Computer Network Testbed at Binghamton University

Andrey Dolgikh, Tomas Nykodym, Victor Skormin
Binghamton University, Binghamton, NY, USA
adolgikh@acm.org

Abstract – The Network Testbed at Binghamton University was designed to facilitate security research in the area of automated IDS. It offers a secure, controlled environment for experimental analysis of the efficiency of various intrusion detection/mitigation and computer forensics systems. It allows for staging large scale experiments with real self-propagating malware on thousands of interacting heterogeneous nodes. This paper addresses some principles implemented in the Testbed design including the architecture, accessibility, security, and visualization. The Testbed provides effective ways to collect data representing the network and software operation. It facilitates secure time sharing of the hardware among different research projects. Its enhanced security is achieved by separation and hardening of the core services. The application of the Testbed is demonstrated by the following three experiments featuring novel IDS technologies: behavior-based IDS extracting predefined malicious functionalities from the system call data by semantic analysis, demonstration of the alarm propagation concept for the minimization of false alarms, and the detection of distributed low and slow attacks, and network-wide IDS capable of automatic detection of functionalities and statistically significant variations of their relative frequencies indicative of information attacks.

Keywords – Testbed, security research, behavioral based, intrusion detection

I. INTRODUCTION

Malware became a scourge of the Internet affecting its numerous users. The interest of modern cyber criminals ranges from unauthorized access to bank accounts to obtaining sensitive information (such as medical records, technology secrets, manufacturing layouts, business practices, etc.) [1]. Consequently, information security should be addressed within the system approach, starting from small embedded controllers and ending at a high end server at the data center.

Modern security research at the level of system protection includes risk analysis [2], vulnerability analysis [3], intrusion detection systems [4], and forensics systems. Particular research problems are commonly addressed by means of network simulation. Simulators or software frameworks usually abstract most of the details focusing on a few characteristic features of the problem. Unfortunately in the field of computer security this is not always beneficial. Malicious software usually demonstrates quite complex behavior which is hard to be simulated and correctly assessed on simplified models.

Therefore one of the substantial requirements in computer security research is the ability to conduct experiments fully reflecting the distributed nature of the attacks by malicious software deployed in large scale networks. It is often important to run unmodified malicious software in the controlled environment which is indistinguishable from the open Internet or real enterprise network.

A. Emulab

One of the first successful large scale projects in the network security research is Emulab [5] that gave rise to many spin-off projects sharing its base structure but featuring some additional functions. Essentially, Emulab is the “Internet in the room”. It allows simulating of thousands of arbitrarily interconnected nodes. It provides a user friendly interface for staging and executing large scale experiments in the Internet-like environment.

The Emulab Testbed consists of the following components:

- Experimental nodes are general purpose PCs connected to the programmable switch over several Ethernet data links. Experimental nodes are also equipped with a serial line, controlled power switch and separate Ethernet connection to the control network. Researchers are usually granted full access to the experimental nodes hardware.
- The programmable switch allows for arbitrary configuration of communication links among experimental nodes. Arbitrary configuration is achieved through the use of VLAN technology.
- A User server supports experimenter’s user accounts. It also provides access to the experiment space and allows collection, storage and retrieval of the user’s experiment data. This server enables researchers to access the Testbed.
- The Boss server is the backbone of the Testbed. It performs all security sensitive operations. It sets up VLAN configurations for the experiments, controls the power, and sets the time slices for the experiment.

In addition to the described above components, Emulab features specialized software which simplifies creation, launch and maintenance of the experiments. The first piece of software that greatly improves the speed of experiment setup is the solver for the network Testbed mapping problem. It allows for fast and efficient mapping of virtual resources requested by the researcher into physical resources available on the Testbed [6]. Another piece of software introduced as a part of Emulab is Frisbee [7]. Frisbee enables fast disk imaging across the whole Testbed. It is very important that a Testbed be shared among multiple researchers. Unfortunately, the experiment state might include multi-gigabyte hard disk drive images of all the experimental PCs. Scrubbing all this information from the disks and

The Binghamton University Testbed is funded by the Air Force Office of Scientific Research (AFOSR) under the DURIP program.
laying out new information on them is very time consuming. This prevents Testbed hardware to be effectively shared. Frisbee solves this problem by making disk imaging a matter of minutes.

