ASSIGNMENT 3

Sanil Sinai Borkar
{*sanilborkar@ufl.edu*}

October 13, 2014

For the first three tasks, **SEEDUbuntu** was the user, **DNS** was the DNS server and **Attacker** was the attacker.

1 HOSTS File Attack

- **Describe the exact steps you took to alter the the IP address that was resolved for a given host.** Since the victim's computer is compromised, the attacker can easily modify the hosts file (/etc/hosts). For the attack ,the hosts present in /etc/ was modified to add the IP address of the malicious site that the attacker wants the victim to be re-directed to. For this, the following entry was added

  1.2.3.4 www.example.com

  Since the hosts file takes up preferrence over the DNS query, whenever the user tried to reach out to *www.example.com*, he/she will be re-directed to the site having an IP address of *1.2.3.4* as per the entry made in the hosts file.

- **Provide screen shots, commands executed, and command output to explain what took place.** First, the IP address of the 'fake' website (1.2.3.4 in this case) is added to the /etc/hosts file as can be seen in Figure **??**:

  1.2.3.4 www.example.com

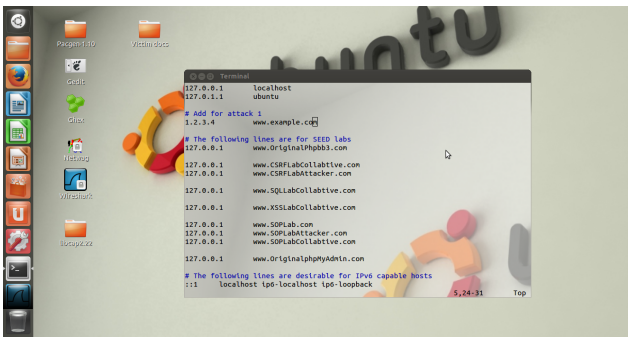

Figure 1: Modify HOSTS file

When the site given by *www.example.com* is to be reached, a timeout is experienced as there is no site having an IP address of *1.2.3.4* as can be seen in Figure 2:

As can be seen above, *www.example.com* was pinged but all the packets were lost. The IP address of *www.example.com* has been modified to *1.2.3.4* as can be seen in the screenshot above.

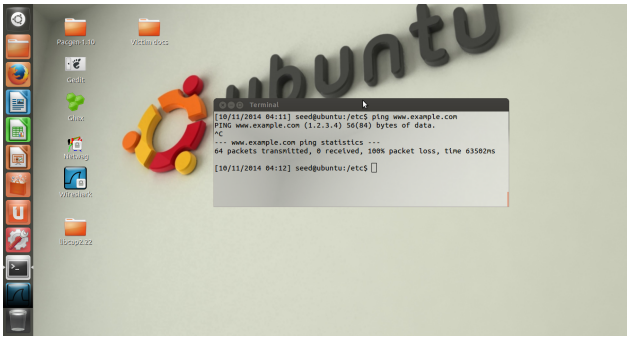

Figure 2: Ping Response After Attack

- **Explain how a real-world attacker could leverage this sort of attack.**
  In this case, the users computer has been compromised. The user can enter any IP address in the hosts file and make the user re-direct to this site easily. This could be done for multiple sites. The user can easily be fooled this way into believing that this is the real site that they had asked for. If this site is made to look genuine to the actual site (for example, a forged banking site), the user can be ealisy fooled to give away his confidential information viz. his username, password etc.

- **Discuss whether you feel this is a viable attack in the real world. What factors contribute to your answer?**
  This attack is a viable one but only to a certain extent as it is possible only if the attacker has control over the user's machine, which is a tough task. In the real world, the difficulty of getting full control over the user's machine is what makes this attack execution and success a bit unfeasible.

