불법 다운로드 및 광고 사이트를 위장한 유포사이트 접속유도
윈도우 업데이트 위장한 랜섬웨어 파일 자동 다운로드
감염시 해커와 연락 및 데이터 복구 불가

Magniber 랜섬웨어

요약

1. 초반에는 **한국어 OS**에서만 작동 → **국내 사용자 집중피해** 발생
2. 초기 **불법 광고 및 위장 피싱 사이트**를 통해 유포
   → **Internet Explorer(IE) 취약점을 악용**하여 유포
3. IE 운영 종료 후에는 불법 다운로드 및 광고 사이트를 위장한
   유포사이트 접속을 유도한 후
   '윈도우 업데이트'로 위장한 랜섬웨어 파일을 자동 다운로드하여 데이터 암호화 진행
4. 감염 후 **해커와 연락 두절되며 데이터 복구조차 불가능**
5. 워너크라이보다 더 많은 피해를 입혔던 **'Cerber'의 변형 랜섬웨어**
6. 기업보다는 **학생 및 일반 사용자 대상으로 피해** 사례 발생

대응 방안

1. Privacy-i EDR과 같은 **EDR 솔루션의 '행위기반 탐지엔진'**으로 차단
   : 일반 Anti-Virus 솔루션에서도 대부분 차단 가능하나 최신 업데이트 필요
2. 악성코드 주요 감염경로인 **P2P, 음란, 도박 등 불법 웹사이트 연결 사전차단**
3. 메일 내용과 보내는이 계정에 연관성이 없거나, 문법적으로 어색하고,
   신뢰할 수 없는 링크 또는 첨부파일 클릭을 유도하는 메일은 실행 금지
4. **OS 및 소프트웨어 보안 업데이트** 최신형상으로 유지

목차

1. 개요

1.1 배경

매그니베르(Magniber) 랜섬웨어의 피해자들은
정상적인 윈도우 업데이트 파일로 위장한 설치파일 또는 제어판파일에 속아
해당 파일을 스스로 실행하여 감염되고 있으며,
감염 이후 해커와 연락이 닿지 않아 복원조차 불가능하여 심각한 피해가 초래되고 있다.

[그림 1] 윈도우 업데이트로 위장한 매그니베르(Magniber) 랜섬웨어 공격 사례

최근 감염 경로는 위 사진과 같이 위장된 광고 사이트로 접속하면
윈도우 업데이트를 위장한 제어판 파일로
매그니베르(Magniber) 랜섬웨어를 유포하는 도메인으로 리다이렉션 되도록 구성되어 있다.

그러나 7월 20일부터 기존 MSI 설치 파일 방식에서
CPL 형태의 제어판 파일 형태로 유포 형태가 변경되며
감염 방식과 윈도우 프로세스를 통한 암호화 진행 등 감염 양상이 바뀌었다.

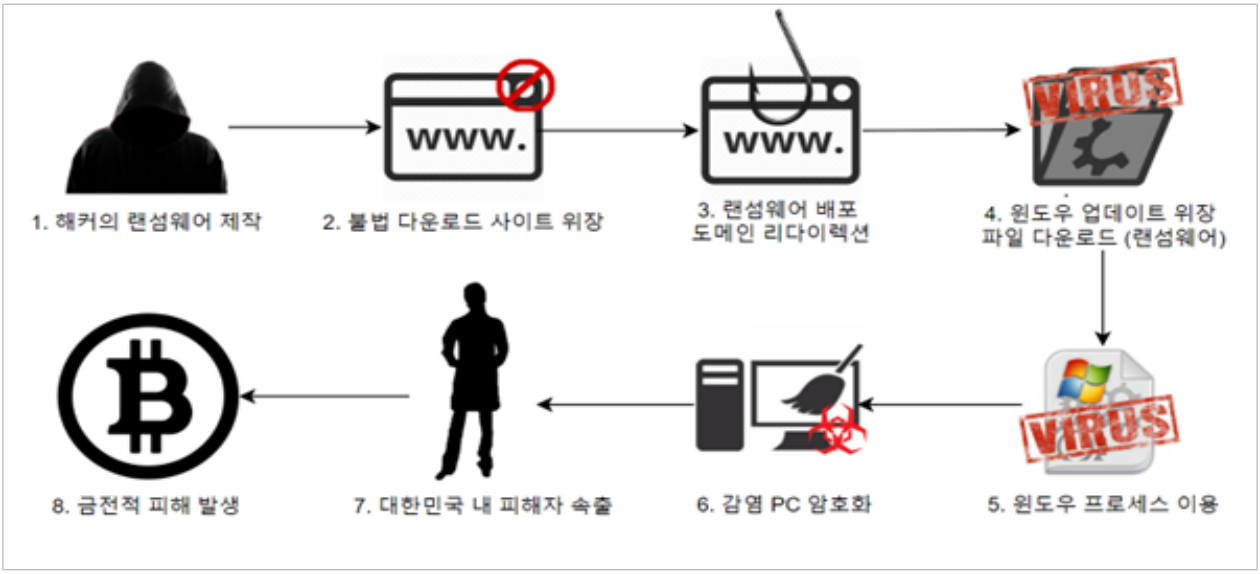
2. 분석

2.1 파일 정보

Name	MS.Upgrade.Database.Cloud.cpl
Type	Windows 실행 파일
Behavior	Ransomware
SHA-256	0e2ceee00815b899f750fd5013b3b839c6d62a946b6b305afc19dda85f6f6e52
Description	Magniber Ransomware

[표 1] 대한민국을 대상으로 무차별적인 공격을 수행하는 Magniber 변종

2.2 관리자 권한 여부 확인



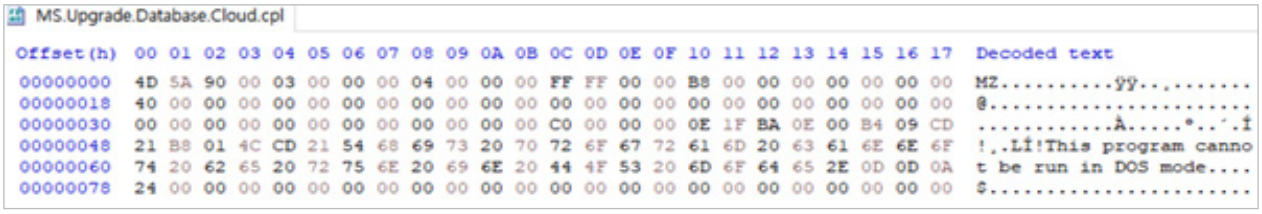
[그림 2] 매그니베르(Magniber) 랜섬웨어 공격 흐름

① ~ ②	해커의 랜섬웨어 제작 및 유포 해커는 랜섬웨어를 제작하고, 불법 다운로드 및 광고 사이트로 위장한 유포 사이트를 제작한다.
③	도메인 리다이렉션 불법 다운로드 및 광고 사이트로 위장한 도메인으로 접속한 피해자들은 랜섬웨어가 유포되는 도메인으로 리다이렉션 되어 접속하게 된다.

④ ~ ⑥	**파일 위장 및 윈도우 프로세스를 통한 감염** 윈도우 업데이트로 위장한 제어판 파일을 다운로드시키며, 이를 실행한 피해자들의 PC 내에서 정상 윈도우 프로세스를 이용해 랜섬웨어의 암호화 행위를 수행한다.
⑦ ~ ⑧	**대한민국 내 피해자 속출 및 금전적 피해 발생** 대한민국을 대상으로 감염을 유발하는 매그니베르(Magniber) 랜섬웨어로 인한 피해가 발생한다.

