# Fortify Security Report

Jul 7, 2017

APPSECMON6

## Executive Summary

### Issues Overview

On Jul 6, 2017, a source code review was performed over the cms code base. 174 files, 7,323 LOC (Executable) were scanned and reviewed for defects that could lead to potential security vulnerabilities. A total of 210 reviewed findings were uncovered during the analysis.

### Issues by Category

| | |
|---|---|
| Access Control: Database | 148 |
| SQL Injection | 23 |
| Poor Error Handling: Overly Broad Catch | 22 |
| Cross-Site Scripting: Persistent | 10 |
| Unreleased Resource: Database | 3 |
| Unreleased Resource: Unmanaged Object | 2 |
| ASP.NET Misconfiguration: Debug Information | 1 |
| Resource Injection | 1 |

### Recommendations and Conclusions

The Issues Category section provides Fortify recommendations for addressing issues at a generic level.  The recommendations for specific fixes can be extrapolated from those generic recommendations by the development group.

## Project Summary

### Code Base Summary

Code location:
Number of Files: 174
Lines of Code: 7323
Build Label: <No Build Label>

### Scan Information

Scan time: 02:16
SCA Engine version: 6.10.0120
Machine Name: Ankita
Username running scan: APPSECMON6

### Results Certification

Results Certification Valid

Details:

Results Signature:

SCA Analysis Results has Valid signature

Rules Signature:

There were no custom rules used in this scan

### Filter Set Summary

Current Enabled Filter Set:
Security Auditor View

Filter Set Details:

Folder Filters:
If [fortify priority order] contains critical Then set folder to Critical
If [fortify priority order] contains high Then set folder to High
If [fortify priority order] contains medium Then set folder to Medium
If [fortify priority order] contains low Then set folder to Low

### Audit Guide Summary

Audit guide not enabled

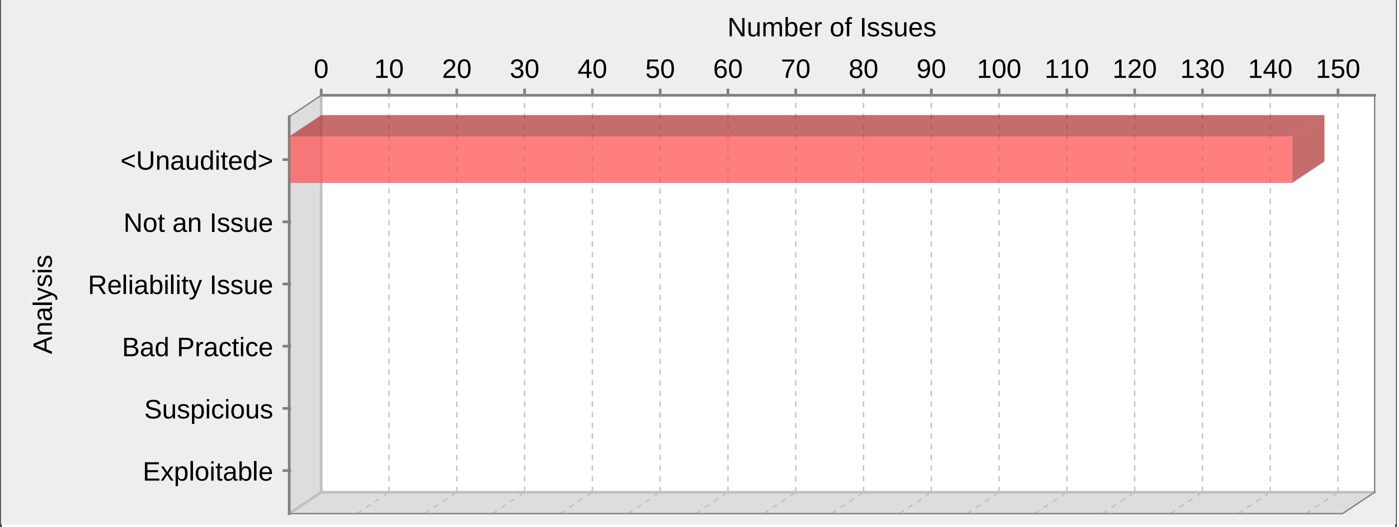
## Results Outline

### Overall number of results

The scan found 210 issues.

### Vulnerability Examples by Category

### Category: Access Control: Database (148 Issues)



**Abstract:**

Without proper access control, executing a SQL statement that contains a user-controlled primary key can allow an attacker to view unauthorized records.

**Explanation:**

Database access control errors occur when:

1. Data enters a program from an untrusted source.

2. The data is used to specify the value of a primary key in a SQL query.

Example 1: The following code uses a parameterized statement, which escapes metacharacters and prevents SQL injection vulnerabilities, to construct and execute a SQL query that searches for an invoice matching the specified identifier. The identifier is selected from a list of all invoices associated with the current authenticated user.

```
...
int16 id = System.Convert.ToInt16(invoiceID.Text);
SqlCommand query = new SqlCommand(
"SELECT * FROM invoices WHERE id = @id", conn);
query.Parameters.AddWithValue("@id", id);
SqlDataReader objReader = query.ExecuteReader();
...
```

The problem is that the developer has failed to consider all of the possible values of id. Although the interface generates a list of invoice identifiers that belong to the current user, an attacker can bypass this interface to request any desired invoice. Because the code in this example does not check to ensure that the user has permission to access the requested invoice, it will display any invoice, even if it does not belong to the current user.

A number of modern web frameworks provide mechanisms for performing validation of user input. ASP.NET Request Validation and WCF are among them. To highlight the unvalidated sources of input, the rulepacks dynamically re-prioritize the issues reported by HP Fortify Static Code Analyzer by lowering their probability of exploit and providing pointers to the supporting evidence whenever the framework validation mechanism is in use. In case of ASP.NET Request Validation, we also provide evidence for when validation is explicitly disabled. We refer to this feature as Context-Sensitive Ranking. To further assist the HP Fortify user with the auditing process, the HP Fortify Software Security Research Group makes available the Data Validation project template that groups the issues into folders based on the validation mechanism applied to their source of input.

**Recommendations:**

Rather than relying on the presentation layer to restrict values submitted by the user, access control should be handled by the application and database layers. Under no circumstances should a user be allowed to retrieve or modify a row in the database without the appropriate permissions. Every query that accesses the database should enforce this policy, which can often be accomplished by simply including the current authenticated username as part of the query.

Example 2: The following code implements the same functionality as Example 1 but imposes an additional constraint requiring that the current authenticated user have specific access to the invoice.

```
...
string user = ctx.getAuthenticatedUserName();
int16 id = System.Convert.ToInt16(invoiceID.Text);
SqlCommand query = new SqlCommand(
"SELECT * FROM invoices WHERE id = @id AND user = @user", conn);
query.Parameters.AddWithValue("@id", id);
query.Parameters.AddWithValue("@user", user);
SqlDataReader objReader = query.ExecuteReader();
...
```

## MemberMaster.aspx.cs, line 453 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 453 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:453 System.Web.UI.WebControls.TextBox.get_Text()

```
451                         cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
452                         cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
453                         cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
454                         cmd.Parameters.AddWithValue("@STATE_CODE",
cmbState.SelectedValue.Trim());
455                         cmd.Parameters.AddWithValue("@state_name",
cmbState.SelectedItem.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:453 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
451                         cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
452                         cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
453                         cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
454                         cmd.Parameters.AddWithValue("@STATE_CODE",
cmbState.SelectedValue.Trim());
455                         cmd.Parameters.AddWithValue("@state_name",
cmbState.SelectedItem.Text.Trim());
```

## Ministry_details.aspx.cs, line 118 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method Savebtn_Click() in Ministry_details.aspx.cs can execute a SQL statement on line 118 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** Ministry_details.aspx.cs:118 System.Web.UI.WebControls.TextBox.get_Text()

```
116                 cmd.Parameters.AddWithValue("@min_name", SqlDbType.VarChar).Value =
mintxt2.Text.Trim();
117                 cmd.Parameters.AddWithValue("@min_name_h", SqlDbType.VarChar).Value =
hmintxt2.Text.Trim();
118                 cmd.Parameters.AddWithValue("@min_ab", SqlDbType.VarChar).Value =
Shortmintxt2.Text.Trim();
119                 cmd.Parameters.AddWithValue("@min_code", SqlDbType.SmallInt).Value =
ViewState["min_code"];
120                 conn.Open();
```

**Sink:** Ministry_details.aspx.cs:118 System.Data.Common.DbParameter.set_Value()

```
116                 cmd.Parameters.AddWithValue("@min_name", SqlDbType.VarChar).Value =
mintxt2.Text.Trim();
117                 cmd.Parameters.AddWithValue("@min_name_h", SqlDbType.VarChar).Value =
hmintxt2.Text.Trim();
118                 cmd.Parameters.AddWithValue("@min_ab", SqlDbType.VarChar).Value =
Shortmintxt2.Text.Trim();
```

```
119                    cmd.Parameters.AddWithValue("@min_code", SqlDbType.SmallInt).Value =
      ViewState["min_code"];
120                    conn.Open();
```

## MemberMasterHindi.aspx.cs, line 597 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 597 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:597 System.Web.UI.WebControls.TextBox.get_Text()

```
595                    cmd.Parameters.AddWithValue("@CONST_CODE",
      cmbConst.SelectedValue.Trim());
596                    cmd.Parameters.AddWithValue("@const_name_h",
      cmbConst.SelectedItem.Text.Trim());
597                    cmd.Parameters.AddWithValue("@mobile2", TxtMobile2.Text.Trim());
598                    cmd.Parameters.AddWithValue("@mobile3", TxtMobile3.Text.Trim());
599                    cmd.Parameters.AddWithValue("@mobile4", TxtMobile4.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:597 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
595                    cmd.Parameters.AddWithValue("@CONST_CODE",
      cmbConst.SelectedValue.Trim());
596                    cmd.Parameters.AddWithValue("@const_name_h",
      cmbConst.SelectedItem.Text.Trim());
597                    cmd.Parameters.AddWithValue("@mobile2", TxtMobile2.Text.Trim());
598                    cmd.Parameters.AddWithValue("@mobile3", TxtMobile3.Text.Trim());
599                    cmd.Parameters.AddWithValue("@mobile4", TxtMobile4.Text.Trim());
```

## MemberMasterHindi.aspx.cs, line 522 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 522 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:522 System.Web.UI.WebControls.TextBox.get_Text()

```
520                        cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
      cmbparty.SelectedValue);
521                        cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
522                        cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
523                        cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
524                        cmd.Parameters.AddWithValue("@MP_CURRENT", 1);
```

**Sink:** MemberMasterHindi.aspx.cs:522 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
520                        cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
      cmbparty.SelectedValue);
521                        cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
522                        cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
523                        cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
524                        cmd.Parameters.AddWithValue("@MP_CURRENT", 1);
```

## MeetingAttendance.aspx.cs, line 468 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method excuteNoquery() in MeetingAttendance.aspx.cs can execute a SQL statement on line 468 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MeetingAttendance.aspx.cs:468 System.Web.UI.WebControls.TextBox.get_Text()

```
466             cmd.Parameters.AddWithValue("@dateofmeet", txtdate1.Text);
467             cmd.Parameters.AddWithValue("@dateofmeet2", txtdate2.Text);
468             cmd.Parameters.AddWithValue("@timeofmeet", txttime.Text);
469             cmd.Parameters.AddWithValue("@cid", cmbcommittee.SelectedValue);
470             cmd.Parameters.AddWithValue("@Cname", cmbcommittee.SelectedItem.Text);
```

**Sink:** MeetingAttendance.aspx.cs:468 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
466             cmd.Parameters.AddWithValue("@dateofmeet", txtdate1.Text);
467             cmd.Parameters.AddWithValue("@dateofmeet2", txtdate2.Text);
468             cmd.Parameters.AddWithValue("@timeofmeet", txttime.Text);
469             cmd.Parameters.AddWithValue("@cid", cmbcommittee.SelectedValue);
470             cmd.Parameters.AddWithValue("@Cname", cmbcommittee.SelectedItem.Text);
```

## MemberMasterHindi.aspx.cs, line 516 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 516 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:516 System.Web.UI.WebControls.TextBox.get_Text()

```
514                     cmd.Parameters.AddWithValue("@HMP_FNAME",
      txtfname.Text.Trim());
515                     cmd.Parameters.AddWithValue("@HMP_LNAME",
      txtlname.Text.Trim());
516                     cmd.Parameters.AddWithValue("@HC_LADDRESS",
      txtlocadd.Text.Trim());
517                     cmd.Parameters.AddWithValue("@Telephone1",
      txtlocph.Text.Trim());
518                     cmd.Parameters.AddWithValue("@HC_PADDRESS",
      txtconadd.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:516 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
514                     cmd.Parameters.AddWithValue("@HMP_FNAME",
      txtfname.Text.Trim());
515                     cmd.Parameters.AddWithValue("@HMP_LNAME",
      txtlname.Text.Trim());
516                     cmd.Parameters.AddWithValue("@HC_LADDRESS",
      txtlocadd.Text.Trim());
517                     cmd.Parameters.AddWithValue("@Telephone1",
      txtlocph.Text.Trim());
518                     cmd.Parameters.AddWithValue("@HC_PADDRESS",
      txtconadd.Text.Trim());
```

## MemberMaster.aspx.cs, line 333 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 333 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:333 System.Web.UI.WebControls.TextBox.get_Text()

```
331                         cmd.Parameters.AddWithValue("@first_name",
      txtfname.Text.Trim());
332                         cmd.Parameters.AddWithValue("@last_name",
      txtlname.Text.Trim());
333                         cmd.Parameters.AddWithValue("@Address1",
      txtlocadd.Text.Trim());
334                         cmd.Parameters.AddWithValue("@Telephone1",
      txtlocph.Text.Trim());
335                         cmd.Parameters.AddWithValue("@Address2",
      txtconadd.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:333 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
331                                cmd.Parameters.AddWithValue("@first_name",
                      txtfname.Text.Trim());
332                                cmd.Parameters.AddWithValue("@last_name",
                      txtlname.Text.Trim());
333                                cmd.Parameters.AddWithValue("@Address1",
                      txtlocadd.Text.Trim());
334                                cmd.Parameters.AddWithValue("@Telephone1",
                      txtlocph.Text.Trim());
335                                cmd.Parameters.AddWithValue("@Address2",
                      txtconadd.Text.Trim());
```

## MemberMaster.aspx.cs, line 445 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 445 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:445 System.Web.UI.WebControls.TextBox.get_Text()

```
443                            cmd.Parameters.AddWithValue("@first_name",
                      txtfname.Text.Trim());
444                            cmd.Parameters.AddWithValue("@last_name",
                      txtlname.Text.Trim());
445                            cmd.Parameters.AddWithValue("@Address1",
                      txtlocadd.Text.Trim());
446                            cmd.Parameters.AddWithValue("@Telephone1",
                      txtlocph.Text.Trim());
447                            cmd.Parameters.AddWithValue("@Address2",
                      txtconadd.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:445 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
443                            cmd.Parameters.AddWithValue("@first_name",
                      txtfname.Text.Trim());
444                            cmd.Parameters.AddWithValue("@last_name",
                      txtlname.Text.Trim());
445                            cmd.Parameters.AddWithValue("@Address1",
                      txtlocadd.Text.Trim());
446                            cmd.Parameters.AddWithValue("@Telephone1",
                      txtlocph.Text.Trim());
447                            cmd.Parameters.AddWithValue("@Address2",
                      txtconadd.Text.Trim());
```

## MemberMaster.aspx.cs, line 336 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 336 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:336 System.Web.UI.WebControls.TextBox.get_Text()

```
334                                cmd.Parameters.AddWithValue("@Telephone1",
                      txtlocph.Text.Trim());
335                                cmd.Parameters.AddWithValue("@Address2",
                      txtconadd.Text.Trim());
336                                cmd.Parameters.AddWithValue("@Telephone2",
                      txtconph.Text.Trim());
337                                cmd.Parameters.AddWithValue("@party_sname",
                      cmbparty.SelectedValue);
338                                cmd.Parameters.AddWithValue("@party_fname",
                      cmbparty.SelectedItem.Text);
```

**Sink:** MemberMaster.aspx.cs:336 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
334                                cmd.Parameters.AddWithValue("@Telephone1",
                      txtlocph.Text.Trim());
335                                cmd.Parameters.AddWithValue("@Address2",
                      txtconadd.Text.Trim());
```

```
336                            cmd.Parameters.AddWithValue("@Telephone2",
          txtconph.Text.Trim());
337                            cmd.Parameters.AddWithValue("@party_sname",
          cmbparty.SelectedValue);
338                            cmd.Parameters.AddWithValue("@party_fname",
          cmbparty.SelectedItem.Text);
```

## MemberMasterHindi.aspx.cs, line 481 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 481 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:481 System.Web.UI.WebControls.TextBox.get_Text()

```
479                        cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
480                        cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
481                        cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
482                        cmd.Parameters.AddWithValue("@last_ls",
          DdlHouseNo.SelectedValue.Trim());
483                        cmd.Parameters.AddWithValue("@STATE_CODE",
          cmbState.SelectedValue.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:481 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
479                        cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
480                        cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
481                        cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
482                        cmd.Parameters.AddWithValue("@last_ls",
          DdlHouseNo.SelectedValue.Trim());
483                        cmd.Parameters.AddWithValue("@STATE_CODE",
          cmbState.SelectedValue.Trim());
```

## MemberMaster.aspx.cs, line 376 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 376 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:376 System.Web.UI.WebControls.TextBox.get_Text()

```
374                            cmd.Parameters.AddWithValue("@C_LADDRESS",
          txtlocadd.Text.Trim());
375                            cmd.Parameters.AddWithValue("@Telephone1",
          txtlocph.Text.Trim());
376                            cmd.Parameters.AddWithValue("@C_PADDRESS",
          txtconadd.Text.Trim());
377                            cmd.Parameters.AddWithValue("@Telephone2",
          txtconph.Text.Trim());
378                            cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
          cmbparty.SelectedValue);
```

**Sink:** MemberMaster.aspx.cs:376 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
374                            cmd.Parameters.AddWithValue("@C_LADDRESS",
          txtlocadd.Text.Trim());
375                            cmd.Parameters.AddWithValue("@Telephone1",
          txtlocph.Text.Trim());
376                            cmd.Parameters.AddWithValue("@C_PADDRESS",
          txtconadd.Text.Trim());
377                            cmd.Parameters.AddWithValue("@Telephone2",
          txtconph.Text.Trim());
378                            cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
          cmbparty.SelectedValue);
```

## MemberMasterHindi.aspx.cs, line 531 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |

|  |  |
|---|---|
| **Kingdom:** | Security Features |
| **Abstract:** | Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 531 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records. |
| **Source:** | MemberMasterHindi.aspx.cs:531 System.Web.UI.WebControls.TextBox.get_Text() |

```
529                                         cmd.Parameters.AddWithValue("@mobile2",
                     TxtMobile2.Text.Trim());
530                                         cmd.Parameters.AddWithValue("@mobile3",
                     TxtMobile3.Text.Trim());
531                                         cmd.Parameters.AddWithValue("@mobile4",
                     TxtMobile4.Text.Trim());
532                                         cmd.Parameters.AddWithValue("@MP_JoinDate",
                     txtMPJoinDate.Text.Trim());
533                                         con.Open();
```

**Sink:** MemberMasterHindi.aspx.cs:531 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
529                                         cmd.Parameters.AddWithValue("@mobile2",
                     TxtMobile2.Text.Trim());
530                                         cmd.Parameters.AddWithValue("@mobile3",
                     TxtMobile3.Text.Trim());
531                                         cmd.Parameters.AddWithValue("@mobile4",
                     TxtMobile4.Text.Trim());
532                                         cmd.Parameters.AddWithValue("@MP_JoinDate",
                     txtMPJoinDate.Text.Trim());
533                                         con.Open();
```

## MemberMasterHindi.aspx.cs, line 599 (Access Control: Database)

|  |  |  |  |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 599 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:599 System.Web.UI.WebControls.TextBox.get_Text()

```
597                                     cmd.Parameters.AddWithValue("@mobile2", TxtMobile2.Text.Trim());
598                                     cmd.Parameters.AddWithValue("@mobile3", TxtMobile3.Text.Trim());
599                                     cmd.Parameters.AddWithValue("@mobile4", TxtMobile4.Text.Trim());
600                                     cmd.Parameters.AddWithValue("@Status",
                     rdobtnStatus.SelectedValue.Trim());
601                                     con.Open();
```

**Sink:** MemberMasterHindi.aspx.cs:599 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
597                                     cmd.Parameters.AddWithValue("@mobile2", TxtMobile2.Text.Trim());
598                                     cmd.Parameters.AddWithValue("@mobile3", TxtMobile3.Text.Trim());
599                                     cmd.Parameters.AddWithValue("@mobile4", TxtMobile4.Text.Trim());
600                                     cmd.Parameters.AddWithValue("@Status",
                     rdobtnStatus.SelectedValue.Trim());
601                                     con.Open();
```

## MemberMasterHindi.aspx.cs, line 517 (Access Control: Database)

|  |  |  |  |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 517 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:517 System.Web.UI.WebControls.TextBox.get_Text()

```
515                                         cmd.Parameters.AddWithValue("@HMP_LNAME",
                     txtlname.Text.Trim());
```

```
516                         cmd.Parameters.AddWithValue("@HC_LADDRESS",
                      txtlocadd.Text.Trim());
517                         cmd.Parameters.AddWithValue("@Telephone1",
                      txtlocph.Text.Trim());
518                         cmd.Parameters.AddWithValue("@HC_PADDRESS",
                      txtconadd.Text.Trim());
519                         cmd.Parameters.AddWithValue("@Telephone2",
                      txtconph.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:517
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
515                         cmd.Parameters.AddWithValue("@HMP_LNAME",
                      txtlname.Text.Trim());
516                         cmd.Parameters.AddWithValue("@HC_LADDRESS",
                      txtlocadd.Text.Trim());
517                         cmd.Parameters.AddWithValue("@Telephone1",
                      txtlocph.Text.Trim());
518                         cmd.Parameters.AddWithValue("@HC_PADDRESS",
                      txtconadd.Text.Trim());
519                         cmd.Parameters.AddWithValue("@Telephone2",
                      txtconph.Text.Trim());
```

## MeetingCommittee.aspx.cs, line 316 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method UpdateSchedule() in MeetingCommittee.aspx.cs can execute a SQL statement on line 316 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MeetingCommittee.aspx.cs:316
System.Web.UI.WebControls.TextBox.get_Text()

```
314                 cmd.Parameters.Add("@title", txttitle.Text);
315                 cmd.Parameters.Add("@dateofmeet", txtdate1.Text);
316                 cmd.Parameters.Add("@dateofmeet2", txtdate2.Text);
317                 if (timchk == "Y")
318                 {
```

**Sink:** MeetingCommittee.aspx.cs:316
System.Data.SqlClient.SqlParameterCollection.Add()

```
314                 cmd.Parameters.Add("@title", txttitle.Text);
315                 cmd.Parameters.Add("@dateofmeet", txtdate1.Text);
316                 cmd.Parameters.Add("@dateofmeet2", txtdate2.Text);
317                 if (timchk == "Y")
318                 {
```

## MemberMasterHindi.aspx.cs, line 490 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 490 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:490
System.Web.UI.WebControls.TextBox.get_Text()

```
488                         cmd.Parameters.AddWithValue("@mobile3",
                      TxtMobile3.Text.Trim());
489                         cmd.Parameters.AddWithValue("@mobile4",
                      TxtMobile4.Text.Trim());
490                         cmd.Parameters.AddWithValue("@MP_JoinDate",
                      txtMPJoinDate.Text.Trim());
491
492                         con.Open();
```

**Sink:** MemberMasterHindi.aspx.cs:490
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
488                         cmd.Parameters.AddWithValue("@mobile3",
                      TxtMobile3.Text.Trim());
```

```
489                         cmd.Parameters.AddWithValue("@mobile4",
        TxtMobile4.Text.Trim());
490                         cmd.Parameters.AddWithValue("@MP_JoinDate",
        txtMPJoinDate.Text.Trim());
491
492                         con.Open();
```

## MemberMaster.aspx.cs, line 451 (Access Control: Database)

**Fortify Priority:** High **Folder** High

**Kingdom:** Security Features

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 451 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:451 System.Web.UI.WebControls.TextBox.get_Text()

```
449                         cmd.Parameters.AddWithValue("@party_sname",
        cmbparty.SelectedValue);
450                         cmd.Parameters.AddWithValue("@party_fname",
        cmbparty.SelectedItem.Text);
451                         cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
452                         cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
453                         cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:451 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
449                         cmd.Parameters.AddWithValue("@party_sname",
        cmbparty.SelectedValue);
450                         cmd.Parameters.AddWithValue("@party_fname",
        cmbparty.SelectedItem.Text);
451                         cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
452                         cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
453                         cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
```

## MemberMaster.aspx.cs, line 493 (Access Control: Database)

**Fortify Priority:** High **Folder** High

**Kingdom:** Security Features

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 493 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:493 System.Web.UI.WebControls.TextBox.get_Text()

```
491                         cmd.Parameters.AddWithValue("@state_name",
        cmbState.SelectedItem.Text.Trim());
492                         cmd.Parameters.AddWithValue("@mobile2",
        TxtMobile2.Text.Trim());
493                         cmd.Parameters.AddWithValue("@mobile3",
        TxtMobile3.Text.Trim());
494                         cmd.Parameters.AddWithValue("@mobile4",
        TxtMobile4.Text.Trim());
495                         cmd.Parameters.AddWithValue("@Status",
        rdobtnStatus.SelectedValue.Trim());
```

**Sink:** MemberMaster.aspx.cs:493 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
491                         cmd.Parameters.AddWithValue("@state_name",
        cmbState.SelectedItem.Text.Trim());
492                         cmd.Parameters.AddWithValue("@mobile2",
        TxtMobile2.Text.Trim());
493                         cmd.Parameters.AddWithValue("@mobile3",
        TxtMobile3.Text.Trim());
494                         cmd.Parameters.AddWithValue("@mobile4",
        TxtMobile4.Text.Trim());
495                         cmd.Parameters.AddWithValue("@Status",
        rdobtnStatus.SelectedValue.Trim());
```

## MemberMaster.aspx.cs, line 390 (Access Control: Database)

**Fortify Priority:** High **Folder** High

**Kingdom:** Security Features

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 390 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:390 System.Web.UI.WebControls.TextBox.get_Text()

```
388                                         cmd.Parameters.AddWithValue("@mobile3",
    TxtMobile3.Text.Trim());
389                                         cmd.Parameters.AddWithValue("@mobile4",
    TxtMobile4.Text.Trim());
390                                         cmd.Parameters.AddWithValue("@MP_JoinDate",
    txtMPJoinDate.Text.Trim());
391                                         con.Open();
392                                         i = cmd.ExecuteNonQuery();
```

**Sink:** MemberMaster.aspx.cs:390 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
388                                         cmd.Parameters.AddWithValue("@mobile3",
    TxtMobile3.Text.Trim());
389                                         cmd.Parameters.AddWithValue("@mobile4",
    TxtMobile4.Text.Trim());
390                                         cmd.Parameters.AddWithValue("@MP_JoinDate",
    txtMPJoinDate.Text.Trim());
391                                         con.Open();
392                                         i = cmd.ExecuteNonQuery();
```

## MemberMasterHindi.aspx.cs, line 530 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 530 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:530 System.Web.UI.WebControls.TextBox.get_Text()

```
528                                     cmd.Parameters.AddWithValue("@const_name_h",
    cmbConst.SelectedItem.Text.Trim());
529                                     cmd.Parameters.AddWithValue("@mobile2",
    TxtMobile2.Text.Trim());
530                                     cmd.Parameters.AddWithValue("@mobile3",
    TxtMobile3.Text.Trim());
531                                     cmd.Parameters.AddWithValue("@mobile4",
    TxtMobile4.Text.Trim());
532                                     cmd.Parameters.AddWithValue("@MP_JoinDate",
    txtMPJoinDate.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:530 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
528                                     cmd.Parameters.AddWithValue("@const_name_h",
    cmbConst.SelectedItem.Text.Trim());
529                                     cmd.Parameters.AddWithValue("@mobile2",
    TxtMobile2.Text.Trim());
530                                     cmd.Parameters.AddWithValue("@mobile3",
    TxtMobile3.Text.Trim());
531                                     cmd.Parameters.AddWithValue("@mobile4",
    TxtMobile4.Text.Trim());
532                                     cmd.Parameters.AddWithValue("@MP_JoinDate",
    txtMPJoinDate.Text.Trim());
```

## MemberMasterHindi.aspx.cs, line 523 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 523 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:523
System.Web.UI.WebControls.TextBox.get_Text()

```
521                             cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
522                             cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
523                             cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
524                             cmd.Parameters.AddWithValue("@MP_CURRENT", 1);
525                             cmd.Parameters.AddWithValue("@C_MP_STATE_CODE",
                  cmbState.SelectedValue.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:523
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
521                             cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
522                             cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
523                             cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
524                             cmd.Parameters.AddWithValue("@MP_CURRENT", 1);
525                             cmd.Parameters.AddWithValue("@C_MP_STATE_CODE",
                  cmbState.SelectedValue.Trim());
```

## MemberMasterHindi.aspx.cs, line 587 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 587 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:587
System.Web.UI.WebControls.TextBox.get_Text()

```
585                         cmd.Parameters.AddWithValue("@H_Address1", txtlocadd.Text.Trim());
586                         cmd.Parameters.AddWithValue("@Telephone1", txtlocph.Text.Trim());
587                         cmd.Parameters.AddWithValue("@H_Address2", txtconadd.Text.Trim());
588                         cmd.Parameters.AddWithValue("@Telephone2", txtconph.Text.Trim());
589                         cmd.Parameters.AddWithValue("@party_sname",
                  cmbparty.SelectedValue);
```

**Sink:** MemberMasterHindi.aspx.cs:587
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
585                         cmd.Parameters.AddWithValue("@H_Address1", txtlocadd.Text.Trim());
586                         cmd.Parameters.AddWithValue("@Telephone1", txtlocph.Text.Trim());
587                         cmd.Parameters.AddWithValue("@H_Address2", txtconadd.Text.Trim());
588                         cmd.Parameters.AddWithValue("@Telephone2", txtconph.Text.Trim());
589                         cmd.Parameters.AddWithValue("@party_sname",
                  cmbparty.SelectedValue);
```

## MemberMasterHindi.aspx.cs, line 592 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 592 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:592
System.Web.UI.WebControls.TextBox.get_Text()

```
590                         cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
591                         cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
592                         cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
593                         cmd.Parameters.AddWithValue("@STATE_CODE",
                  cmbState.SelectedValue.Trim());
594                         cmd.Parameters.AddWithValue("@state_name_h",
                  cmbState.SelectedItem.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:592
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
590                         cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
591                         cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
592                         cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
```

```
593                           cmd.Parameters.AddWithValue("@STATE_CODE",
cmbState.SelectedValue.Trim());
594                           cmd.Parameters.AddWithValue("@state_name_h",
cmbState.SelectedItem.Text.Trim());
```

## MeetingCommittee.aspx.cs, line 328 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method UpdateSchedule() in MeetingCommittee.aspx.cs can execute a SQL statement on line 328 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MeetingCommittee.aspx.cs:328 System.Web.UI.WebControls.TextBox.get_Text()

```
326             cmd.Parameters.Add("@subject", txtpurpose.Text);
327             cmd.Parameters.Add("@venue", txtvenue.Text);
328             cmd.Parameters.Add("@remarks", txtremarks.Text);
329             cmd.Parameters.Add("@TypeOfCommittee", rdoCommitteeType.SelectedValue);
330             cmd.Parameters.Add("@FileNo", TxtFileNo.Text.Trim());
```

**Sink:** MeetingCommittee.aspx.cs:328 System.Data.SqlClient.SqlParameterCollection.Add()

```
326             cmd.Parameters.Add("@subject", txtpurpose.Text);
327             cmd.Parameters.Add("@venue", txtvenue.Text);
328             cmd.Parameters.Add("@remarks", txtremarks.Text);
329             cmd.Parameters.Add("@TypeOfCommittee", rdoCommitteeType.SelectedValue);
330             cmd.Parameters.Add("@FileNo", TxtFileNo.Text.Trim());
```

