

5/8/2018

Web Application Security Audit Report

<http://npccindia.com/>



Recon Business Advisory Pvt. Ltd.
F-8,3RD FLOOR, KALKAJI MAIN ROAD, NEW DELHI
PIN-110019.

Confidential

Confidential : The content of this document is confidential and may not be used by parties other than without authorization

Document and Control information

Item	Client Description
Document Name	Web Application Security Audit Report of NPCC
Client Name	Global Infosys
Audit Duration	07 th Aug to 08 th Aug 2018
Initial Report Date	08 th Aug 2018
Status Report Date	NA
Closing Report Date	NA
Report Version	1.0
Hand Over to	Shikha

Item	Company Description
Author	Recon Business Advisory Pvt Ltd.
Auditor Name	Shudhanshu Singh
Reviewed By	Rishesh Kumar Gupta

Table of Contents

1. Executive Summary.....	5
2. Intended Audience.....	5
3. Assessment Objectives	5
4. Application Credentials and URL	5
5. Assessment Methodology	6
6. OWASP Top 10 Application Security Risks.....	8
7. Assessment Scope.....	10
8. Issues Encountered.....	10
9. Key Findings	11
10. Vulnerability Details.....	12
10.1 Sql Injection	12
Error based SQL injection	12
10.2 Unrestricted File Upload.....	14
Malicious File Upload	14
10.3 Using Component with Known Vulnerability.....	16
Vulnerable Asp.Net Version	16
10.4 Microsoft Tild directory enumeration	17
Microsoft Tild directory enumeration.....	17
10.5 Cookie without HttpOnly Flag set.....	18
Asp .net Version disclosure	18
10.6 Using Known Vulnerable component	20
Vulnerable JQuery.....	20

1. Executive Summary

Recon Business Advisory Pvt. Ltd. carried out Web Application security assessment of the following website **07th AUG to 08th AUG 2018** from Recon, New Delhi.

<http://npccindia.com/>

2. Intended Audience

This document is primarily meant for the Global Infosys client. Further distribution of this document entirely lies to the discretion of the Global Infosys client.

3. Assessment Objectives

The security assessment of the application <http://npccindia.com/> was requested. The application primarily is meant for critical business transactions in Global Infosys client.

The purpose of this assessment is to discover the vulnerabilities in web application and to indicate the subsequent risk level associated with the vulnerabilities.

4. Application Credentials and URL

The security assessment was done on the following URL/s:

<http://npccindia.com/>

5. Assessment Methodology

A hybrid approach is followed to perform the assessment that is a combination of tools is used to discover the wide range of vulnerabilities. Additionally, the assessment being adaptive in nature allows us to control the assessment methodology as per the application functionality to focus on the critical areas of the application. The attack vectors are controlled as per the assessment needs and the attack selection ensures maximum coverage of the application.

Following diagram represents the assessment approach:






The table below describes various levels and types of assessment. The type of assessment done for current assessment is available in the “Assessment Scope” section of the document.

Scan/Audit Type		
Level	Type	Information
1	Safe	Safe scan discovers minimum types and instances of vulnerabilities. The safe scan mode avoid fault injection such as Java Scripts, HTML tags, crafted SQL queries etc. to ensure that the application retains its state at the end of the assessment. Any fault injections that may trigger Denial of Service situation are avoided in safe scans. Safe scan suits most when the assessment is to be done on a live application instance, and has already undergone either <i>Standard</i> or <i>Destructive</i> scan/s.
2	Standard	Standard scan discovers and exploits most standard checks such as OWASP Top 10 checks. The standard scan performs fault injection such as Java Scripts injection, HTML tag injection, crafted SQL queries etc. Any fault injections that may trigger Denial of Service situation are avoided in standard scans. Standard scan suits most when the assessment is to be done on a staging/pre-prod/testing application instance.
3	Destructive	Destructive scan discovers and exploits most comprehensive checks including checks that may trigger Denial of Service Attacks situations for the application. Destructive scan is usually done on staging/pre-prod/testing application instance. A destructive scan on a live environment is avoided on live/production systems unless it is really required.

