Course Guidelines

REVERSE ENGINEERING

deti universidade de aveiro
departamento de eletrónica,
telecomunicações e informática

João Paulo Barraca, Bernardo Cunha, José Luis Azevedo

Faculty

- João Paulo Barraca – jpbarraca@ua.pt
  - IT – Telecommunications and Networks - Aveiro

- Bernardo Cunha – mbc@det.ua.pt
  - IEETA - Intelligent Robotics and Systems

- José Luis Azevedo - jla@ua.pt
  - IEETA - Intelligent Robotics and Systems

Operational aspects

- Lectures in a mixed format: remote + in place (if possible)
  - According to the pandemic situation and actual lecture contents
- Contents: everything available in the Teams Channel
- Languages:
  - Classes may be lectured in English, but will default to Portuguese.
  - Contents will be available in English
- Communication:
  - Announcements will be made through Teams (and or elearning)
  - Direct communication through the Teams Channels. Participation is mandatory!
  - Email if required: jpbarraca@ua.pt, mbc@det.ua.pt, jla@ua.pt

Objectives

- Know the techniques to **identify the components of a system**
- Know the techniques to **observe the behavior** of systems and components
- Know the **methodologies for reverse engineering**
- Know the relevant protocols and technologies **to build systems, applications and devices**
- Understand the techniques, processes and tools for **decomposition of applications**

Objectives

- Understand the techniques, processes and tools for **decomposition of devices and systems**

- Understand the techniques, processes and tools for **decomposition of mobile applications**

- Capability to **perform tasks of reverse engineering**

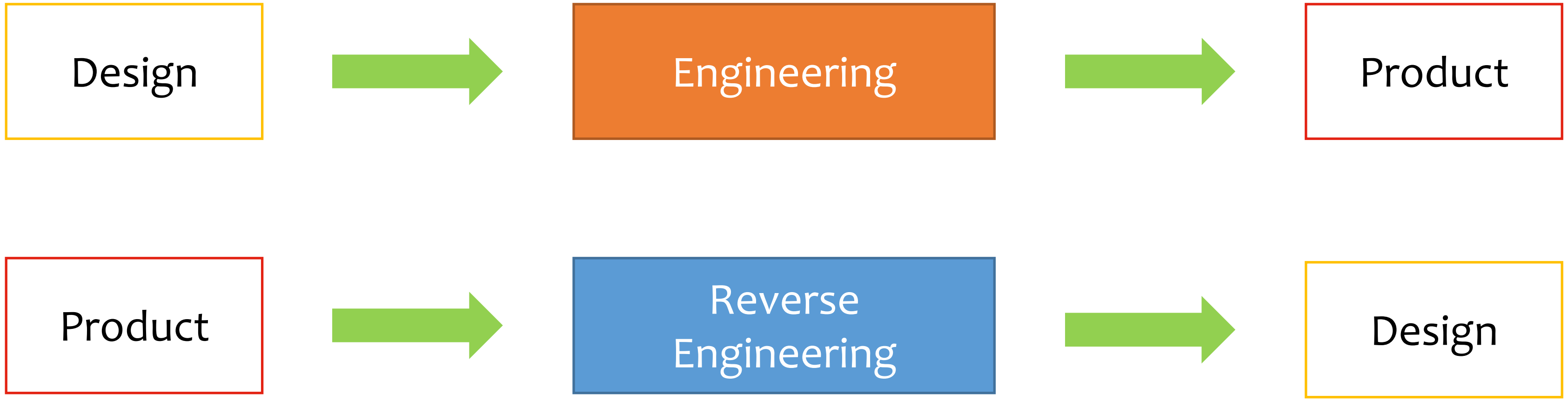
- Capability of **documenting the process of reversing engineering**

- Capability to **replicate components** analyzed through reverse engineering

Objectives

- This will not be a course about hacking, malware analysis, or cracking

- This will be a course about reconstructing software/systems from products



Syllabus

Intro, plus 3 main modules

0. Introduction ~1 week

1. Mobile Applications ~3-4 weeks

2. Binary Applications ~5-6 weeks

3. Devices ~5-6 weeks

Evaluation

- 3 assignments, to be implemented by groups of 2 students:
  - Android – 20%
  - Applications – 25%
  - Devices – 25%
  - Assignments should be returned ~2 weeks after the last lecture on the topic

- 1 final exam – 30%
  - In June/July

- Some variations may be required

Bibliography

- Will be provided in every lecture:
  - Books, papers, reports, videos

- Available on the O’Reily library:
  - A. P. David, Ghidra Software Reverse Engineering for Beginners, Packt Publishing, 2021, ISBN: 9781800207974
  - Bruce Dang, Alexandre Gazet, Elias Bachaalany, Practical Reverse Engineering: x86, x64, ARM, Windows Kernel, Reversing Tools, and Obfuscation, 2014, ISBN: 9781118787311
  - Philip Polstra, Reverse Engineering and Exploit Development, Infinite Skills 2015 (Video)
  - Eldad Eilam, Reversing: Secrets of Reverse Engineering, Willey, 2005, 9780764574818
  - Dennis Andriesse, Practical Binary Analysis, ISBN-13: 9781593279127, 2018

- Relevant website (links): https://beginners.re/main.html

Introduction to Reverse Engineering

REVERSE ENGINEERING

deti universidade de aveiro
departamento de eletrónica,
telecomunicações e informática

João Paulo Barraca

What is Reverse Engineering (RE)

- Reverse Engineering (RE) is the process of extracting features from any man-made artifact (Engineered)
  - Knowledge
  - Design blueprints
  - Function

- It’s not purely scientific research: with RE the artifact was engineered
  - The scientific process doesn’t generically focus on a product
    - Focus is on mechanisms, processes, events, phenomena
    - … and we have no idea whether the universe was engineered or not ☺

What is Reverse Engineering (RE)

The process of **analyzing** a **subject system** to **identify** the system’s **components** and their **interrelationships** and to **create representations** of the system in another form or at a higher level of abstraction

Chikofsky, E. J.; Cross, J. H. (January 1990). "Reverse engineering and design recovery: A taxonomy" (PDF). IEEE Software. 7: 13–17. doi:10.1109/52.43044

Forward Engineering

Images belong to their respective owners

What is Reverse Engineering (RE)

The process of **analyzing** a **subject system** to **identify** the system’s **components** and their **interrelationships** and to **create representations** of the system in another form or at a higher level of abstraction

Chikofsky, E. J.; Cross, J. H. (January 1990). "Reverse engineering and design recovery: A taxonomy" (PDF). IEEE Software. 7: 13–17. doi:10.1109/52.43044

1
2
3
4
5
6
7
8
9
10
11
12
CORELLIAN
ENGINEERING CORPORATION
YT-1300
MILLENNIUM FALCON

Reverse Engineering

Images belong to their respective owners

What is Reverse Engineering (RE)

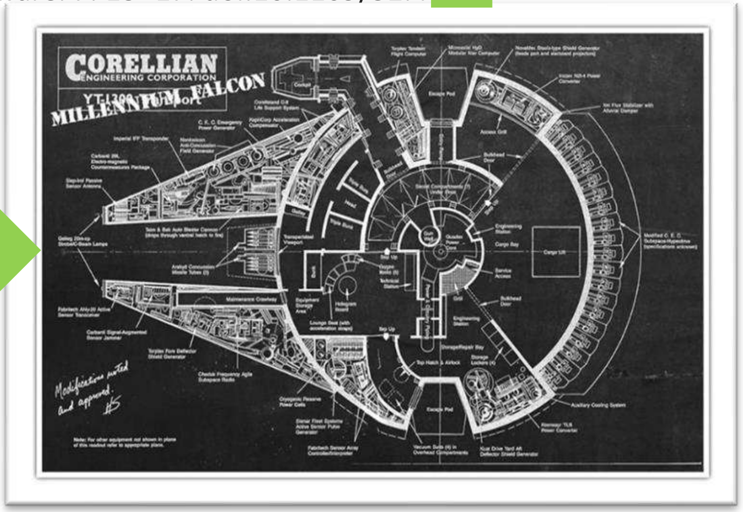
The process of **analyzing** a **subject system** to **identify** the system’

interrelationships and to **create representations** of the system i

higher level of abstraction

Chikofsky, E. J.; Cross, J. H. (January 1990). "Reverse engineering and design recovery: A taxonomy" (PDF). IEEE Software. 7: 13–17. doi:10.1109/52.4

1
2
3
4
5
6
7
8
9
10
11
12

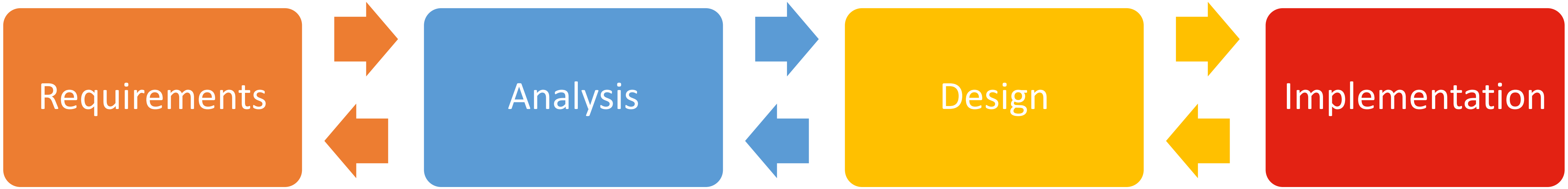


Reverse Engineering

Images belong to their respective owners

What is Reverse Engineering (RE)

Forward Engineering



Reverse Engineering

- Processes are not perfect, in either direction.
- Implementation may not fully comply with requirements, while reversed engineered analysis may not fully represent the implementation design, and design will be limited

RE Concepts

- **Abstraction Level**
  - The result of a RE process will produce a design at a given abstraction level.
  - The higher the better
- **Completeness**
  - Level of detail at the abstraction level.
  - The greater the better
- **Interactivity**
  - How much humans are required for RE.
  - The lesser the better (higher automation)

When do we have RE activities?

- RE **always evolved with engineering** and existed since its dawn
  - It is frequently done informally by everyone in their daily lives

- Every time **we look at a software/device/system** and try to understand how it works, or understand any aspect of its behavior and structure
  - Because we want to make a better one
  - Because we wish to estimate if it suits a purpose…

- Every time we **look at our code** and try to find what it was supposed to do
  - Especially when there is no documentation

Why RE is Relevant and Required

Personal Education

- Observing a product allows anyone to learn from its characteristics.
  - Why it behaves that way
  - What it does
  - How it does something
  - Why something doesn’t happen
- One can **complement engineering** education by observing code/products made by others
  - Open-source software plays an important role here
    - Because it the source is available, it doesn’t mean that structure, components, etc… are readily available or understood
  - Actually… instead of learning from patterns, **why not learn from its application as implemented by other professionals**?
    - There are a lot of “hidden” subtleties due to the experience of their authors

Why RE is Relevant and Required

Work around limitations

- Products are engineered in order to **provide some value**, and **turn profit**
  - Some value = value perceived by the buyers, in relation to other products
  - Profit = max price for the minimal cost
- Products are frequently built to promote further revenue
  - Support contracts, build an ecosystem, help sell other products
  - Closed in their interfaces and limited in their feature set
- Reverse engineering can be used **to increase the feature set**
  - After the product is made, and without cooperation from manufacturer

Why RE is Relevant and Required

Work around limitations

Magic Lantern extends existing Cameras with a huge amount of extra features

https://magiclantern.fm/

3D scanning vehicles enables aftermarket variants to produce alternative parts

https://www.creaform3d.com/

Observing existing parts allows new parts to be designed to improve reliability, performance, design..

Why RE is Relevant and Required

Make a product compatible

- A product is developed for a set of scenarios. **What if we want it to operate on another, unexpected, environment?**

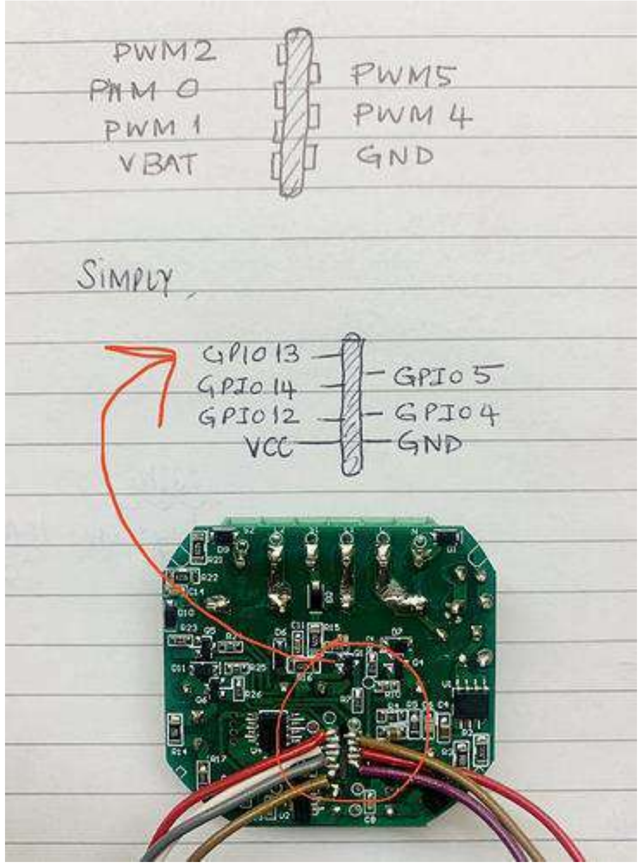
- RE allows obtaining **relevant design/operation information**
  - To modify the product to fit the new environment
    - Some components may be reconstructed
  - To build adapters integrating the product

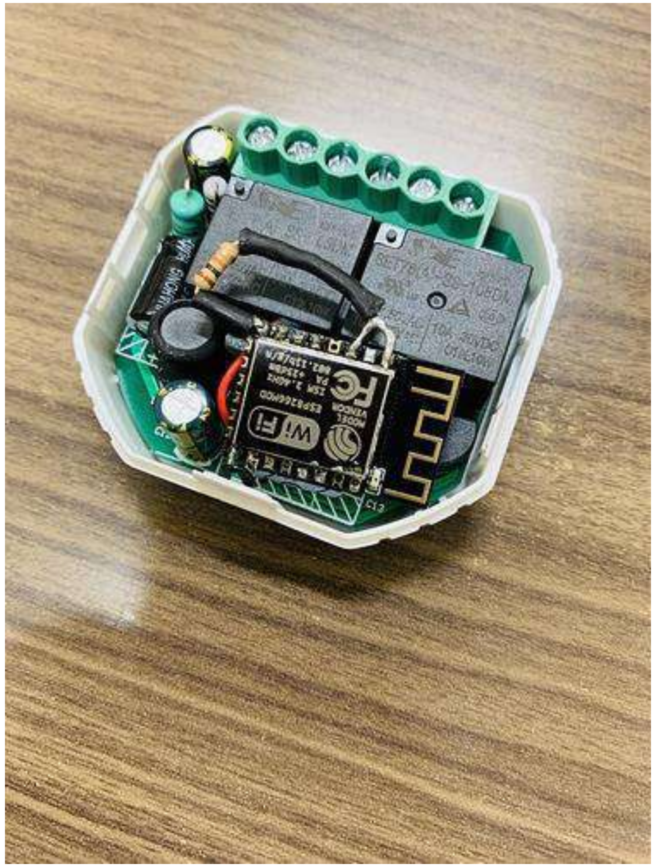
- In corporate world it's standard to have products adapted to a specific use case
  - Process takes a long time, and is expensive
  - RE may provide a simpler route
  - Especially relevant if the manufacturer doesn't provide that service
    - Or simply doesn't exist

Why RE is Relevant and Required

Make a product compatible

- Make/DIY movements are keen on RE
- Driven by integrating and enhancement
  - Mostly for personal use
  - Community driven
- Frequently without cooperation from manufacturers
  - Alarms: ParadoxAlarmInterface/pai
  - Sports bracelets: Gadgetbridge
- Sometimes with some collaboration
  - Magic Lantern





Unkown tuya chip - Hardware - Home Assistant Community (home-assistant.io)

Why RE is Relevant and Required

Learn from other's products or from products of other domains

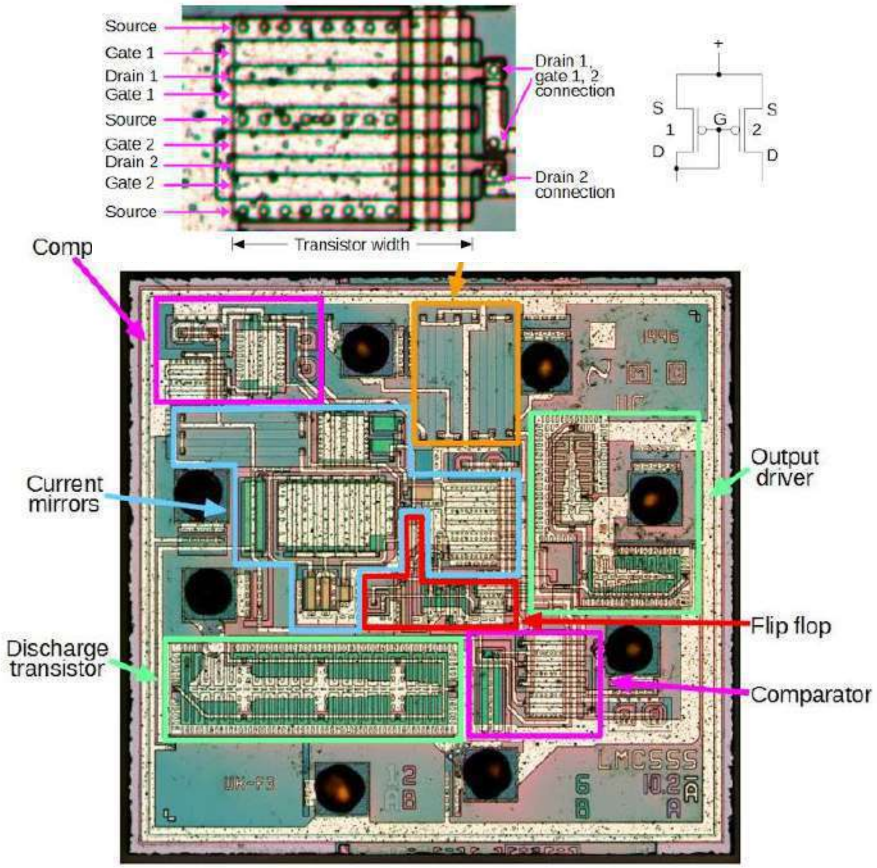
- Companies must determine the values/weaknesses of products in competing markets
  - What strategies/materials/methods/technology are used by competitors
  - Helps segmenting market and setting prices
  - Helps acquiring knowledge to develop new product

- Also: does a certain product violates a patent of ours?
  - Includes patented designs

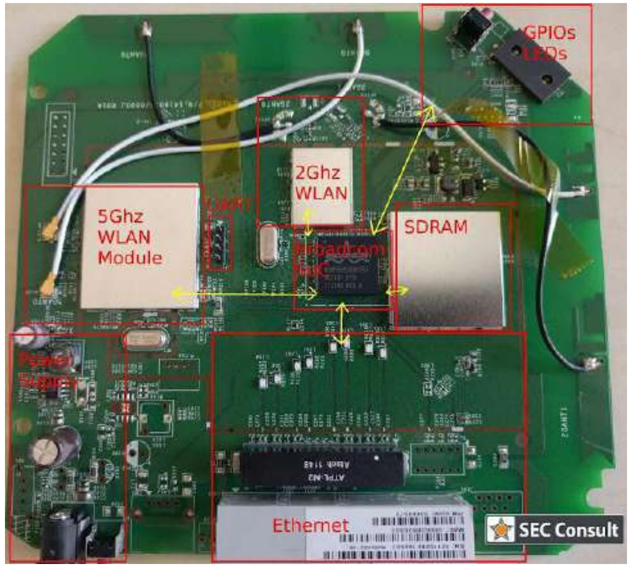
- RE can be used for that purpose
  - and can feed information to engineering
  - determine the need for judicial actions protecting Intellectual Property

Why RE is Relevant and Required

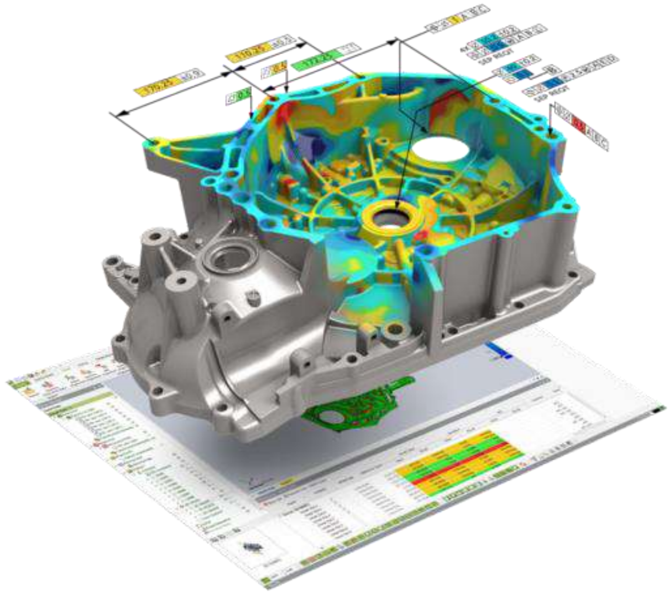
Learn from other’s products or from products of other domains



http://www.righto.com/2016/04/teardown-of-cmos-555-timer-chip-how.html



https://sec-consult.com/



https://dewyseng.com/

No affiliation with referred brands

Why RE is Relevant and Required

Finding the purpose of a certain code/binary blob or part

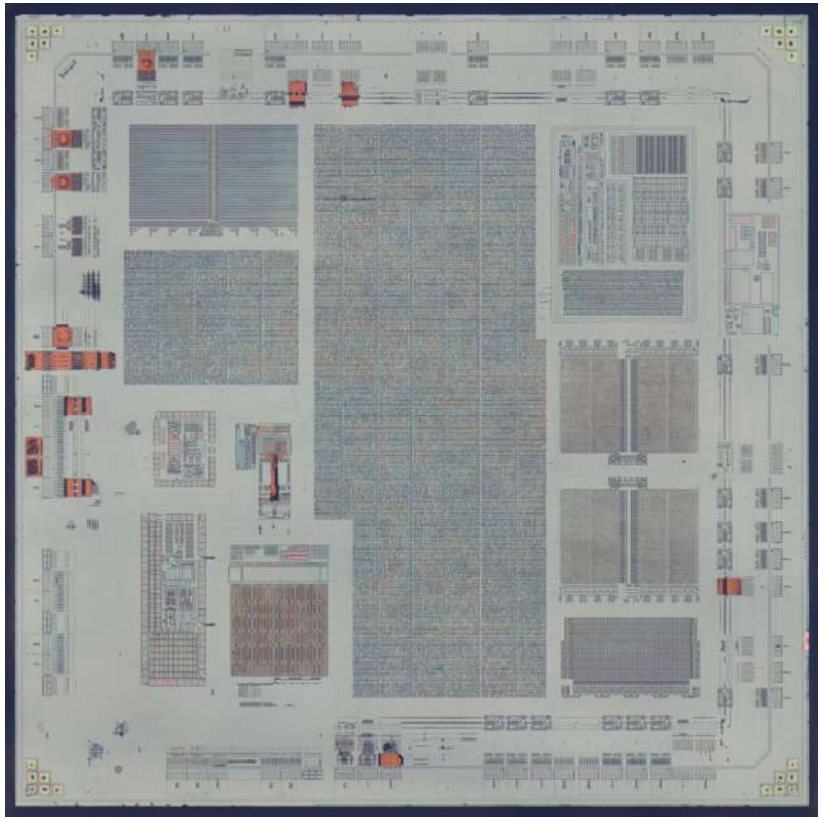
- Engineers frequently assume that an engineered entity is known (They trust dependencies)
  - That is… if you develop something, you know what it does
  - Also assume (or wish) that documentation exists

- What if:
  - documentation is lost?
  - the blob is external to the company?
  - the blob is misbehaving?
  - the blob was modified?
  - the engineer/supplier is not trusted?
  - the part is fake?
  - the company needs to validate the design process?

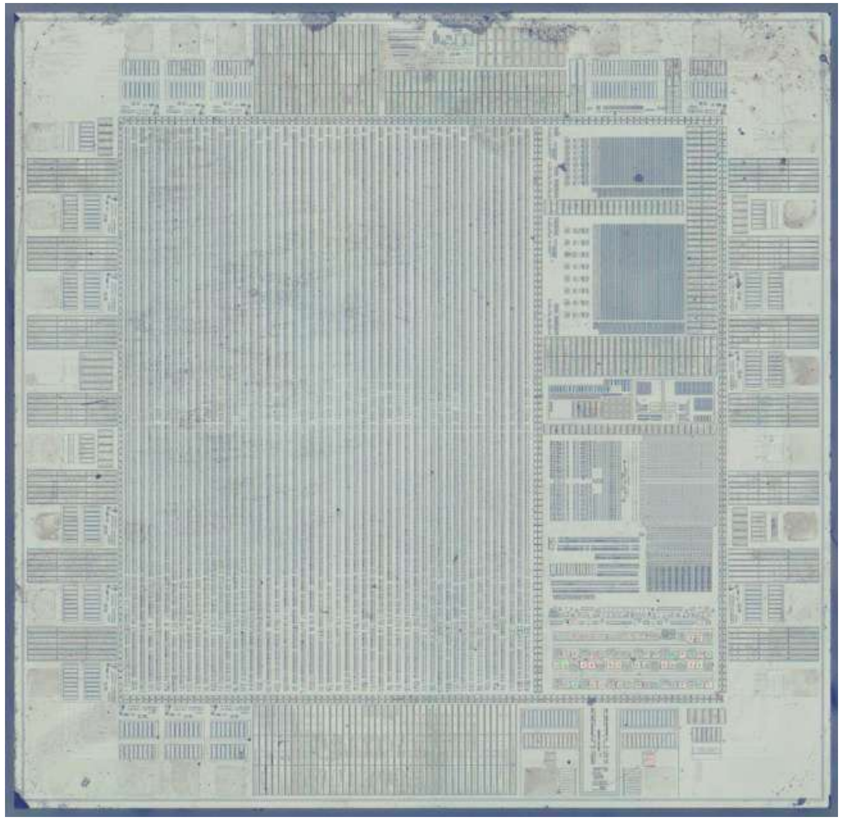
- RE can recover a similar design from the implementation, independently of the documentation, or the original design

Why RE is Relevant and Required

Finding the purpose of a certain code/binary blob or part



Fake FT232RL



Genuine FT232RL

https://zeptobars.com/en/read/FTDI-FT232RL-real-vs-fake-supereal

Why RE is Relevant and Required

Discovering flaws and faults

- Implementation may deviate from design
  - … it always deviates

- Implementation may present flaws due to unseen aspects
  - Processes used
  - Technology used
  - Interaction with additional components
  - Manufacturing flaws
  - Knowledge and experience

- RE is used in the scope of software testing to validate systems
  - Symbolic execution and Fuzzy testing are ways of helping the reverse engineering
  - Characterize if a given implementation reproduces the expected design
  - Identify additional modes

Why RE is Relevant and Required

Find and analyze malicious code

- For Anti-Virus, and Malware researchers, source code is not available
  - Or for offensive/red teams in black box scenarios

- **Malware detection relies on reverse engineering** to understand programs
  - RE allows the identification of patterns of malicious code
  - May rely on:
    - Interaction patterns
    - Bytecode structure
    - Communication with external hosts
    - Binary structure
    - Text contents
    - …

- Some RE is done in real time to find unknown malware
  - Or at least to identify suspect code, triggering further inspection

Limitations

- May be illegal in some cases, or lead to ambiguous situations
  - Higher risk of jeopardizing products developed
- Requires trained and experienced staff
  - Which is not abundant
- It’s costly in terms of time, resources and money
  - Expensive tools, scarce number of researchers, lengthy process
- May lead to incomplete or incorrect designs.
  - No guaranteed result!
  - An RE activity may be a complete waste of resources (time, staff, money)

Legal Framework

- The legality of RE is not assured a priori
  - varies with jurisdiction
  - varies with what is being reversed
  - varies with the purpose of the RE activity
  - varies with the impact to the product owner

- Applicable legislation:
  - USA: Digital Millennium Copyright Act
  - EU: EU Directive 2009/24

- This only applies to third parties
  - Product owners are free to use their own products as they seem fit
  - RE for the purpose of Software Quality Control

Legal Framework

Allowed situations (Europe, Directive 2009/24/EC)

The unauthorized reproduction, **translation**, **adaptation** or **transformation** of the form of the code in which a copy of a computer program has been made available **constitutes an infringement of the exclusive rights of the author**.

- .. circumstances may exist when such a reproduction of the code and translation of its form are indispensable to obtain the necessary information **to achieve the interoperability of an independently created program with other programs**.
- .. in these limited circumstances only, performance of the acts of reproduction and translation by or on behalf of **a person having a right to use a copy of the program** is legitimate and compatible with fair practice…

Legal Framework

Allowed situations (Europe, Directive 2009/24/EC)

- Article 5 b): To learn

The person **having a right to use a copy of a computer program** shall be entitled, without the authorisation of the rightholder, to **observe, study or test the functioning** of the program **in order to determine the ideas and principles** which underlie any element of the program **if he does so while performing any of the acts** of loading, displaying, running, transmitting or storing the program **which he is entitled to do**.

- **Broad Interpretation**: if you own a legitimate copy of the software, and are able to load it/run it/etc… you may analyze it for the purpose of learning

Legal Framework

Allowed situations (Europe, Directive 2009/24/EC)

- Article 5 b): To learn

- Caveats:
  - Replicating an algorithm may not be allowed, as a copy of the work infringes the copyright
  - Copy protection mechanism cannot be overcome
    - If there is a copy protection and you cannot freely execute the program, you do not have authorization to use it
  - Methods for bypassing protections are not legal
    - Crackers, keygens

- EULAs cannot restrict RE tasks

Legal Framework

Allowed situations (Europe, Directive 2009/24/EC)

- Article 6: Decompilation is generally allowed for the purposes listed in this directive, but mostly focusing on interoperability

- (allowed when) indispensable to obtain the information necessary to achieve the interoperability of an independently created computer program with other programs

- Provided that the following conditions are met:
  - those acts are performed by the licensee or by another **person having a right to use a copy of a program**, or on their behalf by a person authorized to do so
  - the information necessary to achieve interoperability **has not previously been readily available** to the persons referred to in point (a); and
  - those **acts are confined** to the parts of the original program which are necessary in order to achieve interoperability.

Legal Framework

Allowed situations (USA, DMCA)

- **Interoperability**: even circumventing DRM
- **Encryption research**: if the protection prevents the evaluation of the technology
- **Security testing**: determine if a software is secure and to improve it
- **Regulation**: to limit what information is presented to minors
- **Government Investigation**: government agencies are not affected
- **Privacy protection**: users may reverse and circumvent data gathering technologies
- EULAs may restrict RE actions, although this is not guaranteed by law

Eldad Eilam, 2005

What RE Recovers?

- **System structure**: its components and their interrelationships, as expressed by their interfaces

- **Functionality**: what operations are performed on what components

- **Dynamic behavior**: system understanding about how input is transformed to output

- **Rationale**: design involves decision making between a number of alternatives at each design step

- **Construction**: modules, documentation, test suites, etc.

Chih-Wei Lu et al, “Reverse Engineering”, Handbook of Software Engineering and Knowledge Engineering, Vol.2

Software Reversing Levels

System Level Reversing

- Observe how the software is provided and how it operates
  - Involves analyzing the environment, packaging, dependencies, and then observed behavior
  - May require tools to intercept traffic, system calls, input/output

- End goal: collect information to direct further analysis
  - Important in order to select tools, processes, and overall strategy
    - Language use, packaging algorithms, encryption
  - Important to characterize behavior and identify external dependencies
    - Remote servers involved, files accessed, communication channels used

Software Reversing Process

Code Level Reversing

- Extract design concepts and algorithms from binaries
  - Compiled to binary code or bytecode.

- It’s a complex, architecture dependent process
  - Some say “an art form”
  - Expensive enough that competitive RE is not usually pursued
    - To fully reverse and reassemble a given competing software (except in some cases)

- Makes use of tools capable of representing the low-level language in something “human compatible”
  - Compiler optimization and obfuscation make this process uncertain
  - Perfect reconstruction is frequently impossible as low-level languages do not use the same constructs as higher-level ones

Software Reversing Activities

- Understanding the processes
  - Large scale observation of the program at a process level
  - Identification of major components and their functionality
- Understanding the Data
  - Understand data structures used
- Understanding Interfaces
  - Which interfaces exist and how the process reacts to them

Software Reversing

- Programs are developed in a high-level programming languages
  - C, C++, C#, Java, Python, Go…
- A compiler converts the high-level instructions to low level instructions
  - Machine Code: instructions that are executed directly by the CPU
  - Bytecode: instructions that are executed by a middleware, VM or Interpreter
- Reverse Engineering involves understanding low level instructions
  - Which is not easy and is costly
  - Requires knowledge of the specific target being analyzed (the VM, the CPU)
    - Different CPUs have different opcodes and execution behavior

Low level languages

Machine Code

- Each CPU has a specific instruction set
  - Associated to rules regarding structure, execution flow,
- When a program is compiled to “binary”, the high-level logic is converted to a sequence of instructions
  - This sequence may be executed by a family of CPUs or a single model
  - Running this sequence on another CPU may involve binary translation (conversion)
- Humans are typically not capable of reading binary instructions, but instructions are always able to be translated to Assembly
  - Good: We can read binary code
  - Bad: each CPU has a specific variant of Assembly. Also, assembly is not simple.

Low level language

Machine Code

[Compiler Explorer (godbolt.org)](https://godbolt.org)

```c
// Original C
int square(int num) {
    return num * num;
}
```

```
//ARM64 GCC 5.4
square(int):
        sub     sp, sp, #16
        str     w0, [sp, 12]
        ldr     w1, [sp, 12]
        ldr     w0, [sp, 12]
        mul     w0, w1, w0
        add     sp, sp, 16
        ret
```

```
//MIPS64 GCC 5.4
square(int):
        daddiu  $sp,$sp,-32
        sd      $fp,24($sp)
        move    $fp,$sp
        move    $2,$4
        sll     $2,$2,0
        sw      $2,0($fp)
        lw      $3,0($fp)
        lw      $2,0($fp)
        mult    $3,$2
        mflo    $2
        move    $sp,$fp
        ld      $fp,24($sp)
        daddiu  $sp,$sp,32
        j       $31
        nop
```

```
//PowerPC GCC 4.8.5
square(int):
        stwu     1,-32(1)
        stw     31,28(1)
        mr      31,1
        stw     3,8(31)
        lwz     10,8(31)
        lwz     9,8(31)
        mullw   9,10,9
        mr      3,9
        addi    11,31,32
        lwz     31,-4(11)
        mr      1,11
        blr
```

```
//x86_64 gcc 5.4
square(int):
        push    rbp
        mov     rbp, rsp
        mov     DWORD PTR [rbp-4], edi
        mov     eax, DWORD PTR [rbp-4]
        imul    eax, DWORD PTR [rbp-4]
        pop     rbp
        ret
```

Low level languages

Machine Code

- For compiled programs, the RE tasks involves extracting information from the sequence of Assembly instructions
  - Disassembly is automatic, the rest frequently it isn't

- Reconstruction is never perfect!
  - Different level of abstraction: e.g., it is not trivial to recover C++ class structure and OOP relations from Assembly code
  - **Different compilers generate different assembly** for the same source code
  - **Same compiler may generate different assembly** for the same source code
    - Optimization flags, CPU matching, protection mechanisms, target object type…

Low level languages

Bytecode

- Some languages are compiled to a bytecode (!= machine code)
  - Intermediate language that is processed by a VM or framework
  - .NET, Java, Python, JS, LISP, LUA, Ocaml, Tcl, FoxPro, WebAssembly
- Bytecode contains a compact (optimized) representation of the higher layer structures
  - Framework/VM will execute bytecode in the target CPU
  - Same bytecode usually can be executed in multiple CPUs, provided there is a native VM implementation
    - The Java moto: Write Once, Run Anywhere
- Bytecode allows easier extraction of information, provided there is such route
  - May recover classes, function names, and even comments (but not always)
  - Traditional decompiling tools will not process bytecode (that easily)

Files and Filetypes

REVERSE ENGINEERING

João Paulo Barraca

deti universidade de aveiro
departamento de eletrónica,
telecomunicações e informática

Files

- Files are containers that are parsed according to a schema
  - Parsing implies knowing the file content

- How to select the adequate parser?
  - Using the file extension
  - Using magic headers
  - Using rules provided by configuration
  - Previous knowledge

- What if the parser is wrong?

File extensions

- File extensions are words appended to the filename, after a dot

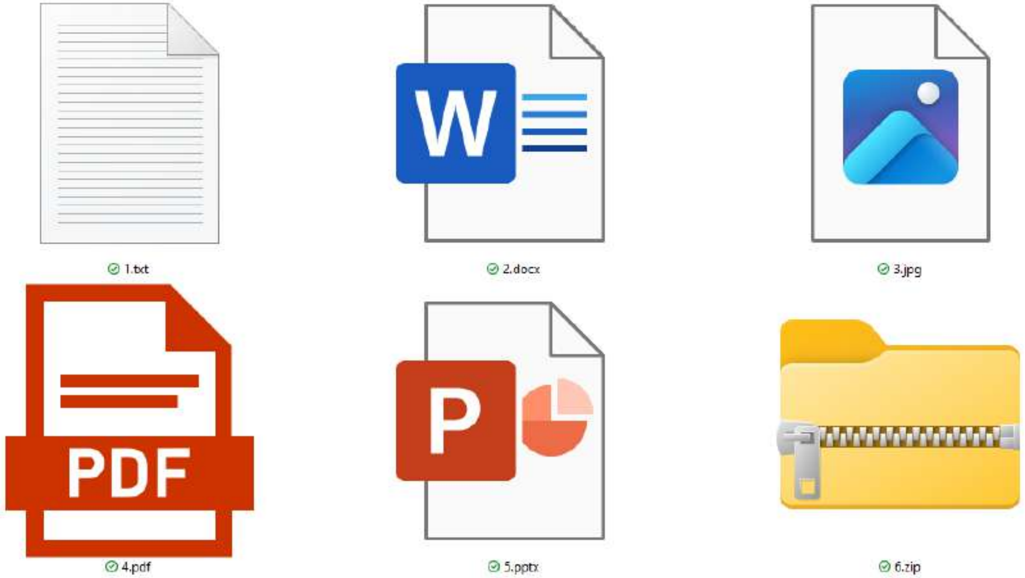
```
lecture.pptx
```

- File extensions are a basic mechanism to know how to handle a file
  - Operating systems uses extensions to select the correct process
  - Applications use it to filter which files are adequate (.e.g images). Mostly an usability aspect
  - Humans use extensions to differentiate files

- Popular file extensions:
  - zip, rar, bz2, gz, 7z: compressed files
  - exe, dll, so, com: executable files
  - jpg, tiff, bmp, fits, png: images

File extensions

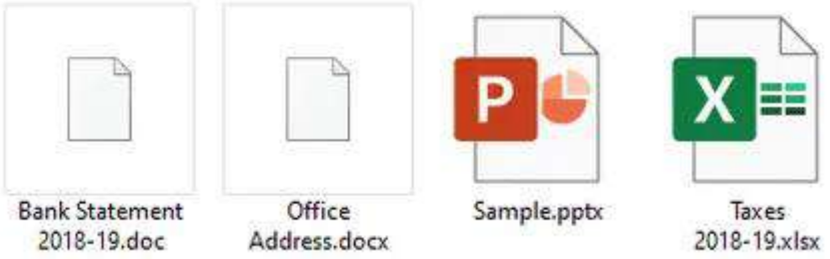
- Knowing the file extension is important to apply the correct analysis process
  - Analyzing a JPG is different from analyzing an EXE, or even a PNG



File extensions

Extensions are misleading!

