ASP.NET and ViewState Security

April 12nd, 2011

alexandre.herzog@csnc.ch
thomas.roethlisberger@csnc.ch

Compass Security AG
Werkstrasse 20
Postfach 2038
CH-8645 Jona

Tel +41 55-214 41 60
Fax +41 55-214 41 61
team@csnc.ch
www.csnc.ch

Agenda

Headlines and ViewState Intro

ViewState Flaw

How to Protect (?)

- Input Validation / Request Validation
- Output Encoding
- How to really avoid ViewState Tampering

Conclusion

Headlines

Multiplatform View State Tampering

From: Trustwave Advisories

Sent: Tuesday, February 9th 2010 23:41

...SpiderLabs has documented view state tampering vulnerabilities ... View states are used by some web application frameworks to store the state of HTML GUI controls. View states are typically stored in hidden client-side input fields, although server-side storage is widely supported.

Credit: David Byrne of Trustwave's SpiderLabs

Headlines

Microsoft | TechNet

MS10-070 Vulnerability in ASP.NET could allow information disclosure

Executive Summary

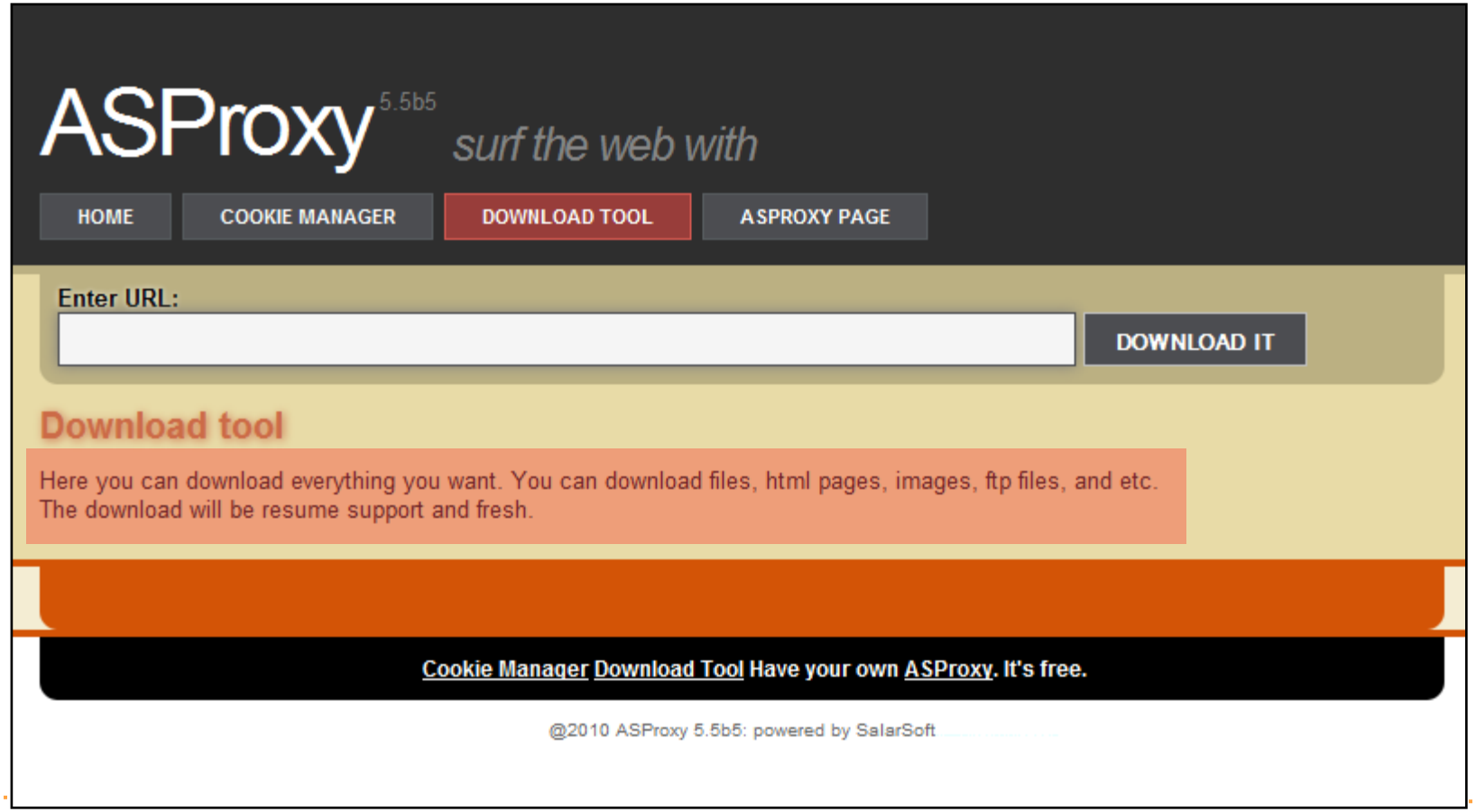
... An attacker who successfully exploited this vulnerability could read data, such as the view state, which was encrypted by the server. This vulnerability can also be used for data tampering, which, if successfully exploited, could be used to decrypt and tamper with the data encrypted by the server.

Microsoft .NET Framework versions prior to Microsoft .NET Framework 3.5 Service Pack 1 are not affected by the file content disclosure portion of this vulnerability.

