קורס אנדרואיד Reverse Engineering

ג'ון ברייס אוגוסט 2023
מרצה: טל מנור

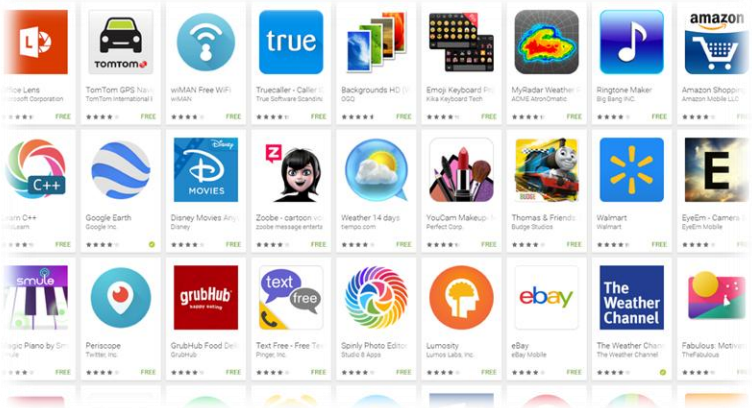
מטרות הקורס

- הנדסה לאחור של אפליקציות אנדרואיד ללא קוד,תיעוד או ידע מוקדם
- במסגרת התרגולים יקבלו התלמידים אפליקציות (apk) לא מוכרות וילמדו לזהות מה מטרתן
- היכרות עם כלים ושיטות בתחום
- יישומי RE ב Malware analysis
- יישומי RE בחקר חולשות
- יישומי RE ב Patching

מבוא לאפליקציות אנדרואיד

מבוא לאפליקציות אנדרואיד

- סוגי אפליקציות
- כלי פיתוח ו-frameworks
- רכיבים עיקריים וארכיטקטורה
- מבנה ופורמט APK
- תהליך קומפילצייה ו build



סוגי אפליקציות

- Native
  - Java / Kotlin
  - NDK (C/C++)
- Hybrid
  - כתוב עם (JS, HTML, CSS) web framework
  - רץ ב webview
  - React Native, Ionic, Cordova
  - משתמש ב Plugins כדי לגשת ליכולות של המכשיר שלא נגישות לדפדפן

כלי פיתוח ו Frameworks

- Android Studio
  - SDK Tools
    - Platform Tools
    - Build Tools
    - CLI Tools
- ADB
- Emulator
- LLDB
- Hybrid Frameworks

רכיבים עיקריים וארכיטקטורה של אפליקציה

- סוגי קבצים:
  - APK
    - פורמט זיפ שמכיל את קבצי האפליקציה כ package להתקנה
  - AAB
    - פורמט של bundle להפצה
- סוגי רכיבים:
  - activities
  - Services
  - Broadcast Receivers
  - Content Providers

- חושפים callbacks לשליטה ב Lifecycle
- רכיב הוא <u>נקודת כניסה</u> לאפליקציה עבור המערכת או המשתמש
- הרכיבים מוגדרים ב androidManifest.XML

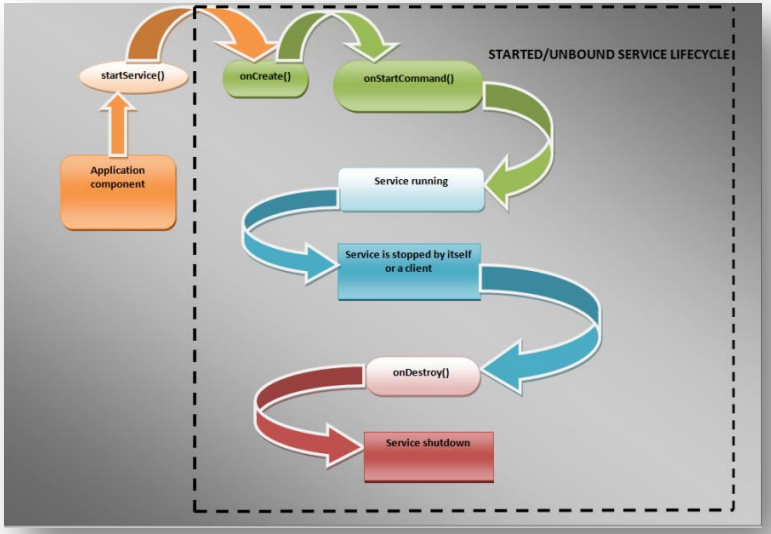
רכיבים עיקריים activities

- נקודת כניסה עבור המשתמש
- מסך UI
- כל activity הוא עצמאי
- אפליקציות אחרות יכולות להפעיל activity ע"י Intents לפי הרשאות

- ניתן להריץ activity ע"י הכלי activity Manager (am)

```
am start -n com.android.insecurebankv2/com.android.insecurebankv2.LoginActivity
```

רכיבים עיקריים Services



- רכיב שרץ ברקע ללא UI
- לדוגמא :מנגן מוזיקה ,מוריד קבצים, מתעדכן מ API
- Foreground
- Background
- Bound
- Started

- ניתן להשתמש ב Service ע"י Intent גם מאפליקציות אחרות כל עוד הוא לא private

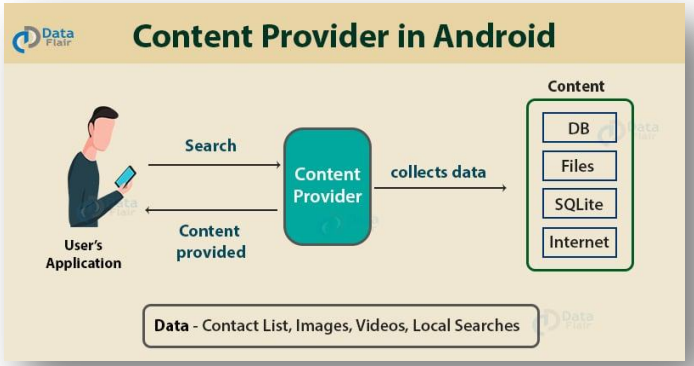
https://developer.android.com/guide/components/services

רכיבים עיקריים Broadcast Receivers

- רכיב שמגיב לאירועים שמפורסמים ע"י המערכת או אפליקציות אחרות
- נרשמים לאירועים ,מקבלים התראה (Intent) ומגיבים
- Static
  - מוגדרים ב Manifest ומופעלים גם אם האפליקציה סגורה
- Dynamic
  - מופעלים רק אם האפליקציה רצה
- דוגמא:
  - android.intent.action.BaTTERY_LOW

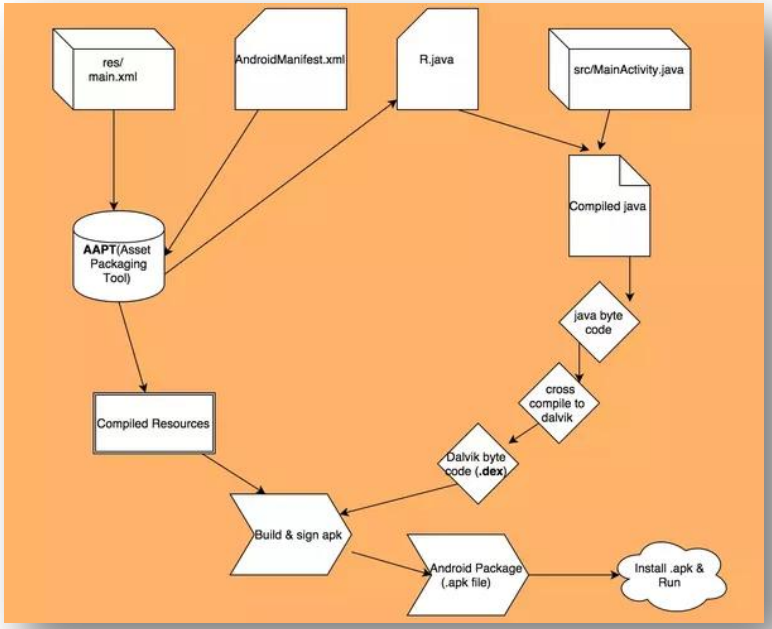
```
<receiver android:name="com.android.tester.C13" android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
    </intent-filter>
</receiver>
```

רכיבים עיקריים Content Providers



- רכיב שנותן גישה ל Data
- אפליקציות אחרות יכולות לבצע שאילותות CRUD
- לדוגמא :
  - Contacts
- הגישה ל Data היא דרך URI Content
  content://
- לא מקבלים Intent אלא מופעלים דרך Content Resolver

