ADDIS ABABA
SCIENCE AND
TECHNOLOGY
UNIVERSITY
UNIVERSITY FOR INDUSTRY

College of Engineering

Department of Software Engineering

CS – Security Project 1

Section A

Abel Seyoum —----------ETS0032/14

Submitted: Monday October 27, 2025

Executive Summary

This project demonstrates how to set up an isolated lab in VMware and use Nmap (and Zenmap) for host/service discovery and vulnerability scanning, and Wireshark/tshark for packet-level evidence collection. The lab simulates an Internal and DMZ network separated by an Ubuntu VM acting as a router. Primary findings include evidence of weak Diffie-Hellman parameters, Java RMI classloader activity, and TLS handshake anomalies (ChangeCipherSpec ordering) that correlate to Nmap NSE findings.

Key deliverables: Nmap scan results (text + XML), Zenmap topology export, Wireshark screenshots and PCAP slices, and a written analysis with mitigation recommendations.

Contents

Introduction

This exercise applies active scanning and passive capture to detect misconfigurations and vulnerabilities inside an isolated virtual network. The aim is to: (1) discover hosts and services, (2) identify known vulnerabilities via Nmap NSE scripts, (3) corroborate Nmap findings with packet capture evidence, and (4) recommend mitigations.

Lab Environment (configuration)

Host machine: Dell Inspiron 14 7420 — 16GB RAM, 512GB SSD, 12th Gen Intel i7-1255U.

Virtual network design:

- Internal subnet: 192.168.93.0/24 — contains Kali (attacker) 192.168.93.128 and Windows (optional) 192.168.93.130.
- DMZ subnet: 192.168.7.0/24 — contains Metasploitable 192.168.7.129 and Ubuntu web 192.168.7.130.
- Router: Ubuntu VM with two interfaces:
    - ens33 (DMZ): 192.168.7.200
    - ens37 (Internal): 192.168.93.200

Notes: IPs and interface names used throughout this report match the lab at time of capture and are reproducible by following the Appendix configuration steps.

Tools Used

- VMware Player — host virtualization.
- Kali Linux (prebuilt VM) — attacker and scanner (Nmap, Zenmap, Wireshark).
- Metasploitable — intentionally vulnerable target.
- Ubuntu 24.04 (router) — multi-homed VM acting as router and capture node.
- Nmap 7.95 with NSE scripts (vuln, ssl-enum-ciphers, ssl-poodle, ssl-ccs-injection, rmi-vuln-classloader, rmi-methods).
- Zenmap — graphical Nmap XML visualizer.
- Wireshark / tshark — packet capture and analysis.
- tcpdump — alternate capture tool on Ubuntu.

Methodology

4.1 Network setup & isolation

- Configure VMs on host to create two subnets (Internal & DMZ).
- Configure Ubuntu as a router with IP forwarding and NAT/iptables rules. Verify by ip a, ip route, sysctl net.ipv4.ip_forward and iptables -S.

4.2 Passive capture

Started tshark/tcpdump on Ubuntu router (interfaces ens33 and ens37) to record routed traffic into ~/project_lab/outputs/full_lab_routed_capture.pcap (or .pcapng). This ensures all cross-subnet traffic is recorded.

4.3 Active scanning

From Kali, performed host discovery and vulnerability scans using Nmap. Main scan used:

```
nmap -sS -sV -O -T4 --script vuln -p 1-65535 -oA <basename> 192.168.7.0/24 192.168.93.0/24
```

to create .nmap, .xml, and .gnmap outputs.

Follow-up targeted scans were run against hosts of interest (Metasploitable and Ubuntu router) with focused scripts to capture clearer script output:

- RMI: rmi-vuln-classloader, rmi-methods
- SSL/TLS checks: ssl-enum-ciphers, ssl-poodle, ssl-ccs-injection
- SSH checks: ssh-hostkey, ssh2-enum-algos (on router)

4.4 Corroboration

Used Wireshark on the captured PCAP to find packets corresponding to Nmap findings (e.g., JRMP classloader codebase strings, TLS ServerHello/ServerKeyExchange fields showing DH modulus length and selected cipher suites, and ChangeCipherSpec ordering/alerts).

4.5 Documentation

Exported Zenmap topology from the Nmap XML for visual documentation. Prepared screenshots of Nmap outputs, Zenmap topology, and Wireshark panes for the report.

