

Coding Guidelines - Security Practice

Version No. : 1.2

Date : 25-May-2016

Copyright Notice

This document contains proprietary information of Delhi Integrated Multi Modal Transit System Ltd. (DIMTS). No part of this document may be reproduced, stored, copied, or transmitted in any form or by means of electronic, mechanical, photocopying or otherwise, without the express consent of Delhi Integrated Multi Modal Transit System Ltd. (DIMTS). This document is intended for internal circulation only and not meant for external distribution.

Revision History

Version No.	Date	Prepared by / Modified by	Significant Changes
1.2	25-May-2016	QMG	Sec. 4 updated with OWASP website reference
1.1	12-May-2015	QMG	SEPG approval
1.0	16-Apr-2015	QMG	Baselined, based on the session done with Sr. Management of 16 th Apr'15
0.4	11-Mar-2015	QMG	Formatting done
0.3	24-Feb-2015	Chandra Prakash Chauhan	Insecure configuration changes
0.2	10-Dec-2014	Chandra Prakash Chauhan	Web config ,new session id
0.1	03-Nov-2014	Chandra Prakash Chauhan	Draft

Glossary / Abbreviation / Definition

Sl. No.	Glossary / Abbreviation / Definition	Description
1	VAPT	Vulnerability Assessment and Penetration Testing
2	SSL	Secure Sockets Layer
3	TLS	Transport Layer Security
4		
5		
6		
7		
8		
9		

Table of Contents

1	Introduction	4
1.1	Purpose	4
1.2	Scope	4
2	Overview of the Document	5
3	Approach	6
3.1	Observations	6
3.1.1	SQL Injection	6
3.1.2	Vertical Privilege Escalation	10
3.1.3	Cross Site Request Forgery (CSRF).....	12
3.1.4	Default login credentials	16
3.1.5	Insecure storage of sensitive information.....	17
3.1.6	Click jacking attack.....	18
3.1.7	Session replay	20
3.1.8	Insecure session management	22
3.1.9	Insecure data transmission	25
3.1.10	Password brute force attack possibility	25
3.1.11	View State not encrypted	30
3.1.12	Insecure value of path attribute in Cookie.....	31
3.1.13	Session cookie set without Secure flag	33
3.1.14	Information disclosure through error messages.....	34
3.1.15	Insecure configuration of session timeout.....	35
4	Reference(s)	37
5	Annexure(s).....	38

1 Introduction

1.1 Purpose

The purpose of this document is to define web application security assessments within DIMTS. Web application assessments are performed to identify potential or realized weaknesses as a result of inadvertent mis-configuration, weak authentication, insufficient error handling, sensitive information leakage, etc. Discovery and subsequent mitigation of these issues will limit the attack surface of services available both internally and externally as well as satisfy compliance with any relevant policies in place.

In today's fast forward world, denial of service, data breaches, and SQL injection attacks are growing faster than on-premises firewalls, DNS, and Infrastructure Security to learn about various solutions and techniques for defending your online network.

Virtually all businesses, most government agencies, and many individuals now have Web sites. The number of individuals and companies with Internet access is expanding rapidly, and all of these have graphical Web browsers. As a result, businesses are enthusiastic about setting up facilities on the Web for electronic commerce. But the reality is that the Internet and the Web are extremely vulnerable to compromises of various sorts. As businesses wake up to this reality, the demand for secure Web services grows.

The topic of Web security is a broad one and can easily fill a book (several are recommended at the end of this chapter). In this chapter, we begin with a discussion of the general requirements for Web security and then focus on two standardized schemes that are becoming increasingly important as part of Web commerce: SSL/TLS and SET.

SSL (Secure Sockets Layer) is a standard security technology for establishing an encrypted link between a server and a client—typically a web server (website) and a browser; or a mail server and a mail client (e.g., Outlook).

SSL allows sensitive information such as credit card numbers, social security numbers, and login credentials to be transmitted securely. Normally, data sent between browsers and web servers is sent in plain text—leaving you vulnerable to eavesdropping. If an attacker is able to intercept all data being sent between a browser and a web server they can see and use that information.

More specifically, SSL is a security protocol. Protocols describe how algorithms should be used; in this case, the SSL protocol determines variables of the encryption for both the link and the data being transmitted.

1.2 Scope

This document covers all web application security assessments requested by any individual, group or department for the purposes of maintaining the security posture, compliance, risk management, and change control of technologies in use at.

2 Overview of the Document

Vulnerability Assessment and Penetration Testing (VAPT) provides enterprises with a more comprehensive application evaluation than any single test alone. Using the Vulnerability Assessment and Penetration Testing (VAPT) approach gives an organization a more detailed view of the threats facing its applications, enabling the business to better protect its systems and data from malicious attacks. Vulnerabilities can be fixed easily once found. Using a VAPT provider enables IT security teams to focus on mitigating critical vulnerabilities while the VAPT provider continues to discover and classify vulnerabilities.

This process is used to analyze the identified vulnerabilities, combined with the information gathered about the IT environment, to devise a plan for penetrating into the network and systems.

Penetration Testing : In this process, the target systems are attacked and penetrated using the plan devised in the earlier process.

Privilege Escalation : After successful penetration into the system, this process is used to identify and escalate access to gain higher privileges, such as root access or administrative access to the system.

Result Analysis : This process is useful for performing a root cause analysis as a result of a successful compromise to the system leading to penetration, and devise suitable recommendations in order to make the system secure by plugging the holes in the system.

Reporting : All the findings that are observed during the vulnerability assessment and penetration testing process need to be documented, along with the recommendations, in order to produce the testing report to the management for suitable actions.

Cleanup : Vulnerability assessment and penetration testing involves compromising the system, and during the process, some of the files may be altered. This process ensures that the system is brought back to the original state, before the testing, by cleaning up (restoring) the data and files used in the target machines.

3 Approach

3.1 Observations

3.1.1 SQL Injection

Impacted URL(s) URL(s)

http://afcsdelhi.com/ETMSPortal/DeportOperations/Infraction/Bus_Infraction.aspx

Observation

It has been observed that application is vulnerable to SQL injection vulnerability throughout the application. For demonstration, we have shown an attack instance on "Bus_infraction.aspx" form at "Infraction Bus No" field. Further, for demonstration, some select commands to extract the following data from the database;

- Database version (MS SQL Server 2008)
- Database logged in user (sa)
- List of all current database (RGTool, ReportServer)

Risk/ Implication

A SQL injection attack consists of insertion or "injection" of SQL queries via the input data from the client to the application.

An attacker may use this vulnerability to reset passwords for any application user obtain complete control over the application. An attacker can also 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 present on the file system and in some cases issue commands to the operating system.

Since the affected application is critical for day to day fare collection operations, unauthorized access to sensitive information, mis-configuration, disruption of service and connectivity to the same would cause significant impact to operations.

Recommendation

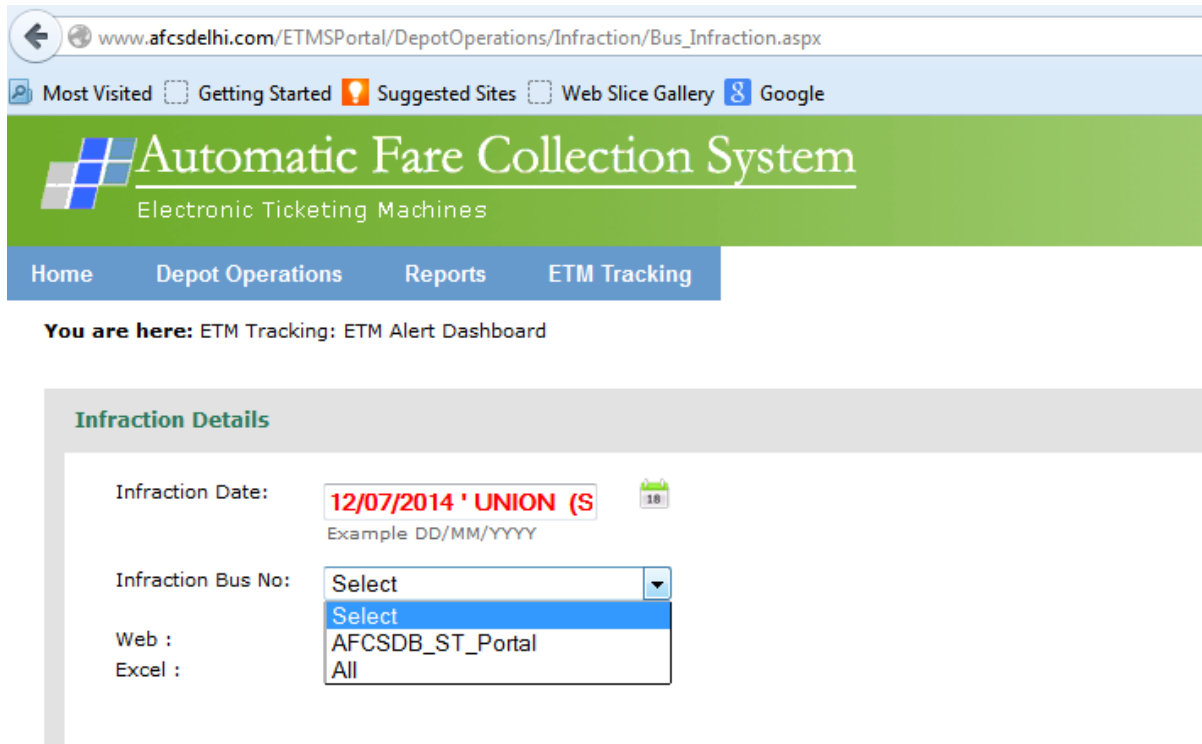
It is recommended to re-write the affected application code to ensure client and server side input validation and follow secure coding practices such as using parameterize query or calling stored procedures while executing query at database.

How to find:

An attacker may inject arbitrary SQL queries via the input data from the client and use this vulnerability to reset legitimate user passwords, execute administrative operations on the database, recover sensitive information and in some cases may also execute operating system level commands. It is recommended to re-write the affected application code to ensure client and server side input validation and follow secure coding practices such as using parameterize query or calling stored procedures while executing query at database.

In ETM Delhi and BMS system SQL injection issue is occurred below some screen short to show how to generated SQL injection issue .

B.1 SQL injection (AFCS Delhi Web Application)



www.afcsdelhi.com/ETMSPortal/DepotOperations/Infraction/Bus_Infraction.aspx

Most Visited Getting Started Suggested Sites Web Slice Gallery Google


Automatic Fare Collection System


Electronic Ticketing Machines

Home Depot Operations Reports ETM Tracking

You are here: ETM Tracking: ETM Alert Dashboard

Infraction Details

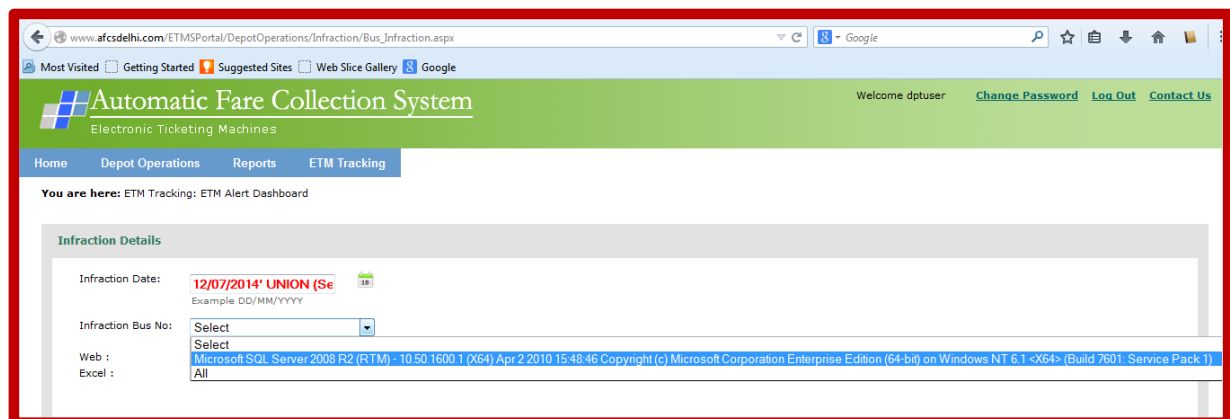
Infraction Date: 12/07/2014 ' UNION (S) 
Example DD/MM/YYYY

Infraction Bus No: Select 

Web : Select
AFCSDb_ST_Portal

Excel : All

Above screenshot shows database current user



www.afcsdelhi.com/ETMSPortal/DepotOperations/Infraction/Bus_Infraction.aspx

Most Visited Getting Started Suggested Sites Web Slice Gallery Google

Automatic Fare Collection System


Electronic Ticketing Machines


Welcome dptuser [Change Password](#) [Log Out](#) [Contact Us](#)

