REvil과 DarkSide랜섬웨어의 치명적인 강점만 통합한

BlackMatter 랜섬웨어 분석리포트

2021.08

SOMANSA

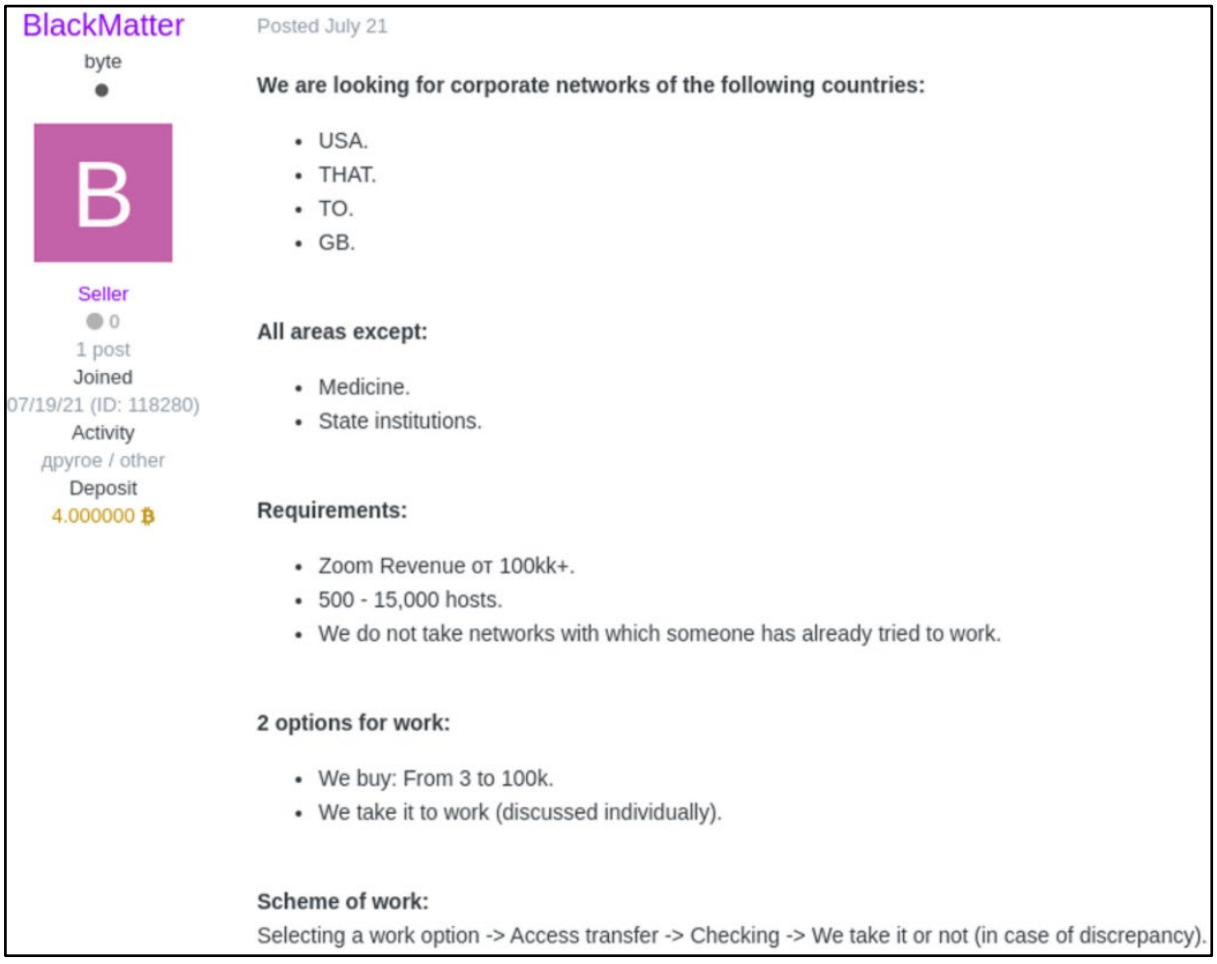
목 차

1. 개 요

1.1 배 경

2021 년 7 월, REvil 과 DarkSide 랜섬웨어는 더 이상의 활동을 하지 않는다는 소식을 발표했다. 하지만 한달도 채 되지않아, 이들 랜섬웨어와 대부분의 코드를 공유하는 새로운 랜섬웨어가 활동을 시작했다. 해당 랜섬웨어는 자신이 REvil 과 DarkSide 의 최고의 기능들을 통합하여 제작되었다고 홍보하였다.

REvil 과 DarkSide 의 최고의 기능들을 통합하여 제작되었다고 홍보한 랜섬웨어는 곧, BlackMatter 라는 이름을 사용해 활동을 시작하였다. BlackMatter 랜섬웨어는 RaaS (Ransomware As A Service) 형태의 랜섬웨어로서 공격을 원하는 사람에게 공격 성공 후, 일정한 금액을 지불 받기로 합의한 후 랜섬웨어를 제공하는 형태이다. 실제로 Exploit 포럼에 BlackMatter 랜섬웨어 광고가 올라왔으며, 3000 달러에서 100,000 달러의 지불 금액을 선택할 수 있다고 홍보했다.



BlackMatter

byte

B

Seller

0

1 post

Joined

07/19/21 (ID: 118280)

Activity

другое / other

Deposit

4.000000 ₿

Posted July 21

We are looking for corporate networks of the following countries:

- USA.
- THAT.
- TO.
- GB.

All areas except:

- Medicine.
- State institutions.

Requirements:

- Zoom Revenue от 100kk+.
- 500 - 15,000 hosts.
- We do not take networks with which someone has already tried to work.

2 options for work:

- We buy: From 3 to 100k.
- We take it to work (discussed individually).

Scheme of work:

Selecting a work option -> Access transfer -> Checking -> We take it or not (in case of discrepancy).

[그림 1] BlackMatter 홍보 게시글

위처럼 공개적으로 홍보가 되었으며, 누구나 돈만 지불할 수 있으면 랜섬웨어를 통해 쉽게 대상 기관 및 기업을 공격할 수 있다. 소만사는 이를 심각한 위협으로 받아드리고, 신속히 BlackMatter 랜섬웨어 샘플을 확보한 후 이를 상세히 분석하였다. 본 보고서에는 BlackMatter 랜섬웨어를 사전에 예방 및 차단할 수 있도록 상세한 분석 내용과 Privacy-i 의 탐지 결과를 서술하였다.

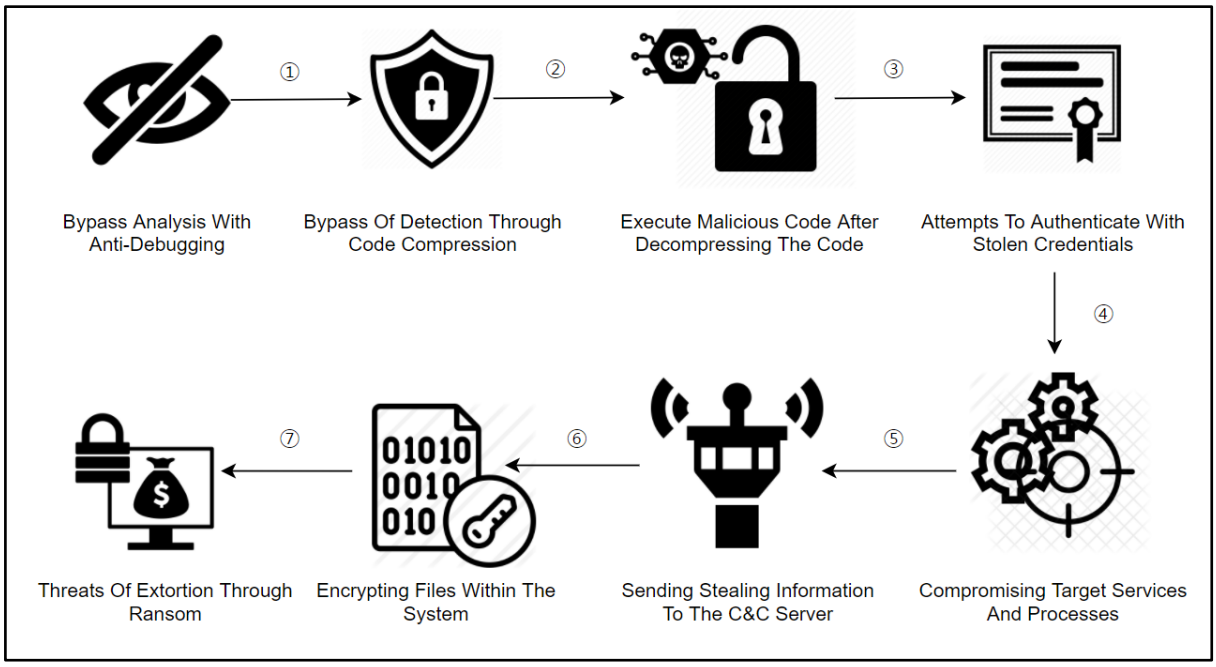
1.2 파일 정보

Name	**[Random].exe**
Type	Windows Portable Executable File
Behavior	Ransomware
SHA-256	22d7d67c3af10b1a37f277ebabe2d1eb4fd25afbd6437d4377400e148bcc08d6
Description	BlackMatter Ransomware

[파일 1] BlackMatter Ransomware EXE File

2. 분 석

BlackMatter 랜섬웨어는 두 가지 Anti-Debugging 기법을 사용하여 악성코드를 숨겼다. 이를 통해 분석 등을 우회하였으며, 주요한 정보를 압축하여 보안 제품의 탐지 또한 우회하였다. 이후 BlackMatter 는 도용한 인증 정보를 통해 시스템에 인증을 시도한다. 이와 함께 특정 프로세스 및 서비스를 종료하고, 감염 PC 에 대한 정보를 C&C 서버에 전송한다. 마지막으로 시스템 내 파일들을 암호화 한 후, 암호화 한 파일들을 빌미로 가상화폐 지불을 요구하는 협박을 수행한다.



[그림 2] BlackMatter 랜섬웨어 공격 흐름

[1 단계: ①~②]
①. 두 가지 Anti-Debugging 기법을 이용하여 분석을 우회한다.
②. 코드 압축을 사용하여 보안 제품의 탐지를 우회한다. 이후 압축이 해제되고 악성코드가 실행된다.

[2 단계: ③~④]
③~④. 도용한 인증 정보를 통한 인증 시도와 대상 프로세스 및 서비스에 대한 종료를 수행한다.

[3 단계: ⑤]
⑤. 감염된 PC 정보 및 탈취한 정보를 C&C 서버에 전송한다.

[4 단계: ⑥~⑦]
⑥~⑦. 시스템 내 파일들을 암호화하며, 암호화한 파일들을 빌미로 가상화폐 지불 협박을 시도한다.

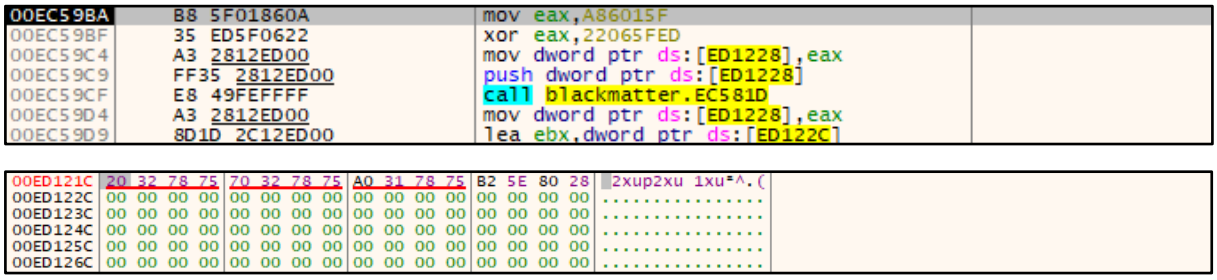
본 보고서는 위 단계의 주요 행위를 아래와 같이 각 단계 별로 상세 분석한다. [2.1.1 ~ 2.1.2] 세 가지의 Anti-Debugging 기법을 통한 분석 우회 [2.1.3 ~ 2.1.12] 권한 탈취 및 상승과 BlackMatter 암호화 수행 데이터 압축 해제 [2.1.13 ~ 2.1.19] 도용 정보를 통한 인증 시도 및 주요 프로세스와 서비스 종료 [2.1.20 ~ 2.1.23] 공유 위반 프로세스 종료 시도와 C&C 연결 및 파일 암호화

[표 1] BlackMatter 랜섬웨어 분석 절차

2.1 BlackMatter Analysis

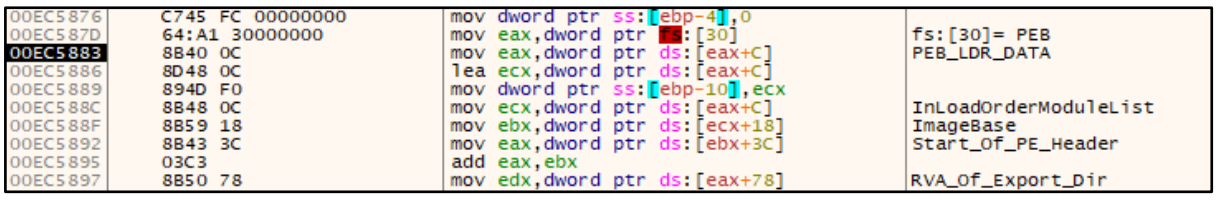
2.1.1 API 동적 호출

BlackMatter 는 Anti-Virus 제품의 탐지와 악성코드 분석가의 분석을 우회하기 위해 필요한 API 주소를 동적으로 가져온다. 이를 통해 PE 헤더의 Import Table 을 통한 탐지 및 분석을 우회할 수 있다. 이는 필요한 DLL 및 API 주소에 대해 이들을 식별할 수 있는 고유한 값을 통한 연산을 하여 수행된다.



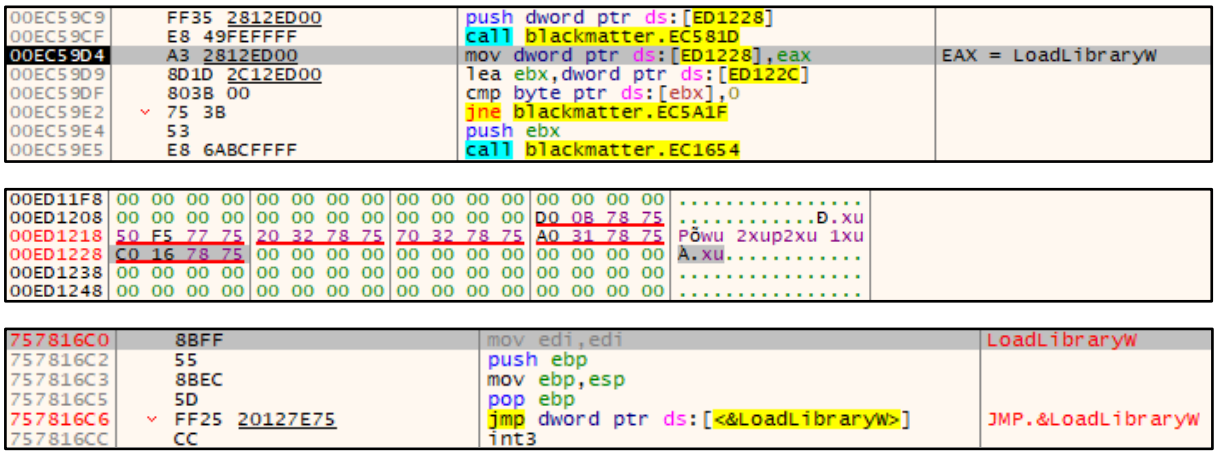
[그림 3] 해시와 XOR 연산을 통한 대상 API 값 획득

대상이 되는 API 에 대한 값을 0x0A86015F 으로 지정한 후 0x22065FED 키를 이용하여 XOR 연산을 수행한다. 이후 연산의 결과는 메모리 0x00ED1228 에 저장된 후 0x00EC581D 서브루틴으로 전달된다.



[그림 4] PEB 를 통한 DLL 주소 동적 획득

DLL 및 API 를 동적으로 가져오기 위해, 먼저 PEB(Process Environment Block)을 주소를 획득한다. PEB(Process Environment Block)는 프로세스의 정보를 담고 있는 구조체로서 프로세스의 다양한 정보를 포함하고 있다. PEB(Process Environment Block) 주소의 0xC 오프셋으로 PEB_LDR_DATA 구조체의 주소를 획득하고 InLoadOrderModuleList 를 구하여 로드된 모듈을 동적으로 구한다.



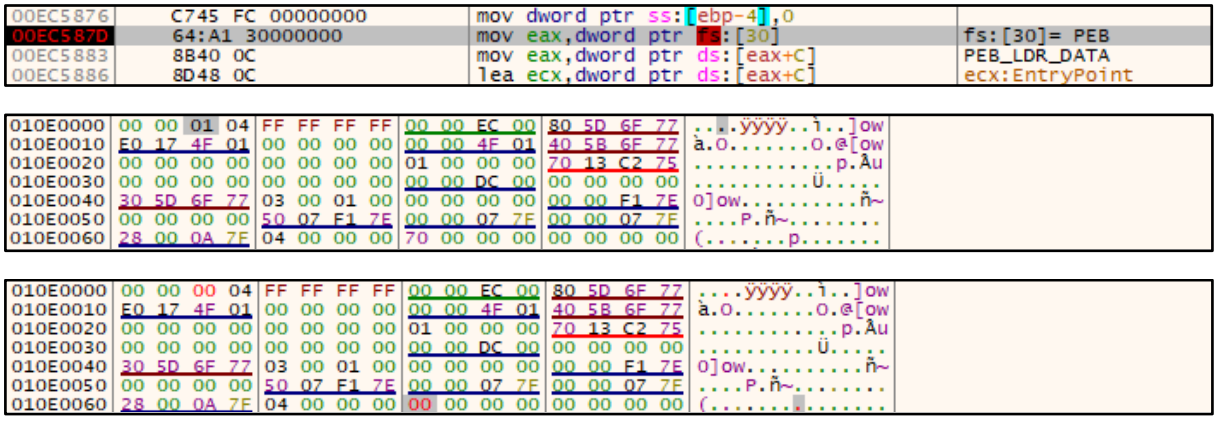
[그림 5] 동적 API 주소 획득

동적으로 모듈을 구하면 모듈의 PE_Header 를 찾은 후 Export Table 을 참고하여 API 주소를 구한다. API 주소를 획득한 후 메모리에 적재한다.

