

# Looking Inside Pandora's Box

: 4/7/2022

---

## FortiGuard Labs Research

**Affected Platforms:** Windows

**Impacted Users:** Windows users

**Impact:** Most files on the compromised machines are encrypted

**Severity Level:** Medium

In Greek mythology, opening of the infamous Pandora's box (jar) introduced terrible things to the world. That can also be said about today's ransomware. The newly emerged Pandora ransomware that crowned the name is no exception. It steals data from the victim's network, encrypts the victim's files, and unleashes the stolen data if the victim opts not to pay. The Greek myth says hope was left in the box. Does that hold true for Pandora ransomware, an emerging malware that shows all techniques used by modern ransomware? In this blog we are taking a hammer and crowbar to look inside today's Pandora's box to find out what mysteries it holds. We will discuss:

- How this ransomware tries to evade detection
- The numerous obfuscation and anti-analysis techniques that are used to hinder analysts
- How multi-threading is used to speed up processing
- How the filesystem is processed
- How and which files are encrypted.

## Pandora Group

The Pandora ransomware group emerged into the already crowded ransomware field as early as in mid-February 2022 and targets corporate networks for financial gain. The group got recent publicity after they announced that they acquired data from an international supplier in the automotive industry. The incident came as surprise as the attack came two weeks after another automotive supplier was reportedly hit with unknown ransomware, which resulted in one of the world's biggest car manufacturers suspending factory operations. The threat group uses the double extortion method to increase pressure on the victim. This means that they not only encrypt the victim's files, but also exfiltrate them and threaten to release the data if the victim does not pay.

The Pandora Group has a leak site in the Dark Web (TOR network), where they publicly announce their victims and threaten them with the data leak. There are currently three victims listed on the leak site (see Figure 1), a U.S.-based real estate agency, a Japanese technology company, and a U.S. law firm.

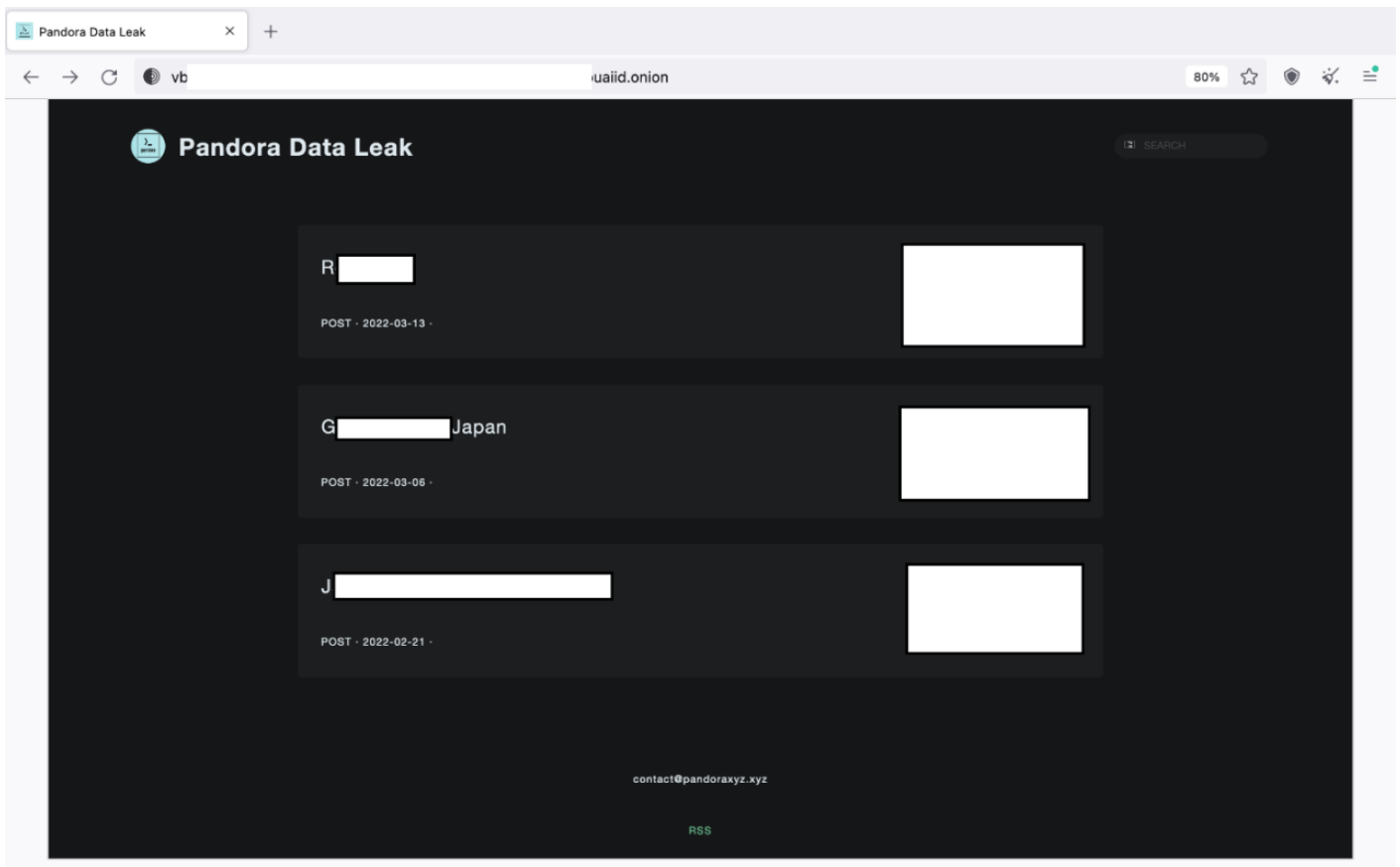


Figure 1 - Pandora's leak site

## Malware Analysis

We analyzed the sample with the SHA-256 hash

5b56c5d86347e164c6e571c86dbf5b1535eae6b979fed6ed66b01e79ea33b7b, which is a 64-bit Windows PE file. It is the ransomware itself, so by the time this file is executed during an attack, the attackers probably already had extensive access to the victim's network, and they had already exfiltrated the data they will use for the extortion. This sample does not have the capability to communicate with the threat actors. Its sole purpose is to find and encrypt files. However, it does this in an interesting and complex manner.

