

# STRRAT: Malware Analysis of a JAR archive - ANY.RUN's Cybersecurity Blog

By khr0x

Published: 2023-04-14 · Archived: 2026-04-05 20:26:22 UTC

The majority of malware on Windows OS is compiled executable files. And their popularity has led to a blockage at the delivery stage to the user. Fortunately, antivirus software on users' PCs is good at detecting and blocking the malicious payload contained in these files.

But malware developers use various tricks to overcome this issue: hackers develop a program using other (less popular) file formats. One of them is JAR.

In this article, we will talk about one of the Java malware representatives – STRRAT. Follow along with our detailed behavior analysis, configuration extraction from the memory dump, and other information about a JAR sample.

## What is a malicious Java archive?

A JAR file, a Java archive, is a ZIP package with a program written in Java. If you have a Java Runtime Environment (JRE) on your computer, the .jar file starts as a regular program. But some antivirus software may miss such malware, as it is not a popular format, but it can be easily analyzed in an [online malware sandbox](#).

Let's look at STRRAT, a trojan-RAT written in Java. Here are typical STRRAT tasks:

- data theft
- backdoor creation
- collecting credentials from browsers and email clients
- keylogging

The initial vector of STRRAT infection is usually a malicious attachment disguised as a document or payment receipt. If the victim's device has already had JRE installed, the file is launched as an application.

 A JAR archive: Shipment 08-24-2022.jar

## How to analyze STRRAT's Java archive

STRRAT usually has the following execution stages:

1. The icacls launch to grant permissions
2. Running a malware copy in the C:\Users\admin folder
3. Persistence via schtasks
4. Running a malware copy in the C:\Users\admin\AppData\Roaming folder

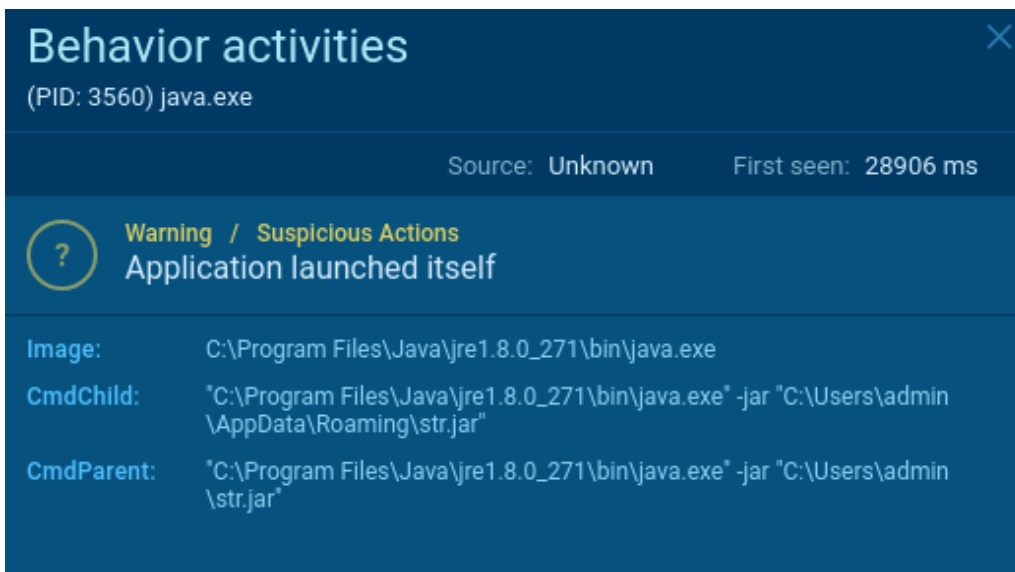
5. Collecting and sending data to the server specified in the program

You can monitor this pattern of malware behavior in the [STRRAT sample](#):



## A JAR file replication

Replication is the first thing that catches your eye. We run the object from the desktop, then STRRAT creates a copy of the file: first in the C:\Users\admin folder and then in C:\Users\admin\AppData\Roaming. After that, they run consistently.



## A Java file gets file access

The next step is that the malware uses icacls to control file access. The command grants all users access to the .oracle\_jre\_usage folder:

```
icacls C:\ProgramData\Oracle\Java\oracle_jre_usage /grant "everyone":(OI)(CI)M
```

## Application launch of STRRAT malware

Then malware creates a task in the Scheduler using the command line:

```
schtasks /create /sc minute /mo 30 /tn Skype /tr "C:\Users\admin\AppData\Roaming\str.jar"
```

The task is to use the Task Scheduler to run malware on behalf of the legal Skype program every 30 minutes.



