ISSN 1870-4069

Obfuscated Information Classification

González Rodríguez Florencio Javier, Aguirre Anaya Eleazar, Salinas Rosales Moisés, Barrón Fernández Ricardo

Instituto Politécnico Nacional, Centro de investigación en Computación, Mexico City, Mexico
fjgonzalezr92@gmail.com, {eaguirre, rbarron }@cic.ipn.mx
msalinasr@ipn.mx

Abstract. When a security assessment is executed, time and resources are limited therefore it is essential to identify those points that are most likely vulnerable and focus on those. Such points are identified in the information gathering stage turning it in a vital step during the assessment. Emerging a question about if gathered information could be manipulated by a third party either by functionality or by obscuring information goals. This paper proposes an information classifier in order to identify obfuscated information, classifying into obfuscated or integral information, with the purpose to be used during a "black box pentesting" assessment.

Keywords: black box pentesting, information gathering, security assessment, network fingerprinting.

1 Introduction

Information gathering is the first and most important phase in Penetration Testing. The goal is to obtain all possible target information with the purpose to get a security target profile like network architecture, features of devices involved, or personal user's information that could be useful in the vulnerability testing. This kind of information should not be accessible for not authorized people, but in many cases it cannot be limited at all. For example, certain information in devices like open ports, protocol implementations, replay information from certain ports and so on. Penetration testers will take advantage of this to find vulnerabilities before attackers do.

Organizations used to mitigate such problem implementing some security controls like firewalls, letting block some patterns and limit access. Even with possible incongruence in configuration as is mentioned in [1]. On the other hand, for those information that cannot be limited at all, there are devices able to modify default protocol implementation fields, like protocol scrubbers that try to avoid protocol fingerprinting with the intention to mislead attackers.

This paper proposes a classifier for information collected from "information gathering" phase. Taking as input implementation features from the traffic generated during the information gathering execution in a black box pentesting, extracting default implementation values that can be changed either by the vendor or a security professional, in order to identify incongruities between network protocol implementations. Incongruities identification refers to compare basically results of three OS Fingerprinting techniques. First one analyzing TCP and IP protocol in order

to fingerprint the Operating System. Second one using ICMP protocol characteristics and third one analyzing some services banners results to identify modifications in the strings and also to fingerprint OS. Classification for each technique is executed using Machine Learning algorithms described in Section 5 “Analysis and Design”.

2 Security Controls for OS Fingerprinting Prevention

Actually does not exist any tool that offers a target profile considering obfuscated information during a security assessment. This research proposes matching results through OS Fingerprinting using three techniques. Such techniques are related to some previous researches but having a different perspective. For example, can be found proposals like in [2] where Jason Barnes and Crowley make OS fingerprinting using a passive traffic fingerprinting mechanism, to identify hosts features involved in communications without interfering in any way. Using essentially SYN TCP/IP flags, obtaining features to identify HTTP clients, physical link types, and even if a host is behind a NAT device on a large net- work, receiving traffic from a Passive Network Appliance “PNA” in [3]. Being not necessary to make any system call, they have made evaluations in two lab- oratories: with constructed traffic and in an operational setting with real world traffic.

They compare their proposal with p0f and k-p0f tools, measuring the average maximum sustainable throughput across 30 second intervals 10 times for each mixture of traffic and type of monitor, resulting k-p0f better than p0f.

Same idea is analyzed in this proposal to fingerprint Operating Systems in order to compare results between different techniques and evaluate possibility of obfuscated information if any incongruities are identified.

In [10] Prigent, Vichot and Harrouet present IpMorph in order to show that fingerprint concealment and spoofing are uniformly possible against different known fingerprinting tools, IpMorph is a counter-recognition software implemented as a user-mode TCP/IP stack, ensuring session monitoring and on the fly packets re-writing used against fingerprinting tools like Nmap, Xprobe2, Ring2, SinFP and p0f. IpMorph cover more characteristics and analyses deeply OS Finger- printers, even mention those able to get services banners in order to identify an Operating System, but they did not cover such aspect.

