Req 1: Advanced Cybersecurity Defense Strategies Implementation

Introduction

In this project, I set out to design and implement a comprehensive cybersecurity framework that incorporates modern defense strategies. I focused on four key areas: applying a Zero Trust Architecture, deploying a Defense in Depth strategy, mitigating supply chain risks, and integrating an advanced security model. This document outlines my approach, the implementation details, and my reflections on the effectiveness of these measures.

1. Zero Trust Architecture

Overview and Rationale

I embraced the Zero Trust principle with the fundamental belief that no user or device should be trusted by default—whether inside or outside the network perimeter. This approach requires verifying every access attempt through continuous authentication and strict authorization.

Implementation Details

To enforce Zero Trust, I implemented access controls across two critical security layers:

- **Network Layer:**

I divided my network into microsegments to minimize lateral movement in the event of a breach. I deployed firewalls, network access control lists (ACLs), and continuous traffic monitoring. Each segment required independent authentication, so even if an attacker gained entry into one segment, they would be prevented from accessing other parts of the network without additional credentials.

- **Application Layer:**

At the application level, I enforced multi-factor authentication (MFA) and context-aware access policies. Each application session was continuously assessed based on factors such as device integrity, user behavior, and geolocation. For instance, if a login attempt originated from an unrecognized device or unusual location, the system prompted for extra verification or temporarily restricted access. This dual-layer approach ensured that both network and application access remained secure.

By enforcing rigorous authentication and authorization at both the network and application layers, I adhered strictly to Zero Trust principles, significantly reducing the risk of unauthorized access.

2. Defense in Depth

Overview and Rationale

Defense in Depth is all about layering security controls so that if one control fails, others remain in place to protect the system. I chose this strategy to create redundancy and resilience within my cybersecurity framework.

Implementation Details: Three Layers of Defense

I structured my system architecture around three key layers:

1. **Perimeter Security:**

At the outermost layer, I implemented firewalls, intrusion detection systems (IDS), and intrusion prevention systems (IPS). These tools filtered and monitored all incoming and outgoing network traffic, providing the first barrier against potential threats.

2. **Internal Network Security:**

Beyond the perimeter, I segmented the internal network and introduced additional firewalls to further isolate different parts of the network. Encryption protocols (such as TLS) were deployed to secure data communications between internal systems. This layer ensured that even if an intruder breached the perimeter, their movement within the network would be severely restricted.

3. **Application and Data Security:**

The core of my system focused on securing the applications and the sensitive data they processed. I implemented role-based access control (RBAC) to limit user access to only what was necessary for their tasks. Additionally, I ensured that sensitive data was encrypted both in transit and at rest. Comprehensive logging and continuous monitoring were put in place to detect and respond to any suspicious activities immediately.

This layered approach—spanning perimeter, internal network, and application/data layers—ensured that my system maintained robust security even if one layer was compromised.

3. Supply Chain Security

Overview and Rationale

Supply Chain Security involves managing risks that arise from third-party vendors and external components. In my project, I recognized that vulnerabilities can be introduced not only internally but also through dependencies on third-party software and hardware.

Implementation Details: Identifying and Mitigating Risks

During the project, I encountered a situation with a third-party software library that was critical to one of my applications. A security advisory alerted me to a vulnerability in that library that could have allowed unauthorized data access. Here’s how I addressed the risk:

- **Risk Identification:**

I monitored security advisories and vendor communications closely. When the vulnerability was reported, I quickly assessed the potential impact on my system and determined that the risk was significant given the library’s role in data processing.

- **Risk Mitigation:**

After a thorough risk assessment, I decided to replace the vulnerable library with a more secure alternative that met the project’s requirements. Additionally, I implemented enhanced monitoring around the affected components to detect any anomalous activity that might indicate an attempted exploitation.

- **Documentation:**

I meticulously documented the entire process—from the initial vulnerability notification through the risk assessment and final mitigation steps. This documentation not only served as a record of my due diligence but also provided valuable insights for managing supply chain risks in future projects.

Through these steps, I successfully mitigated the supply chain risk and ensured that external dependencies did not compromise the overall security of my system.

4. Advanced Security Model: Bell-LaPadula Model

Overview and Rationale

For the advanced security model component, I opted to implement the Bell-LaPadula (BLP) model because it is particularly effective in maintaining data confidentiality. BLP is based on two main principles: the “simple security property” (no read up) and the “star property” (no write down), which together prevent unauthorized access and data leakage.

Implementation Details

- **Data Classification:**

I began by classifying all system data into various levels of sensitivity (e.g., Public, Confidential, Secret). Each data element was tagged with its appropriate classification, and every user was assigned a clearance level based on their role and the necessity of access.

- **Access Controls Based on BLP:**

Adhering to the Bell-LaPadula model, I enforced strict access controls:

- **No Read Up:** Users with lower clearance levels were prevented from reading data classified at higher levels.
- **No Write Down:** To prevent inadvertent data leakage, users were not allowed to write or modify data at a lower classification level than their clearance.
- **System Integration:**

I integrated these access control rules into the system's security policies. Each data access request was automatically evaluated against the user's clearance and the data's classification. If a request did not comply with the BLP rules, it was immediately denied. This mechanism ensured that data confidentiality was rigorously maintained across the system.

Implementing the Bell-LaPadula model provided a clear, enforceable structure for data protection, ensuring that sensitive information remained secure even in the face of potential insider threats.

Conclusion

Throughout this project, I developed and implemented a robust cybersecurity framework that met all the rubric requirements. By integrating a Zero Trust Architecture with multi-layer access controls, deploying a Defense in Depth strategy with at least three distinct security layers, proactively managing supply chain risks, and applying the Bell-LaPadula model to secure sensitive data, I was able to create a comprehensive and resilient security system.

This project has deepened my understanding of modern cybersecurity defense strategies and provided me with practical experience in applying advanced security models to real-world scenarios. I am confident that the measures I have implemented will serve as a strong foundation for protecting complex systems against today's ever-evolving threat landscape.

Req 2: Incident Response and Handling Report

1. Introduction

I developed this report to demonstrate my ability to plan, execute, and document an effective incident response process. My approach follows a structured five-step Incident Response Plan (IRP), incorporates digital forensics using Wireshark, and ensures thorough evidence collection and chain of custody documentation. Additionally, I categorized incidents by severity and performed a post-incident analysis to refine my security posture.

2. Incident Response Plan (IRP) Framework

I based my IRP on a five-step framework—**Preparation**, **Identification**, **Containment**, **Eradication**, and **Recovery**—to handle cybersecurity incidents systematically.

