BlackMatter Ransomware Analysis; The Dark Side Returns

By Alexandre Mundo and Marc Elias on Sep 22, 2021
BlackMatter is a new ransomware threat discovered at the end of July 2021.

This malware started with a strong group of attacks and some advertising from its developers that claims they take the best parts of other malware, such as GandCrab, LockBit and DarkSide, despite also saying they are a new group of developers. We at McAfee Enterprise Advanced Threat Research (ATR), have serious doubts about this last statement as analysis shows the malware has a great deal in common with DarkSide, the malware associated with the Colonial Pipeline attack which caught the attention of the US government and law enforcement agencies around the world.

The main goal of BlackMatter is to encrypt files in the infected computer and demand a ransom for decrypting them. As with previous ransomware, the operators steal files and private information from compromised servers and request an additional ransom to not publish on the internet.

COVERAGE AND PROTECTION ADVICE

McAfee's EPP solution covers BlackMatter ransomware with an array of prevention and detection techniques.

ENS ATP provides behavioral content focusing on proactively detecting the threat while also delivering known IoCs for both online and offline detections. For DAT based detections, the family will be reported as *Ransom-BlackMatter!<hash>*. ENS ATP adds 2 additional layers of protection thanks to JTI rules that provide attack surface reduction for generic ransomware behaviors and RealProtect (static and dynamic) with ML models targeting ransomware threats.

Updates on indicators are pushed through GTI, and customers of Insights will find a threat-profile on this ransomware family that is updated when new and relevant information becomes available.

TECHNICAL DETAILS

BlackMatter is typically seen as an EXE program and, in special cases, as a DLL (Dynamic Library) for Windows. Linux machines can be affected with special versions of it too but in this report, we will only be covering the Windows version.

This report will focus on version 1.2 of BlackMatter while also noting the important changes in the current version, 2.0.

BlackMatter is programmed in C++ and has a size of 67Kb.

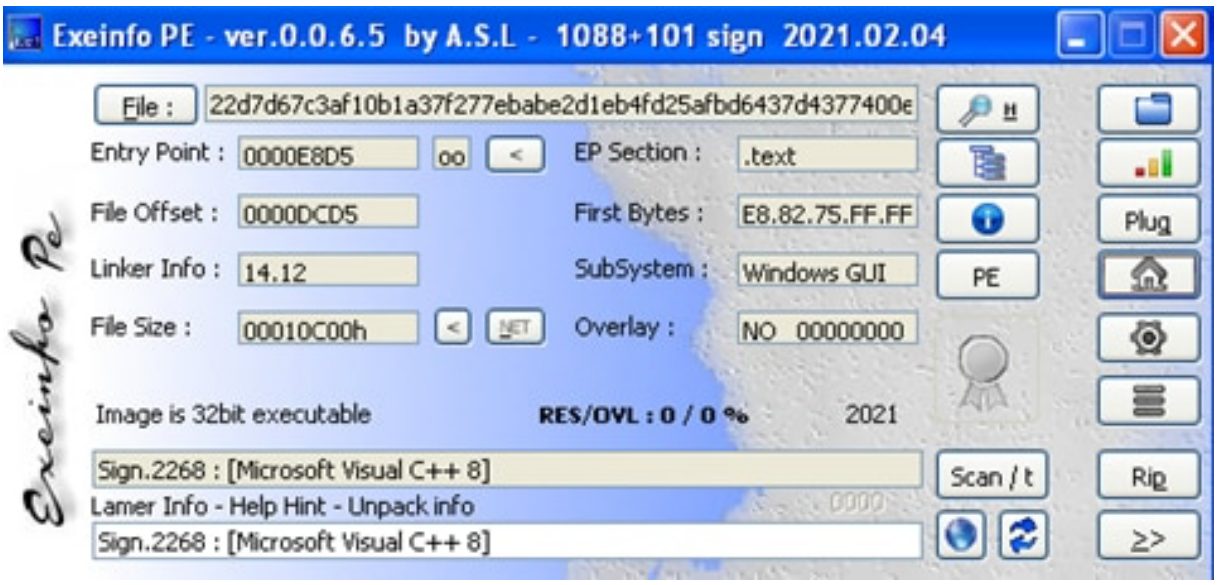


FIGURE 1. Information about the malware

The compile date of this sample is the 23rd of July 2021. While these dates can be altered, we think it is correct; version 1.9 has a compile time of 12 August 2021 and the latest version, 2.0, has a date four days later, on the 16th of August 2021. Is clear that the malware developers are actively improving the code and making detection and analysis harder.

The first action performed by BlackMatter is preparation of some modules that will be needed later to get the required functions of Windows.

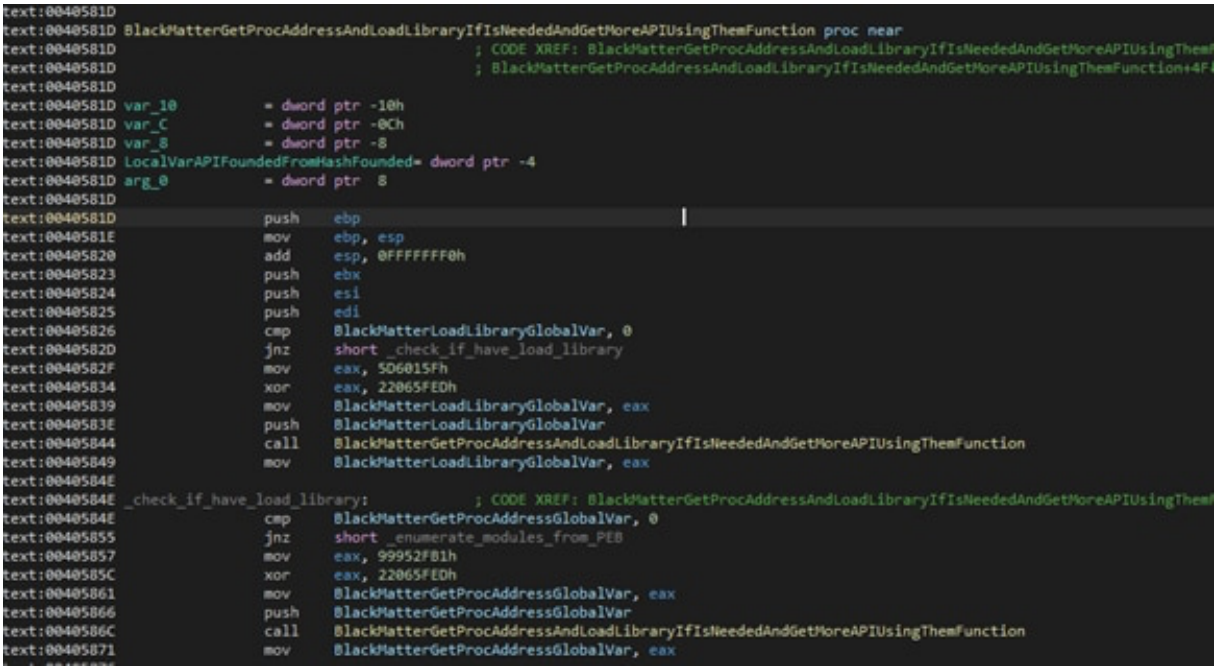


FIGURE 2. BlackMatter searching for functions

BlackMatter uses some tricks to try and make analysis harder and avoid debuggers. Instead of searching for module names it will check for hashes precalculated with a ROT13 algorithm. The modules needed are “kernel32.dll” and “ntdll.dll”. Both modules will try to get functions to reserve memory in the process heap. The APIs are searched using a combination of the PEB (Process Environment Block) of the module and the EAT (Export Table Address) and enumerating all function names. With these names it will calculate the custom hash and check against the target hashes.