Our proposal covers the identification of those devices that protect an OS through the manipulation of banners structure. Therefore, is proposed a technique to identify if a Service Banner could have been manipulated.

Protocol scrubbers modify default fields of multiple protocols in order to reduce the number of techniques than can be used to identify an Operating System. Then they were analyzed to take in consideration protocols fields that usually protocol scrubbers modify. Analyzing congruency between protocols implementation and services banners.

3 Methodology

Experimental methodology was used in this research, since there exists a correlation between variables described in Tables 2 and 4 in Section 4, that were needed to analyze them through certain active and passive experiments, having results that are compared with expected ones, and take most descriptive values. Such methodology was divided in two phases:

Exploratory: This phase identifies questions that were tried to answer in testing phase, in this research such questions are mainly 3:

1. Can be identified obfuscated information?
2. Can be identified a device that obfuscates information?
3. Which features make identifiable obfuscated information?

Testing: In the testing phase planted questions were tried to be solved through experiments where analyzed values that could have been manipulated.

Question 1 was answered through the analysis and comparative between default values for each device analyzed (NAT, Protocol Scrubber, Hardened host).

Questions 2 were answered through the analysis of data type and behavior from values obtained during the analysis for each device.

Questions 3 were answered through the analysis of data type from values obtained during the analysis for each device, then testing Machine Learning algorithms and feature selection step.

4 Analysis and Design

4.1 Algorithms

Classification is a way to solve a categorization problem using Machine Learning, there exist different algorithms used for solving certain problems shown in Fig. 1.

Machine Learning was used in this research to classify an Operating System given some features. Mainly are considered three elements for the classifier input, TCP + IP characteristics implementation, ICMP characteristics implementation and Service Banners analysis. First step is to identify the Operating System through OS Fingerprinting, analyzing TCP and IP headers of packets, due TCP/IP is the most common implemented protocol and most usable, be- sides TCP and IP have many field values that are not specified formally in RFC, it is known that certain security policies involve to block ports that are not used by a host, but it is possible even with blocked port analyze their behavior.

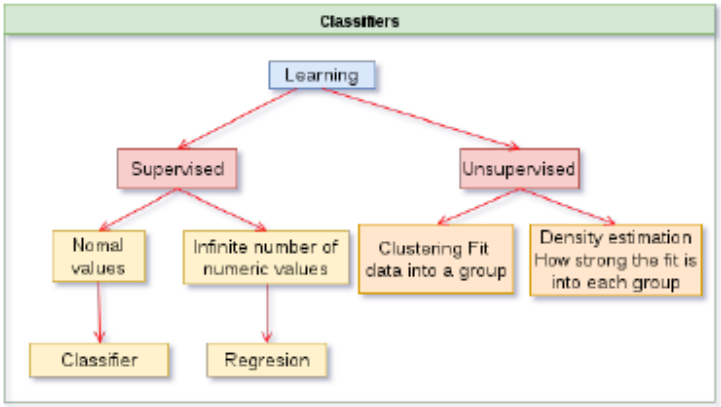


Fig. 1. Top Machine learning algorithms.

Mentioned inputs where selected analyzing related works about Protocol Scrubbers and NAT network analysis presented in Section 2. Characteristics for each input were obtained through an experimental process. First analyzing documentation about network Kernel parameters, Open Source Protocol scrubbers documentation and features analyzed in research papers. Then analyzing real traffic samples on different environments, in order to be compared versus documentation. Finally, such features are presented in subsection 4.3 for each implemented algorithm.

Mainly 3 open source protocol scrubbers were analyzed due others published protocol scrubbers are not open source, and it is not possible to analyze at all. Protocol Scrubbers analyzed were:

- IP Personality in [5]
- Scrub tech in [6]
- IP log in [7]

For the analysis in this research each Protocol Scrubber was installed in a virtual and physical machine. Studying configuration for each of them in order to get a list of values they use to change. As result were found 14 values that are used to identify Protocol Scrubbers. Being mainly TCP, IP, UDP and ICMP protocols those that are modified by Protocol Scrubbers, in order to affect indirectly things like RTT, timers and so on. Even packet length is used to hide an operating system implementation, due different payloads data used by each developer, in order to mislead a fingerprinting process executed by a tester. These variables are shown in Table 1.

