Vulnerability Assessment Assignment

Jacob Krupa

ITMS 543

10/16/18

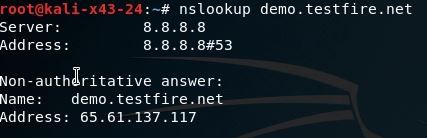
htttp://demo.testfire.net/ Vulnerability Assessment

Executive Summary

There are many high-risk and medium-risk vulnerabilities in the website and many need to be addressed ASAP in order to secure the site and allow for safe use. All these risks put together put the website in a very bad place and make it very vulnerable to attacks. The website is very vulnerable to network eavesdropping and password stealing, but also very vulnerable to other software and coding vulnerabilities. On the bright side, many of these vulnerabilities will not take a lot of resources and time to fix. Many of these fixes are small code adjustments in the code/software that will make the website a lot more secure in the long run and the rest of the fixes require software updates to newer versions. Some risk will require a little more work from the developer of the website, however they are not sometime that should take a very long time to fix. Overall, while the site is very vulnerable right now, it can be reduced to relatively safe levels. Below, I have listed my steps to test the website and the vulnerabilities associated with that website. There are also steps to mitigate these vulnerabilities.

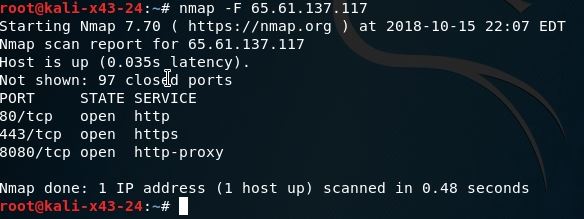
Vulnerability Assessment Steps Taken

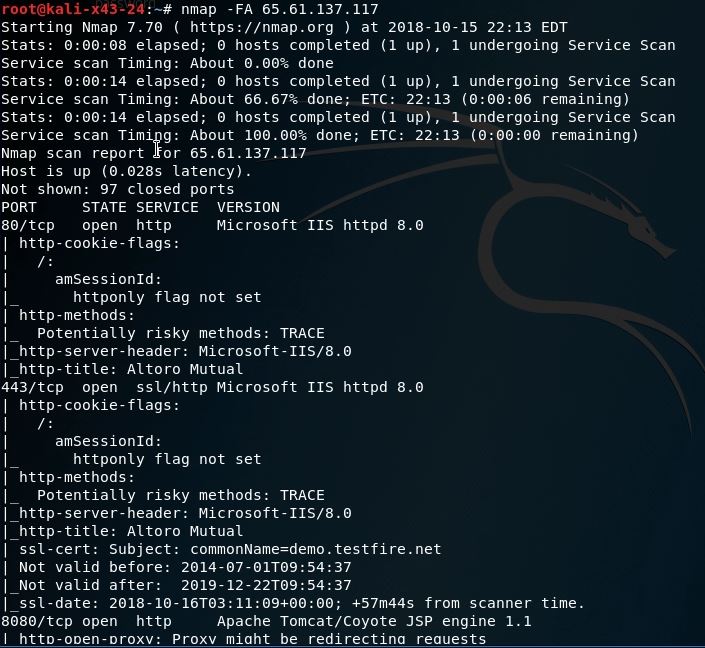
Nslookup

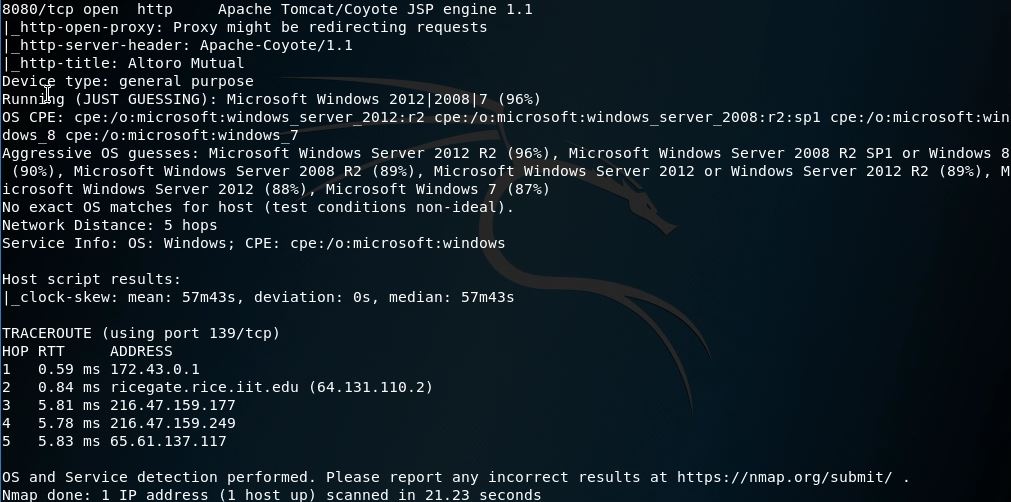


Here, I used the nslookup command to allow me to find the transplantation the original URL of htttp://demo.testfire.net/ into an IP address. This allows me to find out the IP address, which is required in order to use further vulnerability assessment tools. Nslookup, however, was not able to find any associated DNS or mail servers associated with the website, which means less machines that can be broken into and exploited.

