A Deep Dive Into ALPHV/BlackCat Ransomware - SecurityScorecard

Archived: 2026-04-05 14:57:49 UTC

Executive summary

ALPHV/BlackCat is the first widely known ransomware written in Rust. The malware must run with an access token consisting of a 32-byte value (–access-token parameter), and other parameters can be specified. The ransomware comes with an encrypted configuration that contains a list of services/processes to be stopped, a list of whitelisted directories/files/file extensions, and a list of stolen credentials from the victim environment. It deletes all Volume Shadow Copies, performs privilege escalation using the CMSTPLUA COM interface, and enables “remote to local” and “remote to remote” symbolic links on the victim’s machine. The files are encrypted using the AES algorithm, with the AES key being encrypted using the RSA public key contained in the configuration. The extension of the encrypted files is changed to uhwuvzu by the malware.

Analysis and findings

SHA256: 847fb7609f53ed334d5affbb07256c21cb5e6f68b1cc14004f5502d714d2a456 The malware can run with one of the following parameters:

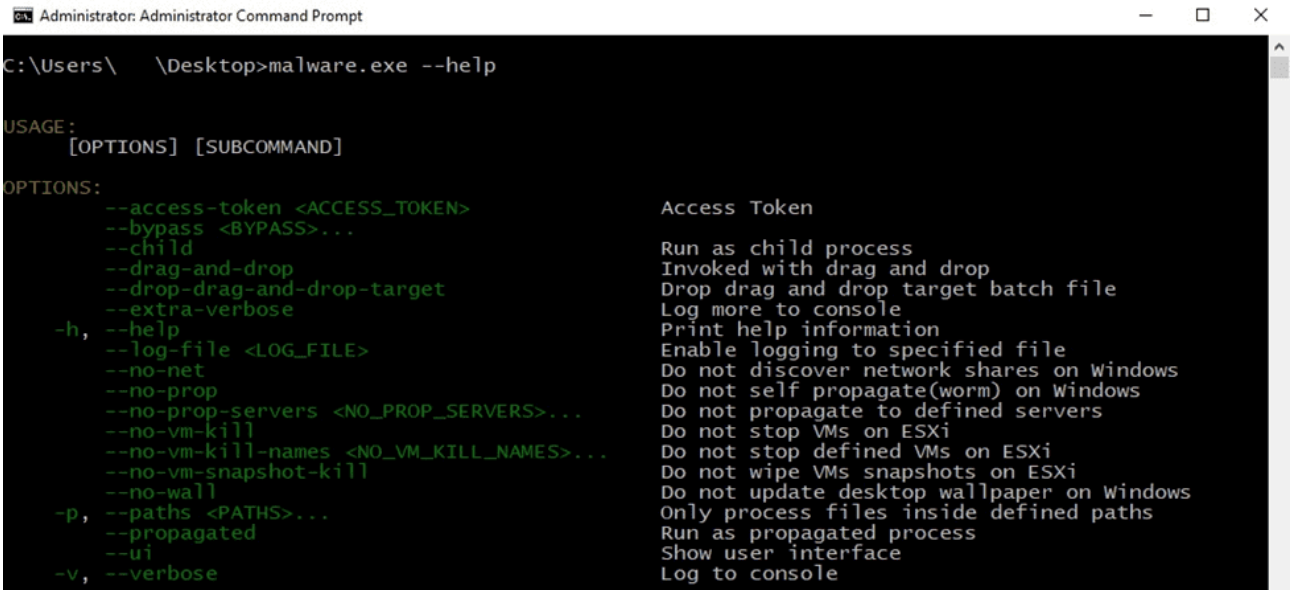


Figure 1 Whether the ransomware is running with no parameters or with an invalid access token, an error message is displayed:

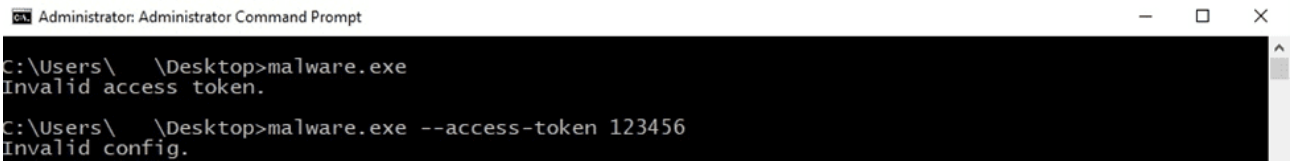


Figure 2 By performing the dynamic analysis, we’ve found that the access token must be a 32-byte value that is

not unique. The binary registers a new top-level exception handler via a function call to SetUnhandledExceptionFilter:

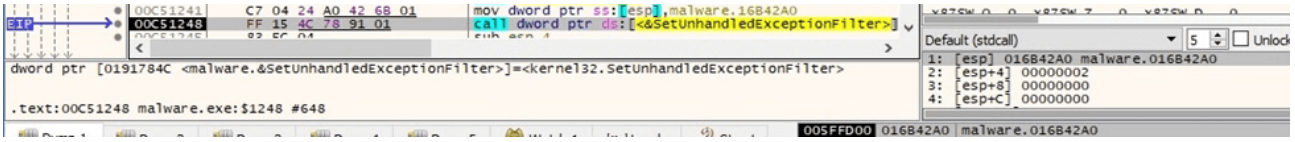


Figure 3 The AddVectoredExceptionHandler API is utilized to register a vectored exception handler:

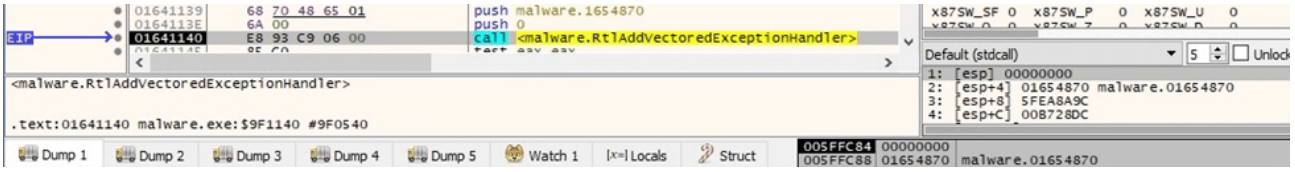


Figure 4 The executable retrieves the command-line string for the process using the GetCommandLineW function:

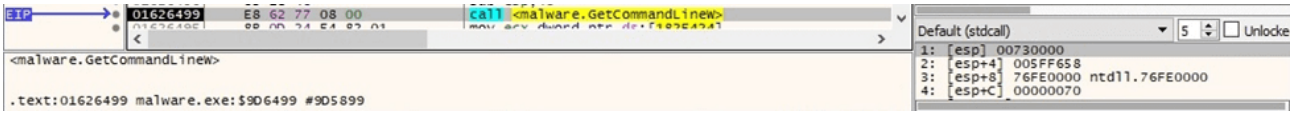


Figure 5 BlackCat opens the “SOFTWARE\Microsoft\Cryptography” registry key by calling the RegOpenKeyExW routine (0x80000002 = **HKEY_LOCAL_MACHINE**, 0x20019 = KEY_READ):

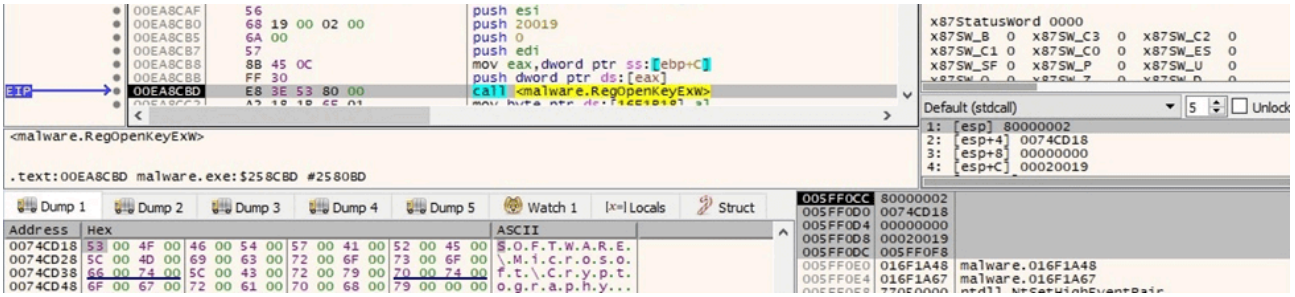


Figure 6 The binary extracts the MachineGUID value from the registry:

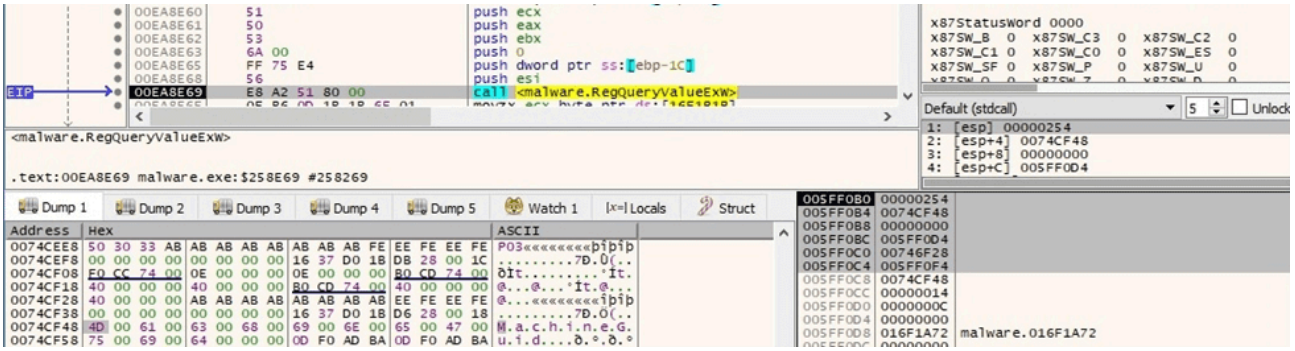


Figure 7 The malicious process searches for cmd.exe in the current directory and then in the System32 directory via a function call to CreateFileW (0x7 = **FILE_SHARE_DELETE | FILE_SHARE_WRITE | FILE_SHARE_READ**, 0x3 = **OPEN_EXISTING**, 0x2000000 = **FILE_FLAG_BACKUP_SEMANTICS**):

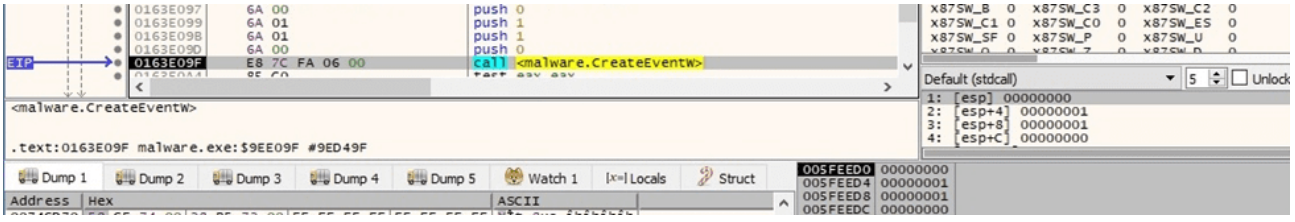


Figure 8 The executable generates 16 random bytes by calling the BCryptGenRandom API (0x2 = **BCRYPT_USE_SYSTEM_PREFERRED_RNG**):

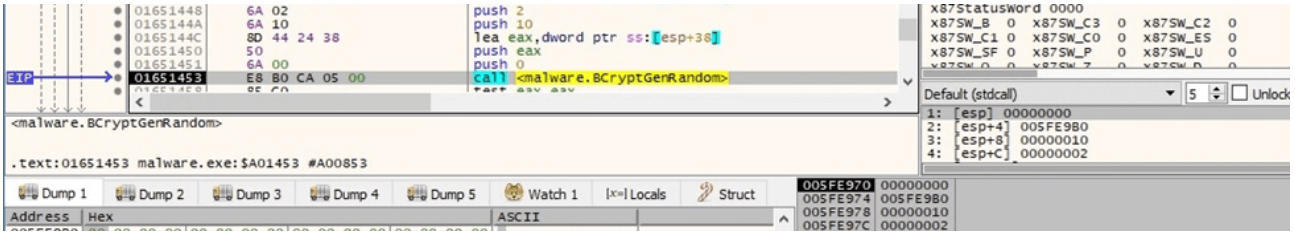


Figure 9 A named pipe whose name contains the current process ID and random bytes generated above is created using CreateNamedPipeW (0x40080001 = **FILE_FLAG_OVERLAPPED | FILE_FLAG_FIRST_PIPE_INSTANCE | PIPE_ACCESS_INBOUND**, 0x8 = **PIPE_REJECT_REMOTE_CLIENTS**):

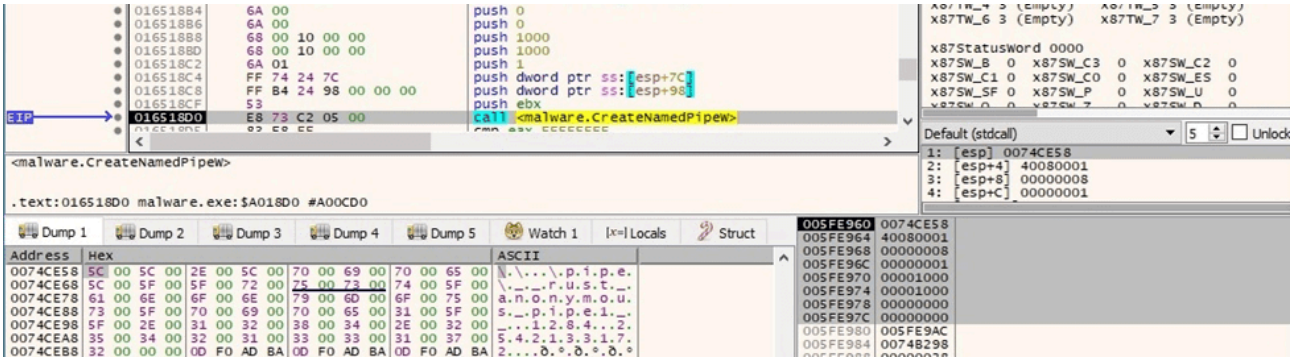


Figure 10 The process opens the named pipe for writing using the CreateFileW routine (0x40000000 = **GENERIC_WRITE**, 0x3 = **OPEN_EXISTING**):

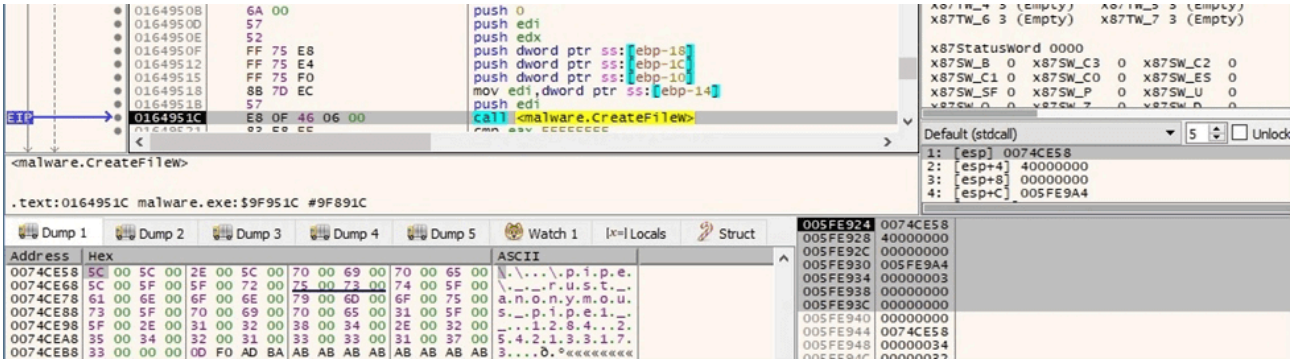


Figure 11 The ransomware creates a read and a write named pipe, respectively. The wmic process is used to extract the UUID (0x08000400 = **CREATE_NO_WINDOW | CREATE_UNICODE_ENVIRONMENT**):

