Resilient Ansible Cyber Education Range

Taegan Williams
University of Idaho
Moscow, Idaho
will8798@vandals.uidaho.edu

***Abstract*—Educational resource for cybersecurity research and software analysis is critical to understanding the nature of communication across a critical infrastructure network. A resilient cyber range for students to practice creating a dynamic defense-in-depth network leverages multiple systems and software to create a realistic learning environment. Providing total visibility through services for inventory management, state monitoring, event monitoring, and incident management need to exist outside of the scope of the environment to ensure the productivity of the lab and its exercises. With a primary goal to use open-source software throughout the environment, this research highlights the use of resilience of the lab through the creation of Ansible scripts. These Ansible scripts remove the human factor from the creation of services such as Nagios, Trac, Security Onion, Suricata, and RT-IR to provide total visibility and fast reproduction of the environment to safely analyze and practice blue team and red team concepts. The final scripts created can be found on Github at https://github.com/taeganw/RACER where they will continue to grow into living code.**

I. Introduction

With an increase in cyber-related crime around the world, there is a need to create new educational resources for students, researchers, and teachers to teach the next generation about defensible network architecture and solutions. With a high risk of activities affecting educational resources outside of the scope of the environment, it is the mission of this research to utilize open-source solutions in a manner that can be easily distributed, redeployed, and productive to its users. The issue with the modern implementation of these lab environments is ensuring the safety and isolation of the activities performed by students while providing total visibility to the administrators, teachers, and researchers. By building robust logging, alerting, and change management solution that analyzes the activity within the secure-by-design infrastructure, students and researchers can perform cybersecurity practices while maintaining multiple detection systems and administrative controls. This paper presents an approach to utilizing open-source software in a resilient manner so that the environment is secure by design.

II. Ansible for Resilience

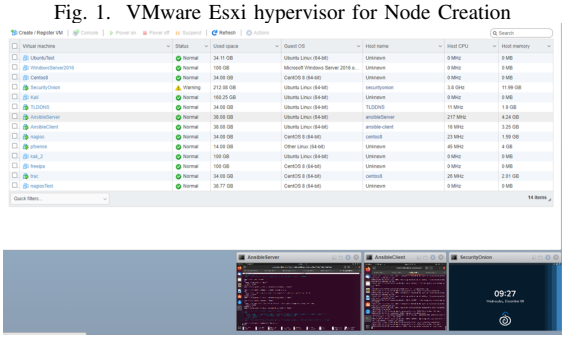
Ansible is a DevOps tool for system provisioning and configuration intended to enable infrastructure as code [1]. Ansible helps reduce the impact of administrator error on security operations by making operations easily repeatable and auditable. This provides resilience by removing the human element from the initial installation and setup [1]. With pre-created Ansible scripts, an administrator would be able to execute a single command to have multiple commands run to install software, configure firewall rules, and change configuration files all remotely for full connectivity. Ansible scripts are better referred to as Ansible-playbooks.

III. Creating the Environment

A. Installing Virtual Machines

I initially began setting up the environment by installing VMware sphere ESXi 6.7.0 on a Dell R710 rack server. This hypervisor software allows me to install virtual machines, create virtual switches, and network interfaces. I installed virtual image files for Centos8, PfSense, Ubuntu 20.04, Kali Linux, and Security Onion (a prebuilt operating system on top of Centos 7). I then created base images for each operating system with up-to-date packages. With base images for each operating system, I can clone the images and install services without going through the initial setup process for each new machine added to the environment. Figure 1 shows the user interface for VMware Esxi for testing and development.

