UPM 2 Factor authentication Scan Report

Project Name	UPM 2 Factor authentication
Scan Start	Tuesday, December 31, 2019 10:59:32 AM
Preset	Checkmarx Default
Scan Time	00h:01m:35s
Lines Of Code Scanned	3310
Files Scanned	33
Report Creation Time	Tuesday, December 31, 2019 11:06:52 AM
Online Results	http://ACTIMIZE-IFM-DE/CxWebClient/ViewerMain.aspx?scanid=1000251&projectid=232
Team	DEVELOPMENT
Checkmarx Version	8.5.0
Scan Type	Full
Source Origin	LocalPath
Density	4/1000 (Vulnerabilities/LOC)
Visibility	Public

Filter Settings

Severity

Included: High, Medium, Low, Information

Excluded: None

Result State

Included: Confirmed, Not Exploitable, To Verify, Urgent, Proposed Not Exploitable

Excluded: None

Assigned to

Included: All

Categories

Included:

Uncategorized	All
Custom	All
PCI DSS v3.2	All
OWASP Top 10 2013	All
FISMA 2014	All
NIST SP 800-53	All

Excluded:

Uncategorized	None
Custom	None
PCI DSS v3.2	None
OWASP Top 10 2013	None
FISMA 2014	None
NIST SP 800-53	None

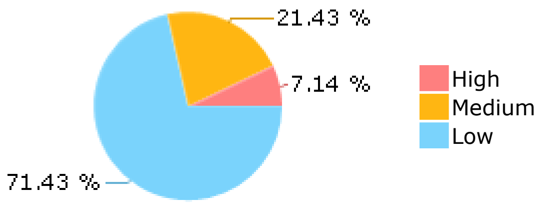
Results Limit

Results limit per query was set to 50

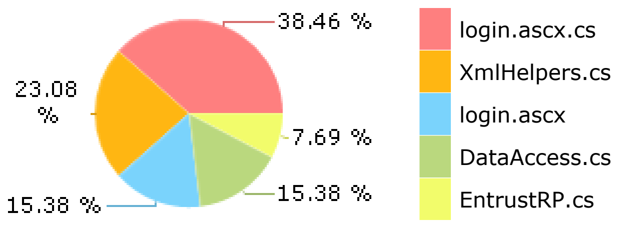
Selected Queries

Selected queries are listed in Result Summary

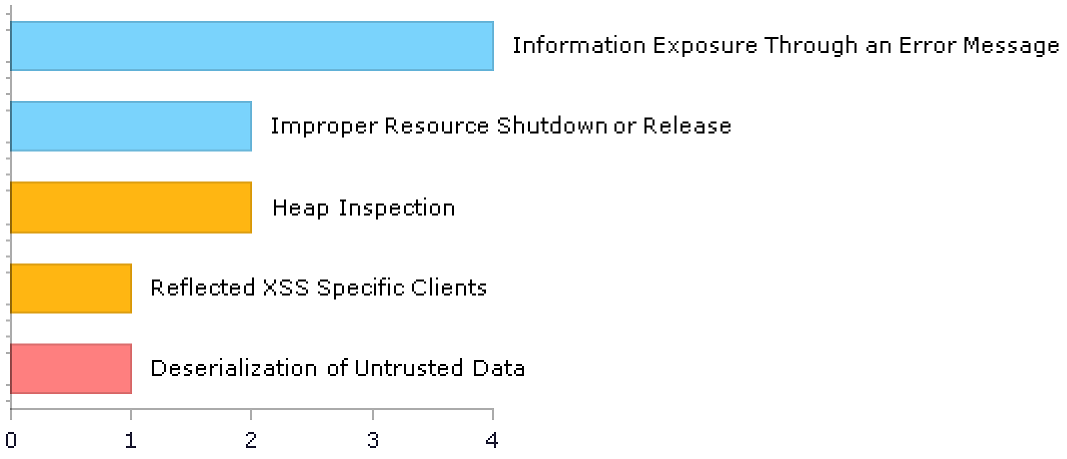
Result Summary



Most Vulnerable Files



Top 5 Vulnerabilities



Scan Summary - OWASP Top 10 2013

Further details and elaboration about vulnerabilities and risks can be found at: OWASP Top 10 2013

Category	Threat Agent	Attack Vectors	Weakness Prevalence	Weakness Detectability	Technical Impact	Business Impact	Issues Found	Best Fix Locations
A1-Injection*	EXTERNAL, INTERNAL, ADMIN USERS	EASY	COMMON	AVERAGE	SEVERE	ALL DATA	0	0
A2-Broken Authentication and Session Management*	EXTERNAL, INTERNAL USERS	AVERAGE	WIDESPREAD	AVERAGE	SEVERE	AFFECTED DATA AND FUNCTIONS	0	0
A3-Cross-Site Scripting (XSS)*	EXTERNAL, INTERNAL, ADMIN USERS	AVERAGE	VERY WIDESPREAD	EASY	MODERATE	AFFECTED DATA AND SYSTEM	1	1
A4-Insecure Direct Object References*	SYSTEM USERS	EASY	COMMON	EASY	MODERATE	EXPOSED DATA	0	0
A5-Security Misconfiguration	EXTERNAL, INTERNAL, ADMIN USERS	EASY	COMMON	EASY	MODERATE	ALL DATA AND SYSTEM	0	0
A6-Sensitive Data Exposure*	EXTERNAL, INTERNAL, ADMIN USERS, USERS BROWSERS	DIFFICULT	UNCOMMON	AVERAGE	SEVERE	EXPOSED DATA	5	4
A7-Missing Function Level Access Control*	EXTERNAL, INTERNAL USERS	EASY	COMMON	AVERAGE	MODERATE	EXPOSED DATA AND FUNCTIONS	1	1
A8-Cross-Site Request Forgery (CSRF)*	USERS BROWSERS	AVERAGE	COMMON	EASY	MODERATE	AFFECTED DATA AND FUNCTIONS	0	0
A9-Using Components with Known Vulnerabilities	EXTERNAL USERS, AUTOMATED TOOLS	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	AFFECTED DATA AND FUNCTIONS	0	0
A10-Unvalidated Redirects and Forwards	USERS BROWSERS	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	AFFECTED DATA AND FUNCTIONS	0	0

* Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - PCI DSS v3.2

Category	Issues Found	Best Fix Locations
PCI DSS (3.2) - 6.5.1 - Injection flaws - particularly SQL injection*	0	0
PCI DSS (3.2) - 6.5.2 - Buffer overflows	0	0
PCI DSS (3.2) - 6.5.3 - Insecure cryptographic storage*	0	0
PCI DSS (3.2) - 6.5.4 - Insecure communications*	0	0
PCI DSS (3.2) - 6.5.5 - Improper error handling*	6	4
PCI DSS (3.2) - 6.5.7 - Cross-site scripting (XSS)	1	1
PCI DSS (3.2) - 6.5.8 - Improper access control*	2	2
PCI DSS (3.2) - 6.5.9 - Cross-site request forgery*	0	0
PCI DSS (3.2) - 6.5.10 - Broken authentication and session management*	0	0

* Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - FISMA 2014

Category	Description	Issues Found	Best Fix Locations
Access Control	Organizations must limit information system access to authorized users, processes acting on behalf of authorized users, or devices (including other information systems) and to the types of transactions and functions that authorized users are permitted to exercise.	0	0
Audit And Accountability*	Organizations must: (i) create, protect, and retain information system audit records to the extent needed to enable the monitoring, analysis, investigation, and reporting of unlawful, unauthorized, or inappropriate information system activity; and (ii) ensure that the actions of individual information system users can be uniquely traced to those users so they can be held accountable for their actions.	0	0
Configuration Management*	Organizations must: (i) establish and maintain baseline configurations and inventories of organizational information systems (including hardware, software, firmware, and documentation) throughout the respective system development life cycles; and (ii) establish and enforce security configuration settings for information technology products employed in organizational information systems.	5	4
Identification And Authentication*	Organizations must identify information system users, processes acting on behalf of users, or devices and authenticate (or verify) the identities of those users, processes, or devices, as a prerequisite to allowing access to organizational information systems.	1	1
Media Protection	Organizations must: (i) protect information system media, both paper and digital; (ii) limit access to information on information system media to authorized users; and (iii) sanitize or destroy information system media before disposal or release for reuse.	2	2
System And Communications Protection	Organizations must: (i) monitor, control, and protect organizational communications (i.e., information transmitted or received by organizational information systems) at the external boundaries and key internal boundaries of the information systems; and (ii) employ architectural designs, software development techniques, and systems engineering principles that promote effective information security within organizational information systems.	0	0
System And Information Integrity*	Organizations must: (i) identify, report, and correct information and information system flaws in a timely manner; (ii) provide protection from malicious code at appropriate locations within organizational information systems; and (iii) monitor information system security alerts and advisories and take appropriate actions in response.	1	1

* Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - NIST SP 800-53

Category	Issues Found	Best Fix Locations
AC-12 Session Termination (P2)	0	0
AC-3 Access Enforcement (P1)	0	0
AC-4 Information Flow Enforcement (P1)	0	0
AC-6 Least Privilege (P1)	0	0
AU-9 Protection of Audit Information (P1)	0	0
CM-6 Configuration Settings (P2)	0	0
IA-5 Authenticator Management (P1)	0	0
IA-6 Authenticator Feedback (P2)	0	0
IA-8 Identification and Authentication (Non-Organizational Users) (P1)	0	0
SC-12 Cryptographic Key Establishment and Management (P1)	0	0
SC-13 Cryptographic Protection (P1)	0	0
SC-17 Public Key Infrastructure Certificates (P1)	0	0
SC-18 Mobile Code (P2)	1	1
SC-23 Session Authenticity (P1)*	0	0
SC-28 Protection of Information at Rest (P1)*	1	1
SC-4 Information in Shared Resources (P1)	2	2
SC-5 Denial of Service Protection (P1)*	2	1
SC-8 Transmission Confidentiality and Integrity (P1)	1	1
SI-10 Information Input Validation (P1)*	0	0
SI-11 Error Handling (P2)*	4	3
SI-15 Information Output Filtering (P0)*	1	1
SI-16 Memory Protection (P1)	0	0

***** Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - Custom

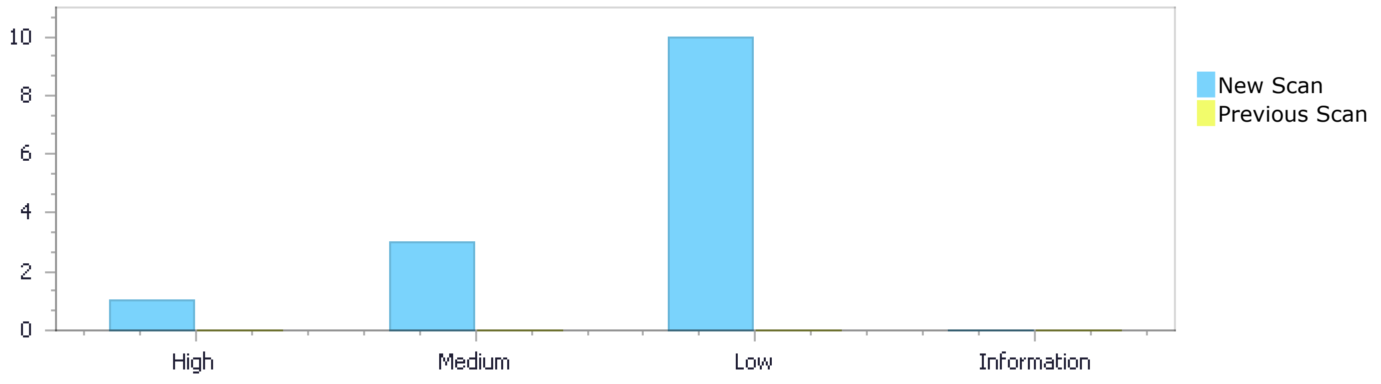
Category	Issues Found	Best Fix Locations
Must audit	0	0
Check	0	0
Optional	0	0

