

## 1. INTRODUCTION

In the modern digital ecosystem, where organizations rely heavily on IT infrastructure, securing and monitoring network activities is of utmost importance. With the rise in sophisticated cyber threats, enterprises are deploying multiple open-source tools for different aspects of monitoring—network, system, host, and perimeter security. However, these tools often operate in silos, making it difficult to correlate and visualize security incidents holistically. This project aims to overcome these challenges by implementing a centralized log integration solution using the ELK Stack (Elasticsearch, Logstash, Kibana). Logs from Nagios (infrastructure monitoring), Suricata (network intrusion detection), OSSEC (host-based intrusion detection), and pfSense (firewall) are collected using Filebeat, parsed and indexed in ELK, and displayed in real time using Kibana dashboards. This approach provides a unified, centralized view of security and performance data, enabling faster detection, correlation, and response to incidents.

## 1.1 PROBLEM STATEMENT

In modern enterprise environments, maintaining the security and stability of IT infrastructure requires multiple tools to monitor networks, systems, and endpoints. However, these tools—such as Nagios for system monitoring, Suricata for network intrusion detection, OSSEC for host-based security, and pfSense for firewall management—are typically deployed in isolation, creating fragmented monitoring silos. Each tool stores logs in different formats and locations, making it time-consuming and difficult for security teams to correlate events or respond to incidents effectively.

The absence of a centralized logging system leads to several issues:

- Delayed threat detection due to scattered logs across various systems.
- Manual investigation overhead, which increases Mean Time to Detect (MTTD) and Mean Time to Respond (MTTR).
- Limited visibility into overall infrastructure health and threat posture.
- No single pane of glass, meaning analysts must switch between interfaces, reducing operational efficiency.

Moreover, most open-source monitoring tools lack native real-time visualization or alert correlation. Security analysts often face challenges in identifying attack patterns or tracing the root cause of incidents due to lack of centralized context.

Thus, there is a clear need for a centralized log management solution that can:

- Collect logs from multiple tools in real-time,
- Normalize and parse these logs for easy analysis,
- Visualize the data using intuitive dashboards,
- And support alerting mechanisms for rapid incident response.

This project aims to solve these challenges by integrating Nagios, Suricata, OSSEC (HIDS), and pfSense into the ELK Stack, using Filebeat as the lightweight log shipper. This setup

## 2. LITERATURE SURVEY

The concept of centralized log management and security monitoring has evolved significantly over the past decade. Academic research, industry whitepapers, and community-driven case studies have continuously emphasized the need for integration between various monitoring and security tools to enable real-time visibility and faster threat detection.

One of the most widely accepted tools in this domain is the ELK Stack—comprising Elasticsearch, Logstash, and Kibana. It is praised for its flexibility, scalability, and powerful search and visualization capabilities. In a 2021 study on open-source SIEM alternatives, researchers found that ELK, when combined with log shippers like Filebeat, offers near enterprise-level performance for small and medium-sized businesses, with the added advantage of customization.

Centralized log management and security monitoring have become essential in modern cybersecurity operations. Research papers, industry whitepapers, and real-world SOC implementations highlight the need for integrating various monitoring tools to achieve real-time visibility and faster threat detection. The ELK Stack (Elasticsearch, Logstash, Kibana) is widely recognized for its flexibility, scalability, and strong visualization capabilities. When combined with log shippers like Filebeat, ELK offers near enterprise-grade SIEM functionality for small to medium organizations.

Nagios is effective for infrastructure monitoring but lacks log analytics, making ELK integration valuable for trend correlation. Suricata, a modern NIDS, outputs structured eve.json logs ideal for visualization in ELK, with community support favoring its performance over Snort due to multi-threading and native JSON support. OSSEC, a powerful HIDS, generates structured alerts that are easily parsed via Filebeat, offering endpoint visibility and detection of file-level changes or rootkit activity. pfSense, though primarily a firewall, can forward syslogs to ELK using Filebeat or Logstash, providing essential perimeter security insights.

### 3. EXISTING SYSTEM

In traditional IT and security infrastructures, monitoring tools are deployed in isolation, each operating independently with little to no integration. System administrators typically rely on manual log reviews, using built-in logging utilities or default system logs such as `/var/log/syslog` in Linux or Event Viewer in Windows. This process is time-consuming, error-prone, and lacks context, especially when correlating events across different systems or network layers.

While tools like Nagios provide system uptime and service health metrics, they do not capture real-time security events. OSSEC, although efficient in detecting unauthorized file changes and system anomalies, stores logs locally and does not provide a visual interface for real-time analysis. Suricata and pfSense generate logs for intrusion attempts and firewall activity, but without centralization, their outputs are limited to raw log files or basic alerts.

Moreover, the logs generated by each tool are stored in different formats (e.g., plain text, JSON, syslog), making it difficult to analyze or correlate them without a dedicated parsing engine. Analysts are often forced to switch between multiple dashboards and terminals to get a partial view of the infrastructure, increasing response time and reducing efficiency.

In such setups:

- There is no single-pane-of-glass for unified log viewing and analysis.
- Alert fatigue is common, as alerts are not prioritized or correlated.
- Root cause analysis is delayed due to lack of contextual data.
- There is no automated mechanism for real-time log ingestion, analysis, or visual reporting.

Ultimately, these fragmented systems are inadequate for modern SOC (Security Operations Center) needs, where real-time threat detection, behavioral analysis, and automated alerting are essential. Without integration into a centralized log management platform like the ELK Stack, organizations remain vulnerable to advanced persistent threats and slow incident response times.

## 4. ELK ARCHITECTURE

The ELK Stack is a powerful open-source log management and analytics platform that combines three key components: Elasticsearch, Logstash, and Kibana. When paired with Filebeat, the architecture enables centralized log collection, processing, indexing, searching, and real-time visualization.

Here's a breakdown of each component and their role in the architecture:

### 1. Filebeat (Log Shipper)

- Installed on log sources such as Nagios, Suricata, OSSEC agents, and pfSense.
- Collects logs in real-time and forwards them to Logstash or directly to Elasticsearch.
- Lightweight and efficient; ideal for edge devices and endpoints.
- Can handle different log types (e.g., syslogs, JSON, plain text).