## MemberMaster.aspx.cs, line 489 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 489 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:489 System.Web.UI.WebControls.TextBox.get_Text()

```
487                         cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
488                         cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
489                         cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
490                         cmd.Parameters.AddWithValue("@C_MP_STATE_CODE",
cmbState.SelectedValue.Trim());
491                         cmd.Parameters.AddWithValue("@state_name",
cmbState.SelectedItem.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:489 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
487                         cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
488                         cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
489                         cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
490                         cmd.Parameters.AddWithValue("@C_MP_STATE_CODE",
cmbState.SelectedValue.Trim());
491                         cmd.Parameters.AddWithValue("@state_name",
cmbState.SelectedItem.Text.Trim());
```

## MemberMasterHindi.aspx.cs, line 626 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 626 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:626 System.Web.UI.WebControls.TextBox.get_Text()

```
624                                        cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
                      cmbparty.SelectedValue);
625                                        cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
626                                        cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
627                                        cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
628                                        cmd.Parameters.AddWithValue("@C_MP_STATE_CODE",
                      cmbState.SelectedValue.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:626
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
624                                        cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
                      cmbparty.SelectedValue);
625                                        cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
626                                        cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
627                                        cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
628                                        cmd.Parameters.AddWithValue("@C_MP_STATE_CODE",
                      cmbState.SelectedValue.Trim());
```

## MemberMasterHindi.aspx.cs, line 471 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 471 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:471
System.Web.UI.WebControls.TextBox.get_Text()

```
469                                            cmd.Parameters.AddWithValue("@mpsno", id.ToString());
470                                            cmd.Parameters.AddWithValue("@initial_h",
                      txtinitial.Text.Trim());
471                                            cmd.Parameters.AddWithValue("@first_name_h",
                      txtfname.Text.Trim());
472                                            cmd.Parameters.AddWithValue("@last_name_h",
                      txtlname.Text.Trim());
473                                            cmd.Parameters.AddWithValue("@H_Address1",
                      txtlocadd.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:471
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
469                                            cmd.Parameters.AddWithValue("@mpsno", id.ToString());
470                                            cmd.Parameters.AddWithValue("@initial_h",
                      txtinitial.Text.Trim());
471                                            cmd.Parameters.AddWithValue("@first_name_h",
                      txtfname.Text.Trim());
472                                            cmd.Parameters.AddWithValue("@last_name_h",
                      txtlname.Text.Trim());
473                                            cmd.Parameters.AddWithValue("@H_Address1",
                      txtlocadd.Text.Trim());
```

## MemberMasterHindi.aspx.cs, line 632 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 632 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:632
System.Web.UI.WebControls.TextBox.get_Text()

```
630                                        cmd.Parameters.AddWithValue("@CONST_CODE",
                      cmbConst.SelectedValue.Trim());
631                                        cmd.Parameters.AddWithValue("@const_name_h",
                      cmbConst.SelectedItem.Text.Trim());
632                                        cmd.Parameters.AddWithValue("@mobile2", TxtMobile2.Text.Trim());
633                                        cmd.Parameters.AddWithValue("@mobile3", TxtMobile3.Text.Trim());
634                                        cmd.Parameters.AddWithValue("@mobile4", TxtMobile4.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:632
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
630                           cmd.Parameters.AddWithValue("@CONST_CODE",
cmbConst.SelectedValue.Trim());
631                           cmd.Parameters.AddWithValue("@const_name_h",
cmbConst.SelectedItem.Text.Trim());
632                           cmd.Parameters.AddWithValue("@mobile2", TxtMobile2.Text.Trim());
633                           cmd.Parameters.AddWithValue("@mobile3", TxtMobile3.Text.Trim());
634                           cmd.Parameters.AddWithValue("@mobile4", TxtMobile4.Text.Trim());
```

## MemberMasterHindi.aspx.cs, line 519 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 519 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:519 System.Web.UI.WebControls.TextBox.get_Text()

```
517                               cmd.Parameters.AddWithValue("@Telephone1",
txtlocph.Text.Trim());
518                               cmd.Parameters.AddWithValue("@HC_PADDRESS",
txtconadd.Text.Trim());
519                               cmd.Parameters.AddWithValue("@Telephone2",
txtconph.Text.Trim());
520                               cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
cmbparty.SelectedValue);
521                               cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:519 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
517                               cmd.Parameters.AddWithValue("@Telephone1",
txtlocph.Text.Trim());
518                               cmd.Parameters.AddWithValue("@HC_PADDRESS",
txtconadd.Text.Trim());
519                               cmd.Parameters.AddWithValue("@Telephone2",
txtconph.Text.Trim());
520                               cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
cmbparty.SelectedValue);
521                               cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
```

## MemberMaster.aspx.cs, line 332 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 332 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:332 System.Web.UI.WebControls.TextBox.get_Text()

```
330                                   cmd.Parameters.AddWithValue("@Initial",
txtinitial.Text.Trim());
331                                   cmd.Parameters.AddWithValue("@first_name",
txtfname.Text.Trim());
332                                   cmd.Parameters.AddWithValue("@last_name",
txtlname.Text.Trim());
333                                   cmd.Parameters.AddWithValue("@Address1",
txtlocadd.Text.Trim());
334                                   cmd.Parameters.AddWithValue("@Telephone1",
txtlocph.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:332 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
330                                   cmd.Parameters.AddWithValue("@Initial",
txtinitial.Text.Trim());
331                                   cmd.Parameters.AddWithValue("@first_name",
txtfname.Text.Trim());
332                                   cmd.Parameters.AddWithValue("@last_name",
txtlname.Text.Trim());
333                                   cmd.Parameters.AddWithValue("@Address1",
txtlocadd.Text.Trim());
```

```
334                                    cmd.Parameters.AddWithValue("@Telephone1",
                     txtlocph.Text.Trim());
```

## MemberMaster.aspx.cs, line 387 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 387 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:387 System.Web.UI.WebControls.TextBox.get_Text()

```
385                                    cmd.Parameters.AddWithValue("@CONST_CODE",
                     cmbConst.SelectedValue.Trim());
386                                    cmd.Parameters.AddWithValue("@CONST_NAME",
                     cmbConst.SelectedItem.Text.Trim());
387                                    cmd.Parameters.AddWithValue("@mobile2",
                     TxtMobile2.Text.Trim());
388                                    cmd.Parameters.AddWithValue("@mobile3",
                     TxtMobile3.Text.Trim());
389                                    cmd.Parameters.AddWithValue("@mobile4",
                     TxtMobile4.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:387 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
385                                    cmd.Parameters.AddWithValue("@CONST_CODE",
                     cmbConst.SelectedValue.Trim());
386                                    cmd.Parameters.AddWithValue("@CONST_NAME",
                     cmbConst.SelectedItem.Text.Trim());
387                                    cmd.Parameters.AddWithValue("@mobile2",
                     TxtMobile2.Text.Trim());
388                                    cmd.Parameters.AddWithValue("@mobile3",
                     TxtMobile3.Text.Trim());
389                                    cmd.Parameters.AddWithValue("@mobile4",
                     TxtMobile4.Text.Trim());
```

## MemberMasterHindi.aspx.cs, line 586 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 586 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:586 System.Web.UI.WebControls.TextBox.get_Text()

```
584                              cmd.Parameters.AddWithValue("@last_name_h", txtlname.Text.Trim());
585                              cmd.Parameters.AddWithValue("@H_Address1", txtlocadd.Text.Trim());
586                              cmd.Parameters.AddWithValue("@Telephone1", txtlocph.Text.Trim());
587                              cmd.Parameters.AddWithValue("@H_Address2", txtconadd.Text.Trim());
588                              cmd.Parameters.AddWithValue("@Telephone2", txtconph.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:586 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
584                              cmd.Parameters.AddWithValue("@last_name_h", txtlname.Text.Trim());
585                              cmd.Parameters.AddWithValue("@H_Address1", txtlocadd.Text.Trim());
586                              cmd.Parameters.AddWithValue("@Telephone1", txtlocph.Text.Trim());
587                              cmd.Parameters.AddWithValue("@H_Address2", txtconadd.Text.Trim());
588                              cmd.Parameters.AddWithValue("@Telephone2", txtconph.Text.Trim());
```

## MemberMaster.aspx.cs, line 481 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 481 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:481 System.Web.UI.WebControls.TextBox.get_Text()

```
479                         cmd.Parameters.AddWithValue("@MP_INIT",
txtinitial.Text.Trim());
480                         cmd.Parameters.AddWithValue("@MP_FNAME",
txtfname.Text.Trim());
481                         cmd.Parameters.AddWithValue("@MP_LNAME",
txtlname.Text.Trim());
482                         cmd.Parameters.AddWithValue("@C_LADDRESS",
txtlocadd.Text.Trim());
483                         cmd.Parameters.AddWithValue("@Telephone1",
txtlocph.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:481 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
479                         cmd.Parameters.AddWithValue("@MP_INIT",
txtinitial.Text.Trim());
480                         cmd.Parameters.AddWithValue("@MP_FNAME",
txtfname.Text.Trim());
481                         cmd.Parameters.AddWithValue("@MP_LNAME",
txtlname.Text.Trim());
482                         cmd.Parameters.AddWithValue("@C_LADDRESS",
txtlocadd.Text.Trim());
483                         cmd.Parameters.AddWithValue("@Telephone1",
txtlocph.Text.Trim());
```

## MemberMaster.aspx.cs, line 484 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 484 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:484 System.Web.UI.WebControls.TextBox.get_Text()

```
482                         cmd.Parameters.AddWithValue("@C_LADDRESS",
txtlocadd.Text.Trim());
483                         cmd.Parameters.AddWithValue("@Telephone1",
txtlocph.Text.Trim());
484                         cmd.Parameters.AddWithValue("@C_PADDRESS",
txtconadd.Text.Trim());
485                         cmd.Parameters.AddWithValue("@Telephone2",
txtconph.Text.Trim());
486                         cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
cmbparty.SelectedValue);
```

**Sink:** MemberMaster.aspx.cs:484 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
482                         cmd.Parameters.AddWithValue("@C_LADDRESS",
txtlocadd.Text.Trim());
483                         cmd.Parameters.AddWithValue("@Telephone1",
txtlocph.Text.Trim());
484                         cmd.Parameters.AddWithValue("@C_PADDRESS",
txtconadd.Text.Trim());
485                         cmd.Parameters.AddWithValue("@Telephone2",
txtconph.Text.Trim());
486                         cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
cmbparty.SelectedValue);
```

## PartyMaster.aspx.cs, line 210 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in PartyMaster.aspx.cs can execute a SQL statement on line 210 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** PartyMaster.aspx.cs:210 System.Web.UI.WebControls.TextBox.get_Text()

```
208                     cmd.Parameters.AddWithValue("@PARTY_FNAME_H",
SqlDbType.NVarChar).Value = txtHPartyName.Text.Trim();
209                     cmd.Parameters.AddWithValue("@PARTY_SNAME_H",
SqlDbType.NVarChar).Value = txtHShortName.Text.Trim();
210                     cmd.Parameters.AddWithValue("@LSTO", SqlDbType.SmallInt).Value =
txtLSTo.Text.Trim();
```

```
211                   cmd.Parameters.AddWithValue("@LEADER", SqlDbType.VarChar).Value =
      txtLeader.Text.Trim();
212                   cmd.Parameters.AddWithValue("@LEADER_H", SqlDbType.NVarChar).Value
      = txtLeaderH.Text.Trim();
```

**Sink:** PartyMaster.aspx.cs:210 System.Data.Common.DbParameter.set_Value()

```
208                   cmd.Parameters.AddWithValue("@PARTY_FNAME_H",
      SqlDbType.NVarChar).Value = txtHPartyName.Text.Trim();
209                   cmd.Parameters.AddWithValue("@PARTY_SNAME_H",
      SqlDbType.NVarChar).Value = txtHShortName.Text.Trim();
210                   cmd.Parameters.AddWithValue("@LSTO", SqlDbType.SmallInt).Value =
      txtLSTo.Text.Trim();
211                   cmd.Parameters.AddWithValue("@LEADER", SqlDbType.VarChar).Value =
      txtLeader.Text.Trim();
212                   cmd.Parameters.AddWithValue("@LEADER_H", SqlDbType.NVarChar).Value
      = txtLeaderH.Text.Trim();
```

## PartyMaster.aspx.cs, line 198 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in PartyMaster.aspx.cs can execute a SQL statement on line 198 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** PartyMaster.aspx.cs:198 System.Web.UI.WebControls.TextBox.get_Text()

```
196                   cmd.Parameters.AddWithValue("@PARTY_SNAME_H",
      SqlDbType.NVarChar).Value = txtHShortName.Text.Trim();
197                   cmd.Parameters.AddWithValue("@LEADER", SqlDbType.VarChar).Value =
      txtLeader.Text.Trim();
198                   cmd.Parameters.AddWithValue("@LEADER_H", SqlDbType.NVarChar).Value
      = txtLeaderH.Text.Trim();
199                   cmd.Parameters.AddWithValue("@PARTY_CODE",
      SqlDbType.SmallInt).Value = ViewState["Key"];
200                   con.Open();
```

**Sink:** PartyMaster.aspx.cs:198 System.Data.Common.DbParameter.set_Value()

```
196                   cmd.Parameters.AddWithValue("@PARTY_SNAME_H",
      SqlDbType.NVarChar).Value = txtHShortName.Text.Trim();
197                   cmd.Parameters.AddWithValue("@LEADER", SqlDbType.VarChar).Value =
      txtLeader.Text.Trim();
198                   cmd.Parameters.AddWithValue("@LEADER_H", SqlDbType.NVarChar).Value
      = txtLeaderH.Text.Trim();
199                   cmd.Parameters.AddWithValue("@PARTY_CODE",
      SqlDbType.SmallInt).Value = ViewState["Key"];
200                   con.Open();
```

## MemberMasterHindi.aspx.cs, line 590 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 590 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:590 System.Web.UI.WebControls.TextBox.get_Text()

```
588                   cmd.Parameters.AddWithValue("@Telephone2", txtconph.Text.Trim());
589                   cmd.Parameters.AddWithValue("@party_sname",
      cmbparty.SelectedValue);
590                   cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
591                   cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
592                   cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:590 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
588                   cmd.Parameters.AddWithValue("@Telephone2", txtconph.Text.Trim());
589                   cmd.Parameters.AddWithValue("@party_sname",
      cmbparty.SelectedValue);
590                   cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
591                   cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
592                   cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
```

## MemberMaster.aspx.cs, line 444 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 444 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:444 System.Web.UI.WebControls.TextBox.get_Text()

```
442                                cmd.Parameters.AddWithValue("@Initial",
      txtinitial.Text.Trim());
443                                cmd.Parameters.AddWithValue("@first_name",
      txtfname.Text.Trim());
444                                cmd.Parameters.AddWithValue("@last_name",
      txtlname.Text.Trim());
445                                cmd.Parameters.AddWithValue("@Address1",
      txtlocadd.Text.Trim());
446                                cmd.Parameters.AddWithValue("@Telephone1",
      txtlocph.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:444 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
442                                cmd.Parameters.AddWithValue("@Initial",
      txtinitial.Text.Trim());
443                                cmd.Parameters.AddWithValue("@first_name",
      txtfname.Text.Trim());
444                                cmd.Parameters.AddWithValue("@last_name",
      txtlname.Text.Trim());
445                                cmd.Parameters.AddWithValue("@Address1",
      txtlocadd.Text.Trim());
446                                cmd.Parameters.AddWithValue("@Telephone1",
      txtlocph.Text.Trim());
```

## depatment_detail.aspx.cs, line 175 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method Savebtn_Click() in depatment_detail.aspx.cs can execute a SQL statement on line 175 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** depatment_detail.aspx.cs:175 System.Web.UI.WebControls.TextBox.get_Text()

```
173                   cmd = new SqlCommand("departmentInsert", conn);
174                   cmd.CommandType = CommandType.StoredProcedure;
175                   cmd.Parameters.AddWithValue("@dep_name", dep1.Text);
176                   cmd.Parameters.AddWithValue("@hdep_name", hdep1.Text);
177                   cmd.Parameters.AddWithValue("@mnstry_code", minstry_id);
```

**Sink:** depatment_detail.aspx.cs:175 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
173                   cmd = new SqlCommand("departmentInsert", conn);
174                   cmd.CommandType = CommandType.StoredProcedure;
175                   cmd.Parameters.AddWithValue("@dep_name", dep1.Text);
176                   cmd.Parameters.AddWithValue("@hdep_name", hdep1.Text);
177                   cmd.Parameters.AddWithValue("@mnstry_code", minstry_id);
```

## MeetingAttendance.aspx.cs, line 449 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method excuteYesquery() in MeetingAttendance.aspx.cs can execute a SQL statement on line 449 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MeetingAttendance.aspx.cs:449 System.Web.UI.WebControls.TextBox.get_Text()

```
447                     cmd.Parameters.AddWithValue("@CommitteeType", rdoCommitteeType.SelectedValue);
448                     cmd.Parameters.AddWithValue("@Title", cmbtitle.Text);
449                     cmd.Parameters.AddWithValue("@dateofmeet", txtdate1.Text);
450                     cmd.Parameters.AddWithValue("@dateofmeet2", txtdate2.Text);
451                     cmd.Parameters.AddWithValue("@timeofmeet", txttime.Text);
```

**Sink:** MeetingAttendance.aspx.cs:449
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
447                     cmd.Parameters.AddWithValue("@CommitteeType", rdoCommitteeType.SelectedValue);
448                     cmd.Parameters.AddWithValue("@Title", cmbtitle.Text);
449                     cmd.Parameters.AddWithValue("@dateofmeet", txtdate1.Text);
450                     cmd.Parameters.AddWithValue("@dateofmeet2", txtdate2.Text);
451                     cmd.Parameters.AddWithValue("@timeofmeet", txttime.Text);
```

## CommitteeMaster.aspx.cs, line 190 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in CommitteeMaster.aspx.cs can execute a SQL statement on line 190 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** CommitteeMaster.aspx.cs:190
System.Web.UI.WebControls.TextBox.get_Text()

```
188                     cmd.Parameters.AddWithValue("@Cid", SqlDbType.Int).Value =
ViewState["Key"];
189                     cmd.Parameters.AddWithValue("@Cname", SqlDbType.VarChar).Value =
txtCommName.Text.Trim();
190                     cmd.Parameters.AddWithValue("@Hcname", SqlDbType.VarChar).Value =
txtHCommName.Text.Trim();
191                     con.Open();
192                     i = cmd.ExecuteNonQuery();
```

**Sink:** CommitteeMaster.aspx.cs:190
System.Data.Common.DbParameter.set_Value()

```
188                     cmd.Parameters.AddWithValue("@Cid", SqlDbType.Int).Value =
ViewState["Key"];
189                     cmd.Parameters.AddWithValue("@Cname", SqlDbType.VarChar).Value =
txtCommName.Text.Trim();
190                     cmd.Parameters.AddWithValue("@Hcname", SqlDbType.VarChar).Value =
txtHCommName.Text.Trim();
191                     con.Open();
192                     i = cmd.ExecuteNonQuery();
```

## MemberMaster.aspx.cs, line 492 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 492 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:492 System.Web.UI.WebControls.TextBox.get_Text()

```
490                     cmd.Parameters.AddWithValue("@C_MP_STATE_CODE",
cmbState.SelectedValue.Trim());
491                     cmd.Parameters.AddWithValue("@state_name",
cmbState.SelectedItem.Text.Trim());
492                     cmd.Parameters.AddWithValue("@mobile2",
TxtMobile2.Text.Trim());
493                     cmd.Parameters.AddWithValue("@mobile3",
TxtMobile3.Text.Trim());
494                     cmd.Parameters.AddWithValue("@mobile4",
TxtMobile4.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:492
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
490                     cmd.Parameters.AddWithValue("@C_MP_STATE_CODE",
cmbState.SelectedValue.Trim());
491                     cmd.Parameters.AddWithValue("@state_name",
cmbState.SelectedItem.Text.Trim());
```

```
492                                        cmd.Parameters.AddWithValue("@mobile2",
                     TxtMobile2.Text.Trim());
493                                        cmd.Parameters.AddWithValue("@mobile3",
                     TxtMobile3.Text.Trim());
494                                        cmd.Parameters.AddWithValue("@mobile4",
                     TxtMobile4.Text.Trim());
```

## MemberMasterHindi.aspx.cs, line 584 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 584 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:584
System.Web.UI.WebControls.TextBox.get_Text()

```
582                                    cmd.Parameters.AddWithValue("@initial_h", txtinitial.Text.Trim());
583                                    cmd.Parameters.AddWithValue("@first_name_h",
                     txtfname.Text.Trim());
584                                    cmd.Parameters.AddWithValue("@last_name_h", txtlname.Text.Trim());
585                                    cmd.Parameters.AddWithValue("@H_Address1", txtlocadd.Text.Trim());
586                                    cmd.Parameters.AddWithValue("@Telephone1", txtlocph.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:584
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
582                                    cmd.Parameters.AddWithValue("@initial_h", txtinitial.Text.Trim());
583                                    cmd.Parameters.AddWithValue("@first_name_h",
                     txtfname.Text.Trim());
584                                    cmd.Parameters.AddWithValue("@last_name_h", txtlname.Text.Trim());
585                                    cmd.Parameters.AddWithValue("@H_Address1", txtlocadd.Text.Trim());
586                                    cmd.Parameters.AddWithValue("@Telephone1", txtlocph.Text.Trim());
```

## MemberMasterHindi.aspx.cs, line 588 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 588 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:588
System.Web.UI.WebControls.TextBox.get_Text()

```
586                                    cmd.Parameters.AddWithValue("@Telephone1", txtlocph.Text.Trim());
587                                    cmd.Parameters.AddWithValue("@H_Address2", txtconadd.Text.Trim());
588                                    cmd.Parameters.AddWithValue("@Telephone2", txtconph.Text.Trim());
589                                    cmd.Parameters.AddWithValue("@party_sname",
                     cmbparty.SelectedValue);
590                                    cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:588
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
586                                    cmd.Parameters.AddWithValue("@Telephone1", txtlocph.Text.Trim());
587                                    cmd.Parameters.AddWithValue("@H_Address2", txtconadd.Text.Trim());
588                                    cmd.Parameters.AddWithValue("@Telephone2", txtconph.Text.Trim());
589                                    cmd.Parameters.AddWithValue("@party_sname",
                     cmbparty.SelectedValue);
590                                    cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
```

## MeetingAttendance.aspx.cs, line 466 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method excuteNoquery() in MeetingAttendance.aspx.cs can execute a SQL statement on line 466 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MeetingAttendance.aspx.cs:466 System.Web.UI.WebControls.TextBox.get_Text()

```
464                 cmd.Parameters.AddWithValue("@CommitteeType", rdoCommitteeType.SelectedValue);
465                 cmd.Parameters.AddWithValue("@Title", cmbtitle.Text);
466                 cmd.Parameters.AddWithValue("@dateofmeet", txtdate1.Text);
467                 cmd.Parameters.AddWithValue("@dateofmeet2", txtdate2.Text);
468                 cmd.Parameters.AddWithValue("@timeofmeet", txttime.Text);
```

**Sink:** MeetingAttendance.aspx.cs:466 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
464                 cmd.Parameters.AddWithValue("@CommitteeType", rdoCommitteeType.SelectedValue);
465                 cmd.Parameters.AddWithValue("@Title", cmbtitle.Text);
466                 cmd.Parameters.AddWithValue("@dateofmeet", txtdate1.Text);
467                 cmd.Parameters.AddWithValue("@dateofmeet2", txtdate2.Text);
468                 cmd.Parameters.AddWithValue("@timeofmeet", txttime.Text);
```

## MemberMaster.aspx.cs, line 381 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 381 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:381 System.Web.UI.WebControls.TextBox.get_Text()

```
379                             cmd.Parameters.AddWithValue("@mobile",
TxtMobile.Text.Trim());
380                             cmd.Parameters.AddWithValue("@email1",
TxteMail1.Text.Trim());
381                             cmd.Parameters.AddWithValue("@email2",
TxteMail2.Text.Trim());
382                             cmd.Parameters.AddWithValue("@MP_CURRENT", 1);
383                             cmd.Parameters.AddWithValue("@C_MP_STATE_CODE",
cmbState.SelectedValue.Trim());
```

**Sink:** MemberMaster.aspx.cs:381 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
379                             cmd.Parameters.AddWithValue("@mobile",
TxtMobile.Text.Trim());
380                             cmd.Parameters.AddWithValue("@email1",
TxteMail1.Text.Trim());
381                             cmd.Parameters.AddWithValue("@email2",
TxteMail2.Text.Trim());
382                             cmd.Parameters.AddWithValue("@MP_CURRENT", 1);
383                             cmd.Parameters.AddWithValue("@C_MP_STATE_CODE",
cmbState.SelectedValue.Trim());
```

## MemberMaster.aspx.cs, line 341 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 341 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:341 System.Web.UI.WebControls.TextBox.get_Text()

```
339                             cmd.Parameters.AddWithValue("@mobile",
TxtMobile.Text.Trim());
340                             cmd.Parameters.AddWithValue("@email1",
TxteMail1.Text.Trim());
341                             cmd.Parameters.AddWithValue("@email2",
TxteMail2.Text.Trim());
342                             cmd.Parameters.AddWithValue("@last_ls",
DdlHouseNo.SelectedValue.Trim());
343                             cmd.Parameters.AddWithValue("@STATE_CODE",
cmbState.SelectedValue.Trim());
```

**Sink:** MemberMaster.aspx.cs:341 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
cmd.Parameters.AddWithValue("@mobile",
TxtMobile.Text.Trim());
cmd.Parameters.AddWithValue("@email1",
TxteMail1.Text.Trim());
cmd.Parameters.AddWithValue("@email2",
TxteMail2.Text.Trim());
cmd.Parameters.AddWithValue("@last_ls",
DdlHouseNo.SelectedValue.Trim());
cmd.Parameters.AddWithValue("@STATE_CODE",
cmbState.SelectedValue.Trim());
```

## MemberCommittee.aspx.cs, line 575 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method CMSUpdateMemCom() in MemberCommittee.aspx.cs can execute a SQL statement on line 575 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberCommittee.aspx.cs:575 System.Web.UI.WebControls.TextBox.get_Text()

```
cmd.Parameters.Add("@MinId", cmbminname.SelectedValue);
cmd.Parameters.Add("@DeptId", cmbdepat.SelectedValue);
cmd.Parameters.Add("@BodiesName", txtBodieName.Text.Trim());
cmd.Parameters.Add("@radio", rdoCommitteeType.SelectedValue);
cmd.ExecuteNonQuery();
```

**Sink:** MemberCommittee.aspx.cs:575 System.Data.SqlClient.SqlParameterCollection.Add()

```
cmd.Parameters.Add("@MinId", cmbminname.SelectedValue);
cmd.Parameters.Add("@DeptId", cmbdepat.SelectedValue);
cmd.Parameters.Add("@BodiesName", txtBodieName.Text.Trim());
cmd.Parameters.Add("@radio", rdoCommitteeType.SelectedValue);
cmd.ExecuteNonQuery();
```

## MemberMaster.aspx.cs, line 448 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 448 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:448 System.Web.UI.WebControls.TextBox.get_Text()

```
cmd.Parameters.AddWithValue("@Telephone1",
txtlocph.Text.Trim());
cmd.Parameters.AddWithValue("@Address2",
txtconadd.Text.Trim());
cmd.Parameters.AddWithValue("@Telephone2",
txtconph.Text.Trim());
cmd.Parameters.AddWithValue("@party_sname",
cmbparty.SelectedValue);
cmd.Parameters.AddWithValue("@party_fname",
cmbparty.SelectedItem.Text);
```

**Sink:** MemberMaster.aspx.cs:448 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
cmd.Parameters.AddWithValue("@Telephone1",
txtlocph.Text.Trim());
cmd.Parameters.AddWithValue("@Address2",
txtconadd.Text.Trim());
cmd.Parameters.AddWithValue("@Telephone2",
txtconph.Text.Trim());
cmd.Parameters.AddWithValue("@party_sname",
cmbparty.SelectedValue);
cmd.Parameters.AddWithValue("@party_fname",
cmbparty.SelectedItem.Text);
```

## PartyMaster.aspx.cs, line 160 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in PartyMaster.aspx.cs can execute a SQL statement on line 160 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** PartyMaster.aspx.cs:160 System.Web.UI.WebControls.TextBox.get_Text()

```
158                     cmd.Parameters.AddWithValue("@PARTY_FNAME",
SqlDbType.VarChar).Value = txtPartyName.Text.Trim();
159                     cmd.Parameters.AddWithValue("@PARTY_SNAME",
SqlDbType.VarChar).Value = txtShortName.Text.Trim();
160                     cmd.Parameters.AddWithValue("@PARTY_FNAME_H",
SqlDbType.NVarChar).Value = txtHPartyName.Text.Trim();
161                     cmd.Parameters.AddWithValue("@PARTY_SNAME_H",
SqlDbType.NVarChar).Value = txtHShortName.Text.Trim();
162                     cmd.Parameters.AddWithValue("@LSFROM", SqlDbType.SmallInt).Value =
ddlLSFrom.SelectedValue.Trim();
```

**Sink:** PartyMaster.aspx.cs:160 System.Data.Common.DbParameter.set_Value()

```
158                     cmd.Parameters.AddWithValue("@PARTY_FNAME",
SqlDbType.VarChar).Value = txtPartyName.Text.Trim();
159                     cmd.Parameters.AddWithValue("@PARTY_SNAME",
SqlDbType.VarChar).Value = txtShortName.Text.Trim();
160                     cmd.Parameters.AddWithValue("@PARTY_FNAME_H",
SqlDbType.NVarChar).Value = txtHPartyName.Text.Trim();
161                     cmd.Parameters.AddWithValue("@PARTY_SNAME_H",
SqlDbType.NVarChar).Value = txtHShortName.Text.Trim();
162                     cmd.Parameters.AddWithValue("@LSFROM", SqlDbType.SmallInt).Value =
ddlLSFrom.SelectedValue.Trim();
```

### MemberMasterHindi.aspx.cs, line 487 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 487 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:487 System.Web.UI.WebControls.TextBox.get_Text()

```
485                         cmd.Parameters.AddWithValue("@CONST_CODE",
cmbConst.SelectedValue.Trim());
486                         cmd.Parameters.AddWithValue("@const_name_h",
cmbConst.SelectedItem.Text.Trim());
487                         cmd.Parameters.AddWithValue("@mobile2",
TxtMobile2.Text.Trim());
488                         cmd.Parameters.AddWithValue("@mobile3",
TxtMobile3.Text.Trim());
489                         cmd.Parameters.AddWithValue("@mobile4",
TxtMobile4.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:487 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
485                         cmd.Parameters.AddWithValue("@CONST_CODE",
cmbConst.SelectedValue.Trim());
486                         cmd.Parameters.AddWithValue("@const_name_h",
cmbConst.SelectedItem.Text.Trim());
487                         cmd.Parameters.AddWithValue("@mobile2",
TxtMobile2.Text.Trim());
488                         cmd.Parameters.AddWithValue("@mobile3",
TxtMobile3.Text.Trim());
489                         cmd.Parameters.AddWithValue("@mobile4",
TxtMobile4.Text.Trim());
```

### Login.aspx.cs, line 152 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method submit_Click() in Login.aspx.cs can execute a SQL statement on line 152 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** Login.aspx.cs:141 System.Web.UI.WebControls.TextBox.get_Text()

```
139                 string pass = String.Empty;
140                 string pageone = String.Empty;
141                 string uname = UserName.Text.ToString().Trim();
142                 Session["uname"] = uname;
143                 //Get Utype
```

**Sink:** Login.aspx.cs:152 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
150                 cmd3.CommandText = "[dbo].[CMS_LoginMasterSP]";
151                 cmd3.CommandType = CommandType.StoredProcedure;
152                 cmd3.Parameters.AddWithValue("@uname", uname);
153                 cmd3.Parameters.Clear();
154                 SqlDataReader dr3 = cmd3.ExecuteReader();
```

