LAMP Server

Server Exploits - Module 3

Components of a Web Application Server

Operating System and Server

To create a web application, you'll need a server and operating system. This is the case for creating most computer-related things.

Linux is extremely popular for web application servers due to its flexibility, low reliance on resources, cost and customization.

Linux servers are traditionally managed through SSH, which you've set up in the previous model.

Windows can be a web application server, and it may be chosen for its enterprise-level support.

Web Server

A web server is a piece of software that can understand web requests and server web pages (usually HTML, CSS and JavaScript). Web applications need some sort of software running on the server to handle sessions and server pages to clients.
There are many kinds, with Apache and Nginx being some of the more popular ones.

APACHE
HTTP SERVER PROJECT

NGINX

Web Server - HTTP Traffic

There are 2 main types of web requests a web server can get - GET and POST. GET is used to get information from the site. This can be like a home page, documents page, announcement page,etc. when visiting the site.

A POST request puts information on the site. For example, when you login to a site, you will make POST request to the site giving you login information. Another example is when posting a comment onto a site.

Method	Meaning
GET	Read data
POST	Insert data

Web Server - Client-side vs Server-side

When it comes to web applications, some processing is done in the client's browser (client-side) and some is done on the web application server (server-side). In security, you will hear about client-side and server-side attacks.

Client-side processing occurs after the client has visited the site. Usually, the client will receive HTML and other client-side field as a response after making a GET or POST request to the server. The client's browser will process these files and render the site on their screen. Some attacks against a web application can maliciously modify the files a client receives when browsing the site. These attacks are referred to as client-side attacks. Common languages for these files are HTML, CSS and Javascript.

Server-side processing is done in the background for web applications, and clients won't see this processing as a response after making requests to the site. Server-side attacks affect the server running the web application and not necessarily the client. An example would be if an attacker could make a specific GET request that cause the server to divulge other user's information. Common languages are PHP, Python and Ruby.

Web Server - A Typical Request

method path protocol

```
GET /tutorials/other/top-20-mysql-best-practices/ HTTP/1.1
Host: net.tutsplus.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: PHPSESSID=r2t5uvjq435r4q7ib3vtdjq120
Pragma: no-cache
Cache-Control: no-cache
```

Web Server - A Typical Request

The HTTP Method

Path to the source on Web Server

Protocol Version Browser supports

```
Post /RegisterDao.jsp HTTP/1.1
Host: www.javatpoint.com
User-Agent: Mozilla/5.0
Accept: text/xml,text/html,text/plain,image/jpeg
Accept-Language: en-us,en
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8
Keep-Alive:300
Connection:keep-alive
User=ravi&pass=java
```

The Request Headers

Message body

Database

A database is where a web application stores all its data. This can be user information, comments, recipes or any other data. Database interactions are done server-side, so a client should not see requests and responses to and from the web server and database.

Common databases are MySQL, PostgreSQL and MSSQL

Scripting Language

Web applications will use a scripting language, like Ruby, Python, Javascript or PHP. A web application can use more than one scripting language.

Scripting can be embedded in HTML forms when a client requests them, to make dynamic pages (pages that change without making requests to the web application). Scripting languages can also be used to communicate between the web server and database.

The Lamp Stack

LAMP

LAMP stands for Linux, Apache, MySQL, PHP.

Why use it?

Pros

- Open-source and free
- Fully Customizable
- Flexible
- Been around for a long time and has a lot of community support
- Still widely-used

Cons

- Require a lot of management and updating
- You need to implement security measures (not built-in)

DVWA - Installation

Installation

We will be installing the Damn Vulnerable Web Application to exploit. First we must install Apache. You may need to apt update before installing.

```
sudo apt install apache2
```

We can then cd into the /srv directory and clone the repository. Make sure you are in /srv. You may need to install git using apt first with apt install git. We use the sudo command with git because we are writing to the /srv directory.

```
bryan@bryan-virtual-machine:/srv$ pwd
/srv
bryan@bryan-virtual-machine:/srv$ sudo git clone https://github.com/ethicalhack
3r/DVWA.git
Cloning into 'DVWA'...
remote: Enumerating objects: 4139, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 4139 (delta 6), reused 11 (delta 2), pack-reused 4119
Receiving objects: 100% (4139/4139), 1.83 MiB | 9.56 MiB/s, done.
Resolving deltas: 100% (1951/1951), done.
bryan@bryan-virtual-machine:/srv$
```

Installation

We will then give full read, write and execute access to everyone on the system with the following command. We use the -R flag for recursive, so all folders in this one inherit the same permissions. Using 777 for chmod is not advisable usually, as it gives access to everyone on the system. This is part of the reason this web app is vulnerable.
chmod 777 -R /srv/DVWA

