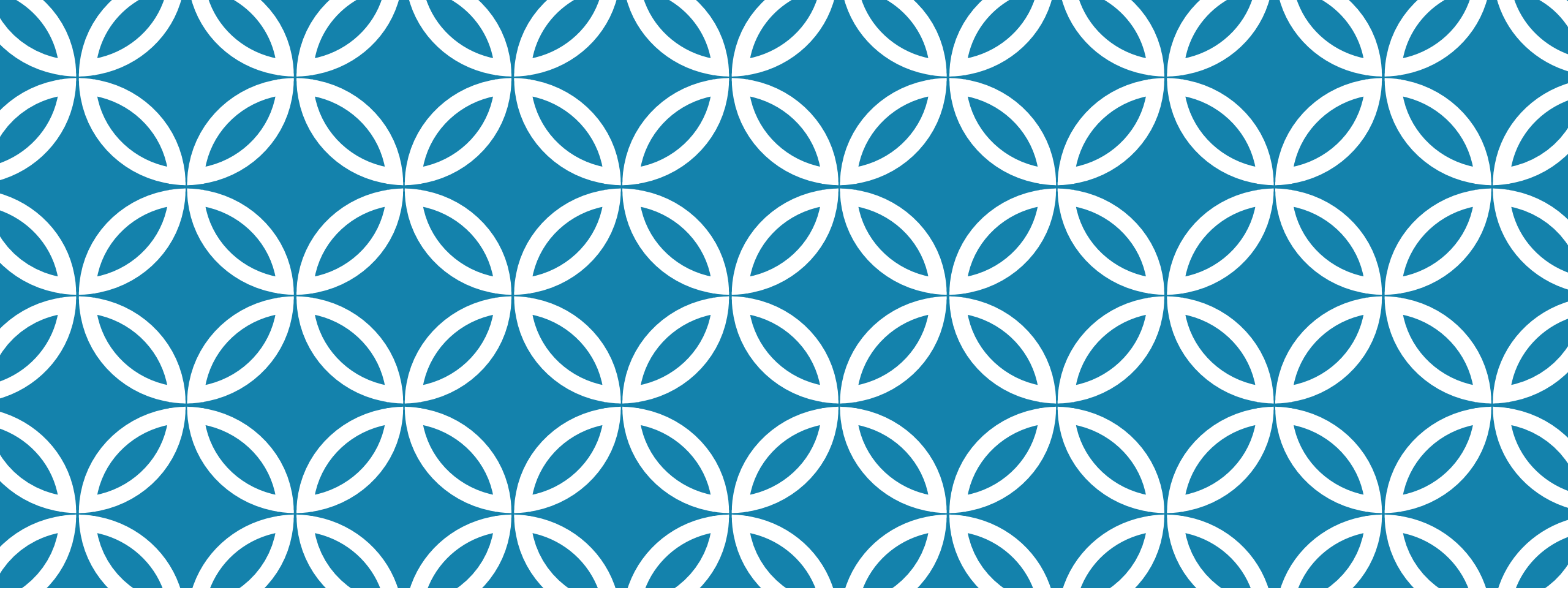


Reducing the attack surface of a .NET OWIN website

By Clarke Bowers

ABOUT THE PRESENTER

Clarke D. Bowers

Clarke Bowers Consulting, LLC

http://www.cbsoftwareengineering.com/

mailto: clarke@cbsoftwareengineering.com

35 years of industry experience

Has architected and developed web sites and web services; public facing and intranet; cloud bases and locally hosted;

You can find this presentation and all the examples at:
https://tinyurl.com/ybwx4b63

OVERVIEW

OWIN is a .NET alternative to the IIS HTTP pipeline

It gives you explicit control of exposed surface of your web application

It run in a variety of environments: Window Server, Windows EXE, IIS Application, Linux

It optionally supports ASP.NET, Routing and MVC

OWIN defines a standard interface between .NET web servers and web applications.

