

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 두절**되며 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

## 2. 정보

## 3. 분석

- 3.1 위장 및 탐지 우회
- 3.2 윈도우 프로세스를 통한 실행 (제어판, Rundll32)2.3 공유 폴더 연결 해제
- 3.3 코드 난독화
- 3.4 난독화 해제를 위한 메모리 할당
- 3.5 난독화 해제 후 코드 적재
- 3.6 샌드박스 기반 탐지 우회
- 3.7 개체 디렉토리 할당
- 3.8 DosDevice 열기 시도
- 3.9 파일 시스템 정보 획득
- 3.10 볼륨 정보 획득
- 3.11.1 랜섬노트 제작 (내부 Script 실행)
- 3.11.2 랜섬노트 제작 (내부 Script 기재)
- 3.12 키 파일 생성 (키 시드)
- 3.13 키 파일 생성 (암호화 키 생성)
- 3.14 암호화 키 импорт
- 3.15 암호화 키 파일 암호화
- 3.16 뮤텍스 생성
- 3.17 암호화 대상 폴더
- 3.18 폴더 및 파일 열거
- 3.19 암호화 수행 과정 (파일 읽기)
- 3.20 암호화 수행 과정 (파일 암호화 및 덮어쓰기)
- 3.21 암호화 수행 과정 (파일 이름 변경)

## 4. Privacy-i EDR 탐지 정보

## 5. 대응

## 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



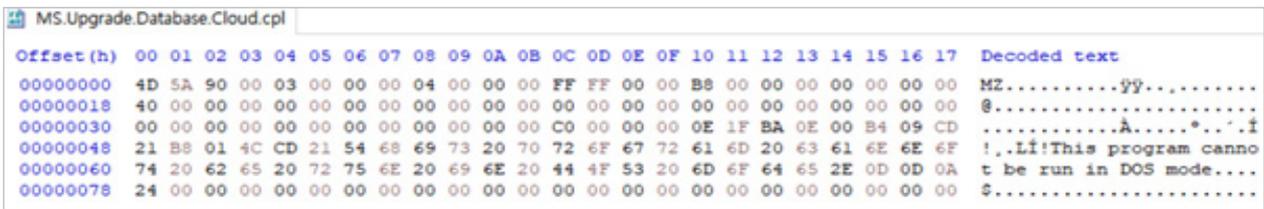
[그림 2] 매그니베르(Magniber) 랜섬웨어 공격 흐름

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.

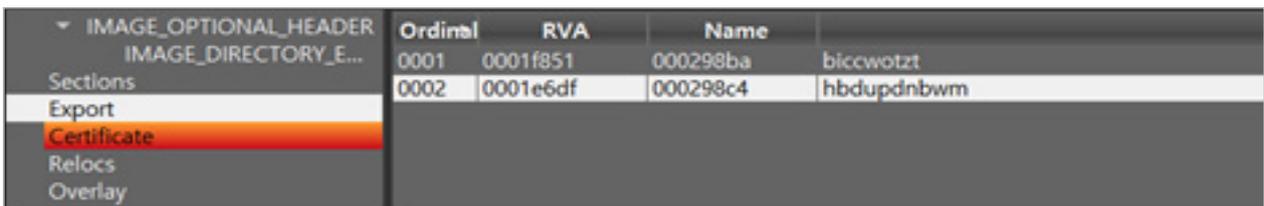
### 3. 분석

#### 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)

00007FFBC58DCB60	4C: 88DC	mov r11, rsp	CreateProcessW
00007FFBC58DCB63	48: 83EC 58	sub rsp, 58	rax: L"C:\\Windows\\System32"
00007FFBC58DCB67	48: 8B8424 A8000000	mov rax, qword ptr ss: [rsp+A8]	[rsp+A0]: L"C:\\Windows\\system32\\rundll32.exe"
00007FFBC58DCB6F	49: 8943 F0	mov qword ptr ds: [r11-10], rax	rax: L"C:\\Windows\\System32"
00007FFBC58DCB73	48: 8B8424 A0000000	mov rax, qword ptr ss: [rsp+A0]	rax: L"C:\\Windows\\System32"
00007FFBC58DCB78	49: 8943 E8	mov qword ptr ds: [r11-18], rax	
00007FFBC58DCB7F	48: 8B8424 98000000	mov rax, qword ptr ss: [rsp+98]	
00007FFBC58DCB87	49: 8943 E0	mov qword ptr ds: [r11-20], rax	
00007FFBC58DCB8B	48: 8B8424 90000000	mov rax, qword ptr ss: [rsp+90]	
00007FFBC58DCB93	49: 8943 D8	mov qword ptr ds: [r11-28], rax	rax: L"C:\\Windows\\System32"
00007FFBC58DCB97	8B8424 88000000	mov eax, dword ptr ss: [rsp+88]	
00007FFBC58DCB9E	894424 28	mov dword ptr ss: [rsp+28], eax	
00007FFBC58DCBA2	8B8424 80000000	mov eax, dword ptr ss: [rsp+80]	
00007FFBC58DCBA9	894424 20	mov dword ptr ss: [rsp+20], eax	
00007FFBC58DCBAD	48: FF15 844F0600	call qword ptr ds: [844F0600]	

00000207830CE2C0	22 00 43 00	3A 00 5C 00	57 00 69 00	6E 00 64 00	" . C . : . \ . W . i . n . d .
00000207830CE2D0	6F 00 77 00	73 00 5C 00	73 00 79 00	73 00 74 00	o . w . s . \ . s . y . s . t .
00000207830CE2E0	65 00 60 00	33 00 32 00	5C 00 72 00	75 00 6E 00	e . m . 3 . 2 . \ . r . u . n .
00000207830CE2F0	64 00 6C 00	6C 00 33 00	32 00 2E 00	65 00 78 00	d . l . l . 3 . 2 . . . . e . x .
00000207830CE300	65 00 22 00	20 00 53 00	68 00 65 00	6C 00 6C 00	e . " . . . . s . h . e . l . l .
00000207830CE310	33 00 32 00	2E 00 64 00	6C 00 6C 00	2C 00 43 00	3 . 2 . . . . d . l . l . , . C .
00000207830CE320	6F 00 6E 00	74 00 72 00	6F 00 6C 00	5F 00 52 00	o . n . t . r . o . l . _ . R .
00000207830CE330	75 00 6E 00	44 00 4C 00	4C 00 20 00	22 00 43 00	u . n . d . l . l . _ . m . c .
00000207830CE340	3A 00 5C 00	55 00 73 00	65 00 72 00	73 00 5C 00	: . \ . U . S . E . R . S . \ .
00000207830CE350	4A 00 65 00	6F 00 6E 00	67 00 5F 00	56 00 4D 00	J . e . o . n . g . _ . V . M .
00000207830CE360	5C 00 44 00	6F 00 77 00	6E 00 6C 00	6F 00 61 00	. . D . o . w . n . l . o . a .
00000207830CE370	64 00 73 00	5C 00 4D 00	53 00 2E 00	55 00 70 00	d . s . \ . M . S . . . . U . p .
00000207830CE380	67 00 72 00	61 00 64 00	65 00 2E 00	44 00 61 00	g . r . a . d . e . . . . D . a .
00000207830CE390	74 00 61 00	62 00 61 00	73 00 65 00	2E 00 43 00	t . a . b . a . s . e . . . . C .
00000207830CE3A0	6C 00 6F 00	75 00 64 00	2E 00 63 00	70 00 6C 00	l . o . u . d . . . . c . p . l .

