

ASC301
Hacked! Understanding and Preventing Web Site Attacks

Paul Litwin
 Deep Training &
 Fred Hutchinson Cancer Research Center


pauLL@deeptraining.com

Paul Litwin

- **Developer**
 - ? Focus: ASP.NET, ASP, VB, C#, SQL Server, ...
 - ? MCSD
 - ? Microsoft MVP
 - ? Lead Programmer with Fred Hutchinson Cancer Research Center
- **Co-Founder and Senior Trainer**
 - ? Deep Training
 - www.deeptraining.com
- **Conference Chair/Speaker**
 - ? Chair, Microsoft ASP.NET Connections
 - ? Member INETA Speakers Bureau
- **Author**
 - ? Author/co-author of a dozen books, including...
 - ASP.NET for Developers
 - Access Cookbook, 2nd edition
 - Access 2002 Desktop/Enterprise Dev Handbook

Attribution

- SQL Injection portion of this presentation is based on article
 I wrote for MSDN Magazine (September, 2004 issue)



- Some slides/examples based on Microsoft DevDays presentation

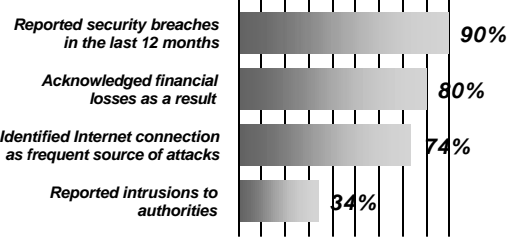
Agenda

- Web Site Security
- SQL Injection Attacks
- Cross-Site Scripting Attacks
- Hidden-Field Tampering Attacks
- Managing Secrets
- Layering Defenses

Why Security?

2002 Computer Crime and Security Survey

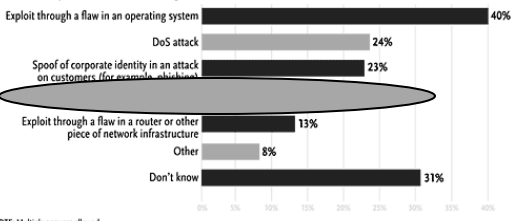
Percentages of companies who participated in the survey



<http://www.gocsi.com/press/20020407.html>

InfoWorld Security Survey (July 2004)

Which of the following security threats has your company been subjected to in the past 12 months?

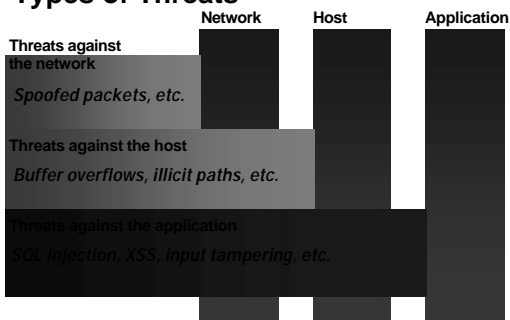


NOTE: Multiple answers allowed. Based on poll of 606 IT professionals in June 2004. Article by Paul F. Roberts in July 26 issue of InfoWorld

Call to Action

- Secure software requires knowledgeable and dedicated IT personnel
 - ? Software isn't secure if the network is not
 - ? Administration is the bedrock of security
- Secure software also requires knowledgeable and dedicated developers
 - ? Proper administration is meaningless if the code you write isn't secure
 - ? Most developers today don't know they're writing insecure code

Types of Threats



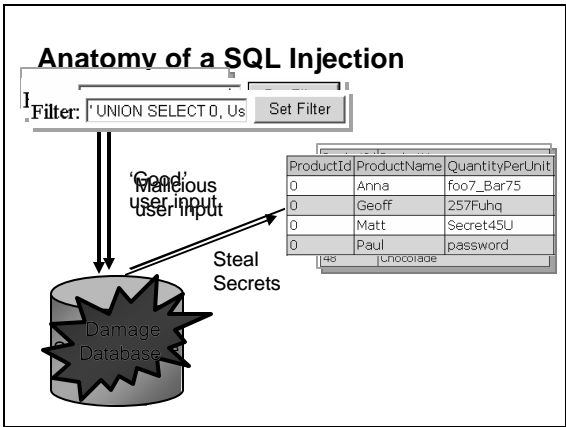
Threats Against the Application

Threat	Examples
SQL injection	Including a DROP TABLE command in text typed into an input field
Cross-site scripting	Using malicious client-side script to steal cookies
Hidden-field tampering	Maliciously changing the value of a hidden field
Eavesdropping	Using a packet sniffer to steal passwords and cookies from traffic on unencrypted connections
Session hijacking	Using a stolen session ID cookie to access someone else's session state
Identity spoofing	Using a stolen forms authentication cookie to pose as another user
Information disclosure	Allowing client to see a stack trace when an unhandled exception occurs