Results Distribution By Status

First scan of the project

	High	Medium	Low	Information	Total
New Issues	1	3	10	0	14
Recurrent Issues	0	0	0	0	0
Total	1	3	10	0	14

Fixed Issues	0	0	0	0	0



Results Distribution By State

	High	Medium	Low	Information	Total
Confirmed	0	0	0	0	0
Not Exploitable	0	0	0	0	0
To Verify	1	3	10	0	14
Urgent	0	0	0	0	0
Proposed Not Exploitable	0	0	0	0	0
Total	1	3	10	0	14

Result Summary

Vulnerability Type	Occurrences	Severity
Deserialization of Untrusted Data	1	High
Heap Inspection	2	Medium
Reflected XSS Specific Clients	1	Medium
Information Exposure Through an Error Message	4	Low
Improper Resource Shutdown or Release	2	Low

Client Insufficient ClickJacking Protection	1	Low
Client Side Only Validation	1	Low
Missing X Frame Options	1	Low
Password in Configuration File	1	Low

10 Most Vulnerable Files

High and Medium Vulnerabilities

File Name	Issues Found
/UserLoader/login/login.ascx.cs	2
/UserLoader/UserLoader/XmlHelpers.cs	1
/UserLoader/UserLoader/EntrustRP.cs	1
/UserLoader/login/login.ascx	1

Scan Results Details

Deserialization of Untrusted Data

Query Path:
CSharp\Cx\CSharp High Risk\Deserialization of Untrusted Data Version:1

Description

Deserialization of Untrusted Data\Path 1:

Severity	High
Result State	To Verify
Online Results	http://ACTIMIZE-IFM-DE/CxWebClient/ViewerMain.aspx?scanid=1000251&projectid=232&pathid=13
Status	New

	Source	Destination
File	/UserLoader/UserLoader/XmlHelpers.cs	/UserLoader/UserLoader/EntrustRP.cs
Line	79	28
Object	ReadToEnd	Deserialize

Code Snippet

File Name	/UserLoader/UserLoader/XmlHelpers.cs
Method	public static string XmlProcessor(string Uri, string xml, bool Token_Islive, string authentication)

```
....
79.                            var esb_result = new
StreamReader(responseStream).ReadToEnd();
```

File Name	/UserLoader/UserLoader/EntrustRP.cs
Method	public AuthenticationResponce TokenAuthenticate(EntrustRequest _param)

```
....
28.                         var s =
(AuthenticationResponce)serializer.Deserialize(stringreader);
```

Heap Inspection

Query Path:
CSharp\Cx\CSharp Medium Threat\Heap Inspection Version:1

Categories

FISMA 2014: Media Protection
NIST SP 800-53: SC-4 Information in Shared Resources (P1)

Description

Heap Inspection\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://ACTIMIZE-IFM-

	DE/CxWebClient/ViewerMain.aspx?scanid=1000251&projectid=232&pathid=4
Status	New

Method loginButton_Click at line 73 of /UserLoader/login/login.ascx.cs defines password, which is designated to contain user passwords. However, while plaintext passwords are later assigned to password, this variable is never cleared from memory.

	Source	Destination
File	/UserLoader/login/login.ascx.cs	/UserLoader/login/login.ascx.cs
Line	81	81
Object	password	password

Code Snippet
File Name /UserLoader/login/login.ascx.cs
Method protected void loginButton_Click(object sender, EventArgs e)

```
....
81.                     string password = this.passwordTextBox.Text.Trim();
```

Heap Inspection\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://ACTIMIZE-IFM-DE/CxWebClient/ViewerMain.aspx?scanid=1000251&projectid=232&pathid=5
Status	New

Method Width="200px" at line 32 of /UserLoader/login/login.ascx defines passwordTextBox, which is designated to contain user passwords. However, while plaintext passwords are later assigned to passwordTextBox, this variable is never cleared from memory.

	Source	Destination
File	/UserLoader/login/login.ascx	/UserLoader/login/login.ascx
Line	32	32
Object	passwordTextBox	passwordTextBox

Code Snippet
File Name /UserLoader/login/login.ascx
Method <asp:TextBox ID="passwordTextBox" runat="server" CssClass="NormalTextBox" Width="200px"

```
....
32.                     <asp:TextBox ID="passwordTextBox" runat="server"
CssClass="NormalTextBox"   Width="200px"
```

Reflected XSS Specific Clients

Query Path:
CSharp\Cx\CSharp Medium Threat\Reflected XSS Specific Clients Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.7 - Cross-site scripting (XSS)
OWASP Top 10 2013: A3-Cross-Site Scripting (XSS)

FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-15 Information Output Filtering (P0)

Description

Reflected XSS Specific Clients\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://ACTIMIZE-IFM-DE/CxWebClient/ViewerMain.aspx?scanid=1000251&projectid=232&pathid=14
Status	New

Method Page_Load at line 17 of /UserLoader/login/login.ascx.cs gets a client-side controlled data for the Cookies_USER_NAME_COOKIE element. This element's value is used in client-side code without being properly sanitized or validated and is eventually integrated into the HTML code in Page_Load at line 17 of /UserLoader/login/login.ascx.cs.

