This document consolidates the methodology, tools, and execution steps report.

🛡️ **Infrastructure Security Audit Report**

Project: Network Discovery & Service Mapping

Task Reference: DEV-352

Date: January 2026

1. 🏗️ Setup & Environment

The audit was conducted using a dedicated Linux-based security environment to ensure tool compatibility and raw socket access.

- **Operating System:** Ubuntu Linux (CLI) / Kali Linux
- **User Privileges:** sudo (Root) access was required to send raw TCP SYN packets (-sS) and perform OS fingerprinting (-O).
- **Network Access:** The scanning machine was whitelisted on the target firewall to prevent IP blocking during the audit.
- **Directory Structure:**

Scan IP Results

IP Address	Port	Protocol	Service	Version / Details	Risk Level
64.23.130.208	22	SSH	OpenSSH	8.9p1 Ubuntu 3ubuntu0.1	🟢 Low
157.230.47.60	22	SSH	OpenSSH	8.9p1 Ubuntu 3ubuntu0.1	🟢 Low
	53	TCP	tcpwrapped	Firewall/Protection	🟢 Low
	8500	HTTP	Consul Agent	Golang net/http	🟡 Medium
	8600	TCP	tcpwrapped	Firewall/Protection	🟢 Low
	9000	SSL/HTTP	MinIO Storage	Golang net/http	🟡 Medium
	9001	SSL/HTTP	MinIO Console	Golang net/http	🟡 Medium
159.223.62.168	22	SSH	OpenSSH	8.9p1 Ubuntu 3ubuntu0.1	🟢 Low

IP Address	**Port**	**Protocol**	**Service**	**Version / Details**	**Risk Level**
	8500	HTTP	Consul Agent	Golang net/http	🟡 Medium
139.59.245.244	22	SSH	OpenSSH	8.9p1 Ubuntu 3ubuntu0.1	🟢 Low
143.198.94.161	22	SSH	OpenSSH	8.9p1 Ubuntu 3ubuntu0.1	🟢 Low
188.166.250.175	22	SSH	OpenSSH	8.9p1 Ubuntu 3ubuntu0.1	🟢 Low
	53	TCP	tcpwrapped	Firewall/Protection	🟢 Low
	80	HTTP	OpenResty	Web App Server	🟢 Low
	443	HTTPS	OpenResty	SSL/HTTPS	🟢 Low
	9111	TCP	Redis	Key-Value Store	🔴 **High**
139.59.117.80	22	SSH	OpenSSH	8.9p1 Ubuntu 3ubuntu0.1	🟢 Low
	53	TCP	tcpwrapped	Firewall/Protection	🟢 Low
	80	HTTP	OpenResty	Web App Server	🟢 Low
	443	HTTPS	OpenResty	SSL/HTTPS	🟢 Low
	9111	TCP	Redis	Key-Value Store	🔴 **High**
134.209.107.38	22	SSH	OpenSSH	8.9p1 Ubuntu 3ubuntu0.1	🟢 Low
	8085	HTTP	Apache httpd	2.4.52 (Ubuntu)	🟢 Low
	8500	HTTP	Consul Agent	Golang net/http	🟡 Medium
146.190.97.129	22	SSH	OpenSSH	8.9p1 Ubuntu 3ubuntu0.1	🟢 Low
	8500	HTTP	Consul Agent	Golang net/http	🟡 Medium
128.199.134.178	22	SSH	OpenSSH	8.9p1 Ubuntu 3ubuntu0.1	🟢 Low
	8500	HTTP	Consul Agent	Golang net/http	🟡 Medium
167.172.66.204	22	SSH	OpenSSH	8.9p1 Ubuntu 3ubuntu0.1	🟢 Low

IP Address	Port	Protocol	Service	Version / Details	Risk Level
	6001	HTTP	Uvicorn	ASGI Server (Python)	🟡 Medium
	6100	HTTP	Uvicorn	ASGI Server (Python)	🟡 Medium
	8500	HTTP	Consul Agent	Golang net/http	🟡 Medium
139.59.113.219	22	SSH	OpenSSH	8.9p1 Ubuntu 3ubuntu0.1	🟢 Low
	8000	HTTP	Uvicorn	ASGI Server (Python)	🟡 Medium
159.89.196.66	22	SSH	OpenSSH	8.9p1 Ubuntu 3ubuntu0.1	🟢 Low
	8500	HTTP	Consul Agent	Golang net/http	🟡 Medium
139.59.230.32	22	SSH	OpenSSH	8.9p1 Ubuntu 3ubuntu0.1	🟢 Low
139.59.99.241	22	SSH	OpenSSH	8.9p1 Ubuntu 3ubuntu0.1	🟢 Low
	443	HTTPS	Generic SSL	SSL Service	🟢 Low

---

2. 🔐 Security & Safety Protocols

To ensure the audit was "Non-Destructive" and safe for production servers, the following rules were enforced:

- **Timing Control:** We utilized the -T4 (Aggressive) timing template[2]. This optimizes speed but includes congestion control to prevent Denial of Service (DoS) conditions on the target servers.
- **No Exploitation:** The scan was limited to **Discovery Only**. No exploit payloads or intrusive vulnerability scripts were executed.
- **Data Sanitization:** All external reports (like GitHub portfolios) have had real IP addresses replaced with placeholders (e.g., Server-01) to prevent information leakage.

3. 🛠️ Tools & Technologies

The following stack was utilized to execute the task:

Tool	Purpose	Key Flags Used
Nmap (v7.95)	Core Scanning Engine	-sV (Version Detect), -p- (All Ports), -Pn (No Ping)
Bash	Automation	Scripting loops to iterate through the targets.txt inventory.
Netcat (nc)	Manual Verification	nc -zv to handshake with ports that showed "Filtered" or ambiguous results.
Curl	Service Validation	curl -I to fetch HTTP headers and confirm web server activity.

4. ⚙️ Methodology & Execution (How It Was Done)

The project followed a strict three-phase methodology defined in **DEV-353**[3].

Phase 1: Host Discovery (Ping Sweep)

Objective: Determine which hosts in the inventory were online.

- **Command:** nmap -sn -iL targets.txt [4]
- **Action:** This sent ICMP Echo requests to list active IPs without scanning ports, establishing our "Live Host" list.

Phase 2: Comprehensive Service Scanning

Objective: Identify every open port and the software version running on it.

- **Command:** sudo nmap -sV -p- -T4 -Pn -iL targets.txt -oN scan_results.txt [5]
- **Technical Breakdown:**
  - sudo: Required for accurate TCP SYN scanning.
  - -sV: Interrogated open ports to find details like "OpenSSH 8.9" instead of just "SSH".
  - -p-: Scanned ports 1–65535 to catch services hidden on non-standard ports (e.g., MinIO on 9000).
  - -Pn: Treated all hosts as online, bypassing firewalls that block Ping.

Phase 3: Validation & Analysis

Objective: Confirm findings and assess risk.

- **Manual Spot Checks:** High-risk ports (like Redis on 9111) were manually tested using netcat (nc -zv [IP] 9111) to prove they were reachable from the public internet.
- **Risk Categorization:** Findings were categorized based on exposure:
  - **Low:** Standard services (SSH/HTTP).
  - **Medium:** Management interfaces (Consul/MinIO) that should be VPN-restricted.
  - **High:** Unprotected databases (Redis) exposed to the public.

5. ✅ Conclusion

The audit successfully mapped 15 active hosts. While the SSH and Web layers are standard, the discovery of exposed Database (Redis) and Object Storage (MinIO) ports indicates a need for immediate firewall remediation to restrict access to internal IP ranges only.