Operational Programme
"*Competitiveness*"

R&D Cooperations with Organizations of non-European Countries

EAR: Early warning system for automatic detection of Internet-based Cyberattacks
(G.S.R.T. code: ΗΠΑ-022)

D1: "Requirements Analysis"

Abstract: This document analyzes the current status of cyberattacks on the Internet and their consequenses. A collection of reviews of state-of-the-art solutions relevant to EAR project and their shortcomings are described. The requirements of the EAR system are finally defined which include functionality requirements as well as performance and deployment constrains.

Contractual Date of Delivery	12/07/2004
Actual Date of Delivery	28/07/2004
Delivery Security Class	Public
Editors	Manolis Petsagourakis Mixalis Stivaktakis Periklis Akritidis
Contributors	FORTH, GA Tech, FORTHnet

The EAR consortium consists of:

FORTH	Coordinator	Greece

GA Tech	Principal Contractor	USA
FORTHnet	Principal Contractor	Greece

Contents

1. Introduction

As Internet threats appear with increased speed and in greater numbers each day, it is more important than ever for enterprises or organizations to use tools for monitoring the traffic and detect threats. The objectives of the early warning system proposed is to provide a comprehensive view of the internet security landscape. It should alert as soon as possible network or system administrators of attacks detected with target their network elements or systems and offer additional analysis about the behavior of this attack.

The rest of the document is organized as follows. In the first section the reasons for the development of the system are analyzed, where by real examples the magnitude of cyber attacks and their consequences are presented.

The second section analyzes the current state of the art tools and detection mechanisms like active monitoring systems, passive monitoring and honeypots. The weaknesses of existing defense mechanisms like intrusion detection system and firewalls are discussed.

Then, the typical worm and future worm characteristics are outlined. The functional requirements are discussed with an emphasis on the types of worms that are detectable, the detection delay, the false positives, the configuration of the system security constrains, privacy issues and how it can be integrated with other systems. The document continues with a short discussion on the performance constrains and the system deployment.

The next section addresses a similar with EAR system commercial product. The last section concludes the document.

1.1 The need for early warning systems

As networks get faster and as network-centric applications get more complex, our understanding of the Internet continues to diminish. Nowadays, we frequently discover, to our surprise, that there exist new aspects of Internet behavior that are either unknown or poorly understood. A couple of years ago, for example, the world was surprised to learn that more than 4,000 Denial-of-Service (DoS) attacks are being launched on the Internet every week. This surprising result attracted the interest of public and of media together.

Although at that time lots of people had heard about Denial-of-service attacks, most of them did not really know that their magnitude was so high. Most of them they were not simply aware of the wars raging on the Internet. Furthermore, organizations are increasingly under attack from viruses, hackers and blended threats, with more than 70 new vulnerabilities and 100 new viruses identified each week.

Besides DoS attacks, malicious self-replicating programs called worms continue to plague our networks, often causing service disruption and unprecedented damage. For example, on the January 15th of 2003 at 05:29, the Sapphire (or Slammer) Worm was launched, exploiting a vulnerability in the software of SQL database servers. Sapphire infected more than 70,000 computers in less than 30 minutes. Figure 1 shows the geographic coverage of the Sapphire worm 30 minutes after the worm was released. We see that the worm infected close to 75,000 computers worldwide including areas as remote as the Fiji islands and Greenland. Indeed, Sapphire was the first worm to demonstrate that worms can spread globally at time scales in which human intervention and response is either limited or in fact not possible.

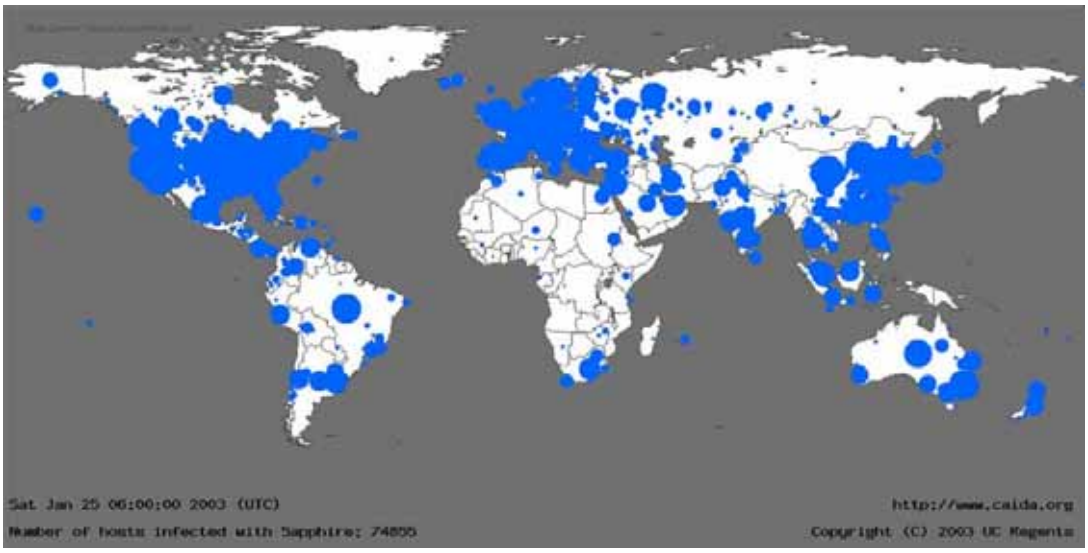

Figure 1 The geographic coverage of the sapphire worm on January 25th 2003. The photograph is courtesy ofwww.caida.org.

