Introduction to OWASP ZAP

Overview

This lab walks you through using ZAP by OWASP. ZAP is a vulnerability analysis tool used to scan Web applications for possible software flaws. As an introduction to using ZAP, you will scan and interrupt http protocols in PHP code we developed in week 4. You will also run the attack scanner on code you developed in week 4.

Important: Do not attempt to use these tools against any live Web site. It is illegal to do so. You can only scan sites you have written permission to scan. You should use the virtual machine on applications you developed running on the localhost only.

Learning Outcomes:

At the completion of the lab you should be able to:

1. Launch ZAP and view Web sites history and input parameters
2. Use ZAP to intercept http messages and change their content to Identify possible vulnerabilities
3. Read and analyze reports produced from ZAP and prioritize and fix alerts associated with software issues

Lab Submission Requirements:
After completing this lab, you will submit a word (or PDF) document that meets all of the requirements in the description at the end of this document. In addition, your associated files should be submitted. You can submit multiple files in a zip file.

Virtual Machine Account Information
Your Virtual Machine has been preconfigured with all of the software you will need for this class. You have connected to this machine in the previous labs. Reconnect again using the Remote Desktop connection, your Administrator username and password.

Part 1 – Launch ZAP and view Web sites history and input parameters

This exercise will walk you through Launching ZAP and allow you to become comfortable with the GUI for ZAP within your AWS virtual machine. We will use ZAP to begin to analyze some of the PHP Web applications we created in week 4.

1. Verify your Firefox browser has the Proxy properly configured. To do this, launch Firefox, open the menu and click on Options. The Firefox menu is located at the top right. Scroll down to the bottom of the Options and select “Settings …” from the Network Proxy section as shown in figure 1.

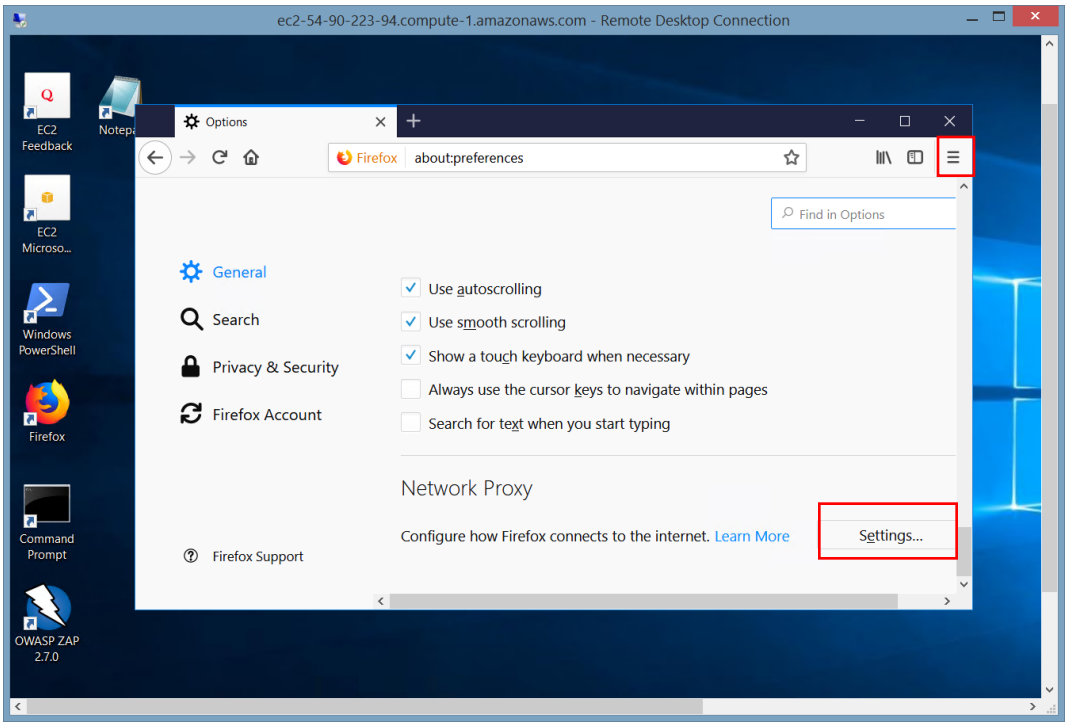

Figure 1 Selecting the Network Proxy Settings

2. Enter localhost for the HTTP Proxy with Port of 8080. Also, be sure Use this proxy server for all protocols is selected as shown in figure 2. Be sure to scroll down and remove anything listed in the "**No Proxy for"** text area.

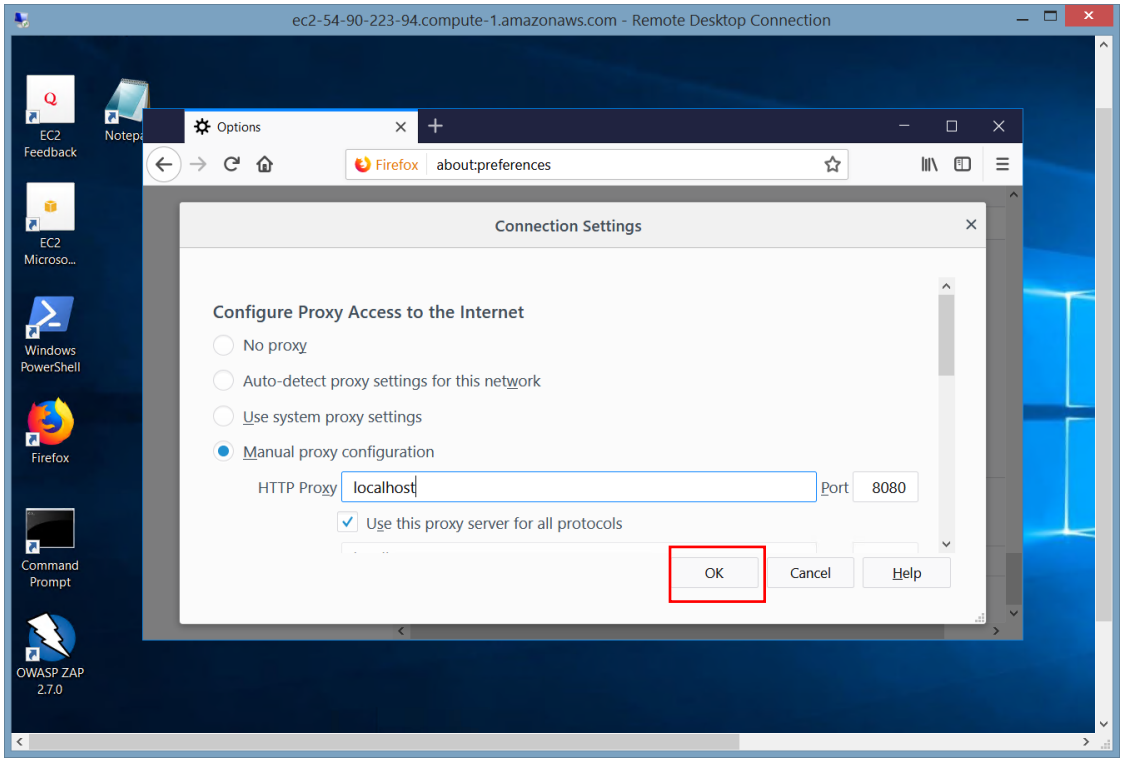

Figure 2 Configure the Network Proxy

Click “Ok” to continue. Now your Browser will send HTTP messages to the already preconfigured ZAP proxy.

Important: You will need to change your Browser settings back to “No proxy” once you are ready to perform normal Browsing to either the localhost or the Internet for the future.

3. To Launch ZAP, double click on the OWASP ZAP icon as shown in figure 3. Select “no, I do not want persist the session at this time” and select start.

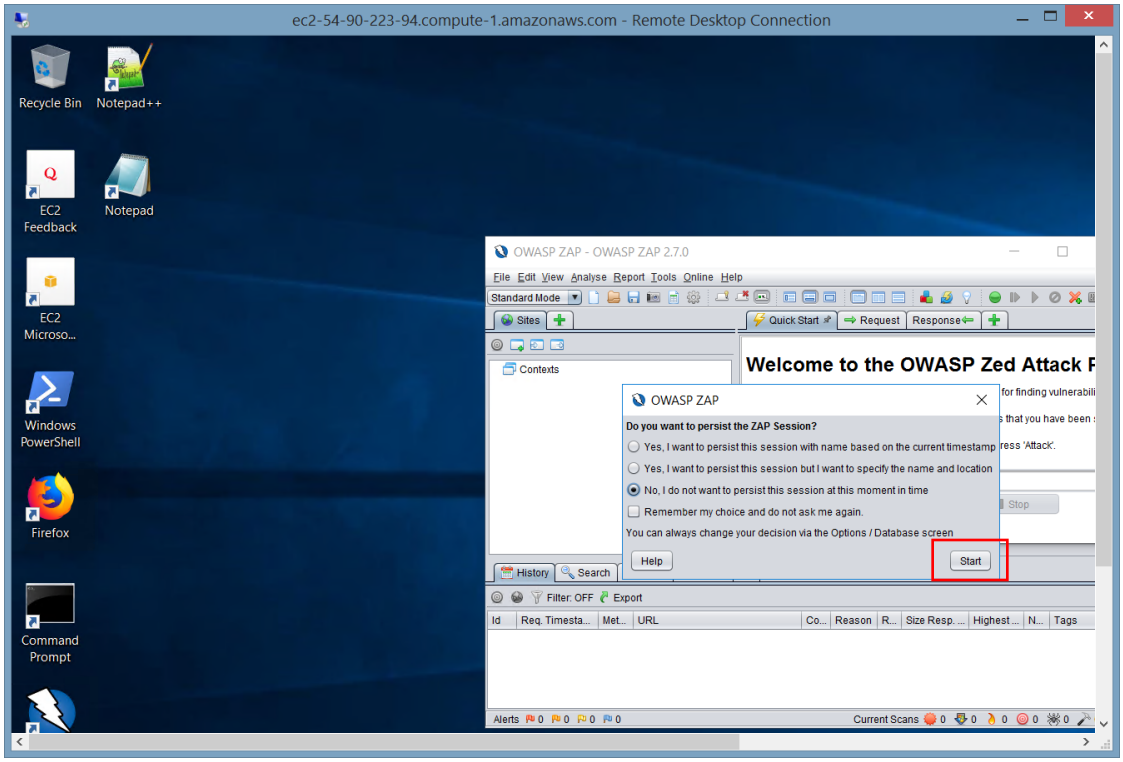

Figure 3 Launching ZAP

The GUI will look very complicated at first as there is much functionality that this tool can perform. Over the next couple of weeks you will be exposed to many of the critical features.