Overall, Emulab proved to be a very successful Testbed system allowing a wide range of simulation tasks to be solved. Unfortunately, it does not provide a sufficient level of security. There are several features of Emulab that could be used with malicious intents. Emulab allows direct access to the Internet from the space of the experiment. It exposes most of the Testbed resources to the experiment space. For the experiments involving malware and offensive technologies this is unacceptable.

B. Deter

The Deter project [8] was also intended as a Testbed for security related research. It is based on the Emulab project and inherits the most of its design principles and software. However, it features additional measures to increase the level of security.

First, its design does not allow any direct link to the Internet from the experiment space. This was achieved by the introduction of the Gatekeeper firewall into the original Emulab scheme. Gatekeeper protects the Testbed from the external attacks as well as keeping potentially dangerous experimental programs from reaching the Internet.

Second, Deter protects internal infrastructure from the experiment space. This is achieved by the separation of the experiment and Testbed network spaces. This double protection introduces a lot of difficulties for the researchers. It makes it hard or almost impossible to access experimental machines from the Internet. To solve this problem, a serial line server was introduced. This allows access to the experiment PC’s serial lines from the Internet over the SSH protocol. Setting a whole experiment through a SSH line might be tedious task. In order to increase the quality and the ease of setting up experiments, Deter provides system software for the proper containment and monitoring.

In spite of the rich set of system configuration programs, Deter (and Emulab) users have limited access to the system hardware. For example, Deter has no usable remote connections to the graphics desktops. This creates difficulties for the researchers who use graphics centric operating systems like MS Windows. Although justified by the security considerations, this prevents junior-level users, especially student researchers, from developing a useful experience.

C. Binghamton University Testbed

The security and hardware access concerns could be simultaneously addressed in a virtualized Testbed, such as the one at Binghamton University (BU) [9]. It uses virtualization technology as the base management mechanism. The virtualization software separates the hardware platform from the system level software by introducing a controllable layer between them. This layer combines the agility typical of software-based simulators with the fidelity of simulators providing access to bare hardware.

In comparison with Emulab and Deter, the BU Testbed has several advantages. It provides an additional layer of security without decreasing the usability of the system, as well as the snapshot, record and replay functionalities unattainable without virtualization. It allows for running simultaneous multi-user interference-free experiments that utilize the hardware to the fullest.

The hardware architecture of the BU Testbed resembles the architecture of Deter. The difference between Testbeds lies in the extensive use of the hypervisor software to deploy experiments on the hardware. For example, three physical nodes of the BU Testbed can be used to configure a full blown experiment with more than twenty virtual nodes in the virtual network. The network configurations can be arbitrarily complex and may include virtual traffic shaping nodes, storage nodes, routers, host machines and servers.

The software part of the Testbed is implemented using the VMware Vsphere that provides a great level of configurability and agility.

II. THE UPGRADE

Increasingly common security research on the level of hypervisor prompted researchers to develop their own hypervisors or modify popular projects in order to achieve some particular tasks. However, a hypervisor cannot be installed under the control of another hypervisor that necessitates full access to hardware. In addition to this requirement, several design deficiencies emerged over time in the BU Testbed:

- It was hard to launch large scale experiments from the network-attached storage due to a massive unordered access from different physical nodes.
- The amount of RAM installed in the physical nodes did not allow for loading more virtual machines with modern Windows 7 due to high RAM requirements by this OS.
- The network connection speed between the nodes and the network-attached storage was not sufficient for collecting high volumes of data at a high rate.

One may notice that the same problems could be observed in Deter or Emulab.

In order to address above deficiencies and provide a firm foundation for large scale experiments we reconsidered the initial design of the BU Testbed in its hardware and software parts.

A. Hardware

The hardware upgrade was intended to resolve such issues as the lack of RAM, lack of high speed interconnect and lack of fast storage. This upgrade includes:

1. Installation of additional RAM. Now the Testbed has 720GB of RAM. This allows us to simulate up to 700 Windows 7 or Server 2008 nodes with reasonable amount of RAM per virtual node.
2. A high speed interconnection network was installed. Each server is connected to others via a 10Gbps interface. This provides great capacity for the aggregation of status/intercepted information from the nodes.
3. A high speed storage was installed. It is capable of making a fast snapshot of all disk drives installed in Testbed servers at 20 MB/s rate. It also means that some sensor information can be stored at this rate as well.

