

LOCAL DNS ATTACK

COMP 434 Local Dns Attack

PROJECT 3

CAN TAYLAN CAPRAZ - 36493

1. INTRODUCTION

DNS (Domain Name System) is the system that converts domain names to their IP address. When a user request a domain, DNS looks if the IP of this domain in its cache. If there is, it replies this request. If not, it asks other DNS servers. In DNS servers, an attacker can corrupt some information in the cache. Security of DNS servers are important since DNS attacks might allow attacker to collect bank account information and passwords of users. If a DNS cache poisoning attack is successful, then user connects the malicious server instead of requested trusted server.

2. ENVIRONMENT

In this project, SEEDUbuntu 12.04 is used in VirtualBox 5.1.20. VM image of SEEDUbuntu is obtained from website of SEEDLABS in Syracuse University ^[2]. In this virtual machine image, Bind9 Server is installed. Also, Netwag software is used to sniff IP.

3. CONFIGURATION

To establish a local network, three virtual machine is installed as seen in figure 1. One of these is for attacker, one of this is for DNS server, other one is for user. Since name of the each virtual machines are the same, they try to use the same UUID. Thus, their paths are set using following command:

```
VBOXMANAGE.EXE internalcommands sethduuid PathOfNewVMDK[3]
```

For each of three virtual machines, network setting is set to Host-Only-Adapter and Promiscuous mode is set as "Allow VM's". IP of these networks are obtained from terminal using "ifconfig".

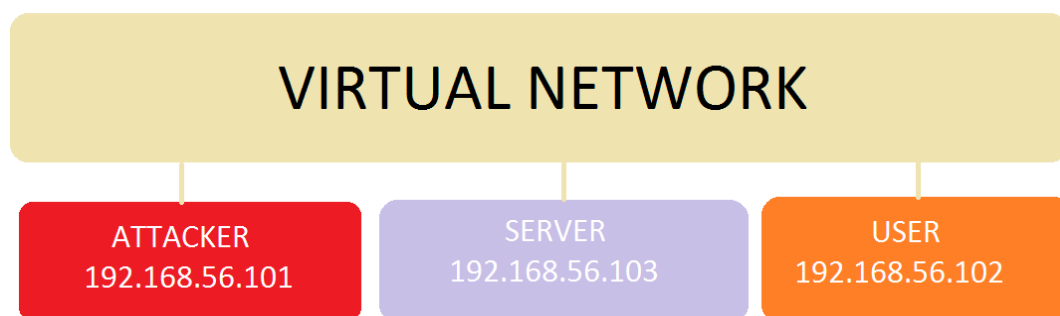


Figure 1: Project Setup

a. Configuration of DNS Server

Firstly, “dump-file “/var/cache/bind/dump.db”;" is added to named.conf.options file under /etc/bind /.

Then, zone is created in named.conf under the same directory. To do this, following code is added to content of named.conf file as mentioned on project description [4].

```
zone "example.com" {
    type master;
    file "/var/cache/bind/example.com.db";
};
zone "56.168.192.in-addr.arpa" {
    type master;
    file "/var/cache/bind/192.168.56";
};
```

After that, example.com.db and 192.168.56 files are created under /var/cache/bind/. Their contents are filled as follows as mentioned on project description [4].

Example.com.db	192.158.56
\$TTL 3D @ IN SOA ns.example.com. admin.example.com. (2008111001 8H 2H 4W 1D) @ IN NS ns.example.com @ IN MX 10 mail.example.com. www IN A 192.168.0.201 mail IN A 192.168.0.202 ns IN A 192.168.0.103 *.example.com. IN A	\$TTL 3D @ IN SOA ns.example.com. admin.example.com. (2008111001 8H 2H 4W 1D) @ IN NS ns.example.com. 201 IN PTR www.example.com. 202 IN PTR mail.example.com. 103 IN PTR ns.example.com.