The goal of the OWIN interface is to decouple server and application, encourage the development of simple modules for .NET web development, and stimulate the open source ecosystem of .NET web development tools.

http://owin.org/, http://owin.org/spec/CommonKeys.html and https://docs.microsoft.com/en-us/aspnet/aspnet/overview/owin-and-katana/an-overview-of-project-katana

RECOMMENDED TOOLS

Visual Studio 2017 (Community Edition is free)

Postman (Chrome application)

Fiddler (https://www.telerik.com/fiddler)

Katana (open source implementation of OWIN for .NET)

- https://github.com/aspnet/AspNetKatana/

SQL Server Profiler 2016

AGENDA

IAppBuild, Run & Use

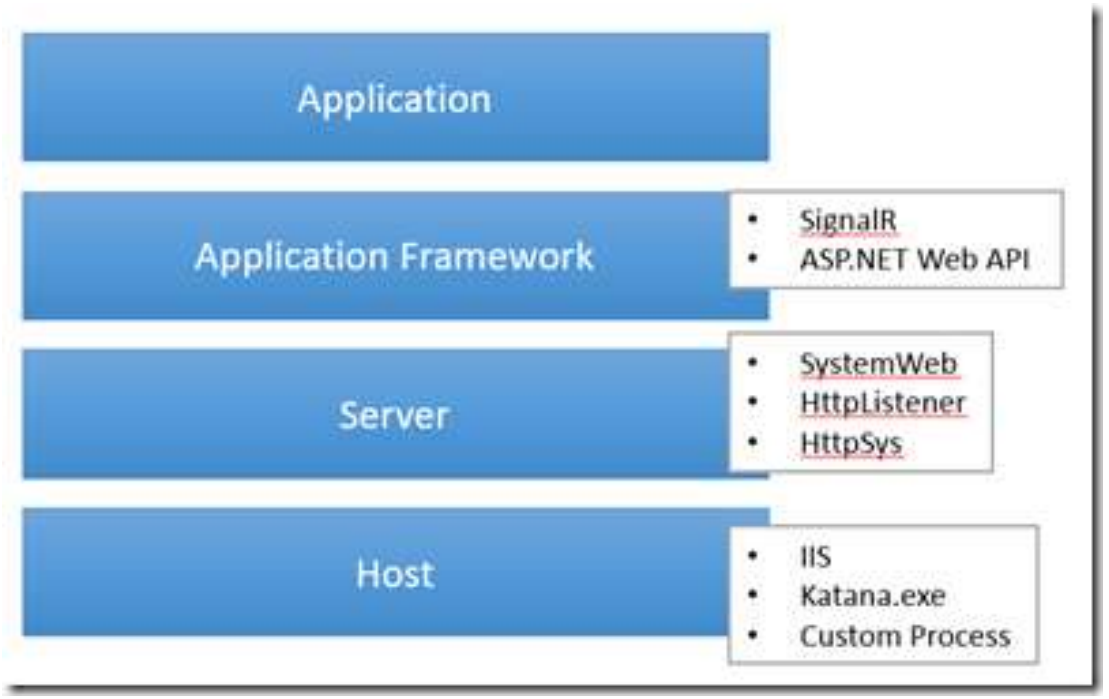
File exposure

Routing & REST service

SQL injection

Object-relational mapping (Entity Framework)

