

# Smartphone Forensics

Dr. Michael Spreitzenbarth



SECURITY SAYS  
YOUR EMPLOYEE  
LOCATOR DEVICE  
ISN'T TURNED ON.

MY  
WHAT?

I THINK  
YOU CALL IT  
A SMART-  
PHONE.

I MIGHT  
HAVE SOME  
QUESTIONS.

PUT THEM  
IN A TEXT  
TO YOUR-  
SELF. I'LL  
READ THEM  
LATER.

# Agenda and Dates



# Agenda

- 2018-05-11: Mobile Device Forensics
- 2018-05-25: Android and Forensic Investigation of this OS
- 2018-06-01: Apple iOS and Forensic Investigation of this OS
- **2018-06-08: Mobile Malware & Hacking Android Apps**



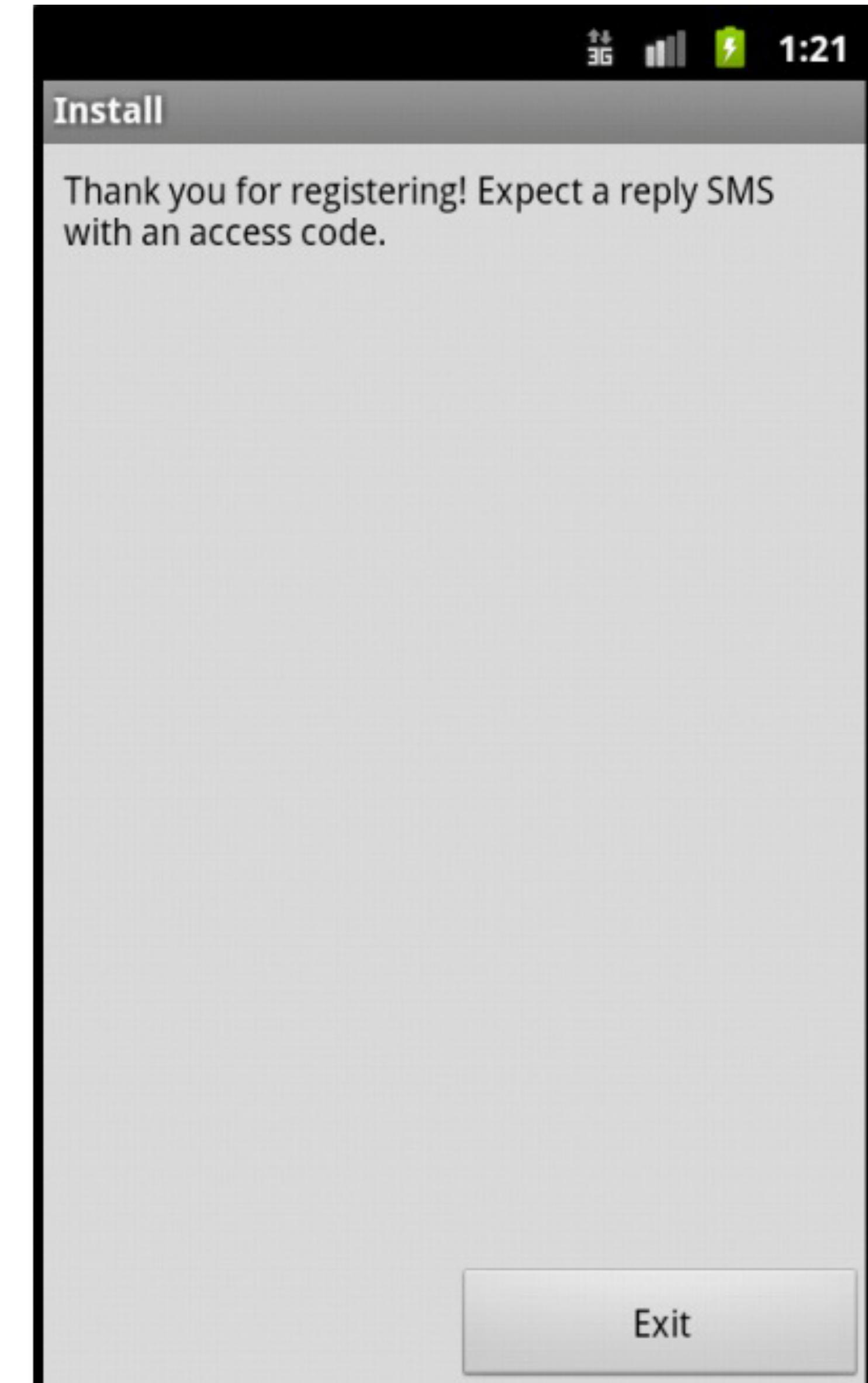
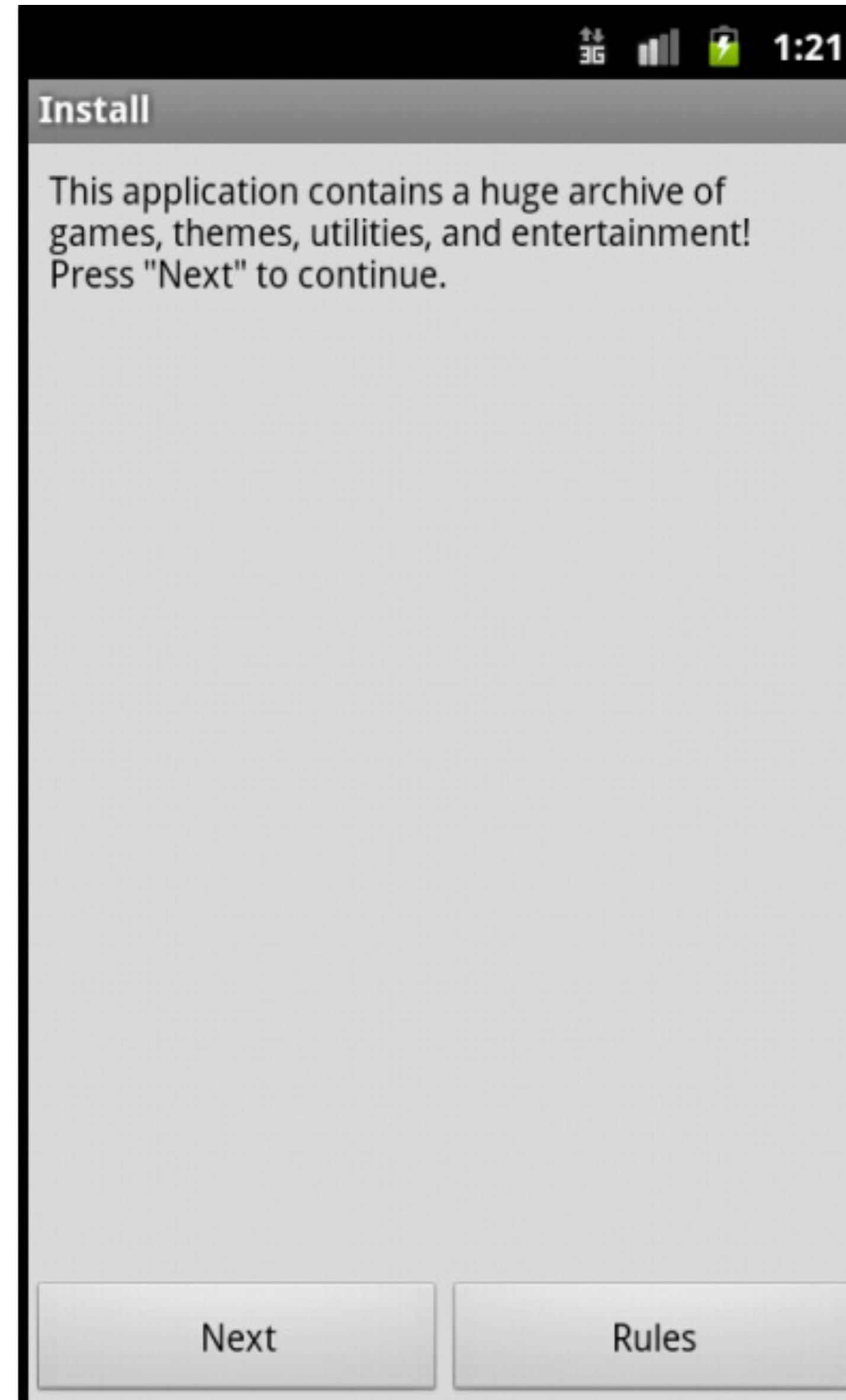
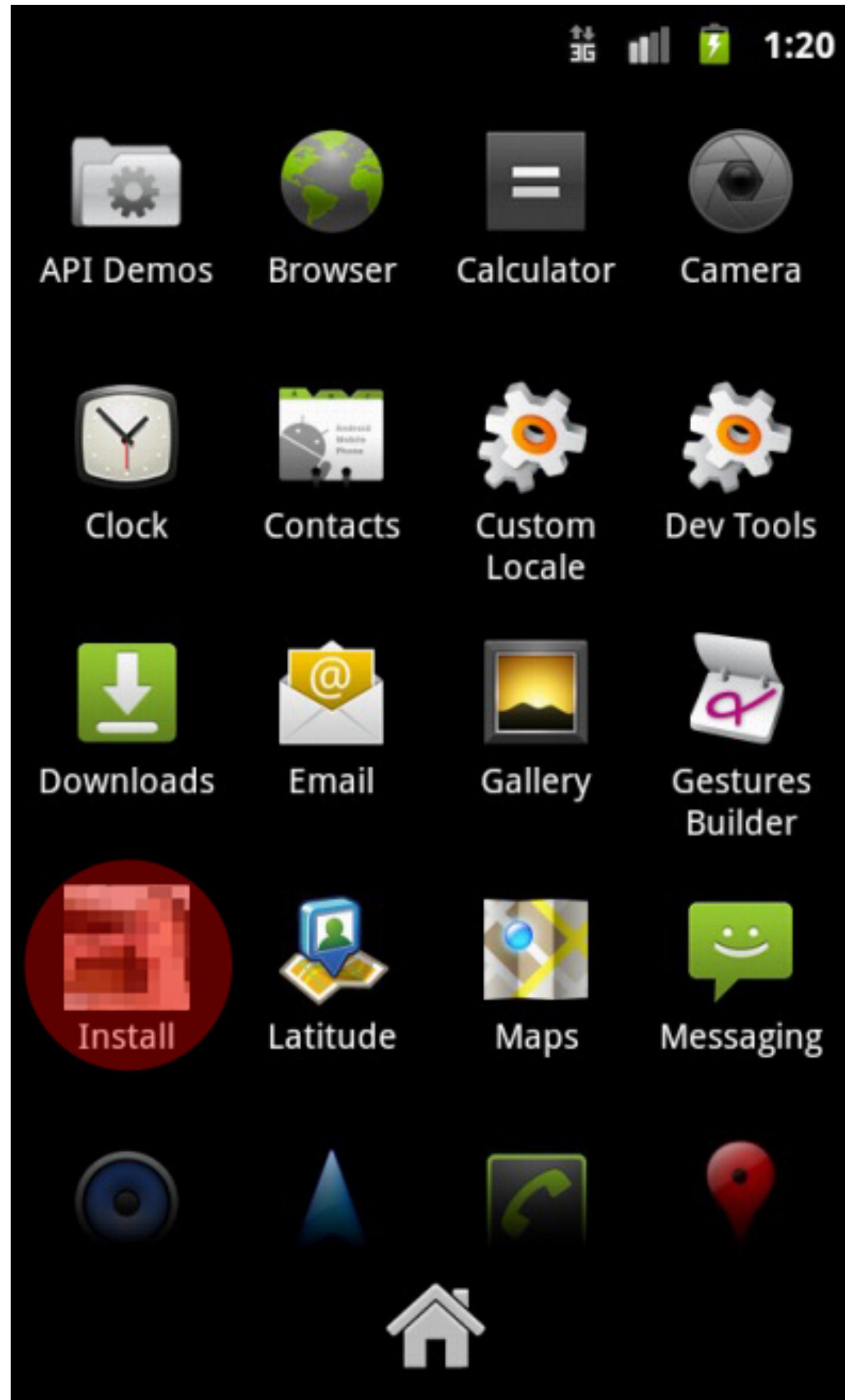
# Mobile Malware



# Types of Mobile Malware

PUA	Fraudware	Malware
often called „hacking tools“	trick fraud apps	clearly malicious
can be dangerous for the device integrity and the user	apps send expensive SMS messages	is hiding its actions
will often be used for pentesting purposes	within the general terms and conditions it is often stated what the app is doing	often also called trojan or virus
can be used to monitor traffic on the smartphone		many versions/families steal personal data
		latest versions also have the capability to encrypt phones

