# Tales of Practical Android Penetration Testing (Mobile Pentest Toolkit)

Alexander Subbotin

**OWASP Bucharest AppSec 2018**

OWASP

The Open Web Application Security Project

OWASP
The Open Web Application Security Project

# • About Me

- IT Security Consultant (https://subbotin.de)
- Penetration Tester/Ethical Hacker with 5 years experience
- Working for enterprise (banking industry, telecommunication companies, wholesale, etc.)
- Trainer for Android and Web Pentesting
- Author and Maintainer of Awesome Pentest Cheatsheets project
  https://github.com/coreb1t/awesome-pentest-cheat-sheets
- Bug Hunter
  Yahoo on HackerOne https://hackerone.com/coreb1t

Alexander
Subbotin
IT-Security Consulting

- Setup Pentest Environment
- Requirements:
  - – Kali like distribution for mobile penetration testing
  - – Updates for most used tools
  - – Extensibility

- Setup Pentest Environment - Current status
- https://github.com/tanprathan/MobileApp-Pentest-Cheatsheet

| Distribution | Notes | Last Update |
|---|---|---|
| MobiSec | Last update 3 years ago | 3 years ago |
| Santoku | Based on Ubuntu 14.04 | |
| Vezir Project | Based on Ubuntu 15.04 | 2,5 years ago |
| Apple | For Window only | 2018-05-08 |
| Android Tamer | Manually updated to last versions of platform-tools, Android SDK, Android Studio and much more | |

- Setup Pentest Environment
- Do we really need to use separated environment/VM?
- 95 % of time we are using the same (few) tools
- adb
- Java Decompiler
- Tools for static analysis
- Tools for dynamic analysis
- Debugger
- Tools allowing runtime modification

- That is how the idea for Mobile-Pentest-Toolkit (MPT) was born
- For each category of tools use just one tool

FRIDA

apktool

abe

drozer

MobSF

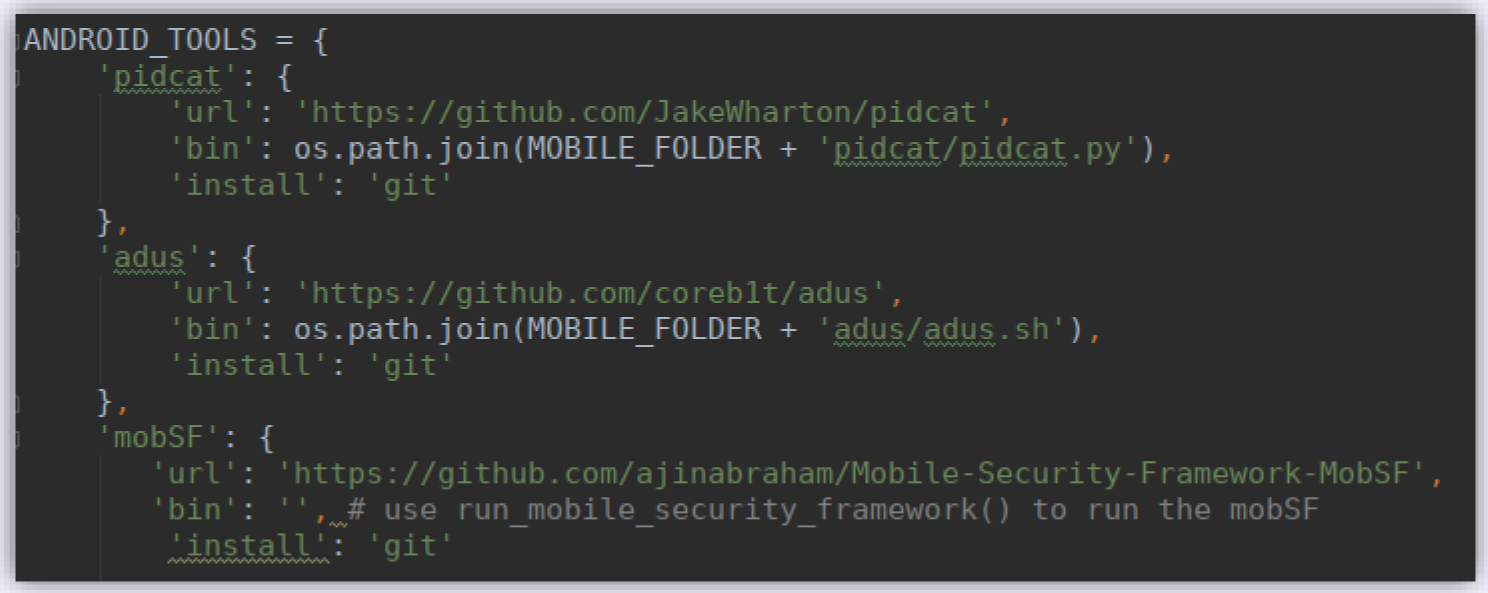
pidcat

signapk

JD-GUI

- Can you remember all the command line parameters for the mentioned tools?
- Example:
- jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore <name> <apk><alias> -storepass <pw>
- frida -R -f <package-name> -l file.js --no-pause
- You have to specify **what** to do and not **how**.
- MPT provides a simplest interface to your tooling related to android security testing.

