ASP.NET Security

Ben Hickman
ben@sftsrc.com

Demo: A Broken Application

The STRIDE Threat Model

- Spoofing identity
- Tampering with data
- Repudiation
- Information disclosure
- Denial of service
- Elevation of privilege

Secure Programming Principles

- Don’t trust user input
  - Assume all user input is malicious

Handling User Input: Why Worry?

- User data attacks:
  - Reveal implementation details
  - Create malicious data
  - Execute malicious script
  - Access restricted resources

Solution: Validate User Data

- Validate accepted range
  - Reject everything else
- Validate on the server

ASP.NET Validation Controls

- ASP.NET validation controls
  - RequiredFieldValidator
  - CompareValidator
  - RangeValidator
  - RegularExpressionValidator
  - CustomValidator
- Client validation is supported

```
<asp:TextBox id="pwd" runat="server"/>
<asp:RequiredFieldValidator
  ControlToValidate="pwd"
  ErrorMessage="Password required."
  EnableClientScript="true"
  id="pwdRequired"
  runat="server"
/>
```

Demo: Validation Controls

Handling User Input: Preventing SQL Injection

- SQL script injection
  - Use parameterized queries or stored procedures
  - Use ADO.NET Parameters collection

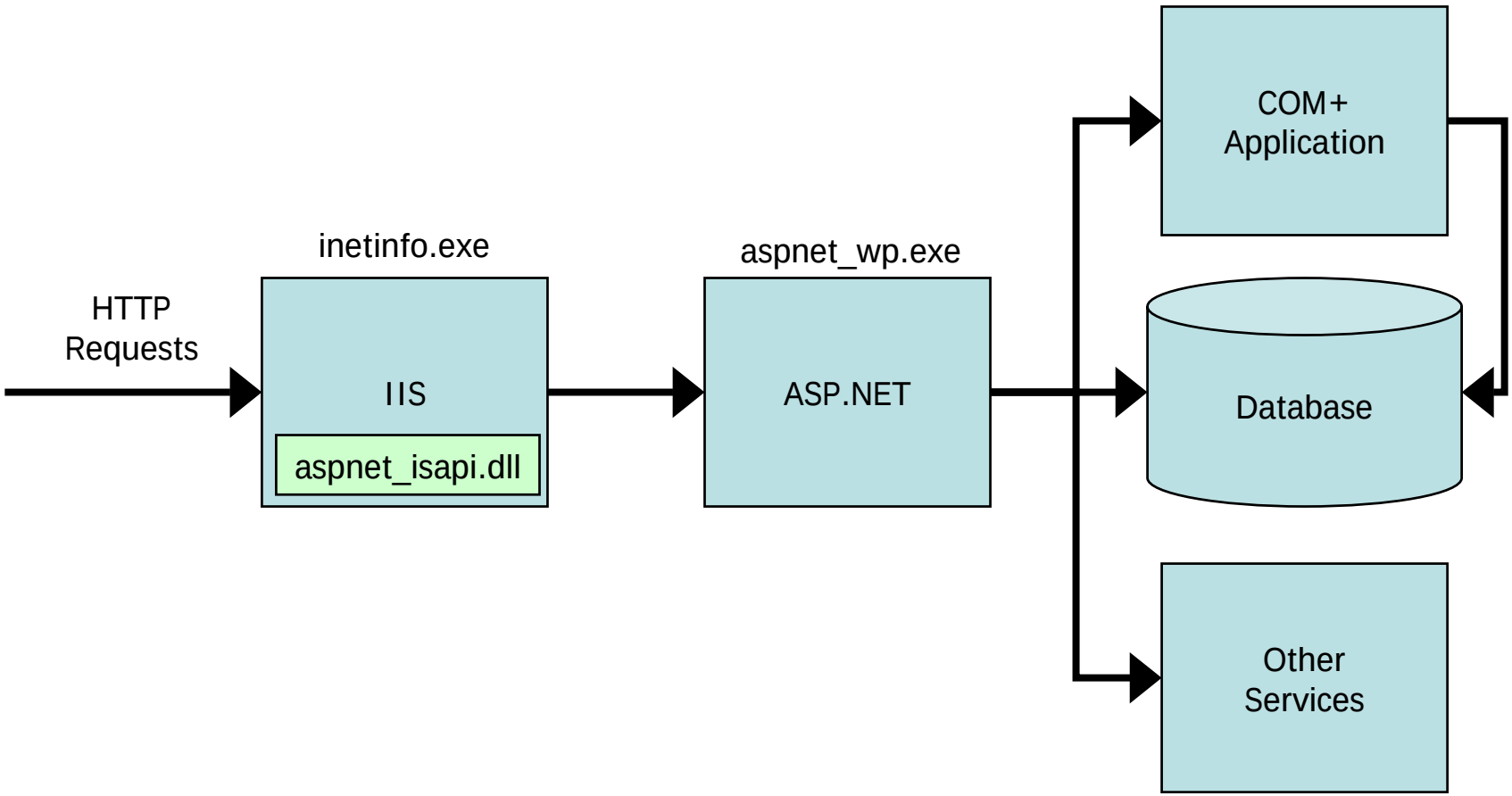
```
string sql = “SELECT COUNT(EmailName) FROM Users WHERE “ +
             “EmailName=@Username AND Password=@Password”;

SqlCommand cmd = new SqlCommand(sql, connection);
cmd.Parameters.Add(“@Username”, txtUsername);
cmd.Parameters.Add(“@Password”, txtPassword);

connection.Open();
int count = (int)command.ExecuteScalar();
connection.Close();
```