Home Depot Operations Reports ETM Tracking

You are here: ETM Tracking: ETM Alert Dashboard

Infraction Details

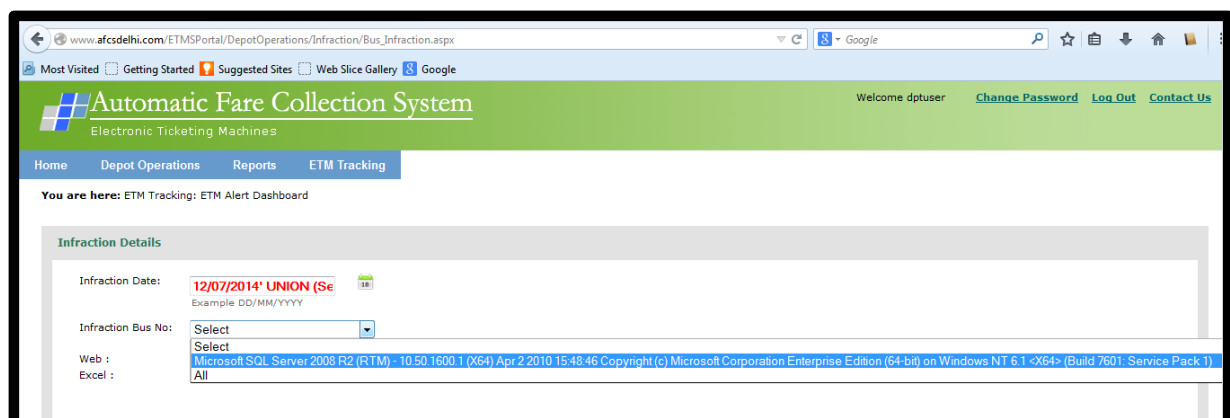
Infraction Date: 12/07/2014' UNION (Se) 
Example DD/MM/YYYY

Infraction Bus No: Select 

Web : Select
Microsoft SQL Server 2008 R2 (RTM) - 10.50.1600.1 (X64) Apr 2 2010 15:48:46 Copyright (c) Microsoft Corporation Enterprise Edition (64-bit) on Windows NT 6.1 <X64> (Build 7601: Service Pack 1)

Excel : All

Above screenshot shows database version



www.afcsdelhi.com/ETMSPortal/DepotOperations/Infraction/Bus_Infraction.aspx

Most Visited Getting Started Suggested Sites Web Slice Gallery Google

Automatic Fare Collection System


Electronic Ticketing Machines


Welcome dptuser [Change Password](#) [Log Out](#) [Contact Us](#)

Home Depot Operations Reports ETM Tracking

You are here: ETM Tracking: ETM Alert Dashboard

Infraction Details

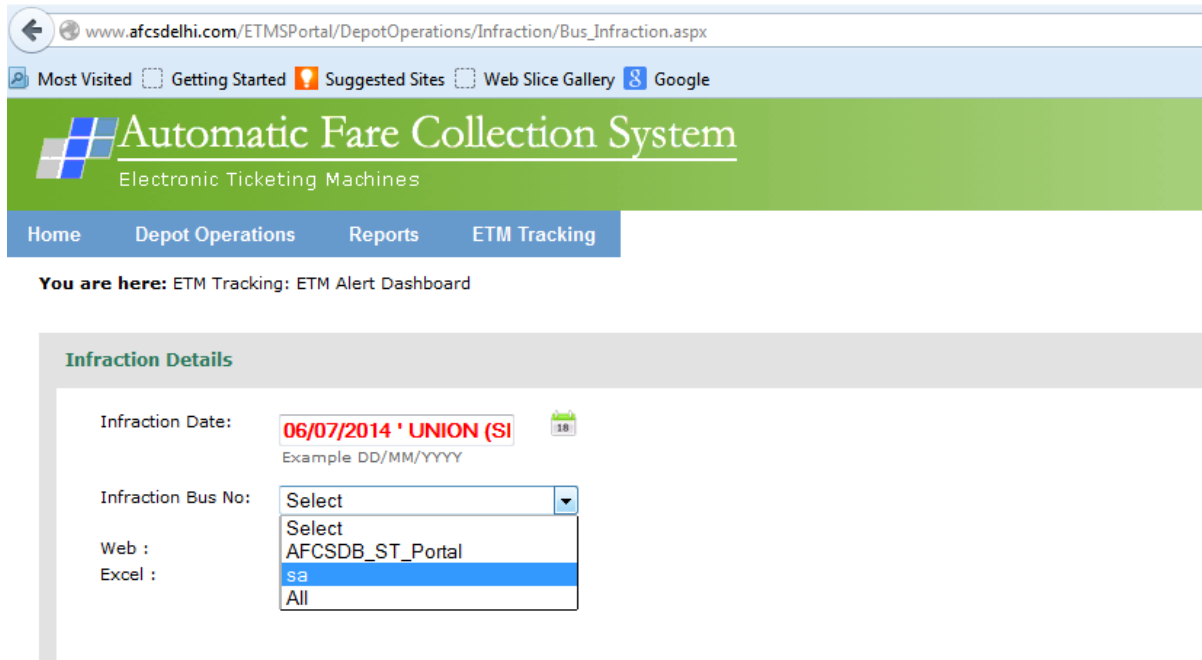
Infraction Date: 12/07/2014' UNION (Se) 
Example DD/MM/YYYY

Infraction Bus No: Select 

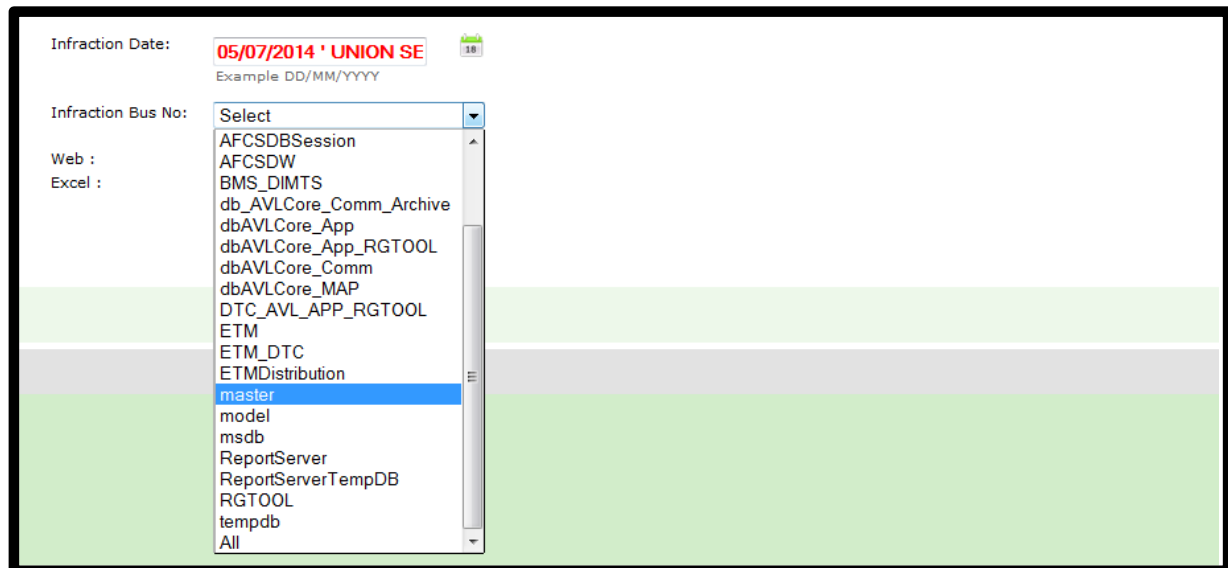
Web : Select
Microsoft SQL Server 2008 R2 (RTM) - 10.50.1600.1 (X64) Apr 2 2010 15:48:46 Copyright (c) Microsoft Corporation Enterprise Edition (64-bit) on Windows NT 6.1 <X64> (Build 7601: Service Pack 1)

Excel : All

Above screenshot shows database version.



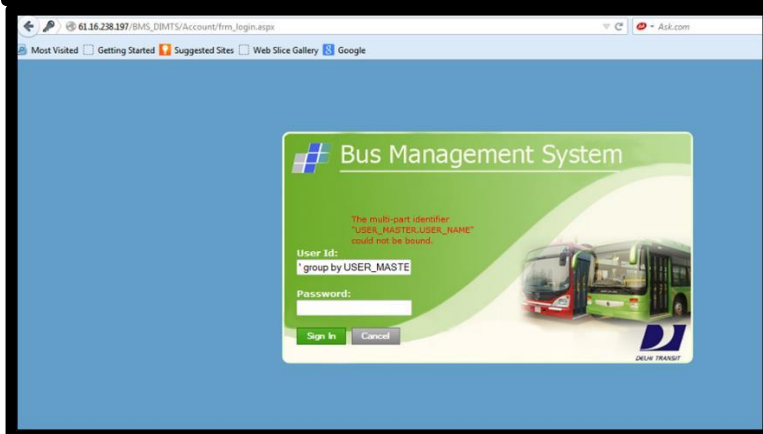
Above screenshot shows database logged in user.



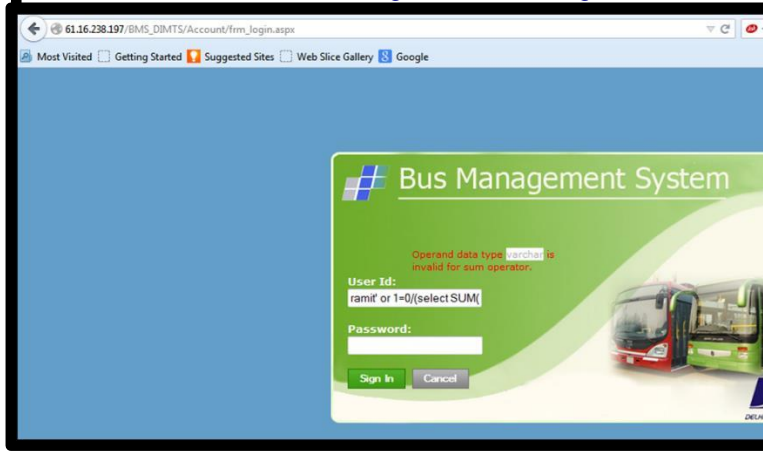
Above screenshot shows list of current database.



Above screenshot shows that User Id field is vulnerable to SQL injection.



Above screenshot shows testing team extracting all column details for USER_MASTER table.



Recommendation

It is recommended to re-write the affected application code to ensure client and server side input validation and follow secure coding practices such as using parameterize query or calling stored procedures while executing query at database.

Solution Code:

1-for date filed

```
to_date(:from_date,'DD/mm/yyyy') and to_date(:to_date,'DD/mm/yyyy')
```

```
cmd.CommandText = QUERY;  
    cmd.Parameters.Add(":from_date", txtFromDate.Text);  
    cmd.Parameters.Add(":to_date", txtToDate.Text);
```

2- for other text box

```
cmd.Connection = conn;  
    cmd.CommandText = QUERY;  
    cmd.Parameters.Add(":card_no", cardnumbertxt.Text);
```

3.1.2 Vertical Privilege Escalation

3.1.2.1 Impacted URL(s)

URL(s)
http://afcsdelhi.com/ETMSPortal/Administration/Users/User.aspx?Mode=Create

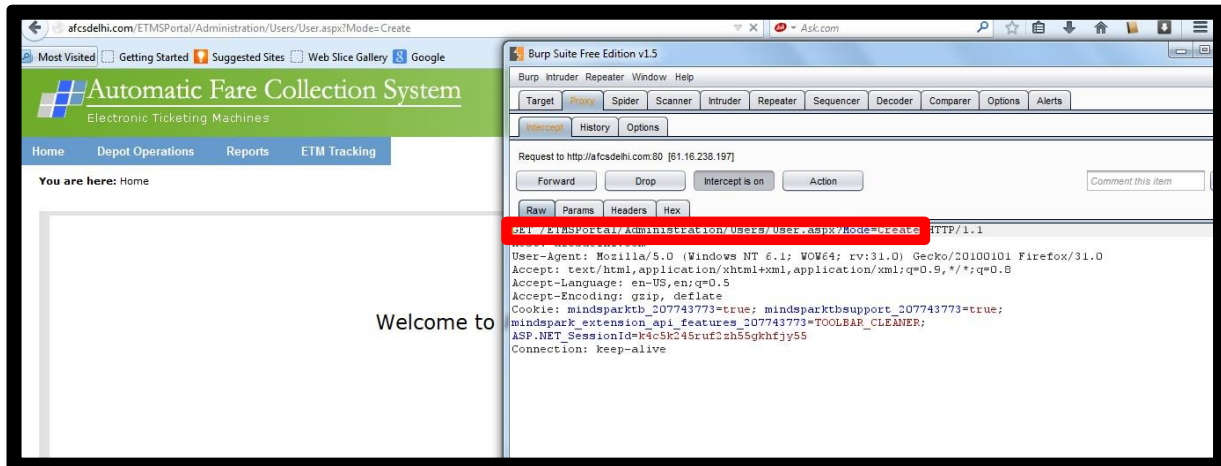
3.1.2.2 Observation

it have been observed that application is susceptible to vertical privilege escalation. It was observed that application segregates the user's privileges based on URL accessibility. That is a user who does not have appropriate privileges will not be able to view a URL. As demonstrated, the review team was able to access the "User Creation "page from a normal account using an unauthorized URL request at;

<http://afcsdelhi.com/ETMSPortal/Administration/Users/User.aspx?Mode=Create>

3.1.2.3 Risk/ Implication

As demonstrated this access control mechanism is not adequate, we observed that a user with less access privileges is able to directly access URLs which require higher privileges through forced browsing (URLs are easily available through content caching and browser history).An attacker may leverage this vulnerability to gain access to higher privilege functionality, using which he can generate unauthorized data such as creating new user.



Above screenshot shows a lower privilege user making a request at user creation

Warning: The username and password to the email ID failed since the email ID does not exist.