	Source	Destination
File	/UserLoader/login/login.ascx.cs	/UserLoader/login/login.ascx.cs
Line	23	27
Object	Cookies_USER_NAME_COOKIE	Text

Code Snippet

File Name	/UserLoader/login/login.ascx.cs
Method	protected void Page_Load(object sender, EventArgs e)

```
....
23.                         HttpCookie usernameCookie =
this.Request.Cookies[USER_NAME_COOKIE];
....
27.                             this.userNameTextBox.Text =
usernameCookie.Value;
```

Information Exposure Through an Error Message

Query Path:
CSharp\Cx\CSharp Low Visibility\Information Exposure Through an Error Message Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.5 - Improper error handling
OWASP Top 10 2013: A6-Sensitive Data Exposure
FISMA 2014: Configuration Management
NIST SP 800-53: SI-11 Error Handling (P2)

Description

Information Exposure Through an Error Message\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://ACTIMIZE-IFM-DE/CxWebClient/ViewerMain.aspx?scanid=1000251&projectid=232&pathid=9
Status	New

Method loginButton_Click at line 73 of /UserLoader/login/login.ascx.cs catches an exception from element ex of an Exception object. This value flows through the code and is eventually output to the user in method loginButton_Click at line 73 of /UserLoader/login/login.ascx.cs. This may enable Information Exposure Through an Error Message.

	Source	Destination
File	/UserLoader/login/login.ascx.cs	/UserLoader/login/login.ascx.cs
Line	86	86
Object	ex	Text

Code Snippet	
File Name	/UserLoader/login/login.ascx.cs
Method	protected void loginButton_Click(object sender, EventArgs e)

```
....
86.                     this.messageLabel.Text = ex.Message;
```

Information Exposure Through an Error Message\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://ACTIMIZE-IFM-DE/CxWebClient/ViewerMain.aspx?scanid=1000251&projectid=232&pathid=10
Status	New

Method VerifyUser at line 25 of /UserLoader/UserLoader/ValidationLoader.cs catches an exception from element InnerException of an Exception object. This value flows through the code and is eventually output to the user in method VerifyUser at line 25 of /UserLoader/UserLoader/ValidationLoader.cs. This may enable Information Exposure Through an Error Message.

	Source	Destination
File	/UserLoader/UserLoader/ValidationLoader.cs	/UserLoader/UserLoader/ValidationLoader.cs
Line	76	76
Object	InnerException	Error

Code Snippet	
File Name	/UserLoader/UserLoader/ValidationLoader.cs
Method	public string VerifyUser(string username, string password, string tokenValue, int portalID, string portalName, string hostName)

```
....
76.                         Logger.Error(ex.InnerException);
```

Information Exposure Through an Error Message\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://ACTIMIZE-IFM-DE/CxWebClient/ViewerMain.aspx?scanid=1000251&projectid=232&pathid=11
Status	New

Method XmlProcessor at line 47 of /UserLoader/UserLoader/XmlHelpers.cs catches an exception from element ex of an Exception object. This value flows through the code and is eventually output to the user in method XmlProcessor at line 47 of /UserLoader/UserLoader/XmlHelpers.cs. This may enable Information Exposure Through an Error Message.

	Source	Destination
File	/UserLoader/UserLoader/XmlHelpers.cs	/UserLoader/UserLoader/XmlHelpers.cs
Line	87	87
Object	ex	Debug

Code Snippet
File Name /UserLoader/UserLoader/XmlHelpers.cs
Method public static string XmlProcessor(string Uri, string xml, bool Token_Islive, string authentication)

```
....
87.                     logger.Debug("xmlProcessor:: ex: " + ex.Message + " StackTrace" + ex.StackTrace);
```

Information Exposure Through an Error Message\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://ACTIMIZE-IFM-DE/CxWebClient/ViewerMain.aspx?scanid=1000251&projectid=232&pathid=12
Status	New

Method XmlProcessor at line 47 of /UserLoader/UserLoader/XmlHelpers.cs catches an exception from element ex of an Exception object. This value flows through the code and is eventually output to the user in method XmlProcessor at line 47 of /UserLoader/UserLoader/XmlHelpers.cs. This may enable Information Exposure Through an Error Message.

	Source	Destination
File	/UserLoader/UserLoader/XmlHelpers.cs	/UserLoader/UserLoader/XmlHelpers.cs
Line	87	87
Object	ex	Debug

Code Snippet
File Name /UserLoader/UserLoader/XmlHelpers.cs
Method public static string XmlProcessor(string Uri, string xml, bool Token_Islive, string authentication)

```
....
87.                     logger.Debug("xmlProcessor:: ex: " + ex.Message + " StackTrace" + ex.StackTrace);
```

Improper Resource Shutdown or Release

Query Path:
CSharp\Cx\CSharp Low Visibility\Improper Resource Shutdown or Release Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.5 - Improper error handling
NIST SP 800-53: SC-5 Denial of Service Protection (P1)

[Description](#)

Improper Resource Shutdown or Release\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://ACTIMIZE-IFM-DE/CxWebClient/ViewerMain.aspx?scanid=1000251&projectid=232&pathid=7
Status	New

The application's GetEmployeeDetails method in /UserLoader/UserLoader/DataAccess.cs defines and initializes the OracleCommand object at 22. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	/UserLoader/UserLoader/DataAccess.cs	/UserLoader/UserLoader/DataAccess.cs
Line	29	34
Object	OracleCommand	Read

Code Snippet
File Name /UserLoader/UserLoader/DataAccess.cs
Method public Employee GetEmployeeDetails(string email)

```
....
29.                     OracleCommand cmd = new OracleCommand();
....
34.                     dr.Read();
```

Improper Resource Shutdown or Release\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://ACTIMIZE-IFM-DE/CxWebClient/ViewerMain.aspx?scanid=1000251&projectid=232&pathid=8
Status	New

The application's GetEmployeeDetails method in /UserLoader/UserLoader/DataAccess.cs defines and initializes the dr object at 22. This object encapsulates a limited computing resource, such as open file streams, database connections, or network streams. This resource is not properly closed and released in all situations.

