uc3m | Universidad Carlos III de Madrid
Departamento de Ingeniería Telemática

Scanning with nmap

Florina Almenares Mendoza, Alfonso de Jesús Pérez
Andrés Marín López
Cyber Attack Techniques
Master in Cybersecurity

Course: 2025/2026

Contents

Abstract

Using virtualized environments, we will use `nmap` from a terminal in Kali VM for performing and developing scanning tests. You can check `(nmap (1)` man page or `(nmap help)`.

In the second part, you find the first assignment. For this assignment, you must submit the script developed with its comments as explained in section 2.1.

1 Scanning with `nmap`

We will use `nmap` from a terminal in Kali. Check the manual page of `nmap` and the documentation provided in Aula Global.

In addition, use `Wireshark` (see `man 1 wireshark`) or `tcpdump` (see `man 8 tcpdump`) to monitor the network activity while performing the tests. If you use `Wireshark`, save the relevant part of the network capture.

1.1 Scanning the internal network

1. Nmap can perform ICMP scans with options `-PE`, `-PP`, and `-PM`. The tool `nping` is included on nmap distribution for easier ICMP scans.

   For host discovery, ensure all VMs are powered on and start capturing the network traffic to analyze what is going on:

   ```
   $ ifconfig eth1
   eth1:  flags=4163<UP, BROADCAST,RUNNING,MULTICAST> mtu 1500
   inet 10.0.3.66  netmask 255.255.255.0  broadcast 10.0.3.6655
   inet6 fe80::250:56ff:fe98:cad prefixlen 64 scopeid 0x20<link>
   ether 00:50:56:98:0c:ad txqueuelen 1000 (Ethernet)
   RX packets 33 bytes 4014 (3.9 KiB)
   RX errors 0 dropped 0 overruns 0 frame 0
   TX packets 53 bytes 4767 (4.6 KiB)
   TX errors 0 dropped 0 overruns 0 carrier 0  collisions 0

   $ sudo nmap -n -sn -PP 10.0.3.0/24 --exclude 10.0.3.66
   ```

   The flag (`-n`) indicates nmap to not use DNS resolution, (`-sn`) to not do port scan and (`-PP`) to do a timestamp ICMP scan in the network `10.0.3.0/24`, excluding `kali` address. Notice that `tcpdump` shows 255 ARQ requests and only answers from the connected VMs and the DHCP server of the internal network are received. If you can not see it, repeat the scan using `Wireshark`.

2. Try now to spoof your IP address with `-S` option. Remember to use `-e` option (`-e eth1`) to tell nmap which interface to send and receive packets on. You can also try to forge your MAC address with `--spoof-mac` option.

3. We are now going to use `nping` to do an ICMP scan. Choose type 8 (echo) using one of the previously detected running machines:

   ```
   ~# sudo nping --icmp --icmp-type 8 --icmp-code 0 -c 1 10.0.3.149

   Starting Nping 0.7.91 (https://nmap.org/nping) at 2021-10-25 07:43 EDT
   SENT (0.0468s) ICMP [10.0.3.66 > 10.0.3.149 Echo request \
     (type=8/code=0) id=36138 seq=1] IP [ttl=64 id=4956 iplen=28 ]
   RCVD (0.0473s) ICMP [10.0.3.149 > 10.0.3.66 Echo reply \
     (type=0/code=0) id=36138 seq=1] IP [ttl=64 id=8477 iplen=28 ]

   Max rtt: 0.301ms | Min rtt: 0.301ms | Avg rtt: 0.301ms
   Raw packets sent: 1 (28B) | Rcvd: 1 (46B) | Lost: 0 (0.00%)
   Nping done: 1 IP address pinged in 1.08 seconds
   ```

In the execution trace we can see how tcpdump output interleaves with the `nping` debug. Tcpdump is always informing us, generating more information (ARP packets not shown by `nping`).

`nping` also provides an interesting functionality: it can be run from two different machines, as a TCP echo server (`--echo-server passphrase`) or echo client (`--echo-client passphrase`) to discover NATs and proxies in between. The client may be configured to send specific probes using flags like `--tcp`, `--icmp`, or `--udp`. Protocol header fields may be manipulated normally using the appropriate options, e.g., `--ttl`, `--seq`, `--icmp-type`, etc. The only exceptions are ARP-related flags.

