S3M Nmap Discovery Module – Architecture Specification

Purpose

Enhance the visibility and profiling capabilities of S3M by integrating **configurable Nmap scanning**. The module performs active reconnaissance on network devices to extract metadata such as:

- Open ports
- Service versions
- OS fingerprints
- Known vulnerabilities (via NSE)

The results are used for risk scoring, asset classification, and attack surface mapping.

Architectural Overview

```
[Frontend (Web GUI)] --> [Backend API (Flask / FastAPI)] --> [Scan Scheduler] --> [Nmap Wrapper] --> [DB Parser + Storage]
```

Configuration Structure

Expose scanning profiles via web interface, allowing users to toggle scan depth per subnet or target range.

Scan Modes

Mode	Options Enabled	Notes
Light	`-sS -T3`	Fast SYN scan for alive hosts and open ports
Standard	`-sS -sV -O`	Adds service version and OS detection
Deep	`-sS -sV -O -A --script=default`	Aggressive mode with traceroute and basic scripts
Vuln Scan	`-sS -sV --script=vuln`	Targets known vulnerabilities via NSE scripts

> All scans support CIDR notation and host lists from DB queries.

High Value NSE Scripts for S3M

Script	Why It's Useful
`smb-os-discovery`	Gets Windows version + domain info from SMB

`nbstat`	Gathers NetBIOS name, domain, and MAC from Windows systems
`http-title`	Reveals web admin panels on devices (routers, IoT)
`ftp-anon`	Detects open FTP with anonymous login — posture risk
`snmp-info`	Extracts system name, uptime, device model if SNMP is open
`broadcast-netbios-ns`	Locates Windows machines via broadcast
`broadcast-dhcp-discover`	Detects DHCP servers on segment
`dns-service-discovery`	Identifies services registered via DNS-SD (IoT use)
`vuln`	NSE scripts for common vulnerabilities. Slow, noisy, likely to trigger alarms

Example Nmap Command Templates

Light:

```
nmap -sS -T3 <target>
```

- `-sS` : stealth SYN scan
- `-T3` : default scan timer

Standard:

```
nmap -sS -sV -O <target>
```

- `-sV` : service version detection
- `-O` : OS detection

Deep:

```
nmap -sS -sV -O -A --script=default <target>
```

- `-A` : aggressive mode (includes OS, traceroute, script scan, etc.)

Windows Device Profiler:

Scripts:

- `smb-os-discovery`
- `nbstat`
- `smb-enum-users` *(optional)*

Ports Required:

- TCP 445
- TCP 139

```
nmap -p 139,445 --script=smb-os-discovery,nbstat,smb-enum-users <target>
```

Output Highlights:

- OS version

- NetBIOS name
- Workgroup/Domain
- Possible list of usernames (if not blocked)

HTTP Panel & Web Service Discovery

Scripts:

- `http-title`
- `http-server-header`
- `http-methods`
- `http-robots.txt` *(optional)*

Ports Required:

- TCP 80
- TCP 443
- TCP 8080

```
nmap -p 80,443,8080 --script=http-title,http-server-header,http-methods <target>
```

Output Highlights:

- Web UI title (e.g. “TP-Link Admin Panel”)
- Server stack info (nginx, Apache, embedded httpd)
- Allowed HTTP verbs (e.g. PUT, TRACE)

SNMP Discovery (Read-Only)

Scripts:

- `snmp-info`
- `snmp-interfaces` *(optional)*

Ports Required:

- UDP 161

Command:

```
nmap -sU -p 161 --script=snmp-info <target>
```

Output Highlights:

- Device model
- System name
- Uptime
- Interfaces

Broadcast Discovery (Local Segment Only)

Scripts:

- `broadcast-netbios-ns`
- `broadcast-dhcp-discover`
- `broadcast-avahi-dos` *(for detection only — don't run this unless you know what it does)*

Ports Required:

- UDP broadcast (no target IP needed)

Command:

```
nmap --script=broadcast-netbios-ns,broadcast-dhcp-discover
```

Output Highlights:

- Responding hosts on local subnet
- NetBIOS names
- DHCP offers

IoT & Smart Device Discovery

Scripts:

- `dns-service-discovery`
- `upnp-info`
- `ssdp-discover`

Ports Required:

- UDP 1900
- Multicast / broadcast

Command:

```
nmap --script=upnp-info,dns-service-discovery,ssdp-discover <target>
```

Output Highlights:

- Device type and model
- Available services
- Location of control URLs

General Vulnerability Scan

Scripts:

- `vulners` *(requires internet connection; CVE mapping)*
- `vuln` *(runs all scripts tagged as vulnerability checks)*

Ports Required:

- Depends on target (use open port list from earlier scans)

```
nmap -sS -sV --script=vuln <target>
```

Or for high-fidelity mapping:

```
nmap -sV --script vulners <target>
```

Output Highlights:

- Correlates known software versions with known CVEs
- Works well on printers, web servers, VoIP gear, and Linux systems

For Advanced Vulnerability Scanning Capabilities (Optional):

Windows Vulnerability Scanner

Scripts:

- `smb-vuln-ms17-010` *(EternalBlue)*
- `smb-vuln-ms08-067`
- `smb-vuln-cve2009-3103`

Ports Required:

- TCP 445
- TCP 139

Command:

```
nmap -p 139,445 --script=smb-vuln-ms17-010,smb-vuln-ms08-067,smb-vuln-cve2009-3103 <target>
```

Output Highlights:

- Checks for critical unpatched SMB vulnerabilities in legacy Windows devices
- Flags systems susceptible to worms or RCE

Web Vulnerability Scanner

Scripts:

- `http-vuln-cve2017-5638` *(Apache Struts RCE)*
- `http-vuln-cve2013-7091` *(phpMyAdmin code execution)*
- `http-vuln-cve2014-3704` *(Drupal SQLi)*

Ports Required:

- TCP 80, 443, 8080

Command:

```
nmap -p 80,443,8080 --script=http-vuln-cve2017-5638,http-vuln-cve2013-7091,http-vuln-cve2014-3704 <target>
```

Output Highlights:

- Application-level vulnerabilities in common CMS or admin interfaces

SSL / TLS Weakness Scanner

Scripts:

- `ssl-ccs-injection`
- `ssl-heartbleed`
- `ssl-poodle`

Ports Required:

- TCP 443, 8443

Command:

```
nmap -p 443,8443 --script=ssl-ccs-injection,ssl-heartbleed,ssl-poodle <target>
```

Output Highlights:

- Checks for protocol-level SSL/TLS vulnerabilities
- May be useful in validating IoT or embedded devices running outdated stacks

---

Suggested Frontend Mappings

Toggle Name	Scripts	Default(Toggle)	Profile
“Windows Device Profiler”	`smb-os-discovery`, `nbstat`	Off	Deep
“HTTP Admin Discovery”	`http-title`, `http-server-header`	On	Standard
“SNMP Device Info”	`snmp-info`	Off	NAS
“Local Broadcast Discovery”	`broadcast-netbios-ns`, `broadcast-dhcp-discover`	On	Discovery Only
“IoT Fingerprinting”	`upnp-info`, `dns-service-discovery`	Off	IoT

Output Handling

- Output stored in **XML** or **Grepable** format.
- Parsed via `libnmap` or custom Python parser.
- Normalized fields include:
  - `ip_address`, `mac_address`, `hostname`
  - `os_name`, `os_accuracy`
  - `open_ports[]`, `services[]`
  - `cve_hits[]` (if vuln scan)

Suggested DB Table

```
CREATE TABLE nmap_scan_results (
    id SERIAL PRIMARY KEY,
    ip_address TEXT,
    mac_address TEXT,
    hostname TEXT,
    os_name TEXT,
    os_accuracy INTEGER,
    open_ports JSONB,
    services JSONB,
    vuln_results JSONB,
    scan_profile TEXT,
    scan_time TIMESTAMPTZ DEFAULT now()
);
```

Scan Triggering

Trigger scans by:

- Scheduled task (e.g. nightly per subnet)
- Event based (Integration with other modules)

Security Considerations

- Scans should be rate-limited and logged.
- No aggressive scans (`-A`) on critical infrastructure by default.
- Web UI must indicate scan profile severity.

Deliverables

Component	Responsibility
Nmap scanner wrapper	Dev
Parser + DB writer	Dev
Scan scheduler service	Dev
Frontend config toggles	UI Dev
Profile-to-command mapping	Architect (this doc)

To-Do / Dev Notes

- Consider adding in nmap options instead of using templates
- A JSON/YAML representation of nmap profiles for modularity
- Allow per-subnet scan configuration
- Prevent duplicate concurrent scans
- Add retry logic for flaky hosts
- Store raw output alongside parsed results
- Correlate with RADIUS/SNMP/OUI data