Furthermore, during the summer of 2003, the Blaster worm managed to infect more than 400,000 computers. Although not as rapidly spreading as Sapphire, Blaster was unique in the following sense: after its original release on the Internet, Blaster was quickly followed by Welchia, a good worm whose goal was to combat Blaster. Welchia exploits the same vulnerability as Blaster and after invading a computer system through this vulnerability, it attempts to patch the system in order to make it immune to Blaster. After patching a system, Welchia would try to spread and patch other systems on the local network. Unfortunately, Welchia was overly aggressive in searching for other vulnerable machines to patch, leading to a situation equivalent to an internal Denial-of-Service attack, with Welchia-infected computers continuously probing computers on the local network in an attempt to find whether they are vulnerable to Blaster. Ironically, it was not Blaster, but Welchia, that caused the most damage.

Worms like the Sapphire are usually called flash worms because they have the potential to conquer the entire Internet within minutes before any human intervention is possible. Interestingly enough, besides rapidly spreading worms, there also exist slowly spreading worms: the stealth worms. Stealth worms capitalize on the fact that automatic worm detection systems usually detect the spread of new worms - not the worms themselves. This is because new worms contain code unknown to worm detection systems, and thus can not be detected. Their spread however, is usually exemplified by a sudden increase in traffic and/or the existence of peculiar traffic patterns that can be detected. Stealth worms spread very slowly trying to elude automatic worm detection systems. Masquerading as ordinary traffic, stealth worms attach themselves to popular programs, such as peer-to-peer file sharing systems, and propagate along with the ordinary traffic of such programs.

In addition to worms, viruses are increasingly starting to represent a significant threat as well. Recent viruses were able to gain access to passwords, bank accounts, email messages and important personal information. While worms are self-replicating programs that multiply without any human intervention, viruses usually depend on human help in order to multiply. Viruses pose as interesting content attached to innocent-looking email messages, prompting the user to "click" on them. When the user "clicks" on the attachment, the virus starts executing and taking control of the

local computer. Figure 2 shows the number of viruses as reported by the F-Secure antivirus software running in numerous systems around the globe.

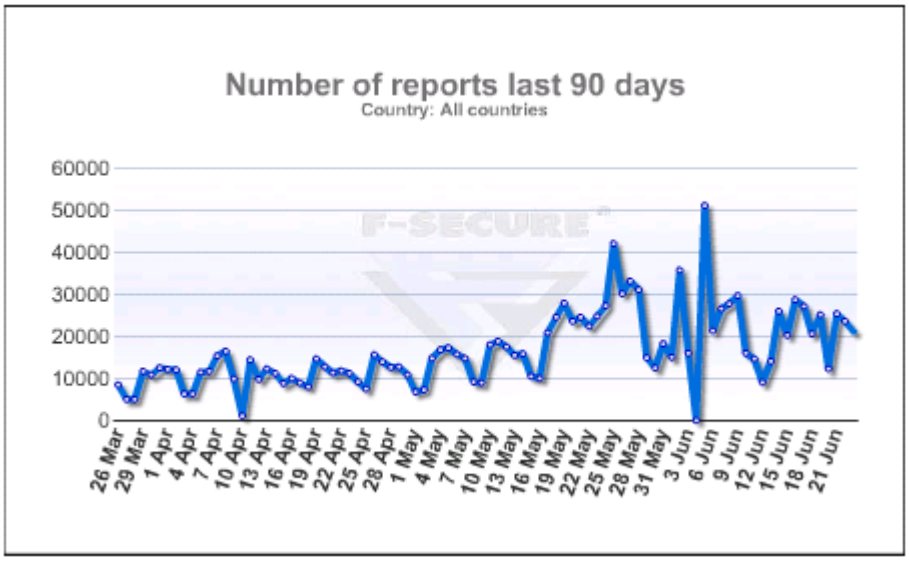

Figure 2 Number of viruses reported by the F-Secure network. [5]

One of the most interesting recent viruses, BugBear-B, installed a keyboard logger, a snooping program that was able to steal passwords and gain access to private information. Keyboard loggers are able to steal confidential information, such as credit card numbers, despite the fact that users may take all standard security precautions when communicating such information over the Internet, such as using secure socket layer (SSL) enabled services or a similar encryption mechanisms. This is because secure socket layer and similar mechanisms protect the information from snoopers that reside outside the user's personal computer, but not from snoopers that have penetrated the user's personal computer. Indeed, keyboard loggers are able to steal confidential information before it reaches the secure socket layer, and thus before it is encrypted

A closely related and emerging threat to Internet security is "spyware": malicious programs that, like viruses, install themselves on a user's computer, and report sensitive information (like online shopping habits, or passwords and credit card numbers) back to a third party. Spyware is usually distributed as part of an application (for example, a new P2P client) installed by the unsuspecting user, and while it may not actively spread like worms and viruses, it introduces a new and significant risk for online services, as illustrated in a recent analysis.

