

FIT3031 - Network Security: Assignment I (2025 S2)

Total Marks 100

Due date: September 25th, Thursday, 11:55:00 PM (Melbourne time)

1 Overview

The learning objective of this assignment is for you to gain a first-hand experience on network attacks (i.e., TCP/UDP scanning techniques and DNS attacks) and get a deeper understanding on how to launch these attacks in practice. All tasks in this assignment can be done on the unit's virtual environment as used in the labs. This is an **individual** assignment. You are **not** allowed to help or seek help from other people. You are **not** allowed to re-use any previous assessment submission materials (even if they were created by you).

2 Submission Policy

For this assignment, you need to submit four files using a two submission links on Moodle:

- **Submission link 1:** A single **video file** containing the recording of you carrying out all tasks in both Section 3 and Section 4
- **Submission link 2:** The final versions of the **tcp-scanner.py**, **udp-scanner.py**, **remote_dns.py** scripts from Section 3.

Name your submission files as the format: **[Your Name]-[Student ID]-FIT3031-Assignment1**, e.g., **HarryPotter-12345678-FIT3031-Assignment1.mp4**.

Moodle allows a maximum submission size of 500 MB. Therefore, the total size of the above three submission files should not exceed 500 MB.

Important notes and penalties:

- It is the student's responsibility that the submitted video file can be opened on a standard Windows computer (without requiring specialised software), and that the images and texts shown in the video are understandable/readable (in English). If the video file cannot be opened, you will receive zero mark. After uploading **draft** files (**before** finalising the submission), we recommend you to download your uploaded files and check that they open and run properly and as intended. Once you finalise your submission, you will **not** be able to revise it.
- Note that draft files are **NOT** accepted and will not be marked. You must finalise your submission (with status shown as "submitted for grading") for your assignment to be considered as valid. Otherwise, you may receive late submission penalty or zero mark.
- Note that during your assignment submission, you may see a Turnitin warning for your **video file** stating something of the form "This file will not be submitted to Turnitin...". **This is completely fine**, and you can simply ignore this message. The video file is not scanned by Turnitin.
- At the beginning of your recording, you must clearly show your face and have your photo ID (preferably your Monash ID with photo) presented in the first slide as shown below (please update the slide contents as appropriate). Make sure the ID card details are clearly readable/visible.

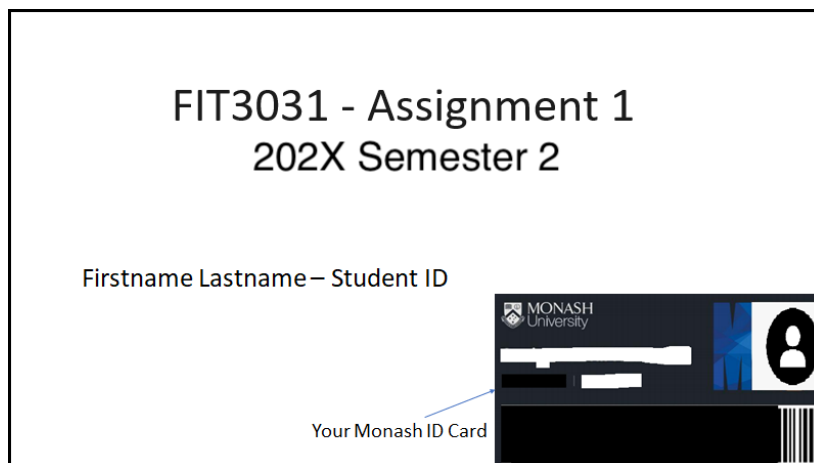


Figure 1: Sample opening slide

- A part of the submitted video (at a corner) must clearly show your face at all times. The assignment will be deemed invalid and receive zero mark when that's not the case.
- Late submissions incur a 5-point deduction per calendar day. For example, if you submit 2 days and 1 hour late, that incurs 15-point deduction. Submissions more than 7 calendar days late will receive a zero mark.
- If you require extension or special consideration, refer to <https://www.monash.edu/students/admin/assessments/extensions-special-consideration>. No teaching team member is allowed to give you extension or special consideration, so please do not reach out to a teaching team member about this. Follow the guidelines in the aforementioned link. When requesting extension/special consideration via the central system, please select the correct assessment name from the provided drop-down menu, and **not manually type** an assessment name.
- **The maximum allowed duration for the recorded video is 20 mins.** Therefore, only the first 20:00 mins of your submitted video will be marked. Any exceeding video components will be ignored. Speeding up the video recording (e.g. using a software) is not allowed and such submissions will receive a zero mark.
- If your device does not have a camera (or for whatever reason you can't use your device), you can borrow a device from Monash Connect or Library. It's your responsibility to plan ahead for this. Monash Connect or Library not having available devices for loan at a particular point in time is not a valid excuse.
- You can create multiple video parts at different times, and combine and submit a single video at the end. Make sure that the final video is clear and understandable.
- All tasks must be live demonstrated instead of explaining an already completed task. You are **not** allowed to add voice-over later on. You are also **not** allowed to read from prepared scripts. At the beginning of each task, please clearly mention what task is being carried out in the video.
- If any task requires installing new software, you are allowed to do that in advance of recording your video. You do not need to demonstrate software installation in the video.
- You can do (online) research in advance, take brief notes¹ and make use of them during your video recording. You may also prepare Python codes in advance. But you cannot simply copy-paste commands to carry out the tasks without any explanations. Explanations (of what the code does) while completing the tasks are particularly important.
- In the remote DNS attack, you may need to run the attack for a long time (in the order of a few hours) due to brute forcing. In this case, you can record the parts until the time-consuming step, pause/stop the video recording, and then continue to record once the time-consuming step concludes. You may merge multiple recordings as mentioned before.
- Zero tolerance on plagiarism and academic integrity violations: If you are found not adhering to the Monash Academic Integrity Policies, penalties will apply, e.g., a zero grade for the unit. The demonstration video is also used to detect/avoid plagiarism. University policies can be found at <https://www.monash.edu/students/academic/policies/academic-integrity>.

