12-2018

Testing the Security Onion

Kayla Jansen
kkjansen@stcloudstate.edu

Follow this and additional works at: https://repository.stcloudstate.edu/msia_etds

Recommended Citation


This Thesis is brought to you for free and open access by the Department of Information Systems at theRepository at St. Cloud State. It has been accepted for inclusion in Culminating Projects in Information Assurance by an authorized administrator of theRepository at St. Cloud State. For more information, please contact rswezelbaum@stcloudstate.edu.
Testing the Security Onion

by

Kayla Jansen

A Thesis

Submitted to the Graduate Faculty of

St. Cloud State University

In Partial Fulfillment of the Requirements

for the Degree of

Master of Science in

Information Assurance

December, 2018

Committee Members:
Dennis Guster, Chairperson
Susantha Herath
Erich Rice
Abstract

Security professionals utilize different types of systems, tools, and software in an attempt to secure an organization from external threats. There are many challenges that professionals face, when attempting to choose and execute a system into their framework. Because of these challenges, professionals may decide to go with a free open source system, such as the Security Onion. However, there is little information or results that show the effectiveness of the system. Several articles indicate ways of configuring the system or examining certain components within it. This project aims to examine the effectiveness of the Security Onion through a controlled test. The results of this project help inform professionals of the effectiveness and provide a baseline for them to make their security decisions off of.
# Table of Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>List of Tables</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>List of Figures</td>
<td>7</td>
</tr>
<tr>
<td>I</td>
<td>Introduction</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Introduction</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Problem Statement</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Nature and Significance of the Problem</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Objectives of the Research</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Research Questions</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Limitations of the Research</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Definition of Terms</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Summary</td>
<td>15</td>
</tr>
<tr>
<td>II</td>
<td>Background and Review of Literature</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Introduction</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Background Related to the Problem</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Literature Related to the Problem</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>Fundamentals of a Network</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>Private Network</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>Types of Attacks</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>Summary</td>
<td>26</td>
</tr>
<tr>
<td>Chapter</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>III. Methodology</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>Introduction</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>Design of the Research</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>Setting up the devices</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>Establishing a private network</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>Ping test</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>Security Onion</td>
<td>39</td>
<td></td>
</tr>
<tr>
<td>DoS attacks</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td>Conclusion</td>
<td>51</td>
<td></td>
</tr>
<tr>
<td>Data Collection</td>
<td>51</td>
<td></td>
</tr>
<tr>
<td>Data Analysis</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>Summary</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>IV. Data Presentation and Analysis</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>Introduction</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>Data Presentation</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>Baseline Data Presentation</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>Security Onion Data Presentation</td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>Data Analysis</td>
<td>78</td>
<td></td>
</tr>
<tr>
<td>Summary</td>
<td>81</td>
<td></td>
</tr>
<tr>
<td>V. Discussion, Future Work, and Conclusion</td>
<td>82</td>
<td></td>
</tr>
<tr>
<td>Introduction</td>
<td>82</td>
<td></td>
</tr>
<tr>
<td>Discussion</td>
<td>82</td>
<td></td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1. IP Address Classes</td>
<td>32</td>
</tr>
<tr>
<td>3.2. Private Network Addresses</td>
<td>33</td>
</tr>
<tr>
<td>3.3. Device A, B, &amp; C Networks</td>
<td>37</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>2.1.</td>
<td>Home Network Diagram</td>
</tr>
<tr>
<td>3.1.</td>
<td>VirtualBox Main Screen</td>
</tr>
<tr>
<td>3.2.</td>
<td>Ubuntu Desktop</td>
</tr>
<tr>
<td>3.3.</td>
<td>Security Onion Desktop</td>
</tr>
<tr>
<td>3.4.</td>
<td>Kali Linux Desktop</td>
</tr>
<tr>
<td>3.5.</td>
<td>VirtualBox Settings</td>
</tr>
<tr>
<td>3.6.</td>
<td>VirtualBox Network Settings</td>
</tr>
<tr>
<td>3.7.</td>
<td>Subnet Masks</td>
</tr>
<tr>
<td>3.8.</td>
<td>Ubuntu Terminal IP</td>
</tr>
<tr>
<td>3.9.</td>
<td>Security Onion Terminal IP</td>
</tr>
<tr>
<td>3.10.</td>
<td>Kali Linux Terminal IP</td>
</tr>
<tr>
<td>3.11.</td>
<td>Ping Test Device A</td>
</tr>
<tr>
<td>3.12.</td>
<td>Ping Test Device B</td>
</tr>
<tr>
<td>3.15.</td>
<td>Security Onion: Welcome Screen</td>
</tr>
<tr>
<td>3.16.</td>
<td>Security Onion: Network Interfaces</td>
</tr>
<tr>
<td>3.17.</td>
<td>Security Onion: Management Interface</td>
</tr>
<tr>
<td>3.18.</td>
<td>Security Onion: Static Addressing</td>
</tr>
<tr>
<td>3.19.</td>
<td>Security Onion: Sniffing Interfaces</td>
</tr>
<tr>
<td>3.20.</td>
<td>Security Onion: Selecting Sniffing Interface</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>3.22.</td>
<td>Security Onion: System Reboot</td>
</tr>
<tr>
<td>3.23.</td>
<td>Security Onion: Reconfigure Network</td>
</tr>
<tr>
<td>3.25.</td>
<td>Security Onion: Monitor Interface</td>
</tr>
<tr>
<td>3.27.</td>
<td>Security Onion: Password</td>
</tr>
<tr>
<td>3.28.</td>
<td>Security Onion: Confirm Password</td>
</tr>
<tr>
<td>3.29.</td>
<td>Security Onion: Implement Changes</td>
</tr>
<tr>
<td>4.1.</td>
<td>Device A UDP Attack</td>
</tr>
<tr>
<td>4.2.</td>
<td>Capture TCP Dump</td>
</tr>
<tr>
<td>4.3.</td>
<td>Device A UDP TCP DUMP</td>
</tr>
<tr>
<td>4.4.</td>
<td>Device A CPU &amp; Network Baseline UDP Attack</td>
</tr>
<tr>
<td>4.5.</td>
<td>Device A UDP #1</td>
</tr>
<tr>
<td>4.6.</td>
<td>Device A UDP DoS Attack</td>
</tr>
<tr>
<td>4.7.</td>
<td>Device A TCP SYN Flood Attack</td>
</tr>
<tr>
<td>4.8.</td>
<td>Device A TCP SYN TCP Dump</td>
</tr>
<tr>
<td>4.9.</td>
<td>Device A CPU &amp; Network Baseline TCP SYN Attack</td>
</tr>
<tr>
<td>4.10.</td>
<td>Device A TCP SYN #1</td>
</tr>
<tr>
<td>4.11.</td>
<td>Device A TCP SYN DoS Attack</td>
</tr>
<tr>
<td>4.12.</td>
<td>Device A ICMP Attack</td>
</tr>
<tr>
<td>4.13.</td>
<td>Device A TCP ICMP Dump</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>4.14</td>
<td>Device A CPU &amp; Network Baseline ICMP Attack</td>
</tr>
<tr>
<td>4.15</td>
<td>Device A ICMP #1</td>
</tr>
<tr>
<td>4.16</td>
<td>Device A ICMP DoS Attack</td>
</tr>
<tr>
<td>4.17</td>
<td>Device B UDP Attack</td>
</tr>
<tr>
<td>4.18</td>
<td>Device B UDP TCP Dump</td>
</tr>
<tr>
<td>4.19</td>
<td>Device B CPU &amp; Network Baseline UDP Attack</td>
</tr>
<tr>
<td>4.20</td>
<td>Device B UDP #1</td>
</tr>
<tr>
<td>4.21</td>
<td>Device B UDP DoS Attack</td>
</tr>
<tr>
<td>4.22</td>
<td>Device B TCP SYN Flood Attack</td>
</tr>
<tr>
<td>4.23</td>
<td>Device B TCP SYN TCP Dump</td>
</tr>
<tr>
<td>4.24</td>
<td>Device B CPU &amp; Network Baseline TCP SYN Attack</td>
</tr>
<tr>
<td>4.25</td>
<td>Device B TCP SYN Attack #1</td>
</tr>
<tr>
<td>4.26</td>
<td>Device B TCP SYN DoS Attack</td>
</tr>
<tr>
<td>4.27</td>
<td>Device B ICMP Attack</td>
</tr>
<tr>
<td>4.28</td>
<td>Device B TCP ICMP Dump</td>
</tr>
<tr>
<td>4.31</td>
<td>Device B CPU &amp; Network Baseline ICMP Attack</td>
</tr>
<tr>
<td>4.32</td>
<td>Device B ICMP #1</td>
</tr>
<tr>
<td>4.33</td>
<td>Device B ICMP DoS Attack</td>
</tr>
</tbody>
</table>
Chapter I: Introduction

Introduction

This work takes a closer look into the functionality and efficiency of a prebuilt, open source, security tool, known as the Security Onion. The Security Onion was selected as the system of choice for this experiment based on the numerous different kinds of tools that are integrated into its design. Many security systems don't incorporate numerous tools into the design, making it a unique system to analyze. The experiment will be conducted using two different computers. The first computer will be a baseline device, that won't have Security Onion installed onto it. The second device will have the Security Onion installed and running on the machine. Each of these machines will have the same kinds of attacks performed on them. The outcomes of each attack will be tracked and documented. The results, from the attacks, will then be used in a comparative analysis to determine the effectiveness of the Security Onion. The findings from the project will then be concluded, and future work will be addressed.