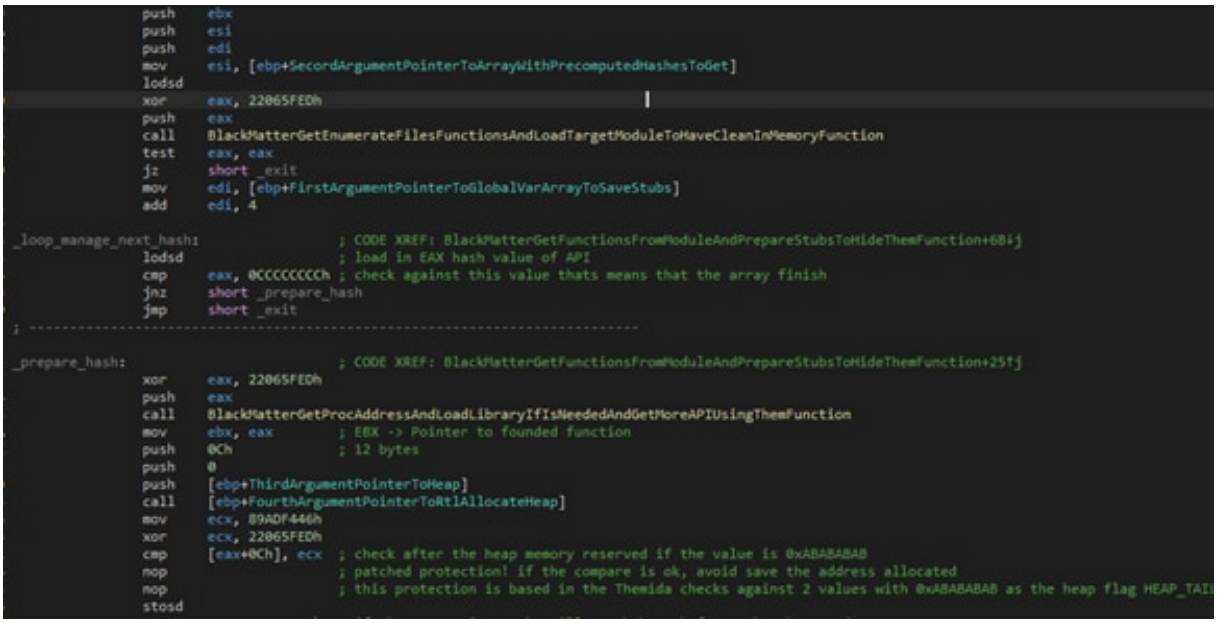


FIGURE 3. BlackMatter detecting a debugger

At this point BlackMatter will make a special code to detect debuggers, checking the last 2 “DWORDS” after the memory is reserved, searching for the bytes “0xABABABAB”. These bytes always exist when a process reserves memory in the heap and, if the heap has one special flag (that by default is set when a process is in a debugger), the malware will avoid saving the pointer to the memory reserved so, in this case, the variables will keep a null pointer.

In Windows operating systems the memory has different conditions based on whether a program is running in normal mode (as usual) or in debugging mode (a mode used by programmers, for example). In this case, when the memory is reserved to keep information, if it is in debugging mode, Windows will mark the end of this memory with a special value, “0xABABABAB”. BlackMatter checks for this value and, if found, the debugger is detected. To avoid having it run normally it will destroy the function address that it gets before, meaning it will crash, thus avoiding the execution.

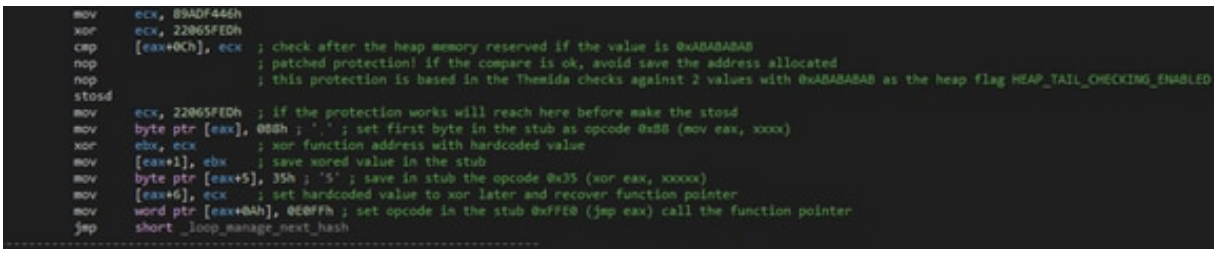


FIGURE 4. Preparing the protection stub function

After this check it will create a special stub in the reserved memory which is very simple but effective in making analysis harder as the stub will need to be executed to see which function is called and executed.

This procedure will be done with all functions that will be needed; the hashes are saved hardcoded in the middle of the “.text” section in little structs as data. The end of each struct will be recognized by a check against the “0xCCCCCCCC” value.

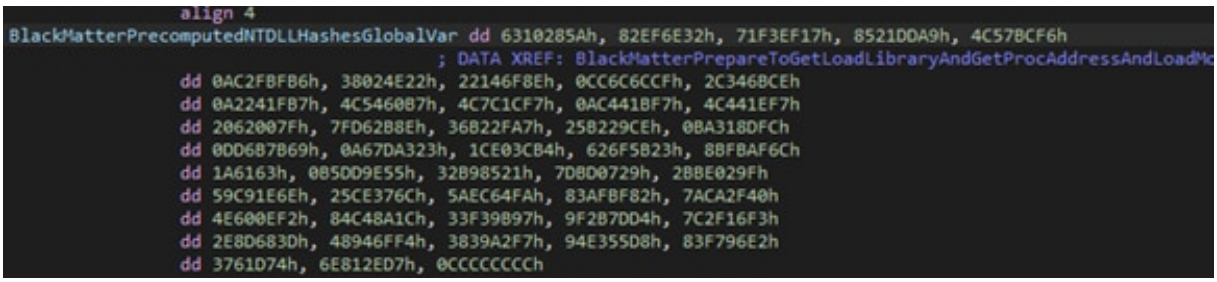


FIGURE 5. Hashes of the functions needed

This behavior highlights that the BlackMatter developers know some tricks to make analysis harder, though it is simple to defeat both by patching the binary.

After this, the ransomware will use another trick to avoid the use of debuggers. BlackMatter will call the function “ZwSetInformationThread” with the class argument of 0x11 which will hide the calling thread from the debuggers.

If the malware executes it correctly and a debugger is attached, the debugging session will finish immediately. This code is executed later in the threads that will be used to encrypt files.

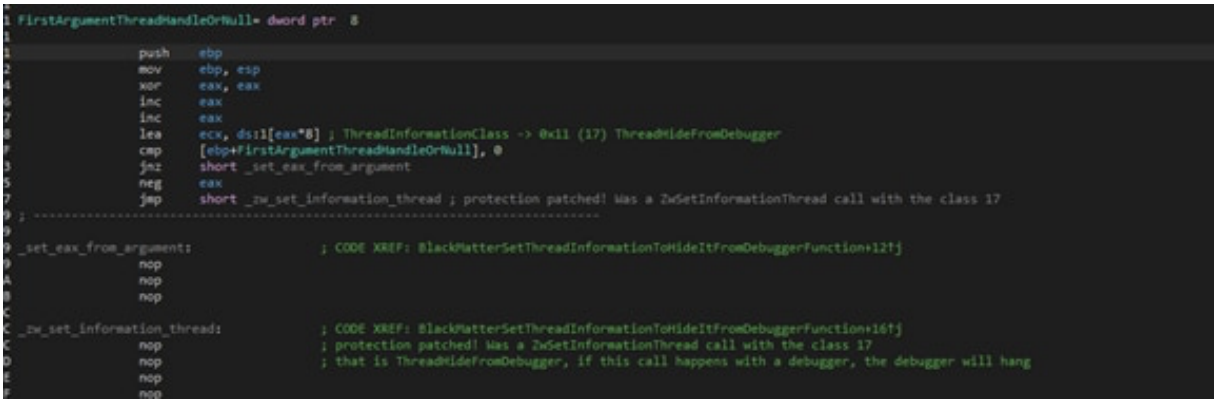


FIGURE 6. Another way to detect a debugger

The next action is to check if the user that launched the process belongs to the local group of Administrators in the machine using the function “SHTestTokenMembership”. In the case that the user belongs to the administrator group the code will continue normally but in other cases it will get the operating system version using the PEB (to avoid using API functions that can alter the version) and, if it is available, will open the process and check the token to see if that belongs to the Administrators group.

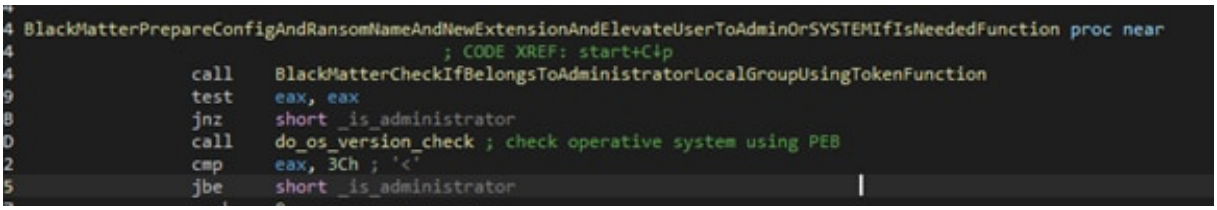


FIGURE 7. BlackMatter checking if it has administrator rights

In the case that the user does not belong to the Administrator group the process token will use a clever trick to escalate privileges.

The first action is to prepare the string “dllhost.exe” and enumerate all modules loaded. For each module it will check one field in the initial structure that all executables have that keeps the base memory address where it will be loaded (for example, kernel32.dll in 0x7fff0000) and will compare with its own base address. If it is equal, it will change its name in the PEB fields and the path and arguments path to “dllhost.exe” (in the case of the path and argument path to the SYSTEM32 folder, where the legitimate “dllhost.exe” exists). This trick is used to

try and mislead the user. For each module found it will check the base address of the module with its own base address and, at that moment, will change the name of the module loaded, the path, and arguments to mislead the user.

