

# INVISIBILITY PURGE

EY

Building a better  
working world

Manipulating Properties of  
**Invisible & Dormant** ASP.Net  
Server-Side Web Controls

Shay Chen, CTO  
[@sectooladdict](#)

Alex Mor, Team Leader  
[@nashcontrol](#)

Hacktics ASC, Ernst & Young

October 27<sup>th</sup>, 2013



# About



- ▶ Formerly a boutique company that provided information security services since 2004.
- ▶ As of 01/01/2011, Ernst & Young acquired Hacktics professional services practice, and the group joined EY as one of the firm's advanced security centers (ASC).



Introducing

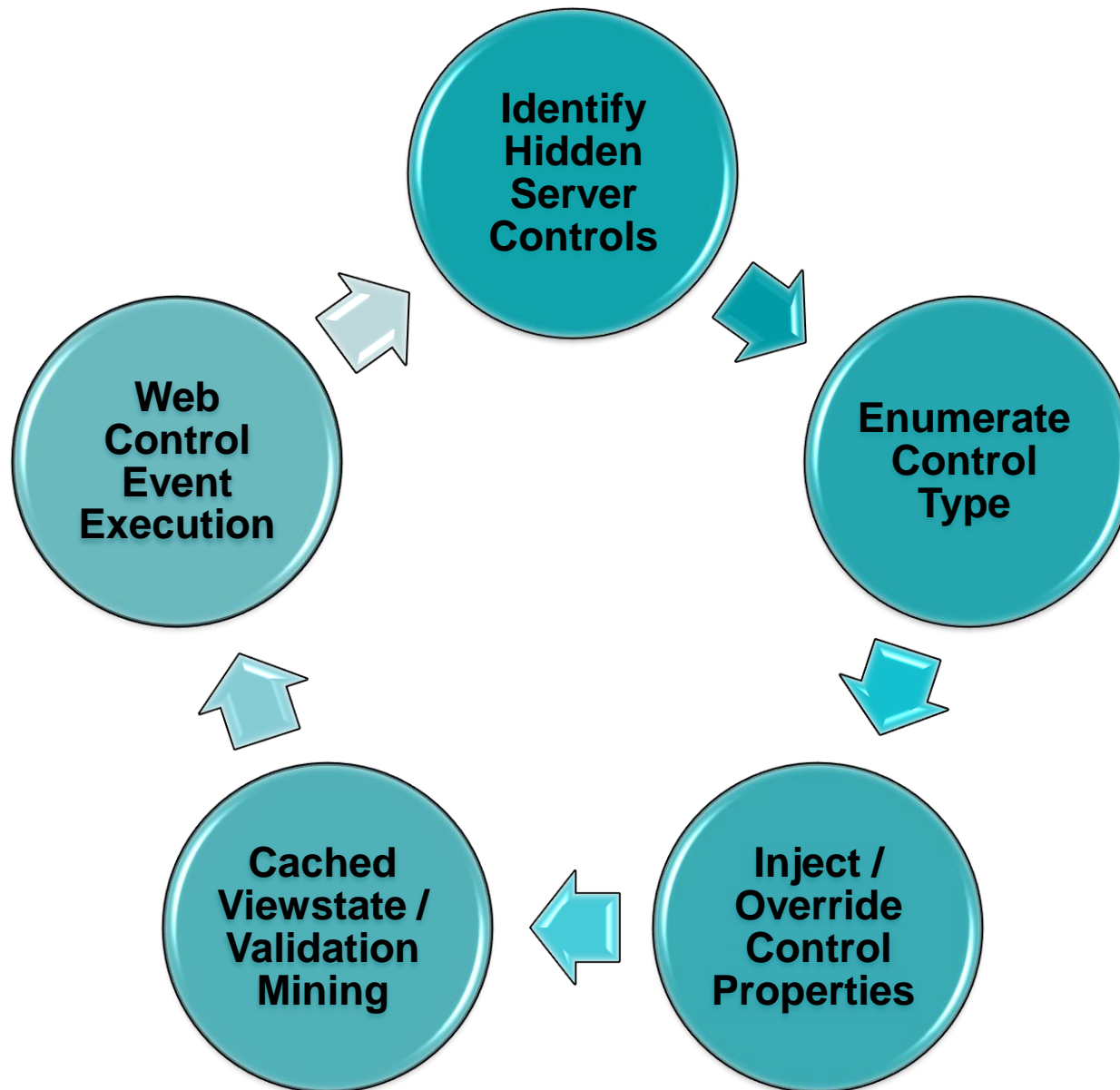
# **VEHICLE!**

**Viewstate Hidden Control Enumerator**  
(formerly known as **ria-scip**)



A project based on a research by **Niv Sela** and **Shay Chen**,  
ZAP Extension Implementation by **Alex Mor**.

# ABUSING ASP.NET MECHANICS



# Property Injection / Property Override

---

- ▶ **Fingerprint visible & hidden control types**
- ▶ **Override server side control properties**
- ▶ **Access additional data sources** in the backend **databases** and systems
- ▶ **Execute application attacks** while **bypassing ASP.net security** mechanisms



# Event Execution Exploits (EodSec)

---

- ▶ Elevate privileges by executing controls/events of high-privileged users
- ▶ Exploit vulnerable code stored in dormant events
- ▶ Corrupt the application data
- ▶ Exceed logical restrictions
- ▶ Etc



# Risk Factors

---

- ▶ Hidden Server Control Identification
- ▶ Server Control Type Fingerprinting
- ▶ Server Control Property Manipulation
- ▶ Server Control Property Injection
- ▶ Execution of Dormant Server Controls
- ▶ Cached Viewstate Reuse



# The Attack Surface of ASP.net / Mono





# Security Features in ASP.net/Mono

---

- ▶ Event Validation
- ▶ Digital Signatures / MAC
  - ▶ Limit to List, Manipulation Prevention
- ▶ Security Filter (XSS)
- ▶ Sandbox
- ▶ Built-in Regular Expressions
- ▶ Etc



# Attack Surface Analysis: Entry Points

---

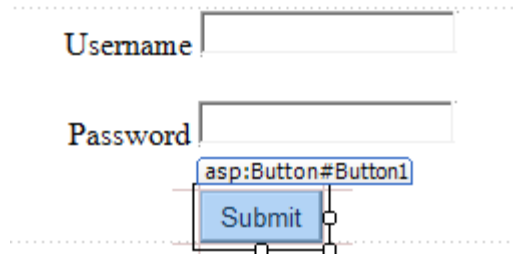
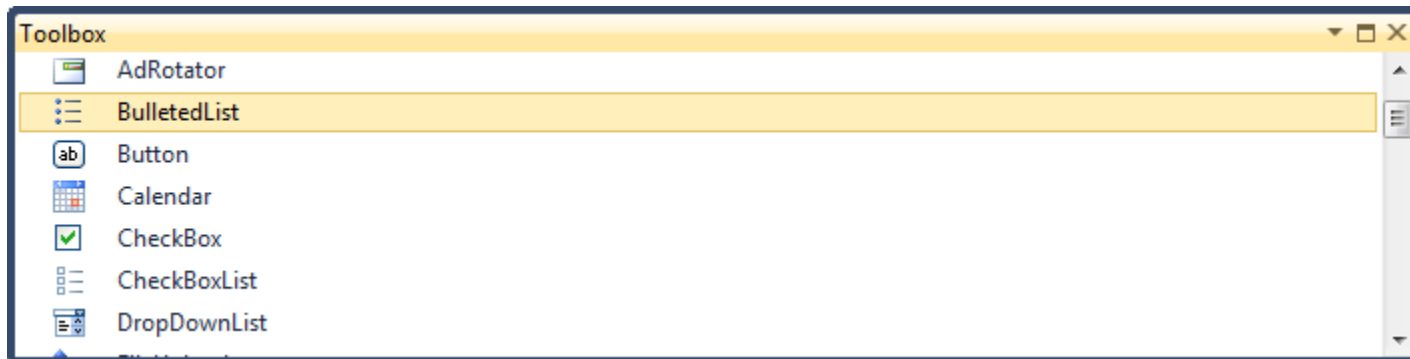
- ▶ **Purpose:** Locating Code to Abuse
  - ▶ Web Pages
  - ▶ Web Service Methods
  - ▶ Global Modules (Filters, Handlers, Etc)
  - ▶ ...
  - ▶ \*Events of Server Web Controls\*



# Server-Side Web Controls

---

- ▶ Rendered into HTML/JS code
- ▶ Include server side implementation
- ▶ Core Controls and Custom Controls (e.g. ascx)



# Server-Side Control Events

---

- ▶ A triggered **server-side code segment**, containing optional functionality (**PostBack/CallBack** in ASP.Net)
- ▶ Client triggering mechanism rely on **EVENTTARGET**, **EVENTARGUMENT** and **VIEWSTATE**
- ▶ Sample Server Side Implementation (**C#, ASP.Net**):

- ▶ **aspx:**

```
<asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Button" />
```

- ▶ **.aspx.cs:**

```
public partial class Demo : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Response.Write("Hello World");
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        Session["action"] = "alterContent";
    }
}
```



# Client-Side Implementation of Events

---

- ▶ Sample client-side implementation (ASP.Net postback):

```
<form name="form1" method="post" action="WelcomeMirror.aspx" id="form1">
<div>
<input type="hidden" name="__EVENTTARGET" id="__EVENTTARGET" value="" />
<input type="hidden" name="__EVENTARGUMENT" id="__EVENTARGUMENT" value="" />
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value="/WEPDWUKLTY1M'
</div>

<input type="button" name="Button1" value="View Service Status" onclick="javascript:__doPostBack('Button1','')">

<script type="text/javascript">
//
var theForm = document.forms['form1'];
if (!theForm) {
    theForm = document.form1;
}
function __doPostBack(eventTarget, eventArgument) {
    if (!theForm.onsubmit || (theForm.onsubmit() != false)) {
        theForm.__EVENTTARGET.value = eventTarget;
        theForm.__EVENTARGUMENT.value = eventArgument;
        theForm.submit();
    }
}
//]]&gt;
&lt;/script&gt;</pre></div><div data-bbox="46 921 98 988" data-label="Image"><img alt="Blue circular logo with a stylized flower or star design inside."/></div><div data-bbox="447 933 514 958" data-label="Page-Footer"><p>Page 13</p></div><div data-bbox="911 935 954 969" data-label="Page-Footer"><p>EY</p></div>
```