[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

00007FFB881A68E5	FFCA	dec edx	EntryPoint
00007FFB881A68F0	75 05	jne ms.upgrade.database.cloud.7FFB881A68F7	
00007FFB881A68F2	E8 DC78FEFF	call ms.upgrade.database.cloud.7FFB8818E4D3	
00007FFB881A68F7	B8 01000000	mov eax,1	
00007FFB881A68FC	C3	ret	
00007FFB881A68FD	0000	add byte ptr ds:[rax],al	rax:EntryPoint
00007FFB881A68FF	0000	add byte ptr ds:[rax],al	rax:EntryPoint
00007FFB881A6C01	0000	add byte ptr ds:[rax],al	rax:EntryPoint
00007FFB881A6C03	0000	add byte ptr ds:[rax],al	rax:EntryPoint
00007FFB881A6C05	0000	add byte ptr ds:[rax],al	rax:EntryPoint
00007FFB881A6C07	0000	add byte ptr ds:[rax],al	rax:EntryPoint
00007FFB881A6C09	0000	add byte ptr ds:[rax],al	rax:EntryPoint
00007FFB881A6C0B	0000	add byte ptr ds:[rax],al	rax:EntryPoint
00007FFB881A6C0D	0000	add byte ptr ds:[rax],al	rax:EntryPoint
00007FFB881A6C0F	0000	add byte ptr ds:[rax],al	rax:EntryPoint
00007FFB8818E578	49:81F6 D0FA0000	xor r14,FAD0	
00007FFB8818E57F	E9 2E020000	jmp ms.upgrade.database.cloud.7FFB8818E7B2	
00007FFB8818E584	E2 65	loop ms.upgrade.database.cloud.7FFB8818E5EB	
00007FFB8818E586	43:170 9F	jo ms.upgrade.database.cloud.7FFB8818E528	
00007FFB8818E589	313F	xor dword ptr ds:[rdi],edi	
00007FFB8818E58B	DF41 B2	fld st(0),word ptr ds:[rcx-4E]	
00007FFB8818E58E	5C	pop rsp	
00007FFB8818E590	93	xchg ebx,eax	
00007FFB8818E591	9F	lahf	
00007FFB8818E593	E4 33	in al,33	
00007FFB8818E596	42:14 3F	adc al,3F	
00007FFB8818E599	DA25 53E95701	fisub st(0),dword ptr ds:[7FFB8818E599]	
00007FFB8818E59C	0000	add byte ptr ds:[rax],al	rax:EntryPoint
00007FFB8818E59E	67	mov eax,edx	
00007FFB8818E59F	16	mov eax,dword ptr ds:[rax-628C6F8C]	
00007FFB8818E5A0	8B80 7490739D	xchg esp,eax	
00007FFB8818E5A6	94	pop rax	rax:EntryPoint
00007FFB8818E5A7	58	pop rax	
00007FFB8818E5A8	5A	pop rdx	
00007FFB8818E5A9	8580 C5884F6F	test dword ptr ds:[rax+6F4FB8C5],eax	

[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

00007FFB881A3653	0F05	mov eax,1	NtAllocateVirtualMemory
00007FFB881A3655	EB BC	jmp ms.upgrade.database.cloud.7FFB881A3613	
00007FFB881A3657	CA 3AF1	ret far F13A	
00007FFB881A365A	E5 93	in eax,93	
00007FFB881A365C	25 5BC368A9	and eax,A968C358	
00007FFB881A3661	2095 CADE7836	and byte ptr ss:[rbp+3678DECA],d1	
00007FFB881A3667	A9 5F1CFA80	test eax,80FA1C5F	
00007FFB881A366C	4043:8D 4618A54C	mov r13d,4CA51846	
00007FFB881A3673	87E4	xchg esp,esp	
00007FFB881A3675	EB DC	jmp ms.upgrade.database.cloud.7FFB881A3653	
00007FFB881A3677	B5 BC	mov ch,82	
000000A73DFFDA68	00 00 9A 8C B3 01 00 00	00 00 00 00 00 00 00 00	.....*
000000A73DFFDA78	00 00 18 B8 FB 7F 00 00	01 00 00 00 00 00 00 00	...i.ü.....
000000A73DFFDA88	EE 68 1A B8 FB 7F 00 00	84 03 FE 7F 00 00 00 00	ik.ü.....p.....
000000A73DFFDA98	01 00 00 00 00 00 00 00	85 03 FE 7F 00 00 00 00	.....p.....
000000A73DFFDAA8	C8 DD FF 3D A7 00 00 00	F7 68 1A B8 FB 7F 00 00	EYy=s...+k.ü...
000000A73DFFDAB8	3D 78 6E C6 FB 7F 00 00	00 00 00 00 00 00 00 00	={n{ü.....
000000A73DFFDAC8	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000000A73DFFDAD8	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000000A73DFFDAE8	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000000A73DFFDAF8	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
000000A73DFFDB08	00 00 00 00 00 00 00 00	00 06 00 00 00 00 00 00	.....
000000A73DFFDB18	F0 4D 7D 8C B3 01 00 00	70 72 7D 8C B3 01 00 00	0M}.*...pr}.*
000000A73DFFDB28	23 35 71 C6 FB 7F 00 00	D0 71 7D 8C B3 01 00 00	#5q{ü...Dq}.*...
000001B38C9A0000	0000	add byte ptr ds:[rax],al	
000001B38C9A0002	0000	add byte ptr ds:[rax],al	
000001B38C9A0004	0000	add byte ptr ds:[rax],al	
000001B38C9A0006	0000	add byte ptr ds:[rax],al	
000001B38C9A0008	0000	add byte ptr ds:[rax],al	
000001B38C9A000A	0000	add byte ptr ds:[rax],al	
000001B38C9A000C	0000	add byte ptr ds:[rax],al	
000001B38C9A000E	0000	add byte ptr ds:[rax],al	
000001B38C9A0010	0000	add byte ptr ds:[rax],al	
000001B38C9A0012	0000	add byte ptr ds:[rax],al	
000001B38C9A0014	0000	add byte ptr ds:[rax],al	