Problem Statement

Security and network professionals are tasked with securing an enterprises network and resources. However, many organizations fail to provide them with the necessary budget, guidance, or resources. Therefore, these professionals decided to use opensource tools, like the Security Onion, to secure their network; without fully understanding the effectiveness or ineffectiveness of the tool. Thus, this work aims to examine the efficiency of the security tool. It will utilize a variety of software, to launch a series of distributed of denial, or DoS attacks, against it. DoS attacks are used by an attacker to flood a device’s system, which can make it unable the user depending on the
magnitude of the attack. These attacks will then be documented, examined, and evaluated to provide professionals with further insight.

**Nature and Significance of the Problem**

Monitoring and securing a network is a daunting task, as new threats evolve and attack daily. Several security tools, such as the Security Onion, have been created to help users identify and protect their networks. However, the question is how efficient or effective are these tools? Users and professionals install them with the expectation that they will identify and alert them to significant problems or issues. Although these tools aren’t a cure-all. The users of these tools need to be aware of the true effectiveness and realize that their system may not be as secure as they believe it to be. Also, that their networks and systems may require more attention and monitoring than just simply installing a security tool and expecting it do the work.

**Objectives of the Research**

The objective of this research is to provide a comparative analysis of a device with the Security Onion installed, and one without it. It is to provide users with greater insight into how effective or ineffective a security tool may actually be. This could potentially help them realize, or understand, the pros and cons of a security tool and how secure their network truly is.

**Research Questions**

There are several research questions addressed in the project. The first is how complex it is to install and utilize the Security Onion? The second is how efficient is the Security Onion at monitoring and securing a network? The third is what should
security and network professionals understand about using the Security Onion as an alert tool? The last question is how does the Security Onion withstand DoS attacks?

Limitations of the Research

This study is limited to the comparative analysis of the Security Onion, although, there are other opensource security tools available for enterprises to utilize. Moreover, this work attempts to launch attacks on each computer to analyze the effects. However, the complexity of the attacks is restricted, and doesn’t include all variations of protentional threats or vulnerabilities.

Definition of Terms

Security Onion: is “a free and open source Linux distribution for intrusion detection, enterprise security monitoring, and log management. It includes Elasticsearch, Logstash, Kibana, Snort, Suricata, Bro, OSSEC, Sguil, Squert, NetworkMiner, and many other security tools” (“Security Onion,” n.d.).

Elasticsearch: is “an open source distributed, RESTful search and analytics engine capable of solving a growing number of use cases” (“Elasticsearch,” n.d).

Logstash: is “an open source, server-side data processing pipeline that ingests data from a multitude of sources simultaneously, transforms it, and then sends it to your favorite stash” (“Logstash,” n.d.).

Kibana: is a software that allows users to manipulate their data through differing interactive visualizations. This would include “histograms, line graphs, pie charts, sunbursts, and more” (“Kibana”, n.d.). It is easy to use and conveniently lets users share their results (“Kibana,” n.d.).
**Snort:** is “an open source intrusion prevention system capable of real-time traffic analysis and packet logging” (“Snort,” n.d.).

**Suricata:** “is a free and open source, mature, fast and robust network threat detection engine” (“Suricata,” n.d.).

**Bro:** is considered to be a network security monitor, that is adaptable, efficient, highly stateful, flexible, and open source. It provides in-depth analysis, along with forensic and open interface capabilities” (“The Bro,” n.d.).

**Open Source Host-based Intrusion Detection System (OSSEC):** “is a scalable, multi-platform, open source Host-based Intrusion Detection System (HIDS). It has a powerful correlation and analysis engine, integrating log analysis, file integrity checking, Windows registry monitoring, centralized policy enforcement, rootkit detection, real-time alerting and active response” (“OSSEC,” n.d.).

**Sguil:** is a software that is designed for network security analysts. It was designed to with a graphical user interface (GUI), “that provides access to realtime events, session data, and raw packet captures” (Visscher et al., n.d.).

**Squert:** “is a web application that is used to query, and view event data stored in a Sguil database (typically IDS alert data).” “It attempts to provide additional context to events through the use of metadata, time series representations and weighted and logically grouped result sets” (“The Squertproject,” n.d.).

**NetworkMiner:** “is an open source Network Forensic Analysis Tool (NFAT) for Windows.” It “can be used as a passive network sniffer/packet capturing tool . . . to detect operating systems, sessions, hostnames, open ports etc. without putting any traffic on the network” (“NetworkMiner,” 2018).
Device A: The baseline computer used in this study, which doesn't possess the installation of the Security Onion.

Device B: The comparative computer used in this study, which has the Security Onion installed and running on the device.

Device C: The neutral computer used to launch attacks and document the results.

Network: A collection of computers that are connected to one another through cables or wirelessly. It allows computers to quickly communicate with one another and share resources, through an established IP connection.

Artificial Intelligence (AI): devices that are capable of simulating the thought and learning process of humans, without continuous alterations or modifications.

Graphical User Interface (GUI): a user-friendly way in which an average person can interact with a computer. The user interfaces are often created by utilizing different lines, shapes, colors, text, and pointing devices (Levy, 2018).

National Institute of Standards and Technology (NIST): NIST is a division within the U.S. Department of Commerce. It was established in the year 1901 and has evolved “to promote U.S. innovation and industrial competitiveness by advancing measurement science, standards, and technology in ways that enhance economic security and improve our quality of life” (Hernandez, 2018).

hping3: hping3 is a network tool that can be used to test perform a variety of tests against the network.
Summary

This research aims to provide professionals with a clear insight into how effective or ineffective a security tool, such as the Security Onion, may be. The test performed, using the Security Onion, outlines the results and conducts a comparative analysis to identify the effectiveness. It attempts to help inform professionals as to how secure their system is, if they only utilize the Security Onion to protect their network.
Chapter II: Background and Review of Literature

Introduction

This section provides additional content relating to the background of the problem; including the primary use of the Security Onion to protect an enterprise’s network. It introduces the common problems or challenges that exist in the security world and the ways in which it impacts a professional’s decisions to use an open source system, such as the Security Onion. It looks at literature to indicate the current understanding of the system and identify what’s missing. Finally, it introduces the concept of a private network and different types of attacks.

Background Related to the Problem

The area of cybersecurity has become a growing commodity for companies over the last decades. Organizations are starting to realize the significance of having a strong security team. However, the rapid growth in this field has led to numerous difficulties. The first being the lack of professionals within the industry. According to Jeff there will be a “2 million global shortage of cybersecurity professionals by 2019” (Kauflin, 2017). This is contributing to the increasing demand within the field and the lack of skilled employees. A study shown in a Silence report stated that “84% [of] organizations believe half or fewer of applicants for open security jobs are qualified” (“Addressing,” n.d.).

Another difficulty related to cybersecurity is the costs or funding required to run the department. In order to have a secured enterprise, organizations have to provide a proper amount of funding. This is often problematic, since top managers will distribute the funds without fully understanding the complexity and needs of this portion of the
This type of funding is used for employees, hardware, software, trainings, security tools, and recovery budgets. For instance, according to Barkly, “[i]t’s predicted that by 2021, cybercrime will cost the world $6 trillion annually” (The Barkly Team, 2018). In addition, “[r]ansomware damage costs alone are on track to hit $11.5 billion in 2019, at which point it’s estimated that a business will fall victim to a ransomware attack every 14 seconds” (The Barkly Team, 2018).

This leads into the next major difficulty, referring to the growing number of cyberattacks. Although, it is often quite hard to identify the true number of attacks that occur each year, a “Cyber Incident & Breach Trends Report,” by Online Trust Alliance (OTA), indicates that “cyber incidents nearly doubled to 159,700 globally” (Wilbur, 2018). However, it has been projected that the number of cyber incidents could possibly surpass 350,000 instances, with this number expected to further increase each year. Although these statistics only include the following areas: “ransomware, business email compromise (BEC), distributed denial-of-service (DDoS) attacks and connected devices vulnerability” (Wilbur, 2018). This is limiting the number of actual attacks that may occur each year by a significant amount.

