

ECE576/676

Lab 3 – Local DNS Attack

Haosong Liu

Lab Goals

1. Explore how DNS works;
2. Set up a DNS server;
3. Perform DNS cache attack;
4. Spoofing DNS responses;

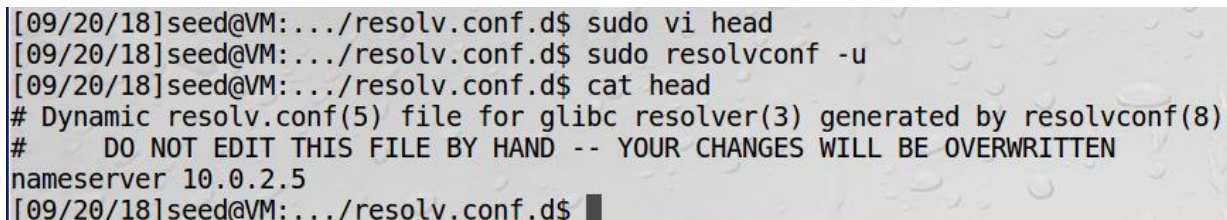
Lab Setup

Kali Linux (10.0.2.15) was used as the attacker machine, one of the Seed machine (Seed, 10.0.2.4) served as user machine, the other Seed machine (SeedClone, 10.0.2.5) served as Local DNS server. Before doing any attack, set all three machines under the same LAN and allow all traffic to be seen for all three machines.

The server and the user machine were configured according to lab manual, details are as follow:

1. DNS server set-up and user machine configurations

First, change the resolve configuration file of the user machine, make SeedClone as its local DNS server. Fig 1 proves that the changes were made and were taken effect.



```
[09/20/18]seed@VM:~/resolv.conf.d$ sudo vi head
[09/20/18]seed@VM:~/resolv.conf.d$ sudo resolvconf -u
[09/20/18]seed@VM:~/resolv.conf.d$ cat head
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 10.0.2.5
[09/20/18]seed@VM:~/resolv.conf.d$
```

Fig 1: User machine DNS resolve file changed, use SeedClone as local DNS server now

Next, on the server side, DNS server was configured and started. In particular, BIND9 server was used and a zone and a reverser look-up zone was hosted at the server machine. After all these configurations were done. Test the result using dig command to dig www.google.com on User machine. The proof of successful set-up is that, the server now changed to 10.0.2.5, which is the local DNS server we host. See Fig 2 on next page for proof.

```
[09/20/18]seed@VM:/etc$ dig www.google.com

; <<> DiG 9.10.3-P4-Ubuntu <<> www.google.com
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 42235
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 4, ADDITIONAL: 9

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:;, udp: 4096
;; QUESTION SECTION:
;www.google.com.                IN      A

;; ANSWER SECTION:
www.google.com.      300     IN      A       74.125.138.147
www.google.com.      300     IN      A       74.125.138.99
www.google.com.      300     IN      A       74.125.138.105
www.google.com.      300     IN      A       74.125.138.103
www.google.com.      300     IN      A       74.125.138.104
www.google.com.      300     IN      A       74.125.138.106

;; AUTHORITY SECTION:
google.com.          172800  IN      NS      ns1.google.com.
google.com.          172800  IN      NS      ns4.google.com.
google.com.          172800  IN      NS      ns2.google.com.
google.com.          172800  IN      NS      ns3.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.      172800  IN      A       216.239.32.10
ns1.google.com.      172800  IN      AAAA    2001:4860:4802:32::a
ns2.google.com.      172800  IN      A       216.239.34.10
ns2.google.com.      172800  IN      AAAA    2001:4860:4802:34::a
ns3.google.com.      172800  IN      A       216.239.36.10
ns3.google.com.      172800  IN      AAAA    2001:4860:4802:36::a
ns4.google.com.      172800  IN      A       216.239.38.10

;; Query time: 1972 msec
;; SERVER: 10.0.2.5#53(10.0.2.5)
;; WHEN: Thu Sep 20 15:14:00 EDT 2018
;; MSG SIZE rcvd: 387
```

Fig 2: Dig result proved that the DNS server now changed to our local DNS server 10.0.2.5

After the setup, we are ready to perform DNS attacks. The procedures and observations as well as any interpretations and explanation are in the next section: Lab Results.

