Advanced Fingerprinting For Inventory Management

Jérôme François, Humberto Abdelnur, Radu State, Olivier Festor

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Advanced Fingerprinting For Inventory Management

Jérôme François — Humberto Abdelnur — Radu State — Olivier Festor

N° 7044

Septembre 2009

ISSN 0249-6399 ISRN INRIA/RR--7044--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
NANCY - GRAND EST

Advanced Fingerprinting For Inventory Management

Jérôme François*, Humberto Abdelnur† , Radu State‡ , Olivier Festor§

Thème : Device Fingerprinting
Équipe-Projet Madynes

Rapport de recherche n° 7044 — Septembre 2009 — 24 pages

Abstract: Identifying the protocol stack or the device version of remote equipment is a powerful tool for security assessment and network management. This paper proposes two novel fingerprinting techniques based on the syntactic tree representation of messages. The first leverages the support vector machines paradigm and needs a learning stage while the second one executed in an unsupervised manner thanks to a new classification algorithm. The approaches are validated through extensive experimentations and show very good behaviors.

Key-words: fingerprinting, inventory management, syntactic tree, SVM, learning machines.

* jerome.francois@loria.fr
† humberto.abdelnur@loria.fr
‡ radu.state@loria.fr
§ olivier.festor@loria.fr

Centre de recherche INRIA Nancy – Grand Est
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

Fingerprinting avancé pour l'inventaire des équipements

Résumé : L'identification des équipements d'un réseau se révèle un atout intéressant dans le domaine de la supervision et de la sécurité des réseaux. Ce papier introduit deux nouvelles méthodes de fingerprinting reposant sur la construction et la comparaison d'arbres syntaxiques. La première met en œuvre les machines à vecteurs supports nécessitant une phase d'apprentissage alors que la seconde est totalement non supervisée et se base sur un nouvel algorithme de classification. De nombreuses expérimentations valident les différentes approches.

Mots-clés : fingerprinting, inventaire, machines à vecteurs de support, arbres syntaxiques

1 Introduction

Assuming a protocol, device fingerprinting aims to determine exactly the device version or the protocol stack implemented by equipment. It is a challenging task covering many domains like security or network management. Identifying the devices helps to get a detailed view of alive equipments on a network for planning future actions when needed. For example, if a new security flaw is discovered for some device types, patching them has to be fast due to zero-day attacks but locating them is not always obvious. Besides, detecting abnormal devices on a network is very useful for disconnecting rogue equipment or for tracking copyright infringements. Furthermore, some authentication systems check the device type like for example on a VoIP (Voice over IP) operator allowing only some specific hardphones. Classical management solutions like SNMP [1] by installing additional software on equipment (agent) are not always feasible since often some machines are not owned by the company itself (personal or partner company devices) or their software does not support. Finally, fingerprinting its customers could be valuable for a company. For instance, a VoIP operator can offer additional services to its customer particularly applications by sending customized advertisement based on the brand and the version of the phone.

Most of the current approaches for device identification is related to some specific field value of the protocol grammar. For instance, the SIP [2] VoIP protocol includes the device identity in the User-Agent field which can be easily omitted or modified by an attacker. Hence, new generic techniques considering the wole message are required. In this paper, each entire message is concerned and represented as a syntactic tree. Relying on underlaying differences of the content and structure of such trees, the two main contributions of this paper are :

- a new supervised syntactic fingerprinting technique which aims to precisely identify equipment (device type) (*Problem 1*),
- a novel unsupervised syntactic fingerprinting technique looking for the number of distinct device types running a given protocol and its distribution (*Problem 2*)

The second method gives general indication about the device type distribution for a given protocol and can exhibit heterogeneity or homogeneity. For instance, when a new service is deployed, proposing a support service is a real benefit for helping the users (company networks) or for doing business (operator). Hence, unsupervised fingerprinting helps to assess its complexity and its feasibility (number of distinct device versions to support). Generally, few software are supported or proposed and most users install other ones. The number of device types used and their distribution is a good hint to evaluate the security risk because the higher the number of non supported version, the higher becomes the risk. Moreover, these techniques are passive *i.e.*, without any interaction with the fingerprinted equipment which avoids to be detected and unnecessary overloading of the network and fingerprinted devices. Assuming majority of messages are not faked, unsupervised fingerprinting can be the foundation of the supervised system since an user can identify manually some components of the discovered clusters.

The next section formally describes the two problems. Section III depicts the general operation of our approach. The message representation is detailed

in section IV before giving the details of the classification methods in section V. Section VI is dedicated to present extensive results. The related work are given in section VII before the conclusion and directions for future work.

2 Problem definition

We consider K different device types represented by the set $D = \{d_1, \ldots, d_K\}$ and a testing set of N messages $T = \{t_1, \ldots, t_N\}$. If the training stage exists, M messages are collected and labelled correctly to form the set: $L = \{l_1, \ldots, l_M\}$. The function $real(t_i) : T \cup L \rightarrow D$ returns the real identifier (device type or implementation stack) of a message.

2.1 Problem 1

The goal is to compute the classifier $\Omega_L : T \cup L \rightarrow D$ assigning the right device identity to each $l_i \in L$ *i.e.*, $\Omega_L(l_i) = real(l_i)$. The same function is then applied to each $t_i \in T$ and is expected to return $real(t_i)$.

2.2 Problem 2

In this scenario, no labelled messages are available and thus training is impossible. The messages have to be directly divided into groups by a classifier $\Psi_T : T \rightarrow \mathbb{N}$. Because no labels can be derived from a training process, the goal is to find the number of device type, *i.e.*, K, and create consistent groups containing in the optimal case only messages of a single device type. Thus, the targeted result is :

$$|\Psi[T]| = K$$

$$\forall < t_i, t_j >,\ real(t_i) = real(t_j) \Leftrightarrow \Psi_T(t_i) = \Psi(t_j)$$

$$\forall < t_i, t_j >,\ real(t_i) \neq real(t_j) \Leftrightarrow \Psi_T(t_i) \neq \Psi(t_j)$$

3 Fingerprinting framework

3.1 SIP overview

Our evaluation is based on SIP protocol [2] since this signaling protocol is gaining support and the number of compliant devices is skyrocketing. Hence, fingerprinting could support these new applications as explained in the introduction. In a few word, the SIP protocol is a text protocol with several primitives (`INVITE`, `NOTIFY`, `REGISTER`, `ACK`, `CANCEL`...) and response codes (three digits number whose the first is between 1 and 6). SIP illustrates also the possibility to develop reinforced fingerprinting-based authentication since its complexity entails some authentication flaws [3].

