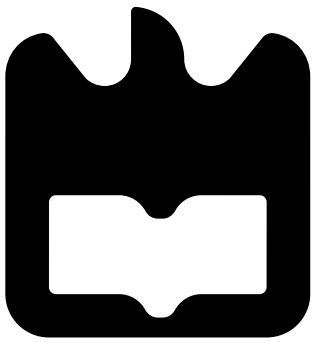
University of Aveiro
2010

Departamento de
Electrónica, Telecomunicações e Informática
Instituto de Telecomunicações de Aveiro

Luís Póvoa

Correlação de informação de rede
Correlation of network information



University of Aveiro
2010

Departamento de
Electrónica, Telecomunicações e Informática
Instituto de Telecomunicações de Aveiro

Luís Póvoa

Correlação de informação de rede
Correlation of network information

"Security is, I would say, our top priority (...) if we don't solve these security problems, then people will hold back."

— Bill Gates

Luís Póvoa

Correlação de informação de rede
Correlation of network information

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Computadores e Telemática, realizada sob a orientação científica de Prof. Dr. António Nogueira, Professor Auxiliar do Departamento de Electrónica Telecomunicações e Informática da Universidade de Aveiro e do Prof. Dr. Paulo Salvador, Professor Auxiliar do Departamento de Electrónica Telecomunicações e Informática da Universidade de Aveiro

o júri / the jury

presidente / president	**Prof. Dr. Atílio Gameiro** Professor Associado da Universidade de Aveiro
vogais / examiners committee	**Prof. Dr. António Nogueira** Professor Auxiliar da Universidade de Aveiro (orientador)
	Prof. Dr. Paulo Salvador Professor Auxiliar da Universidade de Aveiro (co-orientador)
	Prof. Dr. Joel Rodrigues Professor Auxiliar da Universidade Beira Interior

agradecimentos / acknowledgements

Agradeço à minha família por todo o apoio dado ao longo da minha vida académica. Também à minha namorada, amigos e colegas pela ajuda nos bons e maus momentos.

Um agradecimento especial ao Daniel Carvalho, por toda a ajuda, paciência e conhecimento que demonstrou sempre que precisei.

Finalmente não posso deixar de agradecer aos meus orientadores por toda a ajuda e constante disponibilidade, Professor António Nogueira e Professor Paulo Salvador.

Resumo

Nos dias de hoje, é impossível para empresas ou para um utilizador comum de computador, trabalhar sem internet, pesquisar, comunicar, partilhar informação . . .

Todos os dias, novas ameaças são criadas e podem afectar milhões de computadores e centenas de empresas numa questão de minutos ou horas. As soluções existentes ainda dependem da intervenção Humana e análise de tráfego, o que significa que existe sempre mais margem para erro e falhas.

Tudo muda rapidamente e a informação tende a aumentar mais e mais, tanto em redes privadas como na Internet, seja essa informação sensível ou não. Então, toda esta informação deve ser automaticamente reunida e processada, deixando o mínimo possível de responsabilidade para o Administrador de rede.

Esta dissertação pretende lembrar que a segurança não deve nunca ser vista como uma tarefa secundária. E com este princípio em mente, o principal objectivo é criar um sistema, fácil de usar, instalar e de administrar, para obter a informação disponível de servidores e equipamentos de toda a rede. A informação recolhida pode ser depois utilizada para detectar anomalias de rede e deve também ter a capacidade de implementar as contra-medidas mais apropriadas .

Abstract

Nowadays, it is impossible for companies or a common computer user, to work without Internet, searching, communicating, sharing information...
Every day new threats are created and can affect millions of computers and hundreds of companies in a matter of minutes or hours. Existing solutions still depend on Human intervention and network traffic analysis, which means they are open to breaches.
Everything changes faster, runs faster, and information tends to increase more and more, both in private networks and on the Internet, whether it is sensitive or not. So, all this information should be automatically gathered and processed, letting as less as possible behind the responsibility of the Network Administrator.
This dissertation intends to remember that security should never be seen as a secondary task. Having this principle in mind, the main purpose was to create a system, easy to use, install and administrate in order to gather information from the servers and equipments of the entire Network. Collected information can then be used to detect network anomalies and the system should also have the ability to deploy them most appropriate countermeasures.

Contents

List of Figures

Acronym

IDS Intrusion Detection Systems

GUI Graphical User Interface

IPS Intrusion Prevention Systems

MAC Media Access Control

IP Internet Protocol

OS Operating System

SSH Secure Shell

mIRC Microsoft Internet Relay Chat

IRC Internet Relay Chat

ACID Analysis Console for Intrusion Databases

BASE Basic Analysis and Security Engine

CRONTAB Time-based job schedule table

CRONJOB Time-based job schedule job

SNORT SNORT - open source network intrusion prevention system

PHP Hypertext Preprocessor

MySQL My Structured Query Language

ASP Active Server Pages

Pearl Process and Experiment Automation Realtime Language

P2P Peer to Peer

IRCBot Internet Relay Chat Bot

AgoBot Axel "Ago" Gembe Bot

HTTP HyperText Transfer Protocol

TCP Transmission Control Protocol

UDP User Datagram Protocol

SMTP Simple Mail Transfer Protocol

HTTPS HyperText Transfer Protocol secure

ICMP Internet Control Message Protocol

WMI Windows Management Instrumentation

ADSI Active Directory Service Interfaces

MS SQL Microsoft Structured Query Language

PC Personal Computer

PCs Personal Computers

ICMP Internet Control Message Protocol

SMTP Simple Mail Transfer Protocol

NTDS NT Directory Service

POP3 Post Office Protocol

ODBC Open Data Base Connectivity

IOS Internetwork Operating System

IDSM Intrusion Detection System Module

SQL Structured Query Language

ISA Internet Security and Acceleration

VBscript Visual Basic Script

DNS Domain Name System

IBM International Business Machines

DLP Data Loss Prevention

LTS Long Term Support

FTP File Transfer Protocol

SNMP Simple Network Management Protocol

SEO Search Engine Optimization

DDoS Distributed Denial-of-Service

VOIP Voice Over IP

IEEE Institute of Electrical and Electronics Engineers

HW Hardware

IPsec Internet Protocol Security

ARP Address Resolution Protocol

LAN Local Area Network

CGI Common Gateway Interface

PDA Personal Digital Assistant

USB Universal Serial BUS

MSN Microsoft Network Messenger

NFS Network File System

AV Anti-Virus

NTFS New Technology File System

GTbot Global Threat bot

DLLs Dynamic Link Libraries

DSNX Data Spy Network X

VLAN Virtual Local Area Network

MTU Maximum Transmission Unit

PCRE Pearl Compatible Regular Expressions

API Application Programming Interface

POSIX Portable Operating System Interface

TTL Time to Live

Chapter 1

Introduction

1.1 Why Security is Hard

Traditional antivirus software based on packet filtering, port-based, and signature-based techniques do not effectively mitigate botnets that dynamically and rapidly modify the exploit code and control channel. We should never think that it is possible to prevent everything, so it is a good idea to assume that attacks will happen anyway, and we should be concerned with detecting and solving them.[1]

Most of the affected systems run the Microsoft Windows Operating System (OS), but that is not the main reason for vulnerability. The problem is that systems are not frequently updated, patched or properly secured behind firewalls, being an easier target for worms, Trojan horses or backdoor exploits. Most of the successful bots have relied upon exploiting vulnerabilities that have been already corrected by patched systems. In fact, what happens is that bot-masters find exploits by waiting for the vendors patches for a specific vulnerability and reverse engineering them, in order to find their flaws and take advantage of them.[2] In addition, a typical computer user can not detect the majority of the bot activity. So, unless the user has a network traffic capture tool (packet sniffer), the bot will be invisible to the victim and its threat will obviously be difficult to eliminate.

Botnet creation begins with the download of a software program called a bot (for example, Internet Relay Chat Bot (IRCBot), SGBot, or Axel "Ago" Gembe Bot (AgoBot)), along with an embedded exploit (or payload) by an unsuspect user, who might click on an infected e-mail attachment or download infected files or freeware from peer-to- peer (Peer to Peer (P2P)) networks or malicious Websites.

Once the bot and exploit combination are installed, the infected machine contacts a public server that the botmaster has set up as a control plane to issue commands to the botnet. A

common technique is to use public Internet Relay Chat (IRC) servers, but hijacked servers can also issue instructions using HyperText Transfer Protocol (HTTP), HyperText Transfer Protocol secure (HTTPS), Simple Mail Transfer Protocol (SMTP), Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) strings. Control planes are not static and are frequently moved to evade detection; they run on machines that are never owned by the botmaster.

Using the control plane, the botmaster can periodically push out new exploit code to the bots. It can also be used to modify the bot code itself in order to evade signature-based detection or to accommodate new commands and attack vectors.

Initially, however, the botmasters primary purpose is to recruit additional machines into the botnet. Each zombie machine is instructed to scan for other vulnerable hosts. Each new infected machine joins the botnet and then scans for potential recruits. In some hours, the size of a botnet can grow very large, sometimes comprising millions of Personal Computers (PCs) on diverse networks around the world [3].

One of the main ways to get infected is through web browsers, not only because they have flaws, but also because they have low security features. It is often called a "low cost" way, since it will start being affected only by visiting the site (following a link from an email, or simply a search result on the web). Additionally, there are other infection mechanisms:

- ActiveX: used by Microsoft, it allows applications or parts of applications, to be used by web browsers, giving them a freedom that they definitely shouldn't have.
- Java: object-oriented programming language, where a Java Virtual Machine is used to execute the code, usually called "sandbox", and the interaction with the rest of the system is limited. However, various implementations have vulnerabilities.
- Plugins
  Add-ons: similar to ActiveX controls but specific to the: Mozilla Firefox and Safari web browsers, they can have design and programming flaws, like buffer overflow, cross-domain violations etc.
- JavaScript: scripting language that makes web sites more interactive but can also reveal security flaws.
- Cookies: contain information that is needed for each website and also contains credentials for accessing it; persistent cookies remains on the computer until their expiration date and if some attacker obtains the cookie for authentication, it can gain access to the victim account.

- VBScript: not widely used as JavaScript because of is limited compatibilities with other browsers rather than Microsoft Internet Explorer. [4]

Why do anti-virus fail on securing a system or network? Because botnets are dynamically updated and controlled, the strategies used by most anti-virus companies fail because once they update the signature for a specific bot, which identifies one of its variations, the bot controller updates the bot to change the signature, which would force anti-virus software to have a wide array of botnet class variations. So, it is difficult to develop defenses for each particular case. Thus, it is important to create a flexible and structured architecture that can be updated and has classes of countermeasures for each type of bot, and not only for each particular bot or signature.

Securing a system and protecting a network should not be a unique layer of security, instead the network and system administrators should always get together many levels of security, in many areas: network monitoring, server logs, router and switch information. It's fundamental to correlate as much information as possible to get the conclusions that are needed to detect and stop malware and botnets.

Some Malware Numbers

Sophos, a security company that provides a full range of endpoint, encryption, email, web and Network Access Control solutions, says that the Linux and Mac markets are growing and there are also some threats targeting Linux (0.1%) and Mac Systems. And the problem is not only to be the victim but also transmitting malware to Windows computers.

Around 60% of the web servers run Linux/Apache and most of the financial, commercial server applications also run on Linux. Although the threat is not so high at the present, the risk is growing and the consequences can be very serious because a high number of Windows Machines are Personal Computers and not Servers, unlike Linux OS's.

Non-Windows platforms will become more attractive to virus and malware writers that will try to infect more servers and machines with sensitive information. So, the solution resides in a "cross-platform" security system.

Table 1.1 shows the estimated number of Bots and their corresponding SPAM capacity, while Table 1.2 shows the percentage of infection attempts per country.

Name	Estimated no. of Bots	Spam capacity
Conficker	10.000.000+	10billion/day
Kraken	495.000	9 billion/day
Srizbi	450.000	60 billion/day
Bobax	185.000	9 billion/day
Rustock	150.000	30 billion/day
Cutwail	125.000	16 billion/day

Table 1.1: Estimated number of Bots

Table 1.2 shows countries with the higher percentages of infection attempts via web:

Country	Percentage
China	36.2%
Russia	5.8%
USA	4.4%
India	3.9%
India	3.9%
Germany	3.9%
Egypt	3.2%
Mexico	2.9%
Great Britain	2.4%
France	2.3%
Turkey	2.2%
Others	32.8%

Table 1.2: Percentage of infection attempts per country

1.1.1 What is a botnet?

Bot is the short term for *robot*. Normally the term *botnet* is used to refer to any group of bots; nowadays it is being used mostly to refer to a group of infected machines that are controlled by a *bot master*. Usually these machines are controlled via IRC or encrypted IRC channels, and they have capabilities to spread and infect their neighbors in the network.

If the BotMaster has programmed his bot clumsily, the victim might sense his computer is slower and the connection to the network or the internet has slowed down, due to scans in the network looking for new victims, spreading malicious code or simply executing some kind of attack. On other cases, the botnet can be more competent and put the machine working only when noticing some idle period on it, so it can stay invisible for a normal user whose computer has been "hijacked".

Figure 1.1 illustrates a Classic Botnet Architecture, with Command And Control (C&C) and updates based on Microsoft Internet Relay Chat (mIRC) connections.

