DNS Pharming Attack Lab

1 Lab Overview

DNS (Domain Name System) is the Internet's phone book; it translates hostnames to IP addresses (or IP addresses to hostnames). This translation is through DNS resolution, which happens behind the scene. DNS Pharming attacks manipulate this resolution process in various ways, with an intent to misdirect users to alternative destinations, which are often malicious. The objective of this lab is to understand how such attacks work. Students will first set up and configure a DNS server, and then they will try various DNS Pharming attacks on the target that is also within the lab environment.

2 Lab Environment

We need to setup the lab environment as the figure below. To simplify the lab environment, we let the user's computer, DNS server, and attacker's computer be on one physical machine, but using different virtual machines. The website used in this lab can be any website. Our configuration is based on `Ubuntu`, which is the operating system we use in our pre-built virtual machine.

```
User                        DNS Server                 Attacker
192.168.0.100               192.168.0.10               192.168.0.200
        |                           |                          |
        |_____|_____|
        |                   LAN or Virtual Network             |
        |_____|
                                    |
                               Internet
```
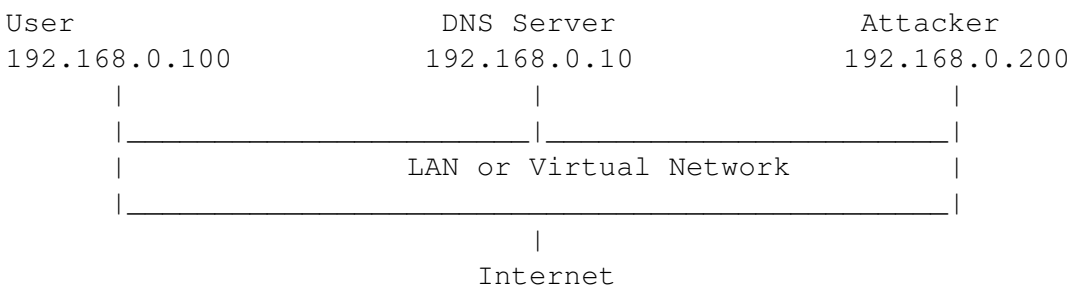
The above is the figure of the lab environment. As you can see, we set up the DNS server, the user machine and the attacker machine in the same LAN. We assume that the user machine's IP address is `192.168.0.100`, the DNS Server's IP is `192.168.0.10` and the attacker machine's IP is `192.168.0.200`.

Note for Instructors: For this lab, a lab session is desirable, especially if students are not familiar with the tools and the environments. If an instructor plans to hold a lab session (by himself/herself or by a TA), it is suggested the following to be covered in the lab session [1]:

---

[1] We assume that the instructor has already covered the concepts of the attacks in the lecture, so we do not include them in the lab session.

1. The use of the virtual machine software.

2. The use of `Wireshark`, `Netwag`, and `Netwox` tools.

3. Configuring the DNS server.

2.1 Install and configure the DNS server

Step 1: Install the DNS server. On `192.168.0.10`, We install the BIND9 DNS server using the following command:

```
# sudo apt-get install bind9
```

The `BIND9 Server` is already installed in our pre-built Ubuntu virtual machine image.

Step 2: Create the `named.conf.options` **file.** The DNS server needs to read the `/etc/bind/named.conf` configuration file to start. This configuration file usually includes an option file called `/etc/bind/named.conf.option`. Please add the following content to the option file:

```
options {
        dump-file          "/var/cache/bind/dump.db";
};
```

It should be noted that the file `/var/cache/bind/dump.db` is used to dump DNS server's cache.

Step 3: Create zones. Assume that we own a domain: `example.com`, which means that we are responsible for providing the definitive answer regarding `example.com`. Thus, we need to create a zone in the DNS server by adding the following contents to `/etc/bind/named.conf`. It should be noted that the `example.com` domain name is reserved for use in documentation, and is not owned by anybody, so it is safe to use it.

