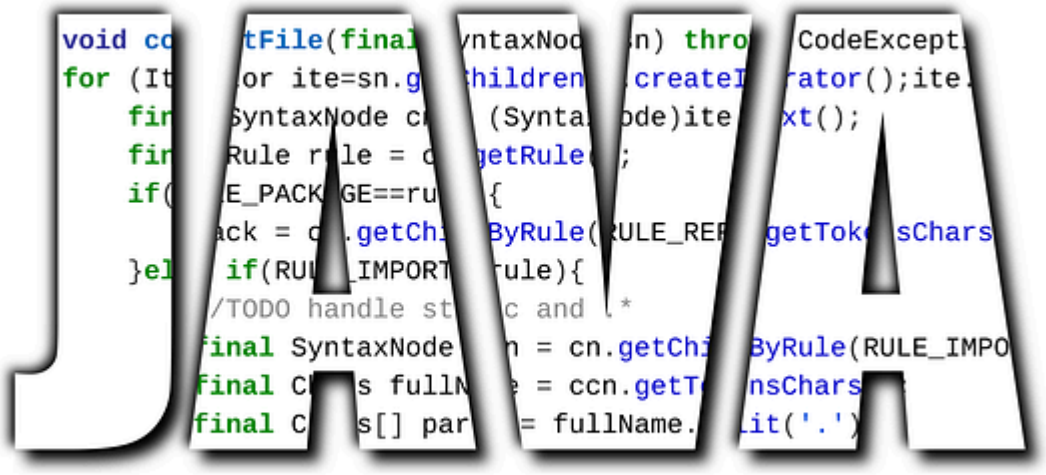
Evolution of jRAT JAVA Malware

By Pavankumar Chaudhari

Published: 2017-10-03 · Archived: 2026-10-25 02:45:14 UTC



03 October 2017



jRAT (Java Based Remote Access Trojans) malware is not new but its activity has increased recently in the last few months and they are targeting various organizations. Per day, Quick Heal Security Labs identifies thousands of spam emails carrying weaponized JAR as attachments targeting users. These malicious JAR i.e., jRAT malware upon execution, result in infecting users. Let’s take a look at a detailed analysis of this malware.

Infection chain

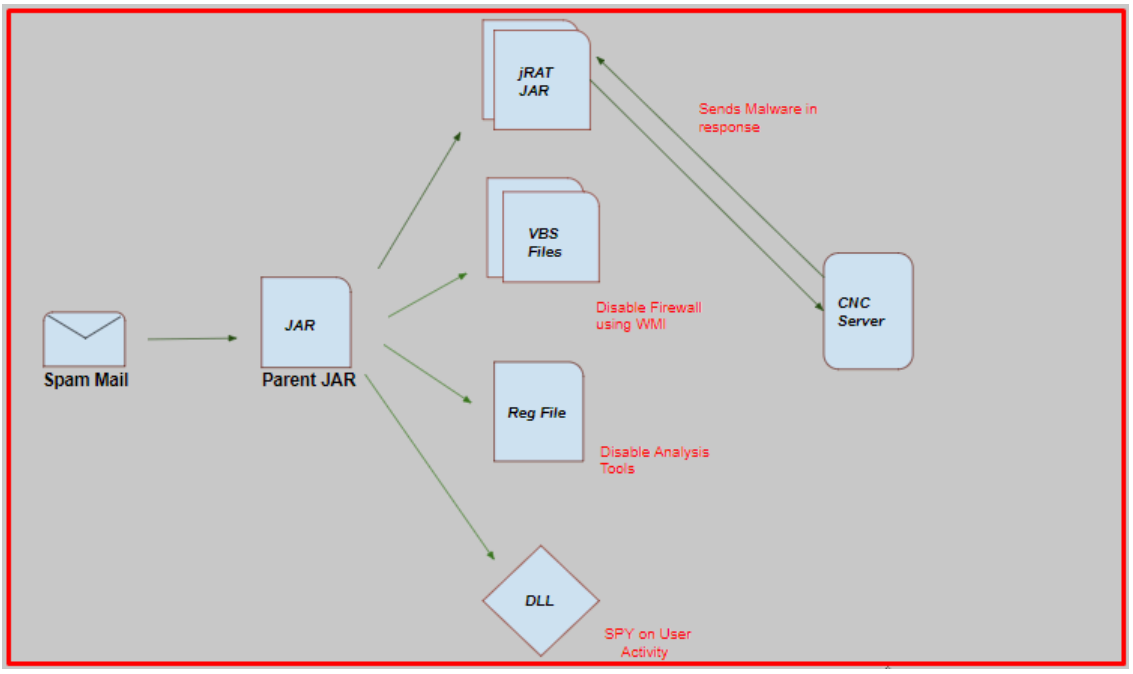


Fig 1: jRat Infection Chain

Here is an initial infection vector which is a spam email. As it looks legitimate, the user is tempted to download and open the attachment.

