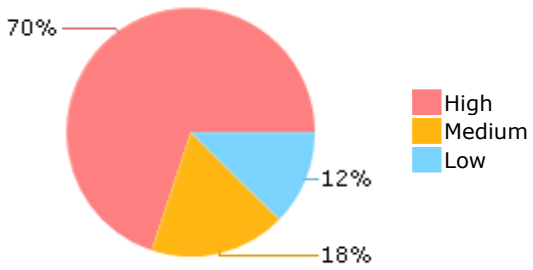
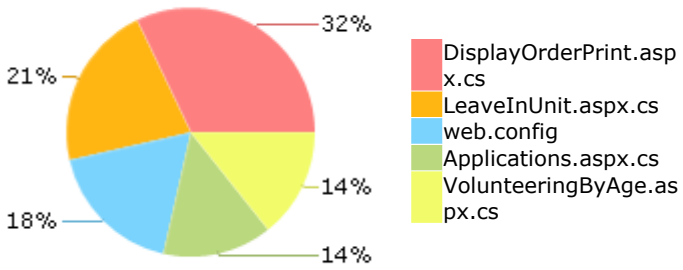
Miluim_Hayalim_Site Scan Report

Project Name	Miluim_Hayalim_Site
Scan Start	Monday, December 28, 2015 6:35:36 PM
Preset	Default 2014
Scan Time	00h:05m:01s
Lines Of Code Scanned	72,997
Files Scanned	201
Report Creation Time	Monday, December 28, 2015 7:38:48 PM
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36
Team	Users
Checkmarx Version	7.1.8 HF2
Scan Type	Full
Source Origin	LocalPath
Density	1/1000 (Vulnerabilities/LOC)

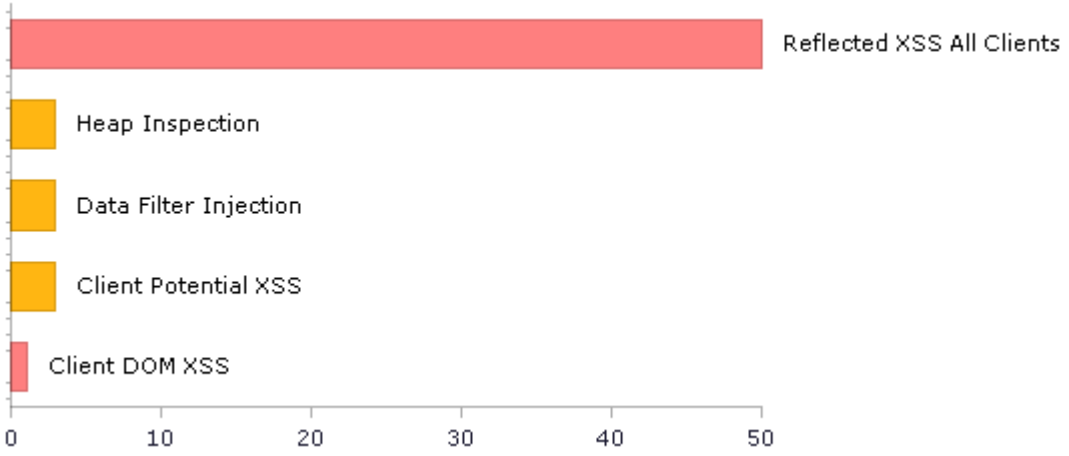
Result Summary



Most Vulnerable Files



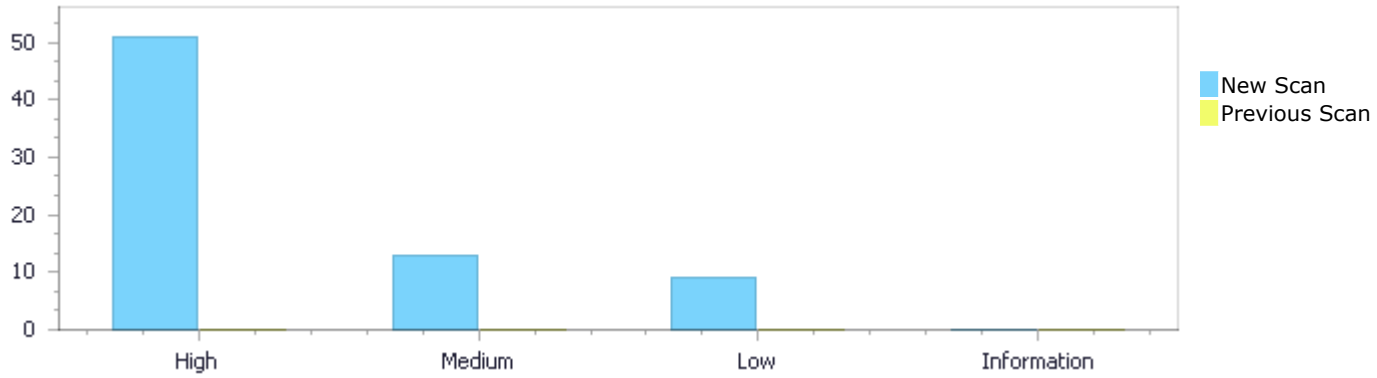
Top 5 Vulnerabilities



Results Distribution By Status

First scan of the project

	High	Medium	Low	Information	Total
New Issues	51	13	9	0	73
Recurrent Issues	0	0	0	0	0
Total	51	13	9	0	73
Fixed Issues	0	0	0	0	0



Results Distribution By State

	High	Medium	Low	Information	Total
To Verify	51	13	9	0	73
Not Exploitable	0	0	0	0	0
Confirmed	0	0	0	0	0
Urgent	0	0	0	0	0
Total	51	13	9	0	73

Result Summary

Vulnerability Type	Occurrences	Severity
Reflected XSS All Clients	50	High
Client DOM XSS	1	High
Client Potential XSS	3	Medium
Data Filter Injection	3	Medium
Heap Inspection	3	Medium
Client Cross Frame Scripting Attack	1	Medium
CookieLess Authentication	1	Medium
Reflected XSS Specific Clients	1	Medium
RequireSSL	1	Medium
Client Heuristic Poor XSS Validation	2	Low
Unprotected Cookie	2	Low
Client DOM Open Redirect	1	Low

Client Insecure Randomness	1	Low
DebugEnabled	1	Low
NonUniqueFormName	1	Low
SlidingExpiration	1	Low

10 Most Vulnerable Files

High and Medium Vulnerabilities

File Name	Issues Found
/MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs	9
/MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs	6
/MiluimHayalim-Site/Templates/Applications/Applications.aspx.cs	4
/MiluimHayalim-Site/Templates/Forms/VolunteeringByAge.aspx.cs	4
/MiluimHayalim-Site/Templates/Login/Login.aspx.cs	4
/MiluimHayalim-Site/Ajax/Order/DisplayOrder.aspx.cs	3
/MiluimHayalim-Site/Templates/Forms/ChangeUnit.aspx.cs	3
/MiluimHayalim-Site/Templates/Forms/GeneralPetition.aspx.cs	3
/MiluimHayalim-Site/Templates/Login/SMSVerify.aspx.cs	3
/MiluimHayalim-Site/Js/Forms/forms.js	3

Scan Results Details

Number of results is limited to 50 for each vulnerability. To get more results please change a setting "Limit results to 50" in report creation wizard.

Reflected XSS All Clients

Description

Reflected XSS All Clients\Path 25:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=25
Status	New

Method DisplayOtekTzavRemark at line 119 of /MiluimHayalim-Site/Ajax/Order/DisplayConfirmMessage.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method DisplayOtekTzavRemark at line 119 of /MiluimHayalim-Site/Ajax/Order/DisplayConfirmMessage.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Ajax/Order/DisplayConfirmMessage.aspx.cs	/MiluimHayalim-Site/Ajax/Order/DisplayConfirmMessage.aspx.cs
Line	130	130
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Ajax/Order/DisplayConfirmMessage.aspx.cs
Method private void DisplayOtekTzavRemark()

```
....
130.                                ltrlDays.Text =
Utils.GetObjectContent(row["sending_order_days"], 0).ToString();
```

Reflected XSS All Clients\Path 26:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=26
Status	New

Method GetDBData at line 75 of /MiluimHayalim-Site/Ajax/Order/DisplayOrder.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetDBData at line 75 of /MiluimHayalim-Site/Ajax/Order/DisplayOrder.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Ajax/Order/DisplayOrder.aspx.cs	/MiluimHayalim-Site/Ajax/Order/DisplayOrder.aspx.cs
Line	88	88

Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Ajax/Order/DisplayOrder.aspx.cs
Method private void GetDBData()

```
....
88.                          ltrlFullName.Text = string.Format("{0}
{1}", Utils.GetObjectContent(row["SHEM_PRATI"]),
Utils.GetObjectContent(row["SHEM_MISHPACHA"]));
```

Reflected XSS All Clients\Path 27:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=27
Status	New

Method GetDBData at line 75 of /MiluimHayalim-Site/Ajax/Order/DisplayOrder.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetDBData at line 75 of /MiluimHayalim-Site/Ajax/Order/DisplayOrder.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Ajax/Order/DisplayOrder.aspx.cs	/MiluimHayalim-Site/Ajax/Order/DisplayOrder.aspx.cs
Line	88	88
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Ajax/Order/DisplayOrder.aspx.cs
Method private void GetDBData()

```
....
88.                          ltrlFullName.Text = string.Format("{0}
{1}", Utils.GetObjectContent(row["SHEM_PRATI"]),
Utils.GetObjectContent(row["SHEM_MISHPACHA"]));
```

Reflected XSS All Clients\Path 28:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=28
Status	New

Method GetDBData at line 75 of /MiluimHayalim-Site/Ajax/Order/DisplayOrder.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetDBData at line 75 of /MiluimHayalim-Site/Ajax/Order/DisplayOrder.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination

File	/MiluimHayalim-Site/Ajax/Order/DisplayOrder.aspx.cs	/MiluimHayalim-Site/Ajax/Order/DisplayOrder.aspx.cs
Line	89	89
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Ajax/Order/DisplayOrder.aspx.cs
Method private void GetDBData()

```
....
89.                          ltrlOrderNum.Text =
Utils.GetObjectContent(row["MISPAR_TZAV"]);
```

Reflected XSS All Clients\Path 29:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=29
Status	New

Method GetDBData at line 79 of /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetDBData at line 79 of /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs	/MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs
Line	92	92
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs
Method private void GetDBData()

```
....
92.                          ltrlFullName.Text = string.Format("{0}
{1}",
Utils.GetObjectContent(row["SHEM_PRATI"]),Utils.GetObjectContent(row["SH
EM_MISHPACHA"]));
```

Reflected XSS All Clients\Path 30:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=30
Status	New

Method GetDBData at line 79 of /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs gets user input for the row element. This element's value then flows through the code without being

properly sanitized or validated and is eventually displayed to the user in method GetDBData at line 79 of /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs	/MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs
Line	92	92
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs
Method private void GetDBData()

```
....
92.                             ltrlFullName.Text = string.Format("{0} {1}",
Utils.GetObjectContent(row["SHEM_PRATI"]),Utils.GetObjectContent(row["SHEM_MISHPACHA"]));
```

Reflected XSS All Clients\Path 31:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=31
Status	New

Method GetDBData at line 79 of /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetDBData at line 79 of /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs	/MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs
Line	93	93
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs
Method private void GetDBData()

```
....
93.                             ltrlOrderNum.Text = Utils.GetObjectContent(row["MISPAR_TZAV"]);
```

Reflected XSS All Clients\Path 32:

Severity	High
Result State	To Verify

Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=32
Status	New

Method SetCommOfficerDetails at line 107 of /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method SetCommOfficerDetails at line 107 of /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs	/MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs
Line	111	111
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs
Method private void SetCommOfficerDetails(DataRow row)

```
....
111.                    ltrlOffMisparIshi.Text =
Utils.GetObjectContent(row["MISPAR_ISHI_KTZINA_BATZAV"]);
```

Reflected XSS All Clients\Path 33:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=33
Status	New

Method SetCommOfficerDetails at line 107 of /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method SetCommOfficerDetails at line 107 of /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs	/MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs
Line	112	112
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs
Method private void SetCommOfficerDetails(DataRow row)

```
....
112.                    ltrlOffDarga.Text =
Utils.GetObjectContent(row["SHEM_DARGA"]);
```

Reflected XSS All Clients\Path 34:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=34
Status	New

Method SetCommOfficerDetails at line 107 of /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method SetCommOfficerDetails at line 107 of /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs	/MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs
Line	113	113
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs
Method private void SetCommOfficerDetails(DataRow row)

```
....
113.                    ltrlOffFName.Text =
Utils.GetObjectContent(row["SHEM_PRATI_KTZINA_BATZAV"]);
```

Reflected XSS All Clients\Path 35:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=35
Status	New

Method SetCommOfficerDetails at line 107 of /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method SetCommOfficerDetails at line 107 of /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs	/MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs

Line	114	114
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs
Method private void SetCommOfficerDetails(DataRow row)

```
....
114.                    ltrlOffLName.Text =
Utils.GetObjectContent(row["SHEM_MISHPACHA_KTZINA_BATZAV"]);
```

Reflected XSS All Clients\Path 36:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=36
Status	New

Method SetCommOfficerDetails at line 107 of /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method SetCommOfficerDetails at line 107 of /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs	/MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs
Line	115	115
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs
Method private void SetCommOfficerDetails(DataRow row)

```
....
115.                    ltrlOffTafkid.Text =
Utils.GetObjectContent(row["TAFKID_KTZINA_BATZAV"]);
```

Reflected XSS All Clients\Path 37:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=37
Status	New

Method SetCommOfficerDetails at line 107 of /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method SetCommOfficerDetails at line 107 of

/MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs	/MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs
Line	116	116
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Ajax/Order/DisplayOrderPrint.aspx.cs
Method private void SetCommOfficerDetails(DataRow row)

```
....
116.                    ltrlOffDoar.Text =
Utils.GetObjectContent(row["DOAR_TZVAI_BATZAV"]);
```