3.2 Architecture

The architecture is depicted on figure 1. The messages are collected through SIP proxies. For each of them, the syntactic tree is constructed based on the protocol grammar. This tree represents its signature. In the case of the unsupervised

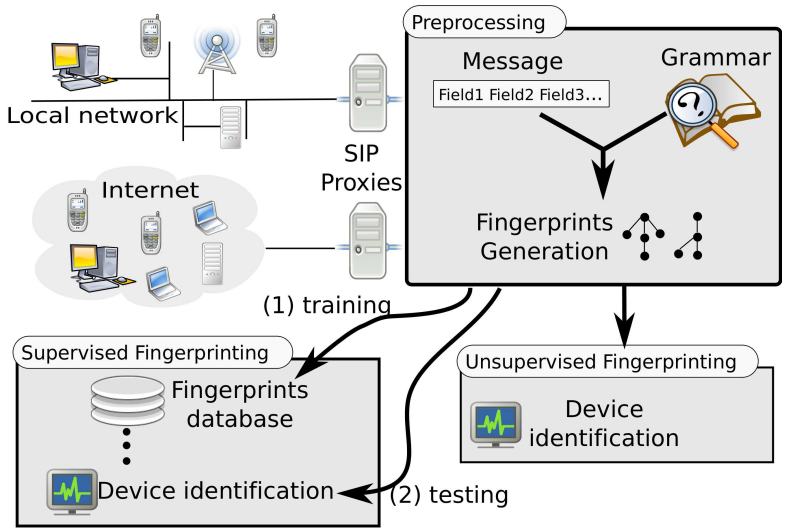


Figure 1: Fingerprinting architecture

fingerprinting, these trees are directly grouped by computing the classifier Ψ. Otherwise, the supervised technique needs two stages :

- the learning stage (1) : the signatures are stored in a database and used for computing the classifier Ω;
- the testing phase (2) : each new generated signature is taken as an input of Ω to assign a specific label device type to the message.

Because a fingerprints of a device type is its general characterization, the proposed scheme implies the following definition : a fingerprint of a device type is the set of signatures belonging to this type in the training set. For the unsupervised technique, the fingerprint of a type is the entire corresponding cluster obtained after the classification.

4 Attributed trees

4.1 Distances

Our techniques use the metrics defined in [4] and this section gives an overview of the theory. An attributed graph is defined by the tuple $G = (V, E, \alpha)$ where V are the different nodes, E the different edges and α is a function such that $\alpha(s)$ gives some characteristics about the node s. A tree is a special kind of graph without cycle. Two trees T_1 and T_2 are considered isomorphic if there exists a bijection ϕ mapping every node in T_1 to every node in T_2 while keeping the same structure (the nodes are connected in the same way). The trees have a subtree isomorphism ϕ if there exists two subtrees T'_1 and T'_2 which are isomorphic. Their similarity is measured as :

$$W(\phi, T'1, T'2) = \sum_{u \in T'_1} \sigma(u, \phi(u))$$

```
Message =  Request  SP *Header SP 0*1Body        Alpha = %x41-5A / %x61-7A       ; A-Z / a-z
Request = Invite / Notify / Cancel               HCOLON =  *SP ":" *SP
Invite = □34INVITE□35                            SP =  %x20                ; space
Cancel = □34CANCEL□35                            Accept = "Accept" HCOLON *Alpha "."
Notify = □34NOTIFY□35                            Date = "Date" HCOLON *Alpha "."
Header = Accept / Date / Call-id / User-Agent    Call-Id = "Call-Id" HCOLON *Alpha "."
Body = *Alpha                                    User-Agent = "user-Agent" HCOLON *Alpha "."
```

Figure 2: Grammar

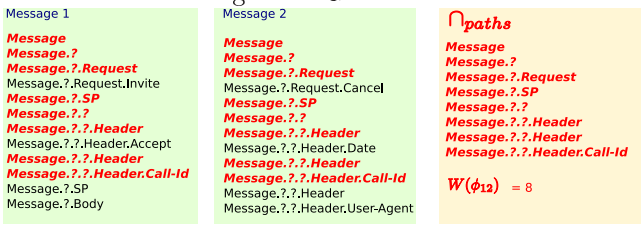


Figure 3: Intersection of ancestor paths

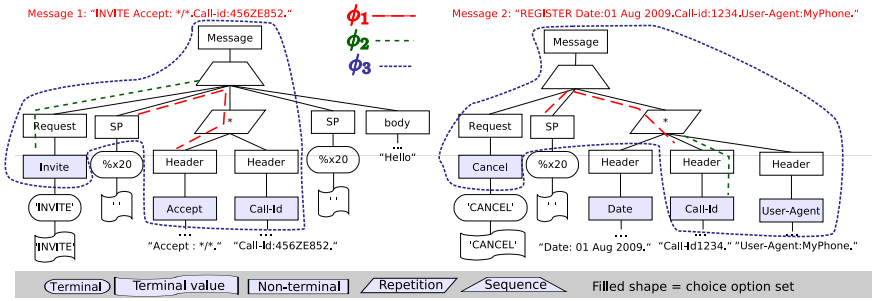


Figure 4: Syntactic trees of 2 messages

where σ is the comparison function between the characteristics (α function) of two nodes. Furthermore $W(\phi_{12})$ is the maximum similarity between two isomorphic trees of T_1 and T_2.

Although classical techniques compute the similarity between two trees by counting the number of transformations (delete, add or subsitute) required to transform the trees into isomorphic ones, the authors of [4] emphasize that resolving this problem is NP-complete unless adding some specific constraint (nodes ordering for instance) to get a polynomial-time complexity. Hence, they propose to define four novel distance metrics (two normalized and two non-normalized) between trees leading also to a polynomial complexity. Keeping in mind to compare the efficiency of our technique with normalized and non-normalized metrics, three of them were selected after preliminary experiments since they provided the best results :

$$d_1(T_1, T_2) = |T_1| + |T_2| - 2W(\phi_{Max\_12})\ (non - normalized) \qquad (1)$$

$$d_2(T_1, T_2) = 1 - \frac{W(\phi_{Max\_12})}{max(|T_1|, |T_2|)} \ (normalized) \tag{2}$$

$$d_3(T_1, T_2) = 1 - \frac{W(\phi_{Max\_12})}{|T_1| + |T_2| - W(\phi_{12})} \ (normalized) \tag{3}$$

where $|T|$ is the number of nodes of the tree T.

