TRƯỜNG ĐẠI HỌC FPT

MINISTRY OF EDUCATION AND TRAINING

FPT UNIVERSITY

CAPSTONE PROJECT DOCUMENT

Develop the anomalies process detection system

IAP491-G1		
Team member	Nguyen Anh Tuan	SE62864
	Nguyen The Lam	SE63326
	Nguyen Quang Dam	SE05820
	Nguyen Quoc Anh	SE06070
	Phan Manh Truong	SE02605
Supervisor	Do Xuan Cho	
Capstone project code		

- Hanoi, August/2020 -

TABLE OF CONTENTS

ACKNOWLEDGMENT

To be able to complete this project, all members of the team were very dedicated and spent a lot of time. The project has received the help of many people. Especially, this topic received enthusiastic help from the instructor, Dr. Do Xuan Cho. Many times he spent his own time guiding our team, including Saturday, Sunday or late at night. Thank you very much.

Our team would like to thank everyone for supporting and helping the group during the past time. We would also like to thank the friends for helping us. They share their knowledge and experience about the process of doing capstone projects. That knowledge pushes us to complete this project.

We would like to thank all teachers in the university for asking to encourage the group's progress. Our team also thanks FPT University for creating the best conditions in terms of facilities and documents for us to complete this project.

Thank you to the family for always supporting us in the process of making this project.

Sincerely thank everyone.

ABSTRACTION

"There are two types of companies: those who have been hacked, and those who don't yet know they have been hacked." - Cisco CEO John Chambers.

Tracking and detecting anomalies to detect cyber attack is an important challenge for any information system, especially important information systems such as banking, e-commerce, military, or organizations. Detecting that the computer system is being attacked helps the organization know how it is being attacked, thereby preventing and protecting information.

The current intrusion detection systems are very diverse and are being developed by many security centers around the world. However, today's cyber attacks are becoming more and more sophisticated and easily bypass the aforementioned software. There are many systems with a new approach that are being researched. In this project we propose an attack detection solution based on event log analysis on Windows Operating System.

Analyze event log to detect anomalies in the system, thereby detecting and preventing attacks happening on that system. This approach is the trend recently. There are many cybersecurity companies out there that have products that use this approach.

In the limitation of the project, we introduce some methods of analyzing a large number of event logs. Examples include hash based analysis, based on IP and the domain that processes are connected to, behavioral analysis on the basis of Miter Att & ck. In which we apply machine learning to solve the problem of detecting malicious domains.

Finally, our team has successfully built a centralized detection system, consisting of many module services operating in parallel and independently. We have also built a web application to show the analysis results.

I. INTRODUCTION

I.1. Project Information

- Project name: Develop the anomalies process detection system
- Project name in Vietnamese: Xây dựng hệ thống phát hiện tiến trình bất thường.
- Timeline: 18/05/2020 - 31/08/2020

I.2. People

I.2.1. Supervisor

Full Name	E-mail	Title
Đỗ Xuân Chợ	Chodx@fe.edu.vn	Lecturer

Table I.1. List of Supervisors.

I.2.2. Members

No	Full Name	Student ID	E-mail	Role
1	Nguyễn Anh Tuấn	SE62864	TuanNASE62864@fpt.edu.vn	Leader
2	Nguyễn Thế Lâm	SE63326	LamNTSE63326@fpt.edu.vn	Member
3	Nguyễn Quốc Anh	SE06070	AnhNQSE06070@fpt.edu.vn	Member
4	Nguyễn Quang Đàm	SE05820	DamNQSE05820@fpt.edu.vn	Member
5	Phan Mạnh Trường	SE02605	TruongPMSE02605@fpt.edu.vn	Member

Table I.2. List of team members.

I.3. Background

Today, information technology has a very important role in enterprises. Many companies and organizations have already applied information technology in storing and processing information, providing online service for their customers. These computer systems and data storage are the most valuable things of these companies and organizations. Therefore, these information systems have become the target of many cyber attacks.

Protection against such attacks is vital for enterprises' information and data systems. Firewall systems, intrusion detection (IDS), and anti-malware systems are invested with large sums of money by enterprises. However, the biggest flaw lies in the users or internal employees of the company. It is very popular for hackers to attack employees' machines to prepare for an internal phishing attack, spread a virus, or escalate privileges in more valuable systems. Because this attack is simple and effective, it takes advantage of the weakest point of the information system. Many organized hacker groups regularly organize attacks of this type on agencies, businesses, and governments to steal information.

Throughout this project, we propose a solution that analyzes anomalies from the user's computer to warn possible attacks. Sources of information for analysis are provided very detailedly and complete thanks to Sysmon software. However, handling a large amount of event information that comes in from multiple machines is a very time-consuming undertaking. So we built this system to automate the detection and warning of unusual behavior of the processes under the user machine.

I.4. Then Initial Idea

As mentioned in the previous section, log collection and processing is a very heavy and time-consuming task. Handling becomes more difficult when there are so many machines to be monitored and monitored. On average, a computer can generate an average of 10 events per second. Each day there will be equivalent to 200,000 to 300,000 events to be handled per workstation.

A centralized processing system needs to be properly designed to handle large amounts of data. Our team had to research to find a solution to this problem. Many technologies to store and query data should be used as ElasticSearch, MongoDB, Redis.

The finished product will feature anomaly analysis and detection and synthesize the results. The software will be publicly available in open-source form, providing a new option to defend against targeted attacks.

I.5. A Brief Overview of The Current Anomaly Detection System In The World

I.5.1. Antivirus software

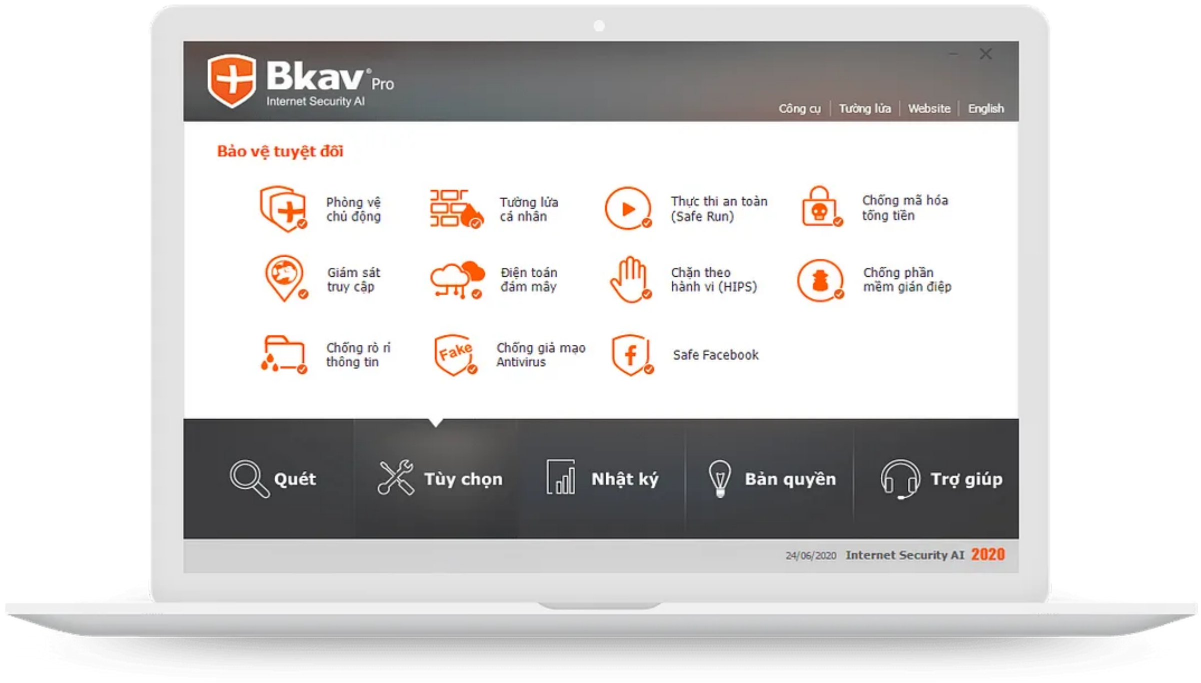


Figure I.1. A famous antivirus software. (source: bkavprovn.com)

Virus and malware detection and removal software are a popular and popular way to detect malicious programs. The general methods these types of software use are against a known malware database. Also, the analysis methods on the behavior of malware or real-time monitoring are applied in major anti-virus software.

There are many famous anti-virus software such as BKAV, Avast, Kaspersky, etc. Currently, most businesses and agencies are implementing integrated anti-virus software for employees' computers.

I.5.2. Splunk

Splunk is software that allows professionals to search and browse logs and IT data in real-time. Users can immediately discover problems in any application, or in servers and devices; warns of potential threats and reports the activities of different network components and services. And here is the troubleshoot solution for the system as well. The downside of this type of software is that it is only suitable for experts with knowledge of information security monitoring.

I.5.3. Kaspersky EDR

Kaspersky EDR is tightly coupled with the Kaspersky ATA(Kaspersky Anti - Targeted Attack Platform) platform to protect against attacks. Besides, Kaspersky EDR enables control of network endpoints, detects malicious code and unknown unauthorized behavior at the level of network protection, and reacts quickly to them. Kaspersky EDR can continuously monitor anomalies, suspicious processes on employee workstations, and react to threats in manual mode, as well as in passive mode[1].

I.5.4. Apex One

Apex One is a solution that combines endpoint feedback and development tools with other automatic detection and response functions. This helps users avoid depending too much on manual operations. Comes with many outstanding features that will automatically detect and block as many end-point threats as possible without any manual intervention from the user, like advanced learning-based threat detection. intelligent machine and IOA behavior analysis; Detect and prevent exploitation of vulnerabilities in the operating system before threats reach the endpoint with virtual patches always updated continuously[2].

I.5.5. Palo Alto Networks Traps

Palo Alto Networks Traps is an advanced endpoint protection solution that helps prevent endpoint threats and coordinates with cloud and network security to prevent network attacks. Traps minimize infection of endpoints by blocking malware, exploits, and ransomware. Integrates with your security platform, providing additional threat analysis, information sharing, and automatic blocking[3].

I.6. The Proposed Idea

From analyzing the urgency of real needs as well as based on assessments of existing technologies and techniques, we will build a system to collect and monitor abnormal processes on workstations based on ruleset and behavioral analysis. To solve the set goals, we carry out the following tasks:

- Build a progress collection system from workstations. In this project, we use the Sysmon tool to collect client event log processes.
- Build a process analysis and evaluation system to detect abnormal and malicious processes. To ensure accurate detection results as well as detection time, in this project, we try to combine two detection methods based on the analysis of abnormal behavior and known malware samples combined with machine learning. Accordingly, for the rule-set detection

method we compare the processes with known attack markers such as has, domain, etc, we also analyze the processes behavior based on Mitre Att&ck knowledge. As for the detection method using machine learning, we focus on analyzing the connected domains of these processes.

- Building a management system and warning of unusual processes including process management and storage module, warning web app.

II. IA PROJECT MANAGEMENT PLAN

II.1. Problem Setting

II.1.1. Name Of The Capstone Project

Develop the anomalies process detection system

II.1.2. Problem Abstraction

In targeted attacks, the hacker's goal is to break into an organization's information systems to steal valuable information or to encrypt and destroy that important information. The information hackers target is usually user information, transaction history, intellectual products, technology secrets, business secrets, business strategies.

People are the weakest point of the system. Therefore, hackers often take advantage of a staff loophole in the organization to initiate a plan to break into these organization computer networks. The methods that target people often include exploiting weak passwords, software with unsecured vulnerabilities, or phishing attacks.

Therefore, it is extremely important to monitor the employee's computer in the enterprise or organization to detect the attack. This helps businesses and organizations to detect impending attacks early and prevent information theft, and protect businesses and agencies from major losses.

Analysis anomalies based on event log data of processes is very potential. This method helps to detect attacks early. Because all the behavior of processes in a computer is recorded in every detail by the operating system. However, processing for many machines with a very large amount of information is a difficult problem. In this project, we propose a solution to that problem.

The goal of this project is to build a centralized system to receive data about the events of the processes. This centralized system will store all event log data to the database and detect if there is any anomaly process. From there, giving warnings about the threat of attacks of the system.

II.2. Project Organization

II.2.1. Agile process model

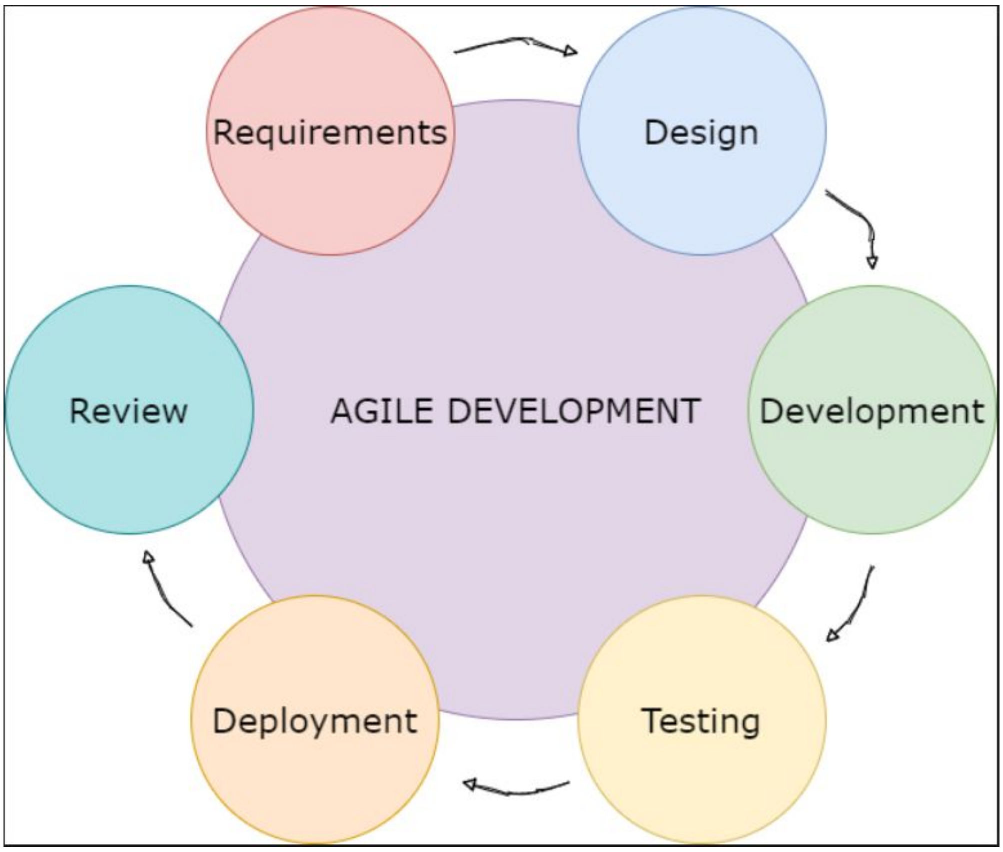


Figure II.1. Agile process.

Introduce

Agile is a flexible software development methodology, which is a specific approach to software project management. It includes an integrated and interactive workflow to get products into the hands of users as quickly as possible.

The meaning of Agile is fast or versatile. "Agile Process Model" refers to an iterative development-based software development approach. Agile methods divide tasks into smaller iterations or parts that are not directly related to long-term planning. The scope and requirements of the project are set at the start of the development process. Plans for the number of iterations, duration, and scope of each iteration are predefined[4].

Each iteration is considered a short time "frame" in the Agile process model, usually lasting one to four weeks. Dividing the entire project into smaller chunks helps to minimize project risk and reduce overall project delivery time requirements. Each iteration involves a team working through the full software development lifecycle including planning, requirement analysis, design, coding, and testing before the working product is demonstrated for the customer.

In 2001, the Agile Manifesto (Agile Manifesto) was unified and released by a group of people with an authority in software development:

- **Individuals and interactions over processes and tools**: Individuals and interactions rather than processes and tools
- **Working software over comprehensive documentation**: Software that runs better is well documented
- **Customer collaboration over contract negotiation**: Collaborate with the customer rather than negotiate a contract
- **Responding to change over following a plan**: Respond to change rather than stick to a plan

Advantage of agile

- **Making changes easy**: Because a project is broken down into small, separate, non-interdependent parts, changes are made very easily, at any stage of the project.
- **There is no need to know all the information at first**: Suitable for projects where there are no clear end goals, as this is not so necessary in the early stages.
- **Faster handover**: Breaking up projects allows the team to perform part inspections, identify and fix problems faster. Therefore the handover will be more consistent and successful.
- **Continuous improvemen**t: Communication between the team and the customer is encouraged. During development, it is possible to improve the product as many times as needed.

Because of these advantages, our team decided to choose Agile to make the development process more adapted to new changes and make the whole process more efficient and independent of each other.

II.2.2. Role And Responsibilities

Phases	Name	Responsibilities

Requirement Gathering	Nguyễn Anh Tuấn Nguyễn Thế Lâm Nguyễn Quang Đàm	- Contact supervisors. - Identify the goals and objectives of the project. - Assigning tasks to each person. - Research about solution process models.
	Nguyễn Quốc Anh Phan Mạnh Trường	- Research web technologies.
Planning & Designing	Nguyễn Anh Tuấn	- Research and determine which technologies will be used. - Research and design the logical model of the project.
	Nguyễn Thế Lâm	- Research ElasticSearch and Kibana
	Nguyễn Quang Đàm	- Research MongoDB, Python Flask, Redis. - Research Machine Learning.
	Nguyễn Quốc Anh Phan Mạnh Trường	- Research React and NodeJs.
Development and Deployment	Nguyễn Anh Tuấn	- Tutorial and help other members about technologies and how to apply them to this project.
	Nguyễn Thế Lâm	- Install the coding environment and database. - Develop anomaly behavior detection services.
	Nguyễn Quang Đàm	- Setup Ubuntu Server - Develop anomaly detection systems based on hash and domain. - Applied machine learning.

	Nguyễn Quốc Anh Phan Mạnh Trường	- Develop web services to show the result of detection systems.
Testing	Nguyễn Anh Tuấn Nguyễn Thế Lâm Nguyễn Quang Đàm Nguyễn Quốc Anh Phan Mạnh Trường	- Test and debug and deploy to make all services work properly.
Evaluation and report	Nguyễn Anh Tuấn	- Test and review the final product. - Writing outline, assigned task for members. - Complete document.
	Nguyễn Thế Lâm Nguyễn Quang Đàm Nguyễn Quốc Anh Phan Mạnh Trường	- Collect information and summarize the document. - Capture image of front-end feature.

Table II.1. Team members and respective responsibilities.

II.2.3. Tool and technologies

Tools and techniques		
Development	VSCode	IDE for Python 3, HTML, css, js, jQuery
	Github	Source code version control

	Sublime Text	Sublime Text is a sophisticated text editor for code, markup, and prose
	Ubuntu 16.04	Operating System for development and coding
	ElasticSearch	Elasticsearch is a search engine based on the Lucene library. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents.
	Trello	Keep tracking of work
	Sysmon	Windows service that monitors and logs system activities.
	Chrome	Web browser, testing environment
	Google Cloud	Setting server Ubuntu
	Kibana	Kibana is an open-source analytics and direct management platform designed to operate closely in conjunction with Elasticsearch. Offers powerful features and ease of use such as charts, line charts,

		histograms, thermography, and non-geographic support.
	Redis	Redis is an open-source (BSD licensed), in-memory data structure store, used as a database, cache, and message broker.
	MongoDB	MongoDB is a document database, it stores data in JSON document format.
	React	React is the most popular JavaScript library for building user interfaces (UI). It gives excellent response speed when users input data using the new method of web page rendering.
Programming Languages	Python 3	Python is an interpreted, high-level, general-purpose programming language.
	HTML	HTML is the standard markup language for creating Web pages. HTML stands for HyperText Markup.
	CSS	CSS is a language that describes the style of an HTML document. CSS describes how HTML elements should be displayed.
	JavaScript	JavaScript, often abbreviated as JS, is a high-level, interpreted programming language that conforms to the ECMAScript specification. JavaScript has curly-bracket syntax, dynamic typing,

		prototype-based object-orientation, and first-class functions
Communication	Skype	Group video calls, chat, file sharing. Contact supervisor.
	Facebook messenger	Group video calls, chat, file sharing
	Mobile phone	Instant call
	Google Meet	Real-time meetings by Google. Using your browser, share your video, desktop, and presentations with teammates and customers.
	Gmail	Share link resources and links between only team members.
Office tools	Google Sheet	Manage and distribute project plans for each team member.
	Google Slide	Present and report work progress.

	Google Docs	Writing documents and reports for the project.
Python Library	Flask	Flask is a lightweight WSGI (Web Server Gateway Interface) web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications.
	Beautiful Soup	Beautiful Soup is a Python library for pulling data out of HTML and XML files

Table II.2. Tools and techniques used in the project.

II.3. Project Management Plan

II.3.1. Task

II.3.1.1. Project initialization, planning

- Description: Research about the problem, think about the solution. Determine technology to solve the problem. Create a plan and assign roles and responsibilities for team members.
- Deliverables: After planning and research, we decide to build a system to analyst process behavior automatically.
- Resources: Internet document, research paper.
- Risk: Think about incorrect or very complex solutions.

II.3.1.2. Technical studies

- Description: Research about Redis, MongoDB, ElasticSearch, Python Flask, Kibana, Sysmon, Winlogbeat, Logstash, Mitre Att&ck, React, and Sysmon.
- Deliverables: Understand the advantage of these technologies and use them correctly.
- Resources: Online tutorials.
- Risk: Lack of knowledge.

II.3.1.3. Design and analysis project

- Description: Design the flow of the application. Design the database schema. Determine the feature for applying machine learning in the malicious domain.
- Deliverables: The overview architecture of the application, database schema and list of machine learning features.
- Resources: Internet tutorial and guide.
- Risk: Lack of knowledge, time to research from the beginning.

II.3.1.4. Implementation

- Description: Implement all the module service, design and develop the UI of web applications.
- Resources: Internet tutorial and guide.
- Risk: Lack of knowledge, misunderstanding and mistakes when coding.

II.3.1.5. Testing and Fix Bug

- Description: Every one creates unit tests for their own module. When testing, try to optimize the application to make it run faster.
- Deliverables: The program runs correctly and is stable.
- Resources: Internet tutorials about the unit test.
- Risk: Lack of knowledge, misunderstanding and mistakes when coding.

II.3.1.6. Deployment

- Description: Register a free Google Cloud VPS to deploy all modules. Config to outside the network can access them.
- Deliverables: Successfully deploy stable applications.
- Resources: One year free subscription Google Cloud. Tutorial about Google Cloud VPS and deployment.
- Risk: Security, mis-config.

II.3.1.7. Document development

- Description: Write documentation about all processes to make this project.
- Deliverables: Full detail document.

- Resources: Google docs, Grammarly, Google translate.
- Risk: disconnecting the internet because we use online Google docs.

II.3.2. Task schedule

Task	Subtask	Detail	Assignment	Time
Start project: 18/05/2020				
Project initialization	Discuss about the idea.	All team members discuss the problem and discuss solutions. We have spent a lot of time on this step.	All team members.	18/05/2020 - 30/05/2020
	Discuss about the technology will be used.	We consider what are the best technologies that fit with our project. We also consider it easy to learn technologies first.		
Technical studies	Programming language and framework.	Python, Python Flask, NodeJs, React.	All team member	01/06/2020 - 20/06/2020
	Technology	Redis, MongoDB, ElasticSearch, Kibana, Winlogbeat, Logstash, Sysmon.		
Design and analysis project	Design Diagram	Logical Flow Diagram.	TuanNA, LamNT, DamNQ	22/06/2020 - 30/06/2020
		Database schema.		
	Web design	Web application design.	AnhNQ, TruongPM	
Implementation	Install framework and tool for programming.	Redis, MongoDB, ElasticSearch, Kibana, Winlogbeat, Logstash, Sysmon.	TuanNA, LamNT, DamNQ	01/07/2020 - 31/07/2020
	Implement module services.	Hash service: for check blacklist sha1 process.	DamNQ, TuanNA	

		Domain service: for check domain in blacklist.	DamNQ	
		Mitre service: for check behavior in sigma rule.	LamNT, TuanNA	
		Domain ML service: to check malicious domains based on ML.	DamNQ	
		Web application.	AnhNQ, TruongPM	
Testing	Write unit test	Mitre services, Hash service;	LamNT, TuanNA	01/08/2020 - 07/08/2020
	Manual testing	Front end web app, domain service.	DamNQ, TruongPM, AnhNQ	
	Review domain machine learning results.	Domain ML service;	DamNQ	
Deployment	Deploy all service and Web applications to Google Cloud VPS server.	Install environment	LamNT, TuanNA, DamNQ,	10/08/2020 - 17/08/2020
		Deploy services	TuanNA, LamNT, DamNQ	
		Deploy web application	TuanNA, AnhNQ.	
Writing documents	Introduction part		TuanNA, TruongPM	18/08/2020 - 28/08/2020
	IA project management part		TuanNA, TruongPM	
	Risk Assessment		TuanNA TruongPM	

	Risk Management plan		TuanNA TruongPM	
	Specifications, development and implementation plan		TuanNA, LamNT, DamNQ,	
	Project Validation		TuanNA	
	Table of content, list of figures, acknowledgement, abstraction.		TuanNA, LamNT	
	Document format		LamNT, TuanNA	
	Slide		TuanNA, LamNT	

Table II.3. Task schedule of the group.

II.3.3. All Meeting Minutes

Date time	**Subject**	**Location**	**Attendees**	**Meeting topic**
18/05 13h to 14h30	Planning meeting	AL-L305, FPT University	Supervisor: ChoDX. All team members.	- Introduce team members. - Discuss about project topic and project name. - Assign research task
25/05 13h to 14h30		AL-L305, FPT University	Supervisor: ChoDX. All team members.	- Members recommend ideas for the project and discuss the technology that will be used.

01/06 13h to 14h30	Weekly meeting: Technical studies	AL-L305, FPT University	Supervisor: ChoDX. All team members.	- Report research result. - Continue to discuss ideas.
08/06 13h to 14h30		AL-L305, FPT University	Supervisor: ChoDX. All team members.	- Discuss about the current anomaly detection system in the world and learn from them.
15/06 13h to 14h30		AL-L305, FPT University	Supervisor: ChoDX. All team members.	- Assign tasks for team members to research programming language, framework and technologies.
22/06 13h to 14h30		AL-R301, FPT University	Supervisor: ChoDX. All team members.	- Report completed task. - Assign new tasks for next week.
29/06 20h to 22h	Weekly meeting: Design and analysis project	Google Meet	Supervisor: ChoDX. All team members.	Discuss about: - Design the flow of all service modules - Design the database schema. - Design web application.
06/07 13h to 14h30	Weekly meeting: Implementat ion	AL-L305, FPT University	Supervisor: ChoDX. All team members.	- Report completed task. - Assign new tasks for next week.
11/07 13h to 14h30		AL-L305, FPT University	Supervisor: ChoDX. All team members.	- Report completed task. - Assign new tasks for next week.

13/07 13h to 14h30		AL-L305, FPT University	Supervisor: ChoDX. All team members.	- Report completed task. - Assign new tasks for next week.
20/07 13h to 14h30		AL-L501, FPT University	Supervisor: ChoDX. All team members.	- Report completed task. - Assign new tasks for next week.
27/07 13h to 14h30		AL-L305, FPT University	Supervisor: ChoDX. All team members.	- Report completed task. - Final review all completed module services.
03/08 20h to 22h	Weekly meeting: Testing	Google Meet	Supervisor: ChoDX. All team members.	- Discuss about the testing process. - Assign new tasks for team members.
10/08 08h to 09h30	Weekly meeting: Deployment	Google Meet	Supervisor: ChoDX. All team members.	- Review completed tasks. - Assign new tasks.
17/08 20h to 22h		Google Meet	Supervisor: ChoDX. All team members.	- Report deploy task. - Assign documents writing tasks.
24/08 20h to 22h	Weekly meeting: Writing documents	Google Meet	Supervisor: ChoDX. All team members.	- Review completed tasks. - Assign new tasks.
31/08 20h to 22h		Google Meet	Supervisor: ChoDX.	- Report work progress and create slides.

			All team members.	- Ready for thesis defense.

Table II.4. Meeting minutes.

III. RISK ASSESSMENT

III.1. The Need of Risk Assessment

Risk assessment is used as a term to describe the process of assessing, identifying risks as well as analyzing and assessing potential threats along with risk. Risk assessments also offer solutions to eliminate risks in the process or control them. When developing a secure information plan for a system, risk assessment is an integral part and plays an important role as it provides a general overview and full awareness of systematic risks. It also helps to identify where the risks come from.[5]

III.2. Identify Critical Information Assets

III.2.1. Information Asset Classification

CIA (confidentiality, integrity, availability) is the goal of the Information System Security[6]. We need to classify Information Assets based on the level of sensitivity and the impact to our system when that information is disclosed, altered, or destroyed without authorisation. It helps us determine what baseline security controls are appropriate to protect that information.

Information class	Description	Assets	Confidentiality level
Highly Restricted	The most private and sensitive information. Should be strictly monitored and controlled all of the time.	- Elasticsearch Database - MongoDB - Redis Server - Source code	- Critical
Confidential	If it was leaked, the system will be accessed or altered without authorisation.	- Elastic search credential. - MongoDB connection string. - Team members github credential. - VPS Server ssh credential.	- High
Internal use only	The information that is only used	- Database schema design.	- Medium

	internally by the team.	- Web application interface design. - All the related diagrams.	
Public	The information that is released to the public without harm to our project.	- Project name. - Project topic. - Name of the members.	- Low

Table III.1. Information assets of our project and priorities.

III.2.2. System Characterization

III.2.2.1. Logical Architecture

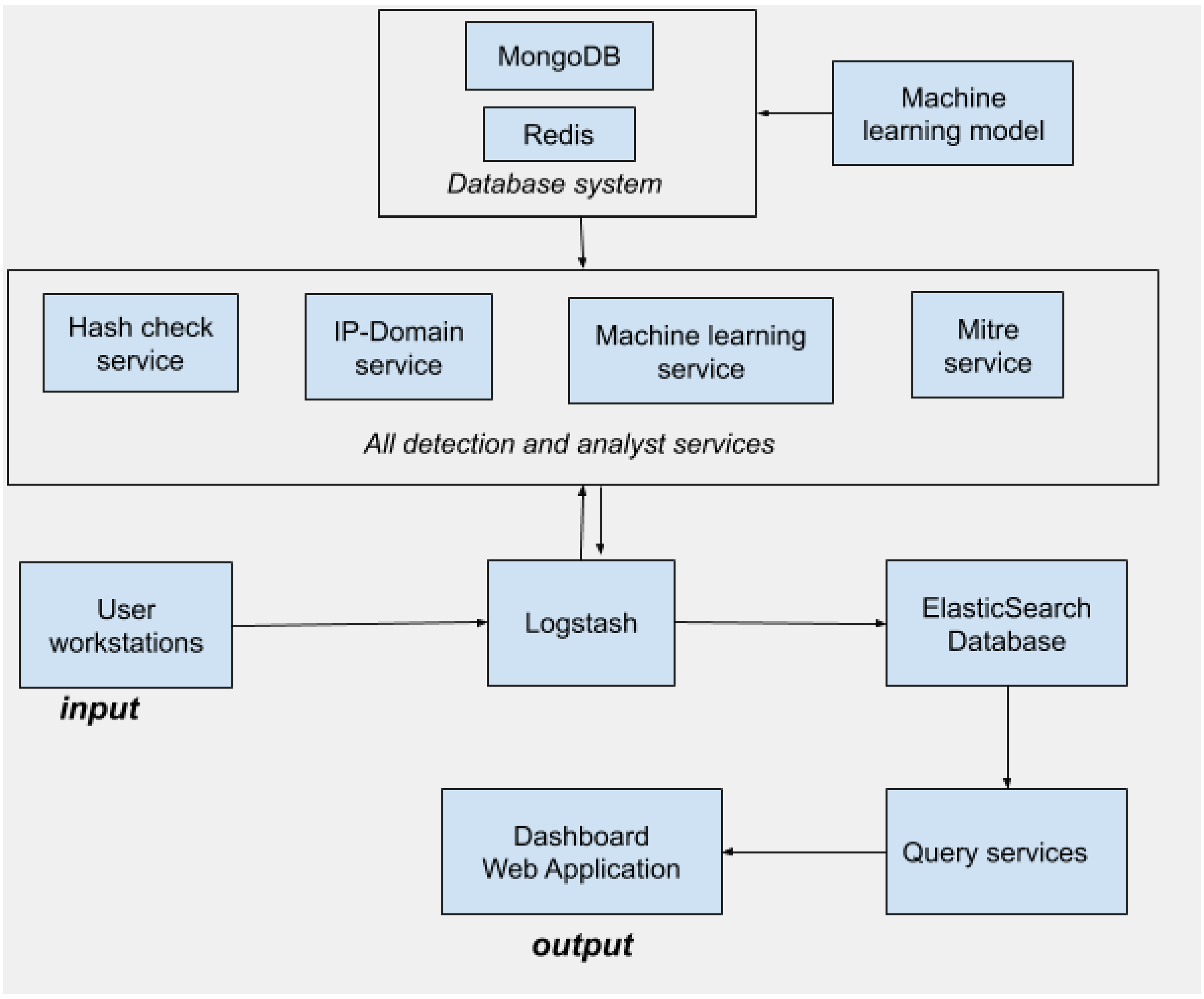


Figure III.1. System model diagram.