- Windows hides extension of known file types
  - **`Sample.pptx`** becomes only **`Sample`**



- Executable files may have an embedded icon
  - Freely defined by the developer
  - Explorer will show that icon
- A file named **`Sample.pptx.exe`** will be shown as **`Sample.pptx`**
  - Users recognize the extension and may think the file is safe
- <u>In a RE task, consider that a file may have bogus extensions</u>

File Signature

Also known as Magic Bytes/Header

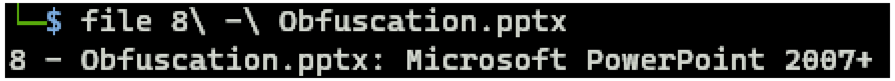
- Most files can also be recognized by a magic value in the file start/end
  - Manipulating headers can lead to incorrect detection and maybe processing
  - Some OS use the magic headers instead of the file extension
  - Also known as File Signatures

- Some magic values:
  - Office Documents: `D0 CF 11 E0`
  - ELF: `7F E L F`
  - JPG: `FF D8`
  - PNG: `89 P N G 0D 0A 1A 0A`
  - Java class: `CA FE BA BE`

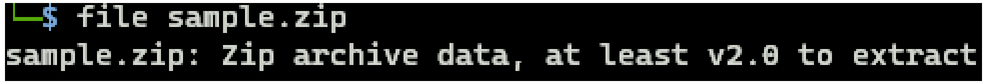
File Signature

Sometimes, magic headers are reused

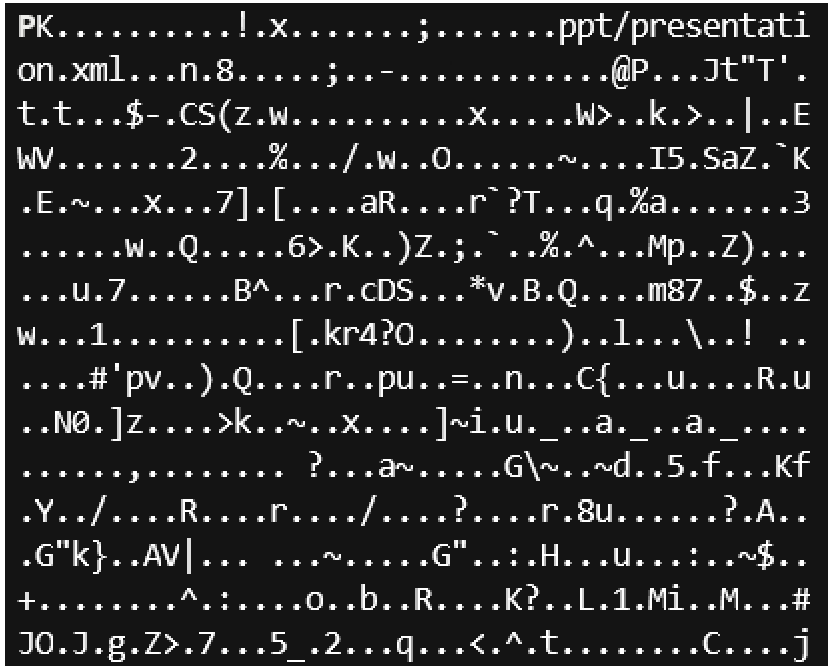
- PK.. (`50 4B 03 04`) is the magic for ZIP files



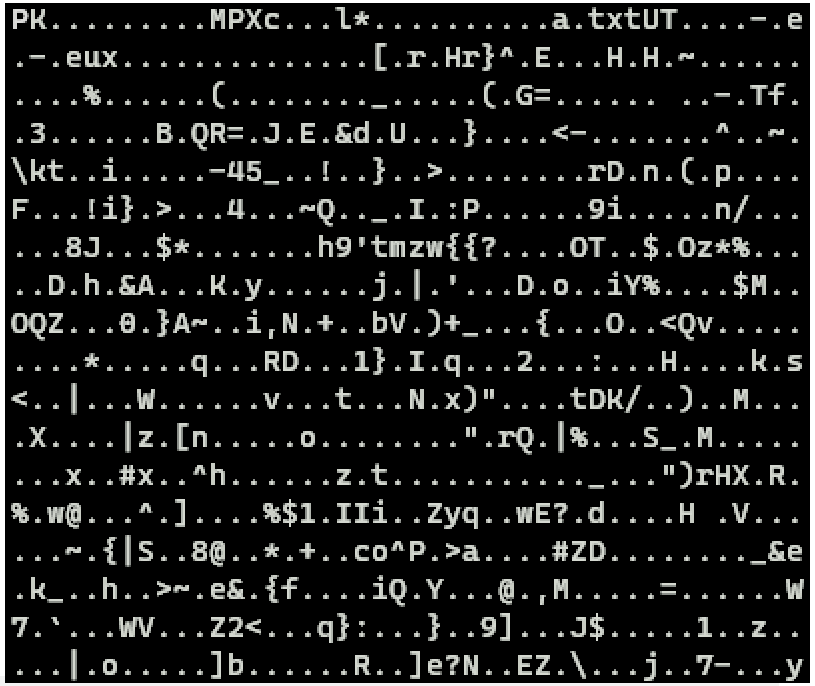
```
└─$ file 8\ -\ Obfuscation.pptx
8 - Obfuscation.pptx: Microsoft PowerPoint 2007+
```



```
└─$ file sample.zip
sample.zip: Zip archive data, at least v2.0 to extract
```



```
PK..........!.x.......;.......ppt/presentati
on.xml...n.8.....;..-............@P...Jt"T'.
t.t...$-.CS(z.w..........x.....W>..k.>..|..E
WV.......2....%.../.w..O.......~....I5.SaZ.`K
.E.~...x...7].[....aR....r`?T...q.%a.......3
......w..Q.....6>.K..)Z.;.`..%.^...Mp..Z)...
...u.7.......B^...r.cDS...*v.B.Q....m87..$..z
w...1..........[.kr4?O.........)..l...\..! ..
....#'pv..).Q....r..pu..=..n...C{...u....R.u
..N0.]z....>k..~..x....]~i.u._..a._..a._....
......,......... ?...a~.....G\~..~d..5.f...Kf
.Y../....R....r..../....?....r.8u......?.A..
.G"k}..AV|... ...~.....G"..:.H...u...:..~$..
+........^.:.....o..b..R....K?..L.1.Mi..M...#
JO.J.g.Z>.7...5_.2...q...<.^.t.........C....j
```

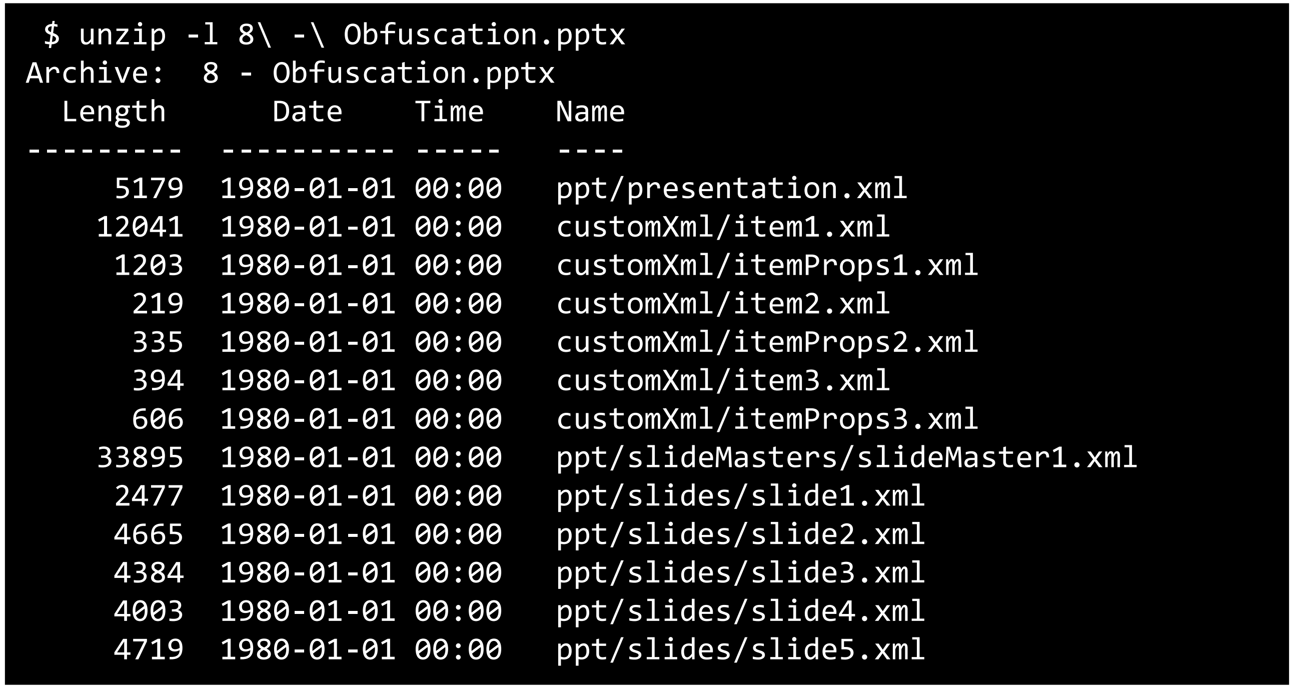


```
PK.........MPXc...l*...........a.txtUT....-.e
.-.eux..............[.r.Hr}^.E...H.H.~.......
....%.......(.........._.....(.G=...... ..-.Tf.
.3.......B.QR=.J.E.&d.U...}....<-........^..~.
\kt..i.....-45_..!..}..>.........rD.n.(.p....
F...!i}.>...4...~Q.._.I.:P......9i.....n/...
...8J...$*.......h9'tmzw{{?....OT..$.Oz*%...
..D.h.&A...K.y......j.|.'...D.o..iY%....$M..
OQZ...0.}A~..i,N.+..bV.)+_...{...O..<Qv.....
....*......q...RD...1}.I.q...2...:...H....k.s
<..|...W.......v...t...N.x)"....tDK/..)..M...
.X....|z.[n......o..........".rQ.|%...S_.M.....
...x..#x..^h......z.t............_...")rHX.R.
%.w@...^.]....%$1.IIi..Zyq..wE?.d....H .V...
...~.{|S..8@..*.+..co^P.>a....#ZD........_&e
.k_..h..>~.e&.{f....iQ.Y...@.,M.....=......W
7.`...WV...Z2<...q}:....}..9]...J$.....1..z..
...|.o......]b.......R..]e?N..EZ.\...j..7-...y
```

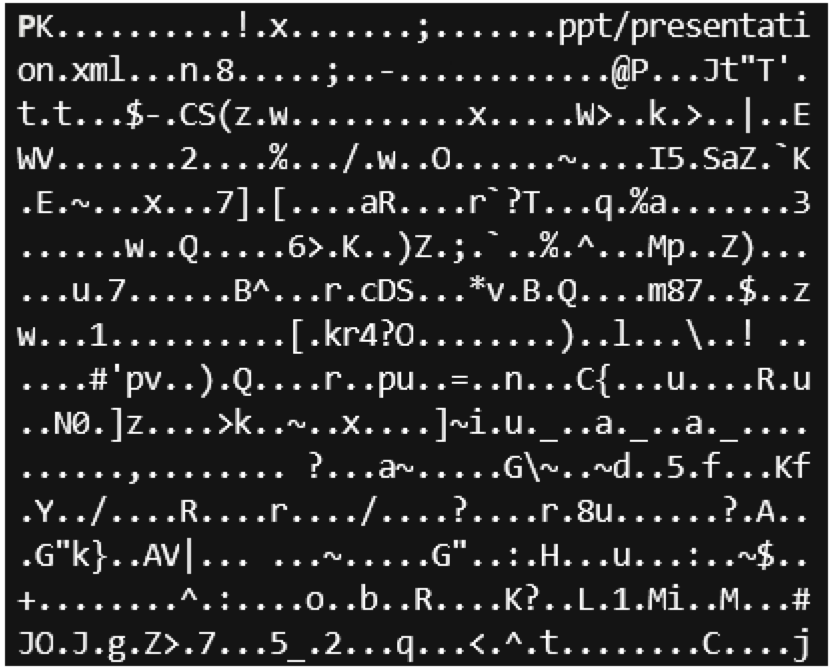
File Signature

Sometimes, magic headers are reused

- Actually, pptx are zip files



```
 $ unzip -l 8\ -\ Obfuscation.pptx
Archive:  8 - Obfuscation.pptx
  Length      Date    Time    Name
---------  ---------- -----   ----
     5179  1980-01-01 00:00   ppt/presentation.xml
    12041  1980-01-01 00:00   customXml/item1.xml
     1203  1980-01-01 00:00   customXml/itemProps1.xml
      219  1980-01-01 00:00   customXml/item2.xml
      335  1980-01-01 00:00   customXml/itemProps2.xml
      394  1980-01-01 00:00   customXml/item3.xml
      606  1980-01-01 00:00   customXml/itemProps3.xml
    33895  1980-01-01 00:00   ppt/slideMasters/slideMaster1.xml
     2477  1980-01-01 00:00   ppt/slides/slide1.xml
     4665  1980-01-01 00:00   ppt/slides/slide2.xml
     4384  1980-01-01 00:00   ppt/slides/slide3.xml
     4003  1980-01-01 00:00   ppt/slides/slide4.xml
     4719  1980-01-01 00:00   ppt/slides/slide5.xml
```



```
PK...........!.x........;.........ppt/presentati
on.xml...n.8.....;..-.............@P...Jt"T'.
t.t...$-.CS(z.w...........x.....W>..k.>..|..E
WV.......2....%.../.w..O.......~....I5.SaZ.`K
.E.~...x...7].[....aR....r`?T...q.%a.......3
.......w..Q.....6>.K..)Z.;.`..%.^...Mp..Z)...
...u.7.......B^...r.cDS...*v.B.Q....m87..$..z
w...1..........[.kr4?O.........)..l...\..! ..
....#'pv..).Q....r..pu..=..n...C{...u....R.u
..N0.]z....>k..~..x....]~i.u._..a._..a._....
.......,.........  ?...a~.....G\~..~d..5.f...Kf
.Y../....R....r..../....?....r.8u.......?.A..
.G"k}..AV|... ...~.....G"..:.H...u...:..~$..
+.........^.:.....o..b..R....K?..L.1.Mi..M...#
JO.J.g.Z>.7...5_.2...q...<.^.t.........C....j
```

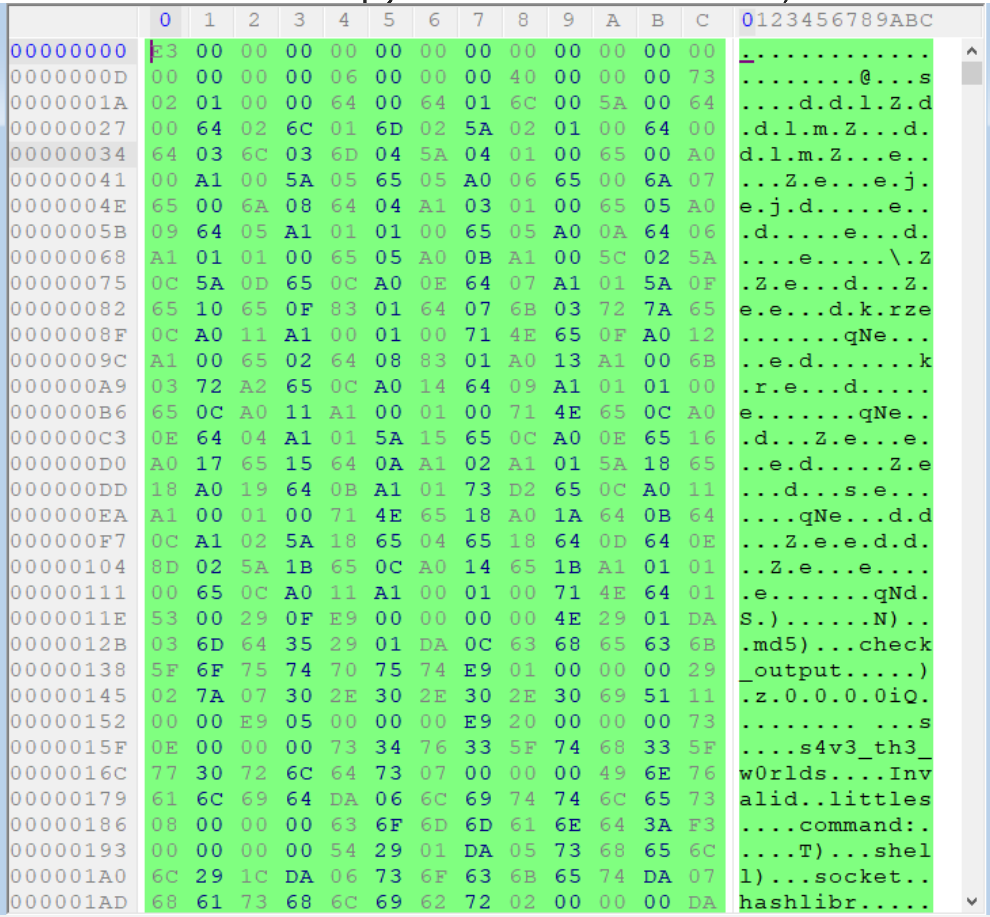
File Signature

Magic Headers can be manipulated if the content is known

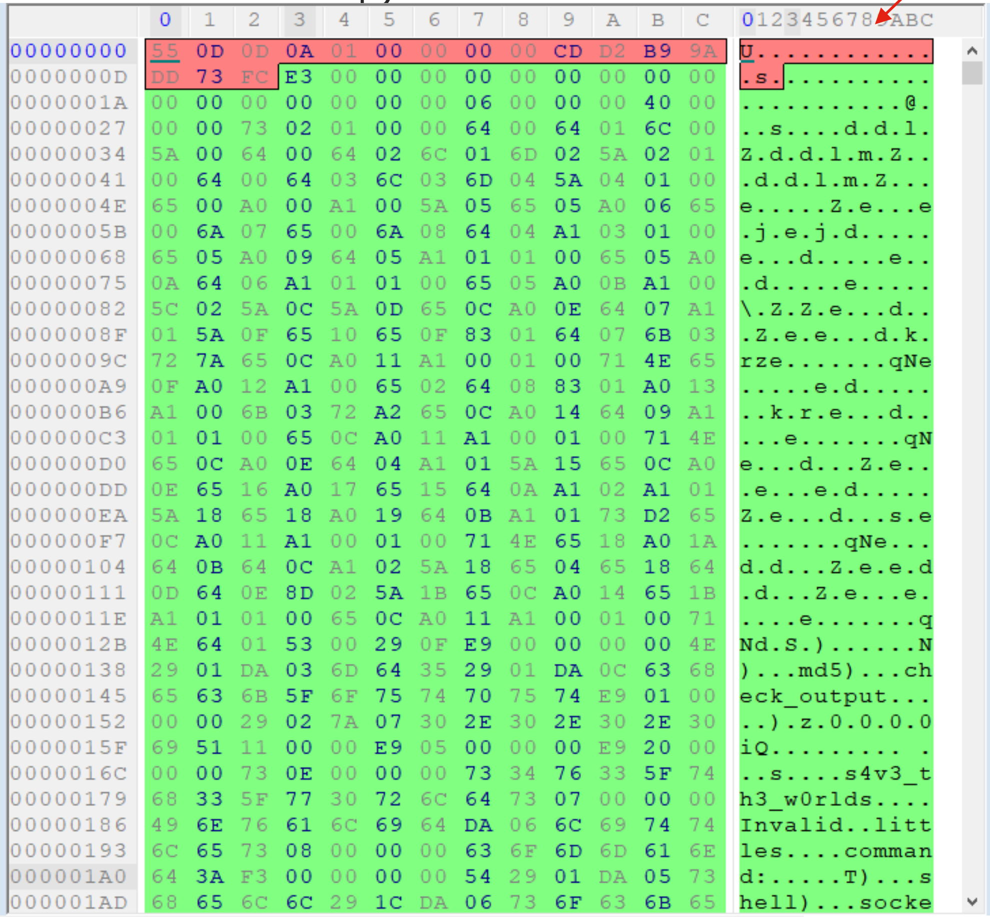
- `PyInstaller` allows conversion of a Python application to an executable application
  - It packs the pyc files into a container. Container is extracted on runtime and compiled python code is executed
  - Headers are omitted from pyc files. If header is added, extracted file executes as a standard pyc file

Added header

Extracted



```
          0  1  2  3  4  5  6  7  8  9  A  B  C  0123456789ABC
00000000  E3 00 00 00 00 00 00 00 00 00 00 00 00 .............
0000000D  00 00 00 00 06 00 00 00 40 00 00 00 73 ........@...s
0000001A  02 01 00 00 64 00 64 01 6C 00 5A 00 64 ....d.d.l.Z.d
00000027  00 64 02 6C 01 6D 02 5A 02 01 00 64 00 .d.l.m.Z...d.
00000034  64 03 6C 03 6D 04 5A 04 01 00 65 00 A0 d.l.m.Z...e..
00000041  00 A1 00 5A 05 65 05 A0 06 65 00 6A 07 ...Z.e...e.j.
0000004E  65 00 6A 08 64 04 A1 03 01 00 65 05 A0 e.j.d.....e..
0000005B  09 64 05 A1 01 01 00 65 05 A0 0A 64 06 .d.....e...d.
00000068  A1 01 01 00 65 05 A0 0B A1 00 5C 02 5A ....e.....\.Z
00000075  0C 5A 0D 65 0C A0 0E 64 07 A1 01 5A 0F .Z.e...d...Z.
00000082  65 10 65 0F 83 01 64 07 6B 03 72 7A 65 e.e...d.k.rze
0000008F  0C A0 11 A1 00 01 00 71 4E 65 0F A0 12 .......qNe...
0000009C  A1 00 65 02 64 08 83 01 A0 13 A1 00 6B ..e.d.......k
000000A9  03 72 A2 65 0C A0 14 64 09 A1 01 01 00 .r.e...d.....
000000B6  65 0C A0 11 A1 00 01 00 71 4E 65 0C A0 e.......qNe..
000000C3  0E 64 04 A1 01 5A 15 65 0C A0 0E 65 16 .d...Z.e...e.
000000D0  A0 17 65 15 64 0A A1 02 A1 01 5A 18 65 ..e.d.....Z.e
000000DD  18 A0 19 64 0B A1 01 73 D2 65 0C A0 11 ...d...s.e...
000000EA  A1 00 01 00 71 4E 65 18 A0 1A 64 0B 64 ....qNe...d.d
000000F7  0C A1 02 5A 18 65 04 65 18 64 0D 64 0E ...Z.e.e.d.d.
00000104  8D 02 5A 1B 65 0C A0 14 65 1B A1 01 01 ..Z.e...e....
00000111  00 65 0C A0 11 A1 00 01 00 71 4E 64 01 .e.......qNd.
0000011E  53 00 29 0F E9 00 00 00 00 4E 29 01 DA S.)......N)..
0000012B  03 6D 64 35 29 01 DA 0C 63 68 65 63 6B .md5)...check
00000138  5F 6F 75 74 70 75 74 E9 01 00 00 00 29 _output.....)
00000145  02 7A 07 30 2E 30 2E 30 2E 30 69 51 11 .z.0.0.0.0iQ.
00000152  00 00 E9 05 00 00 00 E9 20 00 00 00 73 ........ ...s
0000015F  0E 00 00 00 73 34 76 33 5F 74 68 33 5F ....s4v3_th3_
0000016C  77 30 72 6C 64 73 07 00 00 00 49 6E 76 w0rlds....Inv
00000179  61 6C 69 64 DA 06 6C 69 74 74 6C 65 73 alid..littles
00000186  08 00 00 00 63 6F 6D 6D 61 6E 64 3A F3 ....command:.
00000193  00 00 00 00 54 29 01 DA 05 73 68 65 6C ....T)...shel
000001A0  6C 29 1C DA 06 73 6F 63 6B 65 74 DA 07 l)...socket..
000001AD  68 61 73 68 6C 69 62 72 02 00 00 00 DA hashlibr.....
```



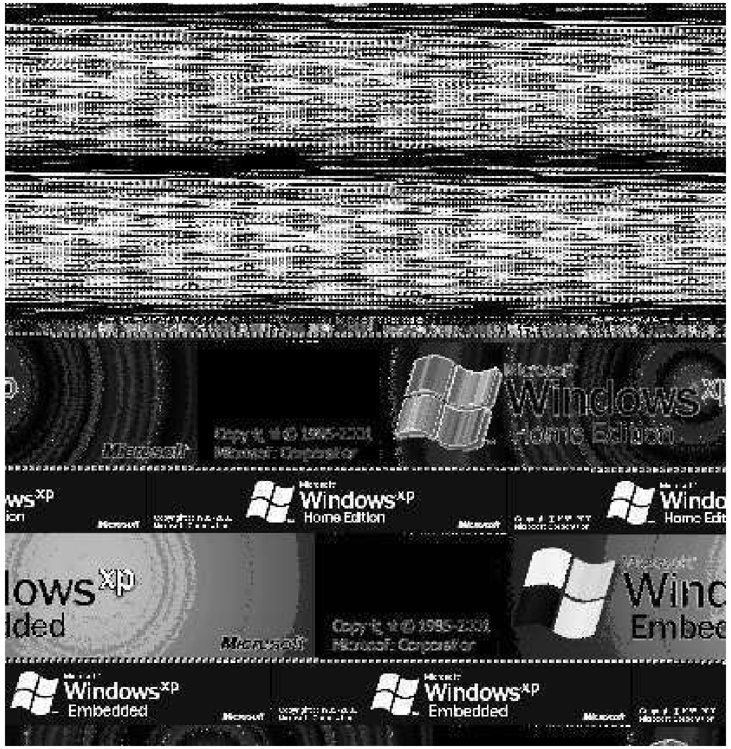
```
          0  1  2  3  4  5  6  7  8  9  A  B  C  0123456789ABC
00000000  55 0D 0D 0A 01 00 00 00 00 CD D2 B9 9A U............
0000000D  DD 73 FC E3 00 00 00 00 00 00 00 00 00 .s...........
0000001A  00 00 00 00 00 00 00 06 00 00 00 40 00 ............@.
00000027  00 00 73 02 01 00 00 64 00 64 01 6C 00 ..s....d.d.l.
00000034  5A 00 64 00 64 02 6C 01 6D 02 5A 02 01 Z.d.d.l.m.Z..
00000041  00 64 00 64 03 6C 03 6D 04 5A 04 01 00 .d.d.l.m.Z...
0000004E  65 00 A0 00 A1 00 5A 05 65 05 A0 06 65 e.....Z.e...e
0000005B  00 6A 07 65 00 6A 08 64 04 A1 03 01 00 .j.e.j.d.....
00000068  65 05 A0 09 64 05 A1 01 01 00 65 05 A0 e...d.....e..
00000075  0A 64 06 A1 01 01 00 65 05 A0 0B A1 00 .d.....e.....
00000082  5C 02 5A 0C 5A 0D 65 0C A0 0E 64 07 A1 \.Z.Z.e...d..
0000008F  01 5A 0F 65 10 65 0F 83 01 64 07 6B 03 .Z.e.e...d.k.
0000009C  72 7A 65 0C A0 11 A1 00 01 00 71 4E 65 rze.......qNe
000000A9  0F A0 12 A1 00 65 02 64 08 83 01 A0 13 .....e.d.....
000000B6  A1 00 6B 03 72 A2 65 0C A0 14 64 09 A1 ..k.r.e...d..
000000C3  01 01 00 65 0C A0 11 A1 00 01 00 71 4E ...e.......qN
000000D0  65 0C A0 0E 64 04 A1 01 5A 15 65 0C A0 e...d...Z.e..
000000DD  0E 65 16 A0 17 65 15 64 0A A1 02 A1 01 .e...e.d.....
000000EA  5A 18 65 18 A0 19 64 0B A1 01 73 D2 65 Z.e...d...s.e
000000F7  0C A0 11 A1 00 01 00 71 4E 65 18 A0 1A .......qNe...
00000104  64 0B 64 0C A1 02 5A 18 65 04 65 18 64 d.d...Z.e.e.d
00000111  0D 64 0E 8D 02 5A 1B 65 0C A0 14 65 1B .d...Z.e...e.
0000011E  A1 01 01 00 65 0C A0 11 A1 00 01 00 71 ....e.......q
0000012B  4E 64 01 53 00 29 0F E9 00 00 00 00 4E Nd.S.)......N
00000138  29 01 DA 03 6D 64 35 29 01 DA 0C 63 68 )...md5)...ch
00000145  65 63 6B 5F 6F 75 74 70 75 74 E9 01 00 eck_output...
00000152  00 00 29 02 7A 07 30 2E 30 2E 30 2E 30 ..).z.0.0.0.0
0000015F  69 51 11 00 00 E9 05 00 00 00 E9 20 00 iQ......... .
0000016C  00 00 73 0E 00 00 00 73 34 76 33 5F 74 ..s....s4v3_t
00000179  68 33 5F 77 30 72 6C 64 73 07 00 00 00 h3_w0rlds....
00000186  49 6E 76 61 6C 69 64 DA 06 6C 69 74 74 Invalid..litt
00000193  6C 65 73 08 00 00 00 63 6F 6D 6D 61 6E les....comman
000001A0  64 3A F3 00 00 00 00 54 29 01 DA 05 73 d:....T)...s
000001AD  68 65 6C 6C 29 1C DA 06 73 6F 63 6B 65 hell)...socke
```

Reconstructed

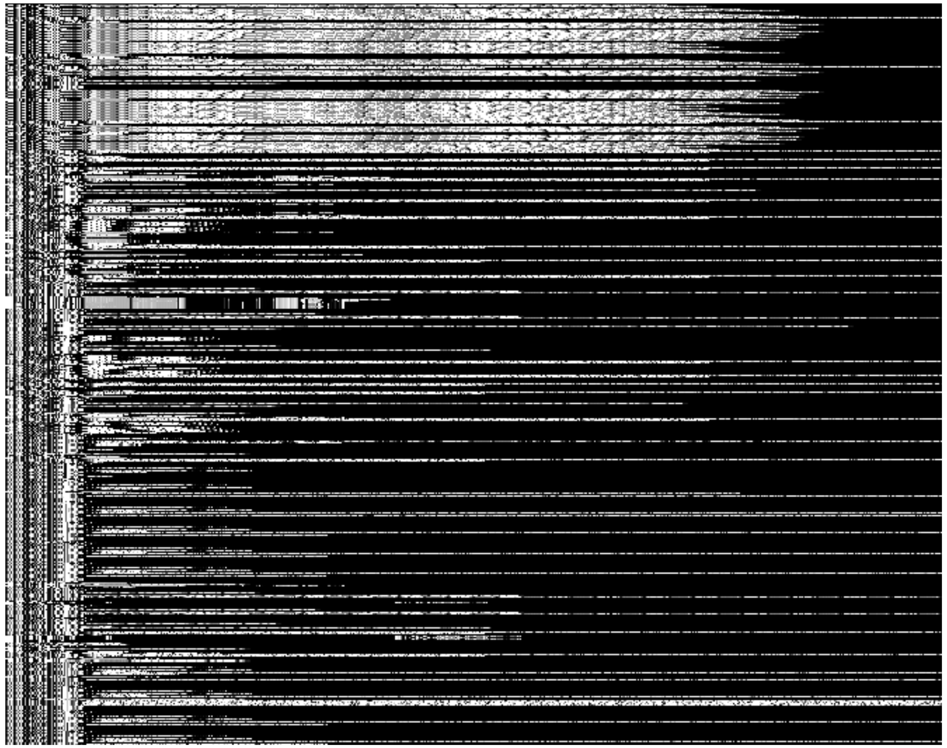
File Signature

Magic Headers can be manipulated if the content is known

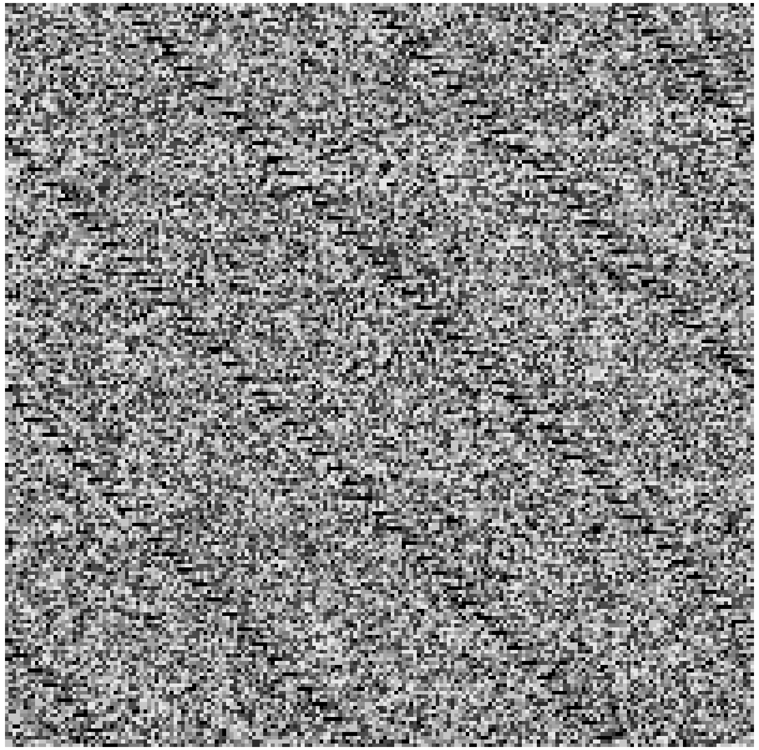
- Direct Visualization may help
  - Direct byte visualization, Mapping to an image, Entropy Analysis, Tuples



shell32.dll



Network traffic



Compressed data

Greg Conti, Sergei Bratus, “Voyage of the Reverser A Visual Study of Binary Species”

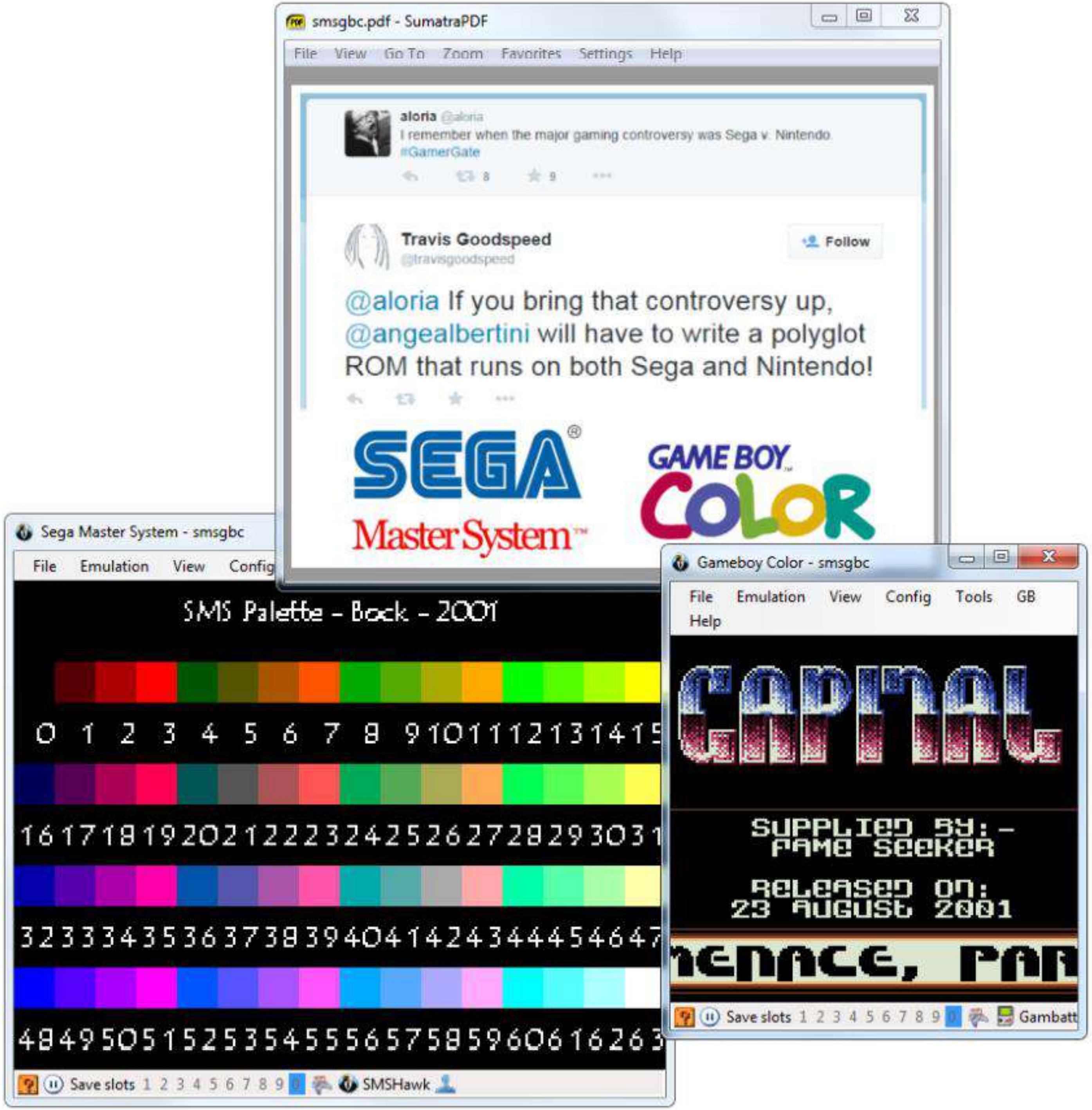
Content Type Obfuscation

Polyglots

A file that has different types simultaneously, which may bypass filters and avoid security counter-measures.

pocorgtfo19.pdf (alchemistowl.org)

Technical Note: This file, pocorgtfo19.pdf, ***is valid as a PDF*** *document,* ***a ZIP archive****, and* ***a HTML page****. It is also available as a* ***Windows PE executable****, a* ***PNG image*** *and an* ***MP4 video****, all of which have the same MD5 as this PDF*



aloria @aloria
I remember when the major gaming controversy was Sega v. Nintendo.
#GamerGate

Travis Goodspeed
@travisgoodspeed

@aloria If you bring that controversy up, @angealbertini will have to write a polyglot ROM that runs on both Sega and Nintendo!

Content Type Obfuscation - Polyglots

Types

- **Simple Polyglot file**: file **has** different types, accessed depending on how it is handled

- **Ambiguous file**: is one that **is interpreted** differently depending on the parser. One parser may crash or fail to process it, while other may return a valid file.

- **Chimera file**: file has some data that is interpreted as different types

Content Type Obfuscation - Polyglots

Use in Malware

https://nvd.nist.gov/vuln/detail/CVE-2009-1862

…allows remote attackers to **execute arbitrary code** or cause a **denial of service** (memory corruption) via (1) a **crafted Flash application in a .pdf file** or (2) **a crafted .swf file**, related to authplay.dll, as exploited in the wild in July 2009.

