Distributed detection of anomalous Internet sessions

Manuel García-Cervigón Gutiérrez

Computer Architecture Department

Universidad Politècnica de Catalunya

A thesis submitted for the degree of

Doctor per la Universitat Politècnica de Catalunya
Advisor: Manel Medina Llinàs

2012 September

Abstract

Financial service providers are moving many services online reducing their costs and facilitating customers' interaction. Unfortunately criminals have quickly found several ways to avoid most security measures applied to browsers and banking sites. The use of highly dangerous malware has become the most significant threat and traditional signature-detection methods are nowadays easily circumvented due to the amount of new samples and the use of sophisticated evasion techniques. Antivirus vendors and malware experts are pushed to seek new methodologies to improve the identification and understanding of malicious applications' behavior and their targets.

Financial institutions are now playing an important role by deploying their own detection tools against malware that specifically affect their customers. However, most detection approaches tend to be based on sequences of bytes in order to create new signatures. This thesis approach is based on new sources of information: the web logs generated from each banking session, the normal browser execution and the customer's mobile phone behavior. The thesis can be divided in four parts:

- The first part involves the introduction of the thesis along with the presentation of the problems and the methodology used to perform the experimentation.
- The second part describes our contributions to the research, which are based on two areas:
  - Server side: Weblogs analysis. We first focus on the real-time detection of anomalies through the analysis of web logs and the challenges introduced by the amount of information generated daily.

  We propose different techniques to detect multiple threats by deploying per user and global models in a graph based environment that will allow the analysis of highly related data.

  - Customer side: Browser analysis. We deal with the detection of malicious behaviors from the other side of a banking session: the browser. Malware samples must interact with the browser in order to retrieve or add information. Such relation interferes with the normal behavior of the browser. We propose to develop models capable of detecting unusual patterns of function calls in order to detect if a given sample is targeting an specific financial entity.

- In the third part, we propose to adapt our approaches to mobile phones and Critical Infrastructure environments. The latest online banking attack techniques circumvent protection schemes such password verification systems sent via SMS. Man in the Mobile attacks are capable of compromising mobile devices and gaining access to SMS traffic. Once the Transaction Authentication Number is obtained, criminals are free to make fraudulent transfers. We propose to model the behavior of the applications related messaging services to automatically detect suspicious actions. Real-time detection of unwanted SMS redirections can improve the effectiveness of second channel authentication and build on detection techniques applied to browsers and Web servers.

  Finally, we describe possible adaptations of our techniques to another area outside the scope of online banking: critical infrastructures, an environment with similar features since the applications involved can also be profiled. Just as financial entities, critical infrastructures are experiencing an increase in the number of cyber attacks, but the sophistication of the malware samples utilized forces to new detection approaches. The aim of the last proposal is to demostrate the validity of out approach in different scenarios.

- Conclusions. Finally, we conclude with a summary of our findings and the directions for future work.

"Don't tell me what I can't do!"

"John Locke, Lost"

A mi sobrino/a

Acknowledgements

I would like to thank my advisor Manel Medina Llinàs for his guidance throughout this research. I am also grateful to my colleagues from the Computer Architecture Department. Several fellow workers from esCERT, Inetsecur and TbSecurity have played a decisive role in my relationship with security. My special thanks go to my family and friends. Juan, thanks for your knowledge and friendship. Sandra, Shakespeare needs some English lessons from you. Tania, I have stolen a lot of quality time from you, I hope I can give it back to you one day. Finally, I must thank Generalitat de Catalunya for supporting part of this research.

Contents

List of Figures

List of Tables

Acronyms

API Application Programming Interface. 4

ATL Active Template Library. 57, 60

BHO Browser Helper Object. 12, 13, 55–62, 76

C&C Command & Control. 70, 84

CII Critical Infrastructures. 88, 89

CLSID Class ID. 56, 58, 59

COM Component Object Model. 56, 57

CORE Computer Research and Education. 7

CRISIS Conference on Risks and Security of Internet and Systems. 7

DDoS Distributed Denial of Service. 7, 36

DOM Document Object Model. 56

GSM Group Special Mobile or Global System for Mobile Communications. 4

HMI Human-Machine Interface. 88–90

IE Intelligent Environments. 7

ISSE Information Security Solutions Europe. 6, 7

MBA Market Basket Analysis. 34, 47, 50

MBR	Master Boot Record. 64
MCF	Microsoft Foundation Class Library. 57
MitB	Man-in-the-Browser. 4, 5, 9, 11, 12, 34, 62–65, 68, 76, 78
MitM	Man-in-the-Middle. 12, 34, 51
MitMo	Man-in-the-Mobile. 4, 5, 84
OSF	Open Software Foundation. 56
PCA	Principal Component Analysis. 23
RTU	Remote Telemetry Units. 88, 90
SCADA	Supervisory Control and Data Acquisition. 87–90
SFCU	Stanford Federal Credit Union. 3
SMS	Short Message Service. 3, 4
svm	Support Vector Machine. 14
TAN	Transaction Authentication Number. 3, 4, 11, 81, 84
UUID	Universally Unique Identifier. 56
ZitMo	Zeus in the Mobile. 81, 83

Part I

Preface

1

Introduction

Today, almost every environment (e.g. airports, home, office, etc.) is populated with a high number of heterogeneous devices like smart-phones, sensors, laptops, tablets or hotspots. Taking advantage of the different communications capabilities that these devices have, new services such as remote systems management, information-sharing or secure online banking have appeared. However, this plethora of devices sharing something opens the doors to new potential threats putting at risk important personal data or misuse of device resources. Therefore, these environments require new mechanisms able to ensure that devices are operated safely, and that are able to detect and inform about any anomalies that might occur during an interaction.

Online banking has allowed millions of people to make financial transactions since the 80s. In 1983 BBCMicro was presented. It included a television and a telephone as a means of communication that allowed users to display account information and make transfers. In 1994 the Stanford Federal Credit Union (SFCU) launched the first Internet banking service. The system allowed actions such as transfers and loans management. The portal SFCU implemented a password authentication system of up to 14 digits for its 30,000 workers. Nowadays, most banks offer a wide range of services through the Internet such as viewing account, transactions or checks information. Customers can also perform actions such investing, transferring between accounts or making bill payments. In order to enter the online banking site, customers connect to a web server through a web browser and enter a number and a password. Depending on the service used, additional security protection is applied. Short Message Service (SMS) Transaction Authentication Number (TAN) allows financial institutions to provide a second communication channel. In this case, users are sent the transaction information along with a TAN to their mobile Group Special Mobile or Global System for Mobile Communications (GSM) phones. The TAN must be introduced in the web site in order to complete the transaction process and is only valid for a short amount of time.

However, this environment is a key target for criminal organizations willing to pay experts to create suitable malicious samples to get as much profit as possible (1). Malware is the term used to describe damaging code developed in order to perpetrate different forms of fraud. Traditional banking malware used to steal authentication information related to their bank accounts. Over time, techniques have changed and evolved. Nowadays some malware samples take advantage of the banking sessions to directly change the destination of customers' transactions.

Many of the recent attacks have moved from server exploitation to host exploitation (2). Online services provided by financial entities have become one of the main targets and most authentication schemes can be avoided by two malicious techniques applied by many banking Trojans: Man-in-the-Browser (MitB) and Man-in-the-Mobile (MitMo). Malware samples using MitB are able to act between the browser and the bank's web servers through techniques such Application Programming Interface (API) hooking. This technique has been widely used to deploy MitB attacks (3) (4). The main purpose of a hook is to redirect the execution of an application to a malicious code. Once the malware is installed, it will monitor users' navigation in order to check if specific banking sites are visited. When the malware detects a suitable banking session it may perform different tasks such as deviating the user queries to malicious sites or sending credentials and navigation information. Meanwhile, a fake bank web page will ask the user to provide a mobile phone number. An SMS with a link to a fake banking application, that includes a MitMo, feature is then sent to the user. Once the application is installed, the hacker has control over the TANs sent to the user, circumventing the second channel authentication. Table 1.1 shows the evolution of authentication schemes along with the effectiveness against different attack vectors.

Detection of malware actions such as credential theft or destination account modification has become a crucial problem (5) but most signature-based detection techniques are obsolete. Customers, browsers and mobile phones are the main actors within an Online banking session: Modeling their behavior within normal circumstances should help to detect malicious activities.

Authentication method	Phishing	Pharming	MitM	MitB	MitMo
Transaction Authentication	YES[a]	YES	YES	YES	NO[b]
Smartcart + PKI	YES	YES	YES	NO	NO
Software PKI	YES	YES	YES	NO	NO
Tokens, Grid Cards	YES	YES	NO	NO	NO
Cookie, Text, Picture	YES	Maybe[c]	NO	NO	NO
IP Geo/device fingerprint	YES	Maybe	NO	NO	NO
Password	NO	NO	NO	NO	NO

[a] YES. The authentication method is valid
[b] NO. The authentication method is not valid
[c] Maybe. The authentication method is valid under certain cricumstances

Table 1.1: Final results

1.1 Research challenges

1.1.1 Motivations

The main motivation of this thesis is that financial companies need to rapidly detect if a malware may affect their customers in order to provide the necessary measures. Many banks are searching for anti-fraud applications capable of alerting when a money laundering(6) action occurs and protect (7) customers from unauthorized actions over banks Web sites. It is well known that signature based techniques are no longer sufficient and reverse engineering procedures are too time consuming to provide a fast response against new threats. This thesis aims to develop new methodologies capable of detecting when a malicious bank session has occurred and which malware samples are targeting specific browsers and sites.

1.1.2 Goals of the Thesis

The main objectives that this PhD thesis pursues are:

- Seek new methodologies to detect malware actions from the web server side
- Create browser behavioral models to analyze suspicious binaries targeting a banking web portal
- Apply the most successful approaches to related areas such mobile phones and Critical Infrastructures

1.2 Contributions

To summarize we make the following contributions:

- García-Cervigón, M.; Vázquez, J.; Medina, M. Web sessions anomaly detection in dynamic environments, 2009 Information Security Solutions Europe. "ISSE 2009: Securing Electronic Business Processes: Highlights of the Information Security Solutions Europe 2009 Conference". Wiesbaden: Vieweg+Teubner, 2010, p. 216-220. An approach to detect anomalous banking sessions through different data mining techniques in a graph-based environment

- García-Cervigón, M.; Vázquez, J. WADS: a web anomaly detection system.: The 5th International Conference on Intelligent Environments. "Intelligent Environments 2009, Proceedings of the 5th International Conference on Intelligent Environments". IOS Press, 2009, p. 223-227. We propose a framework for detecting malware presence in banking sessions

- García-Cervigón, M.; Morales R. Malware Detection in Ubiquitous Environments. "ISSE 2012: Securing Electronic Business Processes: Highlights of the Information Security Solutions Europe 2012 Conference", 2012. An approach for analyzing mobile phone applications in order to detect MitMo attacks

- García-Cervigón, M.;Medina Llins M. Browser Function Calls Modeling For Banking Malware Detection. "CRISIS 2012: International Conference on Risks ans Security of Internet and Systems", 2012. A technique to detect banking malware on the client side through the modelation of internet browsers

With special emphasis placed on case studies and innovative and robust security solutions implemented by European organisations, Information Security Solutions Europe (ISSE) conference focuses on topics related to the thesis such:

- Embedded Security. Emerging Applications, Smart Grid & Automotive Solutions, Ubiquitous Computing, M2M Security.

- Mobile Security Solutions. Platform Security, Transaction Security, Information Security, Treats & Risks, Privacy Aspects, Management of Mobile Devices.

- Critical Infrastructure Protection and physical Security. CERT/CSIRT European and Global Developments, Resilience of Networks and Services, surveillance technics and analytics.

- CyberWar, Cybercrime and Forensics, Fraud Detection & Prevention. Distributed Denial of Service (DDoS), Attacks and Countermeasures against industrial Infrastructures (SCADA).

ISSE conference allowed us to get in contact with security baking experts in 2012 and will also be very valuable to gather impressions from Critical Infrastructures researchers.

Intelligent Environments (IE) conference helped us to get feedback from experts dealing with Data mining techniques and heterogeneous scenarios. It was possible to show some initial results of the weblogs analysis and understand the importance of working with other sources of information.

Conference on Risks and Security of Internet and Systems (CRISIS) is a Computer Research and Education (CORE) ranked conference dedicated to security challenges related to Internet applications, networks and systems. The conference brings together computer and network security researchers from industry, academia and government and is especially focused on critical infrastructures and banks. Figure 1.1 shows the research topics that contitute the thesis.
Some of the ideas expressed in this thesis have been also presented in the following workshops:

- W3CAR Workshop: Augmented Reality on the Web

- XV Jornadas del cable y la banda ancha en Catalunya (Spain, 2010)

- FIRST/TF-CSIRT technical colloquium (Italy, 2012)

- Workshop on Cyber Security and Global Affairs and Global Security Forum (Spain, 2012)

1.3 Dissertation outline

The remainder of this thesis is organized as follows. Chapter 2 discusses the state of the art on the field of malware analysis. Chapter 3 describes the steps followed

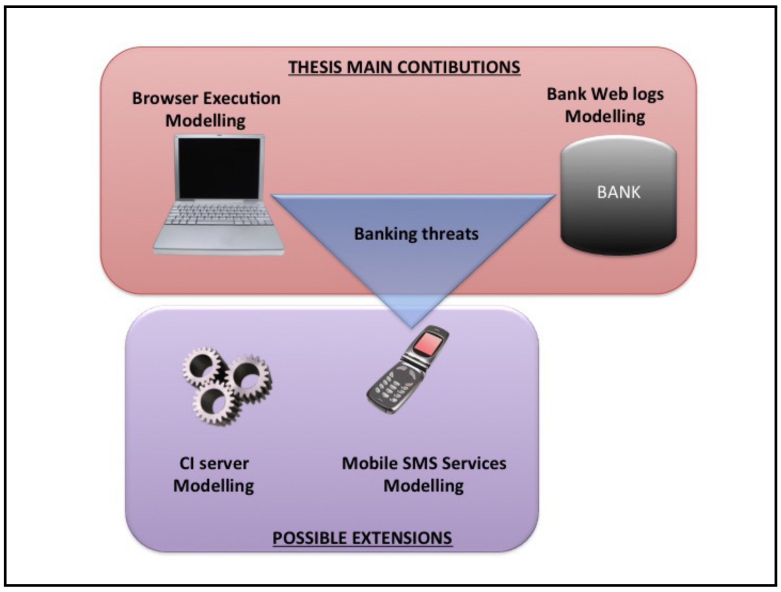


Figure 1.1: Thesis Outcome

to prepare the needed testing environments. Chapter 4 gives an account of the Data Mining techniques applied to banking web logs to develop new detection vectors through semi-supervised techniques among others. Chapter 5 and 6 focus on the detection of malicious actions through information extracted from the victim's browser. New attacks targeting mobile devices to steal banking information will be detailed in Chapter 7. Chapter 8 provides possible applications of our dynamic instrumentation techniques on other areas such Critical Infrastructures. Finally Chapter 9 details the conclusions of the research.

2

State of the art

We must assume that malware writers have been ahead of researchers so far. However, until the late 90s, a deep knowledge of programming and security was needed to develop new samples due the to little information on the area available. Most hackers generated new samples for popularity reasons. However, the introduction of money (financial companies, banking) has changed the rules. Nowadays, malware developer kits are sold on the Internet and almost anyone can create a new trojan targeting a bank site. The remainder of this chapter details the evolution suffered in both sides of the board. A malware classification is first described in Section 2.1; the online banking problem is introduced in Section 2.2; next, MitB techniques are detailed, and finally, malware detection approaches are explained in 2.4.

2.1 Malicious software

The amount of malware samples has experienced an significant growth in the recent years due to the benefits provided to malware creators and mafias. Malware samples may vary depending on their goals and behavior (8):

- Viruses are able to self-replicate by attaching themselves to a file. Once the infected file is executed, the virus is run.

- Worms also self-replicate but do not need a file to do so. A worm normally copies itself through the network.

- Trojan horses simulate the behavior of a real application in order to perform malicious activities such as the hijacking of credentials.
- Backdoors aim to open a port in the infected system to allow remote control.
- Spyware is usually introduced in a system after downloading a trial version of an application. Once installed, spyware samples will gather personal information such as clickstream or key strokes.
- Rootkits provide the attacker with a number of tools to obtain privileged access hiding any possible attack indication. The samples normally include a backdoor so the attacker may reach again the system easily.
- Botnets are a collection of compromised computers. Botnets a generally used for spamming purposes or Distributed Denial of Service (DDoS) attacks.

Traditional malware samples have evolved in order to avoid common detection techniques. Nowadays most malware samples include methods to circumvent detectors. Depending on the hiding technique, the following type of malware can be found:

- **Polymorphic malware**. Polymorphic malware aims to continuously change the code through a polymorphism engine. Polymorphic malware is commonly created along with ciphering techniques so most of the code is ciphered with new keys while the decipher code is modified through the polymorphism engine. Nowadays, polyformism is still one of the most widely spread threats. Server side polymorphism allow attackers generate unique samples of malware, normally attached to spam emails.
- **Metamorphic malware**. In this case, the malware also changes the appearance, but ciphering techniques are not used. Instead garbage code is added to the body. Changes between equivalent opcode instructions may also be performed.
- **"EPO" malware**. Entry-point obscuring malware consists of a sample capable of redirecting the flow of a host application through changes in specific variables or API calls.

2.2 Banking Trojans

The first banking Trojans were modifications of BackOrifice, a well-known program that permitted users to remotely control a computer running Microsoft Windows. The next generation involved malware with some automatic capabilities that included the ability to steal user credentials. Some versions of Win32.Banker (9) could steal credentials from more than 30 banks by intercepting events from mouse and keyboard. In the case of the third generation, new samples such as Win32.Grams (10) , could not only steal and send the credentials but also make transfers to other accounts with details of the victim. Some malware samples such Sinowal have been able to achieve more than 500,000 banking credentials. The next step is for criminals to use the credentials. Banks can handle these types of attacks by using random passwords. Virtual Keyboards with dynamic numbers are just a temporal solution because more sophisticated Trojans such as Nucklus came with the ability to capture an image or video authentication. In some cases, Trojans do not only steal credentials but also the customers' banking information. In order to accomplish such actions, the malware must crawl into all the links within the user web account. As mentioned previously, due to the use of MitB attacks, trojans can directly control the victims' browser so most of TAN techniques become obsolete. MitB attacks can change queries information in real-time so the malware can easily redirect a user to a phishing web site or even change the destination account of a transaction.

2.2.1 Mules

The money mule's technique consists of convincing someone (the mule) to sign up for a new bank account and withdrawal an amount of money after some transactions are made from hacked accounts to the mules new account. Normally there is little time between the transactions and the moment the mule gets a phone call in order to withdraw the money. The automatic analysis of the number of account nodes that just generated edges to the mule's account node as well as the period of time between the transaction and the withdraw can give a rapid idea of the presence of mules, that is, accounts from the same country as the source used to transfer money that will afterwards be again transferred from the mule account to a foreign country in a few amount of transactions. China or Russia are common receivers of mules transactions.

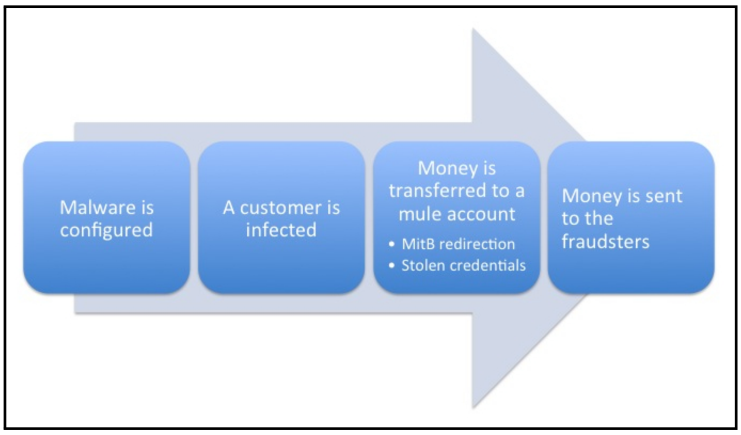


Figure 2.1: Fraud process

Transfers to the mule account can be performed through MitB redirection or with stolen credentials. Figure 2.1 shows the fraud process.

