

REvil과 DarkSide랜섬웨어의 치명적인 강점만 통합한

BlackMatter 랜섬웨어 분석리포트

2021.08

목 차

1. 개 요	3
1.1 배 경	3
1.2 파일 정보	4
2. 분 석	5
2.1.1 API 동적 호출	6
2.1.2 Anti-Debugging	7
2.1.3 관리자 그룹 확인	8
2.1.4 XOR 연산을 통한 문자열 복호화	8
2.1.5 MachineGuid 획득 및 암호화 확장자 생성	9
2.1.6 BlackMatter 랜섬 노트 이름 생성	10
2.1.7 권한 상승 시도 및 확인	10
2.1.8 프로세스 열거	12
2.1.9 토큰 복제	12
2.1.10 리소스 압축 해제를 위한 키 생성	13
2.1.11 리소스 압축 해제	14
2.1.12 자격 증명을 통한 인증 시도	15
2.1.13 실행 인자에 따른 암호화 행위	16
2.1.14 뮉텍스를 통한 중복 실행 방지	17
2.1.15 멀티스레딩 방식 사용	19
2.1.16 64 비트 아키텍처 사용 및 볼륨쉐도우카피 정보 획득	20
2.1.17 대상 서비스 제거	23
2.1.18 대상 프로세스 종료	24
2.1.19 공유 위반이 발생한 프로세스 종료	27
2.1.20 C&C 서버 연결 및 감염 PC 정보 전송	29
2.1.21 시스템 암호화	30
2.1.22 랜섬 노트	32
3. 탐 지	33
3.1 Privacy-i EDR 탐지 정보	33
4. 대 응	34

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 Posted July 21

byte

B

Seller
● 0
1 post
Joined
07/19/21 (ID: 118280)
Activity
другое / other
Deposit
4.000000 B

We are looking for corporate networks of the following countries:

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

All areas except:

- Medicine.
- State institutions.

Requirements:

- Zoom Revenue or 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.

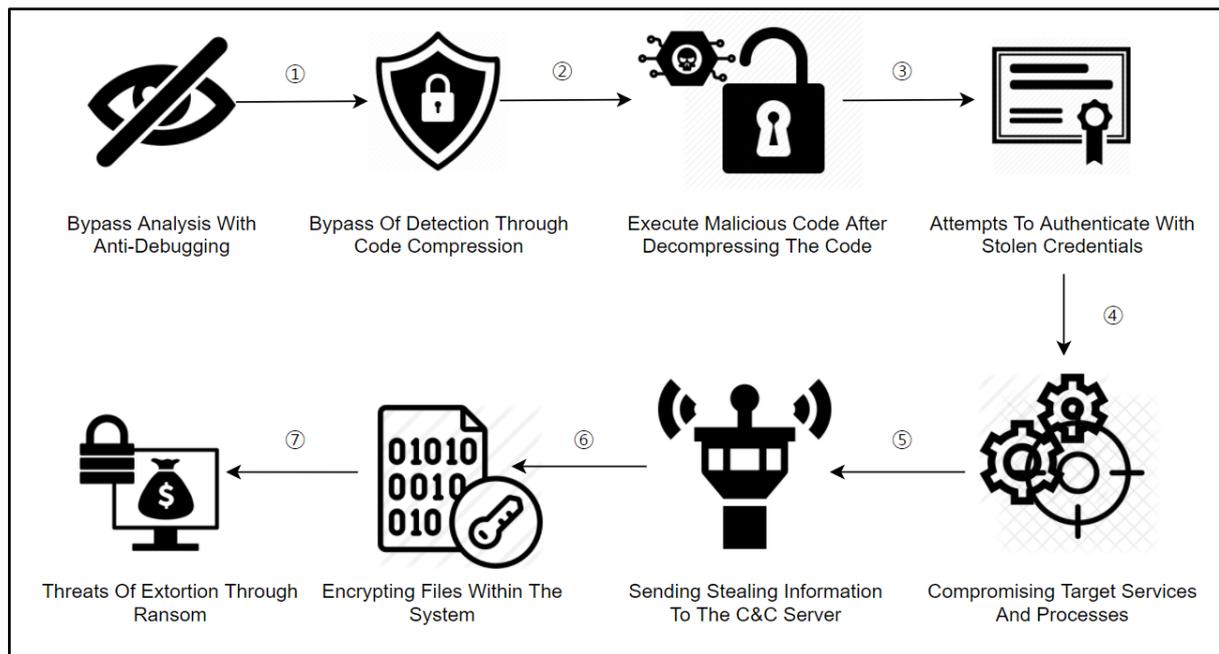
1.2 파일 정보

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

[파일 1] BlackMatter Ransomware EXE File

2. 분석

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

BlackMatter는 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.

00EC59BA	B8 5F01860A	mov eax,A86015F	
00EC59BF	35 ED5F0622	xor eax,22065FED	
00EC59C4	A3 2812ED00	mov dword ptr ds:[ED1228],eax	
00EC59C9	FF35 2812ED00	push dword ptr ds:[ED1228]	
00EC59CF	E8 49FEFFFF	call blackmatter.EC581D	
00EC59D4	A3 2812ED00	mov dword ptr ds:[ED1228],eax	
00EC59D9	8D1D 2C12ED00	lea ebx,dword ptr ds:[ED122C]	

00ED121C	20 32 78 75	70 32 78 75	A0 31 78 75	B2 5E 80 28	2xup2xu 1xu^A.(
00ED122C	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00ED123C	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00ED124C	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00ED125C	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00ED126C	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00

[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.

00EC5876	C745 FC 00000000	mov dword ptr ss:[ebp-4],0	
00EC587D	64:A1 30000000	mov eax,dword ptr fs:[30]	fs:[30]= PEB
00EC5883	8B40 0C	mov eax,dword ptr ds:[eax+C]	PEB_LDR_DATA
00EC5886	8D48 0C	lea ecx,dword ptr ds:[eax+C]	
00EC5889	894D F0	mov dword ptr ss:[ebp-10],ecx	
00EC588C	8B48 0C	mov ecx,dword ptr ds:[eax+C]	InLoadOrderModuleList
00EC588F	8B59 18	mov ebx,dword ptr ds:[ecx+18]	ImageBase
00EC5892	8B43 3C	mov eax,dword ptr ds:[ebx+3C]	Start_Of_PE_Header
00EC5895	03C3	add eax,ebx	
00EC5897	8B50 78	mov edx,dword ptr ds:[eax+78]	RVA_Of_Export_Dir

[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.

00EC59C9	FF35 2812ED00	push dword ptr ds:[ED1228]	
00EC59CF	E8 49FEFFFF	call blackmatter.EC581D	
00EC59D4	A3 2812ED00	mov dword ptr ds:[ED1228],eax	EAX = LoadLibraryw
00EC59D9	8D1D 2C12ED00	lea ebx,dword ptr ds:[ED122C]	
00EC59DF	803B 00	cmp byte ptr ds:[ebx],0	
00EC59E2	75 3B	jne blackmatter.EC5A1F	
00EC59E4	53	push ebx	
00EC59E5	E8 6ABCFFFF	call blackmatter.EC1654	

00ED11F8	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
00ED1208	00 00 00 00	00 00 00 00	00 00 00 00	D0 08 78 75D.xu	
00ED1218	50 F5 77 75	20 32 78 75	70 32 78 75	A0 31 78 75	Pöwu 2xup2xu 1xu	
00ED1228	C0 16 78 75	00 00 00 00	00 00 00 00	00 00 00 00	A.xu.....	
00ED1238	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	
00ED1248	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	

757816C0	8BFF	mov edi,edi	LoadLibraryw
757816C2	55	push ebp	
757816C3	8BEC	mov ebp,esp	
757816C5	5D	pop ebp	
757816C6	FF25 20127E75	jmp dword ptr ds:[&LoadLibraryw]	JMP.&LoadLibraryw
757816CC	CC	int3	

[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.

00EC5876	C745 FC 00000000	mov dword ptr ss:[ebp-4],0	
00EC587D	64:A1 30000000	mov eax,dword ptr fs:[30]	fs:[30]= PEB
00EC5883	8B40 0C	mov eax,dword ptr ds:[eax+C]	PEB_LDR_DATA
00EC5886	8D48 0C	lea ecx,dword ptr ds:[eax+C]	ecx:EntryPoint

010E0000	00 00 01 04	FF FF FF FF	00 00 EC 00	80 5D 6F 77	...yyyy..t..]ow	
010E0010	E0 17 4F 01	00 00 00 00	00 00 4F 01	40 5B 6F 77	a.O.....O.@[ow	
010E0020	00 00 00 00	00 00 00 00	01 00 00 00	70 13 C2 75p.Äu	
010E0030	00 00 00 00	00 00 00 00	00 00 DC 00	00 00 00 00ü.....	
010E0040	30 5D 6F 77	03 00 01 00	00 00 00 00	00 00 F1 7E	0]ow.....ñ~	
010E0050	00 00 00 00	50 07 F1 7E	00 00 07 7E	00 00 07 7E	...P.ñ~.....	
010E0060	28 00 0A 7E	04 00 00 00	70 00 00 00	00 00 00 00	(.....p.....	

010E0000	00 00 00 04	FF FF FF FF	00 00 EC 00	80 5D 6F 77	...yyyy..t..]ow	
010E0010	E0 17 4F 01	00 00 00 00	00 00 4F 01	40 5B 6F 77	a.O.....O.@[ow	
010E0020	00 00 00 00	00 00 00 00	01 00 00 00	70 13 C2 75p.Äu	
010E0030	00 00 00 00	00 00 00 00	00 00 DC 00	00 00 00 00ü.....	
010E0040	30 5D 6F 77	03 00 01 00	00 00 00 00	00 00 F1 7E	0]ow.....ñ~	
010E0050	00 00 00 00	50 07 F1 7E	00 00 07 7E	00 00 07 7E	...P.ñ~.....	
010E0060	28 00 0A 7E	04 00 00 00	00 00 00 00	00 00 00 00	(.....].	

[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.

