

Technical Analysis of Emerging, Sophisticated Pandora Ransomware Group

By No items found.

Published: 2025-08-21 · Archived: 2026-04-05 13:57:00 UTC

2021 saw an outbreak of ransomware groups and attacks that affected every major industry across the globe. This trend is expected to continue and even surpass the previous year’s numbers by a significant margin in 2022.

In March 2022, researchers detected a new ransomware strain known as Pandora which leverages double extortion tactics to exfiltrate and encrypt large quantities of personal data. The operators offer the decryption key once the victim pays the ransom demanded. Pandora ransomware is a relatively new operation and hence its infection techniques are unknown.

However, after infiltrating the target system, the ransomware appends the “.pandora” file extension to the encrypted files and leaves a ransom note “Restore_My_Files.txt” with instructions on how to recover the data. Researchers believe that the Pandora ransomware is a rebranded version of Rook ransomware, which in turn is a spawn of the leaked Babuk code. This article explores the technical analysis of the Pandora ransomware, its evasion tactics, the process of encryption, and more in detail.

Technical Analysis of Pandora

The analysis of Pandora’s binary file sample,

5b56c5d86347e164c6e571c86dbf5b1535eae6b979fede6ed66b01e79ea33b7b , indicates that it is a UPX (Ultimate Packer for eXecutables) packed binary file. UPX is an executable file compressor used by threat actors to add a layer of obfuscation (creation of code that is difficult for humans to understand) to their malware. The ransomware code runs from the original entry point after getting unpacked in the memory.

```
00007FF76B8D6604 48:83EC 28      sub rsp,28
00007FF76B8D6608 E8:83020000    call sample.7FF76B8D68C0
00007FF76B8D660D 48:83C4 28      add rsp,28
00007FF76B8D6611 E9:76FEFFFF    jmp sample.7FF76B8D648C
00007FF76B8D6616 CC              int3
```

Ransomware code running from the entry point

The ransomware uses obfuscated strings and deobfuscates library names and internal functions at runtime. The library modules used by Pandora are dynamically loaded on a per-use basis via the following APIs:

- **LoadlibraryA**
- **GetProcAddress**
- **GetModuleHandleA**

Initially, the ransomware creates a mutex (mutual exclusion object, which enables multiple program threads to take turns sharing the same resource) to make sure only one instance of the malware is running on the system. The mutex string, “ThisIsMutexa”, gets deobfuscated in the memory. It checks for any existing mutex on the system

via **OpenMutexA**, if not present the malware creates a new one with the value “ThisIsMutexa” via **CreateMutexA**.