Content Type Obfuscation - Polyglots

Strategies

- Stacks: Data is appended to the file
- Cavities: Uses blank (non used space) in the file
- Parasites: Uses comments or metadata fields that allow content to be written
- Zippers: mutual comments

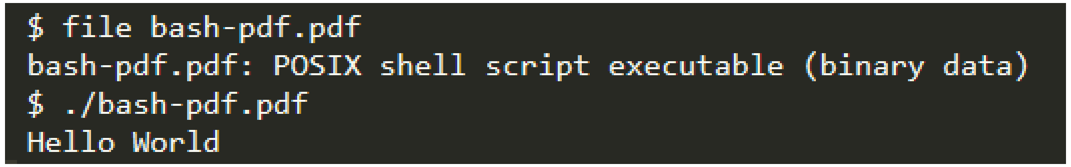
Content Type Obfuscation - Polyglots

Empty Space

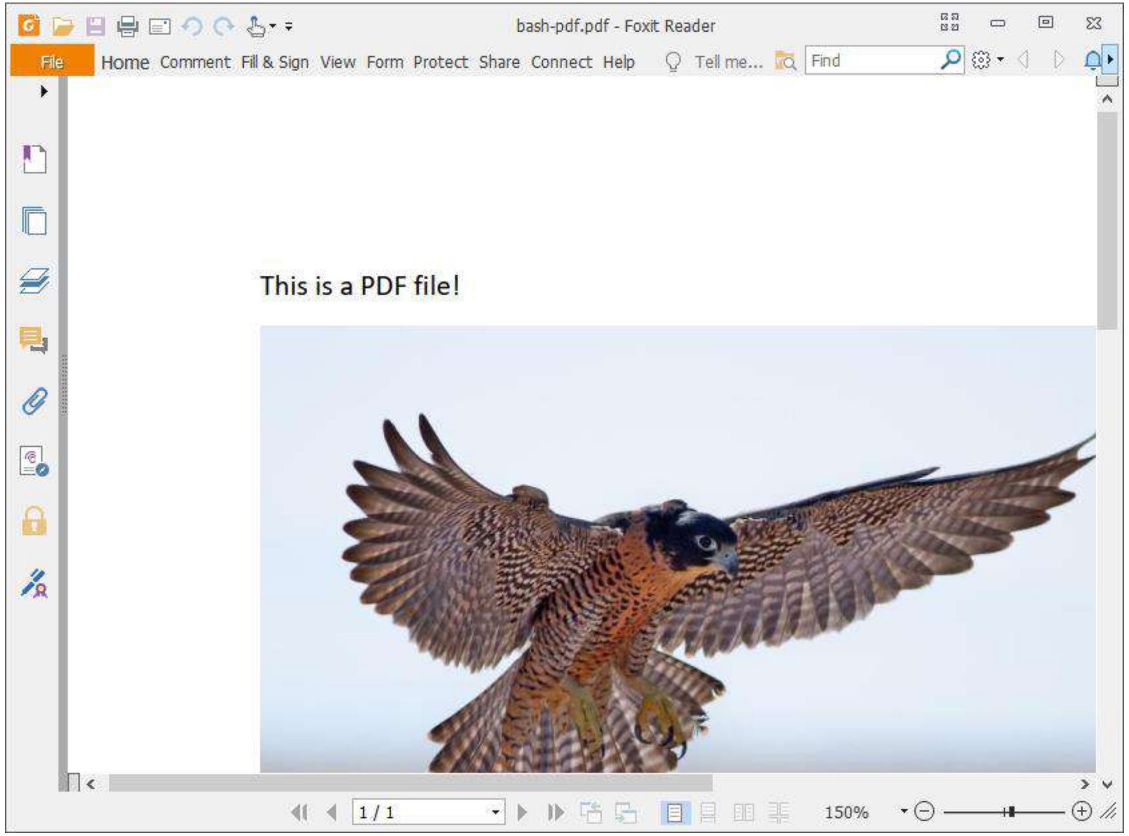
- Files sometimes allow empty or unused space
  - Before, in the middle or after actual content (appended)
  - Most common in Block formats (ISO and ROM dumps, TAR archives)
    - NAND dumps, ROM dumps, ISOs are directly mapped to sectors
  - Some formats allow arbitrary bytes before file start (e.g. PDF)
    - PDFs are processed from the end
- “Empty space” can be abused to inject crafted content

A simple bash-pdf polyglot

bash-pdf.pdf

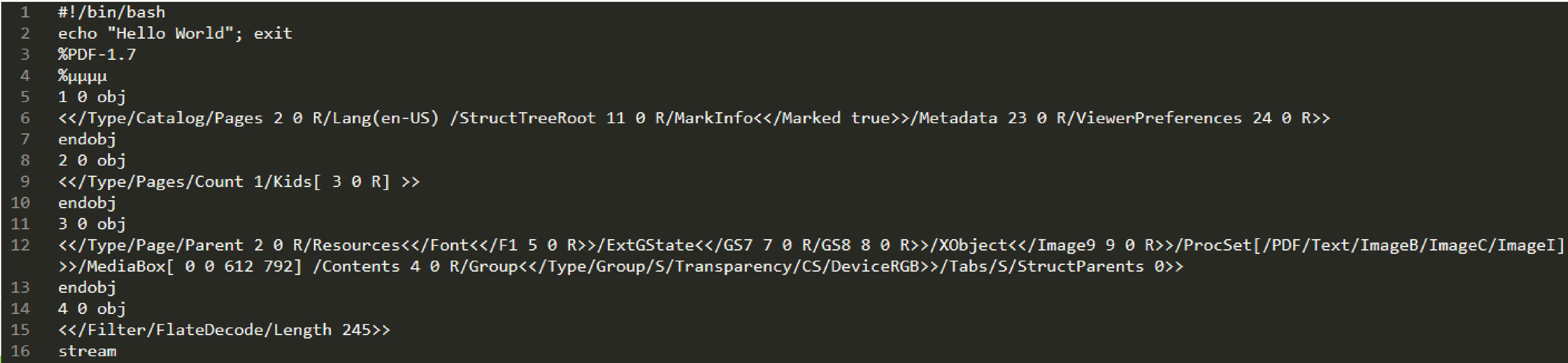


```
$ file bash-pdf.pdf
bash-pdf.pdf: POSIX shell script executable (binary data)
$ ./bash-pdf.pdf
Hello World
```



bash-pdf.pdf - Foxit Reader
File
Home Comment Fill & Sign View Form Protect Share Connect Help
Tell me...
Find
1 / 1
150%

This is a PDF file!



```
#!/bin/bash
echo "Hello World"; exit
%PDF-1.7
%µµµµ
1 0 obj
<</Type/Catalog/Pages 2 0 R/Lang(en-US) /StructTreeRoot 11 0 R/MarkInfo<</Marked true>>/Metadata 23 0 R/ViewerPreferences 24 0 R>>
endobj
2 0 obj
<</Type/Pages/Count 1/Kids[ 3 0 R] >>
endobj
3 0 obj
<</Type/Page/Parent 2 0 R/Resources<</Font<</F1 5 0 R>>/ExtGState<</GS7 7 0 R/GS8 8 0 R>>/XObject<</Image9 9 0 R>>/ProcSet[/PDF/Text/ImageB/ImageC/ImageI] >>/MediaBox[ 0 0 612 792] /Contents 4 0 R/Group<</Type/Group/S/Transparency/CS/DeviceRGB>>/Tabs/S/StructParents 0>>
endobj
4 0 obj
<</Filter/FlateDecode/Length 245>>
stream
```

A simple bash-pdf polyglot

Why?

- PDF is a collection of objects
  - Objects are dictionaries of properties with a named type
  - Called “CosObjects” or Carousel Object System
  - Simply added to file. New revisions will create new objects that are appended
  - A PDF can have unused object
  - Objects can contain executable code (the code is not executed by the pdf reader!)
    - Objects can contain anything!
    - Well…. There is the LAUNCH action, and Javascript is a valid object type…

A simple bash-pdf polyglot

A simple object

```
1 0 obj
<</length 100>>
stream

…100 bytes..



endstream
endobj
```

A simple bash-pdf polyglot

Two objects

```
1 0 obj
<</length 100>>
stream
…100 bytes..
endstream
Endobj
2 0 obj
<</length 100>>
stream
…100 bytes..
endstream
endobj
```

A simple bash-pdf polyglot

Two objects and something else that is not parsed

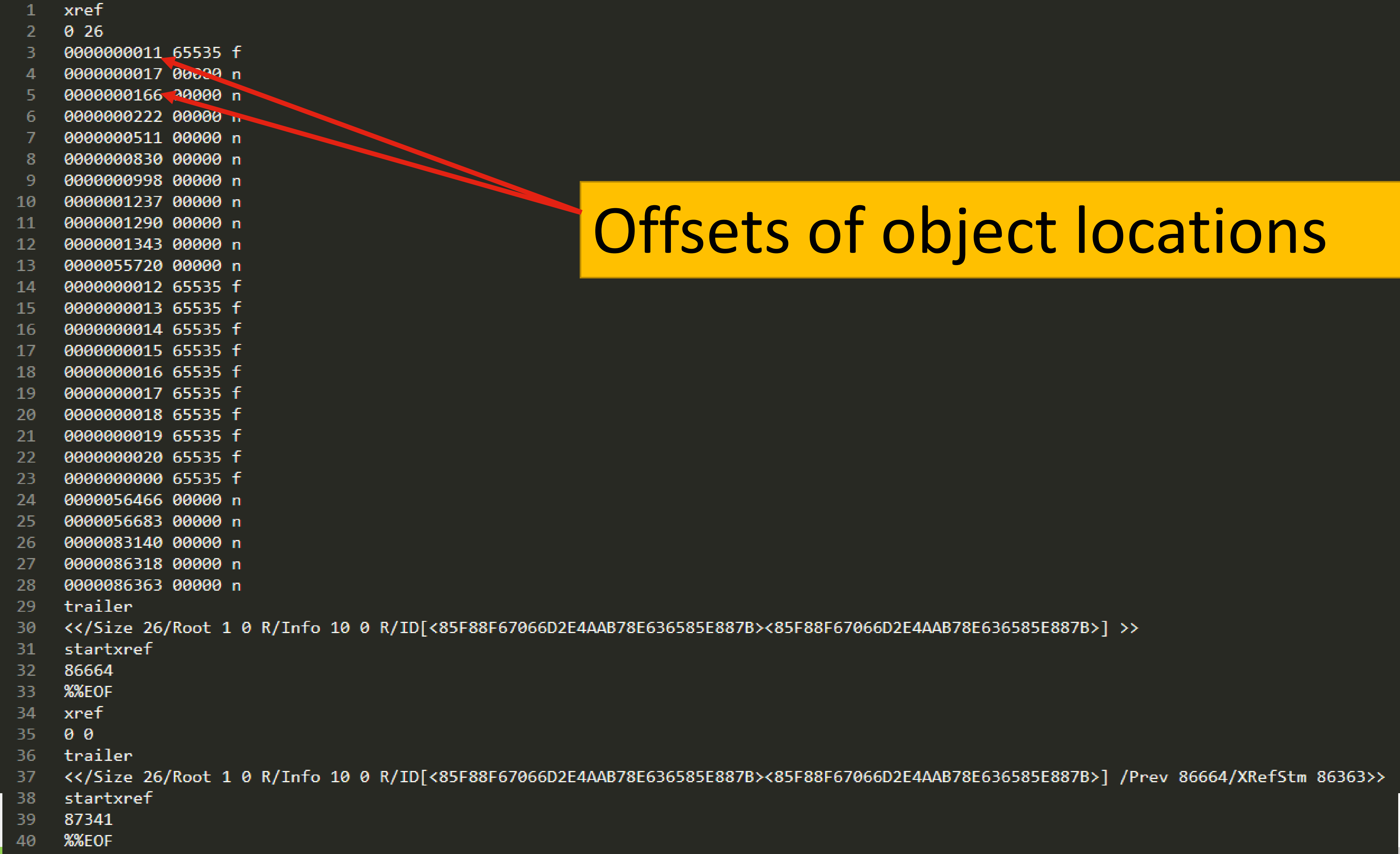
```
1 0 obj
<</length 100>>
stream
…100 bytes..
endstream
Endobj
I should not be here, but who cares. And I could be anywhere
2 0 obj
<</length 100>>
stream
…100 bytes..
endstream
endobj
```

A simple bash-pdf polyglot

The XREF Table

- PDF have a table with the offset of every object
  - In the end!
  - Reader skips to the end of the file, reads the table and parses the objects
    - That’s one reason why it ignores garbage between objects

- XREF table also defines where the file magic (`%PDF-1.5\n\n`) is
  - There may be some bytes before the magic
  - Actually, 1024 random bytes are allowed

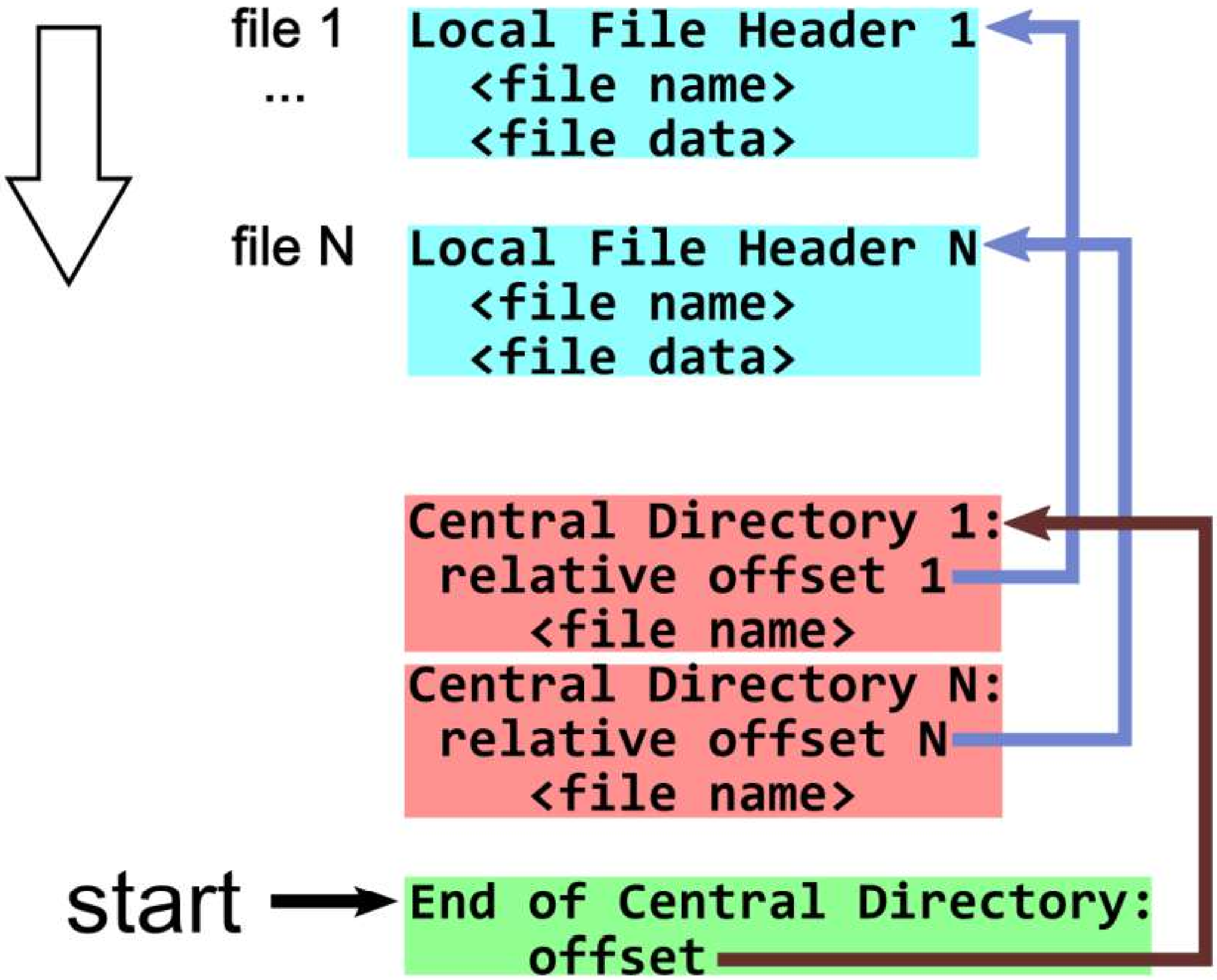


```
xref
0 26
0000000011 65535 f
0000000017 00000 n
0000000166 00000 n
0000000222 00000 n
0000000511 00000 n
0000000830 00000 n
0000000998 00000 n
0000001237 00000 n
0000001290 00000 n
0000001343 00000 n
0000055720 00000 n
0000000012 65535 f
0000000013 65535 f
0000000014 65535 f
0000000015 65535 f
0000000016 65535 f
0000000017 65535 f
0000000018 65535 f
0000000019 65535 f
0000000020 65535 f
0000000000 65535 f
0000056466 00000 n
0000056683 00000 n
0000083140 00000 n
0000086318 00000 n
0000086363 00000 n
trailer
<</Size 26/Root 1 0 R/Info 10 0 R/ID[<85F88F67066D2E4AAB78E636585E887B><85F88F67066D2E4AAB78E636585E887B>] >>
startxref
86664
%%EOF
xref
0 0
trailer
<</Size 26/Root 1 0 R/Info 10 0 R/ID[<85F88F67066D2E4AAB78E636585E887B><85F88F67066D2E4AAB78E636585E887B>] /Prev 86664/XRefStm 86363>>
startxref
87341
%%EOF
```

Offsets of object locations

Credits to Ange Albertini

ZIP



file 1

...

file N

Local File Header 1
<file name>
<file data>

Local File Header N
<file name>
<file data>

Central Directory 1:
relative offset 1
<file name>

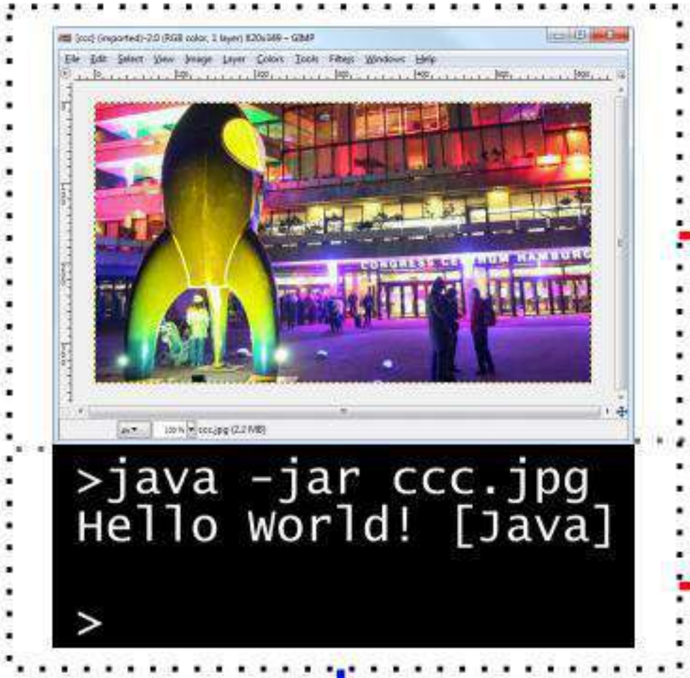
Central Directory N:
relative offset N
<file name>

start

End of Central Directory:
offset

Credits to Ange Albertini

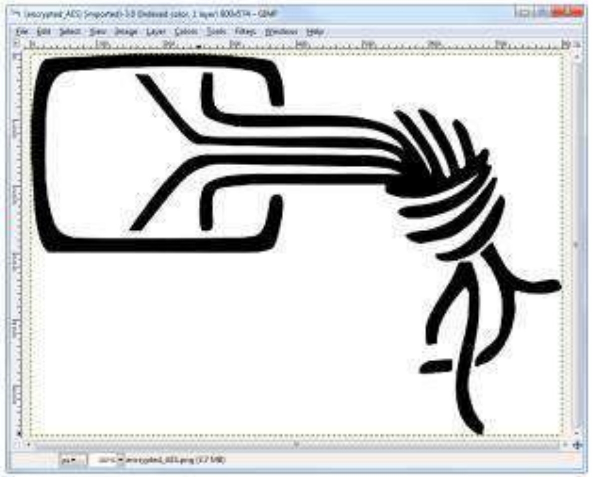
JPG



```
>java -jar ccc.jpg
Hello World! [Java]

>
```

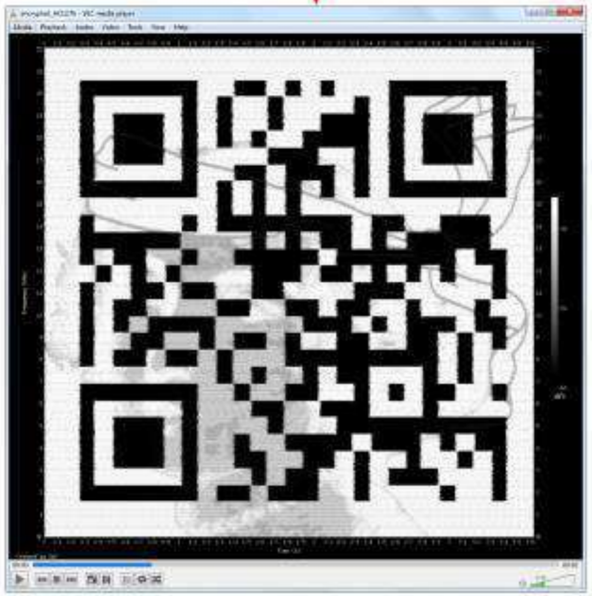
JAR
(ZIP + CLASS)

AES_{K_1}



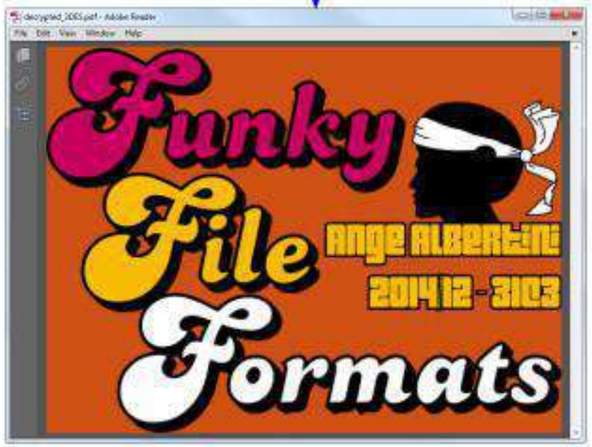
PNG

AES_{K_2}



FLV

3DES



PDF

Large Format Matrix at https://github.com/Polydet/polyglot-database

Content Type Obfuscation - Polyglots

Practical application

- Malware makes use of polyglots are means to circumvent filters
  - A Packet/Email/Web application firewall will block executables, but will it block JPGs?
    - If it does, can it be done with a low rate of false positives?

- General process involves download a polyglot and a decoder
  - Polyglot contains malicious code
  - Decode is implemented in a less suspicious manner (e.g., Javascript)

- From a Reversing Perspective: how much effort will we spend analyzing a JPG?
  - Automated tools such as `binwalk`, `TrId` and file can help (but are limited)

Android – Static Analysis 1

REVERSE ENGINEERING

deti universidade de aveiro
departamento de eletrónica,
telecomunicações e informática

João Paulo Barraca

Java Language

- Strict object-oriented programming language
  - Forces an object-oriented model where the main method is in a class
  - Forces a one-class-per-file approach
    - The name of the file must match the public class in the source code

- Can be used in a wide range of scenarios
  - Mobile: Android applications <-- focus of this class
  - Desktop: CLI or Desktop applications
  - Server: Web apps using application servers
  - Web: Java Applets, and Java Web Start, sometimes via Java Network Launch Protocol.
    - Mostly dead as browsers dropped support due to security concerns

Java Language

- Promotes the moto: Write once, run anywhere
  - Enabled by using bytecode instead of machine code

- Bytecode runs on a Java Virtual Machine
  - JVM implementation interprets bytecode in a pseudo-CPU
  - JVM is implemented natively for each supported architecture
  - Host architectural aspects are not directly exposed to applications
    - Access is mediated (and limited) by the interfaces exposed by the JVM

Java Language

- Source files must have **.java** extension
  - import statement can be used to get features from other classes

- Compiled bytecode is in **.class** files
  - The class filename matches the class inside, which enables dynamic, on demand loading.
  - For nested classes, the name of the .class file also reflects this structure

Simple Example

```java
//HelloWorld.java
import java.io.*;

public class HelloWorld {
  public static void main(String args[]) {
    System.out.println("Hello World");
  }
}
```

```
$ javac HelloWorld.java

$ ls

HelloWorld.java HelloWorld.class

$ java HelloWorld

Hello World
```

Nested Example

```
//Hello.java
import java.io.*;

public class Hello {
  public class World{};
  public void print() {
    System.out.println("Hello World");
  }
}
```

```
$ javac Hello.java
$ ls
'Hello$World.class'   Hello.class   Hello.java
```

Application Entry Points

- An application can be activated by several entry points
  - Present in the AndroidManifest, **and must be considered in order to reversing the logic**

- **Launch Activity**: One activity that is selected to start when the application starts.
  - Has a front facing UI

- **Services**: A block that is executing in the background without a front facing UI.
  - May be activated based on an event or periodically

- **Receivers**: Activated when it receives an Intent.
  - Explicit or a broadcast (e.g. charger connected)

- **Information Providers**: A database that provides information to caller applications

- **Application subclass**: A class defined to run before other components (services, receivers, …)

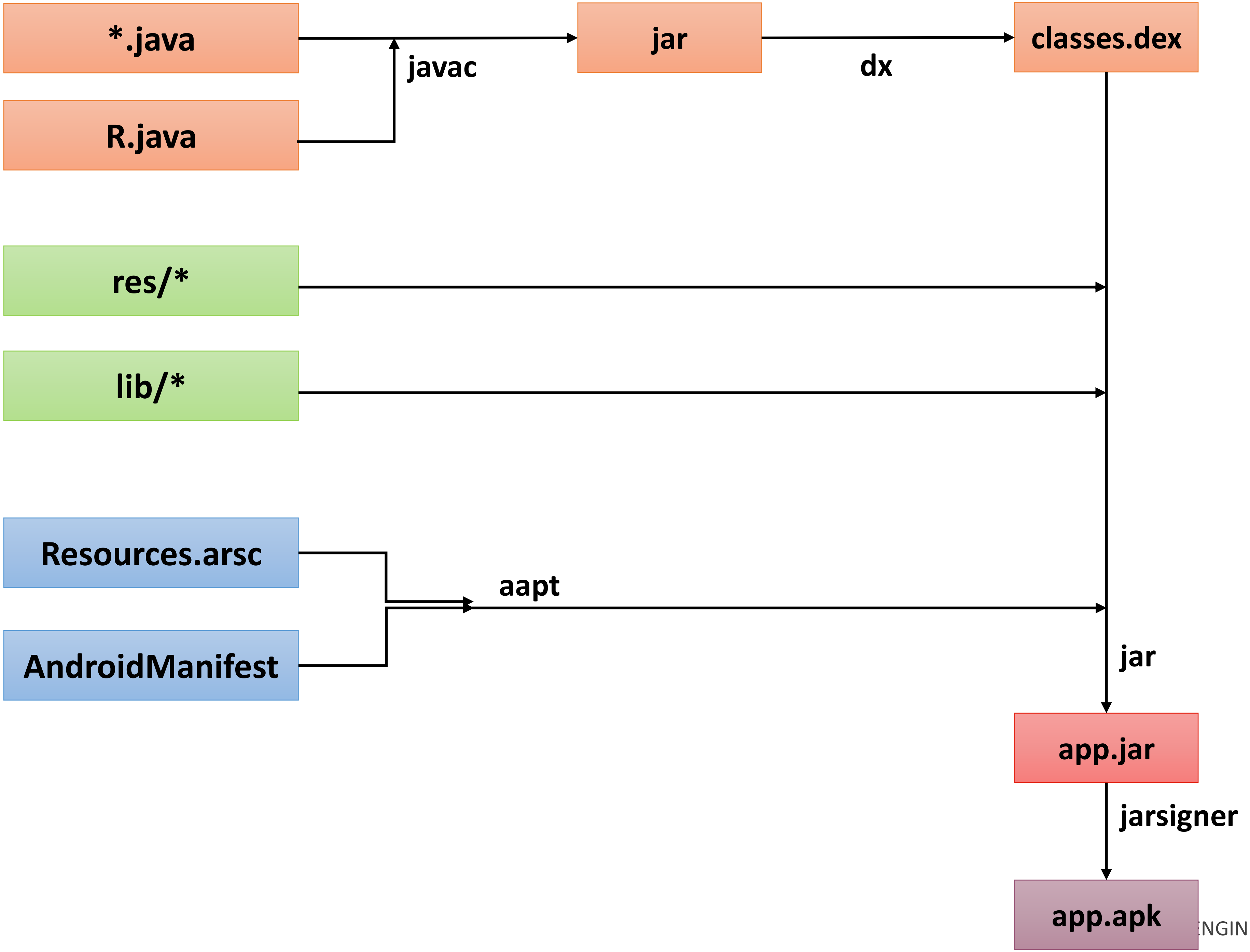
- **Exported components**: Activity, Services, Information Providers available to other applications

Application Structure

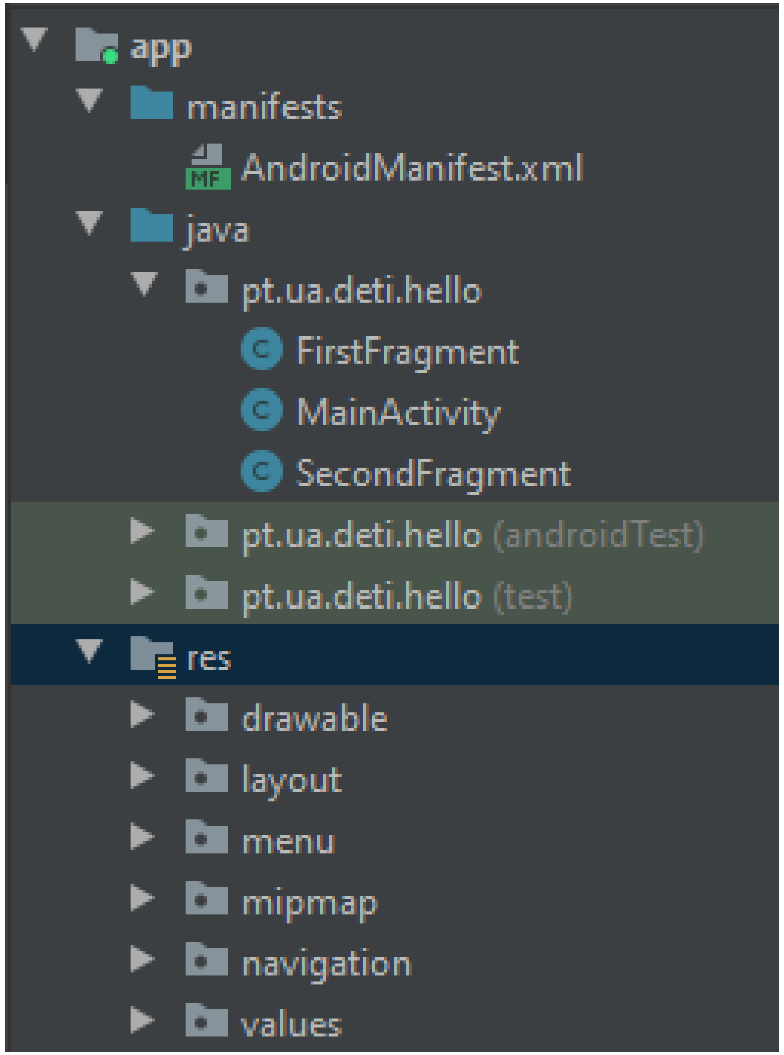
- Applications are packaged into a single file: APK
  - Actual it’s a glorified ZIP bundling different types of resources

- APK Content
  - `ETA-INF/MANIFEST.MF`: Same use as in the JAR format.
    - May have additional key/value pairs for Android-specific metadata
  - `META-INF/*`: Other files (for example *.version) that are used to add more detail
  - `classes.dex`: Compiled and bundled Android classes
    - APK may contain other dex files such as classes1.dex, classes2.dex…
  - `*.properties`: Configuration parameters for frameworks used by the app
  - `res/**`: Static resources bundled so that they could be used at run-time by the app
  - `resources.arsc`: A file of compiled resources that are bundled together
    - similar to classes.dex but for non-executable objects

APK Files

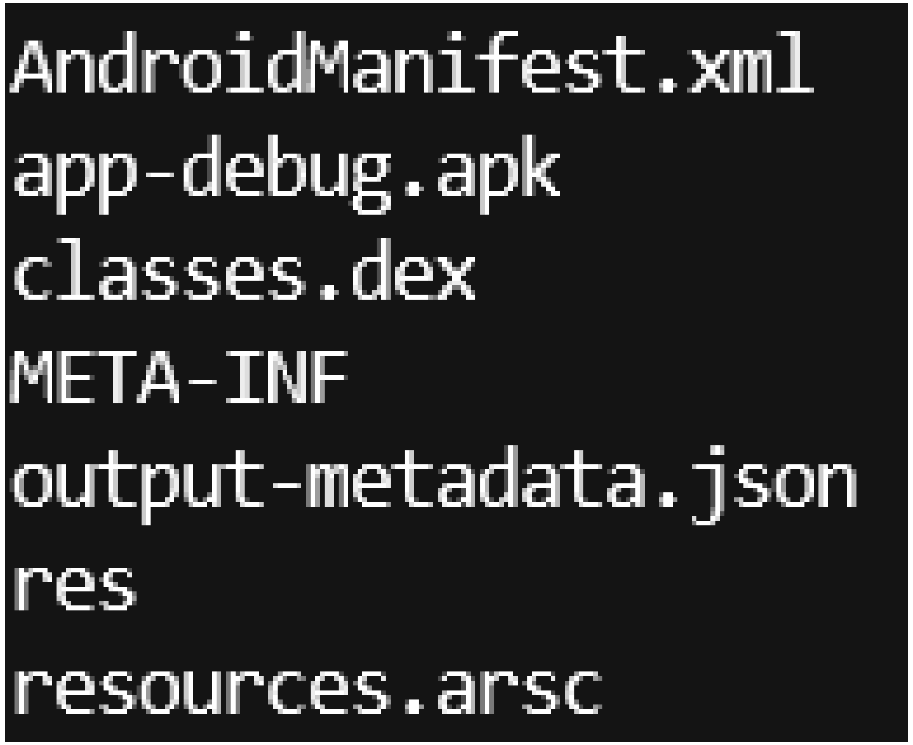

*.java
R.java
javac
jar
dx
classes.dex
res/*
lib/*
Resources.arsc
AndroidManifest
aapt
jar
app.jar
jarsigner
app.apk

APK content –Hello World app



```
app
  manifests
    AndroidManifest.xml
  java
    pt.ua.deti.hello
      FirstFragment
      MainActivity
      SecondFragment
    pt.ua.deti.hello (androidTest)
    pt.ua.deti.hello (test)
  res
    drawable
    layout
    menu
    mipmap
    navigation
    values
```

Android Studio



```
AndroidManifest.xml
app-debug.apk
classes.dex
META-INF
output-metadata.json
res
resources.arsc
```

Unzip app-debug.apk

Full for extraction: Apktool d app-debug.apk

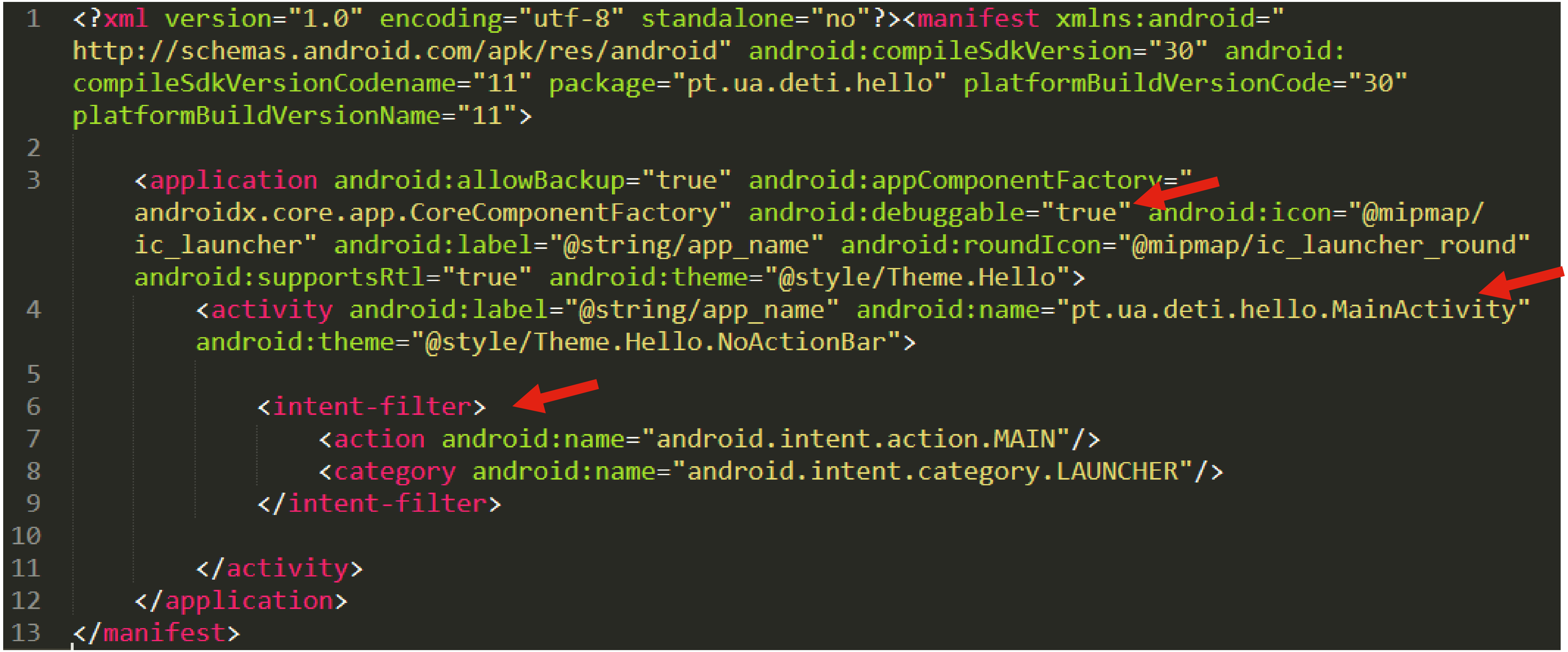
AndroidManifest.xml

- Contains essential information for app execution
  - Permissions
  - Intents exposed
  - Start classes

- Although with an XML extension it is encoded and compressed
  - Can be obtained with **`apktool`**, **`aapt`** and many others

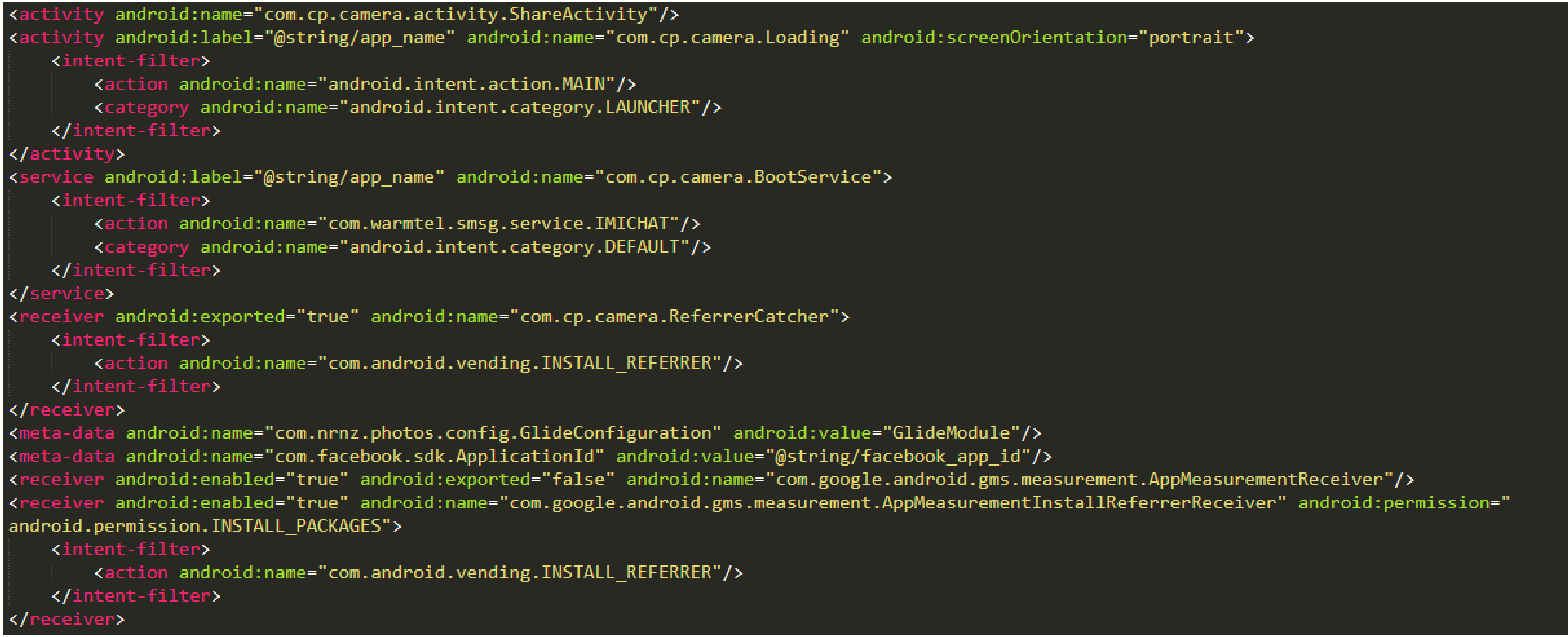
- Access to **`AndroidManifest.xml`** “is an issue” as it exposes public interfaces and data sources
  - Can be explored by simple observation/sniffing/injection and no further RE
  - But there is nothing to do about it. It’s always available

AndroidManifest.xml



```xml
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas.android.com/apk/res/android" android:compileSdkVersion="30" android:compileSdkVersionCodename="11" package="pt.ua.deti.hello" platformBuildVersionCode="30" platformBuildVersionName="11">

    <application android:allowBackup="true" android:appComponentFactory="androidx.core.app.CoreComponentFactory" android:debuggable="true" android:icon="@mipmap/ic_launcher" android:label="@string/app_name" android:roundIcon="@mipmap/ic_launcher_round" android:supportsRtl="true" android:theme="@style/Theme.Hello">
        <activity android:label="@string/app_name" android:name="pt.ua.deti.hello.MainActivity" android:theme="@style/Theme.Hello.NoActionBar">

            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>

        </activity>
    </application>
</manifest>
```

AndroidManifest.xml



```
<activity android:name="com.cp.camera.activity.ShareActivity"/>
<activity android:label="@string/app_name" android:name="com.cp.camera.Loading" android:screenOrientation="portrait">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>
</activity>
<service android:label="@string/app_name" android:name="com.cp.camera.BootService">
    <intent-filter>
        <action android:name="com.warmtel.smsg.service.IMICHAT"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
</service>
<receiver android:exported="true" android:name="com.cp.camera.ReferrerCatcher">
    <intent-filter>
        <action android:name="com.android.vending.INSTALL_REFERRER"/>
    </intent-filter>
</receiver>
<meta-data android:name="com.nrnz.photos.config.GlideConfiguration" android:value="GlideModule"/>
<meta-data android:name="com.facebook.sdk.ApplicationId" android:value="@string/facebook_app_id"/>
<receiver android:enabled="true" android:exported="false" android:name="com.google.android.gms.measurement.AppMeasurementReceiver"/>
<receiver android:enabled="true" android:name="com.google.android.gms.measurement.AppMeasurementInstallReferrerReceiver" android:permission="
android.permission.INSTALL_PACKAGES">
    <intent-filter>
        <action android:name="com.android.vending.INSTALL_REFERRER"/>
    </intent-filter>
</receiver>
```

META/MANIFEST.MF

```
$ cat META-INF/MANIFEST.MF  |head

Manifest-Version: 1.0
Built-By: Signflinger
Created-By: Android Gradle 4.1.3

Name: AndroidManifest.xml
SHA1-Digest: dSIYltCV9rAQ5lchK6i7SgU+lU8=

Name: META-INF/androidx.activity_activity.version
SHA1-Digest: BeF7ZGqBckDCBhhvlPj0xwl01dw=
```

