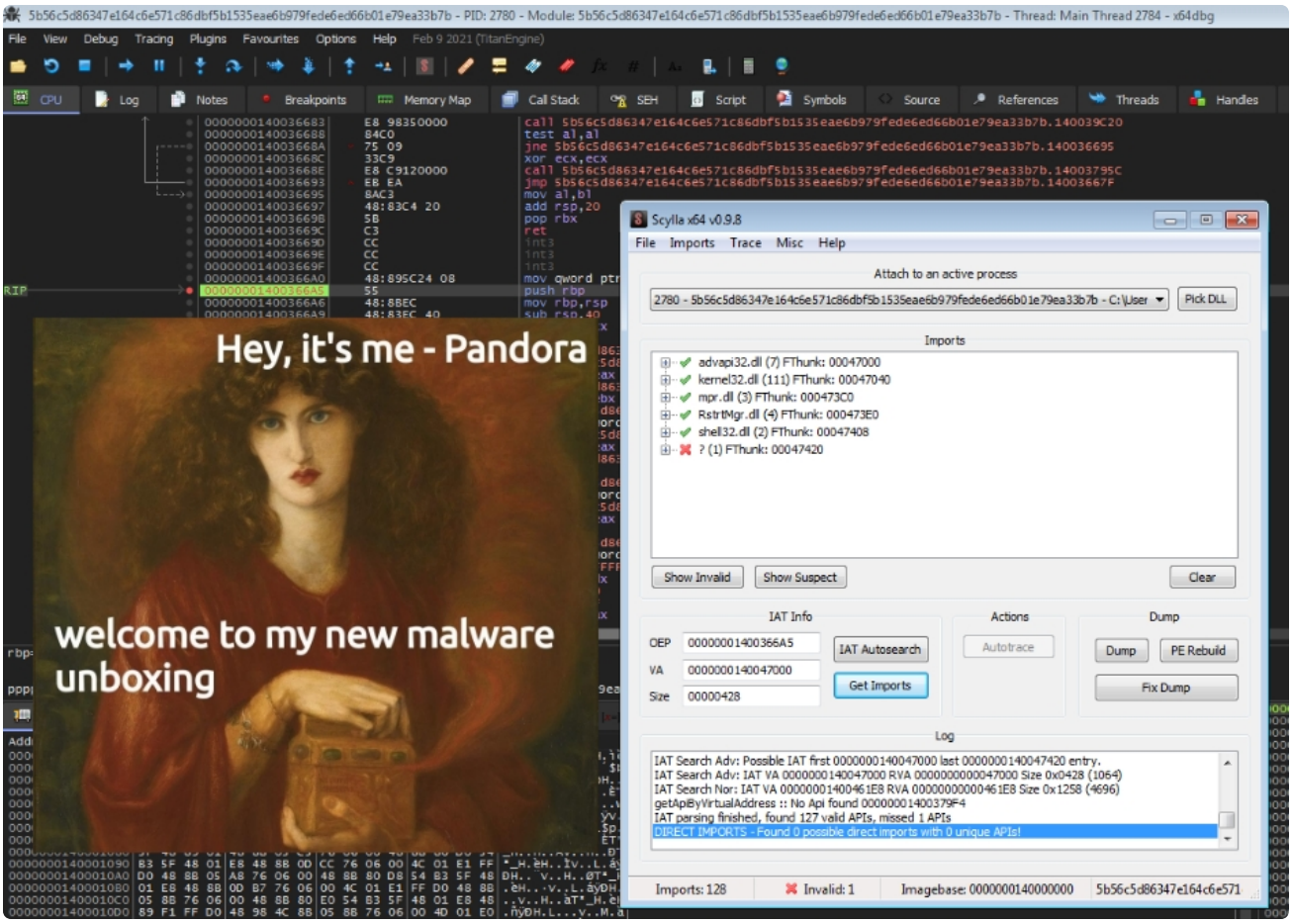
Quick revs: Pandora Ransomware - The Box has been open for a while...