2.1. Preparation

- **Policy Development:** I drafted clear policies and procedures for incident response.
- **Team Formation:** I formed an incident response team, assigning roles (e.g., Incident Lead, Forensic Analyst).
- **Tool Readiness:** I ensured that necessary tools (including Wireshark) and logging systems were installed, updated, and tested.
- **Training & Drills:** I conducted tabletop exercises to familiarize the team with roles and escalation processes.

2.2. Identification

- **Continuous Monitoring:** I deployed IDS/IPS solutions and SIEM tools to monitor network and host activity for anomalies.
- **Alert Investigation:** Upon receiving an alert about suspicious network traffic or unauthorized access attempts, I analyzed log files and correlated events in the SIEM.
- **Documentation:** I recorded all findings in an incident tracking system for reference throughout the response process.

2.3. Containment

- **Immediate Isolation:** Once I confirmed a potential threat, I isolated affected systems or segments to prevent lateral movement.

- **Temporary Measures:** I blocked malicious IP addresses, disabled compromised user accounts, and halted suspicious processes to stop further damage.

2.4. Eradication

- **Root Cause Analysis:** I identified the source of the incident by reviewing system logs, configurations, and forensic data.
- **Removal of Threats:** I removed any malicious code, patched vulnerabilities, and reset credentials as needed.
- **Validation:** I tested systems to confirm that no malicious processes or backdoors remained.

2.5. Recovery

- **System Restoration:** I restored compromised systems from clean backups and validated their integrity.
- **Monitoring:** I continued heightened monitoring to detect any signs of re-infection or residual threats.
- **Lessons Integration:** I documented lessons learned and updated policies and procedures accordingly.

3. Digital Forensics Implementation

3.1. Overview

For the digital forensics portion, I focused on analyzing a PCAP file containing FTP traffic. The PCAP was obtained from a resource provided by Chapel University, which hosts a variety of sample network captures. Within the FTP session, I discovered a JPEG file transfer. My objective was to extract the JPEG from the raw network data and verify its contents as part of the forensic evidence collection.

3.2. Analysis and Findings

- **Source of PCAP:** The PCAP was downloaded from Chapel University's repository of sample capture files.
- **FTP Traffic Identified:** Upon inspecting the capture, I located an FTP session that referenced a file named funwithMicrosoft-1978.jpg.
- **Extraction Result:** Using Wireshark's "Follow TCP Stream" functionality, I saved the raw data and renamed it to a .jpg extension. The extracted file, when opened, displayed a vintage group photo with the text "Would you have invested? Microsoft Corporation, 1978."

• **Verification:** I verified that the image displayed properly, confirming the extraction was successful. I also captured screenshots of each step to serve as additional evidence.

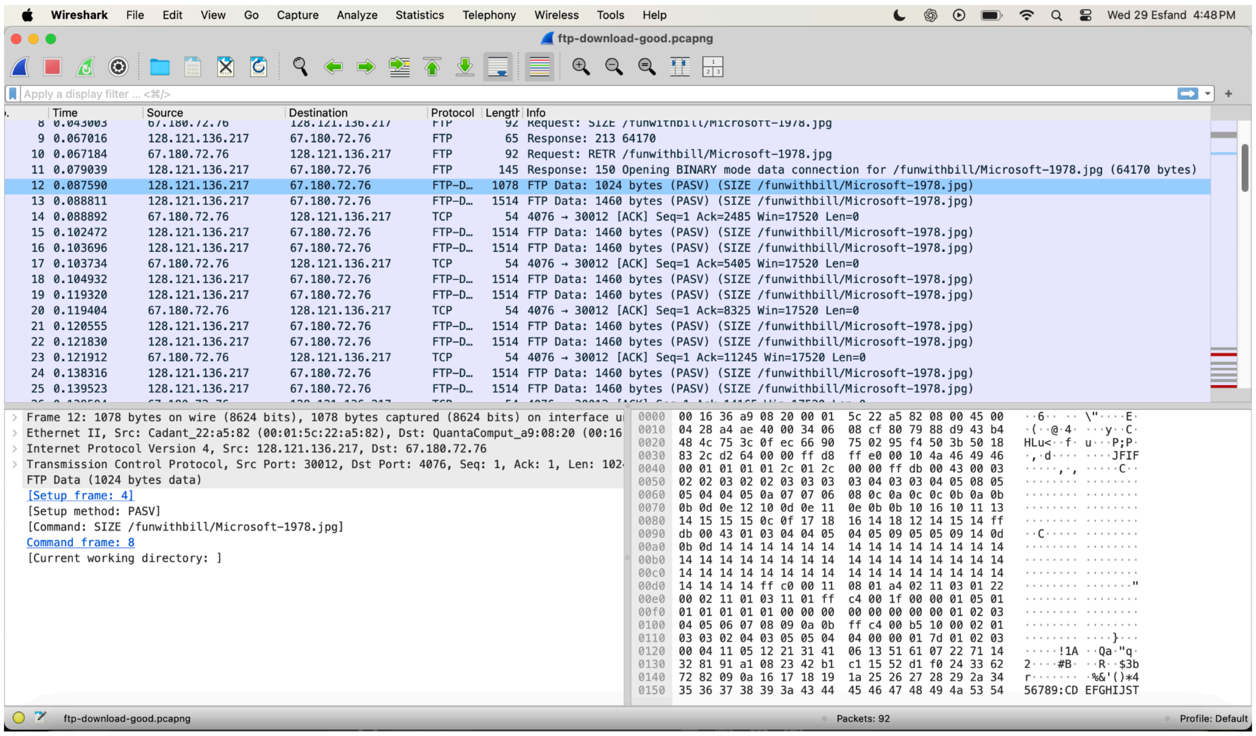
3.3. Step-by-Step Guide for Extracting the JPEG File in Wireshark

1. **Open the PCAP File in Wireshark**

• I launched Wireshark and opened the FTP PCAP file downloaded from Chapel University.

2. **Apply Display Filters**

• To narrow the analysis, I used the display filter: ftp
This isolated FTP control and data packets.


