THE NMAP SCRIPTING ENGINE

David Fifield <david@bamsoftware.com> 6 February 2010

This document shows how to use the Nmap Scripting Engine (NSE)—the combination of an embedded Lua interpreter, networking libraries, and the Nmap scanning apparatus. It tells you how to use the scripting engine in your scans, and gives advice on writing your own scripts.

About Nmap

Nmap is a free network security scanner with many features. The web page http://nmap.org/download.html has source packages and installers for many operating systems. Most free Unixes include a package called `nmap`. To get the very latest, check out from Subversion:

```
$ svn co --username guest --password "" svn://svn.insecure.org/nmap
$ cd nmap
$ ./configure && make
```

In its most basic usage, Nmap will scan a host and report the state of its ports. Try this example:

```
$ nmap scanme.nmap.org
Nmap scan report for scanme.nmap.org (64.13.134.52)
PORT      STATE  SERVICE
25/tcp    closed smtp
53/tcp    open   domain
80/tcp    open   http
113/tcp   closed auth
31337/tcp closed Elite
Nmap done: 1 IP address (1 host up) scanned in 8.62 seconds
```

Using NSE

Nmap can get more information from the remote system by running scripts against it. A script is a program written in the embedded Lua programming language. Nmap ships with dozens of scripts that do various network discovery tasks. (See http://nmap.org/nsedoc/ for the full list). Scripts have access to Nmap's scan results and several networking libraries.

To enable NSE, add the `-sC` or `--script` option. `-sC` enables all the scripts in the `default` category, those that are fast, safe, and generally useful. Other categories are `auth`, `discovery`, `external`, `intrusive`, `malware`, `safe`, `version`, and `vuln`. The `-A` (aggressive) option also implies `-sC`.

Use `--script` to select specific scripts. For example

```
--script=http-date
```

You can select one or more categories, and use Boolean expressions.

```
--script=default,safe,malware
--script='discovery and not intrusive'
```

Shell-like wildcards work as well.

```
--script='http-* and not http-enum'
```

Some scripts can take arguments through the `--script-args` option. See http://nmap.org/nsedoc/ for details.

```
$ nmap --script=safe scanme.nmap.org
Nmap scan report for scanme.nmap.org (64.13.134.52)
PORT      STATE  SERVICE
25/tcp    closed smtp
53/tcp    open   domain
70/tcp    closed gopher
80/tcp    open   http
| http-headers:
|   Date: Sun, 31 Jan 2010 19:44:04 GMT
|   Server: Apache/2.2.3 (CentOS)
|   Accept-Ranges: bytes
|   Content-Length: 739
|   Connection: close
|   Content-Type: text/html; charset=UTF-8
|
|_  (Request type: HEAD)
|_http-date: Sun, 31 Jan 2010 19:44:10 GMT; +6s from local time.
|_html-title: Go ahead and ScanMe!
113/tcp   closed auth
31337/tcp closed Elite

Host script results:
| asn-query:
| BGP: 64.13.128.0/18 | Country: US
|   Origin AS: 8121 - TCH - TCH Network Services
|_    Peer AS: 1299 2516 3356 4565 4657 19080
| whois: Record found at whois.arin.net
| netrange: 64.13.134.0 - 64.13.134.63
| netname: NET-64-13-143-0-26
| orgname: Titan Networks
| orgid: INSEC
|_country: US stateprov: CA
Nmap done: 1 IP address (1 host up) scanned in 34.27 seconds
```

Anatomy of a script

```
description = [[
Gets the date from HTTP-like services. Also prints how much the date
differs from local time. Local time is the time the HTTP request was
sent, so the difference includes at least the duration of one RTT.
]]

---
-- @output
-- 80/tcp open  http
-- |_ http-date: Thu, 23 Jul 2009 23:15:57 GMT; -6s from local time.
-- 80/tcp open  http
-- |_ http-date: Wed, 17 Jan 2007 09:29:10 GMT; -2y187d13h46m53s from

author = "David Fifield"

license = "Same as Nmap--See http://nmap.org/book/man-legal.html"

categories = {"discovery", "safe"}

require("http")
require("shortport")
require("stdnse")

portrule = shortport.port_or_service({80, 443, 631, 8080},
       {"http", "https", "ipp", "http-alt"})

action = function(host, port)
       -- Get the local date in UTC.
       local request_date = os.date("!*t")
       local response = http.get(host, port, "/")
       if not response.status or not response.header["date"] then
              return
       end

       local response_date = http.parse_date(response.header["date"])
       if not response_date then
              return
       end

       -- Should account for estimated RTT too.
       local diff = stdnse.format_difftime(response_date, request_date)

       return string.format("%s; %s from local time.",
              response.header["date"], diff)
end
```

The `description` variable tells what the script does. It may be several paragraphs.

Comments that start with three dashes are NSEDoc documentation. Special tags like `@output` and `@usage` create sections in the online documentation.

`author` and `license` fields define more script metadata.

The categories the script belongs to.

External library imports.

The `portrule` function defines which ports the script will run on. The `shortport` library has functions for a few common patterns. Scripts not attached to a specific port use a `hostrule` instead.

The `action` function is where the script does its work. The function receives `host` and `port` tables from Nmap (`hostrule` scripts get only `host`).

This script retrieves a web page with the `http` library, gets the value of the `Date` header field, and formats the difference from local time with a standard library function.

Scripts return their results as a string. Return `nil` if there is nothing to report.

Tips for script development

It helps to use an existing script as a model for a new one. Read the *Nmap Network Scanning* chapter on NSE (available online) and the *Programming in Lua* manual—Lua is not hard to learn for NSE purposes. When making changes in an Nmap source directory, use the `--datadir` option to force Nmap to use the files in the local directory instead of any that are installed system-wide.

```
$ nmap --datadir . --script=test-script target
```

After installing a new script, remember to run Nmap with the `--script-updatedb` option once, to register the new script in the script database.

```
$ nmap --script-updatedb
```

Run Nmap with the `-d` (debug) option to see a backtrace when a script has an error.

When your new script is finished, send it to the Nmap mailing list at nmap-dev@insecure.org. That list is also the place to send bug reports.

Resources

Nmap Network Scanning chapter on NSE http://nmap.org/book/nse.html
Online NSEDoc documentation for every script and library http://nmap.org/nsedoc/
nmap-dev@insecure.org mailing list http://seclists.org/nmap-dev/
Programming in Lua http://lua.org/manual/