

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY



PROJECT 3

Implementing a SIEM System with the ELK Stack: Log Monitoring and Management

Student: Vũ Ngọc Quyền

Student ID: 20214971

Supervisor: Nguyễn Đức Toàn

School: Information and Communications Technology

HANOI, 2025

TABLE OF CONTENTS

1	Introduction	2
1.1	Overview of the Project	2
1.2	Problem Statement	2
1.3	Objectives and Scope	2
2	Theoretical Background	3
2.1	What is ELK Stack?	3
2.1.1	Elasticsearch	3
2.1.2	Logstash	4
2.1.3	Kibana	4
2.1.4	Elastic Beats	4
2.1.5	Common Functionalities of ELK Stack	5
2.2	Creating Detection Rules (MITRE ATT&CK)	6
2.2.1	MITRE ATT&CK	6
2.2.2	Theory of Detection Rules	7
2.2.3	Steps to Create Detection Rules	7
2.3	Machine Learning Models	7
2.3.1	Convolutional Neural Network (CNN)	7
2.3.2	Core Concepts of CNN	8
3	Methodology and Implementation	9
3.1	General Architecture of the System	9
3.2	Implementation Steps	9
3.2.1	System Requirements	9
3.2.2	Install and Configure ELK Stack	10
3.2.3	Log Collection Setup	15
3.2.4	Creating Detection Rules (MITRE ATT&CK)	20
3.2.5	Configuring Notification System	22
3.2.6	Integrating Machine Learning Model	23
4	Results and Assessment	25
4.1	Detection Rules Results	25
4.2	Notification System Results	26
4.3	CNN Model Training Results	27
5	Future work and conclusion	29
5.1	Future work	29
5.1.1	Integration of the CNN Model	29
5.1.2	Expansion of Detection Rules	30
5.1.3	Integration with YARA for Malware Detection	30
5.1.4	System Evaluation and Validation	31
5.2	Conclusion	31

List of Figures

1	ELK Stack	3
2	Logstash Pipeline	4
3	CNN Architecture	8
4	System Architecture	9
5	elasticsearch.yml - part 1	11
6	elasticsearch.yml - part 2	11
7	elasticsearch.yml - part 3	12
8	elasticsearch.yml - part 4	12
9	Test Elasticsearch connection over HTTPS	13
10	Logstash configuration	14
11	Log Collection Configuration in Winlogbeat	15
12	Winlogbeat Configuration for Sending Logs to Logstash	15
13	Kibana - View and analyze logs collected by Winlogbeat	16
14	auditbeat.yml - Add audit rules	17
15	auditbeat.yml - File integrity module configuration	17
16	auditbeat.yml - Output configuration for Logstash	17
17	Kibana - View and analyze logs collected by Auditbeat	18
18	filebeat.yml - Filestream input configuration to collect log messages	19
19	filebeat.yml - Output configuration for Logstash	19
20	Kibana - View and analyze logs collected by Filebeat	19
21	Detection Rules based on MITRE ATT&CK for Linux and Windows Systems	20
22	Detection Rule: Monitoring PsExec Service Modifications (T1543.003)	20
23	Detection Rule: Monitoring Service Installation CMD (T1543.003)	21
24	Detection Rule: Monitoring Password Policy Discovery (T1201)	21
25	Detection Rule: Monitoring SSH Brute Force Attempt (T1110.001)	21
26	Detection Rule: Monitoring File Integrity Change (T1070.004)	22
27	Connector type of Community Version	22
28	swatchdog-notify.sh	23
29	Overview of Detection Rules for Windows and Linux Systems	25
30	Email Notification Triggered by Matching Rules	27
31	Performance Metrics of the CNN Model	27
32	ROC curve	28
33	Proposed Approach for Distributed Training	30

1 Introduction

1.1 Overview of the Project

The project aims to design and implement a comprehensive Security Information and Event Management (SIEM) system using the ELK (Elasticsearch, Logstash, Kibana) Stack. The goal is to create a solution that can efficiently collect, store, and analyze security logs from various sources within an organization's infrastructure. By leveraging ELK Stack's capabilities, the system will enable real-time monitoring, detection to potential security threats, enhancing overall organizational security.

1.2 Problem Statement

In today's rapidly evolving digital environment, organizations face a growing volume and complexity of security threats, including cyberattacks, data breaches, and system vulnerabilities. The increasing reliance on digital systems and networks has made monitoring and managing security events more critical than ever. A traditional, manual approach to security monitoring is not only time-consuming but also prone to errors, which can lead to delayed responses or missed threats.

The need for an automated and efficient system to monitor, analyze, and respond to security incidents in real-time is imperative. This is where a Security Information and Event Management (SIEM) system comes into play. A SIEM system provides a centralized platform for collecting, storing, and analyzing logs from various sources across an organization's infrastructure, enabling better decision-making and faster detection of security breaches.

1.3 Objectives and Scope

The objectives of this project are to:

- Develop an automated SIEM solution using the ELK Stack.
- Collect logs from various sources, such as servers and security devices.
- Provide actionable insights through powerful data visualization in Kibana.
- Develop detection rules based on the MITRE ATT&CK framework to identify known attack patterns.
- Integrate machine learning models to enhance threat detection and prediction capabilities.

The scope of the project includes the installation and configuration of the ELK Stack, log collection from multiple sources, and setting up detection and notification mechanisms.

2 Theoretical Background

2.1 What is ELK Stack?

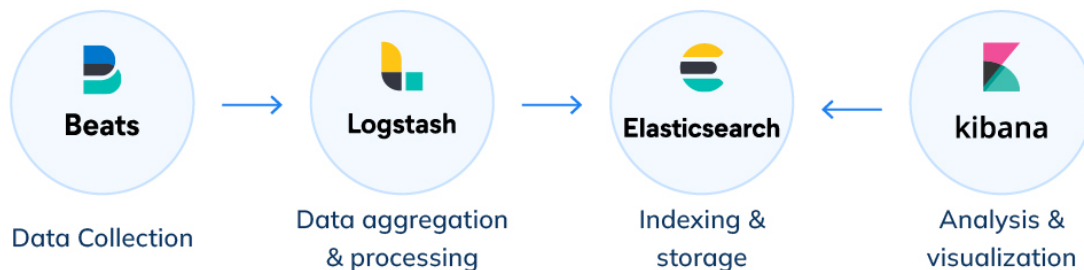


Fig. 1. ELK Stack

The ELK Stack is an open-source suite of tools designed for real-time log management and analytics. It consists of three main components: Elasticsearch, Logstash, and Kibana, which work together to collect, store, and visualize data. Additionally, Elastic Beats acts as a lightweight data shipper, enhancing the stack's flexibility by efficiently collecting and forwarding logs and metrics from various sources to the ELK pipeline.

2.1.1 Elasticsearch

Elasticsearch is a RESTful distributed search engine. It basically provides distributed search capabilities via API and stores data in a NoSQL database. Elasticsearch allows you to perform and combine many types of searches: structured, unstructured, geo, and metric in the way you want.

It stores, indexes, and makes log data searchable in near real-time, providing scalability and efficient querying for large volumes of data in a SIEM system.

2.1.2 Logstash

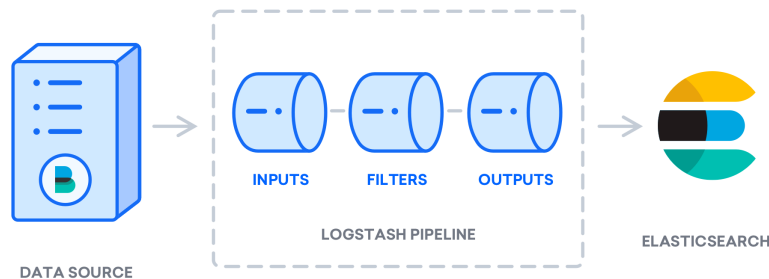


Fig. 2. Logstash Pipeline

Logstash collects and processes log data from multiple sources. It filters, enriches, and formats data before sending it to Elasticsearch for indexing, handling both structured and unstructured logs for security analysis.

2.1.3 Kibana

Kibana offers data visualization for the ELK Stack. It allows users to analyze and visualize log data through customizable dashboards, helping security teams track key metrics, detect anomalies, and respond to threats in real time.

2.1.4 Elastic Beats

The ELK Stack uses lightweight agents, known as Beats, to facilitate log collection from many systems. The Beats are lightweight data shippers, written in Go, that you install on your servers to capture all sorts of operational data (think of logs, metrics, or network packet data). The Beats send the operational data to Elasticsearch, either directly or via Logstash, so it can be visualized with Kibana. There are currently six official Beats from Elastic: Filebeat, Metricbeat, Packetbeat, Heartbeat, Winlogbeat, and Auditbeat.

- **Auditbeat:** Monitors Linux systems, capturing security-related events like file access and user activities. It's used to track system changes and suspicious behaviors through the Linux Audit framework.
- **Filebeat:** Collects logs from various sources, such as system files and network devices, across Linux and Windows. It forwards this log data to Logstash or

Elasticsearch for analysis, simplifying log collection and ensuring centralized storage.