2 Host-Level Response Spoofing

- **Describe the exact steps you took to alter the the IP address that was resolved for a given host. Be sure to provide evidence that shows that the host did, indeed, resolve an incorrect IP address. This should come in the form of packet captures, command line results, etc.**
  Initially, we perform a dig to our site *www.example.com*. This returns the IP address that is set for *example.com* zone in the zone file on the DNS server as seen in Figure 3
  After this, the netwox 105 tool was run to sniff and send DNS answers. The following command was executed on the command line in the User VM:
  sudo netwox 105 –hostname "www.example.com" –hostnameip 1.2.3.4 –authns "ns.example.com" –authnsip 1.2.3.5 –device "Eth1"

  The same can be even configured using the GUI as seen in Figure 4:
  The above command was then run in the command prompt to generate the required results (Figure 5):
  As seen above, the IP address returned by dig was *www.example.com* is now changed to *1.2.3.4* that we sent using the netwox tool. The same result can be achieved using the GUI configuration as mentioned above. The GUI output is given in Figure 6:
  Therefore, our attack worked successfully.

- **Provide screen shots, commands executed, and command output to explain what took place.**
  The steps and the screenshots that were taken have been provided in the previous point. However,

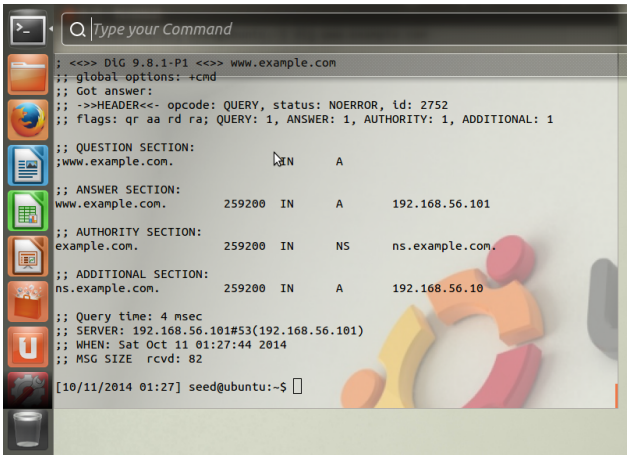

Figure 3: Before Attack - dig www.example.com

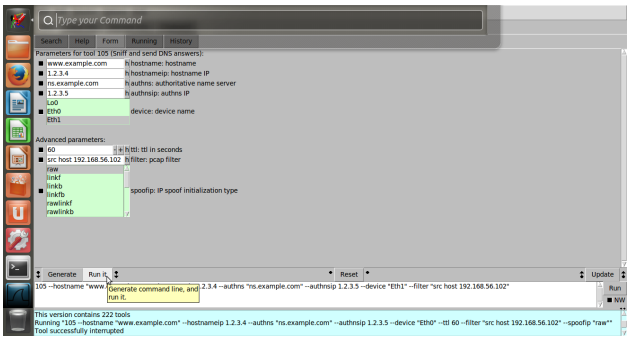

Figure 4: Before Attack - Netwag 105 tool GUI configuration

there is a netwox trace file present at */Task1/netwox trace for example* after the attack was launched.

- **Explain how a real-world attacker could leverage this sort of attack.**
  If the real-world attacker is on the same LAN as the victim, he/she can sniff the DNS request message sent by the victim. They can then create a fake DNS response, and send it back to the victim, before the real DNS server does. This could be done theoretically for every DNS query made by the victim computer. The damage is more as the attacker is on the same LAN as that of the victim and can sniff on the DNS requests sent by the victim.

- **Discuss whether you feel this is a viable attack in the real world. What limitations about this attack contribute to your answer?**
  It is viable attack in the sense that it can be leveraged to create maximum damage to the victim. One of the most important aspects to consider here are that the attacker should be able to sniff the DNS requests, without which he/she would not know when a DNS request was sent by the victim's computer. Secondly, the attacker and the victim should be on the same network. If you, as an attacker, needs to cause such damage to another person, you need to be on the victim's network, which is a tough task in the real-world. This adds to the invialibilty of this attack in the real-world.