ViewState Intro

Sample Application in Hacking-Lab

- http://asproxy.hacking-lab.com example view state app



ViewState Intro

Sample Code Snippet in C#

```
<script runat="server">
    protected void Page_Load(object sender, Event...
        if (!IsPostBack) {
            myLabel.Text = "Here you can download...
        }
    }
</script>

<asp:Content runat="server" ContentPlaceHolderID...
    <asp:Label ID="myLabel" runat="server">
    </asp:Label>
```

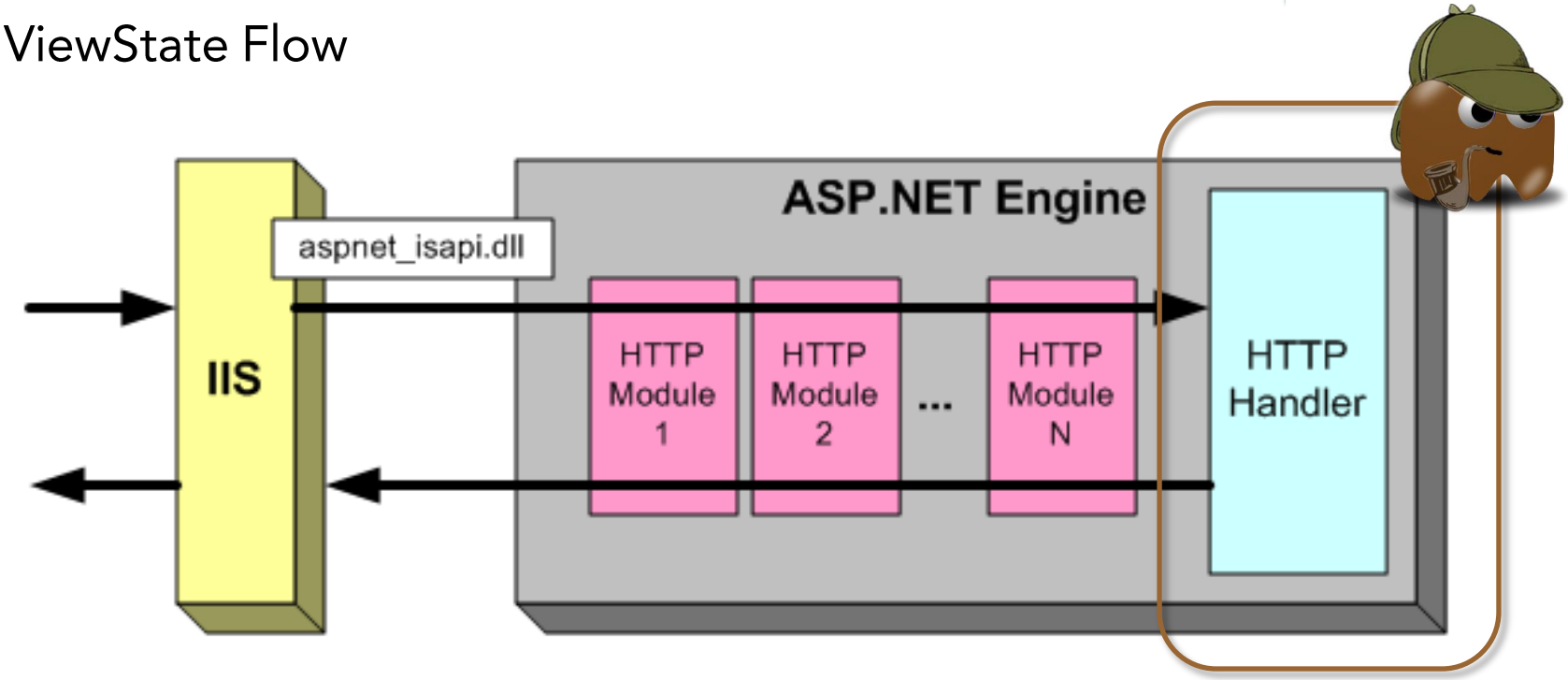
ViewState Intro

Sample HTML Snippet

```
<form name="aspnetForm" method="post" id="asp...
  <input type="hidden" name="__VIEWSTATE" id="__V...
         value="/wEP0aWpA45OkQLP9+4sT2...YW1lcw=" />
   ...
            Download tool</span></h1>
   </div>
   ...
   <div class="entry">
      <span id="ctl00_plhContent_myLabel">
         Here you can download everything you wan...
      </span>
```

ViewState Intro

ViewState Flow

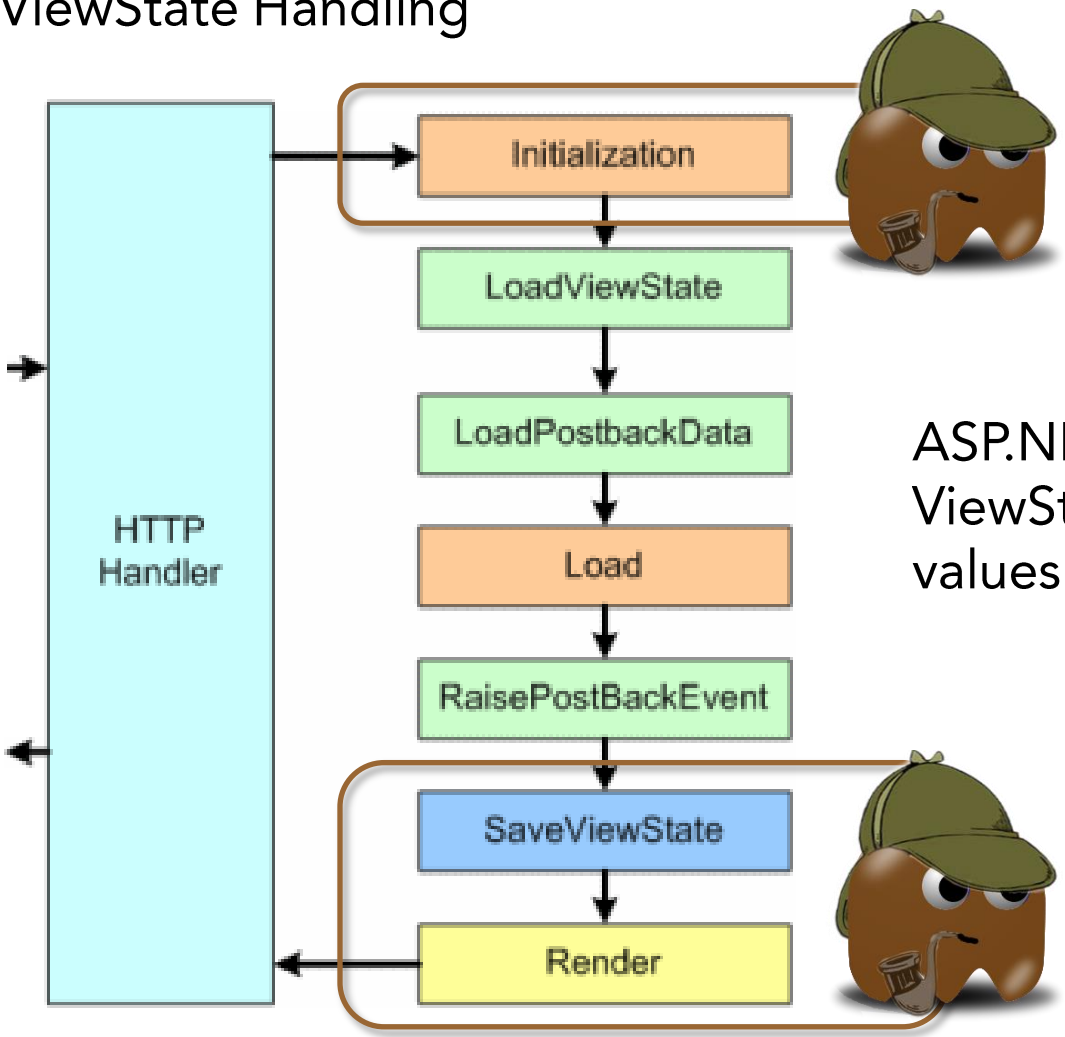



```
...
<input type="hidden" name="__VIEWSTATE" id="__V...
        value="/wEP0aWpA45OkQLP9+4sT2...YW1lcw=" />
```