Anti-debug Mechanism

The malware implements anti-debug checks to hinder analysis.

```

00007FF76B8AC412 41:56 push r14
00007FF76B8AC414 56 push rsi
Breakpoint Not Set 415 57 push rdi
00007FF76B8AC416 55 push rbp
00007FF76B8AC417 53 push rbx
00007FF76B8AC418 48:83EC 28 sub rsp,28
00007FF76B8AC41C C74424 24 392E1469 mov_dword_ptr_ss:[rsp+24],69142F39
00007FF76B8AC424 6548:8B3425 60000000 mov_rsi,qword_ptr_gs:[60]
00007FF76B8AC42D 48:8805 14ED0500 mov_rax,qword_ptr_ds:[7FF76B90B248]
00007FF76B8AC434 48:C7C7 D044C885 mov_rdi,FFFFFFFF85C84400
00007FF76B8AC43B 48:8B80 B41B0DBB mov_rax,qword_ptr_ds:[rax-44F2E44C]
00007FF76B8AC442 48:01F8 add_rax,rdi
00007FF76B8AC445 8B9E BC000000 mov_ebx,dword_ptr_ds:[rsi+8C]
00007FF76B8AC44B FFD0 call_rax
00007FF76B8AC44D 48:8805 F4ED0500 mov_rax,qword_ptr_ds:[7FF76B90B248]
00007FF76B8AC454 48:8B80 BC1B0DBB mov_rax,qword_ptr_ds:[rax-44F2E444]
00007FF76B8AC45B 48:01F8 add_rax,rdi
00007FF76B8AC45E FFD0 call_rax
00007FF76B8AC460 48:8805 E1ED0500 mov_rax,qword_ptr_ds:[7FF76B90B248]
00007FF76B8AC467 48:03B8 C41B0DBB add_rdi,qword_ptr_ds:[rax-44F2E43C]
00007FF76B8AC46E FFD7 call_rdi
00007FF76B8AC470 807E 02 00 cmp_byte_ptr_ds:[rsi+2],0
00007FF76B8AC474 8F95 1121 22 setne_byte_ptr_ss:[rsp+22]
00007FF76B8AC479 85DB test_ebx,ebx
00007FF76B8AC47B 0F954424 23 setne_byte_ptr_ss:[rsp+23]
00007FF76B8AC480 41:BE 987E58C1 mov_r14d,C1587E98
00007FF76B8AC486 48:8805 C3ED0500 mov_rax,qword_ptr_ds:[7FF76B90B250]
00007FF76B8AC48D 41:B8 40000000 mov_r8d,40
00007FF76B8AC493 BA 78E13833 mov_edx,3338E178
00007FF76B8AC498 45:3108 xor_r14d,r14d

```

Anti Debug Check

- The code highlighted in the image above reads data at the offset 0x60 from segment register **GS**. Windows stores the **Thread Information Block (TIB)** in **FS** [x86] and **GS** [x64] segment registers.
- The TIB holds the **Process Environment Block (PEB)** at the offset 0x60. The malware accesses PEB of the process via the GS register.
- Later the malware reads the data at the offset 0x2 in PEB (ds:[rsi+2]), which is the **BeingDebugged** member in the PEB structure, and then compares the obtained value with 0. If the process is being debugged then BeingDebugged will have a non zero value. If the test fails, the malware goes into an infinite loop and does not proceed further.

Evasion Techniques

Instrumentation Callback Bypass

The security endpoints (especially ETWTi) of a device use the instrumentation callback process to check for behavioral anomalies and detect novel malware on the system. Pandora ransomware bypasses such a callback mechanism via `ntsetinformationprocess`, which changes the process information.

- `ntsetinformationprocess` is invoked with `ProcessInstrumentationCallback` as a part of `ProcessInformationClass`.

```

1: rcx FFFFFFFFFFFFFFFF
2: rdx 0000000000000028
3: r8 00000043D24FF930
4: r9 0000000000000010
5: [rsp+28] 0000000000000000

```

ntsetinformationprocess being invoked

- The third argument in the above image is a 10-byte long structure associated with the provided ProcessInstrumentationCallback information class.

Address	Hex	ASCII
00000043D24FF930	00 00 00 00 00 00 00 00 00 00 00 00
00000043D24FF940	70 00 00 00 00 00 00 00 D0 44 C8 85 FF FF FF FF	p.....DDÉ.ÿÿÿÿ
00000043D24FF950	00 80 3D D2 43 00 00 00 4D C4 8A 6B F7 7E 00 00	..=0C...MÄ.k÷...
00000043D24FF960	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043D24FF970	5B E1 90 6B F7 7E 00 00 88 B2 82 EF FE 7F 00 00	[ä.k÷...=.ip...
00000043D24FF980	5B E1 90 6B 39 2F 14 69 82 62 8D 6B F7 7E 00 00	[ä.k9/.i.b.k÷...
00000043D24FF990	FD B8 0B 26 00 00 00 00 BC CA F7 AA FF FF FF FF	y.&...¼É=ÿÿÿÿ
00000043D24FF9A0	38 01 00 00 00 00 00 00 F8 29 65 AE FF FF FF FF	8.....ø)eÿÿÿÿ
00000043D24FF9B0	90 01 00 00 00 00 00 00 67 6D 8A 6B F7 7E 00 00qm.k÷...