Linux can be case sensitive, so it’s best to rename the folder to avoid an error with the LAMP stack. From the /srv directory, run the following command.

```
sudo mv DVWA dvwa
```

Installation

By default, Apache uses /var/www/html to store its web files. However, web files should go in /srv. We need to configure Apache 2 to use the /srv directory. Use a text editor (like nano) to edit /etc/apache2/apache2.conf. Find the Directory /var/www/html part of the file, and modify it to look like the picture below.

```
# the latter may be used for local directories served by
# your system is serving content from a sub-directory in
# access here, or in any related virtual host.
<Directory />
        Options FollowSymLinks
        AllowOverride None
        Require all denied
</Directory>

<Directory /usr/share>
        AllowOverride None
        Require all granted
</Directory>

<Directory /srv/dvwa>
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
</Directory>
```

Installation

Then edit /etc/apache2/sites-enabled/000-default.conf and change the DocumentRoot value.

```
# specifies what hostname must appear in the request's Host: header to
# match this virtual host. For the default virtual host (this file) this
# value is not decisive as it is used as a last resort host regardless.
# However, you must set it for any further virtual host explicitly.
#ServerName www.example.com

ServerAdmin webmaster@localhost
DocumentRoot /srv/dvwa

# Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
# error, crit, alert, emerg.
# It is also possible to configure the loglevel for particular
```

Installation

You can now visit http://localhost on your Ubuntu machine, or http://<ubuntu_ip_address> from your host machine. You should see the welcome page.

Note, that the welcome page may look like PHP code now depending on the browser you use. The picture below uses firefox and is from my Ubuntu server.

Installation

Next we have to install a database for our web application. You can use apt to install mysql with the following command.

```
sudo apt install mysql-server
```

We can now access the mysql shell. By default, the root user does not have a password so we will need to change that. Use the following command to enter the mysql shell. You can hit enter when it asks you for a password.

```
sudo mysql -u root -p
```

Installation

To change the password, we must input the following 3 commands. All mysql commands and queries must end with a semicolon. Make sure to use single quotes around your new password. You would replace <password> with your new password.

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password
BY '<password>';

flush privileges;

exit;
```

You can now login as the root user using your new password with:

```
mysql -u root -p
```

Installation

Now we can create a database for DVWA. I have called it DVWADB as its a database for DVWA.

CREATE DATABASE dvwadb;

We will also create a new user for mysql. Similarly to before, replace <user> with a username of your choice. My username was bryan, and all following examples will be using bryan as the username.

```
Create USER '<user>'@'localhost' IDENTIFIED WITH
mysql_native_password BY '<password>';
```

Installation

We will then grant all privileges for our new database to our new user.

```
GRANT ALL PRIVILEGES ON dvwadb.* TO bryan@localhost;
```

You can make sure the command went through with the following command. If you see something similar to what's below, type exit; to exit the mysql shell.

```
mysql> show grants for bryan@localhost
    -> ;
+-----------------------------------------------------------------+
| Grants for bryan@localhost                                      |
+-----------------------------------------------------------------+
| GRANT USAGE ON *.* TO `bryan`@`localhost`                       |
| GRANT ALL PRIVILEGES ON `dvwadb`.* TO `bryan`@`localhost`       |
+-----------------------------------------------------------------+
```

Installation

Lastly, we need to install PHP and PHP for mysql.

```
sudo apt install php
sudo apt install php-mysql
```

We can check if PHP installed correctly by making a PHPinfo page. In /srv/DVWA create a file called info.php with the following content:

```
<?php phpinfo();?>
```

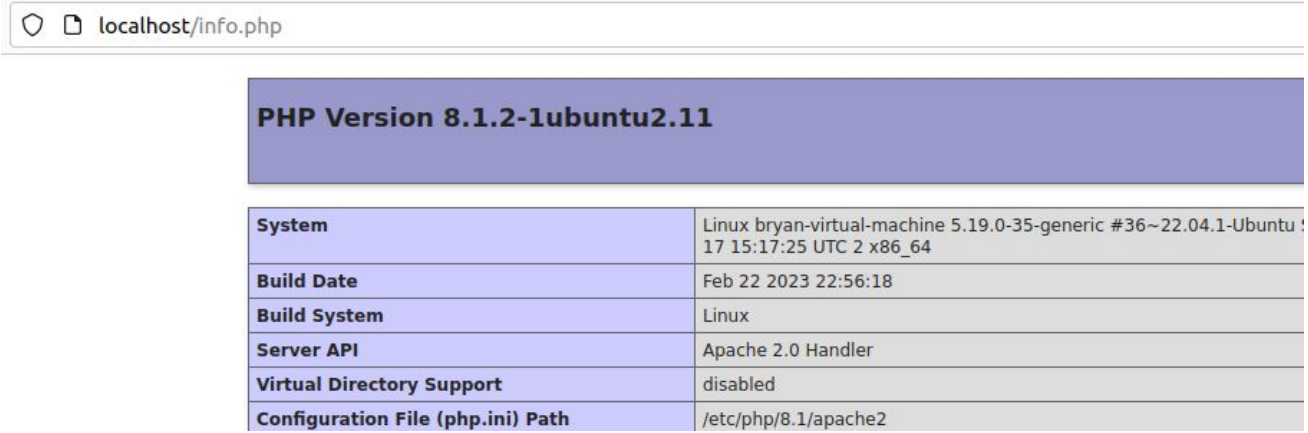
Restart the Apache2 service because we have made changes to the contents of the site.

