DEVCOM DAC-TR-2021-010
February 2021

VulnerVAN: Automating the Generation of Vulnerable Network Scenarios for Cybersecurity Events and Training Exercises

by Sridhar Venkatesan, Jason A. Youzwak, Cho-Yu J. Chiang, Shridatt Sugrim, Alexander Poylisher, Matthew Witkowski, Gary Walther, Michelle Wolberg, Ritu Chadha, E. Allison Newcomb, Blaine Hoffman, and Norbou Buchler

DISCLAIMER

The findings in this report are not to be construed as an official Department of the Army position unless so specified by other official documentation.

WARNING

Information and data contained in this document are based on the input available at the time of preparation.

TRADE NAMES

The use of trade names in this report does not constitute an official endorsement or approval of the use of such commercial hardware or software. The report may not be cited for purposes of advertisement.

DEVCOM DAC-TR-2021-010
February 2021

VulnerVAN: Automating the Generation of Vulnerable Network Scenarios for Cybersecurity Events and Training Exercises

by Sridhar Venkatesan, Jason A. Youzwak, Cho-Yu J. Chiang, Shridatt Sugrim, Alexander Poylisher, Matthew Witkowski, Gary Walther, Michelle Wolberg, and Ritu Chadha
Perspecta Labs Inc., Basking Ridge, NJ

E. Allison Newcomb
DEVCOM Army Research Laboratory

Blaine Hoffman and Norbou Buchler
DEVCOM Data & Analysis Center

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED *(From - To)*
February 2021	Technical Report	01/01/2019–01/01/2021

4. TITLE AND SUBTITLE
VulnerVAN: Automating the Generation of Vulnerable Network Scenarios for Cybersecurity Events and Training Exercises

5a. CONTRACT NUMBER
W911NF-13-2-0045

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Sridhar Venkatesan, Jason A. Youzwak, Cho-Yu J. Chiang, Shridatt Sugrim, Alexander Poylisher, Matthew Witkowski, Gary Walther, Michelle Wolberg, Ritu Chadha, E. Allison Newcomb, Blaine Hoffman, and Norbou Buchler

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Director
DEVCOM Data & Analysis Center
Human Systems Integration Division (FCDD-DAH-N)
6896 Mauchly Street
Aberdeen Proving Ground, MD 21005

8. PERFORMING ORGANIZATION REPORT NUMBER
DEVCOM DAC-TR-2021-010

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
In this report, we present VulnerVAN, a toolset for automating the generation of vulnerable network scenarios to support cybersecurity training and exercise. Cyber training is becoming increasingly important. People use cyber ranges for training Cyber Protection Teams. Currently, it takes months to plan training exercises, and the process is largely manual. This is because it is hard to plan complex scenarios with multistage attacks and is both labor-intensive and time-consuming. As a result, we designed and developed VulnerVAN. We describe the architecture of VulnerVAN and how the design of the components enables automated generation of vulnerable network scenarios. We also provide examples and use cases to illustrate the functionalities of VulnerVAN.

15. SUBJECT TERMS
Vulnerable Network Generator, CyberVAN, Cyber Training, Cyber Exercise, Cyber Test Range, Scenario-based Experimentation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Norbou Buchler
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UU	35	19b. TELEPHONE NUMBER *(include area code)* (410) 278-9403

Table of Contents

List of Figures

Acknowledgements

This research was sponsored by the Office of the Under Secretary of Defense for Research & Engineering Office of the Under Secretary of Defense for Research & Engineering Cyber Technologies Program. Additional support was provided by the U.S. Army Combat Capabilities Development Command Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (DEVCOM Army Research Laboratory Cybersecurity CRA) and Applied Research & Experimentation Partner program. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of ARL or the U.S. government. The U.S. government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation herein.

1. INTRODUCTION AND MOTIVATION

In recent years, the number of cybersecurity incidents has grown rapidly. Organizations suffer financial loss, data breach, and reputation damage from such incidents. Many cybersecurity defense technologies and tools have been developed to help organizations improve their cybersecurity defense. However, cyber defenders of organizations need to learn how to utilize these tools and technologies, which are often complicated and require defenders to take different actions for different situations. While the tools themselves may provide capabilities and techniques that afford better understanding, analysis, and/or mitigation capabilities, the human operators still need to learn how they are used, what tasks and missions they can support, and how they may integrate with existing platforms and tools used. Defenders must continuously learn new technologies and best practices to better protect their organizations from cyber attack.

In response to the continuously evolving cybersecurity landscape, organizations have begun emphasizing the importance of cyber training and cyber exercises to prepare their cyber protection teams (CPTs) to respond to the latest cyber incidents in a timely manner. Many organizations conduct cyber exercises on a regular basis to evaluate the preparedness of their CPTs and to train them against the latest trends of attack vectors. For example, the U.S. Department of Homeland Security (n.d.) conducts Cyber Storm, a biennial exercise series designed to strengthen cyber preparedness in the public and private sectors. As another example, the U.S. National Guard holds annual CyberShield exercises to test the readiness of CPTs on cyber protection, network defense, and forensic analysis against cyber attacks, hackers, or other malicious actors (McCulloch, 2017). Likewise, private companies have begun coordinating efforts to evaluate the readiness of their incident response teams by conducting regular cyber exercises. For instance, in the wake of multiple data breaches, financial institutions conduct Quantum Dawn, a series of cybersecurity exercises to improve coordination with key industry and government partners to maintain equity market operations in the event of a systemic cyber attack (Securities Industry and Financial Markets Association, 2017). These exercises and events require large amounts of planning and resources (logistical, physical, and manpower) to stand up networks, generate realistic activity, conduct attacks and intrusions, and evaluate performance or preparedness. Often the exercise occurs within cyber ranges.

Organizations conduct cybersecurity training and cybersecurity exercises in cyber ranges, which vary in complexity and objectives. Cyber ranges typically support injections of a broad spectrum of cyber-attack-related events. Some events are for training CPTs while others are for evaluating user, defender, process, and technology (Damodaran & Couretas, 2015). A typical cyber range can be used to set up a vulnerable network that captures the characteristics of a mission-critical network to enable cyber attack against which the performance of CPTs is evaluated. Cybersecurity training and exercises vary by (1) the characteristics of the network (e.g., Cyber Storm 2016 focused on a network setting that is typical in critical manufacturing and transportation sectors), or (2) the characteristics of cyber attack (e.g., emulating attack sequences

used in most recent cyber incidents). In these events, a network scenario supporting the objectives of the training or exercises was designed, developed, and tested by cybersecurity experts. The process typically takes weeks or months from planning to completion. As a result, building an environment for supporting cybersecurity training and exercise is both labor-intensive and time-consuming. Often, these ranges are designed and created for the training or exercise and then dismantled upon completion, requiring building again, even if elements are reused for future events.

In general, cyber practitioners construct cyber training or exercise environments in two ways: (1) setting-up machines in a realistic network environment or (2) spinning up network scenarios in a testbed enabled by simulation and/or emulation. Although approach (2) provides the highest fidelity, its major shortcomings include high administration cost, dedicated hardware use, lack of flexibility to support different exercises, and complexity to restore the entire environment to a preset state. In terms of (2), people may build cybersecurity scenarios on top of a distributed network testbed such as PlanetLab (Chun et al., 2003) or use testbeds specifically built to support cybersecurity experimentation such as DETER (Mirkovic et al., 2010) and CyberVAN (Chadha et al., 2016). The latter supports running real applications in target virtual machines (VMs) over simulated/emulated networks. Regardless of taking approach (1) or (2), the task of preparing a vulnerable scenario in which specific attack sequence(s) can be executed needs to be performed manually and validated extensively (Davis & Magrath, 2013). As a result, the process of provisioning cybersecurity scenarios for training or exercise purposes typically requires a considerable amount of time and resources.

