2/14/2016

|  |
| --- |
| In Depth Penetration Testing and Risk Mitigation Using OWASP Zed Attack Proxy (ZAP) | By Stephen Gubenia |



|  |  |
| --- | --- |
| University of Maryland University College | WEB APPLICATION VULNERABILITY ANALYSIS OF ZENWIRELESS eSTORE |

Contents

[Introduction 1](#_Toc442450407)

[Overview 1](#_Toc442450408)

[Lab 5 Submission Requirements 1](#_Toc442450409)

[Sustains and Improvements 10](#_Toc442450410)

[Conclusion 11](#_Toc442450411)

# Executive Summary

During the month of February 2016 extensive penetration testing was conducting on the ZenWireless eCommerce website using the OWASP Zed Attack Proxy Tool (ZAP) in order to find the vulnerabilities of the web application. Both manual and automated scans of the web application were performed using the ZAP tool and as a result several vulnerabilities were found ranging from low to high risk. Once the vulnerabilities were found they were classified and prioritized for mitigation as either immediate or delayed. Vulnerabilities that were considered high risk were prioritized as immediate and mitigation steps began promptly with vulnerabilities of moderate risk scheduled to follow second, and vulnerabilities of low risk would be mitigated last.

# Technology Used

# 

The penetration testing of the ZenWireLess ecommerce web application was conducted inside an Oracle VM VirtualBox using a distribution of the Ubuntu operating system (OS). The web application was mounted and run on an Apache2 web server. The web application was written in the PHP5 language and all testing was done through the Mozilla Firefox browser which was configured to use the ZAP tool as a proxy.

# Manual Testing Results

The ZAP tool was started and a Firefox browser window was opened to begin the testing session. Figure 1 below is the screenshot of the ZAP interface prior to the start of testing. No alerts, sites, or output messages of been populated as of yet.

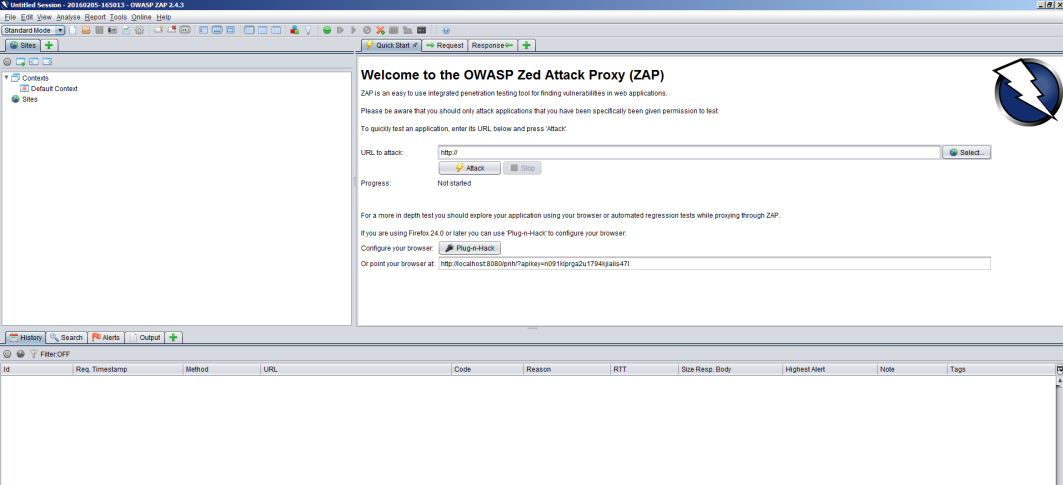


Figure 1 Screenshot of ZenWIreless Login page

To begin testing the browser was pointed toward the ZenWireLess login page. Figure 2 below shows that the web application was working normally as expecting using the ZAP tool as a proxy.

