بسمه تعالی

بررسی آنتی‌ویروس‌های اندرویدی منتشر شده در مارکت های ایرانی

۱ چکیده

بسیاری از برنامه‌هایی که تحت عناوین مختلف در مارکت‌های اندروید منتشر می‌شوند از روی برنامه‌های منبع باز ساخته می‌شوند. این برنامه‌ها صرفاً با تغییر نام و آیکون به عنوان برنامه‌های مختلف منتشر می‌شوند. در بسیاری از موارد هدف از این کار استفاده از سرویس‌های تبلیغاتی داخل این برنامه‌ها و درآمدزایی برای منتشرکننده برنامه است. هرچند این برنامه‌ها به صورت مداوم توسط افراد مختلف منتشر می‌شوند در اغلب موارد هیچ کارایی نداشته و حتی ممکن است بدافزار باشند.

در این مستند برنامه‌های منتشر شده تحت عنوان آنتی‌ویروس مورد بررسی قرار گرفته‌اند. بررسی و مقایسه این برنامه‌ها نشان می‌دهد که عمدتاً کارایی لازم را نداشته و یا عملکرد جعلی دارند و تنها با هدف جذب کاربر و کسب درامد از تبلیغات توسعه یافته‌اند.

نحوه بررسی تشابه این آنتی‌ویروس‌ها در ابتدا بر اساس تشابه لیست apiهایی که فراخوانی می‌کردند و سپس برای اطمینان کامل، بر اساس بررسی کد و نحوه عملکرد دقیق برنامه‌ها بوده است.

۲ بررسی آنتی‌ویروس‌های مارکت های ایرانی

حدود ۱۲۰ آنتی‌ویروس منتشر شده در مارکت های ایرانی جمع‌آوری و مورد بررسی اولیه قرار گرفتند. در این میان ۶۰ آنتی‌ویروس با توجه به شباهت بالایی که به یکدیگر داشتند به هفت دسته تقسیم شدند. برنامه‌های هریک از این دسته‌ها کاملا مشابه یکدیگر هستند. چند نکته جالب در بررسی شباهت آنتی‌ویروس‌های بررسی شده (بر اساس مقایسه apiهای فراخوانی شده در برنامه‌ها) حاصل شد. مورد اول اینکه آنتی‌ویروس‌های معروف مانند Kaspersky, Avira, Avast ,... شباهت بسیار کمی به آنتی‌ویروس‌های دیگر داشتند. در حالی که آنتی‌ویروس‌های ایرانی و کمتر شناخته شده عمدتاً شباهت نسبتا بالایی به یکدیگر داشتند. در مورد رفتار آنتی‌ویروس‌های دسته‌بندی شده به صورت مختصر می‌توان گفت که برخی از یک پایگاه داده آفلاین استفاده می‌کنند که آن را به‌روزرسانی نمی‌کنند، برخی نیز فقط موارد بسیار اولیه همچون مجوزهای برنامه‌ها و یا فرمت فایل‌های ذخیره شده روی دستگاه را بررسی می‌کنند که این موارد از یک برنامه با عنوان آنتی‌ویروس قابل قبول نیست. برخی از آن‌ها هم اساساً کاری انجام نمی‌دهند و صرفاً با ظاهرسازی کاربر را فریب می‌دهند که کار آنتی‌ویروس یا خنک‌کننده پردازنده را انجام می‌دهند.

۳ نتایج بررسی

از بین آنتی‌ویروس‌هایی که مورد بررسی قرارگرفتند بسیاری از آن‌ها شامل کدهای مشابه بوده و عملکرد یکسانی داشتند، با استفاده از نتایج حاصل از بررسی شباهت برنامه‌ها و تحلیل آن‌ها، برنامه‌هایی که شامل

کدهای مشابه بودند در دسته‌های یکسان قرارگرفته و هرکدام از دسته‌ها مورد بررسی قرارگرفتند. لیست نام بسته (package name) هریک از دسته‌ها در ادامه آمده است. تحلیل برنامه‌های هرکدام از دسته‌ها نیز در قسمت‌های بعدی گزارش به صورت کامل بیان شده است.

دسته اول:

- com.farzanehgroup8.antiviruss
- com.kianoreeves2.antivirusemenplus
- com.mrfarshid9.herfeiiantivirus
- com.najmejan4.proantivirus
- com.neginmobi2.antivirussabz
- com.sadrooid15.AntiVirusAspirin
- com.mrfarshid13.antivirоosherfei
- com.zinatidev4.royaantivirus
- com.mahnazinco.antivirus
- com.tannaztalaee2.antivirus

دسته دوم:

- ir.daryasoft.antivirus
- ir.maher.antivirus
- ir.khomrehh.antivirus
- ir.dena.antiviruse

دسته سوم:

- com.pdgroup.antivirus
- anti.topsazeh.virus
- Com.antivirus.pixar
- antivirus.full.gentleteam
- ir.antivi.gamejonyor
- com.antivirus.pishgam.ir
- com.antivirus.hacilk
- ir.antivirus.parnian
- ir.onedevelope.newantivirus
- ir.antivirus.hoshmand.apa
- com.zinabeyon.anti
- ir.golpoone.anti
- ir.apphamid.anti
- phone.tools.orke
- com.alirezamaku.antiviruse
- ir.antivirus.apk

دسته چهارم:

- sey.fordoantivirus.ir
- ap.anti.mola
- com.parsina.antivirus
- ir.mf.santivirus
- ir.mf.m.antivirus

دسته پنجم:

- com.foota.anti_virus
- com.app.data.anti_virus
- com.project.antivirus_fafa70
- com.professional.anty
- com.lemon.CibroSecurity
- com.modern.Antivirus
- com.saman.ss98
- com.ironsecurity.pro
- ir.hanogram.antivirus

دسته ششم:

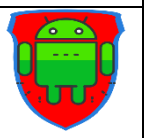
- com.vahid.anitvirus_hamekare
- com.antivirus.gentleteam
- com.appline.security_package

دسته هفتم:

- com.developer.antiscan_v1
- com.sabnam.app
- com.taher.antivirus.mobilesecurity
- com.antivirus.jiro.nikparvar2
- com.amitis.antivirus.security
- com.avin.antivirus.mobilesecurity
- com.appdirac.antivirus.mobilesecurity
- ir.nbnb.antivirus.newmobilesecurity
- com.antivirous.app
- com.example.ehsans11.merikh_antivirus
- com.example.ehsans11.antivirus
- ir.zback.antivirus
- ir.sadra.antivirus

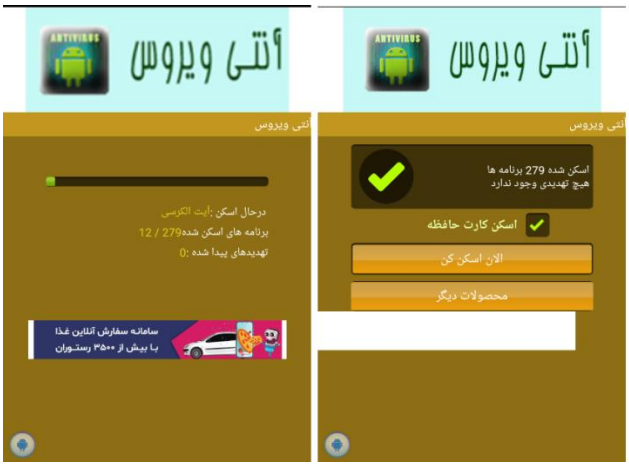
۴ بررسی دسته اول آنتی‌ویروس‌ها

با بررسی کد برنامه‌ها می‌توان متوجه شباهت کامل برنامه‌های زیر شد:

