Alternative Engine to Detect and Block Port Scan Attacks using Virtual Network Environments

Walter Fuertes **Patricio Zambrano** **Marco Sánchez** **Pablo Gamboa**

Post Graduate Program, Escuela Politécnica del Ejército, Sangolquí – Ecuador

Summary
Currently, IP networks are constantly harmed by several attack techniques such as port scans, denial of service, brute force attacks, etc., which can collapse the continuity of business services. To address this problem, this paper focuses on an alternative solution for detection, block, and prevention of port scanning attacks. Particularly, this implementation is an alternative engine to automatically block specialized tool scans, namely PSAD (Port Scan Attack Detector), but it is conceptualized differently from the features that the program offers. To carry out this work, we have designed and implemented a virtual network environment that is to be configured as an experimenting platform with port scan attacks. To neutralize such attacks, we performed a security mechanism that takes the data reported by the PSAD and using parameterized variables (block time and level of category) automatic locks become viable, including custom records and notifications via e-mail. To validate our solution, several tests of port scan attacks have been run on public and private networks. Then we have compared the performance of our alternative engine with *ClearOS* (specialized security tool for Linux) and the PSAD. The results show that our alternative engine is faster and more reliable than the tools previously mentioned.

Keywords:
Network attacks, port scan attack, security, virtual network environments.

1. Introduction

Some of the biggest threats to the security network are the presence of bugs, viruses, Trojans, port scan, phishing and denial of service. These can cause your Web server or client to crash, corrupt your information, or, worst of all, allow outsiders unauthorized access [1]. These intrusions may render its resources inoperative and produce a loss of productivity, causing economic losses and compromising the business continuity. This paper centers its attention on *port scans* attacks since these attacks in actuality represent a considerable part of Internet traffic [2][3]. Thus, this research focuses on an alternative solution for the detection and blocking of port scan attacks performed on a virtual network environment (VNE) [4].

Within this context, the scientific community has demonstrated an ever growing interest in the implementation of solutions, for diminishing network security attacks making use of the virtualization technologies. Under this precept the work proposed by Keller & Naues [5], formulates the implementation of a collaborative security lab using virtual machines. Other works [6][7][8] propose virtual technology integration, with the purpose of securing a network through the implementation of a remote laboratory intrusion detection system. Other researchers [9][10][11] have used virtual machines based on the *Honey net* concept, as a security tool. Within the same scope researches have used virtualization platforms for disaster recuperation and mitigation of real IP attacks [12][13][14]. Regarding mitigation mechanisms of Denial of Service attacks (DoS), Fuertes et al. [15] exposes a research where IP real attacks were evaluated in order to detect and block DoS attacks using VNE. Within this scope Yaar & Song [16] details Internet filter rules (called SIFF). Lastly, Mirkovic & Reither [17] proposes D-WARD which is a Distributed Denial-of-Service defense system, the goal of which is the autonomous detection of these attacks using new traffic profile techniques.

To address the problem mentioned above, this work proposes the design and implementation of an alternate engine for automatic blocking to the already existing system inside the Port Scan Attack Detector (PSAD) [18] that will be more efficient as well as customizable. In essence, the alternative engine is an implemented routine which captures PSAD output data reducing the necessary time to analyze its register files and detect these attacks.

In order to carry out this work, all test infrastructures were conducted in a VNE using Virtual Box, a virtualization tool using virtualization software to be deployed on virtual machines destined for desktop computers and enterprise servers, which also implements full virtualization [19][20][21].

To validate our mechanism and as the main contribution, this paper proposes: *i)* to improve the system's response time when detecting port scans; and, *ii)* to configure a customizable algorithm that acquires PSAD data and notify the system administrator via e-mail.

The remainder of this paper is organized as follows: Section 2 presents the theoretical framework. Section 3 describes

Manuscript received November 5, 2011
Manuscript revised November 20, 2011

the architecture of the VNE implemented, the full process to achieve the mechanism to detect and block port scan attacks, and its response. Section 4 presents a comparison between the systems described above and evaluates the results. Section 5 discusses Related Work. Lastly, the Conclusions and Future work are given in Section 6.

2. Theoretical Framework

2.1 Port Scans

According to [23], a *port scan* is an attack that sends client requests to a range of server port addresses on a host, with the goal of finding an active port and exploiting a known vulnerability of that service. Historically most scan detection has been in the simple form of detecting *N* events within a time interval of *T* seconds. Port scanning is an exploration phase and is considered the first stage of a computer attack. The aim behind scanning is to find open ports on a system. There are a number of tools to accomplish this goal; however, there are few tools available to detect attempts to scan ports [24]. The most popular techniques for this type of attack are: TCP connect scanning, TCP SYN scanning, TCP FIN scanning, TCP reverse indent scanning, FTP bounce attack, UDP ICMP port unreachable scanning, SYN Stealth SCAN, ARP ping SCAN, among others [25][26].

Actually, several open source port scan detection tools exist. For example: Snort, Port Scanner, Honey pots, Scanlogd, PSAD, etc. For this experimentation we have chosen the PSAD since its register files can be analyzed, and the algorithm to reduce the detection times can be improved.

2.2 Port Scan Attack Detector (PSAD)

The PSAD is a collection of three lightweight daemons written in Perl and C, which are designed to work with the Linux firewall system to detect port scans and other suspicious traffic [18]. The PSAD makes use of the activity logs of *IPtables* to detect, alert, and optionally block port scan or any other suspect traffic [27]. On TCP scan, the PSAD analyzes the TCP flags to both determine the type of scan (*syn*, *fin*, *xmas*, etc.) and the corresponding options of the command line that could be used so that *nmap* (Network Mapper) can create its own scan [26].