# The Structure of the Viewstate Field

## ▶ Viewstate HTML Structure:

```
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"  
value="/wEPDwULLTEyNTIyMjExOTQPBZBYCAGMPZBYIAgEPDxYCHgdWaxNpYmxlaGRkAgMPDxYCHwBoZGQCBO8PFg  
IeB0VuyWJzZWROZGQCBw8PFgIfAWNhkZGQd1dsROoEayc+I/Kt9vZTA3JvsHg==" />
```

▼ ViewState v2.0 compatible [MAC enabled]

▼ **Pair**

▼ **Pair**

```
string 65025323
```

▼ **Pair**

**null**

▼ **List**

int 3

▼ **Pair**

**null**

▼ **List**

int 3

▼ **Pair**

▼ **Pair**

▼ **List**

string Visible

```
boolean false
```

null

**null**

- ▶ **Serialized into Base64\***
- ▶ **Signed (MAC), clear-text or encrypted**



# Event Validation Drill Down

---

- ▶ Independent Events (e.g. buttons with `usesubmitbehavior=false`, etc)
- ▶ Programmatic vs. Declarative

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WelcomeChanged.aspx.cs"
    EnableEventValidation="true" EnableViewStateMac="true" Inherits="ViewStateControls.WelcomeChanged" %>

<input type="hidden" name="__EVENTTARGET" id="__EVENTTARGET" value="" />
<input type="hidden" name="__EVENTARGUMENT" id="__EVENTARGUMENT" value="" />
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwULLTEyNTIyMjExOTQpZBYCagMPZBYIAgEPDxYCHgdWaxNpYmxlaGRkAgMPDxYCHwBoZGQCBQ8PFg
IeB0VuYWJsZWROZGQCBw8PFgIfAWhkZGQd1dSROoEayc+I/Kt9vZTA3JvsHg==" />

    <input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATION"
value="/wEwCAK+2oTRBwLWlM+bAgLs0bLrBgKgwpPxgDKF2fXbAwLPhrqxDwLq79fGCQLJx9vaDXc
/16KxF3ZQYvulQC0yedGi1oA7" />

    <input type="button" name="Button6" value="Button6"
onclick="javascript:__doPostBack('Button6','') " id="Button6" />
```



# The Event Validation Mechanism

## ► Name/Value GetHashCode Formula

```
if ([ControlValue] == null)
    return GetStringHashCode([ControlName]);
else
    return GetStringHashCode([ControlName]) ^ GetStringHashCode([ControlValue]);
```

### The EventValidation Field

▼ ViewState v2.0 compatible [MAC is not enabled]

#### ▼ List

int	-1280308489	<-Viewstate
int	-1314758625	Hashcode
int	-1314758624	
int	-1314758619	
int	2087245738	<-Control
int	2087245739	Hashcodes
int	2087245736	
int	-1314758618	
int	2087245737	
int	2087245736	
int	2087245739	
int	0	

- MachineKey and MAC
- Control Name/Value Verification, Prior to Event Execution
- Include viewstate hashcode
- Included in the HTML:

```
<input type="hidden" name="__EVENTVALIDATION"
value="/WEWCAK+2oTRBwLWlM+bAgLs0bLrBgKgwpPxgDKF2f/
/16KxF3ZQYvulQC0yedGi1oA7" />
```





# Evidence of Hidden Controls & Type

## ► Visible/Enabled Controls:

### Control Panel - Zone 1

View Service Status

Shutdown Service

Send Event Notification

Logout

#### Request

Raw Params Headers Hex ViewState

```
▼ ViewState v2.0 compatible [MAC enabled]
  ▼ Pair
    ▼ Pair
      string 65025323
      null
      null
```

#### EventValidation

#### (Viewstate Decoder):

```
▼ ViewState v2.0 compatible [MAC enabled]
  ▼ List
    int 1677238116
    int 1757590412
    int -2134092357
    int 594790998
    int 1835837676
    int 998075525
    int -568008416
```

## ► Invisible / Disabled Controls (Properties in Viewstate):

### Control Panel - Zone 1

View Service Status

Send Event Notification

Logout

Server Is Up

#### Request

Raw Params Headers Hex ViewState

```
int 3
▼ Pair
  ▼ Pair
    ▼ List
      string Visible
      boolean false
    null
    null
  int 5
```

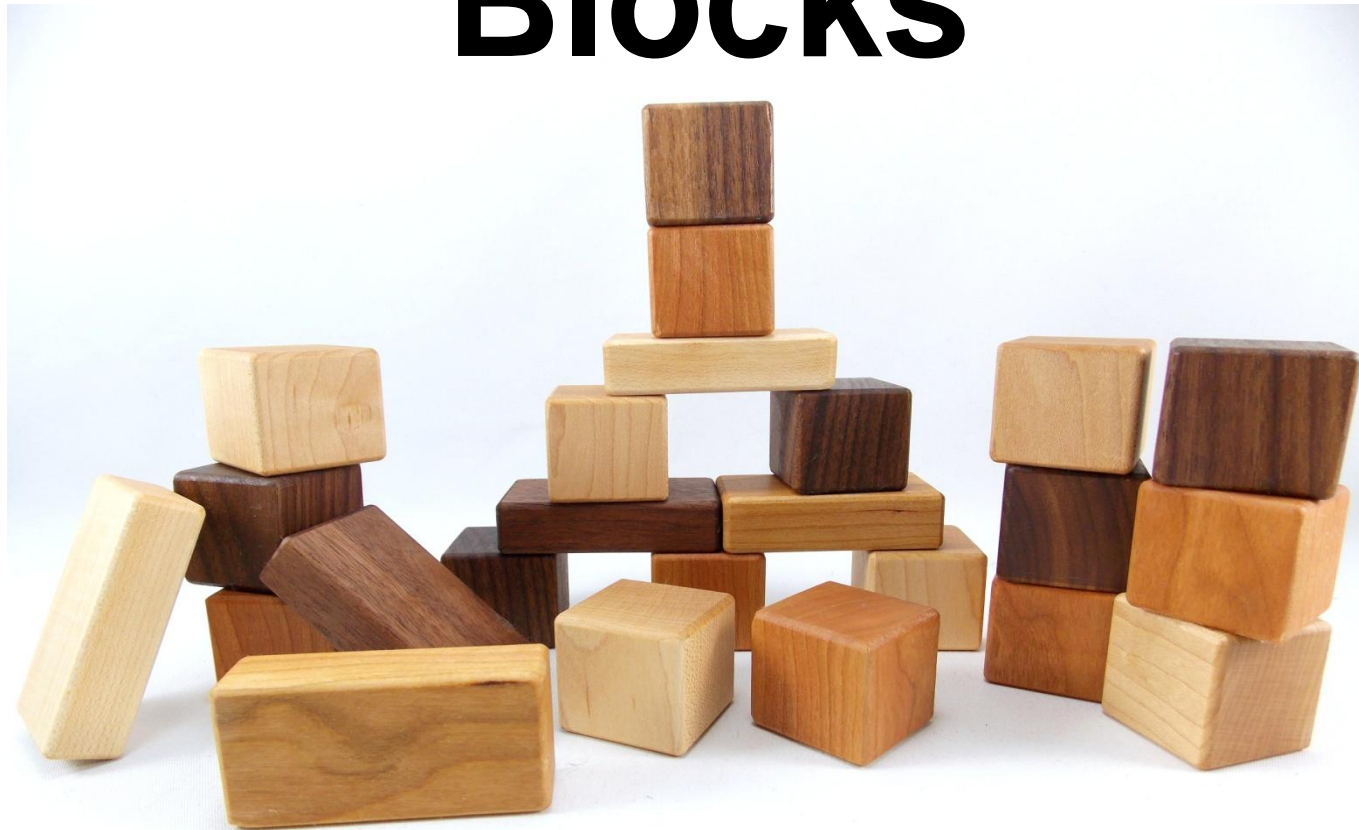
#### EventValidation

#### (Viewstate Decoder):

```
▼ ViewState v2.0 compatible [MAC enabled]
  ▼ List
    int -47520392
    int 1757590412
    int 594790998
    int 1835837676
    int 998075525
    int -568008416
```



# Server Controls / Props: Application Building Blocks



# Fingerprinting Server Controls

---

- ▶ **Control Viewstate Presence**
  - ▶ Collection of Properties for Each Control
  - ▶ Boolean, Numeric, Text, etc
  - ▶ Properties **reloaded** from the viewstate
- ▶ **Rule of Thumb – Properties in Viewstate**
  - ▶ Programmatic value manipulation
  - ▶ Significant In-Design Modifications



# Mapping Viewstate Reloaded Props

---