Now let's see the details of the 3504 process:

- Malware changes the autorun value

**Behavior activities** (PID: 3504) java.exe

Source: registry First seen: 29063 ms

**Danger / Installation**  
Changes the autorun value in the registry

Operation:	WRITE
Name:	STR
Value:	"C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe" -jar "C:\Users\admin\AppData\Roaming\str.jar"
Key:	HKEY_CURRENT_USER\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\RUN
TypeValue:	REG_SZ

- it writes malware into the startup menu

**Behavior activities** (PID: 3504) java.exe

Source: files First seen: 27422 ms

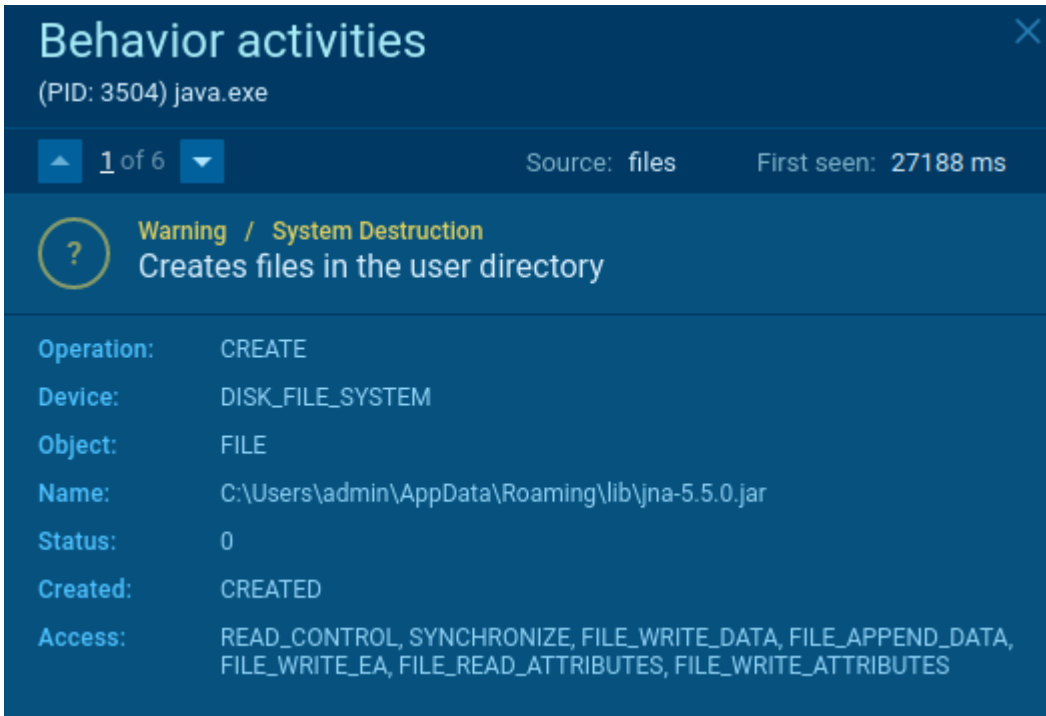
**Danger / Installation**  
Writes to a start menu file

Operation:	WRITE
Device:	DISK_FILE_SYSTEM
Object:	UNKNOWN TYPE
Name:	C:\Users\admin\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\str.jar
Status:	0
Created:	NONE

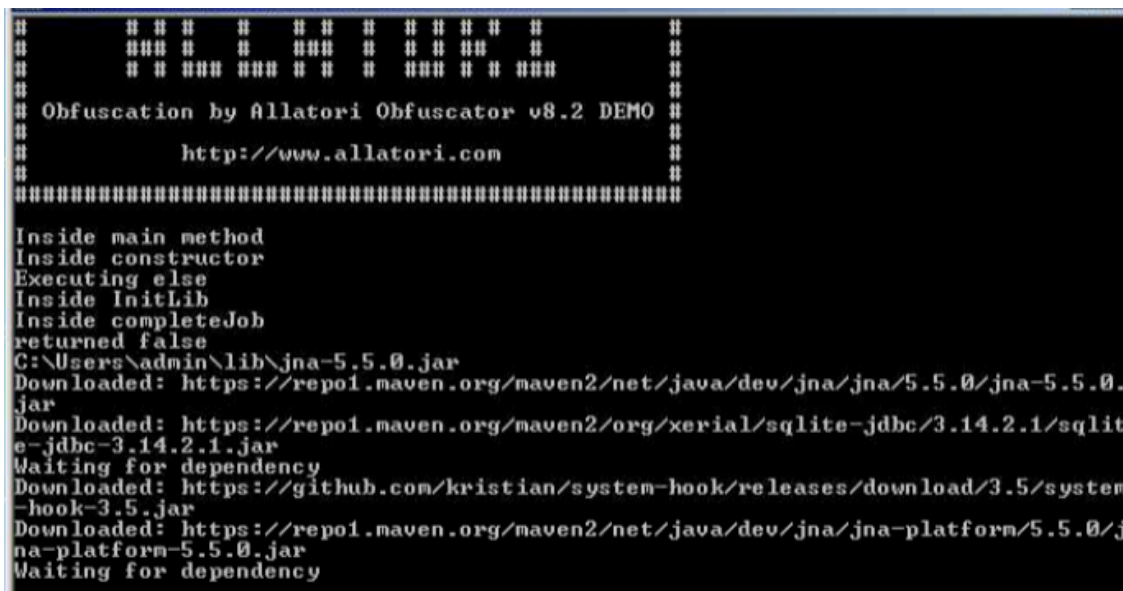
So we can expect STRRAT to launch again after the OS reboot.