00EC92F7	EB 03	jmp blackmatter.EC92FC	
00EC92F9	8B45 08	mov eax,dword ptr ss:[ebp+8]	
00EC92FC	6A 00	push 0	
00EC92FE	6A 00	push 0	
00EC9300	51	push ecx	
00EC9301	50	push eax	
00EC9302	FF15 2413ED00	call dword ptr ds:[ED1324]	0x0
00ED1324	0000	add byte ptr ds:[eax],al	
00ED1326	0000	add byte ptr ds:[eax],al	
00ED1328	0000	add byte ptr ds:[eax],al	

[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.

00EC92F7	EB 03	jmp blackmatter.EC92FC	
00EC92F9	8B45 08	mov eax,dword ptr ss:[ebp+8]	
00EC92FC	6A 00	push 0	
00EC92FE	6A 00	push 0	
00EC9300	51	push ecx	
00EC9301	50	push eax	
00EC9302	FF15 2413ED00	call dword ptr ds:[ED1324]	0x00FC3870
00FC3870	B8 3D496255	mov eax,5562493D	
00FC3875	35 ED5F0622	xor eax,22065FED	
00FC387A	FFE0	jmp eax	
00FC387C	0000	add byte ptr ds:[eax],al	

[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.

02883840	B8 3D496255	mov eax,5562493D	
02883845	35 ED5F0622	xor eax,22065FED	
0288384A	FFE0	jmp eax	NetSetInformationThread
0288384C	0000	add byte ptr ds:[eax],al	
0288384E	0000	add byte ptr ds:[eax],al	
02883850	094F C4	or dword ptr ds:[edi-3C],ecx	
02883853	1000	adc byte ptr ds:[eax],al	
007CF830	00EC9308	blackmatter.00EC9308로 반환 (종말: ???)	
007CF834	FFFFFFFE		
007CF838	00000011		
007CF83C	00000000		
007CF840	00000000		

[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

02884738	B8 7DD01055	mov eax,5510D07D	
0288473D	35 ED5F0622	xor eax,22065FED	
02884742	FF E0	jmp eax	SHTestTokenMember Shep
02884744	0000	add byte ptr ds:[eax],a1	
02884746	0000	add byte ptr ds:[eax],a1	

007CF83C	00EC9319	blackmatter.00EC9319로 반환 (출발: ???)	
007CF840	00000000		
007CF844	00000220		
007CF848	00ECD249	blackmatter.00ECD249로 반환 (출발: blackmatter.00EC930C)	

[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

00EC6258	C700 BE5F4922	mov dword ptr ds:[eax],22495FBE	
00EC625E	C740 04 AB5F5222	mov dword ptr ds:[eax+4],22525FAB	
00EC6265	C740 08 BA5F4722	mov dword ptr ds:[eax+8],22475FBA	

00EC62EF	C740 0C 885F4122	mov dword ptr ds:[eax+C],22415F88	
00EC62F6	C740 10 985F6F22	mov dword ptr ds:[eax+10],226F5F98	
00EC62FD	C740 14 895F0622	mov dword ptr ds:[eax+14],22065F89	
00EC6304	B9 06000000	mov ecx,6	
00EC6309	8130 ED5F0622	xor dword ptr ds:[eax],22065FED	

007CF764	53 00 4F 00 46 00 54 00 57 00 41 00 52 00 45 00	S.O.F.T.W.A.R.E.	
007CF774	5C 00 4D 00 69 00 63 00 72 00 6F 00 9E 5F 69 22	\.M.i.c.r.o.s.o.f.t	
007CF784	8B 5F 72 22 B1 5F 45 22 9F 5F 7F 22 9D 5F 72 22	._r"±_E"._"._r"	
007CF794	82 5F 61 22 9F 5F 67 22 9D 5F 6E 22 94 5F 06 22	._a"._q"._n"._."	

[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. 이후 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.

00EC6322	6A 00	push 0	
00EC6324	6A 00	push 0	
00EC6326	6A 00	push 0	
00EC6328	8D45 9C	lea eax,dword ptr ss:[ebp-64]	
00EC632B	50	push eax	
00EC632C	68 02000080	push 80000002	
00EC6331	FF15 3C14ED00	call dword ptr ds:[ED143C]	
028842B8	B8 CDBB0254	mov eax,5402BB8CD	
028842BD	35 ED5F0622	xor eax,22065FED	
028842C2	FFE0	jmp eax	RegCreateKeyExW
028842C4	0000	add byte ptr ds:[eax],al	

[그림 11] MachineGuid 경로 확인

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

00EC6358	6A 00	push 0			
00EC635A	8D45 DC	lea eax,dword ptr ss:[ebp-24]			
00EC635D	50	push eax			
00EC635E	FF75 FC	push dword ptr ss:[ebp-4]	eax:L"MachineGuid"		
00EC6361	FF15 4414ED00	call dword ptr ds:[ED1444]			
028841B0	B8 3DBF0254	mov eax,5402BF3D			
028841B5	35 ED5F0622	xor eax,22065FED			
028841BA	FFE0	jmp eax	RegQueryValueExW		
028841BC	0000	add byte ptr ds:[eax],al			
028841BE	0000	add byte ptr ds:[eax],al			
007CF6E4	28 00 64 00	64 00 33 00	30 00 34 00	34 00 37 00	8.d.d.3.0.4.4.7.
007CF6F4	2D 00 33 00	35 00 38 00	36 00 2D 00	34 00 64 00	-.3.5.8.6.-.4.d.
007CF704	62 00 37 00	2D 00 38 00	37 00 31 00	36 00 2D 00	b.7.-.8.7.1.6.-.
007CF714	34 00 36 00	61 00 36 00	39 00 38 00	66 00 32 00	4.6.a.6.9.8.f.2.
007CF724	38 00 31 00	62 00 32 00	00 00 00 00	78 4D 9F 22	8.1.b.2.....{M."

[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.

00EC1408	B8 494A4B4C	mov eax,4C4B4A49			
00EC140D	AB	stosd			
00EC140E	B8 4D4E4F50	mov eax,504F4E4D			
00EC1413	AB	stosd			
00EC1414	B8 51525354	mov eax,54535251			
00EC1419	AB	stosd			
00EC141A	B8 55565758	mov eax,58575655			
007CF7C2	7C 00 9A 64	EC 00 3C F8	7C 00 08 00	00 00 E2 F7	..d1.<0a÷
007CF7D2	7C 00 D5 E8	EC 00 D5 E8	EC 00 00 70	9A 00 00 00	.0èi.0èi..p....
007CF7E2	70 62 63 4E	58 56 30 4E	74 36 56 3D	00 00 00 00	pbCNXV0nt6V=....
00EC64B8	75 02	jne blackmatter.EC64BF			
00EC64BD	80 7A	mov al,7A	7A: 'z'		
00EC64BF	66:AB	stosw			
00EC64C1	E8 DD	jmp blackmatter.EC64A0			
00EC64C3	8BC3	mov eax,ebx	eax:L".pbCNXV0nt"		

[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.

00EC99A4	8D45 E4	lea eax,dword ptr ss:[ebp-1C]	
00EC99A7	C700 C85F7522	mov dword ptr ds:[eax],22755FC8	eax:L"%s.README.txt"
00EC99AD	C740 04 C35F5422	mov dword ptr ds:[eax+4],22545FC3	eax+4:L".README.txt"
00EC99B4	C740 08 A85F4722	mov dword ptr ds:[eax+8],22475FA8	eax+8:L".EADME.txt"
00EC99B8	C740 0C A95F4822	mov dword ptr ds:[eax+C],22485FA9	eax+C:L"DME.txt"
00EC99C2	C740 10 A85F2822	mov dword ptr ds:[eax+10],22285FA8	eax+10:L"E.txt"
00EC99C9	C740 14 995F7E22	mov dword ptr ds:[eax+14],227E5F99	eax+14:L"txt"
00EC99D0	C740 18 995F0622	mov dword ptr ds:[eax+18],22065F99	
00EC99D7	B9 07000000	mov ecx,7	ecx:L".pbcNXVONT"
00EC99DC	8130 ED5F0622	xor dword ptr ds:[eax],22065FED	eax:L"%s.README.txt"
00EC99E2	83C0 04	add eax,4	eax:L"%s.README.txt"
00EC99E5	49	dec ecx	ecx:L".pbcNXVONT"
00EC99E6	^ 75 F4	jne blackmatter.EC99DC	
00EC99E8	8B0D F40FED00	mov ecx,dword ptr ds:[ED0FF4]	ecx:L".pbcNXVONT", 00
00EC99EE	8D41 02	lea eax,dword ptr ds:[ecx+2]	eax:L"%s.README.txt"
00EC99F1	50	push eax	eax:L"%s.README.txt"
00EC99F2	8D45 E4	lea eax,dword ptr ss:[ebp-1C]	
00EC99F5	50	push eax	
00EC99F6	53	push ebx	
00EC99F7	FF15 F412ED00	call dword ptr ds:[ED12F4]	

[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.

00EC99F5	50	push eax	
00EC99F6	53	push ebx	
00EC99F7	FF15 F412ED00	call dword ptr ds:[ED12F4]	ebx:L"pbcNXVONT.README.txt"
00EC99FD	83C4 0C	add esp,C	
00EC9A00	6A FF	push FFFFFFFF	
00EC9A02	53	push ebx	ebx:L"pbcNXVONT.README.txt"

[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.

02883738	B8 3D2E6555	mov eax,55652E3D	
0288373D	35 ED5F0622	xor eax,22065FED	
02883742	▼ FFE0	jmp eax	RtlAdjustPrivilege
02883744	0000	add byte ptr ds:[eax],al	
007CF828	00EC954D	blackmatter.00EC954D로 반환 (출발: ???)	
007CF82C	00000011		
007CF830	00000001		
007CF828	00EC954D	blackmatter.00EC954D로 반환 (출발: ???)	
007CF82C	00000014		
007CF830	00000001		
007CF828	00EC954D	blackmatter.00EC954D로 반환 (출발: ???)	
007CF82C	00000024		
007CF830	00000001		

[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	FF E0	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		

028837C8	B8 DD476255	mov eax,556247DD	
028837CD	35 ED5F0622	xor eax,22065FED	
028837D2	FFE0	jmp eax	ZwQueryInformationToken
028837D4	0000	add byte ptr ds:[eax],a1	

007CF7F4	00EC94C4	blackmatter.00EC94C4로 반환 (출발: ???)	
007CF7F8	00000238		
007CF7FC	00000001		
007CF800	007CF810		
007CF804	0000002C		
007CF808	007CF83C		

[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.