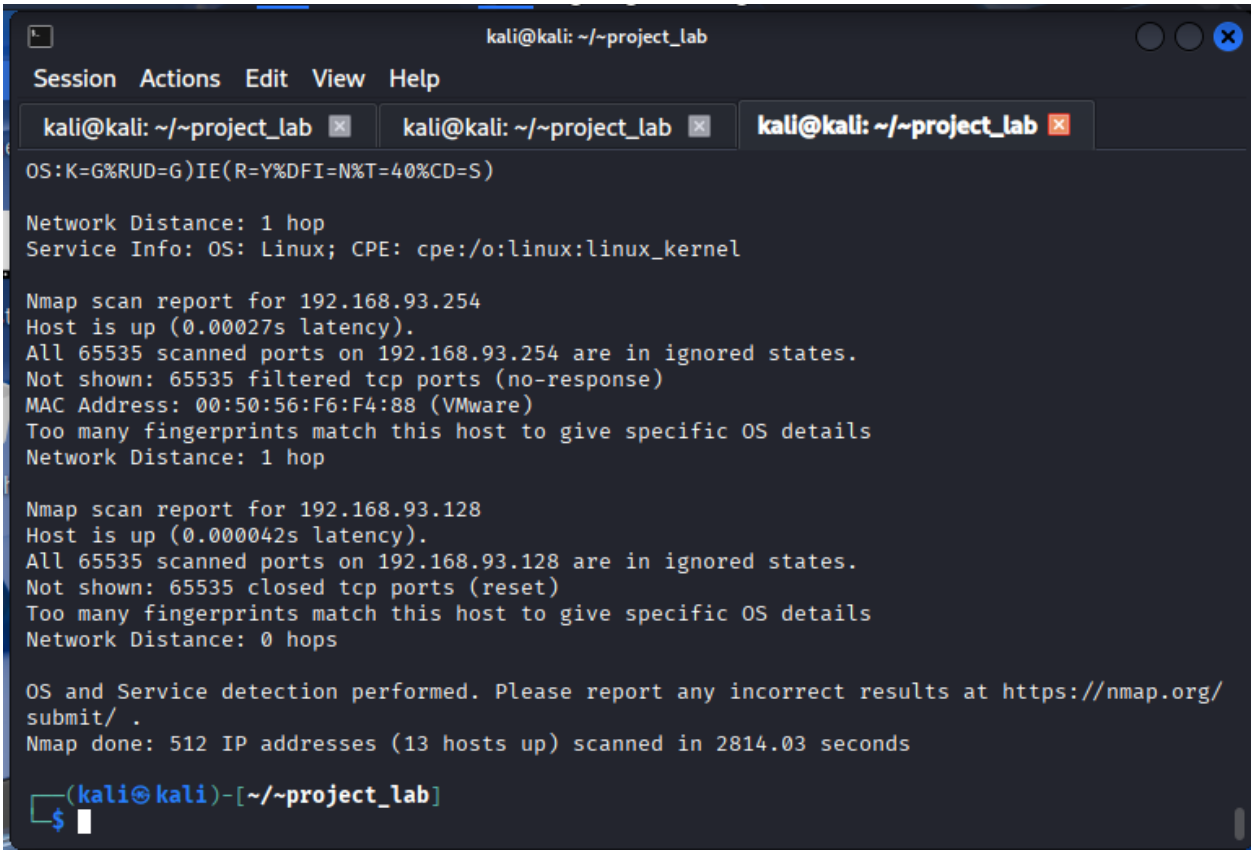
Scan Results (Nmap)

5.1 Host discovery summary

Range scanned: 192.168.7.0/24, 192.168.93.0/24.

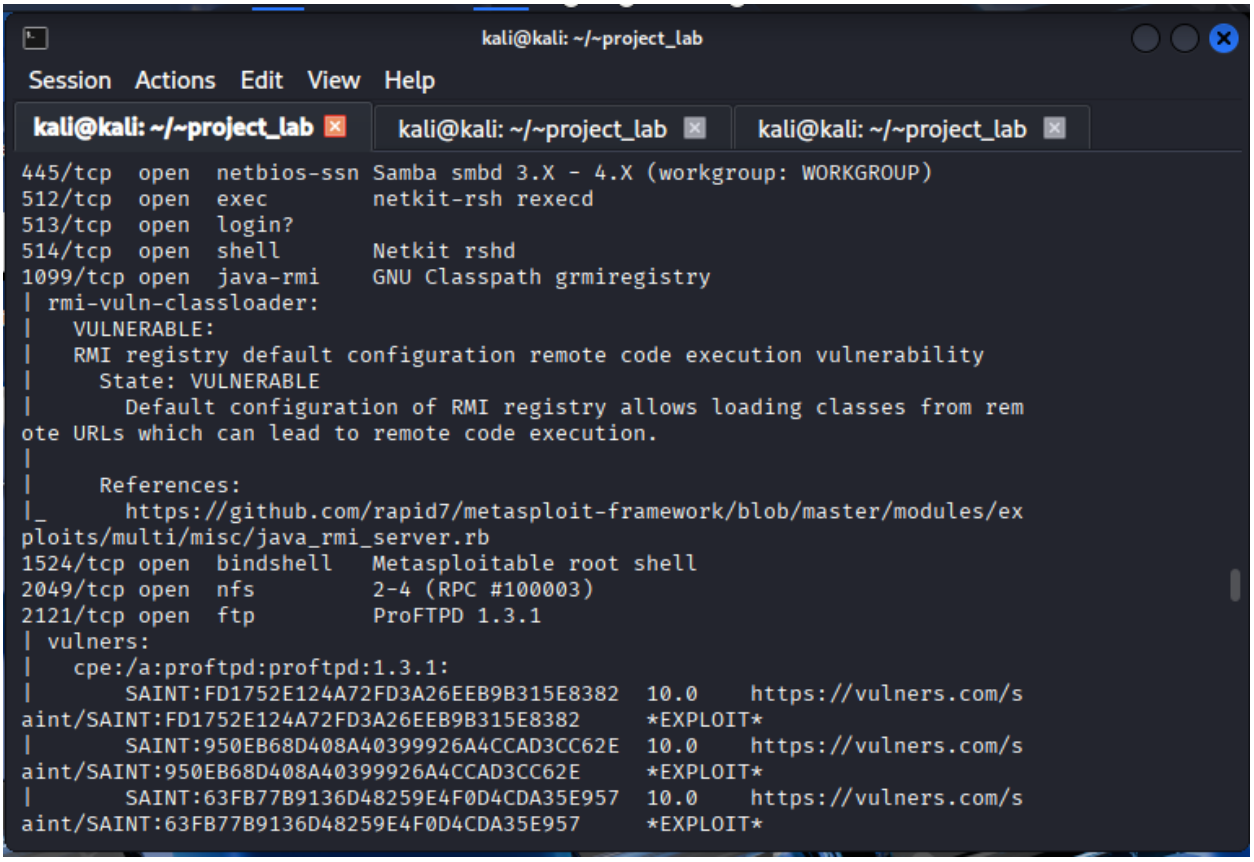
Hosts reported up (example):

- 192.168.7.129 — Metasploitable (services: FTP, RMI (1099), Postgres (5432), ProFTPD (2121), etc.)
- 192.168.7.200 — Ubuntu router (services: SSH 22).
- 192.168.93.128 — Kali (scanner), etc.