Reflected XSS All Clients\Path 38:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=38
Status	New

Method GetKSData at line 83 of /MiluimHayalim-Site/Templates/Applications/Applications.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 83 of /MiluimHayalim-Site/Templates/Applications/Applications.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Applications/Applications.aspx.cs	/MiluimHayalim-Site/Templates/Applications/Applications.aspx.cs
Line	92	92
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Applications/Applications.aspx.cs
Method private void GetKSData()

```
....
92.                             ltrlOrederPreviewText.Text =
Utils.GetObjectContent(row["applications_by_order_preview_text"]);
```

Reflected XSS All Clients\Path 39:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=39

Status	New

Method GetKSData at line 83 of /MiluimHayalim-Site/Templates/Applications/Applications.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 83 of /MiluimHayalim-Site/Templates/Applications/Applications.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Applications/Applications.aspx.cs	/MiluimHayalim-Site/Templates/Applications/Applications.aspx.cs
Line	92	93
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Applications/Applications.aspx.cs
Method private void GetKSData()

```
....
92.                          ltrlOrederPreviewText.Text =
Utils.GetObjectContent(row["applications_by_order_preview_text"]);
93.                          if (ltrlOrederPreviewText.Text !=
string.Empty)
```

Reflected XSS All Clients\Path 40:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=40
Status	New

Method GetKSData at line 83 of /MiluimHayalim-Site/Templates/Applications/Applications.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 83 of /MiluimHayalim-Site/Templates/Applications/Applications.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Applications/Applications.aspx.cs	/MiluimHayalim-Site/Templates/Applications/Applications.aspx.cs
Line	97	97
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Applications/Applications.aspx.cs
Method private void GetKSData()

```
....
97.                          ltrlAppPreviewText.Text =
Utils.GetObjectContent(row["applications_preview_text"]);
```

Reflected XSS All Clients\Path 41:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=41
Status	New

Method GetKSData at line 83 of /MiluimHayalim-Site/Templates/Applications/Applications.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 83 of /MiluimHayalim-Site/Templates/Applications/Applications.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Applications/Applications.aspx.cs	/MiluimHayalim-Site/Templates/Applications/Applications.aspx.cs
Line	97	98
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Applications/Applications.aspx.cs
Method private void GetKSData()

```
....
97.                          ltrlAppPreviewText.Text =
Utils.GetObjectContent(row["applications_preview_text"]);
98.                          if (ltrlAppPreviewText.Text !=
string.Empty)
```

Reflected XSS All Clients\Path 42:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=42
Status	New

Method GetKSData at line 83 of /MiluimHayalim-Site/Templates/Forms/ChangeUnit.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 83 of /MiluimHayalim-Site/Templates/Forms/ChangeUnit.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Forms/ChangeUnit.aspx.cs	/MiluimHayalim-Site/Templates/Forms/ChangeUnit.aspx.cs

Line	92	92
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Forms/ChangeUnit.aspx.cs
Method private void GetKSData()

```
....
92.                          ltrlPreviewText.Text =
Utils.GetObjectContent(row["change_unit_preview_text"]);
```

Reflected XSS All Clients\Path 43:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=43
Status	New

Method GetKSData at line 83 of /MiluimHayalim-Site/Templates/Forms/ChangeUnit.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 83 of /MiluimHayalim-Site/Templates/Forms/ChangeUnit.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Forms/ChangeUnit.aspx.cs	/MiluimHayalim-Site/Templates/Forms/ChangeUnit.aspx.cs
Line	92	93
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Forms/ChangeUnit.aspx.cs
Method private void GetKSData()

```
....
92.                          ltrlPreviewText.Text =
Utils.GetObjectContent(row["change_unit_preview_text"]);
93.                          if (ltrlPreviewText.Text != string.Empty)
```

Reflected XSS All Clients\Path 44:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=44
Status	New

Method GetKSData at line 83 of /MiluimHayalim-Site/Templates/Forms/ChangeUnit.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 83 of /MiluimHayalim-Site/Templates/Forms/ChangeUnit.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Forms/ChangeUnit.aspx.cs	/MiluimHayalim-Site/Templates/Forms/ChangeUnit.aspx.cs
Line	110	110
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Forms/ChangeUnit.aspx.cs
Method private void GetKSData()

```
....
110.                            ltrlTerms.Text =
Utils.GetObjectContent(row["change_unit_terms"]);
```

Reflected XSS All Clients\Path 45:

Severity High
Result State To Verify
Online Results http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=45
Status New

Method GetKSData at line 110 of /MiluimHayalim-Site/Templates/Forms/Form3010.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 110 of /MiluimHayalim-Site/Templates/Forms/Form3010.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Forms/Form3010.aspx.cs	/MiluimHayalim-Site/Templates/Forms/Form3010.aspx.cs
Line	119	119
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Forms/Form3010.aspx.cs
Method private void GetKSData()

```
....
119.                            ltrlPreviewText.Text =
Utils.GetObjectContent(row["preview_text_3010"]);
```

Reflected XSS All Clients\Path 46:

Severity High
Result State To Verify
Online Results http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=46
Status New

Method GetKSData at line 110 of /MiluimHayalim-Site/Templates/Forms/Form3010.aspx.cs gets user input for the row element. This element's value then flows through the code without being

properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 110 of /MiluimHayalim-Site/Templates/Forms/Form3010.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Forms/Form3010.aspx.cs	/MiluimHayalim-Site/Templates/Forms/Form3010.aspx.cs
Line	119	120
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Forms/Form3010.aspx.cs
Method private void GetKSData()

```
....
119.                            ltrlPreviewText.Text =
Utils.GetObjectContent(row["preview_text_3010"]);
120.                            if (ltrlPreviewText.Text == string.Empty)
```

Reflected XSS All Clients\Path 47:

Severity High
Result State To Verify
Online Results http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=47
Status New

Method GetKSData at line 76 of /MiluimHayalim-Site/Templates/Forms/GeneralPetition.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 76 of /MiluimHayalim-Site/Templates/Forms/GeneralPetition.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Forms/GeneralPetition.aspx.cs	/MiluimHayalim-Site/Templates/Forms/GeneralPetition.aspx.cs
Line	85	85
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Forms/GeneralPetition.aspx.cs
Method private void GetKSData()

```
....
85.                             ltrlPreviewText.Text =
Utils.GetObjectContent(row["gen_petition_preview_text"]);
```

Reflected XSS All Clients\Path 48:

Severity High
Result State To Verify
Online Results http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=48

Status	New

Method GetKSData at line 76 of /MiluimHayalim-Site/Templates/Forms/GeneralPetition.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 76 of /MiluimHayalim-Site/Templates/Forms/GeneralPetition.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Forms/GeneralPetition.aspx.cs	/MiluimHayalim-Site/Templates/Forms/GeneralPetition.aspx.cs
Line	85	86
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Forms/GeneralPetition.aspx.cs
Method private void GetKSData()

```
....
85.                             ltrlPreviewText.Text =
Utils.GetObjectContent(row["gen_petition_preview_text"]);
86.                             if (ltrlPreviewText.Text == string.Empty)
```

Reflected XSS All Clients\Path 49:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=49
Status	New

Method GetBakashot at line 121 of /MiluimHayalim-Site/Templates/Forms/GeneralPetition.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetBakashot at line 121 of /MiluimHayalim-Site/Templates/Forms/GeneralPetition.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Forms/GeneralPetition.aspx.cs	/MiluimHayalim-Site/Templates/Forms/GeneralPetition.aspx.cs
Line	144	162
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Forms/GeneralPetition.aspx.cs
Method private void GetBakashot()

```
....
144.                                SugBakashaId =
Utils.GetObjectContent(row["SugBakashaKlalitID"]);
....
162.                            ltrlJS.Text = string.Format("\n <script
type='text/javascript'>{0}</script> \n", sb.ToString());
```

Reflected XSS All Clients\Path 50:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=50
Status	New

Method GetKSData at line 157 of /MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 157 of /MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs	/MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs
Line	166	166
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs
Method private void GetKSData()

```
....
166.                            ltrlPreviewText.Text =
Utils.GetObjectContent(row["leave_in_unit_preview_text"]);
```

Reflected XSS All Clients\Path 51:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=51
Status	New

Method GetKSData at line 157 of /MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 157 of /MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs	/MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs

Line	166	167
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs
Method private void GetKSData()

```
....
166.                            ltrlPreviewText.Text =
Utils.GetObjectContent(row["leave_in_unit_preview_text"]);
167.                            if (ltrlPreviewText.Text != string.Empty)
```

Reflected XSS All Clients\Path 52:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=52
Status	New

Method GetKSData at line 157 of /MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 157 of /MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs	/MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs
Line	183	183
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs
Method private void GetKSData()

```
....
183.                            ltrlTwoBrothersText.Text =
Utils.GetObjectContent(row["leave_in_unit_two_brothers_text"]);
```

Reflected XSS All Clients\Path 53:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=53
Status	New

Method GetKSData at line 157 of /MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 157 of /MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs	/MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs
Line	197	197
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs
Method private void GetKSData()

```
....
197.                            ltrlProfileText.Text =
Utils.GetObjectContent(row["leave_in_unit_profile_text"]);
```

Reflected XSS All Clients\Path 54:

Severity High
Result State To Verify
Online Results http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=54
Status New

Method GetKSData at line 157 of /MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 157 of /MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs	/MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs
Line	198	198
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs
Method private void GetKSData()

```
....
198.                            ltrlViturText.Text =
Utils.GetObjectContent(row["leave_in_unit_vitur_text"]);
```

Reflected XSS All Clients\Path 55:

Severity High
Result State To Verify
Online Results http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=55
Status New

Method GetKSData at line 157 of /MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 157 of /MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs	/MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs
Line	199	199
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Forms/LeaveInUnit.aspx.cs
Method private void GetKSData()

```
....
199.                            ltrlGeneralText.Text =
Utils.fromTextArea(row["leave_in_unit_general_text"]);
```

Reflected XSS All Clients\Path 56:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=56
Status	New

Method GetKSData at line 82 of /MiluimHayalim-Site/Templates/Forms/MedicalCommitteeRequest.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 82 of /MiluimHayalim-Site/Templates/Forms/MedicalCommitteeRequest.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Forms/MedicalCommitteeRequest.aspx.cs	/MiluimHayalim-Site/Templates/Forms/MedicalCommitteeRequest.aspx.cs
Line	91	91
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Forms/MedicalCommitteeRequest.aspx.cs
Method private void GetKSData()

```
....
91.                             ltrlPreviewText.Text =
Utils.GetObjectContent(row["medical_com_preview_text"]);
```

Reflected XSS All Clients\Path 57:

Severity	High

Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=57
Status	New

Method GetKSData at line 82 of /MiluimHayalim-Site/Templates/Forms/MedicalCommitteeRequest.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 82 of /MiluimHayalim-Site/Templates/Forms/MedicalCommitteeRequest.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Forms/MedicalCommitteeRequest.aspx.cs	/MiluimHayalim-Site/Templates/Forms/MedicalCommitteeRequest.aspx.cs
Line	91	92
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Forms/MedicalCommitteeRequest.aspx.cs
Method private void GetKSData()

```
....
91.                             ltrlPreviewText.Text =
Utils.GetObjectContent(row["medical_com_preview_text"]);
92.                             if (ltrlPreviewText.Text != string.Empty)
```

Reflected XSS All Clients\Path 58:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=58
Status	New

Method GetKSData at line 89 of /MiluimHayalim-Site/Templates/Forms/MentalHealthOfficerContacting.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 89 of /MiluimHayalim-Site/Templates/Forms/MentalHealthOfficerContacting.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Forms/MentalHealthOfficerContacting.aspx.cs	/MiluimHayalim-Site/Templates/Forms/MentalHealthOfficerContacting.aspx.cs
Line	98	98
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Forms/MentalHealthOfficerContacting.aspx.cs
Method private void GetKSData()

```
....
98.                          ltrlPreviewText.Text =
Utils.GetObjectContent(row["mental_health_preview_text"]);
```

Reflected XSS All Clients\Path 59:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=59
Status	New

Method GetKSData at line 89 of /MiluimHayalim-Site/Templates/Forms/MentalHealthOfficerContacting.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 89 of /MiluimHayalim-Site/Templates/Forms/MentalHealthOfficerContacting.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Forms/MentalHealthOfficerContacting.aspx.cs	/MiluimHayalim-Site/Templates/Forms/MentalHealthOfficerContacting.aspx.cs
Line	98	99
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Forms/MentalHealthOfficerContacting.aspx.cs
Method private void GetKSData()

```
....
98.                          ltrlPreviewText.Text =
Utils.GetObjectContent(row["mental_health_preview_text"]);
99.                          if (ltrlPreviewText.Text != string.Empty)
```

Reflected XSS All Clients\Path 60:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=60
Status	New

Method GetKSData at line 158 of /MiluimHayalim-Site/Templates/Forms/VolunteeringByAge.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 158 of /MiluimHayalim-Site/Templates/Forms/VolunteeringByAge.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Forms/VolunteeringByAge.aspx.cs	/MiluimHayalim-Site/Templates/Forms/VolunteeringByAge.aspx.cs

Line	167	167
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Forms/VolunteeringByAge.aspx.cs
Method private void GetKSData()

```
....
167.                            ltrlPreviewText.Text =
Utils.GetObjectContent(row["vol_by_age_preview_text"]);
```

Reflected XSS All Clients\Path 61:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=61
Status	New

Method GetKSData at line 158 of /MiluimHayalim-Site/Templates/Forms/VolunteeringByAge.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 158 of /MiluimHayalim-Site/Templates/Forms/VolunteeringByAge.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Forms/VolunteeringByAge.aspx.cs	/MiluimHayalim-Site/Templates/Forms/VolunteeringByAge.aspx.cs
Line	167	168
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Forms/VolunteeringByAge.aspx.cs
Method private void GetKSData()

```
....
167.                            ltrlPreviewText.Text =
Utils.GetObjectContent(row["vol_by_age_preview_text"]);
168.                            if (ltrlPreviewText.Text != string.Empty)
```