Fig. 1. VMware Esxi hypervisor for Node Creation

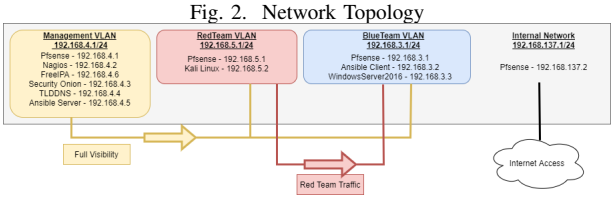


B. Network Topology

Within the environment, I created four separate virtual networks. These networks were created on the hypervisor by instantiating three separate Virtual Local Access Networks (VLAN) on a virtual switch. Outside of the 3 VLANs is another virtual network that allows for communication out to the internet. The firewall virtual machine, PfSense, is connected to each VLAN as the default gateway and allows inter-network communication. The purpose of the PfSense firewall is to limit network traffic from one local network to another, preventing communication directed from the red team network and blue team network to the management network. This would also allow communication to be forwarded to the

management network with full visibility. Figure 2 shows the network segmentation during development for the blue team, red team and management networks.

Fig. 2. Network Topology



IV. Required Services for Ansible Execution

In my testing environment, an Ubuntu 20 image is used as an Ansible server. This server has a configuration file with all of the machine's IP addresses. In practical use, any operating system can be used to execute the Ansible commands as long as the following conditions are met. The Ansible server communicates over SSH using preshared keys to configure hosts on all of the VLANs. Before Ansible can be run successfully, each machine that will be receiving commands from the Ansible server needs to have a preshared SSH public key to allow a user to execute scripts on the remote devices. This also means that each connected device will need to have network connectivity to the Ansible server. To set up SSH, the service SSHD must be installed using the command line application installer. When the ssh server is installed, another executable called ssh-keygen is a part of the package to create public-private key pair for simple login without a password. Ssh-keygen uses the public-private key algorithm RSA for authentication, and the public key can be easily transferred to another device with ssh-copy-id. The designated user account that is signing into the remote device should also be able to elevate to a higher privilege to execute commands. This is best done by giving the user Sudo privileges. To simplify this process, I created bash scripts to be run on Ubuntu20 and Centos 8 to preconfigure a static IP address on the device and ensure that ssh is installed. Figure 3. is an example of the Centos 8 script created.

Fig. 3. Variables File: Bash script to configure static IP

```
[user@centos8 ~]$ cat set_static_ip.sh
#!/bin/bash
nmcli con mod $1 ipv4.method manual
nmcli con mod $1 ipv4.address $2
nmcli con mod $1 ipv4.gateway $3
nmcli con mod $1 ipv4.dns "192.168.4.4 8.8.8.8"
nmcli con mod $1 autoconnect yes
nmcli con down $1
nmcli con up $1
sudo dnf install openssh-server
sudo systemctl start sshd
sudo systemctl enable sshd
sudo firewall-cmd --zone=public --permanent --add-service=ssh
sudo firewall-cmd --reload

[user@centos8 ~]$ sudo bash set_static_ip.sh ens192 10.0.0.2/24 10.0.0.1
```

A. Configuring Pfsense

PfSense is an open-source firewall software that provides similar functionality to modern commercialized firewall applications [2]. Within the environment, the firewall limits the visibility from one VLAN to another. The PfSense router limits visibility by having pre-configured firewall rules.

The main rules allow traffic that was initiated from the management VLAN to the red or blue team networks, allow traffic that is initiated from the red team network to the blue team network, and allow all traffic to and from the blue team network. Outside the scope of this project would further limit the visibility of the blue and red team networks to the management network to prevent the management network from being inside the scope of an attack. Another great implementation would be to implement a 0-trust network design by denying all traffic and only allowing necessary communication. After connecting the three VLANs to the PfSense router, I then configured the networking layer for each VLAN to have a subnet big enough for 254 hosts on the management network, red team network, and blue team network.

1) Configuring Pfsense with Ansible: Ansible can configure the PfSense router with the appropriate settings and firewall rules. In theory, this would be great for resilience if the PfSense router had become compromised; however, the PfSense router is the center for all communication between VLANs. This means that the Ansible-playbooks for the PfSense router would need to be run before the initial configuration of the environment. In practical use, an administrating computer would be created to run Ansible-playbooks on a separate network with the PfSense router. Ansible-playbooks would first be run to configure the management, red team, and blue team networks, then the associated devices would need to be connected manually to configure the network settings to route traffic to the PfSense router as the default gateway before executing the Ansible-playbooks to create the environment.