In this report, we present *VulnerVAN*—**Vulner**able cybersecurity scenario generator for Cyber**VAN** testbeds—a toolset that automates the generation and provisioning of a vulnerable network scenario based on a user-specified attack sequence for a given network configuration. The tool is designed to (1) determine the set of all possible attack paths that could realize an attack sequence in a network, (2) generate the provisioning and configuration of the VMs that are involved in the attack sequence, and (3) generate an "attack blueprint" that can guide an opposing force or a Red Team to execute the attack sequence.

We have developed a working prototype of VulnerVAN. Although we built network and VM provisioning functionalities specifically for CyberVAN, the concepts and automated solution-seeking approach of VulnerVAN can be applied to the development of similar tools for other cyber ranges. VulnerVAN is particularly useful for supporting cybersecurity training and exercises. In terms of cybersecurity training, VulnerVAN allows generation of a set of candidate scenarios based on user input, and it generates an "attack blueprint" for executing attack steps in the generated scenario, either for real-time defender awareness training or for postmortem cyber forensic analysis. In terms of cybersecurity exercise, VulnerVAN provides ground truth to exercise organizers to monitor the progress of participants in the exercise.

The following is a summary of our contributions:

- Created a logic-based cybersecurity attack-specification language that defines conditions under which attack steps can be executed and conditions after attack steps have been executed.
- Created a mechanism that identifies a path through the network to realize an attack sequence. The mechanism identifies a subset of machines that satisfy the necessary conditions to realize each attack step in the attack sequence.
- Created a capability to generate multiple instantiations of an attack sequence to allow researchers/practitioners to observe and study the effects of different instantiations of the attack sequence.
- Designed a mechanism to determine the machine configurations required to allow an attack sequence to play out.
- Created an attack orchestrator that chronologically orders the attack actions and outputs an attack blueprint to assist a Red Team.

The remainder of the report is organized as follows. In Section 2, we provide background for the research and discuss related work. In Section 3, we describe the architecture of VulnerVAN. In Sections 4 and 5, we describe a couple of use cases in detail to illustrate the capabilities of VulnerVAN. In Section 6, we discuss possible future work to further enhance VulnerVAN. We conclude the report in Section 7.

2. BACKGROUND AND RELATED WORK

Cyber testbeds have been designed to enable (1) cyber training, (2) cybersecurity research and development, (3) experimentation, and (4) testing. A survey of existing testbeds is provided in Davis & Magrath (2013). However, existing mechanisms for creating attack scenarios in a cyber testbed are either completely manual or limited in the number of attacks that can be executed (e.g., CyRIS [Pham et al., 2016]), or require weeks of effort from testbed staff to create them. Although SecGen (Schreuders et al., 2017) can be used to create vulnerable machines based on known vulnerabilities and common misconfigurations, they were built primarily for supporting Capture-the-Flag exercises. Such tools neither provide a scalable mechanism to model complex networks and defenses, nor do they implement attack sequences such as advanced persistent threat (APT)-style data exfiltration.

CyberVAN is the testbed environment, and VulnerVAN generates scenarios that can be used within it. As shown in Figure 1, CyberVAN includes a variety of technologies created to support running cybersecurity experiments in a flexible and versatile testbed environment. CyberVAN testbeds use both host virtualization and network virtualization technologies to enable creation of high-fidelity experimentation scenarios and flexible use of testbed resources. Scenarios may include VMs that run different types and versions of operating systems (OSs), currently including those of Windows, Linux, and Android. VMs in a scenario are connected through a virtual substrate enabled by a simulated network. The primary network simulator being used is ns3, but other network simulators such as OPNET and QualNET have been used in the past. Using network simulators that support realistic packet forwarding and control also means mobile networks running wireless protocols are supported in CyberVAN through use of high-fidelity, thoroughly validated simulation models. Users can create their own experimentation scenarios, deploy scenarios on CyberVAN testbeds, and save their scenarios before withdrawing them from the testbed for later use. As a result, CyberVAN testbeds provide a flexible environment for supporting various types of cybersecurity training and exercise activities.

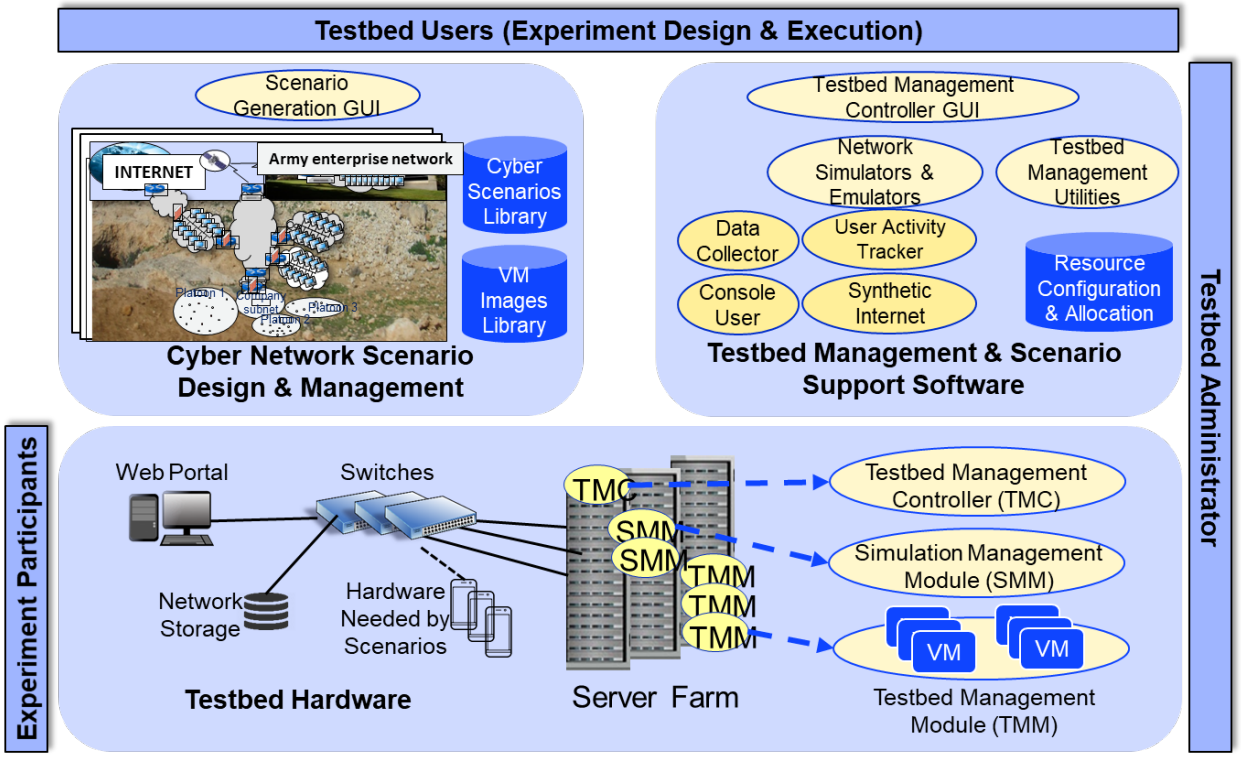

Figure 1 High-level functional architecture of CyberVAN testbed

CyberVAN scenarios are specified in an Extensible Markup Language (XML) scenario-definition schema. Users may use the CyberVAN scenario-generation graphical user interface (GUI) to specify scenarios that include hundreds or more nodes in a complex network consisting of multiple types of wired and wireless subnets, such as Ethernet, Wi-Fi, and satellite networks. By using network simulation that affords high fidelity and control over packets and transmission, CyberVAN supports scenarios that include both enterprise network settings and mobile wireless network settings. CyberVAN also includes a utility called ConsoleUser, which is used to simulate human user activities on machines. This capability is useful in certain training and exercise scenarios because in many cases human actions expose vulnerabilities of cybersecurity.