## MemberMasterHindi.aspx.cs, line 513 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 513 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:513 System.Web.UI.WebControls.TextBox.get_Text()

```
511                         cmd.CommandType = CommandType.StoredProcedure;
512                         cmd.Parameters.AddWithValue("@MP_CODE", id.ToString());
513                         cmd.Parameters.AddWithValue("@HMP_INIT",
      txtinitial.Text.Trim());
514                         cmd.Parameters.AddWithValue("@HMP_FNAME",
      txtfname.Text.Trim());
515                         cmd.Parameters.AddWithValue("@HMP_LNAME",
      txtlname.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:513 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
511                         cmd.CommandType = CommandType.StoredProcedure;
512                         cmd.Parameters.AddWithValue("@MP_CODE", id.ToString());
513                         cmd.Parameters.AddWithValue("@HMP_INIT",
      txtinitial.Text.Trim());
514                         cmd.Parameters.AddWithValue("@HMP_FNAME",
      txtfname.Text.Trim());
515                         cmd.Parameters.AddWithValue("@HMP_LNAME",
      txtlname.Text.Trim());
```

## MemberMaster.aspx.cs, line 379 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 379 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:379 System.Web.UI.WebControls.TextBox.get_Text()

```
377                             cmd.Parameters.AddWithValue("@Telephone2",
      txtconph.Text.Trim());
378                             cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
      cmbparty.SelectedValue);
379                             cmd.Parameters.AddWithValue("@mobile",
      TxtMobile.Text.Trim());
380                             cmd.Parameters.AddWithValue("@email1",
      TxteMail1.Text.Trim());
381                             cmd.Parameters.AddWithValue("@email2",
      TxteMail2.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:379 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
377                             cmd.Parameters.AddWithValue("@Telephone2",
      txtconph.Text.Trim());
```

```
378                                    cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
                      cmbparty.SelectedValue);
379                                    cmd.Parameters.AddWithValue("@mobile",
                      TxtMobile.Text.Trim());
380                                    cmd.Parameters.AddWithValue("@email1",
                      TxteMail1.Text.Trim());
381                                    cmd.Parameters.AddWithValue("@email2",
                      TxteMail2.Text.Trim());
```

## MeetingAttendance.aspx.cs, line 467 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method excuteNoquery() in MeetingAttendance.aspx.cs can execute a SQL statement on line 467 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MeetingAttendance.aspx.cs:467 System.Web.UI.WebControls.TextBox.get_Text()

```
465         cmd.Parameters.AddWithValue("@Title", cmbtitle.Text);
466         cmd.Parameters.AddWithValue("@dateofmeet", txtdate1.Text);
467         cmd.Parameters.AddWithValue("@dateofmeet2", txtdate2.Text);
468         cmd.Parameters.AddWithValue("@timeofmeet", txttime.Text);
469         cmd.Parameters.AddWithValue("@cid", cmbcommittee.SelectedValue);
```

**Sink:** MeetingAttendance.aspx.cs:467 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
465         cmd.Parameters.AddWithValue("@Title", cmbtitle.Text);
466         cmd.Parameters.AddWithValue("@dateofmeet", txtdate1.Text);
467         cmd.Parameters.AddWithValue("@dateofmeet2", txtdate2.Text);
468         cmd.Parameters.AddWithValue("@timeofmeet", txttime.Text);
469         cmd.Parameters.AddWithValue("@cid", cmbcommittee.SelectedValue);
```

## MemberMasterHindi.aspx.cs, line 470 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 470 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:470 System.Web.UI.WebControls.TextBox.get_Text()

```
468                                cmd.CommandType = CommandType.StoredProcedure;
469                                cmd.Parameters.AddWithValue("@mpsno", id.ToString());
470                                cmd.Parameters.AddWithValue("@initial_h",
                      txtinitial.Text.Trim());
471                                cmd.Parameters.AddWithValue("@first_name_h",
                      txtfname.Text.Trim());
472                                cmd.Parameters.AddWithValue("@last_name_h",
                      txtlname.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:470 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
468                                cmd.CommandType = CommandType.StoredProcedure;
469                                cmd.Parameters.AddWithValue("@mpsno", id.ToString());
470                                cmd.Parameters.AddWithValue("@initial_h",
                      txtinitial.Text.Trim());
471                                cmd.Parameters.AddWithValue("@first_name_h",
                      txtfname.Text.Trim());
472                                cmd.Parameters.AddWithValue("@last_name_h",
                      txtlname.Text.Trim());
```

## MemberMaster.aspx.cs, line 389 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 389 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:389 System.Web.UI.WebControls.TextBox.get_Text()

```
387                             cmd.Parameters.AddWithValue("@mobile2",
              TxtMobile2.Text.Trim());
388                             cmd.Parameters.AddWithValue("@mobile3",
              TxtMobile3.Text.Trim());
389                             cmd.Parameters.AddWithValue("@mobile4",
              TxtMobile4.Text.Trim());
390                             cmd.Parameters.AddWithValue("@MP_JoinDate",
              txtMPJoinDate.Text.Trim());
391                             con.Open();
```

**Sink:** MemberMaster.aspx.cs:389 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
387                             cmd.Parameters.AddWithValue("@mobile2",
              TxtMobile2.Text.Trim());
388                             cmd.Parameters.AddWithValue("@mobile3",
              TxtMobile3.Text.Trim());
389                             cmd.Parameters.AddWithValue("@mobile4",
              TxtMobile4.Text.Trim());
390                             cmd.Parameters.AddWithValue("@MP_JoinDate",
              txtMPJoinDate.Text.Trim());
391                             con.Open();
```

## MeetingCommittee.aspx.cs, line 283 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method InsertSchedule() in MeetingCommittee.aspx.cs can execute a SQL statement on line 283 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MeetingCommittee.aspx.cs:283 System.Web.UI.WebControls.TextBox.get_Text()

```
281                 cmd.Parameters.Add("@subject", txtpurpose.Text);
282                 cmd.Parameters.Add("@venue", txtvenue.Text);
283                 cmd.Parameters.Add("@remarks", txtremarks.Text);
284                 cmd.Parameters.Add("@TypeOfCommittee", rdoCommitteeType.SelectedValue);
285                 cmd.Parameters.Add("@FileNo", TxtFileNo.Text.Trim());
```

**Sink:** MeetingCommittee.aspx.cs:283 System.Data.SqlClient.SqlParameterCollection.Add()

```
281                 cmd.Parameters.Add("@subject", txtpurpose.Text);
282                 cmd.Parameters.Add("@venue", txtvenue.Text);
283                 cmd.Parameters.Add("@remarks", txtremarks.Text);
284                 cmd.Parameters.Add("@TypeOfCommittee", rdoCommitteeType.SelectedValue);
285                 cmd.Parameters.Add("@FileNo", TxtFileNo.Text.Trim());
```

## MemberMaster.aspx.cs, line 458 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 458 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:458 System.Web.UI.WebControls.TextBox.get_Text()

```
456                         cmd.Parameters.AddWithValue("@CONST_CODE",
              cmbConst.SelectedValue.Trim());
457                         cmd.Parameters.AddWithValue("@CONST_NAME",
              cmbConst.SelectedItem.Text.Trim());
458                         cmd.Parameters.AddWithValue("@mobile2",
              TxtMobile2.Text.Trim());
459                         cmd.Parameters.AddWithValue("@mobile3",
              TxtMobile3.Text.Trim());
```

```
460                                     cmd.Parameters.AddWithValue("@mobile4",
                      TxtMobile4.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:458
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
456                                     cmd.Parameters.AddWithValue("@CONST_CODE",
                      cmbConst.SelectedValue.Trim());
457                                     cmd.Parameters.AddWithValue("@CONST_NAME",
                      cmbConst.SelectedItem.Text.Trim());
458                                     cmd.Parameters.AddWithValue("@mobile2",
                      TxtMobile2.Text.Trim());
459                                     cmd.Parameters.AddWithValue("@mobile3",
                      TxtMobile3.Text.Trim());
460                                     cmd.Parameters.AddWithValue("@mobile4",
                      TxtMobile4.Text.Trim());
```

## MemberMasterHindi.aspx.cs, line 591 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 591 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:591
System.Web.UI.WebControls.TextBox.get_Text()

```
589                                 cmd.Parameters.AddWithValue("@party_sname",
                      cmbparty.SelectedValue);
590                                 cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
591                                 cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
592                                 cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
593                                 cmd.Parameters.AddWithValue("@STATE_CODE",
                      cmbState.SelectedValue.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:591
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
589                                 cmd.Parameters.AddWithValue("@party_sname",
                      cmbparty.SelectedValue);
590                                 cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
591                                 cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
592                                 cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
593                                 cmd.Parameters.AddWithValue("@STATE_CODE",
                      cmbState.SelectedValue.Trim());
```

## MemberMasterHindi.aspx.cs, line 479 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 479 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:479
System.Web.UI.WebControls.TextBox.get_Text()

```
477                                     cmd.Parameters.AddWithValue("@party_sname",
                      cmbparty.SelectedValue);
478                                     cmd.Parameters.AddWithValue("@PARTY_FNAME_H",
                      cmbparty.SelectedItem.Text);
479                                     cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
480                                     cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
481                                     cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:479
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
477                                     cmd.Parameters.AddWithValue("@party_sname",
                      cmbparty.SelectedValue);
478                                     cmd.Parameters.AddWithValue("@PARTY_FNAME_H",
                      cmbparty.SelectedItem.Text);
479                                     cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
480                                     cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
```

```
481                                      cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
```

## MemberMasterHindi.aspx.cs, line 480 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 480 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:480 System.Web.UI.WebControls.TextBox.get_Text()

```
                    cmd.Parameters.AddWithValue("@PARTY_FNAME_H",
cmbparty.SelectedItem.Text);
                    cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
                    cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
                    cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
                    cmd.Parameters.AddWithValue("@last_ls",
DdlHouseNo.SelectedValue.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:480 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
                    cmd.Parameters.AddWithValue("@PARTY_FNAME_H",
cmbparty.SelectedItem.Text);
                    cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
                    cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
                    cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
                    cmd.Parameters.AddWithValue("@last_ls",
DdlHouseNo.SelectedValue.Trim());
```

## PartyMaster.aspx.cs, line 195 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in PartyMaster.aspx.cs can execute a SQL statement on line 195 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** PartyMaster.aspx.cs:195 System.Web.UI.WebControls.TextBox.get_Text()

```
                cmd.Parameters.AddWithValue("@PARTY_FNAME",
SqlDbType.VarChar).Value = txtPartyName.Text.Trim();
                cmd.Parameters.AddWithValue("@PARTY_SNAME",
SqlDbType.VarChar).Value = txtShortName.Text.Trim();
                cmd.Parameters.AddWithValue("@PARTY_FNAME_H",
SqlDbType.NVarChar).Value = txtHPartyName.Text.Trim();
                cmd.Parameters.AddWithValue("@PARTY_SNAME_H",
SqlDbType.NVarChar).Value = txtHShortName.Text.Trim();
                cmd.Parameters.AddWithValue("@LEADER", SqlDbType.VarChar).Value =
txtLeader.Text.Trim();
```

**Sink:** PartyMaster.aspx.cs:195 System.Data.Common.DbParameter.set_Value()

```
                cmd.Parameters.AddWithValue("@PARTY_FNAME",
SqlDbType.VarChar).Value = txtPartyName.Text.Trim();
                cmd.Parameters.AddWithValue("@PARTY_SNAME",
SqlDbType.VarChar).Value = txtShortName.Text.Trim();
                cmd.Parameters.AddWithValue("@PARTY_FNAME_H",
SqlDbType.NVarChar).Value = txtHPartyName.Text.Trim();
                cmd.Parameters.AddWithValue("@PARTY_SNAME_H",
SqlDbType.NVarChar).Value = txtHShortName.Text.Trim();
                cmd.Parameters.AddWithValue("@LEADER", SqlDbType.VarChar).Value =
txtLeader.Text.Trim();
```

## Ministry_details.aspx.cs, line 152 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method Savebtn_Click() in Ministry_details.aspx.cs can execute a SQL statement on line 152 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** Ministry_details.aspx.cs:152 System.Web.UI.WebControls.TextBox.get_Text()

```
150                     cmd.CommandType = CommandType.StoredProcedure;
151                     cmd.Parameters.AddWithValue("@min_code", SqlDbType.SmallInt).Value =
      r;
152                     cmd.Parameters.AddWithValue("@min_name", SqlDbType.VarChar).Value =
      mintxt1.Text.Trim();
153                     cmd.Parameters.AddWithValue("@min_name_h", SqlDbType.VarChar).Value =
      hmintxt1.Text.Trim();
154                     cmd.Parameters.AddWithValue("@min_ab", SqlDbType.VarChar).Value =
      Shortmintxt1.Text.Trim();
```

**Sink:** Ministry_details.aspx.cs:152 System.Data.Common.DbParameter.set_Value()

```
150                     cmd.CommandType = CommandType.StoredProcedure;
151                     cmd.Parameters.AddWithValue("@min_code", SqlDbType.SmallInt).Value =
      r;
152                     cmd.Parameters.AddWithValue("@min_name", SqlDbType.VarChar).Value =
      mintxt1.Text.Trim();
153                     cmd.Parameters.AddWithValue("@min_name_h", SqlDbType.VarChar).Value =
      hmintxt1.Text.Trim();
154                     cmd.Parameters.AddWithValue("@min_ab", SqlDbType.VarChar).Value =
      Shortmintxt1.Text.Trim();
```

## MemberMasterHindi.aspx.cs, line 621 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 621 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:621 System.Web.UI.WebControls.TextBox.get_Text()

```
619                         cmd.Parameters.AddWithValue("@HMP_LNAME", txtlname.Text.Trim());
620                         cmd.Parameters.AddWithValue("@HC_LADDRESS",
      txtlocadd.Text.Trim());
621                         cmd.Parameters.AddWithValue("@Telephone1", txtlocph.Text.Trim());
622                         cmd.Parameters.AddWithValue("@HC_PADDRESS",
      txtconadd.Text.Trim());
623                         cmd.Parameters.AddWithValue("@Telephone2", txtconph.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:621 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
619                         cmd.Parameters.AddWithValue("@HMP_LNAME", txtlname.Text.Trim());
620                         cmd.Parameters.AddWithValue("@HC_LADDRESS",
      txtlocadd.Text.Trim());
621                         cmd.Parameters.AddWithValue("@Telephone1", txtlocph.Text.Trim());
622                         cmd.Parameters.AddWithValue("@HC_PADDRESS",
      txtconadd.Text.Trim());
623                         cmd.Parameters.AddWithValue("@Telephone2", txtconph.Text.Trim());
```

## MemberMaster.aspx.cs, line 459 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 459 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:459 System.Web.UI.WebControls.TextBox.get_Text()

```
457                             cmd.Parameters.AddWithValue("@CONST_NAME",
      cmbConst.SelectedItem.Text.Trim());
458                             cmd.Parameters.AddWithValue("@mobile2",
      TxtMobile2.Text.Trim());
459                             cmd.Parameters.AddWithValue("@mobile3",
      TxtMobile3.Text.Trim());
460                             cmd.Parameters.AddWithValue("@mobile4",
      TxtMobile4.Text.Trim());
461                             cmd.Parameters.AddWithValue("@Status",
      rdobtnStatus.SelectedValue.Trim());
```

**Sink:** MemberMaster.aspx.cs:459
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
457                         cmd.Parameters.AddWithValue("@CONST_NAME",
      cmbConst.SelectedItem.Text.Trim());
458                         cmd.Parameters.AddWithValue("@mobile2",
      TxtMobile2.Text.Trim());
459                         cmd.Parameters.AddWithValue("@mobile3",
      TxtMobile3.Text.Trim());
460                         cmd.Parameters.AddWithValue("@mobile4",
      TxtMobile4.Text.Trim());
461                         cmd.Parameters.AddWithValue("@Status",
      rdobtnStatus.SelectedValue.Trim());
```

## MemberMasterHindi.aspx.cs, line 489 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 489 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:489
System.Web.UI.WebControls.TextBox.get_Text()

```
487                         cmd.Parameters.AddWithValue("@mobile2",
      TxtMobile2.Text.Trim());
488                         cmd.Parameters.AddWithValue("@mobile3",
      TxtMobile3.Text.Trim());
489                         cmd.Parameters.AddWithValue("@mobile4",
      TxtMobile4.Text.Trim());
490                         cmd.Parameters.AddWithValue("@MP_JoinDate",
      txtMPJoinDate.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:489
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
487                         cmd.Parameters.AddWithValue("@mobile2",
      TxtMobile2.Text.Trim());
488                         cmd.Parameters.AddWithValue("@mobile3",
      TxtMobile3.Text.Trim());
489                         cmd.Parameters.AddWithValue("@mobile4",
      TxtMobile4.Text.Trim());
490                         cmd.Parameters.AddWithValue("@MP_JoinDate",
      txtMPJoinDate.Text.Trim());
```

## MemberCommittee.aspx.cs, line 506 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method UpdateDesignation() in MemberCommittee.aspx.cs can execute a SQL statement on line 506 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberCommittee.aspx.cs:506
System.Web.UI.WebControls.TextBox.get_Text()

```
504               cmd.Parameters.AddWithValue("@DesignationText", DesignationText);
505               cmd.Parameters.AddWithValue("@mpcode",
      Convert.ToInt32(cmbmp.SelectedValue));
506               cmd.Parameters.AddWithValue("@Designation",  txtDesg.Text.Trim());
507               cmd.ExecuteNonQuery();
508               cmd.Dispose();
```

**Sink:** MemberCommittee.aspx.cs:506
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
504               cmd.Parameters.AddWithValue("@DesignationText", DesignationText);
505               cmd.Parameters.AddWithValue("@mpcode",
      Convert.ToInt32(cmbmp.SelectedValue));
506               cmd.Parameters.AddWithValue("@Designation",  txtDesg.Text.Trim());
507               cmd.ExecuteNonQuery();
508               cmd.Dispose();
```

## MemberMasterHindi.aspx.cs, line 622 (Access Control: Database)

**Fortify Priority:** High **Folder** High
**Kingdom:** Security Features

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 622 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:622
System.Web.UI.WebControls.TextBox.get_Text()

```
620                     cmd.Parameters.AddWithValue("@HC_LADDRESS",
      txtlocadd.Text.Trim());
621                     cmd.Parameters.AddWithValue("@Telephone1", txtlocph.Text.Trim());
622                     cmd.Parameters.AddWithValue("@HC_PADDRESS",
      txtconadd.Text.Trim());
623                     cmd.Parameters.AddWithValue("@Telephone2", txtconph.Text.Trim());
624                     cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
      cmbparty.SelectedValue);
```

**Sink:** MemberMasterHindi.aspx.cs:622
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
620                     cmd.Parameters.AddWithValue("@HC_LADDRESS",
      txtlocadd.Text.Trim());
621                     cmd.Parameters.AddWithValue("@Telephone1", txtlocph.Text.Trim());
622                     cmd.Parameters.AddWithValue("@HC_PADDRESS",
      txtconadd.Text.Trim());
623                     cmd.Parameters.AddWithValue("@Telephone2", txtconph.Text.Trim());
624                     cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
      cmbparty.SelectedValue);
```

## MemberMasterHindi.aspx.cs, line 532 (Access Control: Database)

**Fortify Priority:** High **Folder** High
**Kingdom:** Security Features

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 532 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:532
System.Web.UI.WebControls.TextBox.get_Text()

```
530                         cmd.Parameters.AddWithValue("@mobile3",
      TxtMobile3.Text.Trim());
531                         cmd.Parameters.AddWithValue("@mobile4",
      TxtMobile4.Text.Trim());
532                         cmd.Parameters.AddWithValue("@MP_JoinDate",
      txtMPJoinDate.Text.Trim());
533                         con.Open();
534                         i = cmd.ExecuteNonQuery();
```

**Sink:** MemberMasterHindi.aspx.cs:532
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
530                         cmd.Parameters.AddWithValue("@mobile3",
      TxtMobile3.Text.Trim());
531                         cmd.Parameters.AddWithValue("@mobile4",
      TxtMobile4.Text.Trim());
532                         cmd.Parameters.AddWithValue("@MP_JoinDate",
      txtMPJoinDate.Text.Trim());
533                         con.Open();
534                         i = cmd.ExecuteNonQuery();
```

## Ministry_details.aspx.cs, line 116 (Access Control: Database)

**Fortify Priority:** High **Folder** High
**Kingdom:** Security Features

**Abstract:** Without proper access control, the method Savebtn_Click() in Ministry_details.aspx.cs can execute a SQL statement on line 116 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** Ministry_details.aspx.cs:116 System.Web.UI.WebControls.TextBox.get_Text()

```
114                     cmd = new SqlCommand("MinistryUpdate1", conn);
115                     cmd.CommandType = CommandType.StoredProcedure;
116                     cmd.Parameters.AddWithValue("@min_name", SqlDbType.VarChar).Value =
mintxt2.Text.Trim();
117                     cmd.Parameters.AddWithValue("@min_name_h", SqlDbType.VarChar).Value =
hmintxt2.Text.Trim();
118                     cmd.Parameters.AddWithValue("@min_ab", SqlDbType.VarChar).Value =
Shortmintxt2.Text.Trim();
```

**Sink:** Ministry_details.aspx.cs:116 System.Data.Common.DbParameter.set_Value()

```
114                     cmd = new SqlCommand("MinistryUpdate1", conn);
115                     cmd.CommandType = CommandType.StoredProcedure;
116                     cmd.Parameters.AddWithValue("@min_name", SqlDbType.VarChar).Value =
mintxt2.Text.Trim();
117                     cmd.Parameters.AddWithValue("@min_name_h", SqlDbType.VarChar).Value =
hmintxt2.Text.Trim();
118                     cmd.Parameters.AddWithValue("@min_ab", SqlDbType.VarChar).Value =
Shortmintxt2.Text.Trim();
```

## MemberMasterHindi.aspx.cs, line 620 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 620 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:620 System.Web.UI.WebControls.TextBox.get_Text()

```
618                         cmd.Parameters.AddWithValue("@HMP_FNAME", txtfname.Text.Trim());
619                         cmd.Parameters.AddWithValue("@HMP_LNAME", txtlname.Text.Trim());
620                         cmd.Parameters.AddWithValue("@HC_LADDRESS",
txtlocadd.Text.Trim());
621                         cmd.Parameters.AddWithValue("@Telephone1", txtlocph.Text.Trim());
622                         cmd.Parameters.AddWithValue("@HC_PADDRESS",
txtconadd.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:620 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
618                         cmd.Parameters.AddWithValue("@HMP_FNAME", txtfname.Text.Trim());
619                         cmd.Parameters.AddWithValue("@HMP_LNAME", txtlname.Text.Trim());
620                         cmd.Parameters.AddWithValue("@HC_LADDRESS",
txtlocadd.Text.Trim());
621                         cmd.Parameters.AddWithValue("@Telephone1", txtlocph.Text.Trim());
622                         cmd.Parameters.AddWithValue("@HC_PADDRESS",
txtconadd.Text.Trim());
```

## MeetingCommittee.aspx.cs, line 314 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method UpdateSchedule() in MeetingCommittee.aspx.cs can execute a SQL statement on line 314 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MeetingCommittee.aspx.cs:314 System.Web.UI.WebControls.TextBox.get_Text()

```
312                 cmd.CommandType = CommandType.StoredProcedure;
313
314                 cmd.Parameters.Add("@title", txttitle.Text);
315                 cmd.Parameters.Add("@dateofmeet", txtdate1.Text);
316                 cmd.Parameters.Add("@dateofmeet2", txtdate2.Text);
```

**Sink:** MeetingCommittee.aspx.cs:314 System.Data.SqlClient.SqlParameterCollection.Add()

```
312                 cmd.CommandType = CommandType.StoredProcedure;
```

```
313
314                        cmd.Parameters.Add("@title", txttitle.Text);
315                        cmd.Parameters.Add("@dateofmeet", txtdate1.Text);
316                        cmd.Parameters.Add("@dateofmeet2", txtdate2.Text);
```

## MemberMasterHindi.aspx.cs, line 514 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 514 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:514
System.Web.UI.WebControls.TextBox.get_Text()

```
512                        cmd.Parameters.AddWithValue("@MP_CODE", id.ToString());
513                        cmd.Parameters.AddWithValue("@HMP_INIT",
      txtinitial.Text.Trim());
514                        cmd.Parameters.AddWithValue("@HMP_FNAME",
      txtfname.Text.Trim());
515                        cmd.Parameters.AddWithValue("@HMP_LNAME",
      txtlname.Text.Trim());
516                        cmd.Parameters.AddWithValue("@HC_LADDRESS",
      txtlocadd.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:514
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
512                        cmd.Parameters.AddWithValue("@MP_CODE", id.ToString());
513                        cmd.Parameters.AddWithValue("@HMP_INIT",
      txtinitial.Text.Trim());
514                        cmd.Parameters.AddWithValue("@HMP_FNAME",
      txtfname.Text.Trim());
515                        cmd.Parameters.AddWithValue("@HMP_LNAME",
      txtlname.Text.Trim());
516                        cmd.Parameters.AddWithValue("@HC_LADDRESS",
      txtlocadd.Text.Trim());
```

## depatment_detail.aspx.cs, line 144 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method Savebtn_Click() in depatment_detail.aspx.cs can execute a SQL statement on line 144 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** depatment_detail.aspx.cs:144
System.Web.UI.WebControls.TextBox.get_Text()

```
142                  cmd = new SqlCommand("departmentUpdate", conn);
143                 cmd.CommandType = CommandType.StoredProcedure;
144                 cmd.Parameters.AddWithValue("@dep_name", dep2.Text);
145                 cmd.Parameters.AddWithValue("@hdep_name", hdep2.Text);
146                 cmd.Parameters.AddWithValue("@dep_code",  ViewState["dep_code"]);
```

**Sink:** depatment_detail.aspx.cs:144
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
142                  cmd = new SqlCommand("departmentUpdate", conn);
143                 cmd.CommandType = CommandType.StoredProcedure;
144                 cmd.Parameters.AddWithValue("@dep_name", dep2.Text);
145                 cmd.Parameters.AddWithValue("@hdep_name", hdep2.Text);
146                 cmd.Parameters.AddWithValue("@dep_code",  ViewState["dep_code"]);
```

## MemberCommittee.aspx.cs, line 540 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method InsertDesignation() in MemberCommittee.aspx.cs can execute a SQL statement on line 540 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberCommittee.aspx.cs:540 System.Web.UI.WebControls.TextBox.get_Text()

```
538                 cmd.Parameters.Add("@MinId", cmbminname.SelectedValue);
539                 cmd.Parameters.Add("@DeptId", cmbdepat.SelectedValue);
540                 cmd.Parameters.Add("@BodiesName", txtBodieName.Text.Trim());
541                 cmd.Parameters.Add("@radio", rdoCommitteeType.SelectedValue);
542                 cmd.ExecuteNonQuery();
```

**Sink:** MemberCommittee.aspx.cs:540 System.Data.SqlClient.SqlParameterCollection.Add()

```
538                 cmd.Parameters.Add("@MinId", cmbminname.SelectedValue);
539                 cmd.Parameters.Add("@DeptId", cmbdepat.SelectedValue);
540                 cmd.Parameters.Add("@BodiesName", txtBodieName.Text.Trim());
541                 cmd.Parameters.Add("@radio", rdoCommitteeType.SelectedValue);
542                 cmd.ExecuteNonQuery();
```

## MemberMaster.aspx.cs, line 371 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 371 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:371 System.Web.UI.WebControls.TextBox.get_Text()

```
369                                 cmd.CommandType = CommandType.StoredProcedure;
370                                 cmd.Parameters.AddWithValue("@MP_CODE", id.ToString());
371                                 cmd.Parameters.AddWithValue("@MP_INIT",
          txtinitial.Text.Trim());
372                                 cmd.Parameters.AddWithValue("@MP_FNAME",
          txtfname.Text.Trim());
373                                 cmd.Parameters.AddWithValue("@MP_LNAME",
          txtlname.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:371 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
369                                 cmd.CommandType = CommandType.StoredProcedure;
370                                 cmd.Parameters.AddWithValue("@MP_CODE", id.ToString());
371                                 cmd.Parameters.AddWithValue("@MP_INIT",
          txtinitial.Text.Trim());
372                                 cmd.Parameters.AddWithValue("@MP_FNAME",
          txtfname.Text.Trim());
373                                 cmd.Parameters.AddWithValue("@MP_LNAME",
          txtlname.Text.Trim());
```

## MemberMaster.aspx.cs, line 480 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 480 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:480 System.Web.UI.WebControls.TextBox.get_Text()

```
478                             cmd.Parameters.AddWithValue("@MP_CODE", ViewState["Key"]);
479                             cmd.Parameters.AddWithValue("@MP_INIT",
          txtinitial.Text.Trim());
480                             cmd.Parameters.AddWithValue("@MP_FNAME",
          txtfname.Text.Trim());
481                             cmd.Parameters.AddWithValue("@MP_LNAME",
          txtlname.Text.Trim());
482                             cmd.Parameters.AddWithValue("@C_LADDRESS",
          txtlocadd.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:480
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
478                             cmd.Parameters.AddWithValue("@MP_CODE", ViewState["Key"]);
479                             cmd.Parameters.AddWithValue("@MP_INIT",
                    txtinitial.Text.Trim());
480                             cmd.Parameters.AddWithValue("@MP_FNAME",
                    txtfname.Text.Trim());
481                             cmd.Parameters.AddWithValue("@MP_LNAME",
                    txtlname.Text.Trim());
482                             cmd.Parameters.AddWithValue("@C_LADDRESS",
                    txtlocadd.Text.Trim());
```

## MemberMasterHindi.aspx.cs, line 473 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 473 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:473
System.Web.UI.WebControls.TextBox.get_Text()

```
471                             cmd.Parameters.AddWithValue("@first_name_h",
                    txtfname.Text.Trim());
472                             cmd.Parameters.AddWithValue("@last_name_h",
                    txtlname.Text.Trim());
473                             cmd.Parameters.AddWithValue("@H_Address1",
                    txtlocadd.Text.Trim());
474                             cmd.Parameters.AddWithValue("@Telephone1",
                    txtlocph.Text.Trim());
475                             cmd.Parameters.AddWithValue("@H_Address2",
                    txtconadd.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:473
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
471                             cmd.Parameters.AddWithValue("@first_name_h",
                    txtfname.Text.Trim());
472                             cmd.Parameters.AddWithValue("@last_name_h",
                    txtlname.Text.Trim());
473                             cmd.Parameters.AddWithValue("@H_Address1",
                    txtlocadd.Text.Trim());
474                             cmd.Parameters.AddWithValue("@Telephone1",
                    txtlocph.Text.Trim());
475                             cmd.Parameters.AddWithValue("@H_Address2",
                    txtconadd.Text.Trim());
```

## MemberMaster.aspx.cs, line 331 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 331 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:331 System.Web.UI.WebControls.TextBox.get_Text()

```
329                                 cmd.Parameters.AddWithValue("@mpsno", id.ToString());
330                                 cmd.Parameters.AddWithValue("@Initial",
                    txtinitial.Text.Trim());
331                                 cmd.Parameters.AddWithValue("@first_name",
                    txtfname.Text.Trim());
332                                 cmd.Parameters.AddWithValue("@last_name",
                    txtlname.Text.Trim());
333                                 cmd.Parameters.AddWithValue("@Address1",
                    txtlocadd.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:331
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
329                                 cmd.Parameters.AddWithValue("@mpsno", id.ToString());
330                                 cmd.Parameters.AddWithValue("@Initial",
                    txtinitial.Text.Trim());
```

```
331                                          cmd.Parameters.AddWithValue("@first_name",
                     txtfname.Text.Trim());
332                                          cmd.Parameters.AddWithValue("@last_name",
                     txtlname.Text.Trim());
333                                          cmd.Parameters.AddWithValue("@Address1",
                     txtlocadd.Text.Trim());
```

## MemberMaster.aspx.cs, line 339 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 339 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:339 System.Web.UI.WebControls.TextBox.get_Text()