2.3 Virtual Network Environment

Within the scope of this research, a VNE can be defined as a set of virtual equipment (both systems as end-network elements including routers and switches) connected collectively in a given topology deployed on one or multiple hosts, which emulates an equivalent system in which the environment is perceived as if it were real [4]. The VNE encapsulates a set of applications within a virtual network enabling service configurations for a specific network in a realistic way. In the case of this research we have used this concept because Virtualization platforms are a potential technology to reproduce a real network topology.

2.4 Virtualization with Virtual Box

Virtualization in essence is a technique to share hardware resources. It can be used to partition physical equipment to support multiple virtual machines [28], interconnect them, and share hardware resources, such as CPU, memory and input/output devices [29]. It provides an extra abstraction layer between the hardware and operating system (OS). The technique allows, via hardware, to have several guest operating systems of diverse types executing simultaneously [30]. Currently, there are several alternatives in software that make virtualization possible, one of these tools is Virtual Box, developed by a team of researchers and supervised by ORACLE. Virtual Box [31] is X86 virtualization software to be deployed on virtual machines destined to desktop computers and enterprise Servers. Virtual Box allows the execution of Operating Systems without any modifications, including all the software installed on them [20]. In this work, Virtual Box provides the infrastructure to deploy and manage a VNE which can be configured to emulate the execution of port scan attacks.

2.5 Instruction Detection System

The Intrusion Detection System (IDS) is a software application that monitors network activities for malicious packets or policy violations and produces reports for network security. This concept is included in this study because most IDS (i.e. *Snort*) can detect port scan attacks.

According to [32], the IDS can either be host based or network based. A host-based system looks for intrusions on that particular host. Most of these programs rely on secure auditing systems built into the operating system. Network-based systems monitor a network for the tell-tale signs of a break-in on another computer. Most of these systems are essentially sophisticated network monitoring systems that use Ethernet interfaces as packet sniffers. In conclusion, the IDS primarily focuses on identifying possible incidents, logging information about them, and reporting malicious attempts [33].

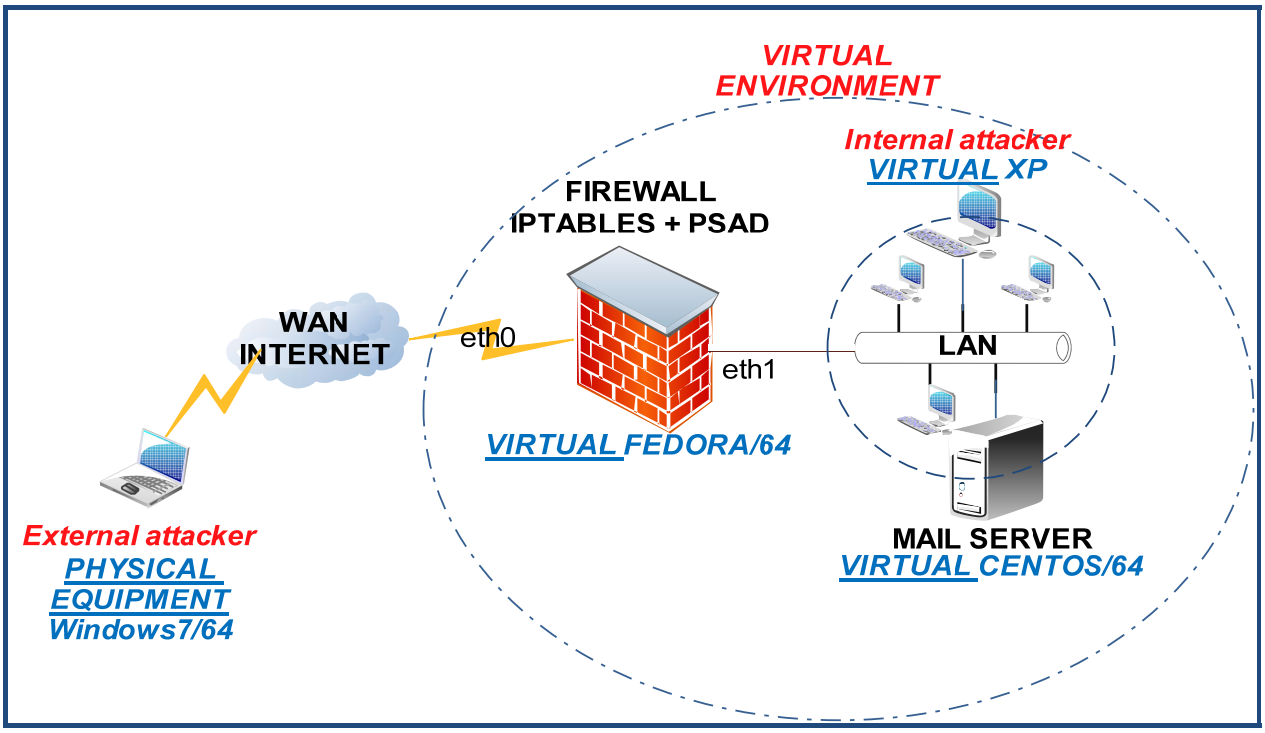

Fig. 1 Experimental Platform based on real and virtual network environment

3. Experimental Setup

This section has been divided into the following four parts: The experimental platform, the development of the mechanism to detect and block port scan attacks, the attack test environment implementation, and the system responses.