- Setup Pentest Environment - **Tools**
- MPT implements a simple package manager
- Currently supported git, http, and zip installation

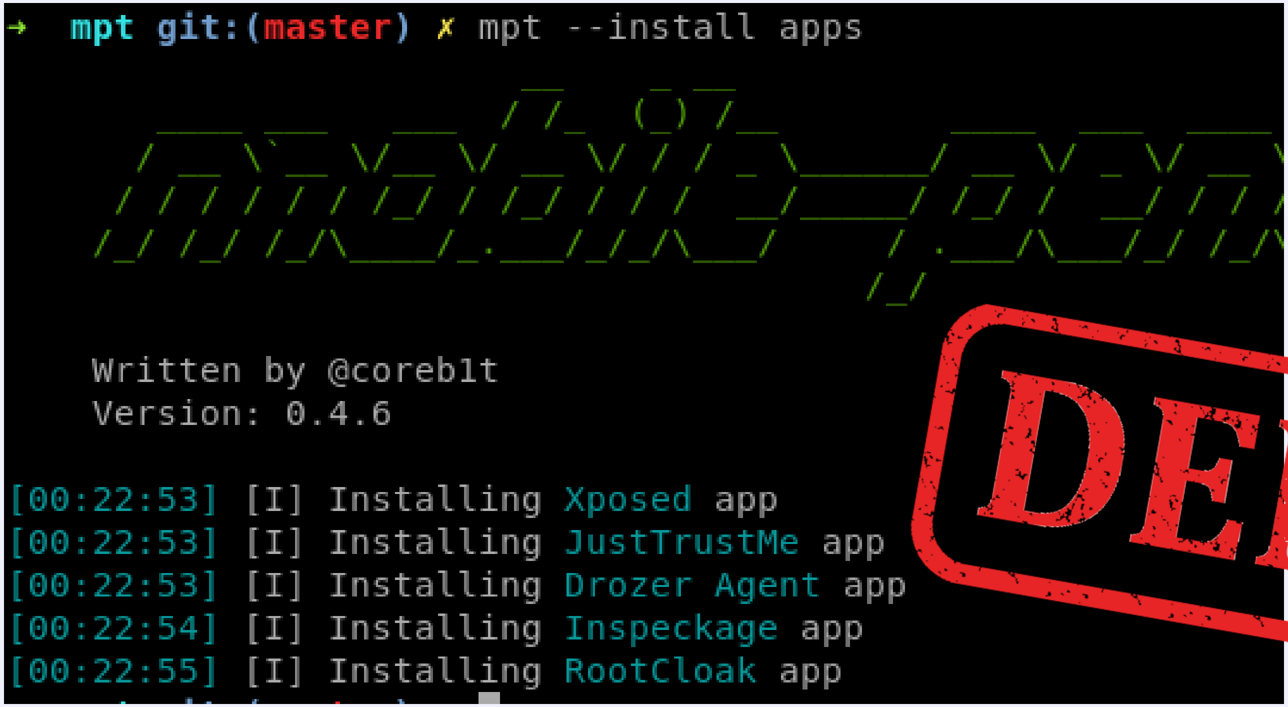


```
ANDROID_TOOLS = {
    'pidcat': {
        'url': 'https://github.com/JakeWharton/pidcat',
        'bin': os.path.join(MOBILE_FOLDER + 'pidcat/pidcat.py'),
        'install': 'git'
    },
    'adus': {
        'url': 'https://github.com/coreb1t/adus',
        'bin': os.path.join(MOBILE_FOLDER + 'adus/adus.sh'),
        'install': 'git'
    },
    'mobSF': {
       'url': 'https://github.com/ajinabraham/Mobile-Security-Framework-MobSF',
       'bin': '', # use run_mobile_security_framework() to run the mobSF
        'install': 'git'
```

- Setup Pentest Environment - **Device**
- Install Pentest tools
- XposedFramework
- Drozer
- JustTrustMe (xposed plugin)
- Inspeckage (xposed plugin)
- …

OWASP
The Open Web Application Security Project

- Setup Pentest Environment - **Device**
- Install Pentest tools



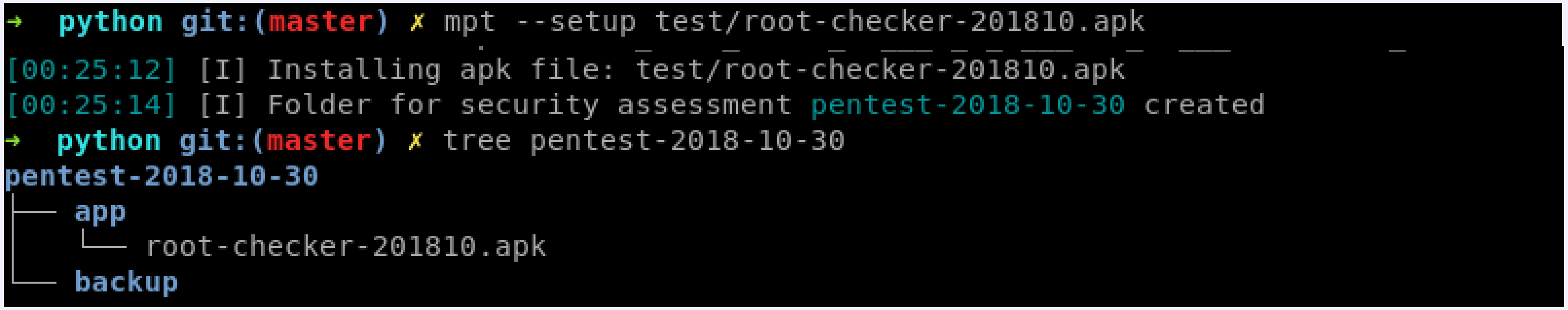
```
➜  mpt git:(master) ✗ mpt --install apps

    Written by @coreb1t
    Version: 0.4.6

[00:22:53] [I] Installing Xposed app
[00:22:53] [I] Installing JustTrustMe app
[00:22:53] [I] Installing Drozer Agent app
[00:22:54] [I] Installing Inspeckage app
[00:22:55] [I] Installing RootCloak app
```

DEMO

- Setup Pentest Environment - **Pentest**
- Install the app
- Create configuration
- Allows to use MPT from everywhere

- Setup Pentest Environment - **Pentest**
- Install the app
- Create configuration
- Allows to use MPT from everywhere