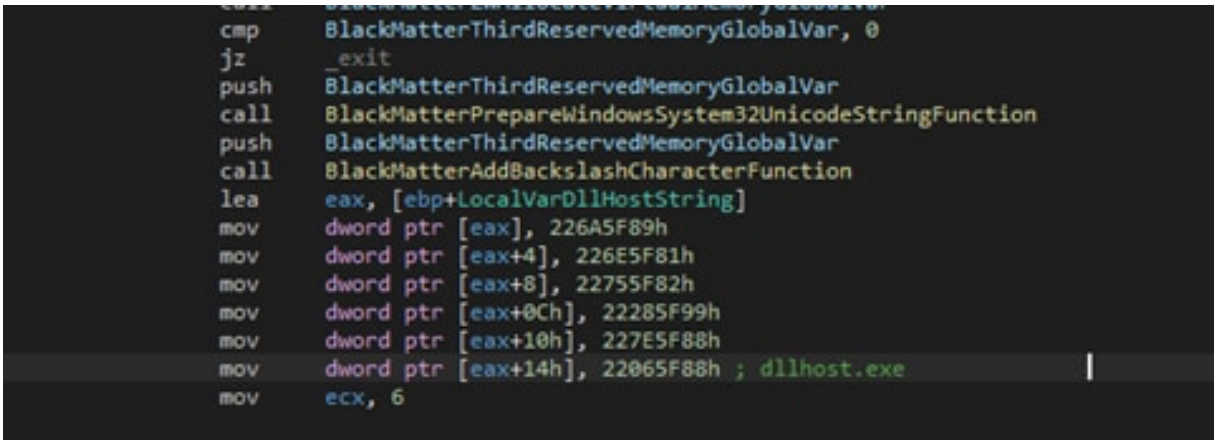


FIGURE 8. Decryption of the string "dllhost.exe"

The process name will be "dllhost.exe" and the path will be the system directory of the victim machine. This trick, besides not changing the name of the process in the TaskManager, can make a debugger "think" that another binary is loaded and remove all breakpoints (depending on the debugger used).

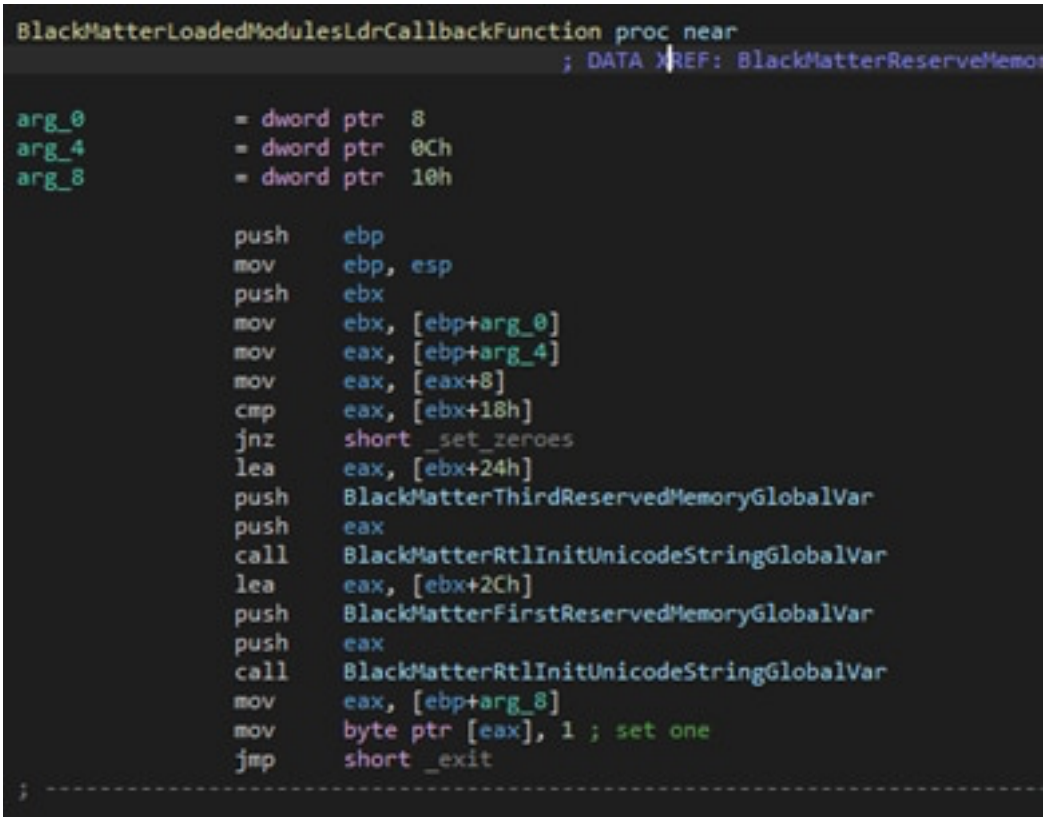


FIGURE 9. Changing the name and path in the PEB

The second action is to use one exploit using COM (Component Object Model) objects to try to elevate privileges before finishing its own instance using the “Terminate Process” function.

For detection, the module uses an undocumented function from NTDLL.DLL, “LoadedModulesLdrCallback” that lets the programmer set a function as a callback where it can get the arguments and check the PEB. In this callback the malware will set the new Unicode strings using “RtlInitUnicodeString”; the strings are the path to “dllhost.exe” in the system folder and “dllhost.exe” as the image name.

The exploit used to bypass the UAC (User Access Control), which is public, uses the COM interface of CMSTPLUA and the COM Elevation Moniker.

In the case that it has administrator rights or uses the exploit with success, it will continue making the new extension that will be used with the encrypted files. For this task it will read the registry key of “Machine Guid” in the cryptographic key (HKEY LOCAL MACHINE).

This entry and value exist in all versions of Windows and is unique for the machine; with this value it will make a custom hash and get the final string of nine characters.

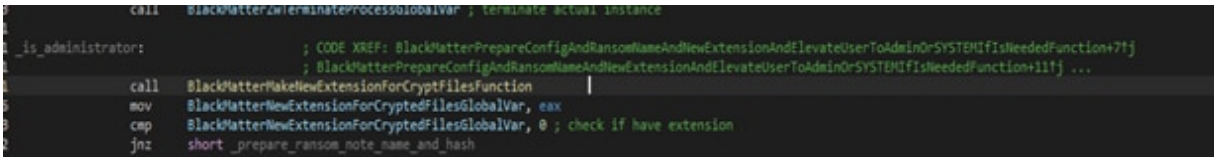


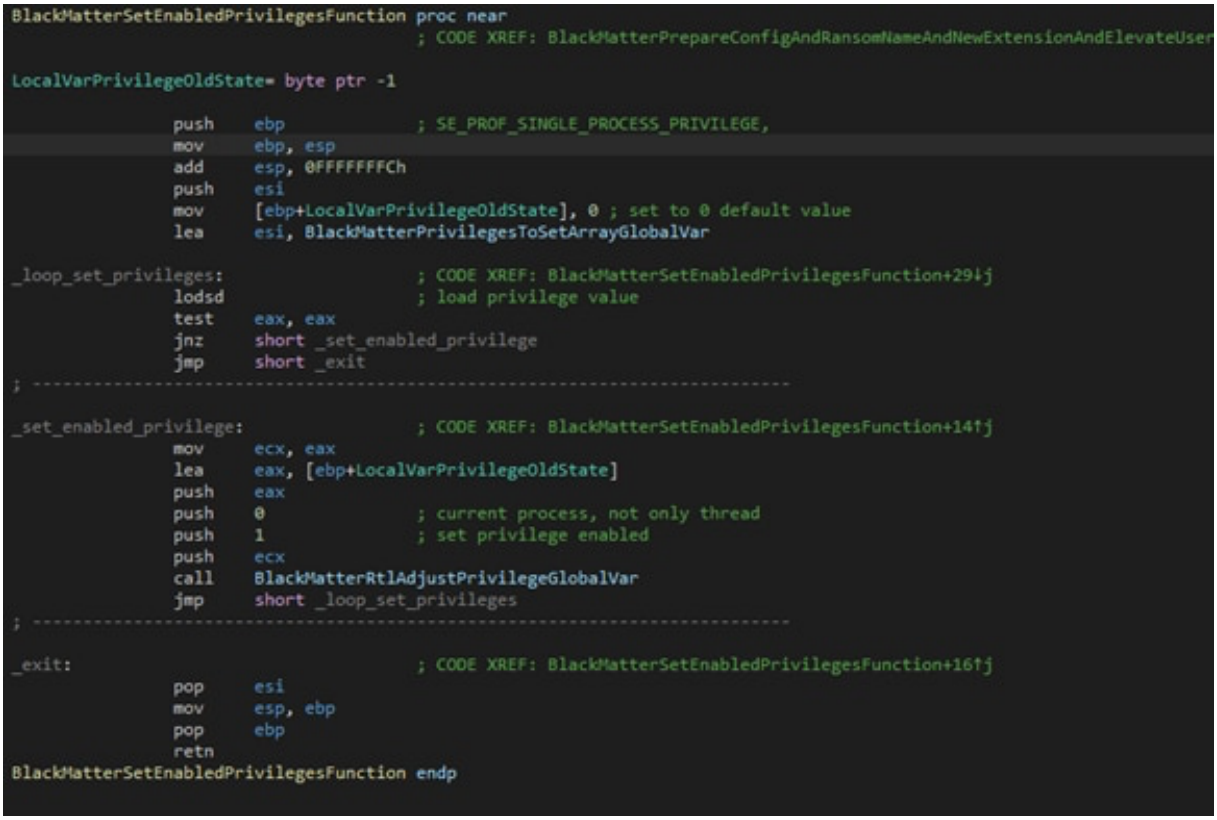
FIGURE 10. Creating the new extension for the encrypted files

Next, the malware will create the ransom note name and calculate the integrity hash of it. The ransom note text is stored encrypted in the malware data. Usually the ransom note name is “%s.README.txt”, where the wildcard is filled with the new extension generated previously.