It is possible to tag these features as an Operating System name, in order to identify an Operating System given some features.

Table 1. Values to identify Protocol Scrubber.

TCP	TS option	Urgent pointer	Winsize	Sack	Ack retries
	Nop option	Winscale	Max window	Options order	
IP	TTL	ID	ToS	Flags	
ICMP	Length	Payload			

In this research feature selection was made through a specialist person knowledge, letting to tag features for some systems. The type of features analyzed in each protocol are nominal values, therefore four classifier algorithms were selected to identify the OS, finally AdaBoost is implemented to choose best decision between classifiers.

Classifiers implemented for OS Fingerprinting using TCP+IP, ICMP and Services banners are:

- KNN
- Bayes
- Decision Tree

Classifiers were selected based on accuracy, training time, linearity, number of parameters and number of features. Taking in consideration data used in this proposal.

Each classifier was implemented in python using each algorithm model. Adapting parameters described in "Implemented algorithms" section, in order to get good results for classification. However, each of them was compared with Scikit framework, obtaining better results for some classifiers using the developed classifier and sometimes using Scikit. Best results are present through this paper.

4.2 Features

Features that were used by each algorithm are presented in section 4.3, however data used for training algorithms have a common structure shown in Table 2a, where x_n is a feature from the protocol and OS is the Operating System that is a label for features row. x_n is a field implementation protocol, or a value that can be obtained from interactions with the target. These values characterize or represent to the target, and can be used to make OS Fingerprinting.

Table 2. Features structure.

(a) Data structure used for classifiers training

feature x_1	feature x_2	...	feature x_n	OS1
feature x_1	feature x_2	...	feature x_n	OS2
...	...	...	...	...
feature x_1	feature x_2	...	feature x_n	OS3

(b) TCP+IP Data values used for classifiers training

TTL	ToS	IP Flags	WinSize	Sack	NopOption	linux
TTL	ToS	IP Flags	WinSize	Sack	NopOption	windows
...	...	...	...	...	...	...
TTL	ToS	IP Flags	WinSize	Sack	NopOption	linux

(c) ICMP Data values used for classifiers training

Length [Payload]	linux
Length [Payload]	windows
...	...	...
Length [Payload]	linux

(d) ICMP Data values used for classifiers training

linux	service	version	operating system
windows	service	version	operating system
...	...	...	...
linux	service code	service	version

The dataset used to classify using TCP+IP fields is shown in Table 2b and has the same structure shown in Table 2a. Such values are result of a previous analysis over Fingerprinting tools and those tools that try to avoid fingerprinting like Protocol Scrubbers.

The dataset used to classify using ICMP fields is shown in Table 2c and has the same structure shown in Table 2a.

Finally, the dataset used to classify using Services Banners is shown in Table 2d and has the same structure shown in Table 2a, but starting with operating system due strings longitude is not the same for all banners.

Data used for training and testing are samples shown in Tables 3 from Open source fingerprinting tools, different implementations extraction, and for Banners were used data from servers extracted using Shodan service. Each sample for TCP+IP and ICMP is a list of values for each field described in Table 2b, 2c with an associated label that is the Operating System name.

Table 3: Number of samples for training and testing.

Traffic samples			
Type	# Sample		Total
	Physical	Virtual	
TCP+IP	45	8	53
ICMP	20	10	30
Banners (HTTP, FTP, SSH)	-	500	500

4.3 Implemented Algorithms

Three algorithms are evaluated using metrics in Equations 1, 2, 3 based on a confusion matrix for each algorithm:

$$Recovery(OS) = A_{ii} \overline{\sum_{J=1}^{n} AiJ}, \tag{1}$$

$$Precision(OS) = A_{ii} / \sum_{J=1}^{n} AJi, \tag{2}$$

$$Accuracy = \sum_{i=1}^{n} A_{ii} / \sum_{i=1}^{n} \sum_{J=1}^{n} AiJ, \tag{3}$$

where Recovery is the proportion of cases correctly identified as belonging to class C among all cases that truly belong to class C. Precision also called true positive rate, is the proportion of cases correctly identified as belonging to class C among all cases of which the classifier claims that they belong to class C. Finally, Accuracy is the ratio of correct predictions to total predictions made.

