

Deanna M. Presswood

SDEV 460 Homework 2

July 12, 2018

Testing Framework and Basic Security Controls

Part 1. Testing Framework

1) Describe what each phase encompasses

While all software product creation starts with implementing one of the Software Development Life Cycle (SDLC) Models, yet critical decisions for Software Engineers are to determine, how to carry out the testing framework and display the software development tasks. The testing framework is comprised of several stages: before development begins, during definition and design, during development, during deployment, and maintenance and operation. In the requirement gather stage also known as (**before development begins**) the software development team looks at existing policy, creates new policies, study and research the system and software, and ask questions to find intended use and expectations of customers, users, and developers (TutorialsPoint, 2017). This stage of the testing framework incorporates the technical documentation to illustrate the metrics and criteria needs for the project. This is outlined in steps that discusses how they will be achieved.

Understanding the principles of secure software design should be a priority for the software development team to ensure that the entire SDLC steps have security infused in the process; reviewing security documents and thoroughly understanding & testing the requirements contents to confirm that the structural processes, the design & architecture, Unified Modeling Language (UML), and threat models created for the project describes and explains the interworking of the system (Meuccia & Muller, 2014, p. 24-25). The above

items are known as (**during definition and design**) stage of the testing framework. These items will be scrutinized to ensure that any vulnerabilities are noted and fixed as early as possible. This will eliminate costly code & design mistakes later on in the project.

The coding models are looked at to determine if it will meet the needs of the design specifications. This is done by performing a code walk through to describe the logic & flow of the implemented code (Meuccia & Muller, 2014, p. 25). It's time for the tester to ask questions of the code and perform static code reviews against business requirements and industry technical guides. This step is known as (**during development**) stage of the testing framework.

The software development team makes a testing plan and applies testing cases from gathering information contained in the architectural design of the application workflow steps. Critical parts of the application system is tested to find expected system results this is known as (**during deployment**) stage of the testing framework. A testing goal is to make sure that software applications are created free of vulnerabilities before the release. Various types of software testing are performed in order to achieve different objectives when testing a software application such as penetration testing and configuration management.

An outline of operation details with a named person who will answer system questions, errors in systems functionality, and who will repair code/system fixes. All the framework and the scripts should be kept in a repository if changes needs to be made to the software codes (Sheshajee Dasari, 2011). The library will support modifications of existing features and allow for reusable code for later iterations/phases of the project. This step is known as (**maintenance and operation**) stage of the testing framework.

2) Explain 3 or more activities you will engage in for each phase.

A. Before development begins

- Looks over the existing test strategy and makes changes if necessary.
- Monitors the documentation process of the project.
- Follow coding standards

B. During definition and design

- Identify test conditions.
- Design the test.
- Evaluate testability of the requirements and system.
- Design the test environment set-up and identify and required infrastructure and tools.

C. During development

- Test team perform test development.
- Develop code module.
- Closely examines each block of code.
- Correct defects with code.

D. During deployment

- Develop and lists test cases with the test data by using defined techniques.
- Create multiple test cases in a test suite, which contains detailed instruction for running the test cases.
- Implements the test cases and makes sure it works with the testing environment.

E. Maintenance and operation

- Check which planned deliverables were actually delivered
- Ensure that all incident reports have been resolved.
- Finalize and archive all testing materials such as scripts and test environments for later reuse.
- Evaluate how the testing went and lessons learned for future releases and projects.

