JAR Malware : #Spyware #RAT

Name : tstt.jar **Size:** 2.4MiB
Type: java JAR **Mime:** application/zip
SHA256:dbd9b15d48b43018f8ea123ea6b942680bd682e12608876cbae88d22cd091ea8

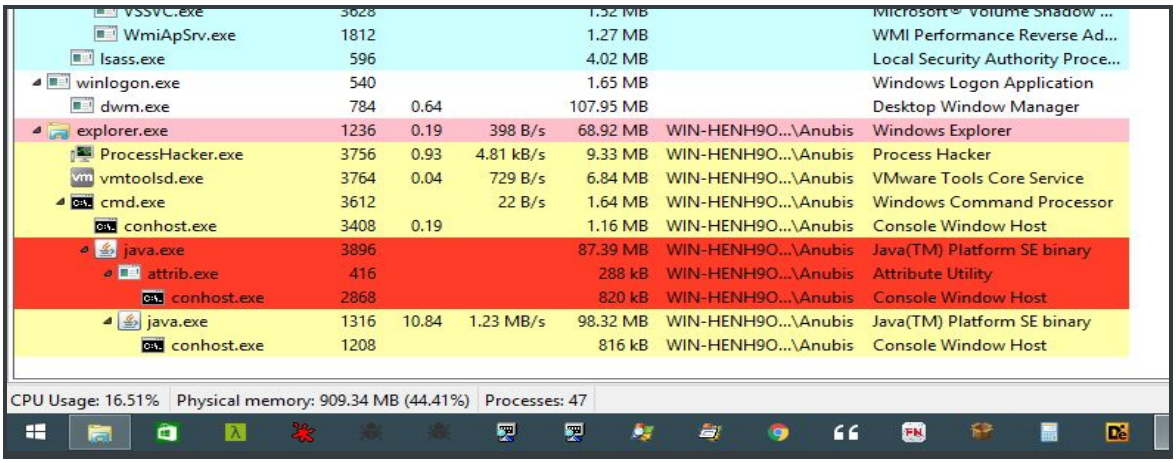
Initial Analysis :

When I've looked at the source code at first glance I noticed it was doing a lot of AES encryption and maybe worse, I thought it was obfuscated! So I've decided that I will only do my dynamic and monitoring analysis and I hope they will be enough to get as much information about the malware and its functionality.

Running the Malware :

