National College of Ireland

Controlling vulnerabilities in open-source libraries through different tools and techniques

MSc Internship
Cybersecurity

Huma Sulthana
Student ID: X20190247

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	HUMA SULTHANA
Student ID:	X20190247
Programme:	……M.Sc Cybersecurity…… **Year:** 2021-22
Module:	M.Sc Internship
Supervisor:	……Vikas Sahni……
Submission Due Date:	7th January,2022
Project Title:	Controlling vulnerabilities in open-source libraries through different tools and techniques
Word Count:	4907 …… **Page Count**……18……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Huma Sulthana ……
Date:	07th January, 2022 ……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Controlling vulnerabilities in open-source libraries through different tools and techniques

Huma Sulthana
X20190247

Abstract

This paper attempts to find the security vulnerabilities that occurs in an open-source libraries. These vulnerabilities are the weak points of open-source libraries which are used by attackers to reduce or/and breach the information security of the system. Hence, this is crucial to identify such vulnerable or weak points in the open-source libraries that harm a system for which this research is conducted. The main motivation behind this research is to identify and prevent security issues and provide a secure environment to students, researchers, and other people or institutes that used open-source libraries. The related work has been done on some of the tools such as Lodash, HTML unit, and handlebars are few of the popular open-source libraries. Among these tools to detect and prevent security issues, the Security Onion application is selected as it is free, open-source, and cloud-based. From the examinations, it is found that the application can detect issues, source and destination of it, set alerts, and check the health of the system. Though, it will be better if it blocked the IP addresses that most occurred as threats else the researcher could use paid tools for this. Though, it will increase the cost of this research.

1. Introduction

Open source libraires are huge source of free code which help researchers as well as academics. Some of the open-source tools such as Lodash and Vulas are full of security issues and attackers target these sites due to huge traffic. Hence, it is very necessary to detect and prevent searching and using these open-source libraries which are attractive to attackers.

Aim and Objectives

The aim of this research is to use a specific tool to detect and prevent security vulnerabilities from open source of information. Hence, the objectives are:

- To understand the concept of security vulnerabilities in open-source library
- To illustrate open-source vulnerabilities at present time
- To evaluate the contribution of open-source library
- To discussed on different vulnerability scanners for open-source library
- To identify an appropriate tool and use it for recognizing security vulnerabilities in open-source library

Research Questions

The questions to be answered as per the aims and objectives of this research are:

1. How different types of security issues are affecting in open-source libraries?
2. What are the current security issues existing in open-source libraries?
3. Which is the most appropriate tool to recognize vulnerability issues in open-source libraries and how it is used to examine and analysis the issues?

2. Related Work

In essence, this chapter of the report deals with the Literature Review where the researcher synthesized, evaluated, and recognized the relevant literature regarding the specific research topic. The researcher has evaluated the need of of open-source library and the contemporary state of thinking about the topic. Additionally, the researcher also recognized the research gap, specifically the unexplored way regarding this topic. Furthermore, articulation is also had been addressed the gap.

2.1 Concept of security vulnerabilities in Open-source Library

As commented by Decan *et al.* (2018), cybercriminals are simultaneously seeking to take benefits of an individual computer security vulnerabilities. Now, putting it into more generic terms, security vulnerabilities in open-source libraries are significant risks which the developers had to face the challenges by looking into the third-party and open-source code within their applications. According to Pashchenko *et al.* (2018), the open-source code often possesses with the vulnerabilities which can significantly impact on the organization's data and platform. Now, this security risks are often known as open-source vulnerabilities, the code can expose this open-source library software to the malicious cyber-attacks. As per Russell et al. (2018), increasing number of the software vulnerabilities are being discovered in each year whether they are being reported publicly or discovered internally in the proprietary code. Hidden flaws within the software can result in securities vulnerabilities and potentially allowing the attackers in compromising applications and systems. Thousands of those vulnerabilities are being reported publicly towards the common vulnerabilities and exposures to the database.

2.2 Open-Source Vulnerabilities in the Contemporary Time

According to Ponta *et al.* (2018), the open-source software has been increasing in rapid number, hence, the vulnerabilities on these open-source libraries are also being disclosed and discovered publicly. The vulnerabilities number are being disclosed for the OSS libraries have been steadily enhancing since the year 2009. However, the vulnerabilities of the OSS have been hitting up the headlines in the mainstream media several times since the past few years.

As per the report of White Source[1],they have able to find out the most up to date security vulnerabilities in the open-source library database[2]. This White Source has simultaneously collected data and information from the various sources like; NVD, open-source project trackers issue and security advisories. Following are the tools for identifying security vulnerabilities:

- ✓ **Lodash:** As per the report[3], prototype pollution security challenge has been found in the vulnerable Lodash version, while using up the ZipObjectDeep. This attack will further result in the disclosing of sensitive information, modification or addition of data as well as Denial of Services.

Figure 1: Function of Lodash

- ✓ **Html Unit:** As per the report, issues regarding the code execution got discovered within the vulnerable version of HtmlUnit[4]. When the HtmlUnit gets initializing up the engine of Rhino in distorted way, malicious javaScript code can get executed arbitrary application of JavaCode.