In the following sections these interesting aspects of the malware will be discussed.

### Execution Flow

The sample goes through the following steps:

Note that "T" followed by numbers within brackets refers to MITRE ATT&CK technique ID, which are summarized at the end of the post.

1) **Unpacking**: The sample is packed with a modified UPX packer (T1027.002), so the first step is to unpack the real content to memory and jump to it. This will be discussed later.

2) **Mutex**: It creates a mutex called ThisIsMutexa.

3) **Disable Security Features**: It can delete Windows shadow copies (T1490), bypass AMSI (T1562.001), and disable Event Logging (T1562.002). More on these features later.

- 4) **Collects system information:** GetSystemInfo() is used to collect information about the local system.
- 5) **Loads Hardcoded Public Key:** A public key is hardcoded in the malware sample. This is used to set up the cryptography for encryption.
- 6) **Store Private and Public Keys in Registry:** A private key is generated, and both the hardcoded public key and the newly generated private key are stored in the registry (T1112).
- 7) **Search Drives:** It searches for unmounted drives on the system and mounts them to encrypt them as well (T1005).
- 8) **Setup Multi-Threading:** The sample uses worker threads to distribute the encryption process. More on this later.
- 9) **Enumerate Filesystem:** The worker threads start to enumerate the filesystems of the identified drives (T1083).
- 10) **Drop Ransom Note:** The ransom note is dropped in every folder in Restore\_My\_Files.txt.
- 11) **Check File Name Blacklist:** For every file and folder a blacklist of file/folder names is checked. If the file/folder is on the blacklist it will not be encrypted. More on this later.
- 12) **Check File Extension Blacklist:** Each file is checked against a file extension blacklist. If the extension is on the list, it will not be encrypted.
- 13) **Unlock File:** If the file is locked by a running process, the sample will try to unlock it using the Windows Restart Manager(T1489).
- 14) **Encrypt File:** The worker threads will encrypt(T1486) the file and write it back to the original file.
- 15) **Rename File:** Once the encryption is finished the file is renamed to[original\_filename].pandora.

## Anti-Reverse Engineering Techniques

One of the most significant aspects of the Pandora ransomware is the extensive use of anti-reverse-engineering techniques. This is not new for malware, but Pandora lies on the extreme side of how much is invested in slowing analysis down. In this section we will go through the different techniques that were identified.

### Packed

The sample is packed with a modified UPX packer, which can be easily detected with Detect It Easy (see Figure 2).

packer	UPX(3.00)[-]	S ?
linker	Microsoft Linker(14.0, Visual Studio 2015 14.0*)[Console64,console]	S ?

Figure 2: Detect It Easy can identify UPX

However, the standard UPX unpacker does not work, which indicates that the packer was modified to make sure that off-the-shelf tools cannot be used to unpack it.

Unpacking is still relatively easy, by scrolling down from the entry point to the end of the code in a debugger. The code will end with a jump (Figure 3). This is typical with packers, that after unpacking the original code somewhere in memory they will jump there, instead of returning from the main function.

00007FF68700BE87	48: 89DA	mov rdx,rbx	rdx:EntryPoint
00007FF68700BE8A	48: 89F9	mov rcx,rdi	
00007FF68700BE8D	FFD5	call rbp	
00007FF68700BE8F	48: 83C4 28	add rsp,28	
00007FF68700BE93	5D	pop rbp	
00007FF68700BE94	5F	pop rdi	
00007FF68700BE95	5E	pop rsi	
00007FF68700BE96	5B	pop rbx	
00007FF68700BE97	48: 8D4424 80	lea rax,qword ptr ss:[rsp-80]	rax:EntryPoint
00007FF68700BE9C	6A 00	push 0	
00007FF68700BE9E	48: 39C4	cmp rsp,rax	rax:EntryPoint
00007FF68700BEA1	75 F9	jne pandora.7FF68700BE9C	
00007FF68700BEA3	48: 83EC 80	sub rsp,FFFFFFFFFFFFFFF80	
00007FF68700BEA7	E9 58A7BFF	jmp pandora.7FF686FC6604	Tail jump to original entry point
00007FF68700BEAC	0000	add byte ptr ds:[rax],al	rax:EntryPoint
00007FF68700BEAE	0000	add byte ptr ds:[rax],al	rax:EntryPoint
00007FF68700BEB0	94	xchg esp,eax	
00007FF68700BEB1	0000	add byte ptr ds:[rax],al	rax:EntryPoint
00007FF68700BEB3	0000	add byte ptr ds:[rax],al	rax:EntryPoint
00007FF68700BEB5	0000	add byte ptr ds:[rax],al	rax:EntryPoint
00007FF68700BEB7	0000	add byte ptr ds:[rax],al	rax:EntryPoint
00007FF68700BEB9	0000	add byte ptr ds:[rax],al	rax:EntryPoint
00007FF68700BEBB	0000	add byte ptr ds:[rax],al	rax:EntryPoint
00007FF68700BEFD	0000	add byte ptr ds:[rax],al	rax:EntryPoint

Figure 3: Tail jump at the end of the unpacking

By putting a breakpoint to the tail jump we can dump the PE file from memory including the unpacked code. With the dumped file we can analyze the ransomware statically as well.

### Control-Flow Flattening

Control-Flow Flattening is an obfuscation technique that can hide the structure of the program by modifying the control-flow. In the simplest case, it replaces the normal control flow of each function with a state machine, thus it makes harder for an analyst to quickly understand how each function works. Pandora uses a more complex control-flow flattening combined with opaque predicates, to complicate the control flow even further.

