Extracting IDS Rules from Honeypot Data: A Decision Tree Approach

Pedro Henrique Matheus da Costa Ferreira, Leandro Nunes de Castro

Natural Computing Laboratory, Graduate Program in Electrical Engineering
Mackenzie Presbyterian University, Brazil

Email: phmatheus@msn.com, lnunes@makenzie.br

ABSTRACT

This work uses data collected by honeypots to create rules and signatures for intrusion detection systems. The rules are extracted from decision trees constructed based on the data of a real honeypot installed on an internet connection without any filter. The results of the experiments showed that the extraction of rules for an intrusion detection system is possible using data mining techniques, in particular the decision tree algorithm. The technique proposed allows the analyst to summarize the data into a tree, where he/she can identify problems and extract rules to help reducing or even mitigate the security problems pointed out by the honeypot.

KEYWORDS

Honeypot, Intrusion Detection System, Datamining, Decision Tree, Dionaea.

1 INTRODUCTION AND MOTIVATION

Over the past ten years there has been an exponential increase of devices connected to the Internet [1], which promoted the emergence of a new and fertile ground for cyber criminals. They see in the system failures, the lack of technical training for network administrators and lack of vision of the companies that information security is a vital area for the health of business [2] the perfect opportunity to take advantage exploiting these flaws.

One of the main difficulties of a network administrator is to keep the network safe from external attacks. According to [3] the attacks reported by companies in the last two years are divided as follows: 43% are of malicious code injection attacks through SQL, other 19.95% are attacks targeted only at companies or services provided by companies (APT); *Botnet's* represent 18.81% and, finally, the denial of service attacks (DoS) reached 18.24%.

Still according to this study, organizations face an average of 66 weekly cyber attacks that cause some sort of damage to business. Organizations in Germany and the United States experience the highest average weekly attacks, 82 and 79, respectively. Brazil and Hong Kong have the lowest average frequency, totaling 47 and 54 attacks per week, respectively.

This type of scenario brought to light some studies, such as [4], which proposed the first intrusion detection system and the work of [5], which launched the first *honeypot*. The work in [6] proposed the creation of virtual honeypots. These works seek to create tools to assist the protection of computing assets by detecting intruders or creating traps to monitor malicious activities.

This work proposes the application of a data mining technique based on the C4.5 decision tree algorithm to a dataset obtained from attacks targeting a Dionaea honeypot. After the application of the technique it was possible to generate rules for the IDS. The method also reduced the volume of data to be analyzed allowing the network administrator to have an analytical overview of the information captured.

This paper is organized as follows. Section 2 provides a brief review of honeypots and Dionaea. Section 3 presents a case study for the Paris dataset, including database details, pre-

processing, the application of the decision tree and the extraction of classification rules from the tree. The article concludes in Section 5 with general discussions about the proposal and future works.

2 HONEYPOTS AND THE DIONAEA

The first intrusion detection Model was designed by [4] to analyze real-time data in order to detect security breaches, invasions and other forms of abuse of computer access. His model was based on the assumption that security breaches could be identified through the system audit logs, making detection of anomalies in usage patterns. Another design feature was the fact that it is independent of a system vulnerability or type of invasion. This model provided a general-purpose framework for intrusion detection systems.

Based on the assumptions used by Denning [4] to audit logs the first honeypots were proposed. The honeypot is a computer security system dedicated to being probed, attacked or compromised [7]. The first available honeypot was created in 1998 by Cohen in [5] and was designed to simulate a system with vulnerabilities. In the early 2000s the WORMS began to proliferate, requiring the collection of these artifacts for examination and the creation of vaccines for antivirus systems. Having identified this need, in [5] it was proposed to create virtual honeypots in which a single device can run multiple honeypots. This work has promoted the creation of the Honeyd project [7], which emulates in a single physical machine several different operating systems and multiple hosts on a network. In an attack the Honeyd tries to, passively, identify the remote host, collecting network traffic and TCP/IP stack information. This system has the capability to emulate all the TCP/IP stack enabling sophisticated network analysis tools such as nmap, to be deceived.

After the Provos [6] proposal, it began to emerge several honeypots to emulate complex operating systems, its network services and specific services independent of an operating system. With this new wave it became necessary to classify the types of honeypot and, therefore, it was proposed to classify them into three categories: low, medium and high interaction.

2.1 Low Interaction Honeypots

Characterized by emulated computer systems through computer programs that contain the minimum operation standards of the service to be monitored [7]. This type of Honeypot records the attack and their respective shellcode, offering little information about the attack to determine the cause or the mechanisms used in the attack. The information collected allows the administrator to identify whether your network is being targeted by attacks and scans.

2.2 Medium Interaction Honeypots

These types of honeypots are in between the high and the low interactivity ones. Its main feature is to provide the virtualization of the application layer where the operating system environment and the communication protocols are emulated in order to provide sufficient answers to deceive the attacker and get the PAYLOAD [8].

One of the challenges of this system is its complexity of development and the remote possibility of the attacker to gain access to the host system, affecting all the equipment and the network in which the system is. This paper will address only the medium interaction honeypots, specifically Dionaea.

2.3 High Interaction Honeypots

Characterized by real systems with known and purposely not corrected failures. These are expected to be attacked and compromised [9]. In the high-interaction honeypot it is possible for the attacker to compromise and gain control of the system to install software artifacts and complete the malicious activity.

ISBN: 978-1-941968-03-1 ©2014 SDIWC

2.4 Dionaea

The Dionaea [10] was the honeypot chosen for this work due to its storage and data organization characteristics and its capability of capturing malicious artifacts. The collected data can be used to compare the techniques used in this work with works from the literature and the malicious artifacts captured will be used, along with data collected, in the feature extraction process.