The next step is to get privileges that will be needed later; BlackMatter tries to get many privileges:

- **SE_BACKUP_PRIVILEGE**
- **SE_DEBUG_PRIVILEGE, SE_IMPERSONATE_PRIVILEGE**
- **SE_INC_BASE_PRIORITY_PRIVILEGE**

- **SE_INCREASE_QUOTA_PRIVILEGE**
- **SE_INC_WORKING_SET_PRIVILEGE**
- **SE_MANAGE_VOLUME_PRIVILEGE**
- **SE_PROF_SINGLE_PROCESS_PRIVILEGE**
- **SE_RESTORE_PRIVILEGE**
- **SE_SECURITY_PRIVILEGE**
- **SE_SYSTEM_PROFILE_PRIVILEGE**
- **SE_TAKE_OWNERSHIP_PRIVILEGE**
- **SE_SHUTDOWN_PRIVILEGE**



```
BlackMatterSetEnabledPrivilegesFunction proc near
                                        ; CODE XREF: BlackMatterPrepareConfigAndRansomNameAndNewExtensionAndElevateUser

LocalVarPrivilegeOldState= byte ptr -1

                push    ebp             ; SE_PROF_SINGLE_PROCESS_PRIVILEGE,
                mov     ebp, esp
                add     esp, 0FFFFFFFCh
                push    esi
                mov     [ebp+LocalVarPrivilegeOldState], 0 ; set to 0 default value
                lea     esi, BlackMatterPrivilegesToSetArrayGlobalVar

_loop_set_privileges:                   ; CODE XREF: BlackMatterSetEnabledPrivilegesFunction+29↓j
                lodsd                   ; load privilege value
                test    eax, eax
                jnz     short _set_enabled_privilege
                jmp     short _exit
; ---------------------------------------------------------------------------

_set_enabled_privilege:                 ; CODE XREF: BlackMatterSetEnabledPrivilegesFunction+14↑j
                mov     ecx, eax
                lea     eax, [ebp+LocalVarPrivilegeOldState]
                push    eax
                push    0               ; current process, not only thread
                push    1               ; set privilege enabled
                push    ecx
                call    BlackMatterRtlAdjustPrivilegeGlobalVar
                jmp     short _loop_set_privileges
; ---------------------------------------------------------------------------

_exit:                                  ; CODE XREF: BlackMatterSetEnabledPrivilegesFunction+16↑j
                pop     esi
                mov     esp, ebp
                pop     ebp
                retn
BlackMatterSetEnabledPrivilegesFunction endp
```

FIGURE 11. Setting special privileges

After getting the privileges it will check if it has SYSTEM privileges, checking the token of its own process. If it is SYSTEM, it will get the appropriate user for logon with the function "WTSQueryUserToken". This function only can be used if the caller has "SeTcbPrivilege" that, by default, only SYSTEM has.

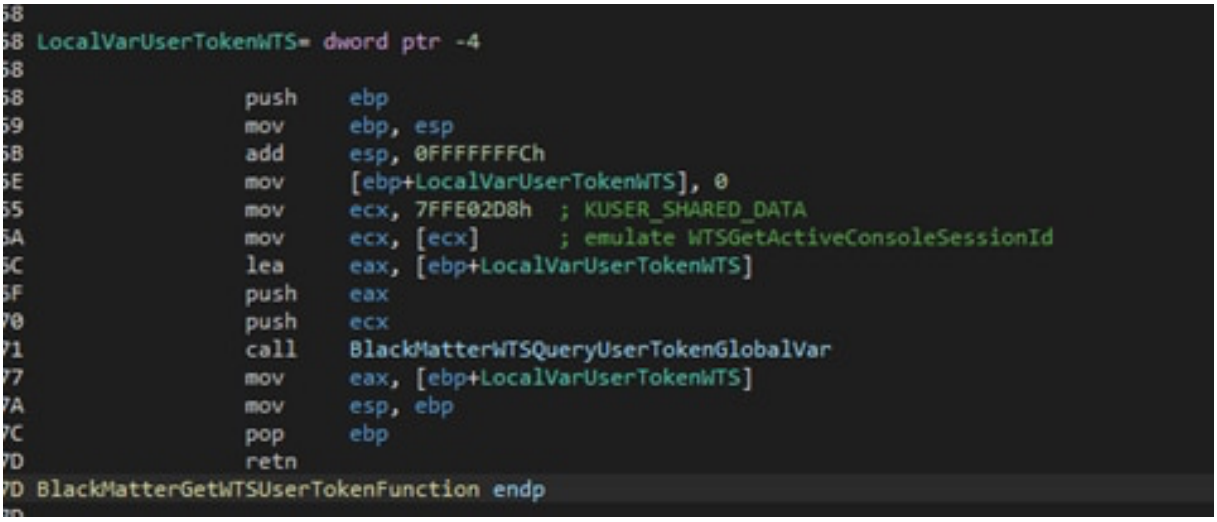


FIGURE 12. Obtaining the token of the logged on user

After getting the token of the logged on user the malware will open the Windows station and desktop.

In the case that it does not have SYSTEM permissions it will enumerate all processes in the system and try to duplicate the token from "explorer.exe" (the name is checked using a hardcoded hash), if it has rights it will continue normally, otherwise it will check again if the token that was duplicated has administrator rights.

In this case it will continue normally but in other cases it will check the operating system version and the CPU (Central Processing Unit) mode (32- or 64- bits). This check is done using the function "ZwQueryInformationProcess" with the class 0x1A (ProcessWow64Information).

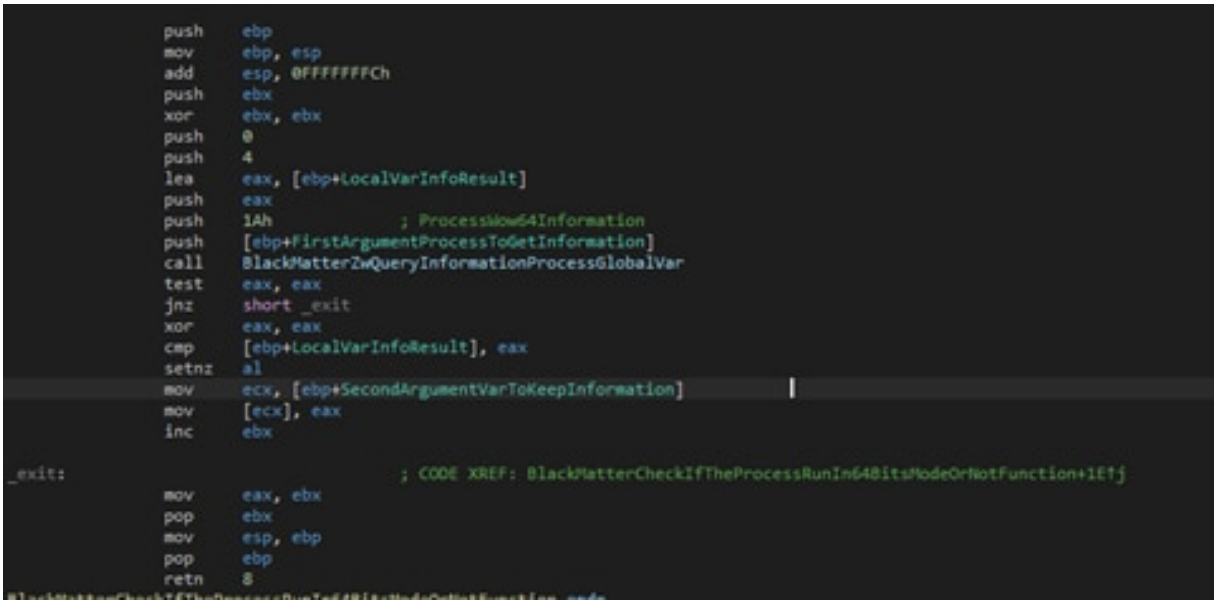


FIGURE 13. Checking if the operating system is 32- or 64-bits

In the case that the system is 32-bits it will decrypt one little shellcode that will inject in one process that will enumerate using the typical “CreateRemoteThread” function. This shellcode will be used to get the token of the process and elevate privileges.

