NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

THESIS

SCALABILITY ASSESSMENTS FOR THE MALICIOUS ACTIVITY SIMULATION TOOL (MAST)

by

Ray Longoria Jr.

September 2012

Thesis Co-Advisors:	Gurminder Singh John H. Gibson

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2012	**3. REPORT TYPE AND DATES COVERED** Master's Thesis
4. TITLE AND SUBTITLE Scalability Assessments for the Malicious Activity Simulation Tool (MAST)		**5. FUNDING NUMBERS**
6. AUTHOR(S) Ray Longoria Jr.		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number: N/A.		
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

MAST – Malicious Activity Simulation Tool – aims to support the conduct of network administrator security training on the very network that the administrator is supposed to manage. A key element of MAST is to use malware mimics to simulate malware behavior. Malware mimics look and behave like real malware except for the damage that real malware causes. MAST enhances training by providing realistic scenarios that are dynamic, repeatable, and provide relevant feedback.

This thesis is meant to test the scalability characteristics of MAST. Specifically, we show that an exponential increase in clients using the MAST software does not impact network and system resources significantly. Additionally, we demonstrate and discuss how MAST is installed on a new network, and delivers feedback to the organization being trained.

14. SUBJECT TERMS Red Teams, Malware, Network Security, Training, Computer Network Defense, Simulation, Scalability			**15. NUMBER OF PAGES** 85
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

SCALABILITY ASSESSMENTS FOR THE MALICIOUS ACTIVITY SIMULATION TOOL (MAST)

Ray Longoria Jr.
Captain, United States Marine Corps
B.A., The Citadel, Military College of South Carolina, 2006

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2012

Author: Ray Longoria Jr.

Approved by: Gurminder Singh
Thesis Co-Advisor

John H. Gibson
Thesis Co-Advisor

Peter J. Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

MAST – Malicious Activity Simulation Tool – aims to support the conduct of network administrator security training on the very network that the administrator is supposed to manage. A key element of MAST is to use malware mimics to simulate malware behavior. Malware mimics look and behave like real malware except for the damage that real malware causes. MAST enhances training by providing realistic scenarios that are dynamic, repeatable, and provide relevant feedback.

This thesis is meant to test the scalability characteristics of MAST. Specifically, we show that an exponential increase in clients using the MAST software does not impact network and system resources significantly. Additionally, we demonstrate and discuss how MAST is installed on a new network, and delivers feedback to the organization being trained.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AdWare	Advertising Software
CND	Computer Network Defense
CNSS	Committee on National Security Systems
COMPOSE	Common PC Operating System Environment
COTS	Commercial Off The Shelf
CPU	Central Processing Unit
CTC	Communication Training Center
DC	Domain Controller
DCM	Device Control Module
DDoS	Distributed Denial of Service
DHCP	Dynamic Host Configuration Protocol
DHS	Department of Homeland Security
DISA	Defense Information Systems Agenct
DNS	Domain Name System
DoD	Department of Defense
ePO	ePolicy Orchestrator
ExComm	Executive Committee
FOC	Full Operating Capability
Gb	Gigabit
GB	Gigabyte
GHz	Gigahertz
GUI	Graphical User Interface
HBSS	Host Based Security System

HIPS	Host Intrusion Prevention System
IA	Information Assurance
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IIS	Internet Information Server
IIT	Infantry Immersion Trainer
IOC	Initial Operational Capability
IP	Internet Protocol
IPS	Intrusion Prevention System
ISNS	Integrated Shipboard Network System
Malware	Malicious Software
MAST	Malicious Activity Simulation Tool
MEF	Marine Expeditionary Force
NNTP	Network News Transfer Protocol
NSA	National Security Agency
OPFOR	Opposing Forces
OTA	Over The Air
RaD-X	Rapid Experience Builder
RAM	Random Access Memory
SMTP	Simple Mail Transfer Protocol
TB	Terabyte
TCP	Transmission Control Protocol
TTP	Tactics, Techniques and Procedures
UPS	Uninterruptable Power Supply
USCYBERCOM	United States Cyber Command

USMC	United States Marine Corps
VM	Virtual Machine

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First and foremost, I give thanks to my Heavenly Father, His Grace, and all the blessings I am so lucky to have.

This thesis would not have been possible without the guidance, patience, and support of my thesis advisor, Professor Gurminder Singh. Thank you for your time and the opportunity to work with you. My thesis co-advisor, Mr. John Gibson, provided guidance and mentorship not only for this thesis, but throughout my NPS career as well. Thank you.

I would be remiss if I did not offer my heartfelt thanks and appreciation to CDR Jim Hammond and LT Justin Neff. I cannot say enough about Jim's effort, hard work, and leadership while working on the MAST project. I thank Justin for his dedication and motivation to not only the MAST project, but to all our endeavors together. Also, I would like to thank Mr. Arijit Das, Mr. Erik Lowney, and Mr. Greg Belli. Their hard work and contributions advanced and elevated this project to levels we hadn't considered possible within our timeline. Thank you gentlemen. Additionally, I want to thank Ms. Susan Hood from SPAWARSYSCEN-PACIFIC and Mr. Quincy Taitt from Man Tech Systems for providing the software and training that allowed us to test MAST in our shipboard simulated environment.

I would also like to acknowledge and thank all my professors and fellow cohort members over the last two years. Specifically, I would like to acknowledge, CDR Al

Shaffer, J.D. Fulp, Scott Cote, Professor Rob Beverly, LT Joey Carter, and of course all my fellow Marines. Semper Fi brothers.

Finally, to my amazing wife Tara, I thank you from the bottom of my heart. I could not have done this without your love and support. I am forever grateful and thankful that God brought us together. I love you. To my beautiful children, Emeline, Everett, Elianna, and the twins, you all are my drive and motivation. Thank you for your love, support, sacrifices, and occasional drama. I wouldn’t have it any other way.

I. INTRODUCTION

During the summer of 2009, then Secretary of Defense Robert Gates directed the establishment of United States Cyber Command (USCYBERCOM). The new command achieved Initial Operational Capability (IOC) the following summer, followed by Full Operating Capability (FOC) on October 31, 2010. USCYBEROM is:

Responsible for planning, coordinating, integrating, synchronizing, and directing activities to operate and defend the Department of Defense information networks and when directed, conducts full-spectrum military cyberspace operations (in accordance with all applicable laws and regulations) in order to ensure U.S. and allied freedom of action in cyberspace, while denying the same to our adversaries. [1]

A key directive in USCYBERCOM’s mission statement is to defend the DoD information network. While there are many methods and techniques used to execute this task, the underlying foundation for each of those methods is training. Training occurs at all levels and stages. It must be relevant, continuous, and above all effective.

A. NETWORK SECURITY AND INFORMATION ASSURANCE TRAINING

As the use of computing devices, Internet connectivity, and cloud-based services rises, the need for more personnel trained to install, maintain, and protect these services also rises. These developments are not isolated to business, government, or private communities. These same technological developments are also in demand

and in use by the U.S. military. However, a key difference between military use and all other is the critical need to protect those services and the network they propagate over due to military’s national defense mission.

Training for U.S. service members and DoD personnel varies based on location, experience, level of expertise required, and mission. Options for training range from classroom-type training, computer-based training, and red team training. Classroom training offers a lot of “hands-on” experience in a controlled setting, while red teams provide a more realistic experience, as their training is conducted on the actual network the administrators maintain.

B. SHORTFALLS WITH CURRENT TRAINING METHODS

While our current training methods are effective, there are a few key shortfalls we wish to address with this thesis. Red teams, for example, are finite resources that are in very high demand. As more commanders understand the threat in the cyber domain, they want to ensure their unit’s preparedness by providing relevant and effective training. While red teams are capable of providing this, the reality is there are not enough of them. Additionally, the training offered through the use of red teams is dynamic in nature, which in turn can lead to inconsistent training results and feedback for the unit or organization being trained or evaluated.

Classroom or laboratory training can also be effective and relevant. However, a potential shortfall is the operating environment in which a trainee will train. The computer systems and network to which they are connected are often not be the same type of systems and configuration they would use on their operational network.

C. MALICIOUS ACTIVITY SIMULATION TOOL (MAST)

To address the shortfalls mentioned above we developed a software-based tool that provides relevant and dynamic training on the very network that the trainee will manage. The Malicious Activity Simulation Tool (MAST) was designed around a command and control architecture that allows training to be executed from a remote location with minimal impact on system and network resources.

MAST uses scenarios, which are made up of multiple modules and commands, to conduct the training. The modules are benign programmed behaviors that mimic the signature of real malware but do not include the “infectious” behavior that causes harm to the network and computing resources. All events and actions are captured and formatted into a report that provides the training unit and their leadership a view into their unit’s cyber security posture and preparedness.

The current MAST architecture leverages the research, development, and framework of CDR Will Taff, LCDR Paul Salevski, and LT Justin Neff [1] [2]. Their efforts have led to the development of a prototype that is used as the foundation for experimentation in this thesis.

D. OBJECTIVES

In this thesis, we detail the properties of MAST with respect to scalability. The intent of the tool is to provide training in an environment consisting of hundreds of clients. In order to continue our development, it is important we understand how MAST uses system and network resources while conducting training. MAST must be able to train hundreds of clients while utilizing minimal resources.

E. ORGANIZATION

Chapter I provides a brief description of current shortfalls in network security and IA training. Additionally, a general description of MAST and its functionality is detailed along with the objectives of this thesis.

Chapter II outlines previous research, current training methods and the work of Taff, Salevski, and Neff. Additionally, we provide a detail description of red teams and some historical examples of their use. We conclude the chapter with a discussion of varying types of malicious software (Malware).

Chapter III discusses our design considerations with respect to MAST and the test platform. Specifically, we detail MAST’s functionality and architecture, and provide an example of a training scenario. We provide details of the test platform’s hardware and software features along with a detailed discussion of training and the aspects involved in conducting training. We conclude the chapter with an overview of the Host-Based Security System (HBSS) software suite.

Chapter IV provides a detailed description of the assessments required to determine MAST’s scalability characteristics. We discuss the installation of the software from a remote location on a network that does not have MAST. Additionally, we show how MAST uses system and network resources when executing a training scenario. We conclude the chapter with a discussion of MAST’s feedback and reporting capabilities.

Chapter V provides conclusions and recommendations as a result of this experiment. We give our assessment of MAST’s implementation of a large network and the utilization of resources by the tool. We conclude the chapter with a discussion of future work to be conducted to prepare MAST for implementation in an operational environment.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

This chapter details some of the varying cyber security and Information Assurance (IA) training methods utilized by the United States (U.S.) uniformed service members and Department of Defense (DoD) personnel. Specifically, we provide some insight into red teams, who they are, and how they operate, and other sources of training within the DoD. Additionally, we discuss some malicious threat signatures and behaviors, and the proof of concept for our system, called Malicious Activity Simulation Tool (MAST).

A. TRAINING METHODS FOR DOD NETWORK ADMINISTRATORS

1. Red Teams

In a 2008 interview, Popular Mechanics was given unprecedented access to a National Security Agency (NSA) red team member. The interviewee revealed that the main task of the red teams was to provide "adversarial network services" to all units and personal within the DoD while ensuring strict adherence to their first rule of operation: "do no harm [4]". Within this context, and in general, a red team is made up of highly skilled and experienced personnel whose mission is to "anticipate and simulate the decision-making and behaviors of potential adversaries [5]". Red teams allow units to "train as [they] fight" by conducting their actions in the actual operational environment, while utilizing the same tactics, techniques, and procedures (TTPs) of a real enemy.

According to the Committee on National Security Systems (CNSS), a red team is defined as:

> A group of people authorized and organized to emulate a potential adversary's attack or exploitation capabilities against an enterprise's security posture. The Red Team's objective is to improve enterprise Information Assurance by demonstrating the impacts of successful attacks and by demonstrating what works for the defenders (i.e., the Blue Team) in an operational environmental. [6]

The use of red teams is not limited to the computer security or computer network domain. Red teams, who are sometimes referred to as an Opposing Force (OPFOR), are utilized for training, planning, and evaluating at the strategic level down to the tactical level.

a. Contemporary Example of a Red Team Implementation

One way in which U.S. Marine Corps infantry units prepare for operations in a hostile urban environment is to send their members through the Infantry Immersion Trainer (IIT) facility located on Marine Corps Base Camp Pendleton, California. IIT is a physical training environment that incorporates computer simulation technology to provide "a vivid and realistic virtual environment to prepare warfighters for a range of possible scenarios" [7]. The scenarios and simulations incorporated into the training program, known as TTPs, are integrated by a red-team-like entity.

b. Historical Example of a Red Team Implementation

At the height of the Cuban Missile Crisis, President Kennedy established the Executive Committee (Ex Comm) of the National Security Council to provide him guidance and response options that were based on a careful analysis of the enemy and their potential courses of action. Specifically, these red-team-like members were non-military members who helped provide information and courses of action that countered the recommendations of the highly influential military members on the committee [8].

c. Red Team Implementation within a Cyber Domain

As stated earlier, the number one rule for NSA red team members is to “do no harm”. This approach to training parallels the methodology taught in the E-Commerce (EC) Council’s Ethical Hacking and Countermeasures course. According to the Certified Ethical Hacking courseware manual, an ethical hacker is an individual “hired by organizations to attack their information systems and networks in order to discover vulnerabilities and verify that security measures are functioning correctly [9]”. The ethical hacker, or red team member, can then provide the organization or unit the status of their security posture, identify any weaknesses, and most importantly, identify remedial actions that can be taken to enhance security.