3 Server-Level Response Spoofing

- **Describe the exact steps you took to alter the the IP address that was resolved for a given host.**

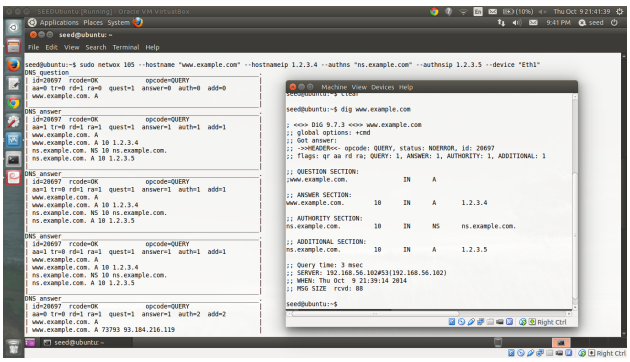

Figure 5: Run netwox 105 tool on Command Prompt

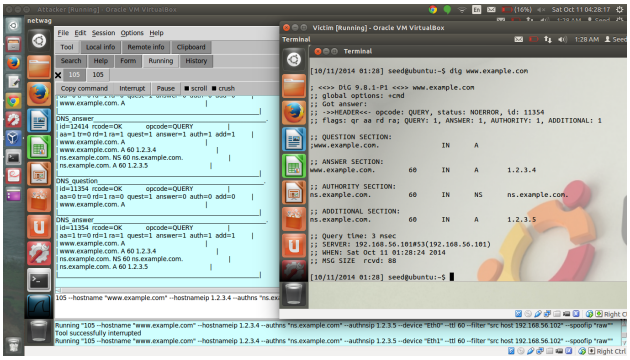

Figure 6: After Attack - dig www.example.com

Be sure to provide evidence that shows that the host did, indeed, resolve an incorrect IP address. This should come in the form of packet captures, command line results, etc.
To start with, before the attack, to be sure about the results, a 'dig' was performed on the site in question, in this case go-rts.com. This is shown in Figure 7

The DNS server cache dump after the 'dig go-rts.com' explained above can be seen in Figure 8
In this case, we are targetting the DNS server directly to poison its cache. For this, netwox was used to sniff the DNS request packets and then send answers. In the netwox GUI, we need to set the hostname that will be used for this attack followed by its 'fake' IP address. The name server and its 'fake' IP address also needs to be specified. Then, we specify the device interface over which the packets will be sent/received. We even specify the Time-To-Live (TTL) as 60 secs. We also filter the source to be the DNS server's IP address as we are spoofing the response on the DNS server. This netwox configuration explained above can be seen in Figure 9
When we click on 'Run' and then perform a 'dig', we can see the DNS requests and response in the netwox GUI, and the same spoofed response shows up on the 'dig' response, as can be seen in Figure 10
To be sure that the DNS server cache has been poisoned, the DNS server cache dump was taken. We can see that it has an entry *1.2.3.4* for *go-rts* in Figure 11, as expected.

- **Provide screen shots, commands executed, and command output to explain what took place.**
  All the screenshots and output are mentioned in the sub-section above. Moreover, the netwox trace for the attack is placed at *Task2/netwox trace.txt*