Finally, DNS server is restarted using “sudo service bind9 restart” with terminal and configuration of DNS Server is completed.

b. Configuration of Attacker

In attacker's Virtual Machine, no configuration is done.

c. Configuration User

In user's Virtual Machine, DNS Server Virtual Machine is set as a default DNS server. To do this, "nameserver 192.168.56.103" line is added to file whose directory is /etc/resolv.conf. In order to avoid overwriting, "method" entry of "IPv4 Settings" is set to "Automatic(DHCP) Addresses Only" and "DNS Server" entry is set as "192.168.56.103" which is the IP address of DNS Server virtual machine.

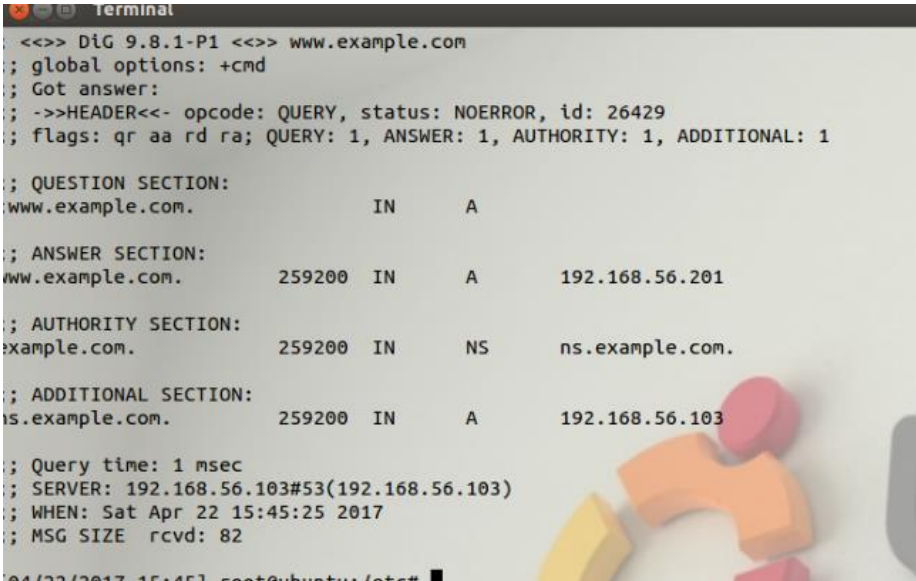
IPv4 Settings is accessed using following:

System Settings → Network → Wired → Options

Then, wired connection is reestablished with new properties and configuration of user's virtual machine is completed.

d. Configuration Control

When "dig www.example.com" command is run in terminal of user's virtual machine, following output seen in Figure 2 is established and it is seen that network is configured successfully.



```

<<>> DiG 9.8.1-P1 <<>> www.example.com
; global options: +cmd
; Got answer:
; ->HEADER<- opcode: QUERY, status: NOERROR, id: 26429
; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

; QUESTION SECTION:
www.example.com.                IN      A

; ANSWER SECTION:
www.example.com.                259200  IN      A      192.168.56.201

; AUTHORITY SECTION:
example.com.                    259200  IN      NS      ns.example.com.

; ADDITIONAL SECTION:
ns.example.com.                259200  IN      A      192.168.56.103

; Query time: 1 msec
; SERVER: 192.168.56.103#53(192.168.56.103)
; WHEN: Sat Apr 22 15:45:25 2017
; MSG SIZE rcvd: 82
04/22/2017 15:45 root@ubuntu: /etc#

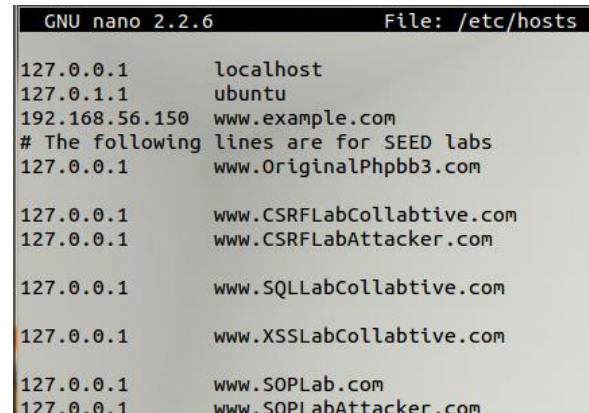
```