The duties of the red team are limited to the time and resources available and the experience of the individuals on the team. Like any other team or group of individuals working together towards a common goal, there are varying levels of competency and experience among the individual members of the team. The amount of red-teaming or depth of penetration a team can make on a respective network is unpredictable and not standardized due to variables associated with the particular red team, the network being probed, and the personnel administering that network. Additionally, feedback to the respective unit being tested or trained is critical to its security enhancements, operational security posture, and most importantly mission accomplishment but it is often inconsistent and neglected.

2. Defense Information Systems Agency (DISA) Training Programs

Another resource for cyber security and IA training for network administrators is the training products offered by the Defense Information Systems Agency (DISA). DISA offers a variety of computer-based and web-based training programs; instructor led training programs; and virtual training environments. One course in particular, the Rapid Experience Builder (RaD-X) course, is designed to expose students to malicious software (malware) and provide hands-on training with firewall log reviews, intrusion detection system (IDS) analysis and configuration, and anomaly detection using computer network defense (CND) tools [10]. Trainees in this course are able to observe and interact with a variety of real malware in a laboratory setting. The laboratory environment is air-gapped, or isolated from all other networks and the Internet. While there are many positive aspects to this hands-on, instructor-led training, there are a few shortfalls. First, the cost of transporting the laboratory for training or sending personnel to be trained can be very high. Second, there is a very high maintenance cost associated with managing and maintaining the systems. After each class, each system within the RaD-X environment must be “wiped,” that is, electronically cleared, and re-imaged to prepare for the next session. Finally, for the trainee, there is no guarantee that the RaD-X computer systems and network topology mirror the operational network with which they are familiar.

3. USMC Communication Training Centers (CTCs)

Within the Marine Corps there exist three Communication Training Centers (CTCs), located respectively within each Marine Expeditionary Force (MEF). The classes available through one of these CTCs range from tactical radios to Cisco routing protocols and concepts. The depth of instruction on cyber security and IA is limited due to the limited resources available at each location and the additional military commitments for all service members. Like the RaD-X architecture mentioned above, the configuration and system design used in training often does not mirror what the actual service member will administer during an exercise or while deployed.

All the training methods mentioned above are undoubtedly beneficial and critical to the continued security of our computer network infrastructure. We propose that the incorporation of MAST will enhance network administrator training by allowing units to train on their very own operational network in a safe and controlled environment. MAST will provide consistent training and, most importantly, provide consistent feedback to the users.

B. MALWARE

Malicious software, or malware, is a general term used to describe software that is specifically designed to cause a computer system, its network, or peripherals to perform actions not intended by the user, or deny the user a resource resident within the computer or network. In a 2005 case study describing attacks against critical infrastructure, the U.S. Department of Homeland Security (DHS) defined malware as:

> Programming (code, scripts, active content, and other software) designed to disrupt or deny operation, gather information that leads to loss of privacy or exploitation, gain unauthorized access to system resources, and other abusive behavior. Examples include various forms of adware, dialers, hijackware, slag code (logic bombs), spyware, Trojan horses, viruses, web bugs, and worms. [11]

The impact of malware on a computer system can range from harmless and annoying to severely devastating and damaging. Advertising software (adware) or spam e-mails, while inconvenient, will have little to no impact on the system's resources and services. A Trojan horse, conversely, could give a hacker complete access to a system at the administrator level, thereby compromising the confidentiality, integrity, or accessibility of files and resources located within the system.

For the scope of this thesis, and MAST in general, the term "malware" will refer to those exploits and their behaviors that can cause catastrophic damages or deny the end user the ability to accomplish the mission. Specific types of malware behavior MAST will simulate include, but may not be limited to, worms, botnets, and viruses.

1. Worms

According to the Froehlich/Kent Encyclopedia of Telecommunications, a worm is defined as “self-replicating programs that spread with no human intervention after they are started” [9]. Gu et al. identify three characteristics common to most Internet worms [12]:

- The first characteristic deals with the volume and type of traffic generated by an Internet worm. A worm is more susceptible to identification based on its patterns and signatures. Since worms are self-replicating, they do not evolve or change as they propagate. A worm’s uniform characteristics make it easier to detect with network traffic analysis software, such as Wireshark and TCPDump.
- A second characteristic deals with the worm’s scanning behavior. Most Internet worms will use a pseudo-random search algorithm to discover open ports on a vulnerable system. A worm with this behavior will attempt to connect to numerous closed ports, which will result in the same number of failed connections. A brief analysis of these failed connections could reveal the presence of a worm.
- The final characteristic is a noticeable increase in system resource utilization. The host uses a lot of resources responding to the initial scanning done by a worm, followed by a further depletion of resources to find more vulnerable systems.

The scanning and propagation features of an Internet worm are normally only part of its behavior. Most malware carry or deliver some sort of malicious payload that can be used to capture sensitive information, report back to a base station, or in the worst case, corrupt or delete essential system files.

Cornell University student, Robert Morris, released the first known instance of an Internet worm in 1988. The Morris worm, which was initially designed to measure the size of the Internet-ancestor, ARPANET, had a self-replicating and self-propagating feature that caused 10% of all computers connected to the ARPANET to become ineffective due to the allocation of resources dedicated to the Morris worm [13].

2. Viruses

Like Internet worms, viruses are also self-replicating software that can carry a malicious payload. The distinguishing characteristic between worms and viruses is that viruses require some sort of action on the part of the end-user to initiate its behavior. Viruses propagate through e-mails or malicious attachments, not through system vulnerabilities as a worm does. Peter Szor, author of *The Art of Virus Research and Defense*, defines a computer virus as:

> Code that recursively replicates a possibly evolved copy of itself. Viruses infect a host file or system area, or they simply modify a reference to such objects to take control and then multiply again to form new generations. [14]

Viruses, like worms, have distinct characteristics and signatures that can be detected with an Intrusion Detection System (IDS). Unfortunately, these combative methods tend to be reactive in nature due to the virus' stealth nature and various infection methods. Viruses can be programmed to attach themselves to other executable files, self-modify, and replicate. The signature database associated with the IDS must be updated constantly and reviewed to ensure maximum protection.

3. Botnets

Another form of malware that has become more widely used, due to the increase in computing systems connected to the Internet, is a “botnet”. A “bot” is a computer system that has been compromised with malicious software and the “net” is the network on which the infected host communicates. While there are many common characteristics among viruses, worms, and botnets, the distinguishing factor for botnets is its command and control architecture. In this command and control architecture there is normally one bot that acts as the master while the other bots execute the commands given by the master.

As stated earlier, the rise in computer usage and Internet connectivity has led to the increase in botnet attacks. The most common attack associated with botnets is the Distributed Denial of Service (DDoS) attack. A DDoS attack is designed to overwhelm the resources of a single entity by sending it more requests than it can handle. These request normally come from multiple machines at the same time, which are all a part of a botnet. However, botnets can be used for more than just a DDoS attack. According to Ellen Messmer, who published an article on the growth of botnet usage:

> It’s not just DDoS attacks that are associated with bots. Botnets are usually specialized, designed for criminal tasks that range from spam distribution; stealing identity credentials such as passwords, bank account data or credit cards and key-logging; click fraud; and warez (stealing intellectual property or obtaining pirated software). [15]

Like viruses, bots that are a part of a botnet, can be difficult to detect. Most bots are programmed to lay dormant until activated by the master bot. When they are activated, the bots exhibit scanning behaviors similar to a worm. The worm-like behavior makes the bot detectable with traffic analysis tools, such as Snort or Wireshark.

C. PROOF OF CONCEPT FOR A MALICIOUS ACTIVITY SIMULATION TOOL

The foundation for this thesis lies in the research started by Commander William Taff, Lieutenant Commander Paul Salevski, and Lieutenant Justin Neff. Taff and Salevski, whose thesis “Malware Mimics for Network Security Assessment” described a “red team” approach to network training, some of the shortfalls in network security training for U.S. military personnel, and a proposed software solution that addresses some of these shortfalls by utilizing techniques associated with red teams. More specifically, the tool they proposed and developed has the following characteristics [2]:

- The tool’s architectural design is based on a command-and-control or server-client model. The operator of the master server is the trainer, while the end-users are the trainees. Additionally, this design allows for the trainer to be located either locally or remotely.

- The tool is designed to allow users to "train as you fight" by executing the training on the users' operational network. All actions and behaviors are benign in nature, thereby causing no threats to the network or end-hosts. Also, the network traffic generated by the system does not overwhelm network resources and impact users not involved in the training.
- Finally, the tool is designed to capture all commands and actions so that a report could be generated to profile the training. This is an important characteristic that is fundamental to any training scenario.

Neff furthered Taff and Salevski's research by verifying and validating their proposed approach to network security training. Specifically, Neff defined various metrics that were used to compare MAST training approach to other methods of training currently available. His research asserted that the MAST system is a viable approach and can improve network security and the IA posture of a unit when augmented to the other resources currently available [3].

The theses authored by Taff, Salevski, and Neff are the proof-of-concept and foundation upon which MAST has been built. It is their work that we intend to expand and further develop.

D. SUMMARY

In this chapter, we discussed varying methods used to train computer network administrators. Specifically, we detailed who and what red teams are, and examples of their implementation, along with other forms of DoD-sourced training. We also discussed the malware domain and some of the categories of malware that fall within that domain.

Finally, we discussed the research and development of a software-based approach to training network administrators. In the following chapter we will expand on this software-based approach by detailing how this approach can augment current training methods. Additionally, we will provide an overview of MAST and describe the implementation platform for experimentation.

III. DESIGN CONSIDERATIONS AND TEST PLATFORM

In this chapter, we detail our assumptions about the training objectives and training environment for which the Malicious Activity Simulation Tool (MAST) is to be implemented. Along with these assumptions, we provide a detailed overview of MAST's functionality, architecture, benefits over current training methods, and an example training scenario MAST could implement. We conclude the chapter with a discussion on the Host-Based Security System (HBSS) and the virtual shipboard network we are using for testing and evaluating.

A. TRAINING

1. Training Objectives and Environment

As stated in the previous chapters, the foundation for this thesis lies in the previous work, research, and development by Taft, Salevski, and Neff [2] [3]. An important topic they helped define and scope for this project is the training paradigm. Specifically, they defined a training objective as "the skill or behavior that we wish to reinforce" [2]. This definition is a foundational principle of the MAST design. Since training objectives vary by unit, size, location, experience, and numerous other factors, MAST is designed to be modular in nature. MAST can be "customized" to fit varying training objectives.

The implementation of MAST assumes a training environment where there is a trainer, trainee, safety observer, and computer network that is inter-connected and accessible by all these individuals. The person(s) or organization responsible for developing training objectives and overseeing the training is the trainer. The individual or organization receiving the training and trying to meet the objectives is the trainee. The person or organization responsible for the safety of the training and the adherence to any constraints or restraints is the safety observer. Finally, the platform upon which the training is conducted is an inter-connected network of computers on an approved DoD computer network. The computer systems attached to this network have a baseline computer image approved by its respective service or agency, and includes the installation of HBSS.