3.1 Experimental Platform

Figure 1 represents the experimental platform (i.e. real and virtual) susceptible to internal and external port scan attacks. The devices for internal and external networks and the hardware and software technical specifications used in this experimentation are described in Table 1.

As can be seen this hybrid topology includes a physical computer which puts all the virtual machines connected together into operation within a VNE. One virtual machine took on the Web server services. Another was configured as a firewall where the PSAD and the alternative engine were installed. An additional virtual machine accomplished the internal attacker functions to attempt to crash the Web Server. Furthermore, a physical computer was connected to the Internet which undertook the external attacker functions.

Table 1: Matrix elements of the experimental platform

VIRTUAL MACHINE - FIREWALL VIRTUAL BOX				Technical Specifications
INTERNAL NETWORK	**Adapter 1**	Bridge Adapter - physical interface (Gigabit Ethernet)	Configuration: IP 192.168.0.1 MASK 255.255.255.0 GW "empty"	**HARDWARE VIRTUAL** RAM: 2048 MB PROCESSOR: 4 NETWORK INTERFACES: 3 **SOFTWARE** SO: Fedora 12- 64bits PSAD: psad-2.1.5
EXTERNAL NETWORK	**Adapter 2**	Internal Network - intnet	Configuration: IP 10.0.0.10 MASK 255.0.0.0 GW "empty"	
MAIL SERVER	**Adapter 3**	Internal Network - intnet	Configuration: IP 192.168.0.10 MASK 255.255.255.0 GW 192.168.0.1	**HARDWARE VIRTUAL** RAM: 1024 MB PROCESSOR: 1 NETWORK INTERFACES: 1 **SOFTWARE** SO: Centos 5.3 - 64bits MAIL: ZIMBRA
VIRTUAL ENVIRONMENT PHYSICAL MACHINE - SPECIFICATION				
HARDWARE VIRTUAL RAM: 7158 MB PROCESSOR: i7 - QUAD CORE NETWORK INTERFACES: 1 Gigabit Ethernet **SOFTWARE** SO: WINDOWS 7- HOME SOFTWARE: VIRTUALBOX				
EXTERNAL ATTACKER PHYSICAL MACHINE - SPECIFICATION				
HARDWARE VIRTUAL RAM: 512 MB PROCESSOR: AMD-Athlon NETWORK INTERFACES: 1 Fast Ethernet **SOFTWARE** SO: FEDORA 12 SOFTWARE (ATTACK): NMAP				

3.2 Development of the mechanism to detect and block port scan attacks

Figure 2 shows a schematic representation of the process necessary to implement the mechanism to detect and block port scan attacks. Some of the elements involved for operation include: the establishment of a firewall, the attacker (i.e. *nmap*), the PSAD execution, and finally, the implementation of our alternative engine "security.sh".

To execute this mechanism, we have executed an exclusive script through two different means: on as a *Demo*, and another by configuring it as a scheduling tool in the *crontab* for automatic execution.

The *Demo* lets us generate a menu to setup the script data manually, in order to see the attacker blocking both in connectivity and in time. In other words, this type of execution is purely demonstrative. The *Demo* runs the script interactively with the user with the aim of understanding how it works.

Working with *crontab*, however, implies an automatic execution of the script (e.g. every 3 minutes) to see if there are any attackers, and if there are, block, register and report them to the network administrator.

As depicted in Fig. 2, the general form of operation of the mechanism is as follows: ***i)*** Run the firewall (i.e. firewall.sh) using the rules that follow the policies described in Table 2 below; ***ii)*** Initialize the PSAD (e.g. #psad -S with category 3). The PSAD supports blocking hosts by adding Iptables rules to special chains.; ***iii)*** Run the network mapper (*nmap*) (i.e. from the computer attacker); ***iv)*** Define the e-mail addresses for those who are going to be sent notifications (i.e. parameterized in the file security.sh); and finally, ***v)*** Run the script in *crontab* (# crontab -e), which is specified automatically each time the script reviews PSAD records, in our case 3 minutes. The full process and its elements will be explained in the following paragraphs:

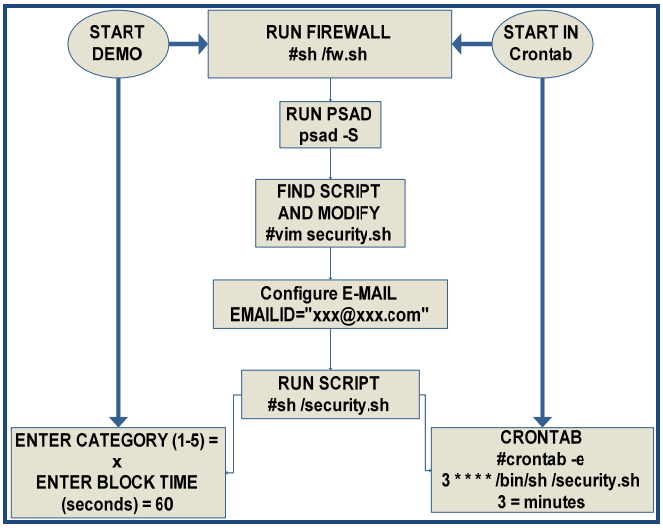

Fig. 2 Mechanism to detect and block port scan attacks (full process)

a. Implementing the firewall

This firewall is a script designed to prevent unauthorized access to or from the topology depicted in Fig. 1. All messages entering or leaving the network pass through the firewall, which examines each message and allows or denies the traffic based on specified security criteria. These criteria are registered in a file script, namely *IPtables*. Most Linux-based firewalls are just *IPtables scripts*. This firewall sets the policies for incoming packages to drop and defines rules for exceptions. This means everything is forbidden unless is explicitly allowed. It is worth mentioning that this script was set based on the functionality requirements of the PSAD. This *IPtables* evaluates: packets that are arriving at the computer from an outside source; packets that are being sent through the computer as a router; packets that are originating from computer and are being sent out (see Table 2).