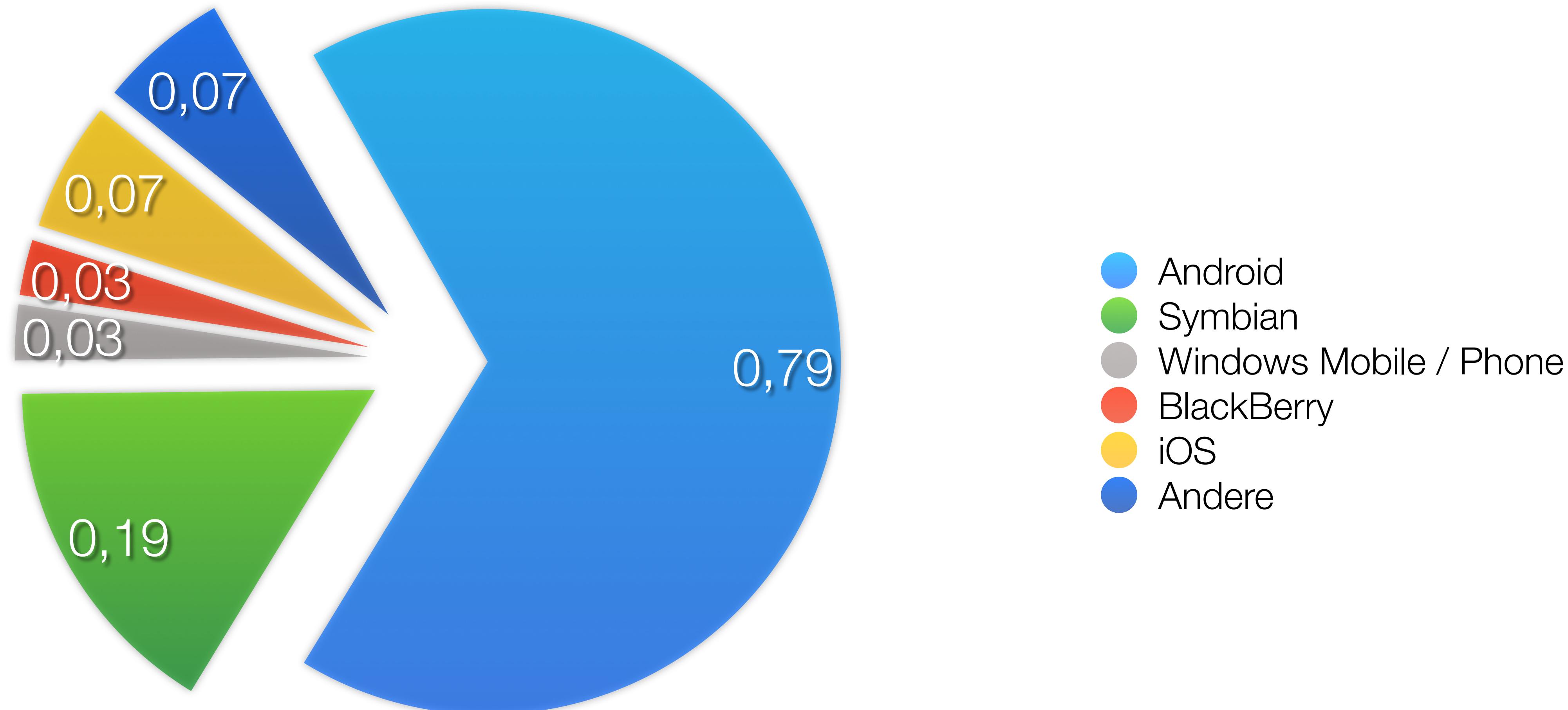
# Fraudware



# Threats and Distribution

Distribution	Threats
Using unofficial app stores	Data espionage
Piggyback on legitimate apps	Sending premium SMS messages
Upgrades of legitimate apps	Calling premium numbers
Social-Engineering / Phishing	Sending SPAM-Emails or -SMS messages
Through side-loading from malicious websites or through ad-networks	Gaining root access to the device Connecting the device to a malicious botnet
	Tracking a device through GPS

# Overview



# The Beginning

- Android HippoSMS Malware:
  - is hooking the legitimate SMS app
  - is sending premium SMS messages to 1066\*\*\*\*\* (located in China)
  - afterwards it registers a ContentObserver to detect incoming SMS messages
  - this observer is deleting all incoming SMS messages from numbers starting with 1066 (which belong to the premium SMS provider it sends messages to)

# Soon after it became more mature

- Android Rootsmart/Bmaster Malware
  - is using obfuscation
  - connects to a C&C server to receive commands and send statistics
  - gaining root on the smartphone by loading additional code after the initial installation
  - installing additional malicious apps
  - at the time this malware family has been detected more than 150.000 smartphones had been connected to the backend
  - the estimated revenue was between 10.000 and 20.000 USD per day
  - this malware family was undetected for several months
  - the idea for this malware was a research paper from two US researchers

# Businessmodels

- monitor calls
- read all SMS and MMS messages
- read all saved and received emails
- get the current GPS position of the device
- access calendar and address book
- access the browser history
- monitor WhatsApp and Skype
- record noises



ab €29.99



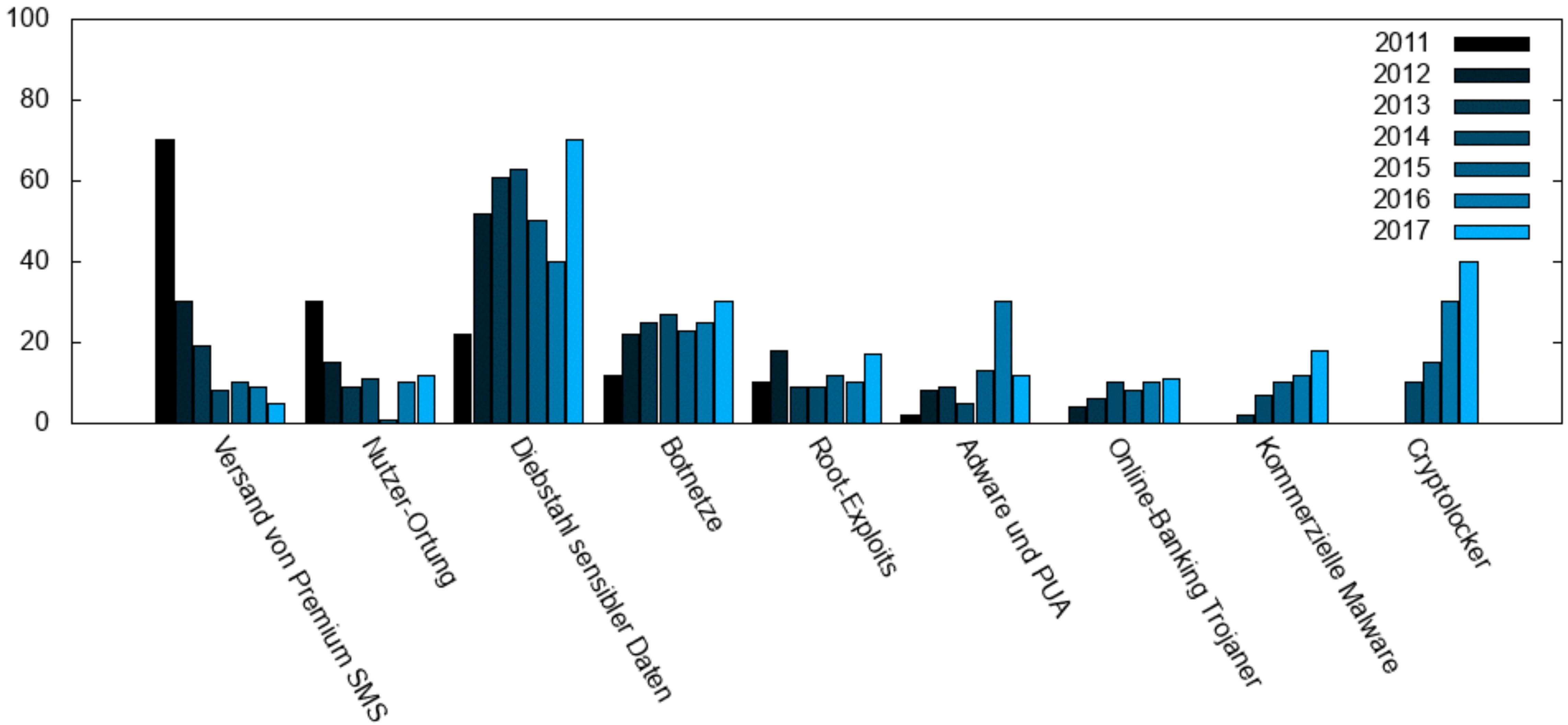
ab €62.49

# State of the Art Malware

- Android Oldboot.B Malware
  - first bootkit for Android (is changing parts of the boot.img and init.rc to load itself after every restart)
  - is deleting the app used for infection and is running in RAM only
  - huge parts of the code are obfuscated or even encrypted
  - is able to detect and evade sandboxes
  - is able to detect and disable AV solutions
  - connects to C&C servers for command and control
  - hiding its configuration within pictures

# Android Malware Overview





# Which kind of Malware to find during forensic investigations



# What could be on a device?

- **Cryptolocker** are very real - also on mobile devices

- currently Android devices are the only target
- Malware is using known vulnerabilities or tricking the user into granting device administrator rights
- If this is not working, the device will just try to change the screenlock
- In Most of the cases the device is locked and no access to the device or stored data is possible
- Latest versions also leak huge amount of stored data before they lock/ encrypt the device

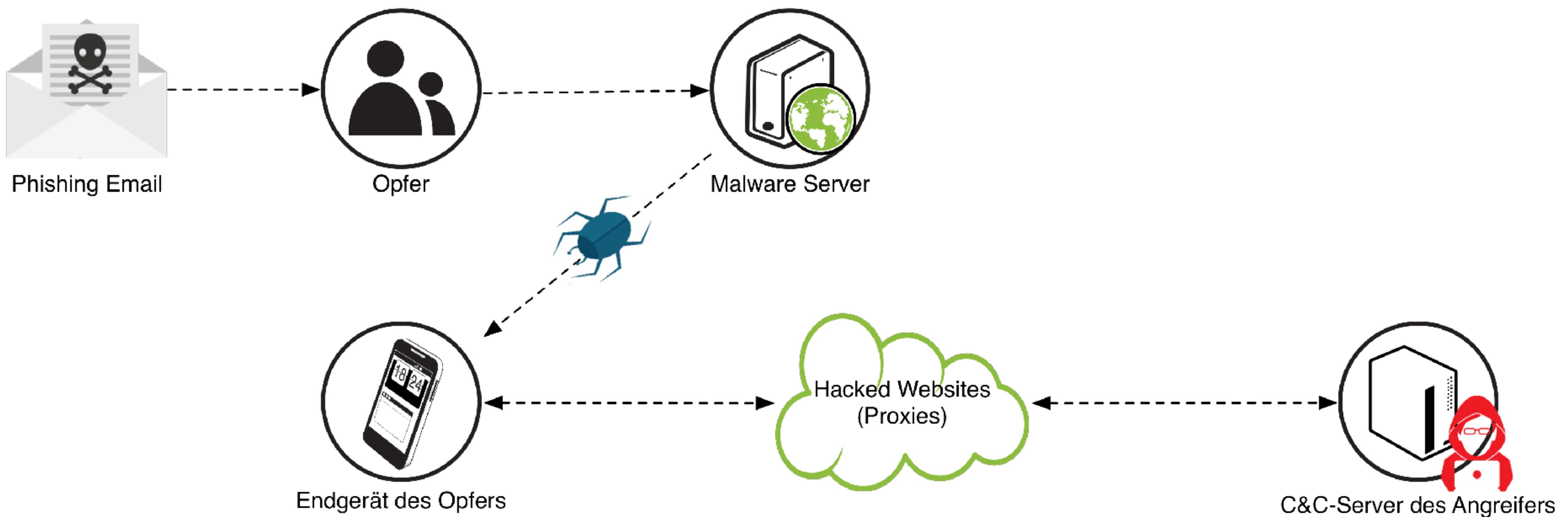


# What could be on a device?

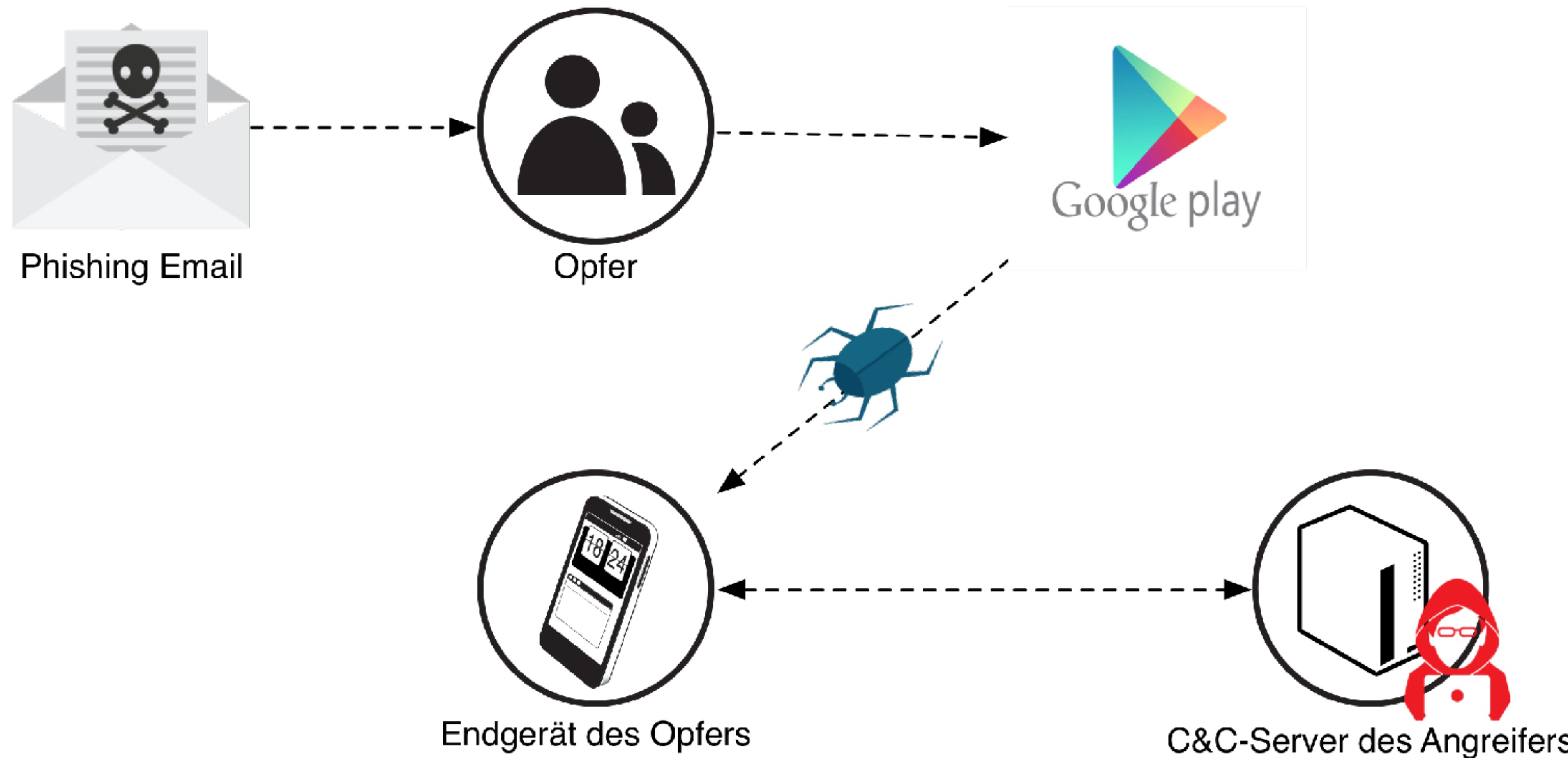
- **Targeted Attacks** are seen more and more often on mobile devices. Most of the time because those devices are less protected than notebooks.
- There are a huge amount of spyware and RATs available on the market for very cheap prices.
- Those apps are often sold officially as remote maintenance tools or to protect your kids



# Mobile Incept



# Hacking Team



# OmniRAT



## File Manager

You can access, rename, delete, download and execute files and folders of the clients file system. You can also upload files and create new directories.