Reflected XSS All Clients\Path 62:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=62
Status	New

Method GetKSData at line 158 of /MiluimHayalim-Site/Templates/Forms/VolunteeringByAge.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 158 of /MiluimHayalim-

Site/Templates/Forms/VolunteeringByAge.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Forms/VolunteeringByAge.aspx.cs	/MiluimHayalim-Site/Templates/Forms/VolunteeringByAge.aspx.cs
Line	189	189
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Forms/VolunteeringByAge.aspx.cs
Method private void GetKSData()

```
....
189.                            ltrlProfile.Text =
Utils.GetObjectContent(row["vol_by_age_profile_text"]);
```

Reflected XSS All Clients\Path 63:

Severity High
Result State To Verify
Online Results http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=63
Status New

Method GetKSData at line 158 of /MiluimHayalim-Site/Templates/Forms/VolunteeringByAge.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 158 of /MiluimHayalim-Site/Templates/Forms/VolunteeringByAge.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Forms/VolunteeringByAge.aspx.cs	/MiluimHayalim-Site/Templates/Forms/VolunteeringByAge.aspx.cs
Line	198	201
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Forms/VolunteeringByAge.aspx.cs
Method private void GetKSData()

```
....
198.                        string highlightsApproval =
Utils.GetObjectContent(row["vol_by_age_highlights_approval"]);
....
201.                        ltrlHighlightsApproval.Text =
highlightsApproval;
```

Reflected XSS All Clients\Path 64:

Severity High

Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=64
Status	New

Method DisplayDesc at line 98 of /MiluimHayalim-Site/Templates/Login/ForgotPassword.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method DisplayDesc at line 98 of /MiluimHayalim-Site/Templates/Login/ForgotPassword.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Login/ForgotPassword.aspx.cs	/MiluimHayalim-Site/Templates/Login/ForgotPassword.aspx.cs
Line	113	113
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Login/ForgotPassword.aspx.cs
Method private void DisplayDesc()

```
....
113.                                ltrlDesc.Text =
row["forgot_password_desc"].ToString();
```

Reflected XSS All Clients\Path 65:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=65
Status	New

Method DisplayDesc at line 98 of /MiluimHayalim-Site/Templates/Login/ForgotPassword.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method DisplayDesc at line 98 of /MiluimHayalim-Site/Templates/Login/ForgotPassword.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Login/ForgotPassword.aspx.cs	/MiluimHayalim-Site/Templates/Login/ForgotPassword.aspx.cs
Line	118	118
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Login/ForgotPassword.aspx.cs
Method private void DisplayDesc()

```
....
118.                                ltrlDesc.Text =
row["new_password_first_login_desc"].ToString();
```

Reflected XSS All Clients\Path 66:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=66
Status	New

Method DisplayDesc at line 96 of /MiluimHayalim-Site/Templates/Login/Login.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method DisplayDesc at line 96 of /MiluimHayalim-Site/Templates/Login/Login.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Login/Login.aspx.cs	/MiluimHayalim-Site/Templates/Login/Login.aspx.cs
Line	105	105
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Login/Login.aspx.cs
Method private void DisplayDesc()

```
....
105.                            ltrlDesc.Text =
row["login_preview_text"].ToString();
```

Reflected XSS All Clients\Path 67:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=67
Status	New

Method DisplayDesc at line 96 of /MiluimHayalim-Site/Templates/Login/Login.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method DisplayDesc at line 96 of /MiluimHayalim-Site/Templates/Login/Login.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Login/Login.aspx.cs	/MiluimHayalim-Site/Templates/Login/Login.aspx.cs
Line	105	106
Object	row	Text

Code Snippet

File Name /MiluimHayalim-Site/Templates/Login/Login.aspx.cs
Method private void DisplayDesc()

```
....
105.                            ltrlDesc.Text =
row["login_preview_text"].ToString();
106.                            ltrlFirstLoginText.Text =
row["first_login_text"].ToString();
```

Reflected XSS All Clients\Path 68:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=68
Status	New

Method DisplayDesc at line 96 of /MiluimHayalim-Site/Templates/Login/Login.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method DisplayDesc at line 96 of /MiluimHayalim-Site/Templates/Login/Login.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Login/Login.aspx.cs	/MiluimHayalim-Site/Templates/Login/Login.aspx.cs
Line	106	106
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Login/Login.aspx.cs
Method private void DisplayDesc()

```
....
106.                            ltrlFirstLoginText.Text =
row["first_login_text"].ToString();
```

Reflected XSS All Clients\Path 69:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=69
Status	New

Method DisplayDesc at line 97 of /MiluimHayalim-Site/Templates/Login/SMSVerify.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method DisplayDesc at line 97 of /MiluimHayalim-Site/Templates/Login/SMSVerify.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Login/SMSVerify.aspx.cs	/MiluimHayalim-Site/Templates/Login/SMSVerify.aspx.cs
Line	114	114

Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Login/SMSVerify.aspx.cs
Method private void DisplayDesc()

```
....
114.                            ltrlDesc.Text =
row["sms_verify_desc"].ToString();
```

Reflected XSS All Clients\Path 70:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=70
Status	New

Method GetUserNayad at line 697 of /MiluimHayalim-Site/App_Code/CacheHelper.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method DisplayDesc at line 97 of /MiluimHayalim-Site/Templates/Login/SMSVerify.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/App_Code/CacheHelper.cs	/MiluimHayalim-Site/Templates/Login/SMSVerify.aspx.cs
Line	709	117
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/App_Code/CacheHelper.cs
Method public static string GetUserNayad(int id)

```
....
709.                                nayad = string.Format("{0}{1}",
row["KIDOMET_TELEFON_NAYAD"], row["TELEFON_NAYAD"]);
```

File Name /MiluimHayalim-Site/Templates/Login/SMSVerify.aspx.cs
Method private void DisplayDesc()

```
....
117.                            ltrlNayad.Text = nayad;
```

Reflected XSS All Clients\Path 71:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=71
Status	New

Method GetUserNayad at line 697 of /MiluimHayalim-Site/App_Code/CacheHelper.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method DisplayDesc at line 97 of /MiluimHayalim-Site/Templates/Login/SMSVerify.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/App_Code/CacheHelper.cs	/MiluimHayalim-Site/Templates/Login/SMSVerify.aspx.cs
Line	709	117
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/App_Code/CacheHelper.cs
Method public static string GetUserNayad(int id)

```
....
709.                                nayad = string.Format("{0}{1}",
row["KIDOMET_TELEFON_NAYAD"], row["TELEFON_NAYAD"]);
```

File Name /MiluimHayalim-Site/Templates/Login/SMSVerify.aspx.cs
Method private void DisplayDesc()

```
....
117.                            ltrlNayad.Text = nayad;
```

Reflected XSS All Clients\Path 72:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=72
Status	New

Method GetKSData at line 54 of /MiluimHayalim-Site/Templates/Messages/Messages.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 54 of /MiluimHayalim-Site/Templates/Messages/Messages.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Messages/Messages.aspx.cs	/MiluimHayalim-Site/Templates/Messages/Messages.aspx.cs
Line	63	63
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Messages/Messages.aspx.cs
Method private void GetKSData()

```
....
63.                          ltrlMessagesPreviewText.Text =
Utils.GetObjectContent(row["messages_preview_text"]);
```

Reflected XSS All Clients\Path 73:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=73
Status	New

Method GetKSData at line 54 of /MiluimHayalim-Site/Templates/Messages/Messages.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 54 of /MiluimHayalim-Site/Templates/Messages/Messages.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Messages/Messages.aspx.cs	/MiluimHayalim-Site/Templates/Messages/Messages.aspx.cs
Line	63	64
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/Messages/Messages.aspx.cs
Method private void GetKSData()

```
....
63.                          ltrlMessagesPreviewText.Text =
Utils.GetObjectContent(row["messages_preview_text"]);
64.                          if (ltrlMessagesPreviewText.Text !=
string.Empty)
```

Reflected XSS All Clients\Path 74:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=74
Status	New

Method GetKSData at line 114 of /MiluimHayalim-Site/Templates/OrdersForms/ShamapChange.aspx.cs gets user input for the row element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method GetKSData at line 114 of /MiluimHayalim-Site/Templates/OrdersForms/ShamapChange.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/OrdersForms/ShamapChange.aspx.cs	/MiluimHayalim-Site/Templates/OrdersForms/ShamapChange.aspx.cs

Line	123	123
Object	row	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/OrdersForms/ShamapChange.aspx.cs
Method private void GetKSData()

```
....
123.                          ltrlPreviewText.Text =
Utils.GetObjectContent(row["shamap_preview_text"]);
```

Client DOM XSS

Description

Client DOM XSS\Path 3:

Severity	High
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=3
Status	New

Method P at line 284 of /MiluimHayalim-Site/Js/Forms/swfobject.js gets user input for the toString element. This element's value then flows through client-side code without being properly sanitized or validated and is eventually displayed to the user in u at line 369 of /MiluimHayalim-Site/Js/Forms/swfobject.js.This may enable a DOM XSS attack.

	Source	Destination
File	/MiluimHayalim-Site/Js/Forms/swfobject.js	/MiluimHayalim-Site/Js/Forms/swfobject.js
Line	309	401
Object	toString	outerHTML

Code Snippet
File Name /MiluimHayalim-Site/Js/Forms/swfobject.js
Method function P(aa, ab, X, Z) {

```
....
309.              ac = "MMredirectURL=" +
O.location.toString().replace(/&/g, "%26") + "&MMplayerType=" + ad +
"&MMdoctitle=" + j.title;
```

File Name /MiluimHayalim-Site/Js/Forms/swfobject.js
Method function u(ai, ag, Y) {

```
....
401.                  aa.outerHTML = '<object classid="clsid:D27CDB6E-
AE6D-11cf-96B8-444553540000"' + ah + ">" + af + "</object>";
```

Client Potential XSS

Description

Client Potential XSS\Path 10:

Severity	Medium
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=10
Status	New

Method FillInTatBakashot at line 183 of /MiluimHayalim-Site/Js/Forms/forms.js gets user input for the text element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method FillInTatBakashot at line 183 of /MiluimHayalim-Site/Js/Forms/forms.js. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Js/Forms/forms.js	/MiluimHayalim-Site/Js/Forms/forms.js
Line	186	188
Object	text	append

Code Snippet
File Name /MiluimHayalim-Site/Js/Forms/forms.js
Method function FillInTatBakashot(val) {

```
....
186.      var firstText = $(".sel-tat-bakasha").find('option')[0].text;
....
188.      $('.sel-tat-bakasha').append($('<option>', {
```

Client Potential XSS\Path 11:

Severity	Medium
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=11
Status	New

Method $ at line 204 of /MiluimHayalim-Site/Js/Forms/forms.js gets user input for the text element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method FillInTatBakashot at line 183 of /MiluimHayalim-Site/Js/Forms/forms.js. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Js/Forms/forms.js	/MiluimHayalim-Site/Js/Forms/forms.js
Line	206	204
Object	text	append

Code Snippet
File Name /MiluimHayalim-Site/Js/Forms/forms.js
Method $('.sel-tat-bakasha').append($('<option>', {

```
....
206.                          text: arrBakashot[i].items[j].text
```

File Name /MiluimHayalim-Site/Js/Forms/forms.js

Method function FillInTatBakashot(val) {

```
....
204.                        $('.sel-tat-bakasha').append($('<option>', {
```

Client Potential XSS\Path 12:

Severity	Medium
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=12
Status	New

Method $ at line 197 of /MiluimHayalim-Site/Js/Forms/forms.js gets user input for the text element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method FillInTatBakashot at line 183 of /MiluimHayalim-Site/Js/Forms/forms.js. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Js/Forms/forms.js	/MiluimHayalim-Site/Js/Forms/forms.js
Line	199	197
Object	text	append

Code Snippet
File Name /MiluimHayalim-Site/Js/Forms/forms.js
Method $('.sel-tat-bakasha').append($('<option>', {

```
....
199.                            text: arrBakashot[i].items[j].text,
```

File Name /MiluimHayalim-Site/Js/Forms/forms.js

Method function FillInTatBakashot(val) {

```
....
197.                        $('.sel-tat-bakasha').append($('<option>', {
```

Heap Inspection

Description

Heap Inspection\Path 17:

Severity	Medium
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=17
Status	New

	Source	Destination
File	/MiluimHayalim-Site/Templates/Login/ChangePassword.aspx.cs	/MiluimHayalim-Site/Templates/Login/ChangePassword.aspx.cs

Line	176	176
Object	pass	pass

Code Snippet
File Name /MiluimHayalim-Site/Templates/Login/ChangePassword.aspx.cs
Method protected void cvTxtPass1_ServerValidate(object source, ServerValidateEventArgs args)

```
....
176.                    string pass=args.Value.Trim();
```

Heap Inspection\Path 18:

Severity Medium
Result State To Verify
Online Results http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=18
Status New

	Source	Destination
File	/MiluimHayalim-Site/Templates/Login/Login.aspx.cs	/MiluimHayalim-Site/Templates/Login/Login.aspx.cs
Line	330	330
Object	pass	pass

Code Snippet
File Name /MiluimHayalim-Site/Templates/Login/Login.aspx.cs
Method protected void CvTxtPass_Validate(object source, ServerValidateEventArgs args)

```
....
330.                    string pass = args.Value.Trim();
```

Heap Inspection\Path 19:

Severity Medium
Result State To Verify
Online Results http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=19
Status New

	Source	Destination
File	/MiluimHayalim-Site/UserControls/PersonalSettings/ChangePassword.ascx.cs	/MiluimHayalim-Site/UserControls/PersonalSettings/ChangePassword.ascx.cs
Line	241	241
Object	pass	pass

Code Snippet
File Name /MiluimHayalim-Site/UserControls/PersonalSettings/ChangePassword.ascx.cs

Method	protected void cvTxtPass1_ServerValidate(object source, ServerValidateEventArgs args)

```
....
241.                    string pass = args.Value.Trim();
```

Data Filter Injection

Description

Data Filter Injection\Path 122:

Severity	Medium
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=122
Status	New

Method rptOrders_ItemDataBound at line 118 of /MiluimHayalim-Site/Templates/Lobby/Lobby.aspx.cs gets user input from the row element. This element's value then flows through the code without being properly sanitized or validated, and is eventually used in a query to the application server's cached data, in rptOrders_ItemDataBound at line 118 of /MiluimHayalim-Site/Templates/Lobby/Lobby.aspx.cs. This may enable a Data Filter Injection attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Lobby/Lobby.aspx.cs	/MiluimHayalim-Site/Templates/Lobby/Lobby.aspx.cs
Line	127	127
Object	row	Select

Code Snippet	
File Name	/MiluimHayalim-Site/Templates/Lobby/Lobby.aspx.cs
Method	protected void rptOrders_ItemDataBound(object sender, RepeaterItemEventArgs e)

```
....
127.                        DataRow[] dr =
dtSpecRequests.Select(string.Format("MISPAR_SEQ={0}",
row["MISPAR_SEQ"]),"SpecialRequestID desc");
```

Data Filter Injection\Path 123:

Severity	Medium
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=123
Status	New

Method rptOrders_ItemDataBound at line 118 of /MiluimHayalim-Site/Templates/Lobby/Lobby.aspx.cs gets user input from the row element. This element's value then flows through the code without being properly sanitized or validated, and is eventually used in a query to the application server's cached data, in rptOrders_ItemDataBound at line 118 of /MiluimHayalim-Site/Templates/Lobby/Lobby.aspx.cs. This may enable a Data Filter Injection attack.