4. In order to provide sufficient space for the experimental data accumulated over time at slower rate, a high volume storage was added. It consists of two arrays. Each of these arrays is connected with a 10Gbit/s link to the high speed storage servers.

5. In order to provide an additional layer of separation for security reasons, additional configurable 1Gbps switches were installed.

Consequently, the hardware upgrade of the BU Testbed resulted in elimination of all bottlenecks found in the original system. The upgrade substantially increased the value of the Testbed as a simulation environment enabling it to simulate more hosts and obtain much more monitoring information than was possible before.

B. Software

There were major changes in the software architecture of the Testbed to support the new workflow model. Now the Testbed users can access most of the nodes to install software of their choice including custom hypervisors. To enforce security and assure the overall manageability a separate system core was designed. It provides the configuration, storage and communication services for the researchers. Since the changes in software are major and extensive, the new Testbed structure is described in the section below.

III. IMPLEMENTATION

A. Requirements

1) Functional requirements

The BU Testbed system should provide an easily accessible experiment space for computer security research. Therefore the following list of requirements should be met:

- Ability to prototype and deploy large scale experiments.
- Secure time sharing of the experimental facility among researchers.
- Ability to collect large amounts of information from the experiment space in near real time.
- Large set of preconfigured environments for the experiments.
- Large set of tools for security monitoring and reporting.
- Full access of the researchers to experiment machines (including access to graphics consoles).

2) Security requirements

Experimental deployment of malware requires that security measures be established and implemented in the hardware and software components of the system. The NIST Recommended Security Controls for Federal Information Systems [10] provides clear guidelines on assessing the information system impact and the appropriate security controls. As a result, the following set of security controls should be implemented and become an integral part of the system operation:

- Access Control
- Awareness and Training
- Audit and Accountability
- Security Assessment and Authorization
- Configuration Management
- Identification and Authentication
- Incident Response
- Maintenance
- Media Protection
- Physical and Environmental Protection
- Risk Assessment
- System and Communications Protection
- System and Information Integrity

B. Design

To meet the above requirements the Testbed team made several design decisions which shaped the current implementation of the Testbed. One of the main problems was the organization of the experiments.

1) Experiment workflow model

The implemented workflow model grants full access to the hardware resources participating in one particular experiment. At the same time the experiment is securely separated from all other experiments. The workflow consists of the following steps:

1. The user creates template images of the operating system that will be deployed on the participating nodes.
2. The user customizes network connectivity for the experiment by configuring the network switches to provide the desired network topology.
3. The Testbed software deploys templates created by the user onto some or all participating nodes.
4. Launching the experiment: all participating nodes are brought up and running.
5. Gathering experimental data. The software utilized by the researcher can gather and store high volumes of data onto connections provided by the Testbed storage.
6. Suspending the experiment:
   a) All participating machines are shut down.
   b) System scrubbing software is booted onto each experiment node.
   c) Disk images from experiment nodes are stored in the central storage by the scrubbing software.
   d) Network connectivity information is stored onto central storage.
7. Resuming the experiment:
   a) Power on the participating machines and boot system scrubbing software.
   b) Fill disks with images stored on the central storage.
   c) Restore the network connectivity.
   d) Return to Step 5.

We must point out that experiments suspended at Step 6 and resumed at Step 7 might not be the same. It allows for the secure time sharing of the Testbed among several researchers.
With additional precautions like blanking switch configuration and instant erasure of hard drives, the security leaks in the experiment space might be brought down to zero.

2) Testbed hardware layout

The overall hardware design is featured in Figure 1. To adopt the chosen workflow model and meet some of the security requirements presented in the previous section, the hardware was assembled in a system that supports three modes of operation:

1. Experiment configuration (Steps 1, 2). In this mode the user can upload the relevant data and preconfigure images of operating systems to be booted in the experiment.

2. Experiment preparation/transition (Steps 3, 6, 7 in workflow). In this mode the network configuration is deployed. The disks of participating machines are being filled with templates or images of operating systems from the storage. In this mode no experiment or external program can be executed.