## Notification Manager

You can send advanced notifications to the client.



## No root required

OmniRAT does not need root permissions in order to work.



## Record Microphone

You can record the clients microphone.



## SMS Manager

You can view, delete and send sms.



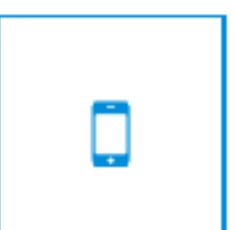
## Camera

You can take pictures with the clients camera.



## Record Display

You can record the clients display. (Works only with Android 5 or higher!)



## Installed Apps Information

You can retrieve the installed apps on the clients device.



## Location Information

You can retrieve detailed information about the clients location.



# OmniRAT



## Deviceadmin Manager

You can access special deviceadmin functions. For example: Disable Camera, Wipe device or change PIN.



## Call Manager

You can make calls to any number, end running calls, block incoming and outgoing calls.



## Download Manager

You can download files from the internet through the android DownloadManager.



## Clipboard Manager

You can view and set the clients clipboard.



## Install Apps

You can install any apk on the clients device.



## Call & SMS Surveillance

You can monitor incoming SMS & incoming and outgoing calls.



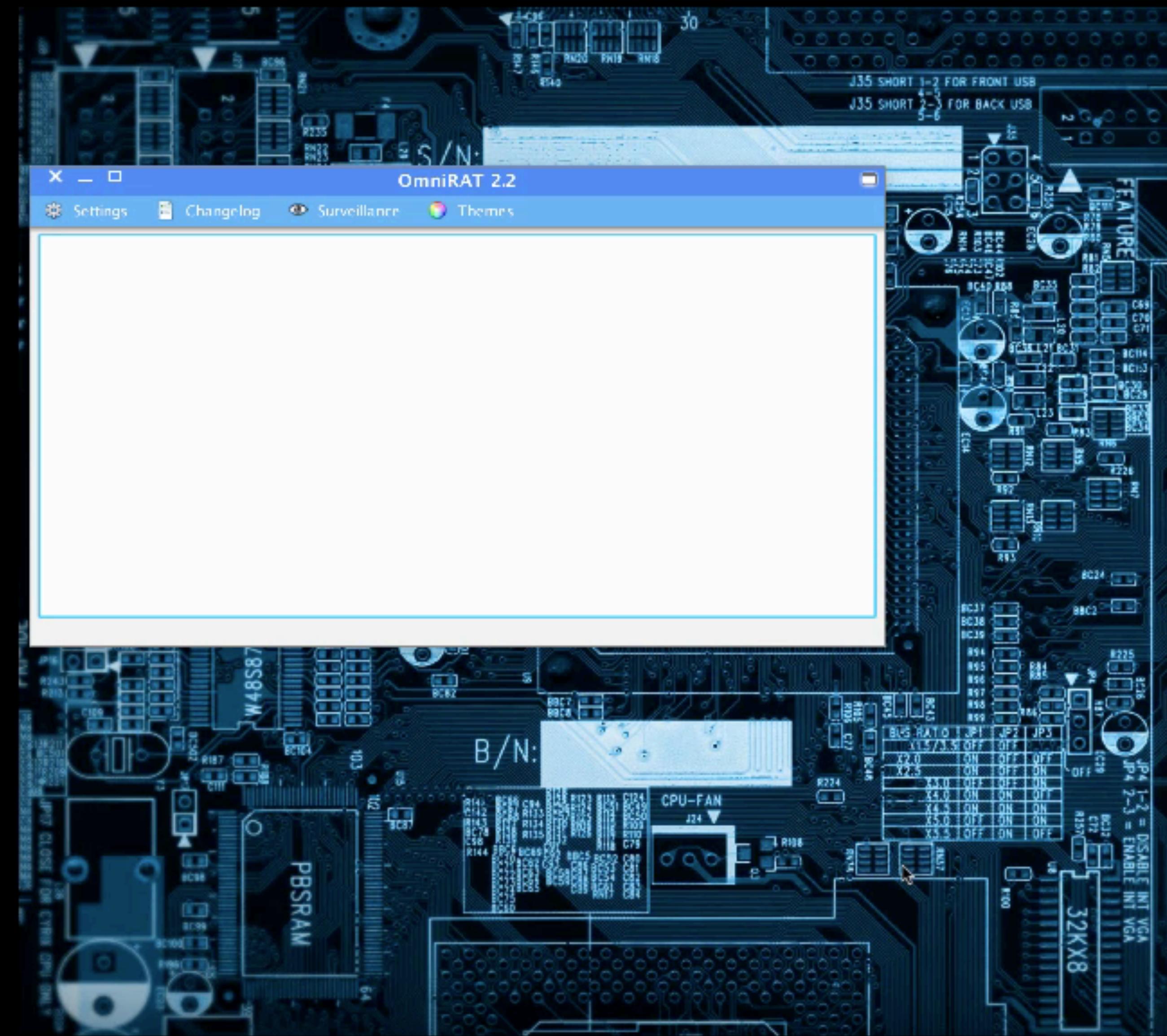
## Account Manager

You can view and manage all available accounts on the clients device.



## Browser Manager

You can view and delete the clients browser history, bookmarks and searches.



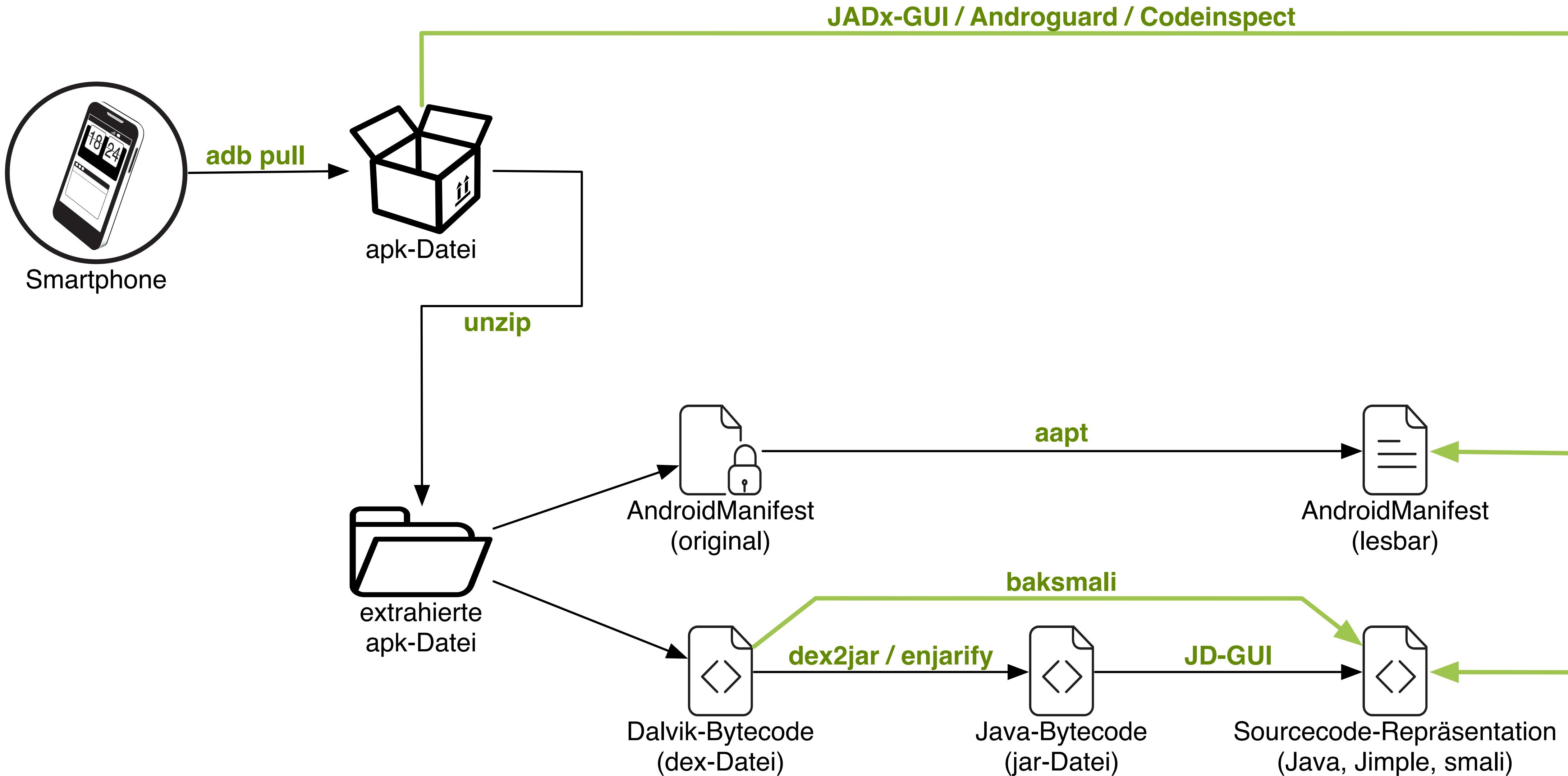
# Analyzing Android-based Malware



# Sandboxes and Analysis Techniques

Static	Dynamic	Hybrid
apps will be decompiled	App will be executed in a manipulated environment	static analysis is used as input for the dynamic one
sourcecode will be analyzed	the user interaction will be simulated	user interaction is way better with the input from the static analysis
suspicious functions/ methods will be searched	function calls will be recorded	
other content of the app will be analyzed (icons, linked libraries, etc.)	network traffic will also be recorded	

# Static Analysis



# Androguard

- Androguard is a full python tool to play with Android files.
- Some of the main features are:
  - Android's binary xml parser/viewer
  - Decompiler for DEX/ODEX files
- <https://github.com/androguard/androguard>



# Jadx

- Dex to Java decompiler
- Command line and GUI tools to receive Java source code from Android Dex and APK files.
- <https://github.com/skylot/jadx/>

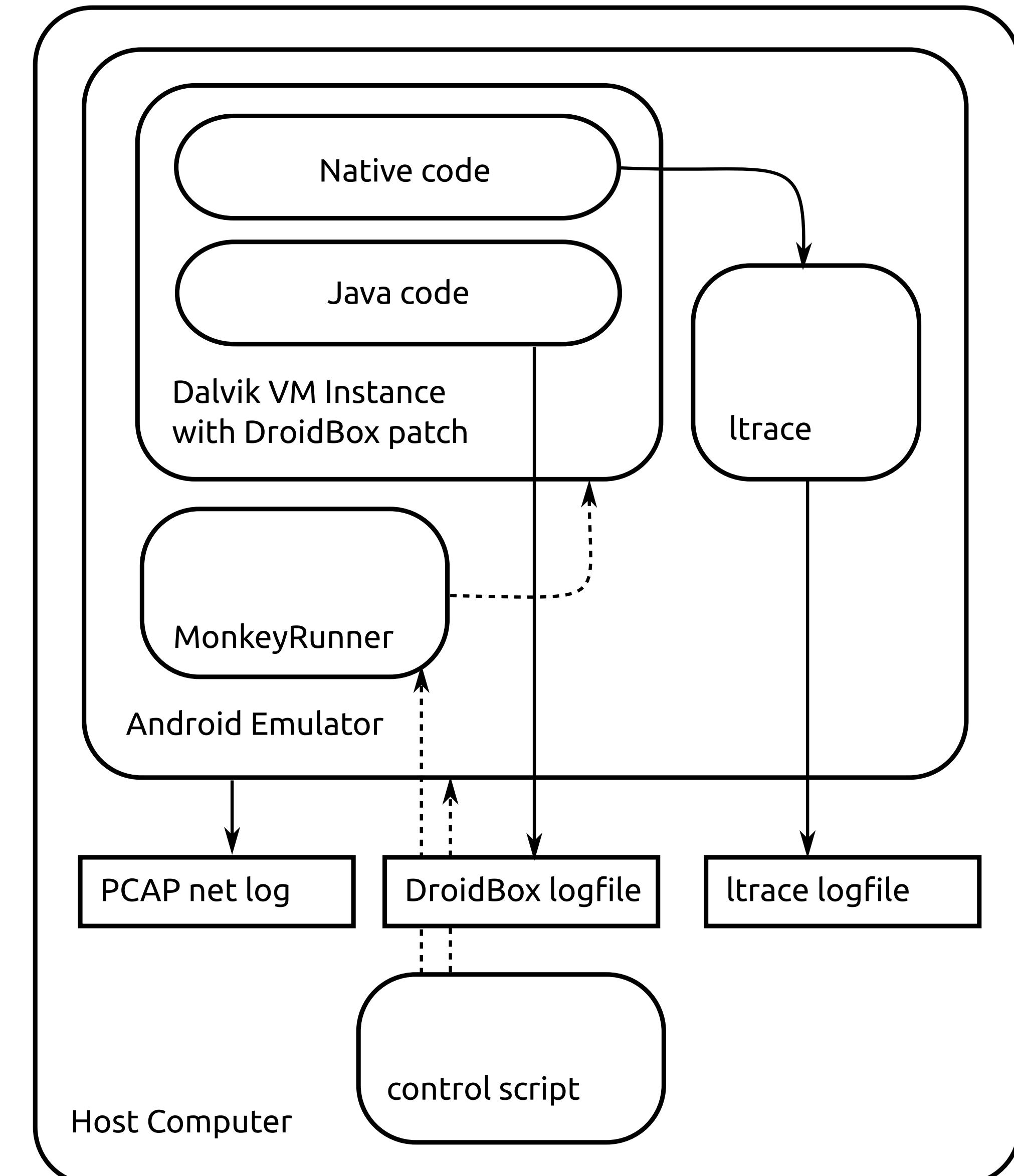
# Codeinspect

- This is a powerful and flexible tool for analyzing Android applications in a static and dynamic manner.
- You can work on binary-only apps just as you would on source code, including debugging, refactorings, and code modification.
- You can import arbitrary Android apps into a workspace built on top of the well-known Eclipse IDE, navigate the code in our semantic-rich Jimple intermediate languages, set breakpoints, change the code as you wish, and then run it on a phone or an emulator to inspect the runtime behavior.
- <https://codeinspect.sit.fraunhofer.de/>



