Scanning Highly Sensitive Networks v3

+1 801-784-2052 // justin@controlthings.io

Scanning Highly Sensitive Networks

Assessment goals

- Identify additional information about each system/device
- Identify vulnerabilities and weaknesses in each system/device
- Avoid causing devices and system services to crash

Scoping information needed

- A list of in-scope and out-of-scope IP addresses
- Credentials and protocol needed to log into each asset
- Mutually agreed upon escalation procedures if problems arise

Network Assessments in ICS

Always do an architecture review SAR and network capture assessment first

- Get to know the environment and teams first
- Identifies which systems and networks are more sensitive/dangerous
- Helps you identify resource needs like skill sets, tools, logistics
- Refines scoping for pentest

Share SAR and network capture results/reports with assessment team

- Gives them a starting point
- Allows them to optimize their tests
- Lets them confirm findings

General guidance for network assessment team

- Know **EXACTLY** how each features in your tools work
- Always attempt to minimize the number of packets/interactions used
- Include network admins and be transparent

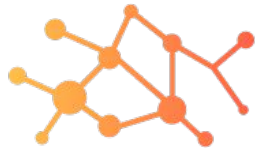
DNS Interrogation

Level of Effort: Low (1-4 hours)

Task Description: Use a tool query hostname for all in-scope IP addresses.

Task Goal: Help identify/verify each IP address for the following tasks.

Use But Don't Abuse DNS



DNS might not be used in all ICS subnets

- Purdue levels 0 and 1 often use static IPs and no DNS
- Purdue level 2 might not even exist, and is hit/miss for DNS when it does
- Purdue level 3 usually has DNS servers via Active Directory

If available, minimize its use

- Do reverse lookups for IPs only once
- Don't run port scans with DNS lookups (minimize packets)
- Don't brute force DNS entries (avoid tools like dnsenum and dnsrecon)
- When in doubt, just ask for a DNS server config dump

Using nmap for DNS lookups (with no discovery or scanning)

```
nmap -sL 192.168.0.1/24
nmap -sL --dns-servers <ip> 192.168.0.1/24
```
(if not using DHCP)

Port Scanning

Level of Effort: Low (1-4 hours)

Task Description: Use a tool query TCP ports for all in-scope IPs using full TCP handshakes, being careful to minimize total traffic sent and to only communicate with one port at a time for each IP. We avoid scanning UDP ports except in specific and limited scenarios.

Task Goal: Identify which ports are open while avoiding actions that might crash legacy embedded devices.

Dangers of Port Scanning by Risk

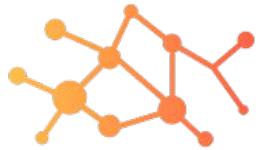
Nmap features which cause the most problems with legacy embedded devices

- OS Fingerprinting
  - By far the most likely cause of crashed legacy embedded devices
  - Don't use the `-O` or `-A` flags in nmap
- SYN scans (half-open)
  - Default when using nmap with sudo or running it as root
  - Not proper RFC behavior, so only mature TCP/IP stacks handles this properly
  - Always specify `-sT` in your scans to avoid this accident
- Scanning too fast
  - yes, the defaults in nmap are too fast for legacy embedded devices
  - Or use nmap's `--scan-delay 0.1` or `--max-parallelism 1` to scan 1 port at a time per host

Nmap features which cause the most problems with services on all systems

- Scanning UDP ports with null payloads
  - Don't use the `-sU` option in nmap
- Service fingerprinting usually safe, but can occasionally cause problems
  - Use nmap's `-sV` selectively on new subnets
  - Or use nmap's `--script=banner`

nmap Suggestions



- Always run nmap with sudo with -sT
  - nmap rarely tells you its needed (only says it for `-O`)
  - Requirements vary from OS to OS
  - Required for all ICMP functions
  - Required for OS fingerprinting
  - Required for some NSE scripts
  - If you don't believe me, make and diff some pcaps
- It is always good practice to use `-v` when scanning

Low Risk Portscans

Ping sweeps

```
sudo nmap -v -n -PE -sn --max-retries 0 <IP/CIDR>
```

- **Risk: 1/10** (only sends ICMP echo request, so 1 packet out per IP)
- **Value: 1/10** (only shows basic list of connected systems, better if in same subnet)
- **Bonus Value: 2/10** (if performed on same subnet, will show MAC address with vendor decoded)

Slow port scans of 20 most common IT services

```
sudo nmap -v -n -PE -sT --scan-delay 0.1 --max-retries 0 --top-ports 20 <IP/CIDR>
```

- **Risk: 2/10** (1 ICMP packet per IP for discovery, then 1 SYN packet per port probe, then 1 ACK and 1 RST to close)
- **Value: 3/10** (tests for most common TCP servers...but not sensitive/proprietary protocols)
- **Top Ports**: `sort -r -k3 /usr/share/nmap/nmap-services | grep '/tcp' | column -t | less -S -N --use-color`

Port sweeps of single TCP ports

```
sudo nmap -v -n -Pn -sT --max-retries 0 -p <single TCP port> <IP/CIDR>
```

- **Risk: 3/10** (no discovery, only sends 1 SYN packet per port)
- **Value: 3/10** (only shows basic list of connected systems with that port open)

