Web security for developers #1

Damien Guard BSc, MBCS
damien@envytech.co.uk

Guernsey Software Developers Forum
http://www.developers.org.gg

Aspects of web security

- Infrastructure - policy, firewall
- Secure channel - encryption, cards
- Software environment - OS, platform
- The human factor
- Your application

Web application concerns

- Injection (SQL, HTML, XPath...)
- Manipulation of input, state, cookies
- Session hijacking
- Denial of service via slow public pages
- Encryption, hashing and key management

SQL injection

Described

- Browser input is sent to the SQL server
- Poor encoding allows user to modify SQL
- Visitors can manipulate or create statements
- Resulting in security bypass, data modification...

SQL injection

Demonstrated

- Typical login page
- Try entering a ‘ in the username field
- System error occurs
- Manipulate the SQL via username field
- Now have a match

SQL injection

Exploited code

```
command.CommandText =
"SELECT * FROM [Users] WHERE " +
"Username='" + UsernameTextBox.Text + "'" +
" AND " +
"Password='" + PasswordTextBox.Text + "'";
```

SQL injection

Exploit explained

Developer expected

```
SELECT * FROM [Users] WHERE
Username='Bill' AND Password='Ted'
```

Actual statement

```
SELECT * FROM [Users] WHERE
Username='' OR 1=1 --Bill' AND
Password='Ted'
```

SQL injection

Do not...

- Rely on JavaScript to prevent it happening
- Refuse to accept single quotes
- Examine input for SQL statements
- Reinvent the encoding mechanism

SQL injection

Parameterised query solution

```
command.CommandText =
  "SELECT * FROM [Users] WHERE " +
  "Username=@Username AND Password=@Password";
command.Parameters.AddWithValue
  ("@Username", UsernameTextBox.Text);
command.Parameters.AddWithValue
  ("@Password", PasswordTextBox.Text);
```

HTML injection

Description

- Can occur where data is used to build page
- Poor encoding allows HTML code in data
- Page is modified by specially crafted data
- Data can inject script into the page
- Could cause forms to submit elsewhere

HTML injection

Demonstration

Test vulnerability with

```
?Error=<script>alert('Hello');</script>
```

Submit form elsewhere

```
?Error=<script>document.getElementById
('form1').action='Cap';</script>
```

HTML injection

Exploited code

```
ErrorLabel.Text = Request["Error"];
```

HTML injection

Possible solutions

No HTML allowed

```
ErrorLabel.Text =
  HttpUtility.HtmlEncode(Request["Error"]);
```

Minimal HTML allowed

```
ErrorLabel.Text =
  MyClass.SafeHtmlEncode(Request["Error"]);
```

Conclusion

- Application security isn’t hard
- Know and utilise provided framework
- Avoid strings where specific classes exist
- Never blindly trust any user's input

More information

- Web Application Security Consortium
  http://www.webappsec.org
- Microsoft ASP.NET Security Practices
  http://msdn2.microsoft.com/en-us/library/ms998372.aspx