Attack sequences are composed of attack steps. Existing attack representation languages such as IODEF (Cain & Jevans, 2010) and STIX (Barnum, 2012) represent attacks in a structured format for automated threat exchange. They are primarily designed and used to share indicators of attack and indicators of compromise between organizations in a timely manner. However, such languages lack the expressive power to represent the conditions under which a machine or a network is vulnerable to cyber attacks. Attack graphs, on the other hand, can be used to represent a cyber attack using pre- and post-conditions indicated by Boolean predicates (Wang et al., 2007). A path through an attack graph is an attack sequence in which each attack step is enabled only if all of its pre-conditions are satisfied. Attack-graph-based representations, however, focus on the conditions pertaining to local or remote exploitations only; they do not

capture the conditions for post-compromise actions such as exfiltration. In other words, attack graphs map to the potential intrusion itself and any associated vulnerabilities but do not cover the consequences of it. In VulnerVAN, we introduce a new attack step specification language to address the shortcomings of existing representations.

3. ARCHITECTURE

3.1 Nomenclature

Prior to describing the architecture of VulnerVAN, we first define the following terms used throughout this report.

- *Scenario Description*: File(s) containing information about the machines (e.g., OS base images, IP address) used in the cyber exercise or training, network topology, and network configurations. Different testbeds use different scenario description formats. For instance, CyberVAN stores machine-relevant and network-related information in an XML format. Users can use the CyberVAN Scenario Generation Graphical User Interface (SGGUI) to generate scenario description files.
- *OS Base Image*: A description file and virtual machine disk file(s) that define the OS and device drives for a system that can be deployed into scenarios; additional hardware details like computer processing unit count, random access memory, network interfaces, and hostname are defined in the scenario description. The description file includes machine properties.
- *Machine Properties*: Properties of a VM in the scenario. Each VM is associated with an OS base image. The underlying capabilities of a base image determine the properties of the machine. There can be two types of properties: intrinsic and runtime. Intrinsic properties are those that are intrinsic to the base image such as OS versions and installed software packages. Runtime properties are those that can be activated through running scripts (e.g., a workstation machine might take the role of an email client if a synthetic user agent is configured to use the email application).
- *Role/Capability*: A property of a machine required for an attack step to execute.
- *Attack Step*: A step the Red Team executes prior to or during training or an exercise to perform a cyber-attack action. For example, sending a phishing email and dumping the hash from memory are each attack steps.
- *Attack Rule*: A Prolog rule representing an attack step.
- *Attack Sequence*: A sequence of attack steps specified by the user. The steps need to be executed by the Red Team in the prescribed order. An example of attack sequence is provided.

3.2 Overview

We show in Figure 2 the high-level architecture of VulnerVAN. To use VulnerVAN to generate vulnerable scenarios, a user first provides a specification of the target network and a specification of an attack sequence. The user may use Network Input Collector and Attack Sequence Input Collector to provide required input. At the core of VulnerVAN is the Vulnerable Scenario Generator (VSG), which takes user input and generates as output (1) a CyberVAN scenario that supports exploits and other actions needed by the attack steps specified, and (2) a

sequence diagram with a realizable attack path in the generated scenario. VulnerVAN also annotates attack steps with references to a VulnerVAN-supplied attacker playbook to facilitate Red Team operations.

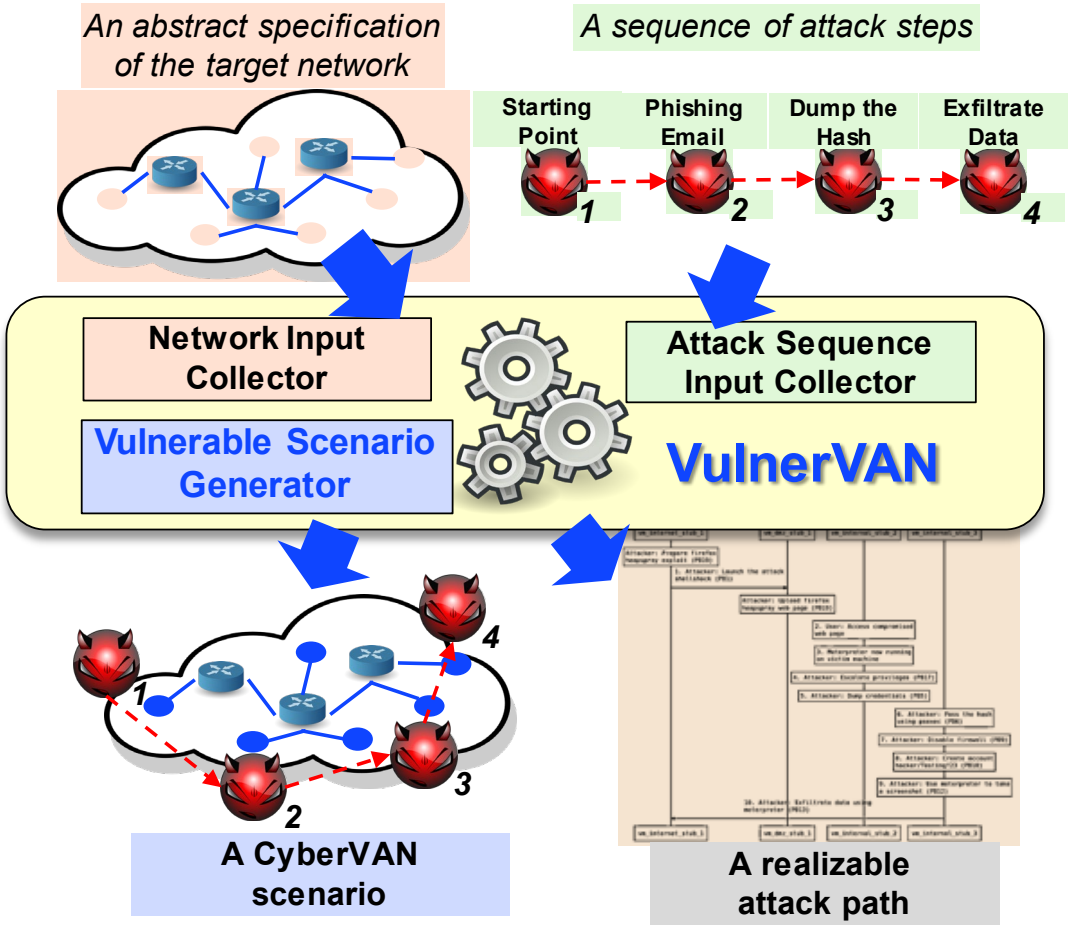

Figure 2 High-level architecture of VulnerVAN

Next, we describe the design and functionalities of VulnerVAN components.

3.3 Network Input Collector and Attack Sequence Input Collector

Network Input Collector is a command line tool that allows a cybersecurity training or exercise designer to specify high-level characteristics of a desired network scenario. Currently the tool supports the specification of a network environment with three segments: Internet, Demilitarized Zone (DMZ) Network, and Enterprise Network. For each network segment, the user specifies machine types, number of machines, and service types by answering a few questions. Currently, machine types include Windows client, Linux client, and Attacker, while service types include Domain Name System (DNS) Server, E-mail Server, and Web Server. Both machine types and service types can be expanded to include additional types to address cyber training and exercise

needs. In Figure 3, we provide an example to illustrate input that may be provided by a user to specify the high-level characteristics of a desired network.

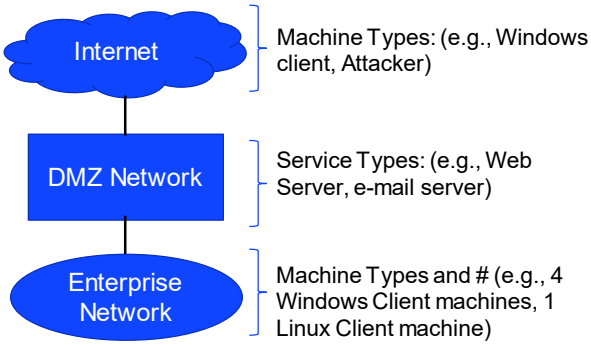

Figure 3. VulnerVAN Network Input Collector

Note that using Network Input Collector is not necessary. A user may use CyberVAN's SGGUI to define a desired network scenario if they have specific requirement on the network topology, network types, and/or network configurations. The downside of this approach is that the user might need to edit scenario-definition files generated by the SGGUI to fit the desired characteristics.