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

zone "0.168.192.in-addr.arpa" {
        type master;
        file "/var/cache/bind/192.168.0";
      };
```

Note that we use `192.168.0.x` as an example. If you use different IP addresses, you need to change `/etc/bind/named.conf` and the DNS lookup files (stated below) accordingly.

Step 4: Setup zone files. The file name after the `file` keyword in the above zones is called the zone file. The actual DNS resolution is put in the zone file. In the `/var/cache/bind/` directory, compose the following `example.com.db` zone file (Note that the configuration files stated in the following can be downloaded from the web page of this lab; typing in these files might introduce errors. If you are interested in the syntax of these configuration files, please refer to RFC 1035 for details):

```
$TTL 3D
@       IN      SOA     ns.example.com. admin.example.com. (
        2008111001      ;serial, today's date + today's serial number
        8H              ;refresh, seconds
        2H              ;retry, seconds
        4W              ;expire, seconds
        1D)             ;minimum, seconds


@       IN      NS      ns.example.com. ;Address of name server
@       IN      MX      10 mail.example.com. ;Primary Mail Exchanger

www     IN      A       192.168.0.101 ;Address of www.example.com
mail    IN      A       192.168.0.102 ;Address of mail.example.com
ns      IN      A       192.168.0.10 ;Address of ns.example.com
*.example.com. IN A     192.168.0.100 ;Address for other URL in
                                      ;example.com. domain
```

The symbol '@' is a special notation meaning the origin from the named.conf. Therefore, '@' here stands for example.com. 'IN' means internet. 'SOA' is short for Start Of Authority. This zone file contains 7 resource records (RRs): a SOA (Start Of Authority) RR, a NS (Name Server) RR, a MX (Mail eXchanger) RR, and 4 A (host Address) RRs.

We also need to setup the DNS reverse lookup file. In the directory /var/cache/bind/, compose a reverse DNS lookup file called 192.168.0 for example.com domain:

```
$TTL 3D
@       IN      SOA     ns.example.com. admin.example.com. (
                2008111001
                8H
                2H
                4W
                1D)
@       IN      NS      ns.example.com.

101     IN      PTR     www.example.com.
102     IN      PTR     mail.example.com.
10      IN      PTR     ns.example.com.
```

Step 5: Start a DNS server.   Now we are ready to start the DNS server. Run the following command:

```
   % sudo /etc/init.d/bind9 restart
or
   % sudo service bind9 restart
```

2.2   Configure the User Machine

On the user machine 192.168.0.100, we need to let the machine 192.168.0.10 be the default DNS server. We achieve this by changing the DNS setting file /etc/resolv.conf of the user machine:

```
  nameserver 192.168.0.10 # the ip of the DNS server you just setup
```

Note: make sure this is the only nameserver entry in your /etc/resolv.conf. Also note that, in Ubuntu, /etc/resolv.conf may be overwritten by the DHCP client. To avoid this, disable DHCP by doing the following (in Ubuntu 9):

```
Click "System" -> "Preferences" -> "Network Connections",
Double-click the network device (e.g. eth1) in "Wired" Tab,
Select "IPv4 Settings" -> "Method" ->"Automatic(DHCP) Addresses Only"
and update only "DNS Servers" entry with IP address of BIND DNS Server.

Now Click the "Network Icon" on the top right corner and Select
"Auto eth0". This will refresh the wired network connection and
updates the changes.
```

You should restart your Ubuntu machine for the modified setting to take effect.

2.3   Configure the Attacker Machine

On the attacker machine, there is not much to configure. The attacker needs to run Netwag or Netwox as the root user.

2.4   Expected Output

After you have set up the lab environment according to the above steps, your DNS server is ready to go. Now, on the user machine, issue the following command:

```
% dig www.example.com
```

You should be able to see something like this:

```
<<>> DiG 9.5.0b2 <<>> www.example.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27136
;; 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 192.168.0.101

