Configure Apache Web Server

Task

1h30m - 2h

Status Incomplete

Introduction

In this activity, you will go through the steps to configure an Apache Web Server.

Investigating Server Capability

Your manager is pleased with all the work you've done so far. Now, as they prepare to launch their new website, PawsitivelyClean.com, they're asking you to verify that the Apache web servers are functional and working as expected.

After a few hours of investigation, you are able to identify a few problems for your manager. You have developed a list of the big five, and you report:

1. The servers have the root user available to be accessed by SSH.
2. The sudoers file had a lot of unnecessary permissions, such as the database user.
3. Many web servers have Python installed without needing it.
4. The Syslog was disabled when the system administrator worked on a proof of concept (POC) for a new centralized log analyzer, and they did the POC in the production environment.
5. All servers are using HTTP and it's possible to use a sniffer and a network protocol analyzer, such as Wireshark, on the network.

> **TIP:** as you know from the previous lessons when you use HTTP, it's possible to see all the data/traffic in plain text.

After the email with the report of all Cyber Security problems that you identified, the manager asks you only to resolve problem five on the list. The manager understands that the problem with the users/passwords "flying" in the network is more urgent than the rest, and the rest of the issues will be resolved by the system administrator next week.

You have to execute the task below to resolve problem five from your list.

Generate an HTTPS certificate and install it on an Apache web server, because there is always a serious threat across the network when you exchange classified information in plain text between a client and a server like Apache. Therefore, you need to implement a security layer between your systems.

This is where an SSL/TLS certificate comes into play, and you have to generate an SSL/TLS certificate. You start by creating a Certificate Signing Request (CSR) on your server, which produces a public/private key pair and a subject identifying the name of your website. Then, you simply send the CSR to a trusted Certificate Authority (CA) for signing at a fee.

Self-signed certificates are highly suitable for internal networks and testing environments. You can deploy and customize certificates using OpenSSL without incurring costs. However, only you can trust the certificate, and anyone trying to connect to your web server will get a warning in their browser.

STEP 1: Generate a private key and a certificate file

```
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/apache.key -out
```

Terminal command flags explained:

Flag	Explanation
`req -x509`	You are using this option to create a self-signed certificate instead of generating a CSR.
`-nodes`	You're telling OpenSSL to avoid encrypting the private key with a passphrase (you don't want to enter the password every time the server restarts. Apache should read this file without your intervention).
`-days 365`	This parameter is the number of days that you want the certificate to remain valid (in this example, 365 days).
`-newkey rsa:2048`	Declaration of an explicit key size and algorithm with this option. The smallest acceptable size is 512 bits, but a more significant value of RSA 2048 offers more security.
`-keyout /etc/ssl/private/apache.key`	Folder location and file name where you're saving the private key.
`-out /etc/ssl/certs/apache.crt`	Folder location and file (.crt) where your signed certificate is saved.

STEP 2: After running the OpenSSL command, you'll have to provide pieces of information to generate your certificate. Enter the appropriate values and press ENTER after every response. The table below is an example of how you can fill out the form.

```
Country Name (2 letter code) [ ]: CA
State or Province Name (full name) [Some-State]: ONTARIO.
Locality Name (eg, city) [ ]: TORONTO.
Organization Name (eg, company) [Internet Widgits Pty Ltd]: Lighthouse Labs
Organizational Unit Name (eg, section) []: Cyber Lab
Common Name (e.g. server FQDN or YOUR name) []: cyberlab.lighthouselabs.ca
Email Address []: [email protected]
```

STEP 3: After filling out the form with the information, the OpenSSL command should generate a private key(.key) file and a certificate(.crt) file in the following locations:

- `/etc/ssl/private/apache.key`
- `/etc/ssl/certs/apache.crt`

You can verify the files using the terminal command LS to list the files. To identify the files, use the parameter -l to check the file's date of creation.

```
$ sudo ls -l /etc/ssl/private/

$ sudo ls -l /etc/ssl/certs/
```

STEP 4: Enable SSL on the virtual host file

Apache maintains a default virtual host file to handle SSL traffic under the /etc/apache2/sites-available directory named default-ssl.conf. You'll make configuration changes in this file for the web server to encrypt data with your certificate.

