

Forum:[Hack Forums](#)

Thread: [{Tutorial} Zenmap - A GUI of Nmap - A Experts Guide](#)

Pr0m3th3us - June 15, 2026, 3:21 am

In this final installment of the series, we're going full offensive. I will cover the Nmap Scripting Engine (NSE) for vulnerability detection, writing your own basic scripts, evading firewalls/IDS, and automating your workflow with custom profiles. This is where Zenmap stops being a "pretty GUI" and starts being a real recon weapon in your toolkit. Check out the last 2 tuts if you haven't already.

[{Tutorial} Zenmap - A GUI of Nmap - A Beginners Guide](#) [{Tutorial} ZenMap - A GUI of Nmap - An Intermediate Guide](#)

The Nmap Scripting Engine (NSE)

NSE is one of the most powerful features of Nmap, and by extension, Zenmap. It allows Nmap to run Lua scripts during a scan to do things way beyond simple port detection. Such as banner grabbing, vulnerability detection, brute forcing, malware detection, and more.

Scripts are categorized:

- Code auth - Authentication bypass attempts
- broadcast - Network discovery via broadcast
- brute - Brute force credential attacks
- default - Scripts run with -sC
- discovery - Service/network enumeration
- dos - Denial of service testing
- exploit - Active exploitation attempts
- external - Scripts that use external services
- fuzzer - Fuzzing for unexpected responses
- intrusive - May crash services, use with caution
- malware - Detects backdoors/malware infections
- safe - Won't crash or disrupt the target
- vuln - Checks for known vulnerabilities
- version - Service version detection helpers

In Zenmap, build a custom command using the Command field directly. Try this against a windows target. This tells Nmap to do version detection (-sV) and then run every script in the vuln category against the open ports it finds.

Code `nmap -sV -Pn --script vuln 192.168.X.X`

If you see something like this Code "Host script results:

`_smb-vuln-ms10-054: false`

`_smb-vuln-ms10-061: Could not negotiate a connection:SMB: Failed to receive bytes: ERROR`

`_samba-vuln-cve-2012-1182: Could not negotiate a connection:SMB: Failed to receive bytes: ERROR"` The (false) is a script returning a definitive negative, the other (Could not negotiate a connection) is an inconclusive error. The latter DOESN'T mean its not vulnerable, it means the script couldn't even test it.

If you find a target with vulnerabilities it will look something like this.

Code PORT STATE SERVICE VERSION

```
80/tcp open http Apache httpd 2.4.29
| http-vuln-cve2017-5638:
|   VULNERABLE:
|     Apache Struts RCE
|       State: VULNERABLE
|       IDs: CVE:CVE-2017-5638
```

Useful Script Combos

Code `nmap -sV -sC 192.168.1.237`Runs version detection + default safe scripts. Good baseline for any host.

Code `nmap --script=smb-vuln* 192.168.1.0/24`Checks every host on the subnet for known SMB vulnerabilities (EternalBlue, etc).

Code `nmap --script=http-enum 192.168.1.237`Enumerates common directories/files on a web server (admin panels, backup files, etc).

Code `nmap --script=ssh-auth-methods,ssh2-enum-algos 192.168.1.237`Pulls SSH configuration details — useful for hardening checks or finding weak auth methods.

Writing Your Own NSE Script

NSE scripts are written in Lua. Every script needs a few key components: a description, categories, the portrule (when does this script run), and the action (what does it do).

Here's a minimal example....a script that grabs the banner from an open port:

```
Code description = [[
Connects to an open port and grabs the service banner.
]]
```

```
author = "YourHackerHandle"
license = "Same as Nmap--See https://nmap.org/book/man-legal.html"
categories = {"discovery", "safe"}
```

```
local nmap = require "nmap"
local shortport = require "shortport"
```

```
portrule = shortport.portnumber({21, 22, 23, 25, 80}, "tcp")
```

```
action = function(host, port)
  local socket = nmap.new_socket()
  socket:connect(host, port)
  socket:set_timeout(5000)
```

```
  local status, banner = socket:receive()
  socket:close()
```

```
  if status then
    return banner
  end
end
```

Save this as `banner-grab.nse` in `/usr/share/nmap/scripts/`, then run:

```
Code sudo nmap --script-updatedb  
nmap --script=banner-grab 192.168.1.237
```

In Zenmap, this command shows up exactly the same in the Command field, you can save it as a custom profile (covered below) so it's one click away.