02883870	B8 6D466255	mov eax,5562466D	
02883875	35 ED5F0622	xor eax,22065FED	
0288387A	FFE0	jmp eax	NtQuerySystemInformation
0288387C	0000	add byte ptr ds:[eax],a1	
0288387E	0000	add byte ptr ds:[eax],a1	
02883880	93	xchg ebx,eax	
02883881	BA 32052704	mov edx,4270532	

007CF80C	00EC7EDA	blackmatter.00EC7EDA로 반환 (출발: ???)	
007CF810	00000005		
007CF814	00C47A88		
007CF818	00000400		
007CF81C	007CF82C		

00C4BF80	53 00 79 00	73 00 74 00	65 00 6D 00	00 00 AD BA	S.y.s.t.e.m....°
00C4BF8C	A8 02 00 00	04 00 00 00	00 90 38 00	00 00 00 008.....
00C4BF90	FC 01 00 00	08 00 00 00	59 87 61 E9	01 00 00 00	ü.....Y.æ.....

[그림 18] 토큰 복제를 위한 프로세스 열거

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

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 C1o.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	ej*....\&....+....

[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, 0x0000D9F 값은 압축된 데이터의 사이즈를 의미한다.

00EC1718	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	ëiÿy....io.q*aaA
00ED2010	AD 37 EF A7 89 5F 95 3E	7A 1D 58 AE A1 F6 37 29	.7i\$....>z.x@i07)
00ED2020	E8 B1 CB 12 1F 7B 98 DE	64 BF E3 C2 EF F2 F3 39	ë±E..{.pd;AAio09
00ED2030	65 6A 2A B9 B0 A6 5C 26	F1 E2 8A 83 2B E0 EB B0	ej*':'\&ña..+aë°
00ED2040	AE A7 3B CC 86 77 0D C1	3A 5D AF 60 F6 B1 F7 B1	°§;I.w.Á:]_0±±±
00ED2050	14 82 04 DF 1A DB 3D 47	B9 DB 93 EB 8C A7 52 94	...B.0=G'U.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.

00EC11A8	8A16	mov d1,byte ptr ds:[esi]	esi:EntryPoint
00EC11AD	46	inc esi	esi:EntryPoint
00EC11AE	12D2	adc d1,d1	
00EC11B0	73 4F	jae blackmatter.EC1201	
00EC11B2	33C0	xor eax,eax	
00EC11B4	02D2	add d1,d1	
00EC11B6	75 05	jne blackmatter.EC11BD	
00EC11B8	8A16	mov d1,byte ptr ds:[esi]	esi:EntryPoint
00EC11BA	46	inc esi	esi:EntryPoint
00EC11BB	12D2	adc d1,d1	
00EC11BD	0F83 DB000000	jae blackmatter.EC129E	
00EC11C3	02D2	add d1,d1	
00EC11C5	75 05	jne blackmatter.EC11CC	
00EC11C7	8A16	mov d1,byte ptr ds:[esi]	esi:EntryPoint
00EC11C9	46	inc esi	esi:EntryPoint
00EC11CA	12D2	adc d1,d1	
00EC11CC	13C0	adc eax,eax	
00EC11CE	02D2	add d1,d1	
00EC11D0	75 05	jne blackmatter.EC11D7	
00EC11D2	8A16	mov d1,byte ptr ds:[esi]	esi:EntryPoint

00ED2000	EA A1 CA FF 9F 0D 00 00 ED 6F 8D 71 2A E0 E5 C1	ëjÿy...io.q*aaA
00ED2010	AD 37 EF A7 89 5F 95 3E 7A 1D 58 AE A1 F6 37 29	.7i§...>z.X®i07)
00ED2020	E8 B1 CB 12 1F 78 98 DE 64 BF E3 C2 EF F2 F3 39	è±È..{.pd;ãAio09
00ED2030	65 6A 2A B9 80 A6 5C 26 F1 E2 8A 83 28 E0 EB B0	ej*''\&ñâ..+ae'
00ED2040	AE A7 38 CC 86 77 0D C1 3A 5D AF 60 F6 B1 F7 B1	®§;i.w.A:} 0±±
00ED2050	14 82 04 DF 1A DB 3D 47 B9 DB 93 EB 8C A7 52 94	...B.O=G'Û.ë.SR.
00ED2060	98 03 DD F0 42 46 B1 FD 9C DD 1E 30 BF 56 83 A3	..Y0BF±ý.Y.0¿V.f
00ED2070	88 FE 78 92 21 96 13 07 4F 89 12 8E 83 4A C5 FB	.bx.!...0...JÀÛ
00ED2080	27 21 64 0F 08 CE D1 19 49 24 44 DF 44 CC EB 2F	'!d..IN.I\$D0D1e/
00ED2090	2F 01 6F 10 64 9F EF C2 7F 57 A3 DE C8 06 D2 A8	/o.d.iA.wEpE.0

[그림 25] aPLib 압축 라이브러리로 압축된 데이터

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

00729380	87 19 A8 30 F4 BA 94 94 92 91 58 2B 66 54 F9 6C	°.00°...X+FTuI
00729390	96 D9 A0 F4 41 9F 52 F3 67 CF 2E 19 B9 C9 5A 98	.Û ðA.R0gI..*EZ.
007293A0	70 91 CB EF AF BE 5A E3 9D AE 28 58 94 59 0A 8D	p.Ei'zâ.®(X.Y..
007293B0	88 87 64 E5 72 FA B5 23 46 46 F8 65 9A DA 2F BD	.darú#FFøe.Û/½
007293C0	8C 37 BF DD D6 07 97 A5 AD 9D AD 2D ED 37 96 9D	.7¿Y0°.¥....-i7..
007293D0	17 9E A4 AD 4C 19 80 D0 E7 08 05 62 41 D3 25 E1	..R.L.Đç..bA0%â
007293E0	88 EB 5C C4 92 5F A5 6A BF 81 0F 91 6E 79 32 D0	.è\A.¥j¿...ny2D
007293F0	16 A8 6E 3A D9 77 49 E7 5F 90 31 11 48 06 08 56	.n;UwTc.1.K..Y
00729400	51 24 78 C0 8D AD A2 AF 19 E4 98 08 FB DA 58 08	0\$xA..c.a..uUI.
00729410	A6 F3 30 80 9C D4 78 4F B9 21 4F 78 36 AA 46 AD	'00°.0!0'!0x6°F.
00729420	00 01 01 01 01 01 01 01 24 00 00 00 A1 00 00 00\$....i...
00729430	E2 00 00 00 00 00 00 00 F3 01 00 00 C8 04 00 00	â.....0...È...
00729440	39 05 00 00 32 06 00 00 1F 07 00 00 72 6F 34 42	9...2.....r04B
00729450	72 6E 58 35 5A 6D 73 31 66 6D 67 6D 70 39 48 79	rnX5Zms1fmgmp9Hy
00729460	70 69 30 68 43 67 50 64 75 4D 72 63 6C 57 55 49	pi0hCgPduMrc1WUI
00729470	71 30 35 4F 41 44 62 31 65 48 41 6D 65 7A 72 65	q050ADb1eHAmeszre
00729480	58 4A 49 34 36 72 66 58 62 45 4C 6A 73 7A 63 36	XJI46rfXbELjszc6
00729490	37 7A 74 69 49 72 72 55 4A 55 74 4D 6C 4F 4E 31	7ztIrrUJUtM10N1
007294A0	4C 73 41 37 70 75 48 4E 67 66 4B 4D 4F 41 76 4C	LsA7puHNgfKMOAvL
007294B0	55 70 54 6D 5A 6C 4E 59 61 63 37 47 4E 58 6E 77	UpTmZ1NYac7GNXnw
007294C0	42 77 41 41 41 41 42 3D 00 55 71 4C 53 67 68 57	BwAAAAB=-.UqL5ghW
007294D0	71 7A 49 59 33 57 5A 66 62 56 71 76 49 2F 4E 48	qzIY3WZfbvqvI/NH
007294E0	33 7A 73 69 62 43 51 63 35 39 61 59 36 77 67 44	3zsi bCQc59aY6wgD
007294F0	73 61 34 53 57 72 67 7A 77 4E 61 72 69 79 2B 52	sa45WrqzwNariy+r

[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.

007298F0	62 51 42 6C	41 48 41 41	62 77 42 6A	41 48 4D 41	bQB1AHAAbwBjAHMA
00729900	41 41 42 74	41 47 55 41	62 51 42 30	41 47 45 41	AABtAGUAbQB0AGEA
00729910	63 77 41 41	41 48 59 41	5A 51 42 6C	41 47 45 41	cwAAAHYAZQB1AGEA
00729920	62 51 41 41	41 48 4D 41	64 67 42 6A	41 43 51 41	bQAAAHMAdgBjACQA
00729930	41 41 42 69	41 47 45 41	59 77 42 72	41 48 55 41	AABiAGEAYwBrAHUA
00729940	63 41 41 41	41 48 4D 41	63 51 42 73	41 41 41 41	cAAAHMACQBsAAAA
00729950	64 67 42 7A	41 48 4D 41	41 41 41 41	41 4D 3D 3D	dgBzAHMAAAAAAM==
00729960	00 61 41 42	30 41 48 51	41 63 41 42	7A 41 44 6F	.aAB0AHQAcABzAD0
00729970	41 4C 77 41	76 41 48 41	41 59 51 42	35 41 47 30	ALWAVAHAAYQBSAGO

00C47DA8	6D 00 65 00	70 00 6F 00	63 00 73 00	00 00 6D 00	m.e.p.o.c.s...m.
00C47DB8	65 00 6D 00	74 00 61 00	73 00 00 00	76 00 65 00	e.m.t.a.s...v.e.
00C47DC8	65 00 61 00	6D 00 00 00	73 00 76 00	63 00 24 00	e.a.m...s.v.c.\$.
00C47DD8	00 00 62 00	61 00 63 00	68 00 75 00	70 00 00 00	.b.a.c.k.u.p...s
00C47DE8	73 00 71 00	6C 00 00 00	76 00 73 00	73 00 00 00	s.q.l...v.s.s...