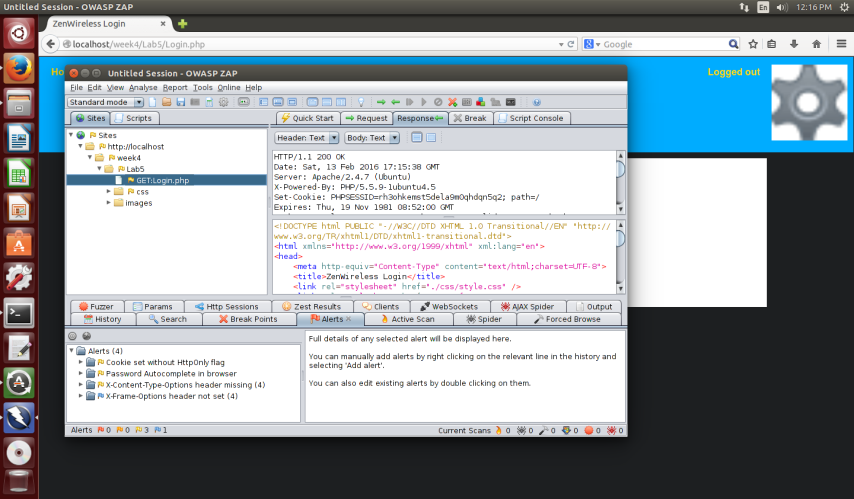


Figure 2 Screenshot of ZAP interface on start up

Next the web form was completed with login information and the submit button was clicked. Figure 4 below shows the response output information obtained by the ZAP tool from the manual scan.

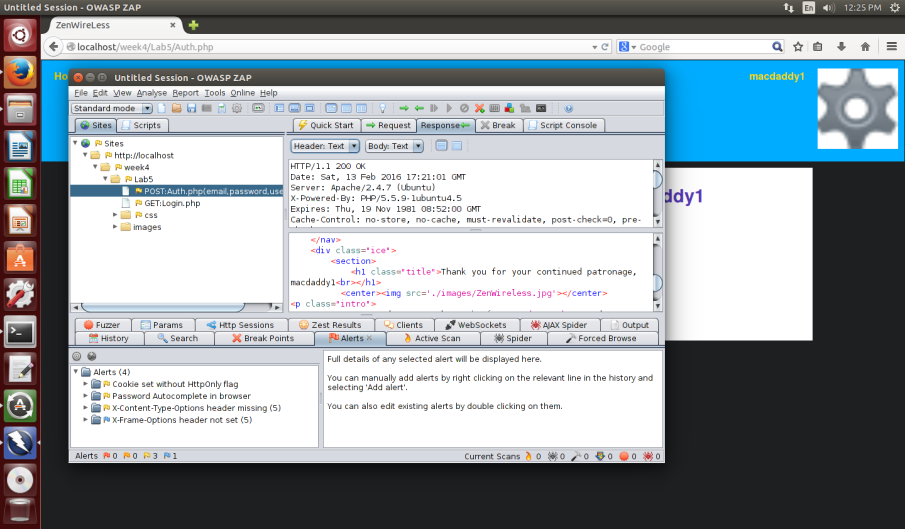


Figure 3 Screenshot of response information from submit to Auth.php

The manual scan then continued by clicking the settings button and choosing the logout option. Figure 5 below shows the results in the ZAP interface from the logout.

