

**INVESTIGATING PARENT PROCESS
ID SPOOFING AND DEVISING AN
AUTOMATED INCIDENT RESPONSE**

Final Year Project

2019-2022



By

Asima Tahir 181254

Afeefa Ahmad 181209

Department of Cybersecurity

Supervisor

Dr. Naveed Anwar Bhatti

Faculty of Computing & Artificial Intelligence (FCAI)

Air University, Islamabad

Type (Nature of project)	<input type="checkbox"/> Development		<input checked="" type="checkbox"/> R&D	
Area of specialization	<input checked="" type="checkbox"/> WebApp		<input type="checkbox"/> Mobile App	
	<input type="checkbox"/> AI based		<input type="checkbox"/> Embedded System	
FYP ID	2102			
Project Group Members				
Sr.#	Reg. #	Student Name	Email ID	*Signature
(i)	181209	Afeefa Ahmad	181209@students.au.edu.pk	
(ii)	181254	Asima Tahir	181254@students.au.edu.pk	

*The candidates confirm that the work submitted is their own and appropriate credit has been given where reference has been made to the work of others

Plagiarism Certificate

This is to certify that, I **Afeefa Ahmad** S/D of **Sarfraz Ahmad**, group leader of FYP under registration no **181209** at Computer Sciences Department, Air University. I declare that my FYP report is checked by my supervisor.

Date: _____ Name of Group Leader: _____ Signature: _____

Name of Supervisor: Dr. Naveed Anwar Bhatti

Designation: Assistant Professor

Signature: _____

Co-Supervisor: (Not Allocated)

Designation: (N/A)

Signature: _____

HoD: Dr. Zunera Jalil

Signature: _____

Investigating Parent Process Id Spoofing and Devising an Automated Incident Response

Change Record

Author(s)	Version	Date	Notes	Supervisor's Signature

APPROVAL

PROJECT SUPERVISOR

Name: **Dr. Naveed Anwar Bhatti**

Date: _____

Signature: _____

PROJECT MANAGER

Date: _____

Signature: _____

CHAIR DEPARTMENT

Date: _____

Signature: _____

Dedication

All praise be to ALLAH the almighty, the sole creator of this world. He who has bestowed man with knowledge. Countless salutations on the noble personage of Hazrat Muhammad ﷺ, and peace and mercy upon his noble companions' eminent members of the family and true followers till the Day of Judgment. We owe our deep gratitude and feel great pleasure and privilege to pay our profound respects to our inspirational and encouraging supervisor Dr.Naveed Anwar Bhatti. His cooperation and kind interest, inspiring guidance, valuable suggestions, and discussions enabled us towards the completion of the project. His untired efforts and constant hard work will always serve us as a beacon in our life. Many thanks to other respected faculty members for continuous encouragement and motivation who motivated and encouraged us to complete this project.

Acknowledgments

Our deepest gratitude goes to Allah Almighty who has provided all that was needed to complete this project. Throughout this entire journey, He took care of everything that would have stopped us in our tracks and strengthened us even through our most difficult times.

We appreciate HOD Ma'am Zunera Jalil who has shown exemplary leadership as a Leader and Mentor. We appreciate the FCAI faculty members, for playing their part in ensuring that the vision speaks in our project. We are thankful to our supervisor Dr. Naveed Anwar Bhatti who had been the major pillar of the project, the Air University for providing its students the vital opportunity of exploring the domain of Cyber Security. Moreover, the faculty members who facilitated us in the project and motivated us throughout the whole journey. Lastly, we wanted to especially thank the Wazuh Community on the Slack and the Shuffle, TheHive, and open secure communities on the discord who helped us in the completion of the project

Executive Summary

*The malicious actors were always active in exploring the techniques to evade the endpoint security mechanisms in order to facilitate the execution of payloads remotely and fulfill other malicious intent. Therefore attackers rely on the attack of PPID spoofing attacker spoof the parent to execute any desired child process and thwart the detection. The project provides the automated detection and elimination of PPID spoofing attacks. The **Krabs ETW framework (C++ library that provides the details about kernel-level processes)** is utilized to integrate the utility of detection on the endpoint specifically for the Window Operating system. Then **Wazuh** is used for alert generation and active response which includes (the Kill child process, preventing PPID spoofing). Further, **Shuffle** is used for the creation of Playbook/workflow by integrating **Shuffle, Wazuh, TheHive, and Discord** into a single platform. **TheHive** automation includes alert creation, artifact addition, and case creation. **TheHive** provides the incident response for the attack of PPID spoofing detected on the endpoint along with it the final automated alert is sent to the **Discord** server.*

Table of Contents

Plagiarism Certificate	3
Dedication	6
Acknowledgments	7
Executive Summary	8
Table of Contents	9
List of Figures	12
List of Tables	12
List of Abbreviations	15
Chapter 1	16
Introduction & Background	16
1.1. Background	17
1.2. Motivations and Challenges	19
1.3. Goals and Objectives	19
1.4. Literature Review/Existing Solutions	19
1.5. Gap Analysis	23
1.6. Proposed Solution	23
1.7. Project Plan	24
1.7.1. Work Breakdown Structure	24
1.7.2. Gantt chart	24
Chapter 2	25
Software Requirement Specifications	25
2.1 Introduction	26
2.1.1 Purpose	26
2.1.2 Document Conventions	26
2.1.3 Intended Audience and Reading Suggestions	26
2.1.4 Product Scope	26
2.1 Overall Description	27
2.1.1 Product Perspective	27
2.1.2 User Classes and Characteristics	27
2.1.3 Operating Environment	27
2.1.4 Design and Implementation Constraints	27
2.1.5 Assumptions and Dependencies	28
2.2 External Interface Requirements	28

2.2.1	User Interfaces	28
2.2.2	Software Interfaces	30
2.2.2.1	Integration between Wazuh agent and Wazuh manager	30
2.3	System Features	36
2.3.1	Krabs ETW	36
2.3.2	Wazuh Manager	36
2.3.3	Wazuh Agent	36
2.3.4	Shuffle	37
2.3.5	TheHive	37
Chapter 3		39
Use Case Analysis		39
3.1	Use Case Model	40
Chapter 4		41
System Design		41
4.1.	Architecture Diagram	42
4.2.	Activity Diagram	43
4.3.	Sequence Diagram	44
4.4.	State Diagram	45
4.5.	Data Flow diagram	46
Chapter 5		47
Implementation		47
5.1.	Important Flow Control/Pseudo codes	48
5.2.	Components, Libraries, Web Service	57
5.3.	Deployment Environment	57
5.4.	Tools Version	57
Chapter 6		58
Business Plan		58
6.2	Market Analysis & Strategy	59
6.3	Competitive Analysis	59
6.4	Products/Services Description	59
6.5	SWOT Analysis	60
Chapter 7		61
Testing & Evaluation		61
Chapter 8		3

Conclusion & Future Enhancements	3
Appendices	5
Appendix A: Information / Promotional Material	6

List of Figures

1.1 Incident Response cycle	17
1.2 TheHive Components	19
1.3 Shuffle Core triggers	22
2.1 Software Versions	26
2.2 Krabs ETW execution	27
2.3 Wazuh Dashboard	28
2.4 Shuffle Interface	28
2.5 TheHive Interface	29
2.6 Discord server alert is shown	29
2.7 Generating Authentication Key	30
2.8 Ossec.conf File configurations	30
2.9 Local file configuration	30
2.10 Verifying from agent side	31
2.11 Decoder	31
2.12 Adding rule in Local rule	32
2.13 Creating the Dedicated organization	32
2.14 Creating the user in an organization	33
2.15 Adding authentication of TheHive in the shuffle	33
2.16 Create a Webhook and generate a unique URL	34
2.17 Server Setting of Discord	34
2.18 Copy and paste it on the Shuffle Discord app	35
2.19 TheHive Components	37
3.1 Usecase diagram	39
4.1 Architecture Diagram	41
4.2 Activity Diagram	42

4.3 Sequence Diagram.....	43
4.4 State Diagram.....	44
4.5 Data Flow Diagram	45
5.1 Tools Version	56
6.1 SWOT Analysis.....	59
7.1 Attack has been launched on the endpoint using the select my parent exe.....	61
7.2 Attack is been launched on the endpoint and Krabs ETW code is running in background monitoring	61
7.3 Attack has been launched on the endpoint using the select my parent exe.....	62
7.4: After detection, the log is been sent to the Wazuh Manager via Wazuh Agent	62
7.5 Active response rule is triggered	62
7.6 Shuffle alerts and execution argument section for the verification of smooth execution	63
7.7 Shuffle alerts and execution argument section for the verification of smooth execution on every node	63
7.8 Case creation on TheHive Case Management Platform	64
7.9 Complete and detailed PPID Spoofing attack alert.....	64
7.10 Alert and detailed process is posted in an automated manner on The Discord.....	65

List of Tables

1.1 Gantt Chart.....	24
2.1 Specifications	27
7.1 Testcase1	66
7.2 Testcase2.....	67
7.3 Testcase3.....	68
7.4 Testcase4.....	68
7.5 Testcase5.....	69
7.6 Testcase6.....	70
7.7 Testcase7.....	70

List of Abbreviations

SOAR	Security Orchestration Automation and incident response
IR	Incident Response
PPID	Parent Process ID spoofing
DLL	Dynamic Link Library
ETW	Event Tracing Window
TI	Threat Intelligence
CISO	Chief Information Security Officer

Chapter 1

Introduction & Background

Chapter 1: Introduction

The project aims to provide the automated incident response for the malicious parent-child processes through the open-source solutions and platforms as they would be used for the incorporation of automation and effective incident response.

1.1. Background

In past the attackers penetrate through the network undetected due to some parent processes that ultimately terminates after calling their child processes, the manipulation is been done with the parent-child relationship and the parent id is been spoofed.

But, the tables had turned with the rise of End-point detection and response, as well as threat hunting was a major setback for the attacker. The use of parent-child process analysis in particular has been a useful technique in detecting anomalous activity generated during nearly every phase of a kill chain. After the setback, the attacker manipulated the API of CreateProcessA, the API

CreateProcessA allows the users to create new processes. By default, inherited processes will create the new process that would act as their child process. This functionality also has a parameter “IpStartupInfo” [19] where the parent process definition can be done by the user which results in the biggest facilitation for the attackers to conduct the PPID. Parent PID Spoofing is an evasion technique used by malware and attackers that can utilize the CreateProcess Microsoft API and execute arbitrary code by injecting a shellcode, Dynamic-Link Library (DLL), or Portable Executable (PE) into the child process and thus evade some defenses like EDR and Anti-Virus.

To detect the PPID spoofing the best approach is ETW (Event Tracing for Windows), which provides real-time data [4] about the events happening on the endpoint or system. ETW process events can easily highlight anomalous parent spoofing and help discover the true origin of a process.

1.1.1 The Incident Response Cycle:

Incident: Incident in cyber security is defined as any event/alert caused by a cyber-threat or a successful cyber-attack that compromises confidentiality, integrity, or availability of data, assets, systems, or the whole organizational network [1, 2, and 3]. To better respond to a cyber security incident, an organization needs a set of procedures and guidelines. When responding to a cyber security incident, generally six steps are followed, which are outlined below and are also shown in figure 1.



Figure 1.1 Incident Response Cycle

1.1.2 Relationship between IR and SOAR:

Incident response is one of the most hectic, challenging, and time-consuming activities faced by every organization dealing with technology. With the limited amount of resources including staff, budget and time, organizations are unable to timely handle and respond to each incident exposing organizations to a high risk of cyber-attacks.[4][5] The primary goal of incident response is to achieve speed and efficiency in identifying, containing, and eradicating the attack. To accomplish that goal incident response process should be orchestrated and automated. Orchestration will help in bringing together/integrating disparate tools with different dashboards, and contexts onto a single platform because the incident response can never be performed through a single tool it requires multiple tools to work together in integration while Automation will reduce the load from security analyst by automating the processes/tasks. [11, 12] For that, security analysts introduced a security tool known as “SOAR (Security Orchestration, Automation and Response) [6, 7]. According to Gartner, “Security Orchestration, Automation and Response (SOAR) is defined as solutions that combine incident response, orchestration, and automation, and threat intelligence (TI) management capabilities in a single platform”. Incident response is the key feature of SOAR. It will not only help in boosting the organization's incident response (improving overall security posture) but also overcome the issue of resource limitation. [8, 9, 10].

1.2. Motivations and Challenges

1. Rise in the complexity and volume of cyber threats
2. Effective management of security threats in the complex business environments
3. Alert fatigue-As innumerable alerts, reports, and processes are generated, analyzing them is a tough grind.
4. Skill shortage gap - not enough skilled professionals to tackle the situation.

1.3. Goals and Objectives

1. Explore the parent and child processes relationship.
2. Understand the detailed investigation process of the Parent PID-Spoofing incident
3. Understanding the Wazuh and its working.
4. Defining playbook for malicious parent-child processes
5. Implementation of the playbook
6. Testing, QA, and Performance Optimization of Implemented Playbook
7. Ultimate goal is the detection of PPID and automated elimination of malicious child processes.

1.4. Literature Review/Existing Solutions

Multiple solutions are providing automated response and are available in commercial and open-source. Few of them are discussed briefly to deliver the idea.

1.4.1 Swimlane: Swimlane leverages incident response up to 80 to 90 percent by the mixture of people, processes, and technology by utilizing the concepts of orchestration and automation. . It allows the mixture of systems, applications, and APIs without extra charges. Currently, the Swimlane allows integration with over 1600 APIs and 200+ security tools. The new security tools are continuously added to the platform. It provides an open API framework that facilitates and allows the combination with any security tool. It's a flexible, dynamic, and interactive interface (Can set or customize the dashboard looking forward to the role [manager, security analyst, CISO], need, and requirement). A load of unattended alerts is reduced by using playbooks (manual, automatic, partially automated) and workflows that can be easily added and modified via simple drag and drop options. Customization is incredibly facilitated for any use case or any role likewise as customization within the blending via python development environment. [28]

1.4.2 TheHive: TheHive is a free web-based open-source automated incident responder tool. It acts as a central repository for case management. It's an interactive GUI, a dashboard displaying pie charts to induce a centralized view of overall security. Its three core functionalities:

Collaborate: TheHive connects people with multiple SOC, CERT, CSIRT, and other security analysts who can work together on a case, share information and keep everyone updated with their progress. [Can make the connection between similar cases]

Elaborate: It provides automation by allowing workflow creation where you'll create a playbook containing predefined procedural steps/ tasks to follow when a specific incident happened. For documentation purposes, it allows case template creation. Users can create/modify a template in step with their requirements by adding different headers like tags (based on the MITRE attack database), timeline, metrics, and far more.

Act / Analyze: TheHive integrates the functionality of MISP (Open source threat Intel platform) and Cortex (Analyze and answer incidents) to perform enrichment, contextualization, analysis, and response. TheHive can have bidirectional communication with MISP which suggests that TheHive may additionally get information (IOCs, TTP, etc.) from MISP also if new IOCs are detected TheHive may update the information on the Malware information sharing platform. [19]

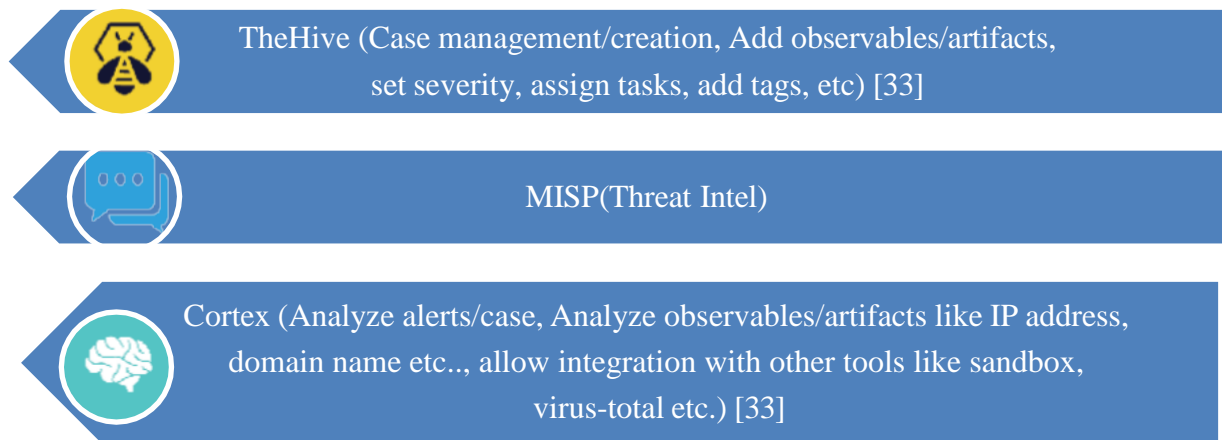


Figure 1.2: TheHive Components

1.4.3 Walk-Off: WALK-OFF could even be a flexible, easy-to-use, automation framework allowing users to integrate their capabilities and devices to chop through the repetitive, tedious tasks slowing them down. Includes the drag and drop mechanism for workflow creation and will be deployed on the software of Windows or Linux with customizable Python 2.7 and 3.4+. The platform also provides the pliability of sharing the workflows and applications. Provides the interactive GUI with different features that provide facilitation in viewing logs, custom reports, visualizations, and telemetry. Currently, supports the CSV format. [27]

1.4.4 Rapid7Insight connect: Incorporates the library of 300+ plugins for providing the siloed security and integration. Provide streamlined security by the import-connect and go workflow that doesn't require coding together with the choice of customization following the precise security automation required. Interactive dashboard to extend visibility and improve the coordination among vulnerability management and incident response processes by using ChatOps systems like Slack or Microsoft Teams. Insight Connect integrates with ITSM solutions like Service Now, JIRA, and far of others to ease collaboration with IT, development, and other teams. [26]

1.4.5 FortiSOAR: FortiSOAR provides the automation within the Visual Playbook Designer with 350+ security platform integrations and 3000+ actions for automated workflows and connectors. Easy and versatile options are provided for the deployment (VM, Hosted, cloud) available on FortiCloud, AWS, Azure, and as management extensions on FAZ/FMG. FortiSOAR's ML-powered recommendation engine is used for enabling false positive filtering and identification of real threats. In-built war-room for crisis management. Role-based dashboards and reporting empower SOC teams to live, track, and analyze investigations and SOC performance granularly with quantifiable metrics. Ingestion of structured and unstructured feeds is supported by the flexibility to import indicators from CSV/STIX files and export indicators in STIX format. [25]

1.4.6 Splunk (Phantom): Splunk provides automated playbooks and comes with 100+ pre-made playbooks. The Splunk SOAR provides integration of about 350+ third-party tools and supports over 2,800 different automatable actions. It's built-in comprehensive case management with incident case management to permit the establishment of defined workflows. Incorporated Built-in threat intelligence and insights from our SURGe cybersecurity research team. It has the five automation use-cases (alert enrichment, Phishing, investigation, Endpoint malware triage, and Command, Control Investigation and Containment, and Threat intelligence). [24]

1.4.7 Threat Connect (threatQ TDR Orchestrator): ThreatQ TDR Orchestrator is the industry's first solution to introduce a simplified, data-driven approach to SOAR and XDR that accelerates threat detection and response across disparate systems, leading to more efficient and effective security operations. Soar has the approach of data-driven automation and orchestration. For detection and response of the actual security, incident uses the data-driven playbooks instead of the process-driven playbooks that need the complexity which grows exponentially together with the rise within the number of playbooks.

ThreatQ TDR Orchestrator is supposed to simplify automation through the unique capabilities of Smart Collections, Data-Driven Playbooks, and also the Threat Library, each of which directly maps to the three stages within automation – Initiate, Run and Learn.