The third argument (10-byte long structure)

- The members and associated values in the structure are as follows:
 - Version=0 (0 for x64, 1 for x86)
 - Reserved=0
 - Callback=0

If the process created for the malware is hooked by security services via callback member, invoking the ntsetinformationprocess in a way mentioned above with callback set to 0, it helps the malware bypass such hooks.

Event Tracing Bypass

Event Tracing for Windows (ETW) is a powerful tracing facility built into the operating system, to monitor various activities of both userland and kernel land applications running on the system. This feature has become a vital instrument to endpoint security solutions to detect anomalous behavior in running programs. As a result, malware developers have started integrating functionalities in their malware to neutralize the tracing capability. One such vector is patching ETW related functions defined in ntdll.dll in the memory.

- The ransomware dynamically loads **ntdll.dll** into the memory and deobfuscates the string “ EtwEventWrite ”.

```

RAX 000000007D7BA4D0
RBX 0000000000000000
RCX 00007FF76B90E1B5 "EtwEventWrite"
RDX 00007FF76B8E87AE sample.00007FF76B8E87AE
RBP 000000005D299DB2
RSP 000000675D8FFC08
RSI 0000000000000000
RDI 0000000000000000

```

Deobfuscation of “EtwEventWrite”

- The address of the EtwEventWrite function is obtained using **GetProcAddress** API. Getting the function address is a very important step in patching, to bypass the ETW feature.

- Before the malware commences patching, the memory protections on the region of committed pages, where EtwEventWrite resides in virtual address space, need to be changed, which is done via **VirtualProtectEx** API.
- The memory region of pages where the first instruction of EtwEventWrite resides is changed to **PAGE_EXECUTE_READWRITE** to be patched.

```

1: rcx FFFFFFFFFFFFFFFF
2: rdx 00007FFEF28126B0 "L<ÜHfîXM%Kè3AE%CàE3ÉI%C0E3AI%CDf%D$ è["
3: r8 0000000000000001
4: r9 0000000000000040
5: [rsp+20] 000000675D8FFC6C
    
```

Arguments passed to VirtualProtectEx

- The **WriteProcessMemory** API is used to write one byte at the beginning of the EtwEventWrite function. The second argument points to the beginning of EtwEventWrite, and the third argument is the one byte long payload that gets written at the address of EtwEventWrite.

```

1: rcx FFFFFFFFFFFFFFFF
2: rdx 00007FFEF28126B0 "L<ÜHfîXM%Kè3AE%CàE3ÉI%C0E3AI%CDf%D$ è["
3: r8 00000050018FF867 "Ape}ú "
4: r9 0000000000000001
5: [rsp+20] 0000000000000000
    
```

The data passed to WriteProcessMemory

- The one byte payload is **0xC3**, which is the opcode for the instruction “**ret**”. This makes EtwEventWrite to simply return back to the caller function, without executing its logic to log an event when EtwEventWrite is invoked by other applications.

000050018FF862	0000	add byte ptr ds:[rax],a]
000050018FF864	F77F 00	idiv dword ptr ds:[rdi]
000050018FF867	C3	ret
000050018FF868	FE	pushfq
000050018FF869	9C	pushfq
000050018FF86A	7D FA	jge 50018FF866
000050018FF86C	2000	and byte ptr ds:[rax],a]

One byte payload – 0xC3

- After patching, the memory protection of EtwEventWrite is reverted back to the initial permission of **PAGE_EXECUTE_READ** via VirtualProtectEx.

```

1: rcx FFFFFFFFFFFFFFFF
2: rdx 00007FFEF28126B0 "L<ÜHfîXM%Kè3AE%CàE3ÉI%C0E3AI%CDf%D$ è["
3: r8 0000000000000001
4: r9 0000000000000020
5: [rsp+20] 0000000000000000
    
```

