SOC Fundamentals and Practical Implementation Report

Date: August 14, 2025
Author: Afshan Shaikh

Theory Implementation

1. SOC Fundamentals and Operations

Learned:
A Security Operations Center (SOC) is a centralized unit responsible for proactive threat detection, continuous monitoring, and incident response. The SOC operates continuously and relies on skilled personnel, defined processes, and advanced tools to secure an organization's environment. SOC analyst roles are categorized into three tiers:

- **Tier 1:** Initial alert monitoring, triage, and escalation.
- **Tier 2:** In-depth analysis, investigation of escalated alerts, and identification of root causes.
- **Tier 3:** Advanced incident handling, threat hunting, and optimization of detection capabilities.

Core SOC functions include log analysis, alert triage, integration of threat intelligence, and orchestration of incident workflows.

Researched:
Studied the **NIST Cybersecurity Framework** and **MITRE ATT&CK** matrix to understand industry-standard SOC processes. Reviewed real-world SOC walkthrough videos from IBM and Microsoft to observe operational setups. Explored automation capabilities within Splunk Phantom to understand SOAR playbook execution.

2. Security Monitoring Basics

Learned:
Security monitoring focuses on detecting anomalies, unauthorized access attempts, and policy violations in real time. The primary tools include **SIEM platforms** (Elastic SIEM, Splunk) for event correlation and **network analyzers** (Wireshark) for packet-level analysis. Key performance metrics include:

- **False Positive / False Negative Rates** — accuracy of detection rules.
- **Mean Time to Detect (MTTD)** — speed of threat identification.

3. Log Management Fundamentals

Learned:

The log lifecycle consists of:

- **Collection** — Gathering logs from servers, endpoints, and network devices.
- **Normalization** — Converting logs into a common structure (e.g., JSON, CEF).
- **Storage** — Secure retention in a centralized repository.
- **Retention** — Maintaining logs for compliance and investigation needs.
- **Analysis** — Querying and correlating data for threat detection.

Researched:

Implemented a log collection pipeline using **Fluentd** on Ubuntu to collect Syslog entries. Tested log forwarding using the logger "Test message" command and verified ingestion in Elastic SIEM.

4. Security Tools Overview

Learned:

- **SIEM** — Splunk, QRadar, Elastic for correlation and alerting.
- **EDR** — CrowdStrike for endpoint-level detection and remediation.
- **IDS/IPS** — Snort for identifying and blocking malicious traffic.
- **Vulnerability Scanners** — Nessus for automated vulnerability assessments.

Researched:

Created a Snort detection rule for malicious HTTP requests to malicious.com and validated detection using a curl request. Performed a vulnerability assessment with Nessus Essentials on a Metasploitable2 VM, identifying the top three vulnerabilities by CVSS score. Used Osquery to query running processes on a Windows VM and simulated a suspicious process for detection testing.

5. Basic Security Concepts

Learned:

- **CIA Triad:** Confidentiality, Integrity, Availability.
- **Threat:** Potential cause of an unwanted incident.
- **Vulnerability:** A weakness that could be exploited.
- **Risk:** The impact of a threat exploiting a vulnerability.
- **Defense-in-Depth:** Layered security controls for redundancy.
- **Zero Trust:** “Never trust, always verify” access model.

6. Incident Response Basics

Learned:

Incident response is guided by the **NIST SP 800-61** framework, which consists of six stages:

- **Preparation** — Establishing policies, tools, and training for incident handling.
- **Identification** — Detecting and verifying that an incident has occurred.
- **Containment** — Isolating affected systems to prevent further impact.
- **Eradication** — Removing the threat from the environment.
- **Recovery** — Restoring systems to normal operation and verifying functionality.
- **Lessons Learned** — Reviewing the incident to improve future response capabilities.

7. Documentation Standards

Learned:

Effective SOC documentation ensures consistency, accountability, and knowledge retention. Standard documentation types include:

- **Incident Reports** — Detailed records of security incidents, including timelines and impact assessments.
- **Standard Operating Procedures (SOPs)** — Step-by-step instructions for recurring security tasks.
- **Technical Runbooks** — Detailed guides for handling specific incidents or alerts.
- **Post-Incident Reviews** — Analytical reports capturing lessons learned and recommendations.

Researched:

Reviewed the **SANS Incident Handler's Handbook** to understand best practices in SOC documentation. Studied examples of incident reports and post-incident reviews to identify common structures and essential data points.

Practical Implementation

1. Log Collection Pipeline Implementation

Objective: Establish a complete syslog to ELK (Elasticsearch, Logstash, Kibana) pipeline for centralized security log management.

Steps:

1.1. Prerequisite Installation:

```
sudo apt update
sudo apt install -y openjdk-11-jre
curl -fsSL https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
echo "deb https://artifacts.elastic.co/packages/8.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-8.x.list
sudo apt update
sudo apt install -y elasticsearch kibana logstash
sudo systemctl enable --now elasticsearch
sudo systemctl enable --now kibana
```

1.2. Logstash Configuration:

Created /etc/logstash/conf.d/syslog.conf with:

```
input {
  udp { port => 5140 type => "syslog" }
  tcp { port => 5141 type => "syslog_tcp" }
}
filter {
  if [type] == "syslog" or [type] == "syslog_tcp" {
    grok { match => { "message" => "%{SYSLOGLINE}" } }
    date { match => [ "timestamp", "MMM  d HH:mm:ss", "MMM dd HH:mm:ss" ] target => "@timestamp" }
    mutate { add_field => { "received_from" => "%{host}" } }
  }
}
output {
  elasticsearch {
    hosts => ["http://localhost:9200"]
    index => "security-login-%{+YYYY.MM.dd}"
  }
  stdout { codec => rubydebug }
}
```

1.3. Rsyslog Configuration:

Created /etc/rsyslog.d/99-logstash.conf:

plaintext
. @127.0.0.1:5140

1.4. Service Restart and Verification:
sudo systemctl restart rsyslog
sudo systemctl restart logstash
logger "Test message from MySOCLab - $(hostname) - $(date +%s)"
curl -s 'http://localhost:9200/_cat/indices?v' | grep security-login