III.2.2.2. System Components

No.	**Name**	**Description**
1	Users workstation	All workstations we need to monitor and analyst.
2	Server	Google Cloud VPS Server for deploying all module services.
3	Redis server	Redis server runs on Google Cloud Server.
4	MongoDB	MongoDB server stores data for known malicious hash, domain and IP.
5	Machine learning model	Trained model used to detect malicious domains.
6	Hash service	A service checks if a hash is in the form SHA1 already known in MongoDB.
7	IP-Domain service	A service checks if a domain or IP is already known in MongoDB.
8	Machine learning service	This is a service to load machine learning models and predict malicious domains.
9	Mitre service	For analyst anomaly behaviour using rules based on Mitre Att&ck.
10	Logstash	This is an important module to format raw event log data and merge it with the results returned from all analyst services.
11	Elasticsearch database	All results returned from logstash will be stored in Elasticsearch, ready to show this data on the dashboard.
12	Query anomaly service	This service is used to query data from Elasticsearch.
13	Web application dashboard	A dashboard call API provided by Query service and show data on dashboard.

Table III.2. System components.

III.2.2.3. Users of the System

No.	**Name**	**Role**
1	System Admin	Full access to all server and database.

		Have the right to install and setup all services. Responsibility to deploy all services and databases.
2	Administrator	Full access to read and modify all data in the database.
3	Viewer	Full access to read the dashboard.

Table III.3. Users of the System.

III.2.2.4. Security and Compliance Requirements

To ensure system security all restricted data and system need to comply with the following security requirements:

- Regular update security patch.
- Only use software or tools from trusted sources.
- Secure stores all important information about credentials.
- Protect all source code and private information about the project.

III.2.2.5. Information Protection Priorities

No.	**Name of information**	**Priority**
1	Database data	1
2	Source code	1
3	All login credential	1
4	SSH private key	1
5	System design and database schema	2
6	Public documentation	3
7	Public project information	3

Table III.4. Information protection priority.

III.3. Risk Identification

The following formula is often used when pairing threats with vulnerabilities [5][7]:

Risk = Threat * Vulnerability

Risk	Threats	Vulnerabilities
business disruption	angry employees	software bugs
financial losses	dishonest employees	broken processes
loss of privacy	criminals	ineffective controls
damage to reputation	governments	hardware flaws
loss of confidence	terrorists	business change
legal penalties	the press	legacy systems
impaired growth	competitors	inadequate BCP
loss of life	hackers	human error
	nature	

*Table III.5. Risk = Threats * Vulnerabilities.*

Threat and vulnerability don't always have numerical values. Instead, the formula shows the relationship between the two.

If we can identify the value of the asset, the formula is slightly modified to:

Total Risk = Threat * Vulnerability * Asset

III.3.1. Threat Identification

In Figure III.3, threats can be categorized into two types: Human or Natural. Human threats can also be internal (angry employees or team members, dishonest employees…) or external (hackers, competitors …).

Some natural threats such as power outages, natural disasters can cause the server system to stop working.

	Reason
Internal	- Threats come from people who are dissatisfied with the group, such as team members. They may collect data and reveal confidential, unwanted information about the project. - Or it may be due to unknown team members.
External	External threats such as - Hackers want to attack to gather information or other data or resources. - Competitors want to get secret data.
Natural	- Unusual reasons such as natural disasters, earthquakes, floods, storms, tsunamis … - Or unusual incidents such as power outages, life outages, explosions, epidemics ...

Table III.6. List of threats to the project.

III.3.2. Vulnerability Identification

Vulnerability is meant for threatening factors to cause damage to individuals and organizations.

The table below shows the vulnerabilities that may be encountered in each section:

Service	**Vulnerability**	**Description**	**Impact**
Data storage and database.	Misconfiguration	Unsafe configurations, usually are unintended open ports or unnecessary services running.	- This can reduce system efficiency or be exploited by hackers. - For example, when the leader Tuan configures the network and accidentally leaves some ports open to access to open-web applications, the attackers could exploit the vulnerability to take control

			of the system or to perform a bigger attack.
	Information disclosure	Intentional or Accidental, credential information for the system access could be leaked. Unauthorized users can acquire confidential data or make major changes to the system.	Information disclosure allows hackers to access confidential system information that jeopardizes the project.
Dashboard web application	Weak password	Weak passwords can help an attacker easily gain access to the system.	Then the hacker will take control of the system and get all secret data.
	Insecure coding makes the application have some bug that can lead to an attack.	Web applications can has some security bug such as XSS, Injection code.	Web applications can be hacked and all important data would leak.
Hardware and software	Outdated software	Failure to update to new versions of the application can lead to errors, even security holes from older versions.	Affect the progress of the project.
	Physical assets damaged	Personal computers and data storage devices may be damaged or lost.	Interfering with the progress of the project because lost data may be in the device.
Team members	Cancels offline meeting	Diseases, accidents, and personal work cause limited meetings which can affect work performance.	Affect the progress of the project.
	Conflicts between team members	Unresolved conflicts and disagreements about	Affect the progress of the project.

		individual opinions in the group.	

Table III.7. Vulnerabilities on the project.

IV. RISK MANAGEMENT PLAN

IV.1. Objectives of RMP

IV.1.1. Lists of Threats & Vulnerabilities

No.	Type	Threats & Vulnerabilities
1	Data storage and database.	Misconfiguration
		Information disclosure
2	Dashboard web application	Weak password
		Web applications vulnerable like XSS, Code Injection.
3	Hardware and software	Outdated software
		Computer corruption
4	Team members	Cancels offline meeting
		Conflicts between team members
		Conflicts idea.

Table IV.1. Lists of Threats & Vulnerabilities.

IV.1.2. Costs associated with risks

Confidentiality

Rate	Description	Point
Low	No affect or effect a little.	1
Medium	The system can accept the risk.	2
High	Seriously affected.	3

Table IV.2. Assessment of security risks.

Integrity

Rate	Description	Point
Low	No effect.	1
Medium	The system can accept the risk.	2
High	Seriously affected.	3

Table IV.3. Assess integrity risk.

Availability

Rate	Description	Point
Low	No affect or effect a little.	1
Medium	The system can accept the risk.	2
High	Seriously affected.	3

Table IV.4. Risk assessment of availability.

Below is the Threat and Vulnerabilities Rate table:

No	Type	Threats and Vulnerabilities	Confidentiality	Integrity	Availability	Total	Rate
1	Data storage and database.	Misconfiguration	2	2	2	6	High
		Information disclosure	3	3	3	9	Critical
2	Dashboard web application	Weak password	3	1	2	6	High
		Security bug	3	3	3	9	High
3	Hardware and software	Outdated software	2	1	2	5	High
		Physical assets damaged	1	1	1	3	Medium
4	Team members	Cancels offline meeting	1	2	2	5	High
		Conflicts between team members	1	2	1	4	High

Table IV.5. Threat and Vulnerability Rates Table.

IV.1.3. List of Recommendations to Reduce the Risks

No	Type	Threats & Vulnerabilities	Recommendation
1	Data storage and database	Misconfiguration	Review after changes
		Information disclosure	Training members about data confidentiality
2	Dashboard web application	Weak password	Use strong password
		Security bug	Scan and patch bugs.
3	Hardware and software	Outdated software	Regularly update new applications
		Physical assets damaged	Back up multiple devices and protect them carefully.
4	Team members	Cancels offline meeting	Streamline meetings reasonably.
		Conflicts between team members	When a disagreement occurs, the team leader is the decider.

Table IV.6. List of recommended mitigation actions.

IV.2. Assigning Responsibilities

Full Name	Role	Description
Nguyễn Anh Tuấn	Leader	- Resolve conflicts - Organize online meeting, - Assign tasks for members and fix vulnerabilities.
Nguyễn Thế Lâm	Member	- Research and collect information about risk. - Fix vulnerabilities. - Secure server configuration.
Nguyễn Quốc Anh	Member	- Secure dashboard application vulnerabilities. - Writing a risk management report.
Nguyễn Quang Đàm	Member	- Research and collect information about risk. - Writing a risk management report.
Phan Mạnh Trường	Member	- Backup data gradually - Review progression

Table IV.7. Responsibilities of each member.

IV.3. Describing Procedures and Schedules for Accomplishment

- Training members about data confidentiality.
- Create online meetings and connect team members.
- Create a backup for server and database.
- Check for regular operating system and software updates.
- Update firewall policies.

IV.4. Reporting Requirements

IV.4.1. Document Management Respond to Recommendation.

Recommendation	Accept	Decline	Modify
Review after changes	x		
Training member about data confidentiality	x		
Use strong password and data policy	x		
Regularly update new applications and OS	x		
Make backups periodically to guarantee availability of data	x		
Scan the web application for vulnerability and fix it.	x		

Table IV.8. Document Management Respond to Recommendation.

IV.4.2. Document and Track Implementation of Accepted

Recommendation	Status
Review after changes	Deployed
Training member about data confidentiality	Deployed
Use strong password and data policy	Deployed
Regularly update new applications and OS	Deployed
Make backups periodically to guarantee availability of data	Deployed

Scan the web application for vulnerability and fix it.	Deployed

Table IV.9. Document and Track Implementation of Accepted.

V. SPECIFICATIONS, DEVELOPMENT AND IMPLEMENTATION PLAN

V.1. Overview Of Architectural

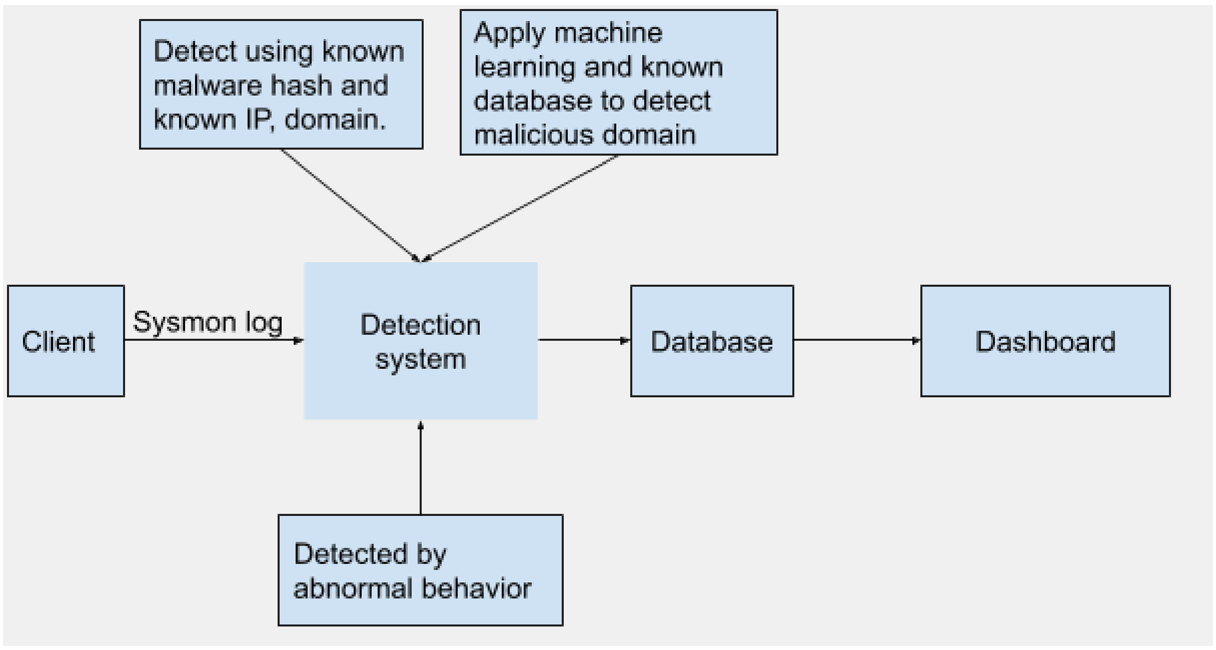

Detect using known malware hash and known IP, domain.
Apply machine learning and known database to detect malicious domain
Client
Sysmon log
Detection system
Database
Dashboard
Detected by abnormal behavior

Figure V.1. Overview of Architectural.

In general, this system has three main part:

The log data collector and delivery system have the responsibility to read log events that have been generated by Sysmon and deliver them to the central server for processing and detecting anomaly.

Detection systems and databases are used to anomaly process based on many factors: process behavior, file hash, and connected URL. This system has been developed by applying the microservice system design. It helps these features run independently and it's easy for our team to design and coding. The result of the detection system will be stored in the ElasticSearch database for further analysis and show the result to administrators.

Dashboard System is a web application to provide visualization data and result to the admin.

V.2. The Client-side Data Collection System

V.2.1. Introduction Sysmon

System Monitor (Sysmon) is a monitoring tool provided by Microsoft Windows[8]. Sysmon works as a service on Windows, so it can automatically start after the Windows Operating System boots successfully. It can provide very detailed information about network connections, process creation, file writing, and deleting, etc. There are 23 types of events that Sysmon can collect.

By collecting the events it generates we can detect the anomalous behavior, detect advanced threats on your network. Compared with other antivirus software or intrusion detection systems, Sysmon provides deep system monitoring and log high confidential indicators of advanced attacks.

List of all event that Sysmon support logging and monitoring:

ID	**Name**	**Description**
1	Process creation	Provide information about a newly created process. It contains a full command line that provides context on the process execution. Each process has a unique GUID to make event correlation easier.
2	A process changed a file creation time	This event helps to track the real creation time for a file.
3	Network connection	It logs both TCP and UDP connections from all processes on the machine. The events also contain IP and port of the source and destination connection. This event is disabled by default.
4	Sysmon service state changed	This event notifies when Sysmon changed its state (start or stop).
5	Process terminated	It provides information about when and which process terminated.
6	Driver loaded	This event provides information about a driver being loaded on the system.
7	Image loaded	This event logs when a module is loaded in a specific process. This event is disabled by default.
8	Create Remote Thread	This event detects when a process creates a thread in another process.

9	Raw Access Read	This type of event detects when a process conducts reading operations from the drive using the \\.\ denotation.
10	Process Access	When a process opens another process, an operation that's often followed by information queries or reading and writing the address space of the target process, Symon will log this event.
11	File Create	These events are logged when a file is created or overwritten.
12	RegistryEvent: Object create and delete	It can be useful for monitoring for changes to Registry autostart locations or specific malware registry modifications.
13	RegistryEvent: Value Set	This event records when a value has been written to the registry.
14	RegistryEvent: Key and Value Rename	This event records when a key or value of a registry has been changed.
15	FileCreateStreamHash	This event logs when a named file stream is created.
17	PipeEvent: Pipe Created	This event generates when a named pipe is created.
18	PipeEvent: Pipe Connected	This event logs when a named pipe connection is made between a client and a server.
19	WmiEvent: WmiEventFilter activity detected	This event logs when a WMI event filter is registered, including namespace, filter name, and filter expression.
20	WmiEvent: WmiEventConsumer activity detected	This event logs the registration of WMI consumers, recording the consumer name, log, and destination.
21	WmiEvent: WmiEventConsumerToFilter activity detected	This event logs the consumer name and filter path when a consumer binds to a filter.
22	DNSEvent	This event is generated when a process executes a DNS query.
23	FileDelete: A file delete was detected	This event is generated when a file is deleted.
255	Error	This event is generated when an error occurs within Sysmon.

Table V.1. All sysmon events.

V.2.2. Setup and configuration

Download Sysmon in the homepage of Microsoft (version for windows)

Uses Sysmon simple command-line options to install and uninstall it, as well as to check and modify Sysmon's configuration, run with administrator rights.

Installation

```
sysmon.exe -accepteula -i [configuration-file.xml]
```

Set new configuration:

```
sysmon.exe -c [update-configuration-file.xml]
```

Configuration files can be specified after switching between -i (install) or -c (update) configuration. They make it easy to configure presets and filter captured events. The configuration file contains the schema version properties on the Sysmon tag. This version is independent of the Sysmon binary version and allows the parsing of older configuration files. Can load the current schema version using the "-? command-line config ". Configuration entries are located just below the Sysmon tab and the filters are in the EventFiltering tab.

Configuration Entries

Configuration entries are similar to command-line switches and include the following

Configuration entries include the following[8][9]:

Entry	**Value**	**Description**
ArchiveDirectory	String	Name of directories at volume roots into which copy-on-delete files are moved. The directory is protected with a System ACL.
CheckRevocation	Boolean	Controls signature revocation checks. Default: True
CopyOnDeletePE	Boolean	Preserves deleted executable image files. Default: False
CopyOnDeleteSIDs	Strings	A comma-separated list of account SIDs for which file deletes will be preserved.
CopyOnDeleteExtensions	Strings	Extensions for files that are preserved on delete.

CopyOnDeleteProcesses	Strings	Process name(s) for which file deletes will be preserved.
DnsLookup	Boolean	Controls reverse DNS lookup. Default: True
DriverName	String	Uses species name for driver and service images.
HashAlgorithms	Strings	Hash algorithm(s) to apply for hashing. Algorithms supported include MD5, SHA1, SHA256, IMPHASH, and * (all). Default: None

Table V.2. Configuration Entries.

Event filtering entries

Event filtering allows us to filter the events you created. In many cases, events can be noisy, and putting things together is impossible. For example, you might be interested in network connections only for a certain process, but not all of them. You can filter the output on the server to reduce the data to collect.

The onmatch filter is applied if the events are matched. It can be changed with the "onmatch" property for the filter tag. If the value is 'include', it means that only matching events will be included. If it is set to 'exclude', the event will be included except if a rule matches. You can specify both the include and exclude filters for each event ID, where the exclusion matches take precedence.

Each filter can contain zero or more rules. Each tag in the filter tag is a field name from the event. Rules that assign a condition to the same field name act as an OR condition, and rules that specify another field name behave as an AND condition. Field rules may also use conditions to match the value. The conditions are as follows (all are case insensitive):

Condition	**Description**
is	Default, values are equals
is any	The field is one of the; delimited values
is not	Values are different
contains	The field contains this value
contains any	The field contains any of the; delimited values
contains all	The field contains any of the; delimited values

excludes	The field does not contain this value
excludes any	The field does not contain one or more of the; delimited values
excludes all	The field does not contain any of the; delimited values
begin with	The field begins with this value
end with	The field ends with this value
less than	Lexicographical comparison is less than zero
more than	Lexicographical comparison is more than zero
image	Match an image path (full path or only image name). For example: lsass.exe will match c:\windows\system32\lsass.exe

Table V.3. Event filtering entries.

Can use a different condition by specifying it as an attribute. This excludes network activity from processes with iexplore.exe in their path:

```
<NetworkConnect onmatch="exclude"> =<Image
condition="contains">something.exe</Image> </NetworkConnect>
```

To have Sysmon report which rule match resulted in an event being logged, add names to rules:

```
<NetworkConnect onmatch="exclude"> <Image name="network something"
condition="contains">something.exe</Image> </NetworkConnect>
```

It is also possible to override the way that rules are combined by using a rule group that allows the rule to combine types for one or more events to be set explicitly to AND or OR.

The following example demonstrates this usage. In the first rule group, a process created event will generate when timeout.exe is executed only with a command-line argument of "100", but a process terminated event will generate for termination of `ping.exe` and `timeout.exe`.

This is a Microsoft Sysinternals Sysmon configuration file template with default high-quality event tracing.

The file provided should function as a great starting point for system change monitoring in a self-contained package. This configuration and results should give a good idea of what's possible for Sysmon.

V.3. Data Transport System

V.3.1. Introduction Beats

Beats are open source data shippers that will install as agents on servers need to collect events to send different data types to Elasticsearch. Beats can send data directly to Elasticsearch or Logstash[10].

Beats is a platform in which small projects are generated running on certain types of data. ELK needs to use "beats" to act as a shipper to send data types from client to server.

Beat index patterns need to be installed on both the ELK server and the clients. On the ELK server, beats combine with components for data filtering, indexing, and rendering.

Some commonly used beats patterns are:

- **Packetbeat**: Send captured data from the ports to the server.
- **Topbeat**: As a monitor agent, we can collect hardware information such as CPU, RAM, ...
- **Filebeat**: Helps to transport logs from client to server.
- **Winlogbeat**: Helps to transport event logs from clients being Windows OS.
- **Metricbeat**: Collect data from the operating system, services such as Apache, HAProxy, MongoDB, Nginx, etc.

Beats platform:

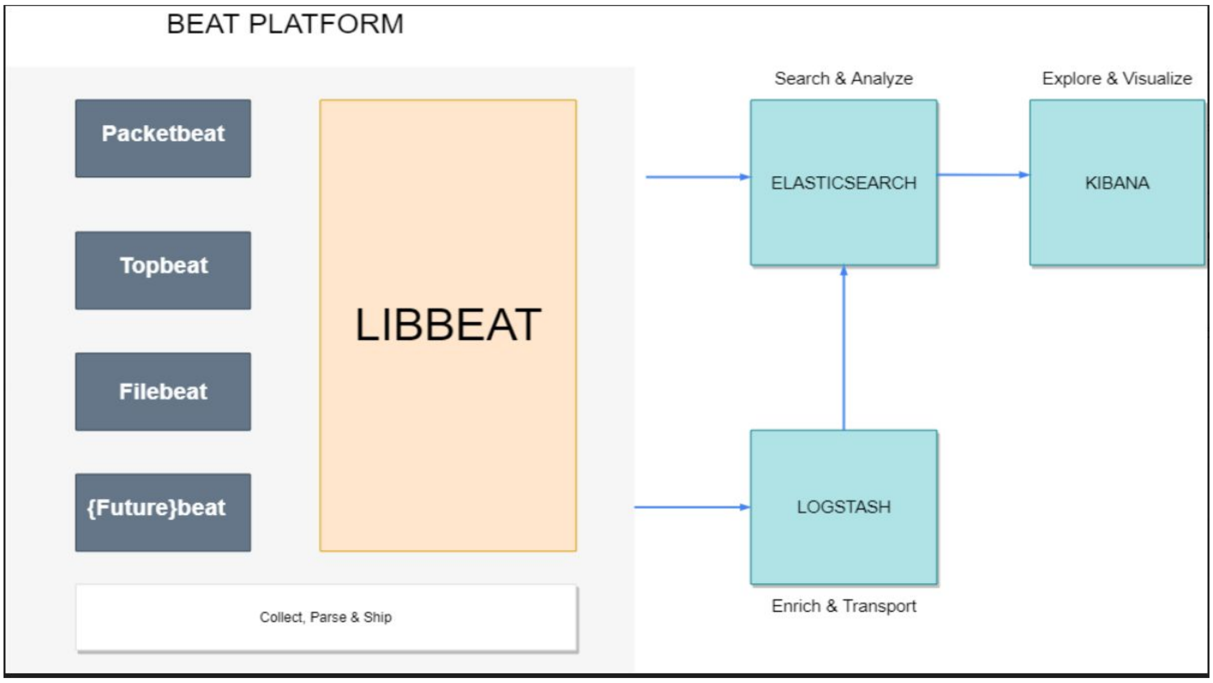


Figure V.2. Install and setup Winlogbeat in windows.

V.3.2. Introduction Winlogbeat

Winlogbeat forwards Windows event logs to Elasticsearch or Logstash.

Winlogbeat collects log data from one or more event logs using the Windows API filters the events based on user-configured criteria then sends these logged data to the configured outputs (Elasticsearch or Logstash). Winlogbeat monitors event logs so that new event data is sent in time. The location of the event logs is saved to the hard drive to allow Winlogbeat to continue after the reboot. (The read position for each event log is persisted to disk to allow Winlogbeat to resume after restarts.)

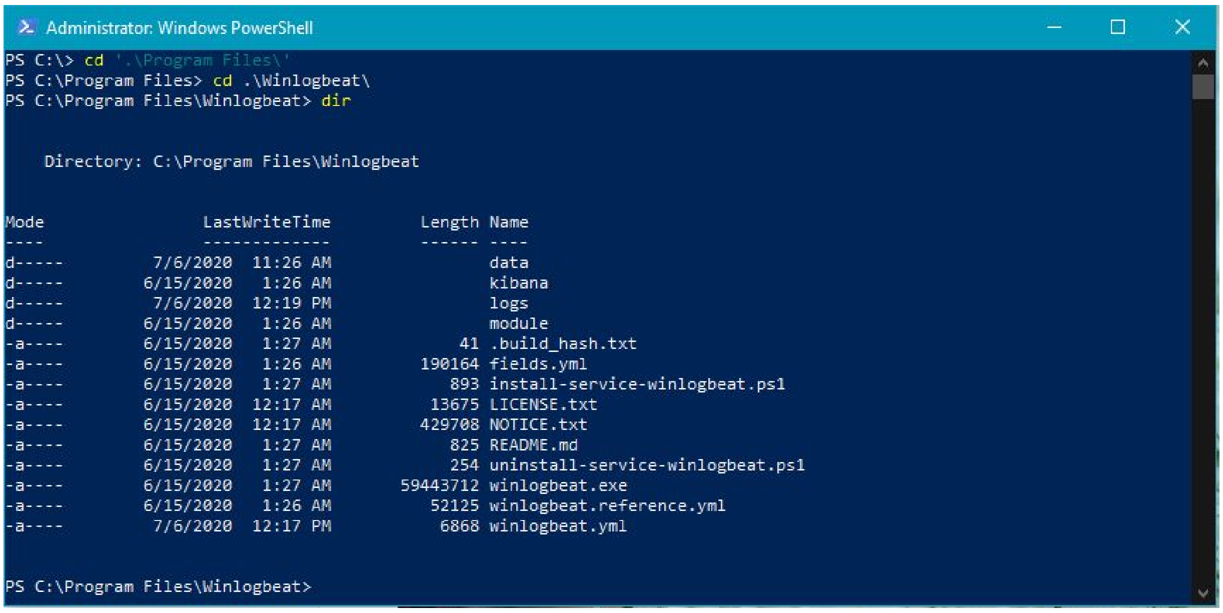
Winlogbeat can collect event data from any event logs running on the system. For example:

- application events
- hardware events
- security events
- system events

But in this project, we only consider and focus on Window Sysmon.

V.3.3. Setup and configuration

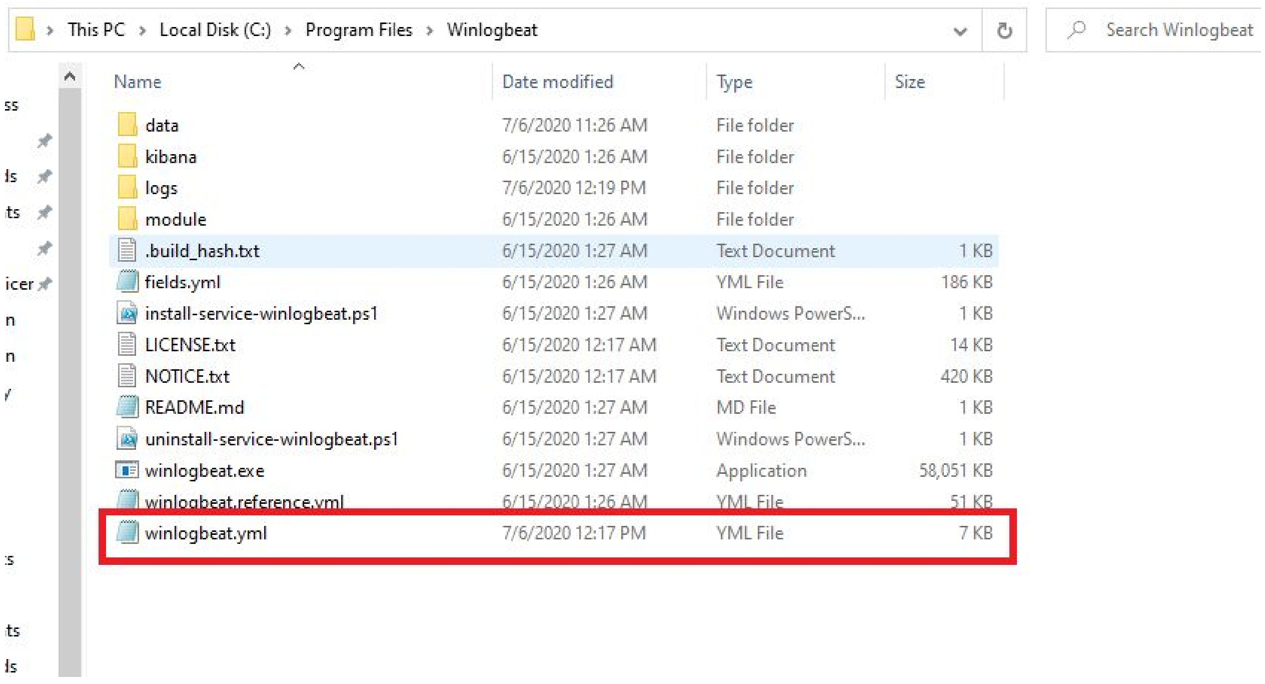
1) Download winlogbeat on the Elasticsearch homepage.
2) Extract the downloaded folder under the path: `C:\Program Files`
3) Rename the directory winlogbeat- <version> to Winlogbeat
4) Open PowerShell with admin rights, run the following command:



```
PS C:\> cd '.\Program Files\'
PS C:\Program Files> cd .\Winlogbeat\
PS C:\Program Files\Winlogbeat> dir


    Directory: C:\Program Files\Winlogbeat


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----         7/6/2020  11:26 AM                data
d-----        6/15/2020   1:26 AM                kibana
d-----         7/6/2020  12:19 PM                logs
d-----        6/15/2020   1:26 AM                module
-a----        6/15/2020   1:27 AM             41 .build_hash.txt
-a----        6/15/2020   1:26 AM         190164 fields.yml
-a----        6/15/2020   1:27 AM            893 install-service-winlogbeat.ps1
-a----        6/15/2020  12:17 AM          13675 LICENSE.txt
-a----        6/15/2020  12:17 AM         429708 NOTICE.txt
-a----        6/15/2020   1:27 AM            825 README.md
-a----        6/15/2020   1:27 AM            254 uninstall-service-winlogbeat.ps1
-a----        6/15/2020   1:27 AM       59443712 winlogbeat.exe
-a----        6/15/2020   1:26 AM          52125 winlogbeat.reference.yml
-a----         7/6/2020  12:17 PM           6868 winlogbeat.yml


PS C:\Program Files\Winlogbeat>
```

Figure V.3. All files after installation.

