

SimpleWebServer

SimpleWebServer

Author: Jeff Malavasi, CSEC 731 Project B

General

SimpleWebServer is a basic HTTP/HTTPS server that supports server side execution. It was designed for educational purposes and is vulnerable to many common web exploits such as SQL injection, command injection, and directory traversal. It should not be deployed to production environments.

Starting the Server

Example 1: HTTP

```
python3 testServer.py http
```

Example 2: HTTPS

```
python3 testServer.py https
```

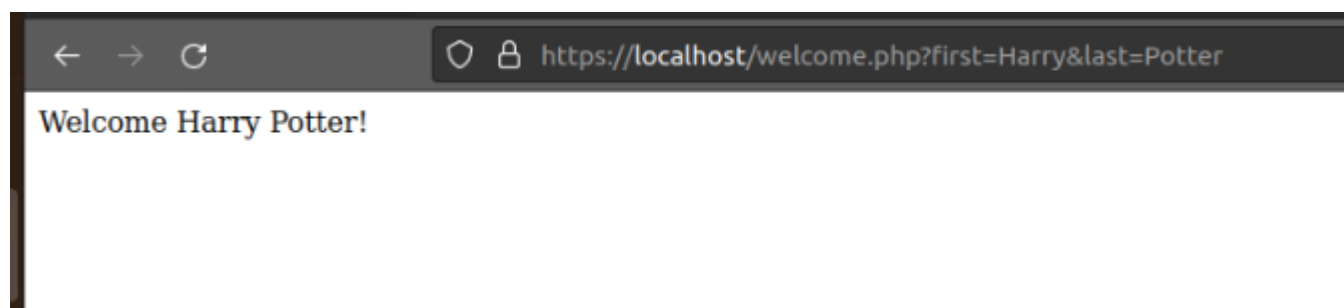
Server Side Execution

Get

1. Start an Instance of Simple Web Server

```
python3 testServer.py https
```

Navigate to <https://localhost/welcome.php?first=Harry&last=Potter>



Source Code

This web page will take in two query parameters (first and last) and use them to generate a welcome page.

```
<?php
// localhost/welcome.php?first=Harry&last=Potter
$fn = $_GET['first'];
$ln = $_GET['last'];

echo "<b>Hello $fn $ln!</b>";

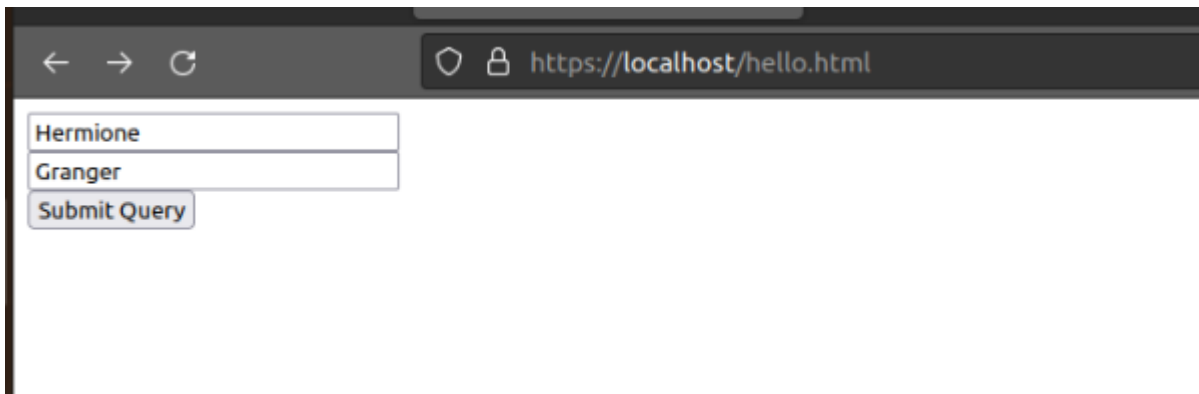
?>
```

Post

1. Start an Instance of Simple Web Server

```
python3 testServer.py https
```

2. Navigate to <https://localhost/hello.html> and enter your name in the form!



A screenshot of a web browser window. The address bar shows 'https://localhost/hello.html'. The page content includes two text input fields. The first field contains the text 'Hermione' and the second field contains 'Granger'. Below these fields is a button labeled 'Submit Query'.

Source Code

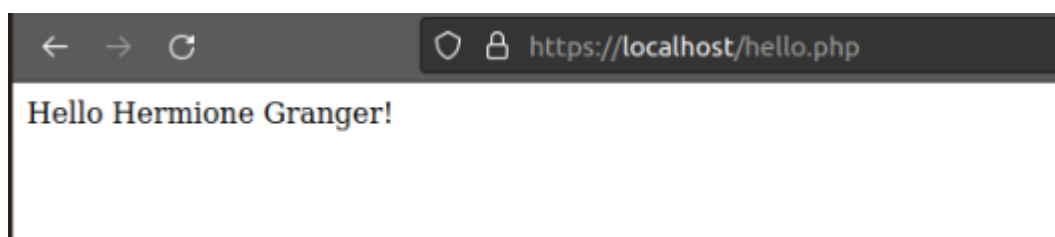
This web form will take in two parameters (first and last) and use them to post to welcome.php and dynamically generate a welcome page.

```
<?php
$fn = $_POST['first'];
$ln = $_POST['last'];

echo "<b>Hello</b> $fn $ln!";

?>
```

Expected Behavior



A screenshot of a web browser window. The address bar shows 'https://localhost/hello.php'. The page content displays the text 'Hello Hermione Granger!'.

Command Injection

HTTP web servers that execute server side code and do not properly parse user input can be vulnerable to command injection. This type of exploit occurs when a bad actor uses a carefully crafted HTTP request to execute arbitrary commands on the host web server. Unsafe data can be injected into all areas of the request including headers, cookies and even the message body. Since the code is executed with the same privileges as the application, it is especially important to avoid running web applications as root where possible.

In The Wild

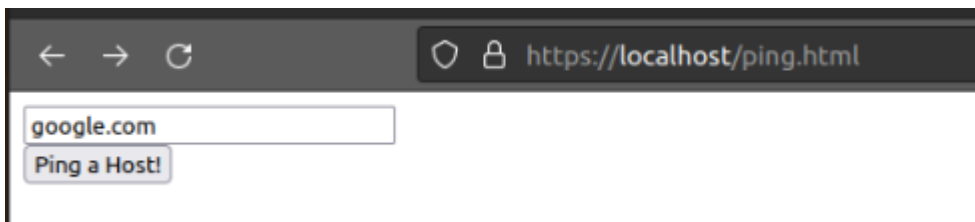
One of the most commonly known command injection exploits was dubbed "Shellshock" and occurred in 2014. At its' peak, there were upwards of 1.5 million attacks per day using this technique.

Example

1. Start an Instance of Simple Web Server

```
python3 testServer.py https
```

2. Navigate to <https://localhost/ping.html>



Source Code

This web form can be used to ping hosts from the web server. It takes a single input string and passes that to the ping command. **Note** that the function **does not** sanitize the input for any special characters like semicolons or confirm that it actually a valid host.

```
<?php
exec(ping -c 4 ". $_GET['host'],$output);
print_r(implode($output));
?>
```

Expected Behavior

Normally when given a valid FQDN or IP address, the web server will return the results of the ping.



Malicious Request

However with the simple introduction of a semi-colon, the server will also execute trailing commands on the host. This can be used to execute arbitrary code, access files outside the web root directory, and even delete files on the web server.