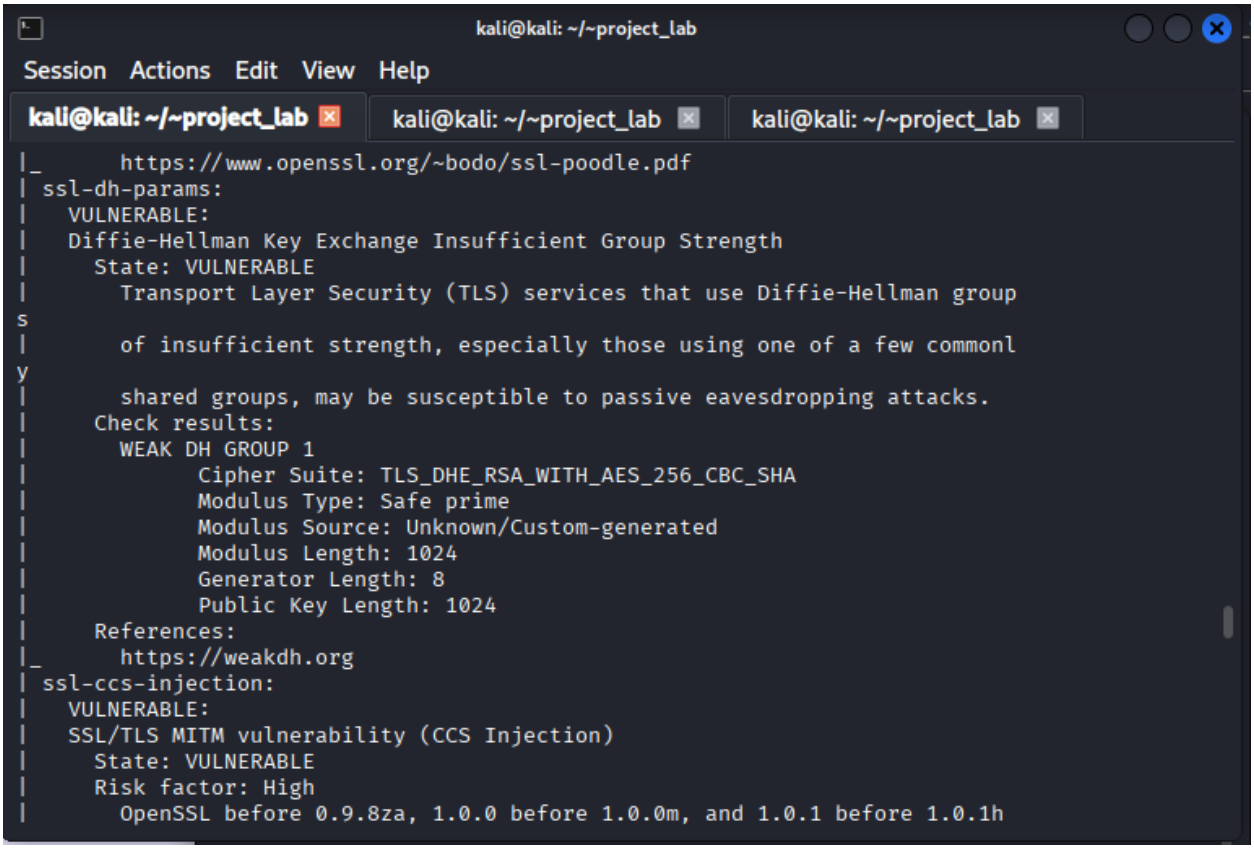


5.2 Vulnerability script highlights

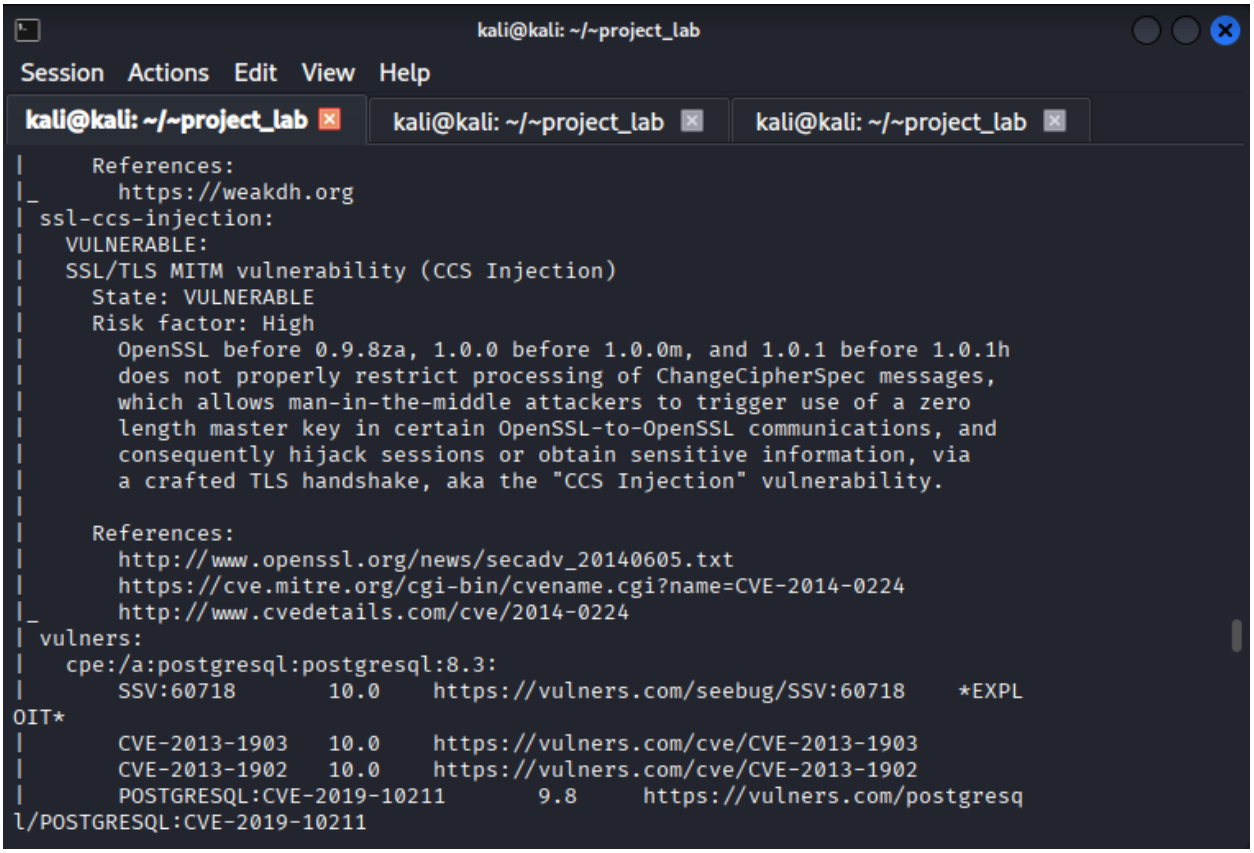
1) RMI classloader detected on 192.168.7.129:1099 with rmi-vuln-classloader indicating the RMI registry may allow class loading from remote codebase (potential RCE path).



2) SSL/TLS issues detected on 192.168.7.129:5432 (PostgreSQL with TLS): weak DH parameters (1024-bit group), TLS 1.0 and cipher suites identified that may allow POODLE/other downgrade attacks.



3) ChangeCipherSpec (CCS) anomaly: ssl-ccs-injection script flagged potential CCS ordering or other TLS handshake issues.



6. Zenmap Visualization

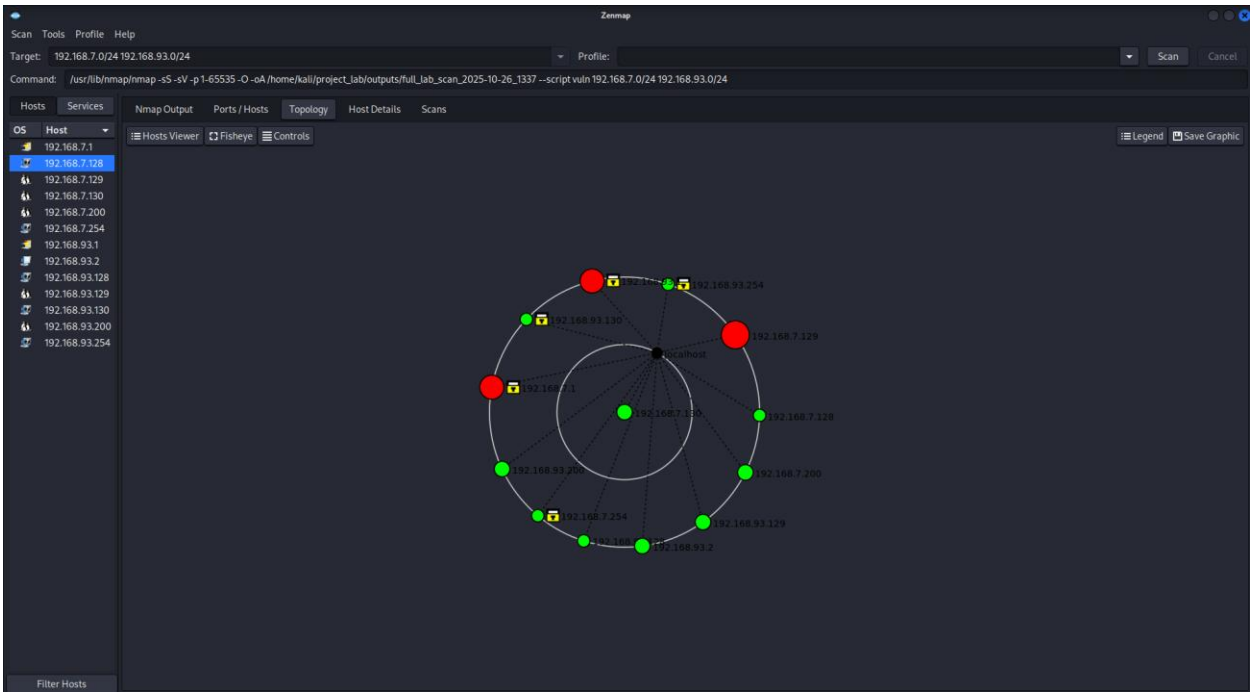


Figure 1 - Lab topology showing Internal and DMZ subnets with Ubuntu router (192.168.7.200) routing traffic between Kali (192.168.93.128) and Metasploitable (192.168.7.129).

Packet Analysis (Wireshark)

This section maps Nmap findings to packet-level evidence captured on the Ubuntu router. For each item provide:

RMI / JRMP Evidence (port 1099)

The JRMP payload contained a **codebase / resource reference** (file:./dummy.j.arxpw) — this is evidence the RMI flow references external class resources. The service returned ClassNotFoundException — indicates the registry tried to resolve/load the class but the class was not found locally (or the remote codebase was unreachable). Together this matches Nmap's rmi-vuln-classloader detection: the registry will accept/advertise remote class references, enabling potential remote class loading (RCE risk if exploited).