2.1.2 Anti-Debugging

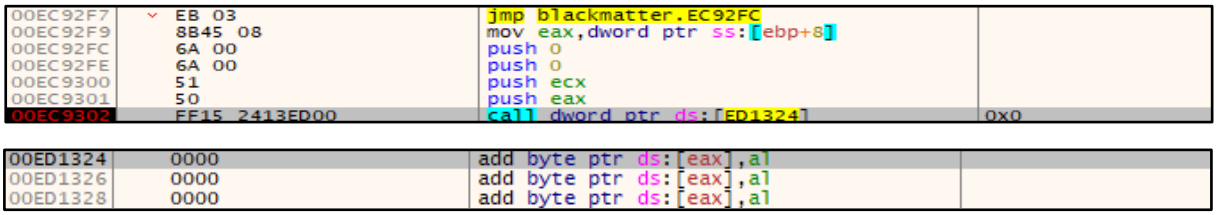
2.1.2.1 Anti-Debugging (BeingDebugged, NtGlobalFlag)

BlackMatter 는 디버깅을 통한 탐지 및 분석을 피하기 위해 Anti-Debugging 기법을 사용한다. 해당 Anti-Debugging 기법을 우회하지 못하면 BlackMatter 의 실제 악성코드를 확인할 수 없도록 설계되어 있다.



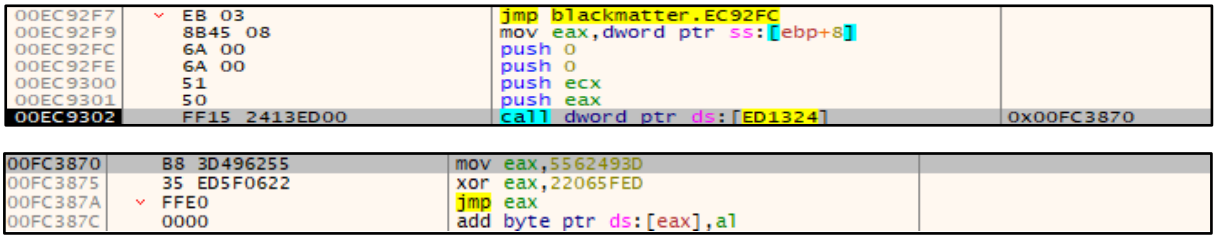
[그림 6] Anti-Debugging 기법 (BeingDebugged, NtGlobalFlag)

PEB 주소를 획득한 후 특정 오프셋의 값을 확인하는데, 오프셋 0x2 에 위치한 BeingDebugged 와 오프셋 0x68 에 위치한 NtGlobalFlag 이다. BeingDebugged 와 NtGlobalFlag 는 해당 프로세스가 디버깅 중인지 확인할 수 있는 값인데, 디버깅 중일 경우 BeingDebugged 는 0x1 로 설정되며 NtGlobalFlag 는 0x70 으로 설정된다. BlackMatter 는 이 값들을 확인하여 탐지 및 분석을 회피한다.



[그림 7] Anti-Debugging 기법에 따른 코드 패치 (이상 코드 수행)

BlackMatter 는 Anti-Debugging 기법에 의해 해당 프로세스가 디버깅 중인 것으로 확인되면 위와 같이 비정상적인 코드의 주소를 호출하도록 설계되어 있다.

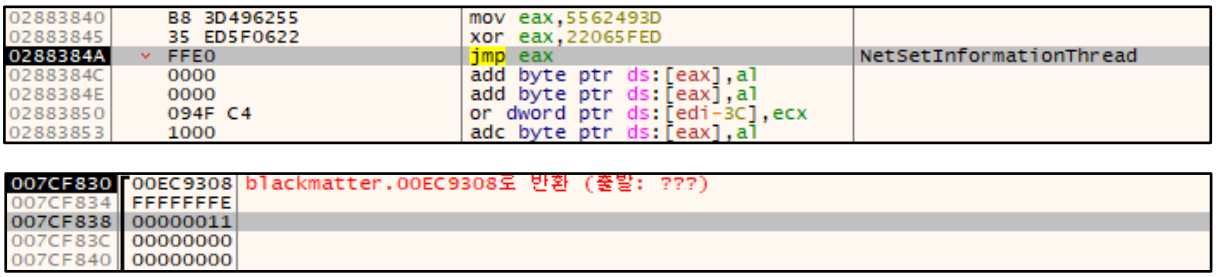


[그림 8] Anti-Debugging 기법에 따른 코드 패치 (정상 코드 수행)

Anti-Debugging 을 우회하고 프로세스의 실행을 진행하면 위와 같이 정상적인 코드의 주소를 호출한다.

2.1.2.2 Anti-Debugging (ThreadHideFromDebugger)

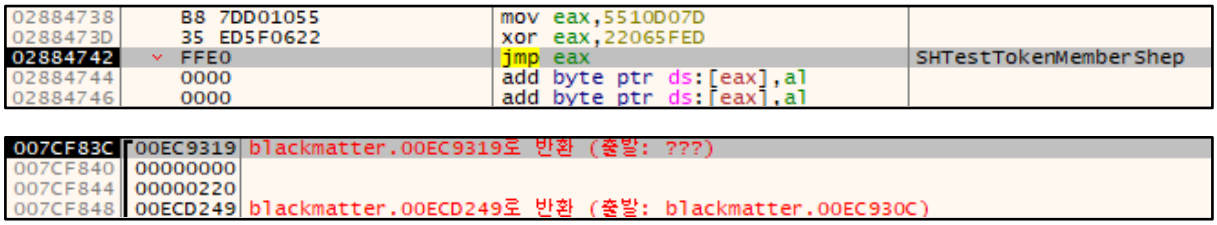
BlackMatter 는 NtSetInformationThread API 호출을 통해 Anti-Debugging 기법을 사용한다. 해당 API 는 호출 시 두번 째 인자에 0x11 (ThreadHideFromDebugger)값을 넣어 호출하면 현재 부착중인 디버거가 해제된다. 이를 통해 악성코드 분석가의 분석을 회피한다.



[그림 8] Anti-Debugging 기법 (ThreadHideFromDebugger)

XOR 연산을 통해 대상 API 의 주소를 구하는데, NtSetInformationThread API 의 주소를 구한다. 이 때, 두 번째 인자로 0x11 (ThreadHideFromDebugger)를 넣어 호출한다. 이를 통해 현재 연결된 디버거와의 연결을 해제한다.

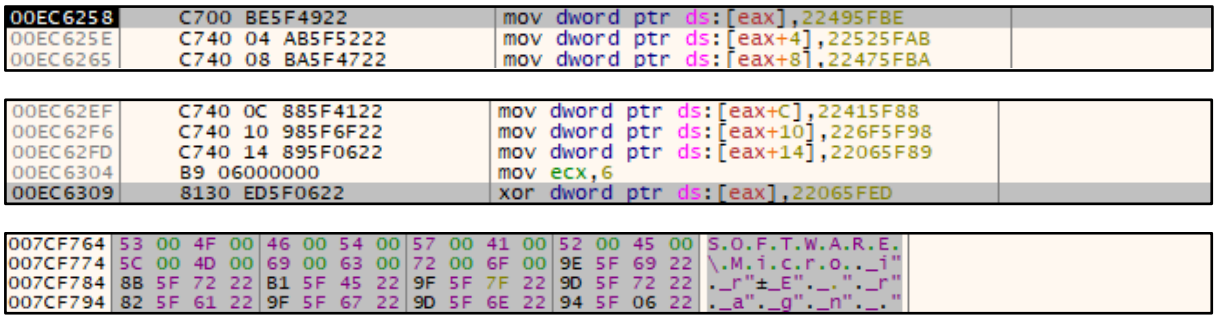
2.1.3 관리자 권한 확인



[그림 9] 관리자 그룹 확인

SHTestTokenMemberShip API 를 Administrators 를 의미하는 RID 인 0x220 을 인자로 호출한다. 이를 통해 BlackMatter 프로세스가 악의적인 행위 수행에 충분한 권한을 갖는 관리자 그룹에 속하는지 확인한다.

2.1.4 XOR 연산을 통한 문자열 복호화

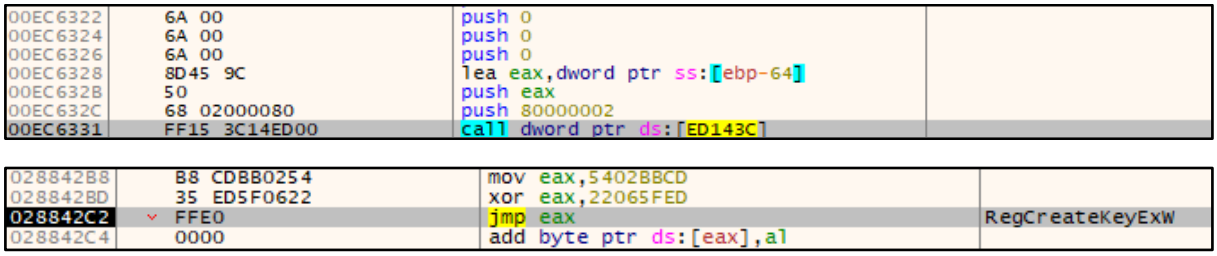


[그림 10] XOR 연산을 통한 문자열 복호화

BlackMatter 는 앞으로 수행 될 악의적인 행위에 사용되는 문자열을 XOR 연산을 통해 복호화한다. 위 사진을 보면 특정한 문자열들을 0x22065FED 키 값과 XOR 연산을 수행하는 것을 확인할 수 있다. 이 후 복호화된 문자열은 악의적인 행위 수행에 사용된다.

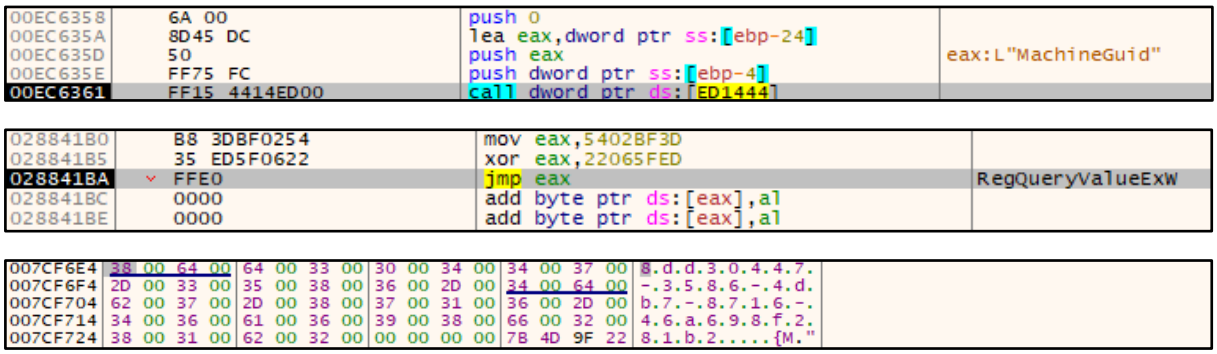
2.1.5 MachineGuid 획득 및 암호화 확장자 생성

BlackMatter 는 HKEY_LOCAL_MACHINE₩SOFTWARE₩Microsoft₩Cryptography 경로의 MachineGuid 값을 확인한다. 해당 값은 PC 마다 고유한 Hardware ID 로서 이를 이용하여 감염 PC 만의 고유한 시그니처를 생성하거나 이를 암호화하여 C&C 서버로 전송한다. BlackMatter 또한 이를 획득하여 감염 PC 의 고유한 시그니처를 생성한다.



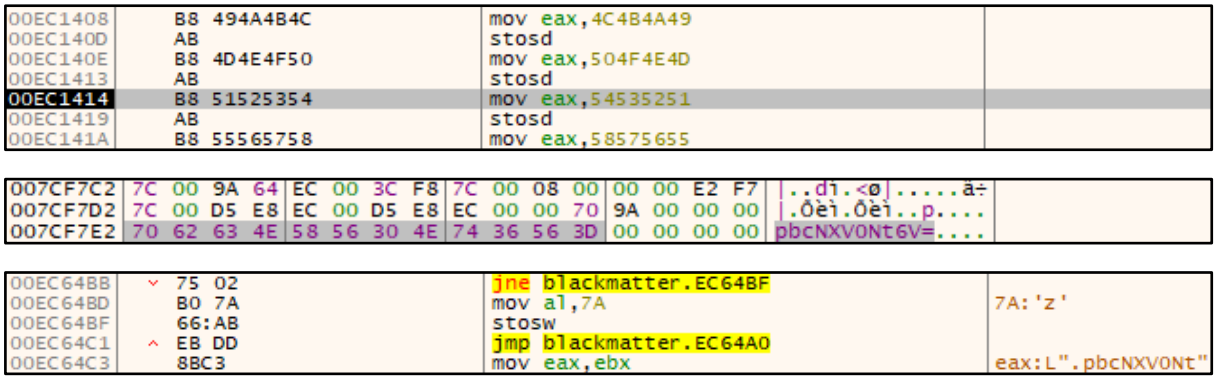
[그림 11] MachineGuid 경로 확인

RegCreateKeyExW API 를 호출하여 HKEY_LOCAL_MACHINE₩SOFTWARE₩Microsoft₩Cryptography 경로의 존재 유무를 확인한다.



[그림 12] MachineGuid 획득

RegQueryValueExW API 를 호출하여 MachineGuid 값을 확인한다. 이에 대한 결과로 0x007CF6E4 의 메모리 주소에는 MachineGuid 값이 적재된 것을 확인할 수 있다. MachineGuid 값의 의미는 Hardware ID 로서, 고유한 값을 갖는다. 이를 이용하여 감염 PC 의 고유한 시그니처를 생성한다.

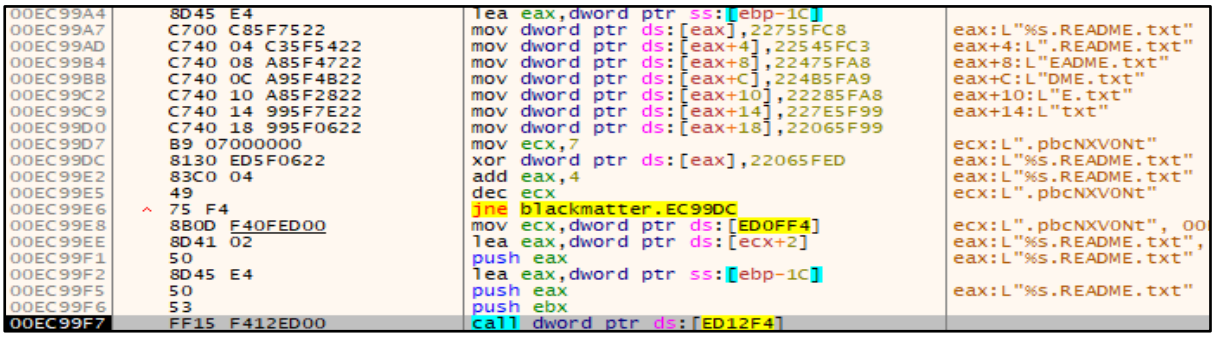


[그림 13] MachineGuid 를 통한 암호화 확장자 생성

획득한 MachineGuid 는 연산을 거쳐 암호화되어 메모리에 적재되며, 암호화 확장자로 사용된다.

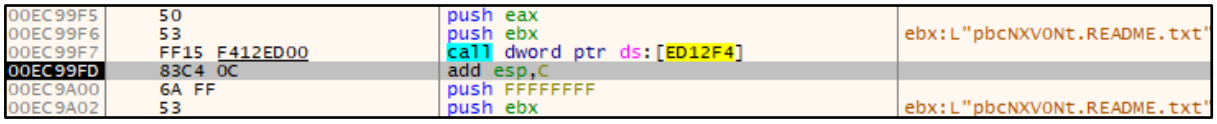
2.1.6 BlackMatter 랜섬 노트 이름 생성

BlackMatter 는 감염 PC 별 고유한 랜섬 노트를 생성한다. 이전에 생성한 암호화 확장자와 암호화된 문자열을 XOR 복호화하여 랜섬 노트 생성에 이용한다.



[그림 14] 랜섬 노트 이름 생성을 위한 암호화 확장자

암호화된 문자열을 XOR 복호화 하는데, 이 때 사용되는 키는 0x22065FED 이다. 이를 통해 README 문자열을 갖는 랜섬 노트 이름이 일부 생성된다.



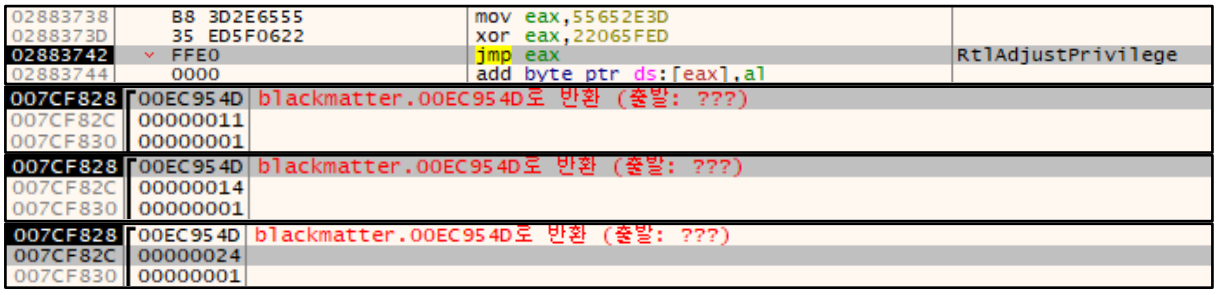
[그림 15] 랜섬 노트 이름 생성

이전에 생성한 암호화 확장자와 README 문자열을 갖는 랜섬 노트 이름을 결합하여 감염 PC 고유의 랜섬 노트 이름을 생성한다.