- ✓ **Handlebars:** As per the report of NPM Security Advisory, the issue regarding arbitrary code execution got found within the vulnerable version of this Handlebars[5]. This vulnerability leads in running of arbitrary code within the service processing of victim's browser and templates of Handlebars.

- ✓ **7 XStream:** In this open source, the issue regarding the remote code execution has been discovered within the vulnerable versions of the Xstream[6]. The attacker can further manipulate up the processed input stream as well as replacing or injecting up the objects which can arbitrary shell commands.

- ✓ **Vulas:** The demand for the Vulas started to increase in rapid number. According to Ponta et al. (2018), this tool underwent the major reimplementation process in order

[1] https://www.whitesourcesoftware.com/resources/blog/top-security-open-source-vulnerabilities-2020

[2] https://www.whitesourcesoftware.com/resources/blog/top-security-open-source-vulnerabilities-2020

[3] https://www.whitesourcesoftware.com/resources/blog/top-security-open-source-vulnerabilities-2020

[4] https://www.whitesourcesoftware.com/resources/blog/top-security-open-source-vulnerabilities-2020

[5] www.whitesourcesoftware.com/resources/blog/top-security-open-source-vulnerabilities-2020

[6] https://www.whitesourcesoftware.com/resources/blog/top-security-open-source-vulnerabilities-2020

to make for scalable and flexible to avoid further security breaching vulnerabilities in the open-source library.

As per Harer et al. (2018), thousand numbers of the security vulnerabilities are being discovered during the production of software every year. This occurs either getting reported publicly towards common vulnerabilities and exposure database or being discovered in the proprietary code of open-source library. According to as per Harer et al. (2018), these vulnerabilities are often occurring due to the errors being made through programmers and propagate in quick way for the relevance of open-source code and software re-use. As per Plate et al. (2015), software applications eventually integrate more open-source library source in order to get advantages from the code reuse. Now, the fostering of the open-source library within the software industry continued to thrive over the past few years and contemporary commercial goods being shipped to OSS libraries.

2.3 Contribution of Open-Source Library

As commented by Chen et al. (2020), open-source libraries are considered to be very critical towards the modern information infrastructure which eventually depends on software being written using up the decencies of open source. As stated by Maruping *et al.* (2019), in the world of computer science, the library eventually refers to the collection of reusable and precompiled files, scripts, functions and routines and other resources which can get referenced through the computer programmers. According to Anzt, *et al.* (2021), open-source library is any kind of library having open-source license which denotes up the software which is free for the modification, reuse and publish without the permission. These libraries are very useful for the programmers as they can get access to the pre-written, reusable, frequently used codes and dramatically reduces up the workload. Following are the contributions being made by open-source libraries:

- ✓ ***Community:*** It is to be noted that open source solutions are being driven through diverse, large and talented community with the generic goal for working together in getting quick improvements and issues regarding troubleshooting (Sittel *et al.* 2018).
- ✓ ***Cost:*** Open-source solutions and libraries are involved in decreasing up the overall cost in deploying up the solutions through the elimination of any licensing fees.
- ✓ ***Reliability:*** With the large and diverse group of humans, they review up the open-source libraries as well as software (Sittel *et al.* 2018). Open-source Output has been thoroughly get tested and tends to get high reliability and robustness.
- ✓ ***Security:*** Having diverse participants gets involved within the development of open-source solutions, this increases up the chances of resolving and discovering security vulnerabilities (Anzt, *et al.* 2021).
- ✓ ***Transparency:*** Full visibility within the code base renders up the transparency allowing users in developing an expectation as to what they will get to work.

According to Fakhlina and Saputra (2019), open-source libraries as the information services of management cannot get turn away from advancement of communication technology and information. Various softwares are being rendered in supporting need for the libraries in

running organization and rendering services to the users. According to Fakhlina and Saputra (2019), open-source software has been widely used by the libraries. This mainly occurs because the libraries are the non-profit organization, often have very low budget operation.
The example can be provided regarding the usage of open-source library. According to Rücker *et al.* (2017), the modern application of geophysics is being often desired for maximizing up the information on subsurface through the combination of various methods. According to Adams (2018), the intended eventually involved in studying up the contribution open-source libraries and this is also involved in creating up the social capital.

Dataset Security Vulnerabilities in the Open-Source Library System:
As per Gkortzis et al. (2018), the investigation of the various software features in relation towards the security vulnerabilities have been the constant topic of interest within the research community. In this paper, the researchers had able to create up the dataset which correlates with diverse software metrics being derived thousands of elements of open source along with their security bugs. According to Piantadosi et al. (2019), securities vulnerabilities are mainly the bugs which are involved to make errors in source code in the open-source libraries. Following can be exploited further.

- Taking control of the system
- Acquiring up the private data (mainly focusing on confidentiality)
- Taking the system down (availability)

As per Piantadosi et al. (2019), during the maintenance of open-source library software, vulnerabilities shall need to get address along with high priority as this can harm users in very severe ways as compared to the normal bugs. As per Pashchenko et al. (2018), vulnerable dependencies are known to be another problem in the open-source library software because these libraries are being developed and interconnected which not always get update within their dependencies.