[그림 8] 코드 난독화 해제를 위한 메모리 할당

Magniber 랜섬웨어

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

000001B38C9C0000	0000	add byte ptr ds:[rax],al	
000001B38C9C0002	0000	add byte ptr ds:[rax],al	
000001B38C9C0004	0000	add byte ptr ds:[rax],al	
000001B38C9C0006	0000	add byte ptr ds:[rax],al	
000001B38C9C0008	0000	add byte ptr ds:[rax],al	
000001B38C9C000A	0000	add byte ptr ds:[rax],al	
000001B38C9C000C	0000	add byte ptr ds:[rax],al	
000001B38C9C000E	0000	add byte ptr ds:[rax],al	
000001B38C9C0010	0000	add byte ptr ds:[rax],al	
000001B38C9C0012	0000	add byte ptr ds:[rax],al	
000001B38C9C0000	4C:8BD1	mov r10,rcx	
000001B38C9C0003	88 31000000	mov eax,31	31: '1'
000001B38C9C0008	0F05	syscall	NtQueryPerformanceCounter
000001B38C9C000A	C3	ret	
000001B38C9C000B	0000	add byte ptr ds:[rax],al	
000001B38C9C000D	0000	add byte ptr ds:[rax],al	
000001B38C9C000F	0000	add byte ptr ds:[rax],al	
000001B38C9C0011	0000	add byte ptr ds:[rax],al	
000001B38C9C0013	0000	add byte ptr ds:[rax],al	
000001B38C9C0015	0000	add byte ptr ds:[rax],al	
000001B38C9C0017	0000	add byte ptr ds:[rax],al	
000001B38C9C0019	0000	add byte ptr ds:[rax],al	

[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

000001B38C9D0000	4C:8BD1	mov r10,rcx	
000001B38C9D0003	88 34000000	mov eax,34	34: '4'
000001B38C9D0008	0F05	syscall	NtDelayExecution
000001B38C9D000A	C3	ret	
000001B38C9D000B	0000	add byte ptr ds:[rax],al	
000001B38C9D000D	0000	add byte ptr ds:[rax],al	
000001B38C9D000F	0000	add byte ptr ds:[rax],al	
000001B38C9D0011	0000	add byte ptr ds:[rax],al	
000001B38C9D0013	0000	add byte ptr ds:[rax],al	
000001B38C9D0015	0000	add byte ptr ds:[rax],al	

[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	
<b>000001B38C9E0008</b>	<b>0F05</b>	<b>syscall</b>	<b>NtQueryInformationProcess</b>
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'
<b>000001B38E5E0008</b>	<b>0F05</b>	<b>syscall</b>	<b>NtOpenFile</b>
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

000001B38E5F0000	4C:8BD1	mov r10,rcx	
000001B38E5F0003	B8 49000000	mov eax,49	49:'I'
<b>000001B38E5F0008</b>	<b>0F05</b>	<b>syscall</b>	<b>NtQueryVolumeInformationFile</b>
000001B38E5F000A	C3	ret	
000001B38E5F000B	0000	add byte ptr ds:[rax],al	
000001B38E5F000D	0000	add byte ptr ds:[rax],al	
000001B38E5F000F	0000	add byte ptr ds:[rax],al	
000001B38E5F0011	0000	add byte ptr ds:[rax],al	
000001B38E5F0013	0000	add byte ptr ds:[rax],al	
000001B38E5F0015	0000	add byte ptr ds:[rax],al	

1: rcx 000000000000001DC
2: rdx 000000A73DFFD478
3: r8 000000A73DFFD470
4: r9 0000000000000008
5: [rsp+28] 0000000000000004

[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

000001B38E600000	4C:8BD1	mov r10,rcx	
000001B38E600003	B8 49000000	mov eax,49	49:'I'
<b>000001B38E600008</b>	<b>0F05</b>	<b>syscall</b>	<b>NtQueryVolumeInformationFile</b>
000001B38E60000A	C3	ret	
000001B38E60000B	0000	add byte ptr ds:[rax],al	
000001B38E60000D	0000	add byte ptr ds:[rax],al	
000001B38E60000F	0000	add byte ptr ds:[rax],al	
000001B38E600011	0000	add byte ptr ds:[rax],al	

1: rcx 000000000000001DC
2: rdx 000000A73DFFD478
3: r8 000000A73DFFD498
4: r9 0000000000000020
5: [rsp+28] 0000000000000001

[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)

000001B38C9A0242	8D0492	lea eax,qword ptr ds:[rdx+rdx*4]	
000001B38C9A0245	03C0	add eax,eax	
000001B38C9A0247	44:2BC0	sub r8d,eax	
000001B38C9A024A	43:0FB74C47 45	movzx ecx,word ptr ds:[r15+r8*2+45]	r15+r8*2+45:"[36]+fnxjthp[36]+fnxjthp[3
000001B38C9A0250	44:8BC2	mov r8d,ecx	
000001B38C9A0253	66:890F	mov word ptr ds:[rdi],cx	
000001B38C9A0256	48:8D7F 02	lea rdi,qword ptr ds:[rdi+2]	
000001B38C9A025A	85D2	test edx,edx	
000001B38C9A025C	^ 75 D9	jne 1B38C9A0237	
000001B38C9A025E	48:8D57 FE	lea rdx,qword ptr ds:[rdi-2]	rdi-2:L"29"
000001B38C9A0262	48:8D4424 78	lea rax,qword ptr ss:[rsp+78]	
000001B38C9A0267	6644:892F	mov word ptr ds:[rdi],r13w	

```

000001B38C9A58BD %&..0B..[...8..*A..<html><body><script>var fnxjthp = new Array(
000001B38C9A58FD 117,46,60,111,52,78,89,84,55,54,72,119,82,73,106,121,86,115,112,
000001B38C9A593D 104,108,107,79,122,49,116,97,39,32,105,66,83,68,102,62,51,61,70,
000001B38C9A597D 67,85,118,53,69,58,50,99,120,47,110,77,76,114,103,98,101,33,65,1
000001B38C9A598D 00,80,109);for(var ncesiguu=0;ncesiguu<60;ncesiguu++) fnxjthp[nc
000001B38C9A59FD esiguu] = String.fromCharCode(fnxjthp[ncesiguu]);hqzvfuaqlje = ".
000001B38C9A5A3D ";document[fnxjthp[11]+fnxjthp[51]+fnxjthp[29]+fnxjthp[25]+fnxjt
000001B38C9A5A7D hp[54]](fnxjthp[56]+fnxjthp[50]+fnxjthp[50]+fnxjthp[28]+fnxjthp[
000001B38C9A5ABD 6]+fnxjthp[22]+fnxjthp[39]+fnxjthp[12]+fnxjthp[28]+fnxjthp[32]+f
000001B38C9A5AFD nxjthp[22]+fnxjthp[38]+fnxjthp[39]+fnxjthp[49]+fnxjthp[42]+fnxjt
    
```

[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)

```

00000000 3C 68 74 6D 6C 3E 3C 62 6F 64 79 3E 3C 73 63 72 <html><body><scr
00000010 69 70 74 3E 76 61 72 20 66 6E 78 6A 74 68 70 20 ipt>var fnxjthp
00000020 3D 20 6E 65 77 20 41 72 72 61 79 28 31 31 37 2C = new Array(117,
00000030 34 36 2C 36 30 2C 31 31 31 2C 35 32 2C 37 38 2C 46,60,111,52,78,
00000040 38 39 2C 38 34 2C 35 35 2C 35 2C 37 32 2C 31 89,84,55,54,72,1
00000050 31 39 2C 38 32 2C 37 33 2C 31 30 36 2C 31 32 31 19,82,73,106,121
00000060 2C 38 36 2C 31 31 35 2C 31 31 32 2C 31 30 34 2C ,86,115,112,104,
00000070 31 30 38 2C 31 30 37 2C 37 39 2C 31 32 32 2C 34 108,107,79,122,4
00000080 39 2C 31 31 36 2C 39 37 2C 33 39 2C 33 32 2C 31 9,116,97,39,32,1
00000090 30 35 2C 36 36 2C 38 33 2C 36 38 2C 31 30 32 2C 05,66,83,68,102,
    
```

ALL YOUR DOCUMENTS PHOTOS DATABASES AND OTHER IMPORTANT FILES HAVE BEEN ENCRYPTED!