### 2. Logstash (Data Processing Pipeline)

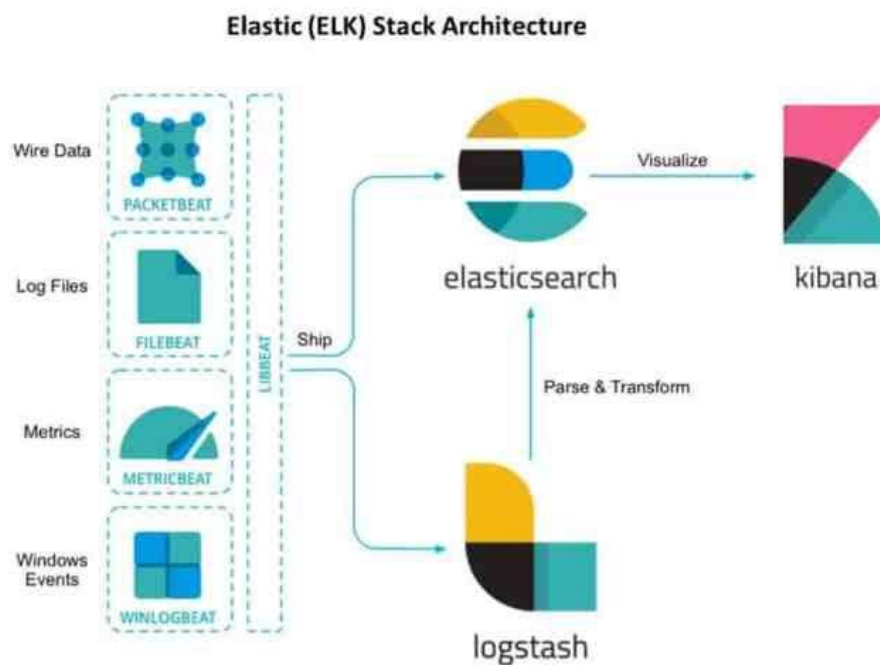
- Acts as the data collector and parser.
- Receives logs from Filebeat, then filters, transforms, and parses data using grok patterns, conditionals, and plugins.
- Useful for structuring logs from Suricata (eve.json), pfSense syslogs, and Nagios or OSSEC alerts before storing in Elasticsearch.
- Can also enrich data with geo-IP or threat intel information.

### 3. Elasticsearch (Search and Storage Engine)

- A distributed NoSQL search engine designed to store, index, and search large volumes of log data quickly.
- Stores structured log documents as JSON and allows for fast, full-text queries.
- Supports horizontal scaling and high availability.
- Acts as the core of the ELK Stack, holding all log data for Kibana to query.

#### 4. Kibana (Visualization Dashboard)

- Web-based frontend for querying and visualizing data stored in Elasticsearch.
- Allows users to create dashboards, visualizations, filters, and alerts based on log data.
- Supports role-based access, making it suitable for SOC analysts, sysadmins, and network engineers.
- Dashboards for Suricata (e.g., attack maps), Nagios status graphs, OSSEC alerts, and pfSense traffic logs provide real-time insights.



## 5. INTEGRATION OF NAGIOS WITH ELK STACK

### Phase 1: Setup Nagios Server (Monitoring Host)

Theory: Nagios is a widely used infrastructure monitoring tool that monitors host availability, service health, and network performance. Before sending logs to ELK, Nagios must be properly installed and running.

```
sudo apt update
sudo apt install nagios3 nagios-plugins-contrib
```

### Phase 2: Locate and Configure Nagios Log File

Theory: Nagios maintains detailed logs about monitoring events such as service alerts, host status changes, and notification history. These logs are necessary for analysis.

```
/var/log/nagios3/nagios.log
sudo nano /etc/nagios3/nagios.cfg
```

```
shuhari@nagi: ~
Nagios Core 4.5.0
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2023-11-14
License: GPL

Website: https://www.nagios.org
Reading configuration data...
  Read main config file okay...
  Read object config files okay...

Running pre-flight check on configuration data...

Checking objects...
  Checked 16 services.
  Checked 2 hosts.
  Checked 1 host groups.
  Checked 0 service groups.
  Checked 1 contacts.
  Checked 1 contact groups.
  Checked 25 commands.
  Checked 5 time periods.
  Checked 0 host escalations.
  Checked 0 service escalations.
Checking for circular paths...
  Checked 2 hosts
  Checked 0 service dependencies
  Checked 0 host dependencies
  Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors: 0

Things look okay - No serious problems were detected during the pre-flight check
shuhari@nagi:~$
```

## Access

UI:

The screenshot shows the Nagios web interface. The top navigation bar includes links for Home, Documentation, and a search bar. The main content area is divided into several sections:

- Current Network Status:** Last Updated: Thu Aug 7 11:42:06 IST 2025. Updated every 90 seconds. Nagios® Core™ 4.5.0 - www.nagios.org. Logged in as nagiosadmin.
- Host Status Totals:** Up: 2, Down: 0, Unreachable: 0, Pending: 0. All Problems: 0, All Types: 2.
- Service Status Totals:** Ok: 10, Warning: 0, Unknown: 0, Critical: 6, Pending: 0. All Problems: 6, All Types: 16.
- Service Status Details For All Hosts:** A table showing details for hosts 'deb-12-1' and 'localhost'.

Host	Service	Status	Last Check	Duration	Attempt	Status Information
deb-12-1	HTTP Service	CRITICAL	08-07-2025 11:31:58	1d 9h 49m 57s	3/3	connect to address 192.168.80.134 and port 80: Connection refused
	Load Average	CRITICAL	08-07-2025 11:32:34	0d 1h 0m 21s	3/3	CRITICAL: Incorrect credentials given.
	Ping Check	OK	08-07-2025 11:36:25	1d 9h 52m 5s	1/3	PING OK - Packet loss = 0%, RTA = 0.61 ms
	Process Count	CRITICAL	08-07-2025 11:37:22	1d 9h 47m 56s	3/3	CRITICAL: Incorrect credentials given.
	SSH Service	OK	08-07-2025 11:38:18	1d 9h 51m 47s	1/3	SSH OK - OpenSSH_9.2p1 Debian-2+deb12u6 (protocol 2.0)
	Swap Usage	CRITICAL	08-07-2025 11:32:54	0d 1h 0m 8s	3/3	CRITICAL: Incorrect credentials given.
localhost	Total Processes	CRITICAL	08-07-2025 11:32:47	1d 9h 49m 34s	3/3	CRITICAL: Incorrect credentials given.
	Virtual Memory Usage	CRITICAL	08-07-2025 11:32:41	0d 1h 0m 35s	3/3	CRITICAL: Incorrect credentials given.
	Current Load	OK	08-07-2025 11:40:01	1d 10h 16m 37s	1/4	OK - load average: 0.05, 0.01, 0.00
	Current Users	OK	08-07-2025 11:40:57	1d 10h 15m 59s	1/4	USERS OK - 3 users currently logged in
	HTTP	OK	08-07-2025 11:41:53	1d 10h 15m 22s	1/4	HTTP OK: HTTP/1.1 200 OK - 10975 bytes in 0.001 second response time
	PING	OK	08-07-2025 11:37:50	1d 10h 19m 44s	1/4	PING OK - Packet loss = 0%, RTA = 0.04 ms
localhost	Root Partition	OK	08-07-2025 11:39:02	1d 10h 19m 7s	1/4	DISK OK - free space: / 12623 MB (86.71% inode=96%):
	SSH	OK	08-07-2025 11:41:07	1d 10h 18m 29s	1/4	SSH OK - OpenSSH_9.2p1 Debian-2+deb12u6 (protocol 2.0)
	Swap Usage	OK	08-07-2025 11:38:14	1d 10h 17m 52s	1/4	SWAP OK - 100% free (3814 MB out of 3814 MB)
	Total Processes	OK	08-07-2025 11:40:31	1d 10h 17m 14s	1/4	PROCS OK: 70 processes with STATE = RSZDT

