TESSOLVE
A Hero Electronix Venture

Vulnerability Assessment and Penetration Testing Model Report

for

SampleCompany

FILE COPY

D

Document Submission Details

Company	SampleCompany
Document Title	VA & PT Report – v 1.0
Date	10/03/2014
Reference	INFY/VAPT/
Classification	
Document Type	Report

Recipient Details

Name	
Title	
Company	SampleCompany
Address	
Contact Numbers	
Other E–Mails	

Document History

Date	Version	Author	Comments
	V1.0		Rough Draft
	V1.0		Rough Draft
	V1.0		Fair Draft
	V1.0		Client Interaction

Notice

Tessolve Semiconductor Pvt. Ltd. Contact Details

Name	Ponni Carlin
Title	Associate Director – Business Development Sales & Marketing
Company	Tessovle Semiconductor Pvt. Ltd.
Address	Plot No: 31 (P2), Electronic City Phase II, Bangalore – 560 100 Karnataka, India
Telephone	+91 80 4181 2626
Mobile	+1 (408) 204-8998
E – Mail	ponni.carlin@tessolve.com

Organisation Details:

S.No	Particulars	Details to be furnished
1	Name of the Information System Auditor/Consultant /Company	Tessolve Semiconductor Private Limited
2	Location of Registered Office (India)	Plot No: 31 (P2), Electronic City Phase II, Bangalore – 560 100 Karnataka, India
3	Location of Registered Office (UK)	Engine Shed Station Approach Temple Meads Bristol BS1 6QH United Kingdom
4	Year of Establishment	Within India: 2004 United Kingdom: 2020
5	Mailing Address (India)	Plot No: 31 (P2), Electronic City Phase II, Bangalore – 560 100 Karnataka, India
6	Mailing Address (UK)	Engine Shed Station Approach Temple Meads Bristol BS1 6QH United Kingdom
7	Registered Company Number	
8	Official Contact Numbers	+91- 80 4181 2626
9	Telephone Numbers of the contact person	+1 (408) 204-8998
10	E-mail address of the contact person	Ponni.carlin@tessolve.com sales@tessolve.com
11	Name and designation of the person authorized to make commitments to the bank	
12	Description of business and business background	Redefining the threshold of semiconductor technology

Contents

Executive Summary

About Application

This is a complete Application Assessment Report comprising the outcomes of testing undertaken on the SampleCompany.com application for SampleCompany. The purpose of the testing was to review the application vulnerabilities. This platform is for the security vulnerabilities and provides remediation advice. Testing was conducted from the perspective of a malicious user attempting to compromise the payment gateway application

This penetration test raised a 39 issues relating to the security stance of the SampleCompany.com web application. There were multiple findings of a High, Medium, Low and Informational severities. Multiple application level vulnerabilities were discovered which are considered contrary to security best practice, and contrary to the OWASP (Open Web Application Security Project) developer guidelines.

Scope of the project

The following checks were performed on web application as part of Web application security Assessment achieved using tool and manual approach.

- Application Vulnerability Assessment
- Penetration Testing – White box&Blackbox.
- OWASP Standard 2013 coverage.

Out of scope

The below are considered as out of scope.

- Functional Testing
- Regression Testing
- Performance Testing
- Secure Code Audit
- Stress and Load (DOS & DDOS) Testing
- Test Environment Management Activities
- Any other testing activity not listed in Section 1.2

Project Summary:

Project Name – SampleCompany.com Penetration Testing						
Project Start Date - 28th February 2014				Project End date - 12th March 2014		
S.No	Activity Description	Planned		Actual		Percentage of completion
		Start Date	End Date	Start Date	End Date	
1	Information gatherings	1-Mar-14	1-Mar-14	1-Mar-14	1-Mar-14	100%
2	Vulnerability Scanning	3-Mar-14	3-Mar-14	3-Mar-14	4-Mar-14	100%
3	Penetration Testing	5-Mar-14	5-Mar-14	5-Mar-14	7-Mar-14	100%
4	Report Preparation	8-Mar-14	8-Mar-14	11-Mar-14	11-Mar-14	100%
5	Report Submission	12-Mar-14	12-Mar-14	12-Mar-14	12-Mar-14	100%

Vulnerability Test Cases

S.No	Test Name	Status	Risk
1	Credentials transport over an encrypted channel - Credentials transport over an encrypted channel	Done	H
2	Testing for user enumeration - User enumeration	Done	H
3	Testing for Guessable (Dictionary) User Account - Guessable user account	Done	H
4	Testing for vulnerable remember password and pwd reset - Vulnerable remember password, weak pwd reset	Done	H
5	Testing for Session Management Schema - Bypassing Session Management Schema, Weak Session Token	Done	H
6	Testing for CSRF - CSRF	Done	H
7	Testing for Stored Cross Site Scripting - Stored XSS	Done	H
8	SQL Injection - SQL Injection	Done	H
9	Buffer overflow - Buffer overflow	Done	H
10	Search Engine Discovery/Reconnaissance	Done	M
11	Identify application entry points	Done	M
12	Testing for Web Application Fingerprint	Done	M
13	Application Discovery	Done	M
14	Application Configuration Management Testing - Application Configuration management weakness	Done	M
15	Testing for File Extensions Handling - File extensions handling	Done	M
16	Old, backup and unreferenced files - Old, backup and unreferenced files	Done	M
17	Testing for Cookies attributes - Cookies are set not 'HTTP Only', 'Secure', and no time validity	Done	M
18	Testing for Exposed Session Variables - Exposed sensitive session variables	Done	M
19	Incubated vulnerability - Incubated vulnerability	Done	M
20	Default / Brute Force Testing - Credentials	Done	L
21	Testing for bypassing authentication schema - Bypassing authentication schema	Done	L
22	Testing for Path Traversal - Path Traversal	Done	L
23	Spiders, Robots and Crawlers	Done	I
24	Analysis of Error Codes	Done	I
25	Infrastructure Configuration Management Testing - Infrastructure Configuration management weakness	Done	I
26	SSL/TLS Testing (SSL Version, Algorithms, Key length, Digital Cert. Validity) - SSL Weakness	Done	I
27	Infrastructure and Application Admin Interfaces - Access to Admin interfaces	Done	I

S.No	Test Name	Status	Risk
28	Testing for HTTP Methods and XST - HTTP Methods enabled, XST permitted, HTTP Verb	Done	I
29	Testing for Logout and Browser Cache Management - - Logout function not properly implemented, browser cache weakness	Done	I
30	Testing for CAPTCHA - Weak Captcha implementation	Done	I
31	Testing for Session Fixation - Session Fixation	Done	I
32	LDAP Injection - LDAP Injection	Done	I
33	Testing for HTTP Splitting/Smuggling - HTTP Splitting, Smuggling	Done	I
34	Testing for SQL Wildcard Attacks - SQL Wildcard vulnerability	Done	I
35	Locking Customer Accounts - Locking Customer Accounts	Done	I
36	WS Information Gathering - N.A.	Done	I
37	Testing WSDL - WSDL Weakness	Done	I
38	XML Structural Testing - Weak XML Structure	Done	I
39	XML content-level Testing - XML content-level	Done	I
40	Testing for Privilege Escalation - Privilege Escalation	Done	I
41	Testing for bypassing authorization schema - Bypassing authorization schema	Not Done	NA
42	Testing Multiple Factors Authentication - Weak Multiple Factors Authentication	Not Done	NA
43	Testing for Race Conditions - Race Conditions vulnerability	Not Done	NA
44	Testing for bypassing authorization schema - Bypassing authorization schema	Not Done	NA
45	Testing for Business Logic - Bypassable business logic	Not Done	NA
46	Testing for Reflected Cross Site Scripting - Reflected XSS	Not Done	NA
47	Testing for DOM based Cross Site Scripting - DOM XSS	Not Done	NA
48	Testing for Cross Site Flashing - Cross Site Flashing	Not Done	NA
49	ORM Injection - ORM Injection	Not Done	NA
50	XML Injection - XML Injection	Not Done	NA
51	SSI Injection - SSI Injection	Not Done	NA
52	XPath Injection - XPath Injection	Not Done	NA
53	IMAP/SMTP Injection - IMAP/SMTP Injection	Not Done	NA
54	Code Injection - Code Injection	Not Done	NA
55	OS Commanding - OS Commanding	Not Done	NA
56	Testing for DoS Buffer Overflows - Buffer Overflows	Not Done	NA
57	User Specified Object Allocation - User Specified Object Allocation	Not Done	NA
58	User Input as a Loop Counter - User Input as a Loop Counter	Not Done	NA
59	Writing User Provided Data to Disk - Writing User Provided Data to Disk	Not Done	NA
60	Failure to Release Resources - Failure to Release Resources	Not Done	NA

S.No	Test Name	Status	Risk
61	Storing too Much Data in Session - Storing too Much Data in Session	Not Done	NA
62	HTTP GET parameters/REST Testing - WS HTTP GET parameters/REST	Not Done	NA
63	Naughty SOAP attachments - WS Naughty SOAP attachments	Not Done	NA
64	Replay Testing - WS Replay Testing	Not Done	NA
65	AJAX Vulnerabilities - N.A.	Not Done	NA
66	AJAX Testing - AJAX weakness	Not Done	NA

Environment Details:

Details of application, environment and access to the same are as below

Item	Description
Website name	SampleCompany.com
URL Details	
Technology	
Type of testing	Vulnerability Assessment & Penetration Testing

Risk Overview:

CVSS and Severity Ratings

Where applicable Security-Assessment rates all discovered vulnerabilities against the CVSS v2 (Common Vulnerability Scoring System). CVSS is an open framework for communicating the characteristics and impact of IT vulnerabilities. The system is a quantitative model which ensures repeatable accurate measurement, while allowing users to see the underlying vulnerability metrics that were used to calculate the final risk.

Severity	Description
High	High severity findings relate to an issue which requires immediate attention and should be given the highest priority by the business. Vulnerabilities will be labelled
Medium	Medium severity finding relates to an issue which has the potential to present a serious
Low	Low severity findings contradict security best practice and have minimal impact on the
Informational	Informational findings relate primarily to none compliance to security best practices or are considered a security feature that would increase the security stance of the environment.

Website DNS Details:

DNS Records:

DNS Records –SampleCompany.com				
Record	**Type**	**TTL**	**Priority**	**Content**
SampleCompany.com	A	1 minute		4.1.20.14 ()
SampleCompany.com	A	1 minute		4.1.20.14 ()
SampleCompany.com	A	1 minute		4.1.20.14 ()
SampleCompany.com	A	1 minute		4.1.20.14 ()
SampleCompany.com	A	1 minute		4.1.20.14 ()
SampleCompany.com	A	1 minute		4.1.20.14 ()
SampleCompany.com	A	1 minute		4.1.20.14 ()
SampleCompany.com	MX	1 minute	1	amxl.google.com
SampleCompany.com	MX	1 minute	10	amx2.googlemail.com
SampleCompany.com	MX	1 minute	10	amx3.googlemail.com
SampleCompany.com	MX	1 minute	10	amx4.googlemail.com
SampleCompany.com	MX	1 minute	10	amx5.googlemail.com
SampleCompany.com	MX	1 minute	5	alt1.amx.l.google.com
SampleCompany.com	MX	1 minute	5	alt2.amx.l.google.com
SampleCompany.com	NS	2 days		ns-115.awsdns-15.com
SampleCompany.com	NS	2 days		ns-153.awsdns-53.org
SampleCompany.com	NS	2 days		ns-164.awsdns-18.co.uk
SampleCompany.com	NS	2 days		ns-81.awsdns-39.net
SampleCompany.com	SOA	15 minutes		ns-184.awsdns-18.co.uk. awsdns-hostmaster.amazon.com. 1 720 900 129600 86400
SampleCompany.com	TXT	1 minute		v=sf1 include:spf-a. SampleCompany.com include:sf-b. SampleCompany.com include:spf-1. SampleCompany.com include:spf-2. SampleCompany.com include:_sf.google.com include:_sf.elasticemail.com ~all
SampleCompany.com	A	1 minute		4.1.20.14 ()

DNS Records –SampleCompany.com				
Record	**Type**	**TTL**	**Priority**	**Content**
SampleCompany.com	A	1 minute		4.1.20.14 ()
SampleCompany.com	A	1 minute		4.1.20.14 ()
SampleCompany.com	A	1 minute		4.1.20.14 ()
blog. SampleCompany.com	A	1 minute		4.1.20.14 ()
blog. SampleCompany.com	A	1 minute		4.1.20.14 ()
blog. SampleCompany.com	A	1 minute		4.1.20.14 ()
blog. SampleCompany.com	A	1 minute		4.1.20.14 ()
help. SampleCompany.com	CNAME	1 minute		samplecompany.zemdesk.com
mail. SampleCompany.com	CNAME	1 minute		ghx.google.com
www. SampleCompany.com	A	1 minute		4.1.20.14 ()
www. SampleCompany.com	A	1 minute		4.1.20.14 ()
www. SampleCompany.com	A	1 minute		4.1.20.14 ()
www. SampleCompany.com	A	1 minute		4.1.20.14 ()
www. SampleCompany.com	A	1 minute		4.1.20.14 ()
www. SampleCompany.com	A	1 minute		4.1.20.14 ()
www. SampleCompany.com	A	1 minute		4.1.20.14 ()

Website hosted on same server:

Reverse IP Look Up details	
SampleCompany.com	4.1.20.14 ()
SampleCompany.com.my	4.1.20.14 ()
SampleCompany.my	4.1.20.14 ()

Application Risk Details:

Risk :	High	Status:	Pass	Reference ID:	01
Vulnerability Name:	Testing for Credentials Transported over an Encrypted Channel				

Description:

There is a flaw in the credentials transported on this application which may lead to disclosure of highly sensitive user information.

Details:

Nowadays, the most common impact of this issue is the login page and the payment page of a web application. It should be aware 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. Testing for credentials transport means to verify that the user's authentication data are transferred via an encrypted channel to avoid being intercepted by malicious users. if the data travels unencrypted from the web browser to the server, or if the web application takes the appropriate security measures using a protocol like HTTPS. The HTTPS protocol is built on TLS/SSL to encrypt the data that is transmitted and to ensure that user is being sent towards the desired site. Clearly, the fact that traffic is encrypted does not necessarily mean that it's completely safe. The security also depends on the encryption algorithm used and the robustness of the keys that the application is using.

Reference:

http://www.instantssl.com/ssl-certificate-products/https.html
http://webdesign.about.com/od/ecommerce/a/aa070407.htm
http://en.wikipedia.org/wiki/HTTP_Secure
http://searchsoftwarequality.techtarget.com/definition/HTTPS
http://www.chmag.in/article/may2012/https-hyper-text-transfer-protocol-secure

Recommendation:

It is always recommended that, whenever the user sends information to the server, like login credentials and purchase information, the values must be encrypted. The encryption is suggested to be triple layered encryption like triple DES or a three layered combination of MD5, SHA and base 64 hashes. This is because cracking those encrypted data will be surely a hard time for the attacker. Even though it is encrypted, to be in a very safer side and also as the best way for transmitting data through web server, using of SSL/TLS in http traffic is highly recommended.

Proof of concept:

Risk :	High	Status:	Pass	Reference ID:	02
Vulnerability Name:	Testing for User Enumeration and Guessable User Account				

Description:

It is possible to collect a set of valid usernames by interacting with the authentication mechanism of the application

Details:

Often, web applications reveal when a username exists on system, either as a consequence of a misconfiguration or as a design decision. For example, sometimes, when we submit wrong credentials, we receive a message that states that either the username is present on the system or the provided password is wrong. The information obtained can be used by an attacker to gain a list of users on system. This information can be used to attack the web application, for example, through a brute force or default username/password attack. The attacker interacts with the authentication mechanism of the application to understand if sending particular requests causes the application to answer in different manners. This issue exists because the information released from web application or web server when we provide a valid username is different than when we use an invalid one. In some cases, we receive a message that reveals if the provided credentials are wrong because an invalid username or an invalid password was used. Sometimes, we can enumerate the existing users by sending a username and an empty password. If the application is vulnerable, we receive a response message that reveals, directly or indirectly, some information useful for enumerating users.

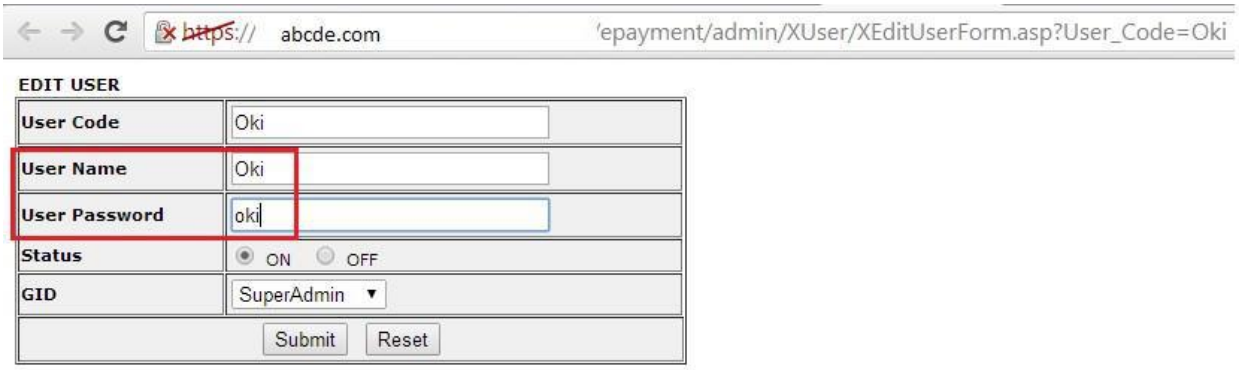
Reference:

http://www.amazon.com/dp/0735617465/?tag=stackoverfl08-20
http://www.steveworkman.com/web-design/2008/best-practice-error-messages/
http://h30499.www3.hp.com/t5/Quality-Center-Support-and-News/Failed-to-Login-Error-message/td-p/5826787
http://stackoverflow.com/questions/117083/error-message-text-best-practices

Recommendation:

Due to over curiosity, the developers set responses for different scenarios like incorrect username, incorrect password and incorrect username & password. It is suggested to provide error message saying “Incorrect login credentials” or other equivalent messages.

Proof of concept:



Risk :	High	Status:	Pass	Reference ID:	03
Vulnerability Name:	Testing for weak password change or reset functionalities				

Description:

Forgot password or password reset function allows the attacker to view the password of the user.

Details:

It is common for an application to have a mechanism that provides a means for a user to gain access to their account in the event they forget their password. Very often the password recovery mechanism is weak, which has the effect of making it more likely that it would be possible for a person other than the legitimate system user to gain access to that user's account.

This weakness may be that the security question is too easy to guess or find an answer to (e.g. because it is too common). Or there might be an implementation weakness in the password recovery mechanism code that may for instance trick the system into e-mailing the new password to an e-mail account other than that of the user. There might be no throttling done on the rate of password resets so that a legitimate user can be denied service by an attacker if an attacker tries to recover their password in a rapid succession. The system may send the original password to the user rather than generating a new temporary password. In summary, password recovery functionality, if not carefully designed and implemented can often become the system's weakest link that can be misused in a way that would allow an attacker to gain unauthorized access to the system. Weak password recovery schemes completely undermine a strong password authentication scheme.

Reference:

http://wordpress.org/extend/plugins/force-strong-passwords/
http://nileshkumar83.blogspot.in/2010/03/weak-password-recovery-mechanism.html
http://chingshiong.blogspot.in/2013/01/facebook-bug-4-password-reset.html
http://threatpost.com/en_us/blogs/facebook-patches-password-reset-vulnerability-010813

Recommendation:

- Make sure that all input supplied by the user to the password recovery mechanism is thoroughly filtered and validated.
- Do not use standard weak security questions and use several security questions.
- Make sure that there is throttling on the number of incorrect answers to a security question. Disable the password recovery functionality after a certain (small) number of incorrect guesses.
- Require that the user properly answers the security question prior to resetting their password and sending the new password to the e-mail address of record.
- Never allow the user to control what e-mail address the new password will be sent to in the password recovery mechanism.
- Assign a new temporary password rather than revealing the original password.

Proof of concept:

Risk :	High	Status:	Pass	Reference ID:	04
Vulnerability Name:	Testing for Bypassing Session Management Schema				

Description:

In order to avoid continuous authentication for each page of a website or service, web applications implement various mechanisms to store and validate credentials for a pre-determined timespan. These mechanisms are known as Session Management and, while they're most important in order to increase the ease of use and user-friendliness of the application, they can be exploited by a penetration tester to gain access to a user account, without the need to provide correct credentials. In this test, we want to check that cookies and other session tokens are created in a secure and unpredictable way. An attacker who is able to predict and forge a weak cookie can easily hijack the sessions of legitimate users.

Details:

Cookies are used to implement session management. In a nutshell, when a user accesses an application which needs to keep track of the actions and identity of that user across multiple requests, a cookie (or more than one) is generated by the server and sent to the client. The client will then send the cookie back to the server in all following connections until the cookie expires or is destroyed. The data stored in the cookie can provide to the server a large spectrum of information about who the user is, what actions he has performed so far, what his preferences are, etc. therefore providing a state to a stateless protocol like HTTP.
A typical example is provided by an online shopping cart. Throughout the session of a user, the application must keep track of his identity, his profile, the products that he has chosen to buy, the quantity, the individual prices, the discounts, etc. Cookies are an efficient way to store and pass this information back and forth (other methods are URL parameters and hidden fields).
Due to the importance of the data that they store, cookies are therefore vital in the overall security of the application. Being able to tamper with cookies may result in hijacking the sessions of legitimate users, gaining higher privileges in an active session, and in general influencing the operations of the application in an unauthorized way.
Usually the main steps of the attack pattern are the following:
cookie collection: collection of a sufficient number of cookie samples;
cookie reverse engineering: analysis of the cookie generation algorithm;
cookie manipulation: forging of a valid cookie in order to perform the attack. This last step might require a large number of attempts, depending on how the cookie is created
Another pattern of attack consists of overflowing a cookie. Here the attempt is amde to overflow a memory area, thereby interfering with the correct behavior of the application and possibly injecting (and remotely executing) malicious code

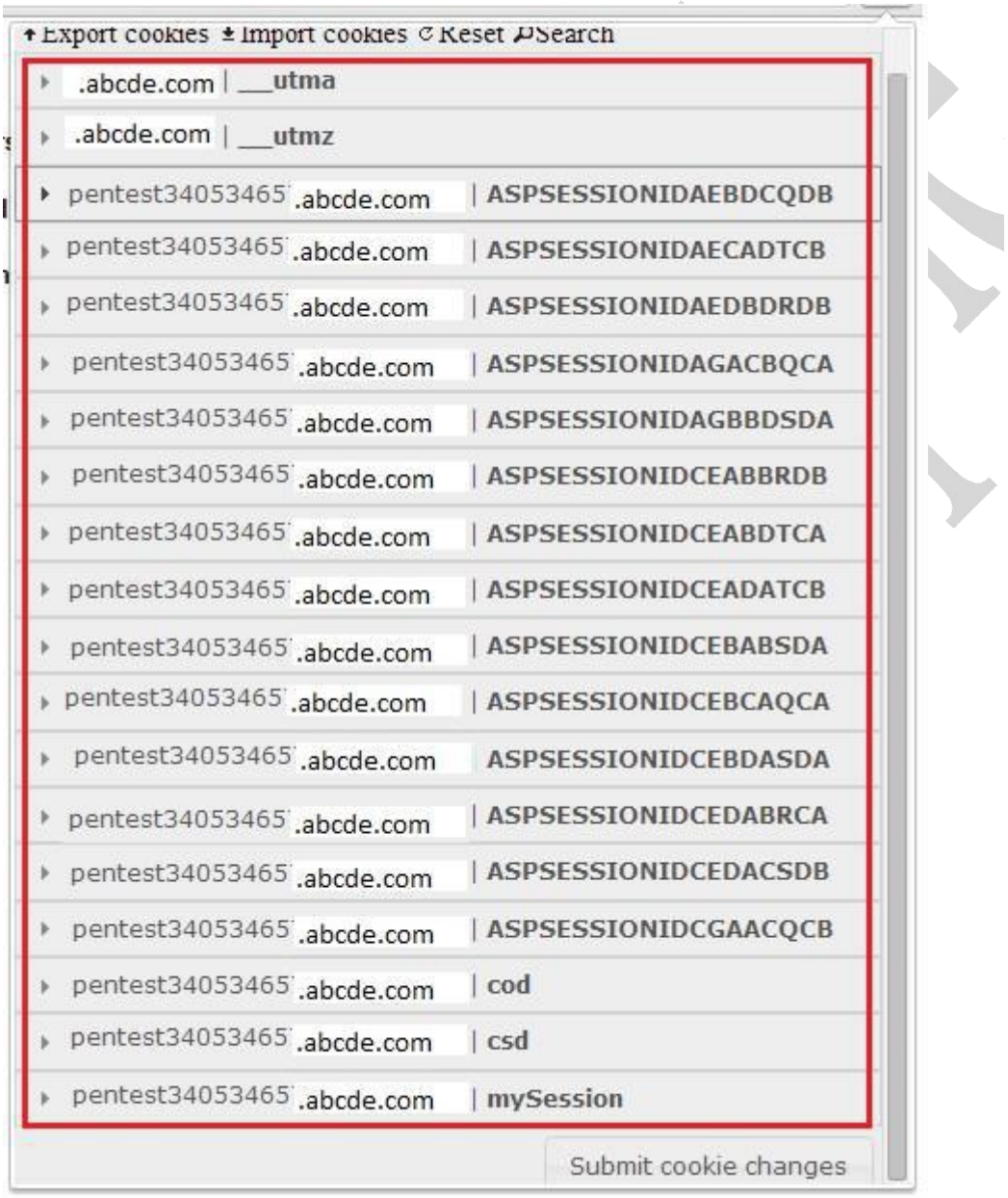
Reference:

http://www.w3schools.com/PHP/php_cookies.asp
http://www.w3schools.com/asp/asp_cookies.asp
http://www.w3schools.com/php/php_sessions.asp
http://www.w3schools.com/asp/asp_sessions.asp
http://wblinks.com/notes/secure-session-management-tips

Recommendation:

Applications should NOT use as variables any user personal information (user name, password, home address, etc.,). Highly protected applications should not implement mechanisms that make automated requests to prevent session timeouts.
Highly protected applications should not implement "remember me" functionality. Highly protected applications should not use URL rewriting to maintain state when cookies are turned off on the client. Applications should NOT use session identifiers for encrypted HTTPS transport that have once been used over HTTP.