Figure 2: Output After Server Connection is Established

4. TASKS

a. Task 1

In this task, it is assumed that user's virtual machine is already hacked by attacker and attacker can modify files. Thus, as an attacker, I modified the HOSTS file, which keeps host name - IP address pairs, under /etc/hosts. "192.168.56.150 www.example.com" line is added to this server as seen in the figure 3. Then, ping is sent to



```

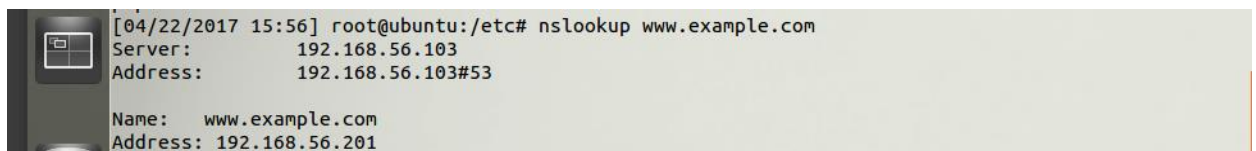
GNU nano 2.2.6 File: /etc/hosts
127.0.0.1 localhost
127.0.1.1 ubuntu
192.168.56.150 www.example.com
# The following lines are for SEED labs
127.0.0.1 www.OriginalPhpbb3.com

127.0.0.1 www.CSRFLabCollabative.com
127.0.0.1 www.CSRFLabAttacker.com

127.0.0.1 www.SQLLabCollabative.com
127.0.0.1 www.XSSLabCollabative.com

127.0.0.1 www.SOPLab.com
127.0.0.1 www.SOPLabAttacker.com
  
```

www.example.com and www.example.com is called via dnslookup. When dnslookup is called, it returns correct IP address 192.168.56.200 for www.example.com from server as seen in Figure 4. This is an expected output since there is no attack to DNS server in this part. However, when ping is sent, computer tried to connect wrong IP address 192.168.56.150 which we wrote in hosts file in /etc/ as it is seen in Figure 5. This shows that attack is successful.

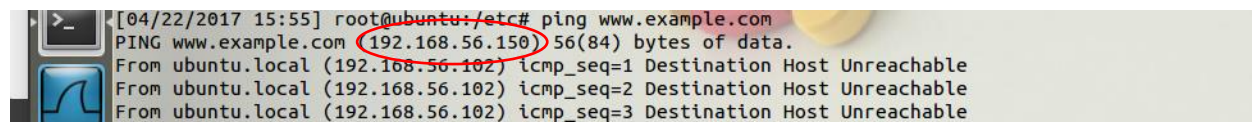


```

[04/22/2017 15:56] root@ubuntu:/etc# nslookup www.example.com
Server:      192.168.56.103
Address:     192.168.56.103#53

Name:   www.example.com
Address: 192.168.56.201
  
```

Figure 4: IP Address of www.example.com taken from DNS Server using nslookup



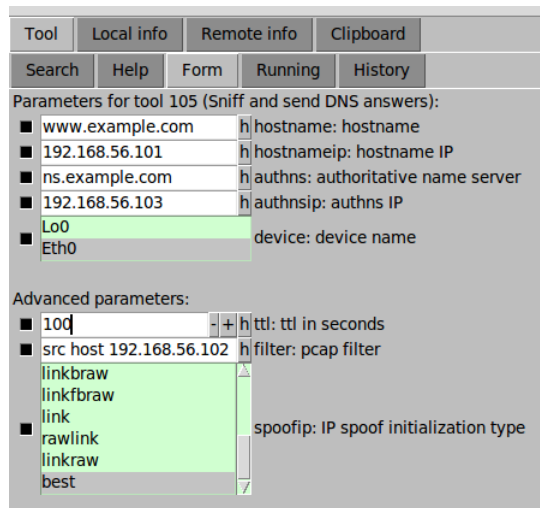
```

[04/22/2017 15:55] root@ubuntu:/etc# ping www.example.com
PING www.example.com (192.168.56.150) 56(84) bytes of data.
From ubuntu.local (192.168.56.102) icmp_seq=1 Destination Host Unreachable
From ubuntu.local (192.168.56.102) icmp_seq=2 Destination Host Unreachable
From ubuntu.local (192.168.56.102) icmp_seq=3 Destination Host Unreachable
  
