VINE: A Cyber Emulation Environment for MTD Experimentation

Thomas C. Eskridge, Marco Carvalho, Evan Stoner,
Troy Toggweiler, and Adrian Granados
Harris Institute for Assured Information
Florida Institute of Technology
Melbourne, FL
teskridge,mcarvalho@fit.edu

ABSTRACT

Dynamic and moving target defenses are generally characterized by their ability to modify their own state, or the state of the protected target. As such, the evolution of these kinds of defenses require specialized experiments that can capture their behavior and effectiveness through time, as well as their broader impacts in the network. While specialized experiments can be constructed to evaluate specific defenses, there is a need for a general approach that will facilitate such tasks. In this work we introduce VINE, a high-fidelity cyber experimentation environment designed for the study and evaluation of dynamic and moving target defenses.

VINE provides a common infrastructure supporting the construction, deployment, execution, and monitoring of complex mission-driven network scenarios that are fully instrumented. The tool was designed to be scalable, extensible, and highly configurable to enable the study of cyber defense strategies under dynamic background traffic and attack conditions, making VINE well-suited for the study of adaptive and moving target defenses. In this paper we introduce the VINE approach, the VINE architecture for MTD experimentation, and provide an illustrative example of the framework in action.

Categories and Subject Descriptors

C.2 [**COMPUTER-COMMUNICATION NETWORKS**]: Network Operations—*Network monitoring; Network management*

General Terms

Experimentation

Keywords

Moving Target Defense Experimentation; Network Emulation; Network Creation; Network Monitoring

1. INTRODUCTION

Virtual Infrastructure for Network Emulation (VINE) provides a common infrastructure for the construction, deployment, execution, and monitoring of complex mission-driven network scenarios. The tool was designed to provide a high-fidelity and highly configurable environment for the study of cyber defense strategies, tools, and techniques. As such, VINE is largely designed as an experimentation platform, with support for automatic scenario generation, deep instrumentation, and hypothesis testing through semi-automated analysis of experimental results.

VINE was originally developed for the high-fidelity emulation of tactical communication systems. The later generations of VINE have extended the original emulators to include broader aspects of the system infrastructure, application support, user models, and mission descriptions. These capabilities were built on top of the emulation infrastructure, which enables VINE to support a wide range of emulation capabilities ranging from user models, down to the individual layers of the communications stack of individual nodes.

VINE relies on both host and network virtualization for the experimentation environment. The framework also supports the integration of physical systems and specialized hardware in the loop. The framework provides the bridge across the virtualized and the physical components and, within the constraints of the integrated devices, provides a unified view of the experimentation environment, including both virtualized and physical nodes.

VINE is actually a collection of tools. As illustrated in Figure 1, there are a number of components that compose the VINE experimentation environment. The numbers to the upper left of each component in the figure indicate in which section or sections of this paper that component is discussed. VINE is an active research project within the Harris Institute for Assured Information. It is used as a testing environment for other cybersecurity projects [1, 3] as well as for a safe space to undertake cyber attacks and defenses for classroom assignments. We are also continually extending VINE to assist and automate more of the experimentation process. As detailed in [8], VINE provides a framework on which a general theory of automated experimentation can be implemented.

2. EXPERIMENTATION PROCESS

Cyber experimentation involves a cyclic process (shown in Figure 2) that includes environment construction; exper-

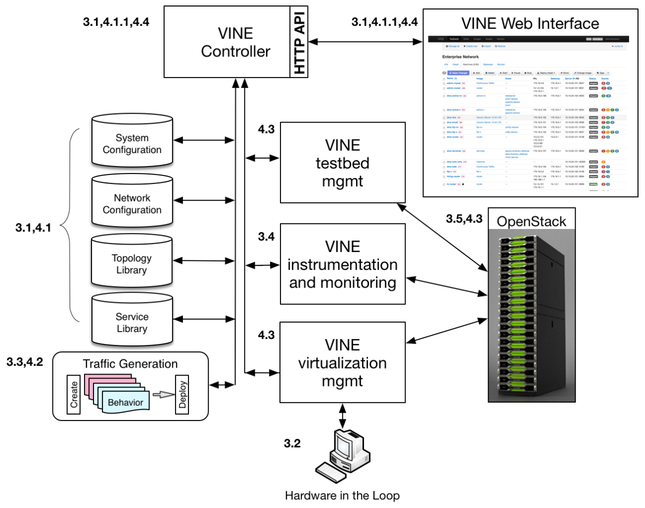

Figure 1: VINE experimentation environment.

iment design, creation, execution, and control; monitoring and data gathering; and data analysis. Here, we will briefly identify the steps involved in cyber experimentation. Further details of the general methodology and process can be found in [8].

