

CYBER SECURITY INTERNSHIP REPORT



SOC CAPSTONE INCIDENT REPORT

ASSIGNMENT

BY

ATHANASIUS J.K GADOSEY

Introduction

Task Overview

Web Application Compromise Leading to Persistence, Lateral Movement, and Data Exfiltration

Executive Summary

This incident involved a coordinated cyberattack against a company's web server that resulted in unauthorised access, persistence, internal network scanning, and successful data exfiltration. The attacker leveraged a common web attack technique to gain initial access, escalated their foothold by brute-forcing credentials, and maintained access through a malicious cron job. The breach was ultimately detected after abnormal outbound data transfers were observed..

Tools Used

- **SIEM Tool:** Splunk Free Trial (cloud instance) – used to query and analyze the provided sample logs using SPL searches.
- **Sample logs:** Final_project_logs – included web server logs, authentication logs, system logs, and network/proxy logs used for investigation.
- **Artifact Analysis:** Base64 decoding of suspicious cron job commands to uncover hidden attacker framework details.
- **Framework Reference:** MITRE ATT&CK Framework – used to map attacker actions to known tactics and techniques.
- **Documentation:** Google Docs / PDF – used to document findings, timelines, IOCs, and final recommendations.
- **Analysis Methodology:** Log filtering, pattern matching, and correlation rules across multiple log sources

Incident Analysis Summary

| Alert ID | Type of Threat | Severity | Description | Action Taken |
|----------|--------------------------------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| A001 | SQL Injection (Reconnaissance) | High | The attacker sent URL-encoded SQL payloads (%27, UNION SELECT) which is via a web request to probe the backend database structure and confirm SQL injection vulnerability. | The suspicious web requests were reviewed and marked as malicious, and the application was recommended for security hardening. |
| A002 | SSH Brute Force Attack | High | An external IP address (103.45.12.90) performed multiple failed SSH login attempts against common usernames, indicating a brute-force attack | The repeated login failures were identified, and the source IP was flagged for blocking. |
| A003 | Unauthorized System Access | Critical | The attacker successfully logged into the server using a compromised service account (svc_backup), gaining unauthorized access to the system | The incident was escalated immediately, and the affected account was identified for password reset and access restriction. |

| | | | | |
|------|----------------------------------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| A004 | Persistence via Malicious Script | Critical | A hidden, encoded script was executed to maintain long-term access to the server. This allowed the attacker to reconnect to the system remotely | The malicious script was decoded, analysed, and marked for removal to prevent further access. |
| A005 | Internal Network Scanning | High | The attacker used nmap to perform a TCP SYN scan (-sS -T4) on the internal subnet 172.16.0.0/24 | The scanning activity was detected, and the internal network was reviewed for additional exposure. |
| A006 | Data Exfiltration | Critical | A large database file (db_snap_v2.sql) was downloaded and secretly transferred large amounts of data to an external domain outside the organization. | The data transfer activity was analyzed, confirmed as malicious, and classified as a serious data breach. |

Incident Timeline Summary

| Time (UTC) | Event |
|-------------|-----------------------------------------------------------------------|
| 14:00:01 | SQL Injection reconnaissance via encoded UNION SELECT |
| 14:05:12 | SSH brute force attempts detected |
| 14:10:45 | Successful SSH login as svc_backup |
| 14:12:00 | File upload endpoint accessed |
| 14:20:05 | Path traversal attempt on /etc/passwd |
| 14:26:10 | nmap installed, and an installed scan initiated |
| 14:30:15 | Database dump downloaded |
| 14:40:01-05 | Data exfiltration to dev-null.io |

Figure 1: Initial Log Overview in Splunk

This screenshot shows all ingested logs before any filters or queries were applied. It displays raw events, including login attempts, access logs, etc. This general view provides situational awareness and sets the foundation for triage and deeper investigation.