- **Winlogbeat:** Winlogbeat collects logs from Windows Event Logs, covering security events, application events, and system logs. One of its key features is its ability to integrate with Sysmon (System Monitor), a Windows system service that provides detailed information about process creation, network connections, file modifications, and other system activities.
- **Metricbeat:** As the name implies, Metricbeat is used to collect metrics from servers and systems. It is a lightweight platform dedicated to sending system and service statistics. Like Filebeat, Metricbeat includes modules to grab metrics from operating systems like Linux, Windows and Mac OS.
- **Heartbeat:** Heartbeat is a lightweight shipper for uptime monitoring. It monitors services basically by pinging them and then ships data to Elasticsearch for analysis and visualization. Heartbeat can ping using ICMP, TCP and HTTP. IT has support for TLS, authentication and proxies. Its efficient DNS resolution enables it to monitor every single host behind a load-balanced server.
- **Packetbeat:** Packetbeat, a lightweight network packet analyzer, monitors network protocols to enable users to keep tabs on network latency, errors, response times, SLA performance, user access patterns and more. With Packetbeat, data is processed in real time so users can understand and monitor how traffic is flowing through their network. Furthermore, Packetbeat supports multiple application layer protocols, including MySQL and HTTP.

2.1.5 Common Functionalities of ELK Stack

a) Centralized log management

- Collect logs from various sources, such as servers, applications, network devices, and cloud environments.
- Provides a unified platform for storing and analyzing logs, eliminating the need to manage logs in disparate systems.
- Ensures high availability (HA) and scalability for handling large volumes of log data. Elasticsearch can be deployed in a HA cluster architecture, thus, allows data accessing smoothly.

b) Real-Time Data Ingestion and Processing

- Logstash processes and parses data in real time, transforming it into customizable structured formats for storage in Elasticsearch.

- Supports multiple input methods, including file, syslog, database, and message queues.
- Enables the enrichment of data with additional metadata, such as geolocation and timestamps.

c) Data Visualization and Dashboarding

- Kibana offers a user-friendly platform for designing and personalizing dashboards and visual representations.
- Includes support for diverse visualization formats such as charts, heatmaps, tables, and timelines.
- Facilitates real-time tracking of system status, security incidents, and application performance.

d) Threat Detection and Alerting

- The ELK Stack is highly effective for identifying potential threats and generating alerts in real time.
 - Custom Rules and Alerts: Users can create specific detection rules tailored to their environment using tools like Kibana and Logstash. Alerts can be configured to trigger based on predefined thresholds or patterns matching MITRE ATT&CK techniques.
 - Automated Notification: Alerts can be sent to teams via some platforms such as Email, Slack, or webhook integrations, ensuring timely response to critical incidents.

2.2 Creating Detection Rules (MITRE ATT&CK)

2.2.1 MITRE ATT&CK

MITRE ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) is a framework developed by MITRE Corporation to describe the tactics and techniques that adversaries may use throughout an attack. This framework includes detailed attack techniques, helping security professionals better understand adversary behavior and how they conduct attacks.

MITRE ATT&CK is divided into several tactics, which represent the adversary's goals or objectives during an attack. These tactics describe what the adversary aims to achieve, such as initial access, execution, persistence, or data exfiltration. Each tactic encompasses a set of techniques - specific methods or actions adversaries use to achieve these goals.

2.2.2 Theory of Detection Rules

In security systems, detection rules help identify adversary behaviors or signs by analyzing collected logs. These rules are typically built upon the attack techniques from MITRE ATT&CK and are designed to search for abnormal indicators in system, application, and network data.

Detection rules can be implemented in monitoring systems such as the ELK Stack or other SIEM tools. These rules can be written using search syntaxes like Elasticsearch Query DSL or through widely used detection languages like Sigma Rules, to help identify suspicious actions in real-time.

2.2.3 Steps to Create Detection Rules

- **Select tactics and techniques from MITRE ATT&CK:** These tactics and techniques should be selected based on the threats that might affect your system.
- **Collect and analyze logs:** Ensure that the system collects sufficient log types from various sources to support the detection of adversary behaviors.
- **Write and configure detection rules:** Detection rules can be written using Elasticsearch query language or other tools, based on the techniques within MITRE ATT&CK you wish to monitor.
- **Deploy and test rules:** After writing the rules, they need to be deployed to the system and tested for effectiveness through simulated attack scenarios or real-world situations.
- **Optimize and maintain rules:** Detection rules need to be continuously maintained and optimized to reflect new threats.

2.3 Machine Learning Models

Machine learning (ML) is a subfield of artificial intelligence (AI) that enables computers to learn from data and make decisions without explicit programming. A variety of machine learning models exist, each designed to solve different types of problems. In this section, we focus on one of the most influential deep learning models: the Convolutional Neural Network (CNN).

2.3.1 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) are a specialized type of deep neural network designed for processing structured grid-like data, such as images. They are especially powerful for tasks such as image recognition, object detection, and other computer vision tasks. CNN has been highly successful in applications such as self-driving cars, facial recognition, and medical image analysis.

2.3.2 Core Concepts of CNN

Convolutional Neural Network (CNN) is a type of Deep Neural Network widely used in pattern and image recognition. Each CNN has important layers under the name:

- **Input Layer:** This layer takes the raw data, such as images, and passes it to the convolutional layers.
- **Convolutional Layers:** These layers apply filters (also known as kernels) to input data (e.g., images). The filter slides over the input image to extract features like edges, textures, and patterns. The result of the convolution is a feature map that represents various aspects of the input data.
- **Activation Function:** After applying the convolution operation, the output is passed through an activation function, typically a Rectified Linear Unit (ReLU), to introduce non-linearity into the model and allow it to learn more complex patterns.
- **Pooling Layers:** Pooling layers, such as max pooling, are used to reduce the spatial dimensions (height and width) of the input data. This reduces the number of parameters, thus reducing computation and preventing overfitting. Max pooling, for example, takes the maximum value from a set of values in the feature map, effectively down-sampling the data.
- **Fully Connected Layers:** These layers are similar to those found in traditional neural networks. After feature extraction, the flattened data is passed to fully connected layers for final classification or regression tasks.
- **Output Layer:** The final layer produces the result (e.g., classification labels).

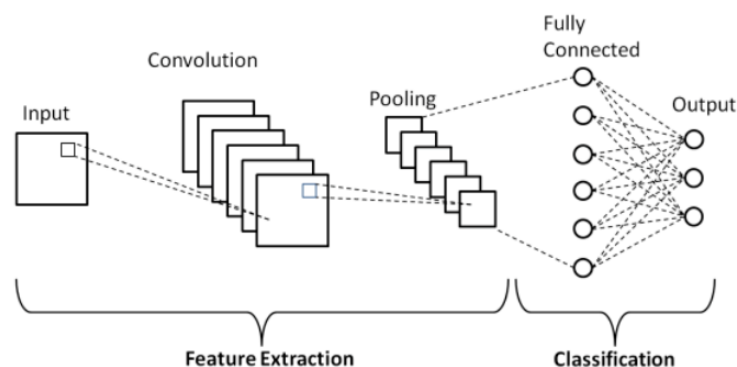


Fig. 3. CNN Architecture

In the context of ELK Stack and log monitoring, CNNs can be applied to automate anomaly detection and classify logs based on patterns, providing valuable insights for monitoring systems.

3 Methodology and Implementation

3.1 General Architecture of the System

Machine Type	Operation System	Installed Tools
Elasticsearch Server	Ubuntu Server 22.04	ElasticSearch, Kibana, Logstash
Log collector from Linux	Kali Linux	Filebeat, Syslog, Auditbeat
Log collector from Windows	Windows Server or Windows 10	Sysmon, Winlogbeat