Fields marked * are mandatory

First Name: * ramit Company: Delhi Transport

Middle Name: Region: North

Last Name: * gupta Depot: SUB_PAL-Subhash Place

Email ID: * ramitgupta@kpmg.com

Login Details

User Name: * ramit User Role: * Cash User

Activate: * Yes ☒ No ☐

Save Clear Close

Above screenshot shows user has been created.

You are here: Administration: Manage Users

Manage User

Create +

User Name: ramit First Name: Middle Name: Last Name: Company: Delhi Transport Depot: SUB_PAL-Subhash Place Region: North

Status: All User Role: All

Search Cancel

Record: 1-2 out of 2

User Name	Employee Name	User Role	Company
ramit	ramit gupta	DTC	NORTH

Above screenshot confirms user has been created.

Solution Code :

You can check

LoginID, RoleID in SQL / Oracle Query and Bind output with Session ['Roleurl'].

And Below code Paste on Page_Load of Every Page in Application or Master Pages

```
if (Session["ROLEURL"] != null)
{
```

```

        DataSet dsurl = new DataSet();
        dsurl = (DataSet)Session["ROLEURL"];
        if (dsurl.Tables[0].Select("url like '%" + Page.AppRelativeVirtualPath + "%'").Count() <= 0)
        {
            sessionreset();
        }
    }
    else
    {
        sessionreset();
    }
}

```

protected void sessionreset()

```

{
    Session.Abandon();
    Response.Redirect("~/Account/frm_login.aspx");
}

```

Below code avoid the Reference URL and Copy Paste URL

Copy this function on every page load to avoid copy paste URL
string PName = "";

```

        if (Request.UrlReferrer != null)
        {
            PName = Request.UrlReferrer.Segments[Request.UrlReferrer.Segments.Length - 1];
        }

        if (PName == "")
        {
            Session.Abandon();
            Response.Cookies.Add(new HttpCookie("ASP.NET_SessionId", ""));
            Response.Redirect("~/Account/frm_login.aspx");
        }
    }
}

```

3.1.3 Cross Site Request Forgery (CSRF)

3.1.3.1 Impacted URL(s)

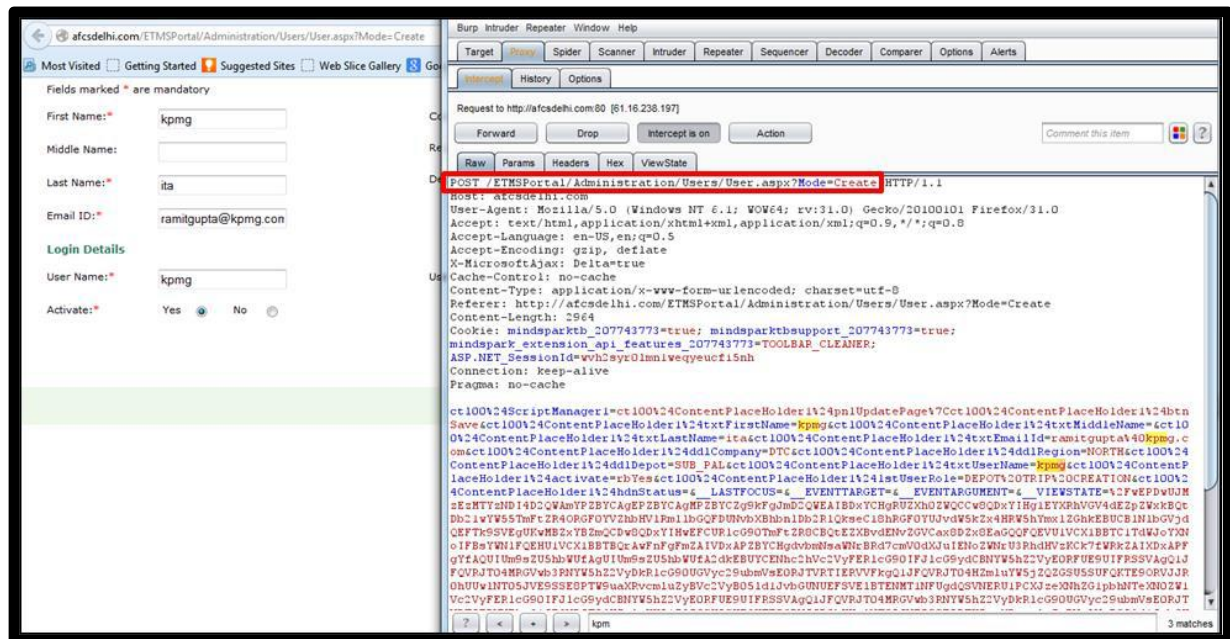
URL(s)
http://afcsdelhi.com/ETMSPortal/Administration/Users/User.aspx?Mode=Create

Risk/ Implication

As demonstrated, it appears that the application is not validating the origin of a request for authenticity. As a result of which it is possible to submit a request on victim's behalf without his / her knowledge. These actions may range from creating new user account. This may lead to compromise of integrity of data within the application.

Recommendation

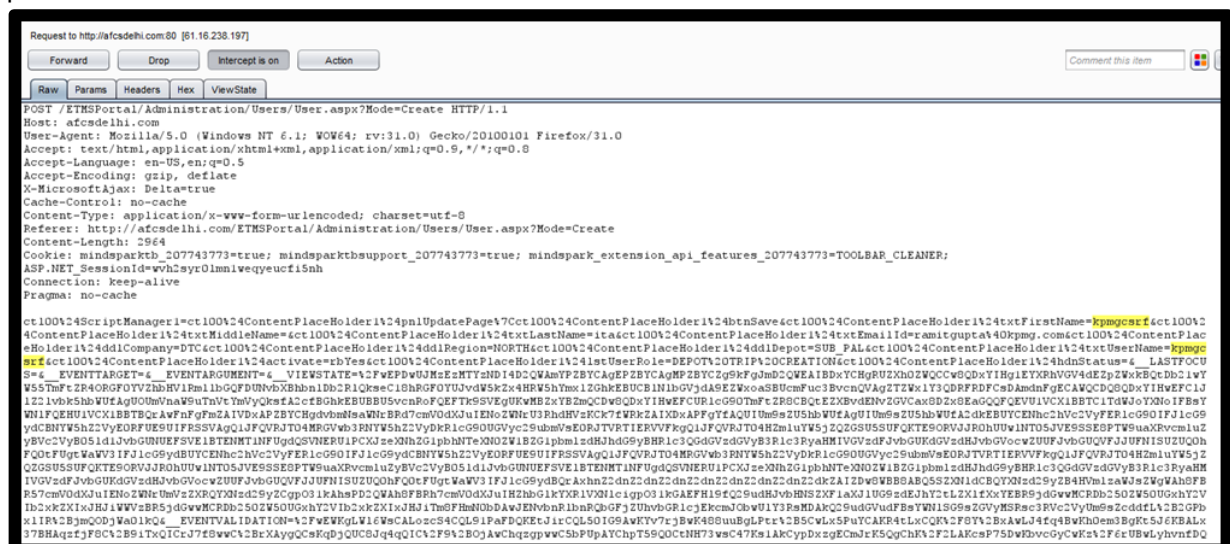
It is recommended to ensure that the application is updated such that it generates random tokens that would be passed with the request and validated at the server. Management may consider implementing CAPTCHA on the page.



Above screenshot shows we making a request to create a user.

```
ctl00%24ScriptManager1=ctl00%24ContentPlaceHolder1%24pnlUpdatePage%7Cctl00%24ContentPlaceHolder1%24btnSave&ctl00%24ContentPlaceHolder1%24txtFirstName=kpmgcsrf&ctl00%24ContentPlaceHolder1%24txtMiddleName=&ctl00%24ContentPlaceHolder1%24txtLastName=ita&ctl00%24ContentPlaceHolder1%24txtEmailId=ramitgupta%40kpmg.com&ctl00%24ContentPlaceHolder1%24ddlCompany=DTC&ctl00%24ContentPlaceHolder1%24ddlRegion=NORTH&ctl00%24ContentPlaceHolder1%24ddlDepot=SUB_PAL&ctl00%24ContentPlaceHolder1%24txtUserName=kpmgcsrf
```

Above screenshot shows we has copied the request and modified the First Name and username parameter.



Above screenshot shows we submitting the copied request again with modified parameter.

Delivering the username and password to the email ID failed since the email ID doesnot exist.

Fields marked * are mandatory

First Name:*	<input type="text" value="kpmgcsrf"/>	Company:*	<input type="text" value="Delhi Transport"/>
Middle Name:	<input type="text"/>	Region:	<input type="text" value="North"/>
Last Name:*	<input type="text" value="ita"/>	Depot:	<input type="text" value="SUB_PAL-Subhash Place"/>
Email ID:*	<input type="text" value="ramitgupta@kpmg.com"/>		

Login Details

User Name:*	<input type="text" value="kpmgcsrf"/>	User Role:*	<input type="text" value="CashUser"/> <input type="text" value="Depot Report Manager"/> <input type="text" value="DEPOT TRIP CREATION"/> <input type="text" value="DepotManager"/>
Activate:*	Yes <input checked="" type="radio"/> No <input type="radio"/>		

Above screenshot shows that the request has been submitted successfully.

Manage User Create

User Name:	<input type="text" value="kpmg"/>	First Name:	<input type="text"/>
User Role:	<input type="text" value="All"/>	Middle Name:	<input type="text"/>
Status:	<input type="text" value="All"/>	Last Name:	<input type="text"/>
Company:	<input type="text" value="Delhi Transport"/>	Depot:	<input type="text" value="SUB_PAL-Subhash Place"/>
Region:	<input type="text" value="North"/>		

Search Cancel

Record:1-2 out of 2

	User Name	Employee Name	User Role	Company
<input type="radio"/>	kpmg	kpmg ita	DTC	NORT
<input type="radio"/>	kpmgcsrf	kpmgcsrf ita	DTC	NORT

Results per page: 10 | 50 | 100

Solution Code:

Use this code on master Page and child master etc.

```
private const string AntiXsrfTokenKey = "__AntiXsrfToken";
private const string AntiXsrfUserNameKey = "__AntiXsrfUserName";
private string _antiXsrfTokenValue;
protected void Page_Init(object sender, EventArgs e)
{
    //First, check for the existence of the Anti-XSS cookie
    var requestCookie = Request.Cookies[AntiXsrfTokenKey];
    Guid requestCookieGuidValue;

    //If the CSRF cookie is found, parse the token from the cookie.
    //Then, set the global page variable and view state user
    //key. The global variable will be used to validate that it matches in the view state form field in the
    Page.PreLoad
    //method.
```



```
if (requestCookie != null && CSRF_TRYPARSE_MAIN.TryParseGuid(requestCookie.Value, out
requestCookieGuidValue)) //Guid.TryParse((( requestCookie.Value, out requestCookieGuidValue))
{
    //Set the global token variable so the cookie value can be
    //validated against the value in the view state form field in
    //the Page.PreLoad method.
    _antiXsrfTokenValue = requestCookie.Value;

    //Set the view state user key, which will be validated by the
    //framework during each request
    Page.ViewStateUserKey = _antiXsrfTokenValue;
}
//If the CSRF cookie is not found, then this is a new session.
else
{
    //Generate a new Anti-XSRF token
    _antiXsrfTokenValue = Guid.NewGuid().ToString("N");

    //Set the view state user key, which will be validated by the
    //framework during each request
    Page.ViewStateUserKey = _antiXsrfTokenValue;

    //Create the non-persistent CSRF cookie
    var responseCookie = new HttpCookie(AntiXsrfTokenKey)
    {
        //Set the HttpOnly property to prevent the cookie from
        //being accessed by client side script
        HttpOnly = true,

        //Add the Anti-XSRF token to the cookie value
        Value = _antiXsrfTokenValue
    };

    //If we are using SSL, the cookie should be set to secure to
    //prevent it from being sent over HTTP connections
    if (FormsAuthentication.RequireSSL &&
Request.IsSecureConnection)
        responseCookie.Secure = true;

    //Add the CSRF cookie to the response
    Response.Cookies.Set(responseCookie);
}

Page.PreLoad += master_Page_PreLoad;
}
protected void master_Page_PreLoad(object sender, EventArgs e)
{
    //During the initial page load, add the Anti-XSRF token and user
    //name to the ViewState
    if (!IsPostBack)
```

```

{
    //Set Anti-XSRF token
    ViewState[AntiXsrfTokenKey] = Page.ViewStateUserKey;

    //If a user name is assigned, set the user name
    ViewState[AntiXsrfUserNameKey] =
    Context.User.Identity.Name ?? String.Empty;
}
//During all subsequent post backs to the page, the token value from
//the cookie should be validated against the token in the view state
//form field. Additionally user name should be compared to the
//authenticated users name
else
{
    //Validate the Anti-XSRF token
    if ((string)ViewState[AntiXsrfTokenKey] != _antiXsrfTokenValue ||
(string)ViewState[AntiXsrfUserNameKey] != (Context.User.Identity.Name ?? String.Empty))
    {
        throw new InvalidOperationException("Validation of Anti-XSRF token failed.");
    }
}
}
}

```

Call this code on master/child page load

```

if (Session["rndNo"] != null && Session["AuthToken"] != null && Request.Cookies["AuthToken"] !=
null)
{
    if (!Session["AuthToken"].ToString().Equals(Request.Cookies["AuthToken"].Value))
    {
        Response.Redirect("~/Login.aspx");
        Session.Abandon();
        Response.Cookies.Add(new HttpCookie("ASP.NET_SessionId", ""));
    }
    else
    {
    }
}
}

```

3.1.4 Default login credentials

3.1.4.1 Impacted URL(s)

URL(s)
http://afcsdelhi.com/ETMSPortal/

Observation

It is observed that AFCS application is being used for creating ETM device conductor Id. It was further observed that newly created account default password is set to his account Id.

