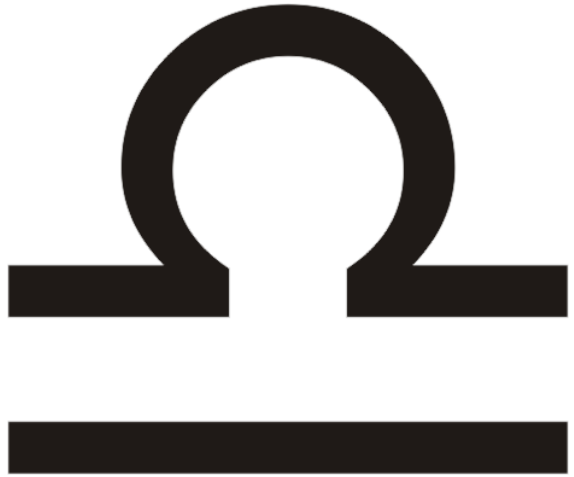
Static Android Malware Analysis

BY MAX ‘LIBRA’ KERSTEN

Table of contents

Who am I?

- Max ‘Libra’ Kersten (@LibraAnalysis)
- Graduated my bachelor cum laude in January
- Worked for ThreatFabric as an Android malware analyst
- I write blogs about reverse engineering
  - Including my own Binary Analysis Course
- Custom tools are released open-source on my Github
  - AndroidProjectCreator is featured in this workshop



Who am I?

- Employed at ABN AMRO
  - Cyber Threat Intelligence & Analytics team
  - Red Team
- Focus on outside threats to provide timely and actionable intelligence to internal departments
- Research focused projects, with the aim to also give something back to the community

What is the workshop about?

- Provides insight in Android malware analysis
- Teaches core concepts of reverse engineering
- Purely focused on static code analysis
  - What is the difference?
- Duration is between 3 and 4 hours
  - Hence the approximation on the online schedule

The Android operating system

- Applications require permissions, before certain actions can be executed
- Applications are sandboxed, making direct process interaction impossible
- System updates are not pushed aggressively
- Multiple phone vendors
  - Fragment the updates even more
  - Have a different ‘code base’

The lay-out of an Android PacKage

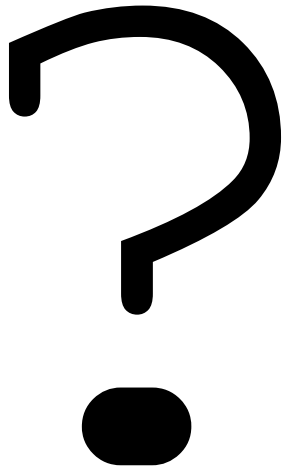
- The `AndroidManifest.xml` file
  - Contains all required permissions
  - Services and intent filters are declared in here as well
- The `classes.dex` file
  - Contains the compiled classes
  - Multiple versions can exist, using `classesN.dex` as naming scheme
- The `resources.arsc` file
  - Contains embedded resources, such as the used views
- The `META-INF` folder
  - Certificate information, can be used to identify the developer
  - Also usable in Yara rules

The lay-out of an Android PacKage

- The `lib` folder
  - Contains native `ELF` libraries that the application uses
  - Libraries are present in multiple architectures
    - `x86`, `x86_64`, `armeabi-v7a`, `arm64-v8a`
- The `assets` folder
  - Contains arbitrary files that are used within the application
  - Malware uses this folder to store an encrypted `classes.dex` file
- The `res` folder
  - Contains resources that are used within the application
  - Examples are background images in different sizes

The analysis of an APK

- Where do I start?
- Do I need an additional phone?
- Should I install an emulator?
- Do I need to use a Linux distribution to work on?