4.2 Syntactic trees

A syntactic tree is an attributed tree built from a message and the Augmented BackusNaur Form (ABNF) [5] protocol grammar. The figure 2 shows a partial grammar of a simple protocol (far from SIP). The non-terminal elements are those which can be derived into other ones (Message, Request) contrary to terminals representing a fixed sequence of or a range of possible characters (terminal values are real values in the message). The elements prefixed by "*" are repeated whereas those separated by "/" are alternatives. Otherwise, the different elements form a sequence.

Thus, each message is mapped to a syntactic tree like in figure 4. A node is created from each terminal value and linked to a parent node representing the sequence, the repetition or the non-terminal from which it is derived. The figure 4 shows two partial syntactic trees.

The syntactic trees are rooted. Thus, two trees are isomorphic if the relationship between parent and child nodes is also kept. Furthermore, terminal values are not taken in account because the containing information is highly dependent of a specific session (call-id, date...).

Some potentially large structure can be derived for many grammar rules as for example the construction of a character sequence built by the expression `*Alpha`. Thus, two subtrees with different `Request` or `Header` branches can contain such a structure whereas their meaning is probably different. Hence, these relatively large structures could bias the similarity measure. The solution is to consider the path of a node to evaluate their similarity. The path is all the nodes between the root node and the considered node. Therefore, the characteristics of a node n defined by $\alpha(m)$ is the tuple $< name_m, path_m >$ with $path_m$ the path. The name of a node is its non-terminal name or "?" otherwise (sequence or repetition). We propose a binary similarity (σ) between two nodes imposing that two similar nodes have to share similar ancestor nodes, *i.e.*, the same path, and the same name. Assuming that par_n returns the parent node of n and r the common root of the trees, the similarity between two nodes u and v can be totally defined as:

$$\sigma(u, v) = \begin{cases} 1 & if\ u = r \wedge v = r \\ 1 & if\ name_u = name_v \wedge \sigma(par_u, par_v) = 1 \\ 0 & else \end{cases} \tag{4}$$

If messages can be derived from different first rule, adding a generic root node r is feasible but leading to a similarity equals zero. Three subtree isomorphisms are represented in figure 4. The subtrees associated to the first ones contain exactly the same node and so $W_{\Phi_1} = 4$. Because sequence and repetition are equivalent in the ancestors path (question mark), the second isomorphism generates two

trees sharing one similar node. However, $W_{\phi_2} = 0$ due to different ancestors path. Finally, $W(\phi_3) = 8$ because the subtrees ares the same except for two nodes (`Accept` and `user-Agent`). The `Call-Id` are matched because there is no order on the nodes.

Though, the first isomorphism is clearly suboptimal as the the subtrees are not rooted on the global root node while the pair of nodes share the same ancestors. Hence, finding the isomorphism candidates has to consider the paths of all node of a tree as illustrated in figure 3. The creation of the lists containing these paths can be done easily during the creation of the trees. The optimal isomorphic subtree is built from all shared paths by the messages. Thus, the subtrees are the intersection $\cap_{paths}$ of similar paths calculated by the algorithm 1 whose the design is straightforward. Indeed, one iteration loops over all paths of the first tree t_1 and looks for the same path in the second one t_2. The line 15 is extremely important for avoiding to take in account the same path twice. For instance, inverting the messages on 3 implies three paths `Message.?.?.header` without this line. Since this algorithm iterates over each path of t_2 for each path of t_1, the complexity is in $O(t_1 t_2)$

Because all paths are rooted on the same node, the prefix of each path (all nodes except the last one) always equals another one. Hence, the similarity is exactly the number of elements in the intersection[1] : $|\cap_{paths}|$. Thus, the similarity between the example messages is eight.

Algorithm 1 similar_paths(t_1,t_2)

1: $res = [\,]$ is the intersaction of shared path initialized to an empty list
2: $paths(t)$ return the paths list of the tree t
3: $l.add(e)$ adds e to the list l
4: $l.remove(e)$ remove e from the list l
5: $len(l)$ is the length of the list l
6: $c_1 = paths(t_1)$
7: $c_2 = paths(t_2)$
8: **for** $c \in c_1$ **do**
9: $\quad i \leftarrow 1$
10: $\quad bool \leftarrow TRUE$
11: $\quad$ **while** $bool \wedge i < len(c_2)$ **do**
12: $\quad\quad$ **if** $c = c_2[i]$ **then**
13: $\quad\quad\quad bool = FALSE$
14: $\quad\quad\quad res = res.add(c)$
15: $\quad\quad\quad c_2.remove(c)$
16: $\quad\quad$ **end if**
17: $\quad\quad i \leftarrow i + 1$
18: $\quad$ **end while**
19: **end for**

[1] This is not a mathematical intersection since a path can be represented several times

5 Fingerprinting approaches

5.1 Supervised classification

Supervised learning techniques are potential solutions for resolving the *Problem 1* since some of the training samples are available. We chose to carry out the recent support vector machines (SVM) technique because it outperforms the classification accuracy in many domains with limited overhad [6]. SVM were already exhibited in network security monitoring and intrusion detection [7, 8]. However, none of them introduces the combination of SVM and syntactic trees. Basically designed for two classes classification, SVM techniques were rapidly extended to multi-class problems like the *Problem 1*. One-to-one classification [9] is known for providing a good accuracy with a low computational time [10].

The method strives to find an hyperplane to highly separate the data points (trees) of different classes (device types). For the one-to-one method, an hyperplane is constructed for each pair of distinct classes as illustrated by the simple example on figure 5 where H_{i-j} is the hyperplane separating points from class i and j. Then, when the new point \$ has to be assigned to a class, its side-position from each hyperplane is computed to judge the more suitable class. Considering the example, the following results are obtained for each hyperplane :

- H_{U-X}: \$ class is U,
- H_{O-U}: \$ class is O,
- H_{O-X}: \$ class is O.

The final decision relies on a voting mechanism where the most represented class, O, is chosen.

Most of the time, the data points are not linearly separable, so they are casted in high dimensional feature space using a mapping function φ. Determining the hyperplanes is the main task. Assuming the notations introduced in previous sections, for each pair of device types $< d_l, d_p >$, the corresponding hyperplane is specified by the vector w^{lp} and a scalar b^{lp}. It has to separate and to be as far away as possible from the trees belonging to d_l and d_p denoted as :

$$\begin{aligned} T_l &= \{t_i | real(t_i) = d_l\} \\ T_p &= \{t_i | real(t_i) = d_p\} \end{aligned} \tag{5}$$

Hence, the resulting problem constraints is defined as :

$$\begin{aligned} &\forall t_i \in \{T_l \cup T_p\} \\ \langle \varphi(t_i) \cdot w^{lp} \rangle + b^{lp} &\geq 1 - \xi_{t_i}^{lp},\ if\ real(t_i) = d_l \\ \langle \varphi(t_i) \cdot w^{lp} \rangle + b^{lp} &\leq -1 + \xi_{t_i}^{lp},\ if\ real(t_i) = d_p \end{aligned} \tag{6}$$

where ξ are slacks variables allowing some misplaced points when a total separation is not possible. For example, on figure 5, if a point O in the surrounding of points X, it is impossible to really separate them.