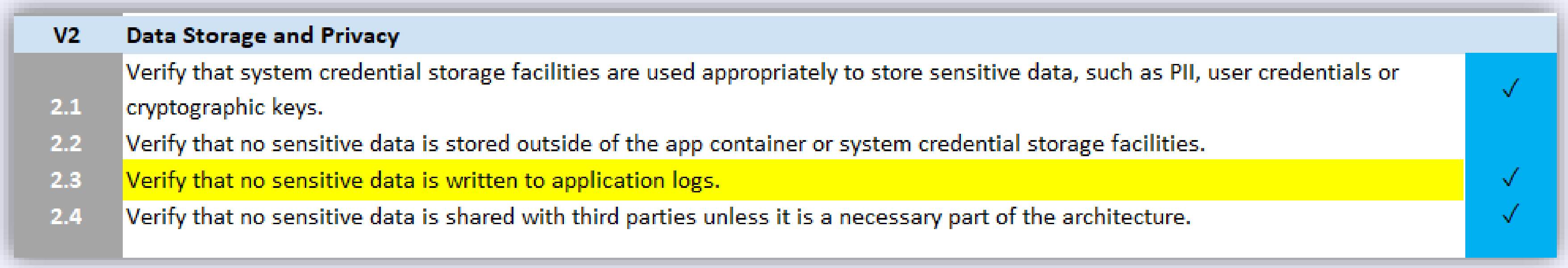
DEMO



```
→  python git:(master) ✗ mpt --setup test/root-checker-201810.apk
[00:25:12] [I] Installing apk file: test/root-checker-201810.apk
[00:25:14] [I] Folder for security assessment pentest-2018-10-30 created
→  python git:(master) ✗ tree pentest-2018-10-30
pentest-2018-10-30
├── app
│   └── root-checker-201810.apk
└── backup
```

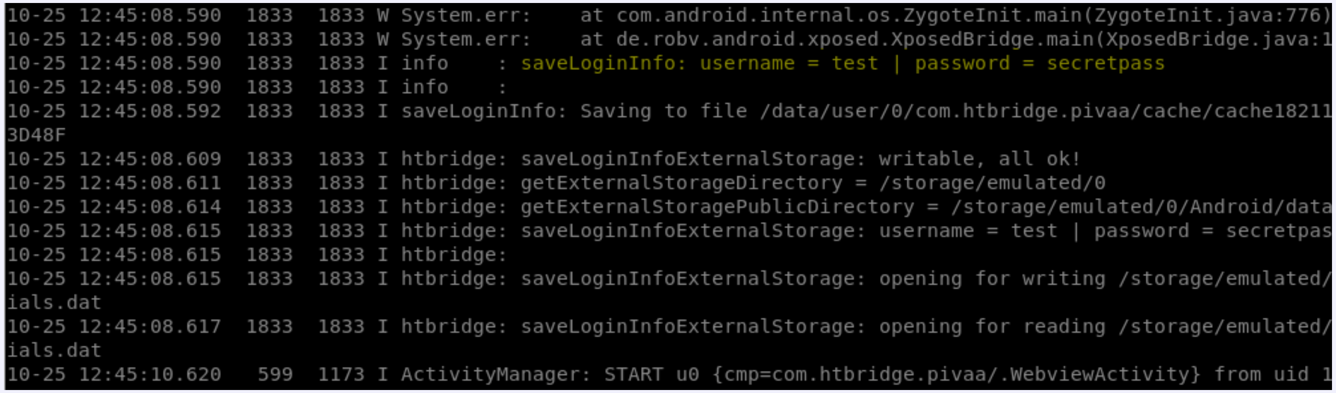
- Starting your favorite tools
- jd-gui (source code review)
- Drozer (android app analysis)
- mobSF (static analysis)
- frida
- and more …

# OWASP testing methodology – Insecure Data Storage



| V2 | Data Storage and Privacy | |
|---|---|---|
| 2.1 | Verify that system credential storage facilities are used appropriately to store sensitive data, such as PII, user credentials or cryptographic keys. | ✓ |
| 2.2 | Verify that no sensitive data is stored outside of the app container or system credential storage facilities. | |
| 2.3 | Verify that no sensitive data is written to application logs. | ✓ |
| 2.4 | Verify that no sensitive data is shared with third parties unless it is a necessary part of the architecture. | ✓ |