The Dionaea is a honeypot of medium interactivity aimed at replacing its precursor, the Nepentes [7]. The great contributions that Dionaea brought to Nepentes were the separation of the core of the system developed in C ++, the inclusion of support for Python [11] as scripting language, the use of the libemu library to shellcodes detection and the native support for IPv6 [12] and TLS [13].

The Python programming language is used to develop the vulnerabilities and supported modules, together with the storage and transmission functions of the information collected. This inclusion brought some indirect benefits to the honeypot, as the possibility of including other types of services not initially planned, for instance, the vulnerabilities in Microsoft SQL Server database [14] and Session Initiation Protocol [15], which is used for controlling multimedia communications sessions, among others.

Dionaea was one of the first honeypots to add support for IPv6 protocol, allowing the analysis of the vulnerabilities that are being exploited in this new communication protocol that will replace IPv4.

3 CASE STUDY: THE PARIS DATASET

Companies are reluctant to release databases with their honeypots data because they contain sensitive information about the structure of their network and the attacks they are facing. In addition to revealing the addresses of their honeypots, it also reveals its configuration. For this reason the creators of Dionaea released a set of data so that the researchers could study the data collected without the need to install a honeypot. Thus, the data set chosen for the experiments reported here is the Paris data set [16].

3.1 Dataset Structure

The information collected by the Honeypot is stored in a SQLite database. SQLite provides a software library that implements an autonomous transactional database service, without the need of servers or setup, as it does not require separate servers or processes. The library reads and writes information directly into the disk [17].

The entity relationship model and the Honeypot database can be viewed in Figure 1 and is divided into five areas:

- A central table where there are the primary information of the attack (*connections* table). This table stores information such as the IP address of the attacker, the IP address of the Honeypot, local and remote ports, time of the attack, connection types, protocol types, etc .;
- On the left there are three other tables that are used to store the information of the attacks against the Microsoft SQL Server service (MSSQL) (tables *mssql_commands, mssql_fingerprints* and *logins*). The information stored in these tables consist of commands sent to the honeypot to compromise the service, users and passwords in brute force attacks, and information about the attackers, such as version connection library, customer signatures, etc .;
- To the right there are four tables that refer to Honeypot firewall logs (p0fs), the resolution of attackers names (resolves) and services emulated by Honeypot (emu_profiles and emu_services). The latter contains the information about the codes used to circumvent the security of the application and send commands so that the Honeypot performs actions aimed at compromising their security and integrity;

ISBN: 978-1-941968-03-1 ©2014 SDIWC

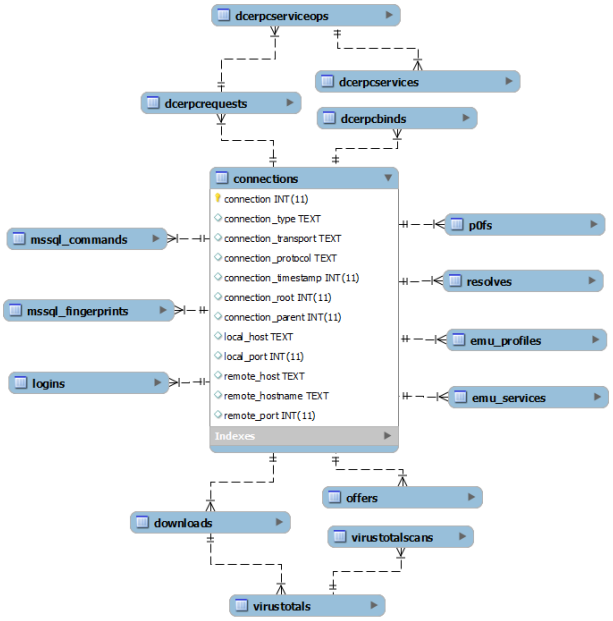


Figure 1: The honeypot database diagram.

- At the bottom there are two sets of interrelated tables. The first one refers to the collection of malicious artifacts, which stores information about where are these files and their MD5 hash (table downloads). This table is linked to two other VirusTotal and virustotalscans tables, which are used to store the information obtained by the VirusTotal tool. The second set relates to the offers table that stores the information concerning the provision of malicious artifacts to the Honeypot. Often it is not possible to obtain the malicious artifact because where it was stored is no longer infected or it is off at the offer time;

- At the top there are the set of tables that refer to the Distributed Computing Environment (DCE) and Remote Procedure Calls (RPC) of the communication protocol of the systems based on the Service Message Block (SMB) (dcerpcservices tables, dcerpcrequests, dcerpcserviceops, dcerpcbinds).

The database has a total of seventy eight attributes divided into sixteen tables, and only forty-two useful attributes, because sixteen of them are connection attributes and relationships between the tables, and sixteen others are sequential indices of the attributes contained in the object table. The forty-two possible attributes to be analyzed are divided into two groups, thirty-seven nominal attributes and nine numeric ones.

The database is divided into five sets that store different information about the attacks that target specific services. This paper addresses three of the five sets of information that are defined by tables: connections; dcerpcbinds; dcerpcserviceops; dcerpcservices; downloads; and Offers.

The set of tables was chosen because it represents attacks on devices with the Microsoft Windows operating system. This type of attack is more than 90% of the attacks recorded by the Honeypot, as will be seen below.