Although it is difficult to measure the damage caused by viruses and worms, some estimates put the cost in the order of billions of dollars. However, this damage may actually be small compared to what these attacks can potentially do, as illustrated in a recent study on so-called "Warhol" worms. Such worms can cause massive damage of unprecedented effect causing severe disruption to the Internet infrastructure and services. After spreading in less than 15 minutes to most of its potential victims, a Warhol worm could install itself in startup scripts so that it is always started when the machine reboots. It could also easily hide any traces of the infection, making it hard to detect post-facto, while resisting cleanup or patching. After establishing itself to several millions of computers, the Warhol worm would start a massive Distributed Denial-of-Service (DDoS) attack to major sites including antivirus sites that may

contain the patch for the Warhol worm, thus hindering users from recovering from the infection and protecting themselves. Effectively, the described Warhol worm could shut down the normal operation for most computers connected to the Internet.

Figure 3 shows the total losses caused by different types of security incidents. As one can easily notice, the most costly incident is the DoS attacks. DoS attacks is an issue that is quite often caused by worms. The detection of worms is the primary objective of the EAR system. DoS attacks is also an issue that the EAR project will attempt to address

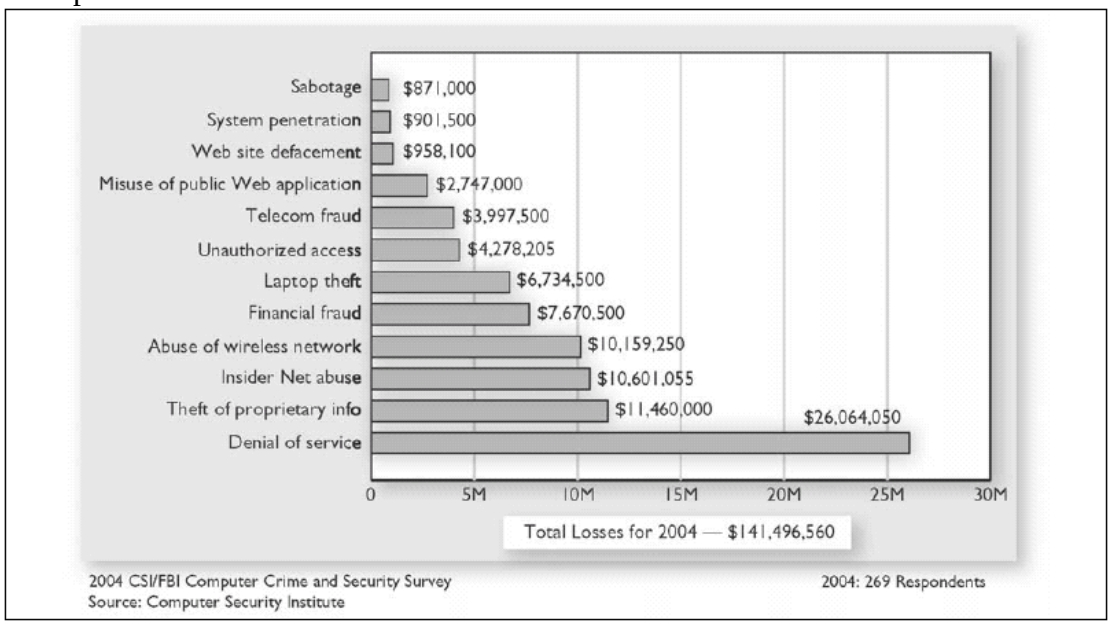

Figure 3 Total losses due to security attacks [2]

2. State of the art of attack detection mechanisms

2.1 IDS categorization

An intrusion detection system (IDS) inspects all inbound and outbound network activity and identifies suspicious patterns that may indicate a network or system attack from someone attempting to break into or compromise a system [4].

There are several ways to categorize an IDS:

- **misuse detection vs anomaly detection:** in misuse detection, the IDS analyzes the information it gathers and compares it to large databases of attack signatures. Essentially, the IDS looks for a specific attack that has already been documented. Like a virus detection system, misuse detection software is only as good as the database of attack signatures that it uses to compare packets against. In anomaly detection, the system administrator defines the baseline, or normal, state of the network's traffic load, breakdown, protocol, and typical packet size. The anomaly detector monitors network segments to compare their state to the normal baseline and look for anomalies.
- **network-based vs host-based systems:** in a network-based system, or NIDS, the individual packets flowing through a network are analyzed. The NIDS can detect malicious packets that are designed to be overlooked by a firewall's simplistic filtering rules. In a host-based system, the IDS examines at the activity on each individual computer or host.
- **passive system vs reactive system:** in a passive system, the IDS detects a potential security breach, logs the information and signals an alert. In a reactive system, the IDS responds to the suspicious activity by logging off a user or by reprogramming the firewall to block network traffic from the suspected malicious source.

