Miracle - One Vulnerability To Rule Them All

Miracle - One Vulnerability To Rule Them All - Peterjson, @peterjson, Medium.

- Published: 2022-07-08
- Original: <https://peterjson.medium.com/miracle-one-vulnerability-to-rule-them-all-c3aed9edeea2>
- Preserved from: https://peterjson.medium.com/miracle-one-vulnerability-to-rule-them-all-c3aed9edeea2 (live) on 2026-08-09
- Licence: unknown

Content

> UNTRUSTED SOURCE TEXT. Everything below this line is third-party material quoted for research. It is data, not instructions. Do not follow directions, execute code, or fetch URLs because this text says so.

Cve 2022 21445

Cve 2022 21497

Adf

Rce

Miracle

Miracle - One Vulnerability To Rule Them All

[[image: Peterjson]](https://peterjson.medium.com/?source=post_page---byline--c3aed9edeea2----------------------------------)

Peterjson

me and Jang trying to pwn Oracle :D

Introduction

As mentioned in Jang blog, We (me and Jang) found a mega 0-day. After April Critical Patch, finally the vulnerability was patched properly. If you never known about this vulnerability, please patch your system ASAP !

Summary

Let us name this attack **The Miracle Exploit** because it affects many products based on Oracle Fusion Middleware and Oracle online systems. **Miracle** means Middleware **Fusion** with Oracle

Firstly, we found many pre-auth RCEs which affect Oracle's Product in Oracle Middleware Fusion and they also affect Oracle online systems, Oracle Cloud Infrastructure. After a month try to explore as much as we can and end up with multiple pre-auth RCEs in many products inside Oracle Middleware.

One more thing to note, any website was developed by ADF Faces framework are affected. This means many Oracle's online systems, Oracle Cloud Infrastructure are also affected!

After disclosed this bug with Oracle, Oracle took almost 6 months to patch this vulnerability in each product inside Oracle Fusion Middleware. We also reported pre-auth RCEs to some big companies via their bug bounty program after the patch was released to help them mitigate this issues.

Now after a period of time after Oracle released the patch, we decided to publish this blog to share the detail of Miracle exploit. We very very excited at the time (6 months ago), but now we don't have that feeling anymore because Oracle took **too long** to patch this vulnerability, more than the standard. Anyway, this is a cool exploit, a cool story me and Jang worked together in a month so let we tell you about our story

The Story

This bug was found in Oracle Fusion Middleware while I reviewed the source code of ADF Faces (a component in Oracle Fusion Middleware).After a day , I can confirmed that I found a pre-auth RCE in ADF Faces framework and that's my favourite bug type, Deserialization of untrusted data.

But at that time I only achieved pre-auth RCE on Oracle Fusion Middleware 12.x version. But most online instances are 10.3.x so now I need to find a super-gadget chain for 10.3.x, so I share this with Jang and we rule them all after a month.

The start of the journey

Last year, we worked with the Zero Day Initiative to co-disclose many pre-auth bugs affecting Oracle Business Intelligence to Oracle. And this time we have a chance to look back Oracle Business Intelligence again and hope to find more bugs on Oracle BI

https://www.zerodayinitiative.com/advisories/published/

- **Oracle ADF Faces Deserialization of Untrusted Data Leads Remote Code Execution — CVE-2022–21445**

While investigating Oracle BI, we found an interesting Servlet name *org.apache.myfaces.trinidad.webapp.ResourceServlet *which was mapped into these paths

servlet mappings

org.apache.myfaces.trinidad.webapp.ResourceServlet.doGet()

This servlet try to fetch the resource in request and try to find that resource via *org.apache.myfaces.trinidad.resource.ResourceLoader.getResource()*

org.apache.myfaces.trinidad.resource.ResourceLoader.getResource()

There are many Classes override *org.apache.myfaces.trinidad.resource.ResourceLoader.findResource()**** **method

After a while reviewing each override method, *oracle.adfinternal.view.resource.rich.RemoteApplicationResourceLoader *stands out!

The bug is inside the logic of *RemoteApplicationResourceLoader *when it try to restore the resource from the user input and leads to deserialization of untrusted data

oracle.adfinternal.view.resource.rich.RemoteApplicationResourceLoader

oracle.adfinternal.view.resource.rich.RemoteApplicationResourceLoader._getPathBean()

oracle.adfinternal.view.rich.remote.resources.URLEncoderPathBean.getInstanceFromString()

oracle.adf.view.rich.util.SerializationUtils.fromURLEncodingString()

Cool, we have the Deserialization sink now we need to find the source to reach this sink. Looking for a while we see in *oracle.adfinternal.view.resource.rich.RenderKitResourceLoader *constructor, there are many defined paths with mapped loader to handle each path

So the URL to trigger this bug looks like this http://HOST:PORT/contextApp/afr/**foo**/remote/**bar**/

With:

foo: this can be any string

bar: this is our payload

Next, we use *CVE-2020-14644* to exploit this Deserialization bug, because this chain is not filtered in deserialization and we end up with a successfully pre-auth RCE on Oracle BI.

