Developing Simulated Cyber-Attack Scenarios Against Virtualized Adversary Networks

Luis Aybar, Gurminder Singh and Alan Shaffer
Naval Postgraduate School, Monterey, USA
leaybar@nps.edu
gsingh@nps.edu
abshaffe@nps.edu

Abstract: Cyberspace is recognized as a critical domain in modern warfare. The ability of military forces to maintain and secure their own operational networks, while simultaneously degrading or denying the ability of adversaries to operate their networks, is a critical strategic objective for military planners and leaders. Conducting effective offensive cyber operations (OCO) against sophisticated adversary networks requires the ability to develop, test, and rehearse cyber-attack actions before they are employed operationally. This requirement is well understood and practiced in the physical warfare domains, where ships, aircraft and tanks can exercise their capabilities against physical targets; it is not, however, well refined in the cyber domain. This research introduces a framework to address this need, and demonstrates a prototype for cyber-attack scenario development and rehearsal in a virtual network environment. By extending the earlier work of the Naval Postgraduate School's Malicious Activity Simulation Tool (MAST), a distributed client-server based software tool designed to launch inert malware attacks on live networks, we were able to demonstrate cyber-attack scenarios based on temporal specificity and target discrimination as attack parameters. Our prototype accurately models an adversary network in a virtual environment, providing the ability to develop cyber-attack actions to achieve specific cyber effects against hosts on the intended target network. The architecture allows cyber forces to rehearse specific cyber actions prior to launching a cyber-attack, in order to provide a more accurate assessment of the efficacy of these actions against a realistic model of the target network. This framework allows military forces to better train and prepare for cyber operations to help achieve cyber superiority in modern warfare.

Keywords: offensive cyber operations, simulated malware, cyber-attack rehearsal, cyber effects development, virtualized networks

1. Introduction

For today's military, the use of offensive cyber operations (OCO) has grown dramatically in recent years, and the Internet has become the military's "theater of spying, sabotage and war" (Gellman and Nakashima, 2013). Computer networks are an essential element of combat capability, providing command and control, communications, and logistics, as well as computer network attack and exploitation. Given the importance of computer networks in carrying out a broad spectrum of warfare functions, they have become high priority targets in modern military conflicts, and by extension, the exploitation of enemy networks has become a key military objective.

In the traditional warfare domains, the ability to ascertain the capabilities and orders-of-battle (OOB) of adversary forces is well understood. Thorough knowledge of an enemy's forces allows for more accurate and complete testing and refinement of warfare capabilities, which facilitates effective operational planning. In the cyber domain, however, the ability to understand an adversary's capabilities, vulnerabilities, and OOB is less defined, and the ability to develop and accurately test offensive cyber capabilities against an enemy's network is not as mature. OCO missions are often executed without the ability to test and rehearse their efficacy against a realistic model of a target network. Having such a capability could help cyber operators better understand the effects of specific cyber-attacks, including their impact on an adversary's systems and operations, and the secondary and collateral effects that could result.

To address these gaps we have developed a software environment for creating, testing, and rehearsing simulated cyber-attack scenarios against virtualized models of an adversary network. The key contributions of this research are the development of a virtual environment to support development and testing of offensive cyber-attack scenarios, a sophisticated control mechanism for developing simulated malware (SimWare) modules, and the enumeration of simulated malware modules to be used in developing cyber-attacks.

Section 2 of this paper describes previously work related to this research. Section 3 describes the architecture of the MAST framework, and section 4 describes key design requirements for the creation of a virtual environment in which to conduct simulated cyber-attacks against a virtualized adversary network. Section 5 details the test environment, the creation of the new software modules, and the testing process. Section 6 concludes with a discussion of the results of this research and future work.

2. Background and related work

The need for military forces to adequately plan and train for operations in the cyberspace realm, in support of both offensive and defensive objectives, has become well-understood in recent years. According to Sheldon (2012), nation-state and terrorist-based cyber threats exist that can do serious harm to national security interests, and that proper defense against these threats requires "extensive, even onerous, preparations and resources." Grant (2013) points out that the national strategies of most modern nations identify cyber operations, and the ability to perform network defense and exploitation, as critical to their national defense. Further, he highlights the need for proper infrastructure and training, among other criteria, to support these objectives.

Offensive cyber operations are divided generally into computer network attack (CNA) and computer network exploitation (CNE). CNA is focused on denying or degrading the ability of an adversary to use its own operational networks and the data stored within, i.e., attacking the network within the cyberspace (as opposed to a physical attack against the network infrastructure). CNA is most often associated with military operations. By contrast, CNE involves gaining access into an adversary network for the purposes of exploiting the sensitive or classified data stored therein, and is usually associated with cyber espionage (Rattray and Healer, 2010).