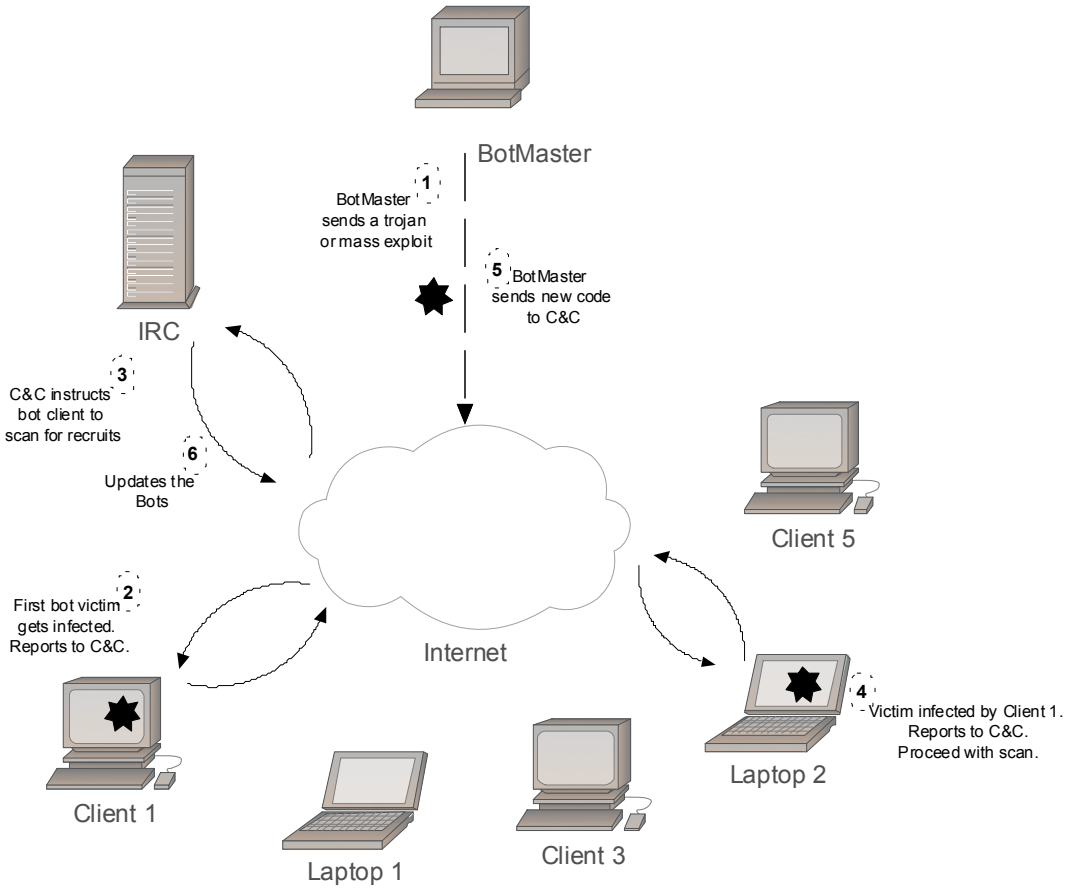


Figure 1.1: Classic Botnet Architecture

There are other similar structures with the only difference of using an HTTP server instead of the mIRC server the Botnet functions with a HTTP server. These botnets are harder to detect because HTTP connections are more common in a network; so, it is harder to distinguish botnet communications from regular traffic.

Storm P2P is even harder to detect, due to its complex structure and working operation: non centralized command and control, so any infected Storm machine can act as the C&C at the botmaster's will, as we can see in Figure 1.2.

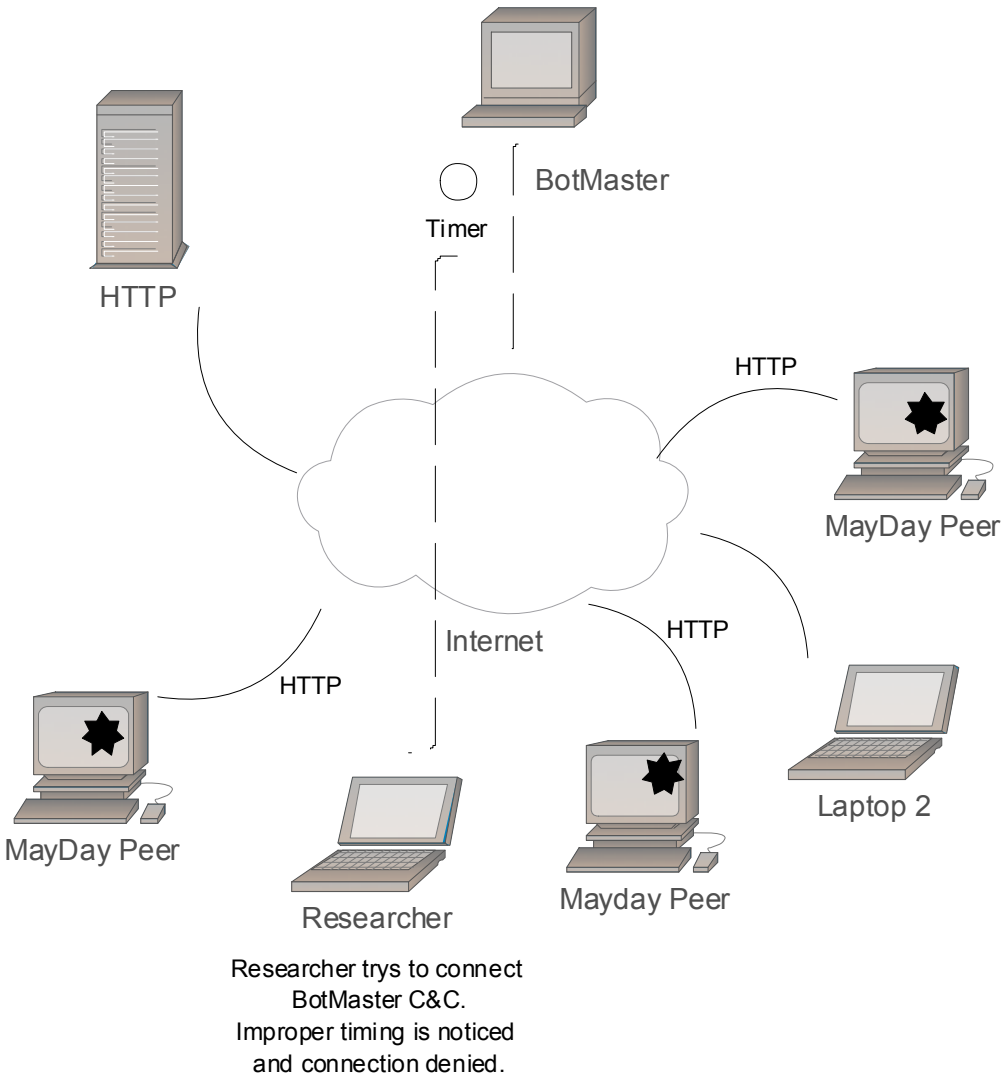


Figure 1.2: Storm P2P Architecture

An "evolution" of the above architectures appeared at the beginning of 2008, known as MayDay, having new ways to get rid of researchers and "bot hunters":

- Communicates between bot clients and the command center using standard HTTP and (even trickier) using proxies.
- Can use P2P communication, if HTTP fails. Can also use Internet Control Message Protocol (ICMP) (encrypted) messages, ports and protocols that administrators dare not to block.
- Have limited C&C traffic, since this botnet has new features that avoid researchers to

discover and disable bots and botnet: it throttles how much traffic passes between the control center and the clients, enforces short window communications between them, and maintains a small number (compared to mostly known botnets) of bots.

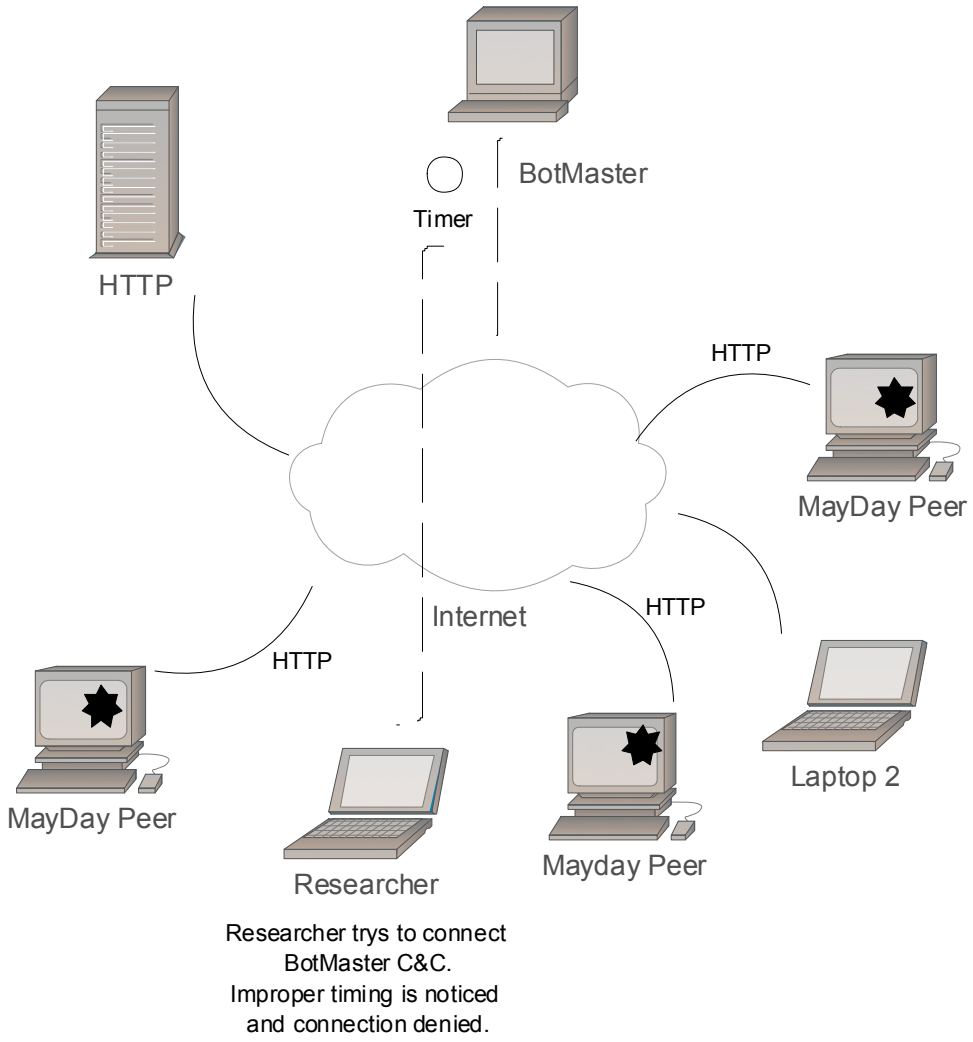


Figure 1.3: MayDay Architecture

1.1.2 Why is it a threat?

It is called the ultimate threat. An army of compromised machines, also known as "zombies", that are commanded by a single botmaster.[3]

The bots can be optimized and can change behavior over time, because the botmaster can command the functions of all compromised machines at once or individually, and upload

a new code for spam the network, instructions to steal banking information, record login credentials and always report to some kind of server that is owned by botmaster.

Due to their size and growing speed, botnets pose serious security threats. Even a small botnet. Let's see, using a simple example, that only 1000 bots can cause a great deal of damage. These 1000 bots have (average of 128Kbit/s) a combined bandwidth of 100Mbit/s, which is, nowadays, a usual Company Internet speed connection; in addiction, the Internet Protocol (IP) distribution is too wide and dificult to filter or control. A botnet could accomplish several threats:

- Distributed Denial-of-Service Attacks: an attack on a computer system or network that causes a loss of service to users, consuming bandwidth or overloading the computational resources of the victim system. It is widely used to take down competitor's websites, but also other services.

- Spamming: phishing or simple spam, it is usually sent by botnets and can be used for nefarious tasks, such as collect email-addresses.

- SYN Flood: the origin of this attack is difficult to identify and it could be launched to specific hosts or against routers or other network systems that will turn on other TCP services. The attacker could force (servers) to open "half-connections", because each SYN begins the acknowledgment process to create new connections, and each one is stored on a finite size structure; obviously this structure will overflow by intentionally creating partially-open connections. The "real" client, and also the victim, will never receive the final ACK message.

- Sniffing Traffic: clear-text data passing by a compromised machine, is mostly used to retrieve usernames, passwords, and even more interesting, it is commonly used to steal the key information of other botnets in order to control them.

- Keylogging: keyloggers are the easiest way to get sensitive information, and they can be multiplied by thousands of infected machines.

- Spreading new malware: this integrated feature of most botnets is used to send and spread the infected file or the malicious code to other victims, via email, File Transfer Protocol (FTP), HTTP, Microsoft Network Messenger (MSN)...

- Google AdSense abuse: the attacker has a lot of valid IP addresses and the "company" earns money each time the add is clicked;

- Attacking IRC Chat Networks: the idea of this attack is to flood the IRC network of the victim, triggering a similar Distributed Denial-of-Service (DDoS) attack.

- Manipulating online polls: because the main way to control online polls is based on the IP address, botnets have a particular easy task on manipulating them because all "zombies" have a different IP.

- Mass identity theft: fake websites pretending to be online banking, PayPal, Ebay, etc, ask the vitims to enter their personal and private information just to gather all important and confidential info .

- Sensitive information theft: suppose that a company is developing some new product and does not want anyone to know its projects; a botnet can steal and publish everything, or even worst, send the ideas and projects to the wrong hands.

- Change Domain Name System (DNS): this is an easy way to get the victim right were the attacker wants to. Usually few people takes care of the "home" router: proper password, secure connections, etc. The Botnet just has to look for "default" passwords and try to connect to the router, then changes the DNS address to "its" DNS server and just redirect every HTTP request to its fake pages instead of the real ones.

- Certainly an Bot or a Infected Client will spread itself in many forms, depending on the botnet type or technology.

- It will download and execute files, put processes running on the victim's machine, as well as update and trigger new orders from the C&C.

Worm, Virus, Trojan, Bot: Differences

Botnets blend many malware technologies, describing them can be tricky exercise. Let us take a look at the most important security threats:

- Worm: an executable program that can spread itself. A Worm can spread a payload that constitutes the bot code;

- Trojan: usually a trojan works alone, and just reports to the creator; unlike bots, it cannot respond to the command center and is not flexible;

- Virus: unlike, malware, adware, spyware, etc, a true virus only has the capability of spreading from a computer to another and somehow execute his code, usually via Universal Serial BUS (USB) drive, cd, dvd or network file system;

- Botnet:

  - Can run on a victim's computer
  - Can connect back to a remote control center

- Can receive and respond to commands from the control center
- Is part of a system automated to control many agents simultaneously
- Deploys without the victim knowledge or permission [2]

1.1.3 Botnet life cycle

The most important phases of the botnet life cycle are [5]:

- **SpreadPhase:** In this phase, the bots propagate and infect systems, trough SPAM, e-mails, web worms or malware downloads.

  The goal of the spread phase is to infect a system for the first time, bot-heritors attempt to either exploit vulnerabilities on the user system, using applications or browsers, or cheating the user by installing the malware payload.