```

Figure 5: Ping to www.example.com after attacking to host file of user

b. Task 2

The goal of this task is to sniff IP addresses sent from server to user. Before starting this task, “192.168.56.150 www.example.com” line in /etc/hosts is removed.



Netwag software is used to sniff packets. In netwag software, “Sniff and Sends DNS Answers” tool numbered as 105 is used. This tool is filled as seen in Figure 6. In form, hostname part is filled as www.example.com and its IP address is written as 192.168.56.101 which is the IP address of attacker’s virtual machine. Nameserver is set as ns.example.com and its IP is set as 192.168.56.103 which is the IP address of DNS Server’s virtual machine. Eth0 is

Figure 6: netwag for DNS Spoofing chosen since connection of virtual machines are established via this. TTL time is set to 100, which makes sniffed packets usable for 100 second. Then, packets go to host is filtered by writing “src host 192.168.56.102” to filter part since user’s IP address is 192.168.56.102. In the output shown in Figure 7, most results for nslookup returns as the IP address of attacker. It shows that attack is successful. Some of the nslookup requests returns correct IP address since netwag couldn’t catch all packets.

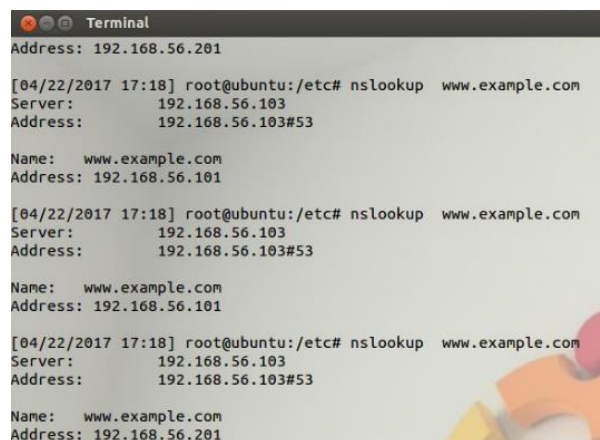


Figure 7: Output with DNS Sniffing

c. Task 3

In this task, aim is to poison DNS Server's cache.

When a client requests IP address of a domain, DNS Server first looks the cache and sends record if there is. If there is not a record for this domain, DNS server asks the IP Address to other servers.

In task 2, some DNS requests are sniffed. However, not all DNS packets are cached. Also, when attacker's virtual machine is turned off, sniffing stops. If DNS cache is modified with wrong IP for a domain name, DNS server returns this wrong IP address to all requests, so attacker doesn't need to try to cache all packets. DNS Server can be poisoned if there is no record for a domain in its cache. Assume that a user requests for a domain such as "www.trustedwebsite.com" and this domain is not in the cache of DNS Server. DNS Server will ask the IP of this domain to other DNS Servers. If a response comes to our DNS Server is sniffed, DNS Server get response contains wrong IP address and save it in its cache. Then, it sends this IP address to each request.

Before starting this task, Nat Network adapter is added to attacker and server. To create a NAT Network, following steps are applied:

Files → Preferences → Network → Adds New NAT Network.

Then, adapter 2 part in network setting of attacker and DNS Server is set to

Network

Adapter 1 Adapter 2 Adapter 3 Adapter 4

☒ Enable Network Adapter

Attached to: NAT Network

Name: NatNetwork

Advanced

Adapter Type: Intel PRO/1000 MT Desktop (82540EM)

Promiscuous Mode: Allow VMs

MAC Address: 080027748617

☒ Cable Connected

Port Forwarding

NAT Network and Promiscuous mode is set as "Allow VM's" as seen in Figure 8.