Naive Bayes

Naive Bayes is a probabilistic classifier based on the Bayes theorem with strong naive independence assumptions between the features. Due this classifier assume that the value of a particular feature is independent of the value of any other feature, given a class C. For this proposal it is important because protocol scrubbers modify values from time to time, then if values were dependent, classifier will not work at all as Decision Tree classifier that will be also described. Equations used in this research for Naive Bayes are show in equation 4:

$$P(C|x_1, x_2, \dots, x_n) = \frac{(\prod_{i=1}^{n} P(X_i|C)P(C)}{P(x_1, x_2, \dots, x_n)} = \frac{P(x_1, x_2, \dots, x_n|C\ )P(C)}{P(x_{1,} x_2, \dots, x_n)}, \quad (4)$$

where:

$$P(C) = \frac{Number\ of\ C\ classes}{Total\ number\ of\ classes},$$

$$P(x_n|C) = \frac{Number\ of\ rows\ that\ have\ x, and\ are\ C\ class}{Number\ of\ rows\ that\ are\ C\ class},$$

$$P(x_1, x_2, \dots, x_n|C) = P(x_1|C\ )P(x_2|C) \dots P(x_n\ |C),$$

$$P(x_1, x_2, \dots, x_n) = P(x_1)P(x_2) \dots P(x_n).$$

Table 4 shows the confusion matrix for Naive Bayes, with 20 samples for each Operating System where 10 were taken from training samples and 10 were different from training samples.

Table 4. Confusion Matrix for Naive Bayes.

Windows 9x/NT	OSX 10.x	Linux 2.2.x	Linux 4.x	Cisco 12.0	OpenBSD 2.x	Windows 9x/NT 20
Linux 2.2.x		18	2			
Linux 4.x		1	19			
Cisco 12.0				20		
OpenBSD 2.x	1				19	

Naive Bayes Classifier evaluation

Table 5. Evaluations metrics for Naive Bayes.

Recovery	Precision
OS	
Windows 9x/NT	1	1
OSX 10.x	0.9	0.9473
Linux 2.2.x	0.9	0.9473
Linux 4.x	0.95	0.9047
Cisco 12.0	1	1
OpenBSD 2.x	0.95	0.9047

Accuracy = 114/120 = 0.95

K nearest neighbours

K nearest neighbors is a classifier that stores all available cases and classifies new cases based on a similarity measure. Equations used in this research are show in Equation 5:

$$d = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \dots + (x_n - y_n)^2}, \quad (5)$$

where:

x_i is the feature in the database
y_i is the input feature to classify
d is the distances that means how different are the input and the database item

Table 6 shows the confusion matrix for KNN, with 20 samples for each Operating System where 10 were taken from training samples and 10 were different from training samples.

Table 6. Confusion Matrix for KNN.

Windows 9x/NT OSX 10.x Linux 2.2.x Linux 4.x Cisco 12.0 OpenBSD 2.x Windows 9x/NT 20						
OSX 10.x	17					3
Linux 2.2.x		17	3			
Linux 4.x		2	16			2
Cisco 12.0				20		
OpenBSD 2.x	3					17

KNN Classifier Evaluation Metrics obtained from confusion matrix in Table 7 are calculated as: Accuracy = 107/120 = 0.8916

Table 7. Evaluations metrics for KNN.

Recovery Precision		
OS		
Windows 9x/NT	1	1
OSX 10.x	0.85	0.85
Linux 2.2.x	0.85	0..8947
Linux 4.x	0.8	0.8421
Cisco 12.0	1	1
OpenBSD 2.x	0.85	0.7727

Decision tree

Decision tree is a predictive model where the target variable can take a discrete set of values, leaves represent class labels and branches represent conjunctions of features that lead to those class labels.

Table 8 shows the confusion matrix for Decision Tree, with 20 samples for each Operating System where 10 were taken from training samples and 10 were different from training samples.

