Auditing WebObjects applications

Ilja van Sprundel <ivansprundel@ioactive.com>
Who am I?

- Ilja van sprundel
- IOActive
- netric
- blogs.23.nu/ilja
Agenda

- Why?
- Introduction
- WebObjects?
- Components
  - html
  - wod
  - java
- Direct actions
- what do Direct actions and Component requests look like?
- response splitting
- escaping data
- Escaping data
- Deployment issues
- todo
- conclusion
- Q&A
Why?

- not really all that common
- I’ve had to codereview and pentest WebObject webapps
- there is virtually _NOTHING_ published about WebObjects (in terms of security)
- These are my notes (in a more coherent form)
Introduction

• This talk is about WebObjects
• How it looks from an code reviewing perspective ...
• ... and a pentesting perspective
• not about new types of webbugs or attackvectors
Introduction

• Will walk through how most WebObjects more or less look and feel
• what’s required to make it work
• what you care about from a security point of view
• will only consider WebObject specifics.
• if it ain’t related to WebObject api’s and classes I’m not covering it
• limited to rendering (for now)
WebObjects?

- An application server
- By Apple
- Application server
- Web Application framework
WebObjects?

• Early versions (up until 4.x) used objective-c

• MacOSX only

• later versions (5.x) are pure java

• and can be deployed anywhere

• this talk will only cover the later versions
Components

• Rather object orientated way of making web applications
• each web application is seen as a collection of components
• each component exist out of 3 basic things:
  • html file (.html)
  • object definition file (.wod)
  • java source code files (.java)
components example

hw.html
<html>
<head>
  <title>Untitled</title>
</head>
<body>
Hello World
<webobject name = "Now"> </webobject>
</body>
</html>

hw.wod
Now: WOString {
  value = now;
  dateformat = "%m%d%Y";
}

hw.java
package your.app.components;
import com.webobjects.foundation.*;

public class hw extends WOComponent {
  private static final long serialVersionUID = 1L;

  public hw(WOContext context) {
    super(context);
  }

  public NSTimestamp now() {
    return new NSTimestamp();
  }
}
.html file

- WebObject html files also support a 
  `<webobject>` tag
- `<webobject name=”name”>... </webobject>`
- only a name is given, nothing else
- it’s defined in the .wod file
.wod

- .wod file specifies what type of objects
- there’s quite a few of them
  - WOString
  - WOHyperlink
  - WOImage
  - WOTextField
  - WOForm
  - WOButton
  - ...
- you can also embed your own objects in there
Each of these types has attributes
most of these types get rendered into html eventually
not really any consistency among them
some do encode, some dont
not documented at all!
the attributes can be static
or can all into java code
• WOString
• does html escaping by default
• has an attribute HTMLescape
• set to True by default
• XSS possible if set to false
name1: WOString {
    value = getvalue;
}

name2: WOString {
    value = getvalue;
    escapeHTML = false;
}
• WOHyperlink
• href attribute
• does not encode with absolute url’s
• does encode with relative ones
• WOImage
• src attribute like WOHyperlink’s href
• filename is properly encoded
• value is properly encoded as well
• WOTextField

• both value and name are properly encoded
- WOForm
- href never encoded, vuln to xss
- has an attribute named queryDictionary
  - callback returns a dict of key/value pairs
  - will be used as <input> tags inside the form
  - key is not encoded!
  - value is properly encoded
- name is not encoded
• many more
• none are documented (as in, how is encoding handled)
• can also include other WOComponenents
Each components is seen as a class

extends from WOComponent

it’s constructor has 1 argument WOContext

basically an http context (contains stuff like request, response, session, ...)

all it’s methods can call context() to get the current WOContext
Classes you want to know about:

- WOResponse is class for the http request
- WOResponse is class for http response
- WOSession holds the session
  - all methods can call session() to get it
- WOCContext is the http context
WOComponent

- all components inherit from this one
- some of its methods (always) get called
- can be seen as entry- and exit-points
- Constructor
- AppendToResponse() (if derived class overwrote it)
public class Main extends WOComponent {
    public Main(WOContext context) {
        super(context);
    }

    public void appendToResponse(WOResponse response, WOContext ctx) {
        super.appendToResponse(response, ctx);
        response.setContent(ctx.request().stringFormValueForKey("xss"));
    }
}
Direct actions

- More light weight than Component based
- easier to wrap your head around
- class that extends from WODirectAction
- no .html file
- no .wod file
- pretty straight forward
Direct actions

- implements methods that look like
  ```java
  public WOActionResults NameAction() {
      ....
  }
  ```

- basically `<anything> Action() ` that looks like that can directly get called with GET or POST
Direct actions

• method request() available
• which provides the current WOREquest
what does it look like

• Calling Component action directly:
  • [http://site/cgi-bin/WebObjects/applicationname.woa/wo/component.wo?...](http://site/cgi-bin/WebObjects/applicationname.woa/wo/component.wo?...)

• Calling Direct action directly:
  • [http://site/cgi-bin/WebObjects/applicationname.woa/wa/action?...](http://site/cgi-bin/WebObjects/applicationname.woa/wa/action?...)
response splitting

• Default redirect object WORedirect

```java
public WOAActionResults toeterAction() {
    WORedirect page = (WORedirect) pageWithName("WORedirect");
    page.setURL(request().stringFormValueForKey("TOETER"));
    return page;
}
```

• Vulnerable to http response splitting

• does not url encode \r or \n
response splitting

• yes, cookies too

```java
public void appendToResponse(WOResponse response, WOContext ctx) {
    super.appendToResponse(response, ctx);
    WOCookie aCookie = new WOCookie("key", ctx.request().stringFormValueForKey("cookieval"));
    response.addCookie(aCookie);
}
```

• also, no encoding of ;

• allows for cookie injection

• same thing with all of WOCookie’s set*() methods
response splitting

• Works on response.setHeader() too ....
escaping data

- WOResponse.appendContentHtmlString()
- WOResponse.appendContentHTMLAttributeValue()
- does not encode single quote (‘)
- think of apps doing:
  - `<... blah=’ [append here] ‘ ...>`
- can still break out of quotes, maybe inject onclick, onload, ..., depends on tag
Deployment issues

- a whole bunch of standard applications
- http://host:1085/cgi-bin/WebObjects/wotaskd.woa/woconfig
- cgi-bin/WebObjects/Monitor
- cgi-bin/WebObjects/WOAdapterInfo
- cgi-bin/WebObjects/<app>.woa/
- wa/WOStats
- wa/WOEventDisplay
- wa/WOEventSetup
- Should be password protected on any decent deployment ...
todo

- should I ever revisit WebObjects
- anything that’s not rendering
- Enterprise objects (database integration)
Conclusion

- hard to wrap your head around
- turns out, browsing webpages really isn’t object orientated!
- framework feels old (web 1.0).
- Security wise it’s not up to par with others
  - no easy XSRF protection
  - almost everything is XSS’able
  - Response splitting is everywhere
Questions ?