Slow port Scans of limited number TCP ports

```
sudo nmap -v -n -PE -sT --scan-delay 0.1 --max-retries 0 -p ??? <IP/CIDR>
```

- **Risk: 4/10** (1 ICMP packet per IP for discovery, then 1 SYN packet per port probe, then 1 ACK and 1 RST to close)
- **Value: 4/10** (tests for whatever services you specify, we suggest all TCP ports seen in network your captures)

Medium to High Risk Port Scans

Scanning all TCP ports

```
sudo nmap -v –n -PE -sT --max-parallelism 1 --max-retries 0 -p- <IP/CIDR>
```

- **Risk: 5/10** (scans each host's TCP ports as fast as possible, but not in parallel)
- **Value: 5/10** (scans all possible TCP ports)

Scanning and fingerprinting all TCP ports

```
sudo nmap -v –n -PE -sT --max-parallelism 1 -sV --script=default -p- <IP/CIDR>
```

- **Risk: 6/10** (we are now interacting with the TCP services themselves)
- **Value: 7/10** (service fingerprints are a HUGE increase in value)
- **NSE Scripts:** `grep ^categories /usr/share/nmap/scripts/* | grep default`

Fast Scanning and fingerprinting all TCP ports with extra discovery

```
sudo nmap -v –n -PE -sT -sV --script="discovery and safe" -p- <IP/CIDR>
```

- **Risk: 8/10** (taking maximum risk without being totally reckless)
- **Value: 8/10** (the extra discover scripts will add some value)
- **NSE Scripts:** `grep ^categories /usr/share/nmap/scripts/* | grep discovery | grep safe`

I want to break something!!!

```
sudo nmap -v -n -sS -sU -T4 -A --script="discovery" -p- <IP/CIDR>
```

- **Risk: 10/10** (highly likely to cause problems on legacy software/hardware, and maybe some modern one too)
- **Value: 10/10** (useful for testing new products in lab settings and reporting problems to vendors)

Technology Fingerprinting

Level of Effort: Low (1-4 hours)

Task Description: Use various tools to identify which services are behind each open port.

Task Goal: Help identify server software and software version.

Build an enhanced asset inventory

Because network scans are more limited, we need data from more sources

Combine datasets from

- Security architecture review
- Network capture assessment
- DNS interrogation
- Port scanner

Seek to use engineering tools from vendors to gather more information

Identify which protocols might provide more needed information

Protocol Enumeration

Level of Effort: Low (1-4 hours)

Task Description: Use a tool to discover additional information about non-ICS services which are safe to enumerate.

Task Goal: Gather additional information about a device or service.

Enumerate Slowly Avoiding ICS Protocols

Do a port sweep for SNMP and carefully enumerate it

Never enumerate ICS protocols in production

- At most, do device ID queries if ICS protocols support that feature
- Tag read requests seem benign, but read rates could cause failures
- In rare cases, tag reads could trigger other actions in controller logic

Vulnerability Scanning

Level of Effort: Low (1-4 hours)

Task Description: Use a vulnerability scanner to log into each system/device, retrieve the list of installed software with versions, and query a vulnerability database without doing active vulnerability probes to open ports.

Task Goal: Identify what known vulnerabilities exist in each system or device.

How Vulnerability Scanners Work

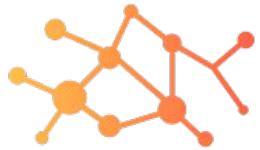
1. **Port Scanning**: to find open services
   - Basically like what nmap does
   - Risks discussed with nmap are just as great with vulnerability scanners
2. **Service Fingerprinting**: to identify each service
   - Retrieves service banner ID from server responses
   - Sometimes sends special probes for some ports/services
   - Most vulnerabilities are identified this way
3. **Vulnerability Probing**: for harder-to-find vulnerabilities
   - This technique is used to find a smaller percentage of vulnerabilities
   - This can be VERY dangerous for highly sensitive services in ICS/OT/IoT/IIoT
4. **Authenticated Scanning**: to pull information straight from OS configs
   - Logs in if correct credentials are provided
   - Often pulls users, installed software, patch levels, and security configs
   - Can also pull listening ports via the netstat command or similar
5. **Custom Audit Checks**: easy method to create your configuration checks
   - Script virtually any OS or application check desired

Low Risk Authenticated Scans with Nessus

- Decreases risk by removing TCP/UDP port scans and vulnerability probes
- To use Nessus audit checks:
  - In Nessus, create a new scan profile
  - Disable all Nessus TCP, SYN, UDP, and SNMP port scans
  - Leave Netstat and Ping port scans open
  - Disable all Nessus plugins except Windows/Linux compliance checks
  - In Preferences, configure the compliance checks to use any third party or custom made audit files (windows security policy or plain text file values)
- Create a new scan and tell it to use your new profile
- Some older scan profiles were made for ICS by Digital Bond
  - Part of their Bandolier Project
  - Nessus has changed their audit language, so they would need updating

Conclusion

Author Contact Information



Justin Searle

training & opensource: justin@controlthings.io

consulting & testing: justin@inguardians.com

cell: 801-784-2052

twitter: @meeas

LinkedIn: www.linkedin.com/in/meeas

GitHub: github.com/meeas, github.com/ControlThingsTools

ControlThings Project:

https://www.controlthings.io