INVISIBILITY PURGE
Unmasking Dormant Events of Invisible Server Web Controls
Advanced Hacking of ASP.Net, Mono and RIA

Shay Chen
Senior Manager, Hacktics CTO
Hacktics ASC, Ernst & Young

OWASP
The Open Web Application Security Project
About Hacktics

• Formerly a boutique company that provided various 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).
Open Source Projects

OWASP
The Open Web Application Security Project

Current/Legacy Projects

**diviner**
An active information gathering platform

**ria-scip**
An OWASP ZAP extension for enumerating and activating events

**burp-log-reviver**
A solution for converting burp logs into sessions

**sn-crawler**
An intelligence gathering platform focused on social networks.

**payload-manager**
An attack payload management tool.

**puzzlemarket**
A vulnerable web application for practicing session puzzling

*Dynamic AJAX CSRF template*
Dynamic AJAX CSRF - POC Code.

*Session Keep Alive*
A POC tool for connection pool consumption delay of service attack.

Ultimate Obsolete File Detection - ZAP Plugin
SCIP!

Server Control Invisibility Purge

A project based on a research by Niv Sela and Shay Chen, OWASP ZAP extension implementation by Alex Mor.
EodSec

Execution of dormant server events & controls
EodSec
Exploitation Scenarios:

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

• The Attack Surface of RIA Applications
• Server Controls, Events and Lifecycles
• Invisible Web Controls & Dormant Events
• Dormant Event Activation, Control Fuzzing & Event Enumeration
• Control Enumeration / Event Execution via SCIP: Diviner/OWASP ZAP Extension
• Risk Mitigation
• Q & A
The Attack Surface of RIA

Facing the Horde of Security Features
• Event Validation
• Digital Signatures: Limit to List, Manipulation Prevention
• Security Filter (XSS)
• Sandbox
• Built-in Regular Expressions
• Secure Database Access Methods
• Etc
• **Purpose:** Locating Code that can be Abused

  • Web Pages
  • Web Service Methods
  • Global Modules (Filters, Handlers, Etc)
  • ...
  • *Events of Web Application Server Controls*
Server Web Controls

- Rendered into HTML/JS code, but include server side implementation
- Core Controls and Custom Controls (e.g. ascx)
Server Control Events

- A triggered server side code segment, containing optional functionality (PostBack/CallBack in ASP.Net)
- Client triggered events rely on the EVENTTARGET, EVENTARGUMENT and VIEWSTATE mechanisms
- Sample Server Side Implementation (C#, ASP.Net):
  - aspx:
  ```xml
  <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Button" />  
  ```
  - aspx.cs:
  ```csharp
  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";
  }
  }
  ```
Sample client-side implementation (ASP.Net postback):

```html
<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="/wEPdWukiLy1M:
  </div>
  <input type="button" name="Button1" value="View Service Status" onclick="javascript:__doPostBack('Button1',""");">
</form>

<script type="text/javascript">
//<![CDATA[
  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();
    }
  }
//]]>
</script>
```
- Independent Events: buttons with usesubmitbehavior=false, checkboxes, etc
- Sample Event Lifecycle
- Programmatic vs. Declarative
Viewstate Structure

- Serialized into Base64*
- Signed (MAC), clear-text or encrypted
• **Name/Value HashCode Formula**

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

**EventValidation (Viewed via Burp Viewstate Decoder):**

- ViewState v2.0 compatible
  - [MAC is not enabled]
  - List
    - int -1280308489
    - int -1314758625
    - int -1314758624
    - int -1314758619
    - int 2087245738
    - int 2087245739
    - int 2087245736
    - int -1314758618
    - int 2087245737
    - int 2087245736
    - int 2087245739
    - int 0

- Viewstate Hashcode
- Control Hashcodes

**Event Validation Mechanism:***

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

```html
<input type="hidden" name="__EVENTVALIDATION" value="/wEWCAK+2oTRBwLW1M+bAgls0bLrBgKgwPxDqKF2f:/16KxF3ZQYvULqQ0yedG1oA7" />
```
Visible / Enabled Controls:

Control Panel - Zone 1
- View Service Status
- Shutdown Service

Request

EventValidation (Viewstate Decoder):

```
ViewState v2.0 compatible {MAC enabled}

List
- Viewstate
  - Shutdown

Evidence of Hidden Controls
```

Invisible / Disabled Controls (Control Trace in Viewstate!):

Control Panel - Zone 1
- View Service Status

Request

EventValidation (Viewstate Decoder):

```
ViewState v2.0 compatible {MAC enabled}

List
- Viewstate
- Shutdown

Evidence of Hidden Controls
```