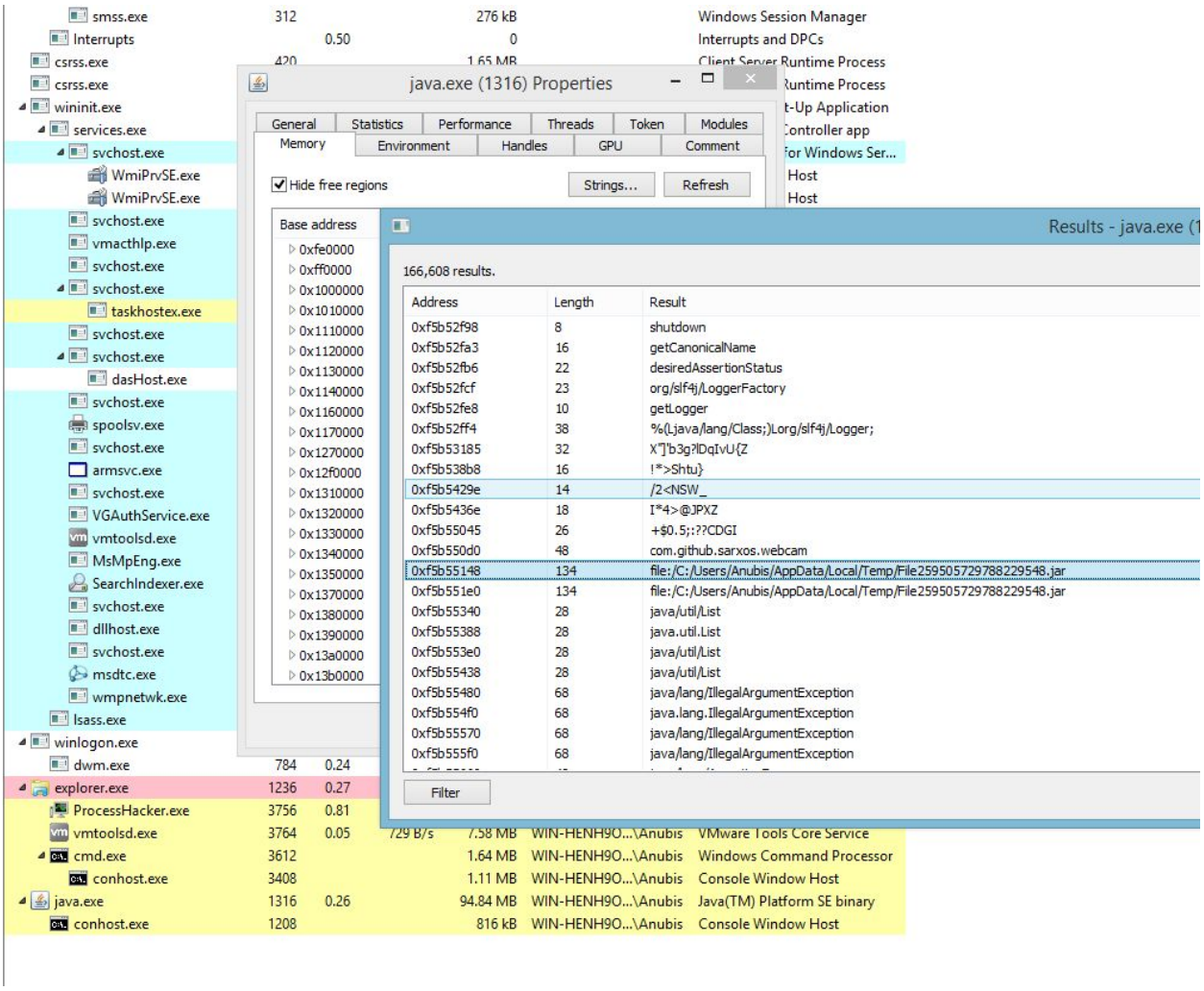
Running the malware from command line and being ready to intercept its suspicious activities using :

- Procmon.
- Microsoft Network Monitor + iNetSim.
- Process Hacker.
- Autoruns.



We can see clearly It spawns two processes : **attrib.exe** & **java.exe** .

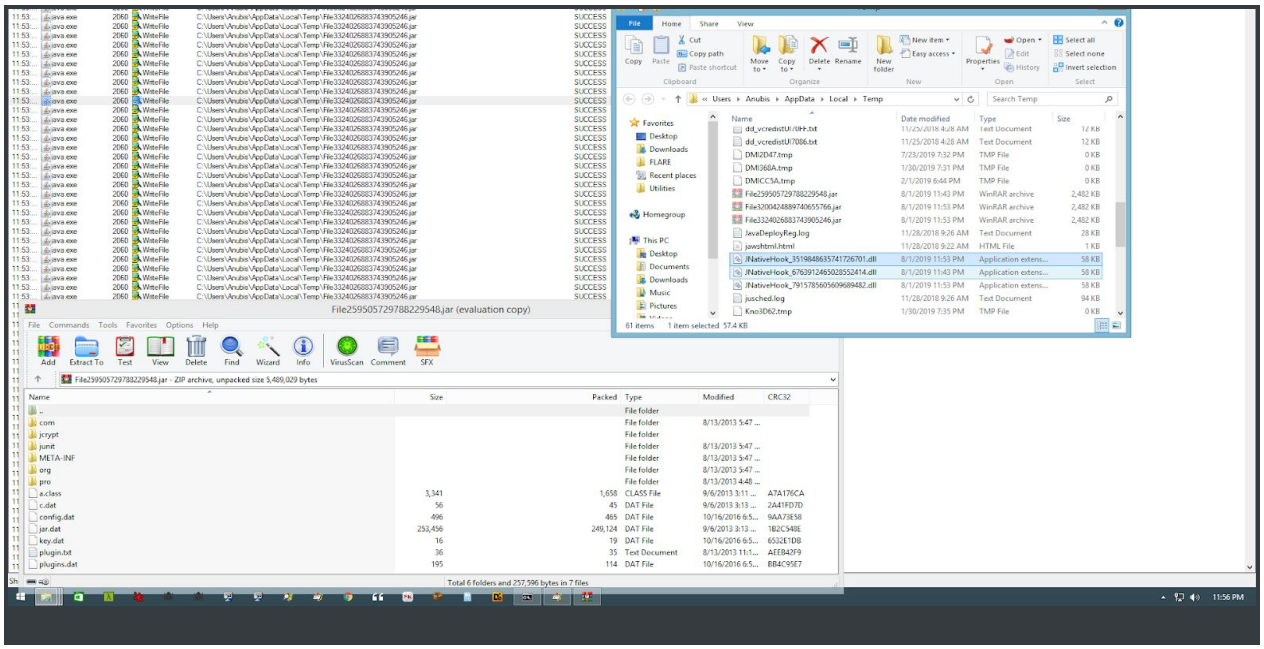
It also contains interesting strings which may lead to files being dropped !