¹The notes cannot be full scripts that you just read in the video. They should only be brief reminders for you.

3 Port Scanning Techniques [50 Marks]

A port scan is a common technique used by attackers to discover entry points in a system. The following task focuses on network traffic analysis and security measures related to port scanning techniques. By following the steps, students will analyse network interactions between a client and a server, identify and interpret the type of scan used, and explore countermeasures to enhance security. Additionally, students will gain practical experience by implementing evasive techniques in a scanner script and observing the effects. The goal is to deepen understanding of port scanning concepts and hands-on mitigation strategies.

3.1 Environment Setup

For this question we will be using a new network topology. Run the command below on the GNS3 VM shell to download the project. If you SSH into the VM from your host OS terminal, you can simply copy and paste the command instead of typing it manually.

```
gdown 10IgcJxFfLd3k6i0UvVX9WAmU9vJqKhqG ; sudo bash ./install_A1-PortScanning.sh
```

Alternatively, you can use the link below to download the same project. However, if you are connected to the **Monash Wi-Fi**, this method may not work. In that case, please use a mobile hotspot. (single command)

```
wget https://sniffnrun.com/install_A1-PortScanning.sh --no-check-certificate ; \
sudo bash ./install_A1-PortScanning.sh
```

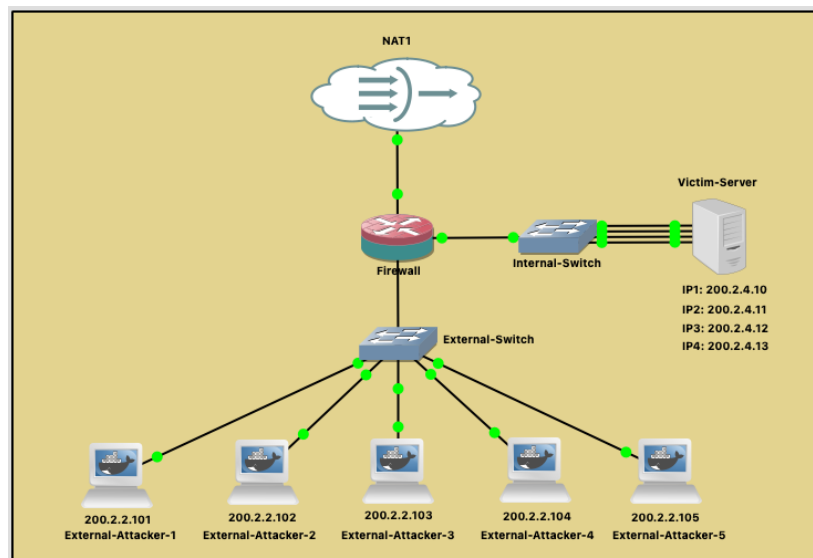


Figure 2: GNS3 Config

Download and copy the `server.py` file on to the Victim-Server and `tcp-scanner.py` and `udp-scanner.py` on to the External-Attacker-1. You will have to create this file in the `/home` directory to make sure the file not deleted after a reboot. Run Wireshark on the link connecting External-Switch to the Firewall and the link connecting the Internal-Switch to the Firewall.

Run `server.py` as below with **your student ID**.

Note: Victim-Server has multiple IP addresses. Each IP has different levels of traffic restrictions configured. You will be using different IP address for different tasks. Using the wrong IP address for a task will render it invalid and result in zero marks.

IMPORTANT: If you don't use the correct student ID, your attempt will be deemed invalid and you will receive zero mark for this Section 4.

```
python3 server.py <Student ID>
```

Running the script with your student ID will always open the same unique set of ports on the server.