1.4.8 DF Labs (IncMan SOAR): The platform enables security managers and CISOs to oversee, manage and measure operational performance and cyber risk across every individual phase of the incident response workflow through role-based dashboards, customizable widgets, and nearly 150 KPIs and reports. DFLabs IncMan SOAR provides a centralized, automated, intelligence-driven command and control security automation and orchestration platform that spans the complete life cycle of incident detection, threat investigation, and orchestration of response. Security operations center (SOC) and computer security incident response teams (CSIRT) analysts, forensic investigators, and incident responders use IncMan to reply to, track, predict and visualize cyber security incidents. IncMan SOAR harnesses machine learning and automation capabilities to bolster human analysts. [23]

1.4.9 Palo Alto Networks (CortexXSOAR): Cortex XSOAR ingests aggregated alerts and indicators of compromise (IOCs) from detection sources like security information and event management (SIEM) solutions, network security tools, threat intelligence feeds, and mailboxes—before executing automatable, process-driven playbooks to counterpoint and answer these incidents. These playbooks aim to help in provide coordination among users, and security teams to create the centralized visibility of data. Thousands of automation scripts, UI-based filters, and transformers facilitate the manipulation of incident data and automate complex tasks during playbook creation. [21]

1.4.10 Exabeam: Soar platform with the pre-built, semi-built, or fully-automated playbooks, standardize incident response, and deliver repeatable and successful resolutions for familiar offenses. Providing the automation with turnkey playbooks. A drag-and-drop interface and flow charts to attach systems founded logic, and make powerful security actions simplify security playbook development. Exabeam has pre-built APIs that connect and integrate with an organization’s existing system, IT, and security tools—whether email servers, and Active directory, or a firewall—for rapid response. Exabeam includes a point-and-click interface that simplifies the method of making complex search queries. [22]

1.4.11 Shuffle Open source soar solution with almost 400+ software updates, and 300+ integrations. Incorporates the customizable drag and drop workflow. Shuffle allows for a user to own multiple

Organizations related to a User and the other way around. Organizations have logical barriers, making users ready to easily swap between them. Shuffle is further extended for MSSP's needs, with Sub-organizations to be controlled by a Parent-organization. Automation wouldn't be automation if you had to try and do manual work [20]. That's why Shuffle has implemented 4 core triggers:

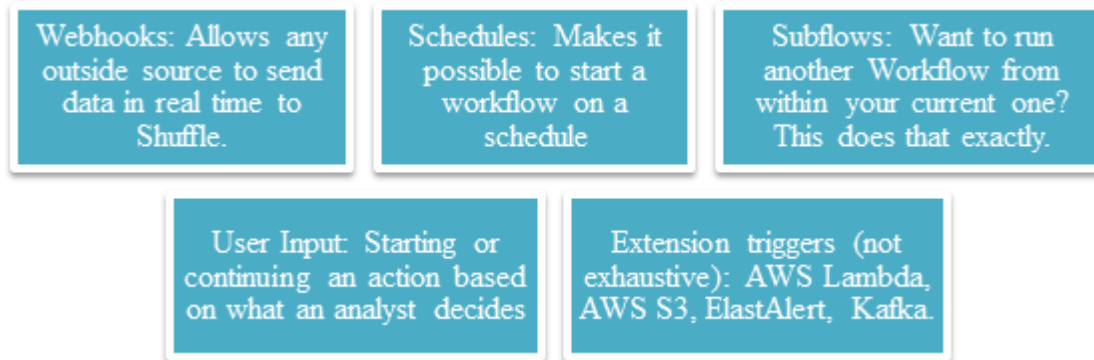


Figure 1.3: Shuffle Core Triggers

1.5. Gap Analysis

Most of the tools related to the scenario are commercial therefore differences can't be stated in the basic usage. The difference can be done based on the proposed solution for the attack of PPID spoofing. The section of the literature review also covered the details and features of other tools as per the claim of the tools in their documentation and websites.

1.6. Proposed Solution

The solution proposed was the detection and elimination of malicious child processes on the endpoint, as it would manage the resources and bandwidth in an effective manner that would help not only in load reduction but also in decreasing the timespan for detection of PPID. Due to non-integration and non-automation, the detection and incident response had become a tiresome and hectic task for the security analyst, as the current state of the art is using the methodology of sending the endpoint logs to the backend for detection and Incident response. The endpoint would use the detection script using the ETW libraries to detect the attack and the detection logs would be sent to the Wazuh server by the Wazuh agent deployed on the endpoint.

The **Krabs ETW framework** is utilized to integrate the utility of detection on the endpoint specifically for the Window Operating system. Then **Wazuh** is used for alert generation and active response which includes (the Kill

Child process, preventing PPID spoofing).

Further, **Shuffle** is used for the creation of Playbook/workflow by integrating Shuffle Wazuh, TheHive, **and Discord** into a single platform. **TheHive** automation includes alert creation, artifact addition, and case creation. TheHive provides the incident response for the attack of PPID spoofing detected on the endpoint along with it the final automated alert is sent to the Discord server.

1.7. Project Plan

1.7.1. Work Breakdown Structure

The project is been divided into 4 phases that include the detection of the endpoint. Further the alert generation and active response. Then, Playbook creation for automated elimination through the TheHive and sending the automated alert to the Discord server.

1.7.2. Gantt chart

Sr	Task	Start Date	End date
1	Detection Using ETW (Event)	December 1, 2021	December 10, 2021
2	Configuration and installation of Wazuh Manager, Wazuh Agent, TheHive and Shuffle	December 10, 2021	December 20, 2021
3	Execution of Active Response	December 20, 2021	December 30, 2021
4	Defining Playbook for the selected scenario using Shuffle	December 30, 2021	January 20, 2022
5	Creation of Webhook to connect Wazuh to Shuffle	January 20, 2022	January 30, 2022
6	Implementation of Playbook PPID Spoofing	January 30, 2022	April 10, 2022
8	Testing, QA, and Performance Optimization of Implemented Playbook Scenario	April 10, 2022	May 10, 2022
9	Documentation	May 10, 2022	May 15, 2022

Table 1.1: Gantt chart

Chapter 2

Software Requirement Specifications

Chapter 2: Software Requirement Specifications

2.1 Introduction

2.1.1 Purpose

The project would utilize the open-source case management system The Hive which would be integrated with the threat intelligence platform MISP, Cassandra the NoSQL database, and Cortex is an open-source utility that would serve as the analyzer and responder. Moreover, the endpoint integration with the Hive required the Wazuh agent as well as the Wazuh server to deliver the alerts to the Hive that is been integrated with Cortex, MISP, and Cassandra beforehand. Afterward, the playbook creation is done via Shuffle and an automated alert is been sent to Discord.

2.1.2 Document Conventions

The document is been divided into six sections, each section relatively explains the details of the particular section from the introduction of the project, till the business plan all of the system requirements, and use cases are explained. It would be suitable to start with the abstract to understand the summary of the project.

2.1.3 Intended Audience and Reading Suggestions

For proper understanding the reader should start from the background of the project that is been thoroughly explained in the first chapter of the report, the explanation was meant to develop the understanding as well as the limitations of the technologies been mentioned. Then, the reader should focus on the project architecture, the components, and their integration respectively. The architecture would clear the project picture and flow of the complete project.

2.1.4 Product Scope

The project is created with the Open source solution, and the malicious parent-child process creation is taken under the consideration through the use cases implemented in the solution. The investigation and detection mentioned in the objectives are achieved via the endpoint script written with the libraries Krabs ETW (Event Tracing Window), The Wazuh agent is transmitting the logs of PPID to the Wazuh manager would show the alerts that contribute to the transmission of logs to the Hive. Shuffle is used for the creation of a playbook to automate the incident response. The development of SOAR from the low level is not included in the scope of the project as open-source tools, and platforms were taken under use to achieve the objectives of the project.

2.1 Overall Description

2.1.1 Product Perspective

The project targets the already developed open source solution The Hive and integrates the feature of detection of PPID on the endpoint. The project would be a follow-on of The Hive. The process creation logs, parent id, name, and the path would be detected from the Endpoint, the Wazuh agent would act as a facilitator to transfer the logs to the Wazuh server through which ultimately logs would be sent to the Hive. In the project, the integrations between multiple platforms are done

2.1.2 User Classes and Characteristics

The end contribution of this project could be used by the security analyst, administrator, or end-point user to resist the PPID spoofing in an automated manner. The analyst would be able to analyze the logs files that had already detected the PPID spoofing, and actions against the attack would be done also in an automated manner to lessen the fatigue and surplus the effectiveness and efficiency. The most important class would be the analyst and security officers of an organization's SOC center as it enables them to work effectively.

2.1.3 Operating Environment

DEVICE	Laptop/PC
CPU	Intel(R)Core(TM)i5-3230M CPU @2.60GHz
RAM	32 GB
STORAGE	1 TB
OS	WINDOW 7 or Above

Table 2.1: Specifications

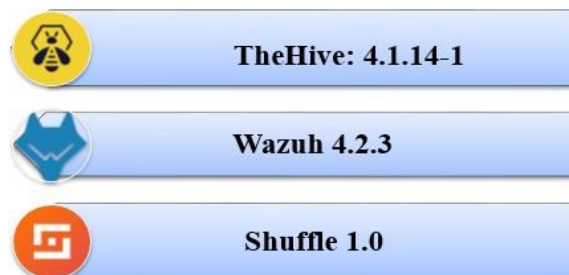


Figure 2.1: Software Versions

2.1.4 Design and Implementation Constraints

The hardware specs required should be a max of 16 to 32 GB for efficient working. Moreover, the requirement of a separate system is also necessary. The integration of the components is time-consuming.

2.1.5 Assumptions and Dependencies

The limitation in the achievement of the project can be possibly the lack of human resource as the project is quite extensible and require human effort as well the resources, the server and manager deployment and integration require a considerable amount of resources and effort. The well-known open-source SOAR solution is been taken under the consideration like (The Hive, and Shuffle). The current platform and case management system version 4.1.14-1 is open source, but the next versions are commercial. The detection code of Krabs ETW is been taken and modified from the referred resource. The project is dependent on the external platforms and their appropriate versions mentioned for the smooth functioning of the project.

2.2 External Interface Requirements

2.2.1 User Interfaces

The end-point would be targeted and the PPID spoofing attack would be launched, afterward, the detection through the Krabs ETW would be done and an agent would transfer the detection logs to the server. Afterward, the alert would be displayed on the dashboard. All of the interfaces are shown in the snippets attached below: This snippet shows the Krabs ETW that is used for detection purposes and should be executed with administrative privileges.

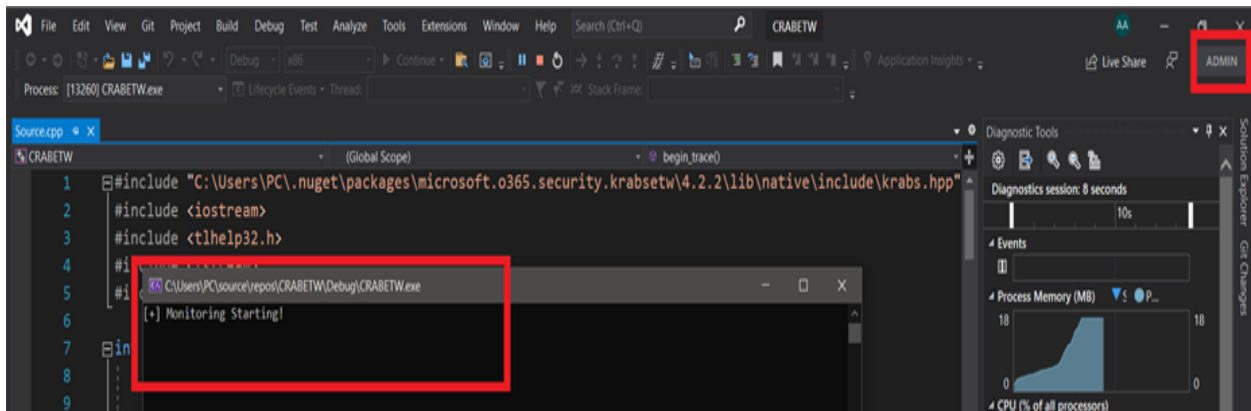


Figure 2.2: Krabs ETW execution

This snippet shows the Wazuh dashboard interface which shows the successful detection of the PPID spoofing attack

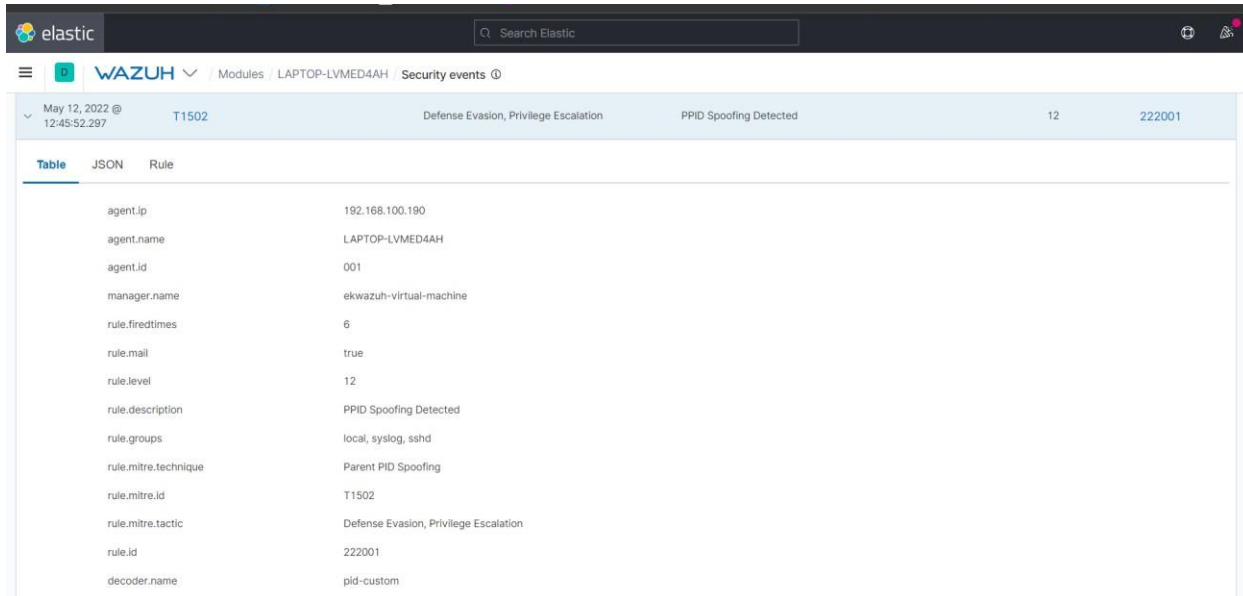
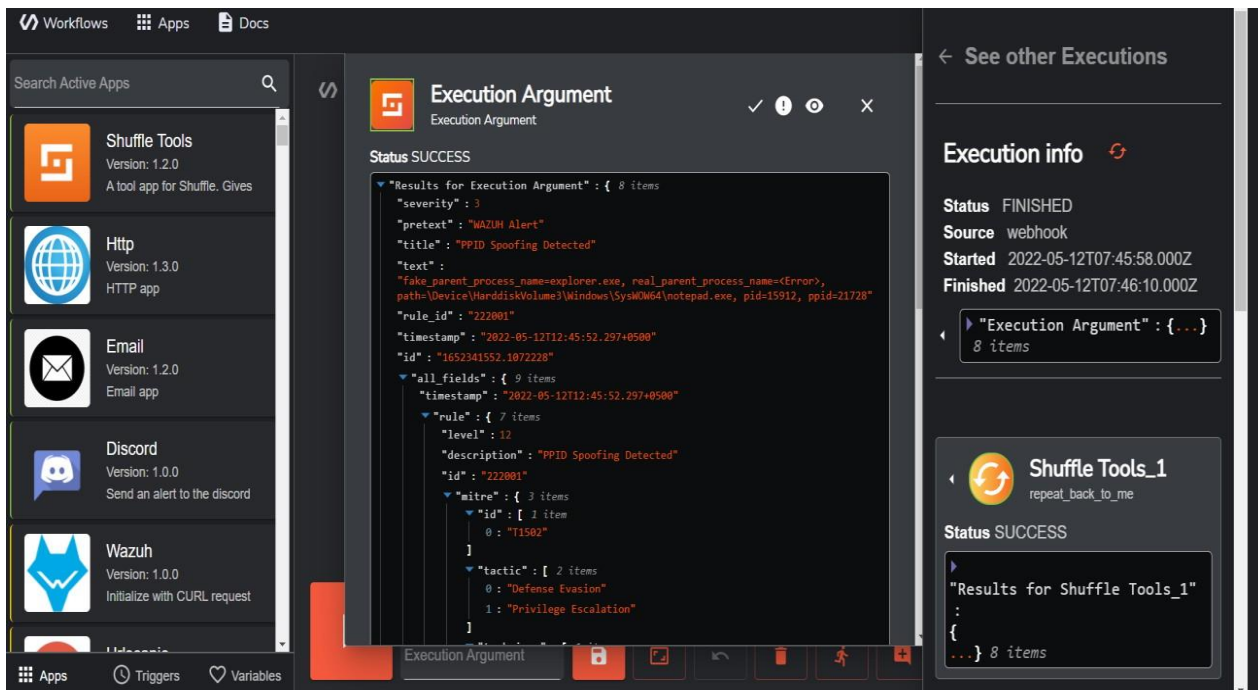


Table	JSON	Rule
agent.ip	192.168.100.190	
agent.name	LAPTOP-LVMED4AH	
agent.id	001	
manager.name	ekwazuh-virtual-machine	
rule.firedtimes	6	
rule.mail	true	
rule.level	12	
rule.description	PPID Spoofing Detected	
rule.groups	local, syslog, sshd	
rule.mitre.technique	Parent PID Spoofing	
rule.mitre.id	T1502	
rule.mitre.tactic	Defense Evasion, Privilege Escalation	
rule.id	222001	
decoder.name	pid-custom	

Figure 2.3: Wazuh Dashboard

The snippet below shows the Shuffle that is used for automated playbook and workflow creation.



The screenshot displays the Shuffle interface with a sidebar of tools (Shuffle Tools, Http, Email, Discord, Wazuh) and a main execution window. The window shows the 'Execution Argument' for a 'PPID Spoofing Detected' event, including details like severity, pretext, title, and timestamp. The results show a successful execution of 'Shuffle Tools_1' with 8 items returned.