The field __VIEWSTATE contains value of myLabel in encoded form

ViewState Intro

ViewState Handling



Deserialize

ASP.NET deserializes the ViewState and copies the values back into the controls

Serialize

ViewState Intro

A ViewState

- Is Base64 encoded
- Can be encrypted
- Can be split into blocks of x bytes (__VIEWSTATEFIELDCOUNT & __VIEWSTATEx fields)
- Is tamper proof by default (HMAC-SHA1)
- Can include user defined values to ensure a unique MAC is generated (Page.ViewStateUserKey property)

A ViewState Contains

- 2 bytes of header (ASP.NET 1.1 versus 2.0+)
- A tree of serialized objects
  - Viewstate Bag
  - Serialized ASP.NET controls of the page
- A MAC (if configured so)

ViewState Intro

ViewState Facts

- Passive controls (eg. Labels) are not rendered as HTML input fields
- Passive controls need their value to be posted back to the server
- Disabling the ViewState will complicate the way to allow a user to work with multiple windows
- Disabling the ViewState will destroy the advantages of the framework
- Disabling the ViewState will result in overhead during development

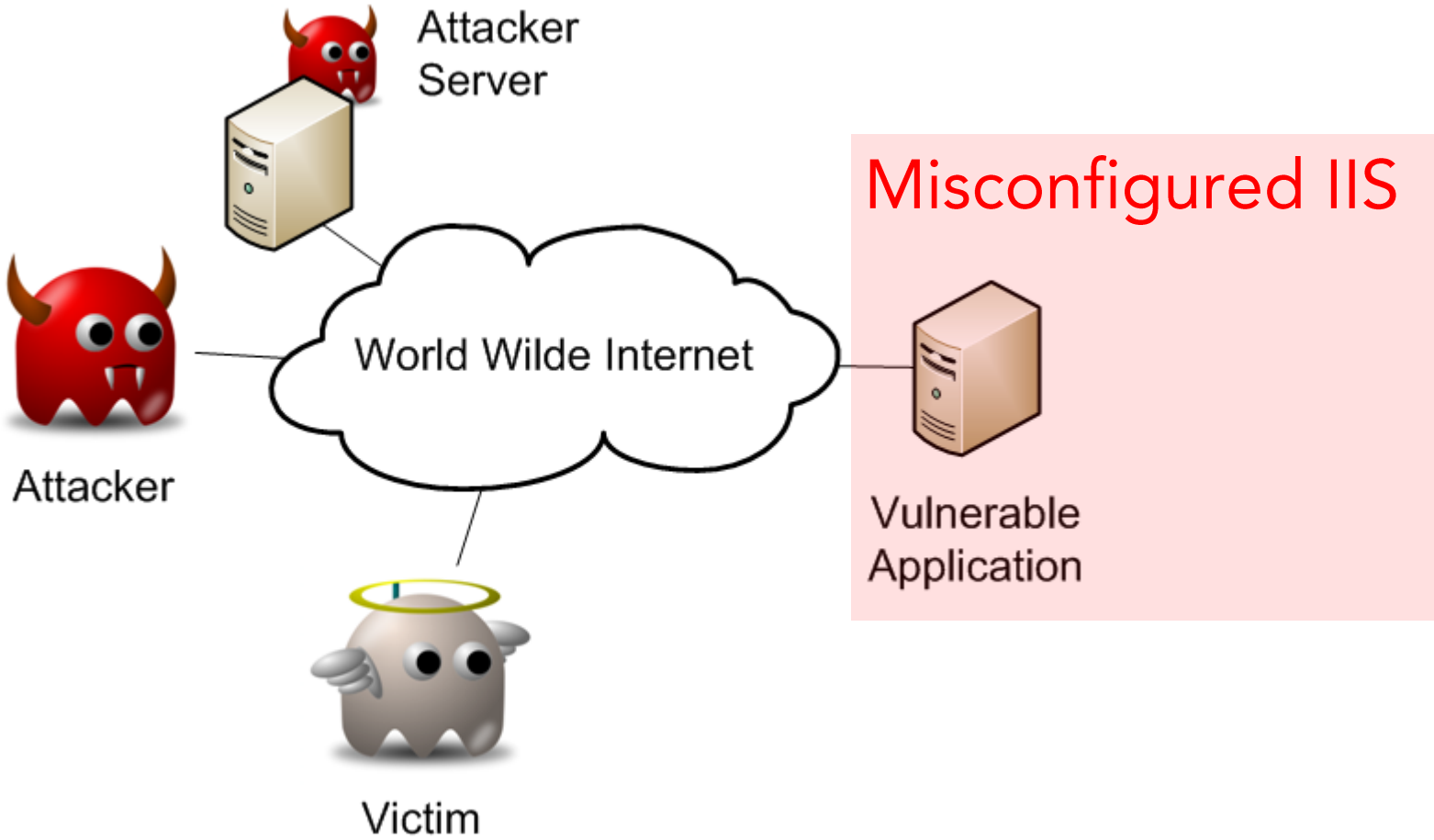
ViewState Flaw

Compass Security AG
Werkstrasse 20
Postfach 2038
CH-8645 Jona

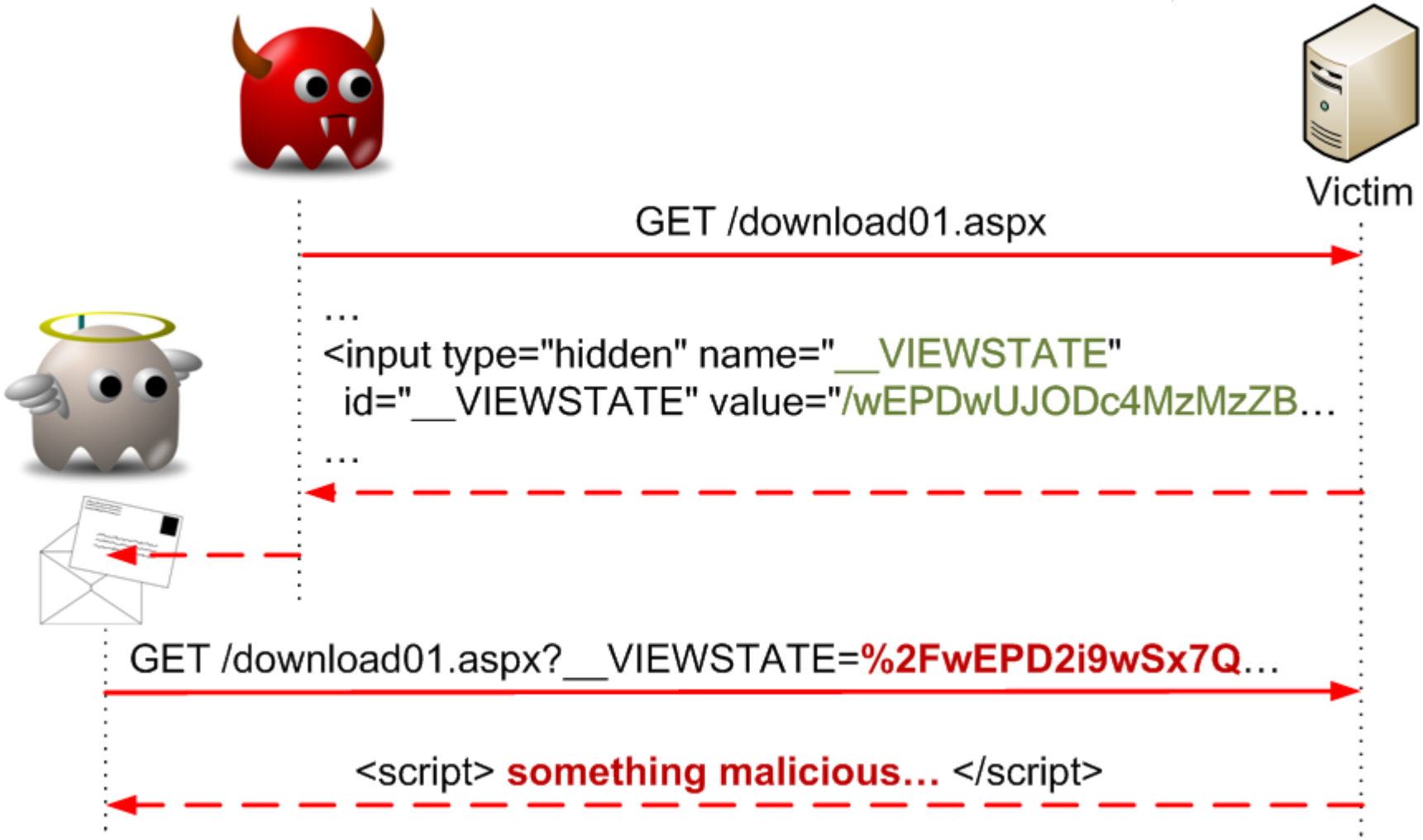
Tel +41 55-214 41 60
Fax +41 55-214 41 61
team@csnc.ch
www.csnc.ch

ViewState Flaw

Example of an Architecture



ViewState Flaw - Sequence



An attacker can inject malicious HTML or JavaScript into the ViewState

JavaScript can be injected by tampering the value of the hidden field `__VIEWSTATE`. The server copies the tampered values back into the server controls. As ASP.NET renders the resulting HTML response the malicious code is included in the page.