```
log.txt
78 C:\Documents and Settings\Administrator\Desktop\withViewState\..\BaseDataList.cs
79 object obj = this.ViewState["DataSourceID"];
80 C:\Documents and Settings\Administrator\Desktop\withViewState\..\BaseDataList.cs
81 this.ViewState["DataSourceID"] = value;
82 C:\Documents and Settings\Administrator\Desktop\withViewState\..\BaseDataList.cs
83 object obj = this.ViewState["UseAccessibleHeader"];
84 C:\Documents and Settings\Administrator\Desktop\withViewState\..\BaseDataList.cs
85 this.ViewState["UseAccessibleHeader"] = value;
86 C:\Documents and Settings\Administrator\Desktop\withViewState\..\BaseDataList.cs

log.txt
1056 C:\Documents and Settings\Administrator\Desktop\withViewState\..\HotSpot.cs
1057 string text = (string)this.ViewState["AccessKey"];
1058 C:\Documents and Settings\Administrator\Desktop\withViewState\..\HotSpot.cs
1059 this.ViewState["AccessKey"] = value;
1060 C:\Documents and Settings\Administrator\Desktop\withViewState\..\HotSpot.cs
1061 object obj = this.ViewState["AlternateText"];
1062 C:\Documents and Settings\Administrator\Desktop\withViewState\..\HotSpot.cs
1063 this.ViewState["AlternateText"] = value;
```



# Demo:

## Inject / Override the GridView sqlDataSource and key fields

### Employees List

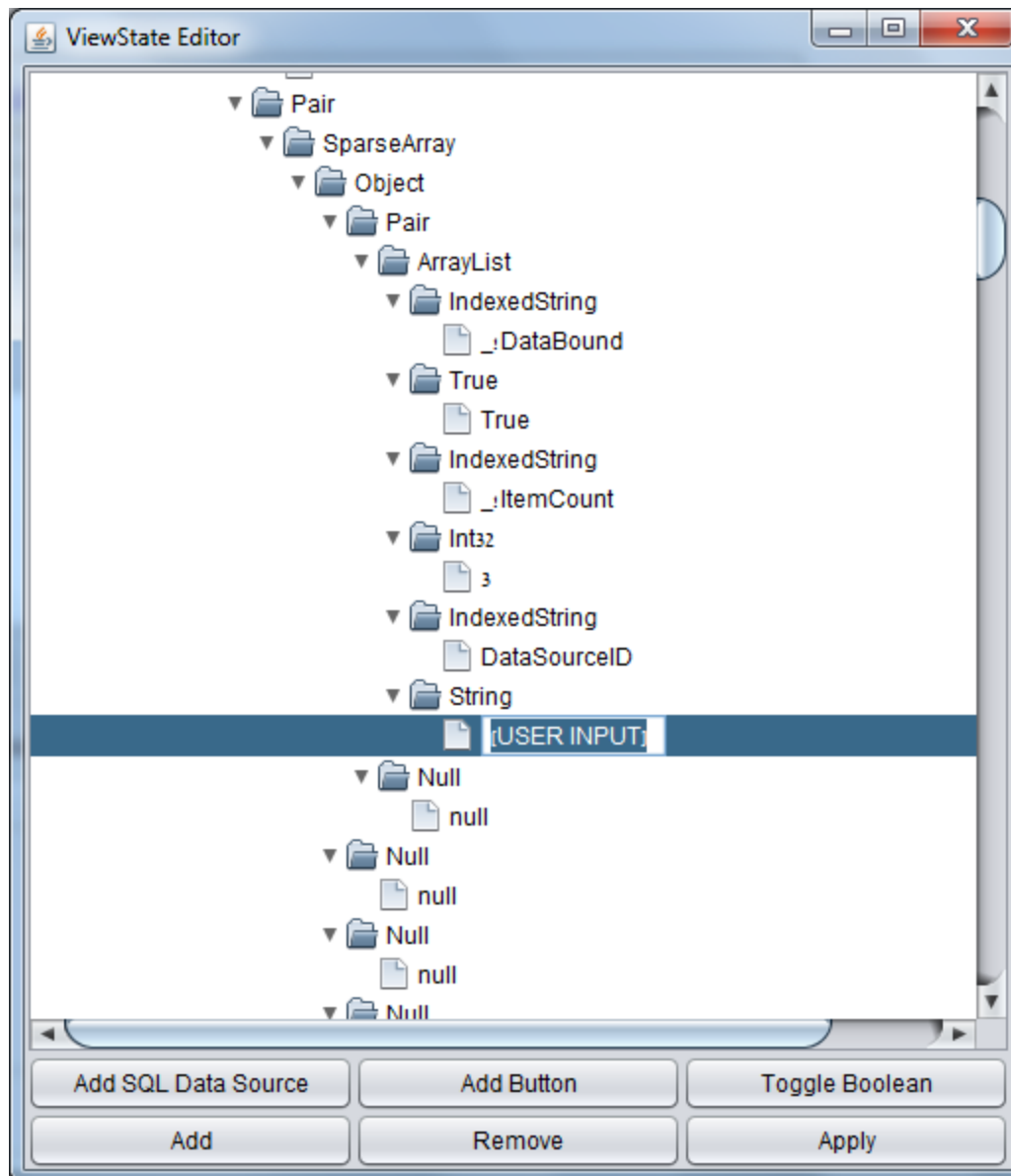
Data Source: SqlDataSource2

Data Field: account\_number

account_id	account_number
1	234234234
2	675675675
3	345345345

Manipulate ViewState

Manipulate ViewState (Client)



# Overriding Control Properties

---

## ▶ Reusing Valid Viewstates

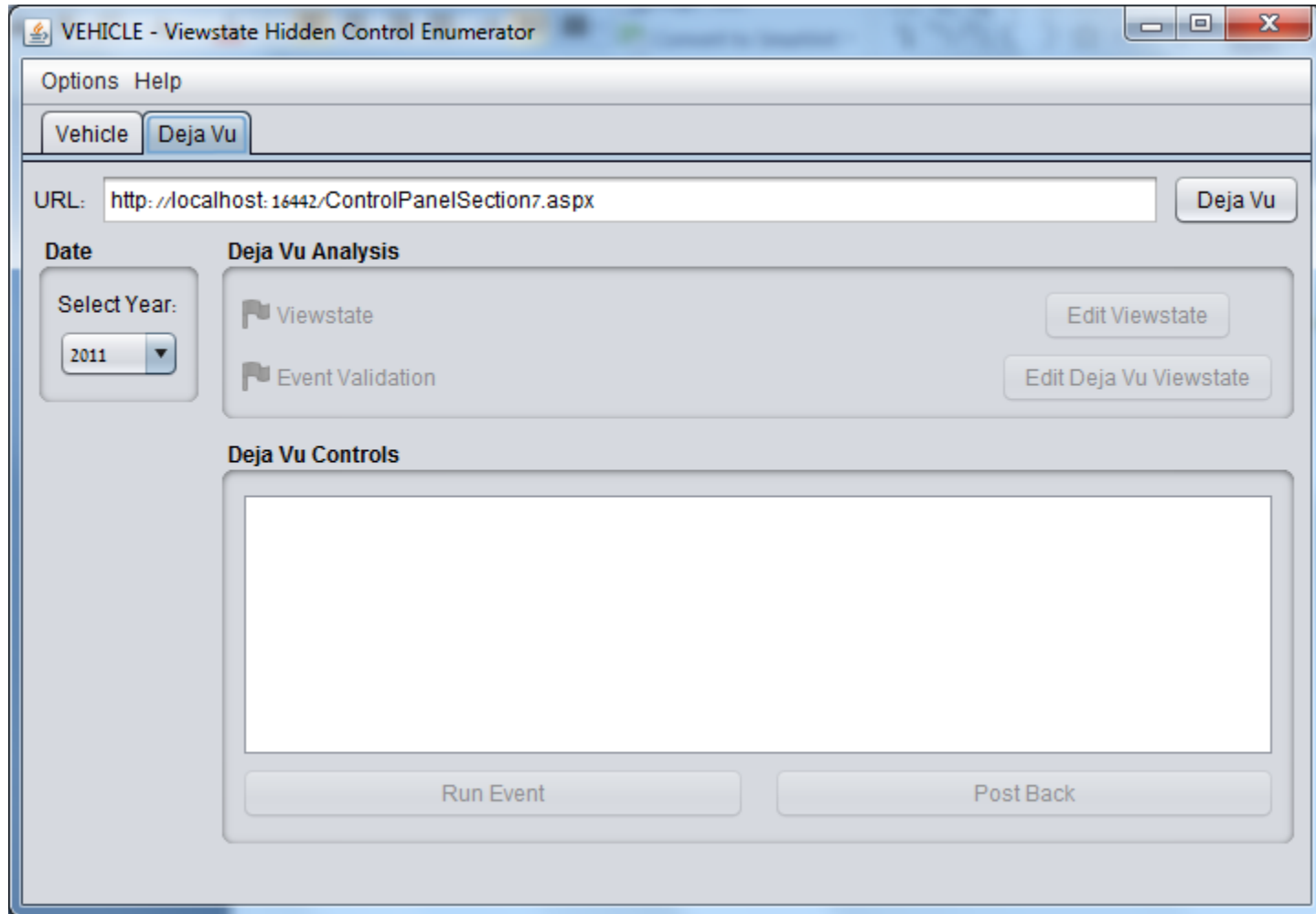
- ▶ MAC Compatibility
- ▶ Mining Sources
- ▶ Structure Similarity

## ▶ Reconstructing Unsigned Viewstates

- ▶ Structure and Order of Properties
- ▶ Insertion of Value Pair Blocks



# Mining VS/EV from Cache Sources





# Hidden Web Controls: Archetypes

The background of the slide is a photograph of a library. It features tall wooden bookshelves filled with numerous books, mostly with brown and red spines. In the center, a wooden door is slightly open, showing a bright, out-of-focus area beyond it, possibly a window or an exit. The text "Hidden Web Controls: Archetypes" is overlaid in a large, white, sans-serif font.

# Dormant Server Web Controls, 1 of 3

## ► Commented Out Controls

- Commented out using HTML comments (<!-- -->)
- **Rendered inside an HTML comment**, but the server code is still active.

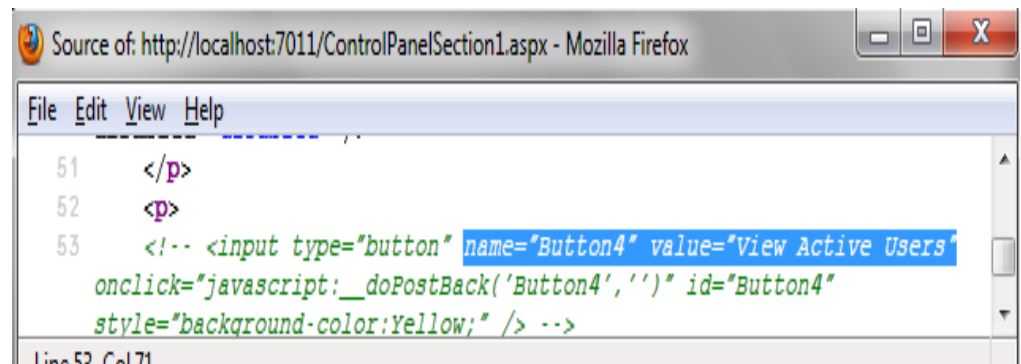
```
<!-- <asp:Button ID="Button4" runat="server" onclick="Button4_Click" |
      Text="View Active Users" UseSubmitBehavior="False" /> -->
protected void Button4_Click(object sender, EventArgs e)
{
    Response.Write("<center><b>Active Users</b></center>");
}
```

## Control Panel - Zone

**View Service Status**

Send Event Notification

Logout



# Dormant Server Web Controls, 2 of 3

---

## ► Disabled Controls

- The control **enabled** property is set to **false** (Server)
- Rendered with the **disabled="disabled"** HTML property
- Rendered **without** an input postback method