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

;; ADDITIONAL SECTION:
ns.example.com. 259200 IN A 192.168.0.10

;; Query time: 80 msec
;; SERVER: 192.168.0.10#53(192.168.0.10)
```

```
;; WHEN: Tue Nov 11 15:26:32 2008
;; MSG SIZE  rcvd: 82
```

Note: the `ANSWER SECTION` contains the DNS mapping. You can notice that the IP address of `www.example.com` is now `192.169.0.101`, which is what we have set up in the DNS server. For a simple and clear answer, we can use `nslookup` instead. To do a DNS reverse lookup, issue `dig -x N.N.N.N`.

2.5   Install Wireshark

`Wireshark` is a very important tool for this lab; you can sniff every package that is going through the LAN. You can get `Wireshark` from `http://www.wireshark.org`. Although `Netwox` also comes with a sniffer, `Wireshark` is a much better sniffer. `Wireshark` is already installed in our pre-built virtual machine.

3   Lab Tasks: Pharming Attacks

The main objective of Pharming attacks on a user is to redirect the user to another machine B when the user tries to get to machine A using A's host name. For example, when the user tries to access the online banking, such as `www.chase.com`, if the adversaries can redirect the user to a malicious web site that looks very much like the main web site of `www.chase.com`, the user might be fooled and give away password of his/her online banking account.

When a user types in `www.chase.com` in his browsers, the user's machine will issue a DNS query to find out the IP address of this web site. Attackers' goal is to fool the user's machine with a faked DNS reply, which resolves `www.chase.com` to a malicious IP address. There are several ways to achieve such an attack. In the rest of the lab description, we will use `www.example.com` as the web site that the user wants to access, instead of using the real web site name `www.chase.com`; the `example.com` domain name is reserved for use in documentation, and is not owned by anybody.

3.1   Attackers have already compromised the victim's machine

Modifying HOSTS file.   The host name and IP address pairs in the HOSTS file (`/etc/hosts`) are used for local lookup; they take the preference over remote DNS lookups. For example, if there is a following entry in the HOSTS file in the user's computer, the `www.example.com` will be resolved as `1.2.3.4` in user's computer without asking any DNS server:

```
1.2.3.4        www.example.com
```

Attacks.   If attackers have compromised a user's machine, they can modify the HOSTS file to redirect the user to a malicious site whenever the user tries to access `www.example.com`. Assume that you have already compromised a machine, please try this technique to redirect `www.example.com` to any IP address that you choose.

Note: `/etc/hosts` is ignored by the `nslookup` command, but will take effect on ping command and web browser etc.
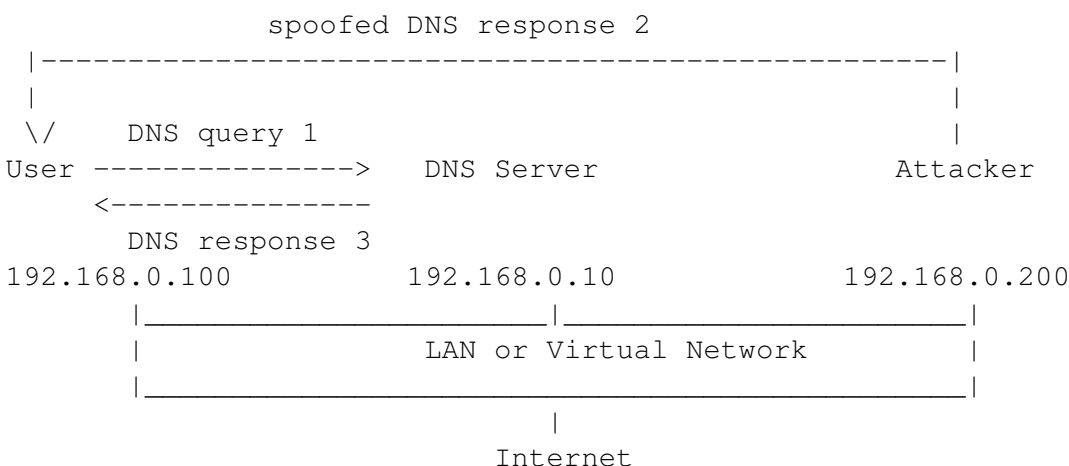
3.2 Directly Spoof Response to User

In this attack, the victim's machine has not been compromised, so attackers cannot directly change the DNS query process on the victim's machine. However, if attackers are on the same local area network as the victim, they can still achieve a great damage.

When a user types the name of a web site (a host name, such as `www.example.com`) in a web browser, the user's computer will issue a DNS request to the DNS server to resolve the IP address of the host name. After hearing this DNS request, the attackers can spoof a fake DNS response. The fake DNS response will be accepted by the user's computer if it meets the following criteria:

1. The source IP address must match the IP address of the DNS server.

2. The destination IP address must match the IP address of the user's machine.

3. The source port number (UDP port) must match the port number that the DNS request was sent to (usually port 53).

4. The destination port number must match the port number that the DNS request was sent from.

5. The UDP checksum must be correctly calculated.

6. The transaction ID must match the transaction ID in the DNS request.

7. The domain name in the question section of the reply must match the domain name in the question section of the request.

8. The domain name in the answer section must match the domain name in the question section of the DNS request.

9. The User's computer must receive the attacker's DNS reply before it receives the legitimate DNS response.

To satisfy the criteria 1 to 8, the attackers can sniff the DNS request message sent by the victim; they can then create a fake DNS response, and send back to the victim, before the real DNS server does. `Netwox` tool 105 provide a utility to conduct such sniffing and responding.

Tip: in the `Netwox/Netwag` tool 105, you can use the "filter" field to indicate the IP address of your target. For example, in the scenario showing below, you can use `"src host 192.168.0.100"`.

```
                    spoofed DNS response 2
  |------------------------------------------------|
  |                                                |
  \/     DNS query 1                               |
 User --------------->   DNS Server            Attacker
      <---------------
        DNS response 3
 192.168.0.100          192.168.0.10          192.168.0.200
        |_____|_____|
        |              LAN or Virtual Network              |
        |_____|
                                 |
                            Internet
```
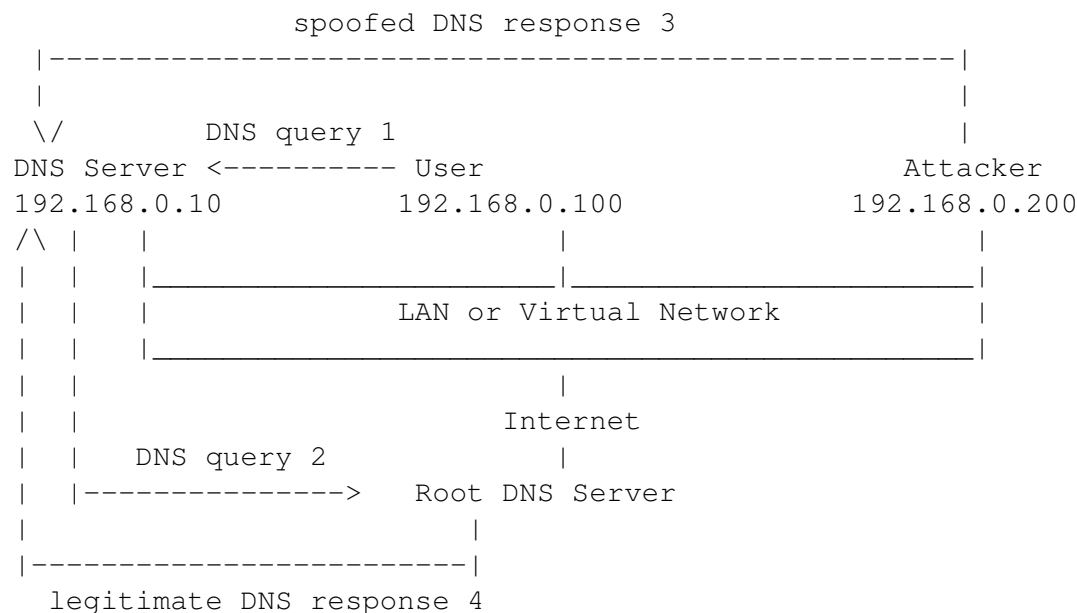
3.3   DNS Server Cache Poisoning

The above attack targets the user's machine. In order to achieve long-lasting effect, every time the user's machine sends out a DNS query for www.example.com, the attacker's machine must send out a spoofed DNS response. This might not be so efficient; there is a much better way to conduct attacks by targeting the DNS server, instead of the user's machine.

When a DNS server Z receives a query, if the host name is not within the Z's domain, it will ask other DNS servers to get the host name resolved. Note that in our lab setup, the domain of our DNS server is example.com; therefore, for the DNS queries of other domains (e.g. www.google.com), the DNS server Z will ask other DNS servers. However, before Z asks other DNS servers, it first looks for the answer from its own cache; if the answer is there, the DNS server Z will simply reply with the information from its cache. If the answer is not in the cache, the DNS server will try to get the answer from other DNS servers. When Z gets the answer, it will store the answer in the cache, so next time, there is no need to ask other DNS servers.

Therefore, if attackers can spoof the response from other DNS servers, Z will keep the spoofed response in its cache for certain period of time. Next time, when a user's machine wants to resolve the same host name, Z will use the spoofed response in the cache to reply. This way, attackers only need to spoof once, and the impact will last until the cached information expires. This attack is called *DNS cache poisoning*. The following diagram illustrates this attack.