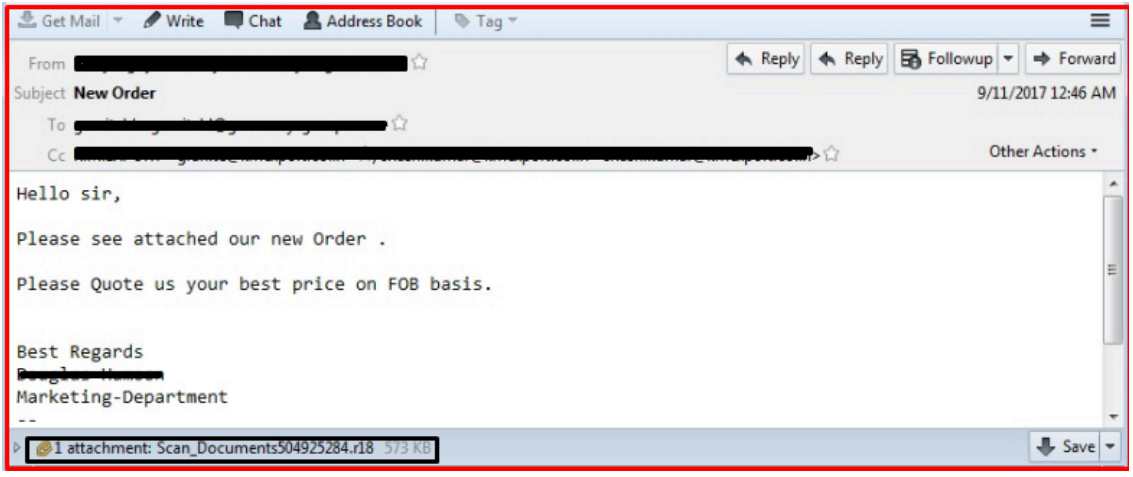


Fig 2: Spam Email

Below is a list of attachment names observed in these spam emails:

- ITD_EFILING_FORM15CB_PR3.2.jar
- MVD_SHPMNT_VSL_0004048_pdf.jar
- Payment Swift Scan Copy 682017.pdf.jar
- SHIPPING DOCUMENTS PDF.jar
- SCAN DOC- 53862100.jar
- FINAL COMPLETE SET OF SHIPPING DOCS.jar
- PAYMENT_ADVISE_PDF.jar
- PAYMENT_APLICATION_PDF.jar

After extracting the parent JAR file, it shows some java packages containing some long random filenames which contain raw data and class files. We have observed that malware actors are evolving malicious JARs with numerous obfuscations patterns. Some of the patterns are as follows:

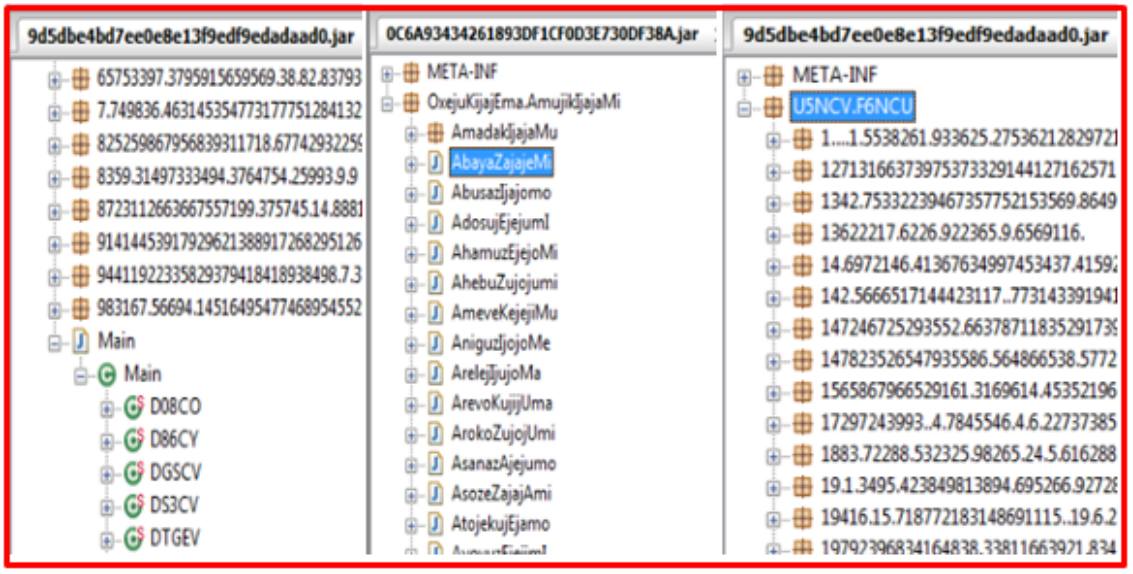


Fig 3: Different Obfuscation Patterns

Well-known decompilers failed to decompile the parent JAR file. Due to variation in obfuscation and encryption, (RSA, AES) makes static analysis more complex. Let's see how malware behaves when it gets executed.

Execution of the parent JAR file drops two ".vbs", two ".Class", one ".Reg" and one ".dll" file at "%TEMP%" location. Every dropped file has a unique role in the infection cycle. Also, the parent JAR checks for a virtual machine using **GlobalMemoryStatusEx()** api which checks for the total physical and virtual memory available.

The parent JAR drops VBS files at %Temp% location with some random names. Also, it drops a JAR file with extension '.class' at %Temp%. The parent JAR executes dropped the JAR file. The dropped JAR file is a jRAT malware.

The below images shows dropped vbs files:

```
Set oWMI = GetObject("winmgmts:{impersonationLevel=impersonate}!\\Aroot\SecurityCenter2")
Set colItems = oWMI.ExecQuery("Select * from AntiVirusProduct")
For Each obj Item in colltems
 With obj Item
  WScript.Echo "{""AV"":""" & .displayName & """}"
 End With
Next
```

Fig 4: VBS File to list down AV's

```
Set oWMI = GetObject("winmgmts:{impersonationLevel=impersonate}!\\Aroot\SecurityCenter2")
Set colItems = oWMI.ExecQuery("Select * from FirewallProduct")
For Each obj Item in colltems
 With obj Item
  WScript.Echo "{""FIREWALL"":""" & .displayName & """}"
 End With
Next
```

Fig 5: VBS File to list down Firewall Products

Then jRAT malware executes VBS files using **cscript.exe.**

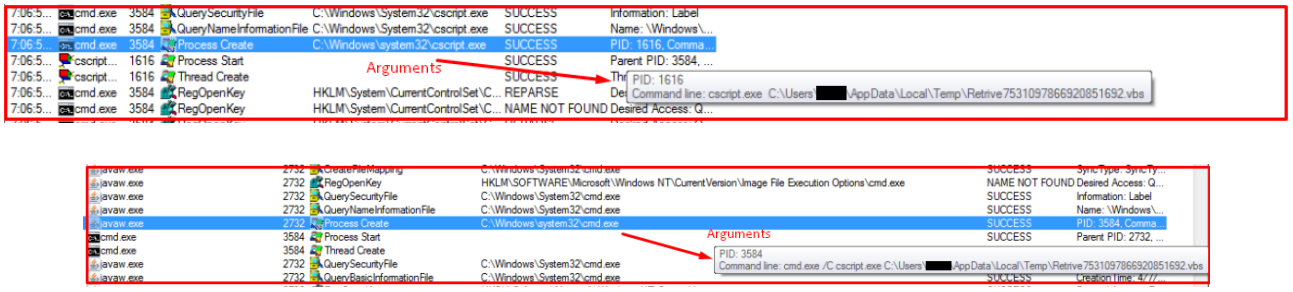


Fig 6: VBS File Execution by JAR

One of the VBS files enumerates a list of different firewall installed using WMI (Windows Management Instrumentation) functionality and the other one enumerates a list of third-party antivirus products using the same functionality upon execution.

The parent JAR also drops the ".Reg File" at %Temp% location and executes it using 'reg.exe'. It creates registry entries of frequently used analysis tools such as 'Procexp.exe' ,'wireshark.exe', 'dumppcap.exe' and some security products processes under "**Image File Execution"**. So, if any process started and if it has an entry under that key then the process gets killed.