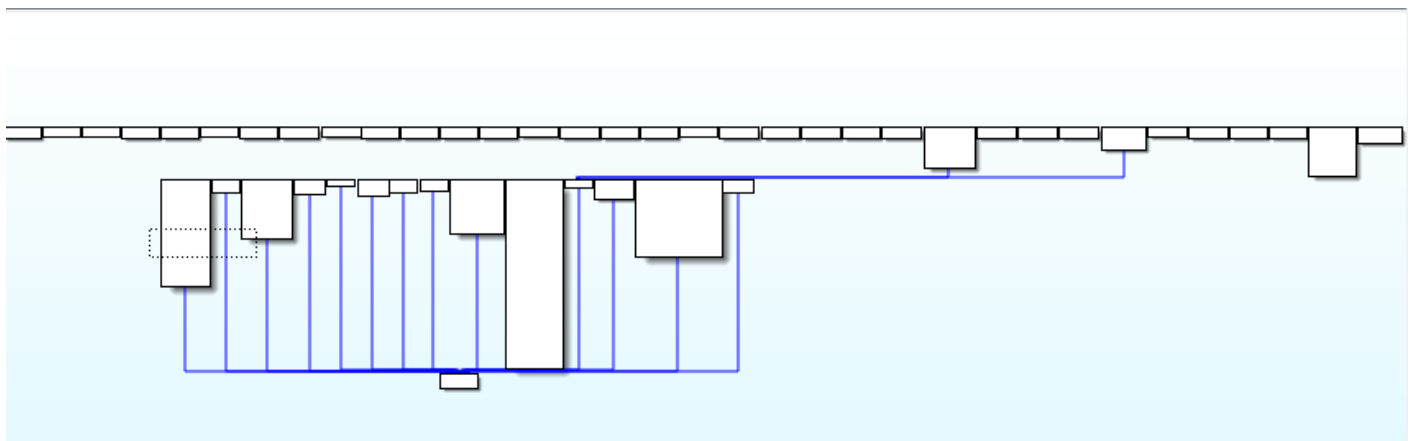
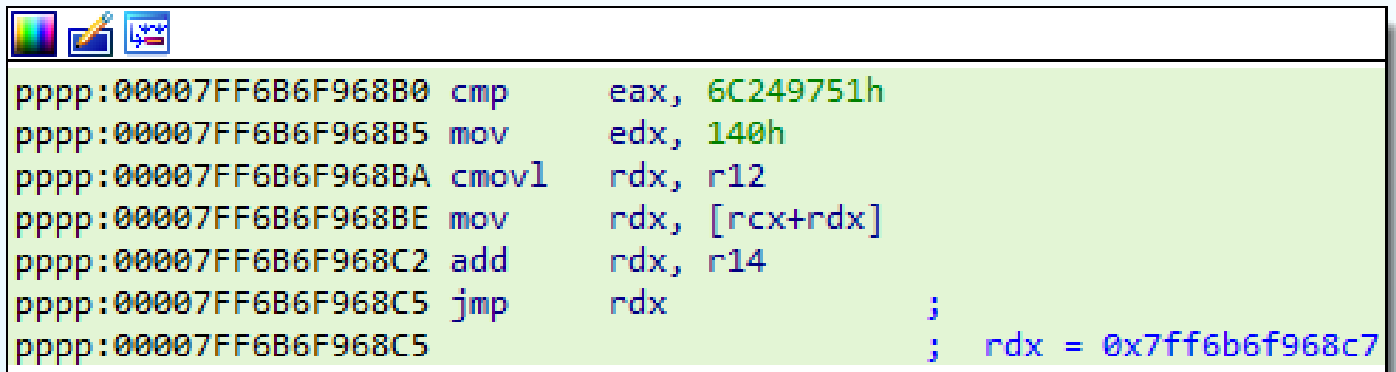


Figure 4: Graph view of main()

Figure 4 shows the graph view of the main function in the unpacked code. We can see that it does not resemble a normal function's control flow. It looks like a huge switch-case statement, which is the result of the control-flow flattening that implements a state machine. However, in Pandora's case most of the

basic blocks are not connected at all. This is the result of the opaque predicates. Most of the jumps between basic blocks are calculated at runtime, as shown in Figure 5.



```
pppp:00007FF6B6F968B0 cmp     eax, 6C249751h
pppp:00007FF6B6F968B5 mov     edx, 140h
pppp:00007FF6B6F968BA cmovl   rdx, r12
pppp:00007FF6B6F968BE mov     rdx, [rcx+rdx]
pppp:00007FF6B6F968C2 add     rdx, r14
pppp:00007FF6B6F968C5 jmp     rdx ;
pppp:00007FF6B6F968C5 ; rdx = 0x7ff6b6f968c7
```