In the case that the system is 64-bits it will decrypt two different shellcodes and will execute the first one that gets the second shellcode as an argument.

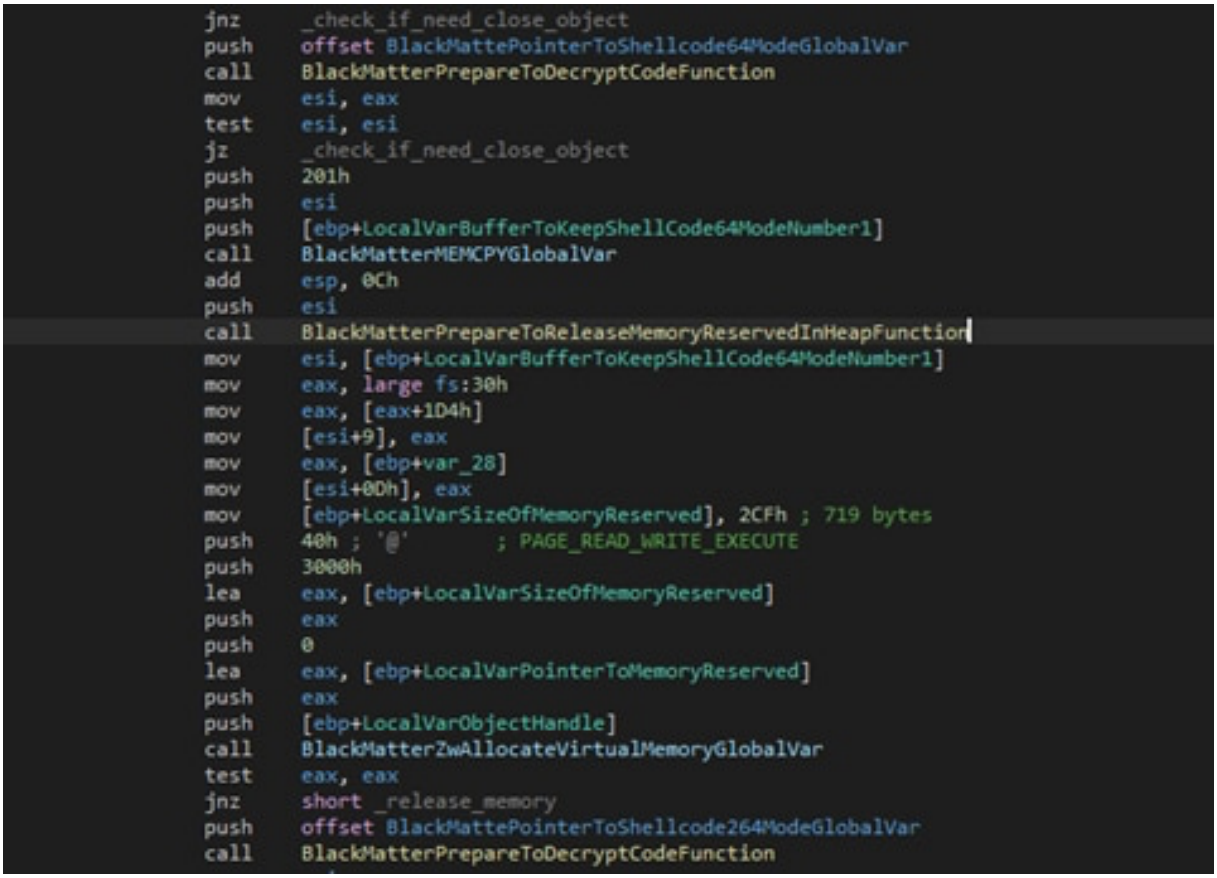


FIGURE 14. BlackMatter preparing shellcodes to steal system token

These shellcodes will allow BlackMatter to elevate privileges in a clean way.

Is important to understand that to get the SYSTEM token BlackMatter will enumerate the processes and get “svchost.exe”, but not only will it check the name of the process, it will also check that the process has the privilege “SeTcbPrivilege”. As only SYSTEM has it by default (and it is one permission that cannot be removed from this “user”) it will be that this process is running under SYSTEM and so it becomes the perfect target to attack with the shellcodes and steal the token that will be duplicated and set for BlackMatter.

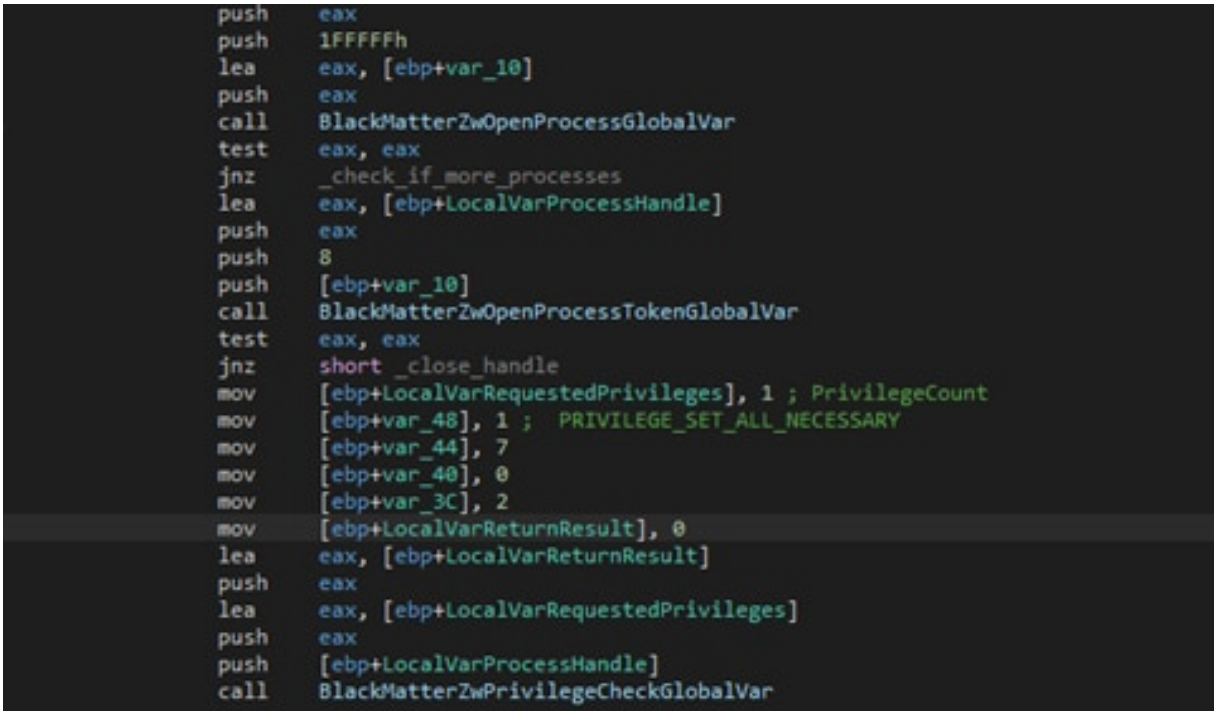


FIGURE 15. Checking if the target process is SYSTEM

After this it will decrypt the configuration that it has embedded in one section. BlackMatter has this configuration encrypted and encoded in base64.

This configuration has a remarkably similar structure to Darkside, offering another clear hint that the developers are one and the same, despite their claims to the contrary.

After decryption, the configuration can get this information:

- **RSA Key used to protect the Salsa20 keys used to encrypt the files.**
- **A 16-byte hex value that remarks the victim id.**
- **A 16-byte hex value that is the AES key that will be used to encrypt the information that will be sent to the C2.**
- **An 8/9-byte array with the behavior flags to control the ransomware behavior.**
- **A special array of DWORDs (values of 4 bytes each one) that keep the values to reach the critical points in the configuration.**
- **Different blocks encoded and, sometimes, encrypted again to offer the field more protection.**

After getting the configuration and parsing it, BlackMatter will start checking if it needs to make a login with some user that is in the configuration. In this case it will use the function “LogonUser” with the information of the user(s) that are kept in the configuration; this

information has one user and one password: “test@enterprise.com:12345” where “test” is the user, “@enterprise.com” is the domain and “12345” the password.

The next action will be to check with the flag to see if a mutex needs to be created to avoid having multiple instances.

This mutex is unique per machine and is based in the registry entry “MachineGuid” in the key “Cryptography”. If the system has this mutex already the malware will finish itself.