آیکون	نام برنامه	نام بسته	توسعه‌دهنده	نصب	صفحه دانلود در مارکت
	آنتی ویروس	com.farzanehgroup8.antivirus	**********	+۱۰۰۰	**********
	آنتی ویروس ایمن پلاس	com.kianoreeves2.antiviruse menplus	**********	+۱۰۰۰	**********
	آنتی ویروس حرفه‌ای	com.mrfarshid9.herfeiiantivirus	**********	+۲۰۰۰	**********
	آنتی ویروس حرفه ای	com.najmejan4.proantivirus	**********	+۵۰۰	**********
	آنتی ویروس سبز	com.neginmobi2.antivirussabz	**********	+۵۰۰	**********
	آنتی ویروس آسپرین	com.sadrooid15.AntiVirusAspirin	**********	+۲۰۰	**********
	آنتی ویروس حرفه ای	com.mrfarshid13.antivirooshe rfei	**********	+۱۰۰۰	**********
	آنتی ویروس رویا	com.zinatidev4.royaantivirus	**********	+۲۰۰	**********
	آنتی ویروس تمام حرفه‌ای	com.mahnazinco.antivirus	**********	+۱۰۰۰	**********
	آنتی ویروس ایمن	com.tannaztalaee2.antivirus	**********	+۵۰۰	**********

۴-۱ محیط برنامه

با ورود به برنامه، مشاهده می‌شود که بلافاصله آنتی ویروس شروع به اسکن کردن می‌کند و ظاهر بسیار ساده‌ای دارد (شکل ۱).



شکل ۱

۴-۲ تحلیل برنامه

در این مجموعه برنامه‌ها عملکرد آنتی ویروس بسیار ابتدایی است، درواقع از یک دیتابیس آماده در همه برنامه‌ها استفاده شده است که نام فایل آن را vxoid.bin قرار داده‌اند.

این آنتی‌ویروس‌ها برنامه‌های نصب شده در دستگاه اندرویدی را با signature هایی که در این دیتابیس وجود دارد تطبیق می‌دهند و بدافزارها را شناسایی می‌کنند اما دیتابیس مربوطه هرگز به‌روزرسانی نمی‌شود در نتیجه عملکرد این آنتی‌ویروس‌ها برای تشخیص تهدیدات مربوطه بسیار ضعیف است و در عمل قابل قبول نیست.

با توجه به تعداد زیاد آنتی‌ویروس‌های این دسته به نظر می‌رسد که همگی از روی یک برنامه منبع باز کپی شده‌اند و وجود سرویس‌های تبلیغاتی همچون «عدد» ، «کلیک یاب»، «پوشه» و ... همگی نشان از هدف تبلیغاتی منتشرکنندگان این برنامه‌ها است.

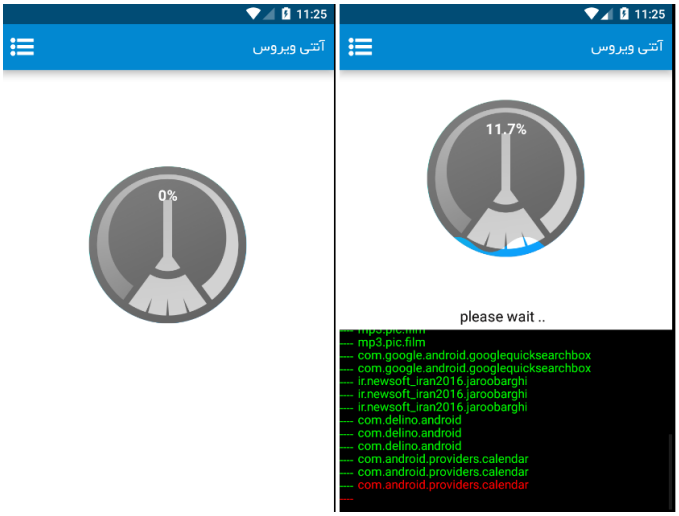
۵ بررسی دسته دوم آنتی‌ویروس‌ها

با بررسی کد برنامه‌ها می‌توان متوجه شباهت کامل برنامه‌های زیر شد:

آیکون	نام برنامه	نام بسته	توسعه‌دهنده	نصب	صفحه دانلود در مارکت
	آنتی ویروس پیشرفته anti v	ir.daryasoft.antiviruses	**********	+۵۰۰۰	**********
	انتی ویروس پیشرفته و هوشمند ۲۰۱۸	ir.maher.antivirus	**********	+۵۰۰۰	**********
	آنتی ویروس پاک کن پیشرفته ۲۰۱۸	ir.khomrehh.antivirus	**********	+۱۰۰۰	**********
	آنتی ویروس پیشرفته	ir.dena.antiviruse	**********	+۲۰۰	**********

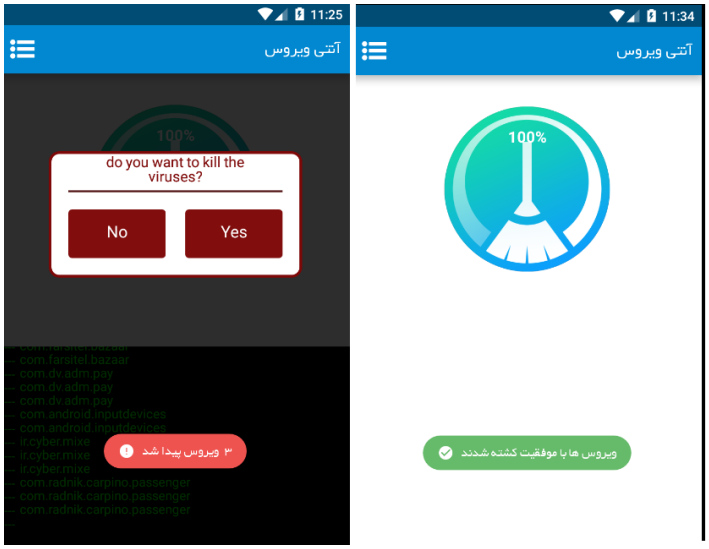
۵-۱ محیط برنامه

با ورود به برنامه، مشاهده می‌شود که بلافاصله آنتی ویروس شروع به اسکن کردن می‌کند و ظاهر بسیار ساده‌ای دارد. پس از زدن روی صفحه برنامه شروع به اسکن کردن فایل‌های روی دستگاه می‌کند.



شکل ۲ اسکن کردن فایل‌ها

پس از اتمام اسکن فایل‌ها پیام زیر ظاهر می‌شود. و سپس طبق ادعای برنامه ویروس‌ها حذف می‌شوند.



شکل ۳ پیام یافتن ویروس و حذف آن‌ها

۲-۵ تحلیل برنامه

با توجه به بررسی کد برنامه، این برنامه اندرویدی با استفاده ابزار Basic4Android ساخته شده است.