4. To use the Sites and History tabs, you will need to launch your Browser. Open your Browser and launch the DemoGetForm.html file from week 4. The URL should be: localhost/SDEV300/DemoGetForm.html.

   Review the ZAP tool you will see the Sites Icon has listed the GET:DemoGetForm.html file. Also, notice the History tab as information related to this site as shown in figure 4/

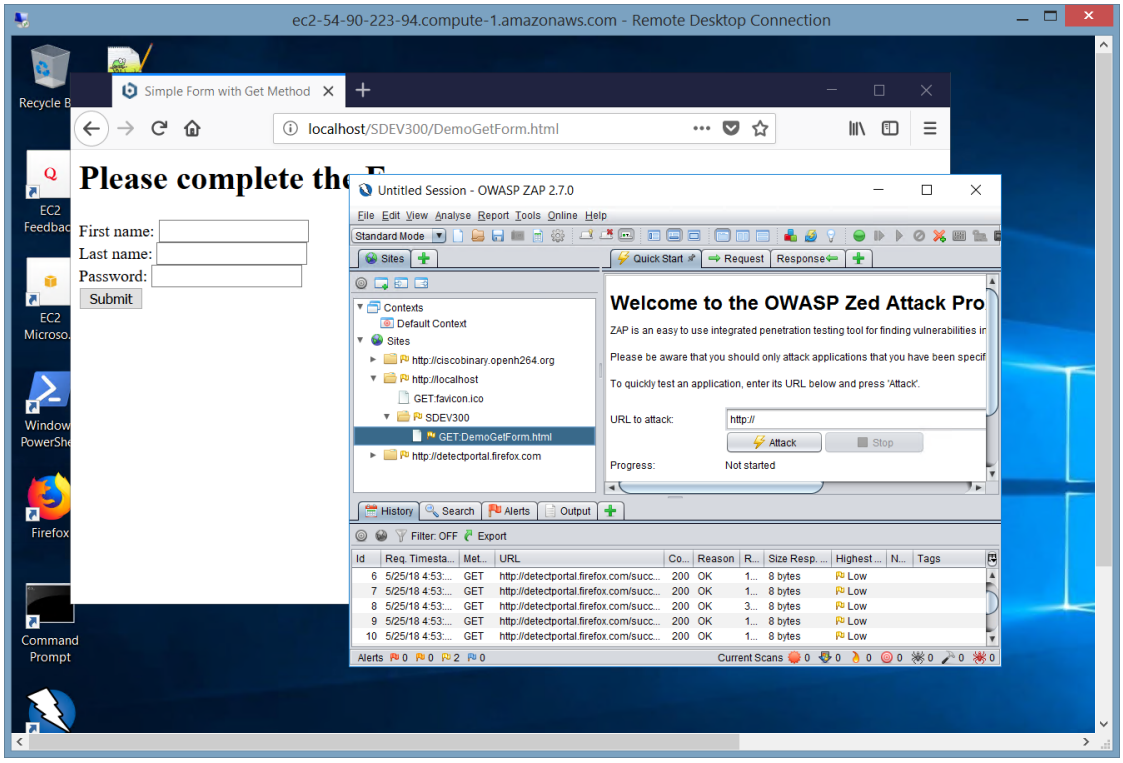

Figure 4 ZAP DemoGetForm Launch

You may also notice some additional Web traffic from the Firefox browser such as detectportal.firefox.com and ciscobinary.openh254.org or others. Depending upon if you are using VPN or other network configurations, these are data sets being sent back and forth for various reasons. Typically, if not disabled, the Firefox will send the detectportal.firefox.com HTTP request about once a minute. Your VPN may also send other requests. Although interesting how often these requests are being sent, for now, these can be ignored.

5. If you click on the Request and Response tabs on the right site of the tool, you will see both the header and body of the DemoGetForm.html file.

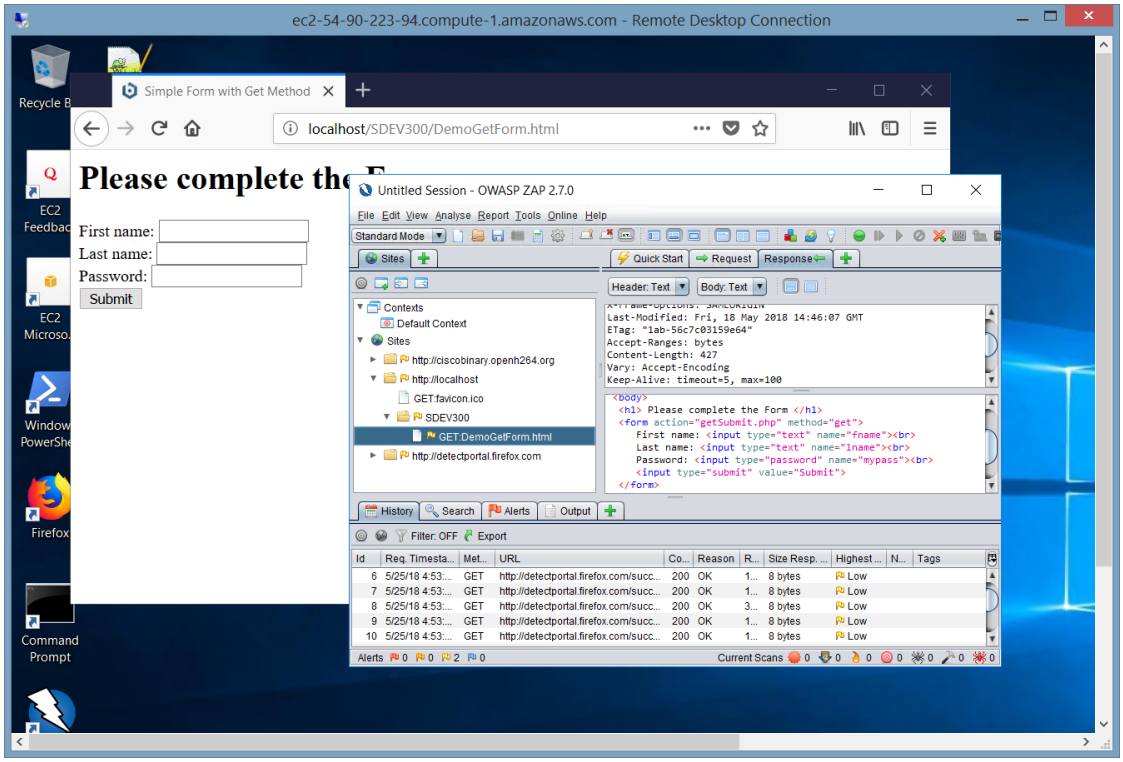

Figure 5 ZAP Response Tab

Notice in the request tab you see the HTTP request sent to the server:

GET http://localhost/SDEV300/DemoGetForm.htm

Then in the response tab, you see what is sent back to the browser. This is essentially just the Form of the Web page we created:

```
<html>
  <head>
     <title>Simple Form with Get Method </title>
 </head>
 <body>
  <h1> Please complete the Form </h1>
  <form action="getSubmit.php" method="get">
     First name: <input type="text" name="fname"><br>
     Last name: <input type="text" name="lname"><br>
     Password: <input type="password" name="mypass"><br>
     <input type="submit" value="Submit">
  </form>
 </body>
</html>
```

You should carefully review the request and response details noting how much information is revealed from this transaction. Information such as Web Server and Operating system are revealed. All HTML source code and comments are also provided.

For example, by reviewing the request header, I can easily see the Browser type and version, the O/S and even language and cookie information:

```
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Cookie: UMUCGamer=CMSC325
```

Although there are other methods for revealing the source code of an HTML page (including using the Browser itself), comments and other possible sensitive information can be gathered by reviewing the code in the response. Notice in the code below, the comments lead a user to a possible password to try:

```
<html>
  <head>
     <title>Simple Form with Get Method </title>
 </head>
 <body>
 <!-- Body of the Form is here -->
  <h1> Please complete the Form </h1>
  <form action="getSubmit.php" method="get">
     First name: <input type="text" name="fname"><br>
     Last name: <input type="text" name="lname"><br>
        <!-- My Password should be 'testing' -->
     Password: <input type="password" name="mypass"><br>
     <input type="submit" value="Submit">
  </form>
 </body>
</html>
```

When reviewing and creating code, be careful your comments don't provide an adversary with a free pass to your sensitive and very valuable corporate and agency data.

6. Complete the DemoGetForm.html and then click on the Submit button.

As shown in figure 6, notice the addition of the get_Submit.php(fname,lname mypass) to the history and sites tab. Also, notice the Request and Response tabs as the additions to the history tab near the bottom.

