안드로이드 악성 앱 패커 식별 및 언패킹 시스템 구현

강민영[1], 서동훈[2], 전유민[3], 김관영[4]
[1]중부대학교 정보보호학전공
[2]중부대학교 정보보호학전공
[3]중부대학교 정보보호학전공
[4]OpenUp
kcy121206@jmail.ac.kr, okko2929@jmail.ac.kr, yumin3404@jmail.ac.kr, gy741.kim@gmail.com

Identification of Android malicious app packer and implementation of unpacking system

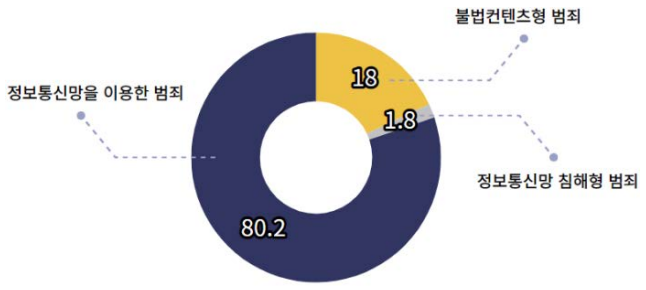
Min-Young Kang[1], Dong-Hun Seo[2], Yu-Min Jeon[3], Gwan-Yeong Kim[4]
[1]Dept. of Information Security, Joong-Bu University
[2]Dept. of Information Security, Joong-Bu University
[3]Dept. of Information Security, Joong-Bu University
[4]OpenUP

요 약

스마트폰 사용자 수가 증가함에 따라 스미싱, 몸캠피싱, 메신저 피싱과 같은 정보통신망을 이용한 범죄가 큰 폭으로 증가하고 있다. 이러한 범죄 피해는 다양한 연령층에서 발생하고 있다. 본 논문에서는 국내 모바일 운영체제 점유율이 가장 높은 안드로이드 운영체제를 대상으로 하는 패킹된 악성 앱 언패킹을 수행하고 시그니처 기반 탐지 도구인 Yara 를 통해 악성 앱에 사용된 패커를 식별하는 통합 악성 앱 언패킹 시스템을 제공하여 악성 앱을 이용한 범죄 대응에 도움을 줄 수 있을 것으로 기대된다.

1. 서론

최근 들어 스마트폰 범죄가 급증하고 있는데, 실제로 경찰청에 따르면 2021 년 전체 사이버 범죄 217,807 건 중에서 정보통신망을 이용한 범죄가 80.2% 를 차지하였다.[1] 이러한 범죄에 악용되는 악성 앱은 보안 기법을 회피하기 위해 주로 패킹 기법을 이용한다.



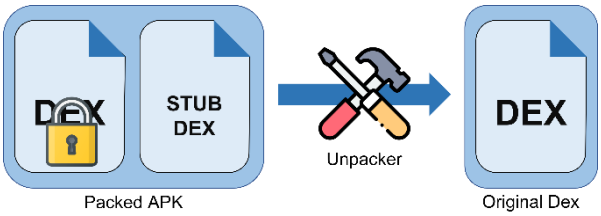
(그림 1) Occurrence rate by type of cybercrime.

본 논문에서는 2 장과 3 장에서 분석한 언패커에 대한 설명 및 결과를 정리하고, 4 장과 5 장에서 Yara 에 대한 설명 및 대표적인 패커를 대상으로 탐지된 시그니처 결과를 정리하고 마지막으로 6 장과 7 장에서는 결론을 도출한다.

2. 언패커 과정 및 원리

안드로이드 앱은 여러 파일을 하나로 압축한 파일을 의미한다. 이 중에는 DEX(Dalvik Executable) 파일이 존재하는데 해당 파일은 바이트 코드로 이루어져 있어 역공학에 취약한 문제가 존재한다. 이러한 점을 해결하기 위해 Dex 파일을 보호하는 패킹 기법이 사용되고 있다. 일반적으로 패킹된 앱은 암호화된 Dex 파일과 Stub Dex 파일로 구성되어 있으며, 실행 시 Stub Dex 파일을 통해 암호화된 Dex 파일을 복호화한다. 그리고 언패커는 패킹된 앱의 Dex 파일 및 앱을 실행하며 발생하는 패커에서 사용하는 라이브러리, 메모리를 이용하여 원본 Dex 파일을 구하는 도구를

의미한다.