OWIN ARCHITECTURE



The Application is your web site or service

Frameworks are additional (Nuget) packages

Server is http pipeline and threads

Host is the process space, self-hosted, IIS, etc…

WEB APPLICATIONS TODAY

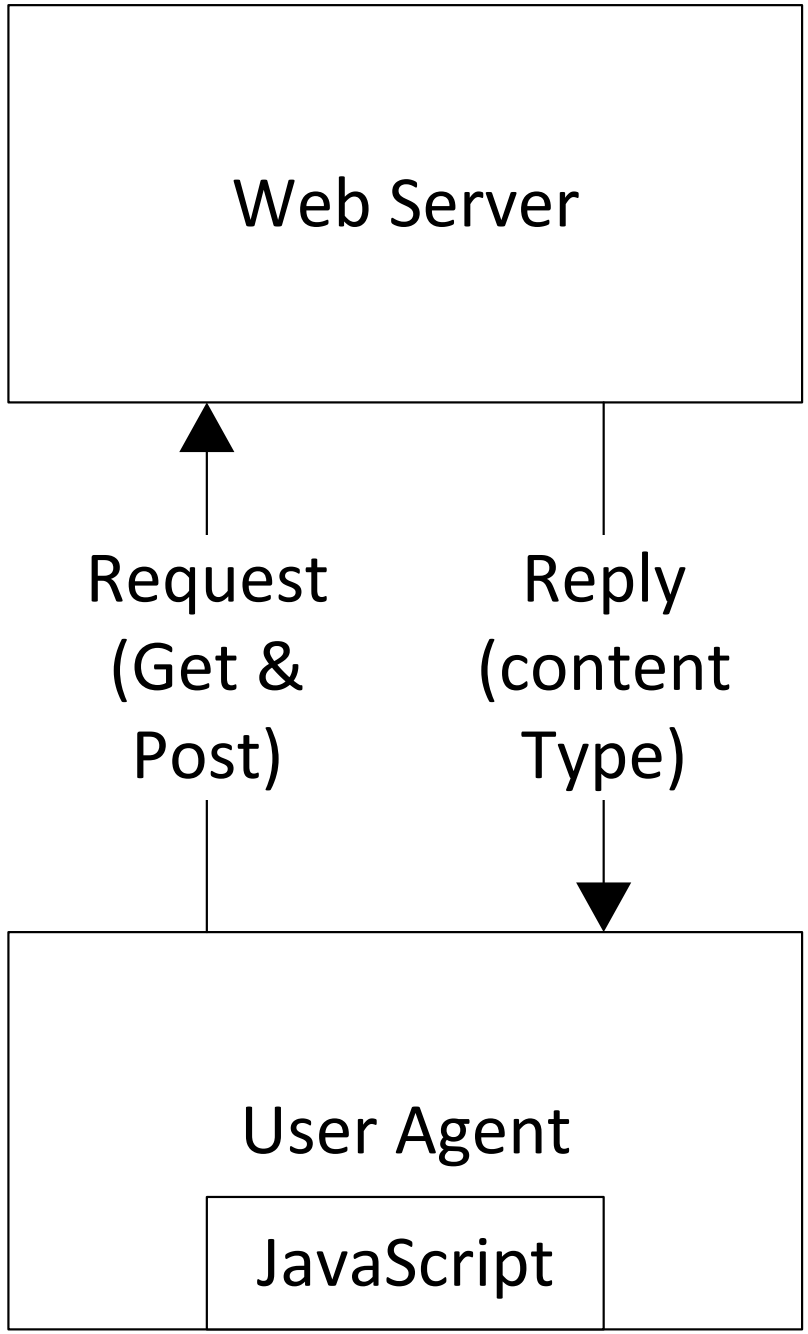
HTTP Request/Reply

SPAs (single page applications) make heavy use of RESTful web services

- Many RESTful services maybe employed for a single web application

Client side packages like Angular allows the business logic to move to the client