The optimization problem implies that the points have to be as far away as possible from the hyperplane :

$$\min_{w^{lp}, b^{lp}, \xi_{t_i}^{lp}} \frac{1}{2} ||w^{lp}|| + C \sum_{t_i \in \{T_l \cup T_p\}} \xi_{t_i}^{lp} \tag{7}$$

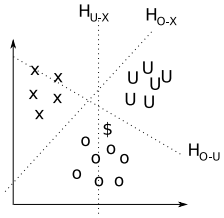


Figure 5: SVM one-to-one classification, $ is a new point to assign

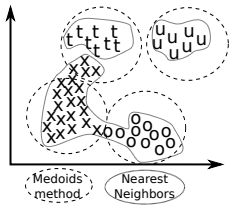


Figure 6: Common classification problems

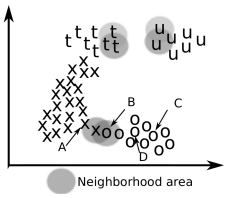


Figure 7: Rock classification

where C is constant representing the trade-off between the minimum classification errors and maximal margins.

The function φ is essential but defining it is often hard. That is why the kernel trick is usually exploited by defining a function $K(x, y)$ directly computable between two trees and also equals to $\langle\ \varphi(x_i).\varphi_(x_j)\ \rangle$. Then, the optimization problem is turned into its dual by introducing the Lagrangian for computing the vector w^{lp} :

$$max \sum_{t_i \in \{T_l \cup T_p\}} \alpha_{t_i}^{lp} - \frac{1}{2} \sum_{\substack{t_i \in \{T_l \cup T_p\} \\ t_j \in \{T_l \cup T_p\}}} \alpha_{t_i}^{lp} \alpha_{t_j}^{lp} \rho_{t_i}^{lp} \rho_{t_j}^{lp} K(t_i, t_j) \tag{8}$$

subject to:

$$\begin{gathered} \sum_{t_i \in \{T_l \cup T_p\}} \alpha_{t_i}^{lp} \rho_{t_i}^{lp} = 0 \\ 0 \leq \alpha_{t_i}^{lp} \leq C,\ t_i \in \{T_l \cup T_p\} \end{gathered} \tag{9}$$

with :

$$\rho_{t_i}^{lp} = 1\ if\ t_i \in T_L, -1\ if\ t_i \in T_P \tag{10}$$

The scalar b^{lp} is calculated thanks the support vector trees (SV^{lp}) which corresponds to the points on the border of each group mathematically defined as the trees t_{sv} such that $\alpha_{t_s v}^{lp} \neq 0$:

$$b^{lp} = \frac{1}{|SV^{lp}|} \sum_{t_i \in SV^{lp}} (\rho_{t_i}^{lp} - \sum_{t_j \in \{T_l \cup T_p\}} \alpha_{t_i}^{lp} \rho_{t_j}^{lp} K(t_j, t_i)) \tag{11}$$

Once the learning phase has solved this optimization problem, each tested tree l_m is given as the input of the decision function :

$$f_{lp}(l_m) = \sum_{t_i \in \{T_l \cup T_p\}} \alpha_{t_i}^{lp} \rho_{t_i} K(t_i, l_m) + b^{lp} \tag{12}$$

Then, the sign of the result indicates the likelyhood of l_m to belong to d_p or d_l. Since, only one hyperplane is defined for each pair of class, these functions are symmetrical $f_{lp} = -f_{lp}$. So only $\frac{K(K-1)}{2}$ hyperplanes and functions have to be found out.

The distance adaptation is compulsory because the kernel function is a similarity measure. For the normalized metric, the definition is straightforward as the similitude is equivalent to one minus the distance. For the nonnormalized, we derived a kernel close to the Gaussian one :

$$d'_1 = e^{-0.01 d_1} \quad d'_3 = 1 - d_3$$

5.2 Unsupervised classification

5.2.1 ROCK and QROCK

Whereas our distance measures are based syntactic trees which can be viewed as categorical data, most well known techniques such as K-means, K-medoids or density based algorithms are suited for numerical values [11]. Therefore, new kind of unsupervised approaches dedicated to categorical data can be found in the literature as for instance the ROCK algorithm [12]. This algorithm is based on a graph representation where two nodes are linked if they share at least one common neighbor. Two points are neighbors if their inter-distance is less than a threshold τ. It is an agglomerative clustering technique and so each unique point is a cluster at the beginning. Then, the clusters are grouped together based on a score measure which measures the linkage degree (the number of shared neighbors) comparing with the estimation of the maximal number of possible shared neighbors.

Figure 6 highlights the results of main types of clustering. Figure 6 points out the bad accuracy of medoid clustering methods which group points around another one. The main disadvantage is that these techniques assume similar points distributed within a common shape (spherical most of the time) close to a medoid. Other well-known techniques consider each point individually. For example, the nearest neighbors technique results is plotted on figure 6: the clusters of the pair of closest points are merged until the corresponding minimal distance is higher than a threshold. The main advantage is the discovery of irregular shapes of cluster. For example, in figure 6, the distinct shapes of clusters “t”, “u” and “x” are easily distinguished because their closest nodes are well separated. However for “x” and “o”, the boundary points are very close and a classic approach merges them. The ROCK algorithm looks for points sharing common neighbors which is not the case for these points as shown on figure 7. However in this case, the points A and B should be linked because they a common neighbor. That is why a score measure is introduced to join two clusters with the maximum number of neighbors. Here, the algorithm prefers to join C and D rather than A and B. Hence, the ROCK algorithm is capable to discover right clusters. Other such methods exist like CURE for example where each cluster is fixed by a limited number of points, so it is a tradeoff between one center and all points. Density based clustering techniques such as DBScan are close to ROCK which is well suited for categorical data like trees. The interested reader can read [11] for a good overview of these algorithms and their use cases.

However, ROCK is heavy computational [13] and a derived version, QROCK, was proposed in [13]. The authors of QROCK observed that in many cases, the computed clusters are equivalent to compute the connected components of the created graph. Hence, the algorithm becomes very simple and is executed very fast. The main disadvantage is that the points A and B in figure 7 will be

joined due to their unique neighbor in common. In fact, QROCK does not take in account the neighborhood density measured by the score measure.

