

POCKET-SIZED BADNESS

WHY RANSOMWARE COMES AS A PLOT TWIST
IN THE CAT-MOUSE GAME

Federico 'phretor' Maggi
Stefano 'raistlin' Zanero



\$ whoami

- Forward-Looking Threat (FTR) researcher @ Trend Micro
 - Research on upcoming/future threats and risks
 - Cybercrime investigation
- Interested in too many things
- Formerly Assistant Professor @ POLIMI
 - ~50 papers published
 - ~25 invited talks & lectures
 - ~100 students supervised
- PC/board member of various conferences/workshops
 - ACSAC (first week of December, Hollywood)
 - AsiaCCS
 - DIMVA
 - OWASP AppSecEU
 - EuroSec

AGENDA

- Quick **retrospective** on ransomware
- The humble beginning of **mobile** ransomware
- **Case studies** of mobile ransomware
- Typical **technical features** of mobile ransomware
 - And how to automatically **detect** them
- **Tool!**
 - How it works
 - Does it work?
 - How to get it!
- **Conclusions:** An economics perspective on ransomware

FROM CRYPTOVIROLOGY TO MOBILE RANSOMWARE

*Supported in part by NSF grant CCR-93-16209 and CISE Institutional Infrastructure grant CDA-90-24735. Published in the Proceedings of the 1996 IEEE Symposium on Security and Privacy, May 6-8. Copyright 1996 IEEE

Cryptovirology: Extortion-Based Security Threats and Countermeasures*

Adam Young
Dept. of Computer Science,
Columbia University.

Moti Yung
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598.

Abstract

Traditionally, cryptography is defensive in nature, a protection, and security to users. The idea of *Cryptovirology* is to change the nature of cryptography, showing that By being offensive we

tion, and security to users. In this paper we present the idea of *Cryptovirology* which employs a twist on cryptography, showing that it can also be used offensively. By being offensive we mean that it can be used to mount extortion based attacks that cause loss of access to information, loss of confidentiality, and information

mount extortion based attacks that cause loss of access to information, loss of confidentiality, and information leakage, tasks which cryptography typically prevents. In this paper we analyze potential threats and attacks that rogue use of cryptography can cause when combined with rogue software (viruses, Trojan horses), and

for such attacks to occur. In this paper we attempt a first step in this direction by presenting a set of cryptography-exploiting computer security attacks and potential countermeasures.

The set of attacks that we present involve the various uses of strong (public-key and symmetric) cryp-

short period of time in which they are in RAM.

The information extortion attack could translate directly into the loss of U.S. dollars if electronic money is implemented. In fact, the potential for attacks on anonymous e-money has been recognized in the cryptographic literature [vSN92, BGK95, SPC95, JY96]; we materialize an attack via a cryptovirus. A specialized cryptovirus could be designed to search for e-money notes and encrypt them. In this way, the virus writer can effectively hold all the money ransom until half of it is given to him. Even if the e-money was previously

short period of time in which they are in RAM.

The information extortion attack could translate directly into the loss of U.S. dollars if electronic money is implemented. In fact, the potential for attacks on anonymous e-money has been recognized in the cryptographic literature [vSN92, BGK95, SPC95, JY96]; we materialize an attack via a cryptovirus. A specialized cryptovirus could be designed to search for e-money notes and encrypt them. In this way, the virus writer can effectively hold all the money ransom until half of it is given to him. Even if the e-money was previously encrypted by the user, it is of no use to the user if it gets encrypted by a cryptovirus. Electronic money

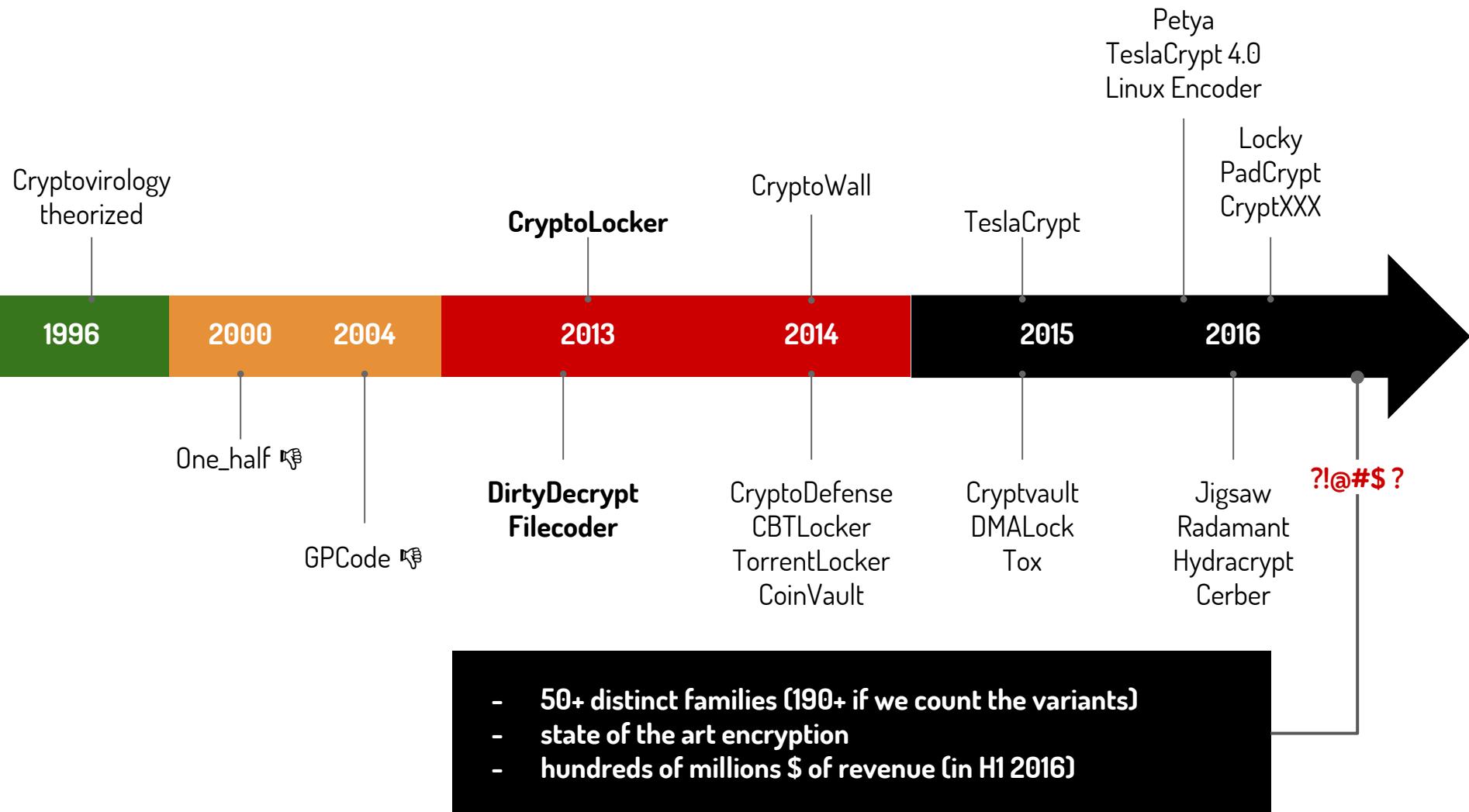
detail. We have shown
in be used in a virus to
ne user cannot retrieve
ypt D' to get D , the
somewhere, since oth

We cannot store the
in the network, since
node the entire private
network as a host we
he power of the user,
ent users who do not

have access to each others data. The secret sharing
virus takes advantage of this property by sharing its
private key among m nodes, where $m > 1$. The virus
therefore exploits the access controls that users place
on themselves to keep its private key secret.

The idea is that a virus will spread itself around the
network, and may act autonomously or wait for out
side control to act as an agent of the writer. Note that
the local users my wipe out parts of the virus (assum-
ing they have back-ups), but then the total network
may be damaged (since we need the entire virus pieces
to recover). It may therefore be useful for the virus to
immediately notify the local machines that if they rid

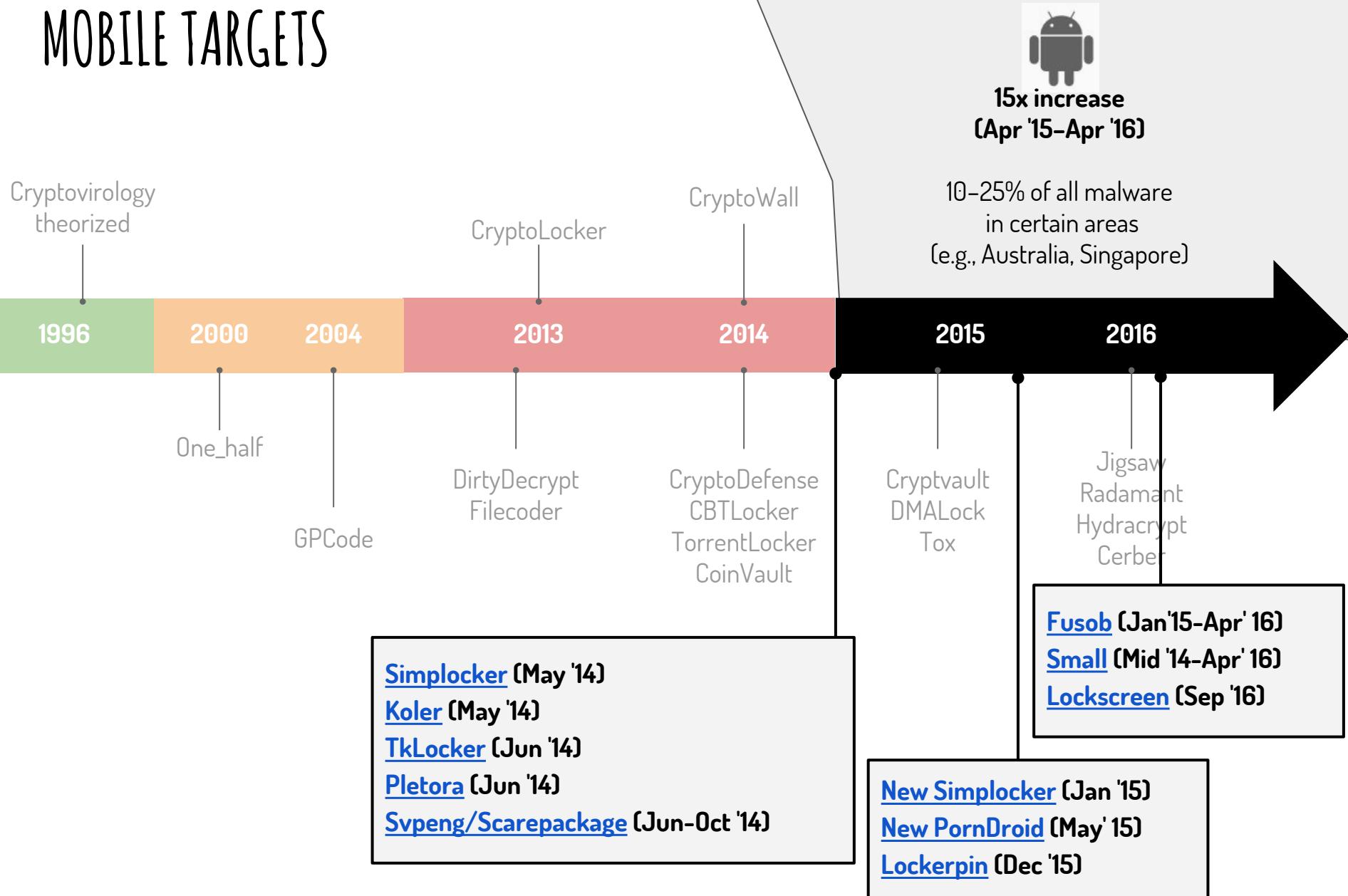
1996 → 2012





THE "LITTLE" WORLD OF MOBILE RANSOMWARE

MOBILE TARGETS



CASE STUDIES OF MOBILE RANSOMWARE



SLOCKER (A.K.A. SIMPLOCKER, OR SIMPLE LOCKER)

- First Android ransomware family
- ~15,600 samples analyzed
- No "real" encryption:
 - Hides files
 - Contains and uses AES routines only in ~50 samples
 - 2015 variant does use encryption with per-device key
- Uses SMS (sometimes Tor) for C&C communication

- **Screen locking** via soft button event hijacking



SCREEN LOCKING: BASIC TECHNIQUE

onKeyDown

Added in API level 1

```
boolean onKeyDown (int keyCode,  
                  KeyEvent event)
```

Called when a key down event has occurred. If you return true, you can first call `KeyEvent.startTracking()` to have the framework track the event through its `onKeyUp(int, KeyEvent)` and also call your `onKeyLongPress(int, KeyEvent)` if it occurs.