Each of the two classes of OCO has its own, often disparate, objectives, however the tools and techniques used are similar. For both, a thorough analysis of intended and collateral effects of a cyber action is critical to ensuring missions success, as well as to ensuring abidance to the laws of armed conflict in cyberspace. As pointed out by Fanelli and Conti (2012), even in cyberspace, the need to exists to limit as much as possible the effects of offensive actions to only intended targets. In order to meet these demands of cyber warfare in both CNA and CNE, military forces require appropriate network-based tools and technologies to fully develop and test specific cyber actions against accurate representations of an adversary network.

Numerous COTS (commercial-off-the-shelf) network penetration testing (pentesting) tools exist that can be used to support the development of cyber-attack tools and methods. Popular pentesting software tools and systems include the Metasploit Framework (MSF), STEPfwd, SafeBreach, Arena, and Core Impact Pro.

The MSF is a suite of pentesting tools that can be used to develop and deploy custom designed cyber-attack modules (i.e., cyber exploits with associated malware payloads) to achieve a desired cyber effect (Kennedy et al, 2011). Since MSF tools are primarily designed for testing singular cyber-attacks, the architecture limits the nature and flexibility for developing a large variety of attack scenarios.

Core Impact Pro and SafeBreach are network pentesting tools that utilize a proprietary collection of known offensive network exploitation techniques via an automated pentesting algorithm that continuously attempts to penetrate a target network (Green, 2016). The SafeBreach platform does not offer a way for exploits to be tested in a rapidly configurable and benign test environment, and both tools lack the ability to design unique attack scenarios

STEPfwd (Simulation, Training and, Exercise Platform) was developed by the Software Engineering Institute at Carnegie Mellon University, and enables virtualized training simulations that closely mimic real-world cyber infrastructures and attacks. The tool uses Virtual Training Environment (VTE) to facilitate the knowledge and skill building phases of training, and Exercise Network (XNET) as a platform for remote instructors to create customized, full-scale cyber exercise scenarios to simulate real-world environments (Hammerstein and May, 2010; Mayes, 2014). STEPfwd can support the development of simulated attack scenarios against virtualized adversary networks, however, it lacks the ability to map new networks, and is not designed to be used in an operational environment.

The University of Rochester developed a system in 2007 that models computer networks, intrusion detection systems (IDS), and the behavior of the network as an alternative way to simulate cyber-attack scenarios (Sudit et al, 2007). Their system allows the user to construct a simulated test network and then devise, build, and execute various cyber-attack scenarios against this network via a GUI, however, the tool's capability is limited to relatively simple scenarios.

Another method of cyber-attack modeling uses attack graphs (or trees) to represent every possible sequence of actions that can lead to a specific attack goal. Paudel et al (2017) used this methodology to analyze advanced persistent threat (APT) cyber-attack methods against smart-grid wide area monitoring systems (WAMS). Kotenko and Chechulin (2013) used attack graphs to evaluate security and provide impact assessments over various stages for determining countermeasure in near-real time. Commercial software products, such as Amenaza SecurITree, make use of attack trees for security analysis but lack the capability to rapidly simulate offensive attacks against a desired network. The main disadvantage of attack trees, however, is the exponentially increasing computational complexity when modeling large systems with thousands of nodes.

3. MAST overview

The Malicious Activity Simulation Tool (MAST) is designed to mimic malware injection on live networks to simulate malware attacks on the same machines and networks that could be attacked in a real malware attack (Belli, 2016; Lowney, 2015). While it is designed to support training and testing of non-privileged users and administrators, we have used it as the foundational platform to build OCO simulation and testing capability. We are able to use MAST in this way because of the following capabilities:

- Support for execution of scripted scenarios,
- Scripted scenarios can be transported and transferred (i.e., remotely controllable),
- MAST malicious behaviors are modular and extensible by scenario authors (i.e., not hard-coded in MAST),
- Scenario behaviors are dynamic and react to the conditions on the workstation and network.

At the heart of MAST is SimWare, or simulated malware modules. SimWare is software meant to mimic some aspect of a computer attack or intrusion. The definition of what mimic means depends on the simulation and whether it is meant to fool software or users into thinking that an unwanted activity is taking place. Such activity can include random network traffic, pop-up windows, system messages, virus signatures, etc. This activity may be caused by existing system commands and executables, or it may be new executables or scripts loaded from the server or as part of MAST installation on a host computer. In MAST, these executables are indexed and accessed by a unique module name.

