📘 Service Version Mapping & Inventory Report

Project: Network Discovery & Service Mapping

Task Reference: DEV-354

Date: January 2026

1. 📋 Objective

The objective of this phase was to move beyond simple port discovery ("Is it open?") to Service Fingerprinting ("What is running?"). This process involved interrogating active ports to determine the specific application protocol, software vendor, and version number. This data is critical for identifying Vulnerabilities (CVEs) associated with outdated software.

2. 🛠️ Toolchain & Techniques

We utilized a combination of automated scanning and manual banner grabbing to ensure accurate identification.

Tool	Category	Command/Technique	Purpose
Nmap	Automation	-sV (Version Detect)	Compares service responses against the Nmap Service Probes Database (nmap-service-probes).
Curl	Manual	curl -I -X GET	Fetches HTTP headers to identify Web Server versions (e.g., "Server: nginx/1.18").
Netcat	Manual	nc -v	Performs TCP Banner Grabbing for non-HTTP services (e.g., Redis, SSH).

3. ⚙️ Execution Methodology

Step 1: Version Interrogation (Automated)

We executed Nmap with the Service Versioning flag enabled.

- Command: nmap -sV -p- -T4 -iL targets.txt
- Mechanism: Nmap connects to the port and sends a series of probes (NULL, Generic, Help, SSL). It then captures the response and uses Regular Expressions (Regex) to match it against known signatures.

Step 2: Protocol Specific Verification (Manual)

For complex services (like the HashiCorp Consul Agent on Port 8500), we used curl to verify the API status code.

- Observation: HTTP/1.1 200 OK (Confirmed Active Agent).

Step 3: Classification

Detected services were categorized into Infrastructure, Application, and Data layers to map the architectural topology.

---

4. 📊 Service Inventory Findings

The following software stack was identified across the 15 active hosts.

🟢 Infrastructure & Remote Access

Service	Port(s)	Version Detected	Deployment Scope
OpenSSH	22	8.9p1 Ubuntu 3ubuntu0.1	Standardized: Deployed on all 15/15 nodes.
Consul Agent	8500	Golang net/http	High Usage: Detected on 10+ nodes (Service Mesh).

🟡 Web Servers & Runtimes

Service	Port(s)	Version Detected	Role
OpenResty	80, 443	*(Signature Masked)*	Primary Ingress / Load Balancer (Nginx-based).
Apache	8085	httpd 2.4.52	Legacy/Specific Application Server.
Uvicorn	6001, 8000	Python/ASGI	Python Application Backend (FastAPI/Django).

🔴 Data & Storage (Critical Assets)

Service	Port(s)	Version Detected	Risk Note
Redis	9111	Key-Value Store	CRITICAL: Database Exposed. Default port is usually 6379, here running on non-standard 9111.
MinIO	9000	MinIO Object Storage	WARNING: S3-compatible storage API exposed.
MinIO Console	9001	MinIO Console	WARNING: Administrative dashboard exposed.

---

5. 🔍 Reproducibility & Output

- Raw Evidence: The full fingerprinting logs are preserved in logs/scan_results.txt.
- Accuracy Check: Manual verification of the Redis service on Port 9111 confirmed the banner matches the scan result, validating the -sV accuracy.