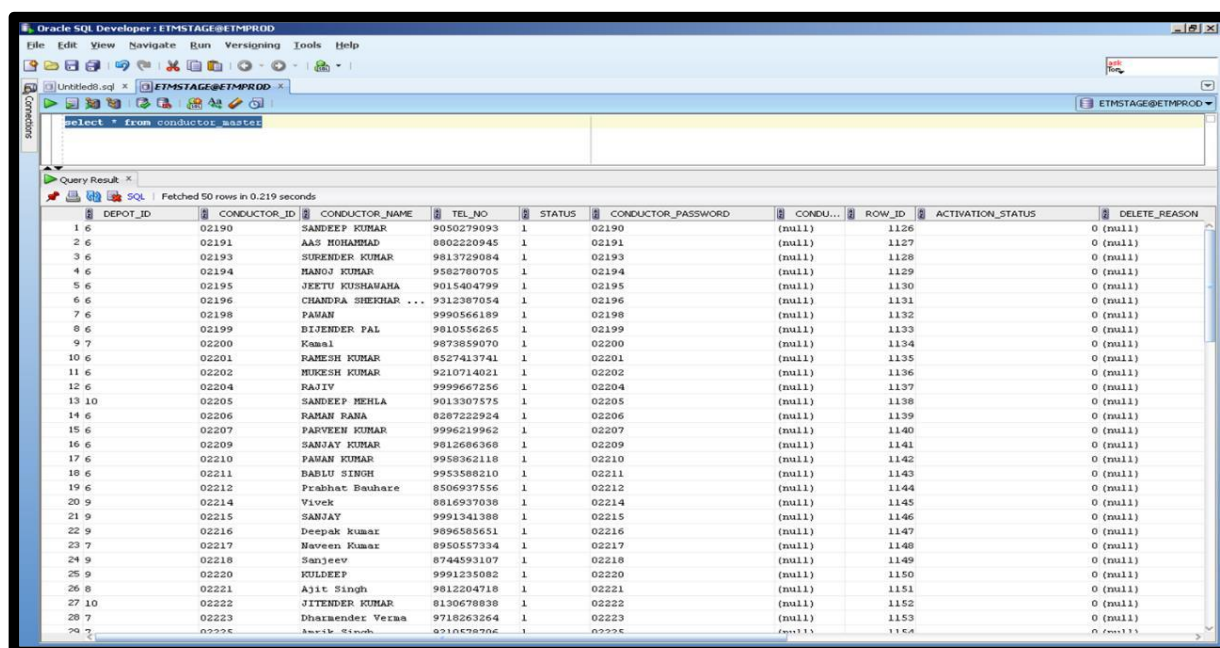
Refer to Appendix B.4 for screenshot(s).

Risk / Implication

In current scenario, a malicious user/conductor may exploit this vulnerability to gain access to other user account on an ETM device. Further, a malicious user may perform unauthorized operations such as performing duty on victim's behalf.

Recommendation

It is recommended to set a password while creating a new user account and follow a secure mechanism to share the password with the user.



DEPOT_ID	CONDUCTOR_ID	CONDUCTOR_NAME	TEL_NO	STATUS	CONDUCTOR_PASSWORD	CONDU...	ROW_ID	ACTIVATION_STATUS	DELETE_REASON
1	02190	SANDEEP KUMAR	9050279093	1	02190	(null)	1126	0 (null)	0 (null)
2	02191	AAS MOHAMMAD	8802220945	1	02191	(null)	1127	0 (null)	0 (null)
3	02193	SURENDER KUMAR	9813729084	1	02193	(null)	1128	0 (null)	0 (null)
4	02194	MANOJ KUMAR	9582780705	1	02194	(null)	1129	0 (null)	0 (null)
5	02195	JEETU KUSHAWAHA	9015404799	1	02195	(null)	1130	0 (null)	0 (null)
6	02196	CHANDRA SHEKHAR ...	9312387054	1	02196	(null)	1131	0 (null)	0 (null)
7	02198	PAWAN	9990566189	1	02198	(null)	1132	0 (null)	0 (null)
8	02199	BIJENDER PAL	9810556265	1	02199	(null)	1133	0 (null)	0 (null)
9	02200	Kamal	9873859070	1	02200	(null)	1134	0 (null)	0 (null)
10	02201	RAMESH KUMAR	8527413741	1	02201	(null)	1135	0 (null)	0 (null)
11	02202	MUKESH KUMAR	9210714021	1	02202	(null)	1136	0 (null)	0 (null)
12	02204	RAJIV	9999667256	1	02204	(null)	1137	0 (null)	0 (null)
13	02205	SANDEEP MEHLA	9013307575	1	02205	(null)	1138	0 (null)	0 (null)
14	02206	RAMAN RAMA	8287222924	1	02206	(null)	1139	0 (null)	0 (null)
15	02207	PARVEEN KUMAR	9996219962	1	02207	(null)	1140	0 (null)	0 (null)
16	02209	SANJAY KUMAR	9812686368	1	02209	(null)	1141	0 (null)	0 (null)
17	02210	PAWAN KUMAR	9958362118	1	02210	(null)	1142	0 (null)	0 (null)
18	02211	BABLU SINGH	9953588210	1	02211	(null)	1143	0 (null)	0 (null)
19	02212	Prabhat Baulhare	8506937536	1	02212	(null)	1144	0 (null)	0 (null)
20	02214	Vivek	8816937038	1	02214	(null)	1145	0 (null)	0 (null)
21	02215	SANJAY	9991341388	1	02215	(null)	1146	0 (null)	0 (null)
22	02216	Deepak kumar	8996585651	1	02216	(null)	1147	0 (null)	0 (null)
23	02217	Naveen Kumar	8950557334	1	02217	(null)	1148	0 (null)	0 (null)
24	02218	Sanjeev	8744593107	1	02218	(null)	1149	0 (null)	0 (null)
25	02220	KULDEEP	9991235082	1	02220	(null)	1150	0 (null)	0 (null)
26	02221	Ajit Singh	9812204718	1	02221	(null)	1151	0 (null)	0 (null)
27	02222	JITENDER KUMAR	8130678838	1	02222	(null)	1152	0 (null)	0 (null)
28	02223	Dharmender Verma	9718263264	1	02223	(null)	1153	0 (null)	0 (null)

3.1.5 Insecure storage of sensitive information

3.1.5.1 Impacted URL(s)

URL(s)

<http://afcsdelhi.com/ETMSPortal/>

Observation

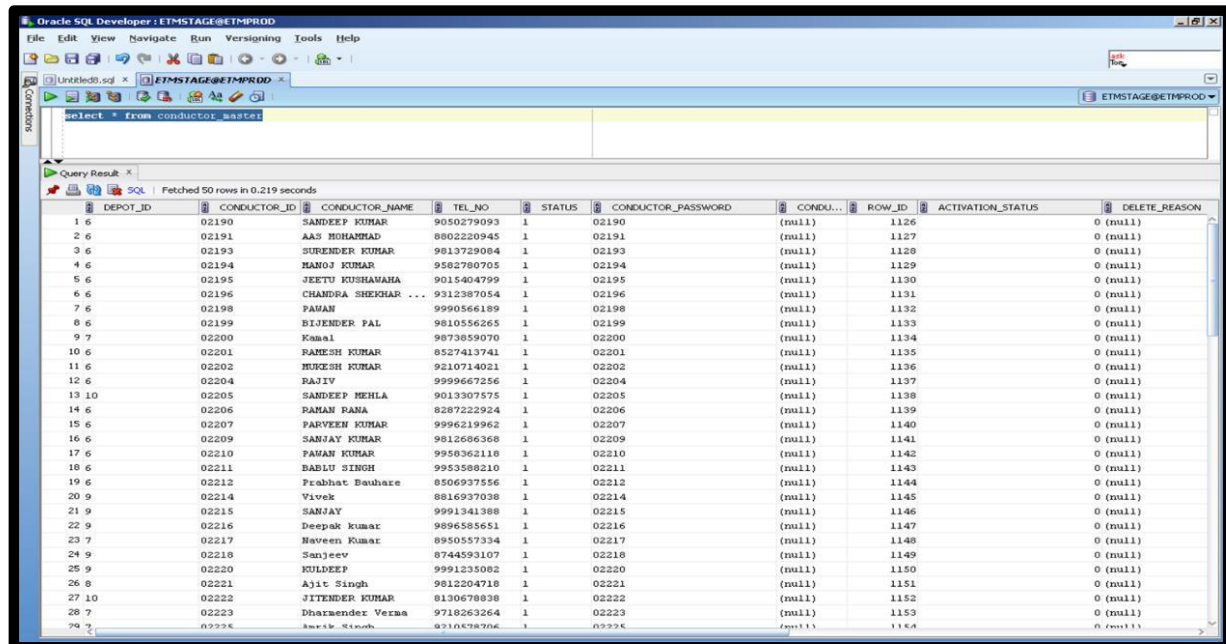
It is observed ETM device user credentials are stored in an unencrypted format in the database. Refer to Appendix B.5 for screenshot(s).

Risk / Implication

Storing sensitive information such as user credentials in clear text is an inherent security risk. In the present scenario, it would be trivial for an attacker to gain access to this information in the event for the database being compromised. Also, a malicious administrator with access to the database can possibly use this information, such as user credentials to gain access to ETM device application.

Recommendation

It is recommended to avoid storing sensitive information in human-readable text or in an easily decoded format, such as Base64 encoding. Instead, information should be encrypted such that, if exposed to an attacker, the attacker cannot easily determine what the sensitive information contains.



Oracle SQL Developer: ETMSTAGE@ETMPROD

Query Result: X

Fetches 50 rows in 0.219 seconds

DEPOT_ID	CONDUCTOR_ID	CONDUCTOR_NAME	TEL_NO	STATUS	CONDUCTOR_PASSWORD	CONDU...	ROW_ID	ACTIVATION_STATUS	DELETE_REASON
1	02190	SANDEEP KUMAR	9050279093	1	02190	(null)	1126	0 (null)	
2	02191	AAS MOHAMMAD	8802220945	1	02191	(null)	1127	0 (null)	
3	02193	SURENDER KUMAR	9813729094	1	02193	(null)	1128	0 (null)	
4	02194	MANOJ KUMAR	9582780705	1	02194	(null)	1129	0 (null)	
5	02195	JEETU KUSHANAJA	9015404799	1	02195	(null)	1130	0 (null)	
6	02196	CHANDRA SHEKHAR ...	9312387054	1	02196	(null)	1131	0 (null)	
7	02198	PAVAN	9990566189	1	02198	(null)	1132	0 (null)	
8	02199	BIJENDER PAL	9810556265	1	02199	(null)	1133	0 (null)	
9	02200	Kamal	9873859070	1	02200	(null)	1134	0 (null)	
10	02201	RAHESH KUMAR	8527413741	1	02201	(null)	1135	0 (null)	
11	02202	MUKESH KUMAR	9210714021	1	02202	(null)	1136	0 (null)	
12	02204	RAJIV	9999667256	1	02204	(null)	1137	0 (null)	
13	02205	SANDEEP MEHLA	9013907575	1	02205	(null)	1138	0 (null)	
14	02206	RAMAN RAMA	8287222924	1	02206	(null)	1139	0 (null)	
15	02207	PARVEEN KUMAR	9996219962	1	02207	(null)	1140	0 (null)	
16	02209	SANJAY KUMAR	9812686368	1	02209	(null)	1141	0 (null)	
17	02210	PAVAN KUMAR	9958362118	1	02210	(null)	1142	0 (null)	
18	02211	BABLU SINGH	9953588210	1	02211	(null)	1143	0 (null)	
19	02212	Prabhat Bawhare	8506937556	1	02212	(null)	1144	0 (null)	
20	02214	Vivek	8816937038	1	02214	(null)	1145	0 (null)	
21	02215	SANJAY	9991341388	1	02215	(null)	1146	0 (null)	
22	02216	Deepak Kumar	9896555551	1	02216	(null)	1147	0 (null)	
23	02217	Naveen Kumar	8950557334	1	02217	(null)	1148	0 (null)	
24	02218	Sanjeev	8744593107	1	02218	(null)	1149	0 (null)	
25	02220	KULDEEP	9991235082	1	02220	(null)	1150	0 (null)	
26	02221	Ajit Singh	9812204718	1	02221	(null)	1151	0 (null)	
27	02222	JITENDER KUMAR	8130678838	1	02222	(null)	1152	0 (null)	
28	02223	Dharmender Verma	9718263264	1	02223	(null)	1153	0 (null)	
29	02224	Ankur K. Sharma	8210578766	1	02224	(null)	1154	0 (null)	

3.1.6 Click jacking attack

Impacted URL(s)

URL(s)

<http://afcsdelhi.com/ETMSPortal/>

Observation

It is observed that application can be opened in an iframe and therefore vulnerable to click jacking based attacks.

Refer to Appendix B.6 for screenshot(s).

