Catching Files over HTTP/S

vendredi 4 octobre 2024 7:11 PM

Web transfer is the most common way most people transfer files because **HTTP/HTTPS are the most common protocols allowed through firewalls**. Another immense benefit is that, in many cases, the file will be encrypted in transit. **There is nothing worse than being on a penetration test, and a client's network IDS picks up on a sensitive file being transferred over plaintext and having them ask why we sent a password to our cloud server without using encryption.**

We have already discussed using the Python3 **uploadserver** module to set up a web server with upload capabilities, **but we can also use Apache or Nginx.** This section will cover creating a secure web server for file upload operations.

Nginx - Enabling PUT

A good alternative for **transferring files to Apache** is **Nginx** because the configuration is less complicated, and the module system does not lead to security issues as Apache can.

When allowing HTTP uploads, it is critical to be 100% positive that users cannot upload web shells and execute them. Apache makes it easy to shoot ourselves in the foot with this, **as the PHP module loves to execute anything ending in PHP**. Configuring Nginx to use PHP is nowhere near as simple.

Create a Directory to Handle Uploaded Files

```
$ sudo mkdir -p /var/www/uploads/SecretUploadDirectory
```

Change the Owner to www-data

.

```
$ sudo chown -R www-data:www-data /var/www/uploads/SecretUploadDirectory
```

Create Nginx Configuration File

Create the Nginx configuration file by creating the file **/etc/nginx/sites-available/upload.conf** with the contents:

```
server {
    listen 9001;

    location /SecretUploadDirectory/ {
        root    /var/www/uploads;
        dav_methods PUT;
    }
}
```

Symlink our Site to the sites-enabled Directory

```
$ sudo ln -s /etc/nginx/sites-available/upload.conf /etc/nginx/sites-enabled/
```

Start Nginx

```
$ sudo systemctl restart nginx.service
```

If we get any error messages, check /var/log/nginx/error.log. If using Pwnbox, we will see port 80 is already in use.

Verifying Errors

```
$ tail -2 /var/log/nginx/error.log

2020/11/17 16:11:56 [emerg] 5679#5679: bind() to 0.0.0.0:`80` failed (98: A`ddress already in use`)
2020/11/17 16:11:56 [emerg] 5679#5679: still could not bind()
```

```
$ ss -lnpt | grep 80

LISTEN 0    100       0.0.0.0:80      0.0.0.0:*   users:(("python",pid=`2811`,fd=3),("python",pid=2070,fd=3),("python",pid=1968,fd=
```

3),("python",pid=1856,fd=3))

☐ **$ ps -ef | grep** 2811

```
user65    2811   1856  0 16:05 ?       00:00:04 `python -m websockify 80 localhost:5901 -D`
root      6720   2226  0 16:14 pts/0   00:00:00 grep --color=auto 2811
```

We see there is already a module listening on port 80. To get around this, we can remove the default Nginx configuration, which binds on port 80.

Remove NginxDefault Configuration

☐ $ sudo rm /etc/nginx/sites-enabled/**default**

Upload File Using cURL

☐ $ **curl -T** /etc/passwd http://localhost:9001/SecretUploadDirectory/users.txt

☐ $ **sudo tail -1** /var/www/uploads/SecretUploadDirectory/users.txt

user65:x:1000:1000:,,,:/home/user65:/bin/bash

Once we have this working, a good test is to ensure the directory listing is not enabled by navigating to http://localhost/SecretUploadDirectory. By default, with Apache, if we hit a directory without an index file (index.html), it will list all the files. This is bad for our use case of exfilling files because most files are sensitive by nature, and we want to do our best to hide them. Thanks to Nginx being minimal, features like that are not enabled by default.

.