Nmap

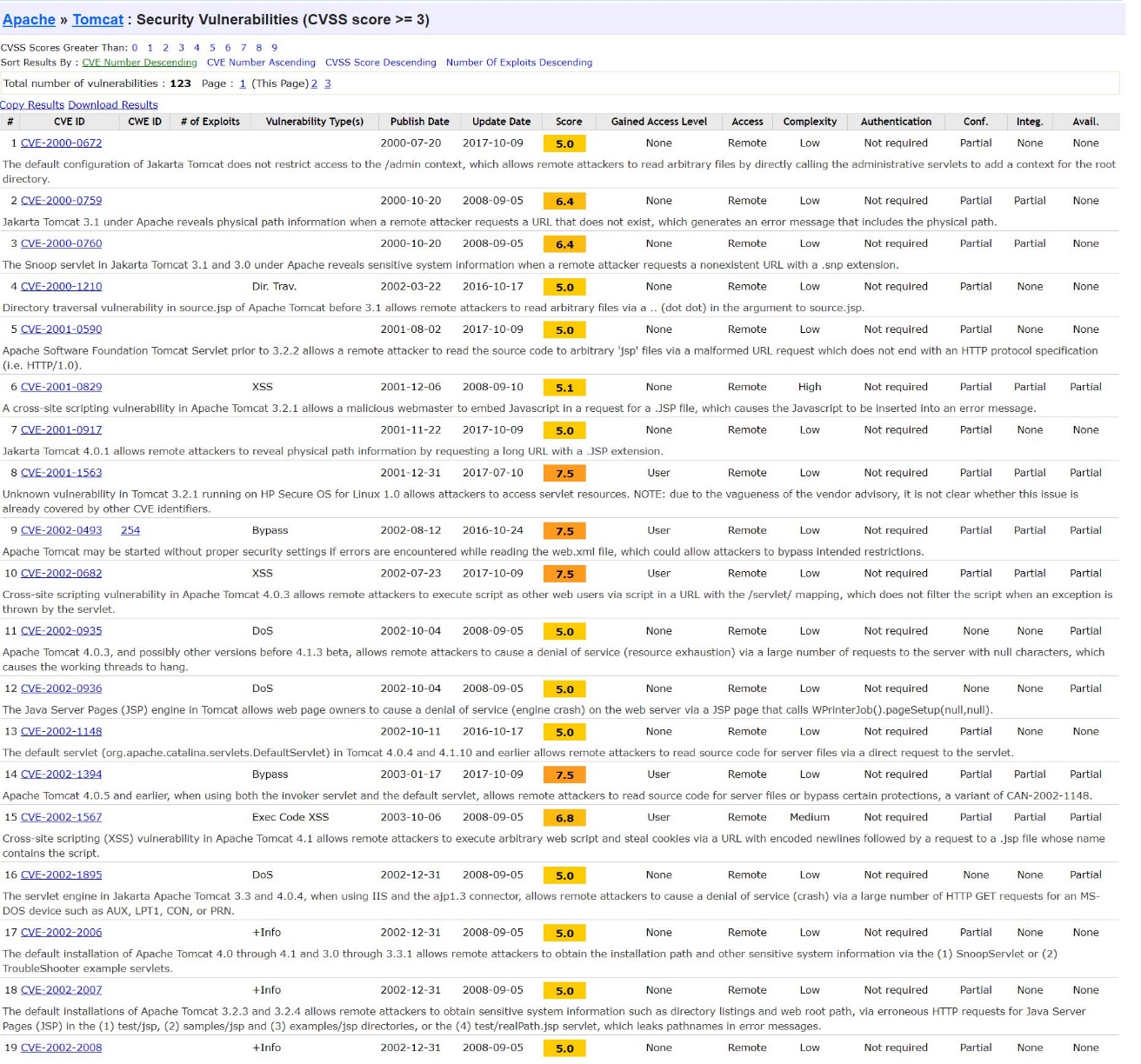


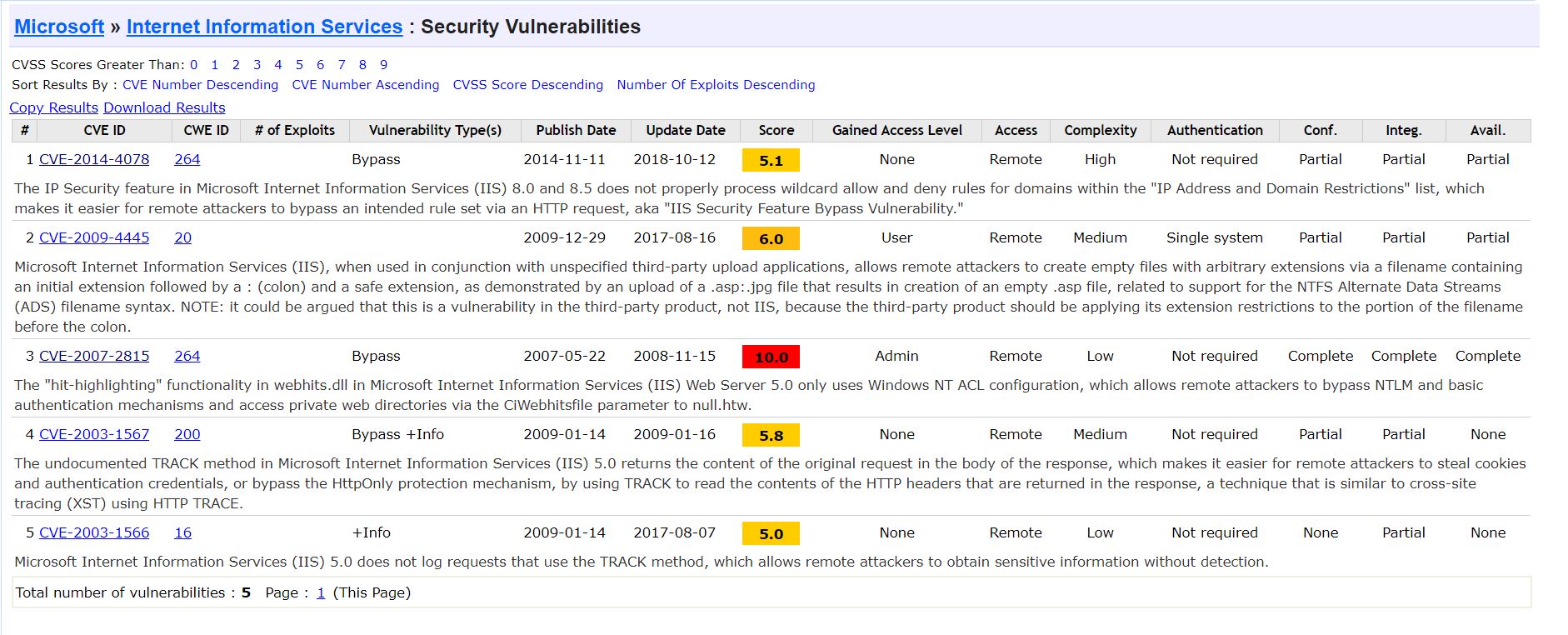




Here, I used the nmap command along with the website’s IP address to find out what ports are open and what services they are running on those ports. Demo.testfire.net is using 3 open ports, which are 80, 443, and 8080. Those ports are running http, https, and http-proxy, respectively. However, I ran a more in-depth nmap command and found the software and software versions that each service was running on the open ports. Http was running Microsoft IIS httpd 8.0, https was also using Microsoft IIS httpd 8.0, and http-proxy was running Apache Tomcat/Coyote JSP engine 1.1.

