SQL Injection and XSS

How they work and how to stop them.

Rob Kraft, Rob@RobKraft.org

What r hackers looking 4?

- Non-specific attacks
  - Identifying vulnerable servers
  - Turning the computer into a zombie (botnet)
  - Hard disk space
- Specific attacks
  - Info to exploit: Financial Data
  - Doing damage

SQL Injection Prevention Overview

- Programmers
  - Don’t use Dynamic SQL
  - Validate Inputs (on the server side)
  - Don’t provide detailed errors to users
- DBAs
  - Restrict database permissions
  - Review Logs
  - Disable unneeded features
  - Encrypt data, Strong Procedures

SQL Injection Prevention Overview

- Network Administrators
  - Configure firewalls to watch for scans
- Windows Administrators
  - Configure IIS to use URL Scanning
  - Require strong passwords with expirations
  - Use different accounts for SQL Services
- Auditing
  - Review configurations and logs

Don’t use dynamic SQL

- Do NOT build SQL Dynamically like this:

```
sql = "select title from titles where id =" + id;
```

Do use Parameters

- DO use parameters (not just on strings)

```
sql = “select title from titles where id = @id";
myCmd.Parameters.AddWithValue("@id",txt1.Text);
```

- Or – Write your own formatter
  - sql = sql.Replace(“'”, “''”);
- Tip: It is usually easier to review code that uses parameters for proper usage.
- Tip: Review code.

Comparison

strCity = “Lee's Summit”;

sql = “… where city = '“ + strCity + “' ”;

…where city = 'Lee's Summit' - Error

sql = “… where city = @city”;

…Parameters.Add(“@city”,strCity);

…where city = 'Lee''s Summit' – No Error

Dangerous Stored Procedure

```
CREATE PROCEDURE dbo.LoginAccount
( @UserName nvarchar(20), @Password nvarchar(20))
AS
EXECUTE ('SELECT USERNAME, USERID FROM
    USERS WHERE USERNAME = ''' + @UserName + '''
    AND PASSWORD = ''' + @Password + '''')
RETURN
```

- Don't build dynamic SQL in stored procedures.

Preventing SQL Injection

- IIS Sites: Set the mode of <customErrors> in web.config to On or RemoteOnly to minimize information returned to the client's browser that a hacker could use to learn about your databases.

Validate Input

- Validate data input
  - Use javascript to provide a friendly and responsive UI to users, but don't rely on it for security
  - Use ASP.Net Validators, but don't rely on them
  - Check input lengths (and log rejections)
  - Use a Regex Whitelist (and log rejections)

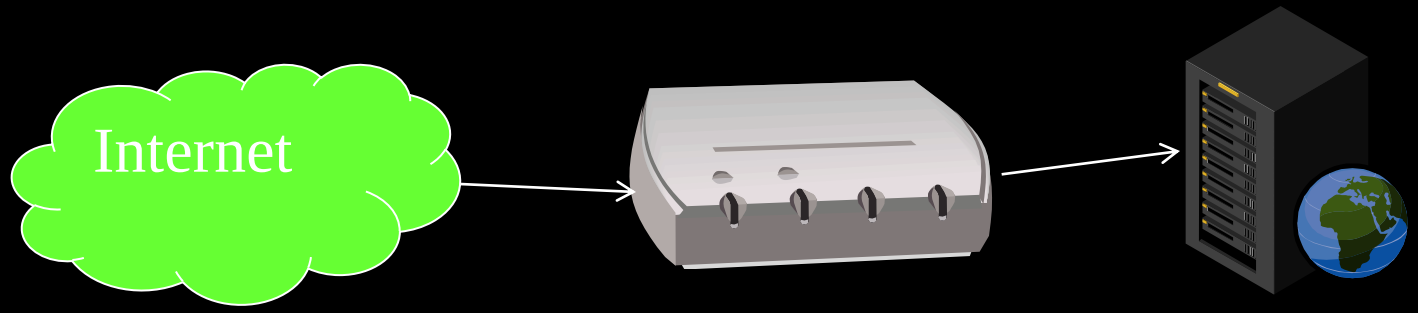
Regex Example

```
Regex allowRegex = new Regex(@"^[a-zA-Z\s\']*$");
// But we disallow common SQL functions
Regex disallowRegex = new Regex("(union|select|drop|delete)");
if ((!allowRegex.IsMatch(textBoxLastName.Text)) ||
    (disallowRegex.IsMatch(textBoxLastName.Text)))
{
        labelErrorMessage.Text = "Invalid name.";
        return;
}
```