Lab Results

1. Attack by modifying Host file on compromised user machine

Suppose the user machine has already been compromised, we can modify the host file directly so that a domain name of our choice will be resolved as an IP address of our choice. Fig 3 shows the result of adding an entry of 204.79.179.200 www.bank32.com in the host file. This entry will make the victim machine to resolve www.bank32.com as 204.79.179.200, which is the IP of www.bing.com. To test the effect, dig www.bank32.com after the changes made and we should expect the IP of www.bank32.com shown in the response to be 204.79.179.200. This is indeed the case, thus, the attack succeeded.

```

204.79.197.200  www.bank32.com
#10.0.2.15      www.example.net
#204.79.197.200 www.google.com
[09/27/18]seed@VM:/etc$ ping www.bank32.com
PING www.bank32.com (204.79.197.200) 56(84) bytes of data.
64 bytes from www.bank32.com (204.79.197.200): icmp_seq=1 ttl=116 time=1.
64 bytes from www.bank32.com (204.79.197.200): icmp_seq=2 ttl=116 time=1.
64 bytes from www.bank32.com (204.79.197.200): icmp_seq=3 ttl=116 time=1.
64 bytes from www.bank32.com (204.79.197.200): icmp_seq=4 ttl=116 time=3.
64 bytes from www.bank32.com (204.79.197.200): icmp_seq=5 ttl=116 time=1.
^C
--- www.bank32.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 34.577/98.759/122.856/32.545 ms
[09/27/18]seed@VM:/etc$

```

Fig 3: DNS attack by directly modifying host file in victim's machine succeed, user now resolve www.bank32.com as 204.79.197.200 which is in fact, www.bing.com

2. Attack by directly spoofing response to user

In this task, first run “sudo rndc flush” command on the server to clear up DNS cache so that victim machine will not receive the cached DNS resolve results from the DNS server. Then a fake DNS spoofing packet was sent using netwag as shown in Fig 4 and 5. The original IP of Google and Bing are shown in Fig 6 and Fig 7 for comparison. The attack was not successful until the server been manually shutdown so that it was unreachable to the user, making it possible for the faked spoofing packets to be received by the User machine (the victim). After the spoofing, as user now got a fake DNS resolution in its cache, it would stop asking the server for resolution but use the faked cache instead. This was verified by digging www.google.com on User machine and got back a response showing the domain name being resolved as google's IP, which is what the attacker's goal, thus this proves that the attack was succeeded.(Fig 4 – Fig 7 are on the next two pages)

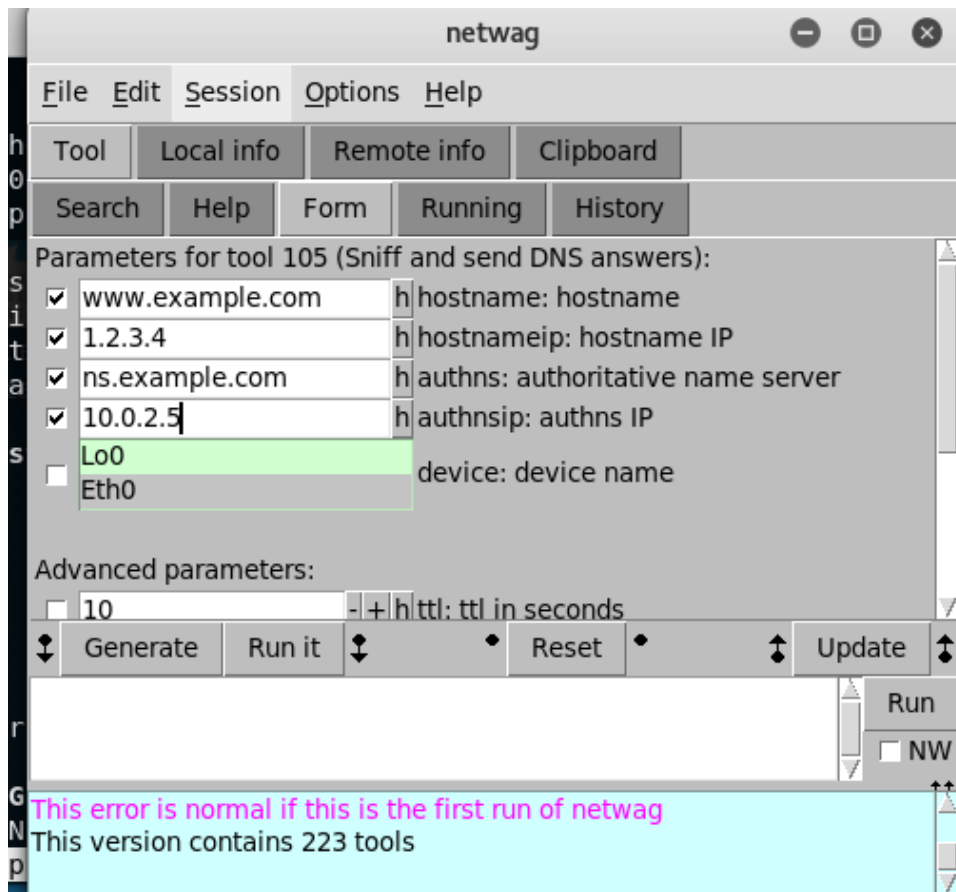


Fig 4: Use netwag to send spoof packet, first line is the domain name to attack on, second line is the desired misleading IP address you wish to let the user machine to resolve domain name as, third line is the hosted name server, fourth line is the machine to attack on, in our case, the user's IP was 10.0.2.4

```
[09/27/18]seed@VM:/etc$ dig www.google.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 45607
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 13, ADDITIONAL: 27

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.google.com.                IN      A

;; ANSWER SECTION:
www.google.com.                7       IN      A      204.79.197.229
```

