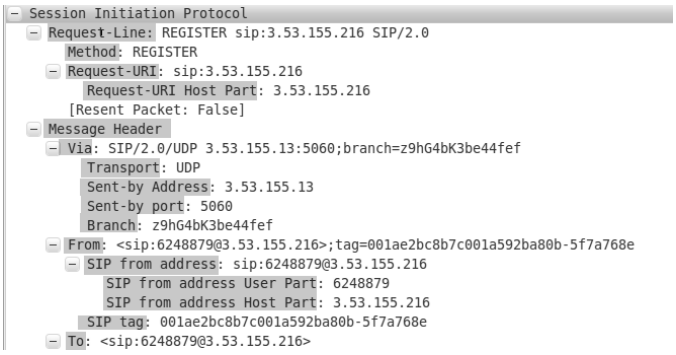


Fig. 2. A SIP message analyzed by Wireshark (partial view)

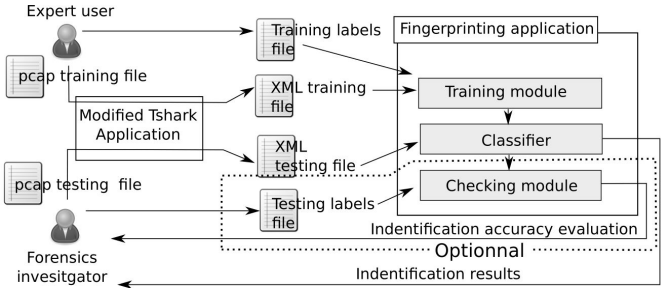


Fig. 3. Architecture

entities (companies, personal users) can be build for improving the accuracy.

III. ANTI-FINGERPRINTING

Structural fingerprinting was designed to be resistant to content modifications because they are easy to make. We have assessed a specific advanced attack against our proposed fingerprinting method and have developed a tool to implement it.

A. Single transformation

The goal is to transform a structural tree t_1 (assuming the second tree in figure 1(c)) in order to be the most similar to a tree t_2 (assuming the first tree in figure 1(c)) which was often sent by a different type of device. So, the goal is to define the function $\Phi(t_1, t_2)$ transforming t_1 into t_2 but whose the semantic is sufficiently preserved to perform the same action than the original message corresponding to t_1.

We consider P_1 and P_2 the set of all paths corresponding to the trees t_1 and t_2. The anti-fingerprinting technique has to transform P_1 in P_2. The generated message will naturally contain the shared paths (I_{12}). These paths are removed from original paths sets and produced two reduced sets P'_1 and P'_2. Different operations can be applied in order to maintain message semantics and provide good results:

- keep the paths from P'_1 since they are extra paths of P_1 but can contain needed information
- add the paths from P'_2 since they are extra paths of P_2 but can increase the performance of the anti fingerprinting

However, adding all paths of P'_2 may add no coherent information with the original semantics. For example (figure 1(c)), this transformation should add the *validity* information although it is not defined and this could entail usage error. In the same way, keeping all information is not always necessarily like for example the location of the service provider which may not really needed for using the service. Thus some paths of P'_1 may be skipped. It corresponds to the **`removing strategy`** since some of the paths of the original message are discarded:

$$remove(P'_1) = \{x \in P'_1, \rho(x) = 0\}$$

where $\rho(x) = 1$ if the information corresponds to path x (end node) is useful, 0 otherwise. In the same way, the ***adding strategy*** corresponds to only add extra paths which does not impact the general meaning and usage of the message:

$$add(P'_2) = \{x \in P'_2, \rho(x) = 0\}$$

Finally, the paths corresponding to the transformed message are: $\Phi(t_1, t_2) = I_{12} \cup add(P'_2) - remove(P'_1)$. The final stage aims to generate a message from the structural tree provided $\Phi(t_1, t_2)$ by adding content to nodes. Since I_{12} and $remove(P'_1)$ represent paths of the original tree, the original content can be retrieved to fill the new message. But $add(P'_2)$ represents new branches in the structural tree for which content needs to generated. Currently, we only copy content from the tree t_2 but it could be randomly generated for example.