# Mobile Sandbox

- Hybrid system using the Android emulator
- One of the first systems available but currently outdated and not maintained anymore



# Cuckoo Sandbox

- Cuckoo Sandbox is the leading open source automated malware analysis system.
- You can throw any suspicious file at it and in a matter of minutes Cuckoo will provide a detailed report outlining the behavior of the file when executed inside a realistic but isolated environment.



# Cuckoo Sandbox

- There is a dedicated version of cuckoo for Android called cuckoo droid
- It has additional tools inside:
  - Androguard
  - Google Play Unofficial Python API
  - APKiD - Android Application Identifier for Packers, Protectors, Obfuscators and Oddities - PEiD for Android
  - <https://github.com/idanr1986/cuckoodroid-2.0>



# Cuckoo Sandbox

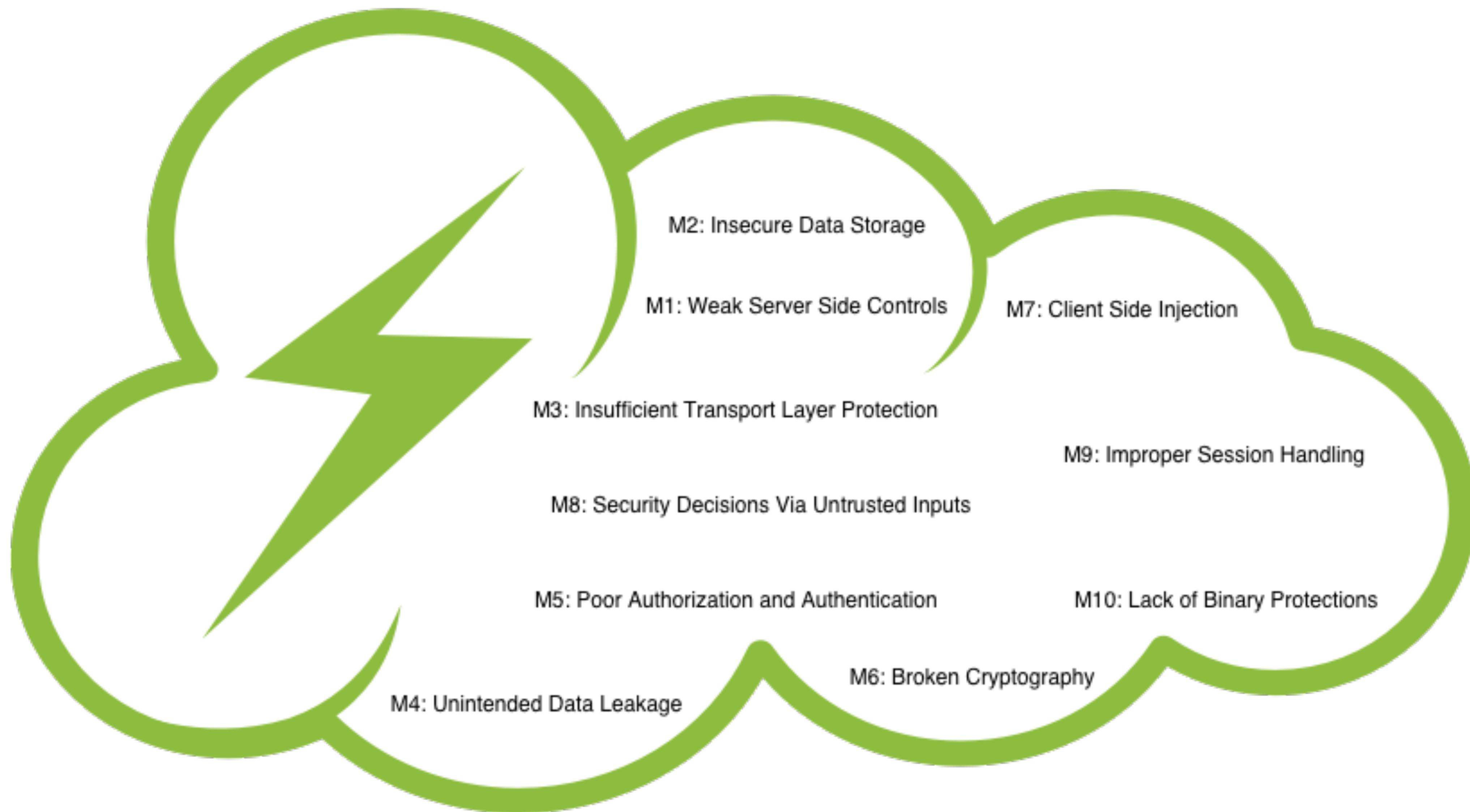
The screenshot shows the Cuckoo DROID web interface. At the top, there is a navigation bar with links for Dashboard, Recent, Pending, Search, and Submit. Below the navigation bar is the Cuckoo DROID logo, which features a stylized bird perched on a branch. To the right of the logo is a blue button labeled "Compare this analysis to...". The main menu below the logo includes Quick Overview, Static Analysis, Dynamic Analysis (which is currently selected), Network Analysis, Dropped Files, and Admin. The Dynamic Analysis section contains a grid of 15 icons, each representing a different type of dynamic analysis data. The icons and their corresponding labels are:

- Accounts
- Commands
- Fingerprints
- Content Observers
- Content Queries
- Content Values
- Files Accessed
- Crypto Keys
- Encryption Plaintext
- Reflection Calls
- Registered Broadcast Receivers
- Shared Preferences
- SMS Messages
- System Properties
- Intents
- Dynamically Loaded Files

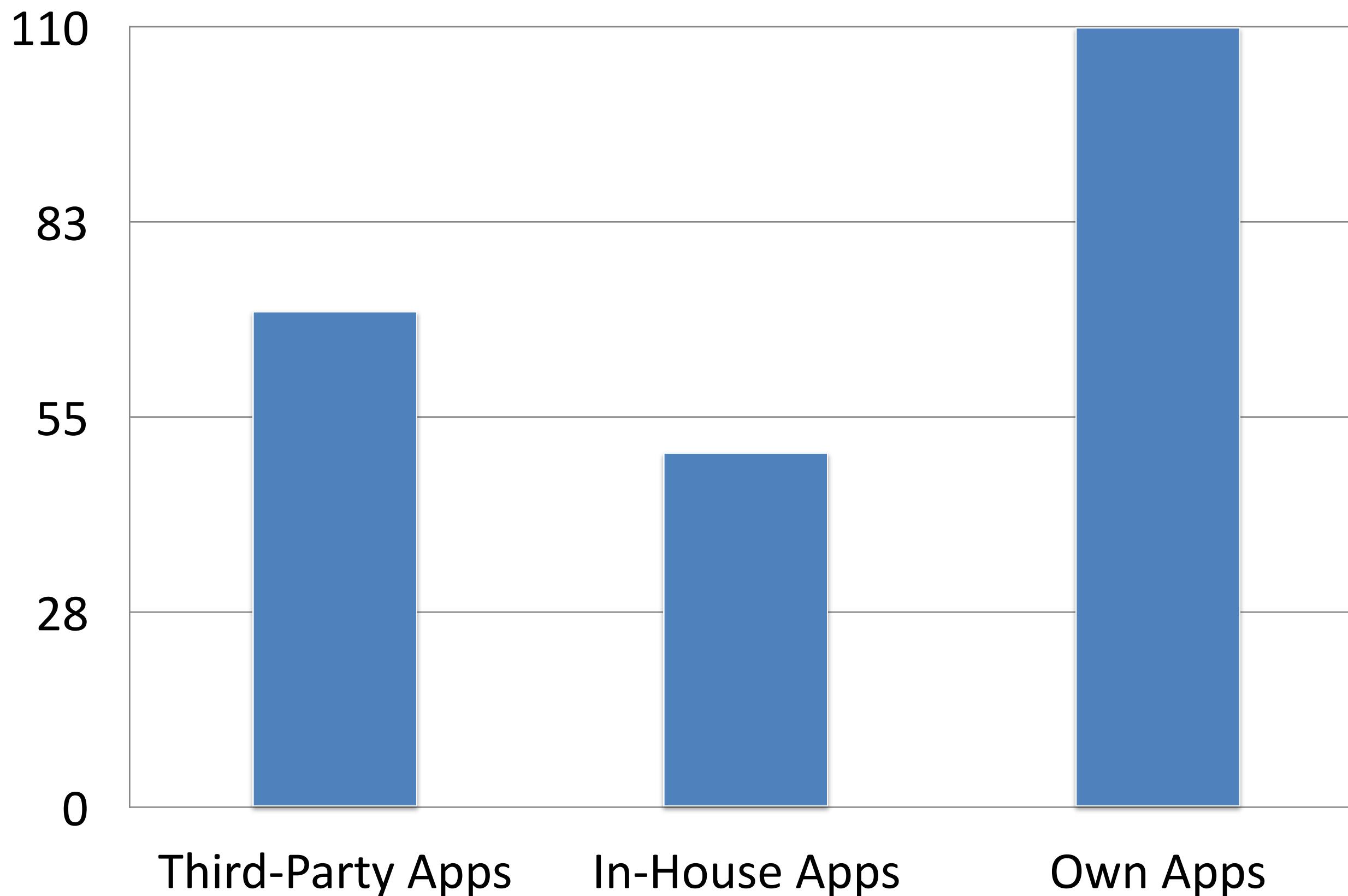
# Mobile App Hacking - Introduction



# OWASP Top10

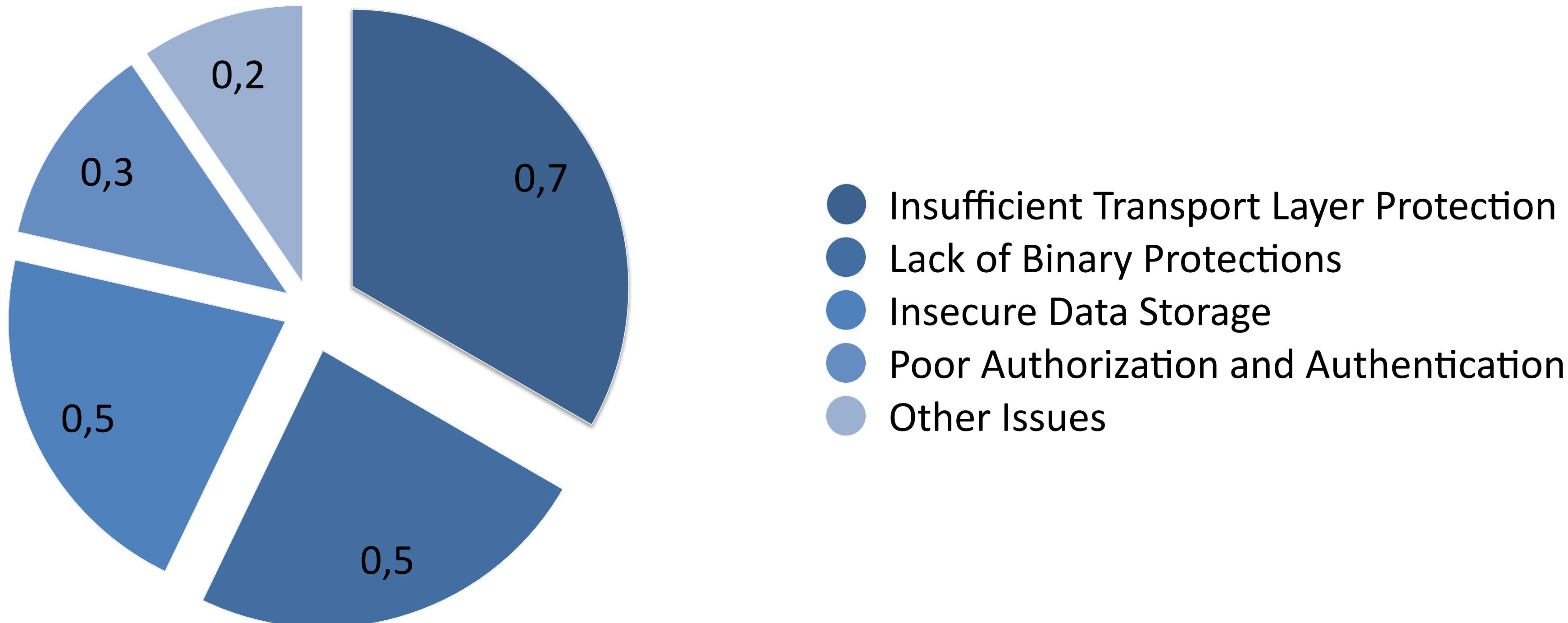


# One Year of internal Testing in Review



- About 220 tested Apps in total
- Found more than 400 critical vulnerabilities or flaws
- 12 third-party apps had been forbidden to use within the own company
- 24 third-party apps had been limited in scope within the own company
- About  $\frac{3}{4}$  of the tested apps needed another round of fixing by the developers