	Source	Destination

File	/MiluimHayalim-Site/Templates/Lobby/Lobby.aspx.cs	/MiluimHayalim-Site/Templates/Lobby/Lobby.aspx.cs
Line	134	134
Object	row	Select

Code Snippet
File Name /MiluimHayalim-Site/Templates/Lobby/Lobby.aspx.cs
Method protected void rptOrders_ItemDataBound(object sender, RepeaterItemEventArgs e)

```
....
134.                            dr =
dtPniot.Select(string.Format("MISPAR_SEQ={0}", row["MISPAR_SEQ"]),
"ApplicationID");
```

Data Filter Injection\Path 124:

Severity	Medium
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=124
Status	New

Method rptOrders_ItemDataBound at line 118 of /MiluimHayalim-Site/Templates/Lobby/Lobby.aspx.cs gets user input from the row element. This element's value then flows through the code without being properly sanitized or validated, and is eventually used in a query to the application server's cached data, in rptOrders_ItemDataBound at line 118 of /MiluimHayalim-Site/Templates/Lobby/Lobby.aspx.cs. This may enable a Data Filter Injection attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/Lobby/Lobby.aspx.cs	/MiluimHayalim-Site/Templates/Lobby/Lobby.aspx.cs
Line	142	142
Object	row	Select

Code Snippet
File Name /MiluimHayalim-Site/Templates/Lobby/Lobby.aspx.cs
Method protected void rptOrders_ItemDataBound(object sender, RepeaterItemEventArgs e)

```
....
142.                            dr =
AllShamapPniot.Select(string.Format("MISPAR_SEQ={0}",
row["MISPAR_SEQ"]), "ApplicationID");
```

Client Cross Frame Scripting Attack

Description

Client Cross Frame Scripting Attack\Path 13:

Severity	Medium
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid

	=36&pathid=13
Status	New

	Source	Destination
File	/MiluimHayalim-Site/Ajax/AddChild.aspx	/MiluimHayalim-Site/Ajax/AddChild.aspx
Line	1	1
Object	CxJSNS_1140557385	CxJSNS_1140557385

Code Snippet
File Name /MiluimHayalim-Site/Ajax/AddChild.aspx
Method <%@ Page Language="C#" AutoEventWireup="true" MasterPageFile="~/MasterPages/MasterPageEmpty.master" CodeFile="AddChild.aspx.cs" Inherits="Scepia.Ajax.AddChild" %>

```
....
1.  <%@ Page Language="C#" AutoEventWireup="true"
MasterPageFile="~/MasterPages/MasterPageEmpty.master"
CodeFile="AddChild.aspx.cs" Inherits="Scepia.Ajax.AddChild" %>
```

Reflected XSS Specific Clients

Description

Reflected XSS Specific Clients\Path 125:

Severity	Medium
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=125
Status	New

Method SoldierPersonalDetails at line 265 of /MiluimHayalim-Site/Templates/PersonalDetails/PersonalDetails.aspx.cs gets user input for the r element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method SoldierPersonalDetails at line 265 of /MiluimHayalim-Site/Templates/PersonalDetails/PersonalDetails.aspx.cs. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	/MiluimHayalim-Site/Templates/PersonalDetails/PersonalDetails.aspx.cs	/MiluimHayalim-Site/Templates/PersonalDetails/PersonalDetails.aspx.cs
Line	329	329
Object	r	Text

Code Snippet
File Name /MiluimHayalim-Site/Templates/PersonalDetails/PersonalDetails.aspx.cs
Method private void SoldierPersonalDetails()

```
....
329.                                    hidMartialStatus.Text =
Utils.GetObjectContent(r["StatusMishpachtiID"]);
```

CookieLess Authentication

Description

CookieLess Authentication\Path 126:

Severity	Medium
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=126
Status	New

	Source	Destination
File	/MiluimHayalim-Site/web.config	/MiluimHayalim-Site/web.config
Line	49	49
Object	FORMS	FORMS

Code Snippet
File Name /MiluimHayalim-Site/web.config
Method <?xml version="1.0" encoding="UTF-8"?>

```
....
49.         <forms loginUrl="~/Login" timeout="20" defaultUrl="~/Lobby"
/>
```

RequireSSL

Description

RequireSSL\Path 129:

Severity	Medium
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=129
Status	New

	Source	Destination
File	/MiluimHayalim-Site/web.config	/MiluimHayalim-Site/web.config
Line	49	49
Object	FORMS	FORMS

Code Snippet
File Name /MiluimHayalim-Site/web.config
Method <?xml version="1.0" encoding="UTF-8"?>

```
....
49.         <forms loginUrl="~/Login" timeout="20" defaultUrl="~/Lobby"
/>
```

Client Heuristic Poor XSS Validation

Description

Client Heuristic Poor XSS Validation\Path 21:

Severity	Low
Result State	To Verify

Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=21
Status	New

	Source	Destination
File	/MiluimHayalim-Site/Js/Forms/swfobject.js	/MiluimHayalim-Site/Js/Forms/swfobject.js
Line	309	433
Object	toString	appendChild

Code Snippet
File Name /MiluimHayalim-Site/Js/Forms/swfobject.js
Method function P(aa, ab, X, Z) {

```
....
309.              ac = "MMredirectURL=" +
O.location.toString().replace(/&/g, "%26") + "&MMplayerType=" + ad +
"&MMdoctitle=" + j.title;
```

File Name /MiluimHayalim-Site/Js/Forms/swfobject.js
Method function e(Z, X, Y) {

```
....
433.          Z.appendChild(aa)
```

Client Heuristic Poor XSS Validation\Path 22:

Severity	Low
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=22
Status	New

	Source	Destination
File	/MiluimHayalim-Site/Js/Forms/swfobject.js	/MiluimHayalim-Site/Js/Forms/swfobject.js
Line	309	401
Object	toString	outerHTML

Code Snippet
File Name /MiluimHayalim-Site/Js/Forms/swfobject.js
Method function P(aa, ab, X, Z) {

```
....
309.              ac = "MMredirectURL=" +
O.location.toString().replace(/&/g, "%26") + "&MMplayerType=" + ad +
"&MMdoctitle=" + j.title;
```

File Name	/MiluimHayalim-Site/Js/Forms/swfobject.js
Method	function u(ai, ag, Y) {

```
....
401.                    aa.outerHTML = '<object classid="clsid:D27CDB6E-
AE6D-11cf-96B8-444553540000"' + ah + ">" + af + "</object>";
```

Unprotected Cookie

Description

Unprotected Cookie\Path 23:

Severity	Low
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=23
Status	New

	Source	Destination
File	/MiluimHayalim-Site/Templates/CellularReference/CellularReference.aspx	/MiluimHayalim-Site/Templates/CellularReference/CellularReference.aspx
Line	64	64
Object	setCookie	setCookie

Code Snippet

File Name	/MiluimHayalim-Site/Templates/CellularReference/CellularReference.aspx
Method	function setCookie(cookieName, cookieValue) {

```
....
64.          function setCookie(cookieName, cookieValue) {
```

Unprotected Cookie\Path 24:

Severity	Low
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=24
Status	New

	Source	Destination
File	/MiluimHayalim-Site/Templates/CellularReference/CellularReference.aspx	/MiluimHayalim-Site/Templates/CellularReference/CellularReference.aspx
Line	27	27
Object	setCookie	setCookie

Code Snippet

File Name	/MiluimHayalim-Site/Templates/CellularReference/CellularReference.aspx
Method	function cellularReferenceClick(url)

```
....
27.                 setCookie("IDF_cellular_dont_show", "");
```

Client DOM Open Redirect

Description

Client DOM Open Redirect\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=1
Status	New

	Source	Destination
File	/MiluimHayalim-Site/UserControls/CellularReference/CellularReference.ascx	/MiluimHayalim-Site/UserControls/CellularReference/CellularReference.ascx
Line	51	46
Object	replace	href

Code Snippet

File Name /MiluimHayalim-Site/UserControls/CellularReference/CellularReference.ascx

Method function getVirtualPath() {

```
....
51.         return window.location.href.replace(_virtual_url, '');
```

File Name /MiluimHayalim-Site/UserControls/CellularReference/CellularReference.ascx

Method function DoRedirect()

```
....
46.             window.location.href = url;
```

Client Insecure Randomness

Description

Client Insecure Randomness\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=5
Status	New

	Source	Destination
File	/MiluimHayalim-Site/Js/bootstrap.min.js	/MiluimHayalim-Site/Js/bootstrap.min.js
Line	930	930
Object	random	random

Code Snippet	
File Name	/MiluimHayalim-Site/Js/bootstrap.min.js
Method	c.prototype.getUID = function (a) {

```
....
930.          do a += ~~ (1e6 * Math.random());
```

DebugEnabled

Description

DebugEnabled\Path 127:

Severity	Low
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=127
Status	New

	Source	Destination
File	/MiluimHayalim-Site/web.config	/MiluimHayalim-Site/web.config
Line	47	47
Object	"true"	"true"

Code Snippet	
File Name	/MiluimHayalim-Site/web.config
Method	<?xml version="1.0" encoding="UTF-8"?>

```
....
47.      <compilation debug="true" targetFramework="4.5" />
```

NonUniqueFormName

Description

NonUniqueFormName\Path 128:

Severity	Low
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=128
Status	New

	Source	Destination
File	/MiluimHayalim-Site/web.config	/MiluimHayalim-Site/web.config
Line	49	49
Object	FORMS	FORMS

Code Snippet	
File Name	/MiluimHayalim-Site/web.config
Method	<?xml version="1.0" encoding="UTF-8"?>

```
....
49.         <forms loginUrl="~/Login" timeout="20" defaultUrl="~/Lobby"
/>
```

SlidingExpiration

Description

SlidingExpiration\Path 130:

Severity	Low
Result State	To Verify
Online Results	http://CXMANAGER/CxWebClient/ViewerMain.aspx?scanid=1000030&projectid=36&pathid=130
Status	New

	Source	Destination
File	/MiluimHayalim-Site/web.config	/MiluimHayalim-Site/web.config
Line	49	49
Object	FORMS	FORMS

Code Snippet
File Name /MiluimHayalim-Site/web.config
Method <?xml version="1.0" encoding="UTF-8"?>

```
....
49.         <forms loginUrl="~/Login" timeout="20" defaultUrl="~/Lobby"
/>
```

Client DOM XSS

Risk

What might happen

An attacker could use social engineering to cause a user to send the website engineered input, such as a URL with an engineered anchor, 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 password, request the user's credit card information, provide false information, or run malware.
From the victim's point of view, this is the original website, and the victim would blame the site for incurred damage.

Cause

How does it happen

The application web page includes data from user input (including the page URL). 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 user 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

1. 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
2. Fully encode all dynamic data before embedding it in the webpage.
3. 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
4. Consider using the ESAPI4JS encoding library.

Source Code Examples

C#
For dynamically creating URLs in JavaScript, use the OWASP ESAPI4JS library:

```
window.location = ESAPI4JS.encodeForURL(input);
```

For creating dynamic HTML in JavaScript, use the OWASP ESAPI4JS library:

```
window.location = ESAPI4JS.encodeForURL(input);
```

Java
For dynamically creating URLs in JavaScript, use the OWASP ESAPI4JS library:

```
window.location = ESAPI4JS.encodeForURL(input);
```

For creating dynamic HTML in JavaScript, use the OWASP ESAPI4JS library:

```
window.location = ESAPI4JS.encodeForURL(input);
```

Reflected XSS All Clients

Risk

What might happen

An attacker could use social engineering to cause a user to send the website engineered input, rewriting web pages and inserting malicious scripts.
The attacker can then pretend to be the original website, which would enable the attacker to steal the user's password, request the user's credit card information, provide false information, or run malware.
From the victim's point of view, this is the original website, and the victim would blame the site for incurred damage.

Cause

How does it happen

The application creates web pages that include data from previous user input. The user input is embedded directly in the page's HTML, 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 user 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

1. 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
2. Fully encode all dynamic data before embedding it in output.
3. Encoding should be context-sensitive. For example:
   - HTML encoding for HTML content
   - HTML Attribute encoding for data output to attribute values
   - JavaScript encoding for server-generated JavaScript
4. Consider using either the ESAPI encoding library, or the built-in platform functions. For earlier versions of ASP.NET, consider using the AntiXSS library.
5. In the Content-Type HTTP response header, explicitly define character encoding (charset) for the entire page.
6. Set the httpOnly flag on the session cookie, to prevent XSS exploits from stealing the cookie.