- **Infection Phase:** Once in the system, the malware infects and obfuscates its presence (polymorphism), that is, the code changes any time there's a new infection, making it harder to discover the true nature of the malware and analyzing it using anti-virus.

  The malicious software (known as rootkit) activates each time the system boots up. It usually uses techniques to actively target anti-virus, services, local firewalls and Intrusion Prevention Systems (IPS) / Intrusion Detection Systems (IDS); for instance, the botnet frequently changes local DNS settings to unable anti-virus software reach the update site.

  It is also common that advanced botnets present a false image of memory or hard disk to the anti-virus to delude vulnerability scanners.

- **Command and Control:** IRC, encrypted P2P (eDonkey/Overnet), HTTP, Voice Over IP (VOIP). These are some techniques used by botmasters to control their "zombies". VOIP is still on the horizon but it is being "studied".

  Modern botnets also re-program or update nodes, using these communication channels. HTTP and P2P are particularly difficult to detect because these are very common network protocols:

  - HTTP - uses forums and newsgroups where infected computers can anonymously check for messages that will be used as C&C (Command&Control).
  - IRC - the most common way to communicate, using messages trough channels, private messages or even the title of the channel. Changing channels is usually used to detect when some node is connected just to study the behavior and gather information, for example a "spy" that wants to disarm the bot.

  - P2P - peer-to-peer is commonly encrypted, so it is hard to figure out what is going on under the C&C.

- **Attack Phase:** perform massive distributed denial-of-service (DoS) attacks to corporate systems, government or other botnets.

1.2 Detection of Botnets

1.2.1 Prevent it

Network-based intrusion prevention systems can't be the only option to defeat botnets. In order to get a secure network and secured systems, it is mandatory to have a multi layered security system:

- Antivirus, antispyware and host-based firewalls: should be included to avoid getting infected in a first phace.

- Network-Based Intrusion Prevention Systems: usually blocks the most popular bots (blocking the botnet control traffic).

- Host-Based intrusion prevention: important layer, limits the various applications running on the system, but it misses the "trusted ones" and grant them freedom to communicate with the exterior, so there is still a problem if the botnet code hides on them.

- Up-to-date patches: centralized patching techniques are recommended.

- Unable web browsers scripting support: blocking JavaScript, ActiveX and other Add-ons.

- Monitor Ports, like:

  - IRC ports: 6667, high-numbered ports (31337 and 54321). In general, all ports above 1024 should be blocked for inbound and outbound traffic, unless special applications need these ports open.

  - Ports like 80 or 7 are used for updates and communication: zombies commonly "wait" for 1:00 a.m. until 5:00 a.m for updates, because they suppose there is no one watching, so checking the network logs should be a good principal; if there is no one working in the company during the night, a high traffic in this period should be suspicious. [2]

1.2.2 Detect it

As in Psychology, the first step is to acknowledge that we have or could have a problem. The idea that the prevention makes everything should not be taken as enough and unique. So, in parallel with prevention, it is important to register relevant information: log files, network traffic, statistics...

It is also important to make network testing routines, including: routers, firewalls, web servers, e-mail servers and all other systems that are accessible to the public or simply are critical for the organization. When doing these types of tests, security systems should be aware of it in order to avoid or ignore false positives.

Classic Botnet Architecture have a particular weakness, which makes them the easiest class of botnets to catch and disable, because all bots are susceptible to a single point of failure. They always use IRC ports and then generates a high traffic volume on them.

1.2.3 Cancel it

There are no magical formulas and each case is different from the others, anyway, there are, if they don't cause troubles for a very specific reason to the company, a couple of measures that can be taken in order to avoid problems:

- Close IRC ports: the majority of current botnets use IRC ports for messaging and commands.
- Close P2P ports: used many times for messaging, updates and share code.
- Prevent users from installing P2P software, such as Kazaa, Grokster, Bear Share, etc, since they have vulnerabilities and downloads can bring virus, worms...
- Search for windows registry entries with botnet names.
- Run cleaning software, searching for worms, virus... Special attention to these applications, many of them are "false" cleaners and will fill the system with even more problems instead of cleaning them.
- Use a good IDS / IPS system (described later on this thesis).

1.3 Motivation

Today, Botnets, are the main threat for organizations, government, banking and common user. So, it is a priority to prevent, detect and cancel these threats. Recently, and ironically,

in a Security Conference sponsored by International Business Machines (IBM), that took place on the Australian Gold Coast between 16 and 21 of May, many companies were selling security products and, unfortunately, one of them offered infected USB drives.

Each day sees the emergence of more and more tools to create and control these *nets*, so special awareness should exist for these new threats and new technologies, even if there is no sensitive information on the network. Although Linux and Mac OS are also affected, as we can see later on this work, Microsoft Windows are the most affected systems since they are also the most used at home and companies computers.

The high number of possible services, equipments and PCs on a Local Area Network (LAN) or the Internet allows different types of attacks with different characteristics and different consequences. So, it is important to gather information from network servers (Apache, FTP, DNS, Post Office Protocol (POP3), SMTP, Network File System (NFS), SAMBA, etc); system logs can also provide important information for analysis (UNIX: syslog; Windows: windows registry).

All services mentioned above, uses a file based method for logging. These files are distributed on each server, and because of that, it is very difficult to analyze and correlate information. So, it is important to gather and centralize the information, having always in mind that these files provides important and sensitive data that should not be accessed by anyone else but network administrators or system administrators.

After gathering all the information, it is necessary to parse and process it. And, finally, store it on a database, the same approach that will be used in this work. The information on the database should be managed through a web interface: this solution was also chosen in this work because it is centralized, fast, provides great integration with already existent technologies and a good possibility to extend its functionalities.

Websense Security Labs Predictions

Websense Security Labs anticipates the emergence and growth of the following trends:

- Web 2.0 attacks will increase in sophistication and prevalence

  In the coming year, Websense Security Labs predicts a greater volume of spam and attacks on the social Web and real-time search engines such as Topsy.com, Google and Bing.com. In 2009, researchers have seen and increasing malicious use of social networks and collaboration tools, such as Facebook, Twitter, MySpace and Google

Wave to spread attackers' wares. Spammers and hackers use of Web 2.0 sites have been successful because of the high level of trust users place on these platforms and on other users.

- Botnet gangs will fight turf wars

  It is possible to anticipate more aggressive behavior between different botnet groups that will try to deploy bots with the ability to detect and actively uninstall competitor bots.

- Email gains traction again as a top vector for malicious attacks

  Email will be used as a vector for spreading malicious attacks and will evolve in sophistication and we will see more emails containing malicious data stealing attachments and malicious URLs.

- Targeted attacks on Microsoft properties, including Windows 7 and Internet Explorer 8

  With the expected fast adoption of Windows 7, more malicious attacks targeting the new OS will be seen with specific tricks to bypass User Access Control warnings, and greater exploitation of Internet Explorer 8.

- Don't Trust Your Search Results

  A malicious Search Engine Optimization (SEO) poisoning attack, also known as a Black-hat SEO attack, occurs when hackers compromise search engine results to make their links appear higher than legitimate results. As a user searches for related terms, the infected links appear near the top of the search results, generating a greater number of clicks to malicious Web sites.

- Smartphones are the hackers' next playground

  Smartphones such as the iPhone and Android, which are increasingly used for business purposes, are essentially miniature personal computers and in 2010 they will face the same types of attacks that target traditional computing. Additionally, poor security for applications on smartphones can put users and organizations data at risk.

- Why corrupt a banner ad serve, when you can buy malvertising space?

  In a high-profile incident in 2009, visitors to the New York Times Web site saw a pop-up box warning them of a virus that directed them to an offer for antivirus software, which was actually rogue Anti-Virus (AV). This attack was served up through an advertisement purchased by someone posing as a national advertiser.

- 2010 will prove once and for all that Macs are not immune to exploits

Hackers have noticed Apple's rapid growth in the market share in both the consumer and corporate segments. There exists additional risk for Mac users because many assume Macs are immune to security threats. There is also the potential for the first drive-by malware created to target Apple's Safari browser.

The dynamic nature of Web 2.0 attacks, the use of email to drive users to malicious Web sites, and tactics like SEO poisoning and rogue AV, all demonstrate the need for organizations to have a unified content security platform that protects against blended Web, email and data security threats."[6]

1.4 Objectives and Contributions of this work

The tasks that were carried out in this Dissertation are included in one of the main goals of our research group: to develop a botnet detection system that will be based on gathering network information, correlating it, processing the most important variables and lauching system alerts and counter measures.

Our particular work aims to gather all the available information on the network that can be useful for the identification problem, keeping also track of where the information comes from. A web interface was created to administrate and configure the information gathering process, enabling also to configure equipments, administrators, snort rules, visualize logs files that were acquired from network servers or equipments.

Another important feature is the possibility to create and configure variables that will facilitate classifying information and detecting network anomalies. It will monitor the server logs of e-mail, web, FTP and other servers. The designed system should be flexible enough allow the introduction of new functionalities and the creation (later on the project) of counter measures, system alerts, etc.

One of the objectives of this work is to turn more flexible and adapt the *shell scripts* that were already created to gather network information [7]. All information is stored in a My Structured Query Language (MySQL) database, as well as the shell script codes, so it is possible to execute and edit them more easily.

1.5 Organization of this thesis

This thesis is devided into four main chapters. Each one briefly describes the work that was done in the different phases of the research, implementation and evaluation of the archi-

tecture.

This chapter contextualizes the thesis objectives, contributions and main purpose. It also briefly explains security aspects and why security at this level is so hard to implement. Finnally, it discusses some of the existing security problems and threats, focusing on Botnets.

The second chapter describes the current state-of-the-art developments in this field. An overview of already developed tools or solutions: freeware, opensource or proprietary software/hardware.

The third chapter presents the architecture of the proposed system, describing its main features and requirements, as well as a brief "Installation and Use" guide of the implemented system.

The last chapter describes possible work that can be done after this thesis, resulting from an evaluation of the deficiencies and possible improvements that could be made to the implementation, as well as new features that can and should be added.

Chapter 2

Related Work

2.1 Introduction

Gathering information from the network and its equipments became one of the most important procedures for institutions and companies. Some of the most important reasons for that are:

- The size of current company networks and the traffic increase.
- The co-existence of several heterogeneous equipments in the same network and the need for interaction between them.
- Economical concerns, advocating that human intervention should be the last "layer" to solve the problem.

A good monitoring of the network is needed in order to grant security. Sometimes, companies thinks they are not big enough to have a network administrator or even a good and well defined security policy. This situation should change, because the attacks are no longer an exclusive of big companies with a high number of enemies or competitors attacks are not even made by a single "hacker" alone in front of a computer - they are usually carried out by an army of already infected computers (called zombies) and the main objective is to gather as much machines as possible so, the hacker, or in some cases organized groups of hackers, can define a single target and the attack objectives.

Security should not be an isolated process, layer or act of goodwill, it is fundamental to think about security as a continuos process, improving existing technologies, upgrading equipments, updating systems, software and implementing or improving policies. This should minimize the risk of attacks, infections, viruses, etc..., which are the main causes security failures.

So, it is crucial to have mechanisms that can monitor and gather information from all the network, then filter and process it in order to show the network administrator what is really relevant.

Table 2.1 shows some of the existing botnets and describes some of their features and capabilities .

Type	Features
Agobot Phatbot Forbot Xtrembot	They are so prevalent that over 500 variants exist in the Internet today. Agobot is the only bot that can use other control protocols besides IRC. It offers various approaches to hide bots on the compromised hosts, including New Technology File System (NTFS) Alternate Data Stream, Polymorphic Encryptor Engine and Antivirus Killer.[8][9]
SDBot RBot UrBot UrXBot	SDBot is the basis of the other three bots and probably many more. Different from Agobot, its code is unclear and only has limited functions. Even so, this group of bots is still widely used in the Internet . [8][9]
SpyBot NetBIOS Kuang Netdevil KaZaa	There are hundreds of variants of SpyBot nowadays. Most of their C2 frameworks appear to be shared with or evolved from SDBot. But it does not provide accountability or conceal their malicious purpose in codebase. [10]
mIRC-based GT-Bots	Global Threat bot (GTbot) is a mIRC-based bot. It enables a mIRC chat-client based on a set of binaries (mainly Dynamic Link Libraries (DLLs)) and scripts. It often hides the application window in compromised hosts to make mIRC invisible to the user.[8][9]
DSNX Bots	The Data Spy Network X (DSNX) bot has a convenient plug-in interface for adding a new function. Although the default version does not meet the requirement of spreaders, plugins can help to address this problem.[8][9]
Q8 Bots	It is designed for Unix/Linux OS with the common features of a bot, such as dynamic HTTP updating, various DDoS-attacks, execution of arbitrary commands and so forth. [8]
Kaiten	It is quite similar to Q8 Bots due to the same runtime environment and lacking of spreader as well. Kaiten has an easy remote shell, thus it is convenient to check further vulnerabilities via IRC [8].
Perl-based bots	Many variants written in Perl nowadays. They are so small that only have a few hundred lines of the bots code. Thus, limited fundamental commands are available for attacks, especially for DDoS-attacks in Unix-based systems [8].

Table 2.1: Types of bots

2.2 Organization

This chapter presents some of the already existing solutions, specially the ones that are more useful for the final solution, that will be proposed in this dissertation. There are many products on the market, some specifically designed to secure only one equipment and others projected for monitoring all the network. Next sections present some of the most important (at the moment) solutions in the market.

Cisco Secure IDS is a network-based intrusion detection system, that uses:

- Signature database to trigger intrusion alarms;
- Sensor platform to provide real-time monitoring of network packets;
- Director platform, that allows the management, configuration, log and display of alarms generated by sensors.

As an essential part of the Cisco Secure Borderless Network, Cisco IPS is one of the most widely deployed intrusion prevention systems, providing protection against more than 30,000 known threats:

- Directed attacks
- Worms
- Botnets
- Malware
- Application abuse

Cisco IPS also helps your organization comply with government regulations and consumer privacy laws. It provides intrusion prevention that:

- Stops outbreaks at the network level, before they reach the desktop;
- Prevents losses from disruptions, theft, or defacement;
- Collaborates with other network components, for end-to-end, networkwide intrusion prevention;
- Supports a wide range of deployment options, with near-real-time updates for the most recent threats;
- Decreases legal liability, protects brand reputation, and safeguards intellectual property [11].