Risk/ Implication

A click jacked page tricks a user into performing undesired actions by clicking on a hidden link. On a click jacked page, the attacker shows a set of dummy buttons and then loads another page over it in a transparent layer. The users think that they are clicking the visible buttons, while they are actually performing actions on the hidden page.

Recommendation

It is recommended to implement following "Frame Bursting Code" in Javascript to ensure that page located in domain should be presented at top;

```
if(self.parent.frames.length!=0) self.parent.location = "about:blank"
```

It is also advisable to include X-frame option in response header with "DENY" and "SAMEORIGIN" policy configured. This will ensure that application cannot be opened in an iframe if it does not belong to same origin.

Refer;

- <http://support.microsoft.com/kb/2694329>



Solution Code:

The [X-Frame-Options header](#) can be used to control whether a page can be placed in an IFRAME. Because the Framesniffing technique relies on being able to place the victim site in an IFRAME, a web application can protect itself by sending an appropriate X-Frame-Options header. To configure IIS to add an X-Frame-Options header to all responses for a given site, follow these steps:

1. Open Internet Information Services (IIS) Manager.
2. In the Connections pane on the left side, expand the **Sites** folder and select the site that you want to protect.
3. Double-click the **HTTP Response Headers** icon in the feature list in the middle.
4. In the Actions pane on the right side, click **Add**.
5. In the dialog box that appears, type **X-Frame-Options** in the **Name** field and type **SAMEORIGIN** in the **Value** field.
6. Click **OK** to save your changes.

If you have other sites that need this configuration, repeat steps 2 through 6 for those sites also. This change will prevent HTML pages on other domains from hosting your site in an IFRAME. For example, if the Contoso IT department applies this change to <http://contoso.com>, pages at <http://fabrikam.com> will no longer be able to display content from <http://contoso.com> in an IFRAME. You can modify the value of the X-Frame-Options header to allow <http://fabrikam.com> to frame <http://contoso.com> while blocking all other domains. To do this, change the value of the X-Frame-

Options header in step 5 to **ALLOW-FROM http://fabrikam.com**. For more information about the X-Frame-Options header, see [this MSDN blog post](#). To revert the change, follow these steps:

1. Open Internet Information Services (IIS) Manager.
2. In the Connections pane on the left side, expand the **Sites** folder, and select the site where you made this change.
3. In the feature list in the middle, double-click the **HTTP Response Headers** icon.
4. In the list of headers that appears, select **X-Frame-Options**.
5. Click **Remove** in the Actions pane on the right side.

Or

Use below code in web config

```
<customHeaders>
  <add name="X-Frame-Options" value="SAMEORIGIN" />
</customHeaders>
```

3.1.7 Session replay

3.1.7.1 Impacted URL(s)

URL(s)
http://afcsdelhi.com/ETMSPortal/

Observation

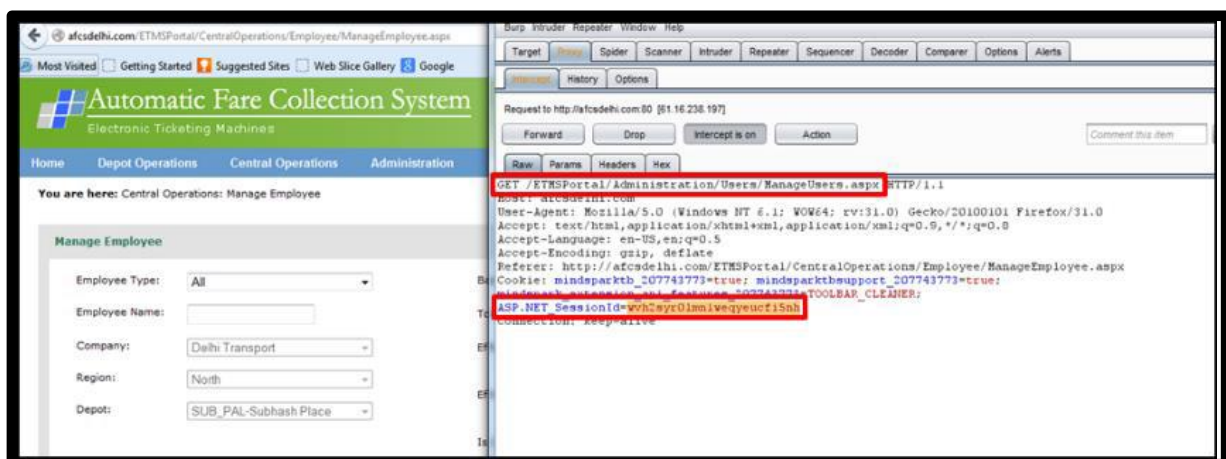
It is observed that an unauthorized user may obtain access to the application with the privileges of another, authorized user by reusing the session ID (.i.e. stored in cookie) of the logged in user.

Risk / Implication

As demonstrated, once an attacker gains the session ID of an authorized user, he may be able to use the same session ID in his web browser and obtain access to the application with the authorized user's privileges. The attacker may obtain the authenticated victim's session ID by launching attacks to exploit XSS.

Recommendation

It is recommended to ensure that the application generates a random token that is bound to session ID of a user. This random token should be passed in the URL or the body of HTML with every request and is validated at web server.

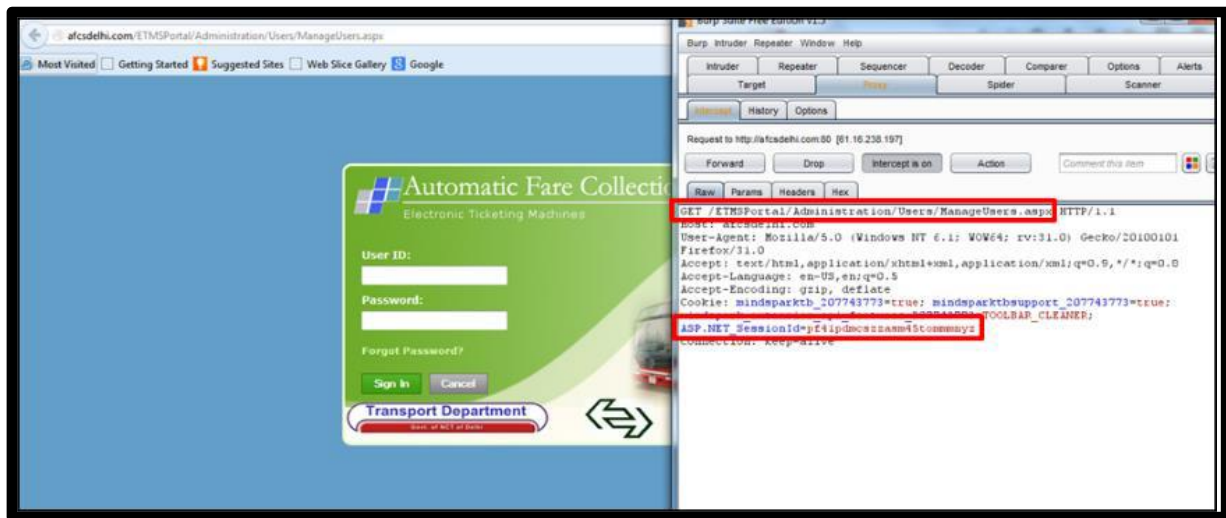


Above screenshot shows an authentic request with valid session id.

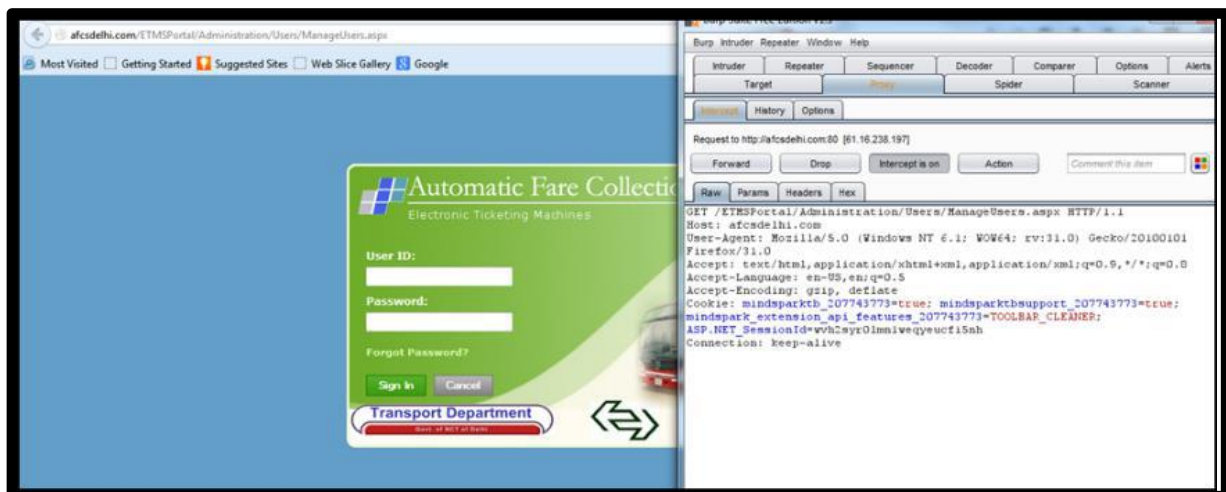
<http://afcsdelhi.com/ETMSPortal/Administration/Users/ManageUsers.aspx>

ASP.NET_SessionId=wvh2syr0lmm1weqyeucfi5nh

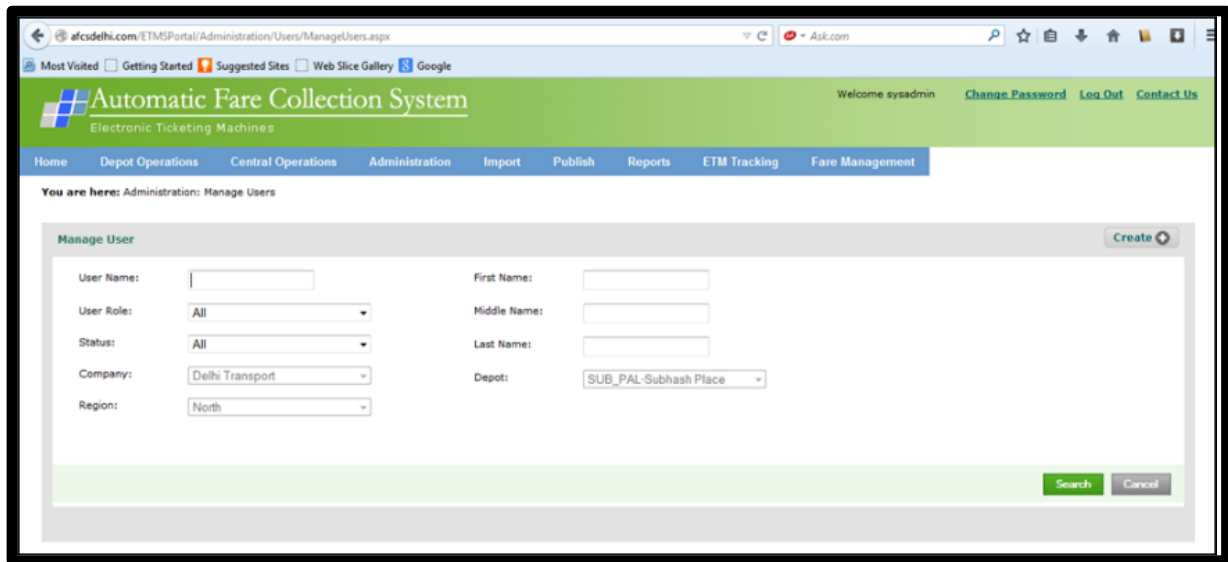
Copied the authentic session id and URL.



Above screenshot shows an unauthorized user making a request to internal page.



Above screenshot shows session id is replaced with active session id.



Above screenshot shows user is inside the application without being authenticated.

Solution Code:

```
string guid = Guid.NewGuid().ToString();
Session["AuthToken"] = guid;
// now create a new cookie with this guid value
Response.Cookies.Add(new HttpCookie("AuthToken", guid));
```

Below code Compare the new generated token and already available token in Cookie should not be same.

```
if (Session["rndNo"] != null && Session["AuthToken"] != null && Request.Cookies["AuthToken"] != null)
{
    if (!Session["AuthToken"].ToString().Equals(Request.Cookies["AuthToken"].Value))
    {
        Session.Abandon();
        Response.Cookies.Add(new HttpCookie("ASP.NET_SessionId", ""));
        Response.Redirect("~/Account/frm_login.aspx");
    }
    else
    {
        //
    }
}
```

3.1.8 Insecure session management

3.1.8.1 Impacted URL(s)

URL(s)
http://afcsdelhi.com/ETMSPortal/

Observation

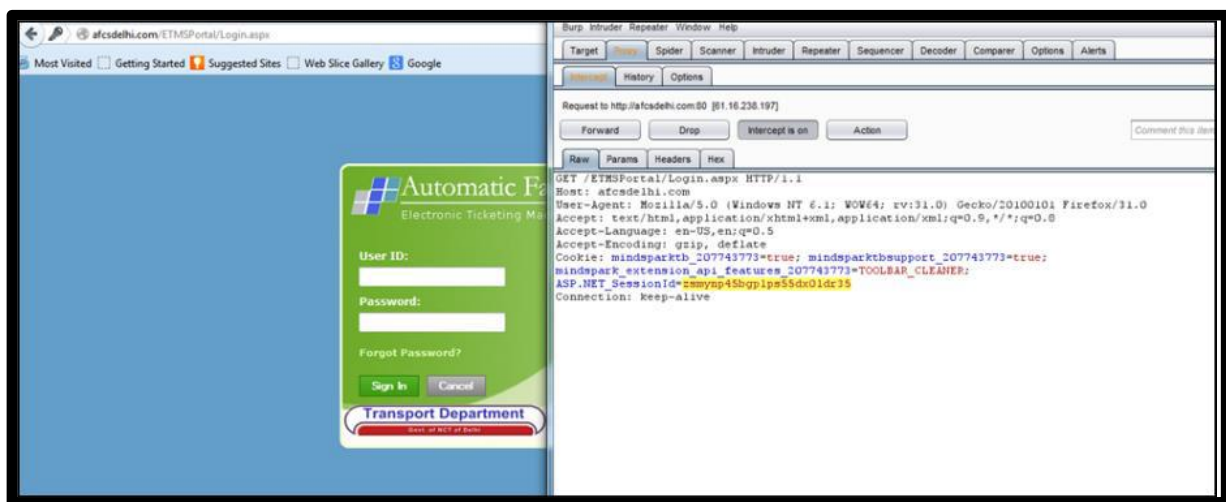
It is observed that application session value remains constant for all pre-login and post-login sessions, till the browser is closed by a user.