B. Top Level DNS

The Domain Name System (DNS) server is configured so all the systems can have a domain name to IP address translations. This is required for services like freeIPA to work properly and is helpful for other services such as Security Onion and Ansible. To configure the top-level DNS, I used a Centos 8 virtual machine that uses bind9 as a service. I created a forward lookup zone and a reverse lookup zone and added the hostname and IP addresses for Nagios, Security Onion, and the Ansible server. The DNS server is not needed for Ansible-playbook execution; however, it is critical to run services within the environment.

1) Configuring Top Level DNS with Ansible: The Top Level DNS can be configured with Ansible. If the DNS server were to become compromised it would be beneficial to run a command to reconfigure the DNS server. In practical use, on initial execution, it would need to be one of the first playbooks to be run before the installation of services that required DNS. This would also require a drop in connection as the network interfaces that will have a newly assigned DNS address would need to restart the networking service for the new configurations to take effect. This would drop the connection to the Ansible server momentarily and would

need to be considered before executing additional Ansible-playbooks on the remote devices.

C. Security Onion

Security Onion is a free and open-source Linux distribution that provides intrusion detection, enterprise security monitoring, and a log management system [3]. To best understand the Security Onion software, I initially downloaded the Security Onion ISO image from the Security Onion Solutions GitHub [4]. I then created a virtual machine and went through the setup process. Security Onion is a toolkit that sits on top of a Centos 7 operating system. The Security Onion setup process could be a preconfigured Ansible-playbook to run through the setup process for a connected Centos 7 device and should be a playbook that is run after the initial DNS configuration and before any additional service playbooks are run.

V. Ansible Files and Folders

A. Ansible Execution Files

Ansible has 3 main files and folders for operation. Variables that can be changed by the user before running the script are found in the all.yml file in the group_vars folder. Some example variables include the users to create, services to deploy, or even the port for the service to operate on. Ansible can also encrypt sensitive passwords that would normally be used within this variables file using Ansible vault. Figure 3. shows the syntax used within the variables file.

Fig. 4. Variables File: all.yml

```
GNU nano 4.8

## Ansible user
# change this if running Ansible as a non-root user
# would require sudo, e.g. AWS EC2 uses ec-user
ansible_system_user: root

### Firewall management ###
# Set to true to deploy firewall rules for nagios server
manage_firewall: true
# Set to true to deploy firewall rules for client nrpe
manage_firewall_client: true

### Nagios Variables ###
nagios_release_version: nagios-4.4.6
nagios_plugins_release_version: nagios-plugins-2.3.3

nagios_plugin_path: /usr/lib64/nagios/plugins
nagios_username: nagiosadmin
nagios_create_guest_user: true
nagios_ro_username: guest
nagios_ro_password: guest
nagios_http_port: 80
nagios_https_port: 443
admin_name_01: nagiosadmin
# change this to your email address if you want notifications
admin_email_01: nagios@localhost
# set to false if you don't want to monitor swap
# on your nagios server
nagios_host_monitor_swap: true

### NRPE variables ###
nrpe_tcp_port: 5666
# threshold for # of processes
warning_proc: 900
```

The initial playbook file holds the main Ansible commands to query information about the client to be configured, tasks to be performed, or roles to be assigned to clients. This file often ends with the extension .yml which is a markup language that has strict formatting. It also limits the amount of tasks that can be performed at once to help with readability. The roles defined in the Ansible-playbook file can lead to additional playbooks defined for each role within the roles directory. Figure 4 shows the initial playbooks that assigns playbooks to those specified in the host file.