On the other hand, Attack Sequence Input Collector is another command-line tool that allows specification of an attack sequence, which is a chain of attack steps that cyber attackers may execute. Each attack step is based on a tactic, technique, and/or procedure (TTP) defined in the MITRE ATT&CK framework (Strom et al., 2017). Cyber training or exercise designers specify desired attack steps, techniques, and exploits by selecting from presented choices. As a proof-of-concept, VulnerVAN currently includes the implementation of 30 MITRE ATT&CK techniques in its attack repository. This is about one-tenth of all techniques currently listed on the MITRE ATT&CK web page (Strom et al., 2017).

3.4 Attack-Specification Language, Attack Sequence, and Attack Library

In VulnerVAN, an attack step corresponds to a technique defined in MITRE's ATT&CK knowledge base. We model each attack step as a five-tuple: a set of pre-conditions (represented as Boolean predicates), a set of pre-execution steps, an action, a set of post-execution steps, and a set of post-conditions (also Boolean predicates). Internally, the attack step definitions are stored in XML format. An attack-step definition example for the ATT&CK's Spear Phishing Link attack technique is shown in Figure 4. MITRE defines the Spear Phishing Link technique as the act of sending a spear phishing email with a link to a malicious webpage. In an attack-step definition, the *pre-conditions* section specifies the prerequisites for an attack step; all pre-conditions must be satisfied by a machine's capabilities and network configuration requirement in order to realize the corresponding attack step. If the pre-conditions are satisfied, the steps in the *pre-execution* section set up the machines to enable the attack step. The *action*

section specifies the instructions to execute the technique. The instructions in the *post-execution* section perform post-processing of the attack to verify or modify the machine's state. Finally, after successfully executing the action of an attack step, the changes to the state of the machine(s) are captured by *post-conditions*.

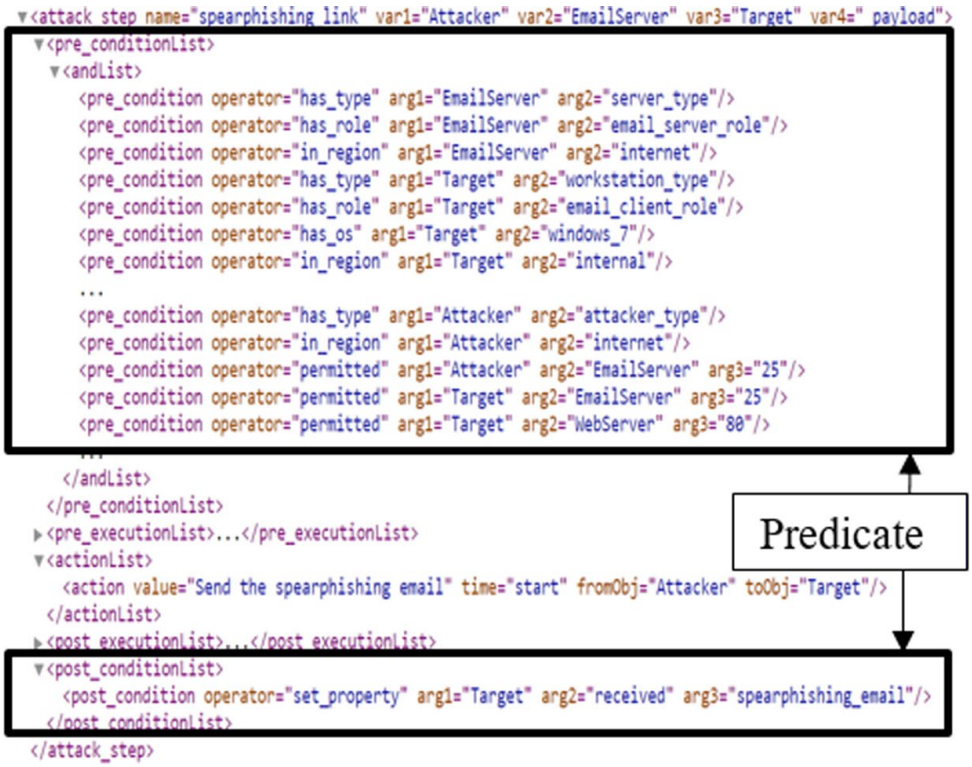

Figure 4. Attack-step definition for Spear Phishing Link technique

Here we use the Spear Phishing Link attack step in Figure 4 to illustrate the concept. The "*pre_conditionList*" construct in Figure 4 specifies the requirements for enabling this attack step. The requirements mandate a list of machines in the generated scenario playing certain roles. Assuming a spear-phishing attack starts from a malicious actor on the Internet, the scenario generated by VulnerVAN must include machines playing the following roles: (1) a *Server* machine in the Internet capable of playing the role of *Email Server* to deliver spear-phishing email to victim user(s); (2) a *Workstation* machine in the enterprise network on which the user could fall victim to the phishing attack; (3) a machine in the Internet of *Attacker* type from which spear-phishing email is sent; and (4) a machine in the Internet that can play the role of *Web Server* to host and serve malicious webpages (not in Figure 3). These requirements are captured through 13 Boolean predicates that can be broadly categorized as machine-based predicates and network-based predicates. Machine-based predicates can be static or dynamic. Static machine-based predicates include *has_os*, *has_software*, *has_default*, *is_vuln*, *has_type*, *has_role*, *equal*, and *not_equal*. The *has_os*, *has_software*, *has_default*, and *is_vuln* predicates capture the OS, software, default application, and vulnerability requirement of the machine respectively, while

has_type and *has_role*, respectively, capture a machine's type (workstation/server/attacker) and the role (e.g., email server or web server) that the machine can assume (e.g., by starting appropriate applications and services). For instance, in the Spear Phishing Link example, VulnerVAN requires a machine, *Target*, that has a workstation type and can play the role of an email client by starting the email application. Dynamic machine-based predicates, on the other hand, capture the ephemeral states that are reached when solving the goal predicate. These ephemeral states are reached as part of the post-condition after the requirements specified in the pre-condition are met and are critical in linking the attack steps. For instance, the spear-phishing attack step is often followed by the "email user execution" attack step in which a user opens the spear-phishing email and clicks on the malicious link. Here, the pre-condition for the email-user-execution attack step is that the email has been received by the target machine in a previous step. As shown in Figure 4, the ephemeral state (i.e., receiving email) is captured by the *set_property* predicate in the attack step's post-condition. The ephemeral states are retrieved by the subsequent attack steps using the *has_property* (or *not_has_property*) predicates.

Network-based predicates include *in_region*, *permitted*, and *can_initiate*. An attack sequence might require the participation of machines from different *regions* of the target network environment. A region is a collection of machines that are grouped together based on a common property. One intuitive property is the network segment on which the machines reside. For instance, a typical enterprise network can be partitioned into three regions: *Internal*, *DMZ*, and *Internet*. An attack sequence might require an attacker that resides in the Internet to compromise a machine that resides in the internal portion of the enterprise network. The *in_region* predicate captures these constraints. The *permitted* and *can_initiate* predicates capture the network-level firewall rules. For instance, for the target workstation machine to access the malicious webpage, the firewall rule must permit all connection requests from the Target to the Web Server on Port 80.