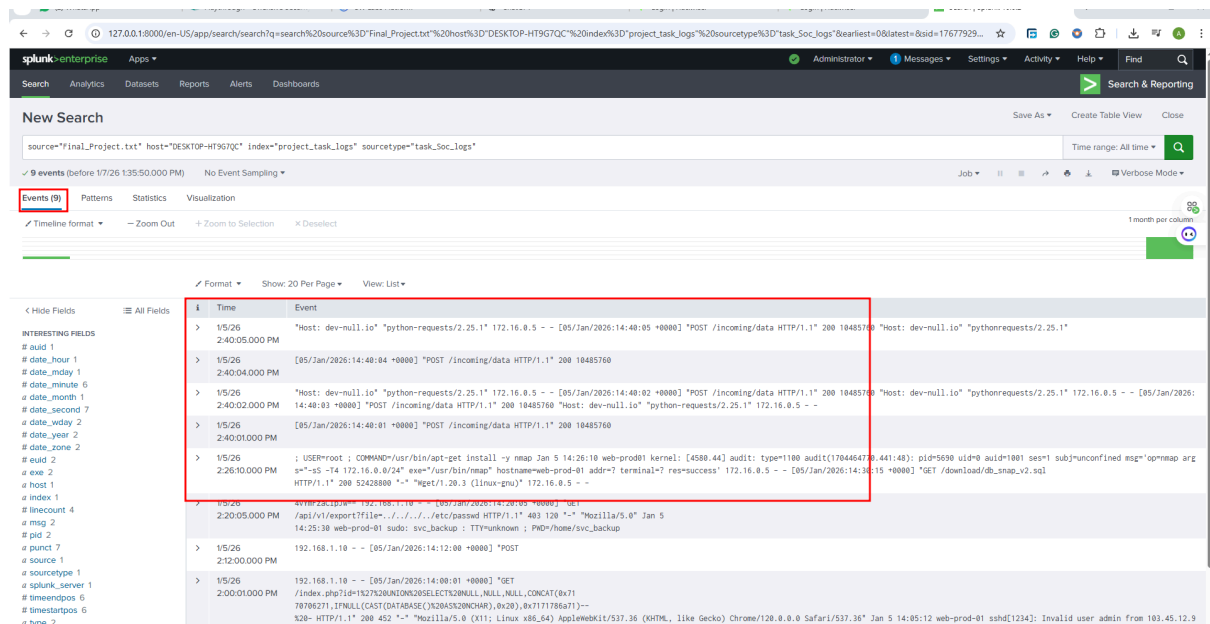


Figure 1: Initial Log Overview in Splunk

1. Initial Reconnaissance

Finding

The screenshot below shows how the attacker used a **URL-encoded SQL Injection** attack to probe the backend database.

The screenshot displays the Splunk Enterprise search interface. The search bar contains the following query:

```
source="Final_Project.txt" host="DESKTOP-HT9G7QC" index="tasks_log" sourcetype="Projects_task_log" | rex field=_raw "(?<src_ip>\d{1,3}(?:\.\d{1,3}){3})" | rex field=_raw "\"(GET|POST)\s+(?<uri>\/[^\s\"]+)" | search uri="*%*" | sort _time | table _time src_ip uri
```

A red arrow points to the query with the text "this shows the SPL query used".

The results table shows the following data:

| i | Time | Event |
|---|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| > | 1/5/26 2:00:01.000 PM | 192.168.1.10 - - [05/Jan/2026:14:00:01 +0000] "GET /index.php?id=127%20UNION%20SELECT%20NULL,NULL,CONCAT(0x7170786271,IFNULL(CAST(DATABASE()ASCHAR),0x20),0x7171786a71)--%20- HTTP/1.1" 200 452 "-" Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36 Jan 5 14:05:12 web-prod-01 sshd[1234]: Invalid user admin from 103.45.12.90 port 54322 Jan 5 14:05:15 web-prod-01 sshd[1234]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=103.45.12.90 user=root Jan 5 14:10:45 web-prod-01 sshd[1234]: Accepted password for svc_backup from 103.45.12.90 port 54322 ssh2 |

A red arrow points to the injected payload in the log entry with the text "this indicates the URL-encoded SQL Injection".