2.1.7 권한 상승 시도 및 확인

2.1.7.1 권한 상승 시도

BlackMatter 는 RtlAdjustPrivilege API 를 호출하여 권한 상승을 시도한다. 단순히 문서화된 권한이 아닌 비문서화된 권한까지 획득하려 시도한다.



[그림 16] 권한 상승

RtlAdjustToken API 는 권한을 상승시키는 기능을 갖고 있다. 이를 사용하여 BlackMatter 는 자신의 권한을 상승시킨다. 상승시키는 권한은 아래의 표와 같다.

0x5	SeIncreaseQuotaPrivilege
0x8	SeSecurityPrivilege
0x9	SeTakeOwnershipPrivilege
0xB	SeSystemProfilePrivilege
0xD	SeProfileSingleProcessPrivilege
0xE	SeCreateGlobalPrivilege
0x11	SeBackupPrivilege
0x12	SeRestorePrivilege
0x13	SeShutdownPrivilege
0x14	SeDebugPrivilege
0x1C	SeManageVolumePrivilege
0x1D	SeImpersonatePrivilege
0x21	SeIncreaseWorkingSetPrivilege
0x24	SeDelegateSessionUserImpersonatePrivilege (Undocumented)

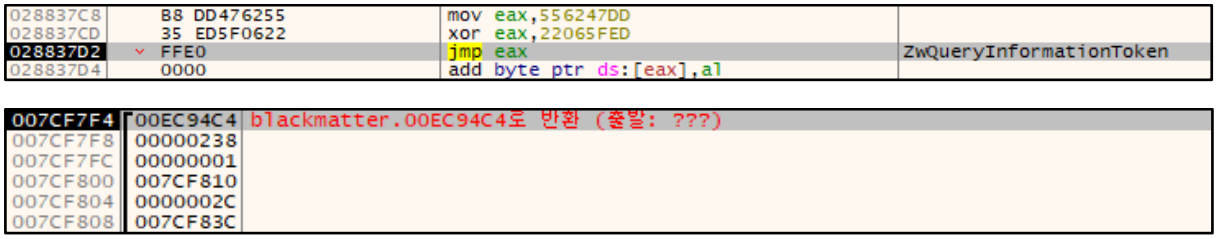
[표 2] BlackMatter 의 상승 대상 권한

2.1.7.2 상승된 권한 확인

BlackMatter 는 권한 상승 시도 후 현재 프로세스의 권한을 확인한다.

```
028836C0    B8 4D776255     mov eax,5562774D
028836C5    35 ED5F0622     xor eax,22065FED
028836CA    FFE0            jmp eax                 NtOpenProcessToken
028836CC    0000            add byte ptr ds:[eax],al
028836CE    0000            add byte ptr ds:[eax],al
```

```
007CF7FC  00EC94AB  blackmatter.00EC94AB로 반환 (출발: ???)
007CF800  FFFFFFFF
007CF804  00000008
007CF808  007CF840
007CF80C  009A7000
007CF810  00000001
```

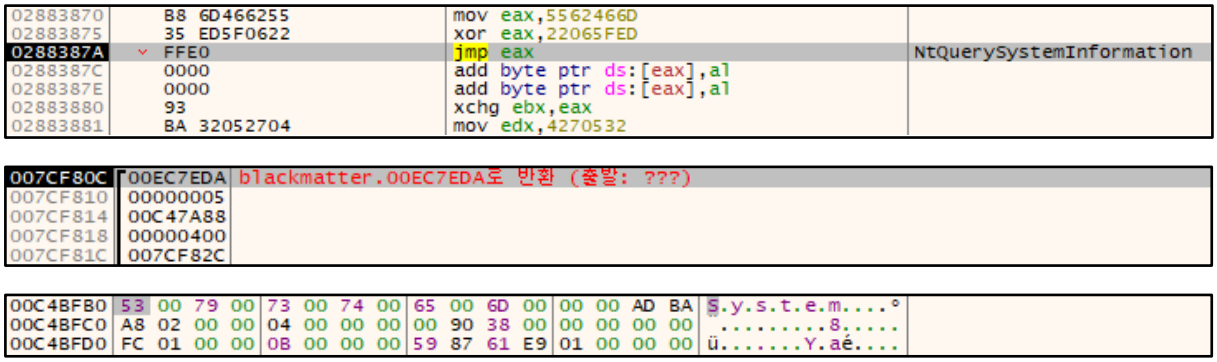


[그림 17] 현재 프로세스의 권한 확인

NtOpenProcessToken API 를 0x8 (PROCESS_VM_OPERATION) 인자, ZwInformationToken API 를 0x1 (TokenUser) 인자로 호출하여 현재 프로세스의 권한을 확인한다.

2.1.8 프로세스 열거

BlackMatter 는 향 후 대상 프로세스의 토큰을 복제하기 위해 프로세스를 열거한다.

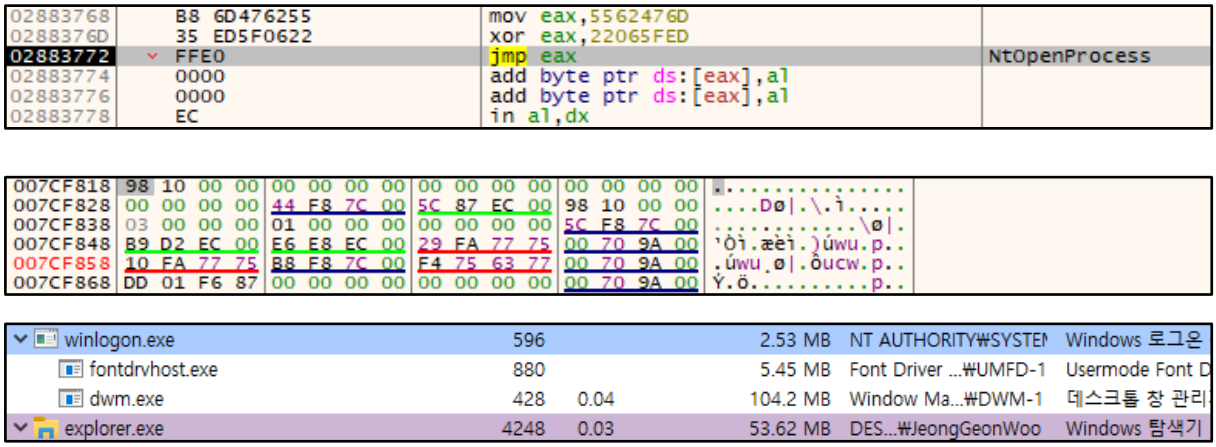


[그림 18] 토큰 복제를 위한 프로세스 열거

NtQuerySystemInformation API 를 0x5 (SystemProcessInformation) 인자로 호출하여 현재 실행중인 프로세스의 목록을 획득한다.

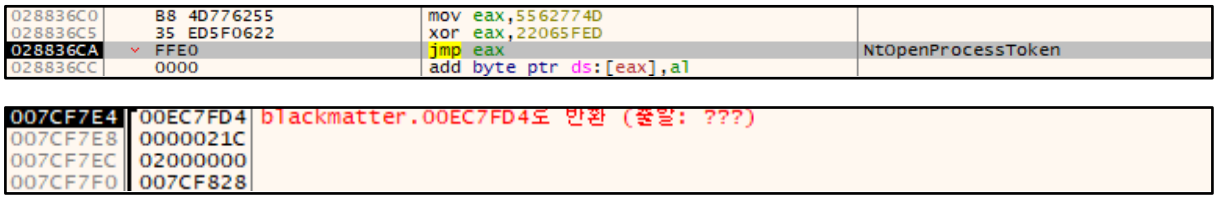
2.1.9 토큰 복제

프로세스 열거를 통해 획득한 대상 프로세스 정보를 이용하여 대상 프로세스의 토큰을 복제한다.



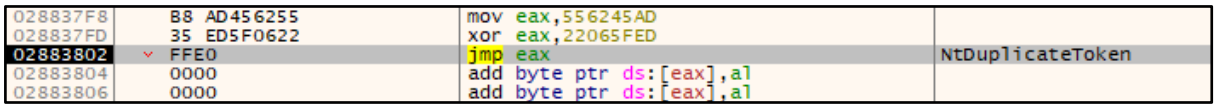
[그림 19] 대상 프로세스 토큰 복제를 위한 핸들 획득

NtOpenProcess API 를 호출하여 explorer.exe 프로세스의 핸들을 획득한다. 이를 위해 메모리에 PID 가 인자로 0x1098 (4248) 이 들어있는것을 확인할 수 있다.



[그림 20] 대상 프로세스의 토큰 핸들 획득

NtOpenProcessToken API 를 호출하는데, 인자로 0x02000000 (MAXIMUM_ALLOWED)가 들어간다. 이는 모든 권한을 의미하는 것으로, explorer.exe 의 모든 권한을 가진 토큰을 획득할 수 있다.



[그림 21] 대상 프로세스의 토큰 복제

토큰을 복제하는 과정에 있어, 마지막으로 NtDuplicateToken API 를 호출한다. 이는 토큰을 복제하는 기능을 가진 API 로서 이를 통해 BlackMatter 는 explorer.exe 의 권한과 같은 토큰을 얻게된다. 획득한 권한은 향 후 악성 행위에 사용된다.

2.1.10 리소스 압축 해제를 위한 키 생성

BlackMatter 는 리소스 영역에 압축된 주요 데이터를 갖고 있다. 리소스 영역 내 해당 데이터는 aPLib 압축 라이브러리를 사용하여 압축되었으며, 내부의 키 값을 이용해 압축 해제된다.

1	.rdata	000003d0	0000f000	00000400	0000e000	00000000	00000000
2	.data	000015ec	00010000	00001000	0000e400	00000000	00000000
3	.rsrc	00000da7	00012000	00000e00	0000f400	00000000	00000000
4	.reloc	000008fc	00013000	00000a00	00010200	00000000	00000000

```
0000F400  EA A1 CA FF 9F 0D 00 00  ED 6F 8D 71 2A E0 E5 C1  . . . . . o . q * . . .
0000F410  AD 37 EF A7 89 5F 95 3E  7A 1D 58 AE A1 F6 37 29  . 7 . . . _ . > z . X . . . 7 )
0000F420  E8 B1 CB 12 1F 7B 98 DE  64 BF E3 C2 EF F2 F3 39  . . . . . { . . d . . . . . 9
0000F430  65 6A 2A B9 B0 A6 5C 26  F1 E2 8A 83 2B E0 EB B0  e j * . . . \ & . . . . + . . .
```

[그림 22] 리소스 영역 내 압축된 주요 데이터

위 사진을 보면 리소스 영역 내 압축된 데이터를 확인할 수 있다. 0x0000F400 오프셋의 0xFFCAA1EA 값은 키 값이며, 0x00000D9F 값은 압축된 데이터의 사이즈를 의미한다.

```
00EC171B   8B5D 0C          mov ebx,dword ptr ss:[ebp+C]
00EC171E   8B75 08          mov esi,dword ptr ss:[ebp+8]
00EC1721   A1 0020ED00      mov eax,dword ptr ds:[ED2000]
00EC1726   8945 FC          mov dword ptr ss:[ebp-4],eax
00EC1729   8D45 FC          lea eax,dword ptr ss:[ebp-4]
00EC172C   50               push eax
00EC172D   FF35 0020ED00    push dword ptr ds:[ED2000]      0xFFCAA1EA
00EC1733   E8 31000000      call blackmatter.EC1769
```

```
00ED2000  EA A1 CA FF 9F 0D 00 00  ED 6F 8D 71 2A E0 E5 C1  ê¡Êÿ....ío.q*àåÁ
00ED2010  AD 37 EF A7 89 5F 95 3E  7A 1D 58 AE A1 F6 37 29  .7ï§._.>z.X®¡ö7)
00ED2020  E8 B1 CB 12 1F 7B 98 DE  64 BF E3 C2 EF F2 F3 39  è±Ë..{.Þd¿ãÂïòó9
00ED2030  65 6A 2A B9 B0 A6 5C 26  F1 E2 8A 83 2B E0 EB B0  ej*¹°¦\&ñâ..+àë°
00ED2040  AE A7 3B CC 86 77 0D C1  3A 5D AF 60 F6 B1 F7 B1  ®§;Ì.w.Á:]¯`ö±÷±
00ED2050  14 82 04 DF 1A DB 3D 47  B9 DB 93 EB 8C A7 52 94  ...ß.Û=G¹Û.ë.§R.
```

[그림 23] 압축 해제를 위한 초기 키 값

0xFFCAA1EA 는 키 값으로 0xEC1769 서브루틴에 전달된다. 이는 이후에 압축 해제를 위해 사용된다.

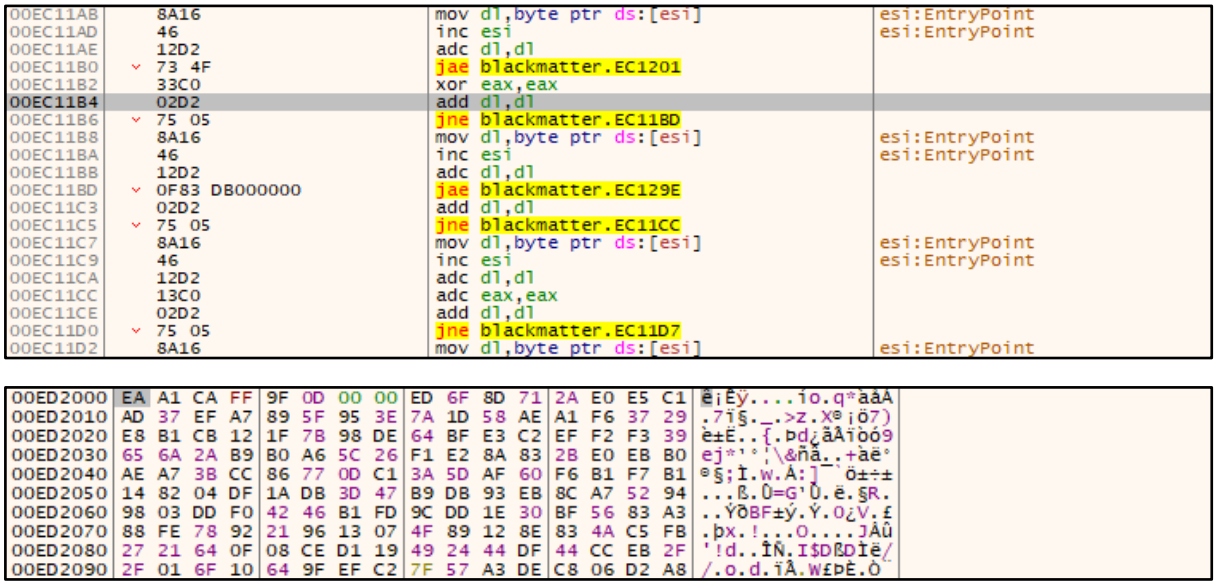
```
00EC176E   8B45 08          mov eax,dword ptr ss:[ebp+8]    EBP+8 = 0xFFCAA1EA
00EC1771   8B5D 0C          mov ebx,dword ptr ss:[ebp+C]    [ebp+C]:EntryPoint
00EC1774   6913 05840808    imul edx,dword ptr ds:[ebx],8088405   IMUL 0xFFCAA1EA, 0x08088405
00EC177A   42               inc edx
00EC177B   8913             mov dword ptr ds:[ebx],edx      EDX = 0xD9C1D193
00EC177D   F7E2             mul edx
00EC177F   8BC2             mov eax,edx                     EAX = 0xD9946C6A
```

[그림 24] 압축 해제를 위한 키 값 생성

전달된 키 값인 0xFFCAA1EA 와 서브루틴 내부에서 0x08088405 로 표기되는 시드는 IMUL 연산 등이 수행되는데, 이러한 연산을 수행하여 압축 해제를 위한 키가 생성된다. 해당 연산은 데이터의 사이즈인 0x0D9F (3487) 회 수행된다.

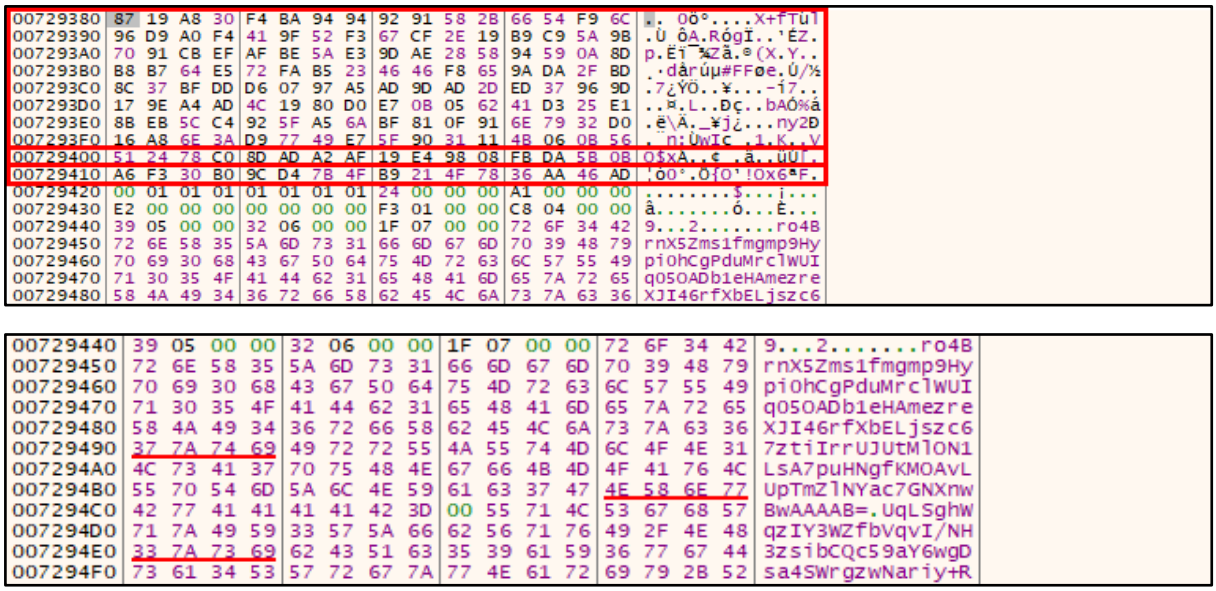
2.1.11 리소스 압축 해제

압축된 데이터는 이전에 생성한 키 값과 aPLib 압축 라이브러리를 사용해 압축 해제된다. 압축 해제된 데이터에는 암호화에 사용되는 공격자의 RSA 공개 키와 C&C 서버와의 통신을 암호화하는 AES 키 및 여러 주요한 데이터가 들어있다.