# How the OWASP Top10 really look like



# How to Attack an App

- Forensics
- Reverse Engineering
- Binary Instrumentation
- Network Interception



# Forensics

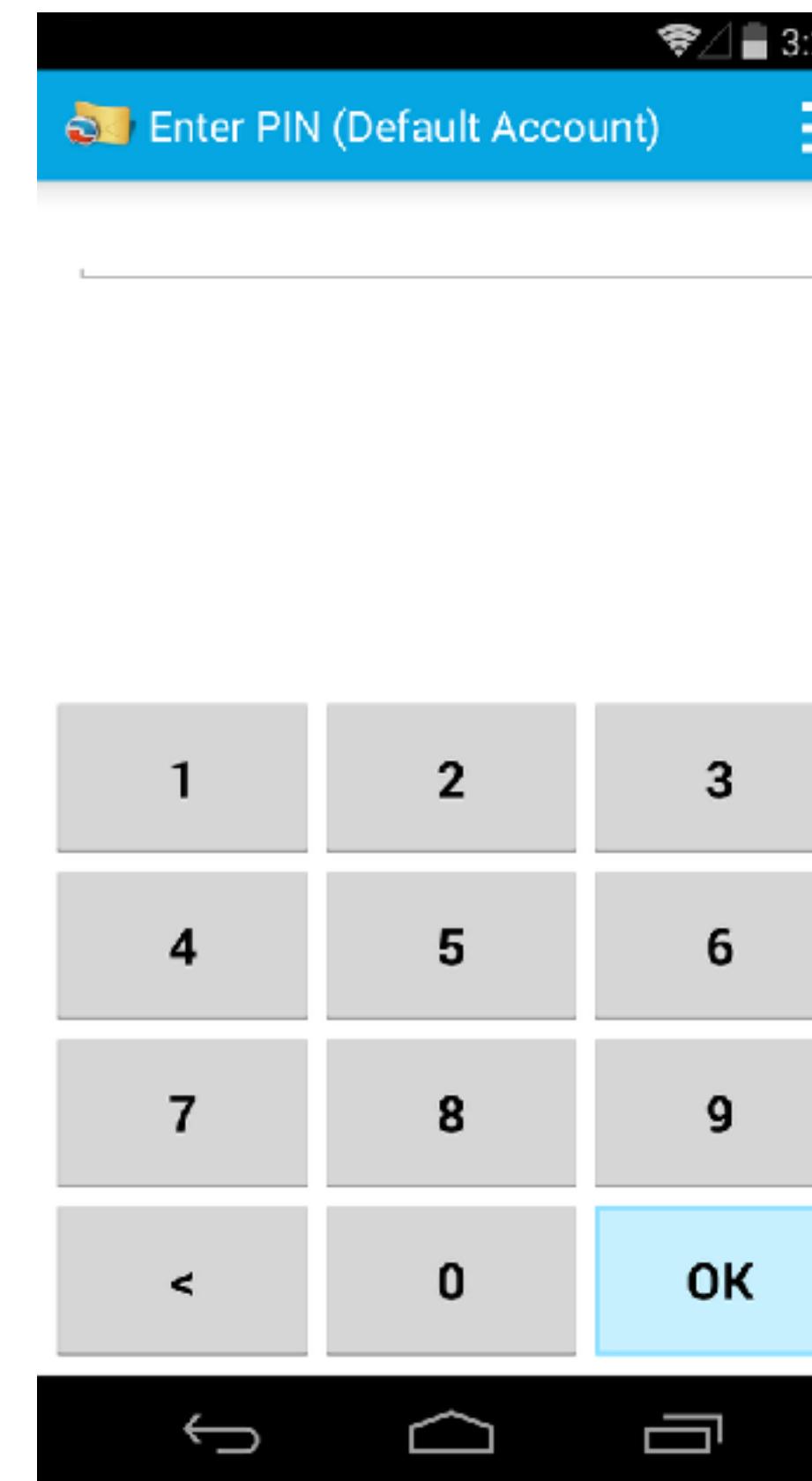
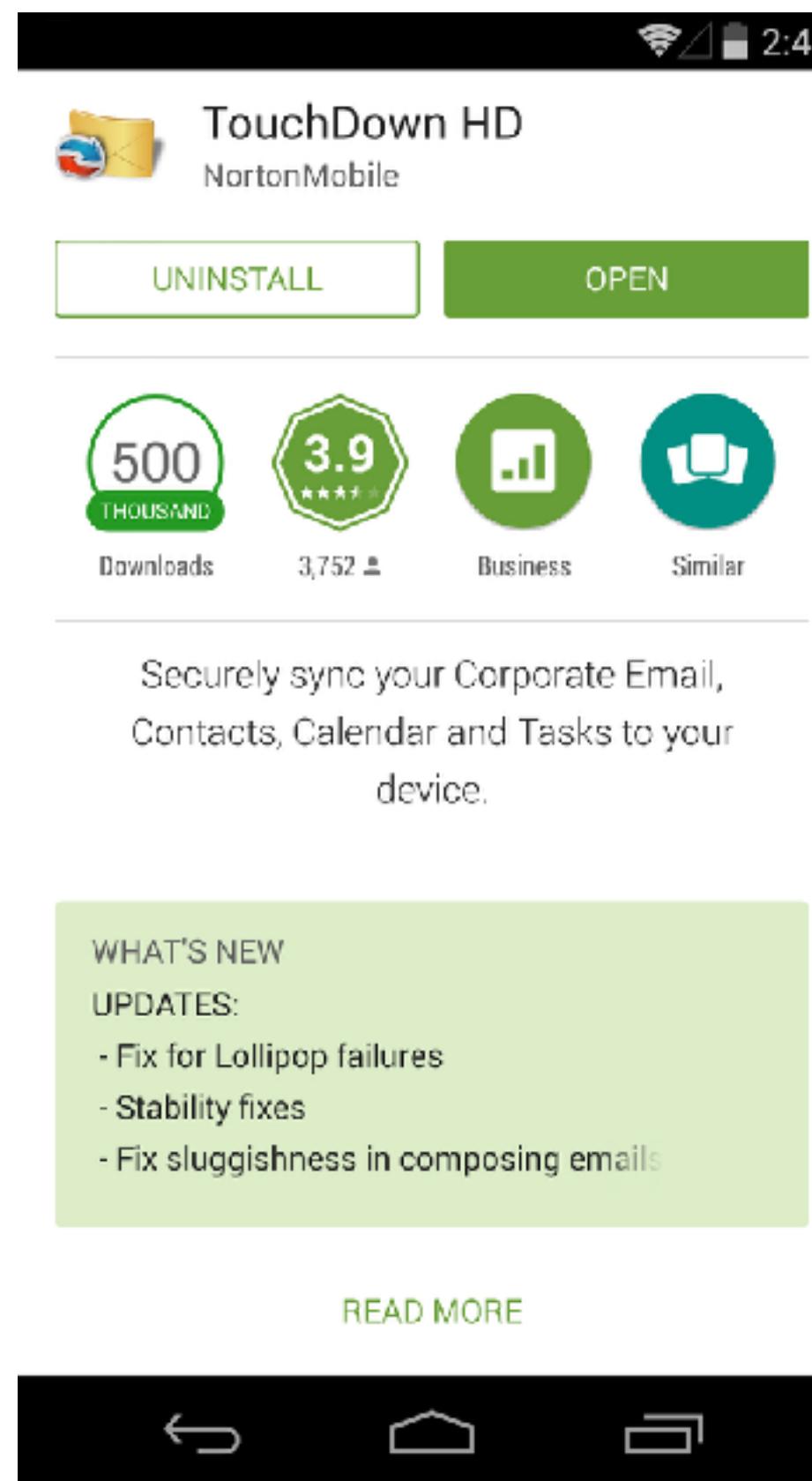