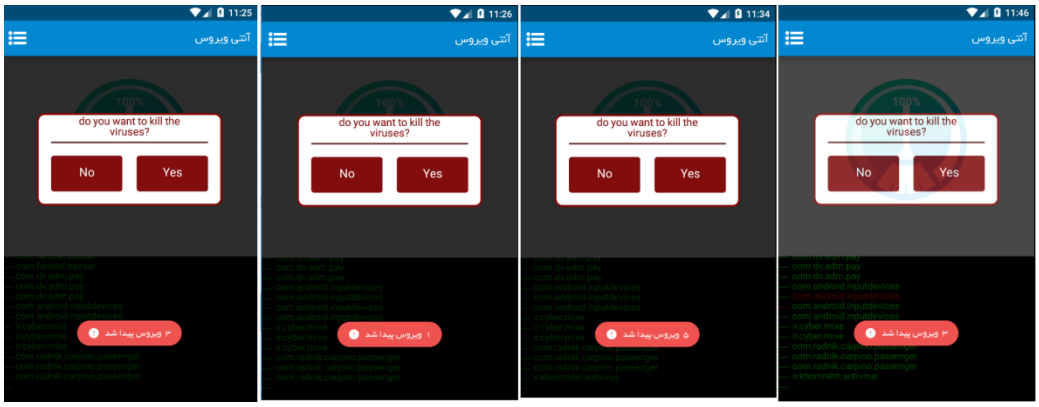
تحلیل کد برنامه نشان می‌دهد که هیچ آنتی‌ویروسی در این برنامه‌ها وجود ندارد و تغییر رنگ برخی از برنامه‌ها در هنگام اسکن به رنگ قرمز به صورت تصادفی رخ می‌دهد. تکه کدهای زیر مربوط به این تغییر رنگ تصادفی است. در این کدها متغیر _eror همان تعداد ویروس‌های کشف شده است که پس از تابع تصادفی افزایش می‌یابد. پس از هر کشف ویروس (به صورت تصادفی!) رنگ نام فایل آن قرمز می‌شود ولی برای کشف ویروس‌ها هیچ تحلیلی رخ نمی‌دهد و صرفا با یک تابع تصادفی انجام می‌شود.

```
public static String _t3_tick() throws Exception {
    if (!_t) {
        mostCurrent._label1.setText(BA.ObjectToCharSequence(BA.NumberToString(((double) _k) / 100.0d) + "%"));
        LabelWrapper labelWrapper = new LabelWrapper();
        labelWrapper.Initialize(mostCurrent.activityBA, "");
        Colors colors = Common.Colors;
        labelWrapper.setTextColor(Colors.Green);
        if (_k % Common.Rnd(400, 600) == 0) {
            _eror++;
            colors = Common.Colors;
            labelWrapper.setTextColor(-65536);
        }
```

```
_p += Common.PerXToCurrent(4.0f, mostCurrent.activityBA);
mostCurrent._scrollview1.setScrollPosition(_p + Common.PerYToCurrent(8.0f, mostCurrent.activityBA));
mostCurrent._scrollview1.getPanel().setColor(-16777216);
mostCurrent._scrollview1.getPanel().setHeight(_p + Common.PerYToCurrent(8.0f, mostCurrent.activityBA));
if (_ss % Common.Rnd(80, 90) == 0) {
    _eror++;
    colors = Common.Colors;
    labelWrapper.setTextColor(-65536);
}
```

شکل ۴ کشف ویروس‌ها به صورت تصادفی

بررسی رفتار آنتی ویروس نیز این نتیجه را تایید کرد و پس از پنج بار اسکن کردن دستگاه و پاک‌سازی آن، آنتی ویروس هر بار ادعای کشف ویروس را می‌کرد. در حالی که قاعدتا پس از پاک‌سازی دستگاه نباید ویروسی روی آن وجود داشته باشد.



شکل ۵ کشف ویروس پس از هر اسکن

۶ بررسی دسته سوم آنتی‌ویروس‌ها

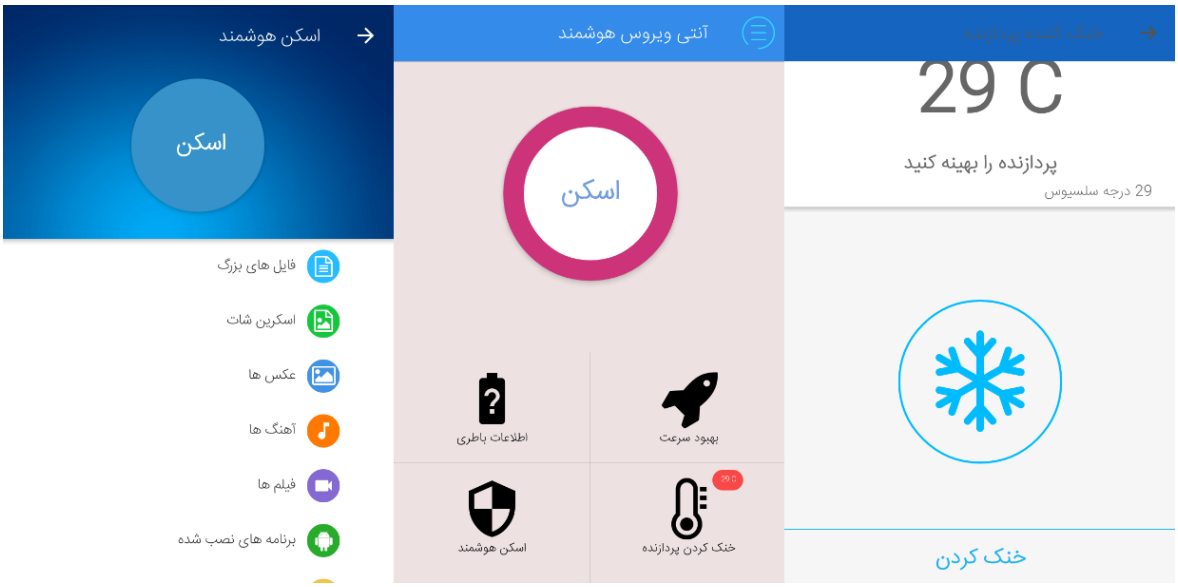
با بررسی کد برنامه‌ها می‌توان متوجه شباهت کامل برنامه‌های زیر شد:

آیکون	نام برنامه	نام بسته	توسعه دهنده	تعداد نصب	صفحه دانلود در مارکت
	ویروس پاک کن امنیت بالا	com.pdgroup.antivirus	**********	+۵۰۰۰۰	********** **********
	آنتی ویروس و بهینه ساز	anti.topsazeh.virus	**********	+۲۰۰۰۰	********** **********
	انتی ویروس پیشرفه پیکسار	Com.antivirus.pixar	**********	+۲۰۰۰۰	********** **********
	آنتی ویروس هوشمند	antivirus.full.gentleteam	**********	+۱۰۰۰۰	********** **********
	آنتی ویروس برتر	ir.antivi.gamejonyor	**********	+۱۰۰۰	********** **********

ویروس کش فوق پیشرفته	com.antivirus.pishgam.ir	**********	+۵۰۰۰	**********	
آنتی ویروس هوشمند (همکاره)	com.antivirus.hacilk	**********	+۵۰۰۰	**********	
ضد ویروس پرنیان	ir.antivirus.parnian	**********	+۲۰۰۰	**********	
آنتی ویروس هوشمند ۲۰۱۸	ir.onedevelope.newantivirus	**********	+۱۰۰۰	**********	
ویروس پاک‌کن (anti v)۲۰۱۹_۲۰۱۸	ir.antivirus.hoshmand.apa	**********	+۲۰۰۰	**********	
آنتی ویروس (امنیت بالا) پاک کن	com.zinabeyon.anti	**********	+۱۰۰۰	**********	
ویروس پاک کن (امنیت بالا)	ir.golpoone.anti	**********	+۱۰۰۰	**********	
آنتی ویروس پیشرو	ir.apphamid.anti	**********	+۵۰۰	**********	
آنتی ویروس قدرتمند اریکه	phone.tools.orke	**********	+۱۰۰	**********	
آنتی ویروس ماکوسافتز	com.alirezamaku.antiviruse	**********	-۱۰۰	**********	
آنتی ویروس گوشی	ir.antivirus.apk	**********	-۱۰۰	**********	

۶-۱ محیط برنامه

با ورود به برنامه، مشاهده می‌شود که برنامه چهار عملکرد ادعایی دارد: بهبود سرعت، نمایش اطلاعات باتری گوشی، خنک‌کردن پردازنده و اسکن هوشمند (شکل ۶).



شکل ۶

۶-۲ تحلیل برنامه

برای بررسی نحوه عملکرد برنامه سراغ سورس کدهای این برنامه می‌رویم. به عنوان مثال منطق عملکرد "خنک‌کردن پردازنده" و "اسکن هوشمند" را بررسی می‌کنیم.