## adb logcat    Is the output really readable?



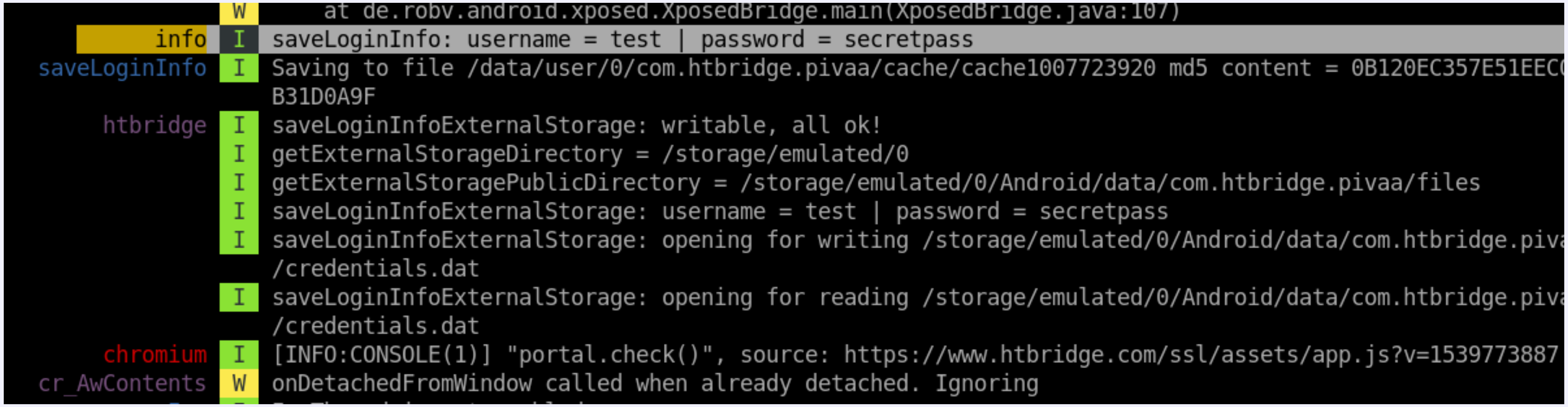
```
10-25 12:45:08.590  1833  1833 W System.err:    at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:776)
10-25 12:45:08.590  1833  1833 W System.err:    at de.robv.android.xposed.XposedBridge.main(XposedBridge.java:1
10-25 12:45:08.590  1833  1833 I info    : saveLoginInfo: username = test | password = secretpass
10-25 12:45:08.590  1833  1833 I info    :
10-25 12:45:08.592  1833  1833 I saveLoginInfo: Saving to file /data/user/0/com.htbridge.pivaa/cache/cache18211
3D48F
10-25 12:45:08.609  1833  1833 I htbridge: saveLoginInfoExternalStorage: writable, all ok!
10-25 12:45:08.611  1833  1833 I htbridge: getExternalStorageDirectory = /storage/emulated/0
10-25 12:45:08.614  1833  1833 I htbridge: getExternalStoragePublicDirectory = /storage/emulated/0/Android/data
10-25 12:45:08.615  1833  1833 I htbridge: saveLoginInfoExternalStorage: username = test | password = secretpas
10-25 12:45:08.615  1833  1833 I htbridge:
10-25 12:45:08.615  1833  1833 I htbridge: saveLoginInfoExternalStorage: opening for writing /storage/emulated/
ials.dat
10-25 12:45:08.617  1833  1833 I htbridge: saveLoginInfoExternalStorage: opening for reading /storage/emulated/
ials.dat
10-25 12:45:10.620   599  1173 I ActivityManager: START u0 {cmp=com.htbridge.pivaa/.WebviewActivity} from uid 1
```

OWASP
The Open Web Application Security Project

# OWASP testing methodology – Insecure Data Storage

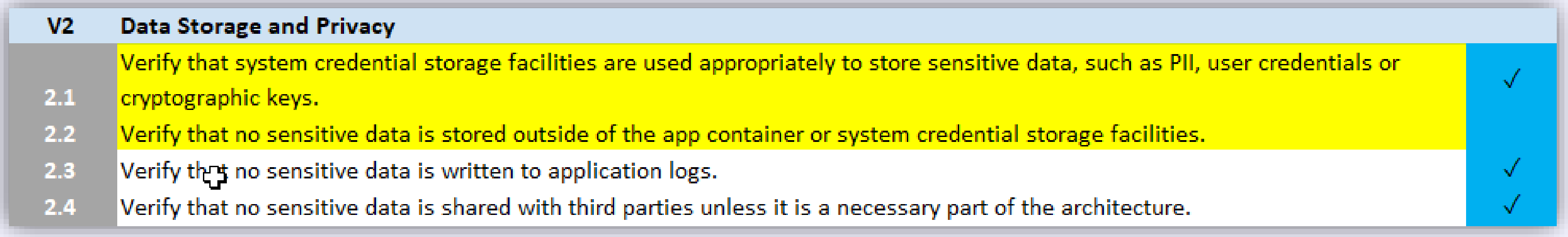
DEMO

Solution: use pidcat
→ colored output for only on process



```
               W      at de.robv.android.xposed.XposedBridge.main(XposedBridge.java:107)
         info  I   saveLoginInfo: username = test | password = secretpass
saveLoginInfo  I   Saving to file /data/user/0/com.htbridge.pivaa/cache/cache1007723920 md5 content = 0B120EC357E51EEC
                   B31D0A9F
     htbridge  I   saveLoginInfoExternalStorage: writable, all ok!
               I   getExternalStorageDirectory = /storage/emulated/0
               I   getExternalStoragePublicDirectory = /storage/emulated/0/Android/data/com.htbridge.pivaa/files
               I   saveLoginInfoExternalStorage: username = test | password = secretpass
               I   saveLoginInfoExternalStorage: opening for writing /storage/emulated/0/Android/data/com.htbridge.piva
                   /credentials.dat
               I   saveLoginInfoExternalStorage: opening for reading /storage/emulated/0/Android/data/com.htbridge.piva
                   /credentials.dat
     chromium  I   [INFO:CONSOLE(1)] "portal.check()", source: https://www.htbridge.com/ssl/assets/app.js?v=1539773887
cr_AwContents  W   onDetachedFromWindow called when already detached. Ignoring
```

# OWASP testing methodology – Insecure Data Storage

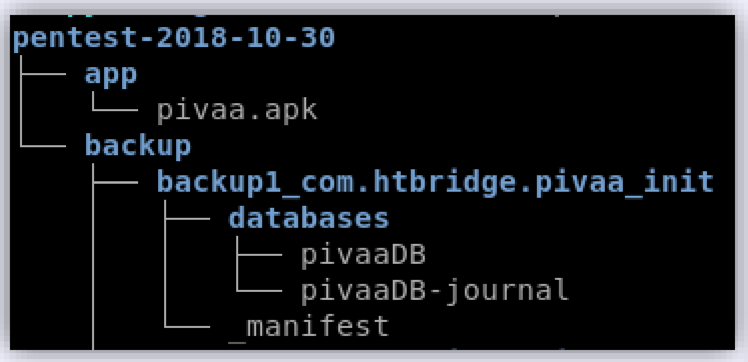


| V2 | Data Storage and Privacy | |
|---|---|---|
| 2.1 | Verify that system credential storage facilities are used appropriately to store sensitive data, such as PII, user credentials or cryptographic keys. | ✓ |
| 2.2 | Verify that no sensitive data is stored outside of the app container or system credential storage facilities. | |
| 2.3 | Verify that no sensitive data is written to application logs. | ✓ |
| 2.4 | Verify that no sensitive data is shared with third parties unless it is a necessary part of the architecture. | ✓ |

