مهندسی معکوس نرم‌افزارهای موبایل

تألیف: سید داود ملک حسینی

(گروه امنیتی هامان)

انتشارات پندار پارس

سرشناسه : ملک‌حسینی، سیدداود، ۱۳۶۷-
عنوان و نام پدیدآور : مهندسی معکوس نرم‌افزارهای موبایل/تالیف سید داود ملک‌حسینی.
مشخصات نشر : تهران : پندار پارس، ۱۴۰۰.
مشخصات ظاهری : ۱۲۲ص.
شابک : 978-622-7785-00-5
وضعیت فهرست نویسی : فیپا
موضوع : تلفن همراه -- برنامه‌های کامپیوتری
موضوع : Cell phones -- Computer programs
رده بندی کنگره : ۶۵۷۰TK
رده بندی دیویی : ۲۶۸/۰۰۵
شماره کتابشناسی ملی : ۷۷۲۷۶۶۴
اطلاعات رکورد کتابشناسی : فیپا

انتشارات پندار

دفتر فروش: انقلاب

تلفن: ۶۶۵۷۲۳۳۵

نام کتاب : مهندسی معکوس نرم‌افزارهای موبایل

ناشر : انتشارات پندار پارس

تالیف : سید داود ملک حسینی

چاپ نخست : تیر ماه ۱۴۰۰

شمارگان : ۲۰۰ نسخه

طرح جلد : رامین شکرالهی

چاپ، صحافی : روز

قیمت : ۷۵۰۰۰ تومان **شابک : ۹۷۸-۶۲۲-۷۷۸۵-۰۰-۵**

فهرست

پیش‌گفتار

هیچ اثری خالی از اشکال نیست. این کتاب تخصصی نیز از این قاعده مستثنی نیست و امید دارم پس از مطالعه کتاب، چنانچه موردی به نظرتان رسید با من مکاتبه فرمایید؛ چرا که هدف اصلی از نگارش این کتاب، ارتقاء سطح دانش فنی علاقه‌مندان در حوزه علوم امنیتی سایبری است و لازمه آن، تبادل دانش ما با یکدیگر است. به امید روزی که شاهد شکوفایی میهن عزیزمان در عرصه‌های علم و صنعت باشیم.

در تمام دوران نگارش این کتاب سرکار خانم مهندس مریم قارونی در تألیف این اثر زحمات بی بدیعی کشیدند. بنا به دلایل کاری ایشان، فعلاً نام ایشان در فهرست مؤلفان این اثر قرار نگرفت ولی جا دارد کمال تشکر و قدردانی را از ایشان داشته باشم. از خداوند منان توفیق درجات عالی را خواستارم. همچنین جا دارد از زحمت‌های پدر و مادر عزیزم که همواره حامی و پشتیبان من بوده‌اند قدردانی کنم و از پروردگار متعال برایشان بهترین‌ها را همراه با توفیق درجات عالی آرزومندم.

.

با احترام

سید داود ملک حسینی

اخطار

پیش از شروع مطالب بهتر است بدانید که همه‌ی برنامه‌های تجاری استفاده شده در این نسخه آموزشی تنها به هدف نمایش و درک بهتر تئوری‌ها و متدهای توضیح داده شده می‌باشند و هیچ توزیعی از نرم‌افزارهای وصله[۱] (پچ) شده تحت عنوان هیچ رسانه یا هاستی انجام نشده است. نرم‌افزارهای استفاده شده اغلب وصله شده هستند و نسخه‌های کرک شده در بیشتر موارد در دسترس هستند. هدف اصلی این نسخه آموزشی تنها اشاعه و به اشتراک گذاری دانشی است که چگونگی وصله کردن نرم‌افزارها، چگونگی دور زدن قفل‌ها و در کل چگونگی ارتقاء هنر مهندسی معکوس را آموزش می‌دهد. ما هیچ نرم‌افزار کرک شده‌ای را ارائه نمی‌دهیم.

[۱] Patch

فصل ۱

معکوس‌سازی سیستم‌هایDRM در اندروید

۱ مقدمه

در سیستم‌های کامپیوتری پس از گذراندن روش‌هایی مثل استفاده از سریال نامبر در برنامه‌ها برای محافظت از برنامه و جلوگیری از کپی کردن غیر قانونی آنها در تاریخ نوامبر ۲۰۰۲ یک استاندارد معرفی شد و در سال ۲۰۰۴ این استاندارد پذیرفته شد و جای خود را برای مقابله با کپی رایت در برنامه‌های کامپیوتری باز کرد و با ورود گوشی‌های موبایل به زندگی مردم و تبدیل آنها به گوشی‌های هوشمند و لزوم استفاده از برنامه‌های کاربردی در آنها DRM به جایگاه واقعی خود رسید.

