تحقیق میان‌ترم درس برنامه‌سازی موبایل
زینب سادات ساقی - ۹۶۱۰۵۸۱۸ / فاطمه بحرانی - ۹۶۱۰۰۰۸۵

## موضوع تحقیق: Protecting your Android App against Reverse Engineering and Tampering

یکی از مشکلات رایج در برنامه‌سازی و به ویژه برنامه‌سازی اندروید، مهندسی معکوس است. اگر برنامه به درستی و با استفاده از روش‌های موجود محافظت نشود، می‌توان از فایل APK. آن، به راحتی کد آن برنامه را به دست آورده و به این ترتیب ارزش آن برنامه از دست خواهد رفت. در این تحقیق، به بررسی روش‌هایی برای جلوگیری هر چه بیشتر از این اتفاق می‌پردازیم.

ابتدا باید در نظر داشته باشیم که هیچ روشی را نمی‌توان به کار بست تا اطمینان داشته باشیم که اپلیکیشن ما به طور کامل از این آسیب در امان خواهد بود؛ (اگر عبارت android reverse engineering را جست‌وجو بکنیم، بیشتر نتایج نشان‌دهنده‌ی راه‌کارهای crack کردن برنامه‌های سایرین است؛ نه محافظت از برنامه‌ی خود) اما به کمک این روش‌ها، می‌توان این فرآیند مهندسی معکوس را پیچیده و زمان‌بر کرد تا کسی حاضر به پرداخت این هزینه‌ها برای دستیابی به کدهای ما نشود.

یکی از ساده‌ترین راه‌ها، این است که موارد secret را در سمت client قرار ندهیم. اطلاعات مهم‌تر را به کمک web service ارائه دهیم و به زبانی مانند PHP در آن داشته باشیم. به این ترتیب و با کمک برنامه‌های سمت سرور، دسترسی clientها به اطلاعات کمتر می‌شود و احتمال این آسیب هم کاهش می‌یابد.

یکی از راه‌های دیگر این است که برای ذخیره‌سازی داده در تلفن‌های همراه، آن‌ها را به یک فرمت ساده ذخیره نکنیم؛ مثلا برای ذخیره‌ی اعداد، آن‌ها را در یک فرمول خاصی بگذاریم که اطلاعات آن در اختیار کاربر نیست.

یک روش دیگر این است که تعدادی کد بدون استفاده‌ی رندوم به کدمان اضافه کنیم؛ شامل متدها و کلاس‌ها و ... . به این ترتیب هکر به دنبال رفرنس این کدها خواهد گشت و دسترسی به کد اصلی، سخت‌تر خواهد بود.

در ادامـــه، تـــعدادی کتابـخانـه و blog بـرای رفـع ایـن مـشکل مـعرفی می‌کـنیـم.

### Proguard:

ایـن مـورد در انـدروید اسـتودیو ساخته شده و اهـدافی را ارائـه می دهـد. روش اول انسـداد و مـبهم و تـاریک کـردنِ کد (obfuscation) است؛ اساساً کد شما را به حالت حـروف شکسته و درهـم تـبدیل می کند تا درک آن دشـوار باشد. ایـن سـناریو ممکن است به راحـتی مـغلوب شـود؛ اما اضـافه کـردن آن به برنـامه بسـیار سـاده اسـت. بـنابـراین اسـتفاده از ایـن متد تـوصیه می‌شـود.
روش دوم فشـردن و کـوچک کـردنِ کد است. اصـولاً، منابع و کدِ اسـتفاده نشـده را حذف می کند. ایـن کـار لـزوما صـحیح نیـست؛ اما به صـورت پیش فرض پیـاده سـازی شـده و می‌ارزد. تـنها راه بـررسی ایـن که آیا چیـزی ناقص است یا خیـر، مـهندسی معکوس APK خودتان است.

### Dexguard:

ایـن ابـزار رایگـان نیـست؛ امـا هـمان تیـم Proguard آن را سـاخته اسـت. ایـن ابـزار، شـامل هـمه مـواردی است که Proguard دارد و ویژگی‌هـای (feature) بیشتری را نیـز اضـافه می کند. بـرخی از اضـافات قـابل توجه عـبارتند از رمـزگذاری رشـته‌هـا (String Encryption) و رمـزگذاری منابع (Resource Encryption).