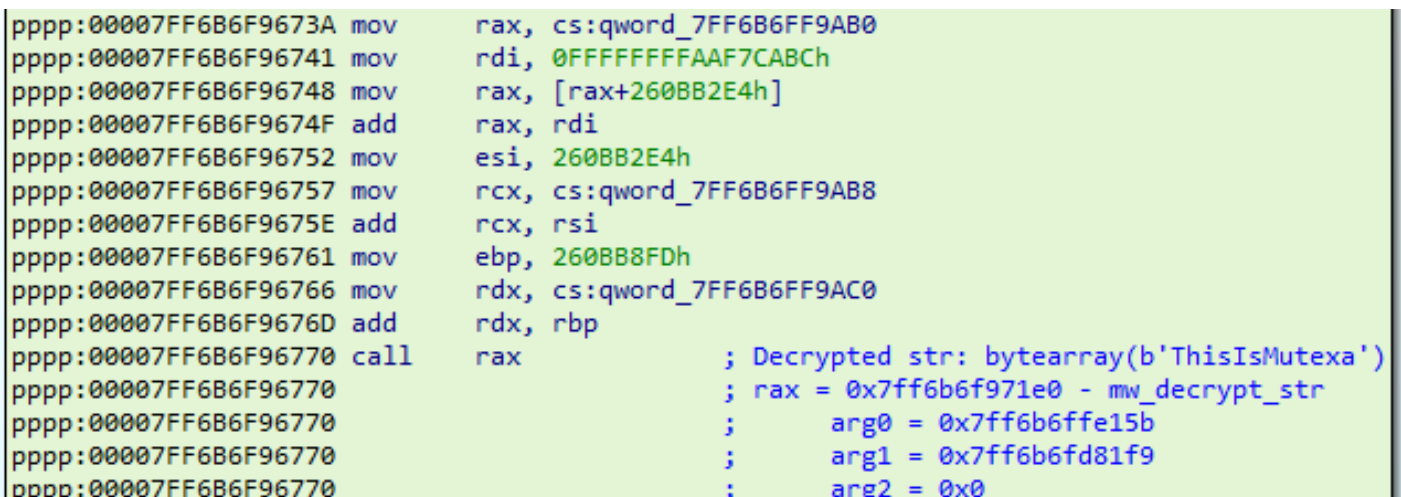
Figure 5: Calculating the address for the jmp in runtime

The first cmp instruction checks the current state of the state machine and depending on that calculates the value of the rdx register for the jmp at the end of the basic block. Because of this static analysis tools, such as IDA Pro cannot understand where the control flow will continue, and thus cannot connect the basic blocks in Figure 4.

Emulation can be used to understand the control flow to a limited degree but debugging had to be applied extensively to be sure how the execution flows.

## String Encoding

Some strings can be found in the unpacked binary, but most of them are from the statically linked libraries. However, the strings that would help us understand what is happening in the code are encoded. Figure 6 shows how one of the string decryption functions is called.



```
pppp:00007FF6B6F9673A mov     rax, cs:qword_7FF6B6FF9AB0
pppp:00007FF6B6F96741 mov     rdi, 0FFFFFFFAAF7CABCh
pppp:00007FF6B6F96748 mov     rax, [rax+260BB2E4h]
pppp:00007FF6B6F9674F add     rax, rdi
pppp:00007FF6B6F96752 mov     esi, 260BB2E4h
pppp:00007FF6B6F96757 mov     rcx, cs:qword_7FF6B6FF9AB8
pppp:00007FF6B6F9675E add     rcx, rsi
pppp:00007FF6B6F96761 mov     ebp, 260BB8FDh
pppp:00007FF6B6F96766 mov     rdx, cs:qword_7FF6B6FF9AC0
pppp:00007FF6B6F9676D add     rdx, rbp
pppp:00007FF6B6F96770 call    rax ; Decrypted str: bytearray(b'ThisIsMutexa')
pppp:00007FF6B6F96770 ; rax = 0x7ff6b6f971e0 - mw_decrypt_str
pppp:00007FF6B6F96770 ; arg0 = 0x7ff6b6ffe15b
pppp:00007FF6B6F96770 ; arg1 = 0x7ff6b6fd81f9
pppp:00007FF6B6F96770 ; arg2 = 0x0
```

Figure 6: Calling one of the string decryption functions

Both the address of the decryption function, which is called through rax, and the address of the encoded string, are calculated at runtime. This way, when looking at this code statically, there is no way to know what is happening here. The comment on the right side is the result of an IDAPython script that uses the [flare-emu](#) project to emulate the code and calculate the addresses of the function call, as well as emulate the decryption function. This solution was very effective in recovering the encoded strings in the binary. The decryption function implements an XOR decoding. The decryption keys are stored together

with each encoded string. As a bonus, the malware uses multiple decryption functions. We identified 14 separate functions that are used for string decoding.

## Function Call Obfuscation

It was already mentioned in the previous section that most function calls are not calling a direct address, but a register. Its value is calculated at runtime.

If we use Figure 6 as an example the address in `rax` is calculated like this:

```
rax =>(*address_table_base + 638300900) - 1426601284)
```

As mentioned, this was solved using emulation. By emulating the execution of a function, we could calculate the register values at `CALL` instructions. This allowed us to resolve function calls at scale.

## Windows API Call Obfuscation

Contrary to other malware, the Windows API function names are not encoded, but another obfuscation technique is used to hide their usage. As shown in Figure 7, the Windows API functions are organized in a jump table. At each address there is a `jmp` instruction that redirects to the library function.

```
pppp:00007FF6B6FC62C4 ; -----
pppp:00007FF6B6FC62C4          jmp     cs:HeapFree
pppp:00007FF6B6FC62CA ; -----
pppp:00007FF6B6FC62CA          jmp     cs:GetProcessHeap
pppp:00007FF6B6FC62D0 ; -----
pppp:00007FF6B6FC62D0          jmp     cs:HeapAlloc
pppp:00007FF6B6FC62D6 ; [00000006 BYTES: COLLAPSED FUNCTION VirtualFree. PRESS CTRL-NUMPAD+ TO EXPAND]
pppp:00007FF6B6FC62DC ; -----
pppp:00007FF6B6FC62DC          jmp     cs:CreateFileMappingW
pppp:00007FF6B6FC62E2 ; -----
pppp:00007FF6B6FC62E2          jmp     cs:MapViewOfFile
pppp:00007FF6B6FC62E8 ; -----
pppp:00007FF6B6FC62E8          jmp     cs:VirtualAlloc
pppp:00007FF6B6FC62EE ; -----
pppp:00007FF6B6FC62EE          jmp     cs:UnmapViewOfFile
```