3.2 Question 1

Now run `tcp-scanner.py` as below to scan the entire TCP port range of the Victim-Server and observe the traffic in Wireshark.

```
python3 tcp-scanner.py 200.2.4.10
```

Identify and describe the type of TCP scan by analyzing the traffic observed in Wireshark. Explain how the scan operates, and using examples from the traffic flow, demonstrate how the scan distinguishes between a closed port and an open port. **(4 marks)**

3.3 Question 2

Detecting port scans is a key aspect of network security, as they often indicate malicious intent. A common method is to monitor for a high volume of requests from the same source IP to the same destination IP within a short time. Once identified, **rate limiting** at control points such as firewalls, IDS/IPS, or load balancers can then be applied to block these scans.

Note: Rate limiting is a network or application control mechanism that restricts the number of requests, connections, or packets a source can send within a specific period of time.

Identify and explain using Wireshark, **THREE other** traffic patterns you observed that could be used to detect the above-mentioned type of TCP port scan. **(3 marks)**

3.4 Question 3

Evading port-scanning countermeasures is a common tactic used by attackers to avoid detection. One method of bypassing **rate limiting** on a source IP is to perform scans at a slower pace, making them more stealthy.

Identify and explain **THREE other** ways the identified TCP scan could be modified to evade the countermeasures described in the previous question. Clearly indicate which evasion technique corresponds to which countermeasure, with one technique per countermeasure.

Note: While these techniques are illegal and unethical when used without proper authorisation, understanding them is essential for strengthening security defences. **(3 marks)**

3.5 Question 4

The victim's organization observed a large number of port scans targeting the server. In response, they implemented a per-source IP rate limit of 100 new connections per second for traffic directed to the server.

The `tcp-scanner.py` script includes the `--conns-per-batch` argument, which defines how many parallel workers are executed per second, with each worker scanning one port. Run the command below to test this argument.

```
python3 tcp-scanner.py 200.2.4.11 --conns-per-batch 100
```

As a result of this, scanning all 65,535 ports now takes more than 10 minutes to complete.

As a penetration tester, you may not have such a long time window to carry out your attack. How can you improve the scanning technique to complete all 65,535 ports under 60 seconds while still evading the firewall's rate-limiting countermeasure?

You may add more **External-Attacker** clients if needed for this scan. However, the attack must be executed from only one client. (For example, you cannot run the same scanning script simultaneously on all five machines.) After the scan, all open ports should be accurately displayed on the console(s).

The restricted IP address for this task is **200.2.4.11**. You should perform the scan for this task on that IP. **(18 marks, or 5 marks if not executed centrally)**

Notes:

1. You must scan the entire TCP port range (0-65535) for the demonstration of this task.
2. The scan must be executed and completed within the demonstration video, and the timer in the scan script should display the total time taken to finish the scan. You are not allowed to edit the video during this scan.
3. You are not allowed to change the provided scan type in this question.
4. Once black-listed (go over the rate limit), the source IP will stay black-listed for 2mins on the firewall.

3.6 Question 5

To further strengthen security, the organization's network security team decided to enforce additional firewall controls to stop TCP scans on the server. Identify these restrictions and modify your scan to bypass them. Your modified scan should identify all the open and closed ports on the destination server. You allowed to do any modification to the scan in this task.

The restricted IP address for this task is **200.2.4.12**. You should perform the scan for this task on that IP.

Demonstration with explanation of the modifications to the scan **(5 marks)**

3.7 Question 6

Now run `udp-scanner.py` to scan the first 100 UDP ports and observe the traffic in Wireshark.

```
python3 udp-scanner.py 200.2.4.10 --end-port 100
```

Observe the traffic and explain how the scan operates, and using examples from the traffic flow, demonstrate how the scan distinguishes between a closed port and an open port. **(2 marks)**.

3.8 Question 7

Match the result of the scanner with the actual open ports on the server. Is the result correct? Why? Use the observations on Wireshark in your explanation. **(2 marks)**.

Run the `udp-scanner.py` script with different arguments, without modifying the script itself, to reduce false positives in the scan results. **(4 marks)**.

3.9 Question 8

Run the same scan from the previous question on the IP address below. Do you still observe fewer false positives? If not, explain why. Use Wireshark to support your explanation, citing any evidence you find in the traffic patterns. **(5 marks)**

```
python3 udp-scanner.py 200.2.4.13 --end-port 100
```

3.10 Question 9

Modify the `udp-scanner.py` script to reduce the false positives of the scan from question 8 and run the scan again to demonstrate.

Briefly explain the techniques used and the modifications to the Python code or/and the arguments given in the `udp-scanner.py` script. **(4 marks)**

Notes.