The vulnerabilities discovered are associated with a risk level that indicates how critical the vulnerability is and helps application owners/developers to prioritize the vulnerabilities and choose an appropriate mitigation approach.

Risk Level Information and Necessary Actions

Risk Level	Risk Description and Necessary Action
	<p>The high risk level indicates maximum risk associated with a specific vulnerability instance. Such vulnerability may enable an attacker to successfully exploit the underlying application and its data and partially or completely to compromise the application and its data to modify application behaviour to become other than its original intended purpose. The vulnerability marked as “High Risk” is recommended to be handled with utmost priority.</p>
	<p>The medium risk level indicates considerable risk associated with a specific vulnerability instance. Such vulnerability may enable an attacker to exploit the underlying application and its data to a particular level so that the attacker can gain low level information about the application. Such information can be used by an attacker to craft more specific attacks based on the information collected. The vulnerability marked with “Medium Risk” should be mitigated at the earliest or soon after “High Risk” vulnerabilities are mitigated.</p>
	<p>The low risk level indicates lowest risk associated with a specific vulnerability instance. Such vulnerability may allow an attacker to gain some information about the application which was not intended to be known otherwise. The attacker may not have exploiting techniques available at that instance based on the information revealed by the system. The vulnerability marked with “Low Risk” can be mitigated soon after high and medium risk vulnerabilities are mitigated.</p>

Ease of Exploit level information

6. OWASP Top 10 Application Security Risks

Open Web Application Security Project (OWASP) is a not-for-profit worldwide charitable organization focused on improving the security of application software.

The table below lists Top 10 identified security risks by OWASP:

	Risk	Information
A1	Injection	Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker’s hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
A2	Broken Authentication and Session Management	Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users’ identities.
A3	Cross-Site Scripting (XSS)	XSS flaws occur whenever an application takes untrusted data

		and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
A4	Insecure Direct Object References	A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.
A5	Security Misconfiguration	Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.
A6	Sensitive Data Exposure	Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.
A7	Missing Function Level Access Control	Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.
A8	Cross-Site Request Forgery (CSRF)	A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
A9	Using Components with Known Vulnerabilities	Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.
A10	Unvalidated Redirects and Forwards	Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

More information about OWASP can be found at: <http://www.owasp.org>

7. Assessment Scope

The application security assessment is done on but not limited to the following controls:

- Authentication
- Authorization
- Session Management
- Input Validation
- Error Handling
- Cryptography

Following were not a part of this assessment:

- DOS

Scan type:

Assessment type	Selected
Safe checks	yes
Standard/OWASP Top 10	Yes
Destructive	No

8. Issues Encountered

There were no issues encountered for this application.

9. Key Findings

No.	Vulnerability Name	Risk	Status
1.	Sql Injection	High	Open
2.	Malicious File upload	High	Open
3.	Asp .net vulnerable version	High	Open
4.	Microsoft Tild directory enumeration	High	Open
4.	Cookie Without HttpOnly Flag Set	Low	Open
5.	Vulnerable jquery	Low	Open

10. Vulnerability Details

10.1 Sql Injection

Name of Vulnerability	Error based SQL injection
URL	http://npccindia.com/admin/Changepassword.aspx
Risk	High
CVE/CWE-ID	CWE-209,89

Description:

SQL injection vulnerabilities arise when user-controllable data is incorporated into database SQL queries in an unsafe manner. An attacker can supply crafted input to break out of the data context in which their input appears and interfere with the structure of the surrounding query.

A wide range of damaging attacks can often be delivered via SQL injection, including reading or modifying critical application data, interfering with application logic, escalating privileges within the database and taking control of the database server.

Proof of Concept:

Incorrect syntax near 'Hack'.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated.

Exception Details: System.Data.SqlClient.SqlException: Incorrect syntax near 'Hack'.