This network is added to allow DNS Server to asks IP addresses to other servers if it doesn't have in its cache. Attacker is added to this network to be able to sniff packets coming from other server via this network.

Figure 8: Network Configuration for NAT Network

After doing these additional configurations, DNS cache is flushed using the following command.

```
sudo rndc flush
```

Then, 105th tool in netwag is used to sniff packets coming to DNS Server. In this task, it is assumed that request is sent to `www.facebook.com` and there is a server controlled by attacker in `192.168.56.155` and its name server is in the `192.168.56.156`. And netwag's form is filled using these informations as shown in Figure 9. As a device, `Eth1` is chosen since it is NAT network and packets coming via this adapter is necessary to be shifted. Ttl is set as 600 second in order to keep this attacked IP-domain pair in the cache longer.

Parameters for tool 105 (Sniff and send DNS answers):

■ <code>www.facebook.com</code>	h hostname: hostname
■ <code>192.168.56.155</code>	h hostnameip: hostname IP
■ <code>ns.facebook.com</code>	h authns: authoritative name server
■ <code>192.168.56.156</code>	h authnsip: authns IP
Lo0	
■ <code>Eth0</code>	device: device name
<code>Eth1</code>	

Advanced parameters:

■ <code>600</code>	- + h ttl: ttl in seconds
■ <code>src host 10.0.2.4</code>	h filter: pcap filter
<code>raw</code>	
<code>linkf</code>	
<code>linkb</code>	
■ <code>linkfb</code>	spoofip: IP spoof initialization type
<code>rawlinkf</code>	
<code>rawlinkb</code>	

Spoofip method is set as raw in order to sniff query in shorter time since server shouldn't catch the query before sniffing. Raw type makes sniffing without modifying MAC address. Filter part is filled using the IP address of DNS Server in NAT Network since it is necessary to sniff query comes to DNS server.

Figure 9: Netwag Form for DNS Cache Poisoning

When it is run, expected response of “`nslookup www.facebook.com`” is `192.168.56.155`. Also, this IP and domain name pair is expected to cache. To learn if attack is successful or not, cache file is read using following commands:

```
sudo rndc dumpdb -cache
sudo cat /var/cache/bind/dump.db
```

In the cache file, it is seen that pair is added as seen in Figure 10.

```
PVGC15 NS DS RRSIG
; answer
www.facebook.com.      81      A      192.168.56.155
; glue
```

Figure 10: Cache of DNS Server After Attack

After tool is closed, www.facebook.com is requested many time and all of these requests are responded as 192.168.56.155. After 10 minutes, since 600 second is finished, this pair is removed from cache and requests started to be responded with real IP address www.facebook.com.

5. ADDITIONAL INFORMATION

In this part, it will be discussing if this attack can be done from outside of the server or not. This is much more difficult to spoof IP since DNS responses cannot be sniffed by a sniffer. However, it is still possible to attack DNS cache.

When DNS Servers request and IP address from other DNS Servers, it needs to verify if the responses are coming from trusted DNS Servers which IP is requested from. If there were no verification method, any attacker can convince DNS server for wrong IP address. In order to avoid this kind of attacks, DNS server creates transaction ID and add in into the packets. It also adds a port number that it will accept responses in. When a response comes, DNS server first checks transaction ID and source port. If they are correct, it accepts the response as a valid response.

As an outsider attacker, only way to attack DNS server is guessing transaction ID and source port. If attacker can guess them correctly, it can convince DNS Server with wrong IP address.

6. REFERENCES

- [1] Mohn E. Domain Name System (DNS). Salem Press Encyclopedia Of Science [serial online]. 2015; Available from: Research Starters, Ipswich, MA. Accessed April 23, 2017.
- [2] "SEED Project," SEED Project. [Online]. Available: http://www.cis.syr.edu/~wedu/seed/lab_env.html
- [3] "How to use VirtualBox to Run Our Pre-built VM Image?" [Online]. Available: <http://www.cis.syr.edu/~wedu/seed/Documentation/VirtualBox/LoadingMultiVMs.pdf>