

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.

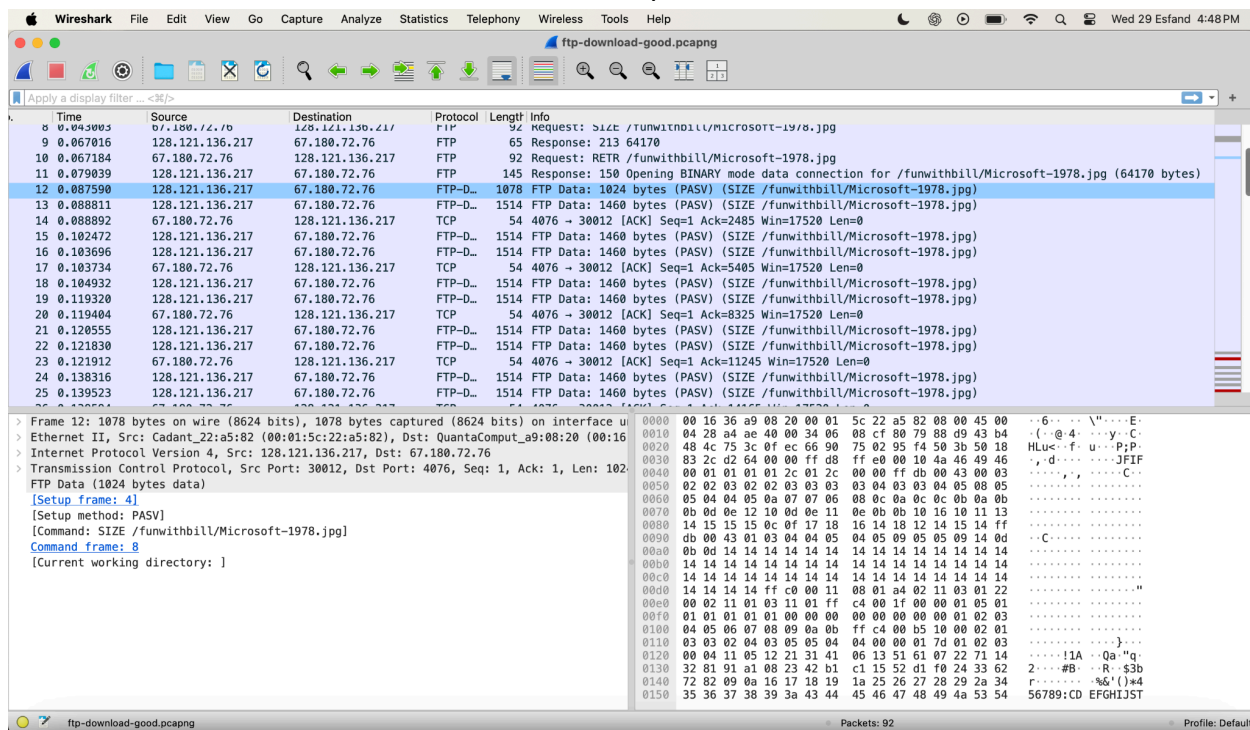
3.3. Step-by-Step Guide for Extracting the JPEG File in Wireshark

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.



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.

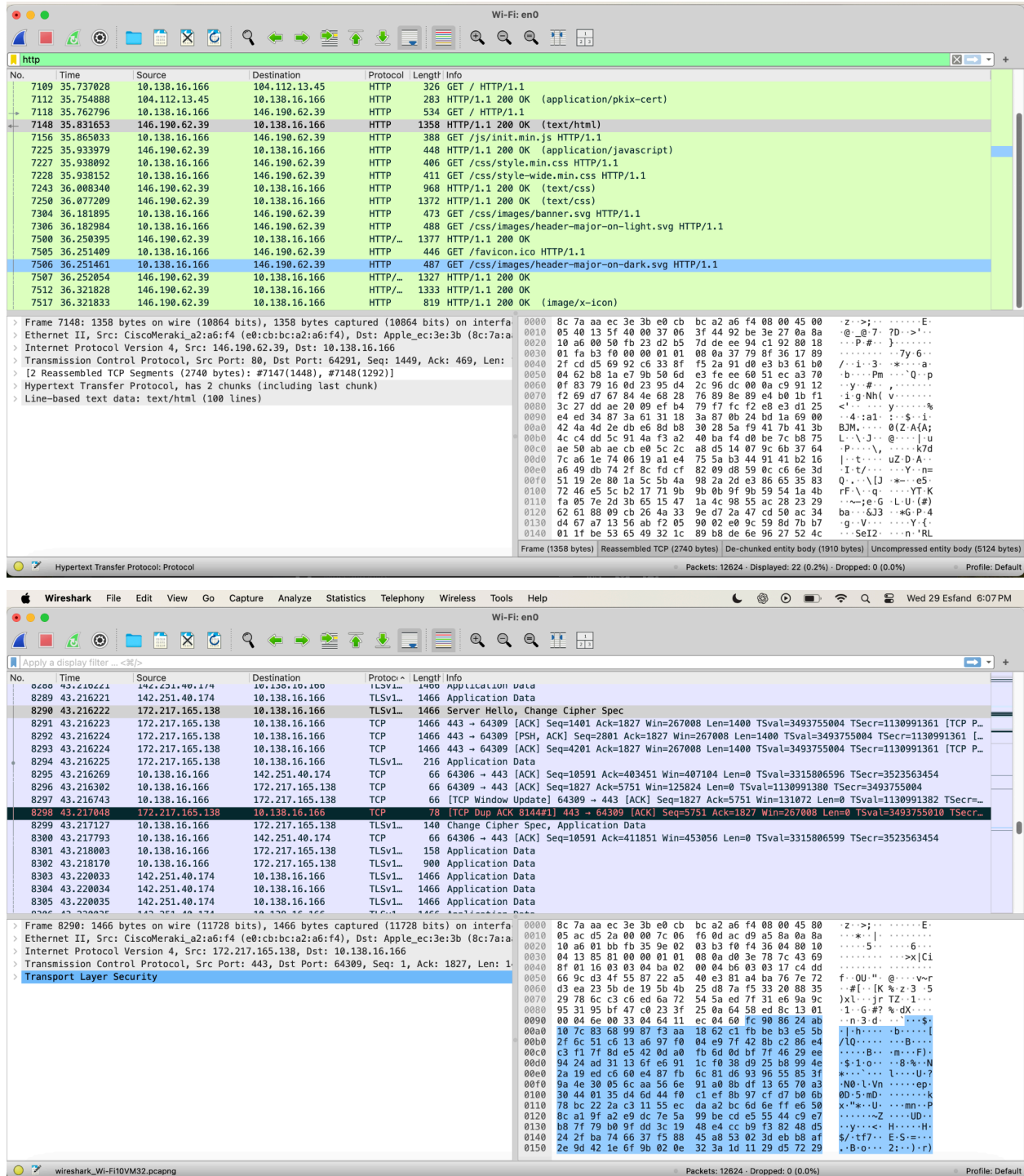
3. **Capture Filters**

- 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.



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.