Memory protection of EtwEventwrite

Pre-encryption Phase

Before the encryption begins, the malicious software changes the shutdown parameters for the system via **SetProcessShutdownParameters** API. This function sets a shutdown order for the calling process relative to the other processes in the system. Here, the malware invokes the API with zero value so that the ransomware program is the last to shut down by the Operating System.

```

1: rcx 0000000000000000
2: rdx 0000000000000000
3: r8 000000675D8F8B88
4: r9 FFFFFFFFC55244B0
5: [rsp+20] 0000000000000000
6: [rsp+28] 0000000000000000
7: [rsp+30] 0000460FD54013AE
    
```

Data passed to SetProcessShutdownParameters

After setting these shutdown parameters, the malware empties the recycle bin via **SHEmptyRecyclebinA** API.

The ransomware raises the priority of the running process to the highest possible priority which is **REALTIME_PRIORITY_CLASS** via **SetPriorityClass** API. The second argument is the “dwPriorityClass” parameter which has a value of 0x100.

```

1: rcx FFFFFFFFFFFFFFFF
2: rdx 0000000000000100
3: r8 0000025A61275801
4: r9 0000000000000001
5: [rsp+20] 0000000000000000
6: [rsp+28] 0000000000000000
7: [rsp+30] 0000460FD54013AE
    
```

Data passed to SetPriorityClass

Finally, the volume shadow copies are deleted by executing a string of commands via **ShellExecuteA**. It uses vssadmin to perform the task of deleting the shadow files.

```

1: rcx 0000000000000000
2: rdx 00007FF76B8E7A26 L"open"
3: r8 00007FF76B8E7A16 L"cmd.exe"
4: r9 00007FF76B8E79C0 L"/c vssadmin.exe delete shadows /all /quiet"
5: [rsp+20] 0000000000000000
6: [rsp+28] 0000000000000000
7: [rsp+30] 0000000000000001
    
```

Deleting shadow files using vssadmin

Encryption Phase: Threading Model

The main thread of malware creates two new threads that are responsible for the encryption of user data.

Number	ID	Entry	TEB	RIP	Suspend Count
3	10892	00007FFEF27F3D60	000000675D717000	00007FFEF285FA64	1
Main	11004	00007FF76B918C40	000000675D711000	00007FFEF285D204	1
1	7308	00007FFEF27F3D60	000000675D713000	00007FFEF285FA64	1
2	4436	00007FFEF27F3D60	000000675D715000	00007FFEF285FA64	1
6	860	00007FFEF27F3D60	000000675D71D000	00007FFEF285FA64	1
4	12200	00007FFEF12D7870	000000675D719000	00007FFEF285CC24	1
5	8072	00007FFEF27F3D60	000000675D71B000	00007FFEF285FA64	1
7	9504	00007FFEF27F3D60	000000675D71F000	00007FFEF285FA64	1
8	5052	00007FFEF1A1ACA0	000000675D721000	00007FFEEFB39A84	1
9	3160	00007FF76B8A4D60	000000675D725000	00007FFEF0DE2170	1
10	1456	00007FF76B8A4D60	000000675D723000	00007FFEF285C154	1

Creation of two new threads

The following APIs are used to create the threads:

- **CreateThread**

- **SetThreadAffinityMask**
- **ResumeThread**

The threads are created with `dwCreationFlags` set to **CREATE_SUSPENDED**, later the execution of threads is resumed via **ResumeThread**.