```
337                                          cmd.Parameters.AddWithValue("@party_sname",
                     cmbparty.SelectedValue);
338                                          cmd.Parameters.AddWithValue("@party_fname",
                     cmbparty.SelectedItem.Text);
339                                          cmd.Parameters.AddWithValue("@mobile",
                     TxtMobile.Text.Trim());
340                                          cmd.Parameters.AddWithValue("@email1",
                     TxteMail1.Text.Trim());
341                                          cmd.Parameters.AddWithValue("@email2",
                     TxteMail2.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:339 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
337                                          cmd.Parameters.AddWithValue("@party_sname",
                     cmbparty.SelectedValue);
338                                          cmd.Parameters.AddWithValue("@party_fname",
                     cmbparty.SelectedItem.Text);
339                                          cmd.Parameters.AddWithValue("@mobile",
                     TxtMobile.Text.Trim());
340                                          cmd.Parameters.AddWithValue("@email1",
                     TxteMail1.Text.Trim());
341                                          cmd.Parameters.AddWithValue("@email2",
                     TxteMail2.Text.Trim());
```

## depatment_detail.aspx.cs, line 176 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method Savebtn_Click() in depatment_detail.aspx.cs can execute a SQL statement on line 176 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** depatment_detail.aspx.cs:176 System.Web.UI.WebControls.TextBox.get_Text()

```
174                            cmd.CommandType = CommandType.StoredProcedure;
175                            cmd.Parameters.AddWithValue("@dep_name", dep1.Text);
176                            cmd.Parameters.AddWithValue("@hdep_name", hdep1.Text);
177                            cmd.Parameters.AddWithValue("@mnstry_code", minstry_id);
178                            cmd.Parameters.AddWithValue("@dep_code", r);
```

**Sink:** depatment_detail.aspx.cs:176 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
174                            cmd.CommandType = CommandType.StoredProcedure;
175                            cmd.Parameters.AddWithValue("@dep_name", dep1.Text);
176                            cmd.Parameters.AddWithValue("@hdep_name", hdep1.Text);
177                            cmd.Parameters.AddWithValue("@mnstry_code", minstry_id);
178                            cmd.Parameters.AddWithValue("@dep_code", r);
```

## MemberMasterHindi.aspx.cs, line 475 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 475 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:475 System.Web.UI.WebControls.TextBox.get_Text()

```
473         cmd.Parameters.AddWithValue("@H_Address1", txtlocadd.Text.Trim());
474         cmd.Parameters.AddWithValue("@Telephone1", txtlocph.Text.Trim());
475         cmd.Parameters.AddWithValue("@H_Address2", txtconadd.Text.Trim());
476         cmd.Parameters.AddWithValue("@Telephone2", txtconph.Text.Trim());
477         cmd.Parameters.AddWithValue("@party_sname", cmbparty.SelectedValue);
```

**Sink:** MemberMasterHindi.aspx.cs:475 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
473         cmd.Parameters.AddWithValue("@H_Address1", txtlocadd.Text.Trim());
474         cmd.Parameters.AddWithValue("@Telephone1", txtlocph.Text.Trim());
475         cmd.Parameters.AddWithValue("@H_Address2", txtconadd.Text.Trim());
476         cmd.Parameters.AddWithValue("@Telephone2", txtconph.Text.Trim());
477         cmd.Parameters.AddWithValue("@party_sname", cmbparty.SelectedValue);
```

## MemberMaster.aspx.cs, line 380 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 380 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:380 System.Web.UI.WebControls.TextBox.get_Text()

```
378             cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE", cmbparty.SelectedValue);
379             cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
380             cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
381             cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
382             cmd.Parameters.AddWithValue("@MP_CURRENT", 1);
```

**Sink:** MemberMaster.aspx.cs:380 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
378             cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE", cmbparty.SelectedValue);
379             cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
380             cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
381             cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
382             cmd.Parameters.AddWithValue("@MP_CURRENT", 1);
```

## MemberMaster.aspx.cs, line 347 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 347 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:347 System.Web.UI.WebControls.TextBox.get_Text()

```
345                                                     cmd.Parameters.AddWithValue("@CONST_CODE",
                     cmbConst.SelectedValue.Trim());
346                                                     cmd.Parameters.AddWithValue("@CONST_NAME",
                     cmbConst.SelectedItem.Text.Trim());
347                                                     cmd.Parameters.AddWithValue("@mobile2",
                     TxtMobile2.Text.Trim());
348                                                     cmd.Parameters.AddWithValue("@mobile3",
                     TxtMobile3.Text.Trim());
349                                                     cmd.Parameters.AddWithValue("@mobile4",
                     TxtMobile4.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:347
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
345                                                     cmd.Parameters.AddWithValue("@CONST_CODE",
                     cmbConst.SelectedValue.Trim());
346                                                     cmd.Parameters.AddWithValue("@CONST_NAME",
                     cmbConst.SelectedItem.Text.Trim());
347                                                     cmd.Parameters.AddWithValue("@mobile2",
                     TxtMobile2.Text.Trim());
348                                                     cmd.Parameters.AddWithValue("@mobile3",
                     TxtMobile3.Text.Trim());
349                                                     cmd.Parameters.AddWithValue("@mobile4",
                     TxtMobile4.Text.Trim());
```

## CommitteeMaster.aspx.cs, line 138 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in CommitteeMaster.aspx.cs can execute a SQL statement on line 138 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** CommitteeMaster.aspx.cs:138
System.Web.UI.WebControls.TextBox.get_Text()

```
136                                         cmd.Parameters.AddWithValue("@Cid", SqlDbType.Int).Value =
                     id.ToString();
137                                         cmd.Parameters.AddWithValue("@Cname", SqlDbType.VarChar).Value =
                     txtCommName.Text.Trim();
138                                         cmd.Parameters.AddWithValue("@Hcname", SqlDbType.VarChar).Value =
                     txtHCommName.Text.Trim();
139                                         cmd.Parameters.AddWithValue("@srno", SqlDbType.Int).Value =
                     sno.ToString();
140                                         cmd.Parameters.AddWithValue("@MaxMeetingNo", SqlDbType.Int).Value
                     = 0;
```

**Sink:** CommitteeMaster.aspx.cs:138
System.Data.Common.DbParameter.set_Value()

```
136                                         cmd.Parameters.AddWithValue("@Cid", SqlDbType.Int).Value =
                     id.ToString();
137                                         cmd.Parameters.AddWithValue("@Cname", SqlDbType.VarChar).Value =
                     txtCommName.Text.Trim();
138                                         cmd.Parameters.AddWithValue("@Hcname", SqlDbType.VarChar).Value =
                     txtHCommName.Text.Trim();
139                                         cmd.Parameters.AddWithValue("@srno", SqlDbType.Int).Value =
                     sno.ToString();
140                                         cmd.Parameters.AddWithValue("@MaxMeetingNo", SqlDbType.Int).Value
                     = 0;
```

## MeetingCommittee.aspx.cs, line 271 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method InsertSchedule() in MeetingCommittee.aspx.cs can execute a SQL statement on line 271 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MeetingCommittee.aspx.cs:271
System.Web.UI.WebControls.TextBox.get_Text()

```
269                             cmd.Parameters.Add("@title", txttitle.Text);
270                             cmd.Parameters.Add("@dateofmeet", txtdate1.Text);
271                             cmd.Parameters.Add("@dateofmeet2", txtdate2.Text);
```

```
272                     if(timchk == "Y")
273                     {
```

**Sink:** MeetingCommittee.aspx.cs:271
System.Data.SqlClient.SqlParameterCollection.Add()

```
269                     cmd.Parameters.Add("@title", txttitle.Text);
270                     cmd.Parameters.Add("@dateofmeet", txtdate1.Text);
271                     cmd.Parameters.Add("@dateofmeet2", txtdate2.Text);
272                     if(timchk == "Y")
273                     {
```

## MemberMasterHindi.aspx.cs, line 598 (Access Control: Database)

**Fortify Priority:** High **Folder** High
**Kingdom:** Security Features

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 598 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:598
System.Web.UI.WebControls.TextBox.get_Text()

```
596                         cmd.Parameters.AddWithValue("@const_name_h",
            cmbConst.SelectedItem.Text.Trim());
597                         cmd.Parameters.AddWithValue("@mobile2", TxtMobile2.Text.Trim());
598                         cmd.Parameters.AddWithValue("@mobile3", TxtMobile3.Text.Trim());
599                         cmd.Parameters.AddWithValue("@mobile4", TxtMobile4.Text.Trim());
600                         cmd.Parameters.AddWithValue("@Status",
            rdobtnStatus.SelectedValue.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:598
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
596                         cmd.Parameters.AddWithValue("@const_name_h",
            cmbConst.SelectedItem.Text.Trim());
597                         cmd.Parameters.AddWithValue("@mobile2", TxtMobile2.Text.Trim());
598                         cmd.Parameters.AddWithValue("@mobile3", TxtMobile3.Text.Trim());
599                         cmd.Parameters.AddWithValue("@mobile4", TxtMobile4.Text.Trim());
600                         cmd.Parameters.AddWithValue("@Status",
            rdobtnStatus.SelectedValue.Trim());
```

## MemberMasterHindi.aspx.cs, line 521 (Access Control: Database)

**Fortify Priority:** High **Folder** High
**Kingdom:** Security Features

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 521 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:521
System.Web.UI.WebControls.TextBox.get_Text()

```
519                             cmd.Parameters.AddWithValue("@Telephone2",
            txtconph.Text.Trim());
520                             cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
            cmbparty.SelectedValue);
521                             cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
522                             cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
523                             cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:521
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
519                             cmd.Parameters.AddWithValue("@Telephone2",
            txtconph.Text.Trim());
520                             cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
            cmbparty.SelectedValue);
521                             cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
522                             cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
523                             cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
```

## MemberMaster.aspx.cs, line 334 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 334 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:334 System.Web.UI.WebControls.TextBox.get_Text()

```
332                         cmd.Parameters.AddWithValue("@last_name",
    txtlname.Text.Trim());
333                         cmd.Parameters.AddWithValue("@Address1",
    txtlocadd.Text.Trim());
334                         cmd.Parameters.AddWithValue("@Telephone1",
    txtlocph.Text.Trim());
335                         cmd.Parameters.AddWithValue("@Address2",
    txtconadd.Text.Trim());
336                         cmd.Parameters.AddWithValue("@Telephone2",
    txtconph.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:334 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
332                         cmd.Parameters.AddWithValue("@last_name",
    txtlname.Text.Trim());
333                         cmd.Parameters.AddWithValue("@Address1",
    txtlocadd.Text.Trim());
334                         cmd.Parameters.AddWithValue("@Telephone1",
    txtlocph.Text.Trim());
335                         cmd.Parameters.AddWithValue("@Address2",
    txtconadd.Text.Trim());
336                         cmd.Parameters.AddWithValue("@Telephone2",
    txtconph.Text.Trim());
```

## MemberMaster.aspx.cs, line 335 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 335 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:335 System.Web.UI.WebControls.TextBox.get_Text()

```
333                         cmd.Parameters.AddWithValue("@Address1",
    txtlocadd.Text.Trim());
334                         cmd.Parameters.AddWithValue("@Telephone1",
    txtlocph.Text.Trim());
335                         cmd.Parameters.AddWithValue("@Address2",
    txtconadd.Text.Trim());
336                         cmd.Parameters.AddWithValue("@Telephone2",
    txtconph.Text.Trim());
337                         cmd.Parameters.AddWithValue("@party_sname",
    cmbparty.SelectedValue);
```

**Sink:** MemberMaster.aspx.cs:335 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
333                         cmd.Parameters.AddWithValue("@Address1",
    txtlocadd.Text.Trim());
334                         cmd.Parameters.AddWithValue("@Telephone1",
    txtlocph.Text.Trim());
335                         cmd.Parameters.AddWithValue("@Address2",
    txtconadd.Text.Trim());
336                         cmd.Parameters.AddWithValue("@Telephone2",
    txtconph.Text.Trim());
337                         cmd.Parameters.AddWithValue("@party_sname",
    cmbparty.SelectedValue);
```

## Login.aspx.cs, line 217 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

| | |
|---|---|
| **Abstract:** | Without proper access control, the method submit_Click() in Login.aspx.cs can execute a SQL statement on line 217 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records. |

**Source:** Login.aspx.cs:141 System.Web.UI.WebControls.TextBox.get_Text()

```
                                string pass = String.Empty;
                                string pageone = String.Empty;
                                string uname = UserName.Text.ToString().Trim();
                                Session["uname"] = uname;
                                //Get Utype
```

**Sink:** Login.aspx.cs:217 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
                                    cmd2.CommandText = "[dbo].[CMS_LoginMasterUserSP]";
                                    cmd2.CommandType = CommandType.StoredProcedure;
                                    cmd2.Parameters.AddWithValue("@uname", uname);
                                    cmd2.Parameters.AddWithValue("@UType", utype);
                                    SqlDataReader dr2 = cmd2.ExecuteReader();
```

## MemberMaster.aspx.cs, line 483 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

| | |
|---|---|
| **Abstract:** | Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 483 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records. |

**Source:** MemberMaster.aspx.cs:483 System.Web.UI.WebControls.TextBox.get_Text()

```
                        cmd.Parameters.AddWithValue("@MP_LNAME", 
txtlname.Text.Trim());
                        cmd.Parameters.AddWithValue("@C_LADDRESS", 
txtlocadd.Text.Trim());
                        cmd.Parameters.AddWithValue("@Telephone1", 
txtlocph.Text.Trim());
                        cmd.Parameters.AddWithValue("@C_PADDRESS", 
txtconadd.Text.Trim());
                        cmd.Parameters.AddWithValue("@Telephone2", 
txtconph.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:483 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
                        cmd.Parameters.AddWithValue("@MP_LNAME", 
txtlname.Text.Trim());
                        cmd.Parameters.AddWithValue("@C_LADDRESS", 
txtlocadd.Text.Trim());
                        cmd.Parameters.AddWithValue("@Telephone1", 
txtlocph.Text.Trim());
                        cmd.Parameters.AddWithValue("@C_PADDRESS", 
txtconadd.Text.Trim());
                        cmd.Parameters.AddWithValue("@Telephone2", 
txtconph.Text.Trim());
```

## MemberMasterHindi.aspx.cs, line 633 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

| | |
|---|---|
| **Abstract:** | Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 633 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records. |

**Source:** MemberMasterHindi.aspx.cs:633 System.Web.UI.WebControls.TextBox.get_Text()

```
                    cmd.Parameters.AddWithValue("@const_name_h", 
cmbConst.SelectedItem.Text.Trim());
                    cmd.Parameters.AddWithValue("@mobile2", TxtMobile2.Text.Trim());
                    cmd.Parameters.AddWithValue("@mobile3", TxtMobile3.Text.Trim());
                    cmd.Parameters.AddWithValue("@mobile4", TxtMobile4.Text.Trim());
                    cmd.Parameters.AddWithValue("@Status", 
rdobtnStatus.SelectedValue.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:633 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
                    cmd.Parameters.AddWithValue("@const_name_h",
cmbConst.SelectedItem.Text.Trim());
                    cmd.Parameters.AddWithValue("@mobile2", TxtMobile2.Text.Trim());
                    cmd.Parameters.AddWithValue("@mobile3", TxtMobile3.Text.Trim());
                    cmd.Parameters.AddWithValue("@mobile4", TxtMobile4.Text.Trim());
                    cmd.Parameters.AddWithValue("@Status",
rdobtnStatus.SelectedValue.Trim());
```

## MemberMaster.aspx.cs, line 330 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 330 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:330 System.Web.UI.WebControls.TextBox.get_Text()

```
                            cmd.CommandType = CommandType.StoredProcedure;
                            cmd.Parameters.AddWithValue("@mpsno", id.ToString());
                            cmd.Parameters.AddWithValue("@Initial",
txtinitial.Text.Trim());
                            cmd.Parameters.AddWithValue("@first_name",
txtfname.Text.Trim());
                            cmd.Parameters.AddWithValue("@last_name",
txtlname.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:330 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
                            cmd.CommandType = CommandType.StoredProcedure;
                            cmd.Parameters.AddWithValue("@mpsno", id.ToString());
                            cmd.Parameters.AddWithValue("@Initial",
txtinitial.Text.Trim());
                            cmd.Parameters.AddWithValue("@first_name",
txtfname.Text.Trim());
                            cmd.Parameters.AddWithValue("@last_name",
txtlname.Text.Trim());
```

## MemberMaster.aspx.cs, line 482 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 482 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:482 System.Web.UI.WebControls.TextBox.get_Text()

```
                        cmd.Parameters.AddWithValue("@MP_FNAME",
txtfname.Text.Trim());
                        cmd.Parameters.AddWithValue("@MP_LNAME",
txtlname.Text.Trim());
                        cmd.Parameters.AddWithValue("@C_LADDRESS",
txtlocadd.Text.Trim());
                        cmd.Parameters.AddWithValue("@Telephone1",
txtlocph.Text.Trim());
                        cmd.Parameters.AddWithValue("@C_PADDRESS",
txtconadd.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:482 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
                        cmd.Parameters.AddWithValue("@MP_FNAME",
txtfname.Text.Trim());
                        cmd.Parameters.AddWithValue("@MP_LNAME",
txtlname.Text.Trim());
                        cmd.Parameters.AddWithValue("@C_LADDRESS",
txtlocadd.Text.Trim());
                        cmd.Parameters.AddWithValue("@Telephone1",
txtlocph.Text.Trim());
```

```
484                                   cmd.Parameters.AddWithValue("@C_PADDRESS",
                      txtconadd.Text.Trim());
```

## MemberMaster.aspx.cs, line 375 (Access Control: Database)

**Fortify Priority:** High **Folder** High
**Kingdom:** Security Features

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 375 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:375 System.Web.UI.WebControls.TextBox.get_Text()

```
373                                      cmd.Parameters.AddWithValue("@MP_LNAME",
                      txtlname.Text.Trim());
374                                      cmd.Parameters.AddWithValue("@C_LADDRESS",
                      txtlocadd.Text.Trim());
375                                      cmd.Parameters.AddWithValue("@Telephone1",
                      txtlocph.Text.Trim());
376                                      cmd.Parameters.AddWithValue("@C_PADDRESS",
                      txtconadd.Text.Trim());
377                                      cmd.Parameters.AddWithValue("@Telephone2",
                      txtconph.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:375 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
373                                      cmd.Parameters.AddWithValue("@MP_LNAME",
                      txtlname.Text.Trim());
374                                      cmd.Parameters.AddWithValue("@C_LADDRESS",
                      txtlocadd.Text.Trim());
375                                      cmd.Parameters.AddWithValue("@Telephone1",
                      txtlocph.Text.Trim());
376                                      cmd.Parameters.AddWithValue("@C_PADDRESS",
                      txtconadd.Text.Trim());
377                                      cmd.Parameters.AddWithValue("@Telephone2",
                      txtconph.Text.Trim());
```

## MemberMaster.aspx.cs, line 487 (Access Control: Database)

**Fortify Priority:** High **Folder** High
**Kingdom:** Security Features

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 487 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:487 System.Web.UI.WebControls.TextBox.get_Text()

```
485                                   cmd.Parameters.AddWithValue("@Telephone2",
                      txtconph.Text.Trim());
486                                   cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
                      cmbparty.SelectedValue);
487                                   cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
488                                   cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
489                                   cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:487 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
485                                   cmd.Parameters.AddWithValue("@Telephone2",
                      txtconph.Text.Trim());
486                                   cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
                      cmbparty.SelectedValue);
487                                   cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
488                                   cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
489                                   cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
```

## MemberMasterHindi.aspx.cs, line 518 (Access Control: Database)

**Fortify Priority:** High **Folder** High
**Kingdom:** Security Features

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 518 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:518 System.Web.UI.WebControls.TextBox.get_Text()

```
516                         cmd.Parameters.AddWithValue("@HC_LADDRESS",
txtlocadd.Text.Trim());
517                         cmd.Parameters.AddWithValue("@Telephone1",
txtlocph.Text.Trim());
518                         cmd.Parameters.AddWithValue("@HC_PADDRESS",
txtconadd.Text.Trim());
519                         cmd.Parameters.AddWithValue("@Telephone2",
txtconph.Text.Trim());
520                         cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
cmbparty.SelectedValue);
```

**Sink:** MemberMasterHindi.aspx.cs:518 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
516                         cmd.Parameters.AddWithValue("@HC_LADDRESS",
txtlocadd.Text.Trim());
517                         cmd.Parameters.AddWithValue("@Telephone1",
txtlocph.Text.Trim());
518                         cmd.Parameters.AddWithValue("@HC_PADDRESS",
txtconadd.Text.Trim());
519                         cmd.Parameters.AddWithValue("@Telephone2",
txtconph.Text.Trim());
520                         cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
cmbparty.SelectedValue);
```

## MemberMaster.aspx.cs, line 374 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 374 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:374 System.Web.UI.WebControls.TextBox.get_Text()

```
372                             cmd.Parameters.AddWithValue("@MP_FNAME",
txtfname.Text.Trim());
373                             cmd.Parameters.AddWithValue("@MP_LNAME",
txtlname.Text.Trim());
374                             cmd.Parameters.AddWithValue("@C_LADDRESS",
txtlocadd.Text.Trim());
375                             cmd.Parameters.AddWithValue("@Telephone1",
txtlocph.Text.Trim());
376                             cmd.Parameters.AddWithValue("@C_PADDRESS",
txtconadd.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:374 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
372                             cmd.Parameters.AddWithValue("@MP_FNAME",
txtfname.Text.Trim());
373                             cmd.Parameters.AddWithValue("@MP_LNAME",
txtlname.Text.Trim());
374                             cmd.Parameters.AddWithValue("@C_LADDRESS",
txtlocadd.Text.Trim());
375                             cmd.Parameters.AddWithValue("@Telephone1",
txtlocph.Text.Trim());
376                             cmd.Parameters.AddWithValue("@C_PADDRESS",
txtconadd.Text.Trim());
```

## MemberMasterHindi.aspx.cs, line 515 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 515 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:515
System.Web.UI.WebControls.TextBox.get_Text()

```
513                         cmd.Parameters.AddWithValue("@HMP_INIT",
        txtinitial.Text.Trim());
514                         cmd.Parameters.AddWithValue("@HMP_FNAME",
        txtfname.Text.Trim());
515                         cmd.Parameters.AddWithValue("@HMP_LNAME",
        txtlname.Text.Trim());
516                         cmd.Parameters.AddWithValue("@HC_LADDRESS",
        txtlocadd.Text.Trim());
517                         cmd.Parameters.AddWithValue("@Telephone1",
        txtlocph.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:515
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
513                         cmd.Parameters.AddWithValue("@HMP_INIT",
        txtinitial.Text.Trim());
514                         cmd.Parameters.AddWithValue("@HMP_FNAME",
        txtfname.Text.Trim());
515                         cmd.Parameters.AddWithValue("@HMP_LNAME",
        txtlname.Text.Trim());
516                         cmd.Parameters.AddWithValue("@HC_LADDRESS",
        txtlocadd.Text.Trim());
517                         cmd.Parameters.AddWithValue("@Telephone1",
        txtlocph.Text.Trim());
```

## MemberMasterHindi.aspx.cs, line 623 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 623 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:623
System.Web.UI.WebControls.TextBox.get_Text()

```
621                     cmd.Parameters.AddWithValue("@Telephone1", txtlocph.Text.Trim());
622                     cmd.Parameters.AddWithValue("@HC_PADDRESS",
        txtconadd.Text.Trim());
623                     cmd.Parameters.AddWithValue("@Telephone2", txtconph.Text.Trim());
624                     cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
        cmbparty.SelectedValue);
625                     cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:623
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
621                     cmd.Parameters.AddWithValue("@Telephone1", txtlocph.Text.Trim());
622                     cmd.Parameters.AddWithValue("@HC_PADDRESS",
        txtconadd.Text.Trim());
623                     cmd.Parameters.AddWithValue("@Telephone2", txtconph.Text.Trim());
624                     cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
        cmbparty.SelectedValue);
625                     cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
```

## MemberMasterHindi.aspx.cs, line 634 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 634 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:634
System.Web.UI.WebControls.TextBox.get_Text()

```
632                     cmd.Parameters.AddWithValue("@mobile2", TxtMobile2.Text.Trim());
633                     cmd.Parameters.AddWithValue("@mobile3", TxtMobile3.Text.Trim());
634                     cmd.Parameters.AddWithValue("@mobile4", TxtMobile4.Text.Trim());
635                     cmd.Parameters.AddWithValue("@Status",
        rdobtnStatus.SelectedValue.Trim());
```

```
636                                  con.Open();
```

**Sink:** MemberMasterHindi.aspx.cs:634 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
632                                  cmd.Parameters.AddWithValue("@mobile2", TxtMobile2.Text.Trim());
633                                  cmd.Parameters.AddWithValue("@mobile3", TxtMobile3.Text.Trim());
634                                  cmd.Parameters.AddWithValue("@mobile4", TxtMobile4.Text.Trim());
635                                  cmd.Parameters.AddWithValue("@Status",
                     rdobtnStatus.SelectedValue.Trim());
636                                  con.Open();
```

## MemberMasterHindi.aspx.cs, line 472 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 472 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:472 System.Web.UI.WebControls.TextBox.get_Text()

```
470                                      cmd.Parameters.AddWithValue("@initial_h",
                     txtinitial.Text.Trim());
471                                      cmd.Parameters.AddWithValue("@first_name_h",
                     txtfname.Text.Trim());
472                                      cmd.Parameters.AddWithValue("@last_name_h",
                     txtlname.Text.Trim());
473                                      cmd.Parameters.AddWithValue("@H_Address1",
                     txtlocadd.Text.Trim());
474                                      cmd.Parameters.AddWithValue("@Telephone1",
                     txtlocph.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:472 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
470                                      cmd.Parameters.AddWithValue("@initial_h",
                     txtinitial.Text.Trim());
471                                      cmd.Parameters.AddWithValue("@first_name_h",
                     txtfname.Text.Trim());
472                                      cmd.Parameters.AddWithValue("@last_name_h",
                     txtlname.Text.Trim());
473                                      cmd.Parameters.AddWithValue("@H_Address1",
                     txtlocadd.Text.Trim());
474                                      cmd.Parameters.AddWithValue("@Telephone1",
                     txtlocph.Text.Trim());
```

## PartyMaster.aspx.cs, line 208 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in PartyMaster.aspx.cs can execute a SQL statement on line 208 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** PartyMaster.aspx.cs:208 System.Web.UI.WebControls.TextBox.get_Text()

```
206                                  cmd.Parameters.AddWithValue("@PARTY_FNAME",
                     SqlDbType.VarChar).Value = txtPartyName.Text.Trim();
207                                  cmd.Parameters.AddWithValue("@PARTY_SNAME",
                     SqlDbType.VarChar).Value = txtShortName.Text.Trim();
208                                  cmd.Parameters.AddWithValue("@PARTY_FNAME_H",
                     SqlDbType.NVarChar).Value = txtHPartyName.Text.Trim();
209                                  cmd.Parameters.AddWithValue("@PARTY_SNAME_H",
                     SqlDbType.NVarChar).Value = txtHShortName.Text.Trim();
210                                  cmd.Parameters.AddWithValue("@LSTO", SqlDbType.SmallInt).Value =
                     txtLSTo.Text.Trim();
```

**Sink:** PartyMaster.aspx.cs:208 System.Data.Common.DbParameter.set_Value()

```
206                                  cmd.Parameters.AddWithValue("@PARTY_FNAME",
                     SqlDbType.VarChar).Value = txtPartyName.Text.Trim();
207                                  cmd.Parameters.AddWithValue("@PARTY_SNAME",
                     SqlDbType.VarChar).Value = txtShortName.Text.Trim();
208                                  cmd.Parameters.AddWithValue("@PARTY_FNAME_H",
                     SqlDbType.NVarChar).Value = txtHPartyName.Text.Trim();
```

```
209                           cmd.Parameters.AddWithValue("@PARTY_SNAME_H",
            SqlDbType.NVarChar).Value = txtHShortName.Text.Trim();
210                           cmd.Parameters.AddWithValue("@LSTO", SqlDbType.SmallInt).Value =
            txtLSTo.Text.Trim();
```

## MemberMaster.aspx.cs, line 377 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 377 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:377 System.Web.UI.WebControls.TextBox.get_Text()

```
375                               cmd.Parameters.AddWithValue("@Telephone1",
            txtlocph.Text.Trim());
376                               cmd.Parameters.AddWithValue("@C_PADDRESS",
            txtconadd.Text.Trim());
377                               cmd.Parameters.AddWithValue("@Telephone2",
            txtconph.Text.Trim());
378                               cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
            cmbparty.SelectedValue);
379                               cmd.Parameters.AddWithValue("@mobile",
            TxtMobile.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:377 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
375                               cmd.Parameters.AddWithValue("@Telephone1",
            txtlocph.Text.Trim());
376                               cmd.Parameters.AddWithValue("@C_PADDRESS",
            txtconadd.Text.Trim());
377                               cmd.Parameters.AddWithValue("@Telephone2",
            txtconph.Text.Trim());
378                               cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
            cmbparty.SelectedValue);
379                               cmd.Parameters.AddWithValue("@mobile",
            TxtMobile.Text.Trim());
```

## MemberMaster.aspx.cs, line 460 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 460 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:460 System.Web.UI.WebControls.TextBox.get_Text()

```
458                         cmd.Parameters.AddWithValue("@mobile2",
            TxtMobile2.Text.Trim());
459                         cmd.Parameters.AddWithValue("@mobile3",
            TxtMobile3.Text.Trim());
460                         cmd.Parameters.AddWithValue("@mobile4",
            TxtMobile4.Text.Trim());
461                         cmd.Parameters.AddWithValue("@Status",
            rdobtnStatus.SelectedValue.Trim());
462                         con.Open();
```

**Sink:** MemberMaster.aspx.cs:460 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
458                         cmd.Parameters.AddWithValue("@mobile2",
            TxtMobile2.Text.Trim());
459                         cmd.Parameters.AddWithValue("@mobile3",
            TxtMobile3.Text.Trim());
460                         cmd.Parameters.AddWithValue("@mobile4",
            TxtMobile4.Text.Trim());
461                         cmd.Parameters.AddWithValue("@Status",
            rdobtnStatus.SelectedValue.Trim());
462                         con.Open();
```

## MemberMaster.aspx.cs, line 494 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |

**Kingdom:** Security Features

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 494 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:494 System.Web.UI.WebControls.TextBox.get_Text()

```
492                         cmd.Parameters.AddWithValue("@mobile2",
      TxtMobile2.Text.Trim());
493                         cmd.Parameters.AddWithValue("@mobile3",
      TxtMobile3.Text.Trim());
494                         cmd.Parameters.AddWithValue("@mobile4",
      TxtMobile4.Text.Trim());
495                         cmd.Parameters.AddWithValue("@Status",
      rdobtnStatus.SelectedValue.Trim());
496                         con.Open();
```

**Sink:** MemberMaster.aspx.cs:494 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
492                         cmd.Parameters.AddWithValue("@mobile2",
      TxtMobile2.Text.Trim());
493                         cmd.Parameters.AddWithValue("@mobile3",
      TxtMobile3.Text.Trim());
494                         cmd.Parameters.AddWithValue("@mobile4",
      TxtMobile4.Text.Trim());
495                         cmd.Parameters.AddWithValue("@Status",
      rdobtnStatus.SelectedValue.Trim());
496                         con.Open();
```

## MeetingAttendance.aspx.cs, line 450 (Access Control: Database)

**Fortify Priority:** High **Folder** High

**Kingdom:** Security Features

**Abstract:** Without proper access control, the method excuteYesquery() in MeetingAttendance.aspx.cs can execute a SQL statement on line 450 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MeetingAttendance.aspx.cs:450 System.Web.UI.WebControls.TextBox.get_Text()

```
448            cmd.Parameters.AddWithValue("@Title", cmbtitle.Text);
449            cmd.Parameters.AddWithValue("@dateofmeet", txtdate1.Text);
450            cmd.Parameters.AddWithValue("@dateofmeet2", txtdate2.Text);
451            cmd.Parameters.AddWithValue("@timeofmeet", txttime.Text);
452            cmd.Parameters.AddWithValue("@cid", cmbcommittee.SelectedValue);
```