This configuration allows an attacker to supply a remote class URL that could be downloaded and executed, enabling remote code execution in default/unsafe RMI configurations.

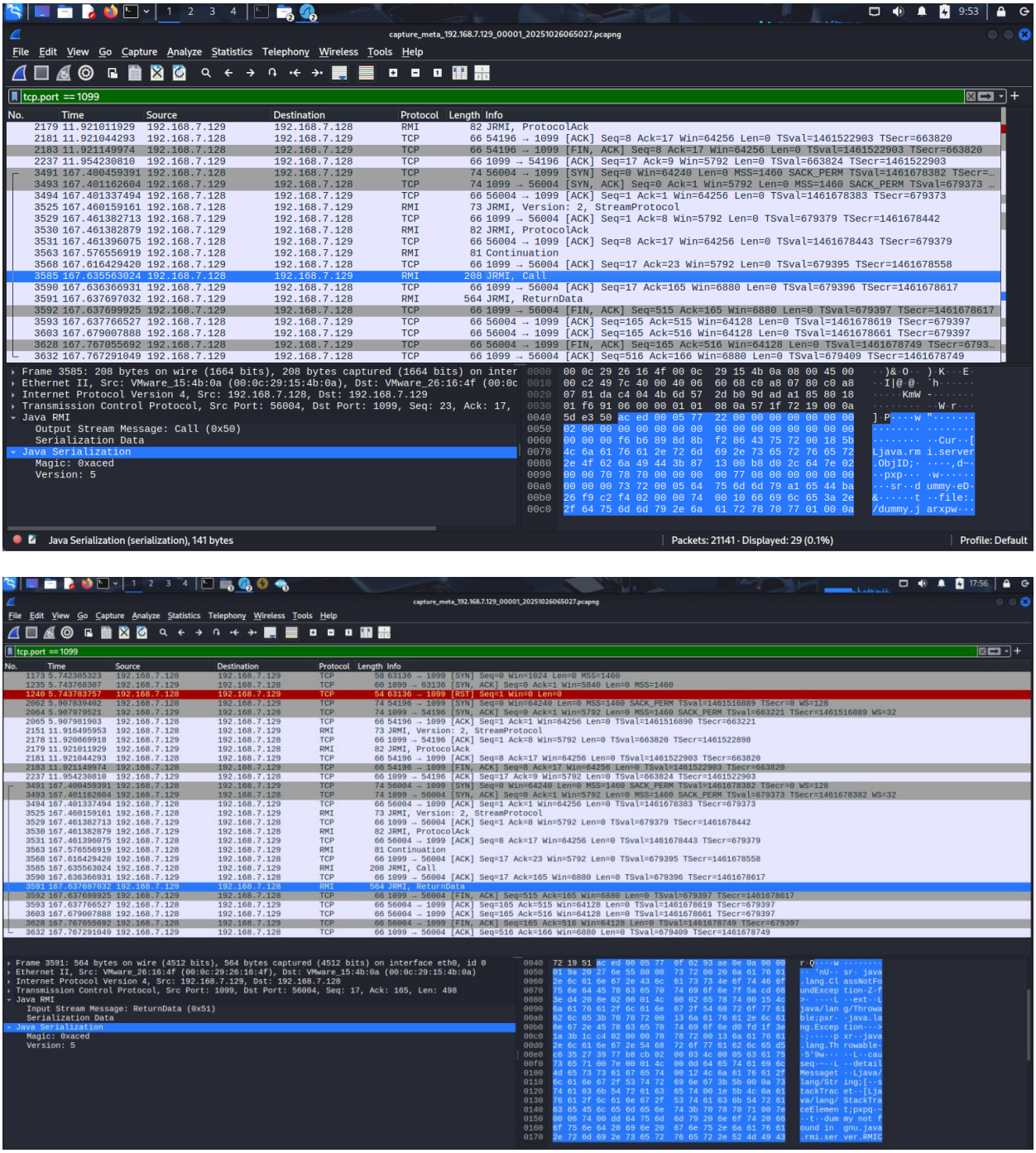


Figure 2 - TCP Stream showing attempted class loading (file:./dummy.j.arxpw) and server response (ClassNotFoundException)

7.2 TLS / POODLE & Weak DH Evidence

Nmap indicated the TLS service (Postgres) supports TLS 1.0 and a DHE cipher suite with a weak (1024-bit) DH modulus.