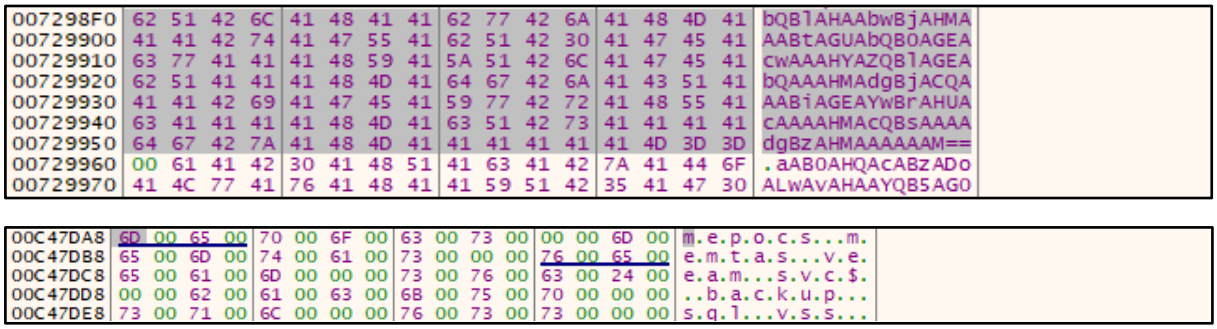
[그림 25] aPLib 압축 라이브러리로 압축된 데이터

위 사진을 보면 aPLib 압축 라이브러리가 구현되어 있으며, 이를 통해 데이터를 압축 해제한다.



[그림 26] 압축 해제된 데이터의 주요 정보

압축 해제된 데이터는 위 사진에 표시된 주요 정보를 갖고 있다. 0x00729380 주소부터 0x007293FF 주소까지는 공격자의 RSA 공개 키 정보가 담겨있다. 0x00729400 에는 Bot_Compay 정보가 담겨 있으며, 이후 0x00729410 주소에는 C&C 서버와의 통신을 암호화하는 AES 키 정보가 담겨있다. 그 아래의 0x00C4AB94 주소에는 종료 대상 서비스 등의 악성 행위에 필요한 주요 정보가 Base64 인코딩 되어 있다.

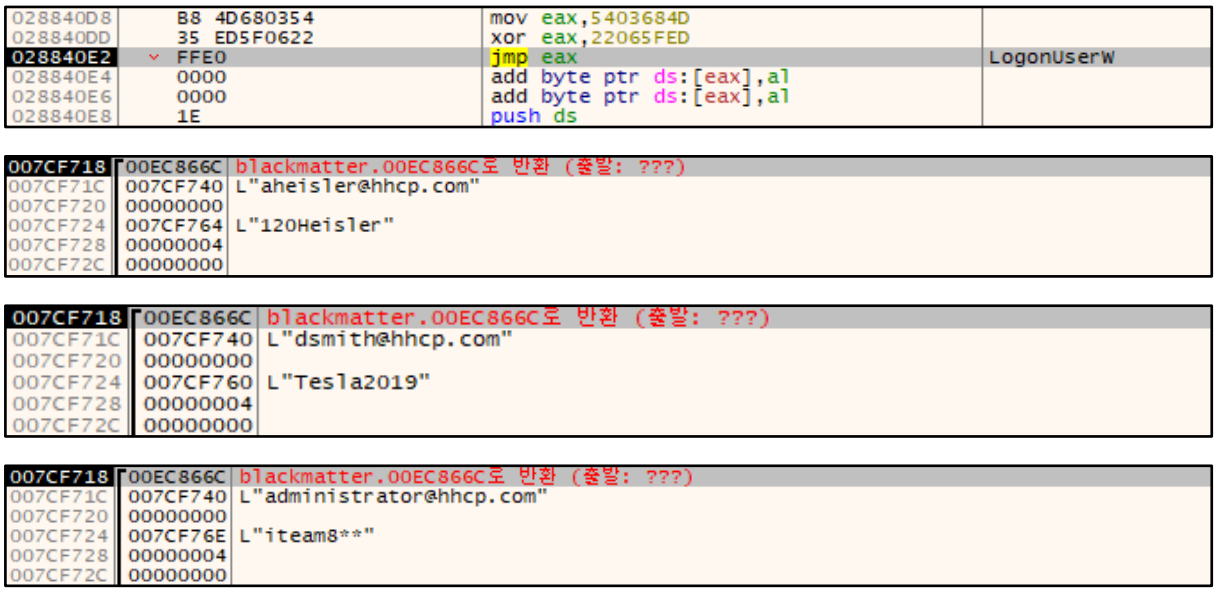


[그림 27] 압축 해제된 데이터 내 종료 대상 서비스 목록

압축 해제된 Base64 로 인코딩된 문자열은 이후 디코딩되어 종료 대상 서비스 등의 문자열을 확인할 수 있다.

2.1.12 자격 증명을 통한 인증 시도

BlackMatter 는 바이너리 내부의 자격 증명을 통한 인증을 시도한다.



[그림 28] 도용한 자격증명을 통한 인증 시도

바이너리 내부에는 압축되어 저장된 자격 증명이 있는데, 이를 압축 해제하여 인증 시도에 사용한다. 이 때 사용되는 API 는 LogonUserW 이며, 인자는 0x4 (LOGON32_LOGON_BATCH)이다. 해당 인자를 사용하면 PC 의 관리 토큰을 획득할 수 있다. 또한 자격 증명 정보는 아래의 표와 같다.

Domain	Password
aheisler@hhcp.com	120Heisler
dsmith@hhcp.com	Tesla2019
administrator@hhcp.com	iteam8**

[표 3] 도용한 자격 증명

2.1.13 실행 인자에 따른 암호화 행위

BlackMatter 는 암호화 수행에 앞서, 인자값을 확인한다. BlackMatter 는 프로세스 실행 시, 인자로 특정 값들을 입력할 수 있는데 이 값들에 따라 암호화 수행 방식이 달라진다.

```
028846F0    B8 8DFE1355     mov eax,5513FE8D
028846F5    35 ED5F0622     xor eax,22065FED
028846FA    FFE0            jmp eax                 CommandLineToArgvW
028846FC    0000            add byte ptr ds:[eax],al
028846FE    0000            add byte ptr ds:[eax],al
02884700    E3 40           jecxz 2884742
```

[그림 29] 입력받은 인자 확인

입력받은 인자를 확인하기 위해 CommandLineToArgvW API 를 호출한다.

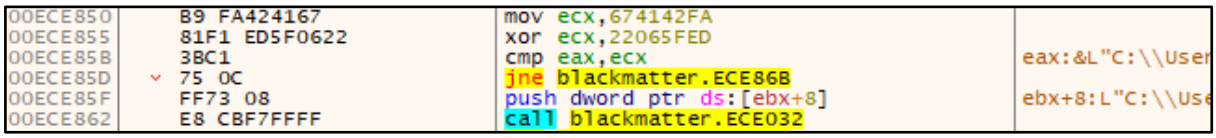
```
00ECE840    837D FC 03      cmp dword ptr ss:[ebp-4],3
00ECE844    75 29           jne blackmatter.ECE86F
00ECE846    6A 00           push 0
00ECE848    FF73 04         push dword ptr ds:[ebx+4]
00ECE84B    E8 6B28FFFF     call blackmatter.EC10BB
00ECE850    B9 FA424167     mov ecx,674142FA
00ECE855    81F1 ED5F0622   xor ecx,22065FED
00ECE85B    3BC1            cmp eax,ecx             eax:&L"C:\\Users\\Jeon
00ECE85D    75 0C           jne blackmatter.ECE86B
00ECE85F    FF73 08         push dword ptr ds:[ebx+8]   ebx+8:L"C:\\Users\\Jeo
00ECE862    E8 CBF7FFFF     call blackmatter.ECE032
00ECE867    EB 5E           jmp blackmatter.ECE8C7
00ECE869    EB 02           jmp blackmatter.ECE86D
00ECE86B    EB 5A           jmp blackmatter.ECE8C7
00ECE86D    EB 58           jmp blackmatter.ECE8C7
00ECE86F    837D FC 02      cmp dword ptr ss:[ebp-4],2
00ECE873    75 4D           jne blackmatter.ECE8C2

008D4798    B8 ADFBF154     mov eax,54F1FBAD
008D479D    35 ED5F0622     xor eax,22065FED
008D47A2    FFE0            jmp eax                 PathIsUNCServerW
008D47A4    0000            add byte ptr ds:[eax],al
```

[그림 30] 특정 인자 입력 시 실행 루틴

주소 0x00ECE840 과 0x00ECE86F 를 확인하면 3 과 2 로 비교 연산을 수행하는 것을 확인할 수 있다. 입력받은 인자의 수가 3 또는 2 자리 인지 확인하는 작업으로, 이와 일치하면 분기를 하여 암호화 수행 방식이 달라진다. BlackMatter 가 입력받을 수 있는 인자의 목록은 아래의 표와 같다.

null (0)	입력받은 인자가 없으면 하드 드라이브 및 네트워크 리소스 등을 암호화한다.
-safe	레지스트리에 자동 실행 등록을 한 후, 안전 모드에서 암호화를 수행한다.
-wall	암호화 메시지가 있는 BMP 이미지 파일을 생성한 후 배경 화면을 변경한다.
-path <path>	네트워크 리소스 등 지정된 개체를 암호화한다.
<path>	입력받은 폴더 및 파일을 암호화한다.

[표 5] BlackMatter 랜섬웨어 실행 인자



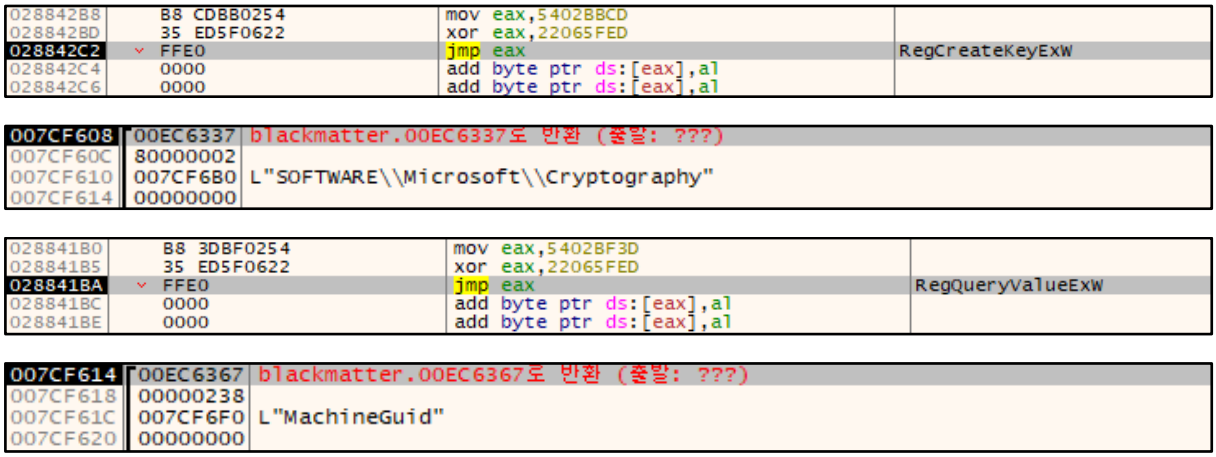
[그림 31] 입력받은 인자 비교 및 검사 루틴

입력받은 인자의 수가 일치하면 이후 일련의 연산을 거쳐, 고정된 특정 값과 비교를 수행한 후 일치할 시 지정된 암호화 방식을 수행한다.

2.1.14 뮤텍스를 통한 중복 실행 방지

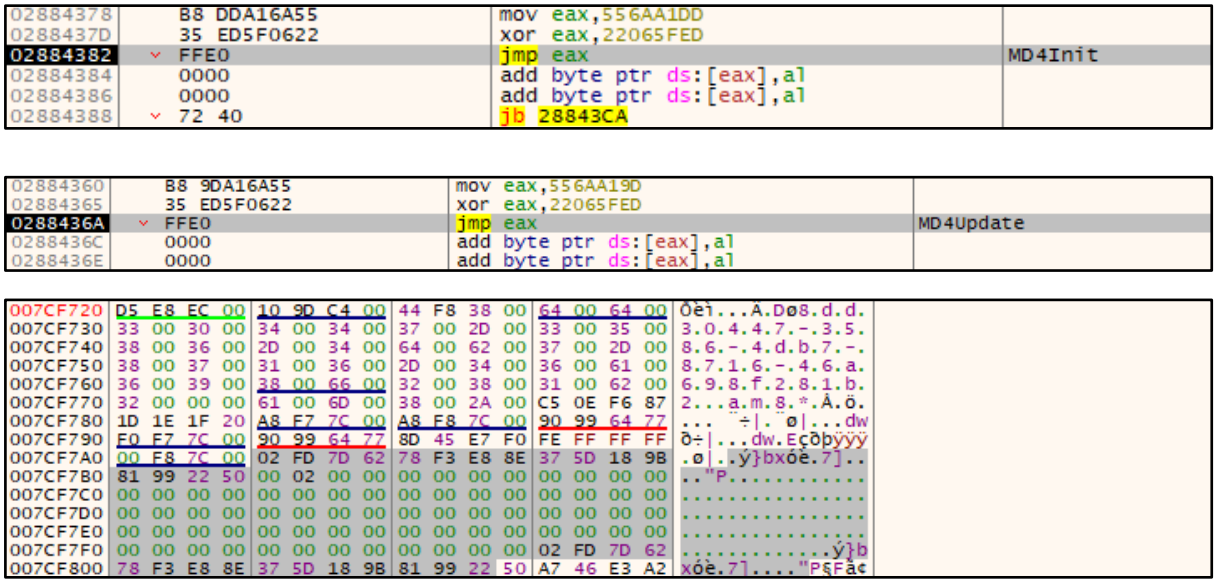
2.1.14.1 뮤텍스 이름 생성

BlackMatter 는 이전에 확인했던 MachineGuid 를 다시 확인 및 획득한다. 획득한 MachineGuid 를 통해 MD4 해시값을 구하는데, 이는 이후 생성된 뮤텍스의 이름이 된다.



[그림 32] 뮤텍스를 위한 MachineGuid 획득

RegCreateKeyExW API 호출을 통해 HKEY_LOCAL_MACHINE₩SOFTWARE₩Microsoft₩Cryptopraphy 경로를 확인하고, RegQueryValueExW API 호출을 통해 MachineGuid 값을 획득한다. 이 때 획득한 MachineGuid 값은 8dd30447-3586-4db7-8716-46a698f281b2 이다.



[그림 33] MachineGuid 와 MD4 를 이용한 해시값 생성

이전에 구한 MachineGuid 를 이용해 MD4Init, MD4Update, MD4Final 로 이어지는 API 들을 호출한다. 이를 통해 MD4 해시값이 생성되는데, 이 값은 주소 0x007CF7A4 부터 시작되는 16 바이트의 값이다. 이를 표기하면 0x627DFD028EE8F3789B185D3750229981 이다. 이 값은 이후 뮤텍스 확인 및 생성에 사용된다.

2.1.14.2 뮤텍스 이름을 통한 중복 실행 방지

이전에 생성한 MD4 해시값으로 뮤텍스를 생성한다. 이를 이용하여 동일한 뮤텍스를 가지고 있는 프로세스가 실행중인지 확인하여, 중복 실행 유무를 판단한다. 만약 중복 실행 중일 경우, 실행중인 현재 프로세스를 종료한다.

```
00EC75D2    8130 ED5F0622       xor dword ptr ds:[eax],22065FED
00EC75D8    83C0 04             add eax,4
00EC75DB    49                  dec ecx
00EC75DC    75 F4               jne blackmatter.EC75D2
00EC75DE    8D75 EC             lea esi,dword ptr ss:[ebp-14]
00EC75E1    6A 50               push 50
00EC75E3    E8 04EAFFFF         call blackmatter.EC5FEC
```

```
00C4D690  47 00 6C 00 6F 00 62 00 61 00 6C 00 5C 00 36 00  G.l.o.b.a.l.\.6.
00C4D6A0  32 00 37 00 64 00 66 00 64 00 30 00 32 00 38 00  2.7.d.f.d.0.2.8.
00C4D6B0  65 00 65 00 38 00 66 00 33 00 37 00 38 00 39 00  e.e.8.f.3.7.8.9.
00C4D6C0  62 00 31 00 38 00 35 00 64 00 33 00 37 00 35 00  b.1.8.5.d.3.7.5.
00C4D6D0  30 00 32 00 32 00 39 00 39 00 38 00 31 00 00 00  0.2.2.9.9.8.1...
```

[그림 34] MD4 해시값을 이용한 뮤텍스 이름 생성

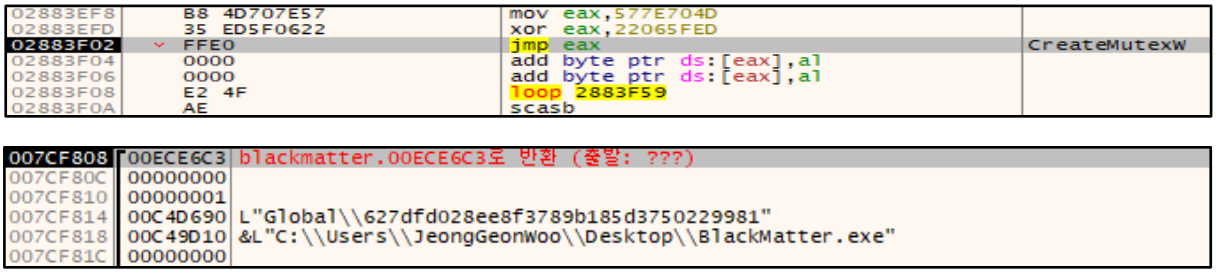
0xEC5FEC 서브루틴 내에서 이전에 생성한 MD4 해시값으로 뮤텍스 이름을 생성한다. 뮤텍스 이름은 Global₩627dfd028ee8f3789b185d3750229981 이다.

```
02883EC8    B8 CD6F7E57         mov eax,577E6FCD
02883ECD    35 ED5F0622         xor eax,22065FED
02883ED2    FFE0                jmp eax                         OpenMutexW
02883ED4    0000                add byte ptr ds:[eax],al
02883ED6    0000                add byte ptr ds:[eax],al
```

```
007CF808  00ECE69F  blackmatter.00ECE69F로 반환 (출발: ???)
007CF80C  00100000
007CF810  00000000
007CF814  00C4D690  L"Global\\627dfd028ee8f3789b185d3750229981"
007CF818  00C49D10  &L"C:\\Users\\JeongGeonWoo\\Desktop\\BlackMatter.exe"
007CF81C  00000000
```

[그림 35] 뮤텍스 이름을 통한 중복 실행 확인

중복 실행 여부를 판단하기 위해 OpenMutexW API 를 이전에 생성한 뮤텍스 이름으로 호출한다. 만약 동일한 뮤텍스가 발견될 경우에는 중복 실행으로 판단하여 현재 프로세스를 종료한다.