Use the NANO text editor to open the `/etc/apache2/sites-available/default-ssl.conf` file.

```
$ sudo nano /etc/apache2/sites-available/default-ssl.conf
```

After opening the file, locate the line ServerAdmin webmaster@localhost. Under the line, add the ServerName name directive followed by your domain name or the public IP address of your server, as shown in the example below;

```
ServerAdmin webmaster@localhost

ServerName 10.0.2.15

DocumentRoot /var/www/html
```

In this case, you will only use the server's IP address; don't forget that IP in the table above is just an example. You must identify the server IP and change it to the IP representing your server. You can use the command `ip a` to show the IP configuration.

```
$ ip a
```

STEP 5: Then, in the same file, locate the SSL settings below:

```
SSLCertificateFile      /etc/ssl/certs/ssl-cert-snakeoil.pem
SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
```

Change the values and specify the full file paths of your certificate (`/etc/ssl/certs/apache.crt`) and private key (`/etc/ssl/private/apache.key`).

```
SSLCertificateFile    /etc/ssl/certs/apache.crt
SSLCertificateKeyFile /etc/ssl/private/apache.key
```

Save and close the file when you're done with editing.

STEP 6: Use the Apache a2enmod command to enable the SSL module.

```
$ sudo a2enmod ssl
```

Then, use the a2ensite to enable the default-ssl.conf virtual host file.

```
$ sudo a2ensite default-ssl.conf
```

Restart the Apache web server to load the new changes.

```
$ sudo systemctl restart apache2
```

STEP 7: The process is almost finished. Now you have to enable the SSL port on the firewall (in case of using Ubuntu). In Apache, you should allow secure traffic to be exchanged through port 443 instead of the default port 80, which is used for HTTP.

As a test, you can enable the HTTPS port using the ufw utility. But remember to disable it after the test.

First, enable your firewall by executing the following command:

```
$ sudo ufw enable
```

Output:

```
Firewall is active and enabled on system startup.
```

Allow Apache to listen on ports 80 and 443 by executing the below command:

```
$ sudo ufw allow 'Apache Full'
```

Output:

```
Rule added
Rule added (v6)
```

To avoid locking yourself out of the server, allow SSH connections through the OpenSSH port by running the following command:

```
$ sudo ufw allow 'ssh'
```

Output:

```
Rule added
Rule added (v6)
```

Then, execute the ufw status command to check if you've successfully added the new firewall rules.

```
$ sudo ufw status
```

Output:

```
To                         Action      From
Apache Full                ALLOW       Anywhere
22/tcp                     ALLOW       Anywhere
Apache Full (v6)           ALLOW       Anywhere (v6)
22/tcp (v6)                ALLOW       Anywhere (v6)
```

This output list can change depending on the services that you already enabled on the Linux Server virtual machine.

You can try accessing using different ports or execute additional commands to scan the ports, such as NMAP. Remember to disable the firewall after the tests, or you can block other ports used for different exercises.

```
$ sudo ufw disable
```

STEP 8: Visit the URL using a web browser, use the correct public IP address, and ensure you're using the https://. It is important to note that you may still get a "Connection not secure" message. This is because you are using a self-signed certificate, and not a certificate signed by one of the main certificate authorities.

> ⚠ Remember, for all the steps below, your website is visible on the Linux web browser at "localhost" or "127.0.0.1" from the Linux system.

Additional Practice

Try to search for information about the other four problems if you don't know why they are a problem yet.

After the lab creation, save the entire process for future reference.

You may have to use it in future LAB exercises or lessons.

✓ Mark Completed

Previous
Standard Protocols - SSH Server

Next
Capture an HTTP and an HTTPS Wireshark PCAP

How well did this activity help you to understand the content?

Let us know how we're doing

☆ ☆ ☆ ☆ ☆

W07D3

Wed Aug 7

- Lectures (1)
- Work (7)

6 hrs

- Configure Apache Web Server
- Capture an HTTP and an HTTPS Wireshark PCAP
- Report to IT Manager
- Hash Functions, Data Integrity and Digital Signatures
- Key Establishment Protocols and Management Techniques
- Data Integrity Email
- Project Reading

W07D3 Schedule »