CVE Details





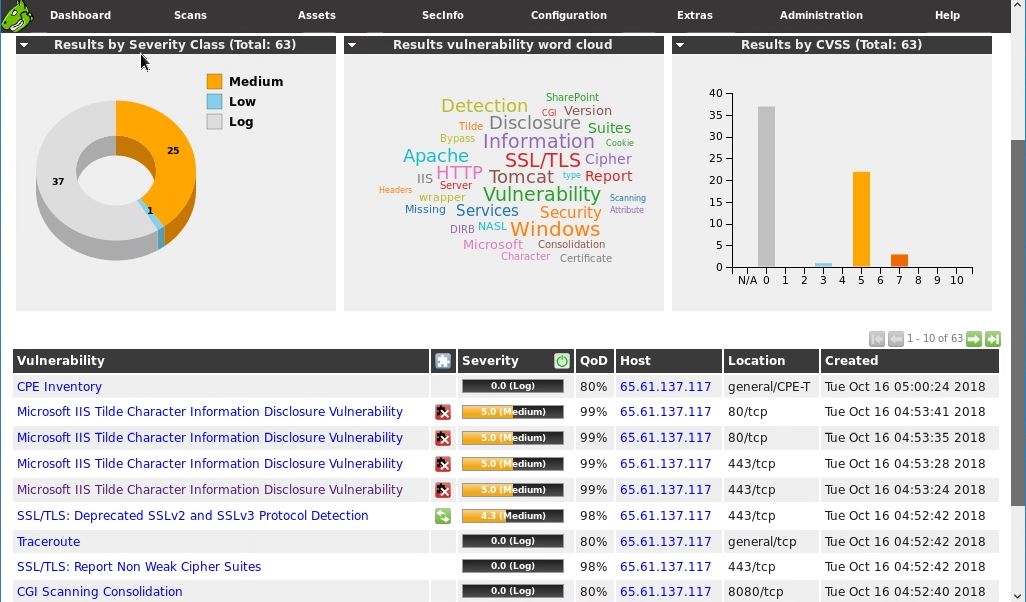
Using information I’ve found using nmap about the software being used on the website, I looked through a site called CVE Details. This site brings up known vulnerabilities on a lot of software. This can give a lot of information on how a website could be exploited. However, the software the website is running on their ports do not have any major vulnerabilities in their version. So, there are no major exploits on the services side, if they are implemented properly.