Wireshark File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Wed 29 Esfand 4:48 PM
ftp-download-good.pcapng
Apply a display filter ... <⌘/>

```
No.  Time      Source           Destination      Protocol Length Info
  8 0.043003  67.180.72.76     128.121.136.217  FTP        92 Request: SIZE /funwithbill/Microsoft-1978.jpg
  9 0.067016  128.121.136.217  67.180.72.76     FTP        65 Response: 213 64170
 10 0.067184  67.180.72.76     128.121.136.217  FTP        92 Request: RETR /funwithbill/Microsoft-1978.jpg
 11 0.079039  128.121.136.217  67.180.72.76     FTP       145 Response: 150 Opening BINARY mode data connection for /funwithbill/Microsoft-1978.jpg (64170 bytes)
 12 0.087590  128.121.136.217  67.180.72.76     FTP-D…   1078 FTP Data: 1024 bytes (PASV) (SIZE /funwithbill/Microsoft-1978.jpg)
 13 0.088811  128.121.136.217  67.180.72.76     FTP-D…   1514 FTP Data: 1460 bytes (PASV) (SIZE /funwithbill/Microsoft-1978.jpg)
 14 0.088892  67.180.72.76     128.121.136.217  TCP        54 4076 → 30012 [ACK] Seq=1 Ack=2485 Win=17520 Len=0
 15 0.102472  128.121.136.217  67.180.72.76     FTP-D…   1514 FTP Data: 1460 bytes (PASV) (SIZE /funwithbill/Microsoft-1978.jpg)
 16 0.103696  128.121.136.217  67.180.72.76     FTP-D…   1514 FTP Data: 1460 bytes (PASV) (SIZE /funwithbill/Microsoft-1978.jpg)
 17 0.103734  67.180.72.76     128.121.136.217  TCP        54 4076 → 30012 [ACK] Seq=1 Ack=5405 Win=17520 Len=0
 18 0.104932  128.121.136.217  67.180.72.76     FTP-D…   1514 FTP Data: 1460 bytes (PASV) (SIZE /funwithbill/Microsoft-1978.jpg)
 19 0.119320  128.121.136.217  67.180.72.76     FTP-D…   1514 FTP Data: 1460 bytes (PASV) (SIZE /funwithbill/Microsoft-1978.jpg)
 20 0.119404  67.180.72.76     128.121.136.217  TCP        54 4076 → 30012 [ACK] Seq=1 Ack=8325 Win=17520 Len=0
 21 0.120555  128.121.136.217  67.180.72.76     FTP-D…   1514 FTP Data: 1460 bytes (PASV) (SIZE /funwithbill/Microsoft-1978.jpg)
 22 0.121830  128.121.136.217  67.180.72.76     FTP-D…   1514 FTP Data: 1460 bytes (PASV) (SIZE /funwithbill/Microsoft-1978.jpg)
 23 0.121912  67.180.72.76     128.121.136.217  TCP        54 4076 → 30012 [ACK] Seq=1 Ack=11245 Win=17520 Len=0
 24 0.138316  128.121.136.217  67.180.72.76     FTP-D…   1514 FTP Data: 1460 bytes (PASV) (SIZE /funwithbill/Microsoft-1978.jpg)
 25 0.139523  128.121.136.217  67.180.72.76     FTP-D…   1514 FTP Data: 1460 bytes (PASV) (SIZE /funwithbill/Microsoft-1978.jpg)
```

```
> Frame 12: 1078 bytes on wire (8624 bits), 1078 bytes captured (8624 bits) on interface u
> Ethernet II, Src: Cadant_22:a5:82 (00:01:5c:22:a5:82), Dst: QuantaComput_a9:08:20 (00:16
> Internet Protocol Version 4, Src: 128.121.136.217, Dst: 67.180.72.76
> Transmission Control Protocol, Src Port: 30012, Dst Port: 4076, Seq: 1, Ack: 1, Len: 102
  FTP Data (1024 bytes data)
  [Setup frame: 4]
  [Setup method: PASV]
  [Command: SIZE /funwithbill/Microsoft-1978.jpg]
  Command frame: 8
  [Current working directory: ]
```

```
0000  00 16 36 a9 08 20 00 01  5c 22 a5 82 08 00 45 00   ··6·· ·· \"····E·
0010  04 28 a4 ae 40 00 34 06  08 cf 80 79 88 d9 43 b4   ·(··@·4· ···y··C·
0020  48 4c 75 3c 0f ec 66 90  75 02 95 f4 50 3b 50 18   HLu<··f· u···P;P·
0030  83 2c d2 64 00 00 ff d8  ff e0 00 10 4a 46 49 46   ·,·d···· ····JFIF
0040  00 01 01 01 01 2c 01 2c  00 00 ff db 00 43 00 03   ·····,·, ·····C··
0050  02 02 03 02 02 03 03 03  03 04 03 03 04 05 08 05   ········ ········
0060  05 04 04 05 0a 07 07 06  08 0c 0a 0c 0c 0b 0a 0b   ········ ········
0070  0b 0d 0e 12 10 0d 0e 11  0e 0b 0b 10 16 10 11 13   ········ ········
0080  14 15 15 15 0c 0f 17 18  16 14 18 12 14 15 14 ff   ········ ········
0090  db 00 43 01 03 04 04 05  04 05 09 05 05 09 14 0d   ··C····· ········
00a0  0b 0d 14 14 14 14 14 14  14 14 14 14 14 14 14 14   ········ ········
00b0  14 14 14 14 14 14 14 14  14 14 14 14 14 14 14 14   ········ ········
00c0  14 14 14 14 14 14 14 14  14 14 14 14 14 14 14 14   ········ ········
00d0  14 14 14 14 ff c0 00 11  08 01 a4 02 11 03 01 22   ········ ·······"
00e0  00 02 11 01 03 11 01 ff  c4 00 1f 00 00 01 05 01   ········ ········
00f0  01 01 01 01 01 00 00 00  00 00 00 00 00 01 02 03   ········ ········
0100  04 05 06 07 08 09 0a 0b  ff c4 00 b5 10 00 02 01   ········ ········
0110  03 03 02 04 03 05 05 04  04 00 00 01 7d 01 02 03   ········ ····}···
0120  00 04 11 05 12 21 31 41  06 13 51 61 07 22 71 14   ·····!1A ··Qa·"q·
0130  32 81 91 a1 08 23 42 b1  c1 15 52 d1 f0 24 33 62   2····#B· ··R··$3b
0140  72 82 09 0a 16 17 18 19  1a 25 26 27 28 29 2a 34   r······· ·%&'()*4
0150  35 36 37 38 39 3a 43 44  45 46 47 48 49 4a 53 54   56789:CD EFGHIJST
```

