Beware of Serialized GUI Objects Bearing Data

David Byrne
Rohini Sulatycki

Schedule

- Definitions
- Is this an 0-day?
- Poor vendor documentation
- MyFaces demo & code explanation
- ASP.Net demo & code explanation
- Recommendations

Definitions: View State

- **Post-back: a request send to the server with the response sent back to the requesting page**
  - Select US from country drop down
  - Trigger post-back to populate state drop down

- **A mechanism used by some web application frameworks to preserve the state of a web page's HTML GUI controls across post-backs.**

JSF:

- Tree like structure of the UI component hierarchy
- State for each component

.NET

- Changes to control properties

Definitions: View State Tampering

- **Malicious data embedded in client-side view states**
- **Server renders attacker's data in its response**
- **Data can be client-side code (XSS) or limited server-side code**

Affected Web Application Frameworks

- JavaServer Faces
    - Apache MyFaces
        - 1.2.8
        - 1.1.7
    - Sun Mojarra
        - 1.2_14
        - 2.0.2
- Microsoft ASP.Net
    - 3.5

Is this an 0-day?

- **Specific attacks never documented before**
- **Prevented by existing configuration options**
- **WEP (wireless encryption)**
  - Introduced in 1997 with known weaknesses, but not attacks
  - In 2001, specific attacks were proposed
  - Late in 2001, AirSnort was released
  - WEP is still available on access points today
  - Solution is to configure access points to use WPA

No Advice From Sun

http://192.9.76.37/Wiki.jsp?page=JavaServerFacesRI

How can I secure view state when using client side state saving?

By default, view state will not be encrypted. However, there is a way to do this within Mojarra.

Specify a environment entry like so:

```
<env-entry>
    <env-entry-name>com.sun.faces.ClientStateSavingPassword</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>[SOME VALUE]</env-entry-value>
</env-entry>
```

The presence of this JNDI entry will cause the state to be encrypted using the specified password. We realize this isn't the most secure way of conveying a password, however, this cannot be accessed easily without having code executed on the server side.

Mostly Bad Apache Recommendation

http://wiki.apache.org/myfaces/Secure_Your_Application

One consequence of client side state saving is that anyone with a decoder and some time to kill can reconstruct the UI object model on the client side. This can be a problem for those of us who make use of the excellent t:saveState tag.

Enabling encryption is as easy as putting the following context parameter in your deployment descriptor. There are two things to note here. First, this uses the default encryption algorithm, DES, so the secret must have a size of eight. Second, although the secret is actually "76543210", we do not put this directly in the deployment descriptor. Instead, we place it's base 64 encoded value. This annoying extra step in the process is required so that secrets are not limited to printable character values.

Bad Microsoft Recommendations

http://msdn.microsoft.com/en-us/library/ms691344.aspx

http://msdn.microsoft.com/en-us/library/
system.web.configuration.pagessection.enableviewstatemac.aspx

http://msdn.microsoft.com/en-us/library/ydy4x04a.aspx

EnableViewStateMac

A read/write Boolean value. true if ASP.NET should run a message authentication check on the page's view state when the page is posted back from the client; otherwise, false. The default is true.

Note:

Do not set this property to true if performance is a key consideration.

Good, But Ambiguous Microsoft Recommendations

http://msdn.microsoft.com/en-us/library/bb386448.aspx
http://msdn.microsoft.com/en-us/library/ms998288.aspx

Securing View State

By default, view state data is stored in the page in a hidden field and is encoded using base64 encoding. In addition, a hash of the view state data is created from the data by using a machine authentication code (MAC) key. The hash value is added to the encoded view state data and the resulting string is stored in the page. When the page is posted back to the server, the ASP.NET page framework re-computes the hash value and compares it with the value stored in view state. If the hash values do not match, an exception is raised that indicates that view state data might be invalid.

By creating a hash value, the ASP.NET page framework can test whether the view state data has been corrupted or tampered with. However, even if it is not tampered with, view state data can still be intercepted and read by malicious users.

Someone at Microsoft Understands

http://support.microsoft.com/kb/829743

If you turn the view state MAC feature off, and then you use view state for controls that do not HTML encode (for example, a Label control), attackers can tamper with the view state data and can put arbitrary data in view state. This arbitrary data is decoded and then used by controls when they render the posted page. As a result, attackers can inject script into the application unless you work to prevent the attack. For example, an attacker could decode the data, inject script into the data where a Label control is, and then link to it from a Web site. Anyone who clicks on the link would be the victim of a script injection attack that could potentially steal their authentication cookies or session id. The script could also let an attacker alter state data for controls that use view state and application specific attacks could occur as a result.