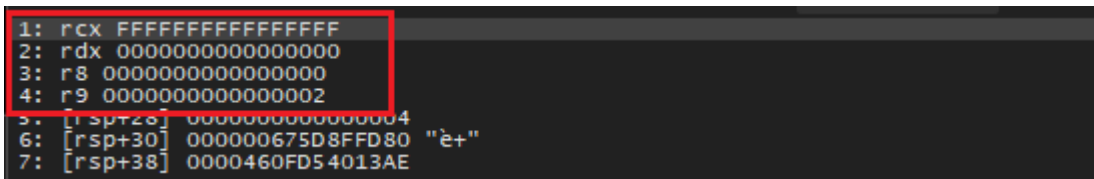
The main thread starts to enumerate the drives present on the system via the following APIs:

- **GetDriveTypeW**
- **FindFirstVolumeW**
- **GetVolumePathNamesForVolumeNameW**
- **SetVolumeMountPointW**
- **FindNextVolumeW**
- **GetLogicalDrives**

Pandora utilizes Windows I/O Completion Ports to efficiently speed up the encryption process. Following APIs are used to orchestrate the search and locking of the user data:

- **CreateIoCompletionPort**
- **PostQueuedCompletionStatus**
- **GetQueuedCompletionPort**

Initially, the main thread of the malware creates an input/ output (I/O) completion port via `CreateIoCompletionPort` API.

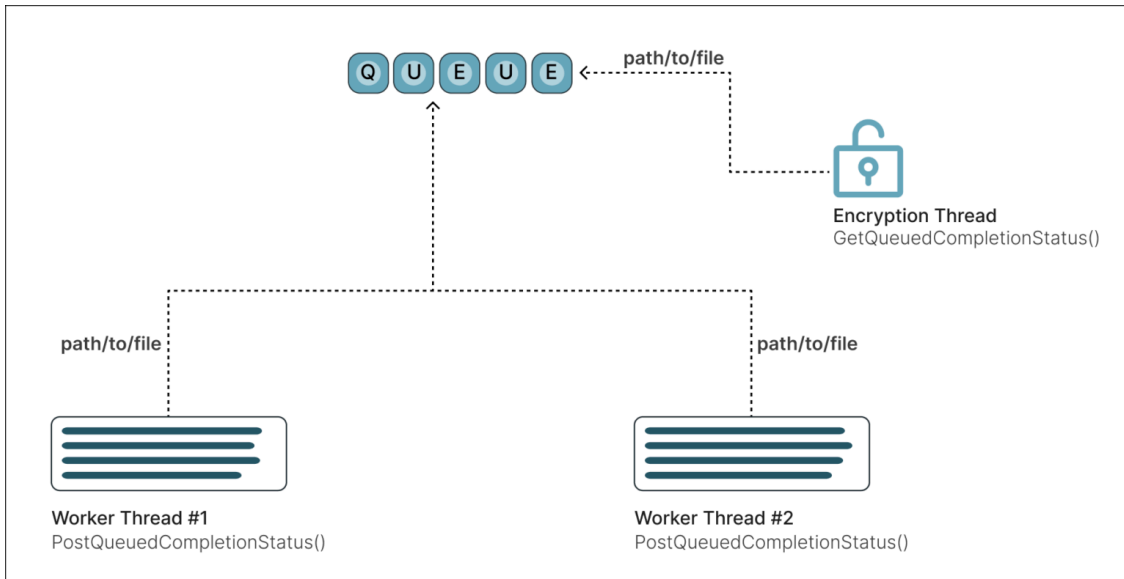


```
1: rcx FFFFFFFFFFFFFFFF
2: rdx 0000000000000000
3: r8 0000000000000000
4: r9 0000000000000002
5: [rsp+28] 0000000000000004
6: [rsp+30] 000000675D8FFD80 "è+"
7: [rsp+38] 0000460FD54013AE
```

Data passed to `CreateIoCompletionPort`

- The fourth argument is “NumberOfConcurrentThreads”. In our case, two threads are allowed to concurrently process I/O completion packets for the I/O completion port.
- After the creation of the I/O port, a queue is created internally, to which threads can push the completion status.
- The two threads created previously will be accessing I/O ports to perform file enumeration and encryption on the infected system.