Demo: Parameterized SQL

Secure Programming Principles

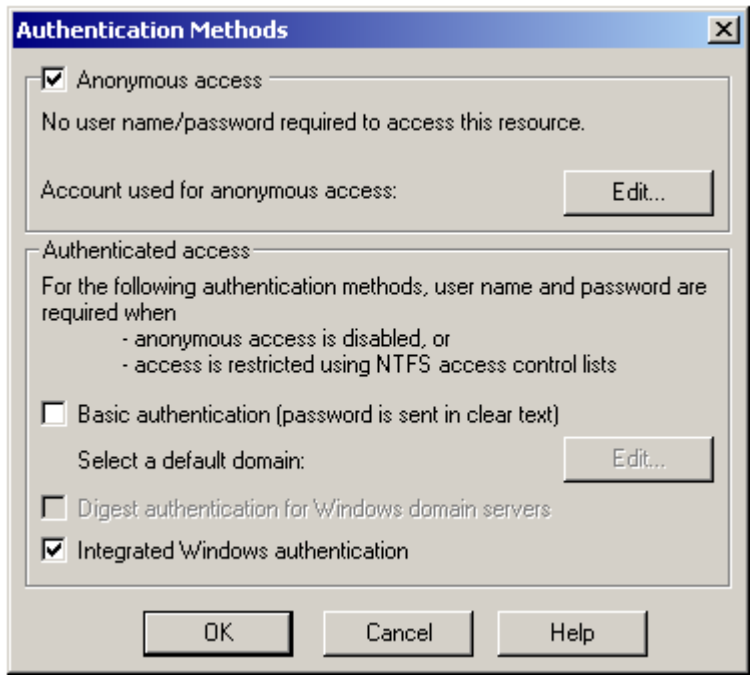
- Run with Least Privilege
  - All code must run with the minimum privileges required and no more.

ASP.NET Application Architecture



Authentication: IIS

- IIS can perform authentication
- Uses Windows accounts for credentials
- Varying levels of browser support and security
- After authentication, IIS impersonates the caller



Authentication: ASP.NET 1/2

```
<authentication mode="Windows" />
<identity impersonate="true" />
```

- Windows
  - Uses IIS authentication settings
  - Can impersonate the caller
- Passport
  - Use Microsoft Passport authentication services
  - Microsoft Passport SDK
- None
  - No authentication or using a custom authentication module

Authentication: ASP.NET 2/2

- Forms
  - HTML login form
  - You write code to authenticate the user

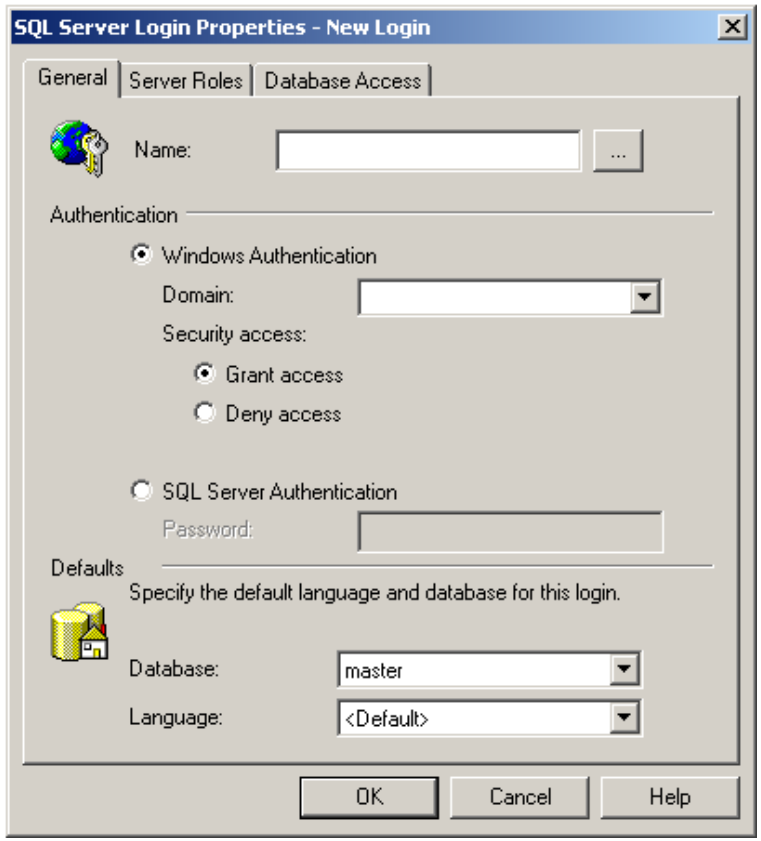
```
<authentication mode="Forms">
    <forms
        loginUrl="login.aspx"
        timeout="30"
        protection="All"
    />
</authentication>
```