Understanding Vulnerability Scanners for Open-Source Library Software
According to Sagar *et al.* (2018), the software application is considered as the program which is being used for running applications over the internet for performing certain tasks. Following are the top 5 threats that can occurs in the open-source library:

- ✓ **Injection:** In this attack, the securities are being compromised through placing up the commands of SQL or strings up within the code. According to Sagar *et al.* (2018), it is considered as one of the most generic tools in which SQL commands get manipulated within the input fields of web application.
- ✓ ***Session Management and Broken Authentication:*** According to Sagar *et al.* (2018), security gets compromised through the exploitation of leaks within the process system of authentication or any kind of flaws within the session management.
- ✓ ***Scripting on the cross site:*** The flaw of XSS can occur when the application eventually includes up the untrusted data within the new web page without having

proper escaping and validation or updating an existing web page with users being supplied data.

- ✓ ***Broken Access:*** Attack which can occurs when restrictions based on the user's activity has not been properly enforced which gives up the attacker an opportunity for exploiting these flaws and accomplishing up the access.
- ✓ ***Misconfiguration:*** Attacks which can occur in seeing up the flaws within the configuration of security application. The small misconfiguration can lead in putting the data of people.

Tools for Recognizing Security Vulnerabilities in the Open-Source Library

According to Jimenez *et al.* (2019), vulnerability prediction tools implement both the developer metrics and software metrics which have also been used for detecting the prediction. For supporting up the secure products of the software and vulnerability fixing, vulnerabilities are usually reported within the publicly available databases. As stated by Geek Flare (2021), application security is considered as the utmost significant thing in each organization today. Many of these applications in the contemporary time run inside the container as they are; cost-effective, very easy for scalability, take less storage capacity, faster deployable and user's resources. Following are the different tools used for scanning up the security vulnerabilities within the open-source library:

- ✓ ***Docker Bench Security Scanner:*** This is eventually a script having various automated tests in checking for best practices for deploying up the containers for production[7]. Now, for running this Docker security, an individual need to have Docker 1.13.0 or upgraded version.
- ✓ ***JFrog Xray:*** This is considered as the continuous open-source security as well as global artifact of analyzing tool[8]. This software involves in scanning up the dependencies and artifacts for the security vulnerabilities as well as license compliance challenges.
- ✓ ***Docker Scan:*** Still in the context of Beta, this Docker Scan involved in leveraging up the Synk Engine and capable of scanning up the local Dockerfile and images as well as its dependencies for finding known vulnerabilities[9].
- ✓ ***Dagda:*** This is considered as the open-source tool for doing up the static analysis of the known vulnerabilities like; malware, Trojans and other type of viruses. This involved in using up the antivirus name; ClamAV for detecting up such vulnerabilities.

According to Song *et al.* (2019), the biggest difference between the two kinds of tool eventually lies within the kind of security policies when they are in enforcement. Exploit migrations are being deployed the policy which aimed at preventing or detecting attacks. However, the sanitizers eventually aim in pinpointing up the precise locations regarding

[7] https://geekflare.com/container-security-scanners

[8] https://geekflare.com/container-security-scanners

[9] https://geekflare.com/container-security-scanners

buggy statements of programs. According to Carlson *et al.* (2019), the implementation of the libraries of third party for managing up the software community possess chance in exposing up the projects towards the vulnerabilities.

2.4 Gap in Literature

After evaluating up all the identified sources on research topic, still the gaps existing regarding the vulnerability issues, specifically in open-source libraries. Different studies had provided the vulnerabilities in the open-source software which the libraries mainly use for further working progression**.** Hence, there is big gap in identifying the most appropriate tools for controlling the vulnerabilities related to open-source libraries**.** Furthermore, still there are no proper research regarding the tools to eradicate all these security vulnerabilities.

3. Research Methodology

3.1. Research process

There are different types of vulnerabilities present in an open-source library and it might depend on the type of attacks as discussed in the previous chapter. For example, it can be caused by SQL injection, flaws in the session management, a leak in authentication, broken access, and misconfiguration. Similarly, there are multiple tools like docker bench, docker scan, dagada, and JFrog Xray which can help identify and deal against such issues. The researcher used these tools to identify and reduce security vulnerabilities in multiple open-source applications such as Lodash, 7 XStream, Vulas, Html unit, and Handlebars. After this, the research process includes analysis and comments on the most effective tools according to particular open-source libraries. However, the research will not be limited only to these certain tools and techniques. For example, JFrog Xray is a part of JFrog Artifactory which can connect with universal respiratory, but it is a static analysis tool. Though, it is a good example of containerized application (Hegedűs *et al.,* 2021). Thus, the researcher is responsible to explore more and use the best tools and techniques to overcome the limitation mentioned in the previous section of this study. For example, tools like SmartCheck, Mythril, and Oyente can analyze open-source applications based on their source codes (Parizi *et al.,* 2018).

3.2. Used Equipment

There are some fundamental tools as well as advanced tools that helps to complete the research. For example, MS Word is a basic tool that is used to write down the findings and entire research as well. The advanced equipment mainly includes software applications that are used to identify and reduce security vulnerabilities in the selected open-source libraries. For simplicity, and cost-saving the researcher used only free/open-source or trial versions of the detection tools. For example, Security Onion, VaraCrypt, Nmap, OSSEC, R-Statistics, PomWalker, JGit are some essential tools (Kula *et al.,* 2018). Squert and Sguil are two software options which required to be installed by the researcher. This can help in detecting attacks in real-time. This has a pf_ring that allowed it to handle a large amount of traffic. On

the other hand, Nmap scan will help to identify the open ports (especially HTTP port 80) vulnerable to attack (Kula *et al.*, 2018).