2.3 Banking malware techniques

Malware techniques have evolved along with the attack goals. Most traditional **key-logging techniques** are based on Browser Helper Object (BHO) using functions such GetAsyncKeyState() provided by keyboard APIs in order to record a key whenever it is pressed. Man-in-the-Middle (MitM) was also a common threat performed by altering proxy settings, making DNS modifications or through changes in the network routing. This attack vector permited it to inject new forms and intercept confidential information. However, errors in the certificate and the generation of blacklists were a limitation.

However, MitB changed the fraud paradigm because the direct control of a banking session was possible. MitB attacks allow hackers to steal login information or modify the content of a transaction circumventing protection mechanisms such as SSL and out-of-band authentication. Samples such Zeus, Spyeye or Nuklus have infected millions of Internet Explorer and Firefox Browsers in order to perpetrate different attacks. API-Hooking is generally used to perform MitB between the executable application (EXE) and its libraries (DLLs). It is known to be one of the most dangerous attack vectors so far (3) (4). The main purpose of a hook is to redirect the execution of an application into a malicious code. Hooking methods aim to intercept events or function calls between differnt software applications. Application Programming Interface APIs are a set of functions offered to be used by other software. In order to spy on banking sessions in Internet Explorer, the Windows Internet API (WinINet) function calls allows applications to interact with internet protocols such as HTTP or FTP. Two techniques can be used to deploy an API-hooking sample:

- **IAT modification**. In this case, firstly, the malware injects itself in some user space processes and when it detects that it is injected in a browser, the IAT of the application is modified so when the browser requests a URL by calling HttpSendRequest, the control is transferred to the malware code. The malware is then able to collect all the POST information like users and passwords. Some malware families such Bancos patches specific functions, such as GetMessage() or PeekMessage(), in order to monitor keystrokes. Unlike traditional key-logging techniques, the malware will certainly obtain browser keystrokes by monitoring browser specific functions.

- **Browser Plugins**. In the case of Internet Explorer, BHO drops a DLL file in the disk so signature based antivirus may easily detect them. Although some BHO based families are easy detectable through Internet Explorer's tools menu, hiding techniques limit the detection rate (see Chapter 5).

2.4 Malware detection

A malware detector can be defined as a function with a set of files and two possible results $\{malicious, benign\}$ (11). The effectiveness of a malware detector is defined based on the number of false positives, false negatives and the hit ratio. Most detection techniques can be classified in two groups: Signature-based detection and anomaly based detection (12). Most Signature-based detectors include signatures which are sequences of bytes belonging to the malware. However, complex malware samples perform changes in the code in such way that, after the sample execution, the new code may differ completely from the original one, allowing the malware to bypass the malware detector. Anomaly detection applications monitor the behavior in order to find deviations. Two approaches should be considered:

- **Behavior-based detection**. Behavior-based detection is performed in two phases: During the training phase, the detector will learn how the system or application behaves through different mining techniques in order to generate a model. The evaluation/detection phase allows the model to be compared against the current state of the system in order find possible deviations (13). However, behavior based techniques show a high rate of false alarms due to possible legitimate behaviors not present during the training phase (14).

- **Specification-based detection**. In this case, the patterns of "good behavior" are manually defined after an analysis of the system execution. This technique allows experts to determine a single signature to detect multiple samples that share a common behavior. Some approaches such Panorama (15) allow experts to collect system information in order to generate a set of rules.

2.4.1 Anomaly detection algorithms

Anomaly detection algorithms and more specifically classification algorithms showed good results to detect new types of threats as deviations from normal usage. In this case, different approaches must be considered:

- Supervised classification. It is based on the availability of information about the outcome of past observations, i.e. the class to which they belong is known *a priori*. KNN algorithm allows experts to group observations with similar characteristics based on the type of elements close in space. Yang Li et al.(16) integrated KNN along with Transductive Confidence Machines to detect anomalies within a network backbone. In many cases it is not possible to classify observations with several variables based on linear classifiers as it is not possible to draw a line separating the two classes. Support Vector Machine (svm) generate a hyperplane of dimensions which enables it to differentiate sets of observations accurately.

- Unsupervised classification. Algorithms with a classification methodology that does not include the use of a priori knowledge, that is, unsupervised learning, are also valid in some situations. Clustering techniques pursue the discovery of groups with common interests and behaviors. Statistical approaches are based on defining a probabilistic model to evaluate if an instance can be generated through such model. Gaussian distributions can also be useful to determine anomalies within sets of data where the assumption is that such data follows a normal distribution. In this case the anomaly score is based on the distance of a certain sample to the mean.

Given the deployment of different classifiers from a data set, it is possible to combine them all through techniques to get a final decision rule (17):

- Hybridization. The final classifier is inducted through one or more paradigms. Bayesian Rules, Bayes Trees Logistic or Model Trees can be applied.
- Bagging. The technique consists in combining the decisions given a base clasifier.
- Boosting. In this case the rule is decided combining different decisions given more than one base classifier.
- Manual scoring. Each classifier is given an importance based on the knowledge of the experts.

2.5 Sources of information

Different sources of information should be taken into account in order to study malware samples attacking financial institutions. Bank web logs provide information related to the source and time of the connection. A preprocessing of the raw logs can also provide the clickstream followed during each web session. On the other hand, data from a banking session can also be extracted from the user side.

2.5.1 Web Logs Analysis

Although numerous detection techniques are being developed and implemented by financial institutions, little research work has been found around the detection of online frauds through the analysis of anomalies. Clickstream analysis is highly related to web predictions techniques. Calculating the probability of transition from one page to another provides valuable information regarding the normality of a given clickstream. One of the main goals of this thesis is to find the most suitable algorithms that can generate more reliable models of detection for the different sources of information.

Profiling classification of clickstream was analyzed by Banerjee et al. (18) using clustering of common subsequences. Xiao et al. (19) performed some analysis on web logs: Use based (UB), Frequency based (FB) and based on sequence order (VOB). It was found out that VOB presents more efficiency for prediction. K-order Markov techniques are useful to predict the next page to be visited from the information regarding the last k pages visited before.

2.5.2 Software Analysis

Software analysis is mostly used by antivirus researchers to study suspicious samples, gather information on its behavior and generate patterns to detect it. However, software analysis can also be useful for determining good behaviors of the applications participating in a ubiquous communication. Two different classifications to extract information from an application should be considered:

- **Static and dynamic analysis**. Static analysis involves the study of an application based on the source code or machine code without running the software. Static analysis techniques include string searching, API call tracing or the study of the program flow. In most cases the use of packing and encryption algorithms involves a previous task of extracting the real code. Sharif et al.(20) designed a prototype to automatically analyze packed samples. Some machine learning techniques were used to detect patterns such sequences of opcode n-gram patterns (21). Static Call graphs (22) can provide representative information on the behavior of an application. Dynamic analysis techniques require the execution of the binary. Vinod et al. (23) implemented MEDUSA, a framework capable of generating signatures from the dynamic execution of applications. Programs performing dynamic analysis need to introduce extra code or instrument the application. Such insertion is normally made inline in such way that injected code should not interfere with the normal performance.

- **Source and binary analysis**. Compilers perform source analysis, which involves analyzing the source code of applications. Binary analysis is more suitable for the study of malware. It involves the analysis at levels of byte-code. It allows experts to gather information regarding instructions and function calls.

2.6 Detection Scenarios

Antivirus manufacturers, financial institutions and users can be part of the fraud detection process. The scenario where Antivirus manufacturers generated a signature given a new malware and users update their software to stay safe has changed. The increase in the number and type of samples has forced banks to also help users by providing awareness information and specific tools such as new authentication schemes or even specific protection software against online banking fraud. New antivirus even include heuristic capabilities and behavioral analysis in order for the user to automatically detect new threats without the need of a signature. Each detection technique has its own source of information and environment. In order to understand our approach, an overview of the pros and cons of detecting malware from each perspective is given next:

- **Antivirus manufacturer's environment**. Antivirus companies analyze dozens of thousands of suspicious samples daily to find out whether they are malicious and subsequently generate the corresponding signatures or heuristics. Symantec stated in the 2012 **Internet Security Threat report** that its automated analysis systems studied 403 million unique variants of malicious software in 2011, a 41 percent increase from the 286 million analyzed in 2010. Two sources of information are used mainly by antivirus manufacturers to collect suspicious samples:(1) **Honeypots** (24) can provide samples trying to attack a trap computer; (2) **users** can also provide information; many research labs publish a website or e-mail account where users can send suspicious samples. Finally, some entities include **malware repositories** (25, 26) available for researchers. A major advantage of **binary analysis** within research labs is the absence of computational limitations. However, most recent samples manifest only in specific circumstances, so incomplete results can be found and virtual environments performing automatic executions can generate wrong results. For example, the recent flashback mac virus includes encryption keys based in the users machine thus being executed only in the victim's computer (27).

- **Financial institution environment**. In the area of malware fighting, the goal of financial companies is to protect users of those malware that may affect customers' connections to its website. Nowadays, Data Mining techniques and binary

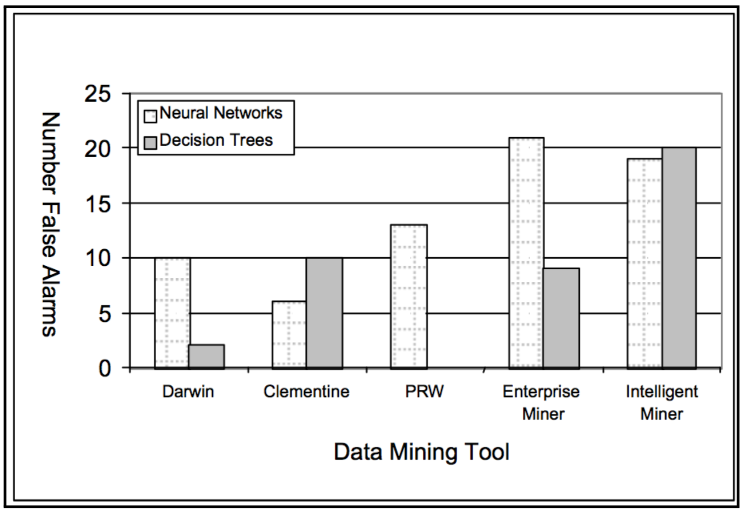


Figure 2.2: Hit ratio

reverse analysis are the most used techniques. However, most companies only tend to use Data Mining tools to assess abnormal behavior. The use of this tools has been extended from the 90s (28, 29, 30, 31). Abbott et al. (32) analyzed five well known data mining tools to grade the accuracy on the detection of fraudulent transactions.

Neural networks and decision trees showed the best results. The numbers are detailed in figures 2.2 and 2.3.

Although the hit ratio is the most important variable when assessing a fraud detector, other variables such as the ability to handle large data sets should be considered. Apart from traditional honeypots and IDS, financial institutions deploy Data Mining detectors to gather malware. The **detection procedure** works as follows: (a) The Data Mining application is deployed and **trained**, (b) If an anomaly is detected during a transaction, the session is stopped and the customer is contacted. In case the anomaly is confirmed as true, the customer disk is copied if a new threat is detected and the installed binary is acquired. Once the suspicious file is gathered, some entities prepare their security team to perform binary reversing techniques on known samples to check if the bank site is present in the configuration file as a possible attack target (33). However, malware reversing techniques are time consuming, thus increasing the probabilities to infect

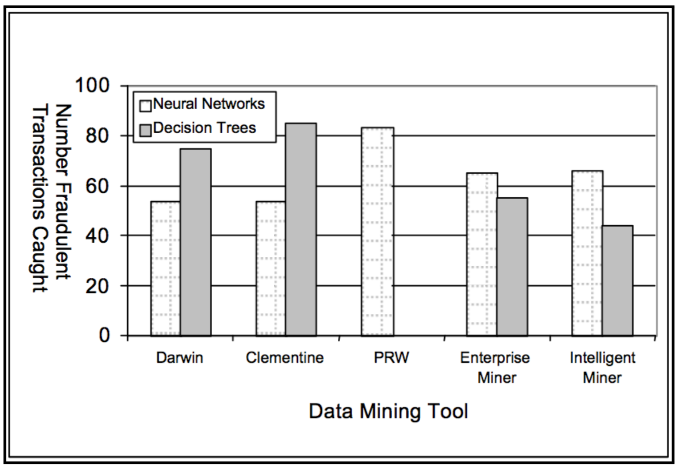


Figure 2.3: False Alarms

customers. Thus, actual techniques suffer from a high rate of false alarms and a time consuming analysis technique.

- **End-user computer**. Since the early signature-based antivirus, detection of known malware was carried out on the computer of users. However, the use of techniques based on behavior involve a run-time overhead. Due to performance limitations, a user computer does not allow a thorough analysis of the behavior of malware. Some approaches (34) combine the power of the research labs with a real user environment. Basic execution information is sent to the cloud lab, where the relevant analysis is conducted.

3

Methodology

This chapter describes the infrastructure developed to accomplish the proposed goals. Weblog and browser execution analysis environments can help Financial entities to detect anomalous sessions and rapidly confirm if a malware sample is targeting the entity. In the case of Server side analysis, a data set based on web logs or similar was needed.

However, it was not possible to set up a device within a banking network due to policy restrictions. Finally, the participation on a Spanish funded project allowed us to obtain anonymized data extracted by the experts from the correlation server. Due to the information contained in web logs (user credentials or transfer data), only a collaboration based on a strong NDA was possible. Section 3.2 details the steps taken in order to analyze and preprocess the data set.

On the other hand, customer side research required the deployment of the specific environment to gather information on browser and mobile behavior. Due to the nature of the scenario, the set up was not complex in terms of hardware requirements. However, a study of the most suitable software for information extraction was needed before starting the experimentation. All the scenarios suffered from a huge amount of observations so special care was taken in order to preprocess the needed information and move it to environments allowing fast queries.

3.1 Web Logs Analysis

Our hypothesis was that logs stored by a financial entity should provide valuable information to generate models capable of detecting fraudulent actions. One of the goals of our research is to find a technique capable of reducing the time needed to detect malware targeting a financial institution. Fortunately, the approach became a project funded by Generalitat de Catalunya. One of the project partners was a financial institution that provided two data sets with the following characteristics:

- The first data set (hereinafter data set A) contained only known valid sessions. The second data set (hereinafter data set B) contained 7 malicious sessions, but it was not labeled. However, both data sets contained queries from the same users and period of time.

- Both data sets were extracted from queries generated by 433 users during the first four months of 2008. It is, indeed, an small trace compared with the bank online activity. According to published statistics, the entity had 5,4 million online users that year, that is 29% of all online-users in Spain. More than 125.000.000 monthly transactions were done through the site.

- Some values of the observations were anonymized due to privacy reasons. Also, the real name of the variables was omitted in this thesis due to non disclosure agreements. Although the source of the logs were the servers, some information apart from the query itself complemented each query. Most bank sites integrate information from various sources such as web log, timestamp or authentication server in a single log. Thus, each observation of the data set contained the following comma separated values:

  - Eventdate. This variable included a timestamp of the query.

  - IP. The IP can help to identify the place where the user is connecting thus providing localization information.

  - Userid. Each user is given a unique user. This field allows us to easily recognize all the events generated from a specific credential.

Variable	Example
eventdate	11122008091222
IP	147.83.132.2
userid	46746288
Browser	Mozilla%2F4.0+compatible%EN
language	english
URL	query?OM=LGN&OD=0& FLAG_DEMO=0& TIPUS_IDENT=2& FLAG_PART_EMPR=1

[a] It includes several sequencial observations

Table 3.1: Web log example

- Browser. This field is equivalent to the *"user-agent"* field found in any web log. It describes the type of application, generally browser, that is connecting to the server. It can include information like version, operating system or the language of the client.
- URL. URL comprises the whole request made by the user's browser. It includes many important tuples variables-value such as the section and specific page. The combination of the variables OM and OD identifies the unique page within the intranet that the user is requesting. OM is referred to a section and OD describes a unique specific page. The total number of combinations found in the data set implied 2.240 different pages. Each combination OM-OD is accompanied by a set of variables indicating specific values such bank accounts or amounts.
- Language. This field is also added to the original web logs by the bank and informs about the language chosen by the user.

Given the information provided, different models can be generated. In order to properly analyze the banking sessions all the queries were grouped in sessions. The preprocessed dataset was a list of banking sessions, where each observation contained a list of the queries performed by the user. This format allows to compare clickstreams among users.

Although data sets with multiple variables are preprocessed many times through techniques such Principal Component Analysis (PCA) in order to reduce the dimensionality (feature extraction), the results of such processes are not easily interpretable, and therefore incomplatible with the use of the experts knowledge in order to finally provide a specification-based framework. Regarding the posibility of applying feature selection algorithms, the option was discarded due the relationship among observations. Different features can be part of thes ame session, thus, pruning observations will affect a clickstream analysis.

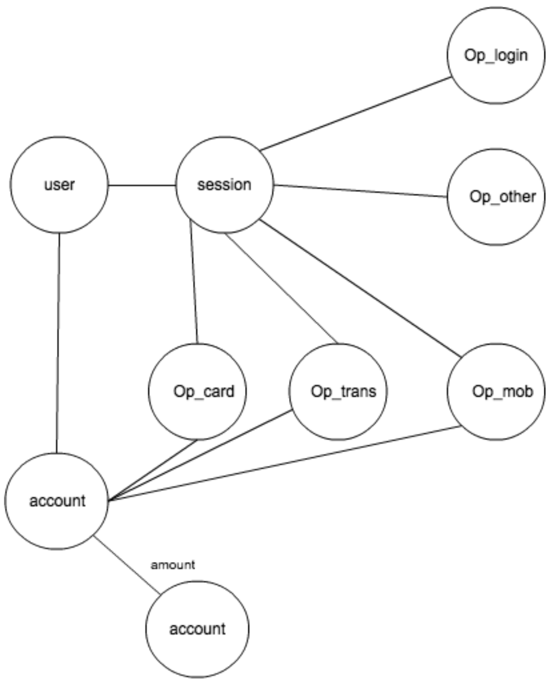
3.1.1 Storage system

The huge amount of information generated from online connections and the need to provide real time analysis implies the deployment of a system capable to handle such a special scenario. The increasing use semi-structured data such as XML texts or trees benefited the use of data mining but the amount information grows exponentially and new ways of structuring data such topological structures have emerged in order to better understand some environments.

As stated before, the amount of information to study and the need to have it analyzed within milliseconds is an important issue in online-banking. Information regarding the money transactions made within a session is also a valuable source of information that must be correlated with the results extracted from the clickstream anomaly detection system. In order to rapidly analyze each transaction the information is copied to Graphs Based Systems, where different types of nodes, new efficient relationships and levels of abstraction can be made in order to reduce the time to perform a query and manage different heuristics. DEX [4] has been chosen as the graph storing system due to the high performance and the possibility of integrating different types of information sources. The query system facilitates the insertion of heuristics. Moving data to nodes and edges involve some challenges when e-banking transactions are handled. Information must be simplified into nodes, edges and attributes. As an example, different types of account identifiers must be merged into one type of node. The graph system should be capable of creating new rules based on the analysis of click stream anomalies, that is, every time a anomalous clickstream is detected, the operations made during that session are compared to ones from other anomalous clickstreams with certain similitude. Operations made during regular sessions are also important to update the different edges (transactions) among nodes (account) and the thresholds. The following nodes were established.

- **User node**. This node will include not only the identification data and information on different threats and anomalies associated to the sessions generated with his credentials. The next chapter will cover different the concept of threat and the anomalies that can be extracted from web sessions.
- **Session node**. The Session node describes the sequence of operations performed by a user. It includes the following attributes:
  - session id which identifies the node.
  - timestamp. It includes the time when the session was initiated. The original value of the time is automatically changed to store the time of the origin country of the connection. This change is of importance in order to generate some models.
  - IP. IP used during the session. In case the IP changes during the session the user is logged out.
  - Session language. Language selected during the session. The language is configured by the user and it does not change unless the user changes the configuration. In the case of the given data set, it was possible to choose among 23 languages.
  - Browser language. The language found in the user agent. Unfortunately only 25% of the queries included such information.
- **Account node**. Each transaction includes a source and destination account and each user may have more than one account.
- **Operation node**. As described before, more than 2.200 actions can be performed within the bank site. However only few of them can involve data theft. The bank provided the selection of the actions where fraud generally occurs: charge of virtual cards, and mobile phones and transactions. Although other actions such investments could also be dangerous, the percentage of users performing such operations is low . Each of these risk actions are detailed in an specific node:
  - Op_login. An Op_login indicates the first query provided by the user.