1. All questions (Q1 to Q9) in Section 3 require a live demonstration and must be included in the final submission video. All explanations should be provided during the demonstration.
2. When explaining scans, explain at a minimum one open port and one closed port.
3. Do not explain Python scripts in your demonstration unless explicitly mentioned in the question.
4. You may use the `--start-port` and `--end-port` arguments with the `scanner.py` to scope the scan into a small port range for demonstration purposes, if scanning the entire port range is not a requirement in the question.
5. You must submit the final `tcp-scanner.py` file and the `udp-scanner.py` file on Moodle as part of your assignment submission.

4 DNS Attacks – Using Scapy [50 Marks]

Domain Name System (DNS) is an essential component of the Internet infrastructure. It serves as the phone book for the Internet, so computers can look up for a “telephone number” (i.e. an IP address) from domain names. Without knowing the IP address, computers will not be able to communicate with one another. Due to its importance, the DNS infrastructure faces frequent attacks. In this section, you will explore the most primary attack on DNS. That is DNS cache poisoning by investigating both Local and Remote DNS cache poisoning attacks.

Due to the large number of computers and networks on the Internet, the domain namespace is organised in a hierarchical tree-like structure. Each node on the tree is called a domain, or sub-domain when referencing to its child node. The following figure depicts a part of the domain hierarchy.

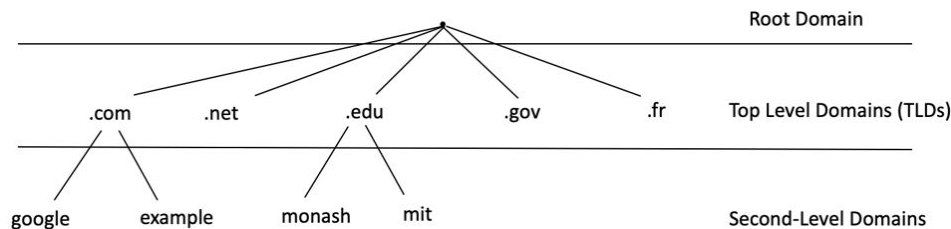


Figure 3: Domain hierarchy

The domain hierarchy tree structure describes how the domain namespace is organised, but that is not exactly how the domain name systems are organised. Domain name systems are organised according to zones. A DNS zone basically groups contiguous domains and sub-domains on the domain tree, and assign the management authority to an entity. Each zone is managed by an authority, while a domain does not indicate any authority information. The following figure depicts an example of the `example.com` domain.

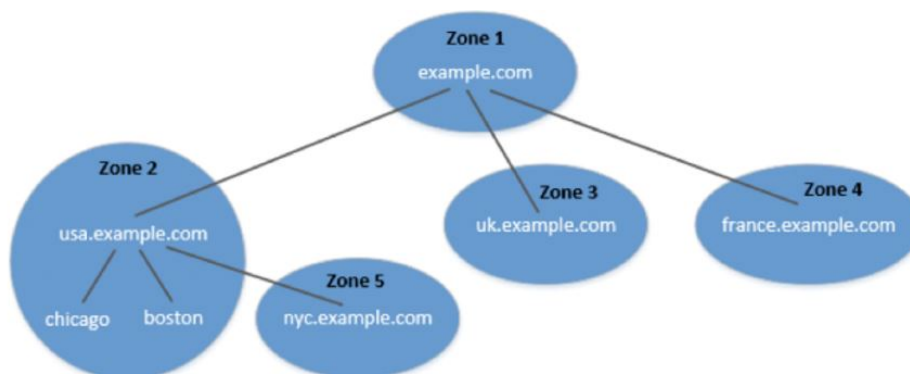


Figure 4: DNS Zones

Assume that `example.com` in the above figure is an international company, with branches all over the world, so the company’s domain is further divided into multiple sub-domains, including `usa.example.com`, `uk.example.com`, and `france.example.com`. Inside US, the `usa` sub-domain is further divided into `chicago`, `boston`, and `nyc` subdomains.

Each DNS zone has at least one authoritative nameserver that publishes information about that zone. The goal of a DNS query is to eventually ask the authoritative DNS server for answers. That is why they are called authoritative because they provide the original and definitive answers to DNS queries, as opposed to obtaining the answers from other DNS servers.

With such arrangement, the root zone for `example.com` only needs to keep records of who the authority is for each of its subdomains. By doing this, it maintains the independence among the branches in different countries and enable the administrative right of those subdomains, so the branch in each country manages its own DNS information. For a given DNS query, if your local DNS server does not know the answer, it asks other DNS servers on the Internet for answer via hierarchical authority servers. The following example demonstrates a dig (DNS query) for the domain `www.example.net` when sending the query directly to one of the root servers (i.e. `a.root-servers.net`).

```

seed@ubuntu:~$ dig @a.root-servers.net www.example.net

(Only a portion of the reply is shown here)
;; QUESTION SECTION:
;www.example.net.                IN      A

;; AUTHORITY SECTION:
net.                172800  IN      NS      m.gtld-servers.net.
net.                172800  IN      NS      l.gtld-servers.net.
net.                172800  IN      NS      k.gtld-servers.net.

;; ADDITIONAL SECTION:
m.gtld-servers.net. 172800  IN      A      192.55.83.30
l.gtld-servers.net. 172800  IN      A      192.41.162.30
k.gtld-servers.net. 172800  IN      A      192.52.178.30

```

Annotations in the image:

- Blue arrow pointing to `a.root-servers.net`: "Directly send the query to this server."
- Blue arrow pointing to the missing answer section: "No answer (the root does not know the answer)"
- Blue arrow pointing to the authoritative servers in the additional section: "Go ask them!"