2. Shortfalls of Current Training Methods

As stated in the previous chapter, there are varying training methods available to network administrators for network security and IA. We believe there are four major shortfalls with these methods that the MAST addresses:

a. Finite Resources

Taft and Salevski stated that the use of red teams for training is “the pinnacle of a unit’s training [13].” But unfortunately, red teams are a finite resource that are over-taxed and in high demand. If a unit is lucky, they may have an opportunity to train with a red team just prior to a deployment or commencement of an exercise.

b. Non-standardized Training Methods

As stated in the previous chapter, the attack methods and probing techniques used by red teams vary due to factors such as experience, time available, complexity of the network, discovered vulnerabilities, and many more. These variables make standardized training with respect to red teams virtually impossible.

c. Inconsistent Feedback

The dynamic training approach and non-standardized training methods offered by red teams can lead to inconsistent feedback for the unit being trained. The task of capturing all events and actions is very manpower intensive and time-consuming. Time and manpower are two resources of which the red teams do not have enough. If detailed feedback is desired, then the amount and quality of training provided by the red team will be diminished.

d. Different Training Platform

While laboratory or schoolhouse type training can mitigate some of the issues with standardization and feedback, there are two issues other issues with this type of training:

- First, the cost of sending personnel to be trained or transporting the laboratory to the training location can be very high. Additionally, the costs for managing and maintaining the laboratories can be very expensive.
- Second, there is no guarantee that the system and network settings and configuration will mirror that of the actual network the trainees will use for their exercise or deployment.

In the following sections we will discuss the benefits and details of the MAST and its role in the training domain.

3. Benefits of Implementing MAST

MAST is designed to address the shortfalls mentioned in the previous section by providing a software-based solution that is realistic, repeatable, modular and dynamic. MAST is designed to simulate and automate some of the training methods conducted by red teams. MAST's training methods, which would be available to all DoD personnel, can be repeated an unlimited number of times to ensure the training objectives are met. One of the MAST's key functions is to provide reports on the events surrounding a training scenario. The reports will help a unit identify its strengths and weaknesses, which in turn will allow it to better focus its training resources. Finally, MAST is designed to be used on the same network the trainees use for their day-to-day operations. The command and control design of MAST allows the trainer to scale the training only to those desired hosts and, most importantly, the training can be ceased expeditiously to allow trainees the ability to resume their operational commitments. Finally, MAST is designed to "do no harm" to the network or the hosts attached to the network.

B. MALICIOUS ACTIVITY SIMULATION TOOL (MAST)

During Taff and Salevski's initial research and prototype development of MAST, formerly known as Malware Mimics, it was determined that MAST be implemented according to a client-server paradigm [2]. As shown in Figure 1, the client-server paradigm allows for the trainer to conduct the training from a local or remote location using a command-and-control architecture. Additionally, there are local and remote databases for capturing actions on all events, and a local and remote "kill switch" to cease training at any time.

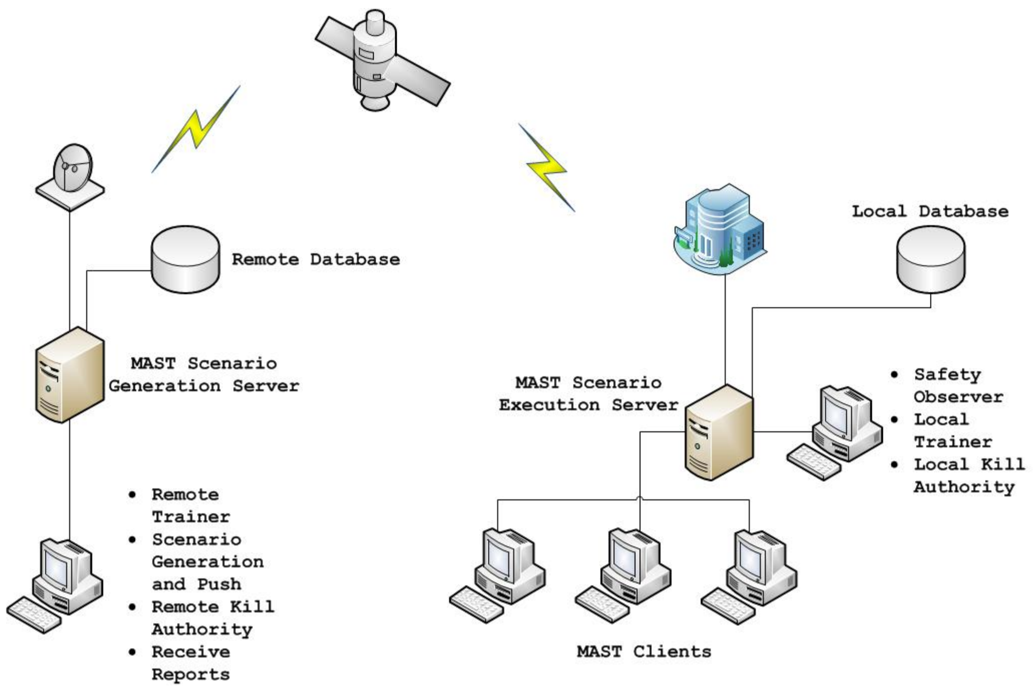


Figure 1. The MAST Architecture Overview

More granular details on the system's functionality, architecture, and safety measures are described below. Additionally, the chapter concludes with an example training scenario utilizing the MAST.

1. System Functionality

In the previous section we described how MAST fills the shortfalls created by the current methods of training. In this section we describe the functionality that exists within MAST to fill these voids.

a. Scenario Generation

Scenario generation is an important function that allows for dynamic and relevant training. As new threats develop, or existing threats remain persistent, it is critical that trainers have the ability to create unique situations that enforce a certain training objective. A scenario is made up of commands, which are executed by the MAST client, and modules, which are pre-programmed behaviors the client will execute. A library of modules will exist at all levels of the MAST and can be combined or used interchangeably to create unique scenarios.

For example, if the signature of a certain piece of malware is to perform a network scan followed by an Internet Control Message Protocol (ICMP) echo-request (“ping”) out of a specific network port to a specific Internet Protocol (IP) address, this action can be recreated into multiple modules for re-use in other scenarios. The scanning behavior is one module while the ping request is another module.

Ideally, the creation of new modules and scenarios is done by the remote trainer whose experience and skills are equivalent to that of an ethical hacker or a member of a red team.

b. Scenario Distribution

The next important system function is scenario distribution. This function is accomplished using a top-down approach. The trainer, from a remote location, pushes new scenarios, modules, or updates from the remote server, known as the Scenario Generation Server (SG Server) to the

MAST-server located locally where the training is to be conducted. The local server, known as the Scenario Execution Server (SE Server), then pushes the updates to the clients as needed.

The distribution of new scenarios or updates can be “pulled” or “pushed” from the respective server. The SG Server can push the updates down to the SE Server, or the SE Server can check-in with the SG Server and determine if any update needs to be pulled. The same process applies to the SE Server and the clients it serves.

c. Scenario Execution

Scenario execution occurs at all levels of the MAST system. A remote trainer can execute a scenario from the SG Server via the SE Server co-located with the training unit. For localized training, a scenario can be executed directly by utilizing only the SE Server. Upon receipt of an execution command, the MAST Client executes the specified scenario.

d. Reporting and Archiving

Following a bottom-up approach, reporting begins when a MAST Client completes a given module or scenario and reports its actions and events to the SE Server. The SE Server, with a limited database capability, archives the information in order to generate reports for the local or remote trainers. The remote trainer, who can leverage the SG Server to manage multiple SE Servers, determines the level of granularity desired from the SE servers. These reports give the trainers and leaders of the unit being trained a snapshot of how the trainees performed, which in turn can be used to create a profile of strengths and weaknesses. This will allow for a better and more efficient use of training resources.

The SE Server and the SG Server have access to a database for data archiving. The database is used to store scenarios, modules, and reports from all clients and servers in the system.

2. System Architecture

The MAST system functions mentioned above are implemented with the use of three main components:

- Scenario Generation Server (SG Server)
- Scenario Execution Server (SE Server)
- MAST Client(s)

All three components are Java-based software programs consisting of multiple classes or files designed to run on a variety of Microsoft Windows-based operating systems. Figure 2 provides a notional implementation view of the system design.

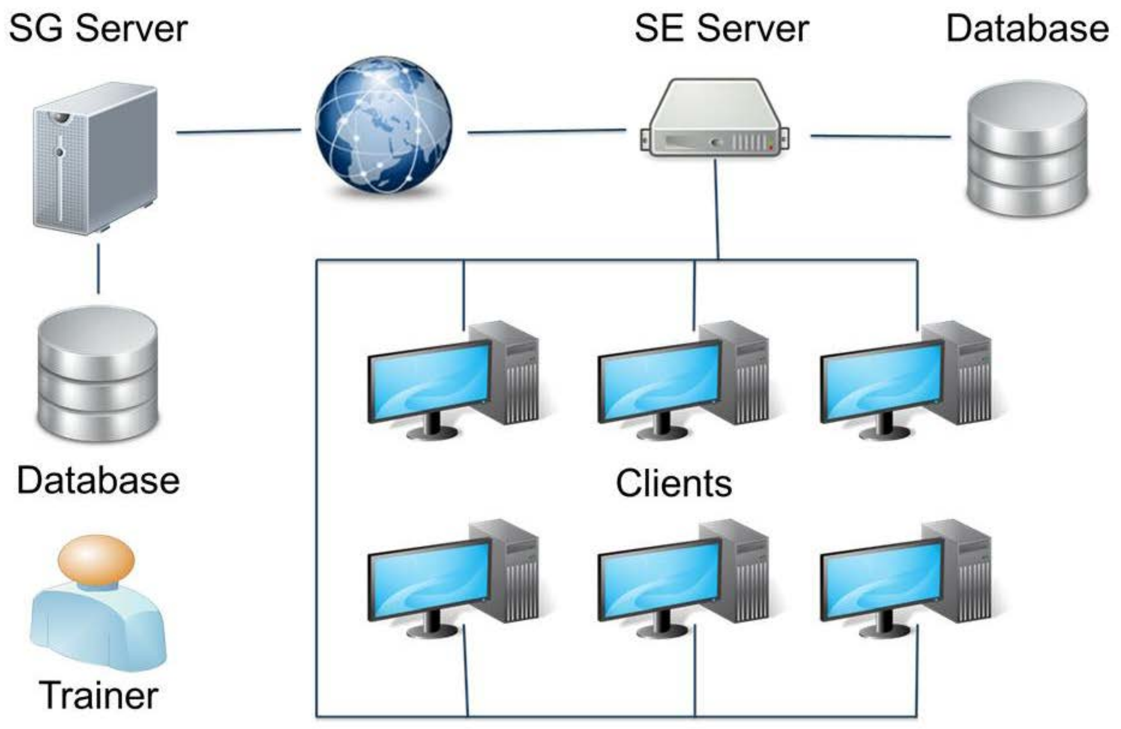


Figure 2. Logical View of MAST Architecture (From Greg Belli and Erik Lowney)

3. Safety Features

Like any military training exercise, safety is always a priority. MAST provides numerous safety features to ensure the integrity of the network and hosts connected to the network.

a. Client Check-in

Prior to the commencement of training, each client or end-host participating in the training must check-in with the SE Server. When the execution of a scenario begins, the SE Server communicates only with those clients on its checked-in list. This ensures non-training users and end-host systems are not affected by the ongoing training and can perform their duties as normal.

b. Kill Switch

The “kill switch” is a simple mechanism or command located at both the SG Server and SE Server. This command, if executed, will cease all training and begin the roll-back module. The “kill switch” ensures immediate and full access to the network and end-hosts in the event the users that are participating in the training need to immediately resume their operational duties.

c. Roll-Back Module

The roll-back module is similar in design to other training modules in that it is designed to run on the MAST Clients. The main purpose is to ensure the end-host system being used as a MAST Client is returned to the state in which it was prior to the commencement of training.

For example, if a training scenario called for the creation of a text file on the user’s desktop, the roll-back module, which is executed after the SE Server receives its reports, will remove or revert to original construct the text file and any other files created or modified, respectively, during the training.

4. Modular Features

A final characteristic about the MAST that makes it an extensible training tool is its modularity. As stated earlier, scenarios are a combination of computer commands and modules. The modules are designed to execute a single behavior and interact effectively with other modules. For example, if a piece of malware performs multiple behaviors, then those individual behaviors are broken down into individual modules. The scenario created to simulate this malicious behavior would consist of multiple modules.

5. A Scenario Example

Now that we have discussed the characteristics and components of MAST, we can view an example of a scenario that can be used for training. Figure 3 overviews the actions that occur when the Drive-by Download scenario is executed.

In this scenario, a pop-up window appears on the user’s desktop. The window is a simple image that performs no action other than recording the user’s response. The pop-up window asks the user if they would like to execute or download a specific file. The user’s actions are recorded in the SE Server’s database.

The objectives of this scenario are to see how the users respond to the download question and if any users report the events to a system or network administrator. Such events may be characteristic of a phishing attack. The results of the training can let a unit know where to focus future training resources.