The best way we can confirm that is by checking Procmon captured events.

Setting the right filters and yes! it actually drops more! the most interesting ones are :

- JNativeHook_4656735465228613791.dll
  // Hidden. DLL used for hooking keystrokes and mouse events..
- File7284155642637045387.jar // Hidden. JAR File.
- bridj.dll // Famous DLL used to invoke native functions.

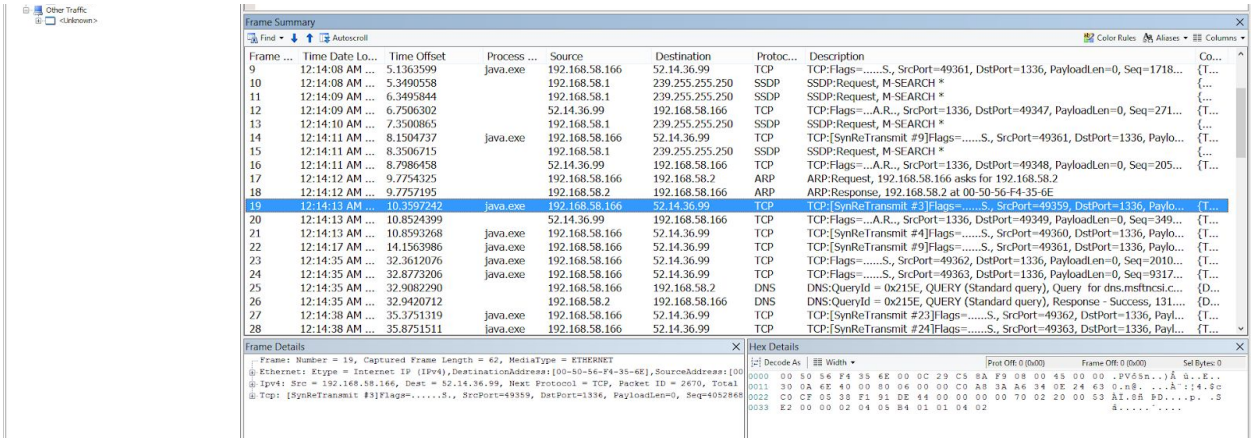


Upon inspecting the new dropped JAR file, it contains a lot of new third party packages like **github.sarxos.webcam** library and **Jrat** and a lot more of key-logging & spyware techniques ! It's safe to assume now we have identified a **#spyware #rat** JAR malware.

And because there is no isolated rat or spyware, it must be in touch with its server / attacker to get orders from, and of course send the stolen data.

Checking Networking Behavior :

Let's check its networking behavior. I Really like to use Microsoft Network Monitor tool, it's much simpler than wireshark and offers attaching the packets with its origin processes :