	Source	Destination
File	/UserLoader/UserLoader/DataAccess.cs	/UserLoader/UserLoader/DataAccess.cs
Line	33	34
Object	dr	Read

Code Snippet
File Name /UserLoader/UserLoader/DataAccess.cs
Method public Employee GetEmployeeDetails(string email)

```
....
33.                 OracleDataReader dr = cmd.ExecuteReader();
34.                 dr.Read();
```

Client Insufficient ClickJacking Protection

Query Path:
JavaScript\Cx\JavaScript Low Visibility\Client Insufficient ClickJacking Protection Version:1

Categories

FISMA 2014: Configuration Management
NIST SP 800-53: SC-8 Transmission Confidentiality and Integrity (P1)

Description
Client Insufficient ClickJacking Protection\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://ACTIMIZE-IFM-DE/CxWebClient/ViewerMain.aspx?scanid=1000251&projectid=232&pathid=1
Status	New

The application does not protect the web page CxJSNS_594921723 in /UserLoader/login/login.ascx from clickjacking attacks, by using proper framebusting scripts.

	Source	Destination
File	/UserLoader/login/login.ascx	/UserLoader/login/login.ascx
Line	1	1
Object	CxJSNS_594921723	CxJSNS_594921723

Code Snippet

File Name	/UserLoader/login/login.ascx
Method	<%@ Control Language="C#" AutoEventWireup="true" CodeFile="~/DesktopModules/GSS_Login/Login.ascx.cs" Inherits="GSS.AppServices.WebUI.Security.Login" %>

```
....
1.  <%@ Control Language="C#" AutoEventWireup="true"
CodeFile="~/DesktopModules/GSS_Login/Login.ascx.cs"
Inherits="GSS.AppServices.WebUI.Security.Login" %>
```

Password in Configuration File

Query Path:
CSharp\Cx\CSharp WebConfig\Password in Configuration File Version:0

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.8 - Improper access control
OWASP Top 10 2013: A6-Sensitive Data Exposure
FISMA 2014: Identification And Authentication
NIST SP 800-53: SC-28 Protection of Information at Rest (P1)

Description
Password in Configuration File\Path 1:

Severity	Low

Result State	To Verify
Online Results	http://ACTIMIZE-IFM-DE/CxWebClient/ViewerMain.aspx?scanid=1000251&projectid=232&pathid=2
Status	New

	Source	Destination
File	/UserLoader/UserLoader/app.config	/UserLoader/UserLoader/app.config
Line	15	15
Object	"Data Source=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=10.100.20.40)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=UBATEST)));User Id=custom;Password=custom;"	"Data Source=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=10.100.20.40)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=UBATEST)));User Id=custom;Password=custom;"

Code Snippet

File Name	/UserLoader/UserLoader/app.config
Method	<?xml version="1.0" encoding="utf-8" ?>

```
....
15.          <add key="upm_connectionstring" value = "Data
Source=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=10.100.20.40)(PORT=1521
))(CONNECT_DATA=(SERVICE_NAME=UBATEST)));User
Id=custom;Password=custom;" />
```

Missing X Frame Options

Query Path:
CSharp\Cx\CSharp WebConfig\Missing X Frame Options Version:0

Categories

NIST SP 800-53: SC-18 Mobile Code (P2)

Description

Missing X Frame Options\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://ACTIMIZE-IFM-DE/CxWebClient/ViewerMain.aspx?scanid=1000251&projectid=232&pathid=3
Status	New

The web-application does not properly utilize the "X-FRAME-OPTIONS" header to restrict embedding web-pages inside of a frame.

	Source	Destination
File	/UserLoader/login/login.ascx.cs	/UserLoader/login/login.ascx.cs
Line	1	1
Object	Import	Import

Code Snippet

File Name	/UserLoader/login/login.ascx.cs

Method	using System;

```
....
1.  using System;
```

Client Side Only Validation

Query Path:
CSharp\Cx\CSharp Low Visibility\Client Side Only Validation Version:0

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.8 - Improper access control
OWASP Top 10 2013: A7-Missing Function Level Access Control

Description

Client Side Only Validation\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://ACTIMIZE-IFM-DE/CxWebClient/ViewerMain.aspx?scanid=1000251&projectid=232&pathid=6
Status	New

No server side validation was found in /UserLoader/login/login.ascx.cs file, using only client side validation is not enough as it is easy to bypass

	Source	Destination
File	/UserLoader/login/login.ascx.cs	/UserLoader/login/login.ascx.cs
Line	13	13
Object	Login	Login

Code Snippet

File Name	/UserLoader/login/login.ascx.cs
Method	public partial class Login : PortalModuleBase

```
....
13.       public partial class Login : PortalModuleBase
```

Deserialization of Untrusted Data

Weakness ID: 502 *(Weakness Variant)* **Status:** Draft

Description

Description Summary

The application deserializes untrusted data without sufficiently verifying that the resulting data will be valid.

Extended Description

It is often convenient to serialize objects for communication or to save them for later use. However, deserialized data or code can often be modified without using the provided accessor functions if it does not use cryptography to protect itself. Furthermore, any cryptography would still be client-side security -- which is a dangerous security assumption.

Data that is untrusted can not be trusted to be well-formed.

Time of Introduction

- Architecture and Design
- Implementation

Applicable Platforms

Languages

All

Common Consequences

Scope	Effect
Availability	If a function is making an assumption on when to terminate, based on a sentry in a string, it could easily never terminate.
Authorization	Code could potentially make the assumption that information in the deserialized object is valid. Functions which make this dangerous assumption could be exploited.

Likelihood of Exploit

Medium

Demonstrative Examples

Example 1

(Bad Code)

Example Language: **Java**

```
try {
File file = new File("object.obj");
ObjectInputStream in = new ObjectInputStream(new FileInputStream(file));
javax.swing.JButton button = (javax.swing.JButton) in.readObject();
in.close();
byte[] bytes = getBytesFromFile(file);
in = new ObjectInputStream(new ByteArrayInputStream(bytes));
button = (javax.swing.JButton) in.readObject();
in.close();
}
```

Potential Mitigations

Phase: Requirements

A deserialization library could be used which provides a cryptographic framework to seal serialized data.