Moreover, the increase in the number of attacks has also become more severe as hackers are starting to incorporate a variety of tactics into a single attack. This is allowing them to penetrate more networks and gain assess. In addition, these attacks are becoming more destructive in nature and in the instance of ransomware, even more expensive. These attacks are expected to only become worse as attackers learn new tactics and start to integrate artificial intelligence (AI) into their strategies (Vizard, 2018).
These three difficulties (professionals, costs/funding, and increase in attacks) are why tools, like the Security Onion, are being implemented into enterprises. The Security Onion is a free and open source software that can be downloaded and used by any potential user. It is created to provide “intrusion detection, enterprise security monitoring, and log management” (“Security Onion,” n.d.). It has been built to include the following tools: “Elasticsearch, Logstash, Kibana, Snort, Suricata, Bro, OSSEC, Sguil, Squert, NetworkMiner” and etc. (“Security Onion,” n.d.). It is designed to monitor alert data, asset data, full content data, host data, session data, and transaction data. All while utilizing a graphical user interface, for easy use.

The Security Onion allows enterprises to automate and control the security process, which can potentially help a department that is lacking proper man and women power. In addition, it is a cost-effective solution since the software is prebuilt and free. Furthermore, the software is comprised of numerous security tools. These tools are designed to work together to help create a more secure network (Vizard, 2018).

On the surface the Security Onion looks like a promising tool that could solve all of an organization’s security problems. However, professionals need to be subjective when deciding on the proper way to secure their company. Just because a software looks promising by description doesn’t mean that it will be the only tool required for security. Tools that provide automation also need a form of manual work to make it more effective. This is why professionals need to question the security measures they choose and analyze their true effectiveness.

If security professionals fail to analyze their security tools and identify their true effectiveness, they may be putting their organization in harm’s way. This is due to
exposing the company to vulnerabilities and threats, which they believe they are safe from. Nevertheless, this is a continuous effort as new attacks evolve, and new tactics are incorporated into their design. Thus, finding new ways to penetrate an organization's system and wreak havoc are a constant threat.

**Literature Related to the Problem**

The Security Onion is a relatively new concept in the cybersecurity world. When reading through different articles, books and journals, there seems to be a lack of information relating to the effectiveness or ineffectiveness of the software. There were many pieces of literature relating to the setup of the system and different ways of configuring it to help prevent certain types of attacks.

The paper “Intrusion Detection System and Intrusion Prevention System with Snort provided by Security Onion,” by Bezborodov Sergey, analyzes the utilization of Snort with and without the Security Onion. It walks through the installation and configuration of the Security Onion, with Snort. It also addresses the setup of Snort on its own. It then compares the two to identify the advantages and disadvantages between using Snort as part of the Security Onion, or as its own individual software (Sergey, 2018).

There were several advantages outlined within the paper. First, the integrated version of the Security Onion and Snort allows for flexibility as users can alter settings and configurations to their personal needs. Second, the integrated version comes preconfigured, for easy use. Third, when looking at the use and installation of Snort, without the Security Onion, it incorporates a more in-depth analysis of data. It takes
more time to configure the software but allows the user to view all packets and scripts (Sergey, 2016).

Moreover, there were several disadvantages identified. First, the integrated version of the Security Onion and Snort isn't configured to work as an intrusion prevention system (IPS) it was built to utilize intrusion detection system (IDS). Second, the integrated version creates complications for setting up the network. If the user wants to use Wi-fi, they have to set it up manually, without the use of the Wizard installation. Third, the integrated version was created without full backups, thus, the user must utilize another software. Fourth, the use of Snort on its own, without the Security Onion, requires a specific level of expertise. The user must have proper knowledge of Linux, to effectively configure and build different rules to execute in the software (Sergey, 2016).

Furthermore, another paper highlighting the use of the Security Onion is “Logging and Monitoring to Detect Network Intrusions and Compliance Violations in the Environment,” by Gupta Sunil. It introduces the concept of security within organizations, of all sizes, and the challenges of meeting “the operational, audit and security needs.” (Sunil, 2012). The paper walks through the different software that is incorporated into the Security Onion system. It then shows how to utilize the different software “to build a system that combines Network Based Intrusion Detection with Log Based Intrusion Detection to create a comprehensive security monitoring platform” (Sunil, 2012).

The paper breaks the installation of the Security Onion system into several main categories. The first deals with configuring the OSSEC through specific rules, to act as a log collector. The second is the configuration of “NIDS Sguil/Snort sensor[s]” and “LIDS Sguil/Snort sensor[s]” through different rules (Sunil, 2012). Third is configuration
of the Sguil server through different file setups. Fourth pertains to “log analysis and correlation”, which is setup through “event analysis”, “database query”, “event correlation”, and “auto categorization” (Sunil, 2012). This allows for the alerts to be collected and examined to determine their severity. The alerts are then stored in a database, along with the different logging of activities. The stored data can then be reanalyzed and further examined to help pinpoint trends in the system and identify were an attack, or significant event, may have occurred (Sunil, 2012).

Fifth is “log alerting and reporting”, which relates to “alert classification and prioritization”, “email alerts”, “Sguil reporting” and “Snorby reporting” (Sunil, 2012). The “alerts classification and prioritization” are done with current data. It utilizes rules to identify the severity of the attack. Depending on how the attack is identified and stored in the system, different alerts are sent to the user. The email alerts are then created and sent by the different software within the Security Onion System, including: Sguil, OSSEC and Snorby. Furthermore, the Sguil report provides a basic plain text report for the user, providing the bare minimum. Meanwhile, the Snorby report is integrated with a GUI to provide a more user-friendly report. The paper’s focus is on how the Security Onion can be configured and used to meet an organization’s needs. However, it once again fails to show the results of these configurations. It blinds professionals from looking at the use of the system from all angles. It focuses only on the positive, without any true representation of any faults. However, the system may be a strong candidate that servers the user’s need, but without any representation of this data, how does the user know the true effectiveness of the system (Sunil, 2012)?
Nevertheless, another paper titled “Information Security Case Study with Security Onion at Kajaani UAS Datacentre Laboratory,” by Raimo Heikkinen, analyzes the use of the Security Onion compared to a NIST guideline. The following is the intrusion detection and prevention system standard, provided by NIST:

Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices. An intrusion detection system (IDS) is software that automates the intrusion detection process. An intrusion prevention system (IPS) is software that has all the capabilities of an intrusion detection system and can also attempt to stop possible incidents. IDS and IPS technologies offer many of the same capabilities, and administrators can usually disable prevention features in IPS products, causing them to function as IDSs. (“Intrusion,” n.d.)

The Security Onion was then downloaded and utilized to identify whether or not it met these standards. The project concluded that the Security Onion is capable of monitoring through a GUI interface (Heikkinen, 2018). It collects data from alerts, stores full packet capture, and tracks different trends. It is categorized as an IDS system, that uses the trends and events stored to create reports and alert the user. This is often done through several different types of email alerts that can be setup through configuration. Thus, it meets the requirements of the NIST guidelines. However, the paper also concluded that it improved the security of the system, by running it on top of already functioning security tools. Along with the fact that the Security Onion requires a
combination of automation and manual labor. Professionals must be willing to analyze and review the outputs of the system on a regular basis to provide effective results (Heikkinen, 2018).

**Fundamentals of a Network**

In order to understand the concept of a private network, one must first have a proper understanding of what a network is. As stated in the definition portion of the paper, a network is a collection of computers that are connected to one another through cables or wirelessly. It allows computers to quickly communicate with one another and share resources, through an established IP connection. There are many different types of networks including the following: Local Area Network (LAN), Wide Area Network (WAN), Wireless Local Area Network (WLAN), Metropolitan Area Network (MAN), Storage Area Network (SAN), Campus Area Network (CAN), and Personal Area Network (PAN). These are just a few of the more popular types of networks. They vary depending on the build of the connection, referring to cables or wireless. In addition to the radius of the signal that it distributed to its users. However, the two most popular types of networks include LAN and WAN (Mitchell, 2018).

![Home Network Diagram](image)

*Figure 2.1. Home Network Diagram (“Home Network,” n.d)*
The framework of a network is comprised of three main parts: a router, a modem, and a switch, this is shown in Figure 2.1. The best way to look at the figure is from left to right. The average person has multiple devices that connect to a network, whether that be a phone, printer, computer, laptop, cellphone or etc. In order for them to connect to the network each of these devices are assigned a unique identifier, often referred to as an IP address. The IP addresses are assigned by a router. A router continuously searches for devices to provide a connection. This can be done either wirelessly or through an ethernet cable (“IP Addressing,” 2010).

The router is then connected to a modem. This is often provided by an outside company, such as Charter Spectrum, Comcast Xfinity, Cox Communications, Frontier Communications, Mediacom, etc. It provides the router with a connection to the internet. Furthermore, between the internet and the router there is usually a firewall, which helps prevent unwanted traffic from the internet (“Networking Basics,” n.d.).

The combination of these devices is what creates a network for them to communicate on. The router allows the devices to connect and communicate with one another. Meanwhile, the switch is integrated into the router, identifying the different communications among the devices. It functions as a stoplight, directing the proper traffic to and from the correct device. This is a basic understanding of how a general homebased network connection would be setup.

**Private Network**

Private networks can be useful, when a user doesn’t want their IP address to be publicly displayed. Or, when they just want to communicate among devices that are on the same physical network. It allows a user to protect and contain the information being
sent between the private connection, without requiring an internet connection. The actual configuration and setup of a network will be further elaborated on in the methodology chapter.