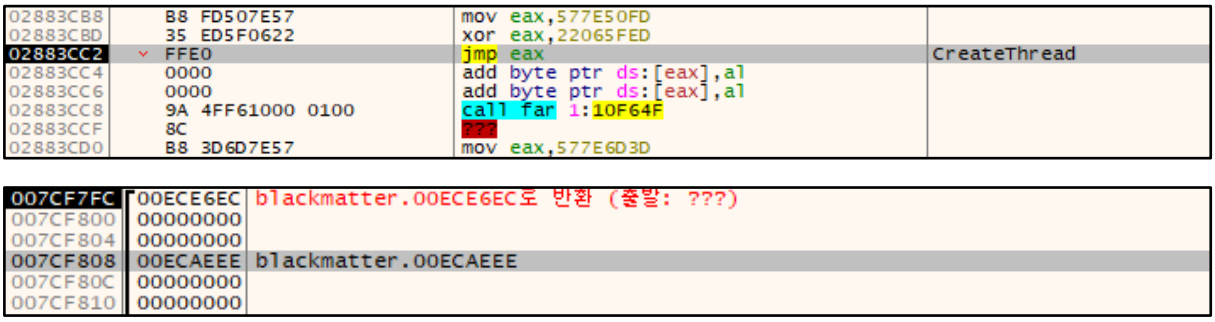


[그림 36] 중복 실행 방지를 위한 뮤텍스 생성

동일한 이름의 뮤텍스가 발견되지 않을 경우, 역시 중복 실행을 방지하기 위하여 뮤텍스를 생성한다. 해당 뮤텍스 이름은 이전에 생성한 값이 사용되며, CreateMutexW API 를 호출하여 수행된다.

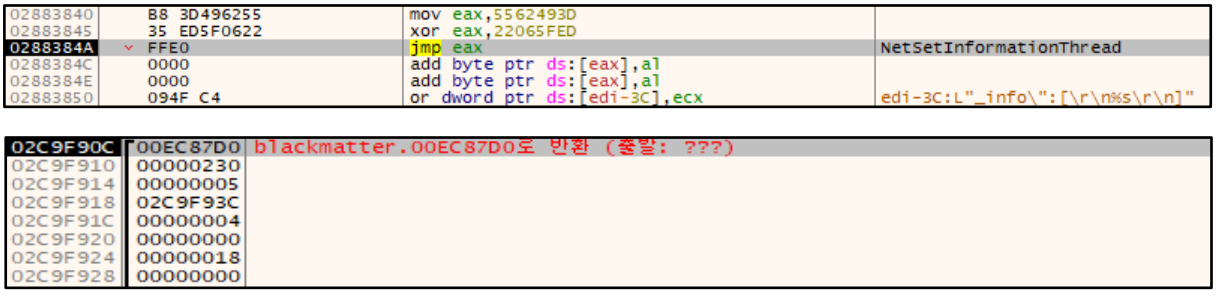
2.1.15 멀티스레딩 방식 사용

BlackMatter 는 멀티 스레드를 사용하는데, 각 스레드는 공격자가 지정한 기능들을 수행한다. 또한 각 스레드 생성 시 스레드의 우선 순위를 높게 설정하여 신속한 행위가 수행된다.



[그림 37] 멀티 스레드 사용을 위한 스레드 생성

CreateThread API 를 호출하여 각 기능을 수행하는 여러 개의 스레드를 생성한다.



[그림 38] 스레드 우선 순위 조정

생성한 스레드들에 대해 각각 NtSetInformationThread API 를 0x5 (ProcessBasePriority)로 호출하여 스레드의 우선 순위를 구한 후, 우선 순위를 높게 설정해주어 신속한 행위를 수행할 수 있도록 한다.

```
02883D90    B8 0D6C7E57     mov eax,577E6C0D
02883D95    35 ED5F0622     xor eax,22065FED
02883D9A    FFE0            jmp eax                 GetLogicalDriveStringsW
02883D9C    0000            add byte ptr ds:[eax],al
02883D9E    0000            add byte ptr ds:[eax],al
02883DA0    B7 4F           mov bh,4F               4F:'O'
02883DA2    DD10            fst qword ptr ds:[eax],st(0)
```

[그림 39] 연결된 드라이브 이름 획득

생성된 특정 스레드는 GetLogicalDriveStringsW API 를 호출하여 현재 연결되어 있는 드라이브 이름을 획득한다.

```
02883FA0    B8 ED6C7E57     mov eax,577E6CED
02883FA5    35 ED5F0622     xor eax,22065FED
02883FAA    FFE0            jmp eax                 GetDriveTypeW
02883FAC    0000            add byte ptr ds:[eax],al
02883FAE    0000            add byte ptr ds:[eax],al
02883FB0    F5              cmc
02883FB1    4F              dec edi                 edi:EntryPoint
02883FB2    9B              fwait
```

[그림 40] 드라이브 유형 획득

생성된 특정 스레드는 GetDriveTypeW API 를 호출하여 암호화 대상 드라이브의 유형을 획득한다.

```
02883CD0    B8 3D6D7E57     mov eax,577E6D3D
02883CD5    35 ED5F0622     xor eax,22065FED
02883CDA    FFE0            jmp eax                 GetDiskFreeSpaceExW
02883CDC    0000            add byte ptr ds:[eax],al
02883CDE    0000            add byte ptr ds:[eax],al
02883CE0    9F              lahf
02883CE1    4F              dec edi
02883CE2    F5              cmc
```

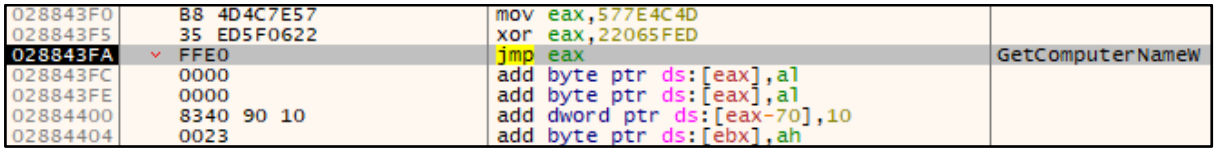
[그림 41] 드라이브 용량 획득

생성된 특정 스레드는 GetDiskFreeSpaceExW API 를 호출하여 대상 드라이브의 총 용량과 사용량을 획득한다.

```
028840F0    B8 4DB10254     mov eax,5402B14D
028840F5    35 ED5F0622     xor eax,22065FED
028840FA    FFE0            jmp eax                 GetUserNameW
028840FC    0000            add byte ptr ds:[eax],al
028840FE    0000            add byte ptr ds:[eax],al
02884100    2340 F0         and eax,dword ptr ds:[eax-10]
```

[그림 42] 사용자 이름 획득

생성된 특정 스레드는 GetUserNameW API 를 호출하여 현재 사용자명을 획득한다.



[그림 43] PC 이름 획득

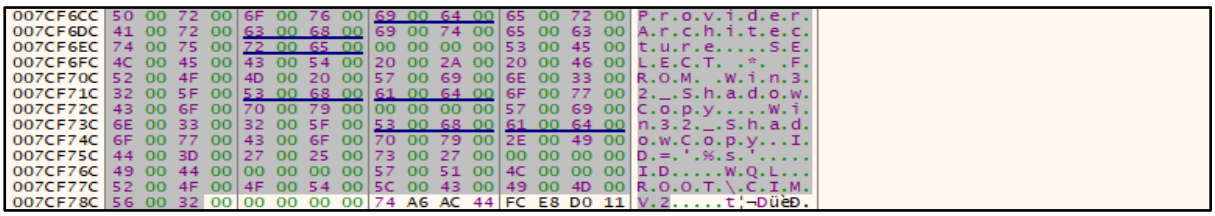
생성된 특정 스레드는 GetComputerNameW API 를 호출하여 현재 PC 명을 획득한다.

각 스레드들을 통해 획득한 정보는 이후 하나의 데이터로 통합되어 C&C 서버로 전송된다. 이를 통해 공격자는 감염 PC 의 정보와 감염 사실을 알 수 있고, 추후 데이터 탈취 등의 행위도 수행 할 수 있다.

[표 6] 멀티 스레드를 통한 악성 행위

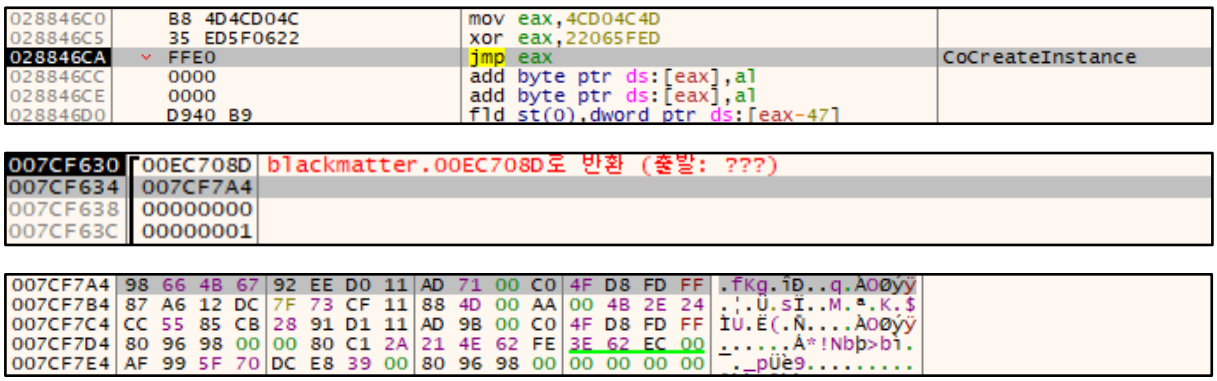
2.1.16 64 비트 아키텍처 사용 및 볼륨쉐도우카피 정보 획득

BlackMatter 는 아키텍처를 강제로 로드하고, IWBemLocator 개체를 사용하여 볼륨쉐도우카피 정보를 획득한다.



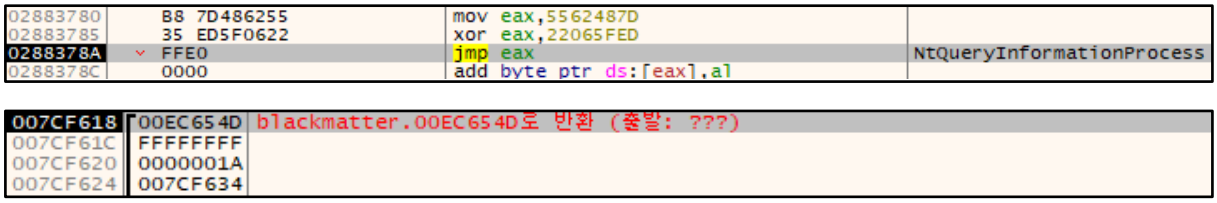
[그림 44] COM 개체 사용을 위한 문자열

64 비트 아키텍처 사용과 볼륨쉐도우카피 정보를 획득하기 위해 필요한 문자열을 메모리에 로드한다. 위 사진을 보면 ProviderArchitecture 문자열와 SELECT * FROM Win32_ShadowCopy 등의 문자열을 확인할 수 있는데, 이들 문자열은 이후 각 API 및 메서드들에 의해 호출된다.



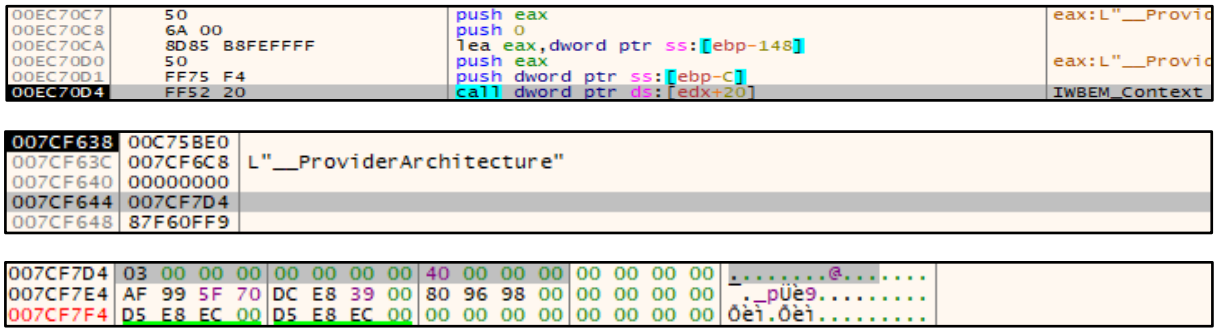
[그림 45] RIID 를 통한 IWBem 사용

CoCreateInstance API 를 호출하는데, 이는 COM 개체를 사용하는 것이다. 이 때, RIID 값을 확인하면 674B6698-EE92-11D0-AD71-00C04FD8FDFF 를 확인할 수 있다. 이는 IWBem Context 를 의미하는데, 이를 통해 WMI 네임 스페이스에 대한 연결을 만들 수 있다.



[그림 46] WOW64 환경 확인

0x1A (ProcessWow64Information)를 인자로 NtQueryInformationProcess API 를 호출하여 프로세스가 WOW64 환경에서 실행중인지 여부를 확인한다.

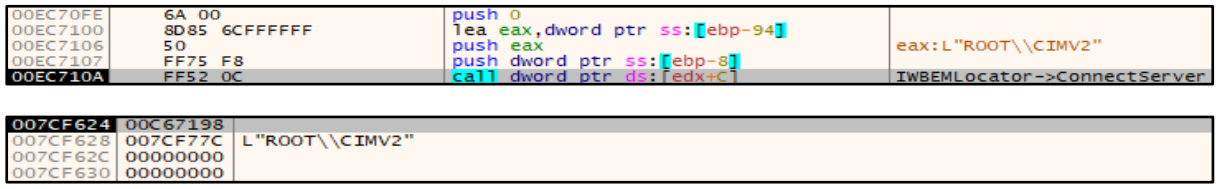


[그림 47] 64 비트 아키텍처 사용 설정

__ProviderArchitecture 인자를 이용한 호출 컨텍스트 개체를 사용하여 64 비트 아키텍처 플랫폼에서 WMI 데이터를 요청할 수 있도록 한다. 이는 스택과 메모리를 보면 알 수 있는데, 0x3 과 0x40 (64)를 확인할 수 있다. 이는 아래의 표와 같은 형태로 호출된다.

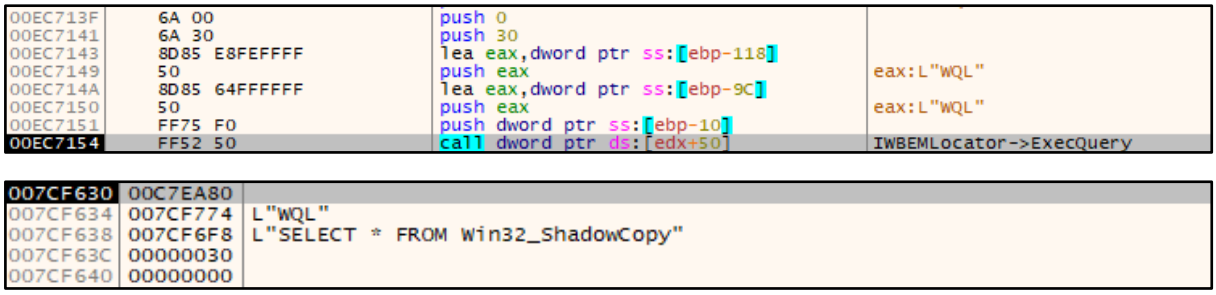
```
Value.v1 = 3
Value.v2 = 64
IWBEM_Context->SetValue(IWBEM_Context, __ProviderArchitecture, 0, &Value)
```

[표 7] IWBem_Context 수행 코드



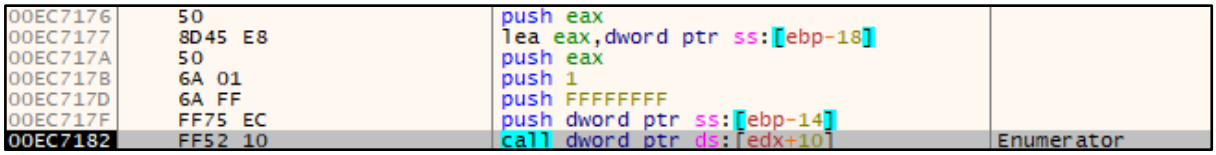
[그림 48] ROOT₩CIMV2 네임스페이스에 대한 연결

IWBemLocator 개체를 사용하여 ConnectServer 메서드를 호출한 후, ROOT₩CIMV2 네임스페이스에 연결하고 IWBemServices 개체에 대한 포인터를 얻는다.