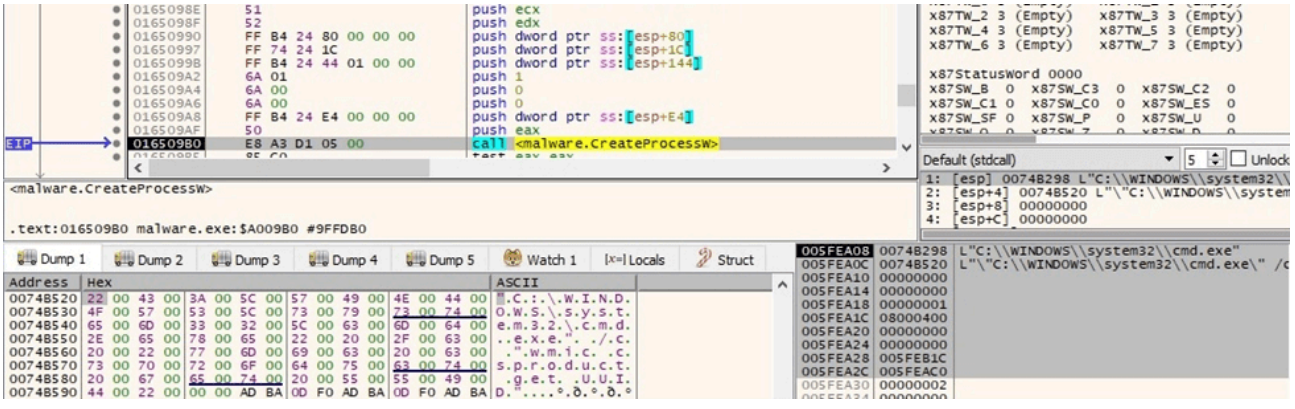


Figure 12 The CreateEventW API is utilized to create two unnamed event objects:

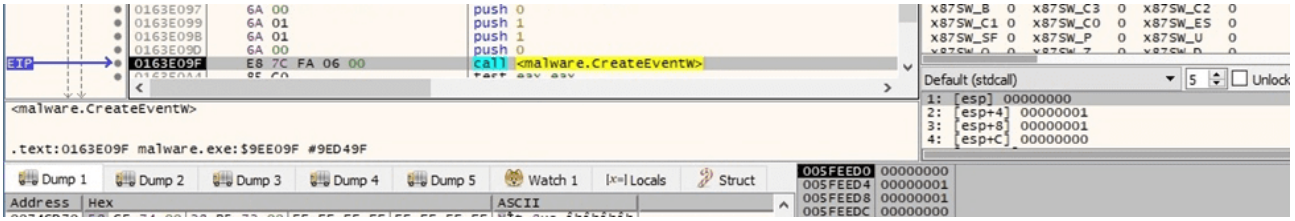


Figure 13 The binary waits until the event objects are in the signaled state by calling WaitForMultipleObjects:

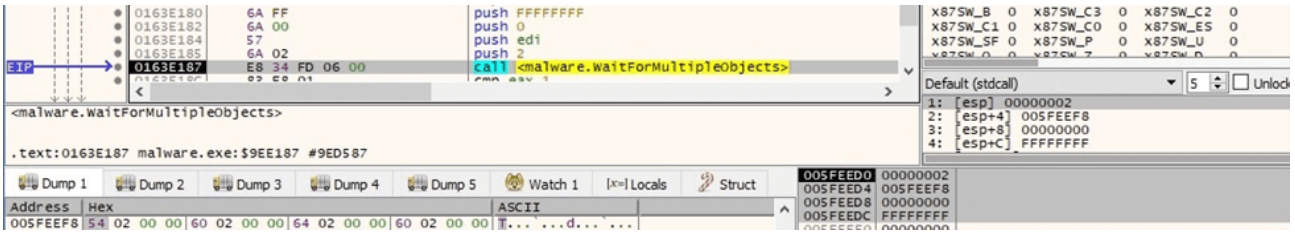


Figure 14 The output of the above process is read from the named pipe using the ReadFile routine:

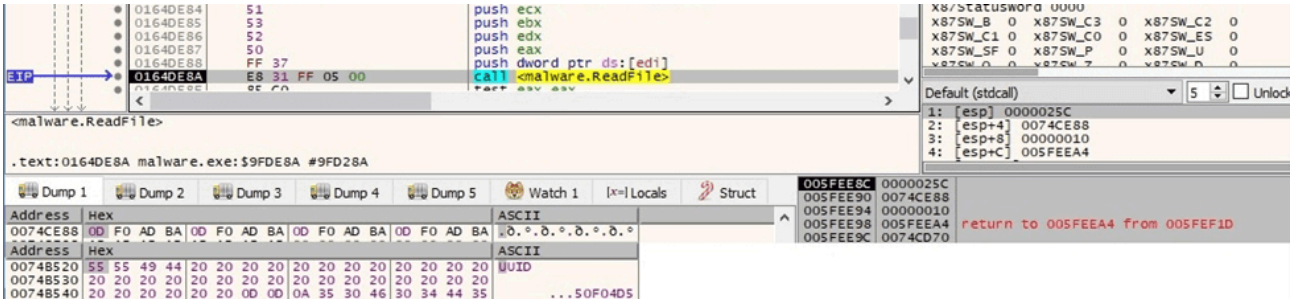


Figure 15 The malware creates multiple threads by calling the CreateThread function (0x00010000 = **STACK_SIZE_PARAM_IS_A_RESERVATION**):

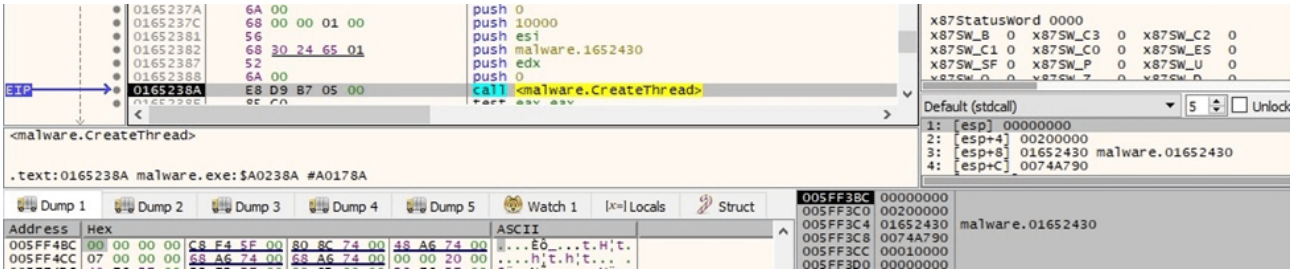


Figure 16 The content of the ransom note and the text that will appear on the Desktop Wallpaper are decrypted by the ransomware:

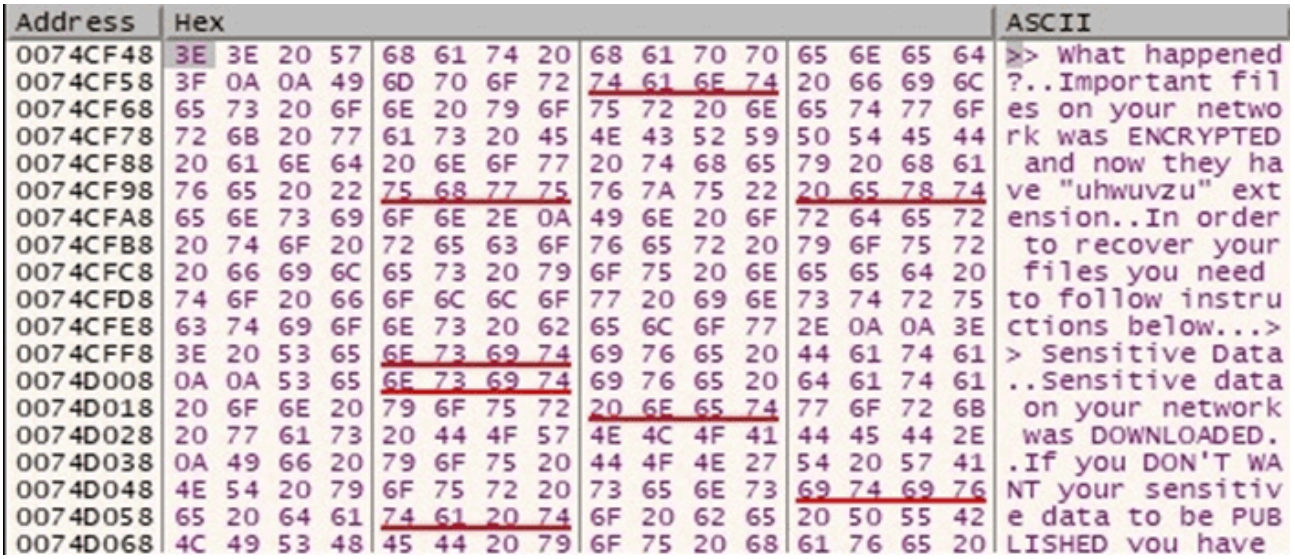


Figure 17

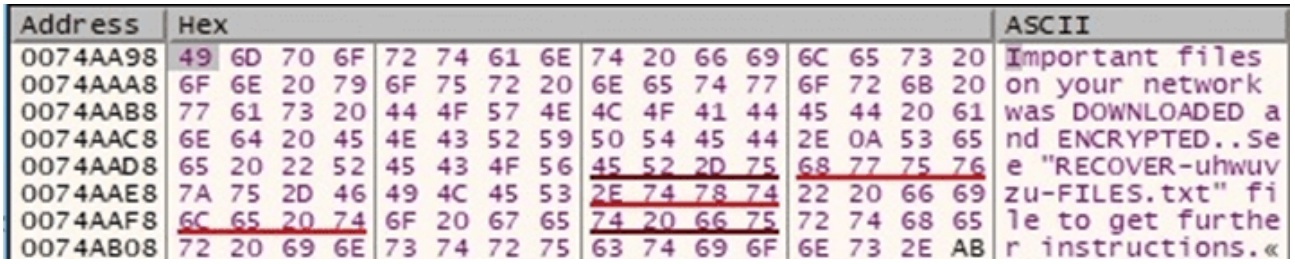


Figure 18 The malicious binary obtains information about the current system via a function call to GetSystemInfo:

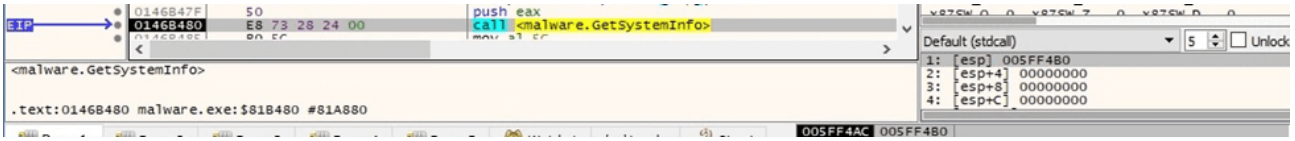


Figure 19 There is a call to SHTestTokenMembership that verifies whether the user token is a member of the Administrators group in the built-in domain (0x220 = **DOMAIN_ALIAS_RID_ADMINS**):

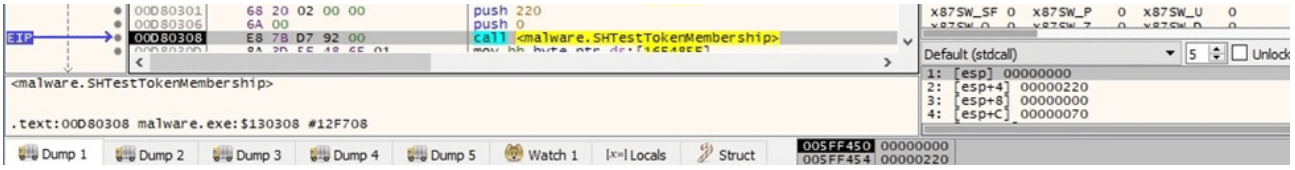


Figure 20 The process opens the access token associated with the current process (0x80000000 = **GENERIC_READ**):

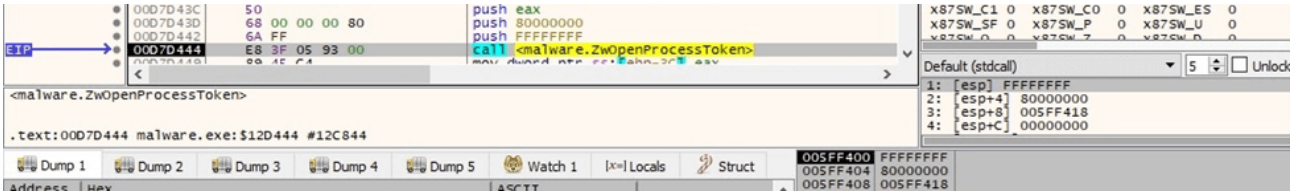


Figure 21 BlackCat extracts a TOKEN_GROUPS structure containing the group accounts associated with the above token using the NtQueryInformationToken function (0x2 = **TokenGroups**):

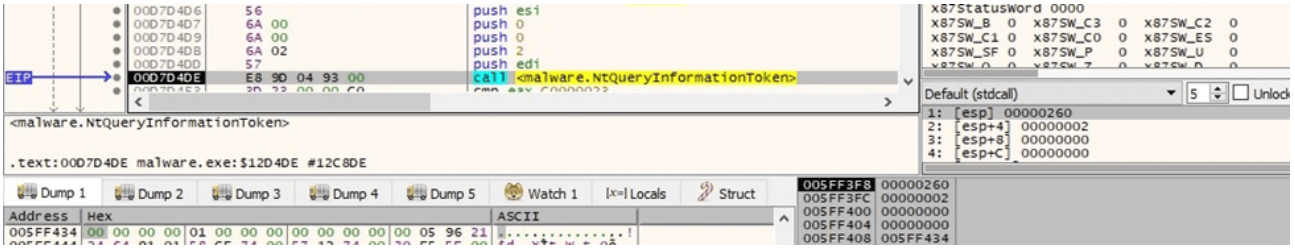


Figure 22 The OpenProcess API is utilized to open a local process object (0x438 = **PROCESS_QUERY_INFORMATION | PROCESS_VM_WRITE | PROCESS_VM_READ | PROCESS_VM_OPERATION**):

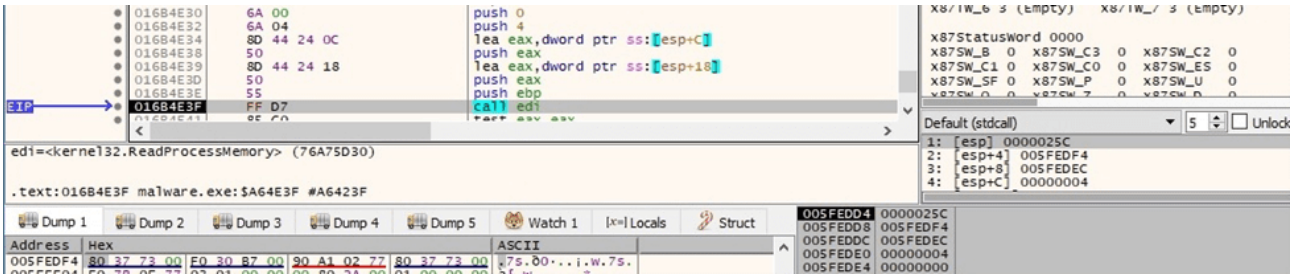


Figure 23 The malicious binary retrieves a pointer to a PEB structure using the ZwQueryInformationProcess routine (0x0 = **ProcessBasicInformation**):