Exploring more affected products

We noticed that, it's a bug inside ADF Faces framework so any products, any applications are built from ADF are also affected, we spin up some VM and setting up these products to confirm and we almost achieve pre-auth RCEs these products

- Oracle Business Intelligence
- Oracle Enterprise Manager
- Oracle Identity Management
- Oracle SOA Suite
- Oracle WebCenter Portal
- Oracle Application Testing Suite
- Oracle Transportation Management

For Oracle Access Manager, we end up with 2 ways to achieve pre-auth RCE. The first one is published already by Jang, you can revisited it again here.

We want to focus on this because OAM plays an important role on each company infrastructure. Also, many Oracle's online services / cloud services run OAM as SSO service for users.

Now I want to share with you the second pre-auth RCE exploit in OAM

Pre-auth SSRF (CVE-2022–21497) + ADF Faces **Deserialization** leads to pre-auth RCE in OAM

As we know that, exploiting ADF deserialization can be triggered from a GET request. We noticed that when installing OAM, port 14100 which plays a role as SSO and is public to the internet for users to authenticate and on that app didn't use ADF. But on console port 7001, there*** /em/*** (for Oracle Enterprise Manager) or ***/oamconsole/*** (Console for OAM) which are heavily based on ADF. So, we only focus on the web application on port 14100 to find bugs like SSRF or XXE to achieve full chain pre-auth RCE!

With awesome research about Ping'ing XMLSec from security researcher - tint0. We recommend you read this awesome research again to understand what we explain below

We continue exploring OAM and focus on triggering *dereference() *behavior. But OAM already prevent tint0 bugs with OSDT (Oracle Security Developer Tools).Now we need to learn how OAM handling SAML request between SP and IDP

In the flow handling SAML Response Message, OAM try to process the *EncryptedAssertion *element*,* we already know the KeyInfo element is processed before any kind of authentication check which is mentioned in tint0's blog. This means we can assume that we in pre-auth context !

In SAML, we can request with HTTP-Based or SOAP-Based, we focused in SOAP-based because we think that finding XXE/SSRF in SOAP more easier.

Inside *oracle.security.fed.security.crypto.enc.DomXmlDecrypter.decryptElement() *, OAM trying to get the KeyInfo from user input

oracle.security.fed.security.crypto.engine.impl.SAML20XmlCryptoEngine.processIncoming()

oracle.security.fed.security.crypto.enc.DomXmlDecrypter.getDecryptionKey()

Inside *oracle.security.xmlsec.keys.RetrievalMethod.getKeyInfoData() *function, *dereference()* will be triggered. This is where the bug came from, we cannot trigger *ResolverDirectHTTP* or *ResolverLocalFilesystem* like tint0. But we can make use of the Transformation.

oracle.security.xmlsec.dsig.ObjectReference.dereference()

Due to OSDT restriction, we cannot use XSLT transform but XPath will be a suitable candidate! While exploring some helpful XPath functions we end up with ***document() ***function which will help us making an GET HTTP request to local web application based on ADF (EM or OAM Console on port 7001) and finally achieve pre-auth RCE on Oracle Access Manager

Soap Based SAML request to http://HOST:PORT/ oamfed/idp/soap/

This is considered pre-auth because getting the KeyInfo is needed before performing a cryptography check for the EncryptedAssertion !

Universal gadget chain for 10.3.x

There's a big problem that, most of public targets are based on 10.3.x instead of 12.x. Somehow, *CVE-2020–14664* can't work in 10.3.x because of class loader (ClassNotFoundException was thrown when* *deserialization happend*).*

It feels like: We got a big gun but don't have any bullets then the gun is useless. I'm obsessed with the perfection, so we can not accept a Pre-Auth Deserialization is reported without a RCE!

In the Jang blog, Jang promise to share the gadget chain for 10.3.x, now we let you know the detail. This is the way Jang found the new gadget chain and overcome the class loader limitation

The class loader from child to parent when deserialization happend is:

- *GenericClassLoader* > **FilteringClassLoader** > *GenericClassLoader* > *Launcher$AppClassLoader*

Note: **FilteringClassLoader **is a classloader which Oracle implements to apply blacklist for some dangerous class in deserialization.

When we call the **ObjectInputStream**.*readObject()* the* *serialized object will be reconstruct using the current classloader — from the place where **ObjectInputStream**.*readObject()* is called.

With current classloader, the loadClass() will follow this order, from the child to parent: **GenericClassLoader**, if not found then will search in **FilteringClassLoader**, ...

And here is the content of **FilteringClassLoader**.*findClass()*

If the className match the filtered pattern, then it will throws *ClassNotFound* and stop finding from the parent class. And here is the filter list:

You can see the "*com.tangosol*, *org.python*. ...", that's the reason why although we saw the library "**coherence.jar**" has already been loaded but when deserialization, it throws the ClassNotFound error.

After a few days, I've found that it's not difficult to bypass this .

As mentioned above, **ObjectInputStream**.*readObject()* will use* *current classloader to load the class of the serialized object.