```
Send Event Notification    Button3.Enabled = false;  
<input type="button" name="Button3" value="Send Event Notification" id="Button3" disabled="disabled" />
```

### Control Panel - Zone 1

View Service Status

Send Event Notification

Logout



# Dormant Server Web Controls, 3 of 3

## ► Invisible Controls

- The control **visible** property is set to **false**
- **Not rendered** in the presentation layer, but the **code** is still **active**

```
Button2.Visible = false;
```

Welcome **admin**

### Control Panel - Zone 1

**View Service Status**

**Shutdown Service**

Send Event Notification

Logout

Welcome **user1**

### Control Panel - Zone 1

**View Service Status**

Send Event Notification

Logout



# Dormant Events of Web Controls

---

- ▶ **Dormant Events of Visible Controls**
  - ▶ **Declarative:** Optional events of controls with multiple events
  - ▶ **Programmatic:** Optional event listeners registered in the code level for controls with at least one active event





# Dormant & Invisible Control Execution



# Commented Control Event Execution

---

- ▶ **Prerequisites (ASP.Net)** - Commented Out Controls:
  - ▶ Comment server control using HTML comments
  - ▶ Event code does not include privilege validation
  
- ▶ **Process**
  - ▶ “**uncomment**” the HTML control and execute the event, or send the appropriate values directly.
  
- ▶ **Advantages**
  - ▶ Exploit works **despite** the **Viewstate MAC** AND **EventValidation**.



# Commented Control Execution, Cont.

VEHICLE - Viewstate Hidden Control Enumerator

Options Help

Vehicle Deja Vu

URL:  Get

**Page Analysis**

☒ Viewstate (Editable).  
☐ ViewState Signed (MAC found).  
☐ ViewState Encrypted  
☒ Event Validation.  
☐ Event Validation Signed (MAC found).

Invisible Controls Found in ViewState.  
 Disabled Controls Found.  
 Commented Controls Found.

Load Controls From History  
Add Controls From URL

**Page Controls**

Visible:  
Button1  
Buttons

Commented or Disabled:  
Button4  
Button3  
TextBox1

Add Enumerate Controls Blind Control Enumeration

**Enumeration Results:**

Control Name	URL	Hidden

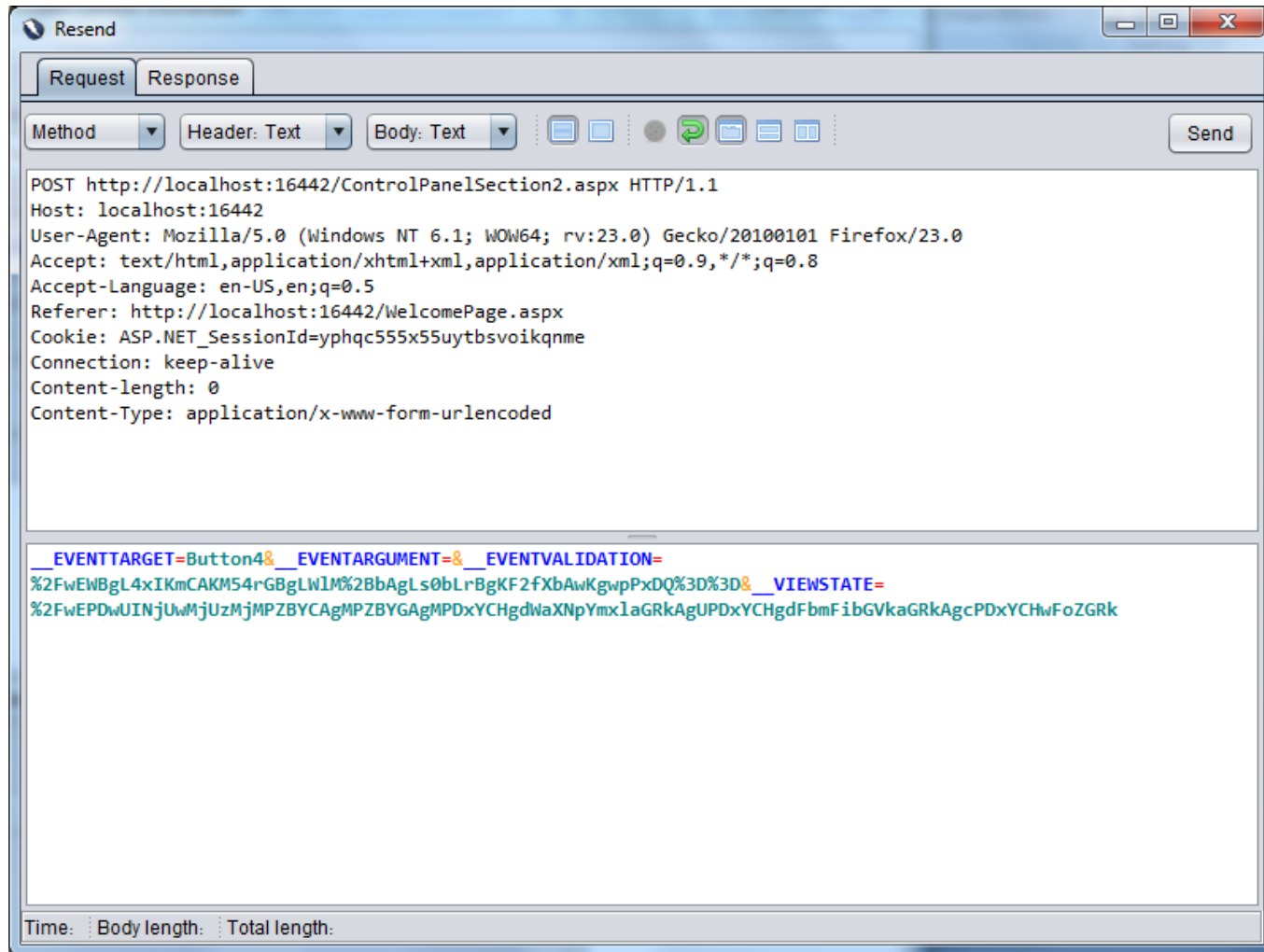
**Events:**

Run Event Post Back





# Commented Control Execution, Cont.



# Disabled Control Event Execution

---

- ▶ **Prerequisites (ASP.Net / Mono)** - Disabled Controls:
  - ▶ The control “enabled” server property is set to FALSE
  - ▶ Event code does not include privilege validation
  
- ▶ **Process**
  - ▶ ASP.Net/Mono: Forge a postback / callback call, or send the appropriate values directly.
  - ▶ Mono: delete the viewstate (!)
  
- ▶ **Advantages**
  - ▶ Exploit works despite an active **Viewstate MAC** AND **EventValidation**.



# Forging a Disabled Control Event

---

## ▶ Forging aPostBack / Callback Method Call

### ▶ Why does it work?

- ▶ ASP.Net: Temporarily disabled controls are a feature
- ▶ Mono: EventValidation' Viewstate hash-code issue

### ▶ How does it work?

- ▶ The control name is exposed in the disabled control

```
<input type="button" name="Button3" value="Send Event Notification" id="Button3" disabled="disabled" />
```

- ▶ Use an interception proxy to “inject” postback calls into HTML control events, or craft requests manually.



# Disabled Control Execution, Cont.

The screenshot shows the 'VEHICLE - Viewstate Hidden Control Enumerator' application window. The interface includes a menu bar with 'Options' and 'Help'. Below the menu bar are tabs for 'Vehicle' and 'Deja Vu'. A URL field contains 'http://localhost:16442/ControlPanelSection2.aspx' with a 'Get' button. The 'Page Analysis' section has checkboxes for 'Viewstate (Editable)', 'ViewState Signed (MAC found)', 'ViewState Encrypted', 'Event Validation', and 'Event Validation Signed (MAC found)'. To the right of these checkboxes are three status indicators: 'Invisible Controls Found in ViewState.', 'Disabled Controls Found.', and 'Commented Controls Found.'. Below these are buttons for 'Load Controls From History' and 'Add Controls From URL'. The 'Page Controls' section is divided into two panes: 'Visible:' containing 'Button1' and 'Button5', and 'Commented or Disabled:' containing 'Button4', 'Button3' (highlighted), and 'TextBox1'. Below these panes are buttons for 'Add', 'Enumerate Controls', and 'Blind Control Enumeration'. The 'Enumeration Results:' section features a table with columns 'Control Name', 'URL', and 'Hidden'. The 'Events:' section has a large empty box with a red arrow pointing to it, and buttons for 'Run Event' and 'Post Back'.

VEHICLE - Viewstate Hidden Control Enumerator

Options Help

Vehicle Deja Vu

URL:  Get

Page Analysis

☒ Viewstate (Editable). ☐ ViewState Signed (MAC found). ☐ ViewState Encrypted ☒ Event Validation. ☐ Event Validation Signed (MAC found).

Invisible Controls Found in ViewState.  
Disabled Controls Found.  
Commented Controls Found.

Load Controls From History  
Add Controls From URL

Page Controls

Visible:

Button1  
Button5

Commented or Disabled:

Button4  
Button3  
TextBox1

Add Enumerate Controls Blind Control Enumeration

Enumeration Results:

Control Name	URL	Hidden
--------------	-----	--------

Events:

Run Event Post Back



# Invisible Control Event Execution 1/4

---

- ▶ **Prerequisites (ASP.Net / Mono) - Invisible Controls:**
  - ▶ (I) Either the ViewStateMAC OR the EventValidation features must be turned off