Source Error:

The source code that generated this unhandled exception can only be shown when compiled in debug mode. To enable this request the URL:

1. Add a "Debug=true" directive at the top of the file that generated the error. Example:

```
<%@ Page Language="C#" Debug="true" %>
```

or:

- 2) Add the following section to the configuration file of your application:

```
<configuration>  
  <system.web>  
    <compilation debug="true"/>  
  </system.web>  
</configuration>
```

Note that this second technique will cause all files within a given application to be compiled in debug mode. This is not recommended for production scenarios.

Important: Running applications in debug mode does incur a memory/performance overhead. You should make sure that you are not deploying into production scenario.

Stack Trace:

```
[SqlException (0x80131904): Incorrect syntax near 'Hack'.]  
System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection) +212  
System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj) +249  
System.Data.SqlClient.TdsParser.Run(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataSt  
System.Data.SqlClient.SqlDataReader.ConsumeMetaData() +127  
System.Data.SqlClient.SqlDataReader.get_MetaData() +112  
System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, RunBehavior runBehavior, String r  
System.Data.SqlClient.SqlCommand.RunExecuteReaderTds(CommandBehavior cmdBehavior, RunBehavior runBehavio  
System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior,  
System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior,  
System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior behavior, String method) +211  
System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior behavior) +19  
System.Data.Common.DbCommand.System.Data.IDbCommand.ExecuteReader(CommandBehavior behavior) +19  
System.Data.Common.DbDataAdapter.FillInternal(DataSet dataset, DataTable[] datatables, Int32 startRecord  
System.Data.Common.DbDataAdapter.Fill(DataSet dataSet, Int32 startRecord, Int32 maxRecords, String srcTa  
System.Data.Common.DbDataAdapter.Fill(DataSet dataSet) +166  
Connection.Retrieve2(String cmd, DataSet& ds) +272  
ChangePassword.ImageButton1_Click(Object sender, ImageClickEventArgs e) +590  
System.Web.UI.WebControls.ImageButton.OnClick(ImageClickEventArgs e) +98
```

Recommendation:

The most effective way to prevent SQL injection attacks is to use parameterized queries (also known as prepared statements) for all database access. This method uses two steps to incorporate potentially tainted data into SQL queries: first, the application specifies the structure of the query, leaving placeholders for each item of user input; second, the application specifies the contents of each placeholder. Because the structure of the query has already been defined in the first step, it is not possible for malformed data in the second step to interfere with the query structure. You should review the documentation for your database and application platform to determine the appropriate APIs which you can use to perform parameterized queries. It is strongly recommended that you parameterize every variable data item that is incorporated into database queries, even if it is not obviously tainted, to prevent oversights occurring and avoid vulnerabilities being introduced by changes elsewhere within the code base of the application.

You should be aware that some commonly employed and recommended mitigations for SQL injection vulnerabilities are not always effective:

- One common defense is to double up any single quotation marks appearing within user input before incorporating that input into a SQL query. This defense is designed to prevent malformed data from terminating the string into which it is inserted. However, if the data being incorporated into queries is numeric, then the defense may fail, because numeric data may not be encapsulated within quotes, in which case only a space is required to break out of the data context and interfere with the query. Further, in second-order SQL injection attacks, data that has been safely escaped when initially inserted into the database is subsequently read from the database and then passed back to it again. Quotation marks that have been doubled up initially will return to their original form when the data is reused, allowing the defense to be bypassed.
- Another often cited defense is to use stored procedures for database access. While stored procedures can provide security benefits, they are not guaranteed to prevent SQL injection attacks. The same kinds of vulnerabilities that arise within standard dynamic SQL queries can arise if any SQL is dynamically constructed within stored procedures. Further, even if the procedure is sound, SQL injection can arise if the procedure is invoked in an unsafe manner using user-controllable data.

10.2 Unrestricted File Upload

Name of Vulnerability	Malicious File Upload
URL	http://npccindia.com/
Risk	High
CVE/CWE-ID	CWE-434

Description:

The HTTP responses returned by this web application include an header named X-AspNet-Version. The value of this header is used by Visual Studio to determine which version of ASP.NET is in use. It is not necessary for production sites and should be disabled.