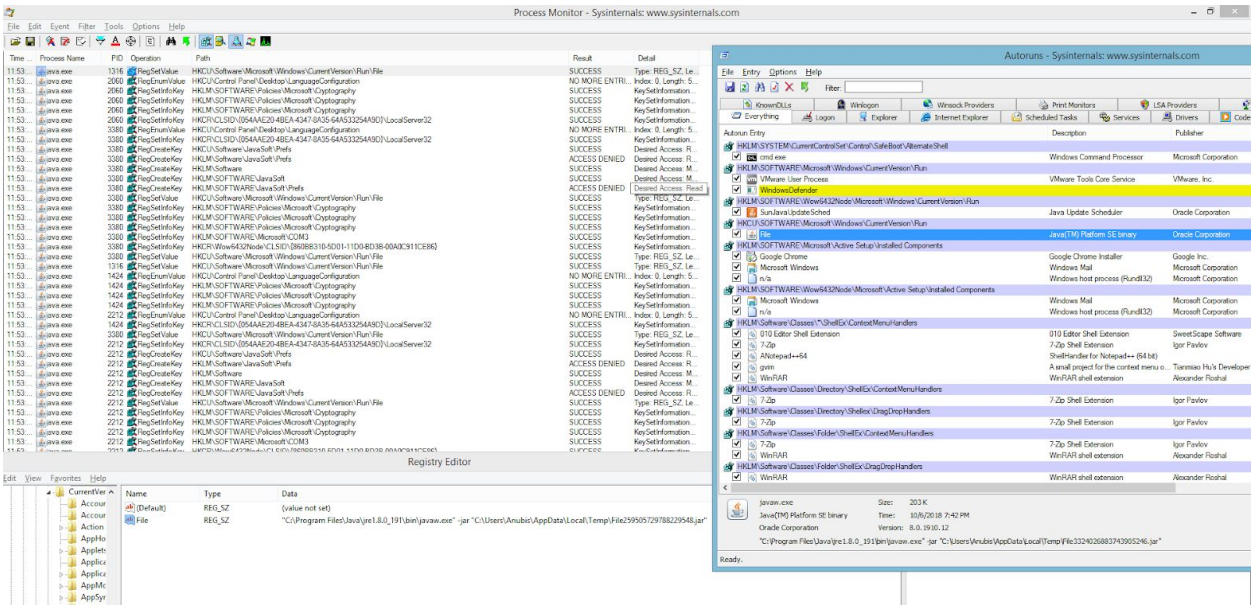


Yes indeed! it contacts with **52.14.36.99** on **port 1336** and sends a lot of TCP traffic but unfortunately I can't decipher its contents.

I searched a lot for the **log file** but I was not able to spot it, maybe the malware sends the data directly over network which I can't see because it's maybe encrypted.

Checking persistence techniques :

Let's check the registry modifications done by the malware :
Using Procmon & Autoruns, we can simply check for any persistence techniques :



We can see it creates multiple keys but the one we are more interested in is **HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run** with value = **"File" and data="C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe" -jar**

> It's using it for persistence and it's a huge indicator / signature for detecting and removing this malware.

Check point : Wrapping up previous notes :

So far we have discovered that

- the JAR malware drops 3 interesting files : one is the actual decrypted copy of it **"Hide it with (attrib +h) command "** + DLL used for hooking keystrokes & mouse events ..etc
  + DLL used for invoking native functions which may be used between the JAR file and the JNativeHook.dll.
- Its Copy contains third party packages used for **spying** and rat **behaviors**.
- Connects with its C2 server with address "**52.14.36.99**" on port "1336".

Sandbox results :

- The sandbox results are strongly supporting our analysis! Labeled as:
Java.Backdoor.Jacksbot #jrat #rat , **also detected by various AV engines**
https://www.virustotal.com/gui/file/dbd9b15d48b43018f8ea123ea6b942680bd682e12608876cbae88d22cd091ea8/detection

- Sets a windows hook :

- "java.exe" sets a global windows hook with filter "WH_KEYBOARD_LL""
- "java.exe" sets a global windows hook with filter "WH_MOUSE_LL"

- Creates mutants :

- "\Sessions\1\BaseNamedObjects\eed3bd3a-a1ad-4e99-987b-d7cb3fcfa7f0 -S-1-5-21-686412048-2446563785-1323799475-1001"
- "eed3bd3a-a1ad-4e99-987b-d7cb3fcfa7f0-S-1-5-21-686412048-2446563785-1323799475-1001"
- "Local__DDrawCheckExclMode__"
- "Local__DDrawExclMode__"
- "\Sessions\1\BaseNamedObjects\Local__DDrawExclMode__"
- "\Sessions\1\BaseNamedObjects\Local__DDrawCheckExclMode__"

- Pattern Matching :

- **YARA signature** "jRat" classified file "File7284155642637045387.jar" as **"rat,jrat"** based on indicators: "META-INF,key.dat,config.dat,a.class,b.class,c.class,r.class,n.class,t.class,h.class,y.class,m.class,o.class,s.class,e.class,d.class,l.class,p.class,i.class,j.class,g.class,f.class,k.class,v.class,w.class,ne.class,ce.class,ge.class,rd.class,pe.class,ib.class,id.class,se.class,le.class,sc.class,rc.class,ef.class,re.class,oc.class,vc.class,ad.class,ve.class,ed.class,ue.class,ld.class,me.class,nd.class,te.class,ke.class,he.class" (Author: Kevin Breen kevin@techanarchy.net)