Experiment Design The design of the experiment involves creating hosts and assigning their roles, including operating system and installed software, benign and malicious traffic generation, and network and moving target defenses. It also includes the identification of indicators of experiment performance, as well as the selection and assignment of experiment variables to be used over several iterations of the experiment. These variables can represent attack rates, defense movements, background traffic levels, or mission requirements.

Experiment Creation. Once the design is complete, the experiment can be instantiated with a variety of parameters for defenses and attacks. For any experiment, the creation through analysis phases are likely to be run numerous times to collect average-case behavior, rather than one-off behaviors.

Experiment Execution and Control. VINE provides the necessary functionality to run the experiment, modify testbed components when necessary, and ensure proper handling of experimental data that can be analyzed at the end of the experiment. Depending on the experimental design, new processes and network components can be set up to initialize and join the network at particular times during the course of the experiment.

Experiment Monitoring. VINE wraps functionality that enables users to monitor the performance of the machines emulated in the testbed, as well as the hardware the testbed is running on. This can help to identify significant events within the experiment itself, as well as events from the virtualization platform used to run the testbed.

Experiment Analysis. The data collected along the experiment backplane connecting all testbed virtual machines is made available for both automated and manual analysis. Depending on the experiment, the logs can be programmatically evaluated, and changes to the network, services, or experiment parameters can be made based on the evaluation results.

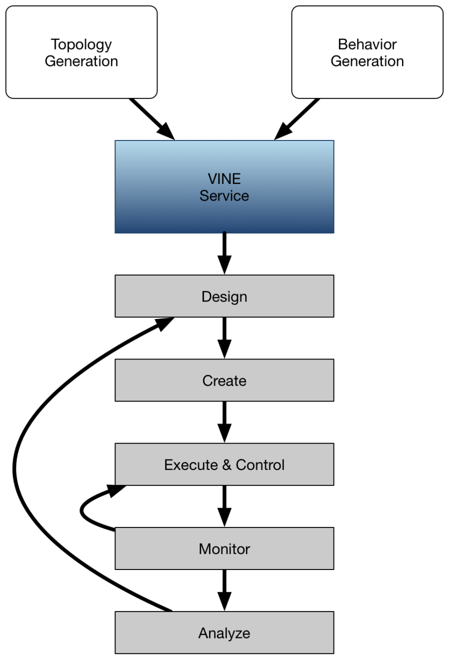

Figure 2: VINE experimentation process.

3. FUNCTIONALITY

A number of features in VINE provide the functionality to enable cyber experimentation. While VINE at its core provides the means to emulate network topologies with a high level of fidelity, additional tools provide the means for connecting physical systems to emulated ones, generating realistic background and attack traffic, instrumenting experiments, and duplicating topologies across installations and platforms.

3.1 Building Network Topologies

VINE abstracts underlying virtualization providers to realize complex network topologies. Through a variety of interfaces, users can build emulated networks whose hosts are heterogeneous in terms of (virtualized) hardware and operating systems. Thus, surrogate environments can be built in VINE that mirror a physical counterpart in size, layout, and configuration. In addition, wireless network emulators such as EMANE [5] can be installed on virtual hosts to create scenarios consisting of both wireless and wired connections.

Because VINE scenarios are powered by cloud computing platforms such as OpenStack [9], their scale is only limited by the compute resources available in the targeted cloud. The overcommitting of physical compute resources by those cloud providers can turn commodity hardware into a backend for medium-to-large-scale network experiments in VINE.

3.2 Hardware in the Loop

Physical devices can be plugged into VINE scenarios using normal networking operations. Each isolated testbed in VINE exposes a routable entry point through which external physical subnets can connect, just as one would connect

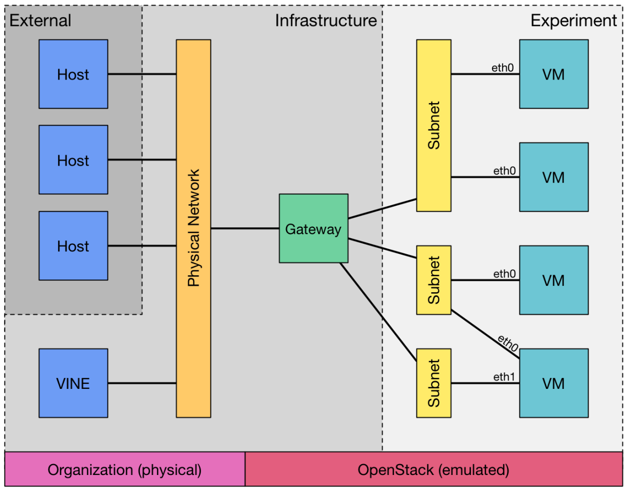

Figure 3: Integrating external hardware into an emulated testbed.