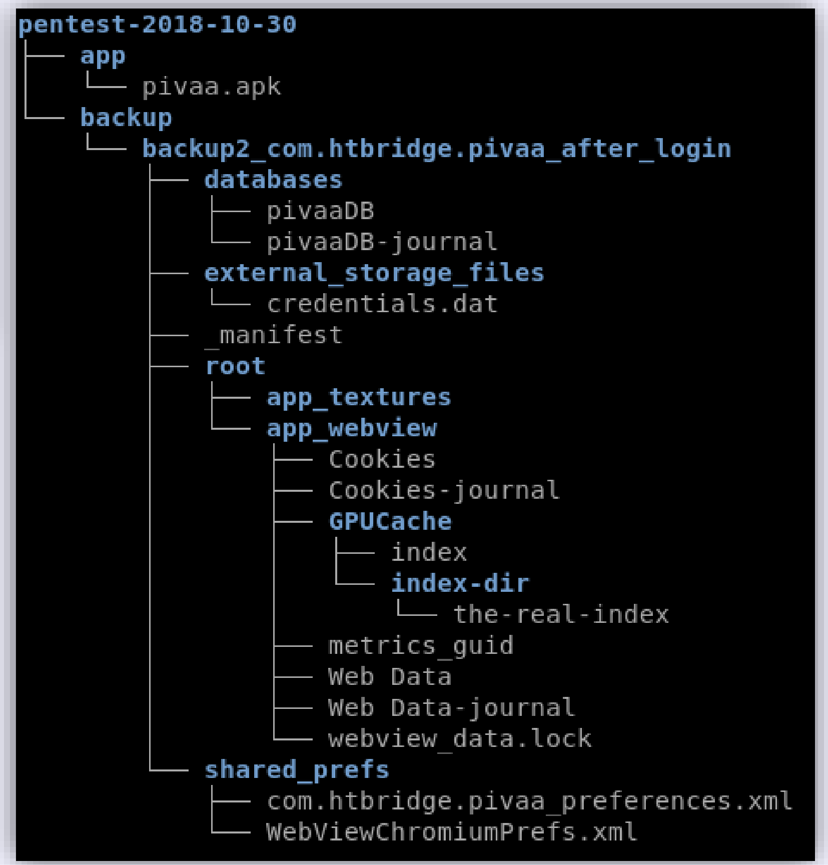
Backup Option
→ Compare two states of application

# OWASP testing methodology – Insecure Data Storage

## Using --backup option create 2 backups for different states



```
pentest-2018-10-30
├── app
│   └── pivaa.apk
└── backup
    ├── backup1_com.htbridge.pivaa_init
    │   ├── databases
    │   │   ├── pivaaDB
    │   │   └── pivaaDB-journal
    │   └── _manifest
```



```
pentest-2018-10-30
├── app
│   └── pivaa.apk
└── backup
    └── backup2_com.htbridge.pivaa_after_login
        ├── databases
        │   ├── pivaaDB
        │   └── pivaaDB-journal
        ├── external_storage_files
        │   └── credentials.dat
        ├── _manifest
        ├── root
        │   ├── app_textures
        │   └── app_webview
        │       ├── Cookies
        │       ├── Cookies-journal
        │       ├── GPUCache
        │       │   ├── index
        │       │   └── index-dir
        │       │       └── the-real-index
        │       ├── metrics_guid
        │       ├── Web Data
        │       ├── Web Data-journal
        │       └── webview_data.lock
        └── shared_prefs
            ├── com.htbridge.pivaa_preferences.xml
            └── WebViewChromiumPrefs.xml
```

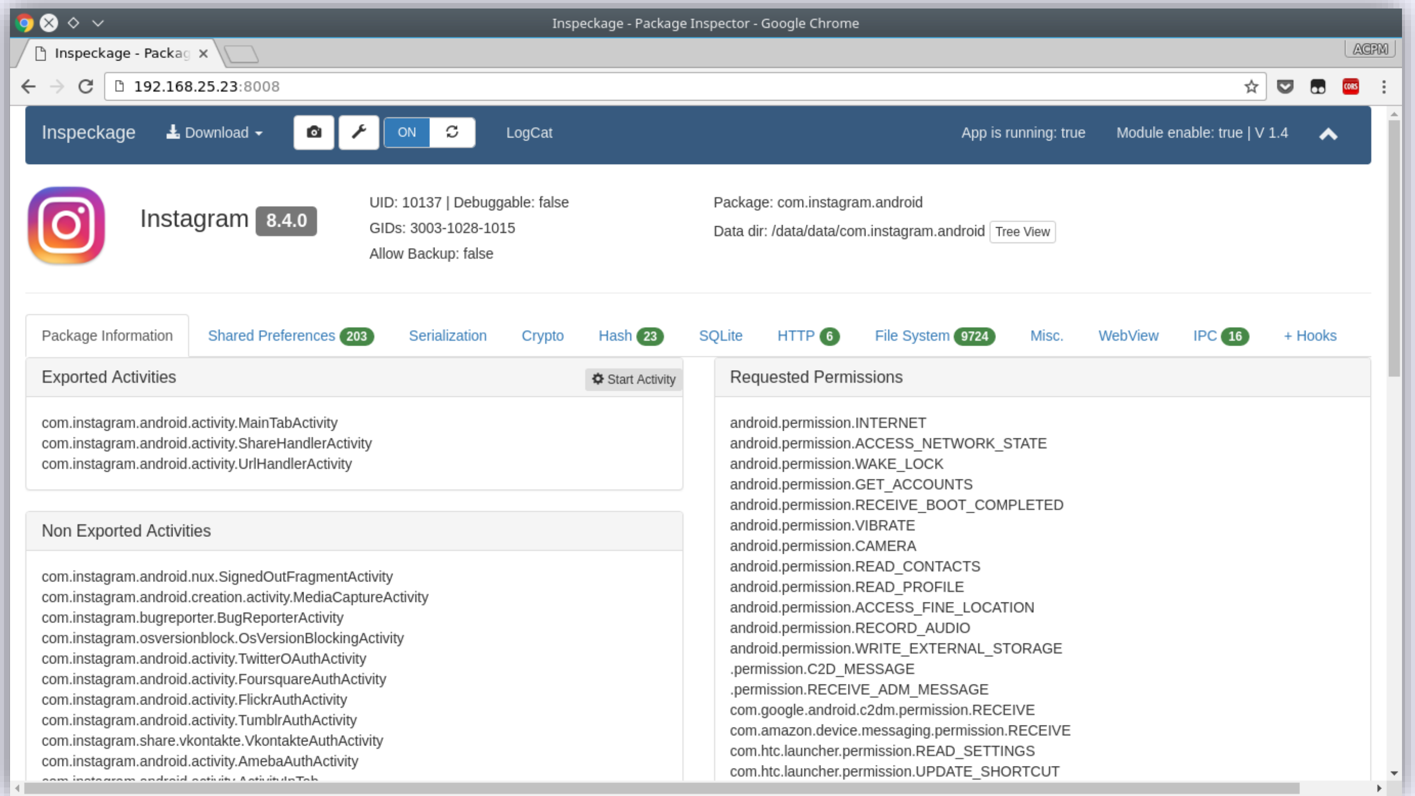
after login the /data/data/<app> folder states differ from each other