**Types of Attacks**

There are numerous types of attacks that are launched at devices daily, in an attempt to penetrate a system. Each of these attacks are created for a unique purpose and can cause a vast variety of symptoms. This could be anything ranging from DDoS to ransomware to a phishing email. Depending on the creator's knowledge base and expertise, these attacks may be generated from scratch, by reusing code, or they could be launched using a software. There are different types of software that can be downloaded and used to project these attacks, they are commonly known as penetration testing frameworks. They are created to help people and organizations test the security of their structure. One major platform that is commonly installed and used is known as Kali Linux. It possesses a wide variety of tools, including: information gathering, vulnerability analysis, exploitation attacks, wireless attacks, forensic tools, web applications, stress testing, sniffing and spoofing, password attacks maintaining access, hardware hacking, reverse engineering, and reporting tools (“Kali,” n.d.).

One of the primary tools utilized in the project is hping3. It is a tool that is built into the Kali Linux platform. Although it is a free opensource tool that anyone can download. It is designed with a TCL language that allows it to be integrated into a device’s terminal. Numerous types of attacks can be launched with this tool pertaining to items such as firewalls, routing protocols, and TCP/IP (“Hping,” n.d.). For this work, the tool will be used to flood a system and create a DoS attack.
Denial of service (DoS) or distributed denial of service (DDoS) attacks are extremely popular and fairly easy to conduct. A DoS attack is “an attack which the attacker or attackers make an effort to make a service or resource out of access” (“Analysis,” 2017). Meanwhile, a DDoS is a more secure way of performing the attack, while remaining undetected. There are numerous kinds of DoS/DDoS attacks, including: flooding attacks and logical attacks. Flooding attacks could be a SYN flood, UDP flood, ICMP flood, ARP flood, Xmas tree or a Rest flood. Logical attacks could be a ping of death, teardrop attack, or a land attack (Dzurenda et al., 2015).

**Summary**

This chapter has outlined the main challenges that have led professionals to use free software, tools, and systems, such as the Security Onion. It examined previous research to illustrate what is known about the Security Onion and what is yet to be discovered. Thus, showing that the general information provided on the system focuses primarily on configuration. However, it fails to truly examine the effectiveness or ineffectiveness of the system. It merely portrays the Security Onion as a cure-all for security professionals.
Chapter III: Methodology

Introduction

This section focuses on the process and steps that are used to conduct the project. It outlines the configuration of the Security Onion and the setup of the baseline computer. It will then walk through the different attacks that are to be launched, at the two machines, to track and analyze the results.

Design of the Research

This research is intended to evaluate the Security Onion system and show its response to three different kinds of denial of service or DoS attacks. It is to provide valuable insight into the effectiveness or ineffectiveness of the system, in an attempt to help inform security professionals. The project is conducted using two primary devices. Device A will be the baseline computer, while Device B will be the altered unit that possesses the Security Onion system. A third device, referred to as Device C, will then be used to launch different attacks at the two devices, and document their responses.

Setting up the devices. This project utilizes VirtualBox to setup the different devices that will be used in the experiment. It can be downloaded from the following URL address https://www.virtualbox.org/wiki/Downloads. There are different versions and types that can be downloaded depending on the user’s operating system. Devices A will be configured to Ubuntu 16.04.03 desktop version. After downloading, installing, and opening the software it should resemble Figure 3.1, which is shown below.
The devices used in this project are then setup with VirtualBox. Device A is configured with an Ubuntu 16.04.3 desktop version disk image file. The download for this disk image file can be found at the following URL address https://www.ubuntu.com/download/alternative-downloads. After creating a new machine and walking through the configuration process, the main screen for the Device A should resemble Figure 3.2, which is shown below.

Device B is then created within VirtualBox and is configured with the Security Onion 16.04.05.2 disk image file. The disk image file can be found at the following URL address https://github.com/Security-Onion-Solutions/security-onion/blob/master/Verify_ISO.md. Once the virtual machine has been created and the disk image has been uploaded the desktop for the Security Onion will look like Figure 3.3, shown below.
Figure 3.2. Ubuntu Desktop

Figure 3.3. Security Onion Desktop
Device C is the last device that needs to be created in VirtualBox. It will be configured with a Kali Linux 2018.3 disk image file, which can be found at the following URL address https://www.kali.org/downloads/. The main desktop screen for this device will look like the screen shown below in Figure 3.4

![Figure 3.4. Kali Linux Desktop](image-url)

Each of these machines were configured in VirtualBox differently depending on the different disk image requirements. Thus, some have more RAM, CPUs, and memory designated to them. After setting up the machines, an internal network was created to contain the DoS attacks.

**Establishing a private network.** Creating a private or internal network is an important part of this experiment. It ensures that the attacks being launched between the machines will be contained and not ejected at an external IP address. The act of launching attacks at an IP address, in which permission hasn’t been granted, is illegal.
The first step in creating an internal network is through the settings found within VirtualBox. In order to do this, the selected virtual machine must be in a *Powered Off* state. The status of the machine will appear just below the name of the machine. If the machine is running, it needs to be powered off. If the machine is in a *Saved* state, it will need to be powered on and then powered off. After checking the state of the machine, it should be selected and will appear highlighted in a blue color compared to the other machines in the list. From there the *Settings* button, on the top of the screen, is selected and a settings box appears, as shown below in Figure 3.5.

![VirtualBox Settings](image)

**Figure 3.5. VirtualBox Settings**

Next, the *Network* tab is selected on the right-hand side, which then displays the different network adapters that are available. For each of the devices the *Attached to:* is switched from *NAT* to *Internal Network.* In this experiment, a *Thesis* internal network was created and the *Name:* is set to it. This is illustrated below in Figure 3.6
The changing of the network settings is done for Device A, Device B, and Device C. It is the first step in creating the internal network required for this experiment. Once the devices have all been configured properly in VirtualBox, each machine is booted, and the IP addresses are setup manually within each machine’s network settings.

The proper way of configuring an IP address depends on the number of devices that are going to connected to the network. There are universal guidelines and rules for IP addresses that break them into different classes, as shown below in Table 3.1.

Table 3.1. IP Address Classes (“What is,” 2018)

<table>
<thead>
<tr>
<th>Class</th>
<th>Address Range</th>
<th>Supports</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A</td>
<td>1.0.0.1 to 126.255.255.254</td>
<td>Supports 16 million hosts on each of 127 networks</td>
</tr>
<tr>
<td>Class B</td>
<td>128.1.0.1 to 191.255.255.254</td>
<td>Supports 65,000 hosts on each of 16,000 networks</td>
</tr>
<tr>
<td>Class C</td>
<td>192.0.1.1 to 223.255.254.254</td>
<td>Supports 254 hosts on each of 2 million networks</td>
</tr>
<tr>
<td>Class D</td>
<td>224.0.0.0 to 239.255.255.255</td>
<td>Reserved for multicast groups</td>
</tr>
<tr>
<td>Class E</td>
<td>240.0.0.0 to 254.255.255.254</td>
<td>Reserved for future use, or Research and Development Purposes</td>
</tr>
</tbody>
</table>
Out of these classes, Class A, Class B, and Class C are the ones that are primarily utilized (“What is,” 2018). They each have corresponding subnets, depending on the class, which is shown below in Figure 3.7.

<table>
<thead>
<tr>
<th>Class A Subnet Mask</th>
<th>255.0.0.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class B Subnet Mask</td>
<td>255.255.0.0</td>
</tr>
<tr>
<td>Class C Subnet Mask</td>
<td>255.255.255.0</td>
</tr>
</tbody>
</table>

**Figure 3.7. Subnet Masks**

The different classes and subnets provide the general information required to configure each of the networks within the machine. However, RFC 1918 identities specific address ranges that have been created for private networks (Rouse, n.d.). Each address then falls into a previously identified classes and will correspond to one the subnets. These IP addresses can be found below in Table 3.2.

**Table 3.2. Private Network Addresses (Soskinsky, 2001)**

<table>
<thead>
<tr>
<th>Class</th>
<th>Address Range</th>
<th>Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A</td>
<td>10.0.0.0 – 10.255.255.255</td>
<td>10/8 prefix</td>
</tr>
<tr>
<td>Class B</td>
<td>172.16.0.0 – 172.31.255.255</td>
<td>172.16/12 prefix</td>
</tr>
<tr>
<td>Class C</td>
<td>192.168.0.0 – 192.168.255.255</td>
<td>192.168/16 prefix</td>
</tr>
</tbody>
</table>

In this project, a private ten IP address is applied to the network settings for each device. The IP addresses increment by one for each of the devices, to ensure that there aren’t any overlapping connections. Then the proper corresponding subnet is used, and the gateway is applied. The private addresses that are used are universal and can be
used by any person or organization. Since they are built to be maintained in the intranet connection created between the devices on that specific router (“IP Addressing,” 2010).