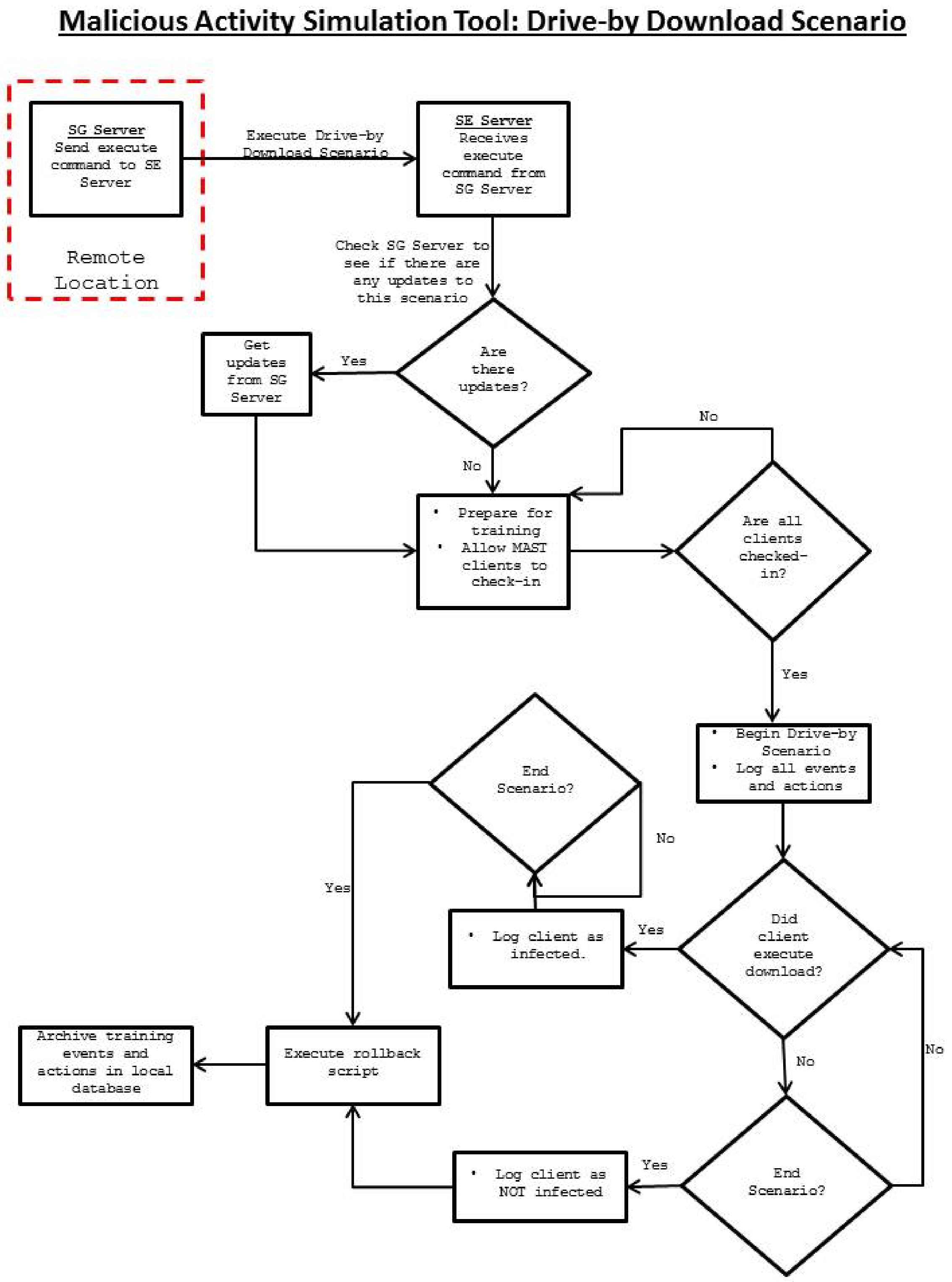

Malicious Activity Simulation Tool: Drive-by Download Scenario
SG Server
Send execute command to SE Server
Remote Location
Execute Drive-by Download Scenario
SE Server
Receives execute command from SG Server
Check SG Server to see if there are any updates to this scenario
Are there updates?
Yes
Get updates from SG Server
No
• Prepare for training
• Allow MAST clients to check-in
Are all clients checked-in?
No
Yes
• Begin Drive-by Scenario
• Log all events and actions
Did client execute download?
Yes
• Log client as infected.
End Scenario?
No
Yes
No
End Scenario?
No
Yes
• Log client as NOT infected
Execute rollback script
Archive training events and actions in local database

Figure 3. Example of a MAST Scenario

C. TESTING PLATFORM

An important aspect in the research and development of MAST is the platform on which the tool is tested and evaluated. In an effort to create a realistic training environment, we have created a virtual environment that simulates the unclassified network of a U.S. Navy ship. By using a network that simulates a real world operational network, we hope to validate MAST as a legitimate training tool for network administrators throughout the DoD.

1. Hardware

The computer hardware used to implement and test the MAST architecture is specifically designed to support virtualization software and the creation of multiple client machines. We are using three Dell PowerEdge R610 servers to run VMware’s ESXi 5.0 software. The hardware specifications for the Dell servers are as follows:

- Server 1: 4 x 1TB Hard Drives, 96GB RAM, (2)Intel® Xeon® Quad-core 2.4GHz processors
- Server 2: 4 x 1TB Hard Drives, 48GB RAM, (2)Intel® Xeon® Quad-core 2.4GHz processors
- Server 3: 4 x 500GB Hard Drives, 24GB RAM, (2)Intel® Xeon® Quad-core 2.4GHz processors

As Figure 4 shows, all three servers are connected using a Dell 2716 Gigabit (Gb) switch. This configuration allows for full duplex communication between the servers and the switch. This setup is important because the three servers, while physically separate, must act as one logical system. The servers need to communicate with each other with little to no latency.

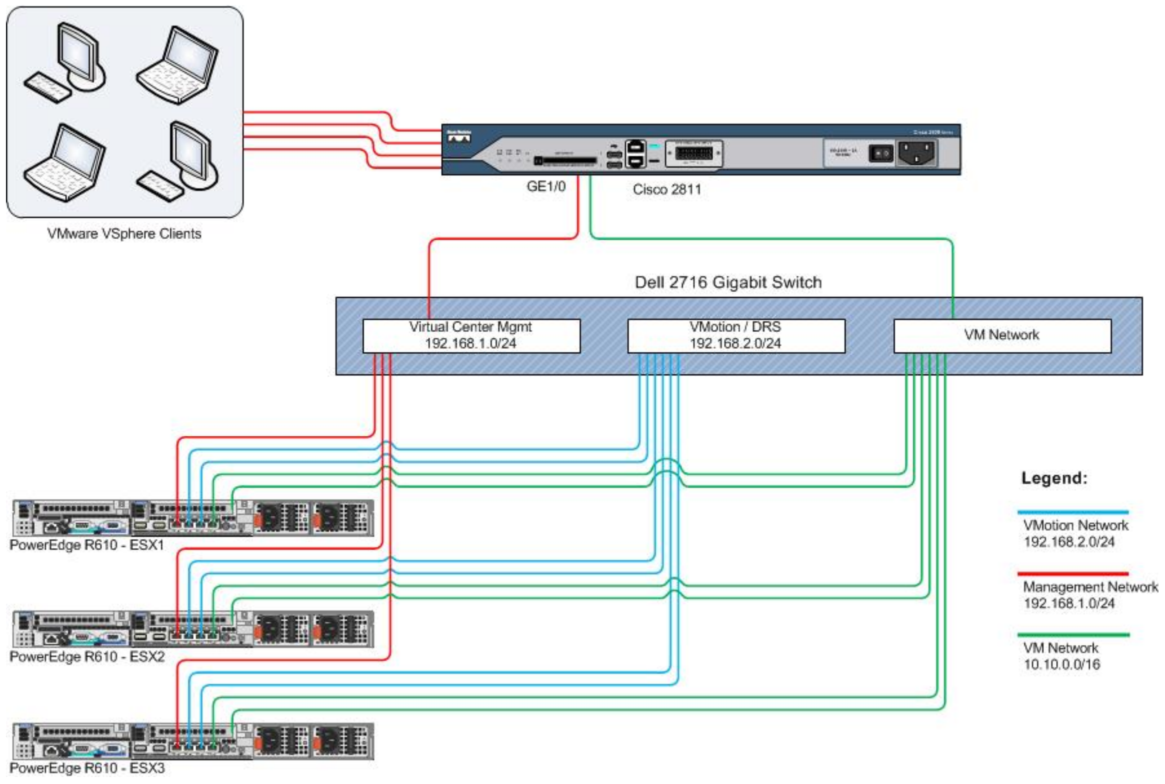


Figure 4. MAST Physical Equipment Setup
(From Greg Belli and Erik Lowney)

Additionally, a Cisco 2811 router is used as an access point for remote hosts to connect to the VMs. Finally, all physical resources are connected to a Dell 1920 Uninterruptable Power Supply (UPS) to ensure protection of the hardware and software in the event of a power loss.

2. Software

The resources required to actually replicate a shipboard network are large and very expensive. A more efficient way to validate the MASTs capabilities is to test the system on a virtualized network. By using virtualization, we were able to reduce the amount of physical resources required to mock the shipboard network. ESXi 5.0 is a specialized operating system developed by

VMware to manage the physical resources available on a server. In our setup, we use VMware software to manage and create virtual machines for testing. A virtual machine (VM), according to VMware, is “a tightly isolated software container that can run its own operating system and applications as if it were a physical computer” [17].

A key element in creating and managing VMs is to ensure you have the appropriate amount of resources available for that virtual machine. For example, if you create a Windows XP VM and allocate 2GB of RAM and 50GB of storage, then those resources will be reserved for that machine on the physical server itself. There is a one-to-one mapping with respect to a VM’s allocated memory and storage and the actual memory and storage on the server on which the VM resides.

In the following section we discuss the actual VMs used for testing. These VMs are managed by the VMware software and reside on the three physical servers mentioned above.

3. Common PC Operating System Environment (COMPOSE) CG-71 Virtual Machines

The virtual machines used to test and develop MAST are a replica of the U.S. Navy cruiser, U.S.S. Cape St. George, also known as CG-71. The VMs, which were developed by Space and Naval Warfare System Center (SPAWARSYSCEN) Pacific contractor, ManTech, are unclassified and have the Common PC Operating System Environment (COMPOSE) installed. COMPOSE is a standardized load for all computers to ensure easier and more efficient management by network administrators. The VMs provided by SPAWARSYSCEN are described below.

a. Integrated Shipboard Network System (ISNS) Domain Controller One and Two

Virtualized Domain Controllers One and Two (DC1 and DC2) have Microsoft Windows Server 2003 Standard Edition installed. The following services are installed as well:

- COMPOSE Data Server
- Dynamic Host Configuration Protocol (DHCP)
- Domain Name System (DNS)
- Active Directory
- Symantec Antivirus Server
- File and Print Servers

b. Integrated Shipboard Network System (ISNS) Exchange Server

The virtualized exchange server has Microsoft Windows Server 2003 Standard Edition installed. The following services are installed as well:

- Exchange Server Standard Edition
- Internet Information Server (IIS) for Simple Mail Transfer Protocol (SMTP)
- Network News Transfer Protocol (NNTP)

c. Integrated Shipboard Network System (ISNS) System Management Server

The virtualized System Management Server has Microsoft Windows Server 2003 Standard Edition installed. The following services are installed as well:

- SQL Server 2005 Standard Edition

- Internet Information Server (IIS) for Simple mail Transfer Protocol (SMTP)
- Network News Transfer Protocol (NNTP)

d. Computer Network Defense-Operating System Environment (CND-OSE) Host-Based Security System (HBSS) Server

The virtualized HBSS Server has Microsoft Windows Server 2003 Standard Edition installed. The following services are installed as well:

- Host-Based Security System (HBSS) Server which includes the ePolicy Orchestrator (ePO)

e. Computer Network Defense-Operating system Environment (CND-OSE) Microsoft Structured Query Language (MSSQL) Server

The virtualized MSSQL Server has Microsoft Windows Server 2003 Standard Edition installed. The server provides a database for HBSS and Secure Configuration Compliance Validation Initiative (SCCVI).

f. CG-71 Common PC Operating System Environment (COMPOSE) Server

The virtualized COMPOSE Server has Microsoft Windows Server 2003 (32 bit) installed. The server manages the COMPOSE environment.

g. CG-71 Common PC Operating System Environment (COMPOSE) Secure Configuration Compliance Validation Initiative (SCCVI) Host

The virtualized SCCVI Host has Microsoft Windows XP Professional (32 bit) installed. The server ensures the COMPOSE workstations are in compliance with HBSS.

h. CG-71 Common PC Operating System Environment (COMPOSE) Workstation

The virtualized COMPOSE Workstation has Microsoft Windows XP Professional (32 bit) installed. The workstation is used by all users and interacts with HBSS through the McAfee Agent installed on the system.