to a physical subnet (see Figure 3). This feature enables users who have existing labs or specialized hardware such as IP hopping appliances to scale their experiments using virtualized hosts. Similarly, the entry point can also serve as a gateway for the virtualized hosts so that they can "reach out" and interact with the existing physical devices.

3.3 Background and Attack Traffic Behavior Generation

Additional tools (see Section 4.2) enable the user to design both benign and malicious traffic generation models. The agents that execute these models understand mission goals and as such are capable of adapting when failure is encountered, just as a human would. By chaining many of these behaviors together, a complex and dynamic narrative can be built which closely reflects the operations of a real-world organization. This includes attackers who possess extensive and advanced toolchains of exploits, and operators who deploy and manage moving target techniques.

The behavior generation functionality is deployed on VINE testbeds by associating a behavior with a testbed virtual machine as a *role*. On VM startup, the roles are executed and the virtual machine takes on the persona of the behaviors running on it.

3.4 Instrumentation and Monitoring

VINE can instrument experiments in two places: The first is at the host, where traditional packet capture can be performed. Packet capture enables a low-level analysis of the state of a host in an experiment and can reveal intricacies of the executions of both attacks and defenses. In addition, monitoring agents installed on the hosts can collect and report information about the host such as CPU and memory consumption and a list of open files.

The second instrumentation location is at the traffic generation agents, which log information about successes and failures of their communication behaviors in order to provide the ground truth of the experiment. Compared to host monitoring data, this information conveys a higher level of information to the user that speaks to mission goals and computing tasks rather than protocols and payloads. Dashboards and other visualizations are used to display behavior data, such as number of successes or failures versus time (see Section 5 for an example of these dashboards). These visualizations support drilling down into the data, for example to show the number of communication failures generated by an FTP activity.

The multiple resolutions of instrumentation that VINE supports enable the user to verify the consistency of multiple executions of the same trial, and produce quantitative results on the effects of a defense across varying trials.

3.5 Portability

Reproducibility is an important component of any scientific endeavor. VINE supports exporting and importing testbed specifications to enable external entities to reinstantiate network topology and configurations and rerun experiments to confirm experimental results. This functionality also allows two organizations to build identical topologies for parallel experimentation. Through the use of IT automation tools, these environments can also be configured identically from a software perspective, including enterprise services like email and FTP servers and experiment tools like wireless emulators and network defenses.

In addition, VINE can export testbed topologies to a format that Emulab-based testbeds such as DeterLab [4] can understand. The ability to instantiate scenarios on multiple providers is a crucial part of experimentation as it ensures the validity of an experiment isn't tied to a specific environment.

4. DESIGN

VINE is implemented as a collection of separate, independent components that include: topology generation, traffic behavior generation, operator interface, and underlying infrastructure. Each of these components can be used independently of the others and, thus, can be altered or replaced by other functionally equivalent implementations to perform the experimental task.

4.1 Topology Generation

VINE currently supports three methods for topology construction:

Manual, where the user interacts with VINE's web-based UI to modify hosts and their attributes one-by-one.

Programmatic, where external tools interact with VINE's HTTP API to build testbeds. The HTTP API also supports the all-at-once instantiation of entire testbeds by importing a JSON description of the testbed.

Tool-based, where the user interacts with a tool called Genesis [2] which is capable of generating topologies that mirror those of actual geographical regions based on population density and distribution of services (banks, hospitals, schools, etc.). By selecting the region and service types, Genesis explores public sources to determine a probably topology.

4.2 Traffic Behavior Generation

VINE uses software agents to provide its background traffic and attack generation capabilities. The modularization of traffic generation with agents allows the background model to be changed. For example, VINE currently supports both service distribution profiles (such as those generated by Genesis) and behavior tree–based behaviors [6]. These models provide suitable emulation of both benign and malicious users, who use real virtualized services (e.g. HTTP, FTP, and email) and real attack libraries (e.g. Metasploit [7]).