SQL Injection Attacks

- Basic Idea
 - ? Application asks user to enter a value into textbox
 - ? User enters malformed data into textbox
 - ? Malformed data causes unintended effects, including
 - Authenticating hacker
 - Modifying database
 - Creating logins
 - Stealing database secrets

Anatomy of a SQL Injection



Demos of SQL Injection

- SQLInjection Project
 - ? BadLogin.aspx – susceptible forms login page
 - ? BadProductList.aspx – susceptible DataGrid page
- Additional Demos
 - ? SQLInjectWinForm
 - ? SQLInjection_MySql

Preventing SQL Injection

- Validate Input
 - ? Use regular expressions
 - ? Specify what's valid and reject everything else
 - Lesser Alternative: Reject bad characters
- Use Parameterized Stored Procedures
 - ? Or at least parameterized SQL
- Execute SQL Using Least Privileged Account
 - ? Don't use SysAdmin (sa) account!

Parameterized SQL

```
string strQry = "SELECT Count(*) FROM Users WHERE UserName= " +
               "@username AND Password=@password";
SqlCommand cmd = new SqlCommand(strQry, cnx);
cmd.CommandType= CommandType.Text;
prm = new SqlParameter("@username", SqlDbType.VarChar,50);
prm.Direction=ParameterDirection.Input;
prm.Value = txtUser.Text;
cmd.Parameters.Add(prm);
```

GoodLogin.aspx

```
SqlCommand cmd = new SqlCommand("procVerifyUser", cnx);
cmd.CommandType= CommandType.StoredProcedure;
prm = new SqlParameter("@username", SqlDbType.VarChar,50);
prm.Direction=ParameterDirection.Input;
prm.Value = txtUser.Text;
cmd.Parameters.Add(prm);
```

BetterLogin.aspx

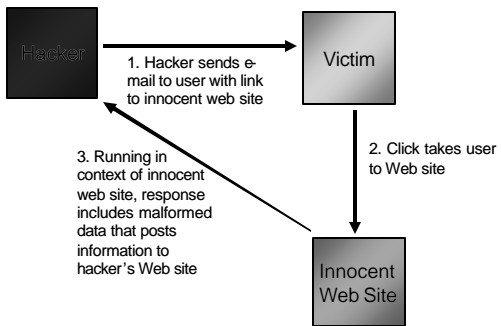
Preventing SQL Injection

- SQLInjection project
 - ? Validating input – GoodLogin.aspx
 - ? Parameterized SQL – GoodLogin.aspx
 - ? Stored Procs – BetterLogin.aspx
 - ? Lower-privileged accnts – GoodLogin.aspx & BetterLogin.aspx

Cross-Site Scripting (XSS) Attacks

- When Good Output Turns Bad
- Hacker Hijacks Site For Evil Purposes

Anatomy of an XSS Attack



XSS Demo

- XSS project
 - ? Home.aspx/Welcome.aspx – Susceptible pages
- XSSHacker project
 - ? LureMe.aspx – Link which sets up XSS hack
 - ? GrabCookie.aspx – Page that secretly steals cookie & logs to XSSHacker.mdb database

Preventing XSS

- Validate Input
 - ? Use regular expressions
 - ? Specify what's valid and reject everything else
 - Lesser Alternative: Reject bad characters
- Encode All Output
 - ? If using ASP or ASP.NET, you can use Server.HtmlEncode method
- ASP.NET 1.1
 - ? ValidateRequest
 - On by default
 - Can turn off w/ ValidateRequest page directive or in web.config
 - Not a panacea
 - You can add handler to global.asa to handle ValidateRequest errors

Encoding Output Welcome2.aspx

```
lblMsg.Text = "Welcome " & _
    Server.HtmlEncode(Request("txtName ")) & _
    " to our site!"
```

Preventing XSS

- XSS Project
 - ? Home1.aspx/Welcome1.aspx – use ValidateRequest attribute
 - ? Home2.aspx/Welcome2.aspx – use HtmlEncode method

Hidden-Field Tampering

- Never put data into hidden fields that could be of value to hackers if it was altered or stolen
 - ? Price
 - ? Credit card
 - ? etc.

Hidden-Field Tampering Demo

- HotCars project
 - ? Default.aspx – Susceptible to hidden-field tampering attack

Managing Secrets

- Storing Secrets
 - ? Connection strings need to be protected
 - Encrypt connection strings
 - ? Passwords should never be stored in databases in clear text
 - Hash or encrypt password
- Passing Secrets
 - ? Encrypt connection (SSL) when passing sensitive data
 - ? Minimize passage of sensitive data

Connection Strings

- Storing plaintext database connection strings in Web.config is risky
 - ? Vulnerable to file disclosure attacks
- Storing encrypted database connection strings increases security
- Encrypting connection strings
 - ? System.Security.Cryptography classes
- Key management issue
 - ? Where do you store the decryption key?