5) Config Winlogbeat (file winlogbeat.yml) send logs to server:

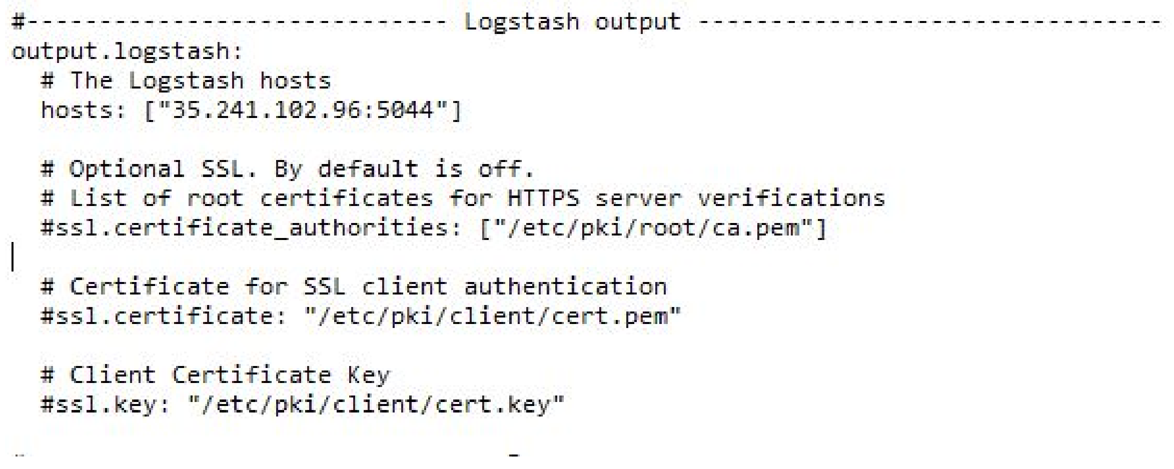


This PC > Local Disk (C:) > Program Files > Winlogbeat

Name	Date modified	Type	Size
data	7/6/2020 11:26 AM	File folder	
kibana	6/15/2020 1:26 AM	File folder	
logs	7/6/2020 12:19 PM	File folder	
module	6/15/2020 1:26 AM	File folder	
.build_hash.txt	6/15/2020 1:27 AM	Text Document	1 KB
fields.yml	6/15/2020 1:26 AM	YML File	186 KB
install-service-winlogbeat.ps1	6/15/2020 1:27 AM	Windows PowerS...	1 KB
LICENSE.txt	6/15/2020 12:17 AM	Text Document	14 KB
NOTICE.txt	6/15/2020 12:17 AM	Text Document	420 KB
README.md	6/15/2020 1:27 AM	MD File	1 KB
uninstall-service-winlogbeat.ps1	6/15/2020 1:27 AM	Windows PowerS...	1 KB
winlogbeat.exe	6/15/2020 1:27 AM	Application	58,051 KB
winlogbeat.reference.yml	6/15/2020 1:26 AM	YML File	51 KB
winlogbeat.yml	7/6/2020 12:17 PM	YML File	7 KB

Figure V.4. Configure the event logs that you want to monitor in winlogbeat.yml

In this project, we use Logstash to capture the data sent by the clients for processing by services detect before pushing them to Elasticsearch and rendering them, so instead of letting the data go straight to the ES, we need to send data to Logstash before.



```
#----------------------------- Logstash output --------------------------------
output.logstash:
  # The Logstash hosts
  hosts: ["35.241.102.96:5044"]

  # Optional SSL. By default is off.
  # List of root certificates for HTTPS server verifications
  #ssl.certificate_authorities: ["/etc/pki/root/ca.pem"]

  # Certificate for SSL client authentication
  #ssl.certificate: "/etc/pki/client/cert.pem"

  # Client Certificate Key
  #ssl.key: "/etc/pki/client/cert.key"
```

Figure V.5. Config host sends data to Logstash.

After the setup config file, run the commands that start Winlogbeat by command:

```
Start-Service Winlogbeat
```

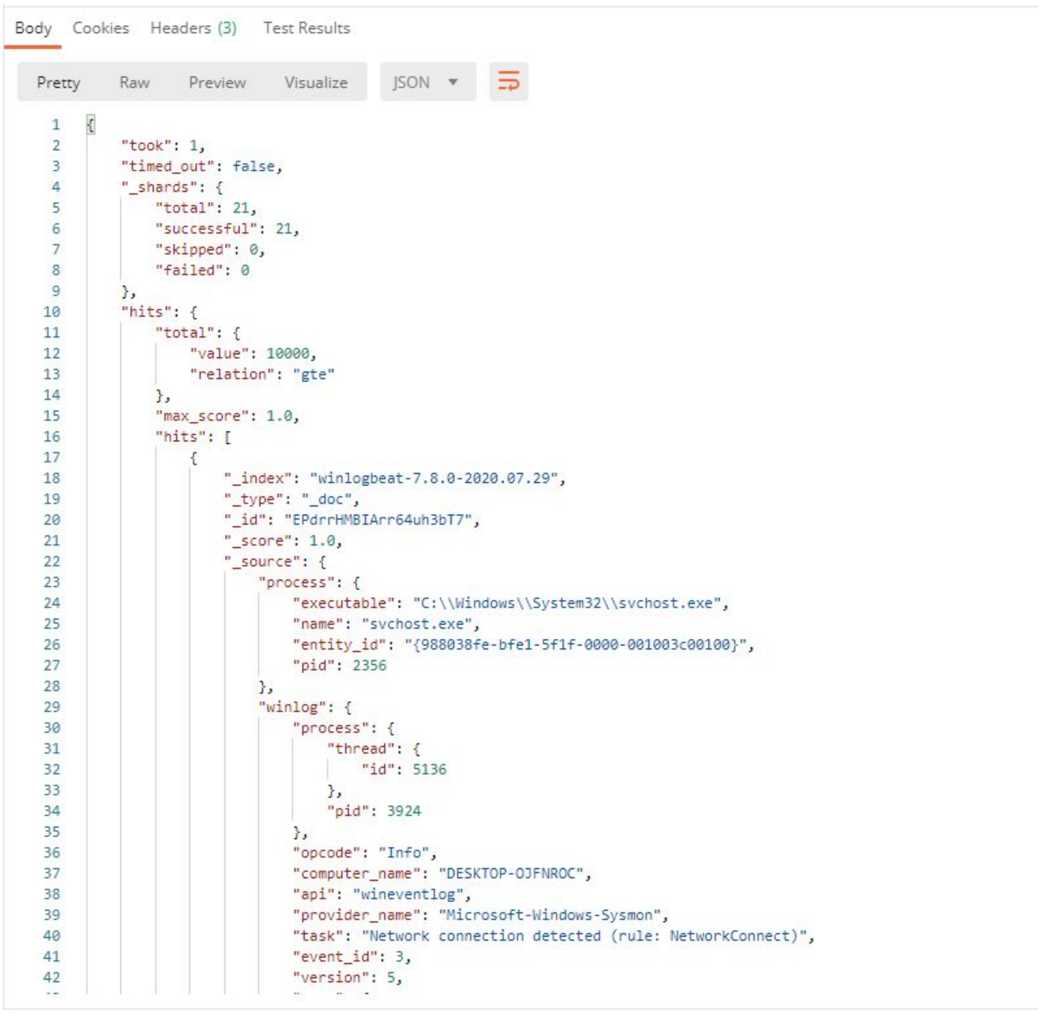
The status of service can be viewed and controlled from the services management console in Windows by command:

```
Status-Service Winlogbeat
```

Turn off Winlogbeat with the command:

```
Stop-Service Winlogbeat
```

Then check the Logs sent on Logstash by checking on Kibana or query using ES to get data by hostname and time:



```
{
    "took": 1,
    "timed_out": false,
    "_shards": {
        "total": 21,
        "successful": 21,
        "skipped": 0,
        "failed": 0
    },
    "hits": {
        "total": {
            "value": 10000,
            "relation": "gte"
        },
        "max_score": 1.0,
        "hits": [
            {
                "_index": "winlogbeat-7.8.0-2020.07.29",
                "_type": "_doc",
                "_id": "EPdrrHMBIArr64uh3bT7",
                "_score": 1.0,
                "_source": {
                    "process": {
                        "executable": "C:\\Windows\\System32\\svchost.exe",
                        "name": "svchost.exe",
                        "entity_id": "{988038fe-bfe1-5f1f-0000-001003c00100}",
                        "pid": 2356
                    },
                    "winlog": {
                        "process": {
                            "thread": {
                                "id": 5136
                            },
                            "pid": 3924
                        },
                        "opcode": "Info",
                        "computer_name": "DESKTOP-OJFNROC",
                        "api": "wineventlog",
                        "provider_name": "Microsoft-Windows-Sysmon",
                        "task": "Network connection detected (rule: NetworkConnect)",
                        "event_id": 3,
                        "version": 5,
```

Figure V.6. The data has been stored on Elasticsearch.

V.4. Monitoring And Anomaly Detection System

V.4.1. Overview of architectural

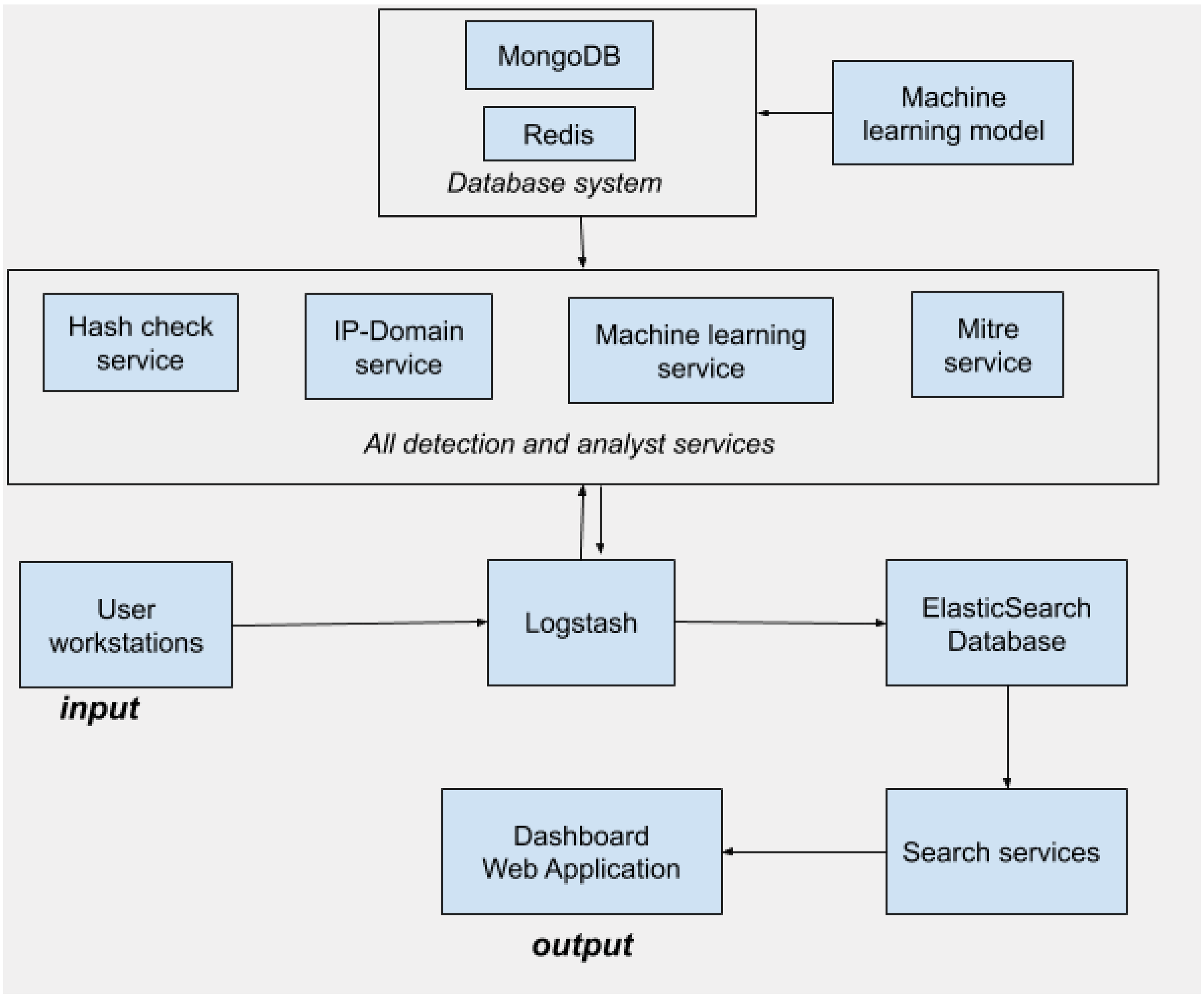


Figure V.7. Overview of architectural.

V.4.2. Detect the anomaly process based on the blacklist. (hash domain)

V.4.2.1. Introduction

The hash-service includes three APIs: the hash API, the IP API, and the domain API. We take the data that is the logs of the computers on the server to process and return the results.

V.4.2.2. Overview

URL	Method	Parameter	Result

/api/check	- POST	- hash .	- Json - `"result": { "info": {}, "status": "malicious" }` - "status“ field is malicious or clean - ”info” field is detail information of hash
/api/add	- POST	- hash_key - hash_value	-Add details of the hash in database
/ip-address	- POST	- ip	- Json `"result": { "data": { "info": { "harmless": 80, "malicious": 4, "suspicious": 0, "timeout": 0, "undetected": 9 }, "ip": "103.234.36.75", "status": "malicious" } }` - “info” field is number of antivirus detection -”status” field is status of IP
/domain		- domain	- Json `"result": { "data": { "domain": "facebook.com", "info": {`

			```
			"harmless": 85,
			"malicious": 0,
			"suspicious": 0,
			"timeout": 0,
			"undetected": 8
			},
			"status": "clean"
			}
			}
			```
			- "info" field is number of antivirus detection
			-"status" field is status of domain

Table V.4. API design.

V.4.2.3. Architecture

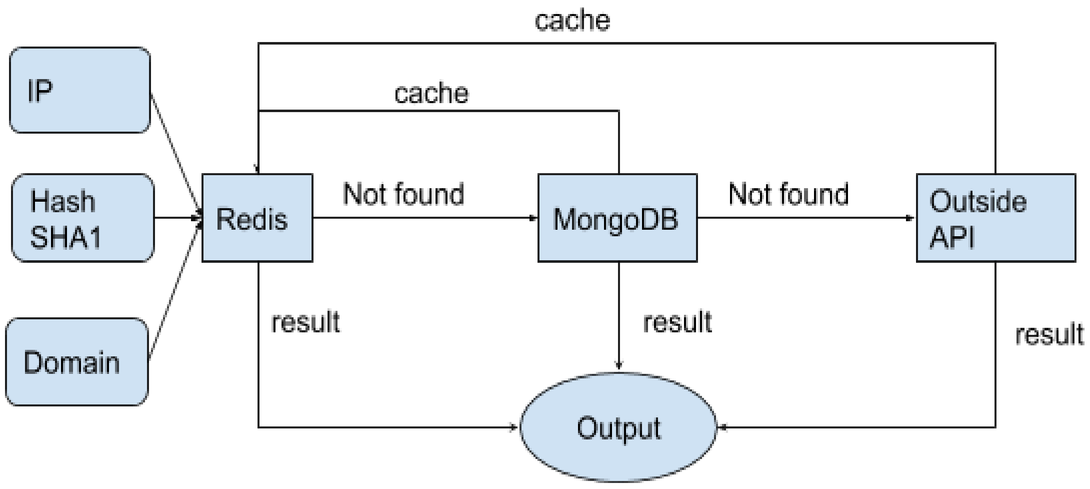


Figure V.8. API architecture.

V.4.2.4. Technology

- Mongodb
- Redis
- crawl data from virusshare
- Virustotal's API
- Flask

V.4.2.4. Hash service

Introduction

Hash-api is an important API. It is created to collect the hashed hash that is posted on the system.

Implementation

We have created a Virusshare's malware database[11] and a hash's collection in MongoDB to store the hashes and their details. The way connects to the database:

```
client = MongoClient(config.MONGO_CONECTION)
mongodb = client.VirusShare
hash_collection = mongodb['hash']
We create a cache and connect to it:
red = redis.StrictRedis(host='localhost', port=6379, db=1)
```

Every request is sent hash code to this API. This hash code will be searched in the cache. In the cache, the data is stored as {key: value}. "key" is the hash code, and "value" is the hash code details. When we search, we need to take the hash code.

```
def search_redis_lru_cache(hash):
    result = red.get(hash)
    if result == None:
        return False, None
    return True, json.loads(result.decode())
```

If it is not found in the cache, the API will search in the database. Input parameters are also key and value. The key is the type of hash code such as MD5, SHA1, SHA256. Value is the value of that hash.

```
def find_hash_db(key, value):
    result = hash_collection.find_one({key: value}, {'_id': False})
    return result
```

Finally, it sends the request to virusshare.com's API to find in virusshare.com's database.

```
def search_virus_share(hash):
    session.headers = {'User-Agent': config.USER_AGENT}
    data = {"search": hash, "start": "0"}
    response = session.post(url="https://virusshare.com/search", data=data)
    if "login" in response.text:
        auth = {"username": config.VIRUSSHARE_USERNAME,
                "password": config.VIRUSSHARE_PASSWORD}
        response = session.post(
            url="https://virusshare.com/processlogin", data=auth)
        response = session.post(
            url="https://virusshare.com/search", data=data)
    if "no results" in response.text:
        return None
    soup = BeautifulSoup(response.text, "html.parser")
    hash_dictionary = analysis_virus_share(soup)
    return hash_dictionary
```

The results will be saved in the database.

```
def add_hash_db(hash_dictionary):
       _id = hash_collection.insert_one(hash_dictionary).inserted_id
```

The result will be stored in the database and the cache.

```
def add_redis_lru_cache(hash, value):
    red.set(hash, json.dumps(value))
    if not value:
        red.expire(hash, config.TIMEOUT)
```

The processing of this API is as follows:

```
def find_hash(hash_type, hash):
    if not hash_valid_check(hash_type, hash):
        print('is not a valid hash')
        return None
    print('is a valid hash')
    found, result = search_redis_lru_cache(hash)
    if not found:
        print('not found in cache')
        result = find_hash_db(hash_type, hash)
        if not result:
            print('not found in db')
            result = search_virus_share(hash)
            if result:
                add_hash_db(result)

        if result is not None and '_id' in result.keys():
            del result['_id']
        add_redis_lru_cache(hash, result)
    return result
```

This is the first time a hash code has been requested in the API. It is checked if invalid then the API returns the result. Next, it is searched in the cache and database. If it is still not found, It will be searched on the internet. It sends requests to virusshare.com's API to search its database. It returns a result that will be analyzed and stored in the database and cache. The result is returned.

The API allows the user to add hash codes in the database.it will be checked by the API.

```
def hash_valid_check(hash_type, hash_value):
    for c in hash_value:
        if c not in string.hexdigits:
            return False
    if hash_type == MD5:
        return len(hash_value) == 32
    if hash_type == SHA256:
        return len(hash_value) == 64
    if hash_type == SHA1:

          return len(hash_value) == 40
      return False
```

It will be stored in another database. If the hash codes add to the database.

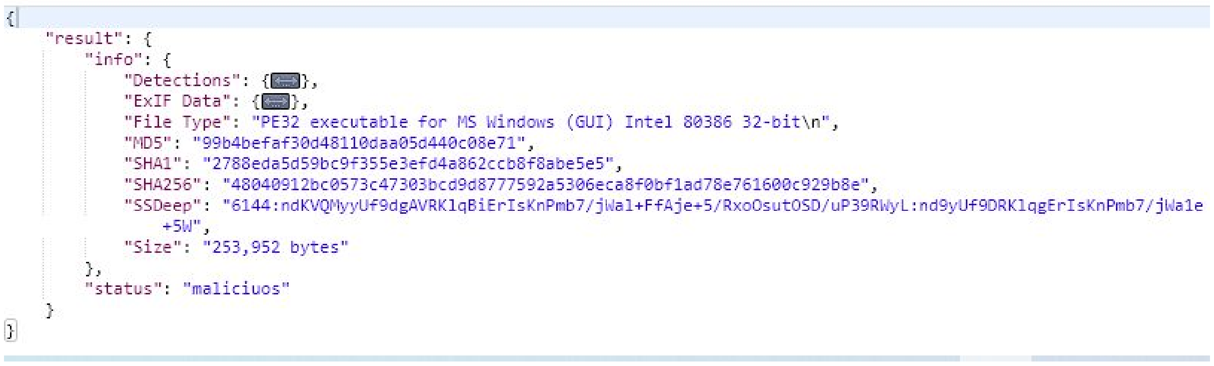
```
def user_add_db(hash_key, hash_value, status, user='user'):
    user_dictionary = {}
    user_dictionary[hash_key] = hash_value
    user_dictionary['status'] = status
    user_dictionary['user'] = user
    _id = hash_collection.insert_one(user_dictionary).inserted_id
```

Result

This is the parameter:

```
{
    "hash": "{\"sha1\":\"2788eda5d59bc9f355e3efd4a862ccb8f8abe5e5\"}"
}
```

This is the result:



```
{
    "result": {
        "info": {
            "Detections": {↔},
            "ExIF Data": {↔},
            "File Type": "PE32 executable for MS Windows (GUI) Intel 80386 32-bit\n",
            "MD5": "99b4befaf30d48110daa05d440c08e71",
            "SHA1": "2788eda5d59bc9f355e3efd4a862ccb8f8abe5e5",
            "SHA256": "48040912bc0573c47303bcd9d8777592a5306eca8f0bf1ad78e761600c929b8e",
            "SSDeep": "6144:ndKVQMyyUf9dgAVRK1qBiErIsKnPmb7/jWal+FfAje+5/RxoOsutOSD/uP39RWyL:nd9yUf9DRK1qgErIsKnPmb7/jWa1e
                +5W",
            "Size": "253,952 bytes"
        },
        "status": "maliciuos"
    }
}
```

Figure V.9. The result returned from hash services.

V.4.2.5. IP-domain service

Introduction

This service has the responsibility to check a connected domain or IP in black list. Here we use Virustotal as a database of known samples.

Implementation

Connect to the MongoDB in Python

```
mongo = MongoClient(config.MONGO_CONECTION)
mongodb = mongo.DomainIP
ip_collection = mongodb['ipList']
```

And access to Redis cache

```
redisip = redis.StrictRedis(host='localhost', port=6379, db=0)
```

The code searches IP that is returned in the cache.

```
def search_redis(obj):
    result = redip.get(obj['ip'])
    if result == None:
        return False, None
    return True, json.loads(result.decode())
```
Next, it searches in the database:
```
def search_db(obj):
   result = ip_collection.find_one(obj['ip'], {'_id': False})
   return result
```

If the IP isn’t found in both cache and database. The API will send requests to Virustotal's API. After that, it will analyze and store the results.
```
def getipdomain(obj):
    session.headers = {'User-Agent': config.USER_AGENT,"x-apikey": config.VIRUSTOTAL_API_KEY,}
    if mode == 0:
        url = "https://www.virustotal.com/api/v3/domains/"+obj['domain']
    else:
        url = "https://www.virustotal.com/api/v3/ip_addresses/"+obj['ip']
    r = session.get(url)
    if r.status_code == 200:
        data = r.json()
    else:
        return None
    result = {}
    result['ip'] = obj['ip']
    if data['data']['attributes']['last_analysis_stats']['malicious']>0:
        result['status'] = 'malicious'
    else:
        result['status'] = 'clean'
    result['info'] = data['data']['attributes']['last_analysis_stats']
    return result
```

The results are stored in the database.
```
def add_db(result):
    _id = ip_collection.insert_one(result).inserted_id
```
And redis:
```
def add_redis(result):
    redip.set(result['ip'],json.dumps(result))
```

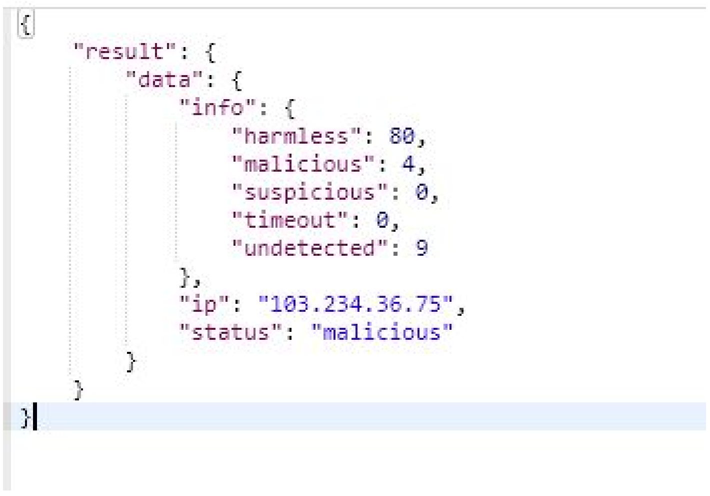
The processing of this API:

```
def checkIpDomain(obj,mode):
    flag, result = search_redis(obj)
    if not flag:
        result = search_db(obj)
        if result == None:
            result = getipdomain(obj)
            if result == None:
                return "Not Found"
            add_db(result,mode)
        add_redis(result, mode)
    return result
```

Result

The parameters are :`{"ip":"103.234.36.75"}`

The result :



```
{
    "result": {
        "data": {
            "info": {
                "harmless": 80,
                "malicious": 4,
                "suspicious": 0,
                "timeout": 0,
                "undetected": 9
            },
            "ip": "103.234.36.75",
            "status": "malicious"
        }
    }
}
```

Figure V.10. The result returned from IP services.

V.4.2.6. Domain service

Introduction

This API collects domains to which computers in the system have ever been connected.

Implementation

Connects to the MongoDB database using Python

```
mongo = MongoClient(config.MONGO_CONECTION)
mongodb = mongo.DomainIP
domain_collection = mongodb['domainList']
```

Connects to the cache:

```
redisdomain = redis.StrictRedis(host='localhost', port=6379, db=2)
```

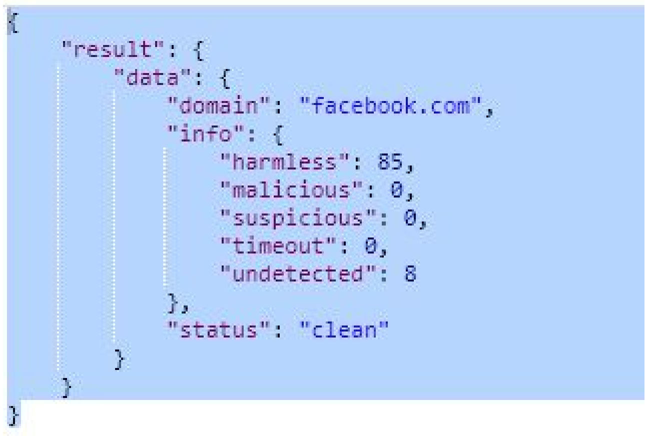
Because the API's implementations are similar to those of the IP API, I won't repeat it.

Result

The parameters:

```
{"domain":"facebook.com"}
```

The result:



```
{
    "result": {
        "data": {
            "domain": "facebook.com",
            "info": {
                "harmless": 85,
                "malicious": 0,
                "suspicious": 0,
                "timeout": 0,
                "undetected": 8
            },
            "status": "clean"
        }
    }
}
```

Figure V.11. The result returned from domain services.

V.4.3. Detect the anomaly process by analyst connected domain using machine learning

V.4.3.1. Introduction to Machine Learning

Artificial Intelligence (AI) is intelligence expressed by a machine system with similarities to human intelligence. These systems have capabilities such as learning new knowledge, identifying and responding to information, or trying to solve a problem on its own.

Machine learning is an application of artificial intelligence (AI) that gives systems the ability to automatically learn and improve from experience without the need for explicit programming. Machine learning focuses on developing computer programs that can access data and use it for self-learning.

Technology revolution 4.0 broke out, witnessing the explosion and strong development of AI and ML technology. Great achievements are reflected in the computing power and the development of models such as big data, cloud computing.

Today, the increasing interest in machine learning is because thanks to machine learning that helps increase the storage capacity of available data, computational processing is much cheaper and more efficient.

It does quickly and automatically create models that allow for larger and more complex data analysis. It results more quickly and accurately.

It is the efficiency at work and the outstanding benefits that it gives us that make machine learning more and more focused and interested.

ML's first commercial applications were pioneered by technology veterans like Google (in search engines), Amazon (with product suggestions), and Facebook (with news feeds). These businesses have built a system that aggregates valuable behavioral data from hundreds of millions of users. To be able to collect, organize, and analyze data in the most effective way, they have continued to build, improve, and research more algorithms and high technologies

It is applied in many industries in practice such as Financial Services, Banking, Government, Health, Marketing and Sales Services, Logistics, etc.

In information assurance, machine learning can automate complex steps and processes to promptly detect and prevent intrusions. Machine learning can also find an application's security weaknesses.

V.4.3.2.Technology

- Hadoop
- Spark
- Pyspark
- MongoDB
- Flask

V.4.3.3. Architecture

System architecture

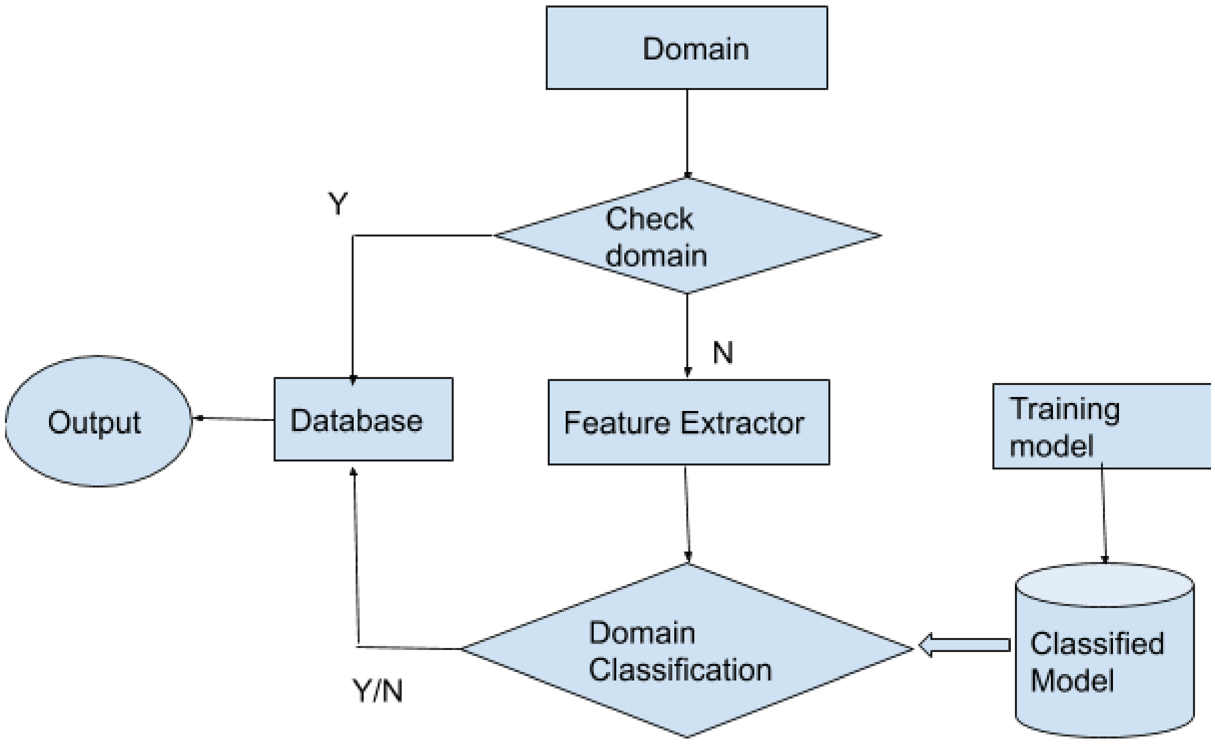


Figure V.12. System architecture.

The domain detection system consists of 4 phases:

First, the system will receive domains from sysmon and check them with my blacklist and white list. If we appear in this list, we will warn users and push my database.

Next, the system will extract features which are important to detect malicious domains.

Third, we will push them in a classified model to predict them.

Fourth,the result will be transferred to the action center. the messages that have been identified as malicious or clean and they will retrieve from the database system by the system to train themselves.