Fig. 5. The Main Playbook File

```
GNU nano 4.8

# role for nagios server
- hosts: nagios
  remote_user: "{{ ansible_system_user }}"
  roles:
    - { role: nagios }
    - { role: firewall, when: manage_firewall|bool }
    - { role: instructions }

# role for trac server
- hosts: trac
  remote_user: "{{ ansible_system_user }}"
  roles:
    - { role: trac }

# role for wazuh SecurityOnion
- hosts: LinuxClients1
  remote_user: "{{ ansible_system_user }}"
  roles:
    - { role: nix_monitor }

#- hosts: WindowsClients1
  remote_user: "{{ ansible_system_user }}"
  roles:
    - { role: win_monitor }


- hosts: WazuhAgents
  become: yes
  remote_user: "{{ ansible_system_user }}"
  roles:
    - { role: wazuh/ansible-wazuh-agent }
  vars:
    wazuh_managers:
```

The last important file to consider is the host's file which groups hosts on the network and defines the user to execute commands via SSH. Figure 5 shows the main host file configuration used during execution.

B. Ansible Playbooks

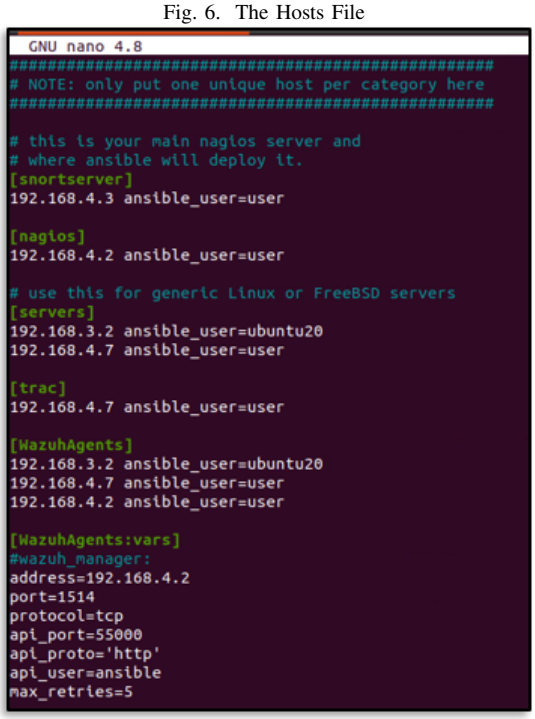
Ansible-playbooks execute commands on a remote device. Simple Ansible commands can install services using the command line application install, and more advanced commands can copy configuration files with considerations from variables imported from the variables file that would normally need to be configured on the remote host from the playbook directory. Configuration files that are copied via Ansible have the extension .j2 and also allow for conditional formatting based on the operating system. This allows for the remote system to be configured with considerations to its operating system.

VI. Ansible Playbooks for Services

A. Trac

Trac is an open-source, web-based project management and bug tracking system [5]. Trac includes a wiki for documentation, milestone tracking, and code versioning [5]. While its primary use is for code development projects, Trac is a great tool to organize lesson plans and documentation for machines

Fig. 6. The Hosts File



and services. We also use Trac to maintain key configuration files. The Ansible scripts to set up Trac within the environment are primarily used to create the service and not to populate the wiki.

1) Trac Playbook: From the initial playbook file, the Trac role is assigned to the specified host in the host's configuration file. The Ansible script is meant to configure Trac on a Centos 8 operating system. The Trac playbook takes inspiration from a GitHub repository that configures Trac with Ansible [6]. The Playbook first installs Trac version 1.4.3 using wget and then installs its dependencies using the yum application manager. Trac is a python based program and once initially installed, the Trac-admin command is used to initialize, create and deploy a project. The Ansible script then copies the configuration file for the project, as well as various files that help Trac host projects to the remote machine. The remote machine then hosts the wiki web server using Apache. Apache is a service used for hosting web applications. The Ansible-playbook then installs various plugins from Trac-hacks.org that are recommended for ticket tracking and workflow visualization. The next task sets up the plugins by setting values within the Trac.ini file. The program then checks to see if there are upgrades that can be performed and changes the permissions of the hosted project file.

