SQL Injection and XSS

How they work and how to stop them.

Rob Kraft, Rob@RobKraft.org

Overview

1. SQL Injection
2. Cross Site Scripting (XSS)
3. Cross Site Request Forgeries (CSRF)

Topic 1: SQL Injection

- What is SQL Injection?
- What can hackers do with SQL Injection?
- A free SQL Injection hacking tool
- Preventing SQL Injection

DEMO - SQL Injection Examples

- '
- ' or 1=1 –
- ' union select password, 1 from logins where logonid='admin' –
- ';update logins set username='hacked' where logonid='rkraft' --

What can hackers do with SQL Injection?

- Retrieve data
- Modify data
- Destroy data
- Xp_cmdshell – manipulate the servers
- What can be done depends on the permissions of the account used to access the database

DEMO - SQL Injection Examples

- http://localhost:51117/WebSite1/URL.aspx?title=comp
- http://localhost:51117/WebSite1/URL.aspx?title=comp' or 1=1—
- http://localhost:51117/WebSite1/URL.aspx?title=xyxy' union select name, "," from sysobjects where type = 'u' --

The tools hackers use

- Hackers use programs to do their hacking for them.
- They don’t require data to be displayed in grids or in specific fields on specific forms.
- Example tool: Absinthe

What r hackers looking 4?

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

SQL Injection Prevention Overview

- Don’t use Dynamic SQL
- Restrict database permissions
- Validate Inputs
- Don’t provide detailed errors to users
- Review Logs

Don’t use dynamic SQL

- Do NOT build SQL Dynamically:

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

Do use Parameters

- DO use parameters

  sql = “select title from titles where id = @id";

  myCmd.Parameters.AddWithValue("@id",txt1.Text);

- Or – Write your own formatter
  - sql = sql.Replace(“'”, “''”);
  - It is usually easier to review code that uses parameters for proper usage
  - Don’t forget to format numerics like above

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.