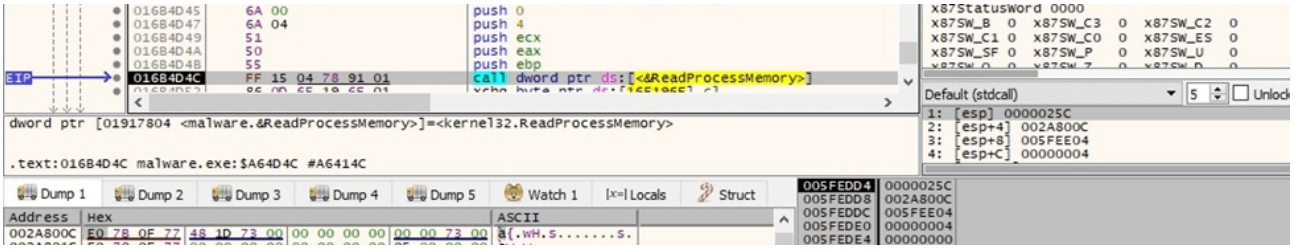


Figure 24 The executable retrieves a pointer to a PEB_LDR_DATA structure containing information about the loaded modules in the process and then to the head of a doubly linked list that contains the loaded modules:

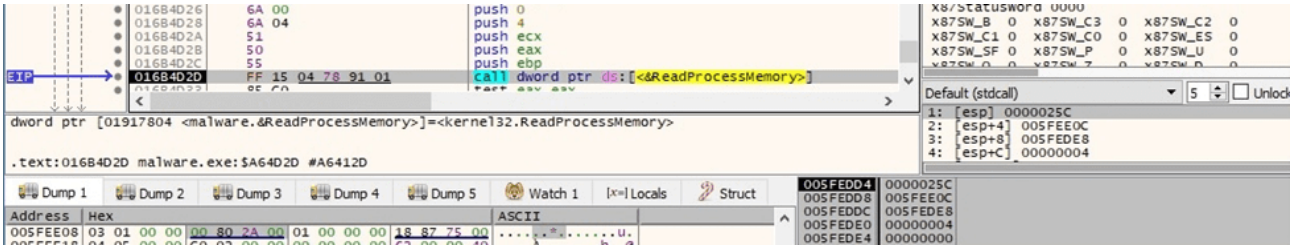


Figure 25

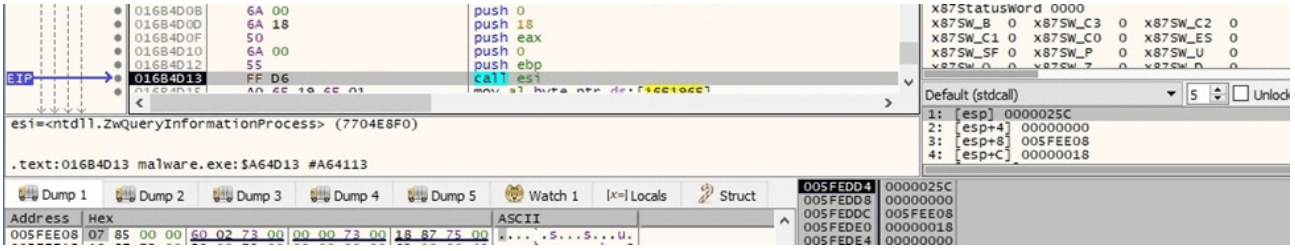


Figure 26 The path of the image file for the current process is retrieved using ReadProcessMemory:

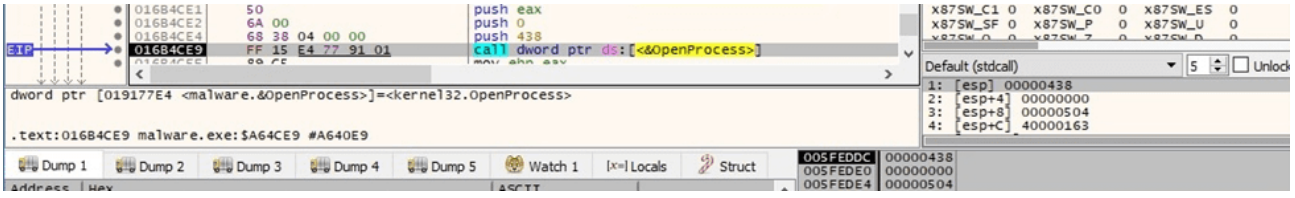


Figure 27

Privilege escalation via UAC bypass using CMSTPLUA COM interface

The ransomware initializes the COM library for use by the current thread via a call to CoInitializeEx (0x2 = **COINIT_APARTMENTTHREADED**):

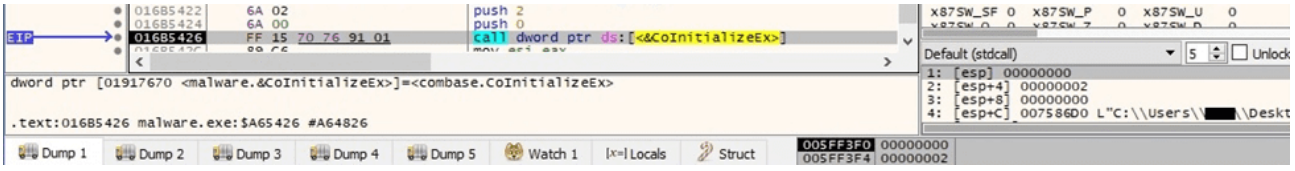


Figure 28 BlackCat ransomware uses the auto-elevated CMSTPLUA interface **{3E5FC7F9-9A51-4367-9063-A120244FBEC7}** in order to escalate privileges:

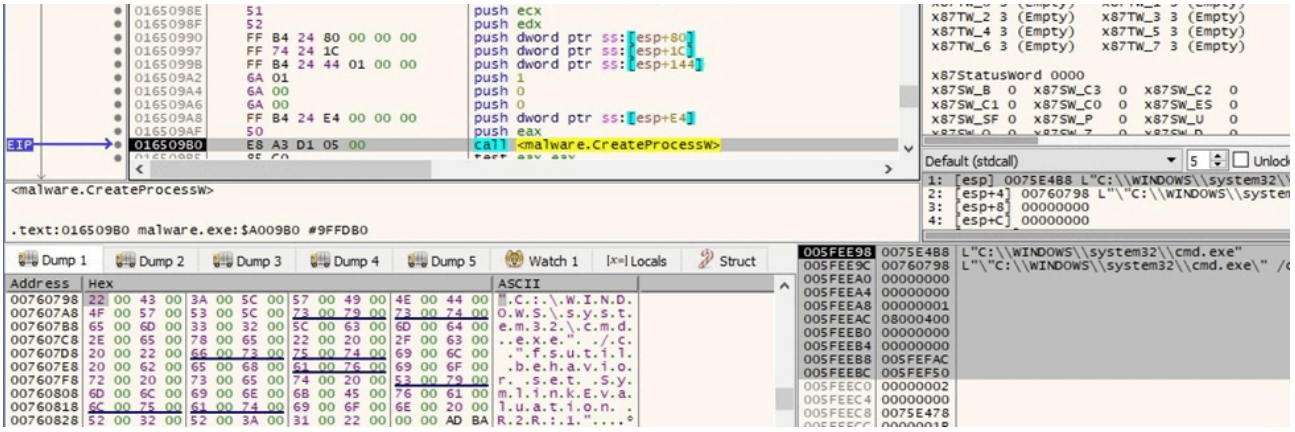


Figure 29 The initial executable is spawned with administrative privileges:

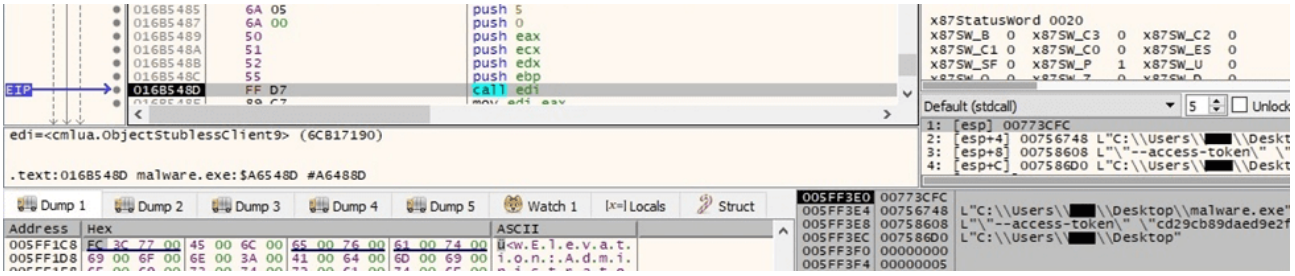


Figure 30 The LookupPrivilegeValueW routine is utilized to retrieve the locally unique identifier that represents the following privileges:

- SeIncreaseQuotaPrivilege SeSecurityPrivilege SeTakeOwnershipPrivilege
- SeLoadDriverPrivilege SeSystemProfilePrivilege SeSystemtimePrivilege
- SeProfileSingleProcessPrivilege SeIncreaseBasePriorityPrivilege
- SeCreatePagefilePrivilege SeBackupPrivilege SeRestorePrivilege
- SeShutdownPrivilege SeDebugPrivilege SeSystemEnvironmentPrivilege
- SeChangeNotifyPrivilege SeRemoteShutdownPrivilege SeUndockPrivilege
- SeManageVolumePrivilege SeImpersonatePrivilege SeCreateGlobalPrivilege
- SeIncreaseWorkingSetPrivilege SeTimeZonePrivilege
- SeCreateSymbolicLinkPrivilege SeDelegateSessionUserImpersonatePrivilege

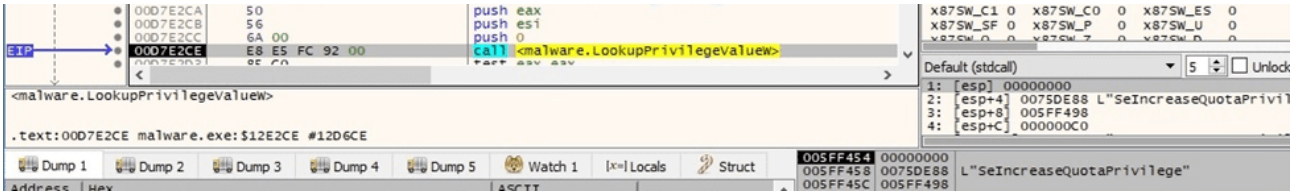


Figure 31 All the above privileges are enabled in the access token using AdjustTokenPrivileges:

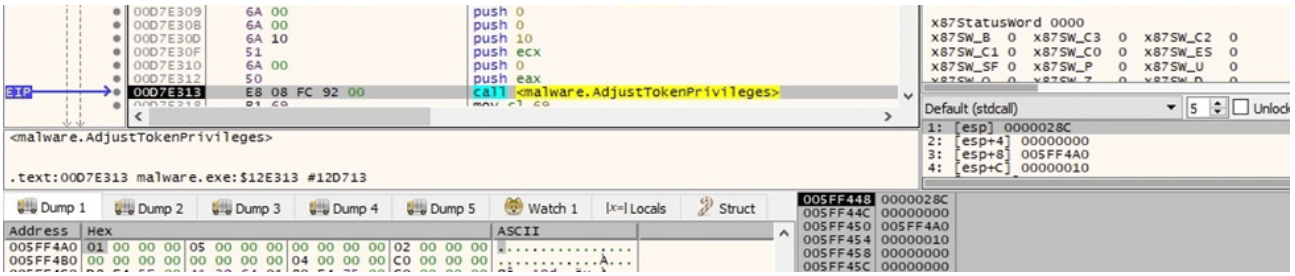


Figure 32 The binary creates the following processes that enable “remote to local” and “remote to remote”

symbolic links on the local machine:

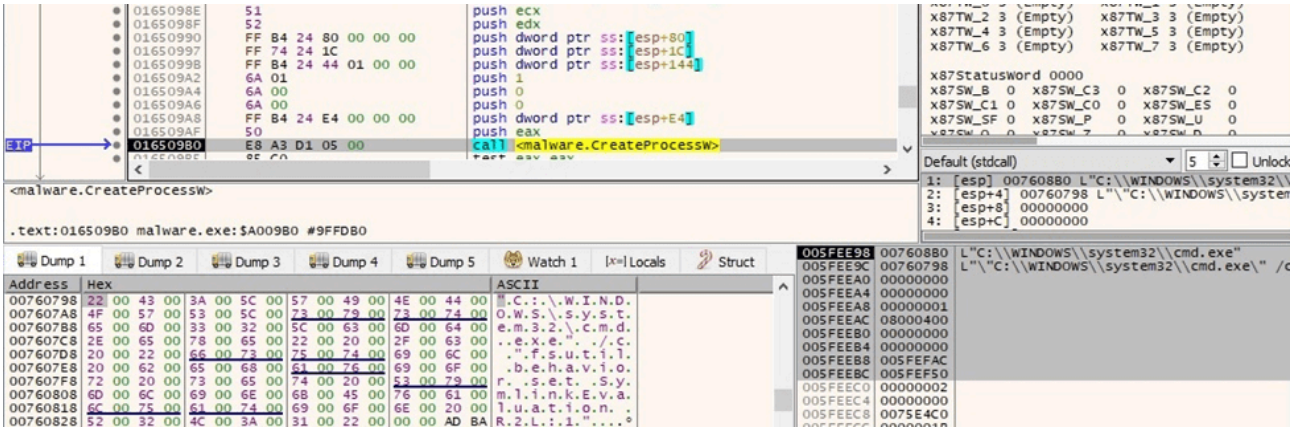


Figure 33

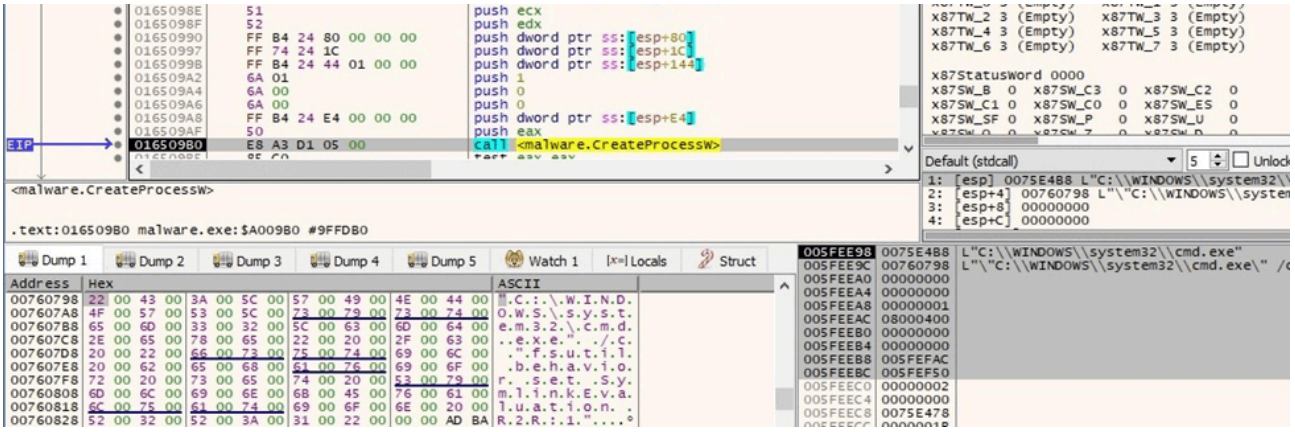


Figure 34 The malware tries to stop the Internet Information service (IIS) using IISReset.exe:

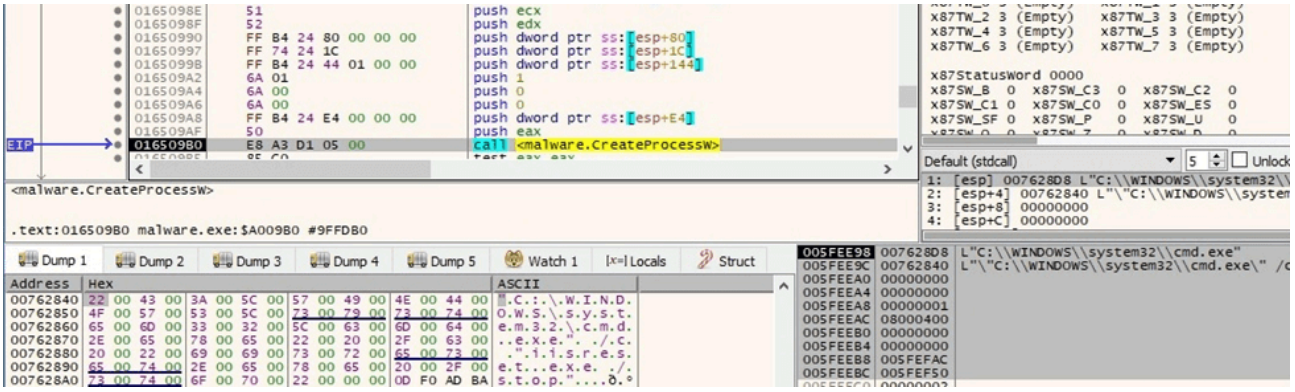


Figure 35 The ransomware deletes all volume shadow copies using the vssadmin.exe utility:

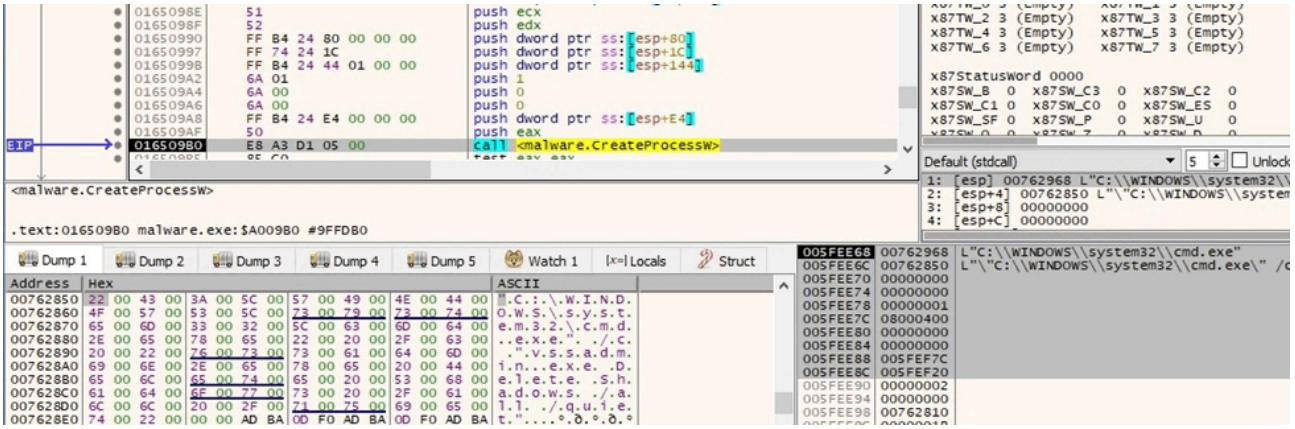


Figure 36 There is also a second process that is responsible for deleting all volume shadow copies with wmic:

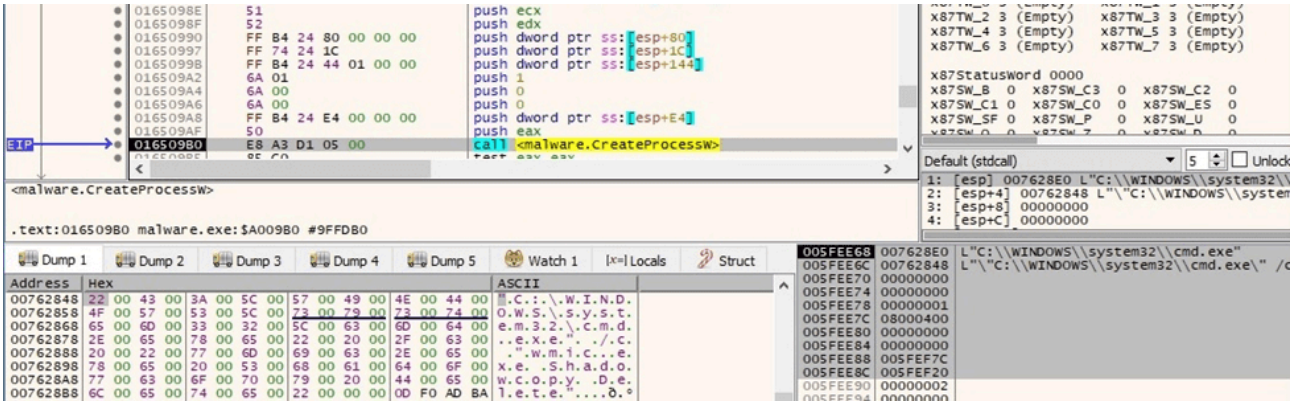


Figure 37 Interestingly, the malware runs the following command that is incomplete and returns an error:

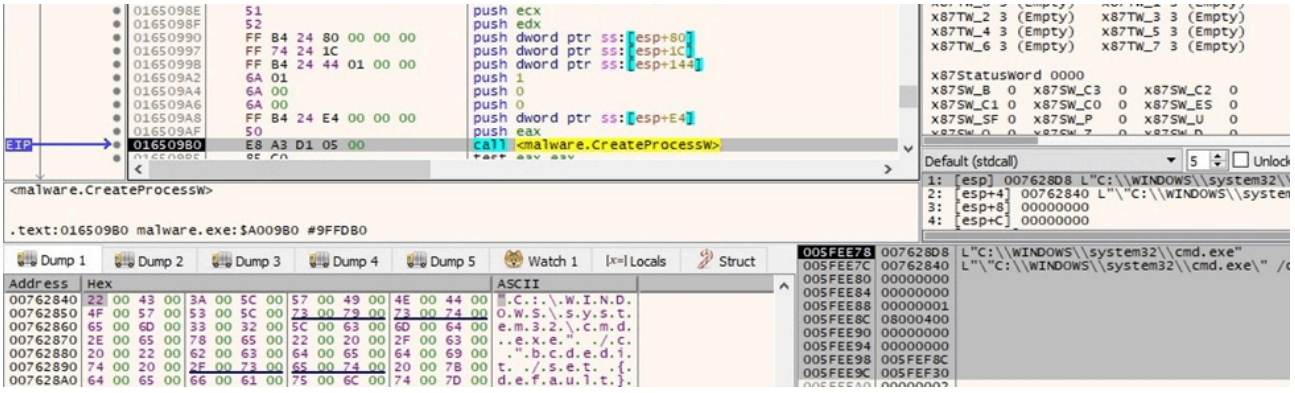


Figure 38

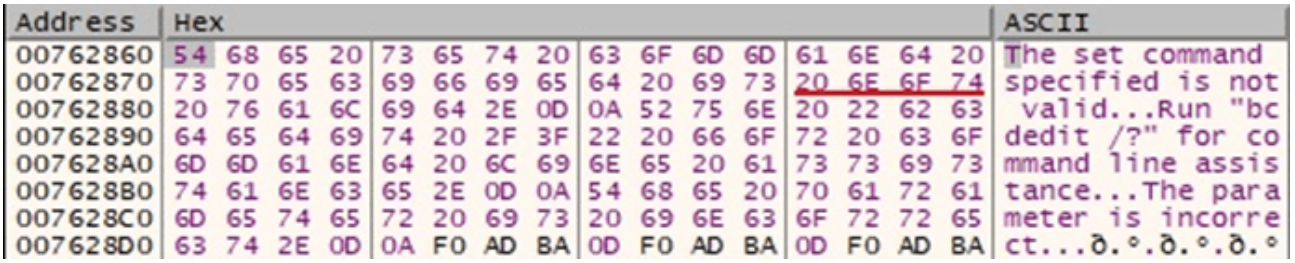


Figure 39 The binary disables Automatic Repair using the bcdedit tool:

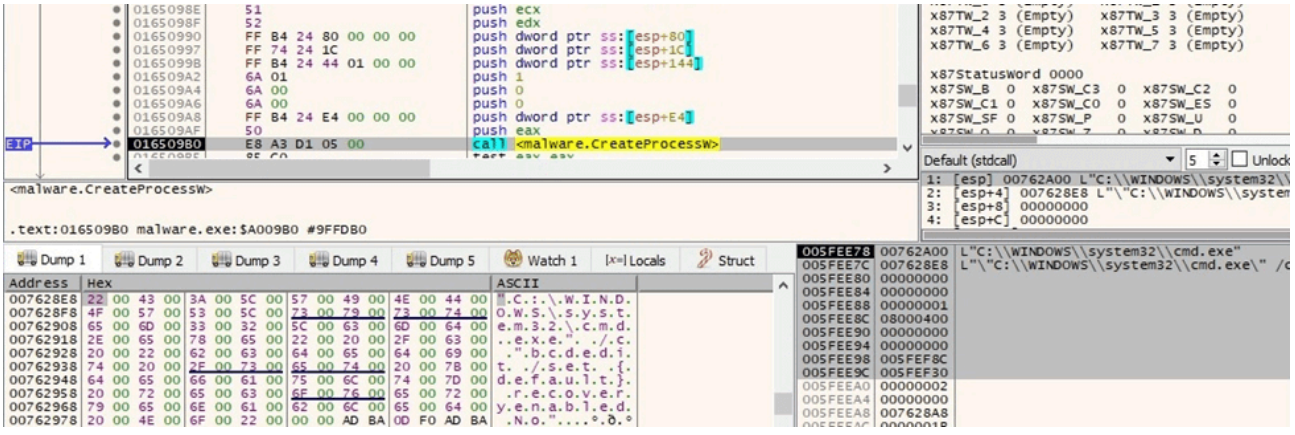


Figure 40 The ransomware tries to clear all event logs, however, the command is incorrect and returns an error, as highlighted below:

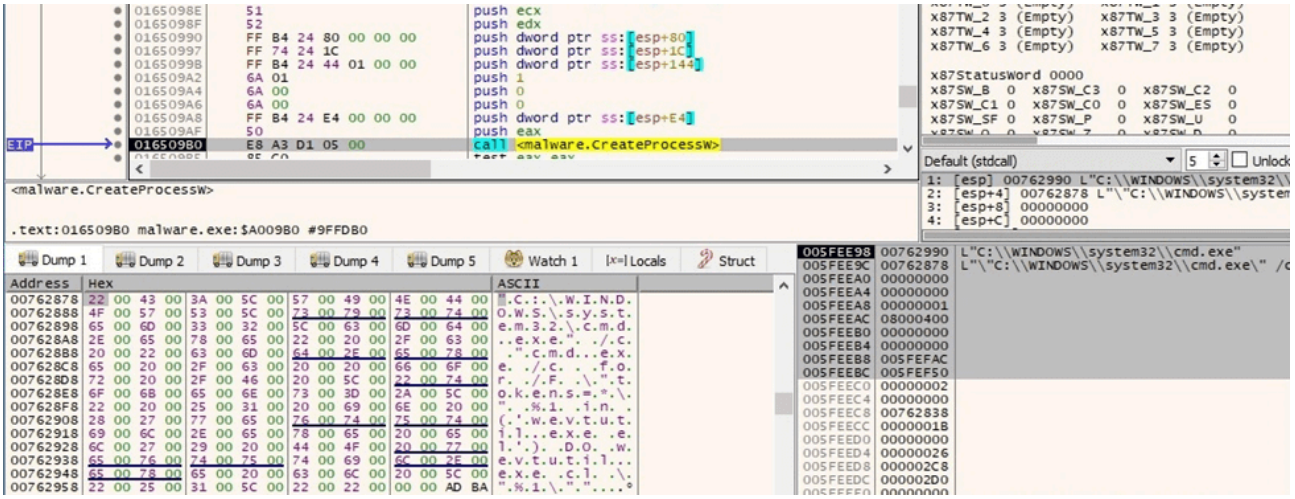
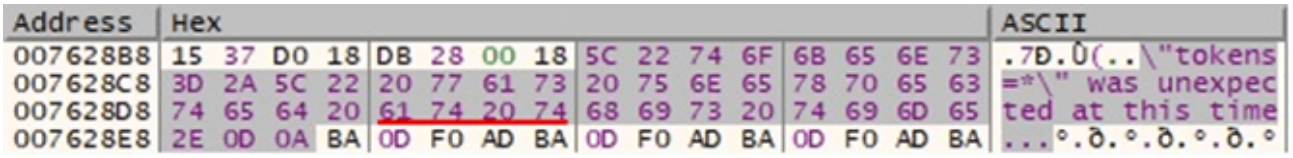


Figure 41



Address	Hex	ASCII
00762888	15 37 D0 18 DB 28 00 18 5C 22 74 6F 6B 65 6E 73	.7Đ.Û(..\"tokens
007628C8	3D 2A 5C 22 20 77 61 73 20 75 6E 65 78 70 65 63	=*\" was unexpec
007628D8	74 65 64 20 61 74 20 74 68 69 73 20 74 69 6D 65	ted at this time
007628E8	2E 0D 0A BA 0D F0 AD BA 0D F0 AD BA 0D F0 AD BA	...°.ð.°.ð.°.ð.°

Figure 42

Killing targeted services

The binary opens the service control manager database via a function call to OpenSCManagerW (0xF003F = **SC_MANAGER_ALL_ACCESS**):

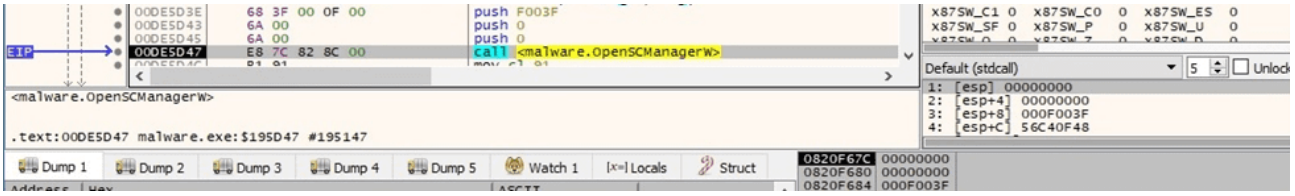


Figure 43 The process obtains a list of active services using EnumServicesStatusExW (0x30 = **SERVICE_WIN32**, 0x1 = **SERVICE_ACTIVE**):

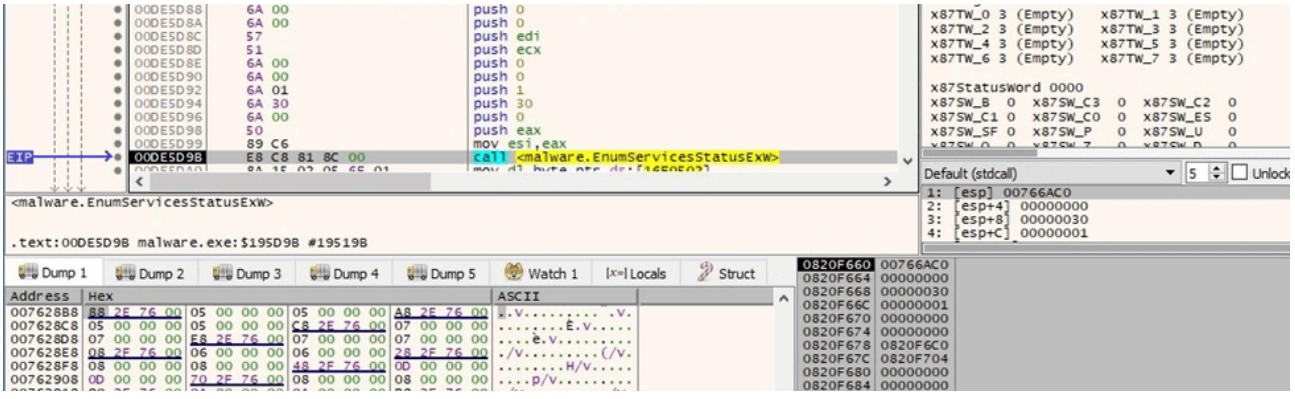


Figure 44 The malware targets the list of services from the kill_services element in the BlackCat configuration. A targeted service is opened by calling the OpenServiceW routine (0x2c = **SERVICE_STOP | SERVICE_ENUMERATE_DEPENDENTS | SERVICE_QUERY_STATUS**):

Figure 45 EnumDependentServicesW is utilized to retrieve the active services that depend on the targeted service (0x1 = **SERVICE_ACTIVE**):