Proof of concept:



Risk :	High	Status:	Pass	Reference ID:	05
Vulnerability Name:	Testing for Cross Site Request Forgery (CSRF)				

Description:

CSRF is an attack which forces an end user to execute unwanted actions on a web application in which he/she is currently authenticated. With a little help of social engineering (like sending a link via email/chat), an attacker may force the users of a web application to execute actions of the attacker's choosing. A successful CSRF exploit can compromise end user data and operation, when it targets a normal user. If the targeted end user is the administrator account, a CSRF attack can compromise the entire web application

Details:

Cross-Site Request Forgery (CSRF) is an attack that tricks the victim into loading a page that contains a malicious request. It is malicious in the sense that it inherits the identity and privileges of the victim to perform an undesired function on the victim's behalf, like change the victim's e-mail address, home address, or password, or purchase something. CSRF attacks generally target functions that cause a state change on the server but can also be used to access sensitive data. For most sites, browsers will automatically include with such requests any credentials associated with the site, such as the user's session cookie, basic auth credentials, IP address, Windows domain credentials, etc. Therefore, if the user is currently authenticated to the site, the site will have no way to distinguish this from a legitimate user request. Synonyms: CSRF attacks are also known by a number of other names, including XSRF, "Sea Surf", Session Riding, Cross-Site Reference Forgery, and Hostile Linking. Microsoft refers to this type of attack as a One-Click attack in their threat modeling process.

Reference:

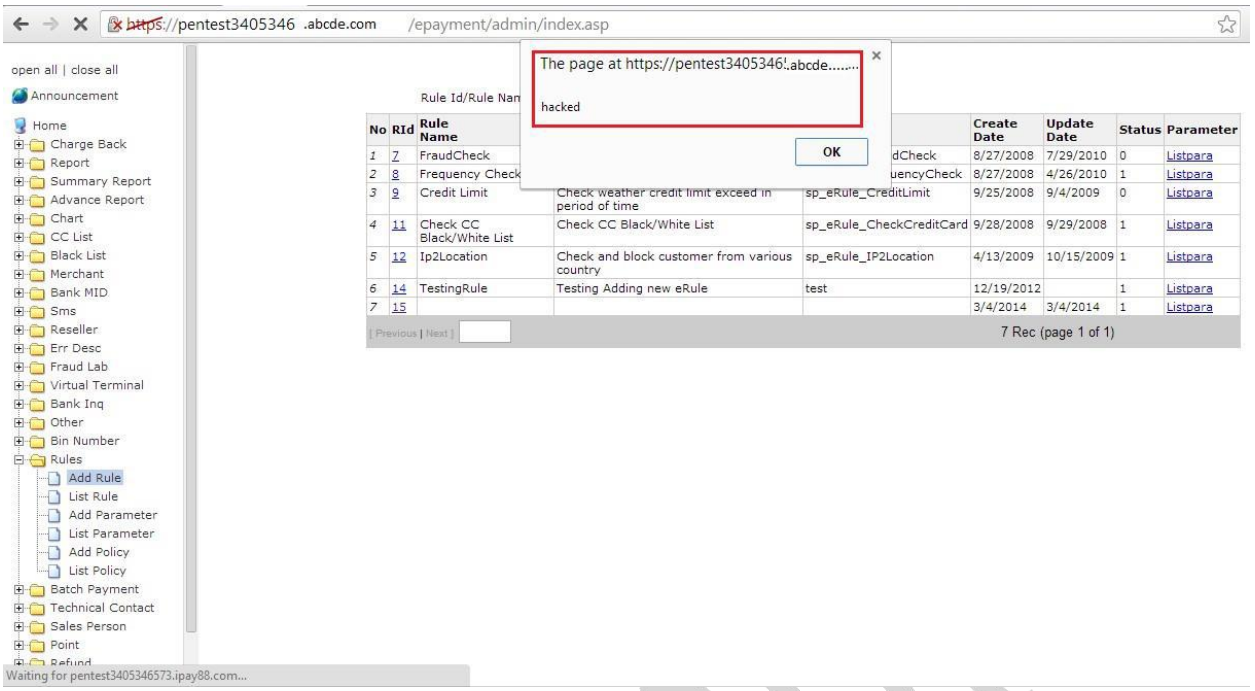
http://www.cgisecurity.com/articles/csrf-faq.shtml
https://www.owasp.org/index.php/File:RequestRodeo-MartinJohns.pdf
https://www.owasp.org/index.php/Category:OWASP_CSRFGuard_Project
https://code.google.com/p/pinata-csrf-tool/
http://yehg.net/lab/pr0js/view.php/A_Most-Neglected_Fact_About_CSRF.pdf

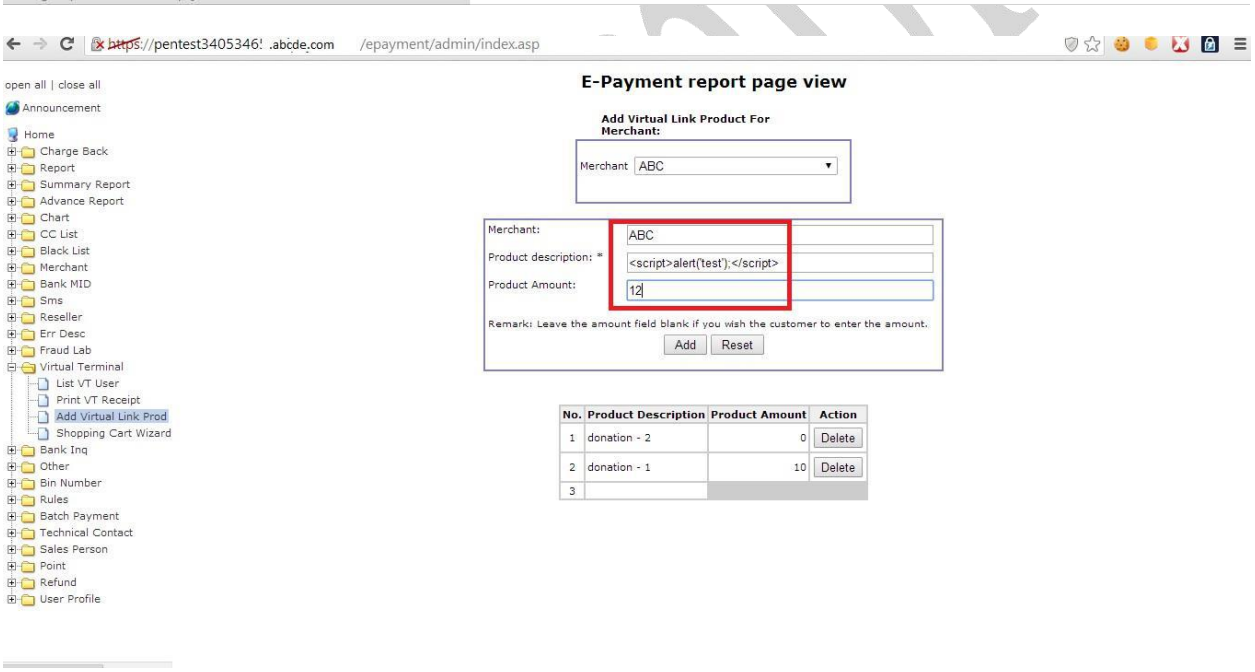
Recommendation:

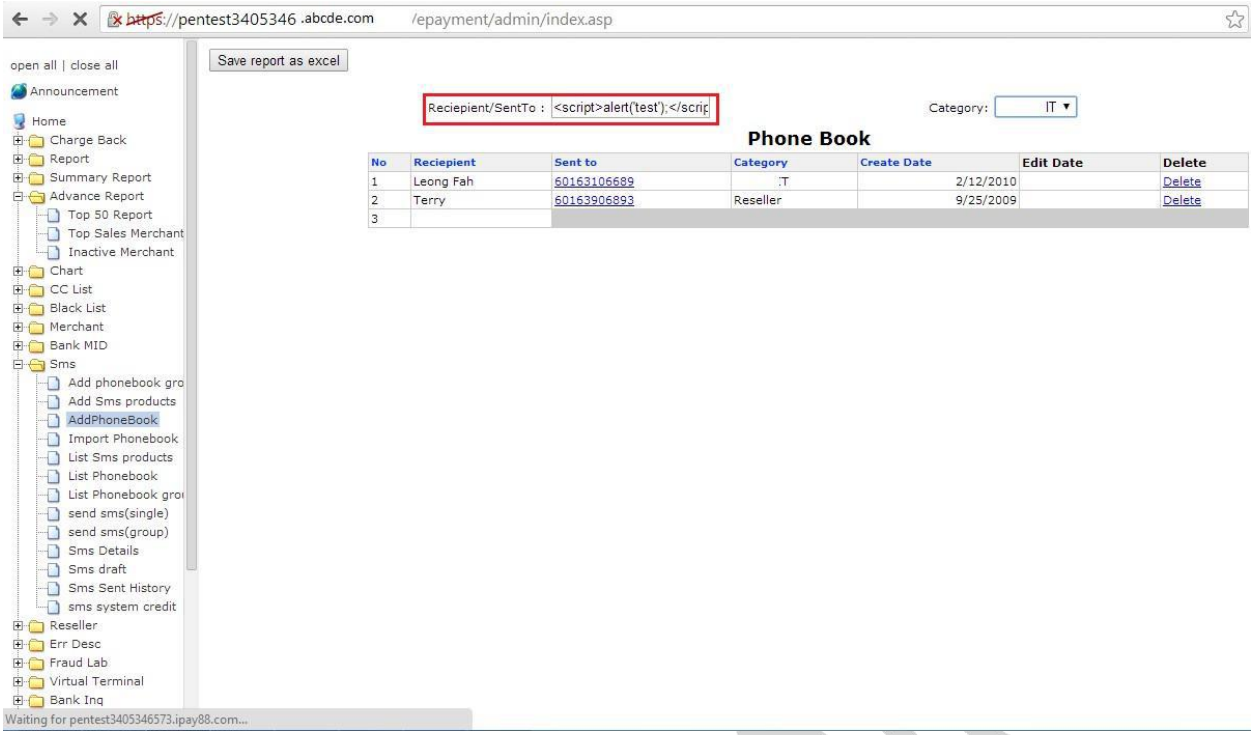
• Add a per-request nonce to URL and all forms in addition to the standard session. This is also referred to as "form keys". Many frameworks (ex, Drupal.org 4.7.4+) either have or are starting to include this type of protection "built-in" to every form so the programmer does not need to code this protection manually.

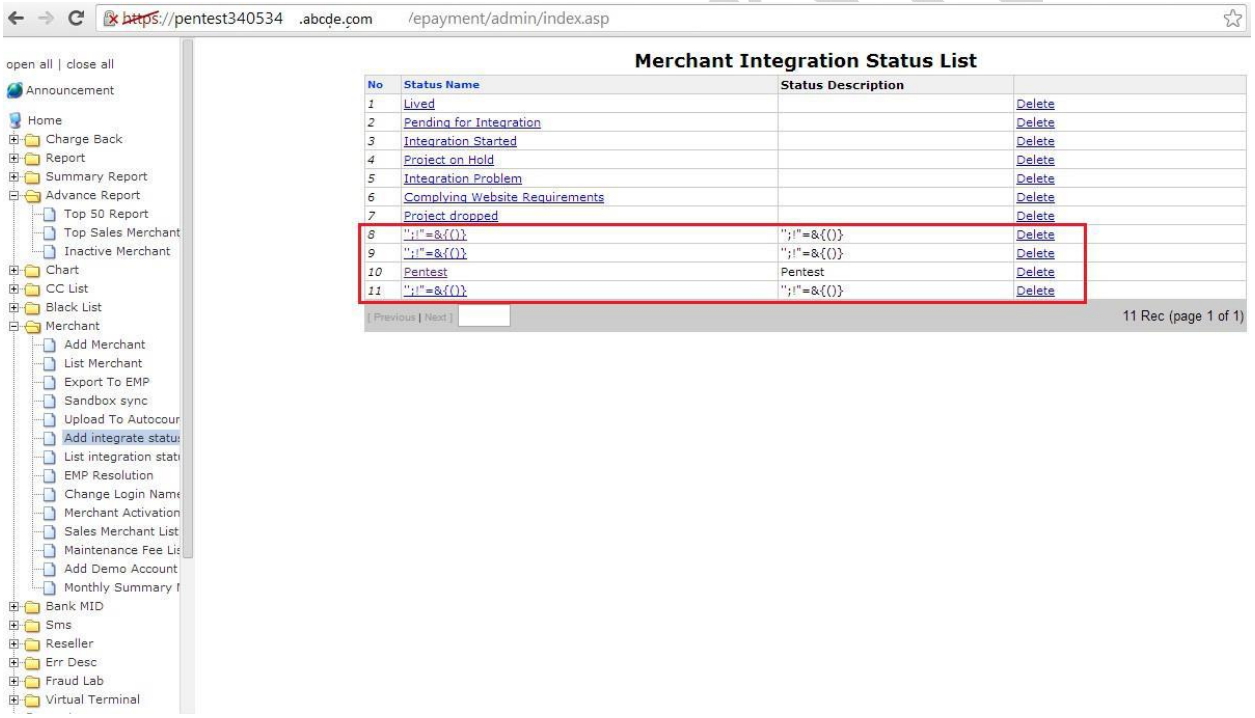
• Checking the referrer in the client's HTTP request will prevent CSRF attacks. By ensuring the HTTP request have come from the original site means that the attacks from other sites will not function. It is very common to see referrer checks used on embedded network hardware due to memory limitations. XSS can be used to bypass both referrer and token based checks simultaneously. For instance the Sammy Worm used an XHR to obtain the CSRF token to forge requests.
• "Although cross-site request forgery is fundamentally a problem with the web application, not the user, users can help protect their accounts at poorly designed sites by logging off the site before visiting another, or clearing their browser's cookies at the end of each browser session."

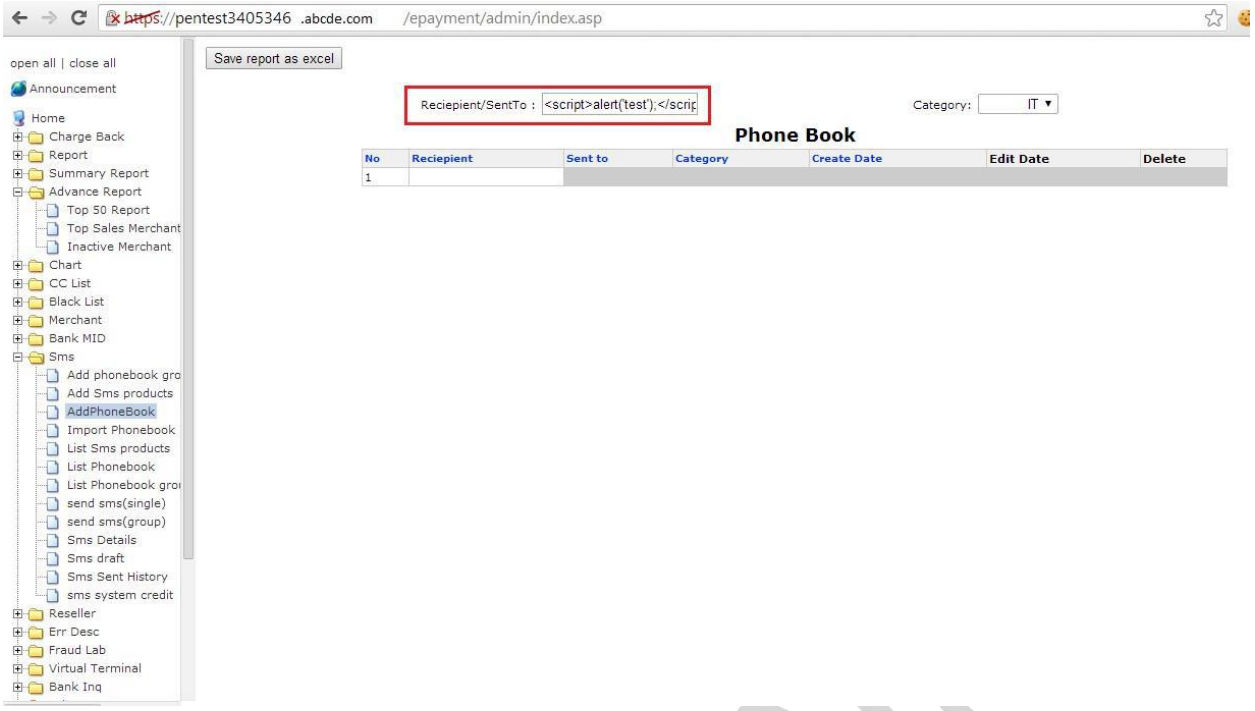
Proof of concept:











Risk :	High	Status:	Pass	Reference ID:	06
Vulnerability Name:	Testing for Stored Cross Site Scripting				

Description:

It is possible to perform Stored Cross Site Scripting (XSS), which has potentially high level threat which stores data in the database.

Details:

Stored XSS occurs when a web application gathers input from a user which might be malicious, and then stores that input in a data store for later use. The input that is stored is not correctly filtered. As a consequence, the malicious data will appear to be part of the web site and run within the user's browser under the privileges of the web application. Since this vulnerability typically involves at least two requests to the application, this may also called second-order XSS.

This vulnerability can be used to conduct a number of browser-based attacks including:
- Hijacking another user's browser
- Capturing sensitive information viewed by application users
- Pseudo defacement of the application
- Port scanning of internal hosts
- Directed delivery of browser-based exploits

Other malicious activities
- Attacker stores malicious code into the vulnerable page
- User authenticates in the application
- User visits vulnerable page
- Malicious code is executed by the user's browser

Stored XSS is particularly dangerous in application areas where users with high privileges have access. When the administrator visits the vulnerable page, the attack is automatically executed by their browser. This might expose sensitive information such as session authorization tokens.