Obviously, the definition the ρ function needs to be done by an expert user depending on the context (user and server needs). For example, filtering out all unnecessary information can be done but this task may be reduced if you consider only the information which differ most of the time.

B. In practice

A database of trees is needed in practice in order to transform tree into a coherent way. Basically, a protocol defines message types and so transforming a message from a type to a different one is not coherent. So, hiding the identity of a device consist in selecting the same message type from another device type. If the goal is just to fool the identification and not really spoofing a real type, this other type should vary regularly. Thus, assuming a message type m and device type d, constructing a set P of messages matching these criteria is essential. Then, the transformation of a message into type d has to select one message from P to perform the transformation Φ. Different strategies can be employed: random (choose a random message and its corresponding tree), choose the less similar message, choose the most similar message. We applied the latter since it minimizes the number of transformation and so reduces drastically the manual analysis for defining ρ (less information are concerned).

IV. EXPERIMENTS

A. Dataset

We have used a SIP [15] dataset [16], [10], [9] in order to allow for a direct and meaningful comparison. This one was

Device Name	#mesg	depth			#nodes		
		Max	Min	Avg	Max	Min	Avg
Asterisk_v1.4.21	1081	4 (28)	3 (23)	3.72 (25)	54 (2517)	30 (883)	36.00 (1284)
Cisco-7940_v8.9	168	4 (25)	3 (23)	3.98 (24)	50 (2784)	31 (812)	34.31 (1352)
Thomson2030_1.59	164	4 (28)	3 (23)	3.64 (24)	50 (2576)	29 (793)	37.12 (1391)
Twinkle_v1.1	195	4 (25)	3 (23)	3.39 (23)	49 (2457)	29 (805)	35.75 (1299)
Linksys_v5.1.8	195	4 (28)	3 (23)	3.52 (25)	51 (2783)	29 (852)	36.11 (1248)
SJPhone_v1.65	288	4 (30)	3 (23)	3.40 (24)	46 (2330)	26 (951)	31.81 (1133)

TABLE I
TREE STATISTICS (NUMBERS IN BRACKETS ARE STATISTICS WHEN USING REAL SYNTACTIC TREE BASED ON THE ABNF GRAMMAR)

generated using 6 SIP distinct device types. The statistics of the dataset are given in table I including tree characteristics. These latter are compared with a syntactic fingerprinting based on the ABNF grammar (numbers in brackets). In both cases, the depth and the cardinality are not dependent on the type of device but there is a significant difference between the structural fingerprinting and the syntactic fingerprinting. Since the computational complexity is quadratic with respect to the number of nodes when the kernel function is computed, the structural fingerprinting is clearly better. There are about 35 times more nodes in syntactic tree.

B. Metrics

Since fingerprinting is a classification problems, standard metrics of this domain are used [17]. Assuming N is the total number of messages which are classified and K is the number of distinct device types. Assuming x_d, the number of trees corresponding to a particular device type $d \in D$, $z_{d_2 d_1}$ the number of trees of types d_2 which were classified as d_1, the sensitivity evaluates the number of trees of a given type d which were assigned to the right type:

$$sens(d) = z_{dd}/x_d \tag{1}$$

The accuracy is the proportion of trees which are assigned to the correct type:

$$acc = \sum_{d \in D} z_{dd}/N \tag{2}$$

Sine there is a distinct sensitivity value per device type, the average sensitivity will be computed. This is a complementary metric of accuracy since a high accuracy is possible by classifying only the most represented types but the average sensitivity is very low in this case.