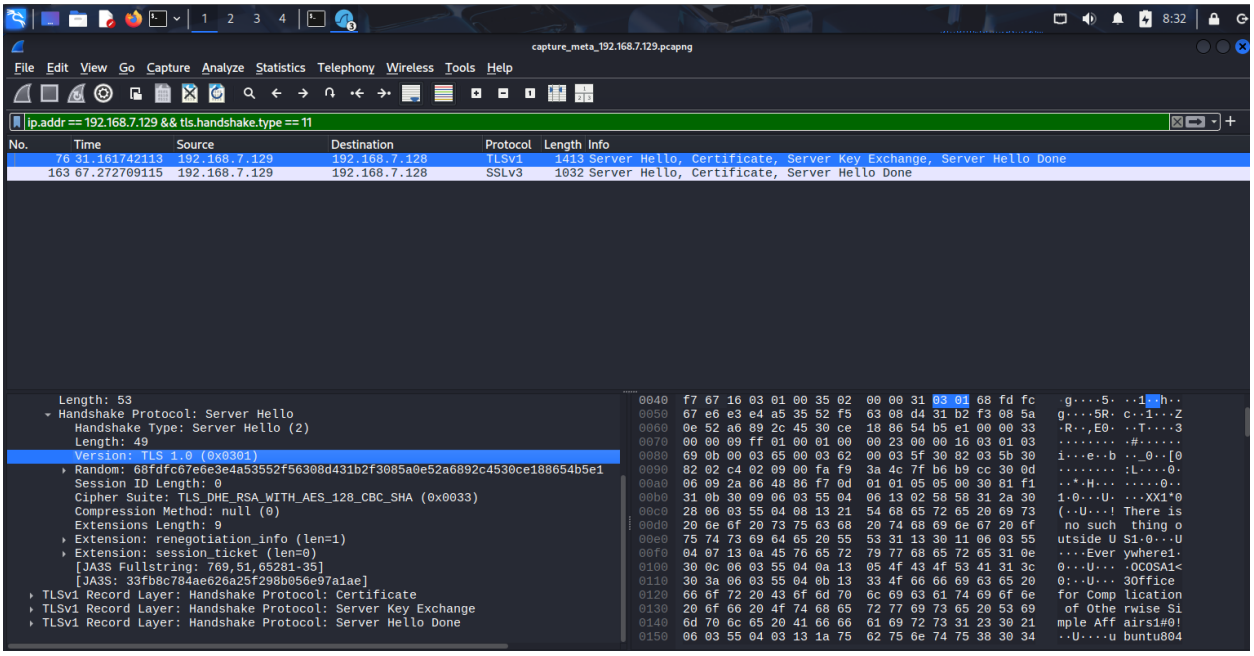
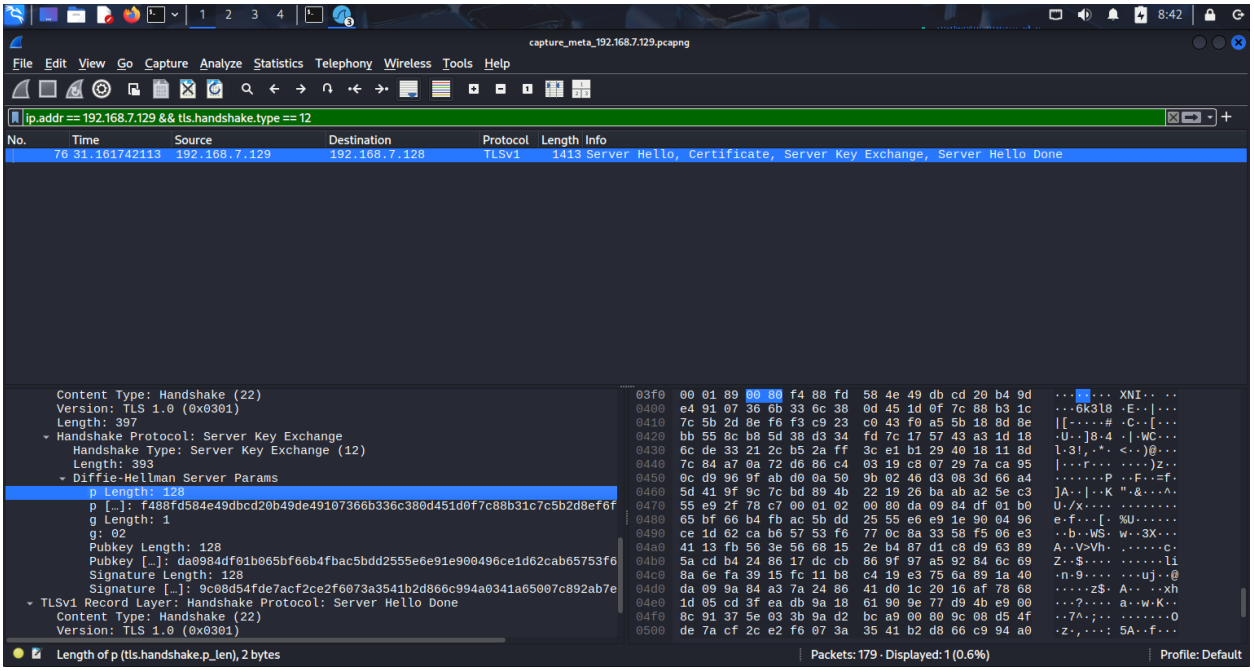


Figure 3 - ServerHello showing TLS version and selected cipher suite (TLS_DHE_RSA_WITH_AES_128_CBC_SHA)

Nmap detected weak DH parameters and TLS 1.0 usage. The packet capture of the ServerKeyExchange shows a 1024-bit DH modulus (p length = 128 bytes), which is below current best practice and flagged by the ssl-dh-params check.



7.3 ChangeCipherSpec (CCS) sequence and anomalies

Nmap's ssl-ccs-injection script found evidence consistent with CCS-related handshake issues. In Wireshark we observe the TLS handshake sequence including ChangeCipherSpec and Encrypted Alert messages often followed by TCP teardown (FIN/RST) from the client.