Fig 5: Proof of successful attack on User machine. In the screenshot, when user dig Google, they get back the resolved name as 204.79.197.229 which is the IP of www.bing.com


```

;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 32513
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.bing.com. IN A

;; ANSWER SECTION:
www.bing.com. 4 IN CNAME afddnsperfctest-a.trafficmanager.net.
afddnsperfctest-a.trafficmanager.net. 5 IN CNAME www-bing-com.a-0026.a-msedge.net.
www-bing-com.a-0026.a-msedge.net. 185 IN CNAME a-0026.a-msedge.net.
a-0026.a-msedge.net. 185 IN A 204.79.197.229

;; Query time: 8 msec
;; SERVER: 10.50.50.100#53(10.50.50.100)
;; WHEN: Thu Sep 27 15:57:40 EDT 2018
;; MSG SIZE rcvd: 163

```

Fig 6: Real IP of www.bing.com : 204.79.97.229

```

root@kali:~# dig www.google.com

; <<>> DiG 9.11.3-1-Debian <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 15849
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.google.com. IN A

;; ANSWER SECTION:
www.google.com. 98 IN A 172.217.15.196

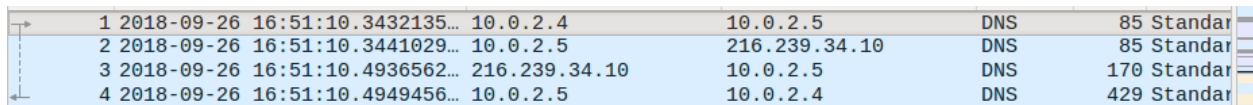
;; Query time: 17 msec
;; SERVER: 10.50.50.100#53(10.50.50.100)
;; WHEN: Thu Sep 27 15:57:06 EDT 2018
;; MSG SIZE rcvd: 59

```

Fig 7: Real IP of www.google.com : 172.217.15.196

3. Attack by poisoning DNS cache on Server

Another way to do DNS attack is to attack the server DNS cache directly, so that any user machine that use the DNS service provided by that particular DNS server would be affected as DNS server would not check outside for DNS resolution if the domain name has already existed in the cache file on server(the name will be resolved as described in the cache file without further checking). This means, if we can poison the server's DNS cache, next time a user asks for resolution, the server would give the fake resolution we injected into the server as the correct result to the user. This use-cache-if-exist behavior can be verified by digging the same website twice and see the differences in the DNS query packets captured by Wireshark, as shown and explained in Fig 8 and 9 below:



1	2018-09-26 16:51:10.3432135...	10.0.2.4	10.0.2.5	DNS	85 Standard
2	2018-09-26 16:51:10.3441029...	10.0.2.5	216.239.34.10	DNS	85 Standard
3	2018-09-26 16:51:10.4936562...	216.239.34.10	10.0.2.5	DNS	170 Standard
4	2018-09-26 16:51:10.4949456...	10.0.2.5	10.0.2.4	DNS	429 Standard

Fig 9: In this figure, the server (10.0.2.5) did not know any resolution of any domain name as its cache was flushed manually. The DNS query first went from user (10.0.2.4) to Server (10.0.2.5), the server then queried 216.239.34.10 for resolution and 216.239.34.10 responded to the server, so server gave this information to the client (last line in the screenshot is the response from server to client)



11	2018-09-26 16:52:21.8704011...	10.0.2.4	10.0.2.5	DNS	85 Standard
12	2018-09-26 16:52:21.8711780...	10.0.2.5	10.0.2.4	DNS	429 Standard

Fig 10: Proof of DNS cache on server was used, after the first dig was done, the server cached the result and when second dig on the same website was performed, the server responded to the client immediately without asking any external DNS servers for result (as it had the first query result cached)

As we confirmed that once the DNS resolution is cached on server, any future queries on the same domain name would get a response directly from the server, the server would not check correctness of the cached content. This makes it possible to perform DNS attack in another way: poisoning the server's DNS cache directly.

The step is the same as in section 2, use netwag to perform the attack this time targeting the server. The effect was the same so readers can just refer to Fig 5-7 for proof.

Appendix A – Summary of commands (and options) used with their meaning

1. dig <domain name you wish to dig> : get the detailed domain name information back including DNS server information, resolved IP information and the status of the query(NoError or SERVFAIL), this command is mainly for testing the success of the attack in this lab

2. sudo service bind9 stop: stop the BIND9 server

3. sudo service bind9 restart: restart the BIND9 server

4. sudo resolvconf -u: for changes in resolv.conf to take effect

5. netwag: A UI for DNS spoofing and more. Choose 105 for sending DNS spoofing packets, in the fields from top to bottom, enter the domain name you wish to attack, the IP you wish the domain name to be resolved as, authoritative nameserver and victim's IP address