***

Web Application Review for *

*** Site Review**

Completed by Brad Causey on *

***

Table of Contents

I. Executive Summary

*****

This Web Application Review outlines finding as the result of testing performed by ****** during the ****.

*** uses the OWASP testing methodology, described below. This site review was performing using the OWASP Testing Guide v3.

What is Web Application Penetration Testing?
A penetration test is a method of evaluating the security of a computer system or network by simulating an attack. A Web Application Penetration Test focuses only on evaluating the security of a web application.
The process involves an active analysis of the application for any weaknesses, technical flaws, or vulnerabilities. Any security issues that are found will be presented to the system owner together with an assessment of their impact and often with a proposal for mitigation or a technical solution.

What is a vulnerability?

A vulnerability is a flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy. A threat is a potential attack that, by exploiting a vulnerability, may harm the assets owned by an application (resources of value, such as the data in a database or in the file system). A test is an action that tends to show a vulnerability in the application.

What is the OWASP testing methodology?

Penetration testing will never be an exact science where a complete list of all possible issues that should be tested can be defined. Indeed, penetration testing is only an appropriate technique for testing the security of web applications under certain circumstances. The goal is to collect all the possible testing techniques, explain them and keep the guide updated.
The OWASP Web Application Penetration Testing method is based on the black box approach. The tester knows nothing or very little information about the application to be tested. The testing model consists of:

- Tester: Who performs the testing activities
- Tools and methodology: The core of this Testing Guide project
- Application: The black box to test

The test is divided into 2 phases:

- Passive mode: in the passive mode, the tester tries to understand the application's logic, and plays with the application. Tools can be used for information gathering, for example, an HTTP proxy to observe all the HTTP

requests and responses. At the end of this phase, the tester should understand all the access points (*gates*) of the application (e.g., HTTP headers, parameters, and cookies). The Information Gathering section explains how to perform a passive mode test. For example, the tester could find the following:

https://www.example.com/login/Authentic_Form.html

This may indicate an authentication form in which the application requests a username and a password.
The following parameters represent two access points (gates) to the application:

http://www.example.com/Appx.jsp?a=1&b=1

In this case, the application shows two gates (parameters a and b). All the gates found in this phase represent a point of testing. A spreadsheet with the directory tree of the application and all the access points would be useful for the second phase.

- Active mode: in this phase, the tester begins to test using the methodology described in the follow paragraphs.

We have split the set of active tests in 9 sub-categories for a total of 66 controls:

- Configuration Management Testing
- Business Logic Testing
- Authentication Testing
- Authorization testing
- Session Management Testing
- Data Validation Testing
- Denial of Service Testing
- Web Services Testing
- Ajax Testing

*** found 3 high risk vulnerabilities, 3 medium risk vulnerabilities, and 2 low risk vulnerabilities. The highest of concern include Reflexive Cross Site Scripting and HTTP server path traversal. These specific high risk vulnerabilities are a result of incorrect cookie handling by the infrastructure and application code. By utilizing Reflexive Cross Site Scripting, an attacker could further compromise a customer computer beyond the browser. This is particularly concerning because authentication is not required for a successful exploit in this case.

II. Assessment Findings

This section describes in detail any vulnerabilities discovered during the web application review.

This section is aimed at a technical level and should be used to gain more understanding of the vulnerabilities for the purposes of mitigation.

The findings section will include:

- An ID number
- A reference number with screenshots where possible
- The affected item
- A technical description of the issue
- A section on resolving the issue
- The risk rating

OWASP-AT-001

Vulnerability Name - Credentials transport over an encrypted channel

Affected Item - http://***:80/

Discovery – Firewall does not use HTTPS for authentication

Resolution – Force HTTPS for all authentication

Risk Rating – Medium

ID number - 001

General Issue Description - Nowadays, the most common example of this issue is the login page of a web application. The tester should verify that user's credentials are transmitted via an encrypted channel. In order to log into a web site, usually, the user has to fill a simple form that transmits the inserted data with the POST method. What is less obvious is that this data can be passed using the HTTP protocol, that means in a non-secure way, or using HTTPS, which encrypts the data. To further complicate things, there is the possibility that the site has the login page accessible via HTTP (making us believe that the transmission is insecure), but then it actually sends data via HTTPS. This test is done to be sure that an attacker cannot retrieve sensitive information by simply sniffing the net with a sniffer tool.

OWASP-IG-006

Vulnerability Name – Analysis of Error Code

Affected Item – https://****

Discovery – Apache Tomcat/6.0.13 500 error message

Resolution – Use standardized, branded error messages and consistent HTTP response codes to mask sensitive infrastructure information.

Risk Rating – Low

ID number - 002

General Issue Description – Often during a penetration test on web applications we come up against many error codes generated from applications or web servers. It's possible to cause these errors to be displayed by using a particular request, either specially crafted with tools or created manually. These codes are very useful to penetration testers during their activities because they reveal a lot of information about databases, bugs, and other technological components directly linked with web applications. Within this section we'll analyze the more common codes (error messages) and bring into focus the steps of vulnerability assessment. The most important aspect for this activity is to focus one's attention on these errors, seeing

them as a collection of information that will aid in the next steps of our analysis. A good collection can facilitate assessment efficiency by decreasing the overall time taken to perform the penetration test.