APKs are signed and all hashes are listed, locking other files

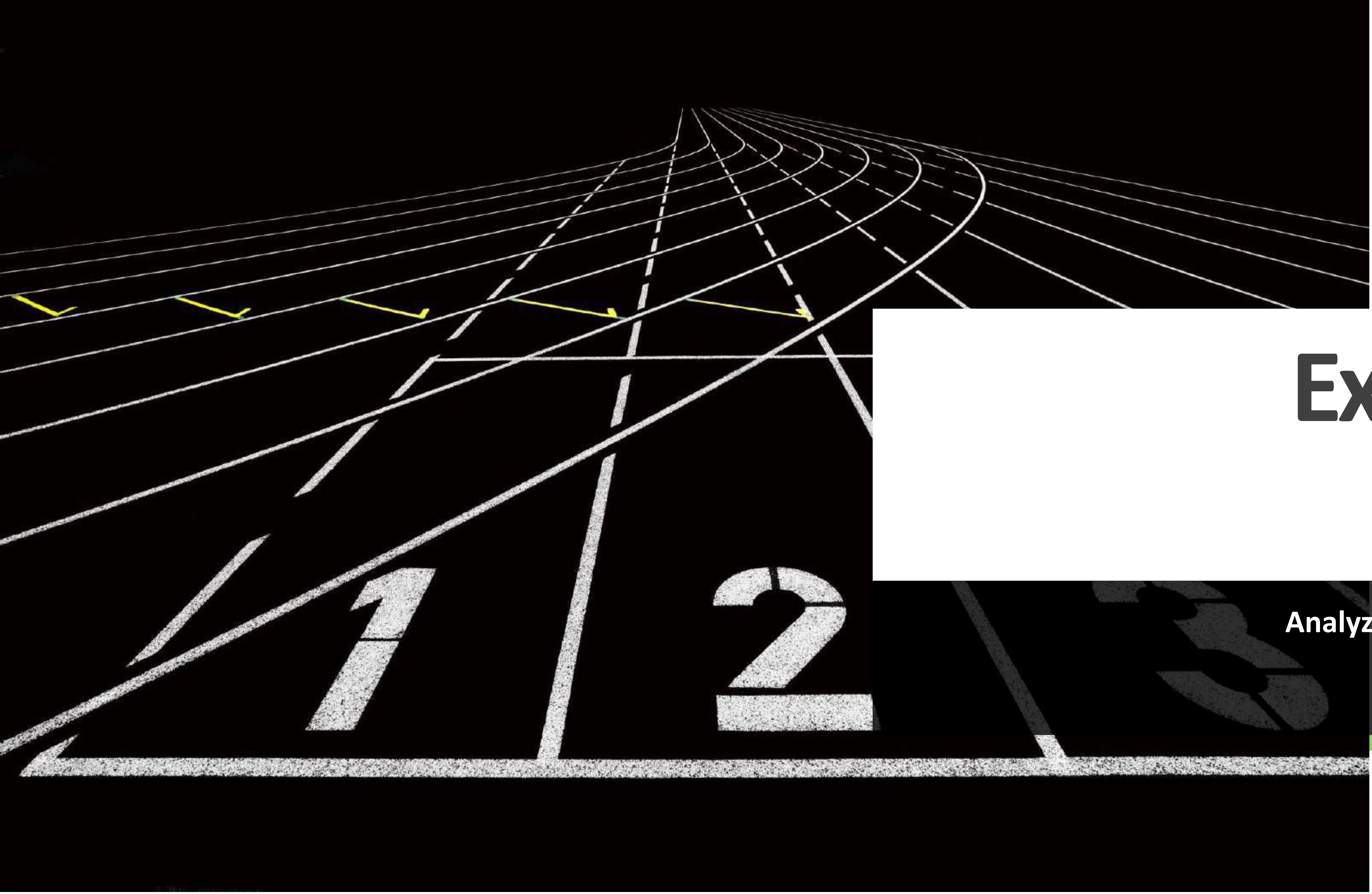
classes.dex

- Contains all Dalvik bytecode
  - Reverse engineering from APKs is always easier
    - A copy of the APK exists on the phone, but only accessible to root
  - Possible to recover most Java code

- Includes both application code and some Java libraries
  - Some android/google optional frameworks
  - Additional frameworks the developers required for development
  - May include unused frameworks
  - Doesn’t include base framework classes

- Reversing **DEX** may follow two approaches
  - Convert to smali, more difficult to understand, but always possible
  - Convert to java sources, easier to understand but not exact

```
./androidx/**
./com/**
./pt
./pt/ua
./pt/ua/deti
./pt/ua/deti/hello
./pt/ua/deti/hello/BuildConfig.smali
./pt/ua/deti/hello/FirstFragment$1.smali
./pt/ua/deti/hello/FirstFragment.smali
./pt/ua/deti/hello/MainActivity$1.smali
./pt/ua/deti/hello/MainActivity.smali
./pt/ua/deti/hello/R$anim.smali
./pt/ua/deti/hello/R$animator.smali
./pt/ua/deti/hello/R$attr.smali
./pt/ua/deti/hello/R$bool.smali
./pt/ua/deti/hello/R$color.smali
./pt/ua/deti/hello/R$dimen.smali
./pt/ua/deti/hello/R$drawable.smali
./pt/ua/deti/hello/R$id.smali
./pt/ua/deti/hello/R$integer.smali
./pt/ua/deti/hello/R$interpolator.smali
./pt/ua/deti/hello/R$layout.smali
./pt/ua/deti/hello/R$menu.smali
./pt/ua/deti/hello/R$mipmap.smali
./pt/ua/deti/hello/R$navigation.smali
./pt/ua/deti/hello/R$plurals.smali
./pt/ua/deti/hello/R$string.smali
./pt/ua/deti/hello/R$style.smali
./pt/ua/deti/hello/R$styleable.smali
./pt/ua/deti/hello/R$xml.smali
./pt/ua/deti/hello/R.smali
./pt/ua/deti/hello/SecondFragment$1.smali
./pt/ua/deti/hello/SecondFragment.smali
```



Exercise 1

Analyze the Hello application

The Java Virtual Machine

- The Java bytecode is built for a **Stack Based Machine**
  - Instructions pop values from stack, and push the result
  - Minimal number of registers (essentially only 2 for arithmetic)
  - Stack stores intermediate data

- Result:
  - very little assumptions about the target architecture (number of registers)
  - maximizes compatibility
  - very compact code
  - simple tools (compiler), simpler state maintenance

- Similar design is used in Cpython, WebAssemble, Postscript, Apache Harmony and many others

The Java Virtual Machine

Register Based

```
mov     edx, DWORD PTR [rbp-20]
mov     eax, DWORD PTR [rbp-24]
add     eax, edx
mov     DWORD PTR [rbp-4], eax
```

Stack Based

```
POP
POP
ADD
PUSH
```

```
POP  20
POP  7
ADD  20, 7
PUSH 27
```

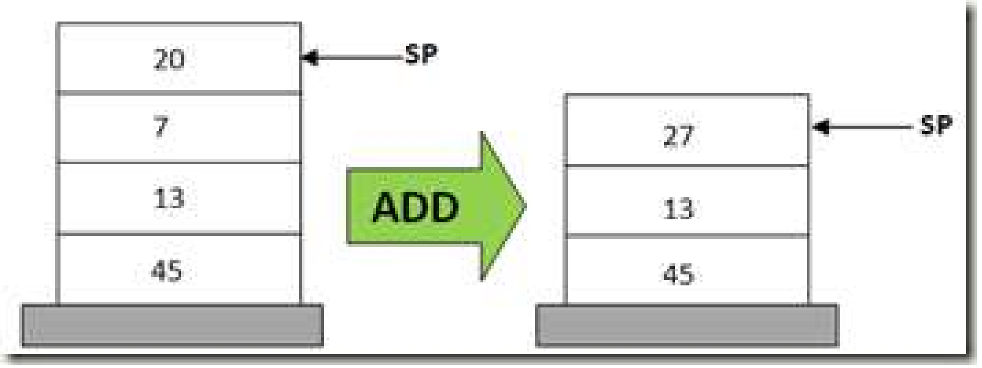


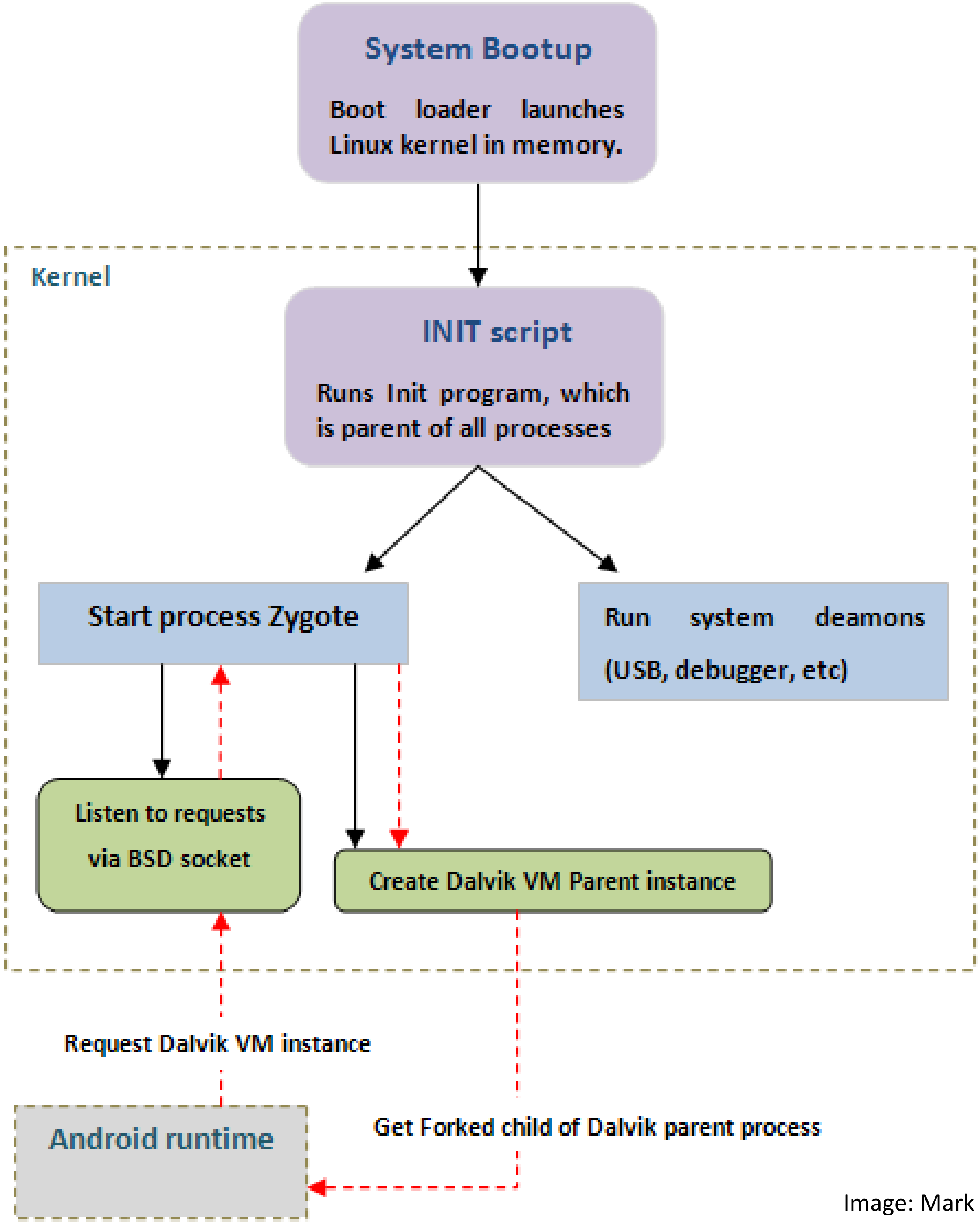
Image: Mark Vinod Sinnathamby

The Android Environment

- Android runs **Linux** with **binary programs** and **Java applications**
  - Most user space applications are Java (or HTML)
  - But can load binary objects through JNI or NDK

- The VM differs from the standard JVM, following a register-based architecture
  - Originally named Dalvik
  - Then evolved to ART after Android 4.4
  - Both environments process the Dalvik bytecode from Dalvik Executable (DEX) files

- Focus on better exploring the capability of the hardware, while having low footprint
  - Each application executed in an independent VM instance
  - Crashes and other side effects are limited to one application
  - Data isolation is ensured by the independent execution environments and forced communication through a single interface

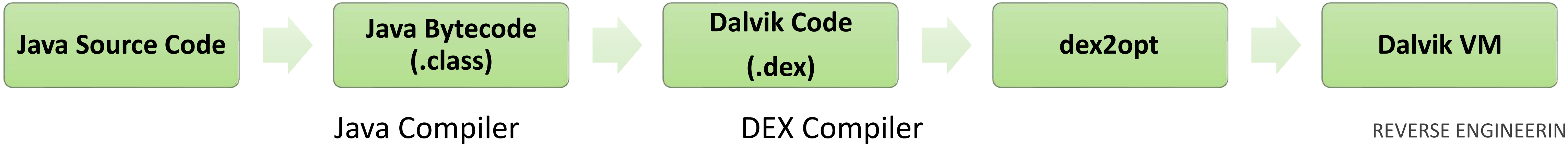


System Bootup
Boot loader launches Linux kernel in memory.
Kernel
INIT script
Runs Init program, which is parent of all processes
Start process Zygote
Run system deamons (USB, debugger, etc)
Listen to requests via BSD socket
Create Dalvik VM Parent instance
Request Dalvik VM instance
Android runtime
Get Forked child of Dalvik parent process

Image: Mark Vinod Sinnathamby

Dalvik VM

- Machine model and calling conventions imitate common architectures and C-style calling conventions
  - The machine is register-based, and frames are fixed in size upon creation.
  - Each frame consists of several registers (specified by the method) as well as any adjunct data needed to execute the method
  - Registers are considered 32 bits wide. Adjacent register pairs are used for 64-bit values
  - A function may access up to 65535 registers, usually only 16, but 256 may be common.



Java Source Code → **Java Bytecode (.class)** → **Dalvik Code (.dex)** → **dex2opt** → **Dalvik VM**

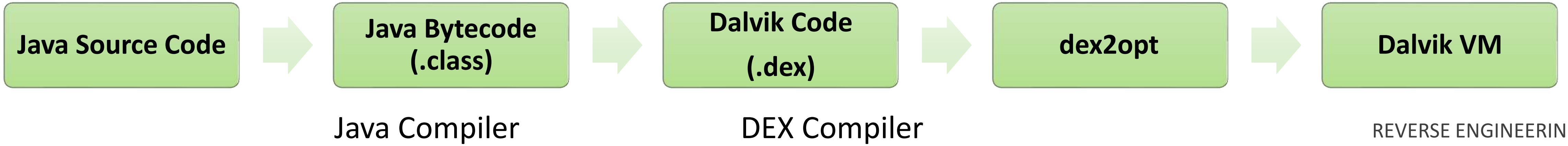
Java Compiler

DEX Compiler

Dalvik VM

- Before execution, **files are optimized** for faster execution
  - Some optimizations include resolving methods and updating the **`vtable`**
    - Methods have a signature that must be resolved to an actual **vtable** entry. Optimization changes bytecode by resolving the method location (index) in the **vtable**
  - Result is stored as an **odex** file in the `/system/cache`
    - Applications are stored “twice” as standard (APK with DEX) and optimized versions (ODEX)

- Bytecode is processed using a **Just-in-time** (JIT) approach
  - The VM will compile and translate code in Real time, during execution
  - Garbage collections tasks also execute in foreground (impact to performance)

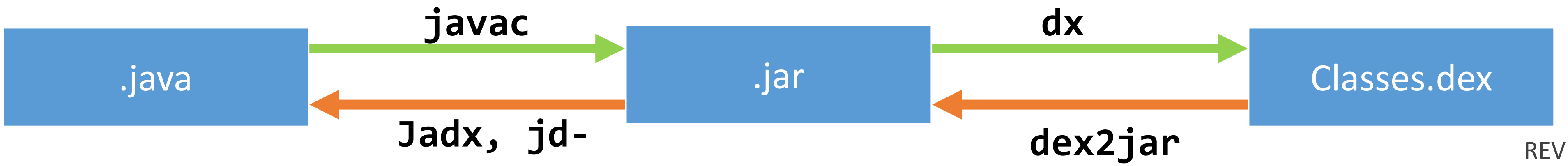


DEX files

- Dalvik EXecutable files are the standard execution format for previous Android versions
  - Created with the **dx** command:
    - In reality: `java -Xmx1024M -jar ${SDK_ROOT}.../lib/dx.jar`
  - But format is still relevant for in current systems

- Contain Java bytecode that was converted to Dalvik bytecode
  - Java uses stack + 4 registers, while Dex uses 0-v65535 registers
    - DEX registers can be mapped to ARM registers (ARM has 10 general purpose registers)
  - Optimized to constraint devices, but not so compact as instructions may be larger
    - 1-5 bytes for java, instead of 2-10 bytes

- DEX is highly like Java and bytecode can be **converted both ways**
  - dx compiles `.jar to .dex`, `dex2jar` decompiles `.dex to .jar`
  - Allows Reengineering applications (download apk, reversing, change, build, sign, publish to store)



DEX and Java Bytecode

DEX Opcode	Java Bytecode	Purpose
60-66:sget-* 52-58:iget-*	b2:getstatic b4:getfield	Read a static or instance variable
67-6d:sput 59-5f:iput	b3:putstatic b5:putfield	Write a static or instance variable
6e: invoke-virtual 6f: invoke-super 70: invoke-direct 71: invoke-static 72: invoke-interface	b6: Invokevirtual ba: invokedynamic b7: invokespecial b8: Invokestatic b9: Invokeinterface	Call a method
20: instance-of	c1: instanceof	Return true if obj is of class
1f: check-cast	c0: checkcast	Check if a type cast can be performed
bb:new	22: new-instance	New (unconstructed) instance of object

Class, Method, and Fields

DEX and Java Bytecode

DEX Opcode	Java Bytecode	Purpose
12-1c: const*	12:ldc 13: ldc_w 14: ldc2_w	Define Constant
21: array-length	be: arraylength	Get length of an array
23: new-array	bd: anewarray	Instantiate an array
24-25: filled-new-array[/range] 26: fill-array-data	N/A	Populate an array

Arithmetic Instructions

DEX and Java Bytecode

DEX Opcode	Java Bytecode	Purpose
32..37: if-* **38..3d: if-*z**	**a0-a6: if_icmp*** **99-9e: if***	**Branch on logical**
2b: packed-switch	**ab: lookupswitch**	**Switch statement,**
2c: sparse-switch	**aa: tableswitch**	**Switch statement**
28: goto **29: goto/16** **30: goto/32**	**a7: goto** **c8: goto_w**	**Jump to offset in code**
27: throw	**bf:athrow**	**Throw exception**

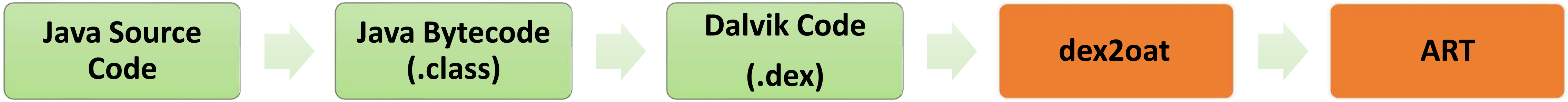
Flow Control

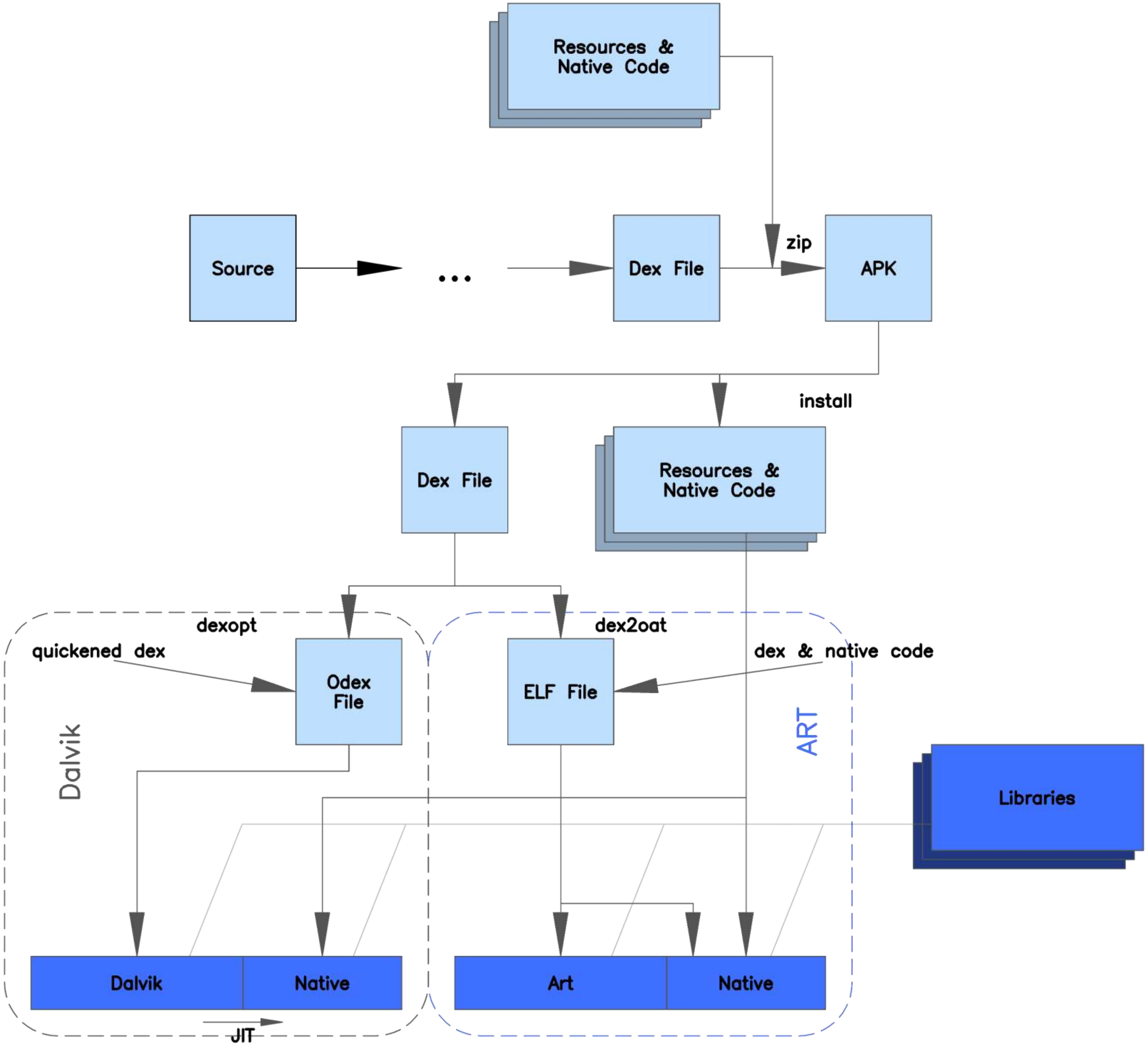
Android RunTime (ART)

- Alternative runtime which presents an **optimized execution path**
  - Introduced in Android 4.4, implemented in C++, and supports 64bits
  - Runs OAT files, which contain native code (not bytecode!)
  - References to Java objects point towards C++ objects managed by the VM
    - While application logic is expressed in Java, framework methods actually execute in native code!

- ART introduces **ahead-of-time** (AOT) compilation
  - **At install time, ART compiles apps using the on-device dex2oat tool.**
  - This utility accepts DEX files as input and generates a compiled app executable for the target device
  - Improves performance over ODEX files as file repetitive load operations are avoided

- Improves Garbage Collection by optimizing memory usage
  - Avoiding GC driven app pauses
  - Overall, it provides much better performance (more on this later)
  - JIT is not that efficient and doing it on real time hurts performance and battery




Resources &
Native Code
Source
...
Dex File
zip
APK
install
Dex File
Resources &
Native Code
dexopt
quickened dex
Odex
File
Dalvik
dex2oat
ELF File
dex & native code
ART
Libraries
Dalvik
Native
Art
Native
JIT

ART specific files

- **`.oat`** – only at `/system/framework/[arch]/boot.oat`
  - Main ART format, OAT: **O**f **A**head **T**ime (from Ahead of Time)
    - *“We went with that because then we say that process of converting .dex files to .oat files would be called quakerizing and that would be really funny.”*, reference to the Quaker Oats Company
  - It’s an ELF file containing OAT data

- **`.odex`** – an .OAT file containing the precompiled applications
  - Although it uses the same extension, .odex files with ART are .OAT files, in reality ELF files
  - Stored in `/data/dalvik-cache`
    - But Dalvik is not used with ART…

- **`.art`** – only at `/system/framework/[arch]/boot.art`
  - An .OAT file containing vital framework classes (base Java classes to be used by ART)

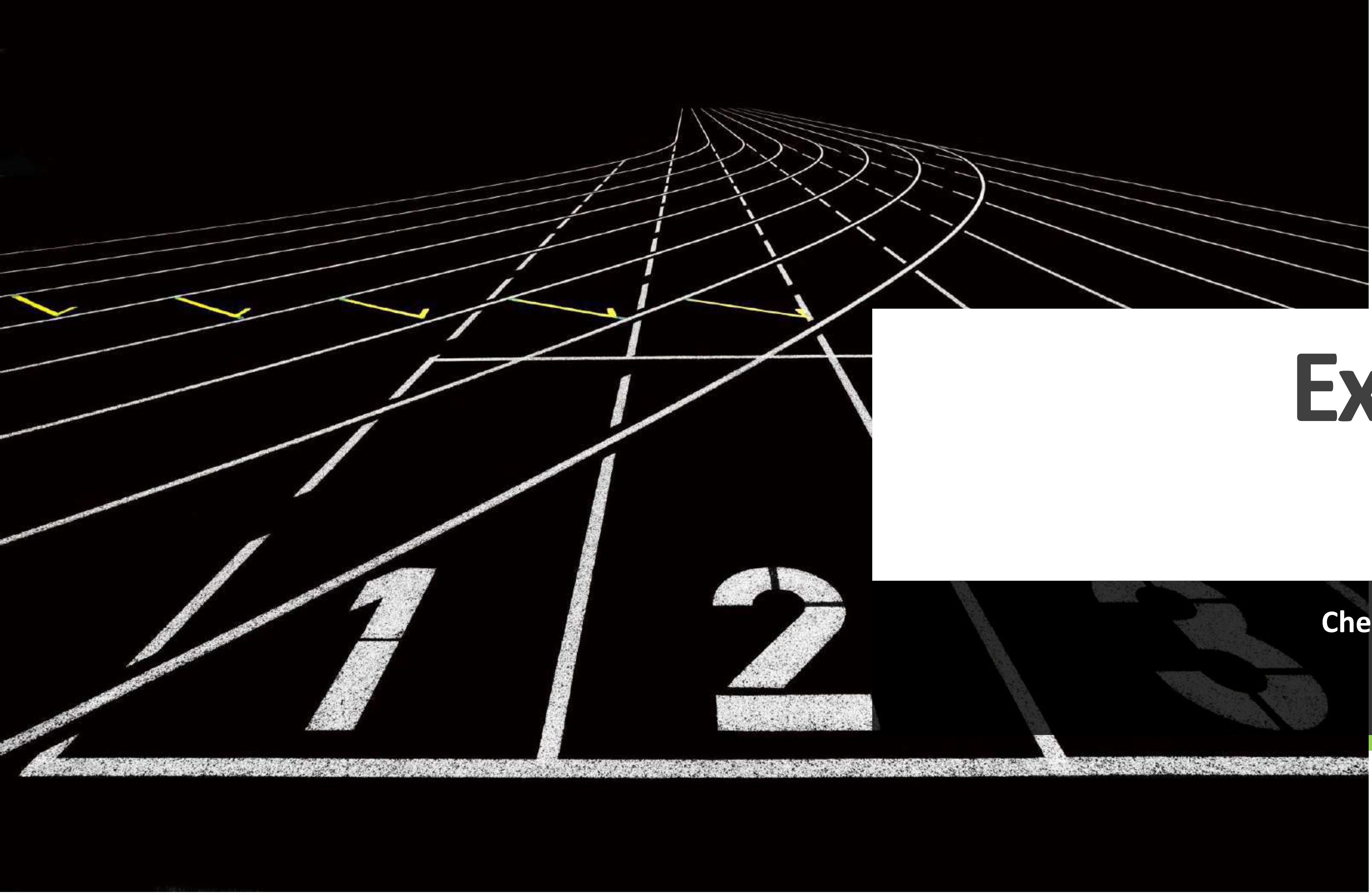
- **`.vdex`** - contains the uncompressed DEX code of the APK, with some additional metadata to speed up verification
  - Assumed to be already verified DEX files

OAT files (or DEX files in ART, which are also OAT)

- Are ELF files containing DEX code
  - OAT Header, followed by DEX files in an ELF container
    - DEX files can be extracted with `oat2dex`

- Java methods in DEX file are mirrored in C++
  - `java.lang.String: -> art::mirror::String`
  - When the Java code creates an object, the object is created in the C++ (native) code by the VM
    - JVM handles references to the C++ object

- On boot, common objects are instantiated (ones in Android Framework) by loading `boot.art`
  - To speed up execution as such classes are required by most applications



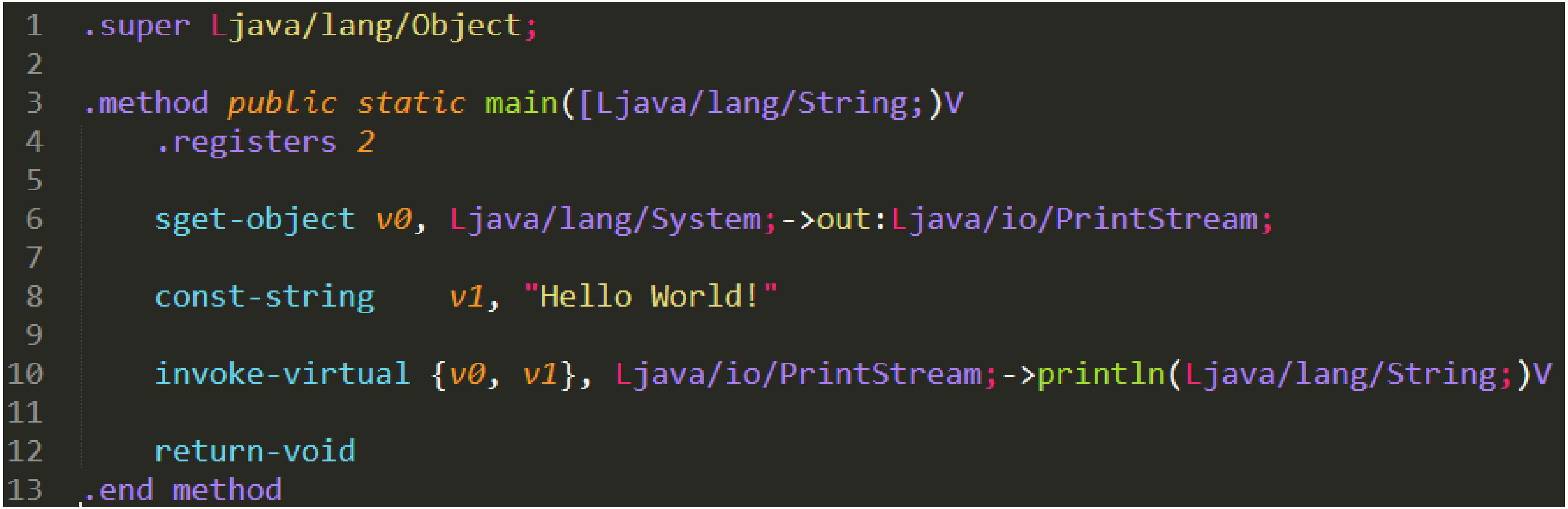
Exercise 2

Check the Class Workbook

Smali and Baksmali

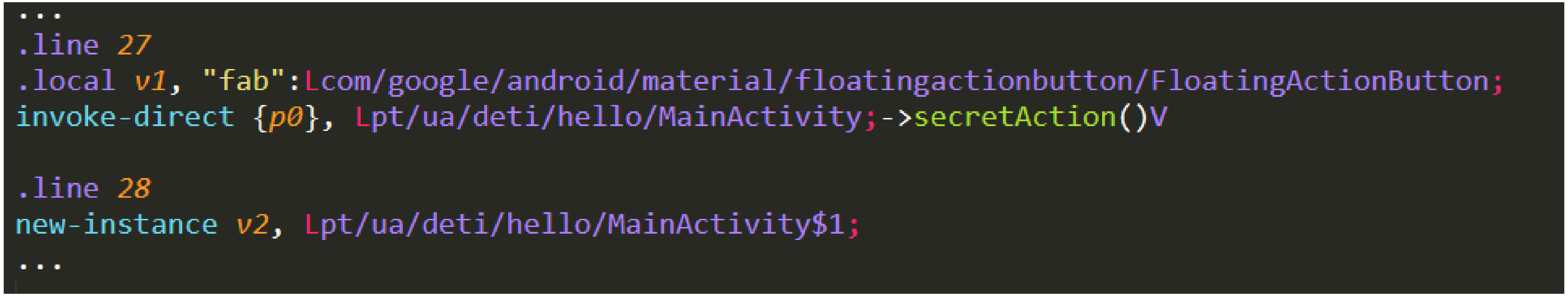
- Assembler/disassembler for the DEX format used by Dalvik
  - smali = “assembly” of the DEX bytecode
  - backsmaling = decompiling to smali
- Allows converting a DEX blob to something “more human friendly”
  - Similar to Assembly language in a common CPU
- Why? Isn’t DEX <-> class possible?
  - With recent compiler optimizations (and Kotlin, and obfuscation) not always…
  - It’s possible to compile DEX `(smali)->class->Java`, but code may not be correct
  - Use of smali enables patching DEX bytecode directly (although it’s more complex)

HelloWorld.smali

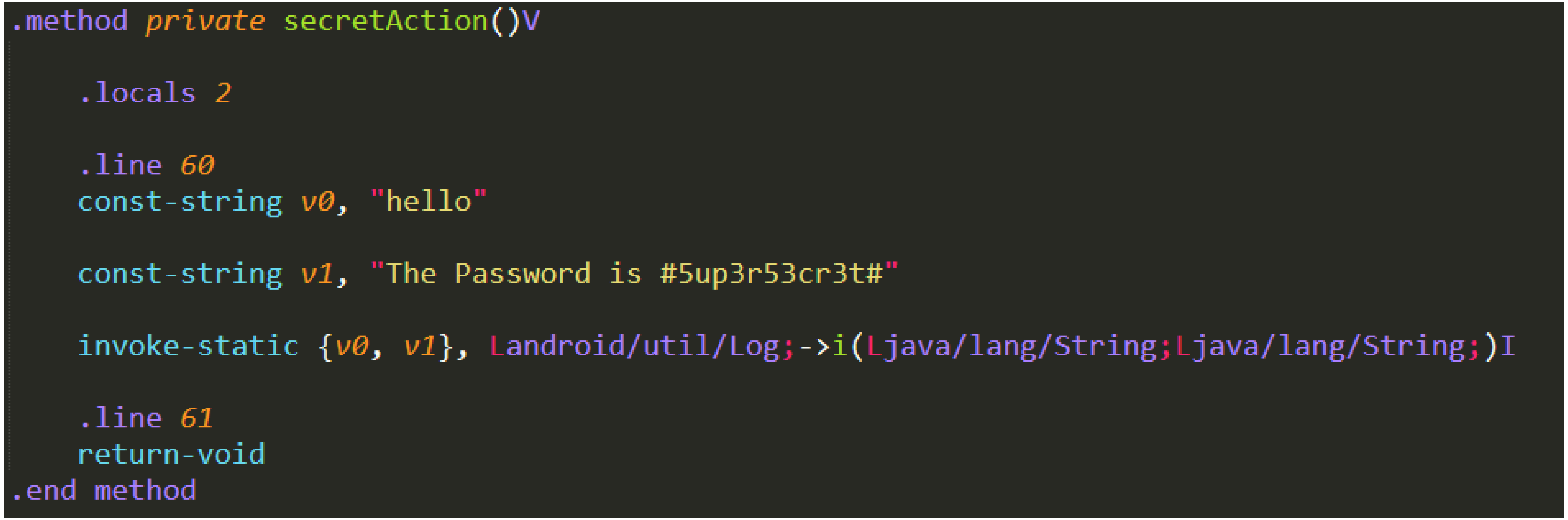


```
.super Ljava/lang/Object;

.method public static main([Ljava/lang/String;)V
    .registers 2

    sget-object v0, Ljava/lang/System;->out:Ljava/io/PrintStream;

    const-string    v1, "Hello World!"

    invoke-virtual {v0, v1}, Ljava/io/PrintStream;->println(Ljava/lang/String;)V

    return-void
.end method
```

Hello Android App



```
...
.line 27
.local v1, "fab":Lcom/google/android/material/floatingactionbutton/FloatingActionButton;
invoke-direct {p0}, Lpt/ua/deti/hello/MainActivity;->secretAction()V

.line 28
new-instance v2, Lpt/ua/deti/hello/MainActivity$1;
...
```



```
.method private secretAction()V

    .locals 2

    .line 60
    const-string v0, "hello"

    const-string v1, "The Password is #5up3r53cr3t#"

    invoke-static {v0, v1}, Landroid/util/Log;->i(Ljava/lang/String;Ljava/lang/String;)I

    .line 61
    return-void
.end method
```

Obfuscation

- Quite a few DEX "obfuscators" exist, with different approaches:
  - Functionally similar to binutils' strip, either java (ProGuard) or sDEX
  - Rename methods, field and class names
  - Break down string operations so as to "chop" hard-coded strings, or encrypt
  - Can use dynamic class loading (DexLoader classes) to impede static analysis
  - Can add dead code and dummy loops (at minor impact to performance)
  - Can also use `goto` into other instructions (or switches)

- Additional advantage: As obfuscators remove dead code, applications become smaller

Obfuscation

- In practice, obfuscation is quite limited, due to:
  - Reliance on Android Framework APIs (which remain unobfuscated)
  - JDWP and application debuggability at the Java level
  - If Dalvik can execute it, so can a proper analysis tool
  - Popular enough obfuscators have de-obfuscators...
  - Cannot obfuscate Activities

- About 25% of applications have some form of obfuscation
  - Dominik Wermke et al, “A Large Scale Investigation of Obfuscation Use in Google Play”, 2018 which analysed 1.7M apps

Obfuscation objectives

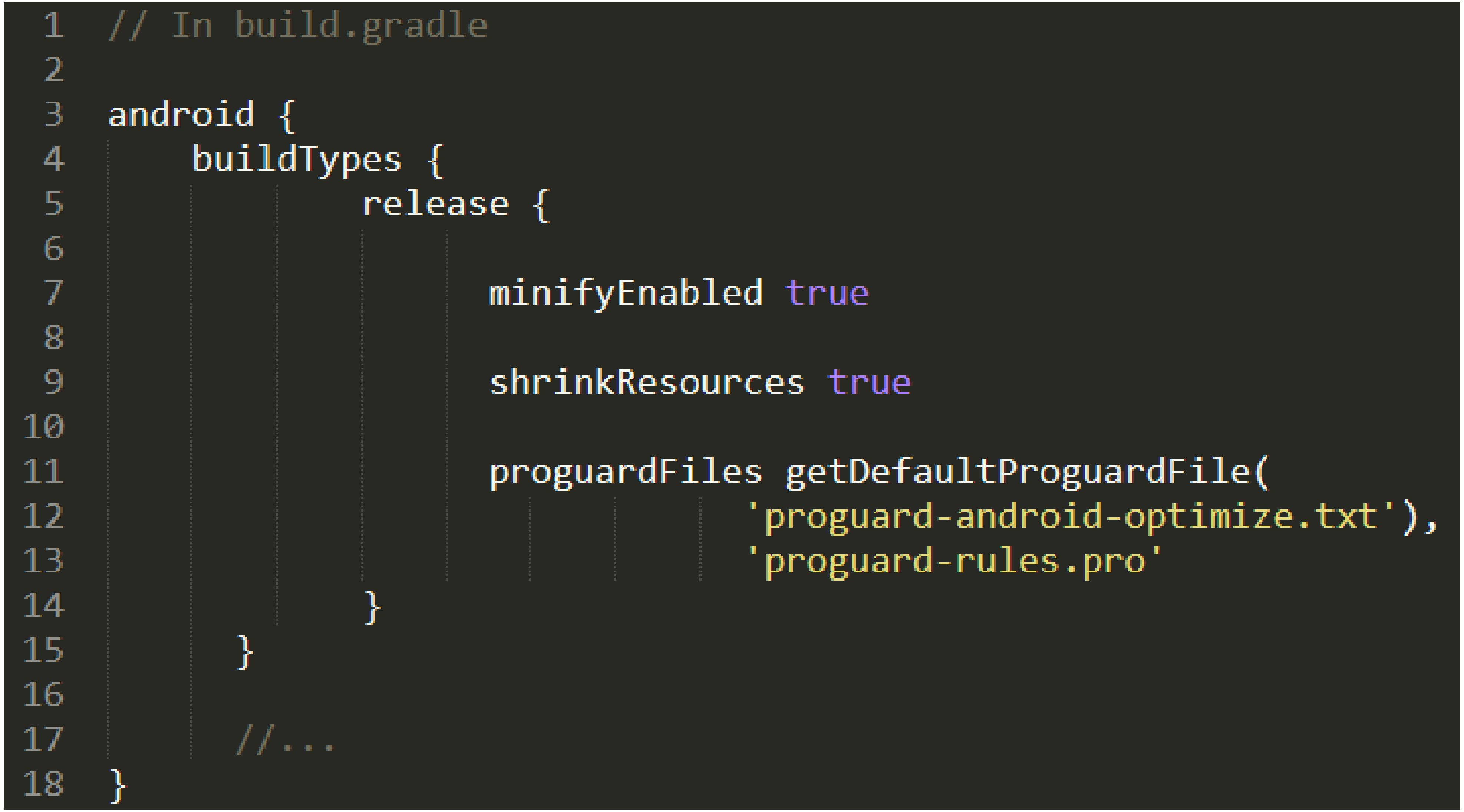
- **Code shrinking (or tree-shaking):** detects and safely removes unused classes, fields, methods, and attributes

- **Resource shrinking**: removes unused resources from a packaged app, including unused resources in the app's library dependencies.