Figure 2.4: Shuffle Interface

TheHive interface is shown for case creation

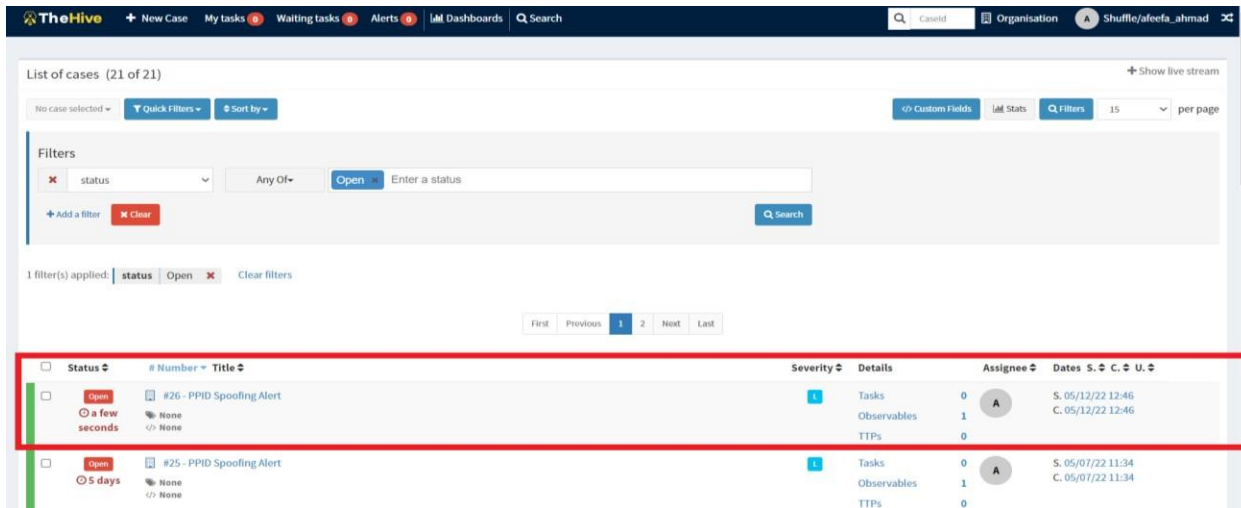


Figure 2.5: TheHive Interface

The Discord server interfaces are shown on which the alert is sent by TheHive.

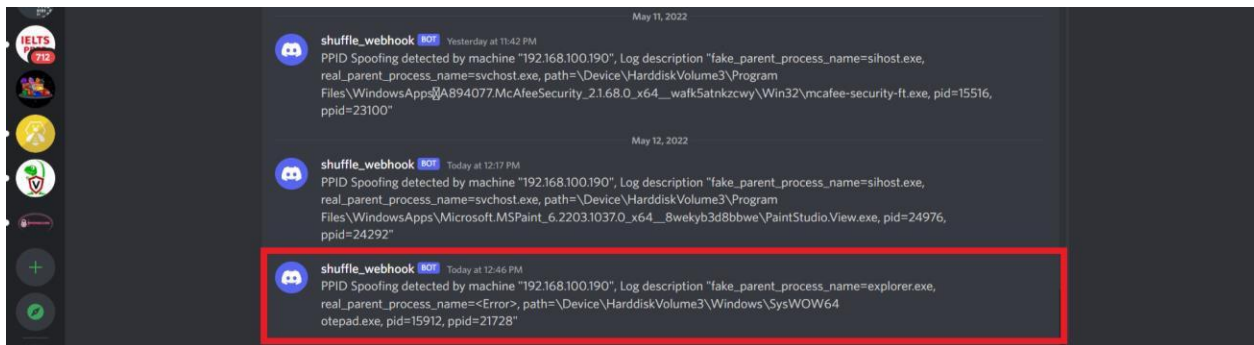


Figure 2.6: Discord server alert is shown

2.2.2 Software Interfaces

The project provides the automated incident response by integration of multiple platforms and tools as shown respectively. The steps done below are described and shows all the involved interfaces of the project.

2.2.2.1 Integration between Wazuh agent and Wazuh manager

- i. Generate Authentication key for Wazuh agent
- ii. Enter the key and Manager IP

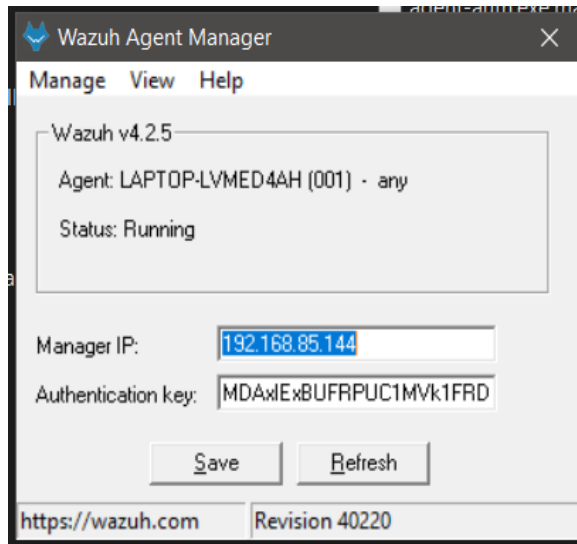


Figure 2.7: Generating the authentication key

Wazuh with Shuffle

Add the integration to Wazuh ossec.conf file

```
<integration>
  <name>custom-shuffle</name>
  <rule_id>222001</rule_id>
  <hook_url>https://192.168.85.138:3443/api/v1/hooks/webhook_a8e9a934-7077-43cd-9c40-9c6c310044eb</hook_url>
  <alert_format>json</alert_format>
</integration>
```

Figure 2.8: Ossec.conf File configurations

Wazuh Log file Monitoring

Add file location to the agent.conf file on the manager side

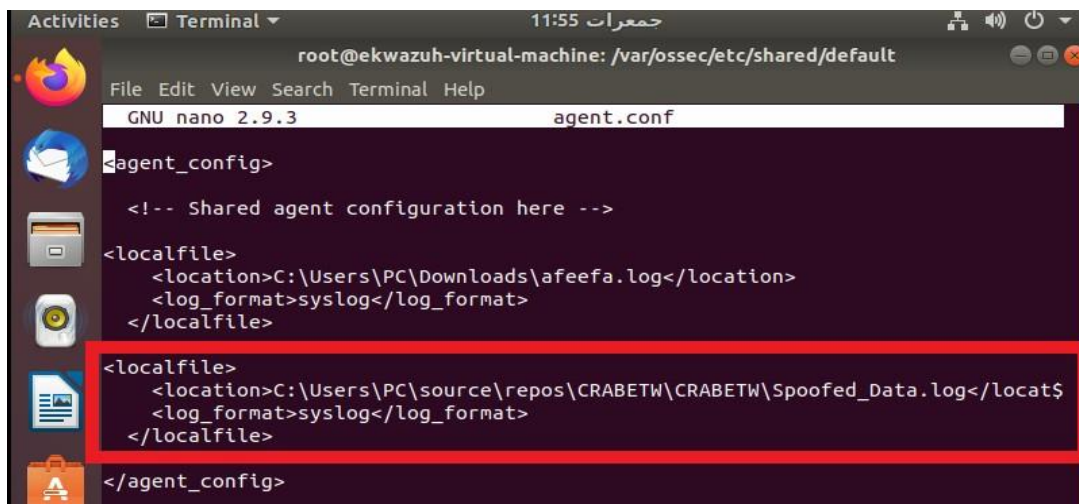


Figure 2.9: Local file Configuration

Make sure this change must appear automatically on the agent side as well

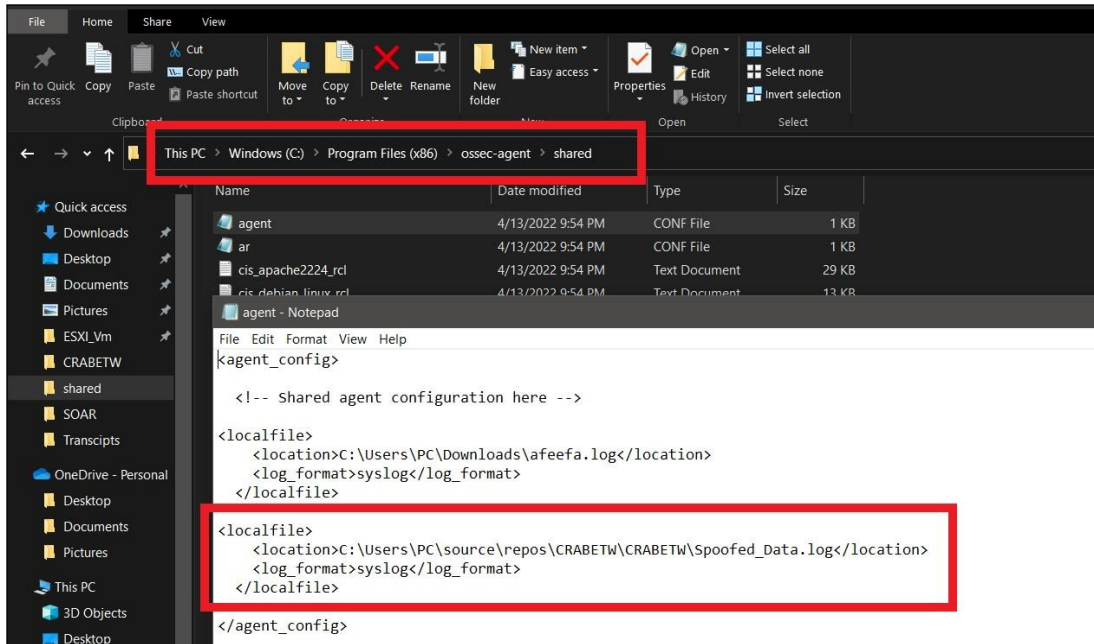


Figure 2.10: Verifying from the Agent side

Add decoder

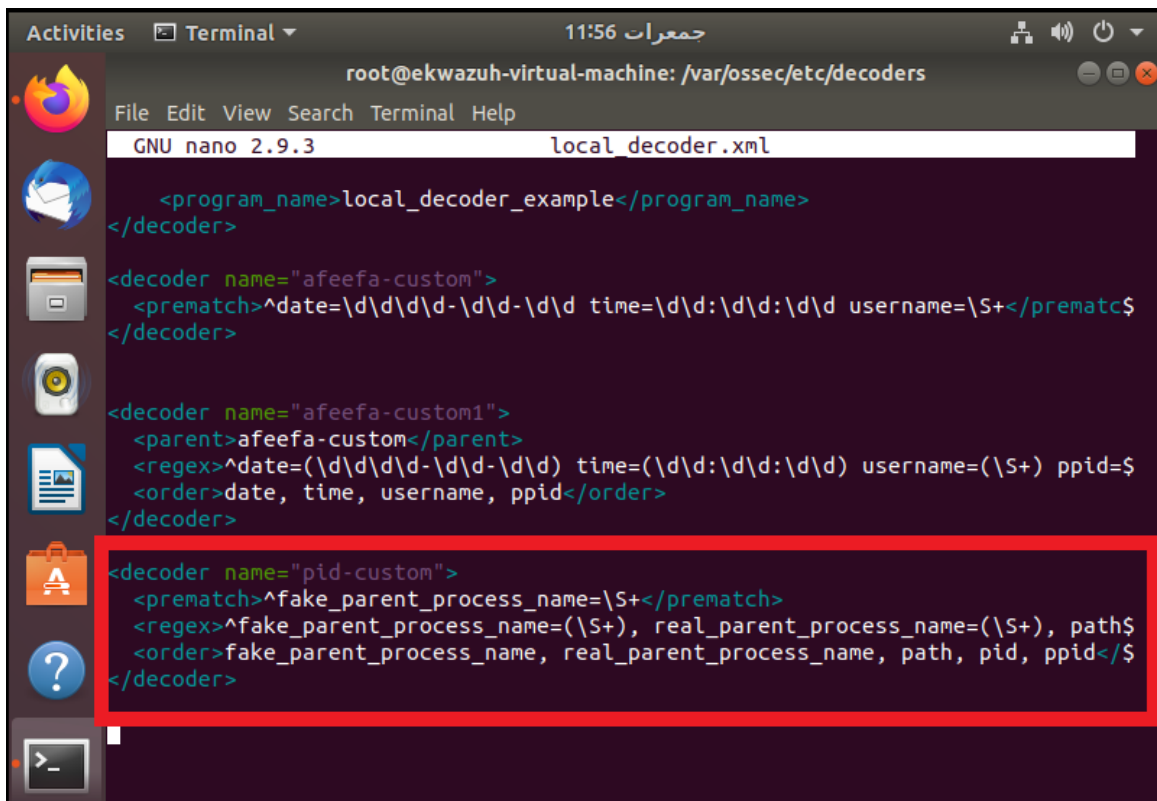


Figure 2.11: Decoder

Add a rule against that particular decoder

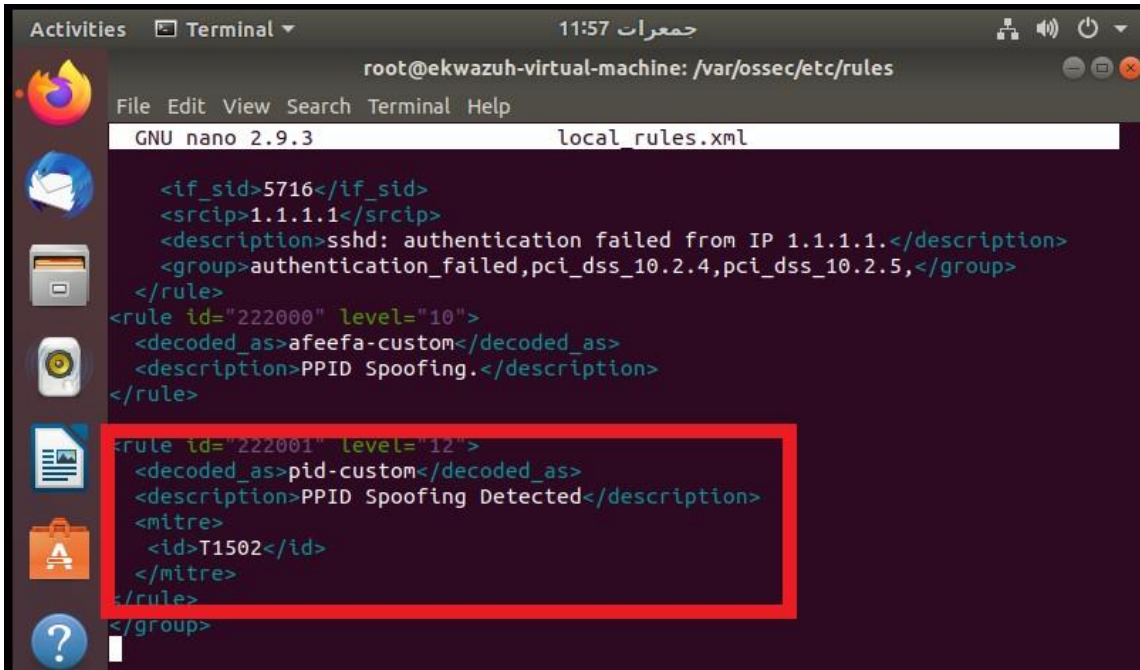


Figure 2.12: Adding Rule in local rule

(NOTE: Complete Rule and Decoder is given below in the section of implementation)

Shuffle with TheHive, Wazuh, and Discord

- TheHive

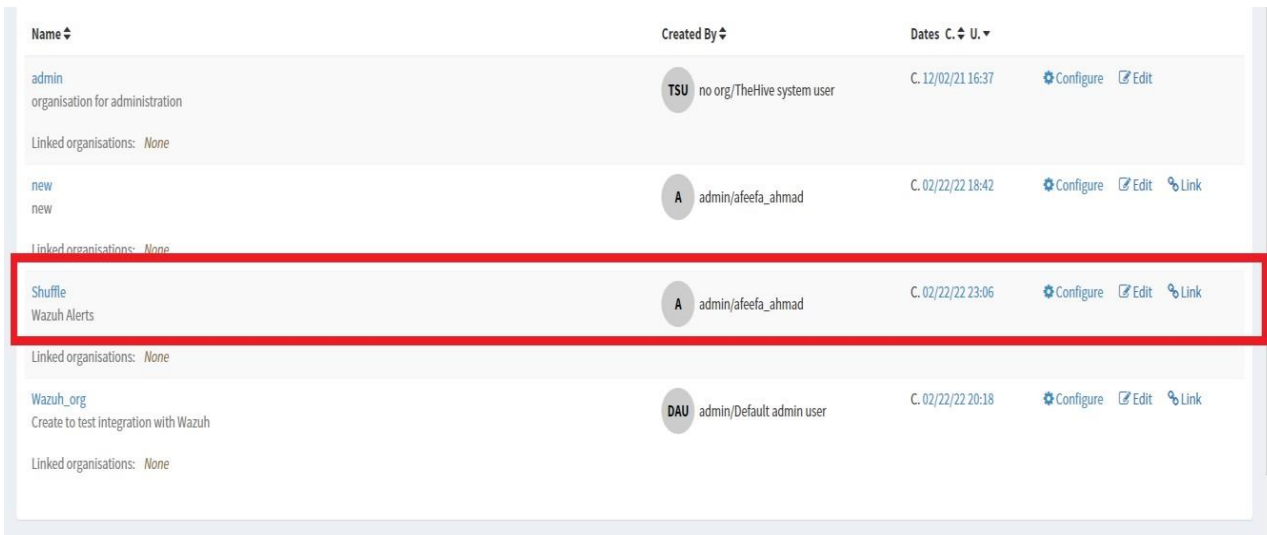


Figure 2.13: Creating a dedicated organization

Create a user within that organization

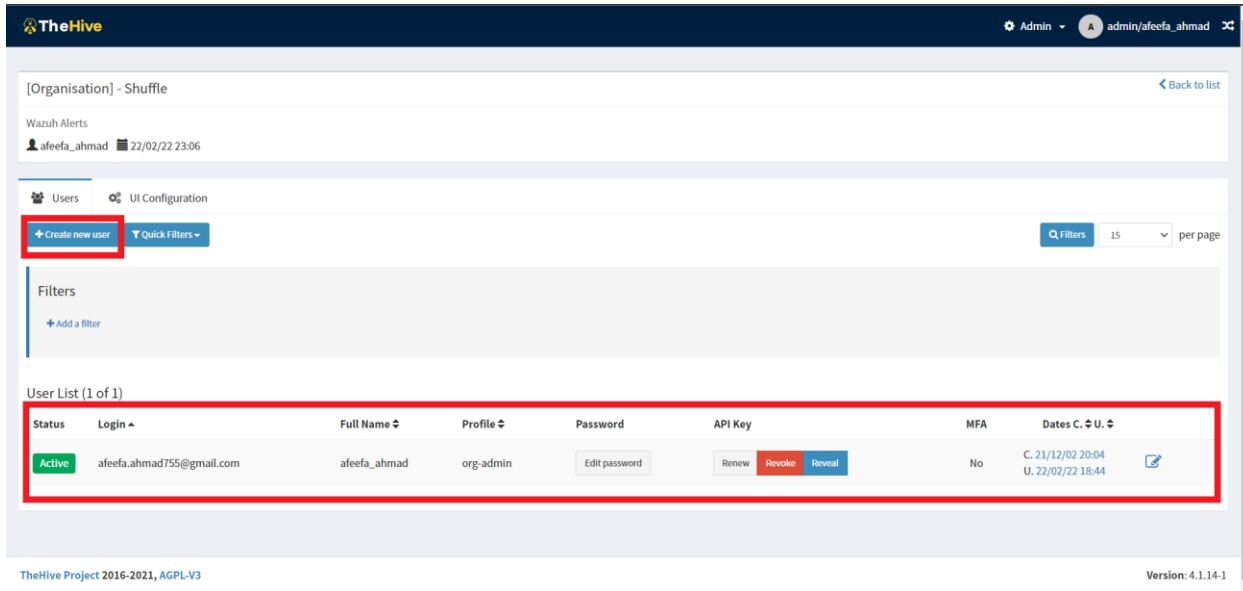


Figure 2.14: Creating a user within the organization

(You can either create an authentication or add it manually to every TheHive app)