Vega



I used a vulnerability analysis tool called Vega. This tool scans websites and looks at the programming and coding vulnerabilities located on a website. I used this tool on demo.testfire.net and came up with a ton of high risk vulnerabilities and some medium-low risk vulnerabilities as well. This website has a lot of programming issues that leave it very vulnerable to attacks through its programming. These risks will have to be addressed in order for this website to reduce vulnerability to attacks.

Openvas



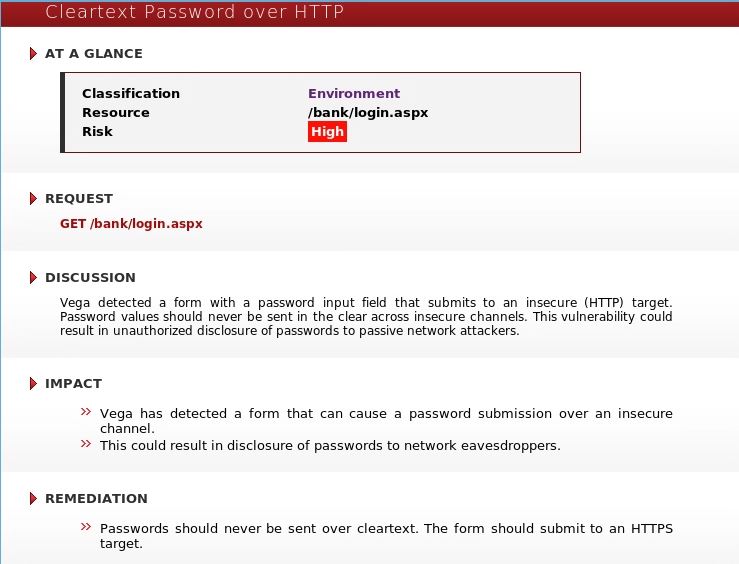
Openvas is a tool that scans and locates vulnerabilities in systems. It gives a lot of information about your system’s vulnerabilities as a whole. Here, using openvas, I have located about 25 medium risk and 1 low risk vulnerabilities regarding the vulnerabilities of the site as a whole. The average CVSS score of the website was 6.8 out of 10, with 10 being critical and 0 being no vulnerabilities.

Vulnerabilities and Mitigation

There were many high and medium risk vulnerabilities associated with demo.testfire.net. These issues need to be addressed and mitigated in order to secure the site for safe use. I will go through the vulnerabilities from issues that need to be addressed ASAP because of their criticality to issues that are less severe. I will also go in order of the easiness of the mitigation of these risk from easy to fix to hard to fix.

The most pressing issues were related to the coding/programming of the website. These issues need to be addressed as they are very vulnerable.

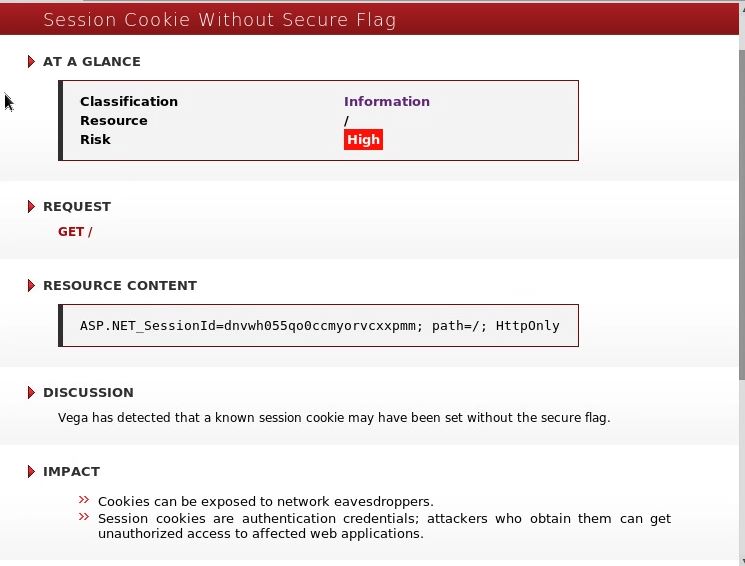
High Risk - Clear Text Password over HTTP



One of the largest vulnerabilities is that your website is sending users’ passwords over http. When passwords are sent over http, they are not encrypted are just shown in plain/clear text, which makes them very easy to listen for, find, and steal. Every password that belongs to any of your users can be stolen very easily if someone is listening to the network traffic.