4. Let us use `nping` to do other ICMP scanning (types and codes [4]). `nping` allows us to specify the source MAC and IP address to do the scanning.
5. Let us do a TCP scan, checking with `Wireshark`/`tcpdump` the real packets delivered to the network. By default, use the 80 port for the probes requiring it, and do not use a DNS server since we have none configured. You can also use the `--packet-trace` option to see all packets sent and received on the terminal (in addition to `Wireshark`). Try the following techniques:
   (a) TCP SYN
   (b) TCP ACK
   (c) TCP connect
   (d) TCP XMAS
   (e) TCP NULL
   (f) UDP
   (g) Service version detection
   (h) Operating System detection

   Take note of the results and your observations for each of the different scans.
6. Nmap offers the possibility of using zombies and decoys in scanning tests. Let us experiment with them. Try using metasploitable3 as zombie and decoy to scan metasploitable2. Try the options:

```
$ sudo nmap -T4 -Pn -p1-65535 -sI <zombie-ip> <target-ip>
$ sudo nmap -T4 -Pn -p1-65535 -D <decoy-IP> <target-ip>
```

   NOTE: If any of the commands fail, try rebooting both the zombie and the target; a reboot can clear processes that generate traffic, resets the IPID counters, and flushes the conntrack/cache/ARP tables. That creates a window in which the zombie is idle and IPID can behave predictably.
7. Experiment with timing and parallelization options offered by nmap: `-T[0-5]`, `--min-hostgroup`/`--max-hostgroup` and `--min-parallelism`/`--max-parallelism` for better controlling the use of scanning probes. Check out how they work in a SYN and a CONNECT scan with the `--packet-trace` option, or observing `tcpdump` and `Wireshark` timestamps.

1.2 Nmap GUI

You can also use `zenmap` as a GUI for nmap and perform the scan of `metasploitable` as indicated in Fig. 1, where option `-A` enables OS detection, version detection, script scanning, and traceroute version detection, and option `-v` increases the verbosity of nmap.

Notice that Zenmap includes a different version of Nmap than the one installed on Kali.

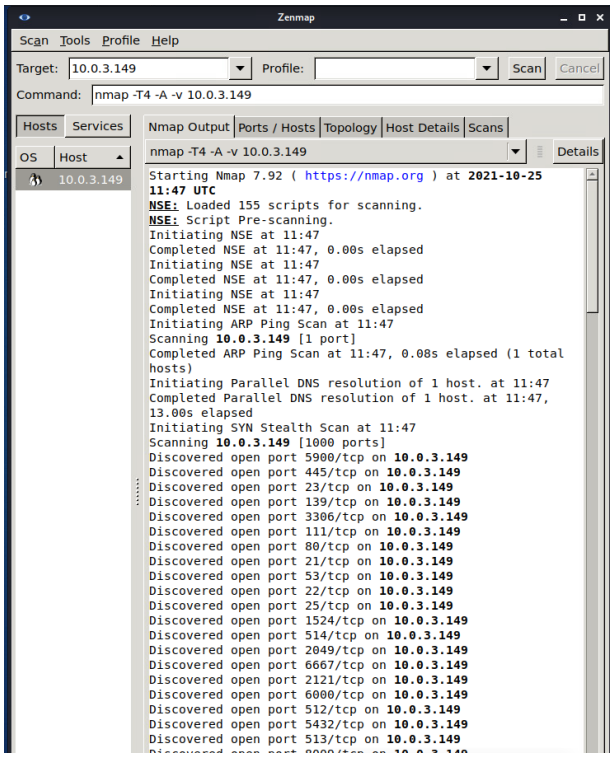

Figure 1: Zenmap from kali to metasploitable

1.3 Scripts in nmap. Usage and modification

We are now going to see how the most powerful nmap feature works: *scripts*. Scripts are written in LUA language [5]. We will start with a basic usage of OS and version detection with `-A` option.

We will analyze a script located in the default nmap scripts directory [1]. The analyzed script, `http-auth-finder.nse`, crawls a web looking for authentication requirements, both **HTTP-based** authentication and **FORM-based** authentication.