The Splunk Query Used below:

```
source="Final_Project.txt" host="DESKTOP-HT9G7QC"
index="tasks_log" sourcetype="Projects_task_log" | rex field=_raw
"^(?<src_ip>\d{1,3}(?:\.\d{1,3}){3})"
| rex field=_raw "\"(GET|POST)\s+(?<uri>\/[^\s\"]+)"
| search uri="*%*"
| sort _time
| table _time src_ip uri
```

The Encoded URL String is observed below:

%27%20UNION%20SELECT%20NULL,NULL,NULL,CONCAT

```
source="Final_Project.txt" host="DESKTOP-HT9G7QC" index="tasks_log" sourcetype="Projects_task_log" | rex field=_raw "^(?<src_ip>\d{1,3})(?\.\\.\d{1,3}){3})"
| rex field=_raw "\^(GET|POST)\s+(?<uri>\/[^\s\"]+)"
| search uri="*%*"
| sort _time
| table _time src_ip uri
```

✓ 1 event (before 1/9/26 12:27:47.000 PM) No Event Sampling ▾

Events (1) Patterns **Statistics (1)** Visualization

Show: 20 Per Page ▾ ☒ Format ▾ ☒ Preview: On

| _time ↕ | src_ip ↕ | uri ↕ |
|---------------------|--------------|-----------------------------------------------------------------|
| 2026-01-05 14:00:01 | 192.168.1.10 | /index.php?id=%27%20UNION%20SELECT%20NULL,NULL,NULL,CONCAT 0x71 |

Attack Type

SQL Injection (UNION-based)

Attacker Objective

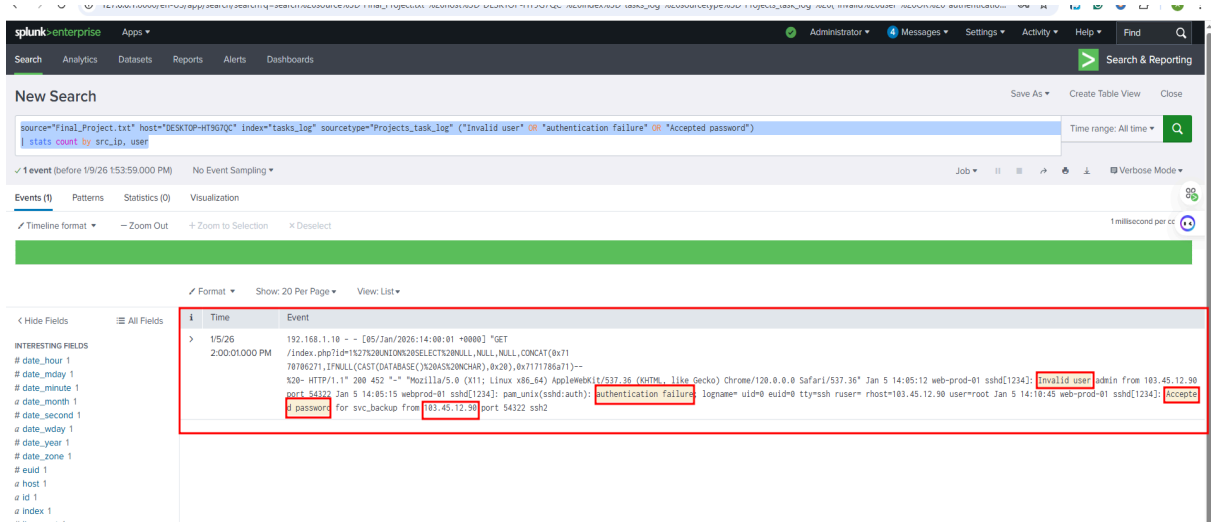
The attacker was trying to:

- Identify database structure
- Enumerate the active database name
- Confirm vulnerability to **UNION SELECT** payloads.