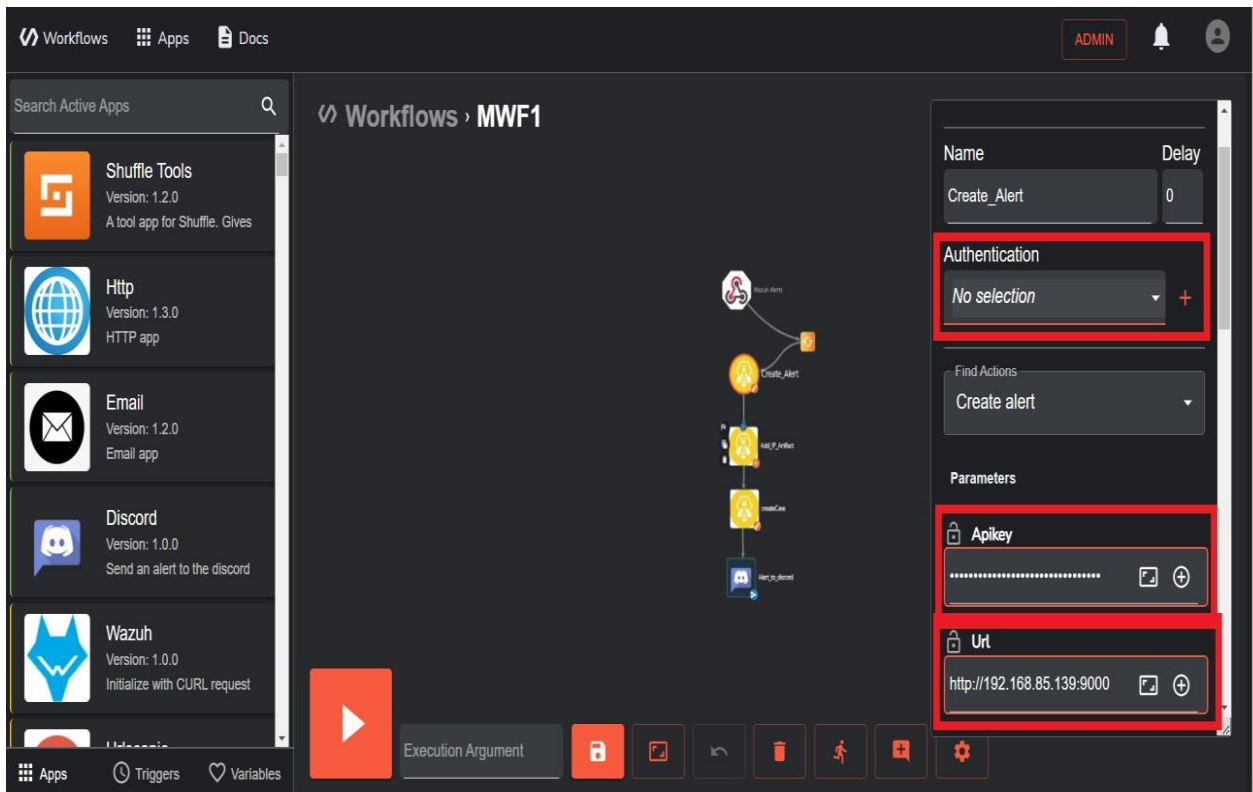


Figure 2.15: Adding Authentication to TheHive app in Shuffle

Wazuh

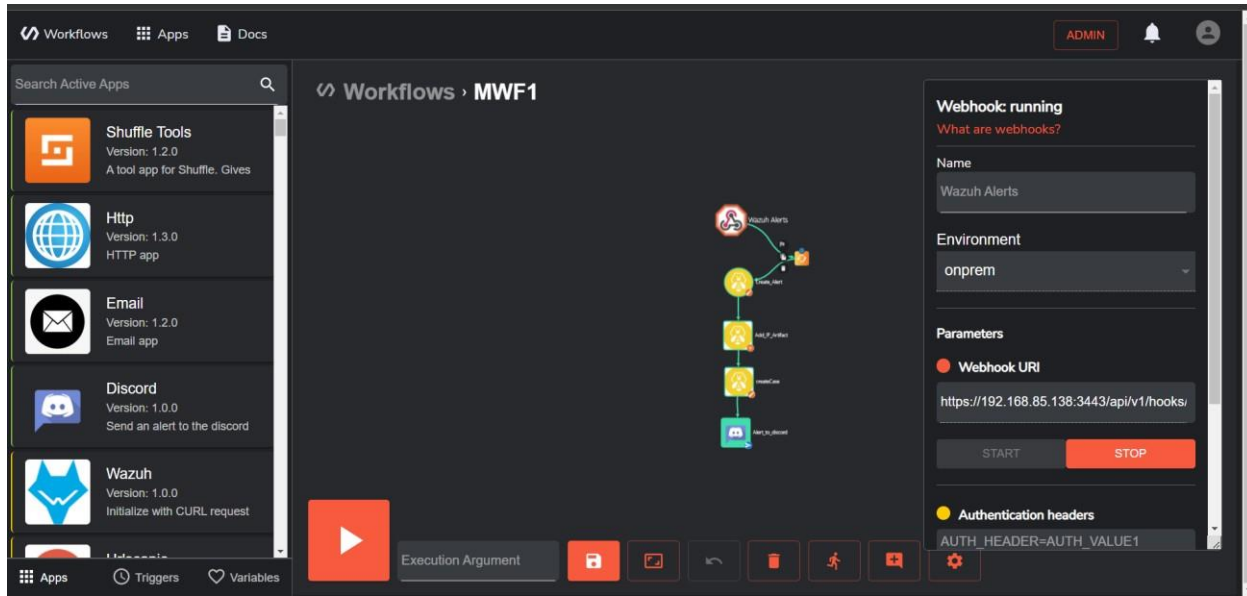


Figure 2.16: Create a Webhook and generate a unique URL

Afterward, the URL would be pasted in Wazuh ossec.conf file integrations area.

Discord

Create a unique Webhook URL

Go to Server settings > Integrations > Webhook

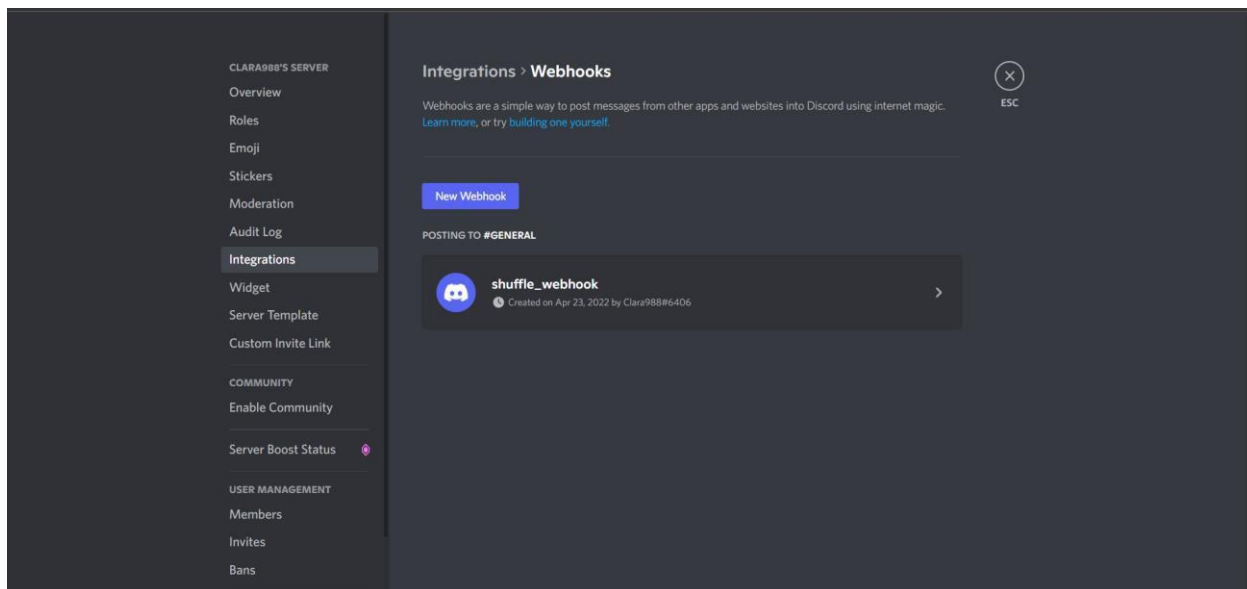


Figure 2.17: Server Setting of Discord

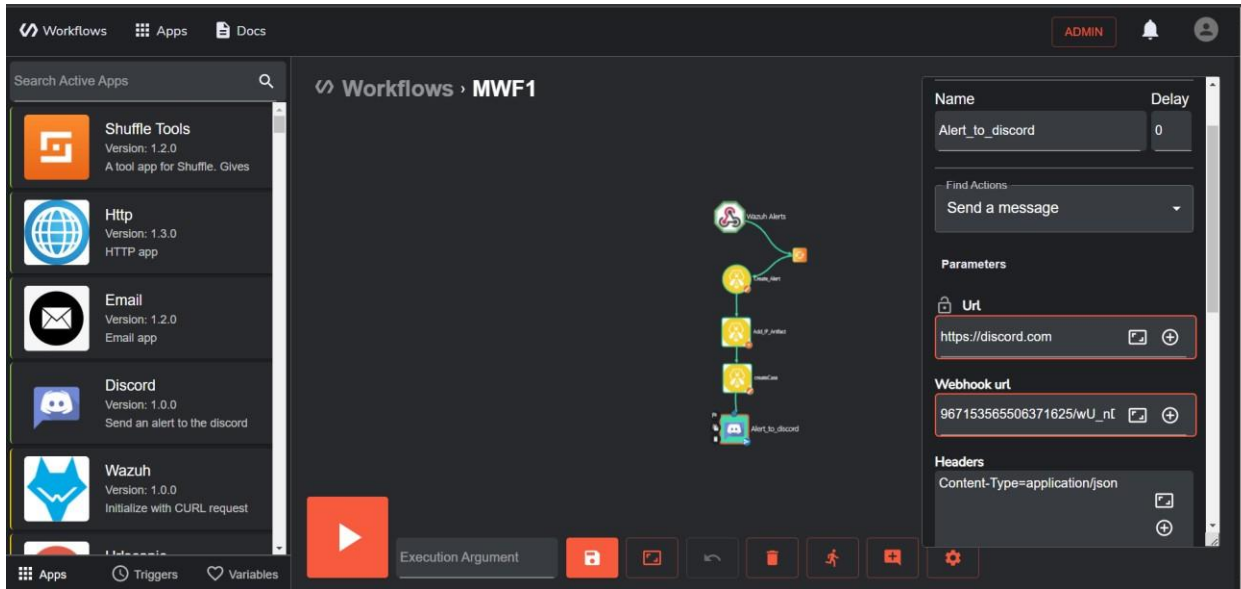


Figure 2.18: Copy and paste it on the Shuffle Discord app

2.3 System Features

2.3.1 Krabs ETW

KrabsETW provides a modern C++ wrapper and a .NET wrapper around the low-level ETW trace consumption functions. KrabsETW is a C++ library that simplifies interacting with ETW. It allows for any number of traces and providers to be enabled and for client code to register for event notifications from these traces. KrabsETW also provides the code to simplify the process of parsing the generic event data into strongly typed data types. [31] Microsoft.O365.Security.Native.ETW is a C++ CLI (.NET) wrapper around KrabsETW. It provides the same functionality as KrabsETW to .NET applications and is used in production by the Office 365 Security team. [31]

2.3.2 Wazuh Manager

The Wazuh manager is the system that analyzes the data received from all registered agents and triggers alerts when an event coincides with a rule, for example, an intrusion detected, a file modified, a configuration not following the policy, or a possible rootkit, among others. The manager also works as an agent on the local machine, which means that it has all the features that an agent has. In addition, the manager can forward the alerts that it triggers through Syslog, emails, or integrated external APIs. [29]

2.3.3 Wazuh Agent

The Wazuh agent is multi-platform and runs on the hosts that the user wants to monitor. It communicates with the Wazuh manager, sending data in near real-time through an encrypted and authenticated channel. The agent was

Developed considering the need to monitor a wide variety of different endpoints without impacting their performance. Therefore, it is supported on the most popular operating systems and only requires about 0.1 GB of RAM. The Wazuh agent provides key features to enhance the system's security. [30]

- Log collector
- Command execution
- File integrity monitoring (FIM)
- Security configuration assessment (SCA)
- System inventory
- Malware detection
- Active response
- Containers security monitoring
- Cloud security monitoring

2.3.4 Shuffle

Open source soar solution with almost 400+ software updates, and 300+ integrations. Incorporates the customizable drag and drop workflow. Shuffle allows for a user to own multiple Organizations related to a User and the other way around. Organizations have logical barriers, making users ready to easily swap between them. Shuffle is further extended for MSSP's needs, with Sub-organizations to be controlled by a Parent-organization. Automation wouldn't be automation if you had to try and do manual work. [20] That's why Shuffle has implemented 4 core triggers:

- **Webhook:** Allows any outside source to send data in real-time to Shuffle.
- **Schedules:** Makes it possible to start a workflow on a schedule.
- **Sub flows:** Want to run another workflow from within the current one.
- **User Input:** Starting or continuing an action based on what an analyst decides
- **Extension triggers (not exhaustive):** AWS Lambda, AWS S3, ElastAlert, Kafka.

2.3.5 TheHive

TheHive is a free web-based open-source automated incident responder tool. It acts as a central repository for case management. It's an interactive GUI, a dashboard displaying pie charts to induce a centralized view of overall security. Its three core functionalities:

2.3.5.1 Collaborate: TheHive connects people with multiple SOC, CERT, CSIRT, and other security analysts who can work together on a case, share information and keep everyone updated with their progress. [Can make the connection between similar cases]

2.3.5.2 Elaborate: It provides automation by allowing workflow creation where you'll create a playbook containing predefined procedural steps/ tasks to follow when a specific incident happened [9]. For documentation purposes, it allows case template creation. Users can create/modify a template in step with their requirements by adding different headers like tags (based on the MITRE attack database), timeline, metrics, and far more.

2.3.5.3 Act / Analyze: TheHive integrates the functionality of MISP (Open source threat Intel platform) and Cortex (Analyze and answer incidents) to perform enrichment, contextualization, analysis, and response. TheHive can have bidirectional communication with MISP which suggests that TheHive may additionally get information (IOCs, TTP, etc.) from MISP also if new IOCs are detected TheHive may update the information on the Malware information sharing platform. [19]

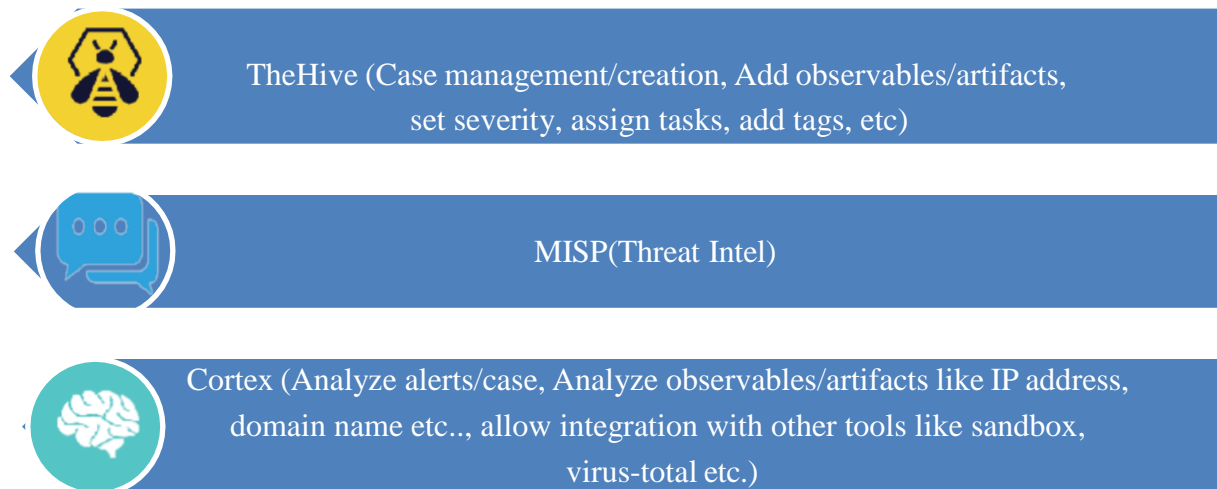


Figure 2.19: TheHive Components

Chapter 3

Use Case Analysis

Chapter 3: Use Case Analysis

In the Usecase diagram, the functionalities and use cases that are provided by the tools integrated are shown as well as the diagram also shows the use cases that the administrator is allowed or can do.

3.1 Use Case Model

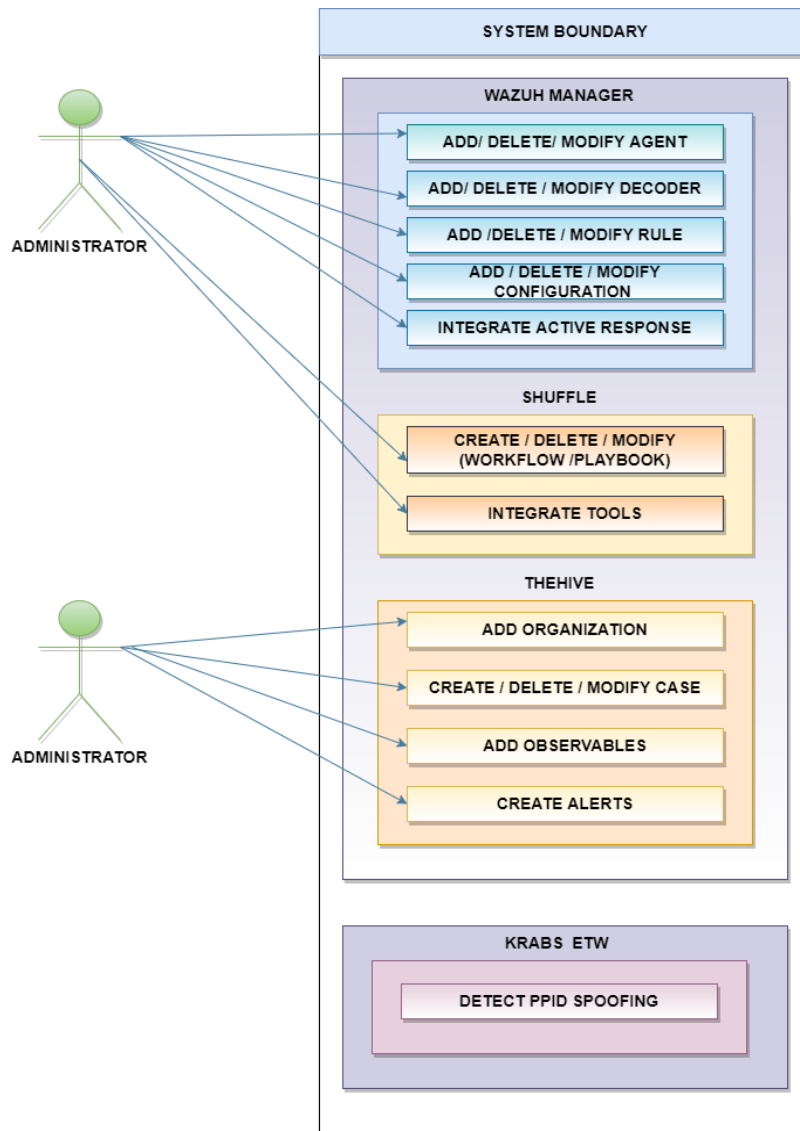


Figure 3.1: Usecase Diagram

Chapter 4

System Design

Chapter 4: System Design

This chapter describes the project architecture, data flow, activities, and major processes that are done to achieve the automated incident response, detection, and notification to the administrator for the attack of Parent Process Id Spoofing.