1.3.1 HTTP authorization NSE. First: script requirements

The script starts by defining the requirements (modules) that have to be loaded before running the script [6]. In this case, loads http, httpspider, shortport, nmap, stdnse, tab and table. They will be used in the script with the defined local variables [7]:

```
local http = require "http"
local httpspider = require "httpspider"
local nmap = require "nmap"
local shortport = require "shortport"
local stdnse = require "stdnse"
local tab = require "tab"
local table = require "table"
```

[1] `/usr/share/nmap/scripts`

1.3.2 HTTP authorization NSE. Second: script description

The script starts with a short `description` and the `usage` documentation (arguments) together with an example output. The usage is included as a LUA comment (lines starting with `--`).

```
description = [[
Spiders a web site to find web pages requiring form-based or HTTP-based
authentication. The results are returned in a table with each url and the
detected method.
]]
-- @usage
-- nmap -p 80 --script http-auth-finder <ip>
--
-- @output
-- PORT   STATE SERVICE
-- 80/tcp open  http
-- | http-auth-finder:
-- |   url                                    method
-- |   http://192.168.1.162/auth1/index.html  HTTP: Basic, Digest, Negotiate
-- |_  http://192.168.1.162/auth2/index.html  FORM
--
-- @args http-auth-finder.maxdepth the maximum amount of directories beneath
--       the initial url to spider. A negative value disables the limit.
--       (default: 3)
-- @args http-auth-finder.maxpagecount the maximum amount of pages to visit.
--       A negative value disables the limit (default: 20)
-- @args http-auth-finder.url the url to start spidering. This is a URL
--       relative to the scanned host eg. /default.html (default: /)
-- @args http-auth-finder.withinhost only spider URLs within the same host.
--       (default: true)
-- @args http-auth-finder.withindomain only spider URLs within the same
--       domain. This widens the scope from <code>withinhost</code> and can
--       not be used in combination. (default: false)
```

To give the variables specific values, you have to specify at the command line, in a comma-separated list, a pair of name=value [8]:

```
$ nmap --script http-auth-finder --script-args http-auth-finder.maxdepth=2
```

After that, different fields are defined: `author` (Patrik Karlsson), `license` and `category`. They are used by the documentation system (`NSEDoc`) to show information of these and many other parameters, e.g., `@usage` or `@args`.

1.3.3 HTTP authorization NSE. Third: rules section

Next is the rules section, every rule is evaluated to *true* or *false*. Only when it leads to *true*, the script is executed with the target. In this case the rule is:

```
portrule = shortport.http
```

Note that the rule is only valid for http, not https.

1.3.4 HTTP authorization NSE. Fourth: functions and actions

Later on, an auxiliary function (local scope) is defined. The function takes as input an HTTP response and returns a two values tuple. It returns *false* and a self-explanatory string in case of no HTTP authentication, while the tuple *true* and the *challenges* (REALM, nonces...) is returned in case HTTP authentication

(`Basic o Digest`) is used. The `http.parse_www_authenticate` greatly simplifies this function.

```
local function parseAuthentication(resp)
  local www_authenticate = resp.header["www-authenticate"]
  if ( not(www_authenticate) ) then
    return false, "Server returned no authentication headers."
  end

  local challenges = http.parse_www_authenticate(www_authenticate)
  if ( not(challenges) ) then
    return false, ("Authentication header (%s) could not be parsed."):
      format(www_authenticate)
  end
  return true, challenges
end
```

Next, as in any NSE script, comes the action that will be executed if the script is activated (if the rule is *true*), it is less than 60 lines of code, thanks to the modules loaded.

```
action = function(host, port)
```

First, it creates an instance of the `crawler`:

```
  -- create a new crawler instance
  local crawler = httpspider.Crawler:new( host, port, nil,
                                          { scriptname = SCRIPT_NAME } )

  if ( not(crawler) ) then
    return
  end
```

It creates a table to store the results and then adds the header to the table. The `crawler` timeout is set to 10secs.

```
  -- create a table entry in the registry
  nmap.registry.auth_urls = nmap.registry.auth_urls or {}
  crawler:set_timeout(10000)

  local auth_urls = tab.new(2)
  tab.addrow(auth_urls, "url", "method")
```

We are finally ready to define the main action loop, it will send HTTP requests to the host for the identified pages:

```
  while(true) do
    local status, r = crawler:crawl()
```