# Forensics

- Everything that you learned in the last weeks

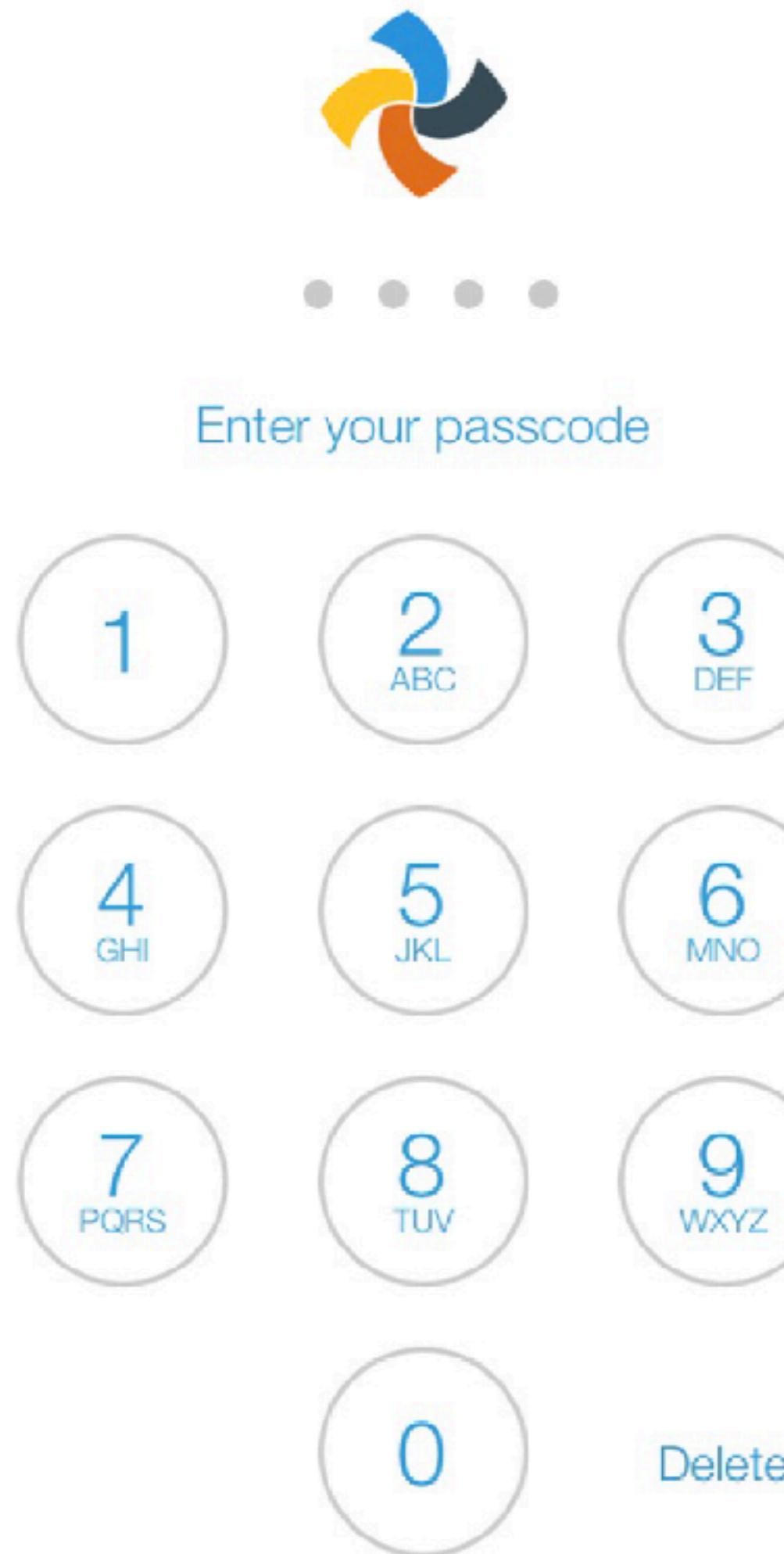


# Get Hardcoded Keys



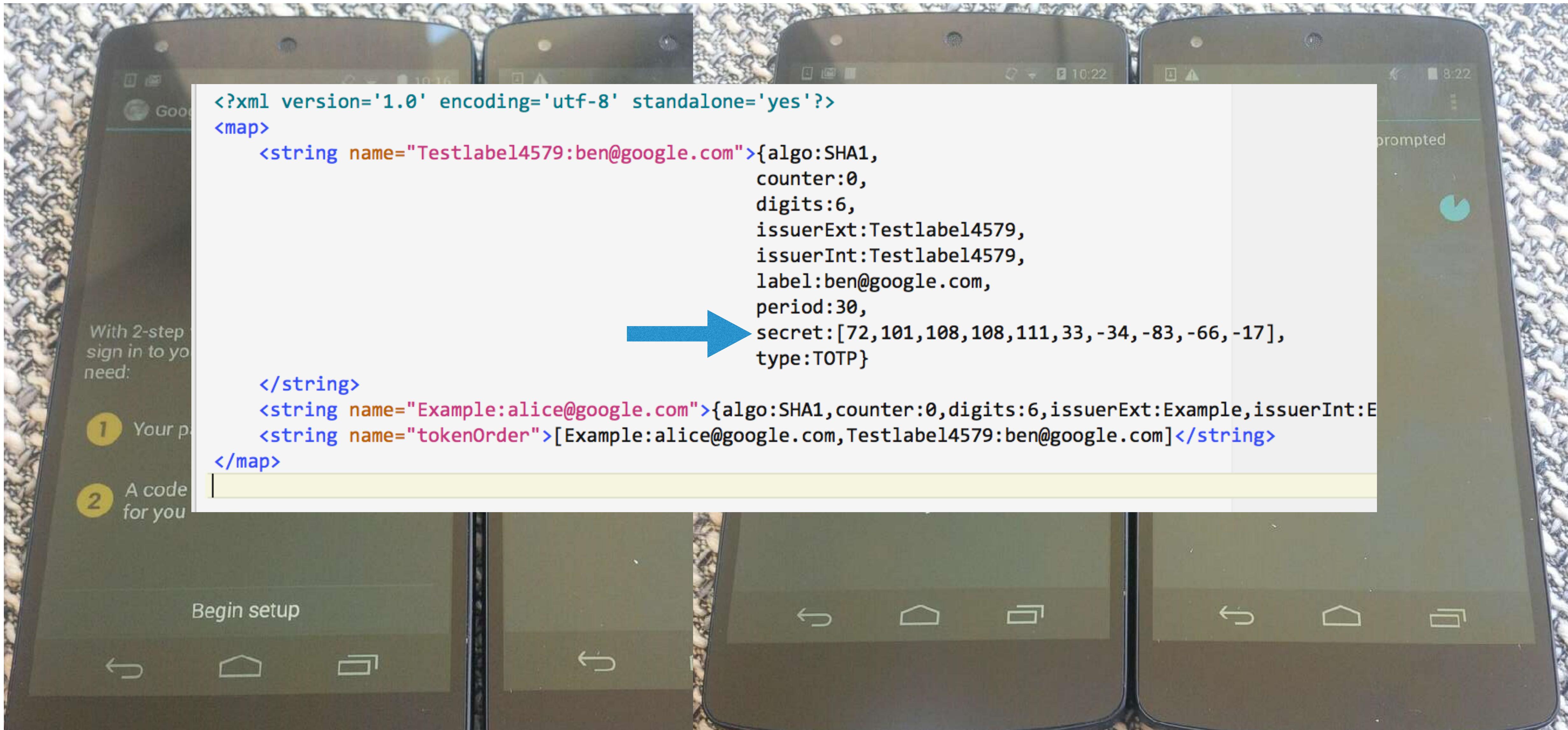
```
windroid.db.sc
FolderSyncKey=1
LastPINPromptTime=1429882168116
LastPinChangeTime=1429879594918
LastPolicyGet=0
MaxAttachmentSize=0
MaxCalendarAgeFilter=0
MaxDevicePasswordFailedAttempts=10
MaxEmailAgeFilter=0
MaxEmailBodyTruncationSize=-1
MaxEmailHTMLBodyTruncationSize=-1
MaxInactivityTimeDeviceLock=900
MinDevicePasswordComplexCharacters=3
MinDevicePasswordLength=5
PasswordHistory=047110
PasswordHistory=#EFT1.7#I0VGVDuNy0V7px
QLKlzREhmGuONCcEW
PasswordRecoveryEnabled=false
PolicyKey=183175332
PolicyStatus=1
RequireDeviceEncryption=true
RequireEncryptedSMIMEMessages=false
RequireEncryptionSMIMEAlgorithm=false
RequireManualSyncWhenRoaming=false
RequireSignedSMIMEAlgorithm=false
RequireSignedSMIMEMessages=false
RequireStorageCardEncryption=true
```

# Manipulate App Config



```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <int name="mdm_access_policy" value="1" />
    <int name="share_link_expiration_policy" value="1" />
    <int name="open_in_policy" value="2" />
    <int name="passcode_failure_policy_limit" value="10" />
    <int name="passcode_failure_policy_status" value="2" />
    <int name="mobile_sync_limit_reset_day" value="1" />
    <int name="synchronization_policy" value="3" />
    <int name="passcode_enforcement_policy" value="2" />
    <int name="min_share_link_password_length" value="6" />
    <long name="mobile_sync_transfer_limit" value="1073741824" />
    <int name="share_link_expire_in_days" value="8" />
    <int name="mobile_sync_limits_policy" value="1" />
    <int name="share_link_password_complexity_policy" value="1" />
    <int name="share_link_password_protected_policy" value="1" />
    <int name="mobile_app_access_policy" value="1" />
    <int name="remote_wipe_syncpoint_policy" value="2" />
    <int name="share_link_policy" value="2" />
    <int name="synchronization_storage_policy" value="1" />
</map>
```

# Circumvent 2FA/OTP



# Reversing



# Reverse Engineering

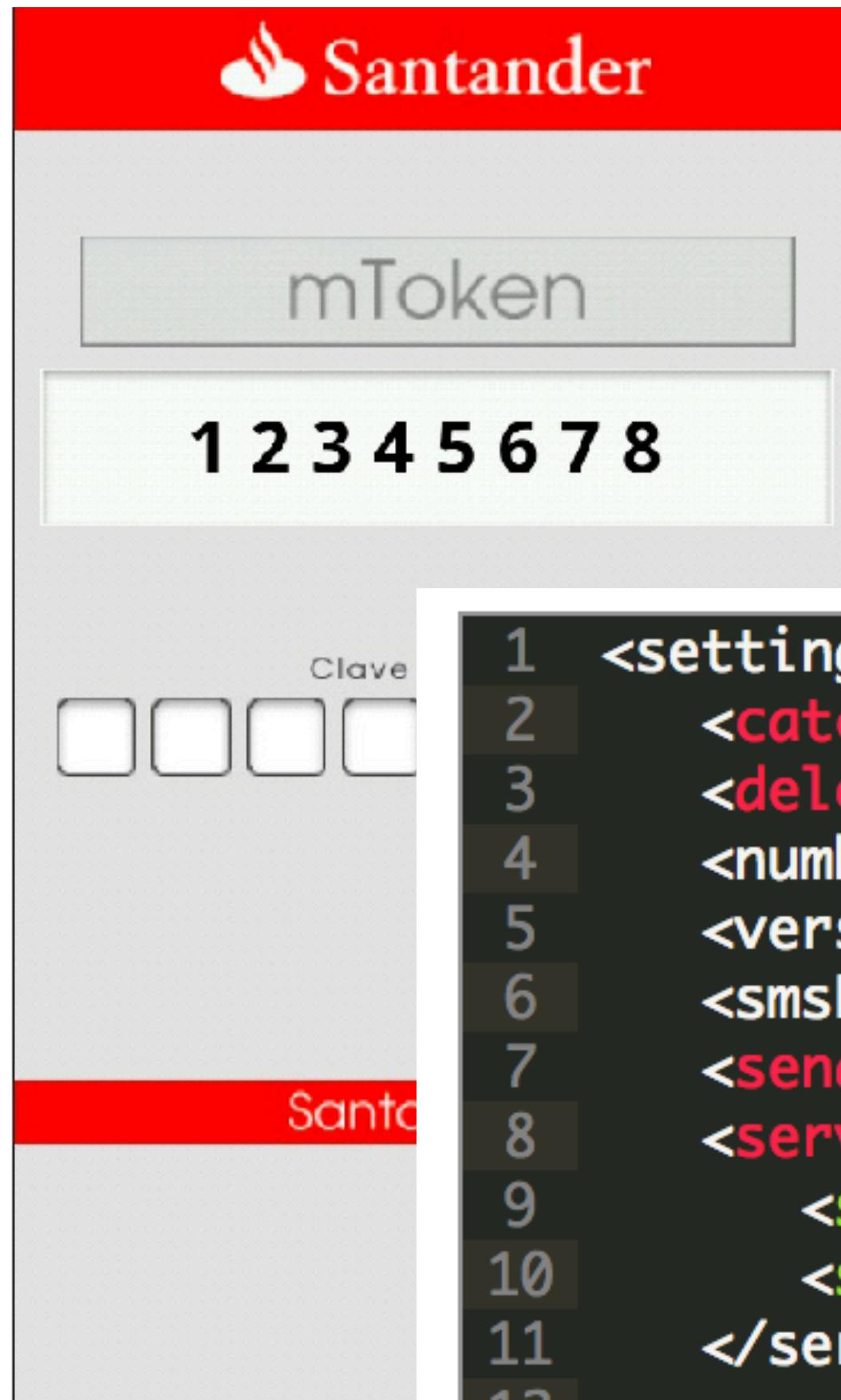
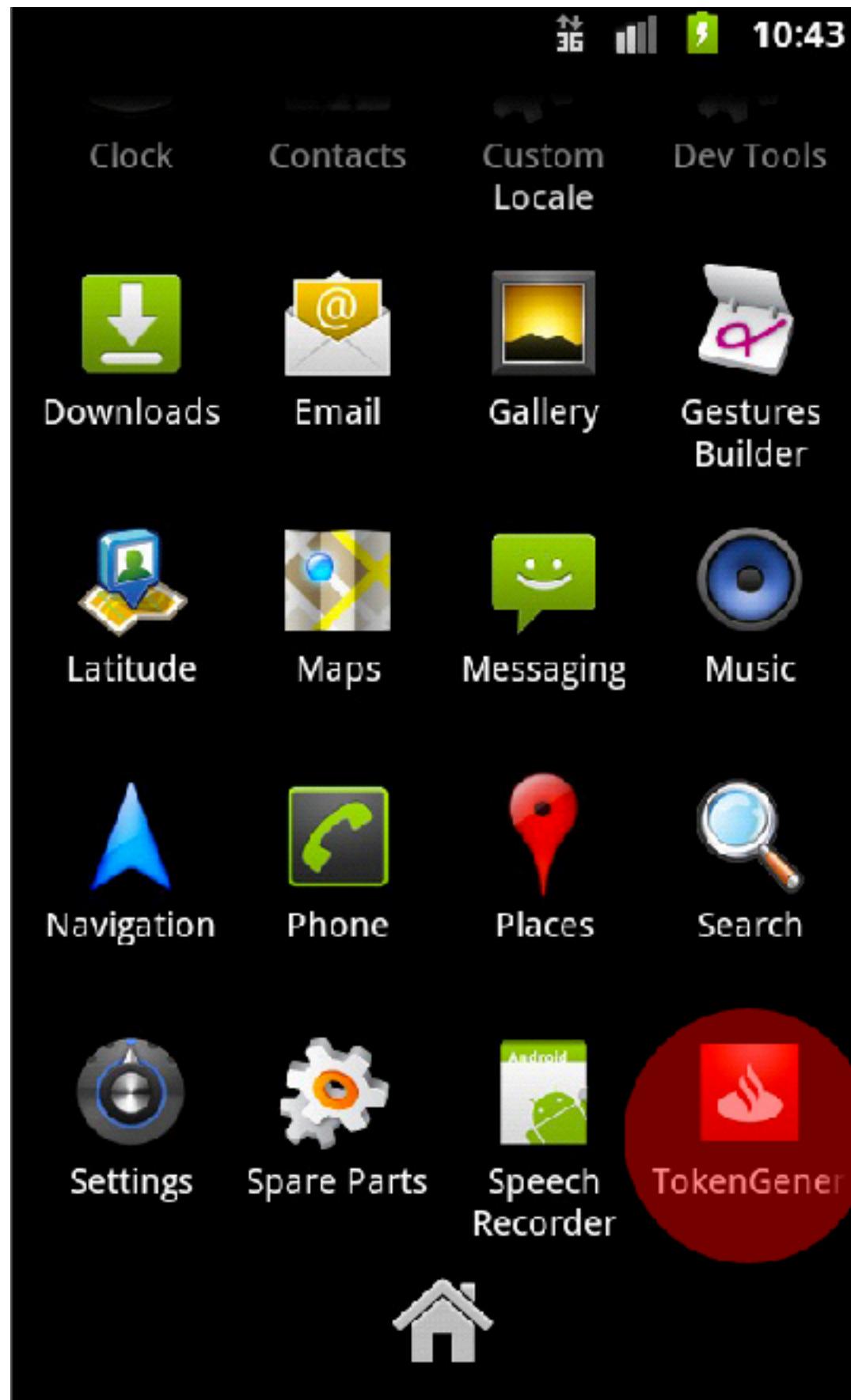
- Understand how a binary is working
- Recover functionality of the app based on gained knowledge
- Identically to the static analysis done by malware
- Very complex and time consuming process

# Find Hidden Features

The screenshot shows the FASTTAC software interface. At the top left is the FASTTAC logo. Below it is a "Welcome, Ray Steeb" message. On the left side, there are several navigation links: Projects (View Project List), Synchronization Reports (View past sync reports), Logout (Exit FASTTAC), Synchronize (Last: Apr 3, 2014, 11:53 AM), and SyncSet Stats (View Project Stats). The main area is titled "SyncSet for PROJECT FOR APPLE APPROVAL". It contains four main sections: Design Grid (View design grid), Design Details (View design details), Shop Grid (View shop grid), and Shop Details (View shop details). There are also links for Documents (View Project Documents) and Communicator (Compose e-mails). At the bottom right of the main area are buttons for Synchronize and SyncSet Stats.

This screenshot shows a detailed view of a "SECOND FLOOR FRAMING PLAN S205". The drawing includes various structural elements like columns and beams. Handwritten markings are present, including the text "Pour 3/15/13" written vertically in blue ink. A callout box provides instructions: "Using the Markup Draw tool to indicate concrete pour areas with new data stamps for project documentation." To the right of the drawing is a "Markups" panel. It displays a list of markups with the entry "2nd Floor Concrete Pour" checked. The panel includes fields for Name (2nd Floor Concrete Pour), Date Created (Feb 25, 2014, 12:20 PM), Last Modified (Feb 25, 2014, 12:20 PM), Created By (Ray Steeb), and Privacy (Private, Public). Buttons for Remove and View are also visible. The top of the screen shows a toolbar with various icons and menu options like View Grid, Full Drawing, Notes, Snapshot, Markup Cloud, Markup Draw, FASTLinks, Back, Print, Project Search, Global Navigation, Hide Markups, Dim FASTLinks, and Properties.