**Sink:** MeetingAttendance.aspx.cs:450 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
448            cmd.Parameters.AddWithValue("@Title", cmbtitle.Text);
449            cmd.Parameters.AddWithValue("@dateofmeet", txtdate1.Text);
450            cmd.Parameters.AddWithValue("@dateofmeet2", txtdate2.Text);
451            cmd.Parameters.AddWithValue("@timeofmeet", txttime.Text);
452            cmd.Parameters.AddWithValue("@cid", cmbcommittee.SelectedValue);
```

## MemberMasterHindi.aspx.cs, line 488 (Access Control: Database)

**Fortify Priority:** High **Folder** High

**Kingdom:** Security Features

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 488 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:488 System.Web.UI.WebControls.TextBox.get_Text()

```
486                         cmd.Parameters.AddWithValue("@const_name_h",
      cmbConst.SelectedItem.Text.Trim());
487                         cmd.Parameters.AddWithValue("@mobile2",
      TxtMobile2.Text.Trim());
```

```
488                                          cmd.Parameters.AddWithValue("@mobile3",
                     TxtMobile3.Text.Trim());
489                                          cmd.Parameters.AddWithValue("@mobile4",
                     TxtMobile4.Text.Trim());
490                                          cmd.Parameters.AddWithValue("@MP_JoinDate",
                     txtMPJoinDate.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:488 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
486                                          cmd.Parameters.AddWithValue("@const_name_h",
                     cmbConst.SelectedItem.Text.Trim());
487                                          cmd.Parameters.AddWithValue("@mobile2",
                     TxtMobile2.Text.Trim());
488                                          cmd.Parameters.AddWithValue("@mobile3",
                     TxtMobile3.Text.Trim());
489                                          cmd.Parameters.AddWithValue("@mobile4",
                     TxtMobile4.Text.Trim());
490                                          cmd.Parameters.AddWithValue("@MP_JoinDate",
                     txtMPJoinDate.Text.Trim());
```

## MemberMasterHindi.aspx.cs, line 582 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 582 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:582 System.Web.UI.WebControls.TextBox.get_Text()

```
580                                      cmd.CommandType = CommandType.StoredProcedure;
581                                      cmd.Parameters.AddWithValue("@mpsno", ViewState["Key"]);
582                                      cmd.Parameters.AddWithValue("@initial_h", txtinitial.Text.Trim());
583                                      cmd.Parameters.AddWithValue("@first_name_h",
                     txtfname.Text.Trim());
584                                      cmd.Parameters.AddWithValue("@last_name_h", txtlname.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:582 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
580                                      cmd.CommandType = CommandType.StoredProcedure;
581                                      cmd.Parameters.AddWithValue("@mpsno", ViewState["Key"]);
582                                      cmd.Parameters.AddWithValue("@initial_h", txtinitial.Text.Trim());
583                                      cmd.Parameters.AddWithValue("@first_name_h",
                     txtfname.Text.Trim());
584                                      cmd.Parameters.AddWithValue("@last_name_h", txtlname.Text.Trim());
```

## MemberMaster.aspx.cs, line 373 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 373 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:373 System.Web.UI.WebControls.TextBox.get_Text()

```
371                                          cmd.Parameters.AddWithValue("@MP_INIT",
                     txtinitial.Text.Trim());
372                                          cmd.Parameters.AddWithValue("@MP_FNAME",
                     txtfname.Text.Trim());
373                                          cmd.Parameters.AddWithValue("@MP_LNAME",
                     txtlname.Text.Trim());
374                                          cmd.Parameters.AddWithValue("@C_LADDRESS",
                     txtlocadd.Text.Trim());
375                                          cmd.Parameters.AddWithValue("@Telephone1",
                     txtlocph.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:373 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
371                                          cmd.Parameters.AddWithValue("@MP_INIT",
                     txtinitial.Text.Trim());
```

```
372                           cmd.Parameters.AddWithValue("@MP_FNAME",
      txtfname.Text.Trim());
373                           cmd.Parameters.AddWithValue("@MP_LNAME",
      txtlname.Text.Trim());
374                           cmd.Parameters.AddWithValue("@C_LADDRESS",
      txtlocadd.Text.Trim());
375                           cmd.Parameters.AddWithValue("@Telephone1",
      txtlocph.Text.Trim());
```

## MemberMaster.aspx.cs, line 488 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 488 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:488 System.Web.UI.WebControls.TextBox.get_Text()

```
486                        cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
      cmbparty.SelectedValue);
487                        cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
488                        cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
489                        cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
490                        cmd.Parameters.AddWithValue("@C_MP_STATE_CODE",
      cmbState.SelectedValue.Trim());
```

**Sink:** MemberMaster.aspx.cs:488 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
486                        cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
      cmbparty.SelectedValue);
487                        cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
488                        cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
489                        cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
490                        cmd.Parameters.AddWithValue("@C_MP_STATE_CODE",
      cmbState.SelectedValue.Trim());
```

## PartyMaster.aspx.cs, line 209 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in PartyMaster.aspx.cs can execute a SQL statement on line 209 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** PartyMaster.aspx.cs:209 System.Web.UI.WebControls.TextBox.get_Text()

```
207                    cmd.Parameters.AddWithValue("@PARTY_SNAME",
      SqlDbType.VarChar).Value = txtShortName.Text.Trim();
208                    cmd.Parameters.AddWithValue("@PARTY_FNAME_H",
      SqlDbType.NVarChar).Value = txtHPartyName.Text.Trim();
209                    cmd.Parameters.AddWithValue("@PARTY_SNAME_H",
      SqlDbType.NVarChar).Value = txtHShortName.Text.Trim();
210                    cmd.Parameters.AddWithValue("@LSTO", SqlDbType.SmallInt).Value =
      txtLSTo.Text.Trim();
211                    cmd.Parameters.AddWithValue("@LEADER", SqlDbType.VarChar).Value =
      txtLeader.Text.Trim();
```

**Sink:** PartyMaster.aspx.cs:209 System.Data.Common.DbParameter.set_Value()

```
207                    cmd.Parameters.AddWithValue("@PARTY_SNAME",
      SqlDbType.VarChar).Value = txtShortName.Text.Trim();
208                    cmd.Parameters.AddWithValue("@PARTY_FNAME_H",
      SqlDbType.NVarChar).Value = txtHPartyName.Text.Trim();
209                    cmd.Parameters.AddWithValue("@PARTY_SNAME_H",
      SqlDbType.NVarChar).Value = txtHShortName.Text.Trim();
210                    cmd.Parameters.AddWithValue("@LSTO", SqlDbType.SmallInt).Value =
      txtLSTo.Text.Trim();
211                    cmd.Parameters.AddWithValue("@LEADER", SqlDbType.VarChar).Value =
      txtLeader.Text.Trim();
```

## MemberMasterHindi.aspx.cs, line 583 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 583 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:583
System.Web.UI.WebControls.TextBox.get_Text()

```
581                         cmd.Parameters.AddWithValue("@mpsno", ViewState["Key"]);
582                         cmd.Parameters.AddWithValue("@initial_h", txtinitial.Text.Trim());
583                         cmd.Parameters.AddWithValue("@first_name_h",
          txtfname.Text.Trim());
584                         cmd.Parameters.AddWithValue("@last_name_h", txtlname.Text.Trim());
585                         cmd.Parameters.AddWithValue("@H_Address1", txtlocadd.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:583
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
581                         cmd.Parameters.AddWithValue("@mpsno", ViewState["Key"]);
582                         cmd.Parameters.AddWithValue("@initial_h", txtinitial.Text.Trim());
583                         cmd.Parameters.AddWithValue("@first_name_h",
          txtfname.Text.Trim());
584                         cmd.Parameters.AddWithValue("@last_name_h", txtlname.Text.Trim());
585                         cmd.Parameters.AddWithValue("@H_Address1", txtlocadd.Text.Trim());
```

## MemberMaster.aspx.cs, line 443 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 443 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:443 System.Web.UI.WebControls.TextBox.get_Text()

```
441                         cmd.Parameters.AddWithValue("@mpsno", ViewState["Key"]);
442                         cmd.Parameters.AddWithValue("@Initial",
          txtinitial.Text.Trim());
443                         cmd.Parameters.AddWithValue("@first_name",
          txtfname.Text.Trim());
444                         cmd.Parameters.AddWithValue("@last_name",
          txtlname.Text.Trim());
445                         cmd.Parameters.AddWithValue("@Address1",
          txtlocadd.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:443
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
441                         cmd.Parameters.AddWithValue("@mpsno", ViewState["Key"]);
442                         cmd.Parameters.AddWithValue("@Initial",
          txtinitial.Text.Trim());
443                         cmd.Parameters.AddWithValue("@first_name",
          txtfname.Text.Trim());
444                         cmd.Parameters.AddWithValue("@last_name",
          txtlname.Text.Trim());
445                         cmd.Parameters.AddWithValue("@Address1",
          txtlocadd.Text.Trim());
```

## MemberMasterHindi.aspx.cs, line 474 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 474 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:474
System.Web.UI.WebControls.TextBox.get_Text()

```
472                         cmd.Parameters.AddWithValue("@last_name_h",
          txtlname.Text.Trim());
473                         cmd.Parameters.AddWithValue("@H_Address1",
          txtlocadd.Text.Trim());
```

```
474                         cmd.Parameters.AddWithValue("@Telephone1",
            txtlocph.Text.Trim());
475                         cmd.Parameters.AddWithValue("@H_Address2",
            txtconadd.Text.Trim());
476                         cmd.Parameters.AddWithValue("@Telephone2",
            txtconph.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:474 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
472                         cmd.Parameters.AddWithValue("@last_name_h",
            txtlname.Text.Trim());
473                         cmd.Parameters.AddWithValue("@H_Address1",
            txtlocadd.Text.Trim());
474                         cmd.Parameters.AddWithValue("@Telephone1",
            txtlocph.Text.Trim());
475                         cmd.Parameters.AddWithValue("@H_Address2",
            txtconadd.Text.Trim());
476                         cmd.Parameters.AddWithValue("@Telephone2",
            txtconph.Text.Trim());
```

## MemberMaster.aspx.cs, line 340 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 340 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:340 System.Web.UI.WebControls.TextBox.get_Text()

```
338                             cmd.Parameters.AddWithValue("@party_fname",
            cmbparty.SelectedItem.Text);
339                             cmd.Parameters.AddWithValue("@mobile",
            TxtMobile.Text.Trim());
340                             cmd.Parameters.AddWithValue("@email1",
            TxteMail1.Text.Trim());
341                             cmd.Parameters.AddWithValue("@email2",
            TxteMail2.Text.Trim());
342                             cmd.Parameters.AddWithValue("@last_ls",
            DdlHouseNo.SelectedValue.Trim());
```

**Sink:** MemberMaster.aspx.cs:340 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
338                             cmd.Parameters.AddWithValue("@party_fname",
            cmbparty.SelectedItem.Text);
339                             cmd.Parameters.AddWithValue("@mobile",
            TxtMobile.Text.Trim());
340                             cmd.Parameters.AddWithValue("@email1",
            TxteMail1.Text.Trim());
341                             cmd.Parameters.AddWithValue("@email2",
            TxteMail2.Text.Trim());
342                             cmd.Parameters.AddWithValue("@last_ls",
            DdlHouseNo.SelectedValue.Trim());
```

## MemberMasterHindi.aspx.cs, line 585 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 585 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:585 System.Web.UI.WebControls.TextBox.get_Text()

```
583                     cmd.Parameters.AddWithValue("@first_name_h",
            txtfname.Text.Trim());
584                     cmd.Parameters.AddWithValue("@last_name_h", txtlname.Text.Trim());
585                     cmd.Parameters.AddWithValue("@H_Address1", txtlocadd.Text.Trim());
586                     cmd.Parameters.AddWithValue("@Telephone1", txtlocph.Text.Trim());
587                     cmd.Parameters.AddWithValue("@H_Address2", txtconadd.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:585 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
                    cmd.Parameters.AddWithValue("@first_name_h",
txtfname.Text.Trim());
                    cmd.Parameters.AddWithValue("@last_name_h", txtlname.Text.Trim());
                    cmd.Parameters.AddWithValue("@H_Address1", txtlocadd.Text.Trim());
                    cmd.Parameters.AddWithValue("@Telephone1", txtlocph.Text.Trim());
                    cmd.Parameters.AddWithValue("@H_Address2", txtconadd.Text.Trim());
```

## PartyMaster.aspx.cs, line 161 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in PartyMaster.aspx.cs can execute a SQL statement on line 161 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** PartyMaster.aspx.cs:161 System.Web.UI.WebControls.TextBox.get_Text()

```
                    cmd.Parameters.AddWithValue("@PARTY_SNAME",
SqlDbType.VarChar).Value = txtShortName.Text.Trim();
                    cmd.Parameters.AddWithValue("@PARTY_FNAME_H",
SqlDbType.NVarChar).Value = txtHPartyName.Text.Trim();
                    cmd.Parameters.AddWithValue("@PARTY_SNAME_H",
SqlDbType.NVarChar).Value = txtHShortName.Text.Trim();
                    cmd.Parameters.AddWithValue("@LSFROM", SqlDbType.SmallInt).Value =
ddlLSFrom.SelectedValue.Trim();
                    cmd.Parameters.AddWithValue("@LSTO", SqlDbType.SmallInt).Value =
99;
```

**Sink:** PartyMaster.aspx.cs:161 System.Data.Common.DbParameter.set_Value()

```
                    cmd.Parameters.AddWithValue("@PARTY_SNAME",
SqlDbType.VarChar).Value = txtShortName.Text.Trim();
                    cmd.Parameters.AddWithValue("@PARTY_FNAME_H",
SqlDbType.NVarChar).Value = txtHPartyName.Text.Trim();
                    cmd.Parameters.AddWithValue("@PARTY_SNAME_H",
SqlDbType.NVarChar).Value = txtHShortName.Text.Trim();
                    cmd.Parameters.AddWithValue("@LSFROM", SqlDbType.SmallInt).Value =
ddlLSFrom.SelectedValue.Trim();
                    cmd.Parameters.AddWithValue("@LSTO", SqlDbType.SmallInt).Value =
99;
```

## MemberMasterHindi.aspx.cs, line 617 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 617 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:617 System.Web.UI.WebControls.TextBox.get_Text()

```
                    cmd.CommandType = CommandType.StoredProcedure;
                    cmd.Parameters.AddWithValue("@MP_CODE", ViewState["Key"]);
                    cmd.Parameters.AddWithValue("@HMP_INIT", txtinitial.Text.Trim());
                    cmd.Parameters.AddWithValue("@HMP_FNAME", txtfname.Text.Trim());
                    cmd.Parameters.AddWithValue("@HMP_LNAME", txtlname.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:617 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
                    cmd.CommandType = CommandType.StoredProcedure;
                    cmd.Parameters.AddWithValue("@MP_CODE", ViewState["Key"]);
                    cmd.Parameters.AddWithValue("@HMP_INIT", txtinitial.Text.Trim());
                    cmd.Parameters.AddWithValue("@HMP_FNAME", txtfname.Text.Trim());
                    cmd.Parameters.AddWithValue("@HMP_LNAME", txtlname.Text.Trim());
```

## MeetingCommittee.aspx.cs, line 269 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method InsertSchedule() in MeetingCommittee.aspx.cs can execute a SQL statement on line 269 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MeetingCommittee.aspx.cs:269 System.Web.UI.WebControls.TextBox.get_Text()

```
267                     cmd.CommandType = CommandType.StoredProcedure;
268
269                     cmd.Parameters.Add("@title", txttitle.Text);
270                     cmd.Parameters.Add("@dateofmeet", txtdate1.Text);
271                     cmd.Parameters.Add("@dateofmeet2", txtdate2.Text);
```

**Sink:** MeetingCommittee.aspx.cs:269 System.Data.SqlClient.SqlParameterCollection.Add()

```
267                     cmd.CommandType = CommandType.StoredProcedure;
268
269                     cmd.Parameters.Add("@title", txttitle.Text);
270                     cmd.Parameters.Add("@dateofmeet", txtdate1.Text);
271                     cmd.Parameters.Add("@dateofmeet2", txtdate2.Text);
```

## MemberMasterHindi.aspx.cs, line 625 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 625 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:625 System.Web.UI.WebControls.TextBox.get_Text()

```
623                         cmd.Parameters.AddWithValue("@Telephone2", txtconph.Text.Trim());
624                         cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
cmbparty.SelectedValue);
625                         cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
626                         cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
627                         cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:625 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
623                         cmd.Parameters.AddWithValue("@Telephone2", txtconph.Text.Trim());
624                         cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
cmbparty.SelectedValue);
625                         cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
626                         cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
627                         cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
```

## Login.aspx.cs, line 180 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method submit_Click() in Login.aspx.cs can execute a SQL statement on line 180 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** Login.aspx.cs:141 System.Web.UI.WebControls.TextBox.get_Text()

```
139                     string pass = String.Empty;
140                     string pageone = String.Empty;
141                     string uname = UserName.Text.ToString().Trim();
142                     Session["uname"] = uname;
143                     //Get Utype
```

**Sink:** Login.aspx.cs:180 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
178                             sqlCommand.CommandText = "[dbo].[CMS_LoginMasterUserSP]";
179                             sqlCommand.CommandType = CommandType.StoredProcedure;
180                             sqlCommand.Parameters.AddWithValue("@uname", uname);
```

```
181                                   sqlCommand.Parameters.AddWithValue("@UType",
                      Convert.ToChar("C"));
182                                   sqlCommand.CommandTimeout = 600;
```

## PartyMaster.aspx.cs, line 196 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in PartyMaster.aspx.cs can execute a SQL statement on line 196 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** PartyMaster.aspx.cs:196 System.Web.UI.WebControls.TextBox.get_Text()

```
194                                   cmd.Parameters.AddWithValue("@PARTY_SNAME",
                      SqlDbType.VarChar).Value = txtShortName.Text.Trim();
195                                   cmd.Parameters.AddWithValue("@PARTY_FNAME_H",
                      SqlDbType.NVarChar).Value = txtHPartyName.Text.Trim();
196                                   cmd.Parameters.AddWithValue("@PARTY_SNAME_H",
                      SqlDbType.NVarChar).Value = txtHShortName.Text.Trim();
197                                   cmd.Parameters.AddWithValue("@LEADER", SqlDbType.VarChar).Value =
                      txtLeader.Text.Trim();
198                                   cmd.Parameters.AddWithValue("@LEADER_H", SqlDbType.NVarChar).Value
                      = txtLeaderH.Text.Trim();
```

**Sink:** PartyMaster.aspx.cs:196 System.Data.Common.DbParameter.set_Value()

```
194                                   cmd.Parameters.AddWithValue("@PARTY_SNAME",
                      SqlDbType.VarChar).Value = txtShortName.Text.Trim();
195                                   cmd.Parameters.AddWithValue("@PARTY_FNAME_H",
                      SqlDbType.NVarChar).Value = txtHPartyName.Text.Trim();
196                                   cmd.Parameters.AddWithValue("@PARTY_SNAME_H",
                      SqlDbType.NVarChar).Value = txtHShortName.Text.Trim();
197                                   cmd.Parameters.AddWithValue("@LEADER", SqlDbType.VarChar).Value =
                      txtLeader.Text.Trim();
198                                   cmd.Parameters.AddWithValue("@LEADER_H", SqlDbType.NVarChar).Value
                      = txtLeaderH.Text.Trim();
```

## MemberMasterHindi.aspx.cs, line 618 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 618 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:618 System.Web.UI.WebControls.TextBox.get_Text()

```
616                                   cmd.Parameters.AddWithValue("@MP_CODE", ViewState["Key"]);
617                                   cmd.Parameters.AddWithValue("@HMP_INIT", txtinitial.Text.Trim());
618                                   cmd.Parameters.AddWithValue("@HMP_FNAME", txtfname.Text.Trim());
619                                   cmd.Parameters.AddWithValue("@HMP_LNAME", txtlname.Text.Trim());
620                                   cmd.Parameters.AddWithValue("@HC_LADDRESS",
                      txtlocadd.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:618 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
616                                   cmd.Parameters.AddWithValue("@MP_CODE", ViewState["Key"]);
617                                   cmd.Parameters.AddWithValue("@HMP_INIT", txtinitial.Text.Trim());
618                                   cmd.Parameters.AddWithValue("@HMP_FNAME", txtfname.Text.Trim());
619                                   cmd.Parameters.AddWithValue("@HMP_LNAME", txtlname.Text.Trim());
620                                   cmd.Parameters.AddWithValue("@HC_LADDRESS",
                      txtlocadd.Text.Trim());
```

## MemberMaster.aspx.cs, line 349 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 349 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:349 System.Web.UI.WebControls.TextBox.get_Text()

```
347                          cmd.Parameters.AddWithValue("@mobile2",
    TxtMobile2.Text.Trim());
348                          cmd.Parameters.AddWithValue("@mobile3",
    TxtMobile3.Text.Trim());
349                          cmd.Parameters.AddWithValue("@mobile4",
    TxtMobile4.Text.Trim());
350                          cmd.Parameters.AddWithValue("@MP_JoinDate",
    txtMPJoinDate.Text.Trim());
351
```

**Sink:** MemberMaster.aspx.cs:349 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
347                          cmd.Parameters.AddWithValue("@mobile2",
    TxtMobile2.Text.Trim());
348                          cmd.Parameters.AddWithValue("@mobile3",
    TxtMobile3.Text.Trim());
349                          cmd.Parameters.AddWithValue("@mobile4",
    TxtMobile4.Text.Trim());
350                          cmd.Parameters.AddWithValue("@MP_JoinDate",
    txtMPJoinDate.Text.Trim());
351
```

## MemberMaster.aspx.cs, line 442 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 442 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:442 System.Web.UI.WebControls.TextBox.get_Text()

```
440                      cmd.CommandType = CommandType.StoredProcedure;
441                      cmd.Parameters.AddWithValue("@mpsno", ViewState["Key"]);
442                      cmd.Parameters.AddWithValue("@Initial",
    txtinitial.Text.Trim());
443                      cmd.Parameters.AddWithValue("@first_name",
    txtfname.Text.Trim());
444                      cmd.Parameters.AddWithValue("@last_name",
    txtlname.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:442 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
440                      cmd.CommandType = CommandType.StoredProcedure;
441                      cmd.Parameters.AddWithValue("@mpsno", ViewState["Key"]);
442                      cmd.Parameters.AddWithValue("@Initial",
    txtinitial.Text.Trim());
443                      cmd.Parameters.AddWithValue("@first_name",
    txtfname.Text.Trim());
444                      cmd.Parameters.AddWithValue("@last_name",
    txtlname.Text.Trim());
```

## Ministry_details.aspx.cs, line 117 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method Savebtn_Click() in Ministry_details.aspx.cs can execute a SQL statement on line 117 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** Ministry_details.aspx.cs:117 System.Web.UI.WebControls.TextBox.get_Text()

```
115                  cmd.CommandType = CommandType.StoredProcedure;
116                  cmd.Parameters.AddWithValue("@min_name", SqlDbType.VarChar).Value =
    mintxt2.Text.Trim();
117                  cmd.Parameters.AddWithValue("@min_name_h", SqlDbType.VarChar).Value =
    hmintxt2.Text.Trim();
118                  cmd.Parameters.AddWithValue("@min_ab", SqlDbType.VarChar).Value =
    Shortmintxt2.Text.Trim();
```

```
119                     cmd.Parameters.AddWithValue("@min_code", SqlDbType.SmallInt).Value =
ViewState["min_code"];
```

**Sink:** Ministry_details.aspx.cs:117 System.Data.Common.DbParameter.set_Value()

```
115                     cmd.CommandType = CommandType.StoredProcedure;
116                     cmd.Parameters.AddWithValue("@min_name", SqlDbType.VarChar).Value =
mintxt2.Text.Trim();
117                     cmd.Parameters.AddWithValue("@min_name_h", SqlDbType.VarChar).Value =
hmintxt2.Text.Trim();
118                     cmd.Parameters.AddWithValue("@min_ab", SqlDbType.VarChar).Value =
Shortmintxt2.Text.Trim();
119                     cmd.Parameters.AddWithValue("@min_code", SqlDbType.SmallInt).Value =
ViewState["min_code"];
```

## MemberMaster.aspx.cs, line 446 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 446 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:446 System.Web.UI.WebControls.TextBox.get_Text()

```
444                         cmd.Parameters.AddWithValue("@last_name",
txtlname.Text.Trim());
445                         cmd.Parameters.AddWithValue("@Address1",
txtlocadd.Text.Trim());
446                         cmd.Parameters.AddWithValue("@Telephone1",
txtlocph.Text.Trim());
447                         cmd.Parameters.AddWithValue("@Address2",
txtconadd.Text.Trim());
448                         cmd.Parameters.AddWithValue("@Telephone2",
txtconph.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:446 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
444                         cmd.Parameters.AddWithValue("@last_name",
txtlname.Text.Trim());
445                         cmd.Parameters.AddWithValue("@Address1",
txtlocadd.Text.Trim());
446                         cmd.Parameters.AddWithValue("@Telephone1",
txtlocph.Text.Trim());
447                         cmd.Parameters.AddWithValue("@Address2",
txtconadd.Text.Trim());
448                         cmd.Parameters.AddWithValue("@Telephone2",
txtconph.Text.Trim());
```

## MemberMasterHindi.aspx.cs, line 529 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 529 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:529 System.Web.UI.WebControls.TextBox.get_Text()

```
527                         cmd.Parameters.AddWithValue("@CONST_CODE",
cmbConst.SelectedValue.Trim());
528                         cmd.Parameters.AddWithValue("@const_name_h",
cmbConst.SelectedItem.Text.Trim());
529                         cmd.Parameters.AddWithValue("@mobile2",
TxtMobile2.Text.Trim());
530                         cmd.Parameters.AddWithValue("@mobile3",
TxtMobile3.Text.Trim());
531                         cmd.Parameters.AddWithValue("@mobile4",
TxtMobile4.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:529 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
527                         cmd.Parameters.AddWithValue("@CONST_CODE",
cmbConst.SelectedValue.Trim());
```

```
528                                 cmd.Parameters.AddWithValue("@const_name_h",
          cmbConst.SelectedItem.Text.Trim());
529                                 cmd.Parameters.AddWithValue("@mobile2",
          TxtMobile2.Text.Trim());
530                                 cmd.Parameters.AddWithValue("@mobile3",
          TxtMobile3.Text.Trim());
531                                 cmd.Parameters.AddWithValue("@mobile4",
          TxtMobile4.Text.Trim());
```

## PartyMaster.aspx.cs, line 165 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in PartyMaster.aspx.cs can execute a SQL statement on line 165 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** PartyMaster.aspx.cs:165 System.Web.UI.WebControls.TextBox.get_Text()

```
163                               cmd.Parameters.AddWithValue("@LSTO", SqlDbType.SmallInt).Value =
          99;
164                               cmd.Parameters.AddWithValue("@LEADER", SqlDbType.VarChar).Value =
          txtLeader.Text.Trim();
165                               cmd.Parameters.AddWithValue("@LEADER_H", SqlDbType.NVarChar).Value
          = txtLeaderH.Text.Trim();
166                               con.Open();
167                               i = cmd.ExecuteNonQuery();
```

**Sink:** PartyMaster.aspx.cs:165 System.Data.Common.DbParameter.set_Value()

```
163                               cmd.Parameters.AddWithValue("@LSTO", SqlDbType.SmallInt).Value =
          99;
164                               cmd.Parameters.AddWithValue("@LEADER", SqlDbType.VarChar).Value =
          txtLeader.Text.Trim();
165                               cmd.Parameters.AddWithValue("@LEADER_H", SqlDbType.NVarChar).Value
          = txtLeaderH.Text.Trim();
166                               con.Open();
167                               i = cmd.ExecuteNonQuery();
```

## MeetingAttendance.aspx.cs, line 451 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method excuteYesquery() in MeetingAttendance.aspx.cs can execute a SQL statement on line 451 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MeetingAttendance.aspx.cs:451 System.Web.UI.WebControls.TextBox.get_Text()

```
449             cmd.Parameters.AddWithValue("@dateofmeet", txtdate1.Text);
450             cmd.Parameters.AddWithValue("@dateofmeet2", txtdate2.Text);
451             cmd.Parameters.AddWithValue("@timeofmeet", txttime.Text);
452             cmd.Parameters.AddWithValue("@cid", cmbcommittee.SelectedValue);
453             cmd.Parameters.AddWithValue("@Cname", cmbcommittee.SelectedItem.Text);
```

**Sink:** MeetingAttendance.aspx.cs:451 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
449             cmd.Parameters.AddWithValue("@dateofmeet", txtdate1.Text);
450             cmd.Parameters.AddWithValue("@dateofmeet2", txtdate2.Text);
451             cmd.Parameters.AddWithValue("@timeofmeet", txttime.Text);
452             cmd.Parameters.AddWithValue("@cid", cmbcommittee.SelectedValue);
453             cmd.Parameters.AddWithValue("@Cname", cmbcommittee.SelectedItem.Text);
```

## MemberMasterHindi.aspx.cs, line 619 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 619 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:619 System.Web.UI.WebControls.TextBox.get_Text()

```
617                                 cmd.Parameters.AddWithValue("@HMP_INIT", txtinitial.Text.Trim());
618                                 cmd.Parameters.AddWithValue("@HMP_FNAME", txtfname.Text.Trim());
619                                 cmd.Parameters.AddWithValue("@HMP_LNAME", txtlname.Text.Trim());
620                                 cmd.Parameters.AddWithValue("@HC_LADDRESS",
                  txtlocadd.Text.Trim());
621                                 cmd.Parameters.AddWithValue("@Telephone1", txtlocph.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:619 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
617                                 cmd.Parameters.AddWithValue("@HMP_INIT", txtinitial.Text.Trim());
618                                 cmd.Parameters.AddWithValue("@HMP_FNAME", txtfname.Text.Trim());
619                                 cmd.Parameters.AddWithValue("@HMP_LNAME", txtlname.Text.Trim());
620                                 cmd.Parameters.AddWithValue("@HC_LADDRESS",
                  txtlocadd.Text.Trim());
621                                 cmd.Parameters.AddWithValue("@Telephone1", txtlocph.Text.Trim());
```

## MemberMaster.aspx.cs, line 350 (Access Control: Database)

**Fortify Priority:** High **Folder** High

**Kingdom:** Security Features

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 350 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:350 System.Web.UI.WebControls.TextBox.get_Text()

```
348                                       cmd.Parameters.AddWithValue("@mobile3",
                  TxtMobile3.Text.Trim());
349                                       cmd.Parameters.AddWithValue("@mobile4",
                  TxtMobile4.Text.Trim());
350                                       cmd.Parameters.AddWithValue("@MP_JoinDate",
                  txtMPJoinDate.Text.Trim());
351
352                                       con.Open();
```

**Sink:** MemberMaster.aspx.cs:350 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
348                                       cmd.Parameters.AddWithValue("@mobile3",
                  TxtMobile3.Text.Trim());
349                                       cmd.Parameters.AddWithValue("@mobile4",
                  TxtMobile4.Text.Trim());
350                                       cmd.Parameters.AddWithValue("@MP_JoinDate",
                  txtMPJoinDate.Text.Trim());
351
352                                       con.Open();
```

## depatment_detail.aspx.cs, line 145 (Access Control: Database)

**Fortify Priority:** High **Folder** High

**Kingdom:** Security Features

**Abstract:** Without proper access control, the method Savebtn_Click() in depatment_detail.aspx.cs can execute a SQL statement on line 145 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** depatment_detail.aspx.cs:145 System.Web.UI.WebControls.TextBox.get_Text()

```
143                             cmd.CommandType = CommandType.StoredProcedure;
144                             cmd.Parameters.AddWithValue("@dep_name", dep2.Text);
145                             cmd.Parameters.AddWithValue("@hdep_name", hdep2.Text);
146                             cmd.Parameters.AddWithValue("@dep_code",  ViewState["dep_code"]);
147                             conn.Open();
```

**Sink:** depatment_detail.aspx.cs:145 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
143                             cmd.CommandType = CommandType.StoredProcedure;
144                             cmd.Parameters.AddWithValue("@dep_name", dep2.Text);
145                             cmd.Parameters.AddWithValue("@hdep_name", hdep2.Text);
```

```
146                          cmd.Parameters.AddWithValue("@dep_code",  ViewState["dep_code"]);
147                          conn.Open();
```