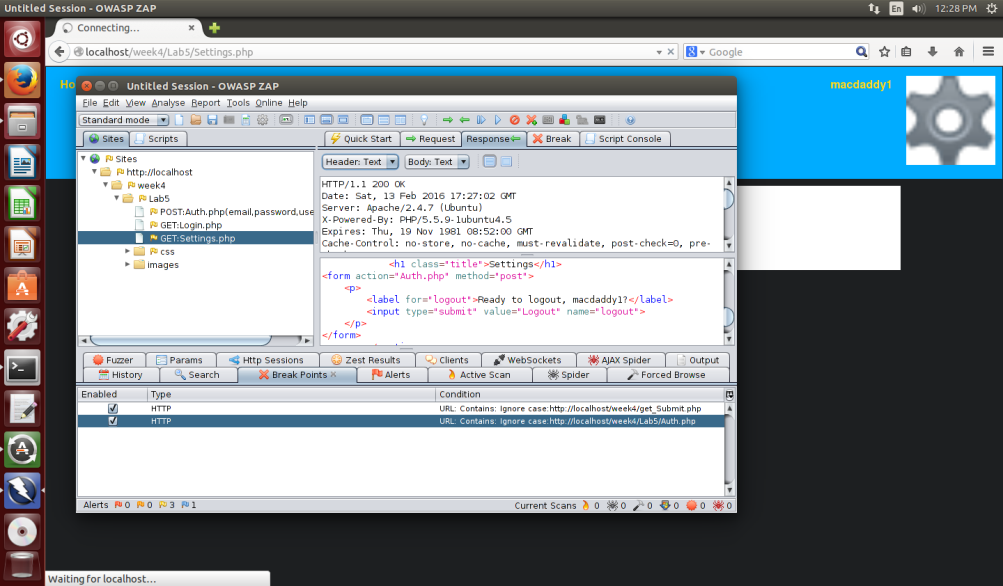


Figure Screenshot of response information from submit to logout.php

Next a breakpoint was set on the authorization page to intercept the request and response and see if new information could be passed in the parameters. Figure 5 below shows the information obtained in the ZAP tool from using the breakpoint.