- Beware of exposing your core intellectual property



CONSOLE OWIN WEB SITE

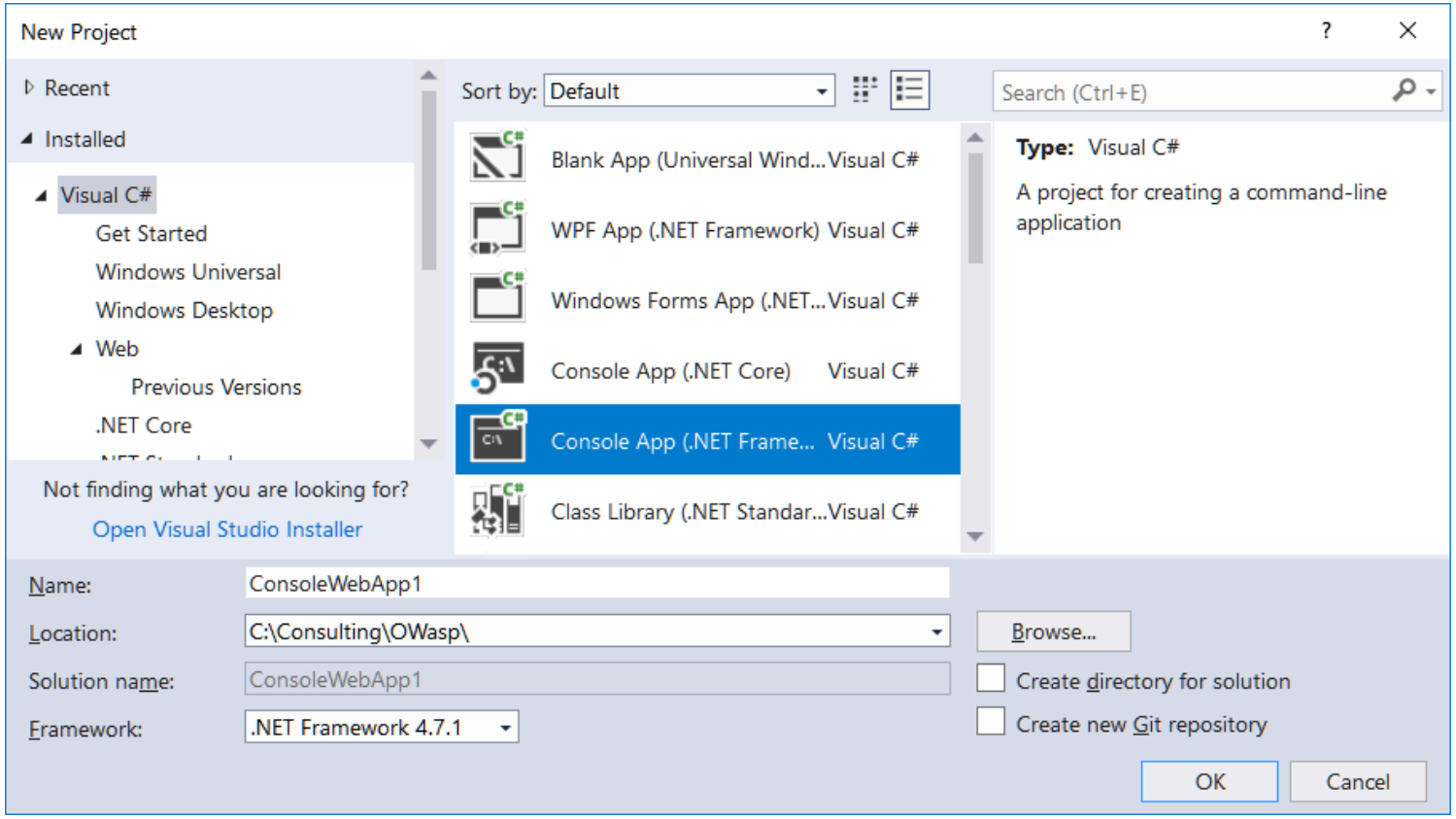
Console App (EXE)

.NET Framework

Add nuget packages

- OWIN
- Microsoft.OWIN
- Microsoft.Owin.SelfHost

Add index.html to root



LISTENING, CONSOLE WEB APPLICATION 1

Starts the web application

Listens on port 9000

Web startup class is generic with *Configuration* method

Index.html in root is not served

Look at traffic with fiddler

IAppBuilder

- Use, adds a handler class to the pipeline
- Run, adds a less flexible method

```
using (Microsoft.Owin.Hosting.WebApp.
 Start<WebStartup>(
 "http://localhost:9000"))
{
  Console.WriteLine(
   "Press [enter] to quit...");
…
public class WebStartup {
    public void Configuration(
       IAppBuilder app)  {
```

FILES, CONSOLE WEB APPLICATION 2

Add additional nuget packages

- Microsoft.Owin.StaticFiles

Create Pages folder and place index.html in it. Marked as content & copy.