Restrict Database Permissions

- NEVER use ‘sa’ for an application
- NEVER use an Admin account for an application
- Use a database logon, or logons with minimal database permissions required by the application.
- Use stored procedures – don’t give the dbms logon account permission to the underlying tables
- Turn off xp_cmdshell
- Use SQL 2008/2005 instead of SQL 2000

Don't depend on front-end devices



- Do NOT rely on signature based detection
  - You can block --; but what about %2d%2d
  - You can block union; but what about 'un' + 'ion'

Preventing SQL Injection

- Review logs:
  - Failed SQL Logins
  - URL Requests

Other technologies

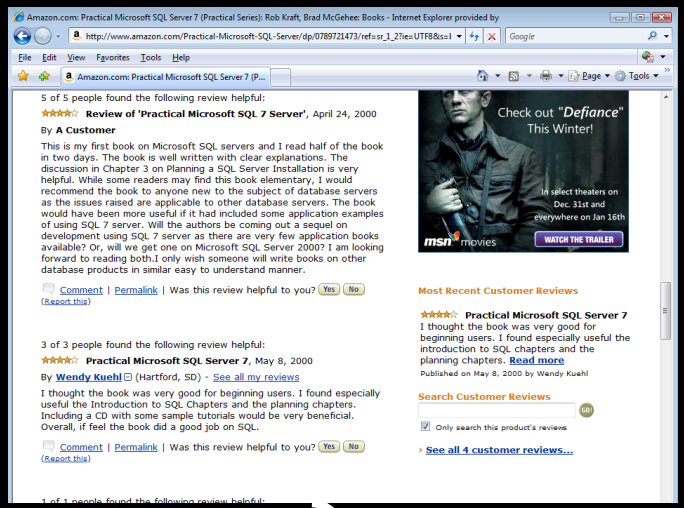
- Web Services
  - Web services are also at risk for SQL Injection
- LINQ to SQL is safe by default
  - Dim q = From c in db.Customers

    Where c.city = “Lee’s Summit”
- Nhibernate is safe by default
- MS Entity Framework is safe by default

Topic 2: Cross Site Scripting (XSS)

- XSS is also called 2nd Order SQL Injection
- Data is injected into a database, and is used later to attack.
  - HTML or Javascript stored in the database that executes when rendered on a web page.

A Script embedded in a web site sends data to the hacker's site. In this example, a hacker may have embedded a script in a book review on Amazon.



When any user navigates to the page with the book review, information from their cookie is sent to the hacker's web site.

How to stop XSS

- Check that ASP.Net request validation is enabled
  - ValidateRequest=true on ASP.Net web pages
- Check input lengths or use a Regex Whitelist
- HttpUtility.HTMLEncode and URLEncode (blacklist)
  - Better: https://www.owasp.org/index.php/ESAPI
  - Better: Microsoft Anti-XSS Libraries
- Use the innerText Property Instead of innerHTML
  - Better: jquery: $('tagname').html(value)

How to stop XSS - Dilemma

- To prevent XSS we must encode output.
- To render output correctly we must not double encode it.
- Therefore we need to know what encodings may be in the raw data:
  - HTML, javascript, CSS, XML
- We need to know where to perform encoding (server side or client side)
- I haven't found a great solution for client-side yet.

Microsoft Resources

- A Microsoft tool for scanning for injection problems:
  - www.pnpguidance.net/Post/MicrosoftSourceCodeAnalyzerSQLInjectionToolFindSQLInjectionProblems.aspx
- Microsoft guidance in SQL 2008:
  - msdn2.microsoft.com/en-us/library/ms161953.aspx
- How To: Protect From SQL Injection in ASP.NET
  - msdn.microsoft.com/en-us/library/ms998271.aspx
- Microsoft Guidance on XSS
  - msdn.microsoft.com/en-us/library/ms998274.aspx
- Microsoft Security Development Lifecycle (SDL)--
- Microsoft Anti-Scripting library 4.0:
  - http://www.microsoft.com/downloads/en/details.aspx?FamilyID=f4cd231b-7e06-445b-bec7-343e5884e651