Figure 3.1: Node relations



- Op_card. This node id is generated when a user charges a virtual card from the bank site. Virtual cards are widely used for ecommerce purposes.

- Op_trans. The transfer node details the amount of money transferred in a given transaction.

- Op_mob. Most banks include the possibility of charging mobile phones' accounts.

- Op_others. The rest of the queries performed are stored in a Op_other node.

Finally, all the nodes are linked through a set of edges. An edge may also include an attribute. The figure 3.1 illustrates the initial relationships among nodes. The graph will evolve in order to support different detection methods. Appendix 1 details the DEX scheme.

3.2 Software Analysis

Anomalies found in applications must be searched with models based on datasets that contain useful information on the software behavior. It is expected to detect abnormal behavior during the execution of the infected application, thus a real-time extraction of information is assumed during such execution. Given this assumption, the dynamic binary analysis is undoubtedly the best choice to study the behavior of the software and create models of normality. Dynamic binary analysis can help to understand how a program works in a minimal amount of time. Changes in the system could be tracked at different levels. Dynamic analysis at function call level of abstraction applied to security has been successfully analyzed in the past(35, 36).

3.2.1 Dynamic analysis

Dynamic analysis is focused on the study of the behavior of an application while it is being executed. Dynamic Instrumentation can greatly help to find valuable patterns and it does not need a binary to be recompiled since it includes the ability of inserting code into a program in order to obtain runtime information. Such information can be valuable to further generate behavior models. Although many dynamic instrumentation tools can be found, next, three of the most widely used are detailed in order to show different procedures:

- **Valgrind** is a framework focused on Linux/Unix systems. It uses dynamic recompilation techniques (also called disassemble-and-resynthesise) to provide Just in time compilation. Once the application is running, Valgrind translates the instructions into a temporary structure called Intermediate Representation. Code transformation can then be applied before it is translated again in order to be run in the host processor.

- **Pin API** was originally developed by Intel. Pins instrumentation is based on copy and annotation techniques, that is, each instruction is copied and tagged with its description. The framework allows users to generate their own instrumentation tools.

- Finally, **gdb** is one of the most well known debugging tools. Written by Richard Stallman, gdb allows tracing and monitoring function calls during execution. One of the mot interesting features is the ability to remotely debug an application, that is gdb may run in one machine while the debugged program run in an embedded system.

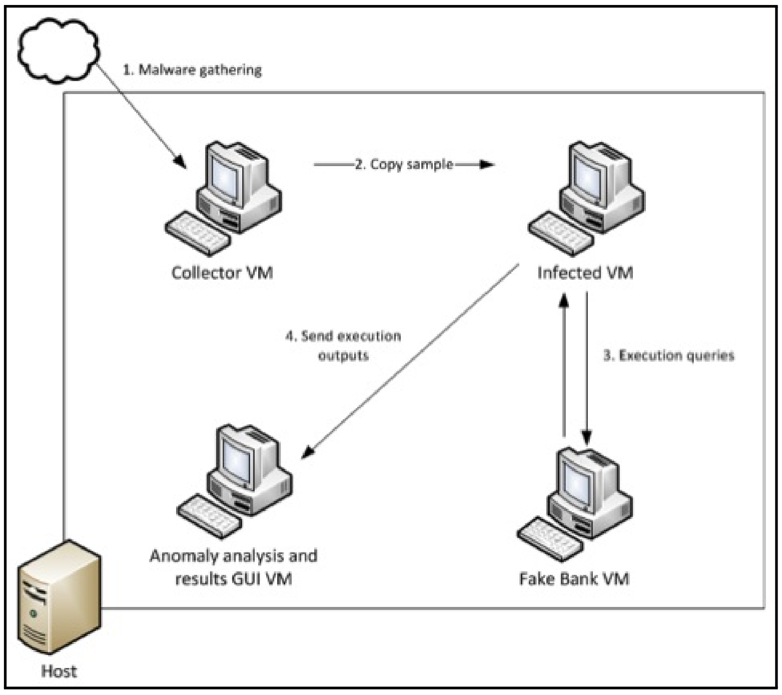
The fastest way of getting an idea of how a suspicious application behaves is to execute it and observe changes in the environment. The following application can greatly help to gather information on the execution of a Windows based application:

- **RegShot**. This application allows to take an snapshot of the registry in order to compare the state of the system before and after a malware is installed.
- **SysAnalyzer**. This tool allows security experts to study the behavior of an executable providing information regarding running processes, open ports, loaded dlls, drivers and possible API injections.

3.2.2 Virtual environment

Executing malicious applications within a virtual environment such VMware, Paralels or VirtualBox provides great advantages such as the easiness to revert the snapshot of the operating system to the normal state in order to repeat experiments. A virtual environment was prepared for the analysis of suspicious binaries. Due to the experimental origin of the tests, possible detection of virtualized environments was omitted. The tests were performed within 4 virtual machines (VMs): (1) A collector module, (2) an execution VM, where the instrumented browser is first executed in a clean environment and later infected for evaluation purposes, (3) an evaluation VM where the models are generated and evaluated and (4) a web server where all the processed results are accessible. The virtual machines where configured within a Intel Core i3-370M Server with 4 GB of memory. Figure 3.2 shows the design of the architecture that also includes a simulated bank server for specific tests. Although the research described in Chapter 5 involves a first model based in dynamic intrumentation, the final architecture is detailed in Chapter 6.

Figure 3.2: Training environment



Part II

Main Contributions

4

Server Side Detection approaches

Human digital behavior does not change much from human behavior in the real world. Most internet users acquire certain habits when browsing sites recurrently such as social networks or online newspapers(37). Most actions performed by users with a browser are stored in web servers. Thus, the study of webservers logs can provide a clear idea of customers' habits. Web mining techniques have been widely used for e-commerce purposes such as generating user profiles in order to deploy specific advertisements within a site. Our hypothesis is that the use of stolen credentials or transaction redirections within online banking sessions generates new data in the web server logs that should be detected in the form of outliers. According to the bank that participates in the research, given a query, anomaly detectors must provide a decision in less than 0.3 seconds in order not to affect the user experience. However, three main problems arise when dealing with anomaly detection: (1) high percentage of false positives, (2) computational limitations that can arise when maintaining a behavior model for each user and (3) the dependence on the training data.

This chapter focuses on the logs generated by bank servers for studying user behavior in order to generate models capable of arising outliers in case a malicious action is performed. Section 1 describes the threats to be detected and the preprocessed data that will be used. Boths parameters provide a clear idea of the type of algorithms to be applied. Section 2 focuses on the deployment of models related with Gaussian distributions. Although some variables extracted from weblogs can be modeled through low computational algorithms, other aspects such as navigation or relationship among users require specific preprocess and modeling techniques. Section 3 will detail different approaches related to more complex models based on different data mining techniques such as clustering, Market Basket Analysis (MBA) or Markov chains. While section 4 describes the prototype deployed to perform the experiments, section 5 will present the results obtained. Finally, some possible improvements are described in section 6.

4.1 Threats

The knowledge on the type of threats is a key aspect to knowing the state of a possible attack. In this regard, given the possible attacks perpetrated upon customers, four possible types of threats have been defined.

- **Robotization**. We refer to robotization as the automated crawling of an authenticated site. It is often performed in order to collect all the information from the victim and prepare the most convenient fraud (38). Before robotization is done, a previous credential theft through phishing, pharming, MitB, MitM, keystroke based trojans or other meanings is necessary. The information gathered from the crawling can also be sold in the underground market.

- **Redirection**. Redirection attacks allow hackers to change part of a query during online banking sessions. Such actions involve changes on the URL, the destination amount, or the destination account thus changing the normal behavior of the user. This attack is mainly performed by trojans that include MitB techniques.

- **Impersonation**. In this case, impersonation threat occurs when the fraudster logs into the system with the previously stolen user credentials and makes a transaction. In this case, second possible authentication information must also be in possession of the attacker.

- **Transaction**. Transaction risk occurs when money is transferred to an unusual destination. An impersonation or redirection anomaly can end up with a transaction anomaly. However it is interesting to have this anomaly separated from the rest because money theft is the most important feature that bank must discover. Each threat can be detected through a number of sensors or detectors that will collect individual anomalies from different variables.

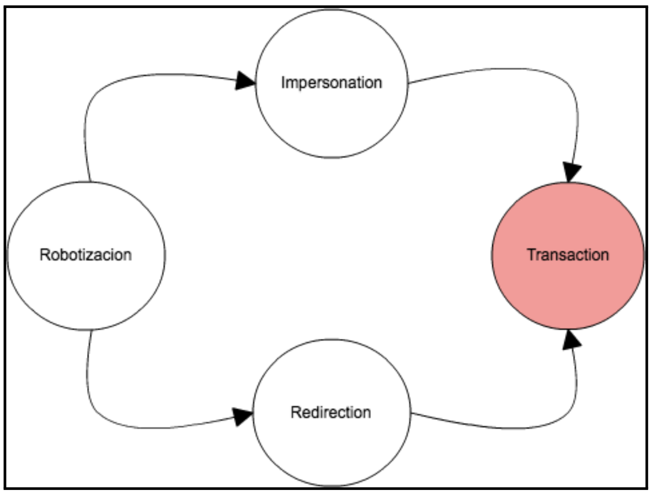


Figure 4.1: Anomaly relations

According to Chandola et al. (39) four characteristics must be analyzed when choosing an anomaly detection technique: (1) Nature of data, (2) labels, (3) anomaly type and (4) output .

- **Nature of Data**. A key aspect of anomaly detection is the type of attributes or features of the data set. Although originally the data set was based on a list of observations, due to the relations among observations, a graph-based environment was deployed. The most interesting node and edge attributes, in terms of model sources, were both categorical and continuous so different approaches were needed. However, clickstream, which is based on a sequence of queries, needs a different data manipulation in order to be a valid source for experimentation. Table 4.1 shows the type of variables finally extracted from the preprocessing phase.

- **Label**. The presence of a label can determine if the detector will include supervised (the set contained labels identifying normal and abnormal observations), semi-supervised (all the observations are labeled as normal) or unsupervised (there are no labels included) machine learning techniques. Since the data set A is used for training, semi-supervised anomaly detection techniques should be applied.

- **Type of anomaly**. Anomalies can be classified into three types depending on

Feature	Nature of data	Source	Type of anomaly
userid	categorical	raw logs	Point
time of the day	continuous	raw logs	Point
session time	continuous	raw logs	Point
IP-country	categorical	country-preprocess	Point
Site Language	categorical	raw logs	Point
Browser language	categorical	raw logs	Point
Clickstream [a]	categorical	session-preprocess	Collective

[a] It includes several sequencial observations

Table 4.1: Variable types

their nature: (1) Point anomalies. In some environments (such as electrocardiograms) an anomaly can appear only when a group of instances are anomalous with respect to the rest of the observations. Each observation of the data set can generate an anomaly (2) Collective anomalies. In this case, multiples outliers generated from the same model raise an anomaly. Although many traffic anomaly detectors can rely on collective anomaly algorithms to detect attacks such DDoS, in our case, a threat scoring will be based on the sum of the scorings gathered from anomalies belonging to different models. In the case of clickstream anomalies, the result of the analysis depends on a set of queries included in different queries. Therefore, a subset of observations are needed to generate an output.

- **Output**. The output of the detectors can be of two types: (1) scoring where each observation will be given a number depending on the degree of the considered anomaly and (2) label, where a label (normal or anomalous) is applied to each instance. Due to the need of real-time detection, if a scoring output is provided, a threshold must be applied to obtain an automatic decision.

Finally, given the type of the information to be analyzed two important features must be decided regarding the model deployment:

- **Individual or global models**. Ideally, all users should have their own models. However, the training time required to generate a model can, in some cases, penalize the detector's hit ratio if an attack is produced during the first sessions. As described before, performance must also be taken in to account. Although a query can be compared to the rule and gaussian based models, more complex information can generate problems in terms of memory consumption.

- **Anomaly calculation time**. Each anomaly calculation is performed when an inclusion of a new node modifies the last calculation. For example, a model based on the time of the day when the sessions are performed, is calculated only when a user starts a new session and a session node is created; however, a clickstream based model will be calculated every time a new operation is performed within the session.

Our approach combines specification-based and anomaly-based detection by defining a set of models based on data mining techniques and the knowledge of how an attack may affect the normal behavior. It is expected to calculate separated models to later define each model weight parametrically for each anomaly. Next sections detail how some of the models can be directly calculated from the first preprocess detailed last chapter, while more complex approaches will need a second preprocess to generate the needed features.

4.2 Initial models

Due to the features of some of the variables and the need for low computational cost comparisons, parametric statistical models were chosen. The central limit theory states that when the results of an experiment are due to a big amount of independent causes it is expected that the results will follow a normal distribution. Many continuous variables may be represented by a density function, f. The area of such function, that is the distribution function, will provide the necessary probabilities to compute the anomaly. The distribution probability of a variable determines the probability of a success to be accomplished. The probability distribution normally clusters on a mean value. The deviation σ is the measure of the width of the distribution. Each observation is compared to thresholds based on the mean μ and the deviation σ . In order to quantify the anomaly, Z-score, that is the number of deviations an observation is above or below the mean, is calculated.

$$z = \frac{x-\mu}{\sigma}$$

An observation is considered anomalous when it exceeds the area compressed between the mean minus the standard deviation and the mean plus the standard deviation. In order to understand the data set an initial study of the information was performed.

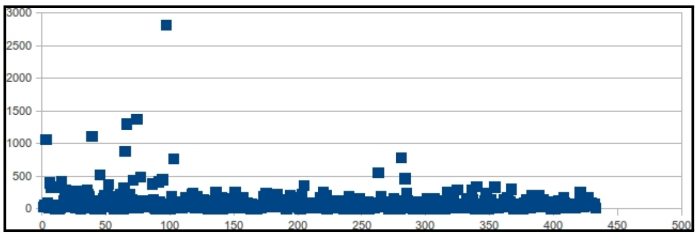


Figure 4.2: Number of sessions per user

A quick analysis of the logs allowed us to gather the necessary information to know the state of the session based on the query structure. Next, the following initial approaches are described: (1) Number of sessions, (2) Time spent during the session, (3) time-of-the-day and (4) location-language.

4.2.1 Number of sessions

A high level of sessions per day can provide information on multiple transaction authentication attempts. This model is not meant to detect brute-force attacks over the first authentication method (normally user-password) because network monitoring systems can easily detect such anomaly through Netflow data (40). However, generally transaction operations require a second authentication process. In cases where the hacker does not to have such credentials or has acquired only part of the information (eg. part of the PIN card) random numbers are inserted. In case the second authentication fails a number of times, the fraudster is logged-out but new logins are allowed.

Based on specific OM-OD value (see Chapter 3), we could identify the beginning of each authenticated session. Figure 4.2 shows that some users generated a high number of sessions. The bank confirmed that such users belonged to companies making hundreds of transactions daily. Since the behavior of such users changes with respect to home-user behavior, they were discarded. After the filter, 47.692 sessions were found, that is, a mean of 110,14 sessions per user. As showed in 4.4, if the results are calculated as the log of the number of abnormal transitions, a log-normal distribution is presented. Given the dates of the first and last session, each user initiated an average of 4 sessions each 100 hours, that is one session daily.

A first histogram was generated without the presence of significant outliers.

The dataset was modified in order to obtain the logarithm for the number of sessions.

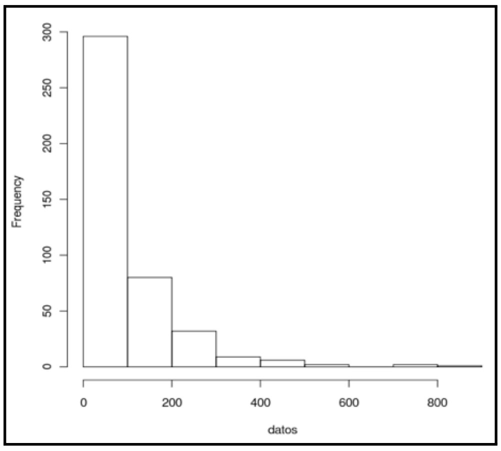


Figure 4.3: Amount of sessions

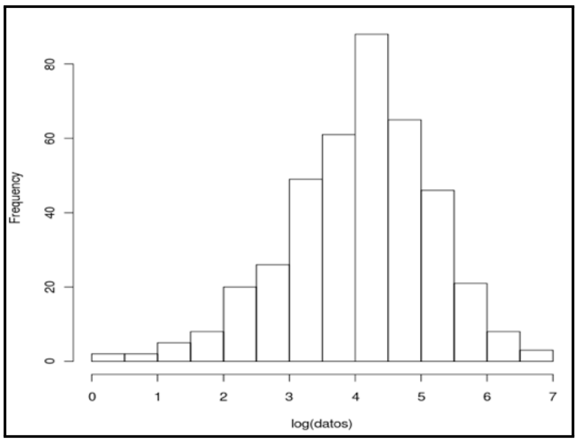


Figure 4.4: Histogram

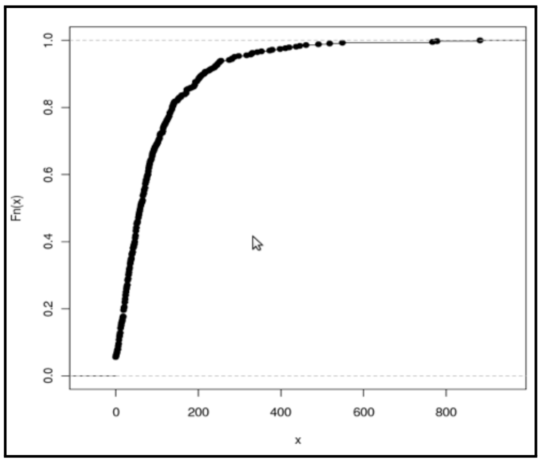


Figure 4.5: Empirical distribution function

Figure 4.5 shows the empirical cumulative distribution function. A clear relationship between the number of sessions and a log-normal distribution graph is noticed.

Each user will have a gaussian distribution generated during the training phase. While the model information (mean and deviation) is stored in the user node, the scoring generated after the beginning of each session is stored in each session node.

4.2.2 Time of the day

Stolen credentials are usually sold through the Internet to fraudsters living in countries different of the victims. The time of the day of connections made by criminals may differ significantly from the normal victim behavior. A per-user analysis of the training data set showed that 83% of the sessions were performed within a range of 5 hours. Similarly to sessions-per-day models, gaussian models based on the time of the day could be of interest in order to detect sessions generated by fraudsters in abnormal hours. In this case every single user must have an individual time-of-the-day model based on his sessions. Time of the day model requires some changes in the data set due to the time changes that can appear depending on the origin country were the connection is made.

When a user logs into the system a new session node will be created. The attribute "time" will be the time in the origin country of the connection. This way, it allows us to track the users' behavior with respect to the time when they connect. In order to generate the model, all attributes “time” from the the session nodes are used to calculate a gaussian distribution.

4.2.3 Time spent

Automatic logins made by trojans with stolen credentials can produce a number of queries in a short period of time. The calculations made from data set A showed a mean session time of 62 seconds. Anomalies based on a very low number of seconds can provide information on possible crawling on customer data using stolen credentials. However, the high value of the variance $\sigma = 40$ shows the difference among users; thus a model is generated for each user with the same characteristics of the rest gaussian models. Unlike the above models, the time-spent model is not updated until a number of queries have been performed. The threshold is defined by the mean of the queries per historical session and is calculated upon the start of each session.

4.2.4 Transaction amount

Transaction amount can also provide information on possible frauds. Although mule transactions do not involve high amount of money per victim, many users only use online banking to perform transactions with different quantities. An individual model on the transaction amount can greatly help to detect undesired transfers. In the case of a transaction id based on sending money from one account to another, an edge with the “amount” attribute connecting source and destination accounts is created. However, if the transaction is a virtual card or mobile charge, the value will be included in the specific operation node.

4.3 Complex models

The following models correspond to new approaches that add new information to the detector in order to better asses the risk of each session.