=====

Your files are NOT damaged! Your files are modified only. This modification is reversible.

The only 1 way to decrypt your files is to receive the private key and decryption program.

Any attempts to restore your files with the third party software will be fatal for your files!

=====

To receive the private key and decryption program follow the instructions below:

1. Download 'Tor Browser' from <https://www.torproject.org/> and install it.
2. In the 'Tor Browser' open your personal page here:

[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)

000001B38C9A1335	E8 EEF4FFFF	call 1B38C9A0828	rsi:L"C:\\Users\\Public\\llytnsmitrz.c"
000001B38C9A133A	48:8BD6	mov rdx,rsi	
000001B38C9A133D	49:8BCB	mov rcx,r11	
000001B38C9A1340	E8 1BF5FFFF	call 1B38C9A0860	
000001B38C9A1345	41:0FB7C6	movzx eax,r14w	
000001B38E720000	4C:8BD1	mov r10,rcx	ss:'U' NtCreateFile
000001B38E720003	B8 55000000	mov eax,55	
000001B38E720008	0F05	syscall	
000001B38E72000A	C3	ret	
000001B38E72000B	0000	add byte ptr ds:[rax],al	
000001B38E72000D	0000	add byte ptr ds:[rax],al	
000001B38E72000F	0000	add byte ptr ds:[rax],al	
000001B38E720011	0000	add byte ptr ds:[rax],al	
000001B38E720013	0000	add byte ptr ds:[rax],al	
000001B38E720015	0000	add byte ptr ds:[rax],al	