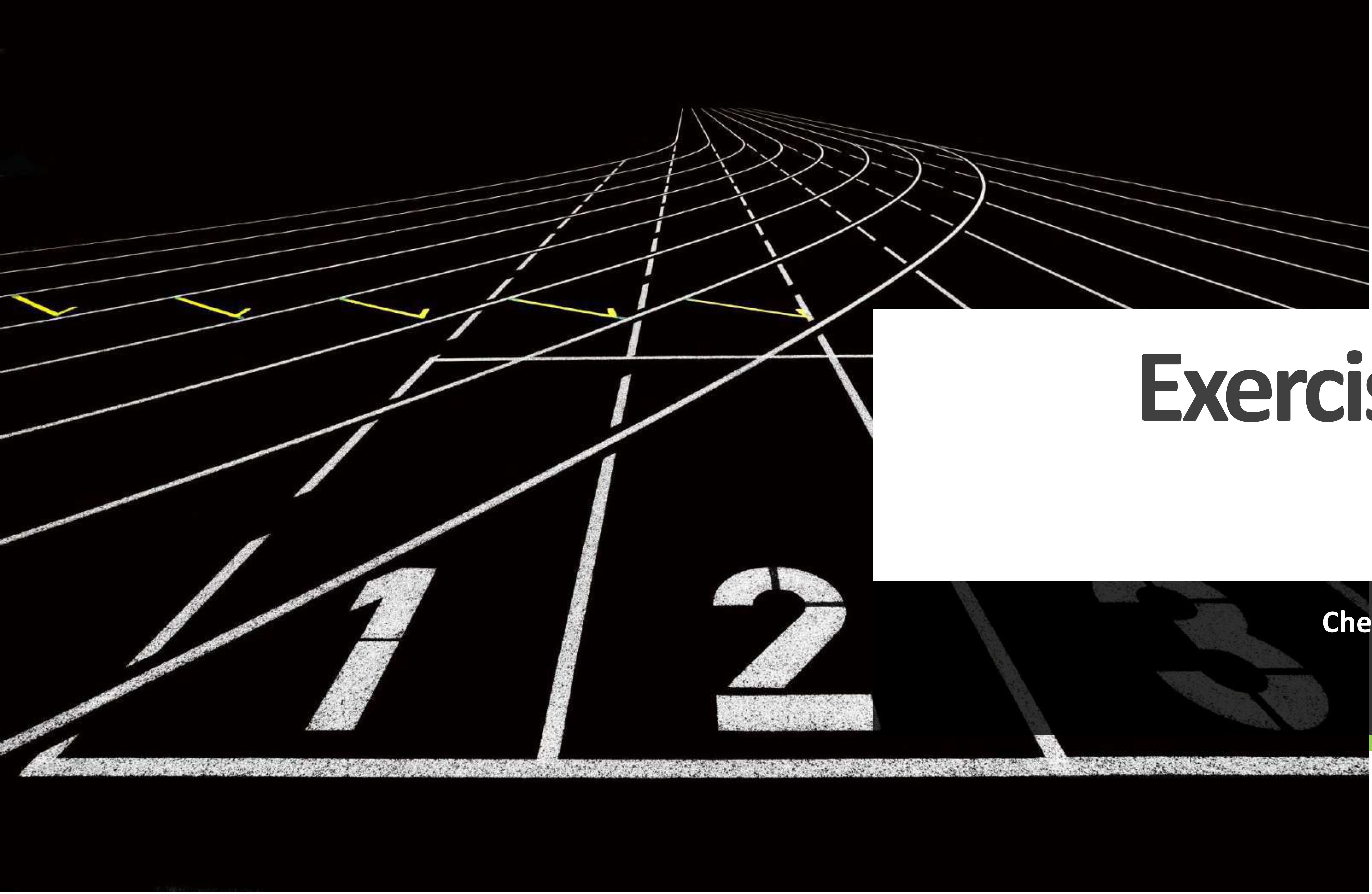
- **Obfuscation**: shortens the name of classes and members, which results in reduced DEX file sizes.

- **Optimization**: inspects and rewrites your code to further reduce the size of your app's DEX files.
  - Unreachable code is removed from the application

How to enable



```
// In build.gradle

android {
  buildTypes {
      release {

         minifyEnabled true

         shrinkResources true

         proguardFiles getDefaultProguardFile(
               'proguard-android-optimize.txt'),
               'proguard-rules.pro'
      }
   }

   //...
}
```



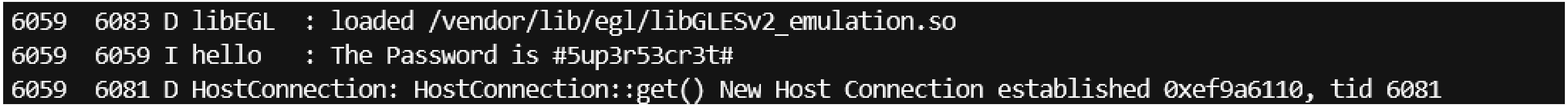
Exercise 3 & 4

Check the Class Workbook

Exercise 3 – Application is leaking data – Fix in Smali

- Process:
  - Extract data from apk with apktool: **`apktool d app-release.apk`**
  - Fix the smali code
  - Repackage the apk: **`apktool b app-release`**

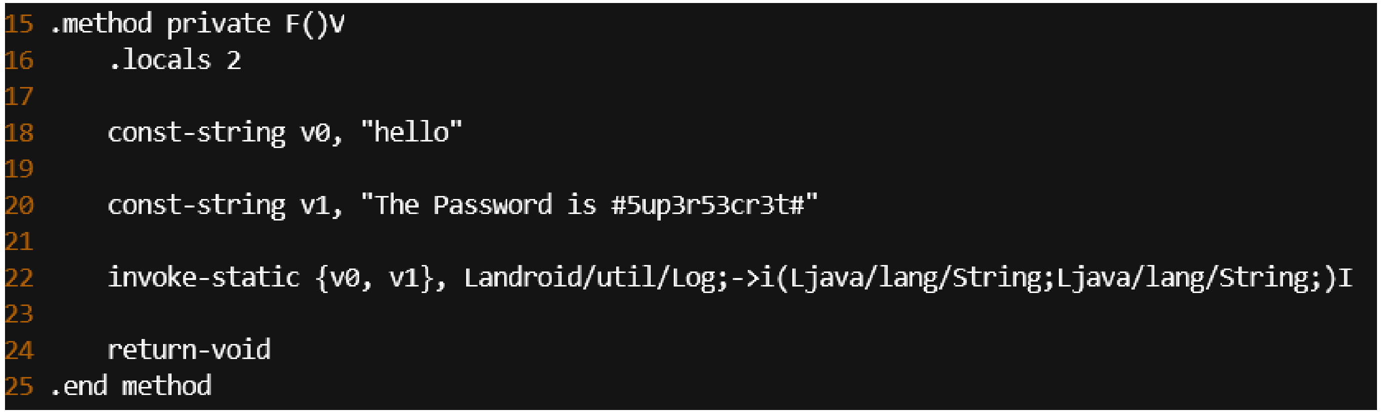
- The issue:
  - Clear the log: **`adb logcat  -c`**
  - Filter by pid: **`adb logcat --pid=$(adb shell pidof pt.ua.deti.hello)`**
  - Ignore all processes, except for tag hello: **`adb logcat -s “*:S hello”`**



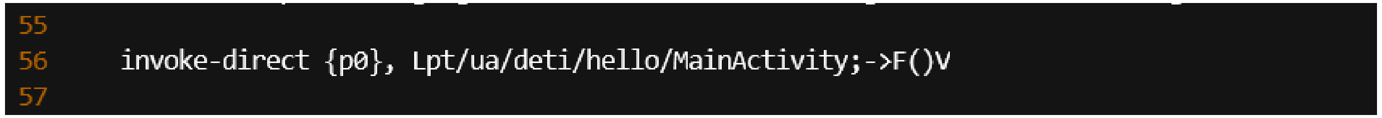
```
6059  6083 D libEGL   : loaded /vendor/lib/egl/libGLESv2_emulation.so
6059  6059 I hello    : The Password is #5up3r53cr3t#
6059  6081 D HostConnection: HostConnection::get() New Host Connection established 0xef9a6110, tid 6081
```

Exercise 3 – Application is leaking data – Fix in Smali

- Offending code: app-release/smali/pt/ua/deti/hello/MainActivity.smali

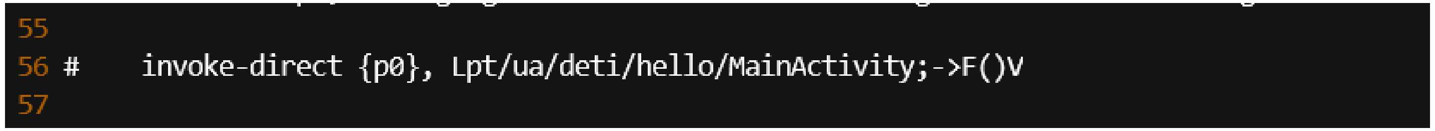


```
.method private F()V
    .locals 2

    const-string v0, "hello"

    const-string v1, "The Password is #5up3r53cr3t#"

    invoke-static {v0, v1}, Landroid/util/Log;->i(Ljava/lang/String;Ljava/lang/String;)I

    return-void
.end method
```



```

    invoke-direct {p0}, Lpt/ua/deti/hello/MainActivity;->F()V

```

- “The FIX”



```

#    invoke-direct {p0}, Lpt/ua/deti/hello/MainActivity;->F()V

```

- Deploy:
  - `apktool b app-release --use-aapt2`
  - `java -jar uber-apk-signer-1.2.1.jar --apks app-release/dist/app-release.apk`
  - `adb uninstall pt.ua.deti.hello`
  - `adb install app-release/dist/app-release-aligned-debugSigned.apk`

Exercise 3 – Application is leaking data – Fix in Smali

- Deploy:
  - apktool b app-release --use-aapt2
  - java -jar uber-apk-signer-1.2.1.jar --apks app-release/dist/app-release.apk
  - adb uninstall pt.ua.deti.hello
  - adb install app-release/dist/app-release-aligned-debugSigned.apk

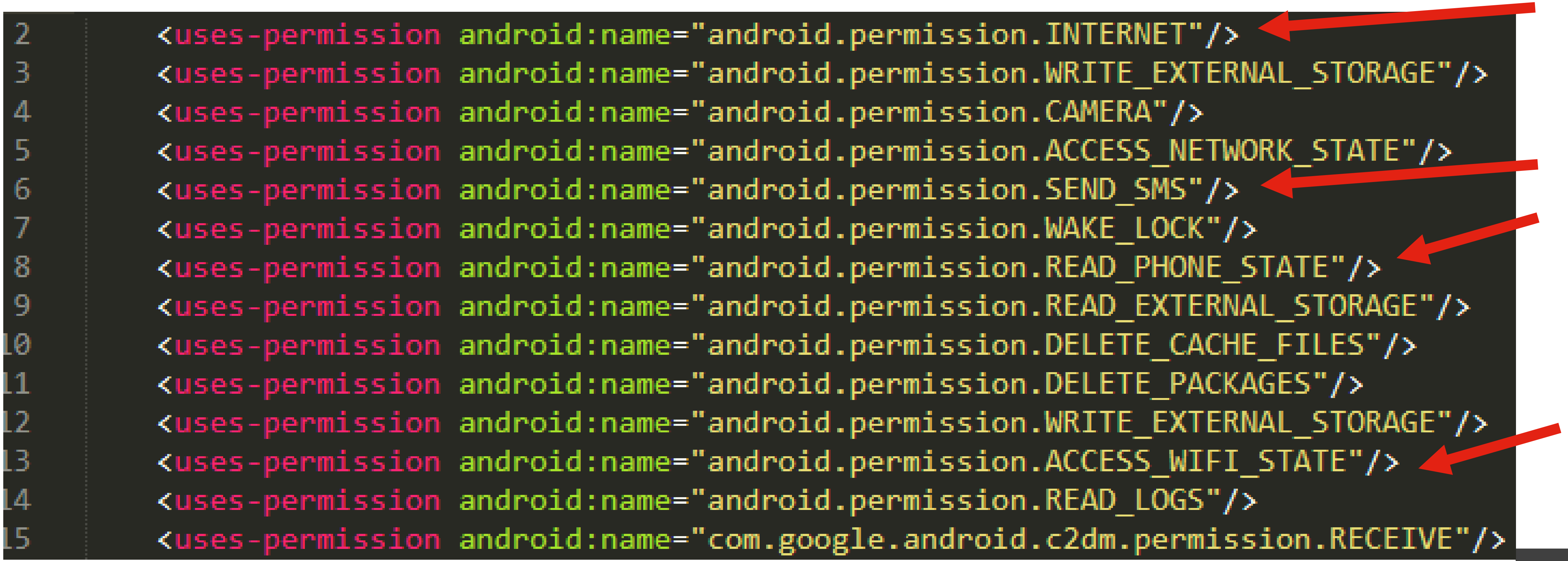
- Verification:
  - `adb logcat –s “*:S hello”`

Exercise 4 – Thai Camera is sending SMS?

- Approach
  - Extract all code and resources: **jadx-gui**
  - Inspect Manifest for a suspicious permission (Send SMS): **AndroidManifest.XML**
  - Determine if the app is sending SMS: Check the java classes, look for SMS send methods
  - Determine if the SMS is sent without interaction from the user
    - How are functions called?
    - What is the call flow?

Exercise 4 – Thai Camera is sending SMS?

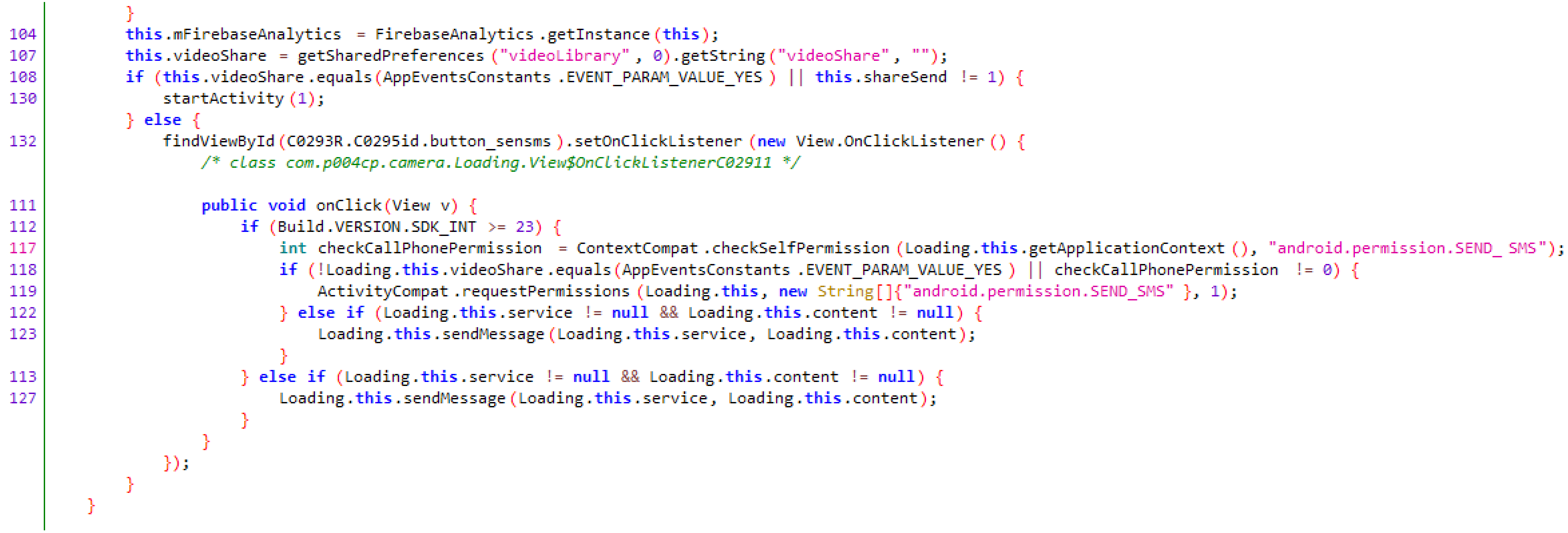
- For a camera application, some permissions are suspicious
  - Including **`android.permission.SEND_SMS`**
  - Therefore, we have indications of possible taints



```
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.DELETE_CACHE_FILES"/>
    <uses-permission android:name="android.permission.DELETE_PACKAGES"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.READ_LOGS"/>
    <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE"/>
```

Exercise 4 – Thai Camera is sending SMS?

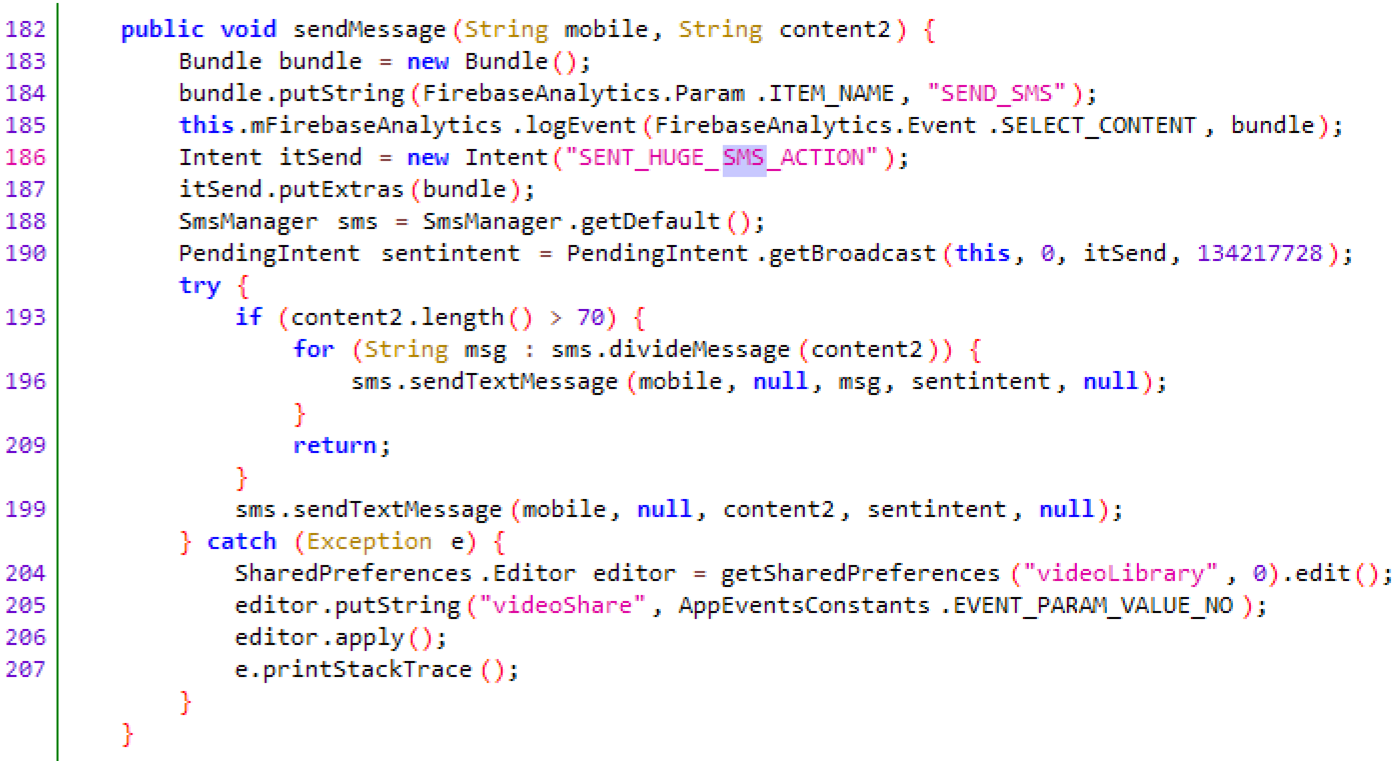
- In com.p004cp.camera.loading an SMS is sent
  - As an action of clicking a button. With static analysis is seems to be ok.



```
    }
    this.mFirebaseAnalytics  = FirebaseAnalytics .getInstance (this);
    this.videoShare  = getSharedPreferences ("videoLibrary" , 0).getString ("videoShare" , "");
    if (this.videoShare .equals (AppEventsConstants .EVENT_PARAM_VALUE_YES ) || this.shareSend  != 1) {
        startActivity (1);
    } else {
        findViewById (C0293R.C0295id.button_sensms ).setOnClickListener (new View.OnClickListener () {
            /* class com.p004cp.camera.Loading.View$OnClickListenerC02911 */

            public void onClick(View v) {
                if (Build.VERSION.SDK_INT >= 23) {
                    int checkCallPhonePermission  = ContextCompat .checkSelfPermission (Loading.this.getApplicationContext (), "android.permission.SEND_ SMS");
                    if (!Loading.this.videoShare .equals (AppEventsConstants .EVENT_PARAM_VALUE_YES ) || checkCallPhonePermission  != 0) {
                        ActivityCompat .requestPermissions (Loading.this, new String[]{"android.permission.SEND_SMS" }, 1);
                    } else if (Loading.this.service  != null && Loading.this.content  != null) {
                        Loading.this.sendMessage (Loading.this.service , Loading.this.content );
                    }
                } else if (Loading.this.service  != null && Loading.this.content  != null) {
                    Loading.this.sendMessage (Loading.this.service , Loading.this.content );
                }
            }
        });
    }
}
```

Exercise 4 – Thai Camera is sending SMS?

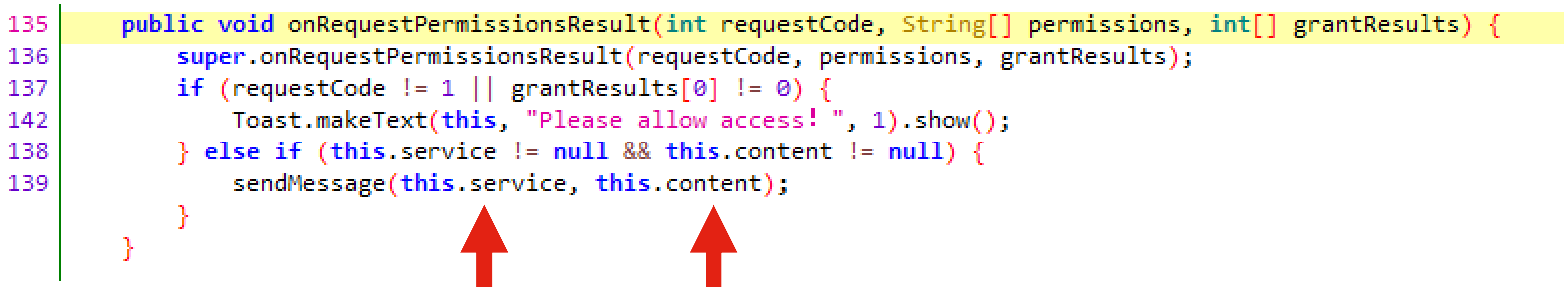
- There is a **`sendMessage`** method with two arguments (number and text)
  - Logs the event to Firebase
  - Splits the message in chunks and submits multiple SMS
  - But… how is this function called?



```java
public void sendMessage(String mobile, String content2) {
    Bundle bundle = new Bundle();
    bundle.putString(FirebaseAnalytics.Param.ITEM_NAME, "SEND_SMS");
    this.mFirebaseAnalytics.logEvent(FirebaseAnalytics.Event.SELECT_CONTENT, bundle);
    Intent itSend = new Intent("SENT_HUGE_SMS_ACTION");
    itSend.putExtras(bundle);
    SmsManager sms = SmsManager.getDefault();
    PendingIntent sentintent = PendingIntent.getBroadcast(this, 0, itSend, 134217728);
    try {
        if (content2.length() > 70) {
            for (String msg : sms.divideMessage(content2)) {
                sms.sendTextMessage(mobile, null, msg, sentintent, null);
            }
            return;
        }
        sms.sendTextMessage(mobile, null, content2, sentintent, null);
    } catch (Exception e) {
        SharedPreferences.Editor editor = getSharedPreferences("videoLibrary", 0).edit();
        editor.putString("videoShare", AppEventsConstants.EVENT_PARAM_VALUE_NO);
        editor.apply();
        e.printStackTrace();
    }
}
```

Exercise 4 – Thai Camera is sending SMS?

- In several places, but one is strange



```java
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode != 1 || grantResults[0] != 0) {
        Toast.makeText(this, "Please allow access! ", 1).show();
    } else if (this.service != null && this.content != null) {
        sendMessage(this.service, this.content);
    }
}
```

this.service = phone number

this.content = text to send

Where are these coming from?

Exercise 4 – Thai Camera is sending SMS?

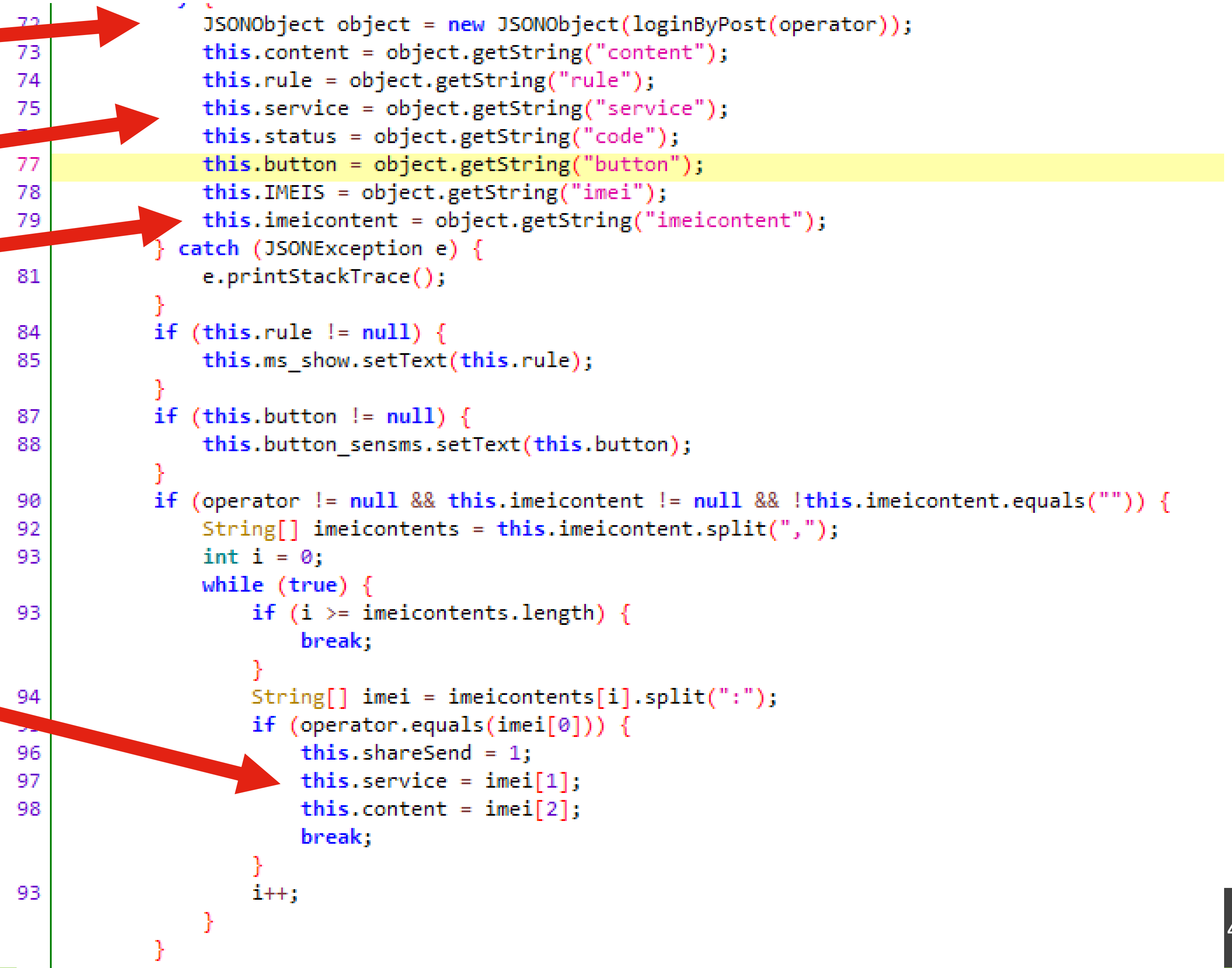
- `Loading::onCreate`

Login and gets data

Sets the `this.service`

IMEI?

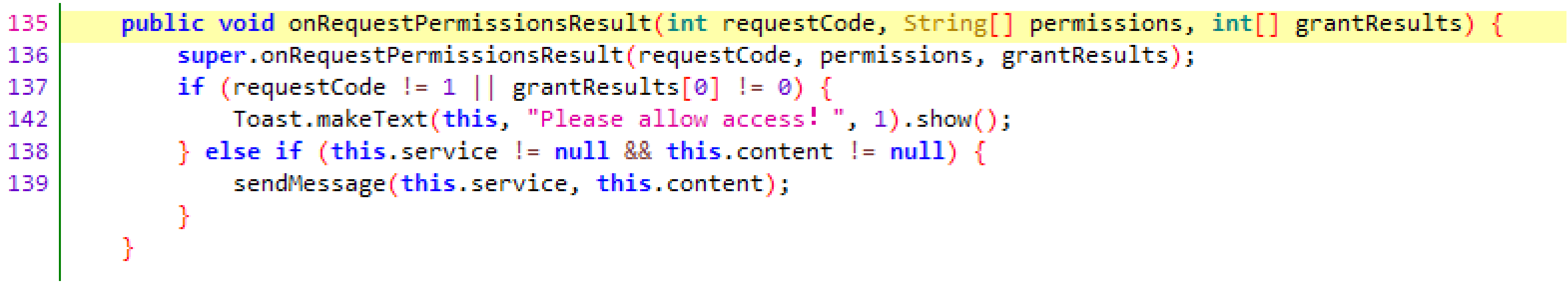
Under some situations, `this.service` is set again and seems to be dependent on the operator or IMEI



```
        JSONObject object = new JSONObject(loginByPost(operator));
        this.content = object.getString("content");
        this.rule = object.getString("rule");
        this.service = object.getString("service");
        this.status = object.getString("code");
        this.button = object.getString("button");
        this.IMEIS = object.getString("imei");
        this.imeicontent = object.getString("imeicontent");
    } catch (JSONException e) {
        e.printStackTrace();
    }
    if (this.rule != null) {
        this.ms_show.setText(this.rule);
    }
    if (this.button != null) {
        this.button_sensms.setText(this.button);
    }
    if (operator != null && this.imeicontent != null && !this.imeicontent.equals("")) {
        String[] imeicontents = this.imeicontent.split(",");
        int i = 0;
        while (true) {
            if (i >= imeicontents.length) {
                break;
            }
            String[] imei = imeicontents[i].split(":");
            if (operator.equals(imei[0])) {
                this.shareSend = 1;
                this.service = imei[1];
                this.content = imei[2];
                break;
            }
            i++;
        }
    }
```

Exercise 4 – Thai Camera is sending SMS?

- Going back to the previous location
  - The permission is requested
  - And if authorized and `this.service` is set, an SMS is sent automatically (without user interaction)



```java
    public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        if (requestCode != 1 || grantResults[0] != 0) {
            Toast.makeText(this, "Please allow access! ", 1).show();
        } else if (this.service != null && this.content != null) {
            sendMessage(this.service, this.content);
        }
    }
```

- To recap:
  - Application sends SMS: True
  - Application sends SMS onClick by the user: True
  - However….
  - An SMS is sent automatically when the permission is granted
  - The destination number is not controlled by the user. Value is set on create, comes from external server
  - It has to do with IMEI and Operator: This is an indication of a Premium SMS Fraud

Can this process be improved? Yes

- **Flow Analysis**: the execution flow can be analyzed and reconstructed, allowing to understand entry and sink points
  - Identify all methods, and their callers: Sources/Entry Points
    - Events, Intent Receivers
  - Identify which arguments are used… eventually do symbolic analysis
  - Identify which Android APIs are called: Sink Points
    - Information is sent/registered using the Android API

- **Taint Analysis**: Identify patterns which may indicate suspicious behavior
  - E.g. access contacts, upload contacts

- **Dynamic Analysis**: actually analyze what the application done, in real time

Flow Analysis and Taint Analysis

- Android Studio:
  - If Java code can be obtained, Android Studio creates call flows
    - Analyze Tab -> Data Flow From Here

- Quark:
  - One of many tools providing Flow Analysis and Taint Analysis
  - Targeted towards malware
    - Identifies malicious or suspicious behavior, and ranks each taint
    - Provides limited call graph information through static analysis
  - Based on smali directly from the apk
  - Available: https://github.com/quark-engine/quark-engine

Quark and Thai Camera?

- install:
  - pip3 install --user quark-engine
  - freshquark

- `quark -s –a “ThaiCamera_v1.2.apk”`
  - `[!] WARNING: Moderate Risk`

- Some indicators (remember, it’s a Camera App!)
  - `Get calendar information`
  - `Read sensitive data(SMS, CALLLOG) and put it into JSON object`
  - `Get the network operator name`
  - `Get data from HTTP and send SMS`
  - `Send IMSI over Internet`
  - `Get the network operator name and IMSI`
  - `Write SIM card serial number into a file`
  - `Write the phone number into a file`
  - `Check if successfully sending out SMS`

- But: It is common to find taints on included SDKs (google, facebook)
  - Analyst must look at the actual location of the taints

Android – Static Analysis 2

REVERSE ENGINEERING

deti universidade de aveiro
departamento de eletrónica,
telecomunicações e informática

João Paulo Barraca

Native Applications

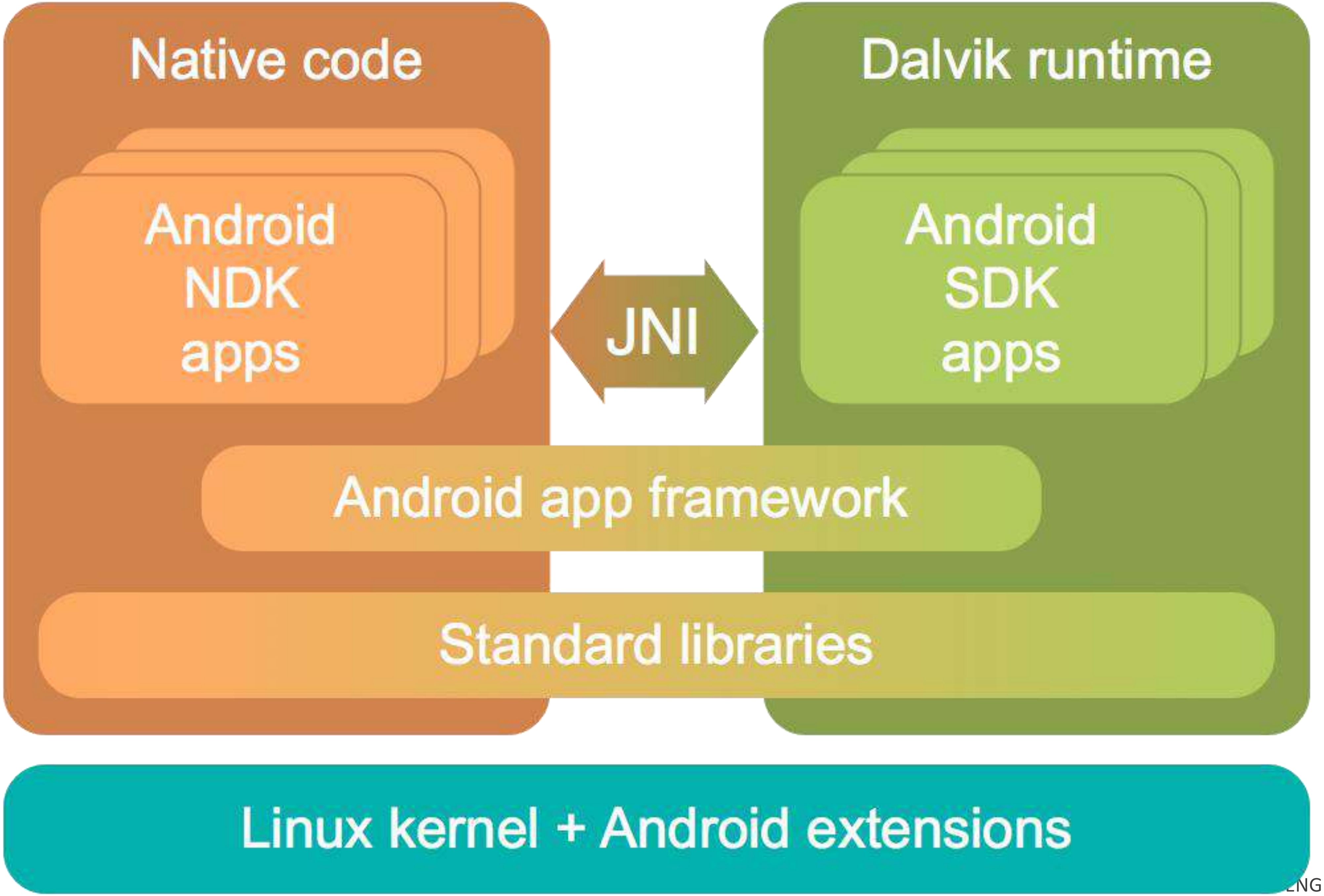
Native Applications

- Apps developed with OS provider’s language and frameworks.
  - Java, Kotlin, Ojective-C, Swift
- Android applications are compiled and packaged with resources
  - Reversing such app can be done to Java (JADX) or Smali (apktool)
- Can access all API’s made available by OS vendor.
- But…
  - SDK’s are platform-specific.
  - Each mobile OS comes with its own unique tools and GUI toolkit.
  - Developing a world wide app requires multiple implementations

Java Native Interface

- Java applications can call functions from external libraries
  - Libraries can be implemented in Java, and packaged as classes
  - Libraries can also be implemented in any other language
    - Providing that an interfaces allows serialization and name resolution
- JNI: allows the definition of Java methods, whose implementation is present in native code
  - When a method is invoked, the objects are serialized, and the respective native symbol is loaded and the code executed.
  - There is a penalty due to serialization, but also a performance boost due to native code execution.
  - References:
    - JNI Functions (oracle.com)
    - Contents (oracle.com)
- Standard mechanism for Java (not specific for android)

JNI



Native code

Android NDK apps

JNI

Dalvik runtime

Android SDK apps

Android app framework

Standard libraries

Linux kernel + Android extensions

Android Native Development Kit (NDK)

- Provides a Dev. Kit allowing C/C++ applications to access Android resources
  - Similar to the standard SDK available to Java applications

- Developers may choose how to develop application code
  - Java: faster development and richer API
  - Native: faster execution, access to Linux subsystem, and more complex reverse engineering
    - Sometimes binary blobs are the only method to access a cryptographic method, DRM or hardware device
    - Sometimes the developer wishes to further obfuscate the code by compiling it to native code