5.2.2 Compromise

The limitations of ROCK and QROCK imply logically to choose a fair trade-off between them with following ambitions :

- keep the advantage of the neighborhood density (ROCK),
- avoid too much computational metric (QROCK).

The first idea is to choose a simple score measure. The most simple should be to sum all links between each pair of clusters but the authors of ROCK advice against it. In fact, it often entails the creation of a single or few big clusters because the bigger a cluster is, the more neighbors it has. In this paper, we present a new simple metric for evaluating the score measure between two clusters: the maximal number of shared neighbors between any pairs of two nodes from each cluster. Assuming, two clusters C_i and C_j, the score measure between the clusters is:

$$good(C_i, C_j) = max_{p_t \in C_i, p_l \in C_j} \#neighbors(p_t, p_l)$$

where $\#neighbors(p_t, p_l)$ returns the total number of shared neighbors between p_t and p_l. This metric is very simple to compute because the distance between two points does not vary whereas the original goodness of ROCK is updated during the clusters merging since the metric is based on all shared neighbors between all points of two clusters. Moreover, estimating the total number of possible neighbors for normalizing this value against the size of the cluster is unnecessary with the new metric.

The clusters are merged until this new score reaches a threshold γ. Thus, the clustering has to join two points p_1, p_2 for which $good(p_1, p_2) > \gamma$.

Theorem 1 *The results of the ROCK algorithm based the score measure good is independent from the order of merging points.*

The proof is direct as the definition of *good* is only dependant on the points themselves and not on the clusters, *i.e.*, other points. This theorem is very important as there is no need to order points following the decreasing value of the goodness measure like in ROCK. Thus, the overall complexity is very degraded. Besides, it corresponds to the QROCK algorithm with one additional constraint. In fact, the graph links are weighted by the number of shared neighbors and the objective is to determine the connected components of vertices with weighted links equal to at least γ to keep the neighborhood density as a valuable information. Hence, the algorithm design is straightforward and split into two main functions :

- the graph construction based on the neighborhood computation;
- the computation of connected components.

The first step is executed by algorithm 2 where L_{ij} (the adjacency matrix) is the number of shared neighbors between i and j. In fact this algorithm iterates over all pairs of points (trees in our case). When two of them are neighbors, the algorithm considers one as the shared neighbors and looks for its other neighbors to update the weighted adjacency matrix (loop of the line 8).

Algorithm 2 Link initialization

```
1: T = {t_1, ..., t_N} a set of tree
2: D_ij is the distance between the tree t_i and t_j
3: τ the maximal distance between two neighbor trees
4: L_ij the number of neighbors between the tree t_i and t_j initialized to 0
5: for i ← 1 to N do
6:   for j ← 1 to N do
7:     if D_ij < τ then
8:       for k ← 1 to N do
9:         if D_ik < τ then
10:          L_ij = L_ij + 1
11:        end if
12:      end for
13:    end if
14:  end for
15: end for
```

Then, algorithm 3 computes the connected components having links with at least of weight of *gamma* neighbors. At the beginning, each tree is associated to a label equals `FALSE` indicating that the tree is not in a cluster yet. The algorithm iterates over all tree searching non visited ones and creates a new cluster. Then, the clustering recursive function is applied on the trees shared the minimum number of neighbors with the initial tree in order to add them and so on.

The two metrics chosen for testing this new algorithm are d_1 and d_2. The latter one is directly applied but we do a simple transformation on d_1 for having a normalized value between 0 and 1:

$$d'_1 = 1 - e^{-0.01 d_1}$$

6 Experimentations

6.1 Metrics

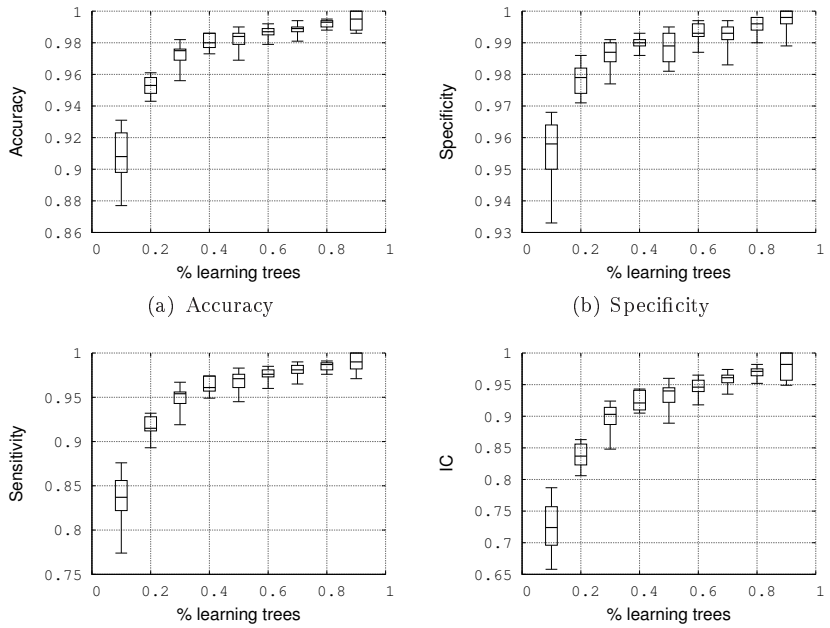
Standard metrics for classification assessment presented in [14] are adapted to our terminology introduced in section 2. Assuming x_d, the number of trees corresponding to a particular device type $d \in D$, y_d the number of trees classified as d, $z_{d_2 d_1}$ the number of trees of types d_2 which were classified as d_1, the sensitivity evaluates the number of trees of a given type d which were assigned to the right cluster:

$$sens(d) = z_{dd}/x_d \tag{13}$$

Algorithm 3 clustering

```
1: T = {t_1, ..., t_N} a set of tree
2: L_ij the number of neighbors between the tree t_i and t_j
3: init(t) creates a cluster with only the tree t
4: c.add(t) add the tree t to cluster c
5: Label_i indicates if t_i is already assigned to cluster and is initialized to 0
6: for i ← 1 to N do
7:    if not Label_i then
8:       c = init(t_i)
9:       Label_i = TRUE
10:      for j ← 1 to N do
11:         if i ≠ j and L_ij > γ and Label_j = FALSE then
12:            clustering(j,c)
13:         end if
14:      end for
15:   end if
16: end for

17: clustering(k,cluster):
18: Label_k = TRUE
19: c.add(T_k)
20: for j ← 1 to N do
21:   if k ≠ j and L_kj > γ and Label_j = FALSE then
22:      clustering(j,c)
23:   end if
24: end for
```