3.2 Descriptive Statistics

The Paris database has the following characteristics (Table 1):

- Number of attacks: 7,822,148 recorded in the connections table. After joining with tables dcerpcbinds, dcerpcservices, dcerpcserviceops, downloads, and offers the number of objects sum up to 19,755,323. This is due to the attacks that use a single connection to explore more than one vulnerability, allowing a record in the connections table to have more than one record in the other tables.
- Collection Period: between 30/11/2009 and 07/12/2009 day.
- Number of attributes: 15, 2 numeric attributes, one attribute of type Date, 1 attribute of type Hour, and 11 nominal attributes.
- Additional information: the attributes are not standardized and those who have objects with missing values were filled with the word EMPTY.

Data were integrated eliminating index and interconnection attributes between the tables.

ISBN: 978-1-941968-03-1 ©2014 SDIWC

This dataset will be used to obtain the rules of the intrusion detection system (IDS).

A descriptive analysis of the resulting data set was performed, identifying that 93.99% of the attacks were directed towards the SMDB service on port 445 (see Table 2). This analysis showed, for example, that the attribute dcerpcbind_uuid presented 95 different classes, being the class 4b324fc8-1670-01d3-1278-5a47bf6ee188 the most frequent one with a percentage of 60.71%. The second most frequent class was the EMPTY one, with a frequency of 27.93%; the other classes had a frequency equal to or less than 0.37%. This class is the DCERPC interface used to connect to the honeypot and it is extremely important to the creation of the IDS rules.

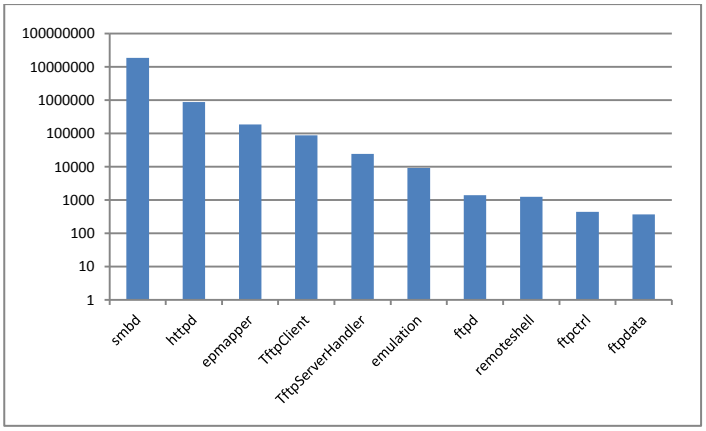
Table 1: Attribute characteristics and descriptions.

Nº	Attribute	Type	Description
1	Connection	Integer	Table Connections Index
2	Connection_Transport	Nominal	Transport Kind (TCP, UDP, TLS)
3	Connection_Protocol	Nominal	Connection Protocol
4	Local_Port	Integer	Honeypot local port (de 0 a 65535)
5	Remote_Host	Nominal	Attacker IP Address
6	DceRpcBind_UUID	Nominal	RPC Bind Interface
7	DceRpcBind_TransferSyntax	Nominal	RPC Bind Interface Transfer Syntax
8	DceRpcService_Name	Nominal	RPC Accessed Service Name
9	DceRpcServiceop_name	Nominal	RPC Operation Service Name
10	DceRpcServiceop_Vuln	Nominal	Microsoft Security Report Name
11	Download_URL	Nominal	Download URL
12	Download_MD5_Hash	Nominal	MD5 Hash from downloaded binary
13	Offer_URL	Nominal	Attackers Offer URL
14	Connection_Date	Date	Date of Attack in format YYYY-MM-DD retrieved from TimeStamp
15	Connection_Time	Time	Time of Attack in format HH:MM:SS retrieved from TimeStamp

Table 2: (a) Protocols Frequency Distribution of Attacks Against *Honeypot*. (b) Frequency Distribution of Protocols Type.

Connection_Protocol	**Quantity**	**Relative Frequency (%)**
smbd	18567380	93,99
httpd	877879	4,44
epmapper	186660	0,94
TftpClient	86686	0,44
TftpServerHandler	24139	0,12
emulation	9126	0,05
ftpd	1396	0,01
remoteshell	1255	0,01
ftpctrl	436	0,00
ftpdata	366	0,00
Total	19755323	100,00

(a)



(b)

The analysis gave rise to several interesting facts about the data set. For example, when analyzing the attribute dcerpcserviceop_name it can be observed that vulnerabilities were exploited in three DCERPC services with an identical frequency of 22.57%, which together represent 67.71% of the data set: NetPathCanonicalize, NetPathCompare and NetShareEnum. The rest is divided into five classes: EMPTY with 31.99%, which is related to attacks that

had not explored DCERPC services and therefore are with EMPTY value; class RemoteCreateInstance with 0.23%; class RemoteActivation with 0.06%; DsRolerUpgradeDownlevelServer class with 0.01%; and NetAddAlternateComputerName class with 0%.

Along with these data it could be observed that 45.14% of the exploited vulnerabilities refer to the Microsoft Security Bulletin MS08-67, in which it is informed that the vulnerability could allow remote code execution on the server. Another vulnerability pointed out by the analysis is provided by the Microsoft Security Bulletin MS04-12 with frequency of 0.23%, followed by the MS03-26 security bulletin with 0.06% and, finally, the bulletin MS04-11 security with 0.01% frequency.

It is noteworthy that the honeypot was not able to relate 22.57% of the attacks reported to a known security bulletin. This can occur because the honeypot is not targeted at the vulnerabilities captured because they are new vulnerabilities or variations of vulnerabilities documented in Microsoft security bulletins. These records have been named as Not_Identified. The rest of the analyzed objects (31.99%) did not explore a DCERPC failure and, thus, have been identified by the EMPTY class.

