2014

CREATING REUSABLE VIRTUAL MACHINES TO SIMULATE NETWORKS FOR CYBER CHALLENGES

Daniel P. Masterson
University of Rhode Island, dmasterson@my.uri.edu

CREATING REUSABLE VIRTUAL MACHINES TO SIMULATE NETWORKS FOR CYBER CHALLENGES

BY

DANIEL P. MASTERSON

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

COMPUTER SCIENCE AND STATISTICS

UNIVERSITY OF RHODE ISLAND

2014

MASTER OF SCIENCE THESIS

OF

DANIEL P. MASTERSON

APPROVED:

Thesis Committee:

Major Professor Victor Fay-Wolfe

Lisa DiPippo

Yan Sun

Nasser H. Zawia
DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2014

ABSTRACT

Cyber challenges are a way for educators to train and assess the desperately needed cyber security professionals of the future. Creating these challenges can be time consuming, error prone, and/or cost-prohibitive. To address these issues, the University of Rhode Island has created the freely-available Open Cyber Challenge Platform (OCCP). To maximize ease of use and re-use of the challenges created on this platform the OCCP required a tool to automate the configuration and deployment of virtual machines. This thesis project created such a tool that allows OCCP challenge implementers to create virtual machines that are reconfigurable and sharable with the OCCP community.

ACKNOWLEDGMENTS

I would like to thank my major professor Dr. Victor Fay-Wolfe for his support and guidance while working with his research group. Without the opportunities he exposed me to, my graduate school career would not have been possible. I would also like to thank my committee members Dr. Lisa DiPippo and Dr. Yan Sun for their support and feedback throughout this process.

Additionally, I would like to thank Dr. Kevin Bryan for his advice and guidance throughout my time at the University of Rhode Island. His wealth of knowledge and patience has been an invaluable resource for me. I am also extremely grateful for the advice that Lorraine Berube and Catherine Robinson have provided me over the years. Finally, I would like to thank my friends and family for their constant motivation and support.

TABLE OF CONTENTS

Page

Page

Page

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Statement of the Problem

There does not exist a tool to create reusable virtual machines for cyber challenges, which makes cyber challenges difficult to create and reuse. This thesis project extends the Open Cyber Challenge Platform by creating a tool to automate the configuration and deployment of virtual machines in cyber challenges.

1.2 Motivation

The market for cyber security jobs is large and continuing to grow. According to Burning Glass International Inc., a Boston-based company that uses artificial intelligence to match jobs and job seekers, cyber security postings have grown 74% from 2007 to 2013. Twice as fast as all IT jobs. The report also indicates that the demand for cyber security talent is exceeding the supply. These job postings took 36% longer to fill than all job postings [1]. There is a dire need to educate more people to fill these positions.

Many agree that cyber challenges are a useful tool for education [2, 3, 4, 5] because they allow participants to both learn and test their abilities in a hands-on manner. Cyber challenges simulate real world networks, and allow participants to practice their skills without harming production systems. However, creating a cyber challenge is a difficult and time-consuming process, and the efforts generally have limited use after the initial offering of the challenge. Challenges require a high degree of technical ability to create and take significant time to plan and implement. Once they are created, they require additional time and effort to change subtle details for reuse.

The Open Cyber Challenge Platform (OCCP), being developed by the Uni-

versity of Rhode Island under funding from the U.S. National Science Foundation, aims to provide a common platform on which challenges could be run. This thesis project extends the OCCP to facilitate reuse and shareability amongst the platform's users. Among these users are different audiences with varying technical abilities. These audiences will make use of this work based on their skill level.

1.3 Goals

The goals of this thesis project are:

Goal 1: Create an application and administrative virtual machine (Admin VM) to configure virtual machines for the OCCP

- The Admin VM should be able to configure virtual machines by performing tasks such as the installation of software, creation of user accounts, configuration of services, transfer files and other content.
- The Admin VM should provide a mechanism to produce automatically generated content such as usernames, passwords, and SSH keys for use in the configuration of the virtual machines.

Goal 2: The extended Admin VM will have reasonable expectations of technical ability for the target audiences.

- The extension should build on the existing syntax used by current Admin VM and extend it by providing the ability to describe the configuration of the desired virtual machine.
- The extension should only include components that are reasonable for different audiences with varying technical skill to operate.

Goal 3: Avoid the introduction of a financial cost

The extension of the Admin VM should not introduce a financial cost to extend the OCCPs current functionality.

Goal 4: The extended Admin VM should provide a distribution mechanism with reasonable expectations of ability for the target audiences

The Admin VM should provide a mechanism by which the virtual machines that it produced could be shared with other OCCP users. The mechanism should consider the skill levels of the target audiences to ensure usability.

1.4 Summary of Accomplishments

This thesis project created an Admin VM and program capable of creating reusable virtual machines for cyber challenges. Challenge contributors are able to create virtual machines that can simulate real networks for the Open Cyber Challenge Platform and share their work with other users of the OCCP at no financial cost. Contributed challenges may be extended and altered by various audiences of this work.

List of References

[1] Bruning Glass. "Job market intelligence: Report on the growth of cybersecurity jobs." Aug. 2014. [Online]. Available: http://www.burning-glass.com/media/4187/Burning%20Glass%20Report%20on%20Cybersecurity%20Jobs.pdf

[2] K. E. Stewart, T. R. Andel, and J. W. Humphries, "Measuring the performance of network virtualization tool n2n in the design of a cyber warfare training and education platform," in *Proceedings of the 2011 Military Modeling & Simulation Symposium*, ser. MMS '11. San Diego, CA, USA: Society for Computer Simulation International, 2011, pp. 28–35. [Online]. Available: http://dl.acm.org/citation.cfm?id=2048558.2048563

[3] R. S. Cheung, J. P. Cohen, H. Z. Lo, and F. Elia, "Challenge based learning in cybersecurity education," in *Proceedings of the 2011 International Conference on Security & Management*, vol. 1, 2011.

[4] P. J. Wagner and J. M. Wudi, "Designing and implementing a cyberwar laboratory exercise for a computer security course," in *Proceedings of the 35th*

SIGCSE Technical Symposium on Computer Science Education, ser. SIGCSE '04. New York, NY, USA: ACM, 2004, pp. 402–406. [Online]. Available: http://doi.acm.org/10.1145/971300.971438

[5] A. Doupé, M. Egele, B. Caillat, G. Stringhini, G. Yakin, A. Zand, L. Cavedon, and G. Vigna, "Hit 'em where it hurts: A live security exercise on cyber situational awareness," in *Proceedings of the 27th Annual Computer Security Applications Conference*, ser. ACSAC '11. New York, NY, USA: ACM, 2011, pp. 51–61. [Online]. Available: http://doi.acm.org/10.1145/2076732.2076740

CHAPTER 2

Background

In this chapter I first summarize URI's Open Cyber Challenge Platform in Section 2.1. Then I define the target audiences of this thesis project in Section 2.2. Next, I review a recently discovered project with similarities to the OCCP in Section 2.3. Finally, I review existing technologies in Section 2.4 and discuss how they may, or may not, be used to meet the goals of this thesis. These sections provide the context for where the work of this thesis begins.

2.1 URI's Open Cyber Challenge Platform

The Open Cyber Challenge Platform (OCCP) [1] is being developed by the University of Rhode Island's Digital Forensics and Cyber Security Center under support from the U.S. National Science Foundation as an open source virtualization platform for cyber security educators and cyber challenge event organizers. The project seeks to increase the number of qualified students entering the fields of information assurance, cyber security, and digital forensics, and to broadly increase the capacity of U.S. higher education to produce professionals in these fields. [2] Goals of the OCCP are to be free, configurable, and extensible. These goals minimize the barriers to entry and help motivate the community to produce a shared base of cyber security education tools that utilize the OCCP.

The platform is designed to work with *Virtual Machines* (VMs) to simulate networked computers for a cyber challenge. Virtual machines are essentially a software emulation of a physical computer. These VMs run on a *hypervisor*, which manages a physical machine's resources to accommodate the virtual machine's resource requests. There are two main categories of hypervisors, often referred to as type one and type two. *Type one hypervisors* are installed directly on a physical

machine; whereas *type two hypervisors* are installed as an application on top of an operating system.

Virtualization eases some of the logistical concerns of deploying and running cyber challenges. Though the hardware requirements of the hypervisor still must be considered, it will likely require fewer physical machines to run a challenge with virtual machines than it would to run the same challenge without virtualization. Additionally, configuring a group of physical machines for a cyber challenge is difficult due to the financial cost, space requirements, and person hours required [3]. Furthermore, such a setup is inherently rigid in its configuration. Even slight modifications to the network topology potentially introduces new hardware and modifying the existing machines. Virtualization can make copying machines or changing the networking as simple as a few mouse clicks instead of relying on tools like Clonezilla [4] or FOG project [5] to clone hard drives.

While virtualization makes it easier to work with the machines involved in a cyber challenge, there are complicating elements to creating challenges that virtualization cannot not take away. For instance, whether virtual or physical, the machines must have an operating system installed. Then, each machine will still need additional configuration, such as the addition of user accounts, software, and other content. Some services may depend on the configuration of other machines and so great care must be taken to ensure that these elements match across the network. These problems are not solved by simply transitioning from physical machines to virtual ones.

The OCCP architecture has two core virtual machines known as the "Game Server" and "Administrative Virtual Machine". The Game Server concept is common with cyber challenges and is typically responsible for the scheduling and scoring of events. The Administrative Virtual Machine, or Admin VM, is a component

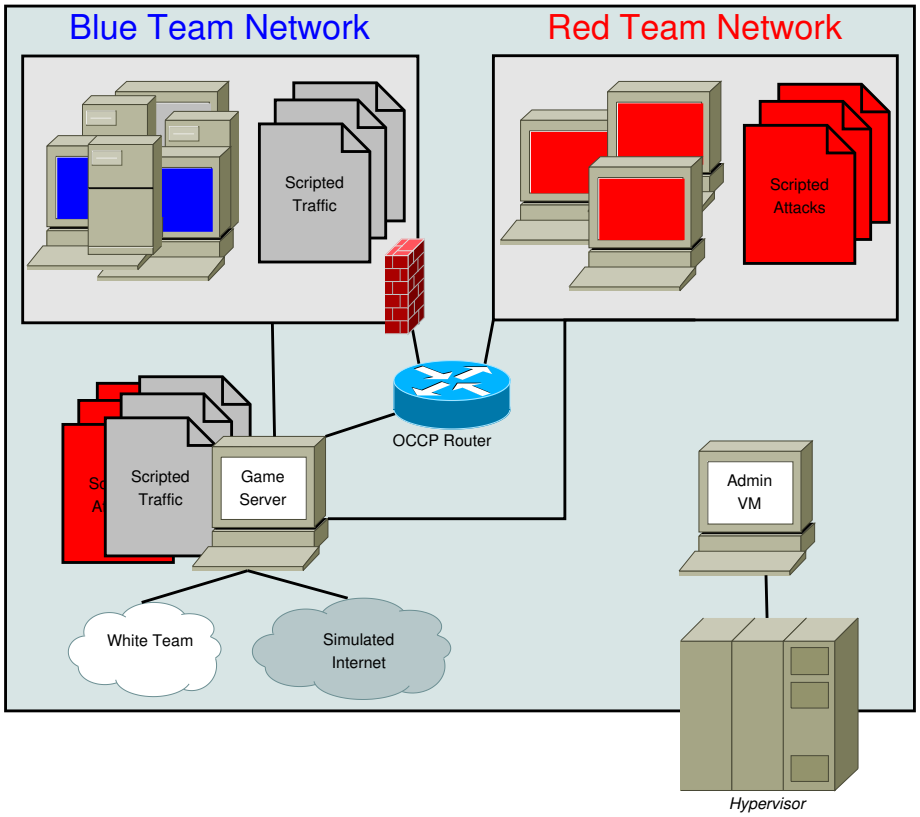
unique to the OCCP. It provides conveniences to the challenge operator by controlling their hypervisor(s) for them. The Admin VM is also able to facilitate the establishment of networking in the cyber challenge. The OCCP refers to the network involving the challenge's virtual machines as the *virtual scenario network* (VSN).