Table 8. Confusion Matrix for Decision Tree.

Windows 9x/NT OSX 10.x Linux 2.2.x Linux 4.x Cisco 12.0 OpenBSD 2.x Windows 9x						
OSX 10.x	20					
Linux 2.2.x		17	3			
Linux 4.x			15			5
Cisco 12.0				20		
OpenBSD 2.x					10	10

Decision Tree Classifier Evaluation Metrics obtained from confusion matrix in Table 9 are calculated as: Accuracy = 102/120 = 0.85

It is worth to mention that Decision Tree had bad results when features do not exist in the database, however this property could be useful to identify specific hardened hosts or NATted networks in future work. Due hardened hosts just change their default implementation values just when is hardened and is not making changes repeatedly,

also it is known that exist some hardened hosts distributions. Such systems can be analyzed and added to the database, ensuring that is going to be identified by Decision Tree algorithm.

Table 9. Evaluations metrics for Decision Tree.

Recovery	Precision	
OS		
Windows 9x/NT	1	1
OSX 10.x	1	1
Linux 2.2.x	0.85	1
Linux 4.x	0.75	0.8333
Cisco 12.0	1	1
OpenBSD 2.x	0.5	1

ADA Boost

ADA Boost is an algorithm for constructing a "strong" classifier as a linear combination of others classifier referenced as "weak". General idea is represented by Equation 6:

$$f(x) = \sum_{i=1}^{T} a_t h_t(x), \tag{6}$$

where:

$h_t(x)$ is a "weak" classifier

α is an assigned weight for each instance in the training dataset.

Each weighted prediction pass through a classifier, which is then weighted as "alpha values". Finally, each alpha value is summed up in the circle that processes the final result as Fig. 2 shows.

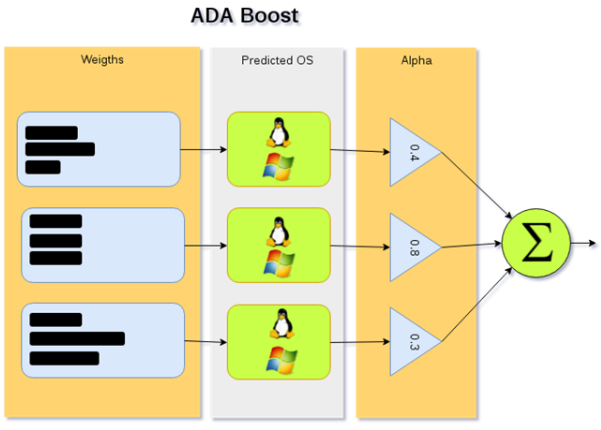


Fig. 2. Banners classification for OS Fingerprinting.

In this research weak classifiers are Naive Bayes, KNN and Decision Tree. Obtaining an Operating system and a weight after classification process. Resulting with a most representative Operating System for the IP address analyzed.

Proposal

Fig. 3 shows the proposal based on results shown in Tables 5, 7 y 9. Where is executed an OS Fingerprinting based on TCP+IP, ICMP and Services Banners analysis. It is divided in three steps, first one is to identify an Operating System using TCP+IP characteristics using three tested classifiers, having as a result three Operating Systems that presents just one IP Address. If some of them are different then they could be manipulating information, that is considered as information obfuscation, decision that is taken by ADA Boost algorithm. Second step is the same idea but using as input ICMP characteristics, also considering last result in order to match and get a congruence value as reference. Finally, Banners are analyzed, considering previous results and having as a result an Operating system name and a value that represents congruence between results.