Part 2. Security Controls to Test

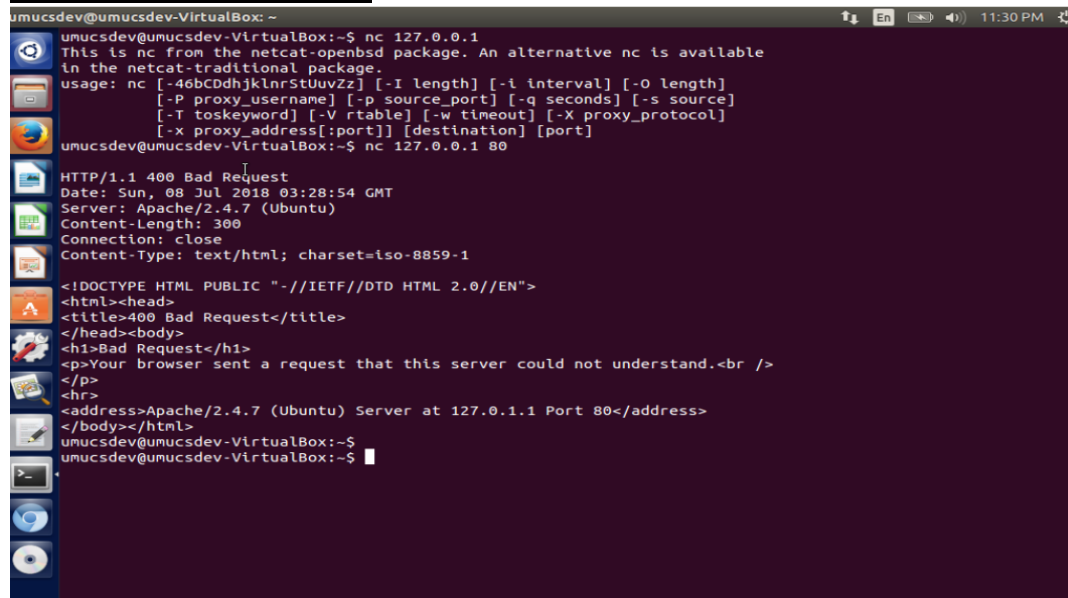
Apply part of this framework in the phase “During development” by engaging in three tests/security controls outlined below to the existing SDEV virtual machine in the default root website.

1) Fingerprint Web Server (OTG-INFO-002)

- Use netcat, httptrint or other tool to discover the web server software vendor and release.

Using netcat nc 127.0.0.1 80 command, I located the web server software vendor and release, Apache/2.4.7 (Ubuntu). The tool output is shown below.

Netcat Web server Output



```

umucsdev@umucsdev-VirtualBox: ~
umucsdev@umucsdev-VirtualBox:~$ nc 127.0.0.1
This is nc from the netcat-openbsd package. An alternative nc is available
in the netcat-traditional package.
usage: nc [-46bCDdhjklmrsStUuvZz] [-I length] [-i interval] [-O length]
        [-P proxy_username] [-p source_port] [-q seconds] [-s source]
        [-T toskeyword] [-V rtable] [-w timeout] [-X proxy_protocol]
        [-x proxy_address[:port]] [destination] [port]
umucsdev@umucsdev-VirtualBox:~$ nc 127.0.0.1 80
HTTP/1.1 400 Bad Request
Date: Sun, 08 Jul 2018 03:28:54 GMT
Server: Apache/2.4.7 (Ubuntu)
Content-Length: 300
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.4.7 (Ubuntu) Server at 127.0.1.1 Port 80</address>
</body></html>
umucsdev@umucsdev-VirtualBox:~$
umucsdev@umucsdev-VirtualBox:~$

```

Figure 1 Netcat Web server Output

- Perform online research about the discovered software vendor and release. Report upon documented vulnerabilities with the release.

The Apache HTTP Server project is managed by volunteers and is part of the Apache Software Foundation. In February 1995, Rob McCool developed the public domain HTTP daemon. Throughout the years with the help of volunteers version 2.0 was released in 2004 to fix problems and to eliminate frequently asked questions with previous versions. Several reported vulnerabilities with Apache version 2.0 were found on the Bug report as of Sunday, July 8, 2018:

1. ID 62508, mutual authentication does not work and gives 400 error message
 2. ID 62519, getting 500 error code in AJP logs intermediately
 3. ID 60717, mod_proxy_http fails with 502 when backend sends 401 and closes connection immediately
- Report upon how you would mitigate any documented vulnerabilities.
 1. To fix ID 62508 error 400 message when trying to use mutual identification is by going through the steps of authentication process which is as follows:
 - a. A client requests access to a protected resource.
 - b. The server presents its certificate to the client.
 - c. The client verifies the server's certificate.
 - d. If successful, the client sends its certificate to the server.
 - e. The server verifies the client's credentials.
 - f. If successful, the server grants access to the protected resource requested by the client.

Then the determination will be made to see where the issue is within the process. For example, if the error is coming from the server's certificate not successfully receiving handshake message to the client. I would then use a packet analyzer, Wireshark to trace the client and the server handshake methods.