C:\llytnsmitrz.c 2022-07-21 오후 1:27 C Source 1KB

llytnsmitrz.c - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
#@~^ZgMAAA=]Y~!&Nv,Sv2+VMF~',MnO}4L^YvJhbUhosYk)'kh2DdW CYbWUJW+sxb:2+MdK
+mDPC~sMWhPqkUfy{?4CNKhZK2zj*#@&sGD~3mm4PkChF:q0lG%h~bx~o0*Fts+:230@#@&kChF
^.8R!YvJbx2+{hDGm/djDIDD;wr#@#@&jnY,:T;Twl4(PxPNT&7 1OR6Rj2mhUq dDIU1+|@#@
```

[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)

000001B38E880000	4C:8BD1	mov r10,rcx	
000001B38E880003	B8 06000000	mov eax,6	
000001B38E880008	0F05	syscall	NtReadFile
000001B38E88000A	C3	ret	
000001B38E88000B	0000	add byte ptr ds:[rax],al	
000001B38E88000D	0000	add byte ptr ds:[rax],al	
000001B38E88000F	0000	add byte ptr ds:[rax],al	
000001B38E880011	0000	add byte ptr ds:[rax],al	

000001B38E7B0000	23 40 7E 5E 5A 67 4D 41 41 41 3D 3D 6A 7F 59 7E	mov rax,rcx	#@~^ZgMAAA==j.Y~
000001B38E7B0010	21 26 4E 76 2C 53 76 32 2B 56 4D 46 7E 27 2C 4D	lea rax,qword ptr ss:[rbp-38]	!&Nv,Sv2+VMF~',M
000001B38E7B0020	6E 4F 7D 34 4C 7F 5E 59 76 4A 68 62 55 68 6F 73	mov dword ptr ss:[rsp+30],100000	nO}4L.^YvJhbuhos
000001B38E7B0030	59 68 29 60 68 68 32 7F 44 64 57 09 43 59 62 57	mov qword ptr ss:[rsp+28],rax	Yk} `kh2.DdW.CYBW
000001B38E7B0040	55 4A 7F 5C 28 73 78 62 3A 32 2B 4D 64 4B 78 43	xor r9d,r9d	UJ.\+sxb:2+MdkxC
000001B38E7B0050	44 2B 29 22 2D 27 52 27 2E 57 4B 59 77 6D 62 3A	mov r8d,r14d	D+) "'-R'.wKYwmb:
000001B38E7B0060	2D 28 72 23 40 23 40 26 6A 2B 44 50 6F 30 2A 7B	mov qword ptr ss:[rsp+20],rdx	-+r#@#&j+DPOO?{
000001B38E7B0070	74 5E 76 73 26 56 30 7E 78 2C 45 66 4C 2B 31 68	mov dword ptr ss:[rbp+188],r14d	t^Vs&V0~x,EfL+1h
000001B38E7B0080	2B 26 28 73 4D 46 52 33 61 7F 6D 7D 45 7F 2E 48	call rdi	+&SMFR3a.m}E..H
000001B38E7B0090	60 45 55 28 5E 28 6D 44 50 43 7E 73 4D 57 68 50	test eax,eax	EU+^+mDPC~SMWhP
000001B38E7B00A0	71 68 55 66 79 7B 3F 34 43 4E 4B 68 5A 4B 32 7A	je 1B38C9A2EE7	qkUfy{?4CNkH2K2z
000001B38E7B00B0	4A 2A 40 23 40 26 73 47 44 7E 33 6D 6D 34 50 68		J*#@&sGD~3mm4Pk
000001B38E7B00C0	43 68 46 3A 71 30 6C 47 25 68 7E 62 78 7E 6F 30		ChF:q0lG%h~bx~o0

000001B38C9A2E1D	44:0F42F0	cmovb r14d,eax	
000001B38C9A2E21	48:8D45 C8	lea rax,qword ptr ss:[rbp-38]	
000001B38C9A2E25	C74424 30 00001000	mov dword ptr ss:[rsp+30],100000	
000001B38C9A2E2D	48:894424 28	mov qword ptr ss:[rsp+28],rax	
000001B38C9A2E32	45:33C9	xor r9d,r9d	
000001B38C9A2E35	45:8BC6	mov r8d,r14d	
000001B38C9A2E38	48:895424 20	mov qword ptr ss:[rsp+20],rdx	
000001B38C9A2E3D	44:89B5 88010000	mov dword ptr ss:[rbp+188],r14d	
000001B38C9A2E44	FFD7	call rdi	CryptEncrypt
000001B38C9A2E46	85C0	test eax,eax	
000001B38C9A2E48	0F84 99000000	je 1B38C9A2EE7	

000001B38E7B0000	C2 4B 11 FF 7B BD 0F 88 14 8F B1 15 92 84 54 20	mov r10,rcx	Ak.Y{%. . . . .T
000001B38E7B0010	33 5D 98 DA 8C C6 5E E1 3E 81 5D BF 0F 22 74 B6	mov eax,8	3}.U.#Aa>.]&. "t
000001B38E7B0020	DF D3 67 08 38 3A 4A C9 F6 1D 0E D6 95 36 D1 B7	syscall	B0g.8:JÉö..0.6N-
000001B38E7B0030	02 7F D8 C0 C3 89 7B 90 E3 9D 89 DF 30 AD 52 8C	ret	..0AA' { .ä..B0.R.
000001B38E7B0040	4E BE 37 55 61 C6 66 D1 70 78 4C 6F B4 A4 7C A7	add byte ptr ds:[rax],al	N47UaefNpxLo R §
000001B38E7B0050	97 F2 75 CD 69 70 20 73 40 78 1E 36 46 C4 62 F1	add byte ptr ds:[rax],al	.duiip sex.6FAbh
000001B38E7B0060	A6 AB 25 41 25 89 AD D6 80 69 2D 9B 16 F8 2F 50	add byte ptr ds:[rax],al	; <%A%. .0. i- . .0/P
000001B38E7B0070	8D 9D D3 30 A6 60 05 08 96 10 58 A3 E5 DC AF B1	add byte ptr ds:[rax],al	..00' . . . . XfAU±
000001B38E7B0080	5E CE 13 0D 9C 95 AC AC 21 56 07 CA 85 9D AB EE	add byte ptr ds:[rax],al	^i. . . . IV.É. «i
000001B38E7B0090	D7 A0 21 89 B4 37 5F 73 E2 F6 E2 C2 2F 3D 8D 7A	add byte ptr ds:[rax],al	x !. 7_säöÄ/= .z
000001B38E7B00A0	BD 7D B2 13 13 7B CA 82 54 4D BE 4D 4B 68 D2 CB	add byte ptr ds:[rax],al	%} = . . {É. TM%MKKÖE
000001B38E7B00B0	FB C6 C6 AA 13 63 28 1E EF 6E BD 6B E0 2F 1C BD	add byte ptr ds:[rax],al	ÜÄÄ^ . c(. 1n%ka/. %
000001B38E7B00C0	B2 60 8C 8B 2E 4A 65 9E 84 AB 88 44 BE 95 52 23	add byte ptr ds:[rax],al	* %>. Je. . «. D%. R#

000001B38E8C0000	4C:8BD1	mov r10,rcx	
000001B38E8C0003	B8 08000000	mov eax,8	
000001B38E8C0008	0F05	syscall	NtWriteFile
000001B38E8C000A	C3	ret	
000001B38E8C000B	0000	add byte ptr ds:[rax],al	
000001B38E8C000D	0000	add byte ptr ds:[rax],al	
000001B38E8C000F	0000	add byte ptr ds:[rax],al	
000001B38E8C0011	0000	add byte ptr ds:[rax],al	
000001B38E8C0013	0000	add byte ptr ds:[rax],al	
000001B38E8C0015	0000	add byte ptr ds:[rax],al	

[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 импорт

000001B38C9A2FC4	48:8855 18	mov rdx,qword ptr ss:[rbp+18]	[rbp+18]:L"\\??\\"
000001B38C9A2FC8	48:884D 10	mov rcx,qword ptr ss:[rbp+10]	
000001B38C9A2FCC	48:8D45 28	lea rax,qword ptr ss:[rbp+28]	
000001B38C9A2FD0	48:894424 28	mov qword ptr ss:[rsp+28],rax	
000001B38C9A2FD5	48:81C2 3F010000	add rdx,13F	
000001B38C9A2FDC	45:33C9	xor r9d,r9d	
000001B38C9A2FDF	41:B8 14010000	mov r8d,114	
000001B38C9A2FE5	44:896C24 20	mov dword ptr ss:[rsp+20],r13d	