Identify a property of a control inside of the ViewState which makes sense to modify (Several properties are vulnerable to such an attack: E.g. `Text` of a label, `InnerHTML` of the main form, etc.)

ViewState Flaw - Text Injection

```
<System.Collections.ArrayList>
  <System.Int32>1</System.Int32>
  <System.Web.UI.Pair>
    <System.Web.UI.Pair>
      <System.Collections.ArrayList>
        <System.Web.UI.IndexedString>Text</System.Web.UI.IndexedString>
        <System.String xml:space="preserve">Hello &lt;script&gt;alert('xss')&lt;/script&gt;X
      </System.Collections.ArrayList>
      <Null>True</Null>
    </System.Web.UI.Pair>
    <Null>True</Null>
  </System.Web.UI.Pair>
  <System.Int32>3</System.Int32>
  <System.Web.UI.Pair>
    <System.Web.UI.Pair>
      <System.Collections.ArrayList>
        <System.Web.UI.IndexedString>Text</System.Web.UI.IndexedString>
```

Reserialized and Base64 encoded

/wEPDwUKMTI2NTY4ODI3MQ9kFgICAw9kFgQCAQ8PFgIeBFRleHQFJkhlbGxvIDxzY3JpcHQ

```
      </System.Collections.ArrayList>
```

ViewState Flaw - InnerHTML Injection

```
<System.Collections.ArrayList>
  <System.Int32>3</System.Int32>
  <System.Web.UI.Pair>
    <System.Collections.ArrayList>
      <System.Web.UI.IndexedString>innerhtml</System.Web.UI.IndexedStrin
      <System.String xml:space="preserve">Hello &lt;script&gt;alert('xss')&lt;/sc
    </System.Collections.ArrayList>
    <Null>True</Null>
  </System.Web.UI.Pair>
</System.Collections.ArrayList>
</System.Web.UI.Pair>
</System.Web.UI.Pair>
<Null>True</Null>
```

Reserialized and Base64 encoded

/wEPDwUKMTI2NTY4ODI3MQ9kFgICAw9kFgQCAQ8PFgIeBFRlcHQFJkhlbGxvIDxzY3JpcHQ

How to protect?