2) Amendments to the Ansible script: After using the Trac Ansible script on the newly created Centos 8 VM, I was not able to get the script to install and set up Trac with its dependencies. The issue was due to Trac is not able to be built and run with python3. Python3 is the default option on Centos 8 after the initial install. Besides python not functioning properly, many of the dependencies that are installed with yum were not able to be found. I opted to install Trac in the way it was recommended on from the creators website. I first downloaded the file to the tmp directory, extracted the file, then setup the installation with python2. I then installed mod_wsgi and svn to help bootstrap Trac to Apache. These dependencies were highlighted in the referenced GitHub repository but were no longer supported. The rest of the installation steps are outlined above as once the service was installed, it was able to configure the hosting repository correctly. The final Trac installation script can be viewed at the following web address: https://github.com/taeganw/RACER/tree/main/install/roles/trac

B. Nagios

Nagios is an open-source network monitoring application [7]. While originally designed to be run on Linux, its client-server model can host Windows machines as clients. Nagios can monitor networking services including SMTP, POP3, HTTP, PING, etc. It can also monitor host resources to identify spikes in processor load or disk usage. The configuration files for Nagios also allow for user-created service checks.

1) Nagios Playbook: Within the Environment, I recreated the Ansible server and client scripts from a community-supported file on GitHub [8]. The setup for this Ansible-playbook was the start of all of the services I built within this project. 4 playbooks are utilized to setup Nagios. 2 Nagios playbooks configure the firewall for the server and the client, and the other 2 configure the Nagios server and the Nagios clients. For the Nagios server, The Ansible-playbook installs the EPEL repository so further dependencies can be installed with the application manager. Nagios, nagios plugins and dependencies to host the webserver are then installed with yum. A new user and group are created in which the hosting file will be changed to have managed permissions. Configuration files are then created specifically for select services and are copied to the Nagios server. This way the server knows how to run commands on the remote hosts. SELinux is a security service installed on Centos machines and often causes issues when configuring network services. Before restarting and enabling Nagios and the HTTP web server, the Ansible-playbook allows Nagios to run on the initial setup with the ability to escalate privileges to run commands as root. For Nagios clients, the steps are similar in adding the EPEL repository, installing the services, and copying over the configuration file. The final Nagios installation script for the Nagios server can be viewed at the following web address: https://github.com/taeganw/RACER/tree/main/install/roles/nagios.

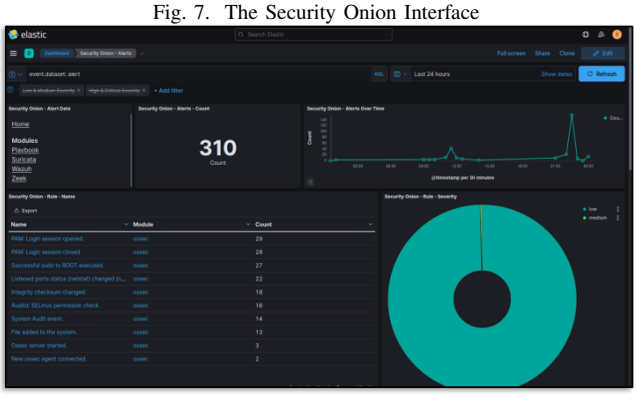
2) Amendments to the Ansible script: From the original script, I removed unnecessary checks for IDRAC devices and changed many of the directory paths so the server can find the appropriate commands to run. The server also runs as a client so that it can perform service checks on its system as well as the clients that send data using NRPE over port 5666. After running the community script on GitHub I had initial problems downloading services and dependencies, so I recreated the script to install Nagios as a zipped package from

the website and run the installation by building from source code. I also added conditional checks to see what operating system the client machine was running. This expanded the current scope for allowed clients to include Ubuntu 20.04. I would like to further expand the script to allow for Windows clients as well. Within the environment all of the machines on the management, blue team, and red team network all report service status to the Nagios server. Current services checks include the availability of the machine through SSH, ping, the number of users on the machine, and the current disk and processing load.