(a) Accuracy (b) Specificity

(c) Sensitivity (d) Mutual information coefficient

Figure 8: Supervised fingerprinting, distance d_1

The specificity of a device type d measures the proportion of trees of this type in the corresponding cluster:

$$spec(d) = z_{dd}/y_d \tag{14}$$

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

$$acc = \sum_{d \in D} z_{dd}/M \tag{15}$$

Furthermore, the average sensitivity and specificity value is easier and faster understandable than multiple values:

$$\begin{aligned} avg\_sens &= \sum_{d_i \in D} sens(d_i)/N \\ avg\_spec &= \sum_{d_i \in D} spec(d_i)/N \end{aligned} \tag{16}$$

Assuming the following distributions $\mathbf{X} = x_i/N$, $\mathbf{Y} = y_i/N$, $\mathbf{Z} = z_{ij}/N$, the mutual information coefficient (IC) is an entropy based measure defined as :

$$IC = \frac{H(\mathbf{X}) + H(\mathbf{Y}) - H(\mathbf{Z})}{H(\mathbf{X})} \tag{17}$$

where H is the entropy function. This ratio varies between 0 and 1 (perfect classification) and is a good complementary metric from the overall accuracy

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 1: Testbed dataset – Tree statistics

because it indicates if the accuracy value is not only due to one or few over-represented classes. For example, assigning all messages to one class can allow to reach 80% of accuracy if 80% of data points are of the same type. However, this case implies $IC = 0$. Hence, this coefficient reflects the sensitivity and the specificity and is more severe than them.

Although the supervised classification creates one labeled cluster per device type which are filled with testing trees, the unsupervised classification can create an arbitrary number of clusters. Even if labeling unsupervised cluster is not done in reality, the classification assessment process begins by labelling each cluster with the most represented device version in the cluster. Then, only the largest cluster of each type is kept and the rest of the trees are assigned to an artificial garbage cluster. However, evaluation the mutual information coefficient with a garbage cluster is meaningless. So, the F-score is another overall possible metric:

$$F - score = \frac{2 \times avg\_sens \times avg\_spec}{avg\_spec + avg\_sens} \tag{18}$$

Like the mutual information coefficient, F-score is a combined measure from sensitivity and specificity but does not reflect the messages distribution. However, if all messages are affected to few classes, this score will be very low too.

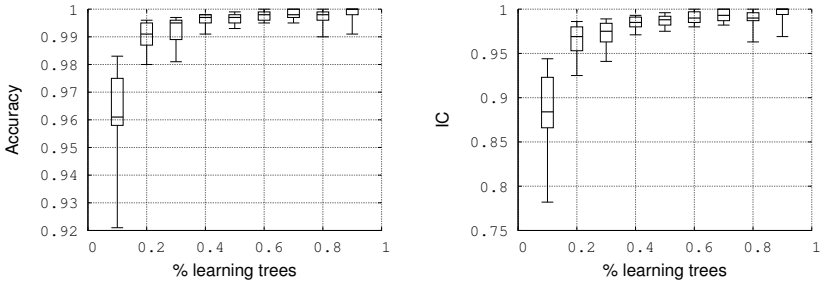
6.2 Dataset

The main dataset having characteristics summarized in table 1 was generated from our testbed with 6 device types (softphones,hardphones and proxy) with a total number of 2091 messages. The syntactic trees are very big, their heights are close to 25-30 and the minimal number of nodes in a tree is more than 800. Therefor illustrating a real example in the paper is impossible.

6.3 Supervised classification

6.3.1 Learning percentage

The first experiment evaluates the efficiency of our supervising method in parallel with the proportion of extracted trees for the learning process (learning percentage). In fact, the messages are randomly selected and each experiment is run ten times to improve the fairness of our results. Considering the distance d_1, the figure 8 plots the accuracy metrics using a quartile representation. The extrema values are plotted and the box delimits 50% the observations with the median value as the horizontal bar. The rest of the observations are outside the box (25% below, 25% above). The overall accuracy shown in figure 8(a) highlights that our approach is very effective because with only 10% of learning, the accuracy is concentrated around 90%. Obviously, increasing the number of



(a) Accuracy (b) Mutual information coefficient

Figure 9: Supervised fingerprinting, distance d_3

learning sample messages improves the accuracy and with 20% the limited range of quartiles indicates stable results. So, expecting an accuracy of 95% is really viable. The average sensitivity illustrated in 8(b) means that the good accuracy is not only due to some well classified classes since this value reaches 90% with a learning percentage of 20% or more The specicity is high also meaning that the misclassified messages are scattered among the different device types and not only one. The figure 8(d) summarizes the previous observations by amplifying the lowest specificity and sensitivity value. Hence, this figure confirms that the information coefficient is clearly more severe than other metrics. That is why the next experiments do not detail sensitivity and specificity values.

6.3.2 Distances

The distance d_3 is normalized contrary to d_1. Such a distance is generally easier to use since many techniques like SVM without considerable tunable transformation. The efficiency of the identification is clearly improved and outperforms the previous ones. With only 20% of messages used for the learning, the accuracy is close to 99% in most cases. The specificity and sensitivity are also very high and illustrated here through the mutual information coefficient in figure 9(b). In fact, obtaining similar results with the previous distance needs to raise the learning percentage. For instance, a percentage of 80% with d_1 is equivalent to 20% with d_2.

To summarize, SVM-based supervised fingerprinting is very effective with the normalized distance d_3.

6.4 Unsupervised classification

Applying unsupervised classification on individual messages was not successful (about 60%). This is mainly due to notable differences between messages targeting different goals (with different types) even emitted from the same kind of device. Hence a message for making a phone call or registering are highly dissimilar. With supervised classification, the learning process capturing different kinds of messages for the same device type counters this problem.

Thus, we propose two methods for improving the classification accuracy :

- identifying the kind of a device based on only one type of messages (*e.g.* INVITE,

- creating small clusters of messages sharing the same IP address and port within a small interval of time ρ (few seconds). This assumption is realistic as these characteristics will not change frequently for a piece of equipment.

6.4.1 Grouping messages

In the first experiment, ρ is set to 5 seconds and the goal is to determine what are the best parameters of the new version of ROCK algorithm :

- τ : the maximal distance between two neighbors,
- γ the minimum number of shared neighbors between to messages for merging them.