۶-۲-۱ خنک‌کردن پردازنده

در میان کلاس‌های موجود در پکیج `com.antivirus.hacilk`، کلاسی با نام `cpu_cooler` وجود دارد که وظیفه خنک‌کردن پردازنده را بر عهده دارد. به هنگام ورود به این قسمت از برنامه، کدهای موجود در تابع `_activity_create` اجرا می‌شوند. پس باید در این تابع کدهای مربوط به دریافت دمای پردازنده وجود داشته باشد. قسمتی از کدهای این تابع که مربوط به نمایش دما است در شکل ۷ نشان داده شده است.

```
public static String _activity_create(boolean z) throws Exception {
    mostCurrent._activity.LoadLayout("cpu_coooler", mostCurrent.activityBA);
    mostCurrent._time.Initialize(processBA, "time", 8000);
    .
    .
    .
  append = new StringBuilder().append(" ");
  main = mostCurrent._main;
  labelWrapper.setText(BA.ObjectToCharSequence(append.append(BA.NumberToString(main._dma)).append(" درجه سلسیوس ").toString()));
  mostCurrent._loadingindicatorview1.setVisible(false);
```

شکل ۷

با توجه به کد بالا، مقدار متغیر `_dma` از کلاس `main` نشان‌دهنده دمای فعلی پردازنده است. با مراجعه به کلاس `main` به دنبال کدهایی می‌گردیم که این متغیر را مقداردهی کرده باشند. کد مربوط به مقداردهی این متغیر در شکل ۸ نشان داده شده است.

```
mostCurrent._pc.Initialize(mostCurrer
_dma = Common.Rnd(20, 32);
mostCurrent._label10_cpu_cooler.setTe
BA ba2 = processBA;
pushejsonservice pushejsonservice = n
```

```
public static int Rnd(int Min, int Max) {
    if (random == null) {
        random = new Random();
    }
    return random.nextInt(Max - Min) + Min;
```

شکل ۸

با توجه به این کدها به سادگی مشخص است که برنامه یک عدد تصادفی از بازه 20 تا 32 انتخاب می‌کند و به عنوان دمای فعلی پردازنده اعلام می‌کند. همچنین در ابتدای تابع `_activity_create` مشخص شده است که پس از سپری‌شدن 8000 میلی‌ثانیه، تابع `_time_tick()` اجرا شود (شکل ۹).

```
public static String _activity_create(boolean z) throws Exception {
    mostCurrent._activity.LoadLayout("cpu_coooler", mostCurrent.activityBA);
    mostCurrent._time.Initialize(processBA, "time", 8000);
```

شکل ۹

با کلیک کاربر روی دکمه "خنک‌کردن"، تایمر 8 ثانیه‌ای شروع می‌شود و نهایتا تابع `_time_tick()` صدا زده می‌شود. با مشاهده کدهای این تابع مشخص می‌شود که برنامه دمای بعد از عملیات "خنک‌کردن" را به صورت تصادفی از بازه 10 تا 20 انتخاب می‌کند (شکل ۱۰).

```
public static String _time_tick() throws Exception {
    :
    main main = mostCurrent._main;
    main._dma = Common.Rnd(10, 20);
    LabelWrapper labelWrapper = mostCurrent._label1_celcious;
    StringBuilder append = new StringBuilder().append(" ");
    main main2 = mostCurrent._main;
    labelWrapper.setText(BA.ObjectToCharSequence(append.append(BA.NumberToString(main._dma)).append(" C").toString()));
    labelWrapper = mostCurrent._label1_daraje_c;
    append = new StringBuilder().append(" ");
    main2 = mostCurrent._main;
    labelWrapper.setText(BA.ObjectToCharSequence(append.append(BA.NumberToString(main._dma)).append("درجه سلسیوس ").toString()));
```

شکل ۱۰

۶-۲-۲ اسکن هوشمند

این قابلیت نیز مانند قابلیت مذکور، تنها چند ثانیه کاربر را منتظر نگه‌می‌دارد و سپس اعلام می‌کند که عملیات اسکن به پایان رسیده است. کدهای مربوط به "اسکن هوشمند" در کلاس `antivirous_activity` از پکیج `com.antivirus.hacilk` قرار دارند. توالی کدهای اجرایی در این قسمت از برنامه در شکل ۱۱ نشان داده شده است.

```
public static String _activity_create(boolean z) throws Exception {
    mostCurrent._activity.LoadLayout("ativirous_scan_dis", mostCurren
    mostCurrent._scrollview1.getPanel().LoadLayout("anti_virous_wio",
    mostCurrent._time.Initialize(processBA, "time", 10000);
    codes codes = mostCurrent._codes;
```

```
public static String _panel2_scan_click()
    mostCurrent._panel2_scan.setVisible(fa
    mostCurrent._percent2.SetVisibleAnimat
    mostCurrent._scan_load.setVisible(true
    mostCurrent._toolbar.setTitle(BA.Objec
    codes codes = mostCurrent._codes;
    codes._toolbar_icon(mostCurrent.activi
    mostCurrent._label2_addres.setVisible(
    mostCurrent._time.setEnabled(true);
    return "";
```

```
public static String _time_tick() throws Exception {
    mostCurrent._time.setEnabled(false);
    mostCurrent._panel2_scan.setVisible(true);
    mostCurrent._percent2.SetVisibleAnimated(350, false);
    mostCurrent._scan_load.setVisible(false);
    mostCurrent._toolbar.setTitle(BA.ObjectToCharSequence("اسکن هوشمند"));
    codes codes = mostCurrent._codes;
    codes._toolbar_icon(mostCurrent.activityBA, mostCurrent._toolbar, "ic_arrow_right_white_24dp");
    mostCurrent._label2_addres.setText(BA.ObjectToCharSequence("فایل های مشکوک پاکسازی و دستگاه بهینه شد"));
```

شکل ۱۱

۷ بررسی دسته چهارم آنتی‌ویروس‌ها

با بررسی ظاهر، عملکرد و کد برنامه‌ها می‌توان متوجه شباهت کامل برنامه‌های زیر شد:

آیکون	نام برنامه	نام بسته	توسعه‌دهنده	تعداد نصب	صفحه دانلود در مارکت
	AntiVirus Fordo	sey.fordoantivirus.ir	**********	+۱۰۰۰۰	**********
	آنتی‌ویروس قوی ۲۰۱۸	ap.anti.mola	**********	+۵۰۰۰	**********

	آنتی‌ویروس پیشرفته پارس	com.parsina.antivirus	**********	+۵۰۰۰	********** *********
	آنتی‌ویروس هوشمند سایا	ir.mf.santivirus	**********	+۱۰۰۰	********** *********
	آنتی‌ویروس هوشمند	ir.mf.m.antivirus	**********	+۱۰۰۰	********** *********

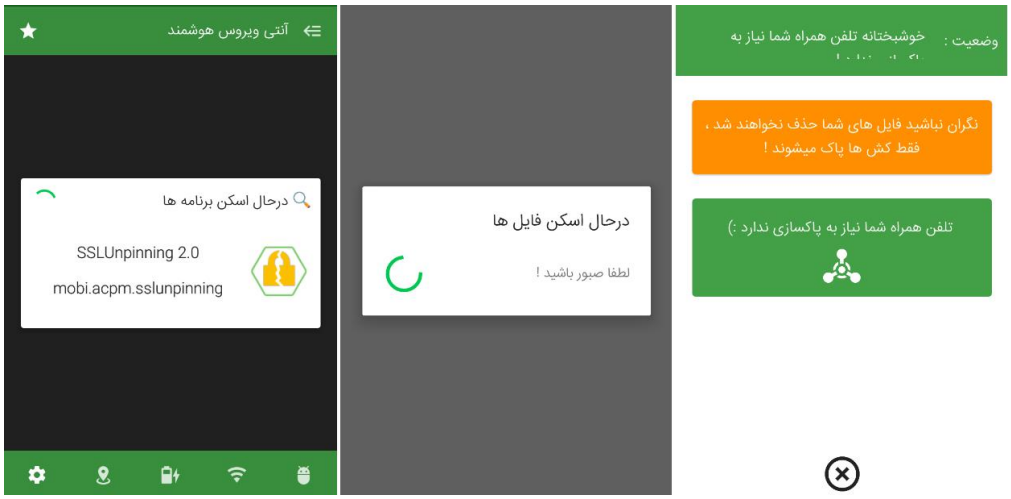
۱-۷ محیط برنامه

محیط اولیه ورود به برنامه مطابق شکل ۱۲ است.