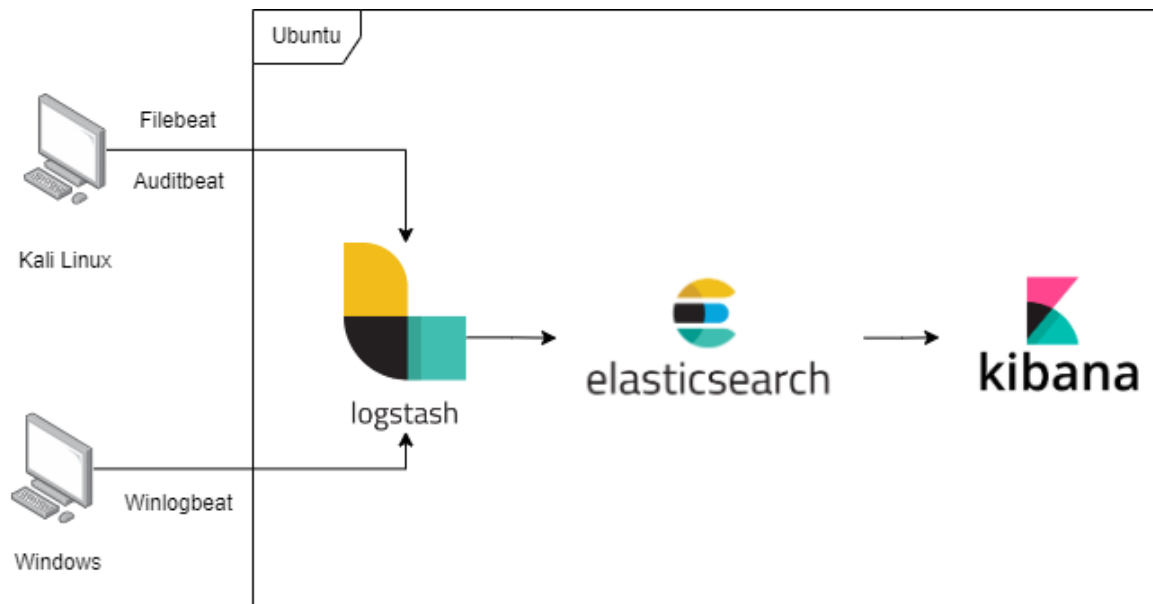


Fig. 4. System Architecture

3.2 Implementation Steps

3.2.1 System Requirements

Server:

- Operating System: Ubuntu 22.04
- Memory: Minimum 4GB (Recommended 8GB for production)
- Disk Space: At least 50GB for storing logs and indices

- Network: Proper connectivity between client and server
- Functionalities:
 - Elasticsearch 8.15 runs at port 9200
 - Logstash 8.15 runs at port 5044
 - Kibana 8.15 runs at port 5601

Log Sources:

- Windows:
 - Winlogbeat for Windows installation
 - Supported Windows versions (Windows 10/11, Server 2016/2019/2022)
- Linux:
 - Filebeat for Linux installation
 - Supported distributions (CentOS, Debian, Ubuntu, etc.)

3.2.2 Install and Configure ELK Stack

a) Elasticsearch

Import the Elastic PGP key (skip this step if already installed packages from Elastic)

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch |  
sudo gpg --dearmor -o /usr/share/keyrings/elasticsearch-  
keyring.gpg
```

Install apt-transport-https package

```
sudo apt install apt-transport-https
```

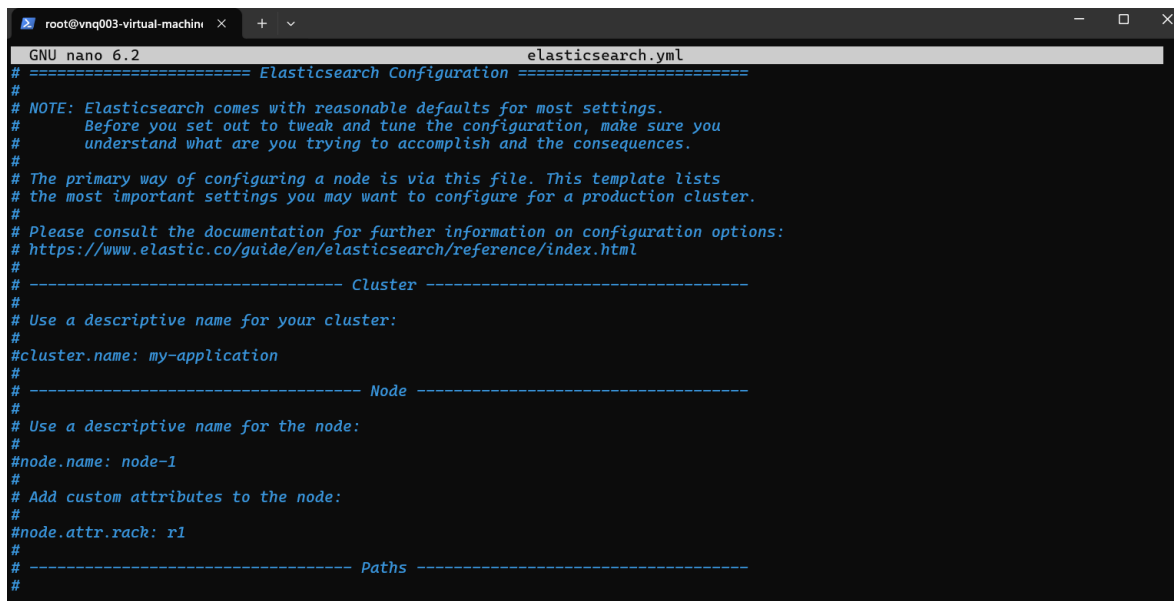
Save the repo, update the repo, and install the package

```
echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.  
gpg] https://artifacts.elastic.co/packages/8.x/apt stable  
main" | sudo tee /etc/apt/sources.list.d/elasticsearch-8.x.list  
sudo apt-get update && sudo apt-get install elasticsearch
```

Enable the service to start automatically on boot

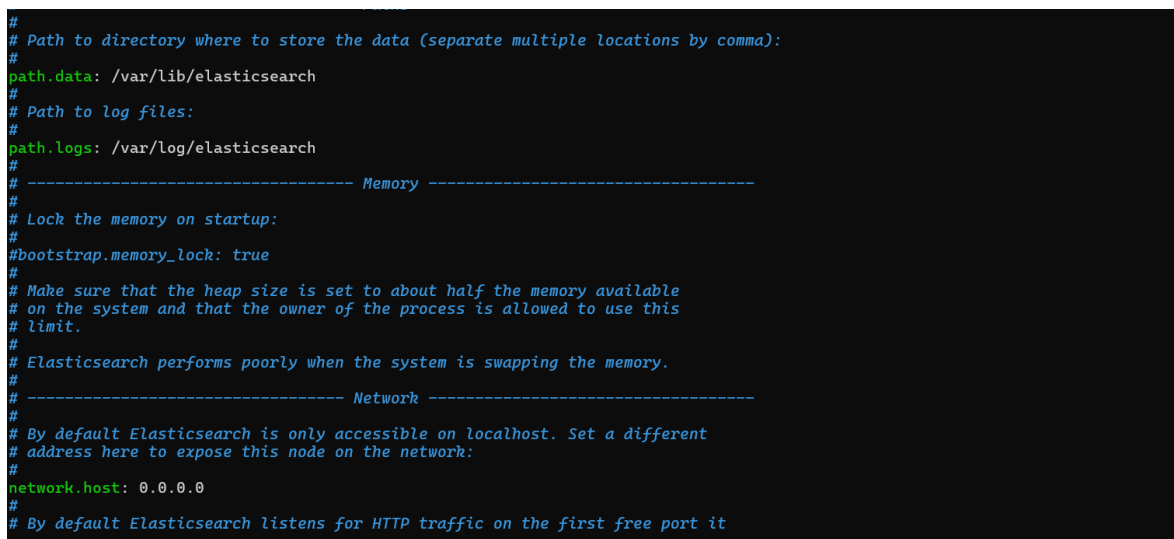
```
sudo systemctl daemon-reload  
sudo systemctl enable elasticsearch
```

I config file elasticsearch.yml as the following:



```
root@vnq003-virtual-machine: ~
GNU nano 6.2 elasticsearch.yml
# ===== Elasticsearch Configuration =====
#
# NOTE: Elasticsearch comes with reasonable defaults for most settings.
#       Before you set out to tweak and tune the configuration, make sure you
#       understand what are you trying to accomplish and the consequences.
#
# The primary way of configuring a node is via this file. This template lists
# the most important settings you may want to configure for a production cluster.
#
# Please consult the documentation for further information on configuration options:
# https://www.elastic.co/guide/en/elasticsearch/reference/index.html
#
# ----- Cluster -----
#
# Use a descriptive name for your cluster:
#
#cluster.name: my-application
#
# ----- Node -----
#
# Use a descriptive name for the node:
#
#node.name: node-1
#
# Add custom attributes to the node:
#
#node.attr.rack: r1
#
# ----- Paths -----
#
```

Fig. 5. elasticsearch.yml - part 1



```
#
# Path to directory where to store the data (separate multiple locations by comma):
#
#path.data: /var/lib/elasticsearch
#
# Path to log files:
#
#path.logs: /var/log/elasticsearch
#
# ----- Memory -----
#
# Lock the memory on startup:
#
#bootstrap.memory_lock: true
#
# Make sure that the heap size is set to about half the memory available
# on the system and that the owner of the process is allowed to use this
# limit.
#
# Elasticsearch performs poorly when the system is swapping the memory.
#
# ----- Network -----
#
# By default Elasticsearch is only accessible on localhost. Set a different
# address here to expose this node on the network:
#
#network.host: 0.0.0.0
#
# By default Elasticsearch listens for HTTP traffic on the first free port it
```

Fig. 6. elasticsearch.yml - part 2