3.3. Techniques used on data collection and analysis

The source code from the open-source libraries is lexed as per token sequence and then filtered using a convolution neural network. The code is classified into vulnerable and non-vulnerable codes. A random forest classifier is used to improve the performance of the system as shown below.

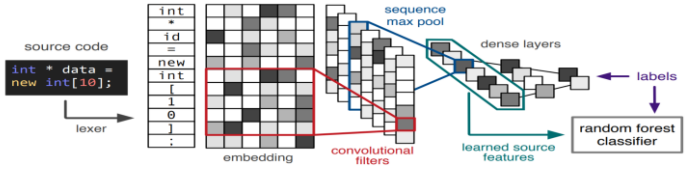

Figure 2: Illustration of source-code classification

(Source: Russell *et al.*, 2018)

Thus, machine learning is used to identify software vulnerabilities and the source code with vulnerabilities is mined from open repositories like GitHub.

3.4. Applications

The experimental setup and application of Security Onion are provided below.

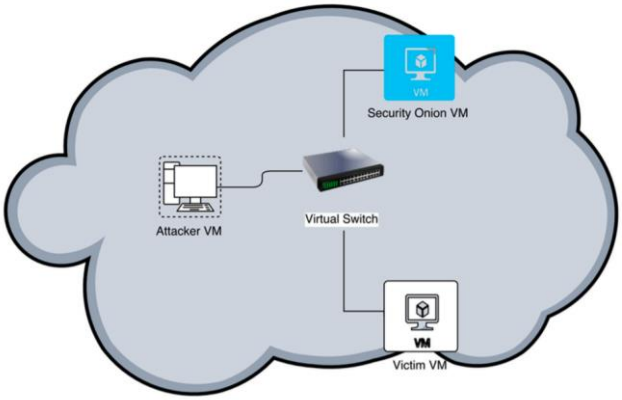

Figure 3: Setup and experimental application

(Source: Kula *et al.*, 2018)

The virtual switch is attached to the victim's machine, and it is also attached with security onion in the VM. When the attacker attacked the machine then the security onion starts its work.

3.5. Case studies/scenarios

There are multiple cases of using neural networks and free open-source tools for vulnerability identification and reduction. Kula *et al.*, (2018) used security onion in its case and Russell *et*

al., (2018) also used neural networks on source code. In this scenario, the researcher will do the same.

4. Design Specification

Here, the techniques to implement security onion software are followed as it is used for installing, configuration and test of IDS (Intrusion detection system). The design is consisting of three stacks known as ELK stack which are Logstash, Elasticsearch, and Kibana Search. At first, Logstash is responsible for collecting the logs which are then indexed by Elasticsearch so those can be searched easily. At last, the safety of security of the operation center or SOC is ensured by Kibana. Kibana is responsible for diving deep into suspected security issues by visualizing and analyzing the logs (Mikail and Pranggono, 2019). The fundamental framework of the software includes workstations, servers, and a firewall. Now, security onion can be used to monitor the traffic or data exfiltration as in this case. The traffic can be monitored from north/south or/and east/west to detect any lateral movements. Security onion does not collect logs from the personal workstations (like the personal computer) but it also gets logs from the server so it can hunt host logs, and network at the same time (Bourks, 2020). This fundamental framework is given below.

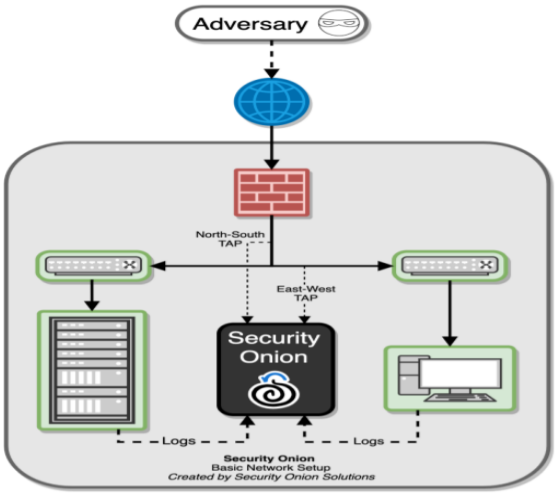

Figure 4: Security Onion Framework

(Source: Bourks, 2020)

The CPU architecture must be x86-64 else security onion will not be supported. This can be achieved through an AMD 64-bit processor or standard intel processor. The minimum specifications are 200 GB of storage, 4GB of RAM, and two different CPU cores. Though, 12GB RAM and 4 CPU cores might require with same storage specification to run the security onion other than so-import-pcap to import pcap. No new algorithm is required to be followed other than the standard algorithm to design Security Onion for this research. Hence, there is no need to discuss the functionality of the model. Standard NIDS and HIDS methods are followed with standard tools like Sguil, Squert, Kibana, and Capme (Gallagher *et al.*, 2018). However, there are certain limitations of the software and this framework, and it is very common. To become successful, it requires some commitments from the users. There

are many malicious activities which do not appear at the beginning. Hence, the research is required to monitor the alerts given in the console of Security Onion. On the other hand, monitoring the network activities together with corresponding system health and passion to learn are also necessary. However, the researcher was performing this task on manager so sudo service docker stop is done after the reboot as shown below.

```
sudo mv /nsm /nsm.old
sudo mkdir /nsm
# add your new file system to mount to /nsm in /etc/fstab
sudo mount -a
# make sure it's mounted correctly before continuing!
sudo mv /nsm.old/* /nsm/
sudo rm -rf /nsm.old
```