- * Article 161
- * Article 148
- * Article 215
- * Article 301

* of the Criminal Code of U.S.A.

Your device was used to visit
websites containing pornography.



Returns

boolean

If you handled the event, return true. If you want to allow the event to be handled by the next receiver, return false.

[https://developer.android.com/reference/android/view\(KeyEvent.Callback\).html#onKeyDown\(int, android.view.KeyEvent\)](https://developer.android.com/reference/android/view(KeyEvent.Callback).html#onKeyDown(int, android.view.KeyEvent))



THREATENING
TEXT

KOLER: POLICE-THEMED RANSOMWARE

- ~5,000 samples analyzed
- <0.05% use encryption
 - The rest just threatens victims
- Interesting and quite complex distribution network

- **Well-localized police-themed locking screen**
 - 60 languages
 - Country-specific LEA and gov-related lingo
 - Key names and photos used



Screenshots courtesy of [Kaffeine](#).



THREATENING
TEXT

ATTENTION!

Your phone has been blocked up for safety reasons listed below.

All the actions performed on this phone are fixed.

All your files are encrypted.

CONDUCTED AUDIO AND VIDEO.

You are accused of viewing/storage and/or dissemination of banned pornography (child pornography/zooophilia/rape etc). You have violated World Declaration on non-proliferation of child pornography. You are accused of committing the crime envisaged by Article 161 of United States of America criminal law.

Article 161 of United States of America criminal law provides for the punishment of deprivation of liberty for terms from **5 to 11** years.

Also, you are suspected of violation of "Copyright and Related rights Law" (downloading of pirated music, video, warez) and of use and/or dissemination of copyrighted content. Thus, you are suspected of violation of Article 148 of United States of America criminal law.

Article 148 of United States of America criminal law provides for the punishment of deprivation of liberty for terms from **3 to 7** years or **150 to 550** basic amounts fine.

It was from your phone, that unauthorized access had been stolen to information of State importance and to data closed for public Internet access.



SVPENG: REAL ENCRYPTION

- ~2,000 samples analyzed
- Used to be a banking trojan used to steal ~\$1M in frauds
- Ransomware features were added later. We found:
 - Encryption
 - Screen locking
 - As illustrated earlier
 - Via Device Admin API abuse
 - Threatening text
- A good summary of all mobile ransomware techniques



DATA
ENCRYPTION

```
.class public Lorg/simplelocker/AesCrypt;

.method public constructor <init>(Ljava/lang/String;)V
    # [...]
    # A cipher instance variable is created
    const-string v2, "AES/CBC/PKCS7Padding"
    invoke- static {v2}, Ljavax/crypto/Cipher;->getInstance(Ljava/lang/String;)Ljavax/crypto/Cipher;
    move-result-object v2
    input-object v2, p0, Lorg/simplelocker/AesCrypt;->cipher:Ljavax/crypto/Cipher;
    # [...]
.end method

.method public encrypt(Ljava/lang/String;Ljava/lang/String;)V
    # [...]
    # The cipher instance is initialized in encrypt mode
    const/4 v6, 0x1
    ige-object v7, p0, Lorg/simplelocker/AesCrypt;->key:Ljavax/crypto/spec/SecretKeySpec;
    ige-object v8, p0, Lorg/simplelocker/AesCrypt;->spec:Ljava/security/spec/AlgorithmParameterSpec;
    invoke-virtual {v5, v6, v7, v8},
        Ljavax/crypto/Cipher;->init(ILjava/security/Key;Ljava/security/spec/AlgorithmParameterSpec;)V
    new-instance v1, Ljavax/crypto/CipherOutputStream;
    ige-object v5, p0, Lorg/simplelocker/AesCrypt;->cipher:Ljavax/crypto/Cipher;
    invoke-direct {v1, v4, v5},
        Ljavax/crypto/CipherOutputStream;-><init>(Ljava/io/OutputStream;Ljavax/crypto/Cipher;)V
    # [...]
.end method
```



DEVICE ADMIN API ABUSE (IN THE MANIFEST)

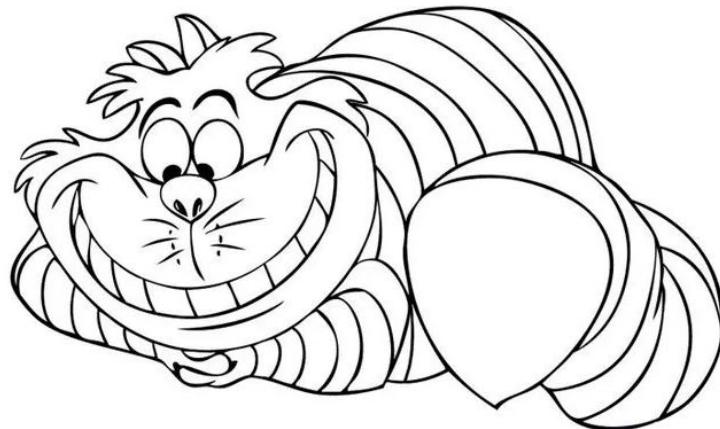
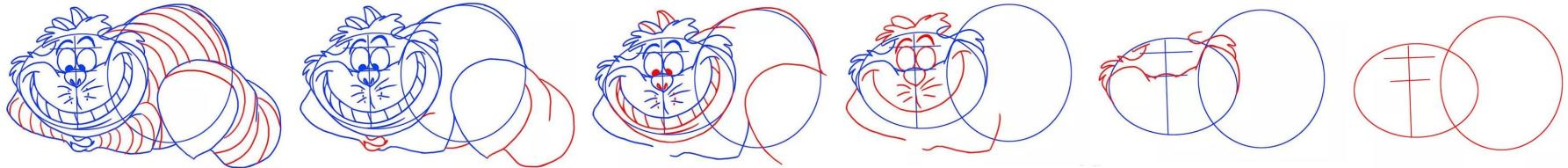
```
<activity android:name=".app.DeviceAdminSample"
          android:label="@string/activity_sample_device_admin">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.SAMPLE_CODE" />
    </intent-filter>
</activity>
<receiver android:name=".app.DeviceAdminSample$DeviceAdminSampleReceiver"
          android:label="@string/sample_device_admin"
          android:description="@string/sample_device_admin_description"
          android:permission="android.permission.BIND_DEVICE_ADMIN">
    <meta-data android:name="android.app.device_admin"
              android:resource="@xml/device_admin_sample" />
    <intent-filter>
        <action android:name="android.app.action.DEVICE_ADMIN_ENABLED" />
    </intent-filter>
</receiver>
```



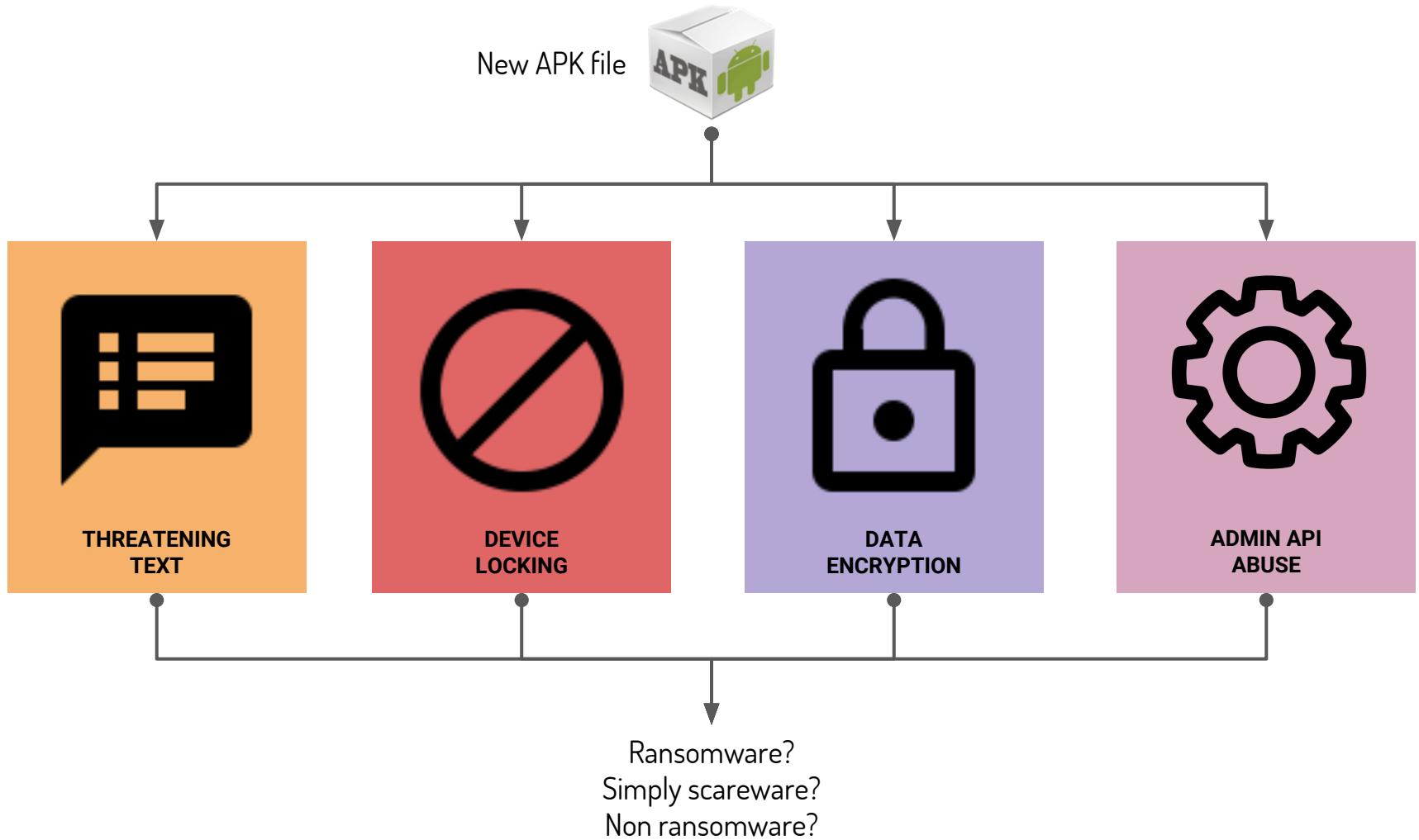
LOCKDROID.E & GOOGLE'S PROMPT PATCH IN NOUGAT

- Newest family: Sep 2016
- Uses
 - `resetPassword(pseudo-random passcode)`
 - `lockNow()`
- In Nougat, `resetPassword()` is "one use"
 - If password is set already
 - `resetPassword()` can't be called
- No backward compatibility
- No (benign) apps can automatically change/reset the password
 - User interaction will always be required

DETECTION TECHNIQUES



ANALYSIS & DETECTION APPROACH





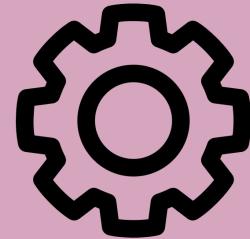
**THREATENING
TEXT**



DEVICE
LOCKING



DATA
ENCRYPTION

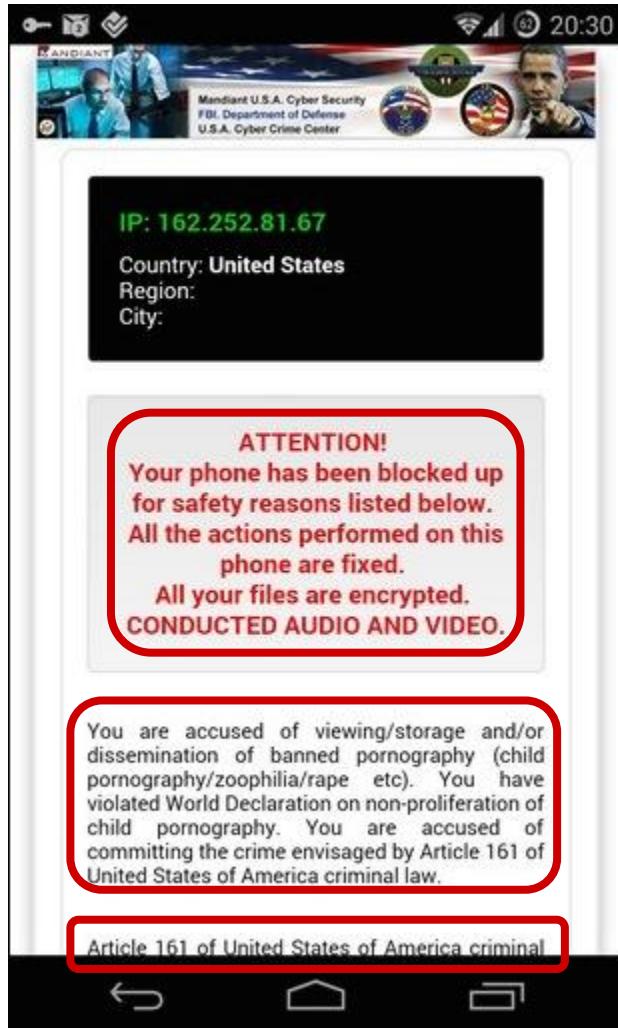


ADMIN API
ABUSE



THREATENING
TEXT

THREATENING TEXT: DETAILS

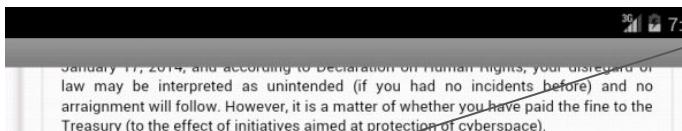


- **must be clear**, understandable and **convincing**
- **coercion** techniques
 - refer to **law codes**
 - various **accusations**
 - **copyright** violation
 - **illegal** content found
 - **prohibited** sites visited
- **detailed payment instructions**