شکل ۱۲ – صفحه اول برنامه

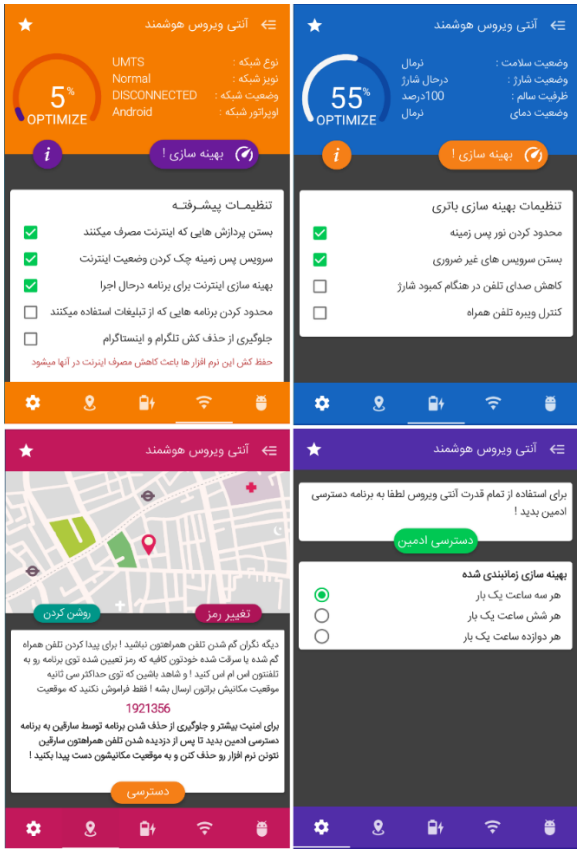
با کلیک بر روی دکمه‌های قسمت ۱ از شکل ۱۲، قابلیت موردنظر فعال یا غیرفعال می‌شود. با کلیک بر روی قسمت ۲ از شکل ۱۲، صفحه‌ای مانند شکل ۱۳ (چپ) نمایش داده است که به نظر می‌رسد برنامه در حال

اسکن برنامه‌های نصب‌شده بر روی گوشی است. بعد از اتمام بررسی برنامه‌های نصب‌شده، شکل ۱۳ (وسط) نشان داده می‌شود. با اتمام این مرحله، شکل ۱۳ (راست) نشان‌داده می‌شود.



شکل ۱۳ – اسکن برنامه‌های نصب‌شده و فایل‌های گوشی

همچنین در قسمت ۳ از شکل ۱۲، می‌توانیم بین صفحات مختلف برنامه حرکت کنیم و قابلیت‌های مختلف آن را مشاهده کنیم. این قابلیت‌ها که عبارتند از بهینه‌سازی شبکه، بهینه‌سازی باتری، ویژگی ضد سرقت و تنظیمات، همگی در شکل ۱۴ نشان داده شده‌اند.

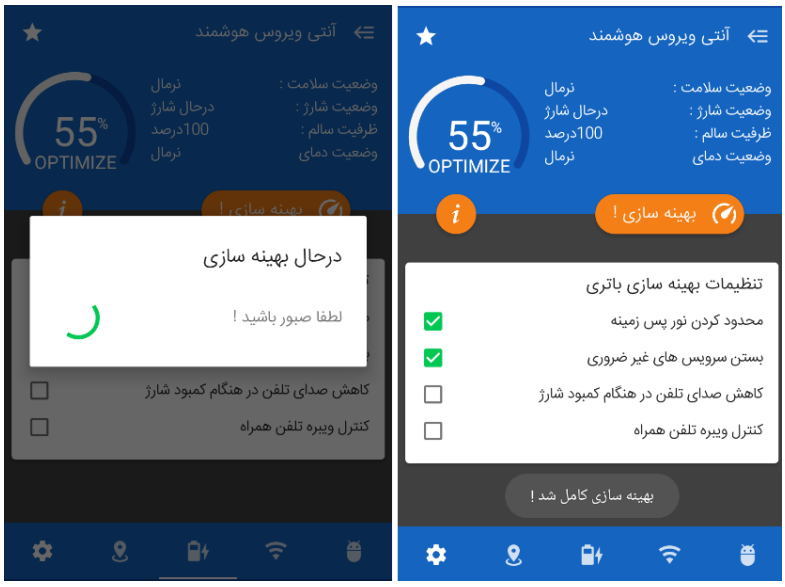


شکل ۱۴ – قابلیت‌های برنامه

با کلیک بر روی دکمه بهینه‌سازی در قسمت بهینه‌سازی شبکه، بعد از چند ثانیه اعلانی با عنوان "بهینه‌سازی با موفقیت انجام شد" نشان داده می‌شود (شکل ۱۵).

شکل ۱۵ – قابلیت بهینه‌سازی شبکه

به طور مشابه، در قسمت بهینه‌سازی باتری نیز، بعد از چند ثانیه اعلانی با مضمون موفقیت‌آمیز بودن عملیات بهینه‌سازی نشان داده می‌شود (شکل ۱۶).



شکل ۱۶ – قابلیت بهینه‌سازی باتری

در قسمت ضدسرقت برنامه، برنامه ادعا می‌کند که به کمک این قابلیت، اگر پیامکی محتوی کد خاص (در شکل ۱۹۲۱۳۵۶) دریافت شود، برنامه مکان گوشی را به فرستنده پیامک، ارسال می‌کند (شکل ۱۷).

شکل ۱۷ – قابلیت ضدسرقت

در قسمت تنظیمات، برنامه به کاربر توصیه می‌کند که کاربر قابلیت مدیر را برای برنامه فعال کند تا در صورت سرقت گوشی، سارقین قادر به حذف برنامه از گوشی نباشند (شکل ۱۸).

شکل ۱۸ – صفحه تنظیمات برنامه

۲-۷ تحلیل برنامه

در این قسمت به بررسی سورس‌کد هر یک از بخش‌های برنامه می‌پردازیم.

۷-۲-۱ قابلیت ضدجاسوسی

این قابلیت در قسمت ۱ از شکل ۱۲، قابل مشاهده است. با بررسی کد مربوط به این قابلیت به این نتیجه می‌رسیم که برنامه با فعال‌بودن و غیرفعال‌شدن این قابلیت هیچ‌کاری در پس‌زمینه انجام نمی‌دهد و تنها عکس روی دکمه را تغییر می‌دهد (شکل ۱۹).

```
public static String _antispy_click() throws Exception {
    ImageViewWrapper imageViewWrapper;
    File file;
    switch (BA.switchObjectToInt(_sp.GetBoolean("spy", Boolean.valueOf(true)), Boolean.valueOf(true), Boolean.valu
        case 0:
            _sp.SaveBoolean("spy", Boolean.valueOf(false));
            imageViewWrapper = mostCurrent._antispy;
            file = Common.File;
            imageViewWrapper.setBitmap((Bitmap) Common.LoadBitmap(File.getDirAssets(), "n_spy.png").getObject());
            break;
        case 1:
            _sp.SaveBoolean("spy", Boolean.valueOf(true));
            imageViewWrapper = mostCurrent._antispy;
            file = Common.File;
            imageViewWrapper.setBitmap((Bitmap) Common.LoadBitmap(File.getDirAssets(), "y_spy.png").getObject());
            break;
    }
    return "";
}
```