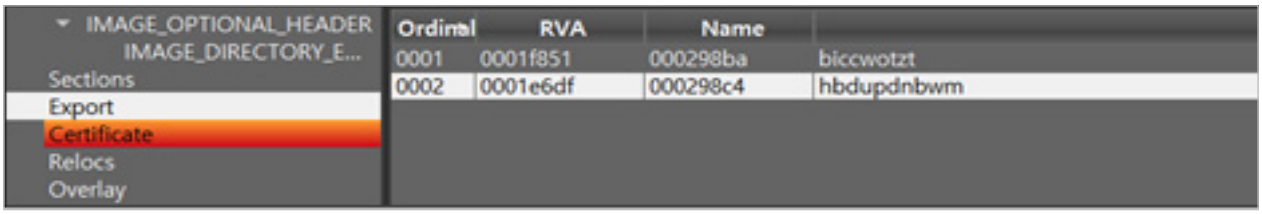
3. 분석

3.1 위장 및 탐지 우회



[그림 3] 매그니베르(Magniber) 랜섬웨어의 위장

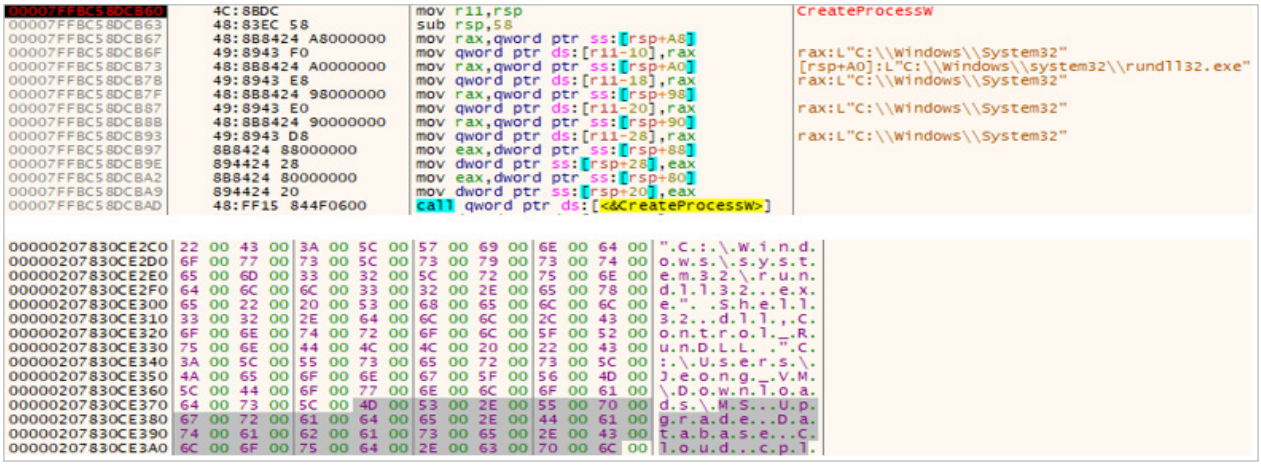
매그니베르(Magniber) 랜섬웨어는 리다이렉션 된 유포 도메인을 통해
피해자 PC에 다운로드 될 때 위와 같은 윈도우 업데이트를 위장한 제어판 파일의 형태로 다운로드된다.
이는 피해자가 의심없이 파일을 실행하도록 위장한 것이다.



[그림 4] 매그니베르(Magniber) 랜섬웨어의 탐지 우회

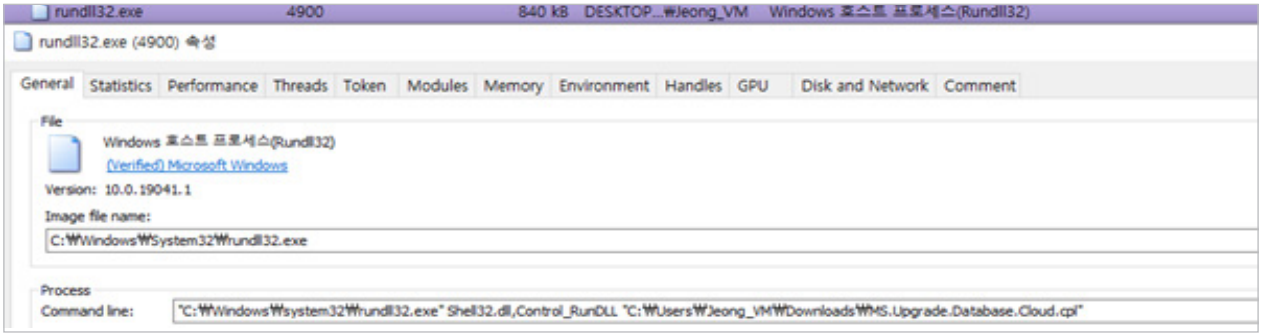
매그니베르(Magniber) 랜섬웨어의 Export Table을 확인하면
의미를 알 수 없는 Export 함수를 확인할 수 있는데,
이는 Export 함수를 통한 시그니처 기반 탐지를 우회하기 위함이다.
실제 매그니베르(Magniber)는 배포 시점마다 Export 함수명을 변경하여
Yara 등 시그니처 기반 탐지를 할 수 없도록 지속적인 변경을 통해 탐지를 우회하고 있다.

3.2 윈도우 프로세스를 통한 실행 (제어판, Rundll32)



[그림 5] 제어판을 통해 1차 실행을 수행한 매그니베르(Magniber) 랜섬웨어

CPL 형태로 위장한 매그니베르(Magniber) 랜섬웨어는 제어판(Control.exe)을 통해 실행되기 때문에 합법적인 윈도우 프로세스를 이용하여 시스템 내에서 실행된다.



[그림 6] Rundll32를 통해 2차 실행을 수행한 매그니베르(Magniber) 랜섬웨어

이전의 제어판을 통해 1차로 실행된 매그니베르(Magniber) 랜섬웨어는
이후 위처럼 DLL을 실행하는 정상 윈도우 유틸리티인 Rundll32를 통해 실행된다.
이는 제어판 프로그램의 실행 흐름을 악용한 것으로서,
실제 제어판 파일은 CPL 형태의 제어판 파일을 읽고 Rundll32를 통해 실행한다.
매그니베르(Magniber)의 공격 흐름은 이처럼 정상적인 윈도우 프로세스를 이용한 공격으로,
화이트 리스트 기반 탐지 또한 우회한다.