## PartyMaster.aspx.cs, line 212 (Access Control: Database)

|  |  |  |  |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in PartyMaster.aspx.cs can execute a SQL statement on line 212 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** PartyMaster.aspx.cs:212 System.Web.UI.WebControls.TextBox.get_Text()

```
210                          cmd.Parameters.AddWithValue("@LSTO", SqlDbType.SmallInt).Value =
                    txtLSTo.Text.Trim();
211                          cmd.Parameters.AddWithValue("@LEADER", SqlDbType.VarChar).Value =
                    txtLeader.Text.Trim();
212                          cmd.Parameters.AddWithValue("@LEADER_H", SqlDbType.NVarChar).Value
                    = txtLeaderH.Text.Trim();
213                          cmd.Parameters.AddWithValue("@PARTY_CODE",
                    SqlDbType.SmallInt).Value = ViewState["Key"];
214                          con.Open();
```

**Sink:** PartyMaster.aspx.cs:212 System.Data.Common.DbParameter.set_Value()

```
210                          cmd.Parameters.AddWithValue("@LSTO", SqlDbType.SmallInt).Value =
                    txtLSTo.Text.Trim();
211                          cmd.Parameters.AddWithValue("@LEADER", SqlDbType.VarChar).Value =
                    txtLeader.Text.Trim();
212                          cmd.Parameters.AddWithValue("@LEADER_H", SqlDbType.NVarChar).Value
                    = txtLeaderH.Text.Trim();
213                          cmd.Parameters.AddWithValue("@PARTY_CODE",
                    SqlDbType.SmallInt).Value = ViewState["Key"];
214                          con.Open();
```

## MemberMaster.aspx.cs, line 447 (Access Control: Database)

|  |  |  |  |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 447 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:447 System.Web.UI.WebControls.TextBox.get_Text()

```
445                              cmd.Parameters.AddWithValue("@Address1",
                    txtlocadd.Text.Trim());
446                              cmd.Parameters.AddWithValue("@Telephone1",
                    txtlocph.Text.Trim());
447                              cmd.Parameters.AddWithValue("@Address2",
                    txtconadd.Text.Trim());
448                              cmd.Parameters.AddWithValue("@Telephone2",
                    txtconph.Text.Trim());
449                              cmd.Parameters.AddWithValue("@party_sname",
                    cmbparty.SelectedValue);
```

**Sink:** MemberMaster.aspx.cs:447 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
445                              cmd.Parameters.AddWithValue("@Address1",
                    txtlocadd.Text.Trim());
446                              cmd.Parameters.AddWithValue("@Telephone1",
                    txtlocph.Text.Trim());
447                              cmd.Parameters.AddWithValue("@Address2",
                    txtconadd.Text.Trim());
448                              cmd.Parameters.AddWithValue("@Telephone2",
                    txtconph.Text.Trim());
449                              cmd.Parameters.AddWithValue("@party_sname",
                    cmbparty.SelectedValue);
```

## Ministry_details.aspx.cs, line 153 (Access Control: Database)

|  |  |  |  |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method Savebtn_Click() in Ministry_details.aspx.cs can execute a SQL statement on line 153 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** Ministry_details.aspx.cs:153 System.Web.UI.WebControls.TextBox.get_Text()

```
151                             cmd.Parameters.AddWithValue("@min_code", SqlDbType.SmallInt).Value =
    r;
152                             cmd.Parameters.AddWithValue("@min_name", SqlDbType.VarChar).Value =
    mintxt1.Text.Trim();
153                             cmd.Parameters.AddWithValue("@min_name_h", SqlDbType.VarChar).Value =
    hmintxt1.Text.Trim();
154                             cmd.Parameters.AddWithValue("@min_ab", SqlDbType.VarChar).Value =
    Shortmintxt1.Text.Trim();
155                             conn.Open();
```

**Sink:** Ministry_details.aspx.cs:153 System.Data.Common.DbParameter.set_Value()

```
151                             cmd.Parameters.AddWithValue("@min_code", SqlDbType.SmallInt).Value =
    r;
152                             cmd.Parameters.AddWithValue("@min_name", SqlDbType.VarChar).Value =
    mintxt1.Text.Trim();
153                             cmd.Parameters.AddWithValue("@min_name_h", SqlDbType.VarChar).Value =
    hmintxt1.Text.Trim();
154                             cmd.Parameters.AddWithValue("@min_ab", SqlDbType.VarChar).Value =
    Shortmintxt1.Text.Trim();
155                             conn.Open();
```

## MemberMasterHindi.aspx.cs, line 627 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 627 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:627 System.Web.UI.WebControls.TextBox.get_Text()

```
625                                 cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
626                                 cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
627                                 cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
628                                 cmd.Parameters.AddWithValue("@C_MP_STATE_CODE",
    cmbState.SelectedValue.Trim());
629                                 cmd.Parameters.AddWithValue("@state_name_h",
    cmbState.SelectedItem.Text.Trim());
```

**Sink:** MemberMasterHindi.aspx.cs:627 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
625                                 cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
626                                 cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
627                                 cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
628                                 cmd.Parameters.AddWithValue("@C_MP_STATE_CODE",
    cmbState.SelectedValue.Trim());
629                                 cmd.Parameters.AddWithValue("@state_name_h",
    cmbState.SelectedItem.Text.Trim());
```

## MemberMaster.aspx.cs, line 372 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 372 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:372 System.Web.UI.WebControls.TextBox.get_Text()

```
370                                         cmd.Parameters.AddWithValue("@MP_CODE", id.ToString());
371                                         cmd.Parameters.AddWithValue("@MP_INIT",
    txtinitial.Text.Trim());
372                                         cmd.Parameters.AddWithValue("@MP_FNAME",
    txtfname.Text.Trim());
```

```
373                                  cmd.Parameters.AddWithValue("@MP_LNAME",
                      txtlname.Text.Trim());
374                                  cmd.Parameters.AddWithValue("@C_LADDRESS",
                      txtlocadd.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:372 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
370                                  cmd.Parameters.AddWithValue("@MP_CODE", id.ToString());
371                                  cmd.Parameters.AddWithValue("@MP_INIT",
                      txtinitial.Text.Trim());
372                                  cmd.Parameters.AddWithValue("@MP_FNAME",
                      txtfname.Text.Trim());
373                                  cmd.Parameters.AddWithValue("@MP_LNAME",
                      txtlname.Text.Trim());
374                                  cmd.Parameters.AddWithValue("@C_LADDRESS",
                      txtlocadd.Text.Trim());
```

## MemberMaster.aspx.cs, line 452 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 452 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:452 System.Web.UI.WebControls.TextBox.get_Text()

```
450                              cmd.Parameters.AddWithValue("@party_fname",
                      cmbparty.SelectedItem.Text);
451                              cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
452                              cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
453                              cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
454                              cmd.Parameters.AddWithValue("@STATE_CODE",
                      cmbState.SelectedValue.Trim());
```

**Sink:** MemberMaster.aspx.cs:452 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
450                              cmd.Parameters.AddWithValue("@party_fname",
                      cmbparty.SelectedItem.Text);
451                              cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
452                              cmd.Parameters.AddWithValue("@email1", TxteMail1.Text.Trim());
453                              cmd.Parameters.AddWithValue("@email2", TxteMail2.Text.Trim());
454                              cmd.Parameters.AddWithValue("@STATE_CODE",
                      cmbState.SelectedValue.Trim());
```

## Ministry_details.aspx.cs, line 154 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method Savebtn_Click() in Ministry_details.aspx.cs can execute a SQL statement on line 154 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** Ministry_details.aspx.cs:154 System.Web.UI.WebControls.TextBox.get_Text()

```
152                       cmd.Parameters.AddWithValue("@min_name", SqlDbType.VarChar).Value =
                      mintxt1.Text.Trim();
153                       cmd.Parameters.AddWithValue("@min_name_h", SqlDbType.VarChar).Value =
                      hmintxt1.Text.Trim();
154                       cmd.Parameters.AddWithValue("@min_ab", SqlDbType.VarChar).Value =
                      Shortmintxt1.Text.Trim();
155                       conn.Open();
156                       int j = cmd.ExecuteNonQuery();
```

**Sink:** Ministry_details.aspx.cs:154 System.Data.Common.DbParameter.set_Value()

```
152                       cmd.Parameters.AddWithValue("@min_name", SqlDbType.VarChar).Value =
                      mintxt1.Text.Trim();
153                       cmd.Parameters.AddWithValue("@min_name_h", SqlDbType.VarChar).Value =
                      hmintxt1.Text.Trim();
154                       cmd.Parameters.AddWithValue("@min_ab", SqlDbType.VarChar).Value =
                      Shortmintxt1.Text.Trim();
155                       conn.Open();
```

```
156                                int j = cmd.ExecuteNonQuery();
```

## MemberMaster.aspx.cs, line 485 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 485 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:485 System.Web.UI.WebControls.TextBox.get_Text()

```
483                         cmd.Parameters.AddWithValue("@Telephone1",
                  txtlocph.Text.Trim());
484                         cmd.Parameters.AddWithValue("@C_PADDRESS",
                  txtconadd.Text.Trim());
485                         cmd.Parameters.AddWithValue("@Telephone2",
                  txtconph.Text.Trim());
486                         cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
                  cmbparty.SelectedValue);
487                         cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:485 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
483                         cmd.Parameters.AddWithValue("@Telephone1",
                  txtlocph.Text.Trim());
484                         cmd.Parameters.AddWithValue("@C_PADDRESS",
                  txtconadd.Text.Trim());
485                         cmd.Parameters.AddWithValue("@Telephone2",
                  txtconph.Text.Trim());
486                         cmd.Parameters.AddWithValue("@C_MP_PARTY_CODE",
                  cmbparty.SelectedValue);
487                         cmd.Parameters.AddWithValue("@mobile", TxtMobile.Text.Trim());
```

## MemberMasterHindi.aspx.cs, line 476 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMasterHindi.aspx.cs can execute a SQL statement on line 476 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMasterHindi.aspx.cs:476 System.Web.UI.WebControls.TextBox.get_Text()

```
474                         cmd.Parameters.AddWithValue("@Telephone1",
                  txtlocph.Text.Trim());
475                         cmd.Parameters.AddWithValue("@H_Address2",
                  txtconadd.Text.Trim());
476                         cmd.Parameters.AddWithValue("@Telephone2",
                  txtconph.Text.Trim());
477                         cmd.Parameters.AddWithValue("@party_sname",
                  cmbparty.SelectedValue);
478                         cmd.Parameters.AddWithValue("@PARTY_FNAME_H",
                  cmbparty.SelectedItem.Text);
```

**Sink:** MemberMasterHindi.aspx.cs:476 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
474                         cmd.Parameters.AddWithValue("@Telephone1",
                  txtlocph.Text.Trim());
475                         cmd.Parameters.AddWithValue("@H_Address2",
                  txtconadd.Text.Trim());
476                         cmd.Parameters.AddWithValue("@Telephone2",
                  txtconph.Text.Trim());
477                         cmd.Parameters.AddWithValue("@party_sname",
                  cmbparty.SelectedValue);
478                         cmd.Parameters.AddWithValue("@PARTY_FNAME_H",
                  cmbparty.SelectedItem.Text);
```

## MemberMaster.aspx.cs, line 348 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 348 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:348 System.Web.UI.WebControls.TextBox.get_Text()

```
346                             cmd.Parameters.AddWithValue("@CONST_NAME",
cmbConst.SelectedItem.Text.Trim());
347                             cmd.Parameters.AddWithValue("@mobile2",
TxtMobile2.Text.Trim());
348                             cmd.Parameters.AddWithValue("@mobile3",
TxtMobile3.Text.Trim());
349                             cmd.Parameters.AddWithValue("@mobile4",
TxtMobile4.Text.Trim());
350                             cmd.Parameters.AddWithValue("@MP_JoinDate",
txtMPJoinDate.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:348 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
346                             cmd.Parameters.AddWithValue("@CONST_NAME",
cmbConst.SelectedItem.Text.Trim());
347                             cmd.Parameters.AddWithValue("@mobile2",
TxtMobile2.Text.Trim());
348                             cmd.Parameters.AddWithValue("@mobile3",
TxtMobile3.Text.Trim());
349                             cmd.Parameters.AddWithValue("@mobile4",
TxtMobile4.Text.Trim());
350                             cmd.Parameters.AddWithValue("@MP_JoinDate",
txtMPJoinDate.Text.Trim());
```

## MemberMaster.aspx.cs, line 479 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 479 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:479 System.Web.UI.WebControls.TextBox.get_Text()

```
477                         cmd.CommandType = CommandType.StoredProcedure;
478                         cmd.Parameters.AddWithValue("@MP_CODE", ViewState["Key"]);
479                         cmd.Parameters.AddWithValue("@MP_INIT",
txtinitial.Text.Trim());
480                         cmd.Parameters.AddWithValue("@MP_FNAME",
txtfname.Text.Trim());
481                         cmd.Parameters.AddWithValue("@MP_LNAME",
txtlname.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:479 System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
477                         cmd.CommandType = CommandType.StoredProcedure;
478                         cmd.Parameters.AddWithValue("@MP_CODE", ViewState["Key"]);
479                         cmd.Parameters.AddWithValue("@MP_INIT",
txtinitial.Text.Trim());
480                         cmd.Parameters.AddWithValue("@MP_FNAME",
txtfname.Text.Trim());
481                         cmd.Parameters.AddWithValue("@MP_LNAME",
txtlname.Text.Trim());
```

## MemberMaster.aspx.cs, line 388 (Access Control: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Security Features | | |

**Abstract:** Without proper access control, the method cmdSave_Click() in MemberMaster.aspx.cs can execute a SQL statement on line 388 that contains an attacker-controlled primary key, thereby allowing the attacker to access unauthorized records.

**Source:** MemberMaster.aspx.cs:388 System.Web.UI.WebControls.TextBox.get_Text()

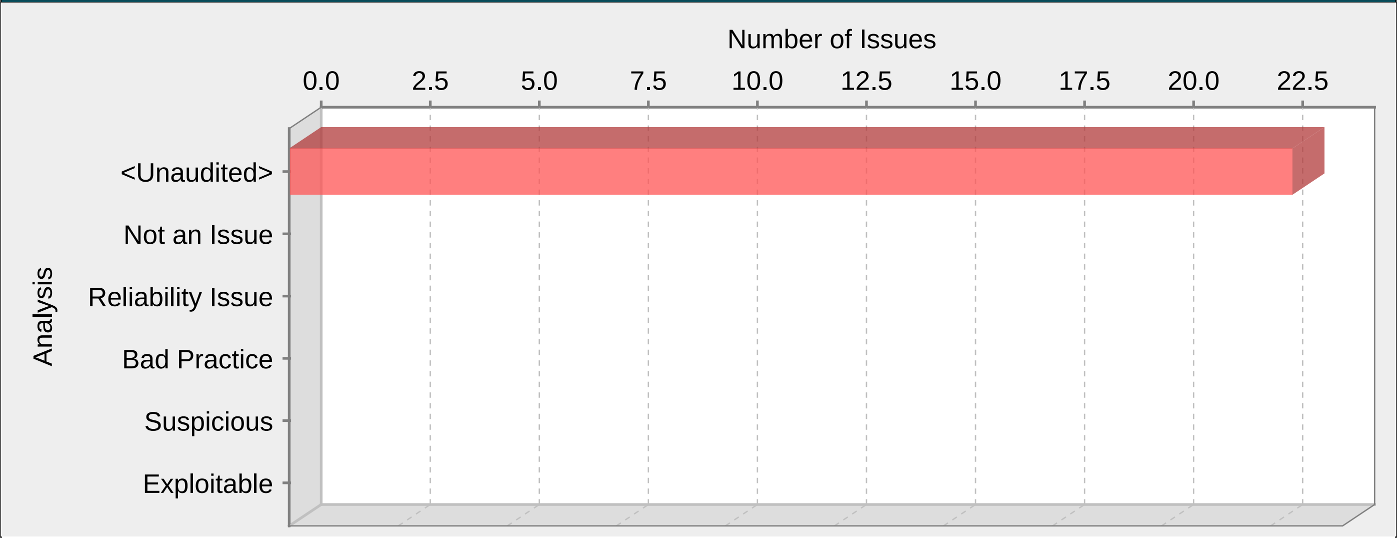
```
386                             cmd.Parameters.AddWithValue("@CONST_NAME",
cmbConst.SelectedItem.Text.Trim());
```

```
387                                          cmd.Parameters.AddWithValue("@mobile2",
                      TxtMobile2.Text.Trim());
388                                          cmd.Parameters.AddWithValue("@mobile3",
                      TxtMobile3.Text.Trim());
389                                          cmd.Parameters.AddWithValue("@mobile4",
                      TxtMobile4.Text.Trim());
390                                          cmd.Parameters.AddWithValue("@MP_JoinDate",
                      txtMPJoinDate.Text.Trim());
```

**Sink:** MemberMaster.aspx.cs:388
System.Data.SqlClient.SqlParameterCollection.AddWithValue()

```
386                                          cmd.Parameters.AddWithValue("@CONST_NAME",
                      cmbConst.SelectedItem.Text.Trim());
387                                          cmd.Parameters.AddWithValue("@mobile2",
                      TxtMobile2.Text.Trim());
388                                          cmd.Parameters.AddWithValue("@mobile3",
                      TxtMobile3.Text.Trim());
389                                          cmd.Parameters.AddWithValue("@mobile4",
                      TxtMobile4.Text.Trim());
390                                          cmd.Parameters.AddWithValue("@MP_JoinDate",
                      txtMPJoinDate.Text.Trim());
```

## Category: SQL Injection (23 Issues)



### MemberMaster.aspx.cs, line 565 (SQL Injection)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Unknown - Custom Issue | | |

**Sink:** C:/Users/APPSECMON6/AppData/Local/Fortify/AWB-4.10/workspace/audit/D__Ankita_WBT_-.NET_PENDING_CMS_mpa_SCAN_3_CMS/MemberMaster.aspx.cs:565 ..()

### DataUtility.cs, line 112 (SQL Injection)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Input Validation and Representation | | |

**Abstract:** On line 112 of DataUtility.cs, the method GetDataScaler() invokes a SQL query built using input coming from an untrusted source. This call could allow an attacker to modify the statement's meaning or to execute arbitrary SQL commands.

**Source:** MemberMasterHindi.aspx.cs:449 System.Web.UI.WebControls.TextBox.get_Text()

```
447
448                                 strsql = "select MP_Code from TempCMS where F_name like '"
449                                     + txtfname.Text.Trim() + "' and L_name like '" +
                      txtlname.Text.Trim() + "' and party_sname='" + cmbparty.SelectedValue + "' and house="
                      + cmbHouseName.SelectedValue + "";
450
451                                 string result = dt.GetDataScaler(strsql);
```

**Sink:** DataUtility.cs:112 System.Data.SqlClient.SqlCommand.SqlCommand()

```
110                         {
111                             openConnection();
112                             SqlCommand cm = new SqlCommand(strsql, con);
113                             Object dr = null;
114                             dr = cm.ExecuteScalar();
```

### DataUtility.cs, line 125 (SQL Injection)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Low | **Folder** | Low |
| **Kingdom:** | Input Validation and Representation | | |

**Abstract:** On line 125 of DataUtility.cs, the method GetDataSet() invokes a SQL query built using input coming from an untrusted source. This call could allow an attacker to modify the statement's meaning or to execute arbitrary SQL commands.

**Sink:** DataUtility.cs:125 SqlDataAdapter()

```
123                             DataSet ds = new DataSet();
124                             openConnection();
125                             SqlDataAdapter da = new SqlDataAdapter(strsql, con);
126                             da.Fill(ds, tb);
127                             closeConnection();
```

## DataUtility.cs, line 112 (SQL Injection)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Input Validation and Representation | | |

**Abstract:** On line 112 of DataUtility.cs, the method GetDataScaler() invokes a SQL query built using input coming from an untrusted source. This call could allow an attacker to modify the statement's meaning or to execute arbitrary SQL commands.

**Source:** MemberMaster.aspx.cs:308 System.Web.UI.WebControls.TextBox.get_Text()

```
                                        //    + txtfname.Text.Trim() + "' and L_name like '" +
txtlname.Text.Trim() + "'";
                                        strsql = "select MP_Code from TempCMS where F_name like '"
                                            + txtfname.Text.Trim() + "' and L_name like '" +
txtlname.Text.Trim() + "' and party_sname='" + cmbparty.SelectedValue + "' and house="
+ cmbHouseName.SelectedValue + "";

                                        string result = dt.GetDataScaler(strsql);
```

**Sink:** DataUtility.cs:112 System.Data.SqlClient.SqlCommand.SqlCommand()

```
                {
                    openConnection();
                    SqlCommand cm = new SqlCommand(strsql, con);
                    Object dr = null;
                    dr = cm.ExecuteScalar();
```

## CommitteeMaster.aspx.cs, line 47 (SQL Injection)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Unknown - Custom Issue | | |

**Sink:** C:/Users/APPSECMON6/AppData/Local/Fortify/AWB-4.10/workspace/audit/D__Ankita_WBT_-.NET_PENDING_CMS_mpa_SCAN_3_CMS/CommitteeMaster.aspx.cs:47 ..()

## DataUtility.cs, line 135 (SQL Injection)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Low | **Folder** | Low |
| **Kingdom:** | Input Validation and Representation | | |

**Abstract:** On line 135 of DataUtility.cs, the method GetDataSet() invokes a SQL query built using input coming from an untrusted source. This call could allow an attacker to modify the statement's meaning or to execute arbitrary SQL commands.

**Sink:** DataUtility.cs:135 SqlDataAdapter()

```
                    DataSet ds = new DataSet();
                    openConnection();
                    SqlDataAdapter da = new SqlDataAdapter(strsql, con);
                    da.Fill(ds);
                    closeConnection();
```

## DataUtility.cs, line 112 (SQL Injection)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Input Validation and Representation | | |

**Abstract:** On line 112 of DataUtility.cs, the method GetDataScaler() invokes a SQL query built using input coming from an untrusted source. This call could allow an attacker to modify the statement's meaning or to execute arbitrary SQL commands.

**Source:** CommitteeMaster.aspx.cs:102 System.Web.UI.WebControls.TextBox.get_Text()

```
                        int i;
                        string strsql;
                        strsql = "select Cid from mpa.Committee where Cname like '" +
txtCommName.Text.Trim() + "'";
                        string result = dt.GetDataScaler(strsql);
                        if (result == null)
```

**Sink:** DataUtility.cs:112 System.Data.SqlClient.SqlCommand.SqlCommand()

```
                {
```

```
111                                 openConnection();
112                                 SqlCommand cm = new SqlCommand(strsql, con);
113                                 Object dr = null;
114                                 dr = cm.ExecuteScalar();
```

## DataUtility.cs, line 65 (SQL Injection)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Input Validation and Representation | | |

**Abstract:** On line 65 of DataUtility.cs, the method ExecuteQuery() invokes a SQL query built using input coming from an untrusted source. This call could allow an attacker to modify the statement's meaning or to execute arbitrary SQL commands.

**Source:** UploadToServer.aspx.cs:67 System.Data.SqlClient.SqlCommand.ExecuteReader()

```
65                          conn.Open();
66                          SqlCommand cmd = new SqlCommand(sqlstr, conn);
67                          SqlDataReader dr = cmd.ExecuteReader();
68                          if (dr.HasRows)
69                          {
```

**Sink:** DataUtility.cs:65 System.Data.Common.DbCommand.set_CommandText()

```
63                              datacomm.Connection = con;
64                              datacomm.CommandType = CommandType.Text;
65                              datacomm.CommandText = str;
66                              int var;
67                              var = datacomm.ExecuteNonQuery();
```

## UploadToServer.aspx.cs, line 66 (SQL Injection)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Low | **Folder** | Low |
| **Kingdom:** | Input Validation and Representation | | |

**Abstract:** On line 66 of UploadToServer.aspx.cs, the method retrive() invokes a SQL query built using input coming from an untrusted source. This call could allow an attacker to modify the statement's meaning or to execute arbitrary SQL commands.

**Sink:** UploadToServer.aspx.cs:66 SqlCommand()

```
64                          SqlConnection conn = new SqlConnection(dbcon);
65                          conn.Open();
66                          SqlCommand cmd = new SqlCommand(sqlstr, conn);
67                          SqlDataReader dr = cmd.ExecuteReader();
68                          if (dr.HasRows)
```

## PartyMaster.aspx.cs, line 87 (SQL Injection)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Unknown - Custom Issue | | |

**Sink:** C:/Users/APPSECMON6/AppData/Local/Fortify/AWB-4.10/workspace/audit/D__Ankita_WBT_-.NET_PENDING_CMS_mpa_SCAN_3_CMS/PartyMaster.aspx.cs:87 ..()

## Attendance_rpt.aspx.cs, line 59 (SQL Injection)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Unknown - Custom Issue | | |

**Sink:** C:/Users/APPSECMON6/AppData/Local/Fortify/AWB-4.10/workspace/audit/D__Ankita_WBT_-.NET_PENDING_CMS_mpa_SCAN_3_CMS/Attendance_rpt.aspx.cs:59 ..()

## CommitteeMaster.aspx.cs, line 102 (SQL Injection)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Unknown - Custom Issue | | |

**Sink:** C:/Users/APPSECMON6/AppData/Local/Fortify/AWB-4.10/workspace/audit/D__Ankita_WBT_-.NET_PENDING_CMS_mpa_SCAN_3_CMS/CommitteeMaster.aspx.cs:102 ..()

## MeetingDetails_rpt.aspx.cs, line 40 (SQL Injection)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Low | **Folder** | Low |
| **Kingdom:** | Input Validation and Representation | | |

**Abstract:** On line 40 of MeetingDetails_rpt.aspx.cs, the method Page_Load() invokes a SQL query built using input coming from an untrusted source. This call could allow an attacker to modify the statement's meaning or to execute arbitrary SQL commands.

**Sink:** MeetingDetails_rpt.aspx.cs:40 SqlCommand()

```
38                  {
39                          //cmd = new SqlCommand("select title,CONVERT(VARCHAR(10), dateofmeet, 101) as dateofmeet,time,subject,venue,remarks from mpa.schdule_srno where cid=" + commi + " and title='" + tit.ToString() + "' and TypeOfCommittee='" + Typecommii.ToString() + "' order by srno desc", conn);
40                          cmd = new SqlCommand("select ltrim(rtrim(day1)) +', the ' + ltrim(rtrim(d1)) + ' ' + ltrim(rtrim(m1)) + ' ' + ltrim(rtrim(y1)) + ' at ' + time as dateofmeet,subject,venue,remarks from mpa.schdule_srno where cid=" + commi + " and title='" + tit.ToString() + "' and TypeOfCommittee='" + Typecommii.ToString() + "' order by srno desc", conn);
41                  }
42                  if (conn.State == ConnectionState.Closed)
```

## MeetingDetails_rpt.aspx.cs, line 35 (SQL Injection)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Low | **Folder** | Low |
| **Kingdom:** | Input Validation and Representation | | |

**Abstract:** On line 35 of MeetingDetails_rpt.aspx.cs, the method Page_Load() invokes a SQL query built using input coming from an untrusted source. This call could allow an attacker to modify the statement's meaning or to execute arbitrary SQL commands.

**Sink:** MeetingDetails_rpt.aspx.cs:35 SqlCommand()

```
33                  {
34                         //cmd = new SqlCommand("select title,CONVERT(VARCHAR(10), dateofmeet, 101) as dateofmeet,time,subject,venue,remarks from mpa.schdule_srno where cid=" + commi + " and TypeOfCommittee='" + Typecommii.ToString() + "' order by srno desc", conn);
35                          cmd = new SqlCommand("select ltrim(rtrim(day1)) +', the ' + ltrim(rtrim(d1)) + ' ' + ltrim(rtrim(m1)) + ' ' + ltrim(rtrim(y1)) + ' at ' + time as dateofmeet,subject,venue,remarks from mpa.schdule_srno where cid=" + commi + " and TypeOfCommittee='" + Typecommii.ToString() + "' order by srno desc", conn);
36                  }
37                  else if (commii.ToString() != "--Select Ministry--" && tit.ToString() != "--Select Title--")
```

## DataUtility.cs, line 112 (SQL Injection)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Input Validation and Representation | | |

**Abstract:** On line 112 of DataUtility.cs, the method GetDataScaler() invokes a SQL query built using input coming from an untrusted source. This call could allow an attacker to modify the statement's meaning or to execute arbitrary SQL commands.