Open browser → <http://<nagios-ip-address>/nagios3>

### Phase 3: Install & Configure Filebeat on Nagios Server

Theory: Filebeat is a lightweight log shipper that forwards logs from files to Logstash or Elasticsearch. It monitors the specified Nagios log file.

```
sudo apt install filebeat
sudo nano/etc/filebeat/filebeat.yml
```

```
filebeat.inputs:
- type: log
  enabled: true
  paths:
    - /usr/local/nagios/var/nagios.log
    - /usr/local/nagios/var/archive/*.log
  fields:
    log_type: nagios
```



```
# ----- Elasticsearch Output -----
output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["192.168.80.128:9200"]

  # Protocol - either `http` (default) or `https`.
  #protocol: "https"

  # Authentication credentials - either API key or username/password
  #api_key: "id:api_key"
  username: "elastic"
  password: "elastic"
```

sudo

systemctl enable filebeat  
sudo systemctl start filebeat

sudo nano /etc/logstash/conf.d/nagios.conf

```
GNU nano 7.2 nagios.conf
filter {
  if [log_type] == "nagios" {
    grok {
      match => { "message" => "%{TIMESTAMP_ISO8601:timestamp} %{GREEDYDATA:details}" }
    }

    date {
      match => ["timestamp", "ISO8601"]
      target => "@timestamp"
    }

    mutate {
      remove_field => ["timestamp"]
    }
  }
}

output {
  if [log_type] == "nagios" {
    elasticsearch {
      hosts => ["http://localhost:9200"]
      index => "nagios-logs-%{+YYYY.MM.dd}"
    }

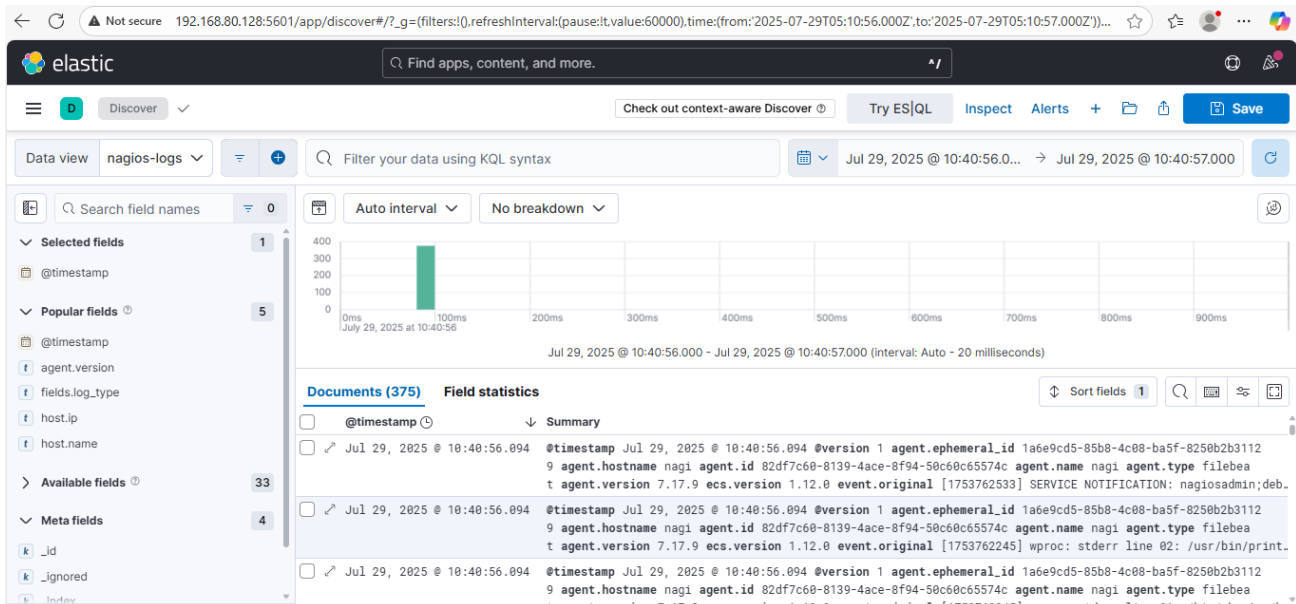
    stdout {
      codec => rubydebug
    }
  }
}
```

sudo systemctl restart logstash

## Phase 6: Verify Logs in Kibana

Theory: Kibana is used to query and visualize logs stored in Elasticsearch. Once logs are indexed, you can analyze service alerts, downtime, etc.

1. Open Kibana in browser:  
`http://<elk-ip>:5601`
2. Go to Discover.
3. Create index pattern:  
`nagios-logs-*`  
or `filebeat-*` if sent directly via Filebeat.
4. View and filter logs based on:
  - Severity (e.g., WARNING, CRITICAL)
  - Host or service
  - Time



## 6. INTEGRATION OF SURICATA WITH ELK

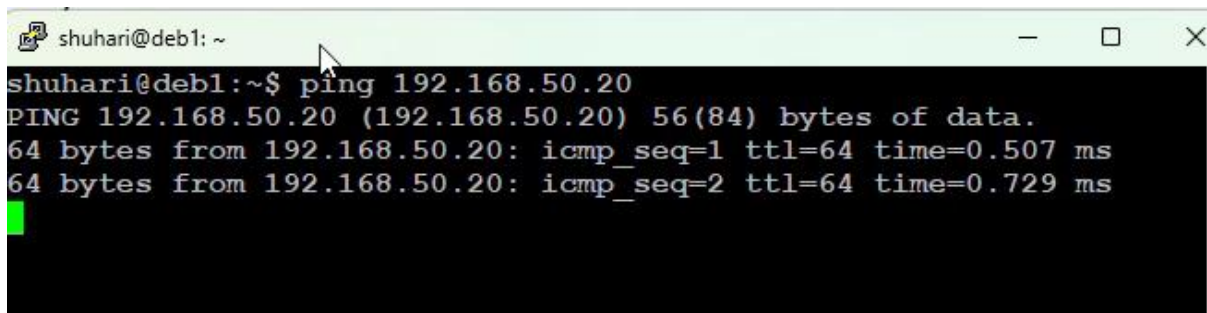
### Phase 1: Install and Configure Suricata