After the descriptive analysis it was found that the attacks on the honeypot were directed to Windows services and, therefore, IDS rules were designed to identify attacks on equipment with Microsoft OS.

3.3 IDS based on Decision Trees

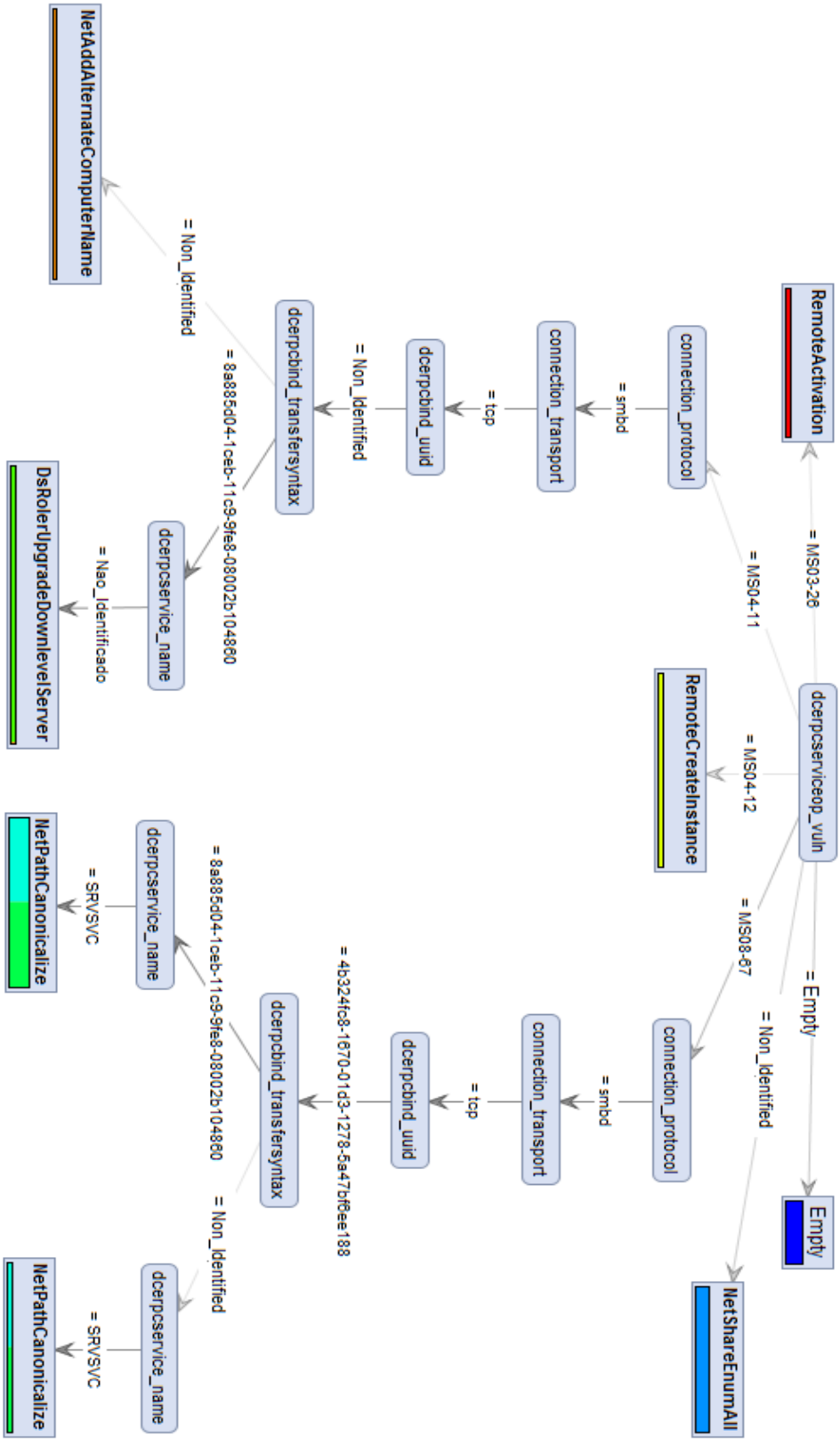
The use of Decision Trees (DT) as a model to classify malicious activity is interesting because of both their classification performance and the possibility to extract rules that identify each type of attack. Moreover, once generated the decision tree it can be used to identify anomalous malicious activity [18].

According Markey [18] decision trees are techniques that help in the analysis of large sets of data for intrusion detection, being able to answer questions like, "What rules should be used to distinguish malicious traffic from legitimate one?" or "What are the most common features of a scanning activity when compared to other data traffic?". In the experiments performed in this paper, it was chosen the software RapidMiner to implement the DT [19]. To evaluate the selected attributes and the decision tree it was defined the DceRPCServiceop_Name as the class attribute, because Microsoft releases the Remote Procedure Call Protocol Extension [20], which indicates which calls and which subscriptions lead to remote procedures (attribute DceRPCService_name).