2. The Pivot (Brute Force & Account Compromise)

Finding

This screenshot highlights an **external IP address** that performed repeated SSH authentication failures and later succeeded.



Splunk Query Used:

```
source="Final_Project.txt" host="DESKTOP-HT9G7QC"
index="tasks_log" sourcetype="Projects_task_log" ("Invalid user"
OR "authentication failure" OR "Accepted password")
| stats count by src_ip, user
```

External Attacker IP

103.45.12.90

Successful Login Timestamp

Jan 5 14:10:45

Compromised Internal Account

svc_backup

Explanation

After multiple failed login attempts against common usernames such as **admin** and **root**, the attacker was successfully able to authenticate using a service account, which indicates either **a weak password, password reuse, or credential leakage**.

3. The Hidden Payload (Persistence Mechanism)

Finding

The screenshot shows a **Base64-encoded payload** that was executed via a cron-like persistence mechanism.

Splunk Query Used below:

```
source="Final_Project.txt" host="DESKTOP-HT9G7QC"
index="tasks_log" sourcetype="Projects_tasks_log" | search
"EXECVE" "sh -c" | table _time a2
```

Decoded Payload Results

After decoding the Base64 string, the payload revealed:

- **Attacker IP:** [45.77.102.5](#)
- **Port:** [4444](#)
- **Script Language:** [Python](#)

Explanation

The script used establishes a **reverse shell**, therefore allowing the attacker to maintain persistent remote access to the compromised host.

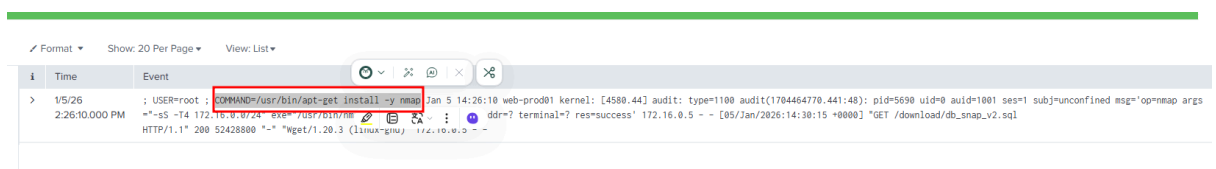
4. Lateral Movement

Finding

The screenshot below shows an attacker conducting **internal network reconnaissance** after gaining access.

Splunk Query Used(Tool installation):

```
source="Final_Project.txt" host="DESKTOP-HT9G7QC"
index="tasks_log" sourcetype="Projects_tasks_log" "apt-get
install"
| table _time user command
```

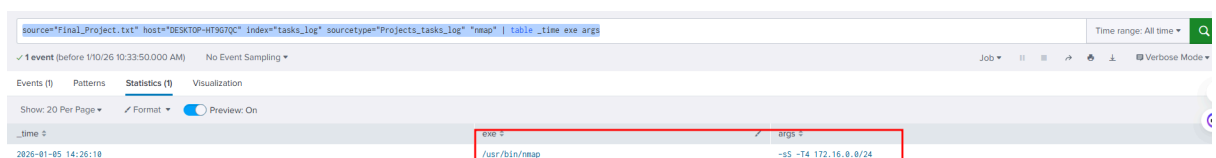


A screenshot of the Splunk search results interface. The search bar contains the query: `source="Final_Project.txt" host="DESKTOP-HT9G7QC" index="tasks_log" sourcetype="Projects_tasks_log" "apt-get install" | table _time user command`. The results table shows a single event from 1/5/26 at 2:26:10.000 PM. The event details include: `COMMAND: /usr/bin/apt-get install -y nmap`, `Jan 5 14:26:10 web-prod01 kernel: [4580.44] audit: type=1100 audit(1704464770.441:48): pid=5690 uid=0 auid=1001 ses=1 subj=unconfined msg='op=nmap args`, and `HTTP/1.1" 200 52428800 "-" "wget/1.20.3 (Linux-gnu) 172.16.0.5 - - [05/Jan/2026:14:30:15 +0000] "GET /download/db_snap_v2.sql`. The `COMMAND` field is highlighted with a red box.

