

Penetration Testing & Deep Code Analyst Report

For EXAMPLE CLIENT

By Matt Dobinson Date: 21/09/2011 Version: V1.02

ATTENTION

This document contains information from Fireworks Websites that is confidential and privileged. The information is intended for the private use of EXAMPLE CLIENT. By accepting this document you agree to keep the contents in confidence and not copy, disclose, or distribute this without written request to and written confirmation from NII. If you are not the intended recipient, be aware that any disclosure, copying, or distribution of the contents of this document is prohibited.

Document Details

Title	Penetration Testing & Deep Code Analyst Report
Version	V1.02
Author	Matthew Dobinson
Pen-testers	Matthew Dobinson
Reviewed By	EXAMPLE CLIENT
Approved By	EXAMPLE CLIENT
Classification	Confidential

Recipient

Name	Title	Company
EXAMPLE CLIENT	Managing Director	EXAMPLE CLIENT

Version Control

Version	Date	Author	Description
V1.01	14/09/2011	Matthew Dobinson	Initial Testing
V1.02	21/09/2011	Matthew Dobinson	Final Report

Table of Contents

1 EXECUTIVE SUMMARY	4
1.1 SUMMARY.....	4
1.1.1 Approach.....	4
1.1.2 Scope of Work.....	4
1.1.3 Project Objectives.....	4
1.1.4 Timeline.....	5
1.1.5 Summary of Findings.....	5
1.2 METHODOLOGY.....	5
1.2.1 Planning	5
1.2.2 Exploitation	5
1.2.3 Reporting	6
4. TECHNICAL REPORT	6
4.1 SESSION HIJACKING.....	6
4.2 GENERIC SQL INJECTION VULNERABILITY (PARAMETERS NAMES)	7
4.3 UNHANDLED EXCEPTIONS ON MULTIPLE PAGES	7
4.4 CROSS SITE SCRIPTING	8
4.5 LOCAL FILESYSTEM PATHS FOUND.....	9
4.6 PHP EXPOSE_PHP INFORMATION DISCLOSURE	9
4.7 WEB SERVER ROBOTS.TXT INFORMATION DISCLOSURE	10
4.8 PHPINFO PAGE PRESENT	10
4.9 EMAIL ADDRESS DISCLOSURE	11
5 CONCLUSION.....	12
6 REFERENCES.....	12
APPENDIX A - NESSUS VULNERABILITY SCANNING REPORT	12
APPENDIX B – WEBSECURIFY REPORTS.....	12

1 Executive Summary

1.1 Summary

EXAMPLE CLIENT has assigned the task of carrying out a deep code analyst and penetration testing of the web application located on EXAMPLE CLIENT.com.au.

This is the second Penetration Testing report. This Penetration Test was performed during 22nd September 2011 and 27th September 2011. The detailed report about each task and our findings are described below.

The purpose of the test is to determine security vulnerabilities in the web applications running on the server specified as part of the scope. The tests are carried out assuming the identity of an attacker or a user with malicious intent. At the same time due care is taken not to harm the server or database.

1.1.1 Approach

- Perform broad scans to identify potential areas of exposure and services that may act as entry points
- Perform targeted scans and manual investigation to validate vulnerabilities
- Test identified components to gain access to: www.EXAMPLE CLIENT.com.au
- Identify and validate vulnerabilities
- Rank vulnerabilities based on risk level of exploitation
- Perform supplemental research and development activities to support analysis
- Identify issues of immediate consequence and recommend solutions
- Develop long-term recommendations to enhance security
- Transfer knowledge

During the web application level we checked the web servers' configuration issues, and more importantly the logical errors in the web application itself.

1.1.2 Scope of Work

The scope of this penetration test was limited to the below mentioned web application.

www.EXAMPLE CLIENT.com.au

1.1.3 Project Objectives

This security assessment is carried out to gauge the security posture of EXAMPLE CLIENT's Internet facing web application. The result of the assessment is then analysed for vulnerabilities. Given the limited time that is given to perform the assessment, only immediately exploitable services have been tested.

The vulnerabilities are assigned a risk rating based on threat, vulnerability and impact.