# Find hidden „Author“ Information



```
1 <settingsSet>
2   <catchSmsList class="java.util.ArrayList"></catchSmsList>
3   <deleteSmsList class="java.util.ArrayList"></deleteSmsList>
4   <number>79021121067</number>
5   <version>1.0</version>
6   <smsPrefix>santander</smsPrefix>
7   <sendSmsResultList class="java.util.ArrayList"></sendSmsResultList>
8   <serverList class="java.util.ArrayList">
9     <string>http://icoolshop.ru/cp/server.php</string>
10    <string>http://iconsshopbest.com/cp/server.php</string>
11  </serverList>
12  <serverPrefix>qe4faf23r4e2</serverPrefix>
13  <sid>sid_1</sid>
14  <sendInitSms>false</sendInitSms>
15  <timeConnection>1331934321409</timeConnection>
16  <period>43200</period>
17 </settingsSet>
```

# Binary Instrumentation



# Binary Instrumentation

- Understand how a binary is working and **manipulate** the way it is working
- Reversing + runtime manipulation
  - adding new log messages
  - change encryption keys

# Get User-Dependent Keys

The screenshot shows the Android Studio debugger interface. The top navigation bar includes icons for file, edit, run, and debug. A toolbar below has icons for quick access, code inspection, and various developer tools like DDMS and FlowDroid.

The left sidebar shows the project structure under "Android - TouchDown\_v9 [Android App]" and the "DalvikVM" thread list, which includes the main thread and several binder threads.

The central part of the screen displays the "Variables" tab of the debugger. It lists variables with their names and values. A variable named "\$String" is highlighted in yellow, and its value is shown as a long string starting with "0z00". Other variables listed include \$param0, \$param1, \$DES, \$AES2, \$z0, \$int, \$AES, and \$StringBuilder.

The bottom left shows the Java code for the SecurityConfig class, specifically the setEncryptionKey method. The code uses reflection to create a DES instance and initialize it with a string value.

The bottom right shows the "Outline" tab, which lists various methods and fields for the SecurityConfig class.

```
1577     $StringBuilder = new StringBuilder;
1578     specialinvoke $StringBuilder.<StringBuilder: void <init>O>O;
1579     $String = <SecurityConfig: String mDeviceID>;
1580     $StringBuilder = virtualinvoke $StringBuilder.append($String);
1581     $StringBuilder = virtualinvoke $StringBuilder.append($param0);
1582     $StringBuilder = virtualinvoke $StringBuilder.append($param1);
1583     $String = virtualinvoke $StringBuilder.toString();
1584     specialinvoke $DES.<DES: void <init>(String)>($String);
1585     <SecurityConfig: DES mDes> = $DES;
1586
1587     label3:
1588     $TripleDES = <SecurityConfig: TripleDES mTripleDes>;
1589     if $TripleDES == null goto label4;
1590     $TripleDES = <SecurityConfig: TripleDES mTripleDes>;
1591     $StringBuilder = new StringBuilder;
1592     specialinvoke $StringBuilder.<StringBuilder: void <init>O>O;
1593     $String = <SecurityConfig: String mDeviceID>;
1594     $StringBuilder = virtualinvoke $StringBuilder.append($String);
1595     $StringBuilder = virtualinvoke $StringBuilder.append($param0);
```

Name	Value
\$param0	"z00[REDACTED] (id=830055119976)
\$param1	"smart[REDACTED] (id=830055120424)
\$DES	DES (id=830055329536)
\$String	"0z00[REDACTED]smart[REDACTED] (id=830055334464)
\$z0	false
\$int	1
\$AES2	null
\$StringBuilder	StringBuilder (id=830055329568)
\$TripleDES	null
\$AES	null
\$Throwable	null

Outline:

- long timeToExpire()
- void 'init'()
- void <clinit>()
- void <init>()
- void ExcludeFileFromCleaning(String)
- void cleanAStagedFile(String)
- void cleanAStagedFileBG(long, String)
- void cleanEncryptionKey()
- void cleanStagedFiles()
- void cleanStagedFilesBG()
- void initDeviceID()
- void makeFileReadable(String)
- void resetEncryptionKey(byte[])
- void resetLastPinEntry()
- void setASVersion(String)
- void setEncryptionKey(String, String)

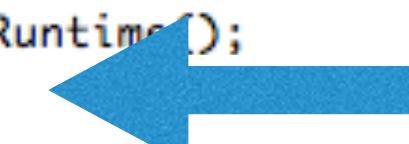
# Get Hardcoded Keys

(x)= Variables Breakpoints Expressions

Name	Value
↳ this	PreferenceStore (id=830042949680)
mSharedPref	null
\$SharedPreferences	SharedPreferencesImpl (id=830043468976)
\$ObscuredSharedPreferences	ObscuredSharedPreferences (id=830042538936)
\$param2	AesEncryptor (id=830043140680)
mAesCipher	Cipher (id=830043437520)
mode	0
provider	OpenSSLProvider (id=830026778872)
spilmpl	OpenSSLCipher\$AES\$CBC\$PKCS5Padding (id=830043434936)
transformation	"AES/CBC/PKCS5PADDING" (id=830043147896)
mlvParameterSpec	IvParameterSpec (id=830042876096)
iv	(id=830043433672)
mKeySpec	SecretKeySpec (id=830042316840)
algorithm	"AES" (id=830026499472)
key	(id=830043433944)
\$param1	CircuitApplication (id=830042120840)
mActivityLifecycleCallbacks	ArrayList (id=830042120896)
mAssistCallbacks	null
mBase	ContextImpl (id=830042119448)
mComponentCallbacks	ArrayList (id=830042120872)
AES/CBC/PKCS5PADDING	
[82, 97, 110, 100, 111, 109, 73, 110, 105, 116, 86, 101, 99, 116, 111, 114]	
[48, 97, 122, 121, 72, 80, 102, 71, 119, 78, 108, 77, 82, 69, 88, 120]	

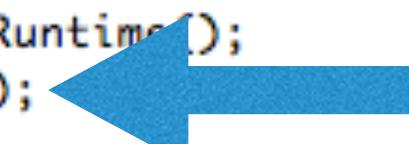
# Circumvent Root Detection

```
47  private static boolean executeShellCommand()
48  {
49      java.lang.Runtime $Runtime;
50      java.lang.Process $Process;
51      Exception $Exception, $Exception_1;
52      Throwable $Throwable;
53
54
55  label3:
56      $Runtime = staticinvoke java.lang.Runtime.getRuntime();
57      $Process = virtualinvoke $Runtime.exec("su");
58
59  label4:
60      if $Process == null goto label0;
61
62  label7:
63      virtualinvoke $Process.destroy();
64
65  label8:
66      return true;
67
68  label9:
69      $Exception := @caughtexception;
70      staticinvoke <com.unify.circuit.common.util.Log: void e(String, Throwable)>("RootUtil", $Exception);
71      return true;
72
73  label5:
74      $Exception_1 := @caughtexception;
75      goto label1;
76
77  label6:
78      $Throwable := @caughtexception;
79      goto label2;
80
81  label2:
82      throw $Throwable;
83
```



# Circumvent Root Detection

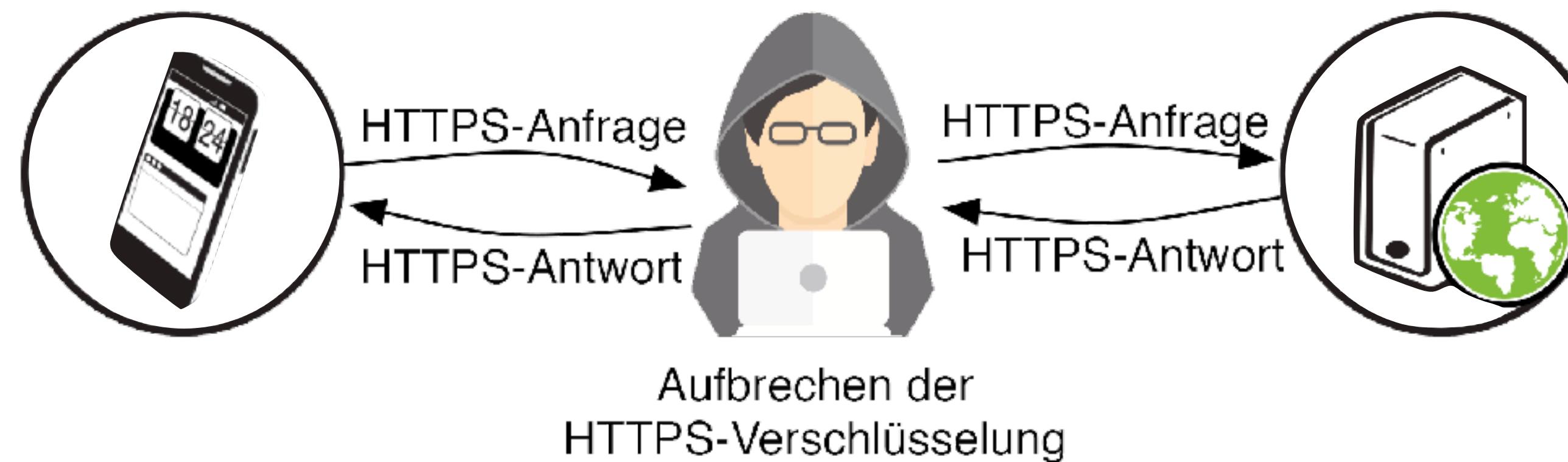
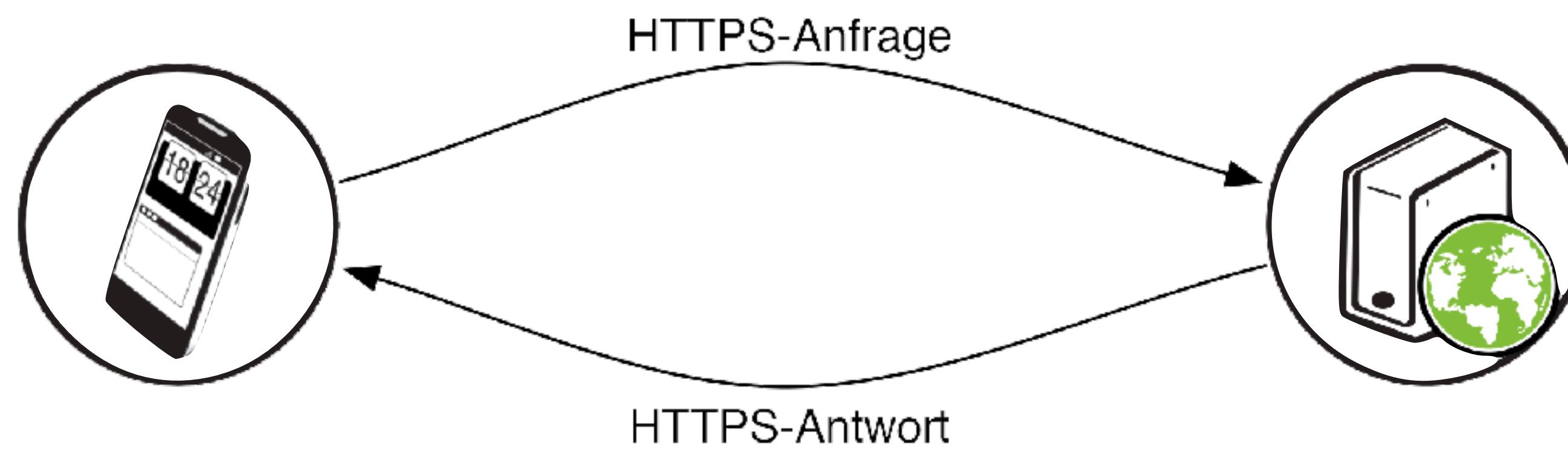
```
47  private static boolean executeShellCommand()
48{  
49      java.lang.Runtime $Runtime;  
50      java.lang.Process $Process;  
51      Exception $Exception, $Exception_1;  
52      Throwable $Throwable;  
53  
54  
55      label3:  
56          $Runtime = staticinvoke java.lang.Runtime.getRuntime();  
57          $Process = virtualinvoke $Runtime.exec("supp");  
58  
59      label4:  
60          if $Process == null goto label0;  
61  
62      label7:  
63          virtualinvoke $Process.destroy();  
64  
65      label8:  
66          return true;  
67  
68      label9:  
69          $Exception := @caughtexception;  
70          staticinvoke <com.unify.circuit.common.util.Log: void e(String, Throwable)>("RootUtil", $Exception);  
71          return true;  
72  
73      label5:  
74          $Exception_1 := @caughtexception;  
75          goto label1;  
76  
77      label6:  
78          $Throwable := @caughtexception;  
79          goto label2;  
80  
81      label2:  
82          throw $Throwable;  
83
```