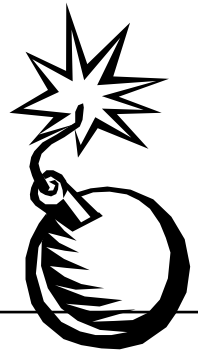
Compass Security AG
Werkstrasse 20
Postfach 2038
CH-8645 Jona

Tel +41 55-214 41 60
Fax +41 55-214 41 61
team@csnc.ch
www.csnc.ch

Input Validation

ASP.NET Request Validation

- Validation against query-string and form variables as well as cookie values
- ASP.NET raises an error if a request contains HTML-encoded elements or certain HTML characters (e.g. <script>)
- Does not validate contents of the ViewState



Search: test'><script>alert(1)</sc Submit Query

Server Error in '/XSSViewState' Application.

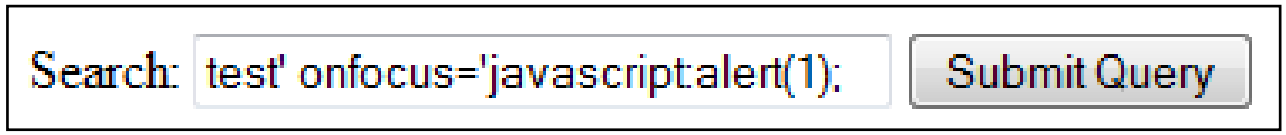
A potentially dangerous Request.QueryString value was detected from the client (search="test'><script>alert(1)</sc...").

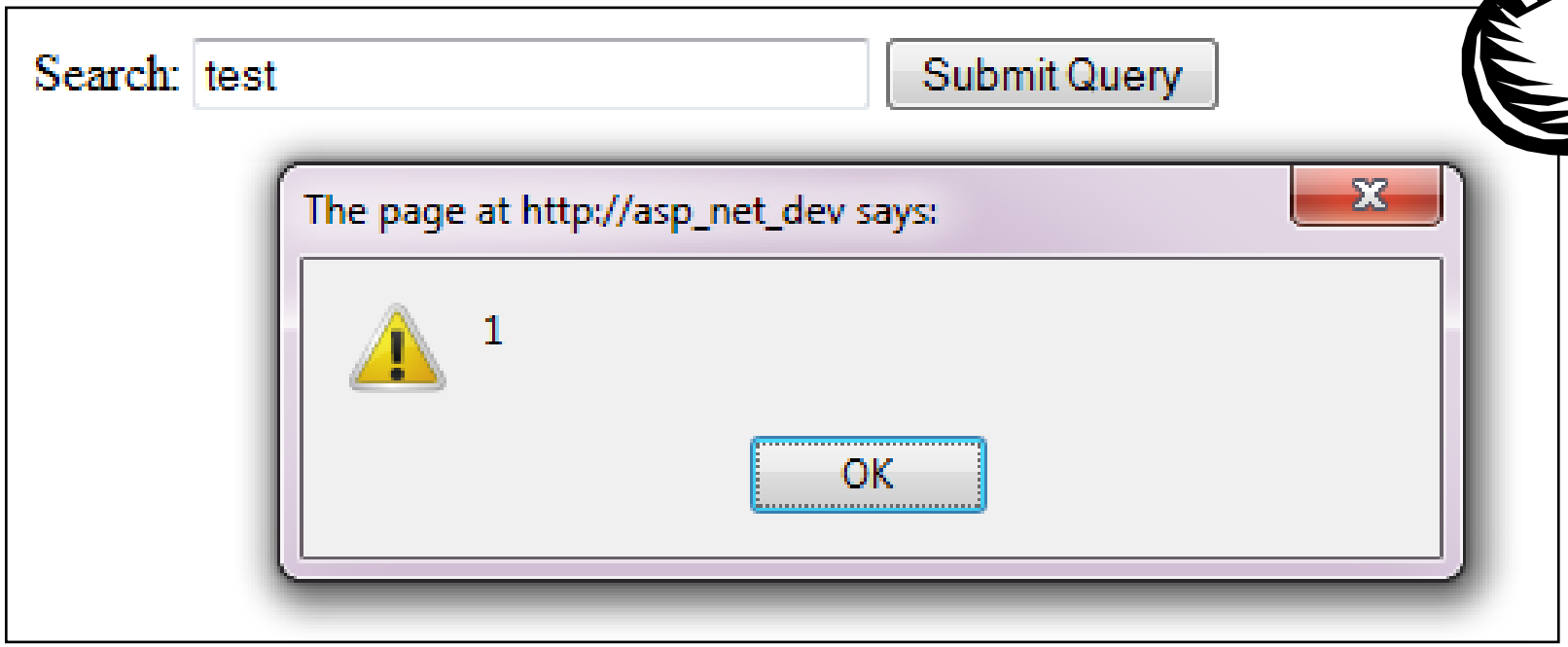
Description: Request Validation has detected a potentially dangerous client input value, and processing of the request

Input Validation

ASP.NET Request Validation Dangers

- Feature often disabled by developers as quick fix for this "strange errors".
- It doesn't filter all dangerous characters (E.g. ', " or &)