However, this is a very easy fix. Since your website is equipped to use https on port 440, all that is needed is a slight change to the website’s html code to use https instead of http. This is a very minimal change and https will automatically encrypt the passwords, which makes them significantly harder to steal and use.

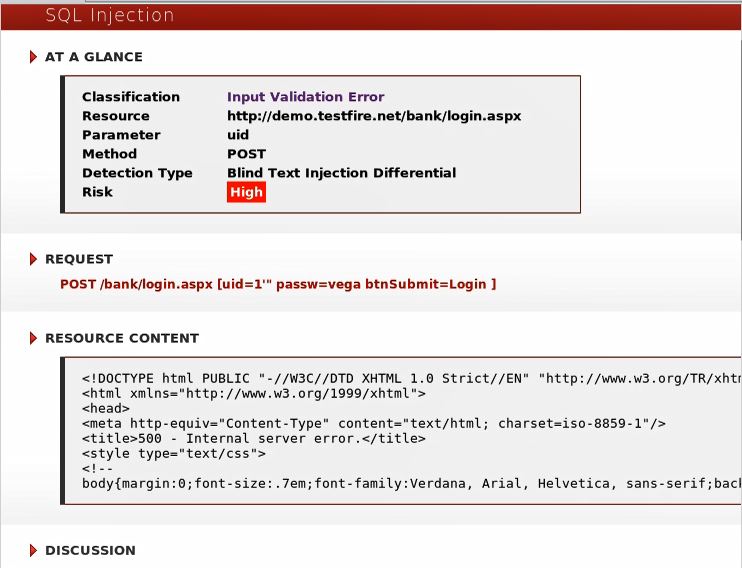
High Risk – Session Cookie Without Secure Flag

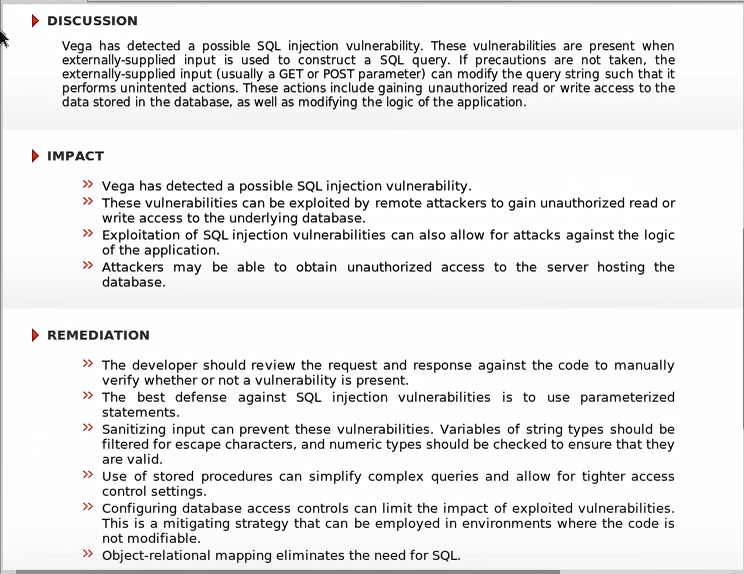


Another high-risk vulnerability is 1 or more session cookies being used the site are not secured with a secure flag. A session cookie is an authentication credential, which means an attacker that obtains them could get unauthorized access to several parts of the website and web applications. These session cookies can be exposed to network eavesdroppers very easily.

Fortunately, this is a relatively easy fix as all it requires you to do is to modify the cookies in your code and set the secure flag to true. Slight modification of the cookie code will should fix this issue.