Table 2: Firewall script

```
#!/bin/sh
echo -n
## FLUSH (rules)
iptables -F
iptables -X
iptables -Z
iptables -t nat -F
## We set the default policy
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT
################################################
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -i eth0 -p tcp --dport 25 -j ACCEPT
iptables -A INPUT -j LOG --log-level warn
################################################
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
echo 1 > /proc/sys/net/ipv4/ip_forward
psad -F
service psad restart
clear
```

b. Implementing the alternative engine

Figure 3 is a visual representation of the sequence of the alternative engine algorithm. The alternative engine is a routine implemented in Shell Script created to capture data provided by the PSAD, to achieve a more efficient alternative block. It is worth mentioning that the operation of the PSAD is based on a continuous review of packets flowing through the firewall, storing the results in the tool's own records which are compared with previously parameterized ranges.

The alternative engine we developed works as follows: ***i)*** PSAD variables are changed so that its mechanization does not perform automatic blocks; ***ii)*** The comparison variables (category level, block time) are adjusted; ***iii)*** Then the script begins to filter PSAD records, comparing them with previously established variables. If these variables meet attack condition, the script automatically blocks the aggressor for a set time; and finally, ***iv)*** Incidents of attack are reported using *Zimbra* [34], which is e-mail freeware. When a block is generated, the script immediately sends an e-mail warning to the address previously defined by the administrator. Note that the script also performs validation in cases where the attacker is a recidivist, thereby safeguarding the resources of the VNE in continuous blocks.

Since this solution is configurable, the two parameters used for comparison must be explained: the category level and the block time, within the conceptualization framework of the PSAD developer:

The *category level* of the attacker is the number of packages required by a port scanner to complete its work. (e.g. category 1: 5 packages; category 2: 15 packages; category 3: 150 packages; category 4: 1500 packages; category 5: 10,000 packages). As can be seen, the interrelationship between the firewall and the PSAD is very important for this categorization. Once the information is flowing through the firewall, the PSAD filters out these packets and applies a category depending on the number of packages found. Finally, the PSAD stores this data in a register that contains the same IP of the attacker.

Block time is a penalty applied to the attacker that does not allow it to have any kind of communication either internally or externally. Since this datum can be parameterized the administrator can block one or more attackers for hours, minutes, or even days. In the case of not wanting to ever give communication to the attacker again the script must be changed so that it always remains blocked.

3.3 Attack test

The proof of attack was developed on a virtual machine using a Linux Centos 5.3-x64 based system (see Table 1). The procedure to continue in the implementation was the following: Install virtual machines and configure IP addresses. Next configure the NAT, filtered by *IPtables*-packets; installation, operation, and development of script to capture information provided by the PSAD; configuration of the attacker with *nmap*, connectivity and attack-defense tests of the system; verification of blocks, unblocks, and delivery of e-mail alerts.

3.4 System responses

In developing tests, *nmap* was used, with different attacks such as: SYN Stealth Scan, ARP Ping Scan, ACK Scan, and TCP Connect. At first, *nmap* was able to detect connected devices, open ports, services and applications being executed, type of operating system and firewall, to name a few. As the number of attacks increased, the PSAD continued to categorize the attacker according to levels 1 to 5. These values are both parameterizable and configurable within the same tool, although not immediately blocked; it is here where the script will read such parameter to later on compare it against a preconfigured category variable (DL: 3). If within the parameters defined by the user (block time and survey time–as configured in *crontab*), the attacker will be immediately blocked (see Figs. 4 and 5).

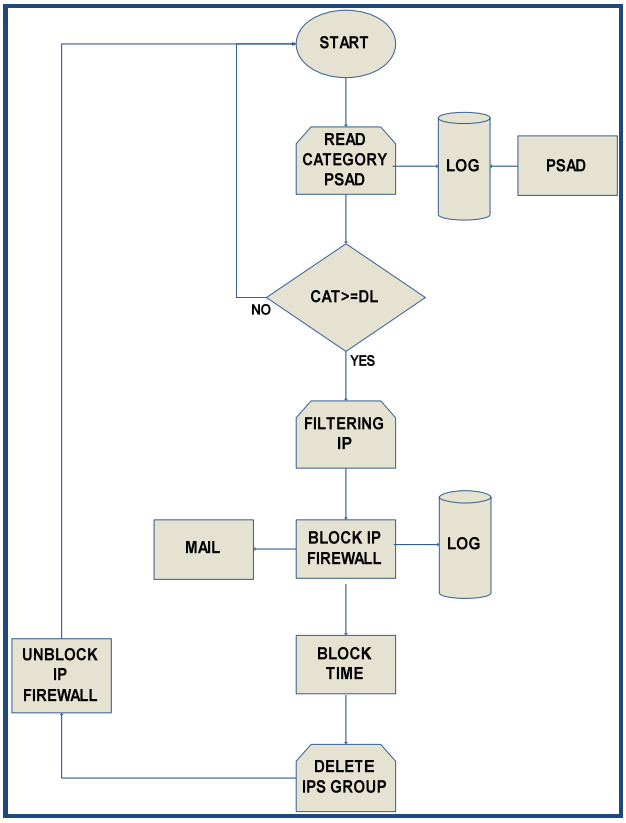

Fig. 3 Flow diagram of the alternative engine

4. Experimental Results and Discussion

This section describes the results obtained by comparing the effectiveness of our alternative engine with the PSAD and *ClearOS*. These results were taken during several tests on the experimental platform described in Fig. 1. The outcomes show the number of packets received, the response (detection) time, time of drop link, the CPU and memory consumption, and network performance, from various port scanning attacks. Finally in this section we explain an interpretation of the findings (Discussion).