There are some products available: the network appliances 4200 Series Sensors, Intrusion Detection System Module (IDSM) for Catalyst 6000 family of switches or the IDS monitoring device - Cisco Internetwork Operating System (IOS) Firewall IDS, which turns the router into an IDS monitoring device.

2.3 GFI Network Monitoring

GFI Network Server Monitor^{TM}is a software solution that scan the network for failures or irregularities. It automatically scans the network so the network administrator can identify issues and fix unexpected problems before users even know they've happened.

GFI Network Server Monitor maximizes network availability by monitoring all aspects of Windows and Linux servers, besided workstations and other devices, such as routers. When a failure is detected, the GFI network monitor will alert by email, pager or SMS, as well as automatically taking the preconfigured corrective action by, for example, rebooting the machine, restarting the service or running a script.

GFI Network Server Monitor actually tests the status of a service, rather than deducing a service status from generated events as other products do. It is the only real way to ensure server uptime. It has built in monitoring rules that include: Exchange Server 2000/2003, Microsoft Structured Query Language (MS SQL), Oracle and Open Data Base Connectivity (ODBC) databases, CPU usage, FTP & HTTP Servers Group Membership, Active Directory & NT Directory Service (NTDS), Disk Drive health, Disk Space, Event Log (with content checking), File Existence (with content checking), TCP, ICMP/Ping, SMTP & POP3 Mail servers, Printers, Processes, Services, UNIX Shell Scripts (RSH), Simple Network Management Protocol (SNMP) & Terminal Server. Custom monitor functions can be created in Visual Basic Script (VBscript), Active Directory Service Interfaces (ADSI) and Windows Management Instrumentation (WMI). GFI Network Server Monitor can virtually monitor almost anything:

- Monitor the network and servers for software and hardware failures;
- Out-of-the-box monitoring of Exchange, Internet Security and Acceleration (ISA), Structured Query Language (SQL), Web servers and more;
- Monitor disk space, services, processes and other parameters on servers and workstations;
- It is an easy to learn, easy to use and easy to deploy solution - no client component or agent.

[12]

2.4 NESSUS

Nessus is a unified security monitoring created by Tenable Network Security, known primarily as Nessus vulnerability scanner product; it can be distributed on an entire company, inside one network or across physically separated networks. It provides hardware and virtual-appliances for easier deployment. It is managed through a Security Center that provides continuous monitoring and increases the communication effectiveness between security, audit and management teams.

Alerts can be in the form of syslog messages, notifications, emails or even launching scans. Parameters are set for number of IP addresses, open ports, critical vulnerabilities, missing patches, etc. The system launches alerts any time the parameter value thresholds are reached.

Recently Nessus included the ability to perform security and policy compliance configuration audits of Cisco IOS routers and switches. This feature provides router and switch administrators the ability to test security and policy compliance settings, requiring encrypted passwords, banning the use of common SNMP community strings, forcing the use of SSH to access the console and avoiding unauthorized users. [13]

2.5 Honeypot-based Tracking

Honeypots are a technique used to discover attacks or flaws in the network. Microsoft Windows 2000 or XP non-patched operating systems are widely used for this purpose because of their well known vulnerabilities, on Internet Explorer for instance. The main principles of a Honeypot are: it should not contain any sensitive information but pretend to have it, in order to attract the attention of attackers and all botnets, so they should first try to compromise this system. After this step, it is important to monitor these machines, trying to understand what the botnet or the attacker tried to do with them and learn how it behave. This procedure can make it possible to predict a real attack to important machines, servers and other network equipments. Nephentes is an example of a Honeypot platform.

These machines should be mounted in an “unsecured” area of the network, attracting hackers, attackers and bots. They should provide “virtual” servers for almost every service expected on the network, so the attacker doesn’t realize he is going to enter into some kind of trap.

2.5.1 Nephentes

The study described in [14] proposes to apply visualization to better understand network attacks. It doesn't apply only blacklisting of schemes and specific patterns, that could be easily subverted by more sophisticated approaches, such as P2P communications schemes. It motivates to combine computer technology, automated processes and human intelligence to get a better result, particularly comparing TreeMap and standard graph representations, thus showing the advantages of TreeMap and NFlowVis system.

Nepenthes is a honeypot platform designed for the collection and analysis of malware, such as botnet software. Together with SNORT - open source network intrusion prevention system (SNORT) and combined with NfSen or Stager web interfaces, they have been proposed as an integrated solution.

NetFlows can be also used as a fast database engine in the background, which aggregates tables with appropriate indices for fast data access.

2.6 BotHunter

Unlike Cisco, GFI or other similar systems, BotHunter is not a firewall, spam blocker or antivirus. This is an software based on a algorithm (network dialog correlation) developed under the Cyber-TA research program. It uses a heavily modified Snort version as a dialog event generation engine; when enough evidence is acquired, an infected host is declared. It produces an infection profile to summarize all evidence.

In short, BotHunter helps to rapidly identify infected machines keeping the threads and discovered infections updated in a service where every emerging exploits and malware create new dialog analysis rules. BotHunter is adaptive and share information between every machine that is using the system, keeping the "knowledge base" of BotHunter bigger and better.

BotHunter activity is not a single and simple search for a pattern, because there are many bots and they can change rapidly. So BotHunter searches, not only for incoming outside threats, but also for threats born and emerging inside the network.

False positives can also be identified and avoided: when a machine produces lots of false positives, it is "tagged", letting the system more aware the next time it has enough information to generate an alert.

Let's take a deeper look on the Bot Hunter system and some of its functioning details.

BotHunter current infection dialog set provides the detection coverage, classes and events that are illustrated in Table 2.6 and Figure 2.1

Dialog Class	**Dialog Event**	**English Description**
E1	Inbound Scan	Inbound scan
E2	Inbound Attack	Inbound infection attempt
E2[dns]	DNS Lookup to Client Exploit	A DNS query was performed to a malicious host associated with client-side exploits
E3	Egg Download	The Host downloaded a binary executable from an external source
E4	Malware C&C	Host appears to have exchanged command and control message with an external malware controller
E4	Russian Business Net Connection	Host attempted to connect to a monitored Russian Business Network site
E4[dns]	DNS Lookup to Botnet C&C	Host performed a DNS query to a known malware control site
E5	Outbound Attack Propagation	Host is conducting outbound attacks, possibly to propagate a malware infection
E5	Outbound Scan	Host may be conducting an outbound IP address sweep
E6	Attack Preparation	Host is conducting activities associated with preparing to launch an attack
E7	P2P Coordination	Host is engaged in P2P communications associated with Malware coordination
E8	Malicious Outbound Scan	Host is conducting an IP address sweep using networks ports associated with malware propagation
E8	Outbound to Malware Site	Host has connected to a known malware control site

Table 2.2: BotHunter Class definition

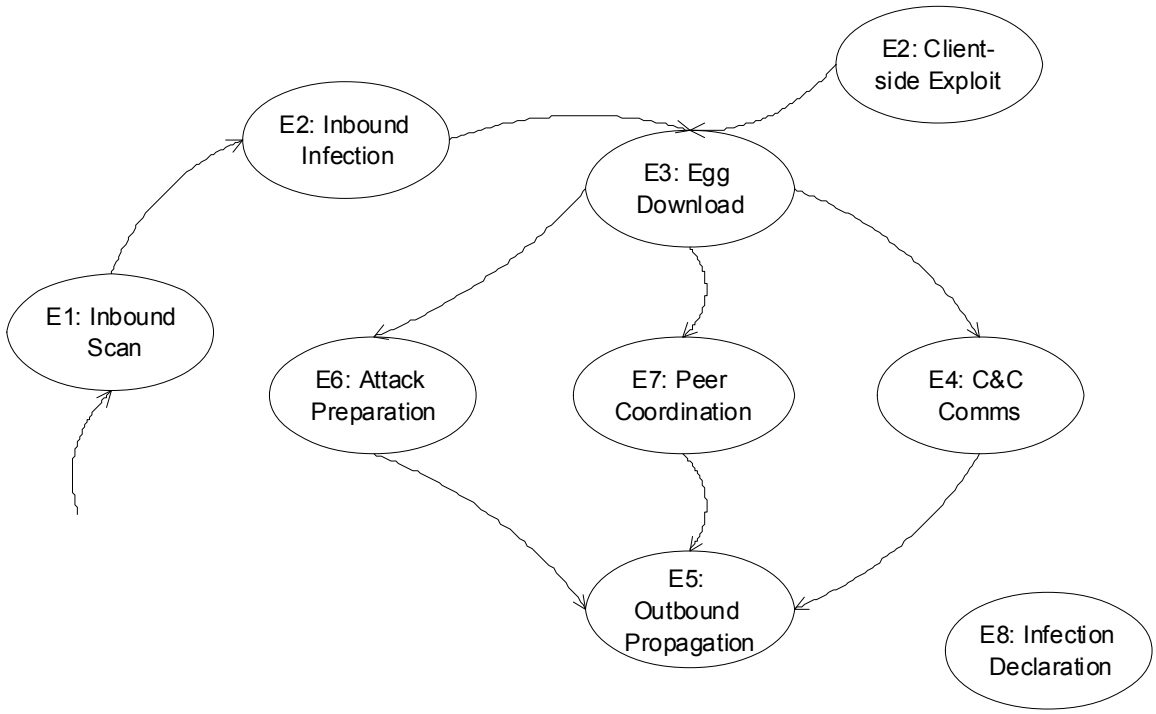


Figure 2.1: BotHunter Infection Cycle Model

The above Figure and Table do not intend to provide a strict order of events or the complete class definition used by BotHunter. This system provides and assumes that bot dialog sequence analysis is robust to the absence of some dialog events. It must comprehend multiple contributing candidates for each of the various dialog phases and could not require strict sequencing in the presented order. BotHunter captures the minimum necessary and sufficient sparse sequences of events under which an infection declaration can be triggered. Each infection declaration is ranked in a range from 0.8 to 3.8, where the greater the dialog evidence trail is, the higher is the score.[15]

BotHunter can be installed in Linux/Unix, Windows or Mac OS .

2.7 Snort

Snort is a network intrusion detection system created by Martin Roesch in 1998, being a free open source detection and prevention system that provides real time monitor to TCP/ IP networks. Snort can detect a wide variety of attacks: buffer overflows, stealth port scans, Common Gateway Interface (CGI) attacks, OS fingerprints attempts, etc. It uses a flexible

rules language to describe traffic that should pass, be blocked or generate an alert. It can be used in three scenarios: packet sniffer like tcpdump or tshark, packet logger or network intrusion prevention system.

The threat prevention components of Snort are [16]:

- Select traffic to be inspected, ensuring that only traffic that could be vulnerable is inspected. By looking only to relevant traffic, Snort performance is improved and the potential for false positives is limited.
- Normalize traffic to ensure that it is in a consistent format and proper order for inspection.
- Match traffic against one of the industry's largest rulesets using a number of different detection methods. By using multiple detection methods and a large ruleset, Snort ensures that a wide variety of threats is detected and false negatives are avoided.

Snort was downloaded 4 million times, being the most widely deployed intrusion prevention technology in the world. The fact of being open source contributed to its success because many communities and interested users can add functionalities to the software and this is how it is possible to create better software when compared to traditional proprietary methods.

How Snort works

Packet Classifier: Snort begins by processing the traffic, classifying it at high level.

- Token Ring, Ethernet ...
- Virtual Local Area Network (VLAN).
- IP.
- TCP, UDP, ICMP.

IP Defragmenter: Snort splits the IP datagrams into two or more smaller IP datagrams. Typically IP datagrams are fragmented when the Maximum Transmission Unit (MTU) is exceeded. In order to detect attacks, Snort reassemble IP datagrams in the same way the destination system of the attack would do. So, it can model the actual targets instead of merely modeling the protocols and looking for attacks within them.

TCP reassembler: because TCP transport layer could face similar issues (TCP segments may arrive out of order or be duplicated), the module stream5 on Snort reassembles TCP

sessions in the same way that the destination operating system would process them.

PortScan Processor: the primary objective in detecting portscans is to detect and track negative response. Snort's portscan detector must also identify scan attempts that never return a response, because some systems suppress it.

Detection Engine:

- Protocol analyzer ensures that correct areas are selected for inspection and information is presented consistently (Protocol Normalization, Data Normalization, Flow Analysis).
- RuleSet Selector and Multi-Rule Search Engine: snort reads and parses all the activated rules (classifying them into rulesets, prior to any packet or stream processing). Each incoming packet is matched to a corresponding ruleset based on the packet's unique parameters (it searches for protocol field, generic content and packet anomaly).
- Rule Inspector: if a Snort rule is validated, an event is generated and added to the event queue.
- Event Selector: the event queue allows Snort to track every occurrence of every rule match event within a packet. The event selector then prioritizes and selects the event information to send to the Snort output system.
- Ouput System: it process events and check if they match suppression and thresholding rules. The events are logged or passed to other systems for remediation and response purposes.

Snort can be installed in Linux/Unix, Windows(with WinPcap) or Mac OS platforms.

2.7.1 ACID

The: Analysis Console for Intrusion Databases (ACID) web interface, is a Hypertext Preprocessor (PHP) based analysis engine that will also be included in this work to build the final solution for botnet detection.

Bellow some features of ACID are described:

- Query-builder and search interface for finding alerts matching on alert meta information (e.g. signature, detection time) as well as the underlying network evidence (e.g. source/destination address, ports, payload, or flags).
- Packet viewer (decoder) will graphically display the layer-3 and layer-4 packet information of logged alerts.

- Alert management by providing constructs that logically group alerts to create incidents (alert groups), deleting the handled alerts or false positives, exporting to email for collaboration or archiving alerts to transfer them between alert databases.
- Chart and statistics generation based on time, sensor, signature, protocol, IP address, TCP/UDP ports, or classification. [17]

ACID can be installed in any system with SNORT, MySQL and PHP.

2.7.2 Websense

Websense created a system called Triton - a unified security architecture - that merges its data-loss prevention and email security products, keeping much of the security-as-a-service approach it already supports, but with a common management and reporting console.

"It's a unified platform of the systems we have for content security, with a single management and single policy to open up visibility," says Dave Meizlik, director of product marketing at Websense. "It will assess risk more clearly."