Figure 5: DIG to the root server

There are four types of sections in a DNS response: *question section*, *answer section*, *authority section*, and *additional section*. From the above result, we can see that the root server does not know the answer (because the reply does not include an answer section, but it tells several authoritative nameservers for the `net` zone (the NS records in the authority section), along with their IP address if possible in the *additional section*). If you continuously dig the domain `www.example.net` on one these authoritative nameservers, you will finally end up with the answer section showing the IP address of the machine hosting the website for `www.example.net`.

When your local DNS server gets information from other DNS servers, it caches the information, so if the same information is needed, it will not waste time to ask again.

4.1 Local DNS Attack targeting Authoritative Nameserver [20 Marks]

4.2 Environment Setup

In this section we will prepare the environment for the DNS attacks in this assessment. Download a new copy of the SecureCorp project using the below link. This is slightly different from the SecureCorp project you had in Week 1. Run the command below on the GNS3 VM shell to download the project. If you SSH into the VM from your host OS terminal, you can simply copy and paste the command instead of typing it manually.

```
gdown 1hhDuc2S7MMdz2K06tfr85PSx7W4t87Pp ; sudo bash ./install_SecureCorp.sh
```

Alternatively, you can use the link below to download the same project. However, if you are connected to the **Monash Wi-Fi**, this method may not work. In that case, please use a mobile hotspot. (single command)

```
wget https://sniffnrun.com/install_SecureCorp.sh --no-check-certificate ; \
sudo bash ./install_SecureCorp.sh
```

Now add an Internal-Server node (Ubuntu-24.04-plus-essentials) to the Corporate LAN and configure it with a static IP.

For this attack both the DNS server and the Internal-Attacker have to be in the same subnet. So first move the DNS server to the Corp LAN and assign a suitable IP configurations (ex: 10.10.10.53).

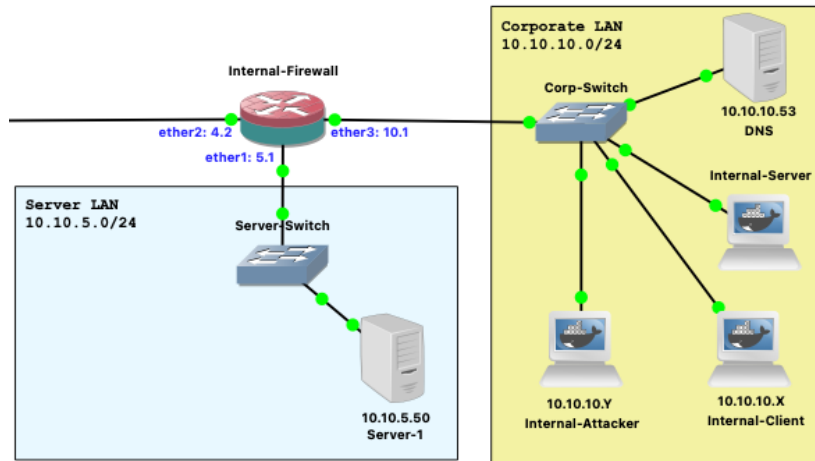


Figure 6: Topology for Local DNS attack

```

auto eth0
iface eth0 inet static
    address 10.10.10.53
    netmask 255.255.255.0
    gateway 10.10.10.1
    up echo nameserver 8.8.8.8 > /etc/resolv.conf

```

Figure 7: Static IP config for DNS

Start the DNS server using the below command.

```
service named start
```

We recalled that a DNS response contains question section, *answer section*, *authority section*, and *additional section*. If we only target the *answer section*, the attack only affects one hostname (as we did in our Week06 lab “DNS Spoofing Attacks”). Real DNS attacks usually target the authority section by providing a fake NS record for the target domain in the authority section. If the fake NS record is cached, when a victim’s local DNS server tries to find any IP address in the target domain, it will send a request to the malicious nameserver specified in the fake NS record. Such an attack can affect all the hostnames in the target domain. In this task, you will explore how to target the authoritative server of **example.net** and replace it with **ns1.attacker.com** and **ns2.attacker.com**.