DRM (Digital Right Management) دارای سه روش برای محافظت از برنامه‌هاست.

روش Forward look: در این روش فایل را کاملاً قفل کرده و اجازه ارسال فایل به خارج را مسدود نموده و در واقع سند را غیر فعال کرده است.

در سیستم‌عامل‌هایی مانند لینوکس و ویندوز، مبحثی با عنوان پرمیژن وجود دارد و این چیزی شبیه به پرمیژن است

روش Combined Delivery: در این روش سیستم فایل و مجوز فایل برای ارسال در هم آمیخته و ارسال می شود.

روش Separate Delevery : در این روش سیستم فایل و مجوز را از هم جدا کرده و مجوز را در یک پیام جداگانه ارسال می کند. این روش بسیار امن‌تر از روش‌های دیگر است.

کاربرد DRM را در بازی‌های رایانه‌ای بیشتر هویداست. شرکت‌های بازی‌سازی برای جلوگیری از کپی رایت به سمت DRM روی آوردند؛ به طوری که برخی از آنان شروع به ساختDRM های سخت‌افزاری نموده‌اند.

مبحث DRM در بین شرکت‌های بازی‌سازی جدی شده است و حتی شرکت انویدیا نیز ساخت کارت گرافیکی که بتواند DRM را پشتیبانی کند در دستور کار خود قرار داد، اما در این مسابقه، شرکت اینتل پردازنده gpu خود که بتواند DRM را پشتیبانی کند را زودتر به بازار عرضه کرد.

در گذشته شاهد DRMهایی بودیم که باعث می‌شد کاربر تنها بازی را که خریداری کرده است تنها به تعداد محدودی مثلا ۳ بار نصب کند که باعث عصبانیت خریداران می‌شد. گاهی برخی از شرکت‌ها DRM را درون سی‌دی و دی‌وی‌دی قرار می‌دادند و موقع بازی حتماً باید لوح فشرده بر روی دستگاه بوده و چون سرعت خواندن دستگاه سی‌دی یا دی‌وی‌دی کم بود باعث کند بودن بازی می‌شد. اما جدیدا شرکت‌های بازی‌سازی رو به سوی DRMهای اینترنتی رفته‌اند؛ یعنی در موقع نصب بازی و هنگام بازی باید سیستم به سرور آن شرکت‌ها متصل شود و ثابت کند که بازی کپی شده را نصب نکرده است. اخیراً با ورود DRMهای ۶۴ بیتی و ۱۲۸ بیتی امکان شکستن قفل آنها نزدیک به صفر شده است اما یک اصل در بین هکرها وجود دارد؛ هر قفلی قابل گذر می‌باشد ولی نیاز به صرف وقت بیشتری دارد. در این کتاب به شما می‌آموزیم که چگونه از DRMهای موبایل عبور کنید و به راحتی نرم‌افزار دلخواه خود را استفاده کنید اما با این حال امروزه در بین مردم استفاده از کامپیوترهای خانگی رو به منسوخ شدن است؛ پس سعی داریم شما را باDRM موبایل که کاربردهای بیشتری دارد آشنا کنیم. امید است با درک نکاتDRM ها بتوانید بهترین نوع این قفل را برای مشتریان خود طراحی کنید.

لازم به ذکر است که DRMها در مبحث ویندوز و لینوکس که بر روی نرم‌افزارها اعمال می‌شوند اغلب به صورت قفل‌های اینترنتی است که نرم‌افزار پس از ورود سریال نامبر باید به سرور متصل شود و تأییدیه اعتبار بگیرد. همان‌طور که می‌دانید عبور از DRM در این سیستم‌عامل‌ها به سادگی بوده و تنها کافی است اجازه دهید نرم‌افزار به اینترنت متصل شود و با وارد کردن سریال غلط پیام خطایی مبنی بر غلط بودن آن سریال دریافت کنید و سپس با جستجوی پیام و یا توابع و یا دیگر روش‌ها در نرم‌افزارهای دیباگر یا دیزا اسمبلرها آن قسمت از سورس که در حال چک کردن شماره سریال بود را پیدا و آن را کرک کنید.