THREATENING
TEXT

THREATENING TEXT: PAYMENT INSTRUCTIONS



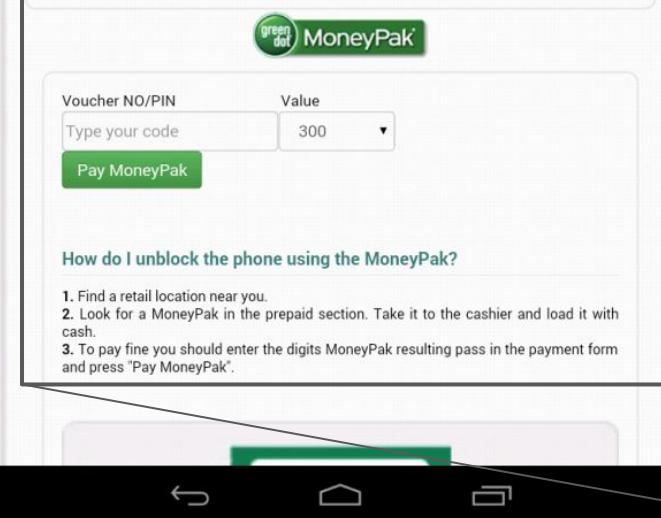
The penalty set must be paid in course of **48** hours as of the breach. On expiration of the term, **48 hours** that follow will be used for automatic collection of data on yourself and your misconduct, and criminal case will be opened against you.

Amount of fine is 300\$. You can settle the fine with MoneyPak xpress Packet vouchers.

As soon as the money arrives to the Treasury account, your phone will be unblocked and all information will be decrypted in course of **24** hours.

Then in **7** day term you should remedy the breaches associated with your phone. Otherwise, your phone will be blocked up and criminal case will be opened against yourself (with no option to pay fine).

Please mind, that you should enter only verified passss of vouchers and abstain from caching out of vouchers once used for fine payment. If erroneous passss were entered, or if attempt was made to cancel vouchers after transaction, then, apart from above breaches, you will be charged with fraud (Article 377 of United States of America criminal law; **1 to 3 years of imprisonment**) and criminal case will be opened.

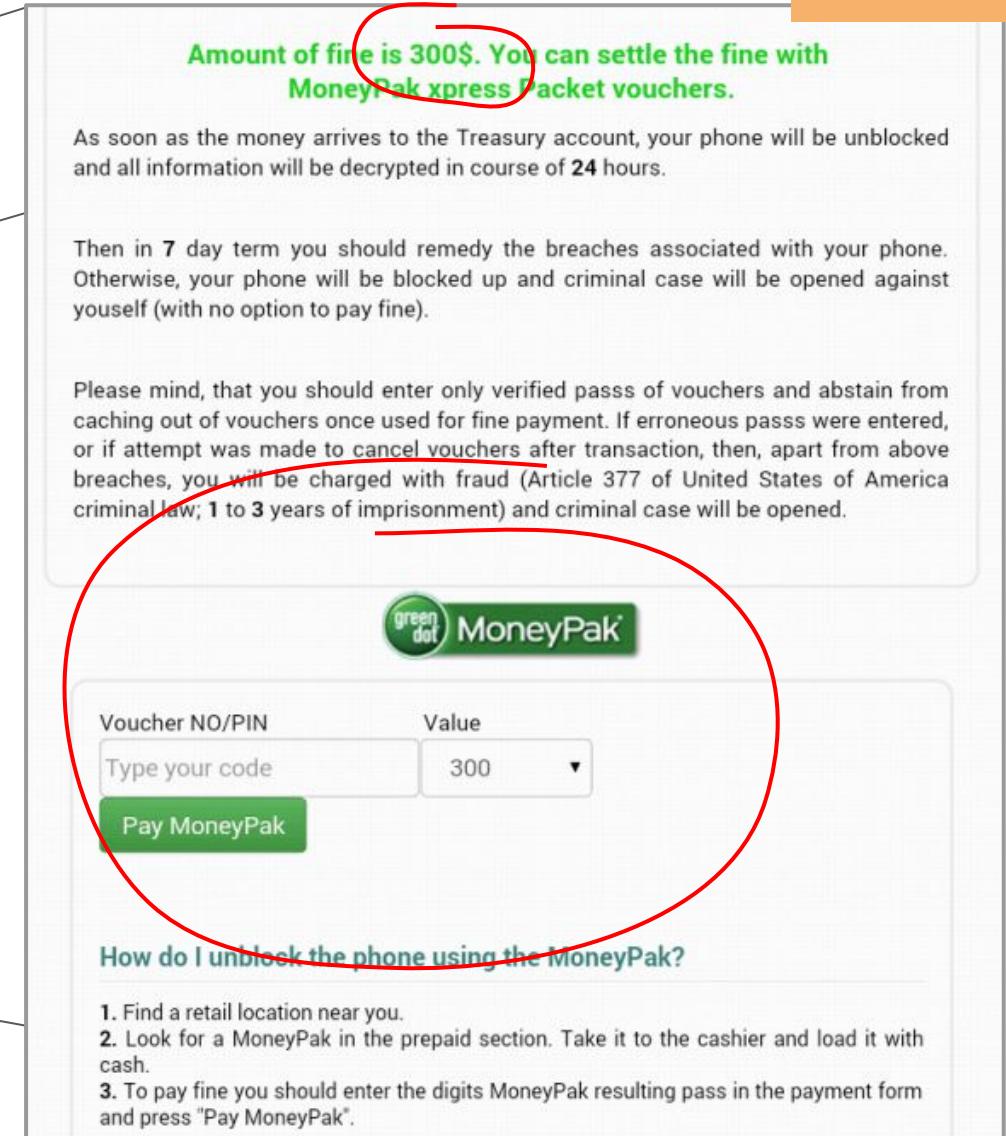


Voucher NO/PIN Value
 Type your code 300

Pay MoneyPak

How do I unlock the phone using the MoneyPak?

1. Find a retail location near you.
2. Look for a MoneyPak in the prepaid section. Take it to the cashier and load it with cash.
3. To pay fine you should enter the digits MoneyPak resulting pass in the payment form and press "Pay MoneyPak".



Voucher NO/PIN Value
 Type your code 300

Pay MoneyPak

How do I unlock the phone using the MoneyPak?

1. Find a retail location near you.
2. Look for a MoneyPak in the prepaid section. Take it to the cashier and load it with cash.
3. To pay fine you should enter the digits MoneyPak resulting pass in the payment form and press "Pay MoneyPak".

Amount of fine is 300\$. You can settle the fine with MoneyPak xpress Packet vouchers.

As soon as the money arrives to the Treasury account, your phone will be unblocked and all information will be decrypted in course of **24** hours.

Then in **7** day term you should remedy the breaches associated with your phone. Otherwise, your phone will be blocked up and criminal case will be opened against yourself (with no option to pay fine).

Please mind, that you should enter only verified passss of vouchers and abstain from caching out of vouchers once used for fine payment. If erroneous passss were entered, or if attempt was made to cancel vouchers after transaction, then, apart from above breaches, you will be charged with fraud (Article 377 of United States of America criminal law; **1 to 3 years of imprisonment**) and criminal case will be opened.



THREATENING
TEXT

→ NLP + ML PIPELINE

- Statically allocated strings & XML resources
 - Parse disassembled code & resources for string variables
- Transmitted via network from the C&C
 - Network trace → ASCII strings
- Rendered on screen (e.g., image, text)
 - Screenshots → OCR → ASCII strings





NATURAL LANGUAGE PROCESSING PIPELINE → ML

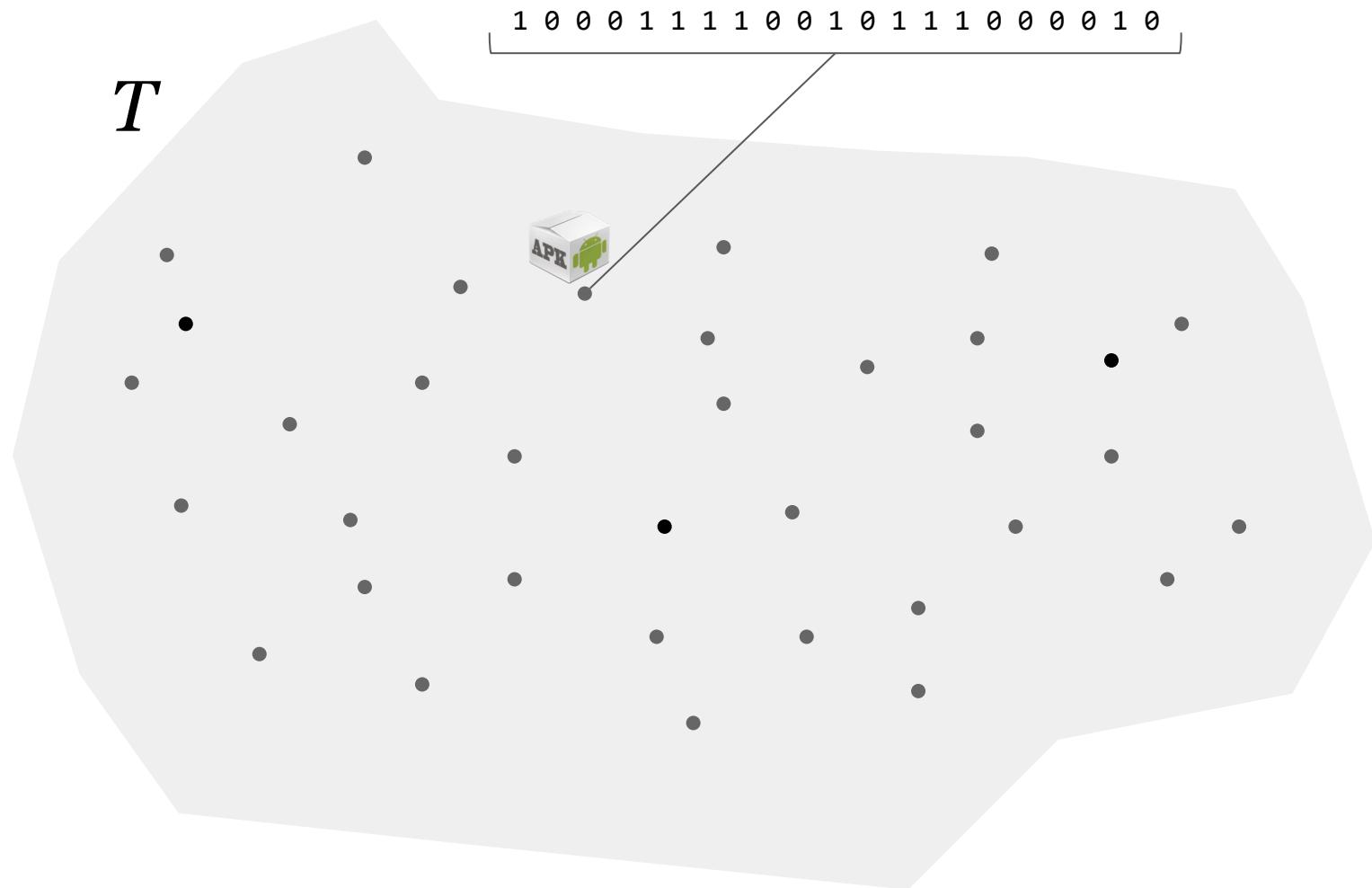
1. Language detection
 - frequency-based analysis (e.g., English, French)
2. Segmentation = Split into sentences
 - "This device has been locked for safety reasons"
 - "All actions performed are fixed"
3. Stop-words removal
 - "This device ~~has been~~ locked ~~for~~ safety reasons"
 - "All actions performed ~~are~~ fixed"
4. Stemming
 - "device ~~locked~~ ~~safety~~ reasons"
 - "actions ~~performed~~ ~~fixed~~"
5. Stem vector