4.1 Alternative engine blocking system vs. PSAD

Figure 4 shows the comparison of the detection mechanisms and its Detection time. As it can be seen, the detection time of our alternative engine is eleven seconds less than the PSAD. Within this context the Detection time is the period that the mechanism takes to react to a given input (i.e. drop connection when the attack was detected).

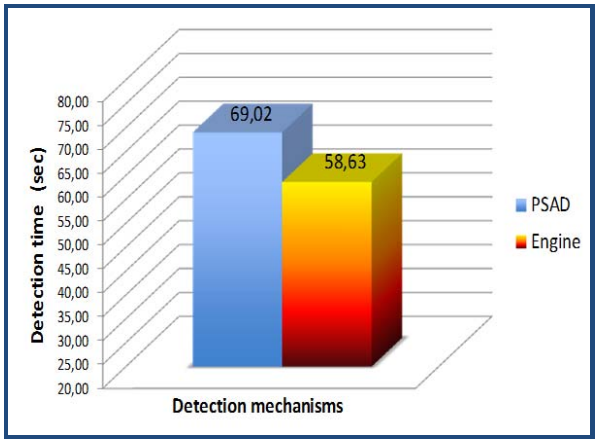

Fig. 4 Comparison of Detection time; Alternative Engine versus PSAD

Figure 5 shows the number of received packets during the time of connection, generated by the ICMP protocol throughout three different connectivity tests. This outline illustrate that the number of packets received using our proposal is less than those generated by PSAD.
As information, according to RFC 792, "ICMP messages are sent in several situations: for example, when a datagram cannot reach its destination, when the gateway does not have the buffering capacity to forward a datagram, and when the Gateway can direct the host to send traffic on a shorter route."

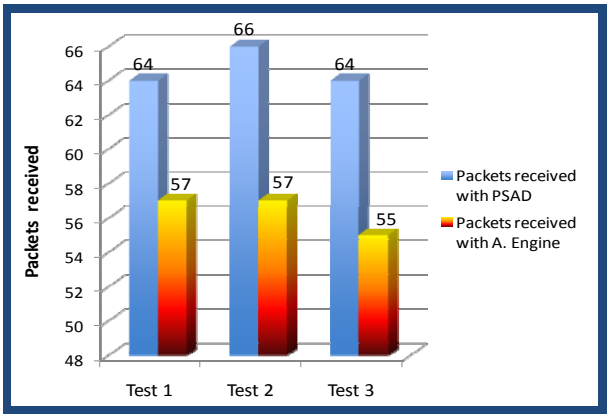

Fig. 5 ICMP packets received during the detection time

Figure 6 shows the time it takes for these detection mechanisms to categorize and block both the internal and external attacker (i.e. through the loss of connectivity) when conducting a port scan with *nmap* directly on the firewall. This illustration shows that the detection time with our alternative engine is less when compared with the PSAD. The connection time of the attack is not part of the

analysis given that the priority is the response time of the script.

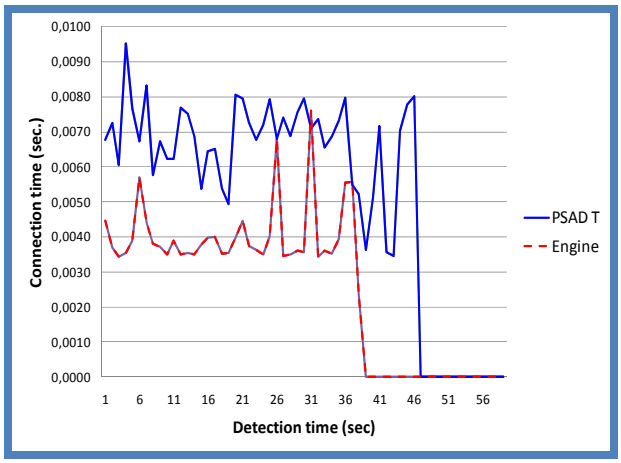

Fig. 6 Comparison of Alternative Engine versus PSAD.

4.2 Alternative engine blocking system vs. ClearOS

Figure 7 shows that *ClearOS* does not block the attack, while our solution detected and blocked immediately. *ClearOS* is a gateway server which comes with an extensive list of features and integrated security services [22]. Even though this open source tool takes less detection time as compared with our alternative engine, it is not effective. *ClearOS* only generates alarms instead, thus making it vulnerable and unreliable for these types of attacks.