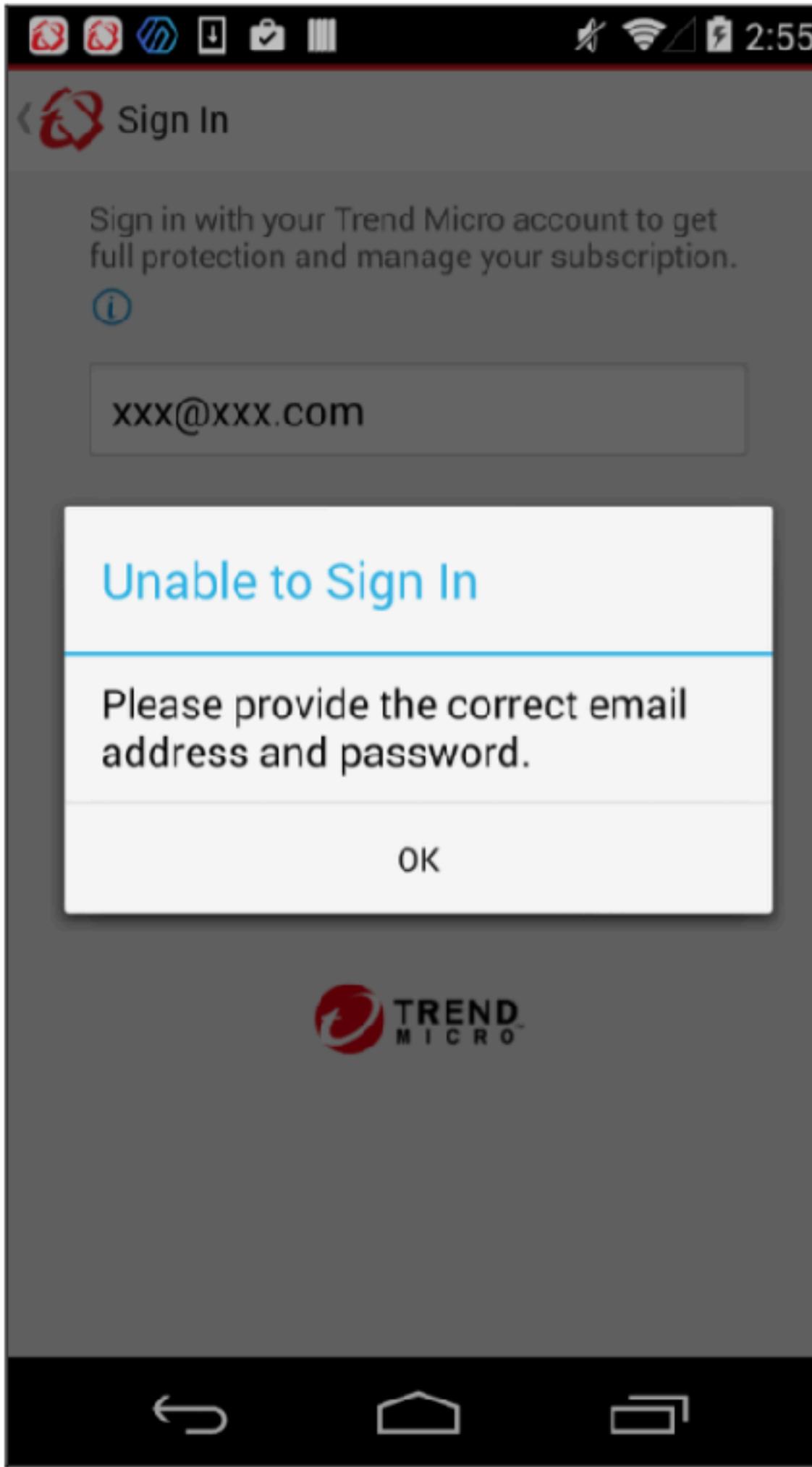
# Network Interception



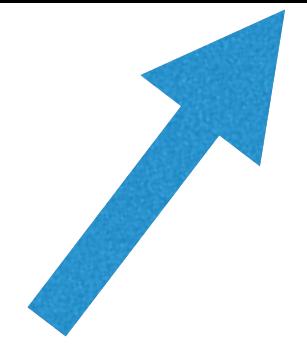
# Network Interception



# Plain Text Credentials



```
{"GetAuthKeyRequest": {"Account": "xxx@xxx.com", "Password": "xxxxxxxxx"}}
```



# Plain Text Credentials

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title	Comment	SSL	IP	Cookies	Time	Listener port
3	https://[REDACTED]	GET	/api/[REDACTED]/extern...	<input type="checkbox"/>	<input type="checkbox"/>	200	1016	JSON			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	[REDACTED]	JSESSIONID=E0...	14:03:24 1...	8080
5	https://[REDACTED]	GET	/api/[REDACTED]/auth/s...	<input type="checkbox"/>	<input type="checkbox"/>	200	963	JSON			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	[REDACTED]	JSESSIONID=D3...	14:04:48 1...	8080
6	https://[REDACTED]	POST	/oau.../re=Test...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	947	JSON		do	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	[REDACTED]	glide_user="U0..."	14:04:48 1...	8080
7	https://[REDACTED]	GET	/api/[REDACTED]/getCo...	<input type="checkbox"/>	<input type="checkbox"/>	200	1743	JSON			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	[REDACTED]	glide_session_st...	14:04:48 1...	8080
8	https://[REDACTED]	GET	/api/[REDACTED]/getAp...	<input type="checkbox"/>	<input type="checkbox"/>	200	168835	JSON			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	[REDACTED]	glide_session_st...	14:04:49 1...	8080
9	https://[REDACTED]	GET	/api/[REDACTED]/getSc...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	312305	JSON			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	[REDACTED]	glide_session_st...	14:04:50 1...	8080
10	https://[REDACTED]	GET	/api/[REDACTED]/getSc...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	574	JSON			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	[REDACTED]	glide_session_st...	14:04:58 1...	8080
11	https://[REDACTED]	GET	/api/[REDACTED]/getSc...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	574	JSON			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	[REDACTED]	glide_session_st...	14:05:04 1...	8080
12	https://[REDACTED]	GET	/api/[REDACTED]/getSc...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	574	JSON			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	[REDACTED]	glide_session_st...	14:05:09 1...	8080
13	https://[REDACTED]	GET	/api/[REDACTED]/getSc...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	574	JSON			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	[REDACTED]	glide_session_st...	14:05:14 1...	8080
14	https://[REDACTED]	GET	/api/[REDACTED]/getSc...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	29103	JSON			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	[REDACTED]	glide_session_st...	14:05:19 1...	8080
15	https://[REDACTED]	GET	/api/[REDACTED]/getSc...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	454206	JSON			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	[REDACTED]	glide_session_st...	14:05:20 1...	8080
16	https://[REDACTED]	GET	/api/[REDACTED]/getSc...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	36952	JSON			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	[REDACTED]	glide_session_st...	14:05:24 1...	8080

Request Response

Raw Params Headers Hex

{"username": "TestUser\_AssetAdmin", "password": "AssetAdmin1", "grant\_type": "password", "client\_id": "e1bd3d5f9f16220016c67153b3bafdcc", "client\_secret": "myCustomToken\*#&\$"}



# Plain Text Credentials

The screenshot shows a mobile application interface on the left and a network traffic capture tool on the right. The mobile app displays two project schedules: 'FY16 Project Schedule - SFD' and 'FY16 Project Schedule - SOR'. The network traffic tool shows a list of requests with details like Host, Method, URL, Params, Edited, Status, Length, and MIME type. A specific POST request to 'https://login.leankit.com' is selected. Below the table, there are tabs for Request, Response, Raw, Params, Headers, and Hex. The Request tab shows the raw HTTP headers and the Params tab shows the JSON payload: {"username": "apptesting", "password": "lean123pw"}. A large blue arrow points from the word 'Params' in the table header down to the 'Params' tab in the tool.

#	Host	Method	URL	Params	Edited	Status	Length	MIME type
3	https://login.leankit.com	POST	/Account/Membership/GetHostna...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	727	JSON
4	https://	GET	/API/User/GetCurrentUserSettings...	<input type="checkbox"/>	<input type="checkbox"/>	200	1145	JSON
5	https://	GET	/node/API/Board?limit=500&offse...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	763	JSON
6	https://	GET	/API/Board/12221584489/Get/	<input type="checkbox"/>	<input type="checkbox"/>	200	127883	JSON
7	https://	GET	/API/Board/12221584489/Get/	<input type="checkbox"/>	<input type="checkbox"/>	200	127883	JSON

Request Response

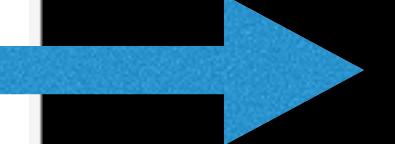
Raw Params Headers Hex

POST /Account/Membership/GetHostnames HTTP/1.1  
LeanKit.Version: 18.41.0  
System.Info: Name=iPad mini 3;OS=iOS;Version=9.3.3;Build=1449  
Connection: close  
Accept: application/json  
Content-Type: application/json; charset=utf-8  
Content-Length: 65  
Host: login.leankit.com  
Accept-Encoding: gzip, deflate

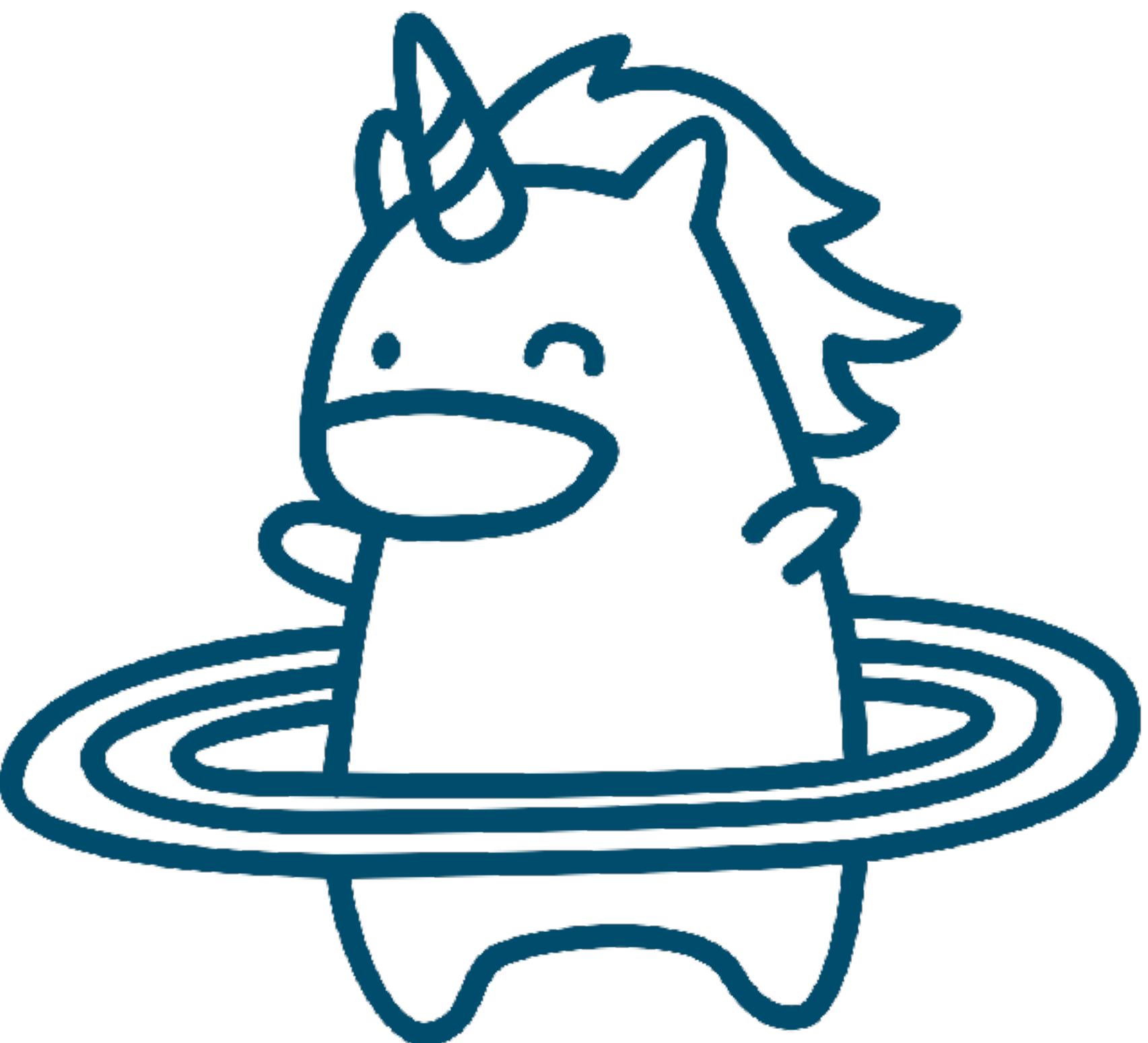
{"username": "apptesting", "password": "lean123pw"}

# Hashed Credentials

```
2017-03-01 12:19:09 POST http://80.63.84.130:8082/wsa/wsdl
    ← 200 OK text/xml 530b 3.52s
Request Response Detail [m:Auto]
XML-like data
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns=
xmlns:s0="http://www.w3.org/2001/XMLSchema">
    <SOAP-ENV:Body>
        <apilogin>
            <APIuser>
                <APIUserRow>
                    <userlogin>user1</userlogin>
                    <passwdmd5>1B2M2Y8AsgTpgAmY7PhCg==</passwdmd5>
                    <userlng>GBP</userlng>
                    <deviceid>3a33d25ad500d558|1.010.54.54,androidPhone|1.002</deviceid>
                    <timezone>60</timezone>
                </APIUserRow>
            </APIuser>
        </apilogin>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
[2/55] :help q:back [*:8080]
```



**Let's start with the  
exercises now!**



# Exercise 1

- Static malware analysis with Androguard

# Exercise 2

- Static malware analysis with Jadx

# Exercise 3

- Hybrid malware analysis with Codeinspect

# Recap

- Understand the different types of mobile malware
- Understand why mobile malware can be important within a forensic investigation
- Know the 3 different types of analysis techniques and understand the limitations
- Understand which of the OWASP Top10 can be also interesting for forensic investigations
- Know the 4 main attack techniques for mobile apps

**See you next week!**

