Nmap presentation for Project 7

By Matthew Kunzman

Volun-told

- Paul made me do this
- I did this today, so that’s why the slides suck

Don’t forget UDP scanning

- nmap –sV scanme.nmap.org
- nmap –sV –sU scanme.nmap.org

Don’t forget IP v6

- nmap scanme.nmap.org
- nmap -6 scanme.nmap.org

Source port manipulation

- Sometimes you'll get a different result
- Some programs respond differently to different source ports
- Examples
  - Bypassing Ipsec - IPsec filters shipped with Windows 2000 and WinXP allow all TCP and UDP traffic using source port 88 (Kerberos)
  - ZoneAlarm Firewall allowed any UDP traffic over port 53 for a while
  - MAC OS Tiger allowed DHCP (67) even with "Block UDP Traffic" box checked
- nmap –sS –v –v –PN –g 88 scanme.nmap.org

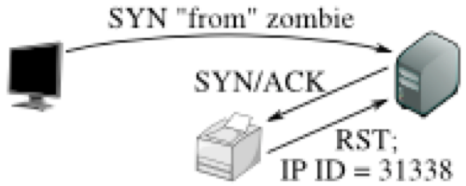
IP ID Idle scanning

- **nmap -sI <Zombie> -Pn -p20-25,110 -r --packet-trace -v <Target>**
- -Pn is necessary for stealth, otherwise ping packets would be sent to the target from Attacker's real address.

Step 1: Probe the zombie's IP ID.

The attacker sends a SYN/ACK to the zombie. The zombie, not expecting the SYN/ACK, sends back a RST, disclosing its IP ID.

Step 2: Forge a SYN packet from the zombie.



The target sends a SYN/ACK in response to the SYN that appears to come from the zombie. The zombie, not expecting it, sends back a RST, incrementing its IP ID in the process.

Step 3: Probe the zombie's IP ID again.

The zombie's IP ID has increased by 2 since step 1, so the port is open!

MAC address spoofing

- --spoof-mac 01:02:03:04:05:06
- --spoof-mac de:ad:be:ef:ca:fe
- --spoof-mac Apple
  - Theres an nmap-mac-prefixes file to find a vendor name and generate a mac address

Proxies or Temp cloud hosts

- Can hide source by going through proxies or spinning up VMs in the cloud

DNS Proxying

- -n disables dns resolution
- --dns-servers will proxy all rDNS queries
- nmap –dns-servers 4.2.2.1,4.2.2.2 –sL scanme.nmap.org

Source routing

- If a route is blocked or causing trouble, try to go around it.
- --ip-options “L 192.168.0.7 192.168.30.9”
  - Requests the packets to be loose routed through those two give way points
  - Use S for strict routing
    - FYI, you’ll have to specify every route along the hop with this

Randomize hosts

- --randomize-hosts

Host spoofing

- -S flag
- Commonly used in DOS attacks
- Can cause innocent hosts to be blocked

Decoys

- **nmap -sS scanme.nmap.org -D 10.0.0.1,10.0.0.2,10.0.0.4**
- Sets up decoys to scan from all of these source IP addresses to hide source origin in traffic and make narrowing down attacker more difficult
- Sometimes set up one noisy scan, and hide true intentions in mess of log files with a second, much slower and targeted scan.

Bypass firewalls blocking ping

- Skip the initial ICMP request to see if host is up
- nmap -sS **-P0** scanme.nmap.org

Slow down to avoid IDS

- -T sneaky
- Or better fine tune it yourself
  - --max-parallelism
  - --min-rtt-timeout
  - --scan-delay
  - --max-hostgroup

NMAP Scripting --- HEAD Section

-- This is a comment. We'll use this to denote the section of the script.
-- HEAD

description = [[This is a multi-line literal string. This is where we offer a simple explanation of what our script aims to do. For instance:Attempts to enumerate "/admins" resource on web apps running on port 3000 and retrieves Admin usernames and passwords.]]

NMAP Scripting -- USAGE

```
---
-- @usage
-- nmap -p 3000 --script rails-admins <host>
--
-- @output
-- PORT STATE SERVICE
-- 3000/tcp open ppp
-- | rails-admins:
-- | <td>Username</td>
-- | <td>Password</td>
--
```

NMAP Scripting – AUTHOR (NOT ME)

author = " Peter Benjamin"

NMAP Scripting – IMPORTS

-- we will be using these imported libraries in the Rule section.
local nmap = require "nmap”

-- we will use these in the action

local http = require "http”

local stdnse = require "stdnse"

NMAP Scripting – RULE (Checks that port 3000 is added)

```
-- RULE
portrule = function(host, port)
  local auth_port = { number=3000, protocol="tcp" }
  local identified = nmap.get_port_state(host, auth_port)
-- "nmap" imported library gives us access to "get_port_state()"
function
  return identified ~= nil -- The operator "~=" is "not equal"
    and identified.state == "open"
    and port.protocol == "tcp"
    and port.state == "open"
end
```

NMAP Scripting – ACTION (GET request to port 3000)

```
--ACTION SECTION
local DEFAULT_URI = “/admins”

-- helper function to check if response contains "password"


local function check_rails_admin(host, port, path)
  local resp = http.get(host, port, path)
  if not http.response_contains(resp, “password”) then
  return false
  end
return resp
end

-- main logic
action = function(host, port)
local vuln_rails = check_rails_admin(host, port, DEFAULT_URI)
local output = {}
if not vuln_rails then
stdnse.print_debug(1,”%s: This does not look like a vulnerable Rails app”, SCRIPT_NAME)
return
else
output = string.match(vuln_rails[“body”], “%<td%>.*%<%/td%>”)
end return output
end
```

Easter egg – l337 speak

```
nmap -oS - scanme.nmap.org
```

```
$tart|ng NMap 5.21 ( http://Nmap.org ) at 2013-09-18 17:45 UTC
Nmap $cAn r3p0rt F0r scanM3.nmaP.oRg (74.207.244.221)
Ho$t 1z Up (0.071z laT3ncy).
Not sh0wN: 998 cl0$Ed p0rt$
POrT   ST4TE $ERV!C3
22/tcp opEn  Ssh
80/tcP 0p3n  HtTp

Nmap d0n3: 1 iP AddrESz (1 h0$t Up) $canNed !n 1.34 secondz
```