Input Validation

Exploitable Code

```
<form id="Form1" method="GET" runAt="server,...
   <label for="inpSearch">Search: </label>
   <input value='<%=Request.QueryString["search"]%>'
          type='text' id='search' name='search'>
   <input type="submit" />
</form>
```

Applications that embbed user input in JavaScript are vulnerable as well

System.Web.CrossSiteScriptingValidation

```
private static char[] startChars = new char[] { '<', '&' };
internal static bool IsDangerousString(…) {
…
char ch = s[num2];
if (ch != '&') {
 if ((ch == '<') && ((IsAtoZ(s[num2 + 1]) ||
     (s[num2 + 1] == '!')) || ((s[num2 + 1] == '/')
     || (s[num2 + 1] == '?'))))
            return true;
}
else if (s[num2 + 1] == '#')
 return true;
```

Looks like Request Validation is not the solution…

Compass Security AG
Werkstrasse 20
Postfach 2038
CH-8645 Jona

Tel +41 55-214 41 60
Fax +41 55-214 41 61
team@csnc.ch
www.csnc.ch

Output Encoding

Compass Security AG
Werkstrasse 20
Postfach 2038
CH-8645 Jona

Tel +41 55-214 41 60
Fax +41 55-214 41 61
team@csnc.ch
www.csnc.ch

Output Encoding

ASP.NET HTML Encoding

- Some of the controls encode their content automatically:
  - Button.Text
  - TextBox.Text
  - ...
- But the rest has to be encoded manually:
  - Label.Text
  - Literal.Text
  - ...

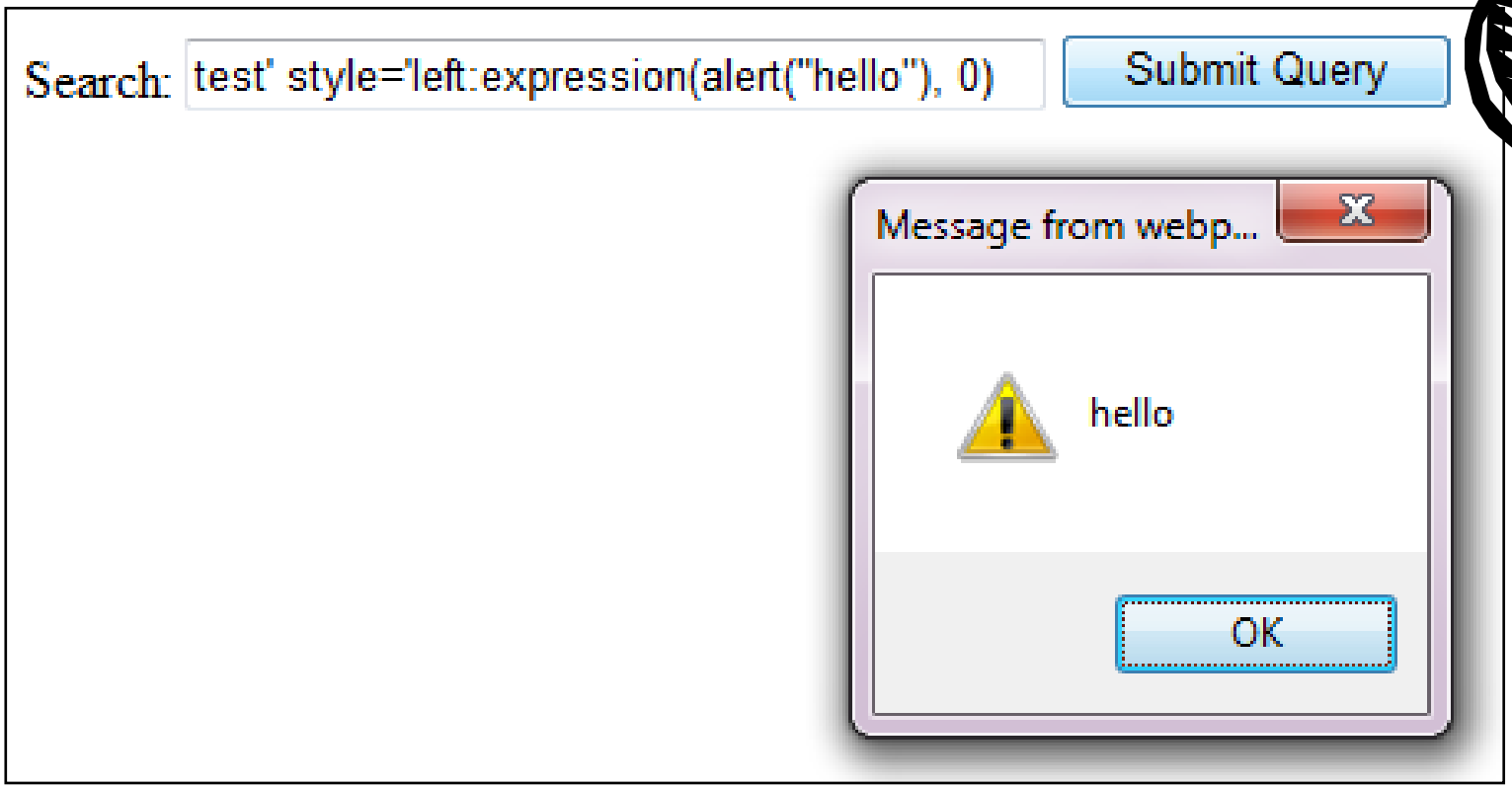
- HtmlEncode converts critical characters using HTML entities

```
Server.HtmlEncode("<b>") => &lt;b&gt;
```

Output Encoding

HTML Encoding Dangers

- Developers often forget to use this method.
- HtmlEncode doesn't encode all dangerous characters (E.g. ', ")



Output Encoding

Exploitable Code

```
<form id="Form1" method="GET" runAt="server...
   <label for="inpSearch">Search: </label>
   <input value='<%=Server.HtmlEncode(
           Request.QueryString["search"]) %>'
           type='text' id='search' name='search'>
   <input type="submit" />
</form>
```

Especially text embedded in JavaScript or JSON fragments will be prone to Cross-site Scripting

Output Encoding

```
// Now in System.Net.WebUtility with .NET 4.0
public static unsafe void HtmlEncode(…) {
...
     switch (ch) {
          case '&': {
               output.Write("&amp;");
               continue;
          }
          case '\'': {
               output.Write("&#39;");
               continue;
          }
          case '"': …
          case '<': …
          case '>': …
…}
```

Okay, how do we really fix it?