Web startup class' *Configuration* method add the file server to the pipeline

Index.html as root is served from pages folder

Look at traffic with fiddler

https://codeopinion.com/asp-net-self-host-static-file-server/

```
var options = new FileServerOptions {
  EnableDirectoryBrowsing = false,
  EnableDefaultFiles = true,
  DefaultFilesOptions = {
    DefaultFileNames = { "index.html" } },
      FileSystem = new PhysicalFileSystem(
        "pages")
};
app.UseFileServer(options);
```

RULES FOR WEB APPLICATION DEVELOPMENT

1. Expose as little as possible. You can always expose more later if required.
2. Do not add code to your site if you do not understand it. Examples are designed to get you started but those examples not to play it safe.
3. Turn on protection (TLS, authentication, authorization, etc…) as soon as possible. This will effect negatively developer productivity, but there is rarely time to add it later.
4. Create a non-interactive OS user to execute the site and only grant it the minimum privileges require to run your site. Generally read and execute on a few directories.
5. Assume whoever will support the application will be less skilled and knowledgeable than you. Companies spend less on maintenance than initial capital project.
6. Determine if your application will be supported by a software developer or a web master. Use IIS to give your web master flexibility via configuration.
7. Never add a back-door. They rarely get shut later and often leak more functionality than intended.

PAGE SERVED

Serves pages from root folder and below from output directory.

File extension must be part of supported file type (http://sourcebrowser.io/Browse/aspnet/AspNetKatana/src/Microsoft.Owin.StaticFiles/ContentTypes/FileExtensionContentTypeProvider.cs#13)

You can create your own content type provider & restrict which pages in the folder

```
HTTP/1.1 200 OK
Content-Length: 215
Content-Type: text/html
Last-Modified: Wed, 16 May 2018 13:42:35 GMT
ETag: "1d3ed3d462cb757"
Server: Microsoft-HTTPAPI/2.0
Date: Wed, 16 May 2018 17:51:29 GMT
<!DOCTYPE html>
<html lang="en“ xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="utf-8" />
    <title>Console Web App 2</title>
</head>
<body>
    Hello Console Web App 2
</body>
</html>
```

ROUTING, CONSOLE APPLICATION 3

Directs a request by URL format to code (a controller)

Add package *Microsoft.AspNet.WebApi.Owin*

Add a folder called *controllers*

Add a controller called MathController

Add HTTP attribute routing to the web startup

Granting access to code is explicit

- No Default Access to code

```
public void Configuration(
    IAppBuilder app) {
     // Configure Web API for self-host.
     var config =
       new HttpConfiguration();

     //  Enable attribute based routing
     config.MapHttpAttributeRoutes();
     app.UseWebApi(config);
```

ATTRIBUTE ROUTING

Controllers must derive from ApiController

Common Verbs: [HttpDelete], [HttpGet], [HttpPost], [HttpPut]

Route Prefix: start of the URL path after the server name

Route: specifies method and optionally parameters format

Example: Verb must be GET and URL must contain two doubles

Try URLS: http://localhost:9000/api/math/add/5/30.2 & http://localhost:9000/api/math/add/hello/30.2

```
[RoutePrefix("api/math")]
public class MathController :
  ApiController
{
    [Route("add/{x:double}/{y:double}")]
    [HttpGet]
    public double Add(double x, double y)
    {
        return x + y;
    }
}
```

SQL, CONSOLE APPLICATION 4

ApiController query a SQL Server

Fetches monumental events since user entered date

Substitutes string parameters into criteria of text bases SQL query

- Does not check contents of string

Try this in postman

http://localhost:9000/api/data/names/01-jan-2009'%20union%20all%20SELECT%20[name]%20FROM%20[WebAppDatabase4].[sys].[Tables]%20where%20''%20=%20‘

And this

http://localhost:9000/api/data/names/01-jan-2009'%20union%20all%20SELECT%20CONCAT([Executive],%20'-',%20[Compensation],%20'-',%20[Amount])%20FROM%20[WebAppDatabase4].[dbo].[ExecutiveCompensation]%20where%20''%20=%20‘

```
[Route("names/{when}")]
[HttpGet]
public IEnumerable<string> Names(string when) {
    var ret = new List<string>();
    using (var cmd = createCommand()) {
        cmd.CommandText =
            "SELECT [Event] FROM [MEvt] " +
            $"WHERE [WHEN] >= '{when}';";
        using (var reader = cmd.ExecuteReader()) {
            while (reader.Read())
                ret.Add(reader.GetString(0));
        } }
    return ret; }
```