As its first product under the Triton umbrella, it is a hardware appliance which makes use of Citrix virtualisation technology to support Data Loss Prevention (DLP) and Web-filtering features. The company will add more email filtering capabilities later this year, under the Websense common management console.

The idea is to roll out content security functions as components for email, Web and DLP under Triton that can be used to centrally administer a common policy.

Websense also expects to continue marketing its stand-alone products, the Websense Web Security Gateway, the Websense Data Security Suite and Websense Email Security. But by opting for the Triton architecture, it will be possible to easily improve security on a shared hardware platform that might include additional software and security-as-a-service functions that are part of it.[18]

Chapter 3

Information Gathering and Administration

3.1 Architecture

3.1.1 Architecture Design

As already said in this thesis, it is difficult or almost impossible to detect botnets using only a stand alone solution. It is important to "divide to conquer", gathering all sensitive and important information and correlating it in order to extract the most specific and accurate conclusions that is possible to extract.

The proposed architecture and implementation have implicit software engineering, database modulation and research in order to be able to integrate PHP with shell commands, crontab, charts, Secure Shell (SSH) fetching scripts, regular expressions, statistics and MySQL simple queries. All these features will be described with more detail later in this chapter.

Diagram 3.1 represents the system and all its components:

- Fetching and Parsing scripts;
- Servers: Email, SSH, HTTP, etc;
- Snort Nodes;
- Databases;
- Anomaly detection and Alarms;
- SSH communications.

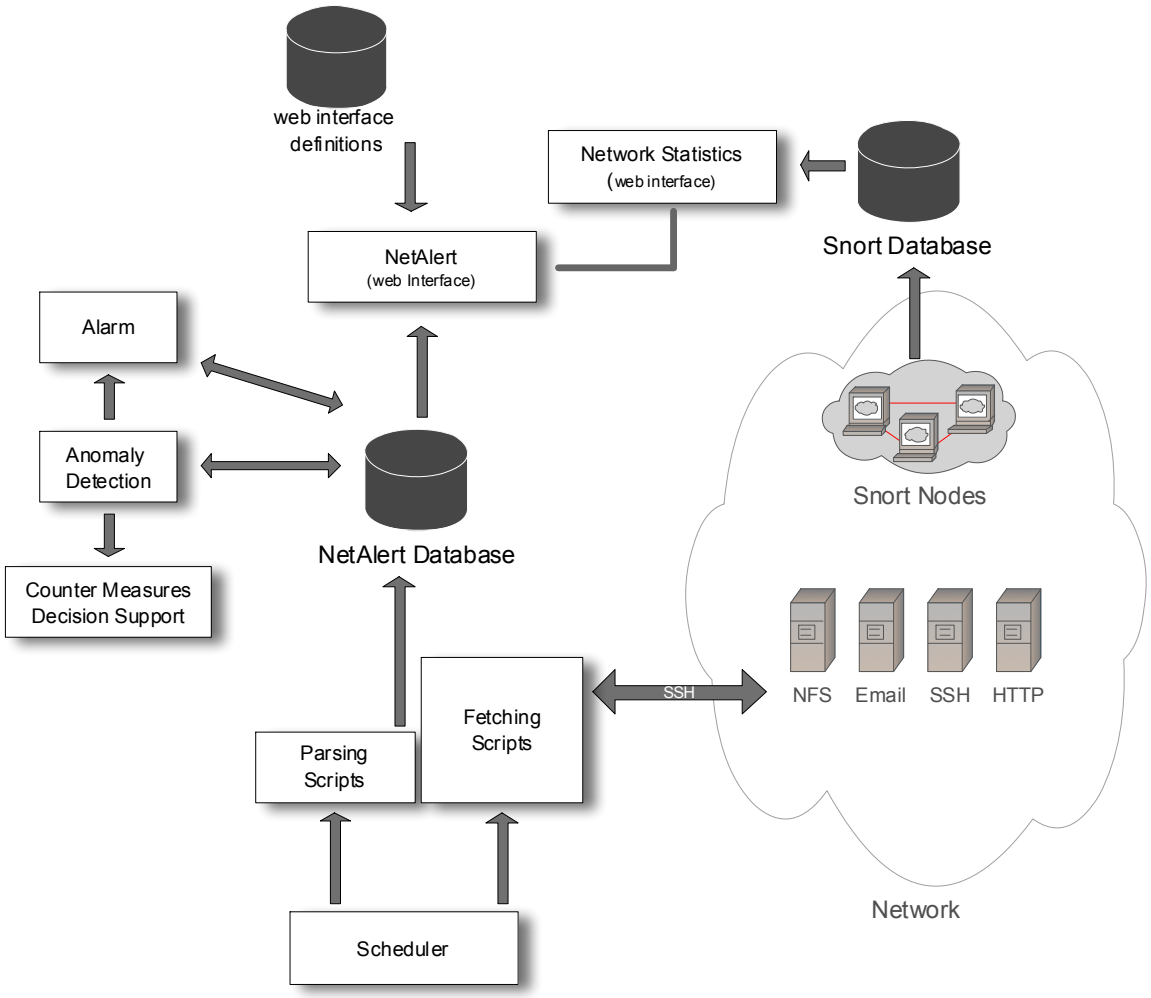


Figure 3.1: Main System Diagram

The information system is based on the Physical Diagram A.1, representing the main database, which handles most of the relevant information for this work. There are two more databases: Physical Diagram A.2, with information of Snort sensors, and Physical Diagram A.3 that gives support to the web Administration.

The Databases are separated for security and performance reasons. In the future, Snort sensors A.2 and web Administration database A.3 could need to be placed in different machines, with a dedicated database server for each one, due to the amount of information that can be generated by the processing and parsing log files. Besides, if the system has many Snort sensors, these will generate high levels of traffic and MySQL operations. Although there are many advantages on separating the machines and services needed for the system,

it is important to have in mind that the hardware architecture should be carefully chosen, monitoring a high number of services and equipments, which will generate a lot of traffic on the network. So, the servers should be "near" and should not interfere with the normal functioning of the network itself.

Why PHP?

Hypertext Preprocessor (PHP), initially named Personal Home Page Tools, was not the only possible choice: Active Server Pages (ASP), Cold Fusion, Process and Experiment Automation Realtime Language (Pearl) or even Python can also be a solution. So why PHP?

- Simple: it is a simple language, designed for creating web applications and has many built-in functions to handle common needs or most advanced ones, such as running shell scripts, handle dates, string comparison or replacing, regular expressions, etc. Anyway, programming in PHP could be quite confusing if some cautions are not taken from the beginning, organizing functions and the code itself.
- Easy to install: this is an easy to install package, in most OSs. Available for Ubuntu through "apt-get" repositories.
- Open Source: improvements are constantly made, since this is a widely used language that is tested by hundreds of developers everyday and bugs are corrected quite fast. Between the starting and ending dates of this thesis, at least 3 new versions were released and many bugs were fixed.
- Flexibility and Stability: because it runs on many platforms (AS/400, Mac OS X, Linux, Novell NetWare, RISC OS, Solaris) and exists since 1994 (actually in v5.3.2). PHP also supports many databases, like Oracle, MySQL, MSSQL, Access, Sqlite.
- Documentation: PHP should be the most widely documented web application programming language, so there are many articles, forums and code examples on the web.

Why MySQL?

MySQL is the world's most popular open source database, it is fast and consistent, it is used by individual web developers, but also by the world's largests companies that want to save time and money. Yahoo!, Google, Nokia... are some of the examples that use MySQL in some applications databases. Other important reason is flexibility, since MySQL runs on: Linux, Windows, Mac OS , Solaris, HP-UX and IBM AIX. These advantages are combined with lots of documentation available on the official website and: forums, personal pages, articles, etc. Finally, integration with PHP is very easy, being also available through the "apt-get" repositories of Ubuntu.

Why Linux?

For a project on botnet detection, Linux is obviously the best choice, since it is affected by only a small part of the immense number of threats that typically affect Windows systems.

Linux is an open source OS, very well documentated and it is easy to use, install and get the information or help that is needed for proper configuration. The specific version/distribution of Linux used in this project is Ubuntu 8.04 Long Term Support (LTS). It is free and supports all the necessary technologies: shell script, Apache, PHP, MySQL, CronJob...

Main System

The System is projected in order to be extensible and able to follow the developments that will be implemented in the future, integrating more services, equipments, new types of servers, new types of alerts, etc. The purpose of the current implementation is not to study or incorporate algorithms to detect botnets but to gather all the necessary information and make it available to a much more complex system and group of algorithms that will be developed in the future.

Although one of the objectives of this work was to evolve previous work that was already developed for the same purpose, the existing platform revealed itself very inflexible, complex to use and extense. So, the solution presented here is a complete new approach and intends to be more flexible, secure, easy to use and configure and more effective. Special attention was also given to the documentation of each file and function.

It is assumed that the machine where the System will be installed is secure and well configured referring to file permissions, users permissions and groups. So, no special care will be given to make the scheduler and fetching scripts more secure. Although the passwords used for SSH connections had to be saved on the database in cleat text, the passwords for web Administration are encrypted, so it is not possible to reverse and discover them A.1.

3.1.2 Administration

Administration is made through a web interface: the decision to use a web interface instead of a conventional software was taken because a web interface allows several advantages:

- Network administrator can access from any location with a web browser;
- Can lock addresses or address classes from accessing the interface;

- It is easy to extend the system capabilities, quality, security, etc, without the need of special configurations, updates or interventions on the client side or even on the monitored servers;

- Network administrator can enclose the server in a secure location and only the sensors will be reporting from other locations on the network.

The first window/page of the system is always the login. The next figure illustrates the login page:

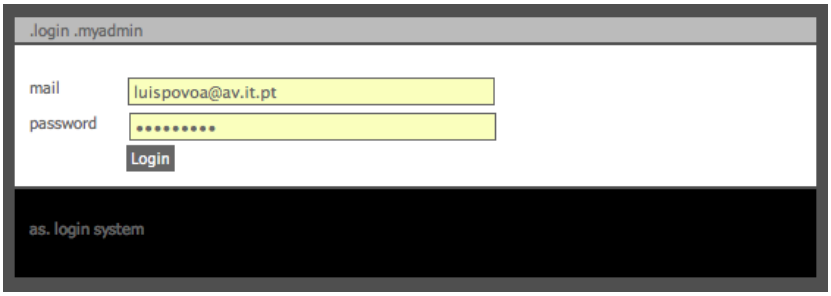


Figure 3.2: Login Window

After the login, the main page of the system will appear, showing the most used options, the update dates, some information about the user and the menu of the whole system.

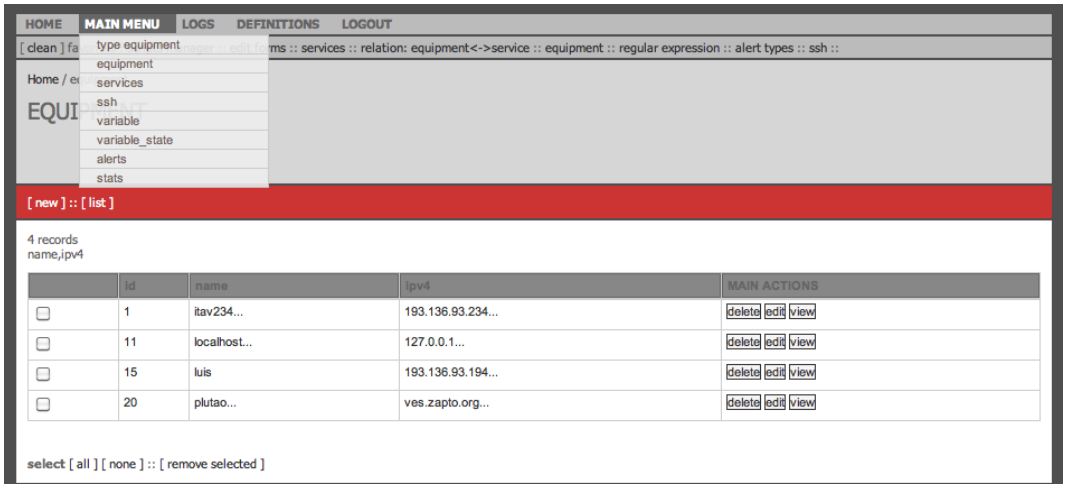


Figure 3.3: Main Page - Home

A more detailed explanation of the system tables created on the database is presented below, where some of them include images that illustrate the menu or correspondent form:

- Equipment: it makes it possible to insert, edit or delete equipments. Each entry on this table holds the IP information, hostname, OS type, Mac Address, the last time the

equipment has been updated, etc. The *rel_equi_service* table will relate each equipment to the services it provides.

- Type Equipment: it stores the SNMP version and/or OS general information. So, it can then be associated to the Equipment entries.

- SSH: in this table it is possible to insert a login (username, password and identity keys), to access equipments via SSH and the date of the last access. An *equipment_id* is associated to each entry, identifying the equipment ssh access (login/password or identity key).

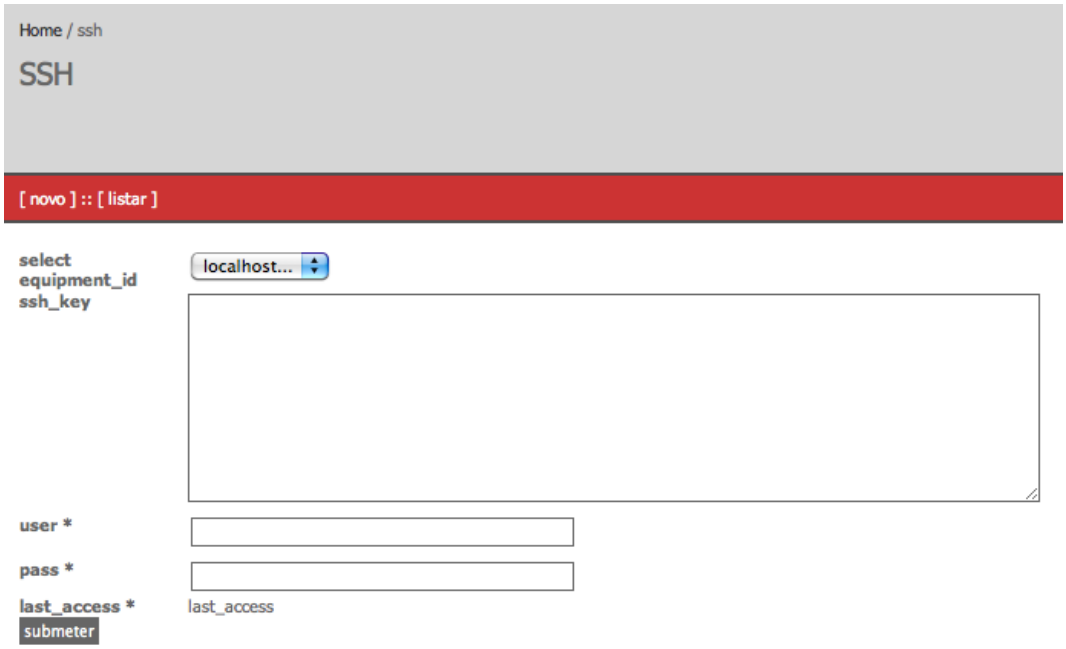


Figure 3.4: SSH new entry form

- Services: this table holds the services available on the system, being possible to add, delete or edit existing services. As referred above in the Equipment case, the *rel_equi_service* table will relate the services with the equipments, where they are available.

