



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.

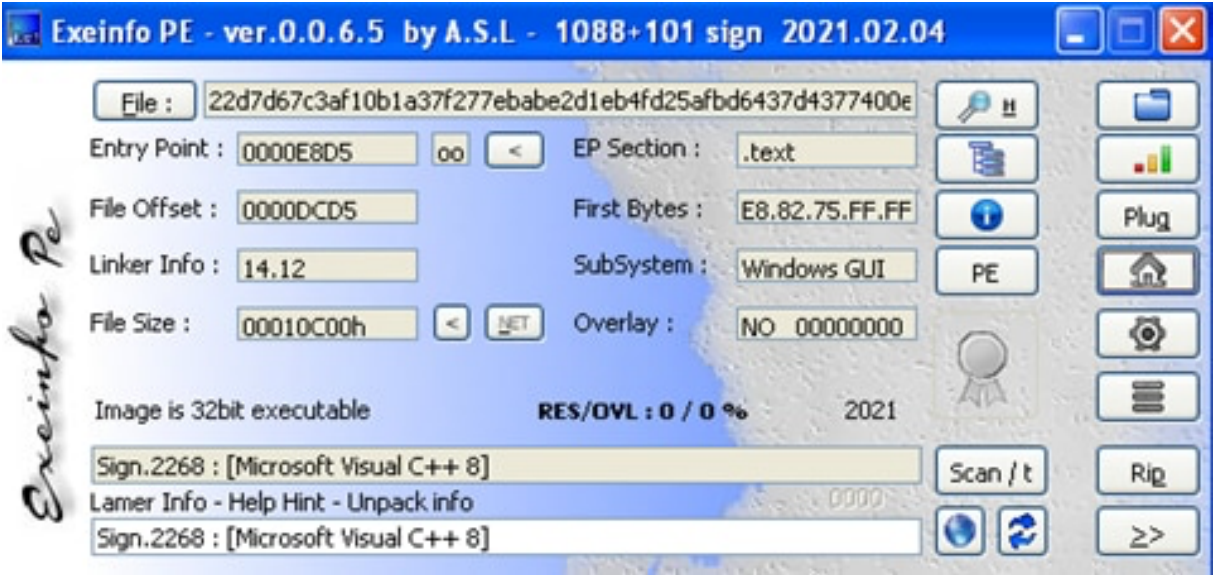


FIGURE 1. Information about the malware

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.

```

text:0040581D
text:0040581D BlackMatterGetProcAddressAndLoadLibraryIfIsNeededAndGetMoreAPIUsingThemFunction proc near
text:0040581D ; CODE XREF: BlackMatterGetProcAddressAndLoadLibraryIfIsNeededAndGetMoreAPIUsingThem
text:0040581D ; BlackMatterGetProcAddressAndLoadLibraryIfIsNeededAndGetMoreAPIUsingThemFunction+4F
text:0040581D
text:0040581D var_10 = dword ptr -10h
text:0040581D var_C = dword ptr -0Ch
text:0040581D var_8 = dword ptr -8
text:0040581D LocalVarAPIFoundedFromHashFounded= dword ptr -4
text:0040581D arg_0 = dword ptr 8
text:0040581D
text:0040581D push ebp
text:0040581E mov ebp, esp
text:00405820 add esp, 0FFFFFF0h
text:00405823 push ebx
text:00405824 push esi
text:00405825 push edi
text:00405826 cmp BlackMatterLoadLibraryGlobalVar, 0
text:0040582D jnz short _check_if_have_load_library
text:0040582F mov eax, 5D6015Fh
text:00405834 xor eax, 22065FEDh
text:00405839 mov BlackMatterLoadLibraryGlobalVar, eax
text:0040583E push BlackMatterLoadLibraryGlobalVar
text:00405844 call BlackMatterGetProcAddressAndLoadLibraryIfIsNeededAndGetMoreAPIUsingThemFunction
text:00405849 mov BlackMatterLoadLibraryGlobalVar, eax
text:0040584E
text:0040584E _check_if_have_load_library: ; CODE XREF: BlackMatterGetProcAddressAndLoadLibraryIfIsNeededAndGetMoreAPIUsingThem
text:0040584E cmp BlackMatterGetProcAddressGlobalVar, 0
text:00405855 jnz short _enumerate_modules_from_PEB
text:00405857 mov eax, 99952FB1h
text:0040585C xor eax, 22065FEDh
text:00405861 mov BlackMatterGetProcAddressGlobalVar, eax
text:00405866 push BlackMatterGetProcAddressGlobalVar
text:0040586C call BlackMatterGetProcAddressAndLoadLibraryIfIsNeededAndGetMoreAPIUsingThemFunction
text:00405871 mov BlackMatterGetProcAddressGlobalVar, eax