**Source:** CommitteeMaster.aspx.cs:161 System.Web.UI.WebControls.TextBox.get_Text()

```
159                         int i;
160                         string strsql;
161                         strsql = "select Cid from mpa.Committee where Cname like '" + txtCommName.Text.Trim() + "' and Cid <> '" + ViewState["Key"] + "'";
162                         string result = dt.GetDataScaler(strsql);
```

**Sink:** DataUtility.cs:112 System.Data.SqlClient.SqlCommand.SqlCommand()

```
110             {
111                 openConnection();
112                 SqlCommand cm = new SqlCommand(strsql, con);
113                 Object dr = null;
114                 dr = cm.ExecuteScalar();
```

## PartyMaster.aspx.cs, line 142 (SQL Injection)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Unknown - Custom Issue | | |
| **Sink:** | C:/Users/APPSECMON6/AppData/Local/Fortify/AWB-4.10/workspace/audit/D__Ankita_WBT_-.NET_PENDING_CMS_mpa_SCAN_3_CMS/PartyMaster.aspx.cs:142 ..() | | |

## DataUtility.cs, line 65 (SQL Injection)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Input Validation and Representation | | |
| **Abstract:** | On line 65 of DataUtility.cs, the method ExecuteQuery() invokes a SQL query built using input coming from an untrusted source. This call could allow an attacker to modify the statement's meaning or to execute arbitrary SQL commands. | | |

**Source:** UploadToServer.aspx.cs:59 System.Web.UI.WebControls.TextBox.get_Text()

```
57          msglbl.Text = "Uploaded successfully";
58          //dt.ExecuteQuery("Update asondate set latestupdate=convert(datetime,'" +
TxtDate.Text + "', 103)");
59          dt.ExecuteQuery("Update asondate set latestupdate='" + TxtDate.Text + "'");
60      }
```

**Sink:** DataUtility.cs:65 System.Data.Common.DbCommand.set_CommandText()

```
63              datacomm.Connection = con;
64              datacomm.CommandType = CommandType.Text;
65              datacomm.CommandText = str;
66              int var;
67              var = datacomm.ExecuteNonQuery();
```

## MemberMaster.aspx.cs, line 208 (SQL Injection)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Unknown - Custom Issue | | |
| **Sink:** | C:/Users/APPSECMON6/AppData/Local/Fortify/AWB-4.10/workspace/audit/D__Ankita_WBT_-.NET_PENDING_CMS_mpa_SCAN_3_CMS/MemberMaster.aspx.cs:208 ..() | | |

## MemberMaster.aspx.cs, line 175 (SQL Injection)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Unknown - Custom Issue | | |
| **Sink:** | C:/Users/APPSECMON6/AppData/Local/Fortify/AWB-4.10/workspace/audit/D__Ankita_WBT_-.NET_PENDING_CMS_mpa_SCAN_3_CMS/MemberMaster.aspx.cs:175 ..() | | |

## DataUtility.cs, line 112 (SQL Injection)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Input Validation and Representation | | |
| **Abstract:** | On line 112 of DataUtility.cs, the method GetDataScaler() invokes a SQL query built using input coming from an untrusted source. This call could allow an attacker to modify the statement's meaning or to execute arbitrary SQL commands. | | |

**Source:** PartyMaster.aspx.cs:142 System.Web.UI.WebControls.TextBox.get_Text()

```
140             {
141                 string strsql;
142                 strsql = "select PARTY_CODE from zparty where LSTO=99 and Party_FName
like '" + txtPartyName.Text.Trim() + "'";
143                 string result = dt.GetDataScaler(strsql);
144                 if (result == null)
```

**Sink:** DataUtility.cs:112 System.Data.SqlClient.SqlCommand.SqlCommand()

```
110         {
111             openConnection();
112             SqlCommand cm = new SqlCommand(strsql, con);
113             Object dr = null;
```

```
114                                  dr = cm.ExecuteScalar();
```

**MeetingCommittee.aspx.cs, line 444 (SQL Injection)**

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Unknown - Custom Issue | | |
| **Sink:** | C:/Users/APPSECMON6/AppData/Local/Fortify/AWB-4.10/workspace/audit/D__Ankita_WBT_-.NET_PENDING_CMS_mpa_SCAN_3_CMS/MeetingCommittee.aspx.cs:444 ..() | | |

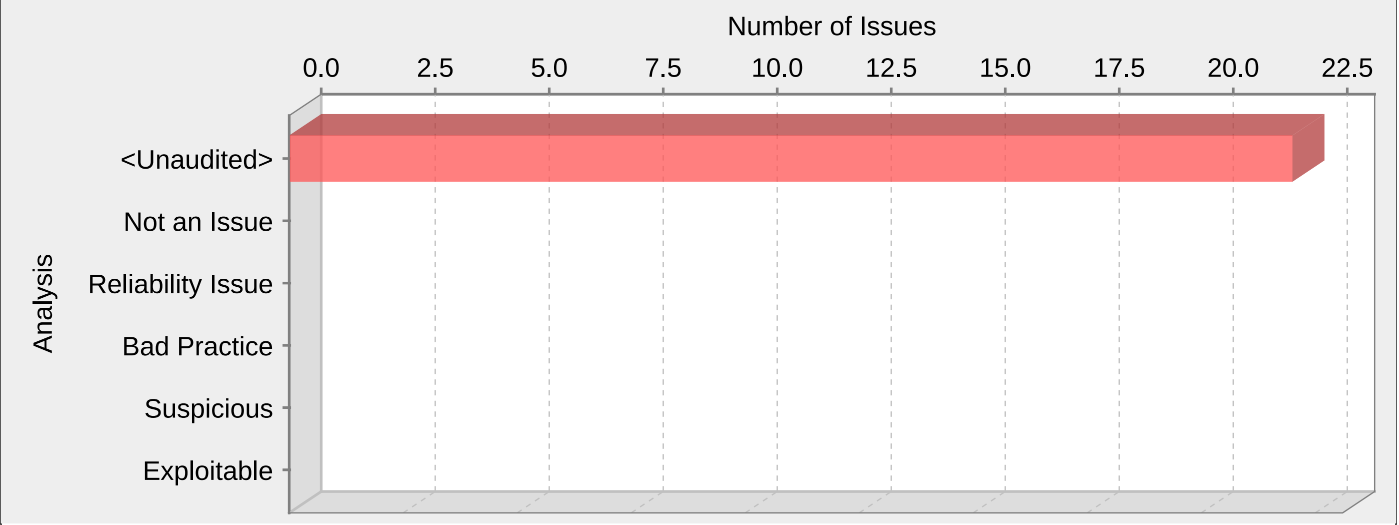
**CommitteeMaster.aspx.cs, line 161 (SQL Injection)**

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Unknown - Custom Issue | | |
| **Sink:** | C:/Users/APPSECMON6/AppData/Local/Fortify/AWB-4.10/workspace/audit/D__Ankita_WBT_-.NET_PENDING_CMS_mpa_SCAN_3_CMS/CommitteeMaster.aspx.cs:161 ..() | | |

**MeetingCommittee.aspx.cs, line 526 (SQL Injection)**

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Unknown - Custom Issue | | |
| **Sink:** | C:/Users/APPSECMON6/AppData/Local/Fortify/AWB-4.10/workspace/audit/D__Ankita_WBT_-.NET_PENDING_CMS_mpa_SCAN_3_CMS/MeetingCommittee.aspx.cs:526 ..() | | |

## Category: Poor Error Handling: Overly Broad Catch (22 Issues)



### Abstract:

The catch block handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

### Explanation:

Multiple catch blocks can get ugly and repetitive, but "condensing" catch blocks by catching a high-level class like Exception can obscure exceptions that deserve special treatment or that should not be caught at this point in the program. Catching an overly broad exception essentially defeats the purpose of .NET's typed exceptions, and can become particularly dangerous if the program grows and begins to throw new types of exceptions. The new exception types will not receive any attention.

Example: The following code excerpt handles three types of exceptions in an identical fashion.

```
try {
DoExchange();
}
catch (IOException e) {
logger.Error("DoExchange failed", e);
}
catch (FormatException e) {
logger.Error("DoExchange failed", e);
}
catch (TimeoutException e) {
logger.Error("DoExchange failed", e);
}
```

At first blush, it may seem preferable to deal with these exceptions in a single catch block, as follows:

```
try {
DoExchange();
}
catch (Exception e) {
logger.Error("DoExchange failed", e);
}
```

However, if DoExchange() is modified to throw a new type of exception that should be handled in some different kind of way, the broad catch block will prevent the compiler from pointing out the situation. Further, the new catch block will now also handle exceptions of types ApplicationException and NullReferenceException, which is not the programmer's intent.

### Recommendations:

Do not catch broad exception classes like Exception, <SystemException>, or <ApplicationException> except at the very top level of the program or thread.

### Tips:

1. The HP Fortify Secure Coding Rulepacks will not flag an overly broad catch block if the catch block in question immediately throws a new exception.

## MemberMasterHindi.aspx.cs, line 87 (Poor Error Handling: Overly Broad Catch)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Low | **Folder** | Low |
| **Kingdom:** | Errors | | |

**Abstract:** The catch block at MemberMasterHindi.aspx.cs line 87 handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

**Sink:** MemberMasterHindi.aspx.cs:87 CatchBlock()

```
85
86                      }
87                      catch (Exception ex)
88                      {
89                          ex.ToString();
```

## MemberMaster.aspx.cs, line 159 (Poor Error Handling: Overly Broad Catch)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Low | **Folder** | Low |
| **Kingdom:** | Errors | | |

**Abstract:** The catch block at MemberMaster.aspx.cs line 159 handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

**Sink:** MemberMaster.aspx.cs:159 CatchBlock()

```
157
158                     }
159                     catch (Exception ex)
160                     {
161                         System.Web.HttpContext.Current.Response.Write(ex.Message.ToString());
```

## MeetingCommittee.aspx.cs, line 246 (Poor Error Handling: Overly Broad Catch)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Low | **Folder** | Low |
| **Kingdom:** | Errors | | |

**Abstract:** The catch block at MeetingCommittee.aspx.cs line 246 handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

**Sink:** MeetingCommittee.aspx.cs:246 CatchBlock()

```
244                     cmd.Dispose();
245                 }
246                 catch (Exception ex)
247                 {
248                     System.Web.HttpContext.Current.Response.Write(ex.Message.ToString());
```

## MemberCommittee.aspx.cs, line 580 (Poor Error Handling: Overly Broad Catch)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Low | **Folder** | Low |
| **Kingdom:** | Errors | | |

**Abstract:** The catch block at MemberCommittee.aspx.cs line 580 handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

**Sink:** MemberCommittee.aspx.cs:580 CatchBlock()

```
578                     cmd.Dispose();
579                 }
580                 catch (Exception ex)
581                 {
582                     System.Web.HttpContext.Current.Response.Write(ex.Message.ToString());
```

## MemberCommittee.aspx.cs, line 545 (Poor Error Handling: Overly Broad Catch)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Low | **Folder** | Low |
| **Kingdom:** | Errors | | |

**Abstract:** The catch block at MemberCommittee.aspx.cs line 545 handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

**Sink:** MemberCommittee.aspx.cs:545 CatchBlock()

```
543                                   cmd.Dispose();
544                               }
545                               catch (Exception ex)
546                               {
547                                   System.Web.HttpContext.Current.Response.Write(ex.Message.ToString());
```

## MemberCommittee.aspx.cs, line 263 (Poor Error Handling: Overly Broad Catch)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Low | **Folder** | Low |
| **Kingdom:** | Errors | | |

**Abstract:** The catch block at MemberCommittee.aspx.cs line 263 handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

**Sink:** MemberCommittee.aspx.cs:263 CatchBlock()

```
261                                       cmd.ExecuteNonQuery();
262                                   }
263                                   catch (Exception ex)
264                                   {
265                                       System.Web.HttpContext.Current.Response.Write(ex.Message.ToString());
```

## MeetingCommittee.aspx.cs, line 336 (Poor Error Handling: Overly Broad Catch)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Low | **Folder** | Low |
| **Kingdom:** | Errors | | |

**Abstract:** The catch block at MeetingCommittee.aspx.cs line 336 handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

**Sink:** MeetingCommittee.aspx.cs:336 CatchBlock()

```
334                                   cmd.Dispose();
335                               }
336                               catch (Exception ex)
337                               {
338                                   System.Web.HttpContext.Current.Response.Write(ex.Message.ToString());
```

## MemberCommittee.aspx.cs, line 321 (Poor Error Handling: Overly Broad Catch)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Low | **Folder** | Low |
| **Kingdom:** | Errors | | |

**Abstract:** The catch block at MemberCommittee.aspx.cs line 321 handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

**Sink:** MemberCommittee.aspx.cs:321 CatchBlock()

```
319                                   cmd.Dispose();
320                               }
321                               catch (Exception ex)
322                               {
323                                   System.Web.HttpContext.Current.Response.Write(ex.Message.ToString());
```

## CareTakerRpt.aspx.cs, line 94 (Poor Error Handling: Overly Broad Catch)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Low | **Folder** | Low |
| **Kingdom:** | Errors | | |

**Abstract:** The catch block at CareTakerRpt.aspx.cs line 94 handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

**Sink:** CareTakerRpt.aspx.cs:94 CatchBlock()

```
92                                    }
93                                }
```

```
                    catch (Exception ex)
                    {
                        System.Web.HttpContext.Current.Response.Write(ex.Message.ToString());
```

## MeetingCommittee.aspx.cs, line 366 (Poor Error Handling: Overly Broad Catch)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Low | **Folder** | Low |
| **Kingdom:** | Errors | | |

**Abstract:** The catch block at MeetingCommittee.aspx.cs line 366 handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

**Sink:** MeetingCommittee.aspx.cs:366 CatchBlock()

```
364                         cmd.Dispose();
365                     }
366                     catch (Exception ex)
367                     {
368                         System.Web.HttpContext.Current.Response.Write(ex.Message.ToString());
```

## MeetingCommittee.aspx.cs, line 291 (Poor Error Handling: Overly Broad Catch)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Low | **Folder** | Low |
| **Kingdom:** | Errors | | |

**Abstract:** The catch block at MeetingCommittee.aspx.cs line 291 handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

**Sink:** MeetingCommittee.aspx.cs:291 CatchBlock()

```
289                         cmd.Dispose();
290                     }
291                     catch (Exception ex)
292                     {
293                         System.Web.HttpContext.Current.Response.Write(ex.Message.ToString());
```

## MeetingAttendance.aspx.cs, line 285 (Poor Error Handling: Overly Broad Catch)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Low | **Folder** | Low |
| **Kingdom:** | Errors | | |

**Abstract:** The catch block at MeetingAttendance.aspx.cs line 285 handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

**Sink:** MeetingAttendance.aspx.cs:285 CatchBlock()

```
283                         CheckBoxList4.DataBind();
284                     }
285                     catch (Exception ex)
286                     {
287                         System.Web.HttpContext.Current.Response.Write(ex.Message.ToString());
```

## Attendance_Report.aspx.cs, line 255 (Poor Error Handling: Overly Broad Catch)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Low | **Folder** | Low |
| **Kingdom:** | Errors | | |

**Abstract:** The catch block at Attendance_Report.aspx.cs line 255 handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

**Sink:** Attendance_Report.aspx.cs:255 CatchBlock()

```
253                         GridView3.DataBind();
254                     }
255                     catch (Exception ex)
256                     {
257                         System.Web.HttpContext.Current.Response.Write(ex.Message.ToString());
```

## MemberCommittee.aspx.cs, line 510 (Poor Error Handling: Overly Broad Catch)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Low | **Folder** | Low |

**Kingdom:** Errors

**Abstract:** The catch block at MemberCommittee.aspx.cs line 510 handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

**Sink:** MemberCommittee.aspx.cs:510 CatchBlock()

```
508                 cmd.Dispose();
509             }
510             catch (Exception ex)
511             {
512                 System.Web.HttpContext.Current.Response.Write(ex.Message.ToString());
```

## Committee_rpt.aspx.cs, line 47 (Poor Error Handling: Overly Broad Catch)

**Fortify Priority:** Low **Folder** Low

**Kingdom:** Errors

**Abstract:** The catch block at Committee_rpt.aspx.cs line 47 handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

**Sink:** Committee_rpt.aspx.cs:47 CatchBlock()

```
45                  GR1.DataBind();
46              }
47              catch (Exception ex)
48              {
49                  System.Web.HttpContext.Current.Response.Write(ex.Message.ToString());
```

## MemberMaster.aspx.cs, line 819 (Poor Error Handling: Overly Broad Catch)

**Fortify Priority:** Low **Folder** Low

**Kingdom:** Errors

**Abstract:** The catch block at MemberMaster.aspx.cs line 819 handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

**Sink:** MemberMaster.aspx.cs:819 CatchBlock()

```
817                 cmbConst.Items.Insert(0, "--Select Const--");
818             }
819             catch (Exception ex)
820             {
821                 Response.Write(ex.Message.ToString());
```

## MemberMaster.aspx.cs, line 91 (Poor Error Handling: Overly Broad Catch)

**Fortify Priority:** Low **Folder** Low

**Kingdom:** Errors

**Abstract:** The catch block at MemberMaster.aspx.cs line 91 handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

**Sink:** MemberMaster.aspx.cs:91 CatchBlock()

```
89                  cmbparty.Items.Insert(0, "--Select Party--");
90              }
91              catch (Exception ex)
92              {
93                  System.Web.HttpContext.Current.Response.Write(ex.Message.ToString());
```

## CaretakerReport.aspx.cs, line 77 (Poor Error Handling: Overly Broad Catch)

**Fortify Priority:** Low **Folder** Low

**Kingdom:** Errors

**Abstract:** The catch block at CaretakerReport.aspx.cs line 77 handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

**Sink:** CaretakerReport.aspx.cs:77 CatchBlock()

```
75                  cmbtitle.Items.Insert(0, "--Select Title--");
```

```
76                             }
77                             catch (Exception ex)
78                             {
79                                 System.Web.HttpContext.Current.Response.Write(ex.Message.ToString());
```

## MemberMaster.aspx.cs, line 125 (Poor Error Handling: Overly Broad Catch)

**Fortify Priority:** Low **Folder** Low
**Kingdom:** Errors

**Abstract:** The catch block at MemberMaster.aspx.cs line 125 handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

**Sink:** MemberMaster.aspx.cs:125 CatchBlock()

```
123                                cmbState.Items.Insert(0, "--Select State--");
124                            }
125                            catch (Exception ex)
126                            {
127                                System.Web.HttpContext.Current.Response.Write(ex.Message.ToString());
```

## code.cs, line 75 (Poor Error Handling: Overly Broad Catch)

**Fortify Priority:** Low **Folder** Low
**Kingdom:** Errors

**Abstract:** The catch block at code.cs line 75 handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

**Sink:** code.cs:75 CatchBlock()

```
73                                 }
74                             }
75                             catch (Exception ex)
76                             {
77                                 System.Web.HttpContext.Current.Response.Write(ex.Message.ToString());
```

## MeetingAttendanceReport.aspx.cs, line 81 (Poor Error Handling: Overly Broad Catch)

**Fortify Priority:** Low **Folder** Low
**Kingdom:** Errors

**Abstract:** The catch block at MeetingAttendanceReport.aspx.cs line 81 handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

**Sink:** MeetingAttendanceReport.aspx.cs:81 CatchBlock()

```
79                                 cmbtitlee.Items.Insert(0, "--Select Title--");
80                             }
81                             catch (Exception ex)
82                             {
83                                 System.Web.HttpContext.Current.Response.Write(ex.Message.ToString());
```

## MeetingAttendanceRpt.aspx.cs, line 274 (Poor Error Handling: Overly Broad Catch)

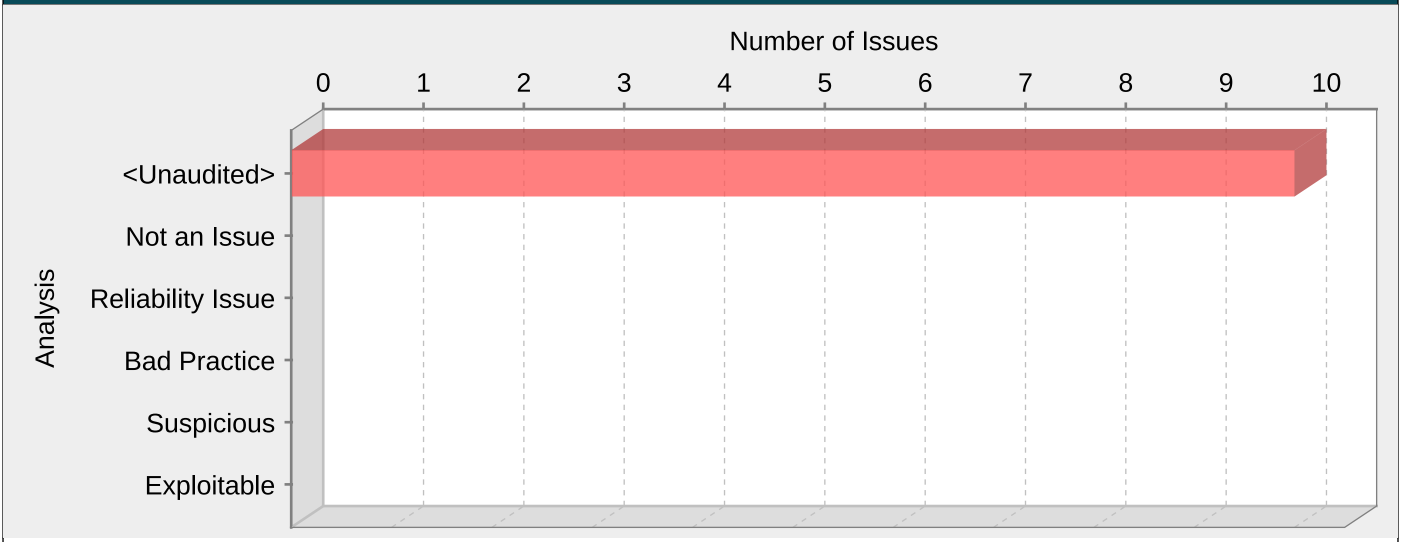
**Fortify Priority:** Low **Folder** Low
**Kingdom:** Errors

**Abstract:** The catch block at MeetingAttendanceRpt.aspx.cs line 274 handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

**Sink:** MeetingAttendanceRpt.aspx.cs:274 CatchBlock()

```
272
273                            }
274                            catch (Exception ex)
275                            {
276                                System.Web.HttpContext.Current.Response.Write(ex.Message.ToString());
```

## Category: Cross-Site Scripting: Persistent (10 Issues)



### Abstract:

Sending unvalidated data to a web browser can result in the browser executing malicious code.

### Explanation:

Cross-site scripting (XSS) vulnerabilities occur when:

1. Data enters a web application through an untrusted source. In the case of Persistent (also known as Stored) XSS, the untrusted source is typically a database or other back-end datastore, while in the case of Reflected XSS it is typically a web request.

2. The data is included in dynamic content that is sent to a web user without being validated.

The malicious content sent to the web browser often takes the form of a segment of JavaScript, but may also include HTML, Flash or any other type of code that the browser may execute. The variety of attacks based on XSS is almost limitless, but they commonly include transmitting private data like cookies or other session information to the attacker, redirecting the victim to web content controlled by the attacker, or performing other malicious operations on the user's machine under the guise of the vulnerable site.

Example 1: The following ASP.NET Web Form queries a database for an employee with a given employee ID and prints the name corresponding with the ID.

```
<script runat="server">
...
string query = "select * from emp where id=" + eid;
sda = new SqlDataAdapter(query, conn);
DataTable dt = new DataTable();
sda.Fill(dt);
string name = dt.Rows[0]["Name"];
...
EmployeeName.Text = name;
</script>
```

Where EmployeeName is a form control defined as follows:

```
<form runat="server">
...
<asp:Label id="EmployeeName" runat="server">
...
</form>
```

Example 2: The following ASP.NET code segment is functionally equivalent to Example 1 above, but implements all of the form elements programmatically.

```
protected System.Web.UI.WebControls.Label EmployeeName;
...
string query = "select * from emp where id=" + eid;
sda = new SqlDataAdapter(query, conn);
```

```
DataTable dt = new DataTable();
sda.Fill(dt);
string name = dt.Rows[0]["Name"];
...
EmployeeName.Text = name;
```

These code examples function correctly when the values of name are well-behaved, but they do nothing to prevent exploits if they are not. 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. This type of exploit, known as Persistent (or Stored) XSS, is particularly insidious because the indirection caused by the data store makes it more difficult to identify the threat and increases the possibility that the attack will affect multiple users. XSS got its start in this form 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.

Example 3: The following ASP.NET Web Form reads an employee ID number from an HTTP request and displays it to the user.

```
<script runat="server">
...
EmployeeID.Text = Login.Text;
...
</script>
```

Where Login and EmployeeID are form controls defined as follows:

```
<form runat="server">
<asp:TextBox runat="server" id="Login"/>
...
<asp:Label runat="server" id="EmployeeID"/>
</form>
```

Example 4: The following ASP.NET code segment shows the programmatic way to implement Example 3 above.

```
protected System.Web.UI.WebControls.TextBox Login;
protected System.Web.UI.WebControls.Label EmployeeID;
...
EmployeeID.Text = Login.Text;
```

As in Example 1 and 2, these examples operate correctly if Login contains only standard alphanumeric text. If Login 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 in order to lure victims into clicking a link. When the victims click the link, they unwittingly reflect the malicious content through the vulnerable web application and back to their own computers. This mechanism of exploiting vulnerable web applications is known as Reflected XSS.

As the examples demonstrate, XSS vulnerabilities are caused by code that includes unvalidated data in an HTTP response. There are three vectors by which an XSS attack can reach a victim:

- As in Examples 1 and 2, the application stores dangerous data in a database or other trusted data store. The dangerous data is subsequently read back into the application and included in dynamic content. Persistent XSS exploits occur when an attacker injects dangerous content into a data store that is later read 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.

- As in Examples 3 and 4, data is read directly from the HTTP request and reflected back in the HTTP response. Reflected XSS exploits occur when an attacker causes a user to supply dangerous content to a vulnerable web application, which is then reflected back to the user 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 victims. URLs constructed in this manner constitute the core of many phishing schemes, whereby an attacker convinces victims to visit a URL that refers to a vulnerable site. After the site reflects the attacker's content back to the user, the content is executed and proceeds to transfer private information, such as cookies that may include session information, from the user's machine to the attacker or perform other nefarious activities.

- A source outside the application stores dangerous data in a database or other data store, and the dangerous data is subsequently read back into the application as trusted data and included in dynamic content.

A number of modern web frameworks provide mechanisms for performing validation of user input. ASP.NET Request Validation and WCF are among them. To highlight the unvalidated sources of input, the rulepacks dynamically re-prioritize the issues reported by HP Fortify Static Code Analyzer by lowering their probability of exploit and providing pointers to the supporting evidence whenever the framework validation mechanism is in use. In case of ASP.NET Request Validation, we also provide evidence for when validation is explicitly disabled. We refer to this feature as Context-Sensitive Ranking. To further assist the HP Fortify user with the auditing process, the HP Fortify Software Security Research Group makes available the Data Validation project template that groups the issues into folders based on the validation mechanism applied to their source of input.

## Recommendations:

The solution to XSS is to ensure that validation occurs in the correct places and checks for the correct properties.

Since XSS vulnerabilities occur when an application includes malicious data in its output, one logical approach is to validate data immediately before it leaves the application. However, because web applications often have complex and intricate code for generating dynamic content, this method is prone to errors of omission (missing validation). An effective way to mitigate this risk is to also perform input validation for XSS.

Web applications must validate their input to prevent other vulnerabilities, such as SQL injection, so augmenting an application's existing input validation mechanism to include checks for XSS is generally relatively easy. Despite its value, input validation for XSS does not take the place of rigorous output validation. An application may accept input through a shared data store or other trusted source, and that data store may accept input from a source that does not perform adequate input validation. Therefore, the application cannot implicitly rely on the safety of this or any other data. This means the best way to prevent XSS vulnerabilities is to validate everything that enters the application and leaves the application destined for the user.

The most secure approach to validation for XSS is to create a whitelist of safe characters that are allowed to appear in HTTP content and accept input composed exclusively of characters in the approved set. For example, a valid username might only include alpha-numeric characters or a phone number might only include digits 0-9. However, this solution is often infeasible in web applications because many characters that have special meaning to the browser should still be considered valid input once they are encoded, such as a web design bulletin board that must accept HTML fragments from its users.

A more flexible, but less secure approach is known as blacklisting, which selectively rejects or escapes potentially dangerous characters before using the input. In order to form such a list, you first need to understand the set of characters that hold special meaning for web browsers. Although the HTML standard defines what characters have special meaning, many web browsers try to correct common mistakes in HTML and may treat other characters as special in certain contexts, which is why we do not encourage the use of blacklists as a means to prevent XSS. The CERT(R) Coordination Center at the Software Engineering Institute at Carnegie Mellon University provides the following details about special characters in various contexts [1]:

In the content of a block-level element (in the middle of a paragraph of text):

- "<" is special because it introduces a tag.

- "&" is special because it introduces a character entity.

- ">" is special because some browsers treat it as special, on the assumption that the author of the page intended to include an opening "<", but omitted it in error.

The following principles apply to attribute values:

- In attribute values enclosed with double quotes, the double quotes are special because they mark the end of the attribute value.

- In attribute values enclosed with single quote, the single quotes are special because they mark the end of the attribute value.

- In attribute values without any quotes, white-space characters, such as space and tab, are special.

- "&" is special when used with certain attributes, because it introduces a character entity.

In URLs, for example, a search engine might provide a link within the results page that the user can click to re-run the search. This can be implemented by encoding the search query inside the URL, which introduces additional special characters:

- Space, tab, and new line are special because they mark the end of the URL.

- "&" is special because it either introduces a character entity or separates CGI parameters.

- Non-ASCII characters (that is, everything above 128 in the ISO-8859-1 encoding) are not allowed in URLs, so they are considered to be special in this context.

- The "%" symbol must be filtered from input anywhere parameters encoded with HTTP escape sequences are decoded by server -side code. For example, "%" must be filtered if input such as "%68%65%6C%6C%6F" becomes "hello" when it appears on the web page in question.

Within the body of a <SCRIPT> </SCRIPT>:

- The semicolon, parenthesis, curly braces, and new line should be filtered in situations where text could be inserted directly into a pre-existing script tag.

Server-side scripts:

- Server-side scripts that convert any exclamation characters (!) in input to double-quote characters (") on output might require additional filtering.

Other possibilities:

- If an attacker submits a request in UTF-7, the special character '<' appears as '+ADw-' and may bypass filtering. If the output is included in a page that does not explicitly specify an encoding format, then some browsers try to intelligently identify the encoding based on the content (in this case, UTF-7).

Once you identify the correct points in an application to perform validation for XSS attacks and what special characters the validation should consider, the next challenge is to identify how your validation handles special characters. If special characters are not considered valid input to the application, then you can reject any input that contains special characters as invalid. A second option in this situation is to remove special characters with filtering. However, filtering has the side effect of changing any visual representation of the filtered content and may be unacceptable in circumstances where the integrity of the input must be preserved for display.

If input containing special characters must be accepted and displayed accurately, validation must encode any special characters to remove their significance. A complete list of ISO 8859-1 encoded values for special characters is provided as part of the official HTML specification [2].

Many application servers attempt to limit an application's exposure to cross-site scripting vulnerabilities by providing implementations for the functions responsible for setting certain specific HTTP response content that perform validation for the characters essential to a cross-site scripting attack. Do not rely on the server running your application to make it secure. When an application is developed there are no guarantees about what application servers it will run on during its lifetime. As standards and known exploits evolve, there are no guarantees that application servers will also stay in sync.

**Tips:**

1. The HP Fortify Secure Coding Rulepacks warn about SQL Injection and Access Control: Database issues when untrusted data is written to a database and also treat the database as a source of untrusted data, which can lead to XSS vulnerabilities. If the database is a trusted resource in your environment, use custom filters to filter out dataflow issues that include the DATABASE taint flag or originate from database sources. Nonetheless, it is often still a good idea to validate everything read from the database.

2. Even though URL encoding untrusted data protects against many XSS attacks, some browsers (specifically, Internet Explorer 6 and 7 and possibly others) automatically decode content at certain locations within the Document Object Model (DOM) prior to passing it to the JavaScript interpreter. To reflect this danger, the rulepacks no longer treat URL encoding routines as sufficient to protect against Cross-Site Scripting. Data values that are URL encoded and subsequently output will cause Fortify to report Cross-Site Scripting: Poor Validation vulnerabilities.

3. Fortify RTA adds protection against this category.

## MemberMasterHindi.aspx.cs, line 147 (Cross-Site Scripting: Persistent)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Input Validation and Representation | | |

**Abstract:** The method MakePartyTable() in MemberMasterHindi.aspx.cs sends unvalidated data to a web browser on line 147, which can result in the browser executing malicious code.

**Source:** DataUtility.cs:104 System.Data.Common.DbDataAdapter.Fill()

```
102                 SqlDataAdapter adp = new SqlDataAdapter(strsql, con);
103                 DataTable dr = new DataTable();
104                 adp.Fill(dr);
105                 closeConnection();
106                 dispose();
```

**Sink:** MemberMasterHindi.aspx.cs:147 System.Web.UI.WebControls.ListItemCollection.Add()

```
145                 lt[pc].Value = mytable.Rows[pc][0].ToString();
146                 restvalue = restvalue + "'" + mytable.Rows[pc][0].ToString() + "',";
147                 rdoparty.Items.Add(lt[pc]);
148             }
149             ViewState["restvalue"] = restvalue.Substring(0, restvalue.Length - 1) + ")";
```

## Sms.aspx.cs, line 85 (Cross-Site Scripting: Persistent)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Input Validation and Representation | | |

**Abstract:** The method Page_Load() in Sms.aspx.cs sends unvalidated data to a web browser on line 85, which can result in the browser executing malicious code.

**Source:** Sms.aspx.cs:261 System.Data.SqlClient.SqlCommand.ExecuteReader()

```
259                 SqlCommand cmd = new SqlCommand("[dbo].[MeetingOfficer_SP]", con);
```

```
260                                cmd.CommandType = CommandType.StoredProcedure;
261                                dr = cmd.ExecuteReader();
262                                CheckBoxList1.Items.Clear();
263                                while (dr.Read())
```

**Sink:** Sms.aspx.cs:85 System.Web.UI.WebControls.ListItemCollection.Add()

```
83                                         item.Value = dr["SRNo"].ToString();
84                                         item.Selected = Convert.ToBoolean(dr["IsSelected"]);
85                                         CheckBoxList1.Items.Add(item);
86                                     }
```

## MemberMaster.aspx.cs, line 257 (Cross-Site Scripting: Persistent)

**Fortify Priority:** Critical **Folder** Critical
**Kingdom:** Input Validation and Representation

**Abstract:** The method MakePartyTable() in MemberMaster.aspx.cs sends unvalidated data to a web browser on line 257, which can result in the browser executing malicious code.

**Source:** MemberMaster.aspx.cs:248 System.Data.Common.DbDataAdapter.Fill()

```
246                            cmd.CommandType = CommandType.StoredProcedure;
247                            SqlDataAdapter adp = new SqlDataAdapter(cmd);
248                            adp.Fill(mytable);
249                            //---------------
250                            restvalue = "(";
```

**Sink:** MemberMaster.aspx.cs:257 System.Web.UI.WebControls.ListItemCollection.Add()

```
255                                lt[pc].Value = mytable.Rows[pc][0].ToString();
256                                restvalue = restvalue +"'" + mytable.Rows[pc][0].ToString() + "',";
257                                rdoparty.Items.Add(lt[pc]);
258                            }
259                            ViewState["restvalue"] = restvalue.Substring(0, restvalue.Length - 1) + ")";
```

## Sms.aspx.cs, line 85 (Cross-Site Scripting: Persistent)

**Fortify Priority:** Critical **Folder** Critical
**Kingdom:** Input Validation and Representation

**Abstract:** The method Page_Load() in Sms.aspx.cs sends unvalidated data to a web browser on line 85, which can result in the browser executing malicious code.