| Time | Event |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1/5/26 2:26:10.000 PM | <code>COMMAND: /usr/bin/apt-get install -y nmap</code> <code>Jan 5 14:26:10 web-prod01 kernel: [4580.44] audit: type=1100 audit(1704464770.441:48): pid=5690 uid=0 auid=1001 ses=1 subj=unconfined msg='op=nmap args</code> <code>HTTP/1.1" 200 52428800 "-" "wget/1.20.3 (Linux-gnu) 172.16.0.5 - - [05/Jan/2026:14:30:15 +0000] "GET /download/db_snap_v2.sql</code> |

SPL Query Used (Nmap scan execution):

```
source="Final_Project.txt" host="DESKTOP-HT9G7QC"
index="tasks_log" sourcetype="Projects_tasks_log" "nmap" | table
_time exe args.
```



A screenshot of the Splunk search results interface. The search bar contains the query: `source="Final_Project.txt" host="DESKTOP-HT9G7QC" index="tasks_log" sourcetype="Projects_tasks_log" "nmap" | table _time exe args`. The results table shows a single event from 2026-01-05 14:26:10. The event details include: `exe: /usr/bin/nmap` and `args: -sS -T4 172.16.0.0/24`. Both the `exe` and `args` fields are highlighted with red boxes.

| Time | Event |
|---------------------|-----------------------------------------------------------------------------|
| 2026-01-05 14:26:10 | <code>exe: /usr/bin/nmap</code> <code>args: -sS -T4 172.16.0.0/24</code> |

Technique for Scanning

- **(-sS) TCP SYN Scan**
- **(-T4) Is Aggressive Timing**

Targeted IP Subnet

172.16.0.0/24

Explanation

The attacker used `apt-get` to install `nmap`, then scanned the internal subnet to identify additional hosts and services, indicating **post-exploitation lateral movement**.

5. The Grand Theft (Data Exfiltration)

After gaining access and performing lateral movement, the attacker proceeded to **exfiltrate sensitive data** from the compromised environment

Filename of the Stolen Data

Splunk Query has been used:

```
source="Final_Project.txt" host="DESKTOP-HT9G7QC"
index="tasks_log" sourcetype="Projects_tasks_log" | search "GET
/download/"
| table _time _raw
```



Stolen Filename

```
Db_snap_v2.sql
```

*This indicates that a **database snapshot** file was successfully downloaded by the attacker*

Total Volume of Data Sent to dev-null.io

The attacker exfiltrated data in multiple POST requests to the external domain dev-null.io.

Splunk Query Used:

```
source="Final_Project.txt" host="DESKTOP-HT9G7QC"
index="tasks_log" sourcetype="Projects_tasks_log" | search
"dev-null.io" | stats sum(bytes) as total_bytes | eval
total_MB=round(total_bytes/1024/1024,2)
```

New Search Save As Create Table View Close

source="Final_Project.txt" host="DESKTOP-HT9G7QC" index="tasks_log" sourcetype="Projects_tasks_log" | search "dev-null.io" | stats sum(bytes) as total_bytes | eval total_MB=round(total_bytes/1024/1024,2)

Time range: All time Q

✓ 2 events (before 1/10/26 12:23:01.000 PM) No Event Sampling Job II III IV V Verbose Mode