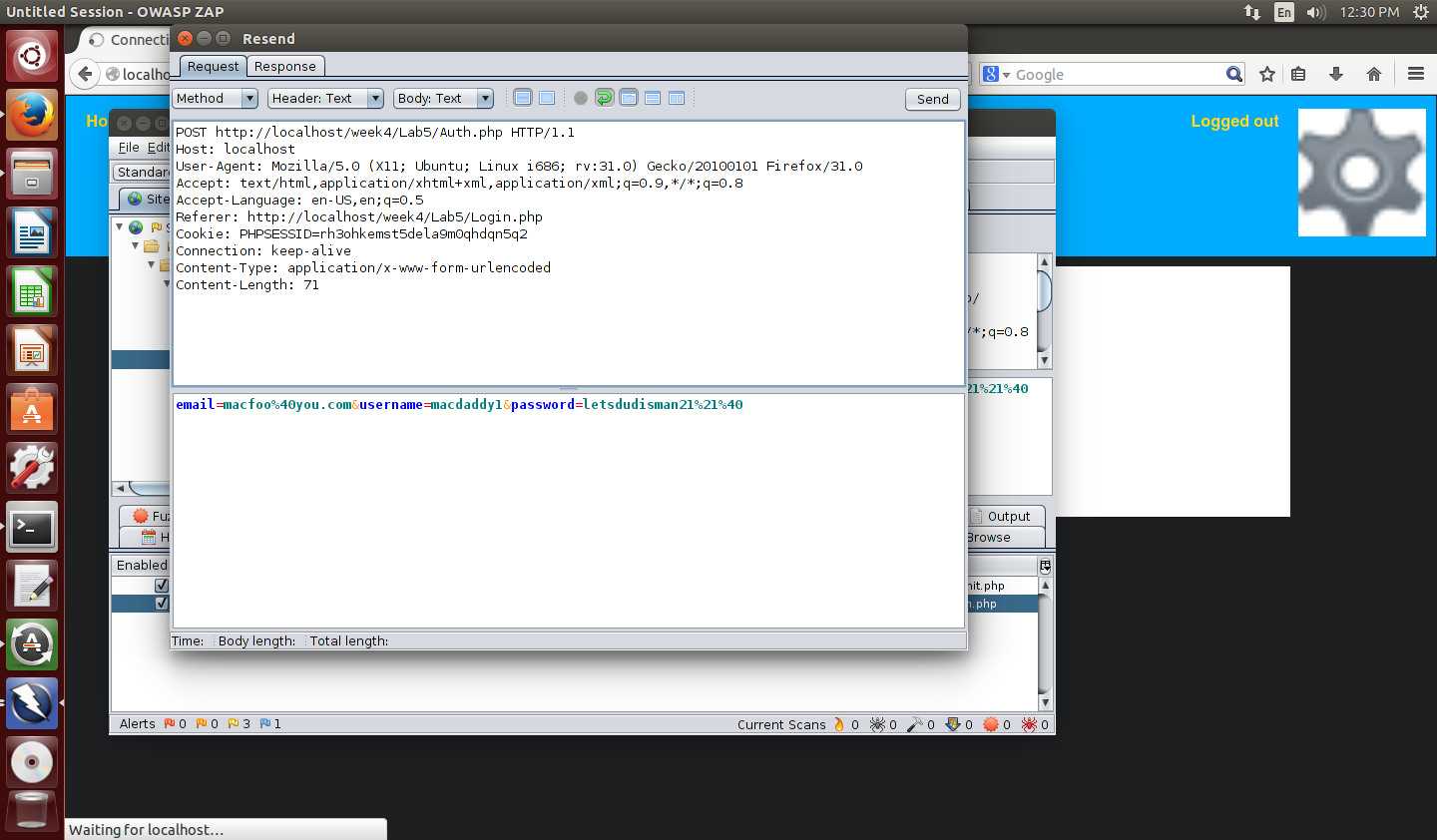


Figure Screenshot of the request information from using breakpoint on Auth.php

New user information was entered into the parameters and sent. Figure 6 below shows the new response in the ZAP tool.

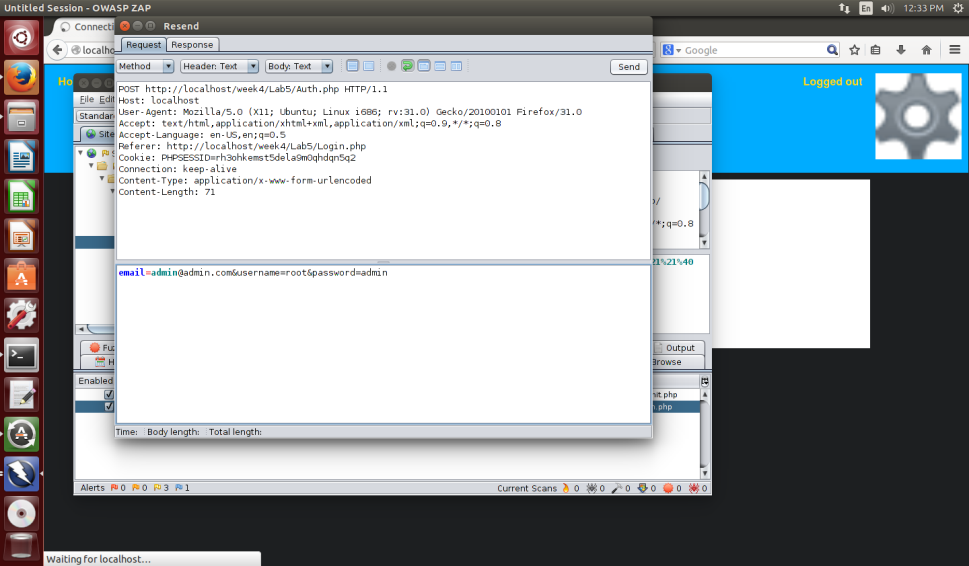


Figure Screenshot of the new response with user info changed after breakpoint

Using breakpoints in ZAP it is also possible to change the response parameters and send the response directly to the browser instead of viewing them in the tool. Figure 7 below shows a rerun of the same test with sending the changed user information directly to the browser.

