ANDROID APP DEOBFUSCATION USING COOPERATIVE ANALYSIS

vb 2018
MONTREAL
3 – 5 October 2018

Yoni Moses
Yaniv Mordekhay

Check Point
SOFTWARE TECHNOLOGIES LTD.

LET’S TALK ABOUT

AUTOMATED MOBILE MALWARE DETECTION

AUTOMATED ANDROID APP ANALYSIS

Analysis cycles from low cost to high cost:

1. Feature extraction
   1. Static analysis
   2. Dynamic analysis
2. Features to threat factors
   1. Predicates
   2. Machine learning
3. Verdict

Static analysis

- Analyzes the app without running it
- Disassembles APK to smali code and looks for sensitive data flows
- Strengths:
  - Covers all available code
- Weaknesses:
  - Cannot analyze encrypted code and data
  - Cannot analyze dynamically loaded code and data
  - Java code analysis doesn't work for native and vice versa

Dynamic Analysis

- Exposes behaviors by actually running the app
- Strengths:
  - Does not care about encryption or obfuscation
  - Analyzes dynamically loaded code
  - Agnostic to programming languages
- Weaknesses
  - Hard to reach full coverage (UI, network, location…)

ONE'S WEAKNESS - OTHER'S STRENGTH

- Benefits of cooperation
  - Strengths and weaknesses complete each other
  - Double validation of behaviors
  - Better coverage
  - Less manual work for analysts
- Why isn't it more common?
  - Different skill sets
  - Very different execution environments
  - Personal rivalry

TOGETHER – THEY ARE **INVINCIBLE!**

Code Extraction and Unpacking

- Dyna extracts dynamically loaded binaries:
  - Binaries bundled as app assets
  - Binaries downloaded in run-time
- Dyna decrypts packed binaries
  - Research presented at DefCon 2017
- The binaries are passed to Static and analyzed along the main binary (classes.dex)

CODE OBFUSCATION

- WIDELY USED BY APP DEVELOPERS (BOTH MALICIOUS AND BENIGN)
- COMMON TECHNIQUES:
  - CLASS AND METHOD RENAMING
  - STRING ENCRYPTION
  - DYNAMIC METHOD BINDING BY REFLECTION (OFTEN COMBINED WITH STRING ENCRYPTION)

Focusing on String Encryption

- Encrypted strings could be:
  - names of sensitive APIs called by reflection
  - paths to content providers; e.g "content://sms"
- Decryption by static analysis is hard
- Decryption is done automatically anyway if we run the app

WHAT IF?

DYNA

READS THE DECRYPTED STRINGS DURING RUNTIME

AND PASSES THEM TO

STATIC

COMMON OBFUSCATION IMPLEMENTATION

Obfuscators create a new binary where string initialization code is replaced with decryption method call

Before obfuscation:

```
const-string v1, “content://sms”
```

After obfuscation:

```
const-string v1, “\u000f\u0003\u000e....”
const/16 v2, 0x1cb
invoke-static {v1, v2}, Lorg/foo/a;->bar(…)String;
move-result-object v1
```

Cooperative Decryption

Naïve Approach

- Static looks for decryption calls and passes them to Dyna
- Before app execution, Dyna places breakpoints at decryption calls
- At runtime, Dyna records decrypted strings and passes them to Static

- Will Dyna cover all decryption calls?

Cooperative Decryption

Practical Approach

1. **Static** collects the decryption calls using predefined signatures
2. The collected calls are passed to **Dyna**, with their argument values
3. **Dyna** loads app code
4. **Dyna** runs, in background, the decryption calls received from
5. **Dyna** returns the decrypted values to **Static**
6. **Static** patches the code and runs the regular analysis