```
# By default Elasticsearch listens for HTTP traffic on the first free port it
# finds starting at 9200. Set a specific HTTP port here:
#
http.port: 9200
#
# For more information, consult the network module documentation.
#
# ----- Discovery -----
#
# Pass an initial list of hosts to perform discovery when this node is started:
# The default list of hosts is ["127.0.0.1", "[:,1]"]
#
#discovery.seed_hosts: ["host1", "host2"]
#
# Bootstrap the cluster using an initial set of master-eligible nodes:
#
#cluster.initial_master_nodes: ["node-1", "node-2"]
#
# For more information, consult the discovery and cluster formation module documentation.
#
# ----- Various -----
#
# Allow wildcard deletion of indices:
#
#action.destructive_requires_name: false

#----- BEGIN SECURITY AUTO CONFIGURATION -----
#
# The following settings, TLS certificates, and keys have been automatically
# generated to configure Elasticsearch security features on 03-10-2024 19:13:48
```

Fig. 7. elasticsearch.yml - part 3

```
# Enable security features
xpack.security.enabled: true

xpack.security.enrollment.enabled: true

# Enable encryption for HTTP API client connections, such as Kibana, Logstash, and Agents
xpack.security.http.ssl:
  enabled: true
  keystore.path: certs/http.p12

# Enable encryption and mutual authentication between cluster nodes
xpack.security.transport.ssl:
  enabled: true
  verification_mode: certificate
  keystore.path: certs/transport.p12
  truststore.path: certs/transport.p12
# Create a new cluster with the current node only
# Additional nodes can still join the cluster later
cluster.initial_master_nodes: ["vnq003-virtual-machine"]

# Allow HTTP API connections from anywhere
# Connections are encrypted and require user authentication
http.host: 0.0.0.0

# Allow other nodes to join the cluster from anywhere
# Connections are encrypted and mutually authenticated
#transport.host: 0.0.0.0

#----- END SECURITY AUTO CONFIGURATION -----
```

Fig. 8. elasticsearch.yml - part 4

In Elasticsearch 8.x (including 8.15), self-signed certificates are automatically generated during the default installation. Elasticsearch automatically generates a CA certificate and a TLS certificate in the configuration directory:

```
/etc/elasticsearch/certs/
```

This certificate includes:

- http_ca.crt: CA certificate used to authenticate HTTPS connections.
- http.p12: PKCS#12 file containing the certificate and private key used for the HTTPS protocol.

To test Elasticsearch connection over HTTPS with CA certificate, using this command:

```
curl --cacert /etc/elasticsearch/certs/http_ca.crt -u elastic:
$ELASTIC_PASSWORD https://localhost:9200
```

```
root@vnq003-virtual-machine:/home/vnq003# curl --cacert /etc/elasticsearch/certs/http_ca.crt -u elastic:
https://localhost:9200
{
  "name" : "vnq003-virtual-machine",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "WMcdjvftSgKRuTwt6_c-Gg",
  "version" : {
    "number" : "8.15.2",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "98adf7bf6bb69b66ab95b761c9e5aadb0bb059a3",
    "build_date" : "2024-09-19T10:06:03.564235954Z",
    "build_snapshot" : false,
    "lucene_version" : "9.11.1",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

Fig. 9. Test Elasticsearch connection over HTTPS

b) Logstash

Import the Elastic PGP key (skip this step if already installed packages from Elastic)

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch |
sudo gpg --dearmor -o /usr/share/keyrings/elasticsearch-
keyring.gpg
```

Install apt-transport-https package

```
sudo apt install apt-transport-https
```

Save the repo, update the repo, and install the package

```
echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.
gpg] https://artifacts.elastic.co/packages/8.x/apt stable
main" | sudo tee /etc/apt/sources.list.d/elastic-8.x.list
sudo apt-get update && sudo apt-get install logstash
```

I config logstash as the following:

```
# Beats -> Logstash -> Elasticsearch pipeline.

input {
  beats {
    port => 5044
  }
}

output {
  elasticsearch {
    hosts => ["https://localhost:9200"]
    index => "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
    user => "elastic"
    password => "bw=Y7BGzOXlm4_v1B0I7"
    ssl => true
    cacert => "/etc/logstash/certs/http_ca.crt"
  }
}
```

Fig. 10. Logstash configuration

To set up an HTTPS connection between Logstash and Elasticsearch, you need to copy the http_ca.crt certificate file from Elasticsearch to the Logstash folder.

```
cp /etc/elasticsearch/certs/http_ca.crt /etc/logstash/certs/
```

c) Kibana

Import the Elastic PGP key (skip this step if already installed packages from Elastic)

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch |
sudo gpg --dearmor -o /usr/share/keyrings/elasticsearch-
keyring.gpg
```

Install apt-transport-https package

```
sudo apt install apt-transport-https
```

Save the repo, update the repo, and install the package

```
echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.
gpg] https://artifacts.elastic.co/packages/8.x/apt stable
main" | sudo tee /etc/apt/sources.list.d/elastic-8.x.list

sudo apt-get update && sudo apt-get install kibana
```

To configure Kibana to start automatically when the system starts, run the following commands:

```
sudo systemctl daemon-reload
sudo systemctl enable kibana.service
```


3.2.3 Log Collection Setup

Log collection involves setting up tools on both Windows and Linux machines to gather logs and send them to the ELK server for processing and visualization. This section details the configuration of Winlogbeat for Windows machines and Auditbeat and Filebeat for Linux machines.

a) Windows Machine

For Windows machines, Winlogbeat is used to collect event logs, such as application, system, and security logs. Steps to Configure Winlogbeat:

- Download and install Winlogbeat from the Elastic website, then extract the contents of the ZIP file into a directory.
- Configure winlogbeat.yml:
 - Specify the logs to be collected

```
winlogbeat.event_logs:
- name: Application
  ignore_older: 72h
- name: System
- name: Security
- name: Microsoft-Windows-Sysmon/Operational
- name: Windows PowerShell
  event_id: 400, 403, 600, 800
- name: Microsoft-Windows-PowerShell/Operational
  event_id: 4103, 4104, 4105, 4106
- name: ForwardedEvents
  tags: [forwarded]
```

Fig. 11. Log Collection Configuration in Winlogbeat

- Configure the output to Logstash

```
# ----- Logstash Output -----
output.logstash:
# The Logstash hosts
hosts: ["192.168.174.139:5044"]
```

Fig. 12. Winlogbeat Configuration for Sending Logs to Logstash

- Run Winlogbeat: Use PowerShell to install Winlogbeat as a service and start the Winlogbeat service:

```
.\install-service-winlogbeat.ps1
Start-Service winlogbeat
```

– Verify Log Collection

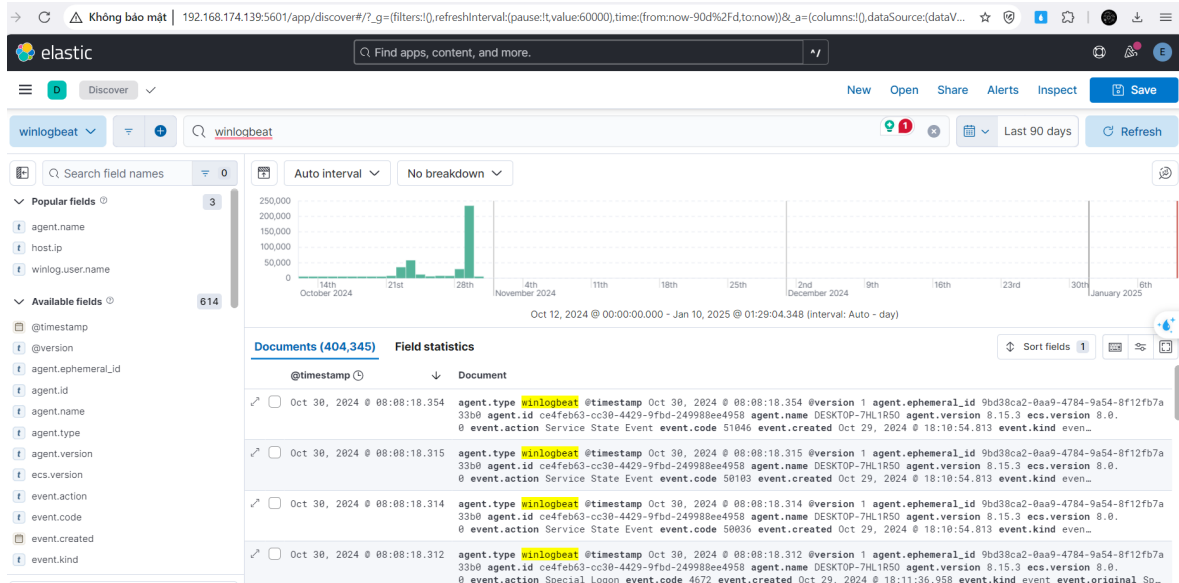


Fig. 13. Kibana - View and analyze logs collected by Winlogbeat

b) Linux Machine

For Linux machines, Auditbeat and Filebeat are used to monitor system activities and collect log files, respectively.