In general, ransomware in the wild has adopted a model to optimize the encryption process. The goal here is to efficiently utilize the power of multicore processors to concurrently perform file enumeration and encryption. A group of worker threads would fetch the file paths and post them in the queue via **PostQueuedCompletionStatus**, and another thread can retrieve the posted files (paths) for encryption via **GetQueuedCompletionStatus**.



Optimization of the encryption process

Pandora uses the RSA 4096 algorithm for encryption, the public key is embedded within the malware.

```

00007FF76B90C138 49 4E 20 50 55 42 4C 49 43 20 48 45 59 2D 2D 2D IN PUBLIC KEY---
00007FF76B90C148 2D 2D 0A 4D 49 49 42 49 6A 41 4E 42 67 6B 71 68 --.MIIBIjANBgkqh
00007FF76B90C158 6B 69 47 39 77 30 42 41 51 45 46 41 41 4F 43 41 kiG9w0BAQEFAAOCA
00007FF76B90C168 51 38 41 4D 49 49 42 43 67 4B 43 41 51 45 41 34 Q8AMIIBCgKCAQEA4
00007FF76B90C178 4D 63 64 31 55 76 66 57 71 6E 50 57 68 53 2B 39 Mcd1UvfwqnPwhS+9
00007FF76B90C188 70 49 69 0A 74 56 37 39 32 30 65 4D 30 4B 35 2B pIi.tv7920eM0K5+
00007FF76B90C198 7A 6A 4E 6A 4B 70 72 74 57 7A 79 30 62 2F 7A 43 zjNjKprtWzy0b/zC
00007FF76B90C1A8 41 2B 52 4A 68 33 69 4D 71 4B 68 79 42 4C 56 46 A+Rjh3iMqKhyBLVF
00007FF76B90C1B8 38 71 6F 6C 5A 64 52 73 6B 6C 72 70 32 75 58 4E 8qo1ZdRsk1rp2uXN
00007FF76B90C1C8 52 49 78 46 0A 74 73 49 6B 4E 32 63 42 39 56 2F RIXF.tsIKN2cB9V/
00007FF76B90C1D8 65 58 36 51 62 61 6B 75 4E 59 6F 6B 34 33 73 45 eX6QbakuNYok43sE
00007FF76B90C1E8 6A 49 45 5A 64 42 33 72 5A 49 4B 56 4F 32 31 58 jIEZdB3rZIKV021X
00007FF76B90C1F8 63 7A 78 46 6B 57 55 5A 70 61 46 39 35 42 7A 51 czxFkWUZpaF95BzQ
00007FF76B90C208 74 61 62 39 77 0A 56 4A 2F 67 44 39 6A 75 6D 73 tab9w.VJ/gD9jums
00007FF76B90C218 50 50 30 33 74 65 56 59 58 6E 4F 33 31 62 6A 63 PPO3teVYXn031bjc
00007FF76B90C228 54 56 2F 37 76 46 6E 34 48 50 63 37 49 4F 42 45 TV/7vFn4HPC7IOBE
00007FF76B90C238 74 55 78 61 4D 58 31 6E 52 34 72 73 78 4A 46 4A tUxaMX1nr4rsxJfJ
00007FF76B90C248 52 6B 36 43 37 56 0A 43 31 71 66 36 54 4B 53 43 Rk6C7V.C1qf6TKSC
00007FF76B90C258 32 37 59 44 2B 37 34 56 32 77 70 7A 2F 38 6F 73 27YD+74V2wpz/8os
00007FF76B90C268 33 48 76 57 39 77 6B 58 66 32 61 64 42 2F 6A 56 3HvW9wkXf2adB/jv
00007FF76B90C278 4D 63 65 6E 56 4D 79 6F 51 55 4C 65 36 34 73 67 McenVMyoQULe64sg
00007FF76B90C288 68 42 30 67 45 76 4D 0A 51 35 72 4C 4C 76 44 39 hB0gEvM.Q5rLLvD9
00007FF76B90C298 79 48 53 64 2F 58 54 73 2B 61 66 47 46 57 68 76 yHsd/XTs+afGFwhv
00007FF76B90C2A8 71 70 55 46 45 34 53 2B 57 2F 44 63 39 73 54 70 qpUFE4S+W/Dc9sTp
00007FF76B90C2B8 44 32 6F 43 57 6F 50 35 47 4D 59 70 6F 53 4C 48 D2cWoP5GMyp0SLH
00007FF76B90C2C8 35 32 34 78 34 68 54 57 0A 63 51 49 44 41 51 41 524x4htw.cQIDAQA
00007FF76B90C2D8 42 0A 2D 2D 2D 2D 45 4E 44 20 50 55 42 4C 49 B.-----END PUBLI
00007FF76B90C2E8 43 20 4B 45 59 2D 2D 2D 2D 2D 0A 00 00 00 00 00 C KEY-----.....
00007FF76B90C2F8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
    
```