```

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.

```

push ebx
push esi
push edi
mov esi, [ebp+SecondArgumentPointerToArrayWithPrecomputedHashesToGet]
lodsd
xor eax, 22065FEDh
push eax
call BlackMatterGetEnumerateFilesFunctionsAndLoadTargetModuleToHaveCleanInMemoryFunction
test eax, eax
jz short _exit
mov edi, [ebp+FirstArgumentPointerToGlobalVarArrayToSaveStubs]
add edi, 4

_loop_manage_next_hashes: ; CODE XREF: BlackMatterGetFunctionsFromModuleAndPrepareStubsToHideThemFunction+6B1j
lodsd ; load in EAX hash value of API
cmp eax, 0CCCCCCCCh ; check against this value thats means that the array finish
jnz short _prepare_hash
jmp short _exit

_prepare_hash: ; CODE XREF: BlackMatterGetFunctionsFromModuleAndPrepareStubsToHideThemFunction+251j
xor eax, 22065FEDh
push eax
call BlackMatterGetProcAddressAndLoadLibraryIfIsNeededAndGetMoreAPIUsingThemFunction
mov ebx, eax ; EBX -> Pointer to founded function
push 0Ch ; 12 bytes
push 0
push [ebp+ThirdArgumentPointerToHeap]
call [ebp+FourthArgumentPointerToRtlAllocateHeap]
mov ecx, 89ADF446h
xor ecx, 22065FEDh
cmp [eax+0Ch], ecx ; check after the heap memory reserved if the value is 0xABABABAB
nop ; patched protection! if the compare is ok, avoid save the address allocated
nop ; this protection is based in the Themida checks against 2 values with 0xABABABAB as the heap flag HEAP_TAI
stosd

```

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.

```

mov     ecx, 09AD7446h
xor     ecx, 22065FEDh
cmp     [eax+0Ch], ecx ; check after the heap memory reserved if the value is 0xABABABAB
nop
nop
nop
; patched protection! if the compare is ok, avoid save the address allocated
; this protection is based in the Themida checks against 2 values with 0xABABABAB as the heap flag HEAP_TAIL_CHECKING_ENABLED
stosd
mov     ecx, 22065FEDh ; if the protection works will reach here before make the stosd
mov     byte ptr [eax], 088h ; ; set first byte in the stub as opcode 0x88 (mov eax, xxxxx)
xor     ebx, ecx ; xor function address with hardcoded value
mov     [eax+1], ebx ; save xored value in the stub
mov     byte ptr [eax+5], 35h ; 5 ; save in stub the opcode 0x35 (xor eax, xxxxx)
mov     [eax+6], ecx ; set hardcoded value to xor later and recover function pointer
mov     word ptr [eax+0Ah], 0E0FFh ; set opcode in the stub 0xFFE0 (jmp eax) call the function pointer
jmp     short_loop_manage_next_hash

```

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.

```

align 4
BlackMatterPrecomputedNTDLLHashesGlobalVar dd 6310285Ah, 82EF6E32h, 71F3EF17h, 8521D0A9h, 4C57BCF6h
; DATA XREF: BlackMatterPrepareToGetLoadLibraryAndGetProcAddressAndLoadM
dd 0AC2F8FB6h, 38024E22h, 22146F8Eh, 0CC6C6CCFh, 2C3468CEh
dd 0A2241FB7h, 4C546087h, 4C7C1CF7h, 0AC441BF7h, 4C441EF7h
dd 2062007Fh, 7FD62B8Eh, 36822FA7h, 250229CEh, 0BA318DFh
dd 0DD687B69h, 0A67DA323h, 1CE03CB4h, 626F5B23h, 8BF8AF6Ch
dd 1A6163h, 0850D9E55h, 32B98521h, 7D8D0729h, 2BBE029Fh
dd 59C91E6Eh, 25CE376Ch, 5AEC64FAh, 83AFBF82h, 7ACA2F40h
dd 4E600EF2h, 84C48A1Ch, 33F39897h, 9F287DD4h, 7C2F16F3h
dd 2E8D683Dh, 48946FF4h, 3839A2F7h, 94E355D8h, 83F796E2h
dd 3761D74h, 6E812ED7h, 0CCCCCCCC