2. To fix ID 62519, 500 (Internal Server Error) error code is as follows:

- Check the permissions on one or more files or folders on the PHP or Common gateway Interface (CGI) script.
- Check the timeout rules or the error handling in the php script.
- Check the .htaccess file to see if there is a coding error.

3. To fix ID 60717, mod_proxy_http fails is as follows:

Reinstall the AJP connector in the conf/server.xml file this will connect the AJP to the Apache server port.

2) Review webpage comments and metadata for information leakage (OTG-INFO-005).

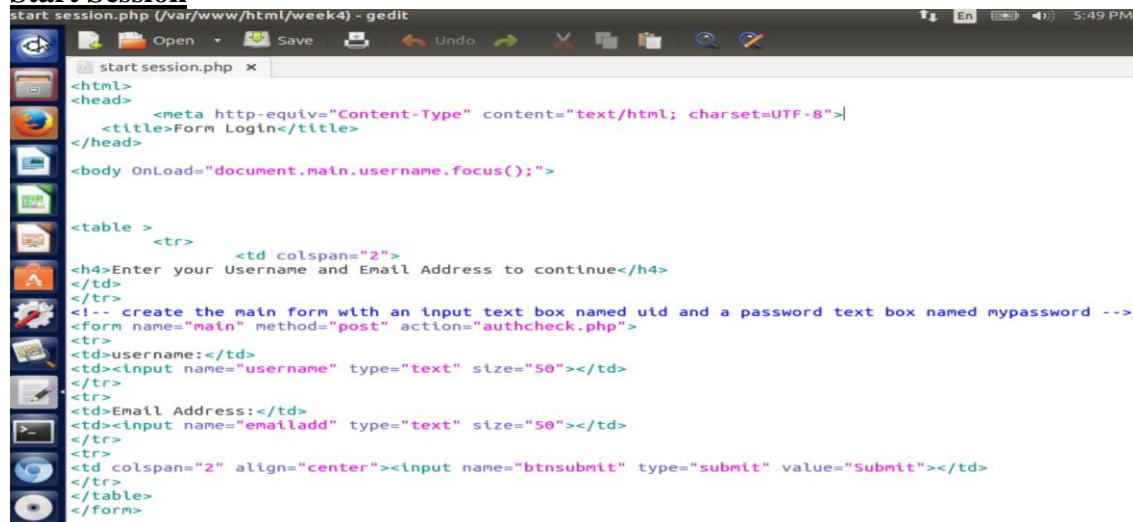
- a. Manually review the sample HTML applications in the Apache Web Server directories. Based upon online research, what are three or more categories of information that would be considered information leakage that is not acceptable?
- According the article, *Information Leakage*, a web site reveals sensitive information such as technical details, environment or user specific data which is an unacceptable practice (Robert Auger, n.d.). Sensitive data can be used to gain access to the application to view files or insert viruses into the application. Information leakage happens because information on the web site is not scrubbed of the HTML/Script comments, improper application or server configuration (Auger, n.d.).

- b. Review the web site to see if there is information leakage in the SDEV information. Report upon what you have discovered and your method of discovery.

- I reviewed the source code for several web files located in week 4 folder.

The start session, loginAuth and post_Submitt php files showed that the Meta tag for the web pages are content type text/html. In the comments of start session and loginAuth it states that the page has user id and password. However, these pages did not show any sensitive data that an attacker could use to access the site.