4.3 Question 10

Using **Internal-Client** as victim and **Internal-Attacker** as the attacker machine, perform a DNS spoofing attack that modifies the authoritative server of **example.net** to be **ns1.attacker.com** and **ns2.attacker.com**. Please record and submit a video showing **and** explaining how you performed the attack. A part of the video must show and explain your Python code. **(20 marks)**

Hint: If the attack works, you should see a result as in the following figures for which the malicious authoritative servers have taken place.


```

root@Internal-Client:/# dig example.net

; <<>> DiG 9.16.1-Ubuntu <<>> example.net
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 60077
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;example.net.                IN      A

;; ANSWER SECTION:
example.net.                 303030  IN      A      10.10.10.1

;; AUTHORITY SECTION:
example.net.                 90000   IN      NS      ns1.attacker.com.
example.net.                 90000   IN      NS      ns2.attacker.com.

;; ADDITIONAL SECTION:
ns1.attacker.com.           90000   IN      A      10.10.10.1
ns2.attacker.com.           90000   IN      A      10.10.10.2

;; Query time: 32 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Sat Aug 21 22:20:55 AEST 2021
;; MSG SIZE rcvd: 202

```

Figure 8: Successful DNS spoofing attack

4.4 Remote DNS Attack targeting Authoritative Server [30 Marks]

4.5 Environment Setup

For this task, the attacker (**Internal-Attacker**) and DNS server need to be in different LANs. We need to move DNS server back to the Server LAN and revert the IP configurations to the original setting to match the Server subnet.

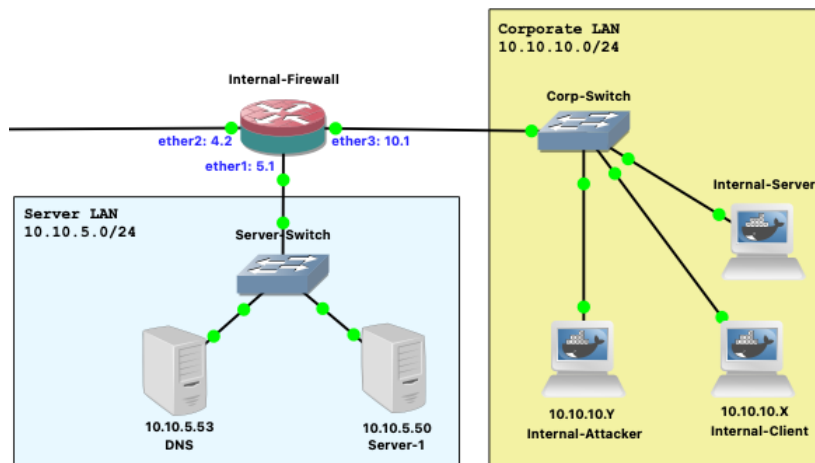


Figure 9: Topology for Remote DNS attack

The previous local DNS attacks assume that the attacker and the DNS victim server are on the same LAN so that the attacker can observe the DNS query message and reply with a forged DNS packet. When the attacker and the DNS server are not on the same LAN, the attack becomes harder since the attacker cannot perform ARP poisoning attack and see the DNS query. When the DNS victim server cannot resolve a DNS query, it forwards the DNS query packet to the forwarder DNS server (Google DNS server in our current setup). The DNS query is sent via a UDP packet where the UDP's source port is a 16-bit random number. In addition, the 16-bit transaction ID in the DNS header is also self-created by the DNS victim server. Hence, if the remote attacker wants to forge the DNS response, the forged packet must contain the correct values of these two numbers; otherwise, the reply will not be accepted.

Without being able to sniff the query packet, the remote attacker can only guess these two numbers. The chance is one out of 2^{32} for each guess. If an attacker can send out 1000 spoofed responses, it may take several days to try up 2^{32} time. In contrast, it only takes few seconds to receive the correct packet response from the forwarder Google DNS. Consequently, that real reply will be cached by the local DNS victim server. To make another try, the attacker has to wait for the server to send out another DNS query when its cache times out. Hence, this attacking chance makes the remote DNS attack unrealistic.

The remote DNS attack had become an open problem until Dan Kaminsky came up with a simple trick in 2008. The attack is depicted in the following figure.

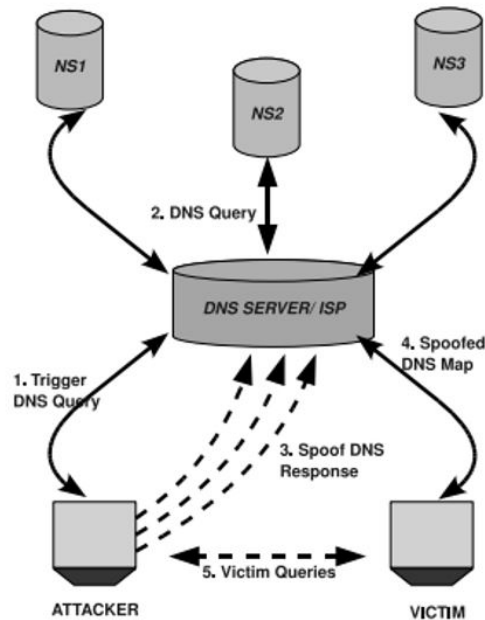


Figure 10: Kaminsky Attack

We choose a domain `test.com` as our targeted domain name in this task. When a client queries the DNS server for `www.test.com`, the attacker (**Internal-Attacker**) wants to cause the DNS server to use their DNS server (`ns.attacker.com`). The following steps with reference to above figure describe the outline of the attack.