Theory: Suricata is an open-source network IDS/IPS that inspects traffic and generates logs (alerts, flows, anomalies). These logs, especially eve.json, are useful for ELK ingestion.

```
sudo apt update
```

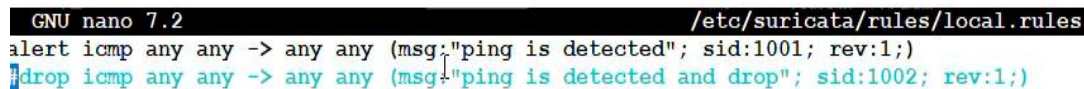
```
sudo apt install suricata -y
```

```
sudo nano /etc/suricata/suricata.yaml
```

A terminal window titled 'shuhari@deb1: ~' with standard window controls. The terminal shows the command 'ping 192.168.50.20' being executed. The output shows two successful ping responses from 192.168.50.20 with TTL=64 and times of 0.507 ms and 0.729 ms respectively. A green cursor is visible on the line following the second ping response.

```
shuhari@deb1:~$ ping 192.168.50.20
PING 192.168.50.20 (192.168.50.20) 56(84) bytes of data.
64 bytes from 192.168.50.20: icmp_seq=1 ttl=64 time=0.507 ms
64 bytes from 192.168.50.20: icmp_seq=2 ttl=64 time=0.729 ms
```

Phase 2:

A screenshot of the GNU nano 7.2 text editor editing the file '/etc/suricata/rules/local.rules'. The editor shows two rules: an alert rule for ICMP ping and a drop rule for ICMP ping. The drop rule is highlighted with a blue selection bar.

```
GNU nano 7.2 /etc/suricata/rules/local.rules
alert icmp any any -> any any (msg:"ping is detected"; sid:1001; rev:1;)
drop icmp any any -> any any (msg:"ping is detected and drop"; sid:1002; rev:1;)
```

### Install Filebeat on Suricata System

Theory: Filebeat monitors the eve.json file generated by Suricata and forwards logs to Logstash or Elasticsearch.

```
sudo nano /etc/filebeat/filebeat.yml
```

```
filebeat.inputs:
```

```
- type: log
  enabled: true
  paths:
    - /var/log/suricata/eve.json
  json.keys_under_root: true
```

```
json.add_error_key: true
```

```
output directly to Elasticsearch:
```

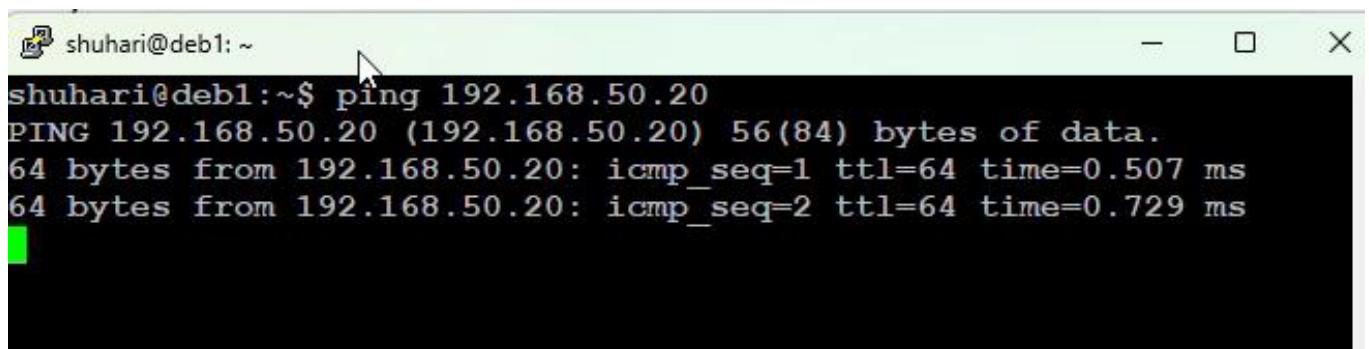
```
output.elasticsearch:
  hosts: ["http://<elk-ip>:9200"]
```

#### Phase 4: Start Filebeat

Theory: Starts the log forwarding process from eve.json to the ELK pipeline.

```
sudo systemctl enable filebeat
```

```
sudo systemctl start filebeat
```