In general the OCCP is meant to be as flexible as possible so that a wide variety of challenges can be conducted with it. Typically the *Blue Team* is considered the defense, while the *Red Team* is considered the offense. In addition, to these two teams, there is also a *Grey Team* that produces miscellaneous network traffic, and a *White Team* that observes and scores the challenge. The Game Server can be used to script actions for the Red, Blue, or Grey Team.

Figure 1 on page 8 shows the general components of the OCCP. The figure shows that all of the VMs are hosted on a hypervisor that is controlled by the Admin VM. The Red Team can either be live participants, actions scripted by the Game Server, or both. The Blue Team's network consists of various servers and Blue Team participant's workstations behind a firewall. Not all challenges will use all of the team components, or even all of the OCCP's components. Regardless of what components are used, the VMs themselves still need to be made by the challenge creator.

The OCCP had an Admin VM, but it was only capable of manipulating VMs by controlling hypervisors. It was capable of checking for the existence of VMs, ensuring those VMs were connected to the correct virtual networks, and performing several other conveniences to the user like snapshotting the VMs before turning them on (snapshots allow VMs to capture a state which can be restored at anytime). It could also configure a small virtual machine called the *Router VM*, which is another component of the OCCP that essentially fulfills the role

Figure 1. OCCP General Concept



of an Internet Service Provider (ISP) by connecting smaller networks within the challenge. Even with such conveniences, the original Admin VM was not capable of creating the desired VMs if they did not already exist.

This thesis project extended the OCCP Admin VM and admin program to provide a mechanism to produce reusable virtual machines, making it easier for cyber security educators to share knowledge and reuse existing work. I will refer to this work as the *OCCP VM Builder* to distinguish it from previously completed work. The OCCP VM Builder encompasses all of my design including my extension of the Admin VM.

2.2 Target Audiences

The OCCP VM Builder considered three target audiences during its development.

- **Advanced Contributor**. This audience has the largest skill set and is accustomed to creating cyber challenges. They have a solid understanding of machine deployment, configuration, and the necessary elements of a cyber challenge. This audience can use and understand technical documentation for software they are not familiar with. They may also have experience maintaining production systems and using tools to support them like configuration management systems.

- **Basic Contributor**. This audience has less expertise than the advanced contributors, but is still familiar with the general work required to create a challenge. A Basic Contributor, given the description of a challenge, may be able to identify pieces that could be replaced or altered without destroying the functionality of the original challenge.

- **Basic Administrator**. This audience does not have the technical expertise of either of the Contributor audiences to produce a cyber challenge, but is capable of running an existing cyber challenge.

The OCCP VM Builder is designed to support all three audiences in creating and using cyber challenges within the scope of their abilities.

The Admin VM's extension allows the Advanced Contributors to share their work with the remaining audiences. The secondary audiences can produce functionally equivalent machines and reuse them by making configuration changes that, when done manually, is tedious and error prone. This project also provides challenge contributors with support mechanisms, such as those to produce randomized

usernames, passwords and SSH keys, which are material, that are common to most machine configurations, but needlessly time consuming to produce manually. The end result of the OCCP VM Builder is a tool that allows challenge creators the ability to share their efforts with others who have varying technical backgrounds while also easing some of the burdens of creating challenges.

2.3 NICE Challenge Project

After the work of this thesis was completed, a similar platform known as the NICE Challenge Project [6] was discovered. Although information about this project is limited at this time, the available documentation seems to suggest that it includes a component with a similar purpose to the tool created by this thesis. Specifically, their "environment modification engine" may make their challenges reusable by modifying content within them. However, since the project has not been released yet, it is difficult to determine how similar the tools are. According to their website, a beta release is scheduled for the beginning of 2015.

2.4 Related Technologies

This section will discuss the technologies related to this thesis project. Table 1 summarizes the information detailed in the subsections for each technology.

Table 1. Comparison of four existing technologies

	Puppet [7]	Ansible [8]	Vagrant [9]	Docker [10]
Provision				
VirtualBox	✗	✗	✓	✗
VMware ESXi	✗	✗	?	✗
VMware Workstation	✗	✗	$	✗
Configure				
Install Software	✓	✓	*	-
Create users	✓	✓	*	-
Modify files	✓	✓	*	-
Transfer content	✓	✓	*	-
Generally x-platform	✓	-	✓	✗
Distributable	-	-	✓	-

Legend

✓	Fully supported
$	Paid feature
*	Relies on third party
-	Partial support
?	May be in development
✗	Does not support

2.4.1 Puppet & Ansible

Puppet [7] and Ansible [8] are both Configuration Management Systems (CMS). They each can install software, configure the software, transfer files, create users, and more.

My approach to extending the Admin VM was to make use of a CMS as the mechanism to configure machines. Unfortunately, these CMSs are not the perfect solution as they are specifically designed for configuration management and not provisioning or deployment. Additionally, using these systems directly, would force all users to learn the respective system to create and use challenges. Despite this, a CMS can be incorporated in to the Admin VM to fulfill some of this project's goals.

When my initial research into Puppet and Ansible started, Puppet supported Windows, whereas Ansible did not. Since then Ansible now partially supports

Windows [11], but the support is still under development, this mature support for Windows led me to choose Puppet as the CMS used by the Admin VM.

2.4.2 Vagrant

Vagrant [9] is the most similar technology to what this project aimed to produce. Vagrant has the ability to provision virtual machines on several different hypervisors and can also make use of CMSs to configure machines. However, Vagrant has several drawbacks.

In order to provision a machine, Vagrant makes use of what it calls "providers". Out of the box Vagrant comes with a free VirtualBox [12] provider, but the VMware provider must be paid for [13]. Since one of the goals of this project is to not introduce a financial cost and since the OCCP Admin VM was already able to work with VirtualBox or VMware free of charge, I decided not to use Vagrant.

2.4.3 Docker

Docker [10] is a tool by which users can package software and its dependencies to run on a wide variety of Linux systems. This is potentially useful in the creation of challenges because of its modular nature, but there are several drawbacks. One drawback is that Docker will only work with Linux. While Linux training is a vital part of security education, networks are often made up of a diverse set of operating systems, not just Linux. Furthermore, Docker works by using Linux containers that share a common kernel. In some cyber challenges a shared kernel could produce unwanted results. For instance, a vulnerability that exploits the kernel could cause all other Docker applications sharing that kernel to be exposed unintentionally. Finally, Docker is a fairly new tool and the makers even say "Please note Docker is currently under heavy development. It should not be used in production (yet)" [14] For these reasons I did not use Docker.

List of References

[1] Digital Forensics and Cyber Security Center. "OCCP." Jan. 2014. [Online]. Available: https://opencyberchallenge.net

[2] R. H. Wagner, "Designing a network defense scenario using the open cyber challenge platform," *Open Access Master's Theses*, no. 73, 2013. [Online]. Available: http://digitalcommons.uri.edu/theses/73

[3] K. E. Stewart, J. W. Humphries, and T. R. Andel, "Developing a virtualization platform for courses in networking, systems administration and cyber security education," in *Proceedings of the 2009 Spring Simulation Multiconference*, ser. SpringSim '09. San Diego, CA, USA: Society for Computer Simulation International, 2009, pp. 65:1–65:7. [Online]. Available: http://dl.acm.org/citation.cfm?id=1639809.1639877

[4] "Clonezilla - about." June 2014. [Online]. Available: http://clonezilla.org

[5] "FOGUserGuide - FOGProject Wiki." June 2014. [Online]. Available: http://www.fogproject.org/wiki/index.php?title=FOGUserGuide

[6] "Solution — NICE Challenge Project." Nov. 2014. [Online]. Available: https://www.nice-challenge.com/

[7] Puppet Labs. "Puppet labs documentation." Feb. 2014. [Online]. Available: https://docs.puppetlabs.com

[8] Ansible Inc. "Ansible documentation." Feb. 2014. [Online]. Available: http://docs.ansible.com

[9] HashiCorp. "Vagrant documentation." Aug. 2014. [Online]. Available: http://docs.vagrantup.com/v2/

[10] Docker Inc. "Docker documentation." Aug. 2014. [Online]. Available: https://docs.docker.com

[11] Ansible Inc. "Windows support - ansible documentation." Aug. 2014. [Online]. Available: http://docs.ansible.com/intro_windows.html

[12] Oracle. "Documentation - Oracle VM VirtualBox." Jan. 2014. [Online]. Available: https://www.virtualbox.org/wiki/Documentation

[13] HashiCorp. "Vmware vagrant environments - vagrant." Feb. 2014. [Online]. Available: http://www.vagrantup.com/vmware

[14] Docker Inc. Apr. 2014. [Online]. Available: https://www.docker.io/learn_more/

CHAPTER 3

Methodology

This chapter describes the methodology I used to develop the OCCP VM Builder. It also describes the experiments used to validate that OCCP VM Builder met its goals. In Section 3.1, I describe the various elements in my design. Sections 3.2 and 3.3 discuss some implementation details of my design. Section 3.4 discusses building a VSN for use with my design and implementing a network defense scenario with it. In Section 3.5, I discuss an addition to my original design that came up during development which optimizes certain aspects of the new system. Finally, Section 3.6 discusses my testing methodology used to evaluate my work.

3.1 Establish a Basic Design

After reviewing existing technologies and the OCCP architecture, I discussed by findings with the OCCP Development Team and decided that re-configuring the Admin VM, installing additional services, and extending its code base would be the best approach to developing the automated configuration and deployment capability of the OCCP. The first step was adding another Network Interface Card (NIC) to the Admin VM and configuring it to perform Network Address Translation (NAT) routing. Then, I needed to install the Configuration Management System and several other services to facilitate the new setup process. In addition to configuring the Admin VM, I had to design the building blocks that Contributors would eventually use to create challenges with this revised OCCP. These building blocks were *Base VMs* and *Content Packs*, both of which will be introduced in later sections.

3.1.1 Choosing a Configuration Management System

My design called for a Configuration Management System, but did not name one explicitly. Though there were several to choose from, I ultimately went with Puppet as described in 2.4.1.

Another factor in my decision to use Puppet as the OCCP VM Builder CMS was the size of Puppet's user base. Puppet was not a new technology, so it had been used in production environments for years. There are many available resources regarding Puppet including articles, tutorials, and books. Additionally, there exists the Puppet Forge where official Puppet Labs developers, as well as, community members publish modules for others to download and use. This preexisting base of reusable work, availability of documentation and tutorials, and large user community played an important role in choosing Puppet. For those reasons, I felt it would be reasonable for someone that had never used Puppet before to get up to speed. Moreover, a portion of the target audiences may already be familiar with Puppet. Though, for some audiences knowing how to use Puppet was unavoidable in my design. Despite this, I worked to use Puppet in such away that the other audiences could use the OCCP VM Builder without having to know how Puppet worked.