Auditbeat Configuration:

- Install Auditbeat
 - Import the Elastic PGP key (skip this step if already installed packages from Elastic)

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor -o /usr/share/keyrings/elasticsearch-keyring.gpg
```

- Install apt-transport-https package

```
sudo apt install apt-transport-https
```

- Save the repo, update the repo, and install the package

```
echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.gpg] https://artifacts.elastic.co/packages/8.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-8.x.list
```

```
sudo apt-get update && sudo apt-get install auditbeat
```

- Configure auditbeat.yml:

- Add audit_rules to monitor access and changes to files related to password policies.

```
GNU nano 8.0 auditbeat.yml
- module: auditd
  # Load audit rules from separate files. Same format as audit.rules(7).
  audit_rule_files: [ '${path.config}/audit.rules.d/*.conf' ]
  audit_rules: |
    ## Define audit rules here.
    ## Create file watches (-w) or syscall audits (-a or -A). Uncomment these
    ## examples or add your own rules.

    ## If you are on a 64 bit platform, everything should be running
    ## in 64 bit mode. This rule will detect any use of the 32 bit syscalls
    ## because this might be a sign of someone exploiting a hole in the 32
    ## bit API.
    #-a always,exit -F arch=b32 -S all -F key=32bit-abi

    ## Executions.
    #-a always,exit -F arch=b64 -S execve,execveat -k exec

    ## External access (warning: these can be expensive to audit).
    #-a always,exit -F arch=b64 -S accept,bind,connect -F key=external-access

    # Monitor reading of /etc/pam.d/common-password
    -w /etc/pam.d/common-password -p r -k password_policy

    # Monitor reading of /etc/pam.d/passwd
    -w /etc/pam.d/passwd -p r -k password_policy

    # Monitor reading of /etc/security/pwquality.conf
    -w /etc/security/pwquality.conf -p r -k password_policy

    # Monitor reading of /etc/login.defs
    -w /etc/login.defs -p r -k password_policy

    ## Identity changes.
    -w /etc/group -p wa -k identity
```

Fig. 14. auditbeat.yml - Add audit rules

- Enable the file_integrity module to monitor critical directories

```
- module: file_integrity
  paths:
    - /bin
    - /usr/bin
    - /sbin
    - /usr/sbin
    - /etc
```

Fig. 15. auditbeat.yml - File integrity module configuration

- Configure the output to send data to Logstash

```
# ----- Logstash Output -----
output.logstash:
  # The Logstash hosts
  hosts: ["192.168.174.139:5044"]
```

Fig. 16. auditbeat.yml - Output configuration for Logstash

- Start Auditbeat

- To start Auditbeat, run

```
sudo service auditbeat start
```

- Verify Log Collection

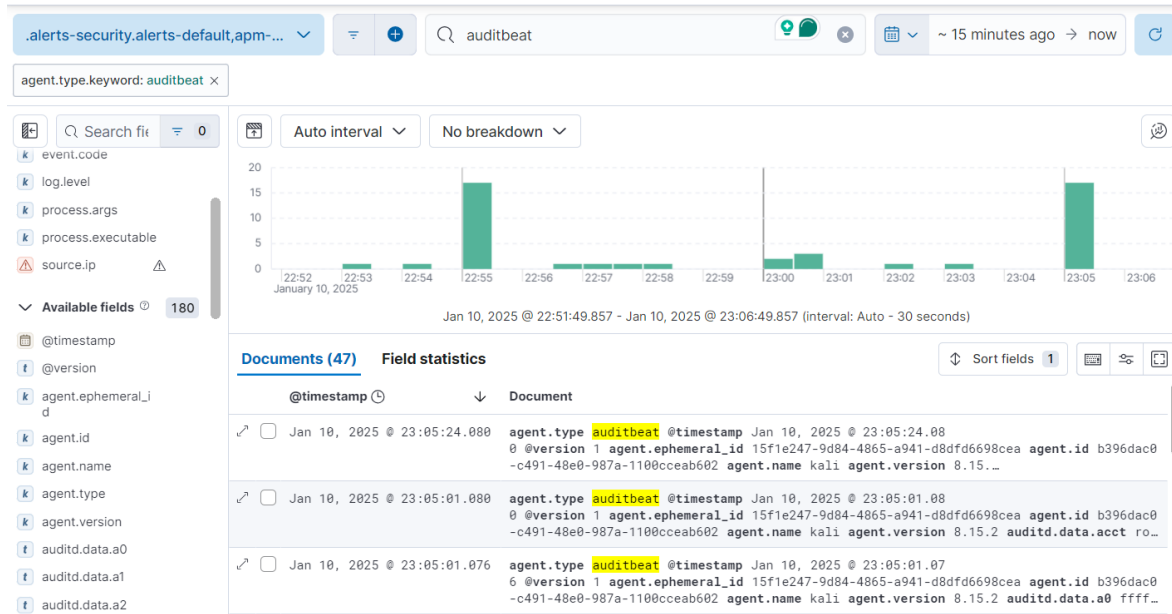


Fig. 17. Kibana - View and analyze logs collected by Auditbeat

Filebeat Configuration:

- Install Filebeat

- Import the Elastic PGP key (skip this step if already installed packages from Elastic)

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor -o /usr/share/keyrings/elasticsearch-keyring.gpg
```

- Install apt-transport-https package

```
sudo apt install apt-transport-https
```

- Save the repo, update the repo, and install the package

```
echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.gpg] https://artifacts.elastic.co/packages/8.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-8.x.list
```

```
sudo apt-get update && sudo apt-get install filebeat
```

- Configure filebeat.yml:
 - Add a filestream input to collect log messages from specific files.

```
# ----- Filebeat inputs -----
filebeat.inputs:
# Each - is an input. Most options can be set at the input level, so
# you can use different inputs for various configurations.
# Below are the input-specific configurations.
# filestream is an input for collecting log messages from files.
- type: filestream
  # Unique ID among all inputs, an ID is required.
  id: my-filestream-id
  # Change to true to enable this input configuration.
  enabled: true
  # Paths that should be crawled and fetched. Glob based paths.
  paths:
    - /var/log/syslog
    #- c:\programdata\elasticsearch\logs\*
```

Fig. 18. filebeat.yml - Filestream input configuration to collect log messages

- Configure the output to send data to Logstash

```
# ----- Logstash Output -----
output.logstash:
# The Logstash hosts
hosts: ["192.168.174.139:5044"]
```

Fig. 19. filebeat.yml - Output configuration for Logstash

- Start Filebeat
 - To start Auditbeat, run

```
sudo service filebeat start
```

- Verify Log Collection

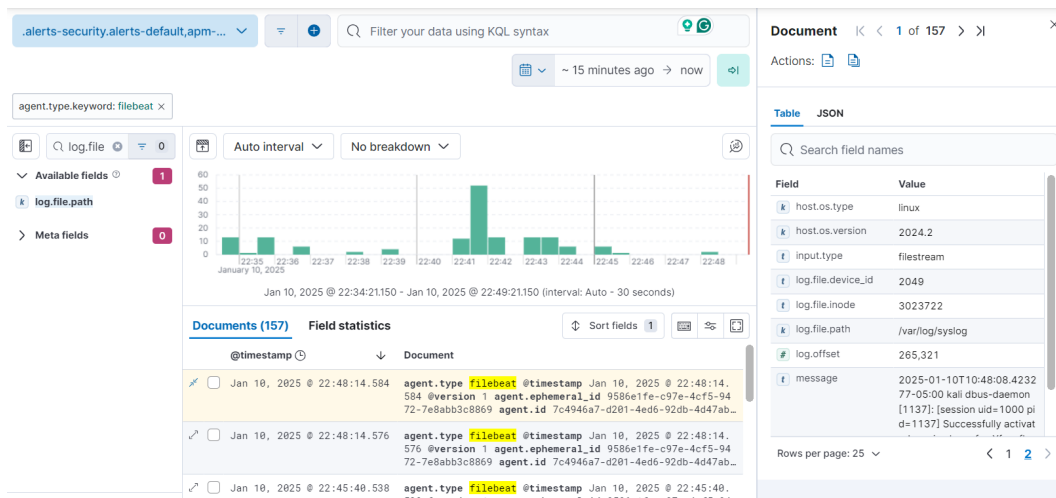
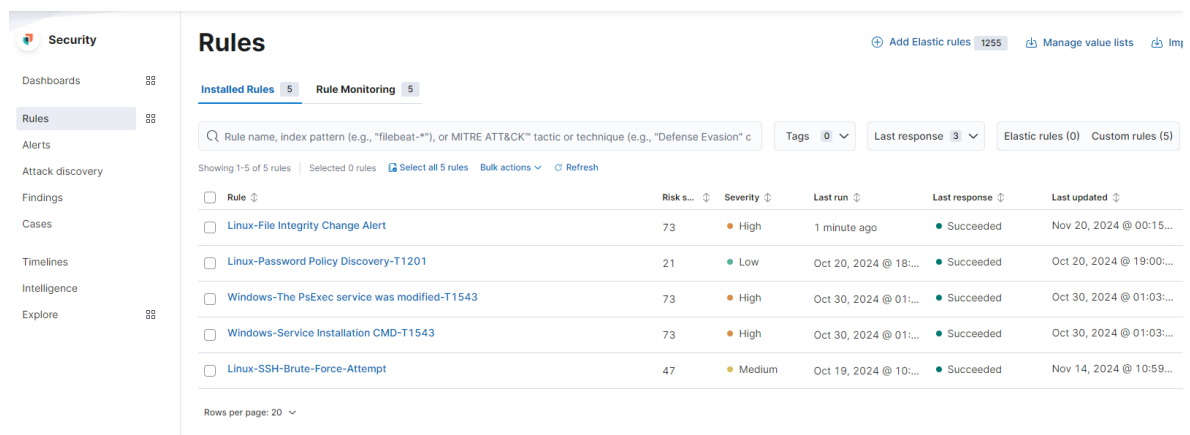