3. Experiment launch/running (Steps 4, 5). In this mode any possible network connection that crosses the experiment safety border is forcefully disconnected by physically shutting down network ports (see Figure 1). In addition, the storage accessible from the inside of the experiment is isolated. Isolation is dropped when experiment is stopped to provide researchers with access to experiment data. When the experiment is running, the only available control for the researcher is the remote graphics or text console.

The described scheme of operation assures that the domains of potentially dangerous experiments are reliably separated from external networks and from each other. This separation provides an opportunity to grant a researcher full unrestricted access to hardware resources.

The IP KVM component provides a secure and efficient link enabling the user to control such an isolated system. It also provides the ability to access participating nodes during the boot process.

The hardware design is featured in Figure 2. As one may see, participating nodes are connected with storage servers and with each other via a high speed (10 Gbps) interconnect. This feature results in a very short transition phase between experiments because of high speed disk imaging. In addition, the Testbed users get the ability to monitor and record experimental data at extremely high rates not achievable earlier.

To keep up with the high speed network, two high speed storages were installed. These storages are backed by slower but larger storages.

3) Testbed services layout

The functionality and security requirements of A.1) and A.2) are met by a specific design for the Testbed backend services. We split backend services needed for successful operation of the Testbed into four domains: core, configuration, execution and time sharing. The communication between them is reduced to necessary minimums: some domains do not even run concurrently at the same moment of time. Backend services are distributed among five dedicated servers using the Xen virtualization platform [11] chosen as a main virtualization solution for several reasons:

- Xen provides great level of flexibility by allowing to redistribute services on physical services.
- The overall security design is hardened through an additional layer of separation.
- The robustness and reliability is increased by failover and snapshot functionalities.

The distribution of backend systems to domains is shown in Figure 3. The core services domain includes: authentication and authorization services (LDAP, Kerberos), naming service (DNS), time service (NTP), installation service, and firewall. The configuration domain comprises: storage server with template images of operating systems, storage server for accessing the results of previous experiment execution, IP KVM, VPN...
and firewall. The time sharing domain has two storage servers, remote boot server, IPMI management server, scheduling server, DHCP server and firewall. The most dangerous execution domain contains only the storage server accessible later as a disk image from the configuration domain.

![Diagram of service domains](image)

Figure 3. Testbed service domains

IV. EXPERIMENTS

In this section we will describe three experiments fully demonstrating the capabilities of the Testbed and its potential for security research. The first experiment features a network-based IDS detecting computer worms. The second experiment demonstrates a novel behavior based IDS extracting predefined malicious functionalities from system call data by semantic analysis. The ability to perform network-wide alarm correlation facilitating the detection of distributed low and slow attacks was also tested. The third experiment demonstrates the feasibility of network-wide IDS capable of automatic detection of functionalities.

A. Experiment 1. Worm propagation over Windows network

This experiment was intended to validate and debug a novel behavior based intrusion detection system utilizing a semantic approach to detect propagation of network worms by their behavioral footprint in the domain of system calls [12].

During this test, attacks perpetrated by various self-propagating malware were deployed within the network comprising 200 Windows hosts. The malware was represented by experimental worms with propagation engines of the real worms, and equipped with some reporting functions instead of payload. For the purpose of the experiment special “Attack station” software was created. It allowed for launching attacks and consequent real time monitoring of the response of the experimental IDS and “reports” of the propagating worms.

The Testbed was an indispensable tool during the system development. It allowed us to tear down hosts damaged by worms and deploy fresh Windows OSes in a matter of minutes.

Isolation framework provided a safe environment for launching unmodified malware.

B. Experiment 2. Behavior based detection of malware in a Windows network

The Testbed IDS implements a behavior-based approach centered on the concept of functionality, i.e. a sequence of computer operations defined not by its specific implementation, but by its outcome. It reflects the notion that malware achieves its malicious goals through its malicious functionalities that could be dynamically detected. While the number of such functionalities is quite limited and most of them are known to security experts, this approach has a great potential for IDS developers. The functionalities of interest could be extracted from system call data, therefore it is required to bridge the semantic gap between the functionalities and system calls at the stage of functionality definition, and between system calls and functionalities, at the detection stage [13].