Behavior tree–based models are especially relevant to MTD experimentation, where the attacker must be capable of adapting to changes in the defender's security posture. A benign example of adaptation is shown in Figure 4, where a "Transfer File" behavior can be accomplished in several different ways. In the face of failure (e.g. FTP server is down), the agent will adapt and try alternative methods (i.e. emailing). Likewise, an attacker may first attempt to exploit one vulnerability, but in the case of failure, try another. The interplay between these two enables measuring the mission impact of an attack. If the mission requires the transfer of a file and the attacker does not remove all three options for accomplishing the transfer, then the mission will still succeed, even though the attacker may have succeeded in compromising one or more possible methods.

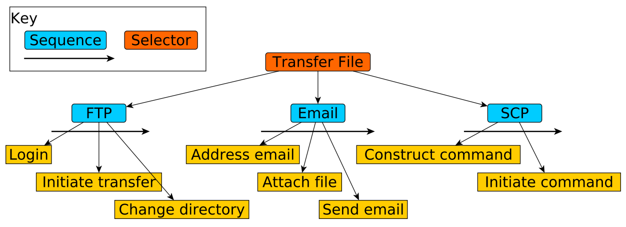

Figure 4: Hierarchical structure of behavior tree–based traffic generation agents.

4.3 Infrastructure

VINE is designed to facilitate cyber experimentation while taking advantage of existing virtualization capabilities. VINE has a clear separation between the abstract specification of the set of tasks needed to define, construct, instantiate, control, and monitor experimental testbeds, and the underlying implementation needed to realize those tasks. Defining these tasks at an abstract level has allowed the development of pluggable backends to interface with the desired or available virtualization software. Because of this, the virtualization environment can be replaced to support experiments at different scales and with different requirements. The actual infrastructure used can be a single machine, a local collection of hardware, or cloud-based installations using OpenStack.

4.4 Interface

VINE has two interfaces that are used to construct, execute, and manage cyber experiments. A web-based interface allows operators to manually create, modify, and monitor experiment testbeds, and to monitor the status of running experiments. The web interface provides functionality for lifecycle management of experiments as well, enabling creating, starting, stopping, and deleting experiment testbeds.

VINE also has a JSON-based HTTP interface that exposes all of the VINE functionality to programmatic operation. This functionality has been particularly useful for automated generation and control of experiments, where the programmatic analysis of experimental results is used to modify the parameters or initial conditions of the experiment testbed, and a new experiment is run.

5. USE: A CASE STUDY

As an example of how VINE might be used in a realistic MTD experiment, consider the construction of an analog environment that mirrors a university research lab in order to study the AppOSDiversity defense. The AppOSDiversity defense periodically changes the application used to implement a service, preventing reliable fingerprinting of the service by the attacker. The example environment consists of four subnets—faculty, labs, servers, and external—with 20 hosts each.

5.1 Topology Construction

To build this topology, we first create a small slice of it manually through the web interface, e.g. four subnets with four hosts each. In addition to hardware and operating systems, the user can assign the proper roles and software configurations to the hosts in this miniature network, including benign and malicious traffic generation and defenses. This is a good time for the user to test and tweak configurations because the systems can be studied without the complexity introduced by scale.

This small topology can then be scaled up by duplicating the machine definitions and changing unique properties, such as IP addresses. The scaling process can be carried out manually (i.e. export the testbed to a text description then copy and paste) or by a tool that can replicate elements of the topology via the API.

5.2 Experiment Execution

The hosts in the testbed can be powered off and on or rolled back to a snapshot at any time, all together or individually. Benign and malicious traffic generators and other experiment software can be controlled likewise. Control is not limited to on/off switches—software agents that are a part of the experiment can support live configuration changes, the effects of which can be observed immediately in dashboards (see Section 5.3). In our moving target experiment, such changes might include changing the dwell frequency, modifying the attack being launched, or altering some attribute of the benign traffic. These tweaks can be performed by a human or another agent, both of which are capable of acting on feedback from the experiment to make their decision.

With appropriate virtualization resources, multiple experiments can be run in parallel. VINE ensures isolation between testbeds so that multiple instances of the same scenario—either identical or slightly varied—can be run at the same time without affecting each other. By parallelizing experimentation, the time from experiment design to results is greatly reduced.

5.3 Data Collection

In an experiment that studies a moving target defense, a variety of data is of interest to the user. Quality of service, service availability, or mission success/failure are all of interest. Figure 5 shows service availability for a user interacting with an FTP server in a control trial where no defense is in place and an attack is occurring at a regular interval. In Figure 6, a moving target defense was added which improves availability.