### File creation of JAR malware

STRRAT's process creates additional JAR files downloaded from public repositories.



The trojan downloaded and then created the library files from the Internet. If you run the malware through CMD, you can see them yourself. And this scenario is quite unusual – we can find the program execution logs if malware is run with CMD.



### STRRAT network traffic analysis

[ANY.RUN online malware sandbox](#) provides detailed information about Network traffic in the Connections tab.



Go to the files tab to see that the library files are loading, which is necessary for further malware execution.



STRRAT downloads the following JAR libraries:

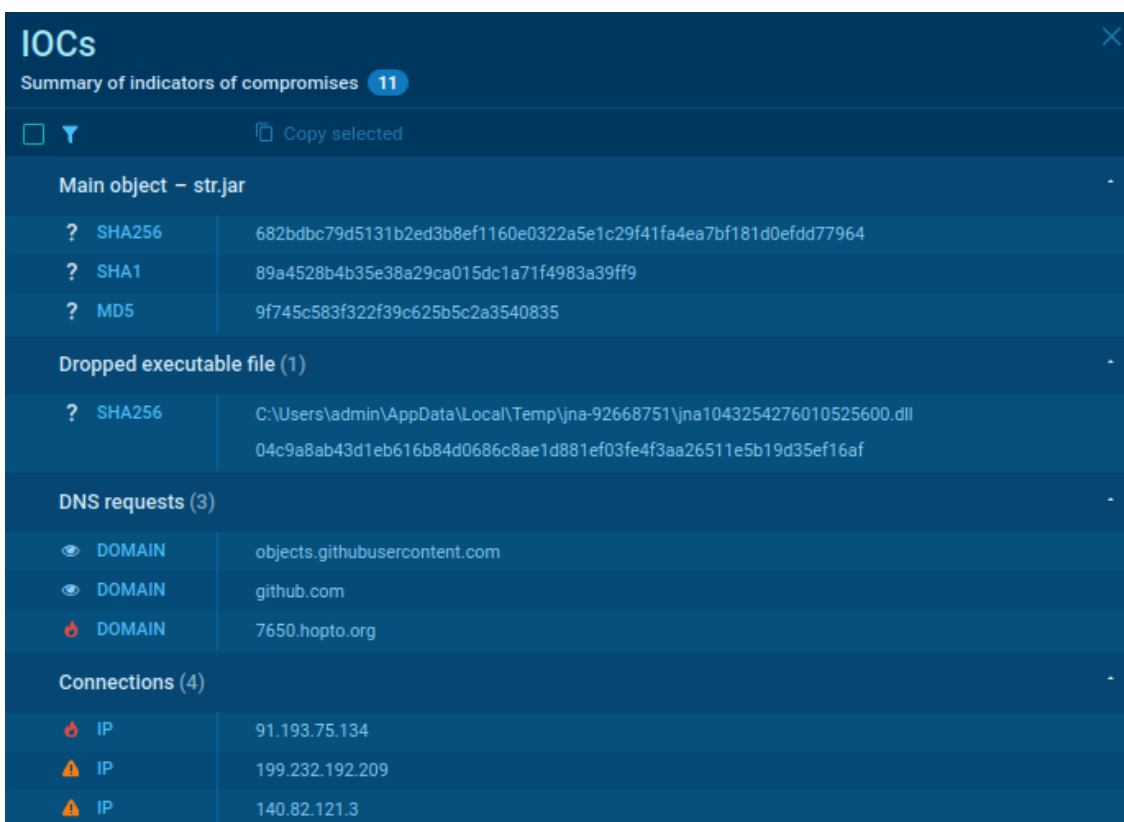
- jne
- sqllite
- system-hook

Besides data transferring, we can notice the constant attempts to connect with the 91[.]193[.]75[.]134 IP address.