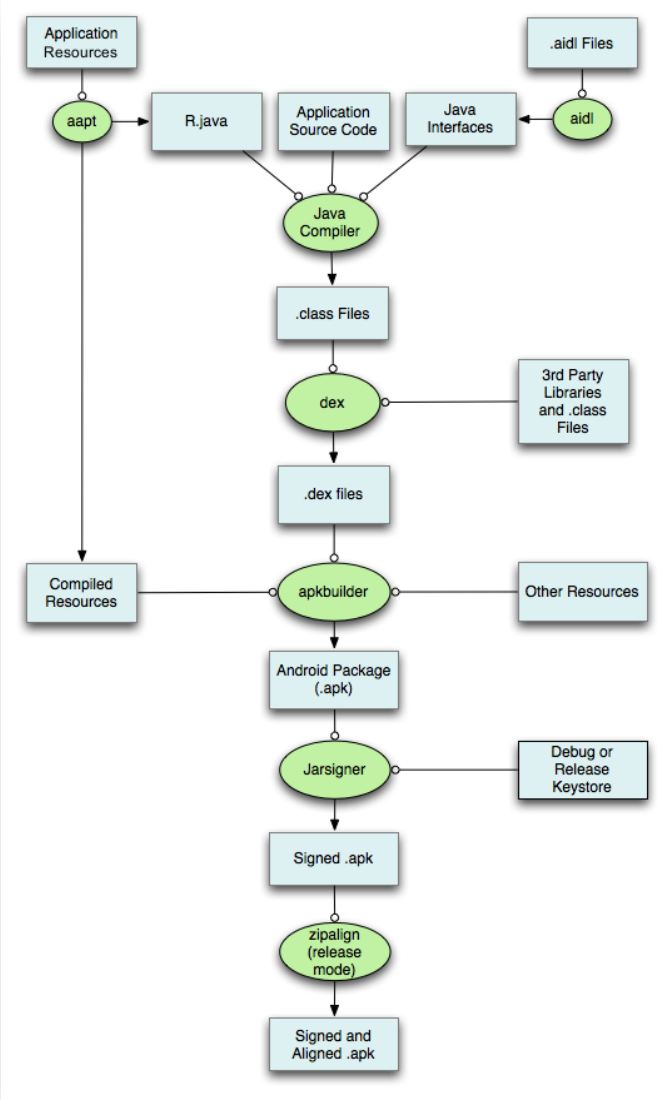
תהליך קומפילצייה \ Build



1. הקוד נכתב ב JaVa או KOTLIN
2. הקוד מקומפל ל Java Class בפורמט bytecode
   - https://www.geeksforgeeks.org/java-class-file/
   - לכל שפה יש קומפיילר משלה javac, kolinc אבל ה Bytecode זהה
3. ה Java bytecode מקומפל ל Dalvik byte code ע"י הכלי D8 ונוצר קובץ classes.dex
4. קובץ זה רץ על ART runtime (או DVM בגרסאות ישנות)
5. ה APK מיוצר ע"י הכלים gradle, aapt, apkbuilder כולל dependencies
6. ה APK נחתם עם self signed certificate ע"י apksigner
7. כדי שהמערכת תוכל לקרוא את הקבצים הדחוסים מתבצע byte alignment עם zipalign

https://developer.android.com/build

תהליך קומפילצייה \ Build



Manifest

- קובץ XML המגדיר את הרכיבים ,ההרשאות ודרישות של האפליקציה מהמערכת , כלי Build ו Google Play
- שמור בפורמט binary xml axml ע"י הכלי aapt2 https://developer.android.com/tools/aapt2

```
aapt2 dump xmltree ..\apks\chrome.apk --file AndroidManifest.xml
```

- לכל רכיב הגדרה משלו כולל שם ה class
- Intent filters מגדירים איך הרכיב מופעל
- דרישות מינימליות של חומרה וגרסאות בשביל להתקין את האפליקציה
- אם אין הגדרה ב manifest המערכת לא תתחיל את הרכיב גם אם הוא מוגדר בקוד

AAPT2

```
aapt2 dump sub-command filename.apk [options]
```

Sub-command	Description
`apc`	Prints the contents of the AAPT2 Container (APC) generated during compilation.
`badging`	Prints information extracted from the APK's manifest.
`configurations`	Prints every configuration used by a resource in the APK.
`overlayable`	Prints the overlayable resources of the APK.
`packagename`	Prints the APK's package name.
`permissions`	Prints the permissions extracted from the APK's manifest.
`strings`	Prints the contents of the APK's resource table string pool.
`styleparents`	Prints the parents of styles used in the APK.
`resources`	Prints the contents of the APK's resource table.
`xmlstrings`	Prints strings from the APK's compiled XML.
`xmltree`	Prints a tree of the APK's compiled XML.

```
aapt2 diff first.apk second.apk
```

https://developer.android.com/tools/aapt2

תרגיל: זיהוי רכיבי אפליקציה

- בתיקיית הקורס פתחו את manifests\chrome.xml
- זהו את רכיבי האפליקציה העיקריים שלמדנו
- כתבו סקריפט בפייתון או שפה אחרת להדפיס את כל הרכיבים:

```
activity: "org.chromium.chrome.browser.usage_stats.UsageStatsConsentActivity"
activity: "com.google.android.gms.common.api.GoogleApiActivity"
activity: "com.google.android.play.core.missingsplits.PlayCoreMissingSplitsActivity"
activity: "com.google.android.play.core.common.PlayCoreDialogWrapperActivity"
activity: "com.google.android.libraries.hats20.SurveyPromptActivity"
activity: "com.google.p010ar.core.InstallActivity"
activity: "com.google.android.libraries.notifications.entrypoints.systemtray.SystemTrayActivity"
provider: "org.chromium.chrome.browser.provider.ChromeBrowserProvider"
provider: "org.chromium.chrome.browser.util.ChromeFileProvider"
provider: "org.chromium.chrome.browser.download.DownloadFileProvider"
provider: "com.google.android.apps.chrome.autofill.AutofillDataProvider"
provider: "com.google.android.apps.chrome.icing.IcingProvider"
receiver: "org.chromium.chrome.browser.sharing.click_to_call.ClickToCallMessageHandler.PhoneUnlockedReceiver"
receiver: "org.chromium.chrome.browser.upgrade.PackageReplacedBroadcastReceiver"
receiver: "org.chromium.chrome.browser.locale.LocaleChangedBroadcastReceiver"
receiver: "org.chromium.chrome.browser.browserservices.ClientAppBroadcastReceiver"
```

- איזה תכונות מעניינות יש לרכיבים (למשל exported, debuggable)?