ASP.NET debugging is enabled on this application. It is recommended to disable debug mode before deploying a production application. By default, debugging is disabled, and although debugging is frequently enabled to troubleshoot a problem, it is also frequently not disabled again after the problem is resolved.

Proof of Concept:

<http://npccindia.com/admin/Addfile.aspx>



Uploading a pdf with malicious script:

**"><b onmouseover=prompt('Hack')>click me! **

The image shows a web browser window with the URL `npccindia.com/admin/Addfile.aspx`. The browser's developer tools are open, showing the raw request data. A red box highlights a PDF file with a JavaScript payload: `<b onmouseover=prompt('Hack')>click me! `. The browser shows a success message "File successfully uploaded." and a path disclosure: `Writereaddata/data/Downloads/java56766.pdf`.

File Successfully uploaded and path disclosure as well
Writereaddata/data/Downloads/java56766.pdf

Recommendation:

Based on this context, the goals here are:

- For Word/Excel/Powerpoint/Pdf documents: Detect when a document contains "code"/OLE package, if it's the case then block the upload process.

- For Images document: Sanitize incoming image using re-writing approach and then disable/remove any "code" present (this approach also handle case in which the file sent is not an image).

Remarks:

- It's technically possible to perform sanitizing on Word/Excel/Powerpoint/PDF documents but we have chosen here the option to block them in order to avoid the risk of missing any evasion technics and then let pass one evil document. The following site show how many way exists to embed Macro into a Microsoft Office documents.
- The other reason why we have chosen the blocking way is that for Word/Excel/Powerpoint, changing document format (for example by saving any document to DOCX/XSLX/PPTX/PPSX formats in order to be sure that no Macro can be executed) can have impacts or cause issues on document structure/rendering depending on the API used.

https://www.owasp.org/index.php/Protect_FileUpload_Against_Malicious_File

https://www.owasp.org/index.php/Unrestricted_File_Upload

10.3 Using Component with Known Vulnerability

Name of Vulnerability	Vulnerable Asp.Net Version
URL	http://npccindia.com/
Risk	High
CVE/CWE-ID	CWE-200,

Description:

The HTTP responses returned by this web application include anheader named X-AspNet-Version. The value of this header is used by Visual Studio to determine which version of ASP.NET is in use. It is not necessary for production sites and should be disabled.

ASP.NET debugging is enabled on this application. It is recommended to disable debug mode before deploying a production application. By default, debugging is disabled, and although debugging is frequently enabled to troubleshoot a problem, it is also frequently not disabled again after the problem is resolved.

Proof of Concept:

← → ↻ ⓘ Not secure | npccindia.com/.aspx

Server Error in '/' Application.

The resource cannot be found.

Description: HTTP 404. The resource you are looking for (or one of its dependencies) could have been removed, had its name changed, or is temporarily unavailable. Please review the fo

Requested URL: /.aspx

Version Information: Microsoft .NET Framework Version:2.0.50727.8789; ASP.NET Version:2.0.50727.8762

Server Error in '/' Application.

Object reference not set to an instance of an object.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.NullReferenceException: Object reference not set to an instance of an object.

Source Error:

The source code that generated this unhandled exception can only be shown when compiled in debug mode. To enable this, please follow one of the below steps, then request the URL:

1. Add a "Debug=true" directive at the top of the file that generated the error. Example:


```
<%@ Page Language="C#" Debug="true" %>
```

 or:
- 2) Add the following section to the configuration file of your application:


```
<configuration>
  <system.web>
    <compilation debug="true"/>
  </system.web>
</configuration>
```

Note that this second technique will cause all files within a given application to be compiled in debug mode. The first technique will cause only that particular file to be compiled in debug mode.

Important: Running applications in debug mode does incur a memory/performance overhead. You should make sure that an application has debugging disabled before deploying into production scenario.

Stack Trace:

[Null]ReferenceException: Object reference not set to an instance of an object.