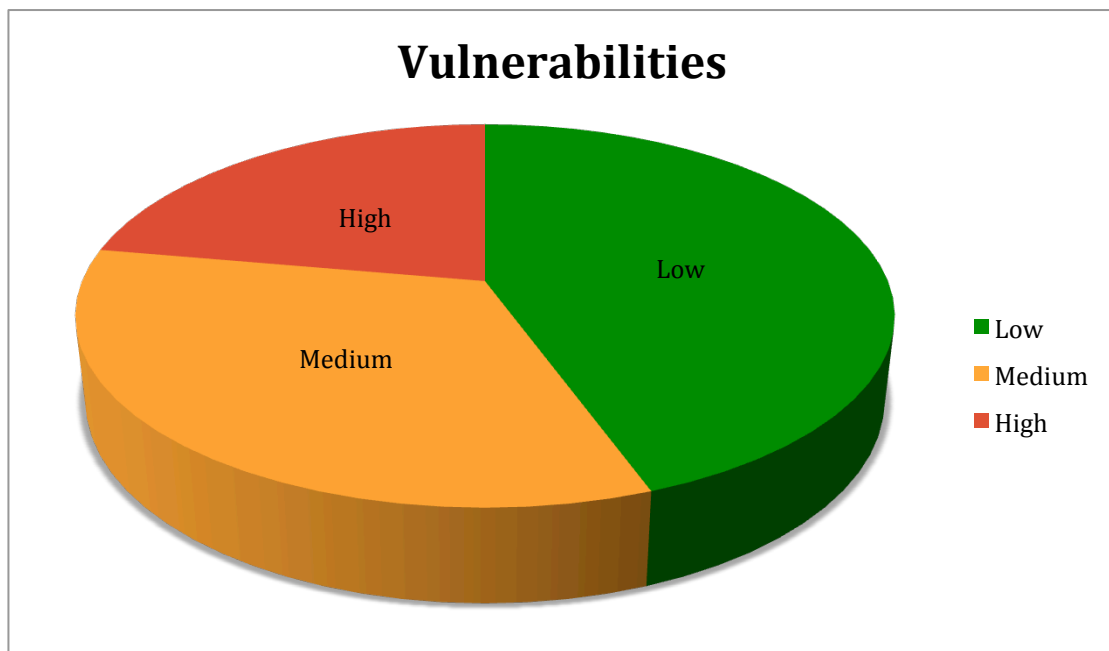
1.1.4 Timeline

The timeline of the test is as below:

Penetration Testing	Start Date/Time	End Date/Time
Initial Testing	14/09/2011	14/09/2011
Final Testing	21/09/2011	27/09/2011

1.1.5 Summary of Findings

Value	Number of Risks
Low	4
Medium	3
High	2



1.2 Methodology

1.2.1 Planning

During planning we gather information from the server in which the web application is installed. Then, we detect the path information and identifiable software and determined the running their versions.

1.2.2 Exploitation

Utilizing the information gathered in planning we start to find the vulnerability for each piece of software and service that we discovered after that trying to exploit it.

1.2.3 Reporting

Based on the results from the first two steps, we start analysing the results. Our Risk rating is based on this calculation:

$$\text{Risk} = \text{Threat} * \text{Vulnerability} * \text{Impact}$$

Threat		Low				Medium				High				Critical			
Vulnerability		L	M	H	C	L	M	H	C	L	M	H	C	L	M	H	C
Impact	Low																
	Medium																
	High																
	Critical																

L	Low	1-16
M	Medium	17-32
H	High	33-48
C	Critical	49-64

After calculating the risk rating, we start writing the report on each risk and how to mitigate it.

**Based on our analysis risks that falls under this category will be considered as High.*

4. Technical Report

4.1 Session Hijacking

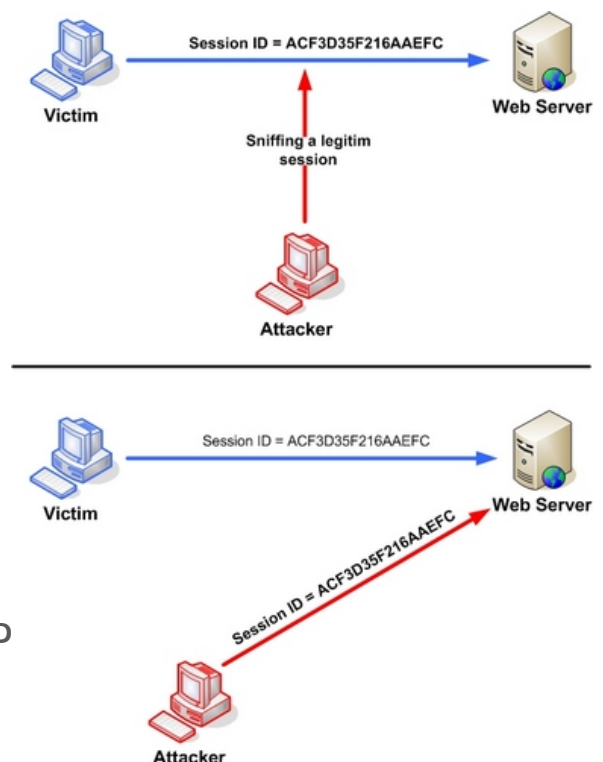
Risk: High

Classification: Authorization

Resource: All shopping cart pages

Analyst:

Fireworks have found that it is possible to hijack another users session using their unique session identifier code. This can be used to hijack any account for which the attacker has a identifier code. This code is contained within the URL and as such can be easily copied using methods like network sniffing, cross site scripting, trojans/malware and viewing browser history.