Fig. 20. Kibana - View and analyze logs collected by Filebeat

3.2.4 Creating Detection Rules (MITRE ATT&CK)

I have written 5 detection rules based on the MITRE ATT&CK framework for Linux and Windows systems. The objective is to monitor and detect suspicious activity through custom queries tailored to specific techniques and behaviors relevant to the system's environment.



Rule	Risk s...	Severity	Last run	Last response	Last updated
Linux-File Integrity Change Alert	73	High	1 minute ago	Succeeded	Nov 20, 2024 @ 00:15...
Linux-Password Policy Discovery-T1201	21	Low	Oct 20, 2024 @ 18:...	Succeeded	Oct 20, 2024 @ 19:00:...
Windows-The PsExec service was modified-T1543	73	High	Oct 30, 2024 @ 01:...	Succeeded	Oct 30, 2024 @ 01:03:...
Windows-Service Installation CMD-T1543	73	High	Oct 30, 2024 @ 01:...	Succeeded	Oct 30, 2024 @ 01:03:...
Linux-SSH-Brute-Force-Attempt	47	Medium	Oct 19, 2024 @ 10:...	Succeeded	Nov 14, 2024 @ 10:59...

Fig. 21. Detection Rules based on MITRE ATT&CK for Linux and Windows Systems

a) Detection Rule for Windows Systems

- **Technique:** T1543.003 - The PsExec service was modified
 - Description: This rule detects modifications to the PsExec service, a technique often used by attackers to establish persistence or execute commands remotely.
 - Custom Query:

Index patterns

logs-* winlogbeat-*

Custom query

```
winlog.event_data.EventType : "SetValue" and
winlog.event_data.TargetObject:
"HKLM\\System\\CurrentControlSet\\Services\\PSEXES
VC*" and winlog.event_data.User : "NT
AUTHORITY\\SYSTEM"
```

Fig. 22. Detection Rule: Monitoring PsExec Service Modifications (T1543.003)

- **Technique:** T1543.003 - Service Installation CMD
 - Description: This rule identifies the installation or modification of services using command-line tools, which attackers commonly exploit to gain persistence or execute malicious payloads. Monitoring these activities enables early detection of unauthorized service creation or changes, reducing the risk of further compromise.

– Custom Query:

Index patterns

logs-* winlogbeat-*

Custom query

```
winlog.event_data.ParentImage :
"C:\\Windows\\System32\\cmd.exe" and
winlog.event_data.Image :
"C:\\Windows\\System32\\sc.exe" and
winlog.event_data.CommandLine : "sc.exe create **"
```

Fig. 23. Detection Rule: Monitoring Service Installation CMD (T1543.003)

b) Detection Rule for Linux Systems

• **Technique:** T1201 - Password Policy Discovery

- Description: This rule detects attempts to gather information about password policies on Linux systems. Activities may include querying configuration files or using commands to check password rules.
- Custom Query:

Index patterns

auditbeat-* filebeat-*

Custom query

```
agent.type: "auditbeat" and event.module : "auditd"
and event.action: "opened-file" and event.outcome:
"success" and (file.path : "*login.defs" or file.path:
"*common-password" or file.path: "*pwquality.conf" or
file.path: "*passwd") and process.executable :
"/usr/bin*" and not ( process.name : "su" or
process.name: "sudo" )
```

Fig. 24. Detection Rule: Monitoring Password Policy Discovery (T1201)

• **Technique:** T1110.001 - SSH Brute Force Attempt

- Description: This rule alerts on brute force attack attempts via the SSH protocol, where an attacker tries to gain access by testing multiple passwords in a short period.
- Custom Query:

Index patterns

auditbeat-* filebeat-* logs-* packetbeat-*

Custom query

```
event.category: authentication and event.outcome:
failure and agent.type: auditbeat and agent.name: kali
and auditd.summary.how: "/usr/sbin/sshd"
```

Fig. 25. Detection Rule: Monitoring SSH Brute Force Attempt (T1110.001)

- **Technique:** T1070.004 - File Integrity Change
 - Description: This rule detects changes to system or application files. Monitoring file integrity helps identify unusual activities, such as attacks or intrusions on the system.
 - Custom Query:

Index patterns

auditbeat-* filebeat-*

Custom query

```
event.category: file and (event.type: creation or
event.type: change or event.type: deletion) and
agent.type: auditbeat and agent.name: kali and
file.path: "/usr/bin/*"
```

Fig. 26. Detection Rule: Monitoring File Integrity Change (T1070.004)

3.2.5 Configuring Notification System

Problem: To send alert notifications, Elastic Stack requires Gold License to connect and send to multiple components. Community edition only supports creating Index and Server logs.

Solution: To send to email and other notification services, we will use bash script in Linux, in which we will use **swatch** to listen for alert events

Select a connector type

[Get more connectors](#)

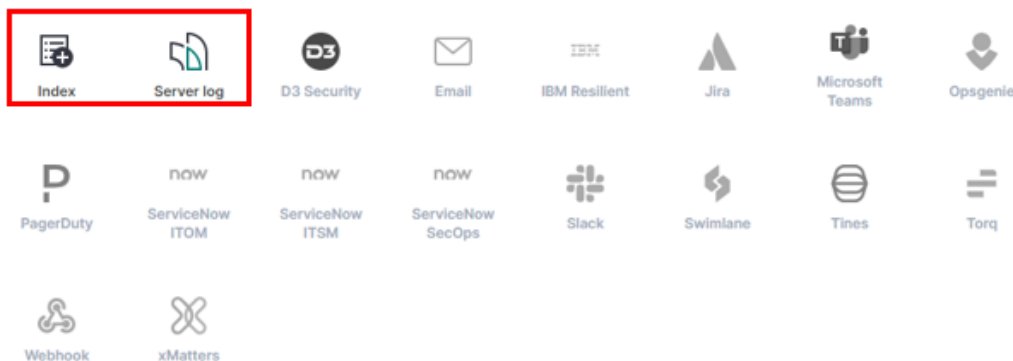


Fig. 27. Connector type of Community Version

Introduction to Swatch: Swatch (Simple Watcher) is a log monitoring tool designed to watch log files and send notifications when matching patterns are detected. It is widely used with the Elastic Stack to monitor log files and send alert notifications when specific events occur.

Installation: On Debian/Ubuntu systems, Swatch can be installed using the following command:


```
sudo apt-get install swatchdog
```

Configuration: Create a file for Swatchdog (e.g., /etc/swatch.conf) to specify the log files to monitor and the alerting criteria. This file should define the patterns to look for and the corresponding actions to take when a match is found.

Bash Script to Handle Alerts: Below is a sample bash script, **swatchdog-notify.sh**, which processes log alerts and sends email notifications. This script does the following:

- Reads the log data.
- Extracts information from the log (such as Severity, Source IP, Host IP, Timestamp, Message, etc.).
- Creates an email notification content and sends it via Postfix.

```
GNU nano 6.2 /root/scripts/swatchdog-notify.sh *
#!/bin/bash
echo $1 > message.txt
message=$(jq ".message" message.txt)

# Check if the message contains 'Server log'
if [[ $(echo "$message" | grep -i "Server log") ]]; then
    # Extract information using grep and regex
    severity=$(echo "$message" | grep -oP '(?<=Severity: ).*(?=;)' )
    description=$(echo "$message" | grep -oP '(?<=Description: ).*(?=;)' )
    source_ip=$(echo "$message" | grep -oP '(?<=Source IP: ).*(?=;)' )
    host_ip=$(echo "$message" | grep -oP '(?<=Host IP: ).*(?=;)' )

    # Set default values if any fields are missing
    severity="${severity:-Unknown}"
    source_ip="${source_ip:-Not Available}"
    host_ip="${host_ip:-Not Available}"
    timestamp=$(jq -r '.@timestamp' message.txt)

    # Create the email content using the template
    CONTENT=$(sed -e 's/{{monitor}}/'"$monitor"'/g' -e 's/{{status}}/'"$status"'/g' -e 's/{{message}}/'"$message"'/g' \
-e 's/{{color}}/'"$color"'/g' -e 's/{{color2}}/'"$color2"'/g' -e 's/{{timestamp}}/'"$timestamp"'/g' /root/scripts/template/template_email_alert.html)

    # Send the email using 'mail' with Postfix
    echo "$CONTENT" | mail -a "Content-Type: text/html" -s "Log Monitor Alert - Severity: $severity" -r "hustedutech@gmail.com" "quyenvungo"

    echo "Email sent successfully for alert with severity: $severity"
    echo "$timestamp"
else
    echo "Log does not match the expected format"
fi
```

Fig. 28. swatchdog-notify.sh

3.2.6 Integrating Machine Learning Model

This section gives information about the integration of a Convolutional Neural Network (CNN) into the methodology for analyzing and classifying URL and payload data. The CNN model is designed to process one-dimensional input sequences, where the input consists of tokenized and padded URLs. After processing these sequences, the model produces a classification output based on the given data.