Visible / Enabled Controls:

Control Panel - Zone 1
- View Service Status
- Shutdown Service

Request

EventValidation (Viewstate Decoder):

```
ViewState v2.0 compatible {MAC enabled}

List
- Viewstate
- Shutdown

Evidence of Hidden Controls
```
Invisible Web Controls: Archetypes
• **Commented Out Controls**
  - The control is commented out using HTML comments
  - **Rendered inside an HTML comment**, but the server code is still active.

```xml
<!-- <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>");
```
• **Disabled Controls**
  - The control `enabled` property is set to `false`
  - Rendered with the `disabled="disabled"` HTML property
  - Rendered **without** an input `postback` method

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

![Control Panel - Zone 1](Image)
• **Invisible Controls**
  - The control `visible` property is set to `false`
  - **Not Rendered** in the presentation layer, but the code is still active

```csharp
Button2.Visible = false;
```
• **Dormant Events of Visible Controls**
  – Optional event listeners registered in the code level, after the optional definition was deleted from a control with at least one active event.
Basic Dormant Event Activation: Commented Controls

```csharp
if (newClass == null)
    throw new ArgumentException("Cannot add a null css class");

//string existingClasses = control.CssClass;
//if (String.IsNullOrEmpty(existingClasses))
//{
//    control.CssClass = newClass;
//    return;
//}

//string[] classes = existingClasses.Split(' ');
foreach (string attributeValue in classes)
{
    //...
}
```
Events of Commented Controls

• **Prerequisites (ASP.Net / Mono) - Commented Out Controls:**
  - The developer should rely solely on the fact that the control is commented.
  - The attacker can simply “uncomment” the HTML control and execute the embedded event, or send the appropriate values directly.

• **Advantages**
  - Exploit works **even** if the **ViewState MAC** AND the **EventValidation** features are **turned ON**.
POST http://localhost:7011/WelcomeMirror.aspx HTTP/1.1
Host: localhost:7011
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:18.0) Gecko/20100101 Firefox/18.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Cookie: ASP.NET_SessionId=egfhhxrr1kpqexzebuhn55
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded

_EVENTTARGET=6\_EVENTARGUMENT=6\_EVENTVALIDATION=
$2FWEBcRw6CHLSXK854rGqLW1M%25aQgL3ObLrBqKwDpFDQKMyF2lXb0a32EdO%687EecCy4qFqVxpUJUN1dAJL%64
_VIEWSTATE=
$2FwEPwUKLY1MT%7D%0A%9kFgICAw9kFgYCAw8PFgIeB17Zpc21lGbGVoZGQCBQ8PFgIeB0V6YWMeZWRvZGQCBw8PFgIfAW
h%2GSLvPBKX7569sFPPFgt0%2BA2Gic3bzO%3D%3DxButton4=Button
Intermediate Dormant Event Activation:

Disabled Controls

### Web Service Extensions

<table>
<thead>
<tr>
<th>Web Service Extension</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Unknown CGI Extensions</td>
<td>Prohibited</td>
</tr>
<tr>
<td>All Unknown ISAPI Extensions</td>
<td>Prohibited</td>
</tr>
<tr>
<td>Active Server Pages</td>
<td>Prohibited</td>
</tr>
<tr>
<td>ASP.NET v2.0.50727</td>
<td>Allowed</td>
</tr>
<tr>
<td>Internet Data Connector</td>
<td>Prohibited</td>
</tr>
<tr>
<td>Server Side Includes</td>
<td>Prohibited</td>
</tr>
<tr>
<td>WebDAV</td>
<td>Prohibited</td>
</tr>
</tbody>
</table>
• **Prerequisites (ASP.Net / Mono) - Disabled Controls:**
  - The developer should rely solely on the control disability and the lack of JS postback/callback method for protecting the control events.
  - The attacker should forge a postback / callback method, or send the appropriate values directly.

• **Advantages**
  • Exploit works **even** if the [Viewstate MAC](#) AND the [EventValidation](#) features are **turned ON**.
The Process of Forging a PostBack / CallBack Method

Why does it work?
- Using temporarily disabled controls in ASP.Net is a feature
- Controls might be disabled without any relation to security, and thus, are currently not protected like invisible controls

How does it work?
- The control name is exposed in the disabled control
- The attacker can use an interception proxy to “inject” postback calls into HTML control events, or craft requests manually by reusing the existing viewstate/validation fields.
Events of Disabled Controls
Advanced Dormant Event Activation: Invisible Controls!
Prerequisites (ASP.Net / Mono) - Invisible Controls:

(I) Either the Viewstate MAC OR the EventValidation features must be turned off.