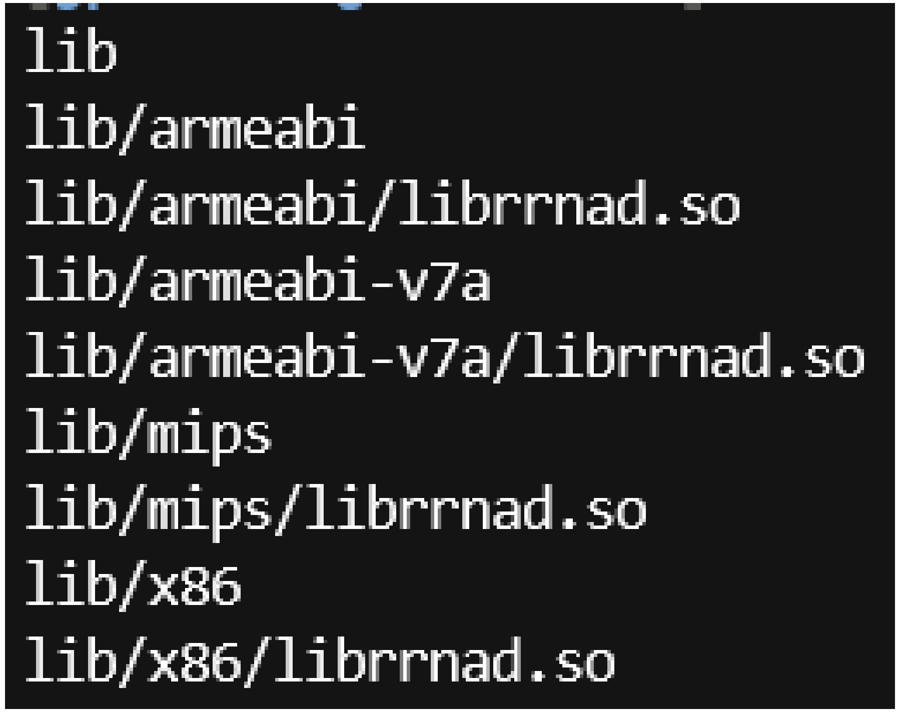
- As **libraries are native**, an application must include multiple implementations
  - **One for each architecture**
  - A new device may not use applications that lack an implementation for that architecture
  - Implies using portable code that works in multiple architectures (arm, armv7, arm64, x86, x64, …)

Android binary libraries – Mediacode.apk

- Application contains DEX code and binary blobs

- One version for each architecture
  - armeabi: ARM 32bits no Floating Point
  - mips: MIPS
  - x86: intel X86 32bits

- Libraries export symbols to be used through JNI
  - `nm -gD lib/x86/librrnad.so | grep JNI`



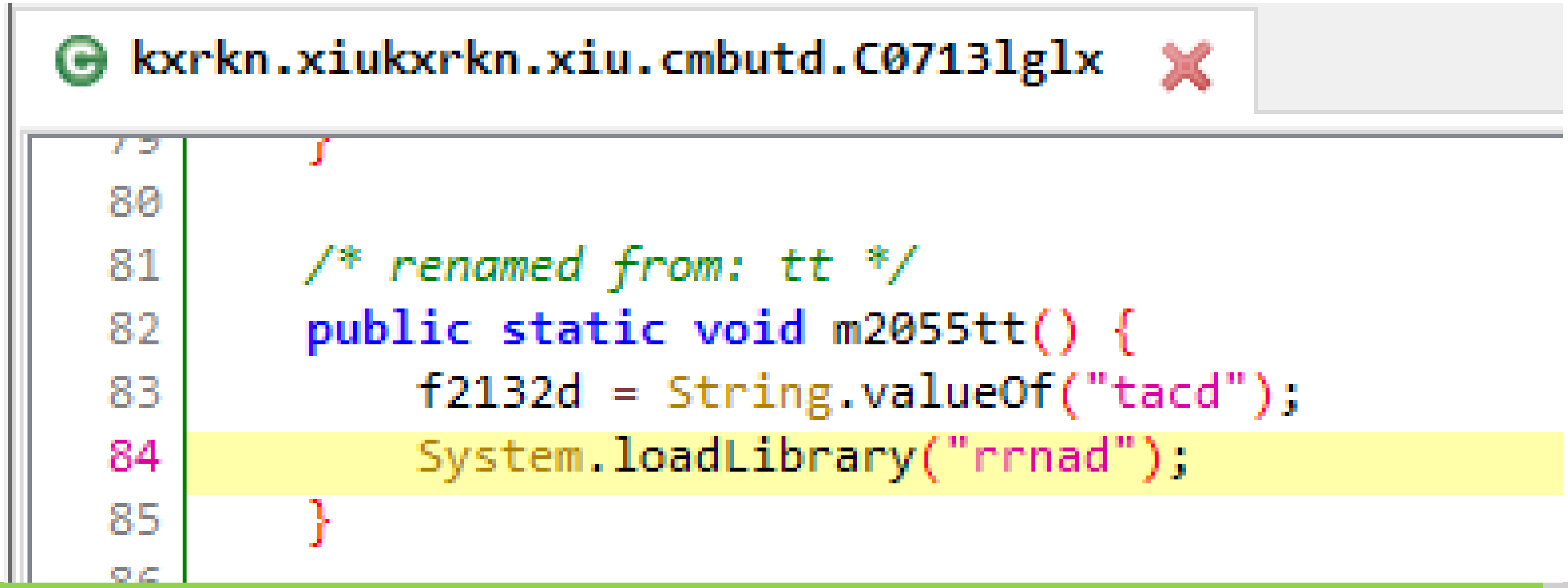
```
lib
lib/armeabi
lib/armeabi/librrnad.so
lib/armeabi-v7a
lib/armeabi-v7a/librrnad.so
lib/mips
lib/mips/librrnad.so
lib/x86
lib/x86/librrnad.so
```

Mediacode.apk

Android binary libraries – Mediacode.apk

- Before the binary libraries can be used, Java must load them
  - **`System.loadLibrary`**: argument is the library name (without lib, architecture or .so)
  - **`System.load`**: generic object load. Argument is the full path to the object
  - The JNI_OnLoad method is called automatically (in the lib)
    - Allows automatic setup of data structures and generic initialization
    - May be abused if malware is present

- Without the library, application will crash when external methods are requested



```
    }

    /* renamed from: tt */
    public static void m2055tt() {
        f2132d = String.valueOf("tacd");
        System.loadLibrary("rrnad");
    }
```

JNI Arguments

- Native methods support arguments from Java code
  - Arguments are pointers to Java structures
  - Must be processed using specific methods, capable of handling the native Java types

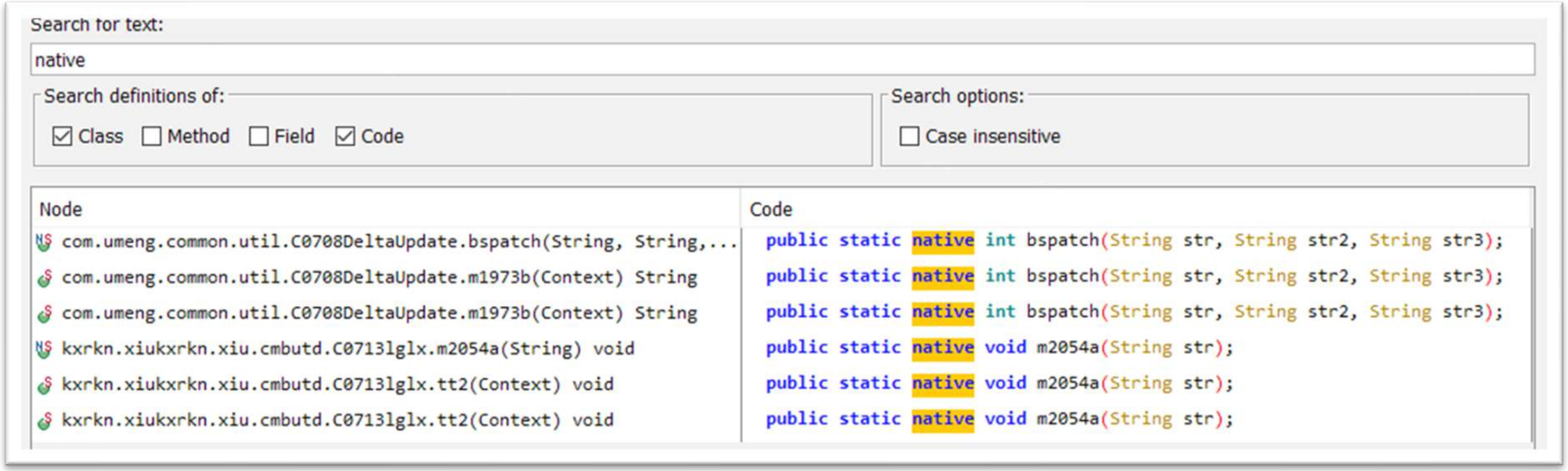
- Native methods can also call Java methods, and classes
  - Mainly achieved by the first argument of any JNI method: `JNIEnv*`

- `JNIEnv*` is a pointer to a structure with a large number of functions.
  - JNI Methods use it to invoke Java methods and handle Java types

Android binary libraries – Mediacode.apk

- In the java world native methods are declared:
  - With the keyword `native`
  - Without implementation

- Easy to spot if we have the java or smali code
  - Java: `public native String decryptString(String)`
  - Smali: `.method public native decryptString(Ljava/lang/String;)Ljava/lang/String`



Search for text:

native

Search definitions of:

☑ Class ☐ Method ☐ Field ☑ Code

Search options:

☐ Case insensitive

Node	Code
com.umeng.common.util.C0708DeltaUpdate.bspatch(String, String,...	public static native int bspatch(String str, String str2, String str3);
com.umeng.common.util.C0708DeltaUpdate.m1973b(Context) String	public static native int bspatch(String str, String str2, String str3);
com.umeng.common.util.C0708DeltaUpdate.m1973b(Context) String	public static native int bspatch(String str, String str2, String str3);
kxrkn.xiukxrkn.xiu.cmbutd.C0713lglx.m2054a(String) void	public static native void m2054a(String str);
kxrkn.xiukxrkn.xiu.cmbutd.C0713lglx.tt2(Context) void	public static native void m2054a(String str);
kxrkn.xiukxrkn.xiu.cmbutd.C0713lglx.tt2(Context) void	public static native void m2054a(String str);

JNI Dynamic Linking

- Dynamic linking is done “automagically” as long as the names of the methods in the library follow a fixed template
  - The library is loaded into the JVM and the methods are linked automatically
  - Implies that symbols are present in the library (not stripped)

- Assuming that our hello world app had a class named Worker, loading a method named doWork, the method in the function would be named:

```
java_pt_ua_deti_hello_Worker_doWork()
```

magic

Package name

Class name

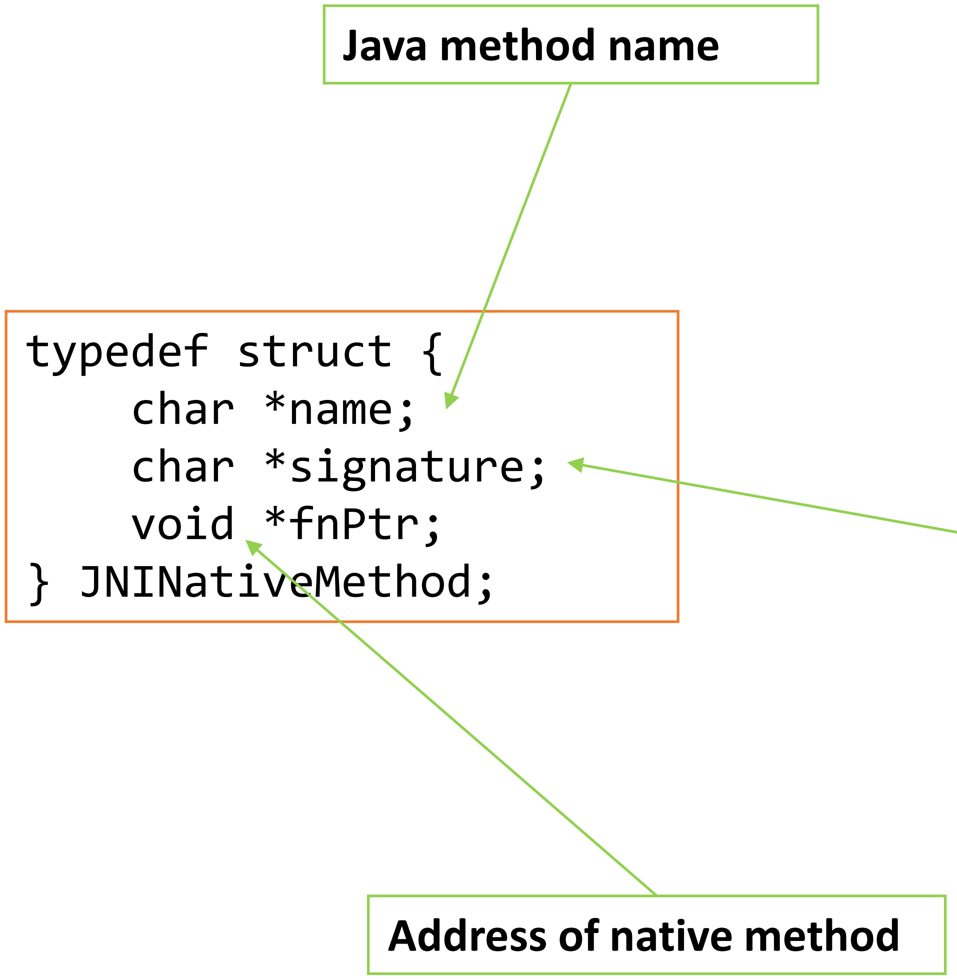
method

JNI Static Linking

- Linking must be done “manually”, by the binary code, before the methods are used
  - Allows methods to have any name (read obfuscation!)
  - A fixed method (`JNI_OnLoad`) is called after the lib is loaded
  - Library registers the mapping between java methods and native methods using `RegisterNatives`.
    - Must do this once for each method called.

```
jint RegisterNatives(JNIEnv *env, jclass clazz, const JNINativeMethod *methods, jint nMethods);

typedef struct {
    char *name;
    char *signature;
    void *fnPtr;
} JNINativeMethod;
```

JNI Static Linking



Java method name

```
typedef struct {
    char *name;
    char *signature;
    void *fnPtr;
} JNINativeMethod;
```

Address of native method

Signature using the following specifiers:

- `Z: boolean`
- `B: byte`
- `C: char`
- `S: short`
- `I: int`
- `J: long`
- `F: float`
- `D: double`
- `L fully-qualified-class ; :fully-qualified-class`
- `[ type: type[]`
- `( arg-types ) ret-type: method type`
- `V: void`

`String foo(Int, Boolean)` would result in:
`(IB)Ljava/lang/String`

[JNI Types and Data Structures (oracle.com)]

JNI Static Linking

- Reverse engineering of the library blob is the most viable alternative
  - Some symbols must always be available: **`JNI_Load`**
  - Remaining symbols usually are available, although they may have obfuscated names

- Process
  - Load the library in a tool: ghidra, IDA, BinaryNinja, R2, etc…
  - Find the `JNI_Load method`
  - Determine when `RegisterNatives` is called
  - Determine the arguments passed to the function
    - Will allow determining the method mapping and the arguments of each function
    - Actually, the arguments may also help identifying the method

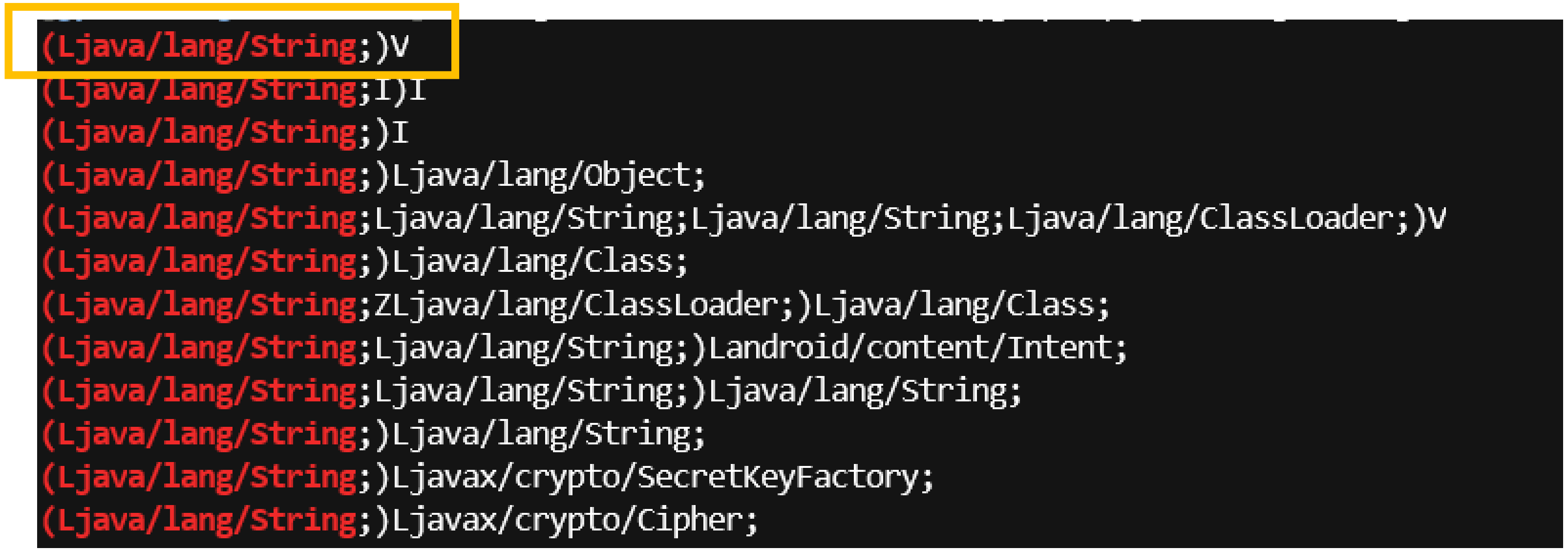
Exercises 1 and 2

Determine which method are actually loaded from the MediaCodec.apk shared libraries.

strings

- Do we have interfaces matching the functions we know to be native?
  - `int bspatch(String str, String str2, String str3)`
  - `void m2054a(String s)`

```
strings lib/x86/librrnad.so    |grep "(Ljava/lang/String"
```



```
(Ljava/lang/String;)V
(Ljava/lang/String;I)I
(Ljava/lang/String;)I
(Ljava/lang/String;)Ljava/lang/Object;
(Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;Ljava/lang/ClassLoader;)V
(Ljava/lang/String;)Ljava/lang/Class;
(Ljava/lang/String;ZLjava/lang/ClassLoader;)Ljava/lang/Class;
(Ljava/lang/String;Ljava/lang/String;)Landroid/content/Intent;
(Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;
(Ljava/lang/String;)Ljava/lang/String;
(Ljava/lang/String;)Ljavax/crypto/SecretKeyFactory;
(Ljava/lang/String;)Ljavax/crypto/Cipher;
```

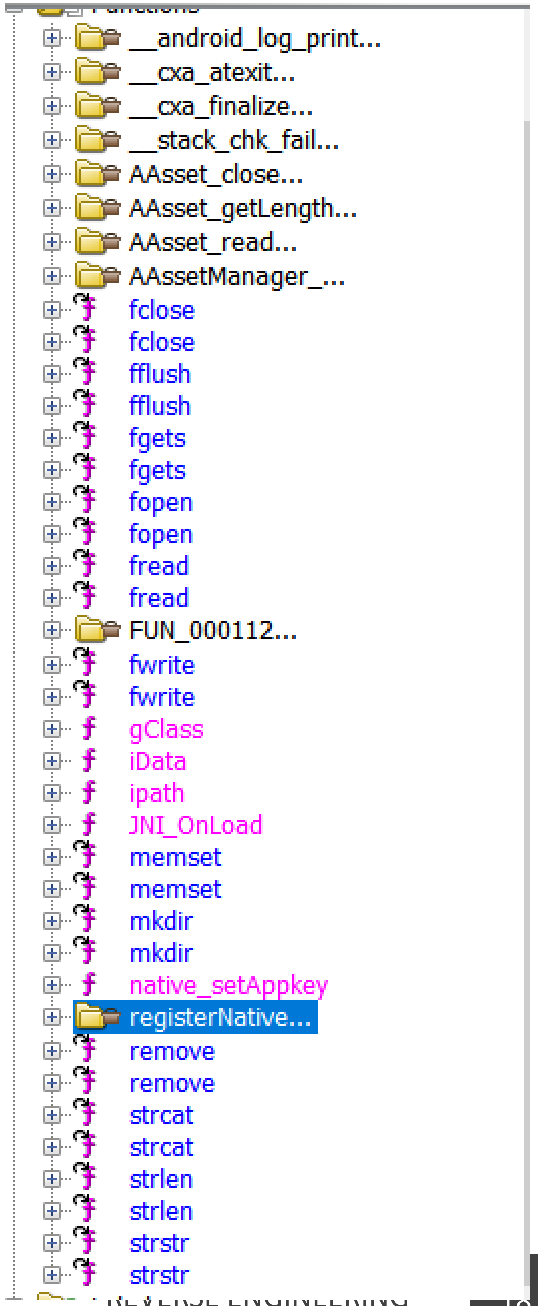
nm

- Do we have dynamic linking?
- Let's look for methods following the known pattern

- `nm –gD lib/x86/librrnad.so |grep java_`
  - None...
- Conclusion
  - We have artifacts pointing to Java types
  - We do not have indication of Dynamic Linking

ghidra

- Open ghidra
- Create a new project
- Load a library
  - I loaded all and selected ARMEABI
- Several interesting functions discovered
  - **`JNI_Load`**
  - `registerNatives, registerNativeMethods`
  - `FUN_00011230, FUN_000270, FUN_11290, FUN_112b4`
  - `native_setAppKey`
- Coherent with Static Linking
- Explore the functions, exports, Classes, etc… lots of info

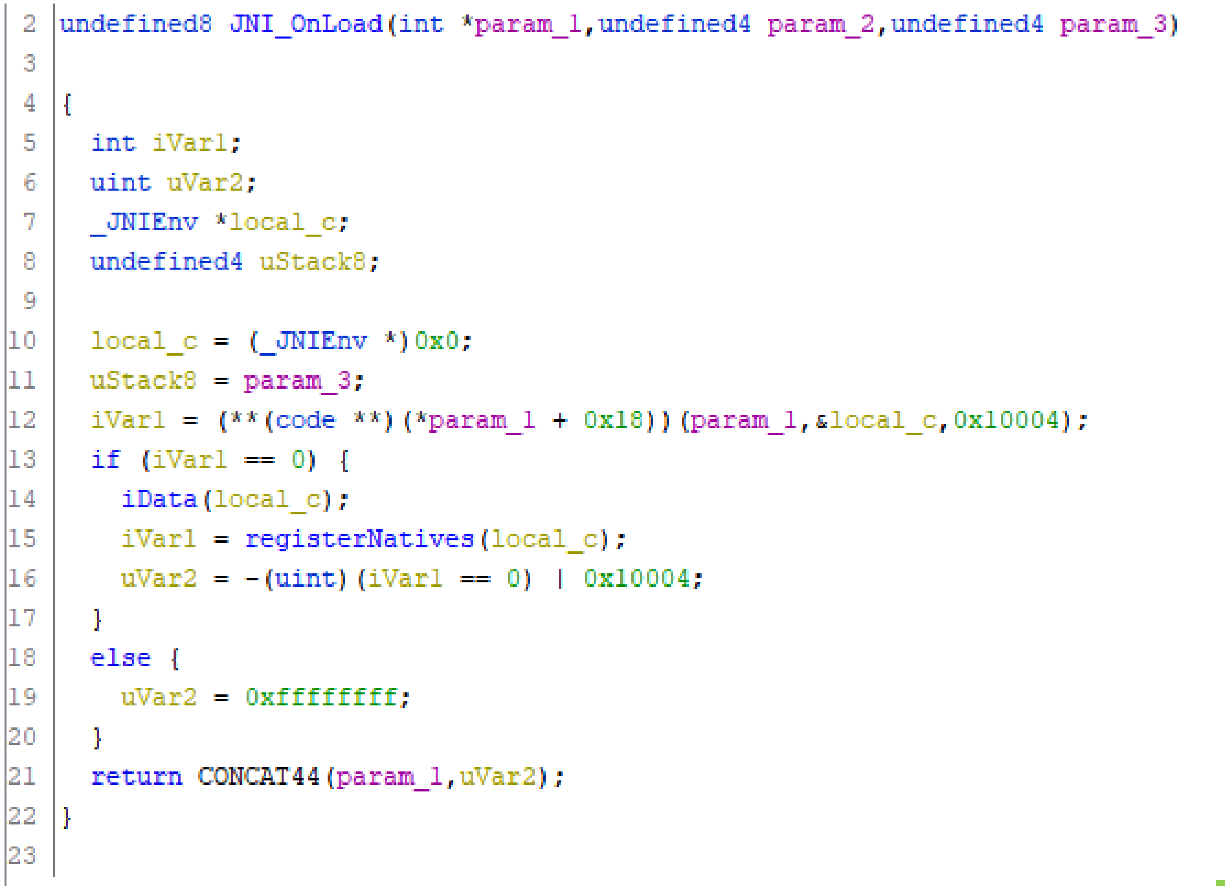


```
Functions
  __android_log_print...
  __cxa_atexit...
  __cxa_finalize...
  __stack_chk_fail...
  AAsset_close...
  AAsset_getLength...
  AAsset_read...
  AAssetManager_...
  fclose
  fclose
  fflush
  fflush
  fgets
  fgets
  fopen
  fopen
  fread
  fread
  FUN_000112...
  fwrite
  fwrite
  gClass
  iData
  ipath
  JNI_OnLoad
  memset
  memset
  mkdir
  mkdir
  native_setAppkey
  registerNative...
  remove
  remove
  strcat
  strcat
  strlen
  strlen
  strstr
  strstr
```

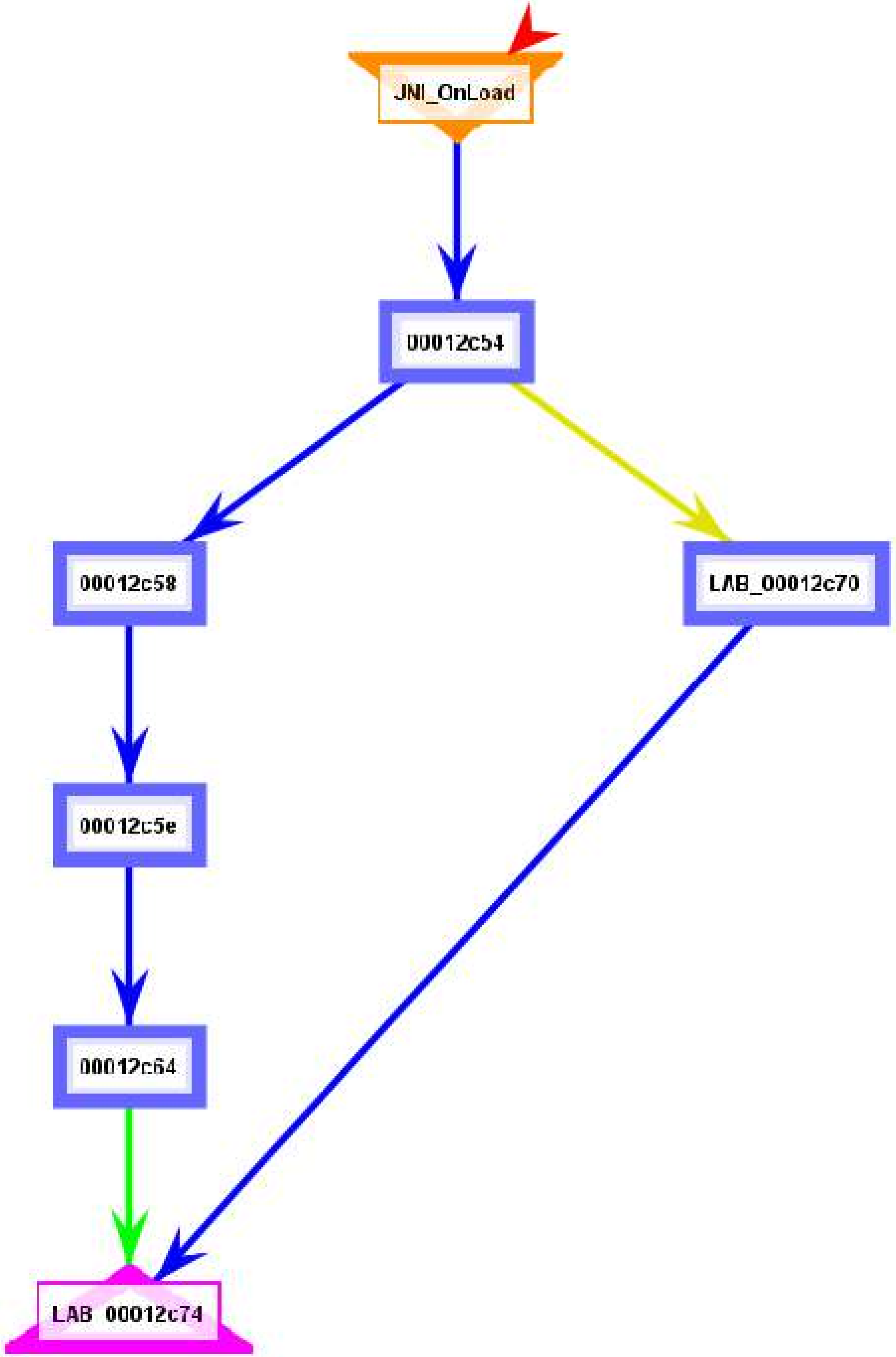
ghidra

- Graph -> Block Flow from JNI_OnLoad

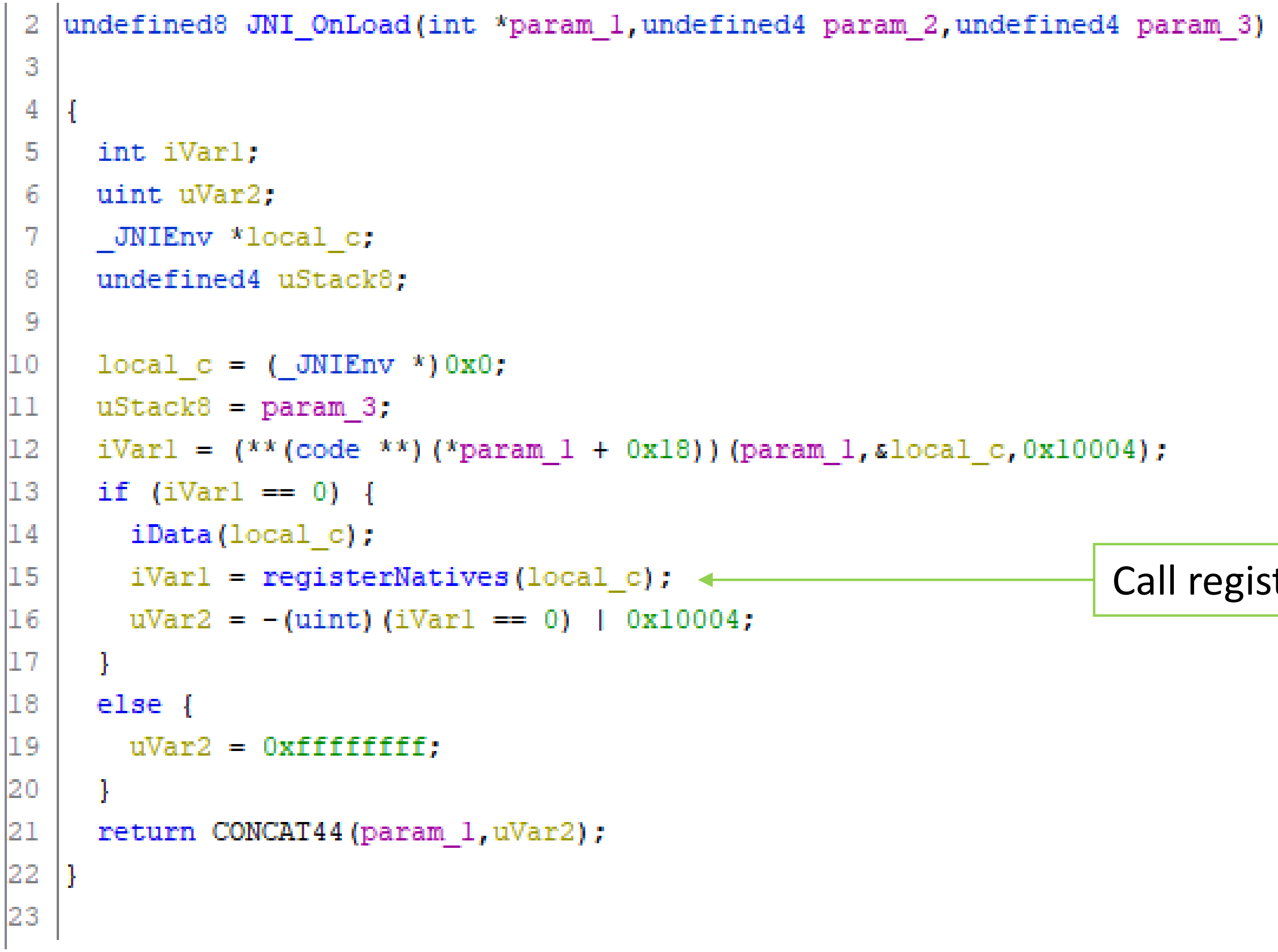
- Decompile JNI_OnLoad



```
undefined8 JNI_OnLoad(int *param_1,undefined4 param_2,undefined4 param_3)

{
  int iVar1;
  uint uVar2;
  _JNIEnv *local_c;
  undefined4 uStack8;

  local_c = (_JNIEnv *)0x0;
  uStack8 = param_3;
  iVar1 = (**(code **)(*param_1 + 0x18))(param_1,&local_c,0x10004);
  if (iVar1 == 0) {
    iData(local_c);
    iVar1 = registerNatives(local_c);
    uVar2 = -(uint)(iVar1 == 0) | 0x10004;
  }
  else {
    uVar2 = 0xffffffff;
  }
  return CONCAT44(param_1,uVar2);
}

```



JNI_OnLoad

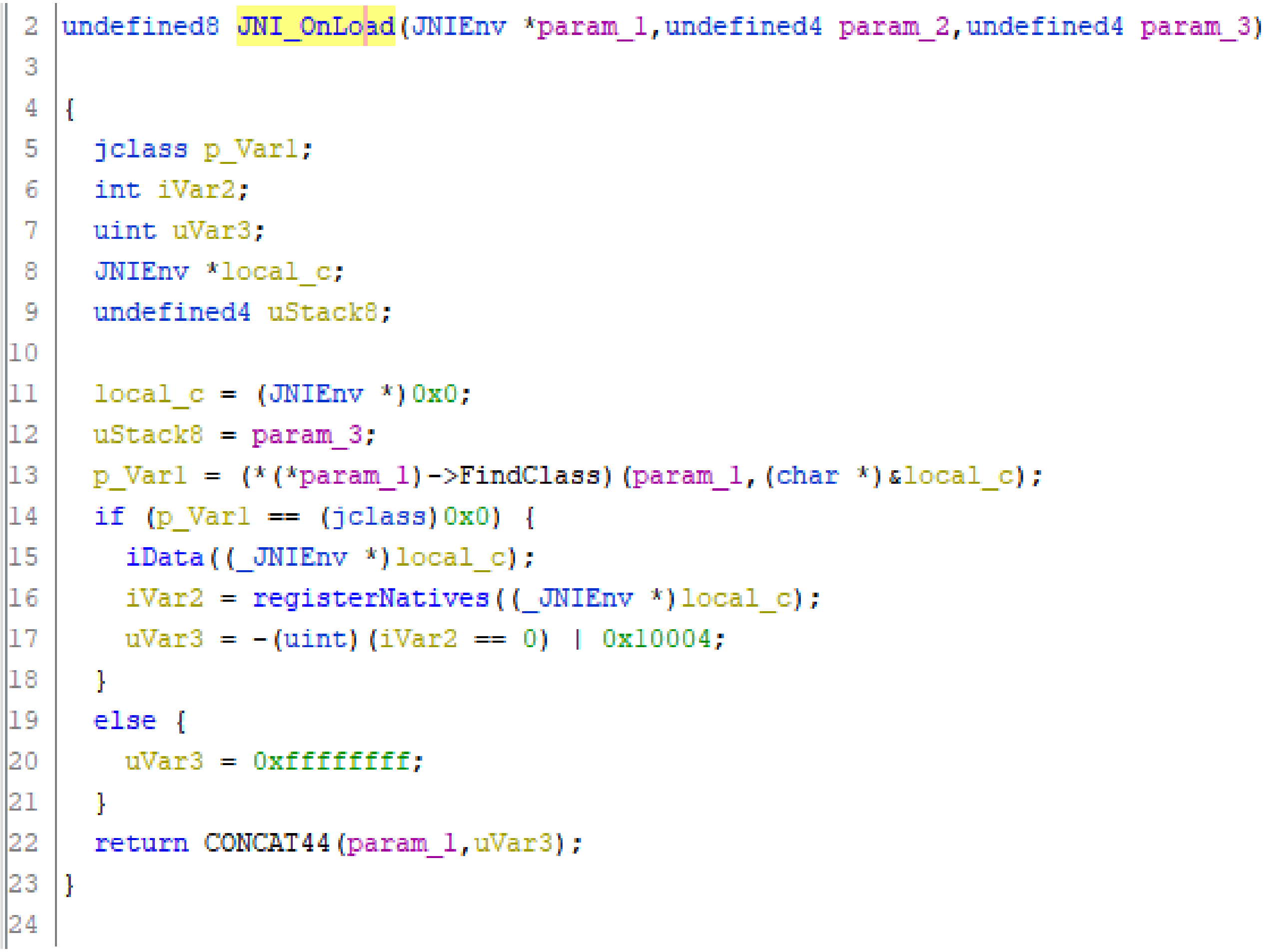


```
undefined8 JNI_OnLoad(int *param_1,undefined4 param_2,undefined4 param_3)

{
  int iVar1;
  uint uVar2;
  _JNIEnv *local_c;
  undefined4 uStack8;

  local_c = (_JNIEnv *)0x0;
  uStack8 = param_3;
  iVar1 = (**(code **)(*param_1 + 0x18))(param_1,&local_c,0x10004);
  if (iVar1 == 0) {
    iData(local_c);
    iVar1 = registerNatives(local_c);
    uVar2 = -(uint)(iVar1 == 0) | 0x10004;
  }
  else {
    uVar2 = 0xffffffff;
  }
  return CONCAT44(param_1,uVar2);
}

```

Call registerNatives

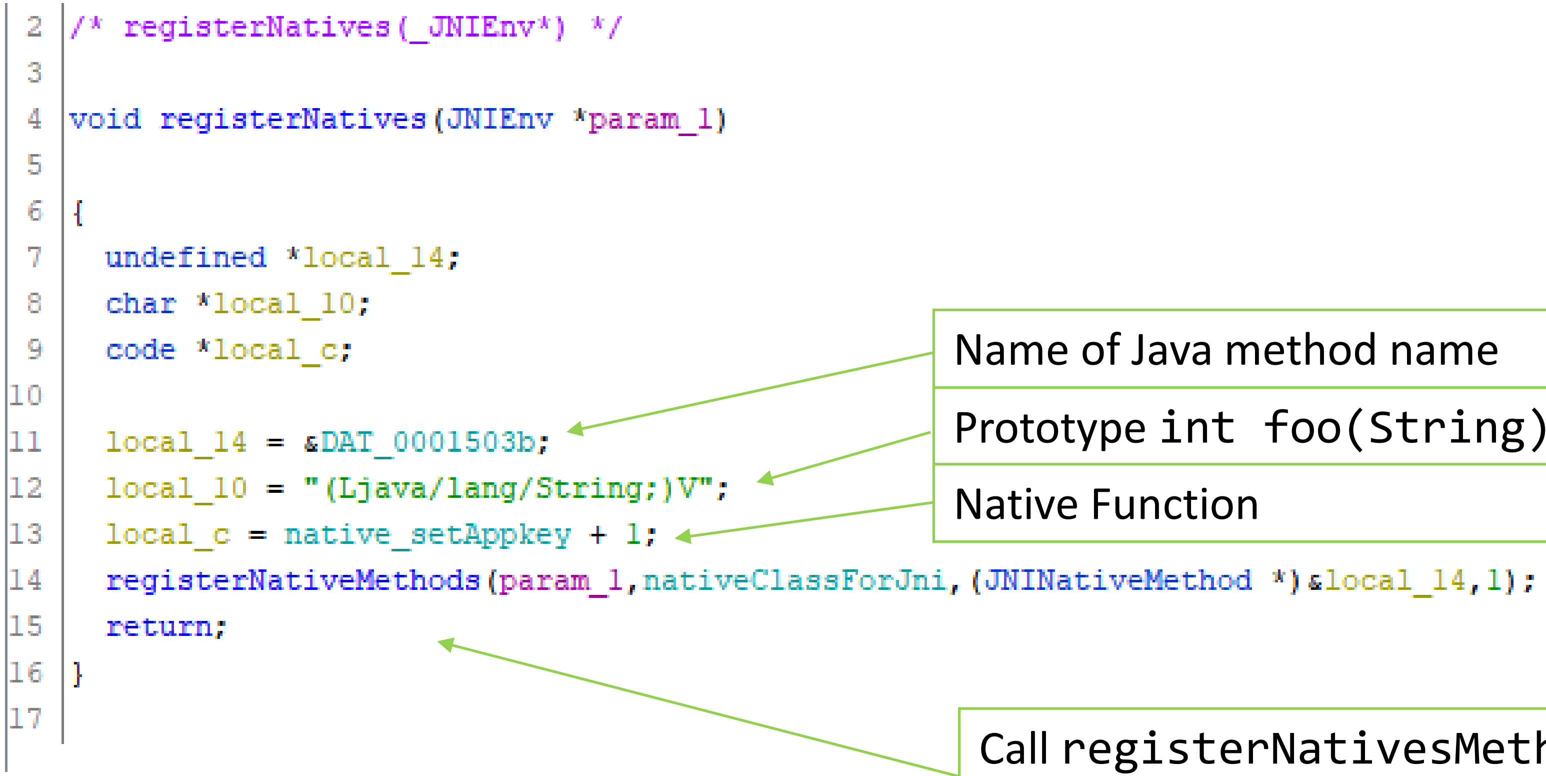
JNI_OnLoad: ghidra with a JNI GDT and retyping