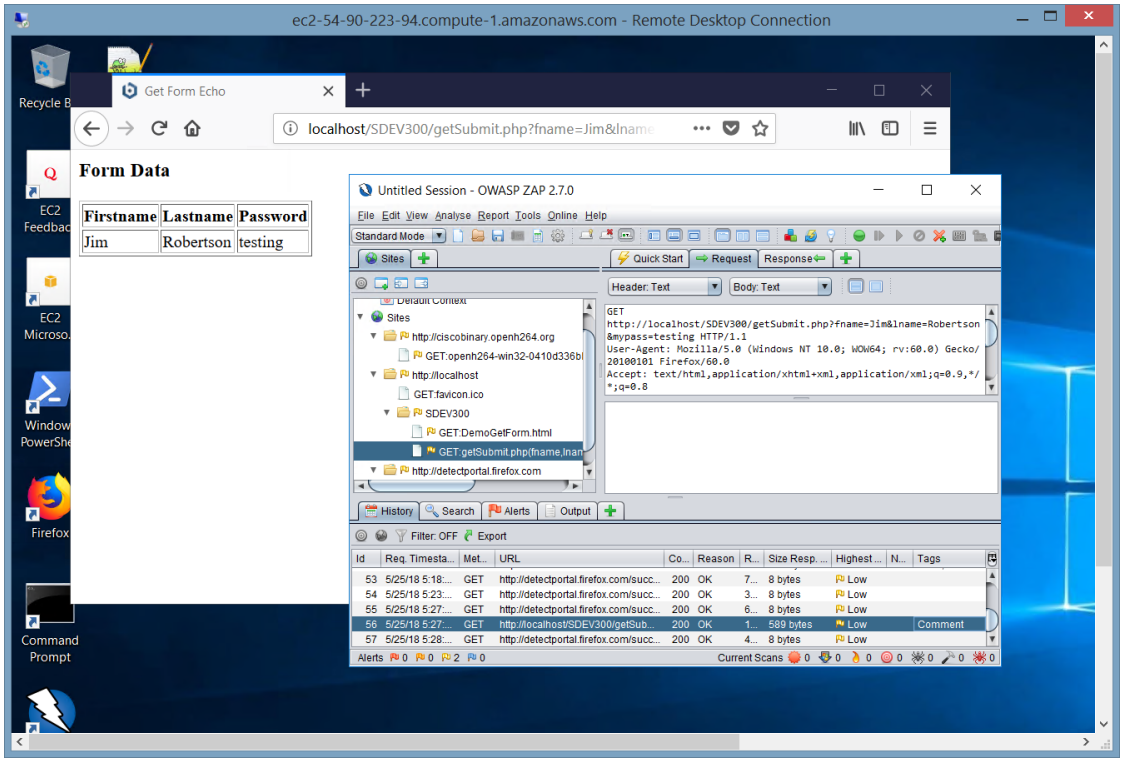

Figure 6 ZAP Form Submit

Notice the HTTP Get request in this case included the Query String of:

```
GET
http://localhost/SDEV300/getSubmit.php?fname=Jim&lname=Robertson&mypass=testi
ng HTTP/1.1
```

Also notice the Response Body provides the HTML matching the Form data response seen on the Browser output:

```
<!-- HTML Forms with Get Submit
 Date: Jan 01, XXXX
 Author: Dr. Robertson
 Title: getSubmit.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>
<h3> Form Data </h3>
       <table border='1'>
            <tr>
             <th>Firstname</th>
             <th>Lastname</th>
             <th>Password</th>
             </tr>
             <tr>
             <td>Jim</td>
             <td>Robertson</td>
            <td>testing</td>
```

```
            </tr>
        </table>
    </body>
</html>
```

The Alerts tab near the bottom of the ZAP page as shown in figure 7, helps to identify possible vulnerabilities. Not only code issues are identified but also Web server and other parameters settings might be listed.

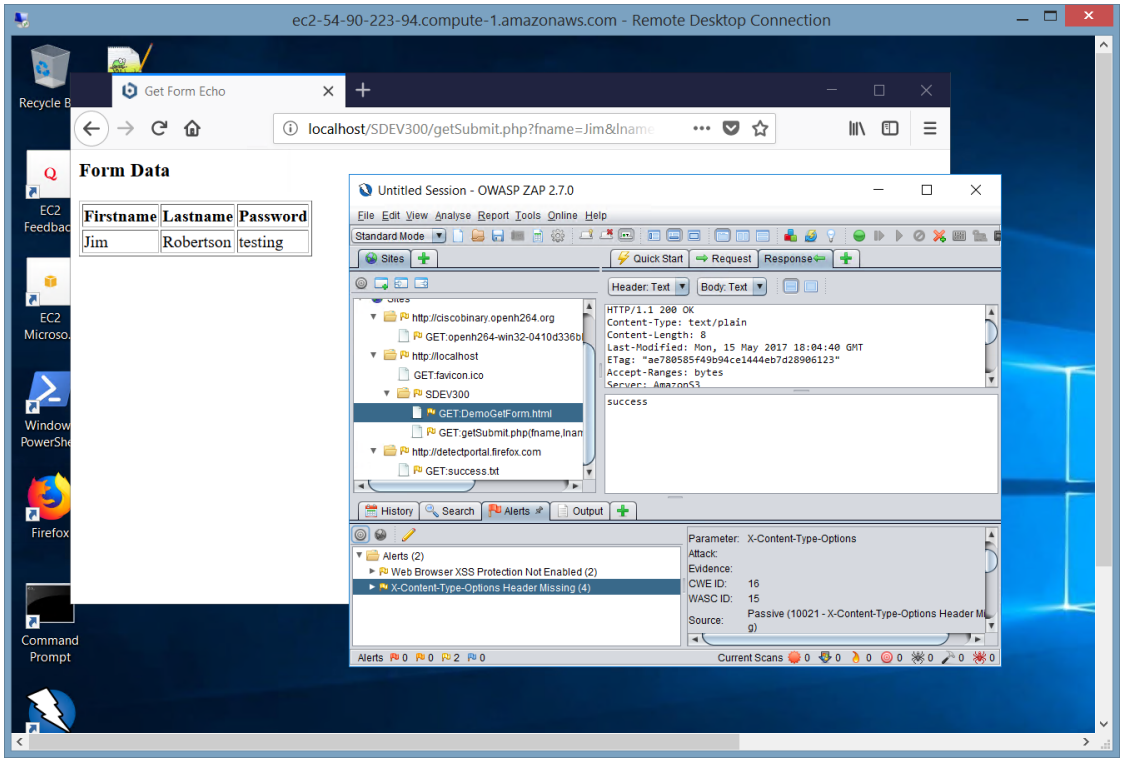

Figure 7 ZAP Alerts tab

We will dissect some of these alerts soon. However; part of this course is to help you develop a process to identify and correct issues you may have not seen before. **Hence, you will often have to research, find and implement fixes based on the research that you have not seen in other web applications presented here.**

You should experiment with using ZAP by running the other Web applications you created in week 4. As you run these applications, be sure to review the Sites, History, Request, Response and other tabs. Look up the issues and alerts found and see if you can correct those vulnerabilities.

Part of the process may also be to prioritize threats. Clearly, severe threats need the first attention.

Part 2 Use ZAP to intercept http messages and change their content to Identify possible vulnerabilities

One of the strong features of the ZAP tool is the ability to interrupt HTTP message and change the values in an attempt to find software flaws. In this exercise, we will use the week4 PHP web applications and demonstrate how the input parameters sent from the form can easily be changed and redirected back to the application with the new parameters.

To interrupt an HTTP transmission, you use the Break Points functionality within ZAP.

1. To set a Break point, right mouse click the specific Site you want to Break on. For example, if we want to put a Break point for the getSubmit.php file, we would right mouse click that Site, select **Break** and then select **Save** as shown in figure 8.

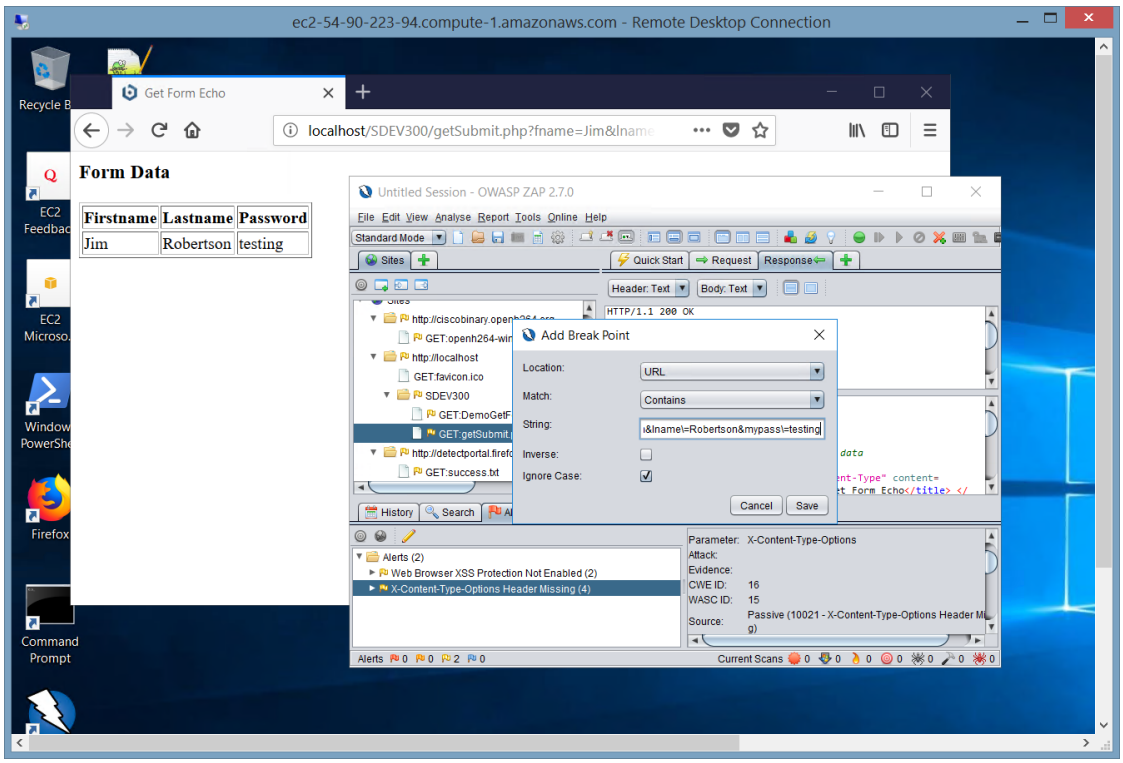

Figure 8 Insert a Break point with ZAP

Once the Break Point is set, it will display in the Break Points tab at the bottom of the ZAP tool as shown in figure 9.