در سیستم‌های اندروید و ios بیشتر به صورت تئوری روال کار به همین صورت است اما تمام نکات کلیدی که هر سیستم‌عامل برای عبور از سیستم‌های DRM لازم دارد را به شما آموزش می‌دهیم. چون همانطور که می‌دانید سیستم DRM شرکت اپل یکی از قدرتمندترین سیستم‌های DRM جهان است ولی ناامید نشوید چون این سیستم راه‌های نفوذ زیادی دارد پس با هم به بررسی این روش‌ها می‌پردازیم.

۱.۱ Slide Lock چیست و چرا از اهمیت ویژه‌ای برخوردار است؟

Slide Lock در واقع یک سیستم DRM برای برنامه‌های سیستم عامل اندروید می‌باشد که از به اشتراک‌گذاری APKهای خریداری شده در بین کاربران، جلوگیری می‌کند. این قفل در کلاس‌های ویژه‌ای قرار گرفته شده است که برنامه نویسان بایستی آن را در برنامه خودشان گنجانده و پیاده سازی کنند. بدین طریق در قسمت سرور عمل چک کردن اطلاعات ویژه دستگاه جهت اطمینان از مجاز بودن کاربر در دسترسی به نرم‌افزار، انجام می‌گردد. هر نرم‌افزاری دارای یک کلید است که این کلید حداکثر ۳۲ کاراکتری کاملاً اختصاصی و منحصر به فرد می‌باشد. بسیار مهم است که این کلید برای کاربران ناشناخته بماند، دلیل و اهمیت موضوع بعداً مشخص می‌شود.

در حالی که فصل DRM به نظر خوب و مناسب می‌رسد اما در مواردی چالش برانگیز می‌باشد. به عنوان نمونه، شما نرم‌افزار بازی Anti Bady را خریداری کرده و با موفقیت روی موبایلتان نصب می‌کنید، و سپس تلفن شما یا می‌شکند یا از کار می‌افتد یا به‌روزرسانی می‌شود، اینجاست که بایستی بازی را دوباره خریداری کنید.

با این دید Slide lock 1.1 را بررسی می‌کنیم و جزئیاتش را بیان کرده و به صورت نسخه آموزشی ارائه می‌دهیم. انتظار می‌رود که پس از پایان این آموزش قادر به حذف قفل Slide lock 1.1 از هر نرم‌افزاری باشند.

ضروری است که به این نکته توجه کنید که این قفل یک راهکار خلاقانه و عالی نیست، و هر دو سناریویی دقیقاً مشابه هم نیستند، بنابراین در این آموزش فقط این موضوع را درک کنید که چرا انجام داده‌ایم و چه چیزی را انجام داده‌ایم، و نباید سعی کنید که گام‌ها و واژه‌ها را تقلید کنید.

نرم‌افزار هدف: (Game) AntiBody 2

ابزارهای مورد نیاز:

- **APK Manager**: برای دیکامپایل[۱] کردن/ بازگرداندن کامپایل[۲] کردن و ورود به فایل Apk
- **ADB**: برای اضافه کردن و نصب Apk ها
- **Notepad**: برای خواندن فایل‌های Xml/ Smali
- **Brain**: بررسی Slide Lock

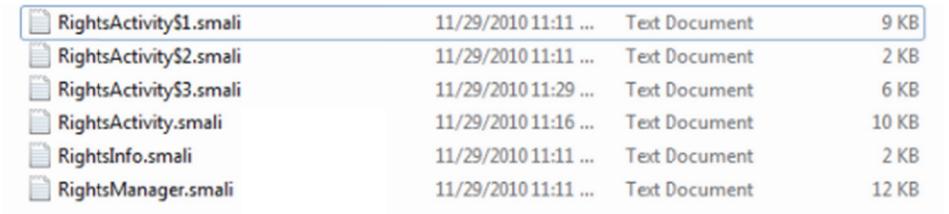
---

[۱] decompile

[۲] recompile

پیدا کردن کلاس‌های Slide Lock

به محض دیکامپایل کردن نرم‌افزار با استفاده از Apk Manager می‌توانیم کلاس‌های خالص Slide Lock را در فولدر smali/org/slideme/andricl/DRM/ پیدا کنیم که همانند شکل زیر می‌باشد:



در حالی که به نظر می‌رسد که ۶ کلاس باشد، اما در واقع تنها ۳ کلاس زیر را داریم:

- Rights Activity
- Rights Info
- Rights Manager

۱.۲ آنالیز کردن کلاس‌ها

Rights Info: در سیستم DRM این کوچکترین کلاس در بین ۳ کلاس نام برده شده است، پس بهتر است ابتدا به آن بپردازیم. پس از باز کردن فایل در برنامه نوت پد، اولین چیزی که خواهیم دید این است که rights Info از یک، ساختار (Constructor) صریح استفاده می‌کند:

```
# direct methods
.method public constructor <init>(Ljava/lang/String;Ljava/lang/Class;)V
    .locals 2
    .parameter "key"
    .parameter "mainClass"

    .prologue
    .line 9
    invoke-direct {p0}, Ljava/lang/Object;-><init>()V
```

همانطور که مشاهده می‌کنید سازنده رشته به نام "key" و یک کلاس به نام "main" را می‌گیرد. این اطلاعات در آینده برای ما مفید واقع خواهند شد. هم اکنون تنها کافی است بدانیم که هر شئ Rights Info به key و main Class از نرم‌افزار نیاز دارد تا در مرحله ایجاد و ساخته شدن پشت سر گذاشته شود.

Rights Manager: این دومین کلاس بزرگ ایست و طبق نامش به نظر می‌رسد که وظیفه مدیریت دسترسی به نرم‌افزار را دارد. بدین معنی که تعیین می‌کند آیا اجازه دسترسی به نرم‌افزار را داریم یا

نه. به محضی که در نرم‌افزار نوت‌پد باز شود مشاهده می‌شود که این کلاس از سازنده پیش فرض (بدون پارامترها) استفاده می‌کند و نیازی به نصب ویژه و به‌خصوصی نیست. همچنین این کلاس دو تابع را تولید می‌کند:

```
.method private static checkForwardLock(Ljava/net/URL;Z)Z
    .locals 6
    .parameter "url"
    .parameter "hasFile"
.method isForwardLocked(Ljava/lang/String;Ljava/lang/String;ILandroid/telephony/TelephonyManager;Z)Z
    .locals 5
    .parameter "key"
    .parameter "version"
    .parameter "versionCode"
    .parameter "mng"
    .parameter "hasFile"
```

در کد Dlvik Annotation؛ مقدار Z نوع بولین را نشان می‌دهد، بنابراین همانطور که در بالا مشاهده می‌کنید، دومین روتین یعنی خط is forward Locked یک مقدار بولین را برمی‌گرداند، و از خارج از کلاس نیز قابل دسترس است. مشاهده می‌کنید که خط اول یعنی Check Forward Lock یک قسمت اختصاصی و شخصی است به این معنی که احتمالاً یک روال کمکی است که از داخل Is Forward Locked صدا زده می‌شود.

باید به خاطر داشته باشیم که مسئولیت این کلاس اعتبارسنجی است. معادل جاوای این روال‌ها را در این نسخه آموزشی و همچنین در آدرس‌های زیر ارائه می‌دهیم:

- checkForwardLock -> http://pastebin.com/fyaxzdRq
- isForwardLocked -> http://pastebin.com/YmLaBiex

پس از تحلیل و آنالیز کردن منابع تبدیل شده، این موضوع مسلم است که در روال Is Forward Locked آنها اطلاعات زیادی از کاربر ایستگاه (که بسیاری از آن برای اهدافشان مورد نیاز نیست) بدست می‌آورند و به سمت سرورشان ارسال می‌کنند. این اطلاعات شامل موارد زیر است:

- Device ID -> Only one really needed
- Network Operator
- Network Operator Name
- Network Country ISO
- Device Software Version
- Phone Type
- Network Type
- Sim Country ISO
- Sim Operator
- Sim Serial Number -> Maybe this one too (For GSM users)
- Subscriber ID
- Network Roaming

در حالی که اطلاعاتی که در اینجا در کلاس Rights Manager ارائه شد مستقیماً ارتباطی به removal از Slide Lock DRM ندارد، و در واقع یک اعلام جهت مشاهده مقدار داده‌هایی است که بدون هیچ دلیل مشخص و آشکاری بیرون کشیده می‌شوند.