A terminal window titled 'shuhari@deb1: ~' with standard window controls. The terminal shows a user executing a ping command to 192.168.50.20. The output shows two successful ping responses with 64 bytes of data, TTL of 64, and response times of 0.507 ms and 0.729 ms respectively. A green cursor is visible on the line following the second ping response.

```
shuhari@deb1:~$ ping 192.168.50.20
PING 192.168.50.20 (192.168.50.20) 56(84) bytes of data.
64 bytes from 192.168.50.20: icmp_seq=1 ttl=64 time=0.507 ms
64 bytes from 192.168.50.20: icmp_seq=2 ttl=64 time=0.729 ms
```

```

shuhari@suricata: ~
GNU nano 7.2 /var/log/suricata/fast.log
08/03/2025-00:01:20.843149 [wDrop] [**] [1:1002:1] ping is detected and drop [**] [Classification: (null)] [Priority: 3] {IPv6-ICMP}
08/03/2025-00:03:43.599992 [wDrop] [**] [1:1002:1] ping is detected and drop [**] [Classification: (null)] [Priority: 3] {IPv6-ICMP}
08/03/2025-00:10:10.662877 [wDrop] [**] [1:1002:1] ping is detected and drop [**] [Classification: (null)] [Priority: 3] {IPv6-ICMP}
08/03/2025-00:13:01.599323 [wDrop] [**] [1:1002:1] ping is detected and drop [**] [Classification: (null)] [Priority: 3] {ICMP} 192.
08/03/2025-00:15:31.608685 [wDrop] [**] [1:1002:1] ping is detected and drop [**] [Classification: (null)] [Priority: 3] {IPv6-ICMP}
08/03/2025-00:16:20.905456 [wDrop] [**] [1:1002:1] ping is detected and drop [**] [Classification: (null)] [Priority: 3] {IPv6-ICMP}
08/03/2025-00:18:43.722332 [wDrop] [**] [1:1002:1] ping is detected and drop [**] [Classification: (null)] [Priority: 3] {IPv6-ICMP}
08/03/2025-00:25:58.684578 [wDrop] [**] [1:1002:1] ping is detected and drop [**] [Classification: (null)] [Priority: 3] {IPv6-ICMP}
08/03/2025-00:31:20.944269 [wDrop] [**] [1:1002:1] ping is detected and drop [**] [Classification: (null)] [Priority: 3] {IPv6-ICMP}
08/03/2025-00:33:43.794507 [wDrop] [**] [1:1002:1] ping is detected and drop [**] [Classification: (null)] [Priority: 3] {IPv6-ICMP}
08/03/2025-00:41:07.703832 [wDrop] [**] [1:1002:1] ping is detected and drop [**] [Classification: (null)] [Priority: 3] {IPv6-ICMP}
08/03/2025-00:42:05.850505 [wDrop] [**] [1:1002:1] ping is detected and drop [**] [Classification: (null)] [Priority: 3] {IPv6-ICMP}
08/03/2025-00:46:21.000345 [wDrop] [**] [1:1002:1] ping is detected and drop [**] [Classification: (null)] [Priority: 3] {IPv6-ICMP}
08/03/2025-00:47:33.653881 [**] [1:1001:1] ping is detected [**] [Classification: (null)] [Priority: 3] {ICMP} 192.168.50.50:8 -> 19
08/03/2025-00:47:33.653921 [**] [1:1001:1] ping is detected [**] [Classification: (null)] [Priority: 3] {ICMP} 192.168.50.20:0 -> 19
08/03/2025-00:48:43.903575 [**] [1:1001:1] ping is detected [**] [Classification: (null)] [Priority: 3] {IPv6-ICMP} fe80:0000:0000:0
08/03/2025-00:48:58.916487 [wDrop] [**] [1:1002:1] ping is detected and drop [**] [Classification: (null)] [Priority: 3] {ICMP} 192.
08/03/2025-00:50:12.376692 [wDrop] [**] [1:1002:1] ping is detected and drop [**] [Classification: (null)] [Priority: 3] {IPv6-ICMP}
08/03/2025-00:51:16.644437 [wDrop] [**] [1:1002:1] ping is detected and drop [**] [Classification: (null)] [Priority: 3] {ICMP} 192.
08/03/2025-00:52:24.205534 [**] [1:1001:1] ping is detected [**] [Classification: (null)] [Priority: 3] {ICMP} 192.168.50.50:8 -> 19
08/03/2025-00:52:24.205571 [**] [1:1001:1] ping is detected [**] [Classification: (null)] [Priority: 3] {ICMP} 192.168.50.20:0 -> 19
08/03/2025-01:00:13.828154 [wDrop] [**] [1:1002:1] ping is detected and drop [**] [Classification: (null)] [Priority: 3] {ICMP} 192.

```

## phase 5: (Optional) Configure Logstash Pipeline for Suricata

Theory: If you are using Logstash, configure parsing rules for Suricata's JSON logs to extract meaningful fields like source IP, alert signature, protocol, etc.

```
sudo nano /etc/logstash/conf.d/suricata.conf
```

```

input {
  beats {
    port => 5044
  }
}

filter {
  if [event_type] == "alert" {

```

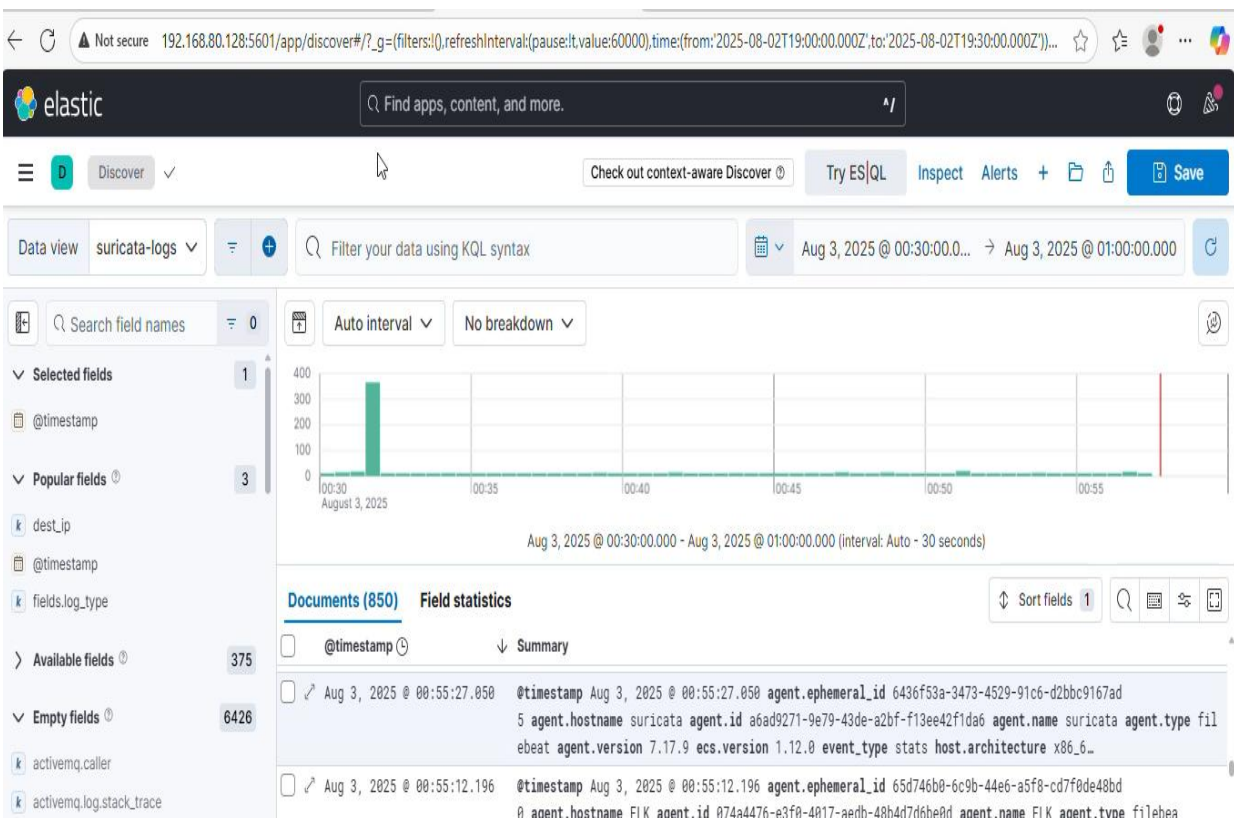
```

mutate {
  add_tag => ["suricata-alert"]
}
}
date {
  match => ["timestamp", "ISO8601"]
}
}

output {
  elasticsearch {
    hosts => ["http://localhost:9200"]
    index => "suricata-logs-%{+YYYY.MM.dd}"
  }
}

```

sudo systemctl restart logstash



## 7. INTEGRATION OF OSSEC WITH ELK

### Objective

Forward OSSEC alerts directly to Logstash, where they are parsed and indexed into Elasticsearch for visualization in Kibana.