MAST utilizes a three-tiered client-server model that was designed to be able to test an organization's users, administrators, and security tools. The MAST architecture, shown in Figure 1, consists of a top-level Scenario Generation Server (SGS), second-tier Scenario Execution Servers (SES), and third-tier MAST client(s).

3.1 Scenario Generation Server (SGS)

The SGS is responsible for many of the command level functions of MAST. It is primarily responsible for generating and disseminating training scenarios, providing a central repository for all available scenarios, controlling connected Scenario Execution Servers, and running independent scenarios on multiple networks simultaneously. The SGS installs as a Java desktop application, and can be run from a local or remote location to control the second-tier SES(s).

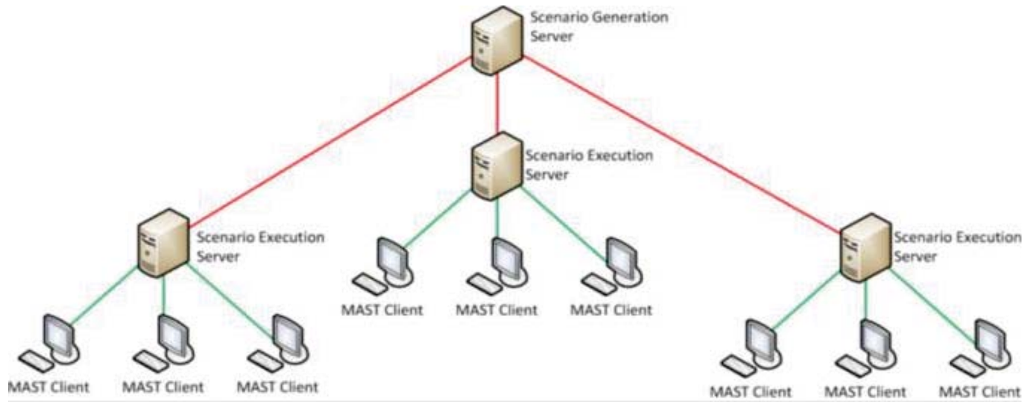

Scenario Generation Server
Scenario Execution Server
Scenario Execution Server
Scenario Execution Server
MAST Client
MAST Client
MAST Client
MAST Client
MAST Client
MAST Client
MAST Client
MAST Client
MAST Client

Figure 1: MAST three-tier client server architecture (Lowney, 2015)

3.2 Scenario Execution Server (SES)

The SES receives control commands from the SGS, and communicates directly with the MAST clients. It manages execution of attack scenarios on the clients, and is locally installed on the operational or target network for training (Lowney, 2015). As the middle manager of MAST, the SES is responsible for a number of intermediate tasks that include distributing SimWare and scenario files to MAST clients, maintaining MAST client and SimWare status, overseeing all scenario activity on MAST clients, handling “kill switch” functionality in case of emergency shutdown, and maintaining a copy of deployed SimWare and scenario files (Diana, 2015).

3.3 Clients

The MAST client uses a Java application that runs on each of the client machines on the MAST test network. The client application is controlled by the SES, and performs a number of client functions: reports the status of the running scenario to the SES, lists all modules that are installed on the host client to the SES, and executes the scenario as directed by the SES**.**

When a MAST Client connects to the SES, the SES immediately spawns an instance of the ClientCommunicatorClass for that socket. This is shown in Figure 2. The ClientCommunicatorObject handles all communication with the specific MAST Client. Once the MAST Client has finished sending and receiving data, its corresponding communicator is destroyed. The MAST Clients do not maintain a persistent connection with the SES. Each MAST Client connects to the server at a set interval and disconnects after sending and receiving data. The first time a MAST Client connects, a ClientObject is created for that MAST Client. This ClientObject is stored in memory until the SE Server is shutdown and stores information about the MAST Client. The ClientObjects are identified by a unique ID generated by the MAST Client.

A basic understanding of how the MAST Client works is necessary if a scenario author wants to write complex scenarios (see “Scenarios” section below). The MAST Client is responsible for running the malicious behaviors and reporting the result of those behaviors back to the SES. The malicious behaviors are contained in Simware modules. Modules are self-contained programs that are invoked by the MAST Client when directed by the SES. The modules are not compiled into the MAST Client so that the scenario author can write new behaviors without needing to change the MAST Client’s code. Modules can be written in any language supported by the operating system. We use the ProcessBuilder library to execute external programs while capturing runtime information such as console output and return codes. This is the mechanism through which the module is able to send data back to the SES.