Rights Activity: در بین سه کلاس نام برده، این بزرگترین کلاس است و می‌توان گفت پیچیده ترین آن. به همین دلیل کلاس را کامل به جاوا تبدیل نخواهیم کرد و تنها قسمت‌های مهم را تبدیل می‌کنیم. اولین چیزی که هنگام باز کردن فایل در نرم‌افزار نوت‌پد توجه ما را جلب می‌کند این است که این یک چکیده‌ای[۱] از کلاس است، بدین معنی که شخصی که از این کلاس استفاده می‌کند (تولید کننده نرم‌افزار) باید کلاسی را طراحی و ایجاد کند که از این کلاس ارث ببرد و سپس توابع ناموجود را پیاده سازی کند.

بایستی به ثابت‌های مشخص شده در فایل توجه کنیم زیرا ممکن است در آینده مفید واقع شوند:

```
.field private static final LOCKED:I = 0x0
.field private static final ERROR:I = 0x1
.field private static final IO_ERROR:I = 0x2
.field private static final NOT_LOCKED:I = 0x3
```

توابع موجود در این کلاس در زیر آورده شده است:

```
.method private startMainApplication()V
.method public exitOnError()Z
.method public abstract getRightsInfo()Lorg/slideme/android/drm/RightsInfo;
.method public final onCreate(Landroid/os/Bundle;)V
.method protected onCreateDialog(I)Landroid/app/Dialog;
```

توجه داشته باشید که در کنار روال get Rights info ( ) کلمه abstract دیده می‌شود، و موردی است که در اینجا پیاده‌سازی نشده است، بلکه در کلاس تولید کننده که این را به ارث می‌برد می‌باشد.

روال Start Main Application تنها آنچه را که می‌گوید انجام می‌دهد، در واقع get Rights info را جهت گرفتن شئ Rights info که با این برنامه مرتبط است را صدا می‌زند، و می‌دانید که Applications KEY و Main Class را نگه می‌دارد. برای همه مقاصد و اهداف می‌توانیم به این Main Class به عنوان OEP توجه کنیم.

---

[۱] abstract

روال ( ) exit On Error تنها به DRM می‌گوید که آیا نرم‌افزار بایستی خارج شود یا خیر. اگر یک پیام خطا دریافت کردیم (به عنوان مثال: ارتباط اینترنت برقرار نیست) و این هنگام ارزیابی نیازمندی‌های ما کاملاً بی اهمیت است. اینکه به صورت پیش فرض این روال مقدار ϕ یا DALSE را برگرداند ممکن است هیچ ارزشی نداشته باشد، اما تولید کننده و برنامه ممکن است که این تابع را لغو کرده تا مقدار TRUE را برگرداند. این کاملاً بستگی به پیاده سازی و موارد مدنظر تولید کننده دارد.

از آنجایی که ۲ روال آخر برای کار ما بی اهمیت هستند فعلاً بحثی روی آنها نداریم.

3 $ Activity Rights: در این کلاس موارد بیشتری را بررسی می‌کنیم. یکی از چیزهایی که در این کلاس مشاهده می‌کنید این است که شامل پیام‌های چالش برانگیزی است:

```
•   const-string v2, "This application is locked to another device. Shutting down."
•   const-string v2, "An error detecting lock on device. Is network active?"
•   const-string v2, "There was a problem with the network."
```

اکنون نیاز است که چیزهایی را در مورد کد smali بیاموزیم. این در واقع مشابه کد ASM برای X&b reversing است. با کمی تمرین متوجه می‌شوید که به اندازه کد جاوا قابل خواندن است اما یکسری از عبارات منطقی همانند حلقه‌ها و شرط‌ها و سوئیچ‌ها می‌توانند گیج کننده باشند. آنهایی که قبلاً تجربه برنامه‌نویسی در جاوا را داشته‌اند می‌دانند که سوئیچ‌ها تنها روی مقادیر عددی کار می‌کنند که با زبان‌های NET. متفاوت است، و در آن زبان‌ها روی یک داده استرینگ (رشته) می‌توان سوئیچ را انجام داد.