Compass Security AG
Werkstrasse 20
Postfach 2038
CH-8645 Jona

Tel +41 55-214 41 60
Fax +41 55-214 41 61
team@csnc.ch
www.csnc.ch

Avoid ViewState Tampering

Compass Security AG
Werkstrasse 20
Postfach 2038
CH-8645 Jona

Tel +41 55-214 41 60
Fax +41 55-214 41 61
team@csnc.ch
www.csnc.ch

Avoid ViewState Tampering

Approaches

- Do not store the ViewState within the HTML page => Implement Handler
- Protect the ViewState contents => Configuration

Avoid ViewState in HTML

- ViewState can be stored on the server
- Override the framework methods
  - SavePageStateToPersistenceMedium()
  - LoadPageStateFromPersistenceMedium()
- Approach leads to challenges
  - Users may work with multiple windows => multiple ViewStates
  - ViewState must be unique for each user and for each page
  - ViewState has to be cleaned up when no longer needed
  - Ensure users cannot access each others ViewStates

Avoid ViewState Tampering

ViewState Protection

- **MAC integrity check**
  The hash of the ViewState data is signed with a key and stored along with the data in the ViewState field. The MAC key might be configured to be recreated on startup, which will cause issues in balaced environments. Moreover, applications could be isolated.
- **Event Validation**
  For each control on the page a unique number is generated (XOR of the hashes of all valid values and the UniqueId of the control). All these numbers are stored in another hidden field on the page called __EVENTVALIDATION.
- **ViewStateUserKey**
  If provided a user session specific id is used as a salt in the MAC. This avoids the ViewState being used by a different user.
- **Encryption**
  The data of the ViewState is encrypted.

- Lockdown the ViewState configuration within the machine.config.

So what?

Compass Security AG
Werkstrasse 20
Postfach 2038
CH-8645 Jona

Tel +41 55-214 41 60
Fax +41 55-214 41 61
team@csnc.ch
www.csnc.ch

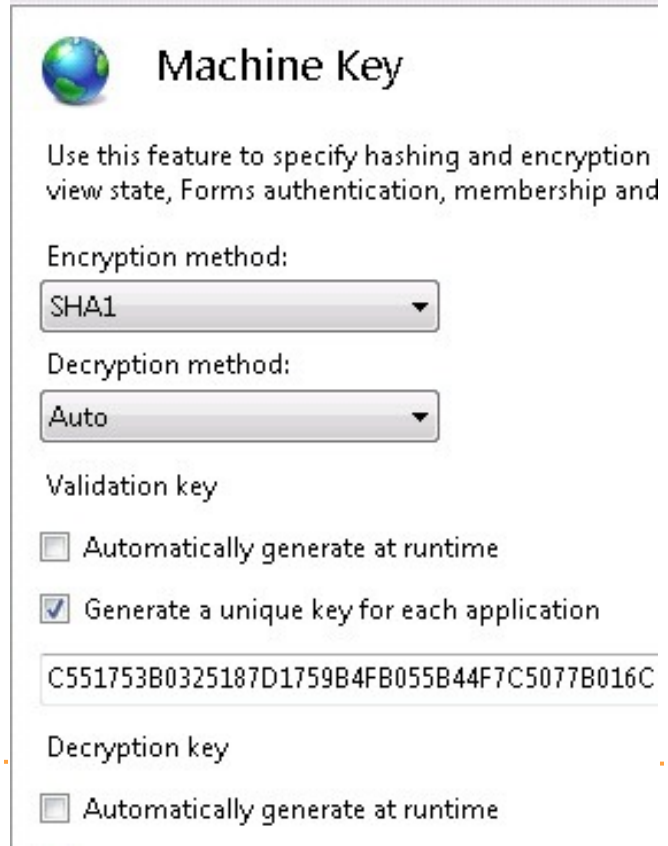
So what?

Am I vulnerable?

- Not by default

Are there variantes of this attack?

- Yes. Where do you get your machine keys from?




```
<!-- web.config file of DotNetNuke
      latest version -->
<system.web>
 <machineKey
   validationKey="F60E6580AE5E29E10C
     F592A687E87F1D09280611"
   decryptionKey="8A3D693693DB497480
     7AC0078A2564C1ED8A19121BCB342C"
   decryption="3DES"
   validation="SHA1"
 />
```

So what?

What are the other risks around the ViewState?

- Information in the ViewState bag
- Details about custom objects serialized into the ViewState

ViewState and Padding Oracle Attack

- You were able to use the ViewState to find out the validation and decryption keys, but only if the custom errors were not enabled.
- Using the parameters of the WebResource.axd / ScriptResource.axd files is more efficient though.

Other implications of compromized keys

- Download of the web.config file (.NET 3.5+)
- Faking cookies

Conclusion

Compass Security AG
Werkstrasse 20
Postfach 2038
CH-8645 Jona

Tel +41 55-214 41 60
Fax +41 55-214 41 61
team@csnc.ch
www.csnc.ch

Conclusion

Input Validation

- Activate the RequestValidation but not solely trust this input filter.
- Use a well tested filtering framework such as the OWASP Enterprise Security API (ESAPI) or the Microsoft AntiXSS library.