The module return code is transmitted to the SES in a StatusPacket. There exist two methods by which a module can produce a return code. The first method is initiated when a module exits. The return code is packaged in a StatusPacket and sent to the SE Server. In the other method, the module writes out a special string to its output. This string is “MMReturnCode=<NUM>.” If the MAST Client detects this string in the output it will send a new StatusPacket with that return code. Return codes are used to initiate additional events in scenarios; therefore, these two methods of sending return codes allows the scenario author to initiate additional events after a module ends or while the module is still running. This can be useful for larger modules that execute several behaviors. The module author may want to start a new behavior while a previous behavior is running.

Module output is transmitted to the SES inside OutputPackets. The output is not processed by the SE Server and is retained for logging purposes. However, the module output is used by the MAST Client to create rollback code that will run once the scenario has finished. The MAST Client monitors the output for any line beginning with “ROLLBACK=.” Any text after the equals sign is written out to a batch file which is run after the scenario ends. Using this method, a module author is able to dynamically create a rollback script that brings the workstation back to the state it was in before the scenario ran.

3.4 SimWare modules

SimWare modules contain code for mimicking target malware behavior, called “mimics”. As mentioned above, SimWare can simulate a range of malevolent behavior, such as scanning, pinging, or hijacking of a host. Additionally, a mimic can be identified as a particular type of attack based on its binary signature matching that of some known malware (Lowney, 2015). The MAST framework can accept multiple executable file types as

SimWare modules, such as compiled Java or C/C++ code, Python scripts, Unix shell scripts, and Windows batch files.

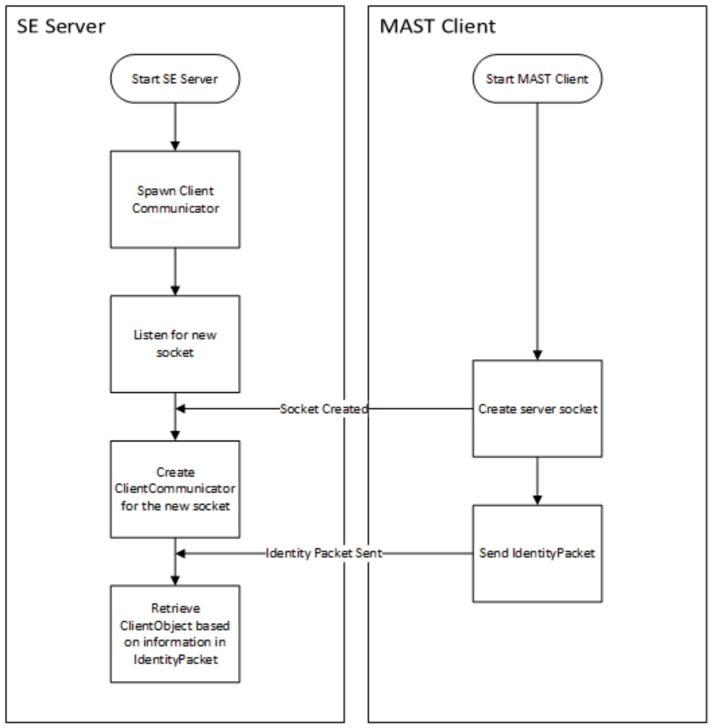


Figure 2: Client connection flow diagram

3.5 Scenarios

MAST Scenario files define SimWare modules with various options and commands needed to execute them. SimWare Modules are controlled by scenarios. A scenario is a set of instructions that direct specific Clients to run specific modules in a controlled way. A scenario can be as simple as directing one client to run one module, or directing multiple clients to run multiple modules. Scenarios are only run by the Scenario Execution Server, and it can only run one scenario at a time.

Scenario files allow the tester to configure different forms of a SimWare module to test different features on the target system. The SGS and SES run these scenario files and compile the results, and in some instances, respond with the appropriate “reply” in accordance with the scenario parameters. Figure 3 shows this relationship between modules and scenarios.