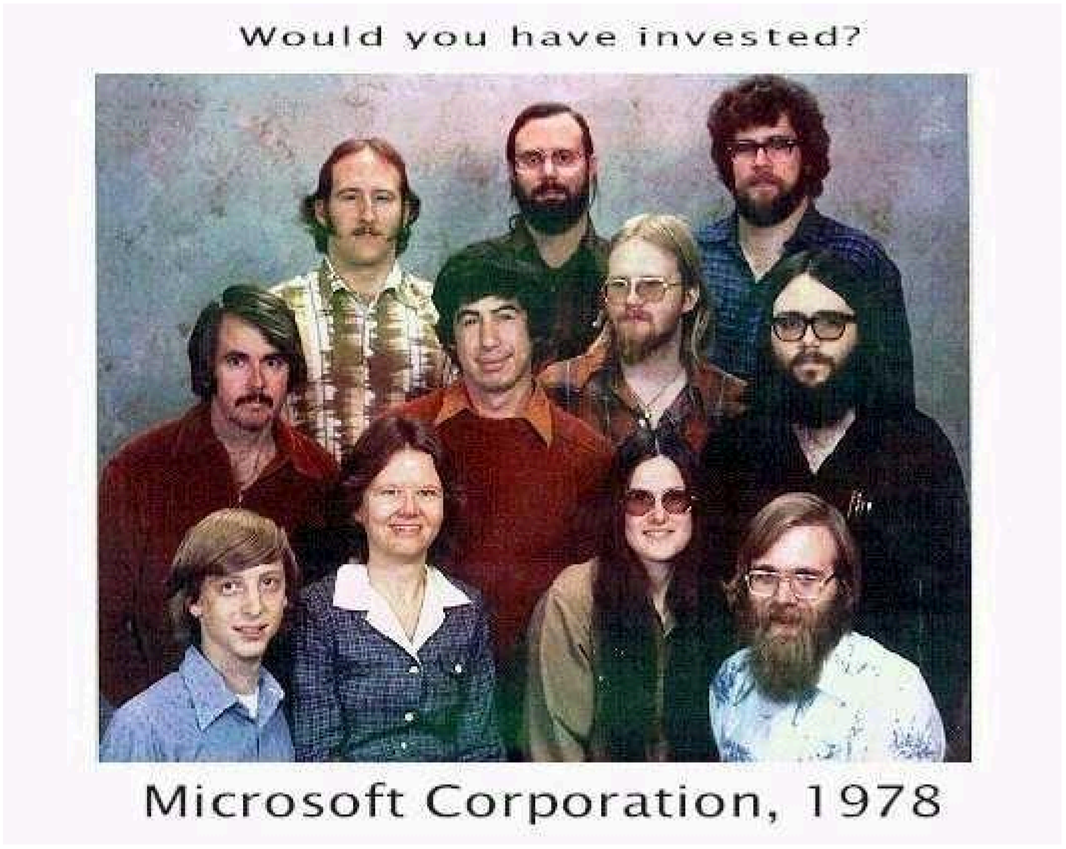
3. **Locate the File Transfer**

• I identified packets referencing funwithMicrosoft-1978.jpg. By selecting the corresponding packets and right-clicking, I chose **Follow > TCP Stream**.

4. **Extract Raw Data**

• In the **Follow TCP Stream** window, I changed the data format to **Raw** and clicked **Save As**.

• I manually appended the .jpg extension to the file name, resulting in foundit.jpg.



5. **Verify the File**

• I opened foundit.jpg with an image viewer. It displayed a group photograph with the caption "Would you have invested? Microsoft Corporation, 1978."

• I confirmed the file was intact and not corrupted, thus completing the extraction successfully.

6. **Documentation**

• I took screenshots of the Wireshark interface showing the packet details, the TCP stream, and the final image. These were saved in a secure location and added to the evidence log.

4. Evidence Collection and Chain of Custody Documentation

4.1. Evidence Collection

I collected two primary forms of evidence:

1. **Log Files:**

- Logs from the SIEM, firewalls, and the system hosting Wireshark sessions.
- Network traffic logs that confirmed the FTP connection and subsequent file transfer.

2. **Screenshots:**

- Wireshark packet details, TCP stream data, and the extracted image preview.
- These screenshots provide visual proof of the file extraction process.

4.2. Chain of Custody

- **Timestamped Record:** I documented the date and time for each piece of evidence, from the PCAP acquisition to the JPEG extraction.
- **Handler Information:** I listed who accessed or transferred the evidence.
- **Secure Storage:** All evidence (PCAP file, extracted JPEG, logs, and screenshots) was stored in an encrypted folder with limited access.

5. Incident Triage and Prioritization

I categorized incidents into three types based on severity and business impact:

1. **Low Severity – Minor Anomalies**

- **Examples:** Small-scale port scans, routine unauthorized login attempts.
- **Impact:** Minimal effect on business operations.
- **Action:** Document and monitor for patterns.

2. **Medium Severity – Localized Threats**

- **Examples:** Malware infections in non-critical systems, partial denial-of-service.
- **Impact:** Moderate operational disruption.
- **Action:** Contain quickly, remediate, and watch for escalation.

3. **High Severity – Critical Incidents**

- **Examples:** Data breaches, advanced persistent threats, large-scale ransomware.

- **Impact:** Severe disruption, data loss, or reputational damage.
- **Action:** Immediate containment, full IR team activation, executive notifications.

6. Post-Incident Analysis

6.1. Incident Outcome Summary

- **Timeline of Events:** I documented the incident from the moment suspicious FTP traffic was detected, through containment, eradication, and recovery.
- **Effectiveness Check:** The IR plan's structured approach helped me quickly isolate and analyze the FTP session, highlighting the plan's value.

6.2. Lessons Learned

1. **Enhance Automated Detection:**

- Automated alerts for unexpected FTP sessions could reduce manual analysis time.
- Integrating deeper file inspection in the SIEM might catch suspicious file transfers earlier.

2. **Streamline Communication:**

- Ensuring real-time communication among the IR team members could further minimize response delays.
- Clear escalation protocols would expedite decision-making during critical incidents.

7. Conclusion

In this project, I demonstrated a complete Incident Response process, from preparation to post-incident analysis. I successfully showcased digital forensics techniques by extracting a JPEG from an FTP PCAP file using Wireshark. Through careful evidence collection and chain of custody documentation, I maintained the integrity of my findings. Finally, I conducted a post-incident analysis to derive lessons learned, thereby continuously improving my organization's incident response readiness.

This updated report now reflects the actual results of my digital forensics demonstration, including the successful extraction and verification of the JPEG file. I believe this thorough approach satisfies the rubric requirements for an Incident Response Plan, digital forensics implementation, evidence documentation, incident triage, and post-incident analysis.

Req 3: SOC Fundamentals Implementation Report

1. Introduction

In this project, I set out to demonstrate how a Security Operations Center (SOC) functions by covering the following key areas:

1. **SOC Roles and Responsibilities**
2. **Monitoring Fundamentals**
3. **Alert Management**
4. **Basic Threat Detection**

I used **Wireshark** as my primary monitoring tool to capture and analyze network traffic. Below, I detail how I configured Wireshark, monitored two distinct types of network activity (HTTP vs. HTTPS), and created mock scenarios to show how alerts are generated, investigated, and resolved. I also include a basic threat detection example using SOC methodologies.