Phase: Implementation

Use the signing features of a language to assure that deserialized data has not been tainted.

Phase: Implementation

When deserializing data populate a new object rather than just deserializing, the result is that the data flows through safe input validation and that the functions are safe.

Phase: Implementation

Explicitly define final readObject() to prevent deserialization. An example of this is:

(Good Code)

Example Language: **Java**

```
private final void readObject(ObjectInputStream in) throws java.io.IOException {
throw new java.io.IOException("Cannot be deserialized"); }
```

Phases: Architecture and Design; Implementation

Make fields transient to protect them from deserialization.

An attempt to serialize and then deserialize a class containing transient fields will result in NULLs where the transient data should be. This is an excellent way to prevent time, environment-based, or sensitive variables from being carried over and used improperly.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Weakness Class	485	Insufficient Encapsulation	**Development Concepts (primary)699**

				Research Concepts (primary)1000

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
CLASP			Deserialization of untrusted data

Content History

Submissions

Submission Date	Submitter	Organization	Source
	CLASP		Externally Mined

Modifications

Modification Date	Modifier	Organization	Source
2008-07-01	Eric Dalci	Cigital	External
	updated Time of Introduction		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Common Consequences, Description, Relationships, Other Notes, Taxonomy Mappings		
2009-10-29	CWE Content Team	MITRE	Internal
	updated Description, Other Notes, Potential Mitigations		

BACK TO TOP

Heap Inspection

Risk

What might happen

All variables stored by the application in unencrypted memory can potentially be retrieved by an unauthorized user, with privlieged access to the machine. For example, a privileged attacker could attach a debugger to the running process, or retrieve the process's memory from the swapfile or crash dump file.
Once the attacker finds the user passwords in memory, these can be reused to easily impersonate the user to the system.

Cause

How does it happen

String variables are immutable - in other words, once a string variable is assigned, its value cannot be changed or removed. Thus, these strings may remain around in memory, possibly in multiple locations, for an indefinite period of time until the garbage collector happens to remove it. Sensitive data, such as passwords, will remain exposed in memory as plaintext with no control over their lifetime.

General Recommendations

How to avoid it

Generic Guidance:

- Do not store senstiive data, such as passwords or encryption keys, in memory in plaintext, even for a short period of time.
- Prefer to use specialized classes that store encrypted memory.
- Alternatively, store secrets temporarily in mutable data types, such as byte arrays, and then promptly zeroize the memory locations.

Specific Recommendations - Java:

- Instead of storing passwords in immutable strings, prefer to use an encrypted memory object, such as SealedObject.

Specific Recommendations - .NET:

- Instead of storing passwords in immutable strings, prefer to use an encrypted memory object, such as SecureString or ProtectedData.

Source Code Examples

Java

Plaintext Password in Immutable String

```
class Heap_Inspection
{
  private string password;

  void setPassword()
  {
      password = System.console().readLine("Enter your password: ");
   }
}
```

Password Protected in Memory

```
class Heap_Inspection_Fixed
{
  private SealedObject password;

  void setPassword()
 {
      byte[] sKey = getKeyFromConfig();
      Cipher c = Cipher.getInstance("AES");
      c.init(Cipher.ENCRYPT_MODE, sKey);

      char[] input = System.console().readPassword("Enter your password: ");
      password = new SealedObject(Arrays.asList(input), c);
   }
}
```

CPP

Vulnerable C code

```
/* Vulnerable to heap inspection */

#include <stdio.h>


void somefunc(){
      printf("Yea, I'm just being called for the heap of it..\n");
}

void authfunc(){
         char* password = (char *) malloc(256);
         char ch;
         ssize_t k;
              int i=0;
         while(k = read(0, &ch, 1) > 0)
         {
                  if (ch == '\n'){
                          password[i]='\0';
                          break;
                  } else{
                          password[i++]=ch;
                          fflush(0);
                  }
         }
         printf("Password: %s\n",&password[0]);
}

int main()

{

    printf("Please enter a password:\n");

      authfunc();
      printf("You can now dump memory to find this password!");
      somefunc();
      gets();
}
```

Safe C code

```
/* Pesumably safe heap */

#include <stdio.h>
#include <string.h>

#define STDIN_FILENO 0

void somefunc(){
        printf("Yea, I'm just being called for the heap of it..\n");
}

void authfunc(){
      char* password = (char*) malloc(256);
      int i=0;
      char ch;
      ssize_t k;
      while(k = read(STDIN_FILENO, &ch, 1) > 0)
      {
             if (ch == '\n'){
                    password[i]='\0';
                    break;
             } else{
                    password[i++]=ch;
                    fflush(0);
             }
      }
      i=0;
      memset(password,'\0',256);
}

int main()

{

      printf("Please enter a password:\n");
      authfunc();
      somefunc();
      char ch;
      while(read(STDIN_FILENO, &ch, 1) > 0)
      {
             if (ch == '\n')
                    break;
      }
}
```

Reflected XSS Specific Clients

Risk

What might happen

An attacker could bypass the regular channels made by the system to avoid Cross-Site Scripting (XSS) attacks in order to inject JavaScript or HTML code. By using social engineering to cause a user to access the website with injection inputs, such as a URL with engineered session attributes, causing the browser to rewrite web pages. The attacker can then pretend to be the original website, which would enable the attacker to steal the user's session, request the user’s credentials, provide false information, or run malware. From the victim’s point of view, this is the original website's action.

Cause

How does it happen

The application web page includes data from a client-side controlled data (including the page session). The user input is embedded in the page, causing the browser to display it as part of the web page. If the input includes HTML fragments or JavaScript, these are displayed too, and the user cannot tell that this is not the intended page. The vulnerability is the result of embedding arbitrary client-side controlled input without first encoding it in a format that would prevent the browser from treating it like HTML instead of plain text.