```

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.

```

1 FirstArgumentThreadHandleOrNull= dword ptr 8
2
3     push    ebp
4     mov     ebp, esp
5     xor     eax, eax
6     inc     eax
7     inc     eax
8     lea    ecx, ds:[eax*8] ; ThreadInformationClass -> 0x11 (17) ThreadHideFromDebugger
9     cmp     [ebp+FirstArgumentThreadHandleOrNull], 0
10    jnz     short _set_eax_from_argument
11    neg     eax
12    jmp     short _zw_set_information_thread ; protection patched! Was a ZwSetInformationThread call with the class 17
13
14 -----
15
16 _set_eax_from_argument:                ; CODE XREF: BlackMatterSetThreadInformationToHideItFromDebuggerFunction+127j
17     nop
18     nop
19     nop
20
21 _zw_set_information_thread:           ; CODE XREF: BlackMatterSetThreadInformationToHideItFromDebuggerFunction+167j
22     nop                                ; protection patched! Was a ZwSetInformationThread call with the class 17
23     nop                                ; that is ThreadHideFromDebugger, if this call happens with a debugger, the debugger will hang
24     nop
25     nop

```

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.

```

4 BlackMatterPrepareConfigAndRansomNameAndNewExtensionAndElevateUserToAdminOrSYSTEMIfIsNeededFunction proc near
5     ; CODE XREF: start+C1p
6     call    BlackMatterCheckIfBelongsToAdministratorLocalGroupUsingTokenFunction
7     test   eax, eax
8     jnz    short _is_administrator
9     call    do_os_version_check ; check operative system using PEB
10    cmp    eax, 3Ch ; '<'
11    jbe    short _is_administrator

```

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.

```
cmp     BlackMatterThirdReservedMemoryGlobalVar, 0
jz      _exit
push   BlackMatterThirdReservedMemoryGlobalVar
call   BlackMatterPrepareWindowsSystem32UnicodeStringFunction
push   BlackMatterThirdReservedMemoryGlobalVar
call   BlackMatterAddBackslashCharacterFunction
lea    eax, [ebp+LocalVarDllHostString]
mov    dword ptr [eax], 226A5F89h
mov    dword ptr [eax+4], 226E5F81h
mov    dword ptr [eax+8], 22755F82h
mov    dword ptr [eax+0Ch], 22285F99h
mov    dword ptr [eax+10h], 227E5F88h
mov    dword ptr [eax+14h], 22065F88h ; dllhost.exe
mov    ecx, 6
```

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).

```
BlackMatterLoadedModulesLdrCallbackFunction proc near
; DATA XREF: BlackMatterReserveMemor

arg_0      = dword ptr 8
arg_4      = dword ptr 0Ch
arg_8      = dword ptr 10h

push     ebp
mov      ebp, esp
push    ebx
mov      ebx, [ebp+arg_0]
mov      eax, [ebp+arg_4]
mov      eax, [eax+8]
cmp      eax, [ebx+18h]
jnz     short _set_zeroes
lea     eax, [ebx+24h]
push    BlackMatterThirdReservedMemoryGlobalVar
push    eax
call    BlackMatterRtlInitUnicodeStringGlobalVar
lea     eax, [ebx+2Ch]
push    BlackMatterFirstReservedMemoryGlobalVar
push    eax
call    BlackMatterRtlInitUnicodeStringGlobalVar
mov     eax, [ebp+arg_8]
mov     byte ptr [eax], 1 ; set one
jmp     short _exit
; -----
```

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.

```
call BlackMatterZwTerminateProcessGlobalVar ; terminate actual instance
; is_administrator: ; CODE XREF: BlackMatterPrepareConfigAndRansomNameAndNewExtensionAndElevateUserToAdminOrSYSTEMIfIsNeededFunction+71j
; ; BlackMatterPrepareConfigAndRansomNameAndNewExtensionAndElevateUserToAdminOrSYSTEMIfIsNeededFunction+111j ...
call BlackMatterMakeNewExtensionForCryptFilesFunction
mov BlackMatterNewExtensionForCryptedFilesGlobalVar, eax
cmp BlackMatterNewExtensionForCryptedFilesGlobalVar, 0 ; check if have extension
jnz short_prepare_ransom_note_name_and_hash
```