In addition to the pre-/post-conditions, the pre-/post-execution and action sections provide instructions to configure the machines prior to the exercise and to create an attack blueprint for the Red Team. For an automated attacker, we envision these instructions being replaced by stand-alone scripts that can automatically configure machines and trigger actions to simulate attack behavior during the course of the exercise. Such an automated attacker can enable the experimenters to accommodate changes to the defenses—as requested by the Blue Team—and route around those defenses when multiple attack paths are present. Building such an automated attacker for VulnerVAN is part of our future work.

Each instruction within the pre-/post-execution/action sections provides information regarding *what to do* and *when to do it*. An example instruction for the action section of the Spear Phishing Link attack step is shown in Figure 4. The *when* attribute is used to chronologically order the instructions. There are three types of *when* objects: boot, start, and delay. Boot and start instructions are executed immediately after the machines boot and at the start of the exercise,

respectively, while delay instructions are triggered during the course of the exercise. The boot and start instructions typically include configuration information while delay instructions include triggering attack events. Since an attack sequence can be composed of multiple attack steps, an instruction can be executed as soon as the previous attack step's action is completed, after the previous attack step's action and post-execution are completed, or after all the instructions in the pre-execution of the current attack step are executed. Therefore, we have included three types of delay instructions, one for each possibility: *action*, *step,* and *pre_exec*.

As a proof of concept, we have constructed a library of 30 MITRE ATT&CK techniques spanning all 12 tactics: Initial Access, Execution, Persistence, Privilege Escalation, Defense Evasion, Confidential Access, Discovery, Lateral Movement, Collection, Exfiltration, Command and Control, and Impact. As part of our ongoing effort, we are adding more attack steps for different tactics to this library.

In VulnerVAN, the list of attack steps that form an attack sequence is represented in an XML format. During the composition of the attack sequence, users specify the attack step names and the arguments to instantiate it. There are two types of arguments for an attack step: user-specified and relational. User-specified arguments allow a user to instantiate exact values for them. For example, the *_payload* argument in the Spear Phishing Link attack step is a user-specified argument requiring the user to specify the payload (e.g., bot) that will be delivered after exploitation. Relational arguments allow users to indicate relationships between the arguments of different attack steps. For instance, when the Spear Phishing Link attack step is followed by the email-user-execution attack step, the user can specify that the workstation machines involved in these steps should be the same in order to form a logical chaining of events. As part of our future work, we plan to develop a GUI to assist users in creating attack sequences through visualizing the logical dependency among steps.

3.5 Vulnerable Scenario Generator

VulnerVAN uses VSG to convert user input, including a high-level network scenario description and an attack sequence. As shown in Figure 5, VSG consists of two major components: Attack Mapper and Attack Orchestrator. Attack Mapper processes user input on the network environment to generate a CyberVAN scenario description file, and uses attack specifications in the VulnerVAN's attack library to generate a mapping of nodes to actual VMs. Attack Orchestrator takes the mapping generated by Attack Mapper as input and generates as output configuration scripts for the VMs that play certain roles in the scenario and an attack-sequence diagram as high-level instructions for executing the attack sequence given by the user.

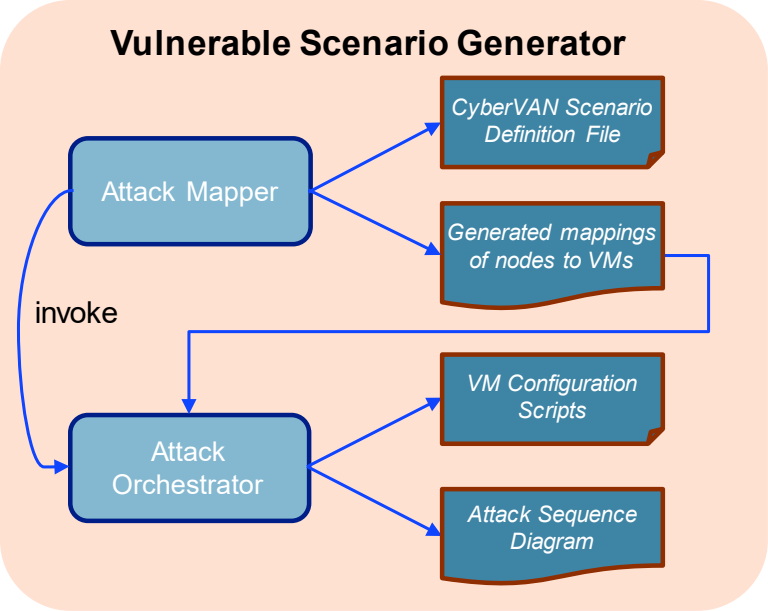

Figure 5. VSG architecture

3.6 Attack Mapper

Attack Mapper begins with generating a CyberVAN scenario template from user input. The template includes generic information of all nodes in the scenario, allowing Attack Mapper to substitute nodes with qualifying OS base image names in the solution seeking process. This allows Attack Mapper to search through all possibilities. In the running example of the Spear Phishing Link attack, the user would specify the number of machines of each type (Windows or Linux) in the Internet. Attack Mapper, leveraging Prolog to perform reasoning (Wielemaker, et al., 2012), would use the information of Spear Phishing Link attack in the attack step library, along with the given constraints on machine types, machine quantities, and other user-specified constraints, to seek solutions. It first enumerates all possible satisfactory OS base image assignments to nodes that can realize the very first attack step and then continue with exploring options for the rest of the steps in the attack sequence.

In the following, we describe this process in detail. After a user creates a CyberVAN scenario and supplies the description of the attack sequence, VSG translates them into Prolog facts and rules. First, the base images chosen for the machines are identified and the capabilities of the base images are then translated into Prolog facts. Next, the network-relevant information is translated. There are three parts to the network information: region, connectivity, and firewall. The region and the connectivity between machines are translated as Prolog facts, and the firewall rules are translated as Prolog rules.

We have developed a mechanism to automate the process of translating a scenario description to its corresponding Prolog facts. We have also developed another off-line mechanism to parse OS base images in the VulnerVAN base image library to automatically retrieve properties that are

required to support the solution reasoning process. As a result, Attack Mapper can retrieve from the OS base image library the properties of all qualified OS base images and convert the properties into Prolog facts.

Next, the attack sequence description is translated. For VulnerVAN, we developed a tool—the Attack Step Translator—that can parse the attack sequence description and automatically generate Prolog rules for each attack step. This tool uses the definition of the corresponding attack step in the attack-step library to construct the head of the attack rule and the relevant pre-conditions to construct the body of the rule. As an example, Figure 6 shows the attack rule for the Spear Phishing Link attack step defined in Figure 4.

```
spearphishing_link(Attacker, EmailServer, WebServer,
                   Target, DNS, [Head1 | Tail], PropsBefore) :-
    has_type(EmailServer, server_type),
    has_role(EmailServer, email_server_role),
    in_region(EmailServer, internet),
    has_type(Target, workstation_type),
    has_role(Target, email_client_role),
    has_os(Target, windows_7),
    in_region(Target, internal),
    ...
    has_type(Attacker, attacker_type),
    in_region(Attacker, internet),
    permitted(_, Attacker, EmailServer, 25),
    permitted(_, Target, EmailServer, 25),
    permitted(_, Target, WebServer, 80),
    ...
    Head1 = [Target, received, spearphishing_email],
    Tail = PropsBefore.
```

Figure 6. Attack rule in prolog for the Spear Phishing Link attack

After constructing individual attack rules, the overall attack sequence is assembled as a rule whose body is composed of individual attack rules. During this process, the variables corresponding to the user-specified parameters are translated as atoms, and the relational arguments are substituted with the appropriate variables to ensure consistency with the user's specifications. The resulting rule is treated as the goal predicate.

In the goal predicate, the free variables represent the roles/capabilities that the machines in the scenario should satisfy. Therefore, after translation, VulnerVAN invokes the Attack Mapper to query the goal predicate. During this process, Attack Mapper enumerates all possible combinations of machines that can be used either "as-is" or can be configured to realize the attack sequence. For each combination, it maps the roles to machines (e.g., *EmailServer* to *machine1*). Presenting the user with multiple mappings allows them to choose among different attack paths that could be mapped onto the scenario and reason about which paths may provide the necessary effects required to evaluate a CPT during their exercise.