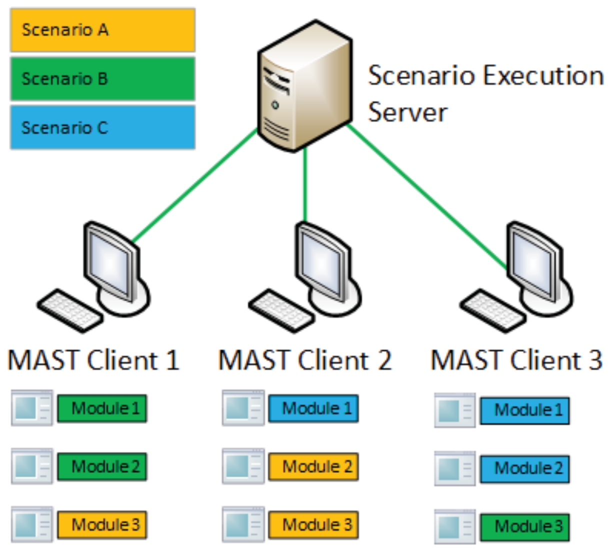


Figure 3: Relationship between scenarios and modules

4. Cyber-attack design

Designing effective cyber-attacks for an offensive cyber operation is critical to replicating real-world cyber operations. Several critical factors and parameters must be considered to properly design and develop the processes and approaches used to test simulated cyber-attack scenarios in a virtualized environment. These include:

- Target network configuration
- Interaction parameters
- Attack rehearsal essentials
- Attack types

4.1 Target network configuration

Creating a cyber-attack simulation that will lead to a high probability of success against a real-world system demands thorough knowledge of the target network. OCO mission success is predicated on knowledge of a network's vulnerabilities in order to exploit them. Detailed knowledge of the network will include enumeration of the network (IP addresses, network topology, domain names); hosts (architecture types, OS variant and version, services running, ports open); security settings (firewalls, IDS/HBSS running with associated detection method, password policy requirements, etc.); and physical security measures (padlocked doors, keycards, hardened facility). This knowledge may enable a cyber operator to bypass target defenses and exploit a system to achieve some intended cyber effect. Knowledge of network vulnerabilities can be gained through active and passive computer scanning, and intelligence gathering through human, signals, and open source means (McClure, et al 2012).

4.2 Interaction parameters

Creating accurate OCO simulations requires interaction parameters that can define the secondary behaviors of the SimWare. These configurable parameters, such as specificity, timing, propagation, skill and stealth, can modify the manner in which the SimWare traverses through the target network to accurately mimic real-world behaviors, i.e., self-propagation vs aided propagation. The interaction parameters are designed to function complimentary, such as skill and stealth, or to be combined to form a new hybrid behavior that affects the SimWare differently than either would independently. For example, a piece of malware that is constructed by a nation-state adversary could be very stealthy, and extremely specific about its target and attack timing; whereas, a lessor adversaries' malware might have very low specificity, timing, and stealth. In this research effort, the ability to identify a target by either IP address or MAC address and specify the timing of that attack were the interaction parameters that were implemented.

4.3 Attack rehearsal essentials

Creating a virtual network environment that can be used effectively to rehearse a cyber-attack will require a number of essential elements. These include the ability to reset and reconfigure the environment rapidly, and also to provide a wide selection of configurable SimWare options to achieve various cyber effects. In addition, the environment must support configurable timelines for activities and their execution, and it must be able to incorporate specific vulnerabilities for a target network or system. Finally, an effective environment must have the ability for automated testing of a previously configured simulation.

4.4 Attack types

Cyber-attacks are often classified according to their objectives, for example, an attack may target data confidentiality (illicit access), integrity (data manipulation), or availability (denial of service). These categories provide a useful breakdown for the development of the different types of SimWare modules we focused on in developing the framework.

5. Implementation

Our test environment was composed of a virtualized network infrastructure, using a Type 1 hypervisor, the virtualized target hosts, and the software tools used for execution of the cyber-attack scenarios. VMware was used as the hypervisor and to deploy the target hosts, which provided a high degree of configurability and virtual resources, a key requirement for accurate modeling of the adversary network. The target hosts were configured on a private network that was created and configured within a test network to allow communication between hosts on their subnet. Each host required its own instance of the MAST client to be run locally.

The network architecture was kept flat to allow better examination of the behaviors of the attack modules, and to avoid additional complexity that could have affected the testing results. The test setup within the VMware environment is shown in Figure 4. The VM hosts are listed on the left-side, within the red box, while the MAST network configuration and running clients are shown on the right-side of Figure 4. The center window allows selection of a cyber-attack scenario from a predefined list. The three new scenarios created in this research are indicated in red, and include: “Attack When Idle,” “Logic Bomb,” and “Targeted Virus Attack.” The scenarios utilized SimWare modules that were developed to implement the interaction parameters of target specificity and timing.