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.

```
58 LocalVarUserTokenWTS= dword ptr -4
58
58      push    ebp
59      mov     ebp, esp
5B      add     esp, 0FFFFFFCh
5E      mov     [ebp+LocalVarUserTokenWTS], 0
65      mov     ecx, 7FFE02D8h ; KUSER_SHARED_DATA
6A      mov     ecx, [ecx] ; emulate WTSGetActiveConsoleSessionId
6C      lea   eax, [ebp+LocalVarUserTokenWTS]
6F      push   eax
70      push   ecx
71      call  BlackMatterWTSQueryUserTokenGlobalVar
77      mov   eax, [ebp+LocalVarUserTokenWTS]
7A      mov   esp, ebp
7C      pop   ebp
7D      retn
7D BlackMatterGetWTSUserTokenFunction endp
7E
```

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).

```
push    ebp
mov     ebp, esp
add     esp, 0FFFFFFFh
push    ebx
xor     ebx, ebx
push    0
push    4
lea    eax, [ebp+LocalVarInfoResult]
push    eax
push    1Ah          ; ProcessWow64Information
push    [ebp+FirstArgumentProcessToGetInformation]
call   BlackMatter2wQueryInformationProcessGlobalVar
test   eax, eax
jnz    short _exit
xor     eax, eax
cmp    [ebp+LocalVarInfoResult], eax
setnz  al
mov    ecx, [ebp+SecondArgumentVarToKeepInformation]
mov    [ecx], eax
inc    ebx

_exit:
; CODE XREF: BlackMatterCheckIfTheProcessRunIn64BitsModeOrNotFunction+1E1j
mov    eax, ebx
pop    ebx
mov    esp, ebp
pop    ebp
retn   8
```

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.

```

jnz    _check_if_need_close_object
push   offset BlackMattePointerToShellcode64ModeGlobalVar
call   BlackMatterPrepareToDecryptCodeFunction
mov    esi, eax
test   esi, esi
jz     _check_if_need_close_object
push   201h
push   esi
push   [ebp+LocalVarBufferToKeepShellCode64ModeNumber1]
call   BlackMatterMEMCPYGlobalVar
add    esp, 0Ch
push   esi
call   BlackMatterPrepareToReleaseMemoryReservedInHeapFunction
mov    esi, [ebp+LocalVarBufferToKeepShellCode64ModeNumber1]
mov    eax, large fs:30h
mov    eax, [eax+1D4h]
mov    [esi+9], eax
mov    eax, [ebp+var_28]
mov    [esi+0Dh], eax
mov    [ebp+LocalVarSizeOfMemoryReserved], 2CFh ; 719 bytes
push   40h ; '@' ; PAGE_READ_WRITE_EXECUTE
push   3000h
lea    eax, [ebp+LocalVarSizeOfMemoryReserved]
push   eax
push   0
lea    eax, [ebp+LocalVarPointerToMemoryReserved]
push   eax
push   [ebp+LocalVarObjectHandle]
call   BlackMatterZwAllocateVirtualMemoryGlobalVar
test   eax, eax
jnz    short _release_memory
push   offset BlackMattePointerToShellcode264ModeGlobalVar
call   BlackMatterPrepareToDecryptCodeFunction

```

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.

```

push    eax
push    1FFFFFFh
lea    eax, [ebp+var_10]
push    eax
call    BlackMatterZwOpenProcessGlobalVar
test    eax, eax
jnz    _check_if_more_processes
lea    eax, [ebp+LocalVarProcessHandle]
push    eax
push    8
push    [ebp+var_10]
call    BlackMatterZwOpenProcessTokenGlobalVar
test    eax, eax
jnz    short_close_handle
mov    [ebp+LocalVarRequestedPrivileges], 1 ; PrivilegeCount
mov    [ebp+var_48], 1 ; PRIVILEGE_SET_ALL_NECESSARY
mov    [ebp+var_44], 7
mov    [ebp+var_40], 0
mov    [ebp+var_3C], 2
mov    [ebp+LocalVarReturnResult], 0
lea    eax, [ebp+LocalVarReturnResult]
push    eax
lea    eax, [ebp+LocalVarRequestedPrivileges]
push    eax
push    [ebp+LocalVarProcessHandle]
call    BlackMatterZwPrivilegeCheckGlobalVar

