DNS Cache Poisoning Attack Lab

Contents

Lab Objectives and Setting 1
Overview 1
Objectives 1
Prerequisites 1
EZSetup 1
Environment Setting 2
License 2
DNS Cache Poisoning Attack 3
Assignment 6

Lab Objectives and Setting

Overview

The Internet protocol suite, also known as TCP/IP suite, is the foundation of the modern Internet. This suite provides various protocols that lie upon the Internet Protocol (IP) for end-to-end data communications. Vulnerabilities in TCP/IP protocols may have serious effects on the upper layer protocols and applications and can endanger communication and data security. In this lab, we will look into DNS protocol, learn how it works and analyze its vulnerabilities, and show a way to attack it.

Objectives

Upon completion of this lab, students will be able to:

- Recognize the operation procedure and data format of DNS protocol;
- Explain how DNS cache poisoning attack works;
- Analyze the vulnerability of DNS and give feasible protection solution.

Prerequisites

- Practical experience with SSH and basic Linux commands;
- Basic knowledge of network protocols.

EZSetup

EZSetup is a Web application capable of creating various user-defined cybersecurity practice environments (e.g., labs and competition scenarios) in one or more computing clouds (e.g., OpenStack or Amazon AWS). EZSetup provides a Web user interface for practice designers to visually create a practice scenario and easily deploy it in a computing cloud, which allows for customization and reduces overhead in creating and using practice environments. End users are shielded from the complexity of creating and maintaining practice environments and therefore can concentrate on cybersecurity practices. More information about EZSetup can be found at https://promise.nexus-lab.org/platform/.

Environment Setting

In this lab, students can access three virtual machines (VM) from EZSetup, attacker, victim, and observer VM. The network topology is shown in Figure 1, the VM properties are listed in Table 1. Please refer to the EZSetup dashboard for the actual public IP addresses and passwords.

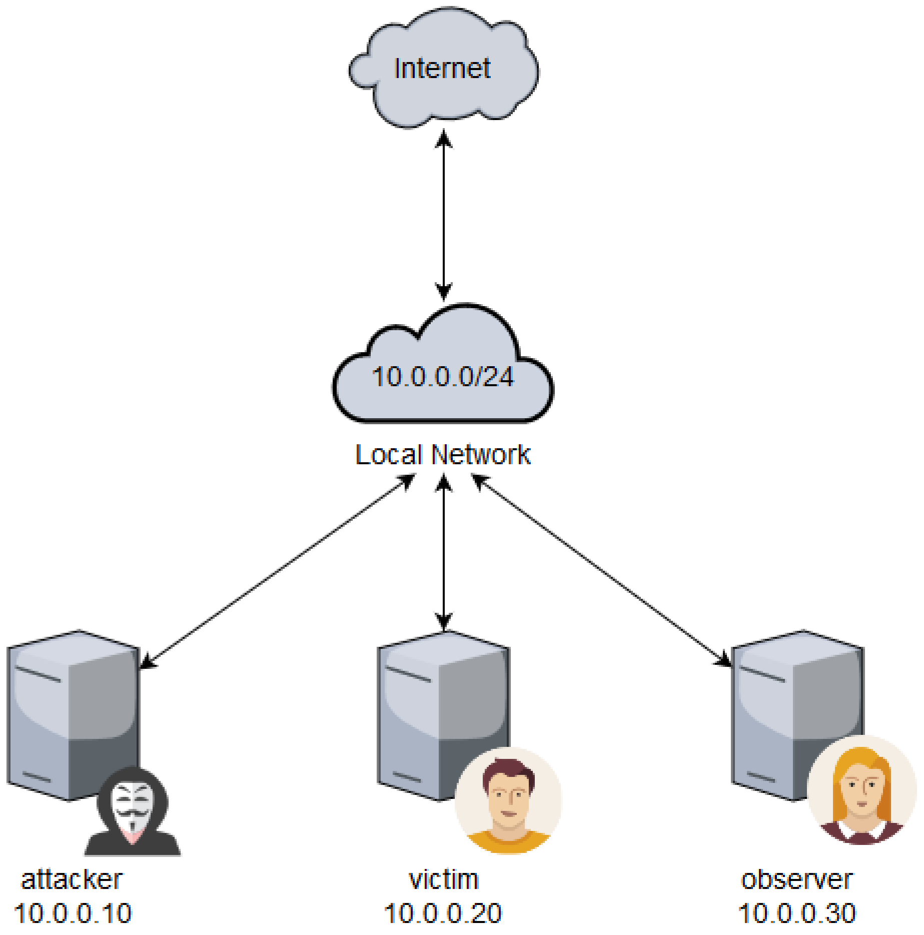

Figure 1 Lab network topology

Table 1 VM properties and access information

Name	**Image**	**RAM**	**VCPU**	**Disk**	**Login account**
ts-attacker	tcpipsecurity-attacker	2GB	2	40GB	See EZSetup
ts-victim	tcpipsecurity-victim	4GB	2	60GB	See EZSetup
ts-observer	tcpipsecurity-observer	2GB	2	40GB	See EZSetup