Start Session

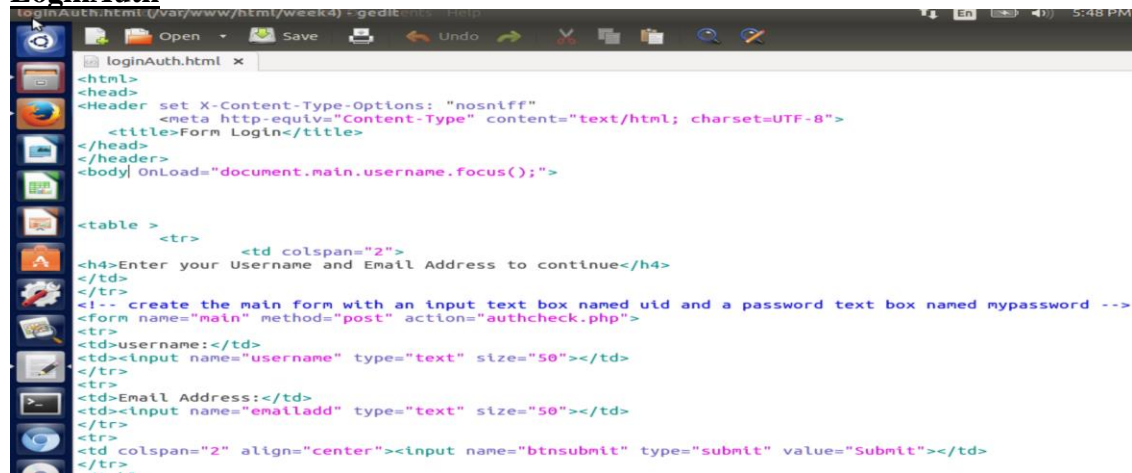


```
start session.php (/var/www/html/week4) - gedit
start session.php x
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Form Login</title>
</head>
<body OnLoad="document.main.username.focus();">

<table >
  <tr>
    <td colspan="2">
      <h4>Enter your Username and Email Address to continue</h4>
    </td>
  </tr>
  <tr>
    <td colspan="2">
      <!-- create the main form with an input text box named uid and a password text box named mypassword -->
      <form name="main" method="post" action="authcheck.php">
        <tr>
          <td>username:</td>
          <td><input name="username" type="text" size="50"></td>
        </tr>
        <tr>
          <td>Email Address:</td>
          <td><input name="emailadd" type="text" size="50"></td>
        </tr>
        <tr>
          <td colspan="2" align="center"><input name="btnsubmit" type="submit" value="Submit"></td>
        </tr>
      </table>
    </form>
  </tr>
</table>
```

Figure 2 Start Session

LoginAuth

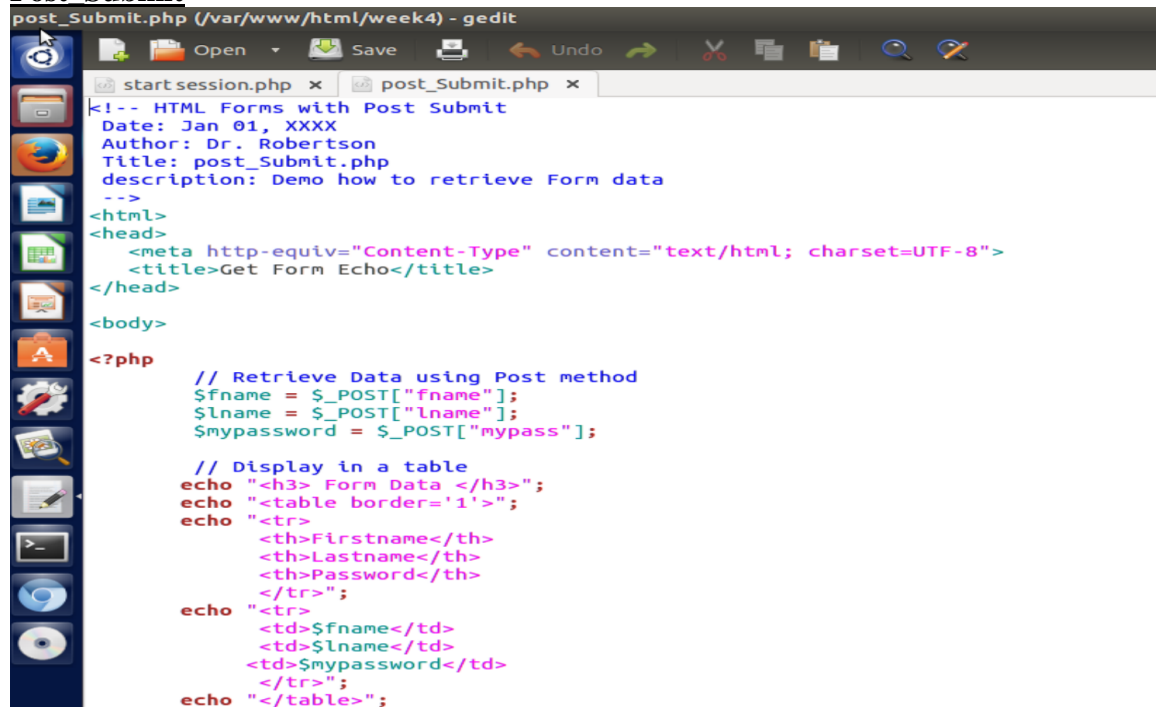


```
loginAuth.html (/var/www/html/week4) - gedit
<html>
<head>
<Header set X-Content-Type-Options: "nosniff"
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
</head>
<body| OnLoad="document.main.username.focus();">

<table >
<tr>
<td colspan="2">
<h4>Enter your Username and Email Address to continue</h4>
</td>
</tr>
<!-- create the main form with an input text box named uid and a password text box named mypassword -->
<form name="main" method="post" action="authcheck.php">
<tr>
<td>username:</td>
<td><input name="username" type="text" size="50"></td>
</tr>
<tr>
<td>Email Address:</td>
<td><input name="emailadd" type="text" size="50"></td>
</tr>
<tr>
<td colspan="2" align="center"><input name="btnsubmit" type="submit" value="Submit"></td>
</tr>
</tr>
```