dissectingmalwa.re/blog/pandora/

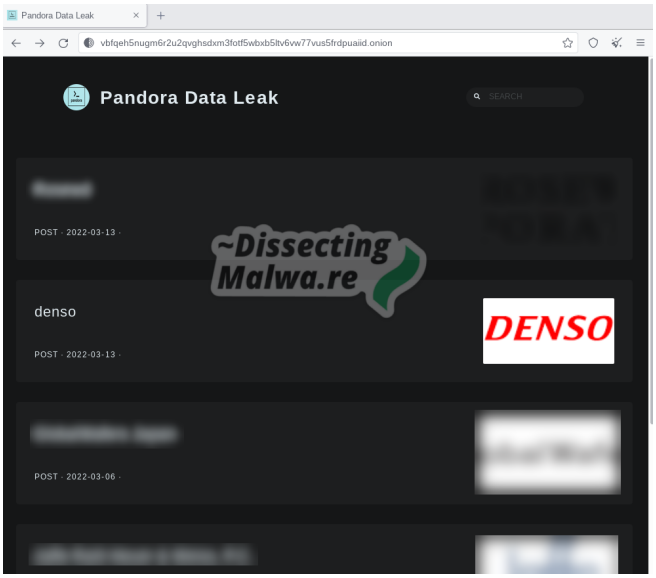


Hey there, I'm finally getting around to introducing the new post category "Quick revs", which will feature short write-ups of various malware analysis and reverse engineering topics. This will allow me to post more frequently, since I don't always have to time to write deep-dive reports in my limited free time.

Today we are going to be looking at "Pandora Ransomware", a novel Ransomware strain that has been monitored for a couple of days, e.g. by MalwareHunterTeam, but at first no sample was available.

At the time of writing Pandora is claiming on their Leak site to have compromised four companies, one of which is the japanese automotive OEM Denso, which has been covered extensively in the media. I'm bringing up Denso here, because they were compromised by *Rook* Ransomware a few months earlier, which beggs the question if the attackers

somehow were able to maintain access and just rebranded from *Rook* to *Pandora*. Of course this is just speculation on my part and I don't consider signifficant similarities in the ransomware samples of both strains as sufficient proof either, but in my opinion one could shed light on this relation by investigating their TTPs and other details of the intrusions.



On the 14th of March 2022 the Pandora sample below was obtained by vx-underground:

Pandora Ransomware (packed)