### Step 1: Install and Configure OSSEC

#### Theory:

OSSEC generates alerts in `/var/ossec/logs/alerts/alerts.json` (JSON format). Logstash will directly read this file and send structured data to Elasticsearch.

- wget <https://github.com/ossec/ossec-hids/archive/master.tar.gz>
  - tar -xvzf master.tar.gz
  - cd ossec-hids-master
  - sudo ./install.sh
- 
- Select server mode.
  - Enable JSON output during installation or in the config file.



```
OSSEC HIDS v3.7.0 Installation Script - http://www.ossec.net

You are about to start the installation process of the OSSEC HIDS.
You must have a C compiler pre-installed in your system.

- System: Linux nagi 6.1.0-15-amd64
- User: root
- Host: nagi

-- Press ENTER to continue or Ctrl-C to abort. --

- You already have OSSEC installed. Do you want to update it? (y/n): n

1- What kind of installation do you want (server, agent, local, hybrid or help)? server

- Server installation chosen.

2- Setting up the installation environment.

- Choose where to install the OSSEC HIDS [/var/ossec]:

- Installation will be made at /var/ossec .

- The installation directory already exists. Should I delete it? (y/n) [y]: y
```

### During Setup:

- Choose: server
- Enable email alerts? → Optional
- Enable integrity checking? → Yes
- Enable rootkit detection? → Yes
- Enable active response? → Yes



```
3.2- Do you want to run the integrity check daemon? (y/n) [y]: y
- Running syscheck (integrity check daemon).

3.3- Do you want to run the rootkit detection engine? (y/n) [y]: y
- Running rootcheck (rootkit detection).

3.4- Active response allows you to execute a specific
      command based on the events received. For example,
      you can block an IP address or disable access for
      a specific user.
      More information at:
      http://www.ossec.net/docs/docs/manual/ar/index.html

- Do you want to enable active response? (y/n) [y]: y
- Active response enabled.

- By default, we can enable the host-deny and the
  firewall-drop responses. The first one will add
  a host to the /etc/hosts.deny and the second one
  will block the host on iptables (if linux) or on
  ipfilter (if Solaris, FreeBSD or NetBSD).
- They can be used to stop SSHD brute force scans,
  portscans and some other forms of attacks. You can
  also add them to block on snort events, for example.

- Do you want to enable the firewall-drop response? (y/n) [y]: y
- firewall-drop enabled (local) for levels >= 6
-
- 192.168.80.2

- Do you want to add more IPs to the white list? (y/n)? [n]: 192.168.50.30
```

---

## Step 5: Start OSSEC

- sudo /var/ossec/bin/ossec-control start
- sudo nano /var/ossec/etc/ossec.conf
- echo \$?

## **PART 2: INSTALL OSSEC AGENT (Client)**

**The OSSEC Agent monitors its own system and sends logs to the Manager.**

### **Step 1: Install Dependencies**

- sudo apt update
- sudo apt install gcc make build-essential inotify-tools libevent-dev zlib1g-dev  
libpcre2-dev libssl-dev curl -y
- sudo apt-get update
- sudo apt-get install libsystemd-dev.
  
- sudo ./install.sh

### **During Setup:**

- Select language (e.g., en)
- Choose: agent
- Enter the Manager IP when prompted

### PART 3: LINK AGENT TO MANAGER

### Step 1: Add Agent on Manager

```
- sudo /var/ossec/bin/manage_agents
```

- sudo

```
*****
* Wazuh v4.7.5 Agent manager.                *
* The following options are available:         *
*****
(A)dd an agent (A).
(E)xtract key for an agent (E).
(L)ist already added agents (L).
(R)emove an agent (R).
(Q)uit.
Choose your action: A,E,L,R or Q: A

- Adding a new agent (use '\q' to return to the main menu).
Please provide the following:
* A name for the new agent: wazuh-agent-01
* The IP Address of the new agent: 172.104.52.103
Confirm adding it?(y/n): y
```

- sudo /var/ossec/bin/manage\_agents

```
root@localhost:~# cd /var/ossec/bin
root@localhost:/var/ossec/bin# ls
agent-auth      wazuh-agentd    wazuh-execd      wazuh-modulesd
manage_agents   wazuh-control   wazuh-logcollector wazuh-syscheckd
root@localhost:/var/ossec/bin# ./manage_agents

*****
* Wazuh v4.7.5 Agent manager.          *
* The following options are available: *
*****
  (I)mport key from the server (I).
  (Q)uit.
Choose your action: I or Q: I

* Provide the Key generated by the server.
* The best approach is to cut and paste it.
*** OBS: Do not include spaces or new lines.

Paste it here (or '\q' to quit): paste_the_random_key_here
```

### Step 3: Start & Restart Agent

- sudo /var/ossec/bin/ossec-control restart

### Step 3: Configure Logstash to Read OSSEC Alerts

Create config file:

```
sudo nano /etc/logstash/conf.d/ossec.conf
```

```

GNU nano 7.2                                ossec.conf
filter {
  if [log_type] == "ossec" {
    grok {
      match => {
        "message" => "\.* Alert %{NUMBER:alert_id}\.([^\n]*)\n%{TIMESTAMP_ISO8601:timestamp} %{GREEDYDATA:rest}"
      }
      tag_on_failure => ["grok_fail"]
    }

    mutate {
      add_field => { "event_source" => "ossec" }
    }

    date {
      match => ["timestamp", "ISO8601"]
      target => "@timestamp"
    }
  }
}

output {
  if [log_type] == "ossec" {
    elasticsearch {
      hosts => ["http://localhost:9200"]
      index => "ossec-logs-%{+YYYY.MM.dd}"
    }

    stdout { codec => rubydebug }
  }
}