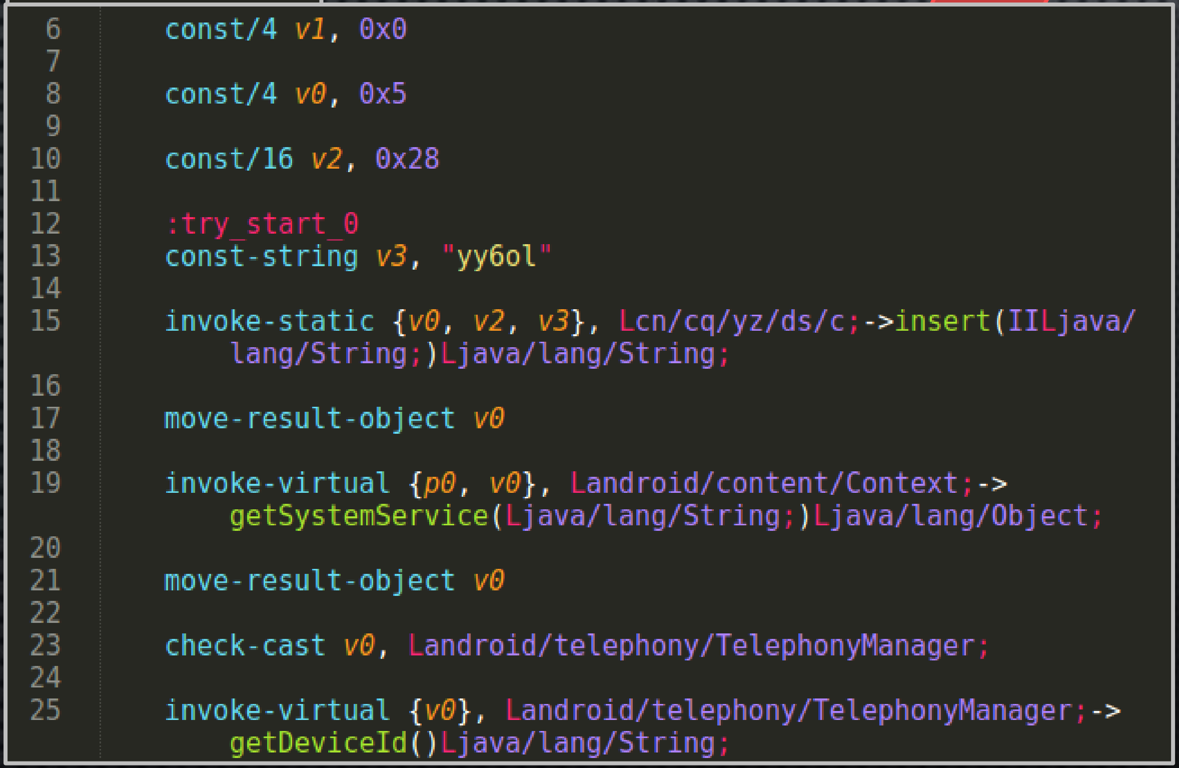
Cooperative Decryption

Practical Approach

- **Static** creates a patched DEX using the data from **Dyna**:
    - Replaces decryption calls with decrypted strings
    - Removes reflection usage:
        1. Looks for calls to `java.lang.reflect.Method.invoke()`
        2. Performs backtrack search for names of invoked methods
        3. Replaces calls to `Method.invoke()` with ordinary calls