3.1.2 Using the Configuration Management System

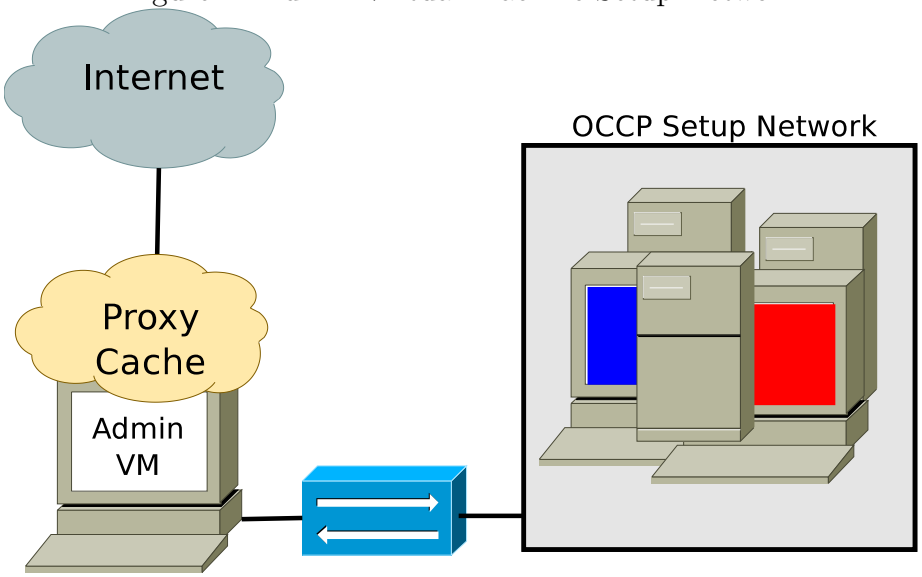
In order to use the Puppet CMS, Advanced Contributors would build their VSNs with *Content Packs*. Content Packs are essentially Puppet modules, and in fact, some Puppet modules can be used directly as Content Packs. When they cannot be used directly, it may be possible to write a Content Pack that wraps the Puppet module, making it suitable for use with this system. The major difference is that Content Packs are written explicitly for the OCCP, and respect OCCP *Configuration Phases*, which are described in 3.1.3. The Advanced Contributor

creates the scenario XML file to reference those Content Packs and to provide any configurable parameters to them. When a secondary audience reuses the Advanced Contributor's work, they do not necessarily have to alter the Content Pack to make changes to the scenario. Though there is nothing forcing the Advanced Contributor to design their Content Packs to be modular, my intent was to encourage them to design modular Content Packs. It is my expectation that eventually there would be enough modular Content Packs available, similar to the Puppet Forge, that Basic Contributors could alter a scenario by swapping out the original Content Packs for new ones. Furthermore, with more starting examples, Advanced Contributors could borrow from and extend existing Content Packs which would make it easier to produce new ones.

3.1.2.1 Setup Network

In order to facilitate the use of the CMS, I added the *setup network* capability to the Admin VM. This required the Admin VM to have two network interfaces. The first interface was connected to a network that could reach the Internet. The second was attached to the setup network. Since only the Admin VM and the VMs being configured would be attached to this isolated network, the Admin VM could have full control over it. This design has the Admin VM as the gateway/router for the other VMs, and as a DHCP server. By running the DHCP server, the Admin VM can assign a known address to each VM under its control. Acting as a router, the Admin VM would forward any traffic to its other interface in order to reach the Internet. Figure 2 shows the new setup. This figure also depicts a "proxy cache", which is described in 3.5. The establishment of the setup network made it possible to use the CMS, however I enhanced the process with the introduction of *configuration phases*.

Figure 2. Admin Virtual Machine Setup Network



3.1.3 Configuration Phases

Regardless of how a VM is setup, the time/date it was setup can play a factor in its final state. For instance, if you take two identical VMs and install software on one, but wait a month to install that same software on the other with a package manager, you can end up with two different versions. For cyber challenges this can prove problematic since particular versions may have vulnerabilities that the challenge was intending to make use of. Furthermore, configuring software and other details about a VM can be relatively quick compared to the time it takes to download and install the software. For these reasons, I decided to separate software installation from the rest of the VM's configuration and do them in two phases.

3.1.3.1 The Installation Phase

This phase is strictly for the installation of software and other configurations that depend on when that software installation occurs. Items configured in this stage should be considered static and not easily reconfigured without starting from

scratch. The Contributor creating the challenge should be the only person that applies this phase. The product of applying the Installation Phase is the state of the VM that will be shared when the Contributor publishes their challenge. This state is captured by the OCCP VM Builder as a VM snapshot called "phase1".

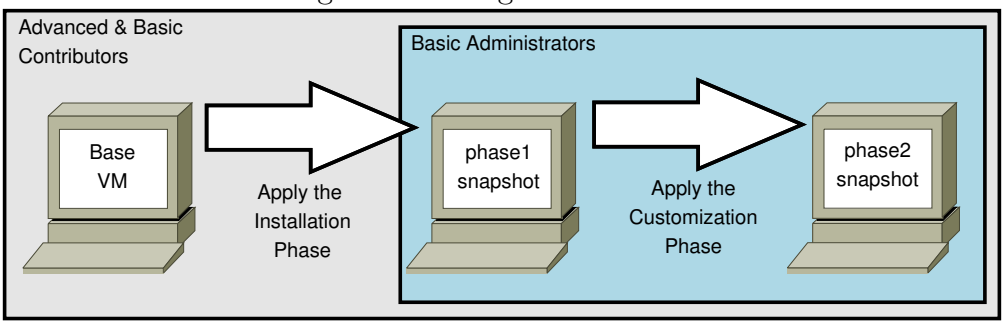
3.1.3.2 The Customization Phase

This phase is where the general configuration of the VM occurs. In this phase, customizations like software being configured, user accounts being added, passwords being set, and/or transferring files are performed. Items configured in this phase are designed to be dynamic and subject to change. No software should be installed during this phase. The only exception are hypervisor-specific tools such as VirtualBox guest additions, if they are required for smoother operation of the final VM. The result of applying this phase is a VM ready for use in the challenge. The OCCP VM Builder captures this state with a VM snapshot called "phase2".

3.1.3.3 Base VMs Configuration Process

A VM created by this process will have initially come from a *Base VM*. A Base VM is a VM that is preconfigured specifically for use with the OCCP VM Builder. In most cases, it is not configured beyond what is required for the OCCP VM Builder so that is can remain generic and flexible. The OCCP VM Builder transforms a Base VM into the final state during the Configuration Phases. A Base VM will have the Installation Phase applied to it only by the Contributor. From that state, all audiences can apply the Customization Phase. Once the second phase has been applied, the VM has reached its final state and is ready to use in the VSN. Figure 3 shows this process and the audiences that would be involved at each step.

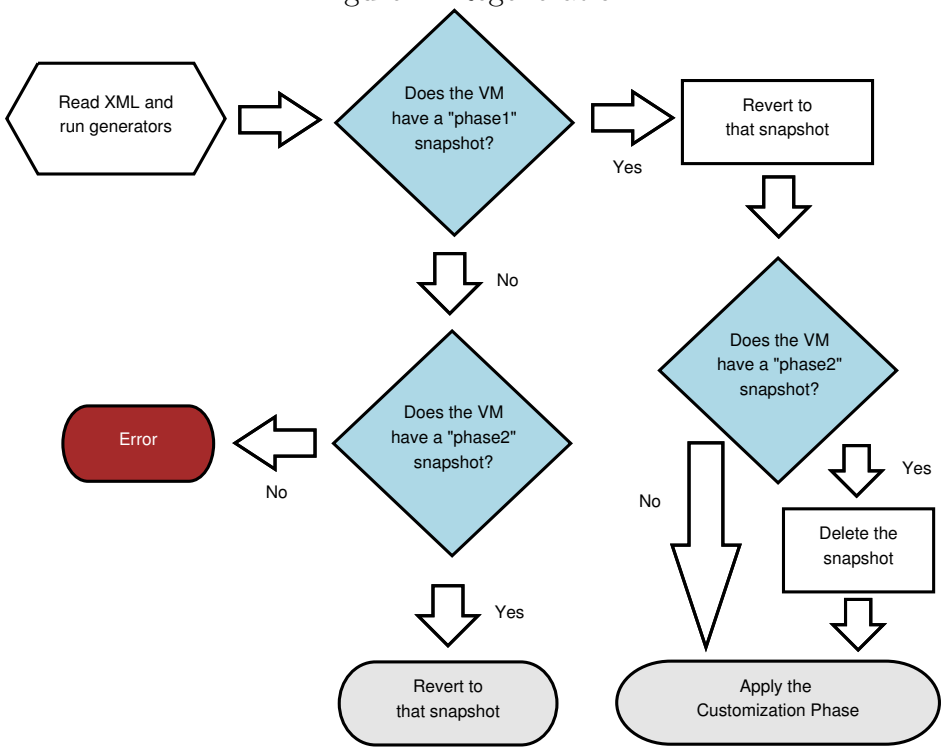
Figure 3. Configuration Phases



3.1.3.4 Regeneration

Because the Customization Phase is where all of the dynamic configuration occurs, changes can be applied by reverting to the *phase1* snapshot and applying the Customization Phase again. Instead of requiring users to do this by hand, I introduce a "regeneration" flag to the OCCP VM Builder. The regeneration process is summarized in Figure 4.

Figure 4. Regeneration



In order to maintain flexibility, my extension still allows Contributors to create

VMs by hand. The only additional requirement is that they must manually take a snapshot called *phase2*. This is because the OCCP VM Builder now considers the *phase2* snapshot as the state of the VM that is ready for use in a challenge. Since manually built VMs do not use Content Packs, there is no point to manually create the snapshot that would have been taken at the end of the Installation Phase. There would be nothing for the OCCP VM Builder to do between the *phase1* and *phase2* snapshots, therefore manually built VMs must not have a *phase1* snapshot. As shown in Figure 4, VMs configured by hand will simply be reverted to the manually taken*phase2* snapshot during regeneration. If, however, a VM has a *phase1* snapshot and is therefore eligible for regeneration, it will be reverted that snapshot. The VMs current *phase2* snapshot will be removed and the Customization Phase will be reapplied. The process of regeneration is most useful when the scenario XML file makes use of *OCCP Variables* and *OCCP Generators* which are described in 3.1.4.

3.1.4 OCCP Variables and Generators

3.1.4.1 Variables

My OCCP VM Builder also added variables to the scenario XML file specification. The variables allow for a single editable location to affect one or more places throughout the scenario XML file. For instance, a variable could hold a username that is used throughout the rest of the scenario file. All of the target audiences should be able to update a variable with ease since it only involves changing the value in one location. Furthermore, before this introduction of variables, they would have had to update their scenario XML file by hand in a similar fashion.

3.1.4.2 Arrays

In order to provide greater functionality to variables, I also introduced variable arrays. The arrays can contain multiple elements per variable which can then be

referenced by their index (location) in the array. Though arrays are more complex than a simple variable, editing an element is no more complex than editing a single variable. Arrays can provide greater quantities of related data more conveniently than multiple variables to Content Packs. An example usage of an OCCP array would be to hold multiple usernames and then reference them throughout the scenario file with their index, or to pass them to a Content Pack all at once.

3.1.4.3 Generators

In addition to these variables, I also introduced generators that fill in the content of the variables and arrays. These generators can take parameters that affect their output. My design called for three proof of concept generators a username, password, and SSH key generator.

3.1.4.4 Username Generator

The username generator reads from a comma separated value file (CSV) that contains the first name, last name, and a username on each line, to produce randomized usernames. The generator randomly picks lines from this file and creates variables for each of the values. The user can optionally specify a count to generate more than one set with the guarantee that they will not get duplicate usernames provided there are no duplicate entries in the CSV file. When using a count greater than one, the variables generated will be arrays.