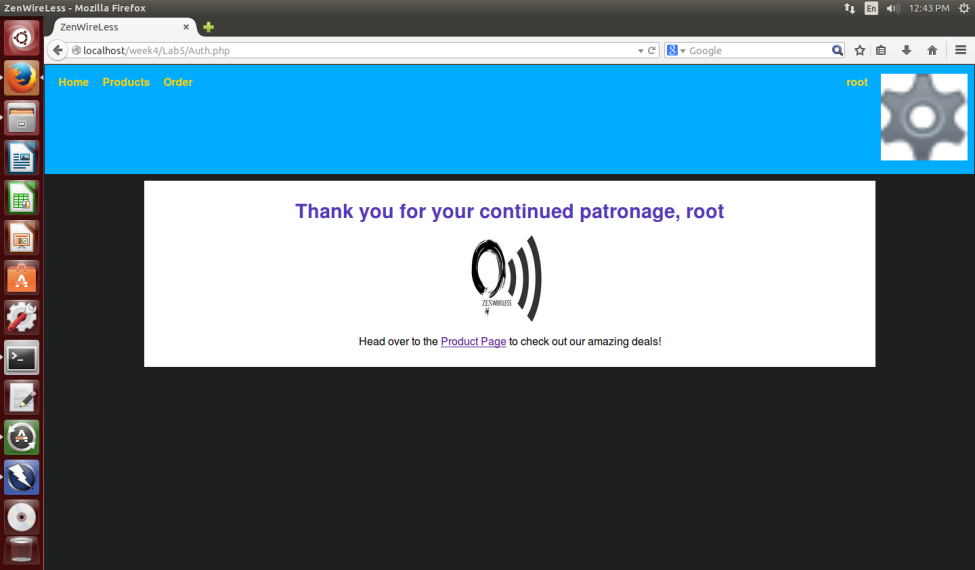


Figure Screenshot of sending breakpoint changes directly to browser

The manual scan did not reveal any significant risks. The main issues were browser related such as password autocomplete functions being enabled which are user related functions and pose a low security risk to the web application. The other issues where content type options header missing on a few pages which would allow MIME sniffing by some older browser versions. There were also a few incidents of cookies being set without the HttpOnly flag. These were the only vulnerabilities found by a quick manual scan being performed. Next a more detailed active scan was performed using the ZAP tool.

# Active Scan Results

Next the web application was directed back to the login page and the tool was the login URL was entered into the attack parameter on the quick start tab of the ZAP tool. The ZAP tool then performed a comprehensive scan of the ZenWireLess ecommerce site. Figure 8 shows the ZAP tool while the scan is in progress.

