Experimental Computer Network Testbed for Information Security Research

Skormin, Victor A.

Binghamton University
(State University of New York)

Watson School of Engineering
Electrical & Computer Engineering
4400 Vestal Pkwy East
Binghamton, NY 13902

Air Force Office of Scientific Research
Suite 325, Room 3112
875 N. Randolph Street
Arlington, VA 22203-1768

Available to general public

The Testbed is designed to facilitate security research in the area of automated IDS. It has been upgraded in terms of hardware and software, including architecture, accessibility, security, and visualization. It 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. Various schematics of the testbed are provided. After the upgrade, the testbed offers a secure, controlled environment for experimental analysis of the efficiency of various intrusion detection/mitigation and computer forensics systems. Its advantages over the most commonly used testing facilities are discussed. The Testbed allows for staging large scale experiments with real self-propagating malware on thousands of interacting heterogeneous nodes. The application of the Testbed is demonstrated by three experiments featuring novel IDS technologies developed at Binghamton University.
TABLE OF CONTENTS

1.0 SUMMARY .................................................................................................................. 1
2.0 INTRODUCTION ........................................................................................................ 2
  2.1. Emulab .................................................................................................................. 2
  2.2. Deter .................................................................................................................... 3
3.0 BINGHAMTON UNIVERSITY TESTBED ARCHITECTURE ..................................... 5
4.0 THE UPGRADE FOR TESTBED. BU TESTBED V2 ............................................. 7
  4.1. Hardware ............................................................................................................ 7
  4.2. Software ............................................................................................................. 9
5.0 IMPLEMENTATION ................................................................................................. 10
  5.1. Requirements .................................................................................................... 10
    5.1.1. Functional requirements ............................................................................ 10
    5.1.2. Security requirements ............................................................................. 10
  5.2. Design ................................................................................................................. 10
    5.2.1. Experiment workflow model ..................................................................... 11
    5.2.2. Testbed hardware design ......................................................................... 12
    5.2.3. Testbed services layout .......................................................................... 14
6.0 ONGOING WORK ....................................................................................................... 16
  6.1. Disk imaging ....................................................................................................... 16
  6.2. Network reconfiguration .................................................................................... 16
  6.3. Visualization ....................................................................................................... 16
    6.3.1. Hardware .................................................................................................. 16
    6.3.2. Software .................................................................................................. 17
  6.4. Access control and authentication ...................................................................... 17
  6.5. Audit and Accountability .................................................................................... 17
  6.6. System and Communications Protection .......................................................... 18
  6.7. Physical and environment protection ................................................................. 18
7.0 FUTURE PLANS ....................................................................................................... 19
8.0 EXPERIMENTS ......................................................................................................... 21
  8.1. Experiment 1. Worm propagation over Windows network .................................. 21
  8.2. Experiment 2. Behavior based detection of malware in a Windows network .......... 21
  8.3. Experiment 3. Automatic detection of functionalities ......................................... 22
LIST OF FIGURES

FIGURE 1. ARCHITECTURE OF THE DETER TESTBED ......................................................... 3
FIGURE 2. ARCHITECTURE OF THE BU TESTBED .............................................................. 5
FIGURE 3. HARDWARE LAYOUT DIAGRAM FOR BU TESTBED ........................................... 6
FIGURE 4. HARDWARE LAYOUT FOR TESTBED V2 (CONTROL RACK) .............................. 8
FIGURE 5. HARDWARE LAYOUT FOR TESTBED V2 (SIMULATION RACKS 1 AND 2) .............. 8
FIGURE 6. EXPERIMENT LIFETIME IN TESTBED V2 ....................................................... 12
FIGURE 7. SIMPLIFIED VIEW OF THE TESTBED V2 HARDWARE LAYOUT ......................... 13
FIGURE 8. TESTBED DETAILED HARDWARE LAYOUT ...................................................... 14
FIGURE 9. TESTBED SERVICE DOMAINS ........................................................................... 15
FIGURE 10. TESTBED VISUALIZATION STRUCTURE ......................................................... 17
1.0 SUMMARY

The Network Testbed at Binghamton University was designed to facilitate security research in the area of advanced 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. The testbed facilitates the implementation of the virtualization technology in a way that all research software, operating systems and malware run on networked virtual machines. This allows for full access to hardware on the time sharing basis and reliably prevents any interference between different projects. Upon completion of one project/experiment in the queue, all disk data is to be stored to disk storage and data for the next project/experiment will be loaded from disk storage for the next user. This Report documents the most recent modifications made to the software and hardware of the testbed, affecting its architecture, and enhancing its accessibility, security, and visualization.