Source Code Examples

C#

The application uses the "Referer" field string to construct the HttpResponse

```
public class ReflectedXssAllClients
{
        public static void foo(HttpRequest Request, HttpResponse Response)
        {
                string Referer = Request.QueryString["Referer"];
```

```
                Response.BinaryWrite(Referer);
        }
}
```

The "Referer" field string is HTML encoded before use

```
public class ReflectedXssAllClientsFixed
{
        public static void foo(HttpRequest Request, HttpResponse Response,
AntiXss.AntiXssEncoder encoder)
        {
                string Referer = Request.QueryString["Referer"];
                Response.BinaryWrite(encoder.HtmlEncode(Referer, true));
        }
}
```

User input is written to a TextBox displayed on the screen enabling a user to inject a script

```
public class ReflectedXSSSpecificClients
{
        public void foo(TextBox tb)
        {
                string input = Console.ReadLine();
                tb.Text = input;
        }
}
```

The user input is Html encoded before being displayed on the screen

```
public class ReflectedXSSSpecificClientsFixed
{
        public void foo(TextBox tb, AntiXssEncoder encode)
        {
                string input = Console.ReadLine();
                tb.Text = encode.HtmlEncode(input);
        }
}
```

The application uses the "filename" field string from an HttpRequest construct an HttpResponse

```
public class UTF7XSS
{
        public void foo(HttpRequest Request, HttpResponse Response
        {
                Response.Charset("UTF-7");
                string filename = Request.QueryString["filename"];
                Response.BinaryWrite(AntiXss.HtmlEncode(filename));
        }
}
```

The "filename" string is converted to an int and using a switch case the new "filename" string is constructed

```
public class UTF7XSSFixed
{
        public static void foo(HttpRequest Request, HttpResponse Response)
        {
                Response.Charset("UTF-7");
                string filename = Request.QueryString["fileNum"];
```

```
                int fileNum = Convert.ToInt32(filename);

                switch(fileNum)
                {
                        case 1:
                                filename = "File1.txt";
                                break;
                        default:
                                filename = "File2.txt";
                                break;
                }

                Response.BinaryWrite(AntiXss.HtmlEncode(filename));
        }
}
```

Client Potential XSS

Risk

What might happen

An attacker could use social engineering to cause a user to send the website engineered input, rewriting web pages and inserting malicious scripts.
The attacker can then pretend to be the original website, which would enable the attacker to steal the user's password, request the user's credit card information, provide false information, or run malware.
From the victim's point of view, this is the original website, and the victim would blame the site for incurred damage.

Cause

How does it happen

The application creates web pages that include data from previous user input. The user input is embedded directly in the page's HTML, 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 user 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

1. 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
2. Fully encode all dynamic data before embedding it in output.
3. Encoding should be context-sensitive. For example:
   - HTML encoding for HTML content
   - HTML Attribute encoding for data output to attribute values
   - JavaScript encoding for server-generated JavaScript
4. Consider using either the ESAPI encoding library, or the built-in platform functions. For earlier versions of ASP.NET, consider using the AntiXSS library.
5. In the Content-Type HTTP response header, explicitly define character encoding (charset) for the entire page.
6. Set the httpOnly flag on the session cookie, to prevent XSS exploits from stealing the cookie.

Source Code Examples

Failure to Preserve Web Page Structure ('Cross-site Scripting')

Weakness ID: 79 *(Weakness Base)* **Status:** Usable

Description

Description Summary

The software does not sufficiently validate, filter, escape, and/or encode user-controllable input before it is placed in output that is used as a web page that is served to other users.

Extended Description

Cross-site scripting (XSS) vulnerabilities occur when:

1. Untrusted data enters a web application, typically from a web request.

2. The web application dynamically generates a web page that contains this untrusted data.

3. During page generation, the application does not prevent the data from containing content that is executable by a web browser, such as JavaScript, HTML tags, HTML attributes, mouse events, Flash, ActiveX, etc.

4. A victim visits the generated web page through a web browser, which contains malicious script that was injected using the untrusted data.

5. Since the script comes from a web page that was sent by the web server, the victim's web browser executes the malicious script in the context of the web server's domain.

6. This effectively violates the intention of the web browser's same-origin policy, which states that scripts in one domain should not be able to access resources or run code in a different domain.

There are three main kinds of XSS:

Type 1: Reflected XSS (or Non-Persistent)

The server reads data directly from the HTTP request and reflects it back in the HTTP response. Reflected XSS exploits occur when an attacker causes a victim to supply dangerous content to a vulnerable web application, which is then reflected back to the victim and executed by the web browser. The most common mechanism for delivering malicious content is to include it as a parameter in a URL that is posted publicly or e-mailed directly to the victim. URLs constructed in this manner constitute the core of many phishing schemes, whereby an attacker convinces a victim to visit a URL that refers to a vulnerable site. After the site reflects the attacker's content back to the victim, the content is executed by the victim's browser.

Type 2: Stored XSS (or Persistent)

The application stores dangerous data in a database, message forum, visitor log, or other trusted data store. At a later time, the dangerous data is subsequently read back into the application and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area that is displayed to either many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user. For example, the attacker might inject XSS into a log message, which might not be handled properly when an administrator views the logs.

Type 0: DOM-Based XSS

In DOM-based XSS, the client performs the injection of XSS into the page; in the other types, the server performs the injection. DOM-based XSS generally involves server-controlled, trusted script that is sent to the client, such as Javascript that performs

sanity checks on a form before the user submits it. If the server-supplied script processes user-supplied data and then injects it back into the web page (such as with dynamic HTML), then DOM-based XSS is possible.

Once the malicious script is injected, the attacker can perform a variety of malicious activities. The attacker could transfer private information, such as cookies that may include session information, from the victim's machine to the attacker. The attacker could send malicious requests to a web site on behalf of the victim, which could be especially dangerous to the site if the victim has administrator privileges to manage that site. Phishing attacks could be used to emulate trusted web sites and trick the victim into entering a password, allowing the attacker to compromise the victim's account on that web site. Finally, the script could exploit a vulnerability in the web browser itself possibly taking over the victim's machine, sometimes referred to as "drive-by hacking."

In many cases, the attack can be launched without the victim even being aware of it. Even with careful users, attackers frequently use a variety of methods to encode the malicious portion of the attack, such as URL encoding or Unicode, so the request looks less suspicious.

Alternate Terms

XSS

CSS: "CSS" was once used as the acronym for this problem, but this could cause confusion with "Cascading Style Sheets," so usage of this acronym has declined significantly.

Time of Introduction

- Architecture and Design
- Implementation

Applicable Platforms

Languages

Language-independent

Architectural Paradigms

Web-based: *(Often)*

Technology Classes

Web-Server: *(Often)*

Platform Notes

XSS flaws are very common in web applications since they require a great deal of developer discipline to avoid them.

Common Consequences

Scope	Effect
Confidentiality	The most common attack performed with cross-site scripting involves the disclosure of information stored in user cookies. Typically, a malicious user will craft a client-side script, which -- when parsed by a web browser -- performs some activity (such as sending all site cookies to a given E-mail address). This script will be loaded and run by each user visiting the web site. Since the site requesting to run the script has access to the cookies in question, the malicious script does also.
Access Control	In some circumstances it may be possible to run arbitrary code on a victim's computer when cross-site scripting is combined with other flaws.
Confidentiality Integrity Availability	The consequence of an XSS attack is the same regardless of whether it is stored or reflected. The difference is in how the payload arrives at the server. XSS can cause a variety of problems for the end user that range in severity from an annoyance to complete account compromise. Some cross-site scripting vulnerabilities can be exploited to manipulate or steal cookies, create requests that can be mistaken for those of a valid user, compromise confidential information, or execute malicious code on the end user systems for a variety of nefarious purposes. Other damaging attacks include the disclosure of end user files, installation of Trojan horse programs, redirecting the user to some other page or site, running "Active X" controls (under Microsoft Internet Explorer) from sites that a user perceives as trustworthy, and

	modifying presentation of content.

Likelihood of Exploit

High to Very High

Enabling Factors for Exploitation

Cross-site scripting attacks may occur anywhere that possibly malicious users are allowed to post unregulated material to a trusted web site for the consumption of other valid users, commonly on places such as bulletin-board web sites which provide web based mailing list-style functionality.

Stored XSS got its start with web sites that offered a "guestbook" to visitors. Attackers would include JavaScript in their guestbook entries, and all subsequent visitors to the guestbook page would execute the malicious code. As the examples demonstrate, XSS vulnerabilities are caused by code that includes unvalidated data in an HTTP response.

Detection Methods

Automated Static Analysis

Use automated static analysis tools that target this type of weakness. Many modern techniques use data flow analysis to minimize the number of false positives. This is not a perfect solution, since 100% accuracy and coverage are not feasible, especially when multiple components are involved.

Effectiveness: Moderate

Black Box

Use the XSS Cheat Sheet [REF-14] or automated test-generation tools to help launch a wide variety of attacks against your web application. The Cheat Sheet contains many subtle XSS variations that are specifically targeted against weak XSS defenses.

Effectiveness: Moderate

With Stored XSS, the indirection caused by the data store can make it more difficult to find the problem. The tester must first inject the XSS string into the data store, then find the appropriate application functionality in which the XSS string is sent to other users of the application. These are two distinct steps in which the activation of the XSS can take place minutes, hours, or days after the XSS was originally injected into the data store.

Demonstrative Examples

Example 1

This example covers a Reflected XSS (Type 1) scenario.

The following JSP code segment reads an employee ID, eid, from an HTTP request and displays it to the user.

(Bad Code)

Example Language: **JSP**

```
<% String eid = request.getParameter("eid"); %>
...
Employee ID: <%= eid %>
```

The following ASP.NET code segment reads an employee ID number from an HTTP request and displays it to the user.

(Bad Code)

Example Language: **ASP.NET**

```
...
protected System.Web.UI.WebControls.TextBox Login;
protected System.Web.UI.WebControls.Label EmployeeID;
...
EmployeeID.Text = Login.Text;
... (HTML follows) ...
<p><asp:label id="EmployeeID" runat="server" /></p>
...
```

The code in this example operates correctly if the Employee ID variable contains only standard alphanumeric text. If it has a value that includes meta-characters or source code, then the code will be executed by the web browser as it displays the HTTP response. Initially this might not appear to be much of a vulnerability. After all, why would someone enter a URL that causes malicious code to run on their own computer? The real danger is that an attacker will create the malicious URL, then use e-mail or social engineering tricks to lure victims into visiting a link to the URL. When victims click the link, they unwittingly reflect the malicious content through the vulnerable web

application back to their own computers.

Example 2

This example covers a Stored XSS (Type 2) scenario.

The following JSP code segment queries a database for an employee with a given ID and prints the corresponding employee's name.

(Bad Code)

Example Language: **JSP**

```
<%
...
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("select * from emp where id="+eid);
if (rs != null) {
rs.next();
String name = rs.getString("name");
%>

Employee Name: <%= name %>
```

The following ASP.NET code segment queries a database for an employee with a given employee ID and prints the name corresponding with the ID.

(Bad Code)

Example Language: **ASP.NET**

```
protected System.Web.UI.WebControls.Label EmployeeName;
...
string query = "select * from emp where id=" + eid;
sda = new SqlDataAdapter(query, conn);
sda.Fill(dt);
string name = dt.Rows[0]["Name"];
...
EmployeeName.Text = name;
```

This code can appear less dangerous because the value of name is read from a database, whose contents are apparently managed by the application. However, if the value of name originates from user-supplied data, then the database can be a conduit for malicious content. Without proper input validation on all data stored in the database, an attacker can execute malicious commands in the user's web browser.

Observed Examples

Reference	Description
CVE-2008-5080	Chain: protection mechanism failure allows XSS
CVE-2006-4308	Chain: only checks "javascript:" tag
CVE-2007-5727	Chain: only removes SCRIPT tags, enabling XSS
CVE-2008-5770	Reflected XSS using the PATH INFO in a URL
CVE-2008-4730	Reflected XSS not properly handled when generating an error message
CVE-2008-5734	Reflected XSS sent through email message.
CVE-2008-0971	Stored XSS in a security product.
CVE-2008-5249	Stored XSS using a wiki page.
CVE-2006-3568	Stored XSS in a guestbook application.
CVE-2006-3211	Stored XSS in a guestbook application using a javascript: URI in a bbcode img tag.
CVE-2006-3295	Chain: library file is not protected against a direct request (CWE-425), leading to reflected XSS.

Potential Mitigations

Phase: Architecture and Design

Strategy: Libraries or Frameworks

Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket.

Phases: Implementation; Architecture and Design

Understand the context in which your data will be used and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies.

For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters.

Parts of the same output document may require different encodings, which will vary depending on whether the output is in the:

- HTML body
- Element attributes (such as src="XYZ")
- URIs
- JavaScript sections
- Cascading Style Sheets and style property

etc. Note that HTML Entity Encoding is only appropriate for the HTML body.

Consult the XSS Prevention Cheat Sheet [REF-16] for more details on the types of encoding and escaping that are needed.

Phase: Architecture and Design

For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server.

Phase: Implementation

Use and specify a strong character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser may choose a different encoding by guessing which encoding is actually being used by the web page. This can open you up to subtle XSS attacks related to that encoding. See CWE-116 for more mitigations related to encoding/escaping.

Phase: Implementation

With Struts, you should write all data from form beans with the bean's filter attribute set to true.

Phase: Implementation

To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set.

Phase: Implementation

Strategy: Input Validation

Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a blacklist). However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright.

When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue."

When dynamically constructing web pages, use stringent whitelists that limit the character set based on the expected value of the parameter in the request. All input should be validated and cleansed, not just parameters that the user is supposed to specify, but all data in the request, including hidden fields, cookies, headers, the URL itself, and so forth. A common mistake that leads to continuing XSS vulnerabilities is to validate only fields that are expected to be redisplayed by the site. It is common to see data from the request that is reflected by the application server or the application that the development team did not anticipate. Also, a field that is not currently reflected may be used by a future developer. Therefore, validating ALL parts of the HTTP request is recommended.