4.1. Architecture Diagram

This project is been divided into the lanes that the Krabs ETW detects the PPID spoofing and spoofing.log file is sent to the Wazuh Agent and alert is displayed on the dashboard of the Manager and after an alert, the AR.exe is executed. The AR log file would also be updated after the Wazuh alerts are sent to the Shuffle via Webhook. The Shuffle then sent an alert to The Hive and lastly on the Discord server the alert is sent after that administrator is allowed to conduct deep inspection or monitoring.

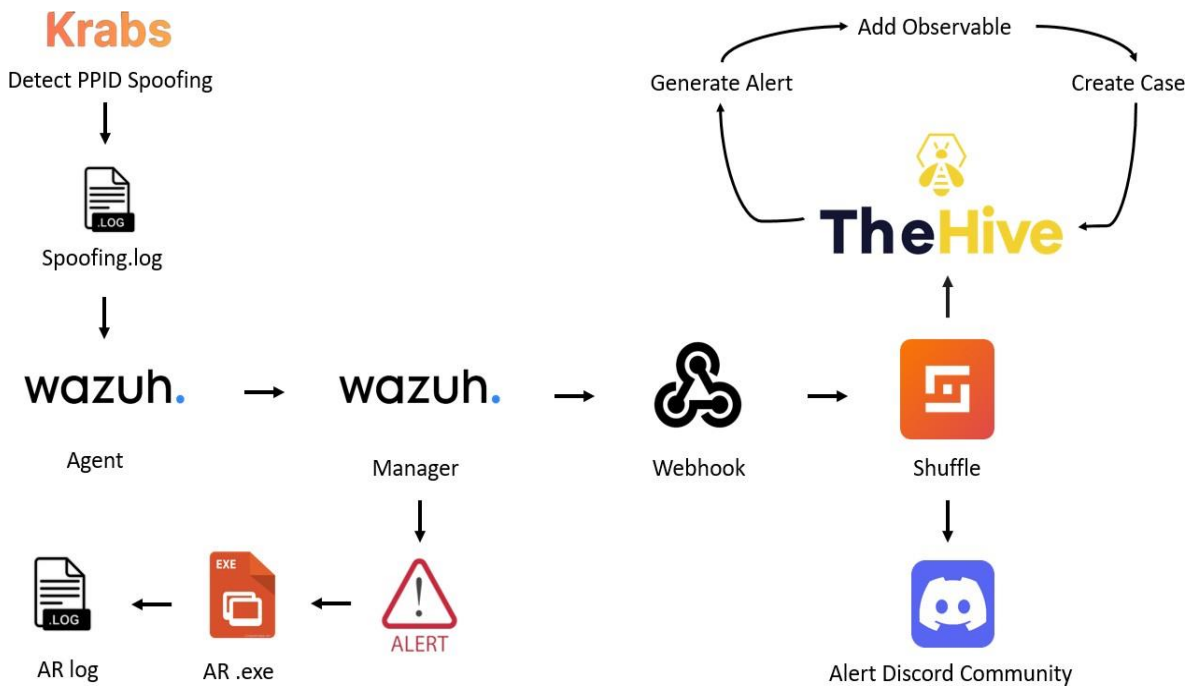


Figure 4.1: Architecture Diagram

4.2. Activity Diagram

The diagram shown below describes the activities that are happening in order from the launch of the attack to elimination till notifying the Administrator via the Discord community server. In this Project, the entities that are involved are both humans and platforms (tools).

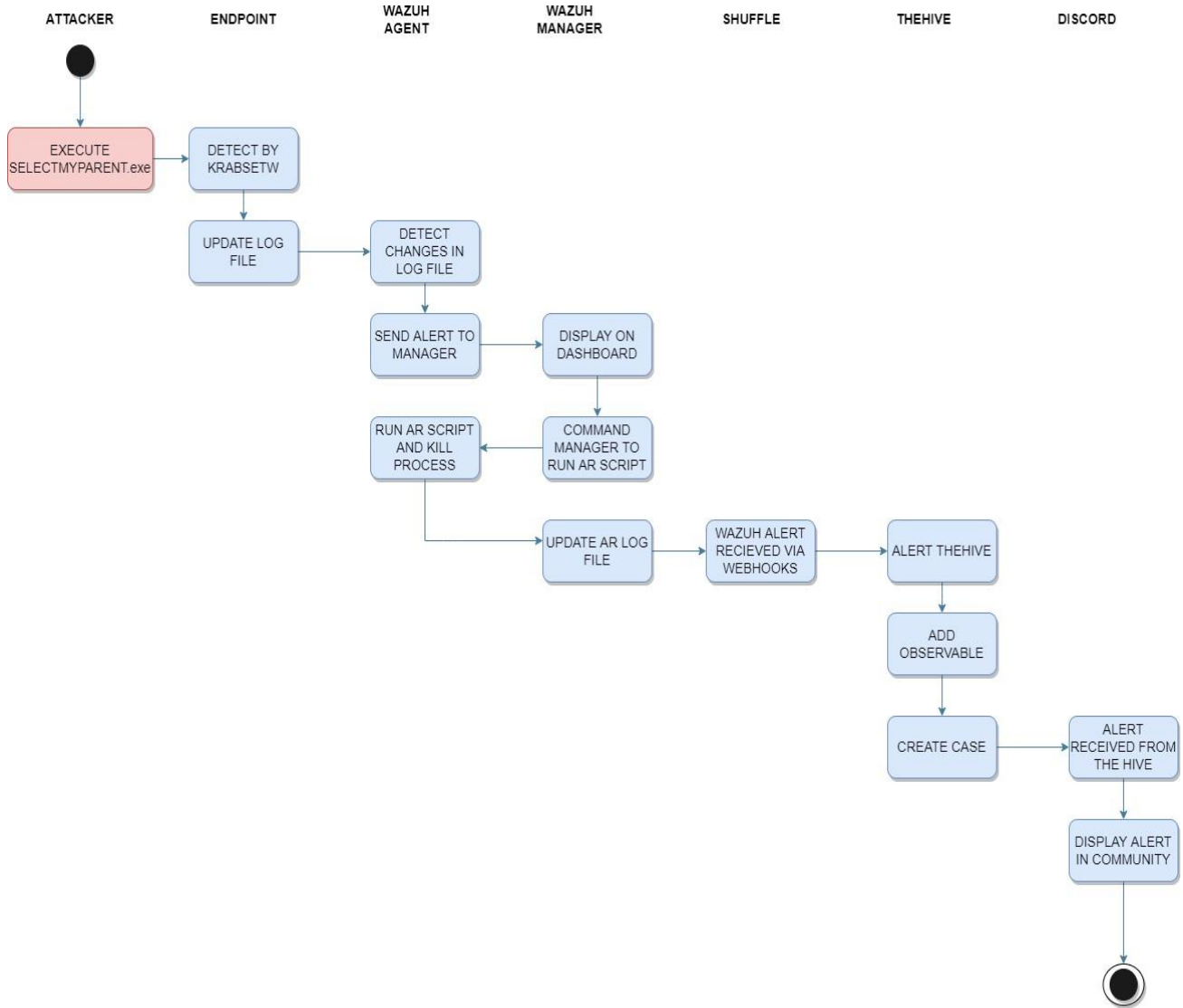


Figure 4. 2: Activity Diagram

4.3. Sequence Diagram

In the Diagram given below the sequential flow of the project is shown, the grey part shows that after the incident if the administrator wanted to take further action. The action of Logging into the endpoint or other tools is allowed. Moreover, the administrator can run analyzers and responders, create, delete or modify the case, and Assign the case to another member of the team or an organization. The alerts within the different tools can also be monitored.

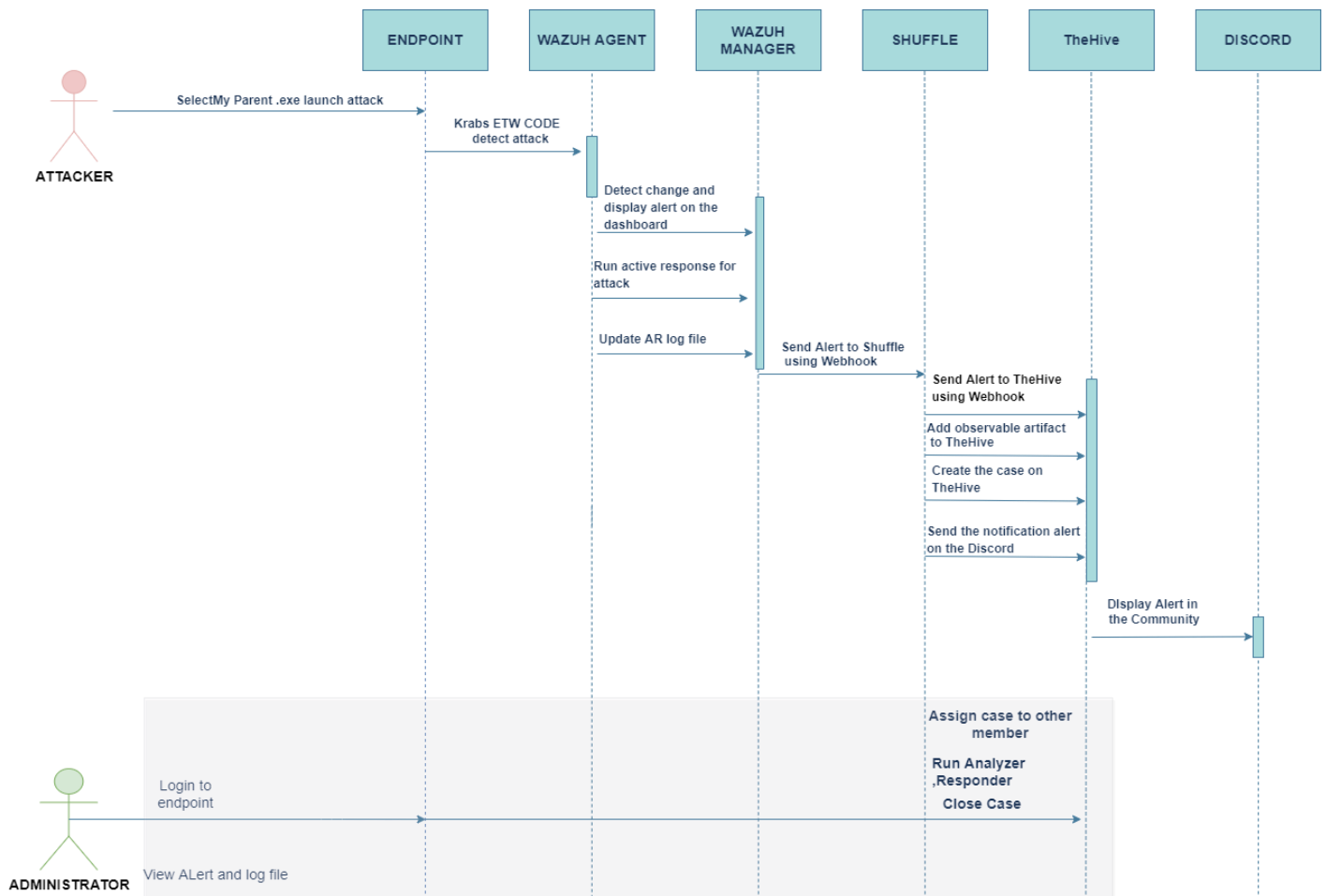


Figure 4.3: Sequence Diagram

4.4. State Diagram

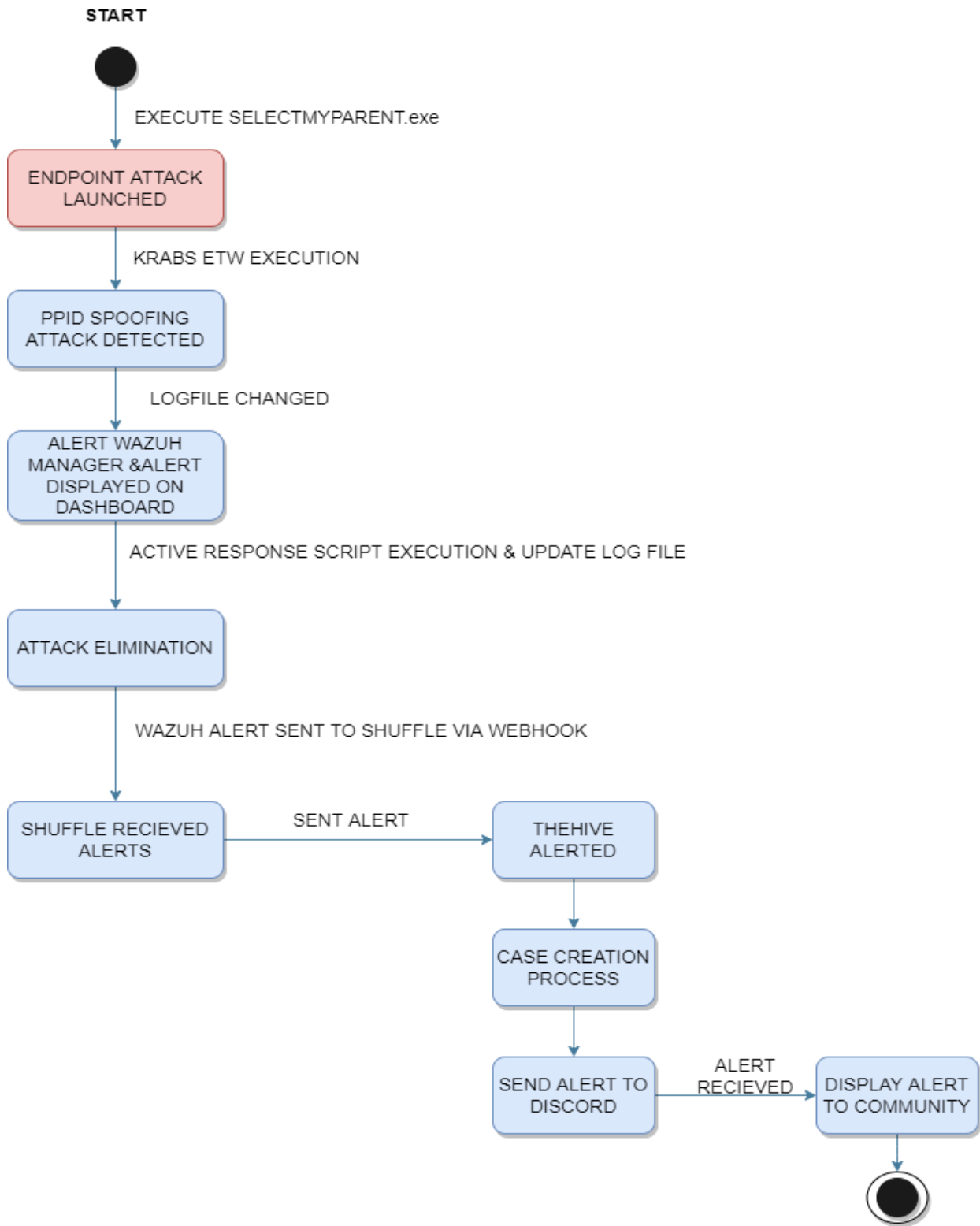


Figure 4.4: State Diagram

4.5. Data Flow diagram

The diagram depicts the flow of data from the attacker who launched the PPID spoofing attack on the endpoint and different tools like TheHive, Shuffle, Wazuh Manager, and Agent. The diagram below is of context also known as Level 0.

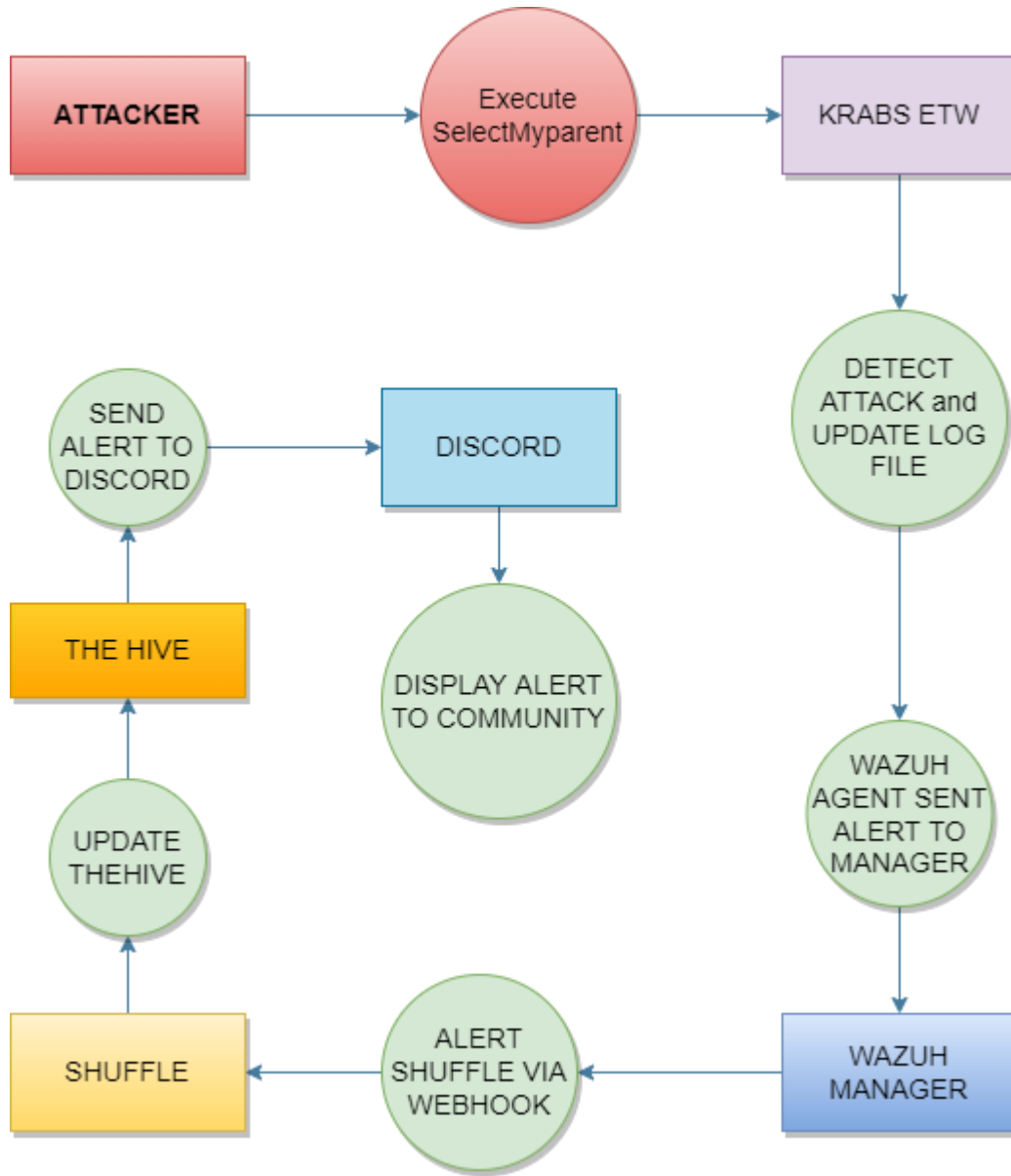


Figure 4.5: Data Flow Diagram (Level 0)

Chapter 5

Implementation

Chapter 5: Implementation

This chapter would include the detailed steps and methodologies used in the project along with the code written for the execution and detection of the attack.

