Advanced network scanning with Nmap 6

Henri DOREAU
henri.doreau@gmail.com

13th LSM - Geneva 2012
Outline

1. **Project presentation**
   - Introduction

2. **Nmap Scripting Engine**
   - Presentation
   - Internals
   - Usage

3. **Nmap 6 new features**
   - IPv6 support
   - Performance improvements
   - Companion tools
   - NSE

4. **Ongoing developments**
   - Upcoming features
   - Project
Outline

1. **Project presentation**
   - Introduction

2. **Nmap Scripting Engine**
   - Presentation
   - Internals
   - Usage

3. **Nmap 6 new features**
   - IPv6 support
   - Performance improvements
   - Companion tools
   - NSE

4. **Ongoing developments**
   - Upcoming features
   - Project
Nmap Security Scanner

Full-featured Network scanner

- Port scanner
- Version and OS fingerprinting
- Lua scripting engine
- Companion tools (zenmap, ncat, nping, ndiff...)
Nmap Security Scanner

Vibrant community

- Fingerprint DBs
- CPEs
- Scripts and NSE libraries
Nmap Security Scanner

Hollywood movie star
Outline

1. Project presentation
   - Introduction

2. Nmap Scripting Engine
   - Presentation
   - Internals
   - Usage

3. Nmap 6 new features
   - IPv6 support
   - Performance improvements
   - Companion tools
   - NSE

4. Ongoing developments
   - Upcoming features
   - Project
Introduction

**Built-in lua scripting engine**
- Network exploration
- Sophisticated version detection
- Vulnerability detection
- Scan results post-processing
NSE development

Script collection growth
Script phases

Four execution modes
- Prerules
- Service
- Host
- Postrules

NSE Pre-scan
1. Host enumeration
2. Host discovery
3. Reverse DNS resolution
4. Port scan
5. Version detection / RPC grind
6. OS fingerprinting
7. Traceroute
8. Script scan
9. Output

NSE Post-scan
Script structure

When to run?

```
hostrule = function(host)
    return host.directly_connected
end
```

```
portule = shortport.http
```

⇒ script can have several rule and action functions
### Sample output

**Nmap scan report for scanme.nmap.org (74.207.244.221)**

<table>
<thead>
<tr>
<th>PORT</th>
<th>STATE</th>
<th>SERVICE</th>
<th>VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>22/tcp</td>
<td>open</td>
<td>ssh</td>
<td>OpenSSH 5.3p1 Debian 3ubuntu7</td>
</tr>
<tr>
<td>80/tcp</td>
<td>open</td>
<td>http</td>
<td>Apache httpd 2.2.14 ((Ubuntu))</td>
</tr>
</tbody>
</table>

HTTP title: Go ahead and ScanMe!

Service Info: OS: Linux; CPE: cpe:/o:linux:kernel

**Host script results:**

```
<p>| firewalk:                  |</p>
<table>
<thead>
<tr>
<th>HOP</th>
<th>HOST</th>
<th>PROTOCOL</th>
<th>BLOCKED PORTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>192.168.0.15</td>
<td>tcp</td>
<td>139</td>
</tr>
<tr>
<td>10</td>
<td>64.62.250.6</td>
<td>tcp</td>
<td>135,445</td>
</tr>
</tbody>
</table>
```
**Design**

**NSE parallelism**
- Single nmap thread
- lua coroutines

⇒ Lightweight and efficient non-blocking mechanism
⇒ Script writers get parallelism for free
⇒ No concurrent memory access concerns ever
Adaptive workflow

Two ways to invoke scripts

Point and shoot

```
nmap --script samba-vuln-cve-2012-1182 <target>
nmap --script +mongodb-info -p80 <target>
```

⇒ No silent dependencies

Aim oriented

```
nmap --script "http-* and not brute" <target>
```
### Script categories

- **default**
- **intrusive**
- **external**
- ...