[그림 49] IWBemServices 개체를 사용한 ExecQuery 수행

IWBemServices 개체를 사용하여 ExecQuery 메서드로 WQL (WMI Query Language)와 SQL 쿼리인 SELECT * FROM Win32_ShadowCopy 를 실행하여, 로컬 서버에 저장된 모든 볼륨쉐도우카피 정보의 열거자를 획득한다.

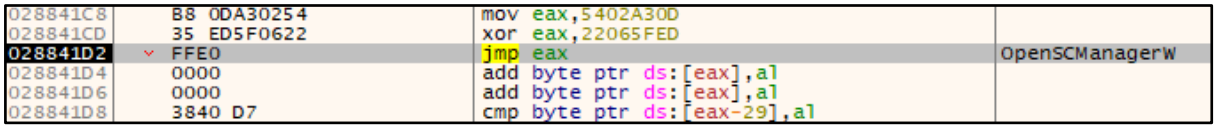


[그림 50] Enumerator 메서드를 이용한 볼륨쉐도우카피 정보 획득

이후 Enumerator 메서드를 수행하여 볼륨쉐도우카피 정보를 열거한다. 이러한 위 과정을 거쳐 획득한 볼륨쉐도우카피 정보는 이후 시스템 복원이 불가능하도록 볼륨쉐도우카피 삭제에 이용된다.

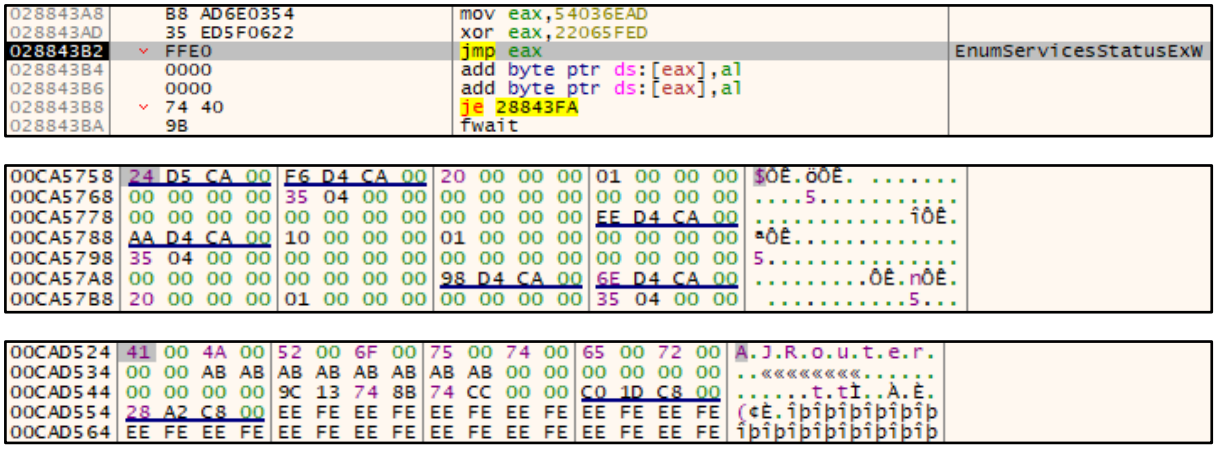
2.1.17 대상 서비스 제거

BlackMatter 는 악성 행위에 방해가 되는 대상 서비스들을 제거한다. 이는 백업 서비스 등이 해당된다.



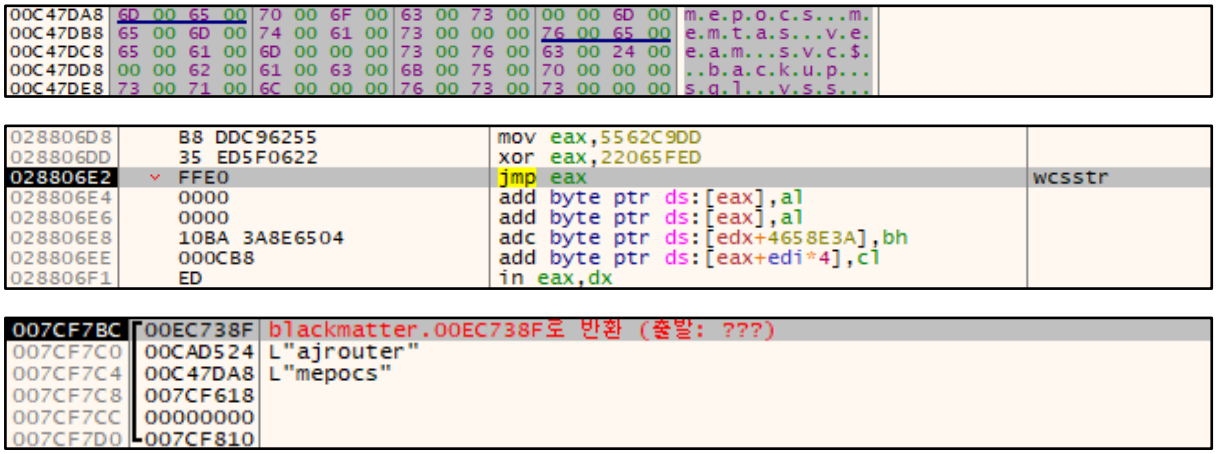
[그림 51] 서비스 제어 관리자 데이터 베이스 접근

OpenSCManagerW API 를 호출하여 서비스 제어 관리자 데이터베이스를 연다.



[그림 52] 서비스 제어 관리자 데이터베이스의 서비스 열거

EnumServicesStatusExW API 를 호출하여 서비스 제어 관리자 데이터베이스의 서비스를 열거한다.

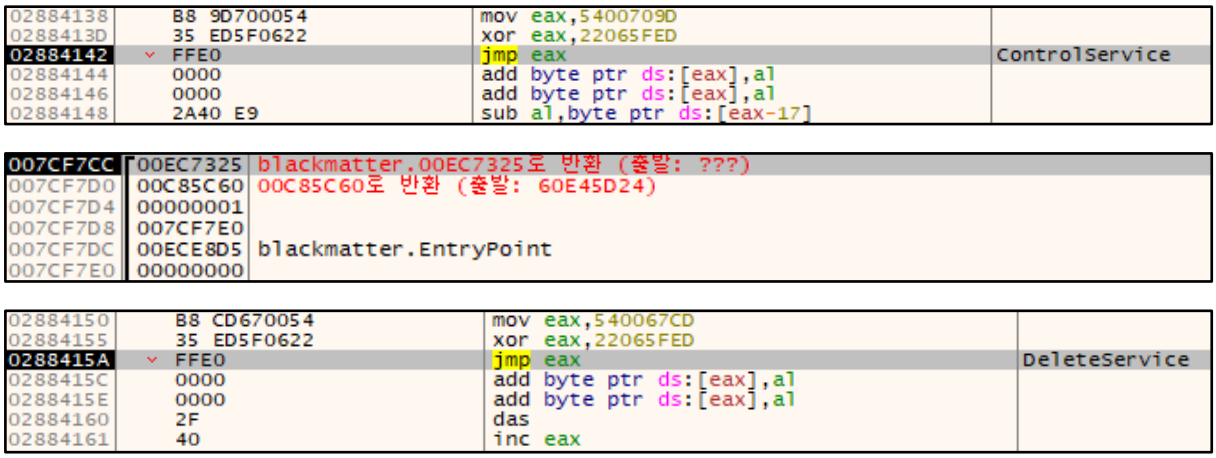


[그림 53] 종료 대상 서비스 목록과 비교

이전의 EnumServicesStatusExW API 를 호출하여 얻은 서비스 목록들을 wcsstr API 를 호출하여 종료 대상 서비스 목록과 비교한다. 만약 일치하는 문자열을 발견하면 이후 대상 서비스를 제거한다. 제거 대상 서비스 목록은 아래의 표와 같다.

veeam	Veeam Backup Solution 관련 서비스
memtas	Mail 관련 서비스
sql	SQL 관련 서비스
backup	Backup 관련 서비스
vss	VolumeShadowCopy 관련 서비스
svc$	SVSVC 등 암호화에 방해가 되는 서비스
mepocs	Mail 관련 서비스

[표 8] 제거 대상 서비스 목록

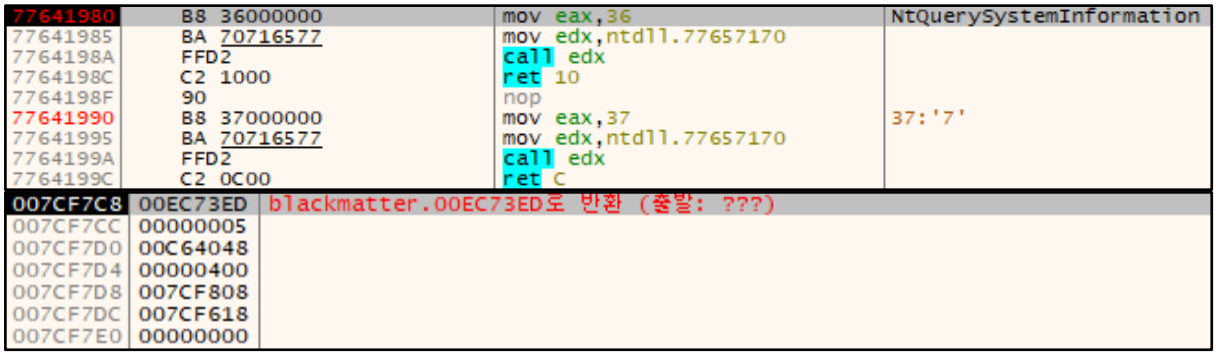


[그림 54] 서비스 제거

제거 대상 서비스가 발견되면 0x00000001 (SERVICE_CONTROL_STOP)를 인자로 ControlService API 를 호출하여 대상 서비스를 중지시킨다. 중지 작업을 수행한 후 DeleteService API 를 호출하여 대상 서비스를 시스템에서 제거한다.

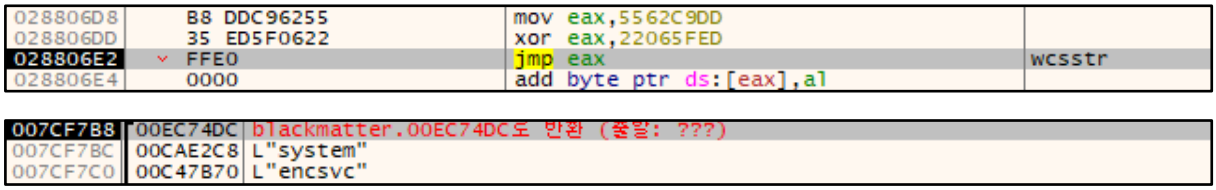
2.1.18 대상 프로세스 종료

BlackMatter 는 악성 행위에 방해가 되는 대상 프로세스들을 종료시킨다.



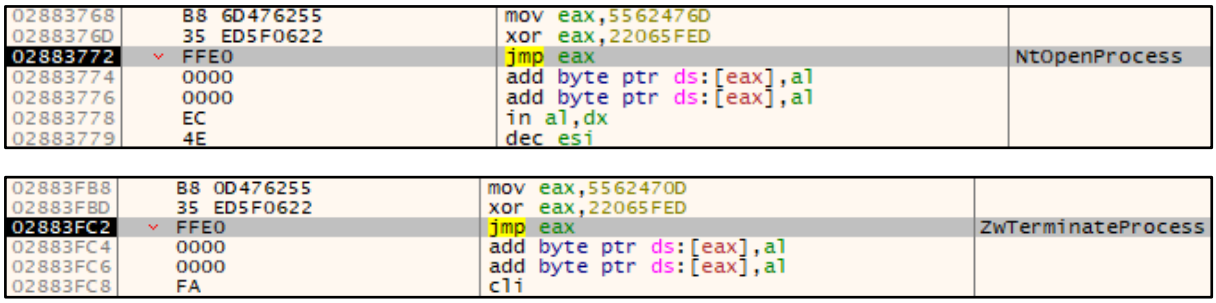
[그림 55] 프로세스 목록 획득

NtQuerySystemInformation API 를 0x5 (SystemProcessInformation) 인자를 주어 호출한다. 이를 통해 SYSTEM_PROCESS_INFO 구조체에 정보를 받아와 실행중인 프로세스 목록을 획득한다.



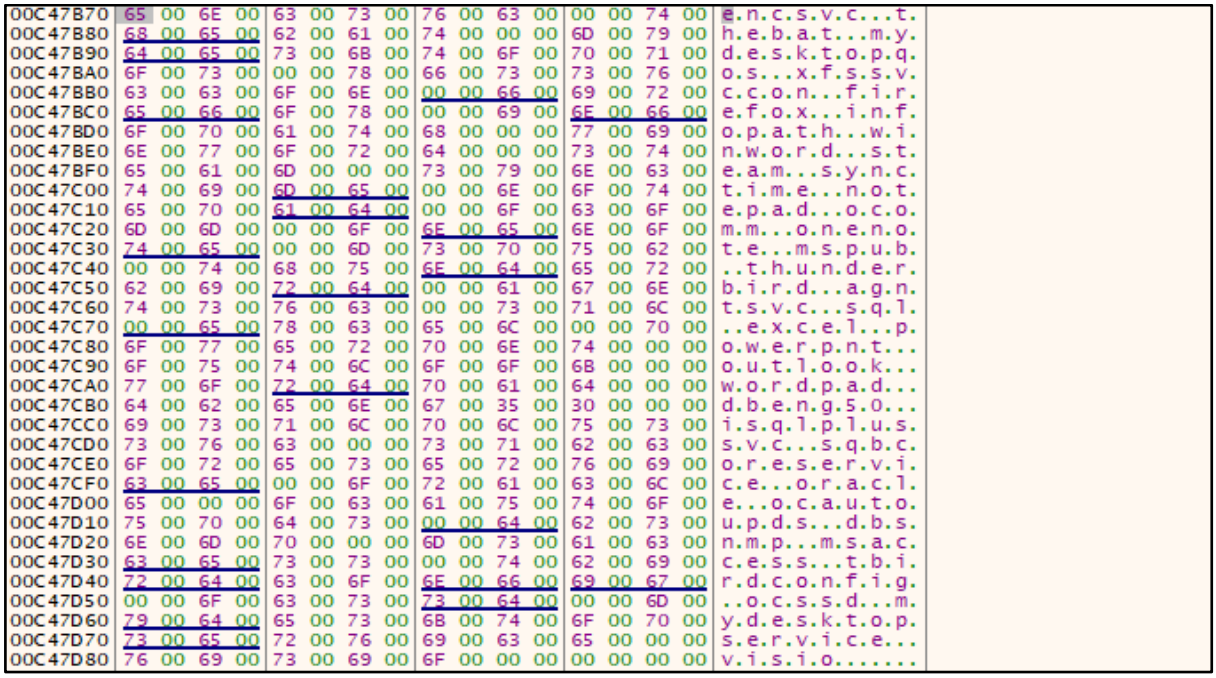
[그림 56] 종료 대상 프로세스 목록 비교

프로세스 목록을 획득한 후 wcsstr API 를 호출하여 종료 대상 프로세스 목록과 비교한다.



[그림 57] 대상 프로세스 종료

비교 후 종료 대상 프로세스로 확인되면, NtOpenProcess 와 ZwTerminateProcess API 를 호출하여 대상 프로세스를 종료한다.



[그림 58] 메모리 내 종료 대상 프로세스 목록

종료 대상 프로세스 목록은 메모리 내 적재되어 위치하고 있으며, 이에 대한 자세한 설명은 아래의 표와 같다.

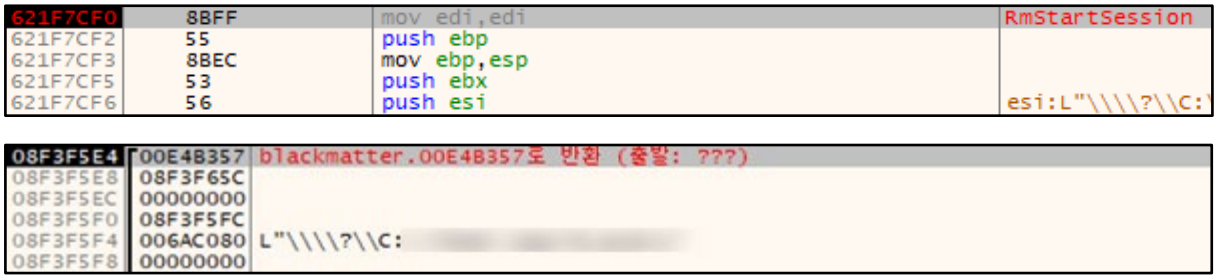
encsvc	Citrix Encryption Service 관련 프로세스
thebat	The Bat! E-Mail Client 관련 프로세스
mydesktopqos	Oracle MyDesktop Service 관련 프로세스
xfssvccon	Oracle WebDav 관련 프로세스
firefox	Firefox Browser 관련 프로세스
infopath	Microsoft InfoPath 관련 프로세스
winword	Microsoft WinWord 관련 프로세스
steam	Valve Corporation 의 Steam 관련 프로세스
synctime	File Synchronization 관련 프로세스
notepad	Microsoft NotePad 관련 프로세스
ocomm	Oracle Communicator 관련 프로세스

onenote	Microsoft OneNote 관련 프로세스
mspub	Microsoft MSPub 관련 프로세스
thunderbird	Mozilla Thunderbird 관련 프로세스
agntsvc	Oracle Intelligent Agent 에 사용되는 관련 프로세스
sql	SQL 관련 프로세스
excel	Microsoft Excel 관련 프로세스
powerpnt	Microsoft PowerPoint 관련 프로세스
outlook	Microsoft Outlook 관련 프로세스
wordpad	Microsoft WordPad 관련 프로세스
dbeng50	DataBase Engine 에 사용되는 관련 프로세스
isqlplussvc	Oracle IPlusSvce 관련 프로세스
sqbcoreservice	SQL Backup Agent Service 관련 프로세스
oracle	Oracle 관련 프로세스
ocautoupds	Oracle Connector Auto Update Service 관련 프로세스
dbsnmp	Oracle Intelligent Agent 에 사용되는 관련 프로세스
msaccess	Microsoft MSAccess 관련 프로세스
tbirdconfig	Mozilla Thunderbird 관련 프로세스
ocssd	Oracle Cluster Synchronization Services (OCSSD) 관련 프로세스
mydesktopservice	Oracle MyDesktop Service 관련 프로세스
visio	Microsoft Visio 관련 프로세스

[표 9] 종료 대상 프로세스 목록

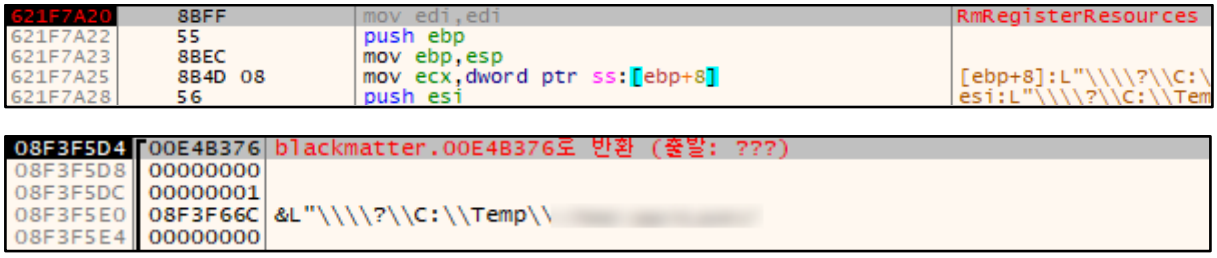
2.1.19 공유 위반이 발생한 프로세스 종료

BlackMatter 는 파일 암호화 수행 중 대상 파일이 사용 중일 경우, 공유 위반 발생 시 해당 파일을 사용 중인 프로세스를 종료한다. 이는 Rstrtmgr.dll 의 API 를 사용하여 수행된다.