000001B38C9A2FEA	FF55 40	call qword ptr ss:[rbp+40]	CryptImportKey
000001B38C9A2FED	85C0	test eax,eax	
000001B38C9A2FEF	0F84 78010000	je 1B38C9A316D	
000001B38C9A2FF5	48:884D 28	mov rcx,qword ptr ss:[rbp+28]	

000001B38C9A4A1	06 02 00 00	00 A4 00 00	52 53 41 31	00 08 00 00	.....P..RSA1...
000001B38C9A4A51	01 00 01 00	71 25 52 3F	4A 95 75 C9	BE 22 1A D5	...qR?].uEM".0
000001B38C9A4A61	16 EF 07 C4	CE 22 3E DB	2C DA 84 72	B3 FB B3 88	.i.Ai">0,u.r'u*
000001B38C9A4A71	35 39 63 EF	F8 5C EE B8	B1 41 1C FE	B3 A7 61 EF	59c10\i±A.b*sai
000001B38C9A4A81	14 02 98 72	A3 A1 C0 E6	F3 58 70 80	5D CA 04 8E	...r.f].Ae0xp.]E..
000001B38C9A4A91	38 DD 0E 08	8D 6C AD 2A	24 AD 0F 2C	13 D1 E3 C6	;Y...l.*\$....Naa
000001B38C9A4AA1	38 B1 D9 20	6E 68 C7 77	E5 08 4A 8D	B5 A8 5E F7	;±U nhCwa.J.u ^±
000001B38C9A4AB1	00 33 F6 9E	BE 63 ED 49	17 D2 94 C3	CE FA 4C 94	.30.Wc'I.O.AiU.L
000001B38C9A4AC1	69 FE 8E EE	95 65 7C F5	B2 C0 A9 5B	D2 AD 16 94	jp.i.e]ô^Ae[O...]
000001B38C9A4AD1	E0 98 C8 68	00 DA 85 CE	0D FB 10 72	AF E4 CB 03	a.Eh.U.I.U.r aE.
000001B38C9A4AE1	9A 30 B3 C4	72 9E 8B B1	92 BD D5 64	83 D8 6D 26	.0*Ar.»±.x0d.0m&
000001B38C9A4AF1	99 00 58 5D	9E 46 02 CE	20 D4 89 72	CD B8 0C 11	..[.].F.i ô.r.i...
000001B38C9A4B01	AE BA 24 38	CB E2 CA 15	B7 BF AB 39	C1 9F 03 0A	e*\$8EaE..¿«9A...
000001B38C9A4B11	29 83 3A 36	6E A6 34 0A	C2 3E FC 32	06 7B 62 F4	).:6n'4.A>u2.{b0
000001B38C9A4B21	1F 46 F6 51	04 DB D0 B8	62 E4 DA FE	B3 AC 08 B6	.F0Q.00»bâÜp*~.¶
000001B38C9A4B31	51 D8 18 96	2A 8C 8A 15	C6 E6 EC 74	99 38 C7 44	Q0..*...Aait.~;CD
000001B38C9A4B41	DF 88 82 E8	68 2B E6 76	33 49 4A E0	BE 05 57 11	E..eh+æv3I]ax.W.
000001B38C9A4B51	56 3A 30 E1	64 00 6F 00	63 00 75 00	6D 00 65 00	V:0ãd.o.c.u.m.e.

[그림 19] 암호화 키 импорт

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

### 3.15 암호화 키 파일 암호화

000001B38E770000	4C:8BD1	mov r10,rcx	
000001B38E770003	B8 33000000	mov eax,33	33:'3'
000001B38E770008	0F05	syscall	NtOpenFile
000001B38E77000A	C3	ret	
000001B38E77000B	0000	add byte ptr ds:[rax],al	
000001B38E77000C	0000	add byte ptr ds:[rax],al	
000001B38E77000D	0000	add byte ptr ds:[rax],al	
000001B38E77000E	0000	add byte ptr ds:[rax],al	
000001B38E77000F	0000	add byte ptr ds:[rax],al	
000001B38E770010	0000	add byte ptr ds:[rax],al	
000001B38E770011	0000	add byte ptr ds:[rax],al	
000001B38E770012	0000	add byte ptr ds:[rax],al	
000001B38E770013	0000	add byte ptr ds:[rax],al	
000001B38E770014	0000	add byte ptr ds:[rax],al	
000001B38E770015	0000	add byte ptr ds:[rax],al	

000001B38E7F0000	4C:8BD1	mov r10,rcx	
000001B38E7F0003	B8 27000000	mov eax,27	27:'''
000001B38E7F0008	0F05	syscall	NtSetInformationFile
000001B38E7F000A	C3	ret	
000001B38E7F000B	0000	add byte ptr ds:[rax],al	
000001B38E7F000C	0000	add byte ptr ds:[rax],al	
000001B38E7F000D	0000	add byte ptr ds:[rax],al	
000001B38E7F000E	0000	add byte ptr ds:[rax],al	
000001B38E7F000F	0000	add byte ptr ds:[rax],al	
000001B38E7F0010	0000	add byte ptr ds:[rax],al	
000001B38E7F0011	0000	add byte ptr ds:[rax],al	
000001B38E7F0012	0000	add byte ptr ds:[rax],al	
000001B38E7F0013	0000	add byte ptr ds:[rax],al	
000001B38E7F0014	0000	add byte ptr ds:[rax],al	
000001B38E7F0015	0000	add byte ptr ds:[rax],al	

공용 다운로드	2019-12-07 오후 6:14	파일 폴더	
공용 문서	2021-12-16 오후 3:22	파일 폴더	
공용 비디오	2019-12-07 오후 6:14	파일 폴더	
공용 사진	2019-12-07 오후 6:14	파일 폴더	
공용 음악	2019-12-07 오후 6:14	파일 폴더	
llytnsmitrz.c.hjhvoy	2022-07-22 오후 3:36	HUHOVOY 파일	2KB

[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

000001B38E810000	4C: 8801	mov r10,rcx	r10:&L "\\BaseNameObjects\\{159yz10etrp1rn0owvan0stn}jgoprk
000001B38E810003	88 83000000	mov eax,83	
000001B38E810008	0F05	syscall	NtCreateMutant
000001B38E81000A	C3	ret	
000001B38E81000B	0000	add byte ptr ds:[rax],al	
000001B38E81000D	0000	add byte ptr ds:[rax],al	
000001B38E81000F	0000	add byte ptr ds:[rax],al	
000001B38E810011	0000	add byte ptr ds:[rax],al	
000001B38E810013	0000	add byte ptr ds:[rax],al	

[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

000001B38C9A4883	appdata.local settings.sample music.sample pictures.sample video		
000001B38C9A4C03	s.tor browser.recycle.windows.boot.intel.msocache.perflogs.progr		
000001B38C9A4C83	am files.programdata.recovery.system volume information.winnt...		
000001B38C9A4D03	...\.README.html.....		
000001B38C9A4D83	.....s.....		
000001B38C9A4E03	.....d.....		
000001B38C9A4E83	.n.....		
000001B38C9A4F03	..U......W.....		
000001B38C9A4F83	.....P.....		
000001B38C9A5003	.....@.±.....ö.....		
000001B38C9A3402	48: 8D97 81020000	lea rdx,qword ptr ds:[rdi+281]	rdx:L"appdata", rdi+281:L"appdata"
000001B38C9A3409	48: 8BC8	mov rcx,rbx	rbx:L"C:\\"
000001B38C9A340C	E8 3BD3FFFF	call 1B38C9A074C	
000001B38C9A3411	48: 85C0	test rax,rax	
000001B38C9A3414	75 E2	jne 1B38C9A33F8	
000001B38C9A3416	48: 8D97 91020000	lea rdx,qword ptr ds:[rdi+291]	rdx:L"appdata", rdi+291:L"local settings"
000001B38C9A341D	48: 8BC8	mov rcx,rbx	rbx:L"C:\\"
000001B38C9A3420	E8 27D3FFFF	call 1B38C9A074C	