One of the challenges with Attack Mapper is that for large networks and complex attack sequences, there could be a large number of mappings. Nevertheless, in our limited experiments with 20–30 nodes in a scenario, we found that the solution seeking process was quite efficient.

For example, in a scenario with 20 nodes and an attack sequence of Length 7, Attack Mapper reports all possible 2000+ solutions in less than 2 s on a modern four-core computer. We have not performed extensive tests on VulnerVAN's performance in relation to the size of the network, as typical cybersecurity training or exercise includes tens of nodes rather than hundreds of nodes. Currently, we have the ability to restrict the number of solutions generated by Attack Mapper.

By default, the first solution will be used to construct the scenario, as defining metrics to rank and recommend solutions are not in the current scope of the project. However, a user could examine all generated solutions and pick one from all qualified candidate solution if preferred. Another point worth mentioning is that many of the generated solutions are similar. For example, two solutions could differ only in two Windows machines in the same subnet switching their roles for the same attack sequence. Such solution similarity could be useful for training, as trainees could practice with a slightly different scenario if they fail the previous training session.

3.7 Attack Orchestrator

Attack Mapper feeds Attack Orchestrator a mapping of nodes to VMs and the roles of VMs for realizing the given attack sequence. For each VM in the scenario, Attack Orchestrator identifies minimal configurations that need to be applied to the VMs to execute the tasks required of their assigned roles. For example, in the case of executing a spear-phishing attack, multiple configurations need to be set up correctly. First, the phishing email needs to be addressed to a recipient recognized by the email server, and the email needs to be delivered to a recipient using the machine that is supposed to be the victim machine of the phishing attack. Next, it is necessary to have the recipient open the phishing email and click the link in the email to download and run malware from the malicious website. To facilitate that in a training scenario on a CyberVAN testbed, Attack Orchestrator needs to configure ConsoleUser, a software agent that can mimic a human user's actions, to click the malicious link. Finally, Attack Orchestrator also needs to install ConsoleUser prior to scenario runtime and configure the boot script on the appropriate machine to activate ConsoleUser during scenario runtime. This example shows that a number of configurations must be planned and provisioned to support the execution of one attack step.

We show sample output of Attack Orchestrator in Figure 7 to illustrate the sequence diagram and scripts generated to support the Spear Phishing Link attack. For each of the victim machines, Attack Orchestrator outputs an attack blueprint to assist a Red Team in realizing the attack sequence during an exercise, or to assist training organizers to set up training scenarios. For each attack step, Attack Orchestrator parses the instructions in the pre-execution, action, and post-execution sections of the attack step (in the chosen attack sequence) and then determines the machines that must execute these steps based on the roles they are assigned in the mapping. Here, the instructions that are expected to be executed at boot time correspond to the machine's

configuration, while those that occur at start or after a delay correspond to an attack action by a Red Team or an automated attacker. For start and delay instructions, Attack Orchestrator chronologically orders them into a logical sequence of actions. Finally, Attack Orchestrator uses the js-sequence-diagrams utility (Brampton, n.d.) to output a sequence diagram containing machine configurations and a timeline of interactions among different machines that realize the attack sequence.

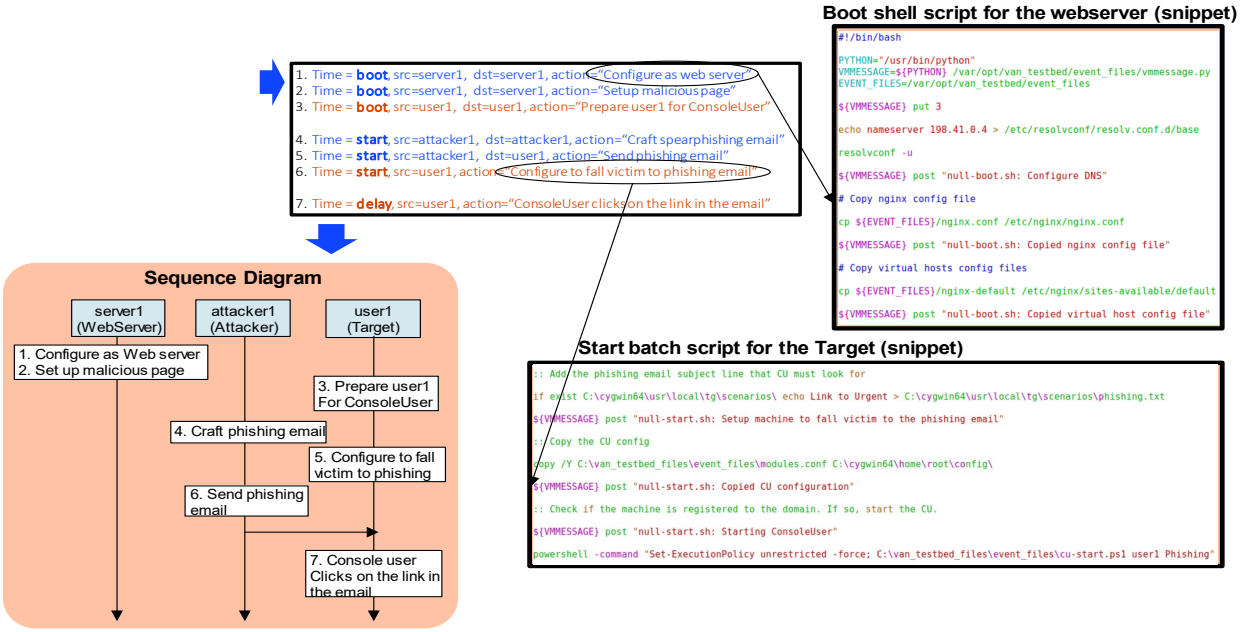

Figure 7. Sample output of Attack Orchestrator

4. USE CASE 1: APT-STYLE DATA EXFILTRATION NO. 1

In this section, we demonstrate the performance of VulnerVAN using an example scenario of an APT-style data exfiltration from an enterprise network.

4.1 Scenario and Attack Description

We consider a network with three regions: Internet, DMZ, and Internal. The Internet region was composed of six machines playing different roles, including attacker; command and control server (C2); drop-site for transferring the exfiltrated data; email server; an attacker-controlled webserver; and DNS. The DMZ region was composed of three machines, each with different capabilities including email server, web server, and database server. The Internal region is composed of workstation machines running Windows 7, with Server Message Block (SMB) services enabled, and running Firefox 50.0.1 as its default web browser. The workstations can play the role of an email client. These machines are also vulnerable to CVE-2017-5375, a Firefox vulnerability, and CVE-2017-0143, an SMB protocol vulnerability exploited by EternalBlue. There are two firewalls, one between the DMZ and the Internet and one between the Internal and the DMZ, and the rules are set to be overly permissive (i.e., allow any packet between all the machines).

Figure 8 depicts an attack sequence that captures a typical APT-style data exfiltration. The attack sequence begins with the attacker sending a spear-phishing email, containing a link to a malicious page, to a user on a workstation machine. After the user clicks the link, the malicious page opens in the Firefox browser. The code embedded in the malicious page exploits the vulnerability in Firefox and downloads a malware to the victim machine. At the start of its life cycle, the malware contacts the C2 server in the Internet. The attacker, or Red Team, first scans for machines and then performs a lateral movement to another machine by using the EternalBlue exploit. Finally, the attacker instructs the malware on one of the machines to search for PDF files with matching keywords and then exfiltrate them to a drop-site (in the Internet).