Note that proper output encoding, escaping, and quoting is the most effective solution for preventing XSS, although input validation may provide some defense-in-depth. This is because it effectively limits what will appear in output. Input validation will not always prevent XSS, especially if you are required to support free-form text fields that could contain arbitrary characters. For example, in a chat application, the heart emoticon ("<3") would likely pass the validation step, since it is commonly used. However, it cannot be directly inserted into the web page because it contains the "<" character, which would need to be escaped or otherwise handled. In this case, stripping the "<" might reduce the risk of XSS, but it would produce incorrect behavior because the emoticon would not be recorded. This might seem to be a minor inconvenience, but it would be more important in a mathematical forum that wants to represent inequalities.

Even if you make a mistake in your validation (such as forgetting one out of 100 input fields), appropriate encoding is still likely to protect you from injection-based attacks. As long as it is not done in isolation, input validation is still a useful technique, since it

may significantly reduce your attack surface, allow you to detect some attacks, and provide other security benefits that proper encoding does not address.

Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere.

Phase: Operation

Use an application firewall that can detect attacks against this weakness. This might not catch all attacks, and it might require some effort for customization. However, it can be beneficial in cases in which the code cannot be fixed (because it is controlled by a third party), as an emergency prevention measure while more comprehensive software assurance measures are applied, or to provide defense in depth.

Background Details

Same Origin Policy

The same origin policy states that browsers should limit the resources accessible to scripts running on a given web site , or "origin", to the resources associated with that web site on the client-side, and not the client-side resources of any other sites or "origins". The goal is to prevent one site from being able to modify or read the contents of an unrelated site. Since the World Wide Web involves interactions between many sites, this policy is important for browsers to enforce.

Domain

The Domain of a website when referring to XSS is roughly equivalent to the resources associated with that website on the client-side of the connection. That is, the domain can be thought of as all resources the browser is storing for the user's interactions with this particular site.

Weakness Ordinalities

Ordinality	Description
Resultant	*(where the weakness is typically related to the presence of some other weaknesses)*

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to	Named Chain(s) this relationship pertains to
ChildOf	Weakness Class	20	Improper Input Validation	**Seven Pernicious Kingdoms (primary)700**	
ChildOf	Weakness Class	74	Failure to Sanitize Data into a Different Plane ('Injection')	**Development Concepts (primary)699 Research Concepts (primary)1000**	
ChildOf	Category	442	Web Problems	Development Concepts699	
ChildOf	Category	712	OWASP Top Ten 2007 Category A1 - Cross Site Scripting (XSS)	**Weaknesses in OWASP Top Ten (2007) (primary)629**	
ChildOf	Category	722	OWASP Top Ten 2004 Category A1 - Unvalidated Input	Weaknesses in OWASP Top Ten (2004)711	
ChildOf	Category	725	OWASP Top Ten 2004 Category A4 - Cross-Site Scripting (XSS) Flaws	**Weaknesses in OWASP Top Ten (2004) (primary)711**	
ChildOf	Category	751	2009 Top 25 - Insecure Interaction Between Components	**Weaknesses in the 2009 CWE/SANS Top 25 Most Dangerous Programming Errors (primary)750**	
ChildOf	Category	801	2010 Top 25 - Insecure Interaction Between Components	**Weaknesses in the 2010 CWE/SANS Top 25 Most Dangerous Programming Errors (primary)800**	
CanPrecede	Weakness Base	494	Download of Code Without Integrity Check	Research Concepts1000	
PeerOf	Compound Element: Composite	352	Cross-Site Request Forgery (CSRF)	Research Concepts1000	
ParentOf	Weakness Variant	80	Improper Sanitization of Script-Related HTML Tags in a Web Page (Basic XSS)	**Development Concepts (primary)699 Research Concepts (primary)1000**	
ParentOf	Weakness Variant	81	Improper Sanitization of Script in an Error Message Web Page	**Development Concepts (primary)699 Research Concepts (primary)1000**	
ParentOf	Weakness Variant	83	Improper Neutralization of Script in Attributes in a Web Page	**Development Concepts (primary)699 Research Concepts (primary)1000**	
ParentOf	Weakness Variant	84	Failure to Resolve Encoded URI Schemes in a Web Page	**Development Concepts (primary)699 Research Concepts (primary)1000**	
ParentOf	Weakness Variant	85	Doubled Character XSS Manipulations	**Development Concepts (primary)699 Research Concepts (primary)1000**	
ParentOf	Weakness Variant	86	Improper Neutralization of Invalid Characters in Identifiers in Web Pages	**Development Concepts (primary)699 Research Concepts (primary)1000**	
ParentOf	Weakness Variant	87	Failure to Sanitize Alternate XSS Syntax	**Development Concepts (primary)699 Research Concepts (primary)1000**	
MemberOf	View	635	Weaknesses Used by NVD	**Weaknesses Used by NVD**	

				(primary)635	
CanFollow	Weakness Base	113	Failure to Sanitize CRLF Sequences in HTTP Headers ('HTTP Response Splitting')	Research Concepts1000	
CanFollow	Weakness Base	184	Incomplete Blacklist	Research Concepts1000	Incomplete Blacklist to Cross-Site Scripting692

f Causal Nature

Explicit

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			Cross-site scripting (XSS)
7 Pernicious Kingdoms			Cross-site Scripting
CLASP			Cross-site scripting
OWASP Top Ten 2007	A1	Exact	Cross Site Scripting (XSS)
OWASP Top Ten 2004	A1	CWE More Specific	Unvalidated Input
OWASP Top Ten 2004	A4	Exact	Cross-Site Scripting (XSS) Flaws
WASC	8		Cross-site Scripting

Related Attack Patterns

CAPEC-ID	Attack Pattern Name	*(CAPEC Version: 1.5)*
232	Exploitation of Privilege/Trust	
85	Client Network Footprinting (using AJAX/XSS)	
86	Embedding Script (XSS) in HTTP Headers	
32	Embedding Scripts in HTTP Query Strings	
18	Embedding Scripts in Nonscript Elements	
19	Embedding Scripts within Scripts	
63	Simple Script Injection	
91	XSS in IMG Tags	
106	Cross Site Scripting through Log Files	
198	Cross-Site Scripting in Error Pages	
199	Cross-Site Scripting Using Alternate Syntax	
209	Cross-Site Scripting Using MIME Type Mismatch	
243	Cross-Site Scripting in Attributes	
244	Cross-Site Scripting via Encoded URI Schemes	
245	Cross-Site Scripting Using Doubled Characters, e.g. %3C%3Cscript	
246	Cross-Site Scripting Using Flash	
247	Cross-Site Scripting with Masking through Invalid Characters in Identifiers	

References

[REF-15] Jeremiah Grossman, Robert "RSnake" Hansen, Petko "pdp" D. Petkov, Anton Rager and Seth Fogie. "XSS Attacks". Syngress. 2007.

[REF-17] Michael Howard, David LeBlanc and John Viega. "24 Deadly Sins of Software Security". "Sin 2: Web-Server Related Vulnerabilities (XSS, XSRF, and Response Splitting)." Page 31. McGraw-Hill. 2010.

[REF-17] Michael Howard, David LeBlanc and John Viega. "24 Deadly Sins of Software Security". "Sin 3: Web-Client Related Vulnerabilities (XSS)." Page 63. McGraw-Hill. 2010.

"Cross-site scripting". Wikipedia. 2008-08-26. <http://en.wikipedia.org/wiki/Cross-site_scripting>.

[REF-11] M. Howard and D. LeBlanc. "Writing Secure Code". Chapter 13, "Web-Specific Input Issues" Page 413. 2nd Edition.

Microsoft. 2002.

[REF-14] RSnake. "XSS (Cross Site Scripting) Cheat Sheet". <http://ha.ckers.org/xss.html>.

Microsoft. "Mitigating Cross-site Scripting With HTTP-only Cookies". <http://msdn.microsoft.com/en-us/library/ms533046.aspx>.

Mark Curphey, Microsoft. "Anti-XSS 3.0 Beta and CAT.NET Community Technology Preview now Live!". <http://blogs.msdn.com/cisg/archive/2008/12/15/anti-xss-3-0-beta-and-cat-net-community-technology-preview-now-live.aspx>.

"OWASP Enterprise Security API (ESAPI) Project". <http://www.owasp.org/index.php/ESAPI>.

Ivan Ristic. "XSS Defense HOWTO". <http://blog.modsecurity.org/2008/07/do-you-know-how.html>.

OWASP. "Web Application Firewall". <http://www.owasp.org/index.php/Web_Application_Firewall>.

Web Application Security Consortium. "Web Application Firewall Evaluation Criteria". <http://www.webappsec.org/projects/wafec/v1/wasc-wafec-v1.0.html>.

RSnake. "Firefox Implements httpOnly And is Vulnerable to XMLHTTPRequest". 2007-07-19.

"XMLHttpRequest allows reading HTTPOnly cookies". Mozilla. <https://bugzilla.mozilla.org/show_bug.cgi?id=380418>.

"Apache Wicket". <http://wicket.apache.org/>.

[REF-16] OWASP. "XSS (Cross Site Scripting) Prevention Cheat Sheet". <http://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet>.

Content History

Submissions

Submission Date	Submitter	Organization	Source
	PLOVER		Externally Mined

Modifications

Modification Date	Modifier	Organization	Source
2008-07-01	Eric Dalci updated Time of Introduction	Cigital	External
2008-08-15	 Suggested OWASP Top Ten 2004 mapping	Veracode	External
2008-09-08	CWE Content Team updated Alternate Terms, Applicable Platforms, Background Details, Common Consequences, Description, Relationships, Other Notes, References, Taxonomy Mappings, Weakness Ordinalities	MITRE	Internal
2009-01-12	CWE Content Team updated Alternate Terms, Applicable Platforms, Background Details, Common Consequences, Demonstrative Examples, Description, Detection Factors, Enabling Factors for Exploitation, Name, Observed Examples, Other Notes, Potential Mitigations, References, Relationships	MITRE	Internal
2009-03-10	CWE Content Team updated Potential Mitigations	MITRE	Internal
2009-05-27	CWE Content Team updated Name	MITRE	Internal
2009-07-27	CWE Content Team updated Description	MITRE	Internal
2009-10-29	CWE Content Team updated Observed Examples, Relationships	MITRE	Internal
2009-12-28	CWE Content Team updated Demonstrative Examples, Description, Detection Factors, Enabling Factors for Exploitation, Observed Examples	MITRE	Internal
2010-02-16	CWE Content Team updated Applicable Platforms, Detection Factors, Potential Mitigations, References, Relationships, Taxonomy Mappings	MITRE	Internal
2010-04-05	CWE Content Team updated Description, Potential Mitigations, Related Attack Patterns	MITRE	Internal

Previous Entry Names

Change Date	Previous Entry Name
2008-04-11	Cross-site Scripting (XSS)
2009-01-12	Failure to Sanitize Directives in a Web Page (aka 'Cross-site scripting' (XSS))
2009-05-27	Failure to Preserve Web Page Structure (aka 'Cross-site Scripting')

BACK TO TOP

Failure to Clear Heap Memory Before Release ('Heap Inspection')

Weakness ID: 244 *(Weakness Variant)* **Status:** Draft

Description

Description Summary

Using realloc() to resize buffers that store sensitive information can leave the sensitive information exposed to attack, because it is not removed from memory.

Extended Description

When sensitive data such as a password or an encryption key is not removed from memory, it could be exposed to an attacker using a "heap inspection" attack that reads the sensitive data using memory dumps or other methods. The realloc() function is commonly used to increase the size of a block of allocated memory. This operation often requires copying the contents of the old memory block into a new and larger block. This operation leaves the contents of the original block intact but inaccessible to the program, preventing the program from being able to scrub sensitive data from memory. If an attacker can later examine the contents of a memory dump, the sensitive data could be exposed.

Time of Introduction

- Implementation

Applicable Platforms

Languages

C

C++

Common Consequences

Scope	Effect
Confidentiality	Be careful using vfork() and fork() in security sensitive code. The process state will not be cleaned up and will contain traces of data from past use.

Demonstrative Examples

Example 1

The following code calls realloc() on a buffer containing sensitive data:

(Bad Code)

Example Language: **C**

```
cleartext_buffer = get_secret();...
cleartext_buffer = realloc(cleartext_buffer, 1024);
...
scrub_memory(cleartext_buffer, 1024);
```

There is an attempt to scrub the sensitive data from memory, but realloc() is used, so a copy of the data can still be exposed in the memory originally allocated for cleartext_buffer.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Weakness Base	226	Sensitive Information Uncleared Before Release	**Research Concepts (primary)1000**
ChildOf	Weakness Class	227	Failure to Fulfill API Contract ('API Abuse')	**Development Concepts (primary)699** **Seven Pernicious Kingdoms (primary)700**
ChildOf	Category	633	Weaknesses that Affect Memory	**Resource-specific Weaknesses (primary)631**
ChildOf	Category	742	CERT C Secure Coding Section 08 - Memory Management (MEM)	**Weaknesses Addressed by the CERT C Secure Coding Standard (primary)734**
CanPrecede	Weakness Class	669	Incorrect Resource Transfer Between Spheres	Research Concepts1000
MemberOf	View	630	Weaknesses Examined by SAMATE	**Weaknesses Examined by SAMATE (primary)630**

Affected Resources

- Memory

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
7 Pernicious Kingdoms			Heap Inspection
CERT C Secure Coding	MEM03-C		Clear sensitive information stored in reusable resources returned for reuse

White Box Definitions

A weakness where code path has:

1. start statement that stores information in a buffer
2. end statement that resize the buffer and
3. path does not contain statement that performs cleaning of the buffer

Content History

Submissions

Submission Date	Submitter	Organization	Source
	7 Pernicious Kingdoms		Externally Mined

Modifications

Modification Date	Modifier	Organization	Source
2008-08-01		KDM Analytics	External
	added/updated white box definitions		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Applicable Platforms, Name, Relationships, Other Notes, Taxonomy Mappings		
2008-10-14	CWE Content Team	MITRE	Internal
	updated Relationships		
2008-11-24	CWE Content Team	MITRE	Internal
	updated Relationships, Taxonomy Mappings		
2009-05-27	CWE Content Team	MITRE	Internal
	updated Demonstrative Examples, Name		
2009-10-29	CWE Content Team	MITRE	Internal
	updated Common Consequences, Description, Other Notes		

Previous Entry Names

Change Date	Previous Entry Name
2008-04-11	Heap Inspection
2008-09-09	Failure to Clear Heap Memory Before Release
2009-05-27	Failure to Clear Heap Memory Before Release (aka 'Heap Inspection')

BACK TO TOP

Data Filter Injection

Risk

What might happen

An attacker could directly access all of the system's data.
Using simple tools and text editing, the attacker would be able to steal any sensitive information stored in the server cache (such as personal user details or credit cards),
and possibly change or erase existing data that could be subsequently used for other users or relied upon for security decisions.
The application stores temporary data in its cache, and queries this data. The application creates the query by simply concatenating strings including the user's input.
Since the user input is neither checked for data type validity nor subsequently sanitized, the input could contain commands that would be interpreted as such.