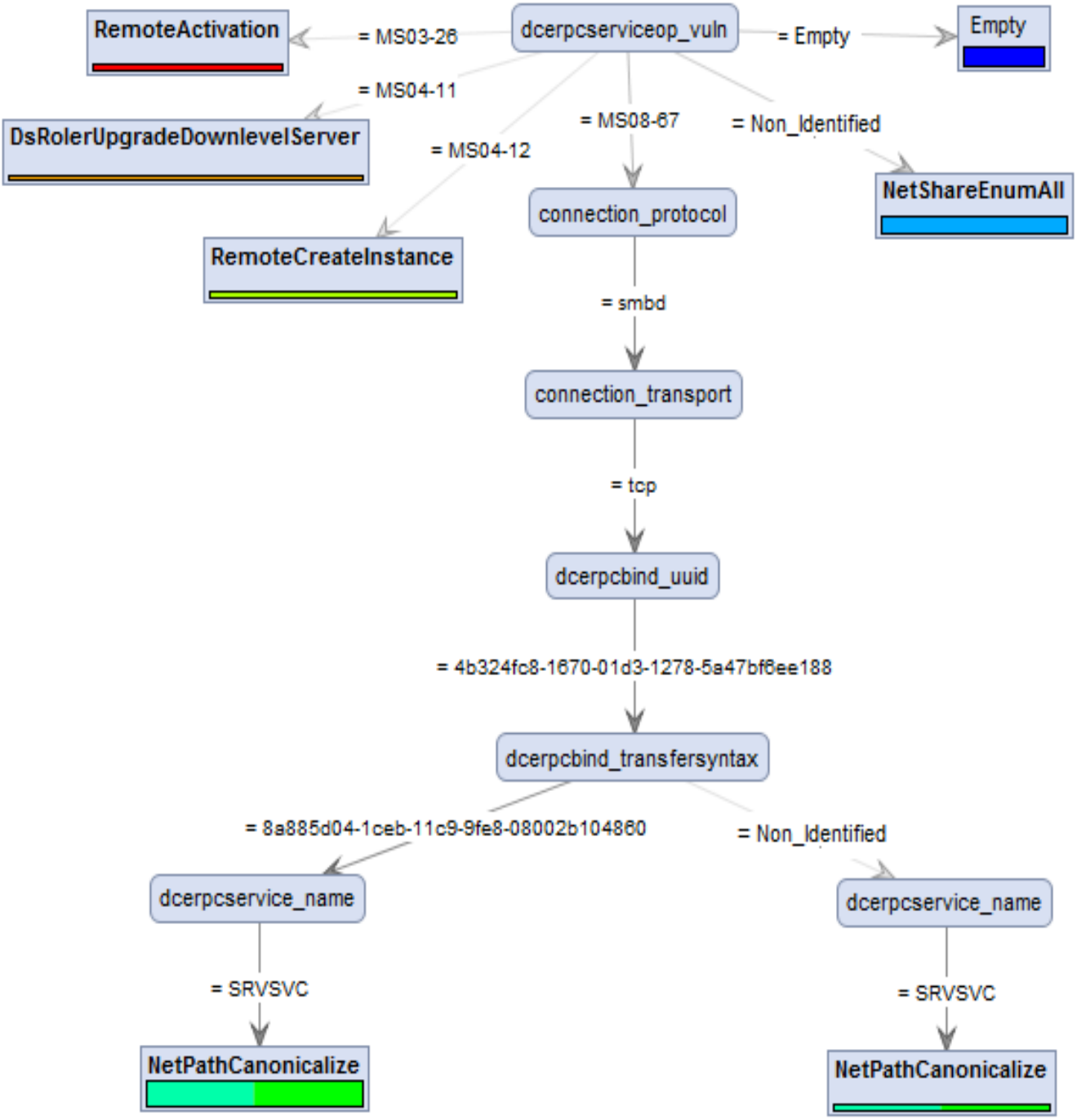
A k-fold cross validation, with k = 10, was used to estimate the classification performance of decision trees. The first difficulty to run the algorithm was the number of existing objects in the database (nearly 20 million). Even running experiments on a computer with 32GB RAM and 128Gb swap, the machine could not handle all the data. Given this difficulty, it was decided to sample the data based on time periods. For the Paris data, the honeypot was active for a period of 8 days and, therefore, it were created 8 data subsets sampled from the total set, each subset relating to one day of collection. For each subset a decision tree was generated.

When analyzing the trees it was noted that three different ones were created, which can be seen in Figure 2.