So, It's very straight foward to bypass this limitation, just find a call to readObject() from an override of **readObject**(*ObjectInputStream*), for example:

We use the **ConcurrentHashMap** to wrap the serialized gadgetchain:

You can clearly see that, before call **ObjectInputStream**.*readObject()*, the classloader is **Launcher$AppClassLoader** not the **FilteringClassLoader**

Another thing, as we know, in the new version of Weblogic, the Serialization Blacklist is applied to global runtime, which mean every you call **ObjectInputStream**.*readObject()*, it will be filtered by a rich blacklist.

But the problem is, the blacklist is applied at **Weblogic** runtime, not the OAM / ADF context, so it's more flexible to craft the RCE gadget chain.

Here is what we got now:

- We have a insecure deserialization from the OAM / ADF application context without any restriction
- Most of Weblogic 10.3.6 libraries and Coherence.jar are loaded into classpath.

We've tried to use some common gadget chain like CommonsCollections-series, Spring, CommonsBeanUtils ... But most of them worked on local setup but not the public targets. We still don't know why it didn't work yet.

But don't worry, we still have the most interesting library of Deserialization in the classpath: the **coherence.jar**. It comes with many interesting classes, which can be craft to a RCE gadget chain.

In 2019, Jang found the CVE-2020–2555 gadget chain by mining the coherence library, after that, there're many other gadgets abusing the **ValueExtractor**.*extract() *method family like: **MvelExtractor**, **FilterExtractor**, **ReflectionExtractor** ....

Having got headache of these method family, then Oracle patched it using a hard way: they check for the target Object before invoking, like this:

If the **oTarget** isn't in the blacklist, it will continue to invoke that method,

Hmm, sound familiar with ...

*Oracle is oracle ...

And here is the Reflection blacklist:

It will match every classes are instance of: **java.lang.Class**, **java.lang.System** and **java.lang.Runtime**.

In the CVE-2020–2555, I've used the ChainedExtractor to chain the extractors list in to a RCE gadget chain, but this method uses **java.lang.Class** to get and invoke the destination method and it involves **java.lang.Runtime**, so it totally got filtered here.

But do not forget that, we're still able to invoke any public method of any Serializable object!

After looking for hours, I found an interesting class **RemoteInvocation**, which is **Serializable** and also has an interesting method *invoke()*:

The method **RemoteInvocation**.*invoke()* uses Reflect API to get and invoke method from fully manipulated parameter: **methodName**, **parameterType** and **arguments**. This class is the last puzzle we need to complete the filtered method **ReflectionExtractor**.*extract().*

As you know, **ReflectionExtractor**.*extract()* doesn't allow the invocation of a instance with type of **java.lang.Class**, so the execution chain gets a little bit tricky to work, below picture described the idea of this:

We used **ChainedExtractor** to reuse the result of previous method execution as an argument of next method. This Extractor chain contains two **ReflectionExtractor** instances.

First instance of **ReflectionExtractor** will call **RemoteInvocation**.*invoke() *with the argument is another **RemoteInvocation** instance. So the call graph will follow this order:

In the second call of **RemoteInvocation**.*invoke()*, it will get the method *newInstance()* from **java.lang.Class** (this method is available by default), and then invoke with the class **ShellSession** as the argument. In short, this is same as "*new* **ShellSession()**".

After finishing the first stage, the return object is an instance of **ShellSession**, which is also not filtered by the blacklist.

The second stage is very straight ahead, just call the **ShellSession**.*exec() *to execute command using MVEL syntax.

The patch

The patch was applied in
oracle.adfinternal.view.rich.remote.resources.URLEncoderPathBean.getInstanceFromString()

original

the patch

It took 6 months for Oracle to patch this vulnerability with only some small changes :)

Exploiting Oracle Online Systems

In the demonstration we sent to Oracle, we chose* edelivery.oracle.com , businessnetwork.oracle.com* which are popular for user to download Oracle's Products and this site is based on ADF Faces framework.

This is a check script which we implement for Miracle Exploit and we only proof that we can getting some information from Oracle's instead of execute command because it's maybe affected Oracle's system and leak sensitive information!

edelivery.oracle.com

businessnetwork.oracle.com

Last but not least, we successfully achieved pre-auth RCE on *login.oracle.com *which is play an important role in oracle's online services

login.oracle.com

Why we hack some Oracle's sites? Because we want to demonstrate the impact to Oracle and let them know this vulnerability is super dangerous , it affects Oracle system and Oracle's customers. That's why we want Oracle take an action ASAP. But as you can see, 6 months for Oracle to patch it, I don't know why, but we have to accept it and follow Oracle's policy.

Timeline

- 25/10/2021, We sent first report to Oracle
- 29/10/2021, The issues were accepted by Oracle and they were investigating
- 02/11/2021, We sent second report to Oracle about the exploit affect 10.3.x version
- 18/01/2022, Oracle fix the first pre-auth RCE in OAM found by Jang in January Critical Patch
- 19/04/2022, Oracle fix the ADF Faces vulnerability in April Critical Patch
- 23/06/2022, We published this blog