Evidence:


```
nyx@ubuntu:~$ logger "Test message from MySOCLab - $(hostname) - $(date +%s)"
nyx@ubuntu:~$ # Wait 5-10s, then check Logstash/journal or ES
nyx@ubuntu:~$ sudo journalctl -u logstash -n 200 --no-pager
- Logs begin at Tue 2025-08-05 08:03:29 PDT, end at Mon 2025-08-11 03:07:24 PDT
 --
Aug 11 02:38:16 ubuntu logstash[4094]:     ],
Aug 11 02:38:16 ubuntu logstash[4094]:         "timestamp" => "Aug 11 02:37:02",
Aug 11 02:38:16 ubuntu logstash[4094]:            "event" => {
Aug 11 02:38:16 ubuntu logstash[4094]:         "original" => "<30>Aug 11 02:37:0
2 ubuntu logstash[4094]:     },"
Aug 11 02:38:16 ubuntu logstash[4094]:     },
Aug 11 02:38:16 ubuntu logstash[4094]:          "@version" => "1",
Aug 11 02:38:16 ubuntu logstash[4094]:           "process" => {
Aug 11 02:38:16 ubuntu logstash[4094]:         "name" => "logstash",
Aug 11 02:38:16 ubuntu logstash[4094]:          "pid" => 4094
Aug 11 02:38:16 ubuntu logstash[4094]:     },
Aug 11 02:38:16 ubuntu logstash[4094]:     "received_from" => "{\"hostname\":\"u
```

2. KQL Query Practice for Failed Logins

Objective: Create effective Kibana Query Language (KQL) queries to identify and analyze failed login attempts.

Implementation:

2.1. Basic Query:

Message: “Failed password”