**Source:** Sms.aspx.cs:139 System.Data.SqlClient.SqlCommand.ExecuteReader()

```
137                                    con.Open();
138                                }
139                                dr = cmd1.ExecuteReader();
140                                if (dr.HasRows)
141                                {
```

**Sink:** Sms.aspx.cs:85 System.Web.UI.WebControls.ListItemCollection.Add()

```
83                                         item.Value = dr["SRNo"].ToString();
84                                         item.Selected = Convert.ToBoolean(dr["IsSelected"]);
85                                         CheckBoxList1.Items.Add(item);
86                                     }
```

## Sms.aspx.cs, line 270 (Cross-Site Scripting: Persistent)

**Fortify Priority:** Critical **Folder** Critical
**Kingdom:** Input Validation and Representation

**Abstract:** The method Clearbtn_Click() in Sms.aspx.cs sends unvalidated data to a web browser on line 270, which can result in the browser executing malicious code.

**Source:** Sms.aspx.cs:139 System.Data.SqlClient.SqlCommand.ExecuteReader()

```
137                                    con.Open();
138                                }
139                                dr = cmd1.ExecuteReader();
140                                if (dr.HasRows)
141                                {
```

**Sink:** Sms.aspx.cs:270 System.Web.UI.WebControls.ListItemCollection.Add()

```
268                                    item.Value = dr["SRNo"].ToString();
```

```
269                             item.Selected = Convert.ToBoolean(dr["IsSelected"]);
270                             CheckBoxList1.Items.Add(item);
271                         }
272                         cmd.Dispose();
```

## Sms.aspx.cs, line 270 (Cross-Site Scripting: Persistent)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Input Validation and Representation | | |

**Abstract:** The method Clearbtn_Click() in Sms.aspx.cs sends unvalidated data to a web browser on line 270, which can result in the browser executing malicious code.

**Source:** Sms.aspx.cs:78 System.Data.SqlClient.SqlCommand.ExecuteReader()

```
76                              SqlCommand cmd = new SqlCommand("[dbo].[MeetingOfficer_SP]", con);
77                              cmd.CommandType = CommandType.StoredProcedure;
78                              dr = cmd.ExecuteReader();
79                              while (dr.Read())
80                              {
```

**Sink:** Sms.aspx.cs:270 System.Web.UI.WebControls.ListItemCollection.Add()

```
268                             item.Value = dr["SRNo"].ToString();
269                             item.Selected = Convert.ToBoolean(dr["IsSelected"]);
270                             CheckBoxList1.Items.Add(item);
271                         }
272                         cmd.Dispose();
```

## Sms.aspx.cs, line 270 (Cross-Site Scripting: Persistent)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Input Validation and Representation | | |

**Abstract:** The method Clearbtn_Click() in Sms.aspx.cs sends unvalidated data to a web browser on line 270, which can result in the browser executing malicious code.

**Source:** Sms.aspx.cs:325 System.Data.SqlClient.SqlCommand.ExecuteReader()

```
323                         cmd1 = new SqlCommand("[dbo].[MeetingOfficerMobile_SP]", con);
324                         cmd1.CommandType = CommandType.StoredProcedure;
325                         dr = cmd1.ExecuteReader();
326                         if (dr.HasRows)
327                         {
```

**Sink:** Sms.aspx.cs:270 System.Web.UI.WebControls.ListItemCollection.Add()

```
268                             item.Value = dr["SRNo"].ToString();
269                             item.Selected = Convert.ToBoolean(dr["IsSelected"]);
270                             CheckBoxList1.Items.Add(item);
271                         }
272                         cmd.Dispose();
```

## Sms.aspx.cs, line 270 (Cross-Site Scripting: Persistent)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Input Validation and Representation | | |

**Abstract:** The method Clearbtn_Click() in Sms.aspx.cs sends unvalidated data to a web browser on line 270, which can result in the browser executing malicious code.

**Source:** Sms.aspx.cs:261 System.Data.SqlClient.SqlCommand.ExecuteReader()

```
259                         SqlCommand cmd = new SqlCommand("[dbo].[MeetingOfficer_SP]", con);
260                         cmd.CommandType = CommandType.StoredProcedure;
261                         dr = cmd.ExecuteReader();
262                         CheckBoxList1.Items.Clear();
263                         while (dr.Read())
```

**Sink:** Sms.aspx.cs:270 System.Web.UI.WebControls.ListItemCollection.Add()

```
268                             item.Value = dr["SRNo"].ToString();
269                             item.Selected = Convert.ToBoolean(dr["IsSelected"]);
270                             CheckBoxList1.Items.Add(item);
271                         }
272                         cmd.Dispose();
```

## Sms.aspx.cs, line 85 (Cross-Site Scripting: Persistent)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Input Validation and Representation | | |

**Abstract:** The method Page_Load() in Sms.aspx.cs sends unvalidated data to a web browser on line 85, which can result in the browser executing malicious code.

**Source:** Sms.aspx.cs:78 System.Data.SqlClient.SqlCommand.ExecuteReader()

```
76                      SqlCommand cmd = new SqlCommand("[dbo].[MeetingOfficer_SP]", con);
77                      cmd.CommandType = CommandType.StoredProcedure;
78                      dr = cmd.ExecuteReader();
79                      while (dr.Read())
80                      {
```

**Sink:** Sms.aspx.cs:85 System.Web.UI.WebControls.ListItemCollection.Add()

```
83                          item.Value = dr["SRNo"].ToString();
84                          item.Selected = Convert.ToBoolean(dr["IsSelected"]);
85                          CheckBoxList1.Items.Add(item);
86                      }
```

## Sms.aspx.cs, line 85 (Cross-Site Scripting: Persistent)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Critical | **Folder** | Critical |
| **Kingdom:** | Input Validation and Representation | | |

**Abstract:** The method Page_Load() in Sms.aspx.cs sends unvalidated data to a web browser on line 85, which can result in the browser executing malicious code.

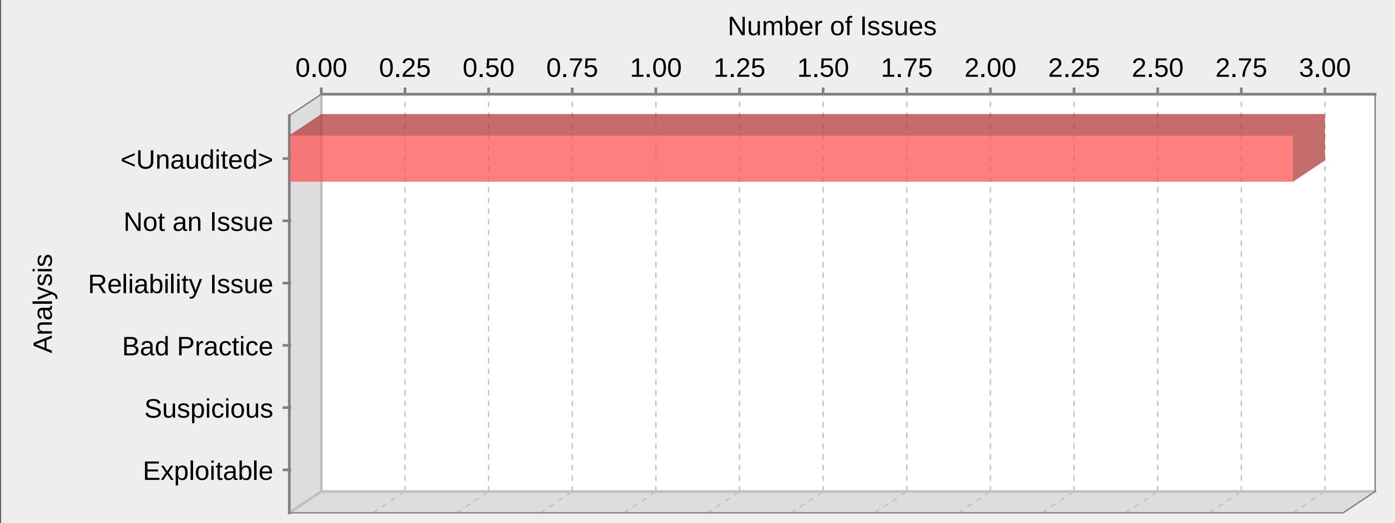
**Source:** Sms.aspx.cs:325 System.Data.SqlClient.SqlCommand.ExecuteReader()

```
323                 cmd1 = new SqlCommand("[dbo].[MeetingOfficerMobile_SP]", con);
324                 cmd1.CommandType = CommandType.StoredProcedure;
325                 dr = cmd1.ExecuteReader();
326                 if (dr.HasRows)
327                 {
```

**Sink:** Sms.aspx.cs:85 System.Web.UI.WebControls.ListItemCollection.Add()

```
83                          item.Value = dr["SRNo"].ToString();
84                          item.Selected = Convert.ToBoolean(dr["IsSelected"]);
85                          CheckBoxList1.Items.Add(item);
86                      }
```

## Category: Unreleased Resource: Database (3 Issues)



### Abstract:

The program can potentially fail to release a system resource.

### Explanation:

The program can potentially fail to release a system resource.

Resource leaks have at least two common causes:

- Error conditions and other exceptional circumstances.

- Confusion over which part of the program is responsible for releasing the resource.

Most unreleased resource issues result in general software reliability problems, but if an attacker can intentionally trigger a resource leak, the attacker might be able to launch a denial of service attack by depleting the resource pool.

Example 1: The following method never closes the file handle it opens. The Finalize() method for StreamReader eventually calls Close(), but there is no guarantee as to how long it will take before the Finalize() method is invoked. In fact, there is no guarantee that Finalize() will ever be invoked. In a busy environment, this can result in the VM using up all of its available file handles.

```
private void processFile(string fName) {
StreamWriter sw = new StreamWriter(fName);
string line;
while ((line = sr.ReadLine()) != null)
processLine(line);
}
```

Example 2: Under normal conditions the following code executes a database query, processes the results returned by the database, and closes the allocated SqlConnection object. But if an exception occurs while executing the SQL or processing the results, the SqlConnection object is not closed. If this happens often enough, the database will run out of available cursors and not be able to execute any more SQL queries.

```
...
SqlConnection conn = new SqlConnection(connString);
SqlCommand cmd = new SqlCommand(queryString);
cmd.Connection = conn;
conn.Open();
SqlDataReader rdr = cmd.ExecuteReader();
HarvestResults(rdr);
conn.Connection.Close();
...
```

### Recommendations:

Never rely on Finalize() to reclaim resources. In order for an object's Finalize() method to be invoked, the garbage collector must determine that the object is eligible for garbage collection. Because the garbage collector is not required to run unless the VM is low on memory, there is no guarantee that an object's Finalize() method will be invoked in an expedient fashion, if it is ever invoked at all (the language does not guarantee that it will be). When the garbage collector finally does run, it can cause a large number of resources to be reclaimed in a short period of time, which can lead to "bursty" performance and lower overall system throughput. The effect becomes more pronounced as the load on the system increases.

Instead of explicitly closing objects that manage resources, use the C# keyword 'using', which employs the IDisposable interface to perform a cleanup. The following two blocks of code achieve the same result:

The following code uses the finally keyword:

```
StreamReader sr;
try {
sr = new StreamReader(myFileStream);
doWork(sr);
} finally {
if (sr != null) {
sr.Close();
}
}
```

The following code uses the using keyword:

```
using (StreamReader sr = new StreamReader(myFileStream)) {
doWork(sr);
}
```

## Login.aspx.cs, line 219 (Unreleased Resource: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Code Quality | | |

**Abstract:** The function submit_Click() in Login.aspx.cs sometimes fails to release a system resource allocated by ExecuteReader() on line 219.

**Sink:** Login.aspx.cs:219 dr2 = ExecuteReader()

```
217                 cmd2.Parameters.AddWithValue("@uname", uname);
218                 cmd2.Parameters.AddWithValue("@UType", utype);
219                 SqlDataReader dr2 = cmd2.ExecuteReader();
220                 if (dr2.HasRows)
221                 {
```

## Email.aspx.cs, line 160 (Unreleased Resource: Database)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Code Quality | | |

**Abstract:** The function CheckBoxList1_SelectedIndexChanged() in Email.aspx.cs sometimes fails to release a system resource allocated by SqlConnection() on line 155.

**Sink:** Email.aspx.cs:160 cmd = new SqlCommand(..., this.con)

```
158                     con.Open();
159                 }
160                 SqlCommand cmd = new SqlCommand("[dbo].[MeetingOfficerUpdate_SP]", con);
161                 cmd.CommandType = CommandType.StoredProcedure;
```

## Login.aspx.cs, line 154 (Unreleased Resource: Database)

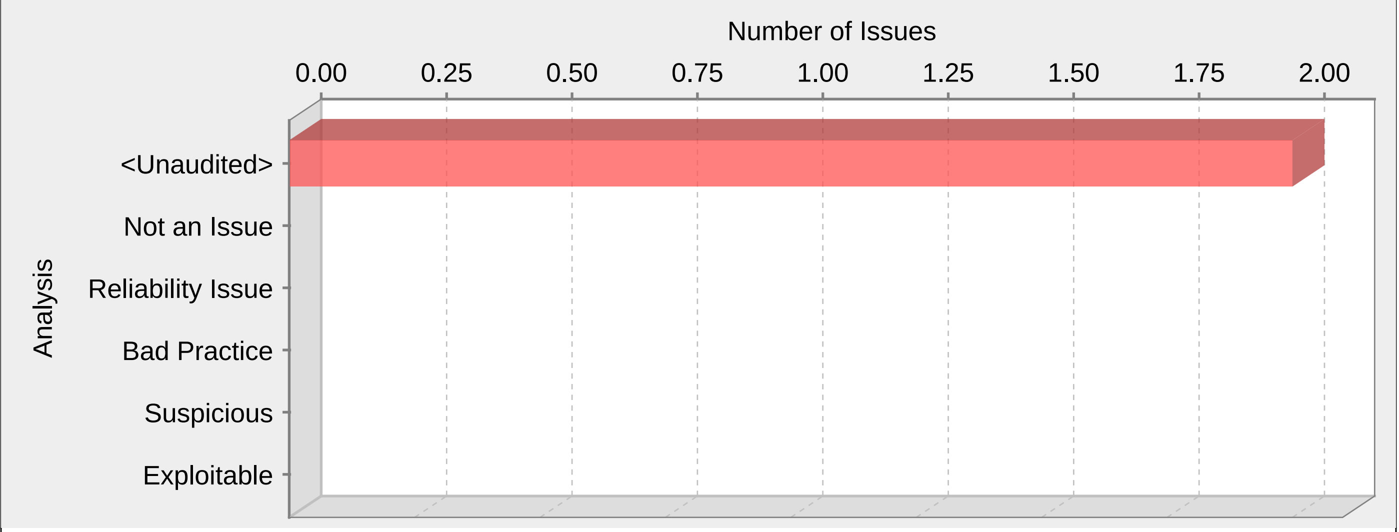
| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Code Quality | | |

**Abstract:** The function submit_Click() in Login.aspx.cs sometimes fails to release a system resource allocated by ExecuteReader() on line 154.

**Sink:** Login.aspx.cs:154 dr3 = ExecuteReader()

```
152                 cmd3.Parameters.AddWithValue("@uname", uname);
153                 cmd3.Parameters.Clear();
154                 SqlDataReader dr3 = cmd3.ExecuteReader();
155                 if (dr3.HasRows)
156                 {
```

## Category: Unreleased Resource: Unmanaged Object (2 Issues)



### Abstract:

The program fails to dispose of a managed object that utilizes unmanaged system resources.

### Explanation:

The program fails to properly dispose of a managed object that uses unmanaged system resources.

Failure to properly dispose of a managed object that uses unmanaged system resources has at least two common causes:

- Error conditions and other exceptional circumstances.
- Confusion over which part of the program is responsible for releasing the resource.

A small subset of managed .NET objects use unmanaged system resources. .NET's Garbage Collector may not free the original managed objects in a predictable way. As such, the application may run out of available memory as the Garbage Collector is unaware of the memory consumed by the unmanaged resources. Most unmanaged resource leak issues result in general software reliability problems, but if an attacker can intentionally trigger an unmanaged resource leak, the attacker might be able to launch a denial of service attack by depleting the unmanaged resource pool.

Example 1: The following method creates a managed Bitmap Object from an incoming stream incomingStream. The Bitmap is manipulated and persisted to the outgoing stream outgoingStream. The Dispose() method of incomingBitmap and outgoingBitmap is never explicitly called.

Normally, one can safely rely upon the Garbage Collector to do this at a safe time for managed objects that do not use unmanaged system resources. The Garbage Collector calls Bitmap.Dispose() when it sees fit. However, the Bitmap object utilizes scarce, unmanaged system resources. The Garbage Collector may fail to call Dispose() before the unmanaged resource pool is depleted.

```
private void processBitmap(Stream incomingStream, Stream outgoingStream, int thumbnailSize)
{
Bitmap incomingBitmap = (Bitmap)System.Drawing.Image.FromStream(incomingStream);
bool validBitmap = validateBitmap(incomingBitmap);
if (!validBitmap)
throw new ValidationException(incomingBitmap);
Bitmap outgoingBitmap = new Bitmap(incomingBitmap, new Size(thumbnailSize, thumbnailSize));
outgoingBitmap.Save(outgoingStream, ImageFormat.Bmp);
}
```

### Recommendations:

Never rely on .NET's Garbage Collector to call Dispose() on objects that use unmanaged system resources. In order for an object's Dispose() method to be invoked, the garbage collector must determine that the object is eligible for garbage collection. Because the garbage collector is not required to run unless the memory is low, there is no guarantee that an object's Dispose() method will be invoked in an expedient fashion. The garbage collector may not be aware of unmanaged system resource pools that need to be freed up due to depletion. It is possible to run out of memory within a .NET environment due to unmanaged resource depletion.

Instead of explicitly closing objects that use managed resources, use the C# keyword 'using', which employs the IDisposable interface to perform a cleanup. The following two blocks of code achieve the same result:

The following code uses the finally keyword:

```
private void processBitmap(Stream incomingStream, Stream outgoingStream, int thumbnailSize)
{
```

```
Bitmap incomingBitmap = null;
Bitmap outgoingBitmap = null;
try {
incomingBitmap = (Bitmap)System.Drawing.Image.FromStream(incomingStream);
bool validBitmap = validateBitmap(incomingBitmap);
if (!validBitmap)
throw new ValidationException(incomingBitmap);
outgoingBitmap = new Bitmap(incomingBitmap, new Size(thumbnailSize, thumbnailSize));
outgoingBitmap.Save(outgoingStream, ImageFormat.Bmp);
} finally {
if (incomingBitmap != null) {
incomingBitmap.Dispose();
}
if (outgoingBitmap != null) {
outgoingBitmap.Dispose();
}
}
}
```

The following code uses the using keyword:

```
private void processBitmap(Stream incomingStream, Stream outgoingStream, int thumbnailSize)
{
using (Bitmap incomingBitmap = (Bitmap)System.Drawing.Image.FromStream(incomingStream))
{
bool validBitmap = validateBitmap(incomingBitmap);
if (!validBitmap)
throw new ValidationException(incomingBitmap);
using (Bitmap outgoingBitmap = new Bitmap(incomingBitmap, new Size(thumbnailSize, thumbnailSize)))
{
outgoingBitmap.Save(outgoingStream, ImageFormat.Bmp);
}
}
}
```

## RandomImage.cs, line 79 (Unreleased Resource: Unmanaged Object)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Code Quality | | |

**Abstract:** The function GenerateImage() in RandomImage.cs fails to properly dispose of unmanaged system resources allocated by Font() on line 79.

**Sink:** RandomImage.cs:79 font = new Font(...)

```
77              {
78                  fontSize--;
79                  font = new Font(FontFamily.GenericSansSerif, fontSize, FontStyle.Bold);
80                  size = g.MeasureString(this.text, font);
81              } while (size.Width > rect.Width);
```

## RandomImage.cs, line 70 (Unreleased Resource: Unmanaged Object)

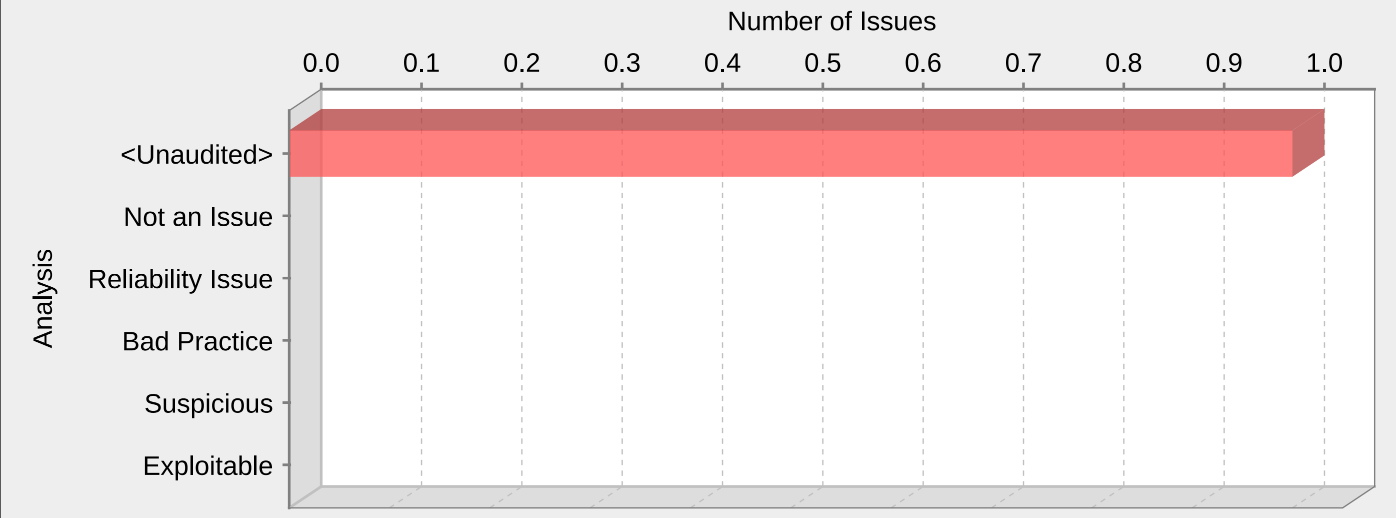
| | | | |
|---|---|---|---|
| **Fortify Priority:** | High | **Folder** | High |
| **Kingdom:** | Code Quality | | |

**Abstract:** The function GenerateImage() in RandomImage.cs fails to properly dispose of unmanaged system resources allocated by HatchBrush() on line 70.

**Sink:** RandomImage.cs:70 hatchBrush = new HatchBrush(...)

```
68              Rectangle rect = new Rectangle(0, 0, this.width, this.height);
69              HatchBrush hatchBrush = new HatchBrush(HatchStyle.SmallConfetti,
                                        Color.LightGray, Color.White);
                                g.FillRectangle(hatchBrush, rect);
                                SizeF size;
```

## Category: ASP.NET Misconfiguration: Debug Information (1 Issues)



**Abstract:**

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

**Explanation:**

ASP .NET applications can be configured to produce debug binaries. These binaries give detailed debugging messages and should not be used in production environments. 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. 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. 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.

**Recommendations:**

Always compile production binaries without debug enabled. This can be accomplished by setting the debug attribute to false on the <compilation> tag in your application's configuration file, as follows:

```
<configuration>
<compilation debug="false">
...
</compilation>
...
</configuration>
```

Setting the debug attribute to false is necessary for creating a secure application. However, it is important that your application does not leak important system information in other ways. Ensure that your code does not unnecessarily expose system information that could be useful to an attacker.

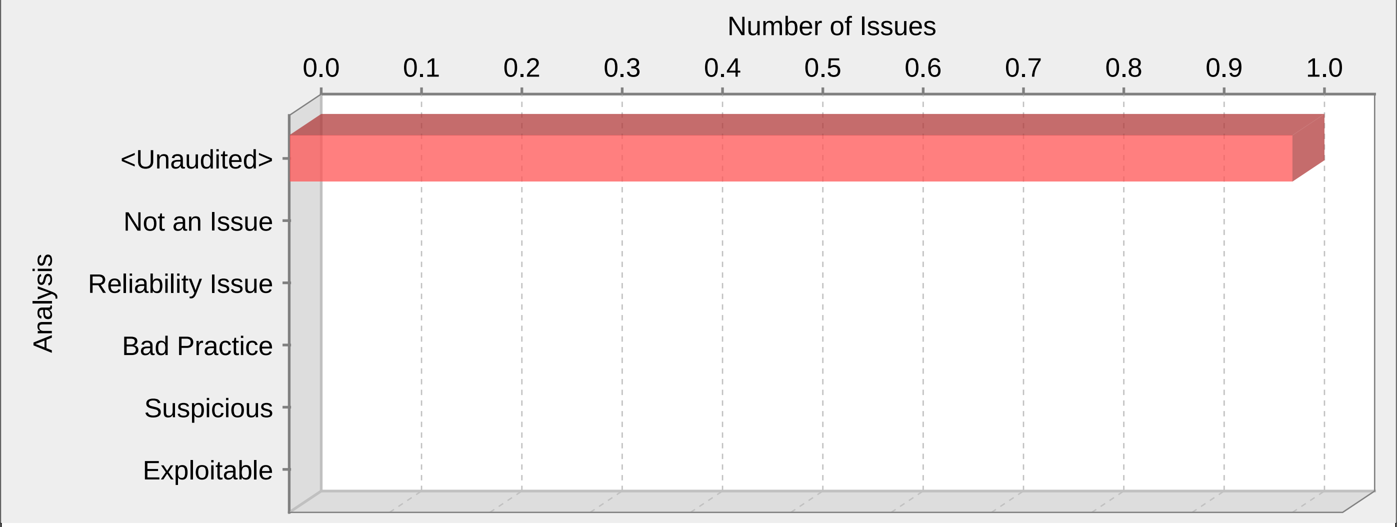
### Web.config, line 24 (ASP.NET Misconfiguration: Debug Information)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Medium | **Folder** | Medium |
| **Kingdom:** | Environment | | |

| | |
|---|---|
| **Abstract:** | Debugging messages help attackers learn about the system and plan a form of attack. |
| **Sink:** | Web.config:24 null() |

```
  </connectionStrings>
  <system.web>
     <compilation debug="true" targetFramework="4.0"></compilation>
    <httpCookies httpOnlyCookies="true"/>
    <httpRuntime targetFramework="4.0"/>
```

## Category: Resource Injection (1 Issues)



### Abstract:

Allowing user input to control resource identifiers could enable an attacker to access or modify otherwise protected system resources.

### Explanation:

A resource injection issue occurs when the following two conditions are met:

1. An attacker can specify the identifier used to access a system resource.

For example, an attacker might be able to specify a port number to be used to connect to a network resource.

2. By specifying the resource, the attacker gains a capability that would not otherwise be permitted.

For example, the program may give the attacker the ability to transmit sensitive information to a third-party server.

Note: Resource injection that involves resources stored on the filesystem goes by the name path manipulation and is reported in separate category. See the path manipulation description for further details of this vulnerability.

Example: The following code uses a port number read from an HTTP request to create a socket.

```
int rPort = Int32.Parse(Request.Item("rPort"));
...
IPEndPoint endpoint = new IPEndPoint(address,rPort);
socket = new Socket(endpoint.AddressFamily,
SocketType.Stream, ProtocolType.Tcp);
socket.Connect(endpoint);
...
```

The kind of resource affected by user input indicates the kind of content that may be dangerous.  For example, data containing special characters like period, slash, and backslash are risky when used in methods that interact with the file system. Similarly, data that contains URLs and URIs is risky for functions that create remote connections.

### Recommendations:

The best way to prevent resource injection is with a level of indirection: create a list of legitimate resource names that a user is allowed to specify, and only allow the user to select from the list. With this approach the input provided by the user is never used directly to specify the resource name.

In some situations this approach is impractical because the set of legitimate resource names is too large or too hard to keep track of. Programmers often resort to blacklisting in these situations. Blacklisting selectively rejects or escapes potentially dangerous characters before using the input. However, any such list of unsafe characters is likely to be incomplete and will almost certainly become out of date. A better approach is to create a white list of characters that are allowed to appear in the resource name and accept input composed exclusively of characters in the approved set.

### Tips:

1. If the program is performing input validation, satisfy yourself that the validation is correct, and use the Custom Rules Editor to create a cleanse rule for the validation routine.

2. It is notoriously difficult to correctly implement a blacklist. If the validation logic relies on blacklisting, be skeptical. Consider different types of input encoding and different sets of metacharacters that might have special meaning when interpreted by different operating systems, databases, or other resources. Determine whether or not the blacklist can be updated easily, correctly, and completely if these requirements ever change.

3. A number of modern web frameworks provide mechanisms for performing validation of user input. ASP.NET Request Validation and WCF are among them. To highlight the unvalidated sources of input, the HP Fortify Secure Coding Rulepacks dynamically re-prioritize the issues reported by HP Fortify Static Code Analyzer by lowering their probability of exploit and providing pointers to the supporting evidence whenever the framework validation mechanism is in use. In case of ASP.NET Request Validation, we also provide evidence for when validation is explicitly disabled. We refer to this feature as Context-Sensitive Ranking. To further assist the HP Fortify user with the auditing process, the HP Fortify Software Security Research Group makes available the Data Validation project template that groups the issues into folders based on the validation mechanism applied to their source of input.

## Sms.aspx.cs, line 193 (Resource Injection)

| | | | |
|---|---|---|---|
| **Fortify Priority:** | Low | **Folder** | Low |
| **Kingdom:** | Input Validation and Representation | | |

**Abstract:** Attackers can control the resource identifier argument to Create() at Sms.aspx.cs line 193, which could enable them to access or modify otherwise protected system resources.

**Source:** Sms.aspx.cs:192 System.Web.UI.WebControls.TextBox.get_Text()

```
190            Check_SSL_Certificate();
191            //ServicePointManager.ServerCertificateValidationCallback = new
System.Net.Security.RemoteCertificateValidationCallback(AcceptAllCertifications);
192            string strURL = "https://smsgw.sms.gov.in/failsafe/HttpLink?username=" +
UName + "&pin=" + Pwd + "&message=" + TxtMessage.Text + "&mnumber=" + TxtMobile.Text +
"&signature=" + strms + "";
193            HttpWebRequest myReq = (HttpWebRequest)WebRequest.Create(strURL);
194            HttpWebResponse myResp = (HttpWebResponse)myReq.GetResponse();
```

**Sink:** Sms.aspx.cs:193 System.Net.WebRequest.Create()

```
191            //ServicePointManager.ServerCertificateValidationCallback = new
System.Net.Security.RemoteCertificateValidationCallback(AcceptAllCertifications);
192            string strURL = "https://smsgw.sms.gov.in/failsafe/HttpLink?username=" +
UName + "&pin=" + Pwd + "&message=" + TxtMessage.Text + "&mnumber=" + TxtMobile.Text +
"&signature=" + strms + "";
193            HttpWebRequest myReq = (HttpWebRequest)WebRequest.Create(strURL);
194            HttpWebResponse myResp = (HttpWebResponse)myReq.GetResponse();
195            StreamReader myReader = new StreamReader(myResp.GetResponseStream());
```

## Issue Count by Category

| Issues by Category | |
|---|---|
| Access Control: Database | 148 |
| SQL Injection | 23 |
| Poor Error Handling: Overly Broad Catch | 22 |
| Cross-Site Scripting: Persistent | 10 |
| Unreleased Resource: Database | 3 |
| Unreleased Resource: Unmanaged Object | 2 |
| ASP.NET Misconfiguration: Debug Information | 1 |
| Resource Injection | 1 |

## Issue Breakdown by Analysis

### Issues by Analysis