* מומלץ גם להוריד apk כלשהו מהמכשיר ולבצע את התרגיל עם aapt2

מחזור חיים של activity

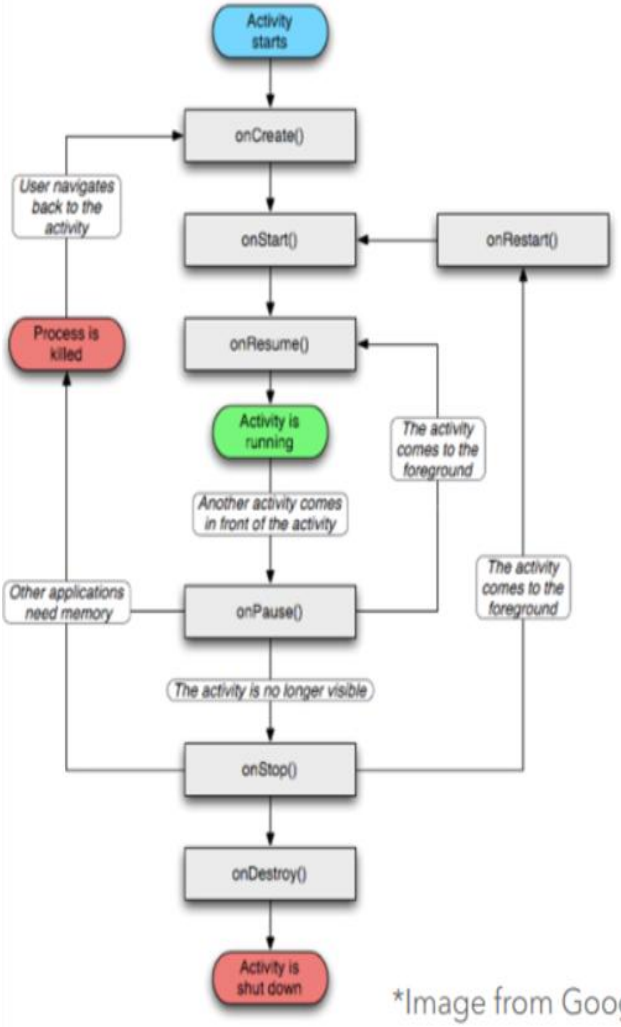


*Image from Google Developer Site

מחזור חיים של Service

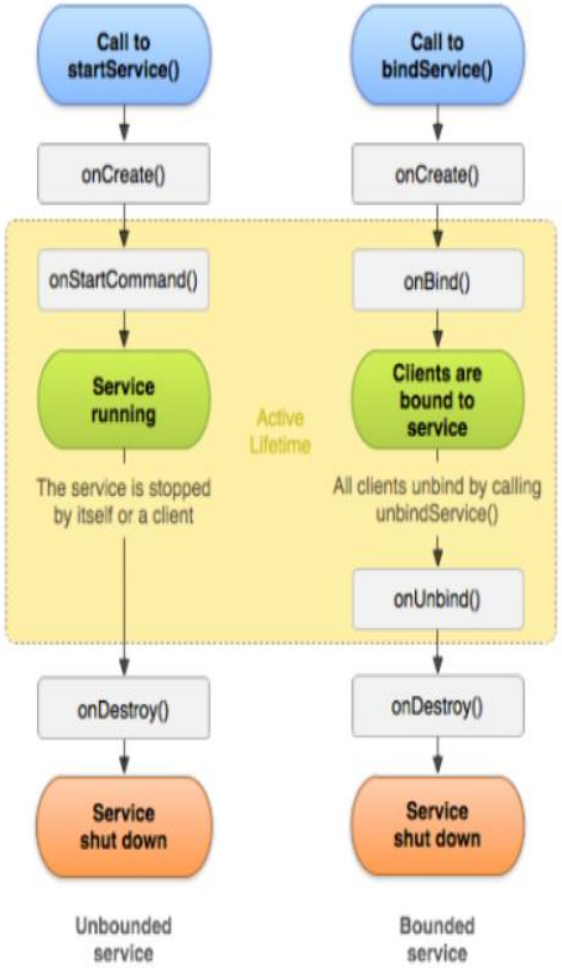
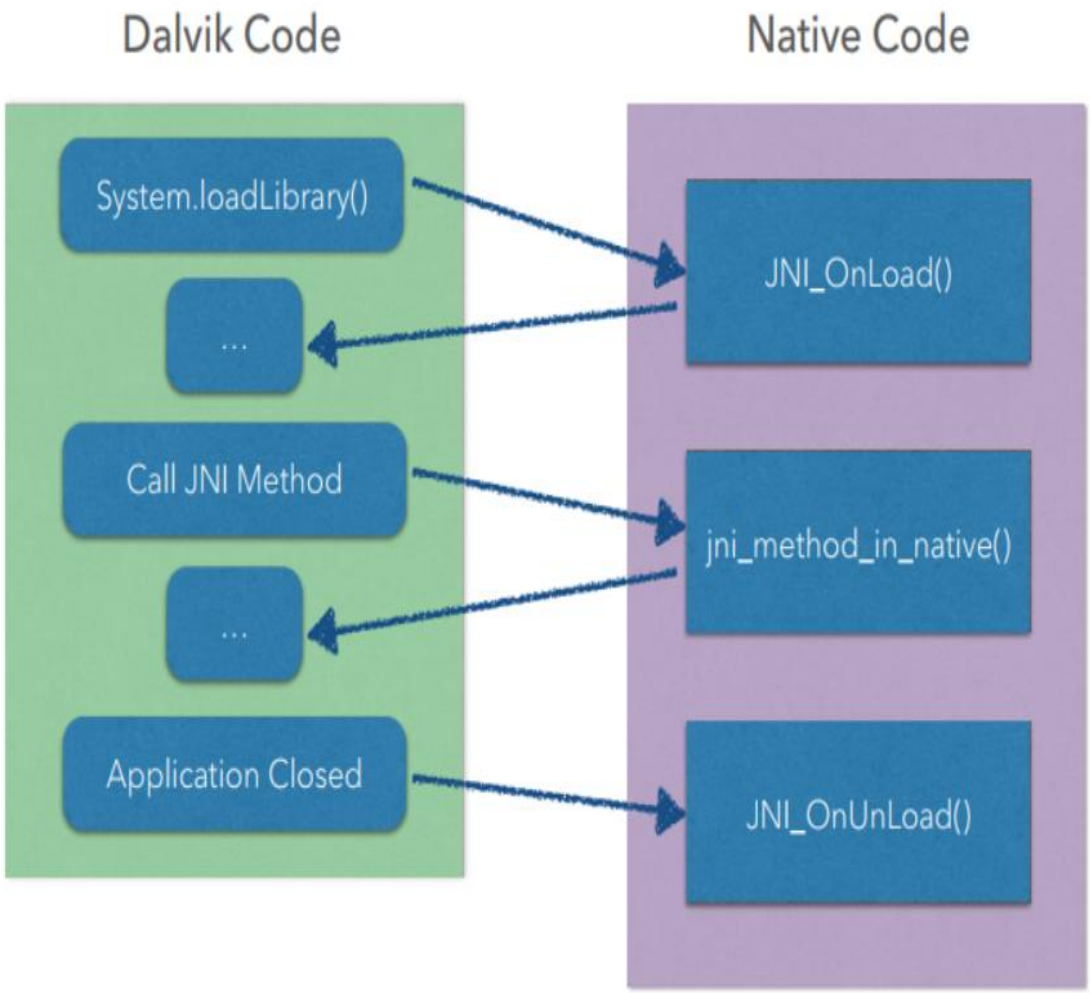


*Image from Google Developer Site

מחזור חיים Native

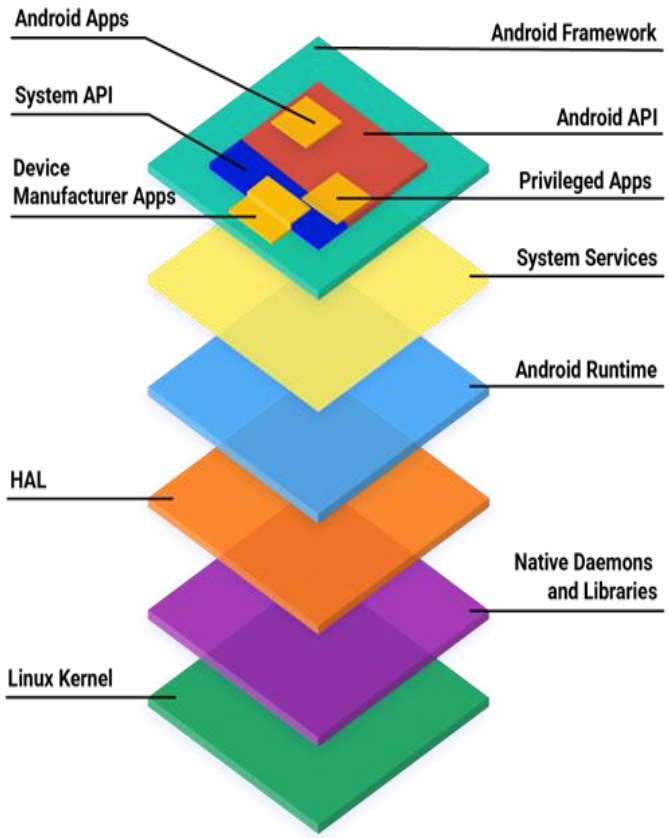


תרגיל : בניית APK

- כתיבת אפליקציה עם כל הרכיבים
- **Targil-building-apk** בתיקיית הקורס
- קישורים להמשך לימוד:
  - https://www.geeksforgeeks.org/introduction-to-activities-in-android/
  - https://www.geeksforgeeks.org/services-in-android-with-example/?ref=lbp
  - https://www.geeksforgeeks.org/broadcast-receiver-in-android-with-example/
  - https://www.geeksforgeeks.org/content-providers-in-android-with-example/

מבוא למערכת ההפעלה Android

ארכיטקטורת אנדרואיד

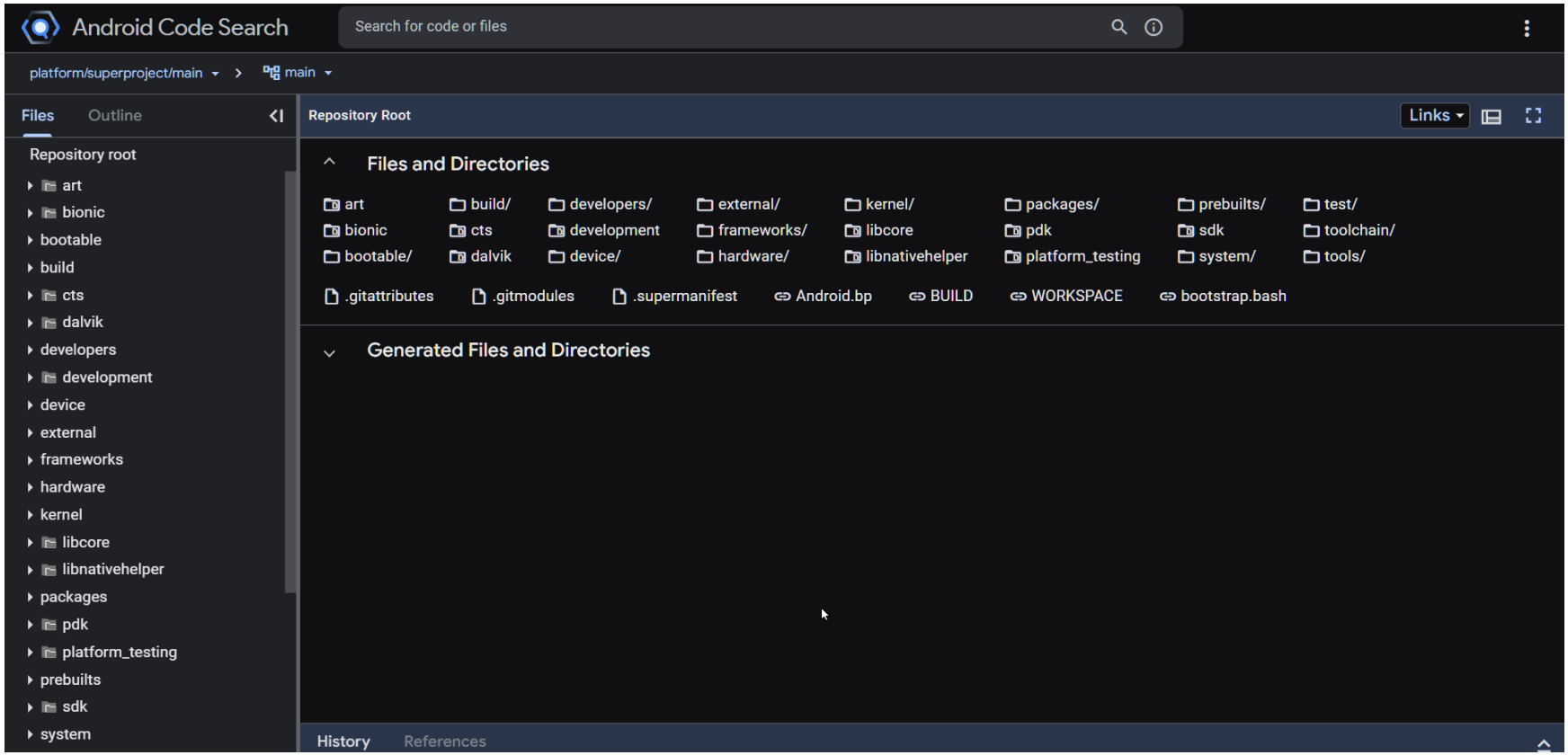


הבסיס: AOSP
Android Open Source Project

https://source.android.com/docs/core/architecture

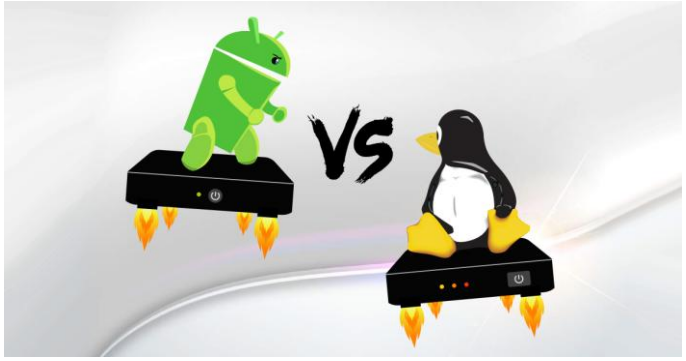
Android Open Source Project (AOSP)