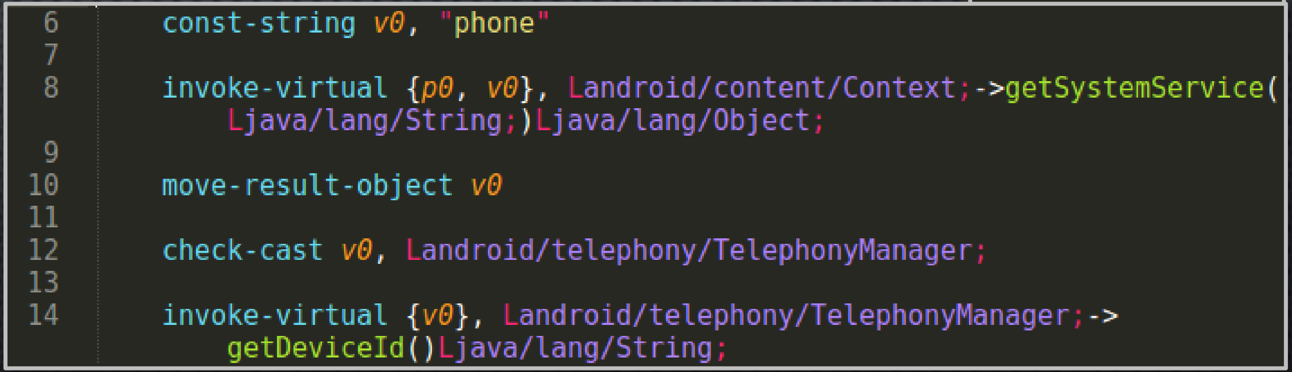
DEX PATCHING

BEFORE



```
    const/4 v1, 0x0

    const/4 v0, 0x5

    const/16 v2, 0x28

    :try_start_0
    const-string v3, "yy6ol"

    invoke-static {v0, v2, v3}, Lcn/cq/yz/ds/c;->insert(IILjava/
        lang/String;)Ljava/lang/String;

    move-result-object v0

    invoke-virtual {p0, v0}, Landroid/content/Context;->
        getSystemService(Ljava/lang/String;)Ljava/lang/Object;

    move-result-object v0

    check-cast v0, Landroid/telephony/TelephonyManager;

    invoke-virtual {v0}, Landroid/telephony/TelephonyManager;->
        getDeviceId()Ljava/lang/String;
```

AFTER



```
    const-string v0, "phone"

    invoke-virtual {p0, v0}, Landroid/content/Context;->getSystemService(
        Ljava/lang/String;)Ljava/lang/Object;

    move-result-object v0

    check-cast v0, Landroid/telephony/TelephonyManager;

    invoke-virtual {v0}, Landroid/telephony/TelephonyManager;->
        getDeviceId()Ljava/lang/String;
```

Limitations

- Showcasing cooperation is more important than covering all the cases
- We made our life easy:
  - Only static methods
  - Only constant arguments
  - Only methods without side effects

Experiment: DashO Decryption

- DashO – common commercial obfuscator
- Signature for its decryption methods:
  - Static method
  - String return value
  - 3-4 arguments: 2-3 ints and one String
  - Exceptions caught only if they inherit from RuntimeException
  - No secondary calls except for String class methods
- The signature yielded 586 samples in our database

WHAT IS HIDING THERE?

```
com.threelm.dm.api.IDeviceManagerApi
<font>%s</font><font color="#%06X"> %s  </font>
PhoneUtils
OUTGOING_SERVER_CMD
eula.version.name
android.app.extra.DEVICE_ADMIN
SELECT DISTINCT familyName FROM trustedPUPTable ORDER BY familyName
 mIndex=
safe_sim
android.intent.action.SEND
, for type token:
getLong(lockscreen.password_type)
TO
fragment
```