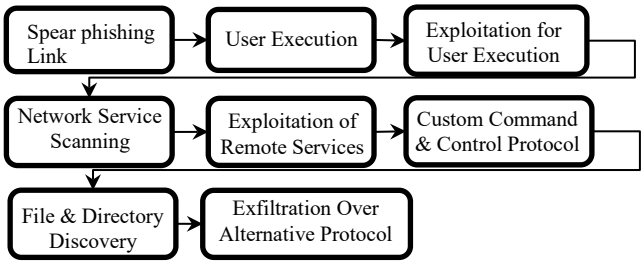

Figure 8. Attack sequence used in an evaluation of VulnerVAN

We use this network and attack setting as our baseline case to evaluate VulnerVAN's performance in generating mappings. We consider two variations of the described scenario: Variation 1, the firewall is configured to deny packets to *one* of two attacker machines; and

Variation 2, the attacker instructs malware to exfiltrate files from *both* the machines. Variation 1 depicts blacklisting of an IP address as configured by a defender while Variation 2 depicts a potential additional action executed by an attacker.

4.2 Evaluation

For each setting—baseline and the two variations—we varied the size of the network by increasing the number of workstation machines in the Internal region. This leads to multiple candidates for the initial foothold and the remote exploitation attack steps. As shown in Figure 9, the number of mappings (i.e., combinations of machines that realize an attack sequence grows quadratic with network size in the baseline setting). With the introduction of a firewall rule to deny packets to one of the attacker-type machines, the number of mappings in Variation 2 is half as many as in the case of the baseline.

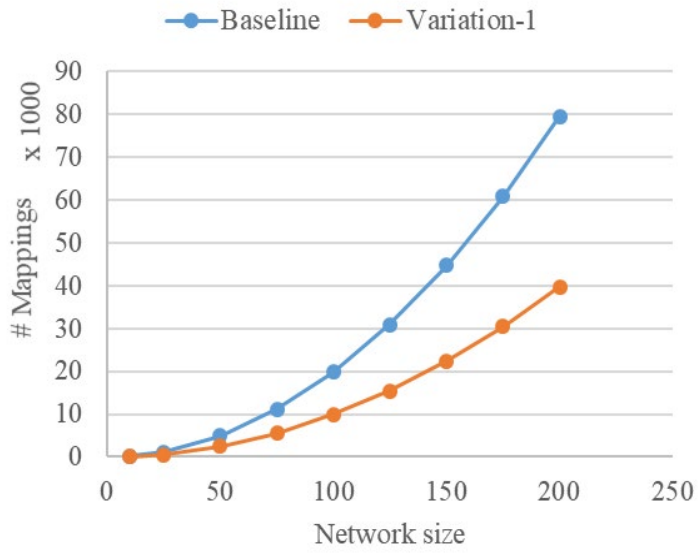

Figure 9. Number of mappings for different network sizes. Note the number of mappings for baseline and Variation 1 is the same.

Figure 10 presents the time taken (averaged over 10 runs) to generate the solutions for different settings. As expected, since Variation 1 has fewer candidate mappings, it generates the solutions quicker than the other two settings. Furthermore, since Variation 2 has an additional attack step—exfiltration from two machines (instead of one)—VulnerVAN takes additional time to verify if both the machines can search and exfiltrate files to the drop-site.

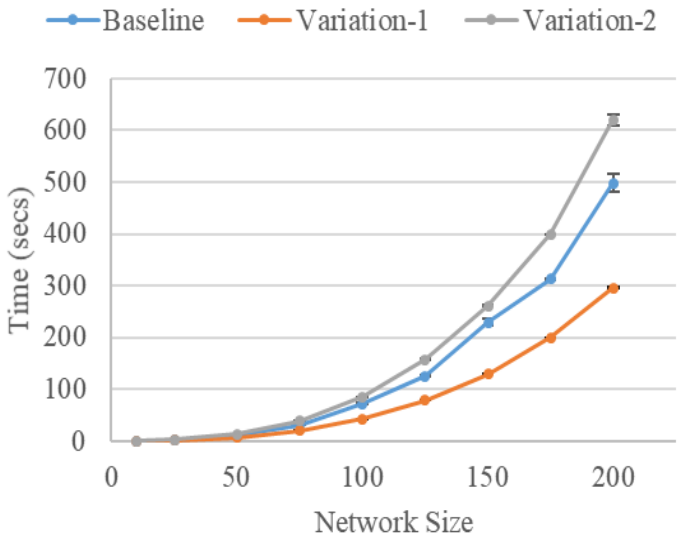

Figure 10. Average mapping generation time for different network sizes

5. USE CASE 2: APT-STYLE DATA EXFILTRATION NO. 2

Here we provide another use case to illustrate the functionalities of VulnerVAN (Figure 11).

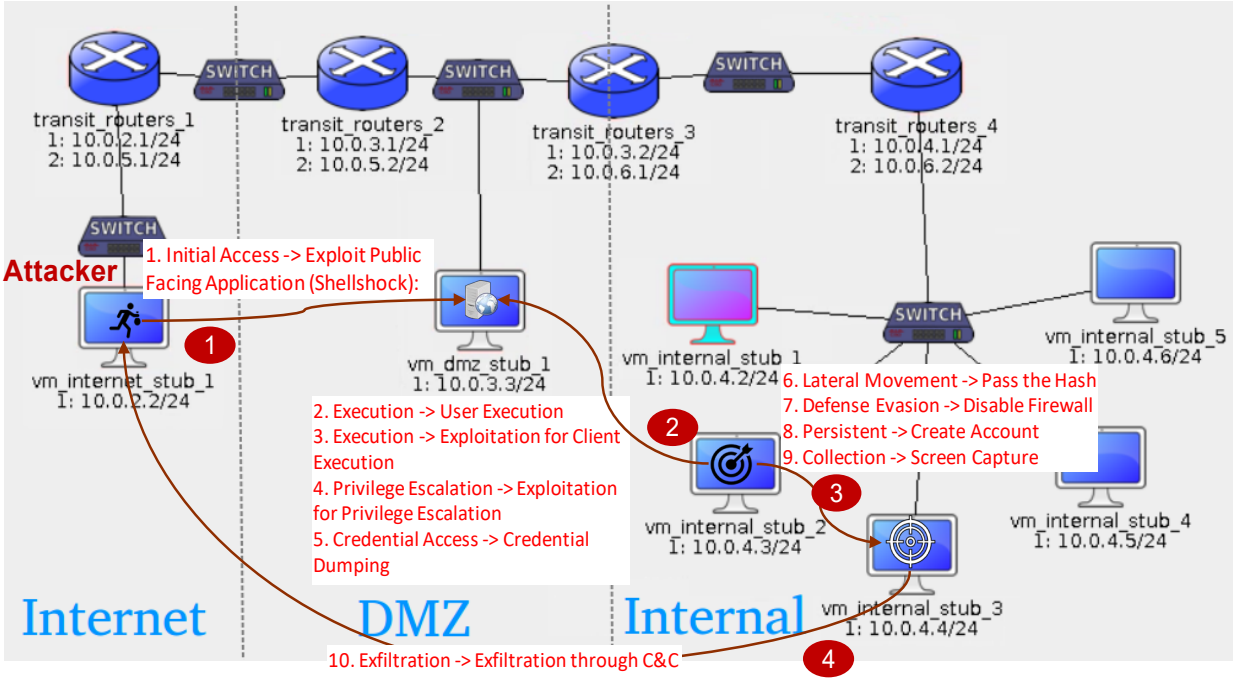

Figure 11. APT-style data exfiltration example

This sample attack sequence consists of 10 attack steps, as shown in the figure. We use VulnerVAN to automate the generation of a network scenario and provision the VMs to support the attack sequence. The sequence starts with an attacker compromising a web server *vm_dmz_stub1* in the DMZ through exploiting a public-facing application, as the web server has an unpatched Shellshock vulnerability. After this attacker gets a foothold on this machine, it exercises a series of local attack actions to achieve privilege escalation in order to dump login credentials from legitimate users such as system admins who maintain the web server. The attacker then uses the stolen login credential to penetrate into the machine *vm_internal_stub2* in the internal network. After getting into the internal network, the attacker identifies the machine that has valuable information and uses a pass-the-hash technique to make another lateral movement to the machine *vm_internal_stub3*. To evade cyber defense and facilitate exfiltration, the attacker disables the firewall on the machine, creates an additional user account on the machine for persistence, and collects valuable information using screen-dumping techniques. The stolen screen captures are then shipped back to the attacker in the Internet.