Figure 7 shows an experiment-specific dashboard which graphs information about the multi-agent system generating background traffic. Dashboards like this one are built using a graphical editor that acts on log messages aggre-

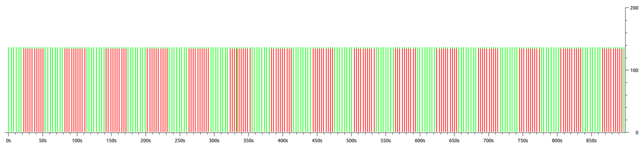

Figure 5: Successful (green) and failed (red) FTP connection attempts by a benign organizational user *without* a defense in place. Failures correspond directly with the interval-based attack.

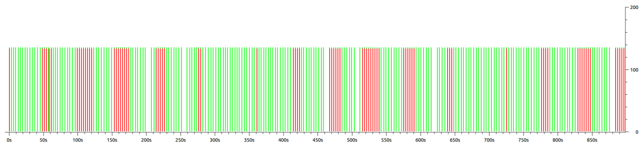

Figure 6: Successful (green) and failed (red) FTP connection attempts by a benign organizational user *with* a moving target defense in place. Although the attack is being generated at the same interval as Figure 5, the defense clearly improves quality of service by thwarting a portion of the attacks.

gated across the experiment. Both real-time and historical views are supported.

6. CONCLUSION

Dynamic and moving target defenses are a promising approach to addressing the need for increased security on computer networks. However, these defenses are significantly different from static defenses and further experimentation is necessary in order to determine their applicability and effectiveness.

VINE provides cybersecurity researchers with a set of tools that accelerate the rate at which they can repeatedly perform high-fidelity experiments at scale. Several interfaces enable the construction of topologies in different ways, each suiting a different type of experiment. Background traffic generators enable the rapid deployment of realistic network conditions to the experiment, including both benign users and attackers. The instrumentation available at several different layers provides researchers with a means to verify the reproducibility of their experiment.

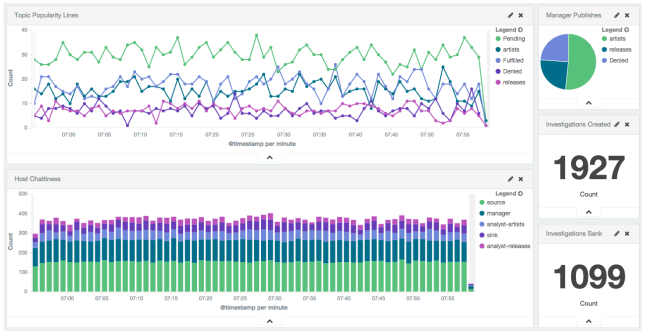

Figure 7: A dashboard depicting information about experiment hosts and mission-specific metrics.

For moving target defenses to be proven useful, they must be tested in a number of configurations, and in a number of operational environments. Network emulation environments such as VINE give researchers a key tool in developing the experimental results necessary to encourage and inform adoption of moving target defenses.

7. REFERENCES

[1] M. Carvalho, T. C. Eskridge, L. Bunch, A. Dalton, R. Hoffman, J. M. Bradshaw, P. J. Feltovich, D. Kidwell, and T. Shanklin. Mtc2: A command and control framework for moving target defense and cyber resilience. In *Resilient Control Systems (ISRCS), 2013 6th International Symposium on*, pages 175–180, 2013.

[2] M. Carvalho and M. Marcon. Genesis. Technical Report HIAI-TR-15-3-1, Florida Institute of Technology, 2015.

[3] M. M. Carvalho, J. M. Bradshaw, L. Bunch, T. C. Eskridge, P. J. Feltovich, R. R. Hoffman, and D. Kidwell. Command and control requirements for moving-target defense. *IEEE Intelligent Systems*, 27(3):79–85, 2012.

[4] J. Mirkovic, T. V. Benzel, T. Faber, R. Braden, J. T. Wroclawski, M. D. Rey, and S. Schwab. The DETER Project: Advancing the science of cyber security experimentation and test. pages 1–7, 2010.

[5] Naval Research Lab. Extendable mobile ad-hoc network emulator (EMANE), 2015.

[6] P. Ogren. *Increasing Modularity of UAV Control Systems using Computer Game Behavior Trees*. American Institute of Aeronautics and Astronautics, 2015/06/15 2012.

[7] Rapid7. Penetration testing software | metasploit, 2015.

[8] E. L. Stoner. A foundation for cyber experimentation. Master's thesis, Computer Science, 2015.

[9] The OpenStack Foundation. OpenStack open source cloud computing software, 2015.