(II) The developer should rely solely on the control invisibility for protecting the invisible control events.

```csharp
<system.web>
    <pages enableEventValidation="false"/>
</system.web>

<system.web>
    <pages enableViewStateMac="false"/>
</system.web>
```
• **EventValidation is ON but the Viewstate MAC is OFF**
  
  • In order for the attack to succeed, we need to forge a valid eventvalidation structure (no MAC)
    
    • Craft a request using SCIP or other viewstate/eventtarget editors

```csharp
<%@ Page Language="C#" AutoEventWireup="true"
EnableEventValidation="true" EnableViewStateMac="false" ...%>
```
• **EventValidation is OFF**
  • Since there's no event validation, any event can be executed, regardless of the viewstate value
    • 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

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

• In all cases, we still need to obtain the control / event name...
The Process of Server Control Enumeration
- In this scenario, the control leaves no client-side traces:
  - Control Name Fuzzing
  - Core Controls vs. Custom Controls

Control Event Enumeration
- Core Events vs. Custom Events
- Dormant Events vs. Active Events
Common 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
Error-Based Enumeration

- Accessing invalid control names will NOT raise exceptions
- Accessing protected will – only works if 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.ArgumentNullException: 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:

• 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;
Events of Invisible Controls

OWASP
The Open Web Application Security Project

SCIP - RIA Event Enumerator

Options: Help

URL: http://localhost:7011/WelcomeChanged.aspx

ViewState

- ViewState
- ViewState Signed MAC found
- ViewState Encrypted
- Event Validation

Page Controls

Visible:
- Buttons
  - Button1
  - Button2
  - Button3
  - Button4

Enumeration Results:

<table>
<thead>
<tr>
<th>Control Name</th>
<th>URL</th>
<th>Hidden</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>onclick</td>
</tr>
</tbody>
</table>

Control Enumeration

URL: http://localhost:7011/WelcomeChanged.aspx

Prefix: 

Run

Trying: Text
Trying: Text
Trying: Text
Trying: Text
Trying: Text
Trying: Text
Trying: Text
Trying: Text
Trying: Text
Trying: Text

Controls Found: 3

Apply

Run Event
Events of Invisible Controls
Master Dormant Event Activation: Locating Hidden Optional Events
• **Prerequisites – Multiple Dormant Events of a Single Control:**
  
  – By default, only a limited amount of basic controls support multiple events (not including custom controls).
  
  – The hidden control must be assigned with multiple valid events (example: Calendar control).
  
  – In addition to fuzzing a valid event target, the tester can execute the “optional” events by locating/fuzzing a valid event argument.
  
  – Different event argument formats can execute different server events (for example V[value] vs. [value])

• **Advanced:** Core Events and Custom Events
  
  • Click, Command, onSelectionChanged, OnVisibleMonthChanged, Etc
Activating Optional Events

```csharp
<asp:Calendar ID="Calendar1" runat="server"
    onselectionchanged="Calendar1_SelectionChanged"
    OnVisibleMonthChanged="Secret_Click">
</asp:Calendar>

protected void Secret_Click(object sender, MonthChangedEventArgs e)
{
    Label1.Text = "<b>Secret!!!</b>";
}

protected void Calendar1_SelectionChanged(object sender, EventArgs e)
{
    Label1.Text = "<b>Normal</b>";
}
```
SCIP Methods for Gurus:
Executing Events of Invisible Controls
DESPITE
Active Event Validation & Viewstate MAC
• **Prerequisites - Execute Events In Spite of Security Features:**
  - Obtain the names of server controls from cached / indexed content: (search engines, browser cache of another high privileged user, etc)
  - Reuse the **cached** VIEWSTATE, EVENTTARGET, EVENTARGUMENT and EVENTVALIDATION to executing dormant events (will work regardless of visibility or security features!)

![Search Results](image-url)
• **Reusing Obsolete Cached / Indexed State Flags**
  - Reusing the state and validation of indexed/cached versions page might work even if the control structure *changed* (!)
  - Controls, State and validation flag must origin from the same page (so the signature will be effective)
  - Controls must be included/include the controls of the page

• **Signed Content Scraping Using Web Attacks**
  - XSS, Clickjacking, Etc
• Shared Hosting Attack Model
  • Can bypass Viewstate MAC and EventValidation
  • Scenarios for Shared Application Pool
  • Scenarios for Isolated Application Pool
Risk Mitigation
Protecting Dormant Events
• 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
• Explicit Privilege Validation in Event Code