High Risk – SQL Injection/SQL Error

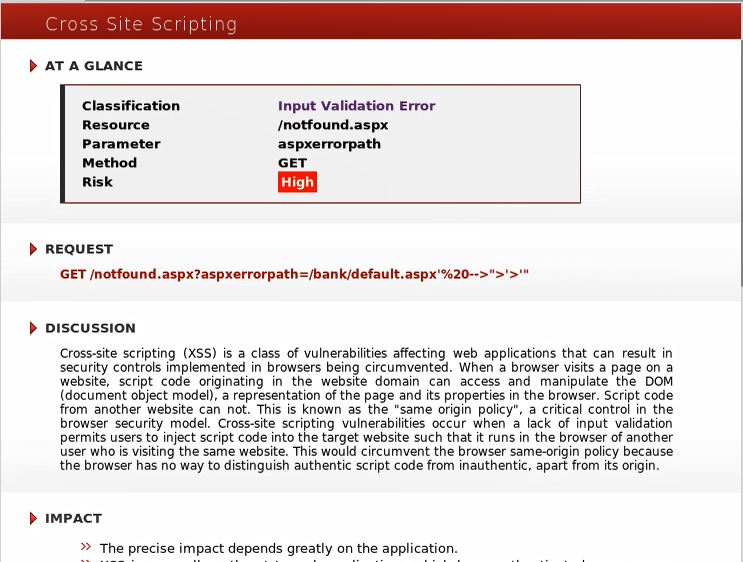


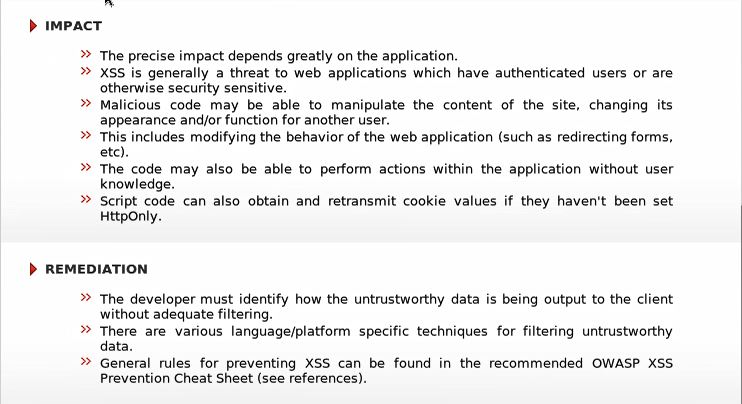


It has been determined that in several places in the programming of the website where the database can be accessed through SQL, that an attacker could use these SQL interactions to gain unauthorized read or write access to the database. They can use SQL injection and be able to gain access to the database.

However, the fix is also relatively easy. It requires minor changes in the code where inputs from users are filtered and sanitized for escape characters and things like that. The best way to defend against this is to replace the normal SQL statements in the code with parameterized statements. These statements will only allow for slight changes of variables in a SQL statement and are used for frequent trips to a database. This will allow you to secure your database from SQL injection as only variables can be changed in the SQL code.

High Risk – Cross Site Scripting

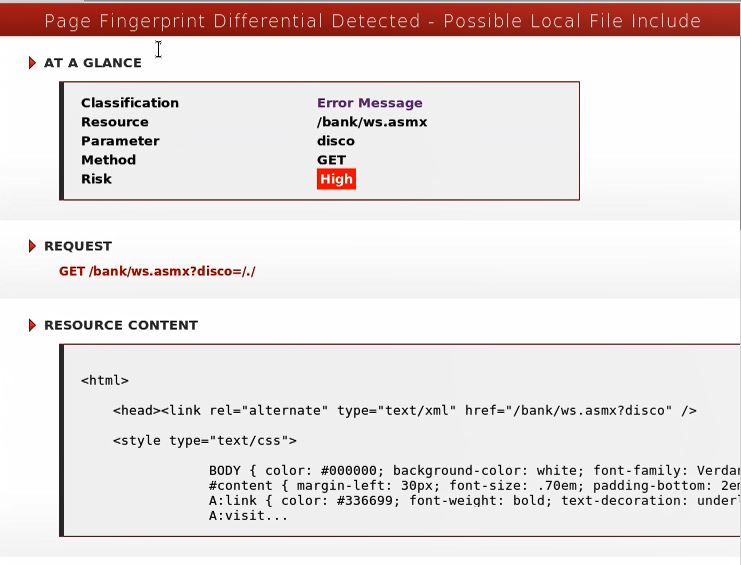


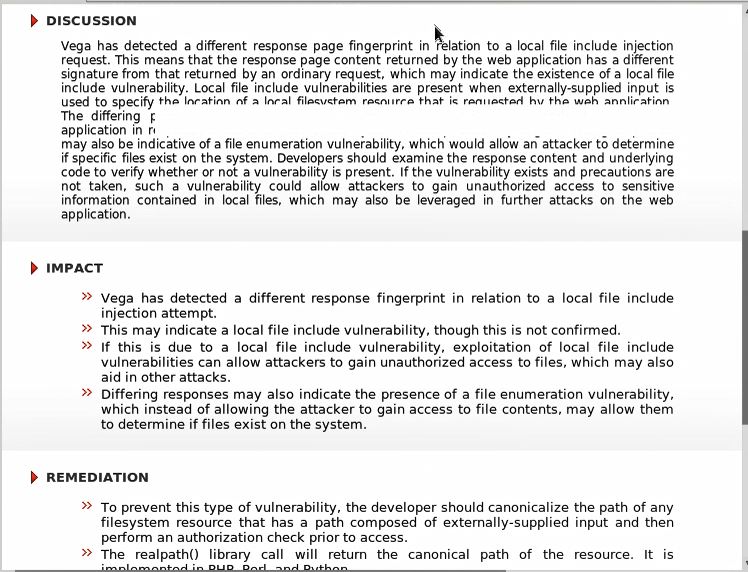


There are several places of the website where cross-site scripting is a large vulnerability and can allow for unauthorized access. It can be used to manipulate the content of the site for other users and it can be used to perform actions within the application without user knowledge. This happens when there is a lack of input validation on the website, which could circumvent the same-origin policy of the website.