- **YARA signature** "jar_jrat_g0" classified file "File7284155642637045387.jar" as **"rat,jrat"** based on indicators: "META-INF,config.dat,key.dat,00636f6e6669672e646174,a.class,b.class,c.class,r.class,n.class,t.class,h.class,y.class,m.class,o.class,s.class,e.class,d.class,l.class,p.class,i.class,j.class,g.class,f.class,k.class,v.class,w.class,ne.class,ce.class,ge.class,rd.class,pe.class,ib.class,id.class,se.class,le.class,sc.class,rc.class,ef.class,re.class,oc.class,vc.class,ad.class,ve.class,ed.class,ue.class,ld.class,me.class,nd.class,te.class,ke.class,he.class" (Author: Kevin Breen / jurg)

- **YARA signature** "jar_jrat_g0" classified file "all.bstring" as **"rat,jrat"** based on indicators: "META-INF,key.dat,enc.dat,a.class,r.class,v.class,va.class" (Author: Kevin Breen / jurg)

- **YARA signature** "jar_jrat_g0" classified file "dbd9b15d48b43018f8ea123ea6b942680bd682e12608876cbae88d22cd091ea8.bin" as **"rat,jrat"** based on indicators: "META-INF,key.dat,enc.dat,a.class,r.class" (Author: Kevin Breen / jurg)

Click here for the full report.

Detection and Removal tool :

We can simply detect & remove the malware by its indicators : process, files and registry keys. You can compile this python script “attached“ into executable with pyinstaller if you want.

```
1. import psutil        #library for process manipulation.
2. import re            #library for regular expressions.
3. import os            #library for OS commands like Files, etc.
4. import tempfile      #library for temp directory.
5. import winreg        #library for registry manipulation.
6.
7.
8.
9. def remove(path):    #Removing the file passed as parameter.
10.     path = path.replace("\\", "/")  #Correcting the slash for win file system.
11.     os.remove(path)
12.     print("File at {} was deleted".format(path))
13.
14.
15.
16. def deleteFiles():   #Traversing the temp directory.
17.     tempDir = tempfile.gettempdir()
18.     for file in os.listdir(tempDir):
19.         if re.search("File(\d)+.jar", file) or re.search("JNativeHook_(\d)+.dll",
    file): #Using regex to search for the malware files.
20.             d = os.path.join(tempDir,file)
21.             remove(d)
22.
23.
24. def deleteRegistry():       #Removing the registry values.
25.     regKey = r"Software\Microsoft\Windows\CurrentVersion\Run"
26.     regValue = "File"
27.        #Opening the hkey & returns handle.
28.     hKey = winreg.OpenKey(winreg.HKEY_CURRENT_USER, regKey, 0, winreg.KEY_ALL_ACCESS)
29.     i = 0
30.     while True:
31.         try:
32.             if regValue == winreg.EnumValue(hKey, i)[0]:  #value matched “File”.
33.                 print("Found Malware registry value")
34.                 winreg.DeleteValue(hKey, regValue)
35.                 print("Deleted Malware registry value")
36.                 break
37.             i += 1
38.         except:
39.             break
40.     winreg.CloseKey(hKey)   #if hkey is not closed using this method, it is closed when
41.                             #the hkey object is destroyed by Python.
42.
```

```
43.
44.
45.
46. for proc in psutil.pids(): #Returns a list of running process PIDs.
47.     try:
48.         p = psutil.Process(proc)
49.         try:
50.             files = p.open_files()       #Returns regular file as a list of tuples.
51.             for f in files:
52.                 if re.search("File(\d)+.jar", f.path):
53.                     try:          #Using regex to search for the malware files.
54.                         p.kill()
55.                         print("Malware Process with PID {} was killed".format(proc))
56.                     except:
57.                         print("Unable to kill the process")
58.                         exit(1)
59.         except:     #Access denied
60.             continue
61.     Except:         #No process with PID
62.         continue
63.
64. deleteFiles()
65. deleteRegistry()
```