With $\rho = 5$, an average of 2.8 messages are grouped in the same cluster beforehand. Except in four cases highlighted by boxed values on table 2, all the different kind of devices are discovered. The shading key helps to rapidly discard bad configurations like the light column ($\tau = 0.01, 0.15, 0.2$) highlighting the great impact of τ. Thus, the accuracy is not a monotonic function of τ. In the same way, it is not a monotonic function of γ and 87% of messages are correctly classified by using a neighbored distance of 0.1 and a minimal of ten shared neighbors for grouping two messages. Moreover, it is ten points better than the best result of the first row which is equivalent to the QROCK algorithm (one shared neighbor only). The high value of F-score indicates that this result is not only due to few device types rightly identified. However, the best configuration seems to fix $\gamma = 15$ and $\tau = 0.1$ with a slightly lower accuracy and a higher F-score. We will consider this configuration for the remaining experiments except when mentionned. The table 3 and 4 give the number of clusters and their sizes from this configuration to another by varying these two parameters. When τ increases, more trees are merged and so less larger clusters are built contrary to γ forcing trees to have more common neighbors for being linked when it increases. Some very small clusters are constructed with only one tree (outlier) since the minimum size is still zero. Furthermore, the original QROCK algorithm corresponding to $\gamma = 1$ is clearly unable to discover so many clusters as for $\gamma = 15$.

The figure 10 shows the evolution of the accuracy depending on the parameter ρ grouping the message arrived in the same interval of time. First, increasing ρ greater than five has no positive impact. Assuming same device type for messages from the same IP address and port within 5 seconds seems reasonable. Second, the normalized distance (d_2 for unsupervised fingerprinting) is better than the nonnormalized one.

6.4.2 Message type

Only the most represented message types are considered : `100`, `200`, `401`, `OPTIONS`, `REGISTER`, `NOTIFY`, `INVITE` and `ACK`. The figure 11 plots the overall accuracy and the F-Score of the classification results depending the type considered. Once again, best results are obtained with the normalized distance. Moreover, this graph points out that some types contains more valuable information than others. For instance `OPTIONS` message includes equipment capabilities which is highly dependant on the device type contrary to the response 200 which

γ (#neighbors)	τ (min distance) 0.01	0.05	0.1	0.15	0.2
1	0.559 (0.674)	0.767 (0.805)	0.721 (0.697)	0.307 (0.339)	0.302 (0.614)
5	0.480 (595)	0.748 (0.787)	801 (0.781)	0.306 (0.336)	0.306 (0.399)
10	0.454 (0.570)	0.742 (0.784)	0.872 (0.879)	0.307 (0.293)	0.307 (0.293)
15	0.424 (0.542)	0.727 (0.767)	0.862 (0.902)	0.525 (0.489)	0.307 (0.293)
20	0.370 (0.497)	0.698 (0.744)	0.804 (0.852)	0.524 (0.488)	0.307 (0.293)

$< 40\%$	40-60%	60-70%	70-80%	80-85%	$\geq 85\%$

Table 2: Unsupervised fingerprinting by grouping similar arrival time messages, distance d_2 - Accuracy (F-Score is put in brackets)

τ	0.01	0.05	0.1	0.15	0.2
#clusters	314	108	61	33	14
Min size	1	1	1	1	1
Max size	126	218	222	480	720
Avg size	2.33	6.79	12.02	22.212	52.357

Table 3: Cluster characteristics with $\gamma = 15$

γ	1	5	10	15	20
#clusters	18	38	47	61	82
Min size	1	1	1	1	1
Max size	353	224	223	222	220
Avg size	40.72	19.29	15.60	12.02	8.94

Table 4: Cluster characteristics with $\tau = 0.1$

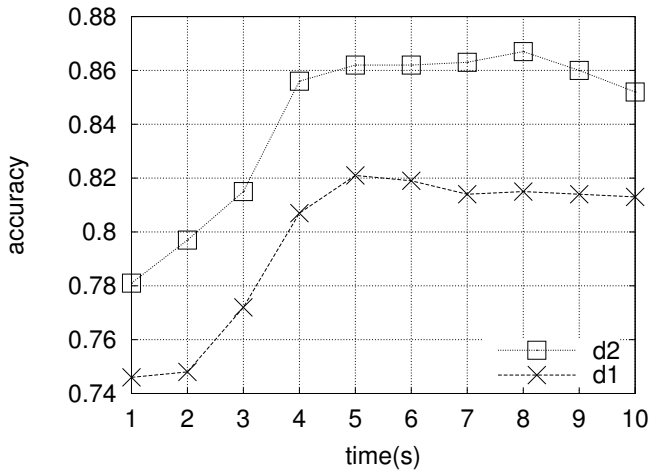


Figure 10: Unsupervised fingerprinting by grouping similar arrival time messages

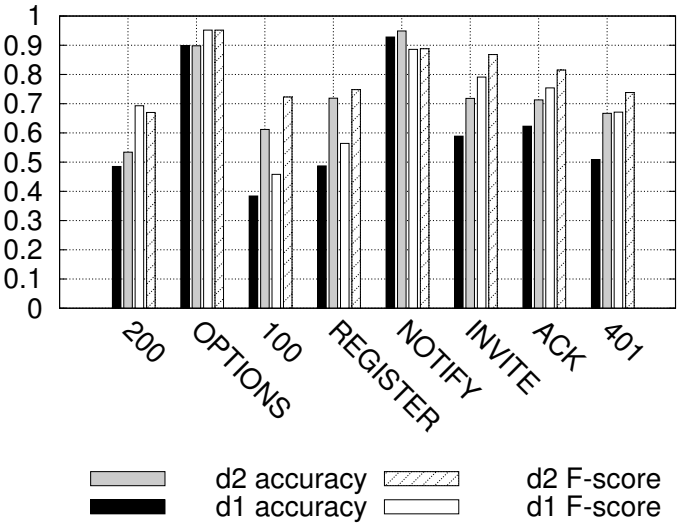


Figure 11: Rock clustering by messages type

is only a kind of acknowledgment. Monitoring the right messages like OPTIONS or NOTIFY is very efficient because 90% or more of messages are well classified with a similar F-score.

6.5 Real VoIP operator network dataset

This section addresses brief results obtained from the network traces of a real VoIP operator. This network contains 40 types of device but the identification

based on the SIP User-Agent return sometimes an unknown type. Some kind of devices are too much under-represented (less than 20 messages) while some others generate 10.000 of the total of 96.000 messages from about 700 distinct devices. Hence, we discard four device types and keep at most 100 messages for each of them. The results are lower than for the testbed dataset. Indeed, the supervised fingerprinting technique is able to correctly identify 70% of equipment and the unsupervised technique groups rightly 90% of OPTIONS message again and 75% based when messages within the same time interval are grouped beforehand. The first conclusion is that OPTIONS message is a very valuable one. By investigating the reason of the relatively limited accuracy in the other cases, we found that some kind of devices cannot be well distinguished like `CiscoATA186v3.1.0` and `CiscoATA186v3.1.1` due to small or lack of stack implementation modifications between the version 3.1.0 and 3.1.1. However, not detecting minor variations is not critical because the flaws and the functionalities of such devices should be very similar. Furthermore, the correctness of this dataset could not be checked and the accuracy assessment is only based on the SIP User-Agent field which can be easily faked.