ISBN: 978-1-941968-03-1 ©2014 SDIWC

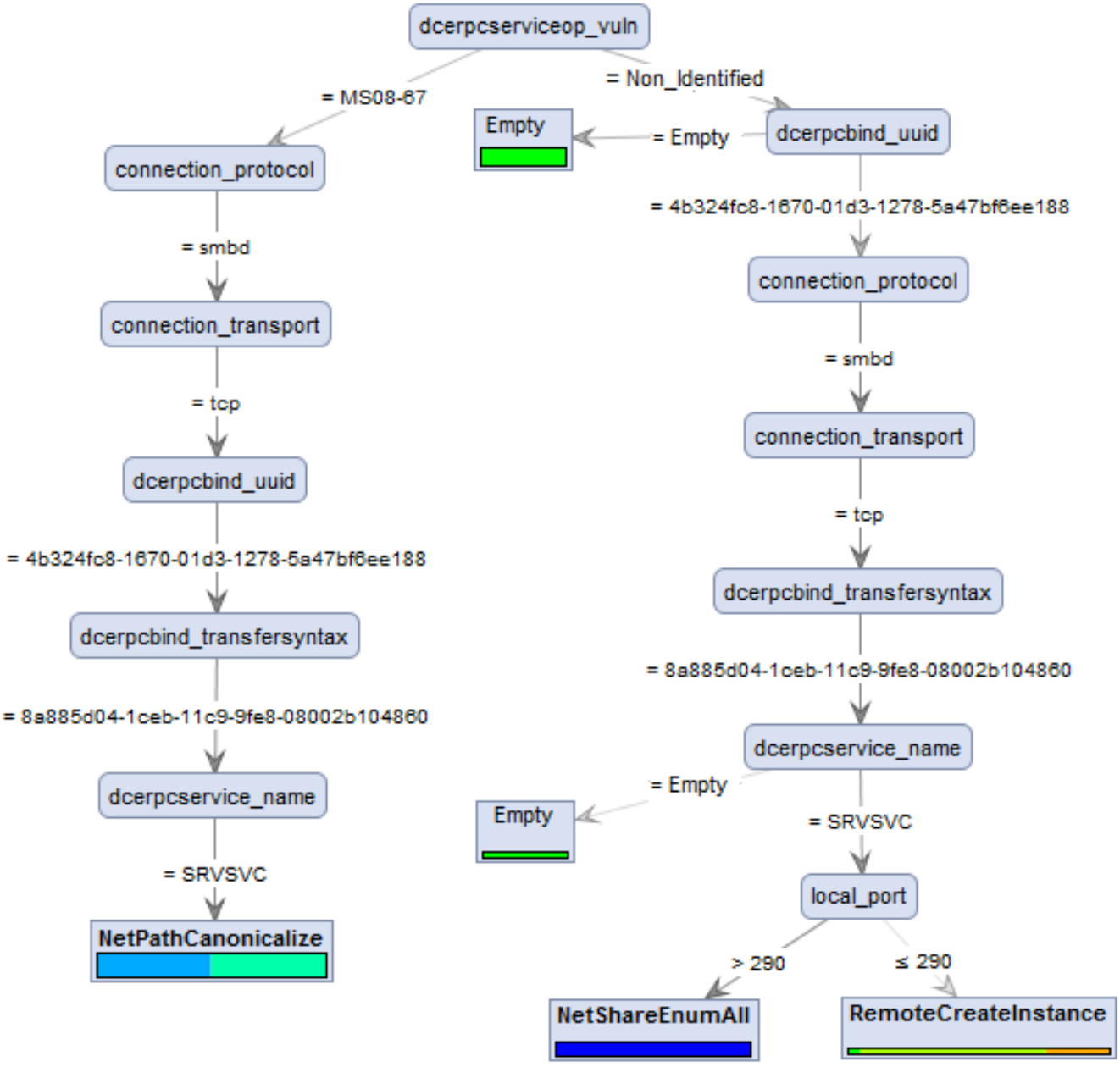

RemoteActivation
= MS03-26
dcerpcserviceop_vuln
= Empty
Empty
= MS04-11
connection_protocol
= smbd
connection_transport
= tcp
dcerpcbind_uuid
= Non_Identified
dcerpcbind_transfersyntax
= 8a885d04-1ceb-11c9-9fe8-08002b104860
dcerpcservice_name
= Nao_Identificado
DsRolerUpgradeDownlevelServer
= Non_Identified
NetAddAlternateComputerName
= MS04-12
RemoteCreateInstance
= MS08-67
connection_protocol
= smbd
connection_transport
= tcp
dcerpcbind_uuid
= 4b324fc8-1670-01d3-1278-5a47bf6ee188
dcerpcbind_transfersyntax
= 8a885d04-1ceb-11c9-9fe8-08002b104860
dcerpcservice_name
= SRVSVC
NetPathCanonicalize
= Non_Identified
NetShareEnumAll
dcerpcservice_name
= SRVSVC
NetPathCanonicalize

(a)



(b)

ISBN: 978-1-941968-03-1 ©2014 SDIWC



(c)

Figure 2: Decision Tree Generated From Paris Dataset. (a) Subset 1. (b) Subset 2, 4, 5, 6, 7 e 8. (c) Subset 3.

3.3.1 Extracting Rules from the Decision Trees

To analyze the decision trees one must follow the path between the root and the leaf nodes. Each path between the root and a leaf generates one decision rule. For the tree of Figure 2(a) starting from dcerpcserviceop_vuln attribute with value MS04-11, the connection_protocol attribute is SMBD, connection_transport is TCP, dcerpcbind_uuid is Not_Identified and the dcerpcbind_transfersyntax attribute splits in two, with Not_Identified and 8a885d04-1ceb-11c9-9fe8-08002b104860 values. If we follow the left side of the leaf, the value is NetAddAlternateComputerName. This rule can be interpreted as follows: a connection that ex-

ploits the vulnerability described in the MS04-11 report used the SMBD protocol on a TCP connection and has not had a DCERPC interface identified neither a transfer syntax, trying to run a call to add an alternative computer name. The resulting rule the right hand side can be interpreted as: a connection that exploits the vulnerability described in the MS04-11 report used the SMBD protocol on a TCP connection, has not had a DCERPC interface identified, the identified transfer syntax was 8a885d04-1ceb-11c9 -9fe8-08002b104860, trying to run a call to change a permission of a domain server. In both cases the Microsoft report says that it is a Buffer Overflow vulnerability, allowing the remote execution of arbitrary commands.

The right hand side branches lead to the exploitation of the vulnerabilities described in MS08-67 and can be interpreted as follows: an attacker exploiting the vulnerabilities described in MS08-67 used the SMBD protocol on a TCP connection to a DCERPC interface 4b324fc8-1670- 01d3-1278-5a47bf6ee188 and a transfer syntax 8a885d04-1ceb-11c9-9fe8-08002b104860 using the SRVSVC service attempted to run the NetPathCanonicalize to convert a path into a canonical name.

3.3.2 Quantitative Analysis

Each of the decision trees generated for each of the eight data subsets was evaluated using a k-fold cross validation method, with k = 10. The percentage accuracy was calculated for each subset, together with the false positive rate (FPR) and the false negative rate (FNR). The values in Table 3 are the average of k-fold for the test set. The results of each subset showed average accuracy values around 75%, average FPR around 3% and average FNR around 14%. Only one set had a lower result with an accuracy of 45%, FPR = 9.74% and FNR = 71.42%.