Decompiling an APK

Decompiling an APK

Decompiling an APK

- APKTool to obtain the manifest, resources and Dalvik bytecode
- Dex2Jar to convert the Dalvik bytecode to Java bytecode
- A Java decompiler to obtain Java code
  - JD-GUI
  - Fernflower
  - JAD-X
  - CFR
  - Procyon

Decompiling an APK

- Direct decompilation/disassembly
  - Radare2 (with r2dec)
  - Ghidra
  - JEB
- Combine multiple tools using AndroidProjectCreator
  - Converts the APK into an Android Studio project
  - Leverages the power of Android Studio to analyse to code, including existing plug-ins

Platforms to find samples

- APKLab
- APKDetect
- Koodous
- VirusBay
- VirusTotal

The goal and strategy of the analysis

- Determine the goal of the analysis
  - Is this application malicious?
  - What applications are being targeted by this malware sample?
  - The Command & Control traffic of the bot is encrypted, can you decrypt it?
  - It is suspected that the malware uses a domain name generation algorithm, can you figure out how the domains are generated and provide a list of the first hundred domains?
- Determine the strategy
  - Dynamic analysis
  - Static analysis

Break

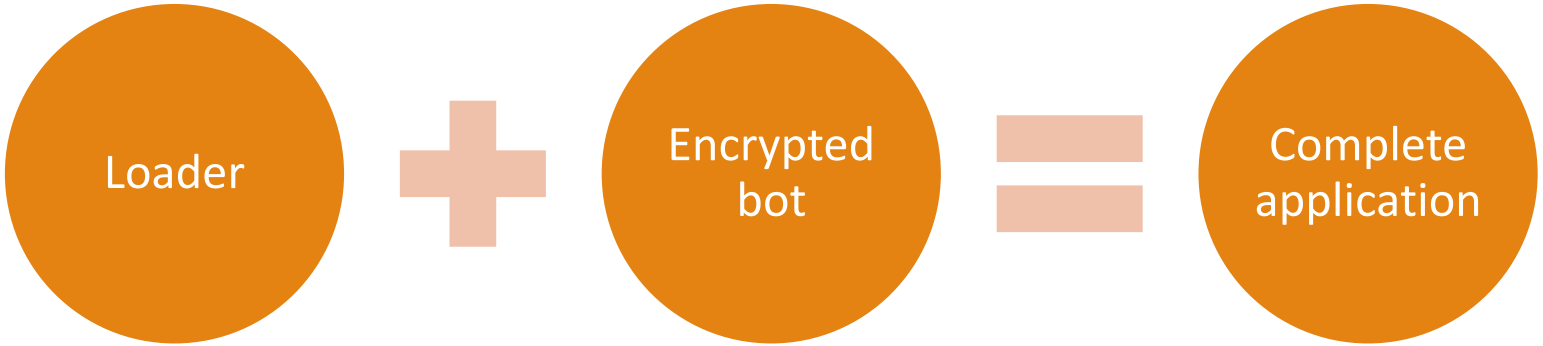
- Take a break, chat with your neighbours, and share some tips if you have them!

Statically analysing malware samples

- Two analysis methods
  - Loading a new `classes.dex` file
  - Finding the command switch
- Three common techniques
  - Default application replacement
  - Overlay attacks
  - Logging keystrokes

Loading a new `classes.dex` file

- The `classes.dex` file contains the compiled Java code
  - Results in additional capabilities being loaded
  - Or the original code only functions as a loader to evade detection systems
- Compared to the desktop platform, there are less
  - Packers available
  - Loading methods