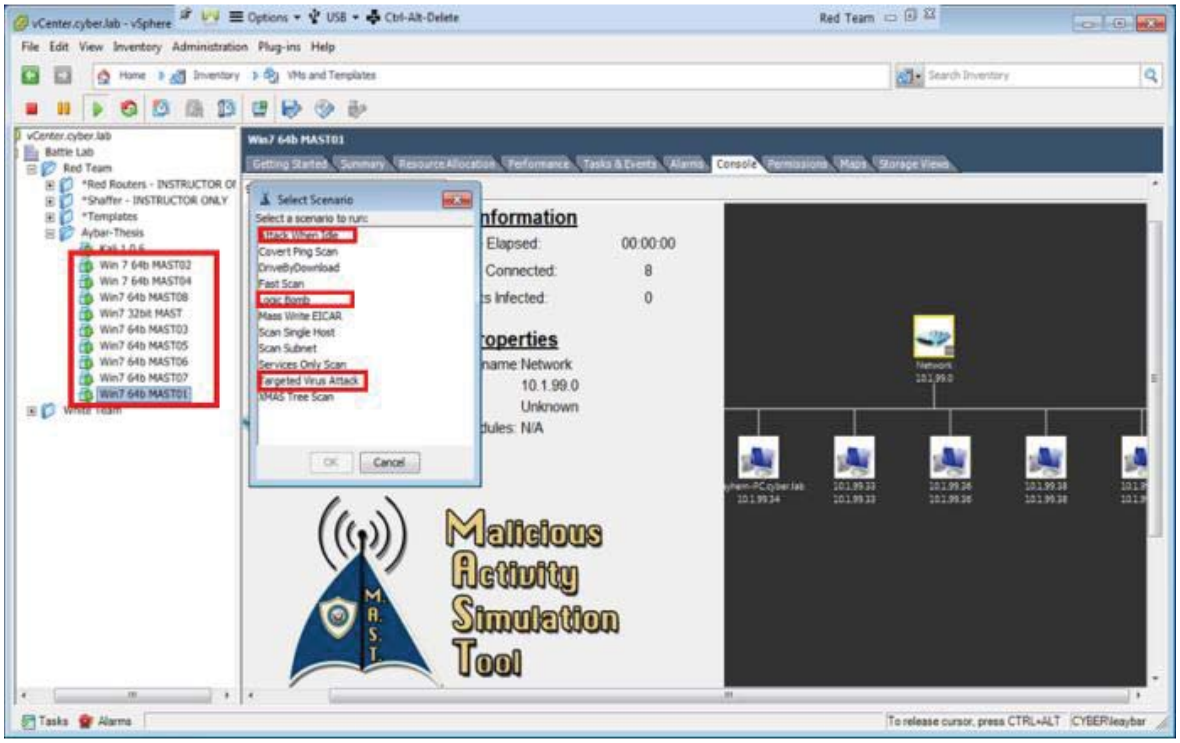


Figure 4: Virtualized test network in MAST

5.1 Scenario file

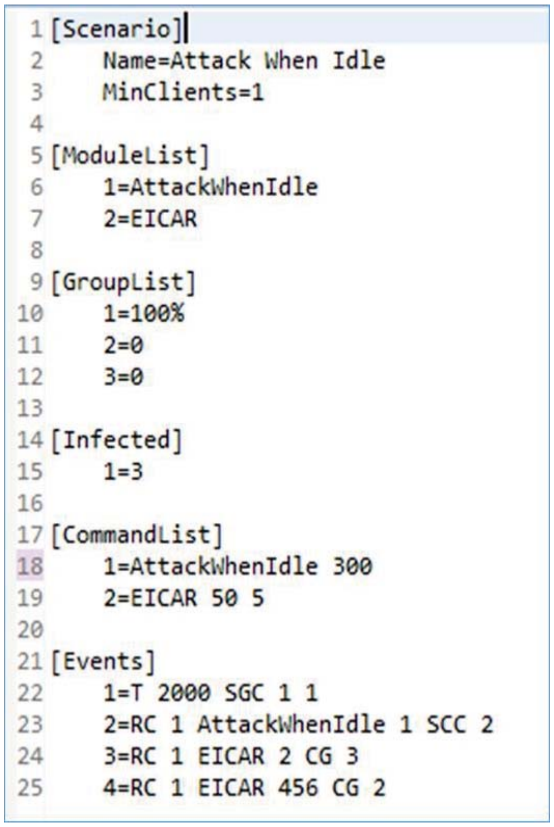
The MAST scenario file consists of key-value pairs that define an attack scenario. The file provides the framework with instructions for how to execute specific SimWare module(s) for the scenario, and how to perform subsequent actions based on return conditions. Each section of the scenario file is delineated by the section keyword surrounded by square brackets. The SES reads-in the scenario file sequentially and parses the commands in the file to direct the execution of commands to the appropriate MAST clients. Each MAST client runs as an application process on the host VMs and maintains an active communication channel to the middle tier execution server to provides status updates and receives command direction.