The hardware upgrades are as follows:

- Additional RAM was installed. Now Testbed has 720GB in total of RAM
- High speed interconnection network was installed. Each server is connected to others via 10Gbps interface
- High speed storage capable of storing snapshot of the whole testbed disk drives was installed
- High volume storage that can store library of experiment data for each researcher was installed
- In order to provide additional layer of separation configurable 1Gbps switches were installed
- In order to enhance visualization a video-wall comprising nine 30” terminals was installed.

The software upgrade includes the most suitable software acquired from open source project EmuLab [http://www.emulab.net/](http://www.emulab.net/). This software allows for remote access to the hardware, storing experiments and isolating experiments from each other.

After the modification, 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.

The Ongoing Work section describes the work being done for the further improvement of Testbed accessibility and applicability. The Future Plans section discusses possible ways to increase Testbed value for the community of cyber security researchers.
2.0 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 high end servers at the data center.

Modern security research at the level of system protection includes risk analysis [2], vulnerability analysis [3], intrusion detection systems (IDS) [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.

2.1. 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 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.

2.2. 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.

![Figure 1. Architecture of the Deter Testbed](image)
First, its design does not allow any direct link to the Internet from the experiment space (see Figure 1). 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.
3.0 BINGHAMTON UNIVERSITY TESTBED ARCHITECTURE

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 (compare Figures 1 and 2). BU Testbed has identical to Deter layout with additional special server for virtualization support (Virtual Center). Each experiment node has a lightweight hypervisor installed. Under this architecture 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.

![Figure 2. Architecture of the BU Testbed](image)

The detailed physical layout of the BU Testbed is presented on Figure 3.
The software part of the Testbed is implemented using the VMware VSphere. VSphere technology provides a great level of configurability and agility. Using VSphere remote researcher could deploy hundreds of virtual machines from templates, connect them into arbitrary network configuration and launch large scale simulation in a matter of minutes. In security research domain capability of hypervisor to “record and replay” proved to be extremely fruitful. It allows running of an experiment with recording of execution path of virtual machines for future analysis.

Over the time the BU Testbed proved to be very useful tool for the malware research [10]. Unfortunately, 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.
- Network-attached storage size was not large enough to store a lot of large scale experiment setups.

This limitations and change in the direction of security research lead us to planning of the upgrade for the Testbed.
4.0 THE UPGRADE FOR TESTBED. BU TESTBED V2.

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 (at current level of technology) that necessitates full access to hardware. Fast development of cloud technologies lead to appearance of new types of malware which exploits cloud environment [11].

In order to follow the move of research interest and provide a firm foundation for large scale experiments we reconsidered the initial design of the BU Testbed in its hardware and software parts.

4.1. 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. 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, high volume storage was added. It consists of two arrays. Each of these arrays is connected with a 1Gbit/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.

Overall upgrade numbers are summarized in table 1.

Table 1. Key Hardware Metrics Before and After the Upgrade

<table>
<thead>
<tr>
<th>Item</th>
<th>Was</th>
<th>Now</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor (Xeon 5300 or more)</td>
<td>46</td>
<td>50</td>
</tr>
<tr>
<td>RAM</td>
<td>216 GB</td>
<td>720 GB</td>
</tr>
<tr>
<td>Network Speed</td>
<td>1 Gb/s</td>
<td>10 Gb/s</td>
</tr>
<tr>
<td>Storage Speed per Node</td>
<td>Less than 3 MB/s</td>
<td>More than 20 MB/s</td>
</tr>
<tr>
<td>Storage Volume</td>
<td>6TB</td>
<td>60TB</td>
</tr>
</tbody>
</table>

Hardware was installed into three racks. One may find detailed rack layout in figures 4 and 5.
Figure 4. Hardware Layout for Testbed V2 (Control Rack)

Figure 5. Hardware Layout for Testbed V2 (Simulation Racks 1 and 2)
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.

4.2. Software

There are 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 implementation section.
5.0 IMPLEMENTATION

5.1. Requirements

5.1.1. 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).