Figure 7: Windows API function jump table

Resolving the API functions was implemented in the same `flare-emu` IDAPython script that resolves the function calls. Whenever a `CALL [register]` points to a `jmp` instruction (see Figure 8), instead of the beginning of a function, then we assumed that it points to the API function jump table. So we took the name of the operand of the jump and used that to generate the comments for the function call (see Figure 9).

```
42     # check if points to the jump table
43     if eh.analysisHelper.getMnem(operand_value).lower() == "jmp":
44         fname = eh.analysisHelper.getName(eh.analysisHelper.getOpndValue(operand_value, 0))
45         print("[+] API call found: {}".format(fname))
```

Figure 8: Recovering API function name in the emulation script

```

pppp:00007FF6B6F9677C mov rax, cs:qword_7FF6B6FF9AB0
pppp:00007FF6B6F96783 mov rax, [rax+260BB2ECh]
pppp:00007FF6B6F9678A add rax, rdi
pppp:00007FF6B6F9678D mov ecx, 1F0001h
pppp:00007FF6B6F96792 xor edx, edx
pppp:00007FF6B6F96794 call rax ; rax = 0x7ff6b6fc627c - OpenMutexA

```

Figure 9: The script recovered that this is a function call to OpenMutexA

## Multi-Threading

Pandora uses multiple threads to speed up the encryption process. For that it uses Windows's [IO Completion Ports](#) concept. This allows threads to wait for a file/network handle to appear in the IO Completion Port queue and process them. Pandora uses unassociated IO Completion Ports and sends any data through it using the OVERLAPPED structure. In this case drives and file paths will be passed to threads to process (enumerate or encrypt) them. The IO Completion Port is set up using the `CreateIoCompletionPort()` API function as shown in Figure 10. By passing `INVALID_HANDLE_VALUE` as the first parameter (`rcx = 0xFFFFFFFFFFFFFFFF`) and `NULL` as the second (`rdx = 0x0`) an unassociated IO Completion Port is created. The fourth parameter is the `NumberOfConcurrentThreads`, which is set to 4 (`r9 = 0x4`), defines that maximum 4 threads are allowed to work with this IO Completion Port.

<pre> mov r9d,dword ptr ss:  mov rax,qword ptr ds: mov rbx,qword ptr ds: add rbx,rdi mov rcx,FFFFFFFFFFFFFFF xor edx,edx xor r8d,r8d call rbx mov rcx,qword ptr ds: mov qword ptr ds:[rcx: mov rax,qword ptr ds: mov rax,qword ptr ds: add rax,rdi call rax </pre>	<table border="0"> <tr><td>RAX</td><td>00007FF690F3E64C</td><td></td></tr> <tr><td>RBX</td><td>00007FF6B6FC62A0</td><td>&lt;pandora.JMP.&amp;CreateIoCompletionPort&gt;</td></tr> <tr><td>RCX</td><td>FFFFFFFFFFFFFFF</td><td></td></tr> <tr><td>RDY</td><td>0000000000000000</td><td></td></tr> <tr><td>RBP</td><td>0000000000000058</td><td>'x'</td></tr> </table> <table border="0"> <tr><td colspan="3">Default (x64 fastcall)</td></tr> <tr><td>1:</td><td>rcx</td><td>FFFFFFFFFFFFFFF</td></tr> <tr><td>2:</td><td>rdx</td><td>0000000000000000</td></tr> <tr><td>3:</td><td>r8</td><td>0000000000000000</td></tr> <tr><td>4:</td><td>r9</td><td>0000000000000004</td></tr> <tr><td>5:</td><td>[rsp+20]</td><td>0000000000000004</td></tr> </table>	RAX	00007FF690F3E64C		RBX	00007FF6B6FC62A0	<pandora.JMP.&CreateIoCompletionPort>	RCX	FFFFFFFFFFFFFFF		RDY	0000000000000000		RBP	0000000000000058	'x'	Default (x64 fastcall)			1:	rcx	FFFFFFFFFFFFFFF	2:	rdx	0000000000000000	3:	r8	0000000000000000	4:	r9	0000000000000004	5:	[rsp+20]	0000000000000004
RAX	00007FF690F3E64C																																	
RBX	00007FF6B6FC62A0	<pandora.JMP.&CreateIoCompletionPort>																																
RCX	FFFFFFFFFFFFFFF																																	
RDY	0000000000000000																																	
RBP	0000000000000058	'x'																																
Default (x64 fastcall)																																		
1:	rcx	FFFFFFFFFFFFFFF																																
2:	rdx	0000000000000000																																
3:	r8	0000000000000000																																
4:	r9	0000000000000004																																
5:	[rsp+20]	0000000000000004																																

Figure 10: Initializing an IO Completion Port

After this, the main function will start the new threads. The communication between the threads is done using the `GetQueuedCompletionStatus()` and `PostQueuedCompletionStatus` API functions. Figure 11 shows how a discovered file (`pydisas.py`) is put in the queue with `PostQueuedCompletionStatus()`. Another thread will pick up this task with `GetQueuedCompletionStatus()`, and since it receives a full path to a file it will encrypt and rename it.