5.1. Important Flow Control/Pseudo codes

5.1.1 Krabs ETW Code

```
#include
"C:\Users\PC\.nuget\packages\microsoft.o365.security.krabs.etw\4.2.2\lib\native\include\krabs.hpp"
#include <iostream>
#include <tlhelp32.h>
#include <fstream>
#include <string>

int Error(const char* msg) {
    std::cout << "[+] Error: " << msg << " " << GetLastError() << std::endl;
    return 1;
}

// Resolve a PID to a process name
std::wstring GetProcessName(uint32_t pid) {
    HANDLE hProcessSnap;
    PROCESSENTRY32 pe32;

    hProcessSnap = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
    if (hProcessSnap == INVALID_HANDLE_VALUE)
    {
        Error("Error in CreateToolhelp32Snapshot");
        return L"<Error>";
    }
    pe32.dwSize = sizeof(PROCESSENTRY32);

    if (!Process32First(hProcessSnap, &pe32))
    {
        Error("Error in Process32First\n");
        CloseHandle(hProcessSnap);
        return L"<Error>";
    }
    do
    {
        if (pid == pe32.th32ProcessID) {
            return pe32.szExeFile;
        }

    } while (Process32Next(hProcessSnap, &pe32));
    CloseHandle(hProcessSnap);
    return L"<Error>";
}

void begin_trace()
{
    krabs::user_trace trace;
    krabs::provider<> provider(L"Microsoft-Windows-Kernel-Process");
}
```



```

provider.any(0x10); // WINEVENT_KEYWORD_PROCESS

auto process_callback = [](const EVENT_RECORD& record, const krabs::trace_context& trace_context) {
    std::wofstream Myfile;
    Myfile.open("C:\\Users\\PC\\source\\repos\\CRABETW\\CRABETW\\Spoofed_Data.log", std::ios::app);
    krabs::schema schema(record, trace_context.schema_locator);
    krabs::parser parser(schema);
    uint32_t ppid = parser.parse<uint32_t>(L"ParentProcessID");
    // record.EventHeader.ProcessId is the creator process ID
    // Check whether the PPID is the same as the creator PID
    if (ppid != record.EventHeader.ProcessId) {
        uint32_t pid = parser.parse<uint32_t>(L"ProcessID");
        std::wstring image_name = parser.parse<std::wstring>(L"ImageName");
        // Get the image name from the real parent process ID
        std::wstring real_parent_process_name = GetProcessName(record.EventHeader.ProcessId);
        std::wstring fake_parent_process_name = GetProcessName(ppid);
        Myfile << "fake_parent_process_name=";
        Myfile << fake_parent_process_name;
        Myfile << ",";
        Myfile << " ";
        Myfile << "real_parent_process_name=";
        Myfile << real_parent_process_name;
        Myfile << ",";
        Myfile << " ";
        Myfile << "path=";
        Myfile << image_name;
        Myfile << ",";
        Myfile << " ";
        Myfile << "pid=";
        Myfile << pid;
        Myfile << ",";
        Myfile << " ";
        Myfile << "ppid=";
        Myfile << ppid;
        Myfile << "\n";
        std::wcout << L"[!] Process PPID Spoofing Detected" << std::endl;
        std::wcout << L"> Real Parent: " << real_parent_process_name << std::endl;
        std::wcout << L"> Fake Parent: " << fake_parent_process_name << std::endl;
        std::wcout << L"> Process Name: " << image_name << std::endl;
        std::wcout << L"> ProcessID " << pid << std::endl;
        std::wcout << std::endl;
        Myfile.close();
    }
};

// real-time process start events
krabs::event_filter process_filter(krabs::predicates::id_is(1));
process_filter.add_on_event_callback(process_callback);
provider.add_filter(process_filter);

trace.enable(provider);
trace.start();

}

int main()
{
    std::cout << "[+] Monitoring Starting!\n" << std::endl;
    begin_trace();
}

```

```
}
```

5.1.2 Decoder (Wazuh Manager)

```
<decoder name="pid-custom">
  <prematch>^fake_parent_process_name=\S+</prematch>
  <regex>^fake_parent_process_name=(\S+), real_parent_process_name=(\S+), path=(\S+), pid=(\d+),
  ppid=(\d+)</regex>
  <order>fake_parent_process_name, real_parent_process_name, path, pid, ppid</order>
</decoder>
```

5.1.3 Rule (Wazuh Manager)

```
<rule id="222001" level="12">
  <decoded_as>pid-custom</decoded_as>
  <description>PPID Spoofing Detected</description>
  <mitre>
    <id>T1502</id>
  </mitre>
</rule>
```

5.1.4 Active Response Code(python)

```
import os
import sys
import psutil
import json
import datetime

if os.name == 'nt':
    LOG_FILE = "C:\\Program Files (x86)\\ossec-agent\\active-response\\active-responses.log"
else:
    LOG_FILE = "/var/ossec/logs/active-responses.log"

ADD_COMMAND = 0
DELETE_COMMAND = 1
CONTINUE_COMMAND = 2
ABORT_COMMAND = 3

OS_SUCCESS = 0
OS_INVALID = -1

class message :
    def __init__(self) :
        self.alert = ""
        self.command = 0

    def write_debug_file(ar_name, msg) :
        with open(LOG_FILE, mode = "a") as log_file :
            log_file.write(str(datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')) + " " + ar_name + ": " +
            msg + "\n")

def setup_and_check_message(argv) :

    # get alert from stdin
    Department of Cyber Security, Air University, Islamabad, Pakistan
```

```
    input_str = ""
    for line in sys.stdin:
input_str = line
break

write_debug_file(argv[0], input_str)

try :
    data = json.loads(input_str)
    except ValueError :
write_debug_file(argv[0], 'Decoding JSON has failed, invalid input format')
message.command = OS_INVALID
return message

message.alert = data

command = data.get("command")

if command == "add":
message.command = ADD_COMMAND
elif command == "delete" :
    message.command = DELETE_COMMAND
else:
message.command = OS_INVALID
write_debug_file(argv[0], 'Not valid command: ' + command)

return message

def send_keys_and_check_message(argv, keys) :

    # build and send message with keys
    keys_msg = json.dumps({ "version": 1, "origin" : {"name": argv[0], "module" : "active-
response"}, "command" : "check_keys", "parameters" : {"keys":keys} })

    write_debug_file(argv[0], keys_msg)

    print(keys_msg)
    sys.stdout.flush()

    # read the response of previous message
    input_str = ""
    while True:
line = sys.stdin.readline()
if line :
    input_str = line
    break

    write_debug_file(argv[0], input_str)

    try :
    data = json.loads(input_str)
    except ValueError :
write_debug_file(argv[0], 'Decoding JSON has failed, invalid input format')
return message

action = data.get("command")

if "continue" == action:
ret = CONTINUE_COMMAND
```

```

elif "abort" == action :
    ret = ABORT_COMMAND
else:
    ret = OS_INVALID
write_debug_file(argv[0], "Invalid value of 'command'")

return ret

def kill_process(keys) :
    log = str(keys)
    pid = " "
    split_log = log.split(",")
    pid = split_log[3][5:]
    p = psutil.Process(int(pid))
    p.kill()

def main(argv) :

    write_debug_file(argv[0], "Started")

    # validate json and get command
    msg = setup_and_check_message(argv)

    if msg.command < 0:
        sys.exit(OS_INVALID)

if msg.command == ADD_COMMAND :

    """ Start Custom Key
    At this point, it is necessary to select the keys from the alert and add them to the keys array.
    """

    alert = msg.alert["parameters"]["alert"]
    keys = [alert["full_log"], alert["rule"]["id"], alert["timestamp"]]
    kill_process(keys[0])

    """ End Custom Key """

    action = send_keys_and_check_message(argv, keys)

# if necessary, abort execution
if action != CONTINUE_COMMAND:

if action == ABORT_COMMAND :
    write_debug_file(argv[0], "Aborted")
    sys.exit(OS_SUCCESS)
else :
    write_debug_file(argv[0], "Invalid command")
    sys.exit(OS_INVALID)

    """ Start Custom Action Add """

    with open("ar-test-result.txt", mode = "a") as test_file :
test_file.write("    *** --- ***    " + ">\n")
test_file.write("Active response triggered by rule ID: <" + str(keys[1]) + ">\n")
test_file.write("Active response triggered at timestamp: <" + str(keys[2]) + ">\n")
test_file.write("Active response Full log is: <" + str(keys) + ">\n")
test_file.write("    *** --- ***    " + ">\n")

```

```

""" End Custom Action Add """

elif msg.command == DELETE_COMMAND:

    """ Start Custom Action Delete """

    os.remove("ar-test-result.txt")

    """ End Custom Action Delete """

    else:
    write_debug_file(argv[0], "Invalid command")

    write_debug_file(argv[0], "Ended")

    sys.exit(OS_SUCCESS)

if __name__ == "__main__":
    main(sys.argv)

```

5.1.5 .BIN FILE

```

#!/bin/sh
# Created by Shuffle, AS. <frikky@shuffler.io>.

WPYTHON_BIN = "framework/python/bin/python3"

SCRIPT_PATH_NAME = "$0"

DIR_NAME = "$(cd $(dirname ${SCRIPT_PATH_NAME}); pwd -P)"
SCRIPT_NAME = "$(basename ${SCRIPT_PATH_NAME})"

case ${ DIR_NAME } in
* / active - response / bin | */ wodles*)
if [-z "${WAZUH_PATH}"]; then
WAZUH_PATH = "$(cd ${DIR_NAME}/../..; pwd)"
fi

PYTHON_SCRIPT = "${DIR_NAME}/${SCRIPT_NAME}.py"
;;
*/ bin)
if [-z "${WAZUH_PATH}"]; then
WAZUH_PATH = "$(cd ${DIR_NAME}/..; pwd)"
fi

PYTHON_SCRIPT = "${WAZUH_PATH}/framework/scripts/${SCRIPT_NAME}.py"
;;
*/ integrations)
if [-z "${WAZUH_PATH}"]; then
WAZUH_PATH = "$(cd ${DIR_NAME}/..; pwd)"
fi

PYTHON_SCRIPT = "${DIR_NAME}/${SCRIPT_NAME}.py"
;;
esac

${ WAZUH_PATH } / ${ WPYTHON_BIN } ${ PYTHON_SCRIPT } "$@"

```

5.1.6 .PY FILE

```
#!/usr/bin/env python
# Created by Shuffle, AS. <frikky@shuffler.io>.
# Based on the Slack integration using Webhooks

import json
import sys
import time
import os

try :
    import requests
    from requests.auth import HTTPBasicAuth
    except Exception as e :
print("No module 'requests' found. Install: pip install requests")
sys.exit(1)

# ADD THIS TO ossec.conf configuration :
# <integration>
#     <name>custom-shuffle</name>
#     <hook_url>http://<IP>:3001/api/v1/hooks/<HOOK_ID></hook_url>
#     <level>3</level>
#     <alert_format>json</alert_format>
# </integration>

# Global vars

debug_enabled = False
pwd = os.path.dirname(os.path.dirname(os.path.realpath(__file__)))
json_alert = {}
now = time.strftime("%a %b %d %H:%M:%S %Z %Y")

# Set paths
log_file = '{0}/logs/integrations.log'.format(pwd)

def main(args) :
    debug("# Starting")

    # Read args
    alert_file_location = args[1]
    webhook = args[3]

    debug("# Webhook")
    debug(webhook)

    debug("# File location")
    debug(alert_file_location)

    # Load alert.Parse JSON object.
    with open(alert_file_location) as alert_file :
json_alert = json.load(alert_file)
debug("# Processing alert")
debug(json_alert)

debug("# Generating message")
msg = generate_msg(json_alert)
if isinstance(msg, str) :
    if len(msg) == 0 :
```

```

return
debug(msg)

debug("# Sending message")
send_msg(msg, webhook)

def debug(msg) :
if debug_enabled :
    msg = "{0}: {1}\n".format(now, msg)
    print(msg)
    f = open(log_file, "a")
    f.write(msg)
    f.close()

    # Skips container kills to stop self - recursion
    def filter_msg(alert) :
        # These are things that recursively happen because Shuffle starts Docker containers
        skip = ["87924", "87900", "87901", "87902", "87903", "87904", "86001", "86002", "86003",
"87932", "80710", "87929", "87928", "5710"]
        if alert["rule"]["id"] in skip :
return False

#try :
#    if "docker" in alert["rule"]["description"].lower() and "
#msg['text'] = alert.get('full_log')
#except:
#    pass
#msg['title'] = alert['rule']['description'] if 'description' in alert['rule'] else "N/A"

return True

def generate_msg(alert) :
    if not filter_msg(alert) :
        print("Skipping rule %s" % alert["rule"]["id"])
        return ""

        level = alert['rule']['level']

        if (level <= 4) :
            severity = 1
        elif(level >= 5 and level <= 7) :
            severity = 2
        else:
severity = 3

msg = {}
msg['severity'] = severity
msg['pretext'] = "WAZUH Alert"
msg['title'] = alert['rule']['description'] if 'description' in alert['rule'] else "N/A"
msg['text'] = alert.get('full_log')
msg['rule_id'] = alert["rule"]["id"]
msg['timestamp'] = alert["timestamp"]
msg['id'] = alert['id']
msg["all_fields"] = alert

#msg['fields'] = []
#    msg['fields'].append({
#        "title": "Agent",

```

```

#         "value": "{0} - {1}".format(
#             alert['agent']['id'],
#             alert['agent']['name']
#         ),
#     })
# if 'agentless' in alert:
#     msg['fields'].append({
#         "title": "Agentless Host",
#         "value": alert['agentless']['host'],
#     })

#msg['fields'].append({ "title": "Location", "value" : alert['location'] })
#msg['fields'].append({
#     "title": "Rule ID",
#     "value": "{0} _(Level {1})_".format(alert['rule']['id'], level),
# })

#attach = { 'attachments': [msg] }

return json.dumps(msg)

def send_msg(msg, url) :
headers = { 'content-type': 'application/json', 'Accept-Charset' : 'UTF-8' }
res = requests.post(url, data = msg, headers = headers)
debug(res)

if __name__ == "__main__":
try :
# Read arguments
bad_arguments = False
if len(sys.argv) >= 4:
msg = '{0} {1} {2} {3} {4}'.format(
now,
sys.argv[1],
sys.argv[2],
sys.argv[3],
sys.argv[4] if len(sys.argv) > 4 else ''
)
debug_enabled = (len(sys.argv) > 4 and sys.argv[4] == 'debug')
else:
msg = '{0} Wrong arguments'.format(now)
bad_arguments = True

# Logging the call
f = open(log_file, 'a')
f.write(msg + '\n')
f.close()

if bad_arguments:
debug("# Exiting: Bad arguments.")
sys.exit(1)

# Main function
main(sys.argv)

except Exception as e :
debug(str(e))

```


raise

5.1.7 SHUFFLE INTEGRATION

```
<integration>
<name>custom-shuffle</name>
<rule_id>222001</rule_id>
<hook_url>https://192.168.85.138:3443/api/v1/hooks/webhook_a8e9a934-7077-43cd-9c40-9c6c310044eb</hook_url>
<alert_format>json</alert_format>
</integration>
```

5.2. Components, Libraries, Web Service

At first, the generation of the authentication key is required to build the connection of the Wazuh agent deployed on an endpoint with the Wazuh Manager deployed on the virtual machine. Afterward, authentication is added to the TheHive app in Shuffle (which can be done by adding the authentication for automating or by adding it manually). For the integration of Wazuh and Shuffle, the webbook is created the unique URL is generated. Lastly, the same process is done for the Discord. The code for detection is written with the help of Krabs ETW which is a library in C++ language that helps in simplifying the interaction with the event tracing window. PsUtil is the python cross-platform library used in the script that allows to access the details about the system and process utilities.

5.3. Deployment Environment

The Visual Studio is used for the execution of the attack on an endpoint before running the attack script detection script should be running to capture and detect the attack. Then the Syslog file would be sent to the Wazuh Manager via the Wazuh Agent. Wazuh would automatically run the AR (active response) exe and kill the child process that could be verified through the log file. The Active response exe is written in python using the library PsUtil.

5.4. Tools Version

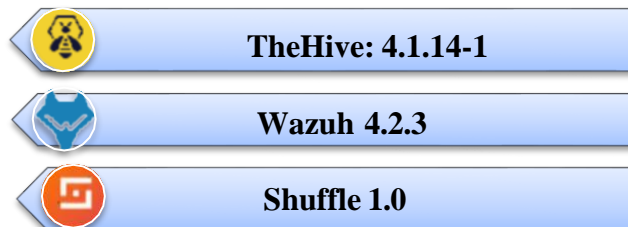


Figure 5.1: Tools version

Chapter 6

Business Plan

Chapter 6: Business Plan

The project is creating value in the market as by providing the automation and incident response in the current open-source existing SOAR solutions, analysts, end-point users, and organizations would be able to safeguard themselves against the attack of parent process identifier spoofing. The project proposes a way for organizations to devise attack-specific automated workflows and incident responses.

6.1 Business Description

The project introduced the ease of use through automation in the process of identification, reporting, and eliminating the attack (PPID Spoofing). The user would be at ease, as the process would be done in an automated manner with efficiency, and time constraints would also be reduced.

6.2 Market Analysis & Strategy

Currently, the market is been divided into the organizations that have been settled for SIEM solutions and others are the ones who are moving towards the SOAR to reduce the human resource and cost, the SOAR is been an addition and innovation to the current SIEM solutions. Furthermore, the market has the paid and open-source platforms to provide the automation which is been discussed in detail in the chapter1

6.3 Competitive Analysis

Comparatively, the addition and the add-on is been done with the open-source SOAR solutions (The Hive), which would serve as the unique feature and point of difference, because specifically dealing with the PPID attack would be paving the way for the addition of other attack scenarios and use cases. The addition of specific attack use cases in the existing open-source Soar would contribute to shielding against the latest attacks and vulnerabilities. It would specifically target the segmentation of the organization that wanted security but had a low budget.

6.4 Products/Services Description

The service of the additional use case of PPID added to the Hive that is an open-source case management platform, the services would be the detection of the PPID attack on the endpoint, later the ingestion of the logs on the Wazuh Agent and Wazuh server, All the logs regarding attack detection would be shown on The Hive as well as an automated incident response would be done. The Shuffle is utilized for the creation of workflow and automation is provided.