2.2 Active Monitoring

Active monitoring is a broad term that collectively describes a family of monitoring methods based on sending probe packets from a sender towards a (usually cooperating) receiver. Based on the behavior and response to packet probes, the sender is able to infer several performance characteristics of the network, including latency, bandwidth, jitter and error rate. Active monitoring is currently being widely used in several countries. For example, RIPE NCC has installed more than 50 test boxes which periodically probe each other to find out the status and performance of their Internet connection. Besides RIPE NCC, GEANT, the pan-European research network also conducts performance monitoring through TF-NGN, which will continue within GN2: the new version of the GEANT network.

At the other side of the Atlantic, several organizations, including NLANR, CAIDA, Surveyor, NIMI, and SLAC have been working in active monitoring. Recently, Internet2 has started installing an active monitoring infrastructure known as E2E PIPES (End-to-End Performance Improvement Performance Environment System). Based on tools such as iperf, traceroute, and OWAMP, PIPES measures latency, bandwidth, and connectivity among various hosts connected in the Internet.

Active-monitoring infrastructures focus more on the performance and status of the network and less on identifying and warning about novel security attacks. In addition, active monitoring infrastructures do not have the necessary information needed to pinpoint the exact form and source of attacks. For example, although they might be able to infer major attacks through bandwidth disturbances, they are usually unable to

provide the source IP address(es) of the attack, the destination IP address(es) and port(s) of the attack, as well as the type of the attack itself.

Summarizing, although active monitoring is being widely used to identify performance characteristics of the Internet, it provides limited support for identifying and tracing novel attacks.

2.3 Passive Monitoring

In addition to active monitoring infrastructures, passive monitoring infrastructures have recently started to appear. For example, NLANR, the National Laboratory for Advanced Network Research in the United States has installed a large number of passive Internet monitors operating at speeds between 155 Mbps and 2.5 Gbit/s. In passive monitoring systems, network sensors capture all packets, including both headers and payload that pass through their monitored network. Based on the headers and payloads of captured packets, passive monitors are able to produce a wealth of information including high-level performance metrics, as well as detection of attacks.

Besides the US-based NLANR infrastructure, there also exist European passive network monitoring projects. SCAMPI [7], for example, is an IST-funded project that builds a hardware monitor along with the necessary system and application software to facilitate passive network monitoring at speeds as fast as 10 Gbit/s. There exist plans underway to deploy SCAMPI to several places in Europe through a new European project supported in part by the European Commission. SCAMPI and other projects may significantly enhance our understanding of Internet traffic including Internet-based attacks. Based on passive monitors, and having access to all network traffic including all packet's headers and pay loads, they have significant amounts of information that can be used to identify attacks. This information may even be used to automatically generate signatures for new - not previously seen attacks. Therefore, passive monitoring systems in general, may significantly help us towards improving our Internet security. However, passive monitoring systems have three major disadvantages: (i) they have high computational cost, (ii) they may have low accuracy due to lots of false positives, and (iii) they may be inadequate against sophisticated new types of attacks.

- **High computational cost:** Passive monitoring projects impose a significant, if not unbearable, overhead to their underlying computational infrastructure. Indeed, processing packets at current line speeds of 10 Gbit/s overwhelms most modern processors. Indeed, a back-of-the-envelope calculation suggests that modern processors have inadequate computing power to perform sophisticated monitoring functions at speeds as high as 10 Gbit/s.
- **False positives:** in order to detect attacks, passive monitoring systems are sometimes based on heuristics that look for changes in the traffic patterns, such as a sudden increase in the number of TCP SYN packets per second. Although such changes may indicate attacks, they may also be due to legitimate reasons, including interesting breaking news, popular software updates, and flash crowds. Using only information available to passive network traffic monitors it may be difficult to distinguish an attack from a legitimate traffic increase, and therefore, security monitoring systems may incur a larger number of false positives, i.e. events that look like attacks but are completely legitimate changes in traffic patterns.
- **Sophisticated new types of attacks:** modern attacks get increasingly sophisticated by, for example, encrypting or otherwise obfuscating their code so that they will be undetectable bypassive network monitoring systems. This

method is frequently used by the so-called polymorphic worms and viruses, which encrypt their body using a different key each time they try to infect a new computer. Therefore, all instances of a polymorphic worm/virus in the network will "look" different from each other, hindering the recognition by an antivirus or an intrusion detection system. Therefore, even if passive monitoring systems capture all the headers and payloads of all the network packets that carry the worm/virus, they will have a difficult time recognizing it, since all copies of the worm/virus will look different from each other.

2.4 Honeypots

In order to track attackers and recognize new types of attacks at their infancy, security scientists have developed honeypots. A honeypot is a computer system that does not provide a regular production service. A schematic of a typical honeypot is shown in figure 4. Under normal conditions, a honeypot would be idle, neither receiving nor generating any traffic. If honeypots receive any traffic, it means that they are likely to be under attack, since no ordinary user would initiate any connection to a honeypot. Similarly, if honeypots generate any outgoing traffic, this means that they may have been compromised by an attacker, who uses the compromised honeypot to launch further attacks. Therefore, honeypots can be thought of as decoy computers that lure attackers, into an environment heavily controlled by security administrators.