https://cs.android.com/android/platform/superproject/main

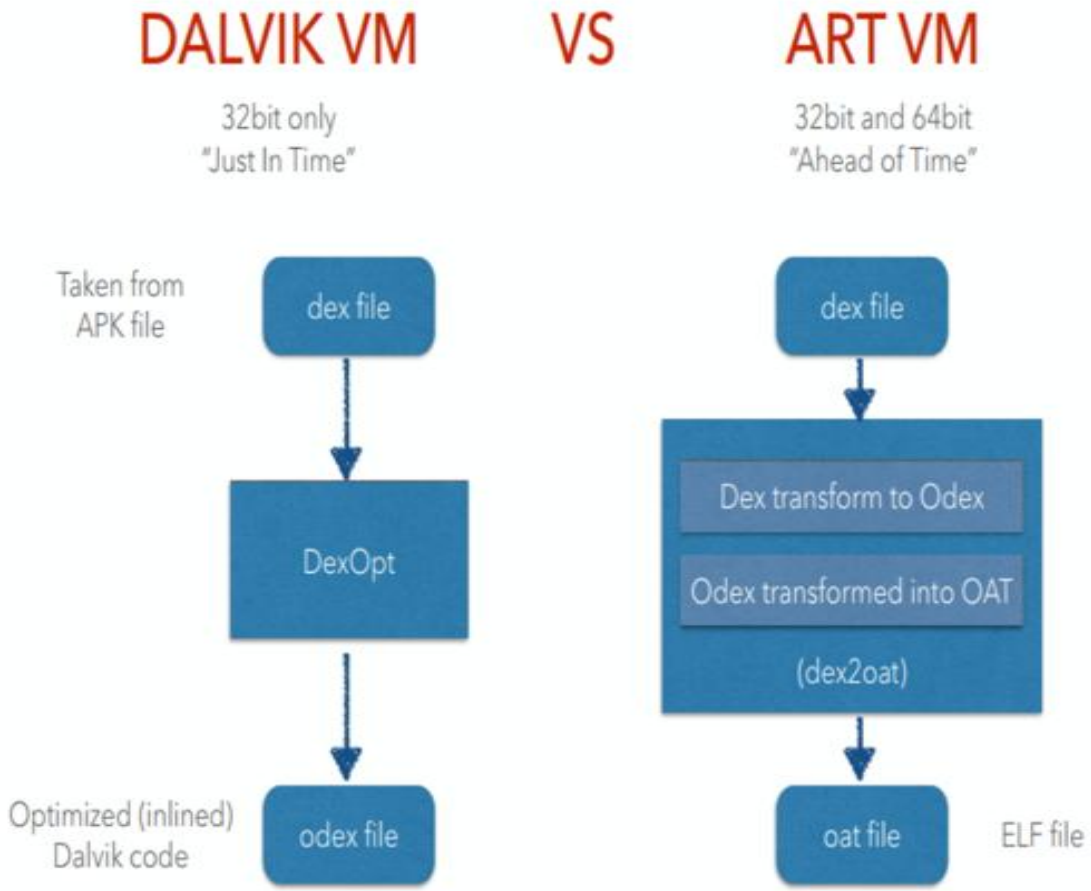


Android vs. Linux User Space

- אנדרואיד מותאם במיוחד למכשירים ניידים ומסתמך על רכיבים מותאמים אישית כמו ART ו-android SDK
- לינוקס user space נועד יותר למטרות כלליות וכולל כלי עזר וספריות סטנדרטיים של GNU
- ה- ART של אנדרואיד מריץ בייטקוד, בעוד שלינוקס בדרך כלל מריץ יישומים כ binaries.
- ה- android SDK מספק API לפיתוח אפליקציות לנייד
- יישומי אנדרואיד ארוזים כקובצי APK בעוד שיישומי לינוקס מופצים בפורמטים של חבילות ספציפיות להפצת לינוקס



ART Runtime



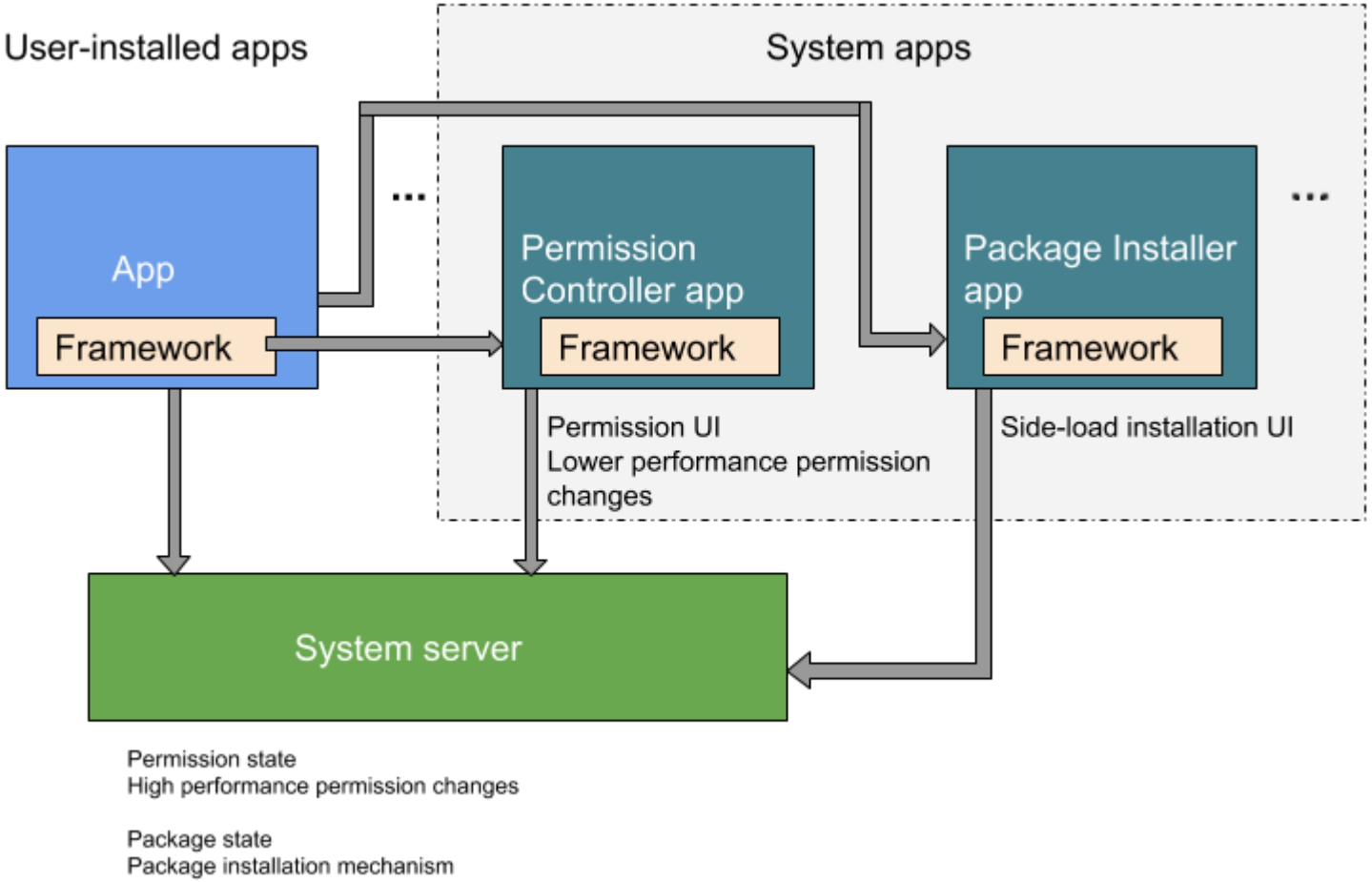
- AOT: ahead of Time compilation:
  - בזמן ההתקנה ART מקמפל אפליקציות ע"י הכלי dex2oat
  - Dex2oat מותקן על המכשיר