C. Fingerprinting

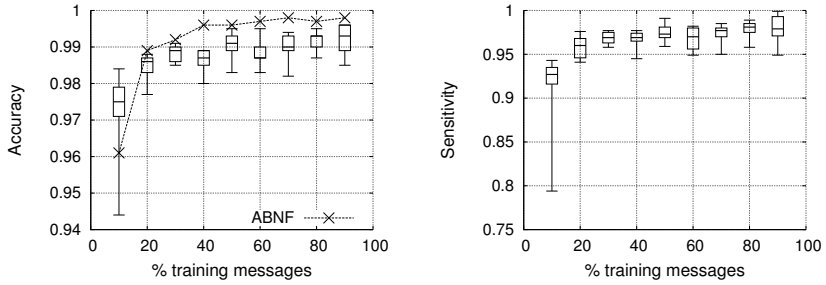
The first experiment aims to assess the performances of our approach. device. Our dataset is divided in two parts: one for the learning and one for the testing. In order to evaluate the efficiency, the percentage of messages used for the training varies from 10 to 90% in figure 4. Each test is run 10 times and the quartiles values are plotted. The extrema values are plotted and the box delimits 50% of the observations with the median value as the horizontal bar. The rest of the observations are outside the box (25% below, 25% above). The line represents our previous approach (syntactic fingerprinting) [9] by presenting only the median value for clarity but results variations are in the same order of magnitude of the structural fingerprinting. The main observation is that our novel fingerprinting approach provides good performances regarding the different metrics and compared with [9] (ABNF line). About 98% of device can be correctly classified and all types of device can be rightly identified with a precision of 95% (figure 4(b)) where only 20% of messages are extracted for training the system which is very low for a classification problem.

D. Anti-fingerprinting

We have investigated the counter methods that an advanced attacker might use. The experiment focuses on `REGISTER` messages since they were always correctly classified by our fingerprinting approach (accuracy = 100%) and since they are required for login in to the server. Because our fingerprinting technique is able to identify the type of a device once it connects, it is important to assess the counter method that can be used against it. Firstly, differences between trees of different types of devices were extracted. The corresponding paths can be divided into two categories:

- optional information like the display format defining a nice format for user information like his name
- mandatory information for sending a correct response to the device.

This last category includes only port information about the device and the proxy in order to route the reply back. Assuming that only divergent paths concerned by 30% of the pairs of trees are kept, this concerns only optional informations. So for all such path $\rho(x)$ equals 1. As explained in section III, a message can be transformed using different strategies (***adding*** or ***removing***). They were tested by doing on of these actions or by combining them as shown in figure 5. For each device type d, devices from all other types apply this strategy to spoof the d type and the general accuracy for all these



(a) Accuracy (b) Sensitivity

Fig. 4. Impact of the number of messages used for training

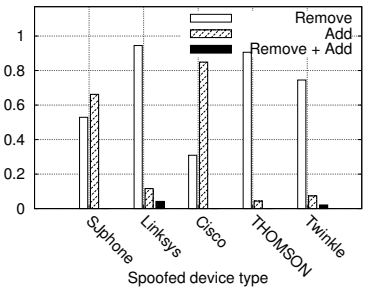


Fig. 5. Device type spoofing – Accuracy for REGISTER messages

Origin \ Target	Cisco	SJphone	Thomson	Twinkle	Linksys
Cisco		0.114 0.661	0.114 **0.000**	0.174 **0.000**	0.205 0.483
SJphone	0.082 0.882		0.155 **0.000**	0.088 **0.000**	0.147 **0.000**
Thomson	0.113 1.000	0.185 0.711		0.232 **0.000**	0.256 0.367
Twinkle	0.144 0.883	0.063 1.000	0.201 **0.000**		0.106 0.950
Linksys	0.149 1.000	0.128 1.000	0.191 0.300	0.064 1.000	

TABLE II
ANTI-FINGERPRINTING PER DEVICE REGISTER MESSAGES – UPPER LINE: AVERAGE NUMBER OF TRANSFORMATIONS NEEDED TO TRANSFORM ONE TYPE (IN ROW) INTO ANOTHER ONE (IN COLUMN) – BOTTOM LINE: ACCURACY PER DEVICE

devices is plotted in figure 5. The ***adding strategy*** is more efficient than the ***removing strategy*** to defeat the fingerprinting but for some types like Cisco, removing information is better. This is dependent of the trees emitted by the devices to spoof. For instance, if such trees contain many information, adding information is easy. In brief, applying the combination of these strategies is the better manner (black bars close to zero).