5.1.2. 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 [12] 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

5.2. 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.
5.2.1. 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. Network is configured according to user specification
5. Launching the experiment: all participating nodes are brought up and running.
6. Gathering experimental data. The software utilized by the researcher can gather and store high volumes of data onto connections provided by the Testbed storage.
7. Suspending the experiment:
   a) All participating machines are shut down.
   b) Network is reconfigured
   c) System scrubbing software is booted onto each experiment node.
   d) Disk images from experiment nodes are stored in the central storage by the scrubbing software.
   e) Network connectivity information is stored onto central storage.
8. 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. Visual representation of an experiment lifetime described above is featured in Figure 6.
5.2.2. Testbed hardware design

The overall hardware design is featured in Figure 7. 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 pre-configure 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 Experiment Space Safety Border in Figure 7). In addition, during the experiment the storage accessibility is also separated into two domains: the inside of experiment and outside of experiment. 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.
More detailed hardware design is featured in Figure 8. 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.
5.2.3. Testbed services layout

The functionality and security requirements of 5.1.1 and 5.1.2 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 the necessary minimum: 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 [13] chosen as a main virtualization solution for several reasons:

- Xen provides a 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 are increased by failover and snapshot functionalities.

The distribution of backend systems to domains is shown in Figure 9. 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
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.

Figure 9. Testbed Service Domains
6.0 ONGOING WORK

Currently large parts of the software environment of the Testbed are being expanded and developed. Most of the work is done to address NIST recommended security controls described before.

6.1. Disk imaging

Clonezilla software [14] was taken as a starting point for deployment of disk imaging solution in Testbed V2. Initially Clonezilla was designed for massive classroom deployment in the National Center for High Performance Computing in Taiwan. Clonezilla provides server and client side software for fast parallel disk imaging. It also has additional options required or beneficial in security enforced environment:

- Complete wipe of the hard disk before writing down a new image
- Computing and verification of image checksums
- Optional image compression
- Support for different destinations and proto

6.2. Network reconfiguration

Network configuration system is based on Emulab network configuration system. Testbed V2 has equivalent but different hardware than supported by Emulab. Most of work is done as one to one translation of commands and adaptation of Emulab network subsystem for our needs.

6.3. Visualization

Visualization is one of the most efficient ways to convey information to the audience. Testbed visual subsystem will become a testing platform for development and testing of visualization-based security solutions. This will enable Binghamton University researchers to develop novel approaches to represent security relevant information and generate alarm messages carrying maximum information for attack detection/location/mitigation.

6.3.1. Hardware

Testbed V2 will feature 90 inch display composed of nine 30 inch high resolution displays. Display will be driven by three graphics workstations connected to Testbed by 10 Gbps link.
6.3.2. Software

Botnet simulator developed in Binghamton University incorporates a visual representation subsystem. It will help better understanding of dynamics of the most complex malicious structures existing today.

6.4. Access control and authentification

Testbed V1 relied on Windows Active Directory technology to implement authentication and access control. VSphere system was tightly integrated with Active Directory to enforce proper access model to the Testbed.

Since Testbed V2 infrastructure is build around Linux/Unix services (see Figure 9). We had to abandon Active Directory based solution. In Linux/Unix environment LDAP/Kerberos tandem proved to be the most successful. Currently Testbed V2 supports only exclusive access mode. It allows us to keep things simple. Unfortunately exclusive access mode prevents effective utilization of the Testbed. With planned computing power increase it will become a major problem. To avoid these problems in future we design and test access/authorization system that allows secure sharing of the Testbed among several researchers.

6.5. Audit and Accountability

Testbed V1 employed extensive audit and accountability features of Active Directory and VSphere technology. In spite of reach set of logged and monitored events Active Directory was not natively designed for Testbed environment. There was large set of functions not covered by audits at all. There was no audit for network configuration operations, file operations, anomalous network events.

Problem of uneven audit coverage is being addressed in Testbed V2 at several levels. Each type of operation is being accounted in place (network configuration, file operations, network ac-
cess, user logins, etc.) as well as reported to central event system. Audit is performed in the same two level fashion. Substantial part of the code here is adapted from APIMon IDS developed in Binghamton University [15]. Testbed V2 employs it as a security information and event management system (SIEM system).

Hardware operations performed on the Testbed are logged in changes/modifications logbook. We plan to install CCTV camera for monitoring of Testbed physical access.

**6.6. System and Communications Protection**

All Testbed communications are divided into four groups:
- Dangerous communications group is directly involved in experiments and may contain malicious traffic.
- Control communications group supports delivery of control signals to Testbed hardware.
- Internal communications group supports interaction among different Testbed services.
- External communications group supports remote logins for researchers.

Dangerous group is kept separate from others at the level of hardware. When an experiment is active there is no direct or indirect link between Dangerous communication group and other groups. When an experiment is not active Dangerous communication links are employed to scrap data and deploy next experiment. At this time tightly controlled link is established between the Dangerous group and the Internal group.

External group is connected to the Internet. Therefore it requires almost as much attention as Dangerous group. We employ firewall and VPN technology to limit number of ways Testbed interfaces with Internet.