شکل ۱۹ – قابلیت ضدجاسوسی

۷-۲-۲ قابلیت اسکن بلادرنگ

این قابلیت نیز مشابه قسمت ۷-۲-۱، هیچ‌کاری انجام نمی‌دهد و تنها عکس روی دکمه را به منزله فعال و غیرفعال‌شدن قابلیت تغییر می‌دهد (شکل ۲۰).

```
public static String _ns_scan_click() throws Exception {
    ImageViewWrapper imageViewWrapper;
    File file;
    switch (BA.switchObjectToInt(_sp.GetBoolean("scan", Boolean.valueOf(true)), Boolean.valueOf(true), Boolean.valu
        case 0:
            _sp.SaveBoolean("scan", Boolean.valueOf(false));
            imageViewWrapper = mostCurrent._ns_scan;
            file = Common.File;
            imageViewWrapper.setBitmap((Bitmap) Common.LoadBitmap(File.getDirAssets(), "n_scan.png").getObject());
            break;
        case 1:
            _sp.SaveBoolean("scan", Boolean.valueOf(true));
            imageViewWrapper = mostCurrent._ns_scan;
            file = Common.File;
            imageViewWrapper.setBitmap((Bitmap) Common.LoadBitmap(File.getDirAssets(), "y_scan.png").getObject());
            break;
    }
    return "";
}
```

شکل ۲۰ – قابلیت اسکن بلادرنگ

۷-۲-۳ قابلیت دیواره‌آتش

این قابلیت نیز مشابه قسمت ۷-۲-۱، هیچ‌کاری انجام نمی‌دهد و تنها عکس روی دکمه را به منزله فعال و غیرفعال‌شدن قابلیت تغییر می‌دهد (شکل ۲۱).

```
public static String _firewall_click() throws Exception {
    ImageViewWrapper imageViewWrapper;
    File file;
    switch (BA.switchObjectToInt(_sp.GetBoolean("firewall", Boolean.valueOf(false)), Boolean.valueOf(true), Boolean.
        case 0:
            _sp.SaveBoolean("firewall", Boolean.valueOf(false));
            imageViewWrapper = mostCurrent._firewall;
            file = Common.File;
            imageViewWrapper.setBitmap((Bitmap) Common.LoadBitmap(File.getDirAssets(), "n_fire.png").getObject());
            break;
        case 1:
            _sp.SaveBoolean("firewall", Boolean.valueOf(true));
            imageViewWrapper = mostCurrent._firewall;
            file = Common.File;
            imageViewWrapper.setBitmap((Bitmap) Common.LoadBitmap(File.getDirAssets(), "y_fire.png").getObject());
            break;
    }
    return "";
}
```

شکل ۲۱ – قابلیت دیواره آتش

۷-۲-۴ اسکن‌برنامه‌ها و فایل‌ها

همان‌طور که در شکل ۱۳ نشان داده شد، برنامه با کلیک بر روی قسمت ۲ از شکل ۱۲، شروع به اسکن برنامه‌های نصب‌شده و فایل‌های روی گوشی می‌کند. در ابتدای کار، برنامه متد _btnscan_click را صدا می‌زند. این متد تنها لیست برنامه‌های نصب‌شده روی گوشی را پشت سرهم به کاربر نشان می‌دهد و بین هر دو برنامه، ۳۰۰ میلی‌ثانیه متوقف می‌شود تا کاربر تصور کند برنامه در حال بررسی برنامه‌ها است در حالی که برنامه هیچ بررسی‌ای روی برنامه‌ها انجام نمی‌دهد (شکل ۲۲).

```
public static String _btnscan_click() throws Exception {
    String NumberToString = BA.NumberToString(_listapp.getSize());
    Colors colors = Common.Colors;
    Common.LogColor(NumberToString, Colors.Magenta);
    _dialogbiulder();
    BitmapDrawable bitmapDrawable = new BitmapDrawable();
    int size = _listapp.getSize() - 1;
    for (int i = 0; i <= size; i = (i + 0) + 1) {
        mostCurrent._lbl2.setText(mostCurrent._pm._getapplicationname(BA.ObjectToString(_listapp.Get(i))
        mostCurrent._lbl3.setText(_listapp.Get(i));
        try {
            mostCurrent._imgapp.setBackground((Drawable) mostCurrent._pm._getapplicationicon(BA.ObjectTo
        } catch (Exception e) {
            processBA.setLastException(e);
            Common.Log(BA.ObjectToString(Common.LastException(mostCurrent.activityBA)));
        }
        _sleep(300);
    }
    _sleep(100);
    mostCurrent._shadow.setVisible(false);
    mostCurrent._pnldialog.setVisible(false);
    BA ba = mostCurrent.activityBA;
    result result = mostCurrent._result;
    Common.StartActivity(ba, result.getObject());
    return "";
}
```

شکل ۲۲ – اسکن برنامه‌ها

پس از اسکن برنامه‌ها، برنامه اکتیویتی result را فراخوانی می‌کند تا به اسکن فایل‌های گوشی بپردازد اما برنامه در حقیقت هیچ اسکن‌ای روی فایل‌های گوشی انجام نمی‌دهد. تنها کاری که این در این قسمت انجام می‌شود، حذف فایل‌های موجود در پوشه /Telegram/Telegram Images است (شکل ۲۳). در صورتی که تلگرام روی گوشی نصب باشد، پیام "تلفن‌همراه شما نیاز به پاکسازی دارد" نشان داده می‌شود، در غیراین صورت پیام "تلفن‌همراه شما نیاز به پاکسازی ندارد" نشان داده می‌شود. پس در این قسمت از برنامه نیز هیچ کار مفیدی انجام نمی‌شود.

```
public static String _createlistfile() throws Exception {
    try {
        List list = mostCurrent._listfile;
        File file = Common.File;
        StringBuilder stringBuilder = new StringBuilder();
        File file2 = Common.File;
        list.Initialize2(File.ListFiles(stringBuilder.append(File.getDirRootExternal()).append("/Telegram/Telegram Images").to
    } catch (Exception e) {
        processBA.setLastException(e);
        Common.Log(BA.ObjectToString(Common.LastException(mostCurrent.activityBA)));
    }
    return "";
}
```

شکل ۲۳ - حذف عکس‌های موجود در پوشه تلگرام

۷-۲-۵ قابلیت بهینه‌سازی شبکه

با کلیک بر روی دکمه "بهینه‌سازی" در شکل ۱۵، متد optimizenet_ فراخوانی می‌شود. همان‌طور که در شکل ۲۴ مشخص است، برنامه تنها به صورت تصادفی با فراخوانی sleep صبر می‌کند تا کاربر تصور کند برنامه در حال انجام کاری در پس‌زمینه است در حالی که هیچ کار مفیدی انجام نمی‌شود. در انتها نیز پیام "بهینه‌سازی با موفقیت انجام شد" به کاربر نشان داده می‌شود.

```
public static String _optimizenet() throws Exception {
    for (int i = 5; i <= 100; i = (i + 0) + 1) {
        mostCurrent._masterpb.setProgress(i);
        _sleep((long) Common.Rnd(10, 120));
    }
    Common.ToastMessageShow("بهینه سازی با موفقیت انجام شد !", true);
    _sp.SaveBoolean("netopt", Boolean.valueOf(true));
    return "";
}
```