[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.

028840D8	B8 4D680354	mov eax,5403684D	
028840DD	35 ED5F0622	xor eax,22065FED	
028840E2	FF E0	jmp eax	LogonUserW
028840E4	0000	add byte ptr ds:[eax],al	
028840E6	0000	add byte ptr ds:[eax],al	
028840E8	1E	push ds	

007CF718	00EC866C	blackmatter.00EC866C로 반환 (출발: ???)
007CF71C	007CF740	L"aheisler@hhcp.com"
007CF720	00000000	
007CF724	007CF764	L"120Heisler"
007CF728	00000004	
007CF72C	00000000	

007CF718	00EC866C	blackmatter.00EC866C로 반환 (출발: ???)
007CF71C	007CF740	L"dsmith@hhcp.com"
007CF720	00000000	
007CF724	007CF760	L"Tesla2019"
007CF728	00000004	
007CF72C	00000000	

007CF718	00EC866C	blackmatter.00EC866C로 반환 (출발: ???)
007CF71C	007CF740	L"administrator@hhcp.com"
007CF720	00000000	
007CF724	007CF76E	L"iteam8**"
007CF728	00000004	
007CF72C	00000000	

[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.EC10B8	
00ECE850	B9 FA424167	mov ecx,674142FA	
00ECE855	81F1 ED5F0622	xor ecx,22065FED	
00ECE85B	3BC1	cmp eax,ecx	
00ECE85D	▼ 75 0C	jne blackmatter.ECE868	
00ECE85F	FF73 08	push dword ptr ds:[ebx+8]	
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

00ECE850	B9 FA424167	mov ecx,674142FA	
00ECE855	81F1 ED5F0622	xor ecx,22065FED	
00ECE858	3BC1	cmp eax,ecx	eax:&L"C:\\User
00ECE85D	75 0C	jne blackmatter.ECE86B	
00ECE85F	FF73 08	push dword ptr ds:[ebx+8]	ebx+8:L"C:\\Use
00ECE862	E8 CBF7FFFF	call blackmatter.ECE032	

[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 뮈텍스를 통한 중복 실행 방지

2.1.14.1 뮈텍스 이름 생성

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

02884288	B8 CDBB0254	mov eax,54028BCD	
0288428D	35 ED5F0622	xor eax,22065FED	
028842C2	FFE0	jmp eax	RegCreateKeyExW
028842C4	0000	add byte ptr ds:[eax],al	
028842C6	0000	add byte ptr ds:[eax],al	
007CF608	00EC6337	blackmatter.00EC6337도 반환 (종말: ???)	
007CF60C	80000002		
007CF610	007CF6B0	L"SOFTWARE\\Microsoft\\Cryptography"	
007CF614	00000000		
02884180	B8 3DBF0254	mov eax,5402BF3D	
02884185	35 ED5F0622	xor eax,22065FED	
0288418A	FFE0	jmp eax	RegQueryValueExW
0288418C	0000	add byte ptr ds:[eax],al	
0288418E	0000	add byte ptr ds:[eax],al	
007CF614	00EC6367	blackmatter.00EC6367도 반환 (종말: ???)	
007CF618	00000238		
007CF61C	007CF6F0	L"MachineGuid"	
007CF620	00000000		

[그림 32] 뮈텍스를 위한 MachineGuid 획득

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

02884378	B8 DDA16A55	mov eax,556AA1DD			
0288437D	35 ED5F0622	xor eax,22065FED			
02884382	FFE0	jmp eax	MD4Init		
02884384	0000	add byte ptr ds:[eax],al			
02884386	0000	add byte ptr ds:[eax],al			
02884388	72 40	jb 28843CA			
02884360	B8 9DA16A55	mov eax,556AA19D			
02884365	35 ED5F0622	xor eax,22065FED			
0288436A	FFE0	jmp eax	MD4Update		
0288436C	0000	add byte ptr ds:[eax],al			
0288436E	0000	add byte ptr ds:[eax],al			
007CF720	D5 E8 EC 00	10 9D C4 00	44 F8 38 00	64 00 64 00	0e1...A.Dø8.d.d.
007CF730	33 00 30 00	34 00 34 00	37 00 2D 00	33 00 35 00	3.0.4.4.7.-.3.5.
007CF740	38 00 36 00	2D 00 34 00	64 00 62 00	37 00 2D 00	8.6.-.4.d.b.7.-.
007CF750	38 00 37 00	31 00 36 00	2D 00 34 00	36 00 61 00	8.7.1.6.-.4.6.a.
007CF760	36 00 39 00	38 00 66 00	32 00 38 00	31 00 62 00	6.9.8.f.2.8.1.b.
007CF770	32 00 00 00	61 00 6D 00	38 00 2A 00	C5 0E F6 87	2...a.m.8.*.A.ö.
007CF780	1D 1E 1F 20	A8 F7 7C 00	A8 F8 7C 00	90 99 64 77	... +dw
007CF790	F0 F7 7C 00	90 99 64 77	8D 45 E7 F0	FE FF FF FF	÷ ...dw.EçDbyÿÿ
007CF7A0	00 F8 7C 00	02 FD 7D 62	78 F3 E8 8E	37 5D 18 98	.ø .ÿ}bxøë.7]..
007CF7B0	81 99 22 50	00 02 00 00	00 00 00 00	00 00 00 00	..P.....
007CF7C0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
007CF7D0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
007CF7E0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
007CF7F0	00 00 00 00	00 00 00 00	00 00 00 00	02 FD 7D 62ÿ}b
007CF800	78 F3 E8 8E	37 5D 18 98	81 99 22 50	A7 46 E3 A2	xøë.7]...."P&F&c

[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 04EAF0FF	call blackmatter.EC5FEC	

00C4D690	47 00 6C 00	6F 00 62 00	61 00 6C 00	5C 00 36 00	G.1.o.b.a.1.\.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 GlobalW627dfd028ee8f3789b185d3750229981 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.00ECE69F도 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.

02883EF8	B8 4D707E57	mov eax,577E704D	
02883EFD	35 ED5F0622	xor eax,22065FED	
02883F02	FFE0	jmp eax	CreateMutexW
02883F04	0000	add byte ptr ds:[eax],al	
02883F06	0000	add byte ptr ds:[eax],al	
02883F08	E2 4F	loop 2883F59	
02883F0A	AE	scasb	

007CF808	00ECE6C3	blackmatter.00ECE6C3로 반환 (출발: ???)	
007CF80C	00000000		
007CF810	00000001		
007CF814	00C4D690	L"Globa1\\627dfd028ee8f3789b185d3750229981"	
007CF818	00C49D10	&L"C:\\Users\\JeongGeonWoo\\Desktop\\BlackMatter.exe"	
007CF81C	00000000		

[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.

02883C88	B8 FD507E57	mov eax,577E50FD	
02883CBD	35 ED5F0622	xor eax,22065FED	
02883CC2	FFE0	jmp eax	CreateThread
02883CC4	0000	add byte ptr ds:[eax],al	
02883CC6	0000	add byte ptr ds:[eax],al	
02883CC8	9A 4FF61000 0100	call far 1:10F64F	
02883CCF	8C	ret	
02883CD0	B8 3D6D7E57	mov eax,577E6D3D	

007CF7FC	00ECE6EC	blackmatter.00ECE6EC로 반환 (출발: ???)	
007CF800	00000000		
007CF804	00000000		
007CF808	00ECAEEE	blackmatter.00ECAEEE	
007CF80C	00000000		
007CF810	00000000		

[그림 37] 멀티 스레드 사용을 위한 스레드 생성

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

02883840	B8 3D496255	mov eax,5562493D	
02883845	35 ED5F0622	xor eax,22065FED	
0288384A	FFE0	jmp eax	NetSetInformationThread
0288384C	0000	add byte ptr ds:[eax],al	
0288384E	0000	add byte ptr ds:[eax],al	
02883850	094F C4	or dword ptr ds:[edi-3C],ecx	edi-3C:L"_info\":[r\n%s\r\n]"

02C9F90C	00EC87D0	blackmatter.00EC87D0로 반환 (출발: ???)	
02C9F910	00000230		
02C9F914	00000005		
02C9F918	02C9F93C		
02C9F91C	00000004		
02C9F920	00000000		
02C9F924	00000018		
02C9F928	00000000		

[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.

028843F0	B8 4D4C7E57	mov eax,577E4C4D	
028843F5	35 ED5F0622	xor eax,22065FED	
028843FA	FFE0	jmp eax	GetComputerNameW
028843FC	0000	add byte ptr ds:[eax],al	
028843FE	0000	add byte ptr ds:[eax],al	
02884400	8340 90 10	add dword ptr ds:[eax-70],10	
02884404	0023	add byte ptr ds:[ebx],ah	

[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.