General Recommendations

How to avoid it

Validate all input, regardless of source. Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns. Check for:

- Data type
- Size
- Range
- Format
- Expected values

Secure-Coding Approach

Fully encode all dynamic data before embedding it in the webpage. Encoding should be context-sensitive. For example:

  - HTML encoding for HTML content.
  - HTML Attribute encoding for data output to attribute values.
  - JavaScript encoding for JavaScript.

It is recommended to use the known libraries for encoding output, such as ESAPI.

Source Code Examples

JavaScript

Dynamically calling a function without sanitizing the input:

```
var input = document.getElementById("id").value;
```

```
window.setInterval( myFunc(input), 1000);
```

Sanitizing input before using it in a function:

```
var input = document.getElementById("id").value;
var trusted = escape(input);
window.setInterval( myFunc(trusted), 1000);
```

For dynamically updating HTML in the DOM, use the OWASP ESAPI4JS library:

```
document.write("<%=Encoder.encodeForJS(Encoder.encodeForHTML(untrustedData))%>");
```

If you must set code to be called dynamically, only call predefined methods or hard-coded Javascript. Never call "eval()" or dynamically create code:

```
window.setInterval( "timedFunction();", 1000);
```

If you must set code to be called dynamically, only call predefined methods or hard-coded Javascript. Never call "eval()" or dynamically create code:

```
window.setInterval( "timedFunction();", 1000);
```

Client Insufficient ClickJacking Protection

Risk

What might happen

Clickjacking attacks allow an attacker to "hijack" a user's mouse clicks on a webpage, by invisibly framing the application, and superimposing it in front of a bogus site. When the user is convinced to click on the bogus website, e.g. on a link or a button, the user's mouse is actually clicking on the target webpage, despite being invisible.
This could allow the attacker to cause the user to perform any undesirable action in the vulnerable application, e.g. enabling the user's webcam, deleting all the user's records, changing the user's settings, or causing clickfraud.

Cause

How does it happen

The root cause of vulnerability to a clickjacking attack, is that the application's web pages can be loaded into a frame of another website. The application does not implement a proper frame-busting script, that would prevent the page from being loaded into another frame. Note that there are many types of simplistic redirection scripts that still leave the application vulnerable to clickjacking techniques, and should not be used.
Additionally, note also that this attack can be best protected by setting appropriate values in the HTTP response headers on the server side, such as a Content Security Policy (CSP) or "X-Frame-Options". If these headers are properly set by the server, this result can be disregarded.

General Recommendations

How to avoid it

Generic Guidance:

- Define and implement a a Content Security Policy (CSP) on the server side, including a frame-ancestors directive. Enforce the CSP on all relevant webpages.
- If certain webpages are required to be loaded into a frame, define a specific, whitelisted target URL.
- Alternatively, return a "X-Frame-Options" header on all HTTP responses. If it is necessary to allow a particular webpage to be loaded into a frame, define a specific, whitelisted target URL.
- Where necessary, implement a proper framebuster script on the client, that is not vulnerable to framebusterbusting.

Specific Recommendations:

- Implement a proper framebuster script on the client, that is not vulnerable to framebusterbusting. In particular, the framebuster script should include CSS to disable the UI by default, and a default disable JavaScript framebuster.

Source Code Examples

JavaScript
Clickjackable Webpage

```
<html>
    <body>

     <button onclick="clicked();">
            Click here if you love ducks
        </button>
    </body>
</html>
```

Bustable Framebuster

```
<html>
    <head>

     <script>
             if ( window.self.location != window.top.location ) {
                    window.top.location = window.self.location;
             }
        </script>
    </head>

    <body>
        <button onclick="clicked();">
            Click here if you love ducks
        </button>
    </body>
</html>
```

Proper Framebusterbusterbusting

```
<html>
    <head>

     <style> html {display : none; } </style>
        <script>
             if ( self == top ) {
                    document.documentElement.style.display = 'block';
             }
             else {
                    top.location = self.location;
             }
        </script>
    </head>

    <body>
        <button onclick="clicked();">
            Click here if you love ducks
        </button>
    </body>
</html>
```

Password in Configuration File

Weakness ID: 260 *(Weakness Variant)* **Status:** Incomplete

Description

Description Summary

The software stores a password in a configuration file that might be accessible to actors who do not know the password.

Extended Description

This can result in compromise of the system for which the password is used. An attacker could gain access to this file and learn the stored password or worse yet, change the password to one of their choosing.

Time of Introduction

- Architecture and Design
- Implementation

Applicable Platforms

Languages

All

Demonstrative Examples

Example 1

Below is a snippet from a Java properties file in which the LDAP server password is stored in plaintext.

(Bad Code)

Example Language: **Java**

```
webapp.ldap.username=secretUsername
webapp.ldap.password=secretPassword
```

Potential Mitigations

Avoid storing passwords in easily accessible locations.

Consider storing cryptographic hashes of passwords as an alternative to storing in plaintext.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Category	254	Security Features	Development Concepts699 **Seven Pernicious Kingdoms (primary)700**
ChildOf	Weakness Base	522	Insufficiently Protected Credentials	**Development Concepts (primary)699 Research Concepts (primary)1000**
ChildOf	Category	632	Weaknesses that Affect Files or Directories	**Resource-specific Weaknesses (primary)631**
ParentOf	Weakness Variant	13	ASP.NET Misconfiguration: Password in Configuration File	**Research Concepts (primary)1000**
ParentOf	Weakness Variant	258	Empty Password in Configuration File	**Development Concepts (primary)699 Research Concepts (primary)1000**

Affected Resources

- File/Directory

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
7 Pernicious Kingdoms			Password Management: Password in Configuration File

References

J. Viega and G. McGraw. "Building Secure Software: How to Avoid Security Problems the Right Way". 2002.

Content History

Submissions

Submission Date	Submitter	Organization	Source
	7 Pernicious Kingdoms		Externally Mined

Modifications

Modification Date	Modifier	Organization	Source
2008-07-01	Sean Eidemiller	Cigital	External
	added/updated demonstrative examples		
2008-07-01	Eric Dalci	Cigital	External
	updated Time of Introduction		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Relationships, Taxonomy Mappings		
2008-10-14	CWE Content Team	MITRE	Internal
	updated Description		

BACK TO TOP

Missing X Frame Options

Risk

What might happen

Allowing setting of web-pages inside of a frame in an untrusted web-page will leave these web-pages vulnerable to Clickjacking, otherwise known as a redress attack. This may allow an attacker to redress a vulnerable web-page by setting it inside a frame within a malicious web-page. By crafting a convincing malicious web-page, the attacker can then use the overlayed redress to convince the user to click a certain area of the screen, unknowingly clicking inside the frame containing the vulnerable web-page, and thus performing actions within the user's context on the attacker's behalf.

