OBFUSCATION

컴퓨터과학전공
빈서윤

What is Obfuscation?

automatic or maual code anaylsis에 대응하기 위해 code를 less obvoius하게 수정

장점

prevention of
reverse engineering,
protection of intellectual property,
reducing the size of an executable

Code obfuscation development

malware development

-Brain Virus

-Cascase virus => use encryption to hide its ture propose

이 시기의 malware detectors은 code의 encrypted 된 부분을 찾을 수 있을 만큼 정교화 되지 않음

Malware Detectors

Before rapid growth of Internet

오직 malware detectors는 오직 알려진 바이러스 프로그램의 signature에 기반한 검사만 수행

After introduction of the WWW

antiivirus industry가 확장
됨
-Firewalls
-Online scanning
-virtual machines

Malware Detectors

most common approach

-Signature Based Detection
- heuristic based Detection
- rootkit based detection

Active scanning approaches

on-access
approaches

1.Signature Based Detection

most basic methods of malware detection

-악성코드가 “wild”에서 발견 => 이를 분석하고 “signature” 추출(수동 or 자동화된 탐지 기법으로 수행)

-일단 signature 가 발견되면 , antivirus software의 malware definitios에 update 된다.

-generc malware에는 효과적, but oligomorhic, ploymorhic, metamorphic malware에는 비효과적

2.Heuristics Based Detection

“multiple malware are created from a single malware”
라는 fact에 기반한 기술

⇒ 모든 malware는 paticular family에 연관이 있다는 것을 사용해서 detection 진행

3.Rootkit Detection

-full administrative access를 가짐

-have the ablility to hide themselves from list of running process

-다른 generic malware programs에 비해 제거하기 어려움

4.On -Access Scanning

antivirus program은 real-time에 발생할 수 있는 threats 주시하고 있음

ex> USB drive is inserted, an email attachment is opened

⇒ malware defintions에 의존해서 virues를 detect하는 게 아니기에 more efffective하다.

Obfuscapk

-ADAM, AAMO tools => limitaion

-free Python tool that is able to obfuscate compiled Android apps(소스코드 필요 없이)

-supports advanced obfuscation features

-community가 쉽게 기술 추가 할 수 있음

-aims= developers와 research communities 모두에게 useful

PERSONAL SKILLS

Obfuscapk

Developers

-developers can use Obfuscapk in cooperation with ProGuard
⇒ default optimizer이고, Android SDK에 포함된 obfuscator이고, 공식 Android Studio IDE의 지원을 받음

Reserarch community

-Obfuscapk를 black-box obfucation tool to apps and malware samples로 사용
ex> building or attacking a machine learning model, improving program analysis techniques

최종적으로 각 사용자들은

can extend the current tool by adding her own or other obfuscation techniques at state of the art.

OBFUSCATION TECHNIQUES

TRIVAL

NON-TRIVAL

Supported obfuscation techniques.

-speicific focus on malware detection techniques

-Android system에서 obfuscation techniques를 2가지로 classification
-trival, non-trival

Trival techniques

No real Obfuscation effects on APK

But anti-malware tools 을 다룸

Trival techniques - Align and resign

-Last mandantory steps for building a working Android APK
-The alignment is done by using zipalign(specific tool of the Android SDK)
⇒Android device에 최적화된 파일 구조를 가진 application 이 생성
-Android 는 모든 APK가 device에 install or update 되기 전 인증서로 디지털 서명이 되어야 함
⇒Re-sing step is the last mandatory step after applying obfuscation

Trival techniques - Rebuild

-classes.dex 파일에 들어 있는 bytecode 가 disassembled(분해), reassembled(재조립) ⇒
obtain different version of the file

-semantic changing 없이 bytecode를 변경 ⇒ app의 original behavior 보존 가능

-rebuild의 목적은 classes.dex 파일의 signature를 사용하는 anti-malware tools(악성코드 탐지 도구)를 혼란시키는 것

Trival techniques - Randomize manifest

-XML tree structure 변경 없이 AndroidManifest.xml 의 항목의 순서들을 무작위로 재배열

-Two goals

1.change the hash of the manifest file

2.fool the N-gram analysis

Non-Trival techniques

-더 복잡하지만 더 적절한 gain을 보장

-obfuscation의 target은 btyecode와 resources(XMLs, asset files, and external libraries)

