Epsilon2

Virtual Information Security Testing System Mark II

De Luna, Lin G.
Computer Technology Department – College of Computer Studies
De La Salle University - Manila
Manila, Philippines
deluna.lin@gmail.com

Detera, Patrick Kevin G.
Computer Technology Department – College of Computer Studies
De La Salle University - Manila
Manila, Philippines
patrick.detera@gmail.com

Guerrero, Samuel David F.
Computer Technology Department – College of Computer Studies
De La Salle University - Manila
Manila, Philippines
samueldavid.guerrero@gmail.com

Mejia, Hiro R.
Computer Technology Department – College of Computer Studies
De La Salle University - Manila
Manila, Philippines
hiro.mejia@gmail.com

Gomez, Miguel Alberto N.
Computer Technology Department – College of Computer Studies
De La Salle University – Manila
Manila, Philippines
gomezm@dlsu.edu.ph

***Abstract*—Epsilon2 is a remodeling of the old Epsilon system. It utilizes virtualization technology in simulating physical networks resulting to effectively reducing resource consumption. The simulated networks are used for introducing Information Security concepts and practices to students and professionals alike. Improvements over the original system include: the simulation of more complex network topologies such as those that use DMZs to enable realistic threat simulations that conform to today's trends; the centralization of storage and system management which enables easier administration; utilization of a web application to conduct administrative tasks remotely; as well as utilization of the BitTorrent protocol for faster file serving over the network.**

KEYWORDS

Information Security, Virtualization, Vulnerabilities, Libvirt, KVM, Testing, BitTorrent

I. INTRODUCTION

Epsilon is a software system that provides an inexpensive hardware-independent solution to simulating information security networks. It makes use of virtualization technology in order to simulate real world scenarios with a library of virtual machines useful in creating an information security laboratory. This includes virtual machines host operating systems and applications that contain the most prevalent vulnerabilities seen nowadays. Along with that, the use of virtual machines instead of real machines helps maximize its flexibility and minimize the resources needed to implement a working laboratory. To effectively use these, the said system mainly has two basic components – the Epsilon Administrator and the Epsilon Server. The first one facilitates the management of the system, while the latter manages the individual host machines. With these components, the system will be able to perform several key tasks which include the deployment of different virtual machines across multiple host machines, and monitoring of user activities.

Although Epsilon is a cost-efficient alternative to physical laboratories, it is still limited in its function.[1] Over time there have been many more advancements in operating systems and applications, and there have been more discoveries of stronger and more persistent malware, making the scenarios it can emulate unrepresentative of most topologies available today. The repository of the virtual machine operating systems is decentralized which leads to a less organized system that is difficult to manage. Its current topology restricts the performance when adapted to newer technology. More functionality is needed as well as the expansion of the capabilities of Epsilon to be able to simulate the threat landscape that is constantly evolving - leading to new network-based and client-based threats.

Epsilon2 is a system developed to address these problems and to be more efficient so that the users will have an ease of use with the system when trying to learn information security concepts.

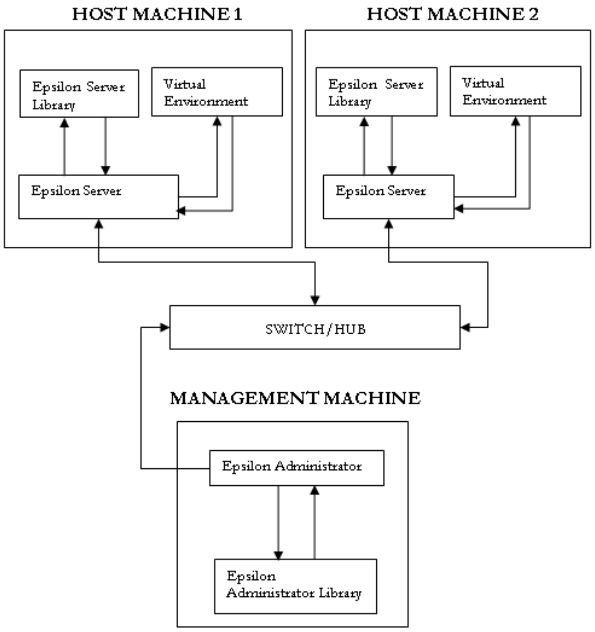

Figure 1. Epsilon System Topology

In Figure 1, it can be seen that each host machine has its own Server and Library. This results to decentralized resources, thus making it hard for the administrator to properly account the files being used by the whole system and read logs from the IDAs.