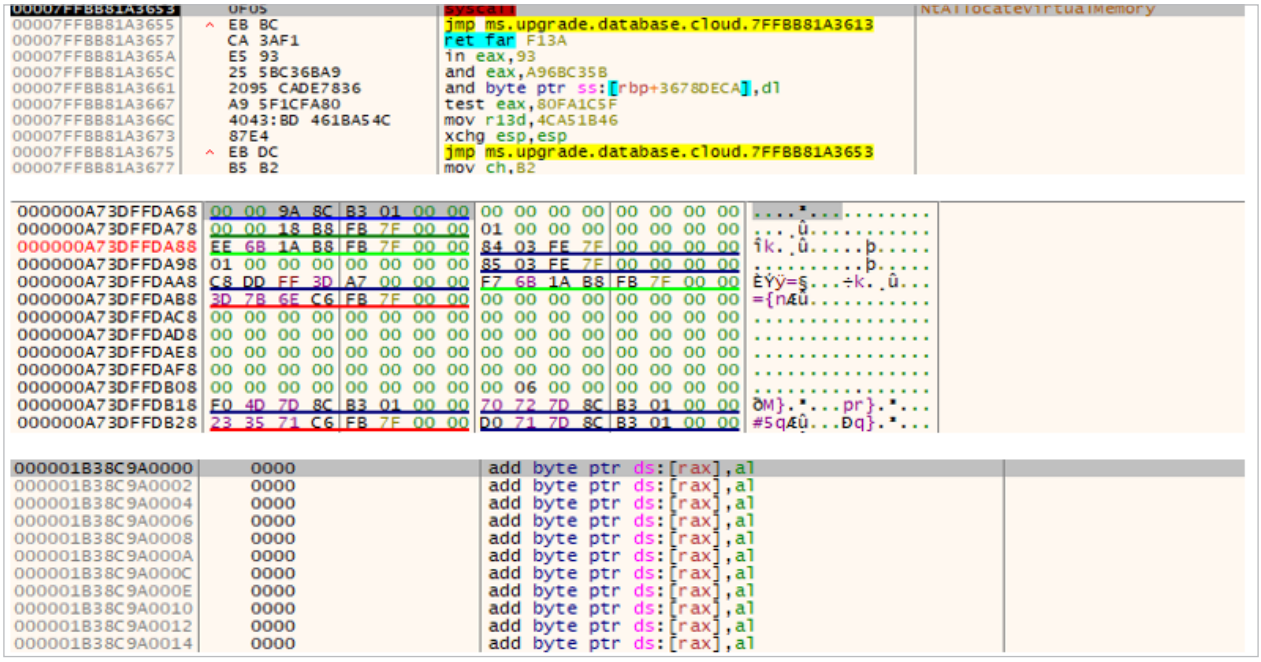
> 제어판 파일과 DLL이란?
> 제어판 파일은 CPL 확장자를 가지고 있지만, 내부적으로는 DLL의 형태로 제작되어 있다.
> 제어판 프로세스를 통해 실행이 되어, 최종적으로는 DLL을 실행하는 유틸리티인 Rundll32를 통해 실행된다. 매그니베르(Magniber) 랜섬웨어 제작자는 이 점을 이용하여 윈도우 프로세스를 이용하였다.

3.3 코드 난독화

[그림 7] 매그니베르(Magniber) 랜섬웨어의 난독화된 코드

매그니베르(Magniber) 랜섬웨어의 코드는 쉘코드와 어셈블리 코드를 이용한 난독화가 이루어져 있다.
실제 EntryPoint 이후의 코드는 연산을 통해 메모리를 할당하고,
할당된 메모리에 난독화 된 코드를 난독화 해제 후 적재하고 실행하는 형태가 반복된다.
이러한 코드 난독화는 코드의 패턴 등을 통해 악성코드를 판단하는 휴리스틱 기반의 안티바이러스를 우회하고
악성코드 분석가의 신속한 분석을 어렵게 하여 탐지와 분석에 어려움을 주기 위함이다.

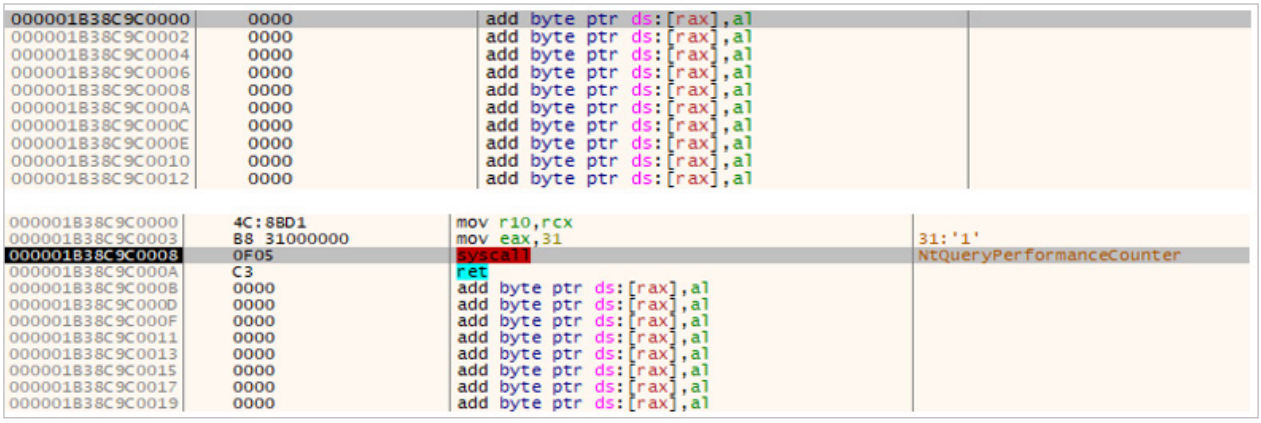
3.4 난독화 해제를 위한 메모리 할당



[그림 8] 코드 난독화 해제를 위한 메모리 할당

본 매그니베르(Magniber) 랜섬웨어는 시스템 콜을 직접 호출하는 형식으로 모든 행위가 수행된다.
위 사진 내 시스템 콜을 확인하면 난독화 된 코드가 실행되며
NTDLL의 NtAllocateVirtualMemory라는 API를 호출 후
임의의 공간에 메모리를 할당하는 것을 확인할 수 있다.
해당 공간은 추후 악성 행위에 필요한 API 들을 난독화 해제한 후 적재하여 호출하기 위해 할당한다.

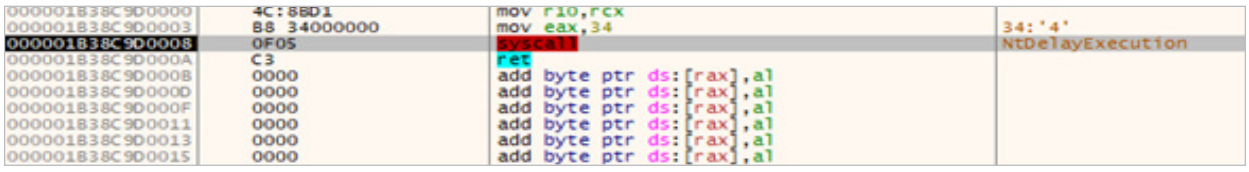
3.5 난독화 해제 후 코드 적재



[그림 9] 난독화 해제 및 코드 적재

위 사진의 메모리 주소 0x000001B38C9C0000를 확인하면
공간이 할당된 시점에는 코드가 없는 빈 메모리 공간이지만,
난독화 해제 후 악성 행위에 필요한 Native API를 호출할 시점이 되면 코드가 적재되고,
이를 실행하는 모습을 확인할 수 있다.
매그니베르(Magniber) 랜섬웨어는 이처럼 모든 코드를 특정 시점에 난독화를 해제하며 실행하고,
실행이 끝나면 코드를 지워버리는 형태로 공격을 수행한다.

3.6 샌드박스 기반 탐지 우회



[그림 10] 유휴 시간을 통한 샌드박스 기반 탐지 우회

매그니베르(Magniber) 랜섬웨어는 실행 중 NtDelayExecution Native API를 호출하여 유휴 시간을 준다.
이는 샌드박스 등에서 프로세스의 실행을 추적하는 방식을 우회하기 위한 것으로
일정 시간동안 프로세스의 움직임을 멈추어 프로세스가 미동작하는 것으로 위장하는 방식이다.