Machine Learning architecture

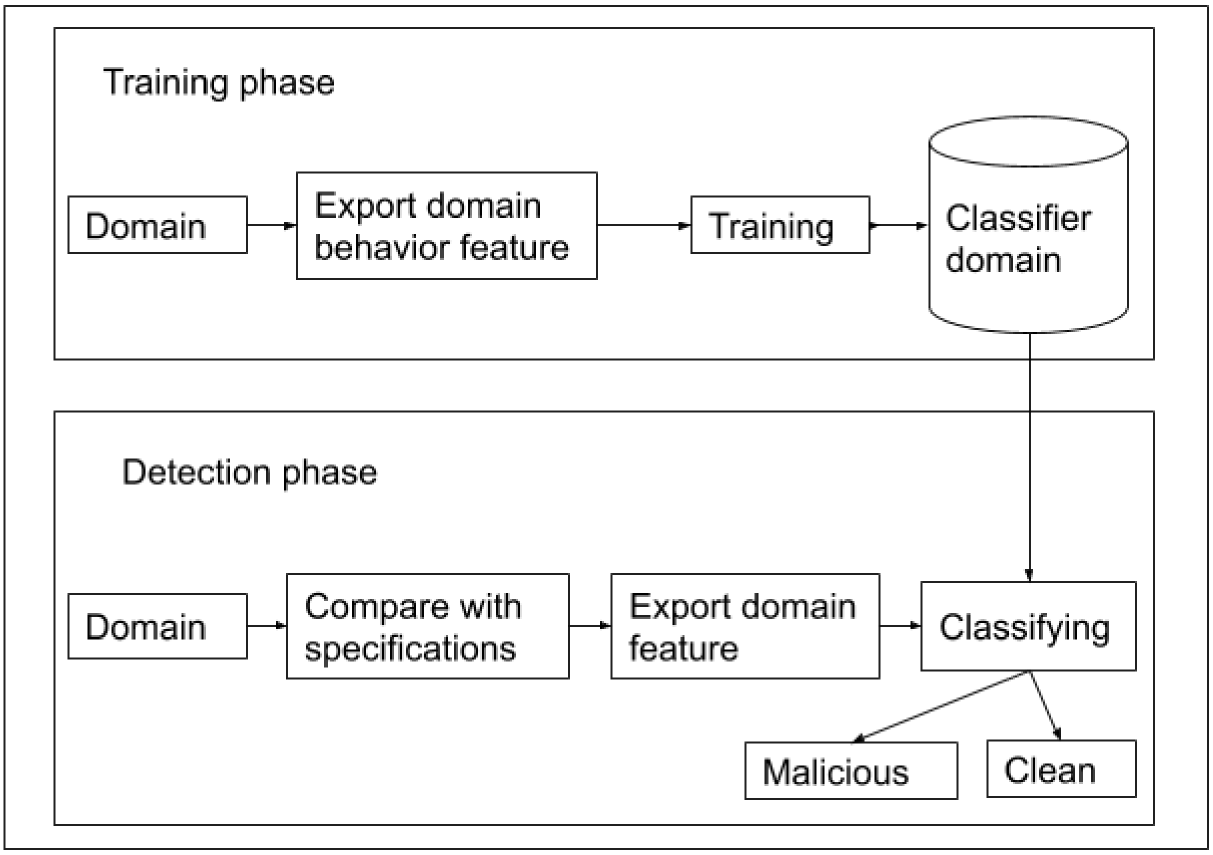


Figure V.13. Machine Learning architecture.

Training and Learning phase: Detector try to learn behavior, features of malicious domain. This is the phase where the model was built for malicious domain detection.

Detection phase: Based on machine learning at phase 1, detectors will detect malicious domains and send notification based on machine learning algorithms.

V.4.3.4.Algorithm

Random Forest is the algorithm that the team will choose to use in the Machine Learning section of the project.

Random Forest is a Decision Tree algorithm. Based on the construction of a "tree forest" in which each tree is a Decision Tree, through trees, Random Forest forms a series of rules and makes judgments based on the aggregation of the results of each tree.

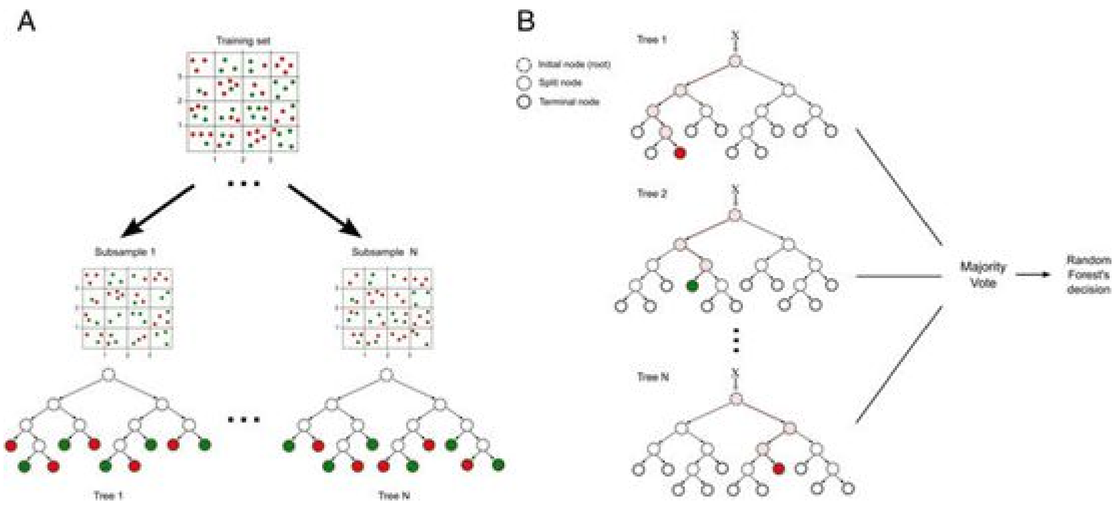


Figure V.14. Random forest algorithm[12]

Advantages of Random Forest

The random forest algorithm is not biased, since there are multiple trees and each tree is trained on a subset of data. Basically, the random forest algorithm relies on the power of "the crowd"; therefore, the overall biasedness of the algorithm is reduced.

This algorithm is very stable. Even if a new data point is introduced in the dataset the overall algorithm is not affected much since new data may impact one tree, but it is very hard for it to impact all the trees.

The random forest algorithm works well when you have both categorical and numerical features.

The random forest algorithm also works well when data has missing values or it has not been scaled well (although we have performed feature scaling in this article just for the purpose of demonstration).

Disadvantages of Random Forest

A major disadvantage of random forests lies in their complexity. They required much more computational resources, owing to the large number of decision trees joined together.

Due to their complexity, they require much more time to train than other comparable algorithms.

Principle

1. Pick N random records from the dataset.
2. Build a decision tree based on these N records:
   Choose the number of trees you want in your algorithm and repeat steps 1 and 2.

   In case of a regression problem, for a new record, each tree in the forest predicts a value for Y (output). The final value can be calculated by taking the average of all the values predicted by all the trees in forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

   So, to build an RF model, you need to go through the following basic stages:

   - Select the "k" features from a collection of features used to detect malicious domains.

   - From the above "k" features groups, build Decision Trees and a series of rules.

   - Training and testing for RF models.

Choose "k" features

As mentioned earlier, Random Forest is a decision-making algorithm based on synthesizing results from Decision Tree. However, there is a problem if all trees are built in the same way, so they will give the same answers and creating trees will not work. What's more, while different "forest" trees are the cause of differences as well as the advantages of the Random Forest algorithm.

So, to ensure that the selection of "k" features is not identical and random, people use a method called bootstrapping. Bootstrapping is a method that allows the selection of an initial random "k" features, creating new "k" features by deleting a portion of features from the original "k" features group and replacing them with random features. another course. Doing so ensures the following features groups remain very close to the original feature group, but there are still changes to ensure that there are differences. In addition, to really make sure that decision trees are different, the Random Forest algorithm will randomly ignore some questions when building the Decision Tree. In this case, if the first question is not selected, the next question will be selected to build the tree. This process is called a sampling feature.

Randomly changing the features and ignoring some of the above questions will work when the dataset in use is incomplete, the algorithm references an unprecedented feature.

In addition, bootstrapping is used not only to increase the accuracy of the algorithm but also to minimize the risk of over fitting. Overfitting is a term used for cases where models are trained too accurately. This model "learns" all the details (including noise) that are in the Training Set to the point that it negatively affects its performance when applied to the new data set. Because in the new data set, the noise that it "learns" from the Training Set does not exist. This will result in a loss of the overall calculation of the model.

The above are the issues as well as the meaningful characteristics of selecting "k" features groups to use to build a decision tree "forest" in the Random Forest algorithm.

Building the Decision Tree

Decision Tree is a development decision chart with a tree structure in which:

- Root: the top node, and the start node of the tree.
- Internal node: is an intermediate node on a single feature.
- Branch: represents the results of the test on the node.
- Leaf node: represents class.

To build a Decision Tree, you can choose different algorithms such as ID3, C4.5, ... to build. However, basically these algorithms are based on using the Entropy function to calculate Information Gain.

$$H(\mathbf{p}) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

Figure V.15. Function entropy.

In which, **p** is the probability distribution of a discrete variable that can receive **n** different values.

Apart from the advantages of high accuracy results, Decision Tree has some advantages such as:

- The Decision Tree construction process does not use knowledge of the research field, or input parameters.
- The process of "learning" is performed in the form of trees, so it is easy to understand and express

However, Decision Tree still has some disadvantages such as:

- For data sets with multiple features, the Decision Tree will be large (depth and horizontal), thus reducing the comprehension and increasing the complexity of the algorithm.
- Ranking features to branches based on previous branching and ignoring the interdependence between features.
- When using Information Gain to determine branch properties, features with multiple values are often preferred.

Training and Testing model

Training process and Testing is the process of building up the Series of Rules from Decision Tree's "forest" and checking the accuracy that the model judges after training. Here, the input dataset will be split into two parts, partly for training and the rest for testing.

Training, from the data used for training, will be used to create the Decision Tree forest, and from each Decision Tree, each branch goes to each leaf of different nodes, creating different rules, summarize all the rules obtained from the above Decision Tree to obtain the Series of Rules.

After acquiring the Series of Rules, testing will test the accuracy of the model given based on the above Series of Rules for records in testing data. And from there it gives Accuracy of algorithms.

Calculation parameter

- Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$
- Precision $= \frac{TP}{TP+FP}$
- Recall $= \frac{TP}{TP+FN}$
- Test Error = 100% - Accuracy

Figure V.16. Calculation parameter.

Parameter	Notes	Calculation process
TP	True Positive – result of predicting domain correctly	Count number of domains predicted is domain and it is correct
TN	True Negative – result of predicting domain correctly.	Count number of domains predicted is normal and it is correct
FP	False Positive - result of predicting domain incorrectly.	Count number of domains predicted is phishing and it is incorrect
FN	False Negative – result of predicting domain is normal incorrectly.	Count number of domains predicted is normal and it is correct
FPR	False Positive Rate.	False alert rate.
FNR	False Negative Rate.	Miss rate.

TPR	True Positive Rate.	Accuracy rate of predicting domains which have a true label is 'malicious'.
TNR	True Negative Rate.	Accuracy rate of predicting domains which have a true label is 'normal'.

Table V.5. Calculation parameter.

V.4.3.5.Implementation

Feature selection

The survey results are given in a number of scientific articles with topics about malicious domain detection with the properties used in the same sample data set. And most studies are based on feature groups including[13]:

Domain name lexical features: The features are extracted domain name.It is basic information of domain. We can based on this information to predict malicious domain

Ranking features: The features are extracted from the famous websites as alexa.com's database ,vv .

DNS query features : We send DNS query packets and the server 's domains reponse the information : ip,mail exchange,....

Other features: The features which register in whois

Featured list that the project team has referenced

Order	Category	Feature	Type	Description
1	Lexical features	Domain name length	Integer	Length of domain
2		Domain name token count	Integer	Number of token in domain which split with ','

3		Average domain token length	Real	Average length of domain
4		Longest domain token length	Integer	Longest token in domain name
5		Number of IP address in domain name	Integer	Number IP address in domain name
6		Number of special characters	Integer	Number of special characters in domain
7		Number of digits	Integer	Number of digits in domain
8		Number of continuous digits	Integer	Number of continuous digits in domain
9		Longest continuous digits length	Integer	Longest continuous digits length
10		Number of continuous letters	Integer	Number of continuous letters
11		Longest continuous letters length	Integer	Longest continuous letters length
12		Brand name presence	Binary	Brand name presence in domain name

13	Ranking features	Rank in Alexa host	Integer	Top domain in 1 million domains list in Alexa host
14		Rank in Alexa country	Integer	Top domain in 1 million domains list in Alexa country
15		Rank in Domcop	Integer	Top domain in 10 million domains list in Domcop
16	DNS query features	Resolved IP count	Integer	The number of IPs return in DNS queries
17		HTTP response status	Integer	HTTP response status
18		Name server count	Integer	The number of name server records return in DNS queries .
19		Mail exchange server count	Integer	The number of mail exchange server records return in DNS queries
20		Time to live (TTL)	Integer	Cache record ‘s TTL is domain name in name server
21	Other feature	AgeDomain	Integer	How long did the domain register ?

Table V.6. Feature selection.

<u>*Build*</u>

Create a connection to MongoDB database in python.

```
mongo = MongoClient(config.MONGO_CONECTION)
mongodb = mongo.ML
collection = mongodb['List']
```

Next, the features are exported and stored in the file.

The features are trained by machine learning algorithms.The algorithm has the parameters:

```
rf = RandomForestClassifier(labelCol="indexedLabel", featuresCol=
"indexedFeatures",numTrees=30)
```

"labelCol" is the header of the label, "featuresCol" is the header of the features fields. "numTrees" is the number of Decision trees created in the algorithm. Results return a model.

Thanks to that model, the API predicts the domains.

The results of the prediction are saved in the database. Each request is sent to the server then it checks in the database before being put into machine learning for prediction. It increases the performance of the whole system.

The processing of this API is as follows:

```
def ml(obj):
    result = search_db(obj)
    print(result)
    if result == None:
        data = detect_ml(1,obj['domain'])
        if data == 0:
            result = {'domain':obj['domain'],'status':'clean'}
        else:
            result = {'domain':obj['domain'],'status':'malicious'}
        add_db(result)
    if '_id' in result:
        del result['_id']
    return result
```

Testing

The dataset includes:

Domain types	**Number of records**
Malicious domain	99155
Clean domain	63898
Total	163053

Table V.7. Quantity statistics.

The dataset is collected from a variety of sources: malwaredomainlist.com 's database, malwaredomains.com 's database, alexa 's database. Domains are checked by Virustotal's API. Then, they have extracted the features. The result will save in file.

We run a dataset with SVM, Naive Bayes, Random Forest. The data set is divided into a 4:1 ratio. In which, four parts train the algorithm, the rest test the algorithm. SVM and Naive Bayes have no parameters. The Random forest parameters are as above. The result :

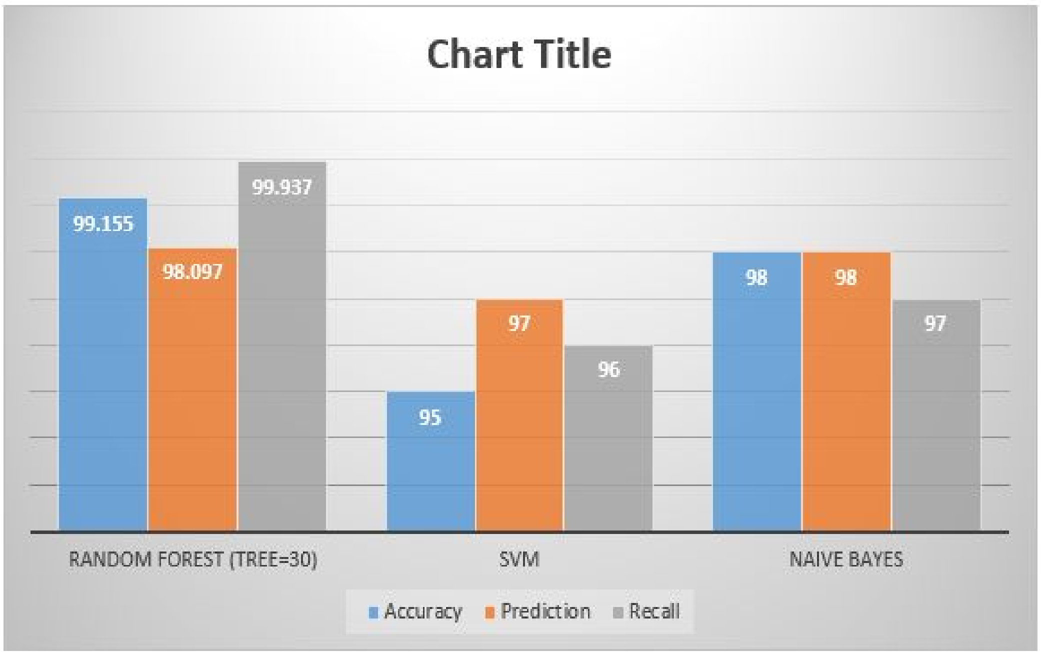


Figure V.17. Compare with another algorithm.

In the figure V.4.3.5-a, we use the Random forest algorithm.

The feature importances:

Feature	**Percent**
domcop_rank	38.4216
host_rank	19.2606
country_rank	10.6246
HTTP_response_status	10.4635
Resolved_IP_count	5.1040
Time_to_live	4.7968
EmbeddedBrandName	3.3256
Mail_exchange_server_count	1.9350
avg_domain_token_len	1.1916
SSL_certification	1.0331
number_digits	0.9944
longest_digits	0.7502
longest_token	0.7410
Name_server_count	0.6436
domain_name_length	0.2459

longest_letters	0.1751
number_letters	0.1126
number_con_digits	0.0957
number_con_letters	0.0476
age_of_domain	0.0334
domain_name_token_count	0.0042

Table V8. Feature Importances.

Evaluate for Flow

Total predictions: 49004

Labels	**count**
1	29742
0	19262

Table V.9. Evaluate for Flow.

Confusion Matrix:

	Positive	**Negative**
True	29116 (97.895%)	19237 (99.870%)
False	626 (2.105%)	25 (0.130%)

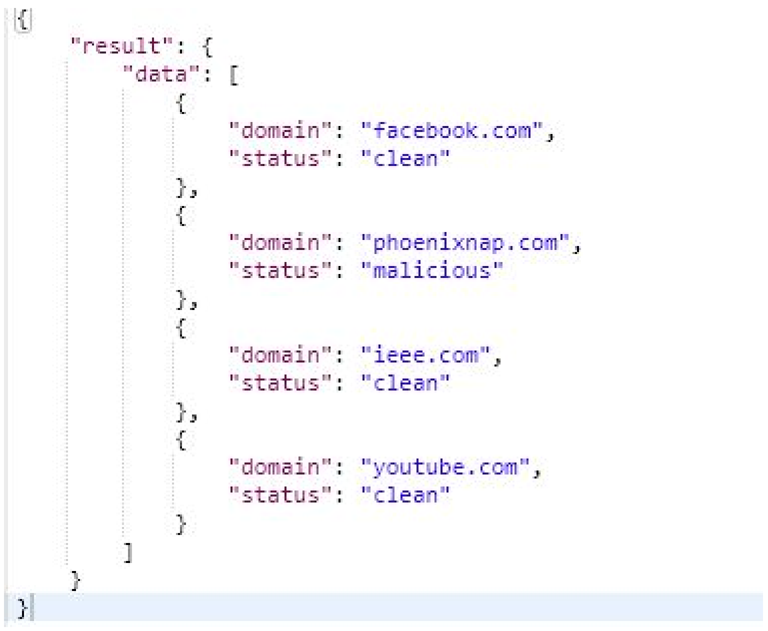
Table V.10. Confusion matrix.

Result

The parameter

```
{
	"domain":"facebook.com,phoenixnap.com,ieee.com,youtube.com"
}
```

The result



```
{
    "result": {
        "data": [
            {
                "domain": "facebook.com",
                "status": "clean"
            },
            {
                "domain": "phoenixnap.com",
                "status": "malicious"
            },
            {
                "domain": "ieee.com",
                "status": "clean"
            },
            {
                "domain": "youtube.com",
                "status": "clean"
            }
        ]
    }
}
```

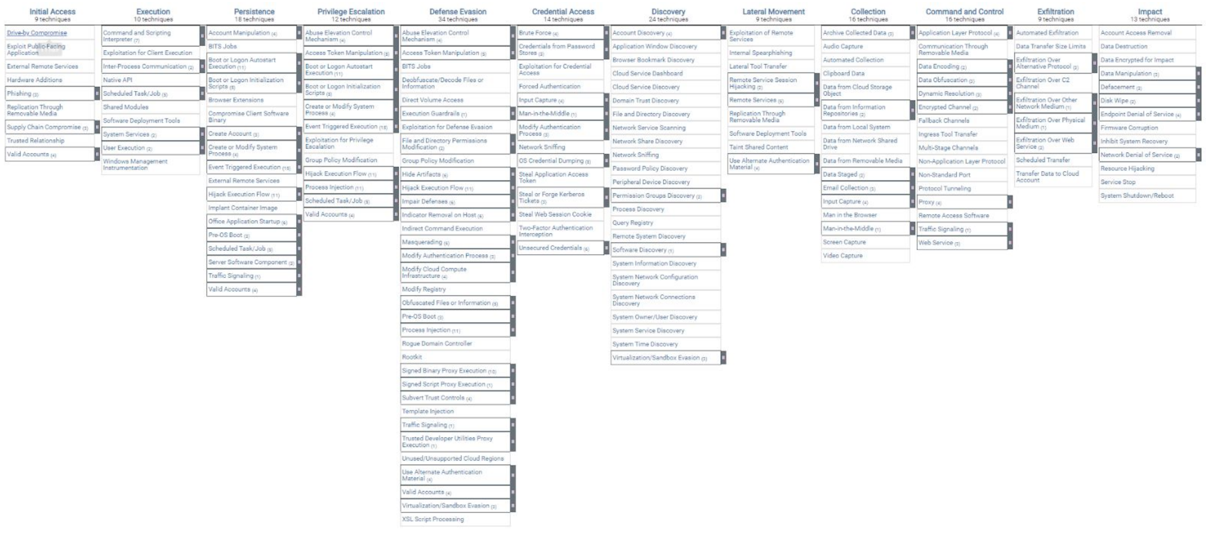
Figure V.18. The result returned from ml services.

V.4.4. Detect the anomaly process by analyst behaviors (Mitre)

V.4.4.1. Introduction Mitre Att&ck

In fact, malicious code will have many different ways and processes to hide and bypass surveillance systems. In recent studies, our team often focuses and extracts the properties and behavior of the malware based on data collected from the sandbox. However, we realize that collecting malicious behavior will not be able to guarantee all of their behavior is fully documented. Also, quick detection time is not guaranteed. Therefore, in this project, we will not directly extract anomalies of malicious code based on data analysis and evaluation from the sandbox. Instead, we use Miter Att&ck to define and behave malicious behavior.

Miter Att&ck is a knowledge base of attack tactics and techniques gathered on observations of actual attacks. Each attack tactic represents a target or a certain stage in the attack (How to escalate privileges, how to fix malicious code on the victim's system). And attack techniques, which describe methods to achieve those goals. Miter Att&ck currently has 11 popular attack tactics including Initial Access, Execution, Persistence, Privilege Escalation, Defense Evasion, Credential Access, Discovery, Lateral Movement, Collection, Exfiltration, Impact[14].



Initial Access 9 techniques	Execution 10 techniques	Persistence 18 techniques	Privilege Escalation 12 techniques	Defense Evasion 34 techniques	Credential Access 14 techniques
Drive-by Compromise	Command and Scripting Interpreter (7)	Account Manipulation (4)	Abuse Elevation Control Mechanism (4)	Abuse Elevation Control Mechanism (4)	Brute Force (4)
Exploit Public-Facing Application	Exploitation for Client Execution	BITS Jobs	Access Token Manipulation (5)	Access Token Manipulation (5)	Credentials from Password Stores (3)
External Remote Services	Inter-Process Communication (2)	Boot or Logon Autostart Execution (11)	Boot or Logon Autostart Execution (11)	BITS Jobs	Exploitation for Credential Access
Hardware Additions	Native API	Boot or Logon Initialization Scripts (5)	Boot or Logon Initialization Scripts (5)	Deobfuscate/Decode Files or Information	Forced Authentication
Phishing (3)	Scheduled Task/Job (5)	Browser Extensions	Create or Modify System Process (4)	Direct Volume Access	Input Capture (4)
Replication Through Removable Media	Shared Modules	Compromise Client Software Binary	Event Triggered Execution (15)	Execution Guardrails (1)	Man-in-the-Middle (1)
Supply Chain Compromise (3)	Software Deployment Tools	Create Account (3)	Exploitation for Privilege Escalation	Exploitation for Defense Evasion	Modify Authentication Process (3)
Trusted Relationship	System Services (2)	Create or Modify System Process (4)	Group Policy Modification	File and Directory Permissions Modification (2)	Network Sniffing
Valid Accounts (4)	User Execution (2)	Event Triggered Execution (15)	Hijack Execution Flow (11)	Group Policy Modification	OS Credential Dumping (8)
	Windows Management Instrumentation	External Remote Services	Process Injection (11)	Hide Artifacts (6)	Steal Application Access Token
		Hijack Execution Flow (11)	Scheduled Task/Job (5)	Hijack Execution Flow (11)	Steal or Forge Kerberos Tickets (3)
		Implant Container Image	Valid Accounts (4)	Impair Defenses (6)	Steal Web Session Cookie
		Office Application Startup (6)		Indicator Removal on Host (6)	Two-Factor Authentication Interception
		Pre-OS Boot (2)		Indirect Command Execution	Unsecured Credentials (6)
		Scheduled Task/Job (5)		Masquerading (6)	
		Server Software Component (3)		Modify Authentication Process (3)	
		Traffic Signaling (1)		Modify Cloud Compute Infrastructure (4)	
		Valid Accounts (4)		Modify Registry	
				Obfuscated Files or Information (5)	
				Pre-OS Boot (3)	
				Process Injection (11)	
				Rogue Domain Controller	
				Rootkit	
				Signed Binary Proxy Execution (10)	
				Signed Script Proxy Execution (1)	
				Subvert Trust Controls (4)	
				Template Injection	
				Traffic Signaling (1)	
				Trusted Developer Utilities Proxy Execution (1)	
				Unused/Unsupported Cloud Regions	
				Use Alternate Authentication Material (4)	
				Valid Accounts (4)	
				Virtualization/Sandbox Evasion (3)	
				XSL Script Processing	

Discovery 24 techniques	Lateral Movement 9 techniques	Collection 16 techniques	Command and Control 16 techniques	Exfiltration 9 techniques	Impact 13 techniques
Account Discovery (4)	Exploitation of Remote Services	Archive Collected Data (3)	Application Layer Protocol (4)	Automated Exfiltration	Account Access Removal
Application Window Discovery	Internal Spearphishing	Audio Capture	Communication Through Removable Media	Data Transfer Size Limits	Data Destruction
Browser Bookmark Discovery	Lateral Tool Transfer	Automated Collection	Data Encoding (2)	Exfiltration Over Alternative Protocol (3)	Data Encrypted for Impact
Cloud Service Dashboard	Remote Service Session Hijacking (2)	Clipboard Data	Data Obfuscation (3)	Exfiltration Over C2 Channel	Data Manipulation (3)
Cloud Service Discovery	Remote Services (6)	Data from Cloud Storage Object	Dynamic Resolution (3)	Exfiltration Over Other Network Medium (1)	Defacement (2)
Domain Trust Discovery	Replication Through Removable Media	Data from Information Repositories (2)	Encrypted Channel (2)	Exfiltration Over Physical Medium (1)	Disk Wipe (2)
File and Directory Discovery	Software Deployment Tools	Data from Local System	Fallback Channels	Exfiltration Over Web Service (2)	Endpoint Denial of Service (4)
Network Service Scanning	Taint Shared Content	Data from Network Shared Drive	Ingress Tool Transfer	Scheduled Transfer	Firmware Corruption
Network Share Discovery	Use Alternate Authentication Material (4)	Data from Removable Media	Multi-Stage Channels	Transfer Data to Cloud Account	Inhibit System Recovery
Network Sniffing		Data Staged (2)	Non-Application Layer Protocol		Network Denial of Service (2)
Password Policy Discovery		Email Collection (3)	Non-Standard Port		Resource Hijacking
Peripheral Device Discovery		Input Capture (4)	Protocol Tunneling		Service Stop
Permission Groups Discovery (2)		Man in the Browser	Proxy (4)		System Shutdown/Reboot
Process Discovery		Man-in-the-Middle (1)	Remote Access Software		
Query Registry		Screen Capture	Traffic Signaling (1)		
Remote System Discovery		Video Capture	Web Service (3)		
Software Discovery (1)					
System Information Discovery					
System Network Configuration Discovery					
System Network Connections Discovery					
System Owner/User Discovery					
System Service Discovery					
System Time Discovery					
Virtualization/Sandbox Evasion (3)					

Figure V.19. The data has been stored on Elasticsearch.

No.	Attack tactic	Description
1	Initial Access	This is the first step for a hacker to get into your network.
2	Execution	After access to your network, attackers will run malicious code to gain more data on your system.
3	Persistence	This step is usually used to maintain the control of the hacker after access to the information system. Such as a backdoor.
4	Privilege Escalation	Hackers try to gain more permission for deeper access sensitive data.
5	Defense Evasion	Attacker avoids being detected.
6	Credential Access	Attacker is trying to steal login credentials such as username and password.
7	Discovery	Adversary is trying to understand network elements. Such as how many computers are in the network?

8	Lateral Movement	The adversary is trying to move through your environment such as, from this computer to another computer.
9	Collection	In this tactic, the attacker is trying to gather data that they are interested in.
10	Command and Control	The adversary is trying to remote control a hacked computer system.
11	Exfiltration	All techniques of this tactic are dangerous, the attacker is trying to steal the data.
12	Impact	Attacker trying to make data corruption. Such as delete or modify valid data and even destroy all systems.

Table V.11. List of all Mitre Att&ck Tatic.

V.4.4.2. Example analyst malware using Mitre Att&ck Technique.

<u>Example 1: Council-of-Europe-.xls[15]:</u>

Site Vmray and VirusTotal analyzed and showed the behavior of this malicious code according to the tree diagram as follows:

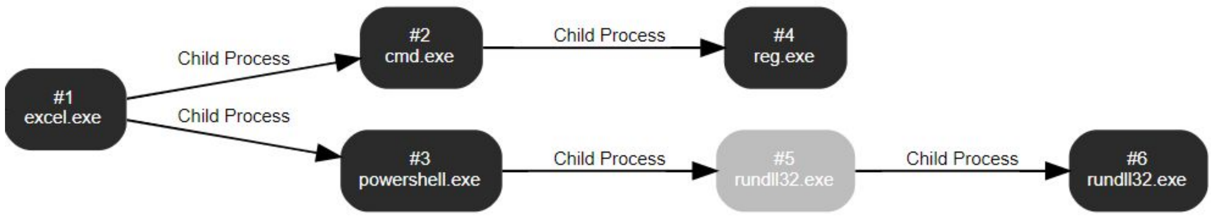
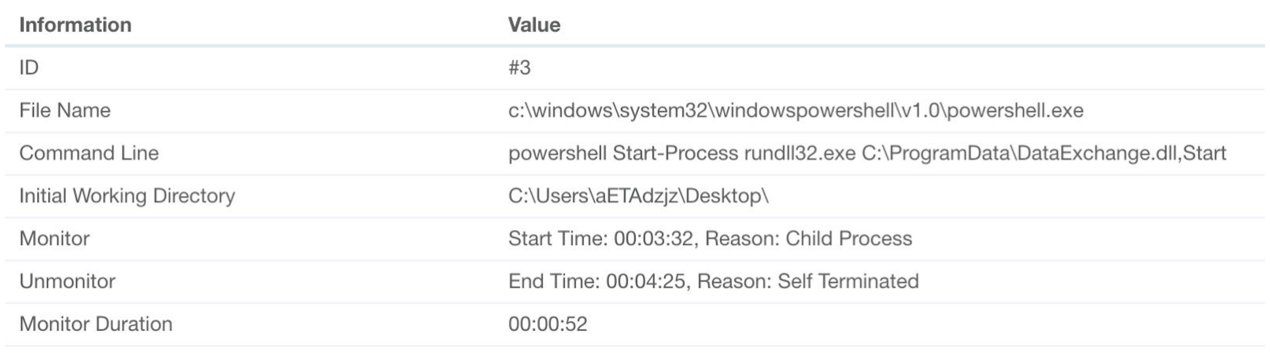


Figure V.20. The tree diagram analysis by Vmray.



Information	Value
ID	#3
File Name	c:\windows\system32\windowspowershell\v1.0\powershell.exe
Command Line	powershell Start-Process rundll32.exe C:\ProgramData\DataExchange.dll,Start
Initial Working Directory	C:\Users\aETAdzjz\Desktop\
Monitor	Start Time: 00:03:32, Reason: Child Process
Unmonitor	End Time: 00:04:25, Reason: Self Terminated
Monitor Duration	00:00:52

Figure V.21. Show detail analysis of the attack by Vmray.

Based on the analysis, we can generalize the behavior of malicious code with the following general tactic: From the open Excel file, the malicious code runs Powershell program with the following command:

```
PowerShell Start-Process rundll32.exe
```

The purpose of that PowerShell command is to run the DataExchange.dll file. The DataExchange.dll file is an operating system library that supports the Dynamic Data Exchange protocol that can help hackers control a remote computer. So this type of attack is under technique:

"*Attack.command and control - Attack.t1071 Sub technique - Attack.t1071.004.*"

In conclusion, this malware signature is:

- Derived from an embedded .xls file with malicious code.
- Launch the *excel.exe* process
- The *excel.exe* runs a PowerShell command to open the Dynamic Data Exchange protocol.