General Recommendations

How to avoid it

1. 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
2. Instead of concatenating strings:
   a. Use secure database components such as stored procedures, parameterized queries, and object bindings (for commands and parameters).
   b. An even better solution is to use an ORM library, such as EntityFramework, Hibernate, or iBatis.
3. Restrict access to database objects and functionality, according to the Principle of Least Privilege.
4. If possible, avoid making security decisions based on cached data, especially data shared between users.

Source Code Examples

C#

The application creates a query using ViewState with cached data that might contain a user injected script

```
public class DataFilterInjection
{
        public void foo(DataView dv)
        {
                string input = ViewState["strFilterFiles"].ToString();
                dv.RowFilter = "FileName like \'%" + input + "%\'";
        }
}
```

The string obtained from the cached data is examined for malicious characters

```
public class DataFilterInjectionFixed
{
        public void foo(DataView dv)
        {
                string input = ViewState["strFilterFiles"].ToString();
                string filtered = input.Replace("'","");
                dv.RowFilter = "FileName like \'%" + filtered + "%\'";
        }
```

```
}
```

Reflected XSS Specific Clients

Risk

What might happen

An attacker could use social engineering to cause a user to send the website engineered input, rewriting web pages and inserting malicious scripts.

The attacker can then pretend to be the original website, which would enable the attacker to steal the user's password, request the user's credit card information, provide false information, or run malware.

From the victim's point of view, this is the original website, and the victim would blame the site for incurred damage.

Cause

How does it happen

The application creates web pages that include data from previous user input. The user input is embedded directly in the page's HTML, 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 user 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

1. 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

2. Fully encode all dynamic data before embedding it in output.
3. Encoding should be context-sensitive. For example:
   - HTML encoding for HTML content
   - HTML Attribute encoding for data output to attribute values
   - JavaScript encoding for server-generated JavaScript
4. Consider using either the ESAPI encoding library, or the built-in platform functions. For earlier versions of ASP.NET, consider using the AntiXSS library.
5. In the Content-Type HTTP response header, explicitly define character encoding (charset) for the entire page.
6. Set the httpOnly flag on the session cookie, to prevent XSS exploits from stealing the cookie.

Source Code Examples

Cookieless Authentication Enabled

Weakness ID: 10704 *(Weakness Base)* **Status:** Draft

Description

Description Summary

When cookieless authentication in the application is enabled, it can cause to the session being hijacked.

Extended Description

When the authentication state of the application is set to cookieless, the authentication token of the application appeares in the page URL and not in the cookie.
In this case, an attacker can take over the session by impersonating legitimate user.

Time of Introduction

- Implementation
- Operation

Applicable Platforms

Languages

ASP.NET

Technology Classes

Web-Server

Demonstrative Examples

Example :

The following code in ASP.NET describes a vulnerable configuration of cookieless authentication in a web.config file:

(Bad Code)

Example Language: ASP.NET

```
<configuration>
<system.web>
<authentication mode="Forms">
<forms cookieless="UseUri">
```

Potential Mitigations

Disable cookieless authentication by setting the "cookieless" attirbute to "UseCookies" .
Store the authentication tokens in cookies.

Sensitive Cookie in HTTPS Session Without 'Secure' Attribute

Weakness ID: 614 *(Weakness Variant)* **Status:** Draft

Description

Description Summary

The Secure attribute for sensitive cookies in HTTPS sessions is not set, which could cause the user agent to send those cookies in plaintext over an HTTP session.

Time of Introduction

- Implementation

Demonstrative Examples

Example 1

The snippet of code below, taken from a servlet doPost() method, sets an accountID cookie (sensitive) without calling setSecure(true).

(Bad Code)

Example Language: **Java**

```
Cookie c = new Cookie(ACCOUNT_ID, acctID);
response.addCookie(c);
```

Observed Examples

Reference	Description
CVE-2004-0462	A product does not set the Secure attribute for sensitive cookies in HTTPS sessions, which could cause the user agent to send those cookies in plaintext over an HTTP session with the product.
CVE-2008-3663	A product does not set the secure flag for the session cookie in an https session, which can cause the cookie to be sent in http requests and make it easier for remote attackers to capture this cookie.
CVE-2008-3662	A product does not set the secure flag for the session cookie in an https session, which can cause the cookie to be sent in http requests and make it easier for remote attackers to capture this cookie.
CVE-2008-0128	A product does not set the secure flag for a cookie in an https session, which can cause the cookie to be sent in http requests and make it easier for remote attackers to capture this cookie.

Potential Mitigations

Always set the secure attribute when the cookie should sent via HTTPS only.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Weakness Base	311	Missing Encryption of Sensitive Data	**Development Concepts (primary)699** **Research Concepts (primary)1000**

Related Attack Patterns

CAPEC-ID	Attack Pattern Name	*(CAPEC Version: 1.5)*
102	Session Sidejacking	

Content History

Submissions			
Submission Date	**Submitter**	**Organization**	**Source**
	Anonymous Tool Vendor (under NDA)		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 Potential Mitigations, Time of Introduction		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Relationships, Taxonomy Mappings		
2008-10-14	CWE Content Team	MITRE	Internal
	updated Observed Examples		
2009-03-10	CWE Content Team	MITRE	Internal
	updated Name		
2009-05-27	CWE Content Team	MITRE	Internal
	updated Related Attack Patterns		
Previous Entry Names			

Change Date	Previous Entry Name
2008-04-11	Unset Secure Attribute for Sensitive Cookies in HTTPS Session

BACK TO TOP

Client DOM Open Redirect

Risk

What might happen

An attacker could use social engineering to get a victim to click a link to the application, so that the user will be immediately redirected to another, arbitrary site.

Users may think that they are still in the original application site. The second site may be offensive, contain malware, or, most commonly, be used for phishing.

Cause

How does it happen

The application redirects the user's browser to a URL provided in a user request, without warning users that they are being redirected outside the site.

An attacker could use social engineering to get a victim to click a link to the application with a parameter defining another site to which the application will redirect the user's browser, and the user may not be aware of the redirection.

General Recommendations

How to avoid it

1. Ideally, do not allow arbitrary URLs for redirection. Instead, create a server-side mapping from user-provided parameter values to legitimate URLs.
2. If it is necessary to allow arbitrary URLs:
   - For URLs inside the application site, first filter and encode the user-provided parameter, and then use it as a relative URL by prefixing it with the application site domain.
   - For URLs outside the application (if necessary), use an intermediate disclaimer page to provide users with a clear warning that they are leaving your site

---

Source Code Examples

C#

Avoid redirecting to arbitrary URLs, instead map the parameter to a list of static URLs.

```
Response.Redirect(getUrlById(targetUrlId));
```

Java

Avoid redirecting to arbitrary URLs, instead map the parameter to a list of static URLs.

```
Response.Redirect(getUrlById(targetUrlId));
```

Use of Insufficiently Random Values

Weakness ID: 330 *(Weakness Class)* **Status:** Usable

Description

Description Summary

The software may use insufficiently random numbers or values in a security context that depends on unpredictable numbers.

Extended Description

When software generates predictable values in a context requiring unpredictability, it may be possible for an attacker to guess the next value that will be generated, and use this guess to impersonate another user or access sensitive information.

Time of Introduction

- Architecture and Design
- Implementation

Applicable Platforms

Languages

Language-independent

Common Consequences

Scope	Effect
Confidentiality	When a protection mechanism relies on random values to restrict access to a sensitive resource, such as a session ID or a seed for generating a cryptographic key, then the resource being protected could be accessed by guessing the ID or key.
Confidentiality Availability	If software relies on unique, unguessable IDs to identify a resource, an attacker might be able to guess an ID for a resource that is owned by another user. The attacker could then read the resource, or pre-create a resource with the same ID to prevent the legitimate program from properly sending the resource to the intended user. For example, a product might maintain session information in a file whose name is based on a username. An attacker could pre-create this file for a victim user, then set the permissions so that the application cannot generate the session for the victim, preventing the victim from using the application.
Integrity	When an authorization or authentication mechanism relies on random values to restrict access to restricted functionality, such as a session ID or a seed for generating a cryptographic key, then an attacker may access the restricted functionality by guessing the ID or key.

Likelihood of Exploit

Medium to High

Demonstrative Examples

Example 1

The following code uses a statistical PRNG to create a URL for a receipt that remains active for some period of time after a purchase.

(Bad Code)

Example Language: **Java**

```
String GenerateReceiptURL(String baseUrl) {
Random ranGen = new Random();
ranGen.setSeed((new Date()).getTime());
return(baseUrl + ranGen.nextInt(400000000) + ".html");
}
```

This code uses the Random.nextInt() function to generate "unique" identifiers for the receipt pages it generates. Because Random.nextInt() is a statistical PRNG, it is easy for an attacker to guess the strings it generates. Although the underlying design of the receipt system is also faulty, it would be more secure if it used a random number generator that did not produce predictable receipt identifiers, such as a cryptographic PRNG.

Observed Examples

Reference	Description
CVE-2009-3278	Crypto product uses rand() library function to generate a recovery key, making it easier to conduct brute force attacks.
CVE-2009-3238	Random number generator can repeatedly generate the same value.
CVE-2009-2367	Web application generates predictable session IDs, allowing session hijacking.
CVE-2009-2158	Password recovery utility generates a relatively small number of random passwords, simplifying brute force attacks.
CVE-2009-0255	Cryptographic key created with an insufficiently random seed.
CVE-2009-0255	Cryptographic key created with a seed based on the system time.
CVE-2008-5162	Kernel function does not have a good entropy source just after boot.
CVE-2008-4905	Blogging software uses a hard-coded salt when calculating a password hash.
CVE-2008-4929	Bulletin board application uses insufficiently random names for uploaded files, allowing other users to access private files.
CVE-2008-3612	Handheld device uses predictable TCP sequence numbers, allowing spoofing or hijacking of TCP connections.
CVE-2008-2433	Web management console generates session IDs based on the login time, making it easier to conduct session hijacking.
CVE-2008-0166	SSL library uses a weak random number generator that only generates 65,536 unique keys.
CVE-2008-2108	Chain: insufficient precision causes extra zero bits to be assigned, reducing entropy for an API function that generates random numbers.
CVE-2008-2020	CAPTCHA implementation does not produce enough different images, allowing bypass using a database of all possible checksums.
CVE-2008-0087	DNS client uses predictable DNS transaction IDs, allowing DNS spoofing.
CVE-2008-0141	Application generates passwords that are based on the time of day.

Potential Mitigations

Phase: Architecture and Design

Use a well-vetted algorithm that is currently considered to be strong by experts in the field, and select well-tested implementations with adequate length seeds.

In general, if a pseudo-random number generator is not advertised as being cryptographically secure, then it is probably a statistical PRNG and should not be used in security-sensitive contexts.

Pseudo-random number generators can produce predictable numbers if the generator is known and the seed can be guessed. A 256-bit seed is a good starting point for producing a "random enough" number.

Phase: Implementation

Consider a PRNG that re-seeds itself as needed from high quality pseudo-random output sources, such as hardware devices.

Phase: Testing

Use automated static analysis tools that target this type of weakness. Many modern techniques use data flow analysis to minimize the number of false positives. This is not a perfect solution, since 100% accuracy and coverage are not feasible.

Phase: Testing

Perform FIPS 140-2 tests on data to catch obvious entropy problems.

Phase: Testing

Use tools and techniques that require manual (human) analysis, such as penetration testing, threat modeling, and interactive tools that allow the tester to record and modify an active session. These may be more effective than strictly automated techniques. This is especially the case with weaknesses that are related to design and business rules.

Phase: Testing

Use monitoring tools that examine the software's process as it interacts with the operating system and the network. This technique is useful in cases when source code is unavailable, if the software was not developed by you, or if you want to verify that the build phase did not introduce any new weaknesses. Examples include debuggers that directly attach to the running process; system-call tracing utilities such as truss (Solaris) and strace (Linux); system activity monitors such as FileMon, RegMon, Process Monitor, and other Sysinternals utilities (Windows); and sniffers and protocol analyzers that monitor network traffic.

Attach the monitor to the process and look for library functions that indicate when randomness is being used. Run the process multiple times to see if the seed changes. Look for accesses of devices or equivalent resources that are commonly used for strong (or weak) randomness, such as /dev/urandom on Linux. Look for library or system calls that access predictable information such as process IDs and system time.

Background Details

Computers are deterministic machines, and as such are unable to produce true randomness. Pseudo-Random Number Generators (PRNGs) approximate randomness algorithmically, starting with a seed from which subsequent values are calculated. There are two types of PRNGs: statistical and cryptographic. Statistical PRNGs provide useful statistical properties, but their output is highly predictable and forms an easy to reproduce numeric stream that is unsuitable for use in cases where security depends on generated values being unpredictable. Cryptographic PRNGs address this problem by generating output that is more difficult to predict. For a value to be cryptographically secure, it must be impossible or highly improbable for an attacker to distinguish between it and a truly random value.