روالی که شامل این رشته‌های چالش برانگیز است از یک عبارت سوئیچ استفاده می‌کند تا اینکه مشخص شود به کدام یک مراجعه کنیم. می‌توانید در زیر راه اندازی عبارت سوئیچ را مشاهده کنید:

```
iget v1, p1, Landroid/os/Message;->what:I
packed-switch v1, :pswitch_data_0
```

کد بالا نشان می‌دهد که مقدار داده integer پیام خوانده می‌شود و مشخص می‌شود که پیام چیست و آن را در متغیر V1 ذخیره می‌کند. شئ پیام که آنها به آن اشاره دارند P1 می‌باشد (اولین پارامتر) که به آن روال فرستاده شده است. پس از اینکه مقدار موجود ذخیره شد، یک سوئیچ روی مقدار در V1 اجرا می‌شود، ϕ pawitch_ sata_ را برای تعریف محدوده‌ی مقادیری که در سوئیچ استفاده می‌شود، بکار ببرید. با زدن کلید Ctrl+F روی آن داده استرینگ مجموعه زیر را دریافت می‌کنید:

```
:pswitch_data_0
.packed-switch 0x0
    :pswitch_0
    :pswitch_1
    :pswitch_2
    :pswitch_3
.end packed-switch
```

این نشان می‌دهد که مقادیر تعیین شده برای سوئیچ از ϕ شروع می‌شوند و در چهار کیس برای چک شدن ارائه می‌شوند، در هر کیس عدد بعدی بالاتر است که این نتیجه در سوئیچ روی مقادیر $\phi - 3$ است. مقادیری را که قبلاً پیدا کرده‌ایم فراخوانی کنید:

```
.field private static final LOCKED:I = 0x0
.field private static final ERROR:I = 0x1
.field private static final IO_ERROR:I = 0x2
.field private static final NOT_LOCKED:I = 0x3
```

خطوط را بین packed- switch. و end packed- switch. برچسبهایی را نشان می‌دهند که در واقع شروع کد case را تعریف و مشخص می‌کنند. بنابراین برای پیدا کردن کد برای کیسی که ϕ==V1 است کافی است کلیدهای Ctrl + F را روی "pawitch_ϕ": فشار دهید. چند خط پایین‌تر از این پیامی مبنی بر قفل بودن نرم‌افزار می‌بینید و تردیدی در این نیست که اینجا نقطه‌ای است که مقادیر بالا استفاده می‌شوند. و به نظر می‌رسد که این نقطه جهت وصله کردن مناسب است.

نکته: می‌توانید منابع جاوا معادل کلاس Rights Activity را در این کتاب آموزشی و همچنین در سایت مقابل پیدا کنید. http://pastebin.com79P8AE9a9

۱.۳ پیدا کردن کلاس مربوط به تولید کنندگان

تا اینجا با آنچه که تاکنون آموخته‌ایم می‌توانیم عمل وصله کردن نرم‌افزار را برای دور زدن و گذشتن از قفل انجام دهیم، اما مهم این است که قادر باشیم تا حد امکان عمل معکوس کردن[۱] را انجام دهیم. پس نکته بعدی که لازم است بدانیم این است که کدام کلاس توسط تولید کننده جهت گسترش

[۱] *revers*

کلاس rights Activity ایجاد گردیده است. راه‌هایی برای انجام این پروسه وجود دارد اما تنها یکی از این راهکارها مطمئن و قابل بررسی است.

می‌توانیم فایل Android Mani Feat.xml را چک کنیم تا متوجه شویم که آیا کلاس مورد نظر به کلاس startup تنظیم شده است یا نه. تنها راه قابل اجرا این است که به فولدر smali در command prompt مراجعه کرده و خط دستوری زیر را جهت تغییر نام همه فایل‌های smali. به txt. اجرا کنیم.

FOR /R %x IN (*.smali) DO ren “%x” *.txt

به محضی که تمام فایل‌ها به txt. تغییر نام پیدا کردند، لازم است که همه محتویات فایل را برای این رشته جستجو کنیم:

```
.super Lorq/slideme/android/drm/
```

نتیجه این جستجو نشان خواهد داد که فایل Slide Lock.smali که در مسیر /smali/creafire/com/antibody2 واقع شده است، کلاس Rights Activity را گسترش می‌دهد. خاطر نشان می‌کنیم که هر کلاسی که کلاس Rights Activity را گسترش می‌دهد بایستی حتماً روال get Rights Info ( ) را پیاده‌سازی کند. اکنون نگاهی به پیاده سازی برنامه‌نویس می‌اندازیم:

```
.method public getRightsInfo()Lorg/slideme/android/drm/RightsInfo;
    .locals 3

    .prologue
    .line 9
    new-instance v0, Lorg/slideme/android/drm/RightsInfo;

    const-string v1, "dnsfgkgi5h44k5uFTJshgui45hgdsfnb"

    const-class v2, Lcreafire/com/antibody2/Main;

    invoke-direct {v0, v1, v2}, Lorg/slideme/android/drm/RightsInfo;-><init>

    return-object v0
.end method
```