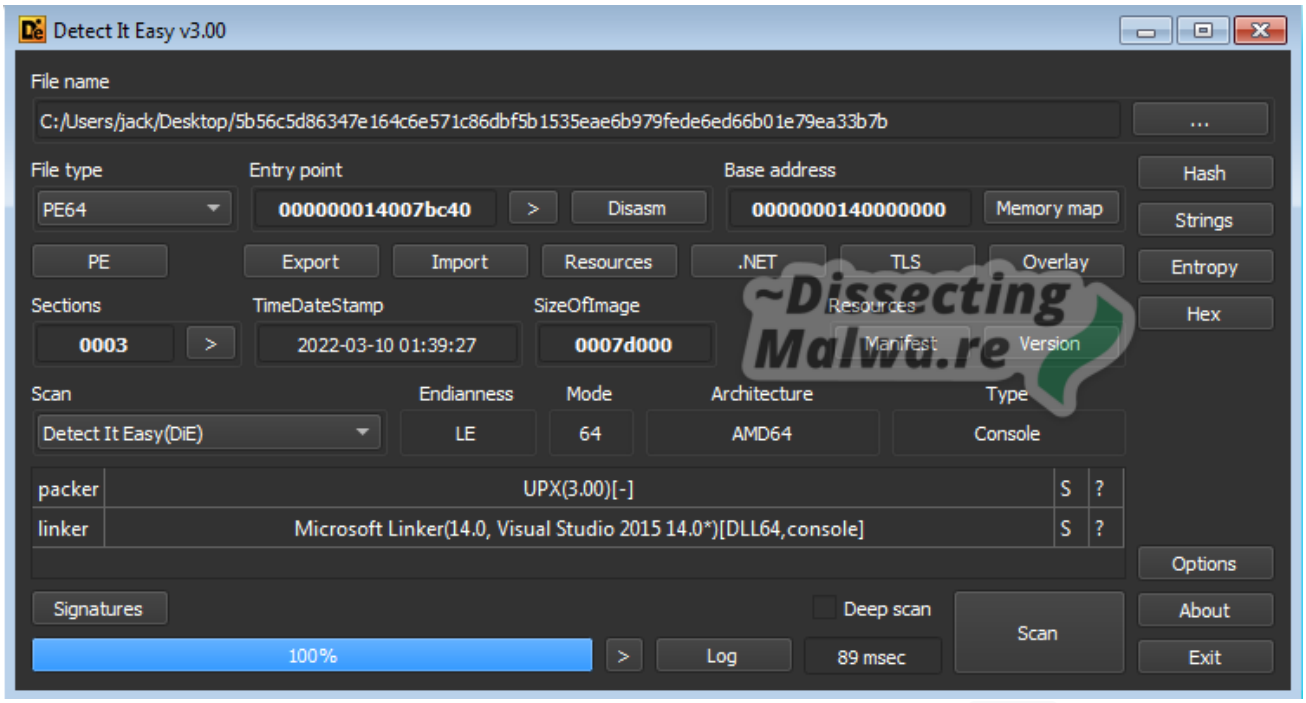
```
Original file names: "1vfrk1jrt.dll", "M3DO2.exe"
File size: 223232 bytes
Architecture: x64
MD5: 0c4a84b66832a08dccc42b478d9d5e1b
SHA-1: 160320b920a5ef22ac17b48146152ffbef60461f
SHA-256: 5b56c5d86347e164c6e571c86dbf5b1535eae6b979fede6ed66b01e79ea33b7b
```

Download: vx-underground | Malware Bazaar | VirusTotal

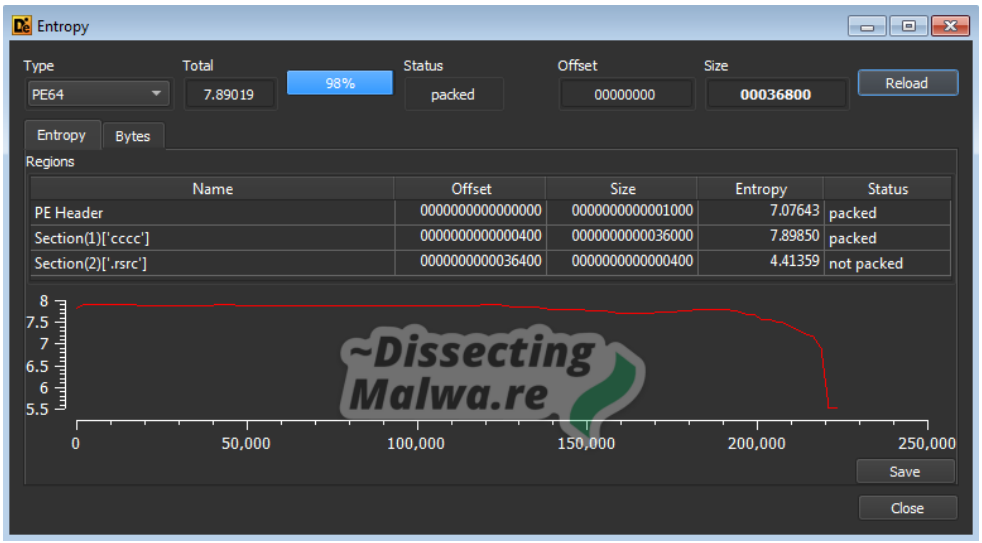
I already tweeted about this sample, but since I got a few questions regarding the unpacking process and similarities to other ransomware strains (specifically Rook and NightSky) I thought I should write it down in a blog post.