Reference:

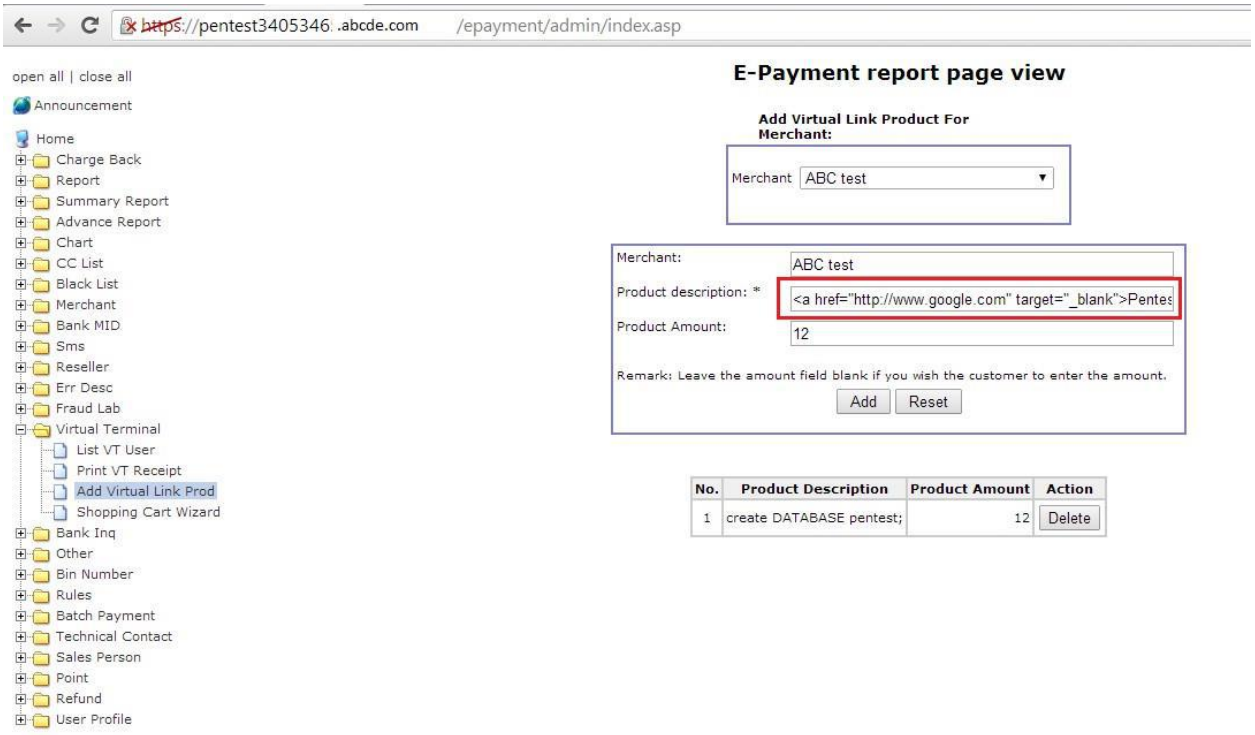
http://en.wikipedia.org/wiki/Cross-site_scripting
http://seclists.org/bugtraq/2013/Feb/84
http://deadlytechnology.com/web-development/xss/

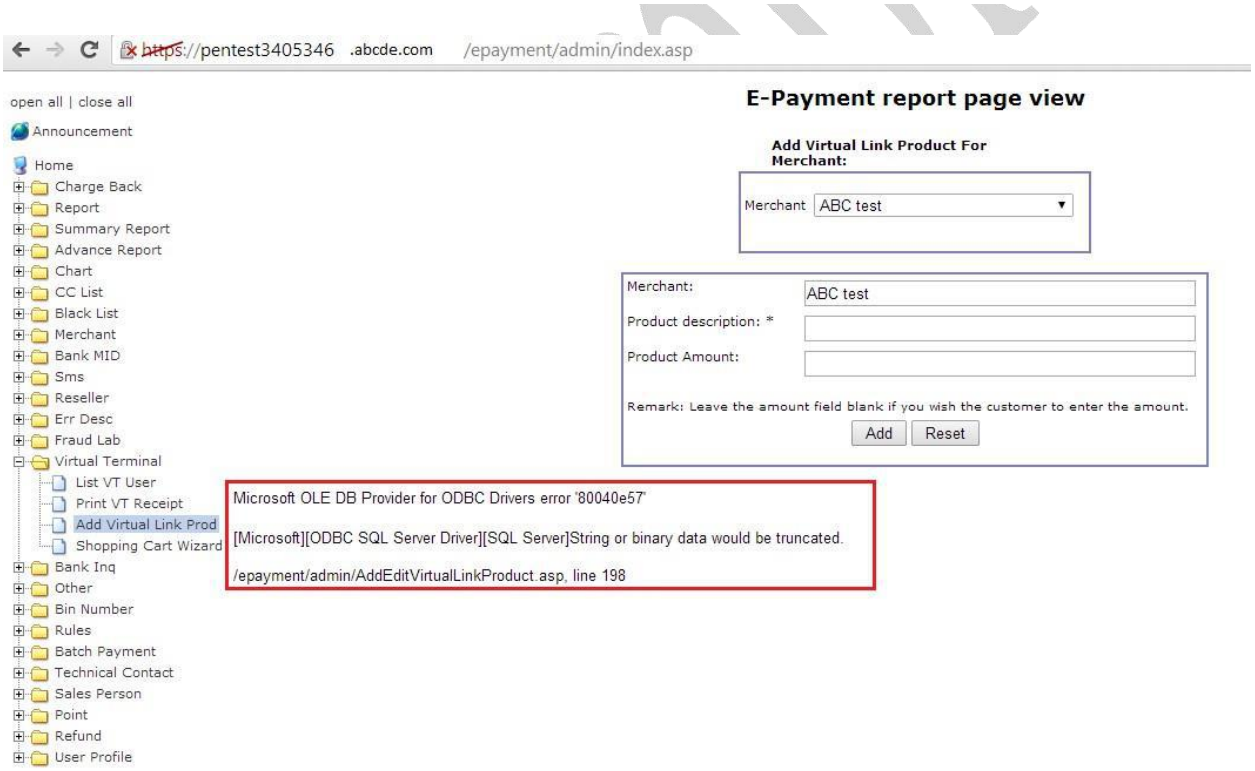
Recommendation:

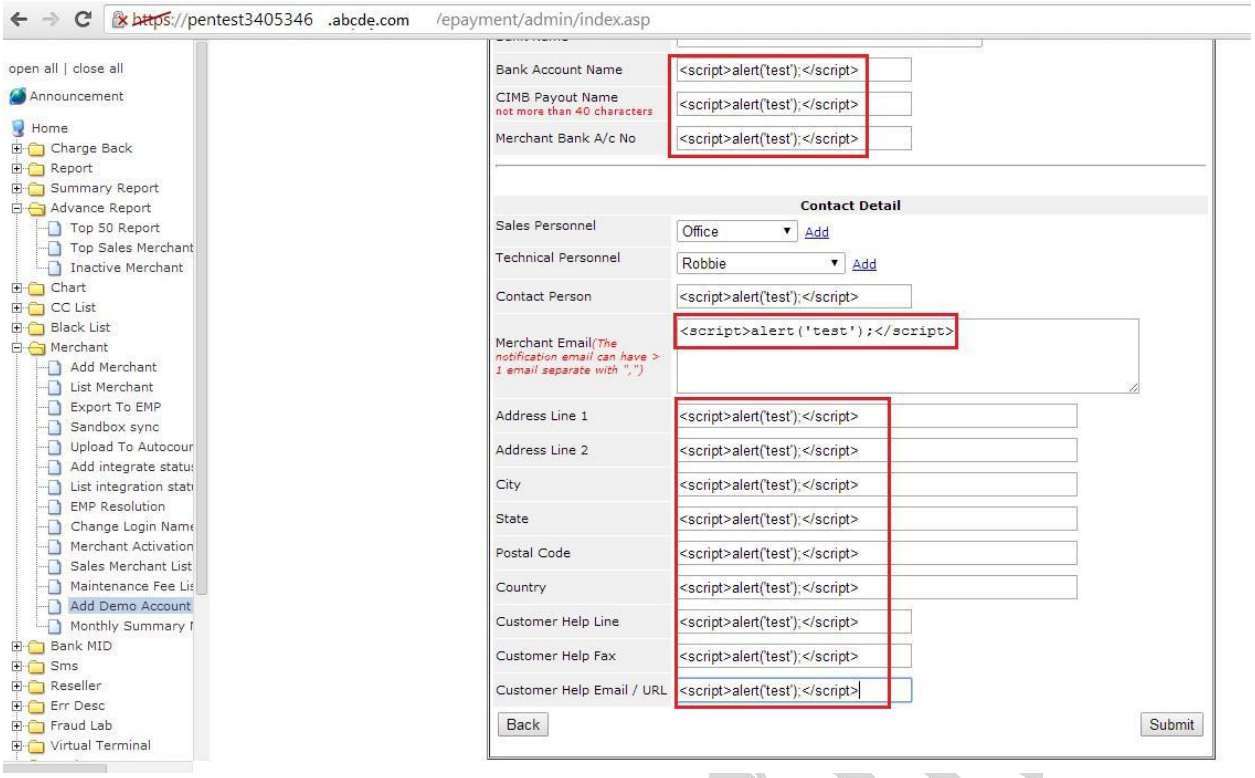
XSS can only be prevented by carefully sanitizing all input which is not known to be secure. Classes of input which is known NOT to be secure include:
- GET parameters
- POST parameters
- window.location
- document.referrer
- document.location
- document.URLUnencoded
- Cookie data

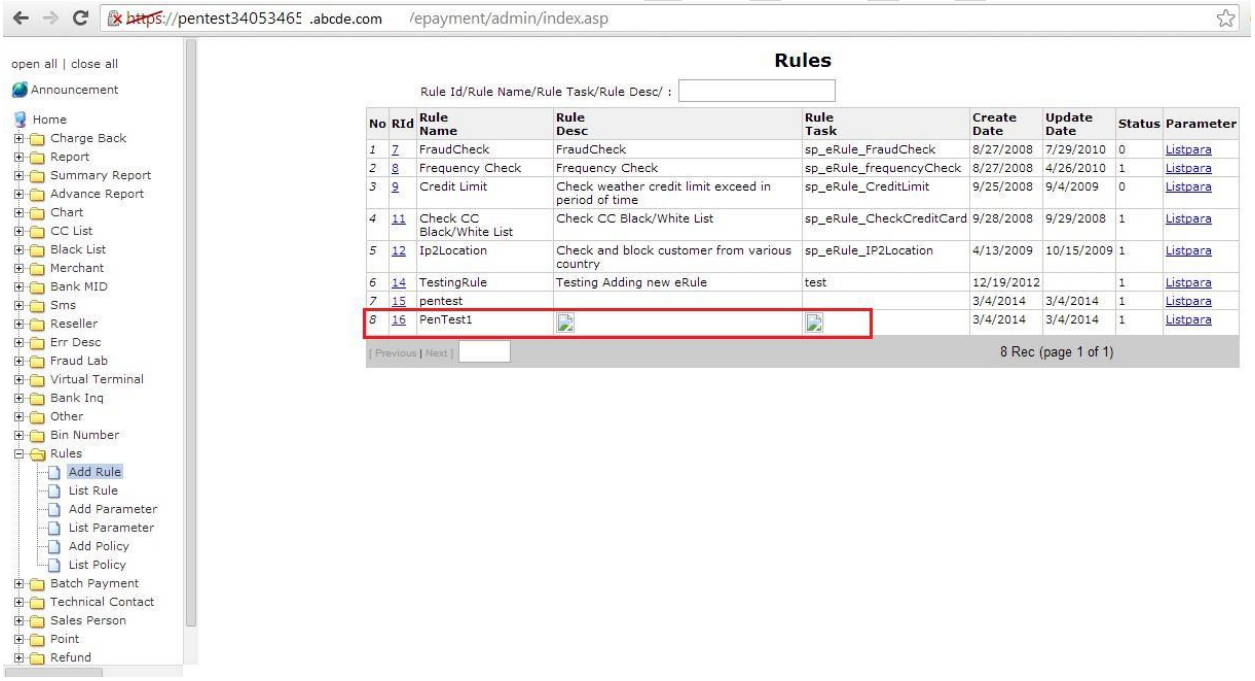
- Potentially data from your own database

Proof of concept:









Risk :	High	Status:	Pass	Reference ID:	07
Vulnerability Name:	Testing for SQL Injection				

Description:

SQL injection vulnerability is found in the application, which is considered as the most potential attack vector, since it can be used and database values are retrieved.

Details:

SQL Injection vulnerabilities occur whenever input is used in the construction of a SQL query without being adequately constrained or sanitized. The use of dynamic SQL (the construction of SQL queries by concatenation of strings) opens the door to these vulnerabilities. SQL injection allows an attacker to access the SQL servers. It allows for the execution of SQL code under the privileges of the user used to connect to the database. A SQL injection attack consists of insertion or "injection" of either a partial or complete SQL query via the data input or transmitted from the client (browser) to the web application. A successful SQL injection attack can read sensitive data from the database, modify database data (insert/update/delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file existing on the DBMS file system or write files into the file system, and, in some cases, issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to affect the execution of predefined SQL commands.

Reference:

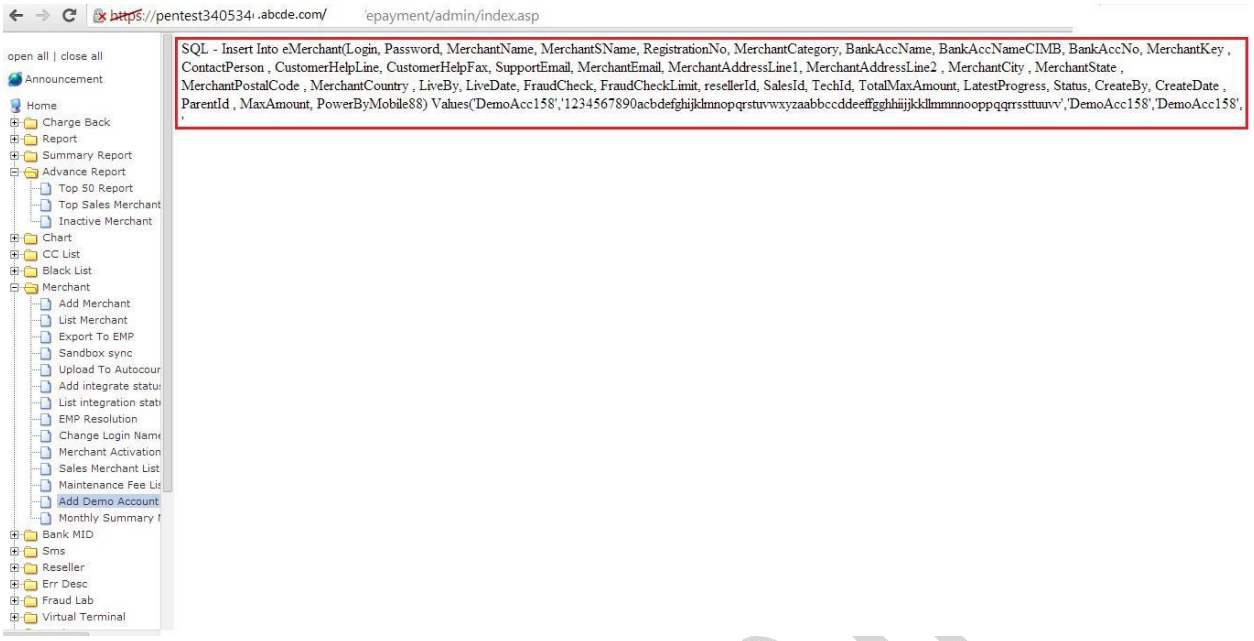
http://en.wikipedia.org/wiki/SQL_injection
http://pastebin.com/ruDvYW7u

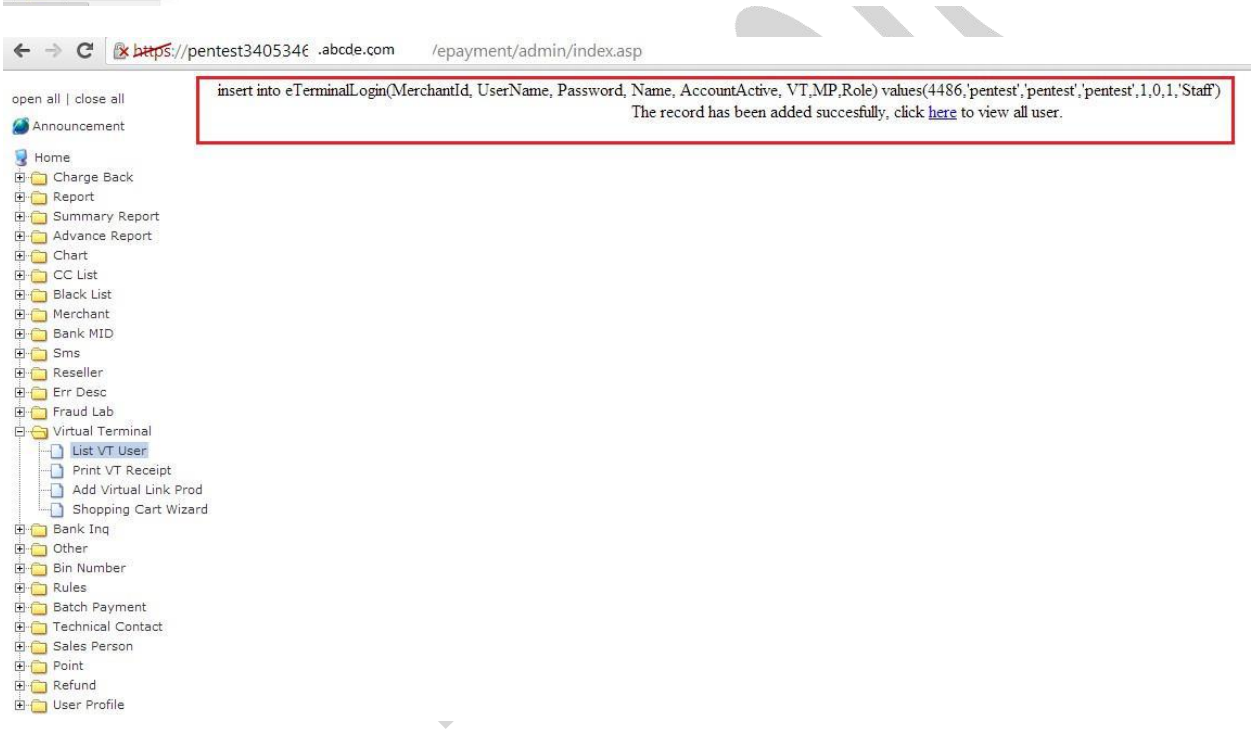
Recommendation:

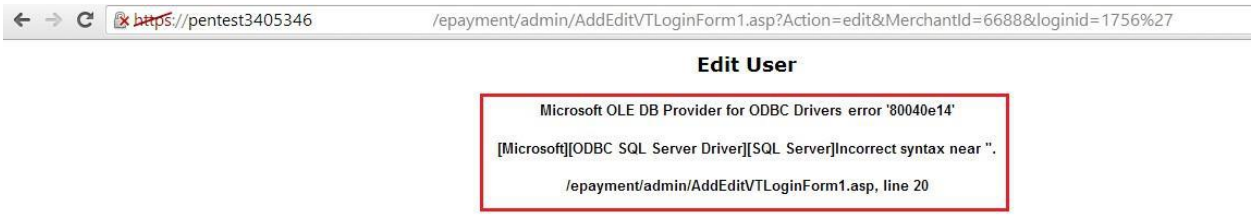
SQL injection can be prevented using the following methods.

- Use dynamic SQL only if absolutely necessary.
- Escape user input.
- Assume magic quotes is always off.
- Install patches regularly and timely.
- Remove all functionality you don't use.

Proof of concept:







https://pentest3405346 .abcde.com /epayment/admin/AddEditVTLoginForm1.asp?Action=edit&MerchantId=6688&loginid=1756having%201=1--

Edit User

Microsoft OLE DB Provider for ODBC Drivers error '80040e14'

[Microsoft][ODBC SQL Server Driver][SQL Server]Column 'eTerminalLogin.MerchantId' is invalid in the HAVING clause because it is not contained in either an aggregate function or the GROUP BY clause.

/epayment/admin/AddEditVTLoginForm1.asp, line 20

idmin/AddEditVTLoginForm1.asp?Action=edit&MerchantId=6688&loginid=1756%20group%20by%20eTerminalLogin.MerchantId%20having%201=1--

Edit User

Microsoft OLE DB Provider for ODBC Drivers error '80040e14'

[Microsoft][ODBC SQL Server Driver][SQL Server]Column 'eTerminalLogin.LoginId' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.

/epayment/admin/AddEditVTLoginForm1.asp, line 20

orm1.asp?Action=edit&MerchantId=6688&loginid=1756%20group%20by%20eTerminalLogin.MerchantId,eTerminalLogin.LoginId%20having%201=1--

Edit User

Microsoft OLE DB Provider for ODBC Drivers error '80040e14'

[Microsoft][ODBC SQL Server Driver][SQL Server]Column 'eTerminalLogin.UserName' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.

/epayment/admin/AddEditVTLoginForm1.asp, line 20

rchantId=6688&loginid=1756%20group%20by%20eTerminalLogin.MerchantId,eTerminalLogin.LoginId,eTerminalLogin.UserName%20having%201=1--

Edit User

Microsoft OLE DB Provider for ODBC Drivers error '80040e14'

[Microsoft][ODBC SQL Server Driver][SQL Server]Column 'eTerminalLogin.Password' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.

/epayment/admin/AddEditVTLoginForm1.asp, line 20

756%20group%20by%20eTerminalLogin.MerchantId,eTerminalLogin.LoginId,eTerminalLogin.UserName,eTerminalLogin.Password%20having%201=1--

Edit User

Microsoft OLE DB Provider for ODBC Drivers error '80040e14'

[Microsoft][ODBC SQL Server Driver][SQL Server]Column 'eTerminalLogin.Name' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.

/epayment/admin/AddEditVTLoginForm1.asp, line 20

https://pentest3405346 .abcde.com /epayment/admin/AddEditVTLoginForm1.asp?Action=edit&MerchantId=6688&loginid=1756%20group%20by%

Edit User

Microsoft OLE DB Provider for ODBC Drivers error '80040e14'

[Microsoft][ODBC SQL Server Driver][SQL Server]Column 'eTerminalLogin.AccountActive' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.

/epayment/admin/AddEditVTLoginForm1.asp, line 20

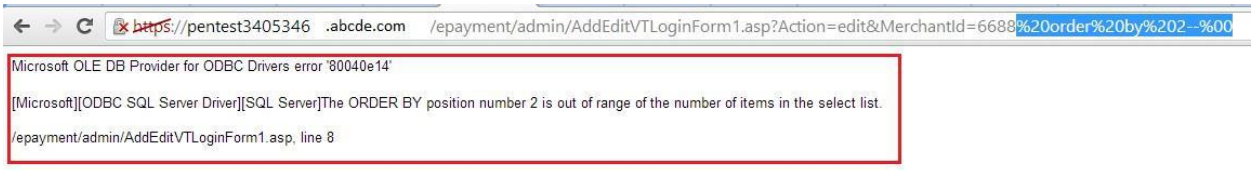
eTerminalLogin.LoginId,eTerminalLogin.UserName,eTerminalLogin.Password,eTerminalLogin.Name,eTerminalLogin.AccountActive%20having%201=1--

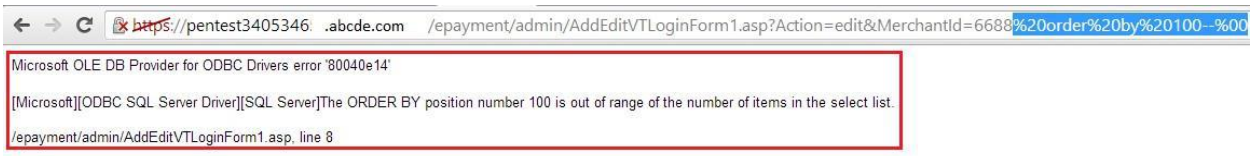
Edit User

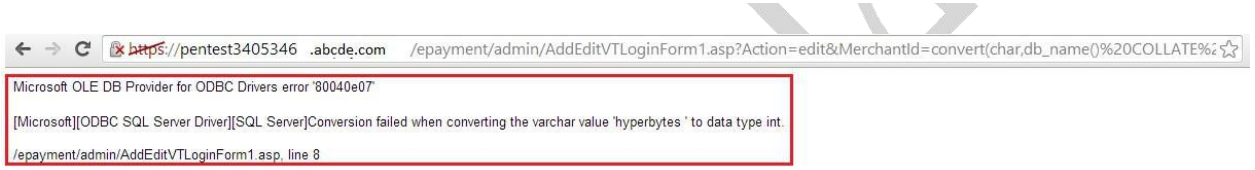
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'

[Microsoft][ODBC SQL Server Driver][SQL Server]Column 'eTerminalLogin.VT' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.

/epayment/admin/AddEditVTLoginForm1.asp, line 20







Risk :	High	Status:	Pass	Reference ID:	08
Vulnerability Name:	Testing for Buffer overflow				
Description:					
A buffer overflow condition exists when a program attempts to put more data in a buffer than it can hold or when a program attempts to put data in a memory area past a buffer. In this case, a buffer is a sequential section of					

memory allocated to contain anything from a character string to an array of integers.

Details:

Buffer overflow is probably the best known form of software security vulnerability. Most software developers know what a buffer overflow vulnerability is, but buffer overflow attacks against both legacy and newly-developed applications are still quite common. Part of the problem is due to the wide variety of ways buffer overflows can occur, and part is due to the error-prone techniques often used to prevent them.

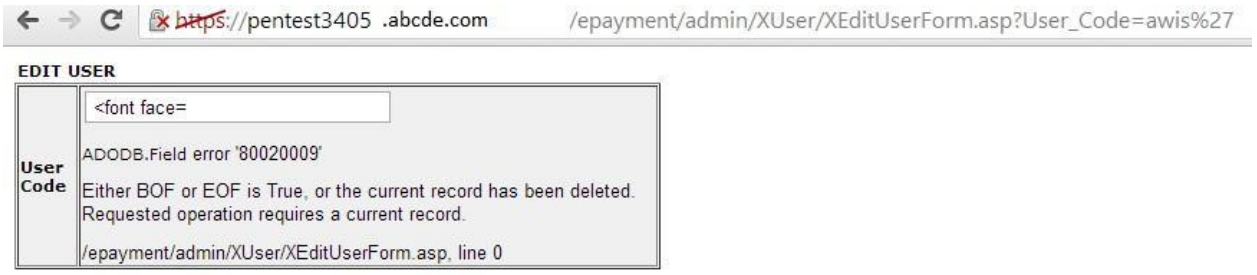
Buffer overflows are not easy to discover and even when one is discovered, it is generally extremely difficult to exploit. Nevertheless, attackers have managed to identify buffer overflows in a staggering array of products and components.