הרשאות

- הרשאות אנדרואיד מספקות בקרות שמגבירות את מודעות המשתמש ומגבילות את הגישה של אפליקציה לנתונים רגישים.
- הגדרת הרשאות ב- android 8.0 ומטה כוללת allow listing, שבלעדיו אפליקציות מורשות מושבתות, גם אם הן נמצאות בנתיב priv-app.
- ב- android 9 ומעלה, מכשיר שמנסה להשתמש באפליקציות שאינן רשומות כראוי לא יאותחל.
- אנדרואיד 10 הציגה את הרעיון של role, שם ייחודי במערכת הקשור לדרישות והרשאות מסוימות. אפליקציות מקבלות role כדי להעניק להם הרשאות למטרה מסוימת, ותפקידי ברירת מחדל
- הגנות המוגברות מפני אפליקציות פוטנציאליות מזיקות (PHa) מזהות התנהגות אפליקציה שעלולה להזיק.

https://source.android.com/docs/core/permissions

הרשאות התקנה

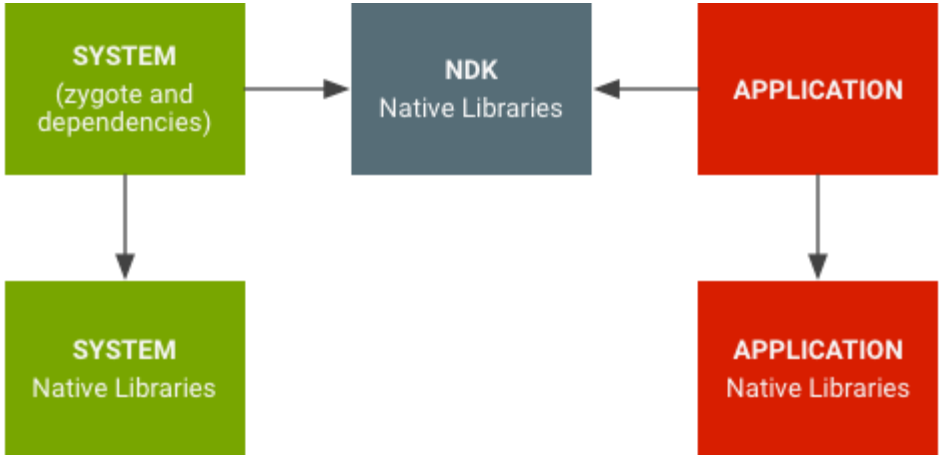


https://source.android.com/docs/core/permissions

Allowlists

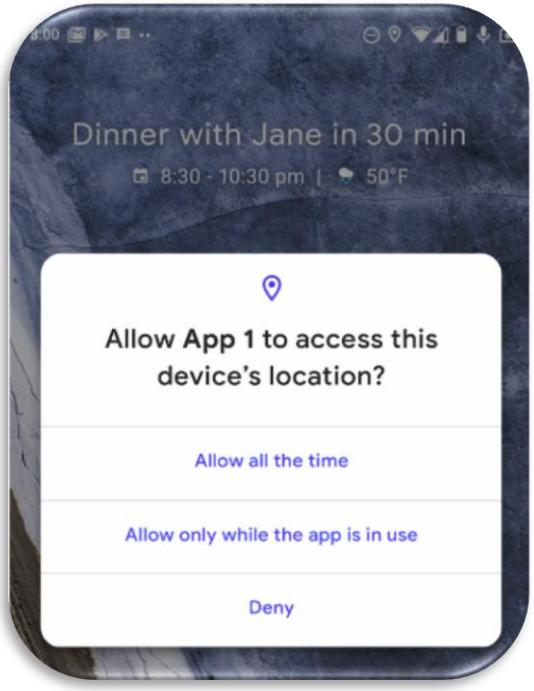
- באנדרואיד 6.0 ומעלה, אפליקציות מבקשות גישה להרשאות מסוכנות בזמן ריצה. אנדרואיד 10 מוסיף הרשאות זמן ריצה לזיהוי פעילות aR שמבקשות מהמשתמש לשנות או לאפשר הרשאות מסוכנות.
- אנדרואיד 8.0 דרש לרשום באופן מפורש אפליקציות פריבילגיות בקובצי ה- XML בספריית /etc/permissions/
- ב- 9 android ומעלה יש לרשום הרשאות מורשות או שהמכשיר לא יכול לאתחל.
- כדי להגביל את החשיפה של ממשק API ולמנוע מאפליקציות לגשת בטעות לספריות פלטפורמה, אנדרואיד 7.0 הציגה namespaces לספריות מקוריות כדי להפריד בין ספריות מערכת מספריות יישומים.
- יצרני מכשירים יכולים להוסיף ספריות מקוריות משלהם.
- החל מ Android 10 לאפליקציות חייבות להיות הרשאות חתימה והסכמת משתמש כדי לגשת לתוכן המסך של המכשיר.
- אפליקציות מועדפות המסתמכות על silent capture, כגון צילום מסך, צריכות להשתמש ב MediaProjection במקום זאת.

Namespaces


SYSTEM
(zygote and dependencies)
NDK
Native Libraries
APPLICATION
SYSTEM
Native Libraries
APPLICATION
Native Libraries

פרטיות

- באנדרואיד 6.0 ואילך, כתובת MAC של המכשיר מוגנת.
- החל מ- Android 10 הגישה של אפליקציות ל device ID דורשת הרשאות.
- ב- android 9 ומטה הרשאת מיקום היא קבועה. החל באנדרואיד 10, יש שלוש אפשרויות לאפשר לאפליקציה גישה למיקום המכשיר.



Privileged app

- אפליקציה שנוצרה באמצעות שילוב של ממשקי API של אנדרואיד ומערכת.
- יש להתקין מראש אפליקציות אלה כאפליקציות מורשות במכשיר.
- מותקנות ב /system/priv-app
- צריכות להיות חתומות עם מפתח ה Platform כלומר מפתח של הספק או מי שבנה את ה AOSP
- צריכות להיות מותקנות ב system, product או vendor
- רצות עם משתמש גבוה android.uid.system
- הרשאות מוגדרות ב -/etc/permissions/privapp

https://source.android.com/docs/core/permissions/perms-allowlist

מנגנוני אבטחה

Sandbox

- פלטפורמת אנדרואיד מנצלת את ההגנה המבוססת על משתמשי לינוקס כדי לזהות ולבודד משאבי אפליקציה.
- לשם כך, אנדרואיד מקצה מזהה משתמש ייחודי (UID) לכל אפליקציית אנדרואיד ומפעילה אותה בתהליך משלה.
- אנדרואיד משתמש ב- UID זה כדי להגדיר ארגז חול לאפליקציה ברמת KERNEL.

מנגנוני אבטחה

app Signing

- חתימת אפליקציה מאפשרת למפתחים לזהות את מפתח האפליקציה ולעדכן את האפליקציה שלהם מבלי ליצור ממשקים והרשאות מסובכים.
- כל אפליקציה שפועלת על פלטפורמת אנדרואיד חייבת להיות חתומה על ידי המפתח.

מנגנוני אבטחה

Biometric API

- אנדרואיד 9 ואילך כולל API BiometricPrompt שמפתחי אפליקציות יכולים להשתמש בו כדי לשלב אימות ביומטרי באפליקציות שלהם
- רק ביומטריה חזקה נתמכת עם BiometricPrompt

מנגנוני אבטחה

Keystore

- אנדרואיד Keystore מגובה חומרה מספק יצירת מפתחות, ייבוא וייצוא של מפתחות אסימטריים, ייבוא מפתחות סימטריים גולמיים, הצפנה ופענוח אסימטריים ועוד.

מנגנוני אבטחה

Security-Enhanced Linux

- כחלק ממודל האבטחה של אנדרואיד, אנדרואיד משתמשת ב- Security Enhanced Linux (SELinux) כדי לאכוף בקרת גישה חובה (MaC) על כל התהליכים, אפילו תהליכים הפועלים עם הרשאות ROOT

מנגנוני אבטחה

Trusty Trusted Execution Environment (TEE)

- Trusty היא מערכת הפעלה מאובטחת עבור אנדרואיד.

- מערכת ההפעלה Trustyפועלת על אותו מעבד כמו מערכת ההפעלה אנדרואיד, אך Trusty מבודדת משאר המערכת ברמת חומרה ותוכנה.

מנגנוני אבטחה

Verified Boot

- אתחול מאומת שואף להבטיח שכל הקוד המופעל מגיע ממקור מהימן (בדרך כלל יצרני OEM של מכשירים)
- מקים שרשרת אמון מלאה, החל משורש אמון מוגן בחומרה ל boot loader , למחיצת האתחול ולמחיצות מאומתות אחרות.

מבוא ל Reverse Engineering

מבוא ל- Reverse Engineering

- הקמת סביבת עבודה
- Rooting
- יתרונות וחסרונות Emulator מול מכשיר פרוץ
- ADB
- תהליך עבודה

הקמת סביבת עבודה

- Android Studio
- NDK
- JADX-GUI
- APKLab VSCODE
- Frida
- Ghidra
- LLDB

כלים מסחריים: JEB

עכשיו נבדוק שכל הכלים הללו מותקנים ועובדים במחשבים שלנו

Rooting

אפשרויות rooting:

1. Bootloader: אם לא נעול, אפשר להוריד את ה ROM מהמכשיר, להוסיף su, לארוז ולטעון מחדש.
2. העלאת הרשאות עם RCE

Rooted Device or Emulator?