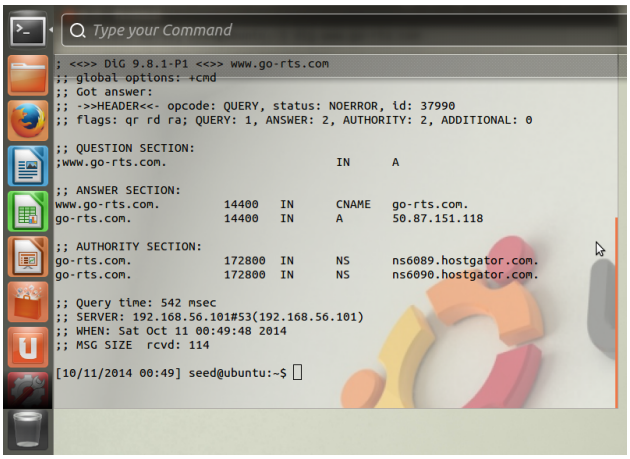

Figure 7: Before Attack - Dig go-rts.com

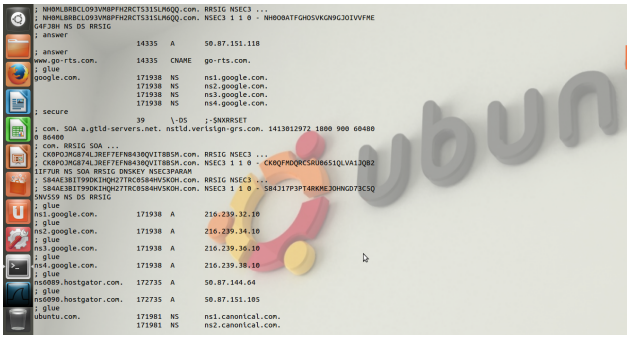

Figure 8: Before Attack - DNS Server Cache after dig go-rts.com

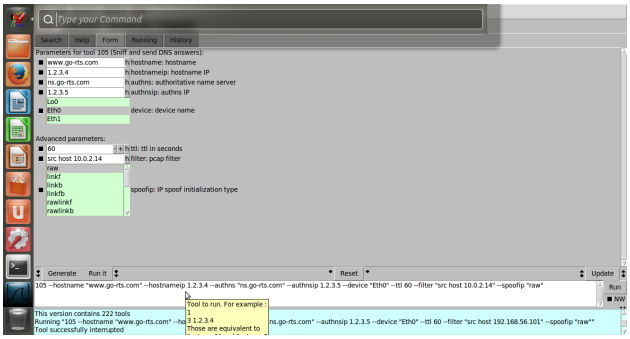

Figure 9: Before Attack - Netwox Configuration

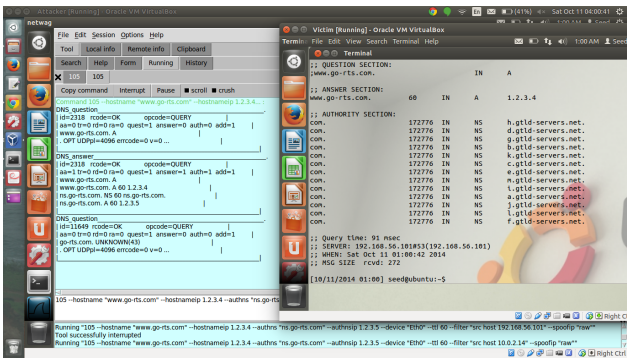

Figure 10: After Attack - dig and Netwox

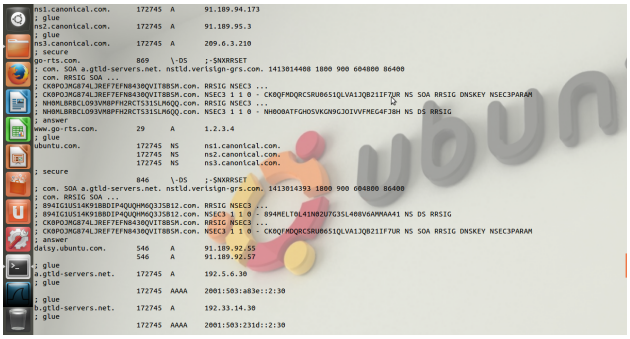

Figure 11: After Attack - DNS Server Cache Dump

- **Explain how a real-world attacker could leverage this sort of attack.**
  Since this attack is directed at the DNS Server in order to poison its cache, this needs to be done only once as compared to the previous case. But once it is done, anyone and everyone who tries to visit the compromised site will be re-directed to this 'fake' one. So this is a Server-level attack. Moreover, if the TTL could be set to a larger value, the impacted site can be compromised for a longer period of time. During this time all the users trying to access this site will be re-directed to a 'fake' site.

- **Discuss whether you feel this is a viable attack in the real world. What limitations about this attack contribute to your answer?**
  This is a viable attack in the real-world in the sense that it can be used to poison the DNS Server cache and then affect anyone who tries to access the compromised site. The limitation in this attack is that the attacker should be able to observe and sniff on the DNS request packets. Moreover, the attacker has to make sure that all the required attributes in the genuine DNS request (transaction ID, source/destination IP addresses and port numbers etc.) match with those in the 'fake' response. Having done this, he/she can then launch an attack which has to be fast enough. If the spoofed response reaches later than the genuine response, the latter will get cached in the DNS server and the attacker will have to wait until the cache expires, which may be even days.