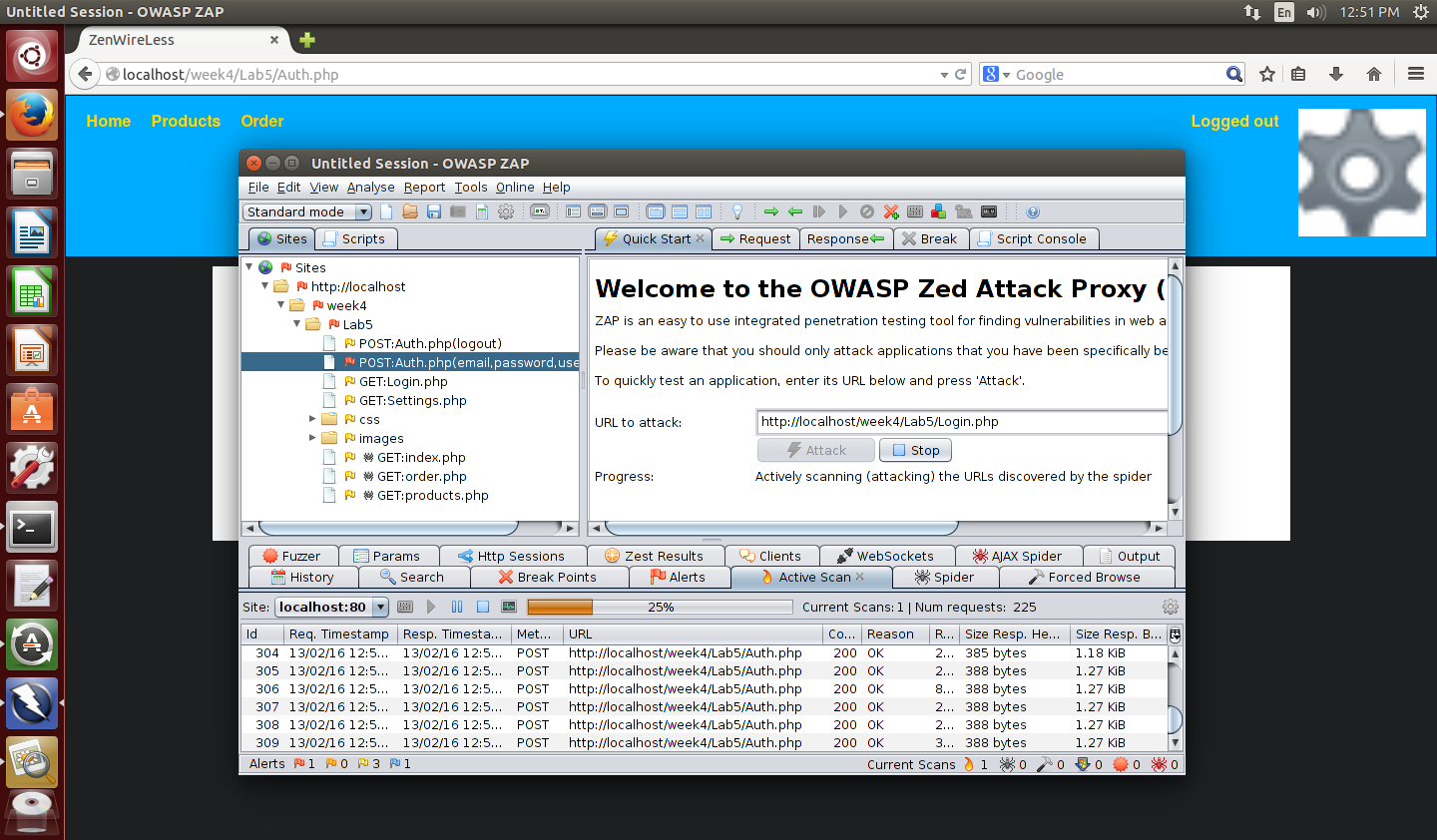


Figure Screenshot of active scan being performed

After the scan completed the alerts tab was reviewed to see what the active scan had revealed. As expected all the same vulnerabilities from the manual scan had been found as well as one new threat. One cross site scripting (XSS) vulnerability had been found on the username parameter which is passed from the login page to the authorization page. Figure 9 below shows the alerts that were found during the active scan using ZAP.

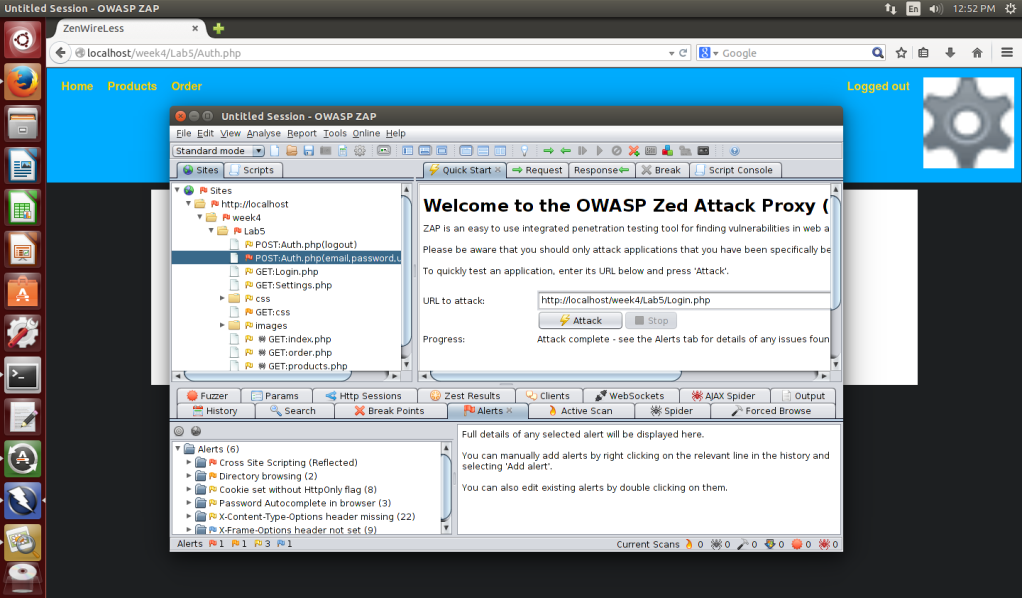


Figure Screenshot of the alerts found during active scan

Once this scan was completed a ZAP was used to generate a report of the risks to get a more detailed analysis and suggestions of possible solutions. That reports is included here.



From this report the risks were categorized as either immediate or delayed and prioritized for mitigation.