In a classic buffer overflow exploit, the attacker sends data to a program, which it stores in an undersized stack buffer. The result is that information on the call stack is overwritten, including the function's return pointer. The data sets the value of the return pointer so that when the function returns, it transfers control to malicious code contained in the attacker's data.

Recommendation:

Keep up with the latest bug reports for your web and application server products and other products in your Internet infrastructure. Apply the latest patches to these products. Periodically scan your web site with one or more of the commonly available scanners that look for buffer overflow flaws in your server products and your custom web applications. For your custom application code, you need to review all code that accepts input from users via the HTTP request and ensure that it provides appropriate size checking on all such inputs. This should be done even for environments that are not susceptible to such attacks as overly large inputs that are uncaught may still cause denial of service or other operational problems.

Proof of concept:



Risk :	Medium	Status:	Pass	Reference ID:	09
Vulnerability Name:	Search Engine Discovery/Reconnaissance				

Description:

It is possible to discover sensitive information through search engines search and passive reconnaissance.

Details:

Once the Google Bot has completed crawling, it commences indexing the web page based on tags and associated attributes, such as <TITLE>, in order to return the relevant search results. Once the Google Bot has completed crawling, it commences indexing the web page based on tags and associated attributes, such as <TITLE>, in order to return the relevant search results. If the robots.txt file is not updated during the lifetime of the web site, then it is possible for web content not intended to be included in Google's Search Results to be returned.

Reference:

http://rusecure.rutgers.edu/category/topic/search-engine-reconnaissance
http://www.google.com/support/webmasters/bin/answer.py?answer=70897
http://www.google.com/help/operators.html
http://code.google.com/apis/soapsearch/reference.html#1_2
http://www.google.com/support/webmasters/bin/topic.py?topic=8459
http://support.google.com/webmasters/bin/answer.py?hl=en&answer=1663691

Recommendation:

Restricting the search engines to certain sensitive folders can be done using robots.txt. Robots.txt can be configured using the following scripts
User-agent: *
Allow: /allowed_folder/
Disallow: /restricted_folder/
Even if it is done, it must be available in Google Cache, on late update of robots.txt. Therefore, it must be removed from the Google Cache. The cached pages can be removed from Google Cache using n number of tools like webmaster tools (Google public URL removal tool) from Google.

Proof of concept:

Registrant Information	
Registrant Name	PDR LTD. D/B/A PUBLICDOMAINREGISTRY.COM
Registry Domain ID	6275432502_DOMAIN_COM-VRSN
Registrar IANA ID	334
Registrar Abuse Contact Email: Email Masking	Image@publicdomainregistry.com
Registrar Abuse Contact Phone	-2013567751
Registry Registrant ID	DI_3456324
Registrant Name	
Registrant Organisation	MobileSampleCompany.com
Registrant Street	
Registrant City	Kuala Lumpur
Registrant State/Province	Wilayah Kuala Lumpur
Registrant Postal Code	55244
Registrant Country	MY
Registrant Phone	603.95678956
Registrant Email: Email Masking	Image@mobileSampleCompany.com
Registry Admin ID	DI_41354674324

SampleCompany.com	
Ip Address of pentest3405346.SampleCompany.com	4.17.24.1
Ip Addresses of SampleCompany.com	4.17.24.1, 4.17.24.2, 4.17.24.3, 4.17.24.4
IP address	4.17.24.1
Country	SG
State/Province	SINGAPORE
City	SINGAPORE
Zip or postal code	-
Latitude	1.26378
Longitude	103.111
Timezone	+08:00
Hostname	ec2-46-137-220-142.ap-southeast-1.compute.s.com
Web Server	IIS 7.5
System Details	Microsoft-HTTP API/2.0
Server technologies	Microsoft ASP.Net
Operating System	Microsoft Windows Server 2008 R2
HTTP version used	1.1

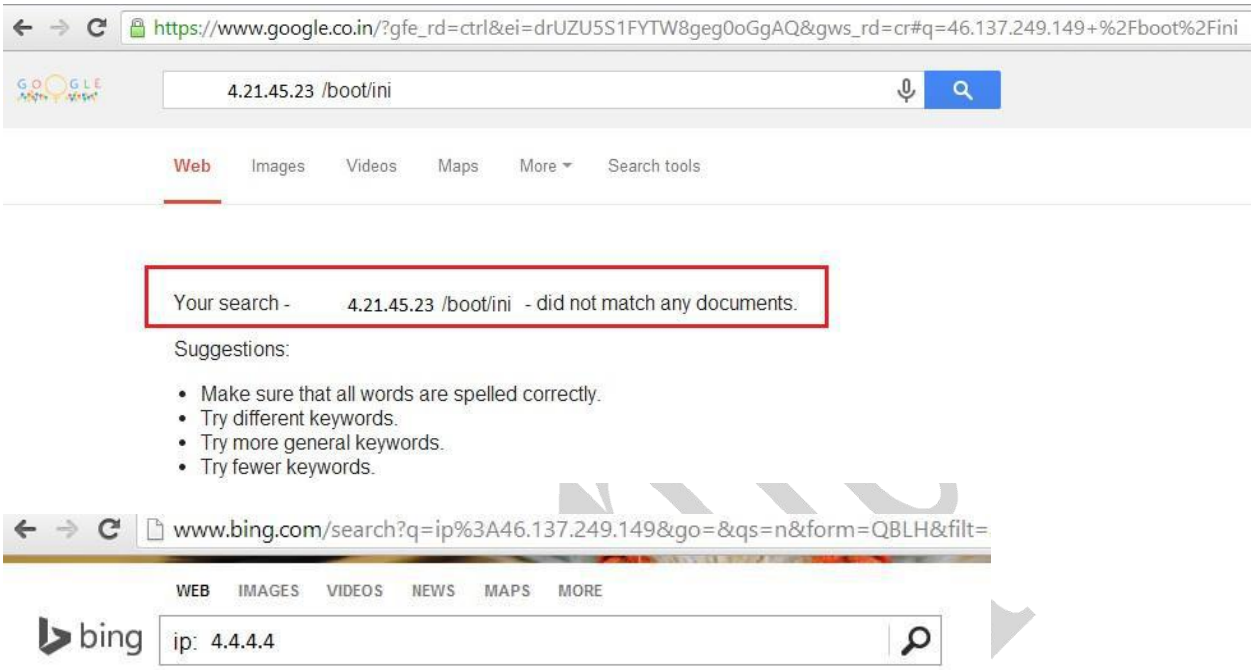
Name Servers	
ns-121.awsdns-15.com	25.21.12.11
ns-1453.awsdns-53.org	25.21.12.11
ns-1684.awsdns-18.co.uk	25.21.12.11
ns-831.awsdns-39.net	25.21.12.11
Expires on	10-Oct-16
Registered on	10-Oct-06
Updated on	10-Aug-11
Sub-Domain	
blog.SampleCompany.com	4.1.24.5
dvl3. SampleCompany.com	21.2.20.2

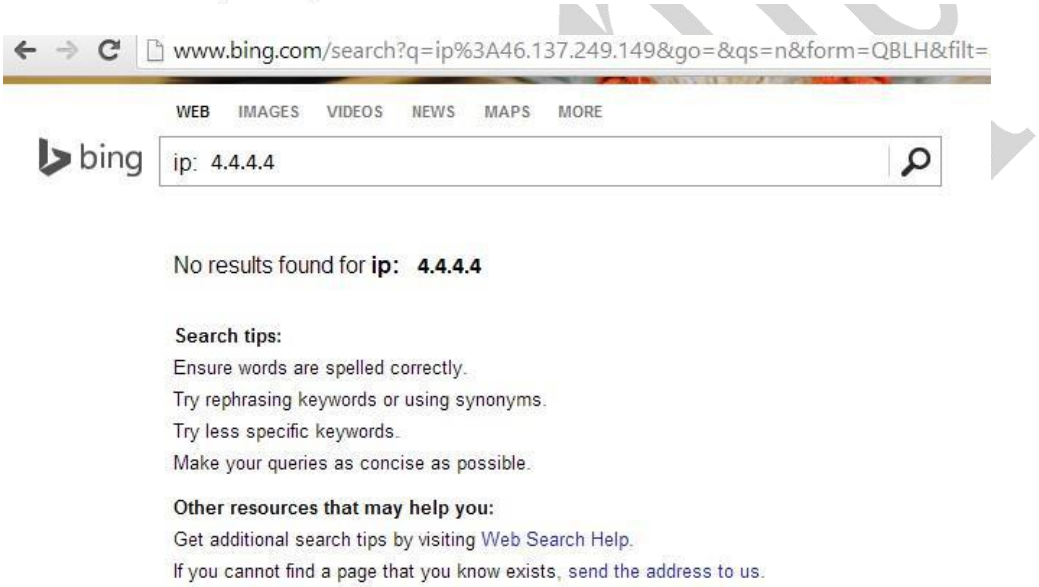
SOA Record – SampleCompany.com	
Name Server	ns-1684.awsdns-18.co.uk
Email	Email Masking Image@amazon.com
Serial Number	1
Refresh	2 hours
Retry	15 minutes
Expiry	14 days
Minimum	1 day

HTTP Request Headers	
Host	SampleCompany.com
Accept	*/*
Cache-Control	no-cache
Connection	keep-alive
Accept-Encoding	gzip,deflate

HTTP Response Headers	
Server	Dungeon9
Date	Mon, 03 Mar 2014 06:51:01 GMT
Content-Type	text/html
Transfer-Encoding	chunked
Connection	keep-alive
Keep-Alive	timeout=600
Vary	Accept-Encoding
Cache-Control	max-age=1800

Set-Cookie	ASPSESSIONIDSCAQTQRA=MHDPFEACAEEMHBFIOEOHPALC; path=/
Set-Cookie	AWSELB=D711A57F12C5D33D241A23D20C225834B6664BC8153E48DFB60248FCE5A2B30BA2D3DD1417D6518BF4B684210682C0BC7952F9867EBBCE3854BCA1F1804367D0E7D882462E;PATH=/;MAX-AGE=7200
X-Powered-By	ASP.NET
Expires	Mon, 03 Mar 2014 07:21:01 GMT
Content-Encoding	Gzip





Risk :	Medium	Status:	Pass	Reference ID:	10
Vulnerability Name:	Identify application entry points				

Description:

Some interesting application entry points can tempt the attacker with information about where to start the attack.

Details:

The input fields can be the following three. Any attacks can be initiated from any one of the three application entry points. They are GET, POST and html tags. The GET and POST methods are used to transfer any information from one web page to the other. The GET method is usually used to get information from the web page, which will be seen in the URL. The POST method is usually used to get information from the form to a web page or self. The main difference between GET and POST is that, GET is visible in the URL and POST is not. However both the GET and POST can be viewed. This GET and POST can be used to get information about the application entry points. The third method which is the entry point through analyzing HTML tags. HTML tags like <input>, <select>, <options> are used to get inputs from the user. So these are attracted by attacker. Also the input tag with hidden field always contains sensitive information. So these are analyzed to gather information about the application entry points.

Reference:

http://social.msdn.microsoft.com/Forums/en-US/sharepointdevelopment/thread/75415586-502d-475c-b2ab-d6df97ae4c17
http://www.w3schools.com/tags/ref_httpmethods.asp
http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html
http://www.w3.org/2001/tag/doc/whenToUseGet-20040321

Recommendation:

Use GET if the interaction is more like a question (i.e., it is a safe operation such as a query, read operation, or lookup). Use POST if the interaction is more like an order, or the interaction changes the state of the resource in a way that the user would perceive (e.g., a subscription to a service), or the user be held accountable for the results of the interaction. You should never change anything in your database (other than logging information or other ephemeral data) from a GET request. The issue is that there is various web spidering software, web accelerators, anti-virus programs, and the like, that will perform a GET request on every URL they find; you would not want them to delete items automatically when they do so. GET is also vulnerable to cross-site request forgery; if an attacker makes one of your users click on a link that performs a bad action (for instance, creating a tinyurl that redirects to a delete URL), then they can trick the user into using their permissions to delete something without realizing it. Making a field "hidden" has pretty much nothing to do with security, and should be considered a UI decision. Any "hacker" will read your HTML source anyway. Better to either not show sensitive information at all, or, if you must, to use SSL (to prevent data interception by network intermediaries) and some combination of login challenges (to prevent unauthorized access).

Proof of concept:

A	Possible Sensitive Directories
1	/epayment/admin
2	/epayment/Admin
3	/epayment/ADMIN
4	/epayment/inc
5	/epayment/include
6	/epayment/testing
B	**Possible Sensitive Files**
1	/epayment/test.asp

Risk :	Medium	Status:	Pass	Reference ID:	11
Vulnerability Name:	Testing for Web Application Fingerprint				

Description:

Knowing the version and type of a running web server allows attackers to determine known vulnerabilities and the appropriate exploits to use during attack.

Details:

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, different versions react same to all HTTP commands.

Reference:

http://pentestlab.wordpress.com/2012/08/01/web-application-fingerprinting/
http://resources.infosecinstitute.com/prototype-model-web-application-fingerprinting/
http://www.quickonlinetips.com/archives/2012/05/turn-off-server-signature/
http://www.unixmen.com/how-to-disable-server-signature-using-htaccess-or-by-editing-apache/
http://www.port80software.com/support/articles/maskyourwebserver
http://httpd.apache.org/docs/2.2/mod/core.html#serversignature

Recommendation:

Most Web servers politely identify themselves and the OS to anyone who asks. Using a network query tool like free ieHTTPHeaders or this Header Check, you can discern the HTTP Server header. Just request a Web site's home page and examine the resulting HTTP headers or "banners" sent back by the server. Among them, you will likely find something like this:

```
Server: Microsoft-IIS/5.0
```

You can remove or obscure this HTTP Server header in a variety of ways, depending on your platform. Apache 2.x users who have the mod_headers module loaded can use a simple directive in their httpd.conf file, as follows:

```
Header set Server "New Server Name Goes Here"
```

Proof of concept:

Risk :	Medium	Status:	Pass	Reference ID:	12
Vulnerability Name:	Application Discovery				

Description:

Finding the applications used in the web server may lead the attacker to a specific approach in compromising the system.

Details:

Many applications have known vulnerabilities and known attack strategies that can be exploited in order to gain remote control or to exploit data. In addition, many applications are often misconfigured or not updated, due to the perception that they are only used "internally" and therefore no threat exists. Unpatched application will always lead to existance of vulnerabilities. With the proliferation of virtual web servers, the traditional 1:1-type relationship between an IP address and a web server is losing much of its original significance. It is not uncommon to have multiple web sites / applications whose symbolic names resolve to the same IP address.

Reference:

http://dcid.me/texts/fingerprinting-web-apps.html
http://resources.infosecinstitute.com/prototype-model-web-application-fingerprinting/
http://www.openbsd.org/faq/pf/
http://www.openbsd.org/faq/pf/config.html
https://calomel.org/pf_config.html
http://en.wikipedia.org/wiki/PF_(firewall)

Recommendation:

It is possible to address specific issues and disable specific types of known fingerprinting software by determining what parameter it relies on most and then changing it. For example, certain packet-filtering solutions, such as pf in OpenBSD, provide a packet normalization service that ensures that all outgoing traffic "looks the same." Although this might prevent some aspects of fingerprinting to some degree or might simply make fingerprinting more difficult by rendering some popular programs less accurate, it does not solve the problem completely.

Proof of concept:

Application Discovery	
Web Server	IIS 7.5
System Details	Microsoft-HTTP API/2.0
Server technologies	Microsoft ASP.Net
Operating System	Microsoft Windows Server 2008 R2

Risk :	Medium	Status:	Pass	Reference ID:	13
Vulnerability Name:	Testing for Weak SSL/TSL Ciphers, Insufficient Transport Layer Protection				

Description:

Insufficient Transport layer protection is found due to weak SSL/TLS ciphers.

Details:

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.

Reference:

http://www.stardothosting.com/blog/2009/05/testing-for-weak-ssl-ciphers-for-security-audits/
http://www.plynt.com/blog/2007/12/enforcing-strong-ssltls-cipher/
http://www.sslshopper.com/article-how-to-disable-weak-ciphers-and-ssl-2.0-in-apache.html
http://www.rapid7.com/vulndb/lookup/ssl-weak-ciphers

Recommendation:

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. The best cipher will be the one which uses triple DES algorithm, with encryption key length of 128 bits and MD5 hash algorithm.

Proof of concept:

Field	Value
Signature hash algorithm	sha1
Issuer	07969287, Go Daddy Secure ...
Valid from	Thursday, February 20, 2014 ...
Valid to	Wednesday, March 4, 2015 1:...
Subject	*.mobile .com, Domain Contr...
Public key	RSA (2048 Bits)
Enhanced Key Usage	Server Authentication (1.3.6....

Field	Value
CRL Distribution Points	[1]CRL Distribution Point: Distr...
Certificate Policies	[1]Certificate Policy:Policy Ide...
Authority Information Access	[1]Authority Info Access: Acc...
Authority Key Identifier	KeyID=fd ac 61 32 93 6c 45 d...
Subject Alternative Name	DNS Name=*.mobile .com, D...
Subject Key Identifier	db c6 76 89 c2 07 f1 d9 00 33 ...
Basic Constraints	Subject Type=End Entity, Pat...
Key Usage	Digital Signature, Key Encipherment
Thumbprint algorithm	sha1
Thumbprint	44 cb 2b 91 bd e9 c5 e6 84 97 d3 43

Risk :	Medium	Status:	Pass	Reference ID:	14
Vulnerability Name:	Testing for Application Configuration Management weakness				

Description:

Improper configuration of an application created a major hole in the entire architecture.

Details:

Proper configuration of the single elements that make up application architecture is important in order to prevent mistakes that might compromise the security of the whole architecture. Many applications that come default in a web server have been later known to be vulnerable. This was the case, for example, for CVE-1999-0449 (Denial of Service in IIS when the Exair sample site had been installed), CAN-2002-1744 (Directory traversal vulnerability in CodeBrws.asp in Microsoft IIS 5.0), CAN-2002-1630 (Use of sendmail.jsp in Oracle 9iAS), or CAN-2003-1172 (Directory traversal in the view-source sample in Apache's Cocoon). CGI scanners include a detailed list of known files and directory samples that are provided by different web or application servers and might be a fast way to determine if these files are present. It is very common, and even recommended, for programmers to include detailed comments on their source code in order to allow for other programmers to better understand why a given decision was taken in coding a given function. Programmers usually do it too when developing large web-based applications. However, comments included inline in HTML code might reveal to potential attacker internal information that should not be available to them. Sometimes, even source code is commented out since functionality is no longer required, but this comment is leaked out to the HTML pages returned to the users unintentionally. The web server or application server configuration takes an important role in protecting the contents of the site and it must be carefully engineered.

Reference:

http://m.safaribooksonline.com/hd/public/content?portal=my&fpid=0735615608&s250=6275&s250w=800&s250h=572&s250uaw=800&s250uah=600#id=0735615608\firstchapter

Recommendation:

The recommended configuration varies depending on the site policy, and the functionality that should be provided by the server software. In most cases, however, configuration guidelines (either provided by the software vendor or external parties) should be followed in order to determine if the server has been properly secured. It is impossible to generically say how a server should be configured, however, some common guidelines should be taken into account:
Only enable server modules (ISAPI extensions in the IIS case) that are needed for the application.
Make sure that the server software runs with minimized privileges in the operating system.
Make sure the server software properly logs both legitimate access and errors.
Do not store sensitive information in these files if it should be for administrator eyes only.
Encrypt sensitive information that should be read by the IIS worker processes only and not by other users on the machine.

S.No	List of file with input
1	/epayment - 1 inputs
2	/epayment/admin/index.asp - 1 inputs
3	/epayment/testing/default.asp - 1 inputs
S.No	**List of external hosts**
1	mart.mobile.com.my
2	twitter.com
3	go.microsoft.com
S.No	**List of client side scripts**
1	/epayment/admin/dtree.js
2	/epayment/admin/admincountdowntimer.js
S.No	**List of file extensions**
1	asp - 8 files
2	css - 2 files
3	js - 2 files

Risk :	Medium	Status:	Pass	Reference ID:	15
Vulnerability Name:	Testing for File Extensions Handling				

Description:

File extension handling must be concentrated for better security of the application.

Details:

File extensions are commonly used in web servers to easily determine which technologies / languages / plugins must be used to fulfil the web request. Using standard file extensions provides the attacker useful information about the underlying technologies used in a web appliance and greatly simplifies the task of determining the attack scenario to be used on particular technologies. In addition, misconfiguration in web servers could easily reveal confidential information about access credentials. Extension checking is often used to validate files to be uploaded, which can lead to unexpected results because if the content is not what is expected, or because of unexpected OS filename handling. Determining how web servers handle requests corresponding to files having different extensions may help us to understand web server behaviour depending on the kind of files we try to access. For example, it can help us understand which file extensions are returned as text/plain versus those which cause execution on the server side. The latter are indicative of technologies / languages / plugins which are used by web servers or application servers, and may provide additional insight on how the web application is engineered. For example, a ".pl" extension is usually associated with server-side Perl support.