Other challenges

- Dynamic analysis
- Dynamic instrumentation and runtime hooking (Frida)
- Root Detection Bypass
- SSL Pinning Bypass

# Other challenges

- Dynamic analysis - Inspeckage



Instagram 8.4.0

UID: 10137 | Debuggable: false
GIDs: 3003-1028-1015
Allow Backup: false

Package: com.instagram.android
Data dir: /data/data/com.instagram.android

Exported Activities

com.instagram.android.activity.MainTabActivity
com.instagram.android.activity.ShareHandlerActivity
com.instagram.android.activity.UrlHandlerActivity

Non Exported Activities

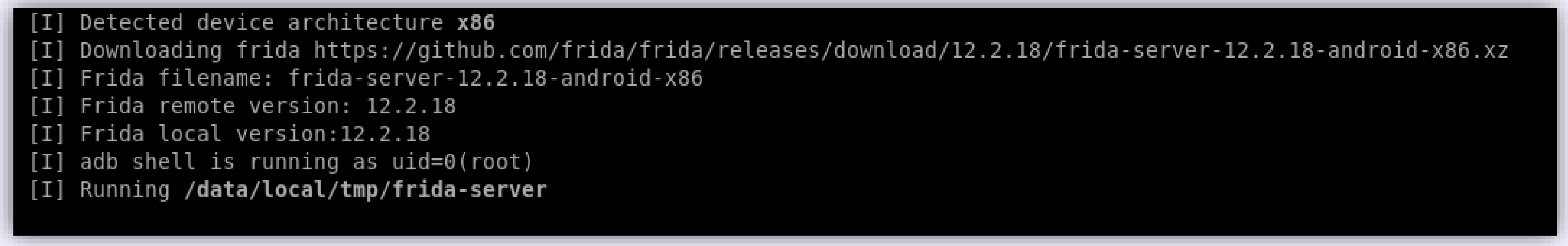
com.instagram.android.nux.SignedOutFragmentActivity
com.instagram.android.creation.activity.MediaCaptureActivity
com.instagram.bugreporter.BugReporterActivity
com.instagram.osversionblock.OsVersionBlockingActivity
com.instagram.android.activity.TwitterOAuthActivity
com.instagram.android.activity.FoursquareAuthActivity
com.instagram.android.activity.FlickrAuthActivity
com.instagram.android.activity.TumblrAuthActivity
com.instagram.share.vkontakte.VkontakteAuthActivity
com.instagram.android.activity.AmebaAuthActivity

Requested Permissions

android.permission.INTERNET
android.permission.ACCESS_NETWORK_STATE
android.permission.WAKE_LOCK
android.permission.GET_ACCOUNTS
android.permission.RECEIVE_BOOT_COMPLETED
android.permission.VIBRATE
android.permission.CAMERA
android.permission.READ_CONTACTS
android.permission.READ_PROFILE
android.permission.ACCESS_FINE_LOCATION
android.permission.RECORD_AUDIO
android.permission.WRITE_EXTERNAL_STORAGE
.permission.C2D_MESSAGE
.permission.RECEIVE_ADM_MESSAGE
com.google.android.c2dm.permission.RECEIVE
com.amazon.device.messaging.permission.RECEIVE
com.htc.launcher.permission.READ_SETTINGS
com.htc.launcher.permission.UPDATE_SHORTCUT