4.3.1 Languages

The use of three variables can help to detect such actions:(1) Origin country official languages (OL), (2) site languages (SL) and (3) browser language (BL). Although the IP could determine the approximate location of the connection, this variable itself should

	BL=SL=OL	BLSL=OL	BL=SLOL	BLSLOL
BL=OL	normal	abnormal	abnormal	abnormal
BLOL	normal	normal	normal	abnormal

Table 4.2: Language combination anomalies

not generate an alarm given the increasing mobility of customers. However, many users connecting from other countries use their own laptop and browser. The country of origin of the session and the official languages can be a great tool to detect sessions initiated with stolen credentials. The first sessions allow the detector to recognize the country from where the user normally connects. A list of official languages spoken in every country is included in the system. A separate graph including relations between countries and languages was deployed. When a connection is initiated, an edge between the session node and the country is established. A first analysis of the training data, along with interviews with the bank confirmed as expected the following:

- The site language does not generally change; only 1% of the sessions from data set A.
- Unfortunately, user-agent information, where the browser language is included, is only present in 25% of the sessions.
- 90% of sessions where initiated from a country were one of the official languages was the same as the browser and the site.
- Recent have experienced an increasing use of public computers which implies a higher number of queries where $BL = OL$. However, there is also a high risk of working on an infected PC.

Given the different combinations, a set of rules, applied only to connections with user-agent, were established. The final rules are showed in table 4.2.

4.3.2 Clickstream analysis

Navigation profiles can determine if a session has been initiated by an authorized customer. In the case that the sequence of queries made during a banking session is not similar to normal navigation an anomaly should occur. This approach would help to detect the use of stolen credentials used by cybercriminals. Although it has been found that some trojans are capable of learning behaviors of an infected user in order to reproduce similar actions, this technique is not widespread. As seen before, some banking trojans crawl the authenticated site and perform specific fraudulent actions. Similar normal sessions can be classified depending on their interactions with a web site. Clustering deals with processing groups whose members are similar in some way. However two problems appear:

- Clustering algorithms have been developed for continuous data (41, 42, 43, 44). Hierarchical clustering allows us to visually analyze different groups from a data set and does not assume an initial number of clusters. This type of clustering connects objects to form clusters based on their distance. Thus, most hierarchical clustering algorithms need a distance matrix that represents how similar all observations to each other. In this case, all sequences of queries must be compared. The final number of clusters can be decided by "cutting" a dendogram, that is, a tree like diagram.
- Once, the cluster is generated, it is needed to compare a given session with the general profiles every time a new query is performed. K-order Markov models can be extracted out of each cluster to better extract normality models for each general profile. K-order Markov techniques are useful to predict the next page to be visited from the information regarding the last k pages visited before.

4.3.2.1 Sequence alignment

One of the challenges to be faced was to compare web sessions with different number of queries in order to generate the distance matrix. Although some interesting approaches have been investigated regarding the use of categorical variables with continuous based algorithms, sequences alignment distances seems to be an interesting way to solve the problem (18). Comparing sequences of objects of different lengths through sequence alignment has been an area of study in the bioinformatic field to align proteins or nucleotids. Sequence alignment is a method to identify regions of similarity within sequences based on adding gaps and changing certain elements to align equal columns of characters. The technique involves the generation of a final score that represents the similarity among two sequences. The score is based on the following elements:

- **Substitution Matrix**. This matrix represents the penalty of "changing" one character of a sequence to another character in order to obtain identical sequences. Our substitution matrix was based on providing a high penalty between pages of different sections.

- **Gap penalty**. In some cases, instead of applying the penalty to many queries in a certain regions on the sequence, it is possible to add a virtual gap that increases the similarity of the rest of the sequence. Such gap also involves a penalty. The algorithms must choose the best option in each case in order to reduce the final result.

Two sequence alignment techniques can be applied:

- **Global Alignment**: This type of alignment is often used to compare two sequences with similar length. Needleman-Wunch algorithm is the most popular technique.

- **Local Alignment**: This type of alignment is performed with sequences with a high number of dissimilar sub-sequences. In this case Smith-Waterman algorithm is the commonly used method.

4.3.2.2 Initial clustering test

Our assumption is that, while normal observations should belong to dense clusters, anomalies should belong to small clusters. A dendogram based on a hierarchical cluster was used to check the effectiveness of the algorithms. Substitution matrix in bioinformatics would involve only four elements (G, T, C and A); however, the number of elements to be compared in our case are all the possible queries, that is, more than 2.000. Due to performance problems, a selection of the most common queries is done through a frequent item set algorithm (45). The data set included 136 sessions, 6 of which contained abnormal navigations. The hypothesis is that one of the clusters would include those sessions different from the rest. The complexity of the training phase in terms of computational power depends on the classification algorithm. Jaligner (46), a Smith-Waterman implementation showed the best results. Figure 4.6 shows a dendogram with four representative groups. The coloured group included 7 elements, which 5 were anomalies, generating a hit radio of 83,3%. However, 2 elements from

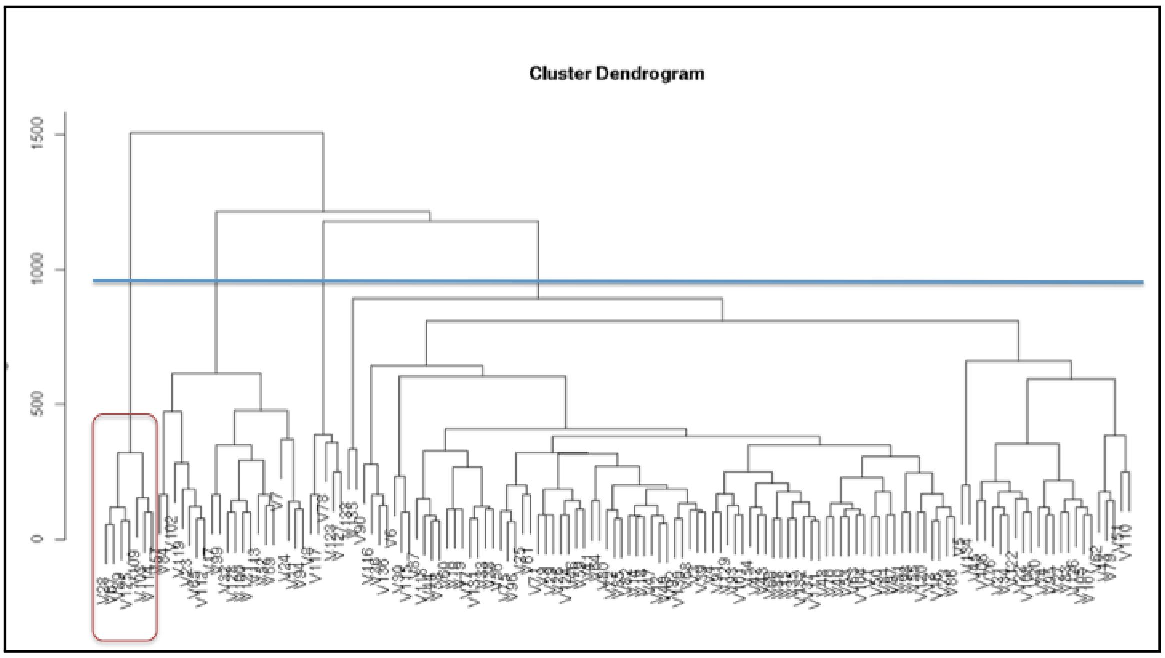


Figure 4.6: Dendogram diagram of the hierarchical clustering extracted from a data set containing clickstream anomalies. The surrounded group contains most of the anomalies.

the coloured group were not anomalies, generating 1,4% of false positives. Finally, 2 elements anomalies found in the rest of the clusters caused 2,2% of false negatives.

The technique presented could be useful for detecting anomalies from a historical data set. However, since the goal is to detect anomalies in real-time, a new clustering approach that will lead to Markov chains models will be detailed next.

4.3.2.3 General Profile Clusters

Due to the number of users that connect to financial institution's sites it is not feasible to maintain individual user profiles. The growth on the number of customers would have an important impact on storage and performance. Groups of normal customers allow to create global models that represent similar behavior. However, comparison of a new session with a cluster is computationally expensive.

Our approach is based on generating a clustering with regular sessions in order to obtain the different profiles of normal use by users. A Markov model is generated for each group and compared. Every new session is compared with each of the models. If none of the comparisons provides a close similarity an anomaly is arisen.

4.3.2.4 Clustering and Markov model

A hybrid technique of clustering and N-order Markov modeling was applied. A Markov model is defined by the tuple of elements $\lambda = (S, V, A, B, \Pi)$:

A vocabulary $V = P$ based on M possible queries to pages of a financial institution site:

$$V = P = \{p_r, 1 \leq r \leq M\}$$

A group of N states, $S = \{s_1, s_2, ..., s_N\}$. An state is considered as any sequence of k queries. thus,

$$N = M^k.$$

A equals to the transition probabilities among states,

$$A = \{a_{ij}, 1 \leq i, j \leq N\}$$

B equals to the probabilities to get specific query any state,

$$B = \{b_{ir}, 1 \leq i \leq N, 1 \leq r \leq M\}$$

where b_{ir} is the probability of getting page p_r when leaving state s. The states and vocabulary will be obtained during the training process. Training is based on counting the times where a state is visited and the performed transitions. The evaluation of a query to a page within a session is based on:

$$W = \{normal, if AS(W) \leq \theta, abnormal, otherwise\}.$$

where Θ is a threshold that must be calculated experimentally. Three types of sessions were established based on their highest similarity to any of the four model: type 1 sessions are those where $AS(W) >= 10^-3$ that is sessions with clickstreams (sequences of queries) closer to the model; type 2 sessions where $10^-3 > AS(W) >= 10^-5$; finally type 3 sessions are those where $AS(W) < 10^-5$. $AS(W)$ was determined parametrically. Ideally a clean session would be tagged as Type 1 or 2 while a fraudulent session should fall into Type 3 category.

4.3.3 URL order

Most queries performed to dynamic sites are based on a domain followed by a set of tuple of variables and values separated by "&". As shown before, many trojans can be programmed to perform activities such as (1) changing some query elements in real-time during a valid session or initiate their own session with stolen credentials to perform different actions. In the first case, a URL must be changed in terms of the variables' values (such as destination account or amount). In the case a trojan logs into the system, a sequence of URLs, such as 1. login, 2. fill the transaction form, 3. validate the transaction and 4. log out, must be programmed and any error in the number or order of variables should trigger an alert. It was found that, due to programming errors, many manually crafted URLs do not respect normal order of the variables within a URL.

Our goal is to find an ordered list of variables for every type of page queried (identified by OM-OD). However, although the order did not change, a first analysis of the data set A revealed that a single page may contain different variables, thus increasing the number of results. Given each page, it was necessary to find the usual combination of variables that are involved.

MBA , widely used by retailers, allows them to know which products are purchased together. In our case, each variable of a query was a "product" and each query was a "basket". The goal is to obtain the most frequent group of variables that appear in each webpage, that is, along with every tuple OM-OD. Apriori (47), a well known MBA algorithm , was used to determine the most common combinations of variables. All combinations appearing at least 0.0001The hypothesis is that, given a new query, if the type or the order of variables included is not within the list, it is considered anomalous. Once the most frequent groups of variables were found for every site, a final list with the ordered variables was generated.

4.3.4 Familiarity model

The term familiarity is here used to determine the level of relationship between a customer source account and the destination of a transaction made from his bank account. Following the research of Watts et al.(48), it is expected to find relationships among customers accounts based on the transfers made among them. Given a graph including

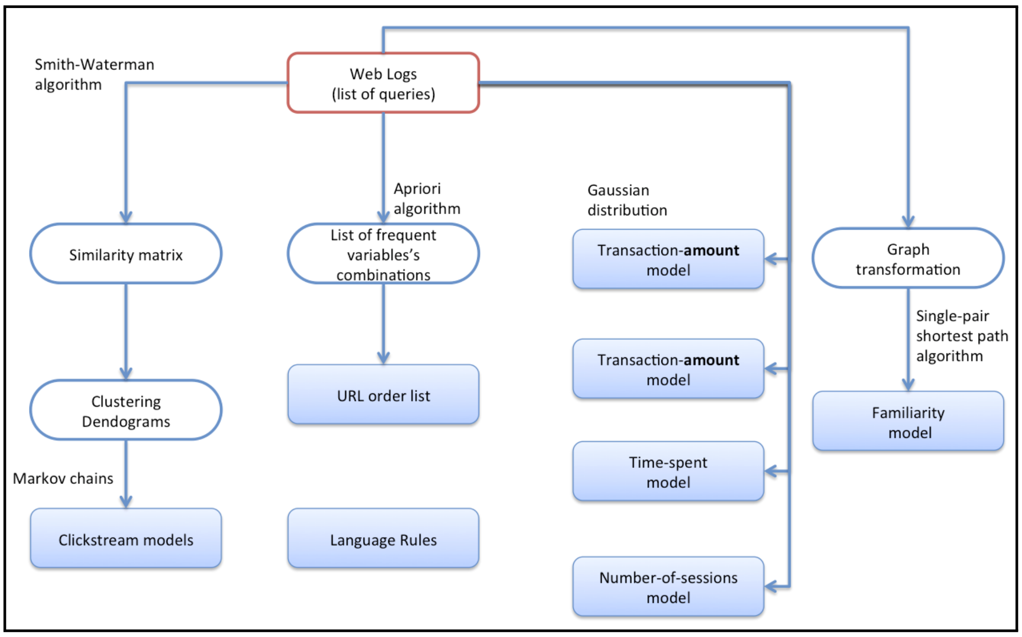


Figure 4.7: General detection scheme

transfers, where nodes are the accounts and the edges are the transactions between accounts it could be possible to determine if a transaction is anomalous based on the number of steps needed to get from the source account to the destination account. The calculations are based on single-pair shortest path in direct graphs, already implemented in DEX. Some exceptions should also be made in order to circumvent account nodes related to very frequent transfers such invoice payments to big companies. Otherwise familiarity among two customers hiring the same ISP would generate a false familiarity result.

4.4 Prototype

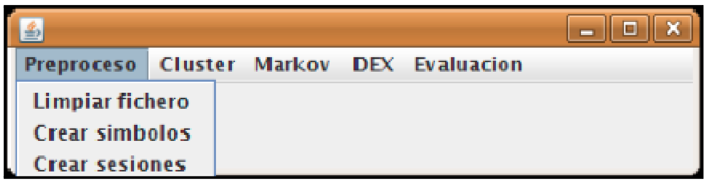
Each model has been given a maximum weight for all possible threats. The weights were decided according to the experts experience and the importance of each anomaly for every threat. Table 4.3 shows both the values of each detector.

A threat is only analyzed when all the related models can be calculated and will trigger an alarm if the final value exceeds z-score in 1. A prototype (see figure 4.7) based on two different applications was developed:

Anomalies	Robotization	Redirection	Impersonation	Transaction
Number of sessions	-	-	22,2%	-
Time spent	40,0%	-	22,2%	-
Time of the day	-	-	11,1%	-
Transaction amount	-	-	-	33,3%
Languages	20,0%	-	11,1%	8,4%
Clickstream	20,0%	-	5,6%	8,4%
URL Order	20,0%	33,0%	22,2%	33,3%
Familiarity	-	66,0%	5,6%	16,6%

Table 4.3: Threat Scoring

Figure 4.8: TexDEX menu



- **TexDEX** allows experts to generate all the global models that need training from a given dataset. The application performs the preprocessing phase and generates the most frequent URLs for the order model and a dendogram. The expert can than choose the numbers of groups in order for the system to generate the different Markov models and add them to a DEX graph. Figure 4.8 shows the application menu.

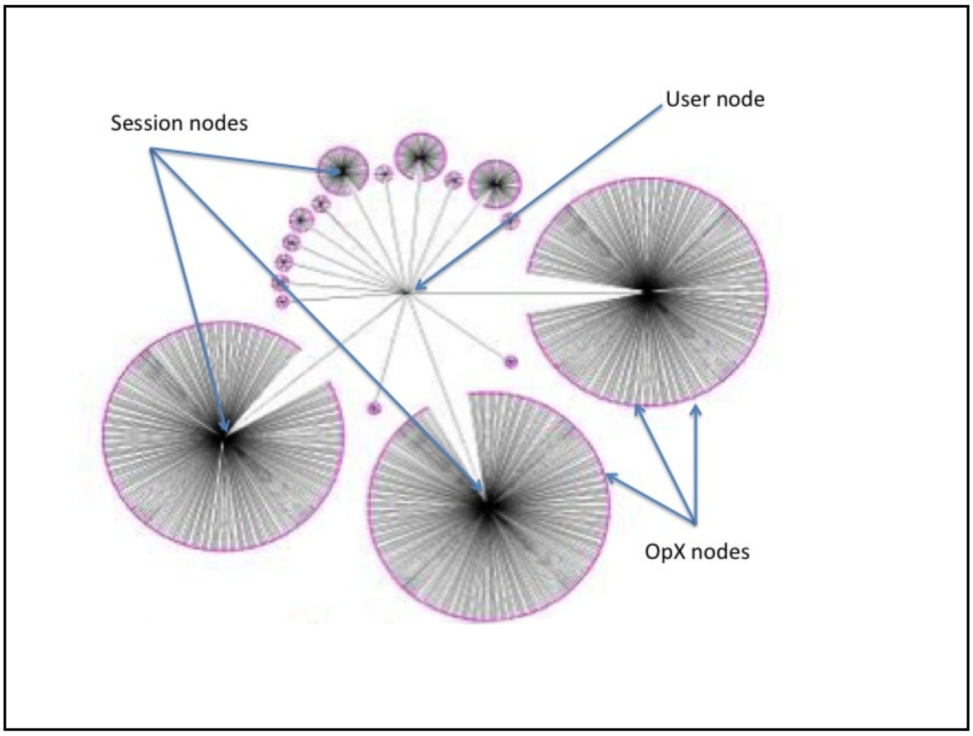
- **WADS** receives information from the weblogs, performs the preprocess and moves every query to the graph. Once certain elements are included (see 4.4), an anomaly calculation is made and the threats' scoring are logged.

  The application also allows to save the graph in order to make visual analysis through graphviz format or ygraphml(see figure 4.9).

Anomalies	Type	Technique	Location	Calc. moment
Number of sessions	Global	Gaussian	file	session node
Time-spent	individual	Gaussian	session node	operation node
Time-of-the-day	individual	Gaussian	session node	session node
Transaction-amount	individual	Gaussian	transaction edge	transaction edge
Language Rules	individual	rule based	session node	session node
Clickstream	global	clustering/Markov	operation node	any node
URL Order	global	MBA	file	Op node
Familiarity	individual	closest path	transaction edge	transaction edge

Table 4.4: Anomalies deployment

Figure 4.9: Visualization sample of a user with session and operation nodes



Markov model results	Type 1[a]	Type 2[a]	Type 3[a]
Clean sessions	61,00%	32,00%	07,00%
Mailicious sessions	16,67%	33,33%	50,00%

[a] Type 1: sessions where $AS(W) <= 10^{-3}$
[b] Type 2: sessions where $10^{-3} < AS(W) <= 10^{-5}$
[c] Type 3: sessions where $AS(W) > 10^{-5}$

Table 4.5: Markov model results

4.5 Experimentation

4.5.1 Training

In order to generate the individual models, a minimum number of 10 sessions and 10 transactions per user was chosen. Given that not all users had logged in that number of times, a subset of 200 users (2.095 queries) that fullfilled the requirement was chosen. All the queries were included in a DEX graph thus generating and storing the values of the gaussian based models. Rule based models did not need any training. Regarding the Markov models, WADS application was used to generate the models and integrate them with the dataset graph. The dendogram of general profiles was divided into eight groups. Once the navigation groups were defined, a Markov model was defined for each group.

4.5.2 Test

A first test with synthetic data was performed to validate the Markov chains approach. The results detailed in Table 4.5 show Markov models based on clean customers click-streams can provide useful information regarding possible anomalous sessions in real-time. However, malware sessions within the similarity 1 group show that false positives could appear if Markov models evaluation is only applied. The use of MitM techniques, where there is no interaction of those other than the customer and his computer, is probably the reason no anomalies were found in one case.

The final test involving all the detection methods was based on the queries from the 200 users extracted from data set B, that is 19.478 queries. The bank confirmed that 5 of the selected queries still belonged to malicious sessions. The results showed that although many queries raised anomalies, few of them (25, that is 1,28%) generated a threat z-score above 1. As shown in 4.6 the results confirmed that all 5 malicious

Anomaly	Type	Robotization	Redirection	Impersonation	Transaction
Anomaly 1	Information theft	X	-	X	-
Anomaly 2	Information theft	X	-	X	-
Anomaly 3	Zeus(MitB) Transaction redirection	-	X	-	X
Anomaly 4	Transaction with stolen cred.	-	-	X	X
Anomaly 5	Transaction with stolen cred.	X	-	-	X

Table 4.6: Anomalies Test results

actions raised at least one threat alarm. As stated before it was expected that a malicious action could generate more than one high threat score. Both information theft anomalies generated Robotization and Impersonation threat alarms due to high values on time-spend and URL-order comparisons.