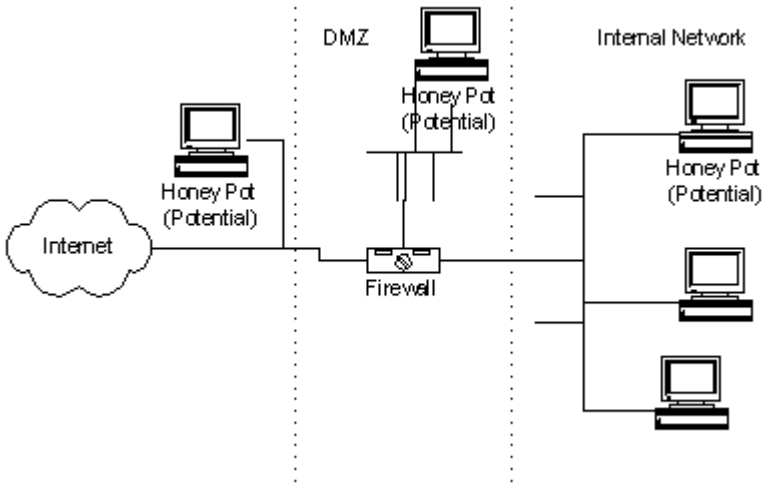

Figure 4 Honeypot schematic [3]

Although the concept of the honeypot was already known in the early nineties, it was not until 1997 that the first honeypot software came out: Fred Cohen's Deception ToolkitToday there exist several honeypot systems, including commercial products, such as Specter and ManTrap, as well as open source systems such as honeyd. Honeypot systems are usually divided in two broad categories: low-interaction honeypots, and high-interaction honeypots. Low-interaction honeypots usually emulate a service, such as a remote login service, at a rather high level. For example, when the attacker invokes the remote login service in a low-interaction honeypot, the system responds with a login: prompt and a password: prompt, where the attacker may enter a login and a password. Then, the honeypot records the attacker's IP address as well as the login and password (s)he used to enter the system. After the honeypot records the attempted attack, it rejects the remote login attempt and possibly

terminates the connection shutting the attacker out of the system. A high-interaction honeypot, on the other hand, does not emulate but it rather implements the services it provides. Thus, a high-interaction honeypot that provides a remote login service would actually let the attacker log into the system, in case (s)he provided the correct login/password combination. The purpose of the high-interaction honeypot is to let attackers into the system in order to study their methods and possibly the preparation of their future attacks. Although high-interaction honeypots provide a wealth of information about an attacker's methods and future plans, they pose a significant risk to an organization's infrastructure, since they may be used by attackers to launch more attacks and/or compromise more systems. Although low-interaction honeypots do not suffer from this risk, they provide limited information about an attacker's methods and tools.

One of the latest high-interaction honeypots that have been developed so far is the honeynet. A honey net, developed by the "Honeynet Project" is a set of honeypots where each one of them runs a different operating system or a different service. All honeynets are located behind a firewall that lets attackers into the honeypots but restricts their outgoing connections so that compromised honeypots can not be used to launch more attacks.

2.5 Shortfalls with current IDS

While the ability to develop and use signatures to detect attacks is a useful and viable approach there are shortfalls to only using this approach which should be addressed.

1. **Variants.** As stated previously signatures are developed in response to new vulnerabilities or exploits which have been posted or released. Integral to the success of a signature, it must be unique enough to only alert on malicious traffic and rarely on valid network traffic. The difficulty here is that exploit code can often be easily changed. It is not uncommon for an exploit tool to be released and then have its defaults changed shortly thereafter by the hacker community.
2. **False positives.** A common complaint is the amount of false positives an IDS[1] will generate. Developing unique signatures is a difficult task and often times the vendors will err on the side of alerting too often rather than not enough. This is analogous to the story of the boy who cried wolf. It is much more difficult to pick out a valid intrusion attempt if a signature also alerts regularly on valid network activity. A difficult problem that arises from this is how much can be filtered out without potentially missing an attack.
3. **False negatives.** Detecting attacks for which there are no known signatures. This leads to the other concept of false negatives where an IDS does not generate an alert when an intrusion is actually taking place. Simply put if a signature has not been written for a particular exploit there is an extremely good chance that the IDS will not detect it.

[1]Intrusion Detection System

2.6 The weakness of existing defense mechanisms

Although there exist tools and systems that can help us protect our infrastructure from attacks, these tools are usually limited to combating only known forms of attacks. Antivirus systems can protect users against known viruses, but are usually helpless when confronted with a new computer virus. A recent report by the Department of Trade and Industry in the UK revealed that 93-99 % of UK companies have antivirus software in place, yet 72% received worms, viruses or trojans, and half of them suffered from an infection or DoS attack. Similarly, Intrusion Detection Systems can generate alerts for known forms of worms but are of little help when confronted with previously unknown attacks. Thus, we need a security infrastructure that is able to detect new forms of attacks, and allows scientists and engineers to study, analyse and rapidly develop defenses, and has the critical mass to detect new types of attacks as early as possible.