3.7 개체 디렉토리 할당

```
000001B38C9E0000  4C:8BD1          mov r10,rcx
000001B38C9E0003  B8 19000000      mov eax,19
000001B38C9E0008  0F05             syscall                    NtQueryInformationProcess
000001B38C9E000A  C3               ret
000001B38C9E000B  0000             add byte ptr ds:[rax],al
000001B38C9E000D  0000             add byte ptr ds:[rax],al
000001B38C9E000F  0000             add byte ptr ds:[rax],al
000001B38C9E0011  0000             add byte ptr ds:[rax],al
000001B38C9E0013  0000             add byte ptr ds:[rax],al
000001B38C9E0015  0000             add byte ptr ds:[rax],al

1: rcx FFFFFFFFFFFFFFFF
2: rdx 0000000000000017
3: r8 000000A73DFFD6A0
4: r9 0000000000000024
5: [rsp+28] 0000000000000000
```

[그림 11] 프로세스 당 개체 디렉토리 할당을 위한 Native API 호출

일반적인 방법으로 볼륨에 대한 개체 디렉토리를 할당하는 것이 아닌,
프로세스 디바이스 맵을 통해 볼륨에 대한 개체 디렉토리를 할당한다.
위 사진의 NtQueryInformationProcess Native API 호출 시 두번째 인자로 입력된
0x17(ProcessDeviceMap)은 Rundll32 내 매그니베르(Magniber) 랜섬웨어에
DosDevice 개체 디렉토리를 쿼리하고 추후 연결시켜주는 작업에 사용된다.

3.8 DosDevice 열기 시도

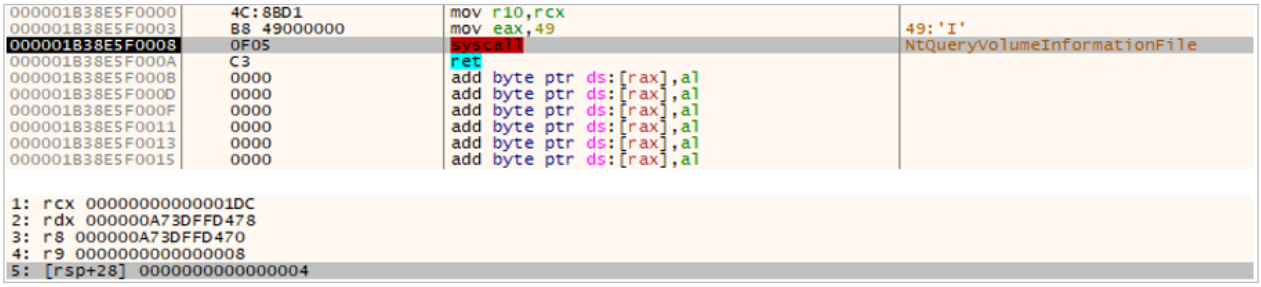
```
000001B38E5E0000  4C:8BD1          mov r10,rcx
000001B38E5E0003  B8 33000000      mov eax,33                 33:'3'
000001B38E5E0008  0F05             syscall                    NtOpenFile
000001B38E5E000A  C3               ret
000001B38E5E000B  0000             add byte ptr ds:[rax],al
000001B38E5E000D  0000             add byte ptr ds:[rax],al
000001B38E5E000F  0000             add byte ptr ds:[rax],al
000001B38E5E0011  0000             add byte ptr ds:[rax],al
000001B38E5E0013  0000             add byte ptr ds:[rax],al
000001B38E5E0015  0000             add byte ptr ds:[rax],al

000001B38C9A4902  5C 00 3F 00 3F 00 5C 00 00 00 43 00 3A 00 5C 00  \.?.?.\...C.:.\.
000001B38C9A4912  00 00 30 00 31 00 32 00 33 00 34 00 35 00 36 00  ..0.1.2.3.4.5.6.
000001B38C9A4922  37 00 38 00 39 00 61 00 62 00 63 00 64 00 65 00  7.8.9.a.b.c.d.e.
000001B38C9A4932  66 00 00 00 30 31 32 33 34 35 36 37 38 39 61 62  f...0123456789ab
000001B38C9A4942  63 64 65 66 00 30 00 31 00 32 00 33 00 34 00 35  cdef.0.1.2.3.4.5
000001B38C9A4952  00 36 00 37 00 38 00 39 00 00 00 68 00 69 00 6A  .6.7.8.9...h.i.j
000001B38C9A4962  00 68 00 76 00 6F 00 79 00 00 00 68 69 6A 68 76  .h.v.o.y...hijhv
```

[그림 12] DosDevice 열기 시도

NtOpenFile Native API를 호출하여 \\??\C:\로 표기되는 DosDevice 경로를 여는 것을 시도한다.
이는 이전의 개체 디렉토리 할당의 결과로서, Rundll32 내 매그니베르(Magniber) 랜섬웨어에
개체 디렉토리가 할당되어 있는지를 확인하는 작업이며
추후 얻을 장치 정보를 얻기 위한 핸들을 얻는 작업이다.

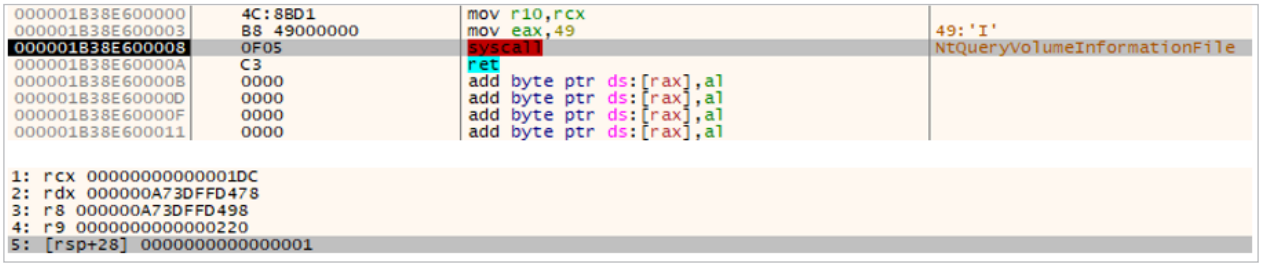
3.9 파일 시스템 정보 획득



[그림 13] 파일 시스템 정보 획득

DosDevice에 대한 핸들을 통해 볼륨 정보를 획득한다.
NtQueryVolumeInformationFile이란 Native API를 사용하며,
인자로 0x4(FileFsDeviceInformation)를 적재한다.
이는 향후 암호화를 수행할 파일 시스템 볼륨과 관련된 장치 정보를 쿼리하는 데 사용된다.

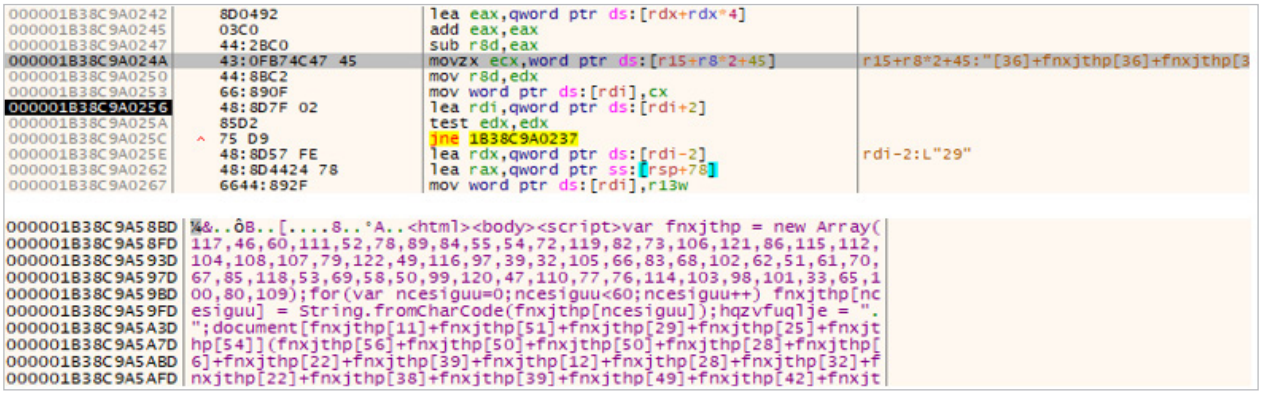
3.10 볼륨 정보 획득



[그림 14] 볼륨 정보 획득

파일 시스템 정보를 획득했던 것과 마찬가지로, 향후 암호화를 수행할 볼륨에 대한 정보를 획득한다.
이 때 사용되는 Native API는 위와 같이 NtQueryVolumeInformationFile이 사용된다.
인자로는 위 사진과 같이 0x1(FileFsVolumeInformation)가 사용되는데,
이는 파일 시스템이 마운트 된 볼륨에 대한 정보를 조회하는 데 사용된다.