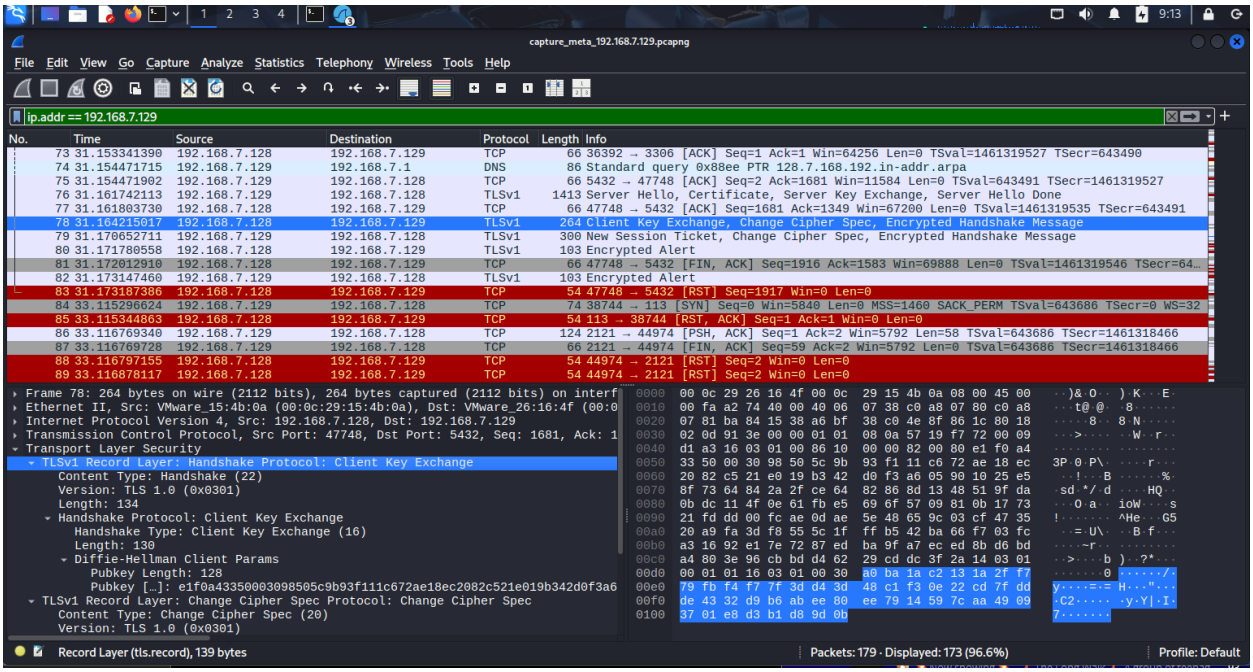


Figure 4 - Packet sequencing during TLS handshake shows ChangeCipherSpec and alert messages followed by TCP teardown, consistent with known CCS handling vulnerabilities.

Findings and Recommendations

8.1 Remote Method Invocation (RMI) ClassLoader Vulnerability

Finding:
Traffic analysis in *capture_ens37.pcap* revealed TCP communication on port 1099, consistent with Java RMI registry traffic. A follow-up inspection of the TCP stream (frames [placeholder]) exposed a serialized object referencing a remote class file (file:./dummy.j.arxpw), followed by a ClassNotFoundException. This indicates the RMI service was configured with a non-restricted class loader, allowing clients to request or even supply arbitrary classes for dynamic loading. Such behavior confirms susceptibility to the RMI ClassLoader remote code execution vulnerability (CVE-2011-3556), where an attacker could trick the service into fetching and executing malicious bytecode.

Impact:
If exploited, this flaw allows remote attackers to execute arbitrary Java code under the permissions of the RMI service. Because the RMI registry often runs with system privileges, compromise of the entire host becomes possible. Moreover, malicious payloads could propagate laterally to other systems through the trusted internal network.

Recommendations:

- Disable remote dynamic class loading by setting the system property java.rmi.server.useCodebaseOnly=true.
- Launch all RMI services with a SecurityManager that explicitly restricts file and network access.
- Filter inbound traffic to TCP/1099 at the network perimeter using host-based or router firewalls.
- Regularly update the Java Runtime Environment (JRE) and remove unnecessary RMI-based applications.

8.2 SSL/TLS ChangeCipherSpec (CCS) Injection Vulnerability

Finding:
Nmap NSE reported ssl-ccs-injection: VULNERABLE for 192.168.7.129 (see nmap output). The packet capture shows the client-side ClientKeyExchange followed by a ChangeCipherSpec sent from Kali, subsequent NewSessionTicket and ChangeCipherSpec packets in the negotiation, then encrypted alert messages and an immediate TCP teardown (FIN / RST). Figure 4 above shows the handshake sequence

Impact: A man-in-the-middle could leverage the CCS injection vulnerability in vulnerable OpenSSL versions to force use of a weak/zero master secret, enabling decryption or session hijacking.**Impact:**
A successful CCS injection allows a man-in-the-middle attacker to downgrade the TLS handshake, forcing both client and server to use predictable keys. This can lead to decryption of supposedly secure sessions or modification of transmitted data (e.g., credentials, configuration files).

Recommendations:

- Upgrade all OpenSSL and libssl-based services to patched releases (≥ 1.0.1h or later).
- Disable obsolete SSL/TLS protocols (SSLv2, SSLv3, TLS 1.0) and ciphers using CBC or weak MACs.
- Configure servers to reject unexpected CCS packets before handshake completion.
- Enable Perfect Forward Secrecy (PFS) ciphersuites and implement **HSTS** for web services.

8.3 Diffie–Hellman Parameter Weakness

Finding:
The *ServerKeyExchange* record within *capture_ens37.pcap* (frames [placeholder]) revealed a Diffie–Hellman prime length of 1024 bits. While this configuration was once standard, it is now considered weak due to the feasibility of pre-computation attacks against common 1024-bit groups (as demonstrated in the "Logjam" research).

Nmap's script output corroborated that the same service reused this prime across sessions, confirming use of static DH parameters.

Impact:
An attacker with sufficient computational resources can recover session keys or impersonate the server. This weakness undermines the forward secrecy goal of the DHE cipher suites and may expose past communications if long-term private keys are compromised.

