Using Android to attack ProGuard

(and saving 2€ for a ticket)

Who am I?

- @laughing_bit
- (C|Python|Twitter|Beamer|Mirabelle) Lover.
- Author of the SRE tool Chrysalide
- Daily job at Risk&Co

Android key points

- Application = code (.java) + dependencies (.class)
- APK = dx(ProGuard(javac(code) + dependencies))
- External repositories: Google, JCenter, ...
  - lots of repositories: https://mvnrepository.com/repos

Getting started

- Starting point: https://github.com/googlesamples
  - 176 results for repositories matching android written in Java
- Let's pick SimpleMediaPlayer as an example!

ProGuard

- Shrinks, optimizes and obfuscates Java bytecode
- Renames classes, fields, and methods (for instance a.a.a())
  - deterministic name obfuscation
  - default obfuscation dictionary: [a-z]+

ProGuard

- Shrinks, optimizes and obfuscates Java bytecode
- Renames classes, fields, and methods (for instance a.a.a())
  - deterministic name obfuscation
  - default obfuscation dictionary: [a-z]+

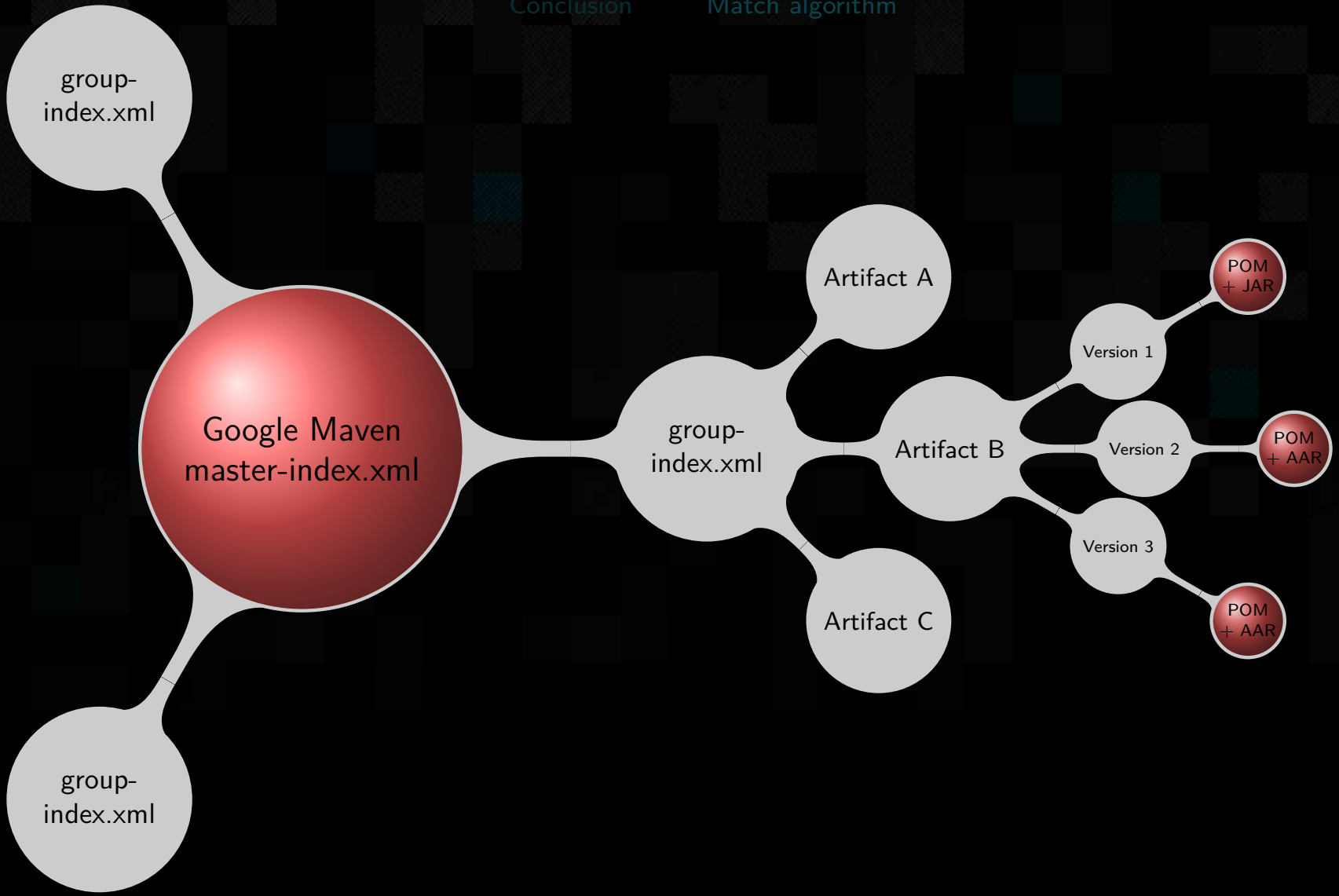
Advanced usage

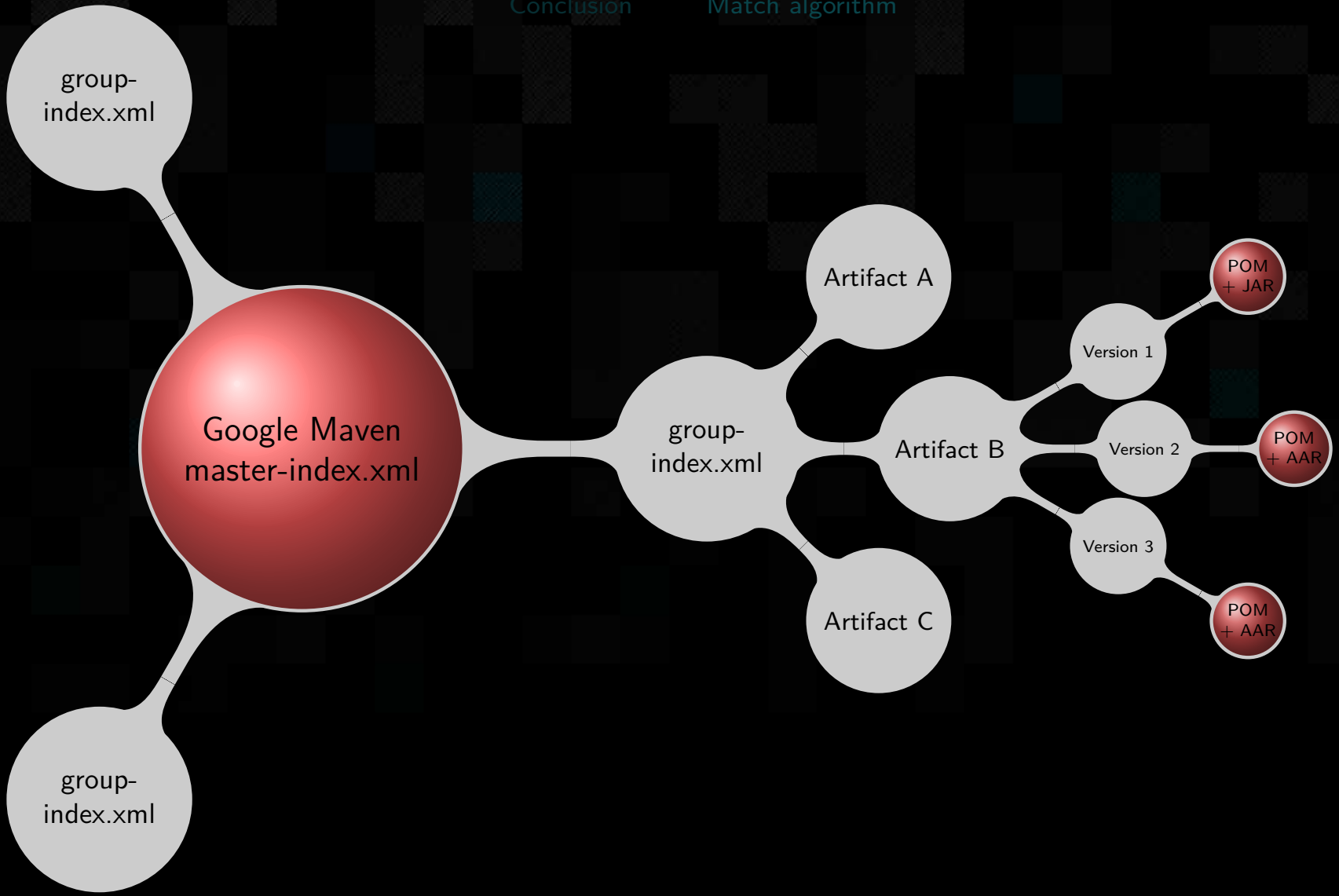
- Repackage all classes to a single root-level package
  - -repackageclasses
- Use custom obfuscation dictionaries (with reserved keywords)
  - -{,package,class}obfuscationdictionary
- Buy DexGuard
  - runtime self-protection
  - extra obfuscation: arithmetic and logical expressions + CFG