If the HTTP server does not return a status code and the crawler returns an error, we output the error:

```
    -- if the crawler fails it can be due to a number of different reasons
    -- most of them are "legitimate" and should not be reason to abort
    if ( not(status) ) then
      if ( r.err ) then
        return stdnse.format_output(true, ("ERROR: \%s"):format(r.reason))
      else
        break
      end
    end
```

Then we check two possible authentications: `HTTP-based` authentication and `FORM-based` authentication. HTTP auth is the case when the status code is `401`, and we pass the response to the previously defined auxiliary function:

```
    -- HTTP-based authentication
    if ( r.response.status == 401 ) then
      local status, auth = parseAuthentication(r.response)
```

If the status is *true*, we insert the schemes (we do not include the params, see RFC 2617) in the results tab:

```
      if ( status ) then
        local schemes = {}
        for _, item in ipairs(auth) do
          if ( item.scheme ) then
            table.insert(schemes, item.scheme)
          end
        end
        tab.addrow(auth_urls, r.url, ("HTTP: %s"):
                   format(table.concat(schemes, ", ")))
      else
        tab.addrow(auth_urls, r.url, ("HTTP: %s"):format(auth))
      end
      nmap.registry.auth_urls[r.url] = "HTTP"
```

If the response status is not 401, we look for `FORM-based` authentication in the returned page with a very basic strategy: searching an input field of type `password` (check all case possibilities) in the response body:

```
    -- FORM-based authentication
    elseif r.response.body then
      -- attempt to detect a password input form field
      if ( r.response.body:match("<[Ii][Nn][Pp][Uu][Tt].-"..
                                 "[Tt][Yy][Pp][Ee]"..
                                 "%s*=\"*[Pp][Aa][Ss][Ss][Ww][Oo][Rr][Dd]")
      ) then
```

We add the new results to the results table and end the loop.

```
        tab.addrow(auth_urls, r.url, "FORM")
        nmap.registry.auth_urls[r.url] = "FORM"
      end
    end
  end
```

Finally, we output the results and the script ends.

```
  if ( #auth_urls > 1 ) then
    local result = { tab.dump(auth_urls) }
    result.name = crawler:getLimitations()
    return stdnse.format_output(true, result)
  end
end
```

Debugging a NSE script

The preferred method to debug a NSE script is the option `--script-trace`. If this option is enabled, all incoming and outgoing communication performed by scripts is printed. It is automatically enabled if the option `packet-trace` is specified. The general debugging option for nmap is `-d <int>`. To have nmap log the debugging information (warnings and errors) use the option `--log-errors`. The `stdnse.debug()` function from the `stdnselib` prints debugging messages:

```
stdnse.debug(<debug level required>, <format string>, arg1, arg2...)
```

More information about nmap debugging can be found in the book *Nmap: Network Exploration and Security Auditing Cookbook*. Get access through UC3M digital library [9].

2 First Assignment

The purpose of this assignment is to gain familiarity with NSE scripts, increase your abilities to understand and improve other people's code, and enrich your (or acquire) competencies in LUA programming. Several LUA scripts have been developed for nmap [11]. Understanding and programming in LUA is very intuitive, and there are tons of scripts to help you. It's acceptable to research and utilize existing scripts, provided that you acknowledge the original creators by including comments and references to them in your work. However, refrain from merely copying; it's important that you write your own code and contribute something unique to the project.

2.1 Vulnerable TLS Server NSE script

Finding vulnerable HTTPS servers is crucial because it can expose critical weaknesses in the network's security infrastructure, giving attackers a path to sensitive data or control over the system. You can exploit SSL/TLS misconfigurations, such as the use of weak ciphersuites, support for deprecated protocols, non-recommendable permissions, etc., or vulnerable certificates.

For that, you can check the relevant module documentation[2]. Here follows a concise description and some hints about the **script you are required to deliver in this assignment**.

2.1.1 Script Name

The script name must be `vulnTLSServer` and will be required to be stored in a file named `vulnTLSServer.nse`.

2.1.2 Script Objective

The script developed must show alerts about the severity levels of the identified vulnerabilities, inspect the web server's certificate, and analyze TLS connection.