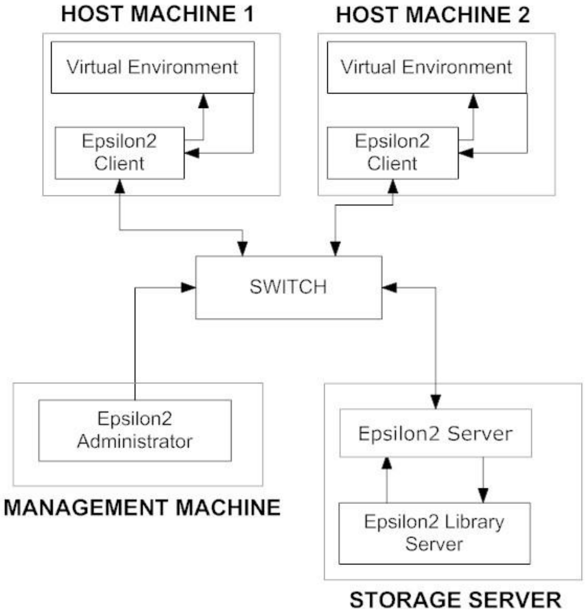

Figure 2. Epsilon2 System Topology

Figure 2 represents the modified system topology that is now used by Epsilon2. The individual Servers in the Host Machines have been migrated into a centralized Storage Server that is accessed by a management machine through the Epsilon2 Administrator web interface.

Comparing Figure 1 and Figure 2, one notices that the server libraries that were in different machines in Figure 1, are now consolidated in Figure 2. The justification for this approach was that it would provide a centralized repository of files needed by the system for easier auditing. It should also be noted that at the time the first Epsilon was developed, the technology was limited that it could not be consolidated into one machine.

This paper focuses on the improvements of the Epsilon2 system over the original Epsilon system. This includes a centralized repository of virtual machines, a web interface for administration, a torrent-dependent file serving system, capability to simulate complex networks such as a DMZ, new lists of vulnerabilities and operating systems that conform to the current (2012) industry, an update on Snort as the Network-based Intrusion Detection Agent (NIDA) and an upgrade to OSSEC as the Host-based intrusion detection system.

II. EPSILON ADMINISTRATOR OVERHAUL

In the old Epsilon System, the Epsilon Administrator is a program deployed and controllable only from one Management Machine. It controls and accesses the host machines in the network through Epsilon Servers deployed in each machine. Since this implementation appears to congest resources and be reliant on one physical machine to administer tasks, the Epsilon2 Administrator (e2Admin) is designed to be more flexible where it can be accessed by any physical machine in the network through a web browser.

The e2Admin is now a web interface accessible through any machine connected to the Epsilon2 System network and is hosted in the Epsilon2 Server (e2Server). It accesses the logs in the e2Server database as well as the file library in the e2Server machine. The e2Admin can be used to create and deploy topologies over the network, import virtual machines, and have control over the virtual machines of the client such as switching it on or off. On top of this, it can also be used to monitor the logs sent to the e2Server by e2Client IDAs.

III. SERVER CENTRALIZATION

The previous Epsilon System deployed one Epsilon Server per host, making the logs and files hard to track. Epsilon2 addresses this issue through Server Centralization.

Epsilon2 provides extensibility by centralizing the repository of virtual machines as well as the database that keeps track of the IDA logs from the e2Clients. The files located in the e2Server are the clean, initial images that are distributed over the network to be loaded by the e2Clients for threat testing.

The use of a torrent system allows faster transfer of large virtual machines and images over the network compared to manually transferring a file to each physical machine. The system utilizes TransmissionRPC as the torrent client and PeerTracker as the torrent tracker. It should be noted that the E2Server is the first seeder of a torrent file.

Centralization of the IDA logs enables easier monitoring of tests on the administrator’s side. The IDA in the E2Clients automatically forwards the logs to the database in the E2Server which can then be viewed by the administrator for monitoring the whole network.

IV. LIBVIRT ARCHITECTURE [2]

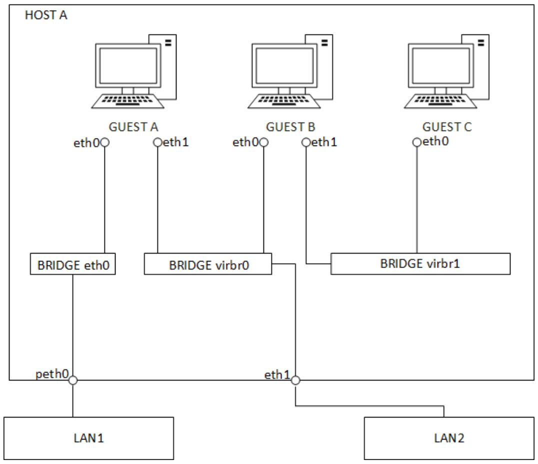

Figure 3. Physical Network Management Architecture

The diagrams in Figures 3 and 4 show a few of the network configurations enabled by the libvirt networking APIs used in creating the virtual networks:

- **VLAN 1.** This virtual network has connectivity to LAN 2 with traffic forwarded and NATed.
- **VLAN 2.** This virtual network is completely isolated from any physical LAN.
- **Guest A.** The first network interface is bridged to the physical LAN 1. The second interface is connected to a virtual network VLAN 1.
- **Guest B.** The first network interface is connected to a virtual network VLAN 1, giving it limited NAT based connectivity to LAN2. It has a second network interface connected to VLAN 2. It acts a router allowing limited traffic between the two VLANs, thus giving Guest C connectivity to the physical LAN 2.
- **Guest C.** The only network interface is connected to a virtual network VLAN 2. It has no direct connectivity to a physical LAN, relying on Guest B to route traffic on its behalf.