Figure 3 LoginAuth

Post Submit



```
post_Submit.php (/var/www/html/week4) - gedit
start session.php x post_Submit.php x
<!-- HTML Forms with Post Submit
Date: Jan 01, XXXX
Author: Dr. Robertson
Title: post_Submit.php
description: Demo how to retrieve Form data
-->
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Get Form Echo</title>
</head>
<body>

<?php
// Retrieve Data using Post method
$fname = $_POST["fname"];
$lname = $_POST["lname"];
$mypassword = $_POST["mypass"];

// Display in a table
echo "<h3> Form Data </h3>";
echo "<table border='1'>";
echo "<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>Password</th>
</tr>";
echo "<tr>
<td>$fname</td>
<td>$lname</td>
<td>$mypassword</td>
</tr>";
echo "</table>";
```

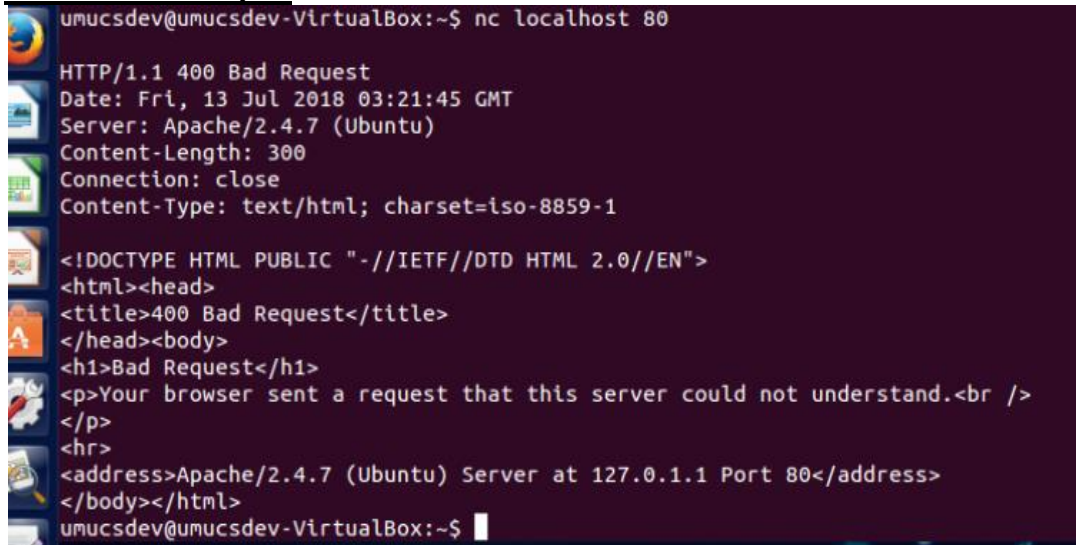
Figure 4 Post_Submit

3) Test HTTP Methods (OTG-CONFIG-006) – See which HTTP methods are available on the virtual machine. Use Netcat or other tool against this SDEV site.

- What HTTP methods are enabled and disabled on this site? Show the output of your tool indicating the HTTP methods.

Tried several times to find out the HTTP methods used with the Apache 2.4.7 server but was unsuccessful. The screen shots are below of my attempts.