Transactions made with previously stolen credentials raised different alarms possibly due to different attack vectors. A Robotization alarm informs that anomaly 5 was probably performed automatically through a program; however, the low values of URL-order and time-spent detectors show that anomaly 4 was probably generated by a manually performed session. Anomaly 3 and anomaly 5 correctly raised an transaction threat alarm. Both sessions were used to perform a money transfer abroad. Zeus was detected because of the anomalous results provided by familiarity and URL-order detectors. Unfortunately, most of the false positives were generated due to the low number of detectors used to analyze the redirection threat.

4.6 Possible improvements: Page Rank

The anomaly scores calculated from new sessions help to determine the presence of a threat. However, information on old anomalies can help to detect suspicious actions due to the existence of a relationship between some threats (e.g. a Eobotization can lead to Impersonation). Thus a Robotization anomaly should alert the experts of a possible fraudulent transaction in the future. It is expected that the detector will provide real-time information on the type of anomaly and the attack phase. A Page

Rank like algorithm could manage a historical level of suspiciousness per account node. If a node is involved in an anomalous clickstream session, a suspiciousness score should be assigned. The level of suspiciousness can increase if the account is targeted by other suspicious accounts, that is, a virus like spread occurs. The algorithm would also solve an important problem: The management of historical anomaly scorings. Although the techniques detailed showed good results, financial entities and antivirus companies need to know which samples can affect their customer. The next chapters will focus on the analysis of banking malware samples from another perspective: the client side. This approach will help to improve the analysis of redirection based threats.

5

Browser Helper Objects

This chapter will allow us to introduce dynamic instrumetation in our research and move from Server based analysis to Client side analysis. We will focus on the static and behavioral analysis of BHO to generate rules that may help to detect BHO malicious actions through dynamic instrumentation of a browser within an isolated environment.

BHOs are defined by Microsoft MSDN as “straightforward way to customize Internet Explorer”(49). Since Internet Explorer 4.0 began to support BHO, this component has been widely used to perpetrate numerous frauds by injecting HTML code and monitoring victims’ requests. New BHO based malware samples keep appearing in the wild every day causing losses to financial entities. Moreover, some BHO have the ability to hide from detectors preventing antivirus from taking countermeasures. The following malware samples give an idea of the attack types and behavior of malicious BHO:

- KillAV. KillAV samples, based in Browser Helper Objects, first appeared in June 2010 and the origin of the family is China. Apart from stealing credentials, the malware is capable of disabling the user access to antivirus and security forum webpages.

- Banker. Detected in April 2009,Banker drops a DLL as a BHO upon installation. It includes keylogging and mouse movement logging capabilites.

- Banbra. Banbra family, appeared back in 2008, was programed in Delphi and normally packed by Yoda Protector. The malware generally arrives to the computer through a picture. This trojans target mostly Brazilian entities, as well as

Bancos. Some variants of this family steal credentials by activating the automatic password storage option of the browser.

5.1 Browser Helper Objects

Win32 processes run in their own address space. Customizing the behavior of an application implies, in some cases, breaking the process limits. BHOs allow developers to write Component Object Model (COM) components that are loaded each time Intenet Explorer starts up. One of the most important features is that BHOs run in the same memory space as the browser so they have the capability tp directly interact with the browser actions. were introduced in October 1997 with the release of version 4 of Internet Explorer. BHO API creates hooks that allow access to Document Object Model (DOM) of a web page. When a browser loads a new page, a set of events are generated and published. DOM allows BHO to control such events and perform different activities. As an example, Acrobat may launch a BHO plugin that allows the browser to show pdf contents.

Since BHOs are Component Objects, they are registered under a registry's key. Whenever the browser starts, the following registry key is checked:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion
\Explorer\Browser Helper Objects.
```

If such key is found, a Class ID (CLSID) below the key is examined and the loadable BHOs are extracted.

The Universally Unique Identifier (UUID) standard , approved by the Open Software Foundation (OSF) specifies how to provide unique identifiers to information without a central coordination. Globally unique identifiers GUID are the implementations made my Microsoft. Since the generated identifier is a 2^{128} number the probability of having two equal identifiers is extremely low. GUIDs are used by Microsoft, among others, to specify certain classes and interfaces of COM objects.

A CLSID is an implementation of the UUID standard that represents a unique identification for each installed BHO and is stored in

```
HKEY\CLASSES\ROOT\CLSID.
```

An interface consists of a pointer to a virtual function table that contains a list of pointers to the functions that implement the functions declared in the interface. Each COM component exposes its functionality through one or more interfaces. All COM interfaces are derived from IUnknown.

Most malware samples will use two faces to interact with the browser:

1. IWebBrowser2. IWebBrowser2, exposes methods implemented by Microsoft Active X or OLE Automation such as get_LocationURL or get_Document. Such methods can be used to retrieve URL information. The interface can be used to perform different malicious actions since it allows monitoring of the following events:

    - DocumentComplete. According to MSDN, "the event fires when a document is loaded and initialized". This information could be used to alter the structure of the document in order to a new form. Such a technique is applied for gathering information such us those related to second factor authentication data.

    - BeforeNavigate2. This event arises when a query is going to be sent. The control over the information provided by the event can allow criminals to collect credential information.

2. DWebBrowserEvents2. This interface is needed in order to get event's notificacions from a WebBrowser control or from the Windows Internet Explorer application.

Eventhandlers establish the actions to be executed when an event appears. The monitorization of eventhandlers can provide information on possible attacks. However, eventhandlers' implementation may vary depending on the development environment. Although a BHO can be programmed entirely, it is generally developed through one of the following libraries:

- Microsoft Foundation Class Library (MCF) is a library provided by Microsoft through Visual Studio. It allows to create Active X controls and applications. MCF provides an easy access to Windows's APIs.

- Active Template Library (ATL) library leaves the programmer with the job of only implementing one method and the events handlers.

Depending on the library, different function calls can be extracted given an isolated execution of an application. Invoke() is used each time an event is generated. Such a method is in charge of deciding if, given an event, an action must be performed. In the case of ATL based samples, _ATL_EVENT_ENTRY should be checked to analise the malware behavior.

5.2 Traditional BHO detection approaches

Some anti-malware tools include databases that offer lists of known malware CLSIDs which are used by some signature-based detectors. David Gloser developed Remote Bho Scanner (50), an application capable of performing a network checking over Windows OS within a domain. The application compares the found CLSID against the CastleCops BHO database, that maintained list of non malicious until 2008. However, the morphing properties of some samples help them to avoid detection.

Microsoft increased the security of BHOs when launching Internet Explorer 6. The new version included Add-on Manager, which is able to list and manage both BHO and ActiveX controls.

5.3 Detection evasion

Parvez Anwar detailed how a malware can hide itself from Add-on Manager (51). Generally, the length of an CLSID is 16 bytes. As an example, Adobe adds the following key in the registry to register its BHO in such a way that when explorer starts up it knows which BHO are needed:

```
[HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer
\Browser Helper Objects]
{18DF081C-E8AD-4283-A596-FA578C2EBDC3}
```

It also informs Internet Explorer to use AcroIEHelperShim.dll:

```
[HKLM\SOFTWARE\Classes\CLSID\
{18DF081C-E8AD-4283-A596-FA578C2EBDC3} InprocServer32]
C:%\Program Files\Common Files\Adobe\Acrobat\
ActiveX\AcroIEHelperShim.dll
```

When PWS-Banker!dtl malware is installed, it adds a CLSID with a length longer than 16 bytes.

```
[HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion
\Explorer\Browser Helper Objects]
{399BFACE-3ADA-4DAE-80D8-E221812243A9}
80D8-E221812243A9}
```

However, when Internet Explorer loads the BHO, it reads only the first 16 bytes and the rest of the characters are ignored; thus the following dll is found:

```
[HKLM%
\SOFTWARE\Classes\CLSID\{399BFACE-3ADA-4DAE-80D8-E221812243A9}
\InprocServer32]C:\WINDOWS\system32\flashcpx.dll
```

Therefore, since the ID is longer than permitted, other applications, for instance the Add-on Manager, will not find the malware sample. However, the malware will be launched upon each Internet Explorer execution.

5.4 Analysis of a BHO malware

In order to understand attack vectors based on BHO implementations, a complete analysis of a sample was done. The analysis included a static and a dynamic study of the sample. PWS-Banker!1d2e aka. Banker is a well known malware since it started to spread as an attachment in e-mails with subjects such as “Naked World Cup game set” or “Soccer fans killed five teens”. As detailed before, the sample could potentially steal an users’ bank account and social network credentials.

5.4.1 Behavioral analysis

Once installed, the malware drops mac.dll and creates the following registry entries:

```
Hkey-Local-Machine\SOFTWARE\Microsoft\Windows\CurrentVersion\
Explorer\Browser Helper Objects \{FCADDC14-BD46-408A-9842-
CDBE1C6D37EB\}
```

Next, it generates the following files:

```
\Windir\sysdir\logo1.jpg (dll file with jpeg extension)
\Windir\Cpu.log (logfile containing infected machines hardware details)
```

When executed, a message box appears, warning the user about a problem regarding insufficient memory. When the message box is closed, the user is redirected to the social network orkut.com. The malware waits for the user to log in in order to steal credentials. Finally, the malware inserts a new entry in the user's wall, showing a url pointing to a malicious website containing an executable, which is a downloader that ultimately will download the malware into other computers.

5.4.2 Static Analysis

A reverse analysis was performed over the sample. The malware was packed with UPack. PE Explorer was used to unpack the dll. Once unpacked, IDA Pro was used to perform the reversing process. A first analysis showed that mshtml!CBase::InvokeEvent call the method Invoke()f from the BHO.

Invoke function handles, among others, the following events:

```
sub\_20002153 =\> DISPID\_HTMLELEMENTEVENTS2\_ONFOCUSIN
sub\_2000271C =\> DISPID\_HTMLELEMENTEVENTS2\_ONCLICK
sub\_200030A8 =\> DISPID\_HTMLELEMENTEVENTS2\_ONFOCUSOUT
sub\_20003A11 =\> DISPID\_HTMLELEMENTEVENTS2\_ONKEYDOWN
```

Once the above events occur, the BHO code may perform a specific action.

5.5 BHO malware development

A BHO was developed to be used during the experimentation. Visual Basic and ATL were used to build it. The only goal of our malware is to receive information from certain events if a query to a certain bank site is performed.

All BHO need to implement the IObjectWithSite interface, which provide a lightweight siting mechanism. IObjectWithSite allows a container to pass the IUnknown pointer through SetSite() method. This is how the BHO will be able to query other inerfaces. The event DocumentComplete has two parameters: (1)pDisp a pointer to IDispatch(), which is needed to generate a connection with the browser and (2) pUrl, the URL. Both parameters are passed to IDispatch::Invoke as part of the event. IDispEventImpl() is the alternative to Invoke() provided by IDispatch. It works with an event sink to route events to the needed handler. In order to intercept events fired from the browser, BHO needs to connect to ir through an IConnectionPoint interface which provides the method Advise(). This method allows us to inform that our BHO is interested in getDocument().

Finally, in order to register the DLL Register a DLL the regsc32 command is used:

```
regsvr32 /s ``c:\ documents and settings\administrator
\my documents\visual studio 2010\ Projects
\mymalware\Debug\mymalware.dll"
```

5.6 Dynamic Instrumentation

Our hypothesis is that dynamic instrumentation can help to determine what events are generated by an application and which eventhandlers are used in order to detect the execution of an unknown BHO within an isolated system. An application created with Pin instrumentation framework, that is, a pintool, was used to take control of a Internet Explorer when it is executed and being uploaded to memory. At that moment PIN recompiles (JIT) different sections before the code is run. Pin provides the function RTN_InsertCall() in order to instrument certain functions. Thus, it is possible to know the number of times it is called, the arguments and even the output. Since several symbols can resolve to the same function address, it is important to check all function names before instrumentation. Initially the Invoke() function was studied. A Gaussian based model, similiar to the ones described in Chapter 3 was developed. In this case the average and deviation is based on the number of invoke() calls found on each Internet explorer execution. Executions of browsers showing z-score values higher than 1 would indicate an anomlay.

5.6.1 Training

A Virtual Machine with Windows XP and clean Internet Explorer 6 was prepared and the pintool counting the number of calls to invoke() was deployed. If the Pin application was able to count more than the normal number of invoke() calls we could state that a BHO is being executed. The normality was calculated based on a Gaussian distribution derived from 100 executions of the browser. During each execution the browser would automatically connect to a set of preconfigured web pages.

5.6.2 Testing

The dynamic instrumentation of the browser execution with *banker* sample installed was performed 10 times.The results showed an z-score higher than 1 in all executions.

In order to prove the validity of the approach, the same test was performed with 32 more BHO with a hit ratio of 100%. Our technique demonstrated to be effective against bho hiding techniques. However, the presence of a BHO extension does not involve the detection of a malware sample.

5.7 Conclusions

This chapter allowed us to develop the first detection schemes based on dynamic instrumentation of browsers. We focused BHOs since many malware samples use such technique to attach to browsers. We demostrated that the appaerance of certain function calls indicate the presence of a BHOs . The experiments showed that dynamic instrumentation allows us to rapidly identify the execution of potentially dangerous BHOs. In the next chapter, more complex Pin applications will allow us to detect specific attacks to banking sites perpetrated, not only via BHO but also with new MitB techniques.

6

Client Side Detection Approaches

As detailed before, attacks that take profit of user connections to perform specific actions while connected to a bank, render difficult the evaluation against normal behavior models based on web logs because most of the extracted variables (geolocation, time, etc.) stay within normal values. Only the above mentioned familiarity graphs could give an idea of a suspicious transfer. However, not only the server side can provide valuable information about anomalies during a banking session. When a MitB attack is performed browsers execute actions that differ from the normal behavior: new files with credentials or user use profiling information can be stored or sent to an email account, changes in the URL can be done, etc.

A prototype framework was developed in order to detect such anomalies from the client side. The prototype can automatically generate behavior models based on parameters extracted from dynamic instrumentation techniques and evaluate if a certain binary is compromising the browser and targeting a specific bank site by comparing the execution of the browser in an infected environment with the normal execution models. The framework involves two critical parts: the binary execution environment and the evaluation environment.

Our approach is based on the idea that when a MitB attack is performed browsers function calls must change from the normal behavior. Three different approaches were studied: (1) Number of functions calls, (2) subsequences of functions calls comparison and (3) transitions of function calls modeling through Markov chains.

This chapter is divided into five sections: First, the deployment of the execution environment is addressed; second, the detection techniques are indicated. Number and subsequences of function calls approaches is detailed, and a new method based on the number of unusual pairs of function calls; third, samples participating in the experiment are described; fourth, a detailed analysis of the different families is performed; finally, section 5 presents the summary of the chapter.

6.1 Malware Data Set

The election of the malware was based on the need to most spread banking attacks. Apart from the three malware families detailed in Chapter 5 (KillAV, Banker and Banbra), 7 more families have been selected in order to obtain different types of attack vectors and techniques. Next, a brief description of each malware is provided:

6.1.1 Bancos

This old family, dated from 2009 and programmed in Delphi, targets mostly online Brazilian entities. Once the user information is captured, it is sent to the hacker by ftp or email.

6.1.2 Nabload

Nabload is similar to Banbra due to the programming language, Delphi, and the target, Brazilian Internet banking websites. It removes security applications such GbPlugin required by some entities to prevent attacks. The stolen credential are sent to fe-mail.net and inbox.com mail accounts. Some versions of Nabload are known for being ditributed through messenger and for showing a funny videos. The malware is known for circumvent many antivirus. After checking the malware against 42 antivirus through Hispasec's engine, it was noticed a low detection level, 60,5%.

6.1.3 Sinowal

Also known as Torping, the first versions of Sinowal appeared in 2008 along with Mebroot rootkit, which would replace the Master Boot Record (MBR) of the infected machine. The malware was capable of stealing 500.000 credentials in less than one year. It was in 2011 when the most harmful versions appeared. They included Windows 7 compatibility and MitB techniques.

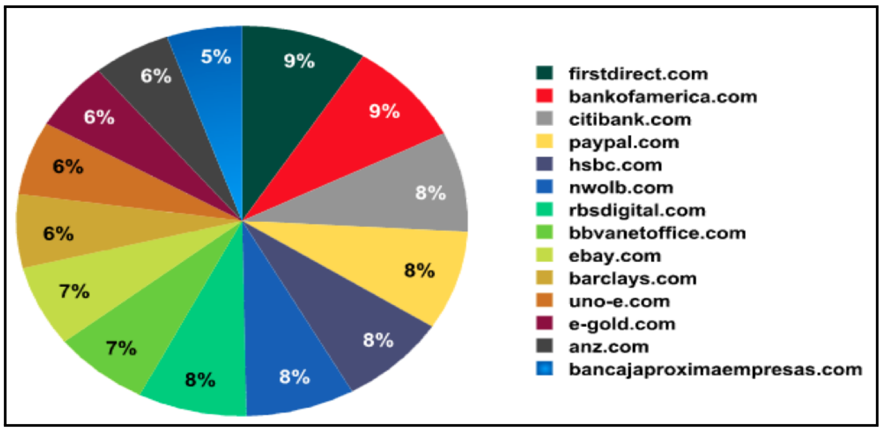


Figure 6.1: Top Level Domains Targeted by Zeus (Kaspersky Lab)

6.1.4 Zeus

Zeus is probably the most spread trojan horse since 2008. Since then more than 2.000 companies have been affected due to the possibility of acquiring a configurable kit. Figure 6.1 shows the most targeted Domains in 2010. The kit, which can be bought through underground forums, contains all the necessary data to create a botnet. The source code of the application was leaked in mid-2011.

6.1.5 Spyeye

Spyeye was Zeus' rival from 2009 until 2011. Some versions of Spyeye were even capable of removing Zeus samples. First versions of the software were written in C++. The software was also available in the form of toolkit to facilitate the fraud. It is also easy to use and no high skills are needed to create bots. Some of the latest versions of the software included interesting inprovements to hide malicious actions by imitating users behavior. Both Zeus and Spyeye have shared some features from the first versions such as the MitB technique, API hooking. Other features, such as the HTML injector, have been copied in Spyeye versions. The original spyeye injection mechanism deleted cached htmls to assure that the pages were always downloaded. However, since version 1.3 Spyeye improved the mechasism by including injection code in the cached pages, a feature previously found in Zeus. Spyeye has been found in multiples Frauds against entities such as Verisign.

6.1.6 Silent Banker

Written in C++, most Silent Banker samples had the capability to capture screen images and record Keystrokes from financial sites(52). Samples known as SilentBanker.D also had the ability to intercept banking transfers through API hooking techniques applied to methods such as Send, HttpOpenRequestW(), HttpSendRequestA() or DispatchMessageA().

6.1.7 Gozi

This Russian trojan appeared first in 2007 and was distributed to steal credit card information as well as banking credentials (38). The technique used by this family is different from the rest in that Gozi sniffs SSL connections by hooking JavaScript engine. The malware performs an analysis of the victim to determine the amount to transfer. Gozi infects their victims by establishing itself as a LSP (layered service provide) in order to bypass SSL/TLS. This technique differs from IAT hooking and BHO described before.

6.2 Environment Modules

The research was based on multiple executions of dynamic instrumented browsers accessing a simulated bank in such a way that low level information can be gathered. During each test, the instrumented browser automatically followed a pre-established clickstream. Different pin-tools were deployed on the execution virtual machine to gather function calls information in order to study the behavior of clean browsers when interacting with web sites. The training environment included the following modules:

6.2.1 Data collector module

In order to test the generated models, real malware samples were collected. Honeypots have become a very useful tool to collect and analize malware. Most of the honeypot implementations are based on the analysis of network traffic and systems changes within an isolated environment. Unfortunately existing honeypots suffer from some known problems: Many resources are needed to properly maintain the infrastructure. While server honeypots offer vulnerable ports in order to collect traffic and binaries from possible attacks, client honeypots(53) generally provide a browser to crawl a list of URLs. Both a server and a client honeypot were deployed in order to collect suspicious samples and all collected files were checked with ClamAV in order to obtain the final data set of found malware. The server honeypot was based in a Diaonea (54) and installed in a public network at Universitat Politècnica de Catalonia. Although, the machine was online for more than three moths only three malware samples were obtained and none of them were banking trojans. The system includes a set of tools such as binary hashes checker and antivirus to alert in case a suspicious binary is encountered. The low number of samples was due to a backbone Firewall filtering some widely used protocols such as Netbios. A client honey pot, based in Trigona was deployed along with a list of URLs extracted from the following sources:

- **Malicious URLs**. Various entities offer a datasets of malicious URLs containing malicious javascripts or binaries. However, although the listed sites are suspicious at the time they are published, many of them are taken down few days later. A first list was taken from MalwareURL Team (55) .

- **Incorrect URLs**, similar to real bank sites can lead users to fraudulent servers containing phishing or drive-by-download malware. Criminals generally register typographically incorrect domains. Many German and US banks have suffered attacks related to such domains. A list of 1300 typosquattered domains was also generated from European, US and South American banks.

- **Mail**. Multiple fake e-mail addresses were configured in order to collect malicious e-mails containing spam and phishing messages. A mail trap system based on Moreno's approach (56) has been deployed. Four fake e-mail adresses were included in different web servers of the University. Four other accounts where created from known email providers (Yahoo and Hotmail). The binary harvester connects to the different accounts, downloads the emails and stores them ina MYSQL server. The attached files are stored whithin the file system. The new Database entry includes the directory where the attached file was placed along with he MD5 and a timestamp. Once an email was received, all possible URLs were extracted. The mail accounts remained available for 3 months and 110 emails containing URLs were obtained.

Technique	Malware samples	Unique Samples	Desired samples
Server Honeypot	3	0	0
Honeyclient (blacklists)	280	30	3
Honeyclient (mail)	4	3	0
Honeyclient (typposquatered)	1	1	0
Manually collected	6	6	7

Table 6.1: Malware collection sources

The rest of the malware came from a variety of repositories such as Offensive Computing (57), VXHeavens (25) and VirusTotal (26).

Table 6.1 shows the results of the collected malware samples:

	Attack vector	Family Aliases	source
Bancos	logger (Keyboard hooking)	Trojan-Spy.Win32.Bancos.uq	Blacklist(VXheavens)
Kill AV	logger (BHO)	Trojan.BAT.KillAV.ae	Blacklist(VXheavens)
Banker	logger (BHO)	Trojan-Banker.Win32.Banker.alr.c	Blacklist(VXheavens)
Banbra	logger (BHO)	Trojan-Banker.Win32.Banbra.j	Blacklist(VXheavens)
Nabload	logger (keyboard hooking & screenshot)	Trojan.Nabload.BUG	Blacklist(VirusTotal)
Sinowal	MitB(IAT hooking)	Trj/Sinowal.DW	Blacklist(VirusTotal)
Spyeye	MitB (API hooking)	Trojan.Spyeye.H	Honeyclient (Blacklist)
Silent Banker	MitB (BHO)	Trojan.Silentbanker.E	Honeyclient (Blacklist)
Gozi	MitB (LSP)	Trojan.Gozi	Blacklist(VirusTotal)
Zeus	MitB (IAT hooking)	Trojan Zeus (Zbot)	Honeyclient (Blacklist)

Table 6.2: Malware features

6.2.2 Information gathering module

The information gathering module is twofold: it initially collects information during the training phase in order to produce the models; later on, a clean snapshot will be continuously infected. Once malware is running, an Internet Explorer instance is executed and a sequences of queries to a fake bank server are performed.

6.2.3 Execution module

The execution module consisted in to two VMs: The execution system and the simulated web bank. Although the simulated bank should be a replica of the real bank site to protect, an invented set of web pages containing authentication and transfers forms was included for testing purposes. A phishing web server was also prepared in order to check the behavior of the browser when a trojan redirects an user to a fake server.

In order to perform automatic sessions and gather the browser execution information, two elements were included in the execution VM: The session automatization and the instrumentation scripts along with Pin application. After different attempts to develop an application capable of simulating human behavior within a browser, a java script containing the URLs to be accessed was programmed. The html containing the java script was set as the homepage of the browser so once the browser was executed, the queries are sequentially accessed.

6.2.4 Anomaly detection module

The anomaly detection module, based in a Ubuntu 10.04 Server, included the neccesary scripts to: (1) receive the output of the execution module, (2) preprocess the results, (3) compare the data against the models and (4) present the information through a web environment.

6.3 Function call level approaches

The hypothesis is that any malware willing to perform any malicious activity during the browser execution will add additional calls. Two interesting features from Pin Framework were exploited: the ability to recognize a certain function during the execution and the possibility of establishing function call counters.

6.3.1 Number of function calls

New banking trojans hook to Wininet.dll library because its functions are in charge of the communication between the browser and the network layer. Each http/https query involves the call of three main functions:(1) InternetConnect() function initiates an specic HTTP and one of the arguments used is the actual web site; (2) later, HttpOpenRequest() makes use of InternetConnect()s handler to establish a connection and generates a query; (3) finally, HttpSendRequest() is in charge of sending the request. Additionally the extracted information is normally saved temporaly in the victim's computer so the CreateFile() function should be called more than usual. An new pintool was developed to get a counter of calls at different parts of each query of a banking session during the instrumentation of a browser and the number of calls

	CreateFile() calls	Counter 1[a]	Counter 2[b]	Counter 3[c]
IE	25	164	561	36285
IE with Zeus	44	147	1834	135763
IE with Spyeye	75	195	675	39824

[a] functions calls from InternetConnect() to HttpOpenRequest()
[b] functions calls from HttpOpenRequest() to HttpSendRequest()
[c] functions calls from HttpSendRequest() to InternetConnect()

Table 6.3: Preliminary test of function calls on Zeus and Spyeye

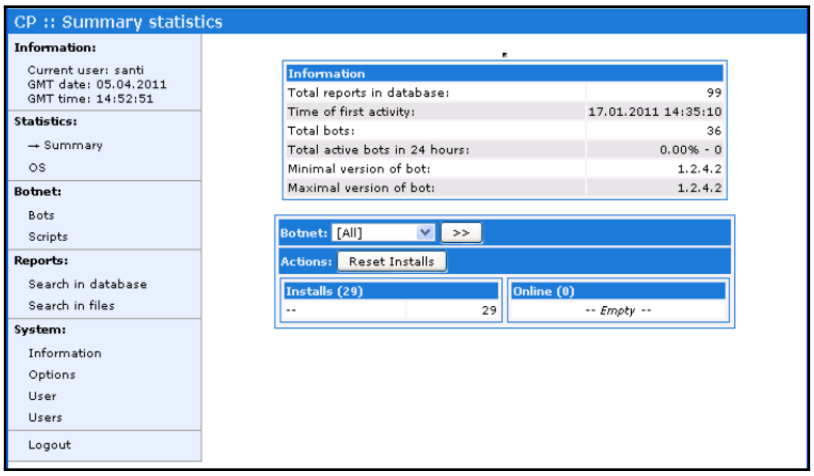
to Createfile() at the beginning of a session. The total number of function calls was divided in three counters:

- Counter 1. Number of functions calls from InternetConnect() to HttpOpenRequest().
- Counter 2. Number of functions calls from HttpOpenRequest() to HttpSendRequest().
- Counter 3. Number of functions calls from HttpSendRequest() to InternetConnect().

6.3.1.1 Toolkit test

A Zeus toolkit version 1.4.2 and Spyeye toolkit version 1.2.60 were obtained through a malware repository. Both software contain three elements to create bots: a builder, a configuration file and the needed php files to manage a Command & Control (C&C) server. The toolkits allows fraudsters to configure the target and the actions to be performed when the user queries an specific site. The configuration file (see Appendix 2) must include the IP of the server in order to send information a receive orders, the target banks and the injections in case the fraudster wants to show extra information such as new authentication forms over the real web site. Figure 6.2 shows Zeus C&C server. Both samples where configured to perform a redirection to the phishing server if an user connected to the simulated bank server. The name of the server was included in the *hosts* file of the Windows execution module to circumvent a DNS query.

In the Zeus case, outliers appear when comparing executions against counter 2 and counter 3 (see 6.3). It was also noticed that both malware samples show more calls to CreateFile() function than usual when initiating the session.



Current user: santi
GMT date: 05.04.2011
GMT time: 14:52:51

Information	
Total reports in database:	99
Time of first activity:	17.01.2011 14:35:10
Total bots:	36
Total active bots in 24 hours:	0.00% - 0
Minimal version of bot:	1.2.4.2
Maximal version of bot:	1.2.4.2

Botnet: [All]

Actions: Reset Installs

Installs (29)	
--	29

Online (0)
-- *Empty* --

Figure 6.2: Zeus screenshot

6.3.1.2 Banking trojans test

After the good results of the previous experimentation, the same test was performed with each malware sample infecting the execution VM. A Gaussian distribution model was generated out of 200 executions of an specic clickstream using Microsoft Internet Explorer. Since each execution involved a clickstream of 10 queries, 2000 observations were obtained. An observation is considered anomalous when it exceeds the area compressed between the mean minus the standard deviation and the mean plus the standard deviation. Table 6.4 shows the average z-score of the executions. The results show that Counter 2, that is, the number of function calls from the HttpOpenRequest() call to a HttpSendRequest() offers promising results, averaging a detection rate of 70%
.

6.3.2 Different Subsequences of function calls

Portions of sequences of function calls from clean browsers were also modeled and compared to sequences extracted from infected malware. In this case, every observation involved a whole execution, that is, a sequence of all the function calls performed during all the simulated session. Again, a pintool was developed to gather the sequence of function calls from clean browser executions. Our detection system separates the list of calls gathered during a web session into subsequences of certain lenght k being k 2, 4, 8 or 16. Figure 6.3

	Counter 1	Counter 2	Counter3
Bancos	0,64	1,23	0,03
KillAV	0,44	1,63	0,02
Banker	0,54	4,60	0,66
Banbra	0,57	0,65	0,03
Nabload	0,66	1,60	0,04
Spyeye	0,12	2,26	0,79
SilentBanker	0,56	0,21	0,08
Zeus	0,29	30,2	41,35
Sinowal	0,58	0,5	0,01
Gozi	0,58	7,57	0,11
Anomalous[a]	0/10	7/10	1/10

[a] A test result is considered anomalous when de mean of the z-scores is higher than 1

Table 6.4: Number of function calls

Different sequences are defined as sequences appearing only in one of the compared executions.

While subsequences of 2 and 8 calls showed interesting results when detecting anomalies regarding different sub-sequences, subsequences of 4 and 16 calls showed a detection rate of 70%.

6.3.3 Distinct Subsequences of function calls

Distinct sequences involve sequences that appear one or more times counting multiples occurrences only once.

In the case of the number unique sequences of functions, that is, the distinct subsequences experimentation, 8 function call subsequence models showed a detection ratio of 90%. Although Bancos was not detected against any model, the Sinowal sample was found anomalous improving the detection rate of different funcion call models.

6.3.4 Uncommon transitions

Finally, 1st-order Markov models were generated from a set of function call sequences extracted from 200 executions of a clean browser. The goal of the experiment was to detect if a banking session represented by a sequence of function calls is abnormal. Similarly to the queries Markov models detailed in Chapter 4, a call sequence Markov

	2 sub-seq	4 sub-seq	8 sub-seq	16 sub-seq
Bancos	-0,62	-1,13	-0,83	-0,35
KillAV	-0,03	-0,01	-0,08	0,35
Banker	11,52	9,16	8,66	-15,74
Banbra	1,63	1,50	1,45	2,80
Nabload	1,40	1,63	1,90	4,08
Spyeye	13,15	17,92	26,00	44,19
SilentBanker	10,63	6,67	5,05	7,54
Zeus	12,27	9,25	7,85	11,14
Sinowal	-0,41	-0,12	-0,20	0,31
Gozi	0,05	0,06	0,42	1,51
Anomalous[a]	6/10	7/10	6/10	7/10

[a] A test result is considered anomalous when de mean of the z-scores is higher than 1(deviation)

Table 6.5: Different function calls sub-sequences

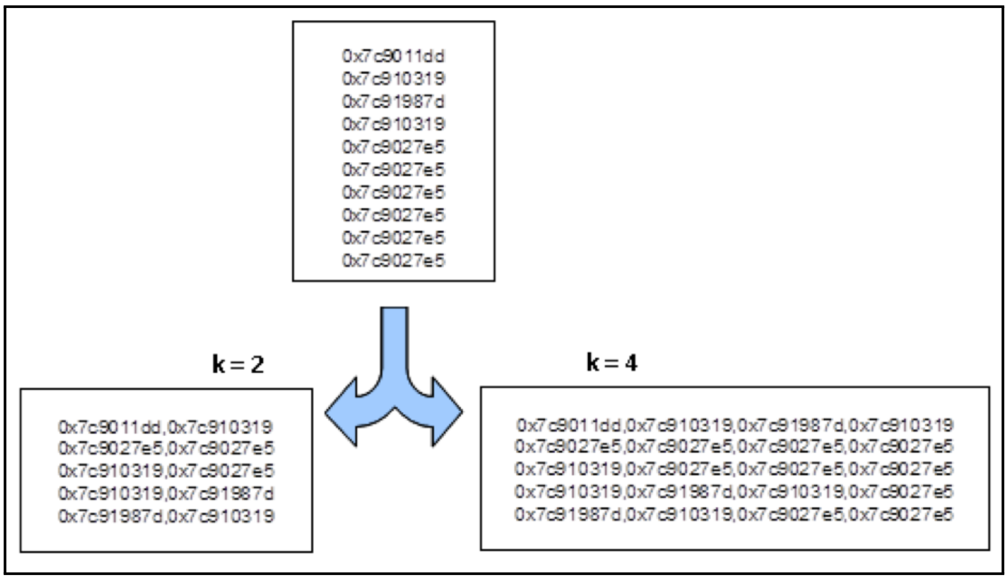

0x7c9011dd
0x7c910319
0x7c91987d
0x7c910319
0x7c9027e5
0x7c9027e5
0x7c9027e5
0x7c9027e5
0x7c9027e5
0x7c9027e5
k = 2
0x7c9011dd,0x7c910319
0x7c9027e5,0x7c9027e5
0x7c910319,0x7c9027e5
0x7c910319,0x7c91987d
0x7c91987d,0x7c910319
k = 4
0x7c9011dd,0x7c910319,0x7c91987d,0x7c910319
0x7c9027e5,0x7c9027e5,0x7c9027e5,0x7c9027e5
0x7c910319,0x7c9027e5,0x7c9027e5,0x7c9027e5
0x7c910319,0x7c91987d,0x7c910319,0x7c9027e5
0x7c91987d,0x7c910319,0x7c9027e5,0x7c9027e5

Figure 6.3: Sub-sequences generation

	2 subseq	4 subseq	8 subseq	16 subseq
Bancos	0,26	0,41	0,30	0,37
KillAV	0,83	2,32	2,16	2,37
Banker	11,48	12,38	10,42	-158,43
Banbra	2,37	4,03	3,58	4,15
Nabload	2,32	3,60	2,94	3,58
Spyeye	-2,98	-5,80	-2,26	0,61
SilentBanker	8,67	7,44	5,16	5,27
Zeus	11,32	11,26	8,67	8,20
Sinowal	0,48	1,71	1,35	1,28
Gozi	0,72	1,19	0,61	0,49
Anomalous [a]	6/10	9/10	8/10	6/10

[a] A test result is considered anomalous when de mean of the z-scores is higher than 1 (deviation)

Table 6.6: Distinct function calls

Model is defined by the tuple of elements $\lambda = (S, V, A, B, \Pi)$:

A vocabulary $V = P$ based on M possible function calls:

$$V = P = \{p_r, 1 \leq r \leq M\}$$

A group of N states, $S = \{s_1, s_2, ..., s_N\}$. A state is considered as any sequence of k function calls. Thus,

$$N = M^k.$$

A equals to the transition probabilities among states,

$$A = \{a_{ij}, 1 \leq i, j \leq N\}$$

B equals to the probabilities to get a specific query any state,

$$B = \{b_{ir}, 1 \leq i \leq N, 1 \leq r \leq M\}$$

where b_{ir} is the probability of getting a function call p_r when leaving state s. The states and vocabulary were obatained during the training process. Training was based on the number of times where a state was visited and the performed transitions. The evaluation of a query to a function call within a session is based in:

$$W = \{normal, if AS(W) \leq \theta, abnormal, otherwise\}.$$

Malware	# Unknown transitions
Bancos	1,26
KillAV	1,49
Banker	98,52
Banbra	2,43
Nabload	83,98
Spyeye	8,68
SilentBanker	2,25
Zeus	21,78
Sinowal	0,94
Gozi	1,00
Anomalous[a]	8/10

[a] A test result is considered anomalous when de mean of the z-scores is higher than 1 (deviation)

Table 6.7: Number of unsual transitions

where θ is a threshold that was calculated experimentally.

However, the high number of transitions generated too complex Markov graphs, thus decreasing the performance of the automatic evaluation. In order to derive profit of the models, the number of unusual transitions was modeled. A transition was considered unusual when the probability b_{ir} of getting a function call p_r when leaving state s was lower than 10^-3.

Table 6.7 shows the average z-score of the number of unknown transitions for each malware sample, that is, the number of deviations with respect to the mean for each set of infected executions. The results show 80% of the malware was correctly detected. The experimentation with Banker, Nabload and Zeus raised a high number of deviations with respect to the model.

6.4 Final results

According to Cyveillance analysis(58), most AV vendors detect around 19% of new malware during the first hours after the appearance of a new malware. Table 6.8 shows that the combined use of the techniques above described would lead to the detection of all the analyzed samples. Although the set of chosen malware were tested against 42 antivirus from VirusTotal and the average detection ratio was 85% , all of

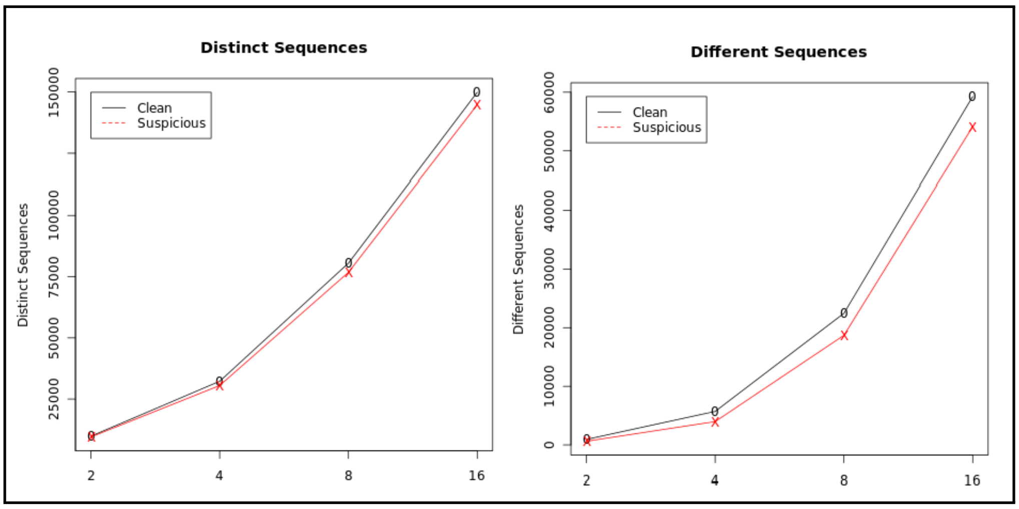


Figure 6.4: Malware sample visual comparison