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 2005 instead of SQL 2000

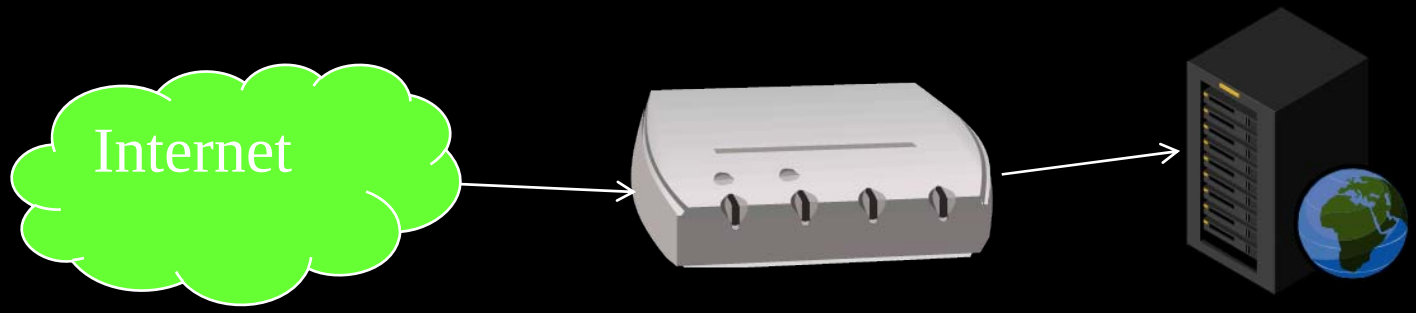
Validate Input

- Validate data input
  - Use javascript, but don’t rely on it
  - 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;
}
```

Don't rely 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'

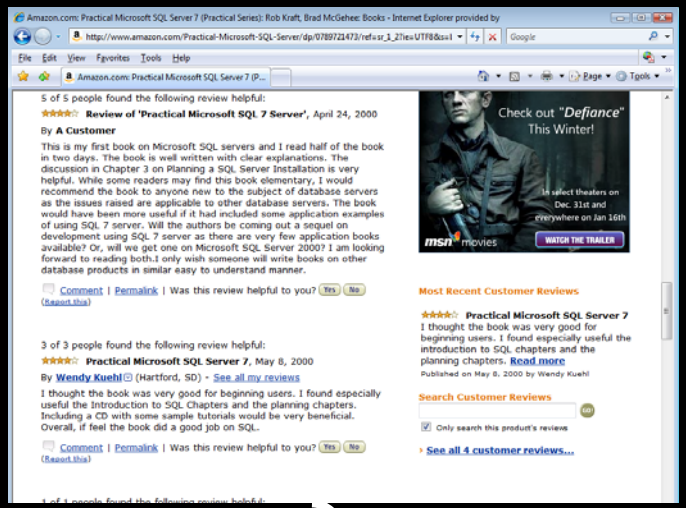
Preventing SQL Injection

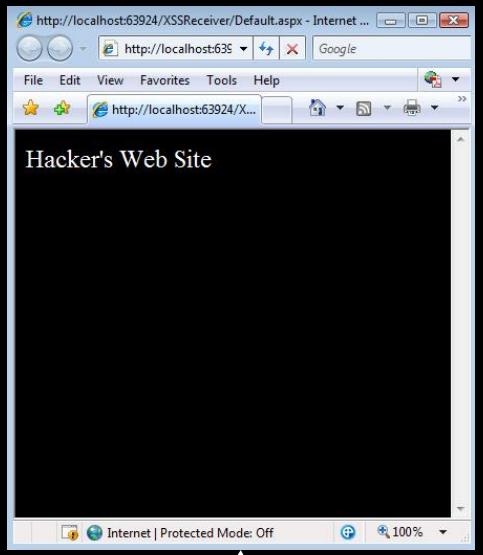
- Set the mode of <customErrors> to On or RemoteOnly
- Review logs:
  - Failed SQL Logins
  - URL Requests
- End of Topic 1 – SQL Injection

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.





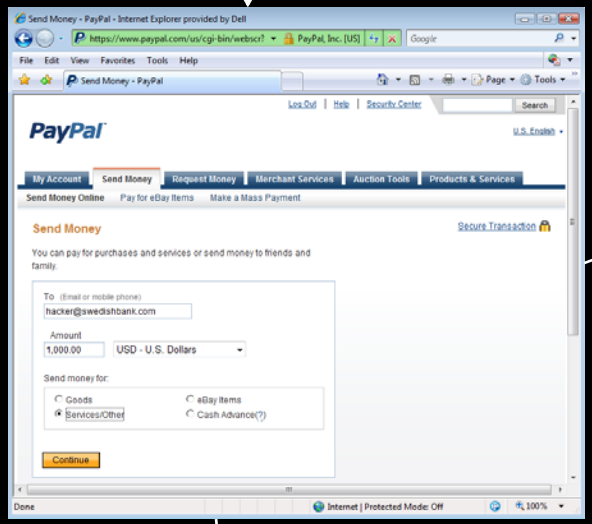
When any user navigates to the page with the book review, information from their cookie is sent to the hacker's web site.

How to stop XSS

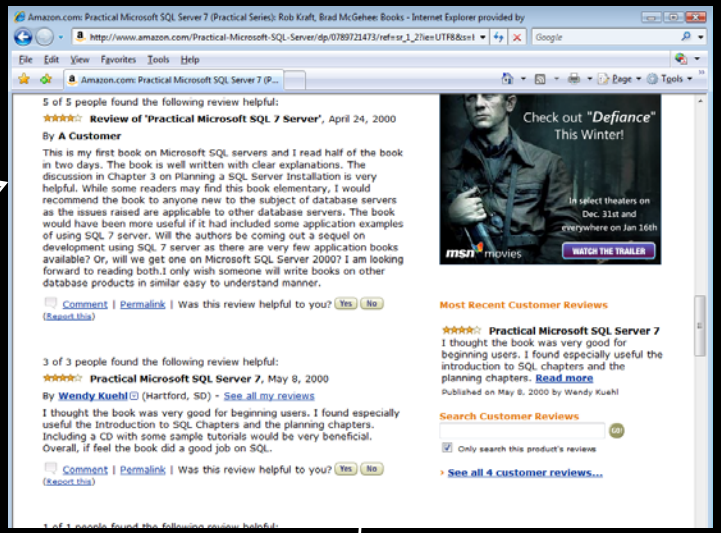
- Check that ASP.Net request validation is enabled
  - ValidateRequest=true on ASP.Net web pages
- Check input lengths (and log rejections)
- Use a Regex Whitelist (and log rejections)
- HttpUtility.HTMLEncode and URLEncode (blacklist)
- Use the innerText Property Instead of innerHTML
- Microsoft Anti-Scripting library (whitelist)

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

Other technologies

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

    Where c.city = “Lee’s Summit”
  - Safe – using ADO.Net Parameters
- Silverlight 2.0
  - Neither safer nor less safe from SQL Injection

Other Presentations

- *Advanced Topics on SQL Injection Protection: www.owasp.org/images/7/7d/**Advanced_Topics_on_SQL_Injection_Protection.ppt***
- Good video showing SQL Injection
  - jumpstarttv.com/Media.aspx?vid=76

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 1.5: (3.0 coming)
  - www.microsoft.com/downloads/details.aspx?familyid=efb9c819-53ff-4f82-bfaf-e11625130c25&displaylang=en

Good articles

- 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

- All the links and slideshow are at:

www.KraftSoftware.com

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 + "%";