1. The attacker queries the DNS server for a non-existing name in `test.com`, for example `xyz123.test.com`, where `xyz123` is a random name.
2. Since the mapping resolution cannot be resolved by the DNS server's cache, the server forwards the query to Google DNS (8.8.8.8) for that name resolution.
3. In the meantime, the attacker floods the DNS server with many spoofed DNS responses, each trying a different transaction ID and source port number (hoping one guess is correct). In that forged response, not only the attacker provides the IP resolution for the hostname `xyz123.test.com`, but also provides an authoritative name server for the domain `test.com`.

Even if the response failed, the attacker would go back to Step 1, and try another non-existing random name until the attacker succeeds.

4. Once the attack succeeds, when the client sends a DNS query to the poisoned DNS server for `www.test.com`, the nameserver returned by the DNS server will actually be set by the attacker.

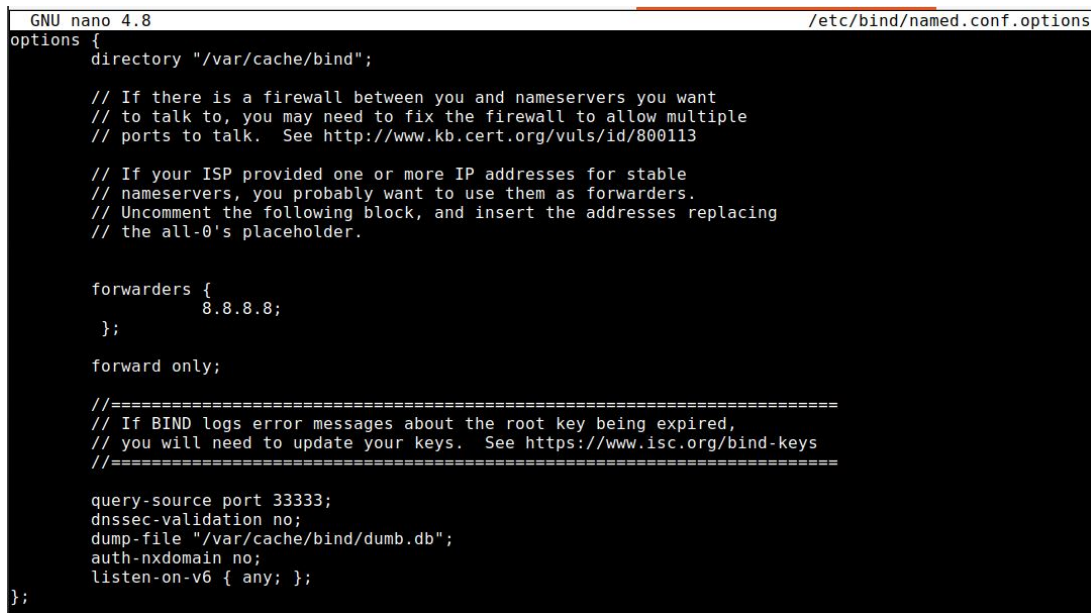
To simplify and shorten the attack's simulation time in this task, we suggest you follow the below steps before doing the task.

1. Double check the IP addresses of the server, attacker, and client to ensure the server and the attacker are not in the same LAN (using `ifconfig` command)
 - DNS: 10.10.5.53
 - Internal-Attacker: 10.10.10.X
 - Internal-Client: 10.10.10.X

2. In the DNS server's terminal, you can type the following command to configure DNS

```
nano /etc/bind/named.conf.options
```

Then, you should configure the forwarder 8.8.8.8, enable recursion and fix the query source port of the DNS server (i.e. 33333). With this constraint, the attacker now only needs to guess the transactionID of the DNS packet when performing remote DNS attacks. You can review the following figure for the correct configuration of DNS server.

A screenshot of a terminal window showing the configuration of the /etc/bind/named.conf.options file using nano 4.8. The file content is as follows:

```
options {
    directory "/var/cache/bind";

    // If there is a firewall between you and nameservers you want
    // to talk to, you may need to fix the firewall to allow multiple
    // ports to talk.  See http://www.kb.cert.org/vuls/id/800113

    // If your ISP provided one or more IP addresses for stable
    // nameservers, you probably want to use them as forwarders.
    // Uncomment the following block, and insert the addresses replacing
    // the all-0's placeholder.

    forwarders {
        8.8.8.8;
    };

    forward only;

    //=====
    // If BIND logs error messages about the root key being expired,
    // you will need to update your keys.  See https://www.isc.org/bind-keys
    //=====

    query-source port 33333;
    dnssec-validation no;
    dump-file "/var/cache/bind/dumb.db";
    auth-nxdomain no;
    listen-on-v6 { any; };
};
```

Figure 11: DNS server config file

3. After making the changes in the above step, you should restart your DNS server by using the following command

```
service named restart
```

We provide you the `remote_dns.py` script template on Moodle that helps to perform the Kaminsky attack.