007CF6CC	50 00 72 00	6F 00 76 00	69 00 64 00	65 00 72 00	P.r.o.v.i.d.e.r.
007CF6CD	41 00 72 00	63 00 68 00	69 00 74 00	65 00 63 00	A.r.c.h.i.t.e.c.
007CF6CE	74 00 75 00	72 00 65 00	00 00 00 00	53 00 45 00	t.u.r.e...S.E.
007CF6CF	4C 00 45 00	43 00 54 00	20 00 2A 00	20 00 46 00	L.E.C.T.*.F.
007CF6D0	52 00 4F 00	40 00 20 00	57 00 69 00	6E 00 33 00	R.O.M..w.i.n.3.
007CF6D1	32 00 5F 00	53 00 68 00	61 00 64 00	6F 00 77 00	2..S.h.a.d.o.w.
007CF6D2	43 00 6F 00	70 00 79 00	00 00 00 00	57 00 69 00	C.o.p.y...W.i.
007CF6D3	6E 00 33 00	32 00 5F 00	53 00 68 00	61 00 64 00	n.3.2..S.h.a.d.
007CF6D4	6F 00 77 00	43 00 6F 00	70 00 79 00	2E 00 49 00	o.w.C.o.p.y..I.
007CF6D5	44 00 3D 00	27 00 25 00	73 00 27 00	00 00 00 00	D.=.%S.....
007CF6D6	49 00 44 00	00 00 00 00	57 00 51 00	4C 00 00 00	I.D...W.Q.L...
007CF6D7	52 00 4F 00	4F 00 54 00	5C 00 43 00	49 00 4D 00	R.O.O.T.\.C.I.M.
007CF6D8	56 00 32 00	00 00 00 00	74 A6 AC 44	FC E8 D0 11	V.2.....t~0ueb.

[그림 44] COM 개체 사용을 위한 문자열

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

028846C0	B8 4D4CD04C	mov eax,4CD04C4D	
028846C5	35 ED5F0622	xor eax,22065FED	
028846CA	FFE0	jmp eax	CoCreateInstance
028846CC	0000	add byte ptr ds:[eax],al	
028846CE	0000	add byte ptr ds:[eax],al	
028846D0	D940 B9	fild st(0),dword ptr ds:[eax-47]	

007CF630	00EC708D	b\ackmatter.00EC708D로 반환 (출발: ???)
007CF634	007CF7A4	
007CF638	00000000	
007CF63C	00000001	

007CF7A4	98 66 4B 67	92 EE D0 11	AD 71 00 C0	4F D8 FD FF	.fkg.id..q.A00yy
007CF7A8	87 A6 12 DC	7F 73 CF 11	88 4D 00 AA	00 48 2E 24	..U.s.I..M..K.\$
007CF7AC	CC 55 85 CB	28 91 D1 11	AD 98 00 C0	4F D8 FD FF	I.U.E(.N...A00yy
007CF7B0	80 96 98 00	00 80 C1 2A	21 4E 62 FE	3E 62 EC 00A*!Nbp>b1.
007CF7B4	AF 99 5F 70	DC E8 39 00	80 96 98 00	00 00 00 00	..püe9.....

[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.

02883780	B8 7D486255	mov eax,5562487D	
02883785	35 ED5F0622	xor eax,22065FED	
0288378A	FF E0	jmp eax	NtQueryInformationProcess
0288378C	0000	add byte ptr ds:[eax],a1	

007CF618	00EC654D	blackmatter.00EC654D도 반환 (출발: ???)	
007CF61C	FFFFFFF		
007CF620	0000001A		
007CF624	007CF634		

[그림 46] WOW64 환경 확인

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

00EC70C7	50	push eax	
00EC70C8	6A 00	push 0	eax:L"__Provid
00EC70CA	8D85 B8FEFFFF	lea eax,dword ptr ss:[ebp-148]	
00EC70D0	50	push eax	eax:L"__Provid
00EC70D1	FF75 F4	push dword ptr ss:[ebp-C]	
00EC70D4	FF52 20	call dword ptr ds:[edx+20]	IWBEM_Context

007CF638	00C75BE0		
007CF63C	007CF6C8	L"__ProviderArchitecture"	
007CF640	00000000		
007CF644	007CF7D4		
007CF648	87F60FF9		

007CF7D4	03 00 00 00	00 00 00 00	40 00 00 00	00 00 00 00@.....
007CF7E4	AF 99 5F 70	DC E8 39 00	80 96 98 00	00 00 00 00	.._pUe9.....
007CF7F4	D5 E8 EC 00	D5 E8 EC 00	00 00 00 00	00 00 00 00	0e1.0e1.....

[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

00EC70FE	6A 00	push 0	
00EC7100	8D85 6CFFFFFF	lea eax,dword ptr ss:[ebp-94]	
00EC7106	50	push eax	eax:L"ROOT\CIMV2"
00EC7107	FF75 F8	push dword ptr ss:[ebp-8]	
00EC710A	FF52 0C	call dword ptr ds:[edx+C]	IWBEMLocator->ConnectServer

007CF624	00C67198		
007CF628	007CF77C	L"ROOT\CIMV2"	
007CF62C	00000000		
007CF630	00000000		

[그림 48] ROOTWCIMV2 네임스페이스에 대한 연결

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

00EC713F	6A 00	push 0	
00EC7141	6A 30	push 30	
00EC7143	8D85 E8FEFFFF	lea eax,dword ptr ss:[ebp-118]	
00EC7149	50	push eax	eax:L"WQL"
00EC714A	8D85 64FFFFFF	lea eax,dword ptr ss:[ebp-9C]	
00EC7150	50	push eax	eax:L"WQL"
00EC7151	FF75 F0	push dword ptr ss:[ebp-10]	
00EC7154	FF52 50	call dword ptr ds:[edx+50]	IWBEMLocator->ExecQuery

007CF630	00C7EA80		
007CF634	007CF774	L"WQL"	
007CF638	007CF6F8	L"SELECT * FROM Win32_ShadowCopy"	
007CF63C	00000030		
007CF640	00000000		

[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.

00EC7176	50	push eax	
00EC7177	8D45 E8	lea eax,dword ptr ss:[ebp-18]	
00EC717A	50	push eax	
00EC717B	6A 01	push 1	
00EC717D	6A FF	push FFFFFFFF	
00EC717F	FF75 EC	push dword ptr ss:[ebp-14]	
00EC7182	FF52 10	call dword ptr ds:[edx+10]	Enumerator

[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.

028841C8	B8 0DA30254	mov eax,5402A30D	
028841CD	35 ED5F0622	xor eax,22065FED	
028841D2	FF E0	jmp eax	OpenSCManagerw
028841D4	0000	add byte ptr ds:[eax],al	
028841D6	0000	add byte ptr ds:[eax],al	
028841D8	3840 D7	cmp byte ptr ds:[eax-29],al	

[그림 51] 서비스 제어 관리자 데이터 베이스 접근

OpenSCManagerW API 를 호출하여 서비스 제어 관리자 데이터베이스를 연다.

028843A8	B8 AD6E0354	mov eax,54036EAD	
028843AD	35 ED5F0622	xor eax,22065FED	
028843B2	FF E0	jmp eax	EnumServicesStatusExW
028843B4	0000	add byte ptr ds:[eax],al	
028843B6	0000	add byte ptr ds:[eax],al	
028843B8	74 40	je 28843FA	
028843BA	9B	fwait	

00CA5758	24 D5 CA 00	F6 D4 CA 00	20 00 00 00	01 00 00 00	50E.00E.
00CA5768	00 00 00 00	35 04 00 00	00 00 00 00	00 00 00 00	...5.....
00CA5778	00 00 00 00	00 00 00 00	00 00 00 00	EE D4 CA 00i0E.
00CA5788	AA D4 CA 00	10 00 00 00	01 00 00 00	00 00 00 00	00E.....
00CA5798	35 04 00 00	00 00 00 00	00 00 00 00	00 00 00 00	5.....
00CA57A8	00 00 00 00	00 00 00 00	98 D4 CA 00	6E D4 CA 000E.n0E.
00CA57B8	20 00 00 00	01 00 00 00	00 00 00 00	35 04 00 005...

00CAD524	41 00 4A 00	52 00 6F 00	75 00 74 00	65 00 72 00	A.J.R.o.u.t.e.r.
00CAD534	00 00 AB AB	AB AB AB AB	AB AB 00 00	00 00 00 00	..««««««.....
00CAD544	00 00 00 00	9C 13 74 8B	74 CC 00 00	CO 1D C8 00t.I.A.E.
00CAD554	28 A2 C8 00	EE FE EE FE	EE FE EE FE	EE FE EE FE	(cE.ipibipibipibip
00CAD564	EE FE EE FE	ipibipibipibipibip			

[그림 52] 서비스 제어 관리자 데이터베이스의 서비스 열거

EnumServicesStatusExW API 를 호출하여 서비스 제어 관리자 데이터베이스의 서비스를 열거한다.

00C47DA8	6D 00 65 00	70 00 6F 00	63 00 73 00	00 00 6D 00	m.e.p.o.c.s...m.
00C47DB8	65 00 6D 00	74 00 61 00	73 00 00 00	76 00 65 00	e.m.t.a.s...v.e.
00C47DC8	65 00 61 00	6D 00 00 00	73 00 76 00	63 00 24 00	e.a.m...s.v.c.\$.
00C47DD8	00 00 62 00	61 00 63 00	68 00 75 00	70 00 00 00	..b.a.c.k.u.p...
00C47DE8	73 00 71 00	6C 00 00 00	76 00 73 00	73 00 00 00	s.q.l...v.s.s...

028806D8	B8 DDC96255	mov eax,5562C9DD	
028806DD	35 ED5F0622	xor eax,22065FED	
028806E2	FF E0	jmp eax	wcsstr
028806E4	0000	add byte ptr ds:[eax],al	
028806E6	0000	add byte ptr ds:[eax],al	
028806E8	10BA 3A8E6504	adc byte ptr ds:[edx+4658E3A],bh	
028806EE	000CB8	add byte ptr ds:[eax+edi*4],cl	
028806F1	ED	in eax,dx	

007CF7BC	00EC738F	blackmatter.00EC738F로 반환 (출발: ???)
007CF7C0	00CAD524	L"ajrouter"
007CF7C4	00C47DA8	L"mepocs"
007CF7C8	007CF618	
007CF7CC	00000000	
007CF7D0	007CF810	

[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

02884138	B8 9D700054	mov eax,5400709D	
0288413D	35 ED5F0622	xor eax,22065FED	
02884142	FFE0	jmp eax	ControlService
02884144	0000	add byte ptr ds:[eax],al	
02884146	0000	add byte ptr ds:[eax],al	
02884148	2A40 E9	sub al,byte ptr ds:[eax-17]	
007CF7CC	00EC7325	blackmatter.00EC7325로 반환 (출발: ???)	
007CF7D0	00C85C60	00C85C60로 반환 (출발: 60E45D24)	
007CF7D4	00000001		
007CF7D8	007CF7E0		
007CF7DC	00ECE8D5	blackmatter.EntryPoint	
007CF7E0	00000000		
02884150	B8 CD670054	mov eax,540067CD	
02884155	35 ED5F0622	xor eax,22065FED	
0288415A	FFE0	jmp eax	DeleteService
0288415C	0000	add byte ptr ds:[eax],al	
0288415E	0000	add byte ptr ds:[eax],al	
02884160	2F	das	
02884161	40	inc eax	

[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.