Table 3: Decision Tree Performance For Each Subset.

Subset	1	2	3	4	5	6	7	8
Accuracy (%)	77.53	76.72	45.42	77.73	77.75	77.56	78.15	75.31
False Positive Rate (FPR)	0,0281	0,0385	0,0974	0,0318	0,0317	0,0320	0,0312	0,0472
False Negative Rate (FNR)	0,3750	0,1428	0,7142	0,1428	0,1428	0,1428	0,1428	0,1666

Table 4: Decision Tree Performance Average From Subsets in Table 3.

	Average	Median	Standard Deviation	C. of Variation
Accuracy (%)	73,27	77,55	11,29	0,15
FPR	0,0422	0,0319	0,0215	0,5110
FNR	0,2462	0,1428	0,1921	0,7804

ISBN: 978-1-941968-03-1 ©2014 SDIWC

Table 5: Confusion Matrix of Subset 3 from Paris Dataset.

	NetShareEnumAll	**NetPathCanonicalize**	**NetCompare**	**EMPTY**	**RemoteCreateInstance**	**RemoteActivation**	**NetAddAlternateComputerName**
NetShareEnumAll:	607351	0	0	844406	6046	2027	4
NetPathCanonicalize:	0	485880	485881	0	0	0	0
NetCompare:	0	121471	121470	0	0	0	0
EMPTY:	0	0	0	0	0	0	0
RemoteCreateInstance:	0	0	0	0	0	0	0
RemoteActivation:	0	0	0	0	0	0	0
NetAddAlternateComputerName:	0	0	0	0	0	0	0

Table 6: Class Analysis of Subset 3 from Paris Dataset.

dcerpcserviceop_name	**Quantity**	**Relative Frequency**
NetAddAlternateComputerName	4	0,00015
RemoteActivation	2027	0,07579
RemoteCreateInstance	6046	0,22606
NetCompare	607351	22,70865
NetPathCanonicalize	607351	22,70865
NetShareEnumAll	607351	22,70865
Empty	844406	31,57206
Total	2674536	100,00000

The analysis of the results shows that subset 3 had the worst performance. To understand this, the Confusion Matrix was evaluated. Table 5 shows that there was a great confusion among the classes and some were not mapped by the decision tree rules. For instance, the classes EMPTY, Remote-CreateInstance, RemoteActivation and NetAddAl-ternateComputerName were not covered by the rules.

Besides the confusion matrix, the distribution of the objects in subset 3 was investigated, as shown in Table 6. It can be seen that: 1) the Net-Compare, NetPathCanonicalize and Net-ShareE-numAll classes have the same number of objects; 2) the model was not able to properly separate the objects in their classes; and 3) when individually analyzing the objects of each class it is found that they have similar characteristics in different classes, which makes it impossible to suitably separate them.

Despite the poor performance for subset 3, interesting features of the attacks reported can be observed. When browsing the tree nodes it can be seen that the algorithm was able to identify that the attacks to the SMBD protocol occurred in non-standard ports (ports> 290 and ≤ 290). This feature raises the hypothesis that the attackers were seeking to compromise other systems or a system configured not to use the default SMB service doors. This may indicate that

the attackers have a knowledge of the network structure in which the honeypot was installed.

3.4 Building IDS Rules Using the Decision Trees

There are many intrusion detection systems on the market and each has specific characteristics for the generation of custom rules. This work presents, as an example, the gereration of a rule for the Snort intrusion detection system. This system was chosen because it has a module capable of processing DCERPC information, which is the main information obtained from the set of honeypot data. Despite that, the trees presented here can be used to generate rules for other systems, such as firewalls or IPS.

To obtain the rule it will be used the tree shown in Figure 3. Vulnerability Exploited (dcerpcserviceop_vul attribute): MS08-67, Protocol used: SMBD, DCERPC Interface (dcerpcbind_uuid attribute): 4b324fc8-1670-01d3-1278-5a47bf6ee188, Sin-tax transfer: (Attribute dcerpcbind_transfersyntax): 8a885d04-1ceb-11c9-9fe8-08002b104860, Service Used (Attribute dcerpcservice_name): SRVSVC, and Not_Identified. Service call used: (dcerpcserviceop_name attribute) NetPathCanonicalize.

The rule generated has the following information obtained from the decision tree:

alert tcp -> The SMBD protocol Works with the TCP protocol

[135,139,445,593,1024:] -> SMBD Ports

(msg:" MS08-67 Vulnerability Attack") -> Message that will be logged in the IDS, this message is based on the vulnerability identifyed by the honeypot.