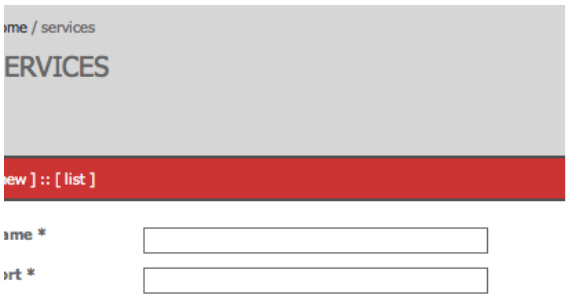


Figure 3.5: Services new entry form

- Relation Equipment Service: this is a relation table that stores the Services for each equipment, being possible to add, edit and delete entries. It is a very important table, because each entry will have an interval time defined and will be used to create a cronjob. It also holds information about the last updates for the *equipment_id* - *service_id* pair.

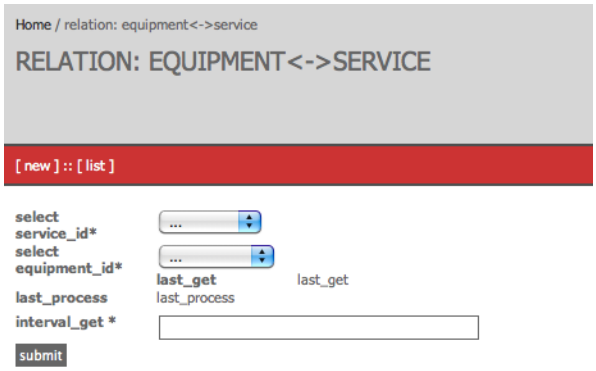


Figure 3.6: Relation between equipment and service - edit

- Variables: this is a fundamental table of the system, containing the system variables, that is, the events. On the Web Administration, it is possible to add, edit, or remove entries.

- Variables State: this table is obviously related to the previous ones, storing the state of the variables. When filling this table, it is very important to add as much information as possible, like *equipment_id* or *service_id*, so it will become easier to identify problems and their origin.

- Alert: this table is not for management on the web interface, but only to view informa-

tion and check alerts on the system.

- Alert Types: it is possible to add, edit or remove alert types, being the responsibility of the user administrator to check if there are some Alerts already using each type. This table gives information on the Alert, describes the possible problem and suggests a solution; it can also have command or a service associated to clearly identify the problem.

- Scripts: executes, inserts, edits and deletes scripts, being possible to choose (if needed) the equipment where the script will be running on. Each time a script is edited, a file with its *script_id* is created or updated. It's also possible to update all scripts at once.

Figure 3.7: Script - list

- Logs: it is possible to view all logs fetched from each server and added to the database; it is not possible to edit these logs registries, as they are fetched by the above explained scripts.

- Regular Expressions: this feature allows to add and edit regular expressions, including also a field to add comments. The image below shows the "edit" example.

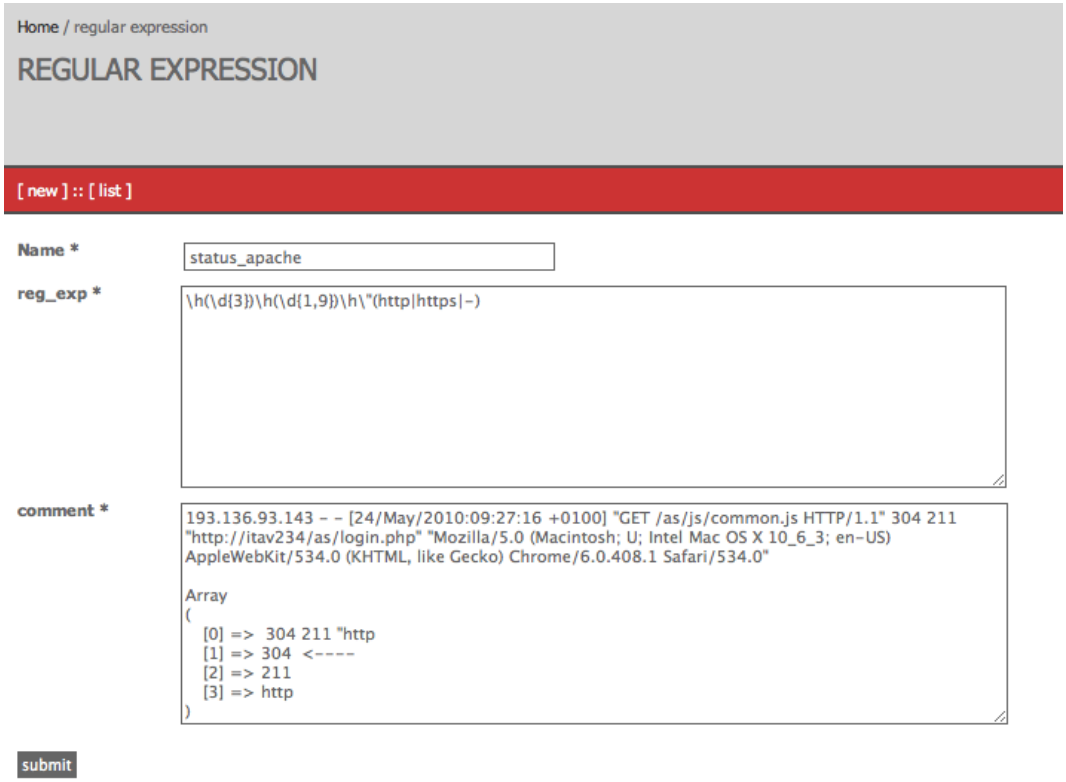
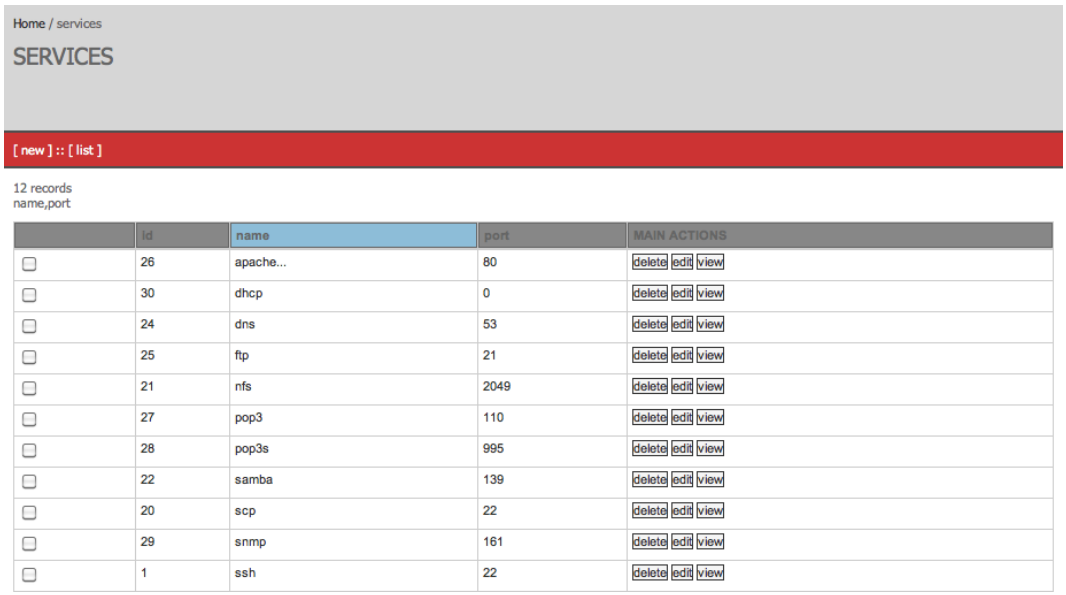


Figure 3.8: Regular Expression edit form

It is possible to show different menus and give different database access for each registered domain in the administrators area. A *domain.mysql.php* file defines the user that accesses the database, so it is possible for example to have a domain with only *viewing* permissions and others that can change information or insert new information.

Another important feature is that, at all listing tables, it is possible to order by each header field, simply clicking on them. The ordering algorithm identifies IP addresses, strings and numbers (this last option can present some problems because sometimes a simple added space can "turn" a number into a string).



	id	name	port	MAIN ACTIONS
☐	26	apache...	80	delete edit view
☐	30	dhcp	0	delete edit view
☐	24	dns	53	delete edit view
☐	25	ftp	21	delete edit view
☐	21	nfs	2049	delete edit view
☐	27	pop3	110	delete edit view
☐	28	pop3s	995	delete edit view
☐	22	samba	139	delete edit view
☐	20	scp	22	delete edit view
☐	29	snmp	161	delete edit view
☐	1	ssh	22	delete edit view

Figure 3.9: List order example

3.1.3 Crontab, Fetching and Parsing Scripts

This section will present the details of the crontab management, fetching and parsing scripts.

Crontab

The crontab is generated automatically on the web interface (Definitions->Crontab Update) creating, based on the entries of the relation table "*rel_equi__service*", all the cronjobs. Each entry on that table will create a different cronjob, with the defined update period. If there is no *interval_get* defined, no cronjob will be generated. It is also important to refer that *interval_get* should be already on the cronjob format (Appendix B).

For each entry, a different PHP file is created and the command that is used by the cronjob is "php file_name.php" (the name will be cron_get_proc_xx.php, where *xx* is the number of the *rel_equi__service_id* correspondent). The only information stored on the file is the *id* of the relation (equipment - service), for security reasons, so there is no need to have the login information (user and password) on the php file.

A script is already written in PHP and will be the basis to generate the individual script files for each service - equipment pair. The script just calls the functions explained on the next sections.

Fetching Scripts

The Fetching Scripts are executed in PHP, using the SSH protocol as the transport layer for the information log files.

Figure 3.10 represents all the process sequence for fetching and processing log files.

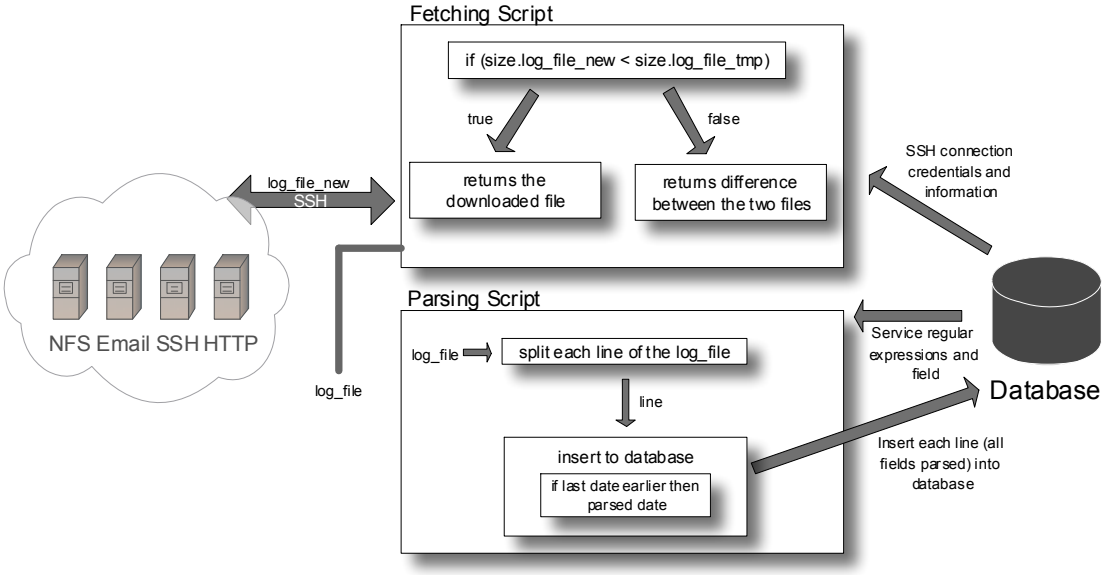


Figure 3.10: Process Script Diagram

Here are the steps of the script described with more detail:

- Gather all the needed information: user, password, IP address, remote path (full path on server), local path (relative path inside the system defined log path: also editable on the Definitions->General Definitions menu of the Administration web interface) and hostname of the equipment.
- Download, via SSH, the log file to a specified folder.
- A copy of the downloaded file is made at the end of the fetching script (named "hostname"_tmp.log) in order to compare the next "new file" with the previous downloaded file. Next, it will create a file that will be passed to parsing scripts, only with the differences between the downloaded file and the previous one.
- Special attention is given to downloaded files with a size smaller than the last one (the "tmp" file), which means that the server log system has changed the last log file (compressing it to a .tgz file for example) and created a new one, so the fetching script will not check for differences, creates the file as it is and passes it to parsing scripts.

Parsing Scripts

The parsing scripts are organized between regular expressions (stored at the database), specific functions to insert and parse each line, a parsing function and some auxiliary functions, such as date comparison, get dates, update dates, etc.

If some new service has to be added to the system, a new function should also be created with the regular expressions already existent on the database or in addition, if needed, creating new regular expressions. The function should be created in the *process.functions.php* file and the new service has to be declared on the process function main switch.

The function created to insert on each database (let us give the apache service example) is organized in this way:

- Name: *insert_to_db_apache*
- Arguments:
  - $query_base: the object used for the connection to database.
  - $list: an array of arguments
  - $dbname: database name
  - $rel_equi__service_id: the id number of the relation table between equipment and service
  - $s: the string to be parsed
  - $log_file: the log file, usefull in some cases where the line to parse doesn't have a complete Date (without the year)
- Return: the return is widely used for text, errors, debug messages, etc.