4.6 Question 11

You need to complete Step 1 in the `remote_dns.py` to create 10000 dummy hostnames. Demonstrate and explain this task as before in your recorded video. **(2 marks)**

4.7 Question 12

You need to complete Step 2 in the `remote_dns.py` to generate a random DNS query for each dummy hostname. Demonstrate and explain this task as before in your recorded video. **(4 marks)**

4.8 Question 13

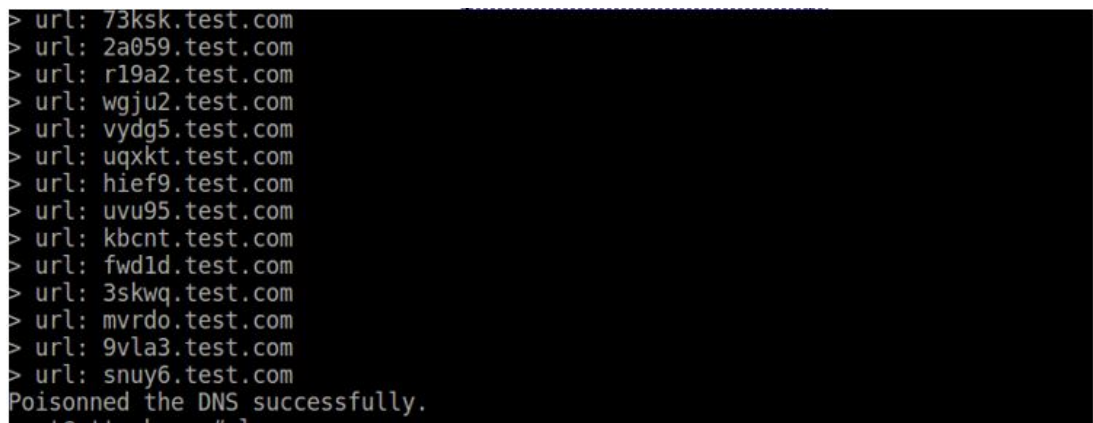
You need to complete Step 3 in the `remote_dns.py` to flood about 100 random forged response packets. Demonstrate and explain this task as before in your recorded video. The demo should show that

- Each packet has a randomly generated transaction ID for DNSpkt. **(5 marks)**
- The malicious DNS server `ns.attacker.com` is included in the nameserver authority for the domain `test.com` when you construct DNSpkt. **(2 marks)**
- *Additional section* showing `ns.attacker.com` has the IP of the attacker 10.10.10.X. **(2 marks)**

4.9 Question 14

Provide further video demonstration evidence to support and verify that you have performed the attack. In the video, you need to demonstrate the following key points:

- Wireshark traffic captured on DNS server on `eth1` shows the `transactionID` in DNS packet sent by DNS server to Google, and the correctly matched transaction ID in the forged packet sent by `Internal-Attacker` to the DNS server. **(5 marks for step by step explanation of the attack using Wireshark in the demonstration video)**
- Once the poisoning is completed, from `Internal-Client`'s terminal use `dig` command to send a DNS query for the specific subdomain for which the attack was successful (e.g. in below screenshot it's `issnuy6.test.com`). Do you get the attacker's IP? **(5 marks)**
- Now from the same terminal send a DNS query for `www.test.com`. If the attack was successful, the response should show `ns.attacker.com` in the *authority section* for the domain `test.com`. Was your attack successful? Explain in detail why or why not. (hint: you may refer to this paper for further information: https://www.cs.cornell.edu/~shmat/shmat_securecomm10.pdf) **(5 marks for your explanation during the demonstration video)**



```
> url: 73ksk.test.com
> url: 2a059.test.com
> url: r19a2.test.com
> url: wgju2.test.com
> url: vydg5.test.com
> url: uqxkt.test.com
> url: hief9.test.com
> url: uvu95.test.com
> url: kbcnt.test.com
> url: fwd1d.test.com
> url: 3skwq.test.com
> url: mvrdo.test.com
> url: 9vla3.test.com
> url: snuy6.test.com
Poisonned the DNS successfully.
```

Figure 12: Attacker's screen shows poisoning was successful

6 Acknowledgement

Parts of this assignment and instructions are based on the SEED project (Developing Instructional Laboratory for Computer Security Education) <https://seedsecuritylabs.org>.