Public key embedded in the malware

As a prior step to the encryption process, the malware accesses directories in the network drives and dumps the ransom note (**Restore_My_Files.txt**). The ransom note is created using the following three APIs:

- **CreateFileW**
- **WriteFileW**
- **CloseHandle**

Address	Hex	ASCII
00007FF76B90DB80	23 23 23 20	57 68 61 74
00007FF76B90DB90	64 3F 0D 0A	0D 0A 23 23
00007FF76B90DBA0	75 72 20 66	69 6C 65 73
00007FF76B90DBB0	72 79 70 74	65 64 21 21
00007FF76B90DBC0	6C 20 79 6F	75 72 20 66
00007FF76B90DBD0	20 70 72 6F	74 65 63 74
00007FF76B90DBE0	72 6F 6E 67	20 65 6E 63
00007FF76B90DBF0	77 69 74 68	20 52 53 41
00007FF76B90DC00	0A 2A 54 68	65 72 65 20
00007FF76B90DC10	62 6C 69 63	20 64 65 63
00007FF76B90DC20	73 6F 66 74	77 61 72 65
00007FF76B90DC30	68 61 76 65	20 73 75 63
00007FF76B90DC40	79 20 73 74	6F 6C 65 6E
00007FF76B90DC50	6E 66 69 64	65 6E 74 69
00007FF76B90DC60	65 6E 74 20	64 61 74 61
00007FF76B90DC70	65 73 2C 20	65 6D 61 69
00007FF76B90DC80	6F 79 65 65	20 69 6E 66
00007FF76B90DC90	2C 20 63 75	73 74 6F 6D
00007FF76B90DCA0	65 61 72 63	68 20 61 6E
00007FF76B90DCB0	70 6D 65 6E	74 20 70 72
00007FF76B90DCC0	2F 2A 6D 6A	6D 6A 23 23

Contents of the ransom note

Encryption Process

The process explained in this section is executed by worker threads highlighted in the image below. These threads can concurrently enumerate and encrypt data via the Windows I/O completion port.

Number	ID	Entry	TEB	RIP	Suspend Count
3	10892	00007FFEF27F3D60	000000675D717000	00007FFEF285FA64	1
Main	11004	00007FF76B91BC40	000000675D711000	00007FFEF285D204	1
1	7308	00007FFEF27F3D60	000000675D713000	00007FFEF285FA64	1
2	4436	00007FFEF27F3D60	000000675D715000	00007FFEF285FA64	1
6	860	00007FFEF27F3D60	000000675D71D000	00007FFEF285FA64	1
4	12200	00007FFEF12D7870	000000675D719000	00007FFEF285CC24	1
5	8072	00007FFEF27F3D60	000000675D71B000	00007FFEF285FA64	1
7	9504	00007FFEF27F3D60	000000675D71F000	00007FFEF285FA64	1
8	5052	00007FFEF1A1ACA0	000000675D721000	00007FFEFB39A84	1
9	3160	00007FF76B8A4D60	000000675D725000	00007FFEF0DE2170	1
10	1456	00007FF76B8A4D60	000000675D723000	00007FFEF285C154	1