(그림 2) Unpacking process.

3. 언패커 분석

언패커 분석은 오픈소스로 공개된 Native Unpacker[2]와 Frida Unpacker[3]를 대상으로 진행하며, 언패킹 결과는 Packergrind[4]에서 제공하는 샘플을 통해 진행하였다. 샘플 앱 파일은 6 개의 패커(Alibaba, Baidu, Bangcle, Ijiami, Qihoo, Tencent)를[5] 사용하여 만들었으며, 사용된 버전은 각 연도에 사용된 버전을 의미한다.

3.1 Native Unpacker

Native Unpacker 는 Defcon 22 에서 공개된 언패커이다. 동작 원리는 프로세스와 스레드 메모리에 접근한 다음, 패커별 시그니처를 통해 사용된 패커의 종류를 식별하고, Dex/Odex 파일의 Magic Number 를 사용하여 Dex 파일 혹은 Odex 파일을 추출하는 흐름으로 진행된다. 샘플을 통해 언패킹을 진행한 결과, <표 1>과 같은 결과를 확인할 수 있었다.

<표 1> Extract the original dex file - Native Unpacker

버전 / 패커	15	16	18	19
Alibaba	O	X	-	-
Baidu	X	O	O	X
Bangcle	X	X	X	X
IIiami	O	X	O	X
Qihoo	O	O	O	X
Tencent	X	X	X	X

*'○' = 성공, 'X' = 실패, '-' = 결과 없음

3.2 Frida Unpacker

Frida Unpacker 는 Frida 도구를 사용하여 제작된 언패커이다. 동작 원리는 안드로이드 버전을 확인한 다음 안드로이드에서 Dex 파일을 파싱할 때 사용하는 함수를 후킹한 후 해당 함수로 넘어오는 Dex/Odex 파일들을 추출하는 흐름으로 진행된다. 앞에서 살펴본 바와 같이 샘플을 통해 언패킹을 진행한 결과, <표 2>와 같은 결과를 확인할 수 있었다.

<표 2> Extract the original dex file - Frida Unpacker

버전 / 패커	15	16	18	19
Alibaba	O	O	-	-
Baidu	O	O	O	O
Bangcle	O	O	O	O
IIiami	O	O	X	X
Qihoo	O	O	X	X
Tencent	O	O	X	X

*'○' = 성공, 'X' = 실패, '-' = 결과 없음

4. Yara 원리 및 과정

Yara 는 악성 파일을 시그니처 기반으로 식별할 수 있게 해주는 도구이며, 리눅스와 윈도우 운영체제에서 모두 사용이 가능하다. Yara 의 시그니처 탐지는 대표적으로 문자열 탐지와 바이너리 탐지가 존재한다.

```
rule Alibaba : packer
{
  strings:
    $lib = "libmobisec.so"

  condition:
    $lib
}
```

(그림 3) Yara 문자열 탐지 방법

문자열 탐지는 Value 타이틀에 속해 있는 문자열들을 탐지하는 방법이며, Condition 을 이용하여 결괏값이 참인지 거짓인지 식별한다.

```
rule is_apk : file_type
{
  strings:
    $zip_head = {50 4B ?? ??}

  condition:
    $zip_head
}
```

(그림 4) Yara 바이너리 탐지 방법

바이너리 탐지는 파일 내부의 Hex 값을 탐지하는 기법으로 문자일 뿐만 아니라 16 진수 값을 Rule 에 적용할 수 있으며, (그림 4)와 같이 Wild Cards 를 사용하여 바이트를 랜덤으로 대체가 가능하다.

본 논문에서는 문자열 탐지 방법을 이용하여 악성 앱 파일 내부에 존재하는 라이브러리나 리소스 파일을 탐지하여 어떠한 패커가 사용되었는지 식별할 수 있는 방법론을 제시한다.

5. Yara 시그니처 탐지를 통한 패커 식별

다음 <표 3>은 총 6 개의 샘플 패커를 분석하여 각 패커별로 탐지된 시그니처 결과를 나타낸다. 여기서 탐지된 시그니처는 각 패커마다 사용되는 라이브러리나 리소스 파일을 의미한다. 따라서 각 패커마다 사용되는 라이브러리나 리소스 파일이 모두 다르기 때문에 Yara 로 악성 앱 파일을 분석하게 되면 탐지된 시그니처를 통해 어떠한 패커가 사용되었는지 식별이 가능하다.