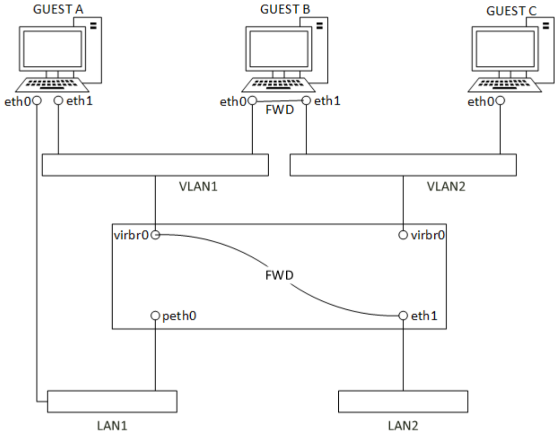

Figure 4. Logical Network Management Architecture

V. LIBVIRT CAPABILITIES [2]

Epsilon 2 utilizes libvirt for better virtual machine management. libvirt is a virtualization API which enables interaction with the virtualization capabilities of different operating systems. libvirt offers a more convenient way to manage virtual machines and offers various virtualization functionalities such as:

A. *VM Management*

This provides numerous development operations such as start, stop, pause, save, restore, and migrate. These features help Epsilon2 provide a better and simpler control over the virtual machines both locally and remotely.

B. *Remote Machine Support*

All libvirt functionalities are accessible on any machine running libvirt, including remote machines. This feature enables Epsilon2 administration to have control over different virtual machines remotely over the network. This feature is also the key to the management capabilities of the Epsilon2 Administrator through the web browser.

C. *Storage Management*

Any machine running libvirt can be used to manage different types of storage such as creating file images of different formats (qcow2, vmdk, raw, etc.), mounting NFS shares, enumerating LVM volume groups, and partitioning raw disk devices. This feature is utilized by Epsilon2 for extensibility and flexibility by being able to create and utilize virtual machines in different file image formats.

D. *Network Interface Management*

Any machine running libvirt can be used to manage physical and logical network interfaces. It offers different functionalities such as enumerating existing interfaces, configuring, creating and editing interfaces, bridges, and vlans. This provides Epsilon2 a necessary and easy control for the specifications of the interfaces on different virtual machines for networking.

E. *Virtual NAT and Route-Based Networking*

Any machine running libvirt can manage and create virtual networks. Virtual networks of libvirt use firewall rules to act as a router, providing VMs apparent access to the host machines' network. This provides Epsilon2 different networking capabilities such as creating different complex topologies by using the routing capabilities of virtual machines through the means of libvirt. The architecture found in Figures 3 and 4 enables these features.

VI. PERFORMANCE IMPROVEMENTS

Epsilon2 features a torrent system that reduces the time needed to transfer the virtual machines throughout the network. The test conducted uses a laptop acting as a server and torrent tracker and three desktops acting as clients downloading the file. The first test uses direct transferring of a file through a network shared folder. The second test uses the torrent system.

The torrent system makes use of the TransmissionRPC python module for controlling the torrent client, connecting to the Transmission JSON-RPC service running in each Epsilon2 Client. This is used for adding and removing torrents and starting torrent transfers. For the torrent tracker located in the Epsilon2 Server, PeerTracker is used.

Table I demonstrates the time it takes for a 1.6GB file to be successfully transferred from a server to multiple hosts through direct downloading. Testing for multiple downloaders has been done with the number of hosts simultaneously starting the download, as well as use of 10/100 Ethernet Cables.

TABLE 1. 1.6GB FILE DIRECT DOWNLOADING

Direct Downloaders	***Time***
1	2:55 (*175 seconds*)

Direct Downloaders	***Time***
2	4:35 (*275 seconds*)
3	6:49 (*409 seconds*)

The equation in (1) shows the approximate time that a download of a 1.6GB file finishes through direct downloading with single or multiple hosts simultaneously downloading the file, based on the initial result as seen in Table I. The variable α corresponds to the number of direct downloaders. 175 seconds is the average time it takes for a 1.6GB file to be transferred and for every simultaneous downloader, it has been inferred that the time increases by 150%.

$$\text{Time} \approx 175\ \text{seconds} \times 1.5 \,{}^\wedge\, (\alpha - 1) \qquad (1)$$

Initial seeders are the hosts that are seeding the complete file, as hosts that are downloading the file are also considered seeders because they automatically upload chunks of data. Downloaders are the number of hosts downloading the file at a time, and Time is the total time it took for all downloads to finish.