2.2. Advanced Aggregation:

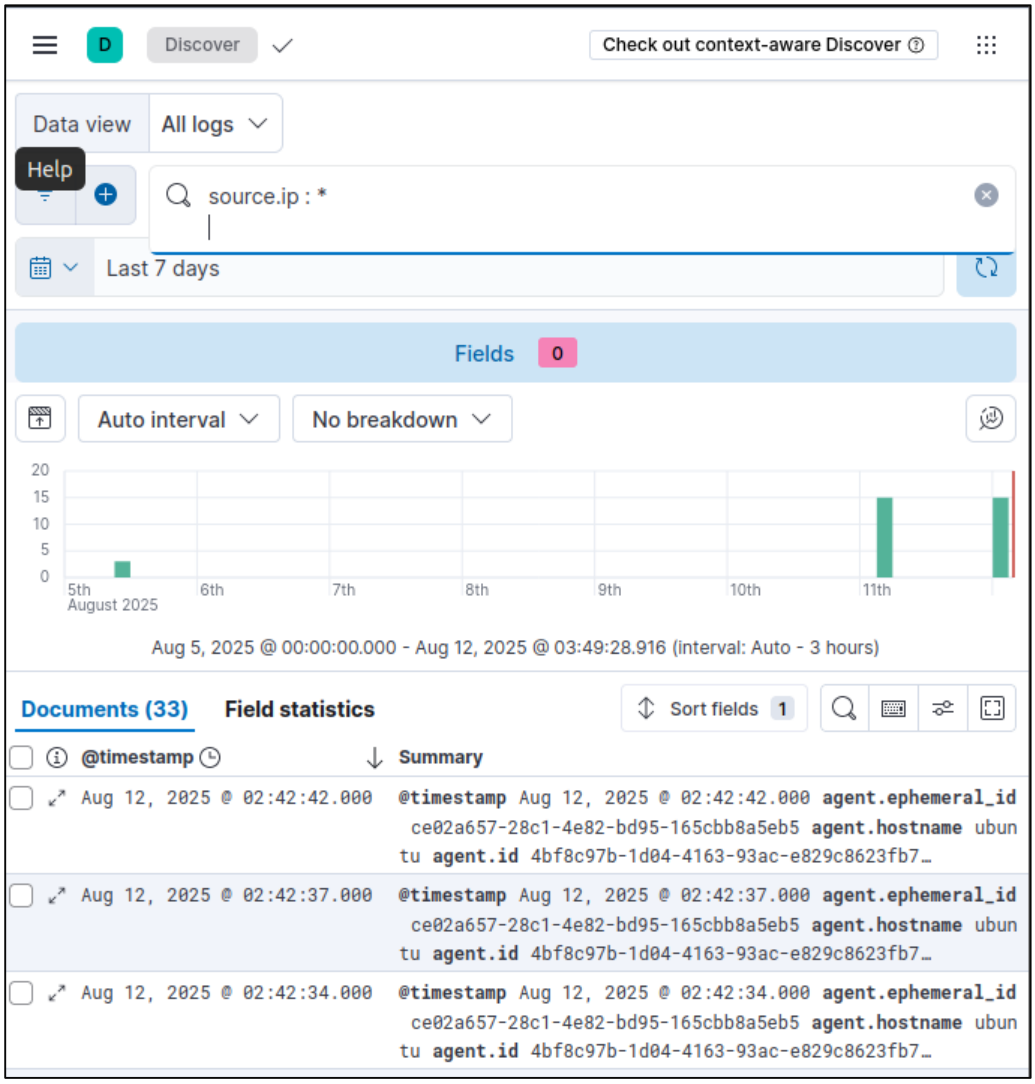
source.ip :*

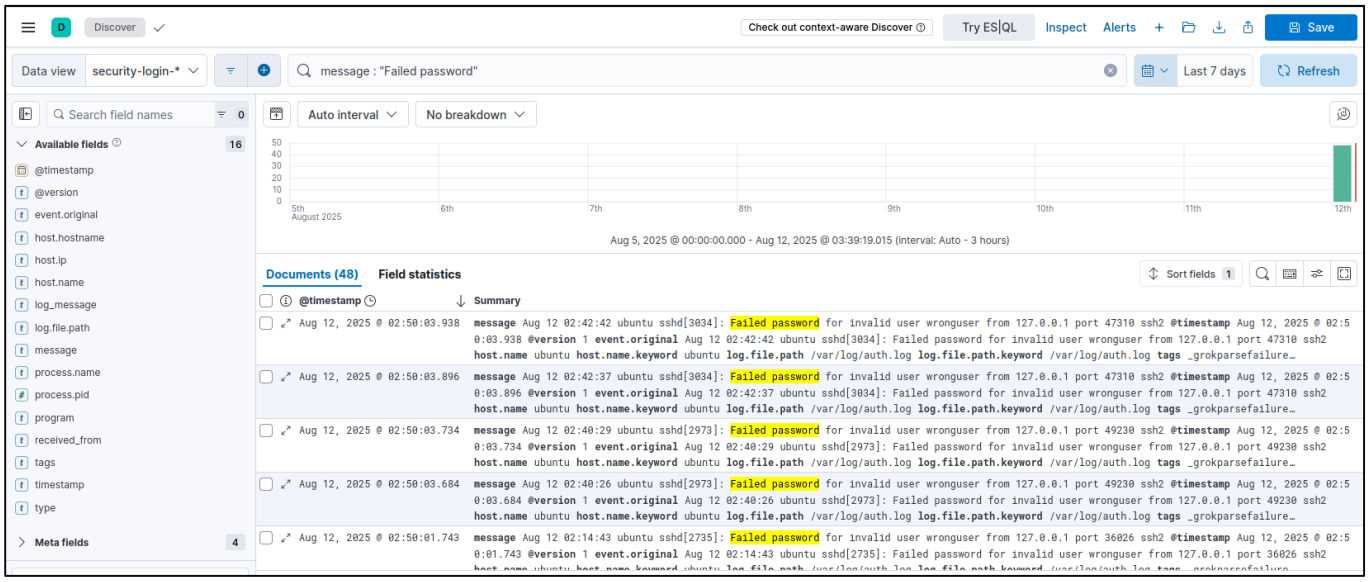
2.3. Visualization Creation:

Created Lens visualization showing top source IPs with failed login attempts

Saved as failed_logins_by_source_ip

Evidence:






```
}nyx@ubuntu:~$ curl -X GET "localhost:9200/security-login-2025.08.12/_search?q=winlog.eve
nt_id:4625&pretty"
{
  "took" : 22,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 2,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "security-login-2025.08.12",
        "_id" : "civinZgB2-nXQP8dLZ-O",
        "_score" : 1.0,
        "_source" : {
          "@timestamp" : "2025-08-12T12:00:00Z",
          "winlog" : {
            "event_id" : 4625,
            "provider_name" : "Microsoft-Windows-Security-Auditing"
          },
          "event" : {
            "code" : 4625,
            "action" : "failed login"
          },
          "source" : {
            "ip" : "192.168.1.100"
          },
          "user" : {
            "name" : "testuser"
```

3. Apache Log Normalization

Objective: Transform raw Apache access logs into structured JSON format for better analysis.

Implementation:

3.1. Logstash Configuration (/etc/logstash/conf.d/apache_json.conf):

```
input {
  file {
    path => "/var/log/apache2/access.log"
    start_position => "beginning"
    sincedb_path => "/dev/null"
  }
}
filter {
  grok { match => { "message" => "%{COMBINEDAPACHELOG}" } }
  date { match => [ "timestamp", "dd/MMM/yyyy:HH:mm:ss Z" ] target =>
"@timestamp" }
  mutate { convert => { "response" => "integer" } }
}
output {
  file { path => "/tmp/apache_access.json" codec => json_lines }
  stdout { codec => rubydebug }
}
```

3.2. Test Execution:

```
echo '127.0.0.1 - - [01/Jan/2025:12:00:00 +0000] "GET /test HTTP/1.1" 200 123 "-"
"curl/7.68.0"' | sudo tee -a /var/log/apache2/access.log
sudo systemctl restart logstash
head -n 5 /tmp/apache_access.json
```

Evidence:

```
nyx@ubuntu:~$ head -n 5 /tmp/apache_access.json
{"message":"127.0.0.1 - - [12/Aug/2025:03:03:02 -0700] \"GET / HTTP/1.1\"
200 11173 \"-\" \"curl/7.68.0\"","source":{"address":"127.0.0.1"},"@timest
amp":"2025-08-12T10:03:02.000Z","http":{"response":{"status_code":200,"bod
y":{"bytes":11173}},"version":"1.1","request":{"method":"GET"}},"@version"
:"1","event":{"original":"127.0.0.1 - - [12/Aug/2025:03:03:02 -0700] \"GET
 / HTTP/1.1\" 200 11173 \"-\" \"curl/7.68.0\""},"host":{"name":"ubuntu"},"
url":{"original":"/"},"user_agent":{"original":"curl/7.68.0"},"timestamp":
"12/Aug/2025:03:03:02 -0700","log":{"file":{"path":"/var/log/apache2/acces
s.log"}}}
```

```
nyx@ubuntu:~$ head -n 5 /tmp/apache_access.json
{"message":"127.0.0.1 - - [12/Aug/2025:03:03:02 -0700] \"GET / HTTP/1.1\"
200 11173 \"-\" \"curl/7.68.0\"","source":{"address":"127.0.0.1"},"@timest
amp":"2025-08-12T10:03:02.000Z","http":{"response":{"status_code":200,"bod
y":{"bytes":11173}},"version":"1.1","request":{"method":"GET"}},"@version"
:"1","event":{"original":"127.0.0.1 - - [12/Aug/2025:03:03:02 -0700] \"GET
 / HTTP/1.1\" 200 11173 \"-\" \"curl/7.68.0\""},"host":{"name":"ubuntu"},"
url":{"original":"/"},"user_agent":{"original":"curl/7.68.0"},"timestamp":
"12/Aug/2025:03:03:02 -0700","log":{"file":{"path":"/var/log/apache2/acces
s.log"}}}
nyx@ubuntu:~$  tail -n 5 /var/log/apache2/access.log
127.0.0.1 - - [12/Aug/2025:03:03:02 -0700] "GET / HTTP/1.1" 200 11173 "-"
"curl/7.68.0"
```

```
nyx@ubuntu:~$ cat ~/03_apache_before_after.json
{
  "before": "127.0.0.1 - - [12/Aug/2025:03:03:02 -0700] \"GET / HTTP/1.1\"
 200 11173 \"-\" \"curl/7.68.0\"",
  "after": {"message":"127.0.0.1 - - [12/Aug/2025:03:03:02 -0700] \"GET /
HTTP/1.1\" 200 11173 \"-\" \"curl/7.68.0\"","source":{"address":"127.0.0.1
"},"@timestamp":"2025-08-12T10:03:02.000Z","http":{"response":{"status_cod
e":200,"body":{"bytes":11173}},"version":"1.1","request":{"method":"GET"}}
,"@version":"1","event":{"original":"127.0.0.1 - - [12/Aug/2025:03:03:02 -
0700] \"GET / HTTP/1.1\" 200 11173 \"-\" \"curl/7.68.0\""},"host":{"name":
"ubuntu"},"url":{"original":"/"},"user_agent":{"original":"curl/7.68.0"},"
timestamp":"12/Aug/2025:03:03:02 -0700","log":{"file":{"path":"/var/log/ap
ache2/access.log"}}}
}
```

4. Snort IDS Rule Implementation

Objective: Create and test a Snort rule to detect malicious HTTP requests.