Good resources

- OWASP
  - https://www.owasp.org/index.php/Template:Cheatsheet_Navigation
- Examples of using Regex against injection
  - www.developerfusion.com/article/5855/beyond-stored-procedures-defenseindepth-against-sql-injection
- Good examples of SQL Injection
  - en.wikipedia.org/wiki/SQL_injection
  - www.unixwiz.net/techtips/sql-injection.html
- CSRF:
  - en.wikipedia.org/wiki/Cross-site_request_forgery
  - www.freedom-to-tinker.com/sites/default/files/csrf.pdf

Tools to help

- Absinthe: blind injection attack and analysis tool
  - www.0x90.org/releases/absinthe
- FXCop
  - msdn.microsoft.com/en-us/library/bb429476(VS.80).aspx
- Microsoft Security Runtime Engine
  - blogs.msdn.com/cisg/archive/2008/10/24/a-sneak-peak-at-the-security-runtime-engine.aspx

Summation

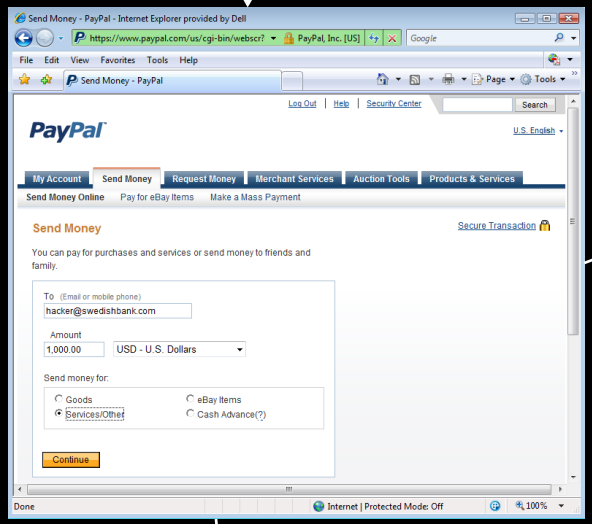
- BAD:
- "select Title from titles where title like '%" + input + "%'"
- GOOD:
- myCmd = new SqlDataAdapter("select Title from titles where title like @query", m_SQLConn);
- parm = myCmd.SelectCommand.Parameters.Add("@query", SqlDbType.VarChar, 10);
- parm.Value = "%" + input + "%";

- All the links and slides are at:

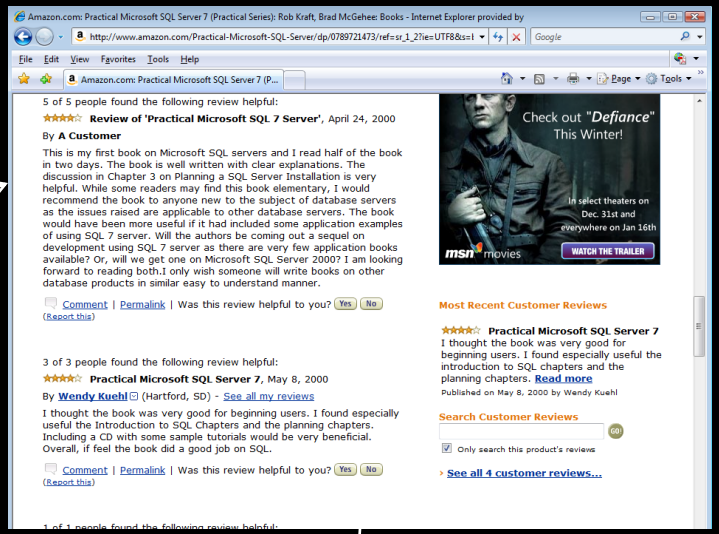
  www.KraftSoftware.com

Cross Site Request Forgeries (CSRF)

1. User goes to site



3. User navigates to injected site



2. Page goes to Paypal Server

PayPal Server

4. Injected site performs action on "still authenticated" site.

<img src="http://paypal.com/withdraw?account=rkraft&amount=1000000&for=hacker">

How to stop CSRF

- Only use POST for modifications, not GET
- Require all requests to include pseudo-random value (session independent) – include on POST form and cookie – must match
- User's should Log Out of sites