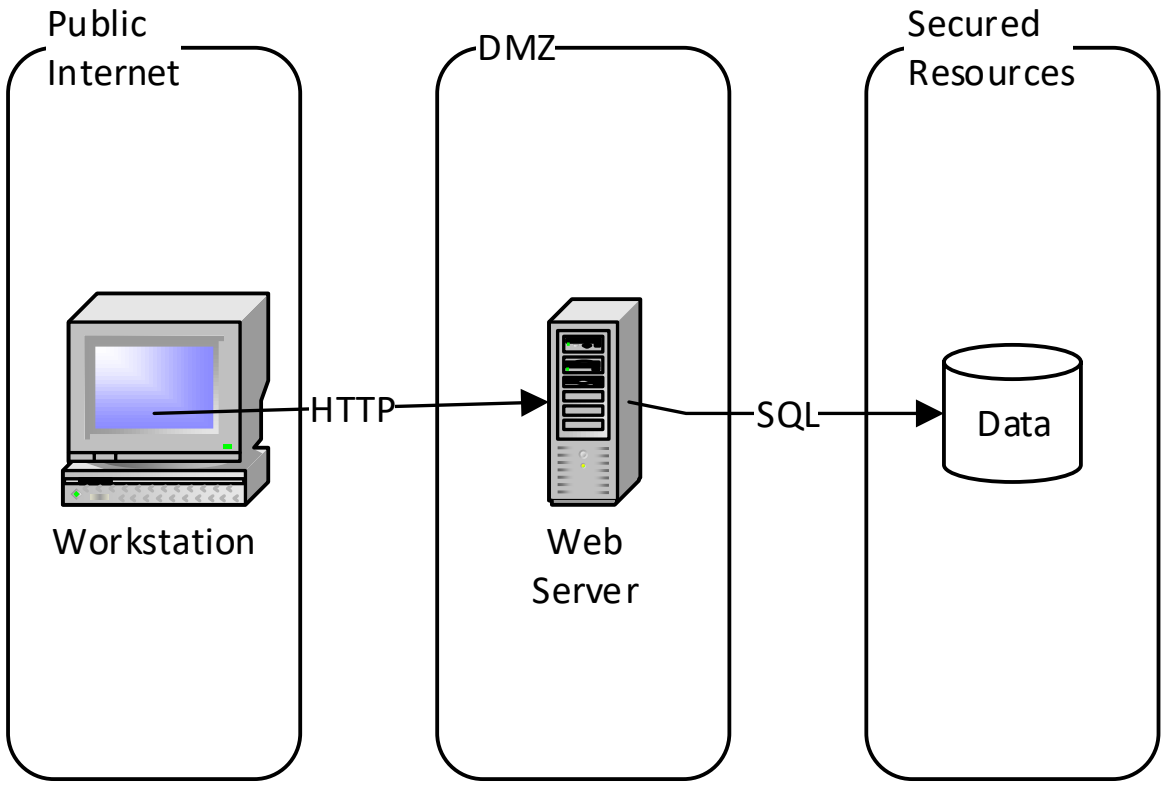
DMZ

Separate web server from public internet with a firewall. Open ports 80,443 only

Separate web server from other corporate asset with a firewall

- Open a pin hole for SQL traffic
- Single IP to single IP on a single port

SQL profiler monitors the traffic from the DMZ to your data

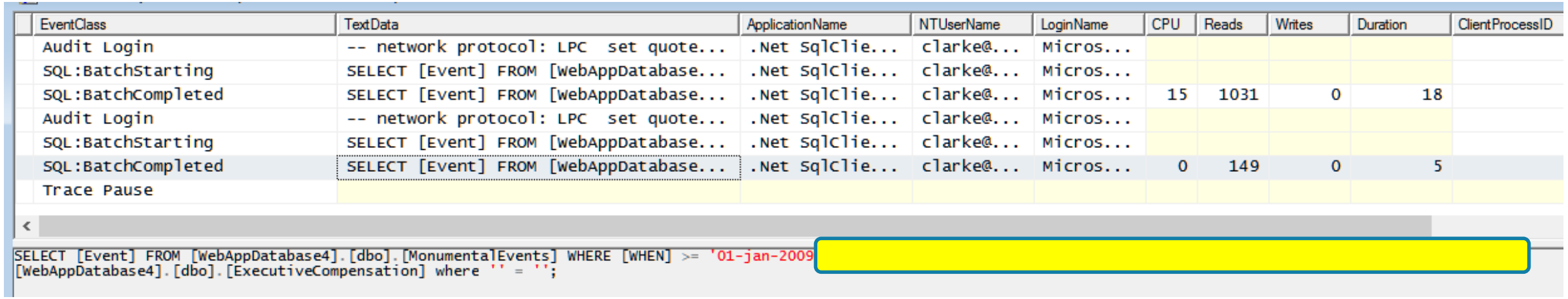


SQL SERVER PROFILER

Monitors traffic to a SQL Server

Can be run from any client and see all traffic to the SQL server

You can see the URL parameter was just appended to the SQL statement



EventClass	TextData	ApplicationName	NTUserName	LoginName	CPU	Reads	Writes	Duration	ClientProcessID
Audit Login	-- network protocol: LPC set quote...	.Net SqlClie...	clarke@...	Micros...					
SQL:BatchStarting	SELECT [Event] FROM [WebAppDatabase...	.Net SqlClie...	clarke@...	Micros...					
SQL:BatchCompleted	SELECT [Event] FROM [WebAppDatabase...	.Net SqlClie...	clarke@...	Micros...	15	1031	0	18	
Audit Login	-- network protocol: LPC set quote...	.Net SqlClie...	clarke@...	Micros...					
SQL:BatchStarting	SELECT [Event] FROM [WebAppDatabase...	.Net SqlClie...	clarke@...	Micros...					
SQL:BatchCompleted	SELECT [Event] FROM [WebAppDatabase...	.Net SqlClie...	clarke@...	Micros...	0	149	0	5	