יתרונות root:

- סביבת בדיקה מציאותית יותר.
- גישה למערכת קבצים אמיתית כוללת כל התוספות של היצרן ותוספות שאולי לא קיימות באמולטור
- אין הפתעות – האפליקציה נמצאת בסביבה האמיתית שלה ומתנהגת בהתאם

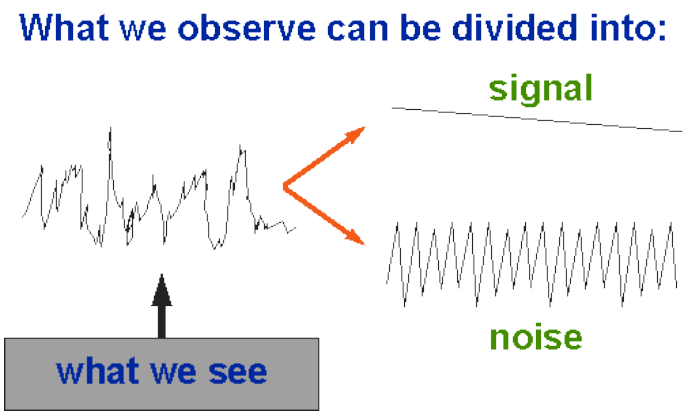
יתרונות emulator:

- סביבה בטוחה יותר
- מהר יותר להקים מאשר תהליך rooting
- snapshots

ADB

- adb devices
- adb shell
- adb kill-server
- adb install
- adb forward
- adb jdwp

תהליך עבודה



- **קביעת המטרה**
  Malware analysis
  Patching
  Exploitation

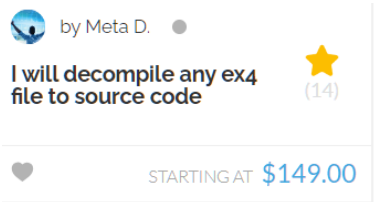
- **תיעדוף**
  קודם כל התנהגות, מניפסט, ספריות
  hooking
  אחר כך לצלול לקוד
  לא לרדוף אחרי צללים (למשל ספריות android)

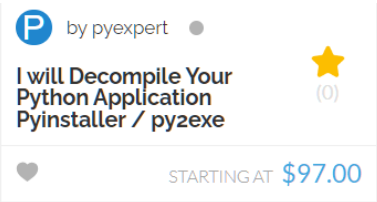
- **תיעוד ודו"ח ממצאים**
  לתעד כל ממצא כולל פרטים טכניים ואיך בדיוק לשחזר!

מבוא לניתוח סטטי

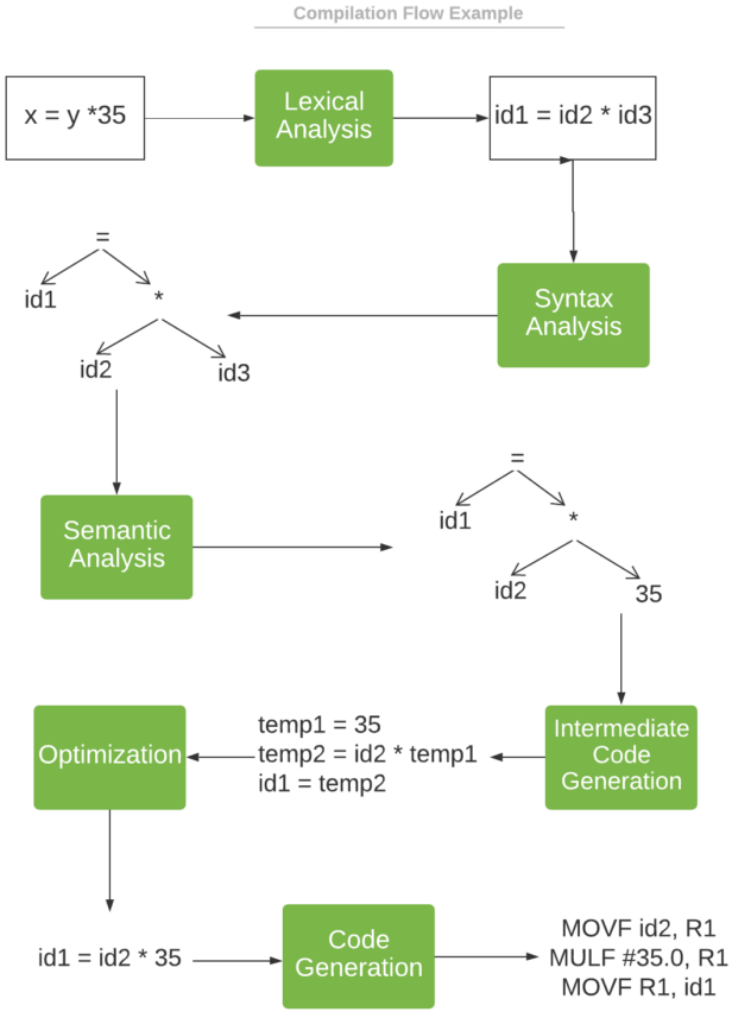
מבוא לניתוח סטטי

- תרגום אחורה לקוד של תוצרי build כגון קבצי DEX או קוד בייטקוד של Java כדי לזהות מבני נתונים, תנאים לוגיים, מידע רגיש, מנגנוני אבטחה ועוד.
- Decompiling
- Disassembling
- Obfuscation/Deobfuscation





Decompiling (native)



https://www.baeldung.com/cs/how-compilers-work

Binary Static analysis


```
Decompile: Java_sg_vantagepoint_helloworldjni_MainActivity_stringFromJNI - (libnative-lib.so)

void Java_sg_vantagepoint_helloworldjni_MainActivity_stringFromJNI(int *param_1)

{
  (**(code **)(*param_1 + 0x29c))(param_1,"Hello from C++");
  return;
}
```

```
public class MainActivity extends AppCompatActivity {
    public native String stringFromJNI();

    static {
        System.loadLibrary("native-lib");
    }
```

מצד המפתח:

- כותבים פונקציות ב C/C++ ובונים ספריית native עם NDK
- מוסיפים את הספרייה ה native ל APK
- קוראים לפונקציות בספריית native עם JNI

מצד ה reverser:

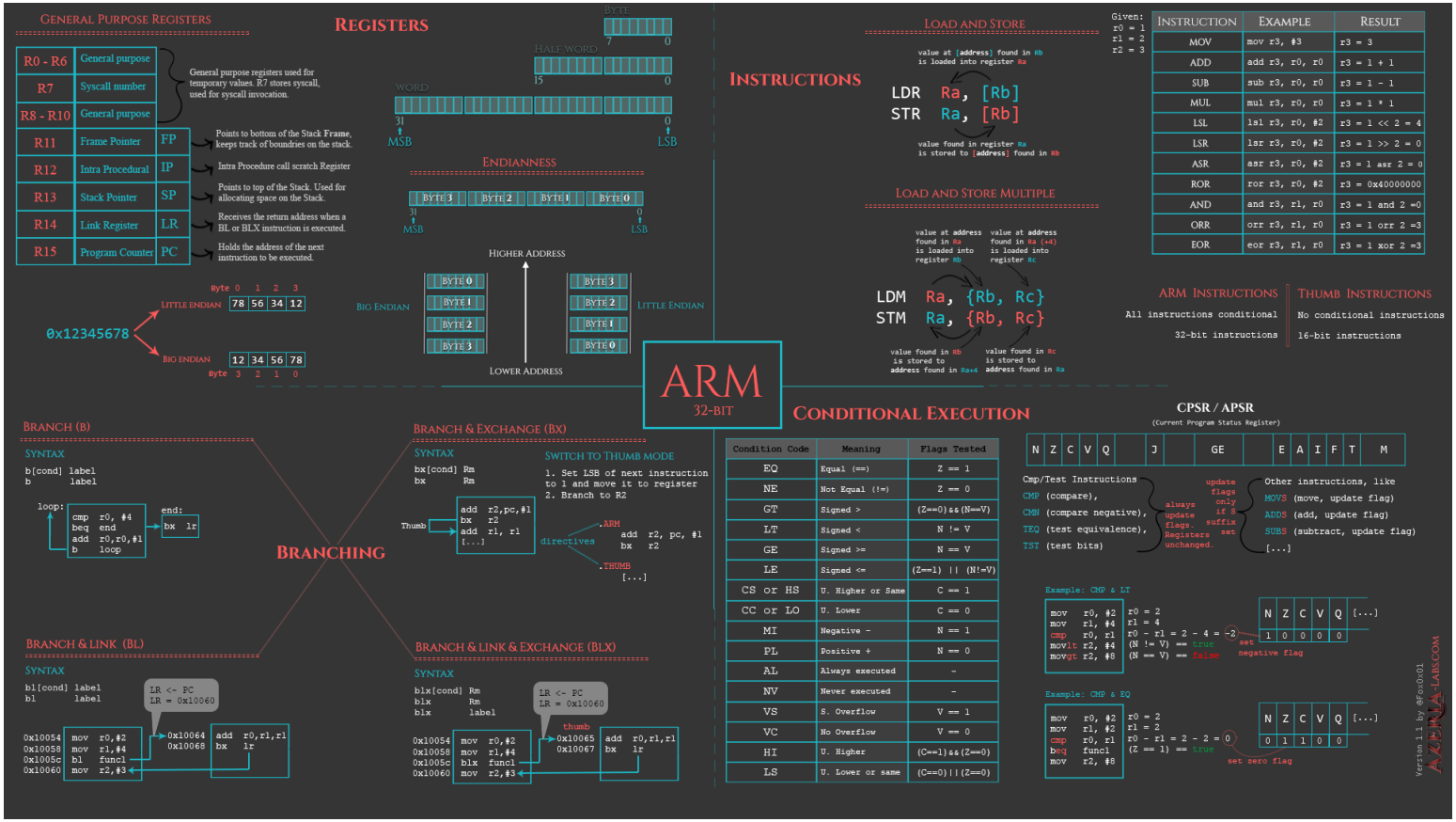
- מזהה ספרייה בינארית ב lib
- מבצע decompile עם native decompiler למשל ghidra
- מחפש טעינה של הספרייה ושימוש בפונקציה ב decompile של ה dex