Weakness Ordinalities

Ordinality	Description
Primary	*(where the weakness exists independent of other weaknesses)*

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Category	254	Security Features	**Development Concepts (primary)699** **Seven Pernicious Kingdoms (primary)700**
ChildOf	Category	723	OWASP Top Ten 2004 Category A2 - Broken Access Control	**Weaknesses in OWASP Top Ten (2004) (primary)711**
ChildOf	Category	747	CERT C Secure Coding Section 49 - Miscellaneous (MSC)	**Weaknesses Addressed by the CERT C Secure Coding Standard (primary)734**
ChildOf	Category	753	2009 Top 25 - Porous Defenses	**Weaknesses in the 2009 CWE/SANS Top 25 Most Dangerous Programming Errors (primary)750**
ChildOf	Category	808	2010 Top 25 - Weaknesses On the Cusp	**Weaknesses in the 2010 CWE/SANS Top 25 Most Dangerous Programming Errors (primary)800**
ParentOf	Weakness Variant	329	Not Using a Random IV with CBC Mode	**Research Concepts (primary)1000**
ParentOf	Weakness Base	331	Insufficient Entropy	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ParentOf	Weakness Base	334	Small Space of Random Values	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ParentOf	Weakness Class	335	PRNG Seed Error	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ParentOf	Weakness Base	338	Use of Cryptographically Weak PRNG	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ParentOf	Weakness Class	340	Predictability Problems	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ParentOf	Weakness Base	341	Predictable from Observable State	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ParentOf	Weakness Base	342	Predictable Exact Value from Previous Values	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ParentOf	Weakness Base	343	Predictable Value Range from Previous Values	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ParentOf	Weakness Base	344	Use of Invariant Value in Dynamically Changing Context	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ParentOf	Weakness Base	804	Guessable CAPTCHA	Development Concepts699 Research Concepts1000
MemberOf	View	1000	Research Concepts	**Research Concepts (primary)1000**

Relationship Notes

This can be primary to many other weaknesses such as cryptographic errors, authentication errors, symlink following, information leaks, and others.

Functional Areas

- Non-specific
- Cryptography
- Authentication

- Session management

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			Randomness and Predictability
7 Pernicious Kingdoms			Insecure Randomness
OWASP Top Ten 2004	A2	CWE More Specific	Broken Access Control
CERT C Secure Coding	MSC30-C		Do not use the rand() function for generating pseudorandom numbers
WASC	11		Brute Force
WASC	18		Credential/Session Prediction

Related Attack Patterns

CAPEC-ID	Attack Pattern Name	*(CAPEC Version: 1.5)*
59	Session Credential Falsification through Prediction	
112	Brute Force	
281	Analytic Attacks	

References

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

[REF-11] M. Howard and D. LeBlanc. "Writing Secure Code". Chapter 8, "Using Poor Random Numbers" Page 259. 2nd Edition. Microsoft. 2002.

Content History

Submissions

Submission Date	Submitter	Organization	Source
	PLOVER		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 Background Details, Relationships, Other Notes, Relationship Notes, Taxonomy Mappings, Weakness Ordinalities		
2008-11-24	CWE Content Team	MITRE	Internal
	updated Relationships, Taxonomy Mappings		
2009-01-12	CWE Content Team	MITRE	Internal
	updated Description, Likelihood of Exploit, Other Notes, Potential Mitigations, Relationships		
2009-03-10	CWE Content Team	MITRE	Internal
	updated Potential Mitigations		
2009-05-27	CWE Content Team	MITRE	Internal
	updated Demonstrative Examples, Related Attack Patterns		
2009-12-28	CWE Content Team	MITRE	Internal
	updated Applicable Platforms, Common Consequences, Description, Observed Examples, Potential Mitigations, Time of Introduction		
2010-02-16	CWE Content Team	MITRE	Internal
	updated References, Relationships, Taxonomy Mappings		
2010-04-05	CWE Content Team	MITRE	Internal
	updated Related Attack Patterns		

Previous Entry Names

Change Date	Previous Entry Name
2008-04-11	Randomness and Predictability

BACK TO TOP

Improper Sanitization of Script-Related HTML Tags in a Web Page (Basic XSS)

Weakness ID: 80 *(Weakness Variant)* **Status:** Incomplete

Description

Description Summary

The software receives input from an upstream component, but it does not sanitize or incorrectly sanitizes special characters such as "<", ">", and "&" that could be interpreted as web-scripting elements when they are sent to a downstream component that processes web pages.

Extended Description

This may allow such characters to be treated as control characters, which are executed client-side in the context of the user's session. Although this can be classified as an injection problem, the more pertinent issue is the failure to convert such special characters to respective context-appropriate entities before displaying them to the user.

Time of Introduction

- Implementation

Applicable Platforms

Languages

All

Likelihood of Exploit

High to Very High

Demonstrative Examples

Example 1

In the following example, a guestbook comment isn't properly sanitized for script-related tags before being displayed in a client browser.

(Bad Code)

Example Language: **JSP**

```
<% for (Iterator i = guestbook.iterator(); i.hasNext(); ) {
Entry e = (Entry) i.next(); %>
<p>Entry #<%= e.getId() %></p>
<p><%= e.getText() %></p>
<%
} %>
```

Observed Examples

Reference	Description
CVE-2002-0938	XSS in parameter in a link.
CVE-2002-1495	XSS in web-based email product via attachment filenames.
CVE-2003-1136	HTML injection in posted message.
CVE-2004-2171	XSS not quoted in error page.

Potential Mitigations

Carefully check each input parameter against a rigorous positive specification (white list) defining the specific characters and format allowed. All input should be sanitized, not just parameters that the user is supposed to specify, but all data in the request, including hidden fields, cookies, headers, the URL itself, and so forth. A common mistake that leads to continuing XSS vulnerabilities is to validate only fields that are expected to be redisplayed by the site. We often encounter data from the request that is reflected by the application server or the application that the development team did not anticipate. Also, a field that is not currently reflected may be used by a future developer. Therefore, validating ALL parts of the HTTP request is recommended.

This involves "HTML Entity Encoding" all non-alphanumeric characters from data that was received from the user and is now being written to the request.

With Struts, you should write all data from form beans with the bean's filter attribute set to true.

Additionally, to help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as Internet Explorer), this attribute prevents the user's session cookie from being accessed by client-side scripts, including scripts inserted due to a XSS attack.

Weakness Ordinalities

Ordinality	Description
Primary	*(where the weakness exists independent of other weaknesses)*

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Weakness Base	79	Failure to Preserve Web Page Structure ('Cross-site Scripting')	**Development Concepts (primary)699 Research Concepts (primary)1000**
MemberOf	View	630	Weaknesses Examined by SAMATE	**Weaknesses Examined by SAMATE (primary)630**

f Causal Nature

Explicit

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			Basic XSS

Related Attack Patterns

CAPEC-ID	Attack Pattern Name	*(CAPEC Version: 1.5)*
18	Embedding Scripts in Nonscript Elements	

White Box Definitions

A weakness where the code path has:

1. start statement that accepts input from HTML page
2. end statement that publishes a data item to HTML where

a. the input is part of the data item and

b. the input contains XSS syntax

Content History

Submissions

Submission Date	Submitter	Organization	Source
	PLOVER		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		
2008-08-01		KDM Analytics	External
	added/updated white box definitions		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Relationships, Taxonomy Mappings, Weakness Ordinalities		
2008-10-14	CWE Content Team	MITRE	Internal
	updated Description		
2009-05-27	CWE Content Team	MITRE	Internal
	updated Demonstrative Examples, Description, Name		
2009-07-17	KDM Analytics		External
	Improved the White Box Definition		
2009-07-27	CWE Content Team	MITRE	Internal
	updated White Box Definitions		

Previous Entry Names

Change Date	Previous Entry Name
2008-04-11	Basic XSS
2009-05-27	Failure to Sanitize Script-Related HTML Tags in a Web Page (Basic XSS)

BACK TO TOP

ASP.NET Misconfiguration: Creating Debug Binary

Weakness ID: 11 *(Weakness Variant)* **Status:** Draft

Description

Description Summary

Debugging messages help attackers learn about the system and plan a form of attack.

Extended Description

ASP .NET applications can be configured to produce debug binaries. These binaries give detailed debugging messages and should not be used in production environments. Debug binaries are meant to be used in a development or testing environment and can pose a security risk if they are deployed to production.

Time of Introduction

- Implementation
- Operation

Applicable Platforms

Languages

.NET

Common Consequences

Scope	Effect
Confidentiality	Attackers can leverage the additional information they gain from debugging output to mount attacks targeted on the framework, database, or other resources used by the application.

Demonstrative Examples

Example 1

The file web.config contains the debug mode setting. Setting debug to "true" will let the browser display debugging information.

(Bad Code)

Example Language: **XML**

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
<system.web>
<compilation
defaultLanguage="c#"
debug="true"
/>
...
</system.web>
</configuration>
```

Change the debug mode to false when the application is deployed into production.

Potential Mitigations

Avoid releasing debug binaries into the production environment. Change the debug mode to false when the application is deployed into production (See demonstrative example).

Background Details

The debug attribute of the <compilation> tag defines whether compiled binaries should include debugging information. The use of debug binaries causes an application to provide as much information about itself as possible to the user.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Category	2	Environment	**Seven Pernicious Kingdoms (primary)700**
ChildOf	Category	10	ASP.NET Environment Issues	**Development Concepts (primary)699**
ChildOf	Weakness Variant	215	Information Leak Through Debug Information	**Research Concepts (primary)1000**

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
7 Pernicious Kingdoms			ASP.NET Misconfiguration: Creating Debug Binary

Content History

Submissions			
Submission Date	**Submitter**	**Organization**	**Source**
	7 Pernicious Kingdoms		Externally Mined

Modifications			
Modification Date	**Modifier**	**Organization**	**Source**
2008-07-01	Eric Dalci	Cigital	External
	updated Demonstrative Example, Potential Mitigations, Time of Introduction		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Relationships, Other Notes, Taxonomy Mappings		
2008-11-24	CWE Content Team	MITRE	Internal
	updated Description, Other Notes		
2009-07-27	CWE Content Team	MITRE	Internal
	updated Background Details, Common Consequences, Demonstrative Examples, Description, Other Notes		

BACK TO TOP

Non Unique Form Name

Compound Element ID: 10707 Status: Draft

Description

Description Summary

When there are multiple ASP.NET applications running on the server,it's very important to set a unique authentication cookie name for each of the applications.
In case there are applications with the same authentication cookie name, when user logges in into one of the application he can gain acces to the other.

Time of Introduction

- Implementation
- Operation

Applicable Platforms

Languages

ASP.NET

Technology Classes

Web-Server

Demonstrative Examples

Example:

This example in ASP.NET shows us a vulnerable configuration of forms name in a web.config file:

(Bad Code)

Example Language: ASP.NET

```
<configuration>
<system.web>
<authentication mode="Forms">
<forms name=".ASPXAUTH">
```

.ASPXAUTH is the default value for the name of the authentication cookie.

In case there is only one ASP.NET application running on the server the code is secure.

In case there is more ,user logging in to one of the applications can gain access to the other.

For example, when a user logges into some online service, he might inadvertently gain access to the administraion application of the service.

Potential Mitigations

Set the authentication cookie name to a unique value for each of your applications.

Insufficient Session Expiration

Weakness ID: 613 *(Weakness Base)* **Status:** Incomplete

Description

Description Summary

According to WASC, "Insufficient Session Expiration is when a web site permits an attacker to reuse old session credentials or session IDs for authorization."

Time of Introduction

- Architecture and Design
- Implementation

Demonstrative Examples

Example 1

The following snippet was taken from a J2EE web.xml deployment descriptor in which the session-timeout parameter is explicitly defined (the default value depends on the container). In this case the value is set to -1, which means that a session will never expire.

(Bad Code)

Example Language: **Java**

```
<web-app>
[...snipped...]

<session-config>
<session-timeout>-1</session-timeout>
</session-config>
</web-app>
```

Potential Mitigations

Set sessions/credentials expiration date.

Other Notes

The lack of proper session expiration may improve the likely success of certain attacks. For example, an attacker may intercept a session ID, possibly via a network sniffer or Cross-site Scripting attack. Although short session expiration times do not help if a stolen token is immediately used, they will protect against ongoing replaying of the session ID. In another scenario, a user might access a web site from a shared computer (such as at a library, Internet cafe, or open work environment). Insufficient Session Expiration could allow an attacker to use the browser's back button to access web pages previously accessed by the victim.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Category	361	Time and State	**Development Concepts (primary)699**
ChildOf	Weakness Base	672	Operation on a Resource after Expiration or Release	**Research Concepts (primary)1000**
ChildOf	Category	724	OWASP Top Ten 2004 Category A3 - Broken Authentication and Session Management	**Weaknesses in OWASP Top Ten (2004) (primary)711**
CanPrecede	Weakness Class	287	Improper Authentication	Development Concepts699 Research Concepts1000
RequiredBy	Compound Element: Composite	352	Cross-Site Request Forgery (CSRF)	Research Concepts1000

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
WASC			
WASC	47		Insufficient Session Expiration

Content History

Submissions			
Submission Date	**Submitter**	**Organization**	**Source**
	WASC		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 Potential Mitigations, Time of Introduction		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Relationships, Other Notes, Taxonomy Mappings		
2009-03-10	CWE Content Team	MITRE	Internal
	updated Relationships		
2010-02-16	CWE Content Team	MITRE	Internal
	updated Taxonomy Mappings		

BACK TO TOP

Scanned Languages

Language	Hash Number	Change Date
CSharp	0170660826203684	5/21/2015
JavaScript	0633654640145729	5/21/2015
VbScript	2005446206231574	5/21/2015