شکل ۲۴ - بهینه‌سازی شبکه

۷-۲-۶ قابلیت بهینه‌سازی باتری

با بررسی سورس‌کد این قابلیت نیز مشخص شد که برنامه تنها با کاهش نور صفحه نمایش گوشی و غیرفعال‌کردن ویبره گوشی اقدام به بهینه‌سازی باتری گوشی می‌کند و کار دیگری انجام نمی‌دهد. به عنوان مثال همان‌طور که در شکل ۲۵ مشخص است، اگر میزان شارژ باتری از ۲۰ درصد کمتر باشد، برنامه میزان نور صفحه نمایش را به ۰,۲ کاهش می‌دهد و سپس ویبره گوشی را نیز خاموش می‌کند.

```
public static String _dialog_batt() throws Exception {
    Phone phone = new Phone();
    mostCurrent._dialogbat.Initialize(mostCurrent.activityBA, "db");
    mostCurrent._dialogbat.title("درحال بهینه سازی");
    mostCurrent._dialogbat.content("لطفا صبور باشید !");
    mostCurrent._dialogbat.autoDismiss(false);
    mostCurrent._dialogbat.canceledOnTouchOutside(false);
    mostCurrent._dialogbat.cancelable(false);
    mostCurrent._dialogbat.progressType(true, 1);
    MaterialDialogBuilder materialDialogBuilder = mostCurrent._dialogbat;
    TypefaceWrapper typefaceWrapper = Common.Typeface;
    Typeface LoadFromAssets = TypefaceWrapper.LoadFromAssets("font.ttf");
    TypefaceWrapper typefaceWrapper2 = Common.Typeface;
    materialDialogBuilder.typeface(LoadFromAssets, TypefaceWrapper.LoadFromAssets
    mostCurrent._dialogbat.build();
    mostCurrent._dialogbat.show();
    mostCurrent._ttt.Initialize(processBA, "ttt", (long) Common.Rnd(4000, 9000));
    mostCurrent._ttt.setEnabled(true);
    if (_getbatterylevel() < 20) {
        Phone.SetScreenBrightness(processBA, 0.2f);
        Phone.SetRingerMode(0);
    } else if (_getbatterylevel() < 30) {
        Phone.SetScreenBrightness(processBA, 0.3f);
        Phone.SetRingerMode(0);
    } else if (_getbatterylevel() < 50) {
        Phone.SetScreenBrightness(processBA, 0.6f);
    } else if (_getbatterylevel() < 70) {
        Phone.SetScreenBrightness(processBA, 0.75f);
    } else if (_getbatterylevel() < 80) {
        Phone.SetScreenBrightness(processBA, 0.8f);
    } else if (_getbatterylevel() < 95) {
        Phone.SetScreenBrightness(processBA, 0.9f);
    }
    return "";
}
```

شکل ۲۵ - بهینه‌سازی باتری

۷-۲-۷ قابلیت ضدسرقت

برنامه سرویسی با نام smslocation دارد که به هنگام دریافت پیامک فعال می‌شود. این سرویس اقدام به استخراج متن و شماره فرستنده پیامک می‌کند اما اقدام به ارسال مکان فعلی گوشی به فرستنده نمی‌کند (شکل ۲۶). این قابلیت روی دو گوشی واقعی نیز تست شد ولی پیامکی حاوی مکان گوشی دریافت نشد. از این رو این عملکرد برنامه نیز کار مفیدی انجام نمی‌دهد.

```
public static String _service_start(IntentWrapper intentWrapper) throws Exception {
    home home = mostCurrent._home;
    home._sp.Initialize("SP");
    JavaObject javaObject = new JavaObject();
    javaObject.InitializeContext(processBA);
    _num = BA.ObjectToString(javaObject.RunMethod(HttpRequest.METHOD_GET, new Object[]{"number", inten
    _body = BA.ObjectToString(javaObject.RunMethod(HttpRequest.METHOD_GET, new Object[]{"body", intent
    String str = _body;
    home home2 = mostCurrent._home;
    if (!str.equals(home._sp.GetString("code", "1921356"))) {
        Common.Log("Dont Send1");
    }
    return "";
}
```

شکل ۲۶ – قابلیت ضدسرقت

اگرچه در سرویس smslocation، متدی با نام _location_locationchanged وجود دارد که اقدام به ارسال پیامک به فرستنده پیامک حاوی کدخاص می‌کند اما این تابع هیچ جایی از برنامه فراخوانی نمی‌شود (شکل ۲۷).

```
public static String _location_locationchanged(double d, double d2, double d3, float f, float f2, String str, float f3, long j)
    String str2 = "";
    str2 = _body;
    home home = mostCurrent._home;
    if (str2.equals(home._sp.GetString("code", "1921356"))) {
        PhoneSms phoneSms = new PhoneSms();
        PhoneSms.Send(_num, "https://www.google.com/maps/@" + BA.NumberToString(d2) + "," + BA.NumberToString(d) + ",16.25z");
    } else {
        Common.Log("Dont Send2");
    }
    return "";
}
```

شکل ۲۷

۷-۲-۸ قابلیت مدیریت

برنامه برای جلوگیری از حذف‌شدن برنامه توسط سارقین، از کاربر درخواست فعال‌سازی مدیر را می‌کند و از این قابلیت در برنامه سوءاستفاده نمی‌کند.

۷-۲-۹ سرویس‌های تبلیغاتی برنامه

برنامه دو سرویس تبلیغاتی به نام‌های bsservice و pushejsonservice دارد که به ترتیب مربوط به سرویس‌های تبلیغاتی onesignal و پوشه هستند. عملکرد هر دو سرویس بسیار شبیه به یکدیگر است از این رو به عنوان نمونه، قابلیت‌های موجود در سرویس pushejsonservice به صورت لیست‌وار ذکر می‌شود. این سرویس بر اساس مقدار متغیر type در پیام دریافتی عملیات زیر را انجام می‌دهد:

- اگر ”type=”dialog باشد: نمایش یک دیالوگ حاوی یک لینک تبلیغاتی
- اگر ”type=”popup باشد: انتشار یک intent از نوع android.intent.action.VIEW برای نمایش یک لینک تبلیغاتی
- اگر ”type=” viewjoin باشد: باز کردن تلگرام (نسخه رسمی یا غیررسمی) نصب‌شده روی گوشی و نمایش یک کانال
- اگر ”type=”download باشد: دانلود یک فایل APK از لینک موجود در پیام دریافتی و نصب آن روی گوشی کاربر

۷-۲-۱۰ وجود کد خطرناک

در کدهای برنامه کلاسی با نام RootCmd وجود دارد که در صورت روت بودن گوشی، این کلاس به راحتی با اجرای یک دستور در ترمینال، می‌تواند به راحتی بدون آنکه کاربر متوجه شود، هر برنامه‌ای را روی گوشی

نصب کند (شکل ۲۸). اگرچه در هیچ جای برنامه از این قطعه کد استفاده نشده است، اما وجود چنین قطعه کد خطرناکی می‌تواند نشان‌دهنده امکان سوءاستفاده توسعه‌دهندگان باشد.

```
public final class RootCmd {
    private static boolean mHaveRoot = false;
    private static String tag = "B4A";

    public Boolean InstallApk(String apkAbsolutePath) {
        if (haveRoot()) {
            String retString = execRootCmd("pm install -r " + apkAbsolutePath);
            Log.i(tag, retString);
            if (retString.indexOf("Success") > -1) {
                return Boolean.valueOf(true);
            }
            return Boolean.valueOf(false);
        }
        Log.i(tag, "NoRoot");
        return Boolean.valueOf(false);
    }
```

شکل ۲۸ – کد خطرناک موجود در برنامه

۳-۷ نتیجه‌گیری

این دسته از آنتی‌ویروس‌ها تنها عملکرد بهینه‌سازی باتری (که جز وظایف یک آنتی‌ویروس نیست) را تا حدودی انجام می‌دهند و از نظر بررسی فایل‌های مخرب هیچ عملکردی مفیدی ندارند و تنها با هدف تبلیغات ساخته شده‌اند