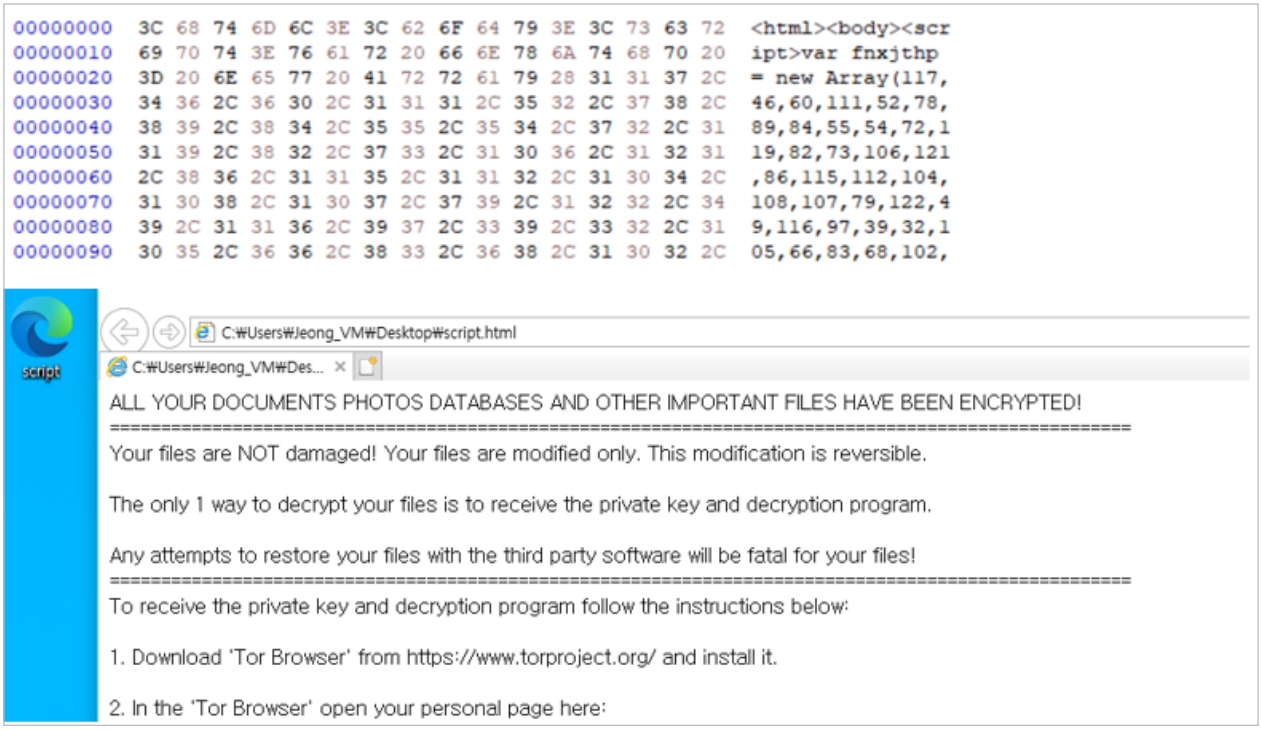
3.11.1 랜섬노트 제작 (내부 Script 실행)



[그림 15] 매그니베르(Magniber) 랜섬웨어 내부의 스크립트

랜섬 행위 수행에 앞서, 매그니베르(Magniber) 랜섬웨어 내부에 있는 스크립트를 실행한다.
해당 스크립트 내부에는 HTML 내 JavaScript가 난독화 된 후 삽입되어 있다.

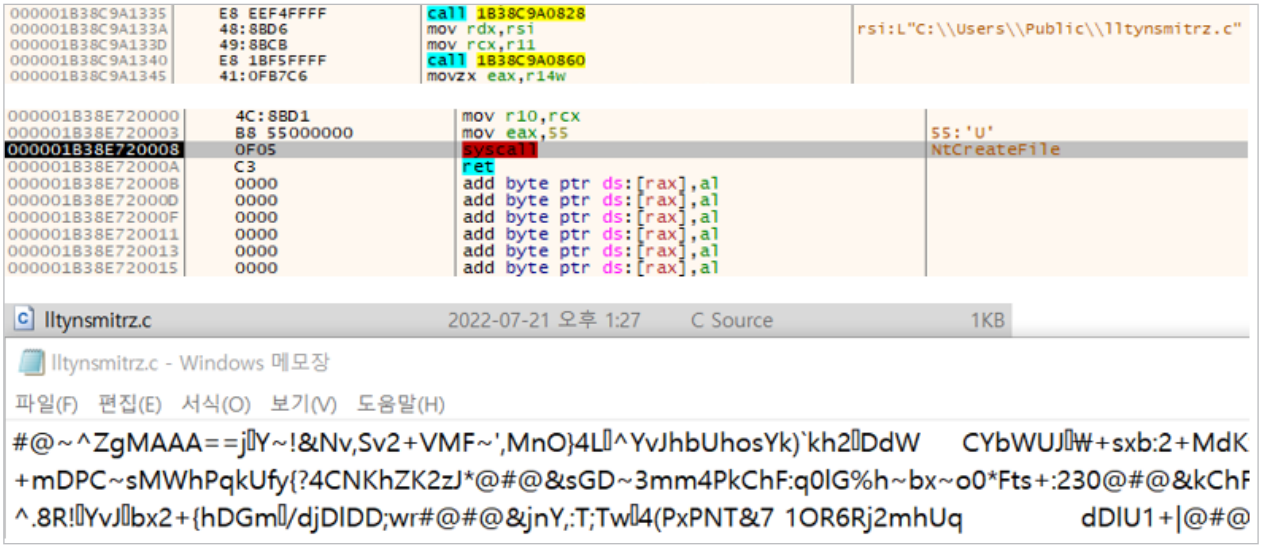
3.11.2 랜섬노트 제작 (내부 Script 기재)



[그림 16] 매그니베르(Magniber) 랜섬웨어 내부의 랜섬노트

HTML 내 난독화 된 JavaScript는 랜섬노트이다.
랜섬노트를 난독화 된 JavaScript 형태로 만든 이유는 메모리 내 문자열기반 탐지를 우회하기 위함이다.
랜섬노트를 생성하는 시점에 메모리 내 문자 형태를
시그니처로 탐지하는 샌드박스를 우회할 수 있도록 제작하였다.