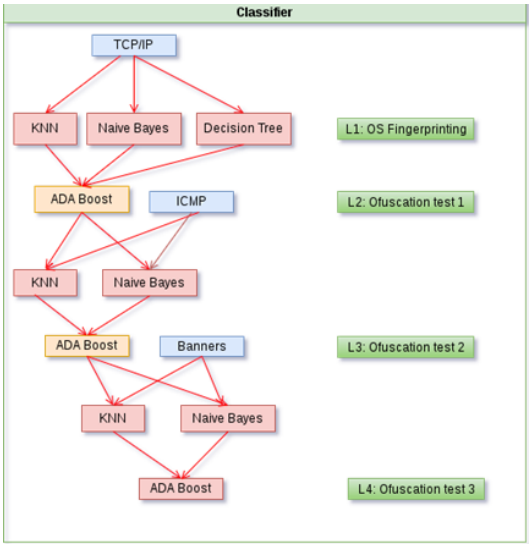


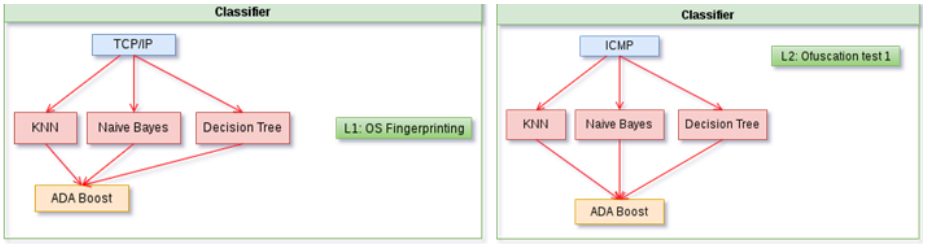
Fig. 3. Classifiers proposal.

Because of the number of features of TCP + IP useful for OS Fingerprinting, these protocols are analyzed to identify with most accuracy the Operating System. Each classifier has a result that should be the same for the three classifiers, because it means that network implementation for the host has not been modified, but if incongruities are identified then it could be obfuscating information.

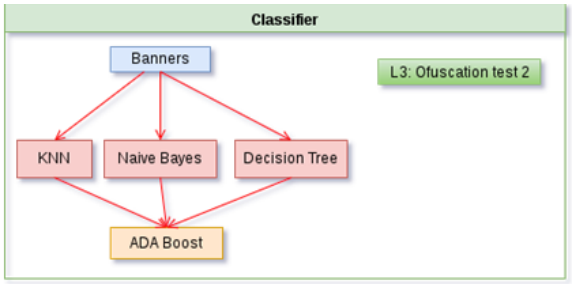
In order to have just one result ADA Boost is implemented to choose the best decision choosing an OS that identifies an IP address. Finally, ADA Boost has just one result as output in order to be analyzed in the next step as Fig. 4a shows.

For the next step is analyzed ICMP, that offers less features to identify an Operating System, but they are enough to detect ambiguities between TCP+IP and ICMP, also joining results for each classifier through ADA Boost as is shown in Fig. 4b.

Finally, Services Banners analysis are the last step due services that were installed after the initial hardening offers clues about the Operating System, obtaining a new feature to compare against TCP/IP and ICMP analysis, having as an output a numeric value that represent if exist ambiguity between the three analysis, letting to classify as obfuscated or integral information. Structure is shown in Fig. 4c



(a) TCP+IP classification for OS Fingerprinting. b) ICMP classification for OS Fingerprinting.



(c) Banners classification for OS Fingerprinting.

Fig. 4. Proposal classification algorithms structures for OS Fingerprinting.

5 Results Analysis

In Table 10 can be seen evaluation metrics for each algorithm obtained from previous reported tests.

For Windows 9x/NT, default values in the dataset are kind of different compared with other OS samples, range of values have big numeric differences versus others, so results are as expected, classifiers are able to identify Windows 9x/NT 100% of tests.

In OSX 10.x percent is not the best for those algorithms that offers how similar is the input versus the training dataset. It is because of as it is known OSX is an Operating System based on BSD Family OS, then it can be seen in Tables 4, 6 that sometimes classifier confuses OSX with BSD. However, Decision Tree did not make any mistake, it is because Decision Tree classifies it just if all characteristics match with a dataset sample.

For Linux Operating system with different Kernels it is the same as OSX analysis. Classifiers confuses Kernel because of Operating Systems Kernels are similar. But if they have unique default values, they are classified correctly by Decision Tree, for example old kernel versions of Linux.