```
                    spoofed DNS response 3
   |-------------------------------------------------------|
   |                                                       |
  \/          DNS query 1                                  |
DNS Server <---------- User                        Attacker
192.168.0.10            192.168.0.100              192.168.0.200
/\ |   |                         |                         |
|  |   |_____|_____|
|  |   |              LAN or Virtual Network               |
|  |   |_____|
|  |   |                         |
|  |                             |
|  |                          Internet
|  |    DNS query 2              |
|  |-------------->  Root DNS Server
|  |                             |
|------------------------|
    legitimate DNS response 4
```

We can use the same tool (Netwox 105) for this attack. Before attacking, make sure that the DNS Server's cache is empty. You can flush the cache using the following command:

```
# sudo rndc flush
```

The difference between this attack and the previous attack is that we are spoofing the response to DNS server now, so we set the filter field to 'src host 192.168.0.10', which is the IP address of the DNS server. We also use the ttl field (time-to-live) to indicate how long we want the fake answer to stay in the DNS server's cache. After the DNS server is poisoned, we can stop the Netwox 105. If we set ttl to 600 (seconds), then DNS server will keep giving out the fake answer for the next 10 minutes.

Note: Please select the `raw` in the `spoofip` field; otherwise, `Netwox 105` will try to also spoof the MAC address for the spoofed IP address. To get the MAC address, the tool sends out an ARP request, asking for the MAC address of the spoofed IP. This spoofed IP address is usually a root DNS server (this is usually the first place that a DNS server will ask if it cannot resolve a name), and obviously the root DNS server is not on the same LAN. Therefore, nobody will reply the ARP request. The tool will wait for the ARP reply for a while before going ahead without the MAC address.

The waiting will delay the tool from sending out the spoofed response. If the actual DNS response comes earlier than the spoofed response, the attack will fail. That's why you need to ask the tool not to spoof the MAC address.

You can tell whether the DNS server is poisoned or not by using the network traffic captured by `wireshark` or by dumping the DNS server's cache. To dump and view the DNS server's cache, issue the following command:

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

3.4   An Advanced DNS Cache Poisoning (Remote Attack)

Note: this task needs a substantial amount of time. Students need to modify an existing program (`pacgen.c`) to forge DNS response packets (UDP packets). However, the program only has less than 400 lines of code, and is not difficult to understand. Students only need to modify a small portion of the code to construct DNS packets. Students also need to spend time to understand the format of DNS response packets.

The previous attack assumes that the attacker and the DNS server are on the same LAN, i.e., the attacker can observe the DNS query message. When the attacker and the DNS server are not on the same LAN, the cache poisoning attack becomes more difficult. The difficulty is mainly caused by the fact that the transaction ID in the DNS response packet must match with that in the query packet. Because the transaction ID in the query is usually randomly generated, without seeing the query packet, it is not easy for the attacker to known the correct ID.

Obviously, the attacker can guess the transaction ID. Since the size of the ID is only 16 bits, if the attacker can forge K responses within the attack window (i.e. before the legitimate response arrives), the probability of success is K over 2^{16}. Sending out hundreds of forged responses is not impractical, so it will not take too many tries before the attacker can succeed.

However, the above hypothetical attack has overlooked the cache effect. In reality, if the attacker is not fortunately enough to make a correct guess before the real response packet arrives, correct information will be cached by the DNS server for a while. This caching effect makes it impossible for the attacker to forge another response regarding the same domain name, because the DNS server will not send out another DNS query for this domain name before the cache times out. To forge another response on the same domain name, the attacker has to wait for another DNS query on this domain name, which means he/she has to wait for the cache to time out. The waiting period can be hours or days.