Unpacking

After the initial assessment of the sample with *Detect it Easy* a signature for UPX Version 3 was found. Packer detections in *Detect it Easy* should always be taken with a grain of salt, but it gives us a first hint as to what to look for in the next steps.



Looking at the Entropy graph we can see that we have one section (`cccc`) with a very high value, which indicates it contains packed code and a `.rsrc` section with a significantly lower value.



Switching over to *pestudio Pro* since its section layout is a lot cleaner than the one in *Detect it easy* we can see that there is another section called `pppp` which is virtualized and therefore has a raw-size of 0 bytes. This section layout closely resembles the one used by UPX. UPX0 (`pppp` in this case) is the empty section where the compressed contents of UPX1 (`cccc`) will be decompressed to by the unpacking program stub. At this point I am fairly confident to say that is packer is based on UPX, but looking at the section names they likely messed around with their UPX version.

property	value	value	value
name	pppp	cccc	.rsrc
md5	n/a	29FE31CEC867A1A57006F5C...	492EA09D908FE2C21EB468D...
entropy	n/a	7.898	4.412
file-ratio (99.54%)	n/a	99.08 %	0.46 %
raw-address	0x00000400	0x00000400	0x00036400
raw-size (222208 bytes)	0x00000000 (0 bytes)	0x00036000 (221184 bytes)	0x00000400 (1024 bytes)
virtual-address	0x0000000040001000	0x0000000040046000	0x000000004007C000
virtual-size (507904 bytes)	0x00045000 (282624 bytes)	0x00036000 (221184 bytes)	0x00001000 (4096 bytes)
entry-point	-	0x0007BC40	-
characteristics	0xE0000080	0xE0000040	0xC0000040
writable	x	x	x
executable	x	x	-
shareable	-		-
discardable	-	-	-
initialized-data	-	x	x
uninitialized-data	x	-	-
unreadable	-	-	-
self-modifying	x	x	-
virtualized	x	-	-
file	n/a	n/a	n/a

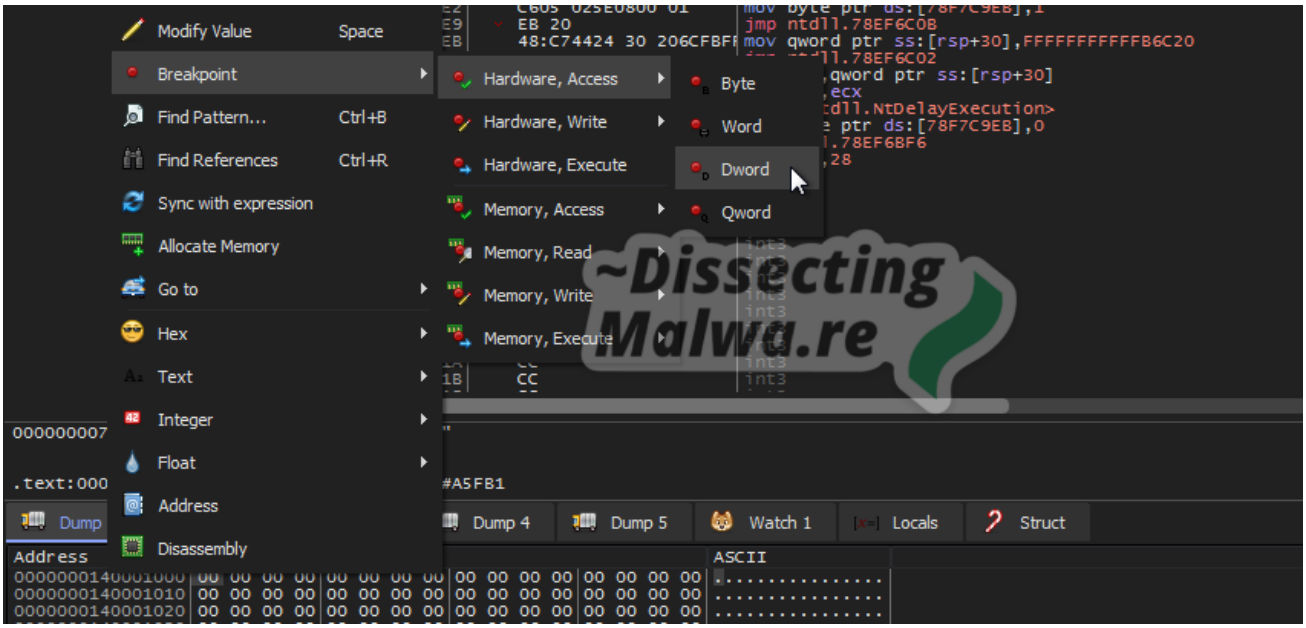
A very simple way to test if we are dealing with a modified version of UPX is to just try to decompress the sample with the vanilla UPX utility. As you can see below it does not decompress! UPX is even telling us that the file was likely messed with. I was able to identify the following modifications to the UPX packer:

- altered section names (as we noticed before)
- old version of UPX / altered version number in the leading header
- missing/overwritten 12-byte trailing header

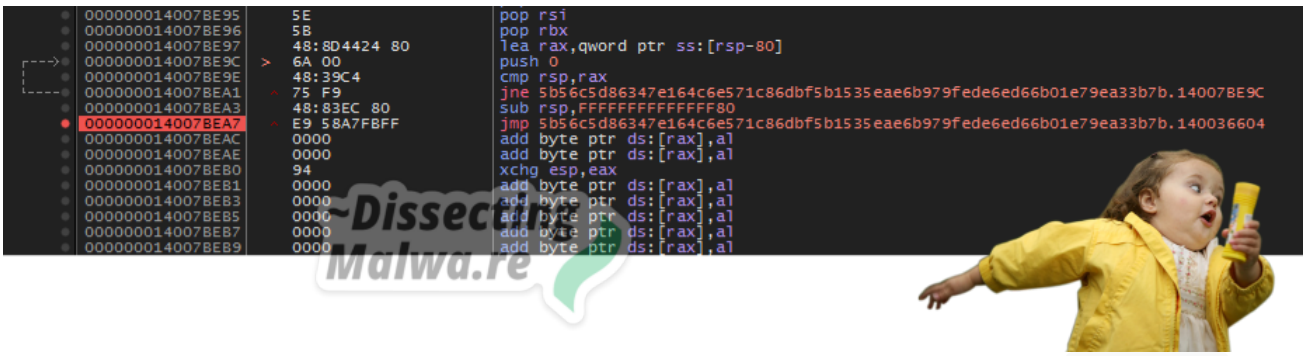
```
f0wl@mw-Lab:~/Tools/upx-3.96-amd64_linux$ ./upx -d ~/Malware/samples/Ransomware/Pandora/sample/5b56c5d86347e164c6e571c86
dbf5b1535eae6b979fede6ed66b01e79ea33b7b
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2020
UPX 3.96        Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
upx: /home/f0wl/Malware/samples/Ransomware/Pandora/sample/5b56c5d86347e164c6e571c86dbf5b1535eae6b979fede6ed66b01e79ea33b
7b: CantUnpackException: file is possibly modified/hacked/protected; take care!

Unpacked 0 files.
```

Alright, as the simple approach does not work and retdec also falls through for now, we'll have to unpack it manually. First of all I'll switch to the *Memory Map* view and *Follow in Dump* on the `pppp` section (UPX0) to monitor it.

