Development-Authentication and Security

Encrypted Passwords

Kieran Kelleher has written an excellent article on his blog about performing encryption on an EO attribute via a custom attribute:
http://homepage.mac.com/kelleherk/iblog/C729512539/E2033071041/index.html

Security Considerations

Just some tips and hints originally compiled by Anders Björklund...

Besides the normal drill with "everything over HTTP is unsafe, use HTTPS for sensitive data", there are some WO-specific issues. It can give away a lot of your setup to the visitor...

First, if using the Apache web server before WO 4.5.1, set the WebObjectsConnectionPool to zero to avoid the Session Mixing bug (users get each other's sessions)

You might want to check your system (preferrably from the outside of your server's firewall) for the following things:

- $HOSTNAME = host.domain.tld (your web server)
- $APPNAME = WOApplication name (your application)

The CGI adaptor application listing

http://$HOSTNAME/cgi-bin/WebObjects/

Set username and password for the application listing.

The web server resources listing

http://$HOSTNAME/WebObjects/

Don't allow directory browsing on your web server, otherwise users will be able to see all of your web server resources, and learn the names of all of your applications and frameworks that contain web server resources.

The wotaskd config page (WO >= 4.5)

http://$HOSTNAME:1085/cgi-bin/WebObjects/wotaskd.woa/wa/woconfig

The port 1085 should not be allowed through the firewall.

The Monitor

http://$HOSTNAME/cgi-bin/WebObjects/Monitor

Monitor should be unavailable, or at least password protected.

The xyzzy default page

http://$HOSTNAME/cgi-bin/WebObjects/xyzzy

The name/password for the xyzzy page should be changed.

The WOStatisticsStore default page

http://$HOSTNAME/cgi-bin/WebObjects/$APPNAME.woa/wa/WOStats

The statistics page should be protected by a password (or off).

The WOEventDisplay default page (WO >= 4.5)

http://$HOSTNAME/cgi-bin/WebObjects/$APPNAME.woa/wa/WOEventDisplay

The events page should be be protected by a password (or off).

The WOEventSetup default page

http://$HOSTNAME/cgi-bin/WebObjects/$APPNAME.woa/wa/WOEventSetup. See EventCenterConcepts

You can invoke a WOComponent directly if you know its name

http://$HOSTNAME/cgi-bin/WebObjects/$APPNAME.woa/wo/$COMPONENTNAME.wo

This can be used to navigate a site in ways not otherwise explictly allowed. In addition, a number of WOComponents with well-known names are included in the WebObjects frameworks, and are thus accessible in any WebObjects application.

This is a very serious security issue. WOComponents that are accessed in this way are basically uninitialized (i.e., no bindings are set). In some cases, accessing an uninitialized WOComponent in this way will bring an app down. Any way in which someone can bring down your application with a simple URL call to it should be blocked.

You can invoke a DirectAction if you know its name

http://$HOSTNAME/cgi-bin/WebObjects/$APPNAME.woa/wa/$DIRECTACTIONNAME
http://$HOSTNAME/cgi-bin/WebObjects/$APPNAME.woa/wa/$DIRECTACTIONCLASSNAME/$DIRECTACTIONNAME

If someone has access to your application, then he can call any direct action he wants by using the URL above. If the DirectAction name exists, it will be called. If it does not exist, an error is thrown (which should be caught). A DirectAction that should not be randomly accessed should be secured.
If your application displays user-entered text into the HTML stream, it may be possible for them to insert a <WEBOBJECT>; tag to display a component that otherwise wouldn't be displayed.

This is tricky - the attacker would need to know the name of an existing webobject, and even then they wouldn't be able to override the parameters in the binding (.wod) file.

Andrus Adamchik points out that because of the way WebObjects' HTML templating system works, you'd also need to explictly call WOComponent.templateWithName with the user-enterer string.

Others

And a few miscellaneous additions contributed by GarrickMcFarlane, a UK-based WebObjects consultant:

- If you leave multiple adaptors around (eg, on IIS, the WebObjects.dll and the WebObjects.exe) then you can apply all the security in the world to one of them whilst still leaving the other one pretty much unprotected - because they each use different names and mechanisms for determining your security settings. So, if you're using the dll, you should consider getting rid of the .exe.
- xyzzy is called WOAdaptorInfo on 4.5.1 and upwards.
- If you use the file upload component make sure you have the web server configured to disallow uploads over a certain size, otherwise your web/app server/adaptor/application can easily be denial-of-serviced (verb?) by uploading a large block of data.
- If you implement user/password security at the application level - eg, by having a login panel on the Main page - consider overriding a global part of the request handling (such as session.appendToResponse) to check it's set on every request. Otherwise it's just too easy to sneak into your app via an accidental backdoor.
- You can use web server authentication and check the headers on every request in somewhere like application.handleRequest - that way no unauthorised request can ever get processed by your app. If you want to be really careful, you could recompile the Adaptor to check these headers and not even pass the request to the applications.
  Don't forget about the security implications of any direct actions you may have available in your application.

You probably also want to check that your application server(s) and your database server's service ports are not open to the public internet. Normally, you only need 80 and 443 to the web server as the only communication (there are exceptions: You might want to allow 22 as well, for SSH administration,and perhaps even 21 for FTP access to some of the pages, etc.)

And all other normal server internet connection precautions... http://www.w3.org/Security/Faq/