them appeared at least three months before the experimentation. While 60% of the samples showed anomalous values in all tests, at least one model showed an infected session as anomalous. A false positive rate of 1% was found when evaluating clean sessions against the models. Although the combination of the different techniques is not the focus of the research a weighted sum of all of them show promising result in a real environment. Finally, the use of an specific simulated bank web server would also help to quickly determine if a sample is targeting a financial entity. In order to visually evaluate the results a web based dashboard was developed. The dashboard implemented in the anomaly detector machine shows graphically the comparison of a given infected session with respect to the detailed models.

A more detailed analysis provided interesting results. The test confirmed that 75% of the MitB banking trojans (Gozi, Zeus and Spyeye) involved a high abnormal increase in the number of functions. Older malware developed with BHOs showed better results with the distinct sequences approach. However an expected exception was found: Bancos, the malware performing keyboard hooking. Since the sample does not hook into the browser process no abnormal values were found.

Malware[a]	# Func.	Distinct Seq.	Different Seq.	Transitions
Bancos	X		X	
KillAV		X		X
Banker	X	X	X	X
Banbra		X	X	X
Nabload	X	X	X	X
Spyeye	X	X	X	X
SilentBanker	X	X	X	X
Zeus	X	X	X	X
Sinowal		X		
Gozi	X	X	X	X

[a] The table shows which models arise anomalies given each malware infection

Table 6.8: Final results

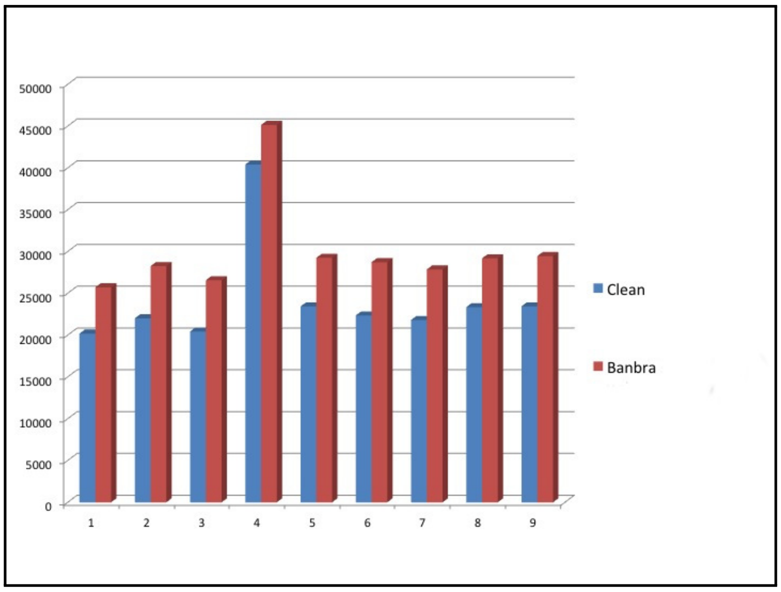


Figure 6.5: Banbra HSR–IC calls

6.5 Conclusions

In this chapter we presented a new approach to detect attacks against financial institutions sites. Models based on clean browser function calls extracted from multiple executions were generated and successfully evaluated against the most spread banking malware samples using MitB techniques. It could be possible to place our detection technique within customers computers by deploying them as browser plugins. Moreover, models based on function calls should be adapted to mobile environments, thus mitigating recents attacks against mobile banking applications. Complementing detection techniques, recent years have experienced an increasing interest in protection techniques mostly based on addning new authentication schemes ton online sessions. Out-of-band authentication based in mobile phones is a cheap and extended method. However, criminals have deployed new techniques to gather such the necessary data from user devices to complement MitB attacks.

Part III

Proposed Extensions of Dynamic Instrumentation on Anomaly Detection

7

Mobile Environments

In September 2010 Zeus in the Mobile Zeus in the Mobile (ZitMo) was discovered. The malware was intended to steal TANs sent by financial entities to customers when a banking transaction was conducted. While in principle the malware was developed for Symbian devices, shortly after samples appeared for Blackberry and Windows Mobile. The malware appeared to be a security application Android Called Premium Security Suite. The malware, which works along with the PC version is able to intercept SMS messages. The use of mobile devices as first and second authentication channels has increased the attention of fraudster and new samples are appearing daily. In this chapter we address new threats targeting mobile devices and detection techniques applied so far. Finally, we provide new approaches based on modeling certain applications in order to generate models that should reveal suspicious actions within a device.

7.1 Mobile threats

More than 450 million mobile phones were sold just in 2011. However, the number of vulnerabilities has also experienced an important increase. Next shows the holes reported by the National Institute of Standards and Technology (NIST) classified by platform.

Although more vulnerabilities have been reported for iOS, the rapid security update process has been of importance in the little appearance malware samples. According to Gartner, Android ships more than 50% of the smartphones sold. More than 200 million android devices have been activated up to date. However, according to a re-

	2009	2010	2011
iOS	50	106	91
Android	11	21	60
Symbian	1	0	0
Windows Mobile	1	1	2

[a] It includes several sequencial observations

Table 7.1: Vulnerabilities per platform

port performed by Harry Sverdlove (59), many Android smartphones are shipped with an out-dated OS version. Moreover, third party applications such Java or Flash are included, thus increasing the risk of the devices.

Once the malware has been introduced, any communication channel can be used to replicate itself. Back in 2004, the first detected virus targeting Symbian OS mobile devices, named Cabir, was already able to spread through Bluetooth. Little after, skulls, the first mobile Trojan appeared (60). Another well know malware, Commwarrior (61) had the ability to propagate through Multimedia Message (MMS) and Bluetooth by sending infected SIS files.

Although the first malware samples were Proof of Concept applications, two factors have leaded the great increase on the malware samples appeared recently:

- The use of 3G and Wireless networks has helped to create different markets providing thousands of applications, some of which have been found to be malicious.
- Nowadays, malware proliferation has expanded to most platforms. Apple has been already hit by malware. Although little malware samples have been discovered for iOS, Kaspersky researchers expect iPhone and iPad devices to be widely targeted by 2013. According to a recent study conducted by AV-Test, most antivirus for mobile devices are, today, almost useless. Many of them only seek evidence of installed malware, so if the malware resides on a removable media it is not detected. On the other hand signature-based techniques are not valid against obfuscation-enabled applications.

Two type of mobile malware are proliferating in the last years:

- Rogue malware performs actions differently than expected while masquerading as a useful application.

- Trojanized applications are based on legitimate files that have been modified by malicious code.

7.2 Malware analysis techniques

The search for malicious patterns through heuristics has helped to increase the detection rate. However due to the increasing hiding techniques, other approaches tend to model he behavior of goodware.

Although manual analysis provides much more information on the studied application it is extremely time consuming and not applicable to the user environment. As an example, we briefly detail Dhawal Desai (62) analysis on the ZitMo Android version. Two different methods were used:

- A behavior analysis was performed through the execution of the sample in Android emulator which allowed to monitor SMS, network and calls.
- Static Code analysis was also performed by unpacking the application and obtaining Java Bytecode with dex2jar and a java decompiler. It was found that SmsBlockerThread was used to intercept messages.

Although the above analysis provide an interesting amount of information to take the proper countermeasures, and static code analysis of an application can involve 2 days of an expert. Apart from the traditional signature-based approaches, some automatic anomaly detection techniques have been also deployed. However, the mining technique used to generate the proper model is crucial to optimize the detection rate. Due to the limited performance of smartphone devices, some approaches such Crowdroid (63) have adopted a distributed environment to share information on goodware syscalls in order to identify malicious samples.

7.3 Proposed approach

We propose to generate behavior models at function call level for goodware applications based on parameters extracted from dynamic instrumentation techniques and evaluate if a certain binary is compromising the environment by comparing the execution of the system in an infected environment with the normal execution models. The following examples illustrate our approach:

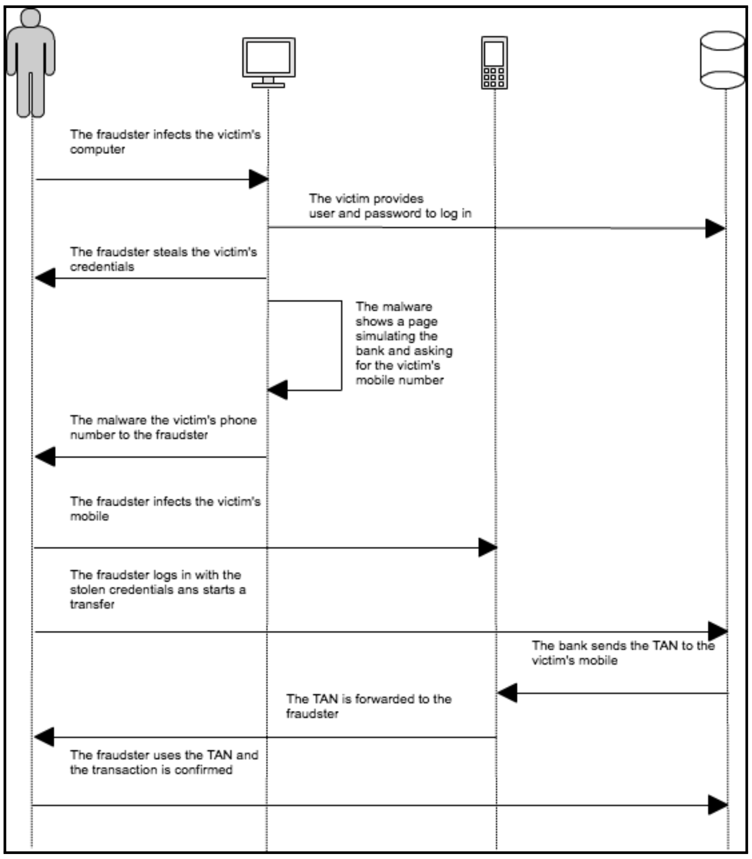

The fraudster infects the victim's computer
The victim provides user and password to log in
The fraudster steals the victim's credentials
The malware shows a page simulating the bank and asking for the victim's mobile number
The malware the victim's phone number to the fraudster
The fraudster infects the victim's mobile
The fraudster logs in with the stolen credentials ans starts a transfer
The bank sends the TAN to the victim's mobile
The TAN is forwarded to the fraudster
The fraudster uses the TAN and the transaction is confirmed

Figure 7.1: Man in the mobile sequence

- Mobile browser. The use of browser installed in mobile devices to perform banking sessions can generate a new form of threat due to the lack of security measures taken so far. The use of HTML5 has increased the number of browser based applications. The modeling of the browser when performing specially sensible actions could greatly help to determine possible threats.

- MitMo and Android. SendBroadcast() allows Android send operations to other applications. All applications willing to get SMSs must register a Broadcast receiver in order to intercept the messages. MitMo malware deploy a Broadcast receiver to forward the SMS and send it to a C&C. An instrumentation application capable od listing all function calls after SendBroadcast() could help to determine the normal behavior that an Android OS follows when received a TAN during the confirmation of a transaction.

	Symbol Information	Common Architectures	DBI	Common SDKs
iOS		Samsung	Valgrind	XCode
Symbian		Texas Instruments		QT/Carbide.c++
Android	Glibc	Qualcomm	Pin/Valgrind/Gdb	Eclipse/Netbeans
Windows Mobile	pdb files	Intel/Xscale/Marvel/Qualcomm	Entrek	Visual Studio

Table 7.2: Mobile analysis tools

Our approach is based on the idea that the list of function calls of an infected device OS execution must be different from the ones extracted from a clean OS. As stated above, we propose Dynamic Binary Instrumentation as the most effective information gathering technique. However, in order to choose the proper research framework, different variables must be studied: architecture, language, symbol information and available SDKs. Table 2 shows different options available for the most used platforms.

ARM compatible architectures can be found in 90% of smartphones devices. However, not enough research has been carried out on the instrumentation over such processor. In order to illustrate an example, the three frameworks described above where deployed in an Android emulator in order to check pros and cons:

- **Pin**. Chi-Keung Luk et al (64) released during 2006 and 2007 different versions of Pin which would support the ARM architecture. Although the framework does compromise the performance of the system, the old compiler version used (3.3.1) and the need of cross-compiling each pintool supposes a timeconsuming effort for the preparation oft he environment.

- **Valgrind**. Last version of the software (3.7.0) is only available for Nexus S and Pandabards. Nethercote et al. published a first approach (65) on the deployment of Valgrind in order to memcheck Firefox. However some difficulties may arise during the process due to the unavailability of a research ROM. The original ROM is not valid for instrumentation purposes due to the lack of symbols information and the small swap space. The generation of a ROM and kernel image is as complex as the possible research to be accomplished after the setup process.

- **Gdb**. Android provides a gdb server in order to attach to the application being analysed. The server can be run through the Android Degub Bridge (adb) included in the Android SDK which is available for Windows, MAC OS and Linux.

The techniques proposed for browser instrumentation in Chapter 6 can also be applied in the mobile field: number of functions calls, subsequences functions calls and finally comparison and transitions of function calls modelling through Markov chains.

7.4 Experimentation

Initial experiments were carried out with gdb and Android OS. Gdb allowed us to partially instrument the execution of webkit, an open source browser for Android. Following the process carried out during the intrumentation of Internet Explorer, we collected the number of function calls of Webkit during certain sequence of queries. The test was repeated 20 times to create a simple Gaussian model of the total number of function calls during the session.

Next, we simulated a phishing by performing a test based in visiting the same sequence of pages, but in this case pertaining to the phishing simulated site. Although all the pages seemed to be equal to the real server, one of them contained a malicious javascript. The instrumentation of the malicious session produced a clear increase in the number of function calls thus generating an outlier. With this simple experiment we demostrated that the aproaches analyzed for Internet Explorer can also be applied to mobile environments in order to detect malicious actions.

8

Critical infrastructures

Nowadays, environments such as home, work, leisure, and transportation among others, are generally populated with a wide range of ubiquitous devices, allowing users to dynamically interact with them via infrastructure (e.g. hotspots) or with other users on an ad-hoc fashion. This interaction is carried out using the device's available communication channels (i.e. Bluetooth, WiFi, GSM, 3G, Zigbee, etc), granting not only information sharing but also providing the capability of sharing the devices resources (e.g. displays) and performing a great variety of tasks, such as, collaborative work sessions, remote execution of tasks, etc. However, the benefits given rise to different sorts of malware that point to a wide variety of targets. Therefore, these environments require new mechanisms able to ensure that resource sharing is conducted safely, and that such environments are able to detect and inform about any anomalies that might occur during an interaction. Additionally, the World Wide Web has become the most popular feature on the Internet and one of the preferred channels of communication between customers and service providers, most of which are moving many services online to reduce their costs and facilitate clients interactions. This market is a key target for criminal organizations willing to pay experts to create suitable malware samples to get as much profit as possible. Unfortunately, traditional signature-based detection methods are nowadays easily circumvented due to the amount of new malware samples and the use of sophisticated evasion technique.

In the last years, the Mark Weiser's vision are realizing such as intelligent spaces are becoming more pervasive than ever (also known as ubiquitous environments). On one hand, airports, shopping malls, museums or even Supervisory Control and Data

Acquisition (SCADA) systems are everywhere and their infrastructures offer different mechanism to communicate with any user carrying a mobile device. For example, museums have Wifi hotspots, sensors to detect presence or temperature, or sensors for guided visits. Two environments are analysed: Mobile devices and SCADA systems. Using their mobile devices, users can control/monitor systems, receive advertisements/notifications, or send arbitrary data, which opens the door to new potential threats. The goal of SCADA systems is to control and monitor large infrastructures such energy plants, transport hubs and other Critical Infrastructures (CII). SCADA networks are based on three elements: (1) **Remote Telemetry Units (RTU)s**, devices connected to physical objects in order to provide and receive information; (2) **Human-Machine Interfaces (HMIs)** presents the information to the operators and allows to control all the elements, and (3) the **communication infrastructure** is based in old proprietary protocols that initially were not connected to the Internet thus not applying special security procedures. According to the World Economic Forum, a major Critical Information Infrastructure breakdown would have a potential global cost of 250 billion dollars. It was estimated that the probability of such event to occur is 10 to 20 %. Although the environments are completely different, both mobile devices and SCADA systems share some similarities:

- **Integration with Internet**. The appearance of 3G has greatly helped smartphones to access to multiple services offered through Internet. SCADA systems have also taken benefit from the Internet facilitating remote administration of the different devices involved through the access to the HMI.
- **Critical Information**. Mobile devices have become one of the main storage systems for customers. The increase in the number of services and hardware capabilities allow users to centralize contacts information, virtual disks or agendas. Many financial entities also take advantage of mobile devices in order to provide second-factor authentication during an online banking transaction. On the other hand, SCADA systems are generally deployed in large infrastructures related to energy production or communication systems that deal with critical information for the continuity of basic necessities of every day life.

The purpose of an attack on a CII could range from a hacker trying to prove he can get through defences, to a terrorist that wants to damage a major nuclear plant.

A successful attack over a CII could result in serious danger to population. Due to the recent connection to TCP/IP networks, various samples have targeted CIIs lately:

- 2003. SQL **Slammer** affected Davis Besse nuclear plant and its monitoring systems.
- 2003. Sobig was blamed for shutting down train signaling systems throughout the east coast of the U.S.
- 2005. **Zotob** was a worm that spreaded itself by exploiting the Microsoft Windows Plug and Play Buffer Overflow Vulnerability. In August 2005, Zotob crashed thirteen of DaimlerChryslers U.S. automobile manufacturing plants forcing them to remain offline for almost an hour. Plants in Illinois, Indiana, Wisconsin, Ohio, Delaware, and Michigan were also forced down.
- 2010. **Stuxne**t was a hot topic in security during 2010 (66). The malware had the ability to exploit four zero-day vulnerabilities. Stuxnet has been successful in targeting Siemens HMI around the world using SCADA (supervisory control and data acquisition) systems. To date Stuxnet has spread from computer to computer via USB memory sticks and through copying itself to new networks protected by weak passwords. Stuxnet, which comes with a rootkit, could be used by cyber criminals or a hostile nation state to disrupt operations in power plant, chemical manufacturing, transportations, and other critical industrial systems.
- 2012. Although **Flame** malware targeted many CIIs, the goal of the sample is still unknown. The application includes some zero-day exploits and is capable of stealing information from USB devices, network connection or keyboard.

Most studies on SCADA Intrusion detection approaches are based to network information. SANS Institute refers to Network Anomaly detection as "Instead of training models on normal behaviour, Network anomaly detectors build models of TCP/IP protocols using their specifications". After the outbreak of virus Nimbda, no signature-based detection systems (NIDS) could detect the presence of the virus. In order to prevent such attacks, anomaly de-tection methods have been applied to network protocols such Modbus. Valdes et al. [(67) proposed a Process Control Systems network detector based in adaptative learning techniques. However, threats such information leaking may not occur within SCADA private network. Given an internal communication protocol detector, malicious actions carried out from the HMI to the RTUs can be detected. However, a malware could intercept certain data in order to send it through the TCP network to a remote server. None of the previous studies include the analysis of the behaviour of the SCADA HMI itself to detect such behaviour. Our approach aims to generate models capable of gathering the behaviour of the HMI in order to perceive a clear idea of the normal actions taken within a SCADA system. The dynamic instrumentation has proved to be a useful tool for detecting malware that only works under specific circumstances. Dynamic binary analysis is undoubtedly the best choice to study the behaviour of the software and create models of normality. The three techniques detailed above (number of function calls, subsequences and Markov call graphs) could be applied to model HMI applications in order to detect anomalous actions in the HMI that could lead to data leakage and remote control of the RTUs.

Part IV

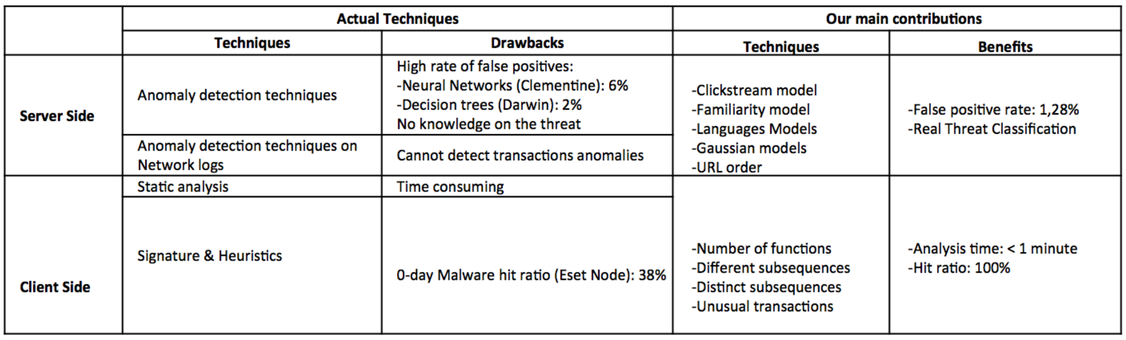
Conclusions

9 Conclusions

Online banking industry has suffered a wide range of illegal practices over the years. Due to the fact that most losses are originated from malware samples installed in the customer side, financial companies have relied on signature-based antivirus and strict authentication controls. However, most of those countermeasures have demonstrated to be useless against threats attacking all communication channels between customers and banks. In this thesis we presented new approaches to help financial institutions to fight against fraudsters using malware samples. We focused on two main goals: (1) real time detection of anomalies and (2) rapid analysis of new suspicious binaries targeting customers. Due to the amount of information generated during a banking sessions, both detection vectors were based in Data Mining techniques such as Markov chains, Market Basket analysis or clustering algorithms.

The preface of the document introduced the background of this thesis and the procedures and tools needed to perform the analysis.

The second part involves the main contributions and was divided in two areas of research. Chapter 4 showed that our new methodology, based on using web queries to detect attack patterns, can successfully detect anomalous transactions before the fraud occurs, thus giving the chance to banks to proactively react in real time during the customer session. Our approach can also help to identify previous malicious actions related to credentials theft. Although the false-positive rate (1,28%) of the results was low, new attacks techniques tend to minimize the interactions with the bank, thus reducing the amount of valuable information. The search of new sources of information led us to the analysis of the malware samples. Our new hypothesis was that dynamic



	Actual Techniques		Our main contributions	
	Techniques	Drawbacks	Techniques	Benefits
Server Side	Anomaly detection techniques	High rate of false positives: -Neural Networks (Clementine): 6% -Decision trees (Darwin): 2% No knowledge on the threat	-Clickstream model -Familiarity model -Languages Models -Gaussian models -URL order	-False positive rate: 1,28% -Real Threat Classification
	Anomaly detection techniques on Network logs	Cannot detect transactions anomalies		
Client Side	Static analysis	Time consuming	-Number of functions -Different subsequences -Distinct subsequences -Unusual transactions	-Analysis time: < 1 minute -Hit ratio: 100%
	Signature & Heuristics	0-day Malware hit ratio (Eset Node): 38%		

Figure 9.1: Contributions

binary instrumentation of samples should allow us to better understand malware attack vectors and rapidly detect malicious actions. The first experimentation of our idea is detailed in Chapter 5, were we address a technique historically used to deploy banking malware: Browser Helper Objects. The analysis of the problem allowed us to introduce binary instrumentation techniques to create models of normal behavior. We deeply analyse the binary instrumentation techniques in Chapter 6 in order to improve the detection of the latest techniques such as Man in the Browser attack vectors. Models based on clean browser function calls extracted from multiple executions were generated and successfully evaluated against the most spread banking malware samples. All of them were detected by, at least one instrumentation based model.

The third part of the this thesis was focused on possible uses of binary instrumentation modeling approaches to different areas. We have addressed the detection of Man in the Mobile based samples, applying instrumentation techniques to certain mobile services in order to mitigate attacks on Transaction Authentication Numbers. Chapter 7 focuses in the last threats related to the use of mobile phones for banking purposes. We proposed the use of models based on normal behavior models to detect malicious activities. Finally, given the good results of our approaches, in Chapter 8 we also propose to extend the techniques to critical infrastructures protection, due to the environment similarities, the increase in the number of vulnerabilities affecting such platforms and exposure due to the recent connection to the Internet.

Due to the number of attacks and malware samples, anomaly detection will probably become the main proactive technique against new threats in the next years. Although multiple combinations of hardware, Operating Systems and applications can generate different execution behaviors, it could be possible to generate repositories of goodware models that would greatly help on the detection of unknown samples. The correlation of different sources of information could also reduce the detection time. The increasing use of cloud may allow to centralize customers' session information from various sources and provide the necessary computing capabilities to use more complex algorithms. Finally, the cloud can also provide a platform to safely connect to a banking session though a clean browser instance.

10

Appendix 1. Database definition