Input Representation:

- The URLs and payloads are **tokenized**, which means they are split into smaller components. For example, a URL like <http://example.com/path?query=1> would be broken down into individual characters such as *h*, *t*, *t*, *p*, *:*, */*, */*, *e*, *x*, *a*, *m*, *p*, *.*, *e*, *.* and so on. Once the URL and payload is split into tokens, each token is then converted into a numerical representation that can be understood by the machine learning model.

- **Padding** is applied to ensure that all input sequences have the same length, which is essential for feeding them into the CNN.
- Instead of converting the URL and payload into an image and then into pixel matrices (which would require extra processing), we directly convert them into matrices of numbers using tokenization and encoding techniques. This eliminates the need for an additional conversion step and allows the model to process the data efficiently.

Layer	Parameters	Activation	Description
Embedding Layer	Embedding(input_dim = num_words, output_dim=64, input_length=max_len)	None	Maps tokenized integers to dense 64-dimensional vectors, learning contextual representations.
Conv1D Layer 1	Conv1D(filters=256, kernel_size=7)	ReLU	Applies 256 filters with kernel size 7 to extract local patterns; uses ReLU activation.
Max Pooling Layer	MaxPooling1D(pool_size = 3)	None	Reduces dimensionality by taking the maximum value over a pooling window of size 3.
Global Max Pooling	GlobalMaxPooling1D()	None	Reduces each feature map to a single vector, representing the most salient features.
Dense Layer	Dense(units=2)	Softmax	Outputs a probability distribution over two classes using a softmax activation function.

- **ReLU (Rectified Linear Unit):** Used in Conv1D layers to create non-linearity, helping the model learn more complex features.
- **Softmax:** Activation function in the last layer, used to convert the output into class probabilities.

4 Results and Assessment

4.1 Detection Rules Results

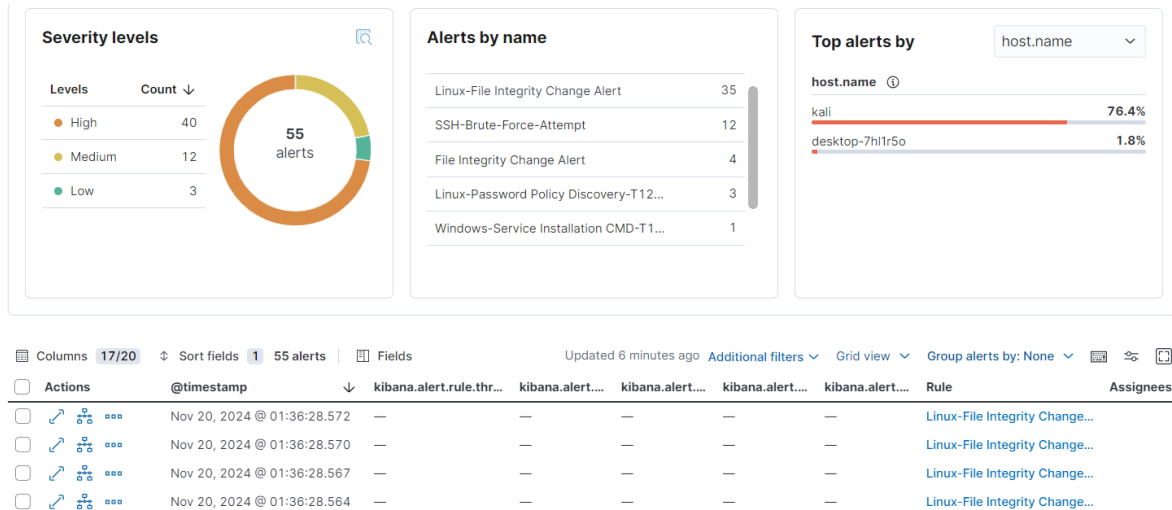


Fig. 29. Overview of Detection Rules for Windows and Linux Systems

Detection Rule for Windows Systems

- **Technique:** T1543.003 - The PsExec service was modified
 - **Effectiveness:** The rule successfully detected modifications to the PsExec service, which is commonly used by attackers for remote execution and persistence. The alerts were triggered immediately after the modification occurred.
 - **Performance:** The rule processed the logs efficiently without noticeable delay, ensuring real-time detection of any malicious activities related to PsExec service modifications.
- **Technique:** T1543.003 - Service Installation CMD
 - **Effectiveness:** The rule successfully detected any unauthorized installations or modifications of services via command-line tools, an attack vector commonly used for persistence. Alerts were consistently triggered when suspicious activities were identified.
 - **Performance:** Log analysis was carried out with minimal delay, ensuring prompt detection of changes in services.

Detection Rule for Linux Systems

- **Technique:** T1201 - Password Policy Discovery

- **Effectiveness:** The rule effectively detected attempts to discover password policies on Linux systems. Alerts were triggered when suspicious activities, such as querying configuration files or password rule checks, were detected.
- **Performance:** Detection occurred with minimal delay in identifying potentially unauthorized actions.
- **Technique:** T1110.001 - SSH Brute Force Attempt
 - **Effectiveness:** The rule was effective in identifying brute force attempts on SSH services. Alerts were successfully triggered whenever multiple failed login attempts were detected within a short timeframe.
 - **Performance:** The rule processed login attempts in real-time, ensuring timely alerts for suspicious brute-force activities.
- **Technique:** T1070.004 - File Integrity Change
 - **Effectiveness:** The rule effectively detected any changes to critical system or application files, which are common indicators of a potential attack or compromise. Alerts were generated for file integrity violations.
 - **Performance:** File integrity monitoring was done efficiently, with quick identification of changes to sensitive files.

4.2 Notification System Results

- **Notification Setup:** The notification system was configured using swatchdog in combination with postfix to send alerts via email.
- **Effectiveness:** The notification system was effective in alerting the security team when detection rules identified suspicious activities. Alerts were successfully triggered for rule matches, providing timely notifications for quick action.



root <hustedutech@gmail.com>
to quyenvungoc1711

Nov 20, 2024, 1:36 AM ☆ 😊 ↩ ⋮

Log Monitor: ELK Server Alert Notification

State: **Available**

Service:

Message: "Server log: Rule Linux-File Integrity Change Alert generated 4 alerts;Severity: high;Timestamp: ;Description: This alert triggers when a monitored file is modified, deleted, or created. It is designed to detect unauthorized changes to critical system files, configuration files, or any files deemed sensitive. Prompt action is required to ensure system integrity and security.;Source IP: ;Host IP: "

Timestamp: 2024-11-20T01:36:31.429+07:00

Monitored from: (ELK server) 192.168.174.139

Fig. 30. Email Notification Triggered by Matching Rules

4.3 CNN Model Training Results

The evaluation of the CNN model was performed using a dataset with two classes: 0 and 1. The metrics used to assess the model's performance included precision, recall, F1-score, and accuracy. Experimental results on the data from CSIC2010 dataset are presented. Train and test data is randomly allocated between 70% and 30%

	precision	recall	f1-score	support
0	0.98	1.00	0.99	10759
1	1.00	0.97	0.98	7561
accuracy			0.98	18320
macro avg	0.99	0.98	0.98	18320
weighted avg	0.99	0.98	0.98	18320

Fig. 31. Performance Metrics of the CNN Model

- **Precision:** High precision across both classes, indicating the model's ability to minimize false positives.
- **Recall:** Excellent recall for class 0 (1.00) and slightly lower for class 1 (0.97), showing the model's strength in identifying true positives.

- **F1-Score:** Balanced performance with high F1-scores for both classes (0.99 for class 0 and 0.98 for class 1).
- **Accuracy:** The overall accuracy of 98% reflects the model's reliable classification capability.