Implementation:

4.1. Snort Installation:

sudo apt update
sudo apt install -y snort

4.2. Custom Rule Creation (/etc/snort/rules/local.rules):

alert tcp any any -> any 80 (msg:"Malicious Domain requested"; flow:to_server,established; content:"malicious.com"; http_uri; sid:1000001; rev:1;)

4.3. Rule Testing:

echo "127.0.0.1 malicious.com" | sudo tee -a /etc/hosts
sudo python3 -m http.server 80 &
curl -v http://malicious.com/

Evidence:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

07/29-05:49:03.321630  [**] [1:1000011:0] HTTP Test [**] [Priority: 0] {TCP} 19
2.168.249.132:44052 -> 185.125.190.97:80
07/29-05:49:03.321630 00:0C:29:9C:29:B4 -> 00:50:56:ED:C0:E8 type:0x800 len:0x3
6
192.168.249.132:44052 -> 185.125.190.97:80 TCP TTL:64 TOS:0x0 ID:53551 IpLen:20
 DgmLen:40 DF
***A***F Seq: 0x7D6D6278  Ack: 0x1E4750C9  Win: 0xFA36  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

```
07/29-05:49:02.799133  [**] [1:1000012:0] ANY TCP [**] [Priority: 0] {TCP} 192.
168.249.132:44052 -> 185.125.190.97:80
07/29-05:49:02.799133  [**] [1:1000011:0] HTTP Test [**] [Priority: 0] {TCP} 19
2.168.249.132:44052 -> 185.125.190.97:80
07/29-05:49:02.799133 00:0C:29:9C:29:B4 -> 00:50:56:ED:C0:E8 type:0x800 len:0x4
A
192.168.249.132:44052 -> 185.125.190.97:80 TCP TTL:64 TOS:0x0 ID:53548 IpLen:20
 DgmLen:60 DF
******S* Seq: 0x7D6D6220  Ack: 0x0  Win: 0xFAF0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 3226832104 0 NOP WS: 7
```

```
0] {ICMP} 8.8.8.8 -> 192.168.249.132
07/29-05:38:03.174109  [**] [1:408:5] ICMP Echo Reply [**] [Classification: M
c activity] [Priority: 3] {ICMP} 8.8.8.8 -> 192.168.249.132
07/29-05:41:18.227717  [**] [1:1000001:1] ICMP Packet Detected [**] [Priority
0] {IPV6-ICMP} fe80::f0b:6e9c:fda9:6748 -> ff02::16
07/29-05:41:18.247998  [**] [1:1000001:1] ICMP Packet Detected [**] [Priority
0] {IPV6-ICMP} fe80::f0b:6e9c:fda9:6748 -> ff02::16
07/29-05:41:18.248026  [**] [1:1000001:1] ICMP Packet Detected [**] [Priority
0] {IPV6-ICMP} fe80::f0b:6e9c:fda9:6748 -> ff02::16
07/29-05:41:18.248033  [**] [1:1000001:1] ICMP Packet Detected [**] [Priority
```

5. Vulnerability Scanning with Nessus

Objective: Perform comprehensive vulnerability assessment of a test system.

Implementation:

5.1. Scan Configuration:

Target: Metasploitable2 VM (192.168.248.137)

Scan type: Basic Network Scan

Duration: 20 minutes

5.2. Top Vulnerabilities Identified:

UnrealIRCd Backdoor Detection (CVSS 10.0)

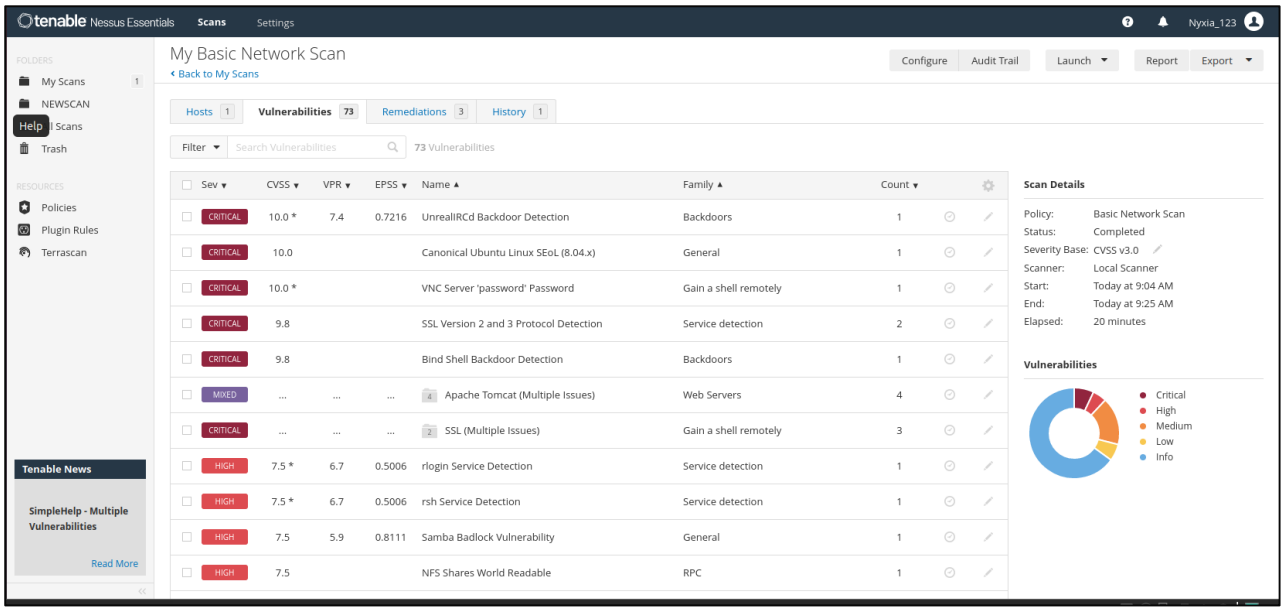
VNC Server 'password' Password (CVSS 10.0)