3.1.4.5 SSH Key Generator

The SSH key generator produces the public and private parts of an SSH key in separate variables. Like the username generator, a count may also be specified to create several key pairs at once as arrays. Optionally, a password may be provided to the generator to encrypt the private key.

3.1.4.6 Password Generator

The password generator is the most complex of the three generators that I built because it can take a variety of parameters. The generator's basic functionality produces the encrypted form of a plain text password, which may be provided to it as a parameter or randomly generated. One plain text password may be provided directly, or a file containing one plain text password per line may be provided for the generator to choose from at random. The user can provide additional parameters, such as length and type, for the generator to use to create random plain text. The user may specify SHA256, SHA512, or MD5 as the encryption algorithm for greater flexibility. Like the username and SSH key generator, a count can be specified but will be ignored if a plain text password was provided directly.

3.1.4.7 Summary of Variables and Generators

Variables and generators are important to regeneration because, in addition to properly reverting the VMs and reapplying Phase Two, generators will regenerate their content. Depending on how the variables and generators were used, the result is a functionally equivalent VSN, but with slightly different content. One use case for this would be an instructor teaching multiple sections of the same class and wishing to have the same environment, but different content for each section. This would prevent students exposed to the challenge before other sections from being able to pass along the passwords or flags that they had discovered. Furthermore, an instructor could conduct a challenge early on in the course and then again at the end of the course, varying the content so that it is not immediately obvious to students that they are participating in the same challenge as before.

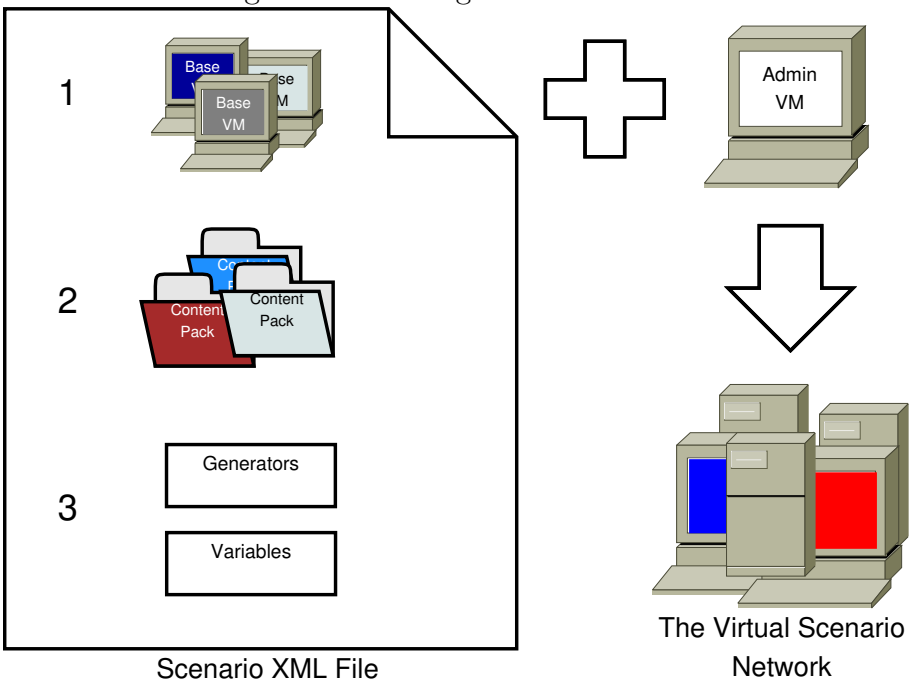
3.1.5 Reports

With the addition of dynamic content from generators and variables, there needed to be a mechanism to collect the generated information when the Admin VM was done building the VSN. To accomplish this I introduced the *report* XML tag to the scenario XML file specification. This tag allows the Contributor to specify any text they wish and reference any OCCP variable in the scenario XML file. The OCCP VM Builder stores all the reports in a predictable location and makes them available to Content Packs during setup. A report could be as simple as a list of variables and their values, or detailed instructions that can contain the dynamic content from variables. A Content Pack can transfer the report to a VSN VM, which would be useful to distribute instructions to player VMs.

3.1.6 The Process of Creating a New Challenge

The following process of creating an OCCP challenge with the OCCP VM Builder is illustrated in Figure 5. First, the Contributor picks or creates the Base VMs that their VSN will require. Their choice in Base VM determines what operating systems will be present in the final VSN. Next, the Contributor chooses or creates Content Packs that will configure the Base VMs to produce the final VM for the VSN. Next, the Contributor writes a scenario XML file that describes their VSN. They can use variables and generators to make this description more dynamic and configurable. The scenario XML file can pass these variables to the Content Packs, which will cause the Admin VM to configure their VMs automatically. Finally, the Contributor uses the OCCP VM Builder to build their VSN and run their challenge. When they are satisfied that the VSN is ready for others to use, the Contributor uses the OCCP VM Builder to package up all the required files to distribute their challenge.

Figure 5. Challenge Creation Process



3.1.7 Export Mode

Once a challenge has been built and is ready to be shared with the OCCP community, the Contributor can make use of the *export mode*. Most hypervisors provide export functionality that will prepare a VM to be brought to another hypervisor. With OCCP VM Builder's introduction of the Configuration Phases and dynamic content from generators, additional consideration is needed before exporting the VMs.

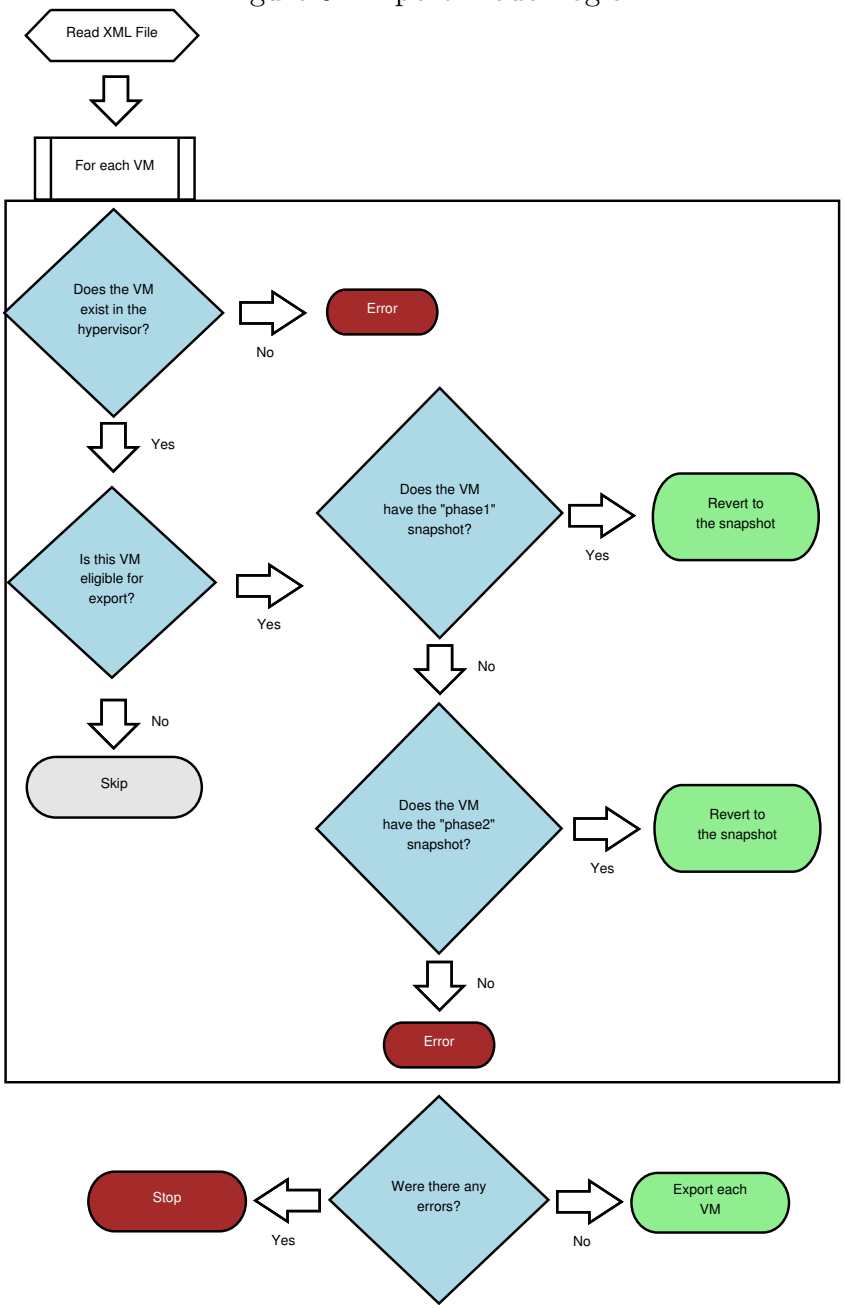
Since the export mode can take significant time to complete, the OCCP VM Builder first checks for any issues that could cause the process to fail later on. The following logic is also illustrated in Figure 6 on Page 27. First, the program ensures that all the VMs in the scenario XML file exist and that they are eligible for export. Some VMs do not make sense to export and are therefore ineligible. For instance, the OCCP Router VM, which is a standard part of each OCCP installation, is not available for export because every other OCCP installation will already have all of the elements required to reproduce it without importing it. Next, the program determines if the VM is capable of regeneration by the existence of a *phase1* snapshot. If the VM is capable, the program will revert the VM to the *phase1* snapshot and move on to the next VM. If the VM does not have a *phase1* snapshot, the program will try to revert to the *phase2* snapshot. After the OCCP VM Builder is finished checking each VM, if there were no errors it will begin the export process. If there were errors, it will report all the errors that it encountered to the user and stop. Reporting these errors at this point saves the user from having to discover them as they happen which could be several minutes or hours in to the process depending on their hypervisor and the size of the VMs.

The reason the export process prefers *phase1* snapshots is because that is the only state from which the VM can be re-configured. For most hypervisors, the

export function cannot retain snapshots, they can only export one state of the VM. Because of this, it is important to export a regeneration capable VM at this state to retain that capability. If it were to export a regeneration capable VM at the state produced by the Customization Phase, any effort the Contributor had to make the VM re-configurable would be lost. This is because the exported VM would not have the option to revert to the state before the Customization Phase. Fortunately, due to the design of the Customization Phase, rebuilding the downloaded VSN is an automatic process and requires no intervention by the user. This automation ensures that, while more technical expertise may have been required to originally create the challenge, even the Basic Administrator audience is capable of running the challenge.

Aside from the exported VMs in their proper states, there are additional files that must also be provided when sharing these OCCP challenges. Among these extra files are the scenario XML file, the Content Packs, and any files on which the Content Packs depended. I ensured that these additional files are included in the scenario package with the exported VMs. As a result, the package contains all of the files required by the OCCP VM Builder to import and rebuild the VSN on a foreign OCCP setup.

Figure 6. Export Mode Logic



3.2 Build a Base Virtual Machine

My design for a Base VM was a generic install of an operating system and the bare minimum additional configurations required by Puppet to function. Generally, this meant having a network interface that would be brought up automatically and use DHCP to configure its networking settings, an administrator account, and a way to run commands remotely.

My first Base VM was an Ubuntu 14.04 LTS Server install. When prompted for a hostname I chose "basevm" since it was descriptive and could easily be searched for if anyone wished to remove traces of being a Base VM. The default installer also asks for an initial username and password, which I choose to be occpadmin:0ccpadmin. During the package selection I opted to install OpenSSH server and nothing else. After the install finished, I used the occpadmin account to set root's password to "0ccpadmin" and then deleted this account. Next, I installed the Puppet client via the package manager. I then disabled the agent service from running automatically and configured it to allow the agent to be run by hand.