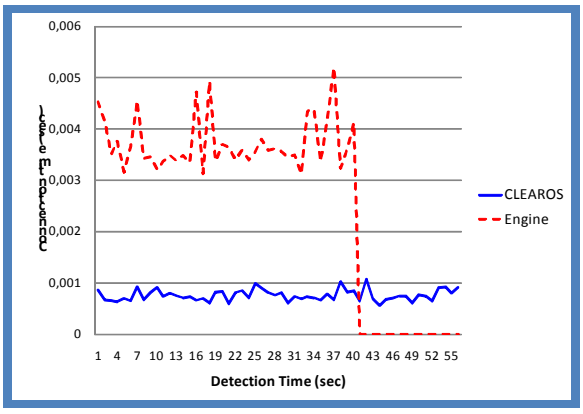

Fig. 7 Comparison of Alternative Engine versus ClearOS

4.3 Performance of PSAD Hardware vs. Alternative Engine

For the analysis of the performance of the VNE, a script was made, which filters data (the use of memory, CPU and capture time). Once this data is captured and processed, the performance between the PSAD tool and our alternative automatic blocking engine, are depicted in Figs. 8 and 9. The time allowed for the tests was 265 seconds, during this time the performance of the CPU and the consumption of the memory of the equipment used in the simulation were analyzed. The results obtained are explained below:

Figure 8 illustrates that in the first 65 seconds from the start of the attack, resource consumption is similar (i.e. between 0 and 65 seconds in tool activity), from there CPU resource consumption increases. PSAD behavior shows a continued stability, which does not happen with our solution. The reason for this is because the implemented SCRIPT consumes more system resources as it is making a steady cyclical census to determine the category of attack. Moreover, virtualization technologies also introduce a non quantified overhead, and CPU resources are shared in the system through the use of a virtualized structure.

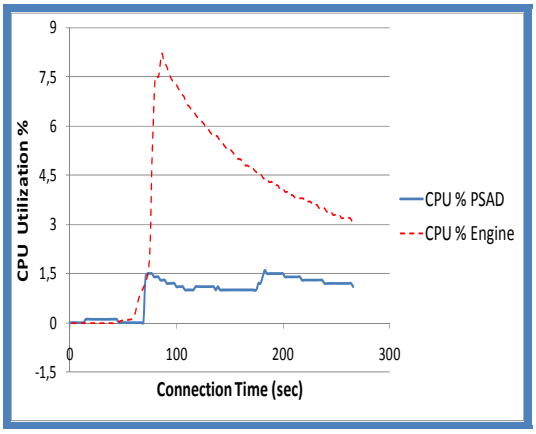

Fig. 8 CPU Consumption comparison during the attack.

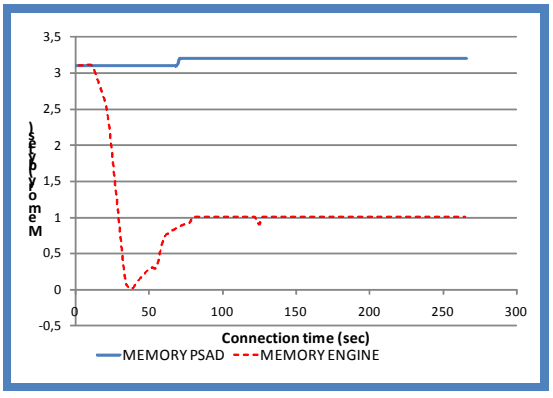

Fig. 9 Comparison of memory consumption during the attack.

Regarding the memory consumption analysis (e.g. data collection every 2 seconds), Fig. 9 shows that the PSAD tool takes more resources to stop an attack, which was not reflected in the alternative engine (i.e. our solution),

therefore being more efficient making an immediate block without requiring the use of virtual server memory.

4.4 Discussion

As an alternate solution for detection, block, and prevention of port scan attacks we have developed an alternative engine that will immediately send orders to execute the necessary block; its execution can be programmed through *crontab*. This order will be executed provided that comparisons between parameterizable values and PSAD records are performed, the latter being in constant communication with *IPtables* which continuously monitors the traffic generated.

As our experimental results have shown, our experimental platform has obtained interesting results to emulate the conditions of a real environment using a VNE. Furthermore, this paper has also demonstrated the viability of conducting these experiments. Moreover, given that a real similar infrastructure is not always available to reproduce real attacks for IP networks, this research has shown that virtualized environments can be used to emulate a specific network attack, whose results have been close to the real environment. This has also been demonstrated in our previous works for example in [4][20]. Therefore, the results of the experiment have provided qualitative data related to how the attacks work; the perceptual quality of mitigation mechanisms in terms of efficiency, resource consumption, and network performance. Thus, the use of VNE allows the execution of all the tests saving time and space in comparison with a real equipment environment.

The construction of multiple VNE helps network managers in the evaluation of security tools, without putting systems or servers in production at risk. Therefore, this implementation should be directed to the analysis and performance testing for the administrator to discern if these tools are attached to their security requirements or, if necessary, redesign them to perform specific functions. The proposal generated to be implemented in such environments does not jeopardize the security of the information, or the continuity of the workflow of active servers on a network in operation.

To highlight the advantages of the implementation of VNE, note that the main contribution of the project was the design and implementation of a reciprocating engine block based on a predesigned tool, an improved reaction time attaching it to the basic requirements of an administrator (quick release, support for events of attacks (LOGS) and the continuous information via e-mail) in a virtual network environment. This environment allowed for quick reinstallations of software and operating systems, achieving a seamless flow of project development.

Finally, from an educational viewpoint, this approach can be used to learn and teach computer network security and information assurance. As a final point, it should be take into account that this solution can be transported (i.e. portability) because they are virtual machines that can be ported to any equipment wherever required.