D. HOST-BASED SECURITY SYSTEM (HBSS)

According to the Defense Information Systems Agency (DISA) HBSS website:

> The Host Based Security System (HBSS) baseline is a flexible, commercial-off-the-shelf (COTS) – based application. It monitors, detects, and counters against known cyber-threats to Department of Defense (DoD) Enterprise. Under the sponsorship of the Enterprise-wide Information Assurance and Computer Network Defense Solutions Steering Group (ESSG), the HBSS solution will be attached to each host (server, desktop, and laptop) in DoD. The system will be managed by local administrators and configured to address known exploit traffic using an Intrusion Prevention System (IPS) and host firewall. DISA PEO-MA is providing the program management and supporting the deployment of this solution. [16]

HBSS is currently being deployed by the DoD to standardize the way DoD manages networks with respect to security and IA. Like the use of the COMPOSE CG-71 VMs mentioned in the previous section, it was important to implement HBSS into our testing and evaluation of the MAST. In his thesis, “Verification and Validation of the Malicious Activity Simulation Tool (MAST) for Network Administrator Training and Evaluation,” Neff provides a detailed description of HBSS and its interaction with the MAST [14].

1. McAfee ePolicy Orchestrator (ePO)

Serves as the central policy management point for all of the systems HBSS manages.

2. McAfee Agent

The agent is the distributed client-side software that communicates directly with the ePO server. It also enforces all HBSS policies on the respective workstation.

3. McAfee Host Intrusion Prevention System (HIPS)

The HIPS is the component of HBSS that provides several fundamental security features, such as application blocking or firewalls. The system's functionality is implemented using the following features:

a. Intrusion Prevention System (IPS)

The IPS monitors all system and Application Program Interface (API) calls. It blocks the execution of any program whose signature matches one of the malicious signatures in its database.

b. Host Intrusion Prevention System (HIPS) Firewall

The HIPS firewall protects managed hosts by analyzing network traffic for malicious content and preventing it from compromising any data, applications, or host operating systems.

c. Host Intrusion Prevention System (HIPS) Application Blocking

The HIPS application blocking feature prevents unauthorized applications from executing or binding themselves to another authorized application.

4. Device Control Module (DCM)

The DCM restricts or limits the access of peripheral devices, such as thumb drives and other removable storage devices.

5. McAfee Asset Baseline Module (ABM)

The ABM, which is an extension of the ePO, conducts baseline scans to ensure the state of the system is captured. The latest baseline scan can then be compared to the previous scan to determine updates or changes made.

6. McAfee Policy Auditor (PA)

The PA validates the integrity of a system by scanning and auditing the configuration settings and options of all systems managed by HBSS.

7. McAfee Virus Scan Enterprise (VSE)

The VSE allows for fast and scalable protection of the entire network against known viruses, worms, and other malicious software.

8. McAfee Rogue System Detection (RSD)

The RSP provides the network with multiple “eyes and ears” to determine if any hosts attached to the network are not authorized or not registered.

E. SUMMARY

In this chapter we discussed some issues with current training methods for network administrators. We introduced the characteristics and components of the MAST that address the training shortfalls and an example training scenario used by the MAST. Additionally, we discussed the hardware and software used for testing and evaluating the MAST. We concluded with an overview of the CG-71 VMs and HBSS. In the next chapter we will describe the methodology and results of scalability testing with the MAST.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. SCALABILITY ASSESSMENT METHODOLOGY AND RESULTS

In this chapter, we discuss the objectives of our experiment and the methodology used to determine MAST's scalability properties. Specifically, we discuss

- the deployment of MAST on a new network
- MAST's impact on system and network resources when a scenario is executed for a varying number of clients participating in training, and
- the procedure for generating and distributing all feedback and final reports.

We conclude the chapter with our analysis of the results.

A. MAST DEPLOYMENT AND INSTALLATION

The objective of this assessment was to discuss the methodology and impact of deploying MAST from a remote location to a new training network that does not have MAST installed. Our demonstration and analysis of the MAST software shows that an over-the-air (OTA) deployment and local installation is fast and efficient.

1. Over-The-Air (OTA) Deployment

Figure 5 shows an example architecture where, from a remote location, MAST software is pushed to a local server, which in turn pushes it out to all clients. MAST deployment from a remote location was shown to be effective and efficient because of the small size of the software and the one-time deployment from a remote location. The size of MAST software, to include the server, client, and modules software, is less than 700KB. These packaged files are transmitted once to the SE server, or local server, which in turn handles the distribution to all clients associated with the training network.

Future OTA transmissions will be limited to updates or feedback in the form of reports and statistics pertinent to the training conducted.

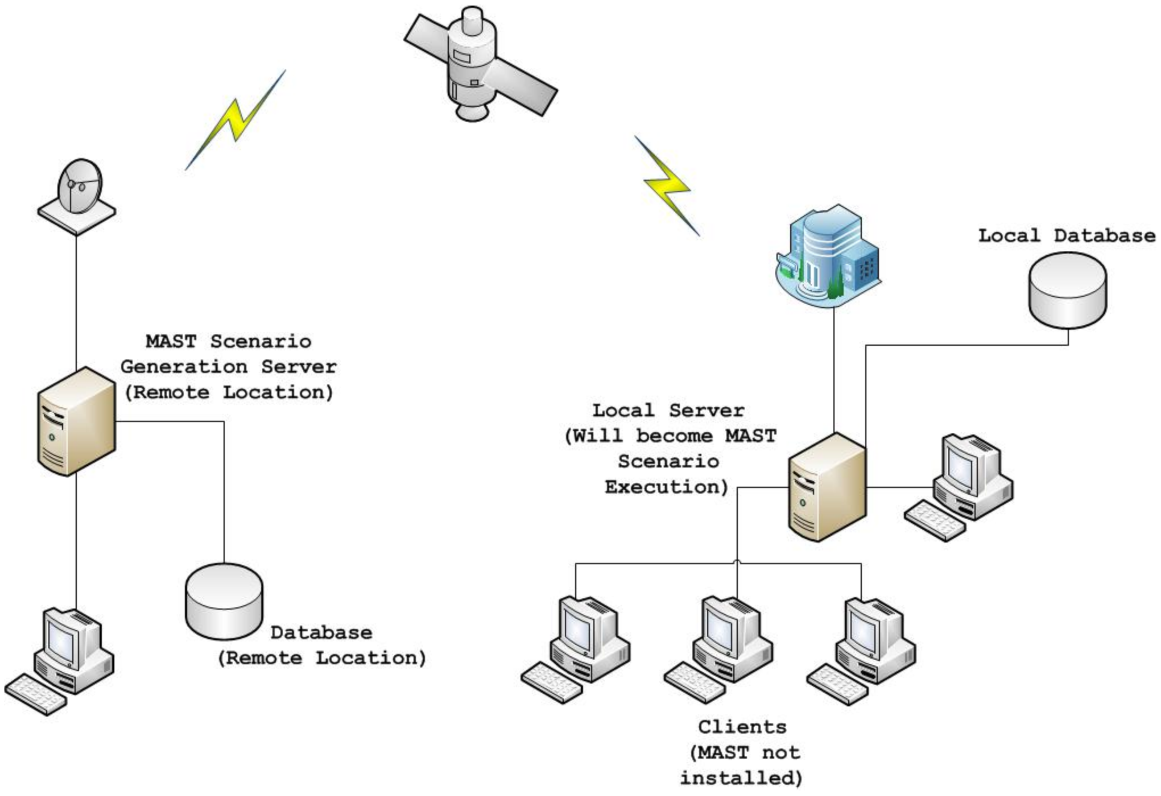

MAST Scenario Generation Server (Remote Location)
Database (Remote Location)
Local Database
Local Server (Will become MAST Scenario Execution)
Clients (MAST not installed)

Figure 5. Architecture for MAST deployment and installation.

2. Local distribution and Installation

Once the local (SE) server receives the software from the remote location, it can distribute the client software to all hosts on the training network. The client software and training modules are less than 400KB in size. The SE server can easily deploy this software during any of standard updates that occur with HBSS, Microsoft software, or any other DoD authorized updates.

Installation of the software on local hosts is as simple as placing a file on the desktop. MAST client software is designed to run, or execute, only when the respective host is participating in training. The software is resident on all hosts, but takes up very little space and zero system resources when not in use. The following section discusses the impact on system resources when a scenario is executed and the software is utilized.

B. SCENARIO EXECUTION

The overall goal of this experiment was to determine how MAST uses and impacts system and network resources. Through a standardized set of input and procedures, we wish to show that MAST performs as expected when utilized in an environment simulating an operational network that consists of multiple clients in a remote location.

1. System Resources

For this objective, our goal was to monitor and report the processing resources utilized by the SE server. It was critical that we understood how much of the server’s central processing unit (CPU) was used to serve as few as five clients and as many as 80 clients. These observations would help us estimate and plan for testing and evaluating on a non-virtual operational network consisting of hundreds of clients.

2. Network Resources

For this objective, our goal was to monitor and report the amount of network traffic generated between the SE server and clients when executing a scenario. Since MAST is designed to run on a network that is not only being used for training, but for operational purposes as well, it was important we understood the impact on the network while conducting training. These observations would help us understand the network traffic attributes of our current scenarios, and assist in the planning and design of future training modules and scenarios.

3. Experiment Design

In order to conduct this experiment, we used the hardware and software described in Chapter III, Section C. Specifically, we created a Windows XP Service Pack (SP) 3 virtual machine, which hosted the MAST SE server software and Wireshark for network traffic monitoring. The machine was configured with a 2.4 GHz Intel Xeon processor and Gigabit (Gbit) network adapter card. The machine was also inter-connected to five COMPOSE servers and 75 COMPOSE workstations, each of which had the MAST Client software installed. Figure 6 shows a graphical view of the simulated environment in which the experiment was conducted. The physical setup upon which these virtual machines are hosted is detailed in Chapter III, Figure 4.

One limitation identified during the creation and configuration of the 75 COMPOSE workstations was the impact of the VMs on the physical servers when all VMs were operational. While the server had plenty of remaining memory and storage for more workstations, the creation of more VMs would have been counter-productive to the experiment due to the workload on the physical server’s CPU.