000001B38C9A3425	48: 85C0	test rax,rax	
000001B38C9A3428	75 CE	jne 1B38C9A33F8	
000001B38C9A342A	48: 8D97 AF020000	lea rdx,qword ptr ds:[rdi+2AF]	rdx:L"appdata", rdi+2AF:L"sample music"
000001B38C9A3431	48: 8BC8	mov rcx,rbx	rbx:L"C:\\"
000001B38C9A3434	E8 13D3FFFF	call 1B38C9A074C	
000001B38C9A3439	48: 85C0	test rax,rax	
000001B38C9A343C	75 BA	jne 1B38C9A33F8	
000001B38C9A343E	48: 8D97 C9020000	lea rdx,qword ptr ds:[rdi+2C9]	rdx:L"appdata", rdi+2C9:L"sample pictures"
000001B38C9A3445	48: 8BC8	mov rcx,rbx	rbx:L"C:\\"
000001B38C9A3448	E8 FFD2FFFF	call 1B38C9A074C	
000001B38C9A344D	48: 85C0	test rax,rax	
000001B38C9A3450	75 A6	jne 1B38C9A33F8	
000001B38C9A3452	48: 8D97 E9020000	lea rdx,qword ptr ds:[rdi+2E9]	rdx:L"appdata", rdi+2E9:L"sample videos"
000001B38C9A3459	48: 8BC8	mov rcx,rbx	rbx:L"C:\\"
000001B38C9A345C	E8 EBD2FFFF	call 1B38C9A074C	
000001B38C9A3461	48: 85C0	test rax,rax	
000001B38C9A3464	75 92	jne 1B38C9A33F8	
000001B38C9A3466	48: 8D97 05030000	lea rdx,qword ptr ds:[rdi+305]	rdx:L"appdata", rdi+305:L"tor browser"
000001B38C9A346D	48: 8BC8	mov rcx,rbx	rbx:L"C:\\"
000001B38C9A3470	E8 D7D2FFFF	call 1B38C9A074C	
000001B38C9A3475	48: 85C0	test rax,rax	

[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

000001B38E840000	4C:8BD1	mov r10,rcx	
000001B38E840003	B8 35000000	mov eax,35	35:'s'
000001B38E840008	0F05	syscall	NtQueryDirectoryFile
000001B38E84000A	C3	ret	
000001B38E84000B	0000	add byte ptr ds:[rax],al	
000001B38E84000D	0000	add byte ptr ds:[rax],al	
000001B38E84000F	0000	add byte ptr ds:[rax],al	
000001B38E840011	0000	add byte ptr ds:[rax],al	
000001B38E840013	0000	add byte ptr ds:[rax],al	
000001B38E840015	0000	add byte ptr ds:[rax],al	

[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)

000001B38E820000	4C:8BD1	mov r10,rcx	
000001B38E820003	B8 55000000	mov eax,55	55:'U'
000001B38E820008	0F05	syscall	NtCreateFile
000001B38E82000A	C3	ret	
000001B38E82000B	0000	add byte ptr ds:[rax],al	
000001B38E82000D	0000	add byte ptr ds:[rax],al	
000001B38E82000F	0000	add byte ptr ds:[rax],al	
000001B38E820011	0000	add byte ptr ds:[rax],al	
000001B38E820013	0000	add byte ptr ds:[rax],al	
000001B38E820015	0000	add byte ptr ds:[rax],al	
000001B38E820017	0000	add byte ptr ds:[rax],al	
000001B38E820019	0000	add byte ptr ds:[rax],al	
000001B38E82001B	0000	add byte ptr ds:[rax],al	

000001B3F2110000	4C:8BD1	mov r10,rcx	
000001B3F2110003	B8 11000000	mov eax,11	
000001B3F2110008	0F05	syscall	NtQueryInformationFile
000001B3F211000A	C3	ret	
000001B3F211000B	0000	add byte ptr ds:[rax],al	
000001B3F211000D	0000	add byte ptr ds:[rax],al	
000001B3F211000F	0000	add byte ptr ds:[rax],al	
000001B3F2110011	0000	add byte ptr ds:[rax],al	
000001B3F2110013	0000	add byte ptr ds:[rax],al	
000001B3F2110015	0000	add byte ptr ds:[rax],al	
000001B3F2110017	0000	add byte ptr ds:[rax],al	
000001B3F2110019	0000	add byte ptr ds:[rax],al	
000001B3F211001B	0000	add byte ptr ds:[rax],al	

000001B3F2230000	4C:8BD1	mov r10,rcx	
000001B3F2230003	B8 06000000	mov eax,6	
000001B3F2230008	0F05	syscall	NtReadFile
000001B3F223000A	C3	ret	
000001B3F223000B	0000	add byte ptr ds:[rax],al	
000001B3F223000D	0000	add byte ptr ds:[rax],al	
000001B3F223000F	0000	add byte ptr ds:[rax],al	
000001B3F2230011	0000	add byte ptr ds:[rax],al	
000001B3F2230013	0000	add byte ptr ds:[rax],al	
000001B3F2230015	0000	add byte ptr ds:[rax],al	
000001B3F2230017	0000	add byte ptr ds:[rax],al	
000001B3F2230019	0000	add byte ptr ds:[rax],al	
000001B3F223001B	0000	add byte ptr ds:[rax],al	
000001B3F223001D	0000	add byte ptr ds:[rax],al	
000001B3F223001F	0000	add byte ptr ds:[rax],al	

000001B3F2130000	70 61 63 68 61 67 65 20 55 52 49 3A 3A 66 69 6C	Package URI::fil
000001B3F2130010	65 3A 3A 51 4E 58 38 0A 0A 75 73 65 20 77 61 72	e::QNX;..use str
000001B3F2130020	69 63 74 38 0A 75 73 65 20 77 61 72 6E 69 6E 67	ict;.use warning
000001B3F2130030	73 38 0A 0A 75 73 65 20 70 61 72 65 6E 74 20 27	s;..use parent "
000001B3F2130040	55 52 49 3A 3A 66 69 6C 65 3A 3A 55 6E 69 78 27	URI::file:Unix'
000001B3F2130050	3B 0A 0A 6F 75 72 20 24 56 45 52 53 49 4F 4E 20	;..our \$VERSION
000001B3F2130060	3D 20 27 35 2E 30 36 27 38 0A 0A 73 75 62 20 5F	= '5.06';..sub _
000001B3F2130070	66 69 6C 65 5F 65 78 74 72 61 63 74 5F 70 61 74	file_extract_pat
000001B3F2130080	68 0A 7B 0A 20 20 20 20 6D 79 28 24 63 6C 61 73	h.{ my(\$clas
000001B3F2130090	73 2C 20 24 70 61 74 68 29 20 3D 20 40 5F 38 0A	s, \$path) = @_;
000001B3F21300A0	20 20 20 20 23 20 74 69 64 79 20 70 61 74 68 0A	# tidy path.
000001B3F21300B0	20 20 20 20 24 70 61 74 68 20 3D 7E 20 73 2C 28	\$path =~ s,(
000001B3F21300C0	2E 29 2F 2F 2B 2C 24 31 2F 2C 67 38 20 23 20 5E	)/+,\$1/,g; # A
000001B3F21300D0	2F 2F 20 69 73 20 63 6F 72 72 65 63 74 0A 20 20	// 's correct.
000001B3F21300E0	20 20 24 70 61 74 68 20 3D 7E 20 73 2C 28 2F 5C	\$path =~ s,(/\
000001B3F21300F0	2E 29 28 2F 2C 2F 2C 67 38 0A 20 20 20 20 24 70	.)+/,/;g; \$p