The vulnerabilities should align with industry best practices, such as:

- OWASP Cheat Sheet,
- Mozilla Server Side TLS Intermediate Configuration

2.1.3 Basic Functionality

The server should generate alerts of the vulnerabilities in the server certificate and the TLS configuration according to the following severity levels:

Critical Alerts:

- **Self-Signed Certificates:** Certificates must be signed by a trusted certificate authority (CA) to be trusted by users. For internal applications, an internal CA may be acceptable.
- **CBC and/or SHA Support:** If cipher suite used in the TLS connection includes CBC mode and/or SHA hash algorithm a critical alert must be shown, because CBC mode is vulnerable to various attacks like BEAST, Lucky Thirteen, and POODLE, and SHA-1 is deprecated.

[2] `https://nmap.org/nsedoc/scripts/ssl-cert.html` and `https://nmap.org/nsedoc/scripts/tls-alpn.html` for help with sslcert and tls connection, respectively.

- **Enable Compression:** TLS compression must be disabled to protect against the CRIME vulnerability, which could allow attackers to recover sensitive information such as session cookies.

High Alerts:

- **Certificate Type:** Certificates should be of type ECDSA (P-256) or RSA (2048 bits). Any type of certificate with a lower key size or different encryption should trigger a high alert.
- **Supported Protocols:** The server must support TLS 1.2 or TLS 1.3 by default. If it does not support these protocols and supports TLS 1.0 (or older versions), a high alert should be raised.
- **Cipher Suites:** Suites supported must include the following ones[3]:
  - `ECDHE-ECDSA-AES128-GCM-SHA256`
  - `ECDHE-RSA-AES128-GCM-SHA256`
  - `ECDHE-ECDSA-AES256-GCM-SHA384`
  - `ECDHE-RSA-AES256-GCM-SHA384`
  - `ECDHE-ECDSA-CHACHA20-POLY1305`
  - `ECDHE-RSA-CHACHA20-POLY1305`
  - `DHE-RSA-AES128-GCM-SHA256`
  - `DHE-RSA-AES256-GCM-SHA384`
  - `DHE-RSA-CHACHA20-POLY1305`

  Any other supported cipher suite should trigger a high alert, indicating the cipher suite.

Medium Alerts:

- **Certificate Lifespan:** Certificates should have a lifespan of 90 to 366 days. If a certificate expires in less than 90 days or more than 366 days, a medium alert should be raised.
- **Domain Name Matching:** The domain name of the certificate must match with the common name (CN) attribute.

Low Alerts:

- **Avoid Non-Qualified Host names:** Non-qualified host names should not be included.
- **Avoid IP Addresses:** IP addresses should not be included in the certificate attributes, e.g., in the subject's CN.

IMPORTANT NOTE: Do not modify or change anything in Nmap's internal libraries (/usr/share/nmap/nselib). You must include all your functions inside your NSE script so the script is portable to any machine and works with a default Nmap installation.

2.1.4 Testing

To test these functionalities, a secure Apache Web server is provided in metasploitable2. This supports two virtual hosts; one listens on port 443 (ssl port), and the other listens on port 9443.

[3]These cipher suites are specified for TLS 1.2, but in this case, you can consider them for that version or older ones.

Additionally, there is a Docker container located in the `Downloads` folder within your `$HOME` directory on the Kali VM. To load and run this container, which hosts an Apache server on localhost with two virtual hosts listening on ports 443 and 9443, use the following commands:

```
~/Downloads$ sudo docker rm -f $(sudo docker ps -q)
c722ef939e72
~/Downloads$ docker load -i tc_lab1_apache.tar
Loaded image: tc_lab1_apache:latest

~/Downloads$ docker image ls
REPOSITORY       TAG       IMAGE ID       CREATED          SIZE
tc_lab1_apache   latest    120140dac5f6   6 minutes ago    489MB

~/Downloads$ docker run -d --name tc_lab1 --restart always \
  -p 8888:80 \
  -p 8443:443 \
tc_lab1_apache
```

Keep the assigned ports (8888 and 8443) unchanged to avoid potential conflicts with other host services. The Apache server is bound to all network interfaces, allowing access to any host IP address. To test different hostnames, edit your local /etc/hosts file accordingly. Finally, open a web browser to confirm access to http://localhost:8888 and https://localhost:8443 .

Your script must work with both in the lab and **keep the evidence there**. We will use it for evaluation.

The Apache configuration files are stored in `/etc/apache2/sites-enabled/`. You can modify the certificate or configuration to test some basic and enhanced functionalities (see section 2.1.6).