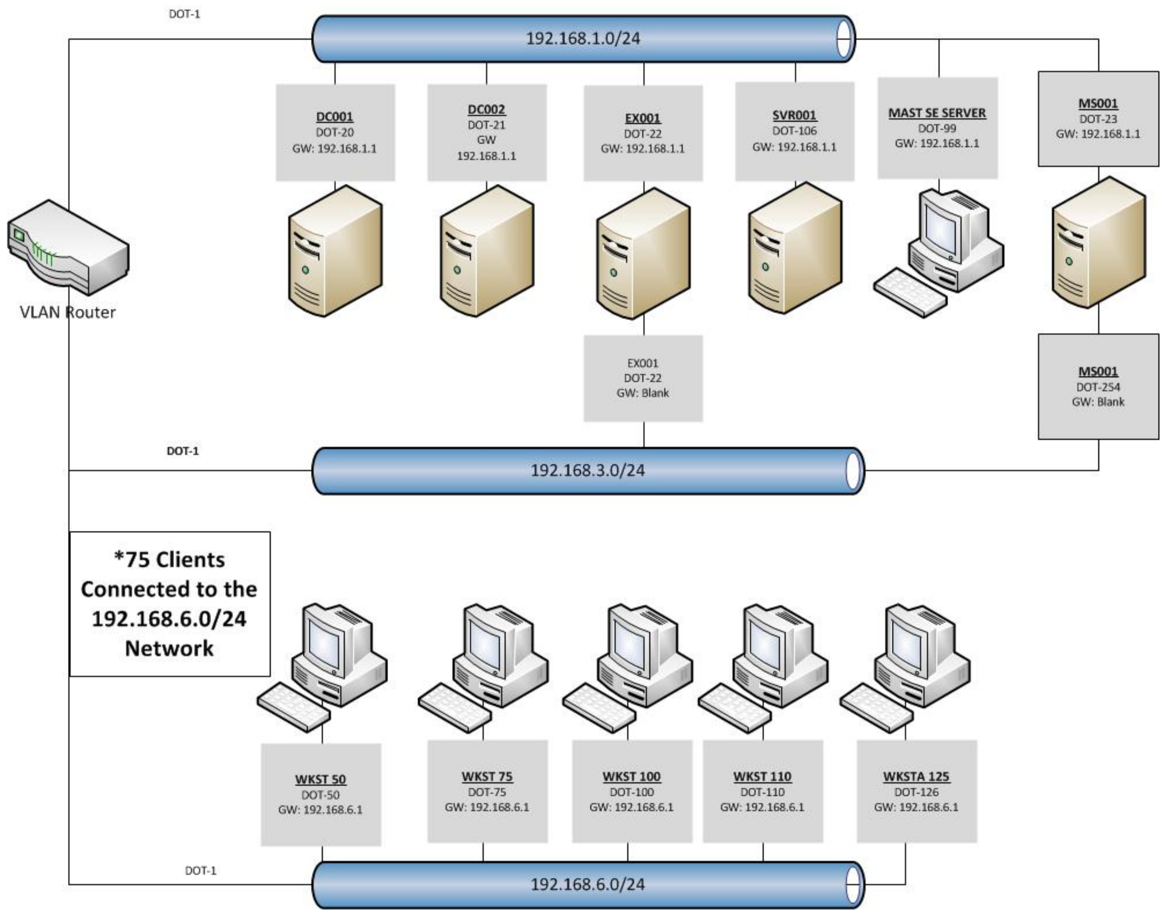
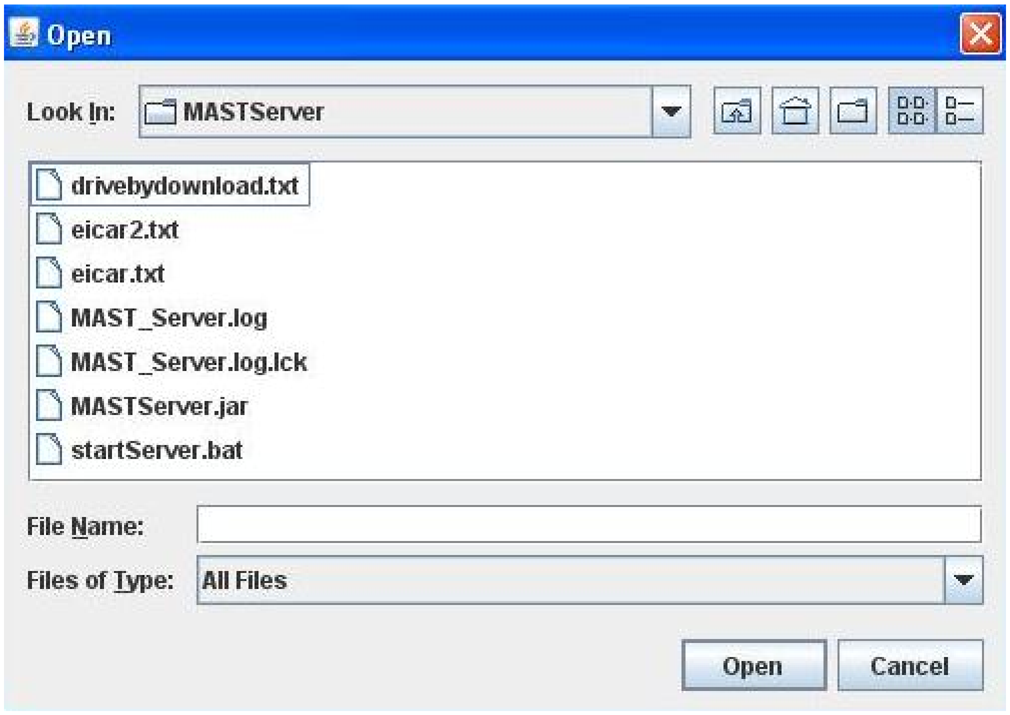


Figure 6. Virtual test bed configuration

In order to create 75 COMPOSE workstations, we created a template from the CG71 COMPOSE workstation VM. That template was then deployed 75 times to create 75 individual machines. Once all 75 were created and deployed, we manually updated the Internet Protocol (IP) address and computer name for each workstation. This ensured there were no conflicts on the network and ease of registration with the network’s domain controllers. Connectivity among all the systems was confirmed with "ping" requests to neighboring systems and systems located on other sub-networks.

The final step in completing the experiment setup was to test the pre-installed scenarios' functionality and correctness. A training scenario is executed by starting the SE server first, followed by all of the clients participating in the training. This order is critical as the server must be operational in order for the clients to "check-in." Once all the clients participating in the training are logged onto the SE server, a training scenario is selected from the SE server menu. The scenario continues until the stop, halt, or quit command is issued.



```
drivebydownload.txt
eicar2.txt
eicar.txt
MAST_Server.log
MAST_Server.log.lck
MASTServer.jar
startServer.bat
```

Figure 7. MAST Scenario selection window

4. Experiment Methodology

In order to determine MAST’s scalability characteristics, we conducted five different experiments using the same scenario for each evolution. Figure 8 shows how we divided the MAST clients.

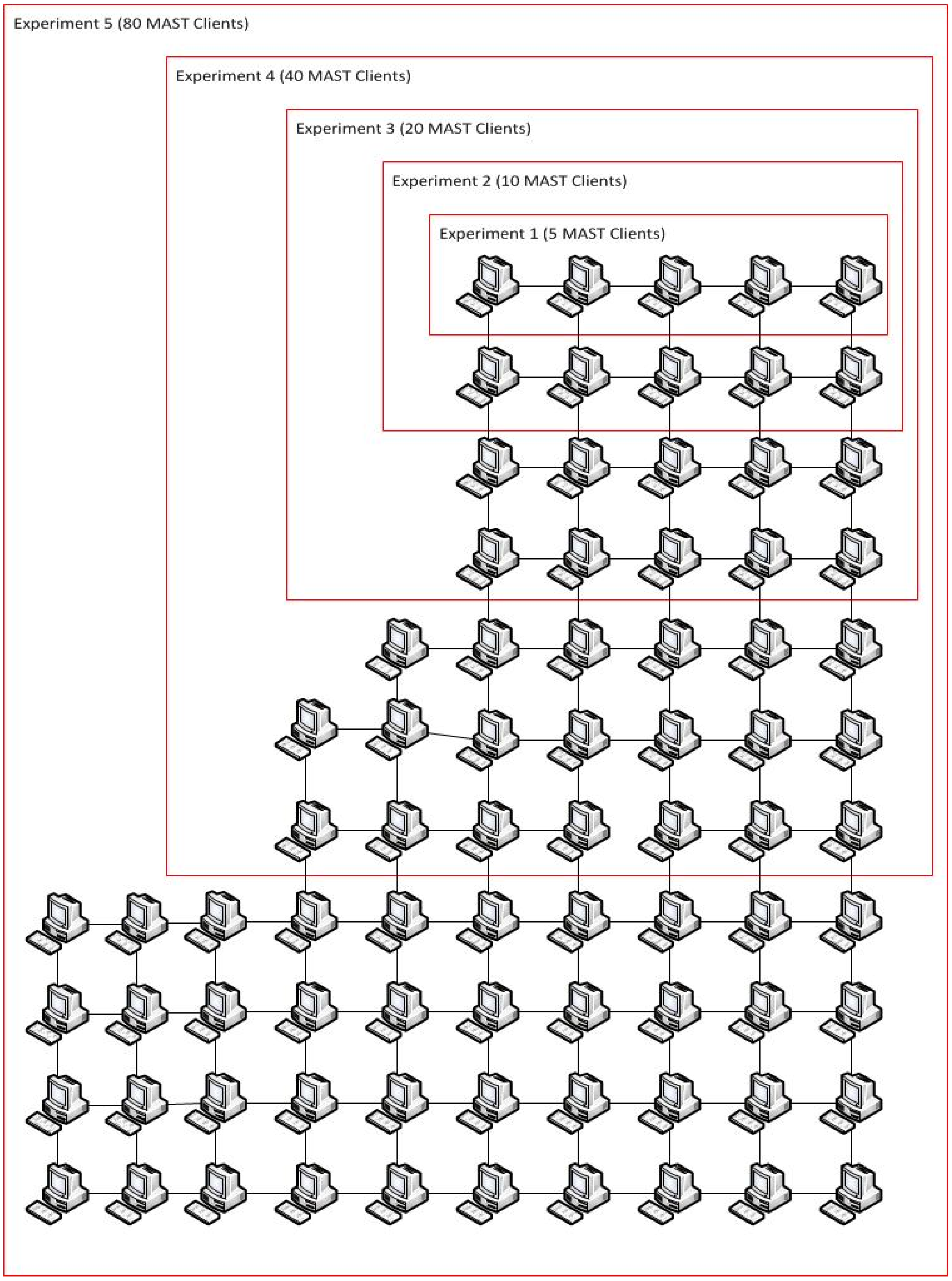
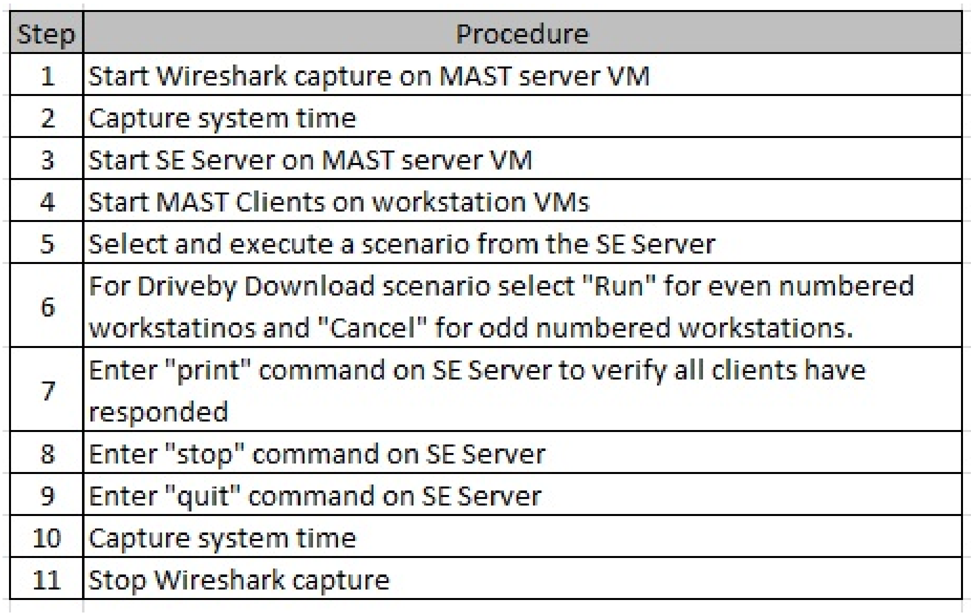

Experiment 5 (80 MAST Clients)
Experiment 4 (40 MAST Clients)
Experiment 3 (20 MAST Clients)
Experiment 2 (10 MAST Clients)
Experiment 1 (5 MAST Clients)

Figure 8. Breakdown of MAST clients for experimentation

Each experiment followed the procedures shown in Figure 9. The only difference between each experiment was the number of clients involved in the training.



Step	Procedure
1	Start Wireshark capture on MAST server VM
2	Capture system time
3	Start SE Server on MAST server VM
4	Start MAST Clients on workstation VMs
5	Select and execute a scenario from the SE Server
6	For Driveby Download scenario select "Run" for even numbered workstatinos and "Cancel" for odd numbered workstations.
7	Enter "print" command on SE Server to verify all clients have responded
8	Enter "stop" command on SE Server
9	Enter "quit" command on SE Server
10	Capture system time
11	Stop Wireshark capture

Figure 9. Experiment procedure

For CPU utilization analysis, we used the “performance” tab offered by the vSphere Client window. Additionally, this same tab was used to gather data on the network resources used during training. A final tool used for analysis was Wireshark. Wireshark captured all traffic traversing the network during all experiments. We then applied a filter to each capture to isolate and view only the traffic to and from the SE server.

The final analysis used all the above resources to compare the amount of network traffic generated by each experiment along with the SE server’s CPU utilization for each experiment.

