

## CPSC 3600 HW2

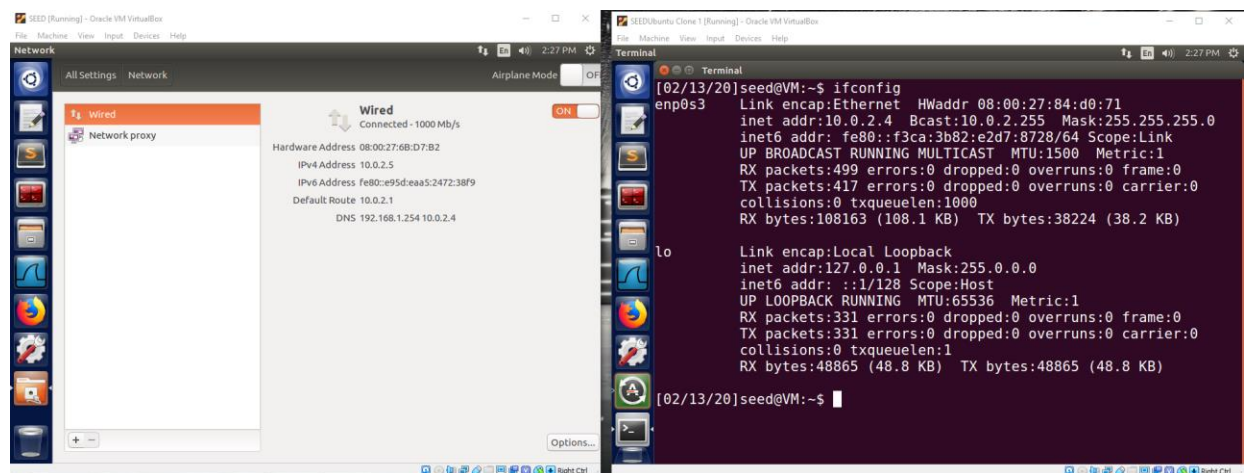
## Lab Task 1

### 2.1 - Configure the User Machine

Network settings in Ubuntu would not allow me to save changes after editing the settings, but I believe I got the process to work with a new DNS server using the “victim” machine’s IP address.