License

DNS Cache Poisoning Attack

Most of the web services and resources today use the domain names as location identifications. Domain Name System (DNS) is a decentralized system responsible for easy-to-remember domain names to IP addresses translations (DNS records). To maintain countless records, DNS uses a distributed hierarchical database system. Top of the hierarchy is root name servers, which directly answer root domain queries or return authoritative name servers for other queries. Below that are the authoritative name servers, which are usually owned by companies and organizations and give answers to domain names in a certain zone managed by them. For instance, top-level domain (TLD) name servers are responsible for resolving top-level domains like .com and .us. Finally, local domain name servers are often deployed at the edge of the network to proxy and cache DNS query results for performance purpose.

User's DNS queries are often sent to its default local DNS server. A local DNS server will first look at its cache to find if there already exists a valid answer, and if there is, it will directly reply to the request sender. Otherwise, it will ask root and authoritative DNS servers for answer. Let's assume a user asks its local DNS server for the IP address of a domain, e.g., netid.ualr.edu. If a valid cache does not exist, local DNS server will first ask the root server to see it has the domain's DNS record. The root domain will reply the IP address if it has the DNS record, or it will return the address of the TLD name server that manages the top-level domain of the requested domain (.edu). The above process is the same between local DNS server and TLD name server, except the TLD server will return the Authoritative DNS server for that domain. Finally, local DNS can reply user with the IP address of netid.ualr.edu or Non-Existent Domain (NXDomain) if no record for that domain is found. Figure 2 depicts the above stated DNS iterative resolving process.

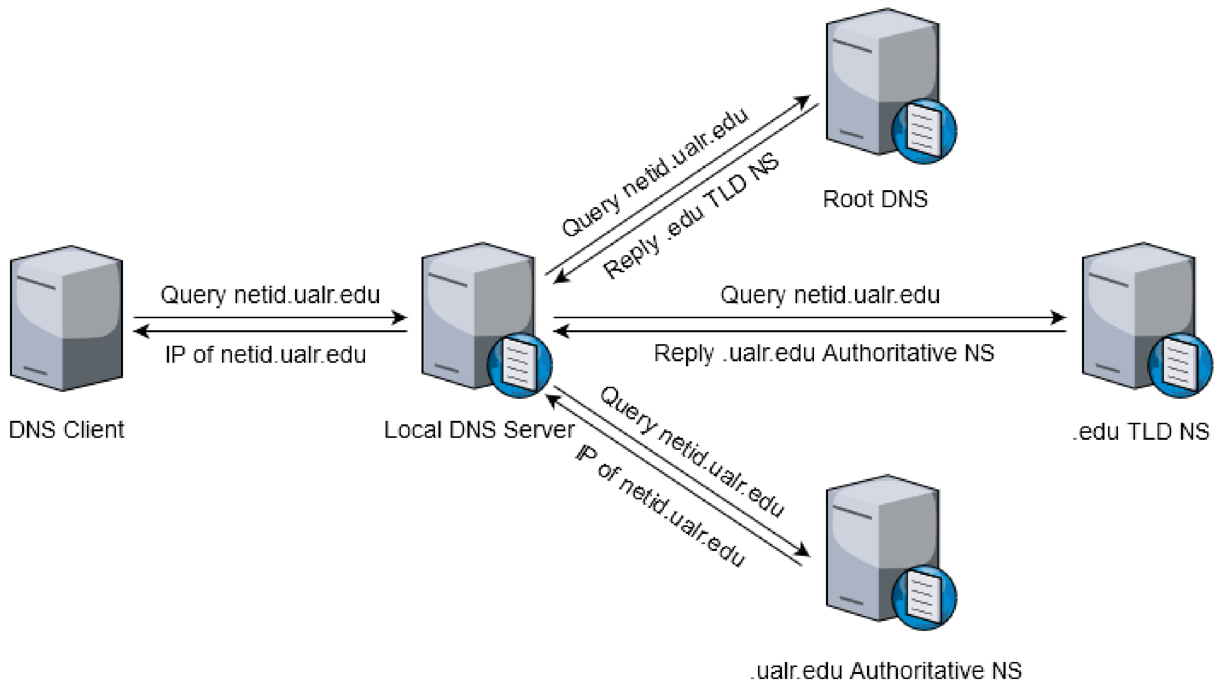

Figure 2 DNS iterative queries resolving process

DNS protocol defines many DNS record types, and the most used ones are A, CNAME, NS, and MX. Type A records are address records, which are used for mapping domain names to IP addresses. CNAME record, or canonical name record, is an alias of another domain like www.ualr.edu is an alias of ualr.edu.