<pre> rip → 00007FF6B6F95BE8 FFD5 call rbp 00007FF6B6F95BEA BF A8040000 mov ecx,dword ptr ss:[rsp+44] 00007FF6B6F95BEF 8B4C24 44 mov eax,C3AC2EBB 00007FF6B6F95BF3 B8 B82EAC33 add ecx,eax 00007FF6B6F95BF8 01C1 xor edx,edx 00007FF6B6F95BFA E9 71F2FFFF jmp pandora.7FF6B6F94E70 00007FF6B6F95BFF 31D2 xor ecx,edx 00007FF6B6F95C01 81F9 488823E1 cmp ecx,E1238848 00007FF6B6F95C07 0F9FC2 setg dl 00007FF6B6F95C0A 48:C1E2 04 shl rdx,4 00007FF6B6F95C0E 48:8B9402 78020000 mov rdx,qword ptr ds:[rdx+rax+2] 00007FF6B6F95C16 4C:01E2 add rdx,r12 00007FF6B6F95C19 FFE2 jmp rdx 00007FF6B6F95C1B 81F9 C90857D8 cmp ecx,D85708C9 00007FF6B6F95C21 BA D8020000 mov edx,308 </pre>	<table border="0"> <tr><td>RAX</td><td>00007FF664CE6890</td><td>L'c'</td></tr> <tr><td>RBX</td><td>0000000000000188</td><td>L'y'</td></tr> <tr><td>RCX</td><td>0000000000000474</td><td></td></tr> <tr><td>RDY</td><td>0000000000000001</td><td></td></tr> <tr><td>RBP</td><td>00007FF6B6FC625E</td><td>&lt;pandora.JMP.&amp;PostQueuedCompletionStatus&gt;</td></tr> <tr><td>RSP</td><td>00000000C03FF670</td><td></td></tr> <tr><td>RSI</td><td>0000000000000380</td><td></td></tr> <tr><td>RDI</td><td>000000007A2C7C88</td><td></td></tr> </table> <table border="0"> <tr><td colspan="3">Default (x64 fastcall)</td></tr> <tr><td>1:</td><td>rcx</td><td>0000000000000474</td></tr> <tr><td>2:</td><td>rdx</td><td>0000000000000001</td></tr> <tr><td>3:</td><td>r8</td><td>0000000000000001</td></tr> <tr><td>4:</td><td>r9</td><td>000001B3041D6380</td></tr> <tr><td>5:</td><td>[rsp+20]</td><td>0000000000000000</td></tr> </table>	RAX	00007FF664CE6890	L'c'	RBX	0000000000000188	L'y'	RCX	0000000000000474		RDY	0000000000000001		RBP	00007FF6B6FC625E	<pandora.JMP.&PostQueuedCompletionStatus>	RSP	00000000C03FF670		RSI	0000000000000380		RDI	000000007A2C7C88		Default (x64 fastcall)			1:	rcx	0000000000000474	2:	rdx	0000000000000001	3:	r8	0000000000000001	4:	r9	000001B3041D6380	5:	[rsp+20]	0000000000000000
RAX	00007FF664CE6890	L'c'																																									
RBX	0000000000000188	L'y'																																									
RCX	0000000000000474																																										
RDY	0000000000000001																																										
RBP	00007FF6B6FC625E	<pandora.JMP.&PostQueuedCompletionStatus>																																									
RSP	00000000C03FF670																																										
RSI	0000000000000380																																										
RDI	000000007A2C7C88																																										
Default (x64 fastcall)																																											
1:	rcx	0000000000000474																																									
2:	rdx	0000000000000001																																									
3:	r8	0000000000000001																																									
4:	r9	000001B3041D6380																																									
5:	[rsp+20]	0000000000000000																																									

<pre> Address UNICODE 000001B3041D6380   .....c:\Python27\Lib\site-packages\xdis\bin\pydisas 000001B3041D6430   m.py 000001B3041D6480   ..... 000001B3041D6530   ..... </pre>	<pre> 00000000C03FF 00000000C03FF 00000000C03FF 00000000C03FF 00000000C03FF </pre>
---	--

Figure 11: Posting the file path to the IO Completion Port's queue

## Restart Manager

The Restart Manager is a Windows feature to reduce the number of restarts needed during installation and updates. The reason for a restart is usually because a file that needs to be updated is locked by a running process. The Restart Manager can save the state and stop the locking process to unlock the target file. Once the update is finished, it can restore the locking process again. Pandora uses the Restart Manager to make sure that even files that are currently locked will be encrypted. For each file the following process is executed:

- 1) Create Restart Manager session with RmStartSession()
- 2) Register the target file as a resource with RmRegisterResource()
- 3) Check if the target file is locked by any process with RmGetList()
- 4) If so, terminate locking processes
- 5) End Restart Manager session with RmEndSession()

## Encryption

- Before a file is encrypted, Pandora does the following checks to make sure that it does not render the machine inoperable. Each target file is checked against the following blacklist of file and folder names. If the target file is on the list, Pandora will not encrypt it.

AppData	\$Recycle.Bin	iconcache.db
Boot	ProgramData	ntldr
Windows	All Users	ntuser.dat
Windows.old	autorun.inf	ntuser.dat.log
Tor Browser	boot.ini	ntuser.ini
Internet Explorer	bootfont.bin	thumbs.db
Google	bootsect.bak	Program Files
Opera	bootmgr	Program Files (x86)
Opera Software	bootmgr.efi	#recycle
Mozilla	bootmgfw.efi	..
Mozilla Firefox	desktop.ini	.

- Each target file is compared to the following list of file extensions. If the file's extension is on the list, the file will not be encrypted:

.hta .exe .dll .cpl .ini .cab .cur .drv .hlp .icl .icns .ico .idx .sys .spl .ocx

The ransom note, shown in Figure 12, promises an RSA-2048 encryption. The fact that malware is shipped with a hardcoded RSA-2048 public key (Figure 13) confirms this claim.



```

### What happened?
#### !!!Your files are encrypted!!!

*All your files are protected by strong encryption with RSA-2048.*
*There is no public decryption software.*
*We have successfully stolen your confidential document data, finances, emails, employee information, customers, research and development products.*

#### What is the price?

*The price depends on how fast you can write to us.*
*After payment, we will send you the decryption tool which will decrypt all your files.*

#### What should I do?

*There is only one way to get your files back -->>Contact us, pay and get decryption software.*
*If you decline payment, we will share your data files with the world.*
*You can browse your data breach here: http://vb[REDACTED]aiid.onion*
(you should download and install TOR browser first hxxps://torproject.org)

#### !!!Decryption Guaranteed!!!

*Free decryption As a guarantee, you can send us up to 3 free decrypted files before payment.*

#### !!!Contact us!!!

email:
contact@pandoraxyz.xyz
#### !!!Warning!!!

*Do not attempt to decrypt your data using third-party software, this may result in permanent data loss.*
*Decrypting your files with the help of a third party may result in a price increase (they charge us a fee), or you may fall victim to a scam.*
*Don't try to delete programs or run antivirus tools. It won't work.*
*Attempting to self-decrypt the file will result in the loss of your data.*