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

• Enable Event Validation / MAC

```csharp
<%@ Page Language="C#" AutoEventWireup="true"
EnableEventValidation="true" EnableViewStateMac="true" ...%>
```
• Disable Browser/Proxy Cache (Sample Code)

```csharp
HttpContext.Current.Response.Cache.SetValidUntilExpires(false);
```

• Restrict SE access in robots.txt (Sample Config)
  • http://www.robotstxt.org/robotstxt.html

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

• Restrict SE caching/crawling via meta tags
  • http://www.robotstxt.org/meta.html
The Original Theory
Research Leads and Progress
• Reuse the viewstate / eventvalidation fields of other pages
  • Pages with similar controls
  • Pages with identical controls
• EventValidation responding differently to manipulations on various control types
• Reuse a partial or included cached viewstate / eventvalidation fields
• Different behaviors for different ASP.Net versions (v1.1, v2.0, v3.5, v4.0...) and Mono versions
Summary

Enumerating Hidden Controls and Events
• **Diviner**
  • OWASP ZAP extension (v1.4+/v2.0+)
  • Requires ZAP to run with Java 1.7+
  • Homepage: [http://code.google.com/p/diviner/](http://code.google.com/p/diviner/)

• **SCIP**
  • OWASP ZAP extension (v2.0+), currently focused at **ASP.net**
  • **Features:** disabled/commented control event execution, error-based detection of invisible controls, manual execution of target events, parameter tampering in-spite of event validation (when MAC is off)
  • **Upcoming features:** cache scraping and analysis, reuse obsolete event-validation fields, blind event enumeration
  • Requires **Diviner** diff methods to support Blind Control Enumeration
Potential Dormant Events:
- Events of Disabled Controls (ASP.Net: .enabled=false)
- Events of Invisible Controls (ASP.Net: .visible=false)
- Events of HTML Commented Controls (aspx: <!-- ... -->)
- Hidden Alternate Events of Core/Custom Controls
• **Prerequisites for Event Execution Methods:**
  • Events of Disabled /Commented Controls - None!
  • Events of Invisible Controls - the EventValidation OR Viewstate MAC must be turned off; can occur per machine, application, page or control
  • Hidden Alternate Events of Core/Custom Controls

• **Advanced Event Execution Methods:**
  • Execute any control event, regardless of viewstate MAC or event validation, by reusing cached values of viewstate, eventtarget, eventargument and eventvalidation fields
  • State fields must include the control’s digitally signed content
And Finally...
• SCIP Homepage (ZAP 2.0+ Extension)
  • http://code.google.com/p/ria-scip/
• Diviner Homepage (ZAP 1.4+/2.0+ Extension)
  • http://code.google.com/p/diviner/
• OWASP ZAP Proxy
  • http://code.google.com/p/zaproxy/
• Great posts on the subject by James Jardine
  • http://www.jardinesoftware.net/
Ernst & Young ASCs

- **Americas**
  - Hacktics IL
  - Houston
  - New York
  - Buenos Aires
- **EMEIA**
  - Dublin
  - Barcelona
- **Asia Pacific**
  - Singapore
  - Melbourne
Ernst & Young is a global leader in assurance, tax, transaction and advisory services. Worldwide, our 130,000 people are united by our shared values and an unwavering commitment to quality. We make a difference by helping our people, our clients and our wider communities achieve potential.

About Ernst & Young’s Technology Risk and Security Services
Information technology is one of the key enablers for modern organizations to compete. It gives the opportunity to get closer, more focused and faster in responding to customers, and can redefine both the effectiveness and efficiency of operations. But as opportunity grows, so does risk. Effective information technology risk management helps you to improve the competitive advantage of your information technology operations, to make these operations more cost efficient and to manage down the risks related to running your systems. Our 6,000 information technology risk professionals draw on extensive personal experience to give you fresh perspectives and open, objective advice – wherever you are in the world. We work with you to develop an integrated, holistic approach to your information technology risk or to deal with a specific risk and security issue. And because we understand that, to achieve your potential, you need a tailored service as much as consistent methodologies, we work to give you the benefit of our broad sector experience, our deep subject matter knowledge and the latest insights from our work worldwide. It’s how Ernst & Young makes a difference.

For more information, please visit www.ey.com.

© 2012 EYGM Limited. All Rights Reserved.
Proprietary and confidential. Do not distribute without written permission.

Ernst & Young refers to the global organization of member firms of Ernst & Young Global Limited, each of which is a separate legal entity. Ernst & Young Global Limited, a UK company limited by guarantee, does not provide services to clients.
QUESTIONS?

Shay Chen (https://twitter.com/sectooladdict)
Niv Sela (https://twitter.com/nivselatwit)
Alex Mor (https://twitter.com/nashcontrol)