To launch effective attacks, the attacker must negate the caching effect. Dan Kaminsky came up with an elegant method to do this. Using his method, attackers will be able to continuously attack a DNS server on a domain name, without the need for waiting. With this method, an attacker can succeed within a very short period of time. Details of the attacks are described in the following URL (see the footnote [2]). In this task, we will try this attack method.

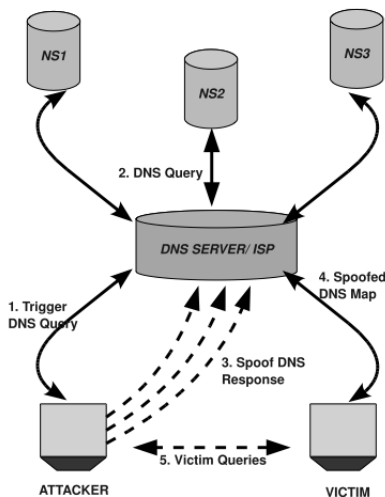[2]`http://spectrum.ieee.org/computing/software/fresh-phish`.

Figure 1: DNS - Phishing Attack Details

We choose a fictitious company `dnsphishinglab.com` as our targeted domain name. Students can choose any domain name they like, but please make sure the attack is targeted at the DNS server in this lab environment, not at a real DNS server. The attacker wants to cause the DNS server to use his malicious IP address when somebody queries the DNS server for `www.dnsphishinglab.com`. Let us call the targeted DNS server `apollo`. The following steps with reference to Figure 1 describe the outline of the attack.

1. The attacker queries the DNS Server `apollo` for a non-existing name in `dnsphishinglab.com`, for example `xyz9392.dnsphishinglab.com`, where `xyz9392` is a random name.

2. Since the mapping is unavailable in `apollo`'s DNS cache, `apollo` queries other DNS servers for the name resolution.

3. The attacker floods `apollo` with many spoofed DNS response, each trying a different transaction ID (hoping one is correct). In the response, not only does the attacker provides an IP resolution for `xyz9392.dnsphishinglab.com`, the attacker also provides an IP address for an additional name `www.dnsphishinglab.com`. This is allowed in the DNS protocol, as long as this additional name and the queried name are from the same domain. If the spoofed response beats the actual responses and the other elements in the response, such as the transaction ID, matches with the original query, `apollo` will accept the response, the spoofed answer will then be cached, and thus `apollo`'s DNS cache will be poisoned.

4. Even if the spoofed DNS response failed (i.e. did not match with the query or was beaten by the actual response), since `xyz9392` will most likely not exist in the `dnsphishinglab.com` domain, the DNS server that is responsible for `dnsphishinglab.com` will simply reply with a negative response, indicating that the name does not exist. The negative response will not affect `apollo`'s DNS cache. Now you should be able to see why we used a random host name at the beginning.

5. If the attack fails, the attacker will go back to step one, and try another non-existing name, until the attack succeeds.

6. Once the attack succeeds, when the victim sends a DNS query to the poisoned DNS server for `www.dnsphishinglab.com`, the IP address returned by the DNS server will not be the actual IP address for `www.dnsphishinglab.com`; instead, it will be the IP address set by the attacker.

7. Therefore, when the victim browses `www.dnsphishinglab.com`, he/she will be actually browsing a malicious web site that looks like the real `www.dnsphishinglab.com`. If victim types any credential information in this malicious web site, those credentials will be stolen by the attacker.

This attack is very similar to the previous DNS server cache poisoning task, but the important distinguishing factor is that this attack is made possible even if the targeted DNS server and the attacker are on different networks. For the sake of simplicity, this attack is performed on two different machines in the same network, but we do not assume that the attacker can observe the DNS request packets.

Attack Configuration. We make the following configuration for this task:

1. *Configuration the Attack Machine:* We will use the user machine as our attack machine. The user machine is already configured in Section 2.2. It uses the targeted DNS server as its default DNS server.

2. *Source Ports:* Some DNS servers now randomize the source port number in the DNS queries; this makes the attacks much more difficult. Unfortunately, many DNS servers still use predictable source port number. For the sake of simplicity in this task, we assume that the source port number is a fixed number. We can set the source port for all DNS Queries from DNS servers to be `33333`. This can be done by adding the following option to the file `/etc/bind/named.conf.options`:

   ```
   query-source port 33333
   ```

3. *DNS Forwarder:* We need to ensure that when you issue queries for non-existent hosts that these requests are not forwarded to the Internet. To accomplish this, you need to configure a forward zone on your existing Bind instance. Add the following configuration to your named.conf:

   ```
   zone "dnsphishinglab.com" {
       type forward;
       forwarders { 192.168.2.100; }; //This is the IP address
                                       //of a second Bind server
   };                                  //you will configure in the next step
   ```

4. *Forward Resolver:* For this experiment, you will need to stand up a second Bind server to which your existing server to forward its requests. This is done just as described in Section 2.1. However on this server, you need to configure the dnsphisinglab.com zone.

   You will need to inject artificial latency into the responses from the forward resolver. To do this, execute the following command on the forward resolver VM image:

   ```
   % sudo tc qdisc add dev eth6 root delay 10ms
   ```

   Note: check the man page for tc to learn how to disable this latency.

5. *DNS Bind Server:* Flush the DNS Bind Server's Cache/ Restart DNS Bind Server with the earlier mentioned commands.

Attack Tool and Method. To be able to send a large number of forged DNS response packets within a short time window, we need to use some automation tools. The `pacgen` tool can be used for this purpose. More details about the tool are described later. We outline the use of this tool here:

1. Modify the `pacgen.c` file by appropriately filling the DNS response fields. To understand the format of DNS response packet, you can use `Wireshark` to capture a few DNS response packets and study them.

2. Your version of `pacgen` should generate DNS queries destined for you Bind server (the first one you configured) that will result in a forward lookup (to the second server you configured). These queries should follow the pattern described in steps 1-5 above.

3. After `pacgen` generates the query for a non-existent host, the program should be able to flood the targeted DNS server with many forged DNS response packets, each trying a different transaction ID.

   As described above, this flood of responses should also include an answer for the host you are trying to cause to be incorrectly cached. In the event that the transaction ID matches a valid request to the forward name server, that host will now have an invalid entry in the name servers cache.

   Subsequent name lookups to your Bind server for the targetted host should now return the incorrect IP address.

More about the `pacgen` **Tool.** `Pacgen` is an open source Ethernet IP TCP/UDP packet generating tool. This tool is developed in C language, is easily configurable at every layer of the network packet, and is much efficient packet generator than the netwag tool. We suggest the students to use this tool for packet generation, since this attack requires the attacker to send out thousands of replies in fraction of a second. Pacgen-1.10 tool can be downloaded from the web [3]. Students can also use other tools that they are familiar with.

1. *Configuration:* This tool requires the `Libnet` library to be installed, which has already been installed in our pre-built virtual machine image. This tool can be downloaded from the the web [4].

   The configuration of the header information is very simple for the `pacgen` tool. There is a header configuration file for each layer in the `pacgen` directory, like `eth_header` for ethernet layer, `ip_header` for IP layer.

2. *Compilation:* Students can read the `INSTALL` file in the `pacgen` tool to check for the compilation information. In short, the following command should do for compiling the code to generate `pacgen` executable.

   ```
   % sh install.sh
   ```

   Note 1: If the compilation gives the following error:

   ```
   pacgen.c: undefined reference to 'libnet_open_link_interface',
   ```

---

[3] `http://linux.softpedia.com/get/System/Networking/pacgen-14284.shtml`
[4] `http://www.hacktoolrepository.com/tool/71/`

goto the menu `System->Administration->Synaptic Package Manager`, and search for `libnet1-dev`; select `libnet1-dev` and click the `"Mark for Removal"` item from the dropdown menu. Make sure that `libnet1` package is enabled.

Note 2: Sometimes the compilation might give the following error: `"link layer error on eth0 SIOCGIFHWADDR No Such Device"`. In this case, just update the `pacgen.c` file's `device` variable to your system's `eth#` number, which can be found using the `ifconfig` command.

3. *Execution:* Students can read the `README` file in pacgen tool to check for the execution information. In short, the following command should do for running `pacgen` (you need to have the root privilege to run the program; otherwise, you will get an error message).