```

Figure 12: Ransom note

```

1  -----BEGIN PUBLIC KEY-----
2  MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAtf03xC/xw91eLp1TsIEc
3  05DrVNYGseE4GUM+bJyaeftX2IdAKwJcdjwkgQD3RwMEQ1/n+UHo0poTIjuSu5qN
4  k+Eg7P62mbKxsJxMMsKoPwn9AcRoL47Z8yk0IUMqtmiHZVbhffF5gzh78mV/f6lF7
5  PzIeTNgU8MDvQD1H+4av1NjD+zW0mIFcbhWhoUCF6aaK6ySJ7VZwhM2wkIAA1j5p
6  v11LrPASyrN1nJlRPCK7CqJ/SQjUBptaWl5VtPGtmGFOPQVRzORISdEFS0fe30WE
7  vq+HuxLZKMZCXaHHbVaZzVp3DMA+gCnGkTx2Pfry1MVJ6yTcBMNzz9zws7VhADaG
8  1QIDAQAB
9  -----END PUBLIC KEY-----

```

Figure 13: Hardcoded RSA public key

A private key is also generated and both of these keys are stored in the registry under HKCU\SOFTWARE\[Private,Public] (see Figure 14).

Computer\HKEY_CURRENT_USER\SOFTWARE			
Name	Type	Data	
(Default)	REG_SZ	(value not set)	
Private	REG_BINARY	5d 8d f7 e1 2a a3 72 61 eb 67 f4 55 b0 56 c8 e0 7f e1...	
Public	REG_BINARY	2d 2d 2d 2d 2d 42 45 47 49 4e 20 50 55 42 4c 49 43 2...	

Figure 14: Cryptographic keys are stored in the Registry

The unpacked binary contains the [Mbed TLS](#) cryptographic library statically linked.

Once a file is encrypted in memory, it is written to disk. After that the file is renamed to have the .pandora extension.

## Disabling Security Features

The Pandora ransomware has the capabilities to disable some of the security measures on the target machine.

### Deleting Shadow Copies

Like a lot of other ransomware, Pandora deletes the Windows Shadow Copies, which could help the operator restore the machine to a state before the infection. Figure 15 shows the call to ShellExecuteW() with the parameters from runtime(T1059). We can see that it uses the vssadmin.exe.

```
pppp:00007FF6B6F945FA call    rbp          ; rbp = 0x7ff6b6fc632a - __imp_ShellExecuteW
pppp:00007FF6B6F945FA          ; 1: rcx 0000000000000000
pppp:00007FF6B6F945FA          ; 2: rdx 00007FF6B6FD7A26 L"open"
pppp:00007FF6B6F945FA          ; 3: r8 00007FF6B6FD7A16 L"cmd.exe"
pppp:00007FF6B6F945FA          ; 4: r9 00007FF6B6FD79C0 L"/c vssadmin.exe delete shadows /all /quiet"
pppp:00007FF6B6F945FA          ; 5: [rsp+20] 0000000000000000
```

Figure 15: Deleting shadow copies with ShellExecuteW

### AMSI Bypass

The Antimalware Scan Interface (AMSI) allows security products to integrate better with Windows to be able to scan all kinds of different objects, such as PowerShell scripts, JavaScript, VBScript, etc. By bypassing AMSI, the malware can take away significant capabilities from the security products running on the machine. Pandora bypasses AMSI by patching the AmsiScanBuffer() function in the amsi.dll in memory.

### Disable Event Log

Similar to the AMSI bypass, Pandora disables the Event Tracing for Windows (ETW) feature, by patching the EtwEventWrite() function in the Windows kernel (ntdll.dll). Figure 16 shows that the first byte of the function is replaced with 0xC3, which is the ret instruction. This renders the EtwEventWrite() function useless, because after every call it return immediately without logging the event.

Address	Disassembly	Comment
00007FF99E17F1A0	C3	ret
00007FF99E17F1A1	8BDC	mov ebx,esp
00007FF99E17F1A3	48:83EC 58	sub rsp,58
00007FF99E17F1A7	4D:894B E8	mov qword ptr ds:[r11-18],r9
00007FF99E17F1A8	33C0	xor eax,eax
00007FF99E17F1AD	45:8943 E0	mov dword ptr ds:[r11-20],r8d
00007FF99E17F1B1	45:33C9	xor r9d,r9d
00007FF99E17F1B4	49:8943 D8	mov qword ptr ds:[r11-28],rax
00007FF99E17F1B8	45:33C0	xor r8d,r8d
00007FF99E17F1B8	49:8943 D0	mov qword ptr ds:[r11-30],rax
00007FF99E17F1BF	66:894424 20	mov word ptr ss:[rsp+20],ax
00007FF99E17F1C4	E8 5F000000	call ntdll.7FF99E17F228
00007FF99E17F1C9	48:83C4 58	add rsp,58
00007FF99E17F1CD	C3	ret

Figure 16: Patching the EtwEventWrite function to return immediately

## Conclusion

The Pandora ransomware contains all of the most important features that state-of-the-art ransomware samples usually contain. The level of obfuscation to slow down analysis is more advanced than average malware. The threat actor also paid attention to unlock files to guarantee the maximum encryption coverage, while still allowing the machine to run. We can already see anti-security product features. We can expect the threat actor to develop these capabilities further. There is currently no proof that Pandora operates as Ransomware-as-a-Service (RaaS), but the time investment in the complexity of the malware might indicate that they are moving in that direction in the long term. The current attacks and leaks might be a way to make their name in the ransomware field, which they could capitalize on if they adopt the RaaS model later. It is worth tracking the threat actor to monitor how their malware changes.