5. Related Work

There are very few papers discussing the effectiveness of using virtualization technologies as a research platform, in order to mitigate real attacks on IP networks. In the case of the educational field, the work proposed by Keller and Naues [5], explains a collaborative lab for experimentation in the security of networks using virtual machines. In the same field, other researches [6][7][8], propose the use of a remote lab for the integration of virtualization technologies for the IDS network security implementation. Another comparable research has been described by [9][10][11]. Here authors have used the concept of the *Honey net* over VMs as a security tool with the purpose of studying the techniques and motivations of hackers. In the same context, the work proposed by Damiani [12], describes a virtual laboratory based on the Xen platform, which is used for the configuration of a firewall in order to protect a server from *IPtables* external attacks.

Concerning Disaster Recovery through the use of virtualization, the work proposed in [13] demonstrates that the use of this technology, as an option to minimize server occupation so that network managers can dispose of an environment equivalent to the real hardware production network having flexibility and much lower costs of management and maintenance.

In another context, the work purposed by Ferrie [14] employed malicious code and service denial attacks against VMware, Virtual PC, Parallels virtual machines. However, in that research there are only recommendations instead of real solutions being developed. Comparing such works with current research we have implemented an alternative engine to automatically detect, control and mitigate such attacks.

In regards to the mitigation mechanisms of Denial of Service attacks (DoS), Fuertes et al., [15] exposes a research where IP real attacks were evaluated in order to detect and block DoS attacks using VNE. Within this scope Yaar & Song [16] details filter rules of Internet (called SIFF). Lastly, Mirkovic & Reither [17] proposes D-WARD which is a Distributed Denial-of-Service defense system the goal of which is the autonomous detection of these attacks using new traffic profile techniques.

6. Conclusions and Future Work

This research is based on actual port scan attacks performed on an IP network using virtualization technologies. Block

tests were conducted using the system's own tools as well as a security-specialized distribution. Results obtained were satisfactory as to the proposed alternative automatic engine block given that the specified objective was fulfilled along with a shorter response time and available adaption based on the user's needs. As an additional alternative, this gives the network administrator the option to choose from an internal block or the proposed block depending on their requirements. It became evident that Linux operating systems provide high scalability as part of their solutions; in the event that a user is not comfortable with a suggested solution, this user may also present improvements for the already established system. It is worth mentioning that most administrators focus their securities on blocking communication ports but do not take into consideration that the origin of attacks falls under exploration.

In this work we have designed and implemented a virtual network environment which was configured as a platform for experimenting with port scan attacks. To neutralize such attacks, we performed an algorithm that takes data reported by the PSAD and by using parameterized variables become viable automatic blocks that include custom records and notifications sent via e-mail (*Zimbra*). To validate our solution there have been several tests of port scan attacks on public and private networks. From there we compared the performance of our alternative engine with *ClearOS* and the PSAD. The results show that the alternative engine is faster and more reliable than the tools previously mentioned.

As for future work, we will be focusing on how to include the integration of our alternative engine into the Snort tool and the monitoring of its performance.

Acknowledgments

The authors would like to thank the comments and good advice of Martha López, who helped to significantly improve this paper. This material is based upon work supported in part by the Electrical and Electric Department-ESPE, under the Master's degree Program in Information Network and Connectivity.

References

[1] S. Garfinkel with Gene Spafford Web Security, Privacy & Comerse, O'Really Book. Second Edition. ISBN 0-596000-456.

[2] B. Lee, C. Roedel, and E. Silenok, "Detection and characterization of port scan attacks," vol. 2004. San Diego, CA.

[3] El-Hajj, W., Aloul, F., Trabelsi, Z., Zaki, N., Coll. of Inf. Technol., "On Detecting Port Scanning using Fuzzy Based Intrusion Detection System", UAE Univ., Al-Ain, 2008.

[4] W. Fuertes and J. E. López de Vergara, "An emulation of VoD services using virtual network environments,". In Proc. GI/ITG Workshop on Overlay and Network Virtualization NVWS'09, Kassel-Germany, March 2009.

[5] J. Keller, R. Naues, "A Collaborative Virtual Computer Security Lab," e-science, In Proc. Second IEEE International Conference on e-Science and Grid Computing, pp. 126, CA, USA, 12/ 2006.

[6] P. Li, T. Mohammed, "Integration of Virtualization Technology into Network Security Laboratory", In Proc. 38th ASEE/IEEE Frontiers in Education Conference, Saratoga, NY, 10/2008.

[7] T. Garfinkel and M. Rosenblum, "A Virtual Machine Introspection Based Architecture for Intrusion Detection". In Proc. Network and Distributed Systems Security Symposium, pps:{191—206}, 2003.

[8] K. Ali, "Algorizmi: A Configurable Virtual Testbed to Generate Datasets for Offline Evaluation of IDS", Electronic Theses and Dissertations, University of Waterloo, 2010.

[9] F. Abbasi, R. Harris, "Experiences with a Generation III virtual Honeynet", In Proccedings of the Telecommunication Networks and Applications Conference (ATNAC), 2009 Australasian, Canberra, ACT , ISBN: 978-1-4244-7323-6. May 2009.

[10] Fermín Galán, David Fernández, "Use of VNUML in Virtual Honeynets Deployment", IX Reunión Española sobre Criptología y Seguridad de la Información (RECSI), Barcelona (Spain), pp. 600-615, September 2006. ISBN: 84-9788-502-3.

[11] Hugo Fernández, Jorge Sznek, Eduardo Grosclaude, "Detección y limitaciones de ataques clásicos con Honeynets virtuales", Publicado en el V Congreso de Seguridad Informática 2009, (CIBSI'09), realizado e1 6 al18 de Noviembre, 2009, Montevideo, Uruguay.