The configuration of the IP addresses is then tested in the terminal of each of the virtual machines. This is done by opening a terminal window, which is a different process for each of the machines. Once the terminal in open, the following command is executed to switch to a root user:

```
sudo -i
```

After executing the command, the terminal prompts the user for their Ubuntu administrative/root password. Once the proper password has been entered, it will process the command and change the user to root. From there, the following command is entered to display the machines network configurations:

```
ifconfig
```

These commands can be executed on Device A, Device B, and Device C, in the proper terminal, since all the virtual machines are Linux based. Below, Figure 3.8 shows the results of these commands ran in Device A’s Terminal. It indicates that there are two network connections currently setup on the Ubuntu device. The first interface is enp0s3 and the second is lo. For the purpose of this experiment, enp0s3 will be the network of focus. It becomes evident that the enp0s3 network is an ethernet connection. In addition, the IP address for the network connection is 10.10.0.10 and the subnet mask is 255.0.0.0. Running this command in the terminal ensures that the configurations made in the network settings, on the device, have been executed and saved properly to the system.
For Device B, the same commands were ran in the Security Onion’s terminal.

Figure 3.9 (below) shows the results that were displayed. The output in the Security Onion terminal include numerous network interfaces and their configurations. However, enp0s8 is the primary interface that is used in this project. Thus, Figure 3.9 only displays a small snippet of all of the outputs. From the outputs it is easy to conclude that the IP address assigned to the network interface is 10.10.0.11 and the subnet mask is 255.0.0.0. Along with the fact, that this interface is also an ethernet connection.
Lastly, the commands were ran in Device C’s terminal. The following figure, Figure 3.10, shows the results that were displayed on the Kali Linux machine. Like Device A, this machine only had two network interfaces, eth0 and lo. The eth0 network interface is that network of focus. It is an ethernet connect, the same as Device A and Device B. The network interface has an IP address of 10.10.0.13 and its subnet mask is 255.0.0.0.
Now that the internal networks have been configured and checked in the terminals, they need to be tested to determine whether or not they can communicate with one another over the internal network.

**Ping test.** Ping test is required to test the connection. If there isn’t a proper connection formed between the devices, they will be unable to communicate, making it impossible to conduct the experiment. Device C, or the Kali Linux device is going to be the device used to launch the attacks. Thus, it will be the device used to test the connections between itself and the other two devices.

A ping test is conducted in the terminal of the machine. It requires the knowledge of another device's IP address to test the connection. The command executed in the terminal resembles the following format:

```
ping xxx.xxx.xxx.xxx
```

In the above ping format, the x’s represent the IP address of the device that is to be contacted. In the previous section the IP addresses were defined for each of the devices; a summary of this information can be show below in Table 3.3.

<table>
<thead>
<tr>
<th>Device</th>
<th>Network Interface</th>
<th>IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device A</td>
<td>enp0s3</td>
<td>10.10.0.10</td>
</tr>
<tr>
<td>Device B</td>
<td>enp0s8</td>
<td>10.10.0.11</td>
</tr>
<tr>
<td>Device C</td>
<td>eth0</td>
<td>10.10.0.13</td>
</tr>
</tbody>
</table>

First, the connection between Device A and Device C is tested. In Device C’s terminal the following command is executed:

```
ping 10.10.0.10
```
Figure 3.11 (below) illustrates the results of the command, as shown in the Kali Linux terminal.

![Ping Test Device A](image)

**Figure 3.11. Ping Test Device A**

From Figure 3.11 it can be concluded that Device C (Kali Linux) was able to communicate with Device A (Ubuntu) over the internal network. This becomes evident as six packets were sent by Device C, and six packets were received by Device A without any packet loss or error messages.

The ping test then needs to be conducted one more time to test the connection between Device B and Device C. However, this time the command executed in Device C’s terminal is a bit different, as show below:

```
ping 10.10.0.11
```

The results gathered from executing this command can be found below in Figure 3.12. It illustrates the outcome of the ping test on Device B (Security Onion). Where Device C (Kali Linux) sent four packets to Device B, which were successfully received. It is evident that Device C (Kali Linux) is able to communicate with Device B without any complications.
After the three devices have been configured to a private network and the connections have been tested with a ping test, Device B will be setup with the Security Onion System. The deployment of the Security Onion system will follow the setup and configuration steps found in the book “The Practice of Network Security Monitoring” by Richard Bejtlich. The following steps will be taken to equip Device B with the Security Onion system.

In the previous sections the Security Onion disk image file was downloaded from the GitHub webpage and was loaded into a virtual machine. When booting the Security Onion machine from VirtualBox it will prompt the system to boot as a live system, which will display a GUI (Bejtlich, 2013). To start the setup of the system, the Setup icon needs to be selected from the desktop. This step is shown below in Figure 3.13.

**Figure 3.12. Ping Test Device B**
After selecting the **Setup** icon, the user is prompted with a screen that requests a password. The password used for this step would have been setup during the initial configuration of the system when it was uploaded into the VirtualBox machine. The password is then entered into the box and the **OK** button is selected (Bejtlich, 2013). This step can be seen below in Figure 3.14.
Once the OK button has been selected the next screen welcomes the user to configure a list of services. The Yes, Continue! button is selected to start the configuration process. This screen can be viewed in Figure 3.15, which is displayed below. The box that appears on the screen asks the user if they would like to configure the network interfaces. The Yes, configure /etc/network/interfaces! button is selected to advance to the next step. The network interfaces step is illustrated below in Figure 3.16.

The next step in the process is to identify which network interface will be used for the management interface. When setting up the virtual machine in VirtualBox there were two network adapters assigned the Security Onion machine. These adapters were automatically assigned a name, one is enp0s3 and the other is enp0s8. They can be viewed, below, in Figure 3.17, where the management interface needs to be identified. In this project enp0s3 was selected to serve as the network management interface and the OK button was clicked (Bejtlich, 2013).

![Figure 3.15. Security Onion: Welcome Screen](image)
After selecting the management network interface, the next step is to decide whether or not the addressing will be static or DHCP. In this project *DHCP* is selected for the addressing type and *OK* will be selected to continue. This step is shown below in Figure 3.18.
Now that the management network interface has been selected and a DHCP type has been assigned. The next screen will ask the user if they’d like to configure a sniffing interface (Bejtlich, 2013). The Yes, configure sniffing interface. is selected, which can be viewed below in Figure 3.19.

The next step is to identify which interface will be used for the sniffing interface. When configuring the management interface, the enp0s3 was assigned to be the interfaced used. This leaves enp0s8 to be setup as the sniffing network interface. This step is shown below in Figure 3.20.
Figure 3.20. Security Onion: Selecting Sniffing Interface

After selecting the OK button, a screen is displayed showing the changes that were made in the last few steps. It asks the user if they would like to make the changes. The Yes, make changes! button is selected, which is shown blow in Figure 3.21.

Figure 3.21. Security Onion: Verify Network Changes

This will complete the network configuration part of the setup. In which case the next screen displayed will ask the user to reboot before continuing on with the second
part of the setup process. The *Yes, reboot!* button is selected, as shown below in Figure 3.22. The virtual machine will then reboot.

After the reboot has finished the *Setup* icon on the desktop is selected to finish the setup process. The same screen, which was shown above, in Figure 3.15, will be displayed. At which point the *Yes, Continue!* button is selected. The next screen will ask the user if they would like to reconfigure the network or skip it. Since, it was already configured in the previous steps, the *Yes, skip network configuration!* is clicked, as shown below in Figure 3.23.

![Security Onion Setup (securityonion-virtualbox)](image)

**Figure 3.22. Security Onion: System Reboot**

![Security Onion Setup (securityonion-virtualbox)](image)

**Figure 3.23. Security Onion: Reconfigure Network**

The next step is to determine what kind of mode will be used. There are two choices: evaluation mode and production mode. Since this machine will never be used
in a production deployment environment, the *Evaluation Mode* is selected, and the *OK* button is clicked to continue (Bejtlich, 2013). This step can be shown below in Figure 3.24.

![Security Onion Setup (securityonion-virtualbox)](image)

**Figure 3.24. Security Onion: Mode Selection**

Once the *OK* button has been selected, the user will be asked what interfaces should be monitored. The network interface that is of primary focus in this project is enp0s8. This interface should be selected, and the *OK* button should be pushed (Bejtlich, 2013). This step can be shown below in Figure 3.25.
Figure 3.25. Security Onion: Monitor Interface

After selecting the monitor interface, the setup process will ask the user to create an account for accessing Kibana, Squert, and Sguil. It will first ask the user to create a username, as shown below in Figure 3.26.

Figure 3.26 Security Onion: Username

Once a username has been selected, a password is created to setup the account. A password will need to be entered and then entered again to verify that they are the same (Bejtlich, 2013). This step is show below in Figure 3.27 and Figure 3.28.
When the requirements for the username and password have been completed, the next screen will ask the user if they are okay with the changes that are about to be made. The following Figure 3.29 shows this step. The *Yes, proceed with the changes!* is then selected to implement them. This will then start the processing of the setup, which takes a few minutes to complete. Once the setup has finished a series of boxes will appear, one after another, as the user clicks *OK* to proceed to the next one. These boxes indicate: where system logs are stored, where IDS alerts can be viewed, where to use Bro, different commands, rules, and information on the firewall settings (Bejtlich, 2013).
This completes the configuration of the Security Onion system for the use of this project. Showing the baseline for how the system was configured can provide others with the proper knowledge to understand the results of the experiment.