4 Kaminsky Attack

- **Describe the exact steps you took to alter to the IP address that was resolved for a given host. Be sure to provide evidence that shows that the host did, indeed, resolve an incorrect IP address. This should come in the form of packet captures, command line results, etc.**

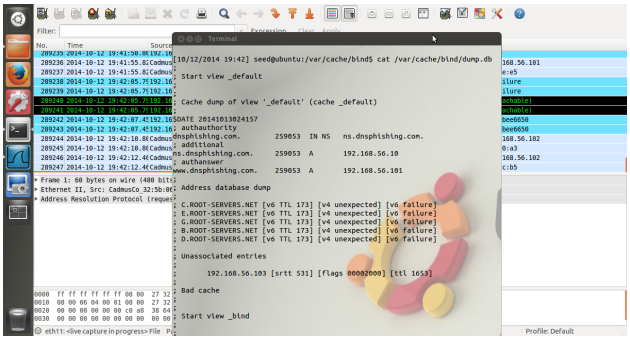

Figure 12: Before Attack - DNS1 Cache After Reply From DNS2

To make this attack successful, a request was sent for a non-existent host which will cause the DNS server to ask the other DNS servers. When the 'fake' DNS response packets were created, it contained the spoofed IP address for the queried domain. If this 'fake' response packet made it way to the DNS server before the arrival of the 'genuine' packet, the DNS server cache would be poisoned with the spoofed IP address. The DNS server will then respond to the user with this spoofed IP address from its cache. It will drop the 'genuine' response that would follow the 'fake' one.

Henceforth, we consider DNS1 to be the DNS Server and DNS2 to be the 'forward resolver'.

- **Provide screen shots, commands executed, and command output to explain what took place.** To start with after the initial setup, a 'dig' was performed to be sure that the DNS2 server responds correctly as per the entries present there. The DNS1 server dump was taken as per Figure 12

  A DNS request query payload was created along with 3 other input files - ip_header_query (IP packet header), eth_header_query (ethernet header), and udp_header_query (UDP packet header). All these 4 input files were fed to a 'C' program *pacgen_res.c*. This program would then create a DNS request packet and send it to the DNS1 server.

  Once this is done, huge amounts of spoofed DNS responses were then targetted towards DNS1 spoofing that they originated from DNS2. This was done using a loop in *pacgen_res.c* that ran 65536 times or till a transaction ID match was found, whichever was earlier. If during this time, a 'genuine' response was not received by DNS1 from DNS2, then this spoofed response would get cached in the DNS1 server cache. After this attack was launched and was successful after a number of failed attempts, the DNS server cache was poisoned as can be seen in Figure 13
  In the Figure 13, we can see that there is an entry for the non-existent hostname that was queried viz. *x-3309570.dnsphishing.com* with an IP address of *1.2.3.4* as was mentioned in the spoofed DNS response.

  The trace files are placed in *Task4/Trace/After Attack*.

- **Describe, in plain English, how the attack tool you wrote for this portion of the assignment is structured and functions.**
  As per the Kaminsky Attack, the DNS server cache is poisoned by an attacker. In the attached tool, there is a single file called pacgen_res.c which takes 4 input files - ip_header_query (IP packet header), eth_header_query (ethernet header), udp_header_query (UDP packet header), and payload_query (DNS request payload).

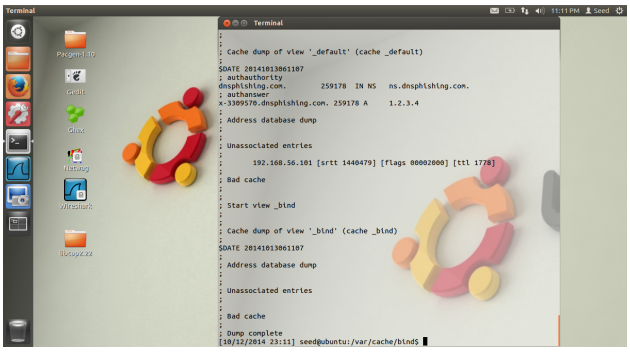

Figure 13: After Attack - DNS1 Cache Poisoned

This program first builds a DNS request from the above 4 files and sends it to the DNS server, say DNS1, (the one that has to be attacked). This request contains a nonce as a subdomain whose entry is not present on DNS1. As a result, the DNS server will forward this query to another server (forward resolver), say DNS2. Now, the program builds various DNS response packets to be sent from DNS2 to DNS1 each having a different transaction ID (generated randomnly). These are all fake responses flowing from DNS2 to DNS1 in the hope that their transaction ID matches with the transaction ID of the "genuine" DNS request sent by DNS1 to DNS2. Once a match is found, DNS1 assumes that this response has indeed come from DNS2 and will cache this "fake" result in its cache. We can they say that DNS1's cache has been posioned.