The Colored Petri Net (CPN) was chosen as the enabling technology for the resultant system. CPNs are very functional for both the off-line functionality definition and the real-time functionality detection. To implement the CPN mechanism, a generic CPN module was developed. The CPN module enabled us to expand the IDS to all participating nodes in the network and to correlate security events at a designated server.

The CPN-based system consists of interception modules installed on the protected hosts and a CPN module installed on the IDS host. The CPN module was programmed to implement a CPN capable of detecting different types of malicious functionalities:

- self-propagation through executables, e-mail, and vulnerabilities
- installation of user-mode keyboard or mouse hooks
- slow step by step malicious operations
- network wide malicious operations

The overall design of the CPN analyzer system is presented in Figure 5.

![Diagram of CPN based analyzer](image)

Figure 4. Overall design of the CPN based analyzer

During the experiment, the participating Windows host population was split into several groups. The first group contained software to simulate normal user activity (opening folders, copying some files, editing documents from shared folders, and executing files). The second group was idle. The third group emulated a community of users launching common
software infected by viruses, trojans, worms, etc., adopted from a modern malware collection. The experiment demonstrated the efficiency and dependability of the Testbed IDS. It performed monitoring the software activity and successfully utilized CPNs for the detection of the predefined malicious behavior patterns. It is important that the low overhead of the proposed technology allows for the aggregating the host alarms over the network (alarm propagation concept) thus addressing the issue of false positives and low and slow attacks. The BU network provided a convincing proof of the feasibility of this IDS feature [13].

C. Experiment 3. Automatic detection of functionalities

The recognition of functionality as a key for behavior definition prompted the development of a technology for automatic extraction of the most common functionalities exposed by the software [14]. Perhaps, this can lead to the establishment of the software “normalcy profile” that could be implemented as a new paradigm in behavioral IDS. The functionality extraction has been implemented on the basis of the frequent subgraph search [15]. This implies monitoring all activities performed by programs on Windows hosts. Over time, a database of system call domain traces of various functionalities implemented by programs is to be established. This database can be processed to obtain the statistics for particular functionalities implemented by programs. Having these statistics is very helpful for anomaly based malware detection; indeed, malware attacks would change the statistics and/or show a deviation from the “normal” database.

To validate the described approach, we composed a network of Windows nodes containing three host types, as described in the previous section. Novel system monitoring software was installed on the nodes to collect extensive amounts of operation data. During the test we manually launched malware samples on several nodes. High network bandwidth available at the Testbed enabled us to collect and process substantial volume of data while maintaining the normal operation of the hosts under the test. This experiment pushed the limit to the hardware and software of the Testbed. Each experiment node produced up to 40MB/s of raw data. This resulted in up to 1GB/s of storage traffic. The profiles in some cases covered hours of operation resulting in terabytes of data.

Since graph processing algorithms [15] required substantial amount of computing power and RAM we didn’t run the analysis online. Profiles were analyzed on the same Testbed hardware in the consequent analysis phase (which was treated as just another experiment). Switching between experiment and analysis phases were done according to Testbed workflow by reimaging node hard drives with corresponding OS.

V. Conclusion

The BU Testbed provides researchers an access to a unique set of the hardware and environment. The Testbed architecture was designed to support cutting edge experiments in the field of computer security. The Testbed has several signature advantages over some existing experimental facilities. It provides a greater level of security without impacting the overall usability. The users can enjoy access to high speed storage and fast interconnect (10Gbps). The backend infrastructure of the Testbed provides a useful set of tools to increase the speed and quality of the experiment prototyping. The Testbed proved to be highly instrumental in a series of research projects, enabling the BU research team to test and fine-tune a set of novel computer network security technologies.

The BU Testbed is available for the entire community of computer security researchers, students and practitioners.

VI. Acknowledgments

Binghamton University Testbed was funded by the Air Force Office of Scientific Research (AFOSR) under the DURIP program. The authors are grateful to Dr. Robert Herklots of AFOSR for supporting this effort, to Dr. A. Volynkin for provided expertise and advice during the Testbed upgrade, and to Dr. A. Tokhtabayev for valuable suggestions and friendly discussions.

VII. Literature

[10] Recommended Security Controls for Federal Information Systems, NIST special publication 800-53, Revision 3, National Institute of Standards and Technology Gaithersburg, MD