At last, the method is completed using the restart services with the help of the code given here:

```
sudo systemctl enable salt-minion
sudo reboot
```

5. Implementation

There are several outputs for several events that occurred in the workstations and logs of the entire framework. Moreover, it also depends on plugins for example, during tcp output plugin the host/port must not be down else it will cause blocking in the logstash pipeline. In that case, the researcher used a different logstash pipeline. The code which is used to change the configuration file for a different pipeline is given below.

Previous code: /opt/so/saltstack/default/pillar/logstash/manager.sls
New code: /opt/so/saltstack/ local/pillar/logstash/manager.sls
On the other hand, codes are written to transport a forward even to an external destination that has minimum modifications. One of the examples of this is given below which is used to configure when Zeek events are forwarded from dns dataset.

```
filter {
  if [module] =~ "zeek" and [dataset] =~ "dns" {
    clone {
        id => "clone_zeek_dns_events"
        clones => ["zeek-dns-clone"]
        add_tag => [ "clone" ]
    }
  }
}
output {
  if "clone" in [tags] {
    tcp {
      id => "cloned_events_out"
      host => "192.168.x.x"
      port => 1001
      codec => "json_lines"
    }
  }
}
```

Figure 5: Forwarding Zeek events

(Source: Bourks, 2020)

In this research, the latest version (beta) of Security Onion for which, the researcher can use the tools which help to build a machine learning model that detects anomalies in the logs formed by the components of security onion. However, logscan is the only machine learning model available at present and the following code is written to use this model (Argyropoulos, 2017).

```
sudo so-learn enable <component>
```

The models used in logscan are:

- K5: Login failures are searched with a high ratio from a single machine (one IP) in five minutes
- K1: Number of logins attempts from one IP address in a window of 1-minute
- K60: Abnormal patterns are searched in the logins from different IP addresses with a window of one hour.

Analysis tools like Sguil, Squert, Kibana, and Capme are used in this project.
Besides this machine learning method, there are different methods to adding new disks and it is done in this research when the researcher has required extra space. There are three such methods to adding new disk or expanding the required based. However, the first method which is Logical Volume Management is required to install LVM at the time of installation of Security Onion in the beginning which is not done by the researcher. That is why; the researcher has used the second method which mounting to add extra space. Though, it is a hectic task still it helps to mount a drive directly to the nsm. At first, the researcher requires to stop all of the services as shown below.

Stop services:

```
sudo systemctl disable salt-minion
sudo reboot
```

Stop and reboot helps to prevent all of the services to start from the beginning which eventually saves time.
An alternative method of the second one is also present which is little easy and time saving as it links nsm with new logging location. Though, for this, the researcher requires certain services such as AppArmor which was not there. Hence, method 2 is followed without any modifications.

6. Evaluation

Security onion is itself helpful in analyzing the result and this is done using different case studies which include a graph, plots to provide better visualization and understanding of the results and main findings. The list of cases is shown in the outputs as presented below.

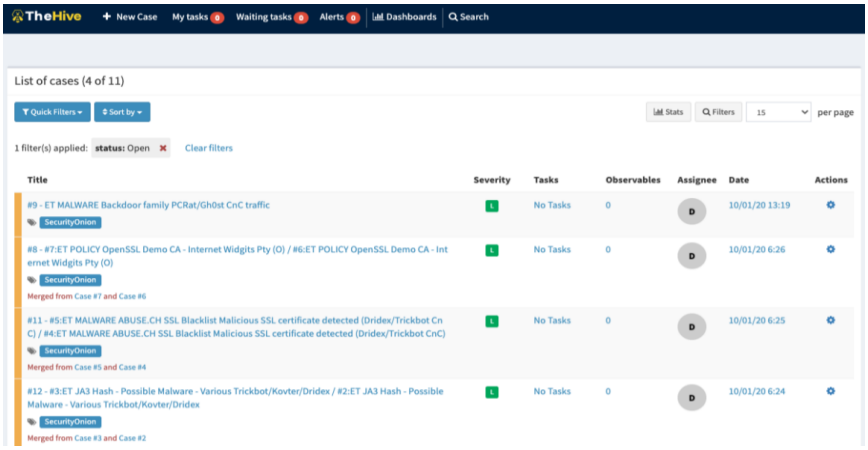

Figure 6: List of Cases

(Source: Bourks, 2020)

The cases are created using The Hive while working on the alerts, Kibana, or Hunt. This makes the analysis easy especially when interesting and suspicious events are sent to The Hive.