This rule will be built on the history and behavior of this malware, thereby based on the signals contained in the law, which can be detected through the event logs posted by the client when the malware starts.

The signature built-in rule follows:

```
detection:
      selection:
            Image|endswith:
                  - '\powershell.exe'
            ParentImage|endswith:
                  - '\excel.exe'
            CommandLine|contains:
                - 'DataExchange.dll'
      condition: selection
```

Example 2: Detects multiple suspicious processes in a limited timeframe[16]

Some commands are commonly used by adversaries to discover the networks and it is not used usually by normal users. By looking to execute these commands in a short amount of time, we can not only know when a malicious user is on the system but also know what they are doing.

Commands of interest: arp.exe, at.exe, attrib.exe, cscript.exe, dsquery.exe, hostname.exe, ipconfig.exe, mimikatz.exe, nbstat.exe, net.exe, netsh.exe, nslookup.exe, ping.exe, quser.exe, qwinsta.exe, reg.exe, runas.exe, sc.exe, schtasks.exe, ssh.exe, systeminfo.exe, taskkill.exe, telnet.exe, tracert.exe, wscript.exe, xcopy.exe

Based on the blacklists for these commands, a detection rule can be built to warn as follows:

```
Detection:
      Selection:
      CommandLine:
            - arp.exe
            - at.exe
            - attrib.exe
            - … { commands}
      Condition: Selection
```

The construction of the rule can be thought of as a 'collection-rule', which will continue to be evolved to collect more malicious code or attack scripts, in addition to the rules that have been introduced into the system will also be monitored and modified regularly for better, accurate updates.

V.4.4.3. Principle of building unusual behavior database based on Miter Att&ck

Evaluation of sigma rule

Sigma is an open-source project [17], it contains a signature format that can simply describe the associated events log. The rule format itself is very flexible and can be applied to any type of log file. Sigma's primary purpose is to provide a structured forum where researchers and analysts can describe and share detection methods with the broader community. Sigmac is Sigma's rule creation tool, Sigmac converts sigma rules into queries or inputs of the supported targets listed below. It acts as a frontend to the Sigma library that may be used to integrate Sigma support in other projects. On that basis, Sigma is building the ruleset by collecting additional threats based on the integration of MITRE ATT&CK framework identifiers to the ruleset. There are many products currently adopting Sigma as a way for them to create rules for research and development such as MISP (since version 2.4.70, March 2017), SOC Prime - Sigma Rule Editor, uncoder.io - Online Translator for SIEM Searches, THOR - Scan with Sigma rules on endpoints, Joe Sandbox, ypsilon - Automated Use, Case Testing, RANK VASA, TA-Sigma-Searches (Splunk App), TimeSketch. There is an age-old saying one should not reinvent the wheel. Yet, organizations and vendors are constantly doing so. With what Sigma offers and the credibility of this product, in our project, we find the use of Sigma's rule-building approach very appropriate and useful, helping us to shorten the time for building and collecting, thereby analyzing and evaluating unusual processes via event logs sent from the client. Of course, we do not completely reuse, there will be appropriate modifications to cater to our processing program, the rule building structure will be available in the next section.

How to build an unusual behavior profile

Specification

- **Structure**

  The rules consist of a few required sections and several optional ones.

```
title
status [optional]
description [optional]
references [optional]
logsource
   category [optional]
   product [optional]
   service [optional]
   definition [optional]
   ...
detection
   {search-identifier} [optional]
      {string-list} [optional]
      {field: value} [optional]
   ...
   condition
```

```
      fields [optional]
      level [optional]
      tags [optional]
      ...
      [arbitrary custom fields
```

Schema

```
Field name                                Data type
title:
        type:                             str
        length:
            min: 1
            max: 256
description:                              str
references:
        type:                             arr
        contents:                         str
status:
        type:                             any
        of:
            - type:                       str
              value: stable
            - type:                       str
              value: testing
            - type:                       str
              value: experimental
logsource:
        type:                             rec
        optional:
            category:                     str
            product:                      str
            service:                      str
            definition:                   str
detection:
        type:                             rec
        required:
            condition:
                type:                   any
                of:
                    - type:             str
                    - type:             arr
                      contents:         str
fields:
        type:                           arr
        contents:                       str
level:
        type:                           any
        of:
            - type:                     str
              value: low
            - type:                     str
              value: medium
            - type:                     str
              value: high
            - type:                     str
              value: critical
tags:
        type:                           arr
        contents:                       str
```

Components

The following table details the components as well as characteristics.

Elements	**Type**	**Description**
Title	String	A brief title for the rule that should contain what the rules is supposed to detect
Description	String	Short description of malicious rules and practices that can be detected

References	Array	Reference to the source derived from the rule. These could be blog articles, technical articles, presentations, or even tweets.
Status	any	Declare the state of the rule
Log Source	record	This section describes the log data that is applied to detection. It describes the log source, platform, application, and type required when detected.
Detection	record	A set of search-identifiers that represent searches on log data
Condition	any	The condition is the most complex part of the specification and will be subject to change over time and arising requirements. In the first release, it will support the following expressions.
Fields	array	A list of log fields that could be interesting in further analysis of the event and should be displayed to the analyst.
Level	any	The level field contains one of four string values. It describes the criticality of a triggered rule. While low and medium level events have an informative character, events with high and critical levels should lead to immediate reviews by security analysts.
Tags	array	A rule can be classified by tags

Table V.12. Components.

Specific details of each ingredient are as follows:

Title

Attribute: title

A brief title for the rule that should contain what the rules is supposed to detect

Description

Attribute: description

Short description of malicious rules and practices that can be detected

References

Attribute: reference

Reference to the source derived from the rule. These could be blog articles, technical articles, presentations, or even tweets.

Status

Attribute: status

Declare the state of the rule:

- stable: the rule is considered stable and can be used in production systems or control panels.
- test: a quasi-stable rule might require some tweaking.
- experimental: a test rule can lead to false or noisy results, but can also identify events.

Log Source

Attribute: logsource

This section describes the log data that is applied to detection. It describes the log source, platform, application, and type required when detected.

It includes three properties that are automatically evaluated by the converter and some optional elements.

- category - examples: firewall, web, antivirus
- product - examples: windows, apache, checkpoint fw1
- service - examples: sshd, applocker

The "category" value is used to select all log files that are written by a certain product group, such as a firewall or web server logs. Automatic conversions use the keyword as a selector for multiple metrics.

The "product" value is used to select all log outputs of a certain product, e.g. all Windows Eventlog types including "Security", "System", "Application" and the new log types like "AppLocker" and "Windows Defender".

Use the "service" value to select only a subset of the product's log, such as "sshd" on Linux or the "Security" event log on Windows systems.

The "definition" can be used to describe the log source, including some information about the granularity of the log or the configurations that must be applied. It is not evaluated automatically by the converter but gives helpful advice to the reader on how to configure the source to deliver the necessary events used in detection.

can use the values of 'categories,' products', and 'services' to point converters to a given index. You can specify a configuration file that the category 'firewall' converts to (index = fw1 * OR index = asa *) during a Splunk search conversion or a 'windows' product converts to "_index" : "logstash-windows *" in ElasticSearch query.

Instead of referring to specific services, generic log sources can be used, for example:

- category: process_creation
- product: windows

Instead of a definition of multiple rules for Sysmon, Windows Security Auditing, and possible product-specific rules.

Detection

Attribute: detection

A set of search-identifiers that represent searches on log data

- ***Search-Identifier:*** A definition that can consist of two different data structures - lists and maps
- ***General:***
    - All values are treated as case-insensitive strings
    - Can use wildcard characters '*' and '?' in strings
    - Wildcards can be escaped with \, e.g. *. If some wildcard after a backslash should be searched, the backslash has to be escaped: \\*.
    - Regular expressions are case-sensitive by default

- ***Lists:*** The list contains strings that apply to the full log message, and are reasonably linked with 'OR', for example:

  Match on 'EvilService' **OR** 'svchost.exe -n evil':

```
Detection:
      Keywords:
            - EvilService
            - svchost.exe -n evil
```

- ***Maps:*** Maps (or dictionaries) consist of key/value pairs, in which the key is a field in the log data and the value of a string or integer value. Lists of maps are joined with a logical 'OR'. All elements of a map are joined with a logical 'AND', for example:

  Matches on `EventLog` 'Security' **AND** (`EventID` 'X' **OR** `EventID` 'Y'):

```
Detection:
      Selection:
            - EventLog: Security
      EventID:
            - X
            - Y
condition: selection
```

  Matches on `EventLog` 'Security' **AND** `EventID` 'X' **AND** `TicketOptions` 0x40810000 **AND** `TicketEncryption` 0x17:

```
Detection:
      Selection:
            - EventLog: Security
      EventID: X
      TicketOptions: '0x40810000'
      TicketEncryption: '0x17'
condition: selection
```

- ***Special Field Values:*** There are special field values that can be used.

  - An empty value is defined with `''`
  - A null value is defined with `null`

OBSOLETE: An arbitrary value except null or null cannot be specified with non-null anymore

The application of these values depends on the target SIEM system.

To get an expression that says is not null, you must create another selection and negate it in the condition.

```
detection:
      selection:
            EventID: 4738
      Filter:
            PasswordLastSet: null
condition: selection and not filter
```

- ***Value Modifiers:*** Values contained in a rule can be modified with value modifiers. The value modifier is added after the field name with the pipe character | as a separator and can also be strung, for example, the field name | mod1 | mod2: value. The value modifier is applied in the order given the value.
- ***Currently Available Modifiers***

  - **endswith**: The value is expected at the end of the field's content (replaces e.g. '*\cmd.exe')
  - **startswith**: The value is expected at the beginning of the field's content. (replaces e.g. 'adm*')

Condition

Attribute: condition

The condition is the most complex part of the specification and will be subject to change over time and arising requirements. In the first release, it will support the following expressions.

- Logical **AND/OR**

```
keywords1 or keywords2
```

- 1/all of search-identifier

  Same as just 'keywords' if keywords are defined in a list. X may be:

  - 1 (logical or across alternatives)
  - All (logical and across alternatives)

  Example: ‘**all of the keywords**’ mean that all items of the list keywords must appear, instead of the default behavior of any of the listed items.

- 1/all of them

  Logical OR (1 of them) or AND (all of them) across all defined search identifiers. The search identifiers themselves are logically linked with their default behavior for maps (AND) and lists (OR).

  Example: 1 of them means that one of the defined search identifiers must appear.

- 1/all of search-identifier-pattern

  Same as *1/all of them*, but restricted to matching search identifiers. Matching is done with * wildcards (any number of characters) at arbitrary positions in the pattern.

  Examples:

  ```
  1 of selection* and keywords
  any of selection* and not filters
  ```

- Negation with 'not'

  ```
  keywords and not filters
  ```

- Brackets

  ```
  selection1 and (keywords1 or keywords2)
  ```

- Operator Precedence (least to most binding)

  - `or`

- `and`
- `not`
- `x of search-identifier`

If multiple conditions are given, they are logically linked with OR.

Fields

Attribute: fields

A list of log fields that could be interesting in further analysis of the event and should be displayed to the analyst.

Level

Attribute: level

The level field contains one of four string values. It describes the criticality of a triggered rule. While low and medium level events have an informative character, events with high and critical levels should lead to immediate reviews by security analysts.

- **low**: Interesting event but rarely an incident. Low events are relevant in high numbers or combined with others. A security analyst has to review the events and identify anomalies or suspicious indicators. Use them in a dashboard panel, e.g. in the form of a chart.
- **medium**: The relevant event that should be reviewed manually on a more frequent basis. A security analyst has to review the events and identify anomalies or suspicious indicators. List the events in a dashboard panel for manual review.
- **high**: The relevant event that should trigger an internal alert and requires a prompt review.
- **critical**: The highly relevant event indicates an incident. We recommend critical events for immediate response actions and external notifications (E-Mail, Ticket).

Tags

Attribute: tags

A rule can be classified by tags. Tags should generally conform to the following syntax:

- Character set: lower case, underscore and hyphen
- There are no spaces
- Cards have a namespace, dots are used as separators.

  For example. `attack.t1234` refers to technique 1234 in the namespace attack; Namespaces can also be nested

V.4.4.4. Implementation

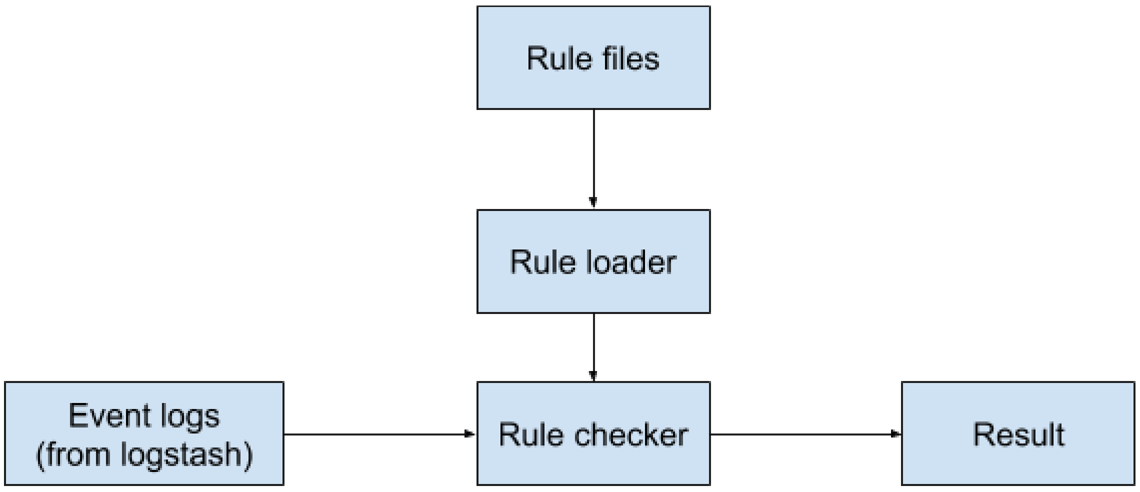


Figure V.22. Analyst behavior module service.

For each posted Event logs, the detailed content of this event logs will be compared with each rule in the database we built earlier and returned the results immediately and stored in Elasticsearch. For event logs that hit the rule, all the detected rule details will be stored and labeled with that event log information.

```
Function rule_checker():
      Give message event logs from client
      Parse and format message
      Try:
            For each rule in the database:
                  if rule check with message is True:
                        Get rule information
                        Call api send notification to dashboard
                        returns the labeled message and rule information
                  else:
                        returns the labeled message is not hit rule
```

About handling messages with the rule, the information provided by the client includes all the details of the process such as EventID, Image, SourceImage, TargetImage, StartModule, StartFunction, CommandLine ... a lot of information is Sent from Symon client, we will check and compare the contents of each field from the message with the characteristics of each rule, if the fields of message meet the rule's conditions, we can conclude that this process has hit the rule.

In each rule, there is a field 'detection' with the contents of the recorded script of an attack and the condition to investigate. Start with parsing each key_field within 'detection' by comparing the fields within the 'logs message'.

```
Function check():
      Get value in field ‘detection’ in file rule
      Get selection in field ‘detection’
      Check selections is true or false
      return result from compare
```

For each content inside the key and value, we need to extract each key and value inside to know the value of the key is OR (list) or AND(dictionaries) .

We first define the return functions that test the condition OR and And.

```
Function or_operator(operands):
      if false in operands:

            return false

      return True

Function and_operator(operands):
      if True in operands:

            return True

      return False
```

With what we defined about how it works inside the 'detection' field we need to deal with going to have to fetch the leaves of each tree then check that outside the branches will be an OR (list) or AND(dictionaries) condition, continue loop test until you reach the first branch.

```
Function detection_operator():
      selections = {}
      For each key, value is in detection field:
            if key is not ‘condition filed’
                  result = Check selection_operator is True or False
                  Put the result into a dictionary selections[key] = result
            else:
                    Get condition
        For each key, value in selections:
                if key in condition:
                        Alter key in condition is value of each selection
                Return condition
```

selection_operator function will be responsible for checking whether the selection is True or False and returning the result to check against the condition, the content inside the selection is a series of conditions according to the rule if dictionaries - condition AND, If the list - condition OR (see section V.4.4.3. How to build an unusual behavior profile)

```
Function selection_operator():
        If selection is lits:
                list_selection = []
                For each element inside the selection:
                        result = selection_operator(element)
                        //Recursively call function A again until get the result
from the leaf
                        Add result to list_selection
                        Return check or_operator(result)

        If selection is dictionaries:
                list_selection = []
                For each element inside the selection:
                        If key is 'condition field':
                                continue
                        If value is list
                                result is return of check_many(key, value)
                                Add result to list_selection
                        else :
                                result is return of check_sigle(key, value)
                                Add result to list_selection
                        Return check and_operator(result )
```

This function executes in case a key has a list of many different values, in order to be able to check, you need to take each value inside and check with each value in the field of the log message, the check_many function will get Each value entered into the check_single function waits for the return result, as mentioned in the form of a key with multiple values, the or_operator function will perform the test with the OR condition.

```
Function check_many(key, values)
      List_value_check = []
      For each value inside the values:
            result is return of check_sigle(key, value)
            Add result to List_value_check
      Return check or_operator(result)
```

The check_single function references each value in the rule in turn with the reference value having the same key as the field from the message, each key has a way to compare the value from the log message like *contains*, *endwith*, *startswith* or nothing, the test and return results are done in the simple_operator function

```
Function check_sigle(key, value)
      elements = number return by split key with “|”
      field = element[0]
      If elements == 1:
            Return simple_operator(message[field], value, “=”)
      condition = elements[1]
      Return simple_operator(message[field], value, condition)

Function simple_operator(left, right, operator)
      If operator is “=”:
            If operator is String:
                  if appears "*" at the beginning and end:
                        simple_operator(left, right[1:-1], "contains")
                  if appears "*" at the end:
                        simple_operator(left, right[1:], "startswith")
                  if appears "*" at the beginning:
                        simple_operator(left, right[1:], "endswith")
      If left is None:
            Return False
      If operator is “endswith”:
            Return left endswith right
      if operator is “startswith”:
            Return left startswith right
```

```
    if operator is “contains”:
          Right in Left
```

V.5. Data Management And Data Storage Systems

V.5.1. Introduction ElasticSearch

Elasticsearch

Elasticsearch is an open-source, distributed search, and analysis tool for all types of data, including text, numbers, geospatial, structured, and unstructured. Elasticsearch is built on top of Apache Lucene and was first released in 2010 by Elasticsearch N.V. (now known as Elastic). Known for its simple REST APIs, distributed nature, speed, and scalability, Elasticsearch is the central component of Elastic Stack, an open-source toolkit for importing, enriching, storing, and distributing. data analysis and visualization. Commonly known as the ELK Stack (after Elasticsearch, Logstash, and Kibana), Elastic Stack now includes a rich collection of light shipping agents called Beats to send data to Elasticsearch[18].

Elasticsearch benefits

Elasticsearch is fast. Since Elasticsearch is based on Lucene, it excels at full-text searches. Elasticsearch is also a near real-time search platform, which means the delay between the document being indexed until it becomes searchable is very short - usually one second. Hence, Elasticsearch is well-suited for time-sensitive use cases like security analysis and infrastructure monitoring.

Elasticsearch is distributed naturally. Documents stored in Elasticsearch are distributed on different containers called segments, which are copied to provide redundant copies of the data in the event of a hardware failure. Elasticsearch's decentralized nature allows it to scale to hundreds (or even thousands) of servers and process petabytes of data.

Elasticsearch comes with a wide range of features. In addition to speed, scalability, and resilience, Elasticsearch has a number of powerful built-in features that make data more efficient for data storage and retrieval, such as scrolling and index lifecycle management.

Elastic Stack simplifies data entry, display, and reporting. Integration with Beats and Logstash makes it easy to process data before, being indexed into Elasticsearch. And Kibana provides real-time visualization of the Elasticsearch data as well as a user interface for quickly accessing application performance monitoring (APM) data, logs, and infrastructure metrics.

Setup and configuration

(Setup for Ubuntu OS)

Install dependencies:

Elasticsearch depends on Java. Issue the following commands to install the dependency:

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer -y
```

To install Elasticsearch, issue the following commands:

```
wget
https://artifacts.elastic.co/downloads/elasticsearch/elasticsear
ch-7.7.1.deb
sudo dpkg -i elasticsearch-7.7.1.deb
```

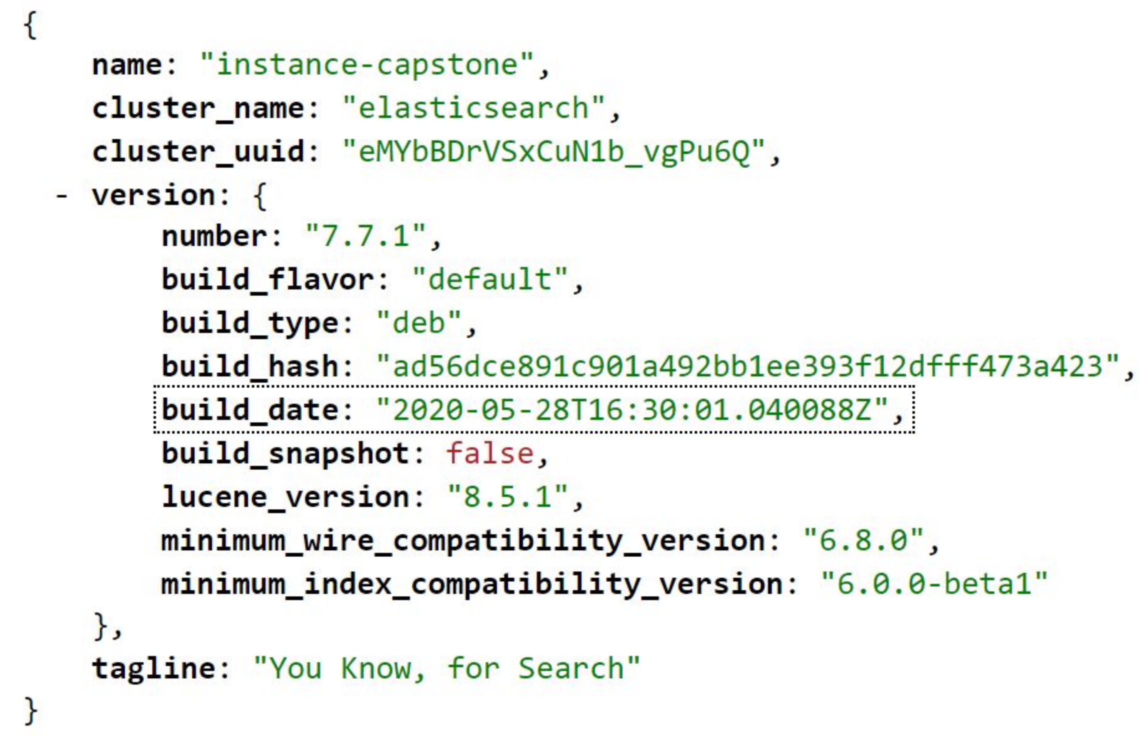
Open the Elasticsearch configuration file with the command:

```
sudo nano /etc/elasticsearch/elasticsearch.yml
```

Finally, start and enable the service with the command:

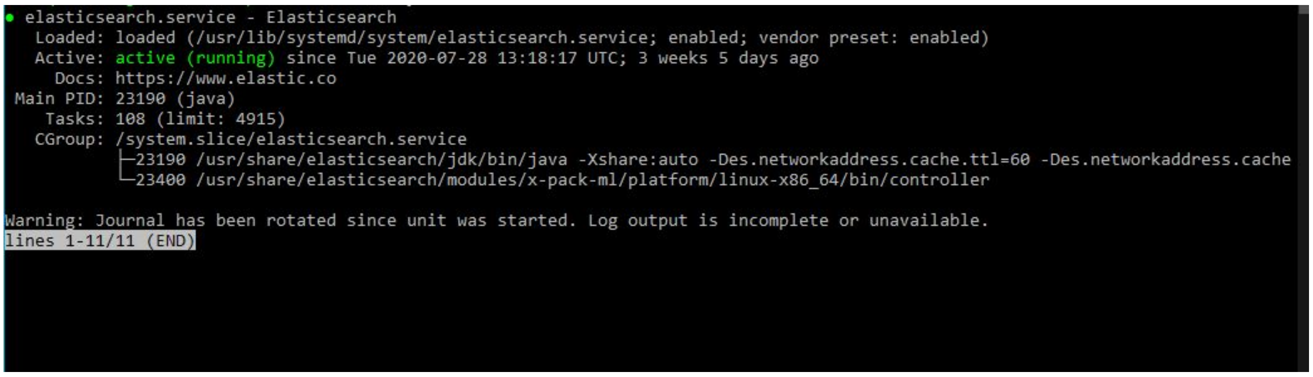
```
sudo systemctl enable elasticsearch.service
sudo systemctl start elasticsearch.service
```

Point a web browser to `http://SERVER_IP:9200/_cat/health?v` (where SERVER_IP is the IP address of my hosting server):



```
{
    name: "instance-capstone",
    cluster_name: "elasticsearch",
    cluster_uuid: "eMYbBDrVSxCuN1b_vgPu6Q",
  - version: {
        number: "7.7.1",
        build_flavor: "default",
        build_type: "deb",
        build_hash: "ad56dce891c901a492bb1ee393f12dfff473a423",
        build_date: "2020-05-28T16:30:01.040088Z",
        build_snapshot: false,
        lucene_version: "8.5.1",
        minimum_wire_compatibility_version: "6.8.0",
        minimum_index_compatibility_version: "6.0.0-beta1"
    },
    tagline: "You Know, for Search"
}
```

Check service working:



```
● elasticsearch.service - Elasticsearch
   Loaded: loaded (/usr/lib/systemd/system/elasticsearch.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2020-07-28 13:18:17 UTC; 3 weeks 5 days ago
     Docs: https://www.elastic.co
 Main PID: 23190 (java)
    Tasks: 108 (limit: 4915)
   CGroup: /system.slice/elasticsearch.service
           ├─23190 /usr/share/elasticsearch/jdk/bin/java -Xshare:auto -Des.networkaddress.cache.ttl=60 -Des.networkaddress.cache
           └─23400 /usr/share/elasticsearch/modules/x-pack-ml/platform/linux-x86_64/bin/controller

Warning: Journal has been rotated since unit was started. Log output is incomplete or unavailable.
lines 1-11/11 (END)
```

V.5.2. Introduction Kibana

Kibana

(Kibana is used to manage data and to assist with search queries)
Kibana is an open-source user interface application on Elastic Stack that provides data search and visualization for indexed data in Elasticsearch. Often referred to as a charting tool for Elastic Stack (formerly known as ELK Stack after Elasticsearch, Logstash, and Kibana), Kibana also acts as a user interface for monitoring, managing, and securing an Elastic Stack cluster - as well as a centralized hub for integrated solutions developed on Elastic Stack. Developed

in 2013 from within the Elasticsearch community, Kibana has grown to be the window to Elastic Stack itself, providing a portal for users and companies.

Kibana benefits

Kibana's tight integration with the larger Elasticsearch and Elastic Stack makes it ideal for supporting the following:

Find, view, and visualize indexed data in Elasticsearch, and analyze data through the creation of bar charts, pie charts, tables, charts, and maps. Dashboard view that combines these visual elements is then shared across the browser to provide a real-time analytic view into large volumes of data to support use cases such as:

- Logging and diary analysis
- Infrastructure index and container monitoring
- Application performance monitoring (APM)
- Analyze and display geospatial data
- Security analysis
- Business analysis

Monitor, manage and secure an Elastic Stack instance through a web interface.
Centralize access to integrated solutions developed on Elastic Stack for visibility, security, and enterprise search applications.

Setup and configuration

Install the Kibana Dashboard, which can display the results of Elasticsearch.
This is done with the following steps:

```
wget https://artifacts.elastic.co/downloads/kibana/kibana-7.7.1-amd64.deb
sudo dpkg -i kibana-7.7.1-amd64.deb
```

Configure Kibana by opening the configuration file with the command:

```
sudo nano /etc/kibana/kibana.yml
```

Locate the following lines:

```
#server.host: "localhost"
#elasticsearch.url: "http://localhost:9200"
```

Change those lines to:

```
server.host: "SERVER_IP"
elasticsearch.url: "http://SERVER_IP:9200"
```

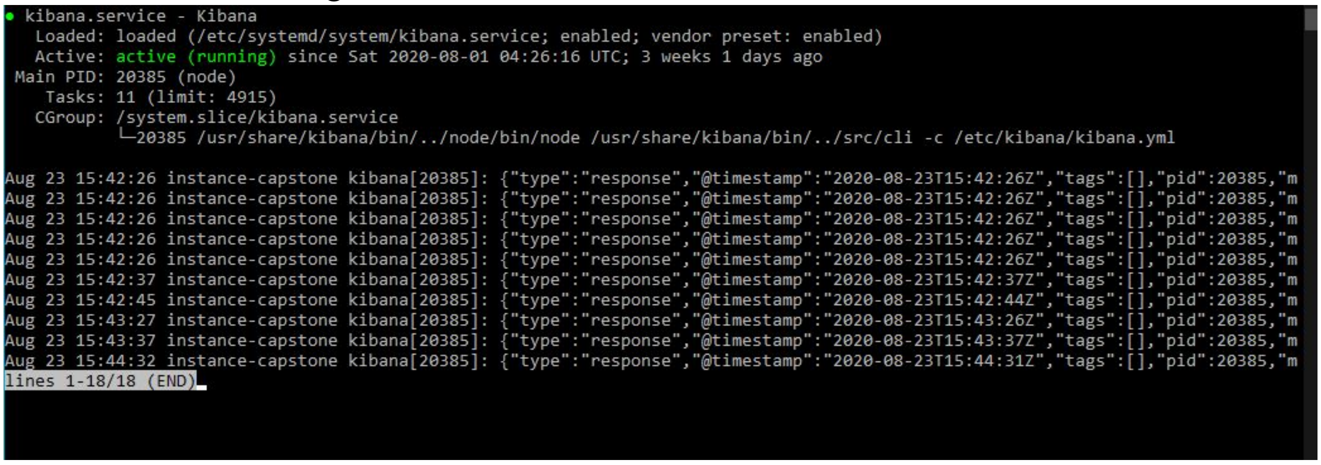
Where SERVER_IP is the IP address of the hosting server. Save and close that file.
Finally, issue the following command:

```
sudo sysctl -w vm.max_map_count=262144
```

Reboot the server. Once the server reboots, start and enable the Kibana service with the commands:

```
sudo systemctl enable kibana.service
sudo systemctl start kibana.service
```

Check service working:



```
● kibana.service - Kibana
   Loaded: loaded (/etc/systemd/system/kibana.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2020-08-01 04:26:16 UTC; 3 weeks 1 days ago
 Main PID: 20385 (node)
    Tasks: 11 (limit: 4915)
   CGroup: /system.slice/kibana.service
           └─20385 /usr/share/kibana/bin/../node/bin/node /usr/share/kibana/bin/../src/cli -c /etc/kibana/kibana.yml