77641980	B8 36000000	mov eax,36	NtQuerySystemInformation
77641985	BA 70716577	mov edx,ntdll.77657170	
7764198A	FFD2	call edx	
7764198C	C2 1000	ret 10	
7764198F	90	nop	
77641990	B8 37000000	mov eax,37	37: '7'
77641995	BA 70716577	mov edx,ntdll.77657170	
7764199A	FFD2	call edx	
7764199C	C2 0C00	ret c	
007CF7C8	00EC73ED	blackmatter.00EC73ED로 반환 (출발: ???)	
007CF7CC	00000005		
007CF7D0	00C64048		
007CF7D4	00000400		
007CF7D8	007CF808		
007CF7DC	007CF618		
007CF7E0	00000000		

[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.

028806D8	B8 DDC96255	mov eax,5562C9DD	
028806DD	35 ED5F0622	xor eax,22065FED	
028806E2	FFE0	jmp eax	wcsstr
028806E4	0000	add byte ptr ds:[eax],al	
007CF7B8	00EC74DC	blackmatter.00EC74DC로 반환 (출발: ???)	
007CF7BC	00CAE2C8	L"system"	
007CF7C0	00C47B70	L"encsvc"	

[그림 56] 종료 대상 프로세스 목록 비교

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

02883768	B8 6D476255	mov eax,5562476D	
0288376D	35 ED5F0622	xor eax,22065FED	
02883772	FFE0	jmp eax	NtOpenProcess
02883774	0000	add byte ptr ds:[eax],al	
02883776	0000	add byte ptr ds:[eax],al	
02883778	EC	in al,dx	
02883779	4E	dec esi	
02883FB8	B8 0D476255	mov eax,5562470D	
02883FBD	35 ED5F0622	xor eax,22065FED	
02883FC2	FFE0	jmp eax	ZwTerminateProcess
02883FC4	0000	add byte ptr ds:[eax],al	
02883FC6	0000	add byte ptr ds:[eax],al	
02883FC8	FA	cld	

[그림 57] 대상 프로세스 종료

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

00C47B70	65 00 6E 00	63 00 73 00	76 00 63 00	00 00 74 00	e.n.c.s.v.c...t.
00C47B80	68 00 65 00	62 00 61 00	74 00 00 00	6D 00 79 00	h.e.b.a.t...m.y.
00C47B90	64 00 65 00	73 00 6B 00	74 00 6F 00	70 00 71 00	d.e.s.k.t.o.p.q.
00C47BA0	6F 00 73 00	00 00 78 00	66 00 73 00	73 00 76 00	o.s...x.f.s.s.v.
00C47BB0	63 00 63 00	6F 00 6E 00	00 00 66 00	69 00 72 00	c.c.o.n...f.i.r.
00C47BC0	65 00 66 00	6F 00 78 00	00 00 69 00	6E 00 66 00	e.f.o.x...i.n.f.
00C47BD0	6F 00 70 00	61 00 74 00	68 00 00 00	77 00 69 00	o.p.a.t.h...w.i.
00C47BE0	6E 00 77 00	6F 00 72 00	64 00 00 00	73 00 74 00	n.w.o.r.d...s.t.
00C47BF0	65 00 61 00	6D 00 00 00	73 00 79 00	6E 00 63 00	e.a.m...s.y.n.c.
00C47C00	74 00 69 00	6D 00 65 00	00 00 6E 00	6F 00 74 00	t.i.m.e...n.o.t.
00C47C10	65 00 70 00	61 00 64 00	00 00 6F 00	63 00 6F 00	e.p.a.d...o.c.o.
00C47C20	6D 00 6D 00	00 00 6F 00	6E 00 65 00	6E 00 6F 00	m.m...o.n.e.n.o.
00C47C30	74 00 65 00	00 00 6D 00	73 00 70 00	75 00 62 00	t.e...m.s.p.u.b.
00C47C40	00 00 74 00	68 00 75 00	6E 00 64 00	65 00 72 00	.t.h.u.n.d.e.r.
00C47C50	62 00 69 00	72 00 64 00	00 00 61 00	67 00 6E 00	b.i.r.d...a.g.n.
00C47C60	74 00 73 00	76 00 63 00	00 00 73 00	71 00 6C 00	t.s.v.c...s.q.l.
00C47C70	00 00 65 00	78 00 63 00	65 00 6C 00	00 00 70 00	.e.x.c.e.l...p.
00C47C80	6F 00 77 00	65 00 72 00	70 00 6E 00	74 00 00 00	o.w.e.r.p.n.t...
00C47C90	6F 00 75 00	74 00 6C 00	6F 00 6F 00	68 00 00 00	o.u.t.l.o.o.k...
00C47CA0	77 00 6F 00	72 00 64 00	70 00 61 00	64 00 00 00	w.o.r.d.p.a.d...
00C47CB0	64 00 62 00	65 00 6E 00	67 00 35 00	30 00 00 00	d.b.e.n.g.5.0...
00C47CC0	69 00 73 00	71 00 6C 00	70 00 6C 00	75 00 73 00	i.s.q.l.p.l.u.s.
00C47CD0	73 00 76 00	63 00 00 00	73 00 71 00	62 00 63 00	s.v.c...s.q.b.c.
00C47CE0	6F 00 72 00	65 00 73 00	65 00 72 00	76 00 69 00	o.r.e.s.e.r.v.i.
00C47CF0	63 00 65 00	00 00 6F 00	72 00 61 00	63 00 6C 00	c.e...o.r.a.c.l.
00C47D00	65 00 00 00	6F 00 63 00	61 00 75 00	74 00 6F 00	e...o.c.a.u.t.o.
00C47D10	75 00 70 00	64 00 73 00	00 00 64 00	62 00 73 00	u.p.d.s...d.b.s.
00C47D20	6E 00 6D 00	70 00 00 00	6D 00 73 00	61 00 63 00	n.m.p...m.s.a.c.
00C47D30	63 00 65 00	73 00 73 00	00 00 74 00	62 00 69 00	c.e.s.s...t.b.i.
00C47D40	72 00 64 00	63 00 6F 00	6E 00 66 00	69 00 67 00	r.d.c.o.n.f.i.g.
00C47D50	00 00 6F 00	63 00 73 00	73 00 64 00	00 00 6D 00	.o.c.s.s.d...m.
00C47D60	79 00 64 00	65 00 73 00	68 00 74 00	6F 00 70 00	y.d.e.s.k.t.o.p.
00C47D70	73 00 65 00	72 00 76 00	69 00 63 00	65 00 00 00	s.e.r.v.i.c.e...
00C47D80	76 00 69 00	73 00 69 00	6F 00 00 00	00 00 00 00	v.i.s.i.o.....

[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
xfssvcon	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
mshpub	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.

621F7CF0	8BFF	mov edi,edi	RmStartSession
621F7CF2	55	push ebp	
621F7CF3	8BEC	mov ebp,esp	
621F7CF5	53	push ebx	
621F7CF6	56	push esi	esi:L"\\\\?\\C:

08F3F5E4	00E4B357	blackmatter.00E4B357로 반환 (출발: ???)	
08F3F5E8	08F3F65C		
08F3F5EC	00000000		
08F3F5F0	08F3F5FC		
08F3F5F4	006AC080	L"\\\\?\\C:	
08F3F5F8	00000000		

[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.

621F7A20	8BFF	mov edi,edi	RmRegisterResources
621F7A22	55	push ebp	
621F7A23	8BEC	mov ebp,esp	
621F7A25	8B4D 08	mov ecx,dword ptr ss:[ebp+8]	[ebp+8]:L"\\\\?\\C:\\
621F7A28	56	push esi	esi:L"\\\\?\\C:\\Tem

08F3F5D4	00E4B376	blackmatter.00E4B376로 반환 (출발: ???)	
08F3F5D8	00000000		
08F3F5DC	00000001		
08F3F5E0	08F3F66C	&L"\\\\?\\C:\\Temp\\	
08F3F5E4	00000000		

[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.

621F78B0	8BFF	mov edi,edi	RmGetList
621F78B2	55	push ebp	
621F78B3	8BEC	mov ebp,esp	
621F78B5	8B4D 08	mov ecx,dword ptr ss:[ebp+8]	[ebp+8]:L"\\
621F78B8	56	push esi	esi:L"\\\\?\\
621F78B9	E8 11FEFFFF	call rstrtmgr.621F76CF	
621F78BE	85C0	test eax,eax	

08F3F5DC	00E4B3C2	blackmatter.00E4B3C2로 반환 (출발: ???)	
08F3F5E0	00000000		
08F3F5E4	08F3F650		
08F3F5E8	08F3F654		
08F3F5EC	006CFA58		
08F3F5F0	08F3F64C		
08F3F5F4	006AC080	L"\\\\?\\C:	
08F3F5F8	00000000		

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

리소스에 등록된 파일을 사용하는 모든 프로세스 및 서비스 목록을 가져온다.