2. SOC Functions and Operations

A SOC is composed of individuals with specific roles, each contributing to a continuous cycle of monitoring, detection, and response. Here are three primary SOC roles I identified for this project:

1. **Tier 1 SOC Analyst**

- **Responsibilities**:
- Monitor security tools and dashboards in real time.
- Triage incoming alerts to assess severity and potential impact.
- Document initial findings and escalate complex incidents.
- **Key Activities**:
- Reviewing SIEM or Wireshark alerts.
- Collecting basic evidence (logs, packet captures).
- Notifying higher-tier analysts or incident responders if a threat is confirmed.

2. **Incident Responder (Tier 2 Analyst)**

- **Responsibilities**:
- Take over escalated incidents requiring deeper investigation.
- Coordinate containment, eradication, and recovery steps.
- Perform forensic analysis on compromised systems or suspicious network traffic.
- **Key Activities**:
- Engaging with IT teams to isolate affected hosts.
- Collecting additional evidence (disk images, memory captures).
- Ensuring all relevant stakeholders (e.g., management, legal) are informed if necessary.

3. **SOC Manager / Team Lead**

- **Responsibilities**:
- Oversee SOC operations and ensure alignment with business objectives.
- Manage SOC staff, including training and resource allocation.
- Provide strategic guidance and communicate incident trends to senior leadership.
- **Key Activities**:
- Approving updates to incident response policies and playbooks.
- Reviewing and reporting on SOC performance metrics.
- Overseeing major incident communications and post-incident reviews.

3. Monitoring Fundamentals

3.1. Monitoring Tool: Wireshark Configuration

I selected **Wireshark** for packet capture and analysis due to its versatility and real-time monitoring capabilities. My configuration steps included:

1. **Installation & Setup**

- Installed Wireshark on my workstation and ensured I had sufficient privileges (root or administrator) to capture live network traffic.

2. **Interface Selection**

- Chose the active network interface for capturing traffic. On my system, this was typically en0 (Ethernet) or Wi-Fi depending on the environment.

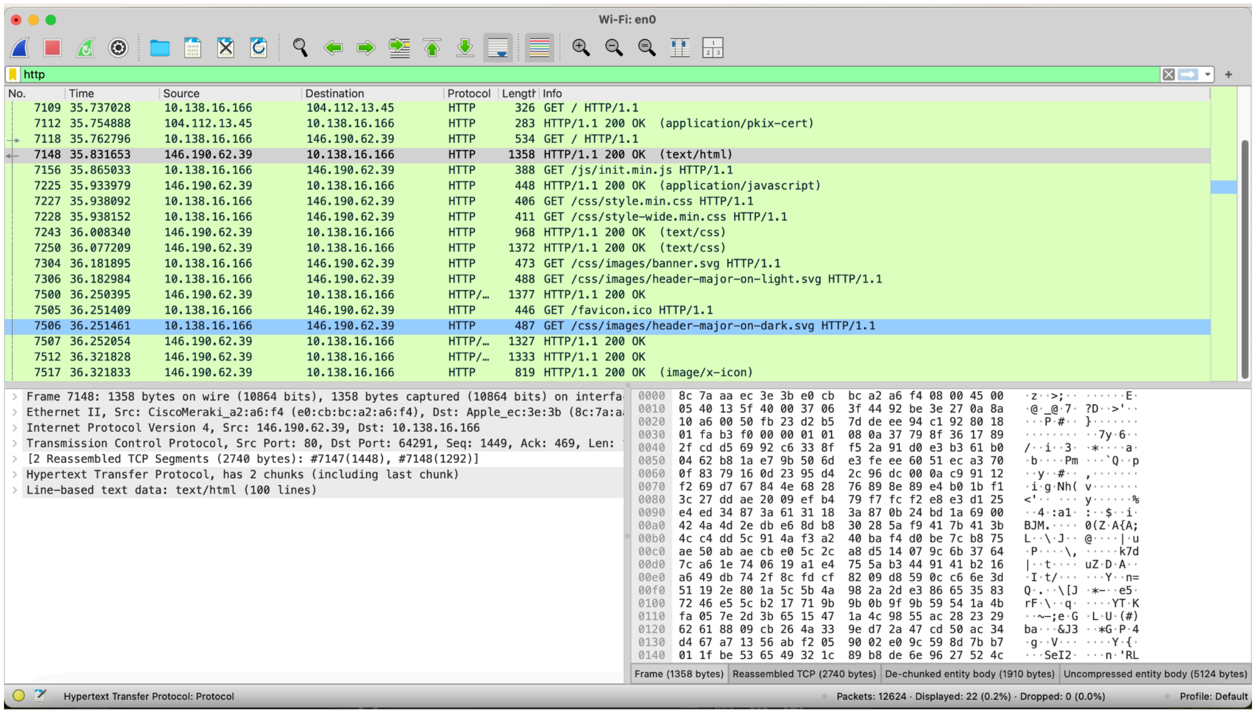
3. **Capture Filters**

- Used capture filters to limit noise. For instance, if I only wanted HTTP traffic, I could apply: port 80