Data Protection API (DPAPI)

- Present in Windows 2000 and higher
- Provides strong encryption, automatic key generation, and secure key storage
 - ? Triple-DES encryption
 - ? PKCS #5 key generation
- Two possible key "stores"
 - ? User store – Per-user keys based on profiles
 - ? Machine store – Per-machine keys with optional entropy values
- DataProtect.cs
 - ? .NET wrapper of DPAPI
 - ? Requires "Allow Unsafe Code Blocks" project configuration property (available in C#, not VB)
 - ? See [Building Secure ASP.NET Applications](#) for more details (URL later in slides)

Decrypting Connection String SecureConnections.cs

```
// Grab encrypted connection string from web.config
string strEncryptedCnx = ConfigurationSettings.AppSettings[configKey];

// Decrypt the connection string
DataProtector dp = new DataProtector (
    DataProtector.Store.USE_MACHINE_STORE);
byte[] dataToDecrypt = Convert.FromBase64String(strEncryptedCnx);
strCnx = Encoding.ASCII.GetString(dp.Decrypt(dataToDecrypt,null));
```

Using Encrypted Connection Strings

- SQLInjection project
 - ? EncryptCnxString.aspx – creates encrypted connection string
 - Need to run this first and paste encrypted cnxstring into Web.config
 - ? BestLogin.aspx – uses encrypted connection string
 - ? SecureConnecton.cs – decrypts connection string
 - ? DataProtect.cs – uses DPAPI to encrypt/decrypt

Storing Login Passwords

- Don't store passwords in login databases
- Store password hashes for added security
- Salt hashes to impede dictionary attacks

Format	Comments
Plaintext passwords	Exposes entire application if database is compromised
Encrypted passwords	Better than plaintext, but still vulnerable if decryption key is compromised
1-way password hashes	Better than encrypted passwords, but still vulnerable to dictionary attacks
Salted password hashes	Less vulnerable to dictionary attacks

Creating Hashed Password with Salt SaltedHash.cs

```
static public string CreateSaltedPasswordHash (string password) {
    // Generate random salt string
    RNGCryptoServiceProvider csp = new RNGCryptoServiceProvider ();
    byte[] saltBytes = new byte[16];
    csp.GetNonZeroBytes (saltBytes);
    string saltString = Convert.ToBase64String (saltBytes);

    // Append the salt string to the password
    string saltedPassword = password + saltString;

    // Hash the salted password
    string hash = FormsAuthentication.HashPasswordForStoringInConfigFile
        (saltedPassword, "SHA1");

    // Append the salt to the hash
    string saltedHash = hash + saltString;
    return saltedHash;
}
```

Storing Hashed Passwords

- SQLInjection project
 - ? Users table – stores plain-text passwords; can be revealed by SQLInjection attacks
 - ? SecureUsers table – stores hashed passwords
 - ? SaltedHash.cs – class lib for hashing passwords
 - ? AddSecureUser.aspx – inserts hashed password into database
 - ? BestLogin.aspx – compares hashed password with value in database

Additional Measures

- Give hacker as little info as possible
 - ? <compilation debug="false" />
 - ? <trace enabled="false">
 - ? <customErrors mode="RemoteOnly" />
- Always execute with least privileges
 - ? Mitigate the damage

Layering Your Defenses

- Can't predict all possible attacks
- If one defense penetrated, then others will be there to stop attack

Defense Summary

- Validate all input
- Avoid dynamic SQL
- Encode all output
- Execute with least privileged account
- Store/pass secrets securely
- Minimize exposure
- Fail gracefully and privately

Conclusion

- Don't have to be a security expert to understand the issues of malicious input
- This stuff is not going away
- Layer your approach

Resources

- Building Secure ASP.NET Applications: Authentication, Authorization, and Secure Communication
 - ? <http://www.msdn.microsoft.com/library/en-us/dnnetsec/html/secnetipMSDN.asp>
- Improving Web Application Security: Threats and Countermeasures
 - ? <http://www.msdn.microsoft.com/library/en-us/dnnetsec/html/ThreatCounter.asp>
- Writing Secure Code, Second Edition
 - ? Michael Howard and David LeBlanc (Microsoft Press, 2003)
 - ? Read 2nd edition, esp. chapters 10-13

Thank You!

- Materials are on the conference CD or you can download them from www.deeptraining.com/litwin
- Please complete your evaluations
 - ? ASC301
Hacked! Understanding and Preventing Web Site Attacks
 - ? Paul Litwin