```
<%@ Page Language="C#" AutoEventWireup="true" EnableEventValidation="false"|  
<system.web>  
  <pages enableEventValidation="false"/>  
</system.web>  
  
<%@ Page Language="C#" AutoEventWireup="true" EnableViewStateMac="false"  
<system.web>  
  <pages enableViewStateMac="False" />  
</system.web>
```

- ▶ (II) The control “visible” server property is set to FALSE
- ▶ (III) Event code does not include privilege validation



# Invisible Control Event Execution 2/4

---

## ▶ EventValidation is OFF

- ▶ No event validation = ANY event can be executed, regardless of MAC

```
<%@ Page Language="C#" AutoEventWireup="true"  
EnableEventValidation="false" EnableViewStateMac="true" ...%>
```

## ▶ Process

- ▶ Craft a request with valid **EVENTTARGET** value **OR**
- ▶ Inject a custom **Postback/Callback** call to the response HTML, and target the event of the invisible control



# Invisible Control Event Execution 3/4

---

## ▶ EventValidation is ON , ViewState MAC is OFF

- ▶ Forge valid viewstate / eventvalidation fields (no MAC)

```
<%@ Page Language="C#" AutoEventWireup="true"  
EnableEventValidation="true" EnableViewStateMac="false" ...%>
```

## ▶ Process

- ▶ Craft a request using **VEHICLE** or other eventvalidation editors



# Invisible Control Event Execution 4/4

## ► HashCode Generation

```
internal static unsafe int GetStringHashCode(string s)
{
    fixed (char* chPtr = s)
    {
        int num1 = 352654597;
        int num2 = num1;
        int* numPtr = (int*)chPtr;
        int length = s.Length;
        while (length > 0)
        {
            num1 = (num1 << 5) + num1 + (num1 >> 27) ^ *numPtr;
            if (length > 2)
            {
                num2 = (num2 << 5) + num2 + (num2 >> 27) ^ numPtr[1];
                numPtr += 2;
                length -= 4;
            }
            else
                break;
        }
        return num1 + num2 * 1566083941;
    }
}
```





# Error-Based Control Enumeration

- ▶ Accessing invalid control names **will NOT** raise exceptions
- ▶ Accessing protected **will** - if the EventValidation is **ON**

Server Error in '/' Application.

*Invalid postback or callback argument. Event validation is enabled using <pages enableEventValidation="true"/> in configuration or <%@ Page EnableEventValidation="true" %> in a page. For security purposes, this feature verifies that arguments to postback or callback events originate from the server control that originally rendered them. If the data is valid and expected, use the ClientScriptManager.RegisterForEventValidation method in order to register the postback or callback data for validation.*

**Description:** An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

**Exception Details:** System.ArgumentException: Invalid postback or callback argument. Event validation is enabled using <pages enableEventValidation="true"/> in configuration or <%@ Page EnableEventValidation="true" %> in a page. For security purposes, this feature verifies that arguments to postback or callback events originate from the server control that originally rendered them. If the data is valid and expected, use the ClientScriptManager.RegisterForEventValidation method in order to register the postback or callback data for validation.

#### Source Error:

An unhandled exception was generated during the execution of the current web request. Information regarding the origin and location of the exception can be identified using the exception stack trace below.

#### Stack Trace:

```
[ArgumentException: Invalid postback or callback argument. Event validation is enabled using <pages enableEventValidation="true"/> in
System.Web.UI.ClientScriptManager.ValidateEvent(String uniqueId, String argument) +8644649
System.Web.UI.Control.ValidateEvent(String uniqueID, String eventArgument) +69
System.Web.UI.WebControls.Button.RaisePostBackEvent(String eventArgument) +35
System.Web.UI.WebControls.Button.System.Web.UI.IPostBackEventHandler.RaisePostBackEvent(String eventArgument) +10
System.Web.UI.Page.RaisePostBackEvent(IPostBackEventHandler sourceControl, String eventArgument) +13
System.Web.UI.Page.RaisePostBackEvent(NameValueCollection postData) +175
System.Web.UI.Page.ProcessRequestMain(Boolean includeStagesBeforeAsyncPoint, Boolean includeStagesAfterAsyncPoint) +1565]
```



# Blind Control Enumeration

## ► Basic Blind Differentiation Formula:

```
ValidControlEvent = False;
```

```
OriginalResponse = getResponse("Page1.aspx?param=value");
```

```
VerificationResponse = getResponse("Page1.aspx?param=value");
```

```
ConfirmationResponse = getResponse("Page1.aspx?param=value");
```

```
InconsistentContent = VerificationResponse - ReflectedValues -  
TimestampTokens;
```

```
ClearResponse = OriginalResponse - ReflectedValues -  
InconsistentContent - TimestampTokens;
```

```
EventExecResponse =  
getResponse("Page1.aspx?param=value&EVENTTARGET=...");  
EventExecResponse = OriginalResponse - ReflectedValues -  
InconsistentContent - TimestampTokens;
```

```
If (Diff (ClearResponse, EventExecResponse ) > 0) ValidControlEvent = True;
```



# Control Naming Conventions

---

- ▶ **Default:** [ControlType][Number]
  - ▶ Button1, Button2, TextBox1, TextBox2 ...
- ▶ **Default II** (v1.1-v3.5/Master): ctl[ID]\${contentScope}\$...
  - ▶ ctl00\$MainContent\$txtName, ctl00\$Content\$cmdSubmit
- ▶ **Legacy:** [ControlTypeShortCut][Number]
  - ▶ txt1, txt2, btn1, btn2, cmd1, cmd2, lst1, lst2 ...
- ▶ **Custom Legacy:** [ControlTypeShortCut][Logic]
  - ▶ txtUsername, txtPassword, btnSubmit, cmdAddUser ...
- ▶ **Plain:** [Logic]
  - ▶ user, pass, submit, delete
- ▶ **Title Match:** [Title]
  - ▶ Username, Password, Origin, Email, Update



# Hidden/Optional Event Execution 1/2

---

- ▶ **Prerequisites** – Multiple Dormant Events of Controls:
  - ▶ Control assigned with multiple events.
  - ▶ (Calendar control, Custom Controls, etc)
- ▶ **Process:**
  - ▶ Fuzz the **eventargument** field, in addition to **eventtarget**
  - ▶ Eventargument variation can execute **different** server events (for example - V[value] vs. [value])
- ▶ **Advanced:**
  - ▶ Core Events vs. Custom Events
  - ▶ Example: Click, Command, onSelectionChanged, OnVisibleMonthChanged, Etc



## Hidden/Optional Event Execution 2/2

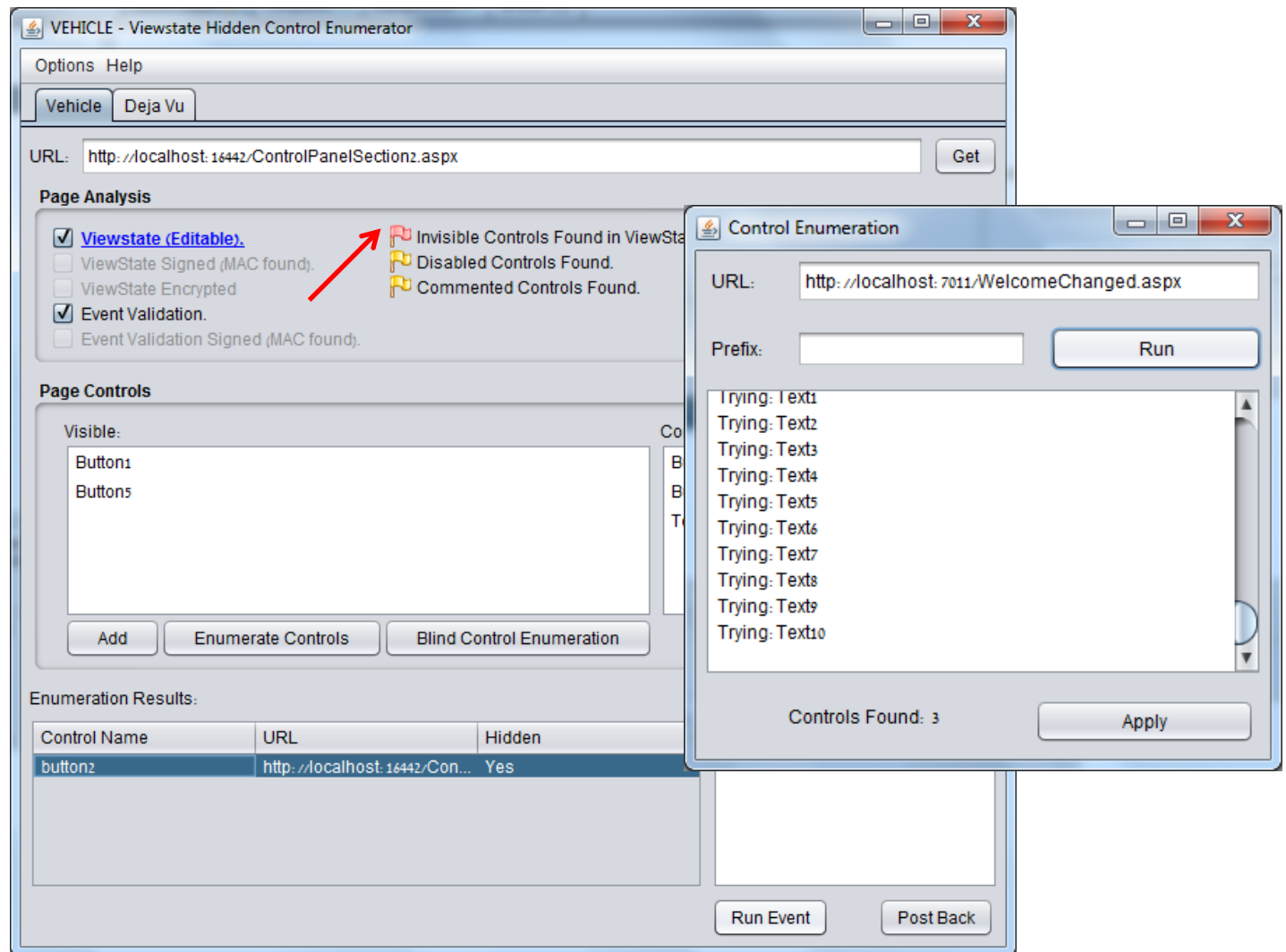
The screenshot shows the code-behind for a calendar control. The code includes two event handlers: `Secret_Click` and `Calendar1_SelectionChanged`. The `Secret_Click` method sets the text of `Label1` to "Secret!!!" in red. The `Calendar1_SelectionChanged` method sets the text of `Label1` to "Normal" in black. The calendar control is shown in the design view, displaying February 2013. The date 28 is highlighted in green, corresponding to the event handler being triggered.