Risk/ Implication

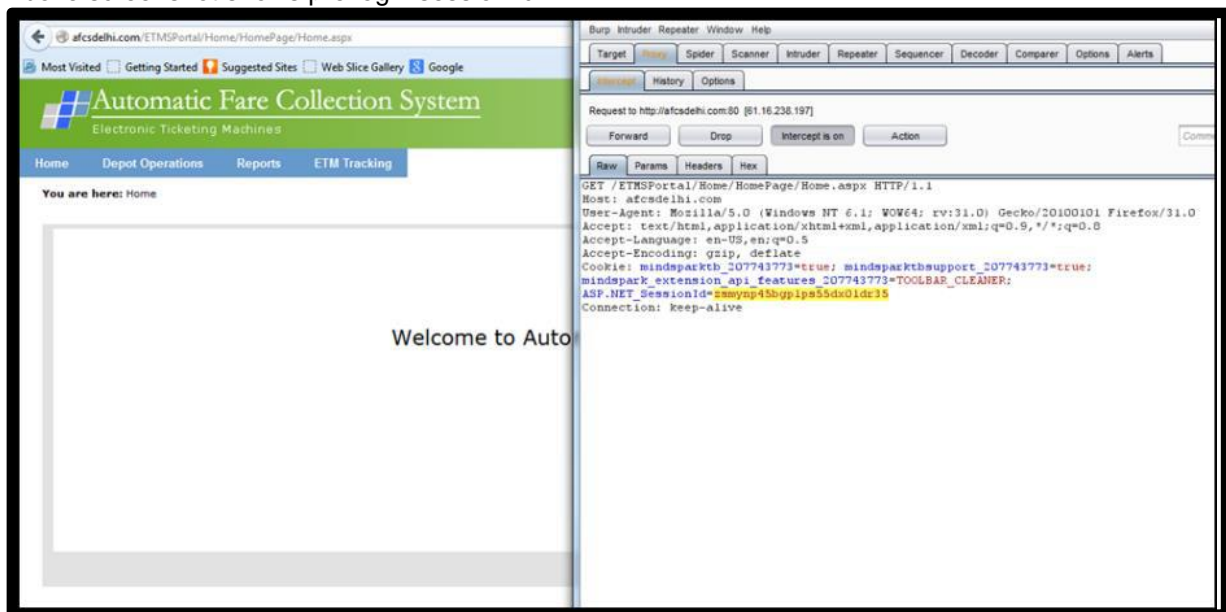
In present scenario, the session ID value remains constant and is present in browser cookie. An attacker/ internal malicious user with adequate access to the system (such as in case of a shared system), may modify the browser cookie value or use a previously saved session ID and perform session replay attack. Thus a legitimate user session may be victimized to an un-authorized access, accountability for which cannot be further determined. Further, this may also lead to session fixation attacks.

Recommendation

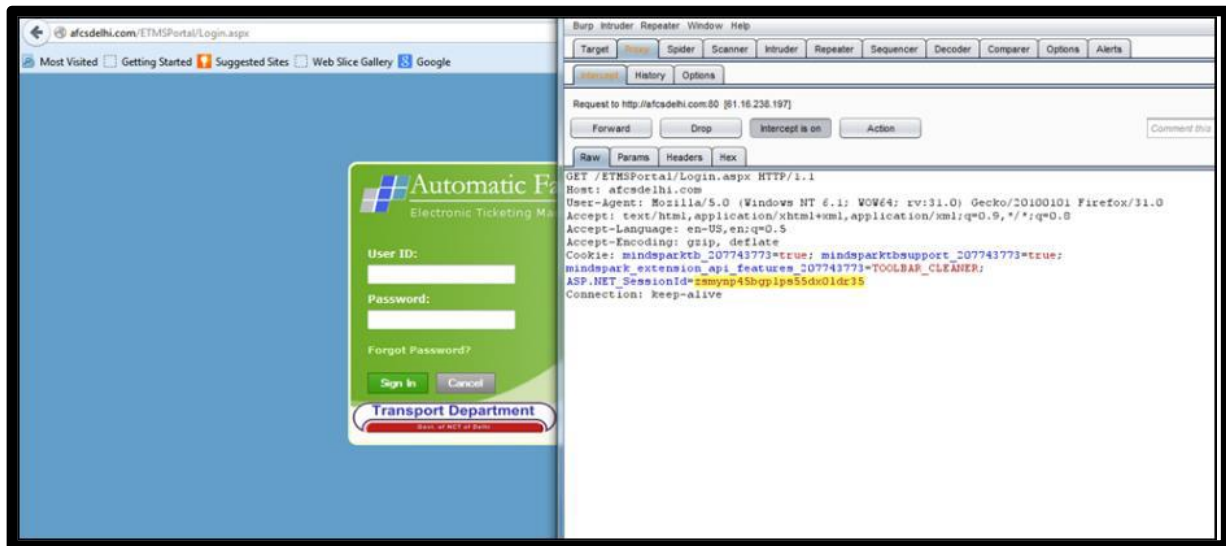
It is recommended to vary the pre-login and post-login session-id value for every unique session.



Above screenshot shows pre login session id.



Above screenshot shows post login session id is the same as the one for pre login.



Above screenshot shows session id remains the same after log out.

```
void regenerateId()
{
    System.Web.SessionState.SessionIDManager manager = new
        System.Web.SessionState.SessionIDManager();
    string oldId = manager.GetSessionID(Context);
    string newId = manager.CreateSessionID(Context);
    bool isAdd = false, isRedir = false;
    manager.SaveSessionID(Context, newId, out isRedir, out isAdd);
    HttpApplication ctx = (HttpApplication)HttpContext.Current.ApplicationInstance;
    HttpModuleCollection mods = ctx.Modules;
    System.Web.SessionState.SessionStateModule ssm =
        (SessionStateModule)mods.Get("Session");
    System.Reflection.FieldInfo[] fields = ssm.GetType().GetFields(BindingFlags.NonPublic |
        BindingFlags.Instance);
    SessionStateStoreProviderBase store = null;
    System.Reflection.FieldInfo rqlIdField = null, rqLockIdField = null, rqStateNotFoundField = null;
    foreach (System.Reflection.FieldInfo field in fields)
    {
        if (field.Name.Equals("_store")) store = (SessionStateStoreProviderBase)field.GetValue(ssm);
        if (field.Name.Equals("_rqlId")) rqlIdField = field;
        if (field.Name.Equals("_rqLockId")) rqLockIdField = field;
        if (field.Name.Equals("_rqSessionStateNotFound")) rqStateNotFoundField = field;
    }
    object lockId = rqLockIdField.GetValue(ssm);
    if ((lockId != null) && (oldId != null)) store.ReleaseItemExclusive(Context, oldId, lockId);
    rqStateNotFoundField.SetValue(ssm, true);
    rqlIdField.SetValue(ssm, newId);
}
```

On Logout we should also regenerate the Session ID

```
protected void InkBtnLogOut_Click(object sender, EventArgs e)
```



```
{
    Session.Abandon();
    Response.Cookies.Add(new HttpCookie("ASP.NET_SessionId", ""));
    Response.Redirect("~/Account/frm_login.aspx");
}
```

3.1.9 Insecure data transmission

3.1.9.1 Impacted URL(s)

URL(s)
http://afcsdelhi.com/ETMSPortal/

Observation

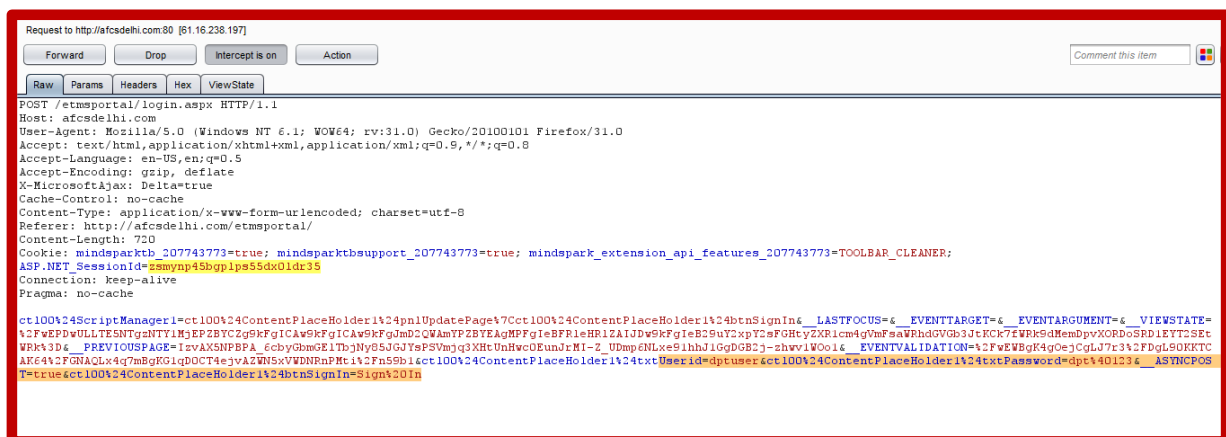
It is observed that user credentials on logon are being transmitted in clear text and no application layer encryption (such as SSL) has been implemented.

Risk/ Implication

In the absence of an encrypted channel, application is vulnerable to interception wherein an attacker, with the ability to intercept traffic between a user and the application, may be able to obtain unauthorized access to sensitive information being communicated.

Recommendation

It is recommended that implementation of a secure encryption channel such as SSL level for sensitive data being communicated.



Solution Code: SSL implement

3.1.10 Password brute force attack possibility

3.1.10.1 Impacted URL(s)

URL(s)
http://afcsdelhi.com/ETMSPortal/

Observation

It is observed that application does not lock a user's account, even after successive five failed attempts with specified user ID at login page. Further it was observed that similar attack is possible over change password field as well.

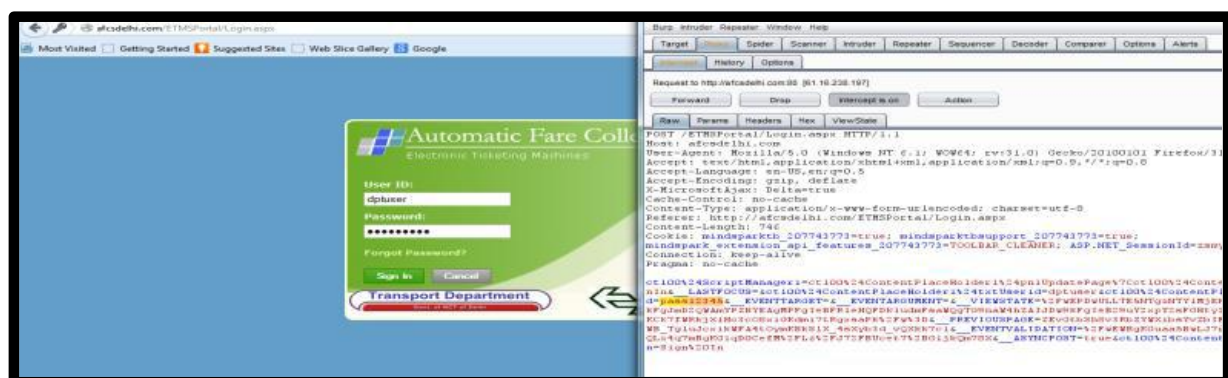
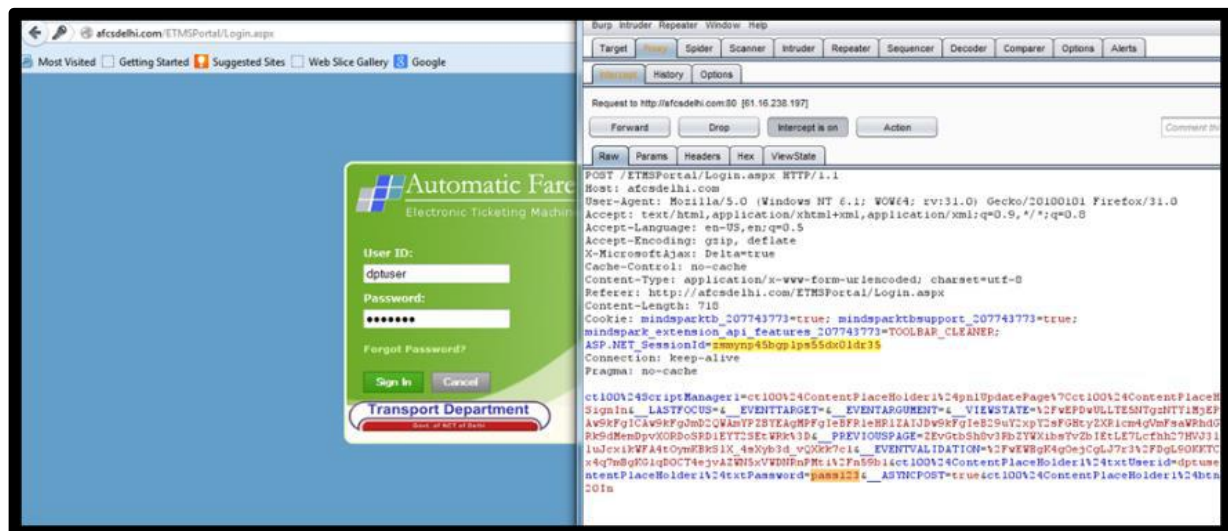
Refer to Appendix B.10 for screenshot(s).