6.5 SWOT Analysis

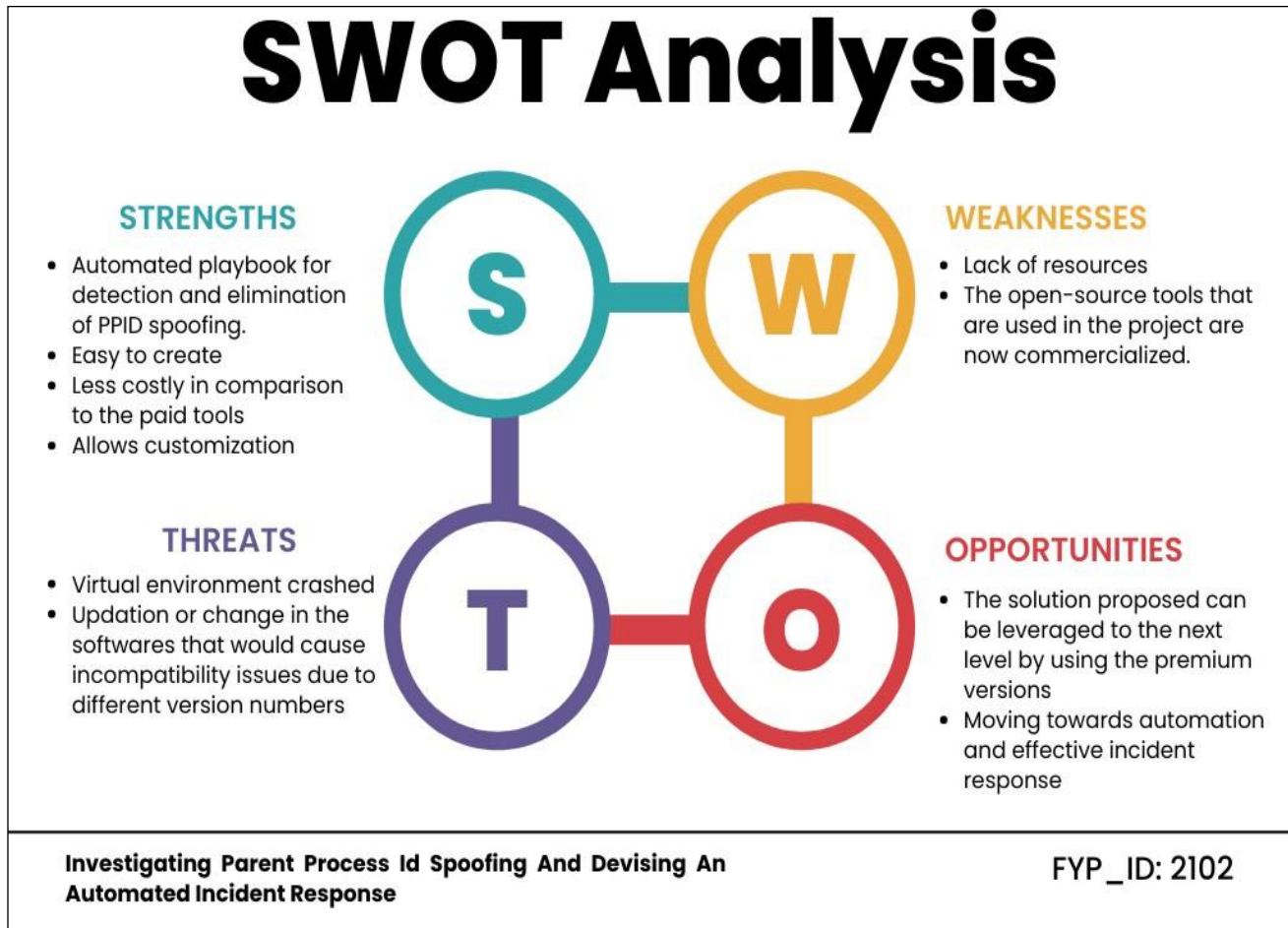


Figure 6.1: SWOT Analysis

Chapter 7

Testing & Evaluation

Chapter 7: Testing and Evaluation

This chapter describes the testing and evaluation that is done to ensure the proper and smooth functionality of the project. The chapter includes the testing following Usecase, performance, data flow, and integration.

The testing of this project is done with the help of an example.

7.1 Generating PPID Spoofing attack (DEMONSTRATION)

7.1.1 Exe for PPID Spoofing attack simulation

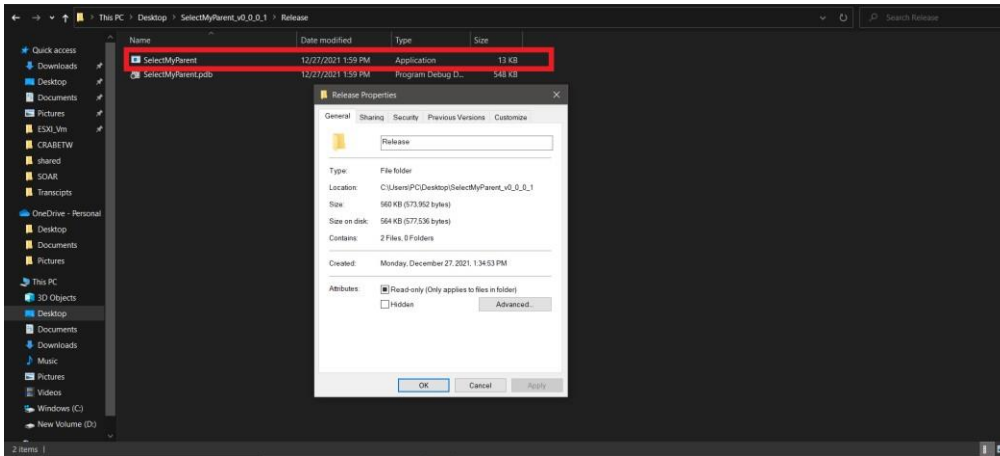


Figure 7.1: Attack is been launched on endpoint using the select my parent exe

Find the PID you would like to spoof (In my case I selected explorer.exe as a spoofed parent)

Note down the Process ID

explorer.exe	21728	Running	PC	00	57,260 K	Disabled
--------------	-------	---------	----	----	----------	----------

7.1.2 Running detection code

Note: Before launching an attack make sure your detector is running then run Visual Studio with admin privileges

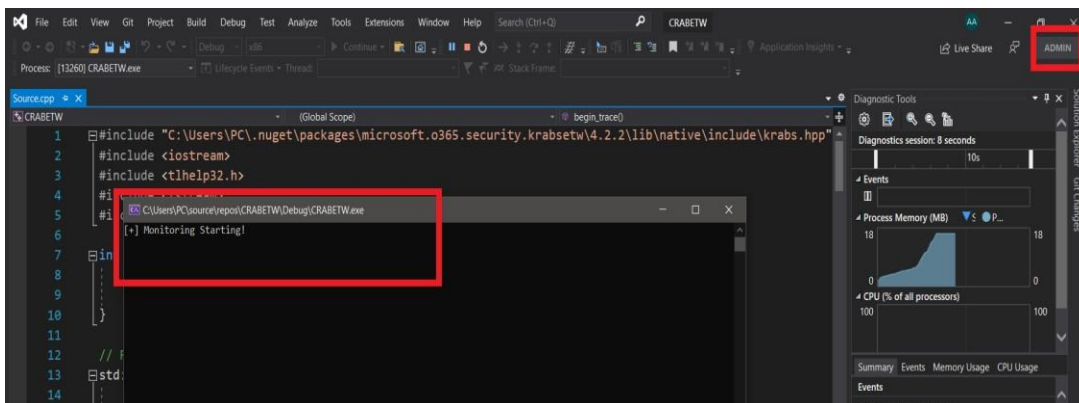


Figure 7.2: Attack is been launched on endpoint and Krabs ETW code is running in background monitoring

Any process can be selected for the spoofing of the parent from any process.

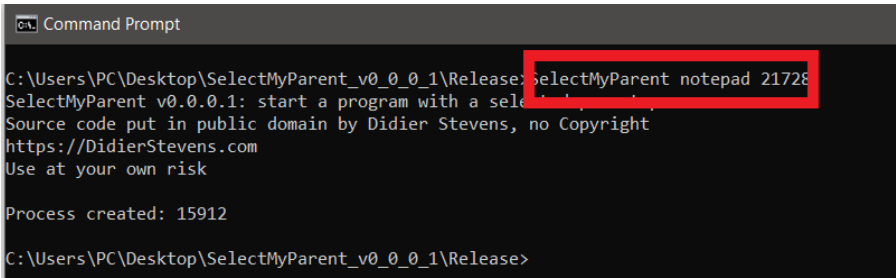


Figure 7.3: Attack is been launched on endpoint using the select my parent exe

7.1.3 Alert on Wazuh

Alert is visible on the Wazuh Manager Dashboard

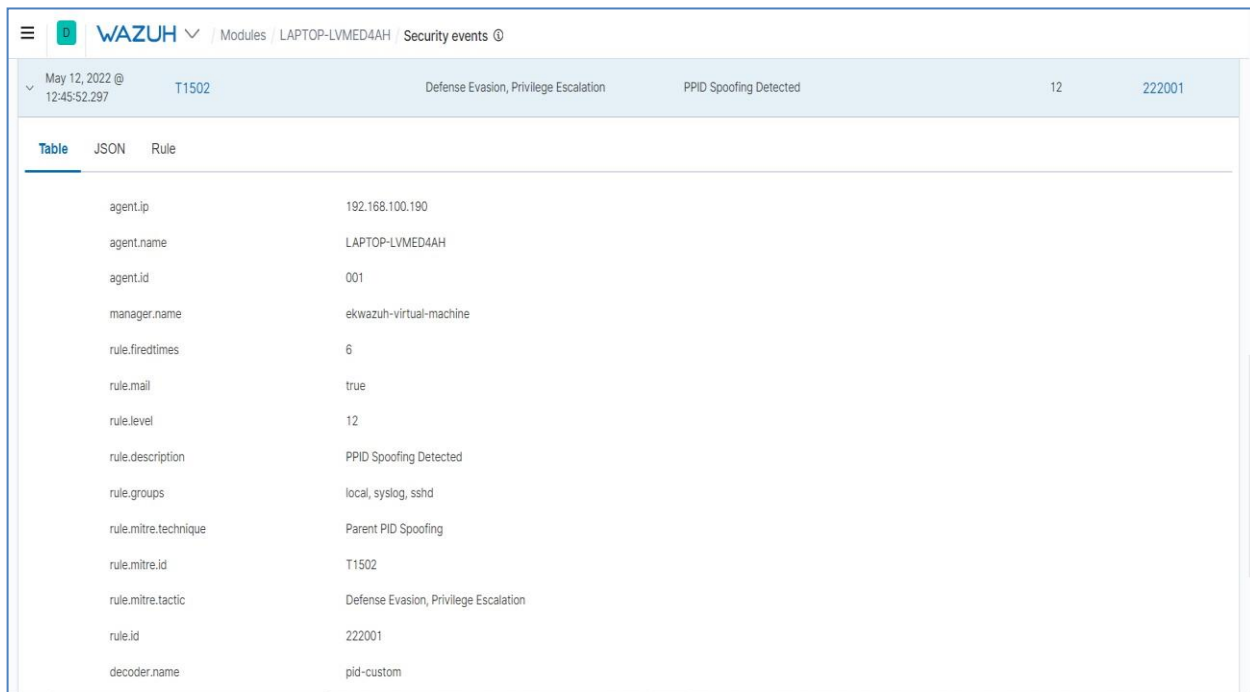


Figure 7.4: After detection the log is been sent to the Wazuh Manager via Wazuh Agent

7.1.4 Wazuh Active Response (AR)

Wazuh Automatically runs the AR exe and kills the child process and to confirm we can check the log file

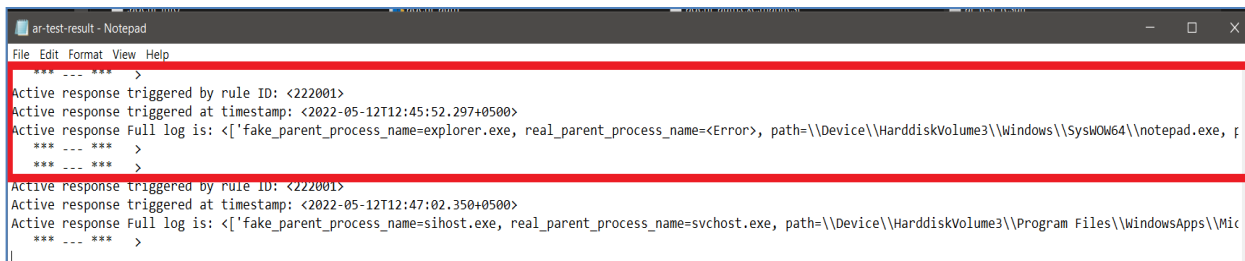


Figure 7.5: Active Response rule is been triggered

7.1.5 Playbook/Workflow Execution

Go to Shuffle and check whether the playbook/workflow executes properly

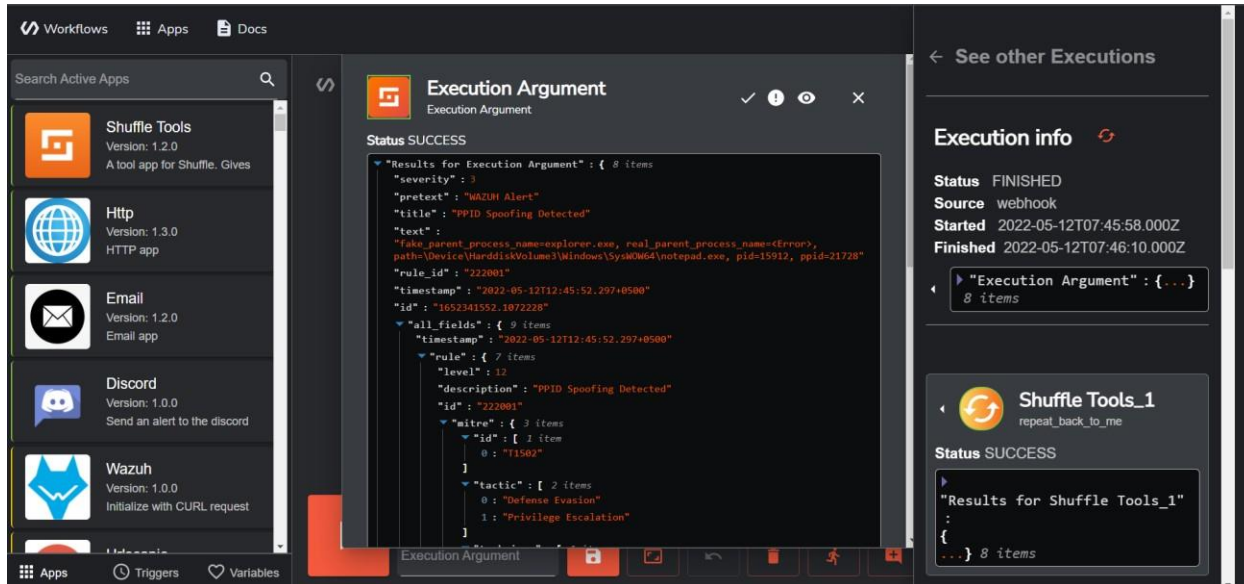


Figure 7.6: Shuffle alerts and execution argument section for the verification of smooth execution

Don't forget to check every node's status

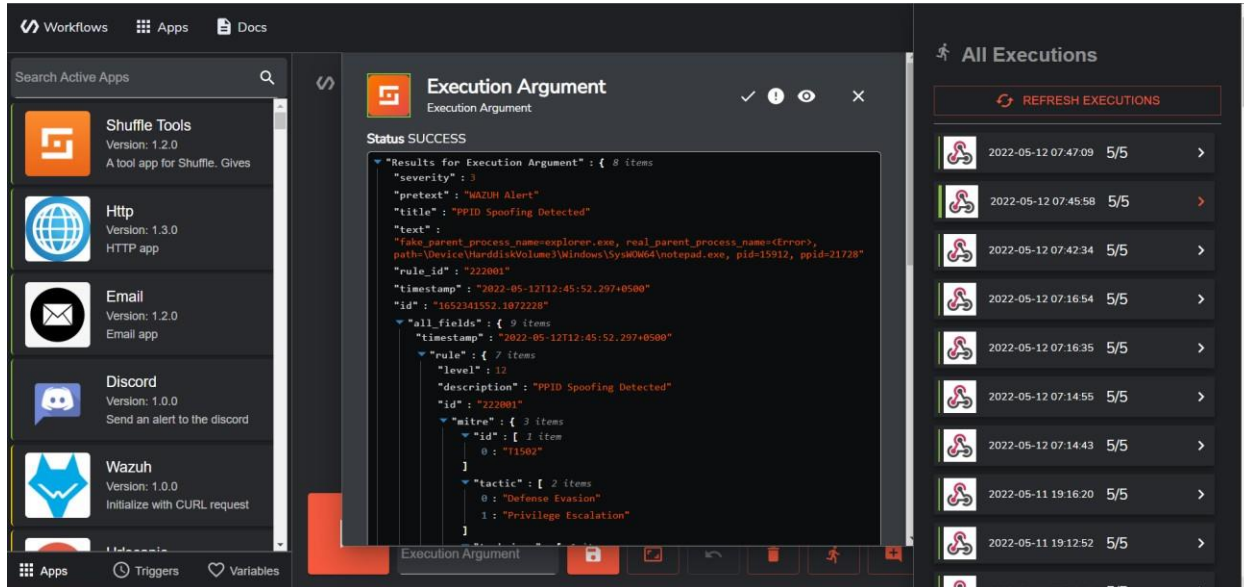


Figure 7.7: Shuffle alerts and execution argument section for the verification of smooth execution on every node