Events (2) Patterns Statistics (0) Visualization

Timeline format Zoom Out Zoom to Selection Deselect 100 milliseconds per column

Format Show: 20 Per Page View: List

| < Hide Fields | All Fields | i | Time | Event |
|--------------------|------------|---|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| INTERESTING FIELDS | | > | 1/5/26 2:40:05.000 PM | "Host: dev-null.io" "python-requests/2.25.1" 172.16.0.5 - - [05/Jan/2026:14:40:05 +0000] "POST /incoming/data HTTP/1.1" 200 18485760 "Host: dev-null.io" "pythonrequests/2.25.1" |
| # date_hour 1 | | > | 1/5/26 2:40:02.000 PM | "Host: dev-null.io" "python-requests/2.25.1" 172.16.0.5 - - [05/Jan/2026:14:40:02 +0000] "POST /incoming/data HTTP/1.1" 200 18485760 "Host: dev-null.io" "pythonrequests/2.25.1" 172.16.0.5 - - [05/Jan/2026:14:40:03 +0000] "POST /incoming/data HTTP/1.1" 200 18485760 "Host: dev-null.io" "python-requests/2.25.1" 172.16.0.5 - - |
| # date_mday 1 | | | | |
| # date_minute 1 | | | | |
| a date_month 1 | | | | |
| a date_second 2 | | | | |
| a date_wday 1 | | | | |
| # date_year 1 | | | | |
| # date_zone 1 | | | | |
| a host 1 | | | | |
| a index 1 | | | | |

Rows read: 4

Explanation

- Looking at the **bytes** field records the size of each HTTP POST response.
- The SPL Query used shows that multiple uploads of **10,485,760 bytes** were observed.
- After that, it was summed and converted to megabytes (MB).

Total Data Exfiltrated is 50 MB

*This helped to indicate or confirm that **large-scale data exfiltration**, rather than normal application traffic.*

Malicious Script Responsible for Exfiltration

Splunk Query used:

```
source="Final_Project.txt" host="DESKTOP-HT9G7QC"
index="tasks_log" sourcetype="Projects_tasks_log" | search
"dev-null.io" | table _time _raw
```

New Search

source="Final_Project.txt" host="DESKTOP-HT967QC" index="tasks_log" sourcetype="Projects_tasks_log" | search "dev-null.io" | table _time _raw

Time range: All time

2 events (before 1/10/26 12:44:07:000 PM) No Event Sampling

Job

Timeline format Zoom Out Zoom to Selection Deselect

100 milliseconds per column

Format Show: 20 Per Page View: List

| | Time | Event |
|---|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| > | 1/5/26 2:40:05.000 PM | "Host: dev-null.io" "python-requests/2.25.1" 172.16.0.5 - - [05/Jan/2026:14:40:05 +0000] "POST /incoming/data HTTP/1.1" 200 10485760 "Host: dev-null.io" "pythonrequests/2.25.1" |
| > | 1/5/26 2:40:02.000 PM | "Host: dev-null.io" "python-requests/2.25.1" 172.16.0.5 - - [05/Jan/2026:14:40:02 +0000] "POST /incoming/data HTTP/1.1" 200 10485760 "Host: dev-null.io" "pythonrequests/2.25.1" 172.16.0.5 - - [05/Jan/2026:14:40:03 +0000] "POST /incoming/data HTTP/1.1" 200 10485760 "Host: dev-null.io" "python-requests/2.25.1" 172.16.0.5 - - |

< Hide Fields All Fields

INTERESTING FIELDS

- # date_hour 1
- # date_mday 1
- # date_minute 1
- # date_month 1
- # date_second 2
- # date_wday 1
- # date_year 1
- # date_zone 1
- # host 1

Finding

The User-Agent field repeatedly showed below:

```
python-requests/2.25.1
```

Mailcious Script Identified

A Python script using **python-request**

Explanation

There is repeated automated POST requests using *python-requests* strongly indicate a custom Python exfiltration script, not a browser.