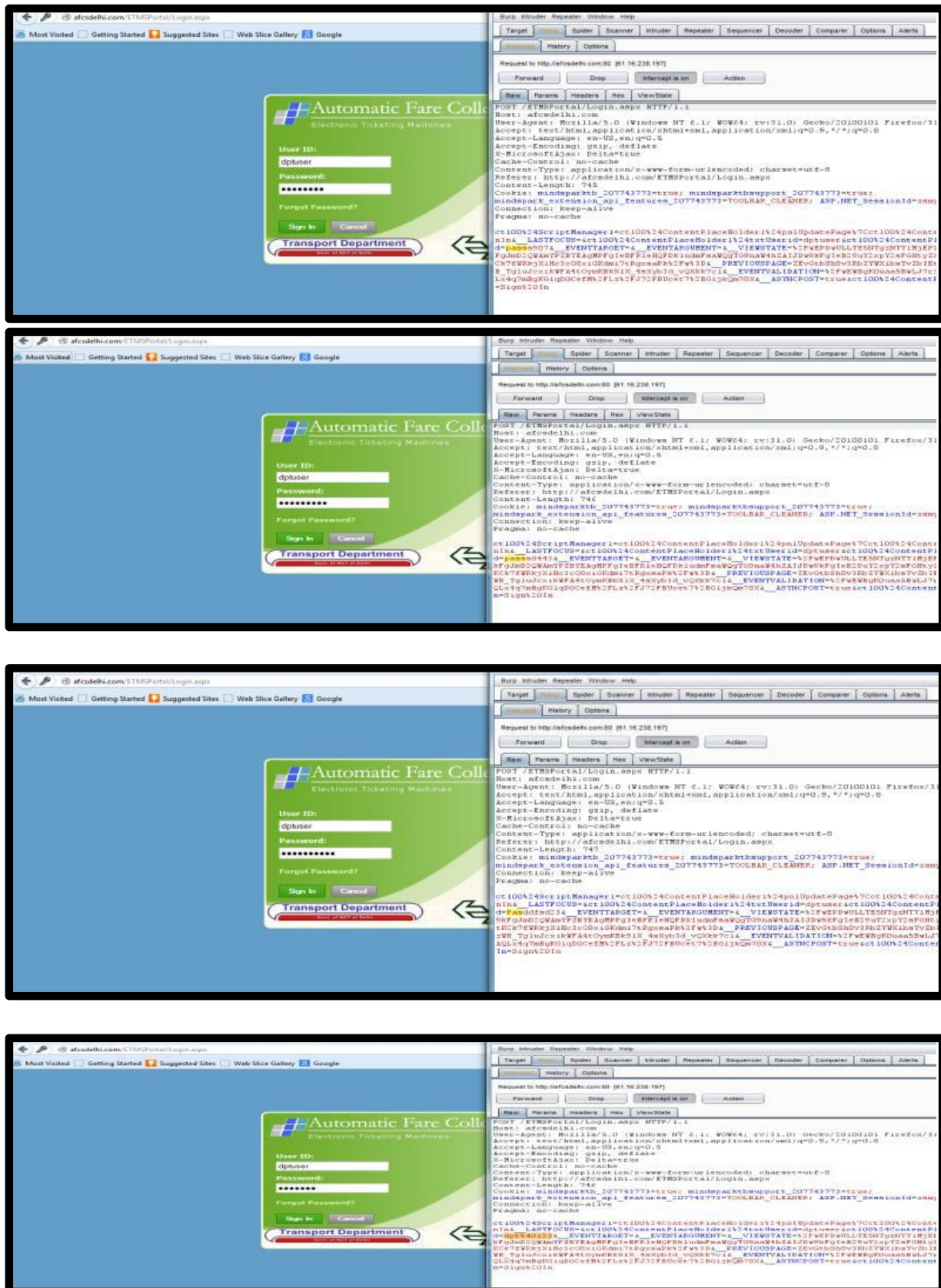
Risk/ Implication

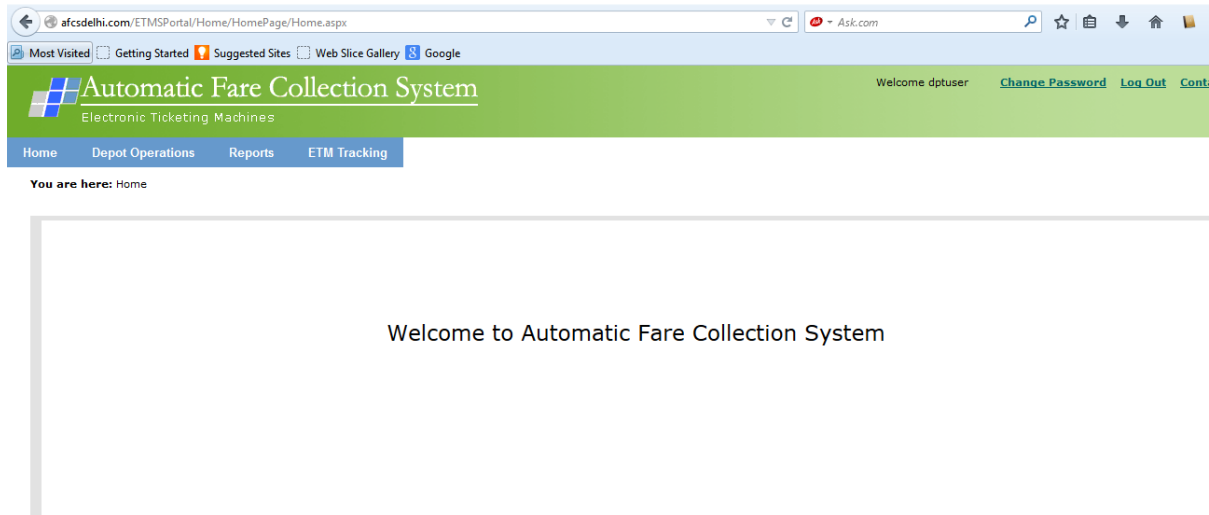
In absence of strong and consistently implemented account lockout policy, an attacker may execute a brute force or dictionary attack in order to enumerate user credentials (i.e. username and password).

Recommendation

It is recommended management should ensure that a strong account lockout policy including locking of user account after 3 successive failed attempts should be implemented. Alternatively management may consider implementing CAPTCHA on login form to avoid dictionary or brute force related attacks.







The series of above screenshots shows obtaining access to the application using brute force attack/multiple iterations of guessing the credentials.

Solution Code:

```
using System;
using System.Drawing;
using System.Drawing.Imaging;
using System.Text;
using System.Drawing.Drawing2D;
public partial class CreateCaptcha : System.Web.UI.Page
{
    private Random rand = new Random();
    protected void Page_Load(object sender, EventArgs e)
    {
        Response.Clear();
        int height = 30;
        int width = 100;
        Bitmap bmp = new Bitmap(width, height);

        RectangleF rectf = new RectangleF(10, 5, 0, 0);

        Graphics g = Graphics.FromImage(bmp);
        g.Clear(Color.White);
        g.SmoothingMode = SmoothingMode.AntiAlias;
        g.InterpolationMode = InterpolationMode.HighQualityBicubic;
        g.PixelOffsetMode = PixelOffsetMode.HighQuality;
        g.DrawString(Session["CaptchaCode"].ToString(), new Font("Thaoma", 12, FontStyle.Italic),
        Brushes.Green, rectf);
        g.DrawRectangle(new Pen(Color.Green), 1, 1, width - 2, height - 2);
        g.Flush();
        Response.ContentType = "image/jpeg";
        bmp.Save(Response.OutputStream, ImageFormat.Jpeg);
        g.Dispose();
        bmp.Dispose();
    }
}
```

```
//if (!IsPostBack)
//{
//  CreateCaptchalImage();
//}

}
/// <summary>
/// method for create captcha image
/// </summary>
private void CreateCaptchalImage()
{
    string code = string.Empty;
    if (Request.QueryString["New"].ToString() == "5")
    {
        code = GetRandomText();
    }
    else
    {
        code = Session["CaptchaCode"].ToString();
    }

    Bitmap bitmap = new Bitmap(170, 40, System.Drawing.Imaging.PixelFormat.Format32bppArgb);
    Graphics g = Graphics.FromImage(bitmap);
    g.SmoothingMode = SmoothingMode.AntiAlias;
    Pen pen = new Pen(Color.Yellow);
    Rectangle rect = new Rectangle(0, 0, 170, 40);

    SolidBrush blue = new SolidBrush(Color.FromArgb(67, 178, 12));
    SolidBrush black = new SolidBrush(Color.Black);

    int counter = 0;

    g.DrawRectangle(pen, rect);
    g.FillRectangle(blue, rect);

    for (int i = 0; i < code.Length; i++)
    {
        g.DrawString(code[i].ToString(), new Font("Tahoma", 5 + rand.Next(10, 15), FontStyle.Italic),
black, new PointF(5 + counter, 5));
        counter += 28;
    }

    // DrawRandomLines(g);
    bitmap.Save(Response.OutputStream, ImageFormat.Gif);

    g.Dispose();
    bitmap.Dispose();

}
/// <summary>
```

```

/// Method for drawing lines
/// </summary>
/// <param name="g"></param>
private void DrawRandomLines(Graphics g)
{
    SolidBrush yellow = new SolidBrush(Color.Yellow);
    for (int i = 0; i < 20; i++)
    {
        g.DrawLines(new Pen(yellow, 1), GetRandomPoints());
    }
}

/// <summary>
/// method for getting random point position
/// </summary>
/// <returns></returns>
private Point[] GetRandomPoints()
{
    Point[] points = { new Point(rand.Next(0, 150), rand.Next(1, 150)), new Point(rand.Next(0, 200),
rand.Next(1, 190)) };
    return points;
}

/// <summary>
/// Method for generating random text of 5 characters as captcha code
/// </summary>
/// <returns></returns>
private string GetRandomText()
{
    StringBuilder randomText = new StringBuilder();
    string alphabets =
"012345679ABCDEFGHIJKLMNPOQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
    Random r = new Random();
    for (int j = 0; j <= 5; j++)
    {
        randomText.Append(alphabets[r.Next(alphabets.Length)]);
    }
    Session["CaptchaCode"] = randomText.ToString();
    return Session["CaptchaCode"] as String;
}
}

```

3.1.11 View State not encrypted

3.1.11.1 Impacted URL(s)

URL(s)
http://afcsdelhi.com/ETMSPortal/

Observation

It is observed that application does not use encryption mechanism for ViewState.

Risk / Implication

An attacker can study the application's state management logic for possible vulnerabilities or read sensitive information from viewstate, if application stores application-critical information in the ViewState.

Recommendation

It is recommended to use encryption mechanism in ViewState using following mechanism:

- Place the following directive at the top of affected page;

```
<%@Page ViewStateEncryptionMode="Always" %>
```

- Apply the following configuration for your application's web.config file.

```
<System.Web>
<pages viewStateEncryptionMode="Always">
</System.Web>
```

```
<pages>
  <controls>
    <add tagPrefix="asp" namespace="System.Web.UI" assembly="System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
    <add tagPrefix="asp" namespace="System.Web.UI.WebControls" assembly="System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
  </controls>
</pages>
```

Solution code:

Use on web config file.

```
<pages enableViewState="true" enableViewStateMac="true">
```

3.1.12 Insecure value of path attribute in Cookie

3.1.12.1 Impacted URL(s)

URL(s)
http://afcsdelhi.com/ETMSPortal/

Observation

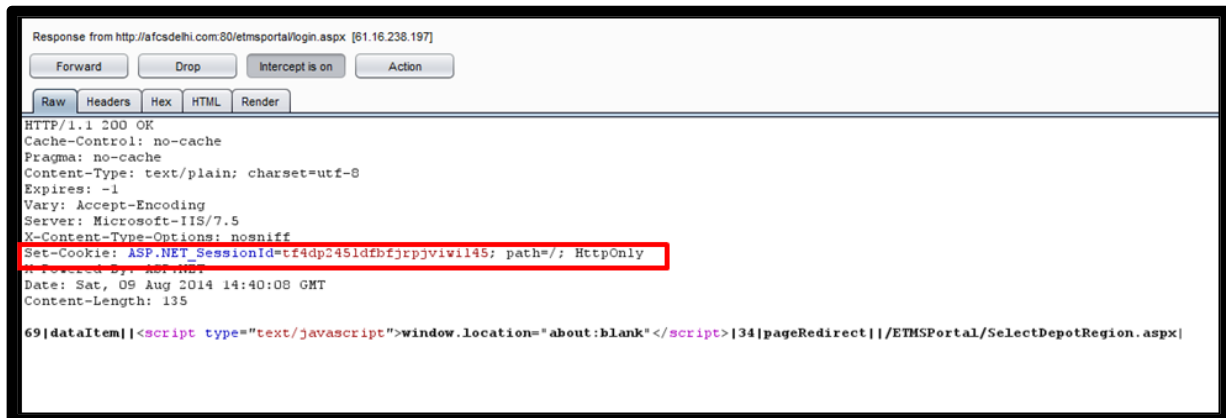
It is observed that the path attribute for the cookie was set to the root directory "/". Refer to Appendix B.12 for screenshot(s).