Next, I had to ensure that the NIC setup would not be preserved because the Base VM could end up on a number of different hypervisors which will use different MAC addresses for the interface and potentially emulate different hardware for the NIC. If the Base VM's operating system were to remember the NIC being used during my setup, it might not work correctly for another user. My solution to this problem was to create a blank file that udev, a Linux device manager, would use to create persistent rules for NICs. Normally, this file would have methods for generating the rules but leaving it blank has the effect of making the VM forget the NICs every time.

Lastly, I ensured that a user could login as root with a password through the

SSH server by editing the configuration file. By default Ubuntu runs SSH at boot and uses DHCP to configure the first interface, so there were no additional tasks to perform to enable remote login. Using the grep command I attempted to remove artifacts about the environment in which the Base VM was created. Having IP addresses and other artifacts in files could cause frustrations for challenge participants who encounter them since they are not actually part of the challenge and may appear suspicious. I also deleted my command history so the Base VM would not have artifacts of creating it left over.

Using the same general procedure, I was also able to create an Ubuntu 12.04 LTS Server and a Kali 1.0.8 Base VM. Each required some variation to the steps followed for the 14.04 base but helped validate the concept of a Base VM. Though my design called for a very generic installation, the concept of a Base VM could be modified. For instance, nothing prevents a Contributor from creating a more specialized Base VM as long as the basic expectations the OCCP VM Builder has about it are not broken. These expectations are that the Base VM will use DHCP, has the ability for remote admin login, and that Puppet is installed. These expectations allow the OCCP VM Builder to communicate with the Base VM and apply the Configuration Phases. This design leaves the Base VM concept open to change as the OCCP grows.

3.3 Install and Configure the Configuration Management System

3.3.0.1 HTTP Server

Not long after installing and attempting to use Puppet, I discovered that the built in web server, WEBrick, would not be suitable for this application. Though VSNs are unlikely to be made of more than a few dozen VMs, there is a good chance that these VMs will make use of the Puppet master server concurrently in my setup procedure. This is because the VMs will not use the agent daemon to check in

periodically since the OCCP VM Builder would not have precise control over when the VMs performed the check in. Instead, the OCCP VM Builder runs the Puppet agent exactly when it is ready for the VMs to check in. Since the program uses multiple threads, and much of the work happens on the individual VMs, it is possible that the OCCP VM Builder will request two or more VMs to check in at the same time. This was problematic because the implementation with WEBrick is unable to handle concurrent connections properly. [1] Fortunately Puppet master can be run as a passenger under Apache [2]. Apache is a more robust web server that is capable of handling greater loads. Passenger is a technology that can run some Ruby applications like Puppet under the web server. By running Puppet with Passenger, Puppet gets the benefits of Apache's HTTP server's robustness. The Puppet master is thus able to handle more requests and do so concurrently. This speeds up the deployment and configuration by allowing the OCCP VM Builder to configure multiple VMs concurrently instead of sequentially.

3.3.0.2 SSL Certificates

Another hurdle to using Puppet was its use of SSL certificates. Communications between agents and the master are secured with SSL in Puppet, which allows for secure communication and trust that the other party is who they claim to be. This use of SSL certificates prevents a bad actor from sniffing traffic or impersonating a client to gain information about a Puppet catalog or manifest. This system works well in environments where agents are added by IT staff and will remain under the master's control for their life cycle. However, OCCP VSN VMs are not being added by a human being, nor is their life cycle particularly long. Furthermore, they will be shared with another Puppet environment if the author chooses to share their challenge.

3.3.0.3 Agent & Master Communication

Typically, a new client will perform the following procedure when attempting to contact the master. First, the client will generate its own private key if it doesn't have one. Next, it will request a copy of the *Certificate Authority* file from the master if it doesn't already have it. Next, the client will attempt to retrieve its signed certificate from the master. If it doesn't get one, it will determine if it has already generated a *Certificate Signing Request*, and if not, it will do so. With default settings, the client will have to wait until a user completes the signing request on the master. In normal environments, the user would check the fingerprint of client's certificate and ensure it matches the request received on the master before signing the request. Due to the nature of the OCCP setup environment, the manual signing of certificates is unfavorable. If the setup process required human intervention to function, it would increase the complexity for OCCP users. Not only would they have to be involved in the setup process, they would have to know how Puppet's certificate system worked. To automate this process, the OCCP Puppet master is configured to automatically sign every Certificate Signing Request it receives. While this solves the certificate signing issue, other issues still remained.

Because of the SSL system, bringing a VM from one Puppet infrastructure to another with a different master would cause problems. One of the features of using SSL is that the agent and master can verify the other party is who they claim to be. If the agent is brought to a new environment with a new master, its certificates will not match. To prevent this and increase the ability to share OCCP material amongst the target audiences, the Admin VM needed to ensure that all certificate material was removed. Before the Admin VM program gets an agent to check in, it first revokes any certificate for that VM. Secondly, at the end of each configuration

phase, it will remove certificate material on the agent. Unfortunately, this means that the agent will need to go through all the work of an initial check in, but it will never encounter a mismatched certificate. This allows Contributors to share their OCCP challenge with other OCCP users without issue.

3.3.0.4 Time Synchronization

A final issue with the SSL system was that all VMs involved need closely synchronized clocks. A majority of the time during testing the master and agents clocks were close enough so that the certificate infrastructure worked. However, there were occasional issues where some VMs would have a clock that had drifted by several hours. The reasons for their drift were unclear but may have been caused by the hypervisor. To get around hypervisor specific time issues, a Network Time Protocol (NTP) server was added to the Admin VM. When agents synced their clock with this server it guaranteed that the master and agent clocks would be synchronized, allowing the SSL infrastructure to work.

3.4 Design a Virtual Network for a Cyber Challenge

Since one of the motivations behind my work was to provide tools and mechanisms for a community to build challenges and contribute back, there needed to be some initial content from which the community could start. Since network defense is a popular challenge type that can teach and assess valuable cyber security skills, it was a natural choice to provide as an example. However, before I created the network defense challenge, I first created a generic reference VSN that modeled a typical small business network.

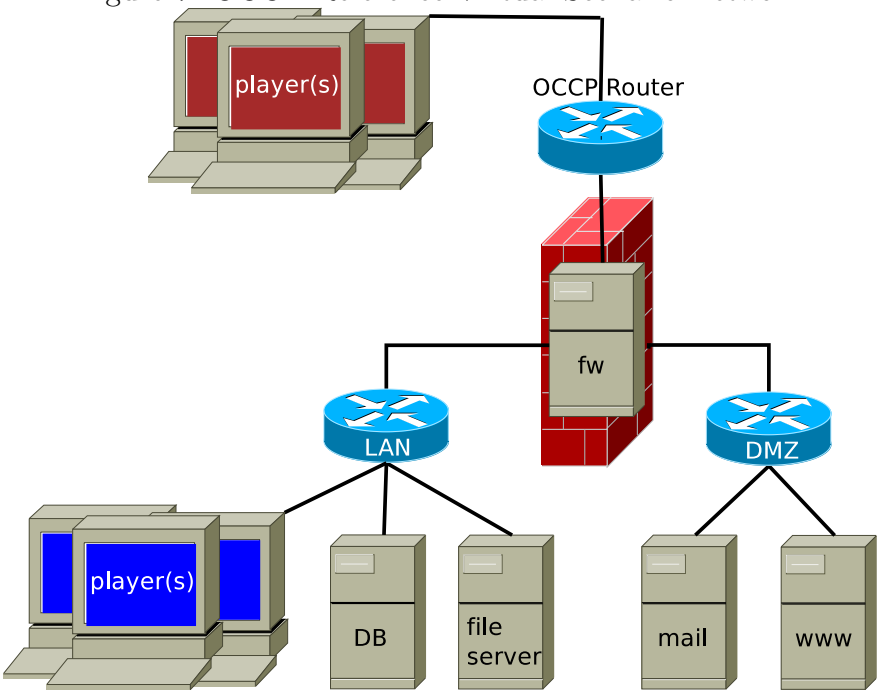
3.4.1 The OCCP Reference Virtual Scenario Network

The reference VSN includes several VMs running services typical of small business networks. While its generic configuration is not suitable for all challenges,

it gives a working example that can be modified to create different VSN topologies.

Figure 7 shows the reference VSN's network topology. At its core are five VMs: a firewall, a database, a file server, a mail server, and a web server. Additionally, there can be Red Team and Blue Team player desktops. The firewall VM acts as the border firewall for the network, but also acts as a router separating two network segments for the LAN and DMZ. This firewall VM also connects to the OCCP router. The database and file server VMs are connected to the LAN and are considered internal, while the mail and web server VMs are connected to the DMZ and are considered public. The firewall VM also provides DHCP and DNS services to the LAN and DMZ. This generic design is not appropriate for use as it is, but instead provides the basic elements to facilitate creation of scenarios through easy modification of its base components.

Figure 7. OCCP Reference Virtual Scenario Network



3.4.2 Network Defense Virtual Scenario Network

To create the Network Defense scenario I started with the creation of the Content Packs. Since the only Content Packs that existed were the ones that I wrote for the reference VSN, I started by copying them. This resulted in the same network topology as the reference VSN. Because there were many services used in the reference, there was a lot of flexibility for the types of elements to include in a network defense context. Basing my design loosely on some of the concepts taught in URI's cyber security courses, I decided to include a vulnerable web application. After searching Rapid7's exploit database [3], I discovered a vulnerable web application that was a good fit for exercising skills taught in URI's classes. Based on those skill sets, I did not expect participants to patch or remove the application, but there were steps they could take to mitigate the threat by fixing file permissions and fixing intentionally poorly-written PHP on another page of the website. That page would connect to the database with root's credentials which is a poor security practice. Another vulnerability that I added was a weak SSH configuration on the publicly facing servers. Due to the nature of this weakness, there were several ways for the participant to mitigate the threat. For instance, they could modify the SSH configuration on each exposed VM, configure local firewall rules, or configure the firewall VM to mitigate the threat. Lastly, I intentionally configured the firewall to be too permissive with some rules for traffic between the DMZ and LAN. Instead of only allowing the essential traffic from the DMZ to the protected LAN, the firewall allowed some non-essential traffic through as well. The idea is that the participant could revise the firewall rules to be more secure by further restricting the allowed traffic. Another security weakness that was already present in the reference VSN was the use of anonymous FTP on the file server. The only "protection" the file server has is its network location and that the computers

that can access it are supposed to be trustworthy. This is a poor security model with many ways that a participant could mitigate it. For instance, they could try to prevent an attacker from penetrating the LAN by securing the firewall or they could implement an authentication system for accessing the file server. Ultimately, the goal is that the participant should recognize that FTP is potentially dangerous and be able to explain why another technology would be a better choice for the file server.

Overall, if the participant did nothing to defend their network, an attacker could start by exploiting the vulnerable web application. Through that exploit the attacker could discover root's credentials to the database in the poorly-written PHP script. With root's credentials they could dump all the databases, one of which has email account names and passwords. After some password cracking and due to inappropriate firewall rules, the attacker could pivot from the web server to the database server located in the LAN. From this network position, the attacker has access to all of the network segments.

To create this challenge I was able to reuse, without modification, six of the ten reference VSN's Content Packs. I was then able to reuse, with slight modification, the remaining Content Packs. I did have to create an additional Content Pack which added smaller configuration details like user accounts and other miscellaneous configurations. As more challenges are built in this way, there will be more Content Packs to choose from, which should ease the burden of composing new challenges.