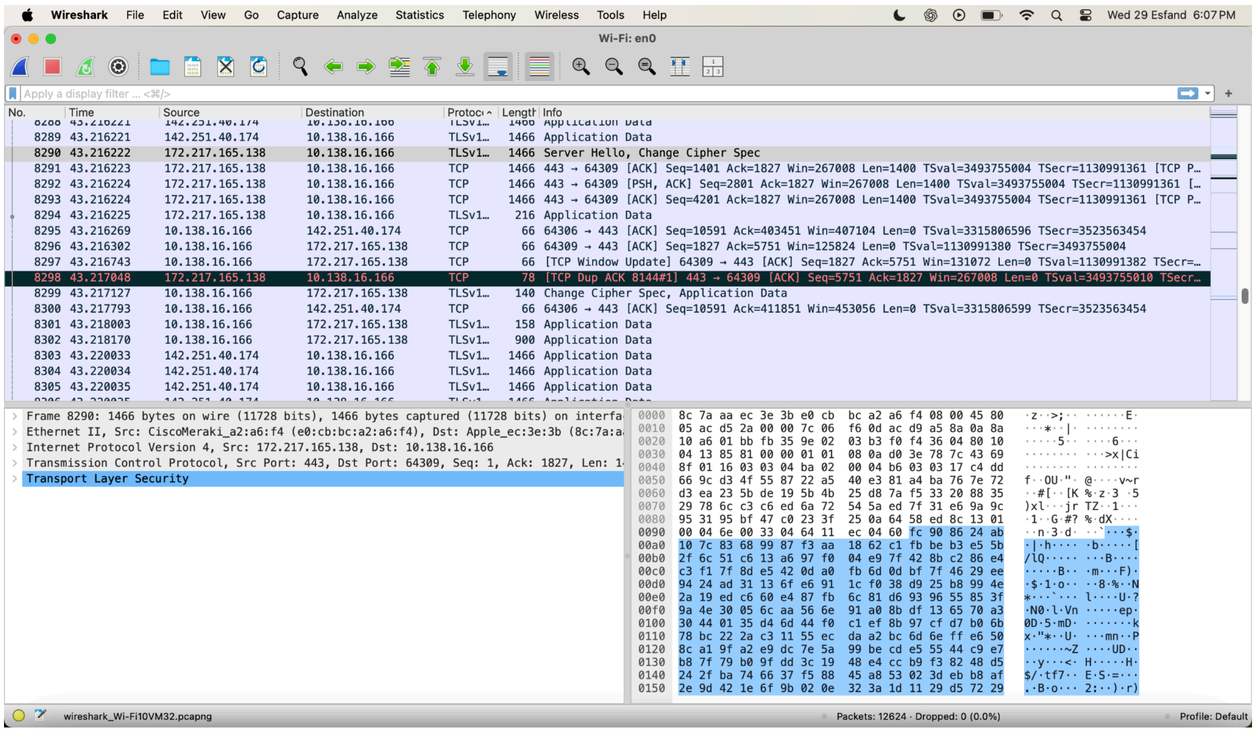
- This helped me focus on specific traffic types without storing unnecessary data.