1	0	0	1	1	0	0	1	0	1	0	1	1	0	0	0	0	...
action	:	:	device	fixed	:	:	lock	:	perform	:	...	reason	safe	:	:	:	:



THREATENING
TEXT

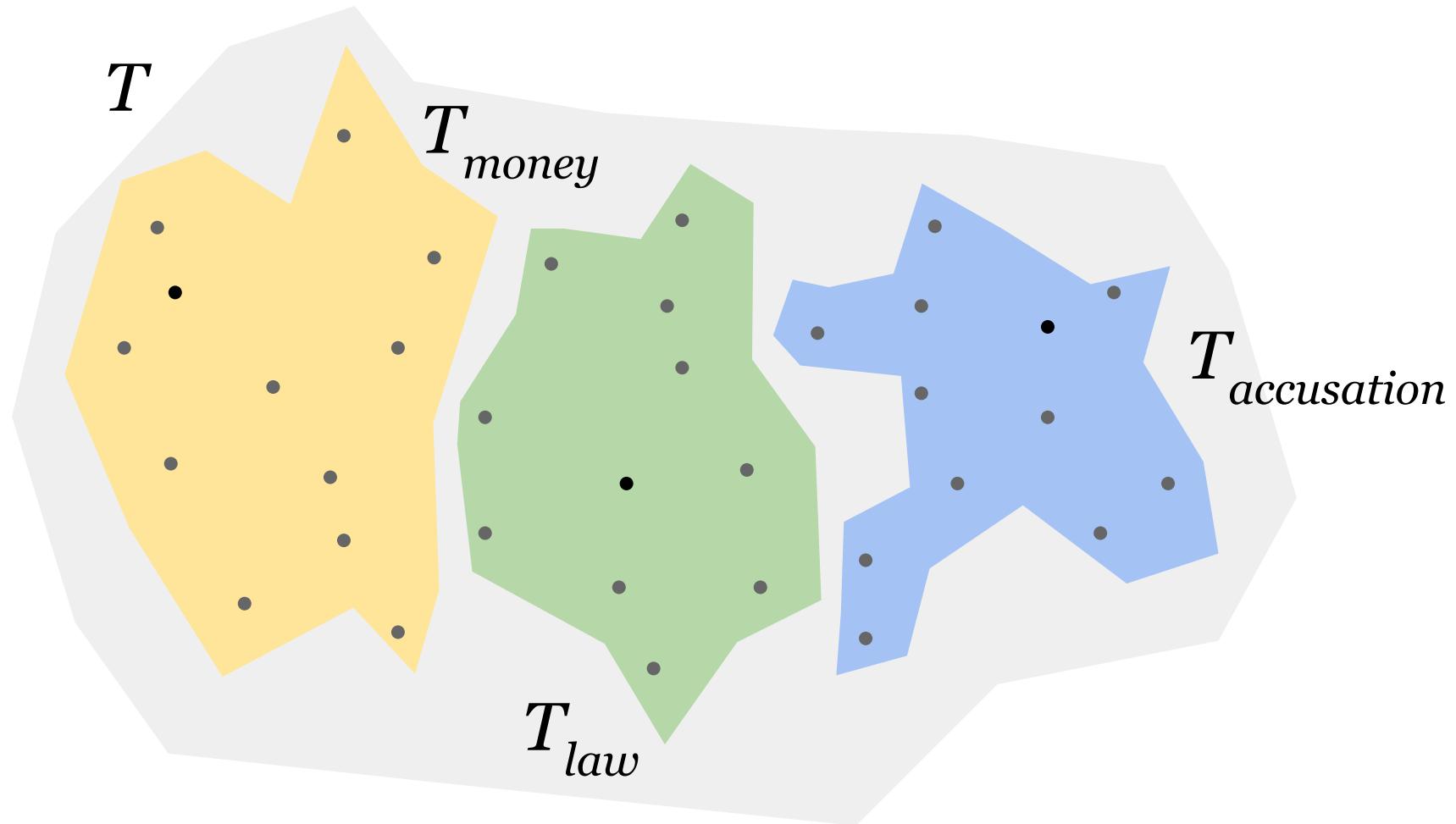
MACHINE LEARNING CLASSIFICATION: TRAINING SET





THREATENING
TEXT

TRAINING SET LABELLING



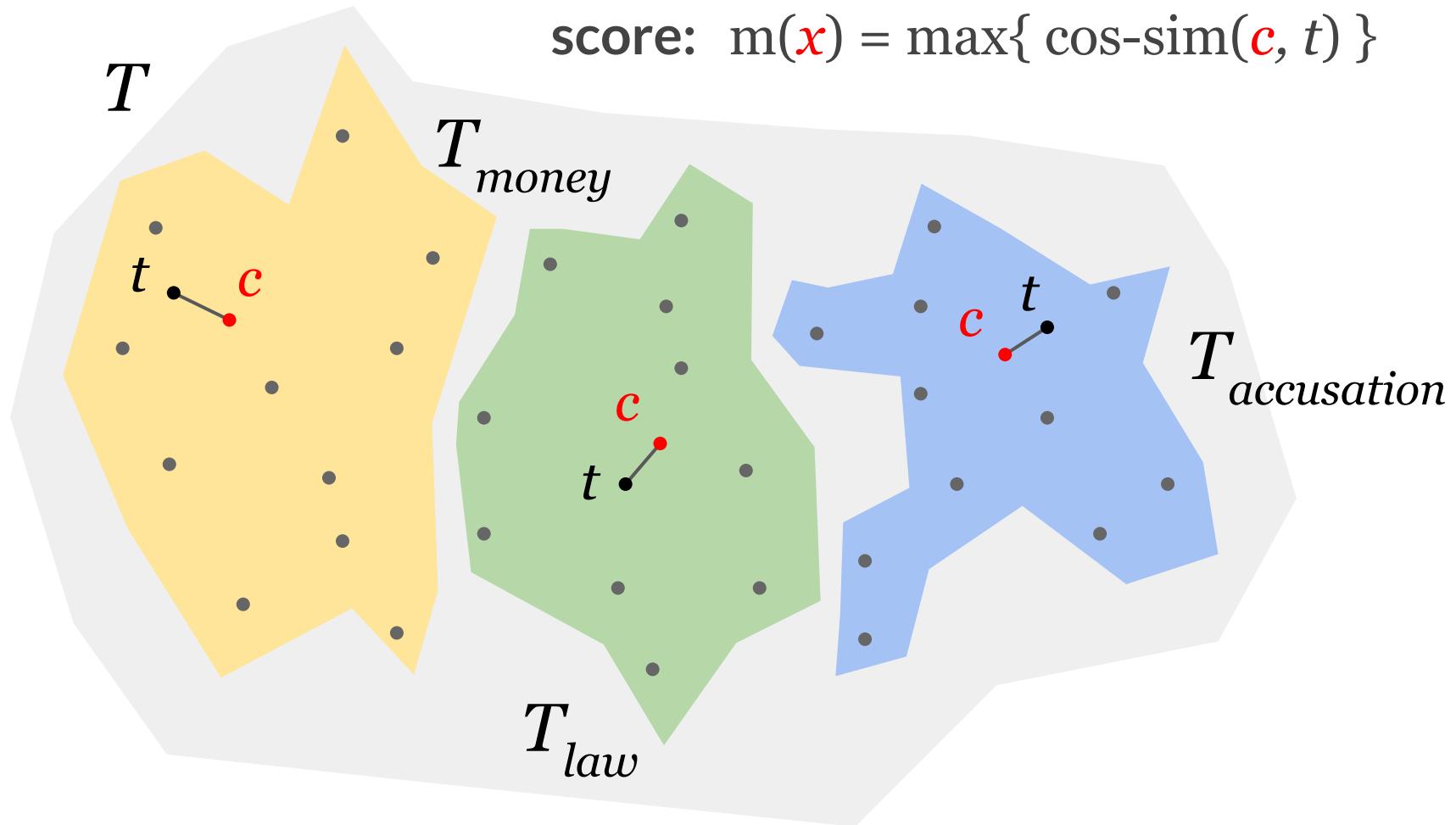


THREATENING
TEXT

SCORING

new text: $x = \{c_1, c_2, \dots, c_n\}$

score: $m(x) = \max\{ \text{cos-sim}(c, t) \}$





SCORING AND DECISION

decision thresholds: minimum to detect known ransomware

if (*best score* in "money")

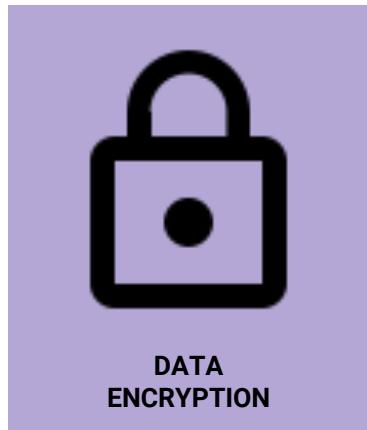
likely **ransomware**

if (*best score* in "accusation" or "law" & ~"money")

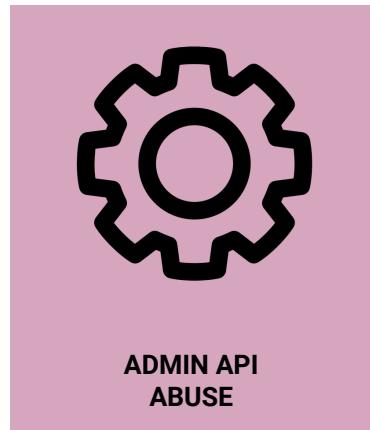
likely **scareware**



THREATENING
TEXT



DATA
ENCRYPTION



ADMIN API
ABUSE



DEVICE LOCKING

(1) Immortal dialogs

- (A) Fill screen with an activity/dialog/window
- (B) Inhibit navigation
 - Hijack `onKeyUp()`/`onKeyDown()`
 - Cover or hide soft keys by using `SYSTEM_ALERT_WINDOW` → draw overlays
 - Fix in Android M makes it more difficult for an attacker, fortunately
- (C) Create non-cancelable dialog
 - Use `setCancelable(false)`

(2) Device admin API abuse

- Call `resetPassword()` and `lockNow()`
 - Upcoming fix in Android N (yay!)



ADMIN API
ABUSE



HIJACK `onKeyUp()`/`onKeyDown()`

- Search code for all `android.app.Activity` (subclasses)
 - that declare `onKeyUp`/`onKeyDown`
- Custom Smali simulator
 - "Execute" all statements
 - Within the scope of `onKeyUp`/`onKeyDown` methods
 - Follow function calls



HIJACK onKeyUp() / onKeyDown()

- Search code for all android.app.Activity (subclasses)
 - that declare onKeyUp/onKeyDown
- Custom Smali simulator
 - Find whether home (0x3) or back (0x4) is targeted
 - Find all possible return values

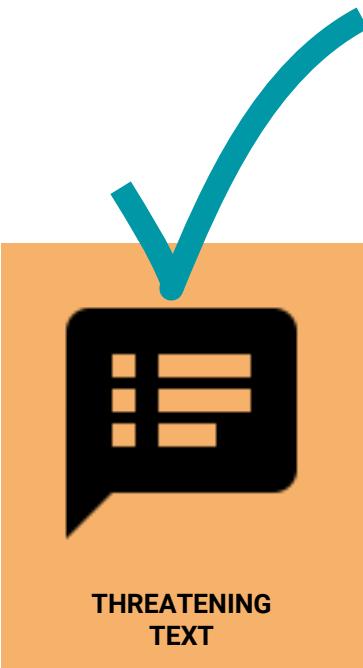
```
.method public onKeyDown(ILandroid/view/KeyEvent;)Z
.locals 1
# p1 = integer with the key code associated to the pressed key.

const/4 v0, 0x4          # 4 = back button
if-ne p1, v0, :cond_0
iget-object v0, p0, Lcom/android/x5a807058/ZActivity;->q:Lcom/android/zics/ZModuleInterface;
if-nez v0, :cond_0
iget-object v0, p0, Lcom/android/x5a807058/ZActivity;->a:Lcom/android/x5a807058/ae;

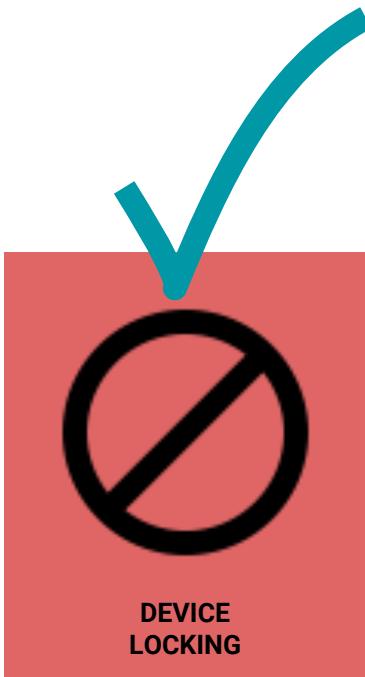
# we track function calls as well
invoke-virtual {v0}, Lcom/android/x5a807058/ae;->c()Z
:cond_0

const/4 v0, 0x1          # True = event handled -> do not forward
return v0
.end method
```