# Other challenges

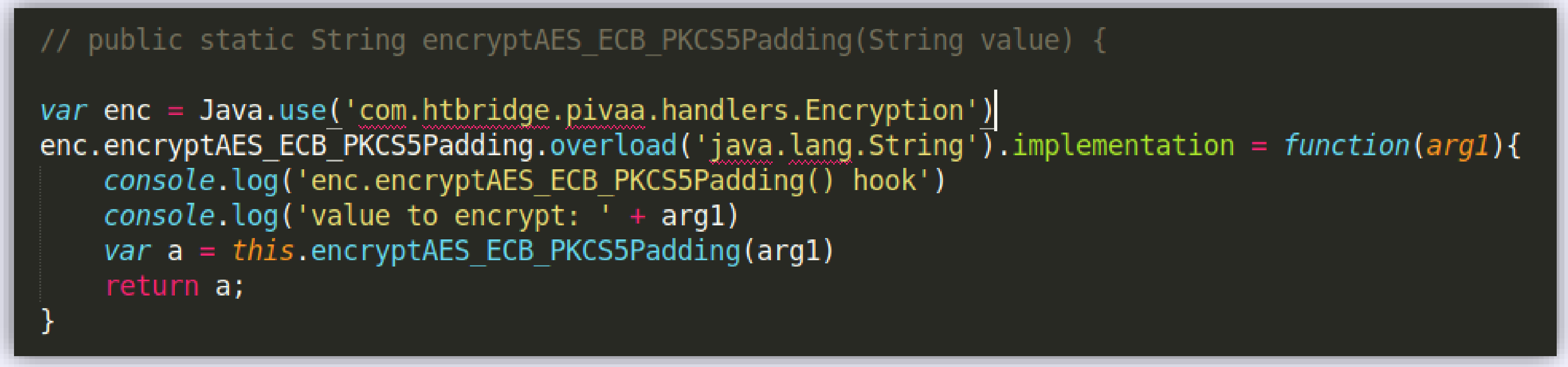
- Dynamic instrumentation and runtime hooking (Frida)
- Download a proper Frida version and execute Frida on the device (--frida option)



```
[I] Detected device architecture x86
[I] Downloading frida https://github.com/frida/frida/releases/download/12.2.18/frida-server-12.2.18-android-x86.xz
[I] Frida filename: frida-server-12.2.18-android-x86
[I] Frida remote version: 12.2.18
[I] Frida local version:12.2.18
[I] adb shell is running as uid=0(root)
[I] Running /data/local/tmp/frida-server
```

# Other challenges

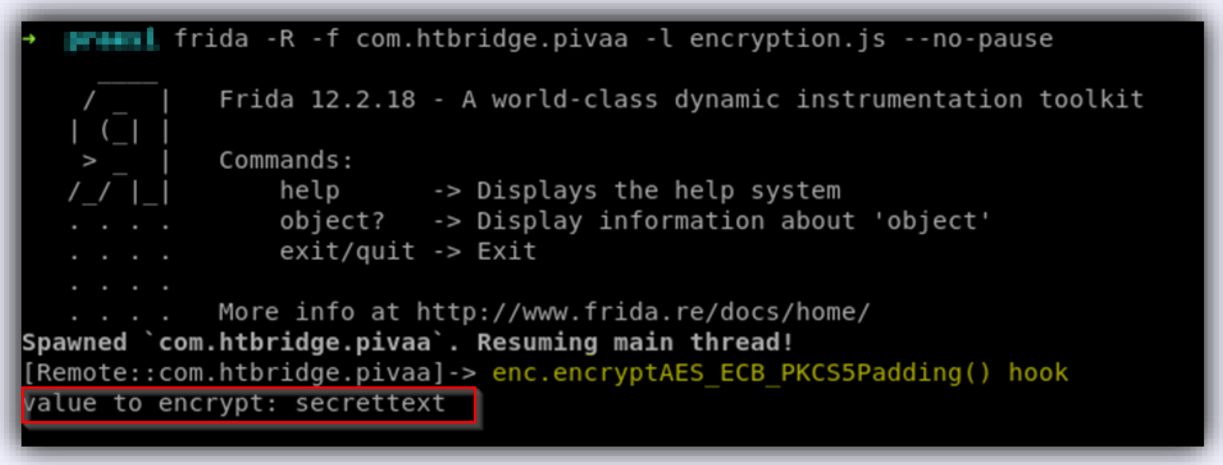
- Dynamic instrumentation and runtime hooking (Frida)
- Use frida to hook cryptographic functions



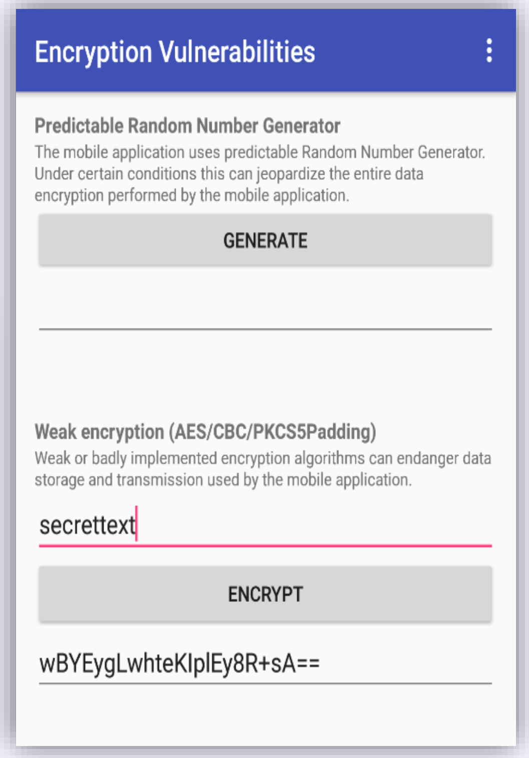
```
// public static String encryptAES_ECB_PKCS5Padding(String value) {

var enc = Java.use('com.htbridge.pivaa.handlers.Encryption')
enc.encryptAES_ECB_PKCS5Padding.overload('java.lang.String').implementation = function(arg1){
    console.log('enc.encryptAES_ECB_PKCS5Padding() hook')
    console.log('value to encrypt: ' + arg1)
    var a = this.encryptAES_ECB_PKCS5Padding(arg1)
    return a;
}
```

# Other challenges

- Dynamic instrumentation and runtime hooking (Frida)
- Use frida to hook cryptographic functions



```
→  [redacted] frida -R -f com.htbridge.pivaa -l encryption.js --no-pause
     ____
    / _  |   Frida 12.2.18 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at http://www.frida.re/docs/home/
Spawned `com.htbridge.pivaa`. Resuming main thread!
[Remote::com.htbridge.pivaa]-> enc.encryptAES_ECB_PKCS5Padding() hook
value to encrypt: secrettext
```



Encryption Vulnerabilities

Predictable Random Number Generator

The mobile application uses predictable Random Number Generator. Under certain conditions this can jeopardize the entire data encryption performed by the mobile application.