Hands on task

- Analyse the given dropper to obtain the malicious `classes.dex` file

Break

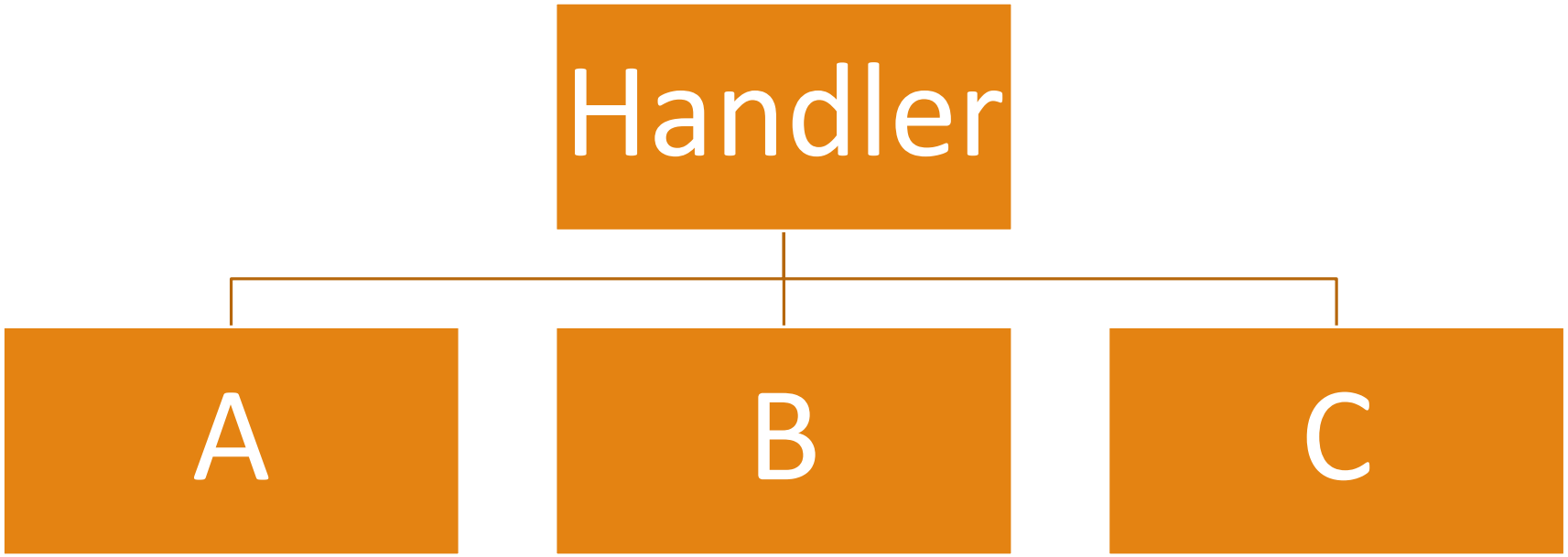
- Take a break, chat with your neighbours, and share some tips if you have them!

Finding the command switch

- To avoid complicating code, classes are used
- Classes serve a single purpose
- Classes can be objects, interfaces, models, views, containers, and much more

- A class that refers to a lot of other classes is often observed as a handler

Finding the command switch



Hands on task

- Analyse the given the given malware to find the command switch
- Figure out which commands there are, and what they do

Break

- Take a break, chat with your neighbours, and share some tips if you have them!

Default application replacement

- Used to manage certain utilities
  - Installing an improved SMS manager
- Often abused by malware as it grants valuable permissions
  - Steal Two Factor Authentication messages
  - Send out texts that link to malware
  - Obtain phone numbers

Hands on task

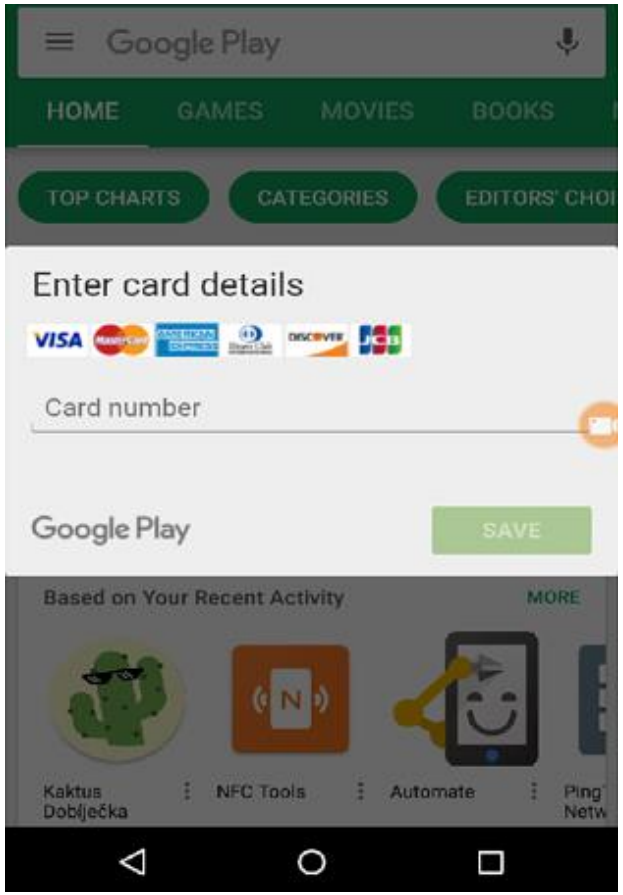
- Analyse the given malware and explain what the malware is capable of

Break

- Take a break, chat with your neighbours, and share some tips if you have them!