3.12 키 파일 생성 (키 시드)



[그림 17] 키 생성 전 시드 파일 생성

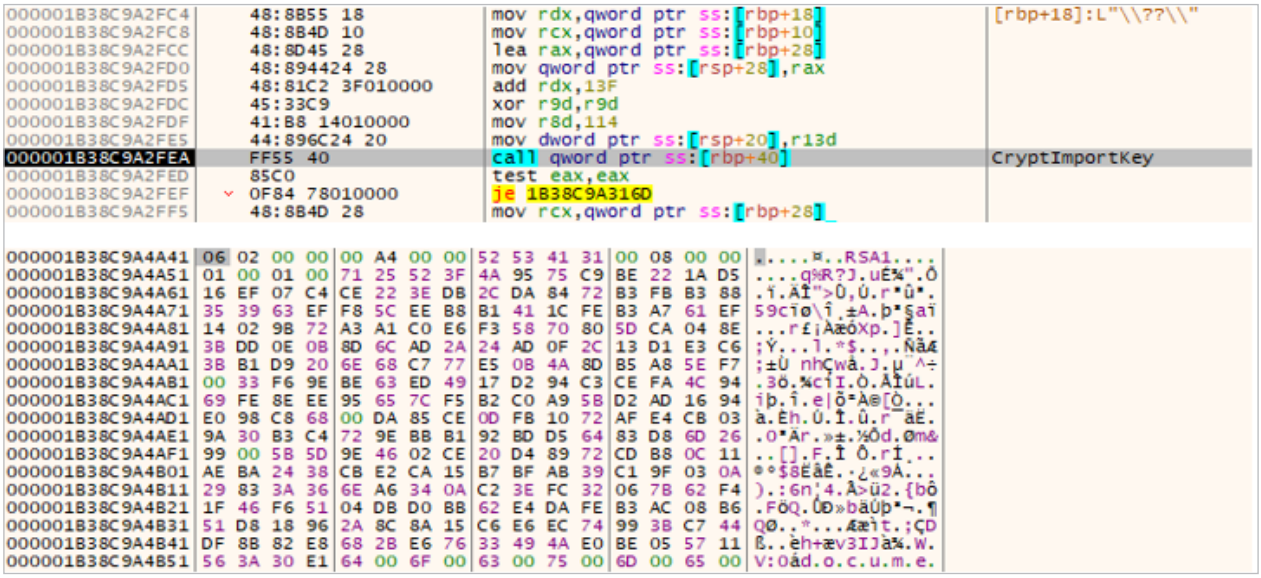
C:\Users\Public 경로에 임의의 파일명으로 키 생성을 위한 시드 파일을 생성한다.
해당 파일의 확장자는 .C로 제작되어 있으며, 내부에는 공격자가 제작한 시드 정보가 담겨있다.

3.13 키 파일 생성 (암호화 키 생성)

[그림 18] 암호화 키 생성 과정

시드 파일을 NtReadFile Native API로 읽은 후, CryptEncrypt API를 호출하여 시드의 암호화를 수행한다. 본 과정 내에서는 RSA1 방식이 사용되었으며, 암호화의 결과로 생성된 키는 NtWriteFile Native API를 사용하여 동일한 경로의 기존 파일에 덮어씌운다.

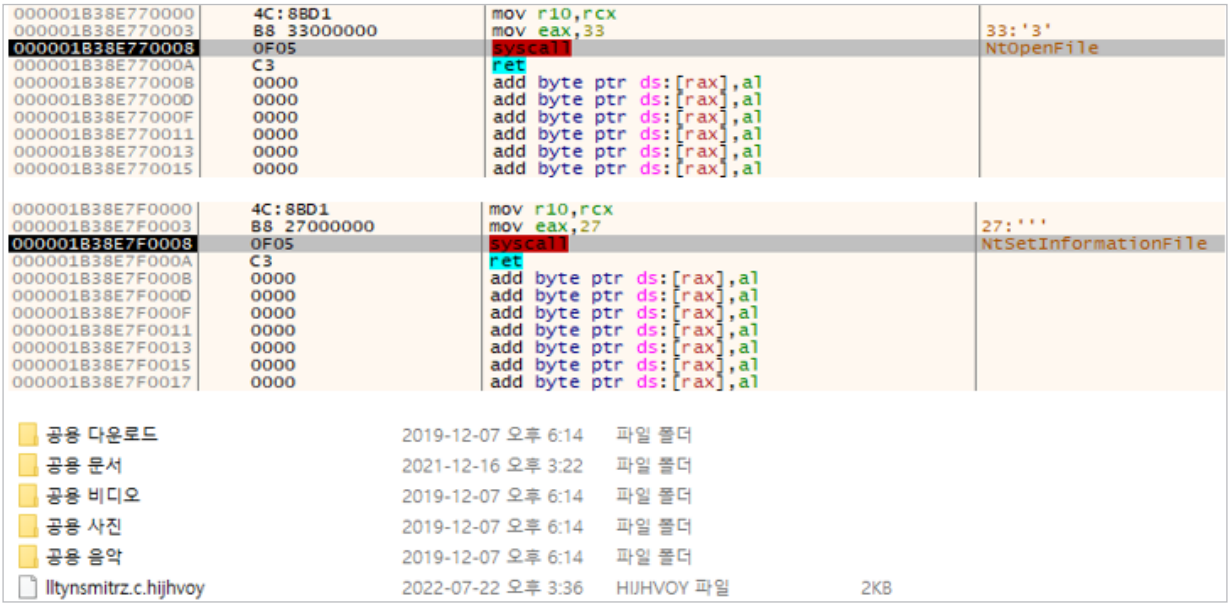
3.14 암호화 키 임포트



[그림 19] 암호화 키 임포트

CryptImportKey API를 호출하여 이전에 만든 키를 Import 한 후 메모리에 적재한다.
해당 키는 파일들의 암호화 행위에 사용된다.

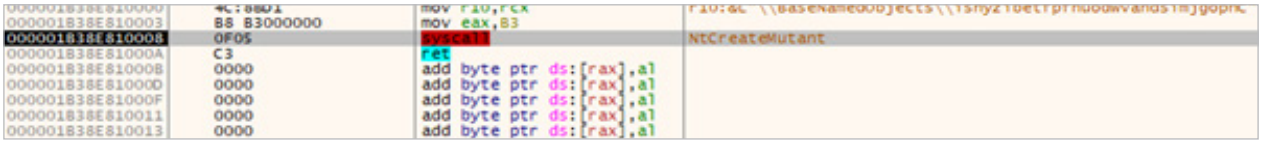
3.15 암호화 키 파일 암호화



[그림 20] 키 파일 암호화

이전에 생성하였던 키 파일을 암호화의 첫 단계로 암호화를 수행한다.
키 파일을 암호화함으로써, 복호화를 수행할 수 있는 단서를 제거한다.
키 파일은 강력한 암호화 알고리즘인 RSA1으로 암호화되어 복구 불가능한 상태로 변경된다.