Admin_Login.ImageButton_Click(Object sender, ImageClickEventArgs e) #400

Recommendation:

A specific policy for how to handle errors should be documented, including the types of errors to be handled and for each, what information is going to be reported back to the user, and what information is going to be logged. All developers need to understand the policy and ensure that their code follows it.

- Update to the latest version.

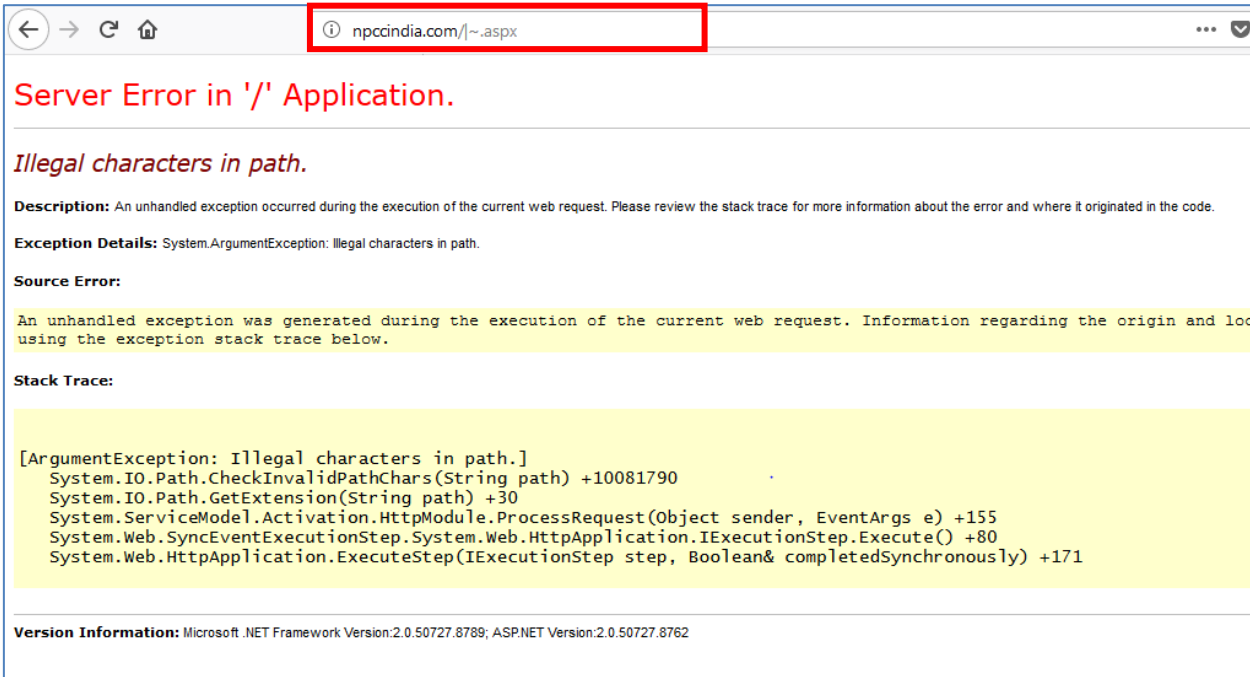
10.4 Microsoft Tild directory enumeration

Name of Vulnerability	Microsoft Tild directory enumeration
URL	http://npccindia.com/
Risk	Medium
CVE/CWE-ID	CWE-200

Description:

It is possible to detect short names of files and directories which have an 8.3 file naming scheme equivalent in Windows by using some vectors in several versions of Microsoft IIS. For instance, it is possible to detect all short-names of ".aspx" files as they have 4 letters in their extensions. This can be a major issue especially for the .Net websites which are vulnerable to direct URL access as an attacker can find important files and folders that they are not normally visible.