مشاهده می‌کنید که در اینجا آنها یک ثابت جدید از شئ Rights Info را می‌سازند، همان تابعی که مقادیر key و کلاس را به عنوان آرگومان‌های سازنده، می‌گیرد. این شئ با این مقادیر تنظیم می‌شود: متغیر $V\phi$ و سپس V1 که یک رسته ۳۲ کاراکتری است (کلید) به آن نسبت داده شده است و نهایتاً V2 که به کلاس Main اختصاص داده شده است و این کلاس در فولدر /samli/creafire/com/antibody2 واقع شده است. سپس آنها شئ Rights Info را صدا زده و این دو متغیر را به آن منتقل می‌کنند. به نظر می‌رسد که کلید و OEP نرم‌افزار را پیدا کرده‌ایم. هم اکنون وقت آن است که مواررد یافت شده را کنار هم قرار دهیم:

۱.۴ وصله کردن/ حذف کردن DRM

با در نظر گرفتن همه اطلاعات جمع شده، به آسانی می‌توانیم به ۲ راهکار جهت وصله یا حذف کردن DRM از نرم‌افزار، دست یابیم. برای تکمیل آموزش، هر دو گزینه را توضیح داده و بررسی می‌کنیم. یکی از راهکارها یک گزینه مطلوب و مناسب است.

۱.۵ وصله کردن عبارت Switch

مبحثی که در آن Switch Statement مقادیر $\phi - 3$ را در بر می‌گرفت را به خاطر می‌آورید؟ ما می‌توانستیم یک خط از کد Smali را درست پیش از عبارت Switch اضافه کنیم و این کار را جهت ایجاد تغییر در مقدار مورد استفاده انجام می‌دهیم. می‌توانیم به Value همیشه مقدار 3 را نسبت دهیم (خوب است) و این کار را پیش از وارد کردن switch statement انجام می‌دهیم.

خط کدی که می‌توانیم از آن استفاده کنیم این است:

```
const/4 v1, 0x3
```

این خط به نرم‌افزار می‌گوید که ثابت ۴ بیتی با مقدار $\phi \times 3$ را به متغیر V1 نسبت دهد. پس از وصله کد به شکل زیر نمایان می‌شود:

```
iget v1, p1, Landroid/os/Message;->what:I
const/4 v1, 0x3
packed-switch v1, :pswitch_data_0
```

کد بالا این اطمینان را به ما می‌دهد که بدون در نظر گرفتن کد ارسال شده، این مقدار همیشه پیش از عبارت switch خواهد بود. پس از اضافه نکردن این خط از کد، به سادگی به کامپایل reinstall /resign/ recompile دوباره نرم‌افزار نیاز داریم.

نکته: چون فایل را تغییر داده در مقایسه با نرم‌افزار اصلی با یک کلید متفاوت آن را امضا و علامت گذاری کرده‌ایم، اجباراً بایستی ابتدا نرم‌افزار را uninstall کرده و سپس نرم‌افزار تغییر یافته خودمان را نصب کنیم.

۱.۶ تغییر دادن کلاس Start up

با فراخوانی مجموعه اطلاعات بدست آورده، متوجه شدیم که کلاس Start up، و نیز OEP نرم‌افزار چیست. در اینجا بایستی تلاش کنیم که کلاس Start up نرم افزار را به Main تغییر داده و همه Slide Lock را کنار بگذاریم.

هر نرم‌افزار اندروید شامل یک فایل Android Manifest.xml می‌باشد که به سیستم عامل درباره نرم‌افزار نکات ویژه‌ای را منتقل کرده و بیان می‌کند. مواردی همانند اینکه چه محورهایی باید جهت درست اجرا شدن داشته باشد، و آیا صفحه نمایش باید عمودی یا افقی نمایان شود. همچنین در این فایل موارد دیگر نیز وجود دارد همچون لیستی از همه Activities یا (اسکرین) و مشخصه‌های ویژه آنها که نرم‌افزار دارا می‌باشد.