5. Results

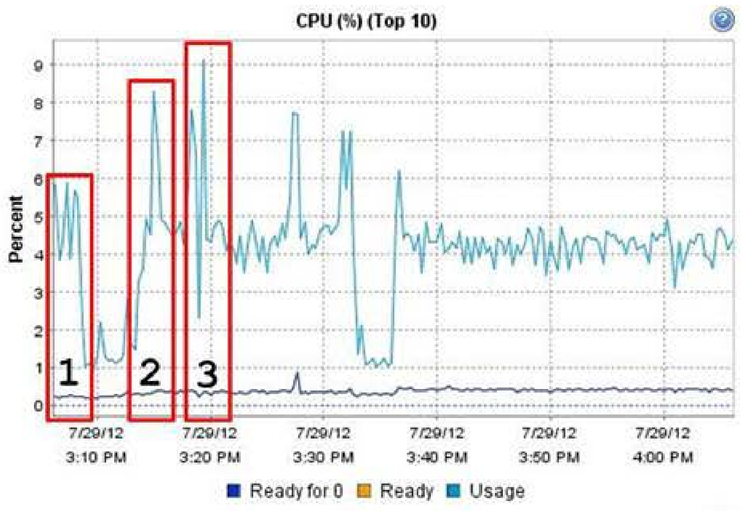
Overall, the experiment verified system performance with respect to scalability. An increase in the number of clients tested did not result in a similar proportional increase in utilization of processing resources. Additionally, an increase in the number of clients and network traffic generated to control those clients resulted in very minimal use of network resources.

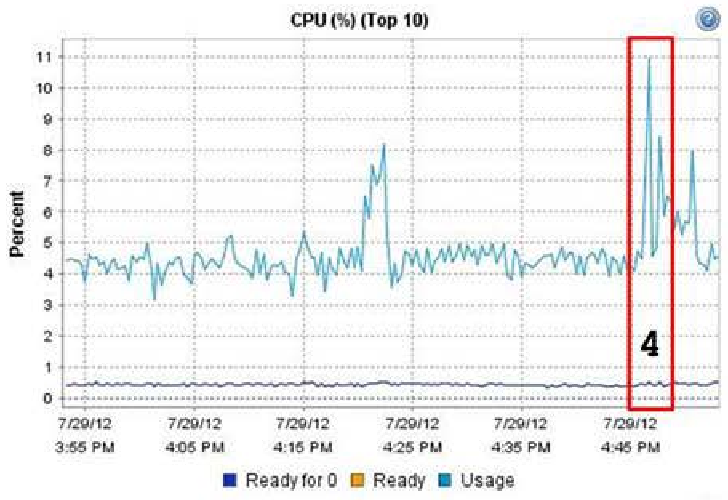
a. System Resources

The performance of the computer hosting MAST showed limited impact as the number clients involved in the experiment grew exponentially.

Figure 10 graphs shows CPU utilization for each experiment when a scenario was executed. Specifically, the rectangles labeled with numbers show the percentage of the CPU’s resources used during that respective experiment. Experiment five for example, which connected to 80 clients simultaneously, utilized just over 15% of the systems CPU resources.

There were some spikes and lulls depicted in the graph that are not associated to the experiment (3:30 – 3:40 PM). Analysis of the network traffic during these periods shows administrative communication between the virtual machine and the vSphere client.





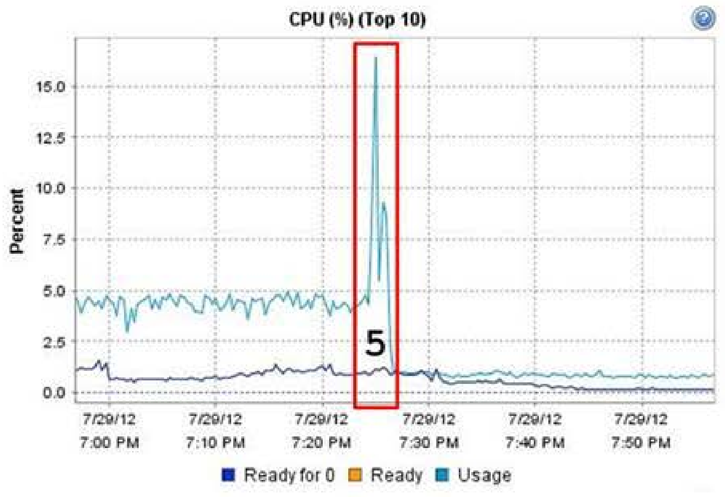


Figure 10. Percentage of CPU resources used for experiments

As Figure 11 depicts, an exponential increase in clients does not exponentially increase the amount of resources needed to conduct training. MAST’s performance demonstrates the minimal impact on CPU resources and the capability to serve more clients with ease.

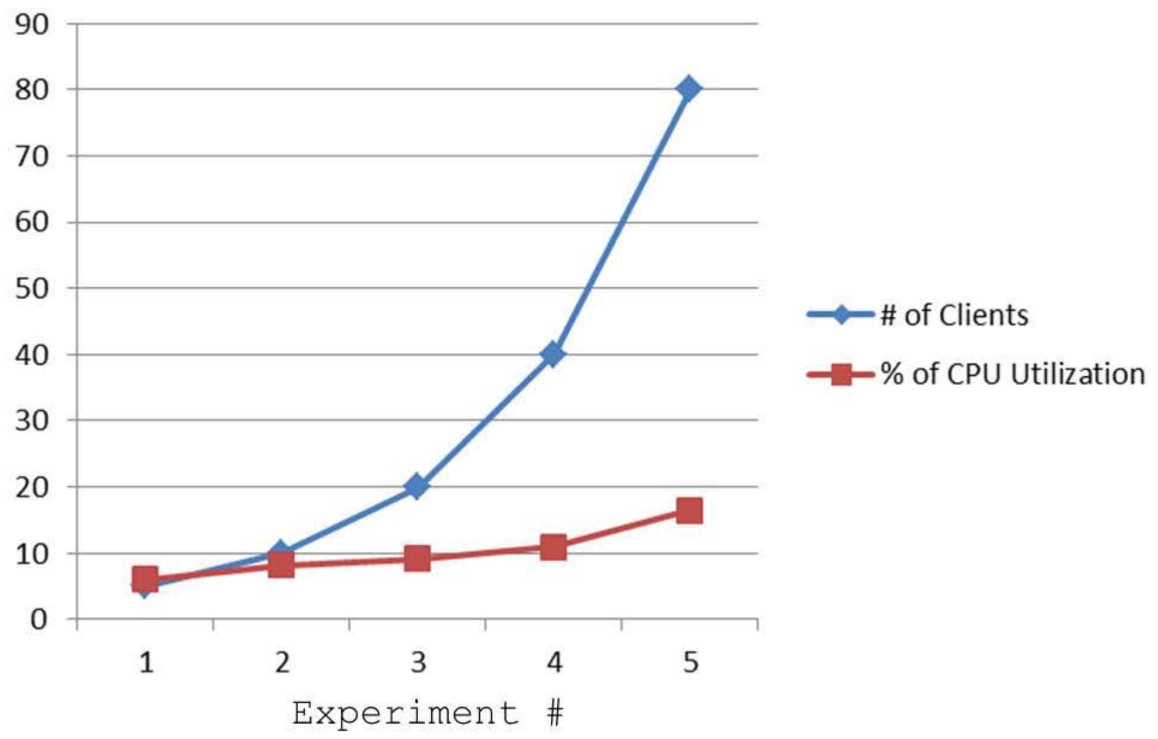


Figure 11. Percentage of CPU used compared to number of clients.

b. Network Resources

The utilization of network resources during the execution of all scenarios was extremely minimal. Figure 12 details the statistics of the network traffic generated during all five experiments.

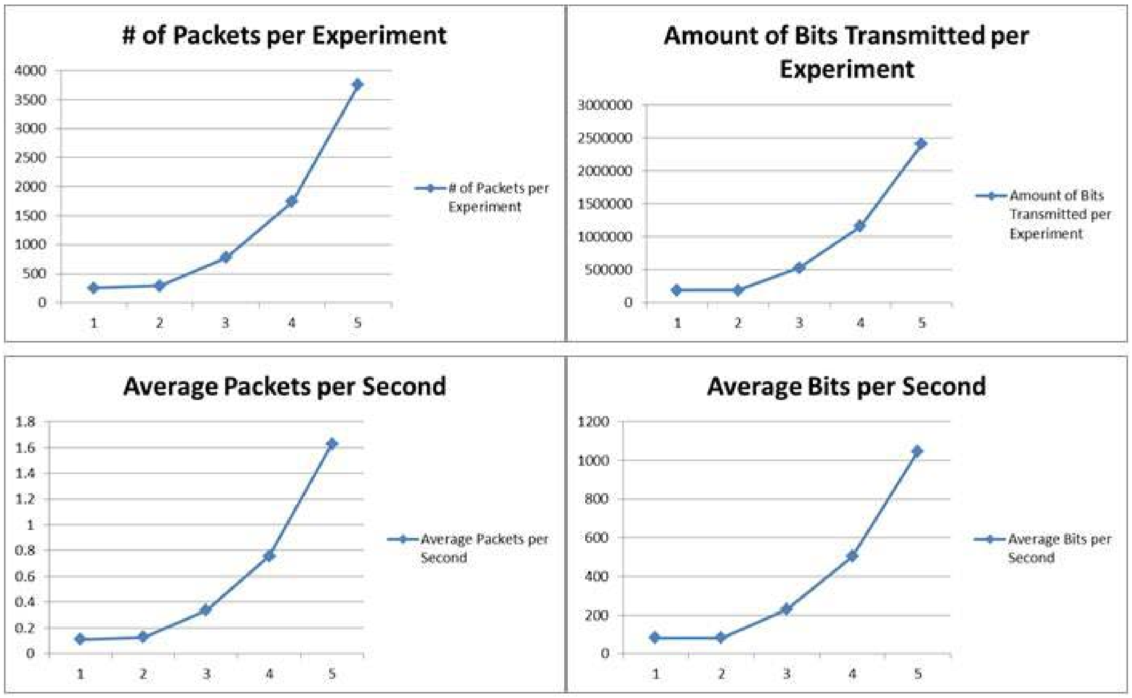


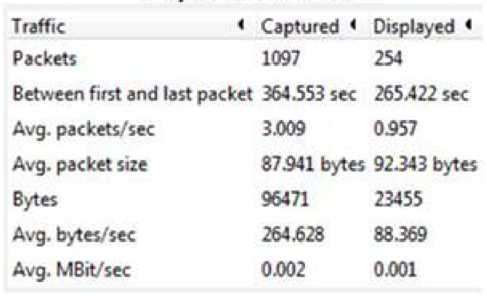
Figure 12. Characteristics of network during experiments

The exponential increase among these characteristics during all experiments was expected. Unlike the use of CPU resources, there is a direct correlation between the number of clients and the amount of traffic generated. An exponential increase in clients means a mirrored increase in network traffic to control those clients.

Despite this increase in network traffic, the percentage of network resources used to support the training was very minimal. The experimental network was configured to support a Gbit/sec throughput between all systems.

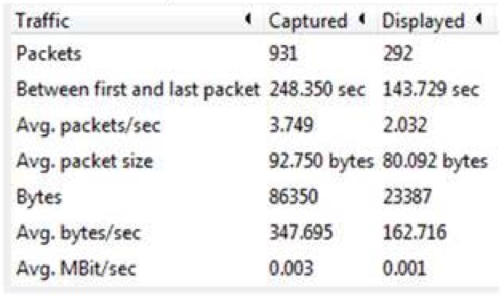
Figure 13 provides a summary of the network statistics captured by Wireshark for all experiments. The "captured" column details all packets captured during the experiment while the "displayed" column shows the details of network traffic directly associated with the SE server and our experiments.