Trace Pause									

```
SELECT [Event] FROM [WebAppDatabase4].[dbo].[MonumentalEvents] WHERE [WHEN] >= '01-jan-2009
[WebAppDatabase4].[dbo].[ExecutiveCompensation] where '' = '';
```

OBJECT-RELATIONAL MAPPING (ORM)

Entity Framework is the most popular ORM for .NET

Linq-To-Sql builds pre-compiled queries that are parametrized

They cannot be altered at runtime by user input or otherwise

Simple POCO objects represent tuples in relationships (tables).

```
public class MonumentalEvent
{
    public int Id
      { get; set; }
    public DateTime When
      { get; set; }
    public string Event
      { get; set; }
}
```

EF, CONSOLE WEB APPLICATION 5

Add additional nuget packages
- EntityFramework

Add an ADO.NET Entity Data Model
- Choose EF Designer from Database

Select the MonumentalEvents table
- A DbContext with one DbSet will be created

Create a ApiController for the data

Use Linq to pull back all events that starts with the string prefix provided

```
[Route("names/{prefix}")]
[HttpGet]
public IEnumerable<string> Names(string prefix)
{
    using (var db = new Entities())
    {
        var ret = db.MonumentalEvents.
            Where(me => me.Event.StartsWith(prefix)).
            Select(me => me.Event);
        return ret.ToArray();
    }
}
```