```

- sudo systemctl restart logstash

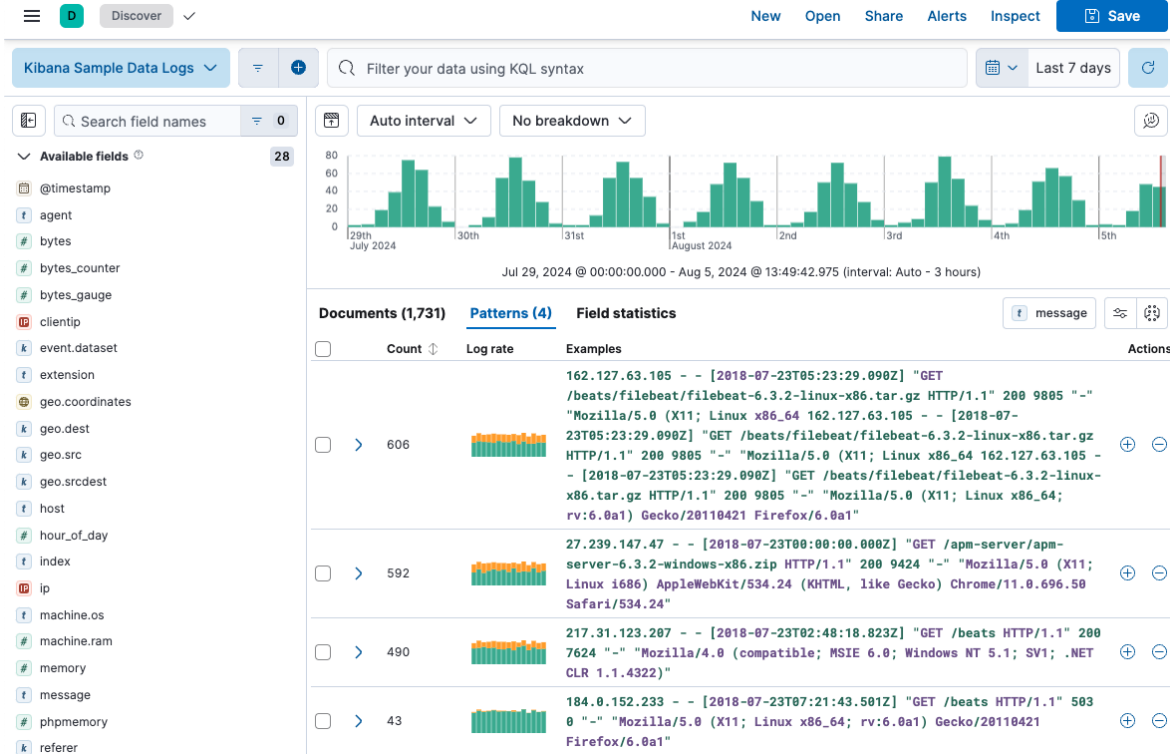
#### Step 4: Verify OSSEC Data in Elasticsearch

- curl -X GET "localhost:9200/\_cat/indices?v"

```

fkane@ubuntu: /usr/share/logstash
.deleted store.size pri.store.size
yellow open  logstash-2017.05.04 rxFwOEuOQQaxGQ6R47HBTQ  5  1  16762
0      10.2mb      10.2mb
yellow open  ratings NO5NFmWaTsCGbMZG0paniQ  5  1  100004
0      15.7mb      15.7mb
yellow open  logstash-2017.05.01 GaoZDmeIQ0iN4CrrOmUSIQ  5  1  15948
0      9.9mb      9.9mb
yellow open  shakespear _cNiumrQTtCtIxK0oWLoFw  5  1  111396
0      28.4mb      28.4mb
yellow open  logstash-2017.04.30 RWzDIkORTC6lm86xjoZOkG  5  1  14166
0      8.9mb      8.9mb
yellow open  series EiS6q9UDRdqhobxemhluCA  5  1  8
0      8.9kb      8.9kb
yellow open  tags 5nHim49KSQk7vazWX7sUA  5  1  1296
0      389.3kb      389.3kb
yellow open  movies XQOEKAOxQzCfxN8e9J8j1g  5  1  9125
0      1.7mb      1.7mb
yellow open  logstash-2017.05.02 XxBVuE_pSwCG_Z3EuPagQQ  5  1  16278
0      10mb      10mb
yellow open  logstash-2017.05.03 rPpnhp9sT_SL-0u7l_nqVg  5  1  21172
0      13.2mb      13.2mb
yellow open  logstash-2017.05.05 wZysoAIIRHCQ7Ofioe8pbG  5  1  18646
0      11.1mb      11.1mb
fkane@ubuntu: /usr/share/logstash$ curl -XG

```



8.

## 8. INTEGRATION OF PFSense WITH ELK

pfSense is a FreeBSD-based firewall and router platform that can generate detailed logs for firewall events, system activity, VPN usage, DHCP leases, and more.

Integrating pfSense logs into the ELK Stack (Elasticsearch, Logstash, Kibana) enables:

- Centralized log collection
- Real-time search & analysis
- Security and traffic visualization dashboards

### step 1: Enable Remote Syslog in pfSense

1. pfSense Web UI → *Status* → *System Logs* → *Settings*
2. Check Enable Remote Logging
3. Enter ELK server IP & Port (e.g., 514 for UDP syslog)
4. Select the log types (Firewall, DHCP, System, etc.)
5. Save.

been made.

### Remote Logging Options

**Enable Remote Logging** ☒ Send log messages to remote syslog server

**Source Address**

This option will allow the logging daemon to bind to a single IP address, rather than all IP addresses. If a single IP is picked, remote syslog servers must all be of that IP type. To mix IPv4 and IPv6 remote syslog servers, bind to all interfaces.

NOTE: If an IP address cannot be located on the chosen interface, the daemon will bind to all addresses.

**IP Protocol**

This option is only used when a non-default address is chosen as the source above. This option only expresses a preference; If an IP address of the selected type is not found on the chosen interface, the other type will be tried.

**Remote log servers**

**Remote Syslog Contents**

☒ Everything

☐ System Events

☐ Firewall Events

☐ DNS Events (Resolver/unbound, Forwarder/dnsmasq, filterdns)

☐ DHCP Events (DHCP Daemon, DHCP Relay, DHCP Client)

☐ PPP Events (PPPoE WAN Client, L2TP WAN Client, PPTP WAN Client)

☐ Captive Portal Events

☐ VPN Events (IPsec, OpenVPN, L2TP, PPPoE Server)

☐ Gateway Monitor Events

☐ Routing Daemon Events (RADVD, UPnP, RIP, OSPF, BGP)

☐ Server Load Balancer Events (relayd)

☐ Network Time Protocol Events (NTP Daemon, NTP Client)

☐ Wireless Events (hostapd)

Syslog sends UDP datagrams to port 514 on the specified remote syslog server, unless another port is specified. Be sure to set syslogd on the remote server to accept syslog messages from pfSense.