My User Machine is on 10.0.2.5

The “victim” second machine is on 10.0.2.4

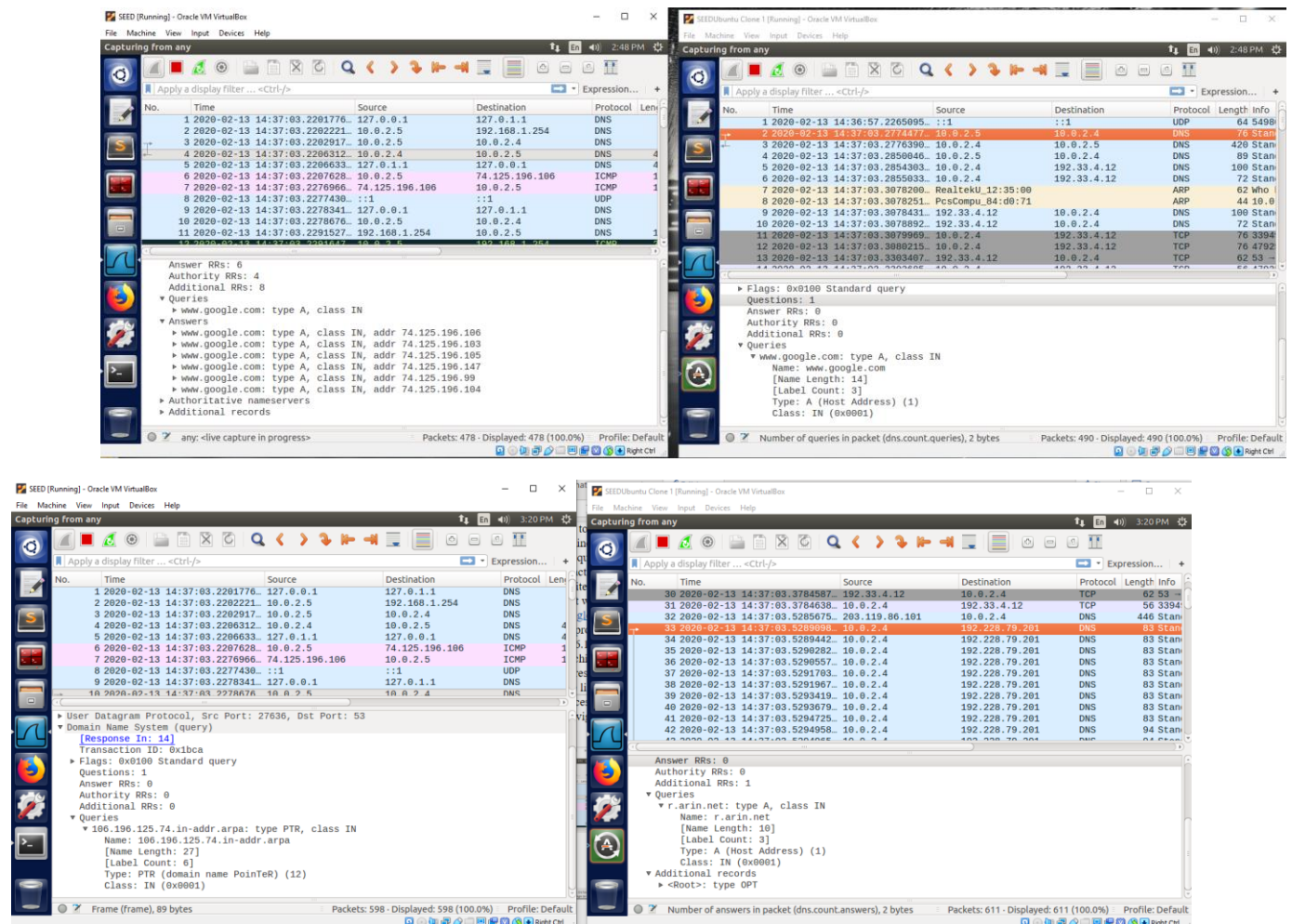


## Lab Task 2

### 2.2 - Setup a local DNS server

After the DNS server was applied to the user machine, I pinged [www.google.com](http://www.google.com) and used Wireshark to monitor both machines’ internet actions. From first observations, the user machine initially accessed the server through the DNS server of the second machine by querying the website. Hopping over to the second machine, one of the very first actions was a DNS request from the user machine asking for access to the website. Wireshark shows that the DNS query was of Type A, class IN. This means it was asking for an address and attempting to access the internet at Name: [www.google.com](http://www.google.com). Directly below the request, the second machine granted it access and provided the user machine with an address under the “Answers” tab of 74.125.196.106, along with multiple other similar

addresses. On line 5 of the second machine, the DNS server queries another server to verify/access the full name and address, resolving the query for the website. A DNS cache is used further down the list when the website is accessed for a second time. The system stored the access information in the cache file for further use and accesses it here for quicker navigation and transportation to the website.



## Lab Task 3

### 2.3 - Host a Zone in the Local DNS Server

After taking a minute to figure out that the only way to access and edit the named.conf file was through the terminal, I was able to access the file and edit it to include the new zone setup using my ip addresses.

The screenshot shows a terminal window titled "SEEDUbuntu Clone 1 [Running] - Oracle VM VirtualBox". The window has a menu bar with "File", "Machine", "View", "Input", "Devices", and "Help". The terminal output shows the configuration of the BIND DNS server, including comments and zone definitions.

```
// This is the primary configuration file for the BIND DNS server named.
// Please read /usr/share/doc/bind9/README.Debian.gz for information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// this configuration file.
// If you are just adding zones, please do that in /etc/bind/named.conf.local

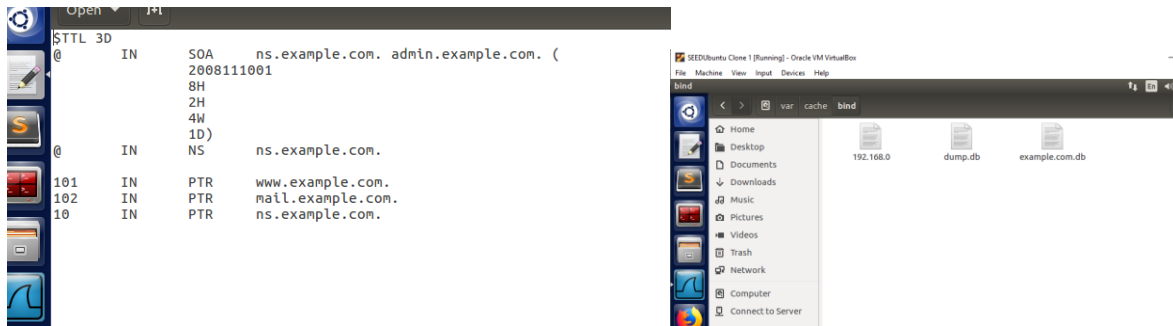
include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";

zone "example.com" {
    type master;
    file "/var/cache/bind/example.com.db";
};

zone "2.0.10.in-addr.arpa" {
    type master;
    file "/var/cache/bind/10.0.2";
};
```