It should be noted that during the tests, the number of seeders connected to by the downloaders do not reach the actual number of seeders present in the network In the test conducted, at most there were five initial seeders in the network, but only two of them were acknowledged by the torrent client. This led to torrent clients whether downloading or seeding to enter an idle state which calls for additional research and verification as there could be miscalculations in the actual time it is needed for multiple hosts to download the complete file.

TABLE 2. 1.6GB FILE TORRENT DOWNLOADING

Initial Seeders	***Downloaders***	***Time***
1	2	4:32
	3	5:13
2	1	2:50
	2	2:38

However, comparing the data presented in Table II to that in Table I, the use of the torrent system is relatively faster compared to directly transferring the file over the network.

Transfer speeds average at 10.33 MB/s with one seed to one downloader, and slowly diminish as the number of downloaders increase. However it becomes difficult to gauge the exact speed of the transfer rate because of how the downloader also seeds the file, but it can be inferred that this method of file transfer is significantly faster when the number of downloaders and seeders scale up.[3]

VII. EPSILON2 USABILITY

The Epsilon2 system can be deployed in classroom laboratories, home offices and work environments. It is designed to simulate a physical laboratory network through the use of virtual machines. The virtual machines may contain different operating systems and applications which are deployed on clients running a Linux environment. The advantage of the Epsilon2 system over a physical network laboratory is that it requires less financial resources, and requires less time to set up the test environment.

The host machines contain virtual machines that are configured in such a way that they have several vulnerabilities which may be exploited or attacked. If the attacks occur, an Intrusion Detection System will log the event and will periodically send it to the Epsilon2 Server. In a classroom setting, the computers of the students will be running the Epsilon2 Client.

The Epsilon2 Administrator is a web application that can be accessed through any host machine in the network. The application allows for the logs to be viewed, and even perform administrative tasks such as starting or stopping a virtual machine remotely. The administrator also handles the distribution of the virtual machine files to the clients using torrents. In a classroom setting, the teacher can use the web application to monitor the client machines and help in their instruction.

The main goal of the experiments conducted on these virtual machines is to observe how the attack occurs and how it affects the system. This gives the user a better idea of how vulnerabilities and exploits work, and to some extent even mitigate the attack. It also introduces concepts of information security. The use of virtualization technology to simulate a network also allows for user creativity in creating the type of network topology needed for the experiment. It also allows a range of different operating systems, applications, and vulnerabilities to be used.

VIII. CONCLUSION AND RECOMMENDATIONS

The current implementation of Epsilon2 succeeds in introducing newer and more efficient technology to the Epsilon system that still retains in being an alternative to an information security laboratory. There are, however, more improvements that could be made on the system other than adapting to newer operating systems and threats such as:

1) *Implementing cloud storage:* to save hard disk space on the server and mitigate computer resource consumption which would allow the heavier processes like running the virtual machines to be faster.

2) *Additional compatibility on Windows-based environments:* to utilize a hypervisor that also works on a Windows environment, as the current hypervisor in use, KVM, only works in Linux environments.

REFERENCES

[1] M.A. Gomez and S. Wong, “Virtual Information Security Testing System (Epsilon),” Manila, 2006.

[2] Redhat. (2012). libvirt: Network management. [Online]. Available: http://libvirt.org/archnetwork.html

[3] R. Bharambe, C. Herley and V. Padmanabhan, “Analyzing and Improving BitTorrent Performance,” Microsoft Research, Microsoft Corp., Redmond, WA, Tech. Rep. MSR-TR-2005-03, Feb. 2005

[4] trigunflame, "peertracker - Simple, Efficient, and Fast BitTorrent Tracker," 1 January 2010. [Online]. Available: http://code.google.com/p/peertracker/. [Accessed 22 August 2012].

[5] VMWare, Inc., "Virtualization Overview," 2006.

[6] K. Scarfone and P. Mell, "Guide to Intrusion Detection and Prevention Systems (IDPS)," Gaithersburg, MD, February 2007.

[7] M. Roesch, "Snort - Lightweight Intrusion Detection for Networks," 2011. [Online]. Available: http://www.snort.org/docs/lisapaper.txt.

[8] M. Richmond, "ViSe: The Virtual Security Testbed," 2005.

[9] N. Sharma and S. S. Sran, "Detection of threats in Honeynet using Honeywall," *International Journal on Computer Science and Engineering*, vol. 3, no. 10, pp. 3332-3336, October 2011.