Proof of Concept:



Recommendation:

Kindly follow the below link

<https://support.detectify.com/customer/portal/articles/1711520-microsoft-iis-tilde-vulnerability>

10.5 Cookie without HttpOnly Flag set

Name of Vulnerability	Asp .net Version disclosure
URL	http://npccindia.com/
Risk	Low
CVE/CWE-ID	CWE-16

Description:

If the HttpOnly attribute is set on a cookie, then the cookie's value cannot be read or set by client-side JavaScript. This measure makes certain client-side attacks, such as cross-site scripting, slightly harder to exploit by preventing them from trivially capturing the cookie's value via an injected scrip

Proof of Concept:

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Vary: Accept-Encoding
X-Frame-Options: DENY
X-AspNet-Version: 4.0.30319
Set-Cookie: AuthToken=5e9fdbel-8871-483c-a225-
f45a81fc81fd; path=/
Set-Cookie: AuthToken=eaa29202-2dd9-434b-be7e-
f821d2ce472a; path=/
Set-Cookie: AuthToken=98f8807d-d35a-49fa-9d6f-
435824e922f5; path=/
Set-Cookie: AuthToken=7b41078e-4ebf-4dde-a26a-
af1e76a44c29; path=/
X-Powered-By: ASP.NET
Date: Mon, 23 Jul 2018 07:14:58 GMT
Content-Length: 46880
Original-Content-Encoding: gzip
```

Recommendation:

There is usually no good reason not to set the HttpOnly flag on all cookies. Unless you specifically require legitimate client-side scripts within your application to read or set a cookie's value, you should set the HttpOnly flag by including this attribute within the relevant Set-cookie directive.

You should be aware that the restrictions imposed by the HttpOnly flag can potentially be circumvented in some circumstances, and that numerous other serious attacks can be delivered by client-side script injection, aside from simple cookie stealing.

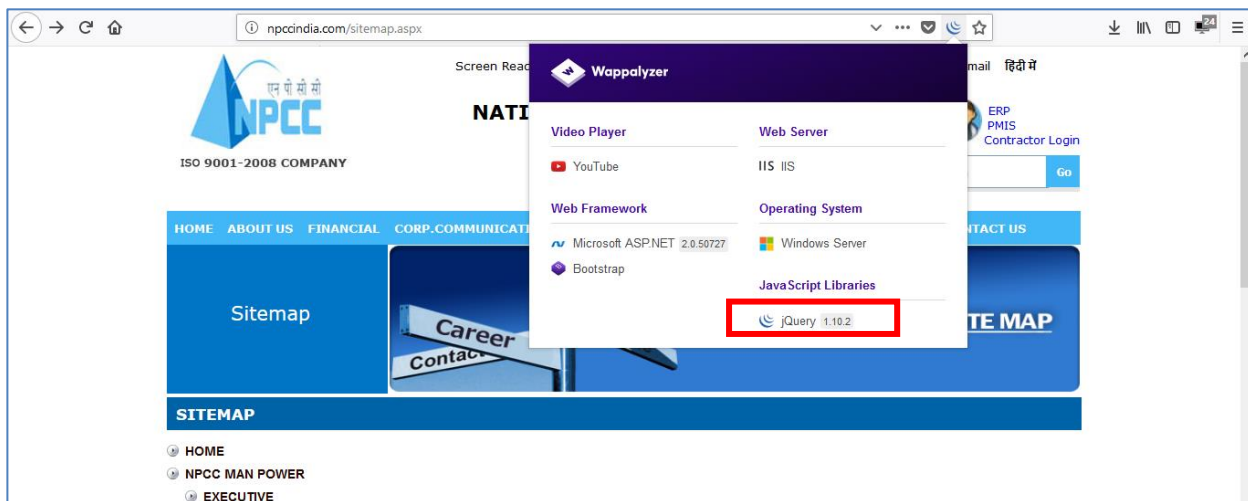
10.6 Using Known Vulnerable component

Name of Vulnerability	Vulnerable JQuery
URL	http://npccindia.com/
Risk	Medium