## Fortinet Protection

The analyzed Pandora ransomware sample is detected by the following (AV) signature:

W64/Filecoder.EGYTYFD!tr.ransom

FortiEDR also detects and mitigates execution of Pandora ransomware through the combination of behavioral analysis, and integration with machine learning and threat intelligence feeds. Execution of the Pandora sample analyzed as part of this blog triggers seven rules resulting in nine security events. Triggered rules were a result of pre-execution analysis and post-execution behaviors. These security events can be observed below in Figure 16.

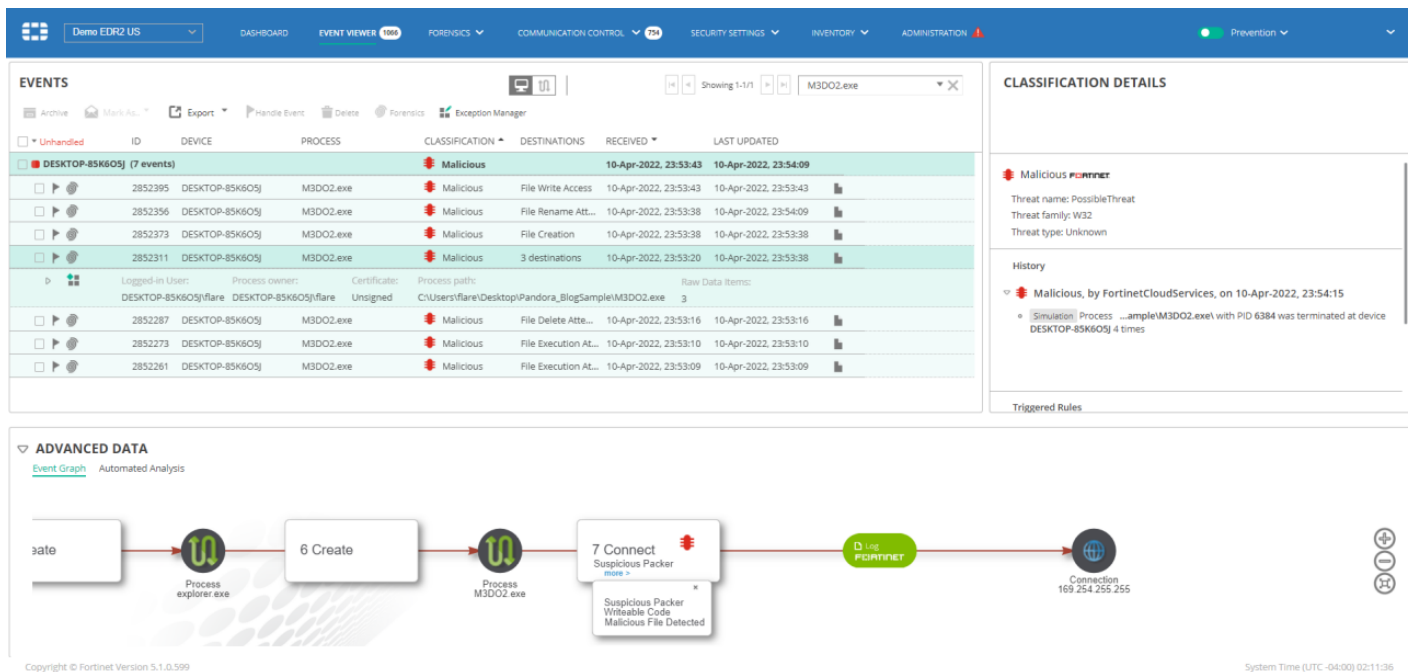


Figure 16. FortiEDR security events generated following execution of Pandora ransomware sample. Note that during this execution FortiEDR was set to only log events rather than mitigate to properly demonstrate detections post-execution.

Pre-execution detections included; identifying the malicious file (hash based), detection of a suspicious packer and presence of writeable code. Post-execution detections included; detection of each file encryption attempt, detection of encrypted file rename attempt, dropping of the ransom-note and attempts to access SMB shares.

In Protect mode FortiEDR will detect and mitigate detected behavior. In the case of Pandora this will prevent execution of the ransomware, mitigating malicious activity before it occurs, and will prevent subsequent file encryption attempts if the adversary is able to execute the sample. The post exploitation detections are not dependent on signature meaning they will effectively mitigate this activity for newer Pandora variants even with no prior knowledge of the samples.

## IOCs:

Mutex: ThisIsMutexa

Ransom note: Restore\_My\_Files.txt

SHA256 hash of hardcoded public key:

7b2c21eea03a370737d2fe7c108a3ed822be848cce07da2ddc66a30bc558af6b

SHA256 hash of sample: 5b56c5d86347e164c6e571c86dbf5b1535eae6b979fede6ed66b01e79ea33b7b

## ATT&CK TTPs

TTP Name	TTP ID	Description
Obfuscated Files or Information: Software Packing	T1027.002	Modified UPX packer
Impair Defenses: Disable Windows Event Logging	T1562.002	Disable Event Logging
Impair Defenses: Disable or Modify Tools	T1562.001	Bypass AMSI
Data from Local System	T1005	Searches unmounted drives and partitions
Modify Registry	T1112	Cryptographic keys are stored in the registry
Data Encrypted for Impact	T1486	As a ransomware it encrypts files
Command and Scripting Interpreter	T1059	Uses cmd.exe to remove the shadow copies
System Information Discovery	T1082	Collects system information with GetSystemInfo()
File and Directory Discovery	T1083	Discovers drives and enumerates filesystems
Inhibit System Recovery	T1490	Deletes shadow copies
Service Stop	T1489	Terminates processes if they lock a file