6. DISCUSSION AND FUTURE WORK

We have verified the functionalities of the VulnerVAN prototype using many combinations of network scenarios and attack sequences to automatically generate vulnerable scenarios. From our hands-on experience with the toolset, we identified the following work items as future work.

First, we plan to augment our attack library to include additional MITRE ATT&CK TTPs to enrich possible training and exercise scenarios. We envision this as an ongoing process, as we plan to add new TTPs as needed.

Second, we plan to develop metrics to rank the solution candidates generated by Attack Mapper. These metrics could be used for different selection purposes. For example, one metric could be developed to measure stealthiness of solution candidates, another metric could be developed to measure similarity among solution candidates, and yet another could be developed to measure difficulty level. For example, we may score different mappings based on a notion of their utility and restrict the user's choice to those mappings that are above a pre-defined threshold score. Such a scoring can potentially also be used to calibrate the difficulty of detecting the attack, in terms of the number of alerts that could be generated by an Intrusion Detection System.

Third, currently both Network Input Collector and Attack Sequence Input Collector are command-line utilities. We plan to develop a GUI to assist training and exercise organizers in developing their scenarios and provide a more convenient interface for comparing and selecting solution candidates.

Finally, developing an automated cyber attacker agent (Automated Attacker in short) is our long-term goal. As mentioned in the report, the sequence of actions included in pre-execution and post-execution are tasks that need to be executed by Automated Attacker. We envision Automated Attacker as not software with predefined logic to perform actions but rather has a reasoning capability to assess the situation and make decisions to reach the goal state. Automated Attacker can be used to perform Red Team functions on behalf of human beings and thus makes VulnerVAN more versatile and adaptive to actions taken by trainees or Blue Team in the live cybersecurity training or exercise.

7. CONCLUSIONS

Generating vulnerable network scenarios for conducting cyber training and cyber exercise is labor-intensive and time-consuming. To this end, we designed and developed VulnerVAN—the first of its kind tool for automating the generation of vulnerable network scenarios. VulnerVAN takes user input on network environment and attack sequence. It uses a Prolog-based reasoning engine to automatically enumerate all possible node assignments that could support the implementation of the given attack sequence in the given network environment. After one of the solutions has been selected, we also designed and implemented orchestration services that prepare a ready-to-go scenario. The services include generation of various configuration scripts to set up VMs for the roles they are assigned as well as an annotated sequence diagram that Red Team could use to execute the attack sequence in the network environment. As part of our ongoing and future work, we plan to expand our attack-step library, develop metrics to rank generated solutions, develop a GUI to further simplify the user input process, and build an automated cyber attacker that can automatically execute the generated attack sequence.

8. REFERENCES AND DOCUMENTS

Barnum, S. (2012). *Standardizing cyber threat intelligence information with the structured threat information expression (STIX)*. The MITRE Corporation.

Brampton, A. (n.d.). *js-sequence-diagrams*. https://bramp.github.io/js-sequence-diagrams/

Cain, P., & Jevans, D. (2010). *Extensions to the IODEF-document class for reporting phishing*. (No. RFC 5901). Internet Engineering Task Force. https://tools.ietf.org/html/rfc5901

Chadha, R., Bowen, T., Chiang C. J., Gottlieb, Y., Poylisher, A., Sapello, A., Serban, C., Sugrim, S., Walther, G., Marvel, L., Newcomb, E. A., & Santos, J. (2016). *CyberVAN: a cyber security virtual assured network testbed*. Proceedings of the IEEE MILCOM Conference, 1125–1130.

Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., & Bowman, M. (2003). Planetlab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, *33*(3), 3–12.

Damodaran, S. K., & Couretas J. M. (2015). *Cyber modeling & simulation for cyber-range events*. Proceedings of the Conference on Summer Computer Simulation, Society for Computer Simulation International, 1–8.

Davis, J. & Magrath, S. (2013). *A survey of cyber ranges and testbeds.* (No. DSTO-GD-0771). Defence Science and Technology Organization, Cyber and Electronic Warfare Division.

McCulloch, R. (2017, April 26). *Exercise Cyber Shield 2017 gets underway*. CCDC Army Research Laboratory. https://www.army.mil/article/186702/exercise_cyber_shield_2017_gets_underway

Mirkovic, J., Venzel, T. V., Faber, T., Braden, R., Wroclawski, J. T., Schwab, S. (2010). *The DETER project: advancing the science of cyber security experimentation and test*. IEEE International Conference on Technologies for Homeland Security (HST), p. 1–7.

Pham, C., Tang, D., Chinen, K., & Beuran, R. (2016). *CyRIS: a cyber range instantiation system for facilitating security training*. Proceedings of the Seventh Symposium on Information and Communication Technology, 251–258.

Schreuders, Z. C., Shaw, T., Shan-A-Khuda, M., Ravichandran, G. Keighley, J., & Ordean, M. (2017). *Security scenario generator (SecGen): a framework for generating randomly vulnerable rich-scenario VMs for learning computer security and hosting CTF events*. Proceedings of the USENIX Workshop on Advances in Security Education (ASE 17). https://www.usenix.org/system/files/conference/ase17/ase17_paper_schreuders.pdf

Securities Industry and Financial Markets Association. (2017 November 7–8). *Cybersecurity exercise: Quantum Dawn IV*. https://www.sifma.org/resources/general/cybersecurity-exercise-quantum-dawn-iv/

Strom, B. E., Battaglia, J. A., Kemmerer, M. S., Kupersanin, W., Miller, D. P., Wampler, C., Whitley, S. M., & Wolf, R. D. (2017). *Finding cyber threats with ATT&CK™-based analytics.* (MITRE Technical Report MTR170202). The MITRE Corporation.

U.S. Department of Homeland Security. (n.d). *Cyber Storm: securing cyber space.* Cybersecurity & Infrastructure Security Agency. https://www.dhs.gov/cisa/cyber-storm-securing-cyber-space

Wang, L., Singhai, A., & Jajodia, S. (2007). *Measuring the overall security of network configurations using attack graphs*. Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy. Springer, 98–112.

Wielemaker, J., Schrivers, J., Triska, M., & Lager, T. (2012). Swi-prolog. *Theory and practice of logic programming*, *12*(1-2), 67–96.

Appendix A – List of Acronyms

APT	advanced persistent threat
C2	command and control
CPT	cyber protection team
DEVCOM	U.S. Army Combat Capabilities Development Command
DNS	Domain Name System
DMZ	Demilitarized Zone
GUI	graphical user interface
IP	Internet Protocol
OS	operating system
PDF	portable document format
SGGUI	Scenario Generation Graphical User Interface
SMB	Server Message Block
TTP	tactic, technique, and/or procedure
VM	virtual machine
VSG	Vulnerable Scenario Generator
XML	Extensible Markup Language

Appendix B – Distribution List

ORGANIZATION

DEVCOM Data & Analysis Center
FCDD-DAH-N/B. Hoffman
FCDD-DAH-N/N. Buchler
6560 Surveillance Loop
Aberdeen Proving Ground, Maryland 21005

DEVCOM Army Research Laboratory
FCDD-RLC-ND/E. A. Newcomb
Aberdeen Proving Ground, Maryland 21005

DEVCOM Army Research Laboratory
FCDD-RLD-DCI/Tech Library
2800 Powder Mill Rd.
Adelphi, MD 20783

Defense Technical Information Center
ATTN: DTIC-O
8725 John J. Kingman Rd.
Fort Belvoir, VA 22060-6218