Some registry entries are shown below:

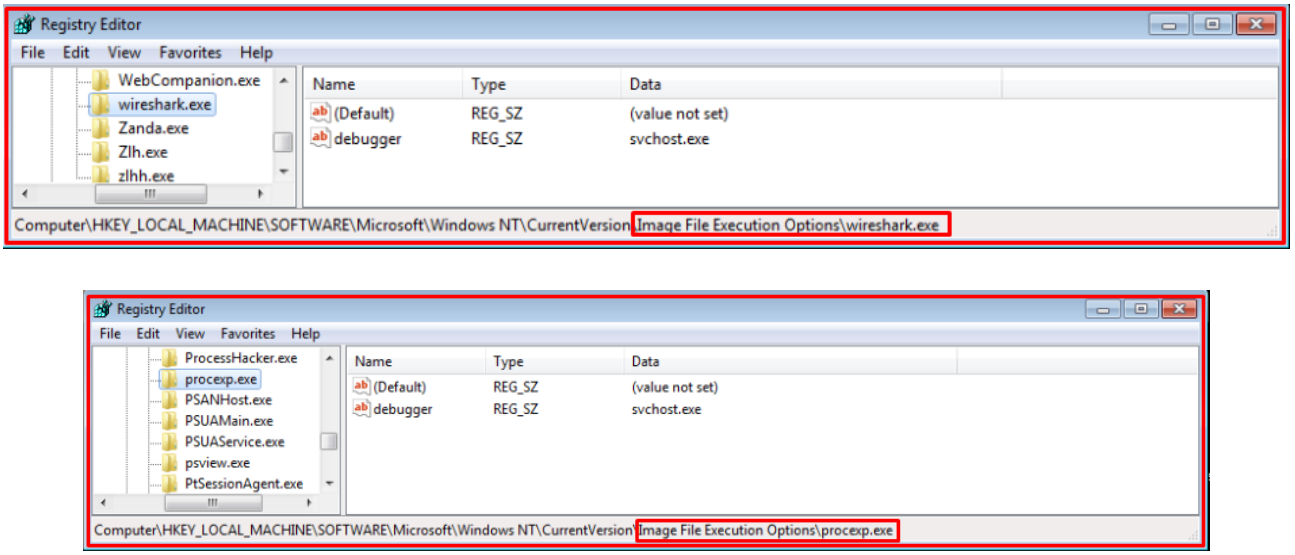


Fig 7: Registry Entry under Image File Execution

The parent JAR executes actual jRat JAR file using java.exe. This jRat file is capable of communicating with a C&C server and can download executable payload.

To achieve persistence, it makes an entry into an auto-run registry so it can launch itself when the system reboots.

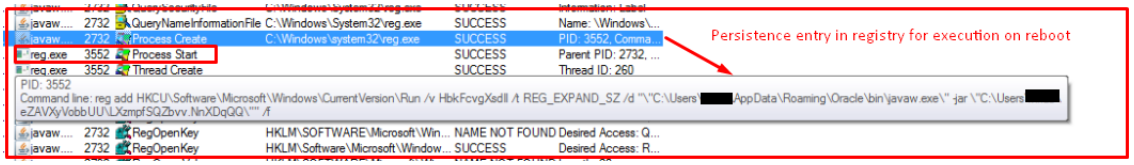


Fig 8: Persistence Entry in Registry

jRat connects with CnC IP "213.183.58[.]42 ". The below image shows the TLS-encrypted SSL traffic after infection. After decoding the TCP stream on port 3012, we found the blacklisted certificate which is associated with jRAT JAR.

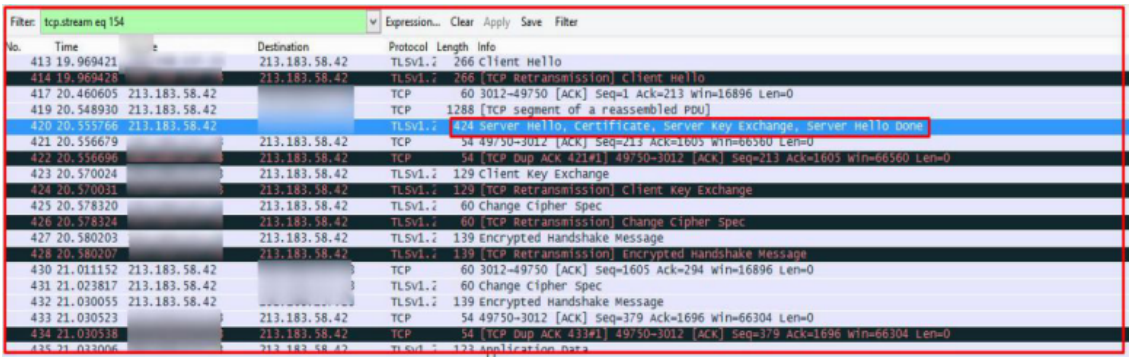


Fig 9: TLS-Encrypted SSL Traffic

Below image shows the SSL certificate information and it has an entry in the SSL Blacklist:



Fig 10: SSL Certificate Information

Seqrite detection stats

Seqrite Email Security generically detects such malicious attachments.

Detection Name: **JAR.Suspicious.A**

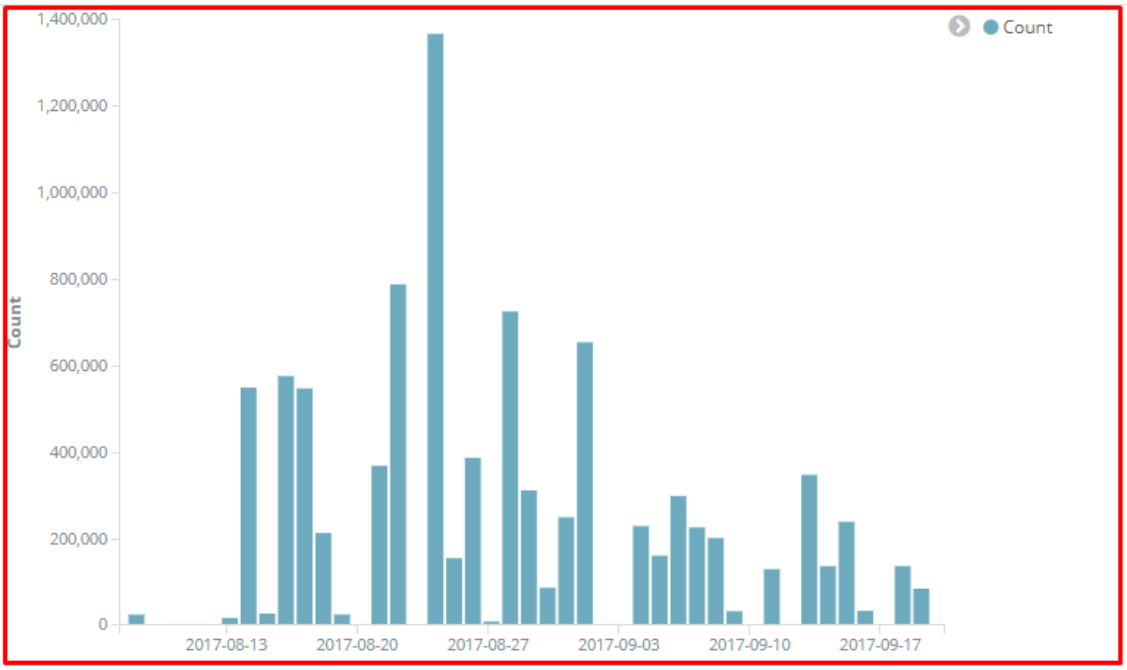


Fig 11: Quick Heal Lab detection stats

After analysing the entire jRAT JAR infection chain, we noticed that the malware authors have been consistently changing the obfuscators to evade signature-based detection and using anti-debugging and anti-VM's techniques. Furthermore, finding and disabling security solutions using numerous ways show how it got evolved.

Indicators of compromise

- 213.183.58.42
- 781FB531354D6F291F1CCAB48DA6D39F
- 0B7B52302C8C5DF59D960DD97E3ABDAF
- 938CF1BA5F8BDB516B5617826E0B08A1
- 76985223E94342D0FAB80D8A4DB1707C

Subject Matter Expert

Prashant Kadam, Pawan Chaudhari | Quick Heal Security Labs

Previous PostAttacks on SSL are on the rise: Know more who is hiding in your e...

Next Post Cybersecurity: New threats taking off in the Aviation Industry



Pavankumar is associated with Quick Heal Technologies as a Technical Lead (Research and Development) and is also a part of Vulnerability Research and Analysis Team....

Articles by Pavankumar Chaudhari »

Related Posts

- 

- 

- 

Source: https://blogs.seqrite.com/evolution-of-jrat-java-malware/