```
<table id="Calendar1" cellpadding="0" cellspacing="0" border="1" title="Calendar" style="width: 100%; border-collapse: collapse;">
    <tr><td colspan="7" style="background-color: Silver; text-align: center; padding: 5px;">
        <table cellpadding="0" cellspacing="0" border="0" style="margin-left: auto; margin-right: auto;">
            <tr><td style="width: 15%; vertical-align: middle; text-align: right;">
                <a href="#" style="color: Black; text-decoration: none;">
                    javascript:__doPostBack('Calendar1','V4749')
                </a>
            </td><td align="center" style="width: 70%; vertical-align: middle;">February 2013</td><td align="right" style="width: 15%; vertical-align: middle;">
                <a href="#" style="color: Black; text-decoration: none;">
                    javascript:__doPostBack('Calendar1','V4808')
                </a>
            </td></tr></table>
            <tr><th align="center" abbr="Sunday" scope="col">Sun</th><th align="center" abbr="Monday" scope="col">Mon</th><th align="center" abbr="Tuesday" scope="col">Tue</th><th align="center" abbr="Wednesday" scope="col">Wed</th><th align="center" abbr="Thursday" scope="col">Thu</th><th align="center" abbr="Friday" scope="col">Fri</th><th align="center" abbr="Saturday" scope="col">Sat</th></tr>
            <tr><td align="center" style="padding: 5px;">
                <a href="#" style="color: Black; text-decoration: none;">
                    javascript:__doPostBack('Calendar1','4775')
                </a>
            </td><td align="center" style="padding: 5px;">
                <a href="#" style="color: Black; text-decoration: none;">
                    javascript:__doPostBack('Calendar1','4776')
                </a>
            </td><td align="center" style="padding: 5px;">
                <a href="#" style="color: Black; text-decoration: none;">
                    javascript:__doPostBack('Calendar1','4777')
                </a>
            </td><td align="center" style="padding: 5px;">
                <a href="#" style="color: Black; text-decoration: none;">
                    javascript:__doPostBack('Calendar1','4777')
                </a>
            </td><td align="center" style="padding: 5px;">
                <a href="#" style="color: Black; text-decoration: none;">
                    javascript:__doPostBack('Calendar1','4777')
                </a>
            </td><td align="center" style="padding: 5px;">
                <a href="#" style="color: Black; text-decoration: none;">
                    javascript:__doPostBack('Calendar1','4778')
                </a>
            </td><td align="center" style="padding: 5px;">
                <a href="#" style="color: Black; text-decoration: none;">
                    javascript:__doPostBack('Calendar1','4778')
                </a>
            </td></tr>
        </table>
    </td></tr>

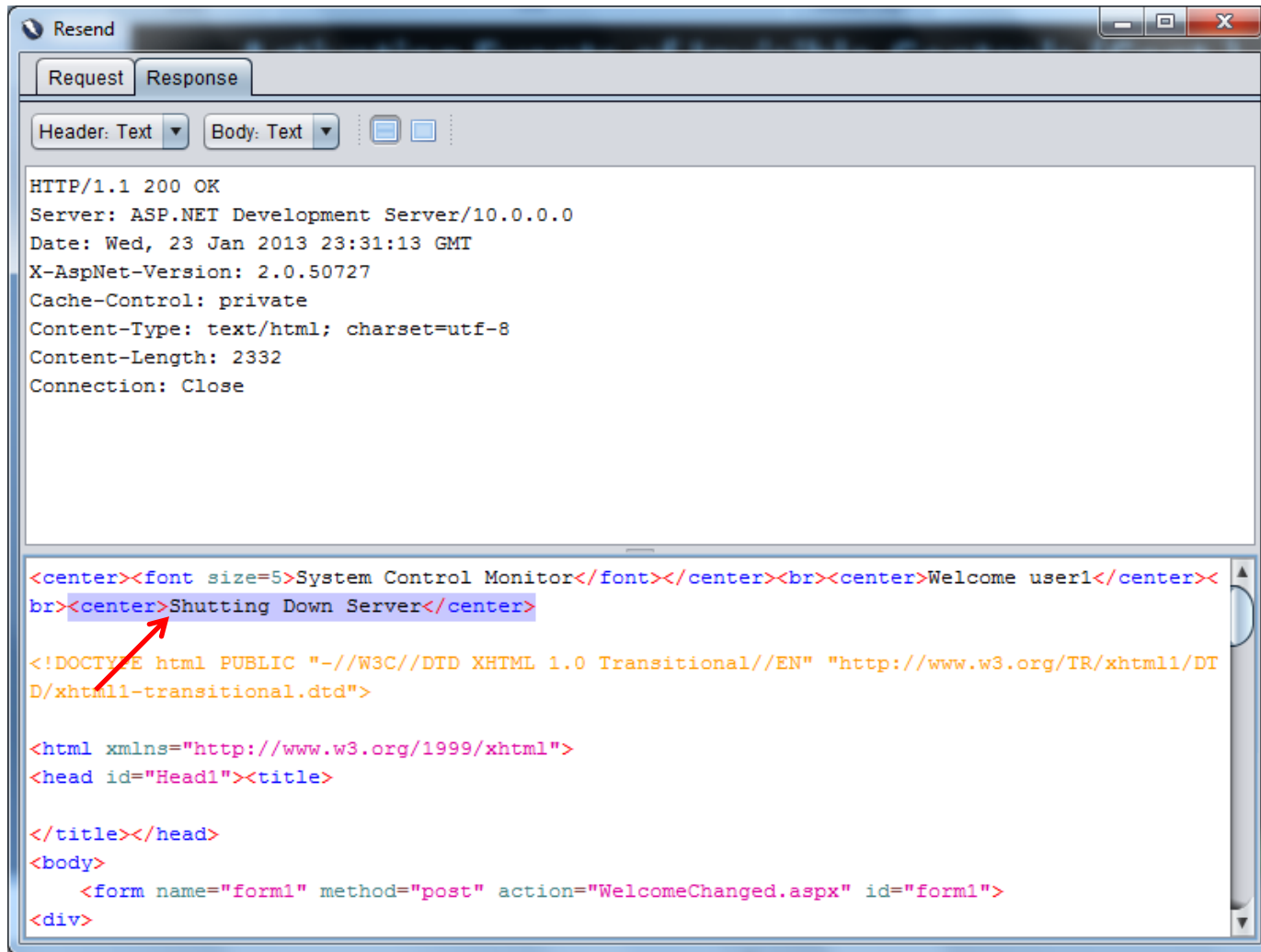
```



# Invisible Control Execution, Cont.



# Invisible Control Execution, Cont.



# Advanced Event Execution Methods



**Executing Events of Invisible Controls  
DESPITE  
EventValidation and Viewstate MAC**



# Missing Callback Code Validation

---

- ▶ Prerequisites – Improper Callback Implementation
  - ▶ Event validation not performed manually in Callback code
  - ▶ Relies on the CALLBACKID and CALLBACKPARAM