For each element of the list, this function will make a comparison with the switch cases, creating a variable with the same name of the element on the list. For each element, a function called process_pattern will be called with the following arguments:

- $query_base: the object used for the connection to the database
- $name: the name of the regular expression to use (stored on the database)
- $s: the string to parse
- $index: each parsed result (the return) is an array of values and with the index it is possible to define each value that will be acquired.

And, finally, the process function, that will call and activate all the others.

Regular expressions

Regular expressions are an important part of the processing module, can be managed on the web interface and have always an example and a description of the result. The regular expression will be used in the parsing function that uses the existent preg_match function of PHP. If the user or the network administrator wants to add new regular expressions, he should also specify all these fields.

Preg Match uses the Pearl Compatible Regular Expressions (PCRE) engine, uses the same syntax and semantics as Perl 5 PCRE, has its own native Application Programming Interface (API), as well as a set of wrapper functions that correspond to the Portable Operating System Interface (POSIX) regular expression API. The PCRE library is free, even for building proprietary software.

Custom query

With this feature, it is possible to create a custom query to the database, searching for a specific content, correlating tables on the database. If the query is often used, is possible to save it and use it a next time (Figure 3.11).

This feature uses jquery (1.3.2) and ajax functions. Due to some issues in the integration of both versions, 1.3.2 and 1.2.6pack, an iframe was used for this specific feature, so it is possible to include 1.2.6pack for the rest of the web administration and 1.3.2 just for this page.

The *sqlbuilder.htm* file creates the interface where the user can generate the query he wants to make. The system can be expanded with more tables and more processing, as shown in Figure 3.11, being possible to define:

- Result columns: the result columns can be a column of the database or a special operation made directly on the query (COUNT, AVG, STD).
- Where clauses: the where clauses can be defined as contains, equal, not contains, etc.
- Group columns method.
- Sort method: as Ascending or Descending.

It automatically updates the query string, so it is possible to see the construction of the final string, checking if there are any predictable errors.

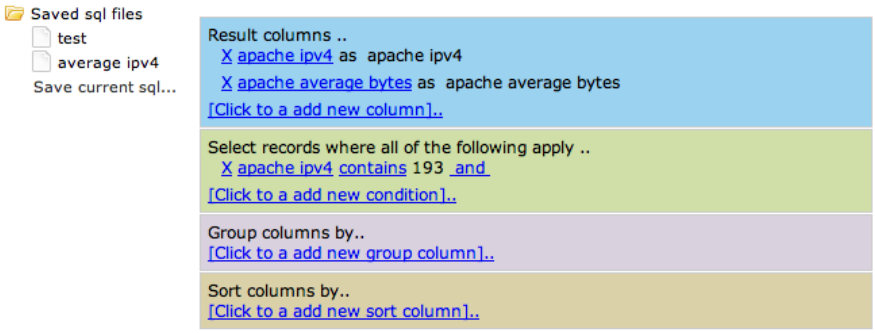


Figure 3.11: Specific Query Interface

When the complete query is defined, the user just has to click *Run SQL* and the results will appear.

3.1.4 Statistics and Chart

The system also includes a chart and statistics section. All data is available in the logs section, where it is possible to view statistics about the gathered information, so it will be easier to identify possible problems or threats. Graphics are shown in the initial page of the Statistics section, but they can be easily changed (on the *stats.php* file).

Since the acquired data could not be related, the system was tested with different log origins: some of them from real servers and others from simulated servers, resulting from a group of experiments that were run at the Institute of Telecommunications laboratory [19].

For each service, some statistical values were considered:

- Top IP addresses clients;
- Top ports used;
- Top contents requested;

- Average size of the contents or requests;
- Variance.

Each service, has a chart representing the most important data values, some representing the last entries on the database and others, properly identified, representing all the data in the log database, making the comparison between the normal behavior and the deviation of the latest result easier. The chart parameters (color, type, data, etc) can be easily changed in the *graph.functions.php* file; the charts use the mc-goog-visualization class, which provides simple support for integrating Google Visualization charts with our own internal database. It includes a complete parser for the Google Visualization Query Language and offers the same ease of pivoting and formatting data from the database as is currently possible with Google Spreadsheets, for example.[20]

The next figure is an example of the graphics that can be found in the administration system Figure 3.12:

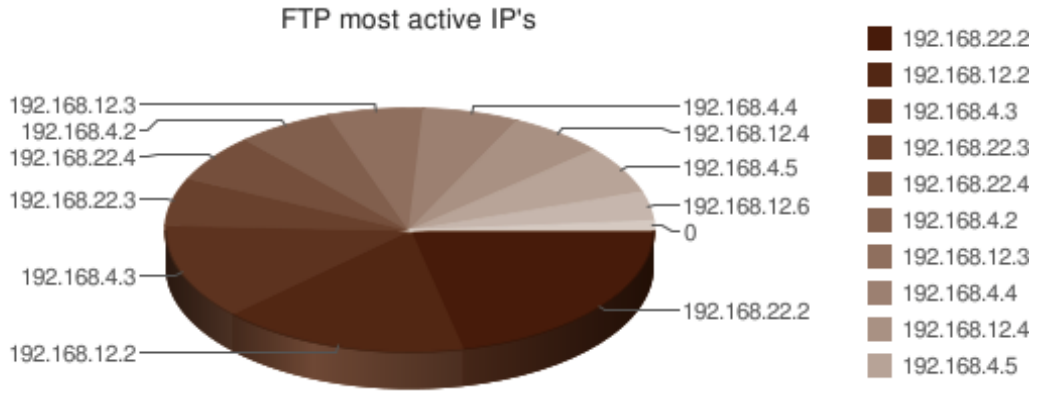


Figure 3.12: FTP chart example

This library provides an easy way to build graphs, using data retrieved from the system databases. For each graph, a function is built in order to enable changing the query arguments, the chart type or other parameters, so the graph really adapts and answers the user/developer needs. It is also important to say that the created charts are png images, so it is not possible to interact. Although Google Visualization allows the creation of this kind of charts, the specific library that is used doesn't currently have these features.

Export file

Each query made on the whole administration system, and related with logs, can be exported to a file (csv file), so it can be processed later on MATLAB or other program more

specific and oriented for statistics calculation. The export is made with Session variables that hold the last query parameters and use them to export the results.

3.1.5 Snort

Snort is part of the proposed solution, that uses snort rules managed through the web interface. Snort is running as a Daemon, with just an extra option of Basic Analysis and Security Engine (BASE) interface and a MySQL plugin.

Snort-Rules

Snort uses a powerful and simple lightweight rules description, where each rule is only composed by one line. It is not the scope of this thesis to explain *Snort-Rules*, but let us just briefly explain how Snort rules are constructed:

- Rule Actions: define: who, where and what. It's the first item and it can have the following actions: alert, log or pass;
- Protocols: TCP, UDP and ICMP. It is possible that more protocols will be added in the future;
- IP Addresses: "any" keyword or the numeric IP address can be used;
- Port Number: "any" keyword or the numeric port number can be used. The "!" negation operator can also be used;
- Direction Operator: <-, -> or <> can be used to indicate the direction of the traffic that the rule applies to;
- Rule Options: there are fifteen rule options, such as: msg, logto, minfrag, Time to Live (TTL) ... They can be used as in the following example: rule_option1:rule_arguments 1.

The Web Interface allows the administrators to edit the files, being immediately used by snort. It is also possible to add or remove rule files. Anyway, the snort daemon has to be configured to use the files, otherwise they will not be used by snort. It is hard to administrate all this environment, requiring root access to the terminal.

ACID

Analysis Console for Intrusion Databases is used on this solution as a second way to view and analyze packets statistics. There is just a link to the Original ACID system, on the Web Administration. As explained above, Snort collects the data and stores it on the MySQL database, while BASE - ACID just reads the data and shows it to the user.

3.1.6 HoneyPot

An HoneyPot, as already explained earlier in this thesis, is a system with a non patched version of windows, that is commonly attacked and affected by botnets. So, this machine should give the right feeling that something wrong is happening on the network much sooner than the botnet compromises a really important machine with real information or a server.

This machine should have some services active, in order to be attractive, easy to get in and trigger some interests on the bot commander, like controling the service that the machine is providing or simply putting it down.

It is also required that it has a SSH server, so it should be possible to connect and gather information provided in its logs. Otherwise, the information gathering system would not provide any other way to get the information and logs.

3.1.7 Installation and Use

Installing this system should be easy on a Ubuntu 8.04+. A compressed file is provided, containing:

- PHP source code and all needed folders.
- Snort Rules and Snort Configuration (snort.conf) file: the user has to compile and install snort (code included on the tar file).
- Install Script:
  - Creates SQL databases (for administration web interface A.3 and Implementation Diagram A.1)
  - Installs via apt-get, MySQL, PHP, Apache, plugins and libraries.
  - Forces right permissions to files and folders.
  - Note that it has to be executed under administrator/root account (sudo).

When installing the web administration and the associated software, it is fundamental to take some special attention to these aspects:

- Folder configurations and permissions;
- Services and Equipment configuration;
- Compile Snort and configure it to work with snort rules (on the system folder) and the mysql table; [21];
- Configure ACID BASE system (some information can be found on the web [17]).

After the system was installed, it has to be configured (already on the web interface):

- Folder for log files;
- Folder for scripts, used for the dynamic scripts created on the web interface;
- Types of equipments, equipments and SSH credentials for login;
- Create new administrators.

3.2 Discussion

3.2.1 Limitations and Potential Solutions

There are some known limitations of the system, like for example the fact that the solution based on signatures is put apart and relies on another system. Maybe a solution for this is try to gather information from most of the available anti-virus, IDS and IPS systems, use and correlate all information in one single system in order to avoid false positives. This way, it would be more difficult to let a virus, worm or botnet pass without detection.

It is important to study each signature and the behavior of its associated virus, botnet..., and then report it to a single system (for example the one created in this work). By knowing the exact behavior of each threat, the system can be very effective in detecting and avoiding the different types of threats. This should work mostly at the Network level, letting the least possible tasks to be solved at the final machine/equipment (personal computer, server, Personal Digital Assistant (PDA), mobile phone).

For now, the system does not store in the database information of each specific bot, it just stores alerts related with some of them. So, in the future it is important to add new functionalities and store a table for each detected and not detected botnets, including the behavior description of each one: the origin, the botnet class, how it works, kinds of threats, signature

(if already exists in some of the shared systems), data sources, detection method(active variables), solutions and its results (with some quantitative evaluation of its effectiveness).

Chapter 4

Conclusions and Future work

4.1 Combining Multiple Techniques towards a future botnet detection system

As already seen on this dissertation the best solution to fight an army is to create another one.

Most of the known network security solutions are based on a signature approach. This is a very good approach for a "virus-enviroment", which has a static component, some history, different versions and always the same signature. The botnet world is quite different, because it is constantly changing, forcing systems to:

- Update (rules, countermeasures, etc);
- Distribute all information;
- Install/update at each node of the network.

These actions should happen every time the "world" changes a little bit or new threats emerge. An isolated solution is not enough to detect these new threats, the main objective should be correlating available information and using as many systems as possible to detect botnet activities.

The solution that was presented in this dissertation combines an already existing system (Snort) with new functionalities that are added to collect information of existing servers and equipments. This allows to create a good information environment, with lots of information sources, which means that it would be easier to identify the source of security attacks or threats. The solution will not be only based on signatures to determine if there is a threat on the network, but also on a vision of the different network equipments and servers. This is a more flexible solution, because it doesn't need to be updated every time a signature

changes and doesn't have duplicated information. In this system, botnets are identified based on network analysis, by correlating server and equipments logs (syslog, services logs, etc).

The developed system is relatively easy to install and use, at least for the majority of its features (for example, it is not so intuitive to create new services). It should be improved, as every system, but it presents a high flexibility degree, it is fast and stable.

4.2 Future work

The purpose of this work was not to create a final system for detecting botnets, but to gather all the necessary information, make some statistical calculations with it and create an easy interface to configure and use the system. The reason for this choice lies in the fact that previous work that was already done with this purpose is very inflexible and hard to configure, so the decision was to start from the beginning and try a different approach to build the platform.

Later on the project it will be possible to correlate all the information, activate variables, launch alerts and trigger countermeasures to mitigate or defeat botnets. Algorithms should be created to correlate all the information provided by the gathering system and Snort, compare them and reflect the conclusions on the variables and alerts values. In the end, the system should be capable to launch alerts any time a group of variables is activated and answer to these alerts with the appropriate countermeasures, that are also stored in the database.

The first thing that should be done with the developed system is to run it in a real environment and test it with servers in the same network. The system has been tested with virtual servers and log files from different networks, so the conclusions and statistics do not give us a credible result. The tests that were performed were not based on real values, because it would take too much time to install, configure and create all the servers. Besides, several tests were already made in the Institute of Telecommunications laboratory, including almost every needed kind of log files. Those that were not possible to acquire from the conducted simulations, were created, installed and configured with the administration system.

Although the developed database A.1 seems to be the most appropriate for the system, later on the project it became clear that some improvements could be made in order to make the system even more flexible:

- The log tables could have been designed to be created dynamically, so it should not be necessary to add a new table each time a different service is added to the system;

- A table with all the possible fields in all services should be added;
- A table relating the fields with the specific service log should be added;
- The function for parsing could be generated in run-time.

It would also be interesting to create some sort of report, to share information and put the system running in more than one network. Then, analyze why and where are the systems failing or not identifying some botnets. Sharing information is the best way to identify threats and current attacks. Like bothunter, that reports all identified botnets and all running systems, the proposed system will also have that integrated information.

In the future, this system should have something like experimental medicine, sharing information between many networks, trying countermeasures and recording his results and effects, so that latter on, in the next known affected network, the best solution can be applied much faster, improving the efficiency of the sequence of commands and countermeasures.

Having a database of DNS monitoring and comparing the DNS logs with the legitimate DNS resolved by the system could also improve the system effectiveness. As already seen, this is the main cause of phishing and similar attack techniques, as stated in a study published on the International Journal of Computer Science and Information Security: 'Detecting Botnet Activities Based on Abnormal DNS traffic". [22]. This means that testing in real time the answers of the DNS servers could be a good solution to avoid the illicit "redirections" for phishing websites.