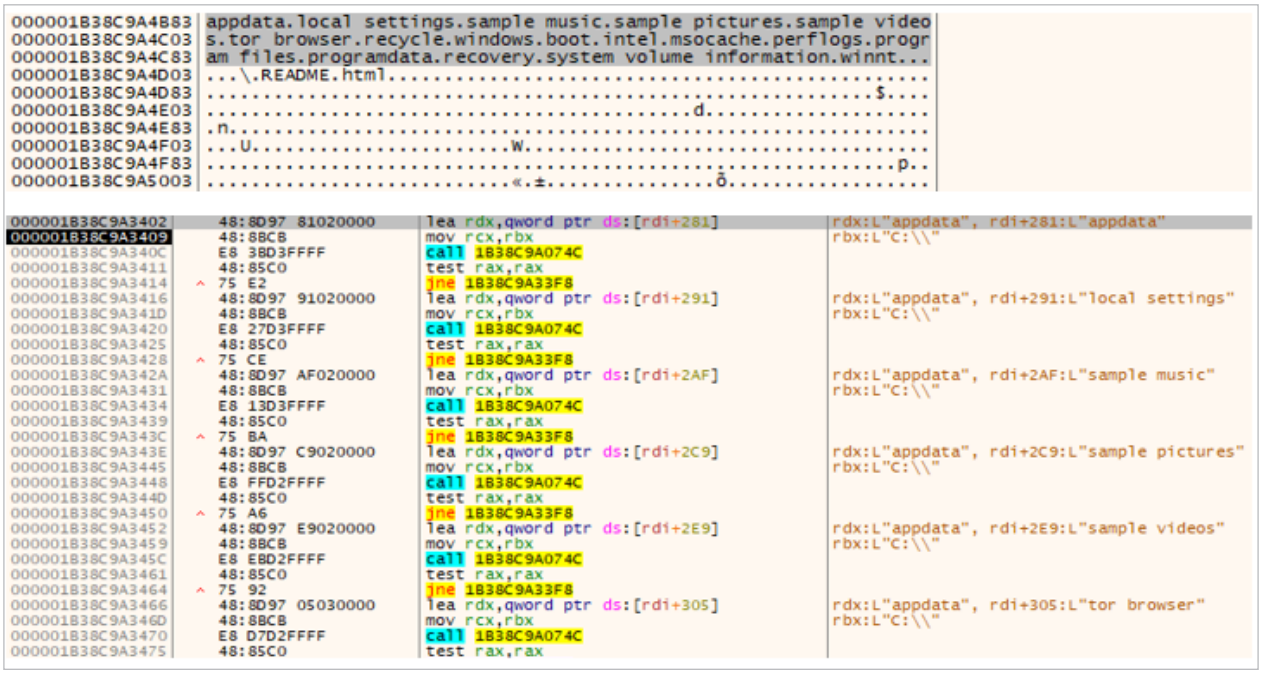
3.16 뮤텍스 생성



[그림 21] 뮤텍스 생성

NtCreateMutant Native API를 호출하여 암호화 시 동기화 문제를 해결하기 위해 뮤텍스를 생성한다.
다중 암호화 방식을 사용하는 매그니베르(Magniber)는 동기화 문제를 위와 같은 방식으로 해결한다.

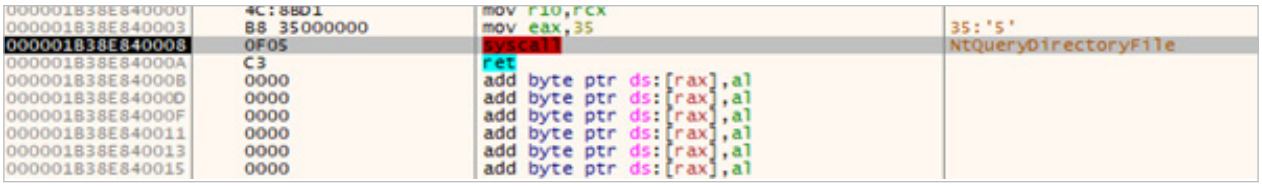
3.17 암호화 대상 폴더



[그림 22] 암호화 대상 폴더

매그니베르(Magniber) 랜섬웨어는 암호화 대상 폴더를 열거하고
해당 폴더가 볼륨 내에 존재하는지를 비교하기 시작한다.
해당 폴더가 볼륨내에 존재한다면 암호화 대상 리스트에 넣어 놓고,
그렇지 않으면 암호화 대상 폴더 리스트에서 제거한다.
해당 방식은 암호화 진행에 있어 신속한 암호화를 하기 위함이다.

3.18 폴더 및 파일 열거



[그림 23] 암호화 대상 폴더 및 파일 열거

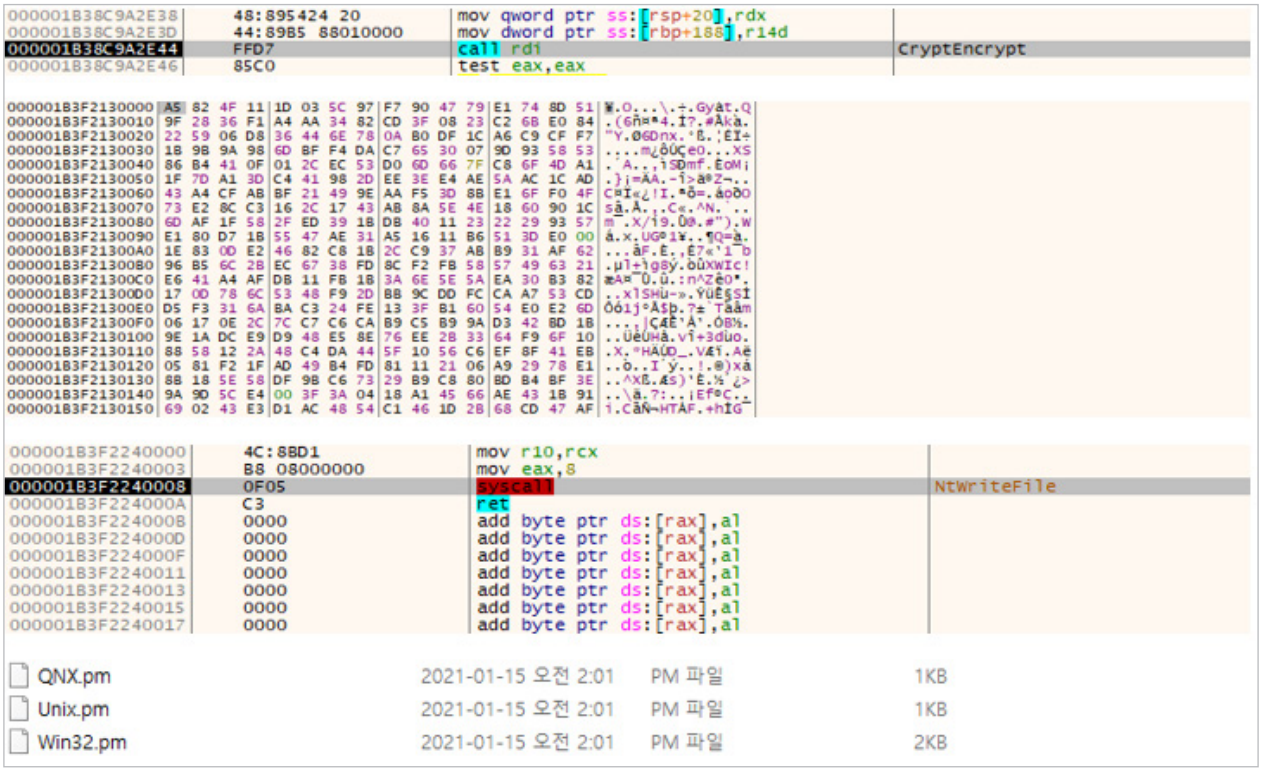
NtQueryDirectoryFile Native API를 호출하여 암호화 대상이 되는 폴더 및 파일을 열거한다.
해당 API는 특정 폴더 및 파일을 조회하고 열거하는 기능을 하는 Native API이다.

3.19 암호화 수행 과정 (파일 읽기)

[그림 24] 암호화 대상 파일 읽기

암호화 대상 파일을 읽어 메모리에 적재하는데, NtCreateFile Native API를 통해 파일의 핸들을 획득하고,
NtQueryInformationFile Native API를 호출하여 파일의 정보를 획득한다.
파일의 크기 및 종류 등 파일의 정보를 획득한 후
NtReadFile Native API를 호출하여 파일을 읽어 메모리에 적재한다.