2.1.5 Nmap Output Format

The alerts should be generated following the following format:

```
**********************
<SEVERITY> ALERTS: <number of these severity alerts>
**********************
- <Title of alert1>. <Description of alert>
- <Title of alert2>. <Description of alert>
- ...

**********************
```

For example:

```
**********************
CRITICAL ALERTS: 2
**********************
- Self-signed certificate detected
- Cipher includes CBC mode and SHA hash algorithm

**********************
HIGH ALERTS: 2
**********************
- Unsupported TLS cipher: TLS-RSA-WITH-AES-128-CBC-SHA
- Server does not support TLS 1.2 or TLS 1.3 by default

**********************
MEDIUM ALERTS: 1
**********************
- Certificate lifespan is 75 days (less than recommended 90 days)

**********************
```

2.1.6 Enhanced Functionality

You can define extra functionality for the script such as:

- **HSTS (Host Strict Transport Security) Configuration:** If the HTTP header HSTS is not configured, a high alert should be raised. In addition, if its `max-age` is lesser than 63072000 (two years), a medium alert should also be raised. For that, you must modify the `VirtualHost` configuration, adding the header, in Apache files.
- **Server Information Disclosure**: A medium alert should be raised if the server includes sensitive information in the HTTP headers (such as version numbers).
- **TLS Curves:** The server should support: 'X25519', 'prime256v1', and 'secp384r1'. Any other supported TLS curve should trigger a high alert, indicating the curve's name. You must issue a new server certificate.
- **DH Parameter Size:** For TLS 1.3 or 1.2, DH parameters must be 2048 (ffdhe2048, RFC 7919). If not, a high alert should be raised. You should change the Apache server default configuration to test it.
- **Wildcard Certificate Scope:** Limit the scope of a wildcard certificate by issuing it for a subdomain (e.g., *.foo.example.org) or for a separate domain. A low alert must be shown if a wildcard is included in the CN or SAN (Subject Alternative Names). This can be tested using the docker container or requires you to issue a new certificate to test it.
- **CN and SAN Attributes:** For compatibility reasons, certificates should have the primary FQDN in the CN, with the full list of FQDNs in the SAN. If not, a low alert should be raised. This can be tested with the docker container or requires you to issue a new certificate to test it.
- **Cipher Preference:** If the client chooses the cipher, a low alert should be raised for any other configuration found.

Other functionalities may be proposed based on the issues published in the Trend Micro report, e.g., indicating whether the certificate pinning header is present in the HTTP response.

2.2 Delivery format

The delivery must include the following files in zip format:

1. The file with the script code, that is, `vulnTLSServer.nse`.
2. A folder named test with the tests performed to metasploitable2, including two files per test:
   - a **file** describing the test with the following fields:
     - **Description**: brief explanation of the test performed
     - **Input**: the arguments of command
     - **Output**: the expected outcome of the script
     - **Command**: the script invocation
   - a **file** with the output (-oN) of the script.

   Both files will be named after the test number: `test01.txt` and `test01.out`.
3. If you change the Web server's certificates, these must be provided to replay your test results.

References

[1] VMware. vSphere Platform. `https://docs.vmware.com/es/VMware-vSphere/index.html`

[2] UC3M. Ciberlab. `https://vciberlab2.lab.it.uc3m.es`

[3] Offensive Security. Kali Linux. `https://www.kali.org`

[4] IANA. ICMP parameters. `https://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml`

[5] LUA. Documentation. `https://www.lua.org/`

[6] LUA. Manual, *require*. `https://www.lua.org/manual/5.3/manual.html#6.3`

[7] LUA. Manual, *local*. `https://www.lua.org/manual/5.3/manual.html#3.3.7`

[8] NMAP. NSE script args. `https://nmap.org/book/nse-usage.html#nse-args`

[9] UC3M. Digital Library. `https://biblioteca3.uc3m.es/RREE/recurso.php?recurso=SBO`

[10] Trend Micro. "The State of SSL/TLS Certificate Usage in Malware C&C (Command-and-Control) Communications". Mohamad Mokbel. `https://www.trendmicro.com/content/dam/trendmicro/global/en/research/21/i/ssl-tls-technical-brief/ssl-tls-technical-brief.pdf`. September, 2021.

[11] NMAP. Documentation. `https://nmap.org/nsedoc`