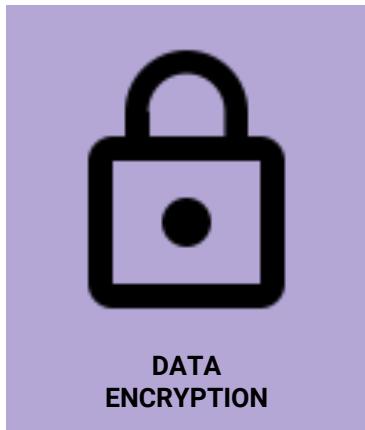




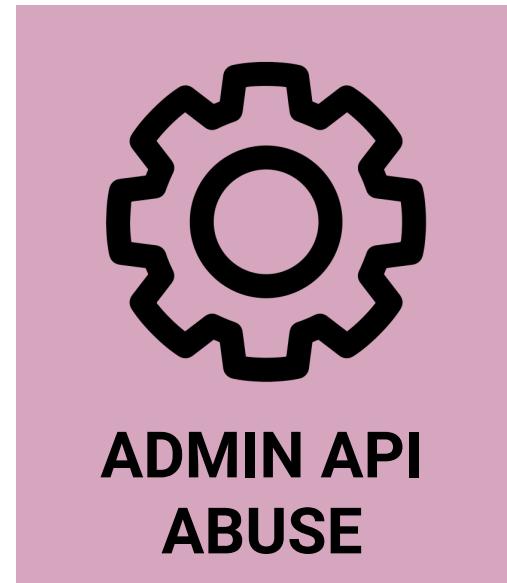
THREATENING
TEXT



DEVICE
LOCKING



DATA
ENCRYPTION



**ADMIN API
ABUSE**



DEVICE ADMIN API ABUSE

```
<activity android:name=".app.DeviceAdminSample"
          android:label="@string/activity_sample_device_admin">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.SAMPLE_CODE" />
    </intent-filter>
</activity>
<receiver android:name=".app.DeviceAdminSample$DeviceAdminSampleReceiver"
          android:label="@string/sample_device_admin"
          android:description="@string/sample_device_admin_description"
          android:permission="android.permission.BIND_DEVICE_ADMIN">
    <meta-data android:name="android.app.device_admin"
              android:resource="@xml/device_admin_sample" />
    <!-- Device Admin XML Configuration -->
    <device-admin xmlns:android="http://schemas.android.com/apk/res/android">
        <uses-policies>
            <limit-password />
            <watch-login />
            <reset-password />
            <force-lock />
            <wipe-data />
            <expire-password />
            <encrypted-storage />
            <disable-camera />
        </uses-policies>
    </device-admin>
```

Look at: [a6dedc5f639b2e1f7101d18c08afc66d](#)



LOOKING AT THE CODE

```
public static void C0010008001(Context arg2, DevicePolicyManager arg3, ComponentName arg4) {  
    C10ll80III3.C0100C(arg2, Boolean.valueOf(false));  
    arg2.stopService(new Intent(arg2, OCI38l1I301.class));  
    if(C10ll80III3.C0I103C3lI8(arg2)) {  
        arg3.removeActiveAdmin(arg4);  
    }  
  
    C11013l3.C1011C80(arg2, arg2.getPackageName());  
}
```

```
public static void C0100C(Context arg1, DevicePolicyManager dpm2) {  
    if(C10ll80III3.C0I103C3lI8(arg1)) {  
        dpm2.lockNow();  
    }  
}
```

Detection? We start from the manifest



MANIFEST → RECEIVER → CFG → REACHABILITY

- Start from the Receiver found in the manifest
- Obtain app's CFG (via FlowDroid)
 - `soot.jimple.infoflow.cfg.SharedCfg.waitForCfg()`
- Calculate all entry points (via FlowDroid)
- Visit CFG breadth first to find calls to `lockNow()` & friends
- If nothing is found, "resolve" reflective calls
 - ...at least, we try to...
 - We "connect" CFG nodes by materializing calls to
`java.lang.reflect.Method.invoke(method)`



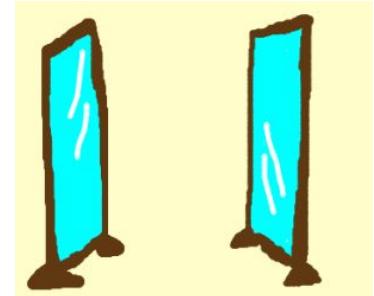
DEALING WITH OBFUSCATED METHOD NAMES

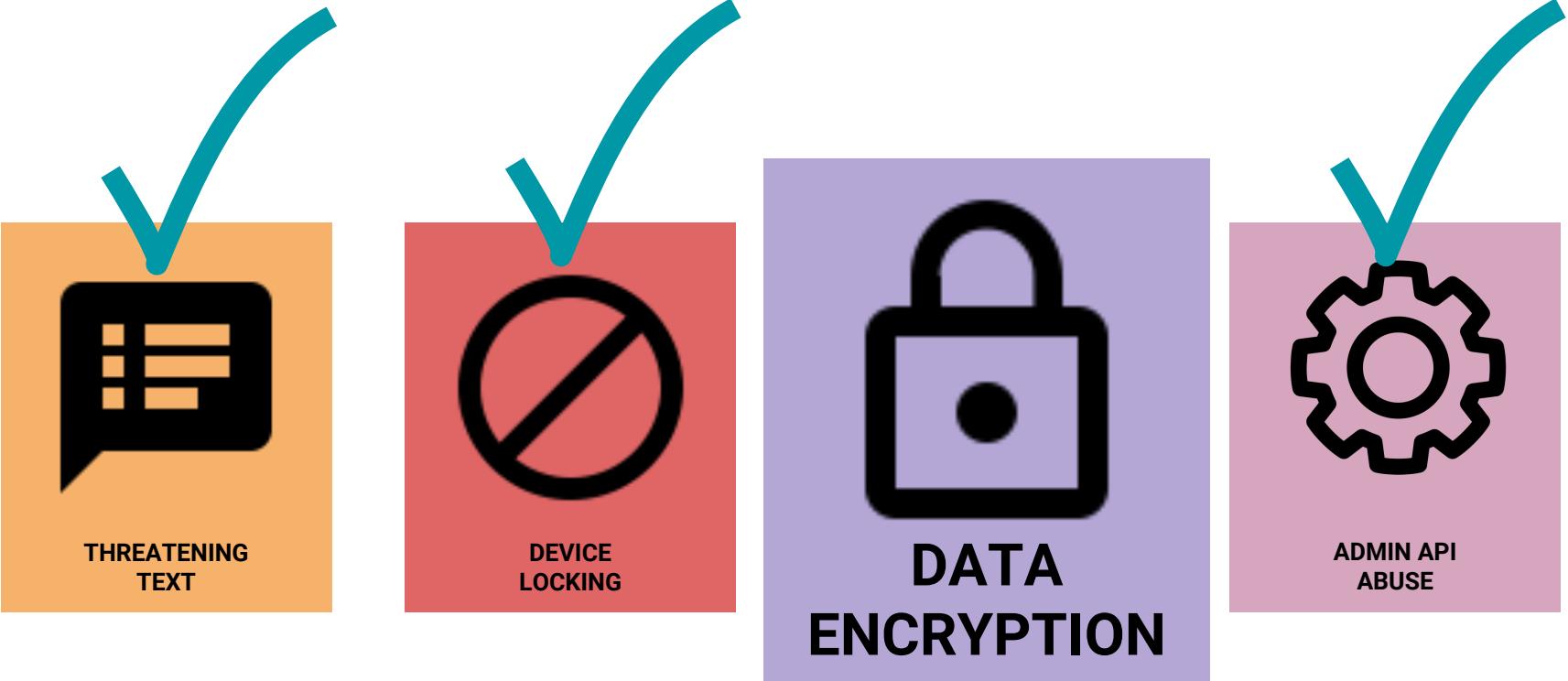
"koOpqUTbcVRhwomXlASpvutejuWHJnQxxaoinoermf"

```
String obfuscated = "koOpqUTbcVRhwomXlASpvutejuWHJnQxxaoinoermf";  
  
String deobfuscated = obfuscated.replaceAll(  
    "[RhwmXlASvutjWHJQxa]",  
    "").substring(10, 14);  
  
Method method = klass.getMethod(  
    deobfuscated,  
    Integer.class  
);  
  
Camera camera = (Camera) method.invoke(cameraId);
```

HOW DO WE DEAL WITH THIS?

```
String obfuscated = "ko0pqUTbcVRhwomXlASpvutejuWHJnQxxaoinoermf";  
6: reflection  
String deobfuscated = obfuscated.replaceAll(  
    "[RhwmXlASvutjWHJQxa]",  
    "#").substring(10, 14);  
4  
Method method = klass.getMethod(  
3 deobfuscated,  
    Integer.class  
);  
1: find call  
Camera camera = (Camera) method.invoke(cameraId);
```





RECAP: ENCRYPTION

```
for (File original : files) {  
    File original = ... // get original file  
    InputStream fis = new FileInputStream(original);  
    while ((nRead = fis.read(buffer, ...)) != 0) {  
        cipher.update(buffer, ...);  
    }  
}
```

- Sources:
 - `java.io.File`: `listFiles()` or `list()`
 - `java.lang.Runtime`: `exec()`
- Sinks:
 - `javax.crypto.Cipher`: `doFinal()`
 - `javax.crypto.CipherOutputStream()`

Ah, of course, no UI initiated!



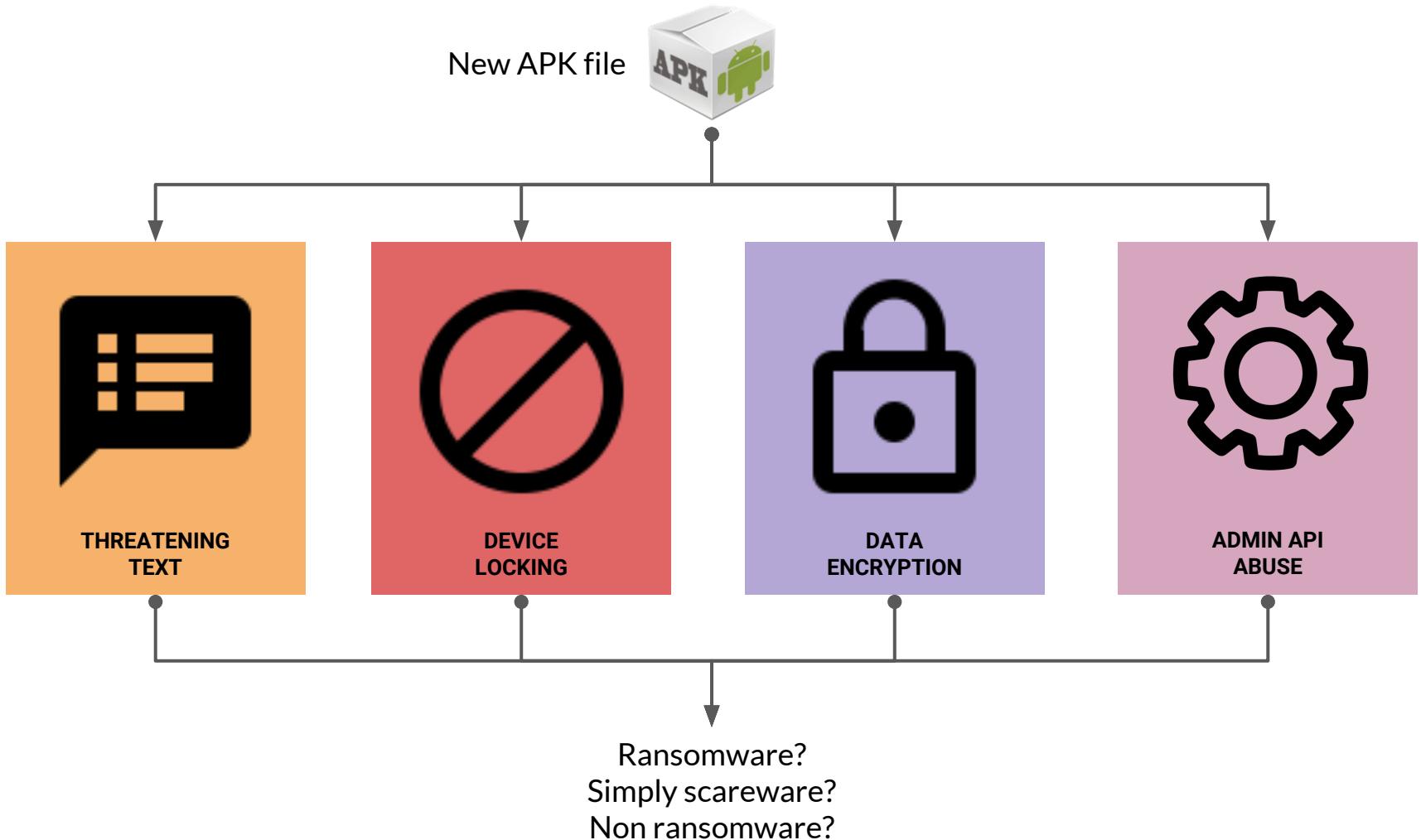
FIND STATIC ENCRYPTION FLOWS → FLOWDROID

```
for (File original : files) {  
    File original = ... // get original file  
    InputStream fis = new FileInputStream(original);  
    while ((nRead = fis.read(buffer, ...)) != 0) {  
        cipher.update(buffer, ...);  
    }  
}
```

file pointer → byte[]
byte[] → cipher's update

Not propagated by default, but very
efficient to do

GREAT! WE HAVE A PIPELINE!