- Loading the `jni_all.gdt`, and retyping the variables, allows resolution of symbols, such as the `FindClass`.



```
undefined8 JNI_OnLoad(JNIEnv *param_1,undefined4 param_2,undefined4 param_3)

{
  jclass p_Var1;
  int iVar2;
  uint uVar3;
  JNIEnv *local_c;
  undefined4 uStack8;

  local_c = (JNIEnv *)0x0;
  uStack8 = param_3;
  p_Var1 = (*(*param_1)->FindClass)(param_1,(char *)&local_c);
  if (p_Var1 == (jclass)0x0) {
    iData((_JNIEnv *)local_c);
    iVar2 = registerNatives((_JNIEnv *)local_c);
    uVar3 = -(uint)(iVar2 == 0) | 0x10004;
  }
  else {
    uVar3 = 0xffffffff;
  }
  return CONCAT44(param_1,uVar3);
}
```

registerNatives



```
/* registerNatives(_JNIEnv*) */

void registerNatives(JNIEnv *param_1)

{
  undefined *local_14;
  char *local_10;
  code *local_c;

  local_14 = &DAT_0001503b;
  local_10 = "(Ljava/lang/String;)V";
  local_c = native_setAppkey + 1;
  registerNativeMethods(param_1,nativeClassForJni,(JNINativeMethod *)&local_14,1);
  return;
}
```

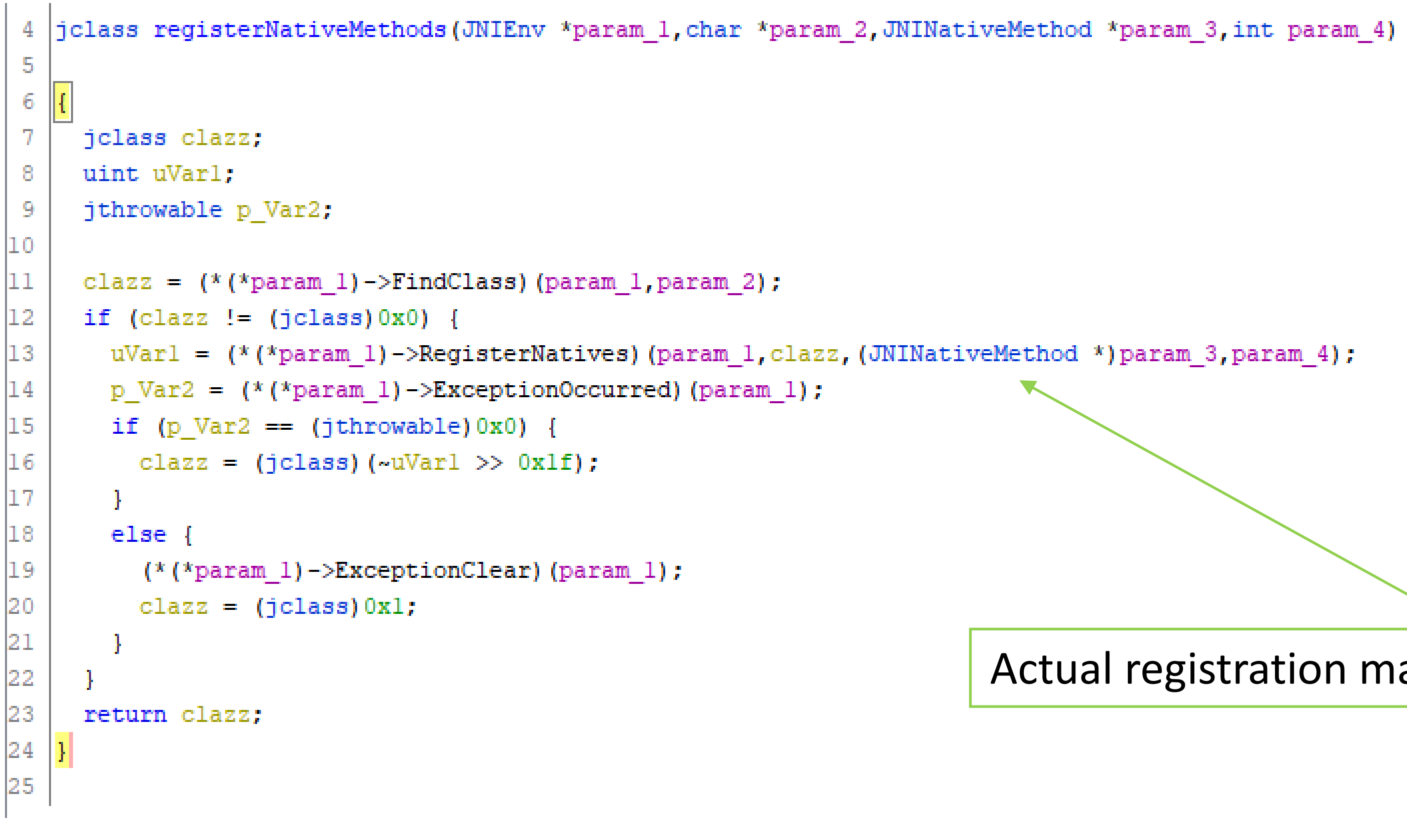
Name of Java method name

Prototype `int foo(String)`

Native Function

Call `registerNativesMethods`
Is this the actual register method?

registerNativeMethods



```
jclass registerNativeMethods(JNIEnv *param_1,char *param_2,JNINativeMethod *param_3,int param_4)

{
  jclass clazz;
  uint uVar1;
  jthrowable p_Var2;

  clazz = (*(*param_1)->FindClass)(param_1,param_2);
  if (clazz != (jclass)0x0) {
    uVar1 = (*(*param_1)->RegisterNatives)(param_1,clazz,(JNINativeMethod *)param_3,param_4);
    p_Var2 = (*(*param_1)->ExceptionOccurred)(param_1);
    if (p_Var2 == (jthrowable)0x0) {
      clazz = (jclass)(~uVar1 >> 0x1f);
    }
    else {
      (*(*param_1)->ExceptionClear)(param_1);
      clazz = (jclass)0x1;
    }
  }
  return clazz;
}
```

Actual registration made through JNIEnv method

Web and Hybrid applications

Why not Native apps?

- Native Apps are not that good (or not always that good)
  - Have low Code Reusability
  - Require more development and maintenance
  - Requires designers and developers' experts on multiple architectures
  - Have low upgrade flexibility

- Once was the traditional way of developing applications
  - Currently being surpassed by web and hybrid applications

- From a RE perspective, the toolset and languages are very different
  - More complex to analyze
  - Better commonly available obfuscators

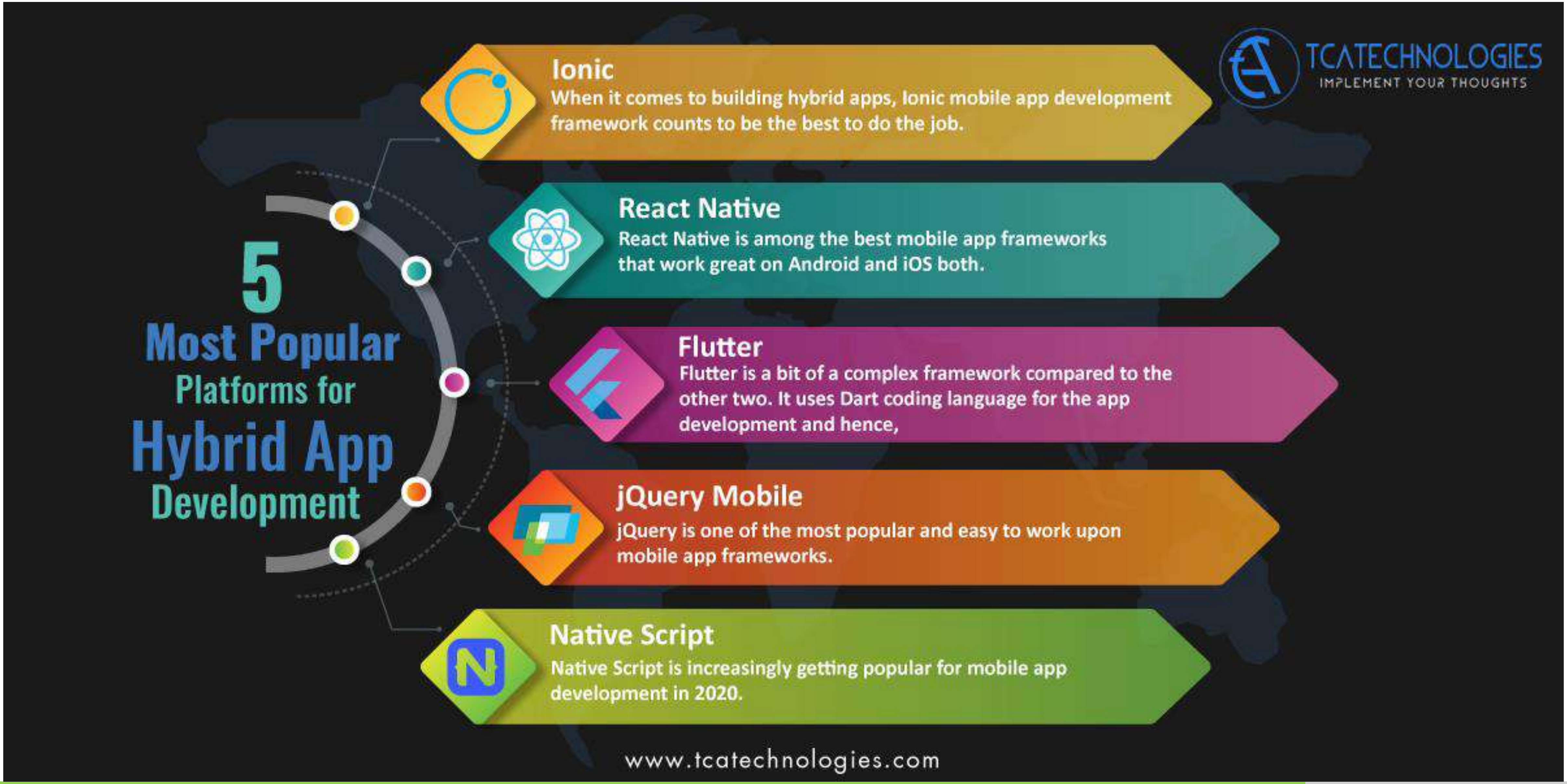
Web apps

- Use standard web technologies (HTML, CSS, Javascript)
- Especially since HTML5 allowed:
  - Advanced UI components
  - Access to media types and sources
  - Access to geolocation
  - Access to local storage
- Look like a standard application (present an Icon)
- Completely different stack
  - Standalone Mobile Web Browser

Hybrid apps

- Combine both worlds: Native and Web
  - a thin Java application with a Web application
- Most commonly:
  - Web for the interface
  - Java for the application backend
  - Custom Interface connecting both levels
- Installable from the store and indistinguishable from native apps
  - As devices are more powerful, these are becoming very common

Typical frameworks



5 Most Popular Platforms for Hybrid App Development

TCATECHNOLOGIES
IMPLEMENT YOUR THOUGHTS

Ionic
When it comes to building hybrid apps, Ionic mobile app development framework counts to be the best to do the job.

React Native
React Native is among the best mobile app frameworks that work great on Android and iOS both.

Flutter
Flutter is a bit of a complex framework compared to the other two. It uses Dart coding language for the app development and hence,

jQuery Mobile
jQuery is one of the most popular and easy to work upon mobile app frameworks.

Native Script
Native Script is increasingly getting popular for mobile app development in 2020.

www.tcatechnologies.com

RE Perspective

- Most frameworks use JS, but sometimes with custom VMs

- Packaging consists of adding the application JS code, HTML, styles and remaining resources
  - May use a bundle, including all resources
  - May leave resources bare in the APK
  - May use binary libs with obfuscated code, but frequently they are just plugins for native functions

- Code is frequently obfuscated
  - An inheritance of the JS obfuscators available

- Code may be compiled to an intermediate representation
  - Decompilers are not that robust as the ones for Java

- RE support is lacking…

IONIC

- Runs on the Apache Cordova infrastructure
  - Framework implemented in Java
  - Application presented through a Web View

- Actual application is a webpage in the assets/www directory
  - Cordova Plugins in www/plugins
    - Implemented in JS, communicating with main framework through interface

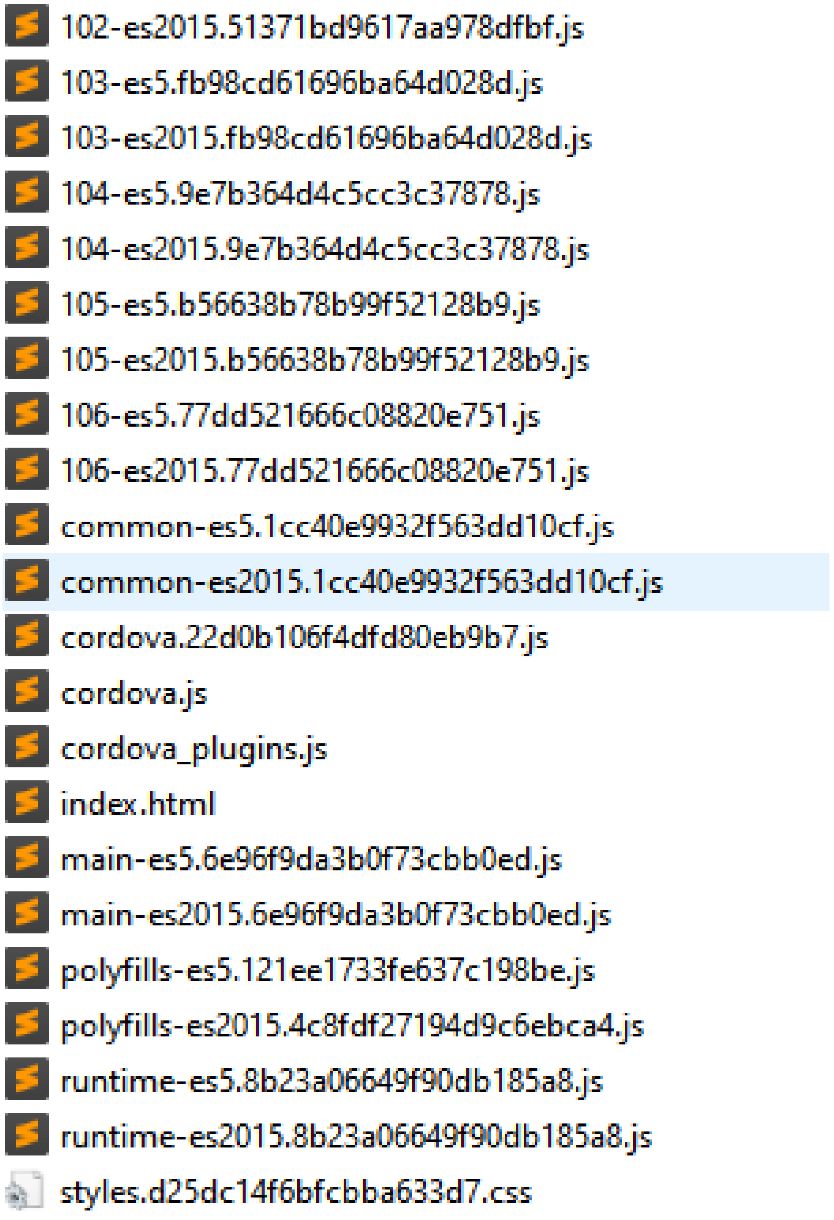
- Framework is event driver with actions activated on interactions

IONIC

- Every file contains a single line
  - Minified code
  - Pushing logic, or a handler

- Index.html as the entry point

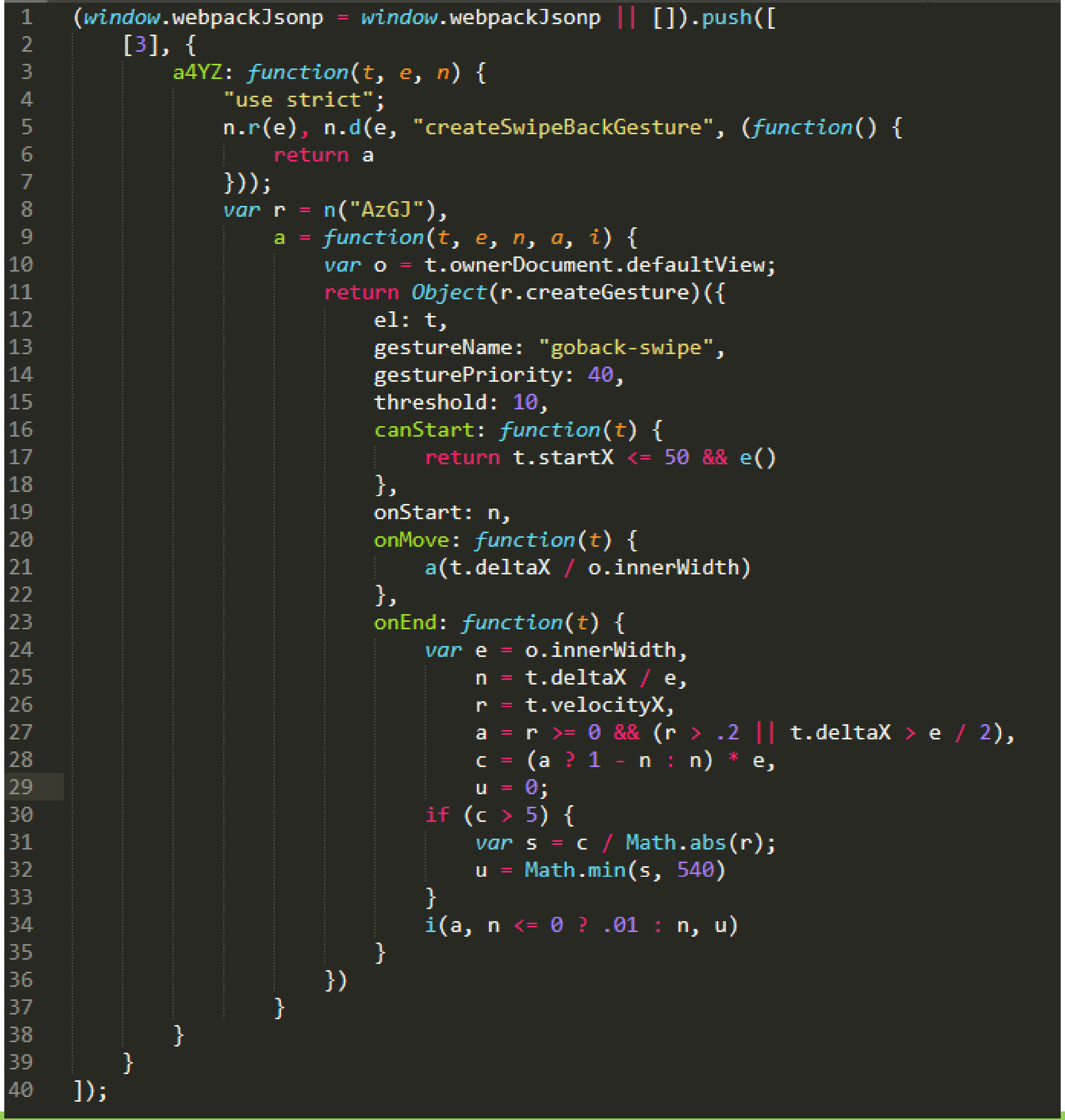
- Workflow:
  - Beautify code and extract information
  - Launching Cordova on local PC
  - Inspection through browser
  - Dynamic Analysis



```
102-es2015.51371bd9617aa978dfbf.js
103-es5.fb98cd61696ba64d028d.js
103-es2015.fb98cd61696ba64d028d.js
104-es5.9e7b364d4c5cc3c37878.js
104-es2015.9e7b364d4c5cc3c37878.js
105-es5.b56638b78b99f52128b9.js
105-es2015.b56638b78b99f52128b9.js
106-es5.77dd521666c08820e751.js
106-es2015.77dd521666c08820e751.js
common-es5.1cc40e9932f563dd10cf.js
common-es2015.1cc40e9932f563dd10cf.js
cordova.22d0b106f4dfd80eb9b7.js
cordova.js
cordova_plugins.js
index.html
main-es5.6e96f9da3b0f73cbb0ed.js
main-es2015.6e96f9da3b0f73cbb0ed.js
polyfills-es5.121ee1733fe637c198be.js
polyfills-es2015.4c8fdf27194d9c6ebca4.js
runtime-es5.8b23a06649f90db185a8.js
runtime-es2015.8b23a06649f90db185a8.js
styles.d25dc14f6bfcbba633d7.css
```

App ionfits.apk

IONIC



```
(window.webpackJsonp = window.webpackJsonp || []).push([
    [3], {
        a4YZ: function(t, e, n) {
            "use strict";
            n.r(e), n.d(e, "createSwipeBackGesture", (function() {
                return a
            }));
            var r = n("AzGJ"),
                a = function(t, e, n, a, i) {
                    var o = t.ownerDocument.defaultView;
                    return Object(r.createGesture)({
                        el: t,
                        gestureName: "goback-swipe",
                        gesturePriority: 40,
                        threshold: 10,
                        canStart: function(t) {
                            return t.startX <= 50 && e()
                        },
                        onStart: n,
                        onMove: function(t) {
                            a(t.deltaX / o.innerWidth)
                        },
                        onEnd: function(t) {
                            var e = o.innerWidth,
                                n = t.deltaX / e,
                                r = t.velocityX,
                                a = r >= 0 && (r > .2 || t.deltaX > e / 2),
                                c = (a ? 1 - n : n) * e,
                                u = 0;
                            if (c > 5) {
                                var s = c / Math.abs(r);
                                u = Math.min(s, 540)
                            }
                            i(a, n <= 0 ? .01 : n, u)
                        }
                    })
                }
        }
    }
]);
```

App ionfits.apk

Flutter

- UI from Google based on the Dart Language
  - Compiled under the scope of the Dart VM (https://github.com/dart-lang/sdk)
  - Designed as a dual purpose framework: Web and Mobile
  - With Native and Web components
    - In mobile devices, Flutter is compiled to a native object (libapp.so)
  - As good reference, check: https://mrale.ph/dartvm/

- Two deployment flavors
  - AOT: Ahead of Time – the most frequent – as a bytecode for the Dart VM
  - JIT: Just in Time – for debug builds, interpreted from Source Code

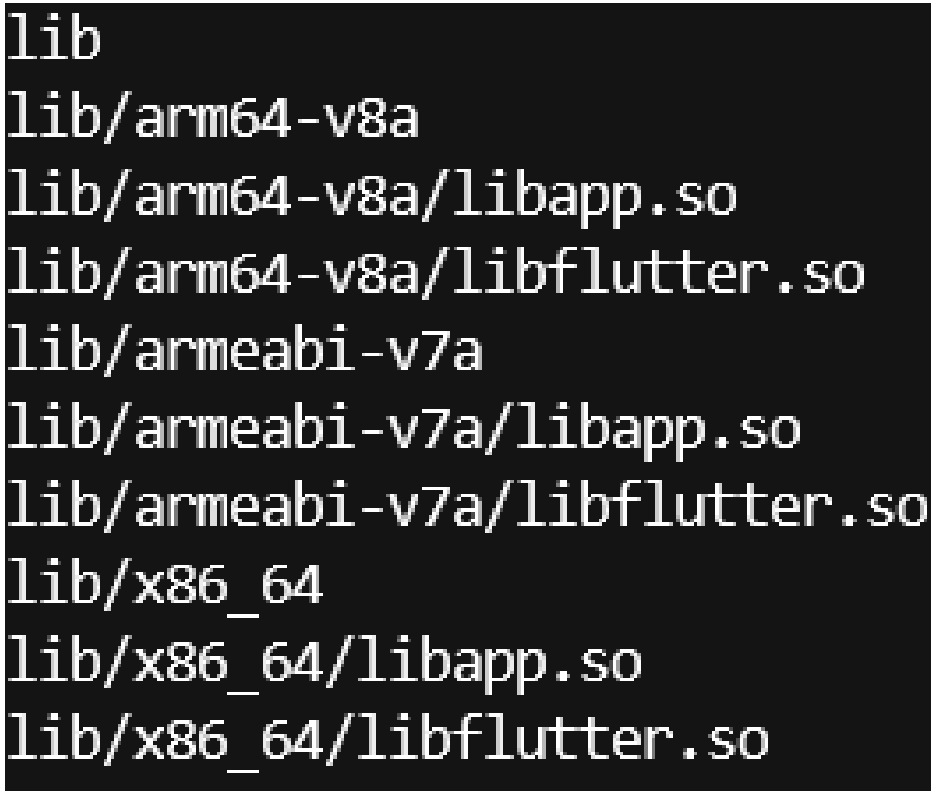
- Project structure:
  - Small java shim to load the actual code
  - Framework in libfutter.so
  - Application in another .so libapp.so (yes, an ELF!)
    - Actually, it contains a snapshot of the VM to be loaded

From a RE perspective

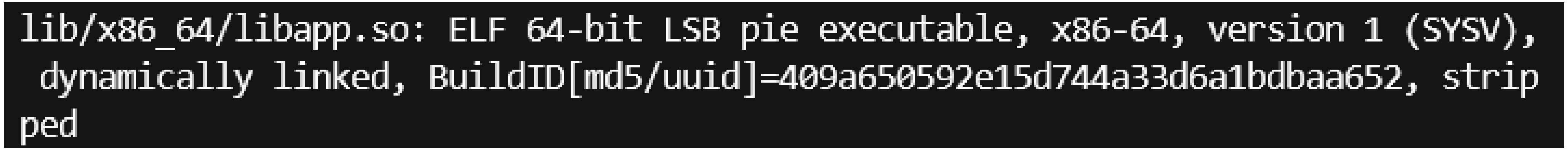
- Flutter compiles Dart to native assembly in a single bundle
  - Internal formats are not publicly known in detail
- By default there is no obfuscation or encryption
  - However the formats are not known
- Flutter applications are difficult to reverse engineer
  - Good for intellectual property
- Some tools start to scratch the surface (mostly extract information from libapp.so), extracting information
  - https://github.com/mildsunrise/darter
  - https://github.com/rscloura/Doldrums

Flutter: Flutter-Weather

- Simple application showing weather info
  - https://github.com/1hanzla100/flutter-weather
- Follows typical structure
  - 2 .so: Framework and App for multiple archs
- Current tools extract classes from VM snapshot, but there is little similarity with original code



```
lib
lib/arm64-v8a
lib/arm64-v8a/libapp.so
lib/arm64-v8a/libflutter.so
lib/armeabi-v7a
lib/armeabi-v7a/libapp.so
lib/armeabi-v7a/libflutter.so
lib/x86_64
lib/x86_64/libapp.so
lib/x86_64/libflutter.so
```



```
lib/x86_64/libapp.so: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV),
 dynamically linked, BuildID[md5/uuid]=409a650592e15d744a33d6a1bdbaa652, strip
ped
```

Android – Dynamic Analysis

REVERSE ENGINEERING

deti universidade de aveiro
departamento de eletrónica,
telecomunicações e informática

João Paulo Barraca

Dynamic Analysis

Static Analysis: Open the application and deduct how it works

Researcher must deduct the Data Flow

External Data or Actions may change the application behavior

- Change the code path
- Inject instructions

Issues may be found on the sequence of events, or on the state machine

Dynamic Analysis: Observe the application while it is running, allowing to obtain information about the dynamic characteristics.

Dynamic Analysis

- **Look into specific aspects of an application, while it is executing**

- **Objective**: Observe dynamic behavior of the application and determine the role of each code

- **What can be analyzed**
  - Messages exchanged with external servers (REST APIs, Web Sockets)
  - Intents sent or received
  - Logs printed (errors, debug messages)
  - Files accessed/created
  - Memory Content
  - With code instrumentation: calls to methods, especially Android API methods

Logs

- Android log can be used to dynamically analyze relevant aspects of application execution
  - Explicit log entry produced by the application or by system components
  - Implicit logs produced with errors
    - Exceptions produce stack traces which expose call flow
  - Some system events
  - May be used to detect leaks

```
$ adb logcat
```

Network MiTM

- **Interactions with external APIs can be intercepted and analyzed**
  - Useful to identify communication with domains with low reputation
  - Useful to identify unprotected communications
    - Especially dangerous if dealing with authentication, private data or download of dynamic components

- **Black box approach**: observe how the app behaves
  - We may simply observe
  - … or we manipulate/filter traffic

Network MiTM

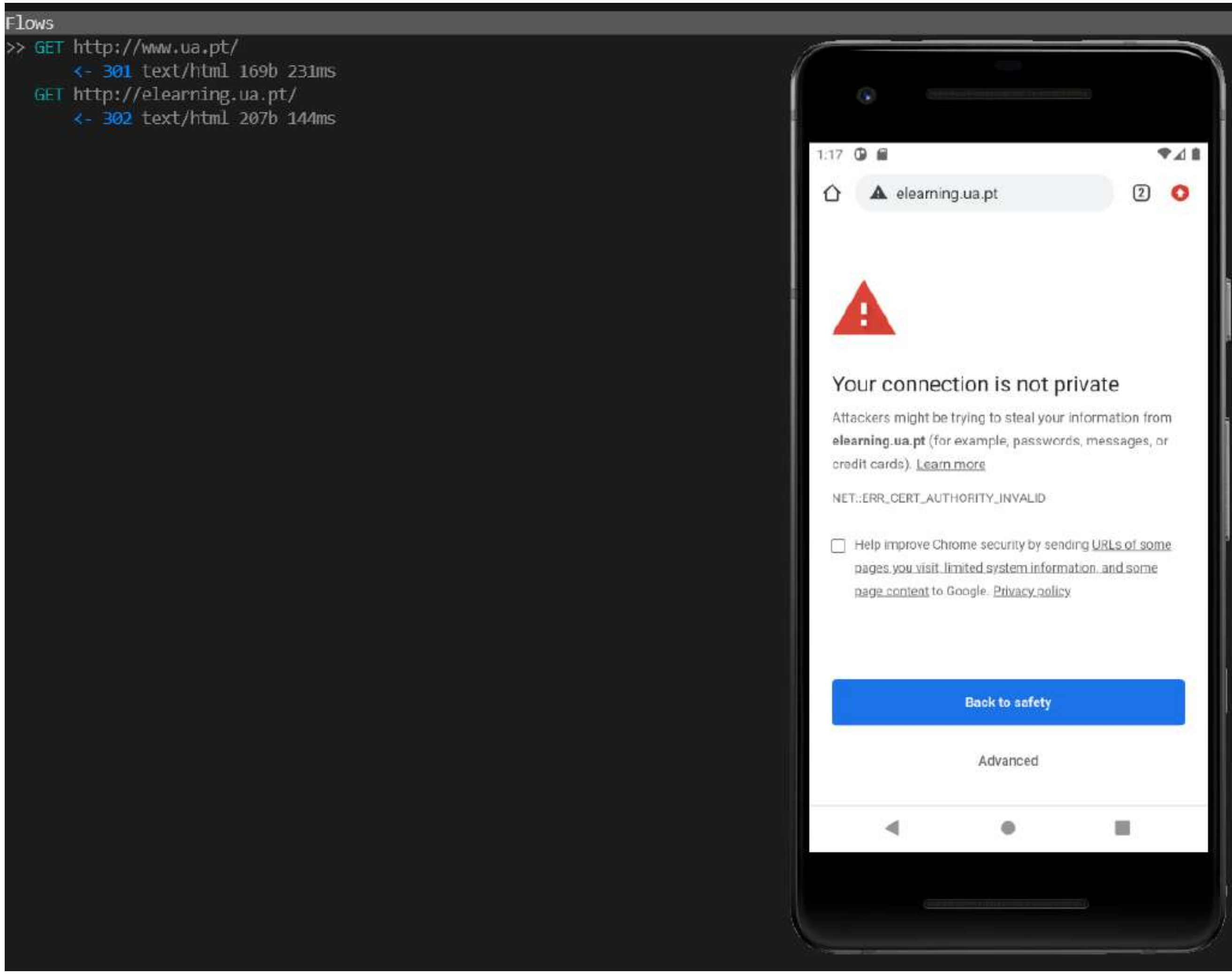
- **Packet Dumps**
  - run applications and capture traffic with a packet sniffer
  - Non encrypted APIs can be analyzed with ease
    - The endpoint IP address may constitute an indicator by itself
      - Communication with flagged domains, validation that a service is invoked
  - Using `wireshark (androiddump)`

- **Traffic flows**
  - run applications with a HTTP/HTTPS proxy configured to intercept all traffic
  - injecting a CA Certificate in the device allows generating custom certificates for secure endpoints

Traffic Flows

- Using an HTTP proxy with Active TLS interception capability
  - Proxy will generate certificates for all hosts accessed
  - Certificates are signed by a single CA
  - CA must be installed to the device

Using mitmproxy, without CA installed
Alternatives: Charles, ZAP, Burp



```
Flows
>> GET http://www.ua.pt/
       <- 301 text/html 169b 231ms
   GET http://elearning.ua.pt/
       <- 302 text/html 207b 144ms
```

1:17

elearning.ua.pt

Your connection is not private

Attackers might be trying to steal your information from **elearning.ua.pt** (for example, passwords, messages, or credit cards). Learn more

NET::ERR_CERT_AUTHORITY_INVALID

□ Help improve Chrome security by sending URLs of some pages you visit, limited system information, and some page content to Google. Privacy policy

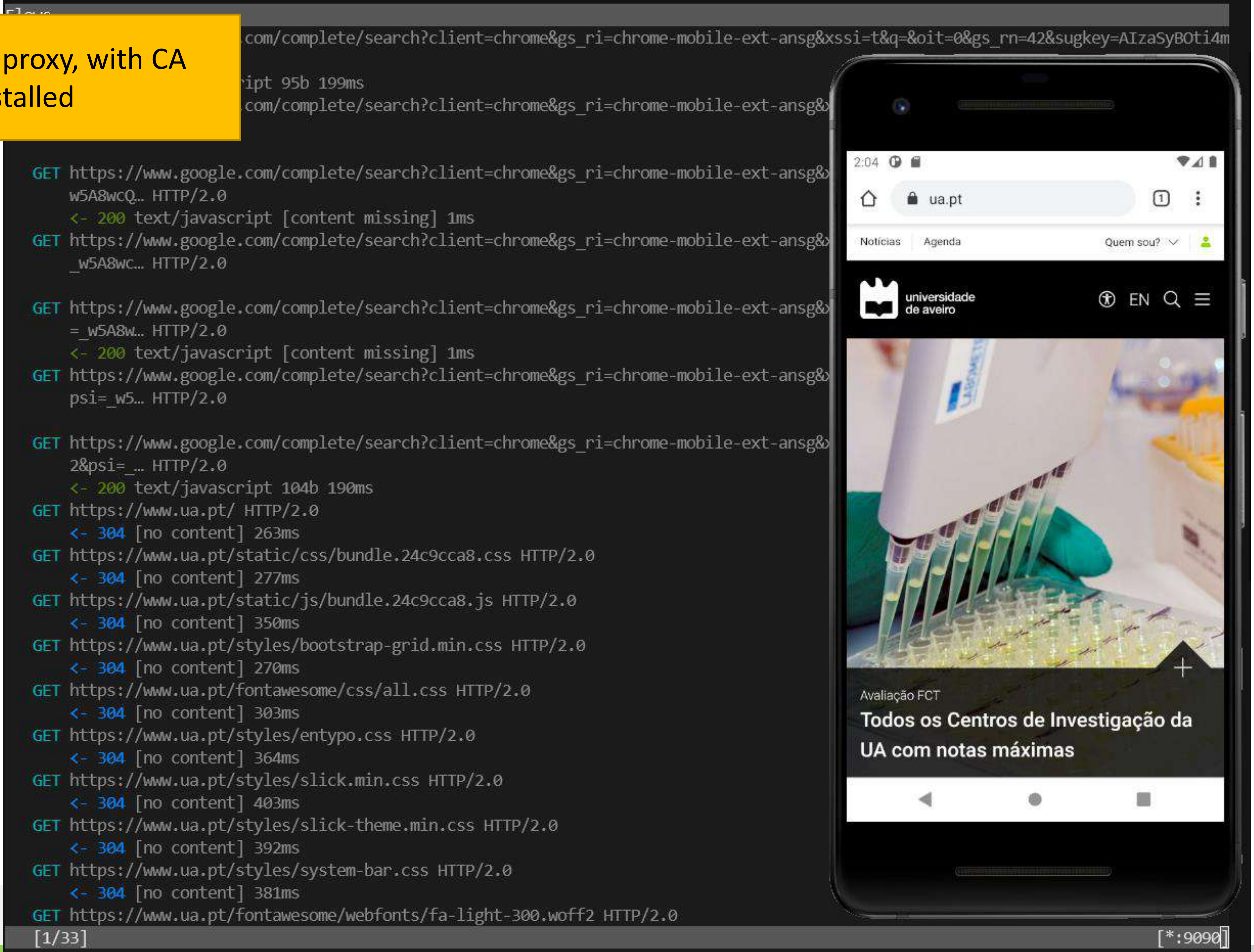
Trusted Certificates

- Standard X509 certificates in PEM format
  - Preinstalled by the manufacturer
  - Cannot be changed by users
  - Users can add custom certificates, but they are frequently ignored by the applications

- On an Android system, trusted roots are at `/system/etc/security/cacerts`
  - Folder with PEM certificates

- /system partition is read only on release devices
  - In recent versions of Android the same is also true for the emulator
  - Alternative: mount a tmpfs at the certificate location
    - But changes are lost on reboot

Using mitmproxy, with CA installed



```
Flows
         .com/complete/search?client=chrome&gs_ri=chrome-mobile-ext-ansg&xssi=t&q=&oit=0&gs_rn=42&sugkey=AIzaSyBOti4m

         ipt 95b 199ms
         .com/complete/search?client=chrome&gs_ri=chrome-mobile-ext-ansg&

GET https://www.google.com/complete/search?client=chrome&gs_ri=chrome-mobile-ext-ansg&
    w5A8wcQ… HTTP/2.0
    <- 200 text/javascript [content missing] 1ms
GET https://www.google.com/complete/search?client=chrome&gs_ri=chrome-mobile-ext-ansg&
    _w5A8wc… HTTP/2.0

GET https://www.google.com/complete/search?client=chrome&gs_ri=chrome-mobile-ext-ansg&
    =_w5A8w… HTTP/2.0
    <- 200 text/javascript [content missing] 1ms
GET https://www.google.com/complete/search?client=chrome&gs_ri=chrome-mobile-ext-ansg&
    psi=_w5… HTTP/2.0

GET https://www.google.com/complete/search?client=chrome&gs_ri=chrome-mobile-ext-ansg&
    2&psi=_… HTTP/2.0
    <- 200 text/javascript 104b 190ms
GET https://www.ua.pt/ HTTP/2.0
    <- 304 [no content] 263ms
GET https://www.ua.pt/static/css/bundle.24c9cca8.css HTTP/2.0
    <- 304 [no content] 277ms
GET https://www.ua.pt/static/js/bundle.24c9cca8.js HTTP/2.0
    <- 304 [no content] 350ms
GET https://www.ua.pt/styles/bootstrap-grid.min.css HTTP/2.0
    <- 304 [no content] 270ms
GET https://www.ua.pt/fontawesome/css/all.css HTTP/2.0
    <- 304 [no content] 303ms
GET https://www.ua.pt/styles/entypo.css HTTP/2.0
    <- 304 [no content] 364ms
GET https://www.ua.pt/styles/slick.min.css HTTP/2.0
    <- 304 [no content] 403ms
GET https://www.ua.pt/styles/slick-theme.min.css HTTP/2.0
    <- 304 [no content] 392ms
GET https://www.ua.pt/styles/system-bar.css HTTP/2.0
    <- 304 [no content] 381ms
GET https://www.ua.pt/fontawesome/webfonts/fa-light-300.woff2 HTTP/2.0
[1/33]                                                                    [*:9090]
```

Network MITM - Limitations

- Packet dumps are limited to unprotected text and metadata
  - Again… it is still relevant as it may produce a valid indicator

- Traffic flow analysis is limited to devices where a CA can be injected
  - And where the APP will not use custom CA Certificates
  - And where the APP will not use Certificate Pinning

Certificate Pinning

- **Applications put constraints on the certificates used for verification (Trusted Roots)**
  - They fix (Pin) a certificate/pub key/hash to a hostname
  - Validation of the host authenticity (in TLS) will also include this additional constraints

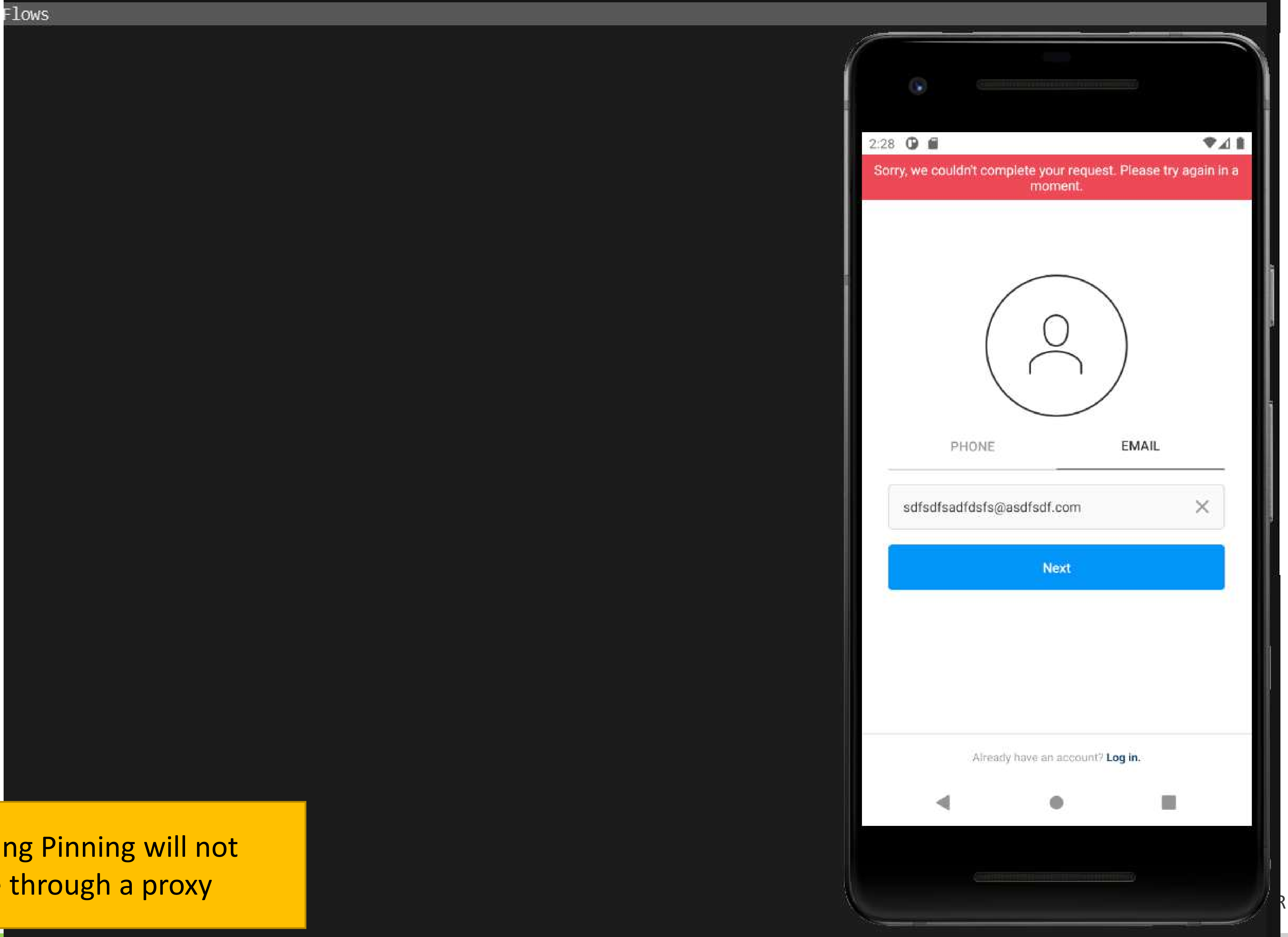
- **Impact**
  - A Trusted Root can be injected but it will be ignored
    - Application will simply not use it
    - Or the application will have additional checks with detect the injection

Certificate Pinning - Approaches

- Applications extend the X509TrustManager, overriding the checkServerTrusted method, with custom checks
  - E.g. the Certificate/Public Key/hash is hard coded, and this value is used to validate the certificate

- Using a KeyStore with a predefined list of certificates, ignoring other sources
  - Pins the host certificate
  - Pins an intermediate Certification Authority
  - Pins a Root Certification Authority

- Pinning may create issues for developers as changes to certificates or PKI must be reflected to the applications
  - Soft Fail: just let the application work, even if with limited functionality
  - Hard Fail: an update is forced for the application to work



Flows

2:28

Sorry, we couldn't complete your request. Please try again in a moment.

sdfsdfsadfdsfs@asdfsdf.com

Already have an account? Log in.