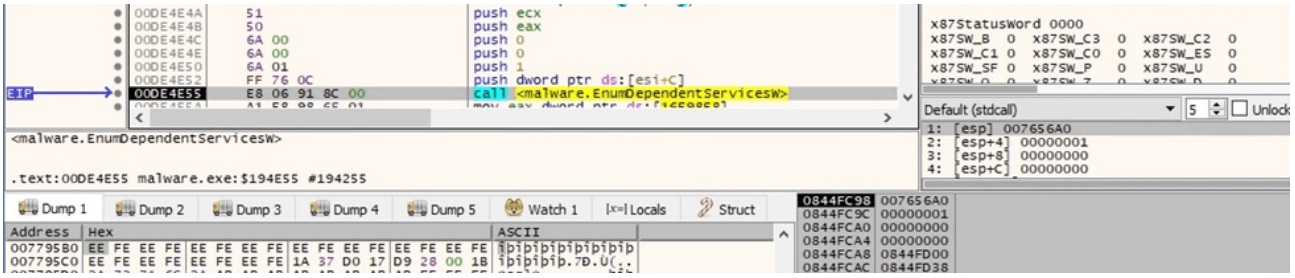


Figure 46 BlackCat stops the targeted service using the ControlService function (0x1 = **SERVICE_CONTROL_STOP**):

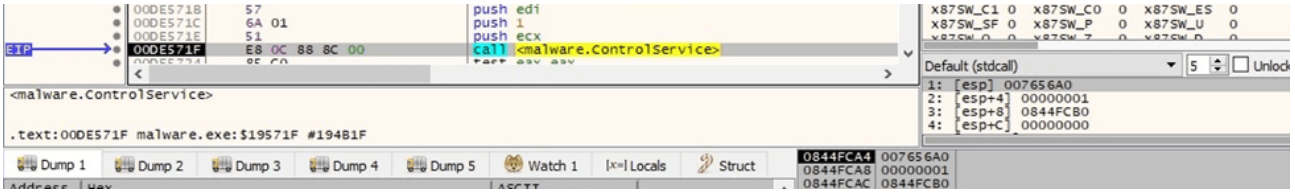


Figure 47

Killing targeted processes

The executable takes a snapshot of all processes and threads in the system (0xF = **TH32CS_SNAPALL**):

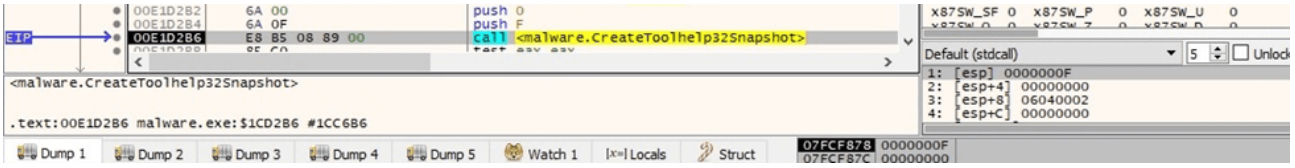


Figure 48 The processes are enumerated using the Process32FirstW and Process32NextW APIs:

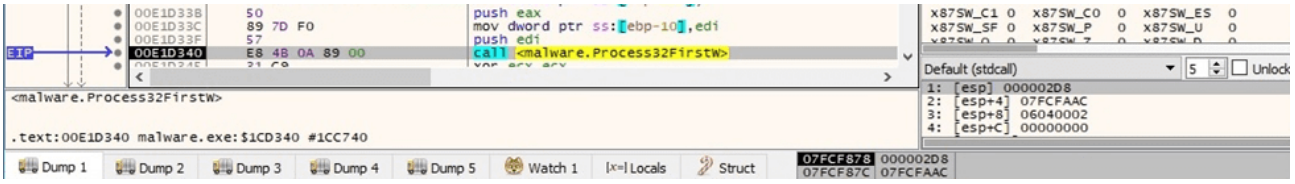


Figure 49

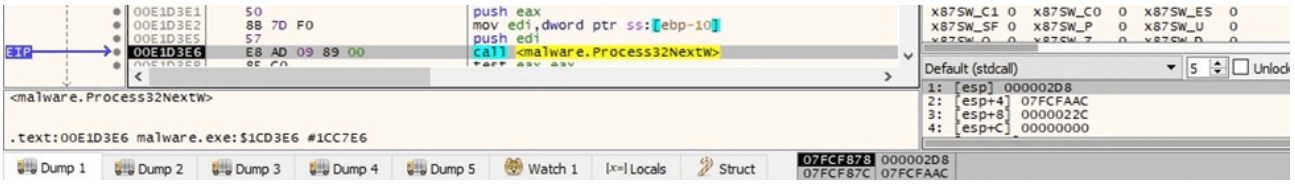


Figure 50 The malware targets the list of processes from the kill_processes element in the BlackCat configuration. It opens a targeted process using OpenProcess (0x1 = **PROCESS_TERMINATE**):

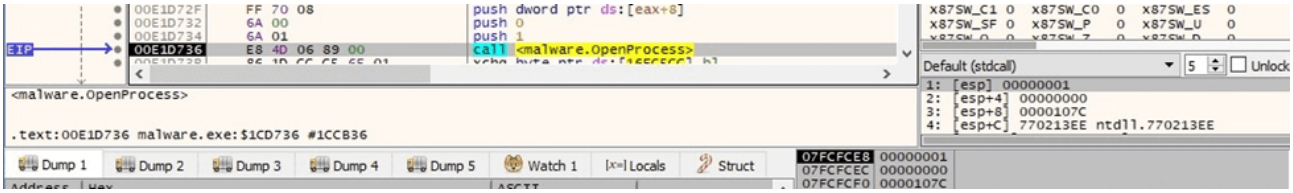


Figure 51 The ransomware terminates the targeted process by calling the TerminateProcess API:

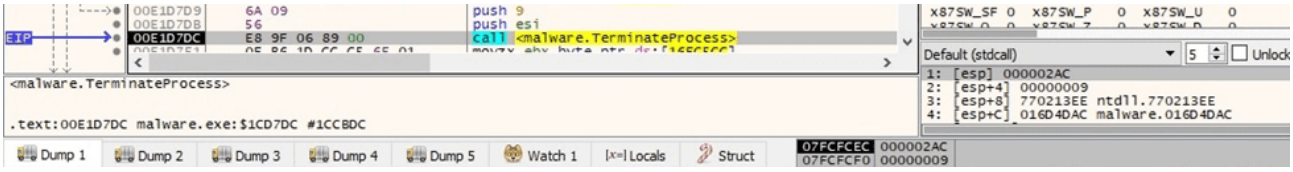


Figure 52 The binary spawns multiple child processes by adding the "–child" parameter to the command line (see figure 53). The new processes run in the security context of credentials that were specified in the credentials entry from the BlackCat configuration.

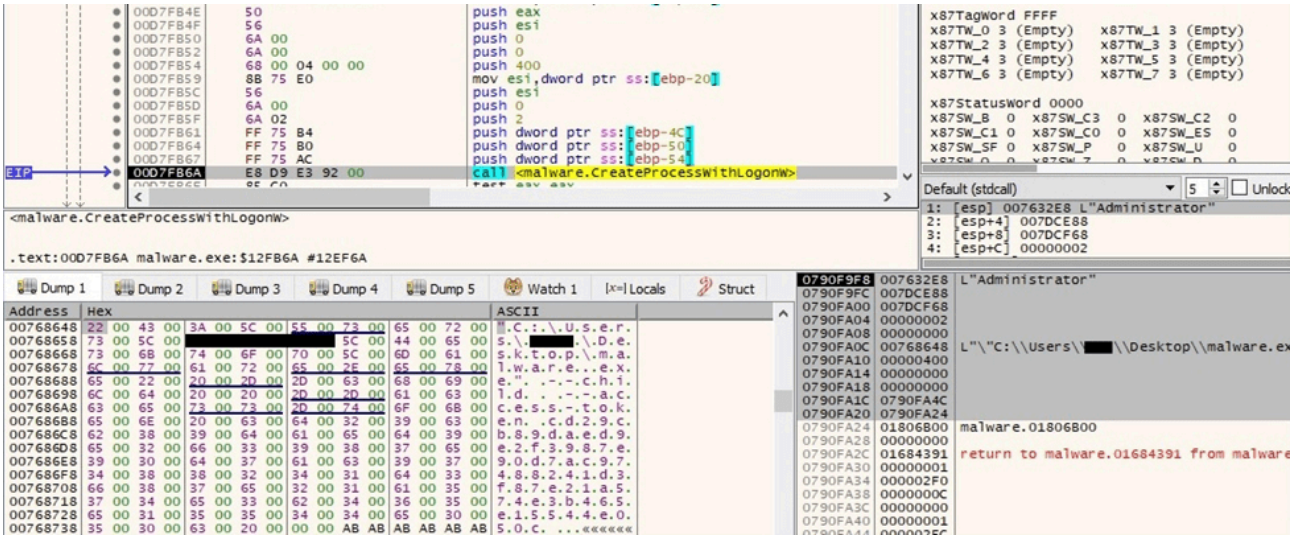


Figure 53 The number of network requests the Server Service can make is set to the maximum by modifying "HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters\MaxMpxCt" Registry value:

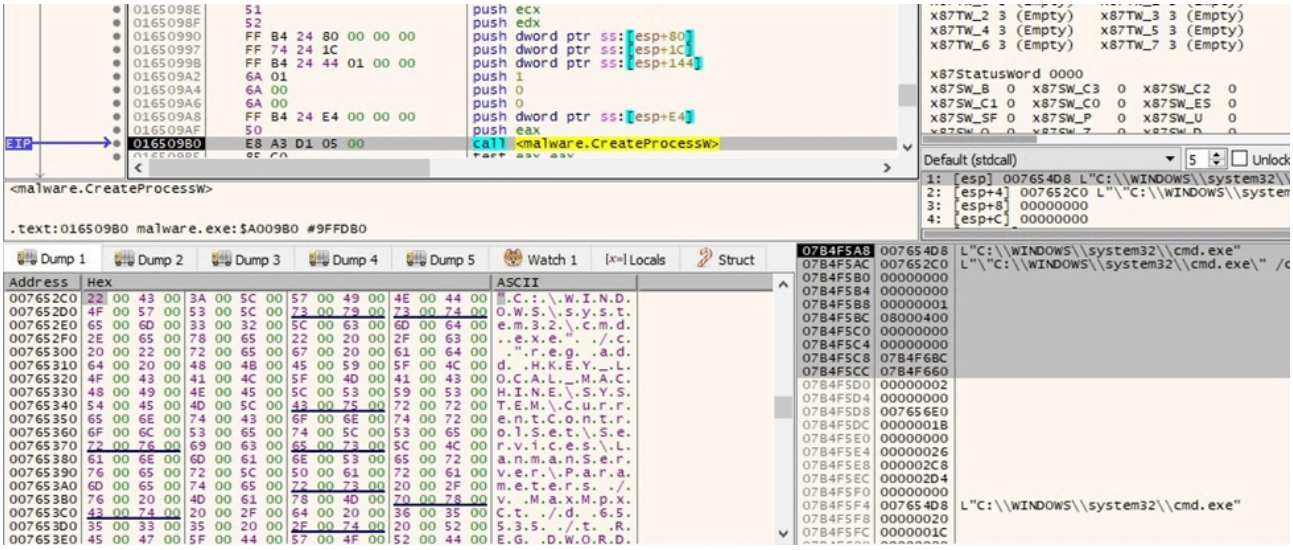


Figure 54 The malicious process obtains the ARP table using the arp command, as shown below:

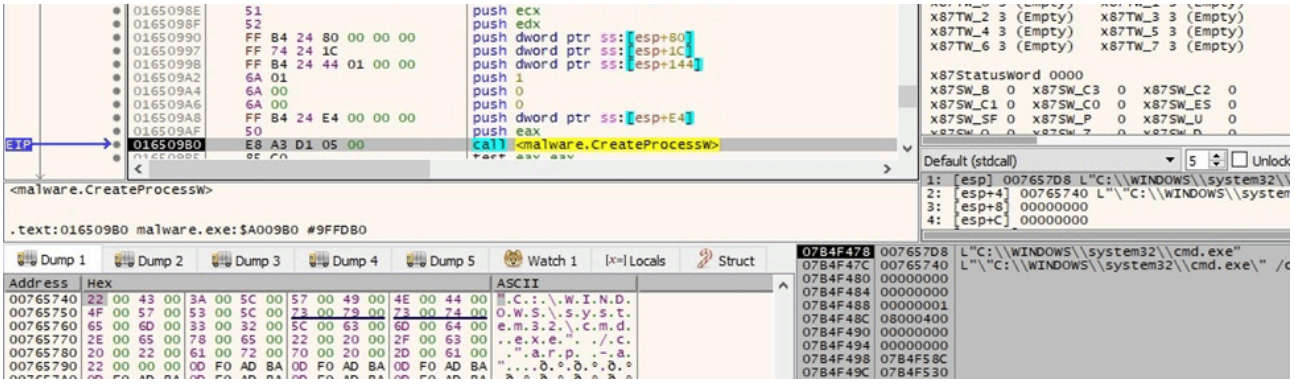


Figure 55 The net use command is utilized to connect to the local computer using different credentials stored in the BlackCat configuration:

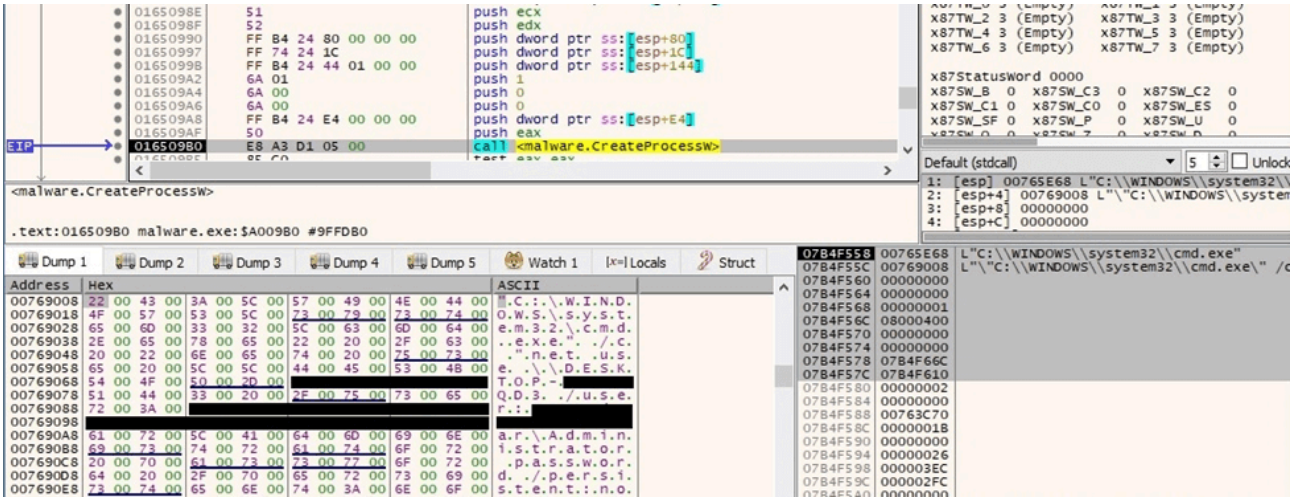


Figure 56 The malware retrieves the currently available disk drives by calling the GetLogicalDrives routine:

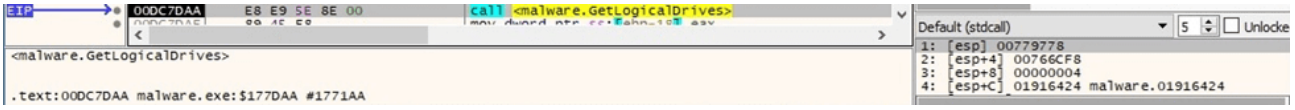


Figure 57 The GetDriveTypeW API is utilized to obtain the drive type:

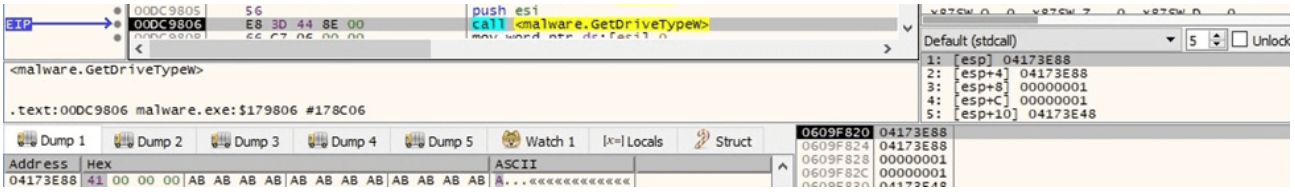


Figure 58 The ransomware starts scanning the volumes on the local machine using FindFirstVolumeW:

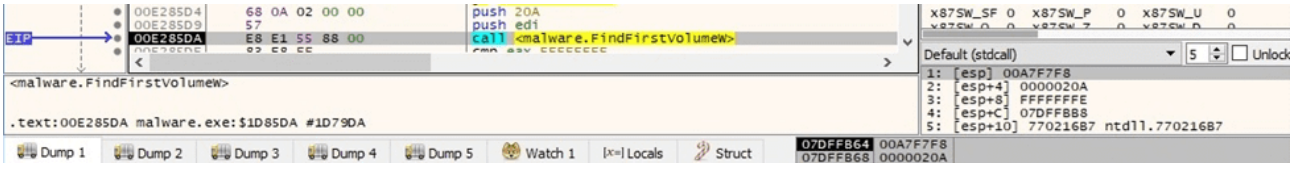


Figure 59 The list of drive letters and mounted folder paths for the above volume is extracted by the malware:

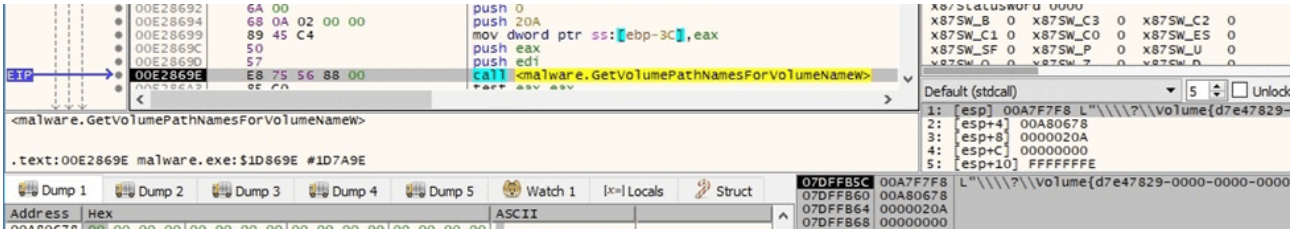


Figure 60 The volume's enumeration continues by calling the FindNextVolumeW function:

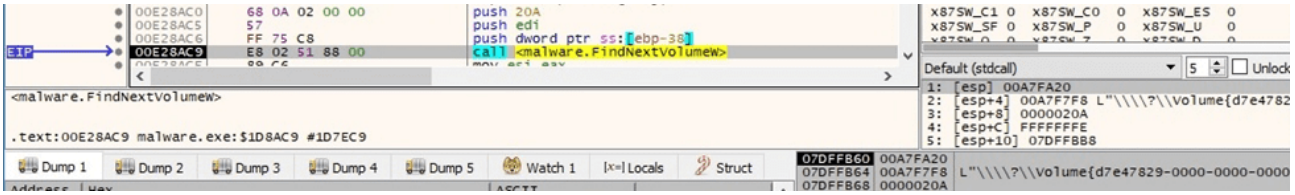


Figure 61 All unmounted volumes are mounted via a function call to SetVolumeMountPointW:

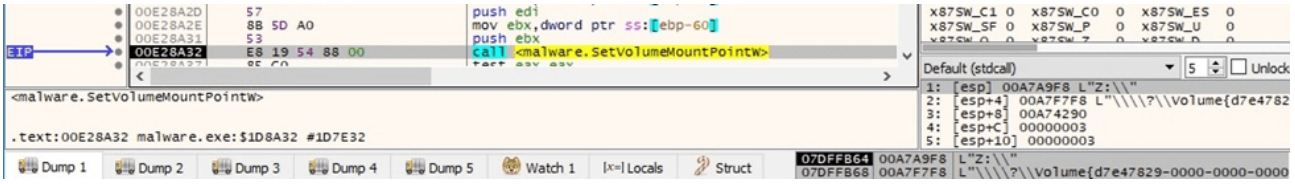


Figure 62 BlackCat traverses the file system using the FindFirstFileW and FindNextFileW APIs:

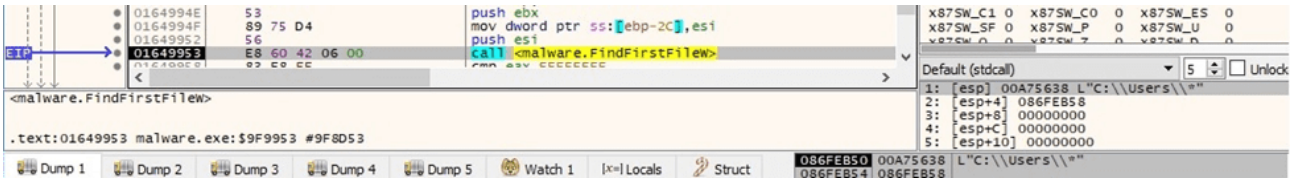


Figure 63

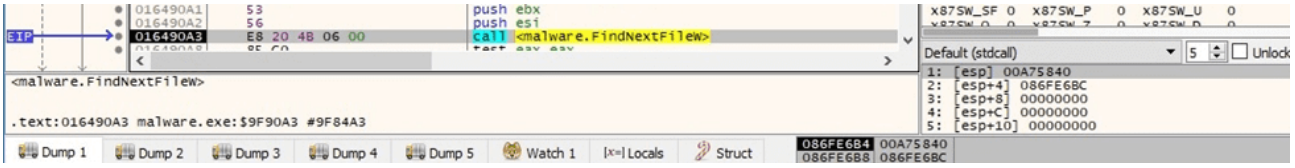


Figure 64 The BlackCat configuration is stored in JSON form and is decrypted at runtime. It contains:

- the extension appended to the encrypted files
- RSA public key that is used to encrypt the AES encryption key
- ransom note name and content
- stolen credentials specific to the victim's environment
- encryption cipher: AES
- list of services and processes to be killed
- list of folders, files, and extensions to be skipped
- boolean values that indicate network discovery, lateral movement, setting the Desktop Wallpaper, killing VMware ESXi virtual machines, removing VMware ESXi virtual machine snapshots, excluding VMware ESXi virtual machines from termination

Figure 65

Files encryption

The CreateFileW API is used to open a targeted file (0xC0000000 = **GENERIC_READ | GENERIC_WRITE**, 0x7 = **FILE_SHARE_DELETE | FILE_SHARE_WRITE | FILE_SHARE_READ**, 0x3 = **OPEN_EXISTING**):

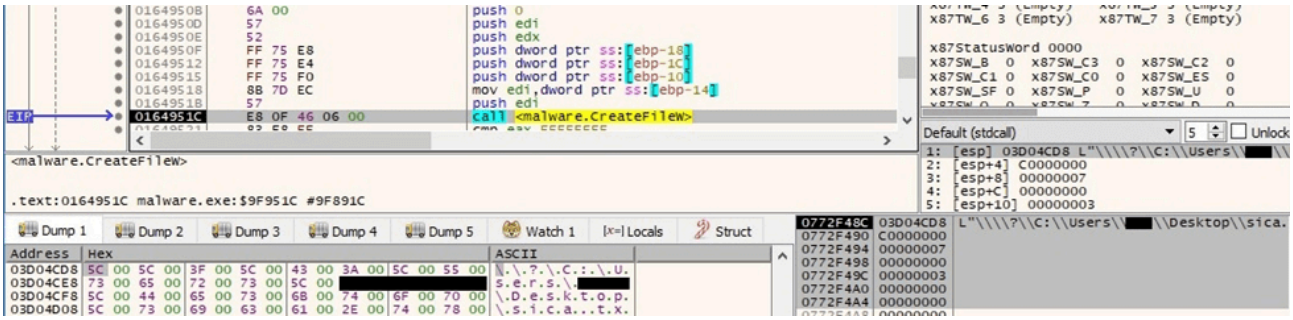


Figure 66 The ransom note is created in every traversed directory (0x40000000 = **GENERIC_WRITE**, 0x7 = **FILE_SHARE_DELETE | FILE_SHARE_WRITE | FILE_SHARE_READ**, 0x2 = **CREATE_ALWAYS**):

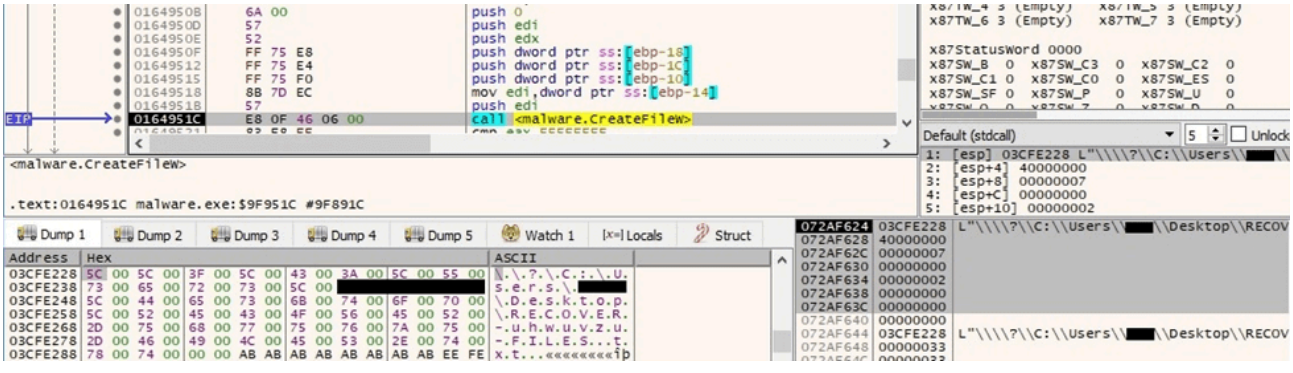


Figure 67 The ransom note is populated using the WriteFile routine:

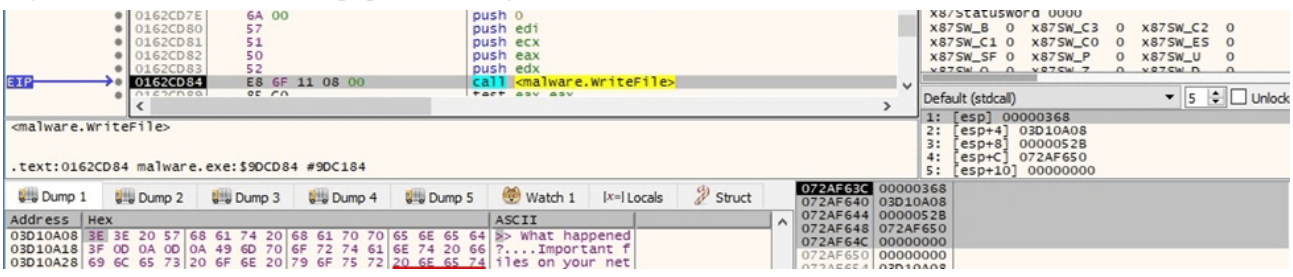


Figure 68

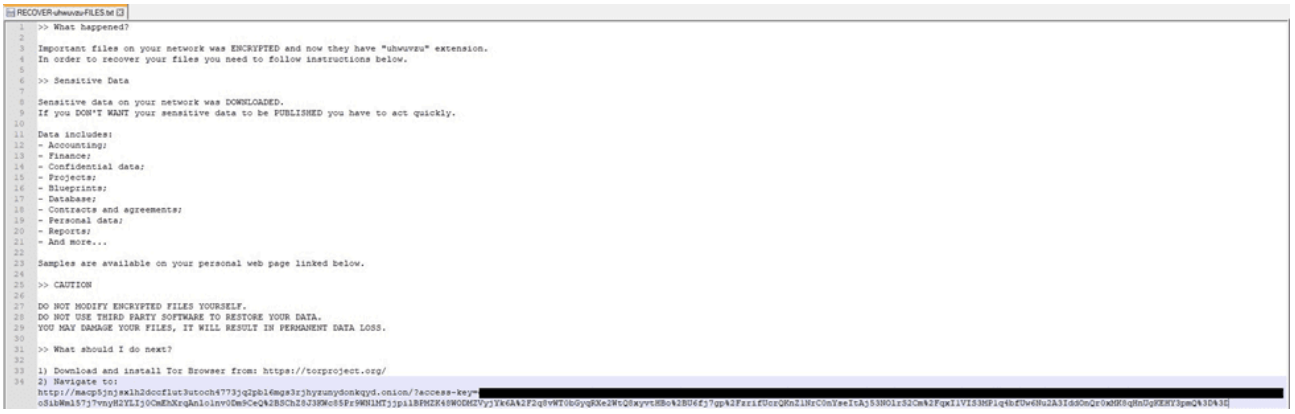


Figure 69 The file's extension is changed using the MoveFileExW function. The renamed file is opened using CreateFileW (0x7 = **FILE_SHARE_DELETE | FILE_SHARE_WRITE | FILE_SHARE_READ**, 0x3 = **OPEN_EXISTING**, 0x02000000 = **FILE_FLAG_BACKUP_SEMANTICS**):

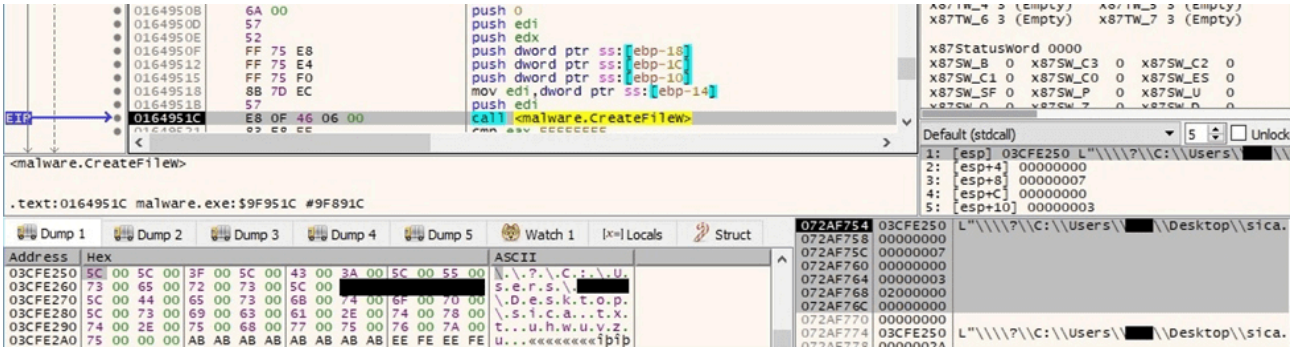


Figure 70 Interestingly, BlackCat creates intermediary files called "checkpoints-<encrypted file name>" during the encryption process:

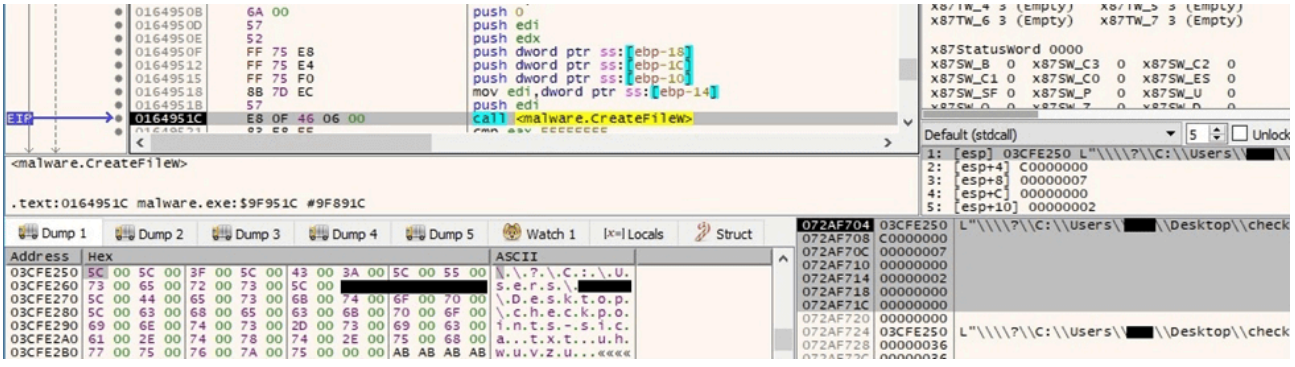


Figure 71 The malware generates 16 random bytes that will be used to derive the AES key:

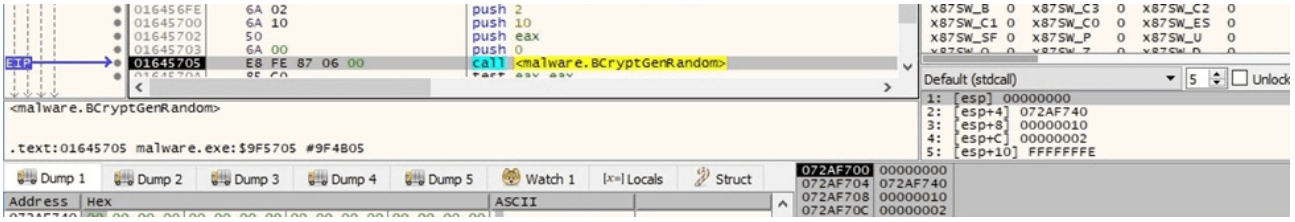


Figure 72 The ransomware moves the file pointer to the beginning of the file by calling the SetFilePointerEx API (0x0 = **FILE_BEGIN**):

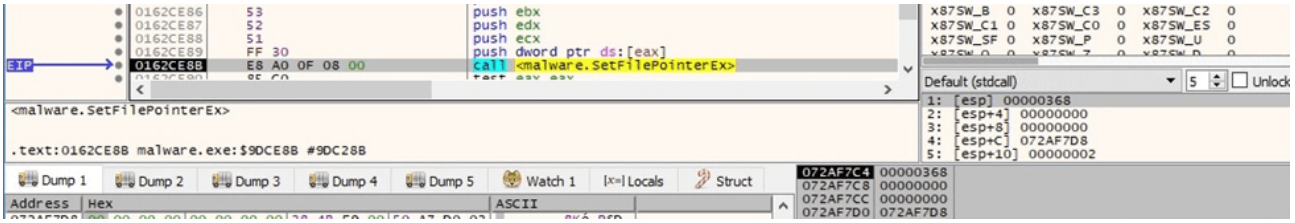


Figure 73 The process reads 4 bytes from the beginning of the file using ReadFile:

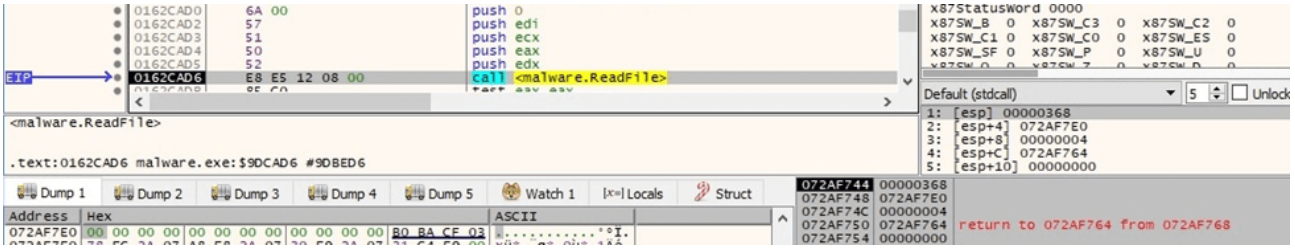


Figure 74 A JSON form containing the encryption cipher (AES), the AES key used to encrypt the file, the data, and the chunk size, is constructed in the process memory:

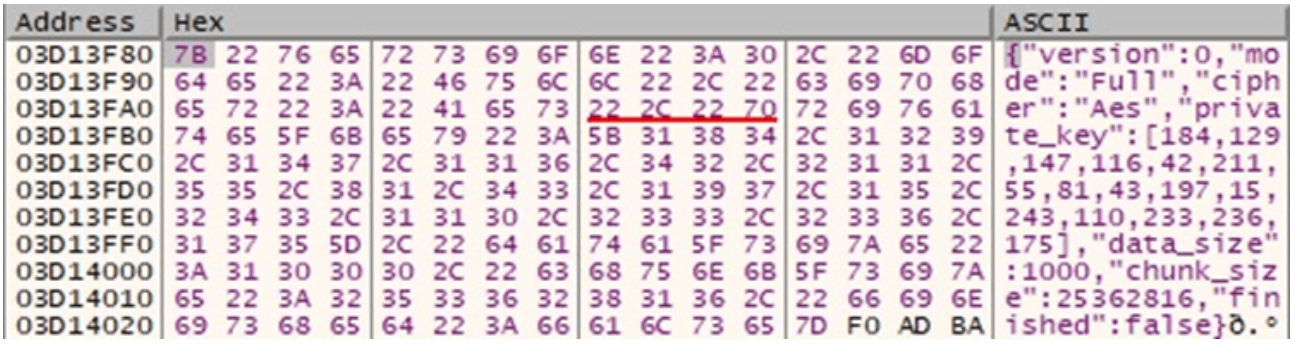


Figure 75 The binary generates 0x50 (80) random bytes that are used to border the JSON form. The resulting buffer has a size of 256 bytes and is rotated using instructions such as pshuflw:

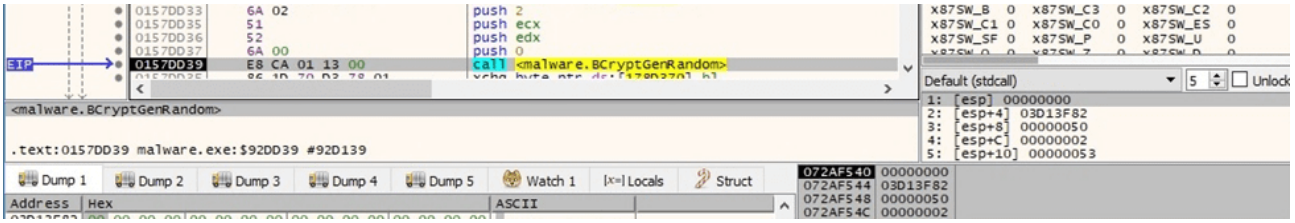


Figure 76

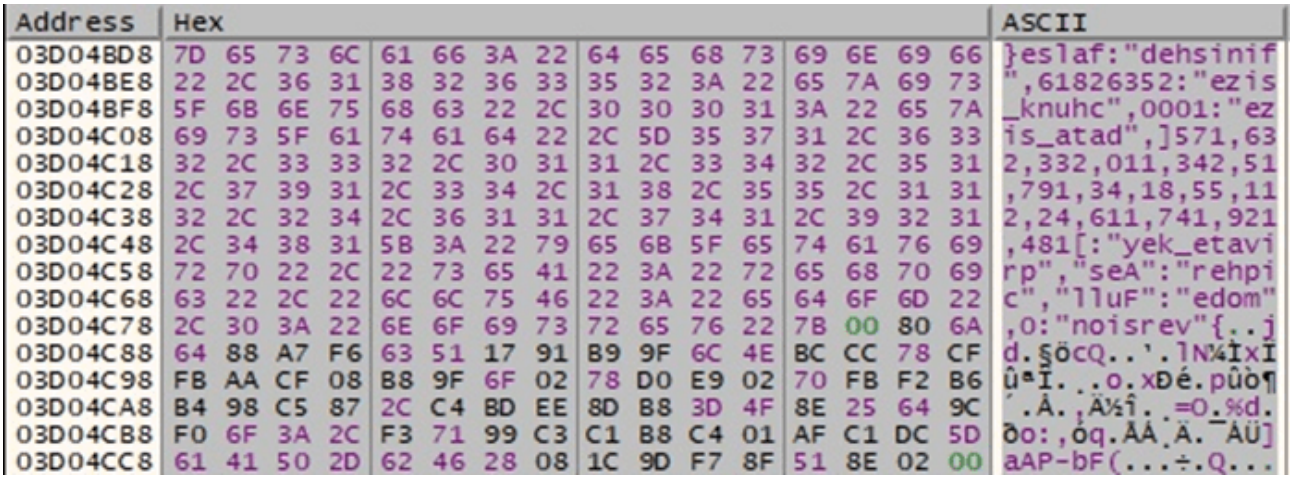


Figure 77 A 4-byte border “19 47 B2 CE” that separates the encrypted file content from the encrypted AES key is written to the file:

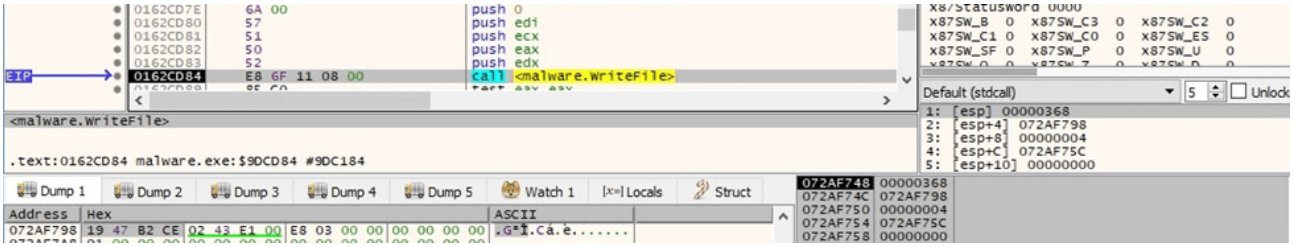


Figure 78 The buffer that contains the AES key presented in figure 77 is encrypted with the RSA public key from the BlackCat configuration. The result is written to the file using WriteFile:

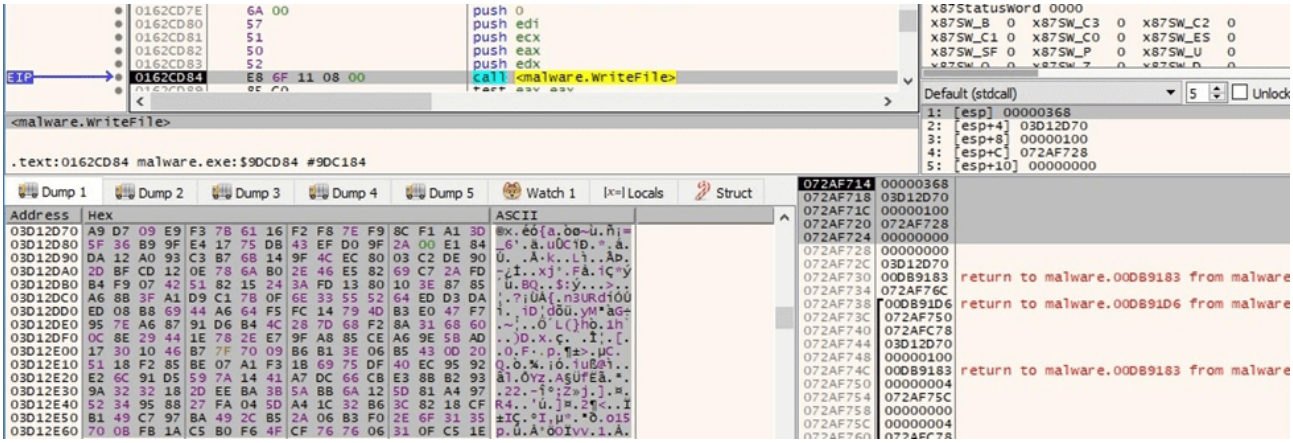


Figure 79 The size of encrypted key (0x100) is written to the file:

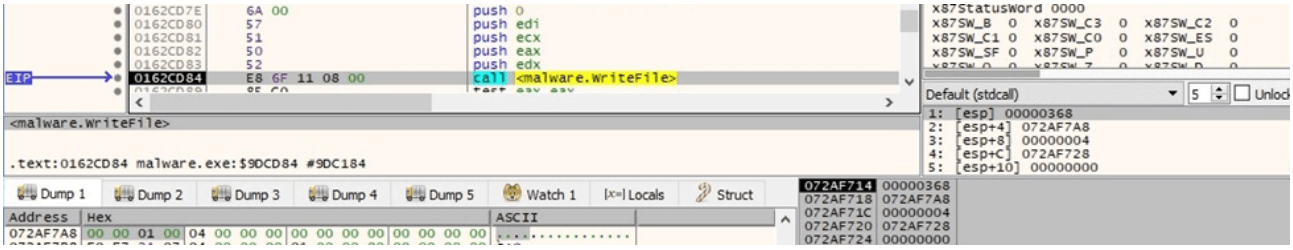


Figure 80 The file content is read by using the ReadFile function:

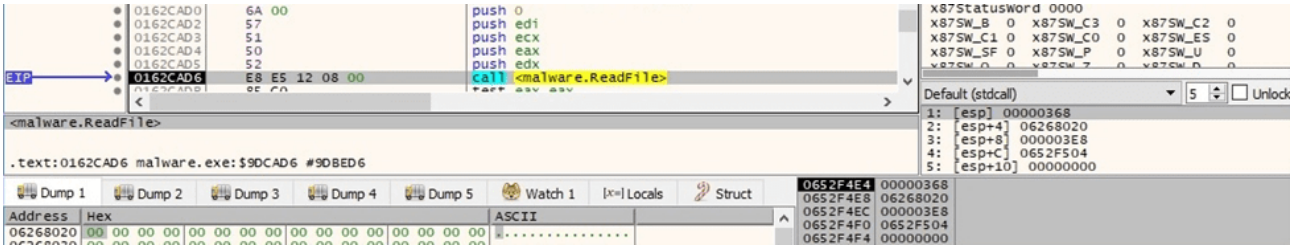


Figure 81 The file content is encrypted using the AES-128 algorithm. The malware uses the aesenc and aesenclast instructions for this purpose:

```
00DD84C7   66 0F 38 DC D0          aesenc xmm2,xmm0
00DD84CC   66 0F 38 DC D8          aesenc xmm3,xmm0
00DD84D1   66 0F 38 DC E0          aesenc xmm4,xmm0
00DD84D6   66 0F 38 DC E8          aesenc xmm5,xmm0
00DD84DB   66 0F 38 DC F0          aesenc xmm6,xmm0
00DD84E0   66 0F 38 DC F8          aesenc xmm7,xmm0
00DD84E5   66 0F 38 DC C8          aesenc xmm1,xmm0
00DD84EA   66 0F 7F 14 24          movdqa xmmword ptr ss:[esp],xmm2
00DD84EF   66 0F 6F 54 24 10       movdqa xmm2,xmmword ptr ss:[esp+10]
00DD84F5   66 0F 38 DC D0          aesenc xmm2,xmm0
00DD84FA   66 0F 6F 41 20          movdqa xmm0,xmmword ptr ds:[ecx+20]
00DD84FF   66 0F 7F 54 24 10       movdqa xmmword ptr ss:[esp+10],xmm2
00DD8505   66 0F 6F 14 24          movdqa xmm2,xmmword ptr ss:[esp]
00DD850A   66 0F 38 DC D8          aesenc xmm3,xmm0
00DD850F   66 0F 38 DC E0          aesenc xmm4,xmm0
00DD8514   66 0F 38 DC E8          aesenc xmm5,xmm0
00DD8519   66 0F 38 DC F0          aesenc xmm6,xmm0
00DD851E   66 0F 38 DC F8          aesenc xmm7,xmm0
00DD8523   66 0F 38 DC C8          aesenc xmm1,xmm0
00DD8528   66 0F 38 DC D0          aesenc xmm2,xmm0
00DD852D   66 0F 7F 14 24          movdqa xmmword ptr ss:[esp],xmm2
00DD8532   66 0F 6F 54 24 10       movdqa xmm2,xmmword ptr ss:[esp+10]
00DD8538   66 0F 38 DC D0          aesenc xmm2,xmm0
00DD853D   66 0F 6F 41 30          movdqa xmm0,xmmword ptr ds:[ecx+30]
00DD8542   66 0F 7F 54 24 10       movdqa xmmword ptr ss:[esp+10],xmm2
```

