Apache ShellShock Attack Lab

Written by Carter Yagemann

Version 1.0

License

Objective

The objective of this lab is to teach students and cybersecurity professionals about the shellshock bash vulnerability by simulating a real world attack scenario. Upon completing this lab, you should have a better understanding on how shellshock can be applied to real world attacks and what environment conditions are early warning indicators of a possible shellshock vulnerability.

Setup

The server for this lab should be a simple Apache server with one website hosting one web page on port 80. The root of this website should be set to the default directory which is `/var/www/`. The file used to generate the homepage should be `index.cgi` and it can be a simple CGI script containing any code. The following is one example of what `index.cgi` might look like:

```
#!/bin/bash

echo “Content-type: text/html”
echo “”

echo '<html>'
echo '<head>'
echo '<meta http-equiv=”Content-Type” content=”text/html; charset=UTF-8”>'
echo '<title>Apache ShellShock Lab</title>'
echo '</head>'
echo '<body>'
echo '<pre>'
echo 'Hello World!'
echo '</pre>'
echo '</body>'
echo '</html>'

exit 0
```

Since this is a CGI file, the Apache server has to have CGI scripting enabled. Information on how to enable this feature is available here: http://httpd.apache.org/docs/2.2/howto/cgi.html.

Goal

As the attacker, your goal is to compromise the Apache server. There are many ways to accomplish this goal and one possible solution will be outlined in the next section.

Example Solution

The Attack

For this attack, we will use `curl` so we can craft a specific HTTP request. Curl is freely available for all Linux machines. For our attack, we are going to use this command:

```
curl -A “() { :; }; /bin/rm /var/www/*” http://10.0.2.4
```

This is assuming that the server is available at the IP address 10.0.2.4. The payload in this case will cause Apache to delete all the files in the root directory of the website, thereby deleting the website.

Detailed Explanation

The vulnerability in bash has to do with how bash parses and handles functions. When a bash session is created, bash will check its environment variables for any variable with a value that looks like a function, and if it finds any it will parse those functions into the bash session. The problem is this parser has a vulnerability where if there is code after the end of the function declaration, that code will be immediately executed.

In this attack, we set our user agent to look like an empty function declaration along with some malicious code. When Apache tries to generate the homepage we requested, it starts a bash session to execute the CGI script and creates an environment variable called `HTTP_USER_AGENT` to hold our user agent. When bash sees this variable, it thinks it's a function and parses it, executing our malicious code and compromising the server.

Remediation

One solution is to disable CGI scripting, but this is a pretty trivial answer to the problem. The

underlining issue is bash's handling of functions. In general, any time a program uses bash and sets environment variables based on user input, these inputs have to be validated like you would to prevent SQL Injection or Cross-site Scripting (XSS). Otherwise, the boundary between data and code can become confused and the result is a vulnerability like the one seen in this lab.