## Malicious Java archive's IOCs

The significant part of the analysis is that you can get IOCs very fast.



Main object – str.jar	
SHA256	682bdbc79d5131b2ed3b8ef1160e0322a5e1c29f41fa4ea7bf181d0efdd77964
SHA1	89a4528b4b35e38a29ca015dc1a71f4983a39ff9
MD5	9f745c583f322f39c625b5c2a3540835

Dropped executable file (1)	
SHA256	C:\Users\admin\AppData\Local\Temp\jna-92668751\jna1043254276010525600.dll 04c9a8ab43d1eb616b84d0686c8ae1d881ef03fe4f3aa26511e5b19d35ef16af

DNS requests (3)	
DOMAIN	objects.githubusercontent.com
DOMAIN	github.com
DOMAIN	7650.hopto.org

Connections (4)	
IP	91.193.75.134
IP	199.232.192.209
IP	140.82.121.3

## How to extract STRRAT malware configuration

To retrieve the malware configuration, we use PH and find all lines. Then filter them by the address we already know in Connections.

As a result, we find only one interesting string.

```
0x30b0c 34 socket://91.193.75.134:7650
0x30bd124 26 91.193.75.134
0x30bd184 36 91.193.75.134:7650
0x30bd1d4 40 //91.193.75.134:7650
0x3e492d4 132 91.193.75.134[7650]http://frost.live/strigo/server/?hwid=1&ld=m&ht=5[7650.hopto.org]7650[true][true][0U4Q-MOEM-D68I-FFUC-9LY4
0x3e49414 264 91.193.75.134[7650]http://frost.live/strigo/server/?hwid=1&ld=m&ht=5[7650.hopto.org]7650[true][true][0U4Q-MOEM-D68I-FFUC-9LY4
0x3e49584 26 91.193.75.134
0x405b294 26 91.193.75.134
0x40f0a7c 26 91.193.75.134
```

Brief string analysis shows that it contains separators in the form of “vertical dashes,” different configuration parameters:

- address
- port
- URL link

Additional options include:

- 2 places where malware needs to install itself (Registry and StartconfigurationSkype task)
- proxy
- LID (license)

These data are included in the configuration we are looking for.

The line of interest is located in the heap area of memory. Let’s extract a dump of it and write a simple Python extractor. Try to extract it by yourself with the [STRRAT malware configuration script](#) that we have shared with you. If you use the code, this is the output data you should get:

```
{'C2': '91.193.75.134', 'Port': '7650', 'URL': 'http://jbfrost.live/strigoj/server/?hwid=1&lid=m&ht=5', 'Options': [{'Startup': 'true', 'Secondary Startup': 'true', 'Scheduled Task': 'true', 'Proxy': '7650.hopto.org', 'LID': '0U4Q-MOEM-D6BI-FFUC-9L'}]}
```

And ANY.RUN’s version is already done for you. There is also a much faster way to get the data you need – review malware configurations right in our service, which will unpack the sample from memory dumps and extract C2s for you:

 Data output in ANY.RUN

## To sum it up

We have carried out the analysis of the malware written in JAVA and triaged its behavior in [ANY.RUN online malware sandbox](#). We have written a simple extractor and derived the data. Copy the script of STRRAT and try to extract C2 servers by yourselves and let us know about your results!

ANY.RUN has already done this part for you, and the malware is detected automatically: it extracts the dump, pulls the configuration data, and presents results in an easy-to-read form.

STRRAT, [Raccoon Stealer](#), what’s next? Please write in the comments below what other malware analysis you are interested in. We will be glad to add it to the series!

Check out other malware samples:

<https://app.any.run/tasks/22ca1640-fcd8-4411-9757-8349af4d163f>

<https://app.any.run/tasks/56076b18-886b-46ca-aadb-e1d7d5de62cd>

<https://app.any.run/tasks/25cb57c8-a018-4ec1-bb98-74e5fe30e504>

<https://app.any.run/tasks/4ed8f7b5-e173-4011-b7fd-08f1bdf40e>

 [ANY.RUN malware analyst](#)

**khr0x**

I'm 21 years old and I work as a malware analyst for more than a year. I like finding out what kind of malware got on my computer. In my spare time I do sports and play video games.

I'm 21 years old and I work as a malware analyst for more than a year. I like finding out what kind of malware got on my computer. In my spare time I do sports and play video games.

---

Source: <https://any.run/cybersecurity-blog/strat-malware-analysis-of-a-jar-archive/>