NS records are name server record, which maps DNS zones to their authoritative name servers. Finally, MX is mail exchange record, which gives the message transfer agents for a domain.

The message format of DNS is shown in Figure 3. DNS messages are based on UDP, hence, there is no connection between the DNS server and the client. To determine which DNS packet is the response to a given query, DNS uses (source IP address, source UDP port, destination IP address, destination UDP port, identification) tuple. The **identification** (ID) here is a 16-bit unsigned integer, which is randomly generated by the DNS request sender, and the response has the same ID as the request. The **flags** field indicates the properties of the DNS packet and the DNS server, e.g., the packet is a query or answer, and if the DNS server supports recursion query. The following fields indicate the number and content of the DNS questions and answers. The **questions** field contains one or more DNS questions; each question contains a domain name and the DNS record type that is being asked. The next three sections (**answers**, **authority**, and **additional information**) share a common format called resource record (RR), which contains the domain name, record type, time-to-live and resource value. The answers section has the DNS records to the questions. Authority section gives the authoritative server information about the queried domain. Additional information section contains other DNS records that are relevant to the questions and may be used sometimes later.

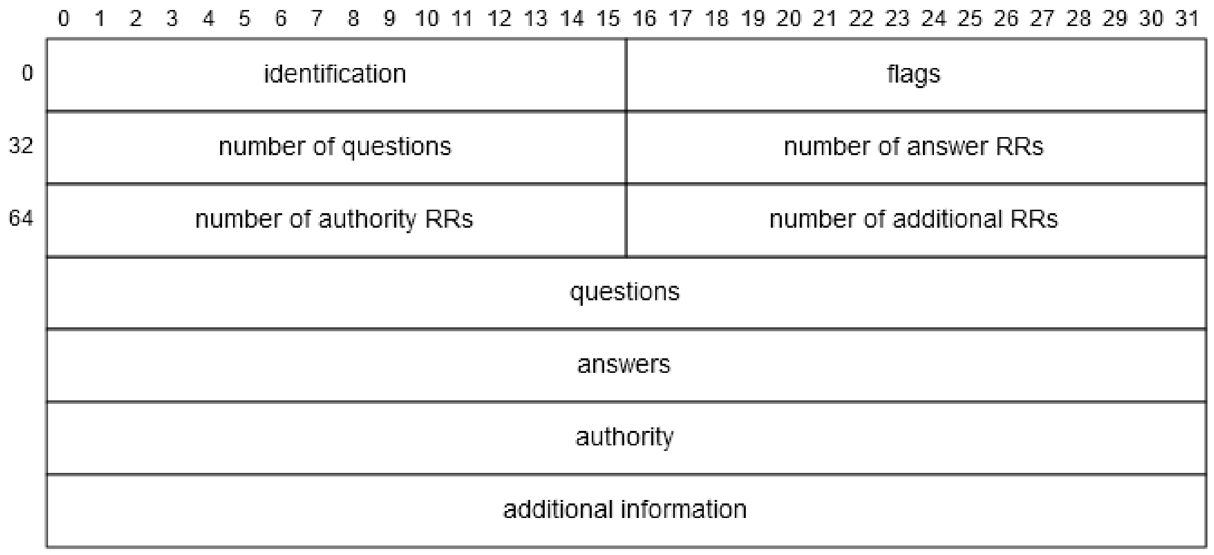

Figure 3 DNS message format

DNS is not encrypted by default, which makes DNS vulnerable to many attacks. Moreover, DNS attacks usually affect not only the server itself but clients that use it. Thus, a successful DNS attack can be very efficient and influential. DNS cache poisoning is a form of attack in which spoofed DNS records are injected into DNS server's cache, making it returns incorrect IP addresses for certain domains. The attacker can force compromised DNS server users to be directed to the wrong hosts chosen by the attacker, and trick them into downloading malicious contents or leaking sensitive information.

To perform a DNS cache poisoning attack, the attacker needs to get the corrupted DNS records cached in the DNS server, by exploiting flaws either in DNS software or protocol. If the attacker is on the same network with the DNS server and he/she can sniff the packets sent by it, this attack would be much simpler to perform. For instance, the attacker wants to insert a DNS record that says www.google.com is at 192.168.0.10. The attacker can first send a DNS request to the server asking about that domain. If the server has no record of that domain in the cache or the cached record expires, it will send out a query to the root or authoritative DNS server. The attacker listens for the server query in the network, learning its source and destination IP and port number, as well as the identification number. Then, the attacker generates a spoofed DNS response with the corresponding information except for a faked answer. If this response beats the one returned from the genuine server, the wrong DNS record will be cached on the server, and hence the attack succeeds.