Localhost Attempt 1



```
umucsdev@umucsdev-VirtualBox:~$ nc localhost 80
HTTP/1.1 400 Bad Request
Date: Fri, 13 Jul 2018 03:21:45 GMT
Server: Apache/2.4.7 (Ubuntu)
Content-Length: 300
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.4.7 (Ubuntu) Server at 127.0.1.1 Port 80</address>
</body></html>
umucsdev@umucsdev-VirtualBox:~$
```

Figure 5 Localhost Attempt 1

Localhost Attempt 2

```
umucsdev@umucsdev-VirtualBox: ~  
umucsdev@umucsdev-VirtualBox:~$ nc -v 10.0.2.15 80  
Connection to 10.0.2.15 80 port [tcp/http] succeeded!  
  
HTTP/1.1 400 Bad Request  
Date: Fri, 13 Jul 2018 03:17:52 GMT  
Firefox Web Browser 7 (Ubuntu)  
Content-Length: 300  
Connection: close  
Content-Type: text/html; charset=iso-8859-1  
  
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">  
<html><head>  
<title>400 Bad Request</title>  
</head><body>  
<h1>Bad Request</h1>  
<p>Your browser sent a request that this server could not understand.<br />  
</p>  
<hr>  
<address>Apache/2.4.7 (Ubuntu) Server at 127.0.1.1 Port 80</address>  
</body></html>  
umucsdev@umucsdev-VirtualBox:~$
```

Figure 5 Localhost Attempt 2

Localhost Attempt 3

```
</body></html>  
umucsdev@umucsdev-VirtualBox:~$ nc 127.0.0.1 80  
  
HTTP/1.1 400 Bad Request  
Date: Fri, 13 Jul 2018 03:23:43 GMT  
Server: Apache/2.4.7 (Ubuntu)  
Content-Length: 300  
Connection: close  
Content-Type: text/html; charset=iso-8859-1  
  
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">  
<html><head>  
<title>400 Bad Request</title>  
</head><body>  
<h1>Bad Request</h1>  
<p>Your browser sent a request that this server could not understand.<br />  
</p>  
<hr>  
<address>Apache/2.4.7 (Ubuntu) Server at 127.0.1.1 Port 80</address>  
</body></html>  
umucsdev@umucsdev-VirtualBox:~$
```

Figure 5 Localhost Attempt 3

- Which methods (and why) have potentially pose a security risk for a web application. Describe how these pose a risk.

The GET and HEAD methods are considered safe methods, because their goal is to retrieve data. GET is used to request data from a specific resource and receive a return on the request. While, HEAD is used to request data, however does not receive a return back (HTTP Request Methods, 2018). These two methods can expose sensitive data in the request. This is dangerous, because it gives a person a way to access the system application resources. Or an “Attacker” can inject a virus in the system by having access through the sensitive data.

The POST method is used to send data to a server to create/update a resource (HTTP Request Methods, 2018). This method allows multiple requests, which produce the same result. According to Testing for HTTP Pollution, authentication bypass can happen where a user can gain access to an application and take control of the system (OWASP Testing Guide, May 7, 2017).

The PUT method is used to send data to a server to create/update a resource (HTTP Request Methods, 2018). This method allows multiple requests that produces the same resource multiple times. Malicious files can be loaded through the PUT command.

References

- Apache HTTP Server Project. (2018). The Apache Foundation. Retrieved from https://httpd.apache.org/security/vulnerabilities_22.html
- Auger, Robert. (n.d.) Information leakage. The Web Application Security Consortium. Retrieved from <http://projects.webappsec.org/w/page/13246936/Information%20Leakage>
- Dasari, Sheshajee. (21 October 2014). *How to Design an Effective Test Automation Framework*. Retrieved from <https://www.evoketechnologies.com/blog/test-automation-framework-design/>
- HTTP Request Methods. (2018). W3schools Website. Retrieved from https://www.w3schools.com/tags/ref_httpmethods.asp
- Meucci, Matteo and Muller, Andrew. (2014). OWASP Testing Guide 4.0. Retrieved from <http://www.owasp.org>
- OWASP Testing Guide v4. (07 May 2017). Testing for HTTP Pollution. Retrieved from [https://www.owasp.org/index.php/Testing_for_HTTP_Parameter_pollution_\(OTG-INPVAL-004\)](https://www.owasp.org/index.php/Testing_for_HTTP_Parameter_pollution_(OTG-INPVAL-004))
- TutorialsPoint. (2017). Software Development Life Cycle. Retrieved from http://www.tutorialspoint.com/software_engineering/software_development_life_cycle.htm