Weak encryption (AES/CBC/PKCS5Padding)

Weak or badly implemented encryption algorithms can endanger data storage and transmission used by the mobile application.

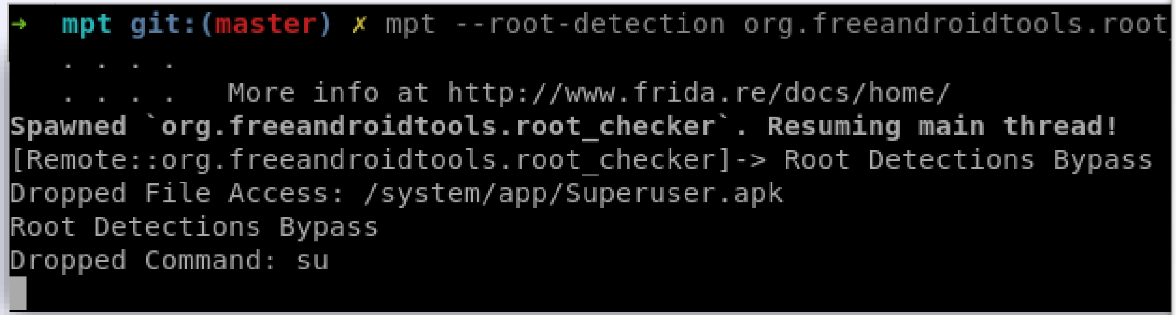
secrettext

wBYEygLwhteKIplEy8R+sA==

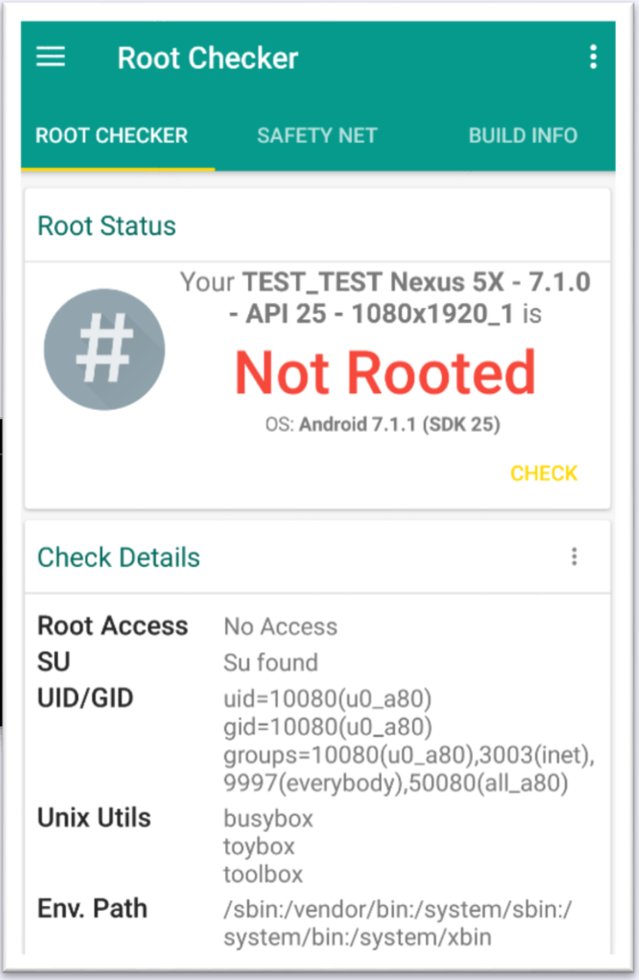
# Other challenges

- Root Detection Bypass

Disable root detection at runtime using frida



```
→  mpt git:(master) ✗ mpt --root-detection org.freeandroidtools.root
   . . . .
   . . . .   More info at http://www.frida.re/docs/home/
Spawned `org.freeandroidtools.root_checker`. Resuming main thread!
[Remote::org.freeandroidtools.root_checker]-> Root Detections Bypass
Dropped File Access: /system/app/Superuser.apk
Root Detections Bypass
Dropped Command: su
```



Root Status

Your **TEST_TEST Nexus 5X - 7.1.0 - API 25 - 1080x1920_1** is

**Not Rooted**

OS: **Android 7.1.1 (SDK 25)**

Check Details

| | |
|---|---|
| **Root Access** | No Access |
| **SU** | Su found |
| **UID/GID** | uid=10080(u0_a80)<br>gid=10080(u0_a80)<br>groups=10080(u0_a80),3003(inet),9997(everybody),50080(all_a80) |
| **Unix Utils** | busybox<br>toybox<br>toolbox |
| **Env. Path** | /sbin:/vendor/bin:/system/sbin:/system/bin:/system/xbin |

# Other challenges

- SSL Pinning Bypass

DEMO

- Other helpful tools
- **Objection** -  is a runtime mobile exploration toolkit, powered by Frida working on not rooted and jailbroken devices.
- https://github.com/sensepost/objection
- **AppMon** - automated framework for monitoring and tampering system API calls of native iOS and android apps
- https://github.com/dpnishant/appmon
- **House** -  runtime mobile application analysis toolkit with a Web GUI, powered by Frida
- https://github.com/nccgroup/house

- MPT - Overview
- Setup Pentest Environment
  - Tools
  - Device
  - Config
- Simple Interface to interact with pentest tools
- Allows to perform static, dynamic analysis
- Support to bypass SSL certificate pinning and root detection
- Supports zsh autocompletion

- Further Ideas
- Automatically rebuild apk with backup and debug flags enabled (in progress)
- Automatically generate PoCs for sending broadcast messages and start activities and services (in progress)
- Integrate file explorer for files on the devices
- Generate Frida hooks for selected code (method) on the fly
- Implement anti-debugging bypass (in progress)

OWASP
The Open Web Application Security Project

# Thank you for your attention!

in Alexander Subbotin

@coreb1t

@coreb1t