CVE/CWE-ID	CWE- 79

Description:

You are using a vulnerable Javascript library. One or more vulnerabilities were reported for this version of the Javascript library. Consult Attack details and Web References for more information about the affected library and the vulnerabilities that were reported.

Proof of Concept:



Recommendation:

Update to the latest version of jQuery.

Observation

1.Email id Hyperlink

[/npcc.hyd@gmail.com.](mailto:npcc.hyd@gmail.com)
[/npccsilchar@yahoo.com.](mailto:npccsilchar@yahoo.com)

Recommendation:

Using a graphic signature is better. Create a graphic that has your email ID written on it. Since the bots cannot read the graphic at this point, your ID will be safe. But when you do that, don't go ahead and link the graphic to your email ID. Linking graphic to email ID will nullify the purpose of that graphic email address or signature

Email: npcc [dot] hyd [at]gmail [dot]com

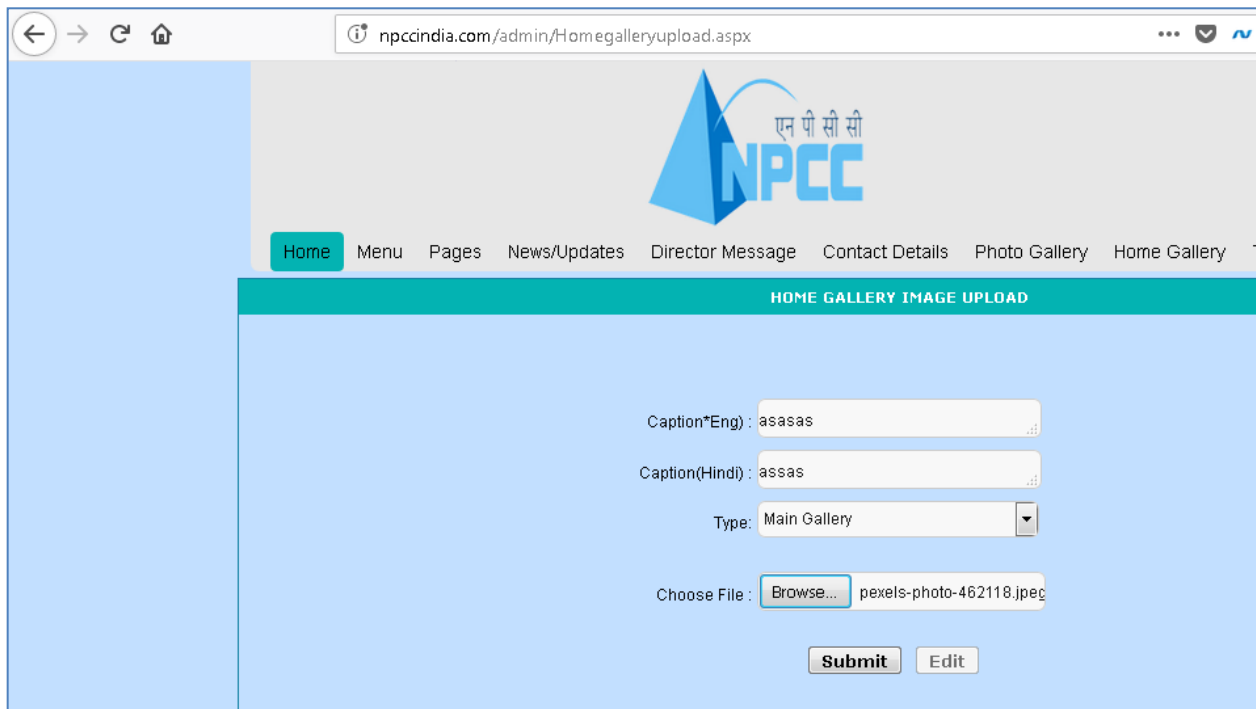
2. Using known Vulnerable Component.

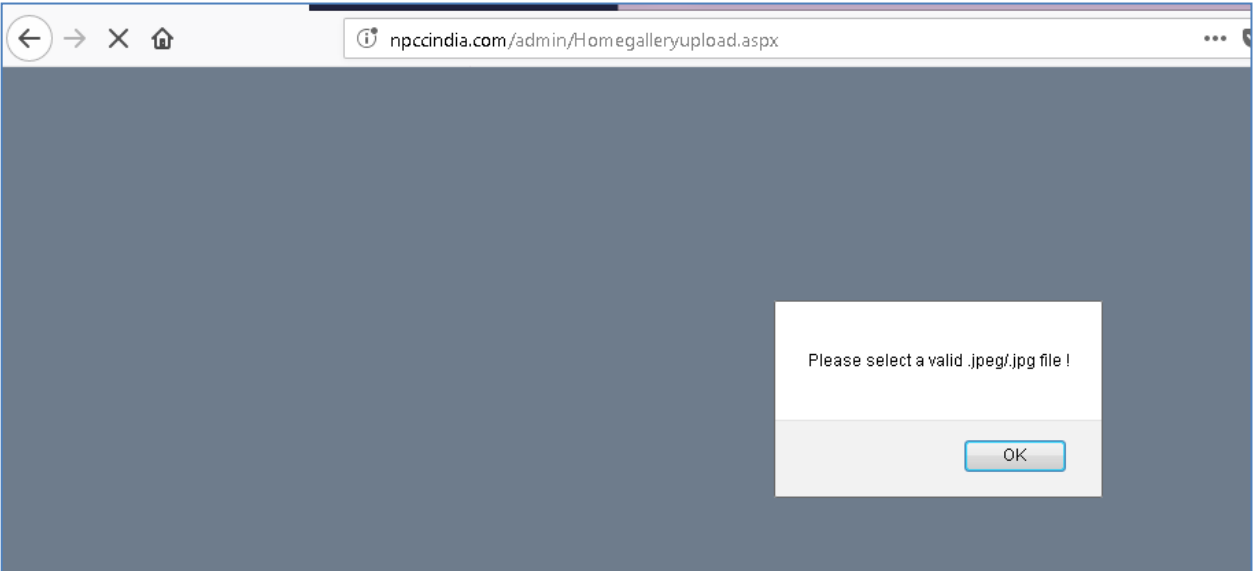
Update to the latest frame work and Library.

3. Application Flow Exception

<http://npccindia.com/admin/Homegalleryupload.aspx>

Even uploading a jpeg image throws us out of the application.





Assessment Limitations

The Web Application Assessment has been limited to the boundaries set in the contract and legal agreement. Following were the limitations while performing the web application audit assessment for Global Infosys client over the proxy server:

- The Web Application Assessment exercise is an attempt to identify the existing vulnerabilities present on the web server. The assessment is limited by the state of technology and functionality of software tools or products deployed at that point in time. Recon however does ensure that the tools and methodologies used are the best available and are the most recent versions.