Control and Internal groups are monitored but not heavily restricted. This is done to simplify design and therefore reduce configuration workload.

**6.7. Physical and environment protection**

Testbed V2 is installed in the room with limited access controlled by card swipe device. In addition access to the Testbed is limited by internal glass enclosure with mechanical lock. It prevents authorized to the room personnel (AC service, cleaning service, etc.) from accessing Testbed racks. Racks are kept closed all the time.
7.0 FUTURE PLANS

We can identify several important directions where Testbed at Binghamton University is likely to move in near future:

a) Successful attacks of the Stuxnet worm have shown vulnerability of major SCADA systems to malicious actions [16]. Most of hardware employed in digital control systems was not designed with security in mind. Replacement or modification of most of the installed hardware base is not achievable now. Some other work around solution should be found in near future to protect systems that can directly affect our life.

Although there various SCADA systems are quite common, it is really hard to get access to them for research purposes. Even operation logs are considered sensitive and are not released for public for research analysis purposes.

This situation calls for establishing pure experimental SCADA system within cyber security Lab.

![Diagram of SCADA system](image)

Figure 11. Preliminary Design for Experimental System for SCADA Security Research

The simplest experimental SCADA setup should contain major elements existing in any SCADA network: controlled plant, real time controller, sensors, actuators, communication bus and SCADA server. Bus-wiretapping devices have to be added to provide data for IDS system (IDS sensors).

b) Substantial amount of research was done in the area of GPU computing. Researchers employ high performance GPU units for security related computations. Proof of concept for GPU assisted malware was introduced in [17]. Our future work in graph matching algorithms might require this kind of computing power as well. Therefore we envision addition of high performance computing modules to Testbed as valuable improvement. Another hot topic is trusted com-
puting and trusted hypervisors. Intel-TXT technology enabled hardware can provide a good starting point for this direction of research.

b) Expansion of installed software base. Essentially software makes Testbed powerful and versatile tool for studying malware.

c) Introduction of high speed network in Testbed V2 opened new areas for expansion of research. High speed (more than 10GBps) traffic monitoring is a separate area for research. High link speed prevents effective monitoring by conventional PC based solutions. Specialized mixed hardware-software processing platforms might be required for effective detection and suppression of high speed malicious traffic.

d) Computing power increase. Modern operating systems put much higher minimum requirements on hardware than it was when Testbed V1 was designed. Simulation of the same number of hosts with Windows 7 requires ten times more RAM and disk space than with Windows XP.
8.0 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 based on the system call domain manifestation of their propagation engines. 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.

8.1. 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 [18].

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, equipped with some reporting functions instead of the 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.

8.2. 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 combination of computer operations defined not by their specific implementations, but by the outcome of the entire combination. 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 [19].

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 12.

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 [19].

8.3. 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 [20]. 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 [21]. 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 from the Windows hosts. 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 to the limit 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 [21] 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.
9.0 CONCLUSION

The BU Testbed enables researchers to access to a unique set of the hardware and software that in combination result in simulation 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.
10.0 LITERATURE


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


### 11.0 LIST OF SYMBOLS, ABBREVIATIONS AND ACRONYMS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>BU</td>
<td>Binghamton University</td>
</tr>
<tr>
<td>CPN</td>
<td>Colored Petri Net</td>
</tr>
<tr>
<td>DHCP</td>
<td>Dynamic Host Configuration Protocol</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System Protocol</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphical Processing Unit</td>
</tr>
<tr>
<td>IDS</td>
<td>Intrusion Detection System</td>
</tr>
<tr>
<td>IPMI</td>
<td>Intelligent Platform Management Interface</td>
</tr>
<tr>
<td>Kerberos</td>
<td>Computer network authentication protocol</td>
</tr>
<tr>
<td>KVM</td>
<td>hardware device that allows a user to control multiple computers from a single keyboard, video monitor and mouse</td>
</tr>
<tr>
<td>LDAP</td>
<td>Lightweight Directory Access Protocol</td>
</tr>
<tr>
<td>NIST</td>
<td>National Institute of Standards and Technology</td>
</tr>
<tr>
<td>NTP</td>
<td>Network Time Protocol</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>SCADA</td>
<td>Supervisory Control and Data Acquisition</td>
</tr>
<tr>
<td>SSH</td>
<td>Secure Shell</td>
</tr>
<tr>
<td>TXT</td>
<td>Trusted Execution Technology</td>
</tr>
<tr>
<td>VLAN</td>
<td>Virtual Local Area Network (LAN)</td>
</tr>
</tbody>
</table>