**DoS attacks.** Device A, Device B and Device C have been setup with a private network. In addition, Device B has been equipped with the Security Onion system. The next step in the methodology is to identify the attacks that will be launch at Device A, the baseline device, and Device B, the Security Onion device.

Device C will be used to launch the attacks at Device A and Device B, it is configured with Kali Linux, which is configured with numerous tools that can be used for penetration testing. In this project, the hping security tool will be utilized to test the efficiency of the system against three different kinds of DoS attacks. These attacks are ran in the Kali Linux Terminal as an executable command.

The first attack is a UDP DoS flood attack. The following is the format of the command that is used to execute the attack:
In the above format the xs represent IP addresses and the #s represent the port. The first IP address in the format is the spoof address that will display as the source address in the TCP/UDP dump. The port that will be used in this project is 6234 and the last IP address in the format will be the IP address of the target device. Thus, the last IP address will either be 10.10.0.10 for Device A, or 10.10.0.11 for Device B (“DoS Attack,” n.d.).

The second attack is a TCP SYN flood DoS attack. The following is the format of the command that is used to execute the attack:

```
hping3 xxx.xxx.xxx.xxx -q -n -d 120 -S -p ## --flood --rand-source
```

In the above format the xs represent the targeted IP address, which once again will either be 10.10.0.10 for Device A or 10.10.0.11 for Device B. Furthermore, -d 120 represents that size of the packet being sent. The -S indicates that it is only sending SYN packets. The ## following the -p is the destination port. Lastly, the --flood --rand-source means that it will flood the targeted device with the packets and the sources IP address will be randomly generated (“Denial-of-service,” 2015).

The third attack is an ICMP DoS attack. The format of the attack is as follows:

```
hping3 xxx.xxx.xxx.xxx -q --icmp -C 3 -K 4 --faster
```

In the above format the x's once again resemble the IP address of the targeted address, being either 10.10.0.10 for Device A or 10.10.0.11 for Device B. This attack floods the targeted system with ICMP packets, which are coming from the source IP address of Device C, 10.10.0.13 (“Denial-of-service,” 2015).
**Conclusion.** Once Device A, Device B and Device C have been setup in VirtualBox and have been configured to a private internal network. The setup configuration steps are completed on the Device B (Security Onion) machine. Lastly, Device C (Kali Linux) is utilized to launch different DoS attacks at Device A and Device B. The results of the attacks are then collected and analyzed to reveal the research findings.

**Data Collection**

While conducting this research project, the primary way in which data is collected is through the use of a snipping tool. This allows the outputs of the experiment to be quickly gathered while the outputs are continuously evolving. Data is collected on numerous components of this research. The first is through the setup and configuration stages. In this part of the project, screenshots are taken to document the procedures used and the outcomes of those step, this would include: the configuration of the private network, identifying internal IP addresses through terminals, and the setup of the Security Onion machine.

The second is through the launching of the attacks at Device A and Device B from Device C. Each of these attacks are launched and timed for 45 seconds before they are forcefully ended. This creates a baseline in which all of the attacks can be compared by. If the attacks were to be launched for different amounts of time, the outputs, based on the number of packets transmitted and the impact on each machine, would vary significantly. Thus, each attack is conducted numerous times, on each machine, for a series of 45 seconds. The outputs of each of these attacks are then gathered. This would include the changes in the CPU processing power during the
length of the attack, in addition to the output of the packets being transmitted in a TCP/UDP dump.

Lastly, the number of packets transmitted in each of the attacks is written down. This is done after each of the attacks have been executed and the results have been documented. Having this information provides another baseline, which can be used to compare the impact that each of these attacks have on Device A and Device B. After the collection of this data, it needs to be analyzed to draw any conclusions.

**Data Analysis**

The analysis of the data is an important component of any experiment. It takes the data brought in and turns it into information and results. For this work, the analysis of data starts with looking at each device separately, Device A and Device B. The attacks for each device are documented and need to be analyzed separately to understand the impact before they can be compared against one another. Each attack launched at a device is looked at on its own to understand the number of packets that were transmitted, the types packets transmitted, and the impact on the device's CPU(s). Once this information is gathered, Device A and Device B can be compared. The same information can be cross examined in a comparative analysis to understand the similarities and differences discovered in the experiment.

**Summary**

The methodology chapter has covered several elements pertaining to the experiment conducted in this work. It explains the utilization of VirtualBox, and the configuration of the three virtual machines. It addresses the concepts behind establishing an internal network to contain the attacks. It also includes the setup steps
of the Security Onion, to illustrate the configurations. Lastly, it addresses the types of
data that are collected and the ways in which it is examined or analyzed.
IV: Data Presentation and Analysis

Introduction

This portion of the paper illustrates the data that is collected during the experiment. The data is presented in formal manner, after which it is analyzed to show the results of the experiment. Conclusions will be drawn, and the outcomes of this research will be highlighted.

Data Presentation

During the execution of this experiment, a significant amount of data was collected. In order to break up the data and make it easier to understand, the presentation portion is split into two sections: the baseline data presentation and the security onion data presentation.

Baseline Data Presentation

The baseline device in this project is referred to as Device A, which is configured with the Ubuntu operating system. Three DoS attacks are launched at the baseline device to show the impact that the attacks can have on a common, every day, device. The first attack launched is a UDP DoS attack, which is ejected by Device C. Once again, device C is configure with the Kali Linux operating system. Figure 4.1 (below) shows the attack that was executed in Device C’s terminal.

```
root@kali:~ # hping3 --flood -a 10.10.10.10 -2 -p 6234 10.10.0.10
HPING 10.10.0.10 (eth0 10.10.0.10): udp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

Figure 4.1. Device A UDP Attack
When the attack was launched at Device A, Device A’s terminal was open and was tracking the TCP/UDP traffic. This is done by executing the command shown, below, in Figure 4.2.

```
root@ubuntu-VirtualBox:~# tcpdump -XX -i enp0s3
```

**Figure 4.2. Capture TCP Dump**

The type of traffic that was transmitted by the attack is shown in Device A’s TCP/UDP dump, which can be seen below in Figure 4.3.

**Figure 4.3. Device A UDP TCP DUMP**

Figure 4.3 shows a small snippet of the numerous packets that were received by Device A during the attack. From the TCP/UDP dump, it is easy to conclude that the attack is a UDP dump, that is coming from the source IP address of 10.10.10.10 and is being received by destination IP address is 10.10.0.10.6234. The packets are continuously received without any return packets being sent, which created the DoS attack.
The UDP attack was conducted five different times to show any variations in the attack. Each time the packets collected by Device A were generally the same as the ones displayed above in Figure 4.3. However, during the attack the CPU and network activity were monitored to show the impact on the device. Figure 4.4 (below), shows the CPU and network activity before any attack was generated.

![CPU History](image1)

**Figure 4.4 Device A CPU & Network Baseline UDP Attack**

Figure 4.4 is used as a baseline to indicate the performance of the machine, while operating in a normal state. Five UDP attacks were then launched at Device A. After each of the attacks the CPU and network activity were collected. Figure 4.5 (below) illustrates the impact on Device A after the execution of the first attack.
The remaining UDP attacks can be view in Appendix A. After each of the UDP attacks were launched for 45 seconds, the impact on the attacks were documented. However, once each of the attacks had been executed and terminated, after the 45 second window, new information about the attack was revealed. This information indicated the number of packets that were transmitted and received during the attack. The number of packets were then documented and placed in a bar chart to show the differences between each attack. This information can be seen below in Figure 4.6.
The next attack that was launched at the baseline machine was a TCP SYN flood Dos attack. The command used in Device C’s terminal to execute the attack can be seen below in Figure 4.7.

```
root@kali:~# hping3 10.10.0.10 -q -n -d 120 -S -p 80 --flood --rand-source
```

HPING 10.10.0.10 (eth0 10.10.0.10): 5 set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown

The packets received by Device A during the attack were viewed by looking at the TCP/UDP dump of the machine. The TCP/UDP dump can be viewed below in Figure 4.8.
The packets shown in the TCP/UDP dump in Figure 4.8 are just a small portion of the numerous packets sent by Device C. From the dump it is evident that the sources destination isn’t set, as different IP addresses are shown. Furthermore, the destination IP address is 10.10.0.10 and is set to HTTP or port 80. Lastly, the packet being sent is a SYN packet, which is continuously received by Device A without any responses being sent out.

The TCP SYN Flood attack was conducted five different times to show the variations in the attack. Each time the packets collected by Device A were generally the same as the ones displayed above in Figure 4.8. However, during the attack the CPU
and network activity were also monitored to show the impact on the device. Figure 4.9 (below), shows the CPU and network activity before any attack was generated.

Figure 4.9. Device A CPU & Network Baseline TCP SYN Attack

Figure 4.9 is used as a baseline to indicate the performance of the machine, while operating in a normal state. Five TCP SYN flood attacks were then launched at Device A. After each of the attacks the CPU and network activity were collected. Figure 4.10 (below) illustrates the impact on Device A after the execution of the first attack. The rest of the TCP SYN flood attacks can be found in Appendix A.
Once each of the TCP SYN Flood attacks had been launched, the results were documented in the above graphs. Although this wasn’t the only criteria that was examined. After each attack had been executed and then terminated, information about the attack was displayed in Device C’s terminal. This information included: the number of packets that were transmitted and received during the attack. The number of packets were then documented and placed in a bar chart to show the differences between each attack. This information can be seen below in Figure 4.11.
The last attack launched at the baseline device, from Device C, is the ICMP attack. The format of the attack, as shown in the Kali Linux terminal can be viewed below in Figure 4.12.