Introduction to JavaServer Faces

- **Aids in development of web-based GUIs**
- **Java EE Standard**
- **Sun Mojarra is the official Java EE reference implementation**
- **Apache MyFaces is another popular implementation**
- **Built on JavaServer Pages, typically runs on Tomcat**
- **Standards process started in 2001 with first release in 2004**
- **No code base in common between the Sun & Apache products**

JSF Request/Response Lifecycle

1. Restore View

- Initial Browser Request
  - Create empty view
  - Go to ProcessResponse
- Browser Post-back
  - Restore view from state
  - Restored component tree

2. Apply Request Value

- Traverse component tree
- Invoke decode() on each component
- Get component values from request
- Validate values
- Queue errors

JSF Request/Response Lifecycle Cont'd

3. Update Model

- Traverse component tree
- Update server-side model properties

4. Invoke Application Events

- Handle code specific to the application

5. Process Response

- Initial Request
  - Delegate to JSP container
  - Add components form page to component tree

JSF Request/Response Lifecycle Cont'd

6. Create View State

- Serialize component tree and encode

7. Generate web page

JSF State Saving Methods

Location where view state information is saved

Client

- Entire serialized view state rendered in HTML markup e.g. hidden field
- Performance issues with large view states

Server

- Serialized view stored in session
- Unique key sent to client to identify view
- Issues with clustered environments

JSP Expression Language (EL)

Part of the JavaServer Pages (JSP) standard

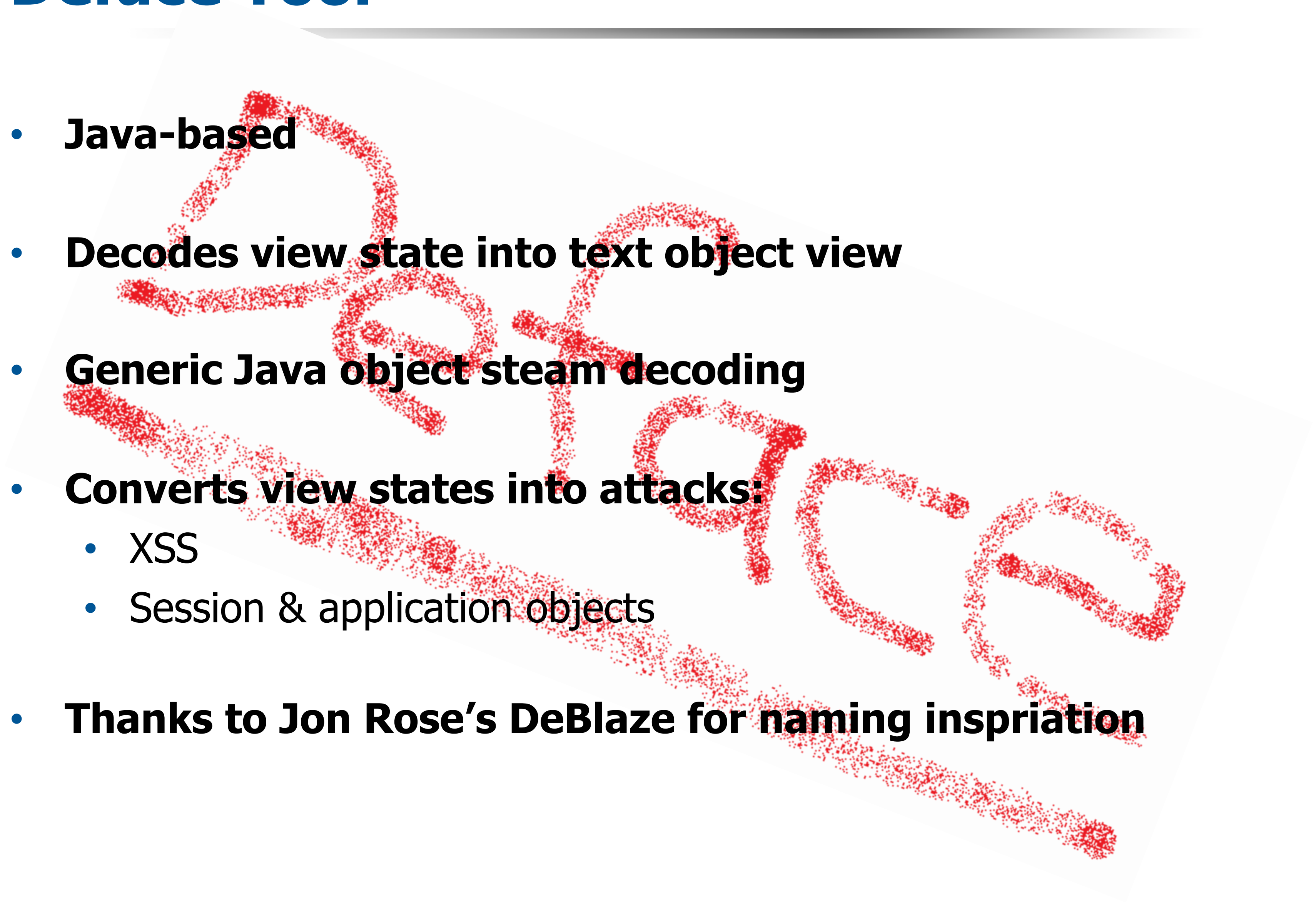
EL allows a page author to access server side data stored in Java Beans