2.7 Firewalls vs Intrusion Detection Systems

A common misunderstanding is that firewalls recognize attacks and block them. This is not true. Firewalls are simply a device that shuts off everything, then turns back on only a few well-chosen ports. A firewall is not the dynamic defensive system that users imagine it to be. In contrast, an IDS is much more of that dynamic system. An IDS does recognize attacks against the network that firewalls are unable to see.

For example, in April of 1999, many sites were hacked via a bug in ColdFusion. These sites all had firewalls that restricted access only to the web server at port 80. However, it was the web server that was hacked. Thus, the firewall provided no defense. On the other hand, an intrusion detection system would have discovered the attack, because it matched the signature[2] configured in the system.

Another problem with firewalls is that they are only at the boundary to your network. Roughly 80% of all financial losses due to hacking come from inside the network. A firewall at the perimeter of the network sees nothing going on inside; it only sees that traffic which passes between the internal network and the Internet.

The example above demonstrates the need for an IDS system. Some other reasons for adding IDS functionality to a firewall are:

- Double-checks misconfigured firewalls.
- Catches attacks that firewalls legitimate allow through (such as attacks against web servers).
- Catches attempts that fail.
- Catches insider hacking.

[2] A unique string of bits, or the binary pattern, of a virus/worm. The signature is like a fingerprint in that it can be used to detect and identify specific viruses/worms. Anti-virus software uses the virus signature to scan for the presence of malicious code. [6]

3. System Overview

The EAR system will try to address the problem of Internet worms and tackle the shortfalls of the existing systems.

The following sections provide a short overview of the system to serve as a context for understanding some of the requirements presented later in this document.

The system operates as a network tap monitoring traffic directed from and to a set of local area networks. It inspects the contents of the monitored traffic and detects worms by identifying common substrings. The result of worm detection is a signature that can be used to block the worm. Unlike traditional NIDS[3] systems, the signature is generated automatically without the need to involve a human person. The exact details of the detection mechanism will be described in an appropriate forthcoming design document, but the main ideas revolve around reducing false positives and increasing performance.

The length of a typical worm varies from 400 bytes to several kilobytes. Very often, however, only part of the payload is sent directly from the infecting host to the targeted victim. We will refer to the first part of the traffic as the "attack". The compromised target downloads the rest of the worm contents by connecting back to the source host. The attack length can be smaller than the total length of the worm, or equal if the entire worm is contained in the attack. The system will focus on detecting the attack of the worm.

A worm performs multiple attacks from multiple infected hosts to multiple targets concurrently. Attacks to different targets that belong to the same worm are similar, and they typically contain common strings. The early warning system will rely on this similarity to identify strings that belong to a worm attack.

Worm traffic travels through the network in the form of network packets. Because worm traffic is similar in the bytes level, very often it also consists of similar packets, or packets of the same length. In this case, it would be possible to check packets as a whole for finding a worm. However, although a worm may happen to have large substrings in common between different instances, it may not have entire packets. Recently, the Witty worm has used random padding of packets, thus deliberately preventing identical packets, or packets of the same length. Therefore entire packets are too course-grained for worm detection and the system should process packet substrings instead of entire packets.

A worm can spread by any protocol that is used by a vulnerable service, the protocol is the same as the one used by the service whose vulnerability is being exploited by the worm. For example, a worm that exploits an application over HTTP, will use the TCP Internet protocol for its attack, while a worm that exploits Microsoft SQL Server, will use the UDP protocol. Worms have been created that spread over TCP as well as others that spread over UDP. The system will miss worms that use an Internet protocol not monitored by the system. However, the possibility of a worm using an Internet protocol other than UDP or TCP is unlikely.

Worm	***Total Length***	***Attack Length***	***Protocol***
Witty	600 Bytes (+ padding)	600 Bytes	UDP
Sapphire/Slammer	376 Bytes	376 Bytes	UDP

[3] Network Intrusion Detection System

Worm	*Total Length*	*Attack Length*	*Protocol*
CodeRedII	3.8KBytes	3.8KBytes	TCP
Welchia	10KBytes	1.7KBytes	TCP

3.2 Future worm characteristics

One concern about future worms is that a worm may easily fragment its packets to hide similarity of the stream contents by using packet boundaries to obscure its patterns. For this reason, the system should process reassembled TCP streams and look for similarities at the stream level and not the IP packet level.

A virus can obfuscate or encrypt its body with a different key each time and include a small decryptor in the beginning, so the only constant portion is the decryptor. Such a virus is called polymorphic. Furthermore, the decryptor can use each time one of multiple equivalent instruction blocks for each one of its instructions, so the decryptor itself is not constant. This is called metamorphism. It has been proposed that future worms will be polymorphic, or even metamorphic, and they will not contain constant portions of content. Such techniques have been used with traditional viruses but no such worm has appeared to this day. It is unclear how such worms can be contained in the network level.

Furthermore, a worm could spread so slowly as to be undetectable. Such a worm is called a stealthy worm. This is opposite to today's worms, which try to spread as fast as possible in an attempt to render any human-mediated response impossible. Our system does not aim to fight stealthy worms. Instead it tries to defend against fast worms, that humans alone cannot fight.