- This exercise does not guarantee the successful exploitation of the vulnerabilities later on, which were identified during the scanning and identification phase. Evidences in terms of screenshots have been presented wherever possible in the report for all successful and unsuccessful tests. To ensure that configuration and application changes do not uncover new vulnerabilities and to utilize advantages in scanning techniques that may uncover vulnerabilities in existing systems regular web application assessments be carried out.
- Web Application Assessment exercise was only limited to the web application mentioned in the scope of the activity with the normal user access privileges. However, considering the other assets as a launch pad was not considered for the application audit exercise scope and having more access to the target systems as compared to normal access, could also develop the additional attack surface for these assets
- The tool used for automated/Manual Web Application Scanning and identifying vulnerabilities. The report developed and the recommendations documented are hence derived from the output of automated/Manual tool and based on the OWASP standard of securing the Web Applications.

Disclaimer

Any advice, opinion, measures or recommendations suggested or supplied by Recon shall not amount to any form of warranty or guarantee that the intended result will be achieved or that any steps taken by the Client pursuant to such advice, opinion, measures or recommendations will guarantee that the Client's IT systems will be free from harmful components or from unauthorized interception or interference. The Client shall be solely responsible for the management, conduct and operation of its business and affairs; including without limitation for deciding on its use of the Results, choosing to what extent it wishes to rely on the Results, and/or implementing the recommendations.