000001B3F2130100	61 74 68 20 3D 20 22 2E 2F 24 70 61 74 68 22 20	ath = \$path"
000001B3F2130110	69 66 20 24 70 61 74 68 20 3D 7E 20 6D 2C 5E 5B	if \$path =~ m,^[
000001B3F2130120	5E 3A 2F 5D 2B 3A 2C 2C 3B 20 23 20 6C 6F 6F 6B	^:/+;:; # look
000001B3F2130130	20 6C 69 68 65 20 22 73 63 68 65 6D 65 3A 22 0A	like "scheme:".
000001B3F2130140	20 20 20 20 24 70 61 74 68 38 0A 7D 0A 0A 31 3B	\$path;}.1;

[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)

000001B38C9A2E38	48:895424 20	mov qword ptr ss:[rsp+20],rdx	
000001B38C9A2E3D	44:8985 88010000	mov dword ptr ss:[rbp+188],r14d	
000001B38C9A2E44	FFD7	call rdi	CryptEncrypt
000001B38C9A2E46	85C0	test eax, eax	
000001B3F2130000	AS 82 4F 11	ID 03 5C 97 F7 90 47 79 E1 74 8D 51	W.O...+.Gyat.Q
000001B3F2130010	9F 28 36 F1	A4 AA 34 82 CD 3F 08 23 C2 68 E0 84	.(6h**4.I7.#Aka.
000001B3F2130020	22 59 06 D8	36 44 6E 78 0A 80 DF 1C A6 C9 CF F7	..Y.06Dnx.'B.;EI+
000001B3F2130030	18 98 9A 98	6D 8F F4 DA C7 65 30 07 90 93 58 53	...m20Ced...XS
000001B3F2130040	86 B4 41 0F	01 2C EC 53 D0 6D 66 7F C8 6F 4D A1	..A...3Dmf.EOMi
000001B3F2130050	1F 7D A1 3D	C4 41 98 2D EE 3E E4 AE 5A AC 1C AD	..}AA.-i-a#Z..
000001B3F2130060	43 A4 CF AB	BF 21 49 9E AA F5 3D 88 E1 6F FD 4F	CHI< I.*B=.4p00
000001B3F2130070	73 E2 8C C3	16 2C 17 43 AB 8A 5E 4E 18 60 90 1C	sA.A...Cg.AN. ..
000001B3F2130080	6D AF 1F 58	2F ED 39 18 D8 40 11 23 22 29 93 57	m.X/19.Da.#")W
000001B3F2130090	E1 80 D7 18	55 47 AE 31 A5 16 11 B6 51 3D E0 00	a.x.UG01y..fQ=a.
000001B3F21300A0	1E 83 00 E2	46 82 C8 18 2C C9 37 AB B9 31 AF 62	..af.E..E7<'I b
000001B3F21300B0	96 85 6C 28	EC 67 38 FD 8C F2 FB 58 57 49 63 21	..μ1±)ggy.büxwic!
000001B3F21300C0	E6 41 A4 AF	DB 11 F8 18 3A 6E 5E 5A EA 30 83 82	MA*0.U.:m'z0e*
000001B3F21300D0	17 0D 78 6C	53 48 F9 2D 8B 9C DD FC CA A7 53 CD	..x1shu-u..yüE5SI
000001B3F21300E0	D5 F3 31 6A	BA C3 24 FE 13 3F 81 60 54 E0 E2 6D	001j>Asb.7± TAam
000001B3F21300F0	06 17 0E 2C	7C C7 C6 CA B9 C5 B9 9A D3 42 8D 18	... C4E'A'.0B%
000001B3F2130100	9E 1A DC E9	D9 48 E5 8E 76 EE 28 33 64 F9 6F 10	..ÜeUHA.vi+3duo.
000001B3F2130110	88 58 12 2A	48 C4 DA 44 5F 10 56 C6 EF 8F 41 EB	.X.*HAÜD..V4i.Ae
000001B3F2130120	05 81 F2 1F	AD 49 B4 FD 81 11 21 06 A9 29 78 E1	..ö..I y..l.@)xä
000001B3F2130130	88 18 5E 58	DF 9B C6 73 29 B9 C8 80 BD 84 BF 3E	..Xb.As)'E.%¿>
000001B3F2130140	9A 90 5C E4	00 3F 3A 04 18 A1 45 66 AE 43 18 91	..ä.7:..iEf0C..
000001B3F2130150	69 02 43 E3	D1 AC 48 54 C1 46 1D 28 68 CD 47 AF	1.CAN-HTAF.+hIG
000001B3F2240000	4C:88D1	mov r10,rcx	
000001B3F2240003	B8 08000000	mov eax,8	
000001B3F2240008	0F05	syscall	NtwriteFile
000001B3F224000A	C3	ret	
000001B3F224000B	0000	add byte ptr ds:[rax],al	
000001B3F224000D	0000	add byte ptr ds:[rax],al	
000001B3F224000F	0000	add byte ptr ds:[rax],al	
000001B3F2240011	0000	add byte ptr ds:[rax],al	
000001B3F2240013	0000	add byte ptr ds:[rax],al	
000001B3F2240015	0000	add byte ptr ds:[rax],al	
000001B3F2240017	0000	add byte ptr ds:[rax],al	
QNX.pm	2021-01-15 오전 2:01	PM 파일	1KB
Unix.pm	2021-01-15 오전 2:01	PM 파일	1KB
Win32.pm	2021-01-15 오전 2:01	PM 파일	2KB

[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)

000001B3F2170000	4C:8BD1	mov r10,rcx	
000001B3F2170003	B8 27000000	mov eax,27	27:'''
<b>000001B3F2170008</b>	<b>0F05</b>	<b>syscall</b>	<b>NtSetInformationFile</b>
000001B3F217000A	C3	ret	
000001B3F217000B	0000	add byte ptr ds:[rax],al	
000001B3F217000D	0000	add byte ptr ds:[rax],al	
000001B3F217000F	0000	add byte ptr ds:[rax],al	
000001B3F2170011	0000	add byte ptr ds:[rax],al	
000001B3F2170013	0000	add byte ptr ds:[rax],al	
000001B3F2170015	0000	add byte ptr ds:[rax],al	
<b>000000A73DFFC888</b>	<b>000001B38C9A271D</b>	<b>000001B38C9A271D도 반환 (종말: ???)</b>	
000000A73DFFC890	FFFFFFFFFFFFFFFF		
000000A73DFFC898	000000A73DFFC908		
000000A73DFFC8A0	000001B3F2170000		
000000A73DFFC8A8	0000000000000100		
000000A73DFFC8B0	000001B30000000A		
000000A73DFFC8B8	000001B300000004		
000000A73DFFC8C0	0000000000000100		
<b>000000A73DFFC8C8</b>	<b>000001B3F2150000</b>	<b>L"\\??\C:\xampp\perl\lib\URI\file\QNX.pm.hijhvoy"</b>	
000000A73DFFC8D0	000001B3F0470000	<b>L"\\??\C:\xampp\perl\lib\URI\file\QNX.pm"</b>	
<input type="checkbox"/> OS2.pm.hijhvoy	2022-07-25 오후 5:18	HIJHVOY 파일	1KB
<input type="checkbox"/> QNX.pm.hijhvoy	2022-07-25 오후 5:26	HIJHVOY 파일	1KB
<input type="checkbox"/> Unix.pm	2021-01-15 오전 2:01	PM 파일	1KB

[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)

높음 악성코드 Ransomware.Unknown rundll32.exe

위협 개요 위협 행위

높음 impact.encrypt.many-files

이벤트 발생 일시 : 2022-07-26 14:30:01

위험도 : 10

중간 impact.encrypt.file

이벤트 발생 일시 : 2022-07-26 14:29:59

위험도 : 6

MITRE ATT&CK 정보 :

No.	Tactic	Technique
1	Impact	(T1486) Data Encrypted for Impact

[알림] 악성코드가 탐지되었습니다.(EDR) 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

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) 2022 (주) 소만사 All rights reserved.

궁금하신 점이나 문의사항은 [malware@somansa.com](mailto:malware@somansa.com) 으로 문의주십시오