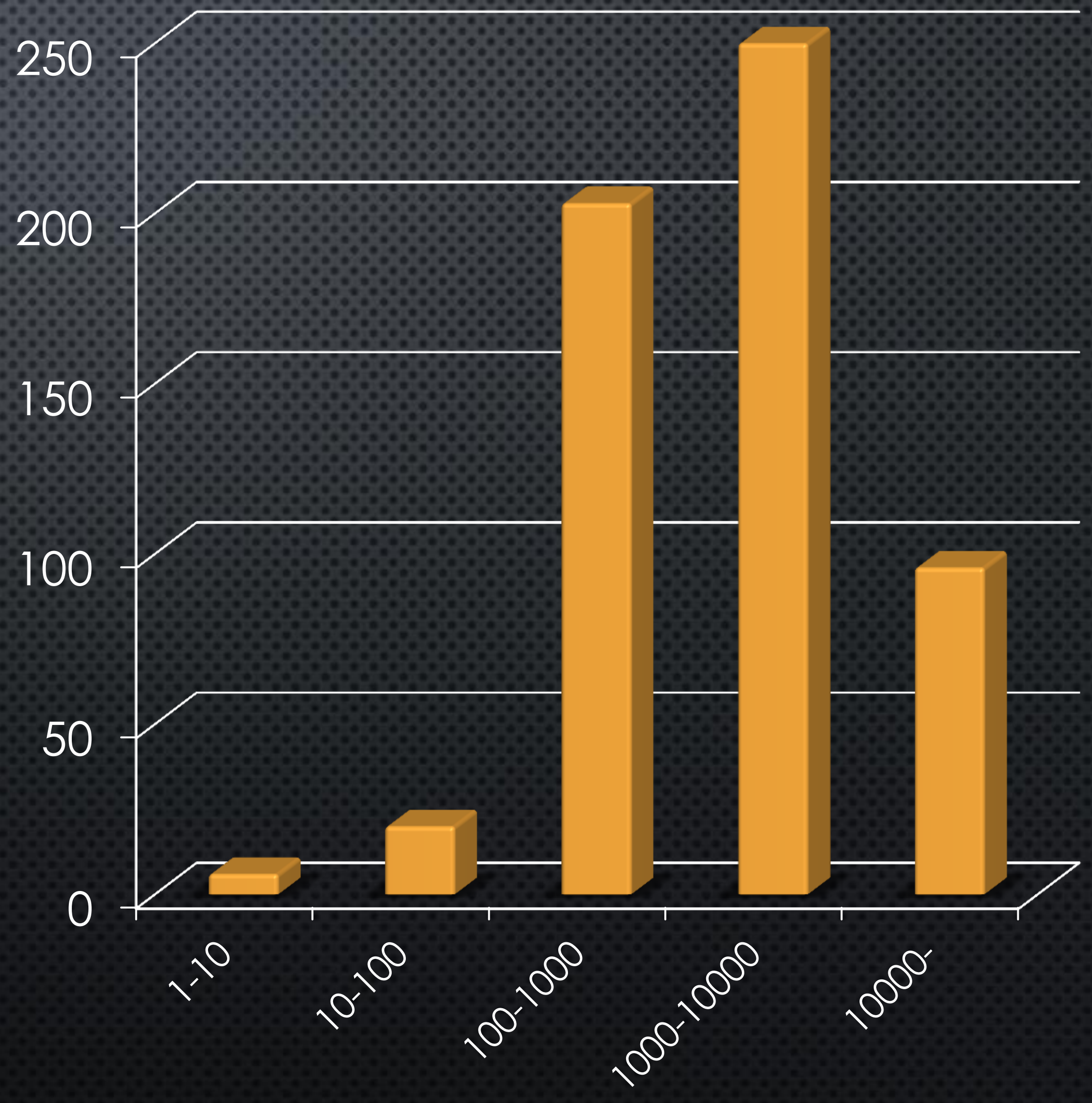
```
familyName
layout_inflater
body
SETTINGS
LoaderManager
buddyNotified
t_url
android.intent.action.MEDIA_MOUNTED
DexHash
Caught exception reading the GList.
544
 filter=
OwnerName
AppVerCode
TopAppMonitor
MUP
```

```
logparse
com.wsandroid.managers.STATE_RECEIVER
pref.debug.settings
LaunchManager
BLD_VER_INCREMENTAL
214
C2dmToken
http
ER
select type from AppTrustInfoBrief where pkg='
CloudReputationDB
Activated
SubscriptionStartTime
911;112;
InvalidInstallIdDeviceTypeMatch
```

EXPERIMENT RESULTS

- Static detected new flows in 10.4% of the samples
  - Access to Google account credentials
  - Access to SMS and contacts content providers
  - Device admin privileges request

Number of decryption calls per app



Easier said than done

- Dynamic-Static communication
  - Non parallel execution
  - Static runs twice
  - Different environments
- Android Runtime hacking
  - Violates the Prime Directive!
- Testing
  - Requires sophisticated infrastructure for real testing

AM I THE ONLY ONE AROUND HERE
THAT CARES ABOUT THE PRIME DIRECTIVE?
made on imgur

And then we deployed to production

- DashO and DexGuard
- Some apps really love encryption
  - Median app contains 13 encrypted strings
  - Maximum encountered: 13,976
  - Most apps decrypt very short strings, some decrypt megabytes
- Much more volatile
- Much more infrastructure dependencies
- Very low performance impact!

What is next?

- Breadth first approach
  - Cover more “simple” obfuscators
- Depth first approach
  - Be able to handle state
  - Be able to run code that is not contained in a method
- Non signature based search
- Reconstruct other types of data
- Use cooperation to improve dynamic coverage
- Feed the data into the ML engines!

THANK YOU!

Yoni Moses: yonimo@checkpoint.com
Yaniv Mordekhay: yanivmo@checkpoint.com

vb 2018
MONTREAL
3 – 5 October 2018

Check Point
SOFTWARE TECHNOLOGIES LTD.