Risk/ Implication

The path attribute set to root allows any application hosted on the server to access the cookie and pass it on to users with requests for such applications. In case any of such applications are vulnerable, an attacker may be able to obtain such cookies and compromise the user session.

Recommendation

It is recommended to set the value of the "path" attribute to the actual virtual directory path of the application or a more restrictive alternative path as applicable.



Code Solution:

```

HttpCookie myHttpOnlyCookie = new HttpCookie("AppCookie", DateTime.Now.ToString());
myHttpOnlyCookie.HttpOnly = true;
/*Set the Cookies Path and Expiring Time*/
myHttpOnlyCookie.Path = "/App path";
myHttpOnlyCookie.Expires = DateTime.Now.AddMinutes(15);
Response.Cookies.Add(myHttpOnlyCookie);
/*Set secure cookies*/
Response.Cookies.Add(new HttpCookie("SecureCookie")
{
    Secure = true
});

```

Use Below code in the Global.asax Page

```

public void setCookieApp()
{
    if (HttpContext.Current != null)
    {
        //Session.Clear();
        /// only apply session cookie persistence to requests requiring session information
        if (HttpContext.Current.Handler is IRequiresSessionState || HttpContext.Current.Handler is
IReadOnlySessionState)
        {
            System.Web.Configuration.SessionStateSection sessionState =
(System.Web.Configuration.SessionStateSection)ConfigurationManager.GetSection("system.web/ses
sionState");
            string cookieName = sessionState != null &&
!string.IsNullOrEmpty(sessionState.CookieName) ? sessionState.CookieName :
Session["ASP.NET_SessionId"].ToString();
            string cookieName1 = sessionState != null &&
!string.IsNullOrEmpty(sessionState.CookieName) ? sessionState.CookieName : ".ASPXAUTH";

```



```

        // string cookieName2 = sessionState != null &&
!string.IsNullOrEmpty(sessionState.CookieName) ? sessionState.CookieName : "__AntiXsrfToken";
        //System.TimeSpan timeout = sessionState != null ? sessionState.Timeout :
TimeSpan.FromMinutes(20);
        /// Ensure ASP.NET Session Cookies are accessible throughout the subdomains.
        if (HttpContext.Current.Request.Cookies[cookieName] != null)
        {
            //Response.Cookies[cookieName].Value = Session.SessionID;
            HttpContext.Current.Response.Cookies[cookieName].Path =
HttpContext.Current.Request.ApplicationPath;
            //Response.Cookies[cookieName].Expires = DateTime.Now.Add(timeout);
        }
        if (HttpContext.Current.Request.Cookies[cookieName1] != null)
        {
            //Response.Cookies[cookieName].Value = Session.SessionID;
            HttpContext.Current.Response.Cookies[cookieName1].Path =
HttpContext.Current.Request.ApplicationPath;
            //Response.Cookies[cookieName].Expires = DateTime.Now.Add(timeout);
        }
    }

    foreach (string k in HttpContext.Current.Request.Cookies.AllKeys)
    {
        if (k.ToString().Contains("ASP.NET_SessionId"))
        {
            HttpContext.Current.Response.Cookies["ASP.NET_SessionId"].Path =
HttpContext.Current.Request.ApplicationPath;

        }
        else if (k.ToString().Contains(".ASPXAUTH"))
        {
            HttpContext.Current.Response.Cookies[".ASPXAUTH"].Path =
HttpContext.Current.Request.ApplicationPath;
        }
    }
}
else
{
}
}
}

```

3.1.13 Session cookie set without Secure flag

3.1.13.1 Impacted URL(s)

URL(s)
http://afcsdelhi.com/ETMSPortal/

Observation

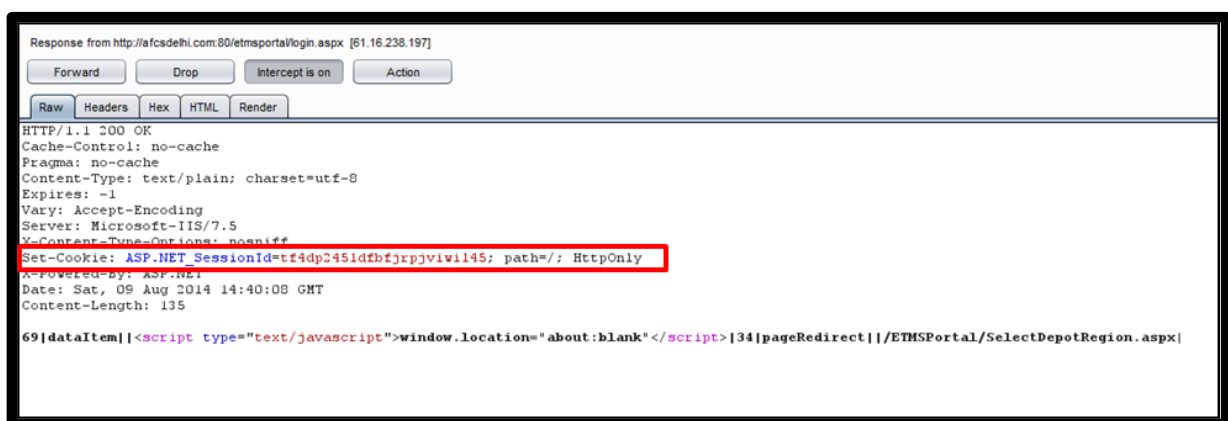
It is observed that the application session cookies do not have the Secure flag set. Refer to Appendix B.13 for screenshot(s).

Risk/ Implication

The purpose of the Secure flag is to prevent cookies from being observed by unauthorized parties due to the transmission of the cookies in clear text. In the current scenario, the browser may transmit cookies over an unencrypted channel thereby allowing an attacker with the ability to intercept traffic to hijack an authenticated user's session and access the application using the same.

Recommendation

It is recommended to rewrite the affected application code ensuring that the Secure flag is set on all cookies set by the application.



Code Solution:

secure encryption channel such as SSL is required

```
<system.web>
  <httpCookies httpOnlyCookies="true" lockItem="true" requireSSL="true" />
</system.web>
```

3.1.14 Information disclosure through error messages

3.1.14.1 Impacted URL(s)

URL(s)
http://afcsdelhi.com/ETMSPortal/reports

Observation

It is observed that sensitive information such as stack traces of web server was disclosed through error messages or application responses. Refer to Appendix B.14 for screenshot(s).

Risk/ Implication

Pages containing error/ warning messages that may disclose sensitive information that are produced through the unhandled exception are an inherent risk to the security of the application. This information can be used to craft further targeted attacks against the application.

Recommendation

It is recommended to ensure that application and the web server configurations are reviewed and changed to restrict error messages from displaying detailed information about the server or the application hosted on it.

It is also advisable to devising an application-wide error reporting mechanism. Such a mechanism should ensure that errors are reported in a consistent, simple and user-friendly language while avoiding disclosure of any sensitive technical details.



Solution Code:

```

        int operation;

        try
        {
            operation = Convert.ToInt32(Request.QueryString["a"]);
        }
        catch (Exception ex)
        {

            lblBreadCrumbrv.Text = "Invalid URL Please try again ";
            return;

        }

        hdnQueryString.Value = operation.ToString();
    
```

3.1.15 Insecure configuration of session timeout

3.1.15.1 Impacted URL(s)

URL(s)
http://afcsdelhi.com/ETMSPortal/ reports

Observation

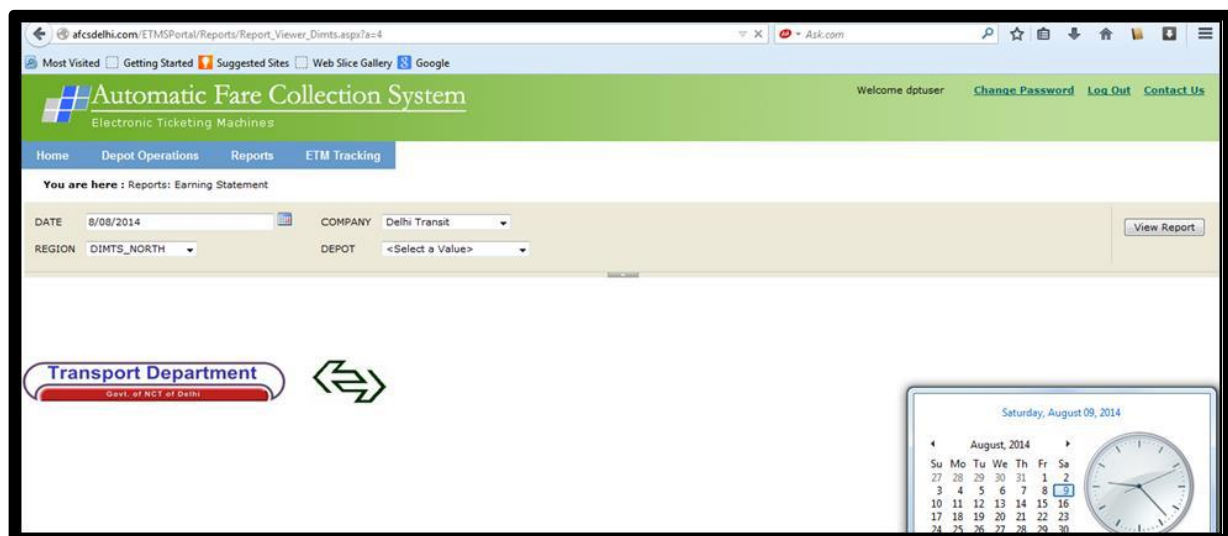
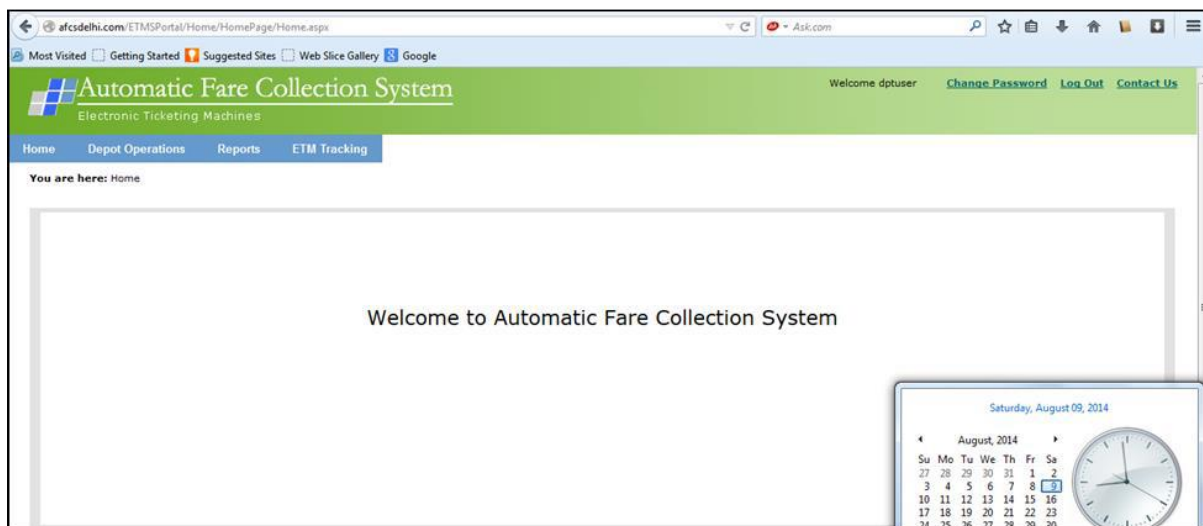
It is observed that when a legitimate authenticated user does not perform any activity on the application for a long period of time (greater than 20 minutes), the session remains valid. Refer to Appendix B.15 for screenshot(s).

Risk/ Implication

Absence of session timeout due to inactivity increases the time frame under which an attacker may hijack an unmanned session and gain privileges of the authenticated user.

Recommendation

It is recommended to ensure that a session time-out in the application is implemented as per the organization's information security policies. If the application remains idle for a prolonged time period, for example 10 minutes, the session should automatically expire.



Solution Code:

```
<sessionState regenerateExpiredSessionId="true" timeout="10"/>
```

4 Reference(s)

- KPMG VAPT Report
- https://www.owasp.org/index.php/Main_Page

5 Annexure(s)

- NA