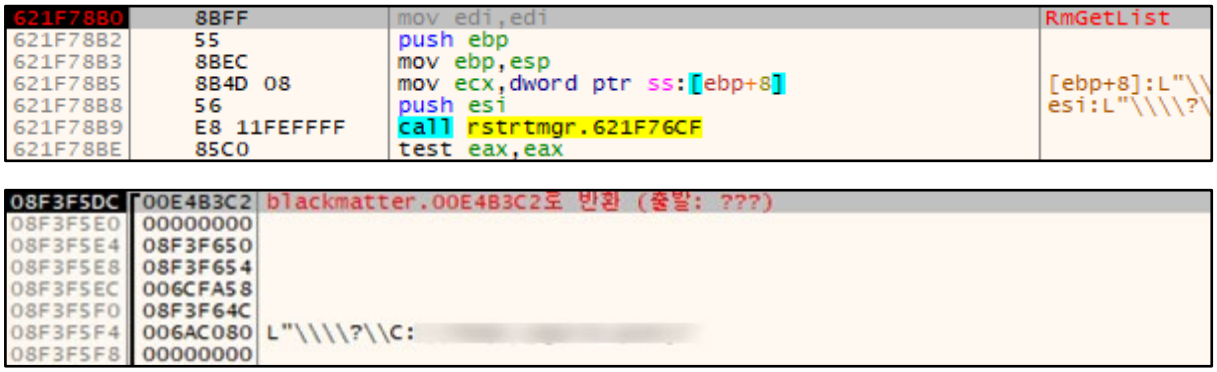
[그림 59] 공유 위반 발생 시 다시 시작 관리자 세션 시작

공유 위반이 발생한 파일을 사용하는 프로세스를 종료시키기 위해 RmStartSession API 를 호출하여 다시 시작 관리자 세션을 시작한다.



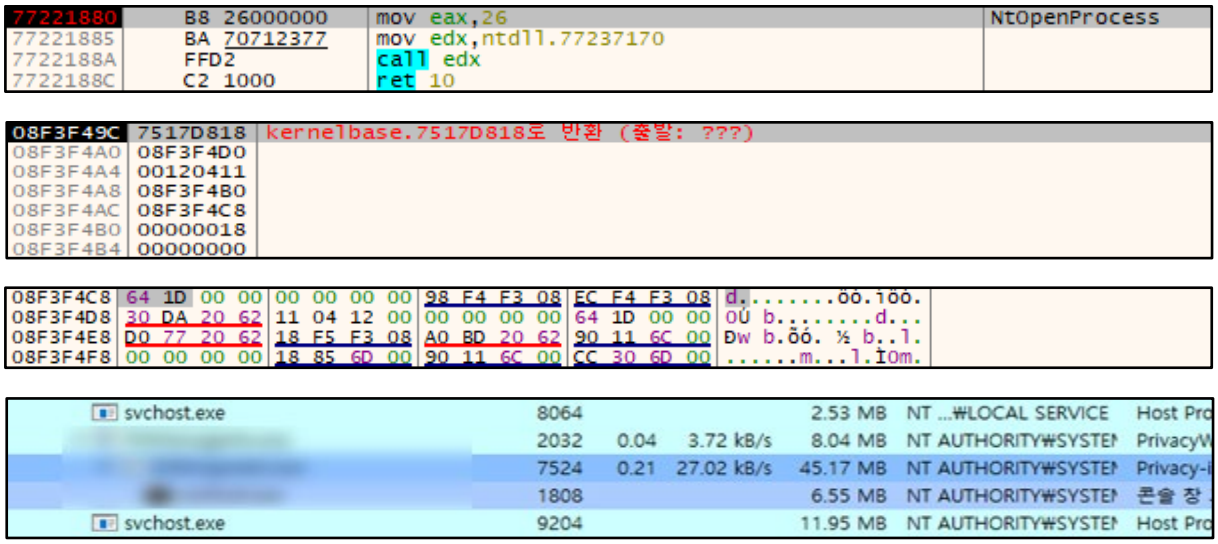
[그림 60] 공유 위반 발생 파일 이름의 리소스 등록

RmRegisterResources API 를 호출하여 대상 파일의 이름을 리소스로 등록한다. 현재 대상이 되는 파일은 Temp 폴더 내 특정 파일로, 이는 Privacy-i EDR 이 사용하는 파일이다.



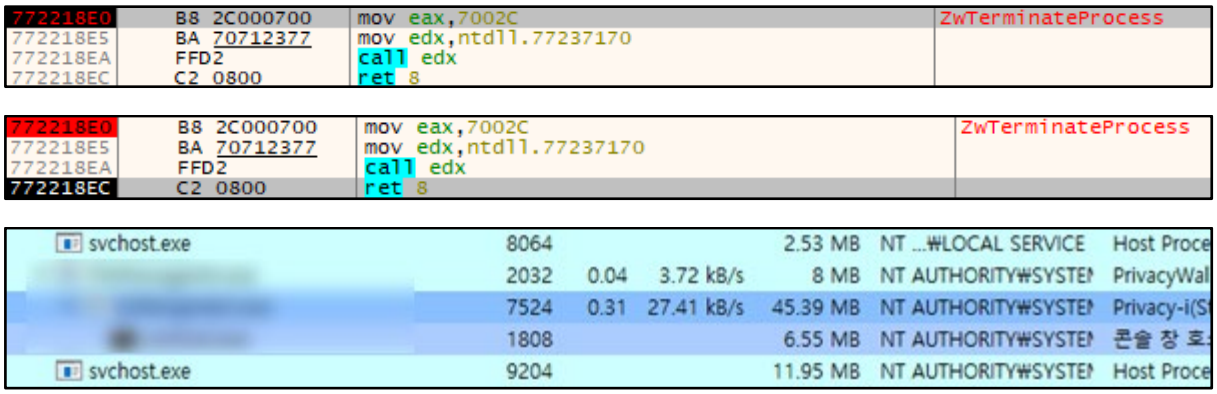
[그림 61] 공유 위반 파일을 사용하는 프로세스 및 서비스 목록 획득

리소스에 등록된 파일을 사용하는 모든 프로세스 및 서비스 목록을 가져온다.



[그림 62] 공유 위반이 발생한 Privacy-i EDR 종료를 위한 핸들 획득

Privacy-i EDR 과 공유 위반이 발생하자, 원인이 되는 [Privacy-i EDR].exe 프로세스를 종료시키려 시도한다. 이는 NtOpenProcess API 를 호출하여 핸들을 획득하는 것으로 시작한다. 메모리의 0x08F3F4C8 위치를 확인하면 0x1D64 (7524)를 확인할 수 있는데, 이는 [Privacy-i EDR].exe 의 PID 값이다.



[그림 63] 공유 위반이 발생한 Privacy-i EDR 종료 시도 및 실패

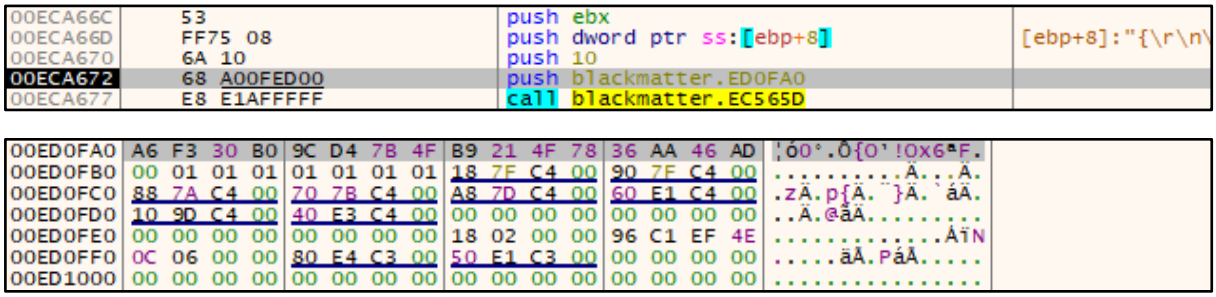
이전에 구한 핸들을 이용하여 ZwTerminateProcess API 를 호출한다. 이를 통해 공유 위반이 발생한 [Privacy-i EDR].exe 프로세스를 종료시키려 시도한다. 그러나, Privacy-i EDR 은 방어 코드가 적용되어 있어 종료에 실패한다.

Privacy-i EDR 은 공유 위반이 발생하는 프로세스에 대해 종료를 수행하는 악성 행위에 대해 사전에 파악하여 이를 방어할 수 있도록 방어 코드를 개발하였다. 이를 통해 BlackMatter 의 프로세스 종료 시도를 사전에 차단할 수 있다.

[표 10] Privacy-i EDR 의 방어 기술

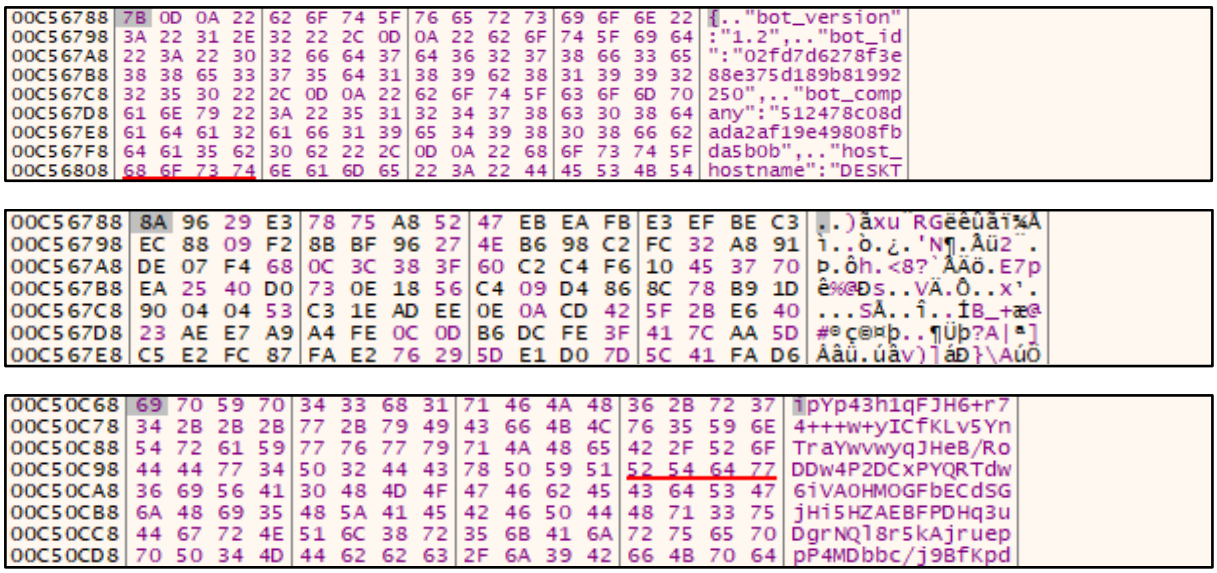
2.1.20 C&C 서버 연결 및 감염 PC 정보 전송

BlackMatter 는 C&C 서버에 감염 PC 에 대한 정보를 전송한다. 이 때, 감염 PC 에 대한 정보는 암호화 알고리즘인 AES 와 Base64 로 암호화 및 인코딩된다. AES 암호화 키는 이전에 리소스 압축 해제 수행 시 확인한 암호화 키 값인 A6F330B09CD47B4FB9214F7836AA46AD 이다.



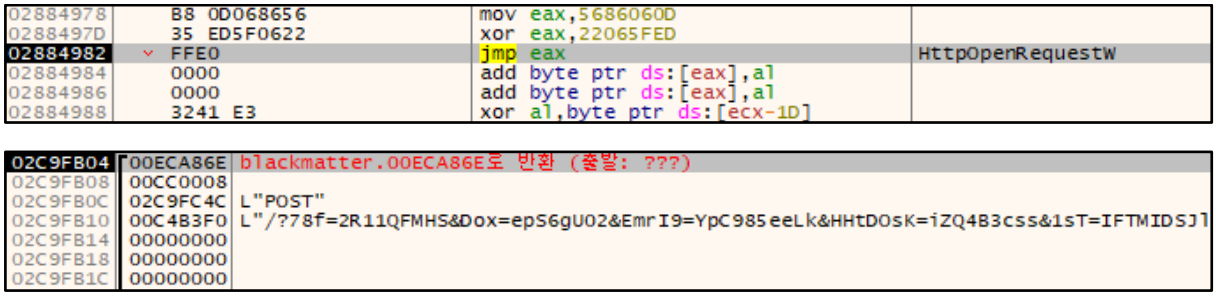
[그림 64] C&C 서버 전송 데이터의 암호화 키

C&C 서버에 감염 PC 정보를 보내기 전, 감염 PC 정보에 대해 AES 암호화를 수행한다. 이 때, 키 값은 이전에 확인했던 A6F330B09CD47B4FB9214F7836AA46AD 이다. 이는 메모리에서 확인할 수 있다.



[그림 65] C&C 서버 전송 데이터의 암호화 및 인코딩

AES 암호화의 결과로 첫 번째 사진의 감염 PC 정보가 두 번째 사진처럼 암호화된다. 이후 Base64 로 인코딩을 수행하여 세 번째 사진의 형태로 감염 PC 정보가 변경된다.



[그림 66] C&C 서버로 암호화 및 인코딩된 데이터 전송

AES 암호화와 Base64 인코딩을 한 감염 PC 정보는 C&C 서버로 전송된다. 대상 C&C 서버 목록은 아래의 표와 같다.

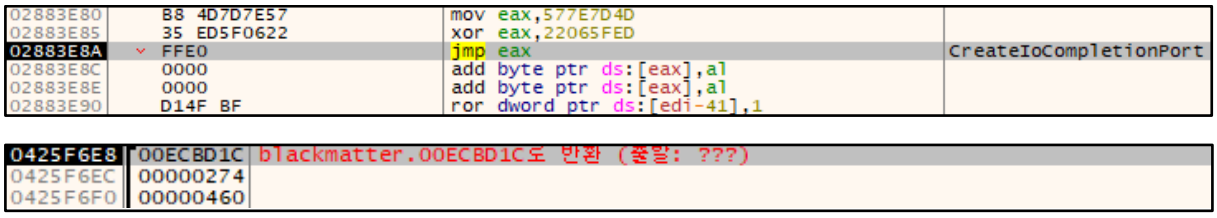
http://paymenthacks.com
https://paymenthacks.com
http://mojobiden.com
https://mojobiden.com

[표 11] C&C 서버 목록

2.1.21 시스템 암호화

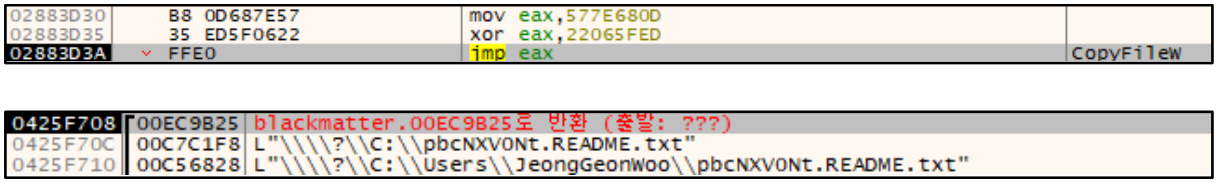
2.1.21.1 시스템 암호화 (파일 속성 변경)

BlackMatter 는 시스템 내 파일 암호화에 앞서, 입출력 I/O 포트를 생성한다. 이를 통해 멀티 스레드 기반의 빠른 속도로 암호화가 진행이 되며, 각 파일의 속성을 변경하여 파일 보호를 우회한다.



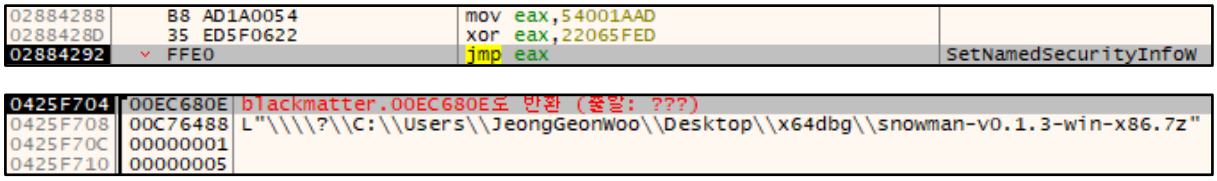
[그림 67] 입출력 I/O 포트 생성

파일 암호화에 앞서, 비동기 I/O 작업 완료 알림을 받는 포트들을 생성한다. 이 포트들은 각 스레드에 할당되며 파일 암호화 시 작업에 대한 알림을 받는다. 이를 통해 효율적인 암호화가 가능하다.