As already seen, the best way to avoid and combat threats is using different kinds and layers of security, so another suggestion for future work is to search information about viruses on the specialized web-site companies (anti-virus, security companies, firewalls or security pages). Most of the anti-virus systems and companies have a database information constantly updated and accessible to everyone. These websites also have information about, how to defeat the virus and threats, so it will be important to gather that information and make it available in the database, for the users of the developed system.

Bibliography

[1] R. Anderson, *Network Attack and Defense*, 2nd ed. Wiley, 2001, ch. 18, pp. 367–390.

[2] S. Pinzon and C. Nachreiner. (2008, Apr.) Understanding and blocking the new botnets.

[3] "Botnets: The new threat landscape," Cisco, 2007.

[4] J. R. Will Dormann, "Securing your web browser."

[5] M. Cremonini and M. Riccardi, "The dorothy project: An open botnet analysis framework for automatic tracking and activity visualization," 2010.

[6] I. A. r. r. Copyright 2009 Websense. Next year in the threat webscape - websense security labs predictions for 2010.

[7] C. Monteiro, "Sistema integrado para recolha de informação de rede," Master's thesis, University of Aveiro, 2008-2009.

[8] M. K. P. Bacher, T. Holz and G. Wichersky. Know your enemy: Tracking botnets.

[9] *Examining the criminology of bot zoo.* Singapore: Pro- ceedings of the 6th International Conference on Information, Communications and Signal Processing (ICICS '07), December 2007.

[10] *An inside look at botnets*, vol. Advances in Information Security. Springer: Proceedings of the ARO-DHS Special Workshop on Malware Detection, 2006.

[11] M. Janson, "Cisco intrusion prevention system."

[12] Networking and security - gfi network server monitor.

[13] J. Huffard. Tenable network security releases security and policy compliance configuration audits for cisco ios.

[14] F. Mansmann, F. Fischer, D. A. Keim, and S. C. North, "Visual support for analyzing network traffic and intrusion detection events using treemap and graph representations." 2009. [Online]. Available: http://dblp.uni-trier.de/db/conf/chimit/chimit2009.html#MansmannFKN09

[15] (2009, jun) How bothunter works. [Online]. Available: www.bothunter.net

[16] SourceFire, "Discover. determine. defend - snort threat prevention components," 2007.

[17] R. Danyliw. Analysis console for intrusion databases.

[18] E. Messmer, "Websense launches triton security platform," Feb 2010.

[19] C. M. D. S. Miranda, "Implementation of realistic scenarios for ground truth purposes," Master's thesis, University of Aveiro, 2009.

[20] Chad, *PHP Google Visualization Library*, May 2009.

[21] D. Gullett, *Snort 2.8.5 and Snort Report 1.3.1 on Ubuntu 8.04 LTS Installation Guide*, 1st ed., Symmetrix Technologies, February 2010.

[22] A. M. Manasrah, A. Hasan, O. A. Abouabdalla, and S. Ramadass. (2009, Nov.) Detecting botnet activities based on abnormal dns traffic. [Online]. Available: http://arxiv.org/abs/0911.0487

[23] P. V. ¡paul@vix.com¿, *Manpages Crontab*, December 1993.

Appendices

Appendix A

Appendix A

A.1 Database and Architecture Diagrams

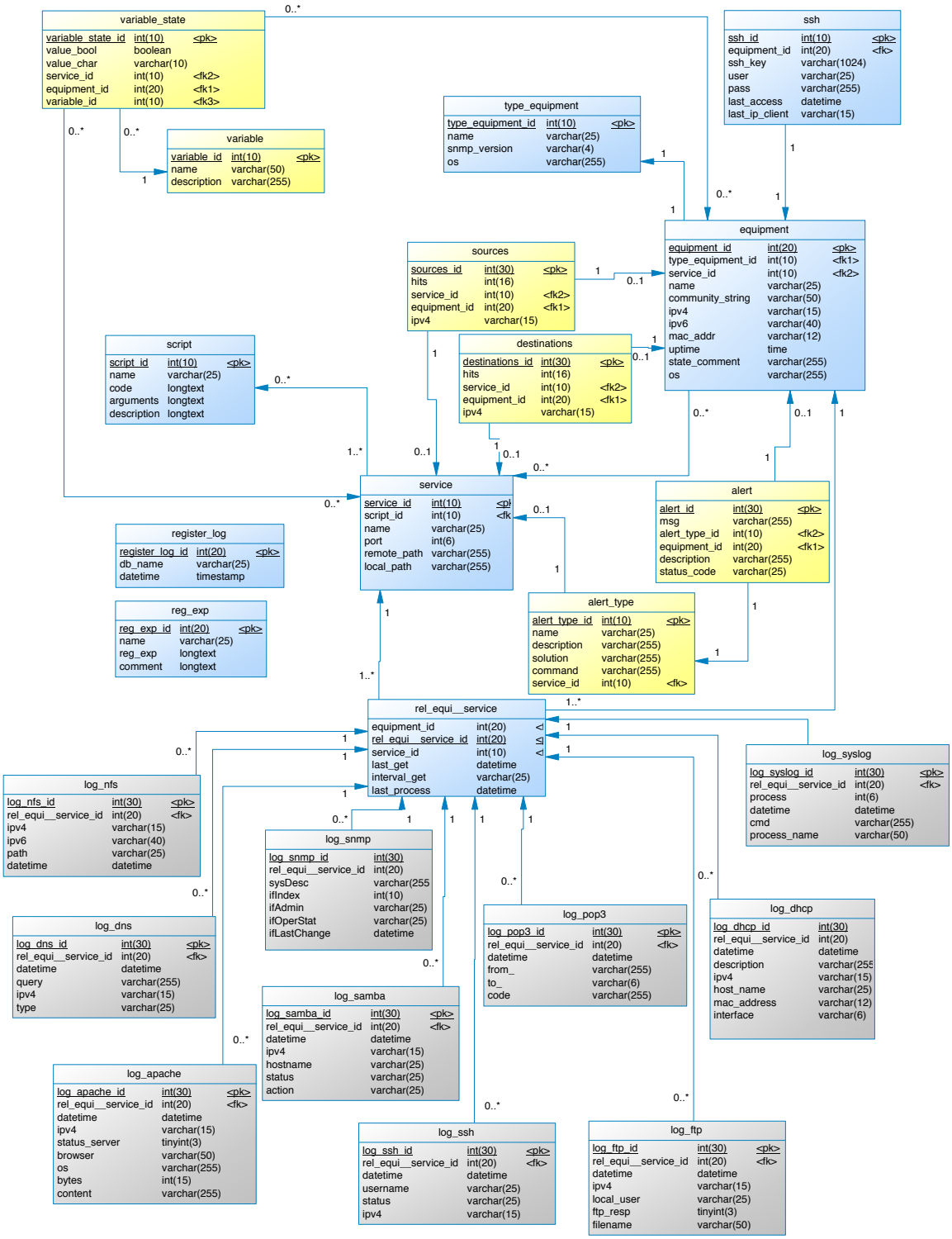


Figure A.1: NetAlert Database Physical Diagram

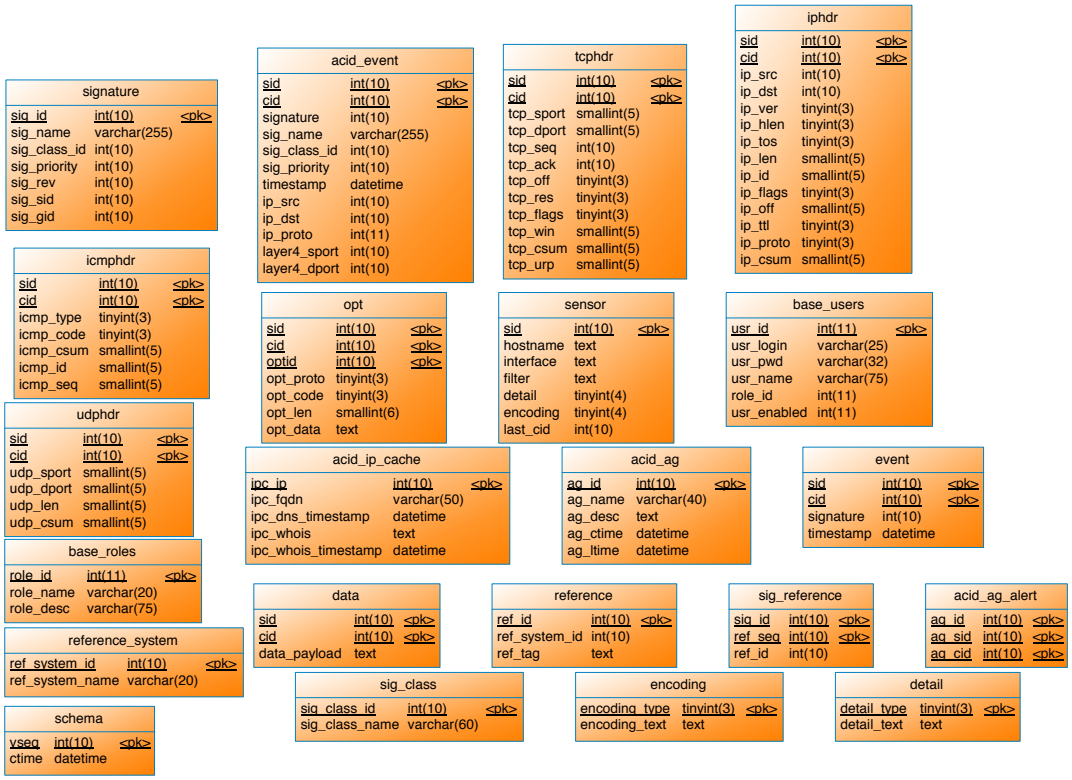


Figure A.2: Snort Base Database Physical Diagram

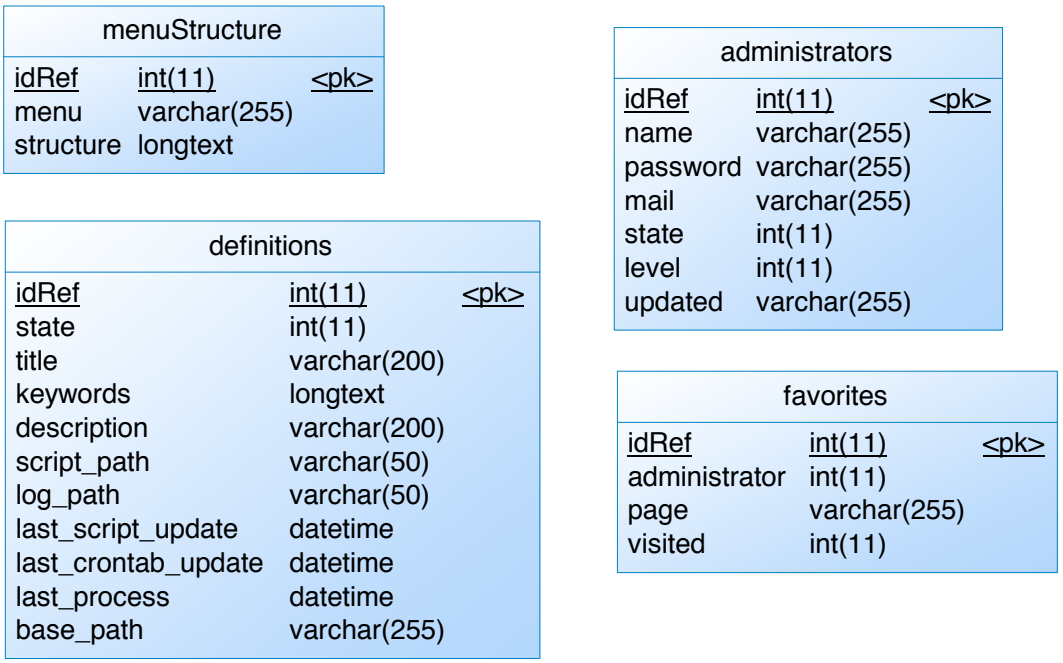


Figure A.3: NetAlert Interface Database Physical Diagram

Appendix B

Appendix B

B.1 Crontab

A crontab contains instructions to the cron daemon. Each user has a different crontab; commands in any of them will be executed as the user who owns it.

Commands are executed by cron when the minute, hour and month of year fields match the current time, and when at least one of the two day fields (day of month, or day of week) matches the current time. Cron examines cron entries once every minute. The time and date fields are:

- minute: 0-59
- hour: 0-23
- day of month: 1-31
- month: 1-12 (or names, see below)
- day of week: 0-7 (0 or 7 is Sun, or use names)

A field may be an asterisk (*), which always stands for "first-last".
Ranges of numbers are allowed. Ranges are two numbers separated with a hyphen. The specified range is inclusive. For example, 8-11 for an "hours" entry specifies execution at hours 8, 9, 10 and 11.

Lists are allowed. A list is a set of numbers (or ranges) separated by commas. Examples: "1,2,5,9", "0-4,8-12".

Step values can be used in conjunction with ranges. Following a range with "/<number>" specifies the step that will be used through the range. For example, "0-23/2" can be used

in the hours field to specify command execution every other hour (the alternative in the V7 standard is "0,2,4,6,8,10,12,14,16,18,20,22"). Steps are also permitted after an asterisk, so if you want to say "every two hours", just use "*/2".

Names can also be used for the "month" and "day of week" fields. Use the first three letters of the particular day or month (case does not matter). Ranges or lists of names are not allowed.

Example: 5 0 * * * $HOME/bin/daily.job >> $HOME/tmp/out 2>&1 It runs every day, five minutes after midnight. [23]

B.2 Diff

This section shows some options of the *diff* unix command, it can be important to change the command in order to get differences between files, last downloaded and the previous file. More information can be found at *info diff* or *man diff*, on UNIX.

The command used is: diff –new-line-format="%L" –unchanged-line-format= *file_temp file_to_compare* > *file_for_output*

The command just creates the new file with the different lines, ignoring the lines that exists in both files. If the new file has some new lines at the top, diff will give a "horizon" margin for search (the default number is enough for this particular case, but it could be changed if needed). It tries to find an equal line, even if these are in different lines, the diff command finds it and compare the next lines with the same "interval".