**Step 2:**

### Configure Logstash to Receive pfSense Logs

On the ELK server, create a Logstash configuration file:

- `sudo nano /etc/logstash/conf.d/pfsense.conf`

```
input {
  beats {
    port => 5044
  }
}
filter {
  grok {
    patterns_dir => ["/etc/logstash/patter"]
    match => {"message"=> "%{IPORHOST:clientip} %{NGUSER:ident
HTTP/%{NUMBER:httpversion}\\\" %NUMBER:response}\""}
  }
}
output {
  syslog {
    hosts => "192.168.2.250"
    port => 514
    protocol => "tcp"
  }
}
```

- `sudo systemctl restart logstash`

- `sudo systemctl enable logstash`



### Step 3: Verify Data in Elasticsearch

Shell Output - pkg search speedtest

py38-speedtest-cli-2.1.3      Command line interface for testing internet bandwidth

Execute Shell Command

pkg search speedtest

<< Execute >> Clear

-  
Open

logstash type:

```

shuhari@ELK: /etc/logstash/conf.d
shuhari@ELK:/etc/logstash/conf.d$
Message from syslogd@192.168.80.131 at Aug  7 15:28:53 ...
  php-fpm[345]: /index.php: Successful login for user 'admin' from: 192.168.50.40 (Local Database)
^C
shuhari@ELK:/etc/logstash/conf.d$ sudo tcpdump -i any port 514
tcpdump: data link type LINUX_SLL2
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
15:35:21.342759 ens33 In  IP 192.168.80.131.syslog > ELK.syslog: SYSLOG local5.info, length: 291
15:35:24.852276 ens33 In  IP 192.168.80.131.syslog > ELK.syslog: SYSLOG user.notice, length: 57
15:35:24.872275 ens33 In  IP 192.168.80.131.syslog > ELK.syslog: SYSLOG local5.info, length: 284
15:35:25.686543 ens33 In  IP 192.168.80.131.syslog > ELK.syslog: SYSLOG user.notice, length: 57
15:35:25.689773 ens33 In  IP 192.168.80.131.syslog > ELK.syslog: SYSLOG local5.info, length: 284
15:35:26.312788 ens33 In  IP 192.168.80.131.syslog > ELK.syslog: SYSLOG local5.info, length: 281
15:35:26.325713 ens33 In  IP 192.168.80.131.syslog > ELK.syslog: SYSLOG user.notice, length: 57
15:35:26.661711 ens33 In  IP 192.168.80.131.syslog > ELK.syslog: SYSLOG user.notice, length: 57
15:35:26.664540 ens33 In  IP 192.168.80.131.syslog > ELK.syslog: SYSLOG local5.info, length: 284
^C
9 packets captured
10 packets received by filter
0 packets dropped by kernel

```

## 9. ADVANTAGES

Integrating Nagios (infrastructure monitoring), OSSEC (host-based intrusion detection), and Suricata (network intrusion detection) into the ELK Stack provides a unified, real-time view of system health, security events, and network traffic.

### 1. Unified Visibility

Combines metrics, host security logs, and network alerts into a single dashboard.

Reduces the need to log into multiple tools separately.

### 2. Faster Incident Response

Correlates alerts from different sources (e.g., OSSEC detecting a file integrity breach and Suricata flagging suspicious traffic from the same host).

Enables quick identification of the root cause.

### 3. Historical Analysis

Stores all logs in Elasticsearch for long-term retention.

Allows searching historical events to investigate past incidents or compliance audits.

### 4. Real-time Monitoring & Alerts

ELK provides near real-time ingestion and visualization.

Alerts can be generated in Kibana or sent from Nagios/OSSEC/Suricata triggers.

### 5. Cross-correlation of Data

Detects multi-vector attacks by correlating:

Nagios service downtime alerts

OSSEC file integrity changes

Suricata malicious traffic alerts

Helps in detecting stealthy or complex threats.

## **6. Centralized Security Operations**

A single point of management for logs and alerts.

Easier onboarding for SOC (Security Operations Center) analysts.

## **7. Reduced Log Analysis Time**

Powerful Elasticsearch queries filter logs by IP, host, timestamp, or event type instantly.

No need to manually search log files on each device.

## **8. Compliance & Audit Readiness**

Supports regulatory frameworks (ISO 27001, PCI-DSS, HIPAA) by maintaining a secure, searchable log archive.

Easy to generate audit reports from Kibana dashboards

.

## **9. Scalability**

ELK can ingest data from multiple Nagios, OSSEC, and Suricata instances across different sites.

Allows centralized management for distributed environments.

## **10. Enhanced Threat Detection**

OSSEC provides HIDS alerts (log anomalies, rootkit detection, integrity checks).

Suricata provides NIDS alerts (suspicious network signatures, anomalies).

Nagios adds availability metrics (service uptime, response time).

Combined, they give full-stack observability: infrastructure, host, and network.

## 10. CONCLUSION

The integration of Nagios, OSSEC, and Suricata with the ELK Stack successfully creates a centralized monitoring and security analytics platform capable of delivering real-time insights, historical analysis, and comprehensive visibility across infrastructure, hosts, and network traffic.

By combining Nagios for infrastructure and service availability monitoring, OSSEC for host-based intrusion detection, and Suricata for network intrusion detection, the system ensures full-spectrum observability. With ELK as the central platform, all logs and alerts are collected, parsed, indexed, and visualized in one place, allowing for faster incident response, easier troubleshooting, and enhanced threat detection capabilities.

This project demonstrates that centralized log monitoring not only improves operational efficiency but also strengthens security posture by enabling correlation between multiple sources of data. The resulting dashboards provide a single pane of glass for administrators and security teams, supporting both real-time monitoring and forensic investigations.

Overall, the solution delivers a scalable, flexible, and effective monitoring ecosystem that can be extended to other security tools and adapted for evolving enterprise needs. It is a step toward building a Security Information and Event Management (SIEM) capability using open-source technologies.

## 11. REFERENCES

1. Elastic, “The Elastic Stack — Elasticsearch, Logstash, and Kibana,” *Elastic Documentation*, [Online]. Available: <https://www.elastic.co/what-is/elk-stack>. [Accessed: Aug. 10, 2025].
2. OSSEC, “OSSEC HIDS Documentation,” *OSSEC Project*, [Online]. Available: <https://www.ossec.net/docs/>. [Accessed: Aug. 10, 2025].
3. Suricata, “Suricata User Guide,” *Open Information Security Foundation (OISF)*, [Online]. Available: <https://suricata.io/documentation/>. [Accessed: Aug. 10, 2025].
4. Nagios, “Nagios Core Documentation,” *Nagios Enterprises*, [Online]. Available: <https://www.nagios.org/documentation/>. [Accessed: Aug. 10, 2025].
5. V. K. Singh and P. Sharma, “Centralized Log Monitoring using ELK Stack: An Open Source Approach for Security and Performance,” *International Journal of Computer Applications*, vol. 182, no. 4, pp. 10–15, Aug. 2018.
6. R. Ahmad, S. Hussain, and M. Faheem, “Integration of IDS Logs into ELK for Network Security Monitoring,” *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 11, no. 5, pp. 422–428, 2020.
7. pfSense, “Remote Logging Options in pfSense,” *Netgate Documentation*, [Online]. Available: <https://docs.netgate.com/pfsense/en/latest/monitoring/remote-syslog.html>. [Accessed: Aug. 10, 2025].
8. T. Smith, “How to Integrate Suricata with ELK for Real-Time Threat Detection,” *Security Onion Blog*, Mar. 2023. [Online]. Available: <https://blog.securityonion.net/>. [Accessed: Aug. 10, 2025].