1. Collect Android package bytecode
   - easy to script
   - https://maven.google.com/: 1.2 Gb

1. Collect Android package bytecode
   - easy to script
   - https://maven.google.com/: 1.2 Gb
2. Fingerprint Android packages
   - easy to script

1. Collect Android package bytecode
   - easy to script
   - https://maven.google.com/: 1.2 Gb
2. Fingerprint Android packages
   - easy to script
3. Compare the fingerprints with obscucated code fingerprints
   - ~~easy to script~~ scriptable
   - if there is a match, obfuscation is reversed!


group-
index.xml
Google Maven
master-index.xml
group-
index.xml
Artifact A
Artifact B
Artifact C
Version 1
Version 2
Version 3
POM
+ JAR
POM
+ AAR
POM
+ AAR
group-
index.xml



https://developer.android.com/studio/build/dependencies#gmaven-access

Method

- Avoid to have to deal with similarity *and* confidence
- Select binary heuristics and hope $\sum similarity = identity$

Method

- Avoid to have to deal with similarity *and* confidence
- Select binary heuristics and hope $\sum similarity = identity$

Used heuristics

Method

- Avoid to have to deal with similarity *and* confidence
- Select binary heuristics and hope $\sum similarity = identity$

Used heuristics

- Small Primes Product
  - each instruction type is linked to a prime number

Method

- Avoid to have to deal with similarity *and* confidence
- Select binary heuristics and hope $\sum similarity = identity$

Used heuristics

- Small Primes Product
  - each instruction type is linked to a prime number
- Cyclomatic complexity
  - #edges - #nodes + 2 * #exits

Method

- Avoid to have to deal with similarity *and* confidence
- Select binary heuristics and hope $\sum similarity = identity$

Used heuristics

- Small Primes Product
  - each instruction type is linked to a prime number
- Cyclomatic complexity
  - #edges - #nodes + 2 * #exits
- Xrefs
  - count of: jumps, branchs, calls, links to strings

Method

- Avoid to have to deal with similarity *and* confidence
- Select binary heuristics and hope $\sum similarity = identity$

Used heuristics

- Small Primes Product
  - each instruction type is linked to a prime number
- Cyclomatic complexity
  - #edges - #nodes + 2 * #exits
- Xrefs
  - count of: jumps, branchs, calls, links to strings
- Machoc hash
  - Murmurhash3(<BB index>:[c,][<dest index>, ...];)

Method

- Avoid to have to deal with similarity *and* confidence
- Select binary heuristics and hope $\sum similarity = identity$

Used heuristics

- Small Primes Product
  - each instruction type is linked to a prime number
- Cyclomatic complexity
  - #edges - #nodes + 2 * #exits
- Xrefs
  - count of: jumps, branchs, calls, links to strings
- Machoc hash
  - Murmurhash3(<BB index>:[c,][<dest index>, ...];)
- Dex `code_item fields`
  - `registers_size ins_size outs_size tries_size` ~~`insns_size`~~

Method

- Avoid to have to deal with similarity *and* confidence
- Select binary heuristics and hope $\sum similarity = identity$

Used heuristics

- Small Primes Product
  - each instruction type is linked to a prime number
- Cyclomatic complexity
  - #edges - #nodes + 2 * #exits
- Xrefs
  - count of: jumps, branchs, calls, links to strings
- Machoc hash
  - Murmurhash3(<BB index>:[c,][<dest index>, ...];)
- Dex `code_item` fields
  - `registers_size ins_size outs_size tries_size` ~~`insns_size`~~
- `Filtered prototypes`
  - `(Ljava/lang/String;)[Landroid/support/a/a/h$b;`

Method

- Avoid to have to deal with similarity *and* confidence
- Select binary heuristics and hope $\sum similarity = identity$