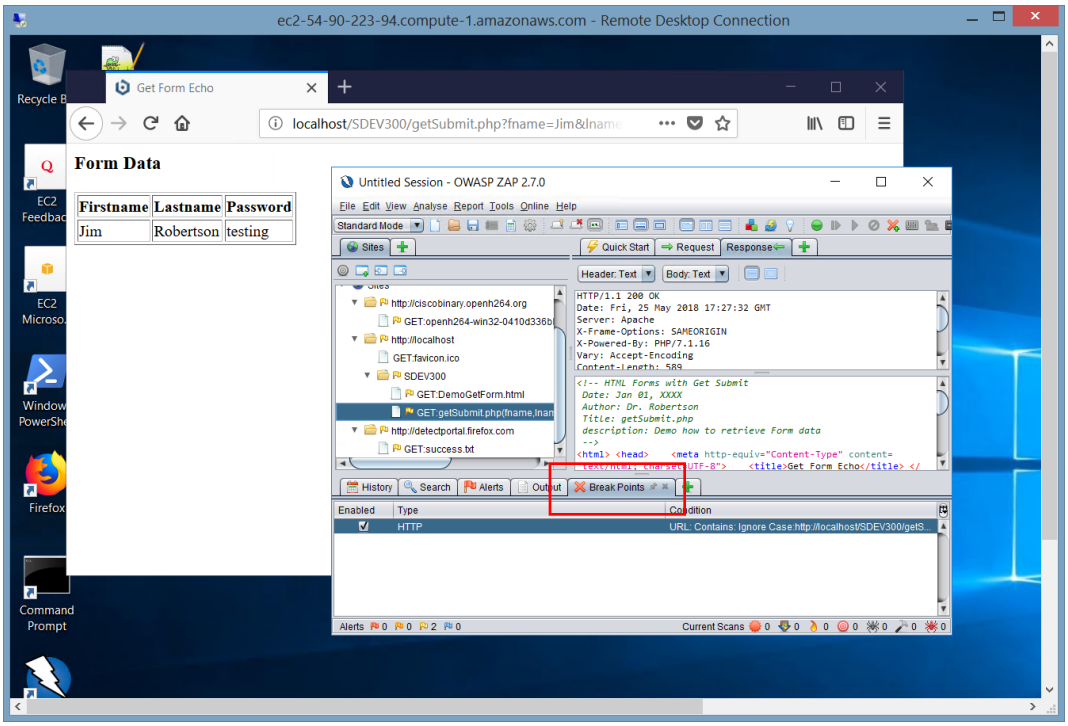

Figure 9 Break point Tab in ZAP

2. To test the break point, launch the DemoGetForm.html application, complete the form and then press Submit. You will notice the browser seems to hang as the http request has been intercepted by the proxy and is awaiting action in ZAP. (See figure 10).

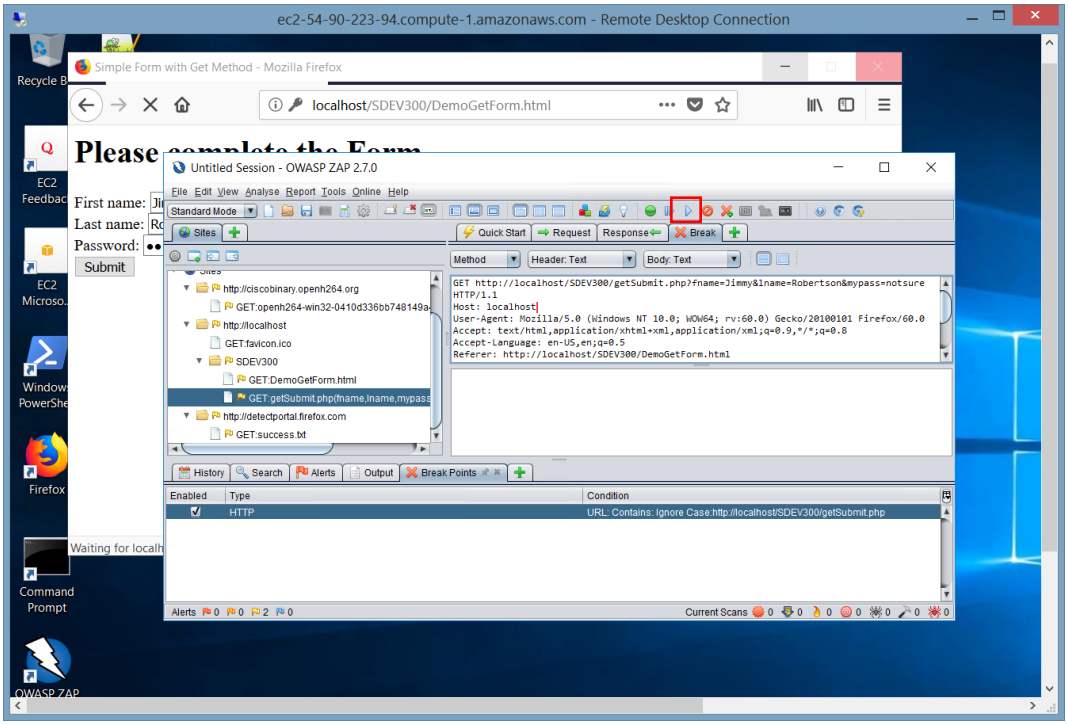

Figure 10 Awaiting form entry in ZAP

3. You can now modify the original data by modify the GET string as appropriate and select the send icon on the ZAP interface. The send icon is the right arrow button near the top of the ZAP tool. It looks like the play button. It is highlighted in the red square in figure 10 to help you locate it.

The following is an example of changing the Query string. The first string was what was originally sent. The second string illustrates changing the variable mypass to testing.

```
GET
http://localhost/SDEV300/getSubmit.php?fname=Jim&lname=Roberts&mypass=notsure
HTTP/1.1
```

```
GET
http://localhost/SDEV300/getSubmit.php?fname=Jim&lname=Roberts&mypass=testing
HTTP/1.1
```