The next experiment considers an initial device type (the origin) and the targeted device type as shown in table II. For each message of the type mentioned in the row, the message is transformed into the type mentioned in column. The upper line of each row indicates the number of average transformation factor applied where the factor of a message transformation is the number of needed operation divided by the original cardinality of the tree. The bottom line of table represents the accuracy. The anti-fingerprinting provides good results as highlighted by bold fonts. The ability to defeat fingerprinting is not linearly dependent to the number of transformations executed because sometimes few transformations are sufficient like the Twinkle type spoofed by SJphone device and sometimes more transformations are applied (Twinkle type spoofed by Thomson devices). In the same manner, applying many modifications to trees is not always efficient: for example when Thomson devices spoof the Linksys type. Thus, there is not a direct dependency between the number of transformations and the accuracy .

Hence, countering these attacks is hard but our evaluation shows that some devices types are more resistant to device spoofing and that some of them may be really advised against to use. Moreover, this attack implies that you have to known the type to spoof (which cannot be evident to know for an attacker) and also having a database of messages for this type. Thus the attack has to be carefully designed and attackers cannot always fulfill these requirements. For example, a company can change a bit the configuration of all devices to infer on the message structure without publish this configuration publicly.

V. RELATED WORK

The pioneer work [18] highlights that protocol implementation variation exists due to unclear or permissive specification. The passive techniques only monitor traffic as for example [19] which looks for specific pattern in TCP headers to infer the operating system (OS). The active techniques sends specific requests to host in order to get specific response containing discriminative pattern. NMAP [20] implements this scheme for OS fingerprinting. Although human expertise is generally needed to define the requests to send, [21] proposes a novel approach for determining them automatically. Traffic fingerprinting aims to infer the nature of traffic, *i.e.*, the protocol [22] or the class (Web, P2P, Chat..) [23]. Our fingerprinting approach is fine grained and complementary since the goal is to infer the type of device using a certain protocol. In this domain, [24] is dedicated to infer the web server type of users by analyzing the headers especially their order. SIP fingerprinting was also studied previously in [25] by analyzing one specific field which is not rightly generated (bad random

generator) by some devices. Other SIP fields are analyzed in [26] but need an active probing unlike the one presented in this paper. Furthermore, our approach does not rely on field values because they can easily faked and they contain private data which we discard voluntarily due to privacy issues. We have addressed temporal fingerprinting in [16] whose the results are a little bit under (between 70 and 80%) the ones obtained with our syntactic approach (around 99%) [9]. The syntactic trees were also employed in [27]. As we argued in this paper, the size of syntactic trees entails a huge computation time wheres structural trees are 35 times smaller.

VI. CONCLUSION

In this paper, we have proposed a new forensic tool for identifying the device types in captured VoIP traffic. Our approach is generic and can be adapted to most of the protocols for which wireshark dissectors exist. We have assessed the performance and obtained very good results. Advanced counter attacks have been discussed and their impact on the proposed scheme was properly identified. Our future work will address parallel and multiple protocol fingerprinting for forensic purpose.

Note: figures include wikimedia contents (visit http://commons.wikimedia.org for license information).

REFERENCES

[1] W. Wang and T. E. Daniels, "A graph based approach toward network forensics analysis," *ACM Trans. Inf. Syst. Secur.*, vol. 12, no. 1, pp. 1–33, 2008.

[2] N. Liao, S. Tian, and T. Wang, "Network forensics based on fuzzy logic and expert system," *Computer Communications*, vol. 32, no. 17, pp. 1881–1892, 2009.