Figure 82

```
EIP ->
00DD87A0   55                      push ebp
00DD87A1   89 E5                   mov ebp,esp
00DD87A3   53                      push ebx
00DD87A4   8B 45 08                mov eax,dword ptr ss:[ebp+8]
00DD87A7   66 0F 6F 02             movdqa xmm0,xmmword ptr ds:[edx]
00DD87AB   B3 F1                   mov bl,F1
00DD87AD   66 0F EF 00             pxor xmm0,xmmword ptr ds:[eax]
00DD87B1   66 0F 38 DC 42 10       aesenc xmm0,xmmword ptr ds:[edx+10]
00DD87B7   66 0F 38 DC 42 20       aesenc xmm0,xmmword ptr ds:[edx+20]
00DD87BD   66 0F 38 DC 42 30       aesenc xmm0,xmmword ptr ds:[edx+30]
00DD87C3   66 0F 38 DC 42 40       aesenc xmm0,xmmword ptr ds:[edx+40]
00DD87C9   66 0F 38 DC 42 50       aesenc xmm0,xmmword ptr ds:[edx+50]
00DD87CF   66 0F 38 DC 42 60       aesenc xmm0,xmmword ptr ds:[edx+60]
00DD87D5   66 0F 38 DC 42 70       aesenc xmm0,xmmword ptr ds:[edx+70]
00DD87DB   66 0F 38 DC 82 80 00 00 00  aesenc xmm0,xmmword ptr ds:[edx+80]
00DD87E4   86 1D FE 8B 6E 01       xchg byte ptr ds:[16E8BFE],bl
00DD87EA   66 0F 38 DC 82 90 00 00 00  aesenc xmm0,xmmword ptr ds:[edx+90]
00DD87F3   66 0F 7F 00             movdqa xmmword ptr ds:[eax],xmm0
00DD87F7   66 0F 38 DD 82 A0 00 00 00  aesenclast xmm0,xmmword ptr ds:[edx+A0]
00DD8800   66 0F 7F 01             movdqa xmmword ptr ds:[ecx],xmm0
```

Figure 83 The encrypted file content is written back to the file using WriteFile:

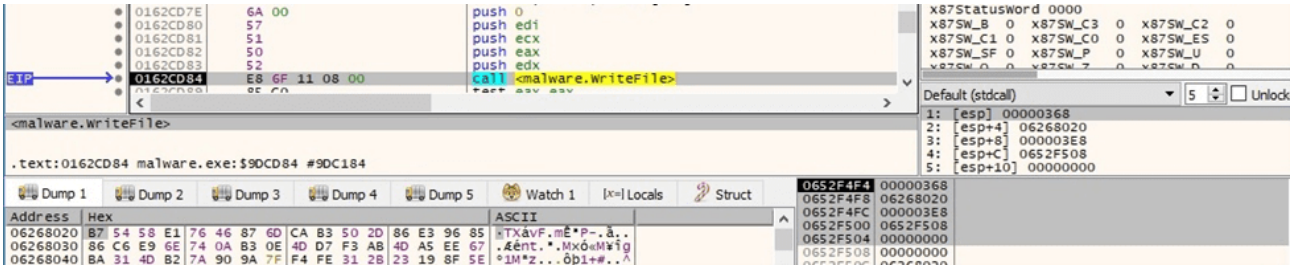


Figure 84 An example of an encrypted file is displayed below:

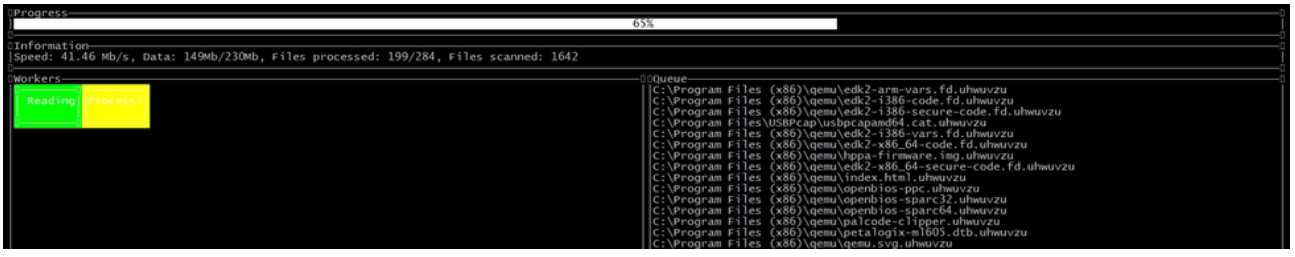


Figure 85 The ransomware creates a PNG image called “RECOVER-uhwuvzu-FILES.txt.png”:

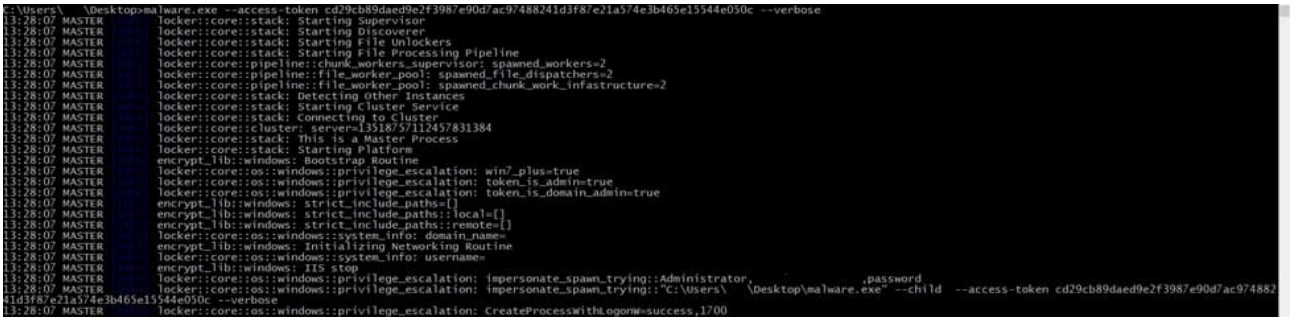


Figure 86

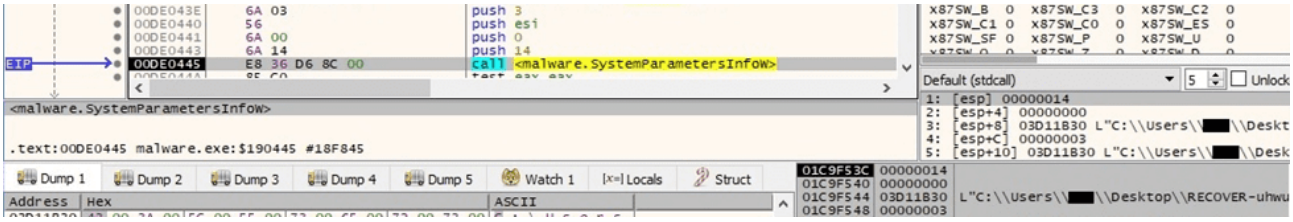


Figure 87 The Desktop wallpaper is changed to the above image by calling the SystemParametersInfoW API (0x14 = **SPI_SETDESKWALLPAPER**, 0x3 = **SPIF_UPDATEINIFILE | SPIF_SENDCHANGE**):

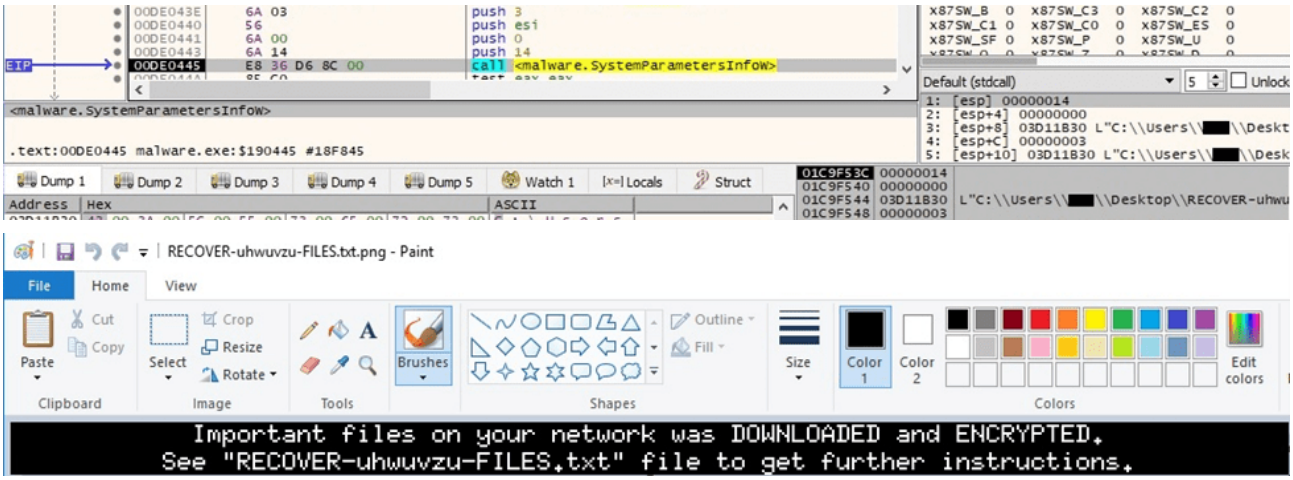


Figure 88

Running with the –verbose parameter

The ransomware writes multiple actions to the command line output:

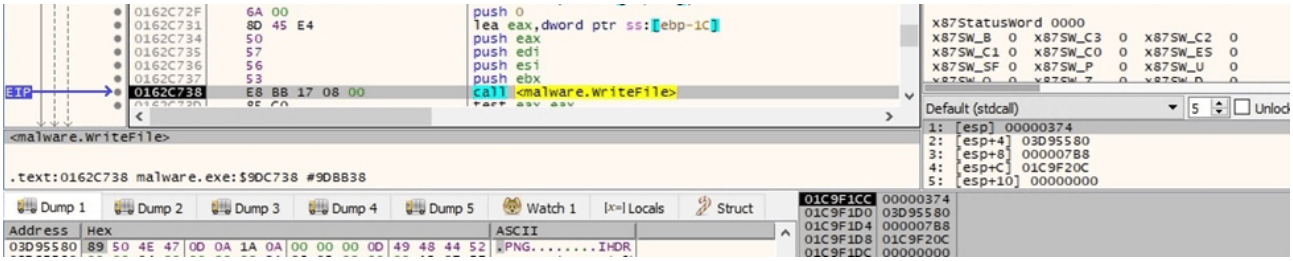


Figure 89

Running with the –extra-verbose –ui parameters

The malware presents the relevant information in the following window:

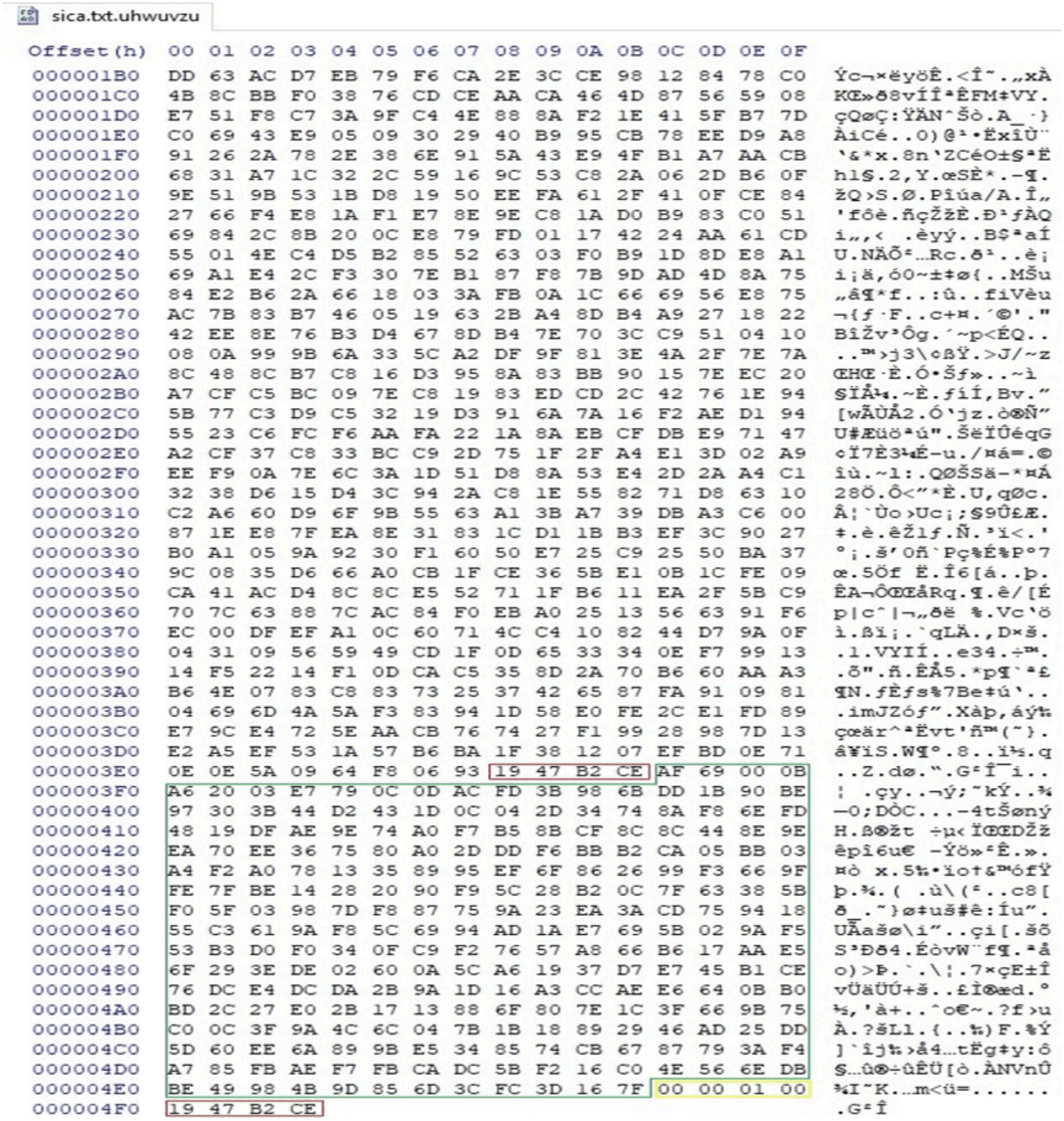


Figure 90

Indicators of Compromise

Pipe \\.\pipe__rust_anonymous_pipe1__.<Process ID>.<Random number> **BlackCat Ransom Note** RECOVER-uhwuvzu-FILES.txt **Files created** checkpoints-<Filename>.uhwuvzu RECOVER-uhwuvzu-FILES.txt.png **Processes spawned** cmd.exe /c “wmic csproduct get UUID” cmd.exe /c “fsutil behavior set SymlinkEvaluation R2L:1” cmd.exe /c “fsutil behavior set SymlinkEvaluation R2R:1” cmd.exe /c “iisreset.exe /stop” cmd.exe /c “vssadmin.exe Delete Shadows /all /quiet” cmd.exe /c “wmic.exe Shadowcopy Delete” cmd.exe /c “bcdedit /set {default}” cmd.exe /c “bcdedit /set {default} recoveryenabled No” cmd.exe /c for /F “tokens=*” %1 in

(‘wevtutil.exe el’) DO wevtutil.exe cl %1 cmd.exe /c “reg add HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters /v MaxMpxCt /d 65535 /t REG_DWORD /f” cmd.exe /c “arp -a”