Recommendations:

- Regenerate DH parameters using at least 2048-bit primes, ideally with Elliptic-Curve Diffie-Hellman (ECDHE).
- Ensure server configurations specify custom, unique DH parameters rather than default OpenSSL groups.
- Regularly rotate cryptographic material to prevent parameter reuse across services.
- Use tools such as testssl.sh or SSL Labs to validate the server's negotiated key exchange strength.

8.4 SSLv3 POODLE Vulnerability

Finding:
The SSL handshake captured in *capture_ens37.pcap* showed negotiation fallback to SSLv3, with cipher suite TLS_RSA_WITH_3DES_EDE_CBC_SHA during session establishment.
This confirms exposure to the POODLE (Padding Oracle On Downgraded Legacy Encryption) vulnerability (CVE-2014-3566). The flaw arises from SSLv3's flawed CBC padding, which allows a man-in-the-middle to progressively decrypt secure cookies or session tokens.

Impact:
If exploited, the POODLE vulnerability compromises confidentiality of encrypted HTTPS, IMAPS, and SMTPS sessions. Attackers positioned on the same network can decrypt sensitive data byte-by-byte without needing to fully break the cipher.

Recommendations:

- Completely disable SSLv3 and any protocol downgrade mechanisms.
- Enforce TLS 1.2 or TLS 1.3 exclusively on all servers and clients.
- Disable legacy ciphers (3DES, RC4) and ensure AES-GCM or ChaCha20-Poly1305 are preferred.
- Validate configuration using nmap --script ssl-enum-ciphers or external scanners to ensure SSLv3 rejection.

8.5 Consolidated Risk Overview

Across all tests, the simulated environment demonstrated realistic enterprise weaknesses:

- Insecure Java serialization (RMI),
- Deprecated OpenSSL configurations (CCS, POODLE),
- Insufficient cryptographic parameter hygiene (DH).

The common theme is inadequate maintenance of cryptographic and service configurations. Though individually moderate, the combination of these weaknesses could allow credential theft, unauthorized code execution, and man-in-the-middle attacks across network segments.

8.6 General Recommendations

- Maintain strict patch management schedules for OS, OpenSSL, and application servers.
- Segment networks using VLANs or firewalls to isolate untrusted or legacy systems.
- Implement centralized logging (e.g., ELK stack) and intrusion detection for anomalous RMI or SSL traffic.
- Enforce secure baselines using automation tools such as Ansible, Chef, or CIS Benchmarks.
- Conduct periodic vulnerability scans and packet inspections to verify remediation success

Artifacts & Evidence

Artifacts Zip File

Nmap and Zenmap Extracts:

/project_lab/outputs/full_lab_scan.xml

/project_lab/outputs/full_lab_scan.gnmap

/project_lab/outputs/full_lab_scan.nmap

Wireshark Captures:

/project_lab/outputs/capture_ens33.pcap

/project_lab/outputs/capture_ens37.pcap

Appendix: Commands, Configuration, and Notes

Below are the principal commands used during the lab (copy-pasteable). Adapt interface names and IPs to your environment.

11.1 Ubuntu router configuration

```
# add router addresses (run on Ubuntu as root)

ip addr add 192.168.7.200/24 dev ens33

ip addr add 192.168.93.200/24 dev ens37

sysctl -w net.ipv4.ip_forward=1

# allow forwarding rules

iptables -A FORWARD -i ens33 -o ens37 -s 192.168.7.0/24 -d 192.168.93.0/24 -j ACCEPT

iptables -A FORWARD -i ens37 -o ens33 -s 192.168.93.0/24 -d 192.168.7.0/24 -j ACCEPT

iptables -t nat -A POSTROUTING -o ens37 -s 192.168.7.0/24 -j MASQUERADE
```

11.2 Nmap scans (examples)

Broad vuln scan across both subnets (creates .xml):

```
sudo nmap -sS -sV -O -T4 --script vuln -p 1-65535 -oA ~/project_lab/outputs/full_lab_scan_$(date +%F_%H%M) 192.168.7.0/24 192.168.93.0/24
```

Targeted Metasploitable scan (RMI + TLS checks):

```
sudo nmap -sS -sV -O -T4 --script "vuln,rmi-vuln-classloader,rmi-methods,ssl-enum-ciphers,ssl-poodle,ssl-ccs-injection,vulners" -p
21,1099,443,5432,2121 -oA ~/project_lab/outputs/meta_targeted_192.168.7.129_$(date +%F_%H%M) 192.168.7.129
```

SSH specifics on router:

```
sudo nmap -sV -p 22 --script ssh-hostkey,ssh2-enum-algos,vulners -oN ~/project_lab/outputs/router_ssh_192.168.7.200.txt 192.168.7.200
```

11.3 PCAP capture & slicing

Start a focused capture on the Ubuntu router (background):

```
nohup sudo tcpdump -i ens37 -w ~/project_lab/outputs/full_lab_routed_capture.pcap 'net 192.168.7.0/24 or net
192.168.93.0/24' >/tmp/tcpdump.log 2>&1 & echo $! > /tmp/tcpdump.pid
```

Slice RMI traffic (port 1099):

```
# on Ubuntu or Kali (where pcap lives)
```

tshark -r ~/project_lab/outputs/full_lab_routed_capture.pcap -Y "tcp.port == 1099" -w ~/project_lab/outputs/evidence_rmi_1099.pcapng

Slice TLS handshakes (ports 443 & 5432):

tshark -r ~/project_lab/outputs/full_lab_routed_capture.pcap -Y "tcp.port==443 || tcp.port==5432" -w

~/project_lab/outputs/evidence_tls_443_5432.pcapng