Binary Static analysis

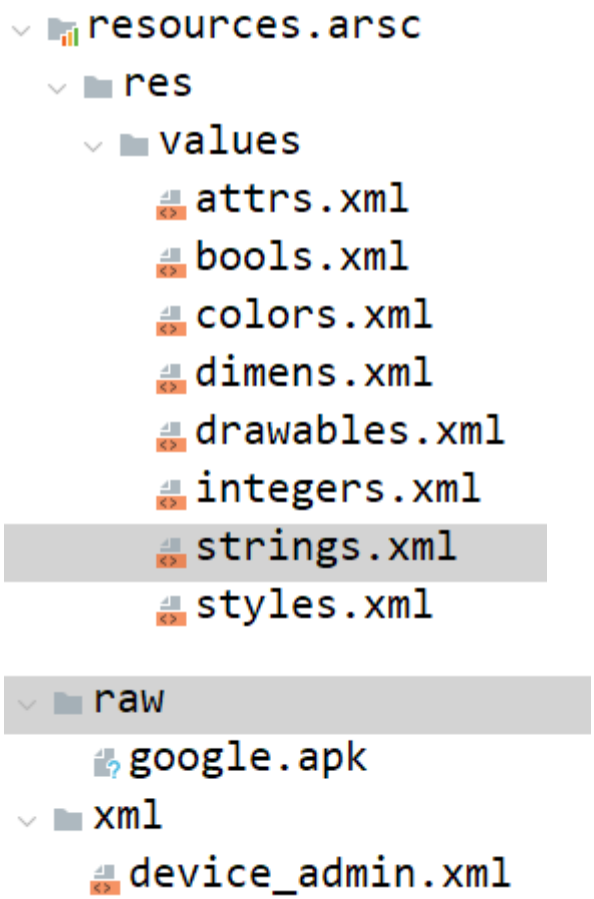
- אפשר גם רק לעשות disassembly

```
                    Java_sg_vantagepoint_helloworldjni_MainActivit... XREF[1]:     Entry Point(*)
00010eb4 80 b5          push        {r7,lr}
00010eb6 00 af          add         r7,sp,#0x0
00010eb8 a7 21          movs        r1,#0xa7
00010eba 89 00          lsls        r1,r1,#0x2
00010ebc 02 68          ldr         r2,[r0,#0x0]
00010ebe 52 58          ldr         r2,[r2,r1]
00010ec0 01 49          ldr         r1,[DAT_00010ec8]                          = 000011CEh
00010ec2 79 44          add         r1=>s_Hello_from_C++_00012094,pc           = "Hello from C++"
00010ec4 90 47          blx         r2
00010ec6 80 bd          pop         {r7,pc}
```

Binary Static analysis



Resources



- לבדוק אם יש דברים מעניינים
- Strings מתורגם מהקוד

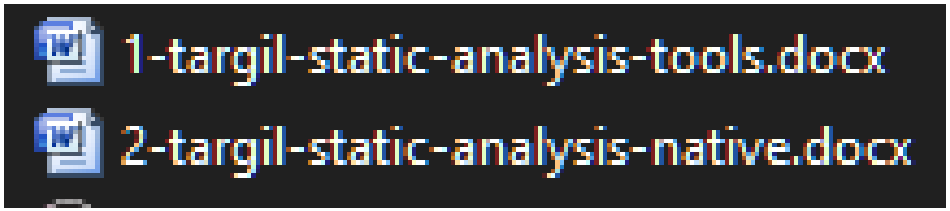
כלים Static analysis

1. JADX-GUI
2. APKLab (תוסף ל VS CODE)
3. Ghidra
4. JEB

תרגול שימוש בכלים

בצעו את התרגילים בתיקיה:

targilim-static-analysis



כלי platform

activity Manager
Package Manager

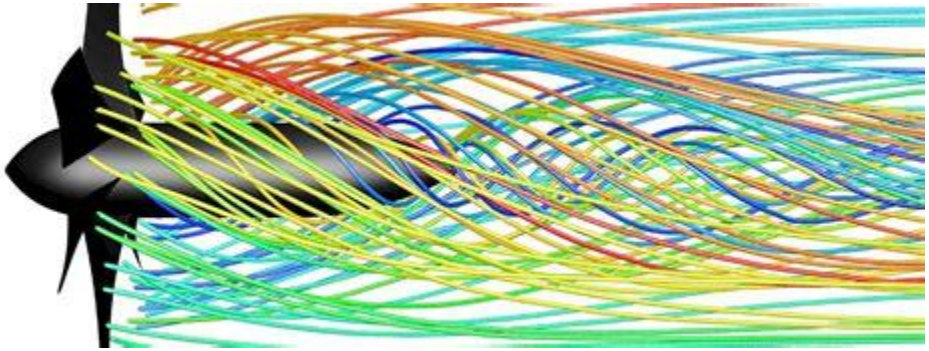
https://cheatography.com/citguy/cheat-sheets/android-activity-manager-am/
https://cheatography.com/citguy/cheat-sheets/android-package-manager-pm/

מבוא לניתוח דינמי

מבוא לניתוח דינמי

ניתוח של התנהגות האפליקציה בזמן ריצה ,כמו שימוש בזכרון ו Stack, תקשורת ברשת ושימוש ב API כדי לזהות מה האפליקציה עושה בזמן ריצה ומהם מנגנוני האבטחה.

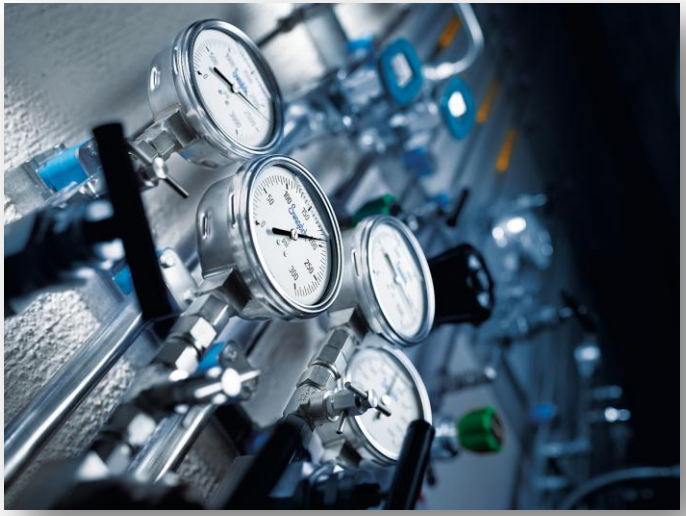
טכניקות חשובות:



- Debugging / Tracing
- Hooking
  - מעקב אחרי ריצת התוכנית
  - שינוי התוכנית בזמן ריצה

מבוא לניתוח דינמי: instrumentation / hooking

- הזרקת קוד נוסף לאפליקציה בזמן ריצה
- מוסיפים hooks בנקודות מפתח כך שאפשר:
- לצפות בערכים של משתנים
- ערכי חזור של פונקציות
- לשנות flow



מבוא לניתוח דינמי: debugging

- עצירת ההרצה של התכנית והתקדמות שורה שורה בקוד
- קריאת ערכי משתנים מהזיכרון

ניתוח דינמי של native code

- attach עם debugger
- lldb
- gdb
- לשים break בפונקציה המתאימה ולהמשיך משם
- לחקור את הזיכרון ולמצוא data
- עקרונית כמו עבודה עם כל בינארי, לאו דווקא Android

כלים לניתוח דינמי

- Frida
- lldb, gdb
- JDWP
- כלים חיצוניים Burp, Wireshark

Frida

- framework ל instrumentation
- מאפשר הזרקת קוד javascript לאפליקציות
- hooking לפונקציות כדי לקרוא זיכרון ולשנות לוגיקה
- כולל python API לאוטומציה
- מותקן כאוסף של כלים CLI pip install frida-tools

```
1|a53x:/ # /data/local/tmp/frida-server-arm64
```

```
~ $ pip install frida-tools
~ $ frida-trace -i "recv*" -i "read*" twitter
recv: Auto-generated handler: …/recv.js
# (snip)
recvfrom: Auto-generated handler: …/recvfrom.js
Started tracing 21 functions. Press Ctrl+C to stop.
    39 ms       recv()
   112 ms       recvfrom()
   128 ms       recvfrom()
   129 ms       recvfrom()
```

FRIDA

JDWP

- Java Debug Wire Protocol
- פרוטוקול לחבר debugger ל JVM
- אפשר להריץ adb jdwp ולראות את ה PID של כל האפליקציות שהם debuggable

adb forward tcp:4444 jdwp:10905
jdb -connect com.sun.jdi.Socketattach:hostname=127.0.0.1,port=4444
stop in com.example.seccon2015.rock_paper_scissors.Mainactivity.onClick
where

```
main[1] where
  [1] com.example.seccon2015.rock_paper_scissors.MainActivity.onClick (MainActivity.java:78)
  [2] android.view.View.performClick (View.java:7,892)
  [3] android.widget.TextView.performClick (TextView.java:16,219)
  [4] android.view.View.performClickInternal (View.java:7,869)
  [5] android.view.View.-$$Nest$mperformClickInternal (null)
  [6] android.view.View$PerformClick.run (View.java:30,880)
  [7] android.os.Handler.handleCallback (Handler.java:942)
  [8] android.os.Handler.dispatchMessage (Handler.java:99)
  [9] android.os.Looper.loopOnce (Looper.java:226)
  [10] android.os.Looper.loop (Looper.java:313)
  [11] android.app.ActivityThread.main (ActivityThread.java:8,757)
  [12] java.lang.reflect.Method.invoke (native method)
  [13] com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run (RuntimeInit.java:571)
  [14] com.android.internal.os.ZygoteInit.main (ZygoteInit.java:1,067)
main[1]
```

```
main[1] step
>
Step completed: "thread=main", android.view.Window.findViewById(), line=1,611 bci=0

main[1] locals
Method arguments:
Local variables:
id = 2131492946
main[1]
```

LLDB

debugger דומה ל gdb שמגיע עם ה toolchain של Android.