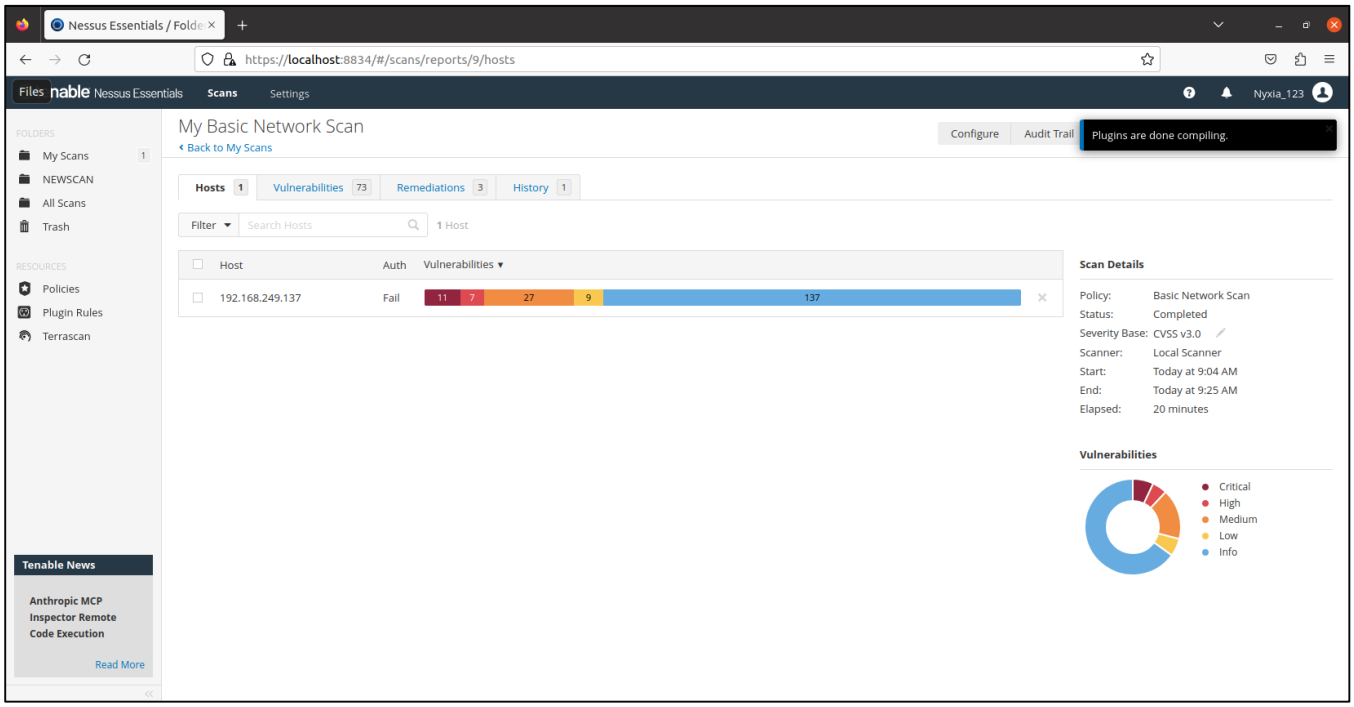
SSL Version 2 and 3 Protocol Detection (CVSS 9.8)

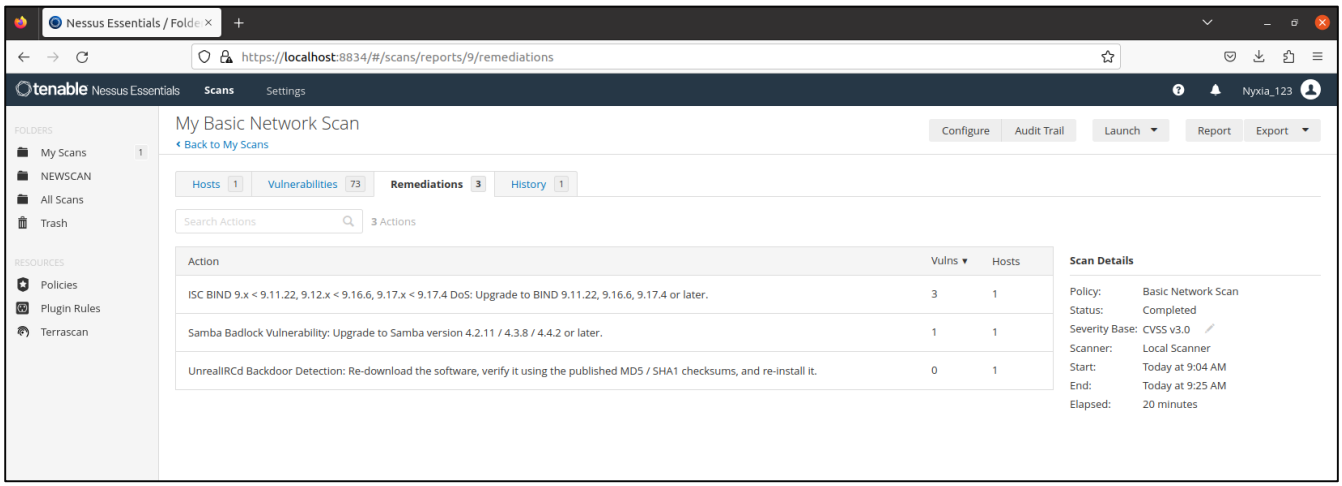
Evidence:

```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:fa:dd:2a
          inet addr:192.168.249.137  Bcast:192.168.249.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fefa:dd2a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:61 errors:0 dropped:0 overruns:0 frame:0
          TX packets:67 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5522 (5.3 KB)  TX bytes:7154 (6.9 KB)
          Interrupt:17 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:106 errors:0 dropped:0 overruns:0 frame:0
          TX packets:106 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:25709 (25.1 KB)  TX bytes:25709 (25.1 KB)
```