Cause

How does it happen

Failure to utilize the "X-FRAME-OPTIONS" header will likely allow attackers to perform Clickjacking attacks. Properly utilizing the "X-FRAME-OPTIONS" header would indicate to the browser to disallow embedding the web-page within a frame, mitigating this risk, if the browser supports this header. All modern browsers support this header by default.

General Recommendations

How to avoid it

Utilize the "X-FRAME-OPTIONS" header flags according to business requirements to restrict browsers that support this header from allowing embedding web-pages in a frame:

- "X-Frame-Options: DENY" will indicate to the browser to disallow embedding any web-page inside a frame, including the current web-site.
- "X-Frame-Options: SAMEORIGIN" will indicate to the browser to disallow embedding any web-page inside a frame, excluding the current web-site.
- "X-Frame-Options: ALLOW-FROM https://example.com/" will indicate to the browser to disallow embedding any web-page inside a frame, excluding the web-site listed after the ALLOW-FROM parameter.

Source Code Examples

Java

Setting the "DENY" Flag on a Response

```java
response.addHeader("X-Frame-Options", "DENY");
```

Client Side Only Validation

Risk

What might happen

Bypassing a client side validation may lead to unexpected and tanpered data to the server.

Cause

How does it happen

Relience on client side validation only

General Recommendations

How to avoid it

It is highly recommended to validate the input in the server side, alongside client side validation.

Source Code Examples

Improper Resource Shutdown or Release

Risk

What might happen

Unreleased resources can cause a drain of those available for system use, eventually causing general reliability and availability problems, such as performance degradation, process bloat, and system instability. If a resource leak can be intentionally exploited by an attacker, it may be possible to cause a widespread DoS (Denial of Service) attack. This might even expose sensitive information between unprivileged users, if the resource continues to retain data or user id between subsequent allocations.

Cause

How does it happen

The application code allocates resource objects, but does not ensure these are always closed and released in a timely manner. This can include database connections, file handles, network sockets, or any other resource that needs to be released. In some cases, these might be released - but only if everything works as planned; if there is any runtime exception during the normal course of system operations, resources start to leak.

Note that even in managed-memory languages such as Java, these resources must be explicitly released. Many types of resource are not released even when the Garbage Collector runs; and even if the the object would eventually release the resource, we have no control over when the Garbage Collector does run.

General Recommendations

How to avoid it

- Always close and release all resources.
- Ensure resources are released (along with any other necessary cleanup) in a `finally { }` block. Do not close resources in a `catch { }` block, since this is not ensured to be called.
- Explicitly call .close() on any instance of a class that implements the `Closable` or `AutoClosable` interfaces.
- Alternatively, an even better solution is to use the try-with-resources idiom, in order to automatically close any defined `AutoClosable` instances.

Source Code Examples

Java

Unreleased Database Connection

```
private MyObject getDataFromDb(int id)  {
      MyObject data = null;

      try {
             Connection con = DriverManager.getConnection(CONN_STRING);

             data = queryDb(con, id);
      }
      catch ( SQLException e ) {
             handleError(e);
      }
}
```

Explicit Release of Database Connection

```
private MyObject getDataFromDb(int id)  {
      MyObject data = null;

      try {
             Connection con = DriverManager.getConnection(CONN_STRING);

             data = queryDb(con, id);
      }
      catch ( SQLException e ) {
             handleError(e);
      }
      finally {
             if ((con != null) && (! con.isClosed())) {
             con.close();
         }
     }
}
```

Automatic Implicit Release Using Try-With-Resources

```
private MyObject getDataFromDb(int id)  {
      MyObject data = null;

      try (Connection con = DriverManager.getConnection(CONN_STRING)) {
             data = queryDb(con, id);
      }
      catch ( SQLException e ) {
             handleError(e);
      }
}
```

Information Exposure Through an Error Message

Risk

What might happen

Exposed details about the application's environment, users, or associated data (for example, stack trace) could enable an attacker to find another flaw and help the attacker to mount an attack.

Cause

How does it happen

The application generates an error message including raw exceptions, either by not being handled, by explicit returning of the object, or by configuration. Exception details may include sensitive information that could leak out of the exception to the users.

General Recommendations

How to avoid it

1. Any method that could cause an exception should be wrapped in a try-catch block that:
   - Explicitly handles expected exceptions.
   - Includes a default solution to explicitly handle unexpected exceptions.
2. Configure a global handler to prevent unhandled errors from leaving the application.

Source Code Examples

CSharp

Do not reveal exception details, instead always return a static message.

```
try
{
// Database access or other potentially dangerous function
}
catch (SqlException ex)
{
LogException(ex);
Response.Write("Error occurred.");
}
```

Java

Do not reveal exception details, instead always return a static message.

```
try
{
// Database access or other potentially dangerous function
}
catch (SqlException ex)
{
LogException(ex);
Response.Write("Error occurred.");
}
```

Scanned Languages

Language	Hash Number	Change Date
CSharp	0143992024714110	10/13/2017
JavaScript	1662591359214638	10/13/2017
VbScript	1349101913133594	10/13/2017
Common	1661208495016619	10/13/2017