[12] E. Damiani, F. Frati, D. Rebeccani, "The open source virtual lab : a case study". In proceedings of the workshop on free and open source learning environments and tools, hosted by: FOSLET 2006; pp. 5-12, Italy nel 2006.

[13] Co-innovation lab Tokyo, "Disaster Recovery Solution Using Virtualization Technology", White paper, http://www.cisco.com/en/US/prod/collateral/ps4159/ps6409/ps5990/N037_COIL_en.pdf.

[14] P. Ferrie, Attacks on Virtual Machine Emulators, Symantec White Paper, 2008.

[15] W. Fuertes, P. Zapata, L. Ayala y M. Mejía, "Plataforma de Experimentación de Ataques Reales a Redes IP utilizando Tecnologías de Virtualización", Memorias del 3er Congreso de Software Libre CONASOL-2010, Talara, Perú, Dic. 2010.

[16] Abraham Yaar, Adrian Perrig, Dawn Song, SIFF: "A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks", C. Mellon University

[17] Jelena Mirkovic, Peter Reiher, "D-WARD: A Source-End Defense Against Flooding Denial-of-Service Attacks", IEEE.

[18] Psad, http://www.cipherdyne.org/psad/ Last update: 02/23/2011.

[19] W. Fuertes, J. E. López de Vergara, F. Meneses, "Educational Platform using Virtualization Technologies: Teaching-Learning Applications and Research Uses Cases". Accepted for its publication in II ACE Seminar: "Knowledge Construction in Online Collaborative Communities",

Albuquerque, NM – USA, October 2009. ISBN: 978-0-9842912-1-2
[20] W. M. Fuertes and Jorge E. López de Vergara, "A quantitative comparison of virtual network environments based on performance measurements", in Proceedings of the 14th HP Software University Association Workshop, Garching, Munich, Germany, 8-11 July 2007.
[21] F. Galán, D. Fernández, W. Fuertes, M. Gómez and J. E. López de Vergara, "Scenario-based virtual network infrastructure management in research and educationnal testbeds with VNUML,", Annals of Telecom, vol. 64(5), pp. 305-323, 2009.
[22] ClearOS, [ONline] http://www.clearfoundation.com/. Last update, february 23, 2011.
[23] RFC 2828 "Internet Security Glossary", [Online:] http://tools.ietf.org/html/rfc2828. Last update: february 23, 2011.
[24] Gadge, J. Patil, A.A., "Port scan detection", Proceedings of 12th IEEE International Conference (ICON 2008), December, 2008.
[25] Avinash Sridharan, Ye, T., Supratik Bhattacharyya, "Connectionless port scan detection on the backbone", Dept. of Electr. Eng., Univ. of Southern California, Los Angeles, CA., 2006.
[26] Nmap, www.nmap.org. Last update: October 2010.
[27] Michael Rash, Linux Firewalls: Attack Detection and Response with IPtables, psad, and fwsnort, ISBN: 10.1-59327-141-7. 2007.
[28] T. Hart-Sears and J. Lofton, "Server Virtualization: The New Future for Midrange Implementation". White paper. Technology Partners International, Inc. July 2007.
[29] J. Humphreys and T. Grieser, "Mainstreaming Server Virtualization: The Intel Approach". White paper. June 2006.
[30] M. Tim Jones, "An overview of virtualization methods, architectures, and implementations". Emulex Corp. Longmont, Colorado, 29 December 2006.
[31] VirtualBox, [Online:] www.virtualbox.org/. Last update: October 2010.
[32] Simson Garfinkel, Web Security, Privacy & Commerce, 2nd Edition, By O'Reilly, Pub Date: November 2001, ISBN: 0-596-00045-6.
[33] Scarfone K, Mell P. Guide to intrusion detection and prevention systems (IDPS). NIST Special Publication 800-94; 2007
[34] Zimbra, [Online]: http://www.zimbra.com/. Last update: October 2010.

Walter Marcelo Fuertes Díaz currently works as Manager of Graduate Studies at the Escuela Politécnica del Ejército of Sangolquí-Ecuador. He is a professor-researcher in the School of Computer Science of that polytechnic, where he received the engineering degree in Computer Systems, in 1995. Then, he received his Master in Science degree in Computer Networking, from the Escuela Politécnica Nacional in Quito-Ecuador, in 1999, and the Ph.D. (honors) degree in Computer Science and Telecommunications engineering from Universidad Autónoma de Madrid (UAM), Madrid, Spain, in 2010.

Patricio Xavier Zambrano Rodriguez currently works as Network Administrator at the Tribunal Contencioso Electoral in the technology's area at Quito-Ecuador. He's a master student of Information Networks and Connectivity at the "Escuela Politécnica del Ejército" of Sangolquí-Ecuador, where he received the engineering degree in Telecomunications, in 2006.

Marco Polo Sanchez Aguayo currently works as Software Designer at the "Grupo TvCable" in the technology's area at Quito-Ecuador. He's a master student of Information Networks and Connectivity at the "Escuela Politécnica del Ejército" of Sangolquí-Ecuador. He received his engineering degree in Electronics, from the Escuela Politécnica Javeriana in Quito-Ecuador, in 2002.

Pablo José Gamboa Vargas currently works as a SCADA Telecommunications Engineer at the Corporación Nacional de Electricidad in Santo Domingo, Ecuador. He earned his engineering degree in Telecommunications at the Escuela Politécnica del Ejército in Sangolquí-Ecuador in 2006. He is currently studying a Masters degree in Information Networking and Connectivity, at the already mentioned institution.