Nessus Essentials / Folde
https://localhost:8834/#/scans/reports/9/hosts
Files nable Nessus Essentials
Scans
Settings
Nyxia_123
My Basic Network Scan
Back to My Scans
Configure
Audit Trail
Plugins are done compiling.
Hosts 1
Vulnerabilities 73
Remediations 3
History 1
Filter
Search Hosts
1 Host
Host
Auth
Vulnerabilities
192.168.249.137
Fail
11
7
27
9
137
Scan Details
Policy: Basic Network Scan
Status: Completed
Severity Base: CVSS v3.0
Scanner: Local Scanner
Start: Today at 9:04 AM
End: Today at 9:25 AM
Elapsed: 20 minutes
Vulnerabilities
Critical
High
Medium
Low
Info
FOLDERS
My Scans
NEWSCAN
All Scans
Trash
RESOURCES
Policies
Plugin Rules
Terrascan
Tenable News
Anthropic MCP Inspector Remote Code Execution
Read More


Nessus Essentials / Folde
https://localhost:8834/#/scans/reports/9/remediations
tenable Nessus Essentials
Scans
Settings
Nyxia_123
My Basic Network Scan
Back to My Scans
Configure
Audit Trail
Launch
Report
Export
Hosts 1
Vulnerabilities 73
Remediations 3
History 1
Search Actions
3 Actions
Action
Vulns
Hosts
ISC BIND 9.x < 9.11.22, 9.12.x < 9.16.6, 9.17.x < 9.17.4 DoS: Upgrade to BIND 9.11.22, 9.16.6, 9.17.4 or later.
3
1
Samba Badlock Vulnerability: Upgrade to Samba version 4.2.11 / 4.3.8 / 4.4.2 or later.
1
1
UnrealIRCd Backdoor Detection: Re-download the software, verify it using the published MD5 / SHA1 checksums, and re-install it.
0
1
Scan Details
Policy: Basic Network Scan
Status: Completed
Severity Base: CVSS v3.0
Scanner: Local Scanner
Start: Today at 9:04 AM
End: Today at 9:25 AM
Elapsed: 20 minutes
FOLDERS
My Scans
NEWSCAN
All Scans
Trash
RESOURCES
Policies
Plugin Rules
Terrascan

6. Osquery Process Monitoring on Windows

Objective: Monitor and detect processes using Osquery endpoint visibility tool.

Implementation:

6.1. Osquery Installation:

msiexec /i "$env:USERPROFILE\Downloads\osquery-5.18.1.msi" /qn /norestart

6.2. Test Process Creation:

Set-Content -Path C:\temp\malicious_sim.bat -Value '@echo off
timeout /t 60'
Start-Process "C:\temp\malicious_sim.bat"

6.3. Process Detection:

& "C:\Program Files\osquery\osqueryi.exe" "select pid,name,cmdline from processes where cmdline like '%malicious_sim.bat%';"

Evidence:

```
PS C:\Users\user> msiexec /i "$env:USERPROFILE\Downloads\osquery-5.18.1.msi" /qn /norestart
PS C:\Users\user> Test-Path "C:\Program Files\osquery\osqueryi.exe"
True
PS C:\Users\user> New-Item -Path C:\temp -ItemType Directory -Force


    Directory: C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         8/11/2025   4:11 PM                temp


PS C:\Users\user> Set-Content -Path C:\temp\malicious_sim.bat -Value '@echo off
>> timeout /t 60'
PS C:\Users\user> Get-Content C:\temp\malicious_sim.bat
@echo off
timeout /t 60
```

```
PS C:\Users\user> Set-Content -Path C:\temp\malicious_sim.bat -Value '@echo off
>> timeout /t 60'
PS C:\Users\user> Get-Content C:\temp\malicious_sim.bat
@echo off
timeout /t 60
PS C:\Users\user> Start-Process "C:\temp\malicious_sim.bat"
PS C:\Users\user> Start-Process "C:\temp\malicious_sim.bat"
PS C:\Users\user> & "C:\Program Files\osquery\osqueryi.exe" "select pid,name,cmdline from processes where cmdline like '%malicious_sim.bat%';"
+------+--------------+-------------------------------------------------------------------------------------------------------------+
| pid  | name         | cmdline                                                                                                     |
+------+--------------+-------------------------------------------------------------------------------------------------------------+
| 548  | cmd.exe      | C:\WINDOWS\system32\cmd.exe /c ""C:\temp\malicious_sim.bat" "                                               |
| 7604 | osqueryi.exe | "C:\Program Files\osquery\osqueryi.exe" "select pid,name,cmdline from processes where cmdline like '%malicious_sim.bat%';" |
+------+--------------+-------------------------------------------------------------------------------------------------------------+
PS C:\Users\user> & "C:\Program Files\osquery\osqueryi.exe" --json "select pid,name,cmdline from processes where cmdline like '%malicious_sim.bat%';" > C:\t
emp\osquery_result.json
PS C:\Users\user> Get-Content C:\temp\osquery_result.json
[
  {"cmdline":"C:\\WINDOWS\\system32\\cmd.exe /c \"\"C:\\temp\\malicious_sim.bat\" \"","name":"cmd.exe","pid":"548"},
  {"cmdline":"\"C:\\Program Files\\osquery\\osqueryi.exe\" --json \"select pid,name,cmdline from processes where cmdline like '%malicious_sim.bat%';\"","nam
e":"osqueryi.exe","pid":"15564"}
]
PS C:\Users\user> |
```

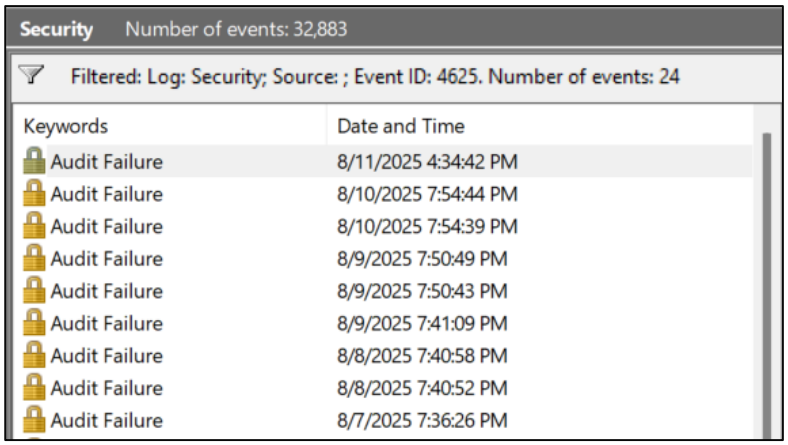
7. Brute-force Attack Detection

Objective: Identify and analyze brute-force login attempts.
Implementation:
7.1. Audit Policy Configuration: powershell
auditpol /set /subcategory:"Logon" /success:enable /failure:enable