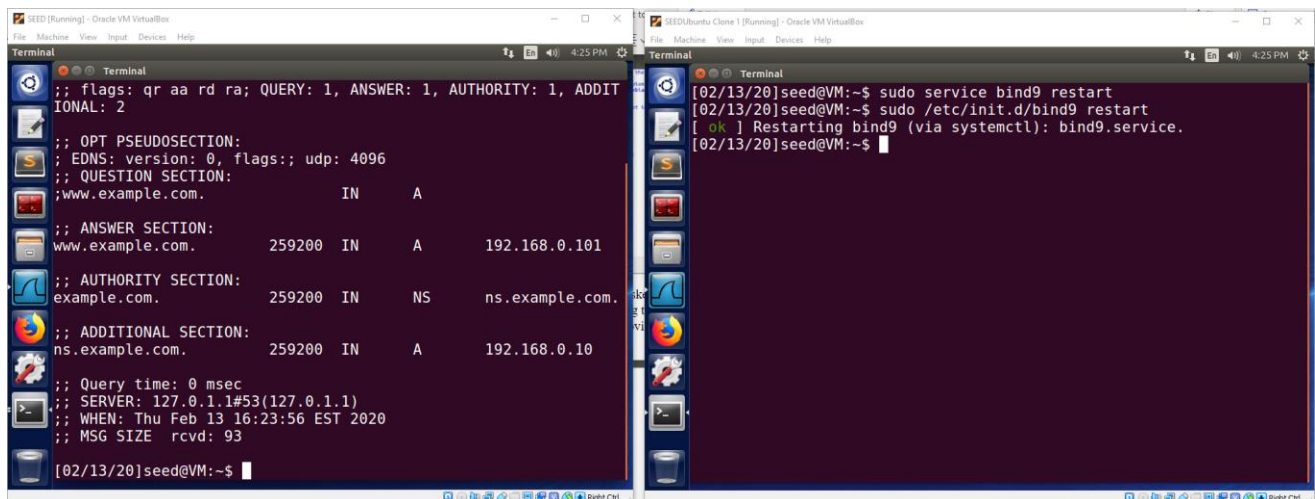
The terminal window also shows a sidebar with various application icons on the left and a status bar at the bottom indicating "Tab Width: 8", "Ln 7, Col 16", and "INS".

After creating the zones, I was then tasked to create two files in the /var/cache/bind directory to set up the zone files. Using the terminal, I created the two files and used the contents from the website provided to fill the files with the necessary code/information.



## 2.4 - Expected Output

After setting up the zones and doing a dig to the server at [www.example.com](http://www.example.com), the following was shown.

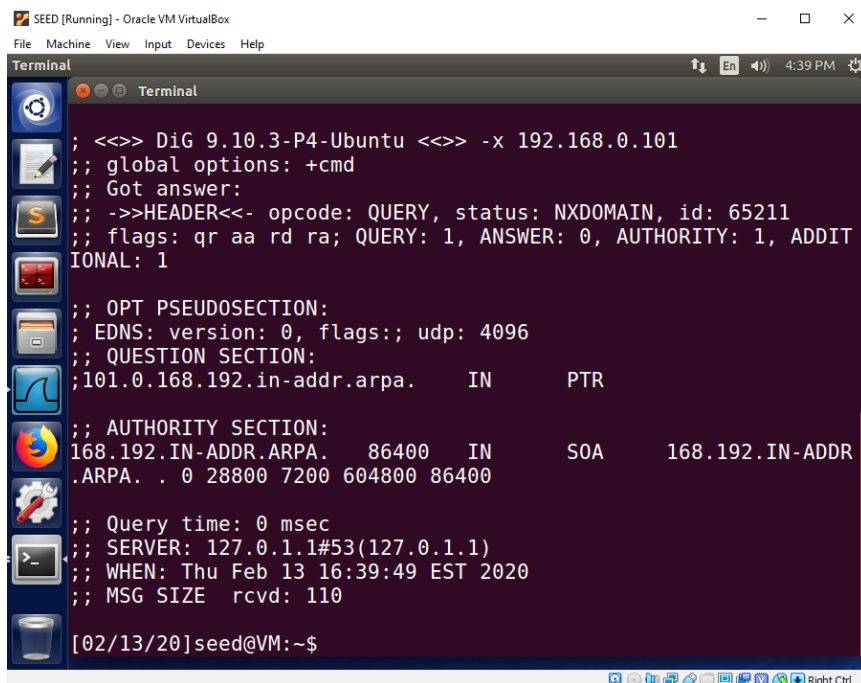


```
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.com.                IN      A
;; ANSWER SECTION:
www.example.com.                259200  IN      A      192.168.0.101
;; AUTHORITY SECTION:
example.com.                    259200  IN      NS      ns.example.com.
;; ADDITIONAL SECTION:
ns.example.com.                259200  IN      A      192.168.0.10
;; Query time: 0 msec
;; SERVER: 127.0.1.1#53(127.0.1.1)
;; WHEN: Thu Feb 13 16:23:56 EST 2020
;; MSG SIZE rcvd: 93
[02/13/20]seed@VM:~$

[02/13/20]seed@VM:~$ sudo service bind9 restart
[02/13/20]seed@VM:~$ sudo /etc/init.d/bind9 restart
[ ok ] Restarting bind9 (via systemctl): bind9.service.
[02/13/20]seed@VM:~$
```