7 Related work

Network and service fingerprinting tools are widely used attackers for designing customized attacks or by network administrator to have a precise view of their network. The first work in this domain [15] highlights that unclear or permissive specification entails implementation variations due to specific choices or misunderstanding of the developers. Two classes of methods exist. The first one is qualified as passive since it only monitors the traffic without interacting with the fingerprinted machines. For instance, [16] is based on rules associating specific values in TCP fields to an operating system (OS). Active techniques send specific request to a machine in order to get discriminative responses. This scheme is implemented by NMAP [17] for determining the OS. The accuracy of these techniques relies on the good definition of messages to send, which is basically done manually. Therefore, [18] describes a mechanism to learn them. Fingerprinting is not limited to OS identification and several works target to correctly distinguish the different kind of network traffic with different granularity level. For instance, [19] gives a good overview of techniques used for determining the different classes of traffic (Web, P2P, Chat..) whereas [20] and [21] try to automatically identify a specific protocol. Our work is different and complementary since its goal is to determine precisely which implementation of a protocol is used. This kind of methods was explored in [22] for determining the HTTP [23] web server 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 [24]. As argued in the introduction, changing these fields is very easy in order to counter fingerprinting. Our technique doesn't consider the value itself or the flat structure of the message but all its hierarchical syntactic structure related to the protocol grammar which contains more valuable information and which is more difficult to fake while keeping a valid message with the same meaning. SIP fingerprinting is also addressed in [25] with other fields protocol and an active probing technique contrary to those presented in this paper which does not need any interaction with equipment. Our previous

work [26] relies only on multiple sessions of messages without syntactic knowledge and is well designed for protocols with partial or without specification and grammar. We also introduced the use of syntactic information in [27] to create one generic global tree per device type. Even if the classification time are equivalent, the learning process is very long and needs a grid of 10 computers during two days (with 2600 messages) contrary to the current approach where the learning process is very fast (few minutes). Thus, updating the system frequently is possible which is primordial with dynamic technology like VoIP with many new devices appearing rapidly. Furthermore, our previous work did not deal with unsupervised fingerprinting.

8 Conclusion

This paper proposes novel supervised and unsupervised device fingerprinting techniques which leverage the advantages of the SVM paradigm and the ROCK classification. A new version of ROCK was introduced taking advantages of different pre-existing versions. The provided results show the viability of such fingerprinting schemes when used with syntactic trees which reflect both the content of messages and their hierarchical structures. Our future work will focus the fingerprinting of other protocols like wireless protocols because their nature implies security problems as rogue machines intruding the network. Other directions include the automatic monitoring of stack protocol implementation evolution of a device series.

References

[1] SNMP and Research, "The mid-level manager. http://www.snmp.com/products/mlm.html (accessed on 07/30/07)."

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

[3] H. Abdelnur, T. Avanesov, M. Rusinowitch, and R. State, "Abusing SIP Authentication," in *Information Assurance and Security (ISIAS) Information Assurance and Security, 2008. ISIAS '08.* Naples Italie: IEEE, 2008, pp. 237–242. [Online]. Available: dx.doi.org/10.1109/{IAS}.2008.29http://hal.inria.fr/inria-00326077/en/

[4] 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.

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

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

[7] L. Khan, M. Awad, and B. Thuraisingham, "A new intrusion detection system using support vector machines and hierarchical clustering," *The VLDB Journal*, vol. 16, no. 4, pp. 507–521, 2007.

[8] M. Nassar, R. State, and O. Festor, "Monitoring SIP traffic using Support Vector Machines," in *11th International Symposium on Recent advances in intrusion detection - RAID 2008 Recent Advances in Intrusion Detection Lecture Notes in Computer Science*, vol. 5230. Springer, 2008, pp. 311–330.

[9] R. Debnath, N. Takahide, and H. Takahashi, "A decision based one-against-one method for multi-class support vector machine," *Pattern Anal. Appl.*, vol. 7, no. 2, pp. 164–175, 2004.

[10] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *Neural Networks, IEEE Transactions on*, vol. 13, no. 2, pp. 415–425, Mar 2002.

[11] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping Multidimensional Data*, 2006, pp. 25–71.

[12] "Rock: A robust clustering algorithm for categorical attributes," in *ICDE '99: Proceedings of the 15th International Conference on Data Engineering*. Washington, DC, USA: IEEE Computer Society, 1999, p. 512.

[13] M. Dutta, A. K. Mahanta, and A. K. Pujari, "Qrock: a quick version of the rock algorithm for clustering of categorical data," *Pattern Recogn. Lett.*, vol. 26, no. 15, pp. 2364–2373, 2005.

[14] 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.

[15] 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

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

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

[18] J. Caballero, S. Venkataraman, P. Poosankam, M. G. Kang, D. Song, and A. Blum, "FiG: Automatic Fingerprint Generation," in *The 14th Annual Network & Distributed System Security Conference (NDSS 2007)*, February 2007.

[19] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: myths, caveats, and the best practices," in *CONEXT '08: Proceedings of the 2008 ACM CoNEXT Conference*. New York, NY, USA: ACM, 2008, pp. 1–12.

[20] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. M. Voelker, "Unexpected means of protocol inference." in *Internet Measurement Conference*, J. M. Almeida, V. A. F. Almeida, and P. Barford, Eds. ACM, 2006, pp. 313–326.

[21] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, "ACAS: automated construction of application signatures." in *MineNet*, S. Sen, C. Ji, D. Saha, and J. McCloskey, Eds. ACM, 2005, pp. 197–202.

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

[23] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616 (Draft Standard), June 1999, updated by RFC 2817. [Online]. Available: http://www.ietf.org/rfc/rfc2616.txt

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

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

[26] J. François, H. Abdelnur, R. State, and O. Festor, "Behavioral and Temporal Fingerprinting," INRIA, Research Report RR-6995, 2009. [Online]. Available: http://hal.inria.fr/inria-00406482/en/

[27] H. Abdelnur, R. State, and O. Festor, "Advanced Network Fingerprinting," in *Recent Advances in Intrusion Detection Lecture Notes in Computer Science Computer Science*, ser. Computer Science, Ari Trachtenberg, Ed., vol. Volume 5230/2008, MIT. Boston United States: Springer, 2008, pp. 372–389.

Centre de recherche INRIA Nancy – Grand Est
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399