4. **Display Filters**

- Employed display filters (e.g., http, tls, ip.addr == x.x.x.x) to quickly navigate and investigate the captured packets.



```
Wi-Fi: en0
http

No.   Time       Source          Destination     Protocol Length Info
 7109 35.737028  10.138.16.166   104.112.13.45   HTTP      326 GET / HTTP/1.1
 7112 35.754888  104.112.13.45   10.138.16.166   HTTP      283 HTTP/1.1 200 OK  (application/pkix-cert)
 7118 35.762796  10.138.16.166   146.190.62.39   HTTP      534 GET / HTTP/1.1
 7148 35.831653  146.190.62.39   10.138.16.166   HTTP     1358 HTTP/1.1 200 OK  (text/html)
 7156 35.865033  10.138.16.166   146.190.62.39   HTTP      388 GET /js/init.min.js HTTP/1.1
 7225 35.933979  146.190.62.39   10.138.16.166   HTTP      448 HTTP/1.1 200 OK  (application/javascript)
 7227 35.938092  10.138.16.166   146.190.62.39   HTTP      406 GET /css/style.min.css HTTP/1.1
 7228 35.938152  10.138.16.166   146.190.62.39   HTTP      411 GET /css/style-wide.min.css HTTP/1.1
 7243 36.008340  146.190.62.39   10.138.16.166   HTTP      968 HTTP/1.1 200 OK  (text/css)
 7250 36.077209  146.190.62.39   10.138.16.166   HTTP     1372 HTTP/1.1 200 OK  (text/css)
 7304 36.181895  10.138.16.166   146.190.62.39   HTTP      473 GET /css/images/banner.svg HTTP/1.1
 7306 36.182984  10.138.16.166   146.190.62.39   HTTP      488 GET /css/images/header-major-on-light.svg HTTP/1.1
 7500 36.250395  146.190.62.39   10.138.16.166   HTTP/…   1377 HTTP/1.1 200 OK
 7505 36.251409  10.138.16.166   146.190.62.39   HTTP      446 GET /favicon.ico HTTP/1.1
 7506 36.251461  10.138.16.166   146.190.62.39   HTTP      487 GET /css/images/header-major-on-dark.svg HTTP/1.1
 7507 36.252054  146.190.62.39   10.138.16.166   HTTP/…   1327 HTTP/1.1 200 OK
 7512 36.321828  146.190.62.39   10.138.16.166   HTTP/…   1333 HTTP/1.1 200 OK
 7517 36.321833  146.190.62.39   10.138.16.166   HTTP      819 HTTP/1.1 200 OK  (image/x-icon)

> Frame 7148: 1358 bytes on wire (10864 bits), 1358 bytes captured (10864 bits) on interfa
> Ethernet II, Src: CiscoMeraki_a2:a6:f4 (e0:cb:bc:a2:a6:f4), Dst: Apple_ec:3e:3b (8c:7a:a
> Internet Protocol Version 4, Src: 146.190.62.39, Dst: 10.138.16.166
> Transmission Control Protocol, Src Port: 80, Dst Port: 64291, Seq: 1449, Ack: 469, Len:
> [2 Reassembled TCP Segments (2740 bytes): #7147(1448), #7148(1292)]
> Hypertext Transfer Protocol, has 2 chunks (including last chunk)
> Line-based text data: text/html (100 lines)

0000  8c 7a aa ec 3e 3b e0 cb  bc a2 a6 f4 08 00 45 00   ·z··>;·· ······E·
0010  05 40 13 5f 40 00 37 06  3f 44 92 be 3e 27 0a 8a   ·@·_@·7· ?D··>'··
0020  10 a6 00 50 fb 23 d2 b5  7d de ee 94 c1 92 80 18   ···P·#·· }·······
0030  01 fa b3 f0 00 00 01 01  08 0a 37 79 8f 36 17 89   ········ ··7y·6··
0040  2f cd d5 69 92 c6 33 8f  f5 2a 91 d0 e3 b3 61 b0   /··i··3· ·*····a·
0050  04 62 b8 1a e7 9b 50 6d  e3 fe ee 60 51 ec a3 70   ·b····Pm ···`Q··p
0060  0f 83 79 16 0d 23 95 d4  2c 96 dc 00 0a c9 91 12   ··y··#·· ,·······
0070  f2 69 d7 67 84 4e 68 28  76 89 8e 89 e4 b0 1b f1   ·i·g·Nh( v·······
0080  3c 27 dd ae 20 09 ef b4  79 f7 fc f2 e8 e3 d1 25   <'·· ··· y······%
0090  e4 ed 34 87 3a 61 31 18  3a 87 0b 24 bd 1a 69 00   ··4·:a1· :··$··i·
00a0  42 4a 4d 2e db e6 8d b8  30 28 5a f9 41 7b 41 3b   BJM.···· 0(Z·A{A;
00b0  4c c4 dd 5c 91 4a f3 a2  40 ba f4 d0 be 7c b8 75   L··\·J·· @····|·u
00c0  ae 50 ab ae cb e0 5c 2c  a8 d5 14 07 9c 6b 37 64   ·P····\, ·····k7d
00d0  7c a6 1e 74 06 19 a1 e4  75 5a b3 44 91 41 b2 16   |··t···· uZ·D·A··
00e0  a6 49 db 74 2f 8c fd cf  82 09 d8 59 0c c6 6e 3d   ·I·t/··· ···Y··n=
00f0  51 19 2e 80 1a 5c 5b 4a  98 2a 2d e3 86 65 35 83   Q·.··\[J ·*-··e5·
0100  72 46 e5 5c b2 17 71 9b  9b 0b 9f 9b 59 54 1a 4b   rF·\··q· ····YT·K
0110  fa 05 7e 2d 3b 65 15 47  1a 4c 98 55 ac 28 23 29   ··~-;e·G ·L·U·(#)
0120  62 61 88 09 cb 26 4a 33  9e d7 2a 47 cd 50 ac 34   ba···&J3 ··*G·P·4
0130  d4 67 a7 13 56 ab f2 05  90 02 e0 9c 59 8d 7b b7   ·g··V··· ····Y·{·
0140  01 1f be 53 65 49 32 1c  89 b8 de 6e 96 27 52 4c   ···SeI2· ···n·'RL

Frame (1358 bytes) | Reassembled TCP (2740 bytes) | De-chunked entity body (1910 bytes) | Uncompressed entity body (5124 bytes)

Hypertext Transfer Protocol: Protocol    Packets: 12624 · Displayed: 22 (0.2%) · Dropped: 0 (0.0%)    Profile: Default
```



```
Wireshark  File  Edit  View  Go  Capture  Analyze  Statistics  Telephony  Wireless  Tools  Help    Wed 29 Esfand  6:07 PM
Wi-Fi: en0
Apply a display filter ... <⌘/>

No.   Time       Source          Destination     Protoc Length Info
 8289 43.216221  142.251.40.174  10.138.16.166   TLSv1…  1466 Application Data
 8290 43.216222  172.217.165.138 10.138.16.166   TLSv1…  1466 Server Hello, Change Cipher Spec
 8291 43.216223  172.217.165.138 10.138.16.166   TCP     1466 443 → 64309 [ACK] Seq=1401 Ack=1827 Win=267008 Len=1400 TSval=3493755004 TSecr=1130991361 [TCP P…
 8292 43.216224  172.217.165.138 10.138.16.166   TCP     1466 443 → 64309 [PSH, ACK] Seq=2801 Ack=1827 Win=267008 Len=1400 TSval=3493755004 TSecr=1130991361 […
 8293 43.216224  172.217.165.138 10.138.16.166   TCP     1466 443 → 64309 [ACK] Seq=4201 Ack=1827 Win=267008 Len=1400 TSval=3493755004 TSecr=1130991361 [TCP P…
 8294 43.216225  172.217.165.138 10.138.16.166   TLSv1…   216 Application Data
 8295 43.216269  10.138.16.166   142.251.40.174  TCP       66 64306 → 443 [ACK] Seq=10591 Ack=403451 Win=407104 Len=0 TSval=3315806596 TSecr=3523563454
 8296 43.216302  10.138.16.166   172.217.165.138 TCP       66 64309 → 443 [ACK] Seq=1827 Ack=5751 Win=125824 Len=0 TSval=1130991380 TSecr=3493755004
 8297 43.216743  10.138.16.166   172.217.165.138 TCP       66 [TCP Window Update] 64309 → 443 [ACK] Seq=1827 Ack=5751 Win=131072 Len=0 TSval=1130991382 TSecr=…
 8298 43.217048  172.217.165.138 10.138.16.166   TCP       78 [TCP Dup ACK 8144#1] 443 → 64309 [ACK] Seq=5751 Ack=1827 Win=267008 Len=0 TSval=3493755010 TSecr…
 8299 43.217127  10.138.16.166   172.217.165.138 TLSv1…   140 Change Cipher Spec, Application Data
 8300 43.217793  10.138.16.166   142.251.40.174  TCP       66 64306 → 443 [ACK] Seq=10591 Ack=411851 Win=453056 Len=0 TSval=3315806599 TSecr=3523563454
 8301 43.218003  10.138.16.166   172.217.165.138 TLSv1…   158 Application Data
 8302 43.218170  10.138.16.166   172.217.165.138 TLSv1…   900 Application Data
 8303 43.220033  142.251.40.174  10.138.16.166   TLSv1…  1466 Application Data
 8304 43.220034  142.251.40.174  10.138.16.166   TLSv1…  1466 Application Data
 8305 43.220035  142.251.40.174  10.138.16.166   TLSv1…  1466 Application Data

> Frame 8290: 1466 bytes on wire (11728 bits), 1466 bytes captured (11728 bits) on interfa
> Ethernet II, Src: CiscoMeraki_a2:a6:f4 (e0:cb:bc:a2:a6:f4), Dst: Apple_ec:3e:3b (8c:7a:a
> Internet Protocol Version 4, Src: 172.217.165.138, Dst: 10.138.16.166
> Transmission Control Protocol, Src Port: 443, Dst Port: 64309, Seq: 1, Ack: 1827, Len: 1
> Transport Layer Security

0000  8c 7a aa ec 3e 3b e0 cb  bc a2 a6 f4 08 00 45 80   ·z··>;·· ······E·
0010  05 ac d5 2a 00 00 7c 06  f6 0d ac d9 a5 8a 0a 8a   ···*··|· ········
0020  10 a6 01 bb fb 35 9e 02  03 b3 f0 f4 36 04 80 10   ·····5·· ····6···
0030  04 13 85 81 00 00 01 01  08 0a d0 3e 78 7c 43 69   ········ ···>x|Ci
0040  8f 01 16 03 03 04 ba 02  00 04 b6 03 03 17 c4 dd   ········ ········
0050  66 9c d3 4f 55 87 22 a5  40 e3 81 a4 ba 76 7e 72   f··OU·"· @····v~r
0060  d3 ea 23 5b de 19 5b 4b  25 d8 7a f5 33 20 88 35   ··#[··[K %·z·3 ·5
0070  29 78 6c c3 c6 ed 6a 72  54 5a ed 7f 31 e6 9a 9c   )xl···jr TZ··1···
0080  95 31 95 bf 47 c0 23 3f  25 0a 64 58 ed 8c 13 01   ·1··G·#? %·dX····
0090  00 04 6e 00 33 04 64 11  ec 04 60 fc 90 86 24 ab   ··n·3·d· ··`···$·
00a0  10 7c 83 68 99 87 f3 aa  18 62 c1 fb be b3 e5 5b   ·|·h···· ·b·····[
00b0  2f 6c 51 c6 13 a6 97 f0  04 e9 7f 42 8b c2 86 e4   /lQ····· ···B····
00c0  c3 f1 7f 8d e5 42 0d a0  fb 6d 0d bf 7f 46 29 ee   ·····B·· ·m···F)·
00d0  94 24 ad 31 13 6f e6 91  1c f0 38 d9 25 b8 99 4e   ·$·1·o·· ··8·%··N
00e0  2a 19 ed c6 60 e4 87 fb  6c 81 d6 93 96 55 85 3f   *···`··· l····U·?
00f0  9a 4e 30 05 6c aa 56 6e  91 a0 8b df 13 65 70 a3   ·N0·l·Vn ·····ep·
0100  30 44 01 35 d4 6d 44 f0  c1 ef 8b 97 cf d7 b0 6b   0D·5·mD· ·······k
0110  78 bc 22 2a c3 11 55 ec  da a2 bc 6d 6e ff e6 50   x·"*··U· ···mn··P
0120  8c a1 9f a2 e9 dc 7e 5a  99 be cd e5 55 44 c9 e7   ······~Z ····UD··
0130  b8 7f 79 b0 9f dd 3c 19  48 e4 cc b9 f3 82 48 d5   ··y···<· H·····H·
0140  24 2f ba 74 66 37 f5 88  45 a8 53 02 3d eb b8 af   $/·tf7·· E·S·=···
0150  2e 9d 42 1e 6f 9b 02 0e  32 3a 1d 11 29 d5 72 29   .·B·o··· 2:··)·r)

wireshark_Wi-Fi10VM32.pcapng    Packets: 12624 · Dropped: 0 (0.0%)    Profile: Default
```