Aug 23 15:42:26 instance-capstone kibana[20385]: {"type":"response","@timestamp":"2020-08-23T15:42:26Z","tags":[],"pid":20385,"m
Aug 23 15:42:26 instance-capstone kibana[20385]: {"type":"response","@timestamp":"2020-08-23T15:42:26Z","tags":[],"pid":20385,"m
Aug 23 15:42:26 instance-capstone kibana[20385]: {"type":"response","@timestamp":"2020-08-23T15:42:26Z","tags":[],"pid":20385,"m
Aug 23 15:42:26 instance-capstone kibana[20385]: {"type":"response","@timestamp":"2020-08-23T15:42:26Z","tags":[],"pid":20385,"m
Aug 23 15:42:26 instance-capstone kibana[20385]: {"type":"response","@timestamp":"2020-08-23T15:42:26Z","tags":[],"pid":20385,"m
Aug 23 15:42:37 instance-capstone kibana[20385]: {"type":"response","@timestamp":"2020-08-23T15:42:37Z","tags":[],"pid":20385,"m
Aug 23 15:42:45 instance-capstone kibana[20385]: {"type":"response","@timestamp":"2020-08-23T15:42:44Z","tags":[],"pid":20385,"m
Aug 23 15:43:27 instance-capstone kibana[20385]: {"type":"response","@timestamp":"2020-08-23T15:43:26Z","tags":[],"pid":20385,"m
Aug 23 15:43:37 instance-capstone kibana[20385]: {"type":"response","@timestamp":"2020-08-23T15:43:37Z","tags":[],"pid":20385,"m
Aug 23 15:44:32 instance-capstone kibana[20385]: {"type":"response","@timestamp":"2020-08-23T15:44:31Z","tags":[],"pid":20385,"m
lines 1-18/18 (END)
```

V.5.3. Introduction Logstash

Logstash

Logstash is a lightweight, open-source, server-side data processing pipeline that allows you to collect data from a variety of sources, quickly convert it, and send it to the destination you want. It is often used as the data pathway for Elasticsearch, an open-source search, and analysis tool. Due to its tight integration with Elasticsearch, its robust log processing capabilities, and over 200 pre-made open-source plugins that can make it easy to index your data, Logstash is a popular choice for downloading data. materials to Elasticsearch.

Logstash benefits

Easily load unstructured data

Logstash allows you to easily import unstructured data from a variety of data sources including system logs, website logs, and application server logs.

Pre-built filters

Logstash provides pre-built filters, so you can easily transform common data types, index them in Elasticsearch, and start queries without having to build data transformation pipes. custom.

Flexible plugin architecture

With over 200 plugins already on Github, chances are that someone has already made the plugin you need to customize your data path. But if none of the available ones fit your requirements, you can easily create one yourself.

Setup and configuration

Need the means to add data into Elasticsearch. This will be done with Logstash. To install this tool, issue the commands:

```
wget
https://artifacts.elastic.co/downloads/logstash/logstash-6.3.2.d
eb
sudo dpkg -i logstash-6.3.2.deb
```

Open the Logstash configuration file with the command:

```
sudo nano /etc/logstash/logstash.yml
```

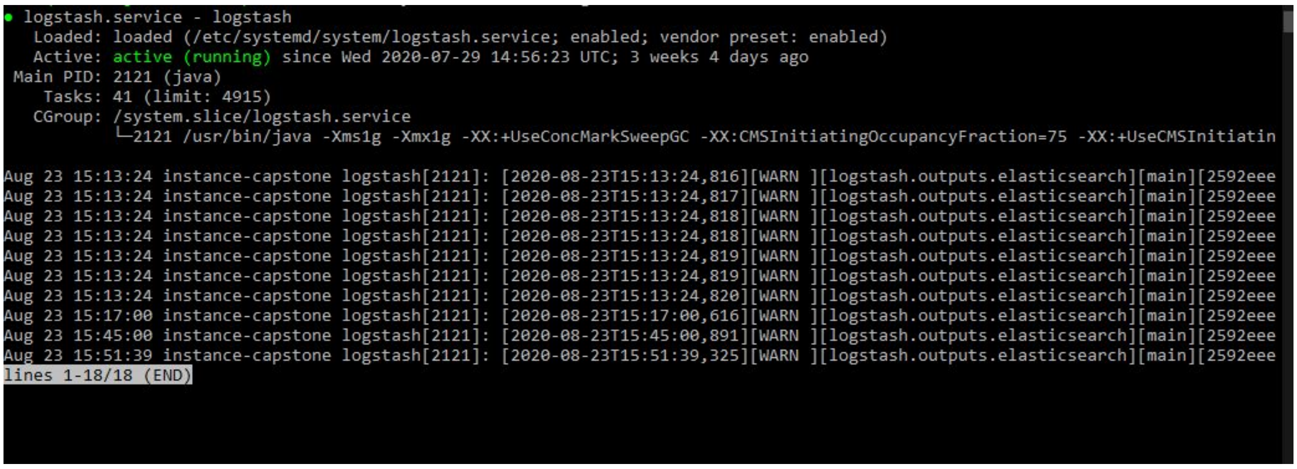
Change the following line:

```
# http.host: "127.0.0.1"
```

Remove the # character and change the IP address hosting server IP. Save and close that file.
Start and enable the Logstash service with the commands:

```
sudo systemctl enable logstash.service
sudo systemctl start logstash.service
```

Point browser to http://SERVER_IP:5601 and ready to begin working.
Check service woking:



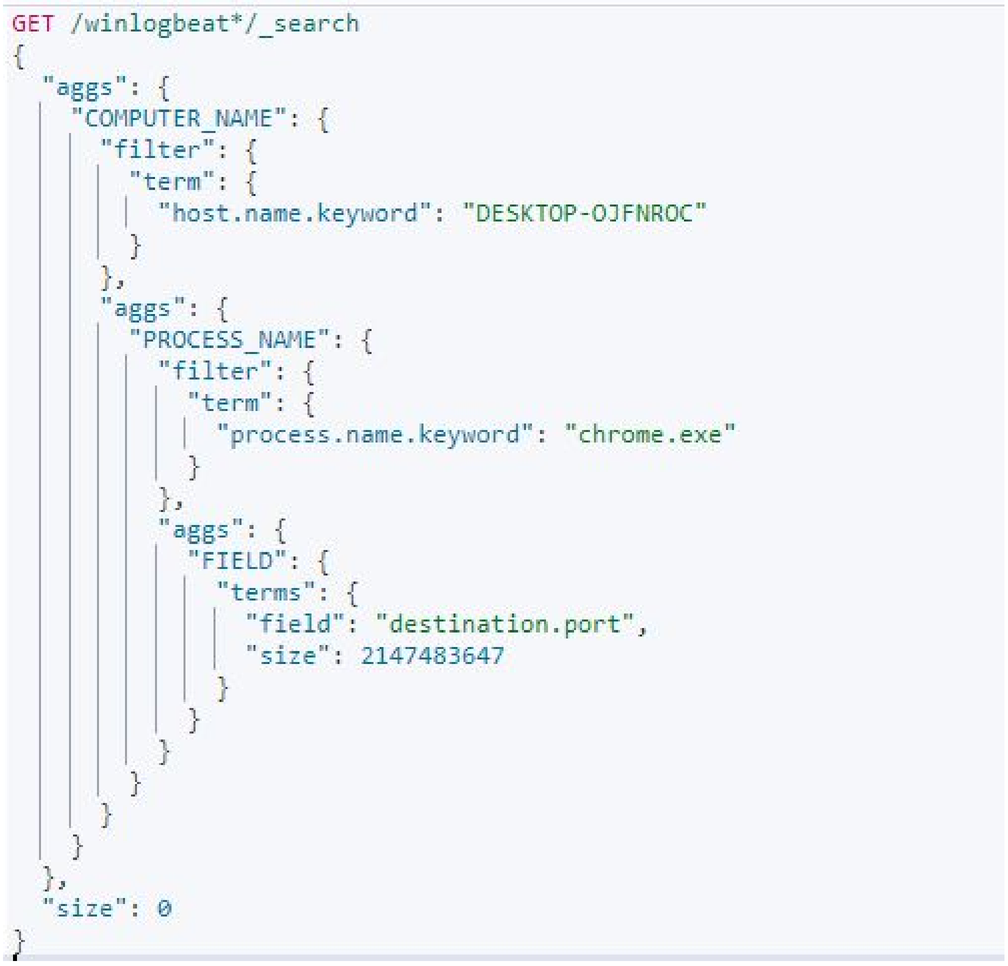
```
● logstash.service - logstash
   Loaded: loaded (/etc/systemd/system/logstash.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2020-07-29 14:56:23 UTC; 3 weeks 4 days ago
 Main PID: 2121 (java)
    Tasks: 41 (limit: 4915)
   CGroup: /system.slice/logstash.service
           └─2121 /usr/bin/java -Xms1g -Xmx1g -XX:+UseConcMarkSweepGC -XX:CMSInitiatingOccupancyFraction=75 -XX:+UseCMSInitiatin

Aug 23 15:13:24 instance-capstone logstash[2121]: [2020-08-23T15:13:24,816][WARN ][logstash.outputs.elasticsearch][main][2592eee
Aug 23 15:13:24 instance-capstone logstash[2121]: [2020-08-23T15:13:24,817][WARN ][logstash.outputs.elasticsearch][main][2592eee
Aug 23 15:13:24 instance-capstone logstash[2121]: [2020-08-23T15:13:24,818][WARN ][logstash.outputs.elasticsearch][main][2592eee
Aug 23 15:13:24 instance-capstone logstash[2121]: [2020-08-23T15:13:24,818][WARN ][logstash.outputs.elasticsearch][main][2592eee
Aug 23 15:13:24 instance-capstone logstash[2121]: [2020-08-23T15:13:24,819][WARN ][logstash.outputs.elasticsearch][main][2592eee
Aug 23 15:13:24 instance-capstone logstash[2121]: [2020-08-23T15:13:24,819][WARN ][logstash.outputs.elasticsearch][main][2592eee
Aug 23 15:13:24 instance-capstone logstash[2121]: [2020-08-23T15:13:24,820][WARN ][logstash.outputs.elasticsearch][main][2592eee
Aug 23 15:17:00 instance-capstone logstash[2121]: [2020-08-23T15:17:00,616][WARN ][logstash.outputs.elasticsearch][main][2592eee
Aug 23 15:45:00 instance-capstone logstash[2121]: [2020-08-23T15:45:00,891][WARN ][logstash.outputs.elasticsearch][main][2592eee
Aug 23 15:51:39 instance-capstone logstash[2121]: [2020-08-23T15:51:39,325][WARN ][logstash.outputs.elasticsearch][main][2592eee
lines 1-18/18 (END)
```

V.5.4. Query anomaly behaviors.

The Multi-Search service provides querying APIs to retrieve data from Elasticsearch. Query ES structure for the APIs domain, IP, port

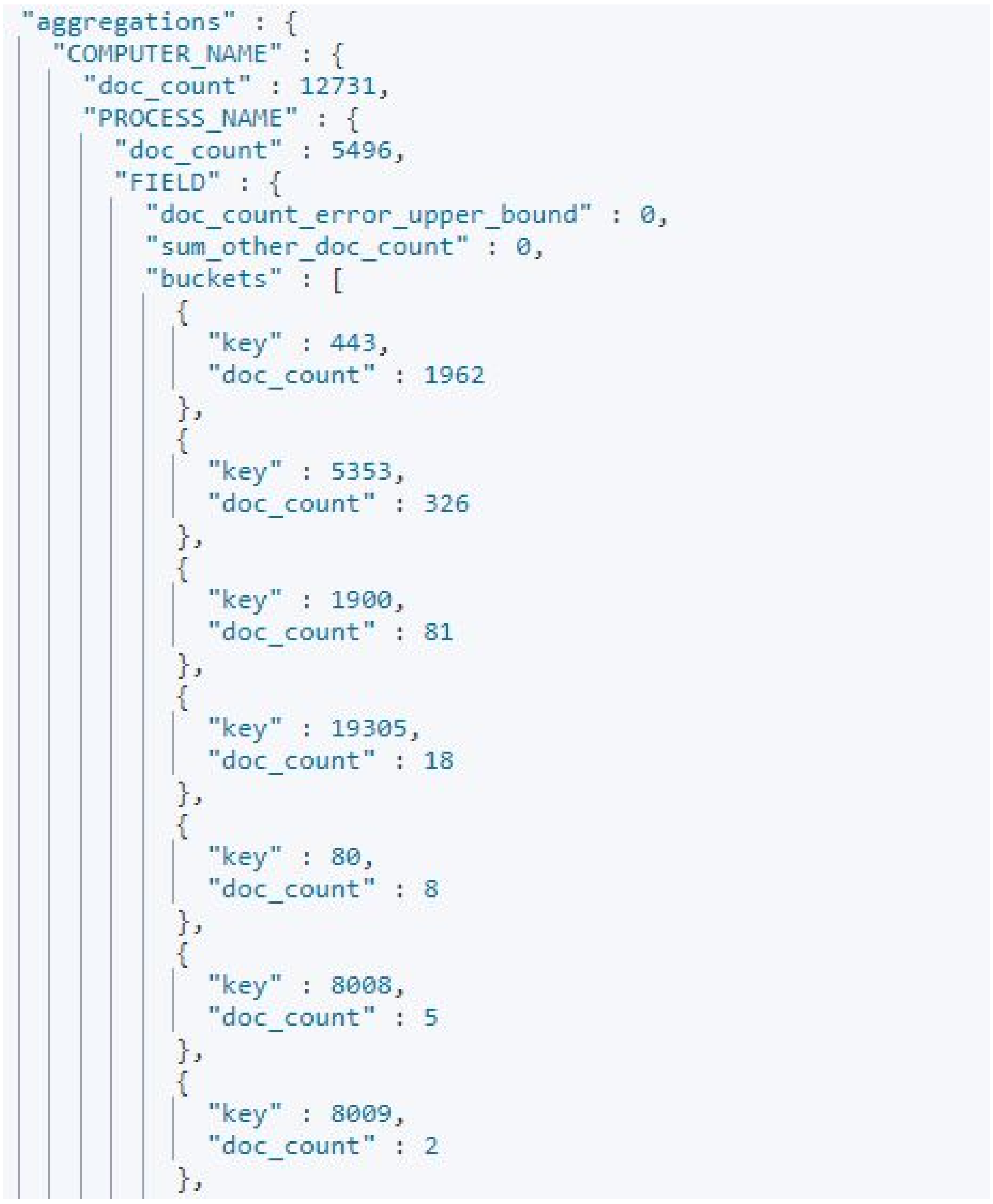
```
"aggs": {
  "COMPUTER_NAME": {
    "filter": {
      "term": {
        "host.name.keyword": ""
      }
    },
    "aggs": {
      "PROCESS_NAME": {
        "filter": {
          "term": {
            "process.name.keyword": ""
          }
        },
        "aggs": {
          "FIELD": {
            "terms": {
              "field": "",
              "size": 2147483647
    ...
"size": 0
```

The query performs a search all by hostname, process name, and keyword, also, to match the period time the user wants to find. With the above query, you can set up options for you to send requests from the client to return the corresponding data. Use Dev Tools in Kibana query:



```
GET /winlogbeat*/_search
{
  "aggs": {
    "COMPUTER_NAME": {
      "filter": {
        "term": {
          "host.name.keyword": "DESKTOP-OJFNROC"
        }
      },
      "aggs": {
        "PROCESS_NAME": {
          "filter": {
            "term": {
              "process.name.keyword": "chrome.exe"
            }
          },
          "aggs": {
            "FIELD": {
              "terms": {
                "field": "destination.port",
                "size": 2147483647
              }
            }
          }
        }
      }
    }
  },
  "size": 0
}
```

Query with params ‘computer name’, ‘process name’, ‘field’, and the returned result is information including the number of logs sent to the computer, the number of logs sent from the process, and detailed information about the number of times of each port.



```
"aggregations" : {
  "COMPUTER_NAME" : {
    "doc_count" : 12731,
    "PROCESS_NAME" : {
      "doc_count" : 5496,
      "FIELD" : {
        "doc_count_error_upper_bound" : 0,
        "sum_other_doc_count" : 0,
        "buckets" : [
          {
            "key" : 443,
            "doc_count" : 1962
          },
          {
            "key" : 5353,
            "doc_count" : 326
          },
          {
            "key" : 1900,
            "doc_count" : 81
          },
          {
            "key" : 19305,
            "doc_count" : 18
          },
          {
            "key" : 80,
            "doc_count" : 8
          },
          {
            "key" : 8008,
            "doc_count" : 5
          },
          {
            "key" : 8009,
            "doc_count" : 2
          },
```

In addition, we have support for queries at intervals as follows:

```
{ "query": {
    "bool": {
      "filter": {
        "range": {
          "@timestamp": {
            "gte": "",
            "lte": ""

      ...
  },
  "aggs": {
    "COMPUTER_NAME": {
      "filter": {
        "term": {
          "host.name.keyword": ""
        }
      },
      "aggs": {
        "PROCESS_NAME": {
          "filter": {
            "term": {
              "process.name.keyword": ""
            }
          },
          "aggs": {
            "myDateHistogram": {
              "date_histogram": {
                "field": "@timestamp",
                "time_zone": "GMT+7",
                "calendar_interval": "1m",
                "format": "yyyy-MM-dd-HH:mm"
              },
              "aggs": {
                "FIELD": {
                  "terms": {
                    "field": "",
                    "size": 2147483647
      }
     "size": 0}
```

The result of the query is information about the number of logs posted, of the computer, the number of logs posted by the process, and the field information you want to statistic, statistical information will be calculated per minute.

<u>API count domain</u>

This API will return a list of domains visited by a computer during a custom time period, including information about name, several times accessed during that time.

URL	Method	Parameter	Result

/_domain	- GET	- computer_name
		- process_name
		.
			`"result": {`
			`  "COMPUTER_NAME": {`
			`    "PROCESS_NAME": {`
			`      "FIELD": {`
			`        "buckets": [{`
			`         "doc_count": count_number,`
			`          "Key": "domain"`
			`        }]`

Table V.13. API count domain.

```
@app.route('/_domain', methods=['GET'])
def GET_DOMAIN_COUNT():
    try:
        json_data = request.get_json(force=True)
        computer_name = json_data['computer_name']
        process_name = json_data['process_name']
        response = elashticsearch_query(computer_name, process_name,
LIST_FIELDS[0])
        return jsonify({
            'result': response["aggregations"]
        })
    except:
        return jsonify({
            'result': 'something went wrong, check your params again'
        })
```

<u>API count IP</u>

This API will return a list of IP visited by a computer during a custom time period, including information about name, a number of times accessed during that time.

URL	Method	Parameter	Result

/_ip	- GET	- computer_name
		- process_name
		
		.
			`"result": {`
			`  "COMPUTER_NAME": {`
			`    "PROCESS_NAME": {`
			`       "FIELD": {`
			`         "buckets": [{`
			`          "doc_count":`
			`count_number,`
			`           "Key": "ip"`
			`          }]`

Table V.14. API count IP.

```
@app.route('/_ip', methods=['GET'])
def GET_IP_COUNT():
    try:
        json_data = request.get_json(force=True)
        computer_name = json_data['computer_name']
        process_name = json_data['process_name']
        response = elashticsearch_query(
        computer_name, process_name, LIST_FIELDS[2])
        return jsonify({
            'result': response["aggregations"]
        })
    except expression as identifier:
        return jsonify({
            'result': 'something went wrong, check your params again'
        })
```

API count Port

This API will return a list of Port visited by a computer during a custom time period, including information about name, a number of times accessed during that time.

URL	Method	Parameter	Result

/_port	- GET	- computer_name
		- process_name
		.
			`"result": {`
			`  "COMPUTER_NAME": {`
			`    "PROCESS_NAME": {`
			`      "FIELD": {`
			`        "buckets": [{`
			`          "doc_count":`
			`count_number,`
			`          "Key": "port"`
			`        }]`

Table V.15. API count Port.

```
@app.route('/_port', methods=['GET'])
def GET_PORT_COUNT():
    try:
        json_data = request.get_json(force=True)
        computer_name = json_data['computer_name']
        process_name = json_data['process_name']
        response = elashticsearch_query(computer_name, process_name,
LIST_FIELDS[1])
        return jsonify({
            'result': response["aggregations"]
        })
    except expression as identifier:
        return jsonify({
            'result': 'something went wrong, check your params again'
        })
```

The domain, IP, and port APIs call the function 'elashticsearch_query', this function will use the input parameters to fill in values that match each option of each API, the options defined in a FIELDS list contain The keyword references Elasticsearch to query the data:

```
LIST_FIELDS = [
    'dns.question.name.keyword',
    'destination.port',
    'destination.ip.keyword',
    'event.code'
]
```

This is perfectly suited to the original design that can accommodate multiple APIs with given keywords. Therefore, only one handler function is needed:

```
def elashticsearch_query(computer_name, process_name, field):
      q = query.QUERY_INFOR_DOMAIN_IP_PORT

      q['aggs']['COMPUTER_NAME']['filter']['term']['host.name.keyword'] =
      computer_name
      q['aggs']['COMPUTER_NAME']['aggs']['PROCESS_NAME']['filter']['term']['process.name.keyword'] = process_name
      q['aggs']['COMPUTER_NAME']['aggs']['PROCESS_NAME']['aggs']['FIELD']['terms']['field'] = field

      res = es.search(index="winlogbeat*", body=q)
      return res
```

This function will fill the parameters and execute the query on the Elasticsearch and return results.

API get data by GUID

Query ES structure:

```
{
  "aggs": {
    "GUID": {
      "filter": {
        "term": {
          "process.entity_id.keyword": ""
        }
      },
      "aggs": {
        "FIELD": {
          "terms": {
            "field": "event.code",
            "size": 2147483647
      ...
  },
  "size": 0
}
```

This API returns all information about a GUID according to the field re-enumerated from the client. The GUID is a unique identifier assigned to each process and on a computer, the GUID is never duplicated to make event correlation easier.

URL	Method	Parameter	Result

/_get_aggs_field_by_GUID	- GET	- _guid
		- _field_count
		.
			"result": {
			  "GUID": {
			    "FIELD": {
			      "buckets": [(
			          "doc_count": count_number,
			          "Key": "field_count"
			        }]

Table V.16. API get data by GUID.

```
@app.route('/_get_aggs_field_by_GUID', methods=['GET'])
def GET_ALL_EVENT_ID_BY_GUID():
    try:
        guid = request.args.get("_guid")
        field_count = request.args.get('_field_count')
        field_count = int(field_count)
        response = get_event_id_by_guid(guid, LIST_FIELDS[field_count])
        return jsonify({
            'result': response["aggregations"]
        })
    except:
        return jsonify({
            'result': 'something went wrong, check your params again'
        })
]
```

The following function will populate the values(field, guid) from the passed param and return the result from Elasticsearch:

```
def get_event_id_by_guid(guid, field):
    q = query.QUERY_GET_ALL_EVENT_ID_BY_GUID
    q['aggs']['GUID']['filter']['term']['process.entity_id.keyword'] = guid
    q['aggs']['GUID']['aggs']['FIELD']['terms']['field'] = field
    res = es.search(index="winlogbeat*", body=q)
    return res
```

<u>API statistics the appearance of the fields</u>

Query ES structure:

```
{
  "query": {
    "bool": {
      "filter": {
        "range": {
          "@timestamp": {
            "gte": "",
            "lte": ""
      ...
  },
  "aggs": {
    "COMPUTER_NAME": {
      "filter": {
        "term": {
          "host.name.keyword": ""
        }
      },
      "aggs": {
        "myDateHistogram": {
          "date_histogram": {
            "field": "@timestamp",
            "time_zone": "GMT+7",
            "calendar_interval": "1m",
            "format": "yyyy-MM-dd-HH:mm"
          },
          "aggs": {
            "FIELD": {
              "terms": {
                "field": "",
                "size": 2147483647
      ...
  },
  "size": 0
```

The API statistics the fields over a period of time and returns the results of the field appearing per minute.

URL	Method	Parameter	Result

/_count_per_mi	- GET	- computer_name
		- process_name
		- time_from
		- time_to
		- field_count
		.
			`"result": {`
			`"COMPUTER_NAME": {`
			`"PROCESS_NAME": {`
			`"myDateHistogram": {`
			`"buckets": [{`
			`"FIELD": {`
			`"Buckets": [{`
			`"Doc_count": count_number,`
			`"Key": "field_count"`
			`}]`
			`...`
			`"key_as_string": "datetime"`

Table V.17. API statistics the appearance of the fields.

```
@app.route('/_count_per_mi', methods=['GET'])
def COUNT_FIELD_PER_MI():
    try:
        computer_name = request.args.get("computer_name")
        process_name = request.args.get("process_name")
        time_from = request.args.get('time_from')
        time_to = request.args.get('time_to')
        field_count = request.args.get('field_count')
        field_count = int(field_count)
        response = count_domain_per_minute(computer_name, process_name,
time_from, time_to, LIST_FIELDS[field_count])
        return jsonify({
            'result': response["aggregations"]
        })
    except expression as identifier:
        return jsonify({
            'result': 'something went wrong, check your params again'
        })
```

Before retrieving the results returned from Elasticsearch, there will be 2 choices including the specified process data or all data sent from that computer, including the time entered, the query will return the results. Statistics are taken from options.

```
def count_domain_per_minute(computer_name, process_name, time_from,
time_to, field):
    if process_name != None:
      q = query.QUERY_COUNT_PER_MI
      q['query']['bool']['filter']['range']['@timestamp']['gte'] =
time_from
      q['query']['bool']['filter']['range']['@timestamp']['lte'] = time_to
      q['aggs']['COMPUTER_NAME']['filter']['term']['host.name.keyword'] =
computer_name

      q['aggs']['COMPUTER_NAME']['aggs']['PROCESS_NAME']['filter']['term'][
      'process.name.keyword'] = process_name
      q['aggs']['COMPUTER_NAME']['aggs']['PROCESS_NAME']['aggs']['myDateHis
      togram']['aggs']['FIELD']['terms']['field'] = field

    else:
        q = query.QUERY_COUNT_PER_MI_EXCLUCE
        q['query']['bool']['filter']['range']['@timestamp']['gte'] =
time_from
```

```
        q['query']['bool']['filter']['range']['@timestamp']['lte'] =
time_to
        q['aggs']['COMPUTER_NAME']['filter']['term']['host.name.keyword'] =
computer_name
      q['aggs']['COMPUTER_NAME']['aggs']['myDateHistogram']['aggs']['FIELD'
]['terms']['field'] = field
    res = es.search(index="winlogbeat*", body=q)
    return res
```

V.6. Web Portal

V.6.1. Overview of architecture

V.6.1.1. Introduction

We build this website to target the audience of computer system administrators. For the purpose of displaying information about the user's actions on the computer such as: processes created; network connection, DNS query. From there, it will alert the user to the malicious processes that have been created and provide information to help administrators investigate and evaluate.

V.6.1.2. Web application structure

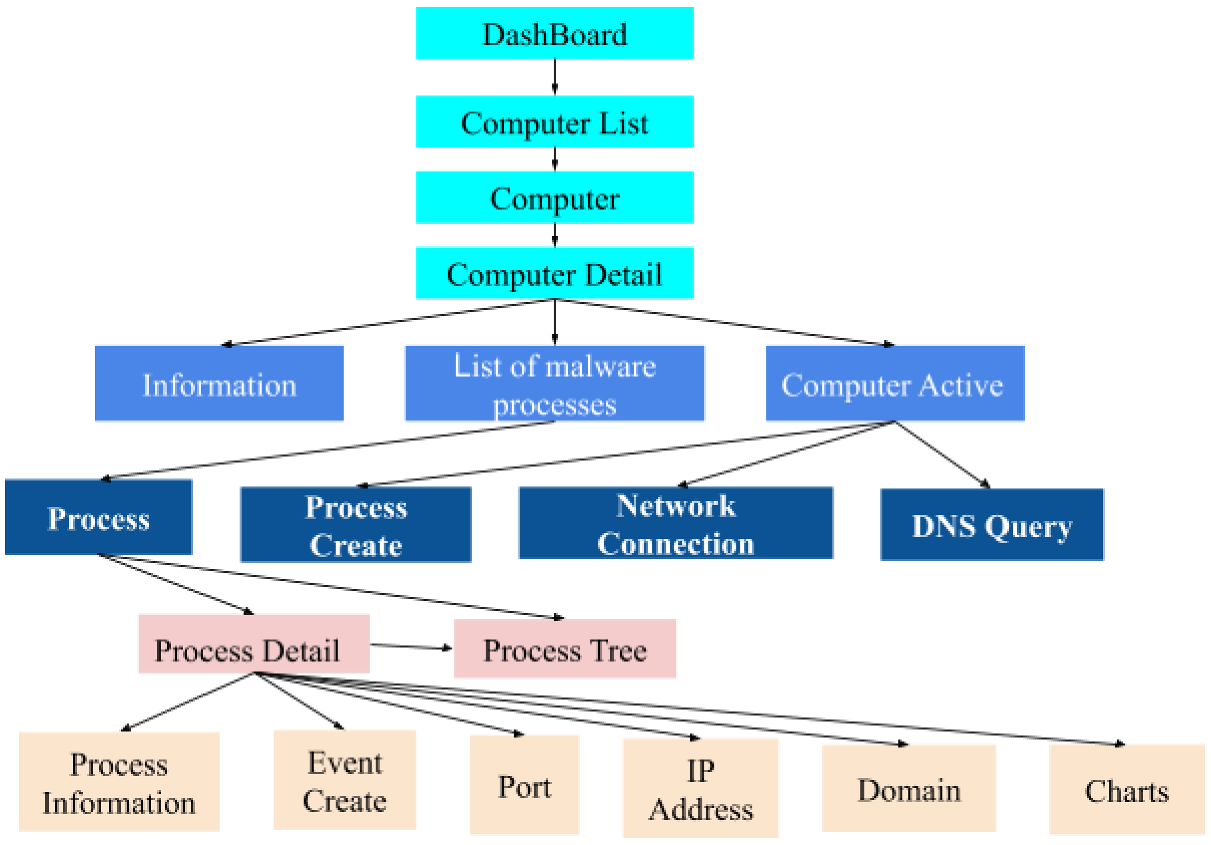


Figure V.23. Web application structure.

V.6.1.3. Feature

- Dashboard: A page that displays the list of computers in the management system and informs the user which computer is active, which computer is inactive, and will warn the user if it detects a malicious process.

- Computer Detail: A page that displays basic information, a list of malicious processes detected during the day (if any), and other computer activities during the day such as created process, network connection, DNS query of the machine. that calculation.

- Process Detail: The page displays basic information, information about malicious hash detection (if any), information about detected Mitre-Attack rule (if any), information about process connections such as events generated. , IP address, port, domain, and a graph that shows the number of queries for those connections.

- Tree Detail: The page that shows the progress tree of the process to serve the user investigation.

V.6.2. Technology

V.6.2.1. React

React is a declarative, efficient, and flexible JavaScript library for building user interfaces. It allows you to compose complex user interfaces out of small, isolated code snippets called "components"[19].

React uses JavaScript extension syntax JSX (short for JavaScript XML) to render components on the UI. When the data changes, the React will automatically update to update and render our components effectively. Below is a code snippet about the structure of a JSX file:

```
class ProcessList extends React.Component {
    render() {
      return (
        <div className="process-list">
          <h1>Processes List for {this.props.name}<h1>
          <ul>
            <li>git.exe</li>
            <li>calculate.exe</li>
            <li>chorme.exe</li>
          </ul>
        </div>
      );
    }
  }
```

V.6.2.2. Node

To use React in the project, you need to install NodeJS on your computer first. You can download NodeJS via the link: "https://nodejs.org/en/download/".

Node.js is a JavaScript runtime environment. This is a platform built on Chrome's JavaScript runtime to make it easy to build fast and scalable network applications. Node.js uses an event-driven, non-blocking I / O model, making it lightweight and efficient, perfect for real-time applications that use a lot of data running on distributed devices[20].

V.6.3. Setting and configuration

V.6.3.1. Create a new react project
The Create React App is an environment for building a new website in React.

It sets up the development environment so that it can use the latest JavaScript features. To set up on a computer will need Node> = 8.10 and npm> = 5.6 on your machine. To create a project, run the following commands on Terminal:

```
npx create-react-app my-app
cd my-app
npm start
```

V.6.3.2. Install Libraries
ReactStrap

ReactStrap is a 4-component library of React Bootstrap which is component-oriented and control-oriented[21].
In Project, I use this for UI design. To use ReactStrap, we do the following:

Install reactstrap and peer dependencies via NPM using the following command:

```
npm install --save reactstrap react react-dom
```

Install Bootstrap from NPM using the following command:

```
npm install --save bootstrap
```

Import Bootstrap CSS in the "src/index.js" file using the following command:

```
import 'bootstrap/dist/css/bootstrap.min.css'
```

FusionCharts

FusionCharts is a JavaScript library for drawing charts on the Web. In my project, I use FusionCharts to plot a "Line with Zoom and Scroll" chart.

Install the react-fusioncharts module and the fusioncharts module with the following command[22]:

```
npm install fusioncharts react-fusioncharts --save
```

React-d3-tree

React-d3-tree is a React library for hierarchical data representation. In the project, I used React-d3-tree to draw the progress tree. Install the React-d3-tree modules using the following command[23]:

```
npm i --save react-d3-tree
```

V.6.4. Implementation

V.6.4.1. Check Mitre-Attack Detect

To check if the process was detected by "Miter-Attack", check the data returned from the api:

http://SERVER_IP:PORT/process_detail

Data returned from the server in the form of "json". From there, we'll check that the value of object-key is “mitre-detected”. If the value is 0 then the process is not detected by the law of mitre-attack. But if the value is 1, then we will check which type of mitre rule it sticks to. The rule detects that a process enters the correct type of rule that the mitre has classified. Below is the code that checks and shows whether the process was found to be malicious by the "mitre-attack" rule:

```
process._source.mitre_detect["mitre-detected"] === "1" ? (
    mitre.map((detail,id) =>
       detail.toLowerCase() ===
      this.state.process._source.mitre_detect.info.tags[0].split('.')[1]
      .toLowerCase()
        ? (
           <th key={id}id="description">
{this.state.process._source.mitre_detect.info.title}
            <UncontrolledTooltip placement="left"  target="description">
            {this.state.process._source.mitre_detect.info.description}
             </UncontrolledTooltip>
           </th>
       ) : <th key={id}></th>
   )
) : (
   null
)
```

V.6.4.2. Check Hash Detect

To check if the process is detected by the "Hash Service", you must check the data returned from the api:

http://SERVER_IP:PORT/process_detail

Data returned from the server in the form of "json". From there we will check that the returned data has an object-key of "hash_detect". If none exists then that process's hash is checked for clean. But if there exists, this process is detected by Hash Service as "malicious", based on that will display the name, type and status of that hash for the user.Below is the code that checks and shows whether the process was found to be malicious by a "hash-service":

```
this.state.process._source.hasOwnProperty("hash") ? (

          this.state.process._source.hasOwnProperty("hash_detect")?(
              this.state.process._source.hash.hasOwnProperty("sha1")?
          (
              <tr>
                  <th>{this.state.process._source.hash.sha1}</th>
                  <th>SHA1</th>


                  <th>{this.state.process._source.hash_detect.result.status
                  }</th>

              </tr>
          ): (
              <tr>
                  <th>{this.state.process._source.hash.md5}</th>
                  <th>MD5</th>

      <th>{this.state.process._source.hash_detect.result.status}</th>

                    </tr>
                )
                ) : null

      ) : null
```

V.6.4.3. Tree Process

To draw a process tree of a process, you must get the data returned from the api:

```
http://SERVER_IP:PORT/api/tree
```

Data returned from the server in the form of "json". From there we will reformat the returned data accordingly to draw the process tree. We write a function as "formatJsonData (data)". This is a recursive function that can check the branches by using the function "hasOwnProperty" to find the object-key "children" of the data returned from the server. And the returned data will be a new data json containing necessary information such as: basic process information, node shape in the tree. As for the shape of the nodes in the tree, they will change the color of the book if the data is returned during reformatting: if detected by the hash-service it will be red; If detected by a mitre rule there are three levels: high corresponds to red, medium corresponds to orange and low corresponds to yellow; otherwise undetected by the hash-service and the mitre will be green.

Once we have the right format data, we will use the "react-d3-tree" library to draw the progress tree on the web along with some other data to be able to draw the complete tree. Below is the code to draw a tree on the UI:

```
<Tree
    defaultOptions={{scrollwheel: false,}}
    data={this.state.data_tree}
    orientation={this.state.orientation}
    translate={{ x: this.state.translateX, y: this.state.translateY }}
    pathFunc={this.state.pathFunc}
    collapsible={this.state.collapsible}
    initialDepth={this.state.initialDepth}
    scaleExtent={this.state.scaleExtent}
    nodeSize={this.state.nodeSize}
    separation={this.state.separation}
    onClick={this.get_node}
    useCollapseData={this.state.useCollapseData}
    transitionDuration={this.state.transitionDuration}
    textLayout={this.state.textLayout}
    allowForeignObjects
    nodeLabelComponent={{
        render: <NodeLabelComponent/>,
        foreignObjectWrapper: {
        width: 150,
        height:150,
        x: -45,
        y: -15
        }
    }}
    styles={this.state.styles}
/>
```

V.6.4.4. Charts Process

To graph a process, you must get the data returned from the api:

http://SERVER_IP:PORT/_count_per_mi

To get data from the server we have to pass parameters such as: computer_name, process_name, time_from, time_to and field_count. We will call api 4 times corresponding to the number of field_count is 0, 1, 2, 3 with the return data of domain, port, ip and event. To be able to plot the graph, we have to reformat the data returned from the server. The data returned is the number of hits per field per minute. So to be able to represent on the graph, we have to sum all the hits of each field for 1 minute and the data for each minute will be separated by a "|". Below is the code to reformat the data from the server:

```
domain.buckets.map(detail =>{

          if(detail.FIELD.buckets.length > 0) {
              let sum = 0;
              detail.FIELD.buckets.map(count =>
                  sum = sum + Number(count.doc_count)
              )
              return line_domain = line_domain + '' + sum + "|"
          }
          return line_domain += "0|";

});
line_domain = line_domain.substr(0, line_domain.length - 1);
```

When we reformat the data of each field, we will aggregate it with some other information into a variable called "dataSource".

After synthesis is completed, use the library of "FusionCharts" to draw the graph on the Web. Below is the code that displays the graph on the UI:

```
<ReactFusioncharts

          className="chart"
          type="zoomline"
          width="100%"
          height="100%"
          dataFormat="JSON"
          dataSource={dataSource}

/>
```

V.6.4.5. Routers

In order to link the pages together, we need to create a file called "router.js". The file contains the list of routers to be used in the project. Each router includes: path, name, element, layout. Here is the code that contains the list of routers:

```
var routes = [

          {
            path: "/dashboard",
            name: "Dash Board",
            component: DashBoard,
            layout: "/home"
          },
          {
            path: "/tree-process",
            name: "Tree Detail",
            component: TreeDetail,
            layout: "/home"
          },...
      ]
```

When we have a list of necessary routers, we will use a common layout to rotate components for each other each time the link arrives. Below is the code that displays the UI of each router on the main layout:

```
get_routes = routes => {

          return routes.map((prop, key) => {
            if (prop.layout === "/home") {
              return (
                <Route
                  path={prop.layout + prop.path}
                  component={prop.component}
                  key={key}
                />
              );
            } else {
              return null;
```

```
                }
            });

};
```

V.6.4.6. Connect with server API

To connect to the server's api on react we use the Fetch API under "Web API Interfaces" which is a simple api for sending and receiving requests using js. In this project, I use 2 methods of passing param to the GET and POST methods:

For the GET protocol, we pass the param directly to the api to send it to the server:

```
get_connection(guid, field) {
      fetch("http://SERVER_IP:PORT/_get_aggs_field_by_GUID?_guid="+guid+"&_
field_count="+field, {

              method: 'GET',
              redirect: 'follow'
          })

      .then(response => response.json())
      .then(res => {})
}
```

For the POST protocol, pass the parameters to send to the "body" to send to the server:

```
get_connection(computer, mode) {
          const   t_from   =   moment().subtract(1,   'days').format('MM/DD/YY
HH:mm:ss');
    const t_to= moment().format('MM/DD/YY HH:mm:ss');
    fetch("http://SERVER_IP:PORT/all", {
      method: 'POST',

      body:
      JSON.stringify({"computer_name":computer,"mode":mode,"from":t_from,"t
      o":t_to}),