7.1.6 Case creation on TheHive

Check if Case is created on the TheHive app

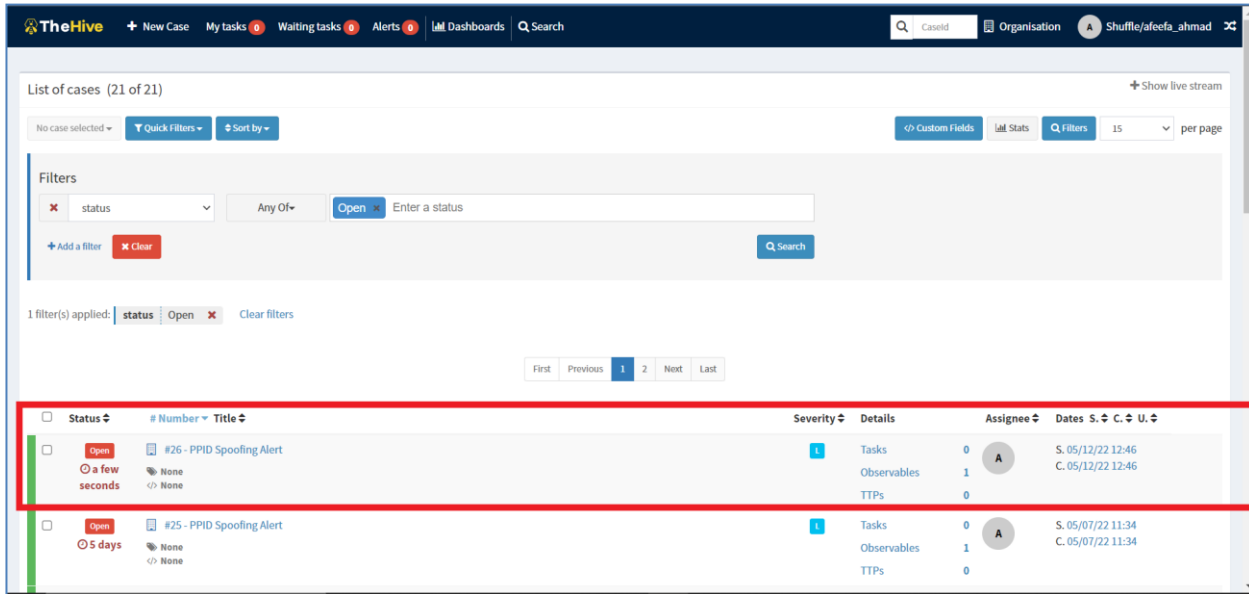


Figure 7.8: Case creation on TheHive Case Management Platform

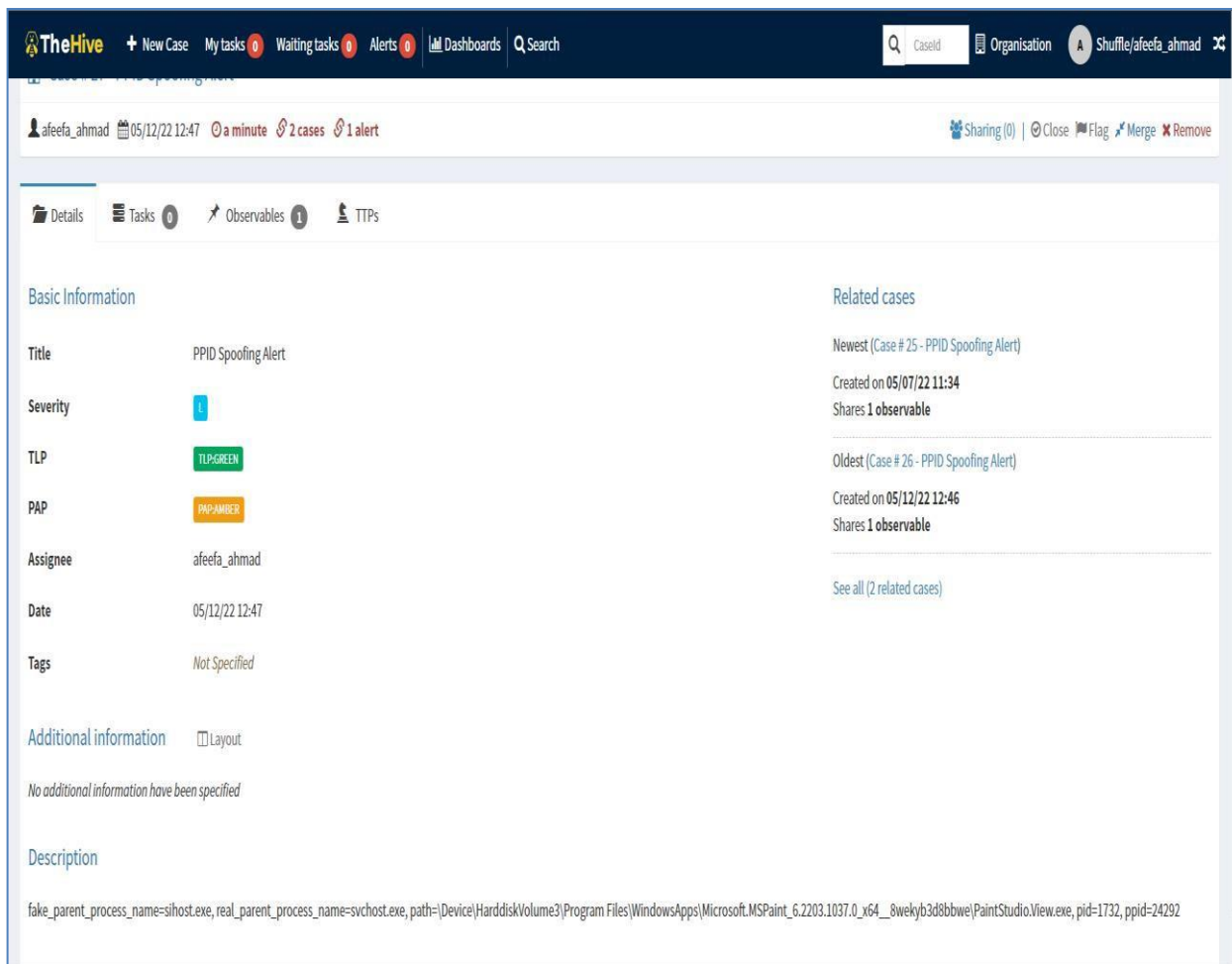


Figure 7.9: Complete and detailed PPID Spoofing attack alert

7.1.7 Alert on Discord

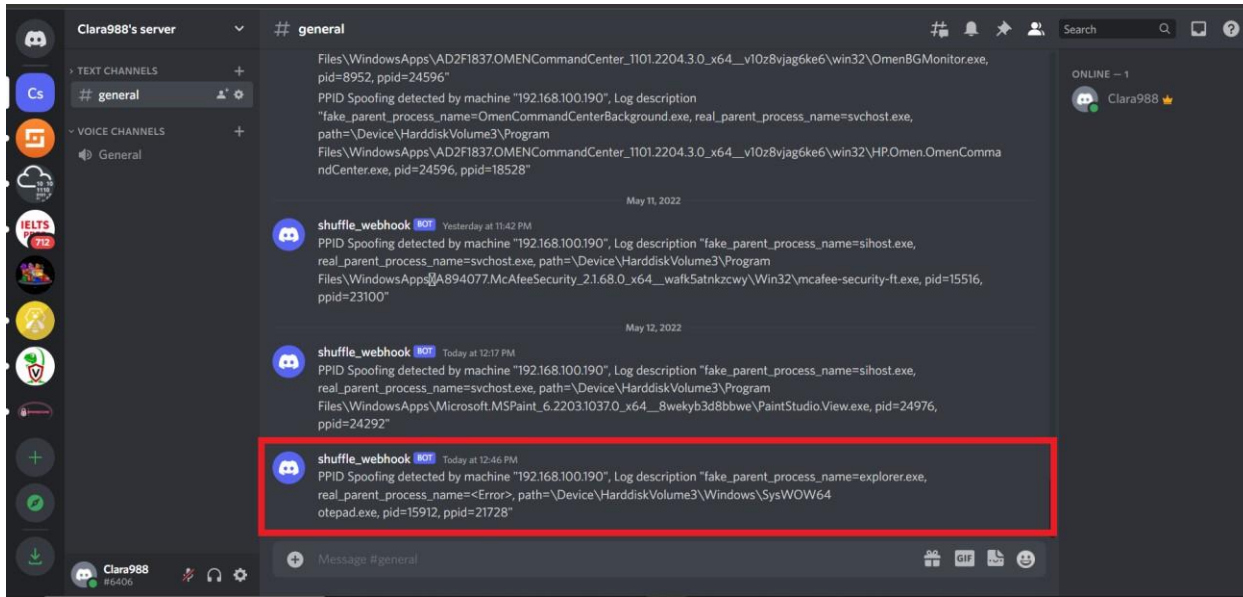


Figure 7.10: Alert and detailed process is posted in automated manner on The Discord

7.2 Use Case Testing (Test Cases)

7.2.1 Test Case - 1

Test Case ID	IR_001	Test Case Description	Testing the launch of PPID spoofing attack		
Created By	Afeefa	Reviewed By	Afeefa	Version	A.1
QA Tester's Log	Refer demonstration heading				

Tester's Name	Afeefa	Date Tested	May 12, 22	Test Case (Pass/Fail/Not Executed)	Pass
----------------------	--------	--------------------	------------	---	------

S #	Prerequisites:
1	Access to Task Manager
2	SelectMyParent.exe

S #	Test Data
1	PPID = 21728
2	Child Process= Notepad

Test Scenario

Verify whether the PPID spoofing attack was successfully launched or not?

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Task manager access	Should have access	As Expected	Pass
2	Enter PPID & child process name	PPID and child process name can be entered	As Expected	Pass

7.2.2 Test Case – 2

Test Case ID	IR_002	Test Case Description	Detecting the PPID spoofing attack		
Created By	Afeefa	Reviewed By	Afeefa	Version	B.1

QA Tester's Log Spoofing_data.log file on the endpoint

Tester's Name	Afeefa	Date Tested	May 12, 22	Test Case (Pass/Fail/Not Executed)	Pass
----------------------	--------	--------------------	------------	---	------

S #	Prerequisites:
1	Access to Command Prompt
2	Success of previous Usecase

S #	Test Data
1	SelectMyParent.exe
2	Spoof.log file

Test Scenario Verify whether the PPID spoofing attack was successfully detected or not?

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Execute Visual Studio with admin privileges	PPID and child process name can be entered	As Expected	Pass

7.2.3 Test Case - 3

Test Case ID	IR_003	Test Case Description	Alert generation on the Wazuh dashboard		
Created By	Afeefa	Reviewed By	Afeefa	Version	C.1

Tester's Name	Afeefa	Date Tested	May 12, 22	Test Case (Pass/Fail/Not Executed)	Pass
----------------------	--------	--------------------	------------	---	------

OA Tester's Log Refer demonstration heading

S #	Prerequisites:
1	Successful execution of Previous use case.
2	Installation of Wazuh manager and Wazuh agent

S #	Test Data
1	PPID = 21728
2	Child Process= Notepad

Test Scenario Verify whether the PPID spoofing attack alert was successfully generated on the Wazuh dashboard or not?

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Task manager access	Should have access	As Expected	Pass
2	Enter PPID & child process name	PPID and child process name can be entered	As Expected	Pass

7.2.4 Test Case - 4

Test Case ID	IR_004	Test Case Description	AR executed by Wazuh Manager		
Created By	Afeefa	Reviewed By	Afeefa	Version	D.1

OA Tester's Log Ar_test_result.log file on the endpoint

Tester's Name	Afeefa	Date Tested	May 12, 22	Test Case (Pass/Fail/Not Executed)	Pass
----------------------	--------	--------------------	------------	---	------

S #	Prerequisites:
1	Successful execution of Previous use case.
2	AR.exe on endpoint with execution permission

S #	Test Data
1	Alert on Wazuh Dashboard

Test Scenario Verify whether the AR was successfully launched or not?

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Run AR.exe on endpoint	Should have access	As Expected	Pass

7.2.5 Test Case - 5

Test Case ID	IR_005	Test Case Description	Playbook Execution by Shuffle tool		
Created By	Afeefa	Reviewed By	Afeefa	Version	E.1

QA Tester's Log Shuffle server logs

Tester's Name	Afeefa	Date Tested	May 12, 22	Test Case (Pass/Fail/Not Executed)	Pass
----------------------	--------	--------------------	------------	---	------

S #	Prerequisites:
1	Successful execution of Previous use case.
2	Shuffle installed and integrated with other tools

S #	Test Data
1	Alert on Wazuh, Captured by Shuffle Webhook

Test Scenario Verify whether the playbook/workflow was executed successfully or not?

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
--------	--------------	------------------	----------------	--

1	Execute playbook	Should have access	As Expected	Pass
---	------------------	--------------------	-------------	------

7.2.6 Test Case - 6

Test Case ID	IR_005	Test Case Description	Case creation on Thehive tool		
Created By	Afeefa	Reviewed By	Afeefa	Version	F.1

QA Tester's Log

TheHive server logfiles

Tester's Name	Afeefa	Date Tested	May 12, 22	Test Case (Pass/Fail/Not Executed)	Pass
----------------------	--------	--------------------	------------	---	------

S #	Prerequisites:
1	Successful execution of Previous use case.
2	TheHive installed and properly configured

S #	Test Data
1	Playbook input data to TheHive using API key

Test Scenario

Verify whether the case on TheHive was successfully created or not?

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Playbook executed	Should have access	As Expected	Pass

7.2.7 Test Case - 7

Test Case ID	IR_006	Test Case Description	Alert on Discord Community		
Created By	Afeefa	Reviewed By	Afeefa	Version	G.1

QA Tester's Log

Discord Community logs

Tester's Name	Afeefa	Date Tested	May 12, 22	Test Case (Pass/Fail/Not Executed)	Pass
----------------------	--------	--------------------	------------	---	------

S #	Prerequisites:
1	Successful execution of Previous use case.
2	Discord installed and configured

S #	Test Data
1	Playbook input data to Discord using Webhook API

Test Scenario

Verify whether the alert successfully transferred to Discord or not?

Steps #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Playbook executed	Should have access	As Expected	Pass

Chapter 8

Conclusion & Future Enhancements

Chapter 8: Conclusion & Future Enhancements

8.1 Achievements and Improvements

Through the project, the automated incident response and successful integrations of the platforms like Wazuh Agent with Wazuh manager, Shuffle with TheHive, Wazuh, and Discord. The effective and timely incident response is achieved successfully. The improvements would be done in the way that the tool's premium version would be used to achieve the promising results that would be less consuming and would be easy to automate, integrate and configure. Furthermore, in the project, the dedicated laptop with the fixed memory and RAM is utilized it would be better to use a server for the integrations and installations as the platform require the server space for efficient and smooth running otherwise the problems of machine corruption (in case of VMs) would be encountered.

8.2 Lessons Learnt

The organizations require an effective team and resources to build the automated incident response for any particular attack scenario. This project aimed to provide the automated incident response for the Usecase of PPID (parent process id spoofing) through the open-source platforms and tools i-e Wazuh, TheHive, and Shuffle. Therefore, it can be concluded that organizations can design their attack-specific workflows to generate automated response and alert notifications.

8.3 Future Enhancements/Recommendations

The future enhancements could be the duplication of the project using the premium version of the mentioned tools or using the paid tools to smooth the process of integration and orchestration. Moreover, the proper memory and storage requirements should be fulfilled to ensure and maximize the end benefit and smooth the process of development. Open source tools come with the issues of compatibility, usability, and lack of support that would result in chaos for an organization and business to manage the daily operations along with security needs and demand.

Appendices

Appendix A: Information / Promotional Material

In this appendix, the designed standee for the presentation and illustration are attached which shows the bird's eye view of the project.

A.1. Standee

Responseify
Investigating Parent Process ID Spoofing and Devising an Automated Incident Response

SECURITY ORCHESTRATION, AUTOMATION AND RESPONSE

"SOAR is defined as solutions that combine incident response, orchestration, automation and threat intelligence (TI) management capabilities in a single platform" *Gartner*

USECASE

PPID Spoofing Incident Response
PPID Spoofing is a technique used by attackers to launch programs with arbitrary parent processes to thwart detection and to perform attacks like Privilege Escalation and Execute Malicious Payloads

GOALS & OBJECTIVES

- Detect PPID Spoofing using *Krabs ETW*
- Creating Rule and Decoder in *Wazuh*
- Sending PPID Spoofing alerts to *Wazuh*
- Generating **Automated Active Response (AR)** with *Wazuh*
- Integrating *Shuffle* with *Wazuh*, *TheHive* and *Discord*
- Playbook** Implementation using *Shuffle*

TOOLS & TECHNOLOGIES

Krabs **wazuh.** **Shuffle** **TheHive** **Discord**

Krabs ETW **Wazuh** **Shuffle** **TheHive** **Discord**

BENEFITS

- Fully Automated
- Agile and Consistent Incident Response
- No Alert fatigue and Context Switching
- Improve overall Security Posture
- All tools are open-source, free and open for modification.

SUPERVISOR
DR. NAVEED ANWAR BHATTI

GROUP MEMBERS
AFEEFA AHMAD 181209
ASIMA TAHIR 181254

AIR UNIVERSITY ISLAMABAD
DEPARTMENT OF CYBER SECURITY
SECTOR E-9 (PAF COMPLEX)
PHONE: (051) 9262557
WEBSITE: WWW.AU.EDU.PK

Reference and Bibliography

- [1] P. Cichonski, T. Millar, T. Grance and K. Scarfone, Computer Security Incident Handling Guide, 2nd ed. National Institute of Standards and Technology, 2012
- [2] P.Kral, Incidents Handler's Handbook. SANS Institute, 2011
- [3] A. Chuvakin and A. Barros, Preparing Your Security Operations for Orchestration and Automation Tools. Gartner, 2018.
- [4] "NCSC - Incidents", Ncsc.govt.nz, 2022. [Online]. Available: <https://www.ncsc.govt.nz/incidents/>. [Accessed: 08- May- 2022].
- [5] P. Wragg, "Is It an Incident or a Breach? Defining the Difference", Insights.integrity360.com, 2021. [Online]. Available: <https://insights.integrity360.com/incident-or-breach>. [Accessed: 08- May- 2022].
- [6] [Online]. Available: <https://www.cyber.gov.au/acsc/view-all-content/advice/guidelines-cyber-security-incidents>. [Accessed: 08- May- 2022].
- [7] J. Creasey, Cyber Security Incident Response Guide Version 1. CREST.
- [8] "Incident Investigation: Definition Examples | Awake Security", Awake Security, 2022. [Online]. Available: <https://awakesecurity.com/glossary/incident-investigation/>. [Accessed: 15- May- 2022]
- [9] J. Hunt, GitLab Security Incident Response Guide. GitLab, 2022.
- [10] C. Labs, "Web Defacement Attack Response | Automated Incident Response | Cyware", Cyware Labs, 2022. [Online]. Available: <https://cyware.com/use-case/web-defacement-response-automation>. [Accessed: 15- May 2022].
- [11] How-to-Response-Against-Web-Security-Incident-signed.pdf. BENGKULUPROV. 2018.
- [12] WEBSITE DEFACTION. Canadian Center for Cybersecurity, 2020.
- [13] N. Labs, "The return of the spoof part 1: Parent process ID spoofing", the return of the spoof part 1: Parent process ID spoofing, 2022.
- [14] i. team, "Parent Process ID (PPID) spoofing", Parent Process ID (PPID) Spoofing.
- [15] "Access Token Manipulation: Parent PID Spoofing, Sub-technique T1134.004 - Enterprise | MITRE ATT&CCK®", Attack.mitre.org. [Online]. Available: <https://attack.mitre.org/techniques/T1134/004/>. [Accessed: 15- May- 2022].
- [16] N. Hyvärinen, "Threats Research", Detecting Parent PID Spoofing, 2018.
- [17] E. Security, Parent Process ID Spoofing. Elastic Search.
- [18] J. Hunt, Security Incident Response Team - SIRT. GitLab.
- [19] "TheHive Project", Thehive-project.org. [Online]. Available: <https://thehive-project.org/>. [Accessed: 15 May 2022].
- [20] "The Open Source SOAR for all purposes", Shuffler.io, 2022. [Online]. Available: <https://shuffler.io/>. [Accessed: 15- May- 2022]
- [21] "Cortex XSOAR Threat Intelligence Management", Palo Alto Networks. [Online]. Available: <https://www.paloaltonetworks.com/resources/datasheets/cortexxsoar-threat-intelligence-management>.
- [22] "Incident Response Automation Orchestration with SOAR | Exabeam", Exabeam. [Online]. Available: <https://www.exabeam.com/siem-guide/incident-response-andautomation/>. [Accessed: 15- May- 2022].
- [23] Dflabs.com.[Online].Available: <https://www.dflabs.com/wpcontent/uploads/2020/12/dflabssolutionbriefincmansoarforsocsv11.58.pdf>. [Accessed: 15 – May – 2022].
- [24] "Splunk SOAR | Splunk", Splunk. [Online]. Available: <https://www.splunk.com/enus/software/Splunk-security-orchestration-and-automation.html>. [Accessed: 15 – M ay – 2022].
- [25] Fortinet.com. [Online]. Available:

<https://www.fortinet.com/content/dam/fortinet/assets/datasheets/fortisoar.pdf>. [Accessed: 15- May- 2022].

[26] "Security Automation Orchestration Tool | InsightConnect | Rapid7", Rapid7. [Online]. Available:

<https://www.rapid7.com/products/insightconnect/>. [Accessed: 15- May- 2022].

[27] "WALKOFF", Nsacyber.github.io. [Online]. Available:

<https://nsacyber.github.io/WALKOFF/>. [Accessed: 15- May- 2022]. 6 VOLUME 4, 2016

Author et al.: Reviewing the Existing SOAR (Security Orchestration Automation and Incident Response) Solutions

[28] "Low-Code Security Automation SOAR Platform", Swimlane. [Online].

Available: <https://swimlane.com/>. [Accessed: 15- May- 2022]

[29] "Wazuh agent - Installation guide · Wazuh documentation", Documentation.wazuh.com. [Online].

Available: <https://documentation.wazuh.com/current/installation-guide/wazuh-agent/index.html>. [Accessed: 18- May- 2022].

[30] "Upgrade guide · Wazuh documentation", Documentation.wazuh.com. [Online]. Available:

<https://documentation.wazuh.com/current/upgrade-guide/index.html>. [Accessed: 18- May-2022].

[31] F. Typing and J. D, "Detecting Parent Process Spoofing using KrabsETW - Securehat", Blog.securehat.co.uk. [Online].

Available: <https://blog.securehat.co.uk/detection-experiments/detecting-parent-process-spoofing-using-krabsetw>. [Accessed: 18- May- 2022].