```
dbgraph TADS into './dades/TADS-schema.dex' (
      node USUARIO (
            ID_USUARIO string unique,
            NIF string indexed,
            TIMESTAMP timestamp
)
      node SESION (
            ID_SESION string unique,
            IP string,
            IDIOMA_NAVEGADOR string,
            IDIOMA_APLICACION string,
            RIESGO_MODELO int,
            RIESGO_IDIOMA int,
            MODELO_CLUSTER int,
            SESION_FINALIZADA bool,
            TIMESTAMP timestamp
)

      node CUENTA (
            ID_CUENTA string unique,
            TIMESTAMP timestamp
)
      node PAIS (
            ID_PAIS string unique,
            VIRUS_PAIS int,
            TIMESTAMP timestamp
      )
      node OP_LOGIN (
            ID_LOGIN string unique,
            VIRUS_LOGIN int,
            TIMESTAMP timestamp
      )
      node OP_CYBERTARGETA (
ID_CYBERTARGETA string unique,
            ID_TARGETA string indexed,
            TIPO string indexed,
            IMPORTE int,
            VIRUS_CYBERTARGETA int,
            TIMESTAMP timestamp
)
      node OP_RECARGA_MOBIL (
            ID_RECARGA_MOBIL string unique,
            ID_MOBIL string indexed,
            TIPO string indexed,
            IMPORTE int,
            VIRUS_RECARGA_MOBIL int,
            TIMESTAMP timestamp
)
      node OP_TRANSFERENCIA (
            ID_TRANSFERENCIA string unique,
            TIPO string indexed,
            IMPORTE int,
            ANOMALIA_TRANSFERENCIA int,
            VIRUS_TRANSFERENCIA int,
            TIMESTAMP timestamp
)
      node OP_OTRAS (
            ID_OTRAS string unique,
            TIPO string indexed,
            VIRUS_OTROS int,
            TIMESTAMP timestamp
)
      edge OP_TIENE_COMO_ORIGEN from OP_TRANSFERENCIA to CUENTA
            ( IMPORTE int indexed )
      edge OP_TIENE_COMO_DESTINO from OP_TRANSFERENCIA to CUENTA
            ( IMPORTE int indexed )
      edge OP_CLIC
            ( SEQUENCIA string indexed, ANOMALIA_URL int indexed )
      edge MODELO_CLICSTREAM
            ( SEQUENCIA string indexed, PORCENTAJE int indexed )
      edge GENERA_SESION from USUARIO to SESION
      edge OFICINA_PERTENECE_PAIS from OFICINA to PAIS
      edge SESION_PERTENECE_PAIS from SESION to PAIS
      edge USUARIO_TIENE_CUENTA from USUARIO to CUENTA
      edge VECINOS from USUARIOS to USUARIOS
)
```

11

Appendix 2. Zeus configuration file

```
;Version: 1.2.4.2
entry "StaticConfig"
;botnet "btn1"
timer_config 60 1
timer_logs 1 1
timer_stats 20 1
url_config "http://192.168.188.131/zeus/config.bin"
url_compip "http://192.168.188.131/zeus/ip.php" 1024
encryption_key "titonano"
;blacklist_languages 1049
end
entry "DynamicConfig"
url_loader "http://192.168.188.131/zeus/bot.exe"
url_server "http://192.168.188.131/zeus/gate.php"
file_webinjects "C:\Zeus\zeus\zeus 1242\webinjects.txt"
entry "AdvancedConfigs"
;"http://advdomain/cfg1.bin"
end
entry "WebFilters"
;"!*.microsoft.com/*"
;"!http://*myspace.com*"
;"https://www.gruposantander.es/*"
```

```
;"!http://*odnoklassniki.ru/*"
;"!http://vkontakte.ru/*"
;"@*/login.osmp.ru/*"
;"@*/atl.osmp.ru/*"
end
entry "WebDataFilters"
;"http://mail.rambler.ru/*" "passw;login"
end
entry "WebFakes"
"http://www.serverok.es/"
"http://www.serverhack.com/zeus/paginas/fake.php" "GP" "" ""
"http://www.serverhack.com/zeus/paginas/hola.php"
"http://www.serverhack.com/ zeus/paginas/deu.php" "GP" "" ""
end
entry "TANGrabber"
;"https://banking.*.de/cgi/ueberweisung.cgi/*" "S3R1C6G" "*&tid=*"
"*&betrag=*"
;"https://internetbanking.gad.de/banking/*" "S3C6" "*" "*" "KktNrTanEnz"
;"https://www.citibank.de/*/jba/mp#/SubmitRecap.do" "S3C6R2"
"SYNC_TOKEN=*" "*"
end
entry "DnsMap"
;127.0.0.1 microsoft.com
end
end
```

References

[1] Jaroslaw Knapik. **Data Monitor: Using technology to Combat Financial Crime in Retqal Banking**, 2005. http://gsa.ca.com/virusinfo/virus.aspx?ID=41657. 4

[2] Mika Stahlberg. **The Trojan Money Spinner**. *In: Virus Bulletin Conference*, 2007. 4

[3] F. Kim and E. Skoudis. **Protecting your web Apps**. *Working papers in Application Security*, 2008. 4, 12

[4] Nattakant Utakrit. **Review of Browser Extensions, a Man-in-the-Browser Phishing Techniques Targeting Bank Customers**. In *Proceedings of the 7th Australian Information Security Management Conference*, 2009. 4, 12

[5] R. Bolton and D. Hand. **Statistical fraud detection: A review**. *Statistical Science*, pages 235–255, 2002. 4

[6] Jaroslaw Knapik. **Data Monitor: Using technology to Combat Financial Crime in Retqal Banking**, 2005. http://gsa.ca.com/virusinfo/virus.aspx?ID=41657. 5

[7] R. Bolton and D. Hand. **Statistical fraud detection: A review**. *Statistical Science*, **17**(3):235–255, 2002. 5

[8] Mihai Christodorescu, Somesh Jha, Douglas Maughan, Dawn Song, and Cliff Wang. *Malware Detection (Advances in Information Security)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. 9

[9] Sophos. **Trojan information pages**, 2004. http://www.sophos.com/en-us//threat-center/threat-analyses/viruses- and-spyware/TrojBanker-AJ.aspx. 11

[10] CA technologies. **Virus Detail Win32.Grams.I**, 2005. http://gsa.ca.com/virusinfo/virus.aspx?ID=41657. 11

[11] Mihai Christodorescu and Somesh Jha. **Testing Malware Detectors**. In *Proceedings of the 2004 ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2004)*, pages 34–44, Boston, MA, USA, July 2004. ACM Press. 13

[12] Vinod P. Nair. **Survey on Malware Detection Methods**. *Hack in 2009*, page 74, 2009. 13

[13] Shih-Yao Dai, Yarochkin Fyodor, Ming-Wei Wu, Yennun Huang, and Sy-Yen Kuo. **Holography: a behavior-based profiler for malware analysis**. *Software: Practice and Experience*, pages n/a–n/a, 2011. 14

[14] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou. **Specification-based anomaly detection: a new approach for detecting network intrusions**. In *Proceedings of the 9th ACM conference on Computer and communications security*, CCS '02, pages 265–274, New York, NY, USA, 2002. ACM. 14

[15] Heng Yin, Dawn Song, Manuel Egele, Christopher Kruegel, and Engin Kirda. **Panorama: capturing system-wide information flow for malware detection and analysis**. In *Proceedings of the 14th ACM conference on Computer and communications security*, CCS '07, pages 116–127, New York, NY, USA, 2007. ACM. 14

[16] Yang Li and Li Guo. **TCM-KNN scheme for network anomaly detection using feature-based optimizations**. In *Proceedings of the 2008 ACM symposium on Applied computing*, SAC '08, pages 2103–2109, 2008. 14

[17] Robert P.W. Duin. **The Combining Classifier: to Train or Not to Train?** 15

[18] Arindam Banerjee and Joydeep Ghosh. **Clickstream Clustering Using Weighted Longest Common Subsequences**. In *In Proceedings of the Web Mining Workshop at the 1st SIAM Conference on Data Mining*, pages 33–40, 2001. 16, 43

[19] J. Xiao, Y. Zhang, and X. Jia. **Measuring Similarity of Interests for Clustering Web-Users**. In *Proceedings of the 12th Australasian Conference on Database Technologies*, pages 107–114, 2001. 16

[20] Monirul I. Sharif, Vinod Yegneswaran, Hassen Saïdi, Phillip A. Porras, and Wenke Lee. **Eureka: A Framework for Enabling Static Malware Analysis**. In *ESORICS*, pages 481–500, 2008. 16

[21] Robert Moskovitch, Clint Feher, Nir Tzachar, Eugene Berger, Marina Gitelman, Shlomi Dolev, and Yuval Elovici. **Unknown Malcode Detection Using OPCODE Representation**. In *Proceedings of the 1st European Conference on Intelligence and Security Informatics*, EuroISI '08, pages 204–215, Berlin, Heidelberg, 2008. Springer-Verlag. 16

[22] B. G. Ryder. **Constructing the Call Graph of a Program**. *IEEE Trans. Softw. Eng.*, **5**(3):216–226, May 1979. 16

[23] P. Vinod, Jain Harshit, K. Golecha Yashwant, Singh Gaur Maonj, and Laxmi Vijay. **MEDUSA:MEtamorfic malware Dynamic analysis Using Signature from API**. In *Proceedings of the 3rd international conference on Security of information and network*, pages 263–269, 2010. 16

[24] Cormac Herley and Dinei A. F. Florncio. **Protecting Financial Institutions from Brute-Force Attacks**. In *SEC'08*, pages 681–685, 2008. 17

[25] Harald Freeman. **VX heavens**, 2009. http://www.vx.netlux.org/. 17, 68

[26] Hispasec. **Virus Total**, 2012. https://www.virustotal.com/es/. 17, 68

[27] Christopher Williams. **Apple Flashback virus outbreak tackled**, 2012. http://www.telegraph.co.uk/technology/apple/9201908/Apple-Flashback-virus-outbreak-tackled.html. 17

[28] Integral Solutions Ltd. **Clementine**. http://www.isl.co.uk/clem.html. 18

[29] Thinking Machines Corp. **Thinking Machines Corp**. http://www.think.com/html/products/products.htm. 18

[30] SAS Institute. **Enterprise Miner**, 2012. http://www.sas.com/software/components/miner.html. 18

[31] IBM. **Intelligent Miner for Data**, 2011. http://www.powershow.com/view/1750fc-NDQwO/iMiner$_Introduction_flash_ppt_presentation.$ 18

[32] Dean W. Abbott, I. Philip Matkovsky, John F. Elder, and IV. **An Evaluation of High-end Data Mining Tools for Fraud Detection**. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 12–14. IEEE, 1998. 18

[33] First. **E-LC CSIRT**, 2011. http://www.first.org/members/teams/. 18

[34] Lorenzo Martignoni, Roberto Paleari, and Danilo Bruschi. **A Framework for Behavior-Based Malware Analysis in the Cloud**. In Atul Prakash and Indranil Sen Gupta, editors, *Information Systems Security*, **5905** of *Lecture Notes in Computer Science*, pages 178–192. Springer Berlin / Heidelberg, 2009. 10.1007/978-3-642-10772-6$_1$4. 19

[35] Stephanie Forrest, Steven Hofmeyr, and Anil Somayaji. **Intrusion Detection Using Sequences of System Calls**. *Journal of Computer Security*, **6**(3):151–180, 1998. 27

[36] Sean Peisert, Matt Bishop, Sidney Karin, and Keith Marzullo. **Analysis of Computer Intrusions Using Sequences of Function Calls**. *IEEE Trans. Dependable Secur. Comput.*, **4**:137–150, April 2007. 27

[37] T. Srivastava, P. Desikan, and V. Kumar. **Web Mining Concepts, Applications and Research Directions**. pages 275–307. 2005. 33

[38] SecureWorks. **Gozi Trojan**, 2012. http://www.secureworks.com/research/threats/gozi/. 34, 66

[39] Varun Chandola, Arindam Banerjee, and Vipin Kumar. **Anomaly Detection: A Survey**. *ACM Computing Surveys*, 2007. 35

[40] Paul Barford and David Plonka. **Characteristics of network traffic flow anomalies**. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, IMW '01, pages 69–73, New York, NY, USA, 2001. ACM. 38

[41] Alexander Strehl, Er Strehl, Joydeep Ghosh, and Raymond Mooney. **Impact of Similarity Measures on Web-page Clustering**. In *In Workshop on Artificial Intelligence for Web Search (AAAI 2000*, pages 58–64. AAAI, 2000. 43

[42] Yue Xu and Li-Tung Weng. **Intelligent information processing II**. chapter Improvement of web data clustering using web page contents, pages 521–530. Springer-Verlag, London, UK, UK, 2005. 43

[43] Xuanhui Wang, Dou Shen, Hua-Jun Zeng, Zheng Chen, and Wei-Ying Ma. **Web page clustering enhanced by summarization**. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, CIKM '04, pages 242–243, New York, NY, USA, 2004. ACM. 43

[44] Morteza Haghir Chehreghani, Hassan Abolhassani, and Mostafa Haghir Chehreghani. **Improving density-based methods for hierarchical clustering of web pages**. *Data Knowl. Eng.*, **67**(1):30–50, October 2008. 43

[45] Michael Hahsler, Bettina Grn, and Kurt Hornik. **arules - A Computational Environment for Mining Association Rules and Frequent Item Sets**. *Journal of Statistical Software*, **14**(15):1–25, 9 2005. 44

[46] Agmed Moustafa. **JAligner: Open source Java implementation of Smith-Waterman**, 2010. http://jaligner.sourceforge.net/. 44

[47] Mohammed Javeed Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, and Wei Li. **New Algorithms for Fast Discovery of Association Rules**. In *In 3rd Intl. Conf. on Knowledge Discovery and Data Mining*, pages 283–286. AAAI Press, 1997. 47

[48] P. S. Dodds and D. J. Watts. **A generalized model of social and biological contagion**, 2005. 47

[49] Dino Esposito. **Browser Helper Objects: The Browser the Way You Want It**, 1999. http://msdn.microsoft.com/en-us/library/bb250436%28v=vs.85%29.aspx. 55

[50] David Gloser. **Remote BHO Scanner**, 2005. http://remotebhoscan.sourceforge.net/. 58

[51] Parvez Anwar. **Hiding Browser Helper Objects**, 2009. http://www.greyhathacker.net/?p=106. 58

[52] Liam O Murchu. **Trojan.Silentbanker**, 2008. http://www.symantec.com/security_response. 66

[53] Ali Ikinci, Thorsten Holz, and Felix Freiling. **Monkey-Spider: Detecting Malicious Websites with Low-Interaction Honeyclients**. In *In Proceedings of Sicherheit, Schutz und Zuverlssigkeit*, 2008. 67

[54] Honeynet Project. **Diaonea**, 2012. http://dionaea.carnivore.it/. 67

[55] Malware URL Team. **Fighting Malware and Cyber Cryminality**, 2012. http://www.malwareurl.com. 67

[56] Joaquin Moreno. **Security ArtWork:Mail Malware Trap**, 2011. http://www.securityartwork.es/2011/11/28/presentamosmailmalwaretrap/. 67

[57] Georgia Tech Information Security Center. **Offensive Computing**, 2011. http://www.offensivecomputing.net/. 68

[58] Cyveillance. *Malware Detection Rates for Leading AV Solutions*, 2010 (accessed December 10 , 2011). `http://www.cyveillance.com/web/docs/WP_MalwareDetectionRates.pdf`. 75

[59] Sverdlove Harry. **The Most Vulnerable Smartphones of 2011**, 2011. 82

[60] Neal Leavitt. **Mobile Phones: The Next Frontier for Hackers?** *Computer*, **38**(4):20–23, April 2005. 82

[61] E. E. Schultz. **Where have the worms and viruses gone? New trends in malware**, 2006. Comput. Fraud Security, vol. 2006, no. 7, pp. 4-8. 82

[62] Dhawal Desai. **Malware Analysis: ZitMo Trojan for AndroidOS**. *Kindsight Report*, 2011. 83

[63] Iker Burguera, Urko Zurutuza, and Simin Nadjm-Tehrani. **Crowdroid: behavior-based malware detection system for Android**. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, SPSM '11, pages 15–26, New York, NY, USA, 2011. ACM. 83

[64] Chikeung Luk, Robert Cohn, Robert Muth, Harish Patil, Artur Klauser, Geoff Lowney, Steven Wallace, Vijay Janapa, and Reddi Kim Hazelwood. **Pin: Building customized program analysis tools with dynamic instrumentation**. In *In Programming Language Design and Implementation*, pages 190–200. ACM Press, 2005. 85

[65] Nicholas Nethercote and Julian Seward. **Valgrind: a framework for heavyweight dynamic binary instrumentation**. In *Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation*, PLDI '07, pages 89–100, New York, NY, USA, 2007. ACM. 85

[66] Chien Eric Falliere Nicolas, Murchu Liam O. **W32.Stuxnet Dossier**. *Symantec Report*, 2011. 89

[67] Alfonso Valdes and Steven Cheung. **Communication Pattern Anomaly Detection in Process Control Systems**. In *2009 IEEE International Conference on Technologies for Homeland Security*, Waltham, MA, May 11–12, 2009. 89