6.1 Experiment/Case Study 1 - Pivot analysis using Elastic in Kibana

Pivot analysis is used by the Elastic tool in Kibana to analyze types of different data. Both HIDS/NIDS and Zeek logs. The output is given below which showed that there are over 98 thousand logs analyzed by the Security Onion Console among which most of the logs are network logs and minimum level of host logs as well. Over 85 thousand Zeek logs are also identified.

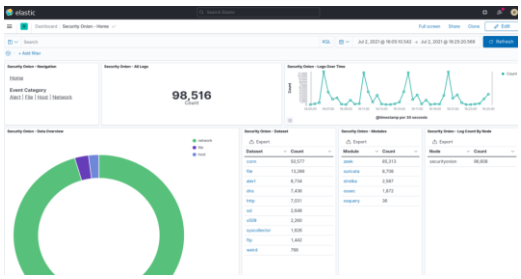

Figure 7: Identified logs

To decode the artefacts in Kibana and even in Alerts, and Hunt, the researcher used CyberChef. This tool also helped to analyze the data quickly. These tools make the security onion able for endpoint and network detection which was an impossible and complex task once.

6.2 Experiment/Case Study 2- Analyse Alerts using Security Onion Console or SOC

After logging into the security onion, SOC is the first thing to see which presents the list of alerts with count, rule.name, event module, and level of severity. For example, alert numbers 4 to 8, 10, and 11 to 13 in the list below have high severity.

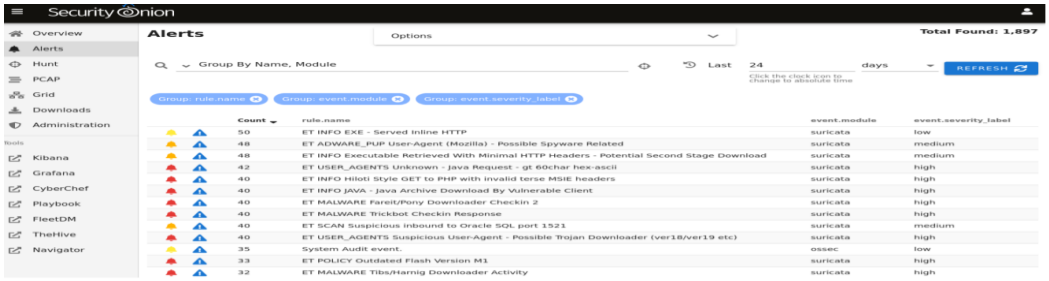

Figure 8: Alerts and severity

6.3 Experiment/Hunt Analysis

Hunt analysis helps to analyze the occurrence of NIDS/HIDS and even Zeek alerts. This provides all of the necessary details for example IP addresses of source and destination together with the ports. The bar graph showed IP address 10.42.42.253 has the highest occurrence between 2nd July and 6th July as given in the timeline while the researcher selected four days.

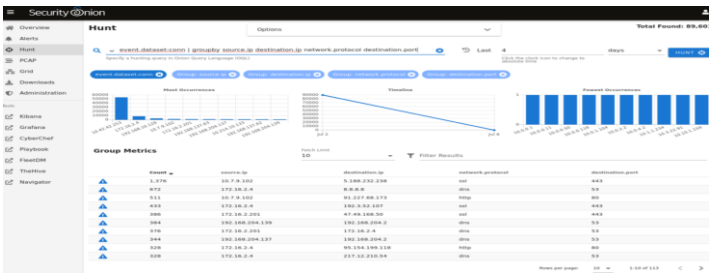

Figure 9: Result of hunting query

6.4 Experiment/Grafana Dashboard to analyse system health

There is a Grafana link in the SOC which is used to see system health. That means, it checked if any unauthenticated and unwanted files, viruses or malicious software, application, or any other such thing enter into the system from the open-source library or other sources.

Figure 10: Grafana Dashboard

This showed that the researcher checks the health of the system from the data of the last fifteen minutes.

6.5 Discussion

From the overall analysis, it is ensured that the Security Onion can be used for identifying alerts, then detect if the attack has occurred or not and at last it is also used to check the

health of the system. For example, different logs are identified in the first experiment, and alerts given by the SOC are analysed in the second experiment. The third experiment identifies the most occurred IP addresses. The researcher can block these addresses to prevent security vulnerabilities. However, the filter could be used to modify the outcomes with only open-source vulnerabilities. Moreover, there are different tricks that could enhance the data analysis in this research. One of the tricks that is used by the researcher is adding Jupyter notebook from https://jupyter-docker-stacks.readthedocs.io/en/latest/index.html
This is instantly included in the Elasticsearch which provides a network-based access to the Security Onion Firewall. Though, the researcher has ensured that the Jupyter has installed at least three python libraries which are pandas, elasticsearch and elasticsearch_dsl. For this the following commands are needed to be given in Jupyter docker container or Jupyter Host.

```
pip3 install elasticsearch
pip3 install elasticsearch_dsl
pip3 install pandas
```

This can be done after installing python prerequisites which is followed by executing the above commands. Though, the researcher was required to import the necessary libraries at first before using the above command. The commands to import necessary python libraries are:

```
from elasticsearch import Elasticsearch
from elasticsearch_dsl import Search
import pandas as pd
```

Installing Jupyter note provides the next level of hunting where the researcher can use different machine learning algorithm to hunt data. For example, results from SOC can be taken and converted into python dict. For example, the converted data provided below in the tabular format showed the process, timestamp, event, and even target file with version.