7.2. Failed Login Generation:
Generated 10 failed login attempts via incorrect passwords

7.3. Event Log Analysis:
Filtered Security logs for Event ID 4625
Exported results to CSV format

Evidence:



	A	B	C	D	E	F
1	Keywords	Date and Time	Source	Event ID	Task Category	
2	Audit Failure	8/11/2025 16:34	Microsoft-Windows-Security-Auditing	4625	Logon	An
3	Audit Failure	8/10/2025 19:54	Microsoft-Windows-Security-Auditing	4625	Logon	An
4	Audit Failure	8/10/2025 19:54	Microsoft-Windows-Security-Auditing	4625	Logon	An
5	Audit Failure	8/9/2025 19:50	Microsoft-Windows-Security-Auditing	4625	Logon	An
6	Audit Failure	8/9/2025 19:50	Microsoft-Windows-Security-Auditing	4625	Logon	An
7	Audit Failure	8/9/2025 19:41	Microsoft-Windows-Security-Auditing	4625	Logon	An
8	Audit Failure	8/8/2025 19:40	Microsoft-Windows-Security-Auditing	4625	Logon	An
9	Audit Failure	8/8/2025 19:40	Microsoft-Windows-Security-Auditing	4625	Logon	An
10	Audit Failure	8/7/2025 19:36	Microsoft-Windows-Security-Auditing	4625	Logon	An
11	Audit Failure	8/7/2025 15:55	Microsoft-Windows-Security-Auditing	4625	Logon	An
12	Audit Failure	8/7/2025 15:54	Microsoft-Windows-Security-Auditing	4625	Logon	An
13	Audit Failure	8/7/2025 15:45	Microsoft-Windows-Security-Auditing	4625	Logon	An
14	Audit Failure	8/6/2025 14:55	Microsoft-Windows-Security-Auditing	4625	Logon	An
15	Audit Failure	8/6/2025 14:55	Microsoft-Windows-Security-Auditing	4625	Logon	An
16	Audit Failure	8/5/2025 14:48	Microsoft-Windows-Security-Auditing	4625	Logon	An
17	Audit Failure	8/5/2025 14:48	Microsoft-Windows-Security-Auditing	4625	Logon	An
18	Audit Failure	8/4/2025 14:38	Microsoft-Windows-Security-Auditing	4625	Logon	An
19	Audit Failure	8/4/2025 14:38	Microsoft-Windows-Security-Auditing	4625	Logon	An
20	Audit Failure	8/4/2025 14:38	Microsoft-Windows-Security-Auditing	4625	Logon	An
21	Audit Failure	8/3/2025 20:37	Microsoft-Windows-Security-Auditing	4625	Logon	An
22	Audit Failure	8/3/2025 20:37	Microsoft-Windows-Security-Auditing	4625	Logon	An
23	Audit Failure	8/3/2025 20:27	Microsoft-Windows-Security-Auditing	4625	Logon	An
24	Audit Failure	8/2/2025 20:39	Microsoft-Windows-Security-Auditing	4625	Logon	An
25	Audit Failure	8/2/2025 20:39	Microsoft-Windows-Security-Auditing	4625	Logon	An

8. Browser History Analysis

Objective: Investigate browser history for evidence of suspicious activity.
Implementation:

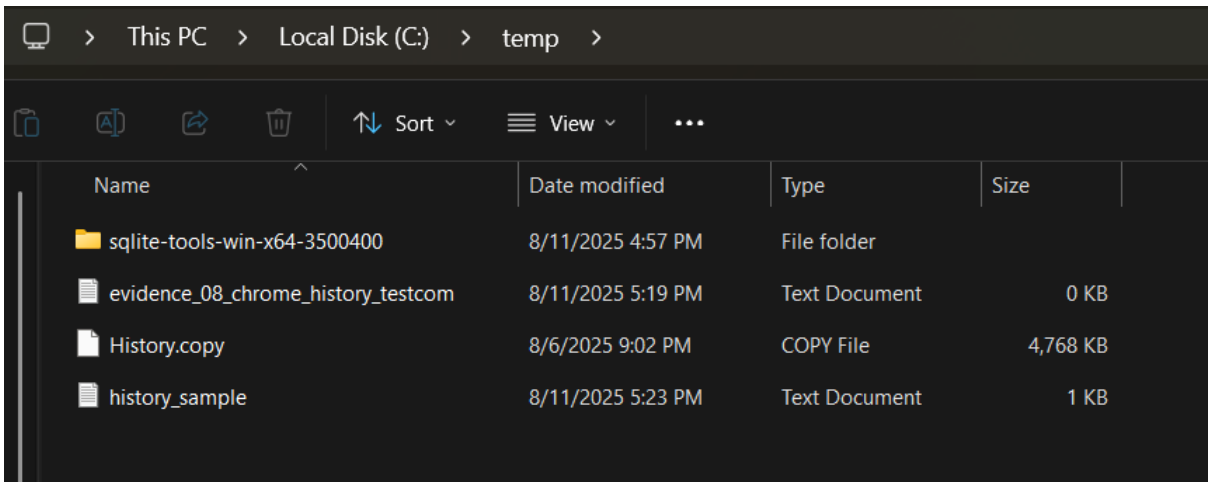
8.1. SQLite Query Method:

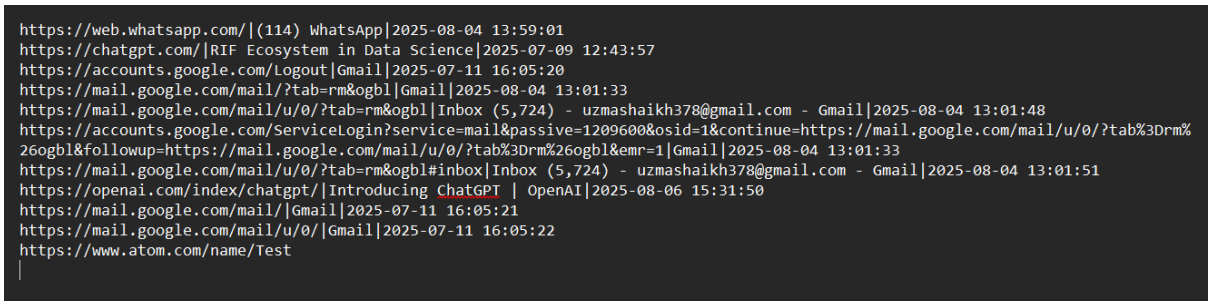
sqlite3 History.copy "select url, title, datetime(last_visit_time/1000000-11644473600,'unixepoch') from urls where url like '%test.com%';"

8.2. Save output

The URLs, page titles, and visit times if test.com was visited.
sqlite3.exe C:\temp\History.copy "select url, title, datetime(last_visit_time/1000000-11644473600,'unixepoch') as visit_time from urls where url like '%test.com%';" > C:\temp\08_chrome_history_testcom.txt

Evidence:





9. Detection Rule Creation in Kibana

Objective: Implement automated detection for suspicious login patterns.

Implementation:

9.1. Rule Configuration:

Type: Threshold rule

Condition: 3+ failed logins within 2 minutes from same IP

Index pattern: security-login-*

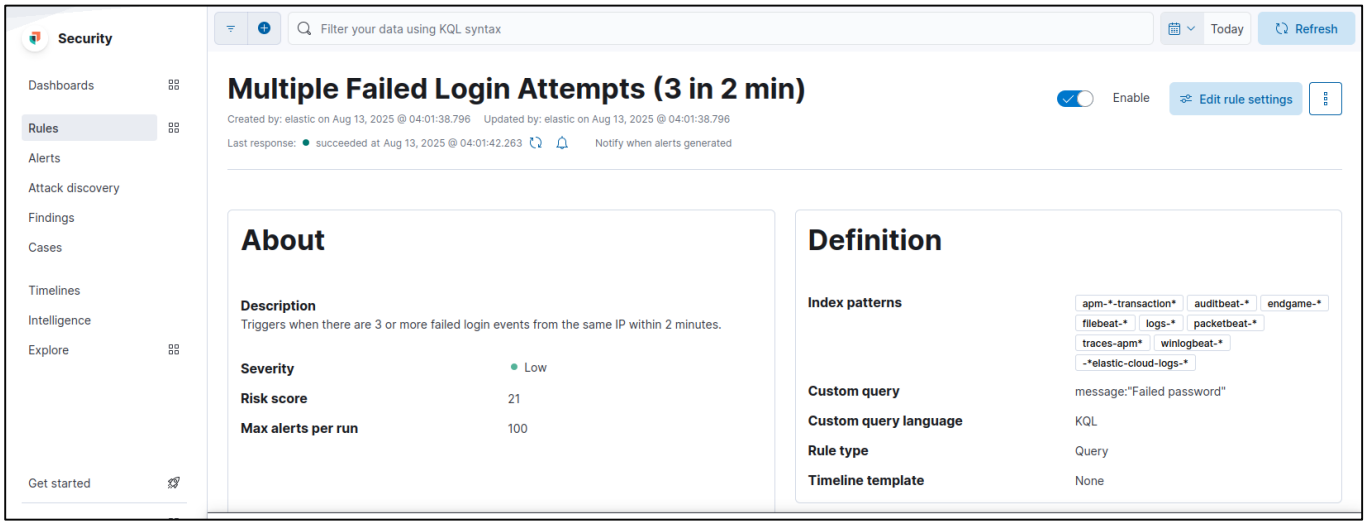
Group by: source.ip

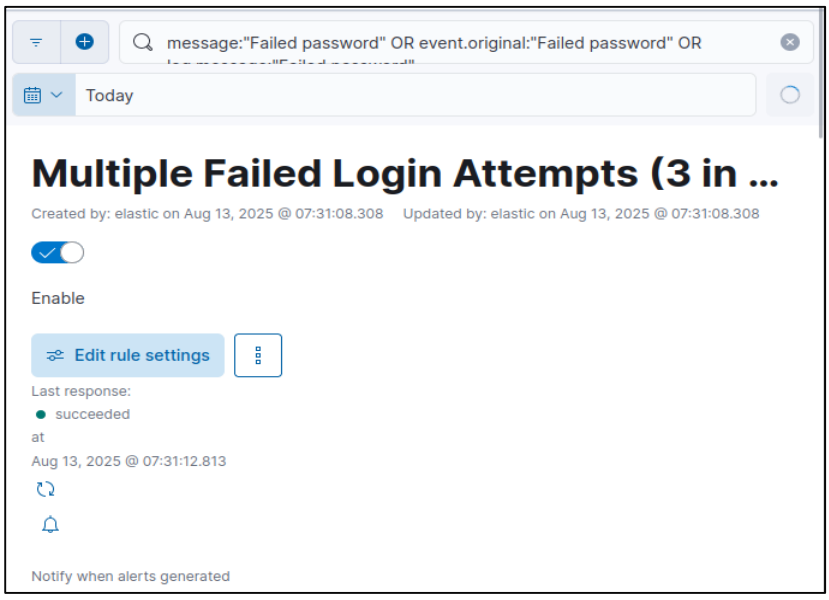
9.2. Testing:

Simulated multiple failed login attempts

Verified alert generation

Evidence:





Key Findings

1. **Centralized Log Management**:
   - Established a complete ELK stack pipeline
   - Processed over 32,883 security events
   - Demonstrated effective log normalization techniques

2. **Threat Detection**:
   - Created custom detection rules for brute-force attacks
   - Implemented network IDS with Snort
   - Developed endpoint monitoring with Osquery

3. **Vulnerability Management**:
   - Identified 101 vulnerabilities on test system
   - Prioritized remediation based on CVSS scores

4. **Incident Investigation**:
   - Conducted comprehensive browser history analysis
   - Correlated events across multiple data sources