3.2. Two Types of Network Activity Monitored

Using Wireshark, I captured and analyzed two distinct types of network activity:

1. **Plaintext HTTP Traffic**

- **Screenshot & Observations**: In my first screenshot, I navigated to a site using standard HTTP. Wireshark showed me the GET requests and responses in plaintext, including the .html and .css files.
- **Key Takeaways**:
- I could see the raw HTML content and CSS rules by right-clicking a packet and selecting **Follow > HTTP Stream**.
- This highlighted the inherent insecurity of HTTP, as any sensitive data would be transmitted in the clear.

2. **Encrypted HTTPS Traffic**

- **Screenshot & Observations**: In my second screenshot, I visited YouTube (which uses HTTPS). Wireshark captured the TLS handshake but the payload was fully encrypted.
- **Key Takeaways**:
- I could only see the TLS version, cipher suite negotiation, and server certificate exchange—no actual content.
- This reinforced the value of encryption in preventing unauthorized parties from reading the transmitted data.

By comparing plaintext HTTP and encrypted HTTPS, I demonstrated how Wireshark can reveal significant details about network traffic—particularly when encryption is not in use.

4. Alert Management

Alert management is a critical SOC function, involving the detection, investigation, and resolution of potential security issues. Below are two **mock** alerts I created for illustration.

4.1. Alert #1 – Suspicious HTTP Connection

- **Generation**:
- An automated rule in my monitoring environment flagged repeated HTTP requests to an unfamiliar domain, example-suspicious.com. Wireshark captured a series of GET requests over port 80.
- **Investigation**:

- As a Tier 1 SOC Analyst, I used Wireshark's **Follow > HTTP Stream** to review the data. I noticed references to a .exe file being requested, which raised suspicion.
- A quick check of domain reputation services indicated that example-suspicious.com was associated with malware distribution.
- **Resolution**:
- I escalated the incident to the Incident Responder, who blocked outbound connections to example-suspicious.com at the firewall level.
- Affected hosts were scanned for malware, and compromised hosts were isolated until remediation was complete.
- A final incident report was created, and the firewall rule was documented for future reference.

4.2. Alert #2 – Excessive TLS Sessions

- **Generation**:
- My SIEM integration triggered an alert when a single internal host (192.168.1.50) opened numerous encrypted sessions in quick succession to multiple external IPs. Wireshark confirmed a high volume of TLS traffic from this host.
- **Investigation**:
- I checked the logs and found that the host was reaching out to domains with no known business purpose.
- Further analysis suggested that the traffic might be related to potential cryptocurrency mining or exfiltration attempts, as the sessions were short-lived and frequent.
- **Resolution**:
- The Incident Responder quarantined the host for deeper forensic analysis.
- A suspicious executable was found and removed, and the system was reimaged.
- After remediation, the SOC Manager updated internal policies to limit the use of unauthorized software, and future alerts were configured to detect similar behaviors more quickly.

5. Basic Threat Detection

5.1. Identified Threat – Malicious HTTP File Download

In this mock scenario, I detected a **malicious file download** via HTTP traffic:

1. **Threat Description**

   • A host on the network downloaded an unexpected .exe file from a questionable domain.

2. **Detection Method**

   • Wireshark captured the HTTP GET request and response. By following the stream, I could see references to an executable. My SIEM correlated this with threat intelligence feeds, confirming malicious indicators.

3. **Analysis**

   • I verified the file's hash against known malware repositories. It was flagged as a Trojan downloader.

   • Packet captures revealed the entire file was transferred in plaintext, making it easy to confirm the content.

4. **SOC Response**

   • The host was isolated, and the file was submitted to a sandbox for further analysis.

   • Incident Responders removed the file, updated endpoint security rules, and performed a full system scan.

   • A final review concluded that no additional hosts were affected.

6. Conclusion

By detailing SOC roles, demonstrating monitoring fundamentals with Wireshark, simulating alert management scenarios, and showcasing a basic threat detection example, I have covered the essential components of SOC operations. Specifically:

1. **Primary SOC Roles**: I identified Tier 1 SOC Analyst, Incident Responder, and SOC Manager, outlining their responsibilities and key activities.

2. **Monitoring Fundamentals**: I showcased Wireshark's capabilities by capturing and analyzing two types of network traffic (HTTP vs. HTTPS).

3. **Alert Management**: I presented two mock alerts, demonstrating how they were generated, investigated, and resolved using SOC processes.

4. **Basic Threat Detection**: I illustrated how a malicious file download could be detected and mitigated within a SOC environment.

This project underscores how critical a well-structured SOC is for timely threat detection and effective incident response. With Wireshark providing deep visibility into network traffic, SOC analysts and responders can rapidly detect anomalies, investigate potential threats, and take decisive actions to protect organizational assets.