	process	winlog	tags	@timestamp	file	@version	event	user
0	{'pid': 3956, 'entity_id': 'EBE732EE-504F-61A5...	{'execution': {'ThreadID': 4400, 'ProcessID': ...	velociraptor	2021-11-30T04:22:45.357Z	{'target': 'C:\Program Files\WindowsApps\Micro...	1	{'code': '11', 'module': 'sysmon', 'category':...	NaN
1	{'pid': 3956, 'entity_id': 'EBE732EE-504F-61A5...	{'execution': {'ThreadID': 4400, 'ProcessID': ...	velociraptor	2021-11-30T04:22:45.357Z	{'target': 'C:\Program Files\WindowsApps\Micro...	1	{'code': '11', 'module': 'sysmon', 'category':...	NaN
2	{'pid': 3956, 'entity_id': 'EBE732EE-504F-61A5...	{'execution': {'ThreadID': 4400, 'ProcessID': ...	velociraptor	2021-11-30T04:22:45.357Z	{'target': 'C:\Program Files\WindowsApps\Micro...	1	{'code': '11', 'module': 'sysmon', 'category':...	NaN
3	{'pid': 3956, 'entity_id': 'EBE732EE-504F-61A5...	{'execution': {'ThreadID': 4400, 'ProcessID': ...	velociraptor	2021-11-30T04:22:45.357Z	{'target': 'C:\Program Files\WindowsApps\Micro...	1	{'code': '11', 'module': 'sysmon', 'category':...	NaN
4	{'pid': 3956, 'entity_id': 'EBE732EE-504F-61A5...	{'execution': {'ThreadID': 4400, 'ProcessID': ...	velociraptor	2021-11-30T04:22:45.357Z	{'target': 'C:\Program Files\WindowsApps\Micro...	1	{'code': '11', 'module': 'sysmon', 'category':...	NaN
...	...	...	...	...	...	...	...	...
3190	{'pid': 3224, 'entity_id': 'EBE732EE-6DD6-61A5...	{'execution': {'ThreadID': 4400, 'ProcessID': ...	velociraptor	2021-11-30T01:00:55.162Z	{'target': 'C:\Windows\SoftwareDistribution\Do...	1	{'code': '', 'module': 'sysmon', 'category': '...	NaN
3191	{'parent': {'entity_id': 'EBE732EE-511E-61A5-9...	{'execution': {'ThreadID': 4400, 'ProcessID': ...	velociraptor	2021-11-30T01:00:55.162Z	NaN	1	{'code': '', 'module': 'sysmon', 'category': '...	{'name': 'NT AUTHORITY\SYSTEM'}

7. Conclusion and Future Work

The main research question considered successful detection and prevention of security vulnerabilities from open-source libraries. To answer this question successfully the researcher first conducts a literature review where information on different open-source libraries and tools to detect and prevent security issues there. The researcher has found that Security Onion is one of the most effective tools for this research. One of the main benefits of this is that it is open-source and cloud based. The result also showed that SOC helps to visualize the threats in alerts by showing their severity even in graphs. Though, such tools focus on detecting the threats rather than preventing them. The researcher can show how to fix the issues for example, by blocking the IP addresses which occurred most of the time in Hunt analysis. From the above discussion it is also concluded that adding tricks can improve the data analysis in Security Onion. For example, the researcher has included Jupyter Notebook which enable to analyze the data with the help of python dict. Moreover, implementing machine learning algorithm also become easy with installing jupyter notebook. However, the researcher could save time if it installs some features at the first as he needs to install those latter which consume time. As an example, LVM was missing so expanding the space for the requirement of this research become little tricky so far. This could be better. Moreover, paid software will be a good idea to use for better results and automatic prevention. Though, the limited monetary resource of this research has limited it to use any paid software.

References

1. Anzt, H., Kuehn, E. and Flegar, G., 2021. Crediting pull requests to open source research software as an academic contribution. *Journal of Computational Science*, *49*, p.101278.
2. Maruping, L.M., Daniel, S.L. and Cataldo, M., 2019. Developer centrality and the impact of value congruence and incongruence on commitment and code contribution activity in open source software communities. *MIS Quarterly*, *43*(3), pp.951-976.
3. Pashchenko, I., Plate, H., Ponta, S.E., Sabetta, A. and Massacci, F., 2018, October. Vulnerable open source dependencies: Counting those that matter. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (pp. 1-10).
4. Sittel, P., Oppermann, J., Kumm, M., Koch, A. and Zipf, P., 2018, August. HatScheT: A Contribution to Agile HLS. In FSP Workshop 2018; Fifth International Workshop on FPGAs for Software Programmers (pp. 1-8). VDE.
5. Russell, R., Kim, L., Hamilton, L., Lazovich, T., Harer, J., Ozdemir, O., Ellingwood, P. and McConley, M., 2018, December. Automated vulnerability detection in source code using deep representation learning. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)* (pp. 757-762). IEEE.
6. Harer, J.A., Kim, L.Y., Russell, R.L., Ozdemir, O., Kosta, L.R., Rangamani, A., Hamilton, L.H., Centeno, G.I., Key, J.R., Ellingwood, P.M. and Antelman, E., 2018. Automated software vulnerability detection with machine learning. *arXiv preprint arXiv:1803.04497*.
7. Sagar, D., Kukreja, S., Brahma, J., Tyagi, S. and Jain, P., 2018. Studying open source vulnerability scanners for vulnerabilities in web applications. *IIOAB JOURNAL*, *9*(2), pp.43-49.
8. Jimenez, M., Rwemalika, R., Papadakis, M., Sarro, F., Le Traon, Y. and Harman, M., 2019, August. The importance of accounting for real-world labelling when predicting software vulnerabilities. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 695-705).
9. Fakhlina, R.J. and Saputra, D.F., 2019. Development and Contribution of Open-Source Software Communities for the Library Progress in Indonesia. *Record and Library Journal*, *5*(2), pp.150-159.