OWASP-IG-006

Vulnerability Name – User Controlled Error Message

Affected Item - http://***

Discovery - <h1>HTTP 404: Not Found Error</h1>
Unable to invoke action named "I should not be able to control screen output"

Resolution – Use universal server-wide error pages that do not echo back user input.

Risk Rating – Low

ID number - 003

General Issue Description - There are several different vendors and versions of web servers on the market today. Knowing the type of web server that you are testing significantly helps in the testing process, and will also change the course of the test. This information can be derived by sending the web server specific commands and analyzing the output, as each version of web server software may respond differently to these commands. By knowing how each type of web server responds to specific commands and keeping this information in a web server fingerprint database, a penetration tester can send these commands to the web server, analyze the response, and compare it to the database of known signatures. Please note that it usually takes several different commands to accurately identify the web server, as different versions may react similarly to the same command. Rarely, however, two different versions have the same response to all HTTP commands. So, by sending several different commands, you increase the accuracy of your guess.

OWASP-CM-001

Vulnerability Name – HTTP running on secure site

Affected Item – http://***

Discovery – HTTP does not redirect or force HTTPS

Resolution – Users should be forced to HTTPS by the infrastructure configuration. It should not be possible to force HTTP and continue operation in the application.

Risk Rating – Medium

ID number - 004

General Issue Description - The http clear-text protocol is normally secured via an SSL or TLS tunnel, resulting in https traffic. In addition to providing encryption of data in transit, https allows the identification of servers (and, optionally, of clients) by means of digital certificates.

Historically, there have been limitations set in place by the U.S. government to allow cryptosystems to be exported only for key sizes of at most 40 bits, a key length which could be broken and would allow the decryption of communications. Since then cryptographic export regulations have been relaxed (though some constraints still hold), however it is important to check the SSL configuration being used to avoid putting in place cryptographic support which could be easily defeated. SSL-based services should not offer the possibility to choose weak ciphers.

Technically, cipher determination is performed as follows. In the initial phase of a SSL connection setup, the client sends to the server a Client Hello message specifying, among other information, the cipher suites that it is able to handle. A client is usually a web browser (most popular SSL client nowadays), but not necessarily, since it can be any SSL-enabled application; the same holds for the server, which needs not be a web server, though this is the most common case. (For example, a noteworthy class of SSL clients is that of SSL proxies such as stunnel (www.stunnel.org) which can be used to allow non-SSL enabled tools to talk to SSL services.) A cipher suite is specified by an encryption protocol (DES, RC4, AES), the encryption key length (such as 40, 56, or 128 bits), and a hash algorithm (SHA, MD5) used for integrity checking. Upon receiving a Client Hello message, the server decides which cipher suite it will use for that session. It is possible (for example, by means of configuration directives) to specify which cipher suites the server will honour. In this way you may control, for example, whether or not conversations with clients will support 40-bit encryption only.

OWASP-IG-006

Vulnerability Name – User controllable error message

Affected Item – https://***

Discovery – **

Resolution – Use universal server-wide error pages that do not echo back user input.

Risk Rating – Medium

ID number - 005

General Issue Description - There are several different vendors and versions of web servers on the market today. Knowing the type of web server that you are testing significantly helps in the testing process, and will also change the course of the test. This information can be derived by sending the web server specific commands and analyzing the output, as each version of web server software may respond differently to these commands. By knowing how each type of web server responds to specific commands and keeping this information in a web server fingerprint database, a penetration tester can send these commands to the web server, analyze the response, and compare it to the database of known signatures. Please note that it usually takes several different commands to accurately identify the web server, as different versions may react similarly to the same command. Rarely, however, two different versions have the same response to all HTTP commands. So, by sending several different commands, you increase the accuracy of your guess.

OWASP-DV-001

Vulnerability Name – Reflexive Cross Site Scripting

Affected Item – https://***

Discovery – The *** parameter *** is vulnerable to rXSS

Resolution – Use a whitelist approach to filter all user controllable parameters

Risk Rating – High

ID number - 006

General Issue Description - Reflected XSS attacks are also known as type 1 or non-persistent XSS attacks, and are the most frequent type of XSS attacks found nowadays.

When a web application is vulnerable to this type of attack, it will pass unvalidated input sent through requests to the client. The common modus operandi of the attack includes a design step, in which the attacker creates and tests an offending URI, a social engineering step, in which she convinces her victims to load this URI on their

browsers, and the eventual execution of the offending code — using the victim's credentials.

Commonly the attacker's code is written in the Javascript language, but other scripting languages are also used, e.g., ActionScript and VBScript.

Attackers typically leverage these vulnerabilities to install key loggers, steal victim cookies, perform clipboard theft, and change the content of the page (e.g., download links).

One of the important matters about exploiting XSS vulnerabilities is character encoding. In some cases, the web server or the web application could not be filtering some encodings of characters, so, for example, the web application might filter out "<script>", but might not filter %3cscript%3e which simply includes another encoding of tags.

IV. Toolbox

This section is often used to describe the commercial and open-source tools that were used in conducting the assessment. When custom scripts/code are utilized during the assessment, it will be disclosed in this section or noted as attachment.

Brief of Tools used:

- Mozilla Firefox Web Browser with Plugins:
    - XSS Me
    - SQL Inject Me
    - Foxy Proxy
    - Tamper Data
- WebScarab
- Nikto
- Nmapfe
- SSLDigger
- Manual Testing
- Burp Suite Pro