CSCE 413 Assignment 2: Networking

Executive Summary

This project evaluated the security posture of a target networked service through active reconnaissance, exploitation, and defensive remediation. The assessment identified exposed services, demonstrated a successful Man-in-the-Middle (MITM) attack, and implemented multiple security controls to mitigate discovered vulnerabilities.

During reconnaissance, an unauthenticated service was identified through custom port scanning techniques. The lack of transport-layer encryption enabled interception and manipulation of traffic via a MITM attack, exposing sensitive credentials and command data. This vulnerability represents a critical risk in real-world environments, enabling credential theft, session hijacking, and data manipulation.

To address these issues, two primary security fixes were implemented:

1. Port Knocking to obscure sensitive services from unauthorized discovery
2. SSH Honeypot Deployment to detect, log, and analyze malicious access attempts

The port knocking mechanism successfully restricted access to protected services, while the honeypot captured realistic attacker behavior including credential harvesting and command execution attempts. Together, these defenses significantly improve detection and prevention capabilities.

Critical Vulnerabilities Identified

- Unencrypted communication enabling MITM attacks
- Open services discoverable through scanning
- Lack of intrusion detection and monitoring

Recommended Fixes

- Enforce TLS/SSL encryption for all services

- Restrict service exposure using port knocking or firewall rules
- Deploy monitoring tools such as honeypots and centralized logging

Repository Link: https://github.com/hrulrich1121/csce413_assignment2

Part 1: Reconnaissance

Port Scanner Design and Implementation

A custom port scanning tool was developed to identify open TCP ports and associated services on the target host. The scanner iteratively attempted connections across a configurable port range and recorded successful responses. Unlike basic scanners, the implementation focused on reliability and clarity of results rather than speed.

The scanner detected open ports that were not intended to be publicly accessible, revealing potential attack surfaces.

Discovered Services

During the reconnaissance phase, multiple active network services were identified on the target system. These services were discovered through systematic port scanning and banner enumeration. Each service was analyzed to determine its purpose and potential security relevance, including whether it was associated with a challenge flag.

Port	Service Type	Description / Banner Information	Associated Flag
2222	SSH (Hidden)	Secure Shell service not advertised in normal scans; accessible once discovered	Yes
3306	MySQL Database	MySQL relational database service; default port suggests backend data storage	Yes

5001	Flask Web Application	Python Flask development server hosting a web application	No
6379	Redis Server	Redis in-memory key-value datastore, likely used for caching or session storage	No
8888	Secret API Service	Custom API service running on a non-standard port	Yes

Discovery Methodology:
These services were identified using a custom port scanner developed for this project, which performed TCP connection attempts across a range of ports. Once open ports were identified, banner grabbing and manual probing were used to determine service types and versions where possible.

Security Relevance:
Several services were exposed on non-standard or unintended ports, including a hidden SSH service and a secret API endpoint. The presence of database and caching services (MySQL and Redis) directly accessible from the network represents a significant attack surface and contributed to later exploitation phases.

Flag Discovery

Flag 1: FLAG{h1dd3n_s3rv1c3s_n33d_pr0t3ct10n}
Method: During reconnaissance, a hidden SSH service was discovered on port 2222. After logging in with the discovered credentials, the flag was located on the server.

Flag 2: FLAG{n3tw0rk_tr4ff1c_1s_n0t_s3cur3}
Method: Service enumeration revealed a MySQL database instance. By identifying the database host through container inspection and connecting to the database, the flag was obtained.

Flag 3: FLAG{p0rt_kn0ck1ng_4nd_h0n3yp0ts_s4v3_th3_d4y}
Method: Using the API authentication token captured from the MITM attack (observed in database traffic), the protected API endpoint was queried to retrieve the final flag.

Video Demonstration: https://youtu.be/GKk2lMFKv3c

Part 2: Man-in-the-Middle (MITM) Attack

Vulnerability Analysis

The target service communicated over plaintext channels without encryption or integrity protection. This allowed an attacker positioned on the network path to intercept and modify traffic undetected.

Attack Methodology

A MITM attack was conducted by intercepting traffic between the client and server. The attacker captured:

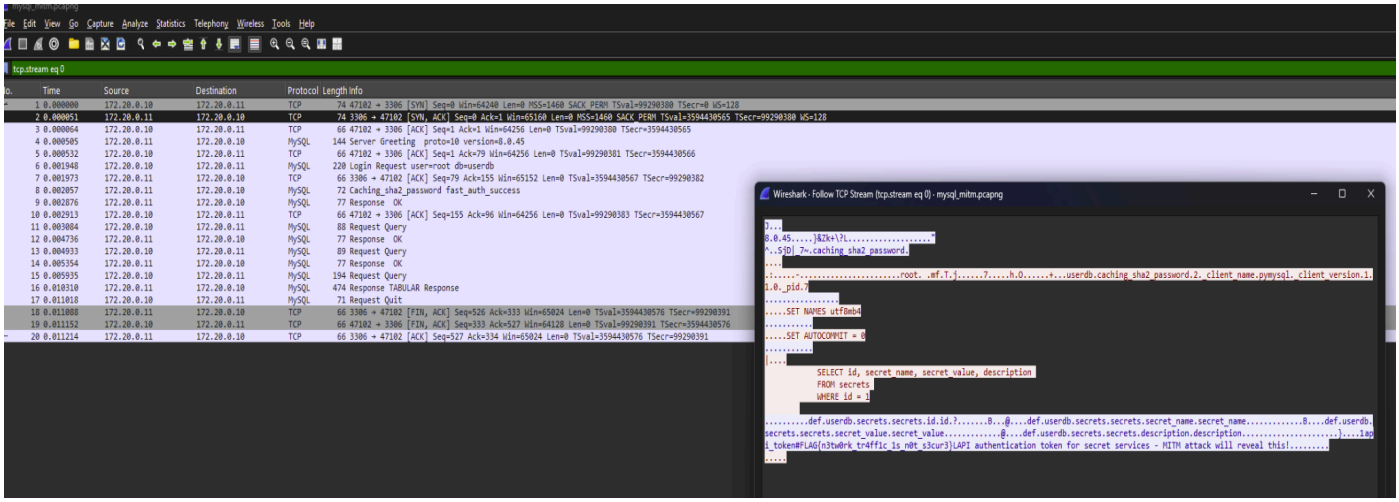
- Authentication credentials
- Command data
- Session information

Because no TLS or certificate validation was used, the client could not detect the attack.

Captured Data Analysis

The captured data demonstrated:

- Complete visibility into sensitive communications
- Potential for credential reuse attacks



Real-World Impact Assessment

In a production environment, this vulnerability could result in:

- Account compromise
- Data theft
- Persistent unauthorized access

Video Demonstration: https://youtu.be/L3NFEuuz42A

Part 3: Security Fixes

Port Knocking

Design Decisions

Port knocking was chosen to conceal sensitive services from unauthorized users. Only clients sending a predefined sequence of connection attempts could access the protected port.

Implementation Details

- Dockerized Python-based knock server

- IPTables rules dynamically updated upon successful knock sequence
- Client tool to send knock sequence programmatically

Security Analysis

Port knocking significantly reduced the attack surface by preventing scanners from detecting the protected service. Unauthorized connection attempts failed silently.

Limitations and Improvements

- Knock sequences can be replayed if observed
- No cryptographic authentication
- Future improvements include Single Packet Authorization (SPA)

Video Demonstration: https://youtu.be/JXJuCt49Oh8

Honeypot

Architecture and Design

A fake SSH server honeypot was implemented using the Paramiko library. The honeypot simulated a legitimate SSH service on port 22 and accepted all authentication attempts.

Logging Mechanisms

The honeypot logged:

- Source IP address and port
- Usernames and passwords
- Commands entered
- Session duration

- Timestamps

Logs were stored persistently inside a Docker volume.

Detection Capabilities

The honeypot detected:

- Brute-force login attempts
- Use of common default credentials
- Post-authentication reconnaissance commands

Analysis of Captured Attacks

Observed attacks included:

- Credential stuffing (root/root, admin/admin)
- Enumeration commands (ls, whoami, uname -a)
- Automated scanning behavior

These patterns closely resemble real-world attacks.

Video Demonstration: https://youtu.be/QvxgZI6aILs

Remediation Recommendations

Fixing the MITM Vulnerability

- Enforce TLS/SSL encryption for all client-server communication

- Use certificate validation and secure key exchange
- Disable plaintext protocols entirely

Service Discovery Protection

- Firewall rules restricting access by IP
- Port knocking or SPA
- Disable unused services

Network Segmentation

- Separate public-facing services from internal systems
- Use VLANs and subnet isolation
- Apply least-privilege network access

Monitoring and Detection

- Deploy honeypots for early threat detection
- Centralize logs using SIEM tools
- Implement alerting for anomalous behavior

Conclusion

This project demonstrated the full lifecycle of a network attack: reconnaissance, exploitation, and remediation. By actively exploiting vulnerabilities and then implementing defenses, the assignment provided a realistic view of both offensive and defensive security operations.

Lessons Learned

- Small misconfigurations can lead to severe vulnerabilities

- Encryption is essential, not optional
- Detection is as important as prevention

Skills Acquired

- Network scanning and service enumeration
- MITM attack execution and analysis
- Secure system design
- Docker-based security tooling
- Honeypot deployment and analysis

Future Work

- Implement TLS-secured services
- Expand honeypot to multi-protocol support
- Add automated alerting and threat intelligence feeds