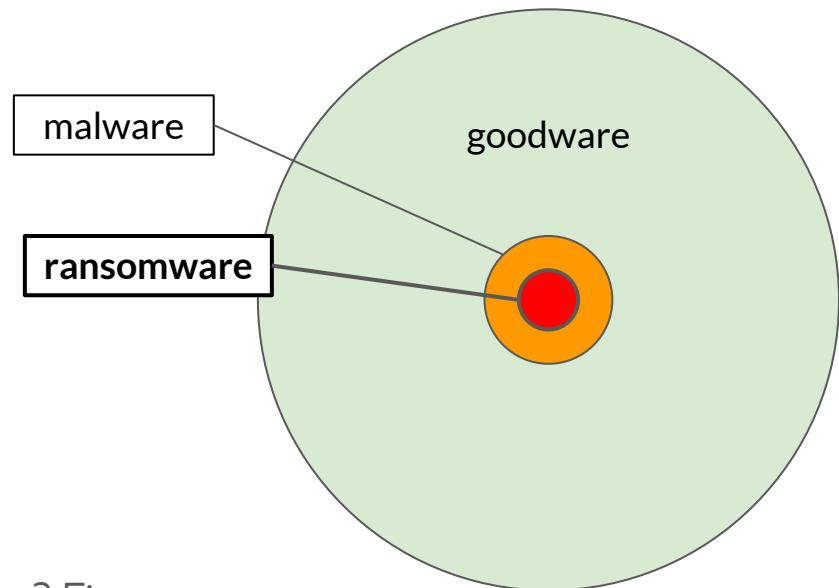


BUT IT TAKES TIME



12 SECS. VS. MANUAL RE IS OK. BUT...

- 12 secs per thousands of samples/day is not fast
 - Remember, we want to run this at the app-store scale
- Pre-filtering to the rescue!
- Design principles
 - Very fast but very precise
 - Confuse a benign app as a suspicious one? Fine.
 - **Confuse a malicious app as a benign one. Hell...no!**



MACHINE LEARNING CLASSIFICATION APPROACH

- Design principle → Cost-sensitive classifiers

$\text{Cost}(\text{False negatives}) \ll \text{Cost}(\text{False positives})$

$\text{Cost}(\text{benign confused as malicious}) \ll \text{Cost}(\text{malicious confused as benign})$

about 15 times

- Ensemble classifier with majority voting between
 - J48 decision tree
 - Random forest
 - Decision table

FEATURES

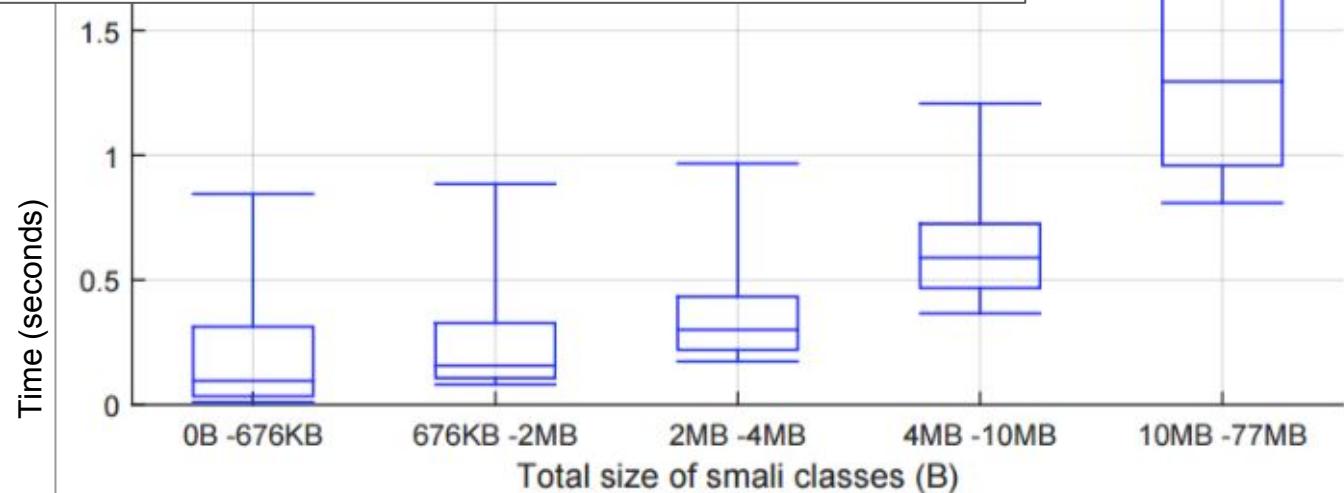
- Lot's of research on that
- We borrow some features, *not the way to use them*
 - Permissions (boolean)
 - yes/no array of requested Android permissions
 - File Statistics (numeric, novel)
 - File size
 - Number of permissions
 - Number of services, activities, receivers, avg. class size, number of packages
 - Lightweight behavioral features (boolean, novel)
 - Send SMS or reads phone info `onStartup()`
 - Calls native utils (e.g., `ls`, `grep`, `root`, `chmod`)
 - ...
 - Other features (boolean, novel)
 - is package name composed by a single part?
 - does the reversed package name match a real domain name?

FAST AND PRECISE!

$\sim 12 \text{ SECS} \rightarrow \sim 1.25 \text{ SECS}$

Classifier(s) Accuracy Precision AUC

J48	93.74%	99.4%	0.979
SGD	90.90%	98.9%	0.916
Decision Table	91.83%	99.5%	0.986
Random Forests	87.18%	99.6%	0.991
J48 + DT + RF	92.75%	99.6%	0.934
J49 + DT + SGD	93.75%	99.6%	0.956
SGD + DT + RF	91.29%	99.6%	0.941



Sample: [39780965255168083534b596c9b28c4e3e99e85decc3ed1f091f11d92eb7159d](#)

EXPERIMENTS



DATASETS



virus**total**

SOURCE	SIZE	USE
AndRadar	172,174	Malware + Goodware (false positive eval.)
AndroTotal.org	12,842	
Malware Genome	1,260	
Generic	1,239	

Known ransomware	207	Text-analysis training (manually vetted)
Unseen ransomware	443	Detection evaluation

Ransomware daily feed ~38,425 (and counting)

RESULTS

- False positives (~0.07%)
 - Corner case: large portions of law- or copyright-related text (EULA)
 - 6 benign apps used to extensively modify the UI
 - 1 adware app
- Detection rate (~99%)
 - 49 samples turned out to be mislabeled
 - 4 were actually ransomware, but somewhat disarmed/not working
 - **11 language unsupported (Spanish, Russian)**
 - Extended the language classifier right away (30 minutes manual work)
 - All the rest was correctly classified

WAIT WAIT...
THE TOOL, THE TOOL!



HELDROID: SOURCE CODE RELEASE!

- [new!] <http://github.com/necst/heldroid>
- REST API <http://ransom.mobi>
- Run daily on VT feed of Android ransomware samples
 - About ~38,500 samples so far
 - Filterable/sortable tables by detected feature
 - Downloadable reports

<http://ransom.mobi/scans/>

Ransomware report

ransom.mobi/scans/

Statistics Koler Locker Lockerpin Porndroid Reveton Scare Scarepackage Simplocker Slocker Svpeng Fusob

Small

Languages

A pie chart titled "Languages" showing the distribution of three languages. The largest slice is English (red), followed by Spanish (purple), and the smallest slice is Russian (teal). The chart is labeled with "russian", "english", and "spanish" pointing to their respective slices.

●russian ●english ●spanish

Show 10 entries Search: CanvasJS.com

Family	LockDetected	TextDetected	Languages	Has RW permission	EncryptionDetected	DeviceAdminUsed	TimedOut	Total
fusob	2569	6	{"english": 6, "spanish": 3}	2567	0	1598	6	2569
koler	3429	611	{"russian": 48, "spanish": 332, "english": 2}	3172	2	2979	494	3640

Ransomware report

ransom.mobi/scans/

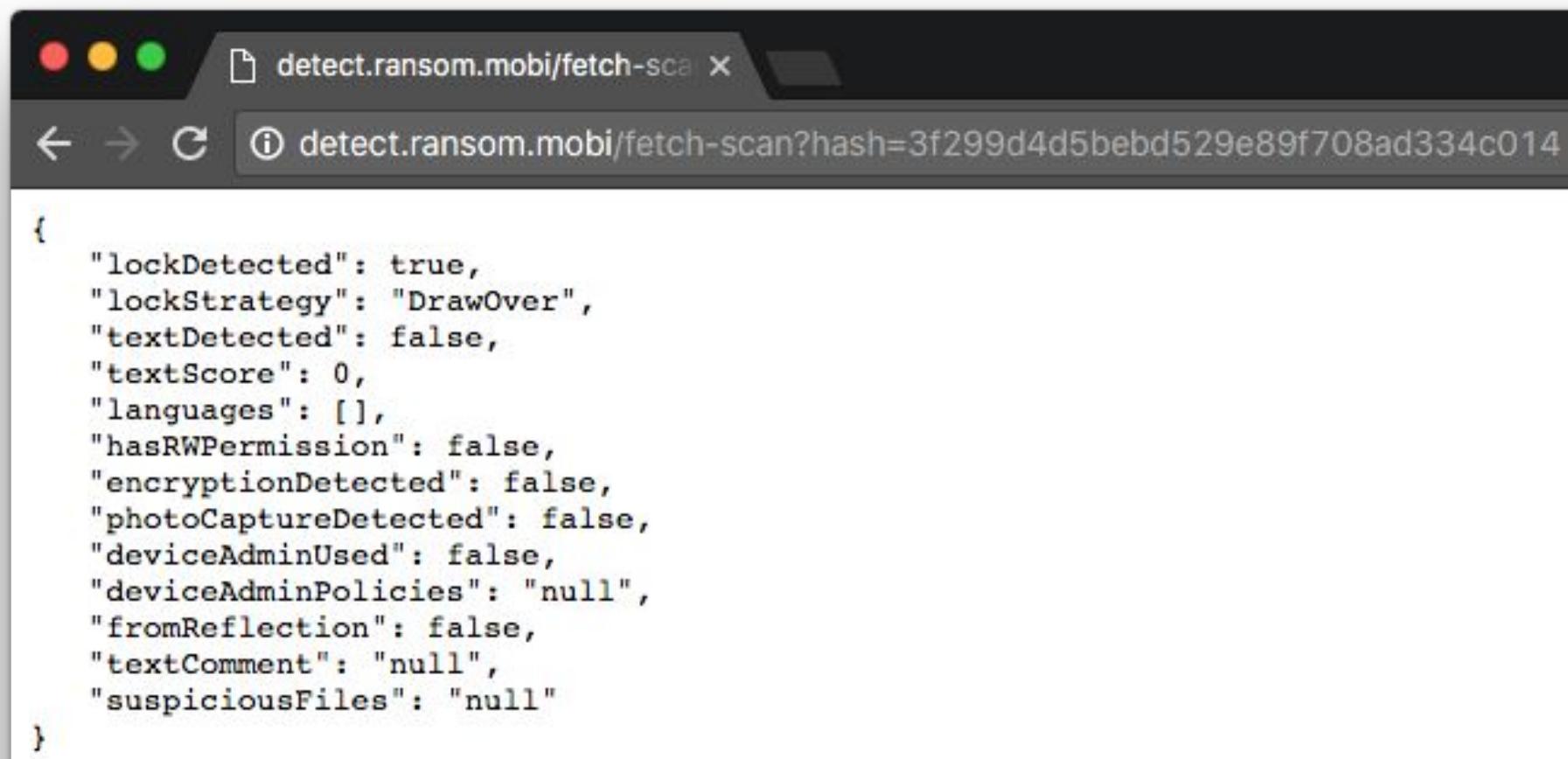
Slocker

This family is the same as Simlocker. They will be merged soon.