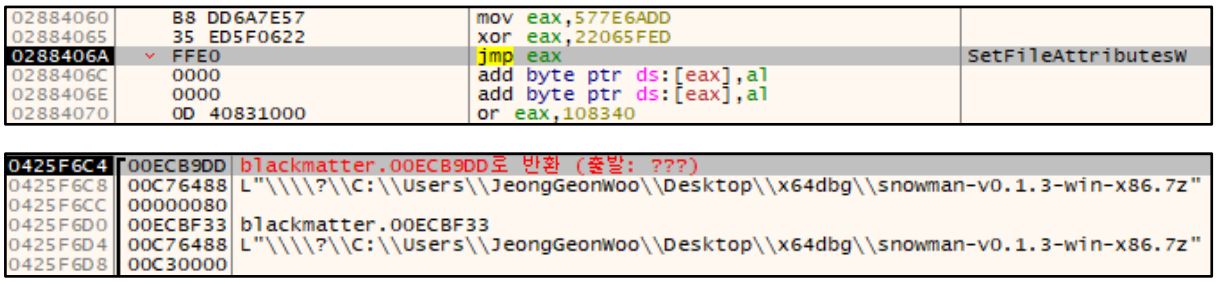
[그림 68] 랜섬 노트 복사

랜섬 노트는 C 드라이브에 생성된 후, 각 폴더에 CopyFileW API 를 호출하여 복사된다.



[그림 69] 암호화를 위한 파일 개체 변경

SetNamedSecurityInfoW API 를 호출하여, 모든 파일에 대해 개체의 형태를 파일 개체로 변경시킨다. 이를 통해 파일의 유형에 상관없이 암호화가 가능하다. 이 때, 인자는 0x1 (SE_FILE_OBJECT)와 0x5 (OWNER_SECURITY_INFORMATION | DACL_SECURITY_INFORMATION)를 사용한다.

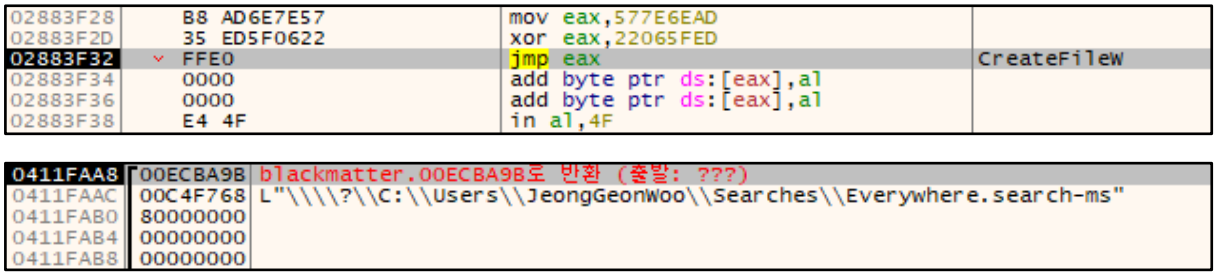


[그림 70] 암호화를 위한 파일 속성 변경

SetFileAttributesW API 를 0x80 (FILE_ATTRIBUTE_NORMAL) 인자를 주어 호출하여 파일 속성을 변경한 후, 파일 보호를 회피한다.

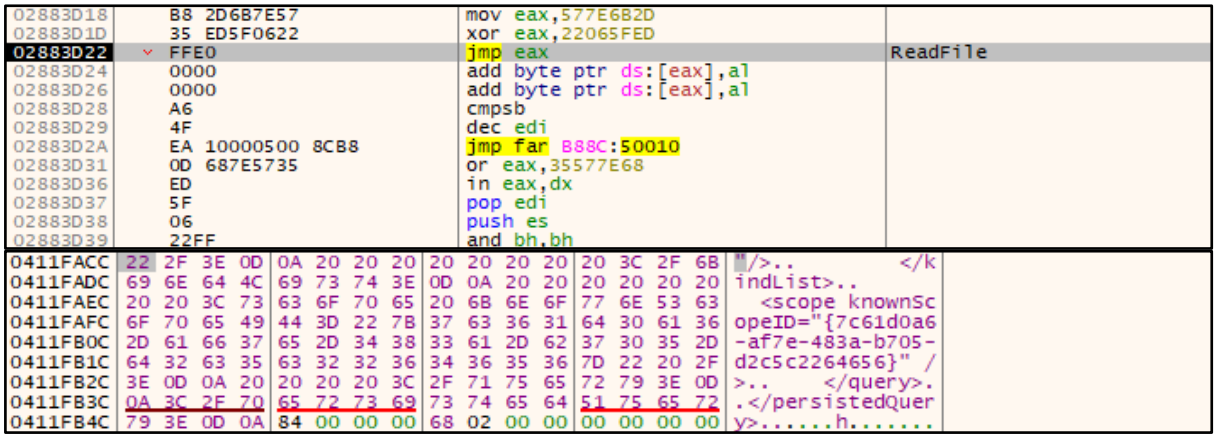
2.1.21.2 시스템 암호화 (파일 암호화)

BlackMatter 는 시스템 내 파일을 암호화하는데, RSA 알고리즘을 사용한다. 이 때 키는 이전에 리소스 압축 해제 시 확인한 키 값을 사용한다.



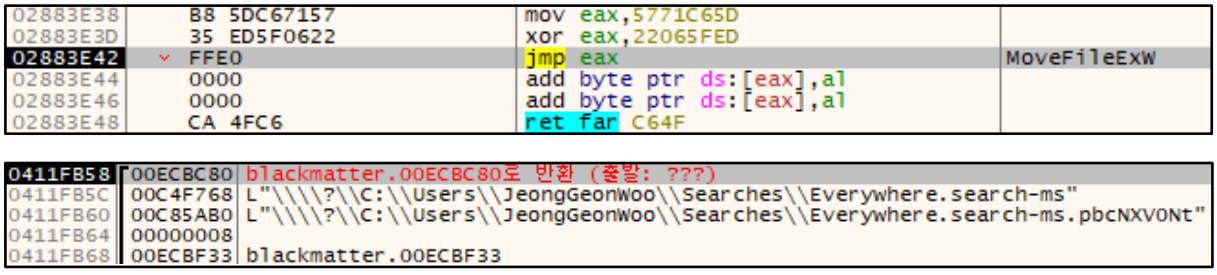
[그림 71] 암호화를 위한 파일 핸들 획득

CreateFileW API 를 이용하여 암호화 대상 파일의 핸들을 획득한다.



[그림 72] 암호화를 위한 버퍼 내 파일 내용 적재

ReadFile API 를 이용하여 암호화 대상 파일의 내용을 버퍼로 읽어들인다.



[그림 73] 암호화 파일 확장자 변경

MoveFileExW API 를 이용하여 확장자명을 암호화 확장자명으로 변경시킨다.

```
00ECBBCB    83C4 0C          add esp,C
00ECBBCE    68 100FED00      push blackmatter.ED0F10          RSA_Key
00ECBBD3    8D46 70          lea eax,dword ptr ds:[esi+70]
00ECBBD6    50               push eax
00ECBBD7    E8 AB5BFFFF      call blackmatter.EC1787
00ECBBDC    68 80000000      push 80
```

```
0411FB44  00CA5E50
0411FB48  00ED0F10  blackmatter.00ED0F10
0411FB4C  00C4F768  L"\\\\?\\C:\\Users\\JeongGeonWoo\\Searches\\Everywhere.search-ms"
0411FB50  0411FBC8  L"Everywhere.search-ms"
0411FB54  00CA5DE0
```

```
00ED0F10  87 19 A8 30 F4 BA 94 94  92 91 58 2B 66 54 F9 6C  .. 0ô°....X+fTùl
00ED0F20  96 D9 A0 F4 41 9F 52 F3  67 CF 2E 19 B9 C9 5A 9B  .Ù ôA.RógÏ..'ÉZ.
00ED0F30  70 91 CB EF AF BE 5A E3  9D AE 28 58 94 59 0A 8D  p.Ëï¯¾Zã.®(X.Y..
00ED0F40  B8 B7 64 E5 72 FA B5 23  46 46 F8 65 9A DA 2F BD  .·dårúµ#FFøe.Ú/½
00ED0F50  8C 37 BF DD D6 07 97 A5  AD 9D AD 2D ED 37 96 9D  .7¿ÝÖ..¥...-í7..
00ED0F60  17 9E A4 AD 4C 19 80 D0  E7 0B 05 62 41 D3 25 E1  ..¤.L..Ðç..bAÓ%á
00ED0F70  8B EB 5C C4 92 5F A5 6A  BF 81 0F 91 6E 79 32 D0  .ë\Ä._¥j¿...ny2Ð
00ED0F80  16 A8 6E 3A D9 77 49 E7  5F 90 31 11 4B 06 0B 56  . ¨n:ÙwIç_.1.K..V
00ED0F90  51 24 78 C0 8D AD A2 AF  19 E4 98 08 FB DA 5B 0B  Q$xÀ..¢¯.ä..ûÚ[.
```

[그림 74] RSA 알고리즘을 이용한 암호화와 키

버퍼 내 파일의 내용을 암호화 시키는데, RSA 알고리즘을 사용한다. RSA 알고리즘 사용 시 키는 이전 리소스 압축 해제 시 확인한 RSA 키를 사용한다.

2.1.22 랜섬 노트

BlackMatter 는 각 폴더 경로에 랜섬 노트를 생성한다.

```
        '     BLACK        |
   ()    .-.,="``"=.    - o -
         '=/_       ₩     |
      *   |  '=._    |
           ₩     `=./`,        '
        .    '=.__.=' `='      *
+              Matter         +
     O       *        '       .

>>> What happens?
  Your network is encrypted, and currently not operational. We have downloaded 1TB from yo
```

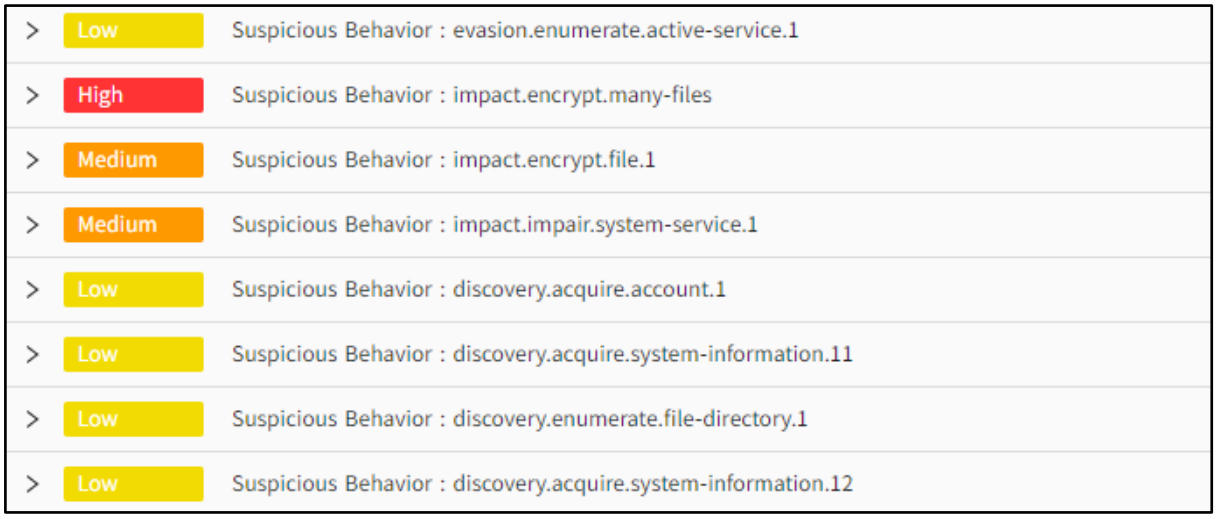
[그림 75] 랜섬 노트

각 폴더 경로에는 위와 같은 BlackMatter 의 랜섬 노트가 생성된다.

3. Privacy-i EDR 탐지 정보

Privacy-i EDR 은 BlackMatter 랜섬웨어에 대해 Ransomware 로 탐지하고 있다.

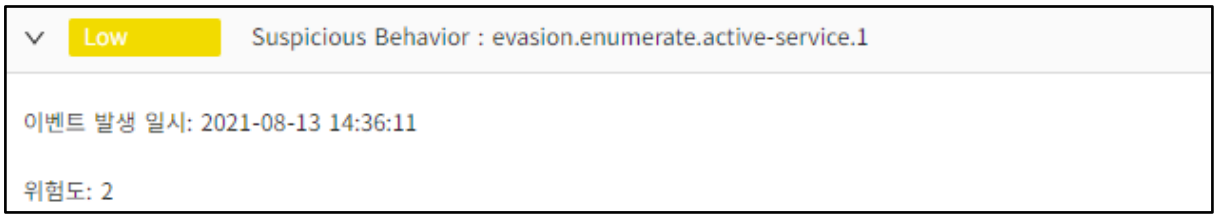
3.1 탐지 행위



[그림 76] Privacy-i EDR 탐지 행위

3.2 주요 탐지 행위

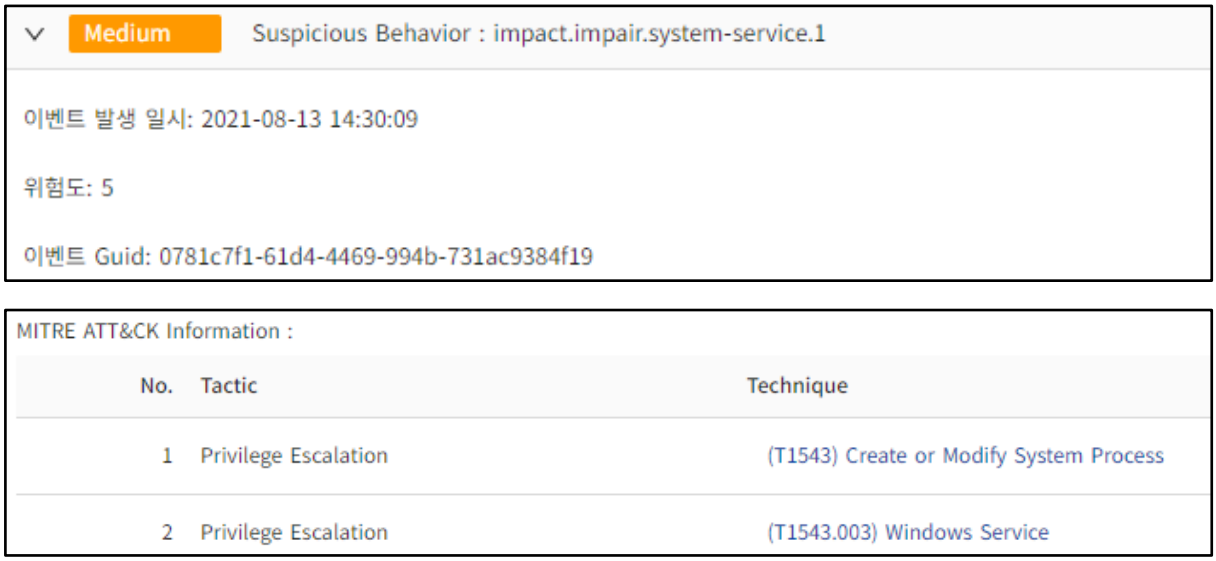
3.2.1 evasion.enumerate.acvtive-service.1



[그림 76] 서비스 열거

BlackMatter 는 종료 대상 서비스를 열거한다. Privacy-i EDR 은 이를 주요 행위로 탐지한다.

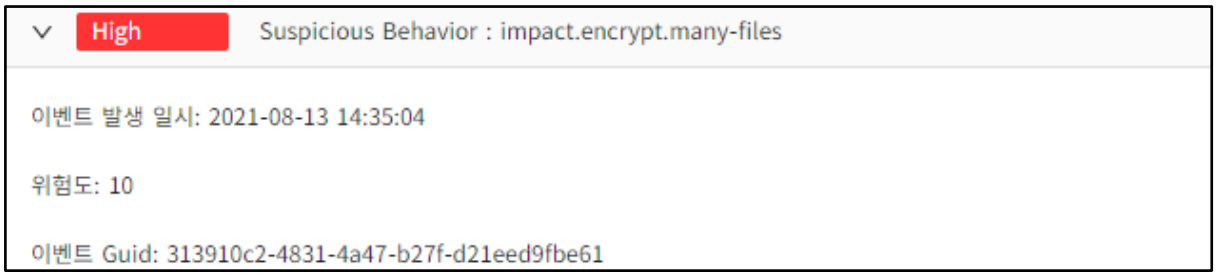
3.2.2 impact.impair.system-service.1



[그림 77] 방화벽 설정 변경

BlackMatter 는 대상 서비스를 종료한다. Privacy-i EDR 은 이를 주요 행위로 탐지한다.

3.2.3 impact.encrypt.many-files



[그림 78] 다수의 파일 암호화

Privacy-i EDR 은 BlackMatter 의 다수의 파일 암호화 행위에 대해 주요 행위 정보로서 탐지한다.

4. 대 응

1. 랜섬웨어 탐지 / 차단이 가능한 EDR 제품을 통해 예방한다.
2. 비 업무 사이트 및 신뢰 할 수 없는 웹사이트의 연결을 차단한다.
3. 주요 문서는 주기적으로 백업하고 물리적으로 분리하여 관리한다.
4. 신뢰 할 수 없는 메일의 첨부 파일은 실행을 금지한다.
5. 비 업무 사이트 및 신뢰 할 수 없는 웹 사이트의 연결을 차단한다.

궁금하신 점이나 문의사항은 malware@somansa.com 으로 해주세요.