10. Rücker, C., Günther, T. and Wagner, F.M., 2017. pyGIMLi: An open-source library for modelling and inversion in geophysics. *Computers & Geosciences*, *109*, pp.106-123.
11. Adams, L.E., 2018. The contribution of library programmes at the Emfuleni Library and Information Services in creating social capital to reduce poverty. *Unpublished master's dissertation, University of South Africa, Pretoria*.
12. Song, D., Lettner, J., Rajasekaran, P., Na, Y., Volckaert, S., Larsen, P. and Franz, M., 2019, May. SoK: sanitizing for security. In *2019 IEEE Symposium on Security and Privacy (SP)* (pp. 1275-1295). IEEE.
13. Pashchenko, I., Plate, H., Ponta, S.E., Sabetta, A. and Massacci, F., 2018, October. Vulnerable open source dependencies: Counting those that matter. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (pp. 1-10).
14. Carlson, B., Leach, K., Marinov, D., Nagappan, M. and Prakash, A., 2019, May. Open source vulnerability notification. In *IFIP International Conference on Open Source Systems* (pp. 12-23). Springer, Cham.
15. Plate, H., Ponta, S.E. and Sabetta, A., 2015, September. Impact assessment for vulnerabilities in open-source software libraries. In *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (pp. 411-420). IEEE.
16. Decan, A., Mens, T. and Constantinou, E., 2018, May. On the impact of security vulnerabilities in the npm package dependency network. In *Proceedings of the 15th International Conference on Mining Software Repositories* (pp. 181-191).
17. Gkortzis, A., Mitropoulos, D. and Spinellis, D., 2018, May. VulinOSS: a dataset of security vulnerabilities in open-source systems. In *Proceedings of the 15th International Conference on Mining Software Repositories* (pp. 18-21).
18. Ponta, S.E., Plate, H. and Sabetta, A., 2018, September. Beyond metadata: Code-centric and usage-based analysis of known vulnerabilities in open-source software. In *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (pp. 449-460). IEEE.
19. Piantadosi, V., Scalabrino, S. and Oliveto, R., 2019, April. Fixing of security vulnerabilities in open source projects: A case study of apache http server and apache tomcat. In *2019 12th IEEE Conference on software testing, validation and verification (ICST)* (pp. 68-78). IEEE.

20. Chen, Y., Santosa, A.E., Sharma, A. and Lo, D., 2020, June. Automated identification of libraries from vulnerability data. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice* (pp. 90-99).
21. Parizi, R.M., Dehghantanha, A., Choo, K.K.R. and Singh, A., 2018. Empirical vulnerability analysis of automated smart contracts security testing on blockchains. *arXiv preprint arXiv:1809.02702*.
22. Hegedűs, C., Varga, P. and Frankó, A., 2021, May. A DevOps Approach for Cyber-Physical System-of-Systems Engineering through Arrowhead. In *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)* (pp. 902-907). IEEE.
23. Kula, R.G., German, D.M., Ouni, A., Ishio, T. and Inoue, K., 2018. Do developers update their library dependencies?. *Empirical Software Engineering*, *23*(1), pp.384-417.
24. Russell, R., Kim, L., Hamilton, L., Lazovich, T., Harer, J., Ozdemir, O., Ellingwood, P. and McConley, M., 2018, December. Automated vulnerability detection in source code using deep representation learning. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)* (pp. 757-762). IEEE.
25. Mikail, A. and Pranggono, B., 2019. Securing Infrastructure-as-a-Service Public Clouds Using Security Onion. *Applied System Innovation*, *2*(1), p.6.
26. Bourks, D., 2020. Security onion documentation. *Copyright© Security Onion Solutions, LLC*.
27. Gallagher, K., Patil, S., Dolan-Gavitt, B., McCoy, D. and Memon, N., 2018, October. Peeling the Onion's User Experience Layer: Examining Naturalistic Use of the Tor Browser. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1290-1305).
28. Argyropoulos, A., 2017. Intrusion Detection for the IaaS Cloud Model.

Websites

1. Geek Flare (2021) ***10 Container Security Scanners to find Vulnerabilities*** Available at: https://geekflare.com/container-security-scanners/ [accessed on 26th October 2021]
2. White Source (2021) ***Top 10 Open Source Vulnerabilities In 2020*** Available at: https://www.whitesourcesoftware.com/resources/blog/top-security-open-source-vulnerabilities-2020/ [accessed on 26th October 2021]