Overlay attacks – a synopsis

- An attack that is used to obtain
  - Credentials
  - Credit card information
- Time based attack based on user-input
- Different Android versions require different techniques
  - Actors and defenders play a cat and mouse game



Hands on task

- Find how and where the overlays are used
  - Trick bonus question: where are all the banking applications located?

Break

- Take a break, chat with your neighbours, and share some tips if you have them!

Logging keystrokes

- Unable to hook the system like the desktop platform
- Can abuse the `Accessibility Service` to get information on keystrokes
  - Easily detected by Google on the Play Store
  - Generic rules will find the sample
- Taking screenshots upon noticing a keypress
  - Password characters are shortly visible on the screen by default
  - Getting more contextual information
- Touch based location keylogging
  - Spotted in the wild and wrote about it
  - Works based on the location where a keypress is made

Hands on task

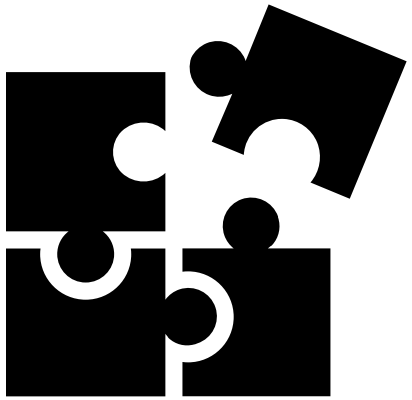
- Find out how the keylogger works in detail

Break

- Take a break, chat with your neighbours, and share some tips if you have them!

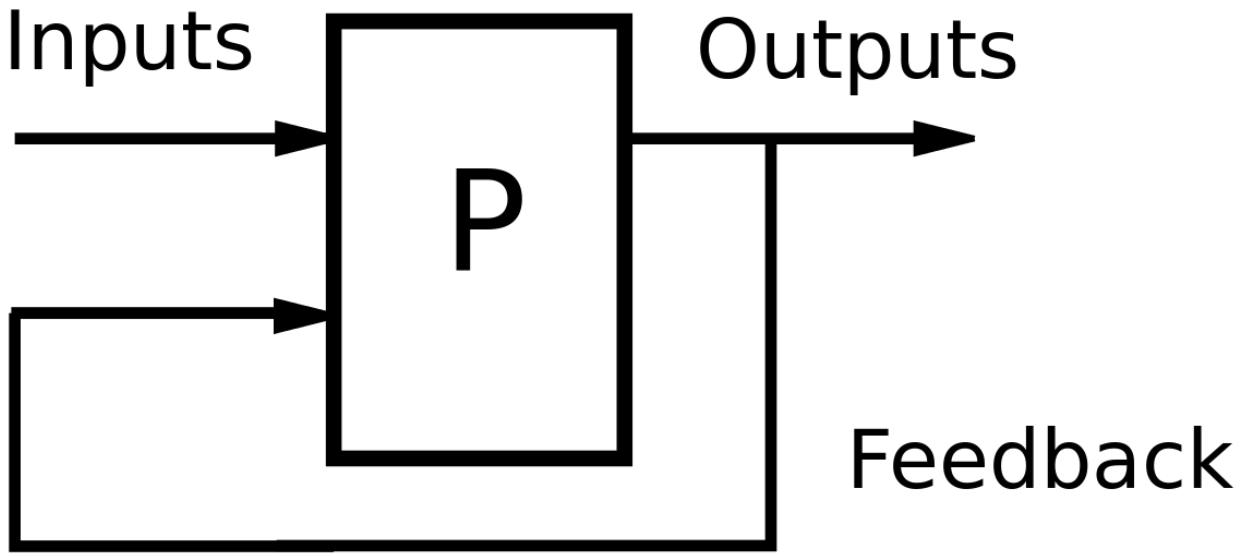
Takeaways

- Android application lay-out
- Decompiling an APK in various methods
- Determining the investigation’s goal and strategy
- Efficient analysis methods
- Common malicious techniques



Evaluation

- Please take a few minutes to fill in the evaluation sheet that is handed out
- Next iterations of this workshop will be improved based on your feedback



Surprise – platform access

- Please leave your name and email with me to get access to
  - APKDetect by Witold Precikowski (@pr3wtd)
  - APKLab by Avast