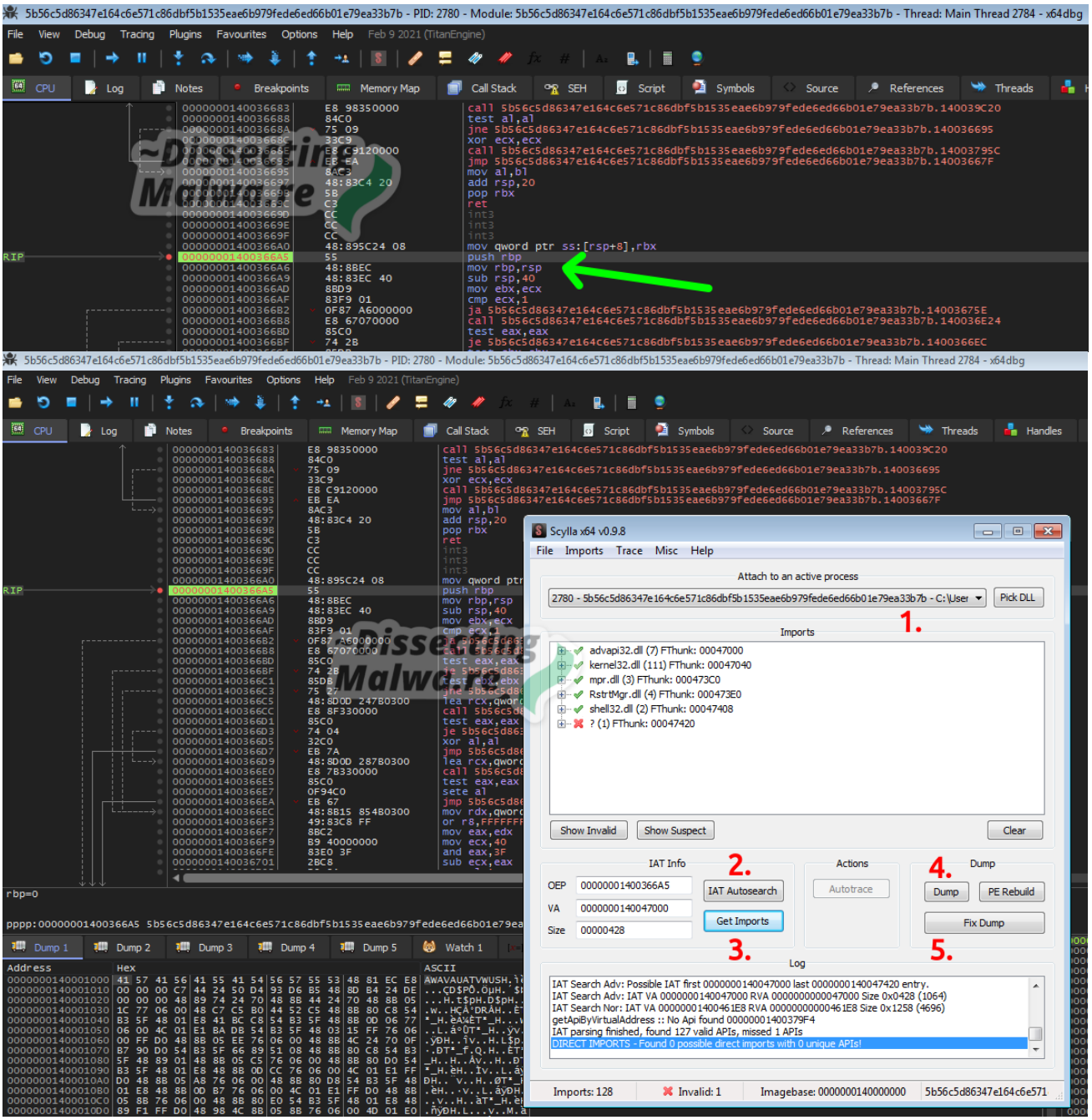
Up next I'm placing a Hardware Breakpoint on Access on the `pppp` section, so we can see when the data is decompressed into it. Hit this breakpoint once or twice to check that it is working and we'll continue onto the next step.



Scroll down until you see the end of the stub with the two jumps followed by junk instructions for padding. The last jump instruction is the so-called *tail jump*, which will transfer to the Original Entrypoint (**OEP**). I'll place a breakpoint on the tail jump to make sure Pandora doesn't run away and potentially encrypt the VM :D Once we hit this breakpoint we can check in the dump of `pppp` that the section should be filled now, so let's jump in!