<표 3> Yara-Rules Signature Detection Results by Packer

패커	시그니처 탐지 (라이브러리, 리소스)
Alibaba	libmobisec.so
Baidu	libbaiduprotect.so
Bangcle	libsecexe.so libsecmain.so bangcle_classes.jar
Ijiami	assets/ijiami.dat assets/ijm_lib/
Qihoo	libprotectClass.so
Tencent	lib/armeabi/libshell.so

6. 언패킹 및 패커 식별 통합 개발

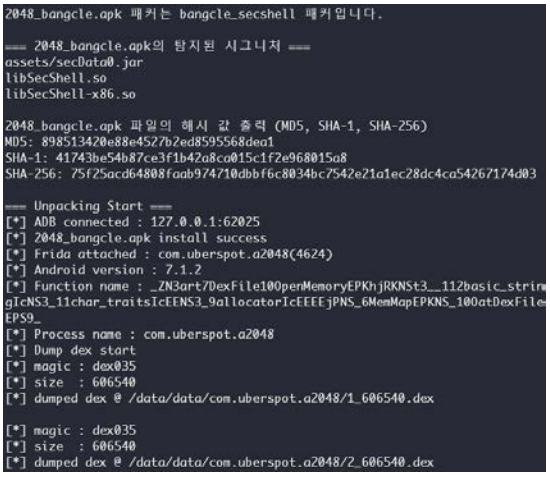
기존 언패커 도구와 Yara 도구를 이용한 패커 식별 기능이 분리되어 있어 악성 앱 분석에 많은 시간이 소요되었지만, 본 연구를 통해 개발된 도구는 기존에 개발된 언패커의 언패킹 기능과 Yara 의 시그니처 탐지 및 패커 식별 기능을 통합하고, 해시값 산출 기능을 추가하여 악성 앱 분석에 들어가는 시간을 줄일 수 있었다.

7. 결론

오픈소스로 공개된 Native Unpacker 와 Frida Unpacker 를 분석하여 원본 Dex 파일을 추출하는 과정과 <표 1>, <표 2>와 같이 언패커 결과를 확인할 수 있었다. 언패커 개발은 결과를 통해 확인할 수 있듯이 성능이 우수했던 Frida Unpacker 를 기준으로 개발을 진행하였다.

또한 패커 식별을 위하여 오픈소스로 공개된 Yara 시그니처[6]를 응용하여 <표 3>과 같이 각 패커마다 사용되는 라이브러리나 리소스 파일을 식별하고 어떠한 패커가 사용되었는지 식별할 수 있었다.

최종적으로 악성 앱 파일을 업로드하면 사용된 패커 식별과 탐지된 라이브러리 및 리소스 확인 후 원본 Dex 파일 추출을 진행한다. 마지막으로 악성 앱을 식별할 수 있도록 MD5, SHA-1, SHA-256 해시값을 산출하는 도구를 개발할 수 있었다.



(그림 5) 개발한 도구 출력 값

- 본 논문은 과학기술정보통신부 정보통신창의인재양성사업의 지원을 통해 수행한 ICT 멘토링 프로젝트 결과물 입니다 -

참고문헌

[1] "2022 년 사이버범죄 트렌드", 본청 국가수사본수 사이버수사국 사이버수사기획과, 2022

[2] {JCASE, DIFF}, “ANDROID HACKER PROTECTION LEVEL 0”, Defcon 22, 2014

[3] fridroid-unpacker Github https://github.com/enovella/fridroid-unpacker

[4] Packergrind Github, https://github.com/rewhy/adaptiveunpacker

[5] {Xue, Lei and Zhou, Hao and Luo, Xiapu and Yu, Le and Wu, Dinghao and Zhou, Yajin and Ma, Xiaobo}, “PackerGrind: An Adaptive Unpacking System for Android Apps”, IEEE Transactions on Software Engineering, 12p, 2020

[6] APKiD Github, https://github.com/rednaga/APKiD/tree/master/apkid/rules/apk