```

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.


```

_crypt_data:
; CODE XREF: BlackMatterPrepareToSendInfoSettingProtocolsAndP
mov     ebx, [ebp+SecondArgumentSizeOfTheStringToSend]
inc     ebx
add     ebx, 0Fh
and     ebx, 0FFFFFF0h
push   ebx ; size aligned and re calculated
push   [ebp+FirstArgumentPointerToFinalStringToSend]
push   10h
push   offset BlackMatterAESKeyToSendInfoPreparedFromConfig
call   BlackMatterPrepareAESKeyAndCryptDataToSendWithAESFunction

```

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.

```

_prepare_info:
; CODE XREF: BlackMatterPrepareAllStringsAndGetInformationOfTheUserAndMachineAndPrepareStringToSendToC2Function+591j
call   BlackMatterPrepareToMakeStringForUnitsAndPrepareStringWithInfoOfThemToSendFunction
mov     [ebp+LocalVarPointerToUnitsStringPreparedToSend], eax
call   BlackMatterPrepareToCreateThreadToGetUserAndPrepareStringToSendFunction
mov     [ebp+LocalVarPointerToUserStringPreparedToSend], eax
call   BlackMatterGetComputerNameFunction
mov     [ebp+LocalVarPointerToComputerNameStringPreparedToSend], eax
call   BlackMatterGetLanguageOfTheSystemFromTheRegisterUsingSIDOfTheUserFunction
mov     [ebp+LocalVarPointerToLanguageStringPreparedToSend], eax
call   BlackMatterPrepareToGetJoinInformationFunction
mov     [ebp+LocalVarWorkgroupOrDomainStringPreparedToSend], eax
call   BlackMatterGetProductTypeOfWindowsAndPrepareStringToSendFunction
mov     [ebp+LocalVarProductNameStringPreparedToSend], eax
call   BlackMatterGetCPUArchitectureAndPrepareStringFunction
mov     [ebp+LocalVarCPUArchitectureStringPreparedToSend], eax
push   [ebp+LocalVarPointerToUnitsStringPreparedToSend]
call   BlackMatterMCSLENGlobalVar

```

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.

```

cmp     eax, 0EB9F5C34h ; https
jnz    short_check_if_http
mov     ebx, 443 ; port https
mov     edi, 800000h
jmp     short_string_copy
; -----
_check_if_http:
; CODE XREF: BlackMatterPrepareToSendInfoSettingProtocolsAndP
cmp     eax, 0EB869000h ; http
jnz    short_check_size_string
mov     ebx, 80 ; port http
mov     edi, 400000h
jmp     short_string_copy
; -----

```

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], 0E9B30A21h
mov     dword ptr [eax+4], 3307CEC5h
mov     dword ptr [eax+8], 0E206C440h
mov     dword ptr [eax+0Ch], 00DFB87A2h ; CLSID_MbemAdministrativeLocator
mov     ecx, 4

_loop_decrypt_data:                                ; CODE XREF: BlackMatterDestroyShadowVolumesUsingCOMFunction+624j
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+914j
xor     dword ptr [eax], 22065FEDh
add     eax, 4
dec     ecx
jnz    short _loop_decrypt_data_
lea     eax, [ebp+LocalVarI@EM_ContextString]
mov     dword ptr [eax], 45403975h
mov     dword ptr [eax+4], 3306817Fh
mov     dword ptr [eax+8], 0E2062E40h
mov     dword ptr [eax+0Ch], 00DFB87A2h ; CLSID_MbemContext
mov     ecx, 4

_loop_decrypt_data_ :                             ; CODE XREF: BlackMatterDestroyShadowVolumesUsingCOMFunction+C04j
```

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.

```
_loop_check_service: ; CODE XREF: BlackMatterEnumerateAllServicesInSystemAndCheckEa
    push    dword ptr [edi]
    call    BlackMatterCheckServiceNameAgainstServicesTargetListFunction
    test    eax, eax
    jz     short _inc_pointer
    push    10020h
    push    dword ptr [edi]
    push    [ebp+LocalVarSCManagerHandle]
    call    BlackMatterOpenServiceWGlobalVar
    mov     [ebp+LocalVarServiceHandle], eax
    cmp     [ebp+LocalVarServiceHandle], 0
    jz     short _inc_pointer
    push    1Ch
    push    0
    lea    eax, [ebp+LocalVarPreviousState]
    push    eax
    call    BlackMatterMemsetGlobalVar
    add    esp, 0Ch
    lea    eax, [ebp+LocalVarPreviousState]
    push    eax
    push    1 ; STOP_SERVICE
    push    [ebp+LocalVarServiceHandle]
    call    BlackMatterControlServiceGlobalVar
    push    [ebp+LocalVarServiceHandle]
    call    BlackMatterDeleteServiceGlobalVar
    push    [ebp+LocalVarServiceHandle]
    call    BlackMatterCloseServiceHandleGlobalVar
```

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”.