dce_iface: 4b324fc8-1670-01d3-1278-5a47bf6ee188 - > Interface DCERPC used in the attack, obtained from *dcerpcbind_uuid attribute*.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET [135,139,445,593,1024:] \
(msg:"MS08-67 Vulnerability Attack"; flow:established,to_server; \
dce_iface: 4b324fc8-1670-01d3-1278-5a47bf6ee188; dce_opnum:0-11; dce_stub_data; \
byte_jump:4,-,relative,align,dce;byte_test:4,>,256,4,relative,dce; reference:\
bugtraq,20081026;reference: CVE,2008-4250; classtype:attempted-admin; sid:1000068;)
```

Rule 1: Generated Snort IDS Sample Rule.

dce_opnum: 0-11 -> Number of DCERPC Call, this information can be obtained after a research about the DCERPC bind interface, and the DCERPC transfer syntax, the service attacked and the procedure call used. The search is necessary due the information required are unique of each interface and transfer syntax. Each service call has received an operation number (opnum).

reference:bugtraq,20081026; -> IDS logged information that reference the MS08-67 vulnerability on BugTraq system.

reference: CVE,2008-4250; - > IDS logged information that references MS08-67 vulnerability on CVE system.

The others parameters are default, and must be changed when need.

4 Discussion and Future Work

Honeypots generate vasts amounts of information, making it difficult to quickly analyze the data. Thus, the application of data mining techniques becomes necessary to extract knowledge that can assist the network administrators to protect their assets. The application of the decision tree algorithm was efficient for the generation of intrusion detection rules, since the data is reduced allowing an analytic understanding of the attacks recorded by the honeypot.

The decision trees generated allowed a fast and clear identification of important features for the

ISBN: 978-1-941968-03-1 ©2014 SDIWC

creation of intrusion detection rules. In addition, they analytically represent a data set with almost 20 million objects in just 3 different trees, facilitating the analysis.

Besides the possibility of generating intrusion detection rules, the trees show clearly the attack profiles that the honeypot cannot handle, allowing the administrator to identify and protect against these attacks by changing other subsystems of the network.

There are many other data mining techniques that can be employed to obtain IDS rules, and to assess the network health as a whole. A possible extension of this work is to create an application to obtain rules automatically, as well as the testing and validation of these rules at any IDS tool.

REFERENCES

[1] Cisco System. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update. [Online] Feb. 2013. [Last view: 2014-11-20.] http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html.

[2] Kaspersky Lab. Informe de Kaspersky Lab: Evaluacion del nivel de amenaza de las vulnerabilidades en programas. *Viruslist.com.* [Online] Feb. 2013. [Last view: 2014-11-20.] http://www.viruslist.com/sp/analysis?pubid=207271202.

[3] Ponemon Institute. The Impact of Cybercrime on Business: Studies of IT practitioners in the United States, United Kingdom, Germany,. *Ponemon.* [Online] May 2012. [Last view: 2014-11-20.] http://www.ponemon.org/local/upload/file/Impact_of_Cybercrime_on_Business_FINAL.pdf.

[4] Denning, D.E. *An Intrusion-Detection Model.* 2s.l. : IEEE, Feb. 1987, IEEE Transactions on Software Engineering, Vols. SE-13, pp. 222-232. ISSN: 0098-5589 DOI: 10.1109/TSE.1987.232894.

[5] Cohen, Fred. The Deception ToolKit. *Risks Digest.* 19, March 1998.

[6] Provos, Niels. *A Virtual Honeypot Framework.* s.l. : USENIX Security Symposium, 2004.

[7] Provos, Niels e Holz, Thorsten. *Virtual Honeypots: From Botnet Tracking to Intrusion Detection.* [ed.] Addison-Wesley. s.l. : Pearson Education, Inc., 2007. p. 440. Vol. 1.

[8] Wicherski, Georg. Medium interaction honeypots. *German Honeynet Project.* 2006.

[9] The Honeynet Project. *Know Your Enemy: Learning About Security Threats - The Honeynet Project.* Second. s.l. : Pearson Education, Inc, 2004.

[10] Dionaea Catch Bugs. Dionaea Catch Bugs. [Online] April 2013. [Last view: 2014-11-20.] http://dionaea.carnivore.it/.

[11] Van Rossum, Guido. *Python Programming Language.* s.l. : USENIX, 2007. USENIX Annual Technical Conference.

[12] Deering, Stephen E. Internet protocol, version 6 (IPv6) specification. *Request for Comments.* [Online] IETF.org, 1998. [Last view: 2014-11-20.] https://tools.ietf.org/html/rfc2460.

[13] Dierks, Tim. The transport layer security (TLS) protocol version 1.2. *Request For Comments.* [Online] IETF.org, 2008. [Last view: 2014-11-20.] https://tools.ietf.org/html/rfc5246.

[14] Buffington, Jason. Microsoft SQL Server. *Data Protection for Virtual Data Centers.* s.l. : Wiley Publishing, Inc., 2010, pp. 267-315.

[15] Rosenberg, Jonathan, Henning Schulzrinne, Gonzalo Camarillo, Alan Johnston, Jon Peterson, Robert Sparks, Mark Handley, and Eve Schooler. *SIP: session initiation protocol*. Vol. 23. RFC 3261, Internet Engineering Task Force, 2002. [Last view: 2014-11-20.] http://www.hjp.at/doc/rfc/rfc3261.html.

[16] Post it yourself. *Carnivore News.* [Online] 08 de 12 de 2009. [Last view: 2014-11-20.] http://carnivore.it/2009/12/08/post_it_yourself.

[17] Hipp, D. Richard, and D. Kennedy. SQLite. *SQLite.* may de 2007.

[18] Markey, Jeff e Atlasis, Dr. Antonios. SANS Intitute Infosec Reading Room. *SANS Institute Reading Room.* [Online] 2011-06-05. [Last view: 2014-11-20.]http://www.sans.org/reading-room/whitepapers/detection/decision-tree-analysis-intrusion-detection-how-to-guide-33678.

[19] Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., & Euler, T. (2006, August). *Yale: Rapid prototyping for complex data mining tasks*. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 935-940). ACM. DOI: http://doi.acm.org/10.1145/1150402.1150531.

[20] Microsoft Corporation. [MS-RPCE]: Remote Procedure Call Protocol Extensions. *Microsoft Developer Network.* [Online] Microsoft Corporation, January 2013. [Last view: 2014-11-20.] http://msdn.microsoft.com/en-us/library/cc243560.aspx.