Recommendation:

Force the session to contain and verify an ip or other system identifier, also restrict session files to be cleared out at a regular interval.

4.2 Generic SQL Injection Vulnerability (Parameters Names)

Risk: High

Classification: Information

Resource: /customers/login.php?'+convert(int,convert(varchar,0x7b5d))+'=1

Analyst:

By providing specially crafted parameters to PHP, we were able to get an error from the underlying database. This error suggests that the PHP is affected by a SQL injection vulnerability.

An attacker may be able to exploit this flaw to bypass authentication, read confidential data, modify the remote database, or even take control of the remote operating system.

Recommendation:

Modify the relevant PHP so that it properly escape arguments.

4.3 Unhandled exceptions on multiple pages

Risk: Medium

Classification: Information

Resource:

<http://EXAMPLE CLIENT.com.au/customers/login.php>

<http://EXAMPLE CLIENT.com.au/subdirectory/wishlist.php?warning=wrong%20email%20or%20password&wish=1&pid=&p=&v1=&t=>

<http://www.EXAMPLE CLIENT.com.au/subdirectory/notify.php>

<http://www.EXAMPLE CLIENT.com.au/subdirectory/specs.html?cid=5625&myorder=price&mydirection=ASC'&limit=150d&mypage=0>

Analyst:

Unhandled exceptions are instances where certain events occur in an unexpected manner in which the programmers did not foresee and therefore the programmed logic is unable to anticipate, resulting in an exception and non-user friendly error message.

Recommendation:

Review all source code and modify logic to handle unexpected events, and declare variables first prior to use. There is also a PHP API function `mysql_real_escape_string`. See: <http://php.net/manual/en/function.mysql-real-escape-string.php>

4.4 Cross Site Scripting

Risk: Medium

Classification: Information

Resource:

`http://www.EXAMPLE CLIENT.com.au/subdirectory/search.php?search=<script>alert(1);</script>`
`http://www.EXAMPLE CLIENT.com.au/subdirectory/editAll.php?email="><script>alert(1);</script>`
`http://www.EXAMPLE CLIENT.com.au//datafolder/aMediaGallery.php?cid=10"/><h1>hello</h1>`
`http://www.EXAMPLE CLIENT.com.au/products/index.php?warning="><script>alert(1);</script>`
`http://www.EXAMPLE CLIENT.com.au/about_EXAMPLE CLIENT.html?cur=<wslite>`
`http://www.EXAMPLE CLIENT.com.au/index.html?cur=<wslite>`
`http://www.EXAMPLE CLIENT.com.au/pages/help_top.html?cur=<wslite>`
`http://www.EXAMPLE CLIENT.com.au/pages/security_new.html?cur=<wslite>`
`http://www.EXAMPLE CLIENT.com.au/pages/help_main.html?cur=<wslite>`
`http://www.EXAMPLE CLIENT.com.au/pages/help_left.html?cur=<wslite>`
`http://www.EXAMPLE CLIENT.com.au/pages/privacy_new.html?cur=<wslite>`
`http://www.EXAMPLE CLIENT.com.au/pages/returns_new.html?cur=<wslite>`
`http://www.EXAMPLE CLIENT.com.au/pages/contact_new.html?cur=<wslite>`
`http://www.EXAMPLE CLIENT.com.au/pages/sizing_new.html?cur=<wslite>`
`http://www.EXAMPLE CLIENT.com.au/pages/faqs_new.html?cur=<wslite>`
`http://www.EXAMPLE CLIENT.com.au/pages/terms_new.html?cur=<wslite>`
`http://www.EXAMPLE CLIENT.com.au/pages/feedback_new.html?cur=<wslite>`

Analyst:

A cross-site scripting vulnerability is a weakness in a website, where malicious content may be injected into a web page and thus attack the website visitor. It is commonly used in 'phishing' campaigns for banking and other financial websites and to impersonate the legitimate business.

Alternatively, it is often used to infect the client browser with malicious HTML or JavaScript where exploits may be inserted, thereby compromising the user experience and gaining access to their system, or stealing their authentication cookies and compromising their online account and password.

The www.EXAMPLE CLIENT.com.au website is vulnerable in almost every form.