بیایید نگاهی به فایل Android manifest این نرم‌افزار بیاندازیم:

```
<activity android:label="@string/app_name" android:name=".SlideLock">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity android:label="@string/app_name" android:name=".Main" android:screenOrientation="lands
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

مشاهده می‌کنید که در اینجا دو اکتیویتی لیست شده است، در ابتدا Slide Lock به عنوان اولین اکتیویتی معرفی شده است و سپس Main که دومین آن است. توجه کنید که هر دو دارای <category> از LAUNCHER و نیز دارای <action> از MAIN می‌باشند. این یک خطای طراحی از سمت تولید کننده نرم‌افزار است. تنها اکتیویتی که به عنوان اصلی و لانچر، در اینجا می‌تواند لیست شده و معرفی شود بایستی Slide Lock باشد. مطمئناً هنگام نصب APK اصلی متوجه شده‌اید که ۲ ورودی برای ترم‌افزار Anti Body2 وجود داشت. این دو ورودی به این دلیل است که:

اولین ورودی، در واقع Slide Lock'd Activity است که از DRM جهت تأیید اینکه مجاز هستید آن را جدا کنید یا نه، استفاده می‌کند. دومین ورودی کاملاً یک اکتیویتی محافظت نشده است.

کل کاری که باید در اینجا انجام دهیم، حذف کردن ورودی <activity> است همان که Slide Lock به عنوان نام اکتیویتی نشان می‌دهد. این کار تنها روی یک ورودی در App Drawer تاثیر می‌گذارد، و آن ورودی اتوماتیک‌وار کلاس Main را (که هیچ DRMای در آن نیست) اجرا می‌کند.

در صورتی که با نرم‌افزاری برخورد کنید که این قسمت به درستی در آن انجام شده باشد، تنها کافی است که نام اکتیویتی‌ها را با آنچه که قبلاً از Main Class واقعی در بخش ۲.۲.۵ پیدا کرده‌اید، جابجا کرده و سپس دوباره عمل کامپایل را انجام دهید. APK نتیجه، همه‌ی کد DRM را کنار گذاشته و شما را در برابر یک برنامه نامناسب قرار می‌دهد.

۱.۷ کلید نرم‌افزار

با این حال که خود کلید برای پاک کردن DRM نرم‌افزار بی فایده است، می‌تواند رد بقیه سناریوها مفید و قابل استفاده باشد. فرض را بر این بگذارید که یک نرم‌افزار ارزان قیمت را خریداری کرده‌اید، همچنین در این میان کلید نرم‌افزار را نیز پیدا کرده‌اید. به محضی که چنین فرصتی را بدست آورید فقط کافی است که از یک نرم‌افزار دیگر (که ممکن است گرانتر باشد) استفاده کرده و کلید نرم‌افزار گرانتر را با کلید نرم‌افزاری که به صورت قانونی خریداری کرده‌اید، جابجا کنید. پس از این جایگزینی هر دو نرم‌افزار دارای یک کلید هستند (کلید نرم‌افزار ارزان قیمت)، از دید تئوری هر دو آنها بایستی مرحله ارزشیابی را با موفقیت پشت سر بگذارند. ممکن است به ایجاد تغییرات دیگر همانند تغییر ورژن و کد ورژن [۱] نیز نیاز داشته باشید که متأسفانه در اینجا نمی‌توانیم این مبحث را توسعه دهیم.

۱.۸ نتیجه‌گیری

با تحلیلی که روی Slide Lock 1.1 داشتیم، توانستیم اطلاعاتی در مورد DRM پیدا کنیم. آموختیم که چگونه می‌توانیم مرحله سرور چک [۲] را پشت سر بگذاریم و این کار توسط وارد کردن کد Smali که روی عبارت سوئیچ تأثیر می‌گذارد انجام می‌گردد، همچنین آموختیم که چگونه به طور کامل از مکانیزم‌های DRM عبور کنیم و این کار با ایجاد تغییرات در فایل Android Manifest.xml انجام می‌شود و نهایتاً آموختیم که از اطلاعات کسی استفاده می‌شود تا مشخص شود که آیا مجوز برای اجرای برنامه دارید یا نه. به نظر می‌رسد که پاک کردن DRM کاملاً امن‌ترین راهکار است. با پاک کردن DRM، Slide Lock را از فرستادن اطلاعاتمان به سرورها محافظت می‌کنیم. بر اساس اطلاعات موجود در سایت، آنها تنها از شماره قطعه (*Device ID*) شما جهت گرفتن تأیید استفاده می‌کنند اما عملاً دیدیم که آنها از داده‌ها و اطلاعات بیشتری استفاده می‌کنند.

---

[۱] *Version Code*

[۲] *Server Check*