```
public void RaiseCallbackEvent(string eventArgument)
{
    try
    {
        //Page.ClientScript.ValidateEvent(this.UniqueID);
        this.OnClick(new EventArgs());
    }
    catch (Exception ex)
    {
        this._eventArg = ex.Message;
        this.OnClick(new EventArgs());
    }
}
```

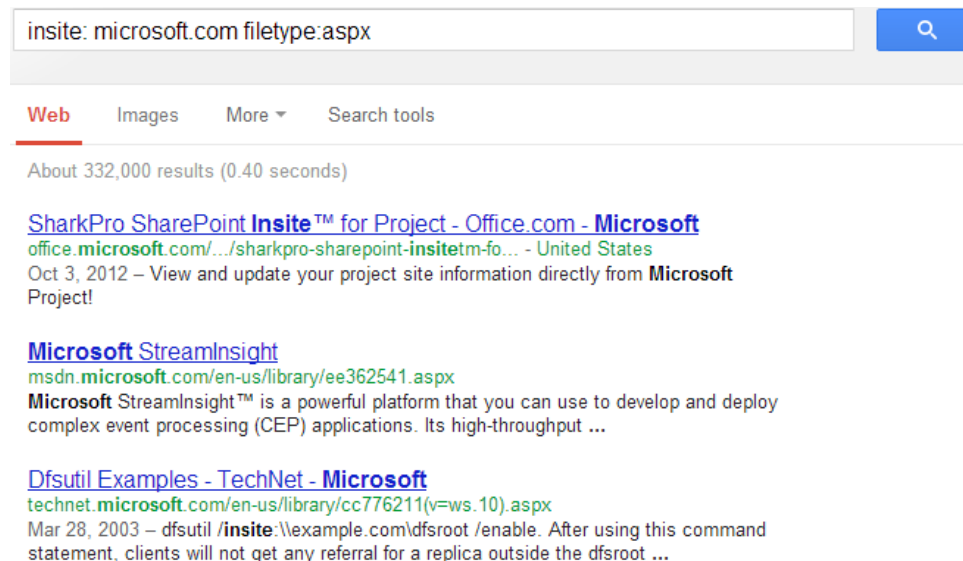
- ▶ Process:
  - ▶ Ignore Event Validation when Forging Event Call
- ▶ Advantage:
  - ▶ Works **despite** of **Viewstate MAC & EventValidation.**



# Mining Cached Viewstate Values 1/2

## ▶ Prerequisites – Reusing Signed Cached Values

- ▶ Obtain control names from cached / indexed content: search engines, proxies, browser cache of privileged users

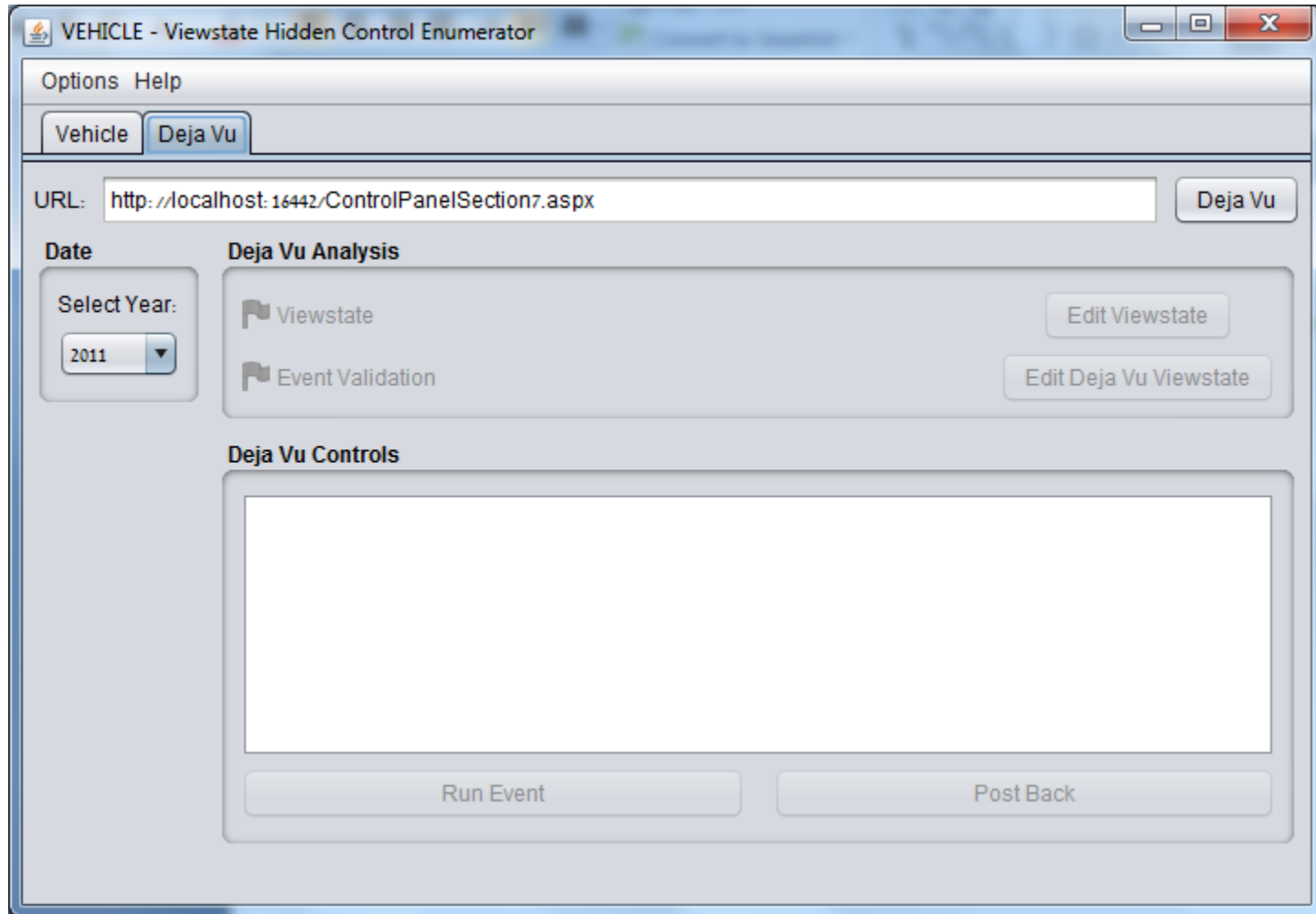


## ▶ Process:

- ▶ Reuse the cached **VIEWSTATE**, **EVENTTARGET**, **EVENTARGUMENT** and **EVENTVALIDATION**



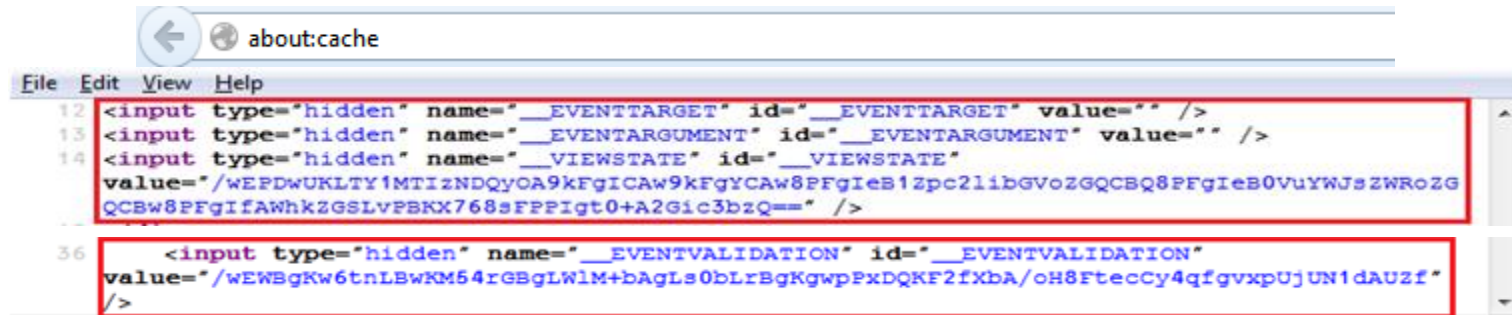
# Mining VS/EV from Cache Sources



# Mining Cached Viewstate Values 2/2

## ► Advantages

- Exploit works DESPITE the Viewstate MAC AND EventValidation



```
12 <input type="hidden" name="__EVENTTARGET" id="__EVENTTARGET" value="" />
13 <input type="hidden" name="__EVENTARGUMENT" id="__EVENTARGUMENT" value="" />
14 <input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwUKLTy1MTIzNDQyOA9kFgICAw9kFgYCAw8PFgIeB1Zpc2libGVozGQCBQ8PFgIeB0VuYWJsZWRoZG
QCBw8PFgIfAWhkZGSLVpBKX768sFPPIgt0+A2Gic3bzQ==" />
36 <input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATION"
value="/wEWBgKw6tnLBwKM54rGBglWlM+bAgLs0bLrBgKgwpFxDQKF2fXbA/oH8FtecCy4qfgvxpUjUN1dAUzf"
/>
```

## ► Shared Hosting Attack Model:

- Can bypass Viewstate MAC and EventValidation
- Scenarios for Shared / Isolated Application Pool





# Risk Mitigation



# Secure Coding Guidelines

---

## Preventing Event Execution & Property Override

- ▶ **Do NOT** use the Disabled property for security purposes
- ▶ **Do NOT** rely on HTML comments to hide controls
- ▶ **Remove** unnecessary dormant events from all layers: HTML, Design (e.g. aspx), CodeBehind (e.g. aspx.cs)
- ▶ **Implement code-level privilege validation** in each event
- ▶ **Enforce** digital signatures (Viewstate MAC)
- ▶ **Activate** event validation mechanisms (EventValidation)
- ▶ **Disable cache / Prevent indexing** in pages with sensitive controls!
- ▶ **Customize** the platform error messages
- ▶ **Replace** the Machine Key (to prevent cache reuse)



# Event Level Privilege Validation

---

## ► Explicit Privilege Validation in Event Code

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (((String)Session["user"]).Equals("admin"))
    {
        ...
    }
}
```

## ► Enable Event Validation / MAC

```
<%@ Page Language="C#" AutoEventWireup="true"
EnableEventValidation="true" EnableViewStateMac="true" ...%>
```