Reference:

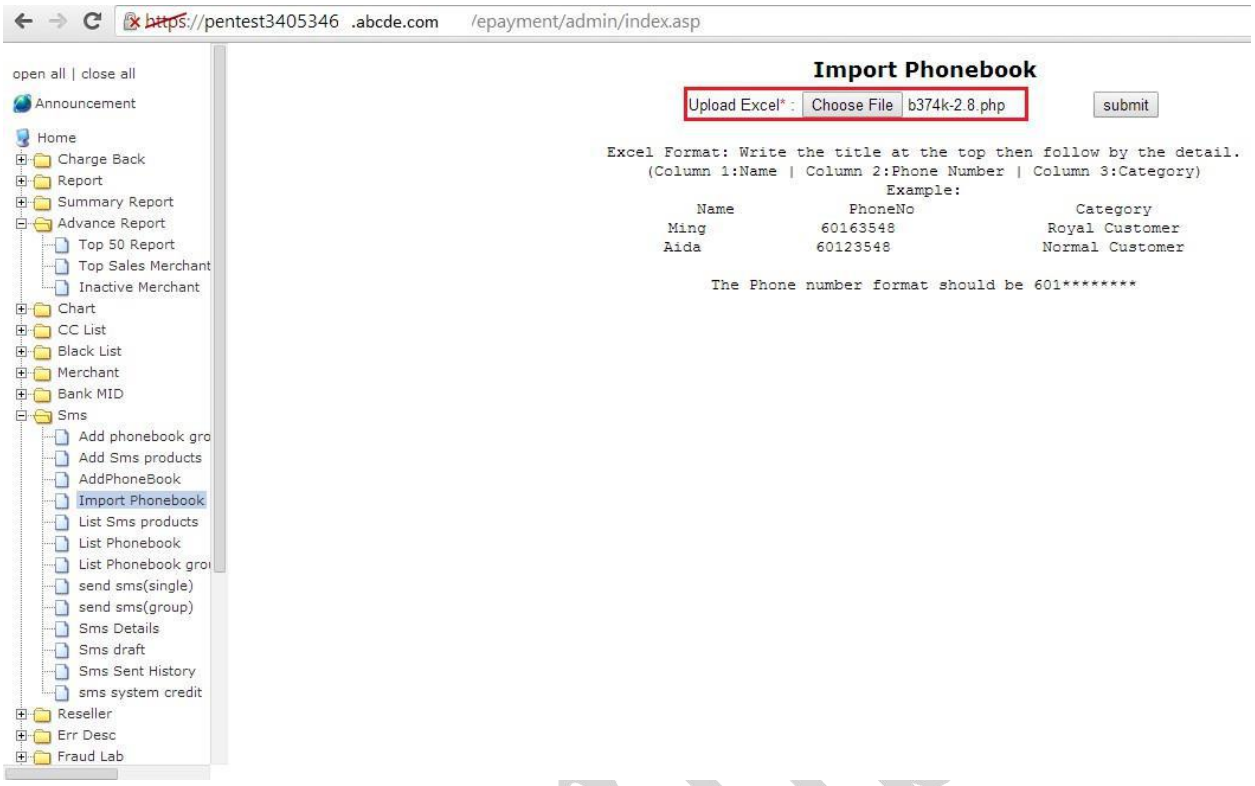
http://en.wikipedia.org/wiki/MIME http://www.ltsw.se/knbase/internet/mime.htp
http://www.iis.net/configreference/system.webserver/security/requestfiltering/fileextensions

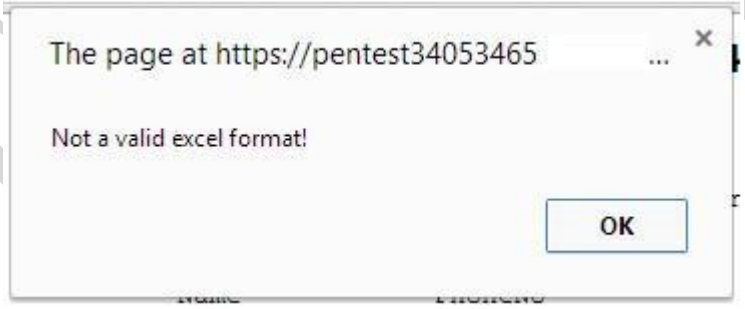
Recommendation:

The following example Web.config file will configure two options. It will configure request filtering to allow WebDAV access to all file name extensions, and it will configure IIS to deny access to files with a file name extension of .inc, which are sometimes used as include files for applications.

```
<requestFiltering>
        <fileExtensions applyToWebDAV="false">
           <add fileExtension=".inc" allowed="false" />
        </fileExtensions>
</requestFiltering>
```

Proof of concept:





Risk :	Medium	Status:	Pass	Reference ID:	16
Vulnerability Name:	Old, Backup and Unreferenced Files				

Description:

Old, backup and unreferenced files are very critical issue in the security and they can even disclose the source code of the application.

Details:

Most common scenario includes the presence of renamed old versions of modified files, inclusion files that are loaded into the language of choice and can be downloaded as source, or even automatic or manual backups in form of compressed archives. All these files may grant the attacker access to inner workings, backdoors, administrative interfaces, or even credentials to connect to the administrative interface or the database server. An important source of vulnerability lies in files which have nothing to do with the application, but are created as a consequence of editing application files, or after creating on-the-fly backup copies, or by leaving in the web tree old files or unreferenced files. That happens because backup copies may be generated with file extensions differing from those of the original files. A .tar, .zip or .gz archive that we generate (and forget) has obviously a different extension, and the same happens with automatic copies created by many editors. As a result, these activities generate files which are not needed by the application, may be handled differently than the original file by the web server.

Reference:

http://technet.microsoft.com/en-us/library/cc736787%28v=ws.10%29.aspx

Recommendation:

As a security best practice, log on to your computer using an account that is not in the Administrators group, and then use the **Run as** command to run IIS Manager as an administrator. At the command prompt, type **runas /user:***administrative_accountname* **mmc %systemroot%\system32\inetsrv\iis.msc.**

To create a portable backup (password required)

1. In IIS Manager, right-click the local computer, click **All Tasks**, and then click **Backup/Restore Configuration**.
2. Click **Create Backup**.
3. In the **Configuration backup name** box, type a name for the backup file.
4. Select the **Encrypt backup using password** check box, type a password into the **Password** box, and then type the same password in the **Confirm password** box.
5. Click **OK**, and then click **Close**.

   The IIS metabase is created in the *systemroot***system32\inetsrv\MetaBack** folder.

Proof of concept:

A	Dirs found with a 200 response:
1	/epayment/
B	**Dirs found with a 403 response:**
1	/epayment/images/
2	/epayment/image/
3	/epayment/security/
4	/epayment/Images/
5	/epayment/general/
6	/epayment/demo/
7	/epayment/registration/
8	/epayment/mobile/
9	/epayment/images/index/
C	**Dirs found with a 302 response:**
1	/epayment/admin/
2	/epayment/report/
D	**Files found with a 200 responce:**
1	/epayment/index.asp

Risk :	Medium	Status:	Pass	Reference ID:	17
Vulnerability Name:	Testing for Cookies attributes (Cookies are set not 'HTTP Only', 'Secure', and no time validity)				

Description:

Cookies are often a key attack vector for malicious users and, as such, the application should always take due diligence to protect cookies. The application has not taken the necessary precautions when assigning cookies and these attributes are not correctly configured.

Details:

If an attacker were by some means able to acquire a session token (for example, by exploiting a cross site scripting vulnerability or by sniffing an unencrypted session), then he/she could use this cookie to hijack a valid session.

The following is a list of the attributes that can be set for each cookie and what they mean.

• Secure - This attribute tells the browser to only send the cookie if the request is being sent over a secure channel such as HTTPS. This will help protect the cookie from being passed over unencrypted requests. If the application can be accessed over both HTTP and HTTPS, then there is the potential that the cookie can be sent in clear text.

• Http Only - This attribute is used to help prevent attacks such as cross-site scripting, since it does not allow the cookie to be accessed via a client side script such as JavaScript. Note that not all browsers support this functionality.

• Domain - This attribute is used to compare against the domain of the server in which the URL is being requested. If the domain matches or if it is a sub-domain, then the path attribute will be checked next.

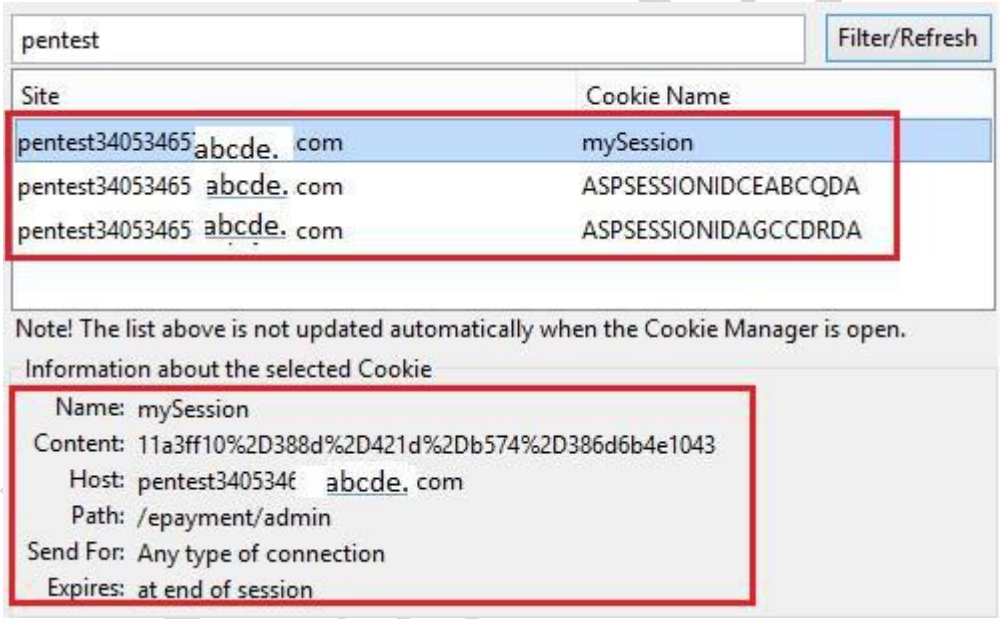
Reference:

https://www.owasp.org/index.php/SecureFlag
https://www.owasp.org/index.php/Httponly

Recommendation:

By the framework cookies marked as httpOnly cannot be accessed from JavaScript and a Major benefit of using these flags are that they stop stealing through XSS vulnerabilities. The cookie cannot be accessed through client side script if the httponly flag is set. The purpose of the secure flag is to prevent cookies from being observed by unauthorized parties due to the transmission of a the cookie in clear text. A Secure cookie is a file that is stored on a user's hard drive. It is used for transmitting http or https over the internet where https is a secure protocol and provides a secure transmission of data over your internet connection.

Proof of concept:



Risk :	Medium	Status:	Pass	Reference ID:	18
Vulnerability Name:	Testing for Exposed Session Variables				

Description:

The Session Tokens (Cookie, SessionID, Hidden Field), if exposed, will usually enable an attacker to impersonate a victim and access the application illegitimately. As such, it is important that they are protected from eavesdropping at all times – particularly whilst in transit between the Client browser and the application servers.

Details:

The information here relates to how transport security applies to the transfer of sensitive Session ID data rather than data in general, and may be stricter than the caching and transport policies applied to the data served by the site. Using a personal proxy, it is possible to ascertain the following about each request and response:
Protocol used (e.g., HTTP vs. HTTPS)
HTTP Headers
Message Body (e.g., POST or page content)
Each time Session ID data is passed between the client and the server, the protocol, cache, and privacy directives and body should be examined. Transport security here refers to Session IDs passed in GET or POST requests, message bodies, or other means over valid HTTP requests.

Reference:

http://www.ietf.org/rfc/rfc2965.txt
http://www.ietf.org/rfc/rfc2616.txt

Recommendation:

The interaction between the Client and Application should be tested at least against the following criteria.
- How are Session IDs transferred? e.g., GET, POST, Form Field (including hidden fields)
- Are Session IDs always sent over encrypted transport by default?
- Is it possible to manipulate the application to send Session IDs unencrypted? e.g., by changing HTTP to HTTPS?
- What cache-control directives are applied to requests/responses passing Session IDs?
- Are these directives always present? If not, where are the exceptions?
- Are GET requests incorporating the Session ID used?
- If POST is used, can it be interchanged with GET?

Proof of concept:

Risk :	Medium	Status:	Pass	Reference ID:	19
Vulnerability Name:	Testing for incubated vulnerabilities				

Description:

It is possible for an attacker to plant a piece of data that will later be retrieved by an unsuspecting user or other component of the system, exploiting some vulnerability.

Details:

Incubated vulnerability is also often refered to as persistent attacks, incubated testing is a complex testing method that needs more than one data validation vulnerability to work. This section describes a set of examples to test an Incubated Vulnerability. This type of asynchronous attack covers a great spectrum of attack vectors, among them the following:

- File upload components in a web application
- Cross-site scripting issues in public forums post
- SQL/XPATH Injection allowing the attacker to upload content to a database
- Misconfigured servers allowing installation of Java packages or similar web site components

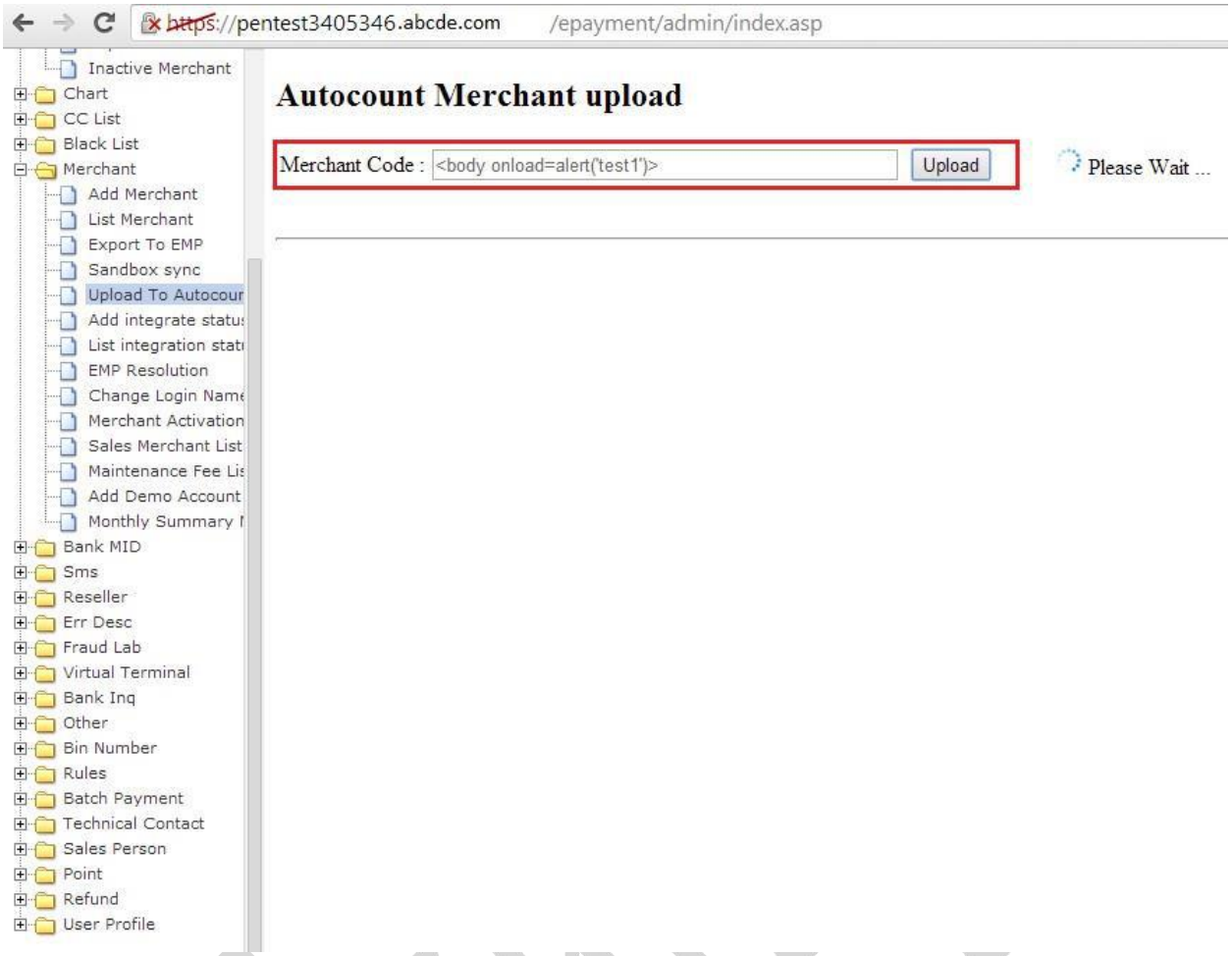
Reference:

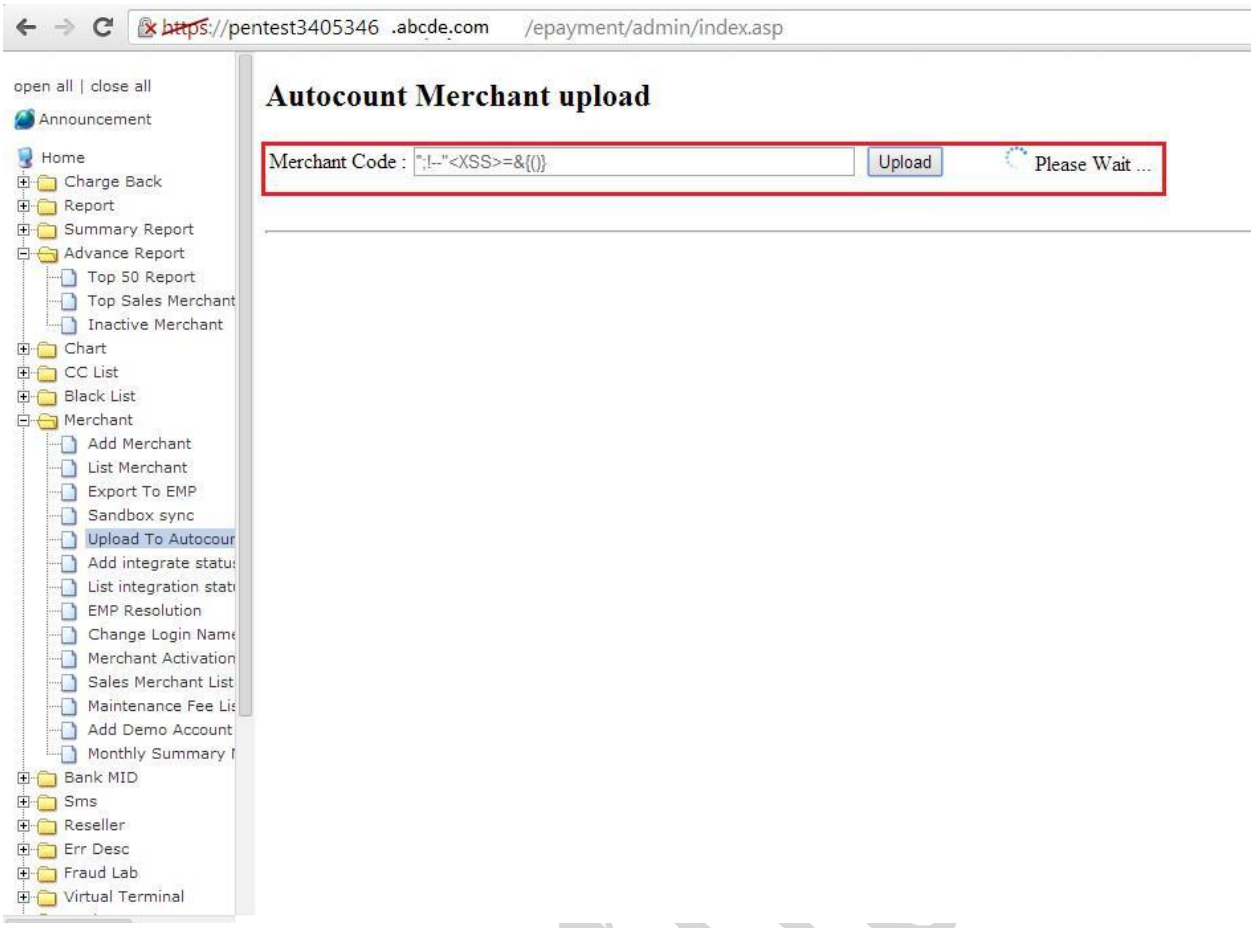
http://www.cert.org/advisories/CA-2000-02.html
http://lists.grok.org.uk/pipermail/full-disclosure/2006-July/048059.html
http://projects.webappsec.org/w/page/13246920/Cross%20Site%20Scripting

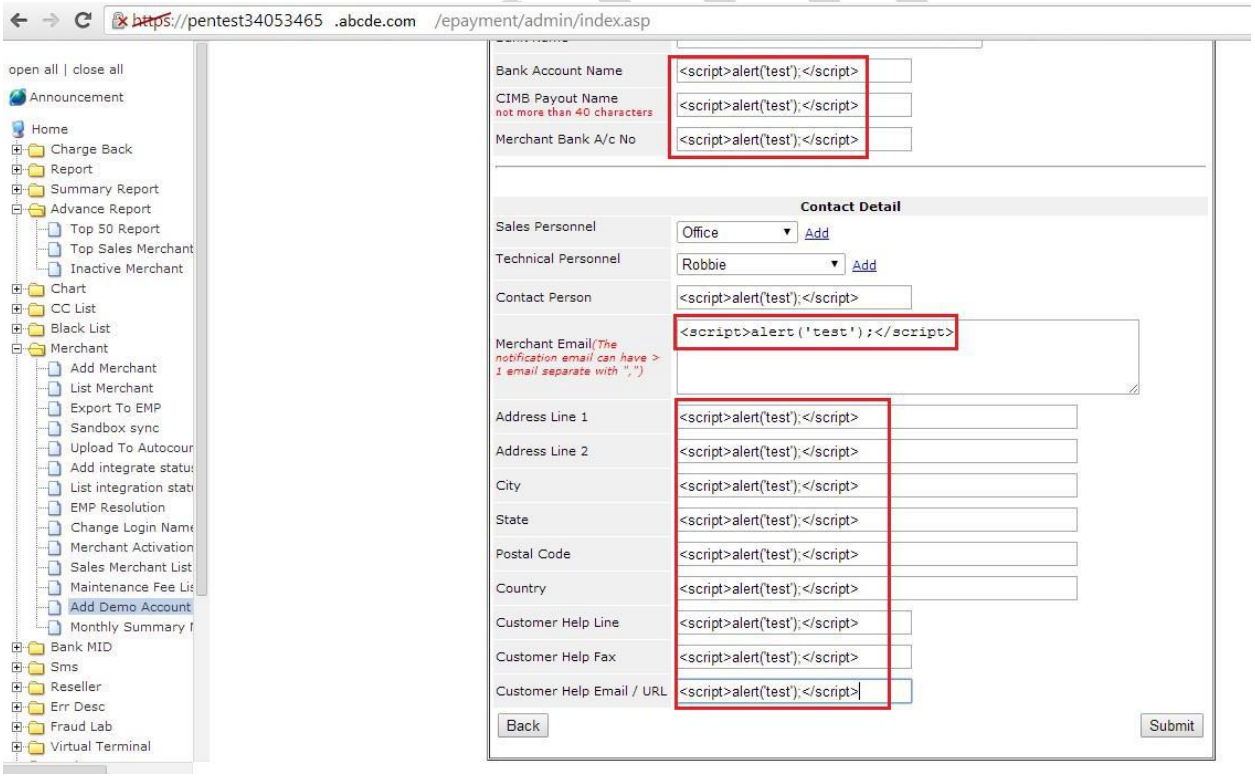
Recommendation:

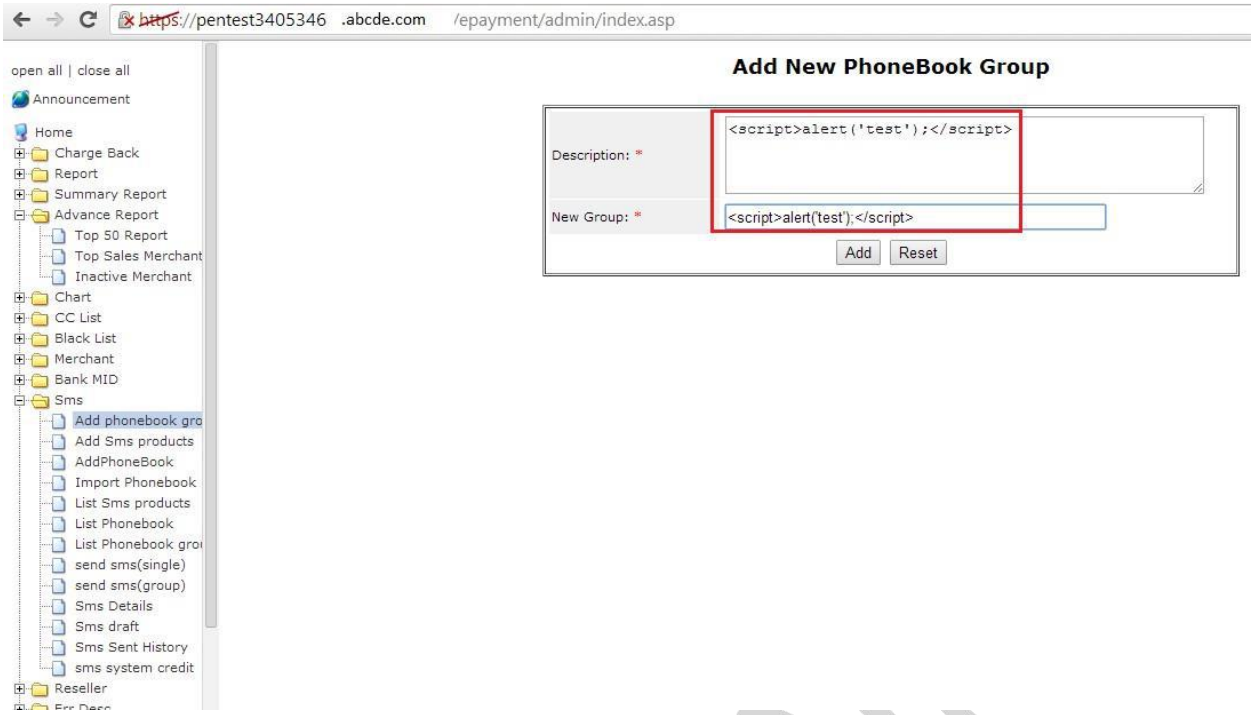
Incubated vulnerabilities must be prevented by validating all the input fields for all the vulnerabilities. This is always exploited due to the coding phase. There will be measures taken for all the attacks. But the last preventive measure could make the first prevention invalid. This must be taken care for better security.