```
change_process_priority: ; CODE XREF: sub_40E677+AB1j
    call    BlackMatterChangeTheOwnProcessPriorityAndClassPriorityFunction
    mov     ecx, 0A2065FECh
    xor     ecx, 22065FEDh ; 0x00000001 -> ES_CONTINUOUS | ES_SYSTEM_REQUIRED
    lea    eax, [ebp+LocalVarPreviousThreadState]
    push    eax
    push    ecx
    call    BlackMatterZwSetThreadExecutionStateGlobalVar ; disable the system idle timer to avoid that the system suspend or hibernate
```

FIGURE 23. Preventing the machine being suspended or hibernated


```

_make_random_string:                                ; CODE XREF: BlackMatterSetPersistenceInRunOnceRegistryEntryWithWallSwitchFunction+61fj
call        BlackMatterMakeStringWithWildcardAnd9RandomCharsFunction
mov        [ebp+LocalVarPointerToRandomStringWithWildCardAnd9Characters], eax
cmp        [ebp+LocalVarPointerToRandomStringWithWildCardAnd9Characters], 0
jnz        short _get_command_line
jmp        short _check_ebx
; -----
_get_command_line:                                  ; CODE XREF: BlackMatterSetPersistenceInRunOnceRegistryEntryWithWallSwitchFunction+74fj
call        BlackMatterGetCommandLineFromPEBFunction
push        6
push        eax
call        BlackMatterCreateMemoryAndCopyStringIntoNewBufferFunction
mov        ebx, eax
test       ebx, ebx
jnz        short _get_size_string
jmp        short _check_ebx
; -----
_get_size_string:                                   ; CODE XREF: BlackMatterSetPersistenceInRunOnceRegistryEntryWithWallSwitchFunction+89fj
push        ebx
call        BlackMatterWCSENGlobalVar
add        esp, 4
lea        esi, [ebx+eax*2]
mov        dword ptr [esi], 22285FCDh
mov        dword ptr [esi+4], 22675F9Ah
mov        dword ptr [esi+8], 226A5F81h ; -wall
xor        dword ptr [esi], XOR_MAGIC_VALUE
xor        dword ptr [esi+4], XOR_MAGIC_VALUE
xor        dword ptr [esi+8], XOR_MAGIC_VALUE
push        ebx
call        BlackMatterWCSENGlobalVar
add        esp, 4
lea        eax, ds:2[eax*2]
push        eax
push        ebx
push        1
push        0
push        [ebp+LocalVarPointerToRandomStringWithWildCardAnd9Characters]
push        [ebp+LocalVarHandle]
call        BlackMatterRegSetValueExWGlobalVar

```

FIGURE 26. Setting the persistence 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.


```

_get_special_path:                                ; CODE XREF: BlackMatterPrepareNewWallPaperAndC
    push    0
    push    23h ; '#'                            ; AppData
    push    [ebp+LocalVarNewMemoryReserved]
    push    0
    call    BlackMatterSHGetSpecialFolderPathGlobalVar
    push    [ebp+LocalVarNewMemoryReserved]
    call    BlackMatterAddBackslashCharacterFunction
    mov     ebx, [ebp+LocalVarNewMemoryReserved]
    mov     ecx, BlackMatterNewExtensionForCryptedFilesGlobalVar
    lea    eax, [ecx+2]
    push    eax
    push    ebx
    call    BlackMatterWCSCATGlobalVar
    add    esp, 8
    push    ebx
    call    BlackMatterWCSENGlobalVar
    add    esp, 4
    lea    eax, [ebx+eax*2]
    mov     dword ptr [eax], 22645FC3h
    mov     dword ptr [eax+4], 22765F80h
    mov     dword ptr [eax+8], XOR_MAGIC_VALUE ; .bmp
    mov     ecx, 3

```

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 <code>icacls "<DriveLetter>:*" /grant Everyone: F /T /C /Q</code> 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: <code>\\<IP>\<drive letter>\$</code>

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.