Making a vaccine with a mutex can sometimes be useful but not in this case as the developers change the algorithm and only need to set the flag to false to avoid creating it.

```
push    ebp
mov     ebp, esp
add     esp, 0FFFFFFE0h
push    ebx
cmp     BlackMatterNeedCreateMutexGlobalVar, 0
jz      short _check_if_need_send_info
call    BlackMatterCreateMutexFromMachineGuidValueFromRegistryFunction
mov     [ebp+LocalVarMutexNameString], eax
push    [ebp+LocalVarMutexNameString]
push    0
push    100000h
call    BlackMatterOpenMutexW
mov     [ebp+LocalVarMutexHandle], eax
cmp     [ebp+LocalVarMutexHandle], 0
jz      short _create_mutex
push    [ebp+LocalVarMutexHandle]
call    BlackMatterZwCloseGlobalVar
pop     ebx
mov     esp, ebp
pop     ebp
retn
```

FIGURE 16. Creation of the mutex to avoid multiple instances

After, it will check if it needs to send information to the C2. If it does (usually, but not always) it will get information of the victim machine, such as username, computer name, size of the hard disks, and other information that is useful to the malware developers to know how many machines are infected.

This information is encoded with base64 and encrypted with AES using the key in the configuration.

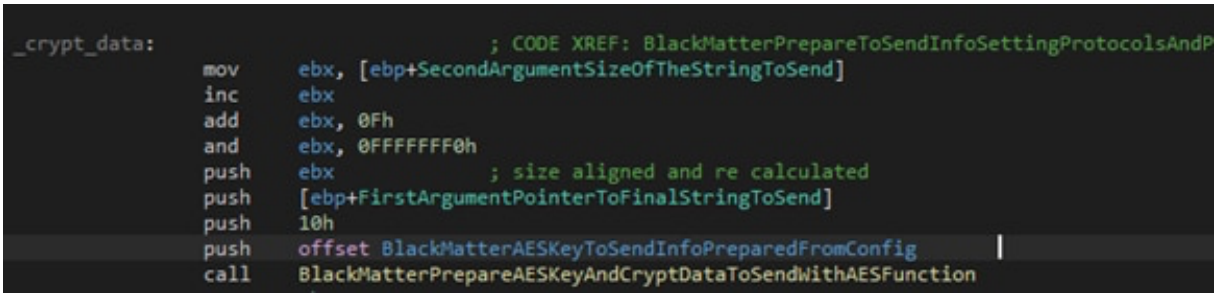


FIGURE 17. Encrypted information sent to the C2

The C2 addresses are in the configuration (but not all samples have them, in this case the flag to send is false). The malware will try to connect to the C2 using a normal protocol or will use SSL checking the initial "http" of the string.

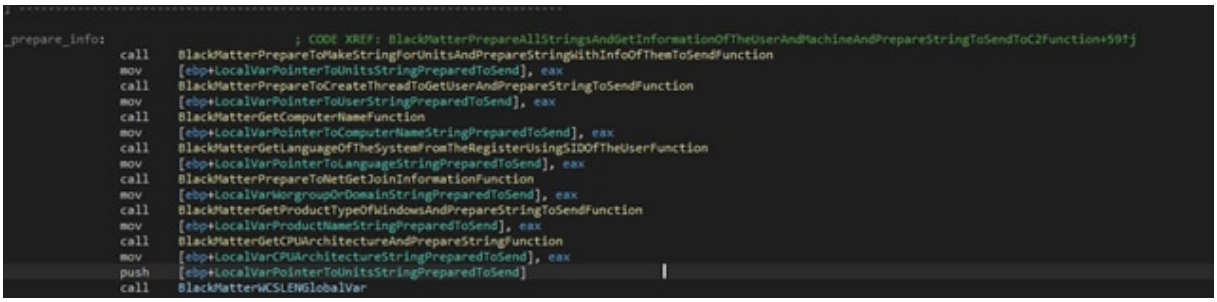


FIGURE 18. Get information of the victim machine and user

The information is prepared in some strings decrypted from the malware and sent in a POST message.

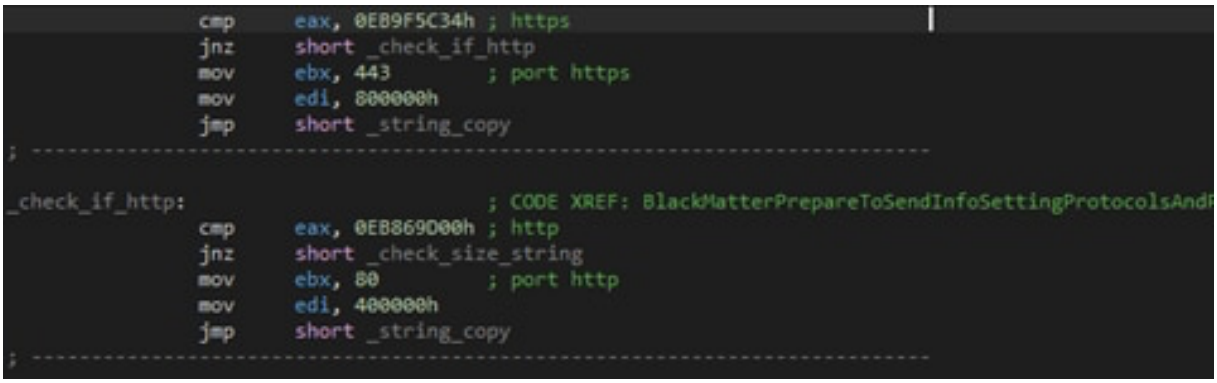


FIGURE 19. Choose to send by HTTP or HTTPS

The message has values to mislead checks and to try and hide the true information as garbage. This "fake" data is calculated randomly.

The C2 returns garbage data but the malware will check if it starts and ends with the characters "{" and "}"; if it does the malware will ignore sending the information to another C2.

```
_check_data_from_c2:                    ; CODE XREF: BlackMatterPrepareToSendI
                mov     eax, [ebp+LocalVarReserveMemory]
                mov     ecx, [ebp+var_2C]
                cmp     byte ptr [eax], 7Bh ; '{'
                jnz     short _release_memory
                cmp     byte ptr [eax+ecx-1], 7Dh ; '}'
                jnz     short _release_memory
                jmp     short _check_if_open_handle
; ---------------------------------------------------------------------------
                jmp     short loc_40AA21
; ---------------------------------------------------------------------------
```

FIGURE 20. Checking for a reply from the C2 after sending

BlackMatter is a multithread application and the procedure to send data to the C2 is done by a secondary thread.

After that, BlackMatter will enumerate all units that are FIXED and REMOVABLE to destroy the recycle bin contents. The malware makes it for each unit that has it and are the correct type. One difference with DarkSide is that it has a flag for this behavior while BlackMatter does not.

The next action is to delete the shadow volumes using COM to try and avoid detection using the normal programs to manage the shadow volumes. This differs with DarkSide that has a flag for this purpose.

```
                mov     dword ptr [eax], 0E9830A21h
                mov     dword ptr [eax+4], 33D7CEC5h
                mov     dword ptr [eax+8], 0E206C440h
                mov     dword ptr [eax+0Ch], 0DDFB87A2h ; CLSID_WbemAdministrativeLocator
                mov     ecx, 4

_loop_decrypt_data:                     ; CODE XREF: BlackMatterDestroyShadowVolumesUsingCOMFunction+62↓j
                xor     dword ptr [eax], 22065FEDh
                add     eax, 4
                dec     ecx
                jnz     short _loop_decrypt_data
                lea     eax, [ebp+LocalVarIID_IWbemLocatorString]
                mov     dword ptr [eax], 0FE14F96Ah
                mov     dword ptr [eax+4], 33C92C92h
                mov     dword ptr [eax+8], 88061265h
                mov     dword ptr [eax+0Ch], 62814EDh ; IID_IWbemLocator
                mov     ecx, 4

_loop_decrypt_data_:                    ; CODE XREF: BlackMatterDestroyShadowVolumesUsingCOMFunction+91↓j
                xor     dword ptr [eax], 22065FEDh
                add     eax, 4
                dec     ecx
                jnz     short _loop_decrypt_data_
                lea     eax, [ebp+LocalVarWBEM_ContextString]
                mov     dword ptr [eax], 454D3975h
                mov     dword ptr [eax+4], 33D6817Fh
                mov     dword ptr [eax+8], 0E2062E40h
                mov     dword ptr [eax+0Ch], 0DDFB87A2h ; CLSID_WbemContext
                mov     ecx, 4

_loop_decrypt_data__:                   ; CODE XREF: BlackMatterDestroyShadowVolumesUsingCOMFunction+C0↓j
```

FIGURE 21. Destruction of the shadow volumes using COM

BlackMatter will check another flag and will enumerate all services based on one list in the configuration and will stop target services and delete them.

This behavior is the same as DarkSide.