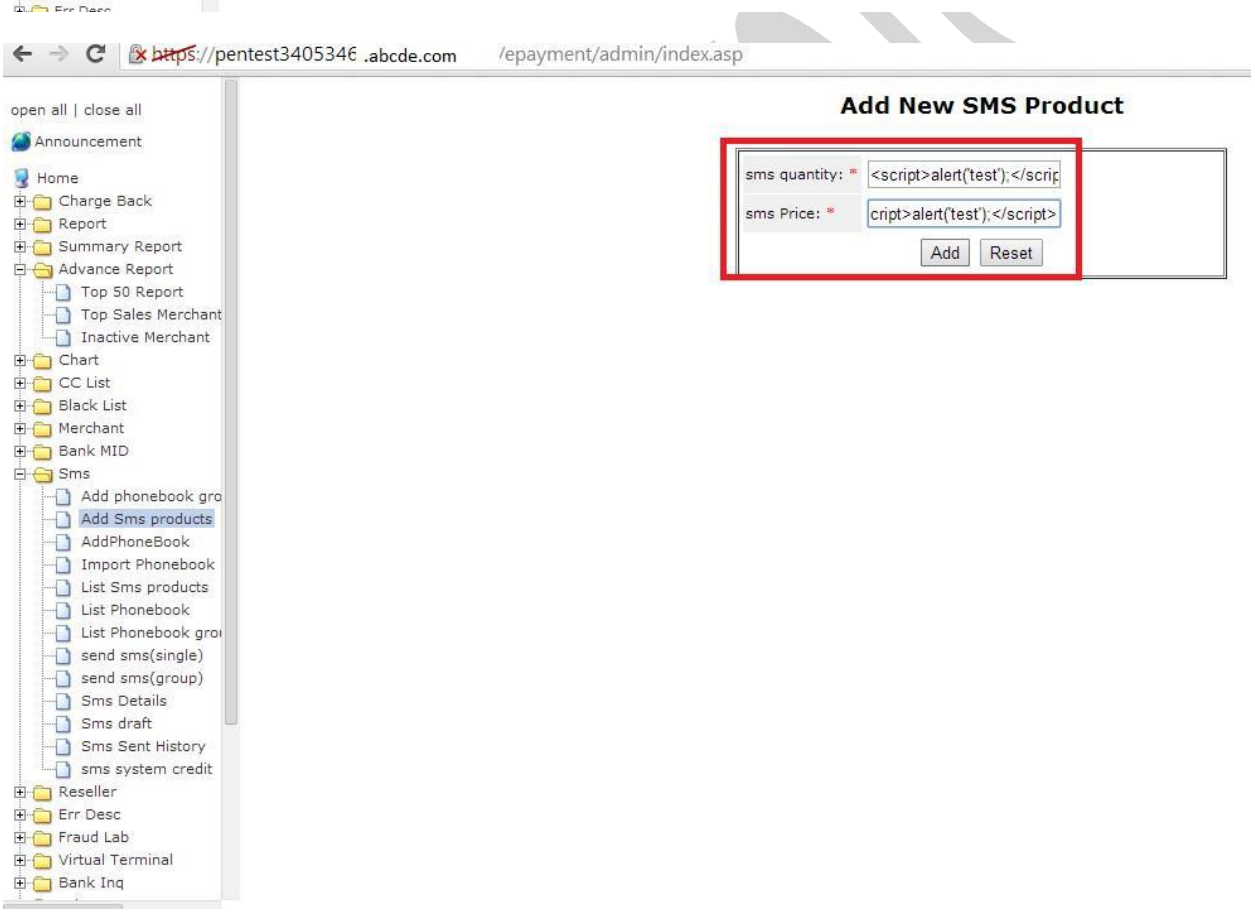
Proof of concept:

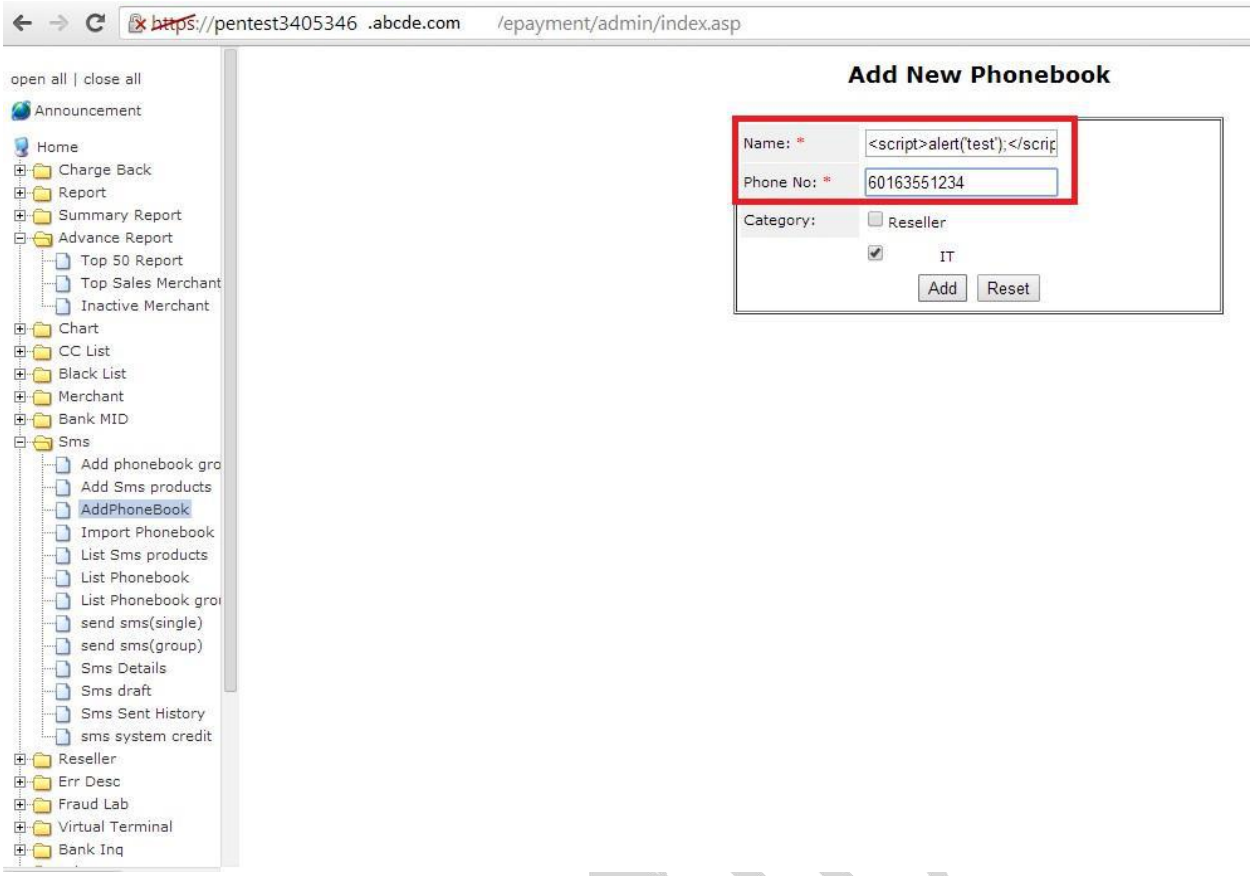


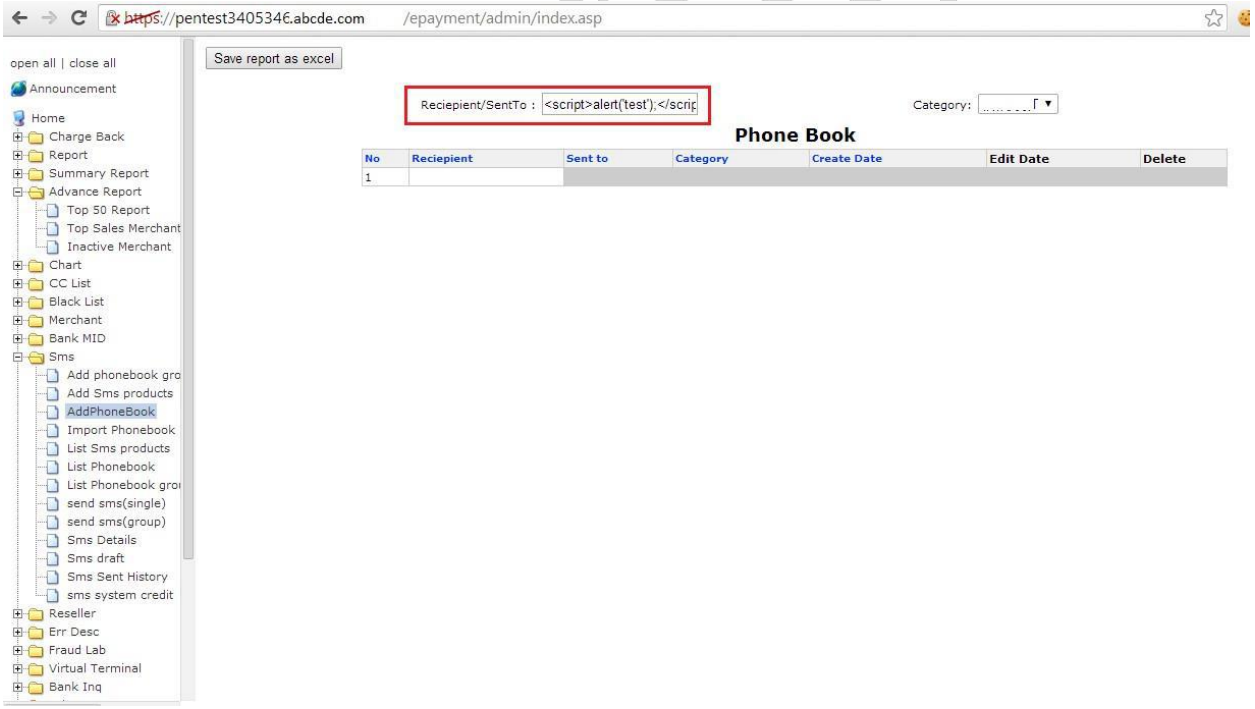


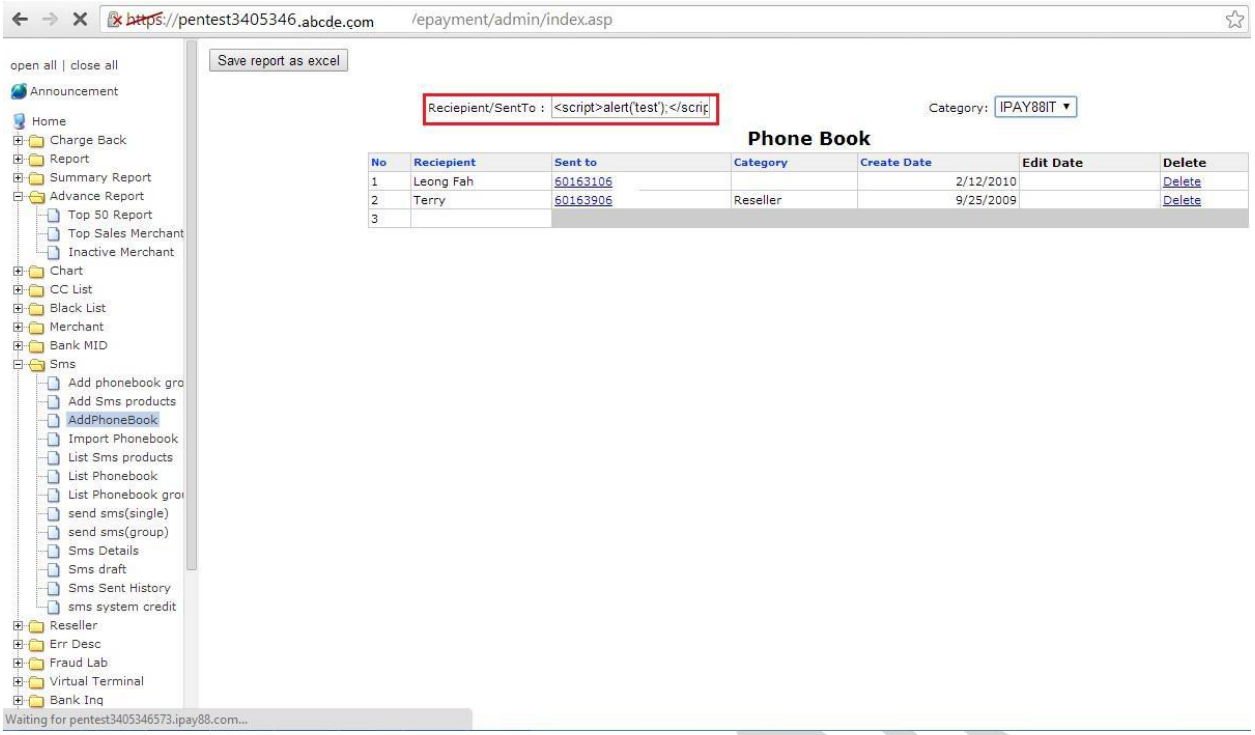


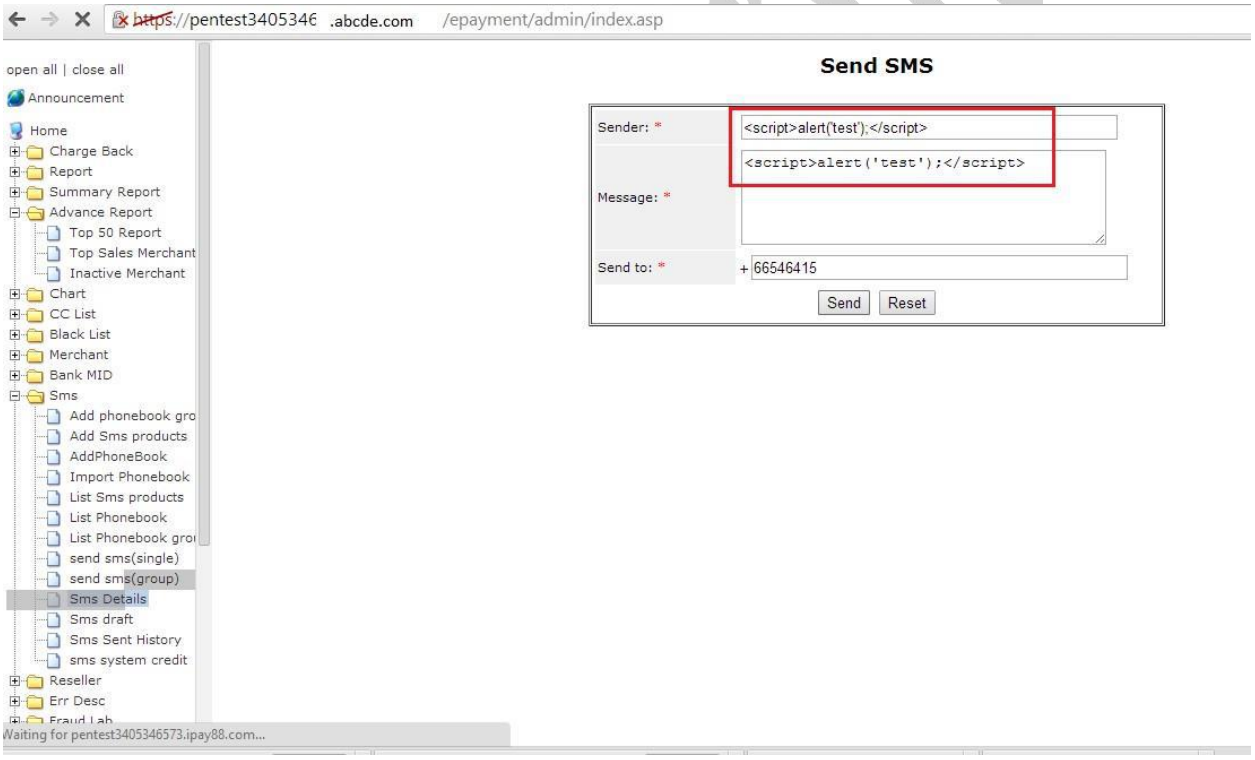


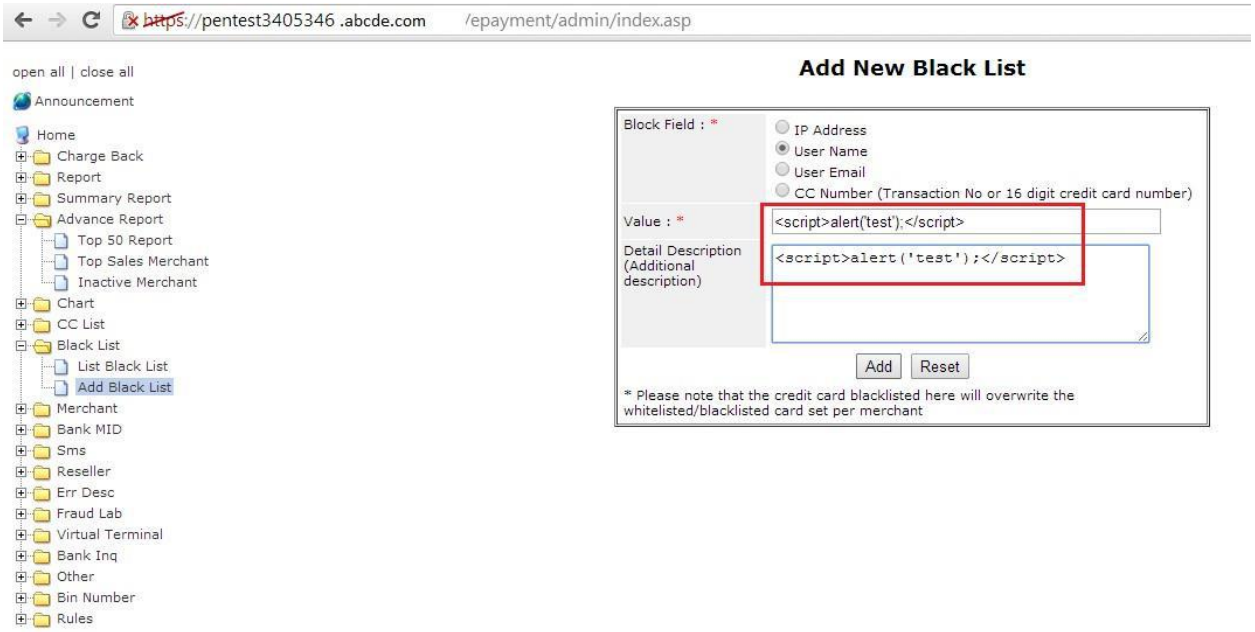


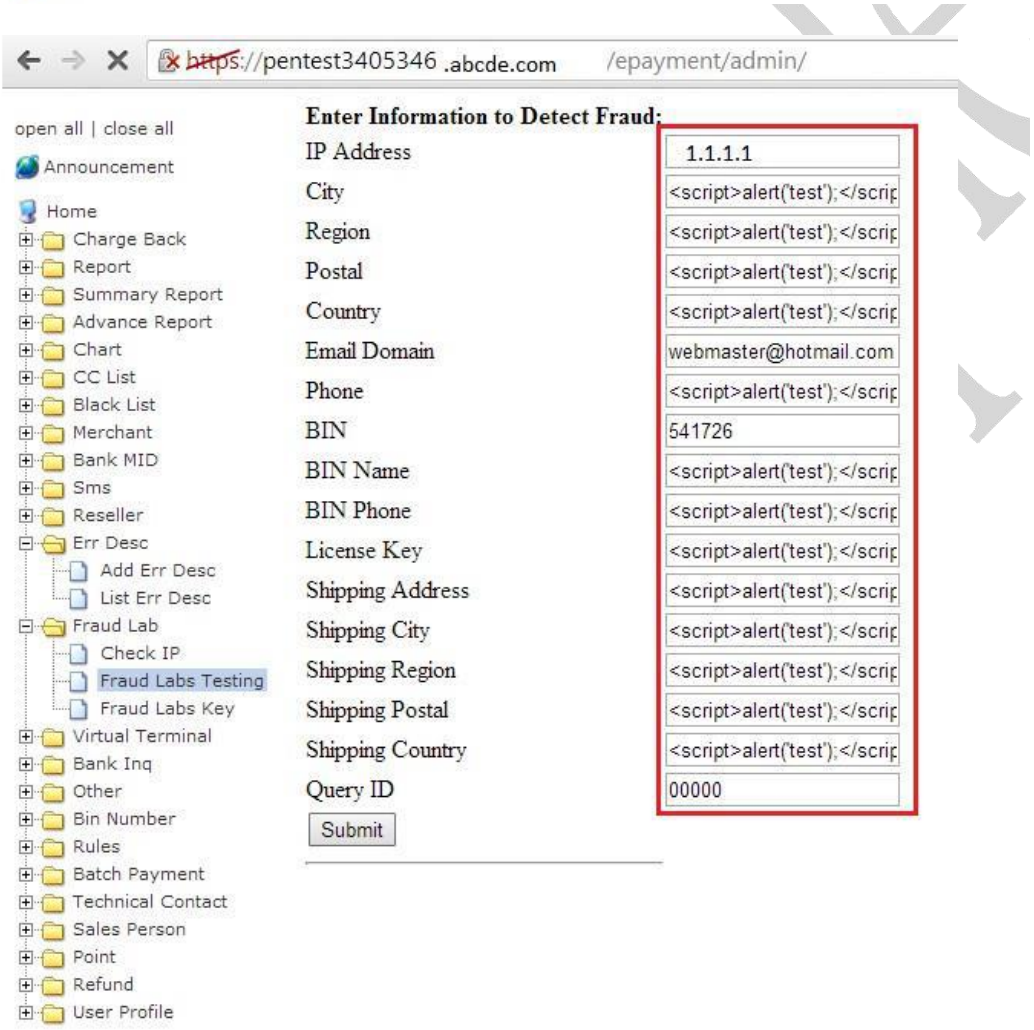


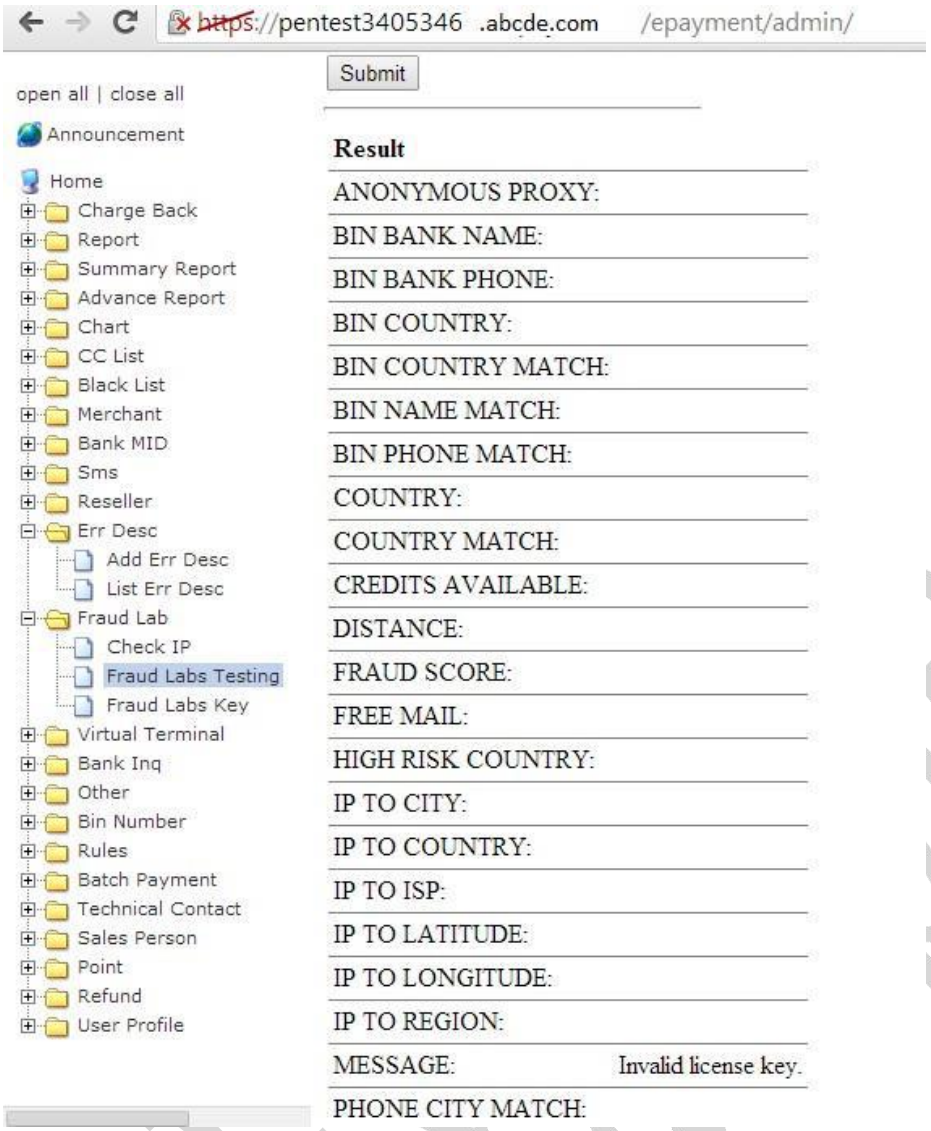












Risk :	Low	Status:	Pass	Reference ID:	20
Vulnerability Name:	Spiders, Robots and Crawlers				

Description:

It is possible to get information about sensitive web pages; the developers don't want the spiders to crawl, through Web crawlers, spiders and robots analysis.