The developer would have to go through the website and see if there is any untrustworthy data being given to the user, then filter that data. Then, the developer would need to patch up these holes by visiting a recommended place called the OWASP XSS Prevention Sheet, which will show the developer how to prevent cross-site scripting.

High Risk – Local File Include/File Enumeration Vulnerability Possibility

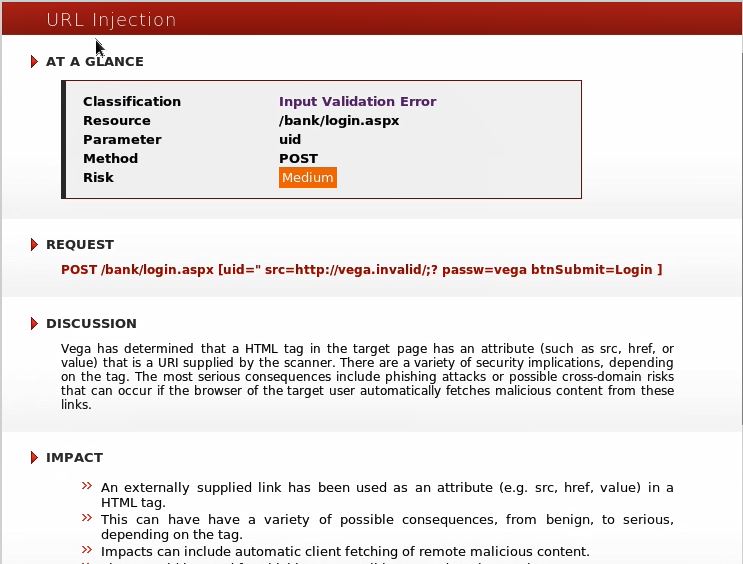


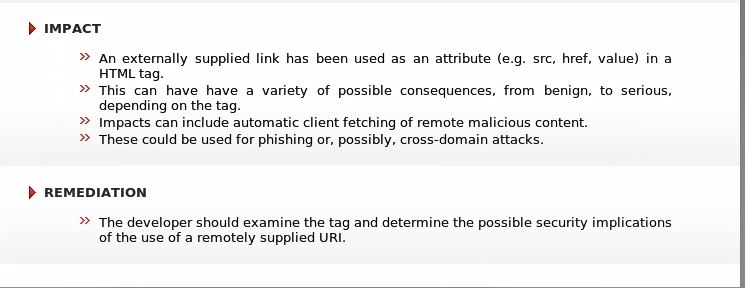


There are several places on several different pages on the website where a local file include vulnerability may exist. This occurs when an externally-supplied input is used to specify the location of a local filesystem resource requested by the web application. These vulnerabilities may allow attackers to gain unauthorized access to files, which may be used to further attacks. It is also possible it can be a file enumeration vulnerability, which allows the attacker to determine if files exist on a system.

This vulnerability will take longer than earlier vulnerabilities to patch as it will require the developer of the website to normalize the path of any filesystem resource made of externally-supplied input and then perform an authorization check prior to its use. This will require the developer to, most likely, to go through a lot of filesystem resources to find these resources and it might take a while to fix them as well.

Medium Risk – URL Injection

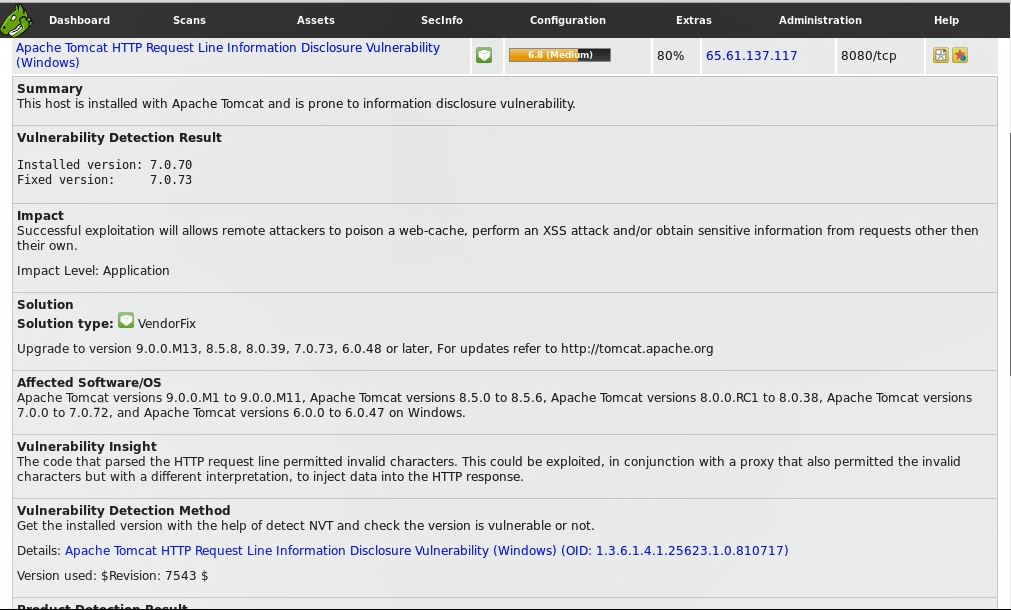




A medium size risk is that there are HTML tags in the target that has an attribute that is a URI provided by the Vega scanner. This can allow for phishing attacks or if a user has malicious content from other sites. This however, can be very serious or benign.