Finally, for other systems with no similar characteristics with dataset samples, for example Cisco 12.0 and Windows 9x/NT, are identified without any problem. But if more similar versions are added, then classifiers will start making mistakes. Even when classifiers made some mistakes, accuracy keeps an acceptable value. Results that are evaluated by ADA Boost algorithm in order to get best results from classifiers.

Table 10. Evaluation metrics for implemented Classifiers.

OS	Bayes		KNN		Decision Tree	
	Recovery	Precision	Recovery	Precision	Recovery	Precision
Windows 9x/NT	1	1	1	1	1	1
OSX 10.x	0.9	0.9473	0.85	0.85	1	1
Linux 2.2.x	0.9	0.9473	0.85	0..8947	0.85	1
Linux 4.x	0.95	0.9047	0.8	0.8421	0.75	0.8333
Cisco 12.0	1	1	1	1	1	1
OpenBSD 2.x	0.95	0.9047	0.85	0.7727	0.5	1
Accuracy	0.95		0.8916		0.85	

6 Future Work

The current proposal represents the base for obfuscated information identification, but it is needed to identify as well those devices that are able to obscure integral information, therefore future work involves identification of devices like NAT Networks, Protocol Scrubbers and Hardened hosts. Having as a result obfuscated information but also those devices able to obfuscate them. Letting to a tester identify these kind of security controls, concentrating on systems emanating real information that can be used to compromise them.

7 Conclusions

Obfuscated information identification hypothesis through protocol implementation analysis is demonstrated using classification algorithms, even with features obtained from passive information gathering.

Each algorithm has advantages and disadvantages as can be seen in result analysis section, then Adaboost complements the work to get best results for each test. Decision Tree and KNN seems to be worst algorithms to classify this kind of problem. However, it is not in this research, because of if a specific obfuscation system was identified and integrated to data base, it has high probability to be identified using this proposal. For example, a specific version of a protocol scrubber, a public hardened host system and so on. For Bayes can be seen that have best results. Due Bayes is probabilistic, letting

modify some equation properties in order algorithm decide based on a binary value, but having a result about how much is similar the input with database samples.

Then based on results it is possible to identify obfuscated information taking advantage of some network protocols implementations. Just analyzing any incongruence in their configurations between those modified protocols. Task that is valuable for penetration testers during a black box security assessment, due it is important to focus over those most likely vulnerable systems in the environment, reducing time and resources. But also having present those possible devices that could be a security control.

Acknowledgment. The authors gratefully acknowledge use of the services and facilities of the Centro de Investigación en Computación (CIC) and Instituto Politécnico Nacional (IPN), also to Consejo Nacional de Ciencia y Tecnología (CONACYT) for supporting this research.

References

1. Antonis, P., Polydoros, P., Miltos, G.: A firewall module resolving rules consistency. TEI of Crete (2017)
2. Jason, B., Patrick, C.: k-p0f: A High-Throughput Kernel Passive OS Fingerprinter. Washington University in St. Louis (2013)
3. Schultz, M., Ben, W., Patrick, C.: A Passive Network Appliance for Real-Time Network Monitoring. Washington University in Saint Louis (2011)
4. Guillaume, P., Florian, V., Fabrice, H.: IpMorph: fingerprinting spoofing unification. Plouzan, France (2009)
5. IP Personality: http://ippersonality.sourceforge.net
6. Scrub tech: http://scrub-tech.sourceforge.net/
7. IPLog: http://ojnk.sourceforge.net/stuff/iplog
8. Massimiliano, A., Ermanno, B., Sushil, J.: A deception based approach for defeating OS and Service Fingerprinting. Naples, Italy (2015)
9. Matthew, S., Robert, M., Farnam, J.: Defeating TCP/IP Stack Fingerprinting. Ann Arbor, Michigan (2000)
10. Steven, M.: A technique for counting NATted Hosts. Marseille, France (2002)
11. Liang, W., Kevin, P., Aditya, A., Thomas, R., Thomas, S.: Seeing through Network Protocol Obfuscation. Denver Colorado, USA (2015)
12. Quinlan, J.: Induction of Decision Trees. Sydney, Australia (2017)