Details:

Web spiders/robots/crawlers retrieve a web page and then recursively traverse hyperlinks to retrieve further web content. Their accepted behavior is specified by the 'Robots Exclusion Protocol' of the 'robots.txt' file in the web root directory.

As an example, the robots.txt file will be like,
User-agent: *
Allow: /allowed_folder/
Disallow: /restricted_folder/

The User-Agent directive refers to the specific web spider/robot/crawler. The Disallow directive specifies which resources are prohibited by spiders/robots/crawlers. Web spiders/robots/crawlers can intentionally ignore the Disallow directives specified in a robots.txt file. But the attackers can intentionally view the sensitive folders and get information about the application.

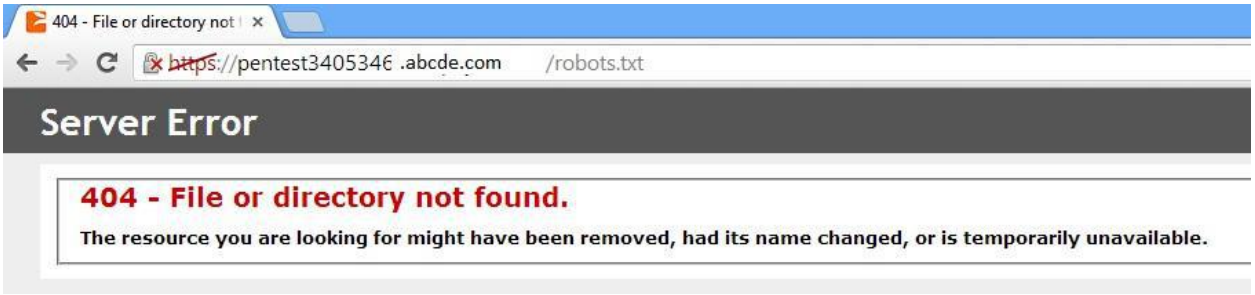
Reference:

http://www.motive.co.nz/glossary/spider.php
http://en.wikipedia.org/wiki/Web_crawler
http://en.wikipedia.org/wiki/Robots_exclusion_standard
http://www.robotstxt.org/
http://tools.seobook.com/robots-txt/

Recommendation:

The sensitive folders which are listed in the robots.txt should not be accessed in public (i.e. available to anyone in the world with internet access). This can be done using .htaccess file. These files provide a way to make configuration changes on a per-directory basis. htaccess file can be configured such that, no public user is allowed to view the content. To be more convenient, it can be configured that a certain users or IPs alone can access those files. Now a days most of the servers are engineered to protect access for htaccess and htpasswd files from public.

Proof of Concept:



Risk :	Low	Status:	Failed	Reference ID:	21
Vulnerability Name:	Testing for default credentials				

Description:

Use of default username and password or forgetting to remove the default credentials could make compromise of the entire system.

Details:

Nowadays web applications often make use of popular open source or commercial software that can be installed on servers with minimal configuration or customization by the server administrator. Moreover, a lot of hardware appliances (i.e. network routers and database servers), offer web-based configuration or administrative interfaces. Often these applications, once installed, are not properly configured and the default credentials provided for initial authentication and configuration are never changed. These default credentials are well known by penetration testers and, unfortunately, also by malicious attackers, who can use them to gain access to various types of applications. Furthermore, in many situations, when a new account is created on an application, a default password (with some standard characteristics) is generated. If this password is predictable and the user does not change it on the first access, this can lead an attacker to gain unauthorized access to the application. The following usernames - "admin", "administrator", "root", "system", "guest", "operator", "super" or "superuser" are popular among system administrators and are often used. Additionally the other usernames frequently used are "test", "test1", "test123", "testing123", "testing". The vulnerable passwords are "password", "pass123", "password123", "admin", or "guest" with the above accounts or any other enumerated accounts.

Reference:

http://www.totaldefense.com/blogs/security-advisor/2012/01/24/password-best-practices.aspx
http://security.stackexchange.com/questions/7982/creating-username-policies-and-best-practices
http://serverfault.com/questions/348912/best-practices-in-username-standards-avoiding-problems
http://community.spiceworks.com/topic/90229-username-best-practices

Recommendation:

Use of the above specified username and password should not be practiced. Especially the above username and password combination should not be done. Use of names like the Personal Identifiable Information like name, company name, firend's name, birthday, age, pet's name. Password must not be used as a full dictionary word. Password must be a combination of alpha numerical, at least.

Proof of concept:

Risk :	Low	Status:	Pass	Reference ID:	22
Vulnerability Name:	Testing for bypassing authentication schema				

Description:

It is possible to bypass authentication using some techniques which should not be done for secured login.

Details:

While most applications require authentication for gaining access to private information or to execute tasks, not every authentication method is able to provide adequate security. Negligence, ignorance, or simple understatement of security threats often result in authentication schemes that can be bypassed by simply skipping the login page and directly calling an internal page that is supposed to be accessed only after authentication has been performed. In addition to this, it is often possible to bypass authentication measures by tampering with requests and tricking the application into thinking that we're already authenticated. This can be accomplished either by modifying the given URL parameter or by manipulating the form or by counterfeiting sessions.

There are several methods to bypass the authentication schema in use by a web application:

- Direct page request (forced browsing)
- Parameter Modification
- Session ID Prediction
- SQL Injection

Reference:

http://googlecode.blogspot.in/2011/03/best-practices-for-user-authentication.html
http://stackoverflow.com/questions/1624846/php-best-practices-for-user-authentication-and-password-security
http://stackoverflow.com/questions/5876859/php-best-practice-on-user-authentication-for-a-website

Recommendation:

It is always recommended to use valid session for authentication. Also it is very important than anything to use a session generation which is very hard to predict. Don't forget to destroy the user's inpersistent session if there is an inactivity/logout/close activity detected. Don't disclose the token which is used to activate session like 'login=failure'. Then it is obvious for the attacker to manipulate the token to 'login=success' to validate the login attempt.

Proof of concept:

S.No	Not Authorised pages
1	Bank - Alliance Bank
2	Point - View Point Maintenance
3	Refund - Search transaction, view transaction, refund transaction report

Risk :	Low	Status:	Pass	Reference ID:	23
Vulnerability Name:	Testing Directory traversal/file include				

Description:

It is possible to traverse directory and files without hyperlinks.

Details:

Many web applications use and manage files as part of their daily operation. Using input validation methods that have not been well designed or deployed, an aggressor could exploit the system in order to read/write files that are not intended to be accessible. In particular situations, it could be possible to execute arbitrary code or system commands. A Path Traversal attack aims to access files and directories that are stored outside the web root folder. By browsing the application, the attacker looks for absolute links to files stored on the web server. By manipulating variables that reference files with "dot-dot-slash (../)" sequences and its variations, it may be possible to access arbitrary files and directories stored on file system, including application source code, configuration and critical system files, limited by system operational access control. The attacker uses "../" sequences to move up to root directory, thus permitting navigation through the file system. This attack is also known as "dot-dot-slash", "directory traversal", "directory climbing" and "backtracking".

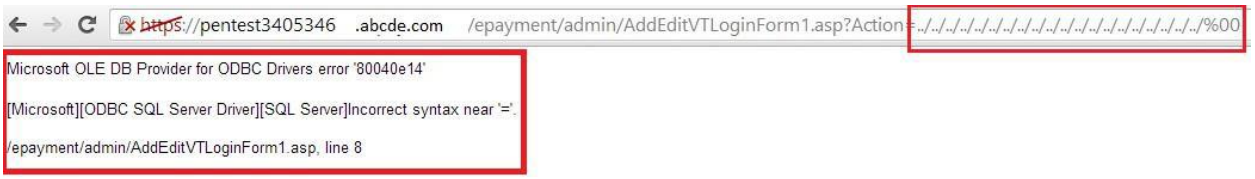
Reference:

http://en.wikipedia.org/wiki/Directory_traversal_attack
https://www.owasp.org/index.php/Path_Traversal
http://www.acunetix.com/websitesecurity/directory-traversal/

Recommendation:

• Use the tighest possible permissions when developing and deploying web applications
• Remove all "Everyone:Full Control" ACLs on Windows, and all mode 777 (world writeable directories) or mode 666 files (world writeable files) on Unix systems
• Strongly consider removing "Guest", "everyone," and world readable permissions wherever possible
• Use robots.txt – this will prevent most search engines looking any further than what you have in mind, but be aware that attackers can view the contents of this directory and fuzz it for content, as well.
• Use a "garbage collector" to delete old temporary files, either at the end of a session or within a timeout period, such as 20 minutes.
• If deployed under Unix-like operating systems, use chroot jails to isolate the application from the primary operating system. On Windows, use the inbuilt ACL support to prevent the IIS users from retrieving or overwriting the files directly.
• Rename include files to be normal extension (such as foo.inc ?foo.jsp or foo.aspx).
• Map all files that need to remain, such as .xml or .cfg to an error handler or a renderer that will not disclose the file contents. This may need to be done in both the web application framework's configuration area or the web server's configuration.

Proof of concept:



Risk :	Informational	Status:	Pass	Reference ID:	24
Vulnerability Name:	Analysis of Error Codes				

Description:

Error codes can disclose information about the application and its version which may be vulnerable or lead to future vulnerabilities.

Details:

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 attackers during their activities in attack 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.

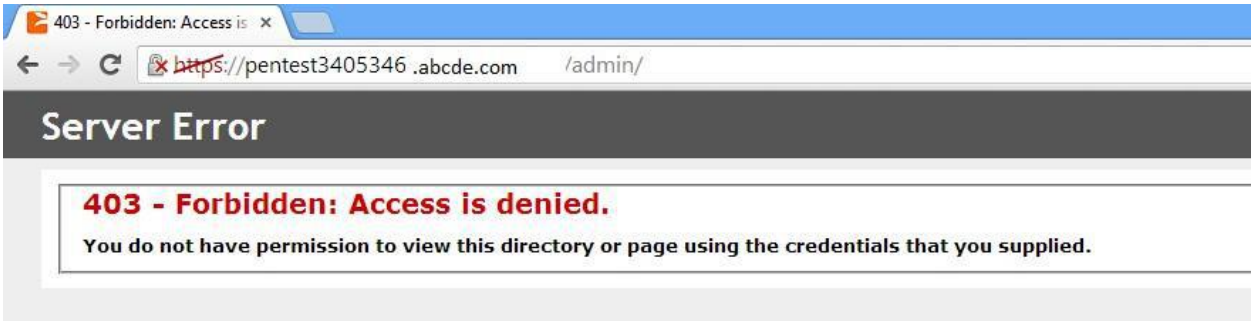
Reference:

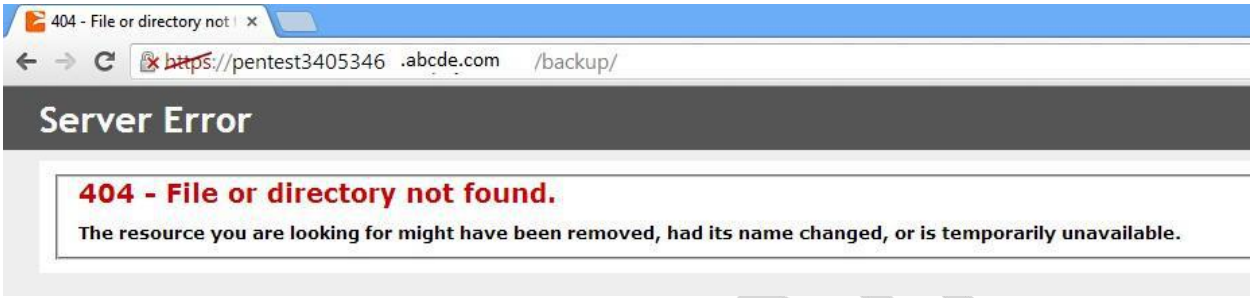
https://www.owasp.org/index.php/Information_Leakage
http://projects.webappsec.org/w/page/13246936/Information%20Leakage
http://www.thesitewizard.com/archive/custom404.shtml
http://wiki.dreamhost.com/Creating_custom_error_pages
http://kb.mediatemple.net/questions/8/Creating+custom+error+pages#gs
http://techtalk.virendrachandak.com/404-error-page-best-practices/
http://www.flintstudio.com/blog/6-best-practices-when-designing-developing-404-error-pages/
http://support.microsoft.com/kb/834452

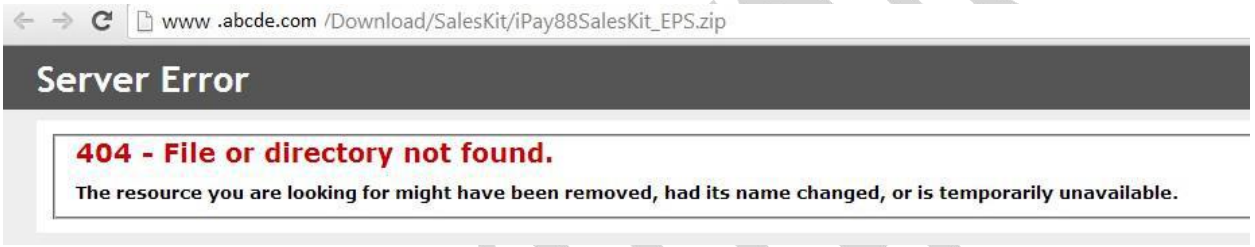
Recommendation:

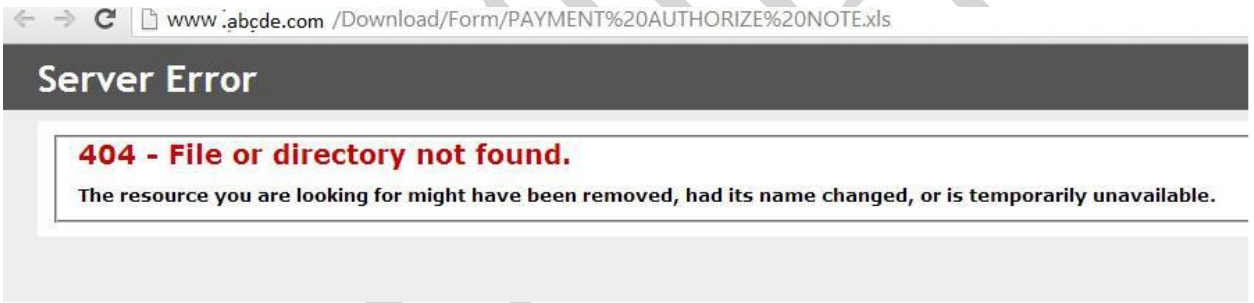
To create a custom error page for your account, please login to cPanel and click Error Pages, under Advanced. Select the domain or subdomain you want and click the page you want to edit. Insert your own custom page code (in HTML or SHTML). The changes will be applied after you click Save. Add error code to your .htaccess file in the root directory.
For Example:
ErrorDocument 403 /403.shtml
ErrorDocument 404 /404.shtml
ErrorDocument 500 /500.shtml

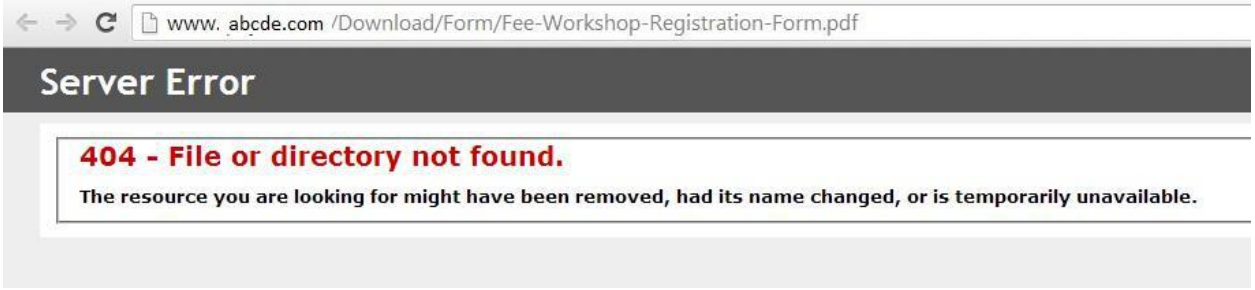
Proof of concept:

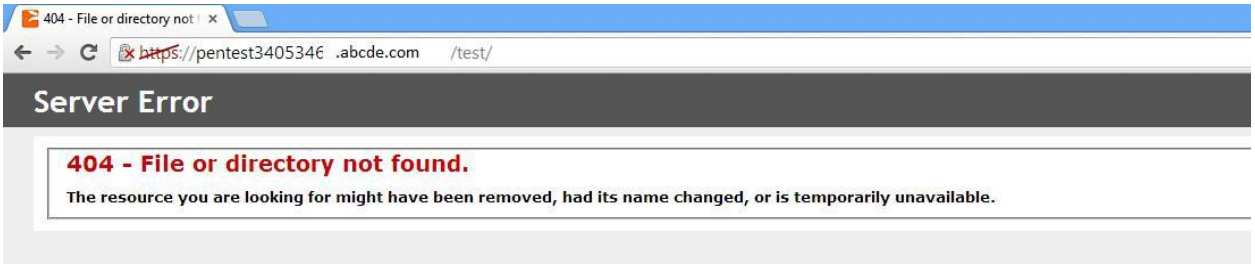


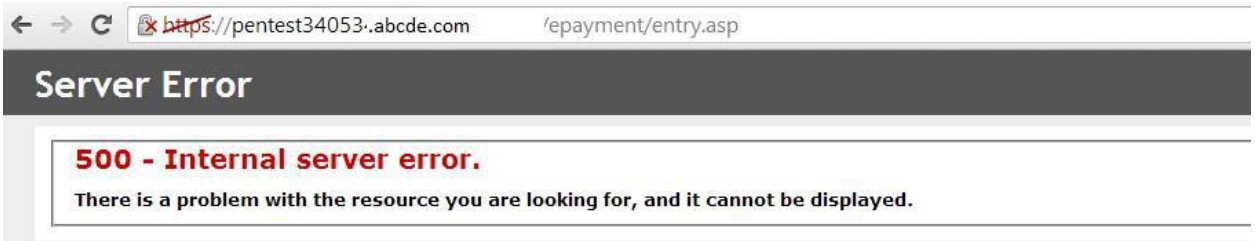












Risk :	Informational	Status:	Pass	Reference ID:	25
Vulnerability Name:	Testing for Infrastructure Configuration Management Testing weakness				

Description:

It is found that the infrastructure configuration is not managed properly and is exposed to various types of exploit.

Details:

Proper configuration management of the web server infrastructure is very important in order to preserve the security of the application itself. If elements such as the web server software, the back-end database servers, or the authentication servers are not properly reviewed and secured, they might introduce undesired risks or introduce new vulnerabilities that might compromise the application itself.

The different elements that make up the infrastructure need are analyzed in order to understand how they interact with a web application and how they affect its security. All the elements of the infrastructure are reviewed in order to make sure that they don't hold any known vulnerabilities. A review is made of the administrative tools used to maintain all the different elements.
The authentication systems, if any, are reviewed in order to assure that they serve the needs of the application and that they cannot be manipulated by external users to leverage access. Lists of ports which are used by the server are analyzed.

In small setups, such as a simple CGI-based application, a single server might be used that runs the web server which executes the C, Perl, or Shell CGIs application, and perhaps also the authentication mechanism. On more complex setups, such as an online bank system, multiple servers might be involved including: a reverse proxy, a front-end web server, an application server and a database server or LDAP server

Reference:

http://arstechnica.com/gadgets/2012/11/how-to-set-up-a-safe-and-secure-web-server/
http://community.spiceworks.com/topic/154956-what-is-the-best-way-to-setup-a-redundant-web-server-and-database
http://security.stackexchange.com/questions/10004/is-it-worth-to-implement-a-firewall-on-a-web-server-you-control
http://support.microsoft.com/kb/309814

Recommendation:

Each of these servers will be used for different purposes and might be even be divided in different networks with firewalling devices between them, creating different DMZs so that access to the web server will not grant a remote user access to the authentication mechanism itself, and so that compromises of the different elements of the architecture can be isolated in a way such that they will not compromise the whole architecture. A list of defined ports which are required for the application should be maintained and kept under change control.

Risk :	Informational	Status:	Pass	Reference ID:	26
Vulnerability Name:	Infrastructure and Application Admin Interfaces				

Description:

Disclosure of admin interface allows the attacker to try brute force which results in gaining access of the entire application.

Details:

Administrator interfaces may be present in the application or on the application server to allow certain users to undertake privileged activities on the site. An application may require an administrator interface to enable a privileged user to access functionality that may make changes to how the site functions. Such changes may include:
- user account provisioning
- site design and layout
- data manipulation
- configuration changes

In many instances, such interfaces are usually implemented with little thought of how to separate them from the normal users of the site. Attackers aim at discovering these administrator interfaces and accessing functionality intended for the privileged users.
Once an administrative interface has been discovered, a combination of the some techniques may be used to attempt in bypassing authentication. If this fails, the tester may wish to attempt a brute force attack.