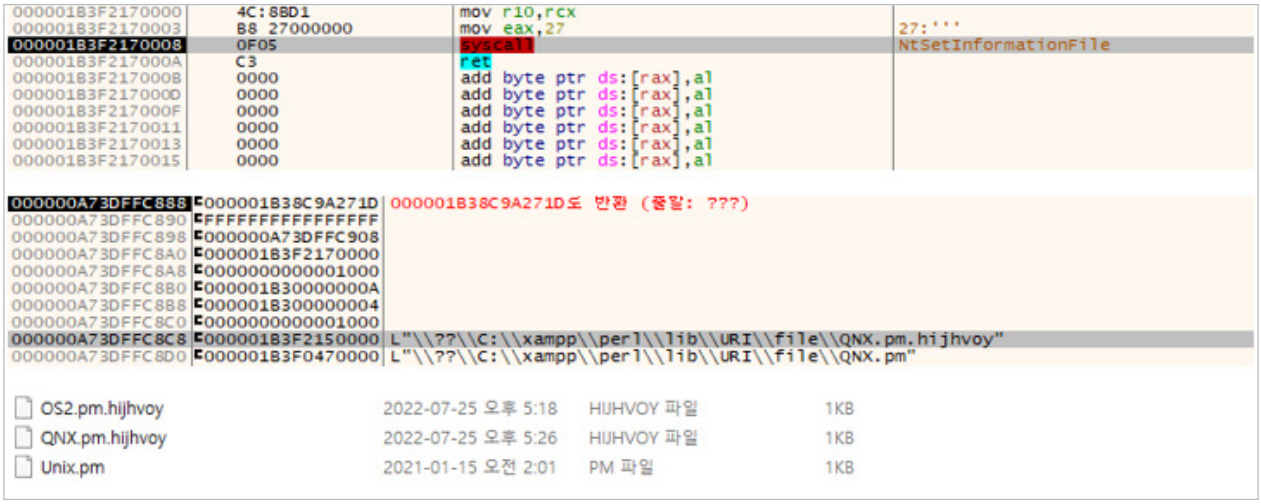
3.20 암호화 수행 과정 (파일 암호화 및 덮어쓰기)



[그림 25] 암호화 대상 파일 암호화 및 덮어쓰기

이전에 메모리 내 적재한 파일의 데이터는 CryptEncrypt API를 호출하여 암호화를 수행한다.
암호화가 수행된 데이터는 NtWriteFile Native API를 이용하여 기존 파일에 덮어쓴다.
해당 과정을 통해 파일에 복구가 불가능한 암호화가 수행된다.

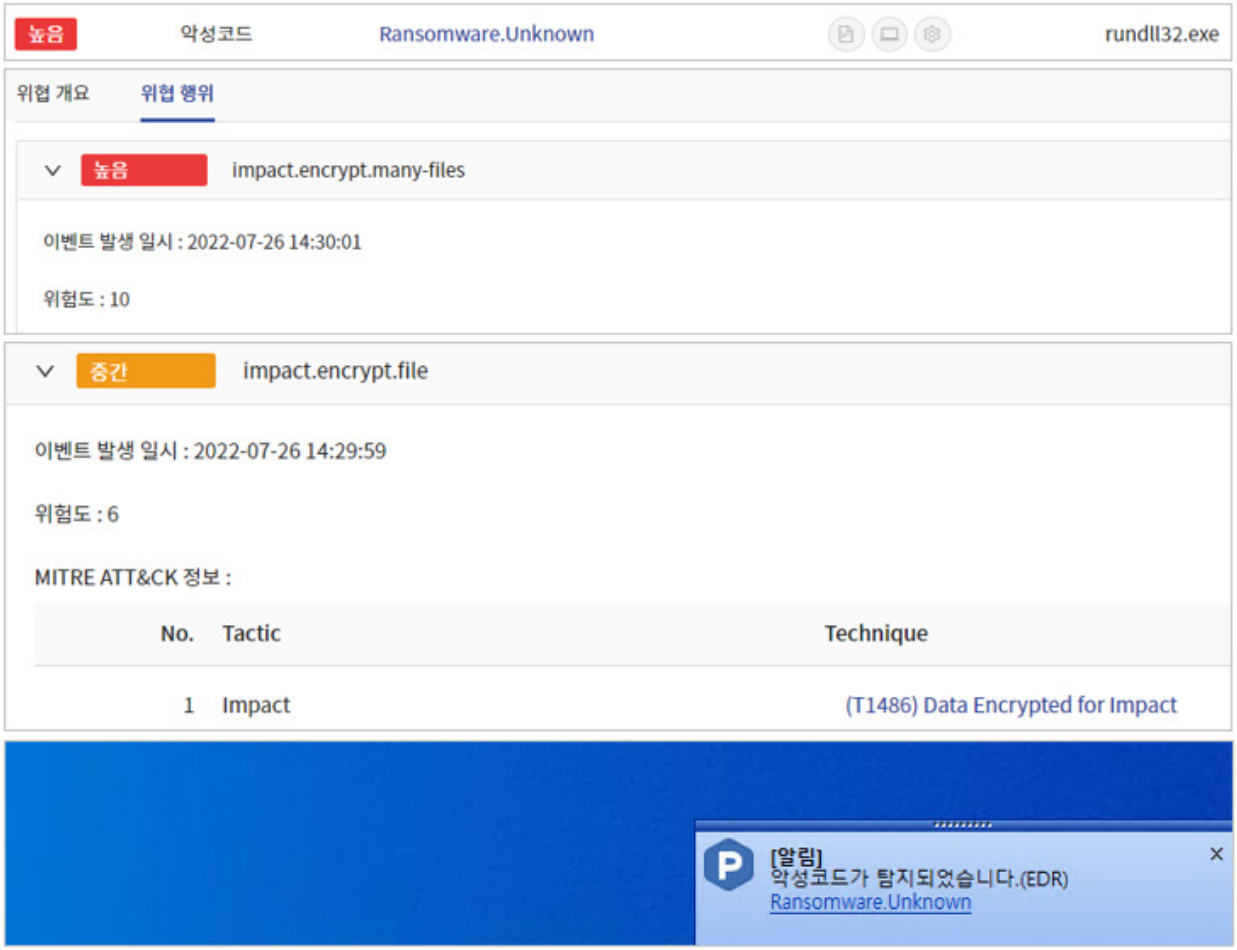
3.21 암호화 수행 과정 (파일 이름 변경)



[그림 26] 암호화 대상 파일 이름 변경

암호화가 완료된 파일에 대해 NtSetInformationFile Native API를 사용하여 파일의 이름을 변경한다. 본 샘플에서는 암호 확장자가 .hijhvoy이며, 이는 암호화 시 마다 변경된다.

4. Privacy-i EDR 탐지 정보 (Ransomware.Unknown)



[그림 37] Privacy-i EDR 랜섬웨어 탐지 정보 (파일 암호화)

Privacy-i EDR의 매그니베르(Magniber) 랜섬웨어는
실시간으로 프로세스의 행위를 모니터링 하는 행위 기반 탐지엔진에서 탐지 됐다.
Rundll32.exe 프로세스에서 암호화 수행 시 정보 획득과 동시에 파일 격리 및 차단이 이루어진다.
랜섬웨어에 대해서는 Ransomware.Unknown으로 탐지하였다.

5. 대응

1. Privacy-i EDR과 같은 **EDR 솔루션의 '행위기반 탐지엔진'**으로 차단
   : 일반 Anti-Virus 솔루션에서도 대부분 차단 가능하나 최신 업데이트 필요
2. 악성코드 주요 감염경로인 **P2P, 음란, 도박 등 불법 웹사이트 연결 사전차단**
3. 메일 내용과 보내는이 계정에 연관성이 없거나, 문법적으로 어색하고,
   신뢰할 수 없는 링크 또는 첨부파일 클릭을 유도하는 메일은 실행 금지
4. **OS 및 소프트웨어 보안 업데이트** 최신형상으로 유지

궁금하신 점이나 문의사항은 malware@somansa.com 으로 문의주십시오