כדי להתחבר ל process על המכשיר, יש לבצע remote debugging.

על המכשיר, העבירו את lldb-server לתיקיה /data/local/tmp/ והריצו:
/data/local/tmp/lldb-server p --server --listen *:33333

על המחשב הריצו:
lldb
(lldb) platform select remote-android
(lldb) platform connect connect://localhost:33333
(lldb) attach 18189 (pid of process you want to debug)

המשך בתרגיל

LLDB
Debugging for Linux

תרגול

בצעו את התרגילים בתיקייה:
Targilim-dynamic-analysis

ולאחר מכן בתיקיה:
Targilim-uncrackable

anti Reversing

- Root detection
- anti Debugging
- File Integrity
- Tool Detection
- Obfuscation

https://bernhard-another.gitbooks.io/owasp-mstg-summit-edition/content/0x05j-Testing-Resiliency-against-Reverse-Engineering.html

Anti Debugging

- JDWP:
- Debuggable flag במניפסט
- האם דיבגר מחובר:

```
public static boolean detectDebugger() {
    return Debug.isDebuggerConnected();
}
```

Anti Debugging

- Native:
- בדיקה של tracer pid:

```
a53x:/ # sleep 1000 &
[2] 26412
```

```
a53x:/ # more /proc/26412/status | grep Tracer
TracerPid:      0
```

```
(lldb) attach --pid 26412
```

```
a53x:/ # more /proc/26412/status | grep Tracer
TracerPid:      26520
```

Anti Debugging

- Native:
- רק דיבגר אחד יכול להתחבר ל process
- אפשר לעשות fork ל child ולחבר את ה child לאבא עם ptrace

```
void fork_and_attach()
{
  int pid = fork();

  if (pid == 0)
    {
      int ppid = getppid();

      if (ptrace(PTRACE_ATTACH, ppid, NULL, NULL) == 0)
        {
          waitpid(ppid, NULL, 0);

          /* Continue the parent process */
          ptrace(PTRACE_CONT, NULL, NULL);
        }
    }
}
```

https://bernhard-another.gitbooks.io/owasp-mstg-summit-edition/content/0x05j-Testing-Resiliency-against-Reverse-Engineering.html

File Integrity

- בדיקת HASH של קבצים חשובים בנוסף לחתימה
- מה הבעיה עם ההגנה הזאת?

https://bernhard-another.gitbooks.io/owasp-mstg-summit-edition/content/0x05j-Testing-Resiliency-against-Reverse-Engineering.html

Tool Detection

- נסיון לזהות כלים כמו frida
- למשל לבדוק processes או פורטים דיפולטיים
  - אותה בעיה כמו ההגנה של file integrity
- אפשר ממש לבדוק אם הפרוטוקול רץ
  - פרידה משתמש ב dbus
- אפשר לבדוק בזיכרון אם הכלי טעון
- אותה בעיה כמו file integrity

https://bernhard-another.gitbooks.io/owasp-mstg-summit-edition/content/0x05j-Testing-Resiliency-against-Reverse-Engineering.html

Obfuscation

- אולי הכי נפוץ
- נסיון לסבך ולהסתיר את הקוד המקורי בלי לפגוע בפונקציונליות
- Packing: דחיסת הקוד
- בעיה בניתוח סטטי (אפשר לעבור לדינמי...)
- לרוב הכלים יש יכולת deobfuscation ☺

https://bernhard-another.gitbooks.io/owasp-mstg-summit-edition/content/0x05j-Testing-Resiliency-against-Reverse-Engineering.html

Obfuscation

טכניקות:

- כיווץ \ zipalign
- Rebuild
- Randomize manifest
- Renaming
- Encryption

https://www.softxjournal.com/action/showPdf?pii=S2352-7110%2819%2930279-1

Obfuscation

טכניקות:

- Debug removal
- Call indirection
- Goto
- Reorder
- Junk Code Insertion
  - NOP
  - Methods Overload

https://www.softxjournal.com/action/showPdf?pii=S2352-7110%2819%2930279-1

מבוא ל Android Malware

מבוא ל Android Malware

- מטרה: תגובה לאירוע
- Forensics
- פיתוח חתימות
- אותן טכניקות שלמדנו רלוונטיות

סוגי Android Malware

MALWARE bazaar
by ABUSE.ch

- Spyware
- Trojan
- Stealer
- Dropper
- Backdoor
- RAT
- לא כל כך משנה מבחינת RE – מה שחשוב זה להבין מה ואיך הוא עושה

מבוא לניתוח חולשות

מבוא לניתוח חולשות

- אחת המטרות העיקריות של RE היא לזהות נקודות חולשה ופגיעות אבטחה בקוד וב- design של אפליקציות או במערכת ההפעלה עצמה. לדוגמא:
  - CVE-2019-11932: Whatsapp buffer overflow in the app's parsing of MP4 files.
  - CVE-2017-13256: Linksys Smart Wi-Fi sensitive information network traffic
  - CVE-2018-14719: CM File Manager buffer overflow in the app's handling of ZIP files.
  - CVE-2019-3560: airDroid buffer overflow in the app's handling of UDP packets.
  - CVE-2020-8913: TikTok not properly encrypting sensitive data.

OWASP Mobile Security
https://mas.owasp.org/MASVS/

MASTG

MASTG Tests

Android

- MASVS-STORAGE
- MASVS-CRYPTO
- MASVS-AUTH
- MASVS-NETWORK
- MASVS-PLATFORM
- MASVS-CODE
- MASVS-RESILIENCE

- **MASVS-STORAGE**: Secure storage of sensitive data on a device (data-at-rest).
- **MASVS-CRYPTO**: Cryptographic functionality used to protect sensitive data.
- **MASVS-AUTH**: Authentication and authorization mechanisms used by the mobile app.
- **MASVS-NETWORK**: Secure network communication between the mobile app and remote endpoints (data-in-transit).
- **MASVS-PLATFORM**: Secure interaction with the underlying mobile platform and other installed apps.
- **MASVS-CODE**: Security best practices for data processing and keeping the app up-to-date.
- **MASVS-RESILIENCE**: Resilience to reverse engineering and tampering attempts.

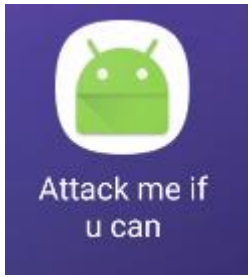
https://mas.owasp.org/MASTG/

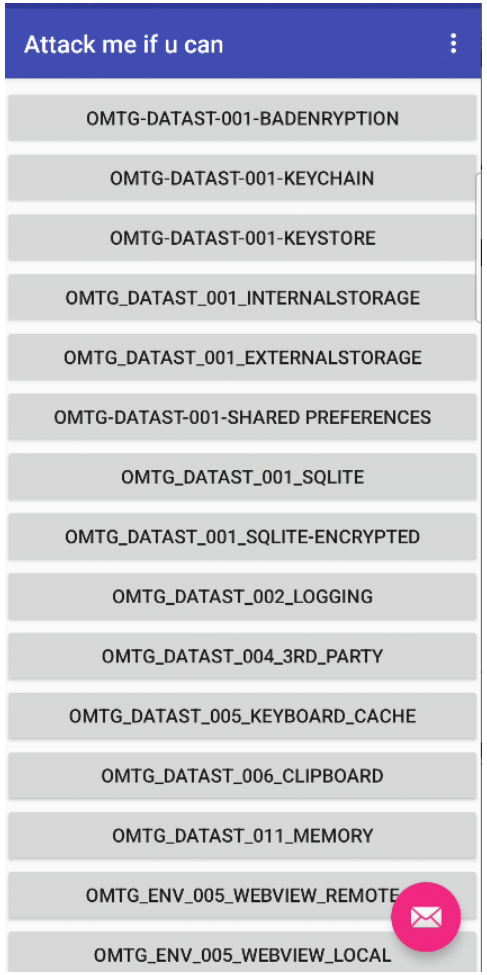
OWASP Mobile Application Security

MASTG

Mobile Application Security Testing Guide

תרגול חקר חולשות





app-arm-debug.apk

מבוא ל Patching

מבוא ל Patching

- שינוי בקוד המקור, קבצי קונפיגורציה ו resources
- APKLab
- jarsigner
- Binary patching

https://vickieli.dev/binary%20exploitation/intro-to-binary-patching/

https://kennethjenkins.net/posts/apk-patching/