        redirect: 'follow'
    })
    .then(response => response.json())
    .then(res => {})
}
```

V.6.5. Demonstration

The home page will display a list of active computers in the system. Users can know whether the machines are running or stopped and notify the user if there is any malicious progress. Figure V-1 shows the list of machines in the system.

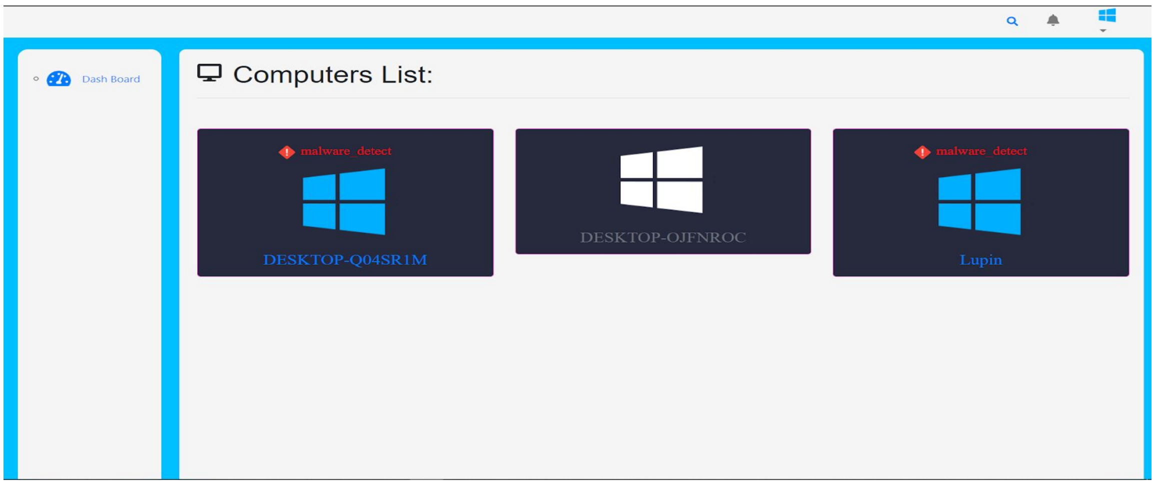
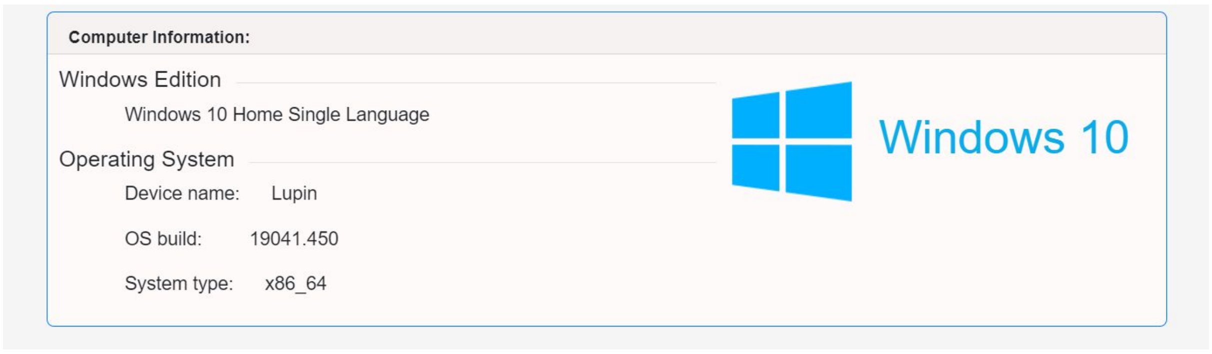


Figure V.24. List of computers active in the system.

When you click on the hostname, the user will be redirected to the machine's details page. The interface will display information such as: basic information about that computer, detected malicious processes (if any) and other operational information such as: created process, network connection, access DNS query. Figure V.2 shows basic computer information.



Computer Information:

Windows Edition

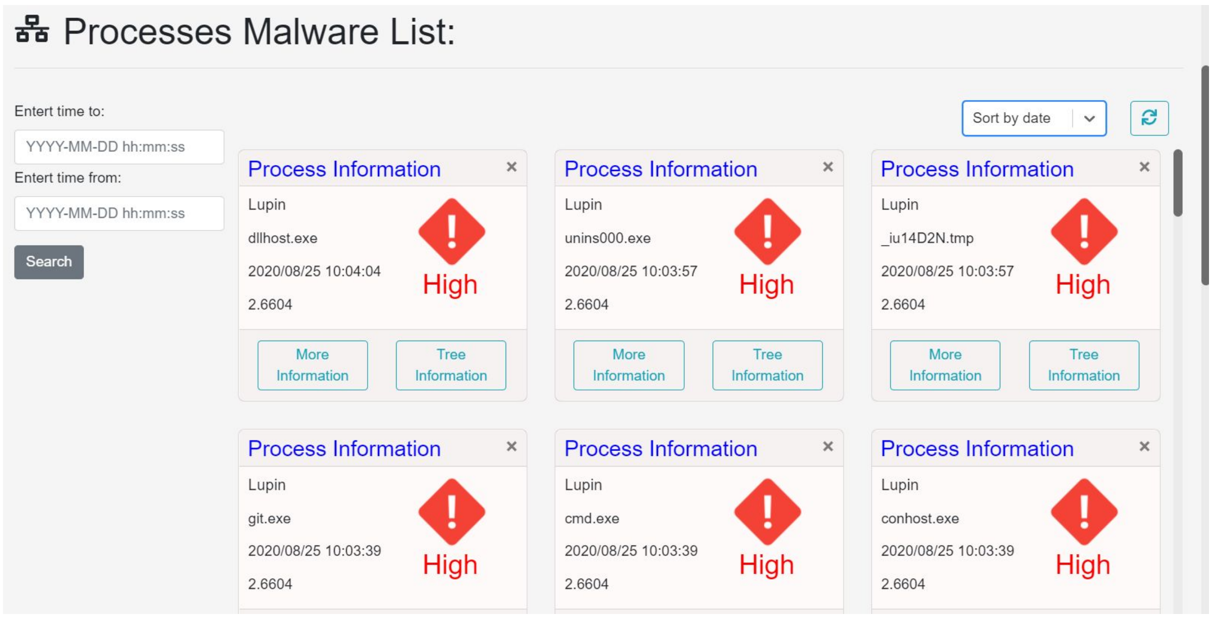
Windows 10 Home Single Language

Operating System

Device name:	Lupin
OS build:	19041.450
System type:	x86_64

Figure V.25. Basic information of the computer.

A portion of the list of malicious processes will show the malicious processes detected during the day. In this section, there will be 3 main functions of categorizing processes by name and date, refreshing the processes and looking for other malicious processes when it is time to check for malicious processes.

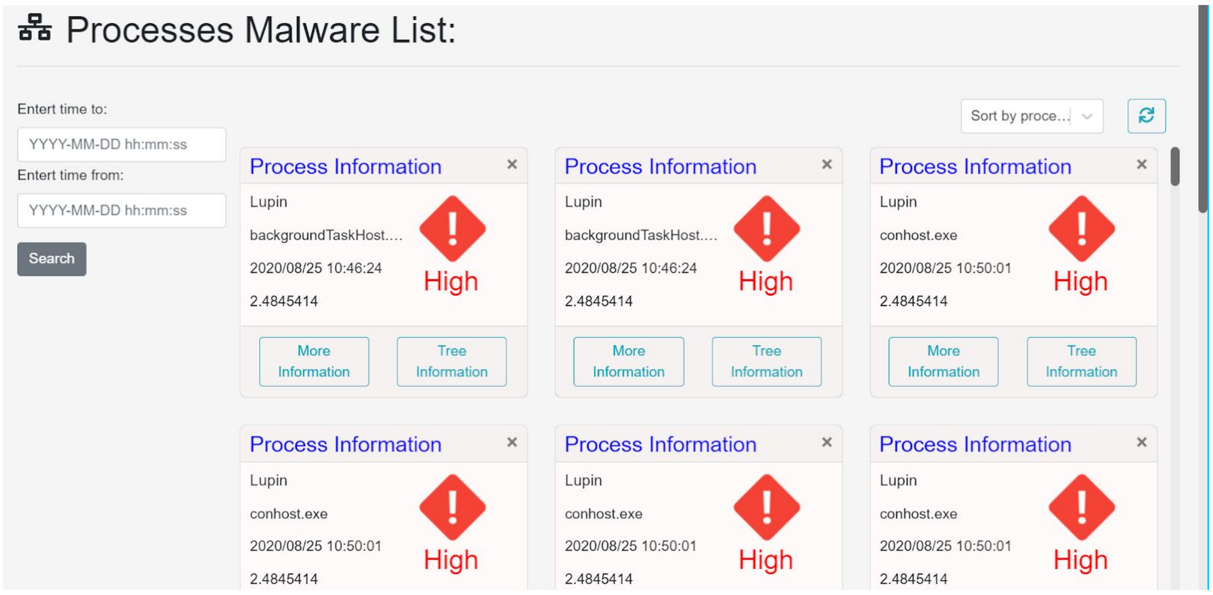

Processes Malware List:

Entert time to: YYYY-MM-DD hh:mm:ss
Entert time from: YYYY-MM-DD hh:mm:ss
Search

Sort by date

Process Information	Lupin	dllhost.exe	2020/08/25 10:04:04	2.6604	High
Process Information	Lupin	unins000.exe	2020/08/25 10:03:57	2.6604	High
Process Information	Lupin	_iu14D2N.tmp	2020/08/25 10:03:57	2.6604	High
Process Information	Lupin	git.exe	2020/08/25 10:03:39	2.6604	High
Process Information	Lupin	cmd.exe	2020/08/25 10:03:39	2.6604	High
Process Information	Lupin	conhost.exe	2020/08/25 10:03:39	2.6604	High

Figure V.26. List of malicious processes sorted by date.

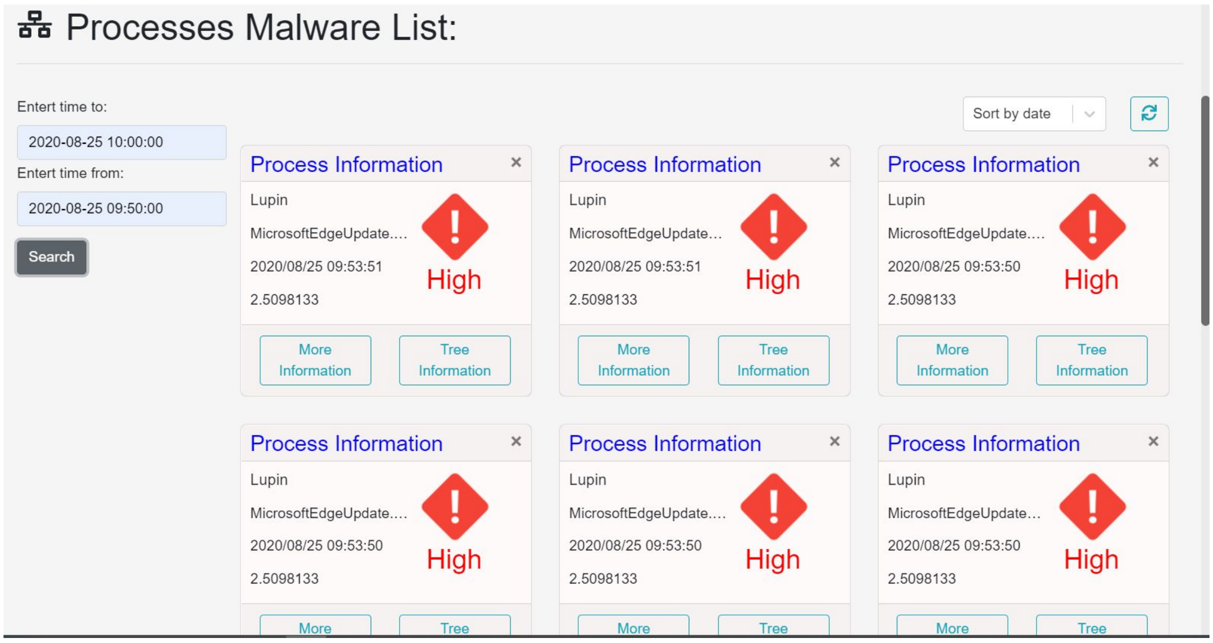

Processes Malware List:

Entert time to: YYYY-MM-DD hh:mm:ss
Entert time from: YYYY-MM-DD hh:mm:ss
Search

Sort by proce...

Process Information	Lupin	backgroundTaskHost....	2020/08/25 10:46:24	2.4845414	High
Process Information	Lupin	backgroundTaskHost....	2020/08/25 10:46:24	2.4845414	High
Process Information	Lupin	conhost.exe	2020/08/25 10:50:01	2.4845414	High
Process Information	Lupin	conhost.exe	2020/08/25 10:50:01	2.4845414	High
Process Information	Lupin	conhost.exe	2020/08/25 10:50:01	2.4845414	High
Process Information	Lupin	conhost.exe	2020/08/25 10:50:01	2.4845414	High

Figure V.27. List of malicious processes sorted by processes name.


Processes Malware List:

Entert time to: 2020-08-25 10:00:00

Entert time from: 2020-08-25 09:50:00

Search

Sort by date

Process Information
Lupin
MicrosoftEdgeUpdate....
2020/08/25 09:53:51
2.5098133
High

Process Information
Lupin
MicrosoftEdgeUpdate...
2020/08/25 09:53:51
2.5098133
High

Process Information
Lupin
MicrosoftEdgeUpdate....
2020/08/25 09:53:50
2.5098133
High

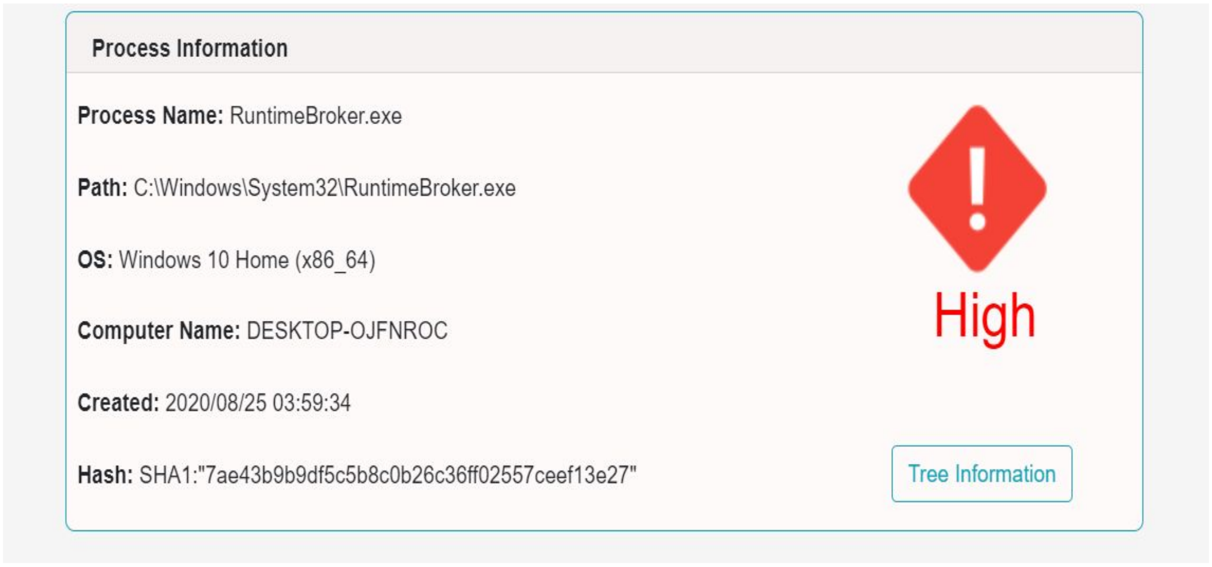
Process Information
Lupin
MicrosoftEdgeUpdate....
2020/08/25 09:53:50
2.5098133
High

Process Information
Lupin
MicrosoftEdgeUpdate....
2020/08/25 09:53:50
2.5098133
High

Process Information
Lupin
MicrosoftEdgeUpdate...
2020/08/25 09:53:50
2.5098133
High

Figure V.28. List of malicious processes when using the search function.

In each section that shows the unique process, in addition to some information of that process, there are 2 buttons "More info" and "Tree information". Clicking the "More Information" button will go to the details page of that process. Will provide users with detailed information about that process including process information; detected by hash or miter-attack; number of operations and connections made such as: event generated, ip address, domain, port, and data representation graph.


Process Information

Process Name: RuntimeBroker.exe

Path: C:\Windows\System32\RuntimeBroker.exe

OS: Windows 10 Home (x86_64)

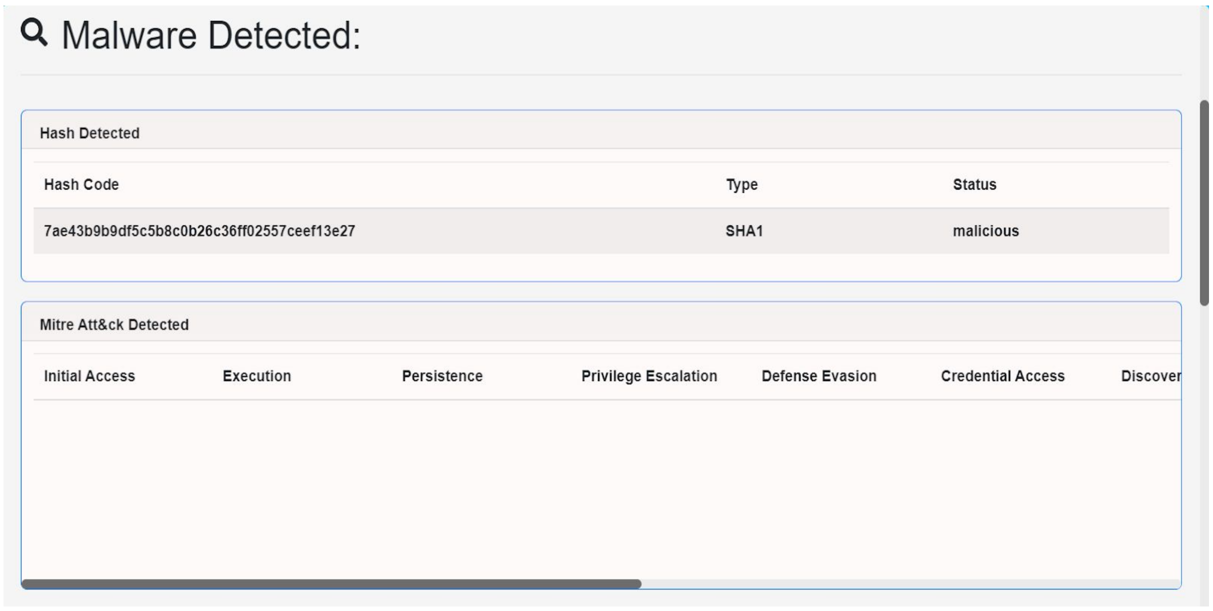
Computer Name: DESKTOP-OJFNROC

Created: 2020/08/25 03:59:34

Hash: SHA1:"7ae43b9b9df5c5b8c0b26c36ff02557ceef13e27"

High

Figure V.29. Basic information about the process.



Malware Detected:

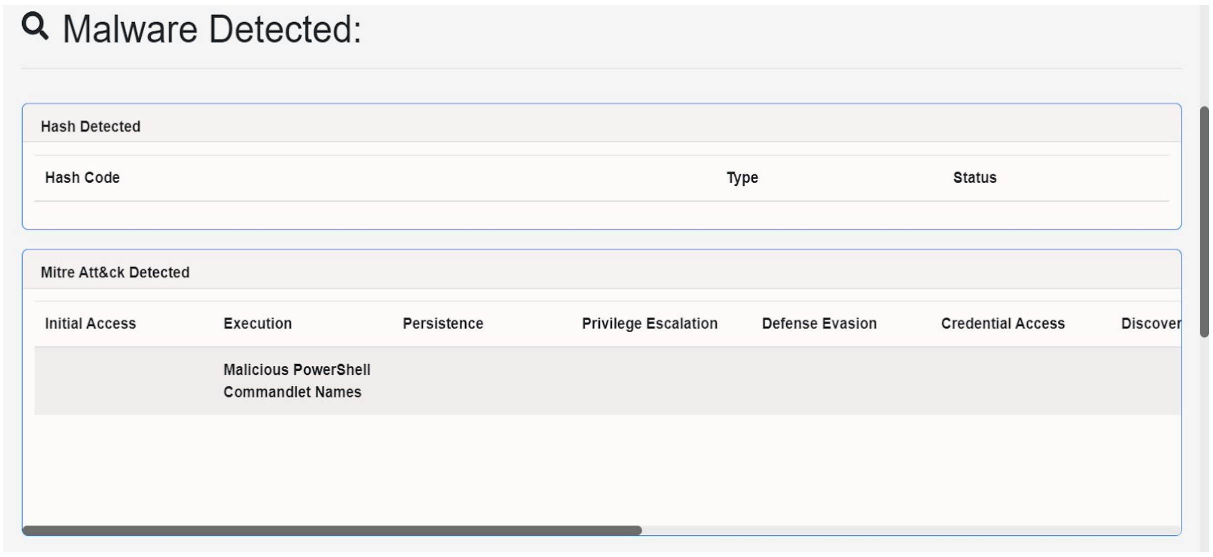
Hash Detected

Hash Code	Type	Status
7ae43b9b9df5c5b8c0b26c36ff02557ceef13e27	SHA1	malicious

Mitre Att&ck Detected

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discover

Figure V.30. Information about detecting malicious process due to hash service.



Malware Detected:

Hash Detected

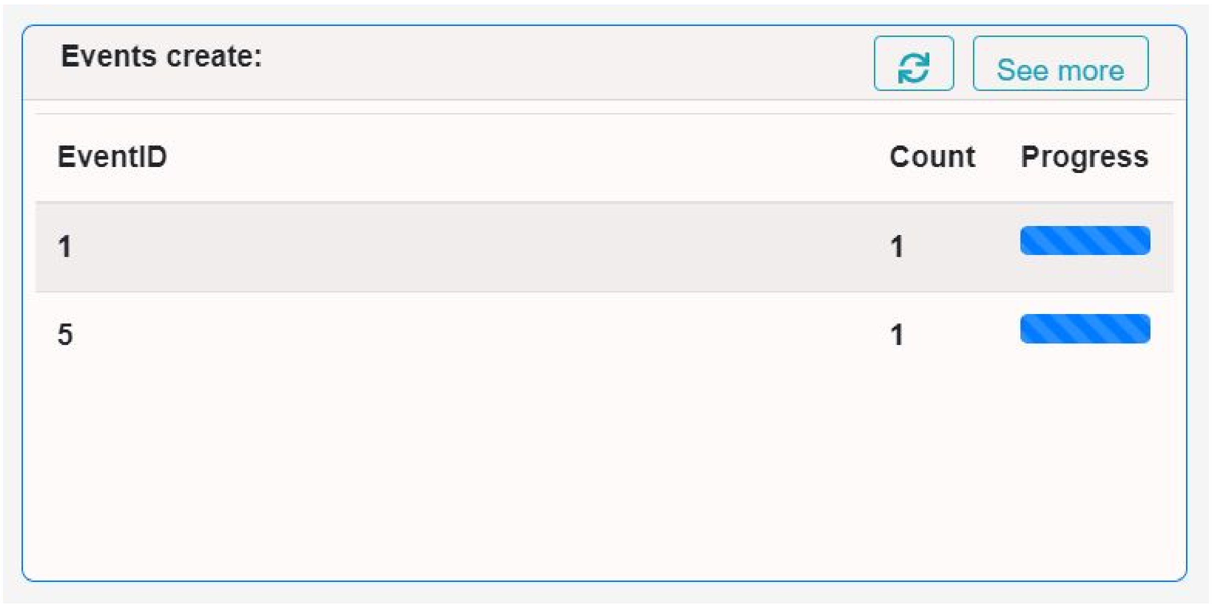
Hash Code	Type	Status

Mitre Att&ck Detected

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discover
	Malicious PowerShell Commandlet Names					

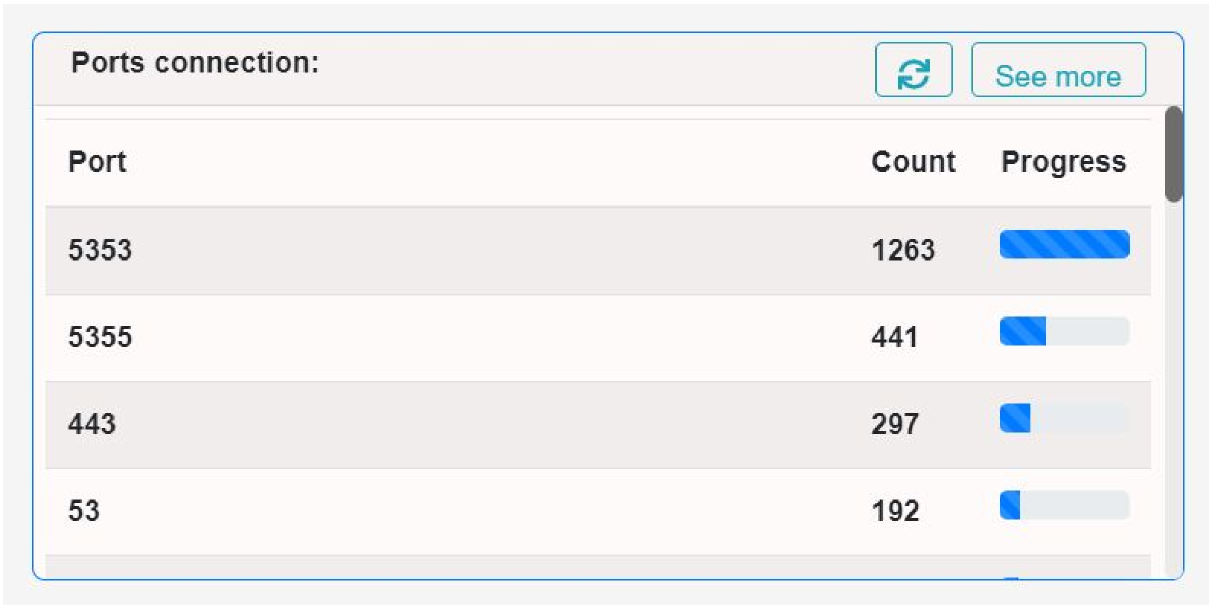
Figure V.31. Mitre-attack malicious process detection information.

In the connection part of the procedure, there are 2 buttons: "refresh" and "see more". With a "refresh" button clicked will refresh the list displayed on the board. And the "see more" button, when clicked, will switch to a page showing more values that are not visible on the table.



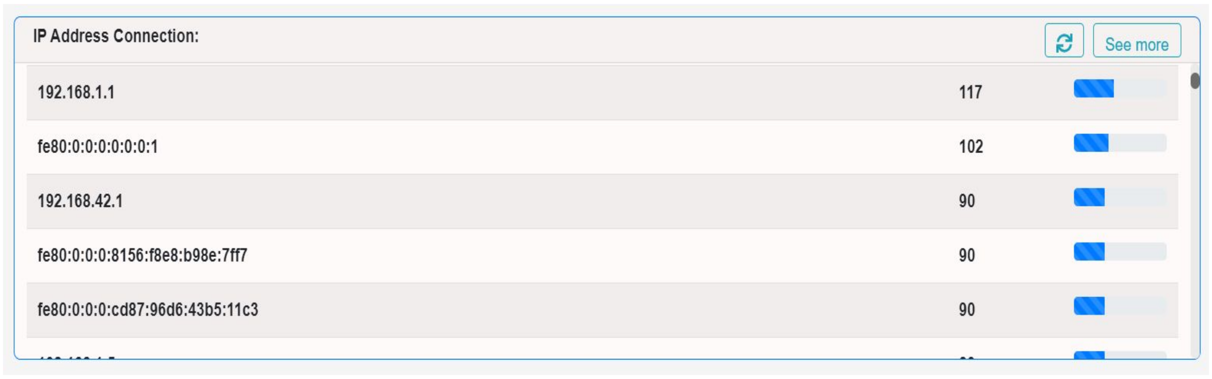
EventID	Count	Progress
1	1	▨
5	1	▨

Figure V.32. Information about the event name and initialization times.



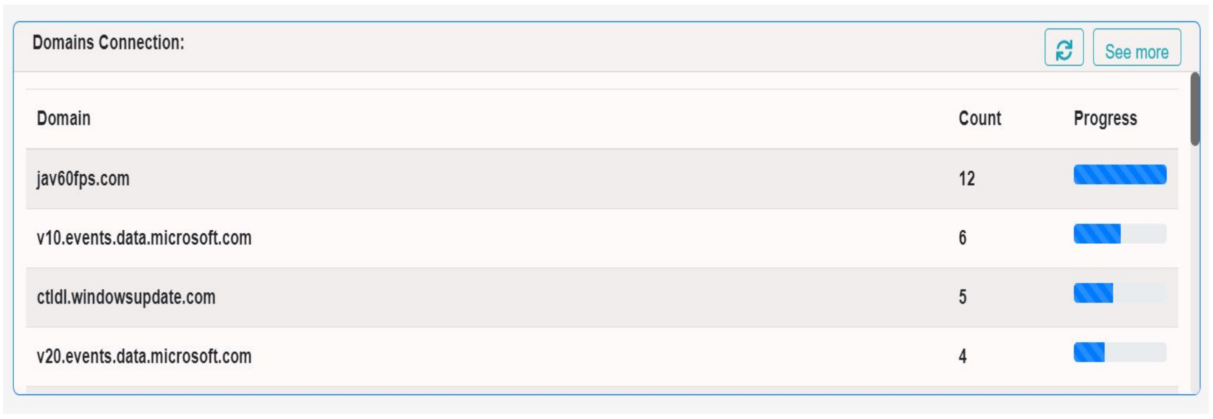
Port	Count	Progress
5353	1263	▨
5355	441	▨
443	297	▨
53	192	▨

Figure V.33. Port information and number of connections.



192.168.1.1	117	▨
fe80:0:0:0:0:0:0:1	102	▨
192.168.42.1	90	▨
fe80:0:0:0:8156:f8e8:b98e:7ff7	90	▨
fe80:0:0:0:cd87:96d6:43b5:11c3	90	▨

Figure V.34. Information about the name and number of connections of the IP Address.


Domains Connection:

Domain	Count	Progress
jav60fps.com	12	
v10.events.data.microsoft.com	6	
ctldl.windowsupdate.com	5	
v20.events.data.microsoft.com	4	

Figure V.35. Information about the names and number of joins of domains.

In a chart, when people click and drag in the cell area to zoom in and scroll, the chart displays more data of smaller timelines. To go back to the zoom level click on the "zoom out by level" button or if you want to refresh the chart click the "Reset chart" button.

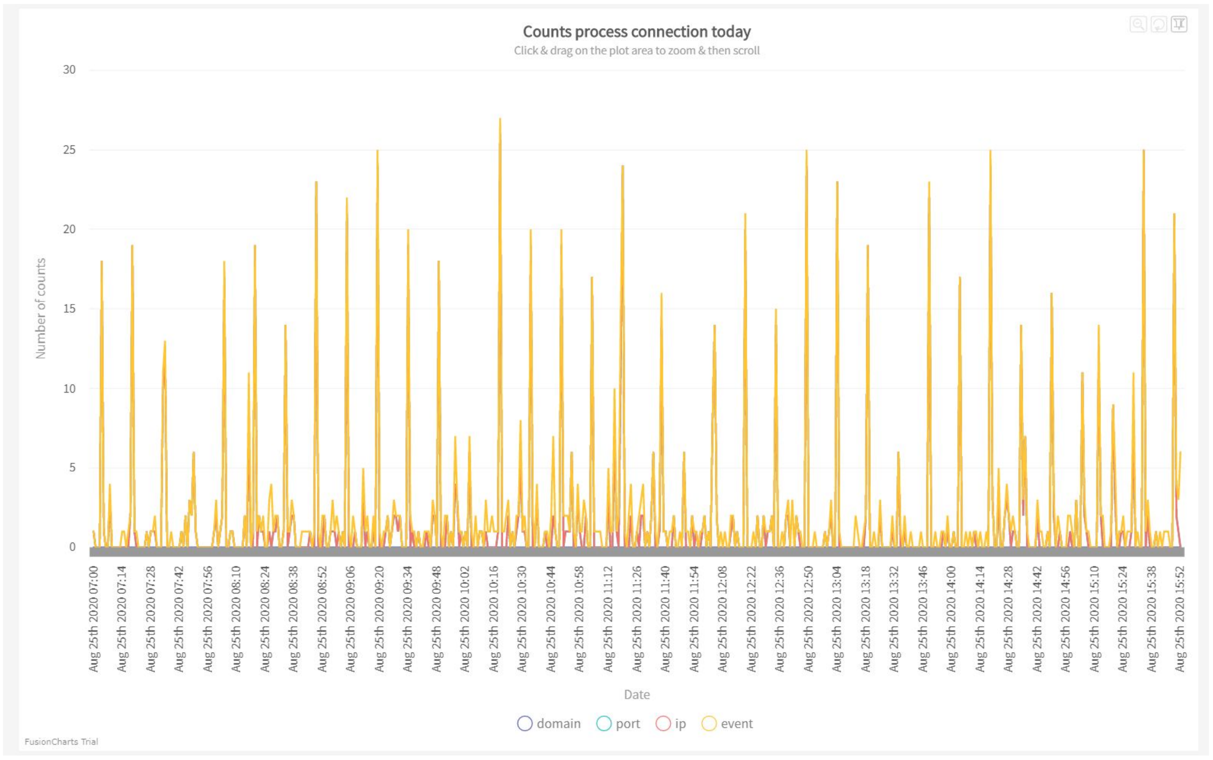
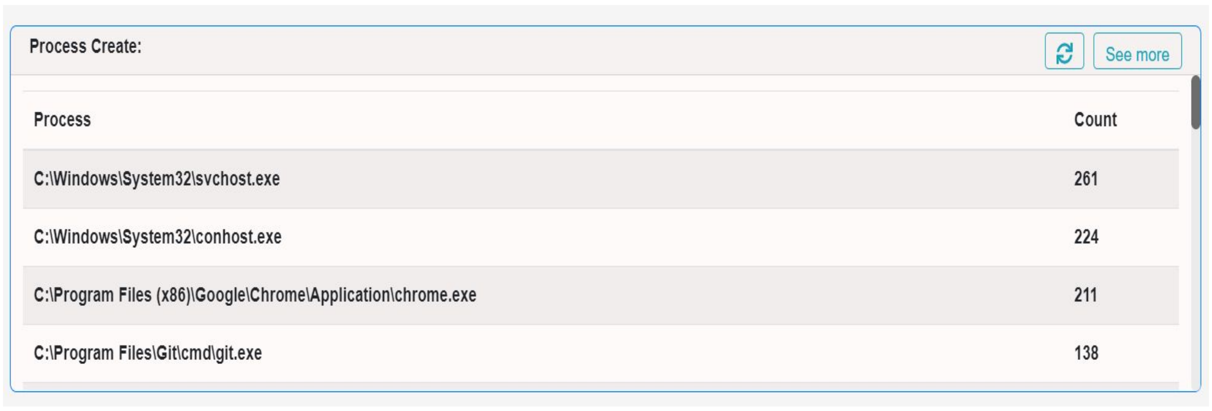


Figure V.36. Graph showing the number of hits of Event ID, IP, domain, port.

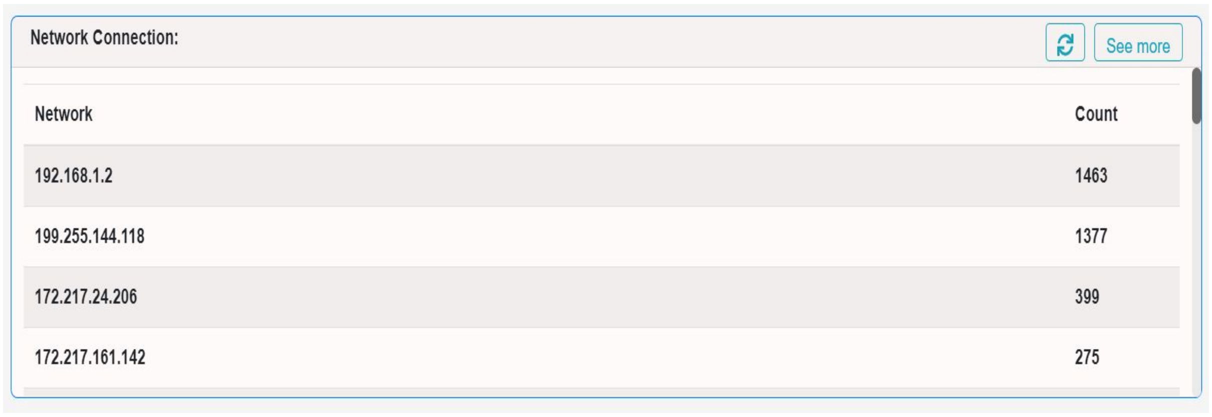
Next is the working part of the computer. Will check how many times the processes has been created, how many times the network is connected, and query dns for today. Also users can view more information by clicking the "See More" button. There will display data for users to customize by day, week, month.



Process Create:

Process	Count
C:\Windows\System32\svchost.exe	261
C:\Windows\System32\conhost.exe	224
C:\Program Files (x86)\Google\Chrome\Application\chrome.exe	211
C:\Program Files\Git\cmd\git.exe	138

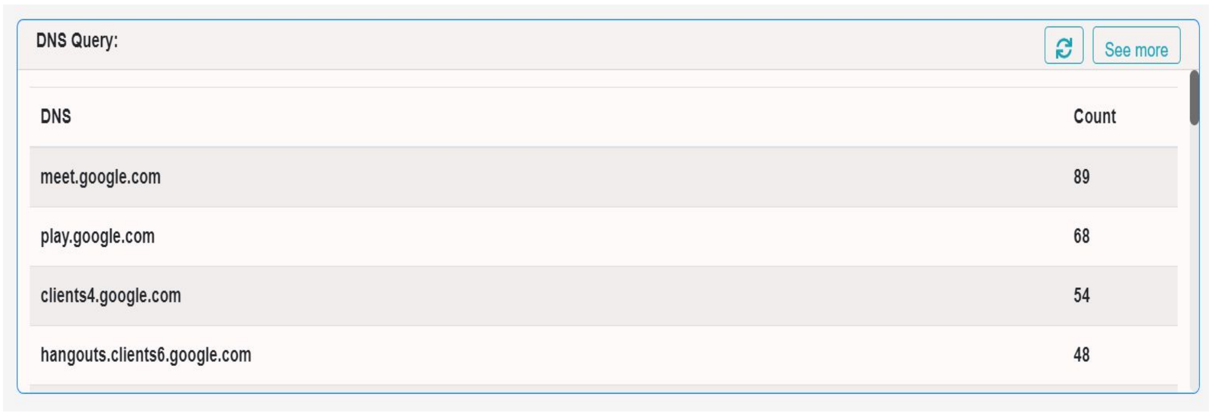
Figure V.37. List of created processes.



Network Connection:

Network	Count
192.168.1.2	1463
199.255.144.118	1377
172.217.24.206	399
172.217.161.142	275

Figure V.38. List of network connections.

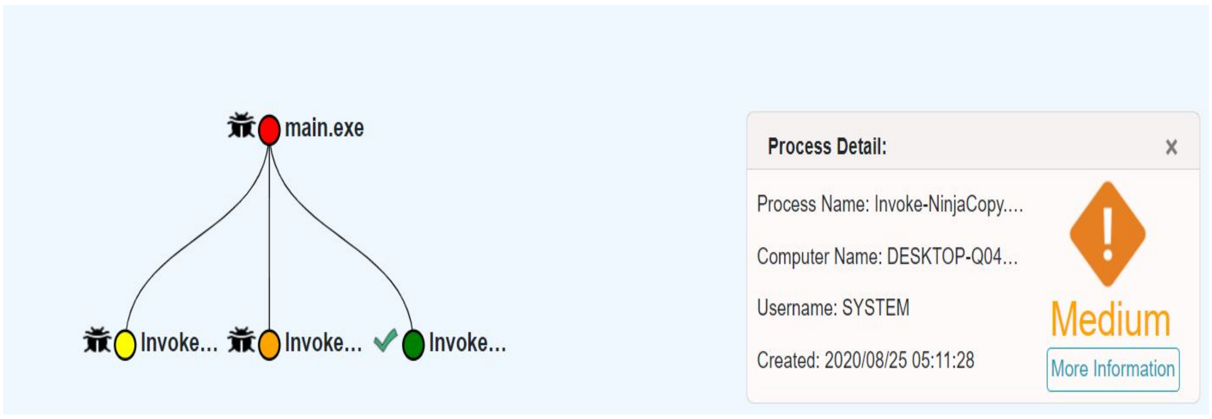


DNS Query:

DNS	Count
meet.google.com	89
play.google.com	68
clients4.google.com	54
hangouts.clients6.google.com	48

Figure V.39. List of DNS queries.

To display tree information for each process, the user clicks the "Tree Information" button on the "Computer Details" or "Process Details" page. The tree information in each process displays colors according to the severity of the process, and when the tree nodes are clicked, the progress panel appears and a button, when clicked, switches to the page "Process Detail".



Process Name: Invoke-NinjaCopy....

Computer Name: DESKTOP-Q04...

Username: SYSTEM

Created: 2020/08/25 05:11:28

Figure V.40. Tree information of process.

VI. PROJECT VALIDATION

VI.1. Project Idea

Today, malware, also collectively known as computer virus, has long been a threat that has been stalking computer users for a long time. For businesses and organizations, this danger is even greater when important and sensitive information of the whole organization can be lost or leaked at any time by malicious code on a certain computer in the network.

Malicious code has continuously changed, become more and more sophisticated and difficult to detect. This poses a big challenge for traditional anti-malware solutions, which only focus on recognizing known patterns (signatures) or relying on a small number of specific behaviors. In fact, nowadays, there are still many types of malicious code that silently perform malicious acts without being detected by any traditional anti-virus software. Because of the reason mentioned above, we want to develop a multi-purpose, in-depth system with the desire to improve the security of the system for businesses to prevent the rapid spread of an attack. This system can help experts to early identify attacks and detect malicious running programs on the system.

VI.2. Result

During the last semester, we built a complete system helping information systems automatically detect attack.

The web application allows administrators to monitor, detect threats early, attack, stop the spread and display information that makes it easier to observe and analyze threats intimidation.

The detection system can be deployed on different servers or single server powerful enough configuration flexibly. This is a place to control event logs from many places, evaluate and label each event log, the system includes many independent modules, allowing multiple tasks to be performed, detection and notification in real-time.

All detection results will be stored in the Elastic Search database for further research or analysis.

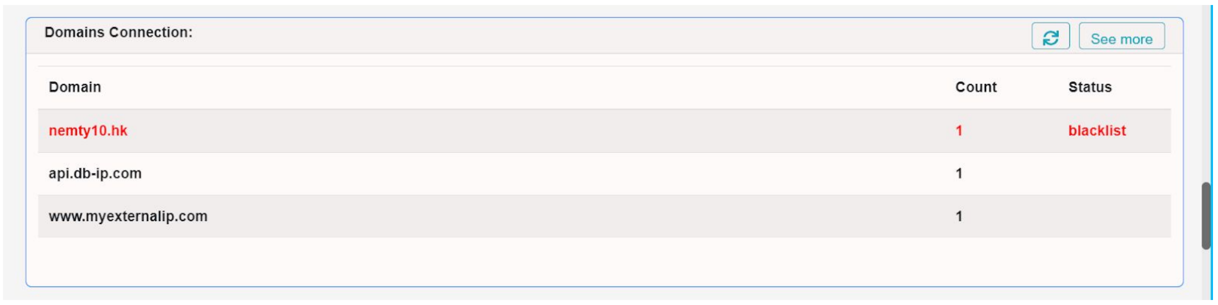
VI.3. Future

In the development phase of the 4th industrial revolution, technology trends are increasingly high and require processes to ensure the security and stability of the system to be upgraded and improved. Enterprises in general and businesses in Vietnam, in particular, are increasingly concerned with data security and dealing with increasingly common cyber attacks, in the future we aim to provide service for small and medium-sized companies that can use this product widely and far is applicable to large domestic and foreign companies. Collecting event log data to detect the attack will help us develop automated monitoring systems using machine learning in the future. Future systems will also support different operating systems such as Linux, Ubuntu, Mac, etc.

VII. EVALUATION AND EXPERIMENTATION

1. Malicious IP domain detection

In this experiment, when we run malware, it is connected to many domains as shown below:



Domains Connection:

Domain	Count	Status
nemty10.hk	1	blacklist
api.db-ip.com	1	
www.myexternalip.com	1	

Figure V.41. .

There is a domain that is notified to the web administrator.

2. Blacklist SHA1 hash detection

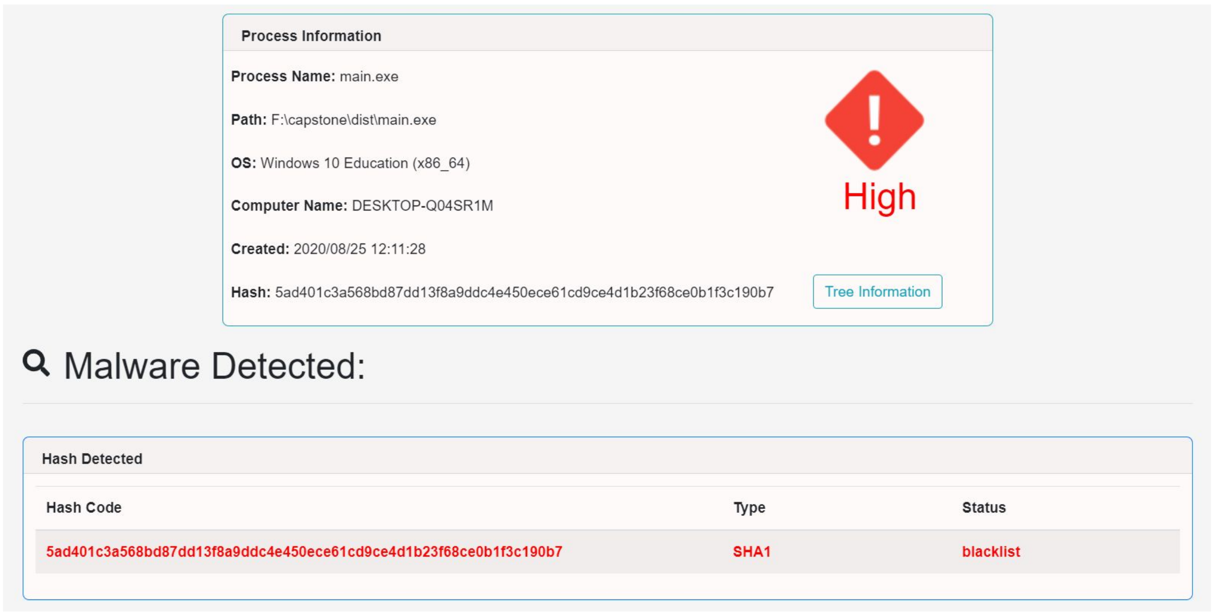
For this experiment, we use a procedure with the hash code in the following list detected by the hash service. Run the file 'main.exe' using the virtual machine:

Name

main.exe

Figure V.42. Program 'main.exe'

The results are immediately notified to the web administrator, detailed information about the hash detect is also displayed:


Process Information

Process Name: main.exe

Path: F:\capstone\dist\main.exe

OS: Windows 10 Education (x86_64)

Computer Name: DESKTOP-Q04SR1M

Created: 2020/08/25 12:11:28

Hash: 5ad401c3a568bd87dd13f8a9ddc4e450ece61cd9ce4d1b23f68ce0b1f3c190b7

Malware Detected:

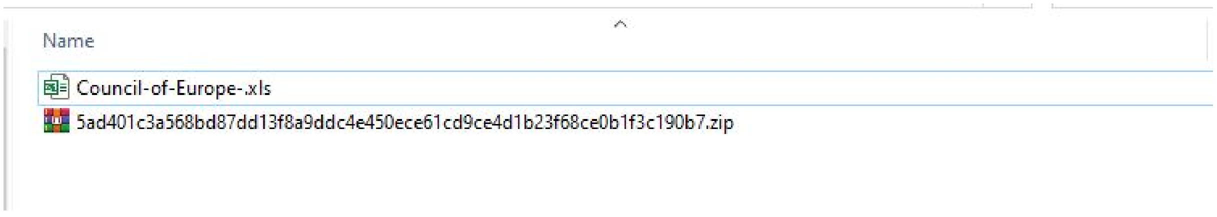
Hash Detected

Hash Code	Type	Status
5ad401c3a568bd87dd13f8a9ddc4e450ece61cd9ce4d1b23f68ce0b1f3c190b7	SHA1	blacklist

Figure VI.42. Information about hash detect is reported and displayed on the web

3. Malicious behavior detection based on Mitre

For this experiment, we use the malicious code 'Council-of-Europe-.xls'. Run the file 'Council-of-Europe-.xls' using a virtual machine:

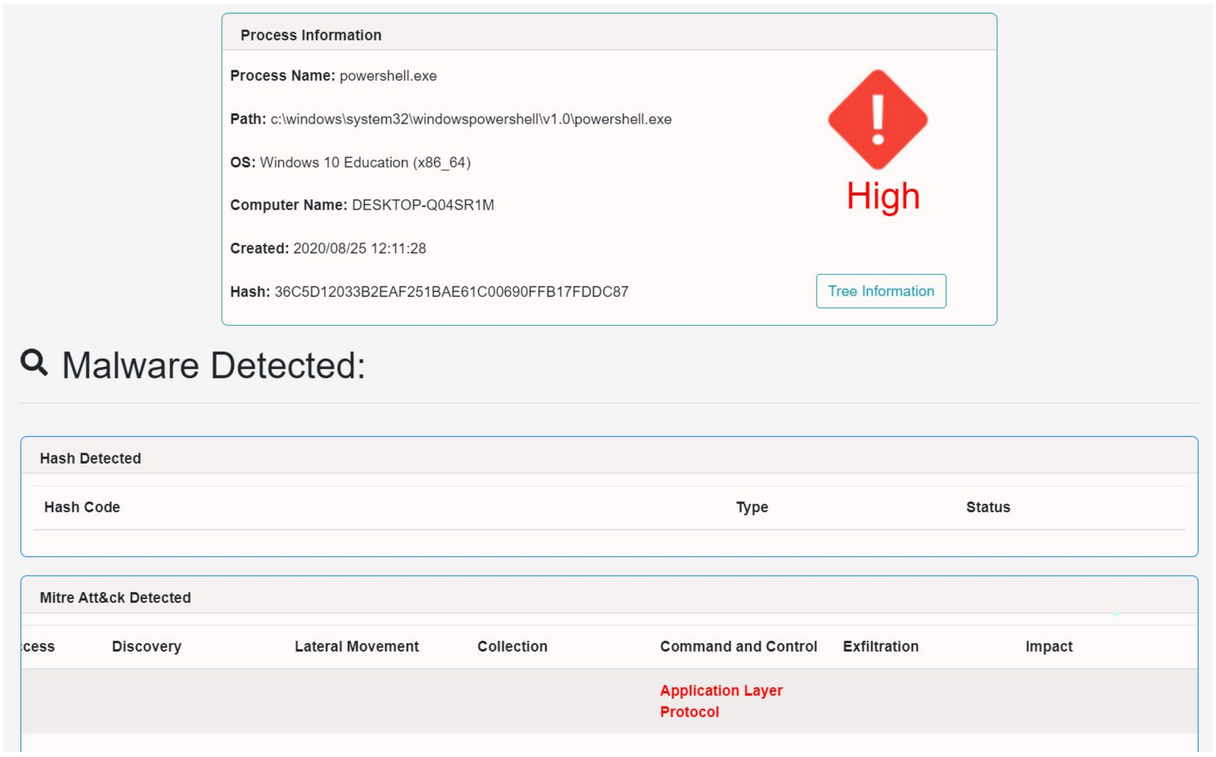

Name

Council-of-Europe-.xls

5ad401c3a568bd87dd13f8a9ddc4e450ece61cd9ce4d1b23f68ce0b1f3c190b7.zip

Figure V.43. Malware 'Council-of-Europe-.xls'

The results are immediately notified to the web administrator, detailed information about the malicious code is also displayed:


Process Information

Process Name: powershell.exe

Path: c:\windows\system32\windowspowershell\v1.0\powershell.exe

OS: Windows 10 Education (x86_64)

Computer Name: DESKTOP-Q04SR1M

Created: 2020/08/25 12:11:28

Hash: 36C5D12033B2EAF251BAE61C00690FFB17FDDC87

High

Tree Information

Malware Detected:

Hash Detected

Hash Code	Type	Status

Mitre Att&ck Detected

cess	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
				Application Layer Protocol		

Figure V.44. Information about malicious code is reported and displayed on the web

Appendix A - Definition and acronyms

Acronym	Definition	Note
ATA	Anti - Targeted Attack	
APT	Advanced Persistent Threat	
C2	Command and Control	
MIC	Ministry of Information and Communication	
EDR	Endpoint Detection and Response	
SIEM	Security Information and Event Management	
IDE	Integrated Development Environment	
VPS	Virtual Private Server	
CVE	The Common Vulnerabilities and Exposures	
CVSS	Common Vulnerability Scoring System	
NSE	Nmap Scripting Engine	
SQL	Structured Query Language	
CPE	Common Platform Enumeration	
API	Application Program Interface	
RCE	Remote Code Execution	
PDF	Portable Document Format	
HTTP	Hypertext Transfer Protocol	
HTTPS	Hypertext Transfer Protocol Secure	
GUI	Graphic User Interfaces	
SSD	Solid State Drive	
RAM	Random Access Memory	
VPS	Virtual Private Server	
CPU	Central Processing Unit	
OS	Operating System	
VPN	Virtual Private Network	
IDS	Intrusion Detection System	
IPS	Intrusion Prevention System	
URL	Uniform Resource Locator	
NMAP	Network Mapper	
IP	Internet Protocol	
JSON	JavaScript Object Notation	
XML	Extensible Markup Language	
WAN	Wide Area Network	

LAN	Local Area Network	
VA	Vulnerabilities Assessment	
XSS	Cross Site Scripting	
WAF	Web Application Firewall	
DOS	Denial of service	
ML	Machine learning	
ES	Elasticsearch	
RF	Random Forest	

Appendix B - List of Figures

Appendix C - List of Table

Appendix D - References

[1]. "Kaspersky Endpoint Detection and Response (EDR) | Kaspersky." https://www.kaspersky.com/enterprise-security/endpoint-detection-response-edr. Accessed 28 Aug. 2020.

[2] "APEX ONE™." https://www.lec-expo.com/public/exposants_files/ApexOneSolutionBrief.pdf. Accessed 28 Aug. 2020.

[3] "TRAPS TECHNOLOGY OVERVIEW." https://cdw-prod.adobecqms.net/content/dam/cdw/on-domain-cdw/brands/palo-alto-networks/traps-6.0-technology-overview.pdf. Accessed 28 Aug. 2020.

[4] "Software Engineering | Agile Development Models ...." 6 Jul. 2018, https://www.geeksforgeeks.org/software-engineering-agile-development-models/. Accessed 28 Aug. 2020.

[5] "Managing Risk in Information Systems, 2nd Edition [Book]." https://www.oreilly.com/library/view/managing-risk-in/9781284055955/. Accessed 28 Aug. 2020.

[6] David Kim, Michael Solomon. (2012). Fundamentals of Information Systems Security. Jones & Bartlett Publishers

[7] "African Cyber Risk Institute - acriafrica.com." http://acriafrica.com/risks.htm. Accessed 28 Aug. 2020.

[8] "Sysmon - Windows Sysinternals | Microsoft Docs." 15 Jul. 2020, https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon. Accessed 28 Aug. 2020.

[9] "SwiftOnSecurity/sysmon-config - GitHub." https://github.com/SwiftOnSecurity/sysmon-config. Accessed 28 Aug. 2020

[10] "Beats - Elastic." https://www.elastic.co/beats/. Accessed 28 Aug. 2020.

[11] "VirusShare.com." https://virusshare.com/. Accessed 28 Aug. 2020.

[12] "A random forest approach - Veterinary Research - BioMed ...." 24 Jul. 2015, https://veterinaryresearch.biomedcentral.com/articles/10.1186/s13567-015-0219-7. Accessed 28 Aug. 2020

[13] Do Xuan Cho, Ha Hai Nam. A Method of Monitoring and Detecting APT Attacks Based on Unknown Domains. Pro. Com. Sci. 150, 316-323 (2019).

[14] "Tactics - Enterprise | MITRE ATT&CK ...." https://attack.mitre.org/tactics/. Accessed 28 Aug. 2020.

[15] “Sample 5ad401c3...90b7 behavior” https://www.vmray.com/analyses/5ad401c3a568/report/behavior_grouped.html

[16] "CAR-2013-04-002: Quick execution of a series of suspicious ...." 11 Apr. 2013, https://car.mitre.org/analytics/CAR-2013-04-002/. Accessed 28 Aug. 2020.

[17] "Neo23x0/sigma: Generic Signature Format for SIEM ... - GitHub." https://github.com/Neo23x0/sigma. Accessed 28 Aug. 2020.

[18] "What is Elasticsearch? | Elastic." https://www.elastic.co/what-is/elasticsearch. Accessed 28 Aug. 2020.

[19] "Getting Started - React." https://reactjs.org/docs/getting-started.html. Accessed 28 Aug. 2020.

[20] "About | Node.js." https://nodejs.org/en/about/. Accessed 28 Aug. 2020.

[21] "Reactstrap." https://reactstrap.github.io/. Accessed 28 Aug. 2020.

[22] "Your First Chart in React using FusionCharts | FusionCharts." https://www.fusioncharts.com/dev/getting-started/react/your-first-chart-using-react. Accessed 28 Aug. 2020.

[23] "react-d3-tree - npm." 16 Dec. 2019, https://www.npmjs.com/package/react-d3-tree. Accessed 28 Aug. 2020.