Now if someone queries for this domain, DNS1 will look into its cache and return this spoofed IP address.

- **How successful was your attack tool? Did you have to run it several times? Why? What could you do to increase its level of success?**
  The attack tool was successful in the sense that the transaction ID, source/destination ports, source/destination IP addresses of the "fake" DNS response matched with those of the "genuine" DNS request. Once this match occurs, the DNS server cache is poisoned and our attack is successful.

  Since the attacker cannot see the "genuine" DNS request transaction, the attack becomes difficult and we need to create transaction IDs randomnly and flood the DNS1 server until a match is found. This process needs to be repeated until a match is found thereby making us run the program several times.

  To increase the level of success, we can use parallelization to generate the random transaction IDs quickly and send them over the network. We can use multiple computers to flood the DNS server to achieve our target in comparatively shorter time.

- **Describe, in detail, the steps you took to prevent any packets associated with this attack from reaching the Internet. Failure to provide a sufficient explanation will result in 0 points awarded for this portion of the report.**
  For preventing any packets reaching the Internet, another DNS server called a 'forward resolver' was set up. This server (DNS2 in our case) contains the zone file for the concerned domain (dnsphishing.com in this case). So when the DNS1 server is queried and it does not find an entry on it or in its cache, it will forward this DNS request to the forward resolver. This is managed by setting this

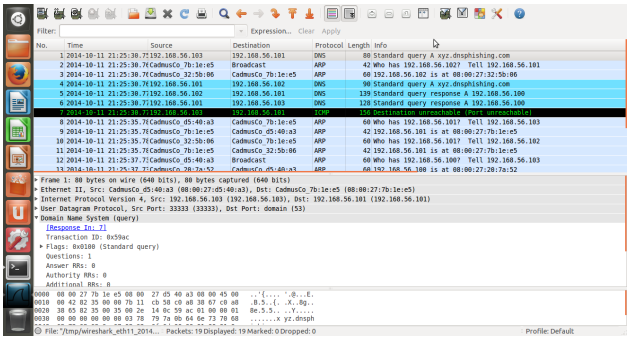

Figure 14: After Attack - DNS Response Through the Forward Resolver

DNS2 server to be a forwarder in the *named.conf* file on DNS1 server. When the DNS request reaches DNS2, it finds the entry on it and returns the results to DNS1.
We can see that the above setup worked as expected from the Figure 14
As can be seen in the above figure, the user/client (IP 192.168.56.103) sent a DNS request to the DNS1 server (IP 192.168.56.101) on Line Number 1.
Line Number 4 - DNS1 server then forwarded this DNS request to DNS2 server (IP 192.168.56.103)
Line Number 5 - DNS2 server replied to DNS1 stating the IP address of *xyz.dnsphishing.com*.
Line Number 6 - DNS1 then returns this response to the user/client.
Even while the attack is performed, we restrict the packets in this fashion. Therefore, as seen above, the DNS request does not fo further than DNS2 server, and definitely not on the Internet.

The DNS1 response trace and DNS2 to DNS1 response trace files (prior to the attack) are kept at *Task4/Trace/Before Attack*.

- **Explain how a real-world attacker could leverage this sort of attack.Discuss whether you feel this is a viable attack in the real world.**
  A real-world attacker can leverage this sort of attack by using spam. In this case, the attacker can provide a link to a non-existent website in some spam mail/link. Once the user clicks on it, a DNS request goes through as described above. This setting can be used to poison the DNS server cache.

  This is a viable attack in the real world as this attack can be used to poison the DNS server cache. Most of the people these days do not look at the links in their mails and this can be taken advantage of to leverage the damage that can be caused due to this attack.