The DNS dig shows that [www.example.com](http://www.example.com) now has an IP address of 192.168.0.101, which confirms that the DNS server has been successfully set up and has altered the address. The dig also provides us with the UDP port of 4096, additional IP addresses, and the server that it is running through.

Doing a reverse lookup using “dig -x 192.168.0.101” provided me with the following:



```
<<>> DiG 9.10.3-P4-Ubuntu <<>> -x 192.168.0.101
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 65211
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;101.168.192.in-addr.arpa.      IN      PTR
;; AUTHORITY SECTION:
168.192.IN-ADDR.ARPA.          86400  IN      SOA      168.192.IN-ADDR
.ARPA. . 0 28800 7200 604800 86400
;; Query time: 0 msec
;; SERVER: 127.0.1.1#53(127.0.1.1)
;; WHEN: Thu Feb 13 16:39:49 EST 2020
;; MSG SIZE rcvd: 110
[02/13/20]seed@VM:~$
```

This simply confirms that the user machine accessed [www.example.com](http://www.example.com) through the IP address 192.168.0.101 and attempted to query.



## 2.5 - Simplified Attack

In an attempt to simulate a DNS attack through modifying a compromised HOSTS file, I modified the HOSTS file to redirect [www.example.com](http://www.example.com) to the IP address owned by [www.google.com](http://www.google.com) (172.217.1.228). The following command line screenshot shows a ping attempt to [www.example.com](http://www.example.com) that immediately tries to access the IP address of [www.google.com](http://www.google.com), showing a successful “attack”

The screenshot shows two windows from an Oracle VM VirtualBox. The left window is a terminal running a ping command to [www.example.com](http://www.example.com). The output shows that the ping is successful, reaching the IP address 172.217.1.228, which is the IP for [www.google.com](http://www.google.com). The right window shows the contents of the `hosts` file, where [www.example.com](http://www.example.com) is mapped to 172.217.1.228.

```

Terminal
ing attribute metadata::gedit-encoding not supported
** (gedit:4683): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-position not supported
[02/13/20]seed@VM:~$ ping www.example.com
PING www.example.com (172.217.1.228) 56(84) bytes of data:
64 bytes from www.example.com (172.217.1.228): icmp_seq=1 ttl=5
1 time=26.8 ms
64 bytes from www.example.com (172.217.1.228): icmp_seq=2 ttl=5
1 time=26.4 ms
64 bytes from www.example.com (172.217.1.228): icmp_seq=3 ttl=5
1 time=26.6 ms
64 bytes from www.example.com (172.217.1.228): icmp_seq=4 ttl=5
1 time=26.8 ms
64 bytes from www.example.com (172.217.1.228): icmp_seq=5 ttl=5
1 time=26.5 ms
64 bytes from www.example.com (172.217.1.228): icmp_seq=6 ttl=5
1 time=26.5 ms
^C
--- www.example.com ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5009ms
rtt min/avg/max/mdev = 26.485/26.640/26.821/0.188 ms
[02/13/20]seed@VM:~$

hosts (/etc) - gedit
Open
127.0.0.1 localhost
127.0.1.1 VM
# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe80::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
127.0.0.1 User
127.0.0.1 Attacker
127.0.0.1 Server
127.0.0.1 www.SeedLabSQLInjection.com
127.0.0.1 www.xsslablgg.com
127.0.0.1 www.csrfllablgg.com
127.0.0.1 www.csrfllabattacker.com
127.0.0.1 www.repackagingattacklab.com
127.0.0.1 www.seedlabclickjacking.com
172.217.1.228 www.example.com

```

## UDP Checksum

Given the following binary sequences, calculate the checksum with the UDP procedure.

1. 10110100111101000
2. 01101101111000111
3. 11100111100111000

Step 1 is to use binary addition and add these three 16-bit segments.

Sum = (10)0000100111100111 - ( ) is the wraparound

Step 2 is to take the 1's complement of this sum.

Complement = 1111011000011000 - This is the checksum for the data.

The recipient will get the three data values as well as the checksum. The checksum includes the pseudo-header, UDP header, and data from the application layer.