77221880	B8 26000000	mov eax,26	NtOpenProcess
77221885	BA 70712377	mov edx,ntdll.77237170	
7722188A	FFD2	call edx	
7722188C	C2 1000	ret 10	

08F3F49C	7517D818	kernelbase.7517D818로 반환 (출발: ???)	
08F3F4A0	08F3F4D0		
08F3F4A4	00120411		
08F3F4A8	08F3F4B0		
08F3F4AC	08F3F4C8		
08F3F4B0	00000018		
08F3F4B4	00000000		

08F3F4C8	64 1D 00 00	00 00 00 00	98 F4 F3 08	EC F4 F3 08	d.....00.100.
08F3F4D8	30 DA 20 62	11 04 12 00	00 00 00 00	64 1D 00 00	0ú b.....d...
08F3F4E8	D0 77 20 62	18 F5 F3 08	A0 BD 20 62	90 11 6C 00	Dw b.ó. % b..l.
08F3F4F8	00 00 00 00	18 85 6D 00	90 11 6C 00	CC 30 6D 00m...l.i0m.

svchost.exe	8064	2.53 MB	NT ...WLOCAL SERVICE	Host Pro
	2032	0.04 3.72 kB/s	8.04 MB	NT AUTHORITYW#SYSTE# PrivacyW
	7524	0.21 27.02 kB/s	45.17 MB	NT AUTHORITYW#SYSTE# Privacy-i
	1808		6.55 MB	NT AUTHORITYW#SYSTE# 콘솔 장
svchost.exe	9204	11.95 MB	NT AUTHORITYW#SYSTE#	Host Pro

[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.

772218E0	B8 2C000700	mov eax,7002C	ZwTerminateProcess
772218E5	BA 70712377	mov edx,ntdll.77237170	
772218EA	FFD2	call edx	
772218EC	C2 0800	ret 8	

772218E0	B8 2C000700	mov eax,7002C	ZwTerminateProcess
772218E5	BA 70712377	mov edx,ntdll.77237170	
772218EA	FFD2	call edx	
772218EC	C2 0800	ret 8	

svchost.exe	8064	2.53 MB	NT ...WLOCAL SERVICE	Host Proce
	2032	0.04 3.72 kB/s	8 MB	NT AUTHORITYW#SYSTE# PrivacyWal
	7524	0.31 27.41 kB/s	45.39 MB	NT AUTHORITYW#SYSTE# Privacy-i(S
	1808		6.55 MB	NT AUTHORITYW#SYSTE# 콘솔 장 호
svchost.exe	9204	11.95 MB	NT AUTHORITYW#SYSTE#	Host Proce

[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.

00ECA66C	53	push ebx	
00ECA66D	FF75 08	push dword ptr ss:[ebp+8]	[ebp+8]:"{r\n"
00ECA670	6A 10	push 10	
00ECA672	68 A0FED00	push blackmatter.ED0FA0	
00ECA677	E8 E1AFFFFF	call blackmatter.EC565D	

00ED0FA0	A6 F3 30 B0	9C D4 7B 4F	B9 21 4F 78	36 AA 46 AD	:ó°.ò{0!0x6^F.
00ED0FB0	00 01 01 01	01 01 01 01	18 7F C4 00	90 7F C4 00Á...Á.
00ED0FC0	88 7A C4 00	70 7B C4 00	A8 7D C4 00	60 E1 C4 00	.zÁ.p{Á. }Á. áÁ.
00ED0FD0	10 9D C4 00	40 E3 C4 00	00 00 00 00	00 00 00 00	..A.@ää.....
00ED0FE0	00 00 00 00	00 00 00 00	18 02 00 00	96 C1 EF 4EÄiN
00ED0FF0	0C 06 00 00	80 E4 C3 00	50 E1 C3 00	00 00 00 00ää.PáÁ.....
00ED1000	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00

[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.

00C56788	7B 0D 0A 22	62 6F 74 5F	76 65 72 73	69 6F 6E 22	!.."bot_ver1on"
00C56798	3A 22 31 2E	32 22 2C 0D	0A 22 62 6F	74 5F 69 64	: "1.2",.. "bot_id
00C567A8	22 3A 22 30	32 66 64 37	64 36 32 37	38 66 33 65	": "02fd7d6278f3e
00C567B8	38 38 65 33	37 35 64 31	38 39 62 38	31 39 39 32	88e375d189b81992
00C567C8	32 35 30 22	2C 0D 0A 22	62 6F 74 5F	63 6F 6D 70	250",.. "bot_comp
00C567D8	61 6E 79 22	3A 22 35 31	32 34 37 38	63 30 38 64	any": "512478c08d
00C567E8	61 64 61 32	61 66 31 39	65 34 39 38	30 38 66 62	ada2af19e49808fb
00C567F8	64 61 35 62	30 62 22 2C	0D 0A 22 68	6F 73 74 5F	da5b0b",.. "host_
00C56808	68 6F 73 74	6E 61 6D 65	22 3A 22 44	45 53 48 54	hostname": "DESKT

00C56788	8A 96 29 E3	78 75 A8 52	47 EB EA FB	E3 EF BE C3	.)äxu RGeëüäiäA
00C56798	EC 88 09 F2	8B BF 96 27	4E B6 98 C2	FC 32 A8 91	i..ò.¿.'Nj.Äü2 .
00C567A8	DE 07 F4 68	0C 3C 38 3F	60 C2 C4 F6	10 45 37 70	p.òh.<8?'AAö.E7p
00C567B8	EA 25 40 D0	73 0E 18 56	C4 09 D4 86	8C 78 B9 1D	è%èDs..VÁ.Ö..x'.
00C567C8	90 04 04 53	C3 1E AD EE	0E 0A CD 42	5F 2B E6 40	...SÄ..i..iB_+æ
00C567D8	23 AE E7 A9	A4 FE 0C 0D	B6 DC FE 3F	41 7C AA 5D	#èç@#b..jÜp?A ä
00C567E8	C5 E2 FC 87	FA E2 76 29	5D E1 D0 7D	5C 41 FA D6	Ääü.úäv)jäd}\ÄüÖ

00C50C68	69 70 59 70	34 33 68 31	71 46 4A 48	36 28 72 37	pYp43h1qFJH6+r7
00C50C78	34 28 2B 28	77 2B 79 49	43 66 4B 4C	76 35 59 6E	4+++w+yICfKLV5Yn
00C50C88	54 72 61 59	77 76 77 79	71 4A 48 65	42 2F 52 6F	TräYwvwyqJHeB/Ro
00C50C98	44 44 77 34	50 32 44 43	78 50 59 51	52 54 64 77	DDw4P2DCxPYQRTdw
00C50CA8	36 69 56 41	30 48 4D 4F	47 46 62 45	43 64 53 47	61VA0HMOGFbECd5G
00C50CB8	6A 48 69 35	48 5A 41 45	42 46 50 44	48 71 33 75	jHi5HZAEBFPDhQ3u
00C50CC8	44 67 72 4E	51 6C 38 72	35 68 41 6A	72 75 65 70	DgrNQ18r5kAjrupe
00C50CD8	70 50 34 4D	44 62 62 63	2F 6A 39 42	66 48 70 64	pP4MDbbc/j9BFKpd

[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.

02884978	B8 0D068656	mov eax,5686060D	
0288497D	35 ED5F0622	xor eax,22065FED	
02884982	FFE0	jmp eax	HttpOpenRequestw
02884984	0000	add byte ptr ds:[eax],al	
02884986	0000	add byte ptr ds:[eax],al	
02884988	3241 E3	xor al,byte ptr ds:[ecx-1D]	

02C9FB04	00ECA86E	blackmatter.00ECA86E로 반환 (출발: ???)	
02C9FB08	00CC0008		
02C9FB0C	02C9FC4C	L"POST"	
02C9FB10	00C4B3F0	L"/?78f=2R11QFMHS&Dox=ep56gU02&EmrI9=YpC985 eeLk&HHTDOsk=iZQ4B3css&1sT=IFTMIDSJ]	
02C9FB14	00000000		
02C9FB18	00000000		
02C9FB1C	00000000		

[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.

02883E80	B8 4D7D7E57	mov eax,577E7D4D	
02883E85	35 ED5F0622	xor eax,22065FED	
02883E8A	FFE0	jmp eax	CreateIoCompletionPort
02883E8C	0000	add byte ptr ds:[eax],al	
02883E8E	0000	add byte ptr ds:[eax],al	
02883E90	D14F BF	ror dword ptr ds:[edi-41],1	

0425F6E8	00ECBD1C	blackmatter.00ECBD1C로 반환 (출발: ???)	
0425F6EC	00000274		
0425F6F0	00000460		

[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.

02883D30	B8 0D687E57	mov eax,577E680D	
02883D35	35 ED5F0622	xor eax,22065FED	
02883D3A	FF E0	jmp eax	CopyFileW

0425F708	00EC9B25	blackmatter.00EC9B25로 반환 (출발: ???)	
0425F70C	00C7C1F8	L"\\\\?\\C:\\pbcNXV0nt.README.txt"	
0425F710	00C56828	L"\\\\?\\C:\\Users\\JeongGeonWoo\\pbcNXV0nt.README.txt"	

[그림 68] 랜섬 노트 복사

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

02884288	B8 AD1A0054	mov eax,54001AAD	
0288428D	35 ED5F0622	xor eax,22065FED	
02884292	FF E0	jmp eax	SetNamedSecurityInfoW

0425F704	00EC680E	blackmatter.00EC680E로 반환 (출발: ???)	
0425F708	00C76488	L"\\\\?\\C:\\Users\\JeongGeonWoo\\Desktop\\x64dbg\\snowman-v0.1.3-win-x86.7z"	
0425F70C	00000001		
0425F710	00000005		

[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.

02884060	B8 DD6A7E57	mov eax,577E6ADD	
02884065	35 ED5F0622	xor eax,22065FED	
0288406A	FF E0	jmp eax	SetFileAttributesW
0288406C	0000	add byte ptr ds:[eax],al	
0288406E	0000	add byte ptr ds:[eax],al	
02884070	0D 40831000	or eax,108340	

0425F6C4	00ECB9DD	blackmatter.00ECB9DD로 반환 (출발: ???)	
0425F6C8	00C76488	L"\\\\?\\C:\\Users\\JeongGeonWoo\\Desktop\\x64dbg\\snowman-v0.1.3-win-x86.7z"	
0425F6CC	00000080		
0425F6D0	00ECBF33	blackmatter.00ECBF33	
0425F6D4	00C76488	L"\\\\?\\C:\\Users\\JeongGeonWoo\\Desktop\\x64dbg\\snowman-v0.1.3-win-x86.7z"	
0425F6D8	00C30000		

[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.