Reference:

http://docs.geoserver.org/latest/en/user/gettingstarted/web-admin-quickstart/index.html
http://getsymphony.com/learn/concepts/view/admin-interface/
http://forum.joomla.org/viewtopic.php?p=919504
http://drupal.org/node/105260

Recommendation:

Renaming the application admin interface for a different name rather than usual names like admin, owner, user, author etc., In content management systems like Joomla, Drupal, Wordpress, the admin previlage can be easily identified. It is better to rename those interfaces manually. Even though it can be found by some advanced methods and brute force is possible. In such an instance the developers should be aware of the potential for administrative account lockout. Emailing after a particular account with reset password is considered as industry's best practice.

Risk :	Informational	Status:	Pass	Reference ID:	27
Vulnerability Name:	Testing for Bad HTTP Methods				

Description:

HTTP methods can be used for gathering information about the web server due to misconfiguration in the server.

Details:

HTTP offers a number of methods that can be used to perform actions on the web server. Many of these methods are designed to aid developers in deploying and testing HTTP applications. These HTTP methods can be used for nefarious purposes if the web server is misconfigured. Additionally, Cross Site Tracing (XST), a form of cross site scripting using the server's HTTP TRACE method is possible.

GET and POST are by far the most common methods that are used to access information provided by a web server, the Hypertext Transfer Protocol (HTTP) allows several other (and somewhat less known) methods. HEAD, GET, POST, PUT, DELETE, TRACE, OPTIONS, CONNECT.

The OPTIONS HTTP method provides some way to figure out which HTTP methods are supported by the web server. TRACE method can be used for performing XST attack. HTTP offers a number of methods that can be used to perform actions on the web server. Many of these methods are designed to aid developers in deploying and testing HTTP applications. These HTTP methods can be used for nefarious purposes if the web server is misconfigured. Additionally, Cross Site Tracing (XST), a form of cross site scripting using the server's HTTP TRACE method is possible.

GET and POST are by far the most common methods that are used to access information provided by a web server, the Hypertext Transfer Protocol (HTTP) allows several other (and somewhat less known) methods. HEAD, GET, POST, PUT, DELETE, TRACE, OPTIONS, CONNECT

The OPTIONS HTTP method provides some way to figure out which HTTP methods are supported by the web server. TRACE method can be used for performing XST attack. All the HTTP methods can be used as per their function.

Reference:

http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html
http://www.httpwatch.com/httpgallery/methods/
http://annevankesteren.nl/2007/10/http-methods
http://www-01.ibm.com/support/docview.wss?uid=swg21201202

Recommendation:

Disable the HTTP methods which are not used

Proof of concept:

The following HTTP methods are allowed in this website

S.No	Allowed Methods
1	OPTIONS
2	TRACE
3	GET
4	HEAD
5	POST

Risk :	Informational	Status:	Pass	Reference ID:	28
Vulnerability Name:	Testing for Browser cache weakness				

Description:

Browser cache weakness can cause the disclosure of browser saved files which may contain the user credentials

Details:

The application must automatically logs out a user when that user has been idle for a certain amount of time, and that no sensitive data remains stored in the browser cache. If actions like logout, page redirect, idle session are not properly carried out, an attacker could replay these session tokens in order to “resurrect” the session of a legitimate user and impersonate him/her (this attack is usually known as 'cookie replay'). Of course, a mitigating factor is that the attacker needs to be able to access those tokens (which are stored on the victim's PC), but, in a variety of cases, this may not be impossible or particularly difficult. The most common scenario for this kind of attack is a public computer that is used to access some private information (e.g., webmail, online bank account): when the user has finished using the application and logs out, if the logout process is not properly enforced, the following user could access the same account, for instance, by simply pressing the “back” button of the browser. Another scenario can result from Cross Site Scripting vulnerability (XSS) or a connection that is not 100% protected by SSL: a flawed logout function would make stolen cookies useful for a much longer time, making life for the attacker much easier. The third test of this chapter is aimed to check that the application prevents the browser to cache sensitive data, which again would pose a danger to a user accessing the application from a public computer.

Recommendation:

Logging out from an application obviously does not clear the browser cache of any sensitive information that might have been stored. Therefore, another test that is to be performed is to check that our application does not leak any critical data into the browser cache. The logout function must effectively destroy all session token, or at least renders them unusable. The server must perform proper checks on the session state, disallowing an attacker to replay some previous token. A timeout must enforce and properly checked by the server. If the server uses an expiration time that is read from a session token that is sent by the client, the token must be cryptographically protected

Risk :	Informational	Status:	Pass	Reference ID:	29
Vulnerability Name:	Testing for CAPTCHA				

Description:

CAPTCHA ("Completely Automated Public Turing test to tell Computers and Humans Apart") is a type of challenge-response test used by many web applications to ensure that the response is not generated by a computer. CAPTCHA implementations are often vulnerable to various kinds of attacks even if the generated CAPTCHA is unbreakable.

Details:

Although CAPTCHA is not an authentication control, its use can be very efficient against:
• enumeration attacks (login, registration or password reset forms are often vulnerable to enumeration attacks - without CAPTCHA the attacker can gain valid usernames, phone numbers or any other sensitive information in a short time)
• automated sending of many GET/POST requests in a short time where it is undesirable (e.g., SMS/MMS/email flooding), CAPTCHA provides a rate limiting function
• automated creation/using of the account that should be used only by humans (e.g., creating webmail accounts, stop spamming)
• automated posting to blogs, forums and wikis, whether as a result of commercial promotion, or harassment and vandalism
• any automated attacks that massively gain or misuse sensitive information from the application

These vulnerabilities are quite common in many CAPTCHA implementations:
generated image CAPTCHA is weak, this can be identified (without any complex computer recognition systems) only by a simple comparison with already broken CAPTCHAs
• generated CAPTCHA questions have a very limited set of possible answers
• the value of decoded CAPTCHA is sent by the client (as a GET parameter or as a hidden field of POST form). This value is often:
• encrypted by simple algorithm and can be easily decrypted by observing of multiple decoded CAPTCHA values
• hashed by a weak hash function (e.g., MD5) that can be broken using a rainbow table
• possibility of replay attacks

Reference:

http://www.captcha.net/
http://securesoftware.blogspot.in/2007/11/captcha-placebo-security-control-for.html
http://www.cs.sfu.ca/~mori/research/gimpy/
http://www.puremango.co.uk/2005/11/breaking_captcha_115/

Recommendation:

Secured CAPTCHAs like google'sreCAPTCHA API can be used which are trustworthy. It contains a large number of combinations. However CAPTCHAs are used OCR(Optical Character Recognition) is used to break CAPTCHAs by reading the characters in the screen. This can be avoided using 3D CAPTCHA and Intelligent CAPTCHA. There are advanced CAPTCHAs which allows you to draw an image using mouse to authenticate.

Risk :	Informational	Status:	Pass	Reference ID:	30
Vulnerability Name:	Testing for Session Fixation				

Description:

When an application does not renew its session cookie(s) after a successful user authentication, it could be possible to find session fixation vulnerability and force a user to utilize a cookie known by the attacker. In that case, an attacker could steal the user session (session hijacking).

Details:

Session fixation vulnerabilities occur when:

• A web application authenticates a user without first invalidating the existing session ID, thereby continuing to use the session ID already associated with the user.

• An attacker is able to force a known session ID on a user so that, once the user authenticates, the attacker has access to the authenticated session.

• In the generic exploit of session fixation vulnerabilities, an attacker creates a new session on a web application and records the associated session identifier. The attacker then causes the victim to authenticate against the server using the same session identifier, giving the attacker access to the user's account through the active session.

• Furthermore, the issue described above is problematic for sites which issue a session identifier over HTTP and then redirect the user to a HTTPS login form. If the session identifier is not reissued upon authentication, the identifier may be eavesdropped and may be used by an attacker to hijack the session.

Reference:

http://shiflett.org/articles/session-fixation
http://www.acrossecurity.com/papers/session_fixation.pdf
https://www.owasp.org/index.php/Session_Fixation
http://vulncat.fortifysoftware.com/
http://www.cookiecentral.com/faq/#3.3
http://en.wikipedia.org/wiki/Session_fixation

Recommendation:

Some platforms make it easy to protect against Session Fixation, while others make it a lot more difficult. In most cases, simply discarding any existing session is sufficient to force the framework to issue a new sessionid cookie, with a new value. Unfortunately, some platforms, notably Microsoft ASP, do not generate new values for sessionid cookies, but rather just associate the existing value with a new session. This guarantees that almost all ASP apps will be vulnerable to session fixation, unless they have taken specific measures to protect against it. The idea is that, since ASP prohibits write access to the ASPSESSIONIDxxxxx cookie, and will not allow us to change it in any way, we have to use an additional cookie that we do have control over to detect any tampering. So, we set a cookie in the user's browser to a random value, and set a session variable to the same value. If the session variable and the cookie value ever don't match, then we have a potential fixation attack, and should invalidate the session, and force the user to log on again.

Risk :	Informational	Status:	Pass	Reference ID:	31
Vulnerability Name:	Testing for Privilege Escalation				

Description:

It is possible to escalate privilege due to improper authorization.

Details:

Privilege escalation occurs when a user gets access to more resources or functionality than they are normally allowed, and such elevation/changes should have been prevented by the application. This is usually caused by a flaw in the application. The result is that the application performs actions with more privileges than those intended by the developer or system administrator. The degree of escalation depends on which privileges the attacker is authorized to possess, and which privileges can be obtained in a successful exploit. For example, a programming error that allows a user to gain extra privilege after successful authentication limits the degree of escalation, because the user is already authorized to hold some privilege. Likewise, a remote attacker gaining superuser privilege without any authentication presents a greater degree of escalation. Usually, we refer to vertical escalation when it is possible to access resources granted to more privileged accounts (e.g., acquiring administrative privileges for the application), and to horizontal escalation when it is possible to access resources granted to a similarly configured account (e.g., in an online banking application, accessing information related to a different user).

Reference:

http://en.wikipedia.org/wiki/Privilege_escalation
http://www.techrepublic.com/blog/security/mitigating-the-privilege-escalation-threat/3445
http://docs.oracle.com/cd/E19253-01/816-4557/privref-20/index.html
http://searchsecurity.techtarget.com/definition/privilege-escalation-attack
http://www.brighthub.com/computing/smb-security/articles/39675.aspx

Recommendation:

Validate session for user with admin privilege, super user privilege and normal user privilege each in different manner.

Risk :	Informational	Status:	Pass	Reference ID:	32
Vulnerability Name:	Testing for LDAP Injection				

Description:

LDAP injection can be performed and it is possible to retrieve username & password of users

Details:

LDAP is an acronym for Lightweight Directory Access Protocol. LDAP is a protocol to store information about users, hosts, and many other objects. LDAP injection is a server side attack, which could allow sensitive information about users and hosts represented in an LDAP structure to be disclosed, modified, or inserted. This is done by manipulating input parameters afterwards passed to internal search, add, and modify functions. A web application could use LDAP in order to let users authenticate or search other users' information inside a corporate structure. The goal of LDAP injection attacks is to inject LDAP search filters meta characters in a query which will be executed by the application.

A successful exploitation of LDAP injection vulnerability could allow the attacker to:

- Access unauthorized content
- Evade application restrictions
- Gather unauthorized information
- Add or modify Objects inside LDAP tree structure.

Reference:

http://www.networkdls.com/articles/ldapinjection.pdf
http://www.redbooks.ibm.com/redbooks/SG244986/wwhelp/wwhimpl/js/html/wwhelp.htm

Recommendation:

The escape sequence for properly using user supplied input into LDAP differs depending on if the user input is used to create the DN (Distinguished Name) or used as part of the search filter. The listing below shows the character that needs to be escape and the appropriate escape method for each case.

Used in DN - Requires \ escape Used in Filter- Requires {\ASCII} escape

& ({\28}
!) {\29}
| \ {\5c}
= * {\2a}
< / {\2f}
> NUL {\0}
,
+
-
"
'
;

Risk :	Informational	Status:	Pass	Reference ID:	33
Vulnerability Name:	Testing for HTTP Splitting/Smuggling				

Description:

It is possible to attacks that leverage specific features of the HTTP protocol, either by exploiting weaknesses of the web application or peculiarities in the way different agents interpret HTTP messages.

Details:

HTTP Smuggling or HTTP response smuggling is a technique to "smuggle" 2 HTTP responses from a server to a client, through an intermediary HTTP device that expects (or allows) a single response from the server. HTTP Splitting (or HTTP Response splitting) is method of attacking web applications by exploiting poor input validation and by taking advantage of the HTTP protocol. We will analyze two different attacks that target specific HTTP headers: HTTP splitting and HTTP smuggling. The first attack exploits a lack of input sanitization which allows an intruder to insert CR and LF characters into the headers of the application response and to 'split' that answer into two different HTTP messages. The goal of the attack can vary from a cache poisoning to cross site scripting. In the second attack, the attacker exploits the fact that some specially crafted HTTP messages can be parsed and interpreted in different ways depending on the agent that receives them. HTTP smuggling requires some level of knowledge about the different agents that are handling the HTTP messages (web server, proxy, firewall) and therefore will be included only in the Gray Box testing section.

Reference:

https://www.owasp.org/images/1/1a/OWASPAppSecEU2006_HTTPMessageSplittingSmugglingEtc.ppt
http://www.securityfocus.com/archive/1/411418
http://packetstormsecurity.com/papers/general/whitepaper_httpresponse.pdf
http://www-142.ibm.com/software/products/us/en/subcategory/SWI10

Recommendation:

Many applications do not plan input validation, and leave it up to the individual developers. This is a recipe for disaster, as different developers will certainly all choose a different approach, and many will simply leave it out in the pursuit of more interesting development. Applications should NOT use as variables any user personal information (user name, password, home address, etc.). Highly protected applications should not implement mechanisms that make automated requests to prevent session timeouts. Highly protected applications should not implement "remember me" functionality. Highly protected applications should not use URL rewriting to maintain state when cookies are turned off on the client. Applications should NOT use session identifiers for encrypted HTTPS transport that have once been used over HTTP.

Risk :	Informational	Status:	Pass	Reference ID:	34
Vulnerability Name:	Testing for SQL Wildcard Attacks				

Description:

SQL wildcard attack results in the unavailability of the service for legitimate user.

Details:

SQL Wildcard Attacks are about forcing the underlying database to carry out CPU-intensive queries by using several wildcards. This vulnerability generally exists in search functionalities of web applications. Successful exploitation of this attack will cause Denial of Service. SQL Wildcard attacks might affect all database back-ends but mainly affect SQL Server because the MS SQL Server LIKE operator supports extra wildcards such as "[]","[^]","_" and "%". In a typical web application, if you were to enter "foo" into the search box, the resulting SQL query might be:

SELECT * FROM Article WHERE Content LIKE '%foo%' In a decent database with 1-100000 records the query above will take less than a second. The following query, in the very same database, will take about 6 seconds with only 2600 records.

SELECT TOP 10 * FROM Article WHERE Content LIKE
'%_[^!_%/%a?F%_D)_(F%)_%([)({}%){()}£$&N%_)$*£()$*R"_)][%](%[x])%a][$*"£$-9]_%'

So, if the tester wanted to tie up the CPU for 6 seconds they would enter the following to the search box:
_[^!_%/%a?F%_D)_(F%)_%([)({}%){()}£$&N%_)$*£()$*R"_)][%](%[x])%a][$*"£$-9]_

Reference:

http://hax.tor.hu/read/MSSQL_DoS/wildcard_attacks.pdf
http://labs.portcullis.co.uk/application/dos-attacks-using-sql-wildcards/
http://www.zdnet.com/blog/security/dos-attacks-using-sql-wildcards-revealed/1134

Recommendation:

SQL wildcard attacks can be prevented by escaping the wildcards (% and _) when using LIKE statements. SQL can make the wildcards escape by using '[]'.

Risk :	Informational	Status:	Pass	Reference ID:	35
Vulnerability Name:	Locking Customer Accounts				

Description:

An attacker can lock valid user accounts by repeatedly attempting to log in with a wrong password.

Details:

The first DoS case to consider involves the authentication system of the target application. A common defence to prevent brute-force discovery of user passwords is to lock an account from use after between three to five failed attempts to login. This means that even if a legitimate user were to provide their valid password, they would be unable to log in to the system until their account has been unlocked. This defence mechanism can be turned into a DoS attack against an application if there is a way to predict valid login accounts.

Reference:

http://www.computerhope.com/jargon/a/accolock.htm
http://www.windowsecurity.com/articles-tutorials/authentication_and_encryption/Implementing-Troubleshooting-Account-Lockout.html
https://www.owasp.org/index.php/Blocking_Brute_Force_Attacks

Recommendation:

There are pros and cons to locking accounts, to customers being able to choose their own account names, to using systems such as CAPTCHA, and the like. Each enterprise will need to balance these risks and benefits.

Risk :	Informational	Status:	Pass	Reference ID:	36
Vulnerability Name:	WS Information Gathering				

Description:

The WS entry points and the communication schema is found which might be a vulnerability at present or in future.

Details:

The input fields can be the following three. Any attacks can be initiated from any one of the three application entry points. They are GET, POST and html tags. The GET and POST methods are used to transfer any information from one web page to the other. The GET method is usually used to get information from the web page, which will be seen in the URL. The POST method is usually used to get information from the form to a web page or self. The main difference between GET and POST is that, GET is visible in the URL and POST is not. However both the GET and POST can be viewed. This GET and POST can be used to get information about the application entry points. The third method which is the entry point through analyzing HTML tags. HTML tags like <input>, <select>, <options> are used to get inputs from the user. So these are attracted by attacker. Also the input tag with hidden field always contains sensitive information. So these are analyzed to gather information about the application entry points.

Reference:

http://social.msdn.microsoft.com/Forums/en-US/sharepointdevelopment/thread/75415586-502d-475c-b2ab-d6df97ae4c17
http://www.w3schools.com/tags/ref_httpmethods.asp
http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html
http://www.w3.org/2001/tag/doc/whenToUseGet-20040321

Recommendation:

Use GET if the interaction is more like a question (i.e., it is a safe operation such as a query, read operation, or lookup). Use POST if the interaction is more like an order, or the interaction changes the state of the resource in a way that the user would perceive (e.g., a subscription to a service), or the user be held accountable for the results of the interaction. You should never change anything in your database (other than logging information or other ephemeral data) from a GET request. The issue is that there is various web spidering software, web accelerators, anti-virus programs, and the like, that will perform a GET request on every URL they find; you would not want them to delete items automatically when they do so. GET is also vulnerable to cross-site request forgery; if an attacker makes one of your users click on a link that performs a bad action (for instance, creating a tinyurl that redirects to a delete URL), then they can trick the user into using their permissions to delete something without realizing it. Making a field "hidden" has pretty much nothing to do with security, and should be considered a UI decision. Any "hacker" will read your HTML source anyway. Better to either not show sensitive information at all, or, if you must, to use SSL (to prevent data interception by network intermediaries) and some combination of login challenges (to prevent unauthorized access).

Risk :	Informational	Status:	Pass	Reference ID:	37
Vulnerability Name:	WSDL Testing				

Description:

Web Service Definition Language (WSDL) discloses most of the information about the working and the data flow of the application.

Details:

The Web services architecture may require exposing a WSDL file that contains information on the publicly accessible services and how callers of these services should interact with them (e.g. what parameters they expect and what types they return). The attacker may find sensitive information located in the WSDL file. The WSDL file is accessible to a wider audience than intended.

• The WSDL file contains information on the methods/services that should not be publicly accessible or information about deprecated methods.
• This problem is made more likely due to the WSDL often being automatically generated from the code.
• Information in the WSDL file helps guess names/locations of methods/resources that should not be publicly accessible.

The WSDL for a service providing information on the best price of a certain item exposes the following method: float getBestPrice(String ItemID) An attacker might guess that there is a method setBestPrice (String ItemID, float Price) that is available and invoke that method to try and change the best price of a given item to their advantage. The attack may succeed if the attacker correctly guesses the name of the method, the method does not have proper access controls around it and the service itself has the functionality to update the best price of the item.

Reference:

http://www.w3.org/TR/wsdl
http://www.w3schools.com/wsdl/
http://en.wikipedia.org/wiki/Web_Services_Description_Language

Recommendation:

1. Limit access to the WSDL file as much as possible. If services are provided only to a limited number of entities, it may be better to provide WSDL privately to each of these entities than to publish WSDL publicly.
2. Make sure that WSDL does not describe methods that should not be publicly accessible. Make sure to protect service methods that should not be publicly accessible with access controls.
3. Do not use method names in WSDL that might help an adversary guess names of private methods/resources used by the service.

Risk :	Informational	Status:	Pass	Reference ID:	38
Vulnerability Name:	Weak XML Structure Testing				

Description:

Weak XML structure can even cause DOS threat to the application.

Details:

XML needs to be well-formed to function properly. XML which is not well-formed shall fail when parsed by the XML parser on the server side. A parser needs to run thorough the entire XML message in a serial manner in order to assess the XML well-formedness. An XML parser is also very CPU labour intensive. Some attack vectors exploit this weakness by sending very large or malformed XML messages. Attackers can create XML documents which are structured in such a way as to create a denial of service attack on the receiving server by tying up memory and CPU resources. This occurs via overloading the XML parser ,which, as we mentioned, is very CPU-intensive.
For example, elements which contain large numbers of attributes can cause problems with parsers. This category of attack also includes XML documents which are not well-formed XML, DOM-based parsing can be vulnerable to DoS due to the fact that the complete message is loaded into memory.

Reference:

http://www.w3schools.com/schema/schema_intro.asp
http://msdn.microsoft.com/en-us/library/ms187508(v=sql.90).aspx
http://www.xfront.com/BestPracticesHomepage.html

Recommendation:

- Define your XML and encoding
- Use a DTD or XSD
- Remember to validate
- Validation isn't always the answer
- XML structure versus attributes
- Use XPath to find information
- You don't always need a parser to extract information
- When to use SAX over DOM parsing
- When to DOM over SAX parsing
- Use a good XML editor

Risk :	Informational	Status:	Pass	Reference ID:	39
Vulnerability Name:	XML Content-Level Testing				

Description:

Insecure XML allows the attacker to do DoS and Buffer Overflow attack.

Details:

Web Services are designed to be publicly available to provide services to clients using the Internet as the common communication protocol. These services can be used to leverage legacy assets by exposing their functionality via SOAP using HTTP. SOAP messages contain method calls with parameters, including textual data and binary attachments, requesting the host to perform some function - database operations, image processing, document management, etc. Legacy applications exposed by the service may be vulnerable to malicious input that when previously limited to a private network was not an issue. In addition, because the server hosting the Web Service will need to process this data, the host server may be vulnerable if it is unpatched or otherwise unprotected from malicious content (e.g., plain text passwords, unrestricted file access).

An attacker can craft an XML document (SOAP message) that contains malicious elements in order to compromise the target system. Testing for proper content validation should be included in the web application-testing plan. Content-level attacks target the server hosting a web service and any applications that are utilized by the service, including web servers, databases, application servers, operating systems, etc. Content-level attack vectors include 1) SQL Injection or XPath injection 2) Buffer Overflow and 3) Command Injection.

Reference:

http://www.osvdb.org/
http://support.citrix.com/proddocs/topic/ns-security-10-map/appfw-checks-xml-sql-con.html
http://carnal0wnage.attackresearch.com/2008/12/so-this-has-been-interesting-week.html

Recommendation:

- Define your XML and encoding
- Use a DTD or XSD
- Remember to validate
- Validation isn't always the answer
- XML structure versus attributes
- Use XPath to find information
- You don't always need a parser to extract information
- When to use SAX over DOM parsing
- When to DOM over SAX parsing
- Use a good XML editor