```
root@kali:~# hping3 10.10.0.10 -q --icmp -C 3 -K 4 --faster
HPING 10.10.0.10 (eth0 10.10.0.10): icmp mode set, 28 headers + 0 data bytes
```

**Figure 4.12. Device A ICMP Attack**

Once again, the tcpdump command was used in Device A’s terminal to track the packets. After the attack was launched by Device C the packets displayed in the dump were collected, they are shown in Figure 4.13.
Figure 4.13 shows a portion of the packets that are sent by Device C during the attack. From the TCP/UDP dump it can be concluded that the source destination IP address is 10.10.0.13, which is the IP address of Device C. In addition, the destination IP addresses is 10.10.0.10 and the packet being transmitted is ICMP.

The ICMP attack was conducted five different times to show the variations in the attack. Each time the packets collected by Device A were generally the same as the ones displayed above in Figure 4.13. However, during the attack the CPU and network activity were also monitored to show the impact on the device. Figure 4.14 (below), shows the CPU and network activity before any attack was generated.
Figure 4.14 Device A CPU & Network Baseline ICMP Attack

Figure 4.14 is used as a baseline to indicate the performance of the machine, while operating in a normal state. Five TCP SYN flood attacks were then launched at Device A. After each of the attacks the CPU and network activity were collected. Figure 4.15 (below) illustrates the impact on Device A after the execution of the first attack. The four other attacks that were launched and documented can be view in Appendix A.
After each of the attacks were launched the data on the CPU and network were documented. However, once an attack was executed and then terminated, information was revealed in Device C's terminal, pertaining to the number of packets that were transmitted. These numbers were documented for each of the attacks and were then inputted into a bar chart to visually show the differences between the attacks. This bar chart is displayed in Figure 4.16, which can be found below. Nevertheless, this concludes the presentation of data for the baseline, Device A, machine.
The next part of this section is the data presentation for the Security Onion. It will be similar to the data presentation found in the baselines section, as the same attacks were launched at both devices. The Security Onion device in this project is referred to as Device B. The three DoS attacks that will be launched at Device B will once again be coming from Device C. Figure 4.17 (below) shows the first attack that was executed in Device C’s terminal.

```
root@kali:~ # hping3 --flood -a 10.10.10.10 -2 -p 6234 10.10.0.11
HPING 10.10.0.11 (eth0 10.10.0.11): udp mode set, 28 headers + 0 data bytes
hping in flood mode, no replicas will be shown
```

**Figure 4.17. Device B UDP Attack**
When the attack was launched at Device B, Device B’s terminal was used to track the TCP/UDP packet traffic. The results from the attack, as shown in Device B’s terminal are depicted below in Figure 4.18.

![UDP TCP Dump](image)

**Figure 4.18. Device B UDP TCP Dump**

The UDP TCP dump displayed in Figure 4.18 shows a small portion of the numerous packets launched at Device B, from Device C. From the TCP/UDP dump it is can be concluded that the sources destination is 10.10.10.10, which was assigned in the attack. The destination IP is the securityonion and the port is 6234. Moreover, it is a UDP packet that is sent over and over again to flood that system.

The UDP attack was launched five different times to illuminate the variations that can exist between the attacks. Each time the packets were collected in the TCP/UDP dump on Device B, they were similar to the output shown above in Figure 4.18.

Although, this wasn’t the only component of interest. The CPU and network activity were monitored to show the impact of each attack on the device. Figure 4.19 (below), shows the CPU and network activity before any attack was launched at the device.
The baseline illustrated in Figure 4.19 is used to compare the results gathered after each attack is launched. Each of the five UDP attacks will be launched separately. After each attack the CPU and network activity are documented. The results of the first attack is shown below in Figure 4.20. The remaining four attacks can be found in Appendix B.

Figure 4.19. Device B CPU & Network Baseline UDP Attack
After each of the UDP attacks were launched for a 45 second interval of time, the impact on the device was documented. When the attack was terminated at the end of the 45 second mark, information on the attack was revealed in Device C’s terminal. The information of interest is the number of packets that are transmitted during the attack. The number of packets were then documented and placed in a bar chart. This will help reveal the differences between the attacks. This information is displayed below in Figure 4.21.
The second attack that was launched at Device B’s machine was a TCP SYN flood DoS attack. The command used in Device C’s terminal to execute the attack is shown below in Figure 4.22.

```
root@kali:~# hping3 10.10.0.11 -q -n -d 120 -S -p 80 --flood --rand-source
HPING 10.10.0.11 (eth0 10.10.0.11): 5 set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
```

**Figure 4.22. Device B TCP SYN Flood Attack**

The results from the attack can be viewed in Device B’s terminal, by executing the command to view the TCP/UDP dump. This was previously addressed while looking at Device A’s results, with Figure 4.2. The packets captured during the attack are displayed below in Figure 4.23.
Figure 4.23. Device B TCP SYN TCP Dump

The packets depicted in the TCP/UDP dump in Figure 4.23 illustrate just a tiny portion of the many packets collected in the dump. However, from this TCP/UDP dump it is evident that the source’s IP address is continuously changing, as it is randomly generated. Thus, making it hard to identify where the packets are coming from. The destination IP address is 10.10.0.10 and is operating on a http port, or port 80. In addition, the packet is a SYN packet which is send over and over again to flood the system and great a DoS.
The TCP SYN Flood attack was conducted five different times to show the differences between the attacks. Although each attack has the same kind of TCP/UDP dump shown above in Figure 4.23. This isn’t the only data collected during these attacks. The CPU and network activity were also monitored to show how the attacks impact the device. Figure 4.24 (below) illustrates the baseline collected for the CPU and network history. Following Figure 4.24 is Figure 4.25, which illustrates the results documented during the first attack. The results collected on the remaining four attacks can be seen in Appendix B.

Figure 4.24. Device B CPU & Network Baseline TCP SYN Attack
Once each of the TCP SYN Flood attacks had been launched, the results were documented in the above graphs. However, this wasn’t the only data that was collected during the experiments. After each attack was launched and then terminated, information on the number of packets transmitted during the attack were collected. This information was then put into a bar chart to display the differences between each of the attacks. This bar chart is shown below as Figure 4.26. This concludes the data presentation for the five TCP SYN Flood attacks that are launched at Device B from Device C.

Figure 4.25. Device B TCP SYN Attack #1
The final attack launched at the Security Onion, or Device B, from Device C, is the ICMP attack. The command launched in the Kali Linux terminal to create this attack can be viewed in Figure 4.27.