02883F28	B8 AD6E7E57	mov eax,577E6EAD	
02883F2D	35 ED5F0622	xor eax,22065FED	
02883F32	FFE0	jmp eax	CreateFileW
02883F34	0000	add byte ptr ds:[eax],al	
02883F36	0000	add byte ptr ds:[eax],al	
02883F38	E4 4F	in al,4F	

0411FAA8	00ECBA9B	blackmatter.00ECBA9B로 반환 (출발: ???)	
0411FAAC	00C4F768	L"\\\\?\\C:\\Users\\JeongGeonWoo\\Searches\\Everywhere.search-ms"	
0411FAB0	80000000		
0411FAB4	00000000		
0411FAB8	00000000		

[그림 71] 암호화를 위한 파일 핸들 획득

CreateFileW API 를 이용하여 암호화 대상 파일의 핸들을 획득한다.

02883D18	B8 2D6B7E57	mov eax,577E6B2D	
02883D1D	35 ED5F0622	xor eax,22065FED	
02883D22	FFE0	jmp eax	ReadFile
02883D24	0000	add byte ptr ds:[eax],al	
02883D26	0000	add byte ptr ds:[eax],al	
02883D28	A6	cmpsb	
02883D29	4F	dec edi	
02883D2A	EA 10000500 8C88	jmp far B88C:50010	
02883D31	0D 687E5735	or eax,35577E68	
02883D36	ED	in eax,dx	
02883D37	5F	pop edi	
02883D38	06	push es	
02883D39	22FF	and bh,bh	

0411FACC	22 2F 3E 0D	0A 20 20 20	20 20 20 20	20 3C 2F 68	"/>.. </k
0411FADC	69 6E 64 4C	69 73 74 3E	0D 0A 20 20	20 20 20 20	indList>..
0411FAEC	20 20 3C 73	63 6F 70 65	20 68 6E 6F	77 6E 53 63	<scope knownSc
0411FAFC	6F 70 65 49	44 3D 22 78	37 63 36 31	64 30 61 36	opeID="{7c61d0a6
0411FB0C	2D 61 66 37	65 2D 34 38	33 61 2D 62	37 30 35 2D	-af7e-483a-b705-
0411FB1C	64 32 63 35	63 32 32 36	34 36 35 36	7D 22 20 2F	d2c5c2264656}" /
0411FB2C	3E 0D 0A 20	20 20 20 3C	2F 71 75 65	72 79 3E 0D	>.. </query>.
0411FB3C	0A 3C 2F 70	65 72 73 69	73 74 65 64	51 75 65 72	./persistedQuer
0411FB4C	79 3E 0D 0A	84 00 00 00	68 02 00 00	00 00 00 00	y>.....h.....

[그림 72] 암호화를 위한 버퍼 내 파일 내용 적재

ReadFile API 를 이용하여 암호화 대상 파일의 내용을 버퍼로 읽어들이는다.

02883E38	B8 5DC67157	mov eax,5771C65D	
02883E3D	35 ED5F0622	xor eax,22065FED	
02883E42	FFE0	jmp eax	MoveFileExW
02883E44	0000	add byte ptr ds:[eax],al	
02883E46	0000	add byte ptr ds:[eax],al	
02883E48	CA 4FC6	ret far C64F	

0411FB58	00ECBC80	blackmatter.00ECBC80로 반환 (출발: ???)	
0411FB5C	00C4F768	L"\\\\?\\C:\\Users\\JeongGeonWoo\\Searches\\Everywhere.search-ms"	
0411FB60	00C85A80	L"\\\\?\\C:\\Users\\JeongGeonWoo\\Searches\\Everywhere.search-ms.pbcNXVONT"	
0411FB64	00000008		
0411FB68	00ECBF33	blackmatter.00ECBF33	

[그림 73] 암호화 파일 확장자 변경

MoveFileExW API 를 이용하여 확장자명을 암호화 확장자명으로 변경시킨다.

00ECB8CB	83C4 0C	add esp,c	
00ECB8CE	68 100FED00	push blackmatter.ED0F10	RSA_key
00ECB8D3	8D46 70	lea eax,dword ptr ds:[esi+70]	
00ECB8D6	50	push eax	
00ECB8D7	E8 AB5BFFFF	call blackmatter.EC1787	
00ECB8DC	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	

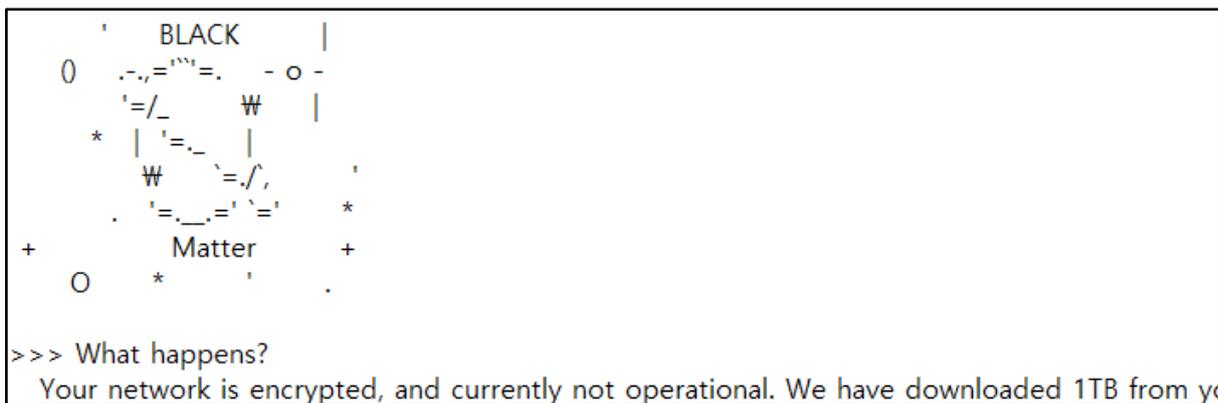
00ED0F10	87 19 A8 30 F4 BA 94 94 92 91 58 2B 66 54 F9 6C	.. 0o°...X+ftul
00ED0F20	96 D9 A0 F4 41 9F 52 F3 67 CF 2E 19 B9 C9 5A 9B	.ù òA.RògI..'ÉZ.
00ED0F30	70 91 CB EF AF BE 5A E3 9D AE 28 58 94 59 0A 8D	p.Éi %Zã.º(X.Y..
00ED0F40	B8 B7 64 E5 72 FA B5 23 46 46 F8 65 9A DA 2F BD	.dârú#FFoe.Ú/%
00ED0F50	8C 37 BF DD D6 07 97 A5 AD 9D AD 2D ED 37 96 9D	.7¿YÖ..¥...-i7..
00ED0F60	17 9E A4 AD 4C 19 80 D0 E7 08 05 62 41 D3 25 E1	..#..L..Đç..bA0%á
00ED0F70	8B EB 5C C4 92 5F A5 6A BF 81 0F 91 6E 79 32 D0	.è\A..¥j¿...ny2D
00ED0F80	16 A8 6E 3A D9 77 49 E7 5F 90 31 11 4B 06 0B 56	. n;UwIç_.1.K..V
00ED0F90	51 24 78 C0 8D AD A2 AF 19 E4 98 08 FB DA 5B 0B	Q\$xA..€ .ä..úÚ[.

[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.



[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

>	Low	Suspicious Behavior : evasion.enumerate.active-service.1
>	High	Suspicious Behavior : impact.encrypt.many-files
>	Medium	Suspicious Behavior : impact.encrypt.file.1
>	Medium	Suspicious Behavior : impact.impair.system-service.1
>	Low	Suspicious Behavior : discovery.acquire.account.1
>	Low	Suspicious Behavior : discovery.acquire.system-information.11
>	Low	Suspicious Behavior : discovery.enumerate.file-directory.1
>	Low	Suspicious Behavior : discovery.acquire.system-information.12

[그림 76] Privacy-i EDR 탐지 행위

3.2 주요 탐지 행위

3.2.1 evasion.enumerate.acvtive-service.1

▼	Low	Suspicious Behavior : evasion.enumerate.active-service.1
이벤트 발생 일시: 2021-08-13 14:36:11		
위험도: 2		

MITRE ATT&CK Information :		
No.	Tactic	Technique
1	Discovery	(T1007) System Service Discovery
2	Discovery	(T1497) Virtualization/Sandbox Evasion
3	Discovery	(T1497.001) System Checks
이벤트:		
No.	이벤트 종류	이벤트 이름
1	API	OpenSCManagerW

[그림 76] 서비스 열거

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

3.2.2 impact.impact.system-service.1

▼ Medium Suspicious Behavior : impact.impact.system-service.1

이벤트 발생 일시: 2021-08-13 14:30:09

위험도: 5

이벤트 Guid: 0781c7f1-61d4-4469-994b-731ac9384f19

MITRE ATT&CK Information :		
No.	Tactic	Technique
1	Privilege Escalation	(T1543) Create or Modify System Process
2	Privilege Escalation	(T1543.003) Windows Service

[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

▼ High Suspicious Behavior : impact.encrypt.many-files

이벤트 발생 일시: 2021-08-13 14:35:04

위험도: 10

이벤트 Guid: 313910c2-4831-4a47-b27f-d21eed9fbe61

[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.

본 자료의 전체 혹은 일부를 소만사의 허락을 받지 않고, 무단게재, 복사, 배포는 엄격히 금합니다.
만일 이를 어길 시에는 민형사상의 손해배상에 처해질 수 있습니다.
본 자료는 악성코드 분석을 위한 참조 자료로 활용 되어야 하며,
악성코드 제작 등의 용도로 악용되어서는 안됩니다.
(주) 소만사는 이러한 오남용에 대한 책임을 지지 않습니다.

Copyright(c) 2021 (주) 소만사 All rights reserved.

궁금하신 점이나 문의사항은 malware@somansa.com 으로 해주세요.