3.5 Incorporating a caching proxy

Though not part of my original design, the need for a caching proxy that could expedite the downloading of software was shown during the development of the OCCP VM Builder. When developing and testing the Installation Phase

components of the Content Packs, I was downloading the same packages each time. Instead of going out to the Internet to download these packages several times, a caching proxy is capable of downloading it once and serving requests for that content locally. So, I added the Squid Proxy Server [4] to the OCCP VM Builder. As seen in Figure 2, the VMs in the setup network have all of their traffic routed through the OCCP VM Builder's Admin VM. The caching proxy examines their requests and determines if it already has the content that is being requested. If it does, it serves the content itself. If it doesn't have the content in its cache, the proxy fetches the content and caches the content so that it may serve the content locally the next time that the content is requested.

3.6 Perform Testing and Evaluation

This section discusses the experiments I used to evaluate whether my goals were met.

3.6.1 Experiment 1: Can the extended Admin VM configure VMs for the OCCP, and how well can it do so?

This experiment showed that the OCCP VM Builder's Admin VM can configure VMs for the OCCP and compared it with manual configuration. I implemented the Network Defense scenario both manually and with the OCCP VM Builder. I compared the processes by recording the number of VMs logged into, the commands run, and the files edited. I also used "time to complete" as a measure of ease of use and complexity.

3.6.2 Experiment 2: Can the extended Admin VM produce distributable VMs?

This experiment showed that the OCCP VM Builder's Admin VM can not only provide the files necessary to distribute virtual machines that it creates, but also that it can receive those files and reproduce the functionally equivalent VMs.

First, I used the Admin VM to export the VMs of the Network Defense scenario and then used it import them in to a different hypervisor also configured for the OCCP. The second hypervisor had its own installation of the OCCP VM Builder Admin VM. I verified that the export and import process produced a functionally equivalent network by running the challenge. Success was determined by an error-free run of the challenge.

3.6.3 Experiment 3: Can the extended Admin VM produce reusable VMs for the OCCP, and how effective is this new method?

This experiment showed that the OCCP VM Builder's Admin VM is capable of producing reusable VMs by reconfiguring existing virtual machines. I first updated my original challenge design by changing all usernames, passwords, SSH keys used, and replaced web page content. I then compared the process of using the Admin VM to perform these reconfiguration operations versus doing them manually by again recording the number of VMs logged into, commands run, and files edited. Time to complete was also used in the comparison.

3.6.4 Experiment 4: Can the username generator generate content properly?

This experiment showed that the username generator can produce content that functions correctly on VMs created by the OCCP VM Builder's Admin VM. I wrote scenario files that used the username generator in the following ways:

- Specified a properly-formatted name CSV file and a name count of one. If functioning correctly, this would produce three variables, one with the first name, one with the last name, and one with the username from a random line in the CSV file.

- Specified a properly-formatted name CSV file and a name count of five. If functioning correctly this would produce three arrays containing five ele-

ments, one with the first names, one with the last names, and one with the usernames. Since there were no duplicate lines in the CSV file, the outputs should not contain duplicates.

- Specified a properly-formatted name CSV file and a name count greater than the number of usernames in the CSV. If functioning correctly, this would produce an error.

- Specified a non-existent name CSV file and a name count of one. If functioning correctly, this would produce an error.

- Specified a non-existent name CSV file and a name count greater than one. If functioning correctly, this would produce an error.

3.6.5 Experiment 5: Can the password generator generate functional content for challenges?

This experiment showed that the password generator is not only capable of producing content, but also that the content produced is functional in VMs produced by the OCCP VM Builder's Admin VM. By providing different sets of parameters to the generator, then assigning its output to users, I tested that I could successfully login with the expected password from each generated case.

3.6.5.1 Algorithm Tests

This series of tests ensured that the algorithms are producing the correct content for the encrypted version of the password. Each output was assigned to a different user. A successful login for each user determined that the content generated was consistent with the algorithm specified. I employed these variations:

- Set the password to "password" and set the algorithm to "MD5"

- Set the password to "password" and set the algorithm to "SHA256"

- Set the password to “password” and set the algorithm to “SHA512”

3.6.5.2 Additional Parameter Tests

This series of tests ensured that the generator can produce the plain text password correctly from the given parameters. Visual inspection confirmed that the results match the parameters provided. The count was set to 5, and the length set to 25 for each test. I varied the type parameter among:

- “ASCII” - If functioning correctly, this would produce an array of five passwords, twenty five characters long.

- “Alpha” - If functioning correctly, this would produce an array of five passwords, twenty five characters long containing only alphabetic characters. Any other characters would indicate failure.

- “AlphaNumeric” - If functioning correctly, this would produce an array of five passwords, twenty five characters long containing only alphabetic and numeric characters. Any other characters would indicate failure.

3.6.5.3 Password Pool File Correctness Tests

This series of tests ensured that the generator can produce content from a specified pool of plain text passwords correctly. To accomplish this, I performed the following:

- I used a valid password pool file, and specified a count of one. The expected output was a variable containing the plain text password chosen, and one containing its encrypted version.

- I used a valid password pool file, and specified a count of four. I expected an array containing the plain text passwords chosen, and another array containing the encrypted versions.

- I used a valid password pool file, and specified a count greater than the number of passwords available in the pool file. I expected an error for this test.

- I specified a non-existent password pool file, and a count of one. I expected an error from this test.

- I specified a non-existent password pool file, and a count greater than one. I also expected an error from this test.

3.6.6 Experiment 6: Can the SSH key generator generate functional content for challenges?

This experiment showed that the SSH key generator can produce content that is functional in VMs created by the OCCP VM Builder's Admin VM. I used the generator to produce the SSH keys and tested them in VMs configured to allow SSH key authentication. A successful login determined the success of this experiment. I produced a key pair without a password, and one with a password of "password".

List of References

[1] Puppet Labs. "Configuring a puppet master server with passenger and apache." June 2014. [Online]. Available: https://docs.puppetlabs.com/guides/passenger.html

[2] The Apache Software Foundation. "Documentation: Apache http server - the apache http server project." June 2014. [Online]. Available: http://httpd.apache.org/docs/

[3] Rapid7. "Vulnerability & exploit database — rapid 7." June 2014. [Online]. Available: https://www.rapid7.com/db/modules/

[4] The Squid Software Foundation. "squid : Optimising web delivery." June 2014. [Online]. Available: http://www.squid-cache.org/Doc/

CHAPTER 4

Findings

This chapter discusses the results from the experiments that are listed in 3.6.

4.1 Experiment 1: Can the extended Admin VM configure VMs for the OCCP, and how well can it do so?

This experiment set out to show that the OCCP VM Builder's Admin VM was capable of configuring VMs for the OCCP. Since there was no other solution before the OCCP VM Builder, the only other comparison point was configuring VMs manually without the OCCP VM Builder's Admin VM. The two processes are inherently different which made comparing them difficult. As per the experiment laid out in Section 3.6.1 I kept track of the number of commands run, VMs logged into, files edited, as well as the time spent. These findings are summarized in Table 4.1.

Table 2. Experiment 1 metrics

	Manually	With Admin VM
No. commands run	133 Commands	1 Commands
No. VMs logged into	6 Machines	0 Machines
No. files edited	45 Files	11 Files
Time to complete	8 Hrs 13 Mins	1 Hrs 35 Mins

4.1.1 Summary

The results of this experiment show that the OCCP VM Builder's Admin VM can in fact configure VMs for the OCCP, however the results do not indicate which method is more valid. Had there not been a reference network and content to start from, the amount of time and number of files edited would have been much larger for the Admin VM method. Likewise, using different commands or techniques could affect the number of commands and files edited but still produce roughly

the same virtual machine configuration. What can be said is that the Admin VM method is more easily shared and extended than the manual method as a result of the new capabilities provided by the OCCP VM Builder's Admin VM.

4.2 Experiment 2 Results: Can the extended Admin VM produce distributable VMs?

This experiment was designed to show that the OCCP VM Builder's Admin VM could produce the files required for another OCCP VM Builder Admin VM to reproduce a functionally equivalent VSN. In other words, could a Contributor use their OCCP VM Builder Admin VM to export their VSN and provide it to another to build and use? The results of this experiment showed that the OCCP VM Builder could produce distributable VMs. The OCCP VM Builder collected all of the necessary files that another OCCP VM Builder would need to produce a functionally equivalent VSN. The existence of the scenario XML file, Content Packs, pack dependencies, and exported VMs were enough for another OCCP VM Builder Admin VM instance to build the VSN to a state ready to conduct the challenge as the Contributor intended. After bringing the archive created by my first OCCP VM Builder to the second, I was able to deploy and launch the challenge on a different OCCP instance. Each VM in the VSN was accounted for and was configured as expected. This experiment validated that the VMs could be used by another OCCP VM Builder's Admin VM that had not initially created them.

Unfortunately, the export and import process can take a significant amount of time to accomplish. There are a number of different factors contributing to this, many of which will vary greatly depending on the OCCP configuration and hypervisors involved. Though the process can take a significant amount of time, the Contributor should only ever have to export once. Likewise the end users of

the challenge will only have to import once. Regeneration after the first deploy of a scenario is much faster because there are no VMs to import, which eliminates a large portion of time required in the initial deployment.

4.3 Experiment 3: Can the extended Admin VM produce reusable VMs for the OCCP, and how effective is this new method?

This experiment was designed to show that the OCCP VM Builder's Admin VM is capable of creating reusable VMs by changing the original content. In Experiment 1 I created my scenario XML file to match the usernames and passwords that I had used in the manual setup. Here in Experiment 3, I instead used generators to create the usernames and passwords for the seven user accounts, postfix's database password, local root password, and MySQL root password. Though the content that was generated did not match the content I used in the manual reconfiguration, the VMs are functionally equivalent. I collected the same metrics from Experiment 1 and summarized them in Table 4.3.

Table 3. Experiment 3 metrics

	Manually	With Admin VM
No. commands run	103 Commands	1 Command
No. VMs logged into	6 Machines	0 Machines
No. files edited	7 Files	2 Files
Time to complete	1 Hr 7 Mins	8 Mins

4.3.1 Summary

The OCCP VM Builder's Admin VM was able to produce VMs whose content differed from their original configuration. Table 3 compares the effort required to reuse the VMs with the OCCP VM Builder versus the manual method. Though not shown in the table, there are additional benefits to using the automated method. By using generators, the regeneration flag can produce functionally equivalent VMs with new content with just a single command. Moreover, there is no chance

of mistyping or forgetting to change something on any of the VMs. This would seem to suggest that the OCCP VM Builder is an improvement over manually creating VMs in terms of reuseability.

4.4 Experiment 4 Results: Can the username generator generate content properly?

The results of this experiment show that the username generator functions correctly. Before beginning the experiment, I created a CSV file with fifty lines of the form "first name, last name, username" with no duplicates. This is the file that I used for tests requiring a valid CSV file.

The following XML fragment caused the generator to produce the expected result of creating three variables for the first name, last name, and username.

```
<var name="test1" generator="username">
    <param name="names">validNames.csv</param>
    <param name="count">1</param>
</var>
```

Table 4. Username generator test 1 output

Variable Name	Variable Value
test1_first	Sherilyn
test1_last	Lundahl
test1	slundahl

Due to this result, this test was successful.

The following XML fragment caused the generator to produce the expected result of creating three arrays for the first name, last name, and username, each containing 5 values.

```
<var name="test2" generator="username">
    <param name="names">validNames.csv</param>
```

```
    <param name="count">5</param>
</var>
<network label="companydmz" />
```

Table 5. Username generator test 2 output

Array Name	Values
test2_first	Louisa, Mayola, Isis, Regena, Carli
test2_last	Crete, Courtemanche, Tsao, Delp, Liggins
test2	lcrete, mcourtemanche, itsao, rdelp, cliggins

This test produced the expected number of arrays, each containing the expect number of elements. Furthermore, no duplicates were generated. As such, this test was considered successful.