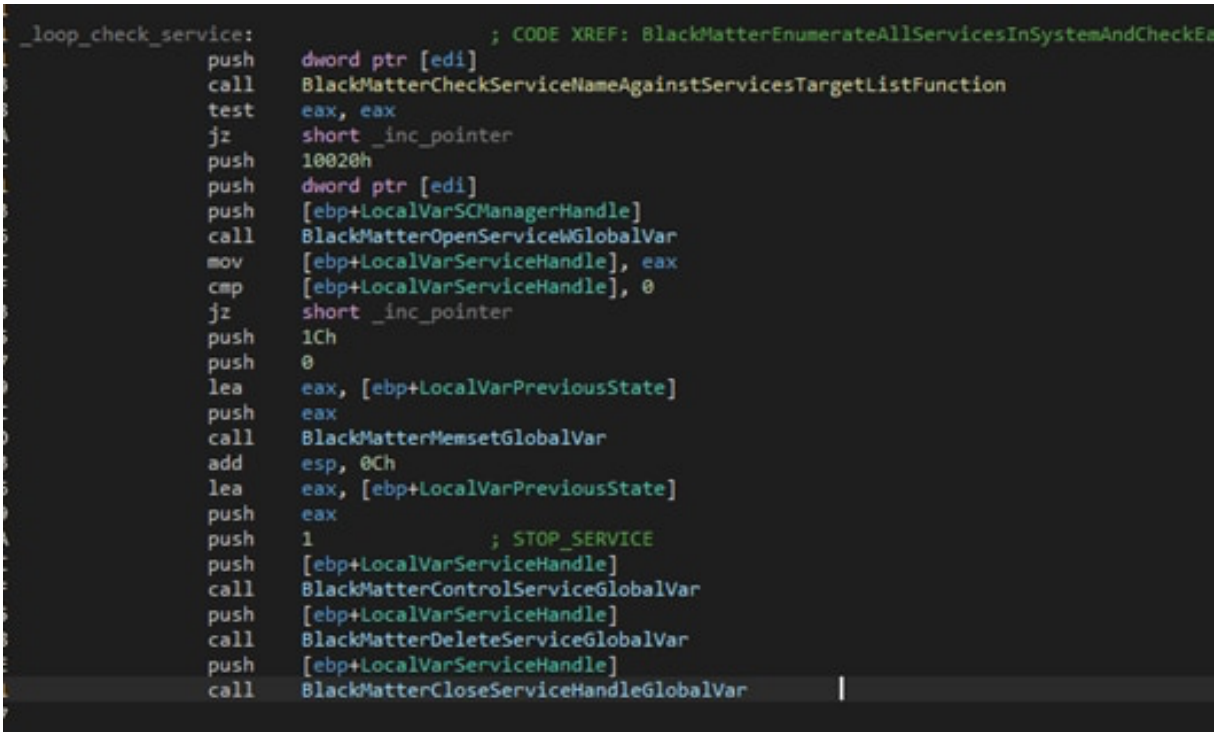


FIGURE 22. Stopping services and deleting them

Processes will be checked and terminated as with DarkSide, based on other configuration flags.

After terminating the processes BlackMatter will stop the threads from entering suspension or hibernating if someone is using the computer to prevent either of those outcomes occurring when it is encrypting files. This is done using the function “ZwSetThreadExecutionState”.

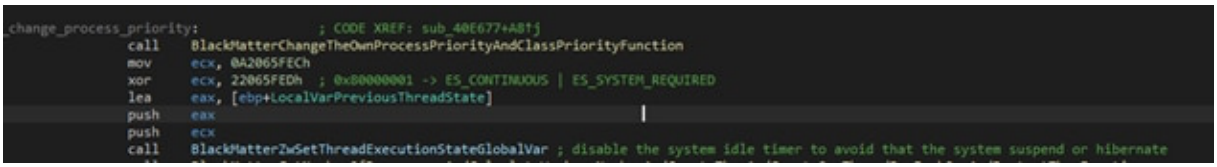


FIGURE 23. Preventing the machine being suspended or hibernated

The next action will be to enumerate all units, fixed and on the network, and create threads to encrypt the files. BlackMatter uses Salsa20 to encrypt some part of the file and will save a new block in the end of the file, protected with the RSA key embedded in the configuration with the Salsa20 keys used to encrypt it. This makes BlackMatter slower than many other ransomwares.

After the encryption it will send to the C2 all information about the encryption process, how many files were crypted, how many files failed, and so on. This information is sent in the manner previously described, but only if the config is set to true.

```
                call    BlackMatterPrepareToSendToC2FilesInformationAfterCryptProcessFunction

_check_if_need_destroy_mutex:           ; CODE XREF: sub_40E677+115↑j
                cmp     BlackMatterNeedCreateMutexGlobalVar, 0
                jz      short _check_if_boot_in_safe_mode
                cmp     [ebp+LocalVarMutexHandle], 0
                jz      short _check_if_boot_in_safe_mode
                push    [ebp+LocalVarMutexHandle]
                call    BlackMatterReleaseMutexGlobalVar
                push    [ebp+LocalVarMutexHandle]
                call    BlackMatterZwCloseGlobalVar
```

FIGURE 24. Release of the mutex

If one mutex was created in this moment it will be released. Later it will check the way that the machine boots with the function “GetSystemMetrics”. If the boot was done in Safe Mode BlackMatter will set some keys for persistence in the registry for the next reboot and then attack the system, changing the desktop wallpaper.

```
_check_if_boot_in_safe_mode:            ; CODE XREF: sub_40E677+15E↑j
                                        ; sub_40E677+164↑j
                push    43h ; 'C'       ; SM_CLEANBOOT
                call    BlackMatterGetSystemMetricsGlobalVar
                test    eax, eax
                jnz     short _boot_in_safe_mode
                push    1               ; change wallpaper in runtime (0 means not, need a reboot for it)
                call    BlackMatterPrepareNewWallPaperAndCreateFileWithItAndSetNewWallPaperIfIsNeededFunction
                jmp     short _exit
; ---------------------------------------------------------------------------
```

FIGURE 25. Determining whether the system boots in safe mode or normal mode

Of course, it will disable the safeboot options in the machine and reboot it (it is one of the reasons why it needs the privilege of shutdown).

To ensure it can launch in safe mode, the persistence key value with the path of the malware will start with a ‘*’.

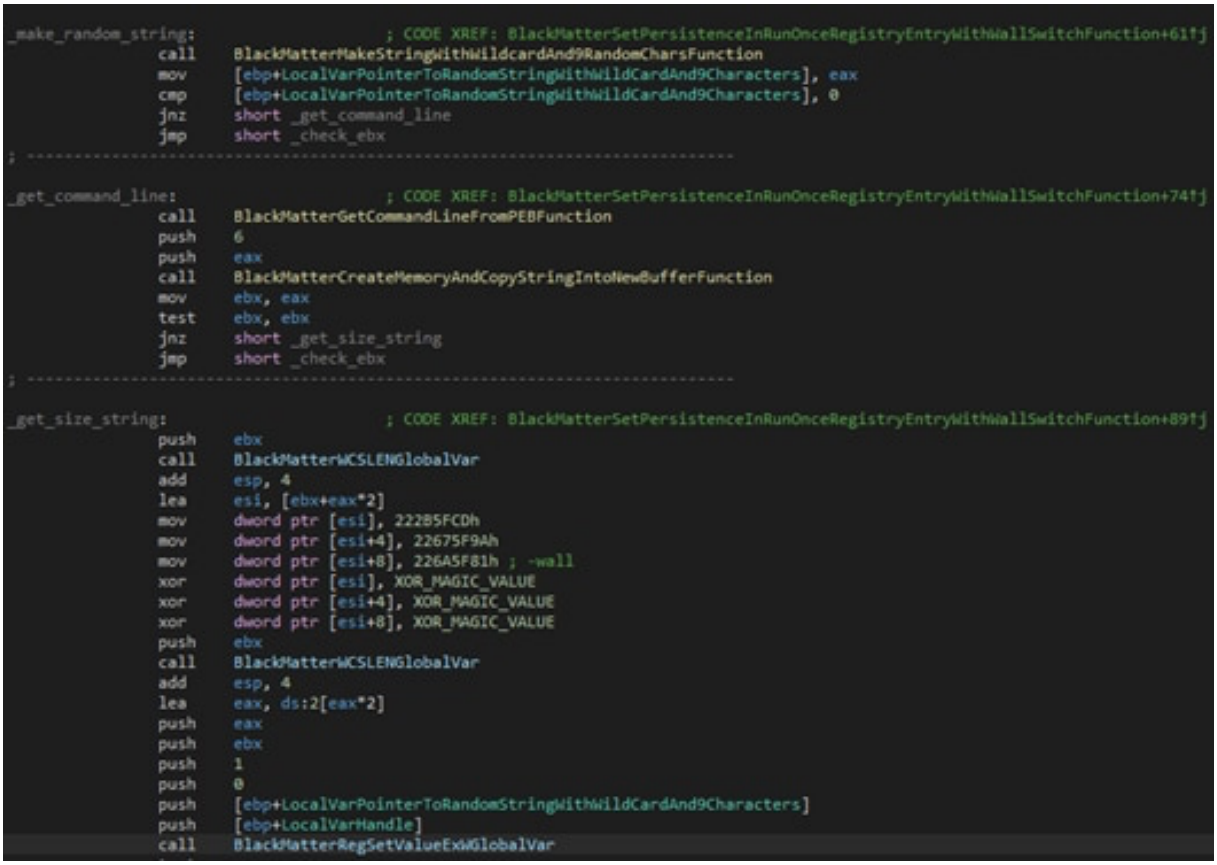


FIGURE 26. Setting the persistance registry key

If the machine starts in the normal way, it will change the desktop wallpaper with an alternative generated in runtime with some text about the ransom note.