Used heuristics

- Small Primes Product
  - each instruction type is linked to a prime number
- Cyclomatic complexity
  - #edges - #nodes + 2 * #exits
- Xrefs
  - count of: jumps, branchs, calls, links to strings
- Machoc hash
  - Murmurhash3(<BB index>:[c,][<dest index>, ...];)
- Dex `code_item` fields
  - `registers_size ins_size outs_size tries_size ~~insns_size~~`
- Filtered prototypes
  - `(Ljava/lang/String;)[Landroid/support/a/a/h$b;`
- Filtered class descriptors
  - `Landroid/support/v7/view/menu/e$2$1;`

1. Build a tree with all obfuscated symbol labels
   - nodes are parts of the labels: (package|class|routine) names
   - leafs contain AOSP candidates

1. Build a tree with all obfuscated symbol labels
   - nodes are parts of the labels: (package|class|routine) names
   - leafs contain AOSP candidates

2. Quickly filter some AOSP candidates
   - android.support.v7.app.b$a.a
   - android.support.v4.app.NoSaveStateFrameLayout.<init>

1. Build a tree with all obfuscated symbol labels
   - nodes are parts of the labels: (package|class|routine) names
   - leafs contain AOSP candidates

2. Quickly filter some AOSP candidates
   - android.support.v7.app.b$a.a
   - android.support.v4.app.NoSaveStateFrameLayout.<init>

3. Drop AOSP candidates by packages
   - select the biggest symbol
   - retrieve its original name by using binary diffing
   - remove AOSP packages which do not match

1. Build a tree with all obfuscated symbol labels
   - nodes are parts of the labels: (package|class|routine) names
   - leafs contain AOSP candidates
2. Quickly filter some AOSP candidates
   - android.support.v7.app.b$a.a
   - android.support.v4.app.NoSaveStateFrameLayout.<init>
3. Drop AOSP candidates by packages
   - select the biggest symbol
   - retrieve its original name by using binary diffing
   - remove AOSP packages which do not match
4. Deobfuscate all remaining ProGuard'ed symbols
   - keep the best match with binary diffing

- Current status
  - Work In Progress...
  - Limitations: only the external dependencies are processed
    - still an extra help for disassembly understanding!
- Full Python bindings
  - https://chrysalide.re/api/python/pychrysalide-analysis-diffing

```
$ python3dm ./debug-deguard.py classes-proguarded.dex android.support.v7.widget.ag.a \
> ~/.config/chrysalide/diffing/9d14147473a6ecef05ede35c50a2e178c572ae063eb01607dfe0d4386249b2d9.tar.xz

==>> android.support.v7.widget.ag.a @ 0x58b14

                                                    | Hash | Order | Primes | Machoc | CC | Xrefs | Dex CI | DexOwner | DexProto
  android.support.v7.widget.ForwardingListener.access$000           |      |       |   X    |   X    | X  |   X   |   X    |    X     |    X

==>> android.support.v7.widget.ag.a @ 0x58b2c

                                                    | Hash | Order | Primes | Machoc | CC | Xrefs | Dex CI | DexOwner | DexProto
  android.support.v7.widget.ForwardingListener.addDetachListenerApi12 |      |       |   X    |   X    | X  |   X   |   X    |    X     |    X

==>> android.support.v7.widget.ag.a @ 0x58b50

                                                    | Hash | Order | Primes | Machoc | CC | Xrefs | Dex CI | DexOwner | DexProto
  android.support.v7.widget.ForwardingListener.onTouchObserved       |      |       |   X    |   X    |    |       |        |    X     |    X

==>> android.support.v7.widget.ag.a @ 0x58c48

                                                    | Hash | Order | Primes | Machoc | CC | Xrefs | Dex CI | DexOwner | DexProto
  android.support.v7.widget.ForwardingListener.pointInView           |      |       |   X    |   X    | X  |   X   |   X    |    X     |    X

==>> android.support.v7.widget.ag.a @ 0x58cb0

                                                    | Hash | Order | Primes | Machoc | CC | Xrefs | Dex CI | DexOwner | DexProto
  android.support.v7.widget.ForwardingListener.toLocalMotionEvent    |      |       |   X    |   X    | X  |   X   |        |    X     |    X
```

- Test with real world samples
  - scale does matter!
  - but Chrysalide does not disassemble large APKs yet... (ENOMEM)
- Check for debug information
  - class names could leak from source files
- Improve processing time by relying on POM dependencies
- Deobfuscate class members as well

Thank you!