The following XML fragment caused the generator to raise an exception as expected. The number of requested usernames exceeded the number of available usernames.

```
<var name="test3" generator="username">
    <param name="names">validNames.csv</param>
    <param name="count">100</param>
</var>
```

Since there were only fifty usernames available in the CSV, requesting one hundred caused an expected error. As such, this test was successful.

The following XML fragment caused the generator to produce an error as expected. The CSV file specified did not exist and therefore could not be used to generate any usernames.

```
<var name="test4" generator="username">
    <param name="names">invalidNames.csv</param>
    <param name="count">1</param>
```

```
</var>
```

Since the admin program was unable to locate the CSV file it produced an expected error. As such, this test was successful.

The following XML fragment caused the generator to produce an error as expected. The CSV file specified still did not exist and the count parameter should have had no affect on the production of an error message.

```
<var name="test5" generator="username">
    <param name="names">invalidNames.csv</param>
    <param name="count">5</param>
</var>
```

Since the admin program was unable to locate the CSV file it produced an expected error regardless of the count parameter provided. As such, this test was successful.

4.4.1 Summary

Because each of the tests was successful, overall the experiment was a success. The experiment set out to prove that the username could generate content properly which it has so demonstrated. This generator has been shown to be functioning as intended and is suitable for use by challenge Basic Contributors.

4.5 Experiment 5 Results: Can the password generator generate functional content for challenges?

Due to the multiple functions of this generator, this experiment's tests were broken down in to different categories. Each category tested a smaller unit that, when combined, ensured the correctness of the overall generator.

4.5.1 Algorithm Tests

Before beginning these tests, I wrote a Content Pack that would configure one VM to set the generated passwords as user passwords on the it. The following

XML fragment was used in conjunction with the Content Pack.

```
<var name="md5" generator="password">
    <param name="password">password</param>
    <param name="algorithm">MD5</param>
</var>
<var name="sha256" generator="password">
    <param name="password">password</param>
    <param name="algorithm">SHA256</param>
</var>
<var name="sha512" generator="password">
    <param name="password">password</param>
    <param name="algorithm">SHA512</param>
</var>
```

Table 6. Password generator algorithm test output

Variable Name	Partial Variable Value
md5_shadow	1LmpEyoFK$1oA
sha256_shadow	5mE6obWIp$kzI
sha512_shadow	6vFT5hu0d$oK2

Because the actual values can be quite long, I've only included the first fifteen characters. Visual inspection shows that each of the encrypted versions used the correct algorithm. The number between the first two dollar signs indicates which encryption algorithm was used. 1 signifies MD5, 5 signifies SHA256, and 6 signifies SHA512. The next set of characters between dollar signs is the salt used to calculate the hash; the hash itself follows the last dollar sign. Both the hash and salt could change if this generator were run again under with the same XML, but the algorithm number should not change. The first part of this test showed that the expected structure was generated, the second part shows that the output

is functional when used on real VMs.

The Content Pack used the outputs to set the password for a user called "md5", "sha256", and "sha512". I was able to successfully able to login as each of these users on an Ubuntu 14.04 server VM using the plain text password "password" as expected.

Since the produced content of this generator matched the expected form and was could be used as a password on a real VM these tests were successful.

4.5.2 Additional Parameter Tests

```
<var name="ascii" generator="password">
    <param name="count">5</param>
    <param name="length">25</param>
    <param name="type">ASCII</param>
</var>
<var name="alpha" generator="password">
    <param name="count">5</param>
    <param name="length">25</param>
    <param name="type">Alpha</param>
</var>
<var name="alphanum" generator="password">
    <param name="count">5</param>
    <param name="length">25</param>
    <param name="type">AlphaNumeric</param>
</var>
```

As Table 7 shows, the expected arrays were generated and were filled with expected content. This experiment demonstrates that the generator is capable of responding to the *type* parameter. The output in Table 7 does not show any errors

Table 7. Password generator parameter test output

Array Name	Index	Value
alpha_plain	0	IVuNQFQnPHgJWHWfqEEioTecw
	1	GyuoHdimuoEXKZzJTYtexeNuh
	2	LsZcbpyKYpnjWwsToiLoMwTQz
	3	tZCDVcWONoJmLwndCtHKoFpYQ
	4	jGGpBCqcKLcCulnpbxJxyaShk
alphanum_plain	0	vyQIjiFzLq6dKAjhS5gw9H8Na
	1	4BlrpwyyLtett6i7LAlSm5DXi
	2	ueKR3U8vRmdeyPnvBM3ZW12lH
	3	yNnujHebqWBrG0W37ySoD00eu
	4	FekNcPKR0gM8nkFhkiglt6M05
ascii_plain	0	A2#*!E^0\c$1qLY'KoWY[J_hK
	1	S'v<pv=-%So&Bw\'\:gc0I}~y
	2	_11MQpk/az1P8F__;QE}4W'{
	3	:4cu~[ukaS>yypDBS6iJ>5Ud|
	4	$&&*UG8!Q9BnSE 46*ZIldkrl

and is indeed random in nature. It could just be that no invalid random characters were selected during these tests. As such this test was successful.

4.5.3 Password Pool File Correctness Tests

Before beginning this test, I created a password pool file that contained ninety passwords each on their own line of the file. The passwords were referenced from Mark Burnett's top ten thousand most commonly used password list [1]. This is the file I used for each test requiring a valid password pool file.

The following XML fragment was used for the first test which should have generated one password randomly chosen from the password pool file.

```
<var name="test1" generator="password">
    <param name="count">1</param>
    <param name="pool">passwordPool.txt</param>
</var>
```

As expected, the generator produced one password, "cheese", from the password pool file. As such this test was successful.

The following XML was used for the second test which should have generated an array of four passwords randomly chosen from the password pool file.

```
<var name="test2" generator="password">
    <param name="count">4</param>
    <param name="pool">passwordPool.txt</param>
</var>
```

Table 8. Password generator pool test two output

Index	Value
0	1111
1	austin
2	secret
3	666666

As expected, the generator produced an array of four passwords from the password pool file. Table 8 shows the content that was produced by the generator. Since this matches the expected output, this test was successful.

4.5.4 Summary

Considering that each of the tests in this experiment were successful, the experiment overall was successful. This generator is capable of producing passwords that function on real VMs according to the parameters used. This experiment demonstrates that the generator behaves as intended for various inputs and can be used by Basic Contributors for their challenges.

4.6 Experiment 6 Results: Can the SSH key generator generate functional content for challenges?

Before beginning this experiment I wrote a Content Pack that would take the keys generated and configure two VMs to make use of them. If the generator performed correctly, running the VMs would show proper functionality.

The following XML produced the public and private parts of an SSH key pair. No password was provided to the generator so the private key is not encrypted.

```
<var name="plainssh" generator="ssh_key">
    <param name="count">1</param>
</var>
```

I was able to login via ssh by explicitly choosing the generated private key to the second VM. This demonstrated that the public and private parts of the key pair were generated correctly and functioned as intended on real VMs. As such, this test was considered a success.

The following XML fragment produced the public and private parts of an SSH key pair. Because a password was provided, the private key was encrypted.

```
<var name="passwordssh" generator="ssh_key">
    <param name="count">1</param>
    <param name="password">password</param>
</var>
```

Again, I was able to login via ssh by explicitly choosing the generated private key to the second VM. Since the private key was encrypted, I had to provide the password to the key before it could be used. Because I successfully logged in, it was demonstrated that the public and private parts of the key pair were generated correctly and functioning as intended on real VMs. As such, this test was also a success.

4.6.1 Summary

Since each test in this experiment was successful, the experiment overall was successful. The experiment demonstrated that the SSH key generator was able to produce SSH key pairs that functioned as intended on VMs. Contributors can

therefore use this generator for their challenges.

4.7 Conclusions

This section discusses how the goals of this thesis were met by summarizing the work performed and the experimentation results.

4.7.1 Goal 1 Conclusions

Goal 1: Create an application and administrative virtual machine (Admin VM) to configure virtual machines for the OCCP

With the incorporation of Puppet and Content Packs, the Admin VM is now capable of configuring virtual machines by performing tasks such as the installation of software, creation of user accounts, configuration of services, transfer files and other content. Experiment 1 showed that it was capable of producing a VSN. Experiment 3 further demonstrated this capability when it reconfigured a VSN with varied content. In both cases virtual machines were produced by the mechanisms introduced by this thesis.

Experiments 4, 5, and 6 showed that the Admin VM is now capable of producing automatically generated content such as usernames, passwords, and SSH keys for use in the configuration of the virtual machines. These experiments also concluded that the content generated was suitable for use in virtual machines and worked as expected.

4.7.2 Goal 2 Conclusions

Goal 2: The extended Admin VM will have reasonable expectations of technical ability for the target audiences.

4.7.2.1 Extending the Existing Syntax

This goal was important because it helped focused the work of this thesis to produce something that was not overly complex, and there by useful to the target

audiences. Part of this goal was to extend the existing syntax already being used by the Admin VM. Although I was not part of the design of the original syntax, the designers had similar goals to make the syntax reasonably easy to work with. Introducing another file, or alternative syntax would add unnecessary complexity to the OCCP. All of my design requirements could be met by simply extending the existing syntax. This was accomplished by adding additional XML tags.

4.7.2.2 Reasonable for Target Audiences

The second part of this goal was to only include elements that were reasonable for the target audiences to operate. For the Basic Administrator audience, there is little to no difference in their use of the OCCP. Since they only run challenges, they do not need to be familiar with any of the additions to the OCCP. The only exception is the regeneration flag and variables. Some members of this audience may be comfortable altering variables or using the regeneration flag but neither is required by this audience to make use of the OCCP. Because of this my extension is very reasonable for this audience to use.

The Contributor audiences are the most affected by the additions of this work. The Basic Contributors should be able to alter variables and parameters to Content Packs. Also, their general familiarity with challenges might allow them to identify Content Packs that could be substituted for others. They do not need to know Puppet because they were not expected to write new Content Packs. It is reasonable for this audience to alter work created by the Advanced Contributors based on their expected level of technical ability.

Finally, since the Advanced Contributors had the highest level of technical ability and familiarity with challenge creation, they often work with unfamiliar technologies in their professional lives or during challenge creation. Despite these high expectations for ability, my design choices still considered the learning curves

that would be required. Puppet's large user base, wide variety of documentation and training material available, and the availability of working examples should ease the learning curve required for this audience to use Puppet in the context of the OCCP. I believe this audience's abilities make it reasonable for them to operate the additions of this thesis to the OCCP.

4.7.2.3 Summary

The Advanced Contributors are able to make use of the mechanisms introduced by the OCCP VM Builder to produce challenges with Content Packs. Their scenario file may include OCCP variables and generators. The Basic Contributors can change the content of the variables or replace Content Packs to produce a different challenge. Challenges created by the Contributors can be used directly by the Basic Administrators who do not need any understanding of the challenge's creation.

4.7.3 Goal 3 Conclusions

Goal 3: Avoid the introduction of a financial cost

This goal was an underlying goal of the OCCP itself. The motivation of this goal was to minimize the barriers to entry to the platform thereby making wider adoption easier. If the work of this thesis had introduced financial cost with its addition, it would have compromised the value of this work and the OCCP in general. The technologies used in this thesis are all available free of charge. Though some technologies used have editions that must be bought, those editions were not needed or used. As such this goal has been met since none of the technologies used require a financial cost to make use of them.