Worker Threads

- After dumping the ransom note, the malware uses `FindFirstFileW` to open a handle to the files on the disk.
- The retrieved handle is checked against a set of directory names and file extensions.
- The following directories are excluded from getting locked:

AppData	Opera Software
Boot	Mozilla
Windows.old	Mozilla Firefox
Tor Browser	ProgramData
Internet Explorer	Program Files
Google	Program Files (x86)
Opera	#recycle

- The following files are excluded from getting encrypted:

Autorun.inf	bootmgfw.efi
boot.ini	desktop.ini
bootfont.bin	iconcache.db
bootsect.bak	ntldr
bootmgr	Ntuser.dat
bootmgr.efi	Restore_My_Files.txt

- And the following extensions are excluded from getting locked:

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

- After performing exclusion checks, the absolute path of the file that passed the check is computed and then the thread calls for **PostQueuedCompletionStatus** to submit the path to the I/O queue previously created via **CreateIoCompletionPort**.
- Right after the **PostQueuedCompletionStatus** call, the same worker thread can resume fetching the absolute path of the next file via **FindNextFileW** API.
- Another worker thread can now call **GetQueuedCompletionStatus** to retrieve the absolute path of the target file to start encrypting the files.
- Next, the file attribute is changed via **SetFileAttributesW** API to **FILE_ATTRIBUTE_NORMAL** and then the file is fetched for encryption via the following APIs:
 - **CreateFileW**
 - **GetFileSizeEx**
 - **ReadFile**
 - **SetFilePointerEx**

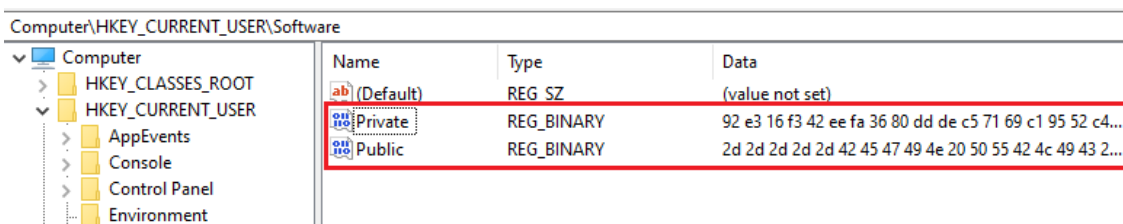
- After setting up the file pointer to the target data, the encryption begins by loading the public key in the memory, and the encrypted data is written to the file via **WriteFile** API. Later the file is renamed via **MoveFileExW** API to add “.pandora” extension to the encrypted file.

```

RAX 00007FF7195F6890
RBX 0000000000000188 L'c'
RCX 0000025A63EB4E84 L"C:\\Users\\[redacted]\\Downloads\\[redacted].zip"
RDY 0000025A612A8F90 L"C:\\Users\\[redacted]\\Downloads\\[redacted].zip.pandora"
RBP 00007FF7688D6276 <sample.JMP.&MoveFileExW>
RSP 000000675E2FF8B0
RSI 0000000000000380
RDI 00000000000004A8 L'q'
    
```

Renamed file with the “.pandora” extension

Registry Keys



HKCU registry key

Pandora ransomware writes two values, **Private** and **Public**, under the **HKCU/ Software** registry key. The public value has the public key used by the ransomware to encrypt the user files, while the private value has the protected private key stored for decryption. The decryptor tool that the victim receives after paying the ransom uses this information stored in the registry to decrypt the locked files.

Indicators of Compromise

Binary
5b56c5d86347e164c6e571c86dbf5b1535eae6b979fed6ed66b01e79ea33b7b
Registry
HKCU\Software\Private
HKCU\Software\Public
Dropped Files
Restore_My_Files.txt