Non-Trival techniques-Renaming

-보통 identifiers(variable names, functions names 등)은 readablility , maintainablility를 위해 meaningful 하게 작성
But, clear name은 정보를 유출 시킬 위험이 있음
⇒ obscure and meaningless 한 이름으로 renaming

-주의 사항 :
methods, fields 등을 renaming하는 건 결함이 적은데, classes나 package name의 renaming은 coplicated (Androidfest.xml도 이에 따라 변경되어야 해서)

Non-Trival techniques-Encryption

APK file에 runtime 시 developer가 요청할 수 있는 resources file이 들어 있음 ⇒
해당 파일은 runtime시 encrypted(암호화) 되거나 decrypted(복호화) 됨 ⇒ resources 에 접근하기 위해 decrytion key가 필요함(resources에 접근하기 위해서 extra calculation가 필요해서 app performances 가 나빠질 수 있음) ⇒
 그래서 Obfuscapk 는 자동적으로 random secret key를 생성해서 암호화에 사용

Non-Trival techniques-Code

classes.dex 내부의 명령어들에 영향을 주는 모든 obfuscation tech 포함

Non-Trival techniques-Code

1.DebugRemoval

-debug meta-data제거

-line numer, types,or method name 같은 디버그 정보 제거

⇒ reverse engineering 에 유용한 정보를 줄여서 분석을 어렵게 만듦

Non-Trival techniques-Code

2.CallIndirection

-원래 method를 호출하는 새로운 매소드를 추가해서 method call를 indirection하게 만드는 것

-cotrol-flow graph(CFG)를 바꿈

ex> m1에 대한 호출을 m2가 대체하고, m2가 내부에서 orginal method인 m1을 호출

Non-Trival techniques-Code

3.Goto

주어진 method에 매서드 끝을 가리키는 goto 명령어 삽입

그리고 fisrt goto 다음 명령을 가르키는 또다른 goto를 삽입

⇒ 새로운 two nodes를 추가해서 CFG 수정

Non-Trival techniques-Code

4.Reorder

basic block의 순서를 변경하는 방법

ex> goto instrction을 사용해서 code를 randomly 재배열

Non-Trival techniques-Code

5.ArithmeticBranch

junk code insertion

-code의 의미는 보존하면서 무의미한 instructions들을 추가

-주의
산술연산과 그 결과에 따라 branch 하는 명령어를 넣는데, branch가 never taken되게 설계해야 함

Non-Trival techniques-Code

6.Nop(no-operation)

every method implementaion에 아무것도 안하는 nop instruction 을 randomly하게 넣음

Non-Trival techniques-Code

7.MethodsOverload

-different methods 에 smame name을 부여하는 Javaprogramming 의 overloading 기능 사용

-이미 존재하는 매소드와 같은 이름과 인자를 가지는 void method를 만들고, 랜덤으로 새로운 arguments를 더함 ⇒ new method를 random arithmetic instructions으로 채움

Non-Trival techniques-Invocation by reflection

Java programming의 기법으로 실행 중에 run time 동작을 examing or modifting

⇒given object의 methods를 호출하는데 사용

SOFTWARE DESCRIPTION

SOFTWARE ARCHITECTURE

TOOL FUNCTIONALITIES

Software architecture

-Obfuscapk is designed to be modular and easy to extend

-each obfuscator(난독화기)는 abstract base class를 상속받고, obfuscate 매서드를 구현해야 함

-APK process 가 시작되면, all the needed information(location of decompiled code)와 internal state of the operations 들이 저장된 obfuscation 객체가 생성됨

Tool functionalities

필수 paramater가 2개 있음

- <APK_FILE>: 난독화할 APK 파일의 경로(상대경로나 절대경로 가능)
- -o : 적용할 난독화 기법 이름들의 목록

Illustrative example

Android APK는 여러 파일들을 포함하고, 악성 component가 거의 모든 곳에 implement될수 있다 ⇒
어떤 techniques의 조합이 가장 effective할지 미리 정하는 것을 불가하다. 각 기법이 APK 내 파일에 미치는 영향이 다 다르기 때문에

Table 2
Detection ratio of different obfuscated versions.

Category	Detection ratio	Percentage
Original	32/58 [29]	55%
Trivial	18/58 [31]	31%
Renaming	16/58 [32]	28%
Reflection	15/58 [33]	26%
Code	8/58 [34]	14%
Encryption	0/58 [35]	0%