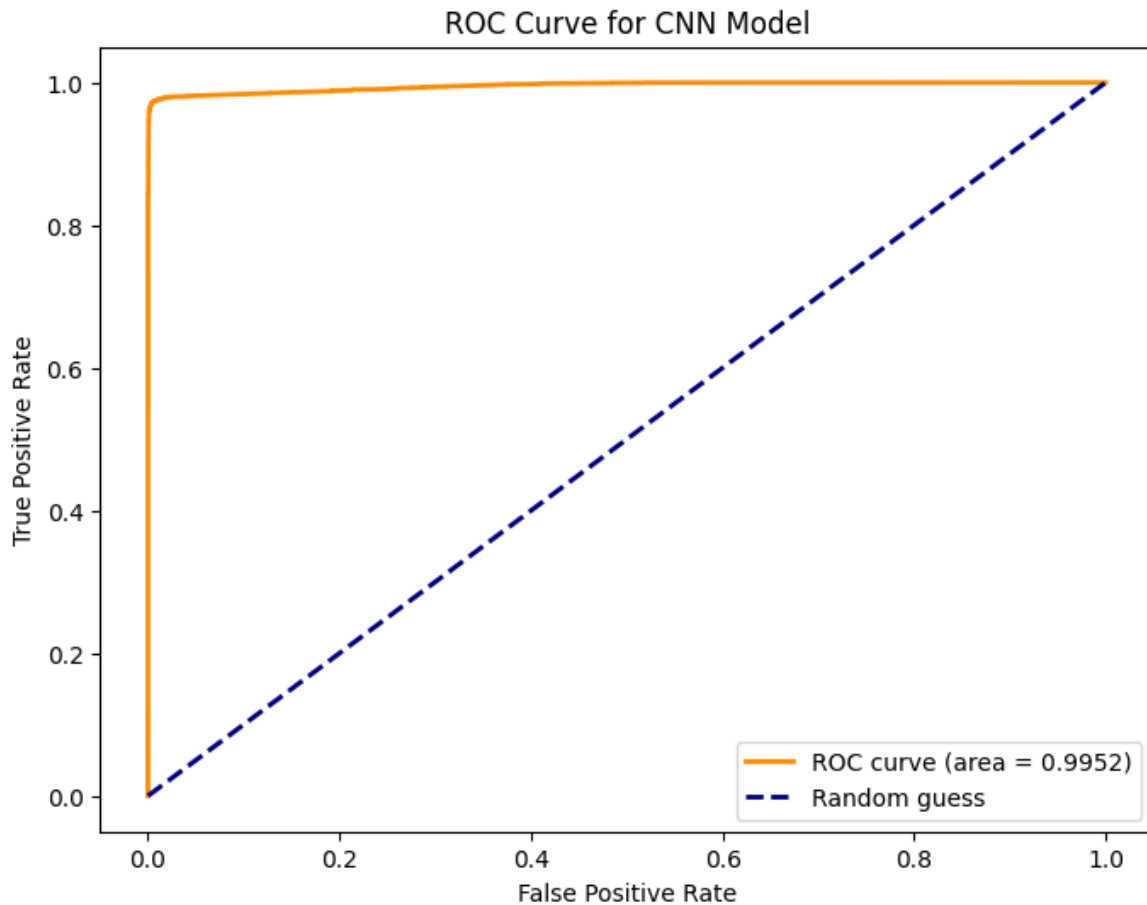


Fig. 32. ROC curve

Performance evaluation and measurement are important issues in machine learning. The AUC-ROC curve serves as a valuable tool for assessing model performance by visualizing the trade-off between true positive rates (TPR) and false positive rates (FPR) at various threshold settings. ROC is a probability curve for different classes. AUC-ROC curve is a performance measurement for classification problems at various thresholds settings. ROC is a probability curve and AUC represents the degree or measure of separability. The ideal value for AUC is 1. In **Fig. 32** for the dataset AUC value is 0.9952. This means that proposed architecture classification is successful for used dataset.

5 Future work and conclusion

5.1 Future work

Although the system has achieved the primary objectives of the project, there is significant potential for further improvement and expansion. The following areas outline the future development directions:

5.1.1 Integration of the CNN Model

- The Convolutional Neural Network (CNN) model has been trained externally to detect abnormal behaviors on websites. In the future, this model will be integrated into the system to complement existing detection rules by monitoring web server logs and identify anomalies that may indicate potential web-based attacks, such as SQL injection, XSS or brute-force attempts.
- This integration will require creating a pipeline between web server logs, Elasticsearch, and the CNN model for seamless real-time analysis. Two potential approaches are proposed for implementing model training for detection:
 1. Centralized Model Training and Deployment:
 - A specific server will be used to train models.
 - After training, the models will be saved and deployed on the ELK Stack server, ensuring that pre-trained models can perform near real-time anomaly detection within the ELK environment.
 2. Distributed Training:
 - Logs from the ELK Stack will be normalized and forwarded to a separate server specifically configured for training and running the model.
 - Based on this analysis, the ELK Stack server can call APIs hosted on the training server to classify the logs as either abnormal or normal.
 - Although this approach brings flexibility, there may still be a slight delay between log collection, model training, and prediction. This delay, however, is expected to be minimal, providing near real-time predictions.
 - This method allows for dynamic model updates and efficient separation of computational tasks between log management and machine learning operations.

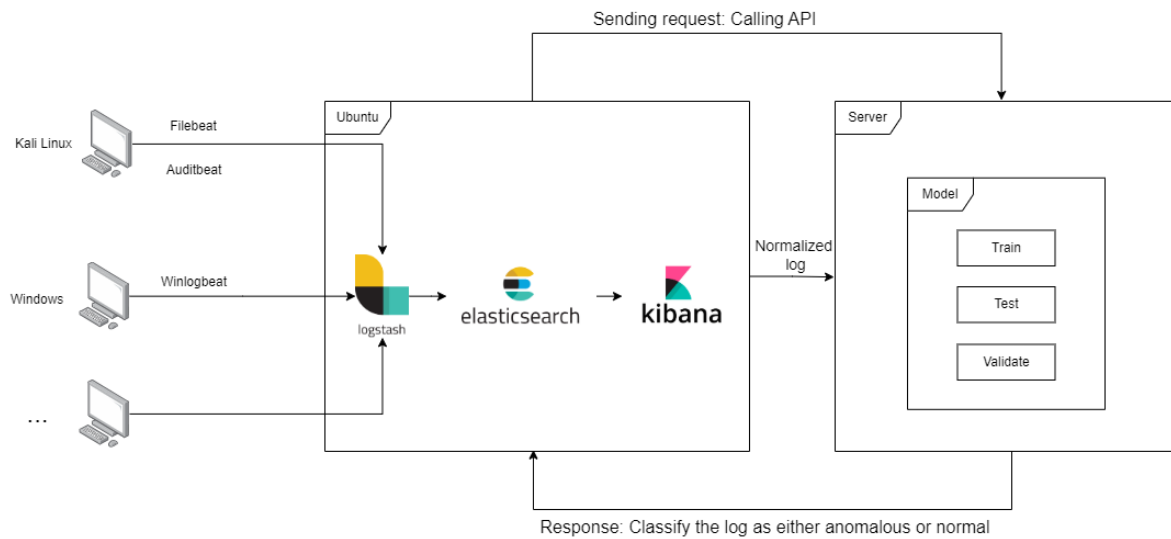


Fig. 33. Proposed Approach for Distributed Training

5.1.2 Expansion of Detection Rules

- Develop additional detection rules for more techniques in the MITRE ATT&CK framework, particularly focusing on phases such as Persistence, Privilege Escalation, and Lateral Movement.
- Incorporate additional data sources such as Active Directory logs or cloud application logs to gain a more comprehensive view of the ecosystem.

5.1.3 Integration with YARA for Malware Detection

- YARA rules will be incorporated to detect malware patterns and suspicious binaries in the collected logs.
- Specific use cases include:
 - Scanning file changes on Linux systems to detect potential malware intrusions.
 - Identifying malware-related indicators in Windows logs, such as malicious executable files or registry modifications.
- This integration will require:
 - Defining custom YARA rules tailored to the organization's environment.
 - Establishing workflows to scan log files and file integrity monitoring outputs against YARA signatures.

5.1.4 System Evaluation and Validation

- Conduct system testing with real-world attack scenarios (redteam) to validate detection effectiveness.
- Compare performance with other commercial SIEM solutions to identify strengths and weaknesses.

5.2 Conclusion

The project has successfully accomplished its goals, including the development and deployment of a complete ELK-based system for log collection, detection, and analysis of suspicious activities leveraging the MITRE ATT&CK framework. The detection rules created have proven effective in identifying common attack techniques on both Linux and Windows operating systems.

The externally trained CNN model provides a strong foundation for future advancements, extending threat detection capabilities beyond static rules. However, integrating the model into the system and optimizing its performance remains a significant challenge to address in the future.

In conclusion, the ELK system serves not only as a powerful log monitoring solution but also as a platform for advancing detection. With further improvements and rigorous testing, the system is poised to better meet the demands of modern cybersecurity environments, effectively countering increasingly sophisticated threats.