Show 10 ↑ entries

Search:

Sample	LockDetected	LockStrategy	TextDetected	TextScore	Languages	RV	Pe
00028b31d9e5971438f522f63335b9389a30a565231203da82c23a97ada1465f	true	DrawOver	false	0.000000	null	fa	
	VT Report						
	Analysis result						
	Download Analysis result						
000ddbac36cf76dea6a4354a00605761c3a66ead640ae91814c13be4fa7e4a45	false	null	false	0.000000	null	fa	
	VT Report						
	Analysis result						
	Download Analysis result						
0014ae26ece1eebda2179e6ced67728c709df215b4b25ef03ba98c792e4723fb	true	DrawOver	false	0.000000	[english]	tru	
	VT Report						
	Analysis result						
	Download Analysis result						
0017108fb58a1cebf704bc80569dea891916dc6402dfe1839efa6dc43c712f5e	true	DrawOver	true	1.000000	[spanish, english]	tru	
	VT Report						
	Analysis result						
	Download Analysis result						



A screenshot of a web browser window. The title bar shows the URL "detect.ransom.mobi/fetch-sca" and the address bar shows "detect.ransom.mobi/fetch-scan?hash=3f299d4d5bebd529e89f708ad334c014". The main content area displays a JSON object:

```
{  
  "lockDetected": true,  
  "lockStrategy": "DrawOver",  
  "textDetected": false,  
  "textScore": 0,  
  "languages": [],  
  "hasRWPermission": false,  
  "encryptionDetected": false,  
  "photoCaptureDetected": false,  
  "deviceAdminUsed": false,  
  "deviceAdminPolicies": "null",  
  "fromReflection": false,  
  "textComment": "null",  
  "suspiciousFiles": "null"  
}
```

overall idea + implementation

HELDROID: Dissecting and Detecting Mobile Ransomware

Nicoló Andronio, Stefano Zanero, and Federico Maggi^(✉)
DEIB, Politecnico di Milano, Milano, Italy
nicolo.andronio@mail.polimi.it,
{stefano.zanero,federico.maggi}@polimi.it

Abstract. In ransomware attacks, the actual target is the human, as opposed to the classic attacks that abuse the infected devices (e.g., botnet remote control). This makes mobile ransomware attacks more dangerous than ransomware attacks. However, there is little research work on this matter and only traditional protections are available. Even state-of-the-art mobile security tools have difficulties detecting ransomware apps because of the subtle attack schemes. As a consequence, the ample attack surface formed by the billion mobile devices is left unprotected.

First, we present the HELDROID system for automatically dissecting mobile ransomware families, describing their common characteristics. Second, we present HELDROID, a fast, efficient and fully automated approach for detecting ransomware attacks on mobile devices using ransom samples from goodware. Our approach is based on detecting the “building blocks” that are typically used in modern ransomware applications. Specifically, HELDROID detects, in a generic way, if an app is attempting to lock or encrypt the device without the user’s consent, and to do so it does not need to be signed. HELDROID also works without requiring that a sample of a certain family is available beforehand.

We implemented HELDROID and tested it on real-world Android ransomware samples. On a large dataset comprising hundreds of thousands of APKs including goodware, malware, scareware, and ransomware, HELDROID exhibited nearly zero false positives and the capability of recognizing unknown ransomware samples.

1 Introduction

Theorised back in 1996 [1], ransomware attacks have now become a reality. A typical ransomware encrypts the files on the victim’s device and asks for a payment to release them. The most common implementation of extortion tactics (as explained in [2]) is to lock the screen, which are both slow and extremely effective. In the best case, the device is locked but personal data is actually left in place in untouched; in the worst case, personal data is effectively encrypted. Therefore, even if the malware is somehow removed, in absence of a fresh backup, the victims have no other choice but to pay the ransom in order to regain access to their data. McAfee Labs [2] and the FBI [3] recently concluded that the ransomware trend

© Springer International Publishing Switzerland 2015
R. Kruegel et al. (Eds.), ADVANCED SWEET 2014, pp. 382–404, 2015.
DOI: 10.1007/978-3-319-20362-5_18.

HELDROID: Dissecting and Detecting Mobile Ransomware

Nicoló Andronio, Stefano Zanero, and Federico Maggi^(✉)

DEIB, Politecnico di Milano, Milano, Italy
nicolo.andronio@mail.polimi.it,
{stefano.zanero,federico.maggi}@polimi.it

GreatEaton: Fast, Static Detection of Mobile Ransomware

Chengyu Zheng, Nicola Dellarocca, Niccolò Andronio, Stefano Zanero, and Federico Maggi
DEIB, Politecnico di Milano, Italy
Email: {name.surname}@polimi.it

Abstract

Ransomware is a class of malware that aim at preventing victims from accessing valuable data, typically via data encryption or device locking, and ask for a payment to release the target. In the past year, instances of ransomware attacks have been spotted on mobile devices too. However, despite their relatively low infection rate, we notice that the techniques used by mobile ransomware are quite sophisticated, and different from those used by ransomware against traditional computers.

Through an in-depth analysis of about 100 samples of currently active ransomware apps, we conclude that most of them pass undetected by state-of-the-art scanners, which are based on signature-based detection and incomplete set of features. The most notable examples are the abuse of reflection and device-administration APIs, appearing in modern ransomware to evade analysis and detection, and to elevate their privileges (e.g., to lock or wipe the device). Moreover, current approaches are unable to detect ransomware because they detect cryptographic APIs abuse, flagging goodware signs as ransomware merely because they rely on cryptographic libraries. Last but not least, the performance overhead of current approaches is unacceptable for appstore-scale workloads.

In this work, we tackle the aforementioned challenges by proposing GreatEaton, a novel static detection system for mobile ransomware. We focus GreatEaton deployed on the appstore side, as a preventive countermeasure. At its core, GreatEaton uses static program-analysis techniques to “resolve” reflection-based, and analysis attempts, to detect the absence of the device-administration API, and extract metadata and diverse information to detect truly malicious uses of cryptographic APIs. Given the significant resources utilized by GreatEaton, we prepend to its core a fast pre-filter that quickly discards obvious goodware, in order to avoid wasting compute cycles.

Our evaluation shows that GreatEaton is able to detect thousands of goodware, generic malware and ransomware applications, and showed that it surpasses current approaches both in speed and detection capabilities, while keeping the false negative rate below 1.3%.

Pre-filter + enhancements

GreatEaton: Fast, Static Detection of Mobile Ransomware

Chengyu Zheng, Nicola Dellarocca, Niccolò Andronio, Stefano Zanero, and Federico Maggi

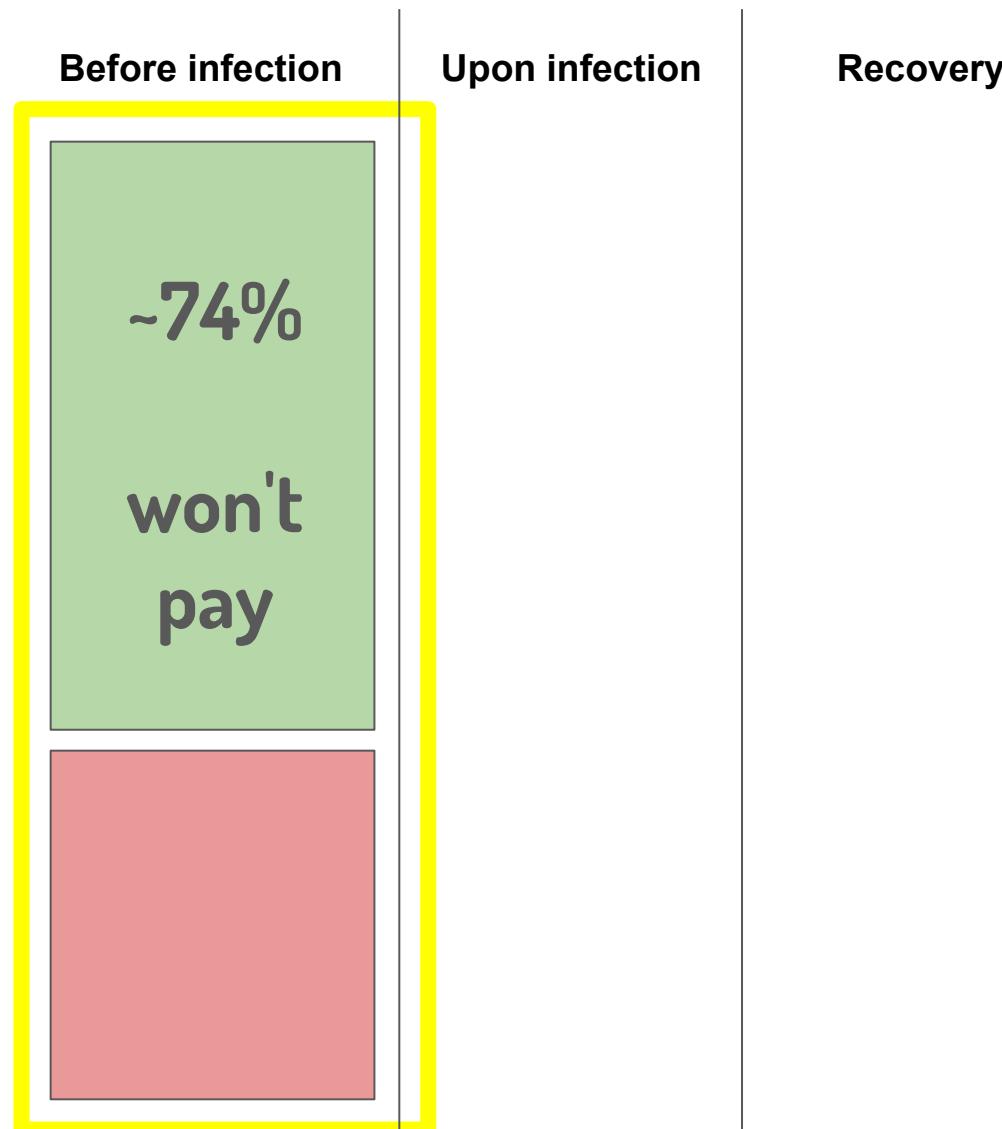
DEIB, Politecnico di Milano, Italy
Email: {name.surname}@polimi.it