```
% sudo sh run_default.sh
```

Attack Tips:

1. *DNS Response Data:* The DNS response should be updated to the `payload_location` variable.

2. *Source Port:* The destination port of the DNS response should match with the source port in the DNS query, which is fixed on 33333 in this lab.

3. *Source IP:* The source IP of the DNS reply should match with the destination IP of the DNS query. Normally the targeted DNS Server contacts several other DNS servers for the name resolution. The behavior is quite predictable. Students can use `Wireshark` to understand the behavior of the targeted DNS server before launching the attack.

4. *Transaction ID:* The transaction ID of the DNS reply should match with the DNS query for the attack to be successful. The transaction ID varies randomly between `0-65535`.

4 Submission

Students need to submit a detailed lab report to describe what they have done and what they have observed. Report should include the evidences to support the observations. Evidences include packet traces, screen-dumps, etc.

References

[1] RFC 1035 Domain Names - Implementation and Specification : http://tools.ietf.org/html/rfc1035

[2] DNS HOWTO : http://www.tldp.org/HOWTO/DNS-HOWTO.html

[3] BIND 9 Administrator ReferenceManual : http://www.bind9.net/manual/bind/9.3.2/Bv9ARM.ch01.html

[4] Pharming Guide : http://www.ngssoftware.com/papers/ThePharmingGuide.pdf

[5] DNS Cache Poisoning: http://www.secureworks.com/research/articles/dns-cache-poisoning/

[6] DNS Client Spoof: http://evan.stasis.org/odds/dns-client_spoofing.txt

[7] Phishing: http://en.wikipedia.org/wiki/Phishing