Applications using Pinning will not communicate through a proxy

Certificate Pinning - Circumvention

- **If restricted KeyStores are used:** use an emulator or rooted device
  - Enables free manipulation of the keystores, injecting custom certificates
  - Inject certificates to the system keystores

- **If Pinned with hard coded information:** modify the application
  - Unpack the application
  - Edit the code, changing the Pin or removing it
    - `smali` may be enough and full decompilation to java is not required
  - Repack and install the application

Dynamic Code Instrumentation

- Applications are implemented with functions
  - Functions have addresses that may be determined
    - Or subverted
  - Java uses further abstractions when using native code
    - Creating strict interaction points towards which is possible to access the external world

- This enables the possibility of manipulating symbols/addresses to instrumentalize code
  - Observe internal structure of the program flow
  - Inject new code by replacing the implementation of a function represented by a symbol

Dynamic Binary Instrumentation - Why

- Requests to APIs are further encrypted, signed or MITM is not available
  - MITM and packet sniffers are useless

- Application has obfuscated values in RAM, created dynamically, received from the network
  - Static analysis and Decompilation is useless

- Code is loaded dynamically with objects received
  - Static analysis and Decompilation will have no code to analyze

- Many values are hard coded (keys, urls…)
  - Patching takes too long and becomes expensive

Custom Signatures are used

```
POST /login HTTP/1.1
Host: social.io
Proxy-Connection: keep-alive
Content-Length: X
Accept: text/html, application/xhtml-
xml,application/xml;q=0.9,image/webp,*/*,q=0.8
Origin: http://social.io
Content-Type: application/x-www-form-urlencoded
Cookie: SessionId=O+qxnaYZLjpnLwHBcKmRcTexTWk=

username=john&password=xpto&signature=2rf+roJPEdCOSL0XXusHBcA0BGk=
```

Data is encrypted

```
POST /login HTTP/1.1
Host: social.io
Proxy-Connection: keep-alive
Content-Length: X
Accept: text/html, application/xhtml-
xml,application/xml;q=0.9,image/webp,*/*,q=0.8
Origin: http://social.io
Content-Type: application/x-www-form-urlencoded
Cookie: SessionId=O+qxnaYZLjpnLwHBcKmRcTexTWk=

authData=3NH71S+7P8YeafgnBvXzJ1RzJdXm51VNPQYMWFiIMl8ZNr7+vGDNTcms8LHDUaC/lK2xRF/L
bPMwQ0pB+ZyB6PfYNaf5fIh/IGdlQZJrgXXgDDT7Mn2d259vzcdmBA3pJ04cLxGNnLSvdorYF+mLN7yik
zEagUWGfQe1nYzu3OT3947kqSORQuc4PTzuFKUXlolCcuVYvr5gt6ykfk9ACGVwyywGBG3OeFxNKi0kme
iBYxB8EJlmCF/xojM59gcGDv61ytidhVs=
```

Many others

- Retrieving a call flow
  - Map which methods are used, and what is the actual code execution flow

- Identify arguments of Android API methods
  - Log traffic and calls
  - Allows intercepting data even with encrypted connections
    - Interception happens before data is encrypted

- Modify arguments of Android API methods
  - Fuzzing
  - Filter/modify data to trigger additional behavior
  - Trigger custom events

- Circumvent protections to enable further analysis

- Application is obfuscated and it is difficult to obtain the actual algorithm

FRIDA – How?

- Set of tools (framework) for Dynamic Code Instrumentation
  - Instruments the Application Code with hooks
  - Not specific for Android, and may be used on other applications and Operating Systems

- Allows:
  - Tracing network communications at the method level
  - Understand how the application behaves
  - Manipulate the methods called, arguments and return codes
  - …

FRIDA – How?

- Frida-core injecting a Google V8 JS Engine into the App scope
  - Frida-core Written in C
  - GumJS (the JS part) is packed as a shared library and loaded into the app

- GumJS has access to the application memory
  - Can be hooked to methods and intercept calls, even native APIs

- GumJS API allow interaction with GumJS from an external client
  - Because GumJS resides the application address space, it has full access to its structures

FRIDA – mode Embedded

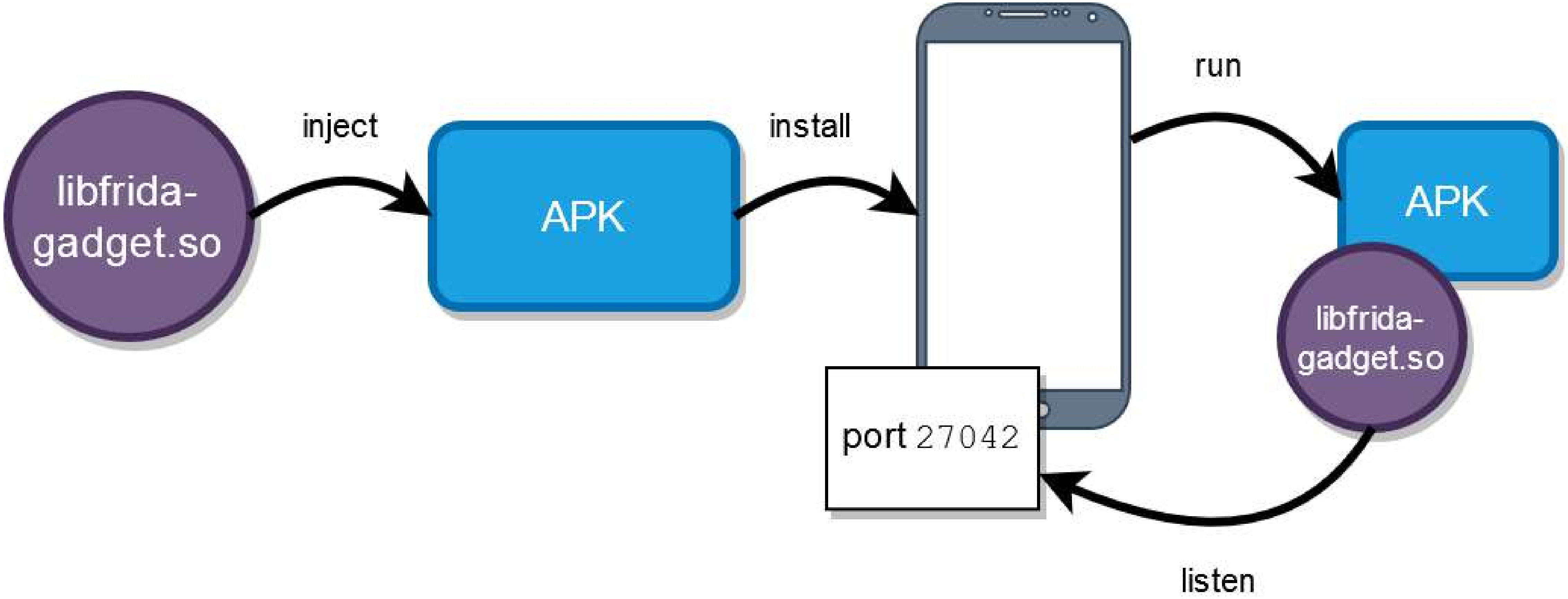
- Frida agent is embedded as a dependency of an existing application library
  - Requires the application to have an .so
  - `libfrida-gadget.so` is injected into the existing library and loaded at the same time

- Alternative: existing library is modified in order to load the additional library
  - Requires patching the code in a function will be called (e.g. `JNI_OnLoad`)

- Alternative: patching the `smali` code to load the library
  - Obtain APK
  - Extract smali
  - Change smali
  - Pack it and install

- Method implies that the application is repacked/resigned

FRIDA – mode Embedded



FRIDA – mode Embedded – How?

```
$ apktool d app.apk
$ cp libfrida-gadget.so target/lib/arm
$ python3
>>> import lief
>>> native = lief.parse(“target/lib/arm/libsomething.so”)
>>> native.add_library(“libfrida-gadget.so”)
>>> native.write(“target/lib/arm/libsomething.so”)
>>> exit

$ apktool b target
… sign … install
```

FRIDA – mode Embedded – Smali - How

- Unpack the app using `apktool`

- Patch the `smali` with

```
const-string v0, "frida-gadget"
invoke-static {v0}, Ljava/lang/System;->loadLibrary(Ljava/lang/String;)V
```

- Where? In the main activity constructor
  - Even in as a static property of the class

- repack, sign, install

FRIDA – mode Embedded - Caveats

- Applications may search for the library name as an anti-debug technique
  - May need to change the library name
- Must use a version compatible with the target architecture
  - https://github.com/frida/frida/releases
- Agent may only be loaded after the JNI library is loaded or code is reached
  - After `System.loadLibrary(“lib.so”)`
- Agent may impose the need for permissions to access the INTERNET
  - Manifest may need to be updated

FRIDA – mode Injected

- Run a Frida Server which injects the agent into the target process
  - Server provides an API for remote use
  - Server injects the agents into applications

- Requires the smartphone to be rooted or to be an emulator
  - In order to run the server and inject applications

- Cannot be used in production builds, only development
  - When in an Emulator, use a base without WITHOUT “Google Services”

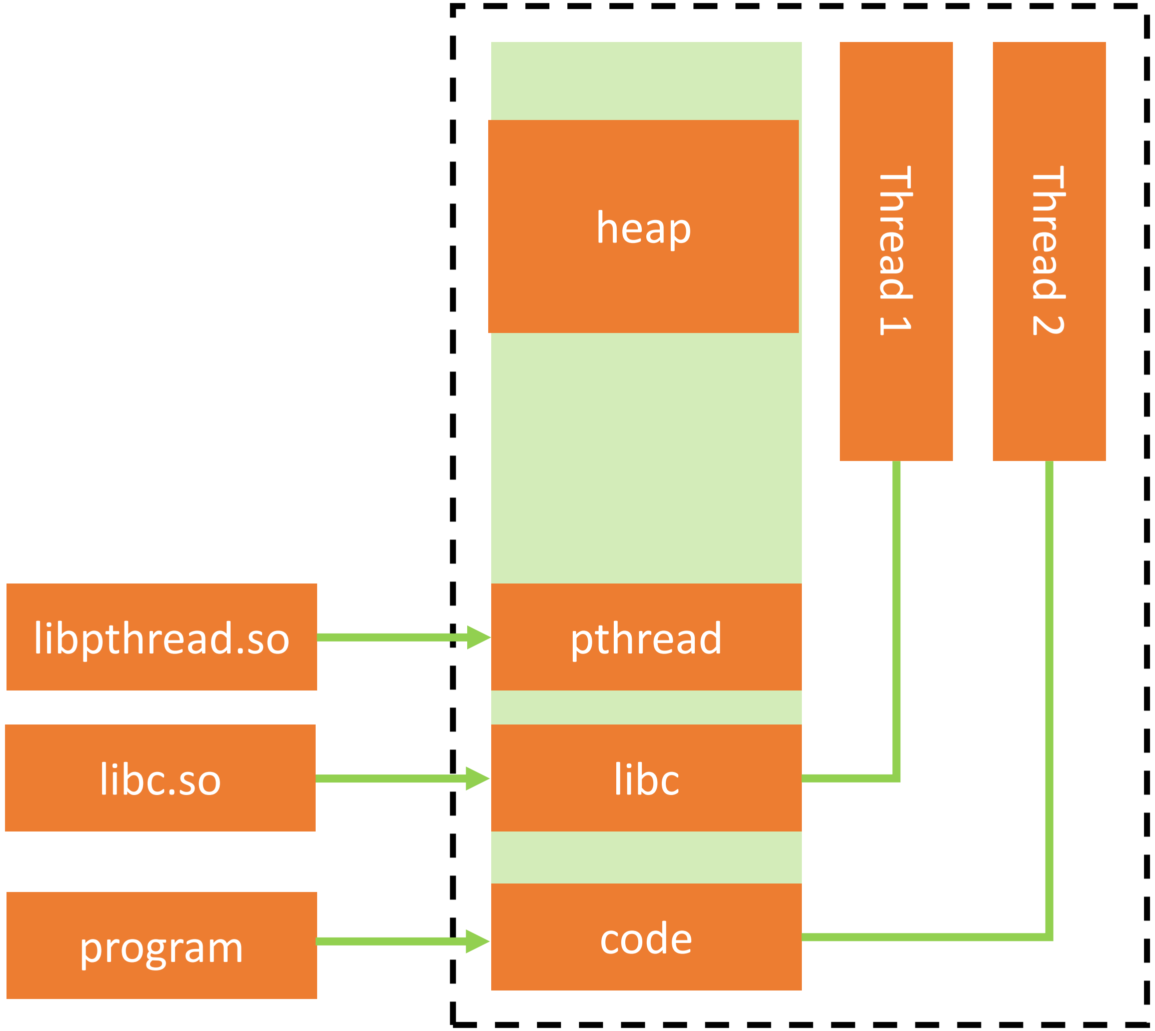
FRIDA – mode Injected – How it works?

- Create an Agent: it’s an `.so` with some custom code

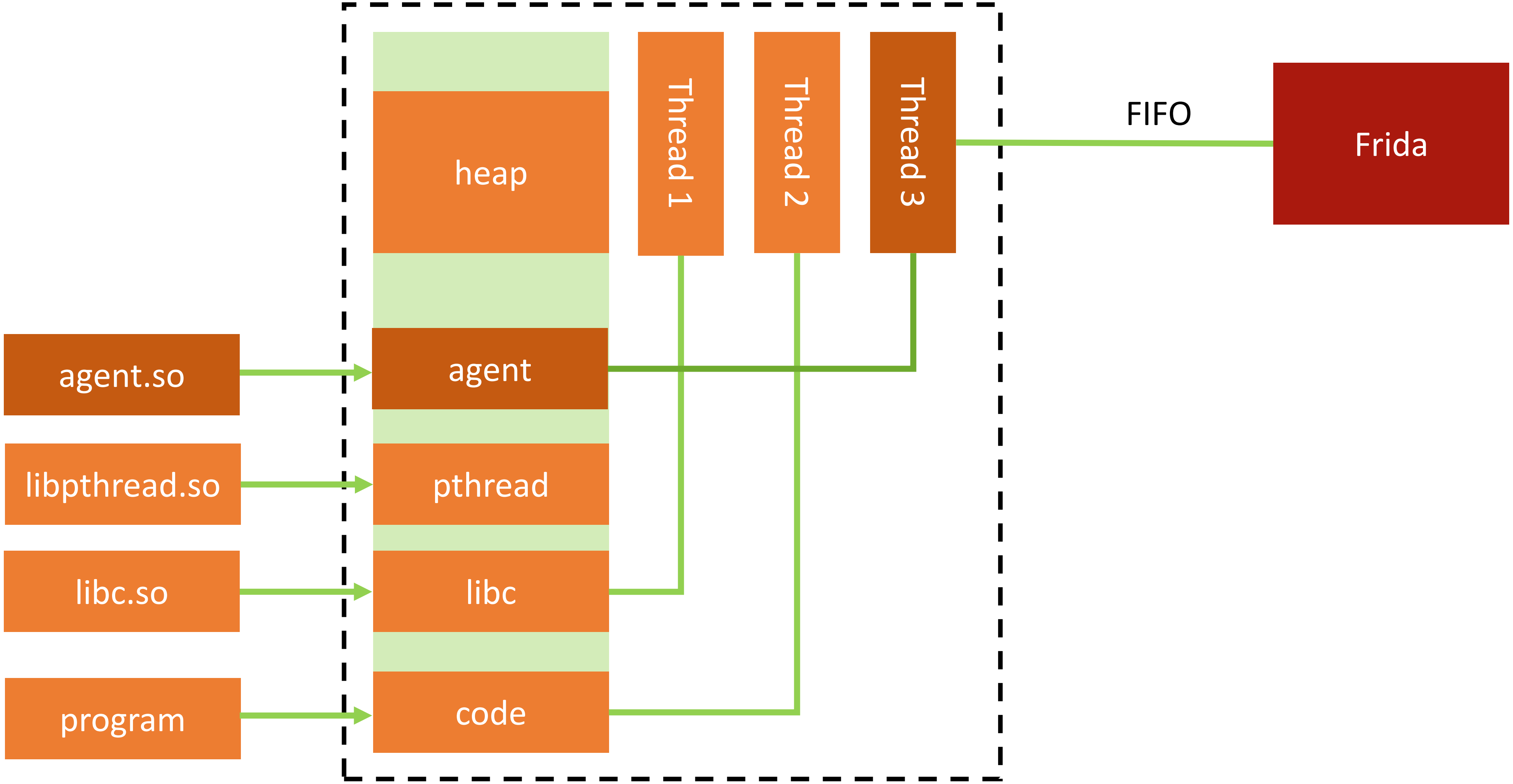
- Start a server that will be ready to attach to processes

- Injection:
  - Create thread in the remote process using `ptrace`
    - `PTRACE_ATTACH, PRACE_GETREGS`
  - Allocate memory for a bootstrapping code
    - Minimal amount of code required for pulling the agent
  - Write bootstrapper to memory
  - Execute bootstrapper in remote process
    - Open communication channel to server (FIFO)
    - Loads the agent from a shared library (.so)
    - Executes the agent
    - Closes communication channel (agent will expose an API)

FRIDA – mode Injected



FRIDA – mode Injected



FRIDA – mode Injected

- Required functionality:

- ptrace
  - Process tracing

- mmap
  - Map files to memory. In particular, the agent .so

- dlopen
  - Open the .so with the agent

- dlsym
  - Retrieves addresses of loaded symbols

- signal
  - To handle system signals

FRIDA – mode Injected - Howto

```
### On the first PC terminal

# wget frida-server from github

adb push frida-server /data/local/tmp
adb shell
su
cd /data/local/tmp
chmod +x frida-server
./frida-server

### On the second PC terminal
# List processes
$ frida-ps -U
```

FRIDA – How to use

- Command line tools: frida, frida-trace, frida-ps, frida-discover..

- Python interface
  - Provides a more advanced, programmatic interface
  - Allows predictable and repeatable instrumentation

- How to instrument code: using JS that overload existing functions
  - Large repository at: Frida CodeShare

Example: com.re.lab1

- Application requires a pin to unlock the flag
  - Pin is created dynamically and stored to an encrypted database
  - Application cannot be tampered as it checks the signature

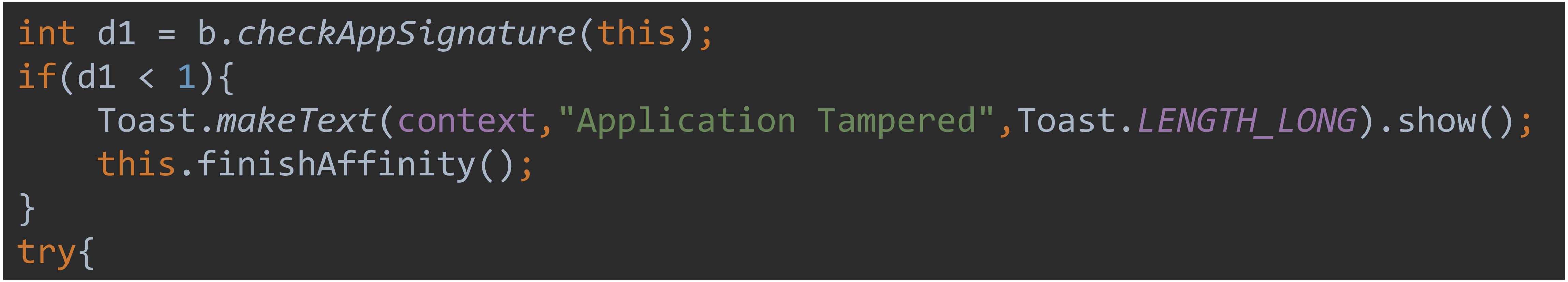
- Static analysis will yield little as the value is created on real time

- Approaches to dynamic analysis:
  - Insert a function to access the correct pin and log it to the terminal

com.re.lab1

Let’s break the check. Just to test.

Objective: make b.checkAppSignature return false

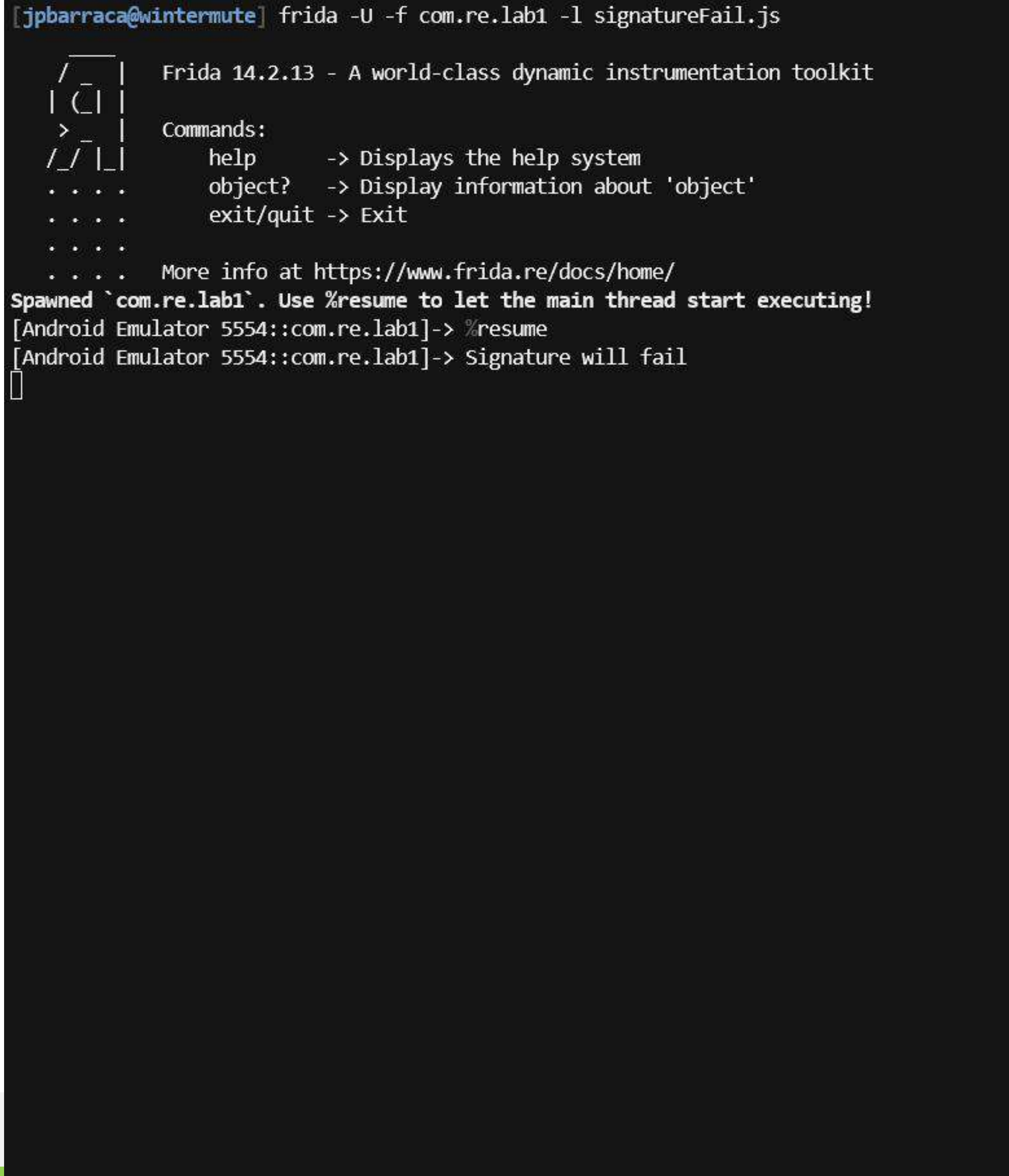


```
int d1 = b.checkAppSignature(this);
if(d1 < 1){
    Toast.makeText(context,"Application Tampered",Toast.LENGTH_LONG).show();
    this.finishAffinity();
}
try{
```

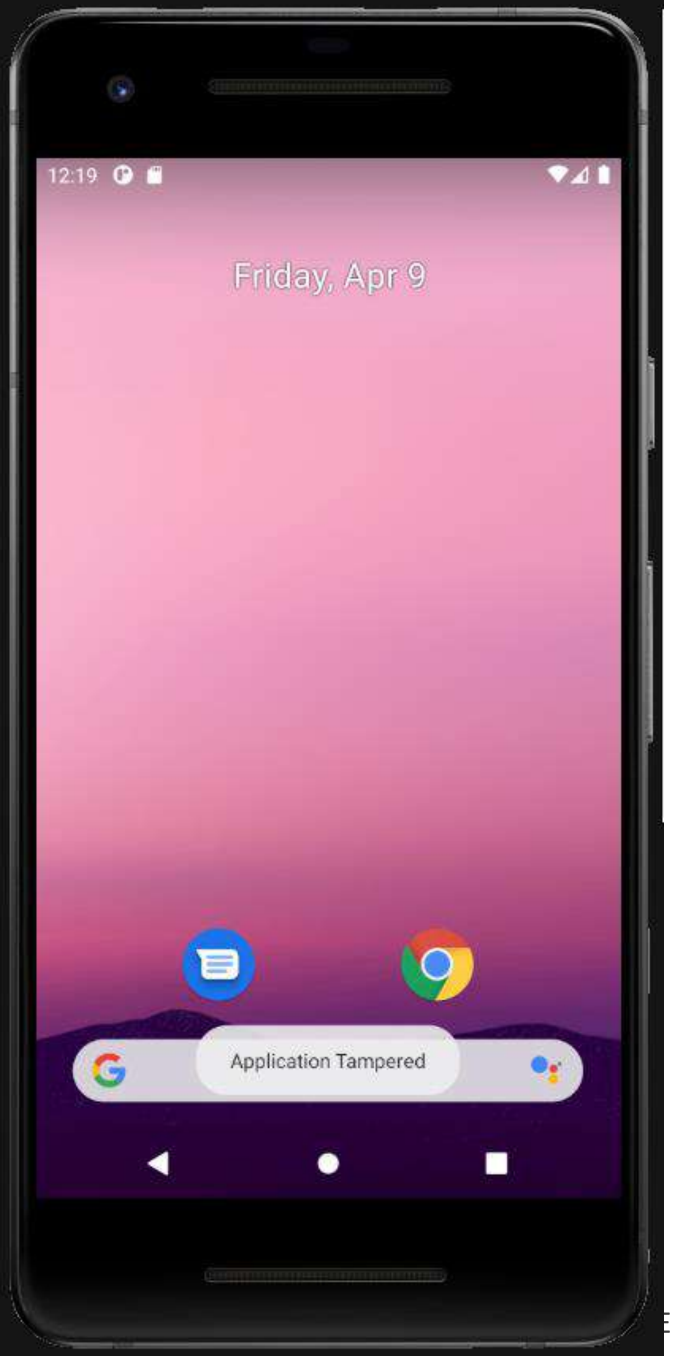
com.re.lab1 - Java.perform: executes the given payload

Snippet provides an alternative implementation of the method.

```
Java.perform(function(){
  Java.use("com.re.lab1.b").checkAppSignature.implementation = function(a) {
     console.log("Signature will fail");
     return 0;
  };
});
```



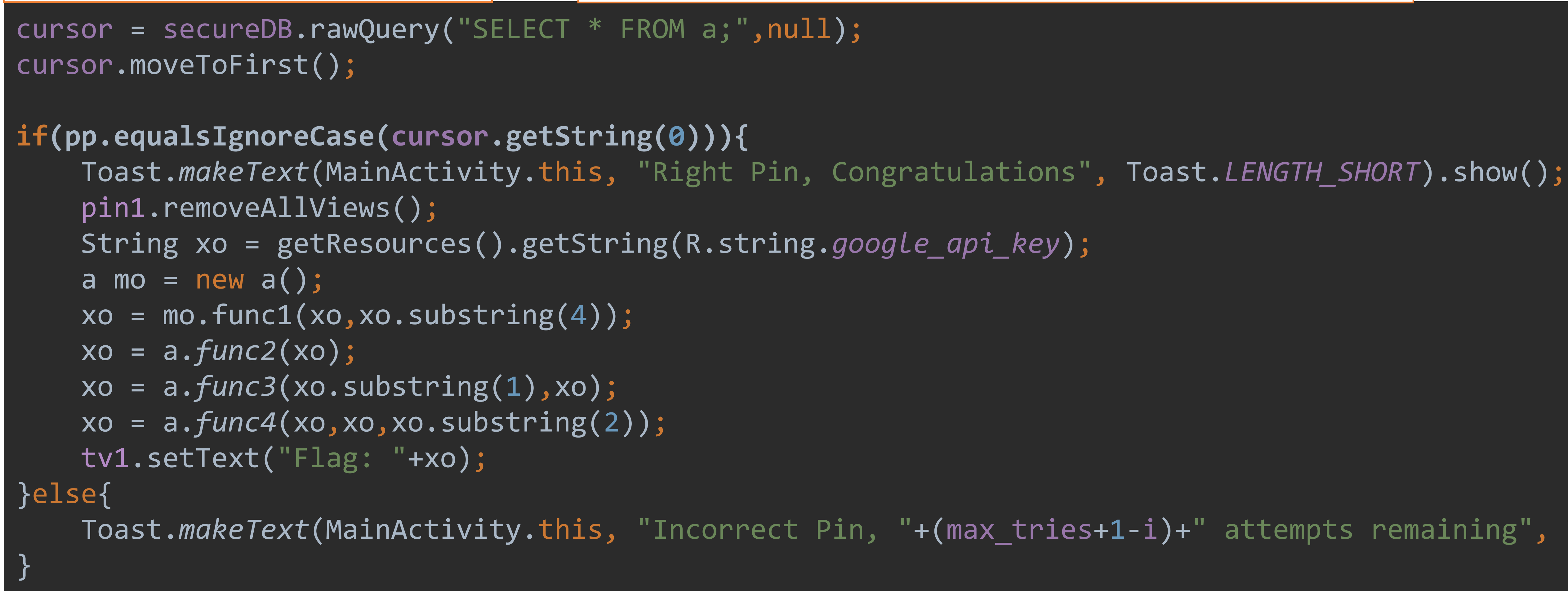
```
[jpbarraca@wintermute] frida -U -f com.re.lab1 -l signatureFail.js
     ____
    / _  |   Frida 14.2.13 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://www.frida.re/docs/home/
Spawned `com.re.lab1`. Use %resume to let the main thread start executing!
[Android Emulator 5554::com.re.lab1]-> %resume
[Android Emulator 5554::com.re.lab1]-> Signature will fail
```



Application Tampered

pp has the pin provided

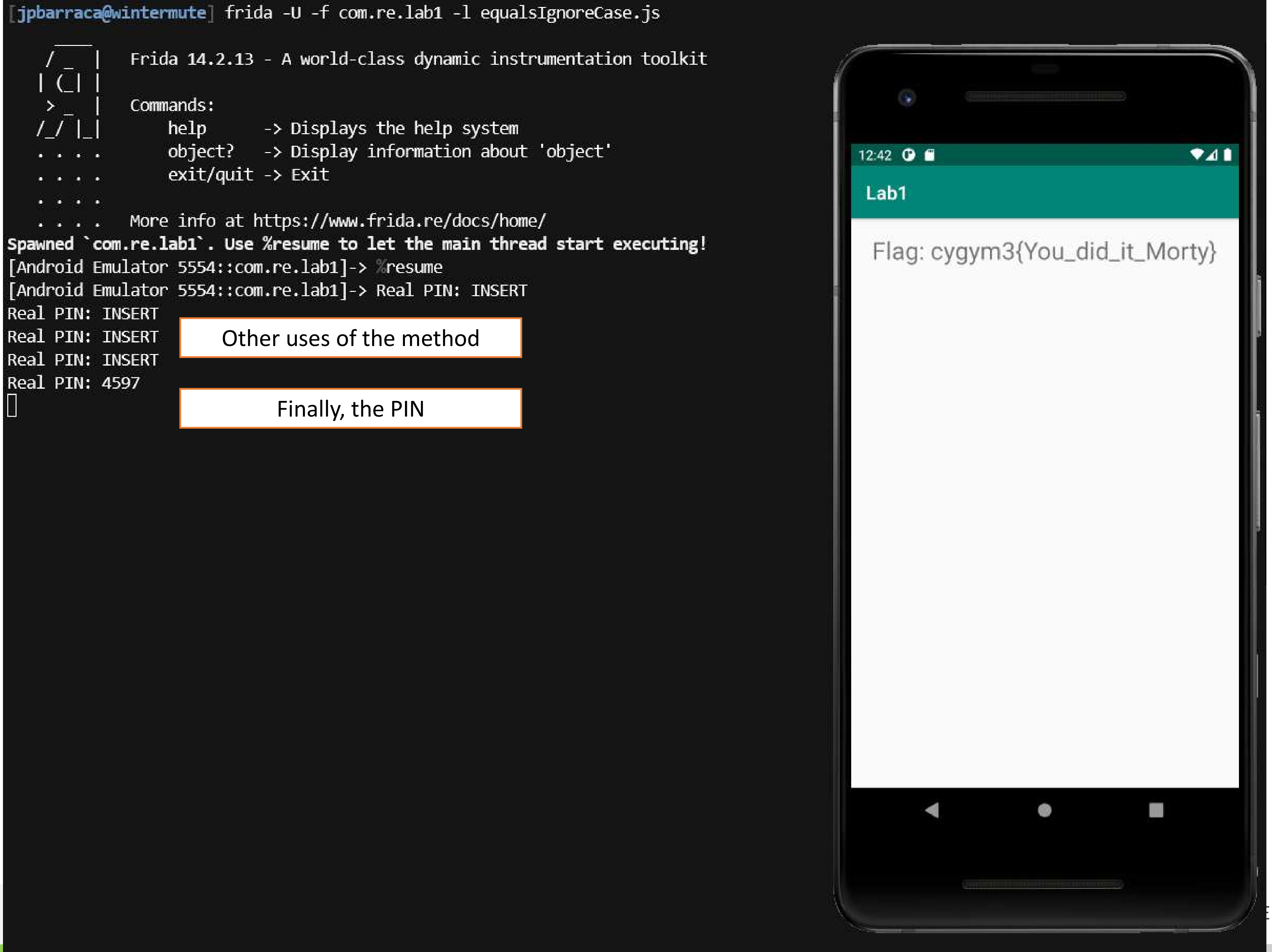
Cursor has the value obtained form the DB



```
cursor = secureDB.rawQuery("SELECT * FROM a;",null);
cursor.moveToFirst();

if(pp.equalsIgnoreCase(cursor.getString(0))){
    Toast.makeText(MainActivity.this, "Right Pin, Congratulations", Toast.LENGTH_SHORT).show();
    pin1.removeAllViews();
    String xo = getResources().getString(R.string.google_api_key);
    a mo = new a();
    xo = mo.func1(xo,xo.substring(4));
    xo = a.func2(xo);
    xo = a.func3(xo.substring(1),xo);
    xo = a.func4(xo,xo,xo.substring(2));
    tv1.setText("Flag: "+xo);
}else{
    Toast.makeText(MainActivity.this, "Incorrect Pin, "+(max_tries+1-i)+" attempts remaining",
}
```

Objective: reimplement Java.lang.String.equalsIgnoreCase so that it return true, and prints the correct ping

```
Java.perform(function(){
 Java.use("java.lang.String").equalsIgnoreCase.implementation = function(a) {
    console.log("Real PIN: " + a);
    return true;
 };
});
```



```
[jpbarraca@wintermute] frida -U -f com.re.lab1 -l equalsIgnoreCase.js
     ____
    / _  |   Frida 14.2.13 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://www.frida.re/docs/home/
Spawned `com.re.lab1`. Use %resume to let the main thread start executing!
[Android Emulator 5554::com.re.lab1]-> %resume
[Android Emulator 5554::com.re.lab1]-> Real PIN: INSERT
Real PIN: INSERT
Real PIN: INSERT
Real PIN: INSERT
Real PIN: 4597
```

Other uses of the method

Finally, the PIN

Flag: cygym3{You_did_it_Morty}

Interceptors: Intercepts calls to a function

- Define two events where code can be executed
  - **OnEnter**: When the function is called
  - **OnLeave**: After the function returns

- Can be used as generic logger, or to trigger other actions
  - Can intercept calls on lower layers of the application stack
    - Data that is to be written, sql queries, etc…

```
function foo(){
    Interceptor.attach(Module.findExportByName(“libc.so”, “open”), {
    onEnter: function(args){
        console.log(“Entering the function”);
    },
    onLeave: function(args){
       console.log(“Leaving the function”);
   },
 });
}
```