```
sudo systemctl restart apache2
```

Installation

If you navigate to http://localhost/info.php, you should see a PHPinfo page like the one below. Having one of these is good to see PHP is installed correctly, but is a vulnerability within itself as it discloses so much information about the server. Once you've confirmed PHP is installed correctly, be sure to remove the info.php file with:

```
rm info.php
```



DVWA - Configuration

Configuration

First, edit /srv/dvwa/config/config.inc.php.dist with a text editor and change the following values, changing <password> to be the password of the mysql user you created previously. The values are spread out throughout the document, so you'll need to do some scrolling.

```
$_DVWA[ 'db_database' ] = 'dvwadb';
$_DVWA[ 'db_user' ]     = 'bryan';
$_DVWA[ 'db_password' ] = '<password>';
$_DVWA[ 'default_security_level' ] = 'low';
```

After you save the file, copy it with the following command.

```
cp /srv/dvwa/config/config.inc.php.dist
/srv/dvwa/config/config.inc.php
```

Configuration

Next we will edit out PHP's configuration to enable a setting required by DVWA. The file path will depend on your version of PHP, where mine is 8.1. Therefore, my filepath is /etc/php/8.1/apache2/php.ini. Edit this file with a text editor to have the following value. By default, the allow_url_include value is off, so we'll need to switch it to on. If you use nano, you can use Ctrl+w to find a string in a file.

```
allow_url_include = off
```

We then need to restart Apache and MySQL

```
sudo systemctl restart apache2
sudo systemctl restart mysql
```

Configuration

To check everything is good, navigate to http://localhost/setup.php. You should see everything in green except PHP module gd and reCAPTCHA key.

Setup Check

Web Server SERVER_NAME: **localhost**

Operating system: ***nix**

PHP version: **8.1.2-1ubuntu2.11**
PHP function display_errors: **Disabled**
PHP function safe_mode: Disabled
PHP function allow_url_include: Enabled
PHP function allow_url_fopen: Enabled
PHP function magic_quotes_gpc: Disabled
PHP module gd: **Missing - Only an issue if you want to play with captchas**
PHP module mysql: Installed
PHP module pdo_mysql: Installed

Backend database: **MySQL/MariaDB**
Database username: **bryan**
Database password: ******
Database database: **dvwadb**
Database host: **127.0.0.1**
Database port: **3306**

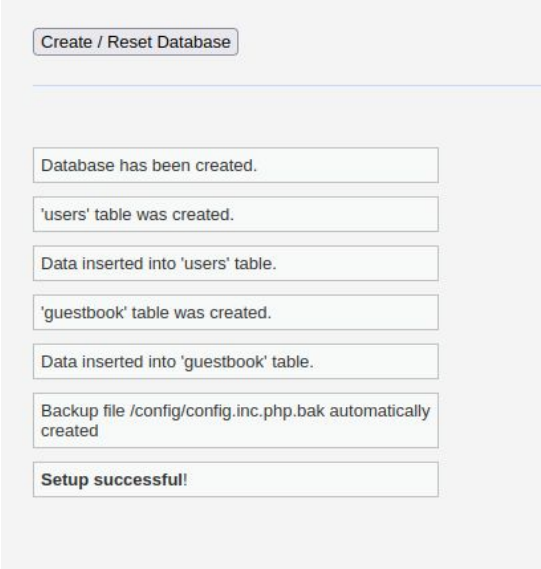
reCAPTCHA key: **Missing**

[User: root] Writable folder /srv/dvwa/hackable/uploads/: Yes
[User: root] Writable file /srv/dvwa/external/phpids/0.6/lib/IDS/tmp/phpids_log.txt: Yes

[User: root] Writable folder /srv/dvwa/config: Yes

Configuration

Click “Create Database” at the bottom and you should see the following steps. If you do, click “Logout” on the left-hand side of the site.



Configuration

You'll be taken to a login page, where you can use the default credentials of admin and password to login (very secure!). Once logged in, you'll see the welcome page and we are ready to begin exploiting!