```
root@kali:~# hping3 10.10.0.11 -q --icmp -C 3 -K 4 --faster
HPING 10.10.0.11 (eth0 10.10.0.11): icmp mode set, 28 headers + 0 data bytes
```

**Figure 4.27. Device B ICMP Attack**

As previously stated, Device B’s terminal will be used to start a TCP/UDP dump. In which case the packets transmitted in the ICMP attack will appear in the dump. The results of this dump can be viewed in Figure 4.28 (below). The TCP/UDP dump results displayed in the figure only show a small snippet of packets from dump.
Figure 4.28. Device B TCP ICMP Dump

From Figure 4.28 it is evident that the source IP address is 10.10.0.13, which is the actual IP address assigned to Device C. The destination IP address is then securityonion, which would translate to 10.10.0.11. Furthermore, the packet is an ICMP packets that is flooding the Device B’s system with an extreme number of packets.

The ICMP attack was launched five separate times at Device B, from Device C. Each time the packets were displayed in the TCP/UDP dump and resembled the output displayed above in Figure 4.28. During the attack another piece of data was collected to be examined. This is referring to the CPU and network activity before and during an attack. Below, Figure 4.29 is used as a baseline, to show the machines state at a normal processing mode. After the baseline graph is the results of the first attack, shown in Figure 4.30 (below). The remaining attacks can be found in Appendix B.
Figure 4.31. Device B CPU & Network Baseline ICMP Attack

Figure 4.32. Device B ICMP #1
The ICMP attacks have all been launched and the data on the CPU and networking activity have been documented in the above graphs. Moreover, once an attack is executed and then terminated after the 45 second window, more information is displayed in Device C’s terminal. The information of interest pertains to the number of packets that are transmitted during an attack. This information was recorded for all five of the attacks and were put into a bar chart to show the variations. These results are available below in Figure 4.33.

![ICMP DoS Attack](image)

**Figure 4.33. Device B ICMP Dos Attack**

This concludes the data presentation portion of the paper for both the baseline and Security Onion portion of the paper. The next step is to analyze this data and draw conclusions.
Data Analysis

The data presentation for the baseline machine, Device A, and the Security Onion machine, Device B, were displayed in the previous section as two separate pieces. In this section of the paper they will be compared against one another to show any similarities or differences that may arise. The three different attacks launched each of the machines will be examined, starting with the UDP DoS attack.

The UDP DoS attack was the first attack launched at both of the devices. To understand the impact of each attack on a machine, the baselines need to be discussed. For Device A the baseline CPU hovers around 40% fluctuating up into the 50% range, on occasion. Its network is fluctuating up and down between the 0 to 0.1Kib/s range. For Device B the baseline CPUs fluctuate between 0 and 40%, while the network runs between 0 and 0.2Kib/s. Just from comparing the two baselines, it becomes evident that Device A requires more CPU power, which can be contributed to the fact that it only has one CPU. Meanwhile Device B uses less CPU power between the two of them.

Furthermore, when looking at the five UDP attacks launched at Device A the CPU fluctuates from the 40% range up into the 60% range remaining there until the attack has stopped. The network activity varies depending on the attack. For the first attack the network spikes to 1.6Mib/s and then even further to around 1.8Mib/s where it maintains until the attack is done. For attack two and five it spikes to around 1.8Mib/s and maintains around that point. For attack three and four, they both have spots where they run around 3.0Mib/s and 1.6Mibs.
When looking at the five UDP attacks launched at Device B, both CPUs fluctuate significantly with CPU1 working harder than CPU2. CPU1 fluctuates between 40% to 80%, while sometimes hitting 100%. CPU2 maintains between 40% and 60% during an attack. Device B's network activity varies, depending on the attack. For attack one, two, three, and five the network maintains around 1.6Mib/s. In attack four the network starts out around 1.6Mib/s and then fluctuates and increases to around 3.0Mib/s. From the first attack, the following can be concluded: Device B is better at maintaining the network activity, primary staying around 1.6Mib/s for each of the attacks. However, Device A maintains a better CPU processing.

The next attack that was launched was the TCP SYN Flood DoS attack. The baselines were similar in all three cases, with the description above being applied to the last two cases as well. When looking at the five attacks on Device A, there were fluctuating results in both the CPUs and the network activities. For attack one and two the CPU maintained around 60% with random jumps up, nearing 100%. In attack two the CPU was around 60% with a jump up to 90%. For attack four and five the CPU maintained around 60%. When looking at the network, in attack one it jumped to 7.0Mib/s and then dropped and maintained around 4.8Mib/s. In attack two and four the network maintains around 4.8Mib/s. Lastly, in attack five and three the network maintained around 4.8Mib/s with a few jumps up around 7.0Mib/s and 8.0Mib/s.

The five attacks that were launched at Device B also had fluctuating results. The CPUs fluctuated primarily between 20% and 60%. In attack one, tree, and five the network jumped to 4.8Mib/s and maintained around there until the attack was finished. In attack two the network jumped to around 8.0Mib/s and then fell and maintained
around 5.0Mib/s. For the last attack, attack four, the network climbs to around 7.2Mib/s and then drops to around 5.0Mib/s. In the TCP SYN Flood attack Device B is more efficient at maintaining a lower CPU average compared to Device A. It is difficult to say which machine is better at maintaining the network activity as they both are jumping around and maintaining around the same levels.

The last attack that needs to be analyzed is the ICMP attack. When the five attacks were launched at Device A there were varying results. In attack one, three, and four the CPU operated in intervals of 60% and 100%. In attack two the CPU maintained around 60% and in attack five it was primarily around 60% with a random jump up into the 90% range. For the networks, attack one, three and four operated at intervals of 0.0Mib/s and 1.8Mib/s. In attack two and five the network maintained around 1.8Mib/s.

When looking at the impact that the five ICMP attacks had on Device B, it becomes evident that results fluctuate. In attack one and two the CPUs ranged between 20% and 100%. Meanwhile, in attack three, four, and five the CPUs operated between 40% and 100%. Moving on to the network, in attack one the network started at 0.0Mib/s and jumped to 1.6Mib/s with a few drops. In attack two the network activity showed intervals of 0.0Mib/s and 1.6Mib/s. Finally, in attack three, four, and five, the network activity was around 1.6Mib/s. Once again, in this case Device B is primarily better at maintaining its CPU levels. It is also better at limiting the network activity remaining at a lower level than Device A.

Furthermore, it should be noted that the fluctuations in the different attacks can be contributed to the number of packets transmitted to the device, which was shown in the bar charts provide in the previous section.
Summary

This section of the paper introduced the data presentation from the research. It broke it down into two categories: baseline data presentation and Security Onion data presentation. After providing the presentation data, it was analyzed in the data analysis portion of the chapter to draw conclusions out of the significant amount of data provided.
V: Discussion, Future Work, and Conclusion

Introduction

This section of the paper aims to highlight the key points of the research, including complications and discussions. It will then outline future work for the project and conclude with remaining thoughts.

Discussion

This project was created to evaluate the efficiency of the Security Onion through a comparative analysis. A baseline machine was created to serve as the other device in the comparative analysis. Three different DoS attacks were then projected at the machines to examine their ability handle the attacks. The TCP/UDP dumps, CPU and network activity were used as the evaluation criteria. After completing the research project several discussion points have been identified.

First, when looking at the results of the experiments Device B, or the Security Onion, is more efficient that the baseline machine. It is capable of maintaining a lower CPU for each of the CPUs, with it performing better in two of the three attacks. Moreover, it was also better at maintaining the network activity, once again performing better in two out of three attacks. The network activity can be contributed to the Firewall changes that are implemented in the setup of the Security Onion.

The second discussion point refers to the complications encountered during the experiment. The Security Onion became a difficult tool to work with. The system kept reverting changes on its own and removing the set IP addresses. One minute the internal network would be working and the next it wouldn’t connect. During the experiment the network was setup in the GUI. After going through the setup, the IP
would reverted back, it was then set again through terminal commands. It would show in the terminal that it was configured properly, however, when rebooting or refreshing the network the IP address would once again revert back to the original state.

The third discussion point is whether or not the Security Onion is a recommendable tool. With any system there are advantages and disadvantages. The Security Onion is equipped with a variety of tools that can be downloaded and used. However, a user requires a certain knowledge base. The users of the system have to be aware of different kinds of packet and should be able to read them. If the user is capable of reading packets, then this tool could be useful in the right environment. It allows a user to sniff their network and categorized the types of packets. Users can then setup an alert system to allow the system to run somewhat on its own. It still requires continuous monitoring but introduces a level of efficiency.

**Future Work**

Future work for this project would start with resolving the network complications. After the system is running smoothly, the same DoS attacks or other variations could be ran to examine the system’s ability to identify the attacks in the different tools and send an alert to the user. This was anticipated to be a portion of this project; however, the tools were only able to detect one ping test and then the IP addresses kept reverting back to the original format. Thus, making it impossible to conduct this portion of the project. In an attempt to include this in the work, several different Security Onion Machines were created to try and get the connection back. Nevertheless, each of the machines failed to keep the configured network settings.
Conclusion

In conclusion, the Security Onion is an efficient tool that could help professionals monitor their system. It was more efficient at handing the different DoS attacks than the baseline Ubuntu machines was capable of doing. However, the Security Onion itself could use some updates and changes in its configurations. After some changes the tool could become a valuable tool for all companies, but a proper knowledge base is required to understand the configurations, the networks and the packets being transmitted and received.
References

https://www.silensec.com/downloads-


Appendix A: Device A’s Attack Results

Figure A.1. Device A UDP #2
Figure A.2. Device A UDP #3
Figure A.3. Device A UDP #4
Figure A.4. Device A UDP #5
Figure A.5. Device A TCP SYN #2
Figure A.6. Device A TCP SYN #3
Figure A.7. Device A TCP SYN #4
Figure A.8. Device A TCP SYN #5
Figure A.9. Device A ICMP #2
Figure A.10. Device A ICMP #3
Figure A.11. Device A ICMP #4
Figure A.12. Device A ICMP #5
Appendix B: Device B’s Attack Results

Figure B.1. Device B UDP #2
Figure B.2. Device B UDP #3

Figure B.3. Device B UDP #4
Figure B.4. Device B UDP #5

Figure B.5. Device B TCP SYN Attack #2
Figure B.6. Device B TCP SYN Attack #3

Figure B.7. Device B TCP SYN Attack #4
Figure B.8. Device B TCP SYN Attack #5

Figure B.9. Device B ICMP #2
Figure B.10. Device B ICMP #3

Figure B.11. Device B ICMP #4
Figure B.12. Device B ICMP #5