[3] "A new approach to china," http://googleblog.blogspot.com/2010/01/new-approach-to-china.html.

[4] B. Carrier, "Defining digital forensic examination and analysis tools using abstraction layers," *International Journal of Digital Evidence*, vol. 1, 2002.

[5] "Safeback," http://www.forensics-intl.com/safeback.html.

[6] J. McHugh, R. McLeod, and V. Nagaonkar, "Passive network forensics: behavioural classification of network hosts based on connection patterns," *SIGOPS*, vol. 42, no. 3, pp. 99–111, 2008.

[7] V. Corey, C. Peterman, S. Shearin, M. S. Greenberg, and J. V. Bokkelen, "Network forensics analysis," *IEEE Internet Computing*, vol. 6, pp. 60–66, 2002.

[8] A. D. Keromytis, "Voice over ip: Risks, threats, and vulnerabilities," in *International Conference on Information Systems Security (ICISS)*, 2009.

[9] J. François, H. Abdelnur, R. State, and O. Festor, "Advanced Fingerprinting For Inventory Management," INRIA, Research Report, 2009.

[10] J. François, H. Abdelnur, R. State, and O. Festor, "Automated behavioral fingerprinting," in *Recent Advances in Intrusion Detection*, 2009.

[11] D. H. Crocker and P. Overell, "RFC: Augmented BNF for Syntax Specifications: ABNF," United States, 1997.

[12] L. Wang, Ed., *Support Vector Machines: Theory and Applications*, ser. Studies in Fuzziness and Soft Computing. Springer, 2005, vol. 177.

[13] A. Torsello, D. Hidovic-Rowe, and M. Pelillo, "Polynomial-time metrics for attributed trees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 7, pp. 1087–1099, 2005.

[14] A. Orebaugh, G. Ramirez, J. Burke, and L. Pesce, *Wireshark & Ethereal Network Protocol Analyzer Toolkit (Jay Beale's Open Source Security)*. Syngress Publishing, 2006.

[15] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "RFC: SIP: Session Initiation Protocol," United States, 2002.

[16] J. François, H. Abdelnur, R. State, and O. Festor, "Automated Behavioral Fingerprinting," in *12th International Symposium on Recent Advances in Intrusion Detection (RAID)*, St Malo France, 2009.

[17] P. Baldi, S. Brunak, Y. Chauvin, C. A. Andersen, and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: an overview." *Bioinformatics*, vol. 16, no. 5, pp. 412–24, 2000.

[18] Douglas Comer and John C. Lin, "Probing TCP Implementations," in *USENIX Summer*, 1994, pp. 245–255. [Online]. Available: citeseer.ist.psu.edu/article/comer94probing.html

[19] "P0f," http://lcamtuf.coredump.cx/p0f.shtml.

[20] G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. USA: Insecure, 2009.

[21] J. Caballero, S. Venkataraman, P. Poosankam, M. G. Kang, D. Song, and A. Blum, "FiG: Automatic Fingerprint Generation," in *Distributed System Security Conference*, 2007.

[22] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. M. Voelker, "Unexpected means of protocol inference." in *Internet Measurement Conference*, 2006.

[23] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: myths, caveats, and the best practices," in *ACM CoNEXT Conference*, 2008.

[24] "Httpprint," http://www.net-square.com/httprint/httprint_paper.html.

[25] H. Scholz, "SIP Stack Fingerprinting and Stack Difference Attacks," *Black Hat Briefings*, 2006.

[26] H. Yan, K. Sripanidkulchai, H. Zhang, Z. yin Shae, and D. Saha, "Incorporating Active Fingerprinting into SPIT Prevention Systems," *Annual VoIP Security Workshop*, 2006.

[27] H. Abdelnur, R. State, and O. Festor, "Advanced Network Fingerprinting," in *Recent Advances in Intrusion Detection*. Springer, 2008.