This attack method has some obvious disadvantages, and will be very limited in real life because of the following facts:

1. If the attacker is not in the same network with the target DNS server, or the underlying network does not allow packet sniffing, the attack would have no idea of the source port, destination IP address and identification number of server's DNS query. In addition, DNS server software now use random query UDP port and identification number. It is very difficult to create a legitimate reply by purely guessing these numbers.

2. Even if the attacker can guess all the information he/she need to create a spoofed reply, he/she still needs to make sure the spoofed packet arrives before the real one does. If not, the correct response will be cached, usually for hours or days. This makes another poisoning attack of the same domain impossible for a while because the server will not initiate a query before cache expires.

3. Some DNS software enforces a spoof-defense mechanism called Domain Name System Security Extensions (DNSSEC). DNSSEC requires DNS resource records to be signed by asymmetrical cryptography. Without the private key, it is impossible for the attacker to forage an authenticated DNS response.

For the sake of experiment simplicity, we use Dnsmasq as the DNS server. Dnsmasq is a networking service toolkit which provides DNS, DHCP, and network boot functions. When acting as a local DNS server, Dnsmasq forwards DNS queries to the upstream DNS server and caches responses. To make the attack easier, we made the following changes to Dnsmasq and its configurations:

1. Reduce the value space of DNS identification number from 2^16 to 2^8;

2. Use a fixed DNS query port 8053. In other words, DNS server sends DNS requests through a single port 8053;

3. There is only one upstream DNS server, which is at 8.8.8.8;

4. DNSSEC features are disabled.

In order to mitigate cache's impact on the attack, we do not try to poison the target domain directly. During the attack, the attacker floods the victim DNS server with queries and replies of nonexistent domains like 4c6ef977.example.com, hoping the spoofed replies beat the real ones. Because the server does not have cached answers for these domains, it will send queries the upstream DNS server with random identification, fixed source/destination IP addresses and ports. Also, as the identification number space is decreased to [0, 2^8-1], it is feasible for the attacker to guess the right identification number in a short time. Furthermore, in every DNS reply sent by the attacker, we set the made-up domain as a CNAME to the target domain along with the A record for the target domain, e.g.

CNAME 4c6ef977.example.com → ualr.edu
A ualr.edu → 192.168.0.10

Once the bogus DNS reply is accepted by the server, the A record of target domain will also be cached. Thus, the attack is completed.

To bring up the Dnsmasq service, please using the following command:

```
$ sudo dnsmasq -C /etc/dnsmasq.conf -Q <query_port_number>
```

We also provide a tool for performing DNS cache poisoning attack, and you can use it by executing the following command:

```
$ sudo dnspoisoning -u <upstream_dns> -d <dmoain> -s <domain_ip> <target_dns_ip>:<target_dns_port>
```

For example, the victim DNS server is at 192.168.0.10, and it uses UDP port 9000 as query port and 8.8.8.8 as upstream DNS server. You can insert a spoofed record to point www.google.com to 192.168.0.20 by

```
$ sudo dnspoisoning -u 8.8.8.8 -d www.google.com -s 192.168.0.20 192.168.0.10:9000
```

Also, Dnsmasq caches records in the memory. To clear cached DNS records, you may use the following command to stop Dnsmasq first:

```
$ sudo killall dnsmasq
```

And please check cached DNS record by this command:

```
$ sudo pkill -USR1 dnsmasq && cat /var/log/syslog | grep dnsmasq
```

Assignment

1. Use “dig” tool to make a DNS query about domain “ualr.edu”

   ```
   $ dig ualr.edu
   ```

   and use Wireshark to capture the DNS request and response with the following filter options:

   ```
   dns.qry.name == ualr.edu
   ```

   What is the flags field of the DNS request? What is the query type (A, CNAME, NX or MX)? What is the answer?

2. Start Dnsmasq on the victim VM and make the observer use it as its default DNS server by editing /etc/resolv.conf

   ```
   $ sudo nano /etc/resolv.conf
   ```

   Then, use the DNS poisoning tool to insert a spoofed record for pointing ualr.edu to the attacker’s machine. Finally, enter the fake_site directory and start an HTTP server on the attacker VM using

   ```
   $ cd ~/fake_site && sudo python -m SimpleHTTPServer 80
   ```

Open a browser on the observer VM and visit http://ualr.edu. Also, use Wireshark to capture DNS packets on the victim during the attack. What command(s) do you use for this attack? Also, describe your observations.

Complete all the tasks and save your answer (with screenshots) to each of the tasks into a PDF file. Submit the PDF file.