EF SQL PROFILE

Trace Entity Framework's Linq-To-Sql query

It is parameterized

Parameter is strongly typed

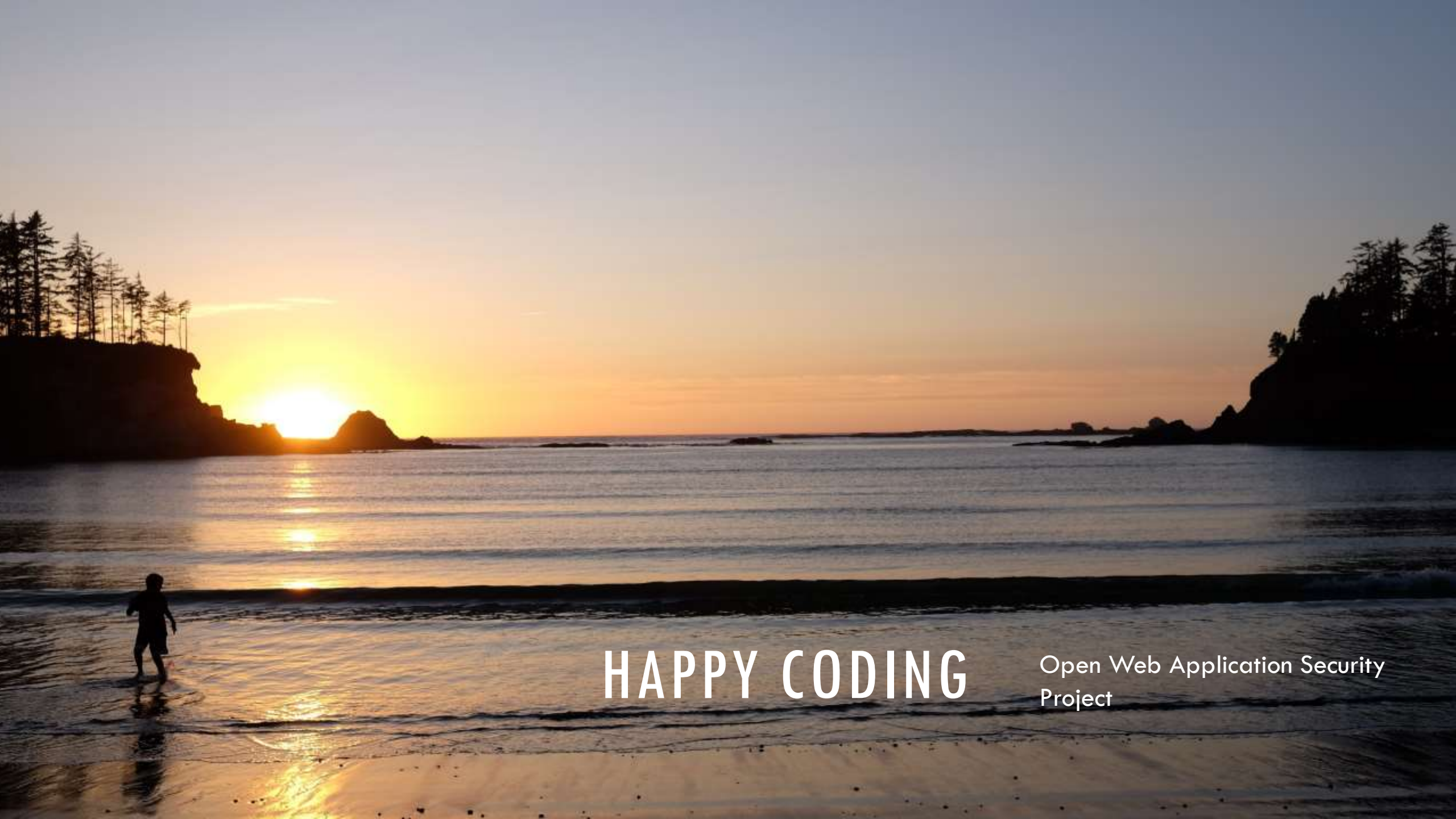
Binding the parameter will not alter the type of statement

Therefore, SQL injection cannot be performed!

```
exec sp_executesql
 N'SELECT [Extent1].[Event]
 AS [Event]
 FROM [dbo].[MonumentalEvents]
 AS [Extent1]
 WHERE [Extent1].[Event]
 [redacted]
 ESCAPE ''~''',
 [redacted]
 @p__linq__0=
  'q'' union all select 10%';
```

RULES FOR WEB APPLICATION DEVELOPMENT, REDUX

1. Expose as little as possible. You can always expose more later if required.
2. Beware of exposing your businesses' core intellectual property
3. Do not add code to your site if you do not understand it. Examples are designed to get you started but those examples not to play it safe.
4. Turn on protection (TLS, authentication, authorization, etc…) as soon as possible. This will effect negatively developer productivity, but there is rarely time to add it later.
5. Create a non-interactive OS user to execute the site and only grant it the minimum privileges require to run your site. Generally read and execute on a few directories.
6. Assume whoever will support the application will be less skilled and knowledgeable than you. Companies spend less on maintenance than initial capital project.
7. Determine if your application will be supported by a software developer or a web master. Use IIS to give your web master flexibility via configuration.
8. Never add a back-door. They rarely get shut later and often leak more functionality than intended.


HAPPY CODING
Open Web Application Security Project