<table>
<thead>
<tr>
<th>NSEE Doc Index</th>
<th>NSEE Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Categories</strong></td>
<td></td>
</tr>
<tr>
<td>auth</td>
<td>broadcast</td>
</tr>
<tr>
<td>brute</td>
<td>default</td>
</tr>
<tr>
<td>discovery</td>
<td>dos</td>
</tr>
<tr>
<td>exploit</td>
<td>fuzzer</td>
</tr>
<tr>
<td>intrusive</td>
<td>malware</td>
</tr>
<tr>
<td>safe</td>
<td>version</td>
</tr>
<tr>
<td>vun</td>
<td></td>
</tr>
<tr>
<td><strong>Scripts (395)</strong></td>
<td></td>
</tr>
<tr>
<td>acarsd-info</td>
<td>address-info</td>
</tr>
<tr>
<td>afp-brute</td>
<td>afp-competitions</td>
</tr>
<tr>
<td>afp-competitions</td>
<td>afp-isp</td>
</tr>
<tr>
<td>afp-path-vuln</td>
<td>afp-serverinfo</td>
</tr>
<tr>
<td>afp-showmount</td>
<td></td>
</tr>
<tr>
<td>aip-auth</td>
<td></td>
</tr>
<tr>
<td>aip-brute</td>
<td></td>
</tr>
<tr>
<td>aip-headers</td>
<td></td>
</tr>
<tr>
<td>aip-methods</td>
<td></td>
</tr>
<tr>
<td>aip-request</td>
<td></td>
</tr>
<tr>
<td>amqp-info</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scripts</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>acarsd-info</td>
<td>Retrieves information from a listening acarsd daemon. Acarsd decodes ACARS (Aircraft Communication Addressing and Reporting System) data in real time. The information retrieved by this script includes the airframe version, API version, administrator e-mail address and listening frequency.</td>
</tr>
<tr>
<td>address-info</td>
<td>Shows extra information about IPv4 addresses, such as embedded MAC or IPv4 addresses when available.</td>
</tr>
<tr>
<td>afp-brute</td>
<td>Performs password guessing against Apple Filing Protocol (AFP).</td>
</tr>
<tr>
<td>afp-isp</td>
<td>Attempts to get useful information about files from AFP volumes. The output is intended to resemble the output of lsa.</td>
</tr>
<tr>
<td>afp-path-vuln</td>
<td>Detects the Mac OS X AFP directory traversal vulnerability, CVE-2010-0533.</td>
</tr>
<tr>
<td>afp-serverinfo</td>
<td>Shows AFP server information. This information includes the server's hostname, IPv4 and IPv6 addresses, and hardware type (for example Macmini or MacBookPro).</td>
</tr>
<tr>
<td>afp-showmount</td>
<td>Shows AFP shares and ACLs.</td>
</tr>
<tr>
<td>aip-auth</td>
<td>Retrieves the authentication scheme and realm of an AJP service (Apache JServ Protocol) that requires authentication.</td>
</tr>
<tr>
<td>aip-brute</td>
<td>Performs brute force passwords auditing against the Apache JServ protocol. The Apache JServ Protocol is commonly used by web servers to communicate with back-end Java application server containers.</td>
</tr>
<tr>
<td>aip-headers</td>
<td>Performs a HEAD or GET request against either the root directory or any optional directory of an Apache JServ Protocol server and returns the server response headers.</td>
</tr>
<tr>
<td>aip-methods</td>
<td>Discovers which options are supported by the AJP (Apache JServ Protocol) server by sending an OPTIONS request and lists potentially risky methods.</td>
</tr>
<tr>
<td>aip-request</td>
<td>Requests a URI over the Apache JServ Protocol and displays the result or stores it in a file. Different AJP methods such as GET, HEAD, TRACE, PUT or DELETE may be used.</td>
</tr>
<tr>
<td>amqp-info</td>
<td>Gathers information (a list of all server properties) from an AMQP (advanced message queuing protocol) server.</td>
</tr>
</tbody>
</table>

See [http://nmap.org/nsedoc](http://nmap.org/nsedoc)
Outline

1. Project presentation
   - Introduction

2. Nmap Scripting Engine
   - Presentation
   - Internals
   - Usage

3. Nmap 6 new features
   - IPv6 support
   - Performance improvements
   - Companion tools
   - NSE

4. Ongoing developments
   - Upcoming features
   - Project
Full IPv6 support

Long standing wish

- All features (provided it makes any sense)
- All supported platforms
Full IPv6 support

Long standing wish

- All features (provided it makes any sense)
- All supported platforms

YEAH!!!
Brand new OS fingerprinting engine

Innovative approach: machine learning techniques
- Reduced dataset
- Increased adaptiveness
- Very accurate

⇒ See http://nmap.org/book/osdetect
IPv6 support

<table>
<thead>
<tr>
<th>Honestly, who cares?</th>
<th></th>
</tr>
</thead>
</table>
IPv6 support

Honestly, who cares?

The future is already there!
Enhanced performances

Three main axis of improvement

- Memory footprint
- High performance and scalable I/O notification facilities
- Application-specific optimizations (NSE)

cf. *Scanning the Internet*, by Fyodor
Nping

Reimplementation of the venerable hping2

- Modern, high performance tool
- Leverages nmap libraries
- Provides new packet crafting classes to nmap
Nping Echo mode

Replacement for ping+tcpdump

1. nping in server mode on target
2. client probes the target
3. server returns captured probes to the client(s) as encrypted payloads
Finally: actual network maps from the network mapper!
Better web scanning

**Big focus on web technologies**

- Pipelining
- Built-in web crawler
- Caching
- Web-specific security checks
## NSE frameworks

**Implemented as NSE libraries**

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>brute</strong></td>
<td>Parallel network authentication cracking module.</td>
</tr>
<tr>
<td><strong>credentials</strong></td>
<td>Leverage and report discovered credentials.</td>
</tr>
<tr>
<td><strong>vulns</strong></td>
<td>Consistent vulnerability reports and efficient post-processing.</td>
</tr>
</tbody>
</table>
Outline

1. Project presentation
   - Introduction

2. Nmap Scripting Engine
   - Presentation
   - Internals
   - Usage

3. Nmap 6 new features
   - IPv6 support
   - Performance improvements
   - Companion tools
   - NSE

4. Ongoing developments
   - Upcoming features
   - Project
Upcoming: web scanning

Continued effort on HTTP

- Implement latest performance-related protocols and paradigms
- WebSocket mode to ncat
Expand the role and features of NSE

- Leveraging native libraries from lua
- NSE-based port scanning
- Re-implementing older code within NSE
- Adapting NSE to the companion tools
Upcoming: misc

but also...

- Combining IP v4/v6 scans
- Improving scalability
- Scanning through proxies
- Remote checks through authenticated SSH connections
- Updater
Get involved!

Your own awesome idea!

...and code? ;}
Increasing development pace

- 2011 was the most active year ever in the project history! (ohloh.net).
- 8th consecutive Google Summer of Code
Happy birthday nmap!

15\textsuperscript{th} birthday this year (Sept. 1\textsuperscript{st})
Questions?

http://nmap.org

nmap-dev@insecure.org (it’s cool, join!)