4.7.4 Goal 4 Conclusions

Goal 4: The extended Admin VM should provide a distribution mechanism with reasonable expectations of ability for the target audiences

The export logic expressed in Figure 6 is used by the export mode to produce files required by another Admin VM instance to reproduce the VSN. Experiment 2 tested that the export and import process produced functionally equivalent VSNs. The Advanced and Basic Contributors simply use the export mode to bundle the required files they would need to distribute to share their creation. The Basic Administrators then can use the *deploy* or *launch* mode to import the foreign VSN. Since the Basic Administrator does not have to intervene in this process or manipulate any files, this is reasonable for them to operate. If they were incapable of this, then they were incapable of using the OCCP before the OCCP VM Builder work was introduced. Likewise, the Advanced and Basic Contributors do not have to do any additional work other than running export mode. The OCCP VM Builder's Admin VM does all the decision making for them. Their technical abilities would allow them to change the mode argument to the program to export. The alternative would be for them to follow the same logic that the Admin VM now uses and then to manually revert and export the correct VMs. As such, the export mode logic makes it easier for Contributors to prepare their work for distribution.

List of References

[1] M. Burnett. "10,000 top passwords." July 2014. [Online]. Available: https://xato.net/passwords/more-top-worst-passwords/#.VE6Xp0tlxj4

CHAPTER 5

Conclusion

5.1 Conclusions

This thesis was successful in extending the Open Cyber Challenge Platform's Administrative Virtual Machine and program. The Admin VM is now capable of producing and reconfiguring VMs for use with the OCCP. The technologies used in the extension did not introduce any financial cost and are reasonable for each of the target audiences to use within the scope of their abilities. Though this work met the goals established in Section 1.3, there are elements that could still be improved.

Even with the work of this thesis, creating challenges can still be a time consuming and difficult task. Instead of creating the VMs manually, someone still must produce the content packs required to configure the machines. In some instances content packs can be reused. In other circumstances the content packs must be created from scratch. Though it can be time-consuming to create content packs, the content packs provide the benefit of reuse. Trying to reconfigure a VM that was configured manually can be tedious and error prone with limited return on investment. With careful creation of content packs and the use of variables and generators, the effort invested can be reused. Furthermore, the work can be shared with other users of the OCCP. The ability to share and reuse work, despite the effort required to make the original version, is valuable to the cyber security education community.

Although operating system diversity was a consideration when choosing the configuration management system, free Linux distributions were the focus of my tests. This was largely due to licensing restrictions imposed by Microsoft and Apple for their operating systems. Puppet was chosen because it was compatible with all

the major operating systems so there is nothing immediately obvious preventing licensed operating systems like Windows or Mac OS X from being used with the extended Admin VM with future work.

5.2 Future Work

This section discusses possible future work that could improve the work of this thesis.

5.2.1 Licensed Operating Systems

As previously stated, licensed operating systems such as Windows and Mac OS X were considered, but largely untested. The major hurdle to using Mac OS X with this work is the license agreement disallows installing it on non-Apple hardware, which limits the hypervisors that can be used. Though Microsoft Windows does not have such installation restrictions, its activation requirements are more strict. Another hurdle that would need to be worked out is remotely controlling Puppet. Remote command execution is not as easy with Windows as it is with Unix-based operating systems. Finally, these systems cannot be shared due to their licensing. Base VMs or scenarios using these operating systems could not be distributed as is. Instead instructions for creating these bases would need to be provided. Future work could determine the best practices for overcoming these challenges.

5.2.2 Automated Installations

Base VMs were created to have a common starting point and simplify some aspects of the this work. By cloning a machine with the operating system already installed, the user saves the time it takes to install the operating system each time they make use of that Base VM. The draw back is that the user is limited to the available Base VMs or must make their own. It may be possible with future work for the Admin VM to automatically create Base VM's from installation media or

ISOs. Some operating system vendors provide systems that can provide answers to the questions the installer would ask during installation. With this mechanism it could be possible to automatically produce Base VMs. Since the systems varied from distribution, attempting to include this automated installation was beyond the scope of this thesis project and would simply be a convenience from having to create a Base VM by hand.

5.2.3 Additional Generators

More generators would provide challenge Contributors with more tools to work with and more variation during regeneration. A network generator would be useful for challenges to initially appear different despite having the same network topology. It should be possible to have users be able to request networks capable of supporting at least the number of machines required. Users could also specify if the network should be publicly routable, or one of the reserved for internal-use networks. The generator would generate the appropriate elements such as the subnet mask, gateway address, and broadcast address in addition to an array of all the valid IP addresses for the network. Creating this generator is complicated due to various reserved networks and sanity checking requirements but is still feasible to produce with additional time.

5.2.4 Further Evaluation

In order to draw more solid conclusions on the effectiveness and utility of the OCCP VM Builder, more testing and evaluation should be performed. Experiments 1 & 3 compare manually creating and reusing a challenge against the OCCP VM Builder, however I was the only one to perform the experiments. My own technical abilities may have influenced the results. With additional time, more people should conduct the experiments with a common challenge design. A larger

and more diverse group of testers would introduce different levels of technical abilities performing the same set of actions. This data could be used to find averages for the metrics listed in the experiments.

In addition to a larger group of testers, having the testers repeat the experiments with multiple challenge designs could be useful. Although different designs should produce varied results, since the work required to implement them is different, it could help validate the results. It would show that the OCCP VM Builder is applicable to different challenges, and give a wider sample size. The challenge designs must be descriptive in order for testers to be able to produce the challenge. The design should describe each VM involved with the following:

- What software should be installed.
- How the software is configured differently from the default settings.
- What user accounts should be present.
- What the account passwords should be set to, or describe set of acceptable passwords if it can be randomized.
- What network(s) the VM is connected to.
- What content needs to be transferred.
- What the Unix ownership and permissions should be for content.

5.3 Accomplishments

Previous to the work of this thesis project, there did not exist a tool by which reusable virtual machines for cyber challenges. By extending the already free OCCP in a way that did not introduce a financial cost, there now exists such a tool. The effort exerted by challenge creators can now be reused by themselves and

others when they share their challenges with the OCCP user base. Variables and generators can make functionally equivalent, but slightly different VSNs which has great benefits to the education community. With greater adoption of the OCCP, the work of this thesis should ease the burdens of creating new challenges.

BIBLIOGRAPHY

Docker Inc. Apr. 2014. [Online]. Available: https://www.docker.io/learn_more/

Ansible Inc. "Ansible documentation." Feb. 2014. [Online]. Available: http://docs.ansible.com

"Clonezilla - about." June 2014. [Online]. Available: http://clonezilla.org

Puppet Labs. "Configuring a puppet master server with passenger and apache." June 2014. [Online]. Available: https://docs.puppetlabs.com/guides/passenger.html

Docker Inc. "Docker documentation." Aug. 2014. [Online]. Available: https://docs.docker.com

Oracle. "Documentation - Oracle VM VirtualBox." Jan. 2014. [Online]. Available: https://www.virtualbox.org/wiki/Documentation

The Apache Software Foundation. "Documentation: Apache http server - the apache http server project." June 2014. [Online]. Available: http://httpd.apache.org/docs/

"FOGUserGuide - FOGProject Wiki." June 2014. [Online]. Available: http://www.fogproject.org/wiki/index.php?title=FOGUserGuide

Bruning Glass. "Job market intelligence: Report on the growth of cybersecurity jobs." Aug. 2014. [Online]. Available: http://www.burning-glass.com/media/4187/Burning%20Glass%20Report%20on%20Cybersecurity%20Jobs.pdf

Digital Forensics and Cyber Security Center. "OCCP." Jan. 2014. [Online]. Available: https://opencyberchallenge.net

Puppet Labs. "Puppet labs documentation." Feb. 2014. [Online]. Available: https://docs.puppetlabs.com

"Solution — NICE Challenge Project." Nov. 2014. [Online]. Available: https://www.nice-challenge.com/

The Squid Software Foundation. "squid : Optimising web delivery." June 2014. [Online]. Available: http://www.squid-cache.org/Doc/

HashiCorp. "Vagrant documentation." Aug. 2014. [Online]. Available: http://docs.vagrantup.com/v2/

HashiCorp. "Vmware vagrant environments - vagrant." Feb. 2014. [Online]. Available: http://www.vagrantup.com/vmware

Rapid7. "Vulnerability & exploit database — rapid 7." June 2014. [Online]. Available: https://www.rapid7.com/db/modules/

Ansible Inc. "Windows support - ansible documentation." Aug. 2014. [Online]. Available: http://docs.ansible.com/intro_windows.html

Burnett, M. "10,000 top passwords." July 2014. [Online]. Available: https://xato.net/passwords/more-top-worst-passwords/#.VE6Xp0tlxj4

Cheung, R. S., Cohen, J. P., Lo, H. Z., and Elia, F., "Challenge based learning in cybersecurity education," in *Proceedings of the 2011 International Conference on Security & Management*, vol. 1, 2011.

Childers, N., Boe, B., Cavallaro, L., Cavedon, L., Cova, M., Egele, M., and Vigna, G., "Organizing large scale hacking competitions," in *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2010, pp. 132–152.

Doupé, A., Egele, M., Caillat, B., Stringhini, G., Yakin, G., Zand, A., Cavedon, L., and Vigna, G., "Hit 'em where it hurts: A live security exercise on cyber situational awareness," in *Proceedings of the 27th Annual Computer Security Applications Conference*, ser. ACSAC '11. New York, NY, USA: ACM, 2011, pp. 51–61. [Online]. Available: http://doi.acm.org/10.1145/2076732.2076740

Fanelli, R. L. and Oconnor, T. J., "Experiences with practice-focused undergraduate security education," in *Proc. of the 3rd Workshop on Cyber Security Experimentation and Test, Washington, DC*, 2010.

Stewart, K. E., Andel, T. R., and Humphries, J. W., "Measuring the performance of network virtualization tool n2n in the design of a cyber warfare training and education platform," in *Proceedings of the 2011 Military Modeling & Simulation Symposium*, ser. MMS '11. San Diego, CA, USA: Society for Computer Simulation International, 2011, pp. 28–35. [Online]. Available: http://dl.acm.org/citation.cfm?id=2048558.2048563

Stewart, K. E., Humphries, J. W., and Andel, T. R., "Developing a virtualization platform for courses in networking, systems administration and cyber security education," in *Proceedings of the 2009 Spring Simulation Multiconference*, ser. SpringSim '09. San Diego, CA, USA: Society for Computer Simulation International, 2009, pp. 65:1–65:7. [Online]. Available: http://dl.acm.org/citation.cfm?id=1639809.1639877

Vigna, G., "Teaching network security through live exercises," in *Security education and critical infrastructures*. Springer, 2003, pp. 3–18.

Wagner, P. J. and Wudi, J. M., "Designing and implementing a cyberwar laboratory exercise for a computer security course," in *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE

'04. New York, NY, USA: ACM, 2004, pp. 402–406. [Online]. Available: http://doi.acm.org/10.1145/971300.971438

Wagner, R. H., "Designing a network defense scenario using the open cyber challenge platform," *Open Access Master's Theses*, no. 73, 2013. [Online]. Available: http://digitalcommons.uri.edu/theses/73

Wang, K., Rao, J., and Xu, C.-Z., "Rethink the virtual machine template," in *Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, ser. VEE '11. New York, NY, USA: ACM, 2011, pp. 39–50. [Online]. Available: http://doi.acm.org/10.1145/1952682.1952690

Werther, J., Zhivich, M., Leek, T., and Zeldovich, N., "Experiences in cyber security education: The mit lincoln laboratory capture-the-flag exercise," *Cyber Security Experimentation And Test*, vol. 8, 2011.