Figure 5 shows an example scenario file for an “Attack When Idle” scenario. It highlights the modularity of the MAST design by incorporating a range behaviors and modalities in the scenario file. Each individually designed module, whether written in Java or C, or a Unix script file, can be utilized by the MAST framework. The additional arguments needed for either the scenario configuration or executable module are catalogued in the various list sections (ModuleList, GroupList, CommandList) of the scenario file.

5.2 Idle user attack module

In an offensive cyber-attack, the timing of an attack can be just as important as the affect it achieves. In many cases, an attack should be launched during an idle period of activity on the target host, under the assumption that no user is interacting with the computer and thus the attack will be less likely to be detected. With this in mind, the ability to support temporal specificity of an attack is important in the development of realistic cyber scenarios. For this module, we implemented the ability to launch a cyber-attack only after no user has interacted with the target machine for a specified period, to further hide the attacker's activity.

This scenario file contains all the necessary information for MAST to execute an attack scenario. The file is processed as a series of linear actions as shown in the "Events" section. In Figure 5, events 2-4 perform similarly to an 'if-then-else' programming structure where the determination of what gets executed is based on the received return code.



```
[Scenario]
    Name=Attack When Idle
    MinClients=1

[ModuleList]
    1=AttackWhenIdle
    2=EICAR

[GroupList]
    1=100%
    2=0
    3=0

[Infected]
    1=3

[CommandList]
    1=AttackWhenIdle 300
    2=EICAR 50 5

[Events]
    1=T 2000 SGC 1 1
    2=RC 1 AttackWhenIdle 1 SCC 2
    3=RC 1 EICAR 2 CG 3
    4=RC 1 EICAR 456 CG 2
```

Figure 5: "Attack When Idle" example scenario file

In Figure 5, the [CommandList] input parameter in the module represents the minimum number of seconds in the timeout period. For example, if the scenario developer required two hours of user inactivity before executing a SimWare event, he would pass argument '7200' to this input parameter.

To determine user inactivity, the "AttackWhenIdle" module detects user inputs (mouse movements and clicks, or keyboard entries), and tracks the amount of time since the target system last received any of these; this represents system idle time. If the idle time is greater than a user passed timeout value, the module exits a successful return code to the SES. If a user input event is detected before the specified timeout is reached, the timer is reset and the idle period begins anew. The following pseudocode represents this system idle logic:

```
while ( idleTime < timeToWait ) {
        sleepTime = (timeToWait – idleTime) + 1
        sleep( sleepTime );
        idleTime = getIdleTimeOS_SystemCall()
        if (idleTime >= timeToWait) {
                return success_code
        }
}
```

5.3 Timing specific attack module

The second module implemented replicates the ability to launch a SimWare module that will activate when a specific set of conditions occur, or after at a predesignated time period, known commonly as a “logic bomb.” The module provides the ability to manipulate the temporal targeting aspect of the scenario, albeit differently than was done in the “AttackWhenIdle” module. The logic for this module used a loop construct to periodically check the current system date against the passed date-time to execute. If the execute date-time is after the current date-time, a differential time is calculated and is used as the parameter to initiate a sleep function. For efficiency, the module does not proceed until the current date-time falls after the execute date-time, as there is nothing gained by checking more frequently. When the execute date-time falls before the current date-time, the module will return a successful exit code. Whenever the passed argument for the date-time is before the current date-time, the module will immediately return a successful exit code, simulating a logic bomb that would execute immediately once its criteria has been met.

5.4 Target specific attack host module

The final attack module enabled target specificity on the network or link layer. The ability to target an adversary host or subnet by IP address or MAC address is vital for developing offensive scenarios against a large, dispersed network. This module accepts one or more IP addresses, MAC addresses, or computer host names as input parameters. For example, a scenario might have several hosts, with the goal to have the SimWare launch only on the hosts that match the IP addresses passed as arguments.

Since a separate instance of the “TargetSpecificHost” module will run on each host, there can only be one matching IP address, even if two IP addresses were passed to the module. The module logic proceeds linearly, and loops through the input parameters to compare each with the host’s IP address. Java library functions are able to return the IP address for each host and for all its network interfaces (Ethernet, Wi-Fi, virtual). For this module, the input for the [CommandList] of the scenario file in Figure 5 would be similar to the following:

```
[CommandList]
1=TargetSpecificHost 10.1.99.11  192.168.74.1
  or
1=TargetSpecificHost 00-50-56-9C-4A-EF
```

The ability to target hosts on a particular subnet, or hosts behind a Network Address Translation (NAT) server, by being able to discern a specific MAC address is vitally important. This module was able to achieve that behavior by being able to discern different hosts via their MAC address and determine if the target criteria was met based on the passed arguments in the [CommandList] of the scenario file.