Experiment 1



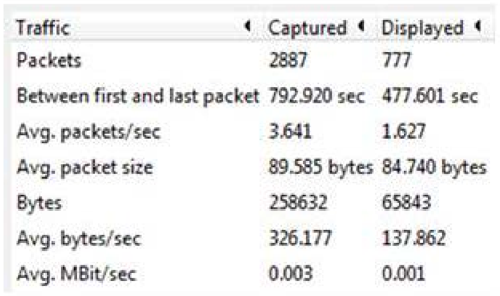
Traffic	Captured	Displayed
Packets	1097	254
Between first and last packet	364.553 sec	265.422 sec
Avg. packets/sec	3.009	0.957
Avg. packet size	87.941 bytes	92.343 bytes
Bytes	96471	23455
Avg. bytes/sec	264.628	88.369
Avg. MBit/sec	0.002	0.001

Experiment 2



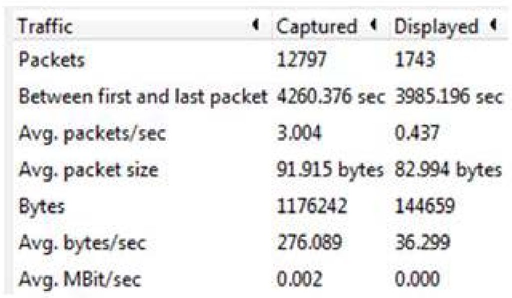
Traffic	Captured	Displayed
Packets	931	292
Between first and last packet	248.350 sec	143.729 sec
Avg. packets/sec	3.749	2.032
Avg. packet size	92.750 bytes	80.092 bytes
Bytes	86350	23387
Avg. bytes/sec	347.695	162.716
Avg. MBit/sec	0.003	0.001

Experiment 3



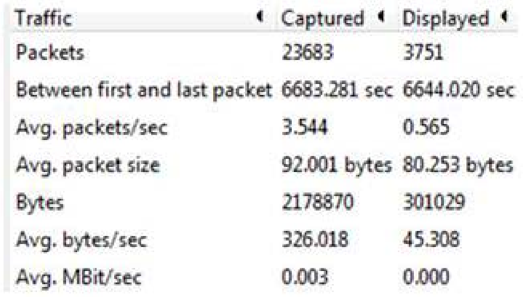
Traffic	Captured	Displayed
Packets	2887	777
Between first and last packet	792.920 sec	477.601 sec
Avg. packets/sec	3.641	1.627
Avg. packet size	89.585 bytes	84.740 bytes
Bytes	258632	65843
Avg. bytes/sec	326.177	137.862
Avg. MBit/sec	0.003	0.001

Experiment 4



Traffic	Captured	Displayed
Packets	12797	1743
Between first and last packet	4260.376 sec	3985.196 sec
Avg. packets/sec	3.004	0.437
Avg. packet size	91.915 bytes	82.994 bytes
Bytes	1176242	144659
Avg. bytes/sec	276.089	36.299
Avg. MBit/sec	0.002	0.000

Experiment 5



Traffic	Captured	Displayed
Packets	23683	3751
Between first and last packet	6683.281 sec	6644.020 sec
Avg. packets/sec	3.544	0.565
Avg. packet size	92.001 bytes	80.253 bytes
Bytes	2178870	301029
Avg. bytes/sec	326.018	45.308
Avg. MBit/sec	0.003	0.000

Figure 13. Network traffic statistics captured by Wireshark

Figure 14 details the percentage of network resources used during each experiment. The amount of traffic generated for all experiments was so low, it was not reported by the vSphere client. We used our Wireshark captures to determine the amount and size of packets generated during all experiments.

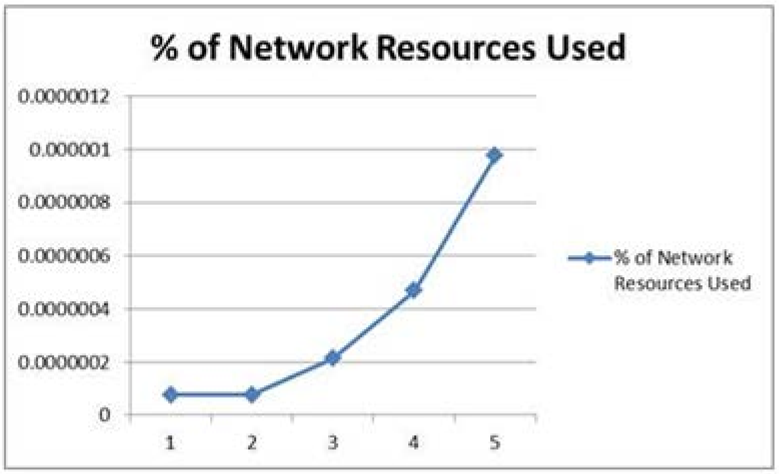


Figure 14. Percentage of network resources used

As the analysis of the network traffic has shown, an exponential increase does not significantly impact the resources available. A correlation between the two does not exist. The demonstration of the MAST design and implementation and the scenarios utilized assert its ability to have very minimal impact on a network.

C. TRAINING FEEDBACK AND DISTRIBUTION

The final scalability factor that we analyzed was the distribution of feedback and results to the SE server and the SG server. As stated in the previous chapter, one of MASTs key functionalities is its reporting capability.

The method in which the SE server captures user actions for reports and feedback is network and system efficient. MAST client software is designed to report all user actions to the SE server as they happen and capture only those actions on the local client that affect the state of the machine. It captures these actions by documenting all changes to the system in a text file, which is then used as part of the roll-back module. The roll-back module parses the file to determine what needs to be done to return the system to its pre-training state. Once the host is back to its original state, the text file is deleted.

While training is being conducted, MAST captures all user actions by sending them directly to the SE server, as they happen, for storage in the local database. This approach does increase the amount of traffic but the size of the traffic is very small with very minimal impact on network resources.

Finally, the report to the SG server can vary based on the needs of the trainer or evaluator. The SE server and local database can customize reports to send only high-level statistics or compile all data into a user friendly text file to transmit back to the SG server. The SG server can take the data and produce its own reports and diagrams.

D. SUMMARY

Overall, the analysis and experiment demonstrated MASTs ability to be deployed, scale up with little to no impact on network and system resources, and submit feedback and reports with efficiency. All the experiment objectives were met along with an observable validation for each portion of the experiment.

Next, in Chapter V, we discuss our conclusions. We also discuss our thoughts on the development of future scenarios and the implementation of MAST on an operational network.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS AND FUTURE WORK

A. CONCLUSIONS

In this thesis, we showed that MAST’s use of system and network resources is minimal and the ability to scale up to train more clients will not impact other users and processes not participating in the training. We also discussed and analyzed the method in which MAST would be installed on a network and the process and procedures for providing reports on all events and actions.

In Chapter III, we outlined our assumptions about training objectives and the training environment in which MAST would be implemented. We discussed the shortfalls with current network security and IA training methods and the benefits of implementing MAST to address those shortfalls. We detailed MAST’s architecture and functionality along with an example training scenario using MAST. We described and defined the hardware and software configurations used to test MAST’s scalability properties.

In Chapter IV, we discussed three factors of MAST that are critical to scalability. First, we discussed how MAST would be installed on a new network and the impact of that installation from a remote location. We followed that analysis with a set of experiments of MAST on a simulated shipboard network. The results showed that an exponential increase in host systems being trained did not result in an exponential increase in utilization of processing resources. Additionally, we showed that the network traffic generated to control all the clients being trained was minimal in size and barely noticeable when monitoring all network traffic. We concluded the chapter with a demonstration of MAST’s reporting capabilities.

We demonstrated that MAST can scale up to train more clients while minimizing the use of system and network resources. Additionally, we demonstrated that MAST can be effectively and efficiently installed on a new network and provide reports and feedback as needed to meet projected training goals and objectives.

B. FUTURE WORK

1. Continued Development of Module Library

A critical component of MAST is the modules used to create scenarios. Currently, there are a limited number of modules that can be used for creating scenarios. As discussed in Chapter II, modules are the actions or behaviors we program that simulate a real world threat. Varying types of modules are needed to ensure the training provided is realistic and relevant. As malware is created or evolves, it is important to develop modules that simulate their behavior to ensure new and updated scenarios can be created and used. The development of such may be appropriate for small student projects in a network security course. Developing a methodology for developing the modules that could be exported to other organizations, such as the red teams units. This methodology could also be used to capture lessons-learned at Cyber Defense Exercises (CDX).

2. Graphical User Interface

As the reporting functionality of MAST improves, it is important to maximize this value by providing a graphical user interface (GUI) that is informative and user friendly. Currently, the GUI for interaction, feedback, and results is limited. Areas that will benefit from the implementation of a GUI include the scenario generation function and the reporting function.

As the module library becomes more populated, the trainer will have the ability to create more scenarios that are unique or robust. The manner in which these scenarios are created and tested can be expedited with the use of a GUI. Additionally, the reporting functionality of MAST is critical to the feedback required for any training evolution. A report GUI would allow for immediate feedback, which in turn can help prioritize and utilize training resources for future evolutions.

3. Test and Evaluation on Operational Network

Finally, as MAST continues to evolve, develop, and perform as expected in a simulated training environment, it is important to begin some assessments on a physical network. Currently, all assessments on performed in a virtual environment. Utilizing a physical environment will help further test and evaluate MAST’s system properties and scalability characteristics. Additionally, it will allow for assessments of the module library and their performance on host systems with varying operating systems. Such assessments and demonstrations are critical to its acceptance by the operational community and its subsequent porting to the target objective: operational networks.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] U.S. Department of Defense. *Cyber Command Fact Sheet* [Online]. Available: http://www.defense.gov/home/features/2010/0410_cybersec/docs/cyberfactsheet%20updated%20replaces%20may%2021%20fact%20sheet.pdf

[2] W. R. Taff Jr. and P. M. Salevski, "Malware Mimics for Network Security Assessment," M.S. thesis, Dept. Comput. Sci., Naval Postgraduate School, Monterey, California, 2011.

[3] J. M. Neff, "Verification and Validation of the Malicious Activity Simulation Tool (MAST) for Network Administrator Training and Evaluation," M.S. thesis, Dept. Comput. Sci., Naval Postgraduate School, Monterey, California, 2012.

[4] G. Derene, "Inside NSA Red Team Secret Ops With Government's Top Hackers," *Popular Mechanics*, 30-Jun-2008. [Online]. Available: http://www.popularmechanics.com/technology/how-to/computer-security/4270420. [Accessed: 02-Apr-2012].

[5] J. F. Sandoz, "Red Teaming: A Means to Military Transformation," Final IDA Paper P-3580, Jan 2001. Available from DTIC, Alexandria, VA.

[6] "National Information Assurance (IA) Glossary." Committee on National Security Systems, 26 Apr 2010.

[7] C. Babb, "ONR Demonstrates Revolutionary Infantry Immersion Trainer to Joint Chiefs Chairman," *Office of Naval Research (ONR), Media Release*, 2008.

[8] T. Gold and B. Hermann, "The Role and Status of DoD Red Teaming Activities," Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics, Defense Science Board (DSB) Task Force, Washington D.C., 2003.

[9] S. Cote, R. Petrunic, P. Branka, N. T. Khalil, M. Schumak, C. Chavez and A. Silva, Certified Ethical Hacker: Ethical Hacking and Countermeasures, Courseware Guide v7.1, vol. 1. Albuquerque, NM: EC-Council USA, 2011.

[10] R.S. Greenwell, “DoD IA Training Products, Tools Integration, and Operationalization.” [Online]. Available: http://www.disa.mil/News/Conferences-and Events/~/media/Files/DISA/News/Conference/CIF/Briefing/IA_DOD_IA_Training_Products.pdf

[11] T. Nash, “An Undirected Attack Against Critical Infrastructure: A Case Study for Improving Your System Security.” U.S. Department of Homeland Security (DHS), Vulnerability & Risk Assessment Program (VRAP), Sept 2005.

[12] F. E. Froehlich and A. Kent, “The Froehlich/Kent Encyclopedia of Telecommunications,” vol. 15, New York, New York: Marcel Dekker, Inc., 1997, pp. 231–255.

[13] G. Gu, M. Sharif, X. Qin, D. Dagon, W. Lee and G. Riley,“Worm Detection, Early Warning and Response Based on Local Victim Information,” in Computer Security Applications Conference, 2004, pp. 136-145.

[14] P. Szor, The Art of Computer Virus Research and Defense, 1st ed. Upper Saddle River, NJ: Addison Wesley, 2005.

[15] E. Messmer.(2009, Jul. 9).*The Botnet World is Booming*. [Online].Available: http://www.networkworld.com/news/2009/070909-botnets-increasing.html

[16] “Virtual Machines, Virtual Server, Virtual Infrastructure,” *Virtualization Basics*. [Online]. Available: http://www.vmware.com/virtualization/virtual-machine.html. Accessed 10 Jul 2012.

[17] “Host Based Security System.” [Online]. Available: http://www.disa.mil/Services/Information-Assurance/HBS/HBSS. Accessed 12 Jul 2012.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Fort Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Marine Corps Representative
   Naval Postgraduate School
   Monterey, California

4. Director, Training and Education, MCCDC, Code C46
   Quantico, Virginia

5. Director, Marine Corps Research Center, MCCDC, Code C40RC
   Quantico, Virginia

6. Marine Corps Tactical Systems Support Activity
   (Attn: Operations Officer)
   Camp Pendleton, California

7. Captain David Aland, USN (Ret.)
   Office of the Director, Operational Test & Evaluation
   Washington, D.C.

8. Dr. Gurminder Singh
   Naval Postgraduate School
   Monterey, California

9. Mr. John H. Gibson
   Naval Postgraduate School
   Monterey, California