# Mitigation Strategy

From the report the number one risk was determined to by the XSS threat. Possible solutions were to set session cookies to HttpOnly, ensure that all web application files specified encoding, and to use input validation strategies such as “accept known good.” The first step taken was to ensure all web application files specified character encoding. One file was found to not specify any encoding. This was rectified and the file now specifies UTF-8 encoding. Code was then added to the Auth.php file to set the cookie parameters to HttpOnly. Finally there was an attempt made to add a function for data sanitization and validation for the username parameter. It was believed that mitigating this vulnerability would also take care of most if not all of the cookies set without HttpOnly flags.

# Mitigation Results

During the process of mitigation the web application had to be tested. Much of the new code proved to cause many issues with the web application, particularly the authentication process. Ultimately the function for data sanitization and validation had to be removed. Once the web application was in working order again a new active scan was performed. Figure 10 below shows the results of the new scan in the ZAP tool.

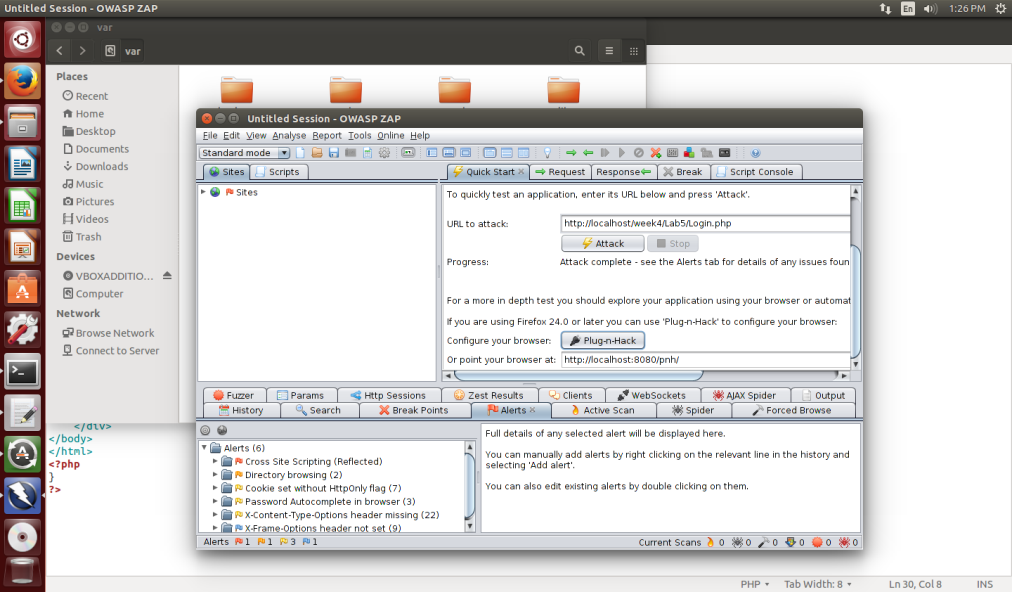


Figure Screenshot of active scan after risk mitigation

Ultimately none of the risks were removed from the initial round of mitigation. A new report was ran using the ZAP tool. That report is included below.



Mitigation continued for the duration of the project time line but no major results were achieved. Further investigation was required.

# Vulnerability Solutions

When results were not being achieved further investigation was conducted. The simplest way to resolve the cookies set without HttpOnly flags would be to set this value in the php.ini file. This was attempted but I do not have write permission on the file so it could not be done. This would possibly solve the XSS issue as well, but I cannot be certain as there are three suggested solutions in the report one of which was specifying character encoding which has been handled. It is possible that data sanitization and validation will still need to be conducted but it will require more man hours and a greater budget to rectify the source code.

The simple solution at this time will be to have the system administrator access the php.ini file and set the session.cookie\_httponly variable equal to 1. This should rectify all of the HttpOnly flags and may possibly solve the XSS issue. Once the php.ini file has been modified penetration testing can be conducted again but will require a new project team.

# Conclusion

Overall the ZenWireless ecommerce web application is fairly secure. It only had one high risk vulnerability and a few moderate risk vulnerabilities. A lot of the information generated by the active scan was low risk browser related issues or purely informational. Valiant attempts were made to mitigate all issues on the project time line. The source code was modified and improved, but at this time the vulnerabilities still remain. Further mitigation and testing will be required but it is beyond my current level of system permissions and capabilities.