After entering the new mypass value and clicking the send icon, the following data becomes available in the response. Notice the original password has been altered to “testing”.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Get Form Echo</title>
</head>
<body>
<h3> Form Data </h3>
<table border='1'>
<tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Password</th>
 </tr>
<tr>
    <td>Jim</td>
    <td>Roberts</td>
    <td>testing</td>
</tr>
</table>
</body>
</html>
```

If you click the send button again, you will then be able to see the new response in the window as shown in figure 11.

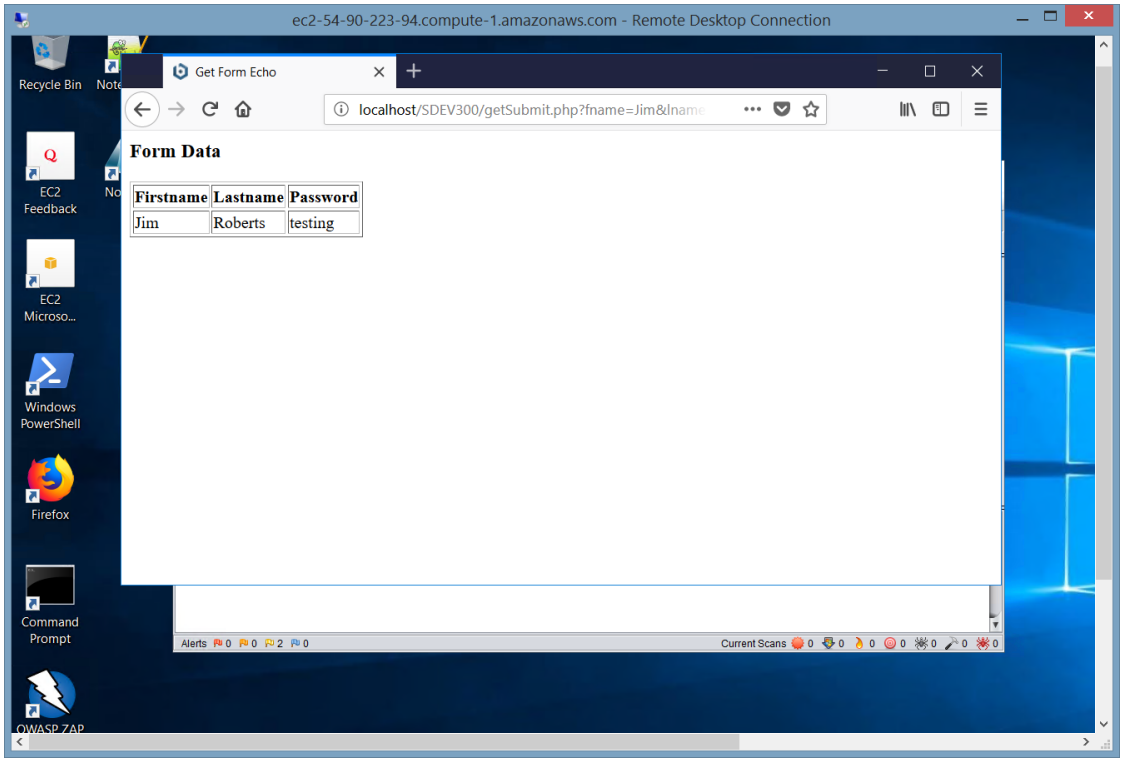

Figure 11Intercepting and Modifying the Query String

4. To disable the Break point, simply uncheck the enabled option in the list of Break points at the bottom of the ZAP tool as shown in figure 12.

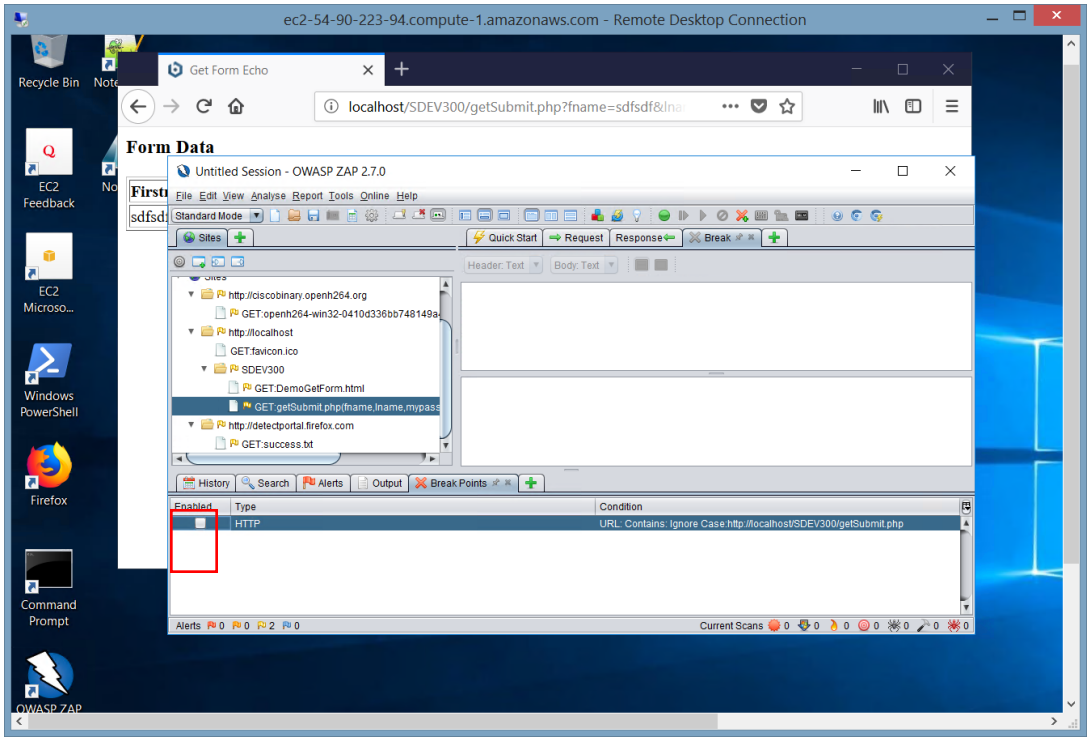

Figure 12 Disable the Break point

Although the Query string is displayed in a different location in the ZAP tool for POST HTTP requests, the data is still viewable and modifiable. For example, figure 13, illustrates the results of modifying the post results after placing a Break point on the postSubmit.php file. Notice the mypass value was modified.

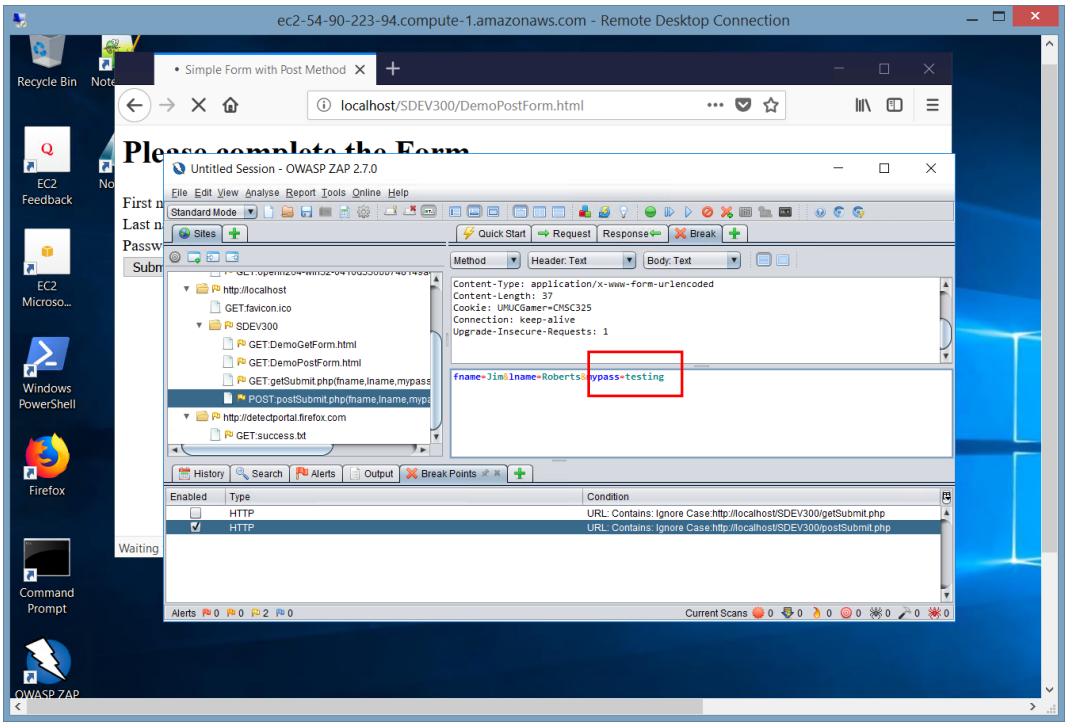

Figure 13 ZAP Post Modify

This is where the person-in-the-loop analysis and discovery takes place. Manipulating the data sent into the applications provides insight into the vulnerabilities of the application. For example, you may discover that sending in a null password allows access to the system. You may also discover providing an admin username and brute force password guess may provide additional system privileges.

You should experiment with all of the week4 PHP applications to see how parameters and information can be changed and the resulting impact on the application.

Part 3 Read and analyze reports produced from ZAP

In this lab, we will run the automatic scanning feature of ZAP and then generate HTML Alert reports for the DemoGetForm.html and DemoPostForm.html and discuss approaches to prioritize and mitigate the issues found in each Web applications.

1. Under the Quick Start tab of ZAP, enter the http://localhost/SDEV300/DemoGetForm.html URL and click “Attack” as shown in figure 14.

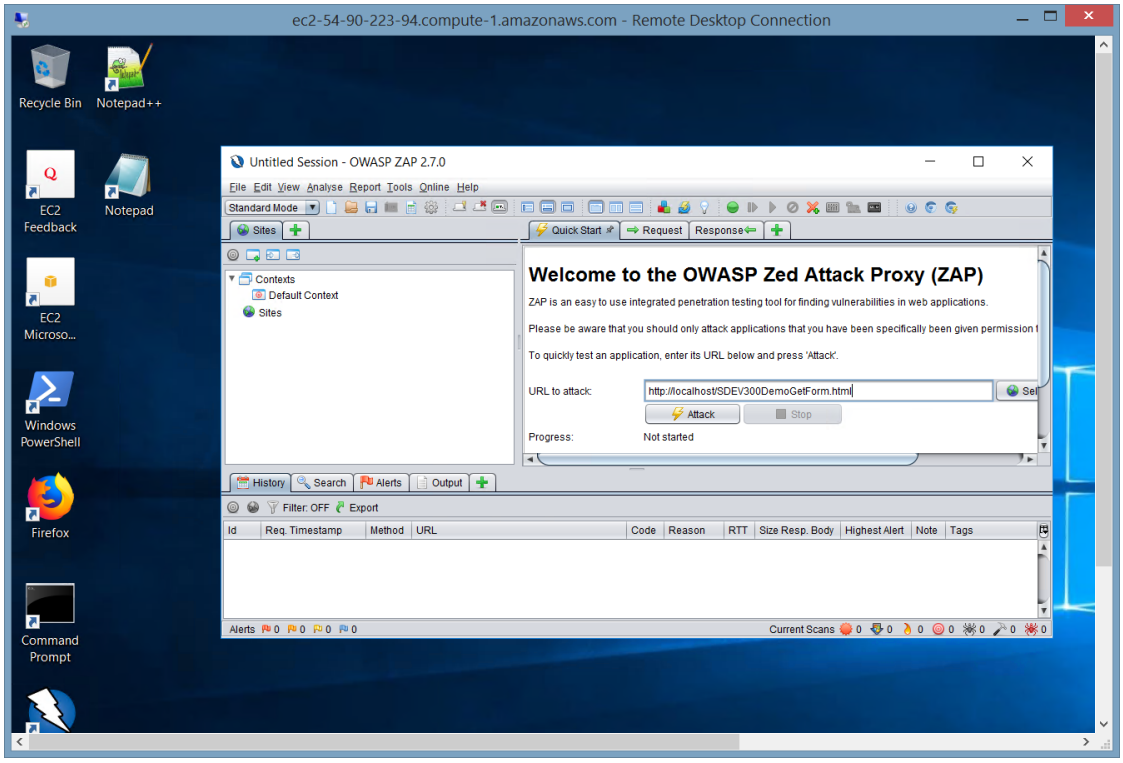

Figure 14 ZAP Attack

2. As the scan runs, you will see hundreds of requests logged into the Active Scan tab (see figure 15). You will also see several Alerts. Alerts from the scan provide possible vulnerabilities. The color of flag indicates the risk level of issue found. (See figure 16).

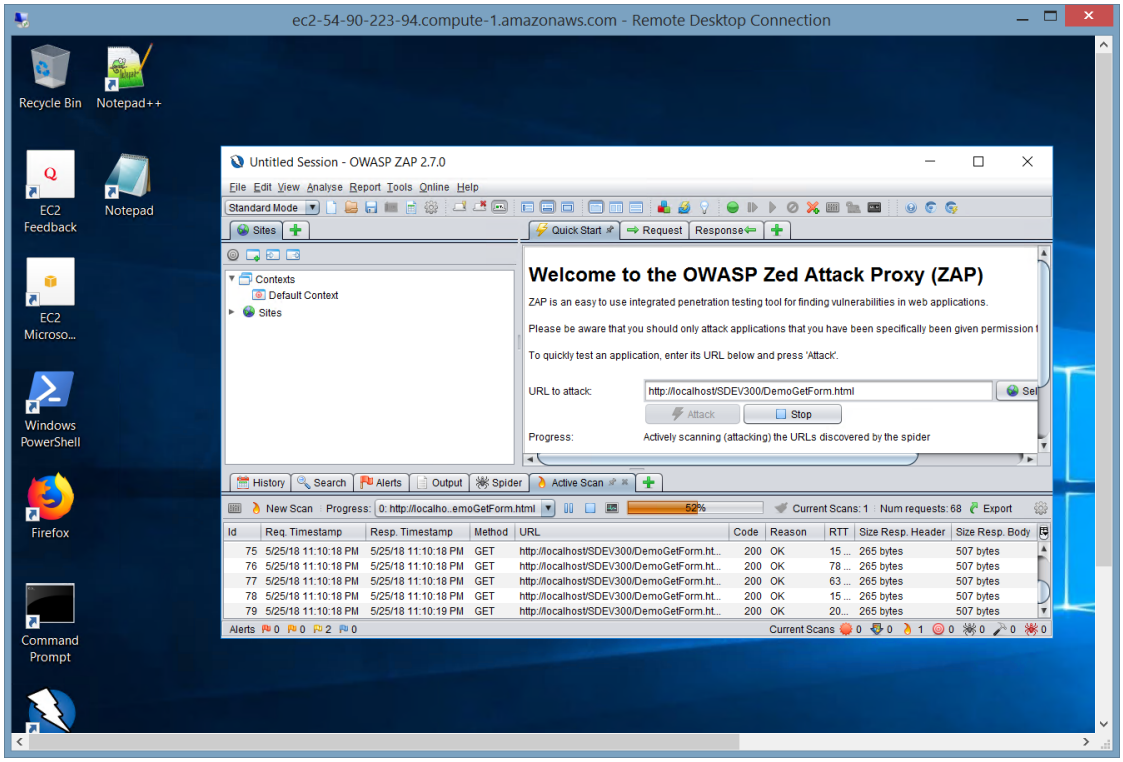