C. Wazuh

Wazuh is an open-source security monitoring solution for host intrusion detection [9]. Wazuh utilizes a client-server model and validates host settings to ensure the integrity of files and compliance to industry standards [9]. Wazuh is configured in the environment to be a part of the Security Onion monitoring solution. While Wazuh is installed with Security Onion by default, Wazuh clients must be installed within the environment in order to report host intrusion detection to the monitoring server. At the time of writing, the Wazuh Ansible-playbook can set up Wazuh clients on Ubuntu or Centos operating systems. Figure 7 shows the threat hunting interface configured on Security Onion with live information I am receiving from the Wazuh clients.

Fig. 7. The Security Onion Interface



1) Wazuh Playbook: Wazuh has great documentation and a GitHub repository maintained by Wazuh [10]. I did not change this playbook; however, after its first use, there was complications in version maintenance as the Wazuh server that is preinstalled on Security Onion is outdated. This caused the clients that were set up with the GitHub repository to be incompatible with the Wazuh server. To correct this issue, I changed the initial install steps to install the correct version of Wazuh to match the server. Further research is needed to discover a human-free method of installation. Wazuh operates on port 1514 to send information to the server and port 1546 to register the client to the server. After the client is registered, the key generated on the server must be manually copied to the client to complete the registration process. The playbook finishes by allowing ports 1514 and 1546 through the firewall to communicate with the Wazuh server. The configuration file is copied from the Ansible server and the Wazuh service is restarted and enabled.The final wazuh client installation script can be viewed at the following web address: https://github.com/taeganw/RACER/tree/main/install/roles/nix_monitor.

D. Suricata

Suricata is the leading independent, open-source, network intrusion detection system [11]. In the initial research phase, I considered using snort for intrusion detection; however, Suricata was built more recently and allows for multi-threading [11]. Multi-threading allows for faster response time when a network intrusion is detected. Suricata works by recording traffic, analyzing traffic, and either alerting or preventing communication if it matches an identifying rule. The following resource has a large repository of identified malicious behavior [12]. All identified traffic is stored in an eve.json file. To analyze the traffic on our monitoring system, we can utilize Wazuh to send alerts from the JSON file to the Wazuh server. This way no further ports are opened on the management, red team, or blue team networks.

1) Suricata Playbook: While the Suricata Ansible-playbook has not been created, the steps are very similar to the previously highlighted services. The Suricata dependencies for network packet capture would be installed and the Suricata service would be run with the emerging-all rule set. Suricata would then be installed from the development website via source code, and the referenced rule-set would be added with the service at the time of execution. The Wazuh configuration file would then be updated to receive alert logs from Suricata and send them to the Wazuh server. The Wazuh service would then be restarted to account for the updated configuration file.

E. RT-IR

RT-IR is a ticket Tracking system for incident response [13]. While Trac is used for versioning and documentation, the separation of responsibility promotes the idea of using incident response separate from the documentation web server [13]. RT-IR is simple and effective to allow team members to respond to reported incidents with an industrial-grade incident handling tool [13].

1) RT-IR Playbook: RT-IR is built from source code using Perl. Within its installation process, there are multiple dependencies that the application will require before RT-IR is installed successfully. While I was not able to build out the Ansible script at the time of this project, I did review a GitHub repository that sets up RT-IR and believe the commands to set up RTIR on the Security Onion box to be straightforward. The Ansible file would first install RT-IR from source code and install plugins that are recommended by its supporting community. The Ansible-playbook would then restart and enable the service to run on the Security Onion server on port 8000. RT-IR should be built on the Security Onion server to limit the need for networking capability to analyze artifacts from the Security Onion dashboard to the RT-IR incident response tool.