- #{name.property}

Provides access to Implicit Objects

- #{sessionScope}
- #{applicationScope}
- #{requestScope}
- …

Deface Tool



- **Java-based**
- **Decodes view state into text object view**
- **Generic Java object steam decoding**
- **Converts view states into attacks:**
  - XSS
  - Session & application objects
- **Thanks to Jon Rose's DeBlaze for naming inspriation**

Demo Overview

Environment

- Running latest versions of all software:
    - Apache Tomcat 6.0.24
    - Apache MyFaces 1.2.8

Web application

- Credit card form
- Submits to a thank you page
- Puts credit card number and security code in session
- Security code (CVV) should not be permanently persisted and a common place to put is in the session

Deface Demo

JSF View State Tree

```
Structure object:
ARRAY (java.lang.Object[3]) (#19739141):
 [0]: (org.apache.myfaces.application.TreeStructureManager.TreeStructComponent)
  field "_componentClass": 'javax.faces.component.UIViewRoot' (String)
  field "_componentId": 'j_id_jsp_1379279123_0' (String)
  field "_children": ARRAY (org.apache.myfaces.application.TreeStructureManager
$TreeStructComponent[2]):
   [0]: (org.apache.myfaces.application.TreeStructureManager.TreeStructComponent)
    field "_componentClass": 'javax.faces.component.html.HtmlOutputText'
    field "_componentId": 'j_id_jsp_1379279123_2' (String)
   [1]: (org.apache.myfaces.application.TreeStructureManager.TreeStructComponent)
    field "_componentClass": 'javax.faces.component.html.HtmlForm' (String)
    field "_componentId": 'form2' (String)
    field "_children": ARRAY (org.apache.myfaces.application.TreeStructureManager
$TreeStructComponent[1]):
```

JSF View State Restore

```
private UIComponent internalRestoreTreeStructure(TreeStructComponent treeStructComp) {
    String compClass = treeStructComp.getComponentClass();
    String compId = treeStructComp.getComponentId();
    UIComponent component = (UIComponent)ClassUtils.newInstance(compClass);
    component.setId(compId);

    TreeStructComponent[] childArray = treeStructComp.getChildren();
    if (childArray != null) {
        List<UIComponent> childList = component.getChildren();
        for (int i = 0, len = childArray.length; i < len; i++) {
            UIComponent child = internalRestoreTreeStructure(childArray[i]);
            childList.add(child);
        }
    }

    Object[] facetArray = treeStructComp.getFacets();
    if (facetArray != null) {
        Map<String, UIComponent> facetMap = component.getFacets();
        for (int i = 0, len = facetArray.length; i < len; i++) {
            Object[] tuple = (Object[])facetArray[i];
            String facetName = (String)tuple[0];
            TreeStructComponent structChild = (TreeStructComponent)tuple[1];
            UIComponent child = internalRestoreTreeStructure(structChild);
            facetMap.put(facetName, child);
        }
    }

    return component;
}
```

ASP.NET View State Lifecycle

View state is utilized when a control's state are modified programmatically

1. Initial Browser Request

- Class auto generated
- Create page control hierarchy programmatically

2. Post-back

- Load view state
- Populate page control hierarchy with view state data
- Validate for tampering etc

ASP.NET View State Lifecycle Cont'd

3. Load post-back data

- Obtain data from request
- Check if corresponding control in hierarchy implements iPostBackdataHandler interface?
- Pass value to server control to update state

4. Fire Page Load event

ASP.NET View State Lifecycle Cont'd

5. Save View State

- Recursively call control hierarchy SaveViewState()
- Construct Page ViewState
- Serialize to Base64 encoded string
- Placed in a hidden field __VIEWSTATE

6. Generate Web Page

ASP.Net Demo

Environment

- Windows Server 2008 R2
- ASP.Net 3.5
- All current patches

```
<%@ Page EnableViewStateMac="False"
ValidateRequest="True" %>

<html runat="server">
    <form runat="server"/>
</html>
```

ASP.Net Demo

What can be done with this?

- **Framework exploits**
  - XSS
  - Steal session and application data from JSF

- **Developer mistakes**
  - SQL injection
  - Authentication bypass
  - Authorization bypass
  - Anything really

- **Attack scenarios**
  - Phishing attacks
  - XSS shipping session variables to other web servers
  - Changes to data grid

Recommendations

- Always, always, always secure the view state
- Move the view state to the server whenever possible
- Performance benefits
- Server farm environments
- Documentation changes by vendors
- We don’t make up best-practices for fun

Automated vs. Manual

- Complex vulnerabilities require manual testing
- Automated tools only find known flaws
- Automated vs. Manual: You can’t filter The Stupid

Links

Deface tool & demo environment
https://www.trustwave.com/spiderLabs-tools.php

Apache MyFaces security
http://wiki.apache.org/myfaces/Secure_Your_Application

Sun Mojarro security
http://192.9.76.37/Wiki.jsp?page=JavaServerFacesRI

ASP.Net view state security
http://msdn.microsoft.com/en-us/library/ms178199(VS.85).aspx

ViewStateHacker
http://www.woany.co.uk/viewstatehacker/

.NET Reflector
http://www.red-gate.com/products/reflector/

Trustwave® SpiderLabs℠

Questions