### Android NDK:

نوشـتن بـخش‌هـایی از برنـامه با کدهـای native مـانند c یا c++، قـطعا مـردم را از مـهندسی معکوس آن برنـامه بـاز می‌دارد. الـبته چـندین نکتـه‌ی جـانبی بـرای اسـتفاده از NDK قـابل ذکـر اسـت؛ مـانند مشکل کـاهش کـارآیی که هـنگام بـرقـراری تـماسهای JNI وجـود دارد. هـمچنین بـاید garbage collection را خـود انـجام دهیـم که بـرای افـراد مـبتدی، سـاده نـخواهـد بـود.

### PiracyChecker:

یک کتابـخانـه مـحبوب در github است با چـند روش اسـاسی بـرای کـاهـش مـهندسی معکوس. حـتی با وجـود اسـتفاده از ایـن کتابـخانـه نیـز اپلیکیشن‌هـــای زیـادی crack شـده‌انـد. چـندین روش بـرای جـلوگیری از ایـن crackها وجـود دارد که می‌تـوان از آن‌هـا کـمک گـرفت؛ از جـمله اجـرای بـررسی لایـسنس Google Play (LVL) که متن بـاز اسـت. بـنابـراین می‌تـوان کد آن را بـررسی کـرده و هـمچنین در آن هـمکاری کـرد.

### Google's SafetyNet Attestation API:

ایـن یک option متحیرکننده است. در اصـل، با درخـواسـت از Google's Attestation API) گـواهی گـوگـل)، آن‌هـا می‌تـوانـند بـگویند دسـتگاهی که برنـامه روی آن در حـال اجـرا اسـت امـن و مـطمئن اسـت یا خیـر؛ اگـر ریشه داشـته باشد یا از یک مـورد LuckyPatcher اسـتفاده کند.

### Deguard:

استفاده کردن از وب‌سایت‌هایی مانند «http://apk-deguard.com». با بارگذاری یک فایل APK، این سایت از برخی الگوریتم‌ها برای معکوس کردن آنچه در proguard معکوس می‌شد، استفاده می کند. بعد از آن، شما می‌توانید کلاس‌ها را باز کنید؛ حتی گاهی اوقات با نام کامل آن کلاس! با استفاده از این نسخه‌های اصلاح شده، محتوای آن کمابیش تغییر کرده‌است. فرآیندهای دستی برای دستیابی به نتایج مشابه وجود دارد؛ اما این روش سریع‌تر است و به کار کمتری هم نیاز دارد.

### Android Anti-Reversing Defenses:

خود شامل چند روش مختلف است که به تفصیل در این لینک توضیح داده شده و اینجا فقط خلاصه‌ای بیان می‌شود. شامل روش‌های SafetyNet (استفاده از سرویس‌ها و APIها.) و Recommendations when using SafetyNetApi.attest (استفاده از اعداد رندوم رمزنگاری شده در سرور که در صورت دسترسی هکر به اطلاعات، برای مرتبه‌ی بعدی قابل استفاده نباشد.) و ... است.

### Android Security (Adding Tampering Detection to Your App):

می‌توان این تکنیک را به صورت مراحل زیر خلاصه کرد:
۱. امضای گواهینامه برنامه نویس خود را پیدا کنید.
۲. امضای خود را در یک رشته رشته در برنامه خود قرار دهید.
۳. بررسی کنید که امضای در زمان اجرا با امضای توسعه دهنده تعبیه شده ما مطابقت دارد.
در واقع شبیه داشتن لایسنس عمل‌می‌کند و تنها در صورتی که برنامه را خود ما نوشته باشیم، اجرا خواهد شد.

### MobSF:

این برنامه‌ی شگفت‌انگیز در ویندوز، لینوکس و مک کار می‌کند. به طور خلاصه، این برنامه به صورت محلی اجرا می‌شود. بعد از بارگذاری APK(هنوز هیچ AAB نداریم)، کد را از نظر آسیب‌پذیری‌ها تجزیه و تحلیل می کند. این بررسی‌های اولیه را انجام می‌دهد و اطلاعات زیادی در مورد APK نشان می‌دهد. مانند کسی که گواهی، دسترسی‌های برنامه، تمام رشته‌ها (stringها) و موارد دیگر را امضا کرده است!

### منابع:


- https://medium.com/avi-parshan-studios/protecting-your-android-app-against-reverse-engineering-and-tampering-a727768b2e9e
- https://stackoverflow.com/questions/13854425/how-to-avoid-reverse-engineering-of-an-apk-file
- https://www.airpair.com/android/posts/adding-tampering-detection-to-your-android-app
- https://mobile-security.gitbook.io/mobile-security-testing-guide/android-testing-guide/0x05j-testing-resiliency-against-reverse-engineering