This, however, is relatively easy to fix. The developer of the website should examine the HTML tag and determine what vulnerabilities it could cause. The developer should modify the tag if necessary.

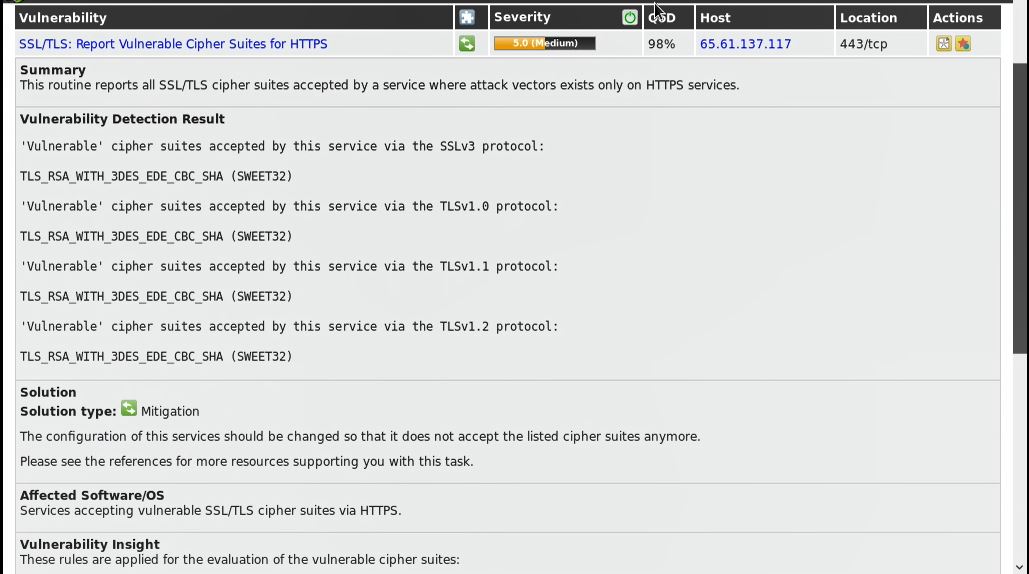
Medium Risk – Apache Tomcat Vulnerabilities



There are several medium-risk vulnerabilities on the system regarding Apache Tomcat and its version. There are several parts of it that are vulnerable to information disclosure such as Security Manager and HTTP Line Request, which would affect confidentiality of data in the system. There also some bypass vulnerability and redirect vulnerabilities with Apache Tomcat, which could allow for higher-risk of attacks. About 11 out of the 25 found medium-risk were due to Apache Tomcat and its vulnerabilities.

The fix, however, will depend on the how many services and web servers are holding up this website. This is because the fix for these issues is a vendor fix and all the issues should be patched if the Apache Tomcat running would be upgraded to at least software version 7.0.91 from 7.0.70. If the upgrade occurs many of these issues should not be vulnerable anymore.

Medium Risk – SSL/TLS Vulnerabilities and Weak Cipher Suites



There are several vulnerabilities regarding SSL/TLS on the website. All these vulnerabilities allow for different attacks and are slightly different in why they occur. They can allow an attacker to break through more easily than usual and are because of the cipher suites.

While the SSL/TLS vulnerabilities are slightly different, the solution is the same for all of them. The solution is to go through a list of several cipher suites listed as weak and/or vulnerable and stop them from being accepted by the SSL/TLS protocols. This is a simple fix but requires the developer to go through a list and disable several insecure cipher suites.

Medium Risk – Microsoft IIS Tilde Character Information Disclosure Vulnerabilities



There are several vulnerabilities regarding the Microsoft II webserver and it is prone to information disclosure. Microsoft IIS fails to validate a specially crafted GET request containing a tilde character, which discloses information of many names of files and folders.

Unfortunately, there are no known fixes for these issues from either Microsoft or third-parties. So, this unfortunately cannot be patched up as of right now. However, it lower risk vulnerability and should not cause too many issues in the long run.

Conclusion

The website http://demo.testfire.net/ is at serious risk of being attacked. The website has large amounts of high-risk and medium-risk vulnerabilities that really risk the safety of the website. I, personally, would not be comfortable to use or trust this site in its current state. It has many vulnerabilities that risk the data on the website and the information of its users. It is very easy, currently, to gain access to the website and use other methods to break in as well. However, there are mitigation strategies which can bring the website to safe levels, where a user can be comfortable using the website and not have to worry about their security often. The website has been found to be dangerously insecure, but it can be fixed with some work and resources put to fix it.