For more in depth learning, i highly recommend checking out the official NSE script library. It is installed at `/usr/share/nmap/scripts/` — open a few of the vuln or auth category scripts and read through them. Most are well-commented and a great way to learn Lua + NSE patterns by example.

Firewall & IDS Evasion

If your target has a firewall, IPS, or is being monitored, a loud intense scan will get you flagged or blocked instantly. Nmap has several techniques to slip under the radar, useful for pen testing and for understanding how attackers operate.

Timing Templates

Nmap has 6 timing templates (-T0 through -T5):

Code -T0 - Paranoid (1 probe every 5 min, IDS evasion)

-T1 - Sneaky (1 probe every 15 sec)

-T2 - Polite (slows down to use less bandwidth)

-T3 - Normal (default)

-T4 - Aggressive (faster, assumes good network)

-T5 - Insane (fastest, sacrifices accuracy)

For evasion, -T0 or -T1 spreads probes out so far that many IDS systems won't correlate them as a scan.

Decoys

Code `nmap -D 192.168.1.50,192.168.1.51,ME 192.168.1.237`

This makes it look like the scan is coming from multiple IPs at once, with your real IP (ME) hidden in the noise. Decoy IPs need to be live hosts or the packets may just get dropped.

Packet Fragmentation

Code `nmap -f 192.168.X.X`

Splits TCP headers across multiple packets, which can slip past older packet-filtering firewalls and IDS that don't reassemble fragments before inspection.

Spoofing Source Port

Code `nmap -g 53 192.168.X.X`

Some firewalls trust traffic from specific ports (like 53/DNS or 80/HTTP). Forcing Nmap to scan from that source port can sometimes bypass those rules.

MAC Address Spoofing

Code `nmap --spooof-mac 00:11:22:33:44:55 192.168.1.237`

Useful on local networks to avoid MAC-based filtering or to make your traffic blend in with other vendor devices.

Custom Profiles & Automating Your Workflow

By now you're building some pretty complex commands, i cant remember shit nowadays so i save them as profiles:

1. Go to Profile -> New Profile or Command
2. Give it a name (e.g. "Vuln Scan - Aggressive")
3. In the Scan tab, paste your full command syntax
4. Save — it now appears in your Profile dropdown for one-click reuse

Example profiles i personally have saved:

Code # Quiet vuln check

`nmap -sV -T2 --script=vuln,safe %TARGET%`

```
# Full aggressive recon
nmap -A -T4 -p- --script=default,vuln %TARGET%
```

```
# Stealthy SMB sweep
nmap -T1 -f --script=smb-vuln* %TARGET%
```

Scripting Zenmap Scans from the CLI

Zenmap saves all your profiles in ~/.zenmap/scan_profile.usp. You can also drop straight to the terminal and run nmap directly with the same flags, output to XML, and reopen the results in Zenmap later:

```
Code nmap -A --script=vuln -oX scan_results.xml 192.168.1.0/24
zenmap scan_results.xml
```

This is handy for scheduling scans with cron and reviewing the results visually later, combine with the compare results feature from part 1 to track vulnerability changes over time.

That wraps up the series! Hope you enjoyed it and got some use out of it, any questions just post them below. Thanks.

Entering the Brotherhood Tutorial Contest.

[7 Bastille Rsynced Weekend Offender](#)

Wake - June 15, 2026, 4:25 am

Very nicely laid out and a good read, well done mate.

Ivanov_1 - June 15, 2026, 3:22 pm

Awesome tutorial,thank you.

pepe2233 - June 15, 2026, 5:54 pm

very nice tutorial

Carlos Silvestre - June 17, 2026, 8:02 am

Totutrial extremamente excelente [warn]