6. Conclusions and future work

Our framework enabled us to develop, simulate and test a number of different cyber-attacks on a variety of network configurations. Although this research was able to achieve a number of important milestones toward the creation of an offensive cyber scenario development platform, it is only the first step in this process. Future research will focus on two areas for extending the framework: further development of attack scenario target parameters, and enhancement of the MAST framework to support more robust scenario development.

We will extend the MAST framework to make it a more robust cyber-attack development platform, to include a menu of available SimWare modules that can be selected from a GUI-style menu and auto-loaded into a cyber scenario file shell. Other improvements will add support for common programming constructs to the scenario file, enabling the developer to essentially program or script a cyber-attack scenario. MAST currently allows linear processing of an attack scenario file, with no looping constructs, complex conditionals, or variable assignments within the current framework. These advanced programming features are common in modern scripting languages such as Java Script, Bash shell, and others. Extending the framework to allow advanced scripting constructs will allow the creation of much more complex scenarios. We feel these extensions will greatly increase the effectiveness of MAST as an offensive cyber-attack scenario development and testing platform.

Attack parameters such as skill, efficiency and stealth would add greater realism and nuanced behavior to the developed scenarios. These could be developed and integrated within the MAST framework, and made selectable via user menus. In this way, the parameters could be applied to any of the scenarios developed within the framework. These improvements, if implemented, would greatly increase the effectiveness of MAST as an offensive cyber-attack scenario development platform.

References

Belli, G. (2016) “Extensible simware architecture for flexible training scenarios”, MSCS thesis, Naval Postgraduate School, Monterey, CA.

Diana, B. (2015) "Malicious activity simulation tool (MAST) and trust," MSCS thesis, Naval Postgraduate School, Monterey, CA.

Fanelli, R. and Conti, G. (2012) "A methodology for cyber operations targeting and control of collateral damage in the context of lawful armed conflict", *Proceedings of IEEE 4th international conference on cyber conflict (CYCON)*.

Gellman, B. and Nakashima, E. (2013) “U.S. spy agencies mounted 231 offensive cyber-operations in 2011, documents show”, *The Washington Post*, http://wapo.st/17sEENT?tid=ss_mail&utm_term=.1e048333 c62f.

Grant, T. (2013) "Tools and Technologies for Professional Offensive Cyber Operations", *International Journal of Cyber Warfare and Terrorism (IJCWT)*, Vol. 3 No. 3, pp 49-71.

Hammerstein, J. and May, C. (2010) “The CERT approach to cybersecurity workforce development”, Carnegie Mellon Univ. Software Engineering Institute, Pittsburgh, PA, Tech. Rep. CMU/SEI-2010-TR-45.

Kennedy, D., O'Gorman, J., Kearns, D., and Aharoni, M. (2011) *Metasploit: The Penetration Tester's Guide*, San Francisco: No Starch Press.

Kotenko, I., and Chechulin, A. (2013) “A cyber attack modeling and impact assessment framework”, *Cyber Conflict* (*CyCon 2013*), 5th International Conference on Cyber Conflict, pp 1-24, IEEE.

Lowney, E. (2015) "Network communications protocol for the malicious activity simulation tool (MAST)," MSCS thesis, Naval Postgraduate School, Monterey, CA.

Mayes, J. (2014) “Modeling large-scale networks using virtual machines and physical appliances”, Carnegie Mellon Univ. Software Eng. Inst., Pittsburgh, PA, Tech. Rep. DM-0000921.

McClure, S., Scambray, J. and Kurtz, G. (2012) *Hacking Exposed 7, Network Security Secrets and Solutions*. New York: McGraw Hill, 2012.

Paudel, S., Smith, P., and Zseby, T. (2017) "Attack models for advanced persistent threats in smart grid wide area monitoring", *Proceedings of the 2nd Workshop on Cyber-Physical Security and Resilience in Smart Grids*.

Rattray, G. and Healey, J. (2010) "Categorizing and understanding offensive cyber capabilities and their use", *Proceedings of a Workshop on Deterring Cyber Attacks: Informing Strategies and Developing Options for US Policy*.

Sheldon, J. (2012) "State of the art: Attackers and targets in cyberspace", *Journal of Military and Strategic Studies*, Vol. 14, No.2.

Sudit, M., Kistner, M., Kistner, J. and Costantini, K. (2007) “Cyber attack modeling and simulation for network security analysis”, *IEEE Winter Simulation Conference*, Washington DC.