VII. Future Work on the Project

A. Vagrant to Configure Virtual Machines

The purpose of this project is to give students, researchers, and educators the ability to run red team/blue team exercises in a safe and secure environment. The environment is to provide full visibility via network communication with an incident response device configured outside the scope of an attack. Most students do not have access to the hardware resources to configure a full-scale production environment. To make these scripts more usable by students, further work on this project should include the service Vagrant. Vagrant is a tool to manage virtual machines with the type 2 hypervisor software Virtual Box. The Ansible script would then be able to configure the hardware and networking resources for the environment and remove the human element during resource allocation. Moving forward, two separate projects should be considered for educational and personal use. Some services such as Security Onion is a resource expensive service that students would not find as necessary as researchers. For students, Ansible playbooks to configure 3 systems on 3 segmented networks should be hard requirement. Many of the services highlighted in this research are still light-weight enough for students to utilize. The management network can be a Centos 8 system with host and network intrusion detection capability with Suricata and Wazuh. The blue team network for the ansible scripts should be reconfiguarble for different hardening and attack exercises.

B. Conditional Checks for Unnecessary Tasks

When an Ansible-playbook is run that is not configured with conditional statements to check if a task has already been completed, it adds clutter and can cause unnecessary action on the device. With conditional statements, the ansible-script would be far more efficient.

C. Security before Productivity

Having Wazuh within the environment there are multiple indicators of insecure settings on the Linux devices. These security gaps should be addressed before the Ansible scripts are used within production settings. One could create Ansible scripts that harden the remote device before the rest of the playbooks download services. This would further prevent the management network from being within the scope of attack, but still allow communication with the red team and blue team networks.

VIII. Conclusion

A. Final Deliverable

The Ansible scripts from this research are available on GitHub for public download. GitHub offers version Tracking and the ability for communities to build off of live projects. By sharing this research on GitHub, I hope to continue to build these scripts to work with multiple clients and become more resilient on vastly different networks. The github repository can be found at https://github.com/taeganw/Racer.

B. Lessons Learned

By using Ansible-playbooks there is a clear benefit in removing the human element to create environments for students, educators, and researchers. Ansible scripts are readable and simple for a beginner to configure and execute. The strict syntax in Ansible YAML files helps the user review the scripts before execution. Ansible scripts are still very fragile in the sense that if a task fails without permission to continue, the execution of the playbook will stop preventing the setup of services. There is also a possibility that the script executes services at root privileges that should be run at a user level. This could lead to further vulnerabilities that should be noted before a complete release of the project. While I was unable to complete the full list of services for this project, I overcame many challenges during the initial setup of the project and became more aware of the current state of ansible playbooks. While ansible provides a great foundation to build powerful services, there are few maintained examples that are professionally built and functional. With the creation of these scripts there is hope to give security researchers, students, and educators more resources to prepare for tomorrow's cyberattacks.

References

[1] *How ansible works*. [Online]. Available: https://www.ansible.com/overview/how-ansible-works.

[2] *Open source security*. [Online]. Available: https://www.pfsense.org/.

[3] *About security onion*. [Online]. Available: https://docs.securityonion.net/en/2.3/about.html.

[4] *Security onion github*. [Online]. Available: https://github.com/Security-Onion-Solutions/security-onion.

[5] *Welcome to the trac open source project*. [Online]. Available: Welcome%20to%20the%20Trac%20Open%20Source%20Project.

[6] *Trac github*. [Online]. Available: https://github.com/jermon/trac.

[7] *About nagios core*. [Online]. Available: https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/about.html#whatis.

[8] *Nagios github*. [Online]. Available: https://github.com/sadsfae/ansible-nagios.

[9] *Wazuh documentation*. [Online]. Available: https://wazuh.com/.

[10] *Wazuh github*. [Online]. Available: https://github.com/wazuh/wazuh-ansible.

[11] *Suricata documentation*. [Online]. Available: https://suricata.readthedocs.io/en/latest/.

[12] *Suricata rules*. [Online]. Available: https://rules.emergingthreats.net/open/suricata/.

[13] *Rt incident response*. [Online]. Available: https://github.com/bestpractical/rtir.