Authentication: ASP.NET Identities

- Once the caller is authenticated, what identity will the code execute under?
- The ASPNET user
  - Default settings
- The caller
  - <identity impersonate="true"/>
  - The caller must have a Windows identity
- A particular user
  - <identity impersonate="true"
    userName="" password=""/>

SQL Server: Authentication

- SQL Server
  - Works without Windows auth
- Windows
  - Uses system security services
  - Uses caller's identity for authentication



SQL Server: Identities

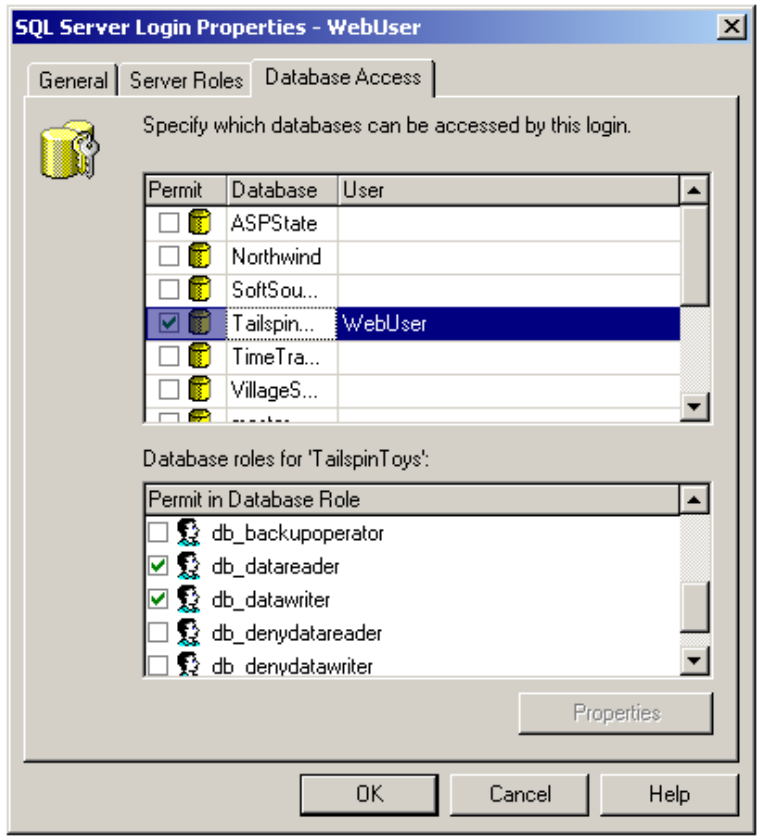
- SQL Server Authentication
  - Don’t use sa!
  - Embed credentials in connection strings
- Windows Authentication
  - Use with a fixed identity:
    - ASPNET user
    - Other ASP.NET identity
    - Serviced Component identity
  - Use when impersonating the client
    - Can defeat connection pooling

SQL Server: Connection Strings

- SQL Server Authentication
  - “User ID=Customer; Password=P@ssw0rd”
  - Connection string is now a secret
- Windows Authentication
  - “Integrated Security=true”
  - “Trusted_Connection=true”

SQL Server: Authorization

- Run with least privilege
  - Create logins for your application
  - Only grant access to necessary databases and roles



Demo: SQL Authentication

Secure Programming Principles

- Defend with Depth
  - Create multiple points of authentication and authorization.

Secure Programming Principles

- Secure secrets
  - Identify the secrets in your application and store them appropriately

Storing Secrets

- Encrypt
  - System.Security.Cryptography
  - DPAPI
    - CryptProtectData
    - CryptUnprotectData
- Secure
  - Consider securing storage with ACLs

Secure Programming Principles

- If you don’t need it, disable it
  - Only enable exactly what you use.

Error Reporting

- Conceal information
  - Logon credentials
  - Implementation details
  - Don’t deploy source files to production servers
- Don’t run in debug mode
  - <compilation debug=“false” />
- Don’t return error details to users
  - <customErrors mode=“RemoteOnly” />

Resources

- Writing Secure Code, Second Edition
  - By Michael Howard, David C. LeBlanc
- Building Secure ASP.NET Applications
  - http://msdn.microsoft.com/library/en-us/dnnetsec/html/secnetlpMSDN.asp
- .NET Framework Security
  - by LaMacchia, Lange, Lyons, Martin, and Price
- Developing Secure Web Applications
  - Microsoft Course #2300
  - http://www.microsoft.com/traincert/syllabi/2300AFinal.asp

SoftSource Consulting

Passionate about technology

- Strategy & Consulting
- Education & Mentoring
- Application Development
- Security

http://www.sftsrc.com