4. Functional Requirements

The aim of this Early Detection system is to provide a comprehensive view of the internet security landscape. It should alert organizations of attacks with target their infrustructure and offer detailed analysis in order to mitigate the risk. As Internet threats appear with increased speed and in greater numbers each day, it is more important than ever for corporations to be vigilant in monitoring the current threat environment. EAR will provide customized and detailed notification of vulnerabilities and malicious code as they are discovered. Corporations will be able to protect from emerging methods of attack. The EAR system will help to protect networked PCs, critical systems, and users from worms. Furthermore, as (D)DoS are often initiated by worms it is reasonable to assume that (D)DoS will be reduced. In this section we present the functional requirements, what the system should be able to do, the functions it should perform.

4.1 Attacks detection

In this section we present the types of attacks the system should successfully detect. Attacks which do not fall within the requirements presented here may evade the system.

F1.1: The system should detect attacks that use unfragmented packets, as well as fragmented packets to spread their payload over multiple packets in order to obfuscate their signatures. Strings of small length are often popular without belonging to a worm. For example, many unrelated HTTP requests contain the string “GET /”, or “HTTP/1.1”. Therefore, a minimum detectable string length must be established.

F1.2: The system should detect worms with an attack that contains at least 300 consecutive bytes.

F1.3: The system must detect worms that spread over the TCP protocol. However, it is highly desirable, but not initially required, to include support for UDP worms as well.

F1.4: The system is not expected to detect stealth worms.

F1.5: The system is not expected to detect completely polymorphic or metamorphic worms, unless requirement F1.2 applies. The only constant portions in polymorphic worms can be the decryptor and the conversation with the vulnerable service. In the case of metamorphic worms, possibly only the conversation remains constant. If neither of these is long enough (F1.2), detection of such a worm is impossible with our system.

4.2 Detection delay

The detection delay is a crucial parameter of the system. The system must be fast enough to detect the attack, to alert the administrators or the users and to prevent possible damages.

F2.1: The detection delay of the system, measured in elapsed attacks before an

alert has been triggered, should be evaluated theoretically and experimentally for worms with different levels of aggressiveness.

4.3 False positives

F3.1: Timely detection is worthless in the presence of false positives, therefore the system is required to have a zero false positives rate.

F3.2: As a last means of preventing false positives, the system should support a white-list that allows handling persisting false positives. Strings listed in the white-list should not be considered as worm signatures.

4.4 Configuration – customization

The system will detect strings that appear frequently as worms. The threshold used for determining what is frequent affects the sensitivity of the system, and therefore its detection delay, as well as the probability of false positives. This is a tradeoff and it should be possible to adjust this at will.

F4.1: It should be possible to adjust the sensitivity of the system using a threshold. The system could be adjusted for faster detection with the cost of the false positives rate going up.

F4.2: It must be possible to adjust the amount of information that will be recorded, so as to cater for those that worry about their privacy being revealed.

F4.3: It should be possible to configure whether log-files, result packets, or both will be used to report alerts.

F4.4: It should be possible to configure the log-file location and the destination to which result packets are sent.

4.5 Security constrains

The purpose of the EAR system is to provide detection of certain type of attacks. However, there are certain security requirements that must be met by the system itself in order to allow deployment in the production environment of an ISP, an organization or an enterprise.

F5.1: The network where the system is hosted must not be exposed to vulnerabilities because of the presence of the early warning system.

F5.2: The system must also be resilient to malicious traffic targeted to attack the system itself. Finally it must be carefully engineered so that it will be immune to any kind of attack. This is particular important as a poorly configured/engineered system can pose a huge security risk for the whole network. Thus is essential that the system is programmed with security practices in mind.

4.6 Privacy issues

The system's objective is to protect the corporations and the end user from malicious attacks by worms. The results produced by the system should not compromise the

privacy of any monitored entities. The end user's privacy is equally important to the security provided by the system.

F6.1: The data gathered should be strictly used for analyzing, identifying a possible attack, implementing ways to protect and for absolutely no other reason. The EAR framework will operate in a way that the data do not have to be shared among various individuals and/or companies thus minimizing the risk of revealing important information about the end-user or corporate data.

F6.2: The data analysis must take place within the organization data center which is considered to be a trusted body.

F6.3: The issued alerts and reports must not include any information that will reveal the identity of the user (eg. the IP addresses belonging to the infected hosts). Furthermore, only content that belongs to traffic that has been identified as traffic initiated by a worm is going to be included by the system in alerts.

4.7 Integration with other systems

The system's sole purpose will be to provide an early identification of possible security problems. In addition, it will operate as a stand alone system meaning that it will be cut off from the rest of the world and is not aware of a new worm appearing at another network.

F7.1: The system should be able to integrate with an external application that will take over the task of transmitting the alerts to the administrators.

F7.2: The necessary information should be recorded in log files. The format of log files should be specified in detail when the system has been developed completely, it shall contain enough information to create a filtering signature. The task of the application will vary according to the kind of output required and the actions that need to be taken. The general objective will be to provide the administrators with an early warning on impending attacks, as well as a method of quickly applying some sort of defense.