Figure 15 Results from ZAP Attack

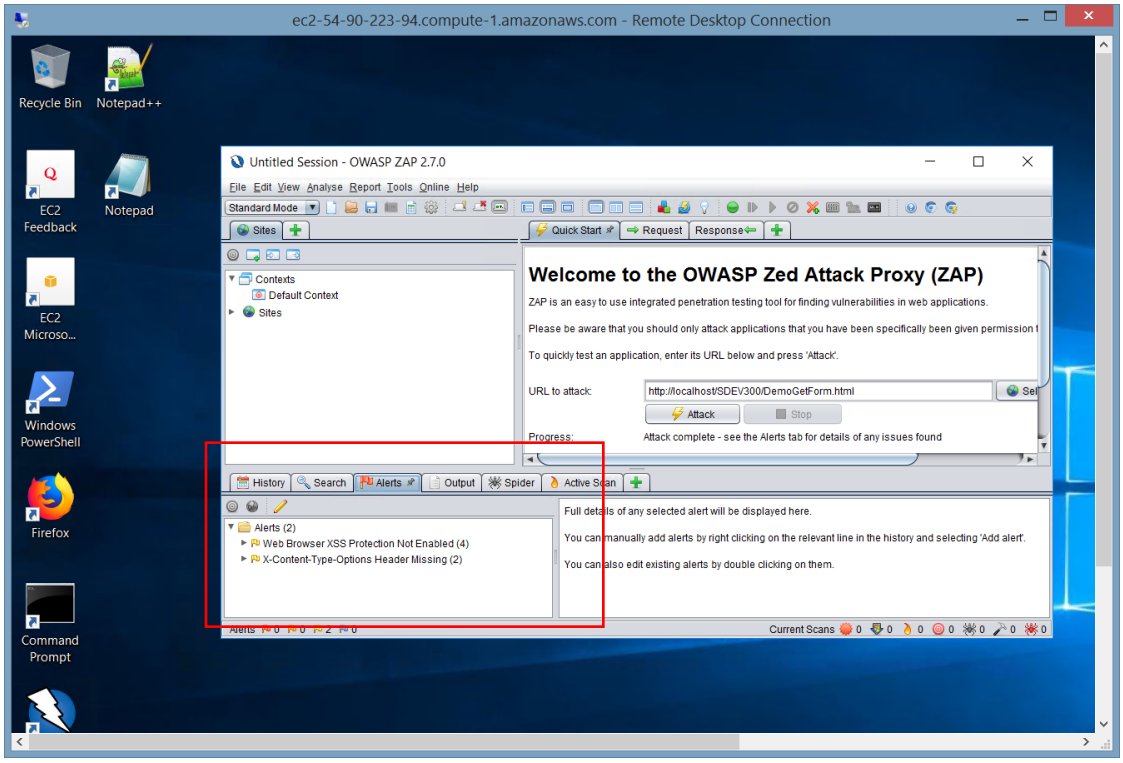

Figure 16 Alerts from ZAP attack

As shown in the Alerts tab, 2 different Low Priority Alerts were discovered in this very simple Web application including:

- Web Browser XSS protection Not Enabled
- X Content-Type Options Header Missing

The next step shows how to generate a report that provides more details about the Alerts and possible actions. Note, you can click on each alert and use the right frame of the tab to scroll down to read additional information about this alert and possible approaches to mitigate.

Keep in mind, the information provided is good start to fixing the issue; however; additional research on the Apache2 server or the programming language being used might be needed to fully fix the issue.

3. To generate a report from the scan in HTML format, open the Report menu from the top of tool bar and select Generate HTML report as shown in figure 17.

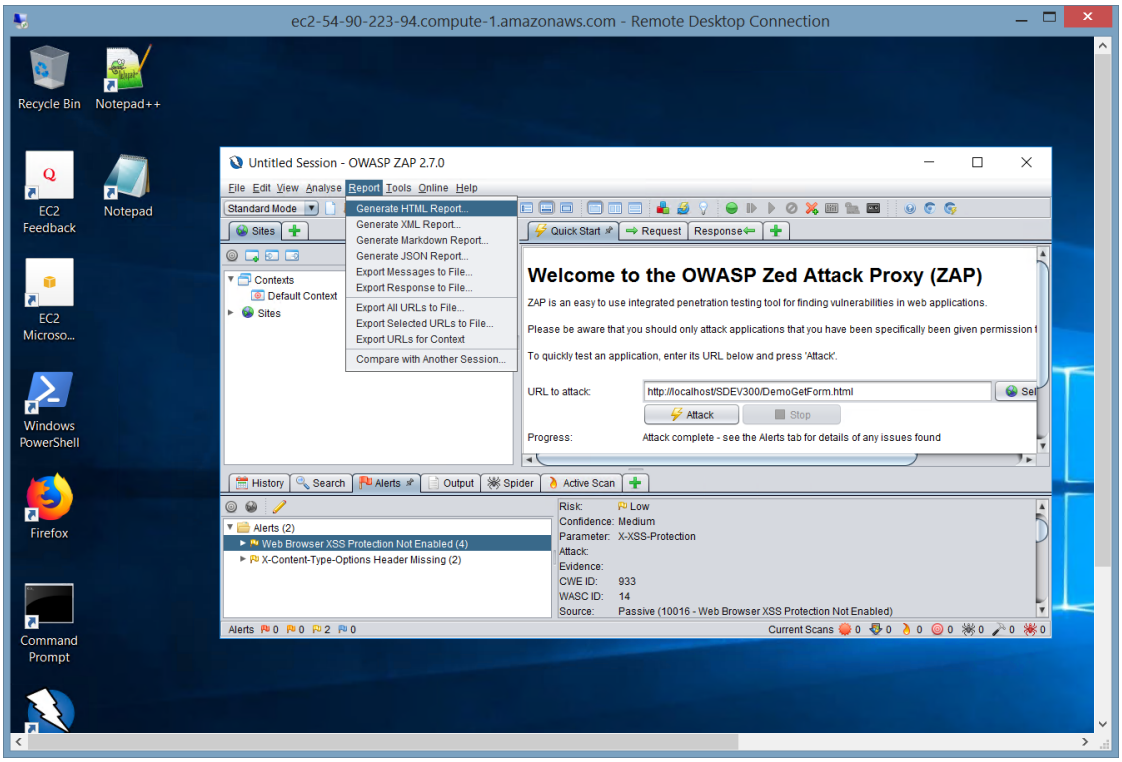

Figure 17 Generate an HTML Report from ZAP

4. Save the report to a folder and filename of your choice. For this example, the documents folder was used to save a file named DemoGetFormV1.html.

As shown in figure 18, once the file is generated, the HTML report should automatically open in your Browser. If not, use the file manager to open the report.

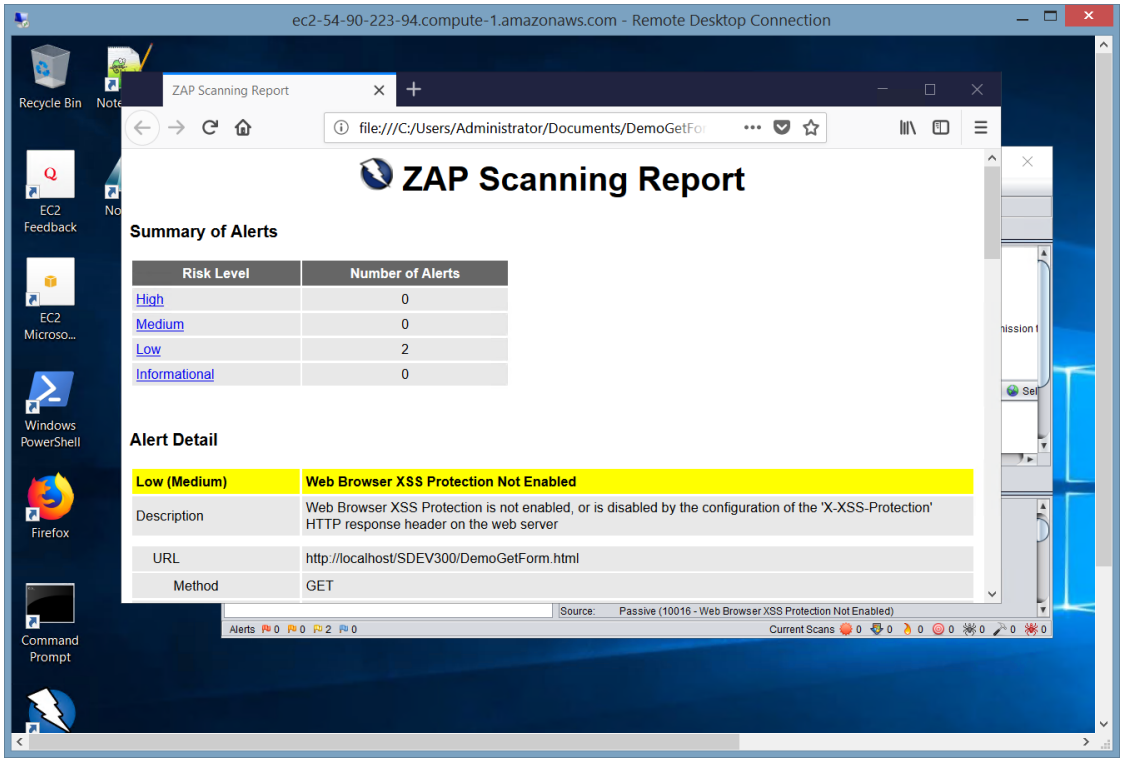

Figure 18 ZAP HTML report

As you scroll through the report, you should always fix high priority alerts first and then go down the list to fix the other lower risk items. For this course, the goal is to always eliminate all alerts. There are some subtle issues with ZAP resulting in the inability to eliminate some alerts. So leaving those can be explained.

For the first alert we see, a Low (Medium) Alert is found: “Web Browser XSS Protection Not Enabled”. Recall XSS is a top ten OWASP vulnerability that could result in someone performing Cross-Site Scripting on your Web application. The recommended solution from the ZAP tool is to “Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.” There are also references with more details on how to do that. The specific fix for Apache2 takes a bit more digging than provided in those two references ZAP provided. There are many results from google when you search for XSS-protection. I found this reference to be useful, but there are many others:

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection

Often, you have to look for and try multiple recommendations as the information you find isn’t always current, or correct.

This site recommends adding this code to the .htaccess file:

```
<IfModule mod_headers.c>
  Header set X-XSS-Protection "1; mode=block"
</IfModule>
```

Of course, on Windows, we don't have a .htaccess file, so you have to search a bit more to find out there is a httpd.conf file in the C:\Bitnami\wampstack-7.1.16-0\apache2\conf folder. So opening that file with notepad++ and adding the right before the # security line should eliminate some of those issues.

Figure 19 shows the file open in Notepad on the AWS cloud VM.

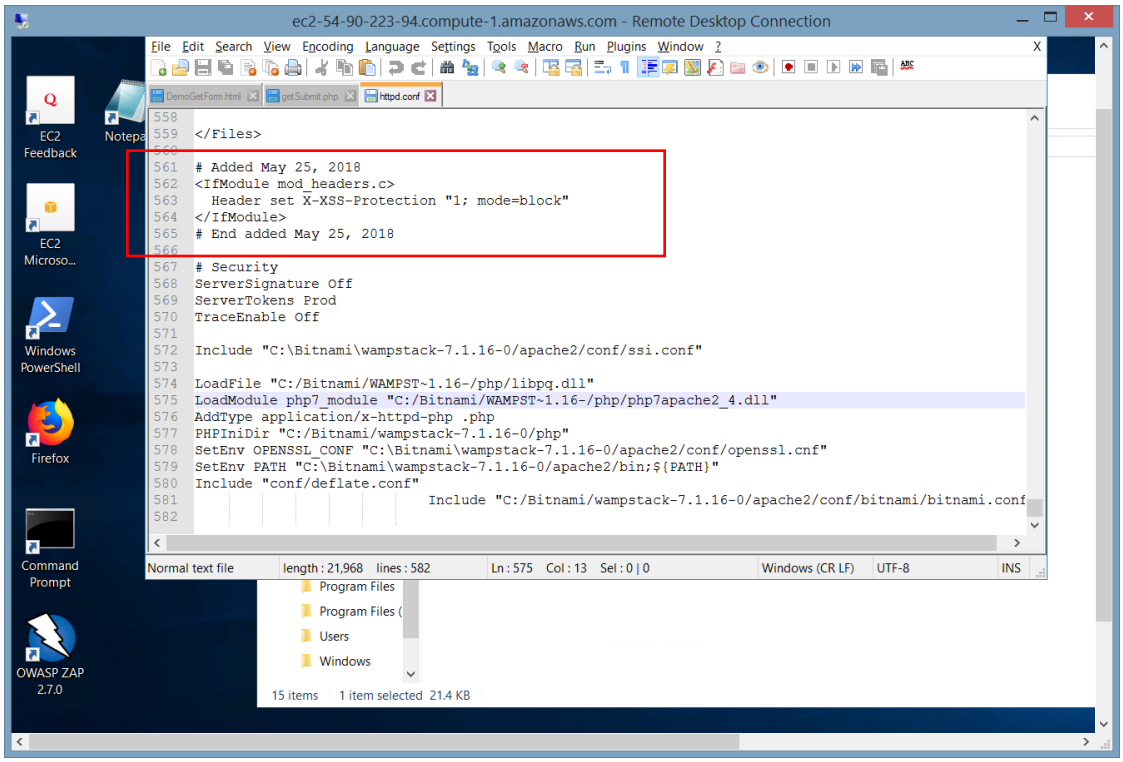

Figure 19 Editing the httpd.conf file

Important: To allow the configuration changes to the Apache server to take effect you must restart the Apache server. To do this, go to the C:\Bitnami\wampstack-7.1.16-0\apache2\bin folder and double click the ApacheMonitor. Figure 20, shows this tool in that path.

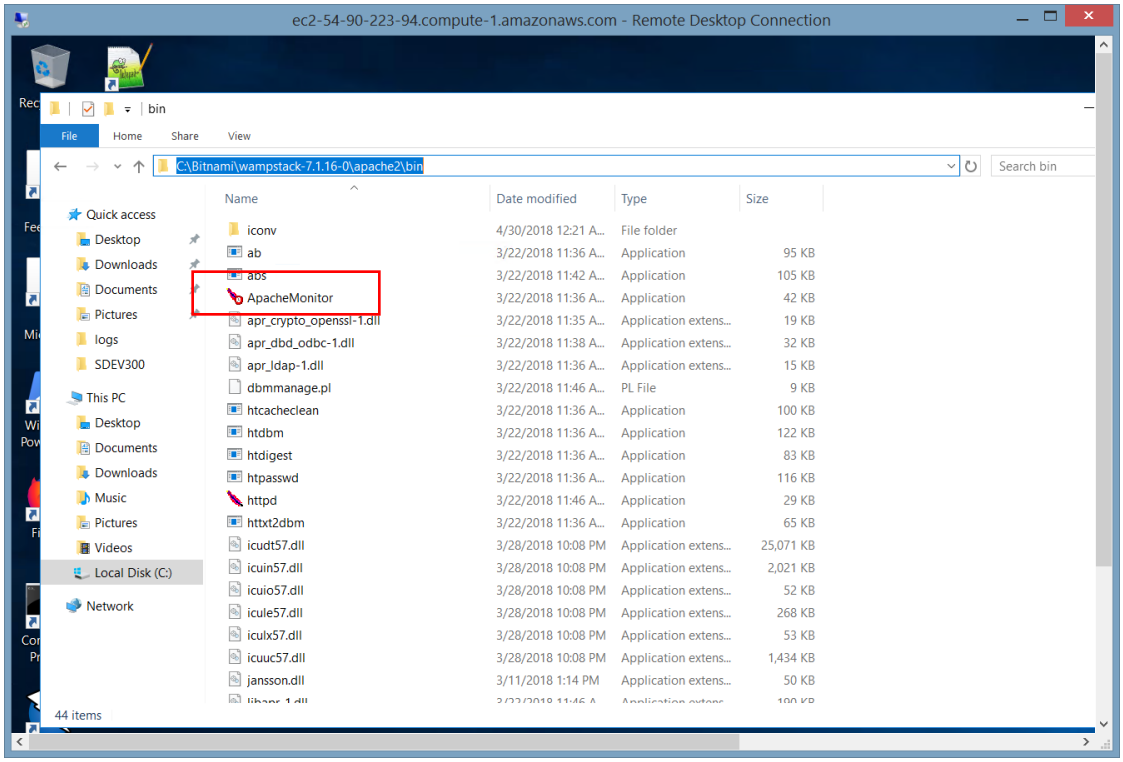

Figure 20 Starting the ApacheMonitor

Once the monitor is started, you will see it on the bottom right side of your Windows screen on the AWS cloud VM. Note you may need to click the ^ symbol to reveal the ApacheMonitor and the click that to reveal the stop/start/restart options. (See figure 21).

Make sure you have saved the changes to your httpd.conf file and then select stop, and then about a minute later select start from the ApacheMonitor. This will allow the new configuration to take hold.

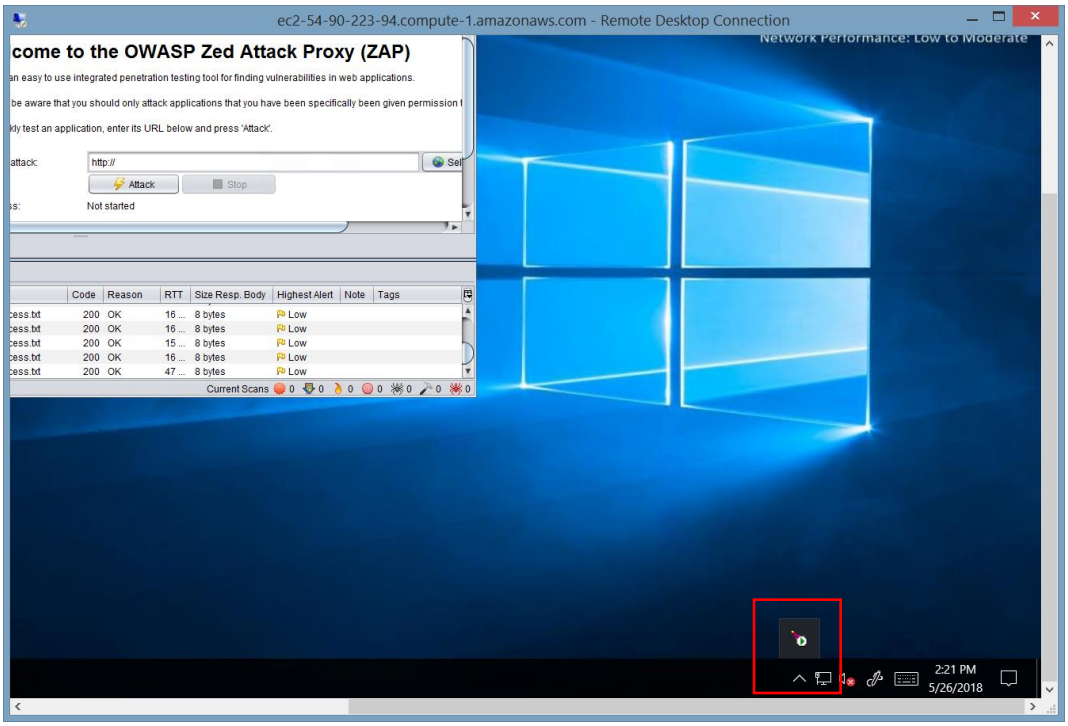

Figure 21 Stopping the Apache Server

When you rerun the attack you should see that two of the application specific XSS alerts went away. As shown in figure 22, the only remaining XSS protection issues are for the robots.txt and sitemap.xml file.

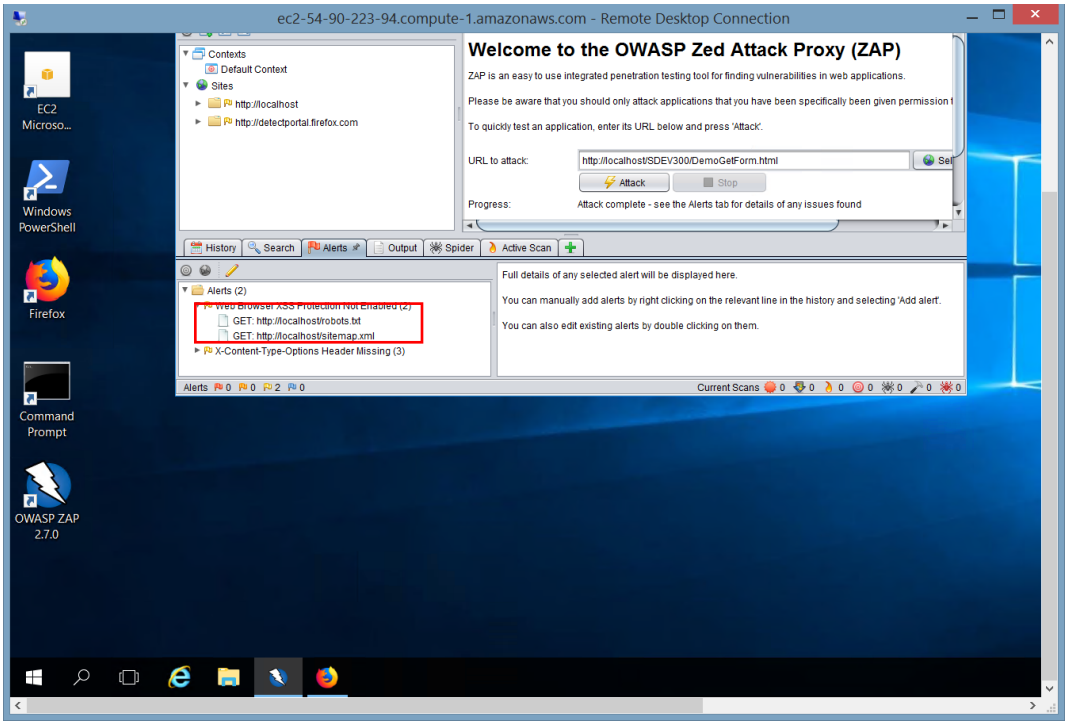

Figure 22 Rerunning the Attack

Interestingly, ZAP will always provide these warnings even though for this specific localhost site there is no robot.txt or sitemap.xml files. Both of these files, are standard for most commercial web sites and are used by spider crawlers to look før information in your site. Typically, this is for search engine optimization and marketing. However; certain information in the files may provide information about the site and potential vulnerabilities. So, this low warning will always be listed, even when the files are present. For the purposes of this course, you can ignore those warnings.

For the next alert, the issue is “X-Content-Type-Options Header Missing”. The ZAP tool describes this issue as that may result in an attempt to prevent MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as the wrong content type. This problem is often associated with older browser versions, which many hackers keep running to exploit to their benefit.

The recommend solution ZAP provides is to set the X-Content-Type-Options header to 'nosniff' for all web pages. Some additional references are provided that help performing this operation. These references are marginal in terms of how to fix the issue, so some additional research provided the following line of code that needs to be added to the httpd.conf file in the same location as before:

```
Header set X-Content-Type-Options nosniff
```

Modify the httpd.conf file to include this code:

```
<IfModule mod_headers.c>
  Header set X-XSS-Protection "1; mode=block"
  Header set X-Content-Type-Options nosniff
</IfModule>
```

Then, save the httpd.conf and be sure to stop and start the Apache server. Then rerun the attack and you will see only the robots.txt and sitemap.xml issues remain. (See figure 23)

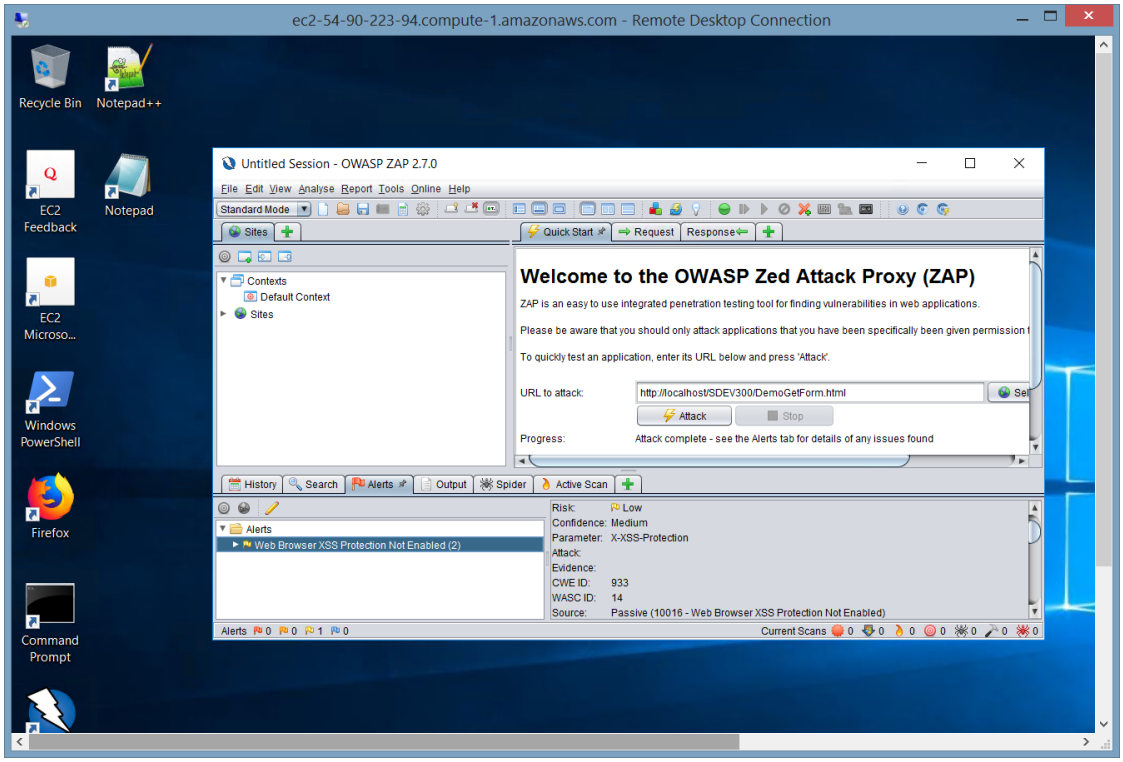

Figure 23 Rerunning ZAP to fix X-Content type

A couple of other important points about ZAP.

- It is open source, free software. It isn't perfect
- Sometimes you might need to stop ZAP and restart it to see the resulting effect of your changes. So be sure you do try that if for example, you make a change, have restarted the Apache server and still see the vulnerability
- You can also use the File->New session option to clear the current session

5. For practice purposes, it is recommended to run a similar scan, analysis and mitigation of the DemoPostForm.html application. Interestingly, you should see many of the vulnerabilities that would have appeared in the Post version are not present because of the global changes you made in the httpd.conf file.

The analysis and mitigation of issues is a repetitive process that should be done often in development and after each release to make sure issues are not introduced during updates. ZAP is just one tool for use in this process. Be aware that ZAP is not the only approach for finding software vulnerabilities nor will it eliminate or find all of the issues. Other scanners and techniques should be used to help secure your web application.

Lab submission details:

For this lab, you will provide a detailed analysis using both manual interception techniques and automatic scanner attacks on a Web application provided by your instructor and attached to this assignment. You will need to download the files to your desktop and then move the application to the AWS Cloud VM and place in an appropriate htdocs folder on the VM. You should also analyze and run the web application as is so you know what it does from a functional standpoint. **The AWS Cloud VM must be used for this lab.**

You should run the manual interception techniques first, and describe in detail the information revealed to you during your analysis. Be sure to provide screen captures of you running the ZAP tool and analyze all files in the application. Try to modify the http messages and look for possible vulnerabilities. This is the important discovery portion of your analysis.

Next, when you run the automatic scan, be sure to generate an HTML report showing all alerts. Also, describe the active scan activity. For each alert, discuss all of the output and try possible solutions. Be sure to describe how you prioritized alert messages. Resolve all alerts and document specifically your process in resolving those alerts. Looking up these errors and performing research and how to resolve them is a part of the exercise and is a good representation of how this is accomplished in the real world. Rerun the scanner after you have fixed the issues.

For your deliverables, you should submit a zip file containing your word document (or PDF file) with screen shots with detailed descriptions of your scans. Be sure to include the descriptions and analysis of your results, your prioritization and approach to mitigating the issues. Also, include the reports from your scan. You should also include any configuration or other files you modified clearly describing what was changed and why you changed it. Also, **the log access file must be included in all submissions to be accepted.**

Your report should be well-organized and clearly written. This report is aimed at your Chief Security officer who pays your salary. He/she is a technical geek and wants details, clarity and something they can pass on to others.

Include your full name, class number and section, instructor name, and date in the document.
Hints:

- Be sure to look at the code visually as well as just running the ZAP tools
- Delve deep into the OWASP Top 10 vulnerabilities
- Use the application as it exists to understand the intended functionality and think about ways a hacker could take advantage of that functionality

Grading Rubric:

Attribute	**Meets**
ZAP analysis	**60 points** Runs the manual interception techniques, and describes in detail the information revealed to you during your analysis. (10 points) Analyzes all files used for the application. (10 points)

	Modifies the http messages and looks for possible vulnerabilities. (10 points) Runs the automatic scan, and generates an HTML report showing all alerts. (10 points) Discusses all of the output and provides solutions for all alerts. (10 points) Reruns the scanner after you have fixed the issues to demonstrate the success. (10 points) **Does not include access.log (-100)** **Does not use the SDEV AMI. (-100)**
Documentation and submission	**40 points** Submits a zip file containing your word document (or PDF file) with detailed descriptions of screen captures of your scans. (5 points) Includes all files, file changes and description as to the why the file was changed. (5 points) Includes the descriptions and analysis of your results, your prioritization and approach to mitigating the issues. (20 points) Includes the reports from your scan. (5 points) Your written report is well-organized and clearly written and includes references in APA style. (5 points)