~~MOBILE RANSOMWARE~~ → MY PERSONAL TAKE:
AN "ECONOMICS" PERSPECTIVE ON RANSOMWARE

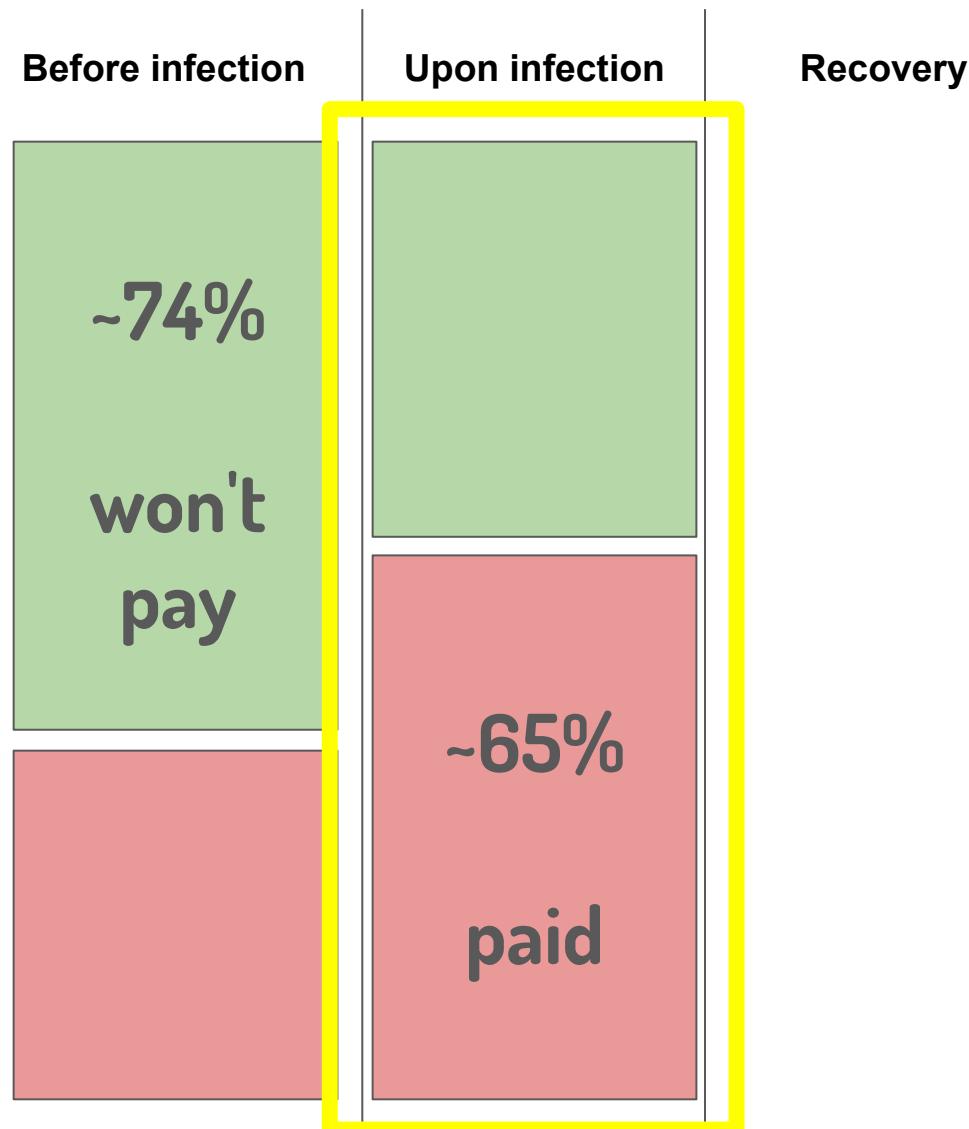
FROM A SURVEY (RUN BY TREND MICRO IN UK)

Before infection	Upon infection	Recovery

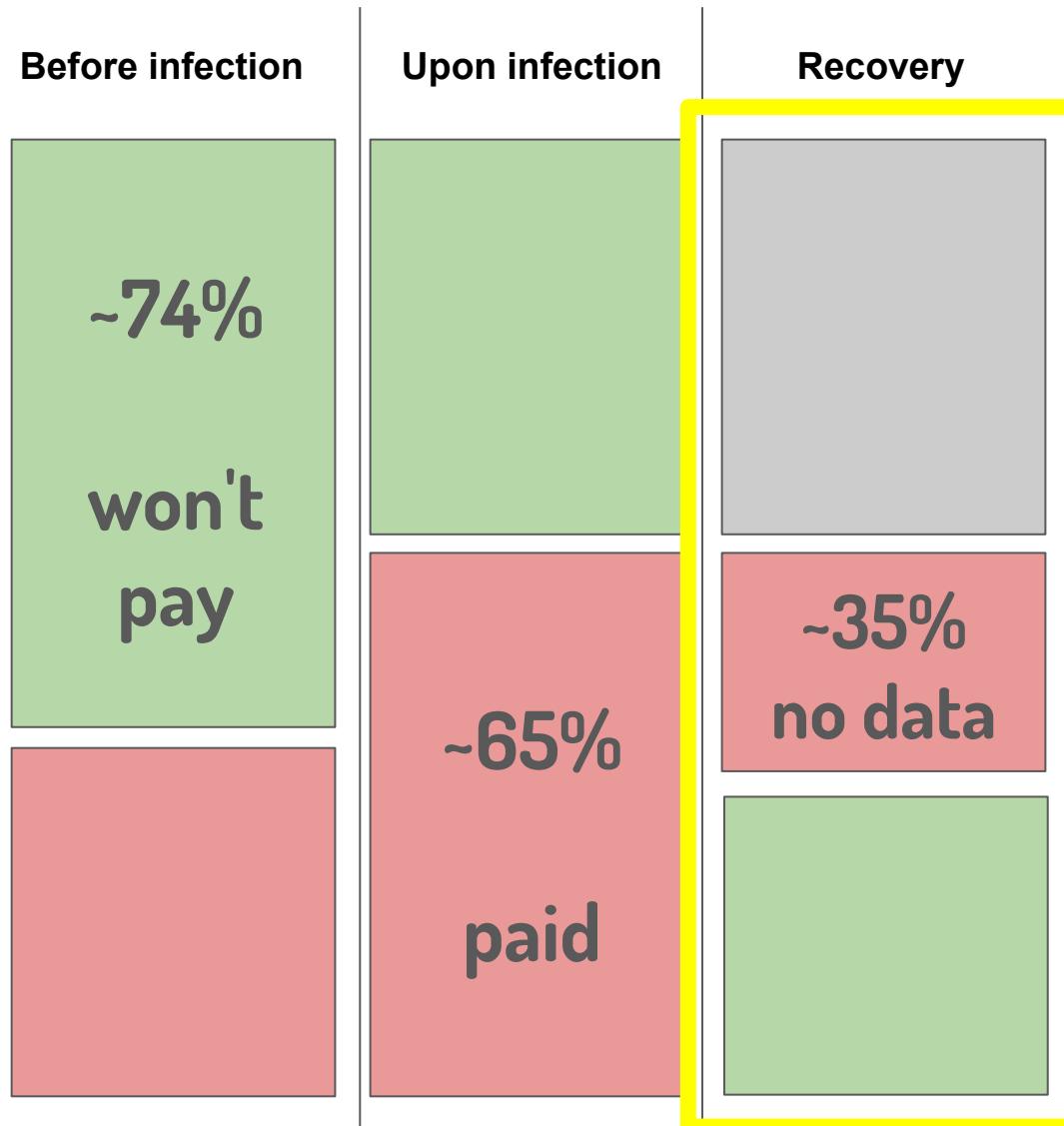
FROM A SURVEY (RUN BY TREND MICRO IN UK)



FROM A SURVEY (RUN BY TREND MICRO IN UK)



FROM A SURVEY (RUN BY TREND MICRO IN UK)



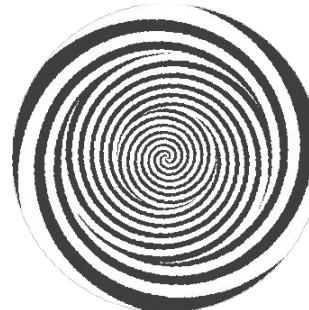
WHY RANSOMWARE IS DIFFERENT

- 1: It is destructive – Changes everything
 - It *has to be!*
 - No show → No gold
- 2: "Pure profit" business (very modest costs)
 - More profit than 68.7% of the businesses in the Forbes top 2000
- 3: Business model backed by honesty
 - Honest attacker → successful business
 - Non negligible fraction of dishonest attackers

MORE SURRENDERS → MORE PAYMENTS

→ **Vicious circle**

- Victims pay → Attackers expectation increases
- Attackers gain confidence in the business → Prices increase



Secret sauce = how much can the victim afford?

VICIOUS CIRCLE AMPLIFIERS: INSURANCES

"[...] we'll reimburse [...] up to \$1000 per endpoint, or \$1,000,000 in protection overall for the company. Guaranteed."

(source: easy to find)

- Thanks! The crooks are *super happy* now!
- All they need to do is target all customers of this vendor, and get to ~1000 endpoints, asking \$1,000,000 overall

VICIOUS CIRCLE AMPLIFIERS: INSURANCES

"It's time for security companies to back their technology and provide users with the financial assurance they deserve against ransomware attacks."

No **seriously**? Think for a minute about the *global*, long-term effect of this idea...

THANK YOU

Why Ransomware Comes as a Plot Twist in the Cat-Mouse Game

Federico 'phretor' Maggi

~
@phretor - <http://maggi.cc>

&&
Stefano 'raistlin' Zanero

Nicolò Andronio

Nicola della Rocca

Chengyu Zheng

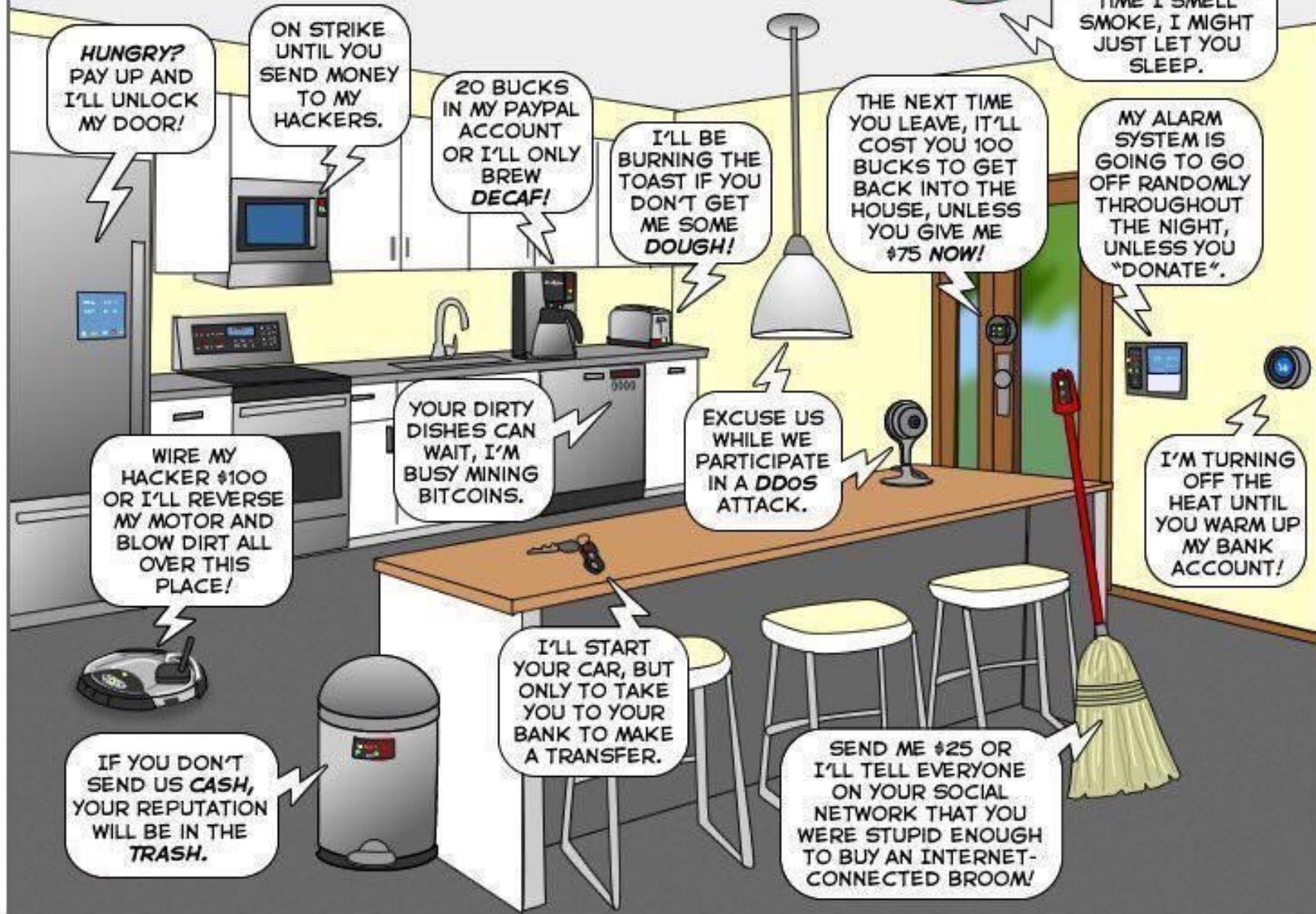


POLITECNICO
MILANO 1863



EXTRA SLIDES

The Internet of ransomware things...



The Internet of ransomware?!

HUNGRY?
PAY UP AND
I'LL UNLOCK
MY DOOR!

ON STRIKE
UNTIL
SEN

30 BUCKS IN
BITCOIN, OR NEXT
TIME I SMELL
SMOKE, I MIGHT
JUST LET YOU
SLEEP.

MY ALARM
SYSTEM IS
GOING TO GO
OFF RANDOMLY
THROUGHOUT
THE NIGHT,
UNLESS YOU
DONATE".

I'M TURNING
OFF THE
HEAT UNTIL
YOU WARM UP
MY BANK
ACCOUNT!

WE
HACK
OR I'L
MY MO
BLOW
OVER
PLAY

IF YOU DON'T
SEND US CASH,
YOUR REPUTATION
WILL BE IN THE
TRASH.

Mobile Applications: a Backdoor into Internet of Things?

Axelle Apvrille - FortiGuard Labs, Fortinet