# Disable Cache & Prevent Indexing

---

## ▶ Disable Browser/Proxy Cache (Sample Code)

```
HttpContext.Current.Response.Cache.SetExpires(DateTime.UtcNow.AddDays(-1));  
HttpContext.Current.Response.Cache.SetValidUntilExpires(false);  
HttpContext.Current.Response.Cache.SetRevalidation(HttpCacheRevalidation.AllCaches);  
HttpContext.Current.Response.Cache.SetCacheability(HttpCacheability.NoCache);  
HttpContext.Current.Response.Cache.SetNoStore();
```

## ▶ Restrict SE access in robots.txt (Sample Config)

▶ <http://www.robotstxt.org/robotstxt.html>

```
User-agent: *  
Disallow: /
```

## ▶ Restrict SE caching/crawling via meta tags

▶ <http://www.robotstxt.org/meta.html>

















# SUMMARY

# Dormant Control Execution Summary

---

Control / Event Type	ONLY Event Validation Is ON	ONLY Viewstate MAC is ON	Event Validation AND Viewstate MAC are ON
Commented			
Disabled			
Invisible			
Optional			



# Advanced Invisible Control Execution

---

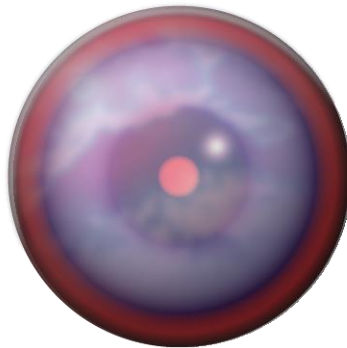
- ▶ **Execute Control Events DESPITE MAC/Validation**
  - ▶ **Reuse cached / indexed ViewState** and EventValidation fields of the same Page
  - ▶ Abuse **insecure callback** implementations
  - ▶ Abuse **MachineKey** in Shared Hosting Model



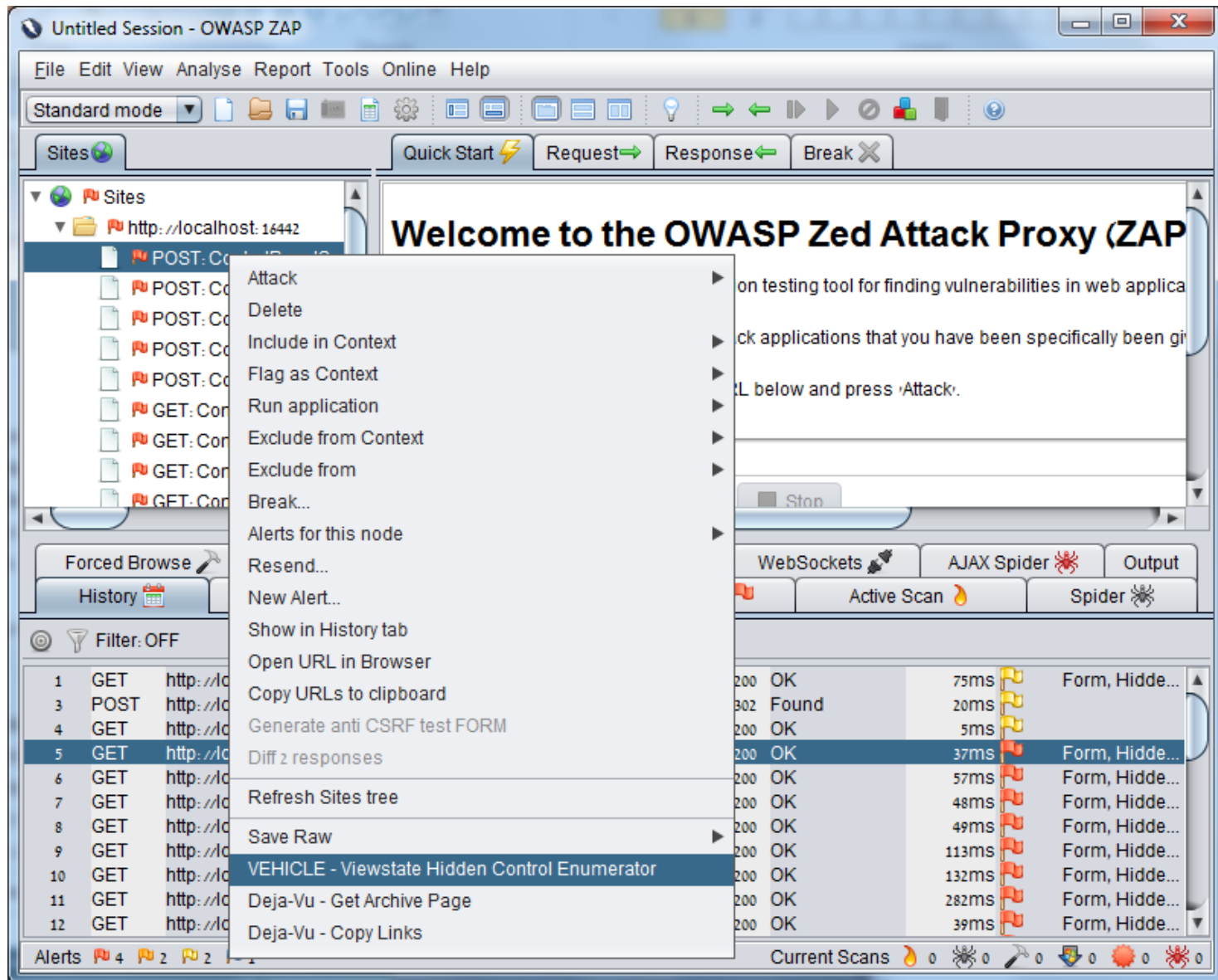
# The VEHICLE Project (a.k.a RIA-SCIP)

---

- ▶ **Homepage:** <http://www.github.com/hacktics/vehicle/>
- ▶ **Alternate:** <http://code.google.com/p/ria-scip/>
- ▶ OWASP ZAP extension (v2.0+), requires Java 1.7
- ▶ Included in ZAP's Marketplace
- ▶ Currently focused at **ASP.net**



# Activating the Vehicle Extension in ZAP



# The Vehicle Project, Cont.

---

## ► Current Features

- Passively identifies traces of Invisible / Dormant Controls
- Error-based invisible control name enumeration
- Blind-based invisible control name enumeration (**NEW!**)
- Disabled/commented control event execution
- Invisible control event execution
- Viewstate manipulation for manual parameter tampering, when MAC is OFF (**NEW!**), **Control Injection Templates**
- Manual execution of target events
- Cached viewstate/eventvalidation scraping - déjà vu (**NEW!**)

## ► Upcoming Features

- Control type fingerprinting & exploitation



# Additional Resources

---

- ▶ **déjà vu (cache analysis) ZAP Extension:**  
<https://github.com/hacktics/deja-vu>
- ▶ **Woanware Viewstate Hacker:**  
<http://www.woanware.co.uk/application/viewstatehacker.html>
- ▶ **OWASP ZAP:** <https://code.google.com/p/zaproxy/>
- ▶ **James Jardine blog posts:**  
<http://www.jardinesoftware.net/>



# EY Advanced Security Centers

- **Asia Pacific**

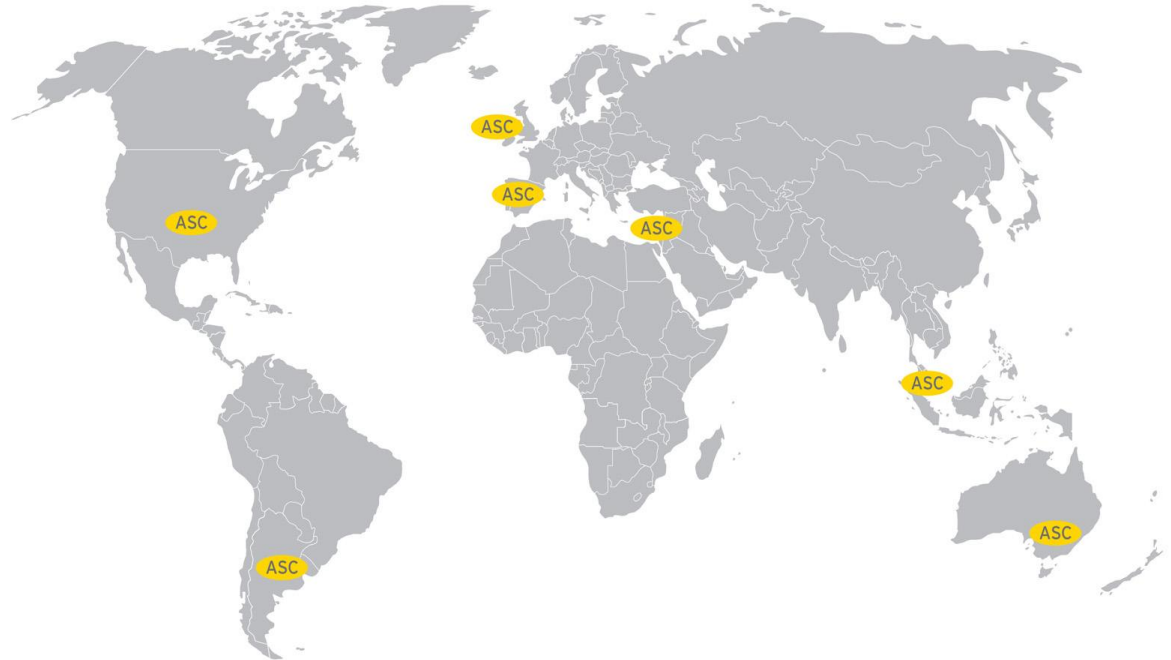
- Melbourne
- Singapore

- **Americas**

- Hacktics IL
- Houston
- New York
- Buenos Aires

- **EMEIA**

- Dublin
- Barcelona





# QUESTIONS?

Shay Chen ([@sectooladdict](#))

Niv Sela ([@nivselatwit](#))

Alex Mor ([@nashcontrol](#))



Building a better  
working world