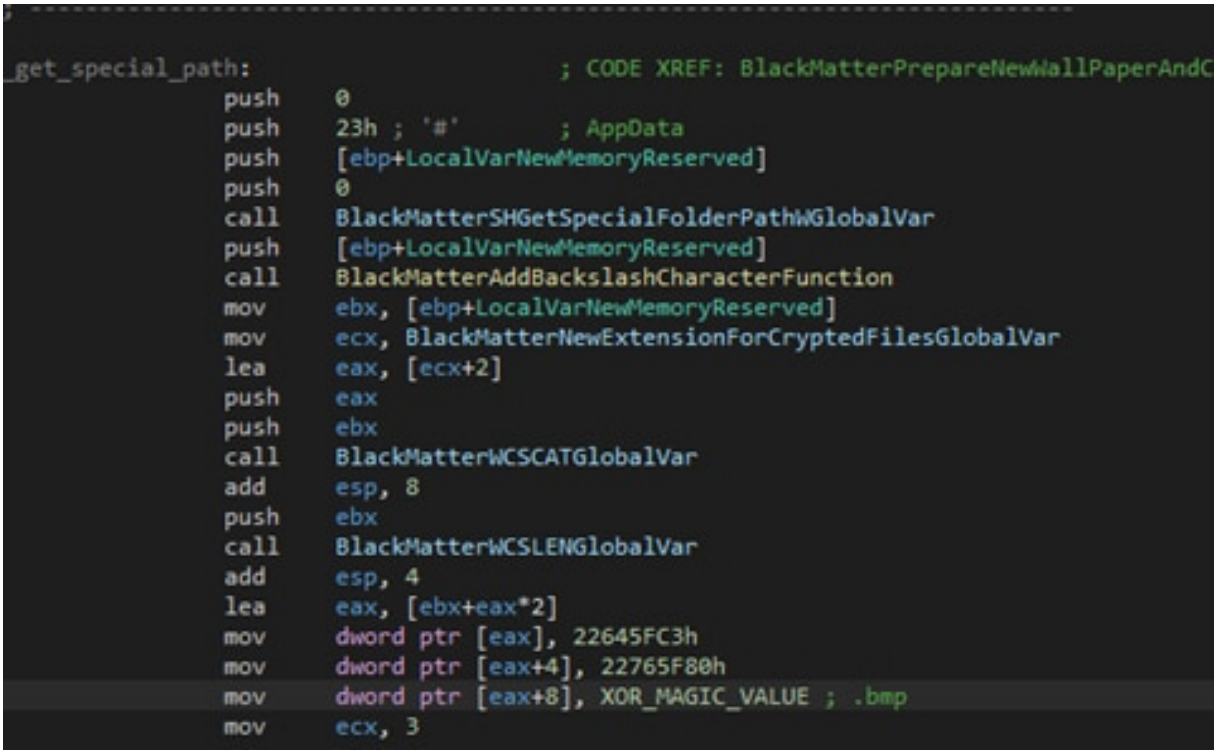


FIGURE 27. BlackMatter makes the new wallpaper in runtime

VERSIONS 1.9 AND 2.0

The new versions have some differences compared with versions 1.2 to 1.6:

- Changes in the stub generation code. Previously only one type of stub was used, but in more recent versions several types of stubs are employed, with one chosen randomly per function. Anyways the stubs can be removed without any problem by patching the binary.
- A new byte flag in the configuration that remarks if it needs to print the ransom note using the available printer in the system. Very similar to Ryuk but instead BlackMatter uses APIs from “winspool.drv”.
- Removed one C2 domain that was shut down by the provider.

Additional changes in version 2.0:

- This version changes the crypto algorithm to protect the configuration making it more complex to decrypt it.
- Removed the last C2 that was shut down by the provider.
- Added a new C2 domain.

These changes suggest the developers are active on social media, with an interest in malware and security researchers.

VACCINE

Unlike some ransomware we've seen in the past, such as GandCrab , BlackMatter has good code, but it does have some design flaws that can be used in some cases to avoid having the malware encrypt the files.

This vaccine **is not intended to be used** in the normal way, rather only in special cases as, while it works, other **programs can be affected** (we obviously cannot test all third party programs but potential issues are likely to include **data corruption** and **unpredictable behavior**), and the fix **is not permanent**.

Steps to make the vaccine (**proceed at your own risk**):

- Open regedit (or another registry editor) and go to the key in HKEY_LOCAL_MACHINE> Cryptography.
- In this key can be seen a string value named "MachineGuid" with a special value. This value is unique for the machine and is used for some applications to identify the machine. BlackMatter uses it to make the mutex and, very importantly, the new extension for the encrypted files.
- Make a new value of type string with a random name and put the same value as seen in "MachineGuid" to have a backup of it.
- Remove the "MachineGuid" value, and then make it again but with the binary type Instead of string type, with the same name, "MachineGuid".
- Close the registry editor.

In this moment BlackMatter cannot affect the machine as it needs the registry key to make the ransom extension, and the most important thing is, if it cannot make it, it will return the function WITHOUT decrypting the config that is needed too. In this case it will destroy the recycle bin and shadow volumes anyways but later it will finish as it does not have any behavior to do, RSA Key to protect the files, or anything to send to the C2 as the flag was never read from the config (and the default values are false for all of them).

Though the behavior of other programs may be unpredictable, the vaccine is easy to make, and the system will boot, showing that the BlackMatter programmers made a mistake in the design of the code.

This vaccine works for all versions, including 2.0.

MITRE ATT&CK

The sample uses the following MITRE ATT&CK™ techniques:

Technique ID	Technique Description	Observable
T1134	Access Token Manipulation	BlackMatter accesses and manipulates different process tokens.
T1486	Data Encrypted for Impact	BlackMatter encrypts files using a custom Salsa20 algorithm and RSA.
T1083	File and Directory Discovery	BlackMatter uses native functions to enumerate files and directories searching for targets to encrypt.
T1222.001	Windows File and Directory Permissions Modification	BlackMatter executes the command icacls “<DriveLetter>:*” /grant Everyone: F /T /C /Q to grant full access to the drive.
T1562.001	Disable or Modify Tools	BlackMatter stops services related to endpoint security software.
T1106	Native API	BlackMatter uses native API functions in all code.
T1057	Process Discovery	BlackMatter enumerates all processes to try to discover security programs and terminate them.
T1489	Service Stop	BlackMatter stops services.
T1497.001	System Checks	BlackMatter tries to detect debuggers, checking the memory reserved in the heap.
T1135	Network Share Discovery	BlackMatter will attempt to discover network shares by building a UNC path in the following format for each driver letter, from A to Z: \\<IP>\<drive letter>$

T1082	System Information Discovery	BlackMatter uses functions to retrieve information about the target system.
T1592	Gather Victim Host Information	BlackMatter retrieves information about the user and machine.
T1070	Valid Accounts	BlackMatter uses valid accounts to logon to the victim network.
T1547	Boot or Logon Autostart Execution	BlackMatter installs persistence in the registry.
T1102	Query Registry	BlackMatter queries the registry for information.
T1018	Remote System Discovery	BlackMatter enumerates remote machines in the domain.
T1112	Modify Registry	BlackMatter changes registry keys and values and sets new ones.

CONCLUSION

BlackMatter is a new threat in the ransomware field and its developers know full well how to use it to attack their targets. The coding style is remarkably similar to DarkSide and, in our opinion, the people behind it are either the same or have a very close relationship.

BlackMatter shares a lot of ideas, and to some degree code, with DarkSide:

- Configurations are remarkably similar, especially with the last version of Darkside, besides the change in the algorithm to protect it which, despite having less options, remains with the same structure. We do not think that the developers of BlackMatter achieved this similarity by reversing DarkSide as that level of coding skill would have allowed them to create an entirely new ransomware from the ground up. Also, the idea that the DarkSide developers gave or sold the original code to them does not make any sense as it is an old product.
- Dynamic functions are used in a similar way to DarkSide.
- It uses the same compression algorithm for the configuration.
- The victim id is kept in the same way as DarkSide.

It is important to keep your McAfee Enterprise products updated to the latest detections and avoid insecure remote desktop connections, maintain secure passwords that are changed on a regular basis, take precautions against phishing emails, and do not connect unnecessary devices to the enterprise network.

Despite some effective coding, mistakes have been made by the developers, allowing the program to be read, and a vaccine to be created, though we will stress again that it can affect other programs and is not a permanent solution and should be employed only if you accept the risks associated with it.