F7.3: Thus, the final system may include a centralized configuration, deployment, installation, reporting, alerting, logging and policy management.

5. Performance Constrains

In this section we outline the required performance characteristics of the early warning system that will be implemented

5.1 Monitoring capacity

A security system can be placed in-line or not. An Intrusion Detection System for example is typically not placed in-line. Therefore it can issue alerts but cannot, for example, discard packets. The performance of the network is not affected by such a system. On the other hand, a firewall or an Intrusion Prevention System must make a decision for each packet and therefore the performance of the network is affected by the performance of such a system.

P1.1: The system must be capable of operating at 100 Mbits/sec. However, it is highly desirable to achieve operation speeds up to 1 Gbit/sec.

P1.2: The system does not operate in-line and therefore does not impose any limitations or delay on the traffic capacity of the monitored network.

P1.3: It should be possible to operate the system by processing only part of the traffic.

6 Deployment

6.1 Software and hardware platform

D1.1: The system will be developed for the GNU/Linux operating system and the x86 hardware platform.

D1.2: It should be easily portable to other platforms and Unix-class operating systems.

D1.3: The system will require a dedicated machine for production operation.

6.2 Placement

D2.1 The system will not be placed in-line, but will operate as a network tap, processing all traffic visible to its network interface. Therefore, it will not be able to interrupt the operations of other production systems.

6.3 Monitored area

D3.1: The networks to be monitored will be selected by mirroring the appropriate traffic to the monitoring system's network interface.

D3.2: The number of the hosts that the system shall be capable of monitoring will be in the order of hundreds.

D3.3: The system must be able to see symmetric traffic. Sometimes routers only see one direction of the traffic. This is called assymetric routing. Our system will not have to tackle with this.

6.4 Software distribution

D4.1: The system relies on the Snort NIDS [8] and is distributed as a Snort plug-in in the form of a non-intrusive patch against the standard Snort distribution. This means that there are no compatibility issues with existing systems, thus it will be easy to deploy.

6.5 Initial setup and periodic updates

D5.1: Initial installation will require building the software from source. New versions or configuration file modifications may require stopping and starting the system on software level.

6.6 Hardware performance requirements

D6.1: The system should be designed to operate on commodity PC-class hardware.

Sample specifications:
3GHz Processor
1GByte RAM

7. Competitive analysis

There is a lot of research undergoing in the field of internet security. This has resulted in a few commercial products offering different levels of protection. However each one is quite unique in the way it tackles the problems and provides a valid solution.

We are aware of only one commercial product that offers similar features with the system we are developing: Silicon Defense has developed Counter Malice, an enterprise-scale containment system based on detection and containment of scanning worms. They separate the enterprise network into cells that communicate through worm containment devices and are quarantined in case of infection. The system relies on the fact that with a low vulnerable/probed host ratio, a scanning work can be detected and contained at a faster rate than it can spread.

We should emphasize, however, that all methods based on scan detection provide reasonable level of defense against scanning worms but are of limited use against hit-list worms or worms that discover targets without scanning. In contrast, our work is able to detect hit-list worms, as well as worms that, due to random scanning, do not appear scanning. Also, the signatures produced by our system can be used effectively at a different point in the network to fight against worm coming from different source hosts. Containment based on addresses on the other hand can only be used to contain attacks from hosts already identified as scanning.

9. Conclusions

The damages suffered every year due to cyber attacks are in the range of millions of euros. The current protection mechanisms prove inadequate to provide a reliable solution due to various problems. The EAR project aims to tackle these problems and provide a viable solution for the early detection of new worms. Upon project completion, the network operators will have a valuable tool that will supply them with real time warning mechanism. The final system may include a centralized configuration, deployment, installation, reporting, alerting, logging and policy management.

The EAR system will be used for the detection of worms appearing in the monitored network. However, in the case of a wide deployment of the EAR system, it is reasonable to assume that the (D)DoS attacks will be also reduced. It is the hosts that are infected by a worm that cause the (D)DoS attacks, causing corporations to lose time, money and prestige.

The system is going to detect worms that probably use new techniques such as packet fragmentation and some types of polymorphic or metamorphic worms. This guarantees that the system should be of use for the years to come. It is also important to note that the detection mechanism employed will not impose any delay to the data flow providing for an uninterrupted network operation. The false positives rate must be zero and there must be a timely detection assuring a reliable system. The system will operate with respect of users privacy and corporate sensitive data. Furthermore, the system will rely on the open source Snort NIDS system and will be distributed as a plug-in. This is quite important as there is already compatibility with existing systems, thus it will be easy to deploy.

10. References

[1] http://www.sans.org/resources/idfaq/data_mining.php
[2] http://i.cmpnet.com/gocsi/db_area/pdfs/fbi/FBI2004.pdf
[3] http://www.sans.org/resources/idfaq/honeypot3.php
[4] http://www.robertgraham.com/pubs/network-intrusion-detection.html
[5] http://www.f-secure.com/virus-info/statistics/
[6] http://www.webopedia.com/TERM/V/virus_signature.html
[7] http://www.ist-scampi.org/
[8] http://www.snort.org/