Recommendation:

Modify all user input scripts to strip malicious characters first, such as ' " < > * and other symbols. The PHP API provides the htmlentities function to facilitate this.

See: <http://php.net/manual/en/function.htmlentities.php>

4.5 Local Filesystem Paths Found

Risk: Medium

Classification: Information

Resource:

/subdirectory/file.html?mycname=must-haves
/products/pages/search.php?search=;

Analyst:

Fireworks have detected an absolute filesystem path (i.e. one that is not relative to the web root). This information is sensitive, as it may reveal things about the server environment to an attacker. Knowing filesystem layout can increase the chances of success for blind attacks. Full system paths are very often found in error output. This output should never be sent to clients on production systems. It should be redirected to another output channel (such as an error log) for analysis by developers and system administrators.

Recommendation:

Absolute paths are often found in error output.

Both the system administrators and developers should be made aware, as the problem may be due to an application error or server misconfiguration.

Error output containing sensitive information such as absolute system paths should not be sent to remote clients on production servers.

This output should be sent to another output stream, such as an error log.

4.6 PHP expose_php Information Disclosure

Risk: Low

Classification: Information

Resource: http://EXAMPLE
CLIENT.com.au/customers/login.php/?=PHPB8B5F2A0-3C92-11d3-A3A9-
4C7B08C10000

Analyst:

The PHP install on the remote server is configured in a way that allows disclosure of potentially sensitive information to an attacker through a special URL. Such an URL triggers an Easter egg built into PHP itself.

Other such Easter eggs likely exist, but Nessus has not checked for them.

Recommendation:

In the PHP configuration file, php.ini, set the value for 'expose_php' to 'Off' to disable this behavior. Restart the web server daemon to put this change into effect.

See Also

http://www.0php.com/php_easter_egg.php
<http://seclists.org/webappsec/2004/q4/324>

4.7 Web Server robots.txt Information Disclosure

Risk: Low

Classification: Information

Resource: /robots.txt

Analyst:

The remote host contains a file named 'robots.txt' that is intended to prevent web 'robots' from visiting certain directories in a web site for maintenance or indexing purposes. A malicious user may also be able to use the contents of this file to learn of sensitive documents or directories on the affected site and either retrieve them directly or target them for other attacks.

Recommendation:

Review the contents of the site's robots.txt file, use Robots META tags instead of entries in the robots.txt file, and/or adjust the web server's access controls to limit access to sensitive material.

4.8 PHPINFO Page Present

Risk: Low

Classification: Information

Resource: http://www.EXAMPLE CLIENT.com.au/phpinfo.php

Analyst:

Many PHP installation tutorials instruct the user to create a PHP file that calls the PHP function 'phpinfo()' for debugging purposes. Various PHP applications may also include such a file. By accessing such a file, a remote attacker can discover a large amount of information about the remote web server, including :

- The username of the user who installed php and if they are a SUDO user.
- The IP address of the host.
- The version of the operating system.
- The web server version.
- The root directory of the web server.
- Configuration information about the remote PHP

Recommendation:

The phpinfo.php script is harmless and may be deleted. Removal of the script will prevent disclosure of sensitive information such as CPU type, memory, kernel version, internal system paths, security features enabled or disabled and installed software. Another possibility is that the server is automatically configured to include e-mail addresses. Tweaking configuration can usually remove these.

4.9 Email Address Disclosure

Risk: Low

Classification: Information

Resource: All *REMOVED* Pages

Analyst:

Fireworks have found patterns that resemble e-mail addresses in scanned content. These may be user the addresses of system users, addresses inserted in user-supplied content, or third-party addresses embedded in components of the application (such as Javascript libraries). Automatically scraping websites is one way that spammers and phishers collect e-mail addresses for their distribution lists. It is recommended that e-mail addresses not be displayed on exposed parts of the web application, directly or indirectly.

Recommendation:

If the e-mail addresses are those of users, developers should investigate why they are being output and try to remove or obfuscate them.

E-mail addresses embedded in user-supplied content should be filtered or obfuscated to prevent unintended disclosure.

It is common that e-mail addresses may be automatically included in third-party components, such as Javascript libraries.

Another possibility is that the server is automatically configured to include e-mail addresses. Tweaking configuration can usually remove these.

5 Conclusion

For systems to remain secure, security posture must be evaluated and improved continuously. Establishing the organizational structure that will support these ongoing improvements is essential in order to maintain control of corporate information systems.

We conclude that the overall security has been improved. We hope that the web application is review every 6 months or year depending on the amount of changes to the source code.

6 References

Appendix A - Nessus Vulnerability Scanning Report

Attached nessus scan executive report.

Appendix B – Websecurify Reports

Attached Websecurify vulnerability report.