Output Encoding

- Consequently encoding the content of every control before it gets rendered into the page.
- Use a well tested encoding framework such as the OWASP Enterprise Security API (ESAPI) or the Microsoft AntiXSS library

Avoid ViewState Tampering

- Enable MAC integrity check to prevent tampering
- Enable EventValidation to restrict values for each specific control
- Enable ViewStateUserKey to bind ViewStates to the user session
- Encrypt ViewState to avoid disclosure and caching of confidential information

Conclusion

Assure

- Developers need to understand the security implications and features
- Enforce input filtering and output encoding
- Make use of approved filter and encoding frameworks
- Enroll a secure development life cycle (use CAT.NET to analyze assemblies)

Mind

- Server and client logic my be exploited due to malicious data formats
- Entry server, WAF protection may be bypassed using custom data formats
- Other frameworks (Java Server Faces) know the concept of a ViewState as well
  - Apache MyFaces
  - SUN Project Mojarra

Conclusion

Implement Best Practices

- Guard Against Malicious User Input
  - ViewState Security
  - Input Filter
  - Output Encoding
- Run Applications with Least Privileges and Know Your Users
  - Authentication
  - Authorization
  - ACLs
- Keep Sensitive Information Safely
  - SSL protect traffic
  - Safeguard configuration
  - Keep sensitive information assets at the server side
  - Strong encryption (System.Security.Cryptography)

Conclusion

Implement Best Practices

- Use Cookies Securely
  - No sensitive information
  - Secure cookies settings
- Access Databases Securely
  - Inherent security
  - Parameterized queries
  - Protected configuration
- Create Safe Error Messages
  - Custom error pages
  - No detailed error messages

Swiss Cyber Storm 3

COMPASS SECURITY

MEET THE LEAD

- Cyber Storm Briefings
- Security Talks
- Prof. A. Gloor (USA)
- Raoul Chiesa (ITA)
- Marc Henauer (MELANI)
- Dr. Bruce Nikkel (UBS)
- Regula Späni
- VIP Schiff

DO/FR

MEET THE GEEK

- Cyber Storm Wargames
- Security Challenges
- Capture the Flag
- Hack-in-the-Box
- 0-day Exploits
- Lock-Picking
- OWASP Conference
- HackNight Party

SA/SO

12-15 May

OWASP Track

Thursday

- OWASP Initiatives and Strategy - Antonio Fontes
- HTTP Parameter Pollution - Marco Balduzzi
- Become fully aware of the potential dangers of ActiveX attacks - Brian Mariani
- Application Security as a Team Effort - Jörg Ewald
- HTML5 (In)Security - Thomas Röthlisberger

Friday

- Hunting Slowloris and Friends - Christian Folini
- DomXSS identification and exploitation - Stefano Di Paola
- Botnet Resistant Coding - Peter Greko & Fabian Rothschild
- Cookie Jacking - UI Redressing - Rosario Valotta
- Web Port Knocking - Yiannis Pavlosoglou
- Strong Authentication in Web Application - Sylvain Maret

Appendix

Compass Security AG
Werkstrasse 20
Postfach 2038
CH-8645 Jona

Tel +41 55-214 41 60
Fax +41 55-214 41 61
team@csnc.ch
www.csnc.ch

References

ViewState Documentation and Analysis Tools

- ViewState
  http://msdn.microsoft.com/en-us/library/ms972976.aspx
- Trustwave's SpiderLabs Security Advisory TWSL2010-001
  https://www.trustwave.com/spiderlabs/advisories/TWSL2010-001.txt
- ViewStateViewer
  http://labs.neohapsis.com/2009/08/03/viewstateviewer-a-gui-tool-for-deserializingreserializing-viewstate/
- Fiddler 2 & ViewState Interceptor Plugin
  http://www.fiddler2.com/
  http://www.binaryfortress.com/aspnet-viewstate-helper/

Security Guides

- Microsoft Web Application Best Practices
  http://msdn.microsoft.com/en-us/library/zdh19h94.aspx
  http://msdn.microsoft.com/en-us/library/330a99hc.aspx
- OWASP Development Guide
  http://www.owasp.org/index.php/Category:OWASP_Guide_Project

References

Security Tools and Frameworks

- OWASP Enterprise Security API (ESAPI) for .NET
  http://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API
- AntiXSS API from Microsoft
  http://www.microsoft.com/downloads/details.aspx?familyid=051EE83C-5CCF-48ED-8463-02F56A6BFC09
- CAT.NET (Assembly Analyzer)
  http://www.microsoft.com/downloads/details.aspx?FamilyID=0178e2ef-9da8-445e-9348-c93f24cc9f9d

.NET Hints

- List of .NET Controls and Encoding
  http://blogs.msdn.com/b/sfaust/archive/2008/09/02/which-asp-net-controls-automatically-encodes.aspx
- Page State Persister Example
  http://msdn.microsoft.com/en-us/library/aa479403.aspx
- Securing the ViewState
  http://msdn.microsoft.com/en-us/library/ms178199(VS.85).aspx
- Configuration Locking
  http://learn.iis.net/page.aspx/145/how-to-use-locking-in-iis-70-configuration/

References

Sample CAT.NET output

Result #9

Summary

Problem	A cross-site redirection vulnerability was found through a user controlled variable that enters the application at adminlogin.aspx. variable stack0 which eventually leads to a cross-site redirection issue at adminlogin.aspx.cs:60.
Resolution	Do not allow off-site redirections to absolute URLs that can be specified by the user.
Entry Variable	stack0
Confidence	High

Source Context	Line	Input Variable	Statement
adminlogin.aspx.cs	57		string returnUrl = Request.QueryString["ReturnUrl"];
adminlogin.aspx.cs	57	Return from HttpRequest.get_QueryString	string returnUrl = Request.QueryString["ReturnUrl"];
adminlogin.aspx.cs	60	returnUrl	Response.Redirect(returnUrl, true);

Cross-Site Scripting (ACESEC05)

35 results

Result #10

Summary

Problem	A cross-site scripting vulnerability was found through a user controlled variable that enters the application at noscript.aspx:23 variable stack1 which eventually leads to a cross-site scripting issue at noscript.aspx:160.
Resolution	Use the Anti-XSS library to properly encode the data before rendering it
Entry Variable	stack1
Confidence	Low (External call to non-system method, cannot verify data taints result.)

Source Context	Line	Input Variable	Statement
noscript.aspx	232		engine.RequestInfo.RequestUrl = txtUrl.Text;
noscript.aspx	232	Return from TextBox.get_Text	engine.RequestInfo.RequestUrl = txtUrl.Text;

COMPASS SECURITY

COMPASS SECURITY
NETWORK COMPUTING AG