After following the tail jump we can scroll down a bit again to find the OEP (`push rbp`) and place a breakpoint there. Get ready to dump it like it's hot 🔥



Fire up Scylla and make sure that the correct process is selected (1). After that we'll run the *IAT Autosearch* (2) and *Get Imports* (3) to show them in the textbox above (notice that they significantly differ from the functions imported by the UPX unpacking stub). Finally dump the process to disk (4) and fix the dump (5) to complete the unpacking process. Congratulations to the ones playing along at home, you are now able to manually unpack UPX (and it works for x86 binaries as well).

If you want to skip this step of the analysis you can also download my unpacked sample below:

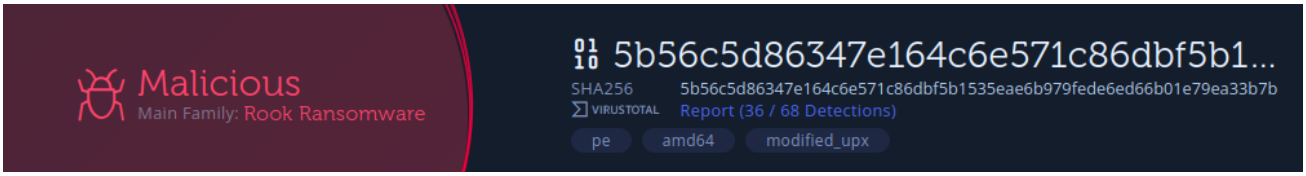
Pandora Ransomware (unpacked)

```
File size: 509440 bytes
Architecture: x64
MD5: 511501033ca23754113686ac70f629db
SHA-1: 26a02a149aca6a8a43e2dca5c75a6360cfe54c50
SHA-256: 2c940a35025dd3847f7c954a282f65e9c2312d2ada28686f9d1dc73d1c500224
```

Download: Malshare | VirusTotal

Similarities with Rook Ransomware

According to the automated analysis by Intezer the Pandora sample from above is related to Rook Ransomware. Since Rook is based on the leaked source code of Babuk Ransomware so is Pandora probably.



As I already mentioned I'm not planning to do a deep-dive analysis of the features of Pandora, so we'll just try to do a high-level comparison between Pandora and Rook. If you are looking for a very in-depth analysis of Rook Ransomware, check out Chuong Dong's post about it.

Rook Ransomware

```
Original file names: "unknown", "7NM2J.txt"
File size: 174080 bytes
Architecture: x64
MD5: bec9b3480934ce3d30c25e1272f60d02
SHA-1: 104d9e31e34ba8517f701552594f1fc167550964
SHA-256: f87be226e26e873275bde549539f70210ffe5e3a129448ae807a319cbdcf7789
```

Download: vx-underground | Malware Bazaar | VirusTotal

Since this sample of Rook Ransomware is also packed with a modified version of UPX (which differs from the one used for Pandora though) I manually unpacked this sample as well using the process described above. You can download the unpacked sample here:

Rook Ransomware (unpacked)

```
File size: 415744 bytes
Architecture: x64
MD5: afdf739eb186e2ec8088b008797d1f6d
SHA-1: f611c2976ebb080214eddd905d30628230f2280d
SHA-256: ebfdee6e5fe2aa5699280248a5e7b714ca18e5bfd284cac0ba4fb88ccbcec5b6
```

Download: Malshare | VirusTotal

Comparing the imported Windows functions of Pandora and Rook we can see the following changes in Pandora (+ = added, - = removed):

```
advapi32.dll:
    - EnumDependentServicesA
    - CloseServiceHandle
    - OpenSCManagerA
    - ControlService
    - QueryServiceStatusEx
    - OpenServiceA

kernel32.dll:
    + GetQueuedCompletionStatus
    + PostQueuedCompletionStatus
    + SetPriorityClass
    + CreateIoCompletionPort
    + SetThreadAffinityMask
    + ResumeThread
    + VirtualFree
    + CreateFileMappingW
    + MapViewOfFile
    + VirtualAlloc
    + UnmapViewOfFile
    + LoadLibraryW
    + VirtualProtect
    + VirtualProtectEx
    + WriteProcessMemory
    - GetTickCount
    - GetModuleFileNameW
    - ExitThread
    - SetFileInformationByHandle
    - ReleaseSemaphore
    - CreateSemaphoreA
    - RaiseException
    - Process32FirstW
    - Process32NextW
    - Sleep
    - CreateToolhelp32Snapshot

mpr.dll:
    - WNetGetConnectionW

shell32.dll:
    - CommandLineToArgvW

shlwapi.dll:
    - PathFileExistsW

user32.dll:
    - wsprintfA
```

From this comparison we can deduct that there have been changes in file handling and thread/process control.

Comparing the strings in both samples I found that Pandora removed the debug messages which are present in the Babuk source and the Rook sample. Additionally the Ransomnote of Pandora Ransomware has been obfuscated whereas the Rook contained it in plain text.

Of course I can't wrap this post up before trying out a new tool, which kind of has become a tradition here. Since the code similarities detected by Intezer are a black box for us we can try and replicate this analysis with `binlex`:

Binlex allows us to extract basic blocks and functions from to samples we feed it as so-called *traits*. In the case of Pandora and Rook these traits contain the re-used code and, with some careful filtering, we can use some of them to build a Pandora-Rook Yara rule. Unfortunately I currently don't have the time to sift through all the extracted traits manually (I don't have a Goodware/Malware Traits Corpus yet to discard traits based on that), but I will get back to this in a few weeks. The long-boi bash command below shows my testing approach in this case, which extracted over 1700 unique (but unfiltered) shared traits from the Pandora and Rook samples.

```
find sim/ -type f | while read i; do binlex -m pe:x86_64 -i $i | jq -r '.[] |
select(.type == "block" and .size < 32 and .size > 8) | .bytes' | sort | uniq; done
| sort | uniq -c | sort -rn
```

One example were I successfully used binlex for a Yara rule a couple of weeks earlier is for my *BlackMatter Ransomware ESXi* rule, which you can find here.

If you would like to give binlex a try I recommend to watch the excellent demo below, which is a recording of a live cooperation between c3rb3ru5d3d53c and OALabs. binlex is very easy to install and well documented, so you should definitely give it a try.

Alright, that should conclude this first look into Pandora Ransomware. I'm sure there will be more in-depth reports about the ransomware itself and the modus operandi of the attackers in the coing days and weeks. As I already mentioned I do not consider the relation "Pandora == Rook" proven based on the findings of this post, but a connection is certainly plausible. Also Pandora will most likely not be the last Ransomware variant based on the Babuk source, since with the leak the metaphorical box cannot be closed again.

I included a small Yara rule for the modified UPX packer below, happy hunting!

Thanks for reading this post and if you have any questions feel free to send me a message :)

Modified UPX Hunting rule

```
import "pe"

rule upx_packer_modified_pandora : Packer {
meta:
   author = "Marius 'f0wL' Genheimer <hello@dissectingmalwa.re>"
   description = "Detects modified UPX packer used by Pandora Ransomware"
   reference = "https://dissectingmalwa.re/blog/pandora/"
   date = "2022-03-16"
   tlp = "WHITE"
   hash = "5b56c5d86347e164c6e571c86dbf5b1535eae6b979fede6ed66b01e79ea33b7b"

strings:
   $header = {33 2E 30 30 00 55 50 58 21} // 3.00.UPX!

condition:
   uint16(0) == 0x5a4d
   and pe.imphash() == "51a8b4c9f41b0c0ca57db63e21505b0d"
   and $header
   and for any i in (0..pe.number_of_sections):(
       pe.sections[i].name == "pppp" and
       pe.sections[i+1].name == "cccc")
   and filesize > 112KB // Size on Disk/2
   and filesize < 1MB // Size of Image*2
}
```