۸ بررسی دسته پنجم آنتی‌ویروس‌ها

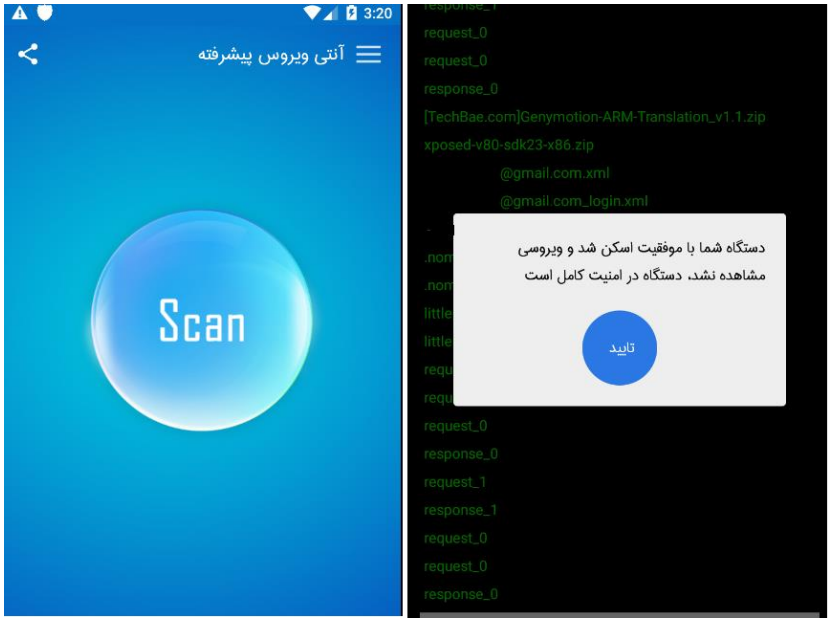
با بررسی ظاهر، عملکرد و کد برنامه‌ها می‌توان متوجه شباهت کامل برنامه‌های زیر شد:

آیکون	نام برنامه	نام بسته	توسعه‌دهنده	تعداد نصب	صفحه دانلود در مارکت
	آنتی‌ویروس پیشرفته	com.foota.anti_virus	**********	+۲۰۰۰۰	**********
	آنتی‌ویروس گلادیاتور	com.app.data.anti_virus	**********	+۱۰۰۰۰	**********
	ضد ویروس	com.project.antivirus_fafa70	**********	+۱۰۰۰۰	**********

	آنتی‌ویروس حرفه‌ای	com.professional.anty	**********	+۵۰۰۰۰	**********
	آنتی‌ویروس هوشمند سیبرو	com.lemon.CibroSecurity	**********	+۲۰۰۰۰	**********
	آنتی‌ویروس مدرن	com.modern.Antivirus	**********	+۲۰۰۰۰	**********
	آنتی‌ویروس همه کاره	com.saman.ss98	**********	+۱۰۰۰۰	**********
	آنتی‌ویروس هوشمند آیرون - ۲۰۱۸	com.ironsecurity.pro	**********	+۵۰۰۰	**********
	آنتی‌ویروس هوشمند هانوگرام	ir.hanogram.antivirus	**********	+۲۰۰۰	**********

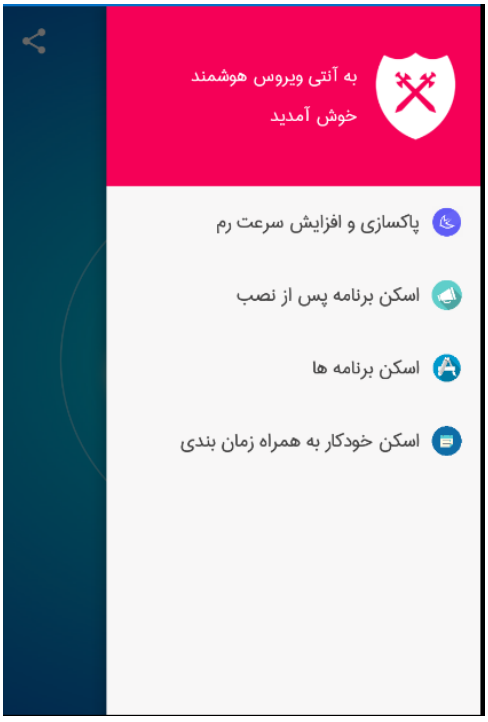
۸-۱ محیط برنامه

با ورود به برنامه، مشاهده می‌شود که آنتی‌ویروس ظاهر بسیار ساده‌ای دارد. پس از زدن روی صفحه برنامه شروع به اسکن کردن فایل‌های روی دستگاه می‌کند.



شکل ۲۹ اسکن کردن فایل‌ها

منو کناری برنامه نیز شامل گزینه‌های پاکسازی و افزایش سرعت رم، اسکن پس از نصب و اسکن خودکار به همراه زمان‌بندی است.



شکل ۳۰ منو کناری برنامه

تصویر زیر نیز مربوط به پاکسازی و افزایش سرعت رم است که در همه برنامه‌های لیست بالا، به شکل زیر است

شکل ۳۱ پاکسازی و افزایش سرعت رم

۸-۲ تحلیل برنامه

همان‌طور که گفته شد این برنامه‌ها عملکردهای کمی دارند در ادامه هر یک از این عملکردها بررسی می‌شوند.

۸-۲-۱ اسکن

اسکن فایل‌ها و برنامه‌ها به منظور یافتن ویروس و برنامه‌های مشکوک انجام می‌شود. با فشردن دکمه scan در صفحه اصلی آنتی‌ویروس این کار انجام می‌شود. تحلیل کد برنامه نشان می‌دهد که این آنتی‌ویروس‌ها صرفا در صورتی که فرمت یک فایل، یکی از انواع زیر باشد آن را به عنوان مورد مشکوک در نظر گرفته (بدون لحاظ کردن محتوا) و با توجه به خواست کاربر حذف می‌کند:

- .exe
- .elf
- .app
- .exp
- .tmp
- .temp
- .cmd
- .ime
- .bin
- .bis
- .ini
- .chm

- .ipa
- .mex
- .msi
- .prg

قطعه کدهای زیر مربوط به تشخیص چنین فایل‌هایی به عنوان فایل مشکوک و سپس نمایش دادن تعداد آن‌ها به کاربر است.

```
class getFileListTask extends AsyncTask {
    getFileListTask() {
    }

    protected String doInBackground(Object[] params) {
        List<File> files = FilesListActivity.this.storage.getNestedFiles("/");
        Log.i("myLog", "doInBackground: " + files.size() + "");
        int i = 0;
        while (i < files.size()) {
            FilesListActivity.fileArray.add(((File) files.get(i)).getName().toString());
            if (((File) files.get(i)).getName().toString().endsWith(".exe")
                || ((File) files.get(i)).getName().toString().endsWith(".elf")
                || ((File) files.get(i)).getName().toString().endsWith(".app")
                || ((File) files.get(i)).getName().toString().endsWith(".exp")
                || ((File) files.get(i)).getName().toString().endsWith(".tmp")
                || ((File) files.get(i)).getName().toString().endsWith(".temp")
                || ((File) files.get(i)).getName().toString().endsWith(".cmd")
                || ((File) files.get(i)).getName().toString().endsWith(".ime")
                || ((File) files.get(i)).getName().toString().endsWith(".bin")
                || ((File) files.get(i)).getName().toString().endsWith(".bis")
                || ((File) files.get(i)).getName().toString().endsWith(".ini")
                || ((File) files.get(i)).getName().toString().endsWith(".chm")
                || ((File) files.get(i)).getName().toString().endsWith(".ipa")
                || ((File) files.get(i)).getName().toString().endsWith(".mex")
                || ((File) files.get(i)).getName().toString().endsWith(".msi")
                || ((File) files.get(i)).getName().toString().endsWith(".prg")) {

                    FilesListActivity.this.virusArray.add(((File) files.get(i)).toString());
            }
            i++;
        }
        return "";
    }
```

شکل ۳۲ تشخیص فایل‌ها با فرمت‌های خاص به عنوان فایل مشکوک

```
private void showDialog() {
    Typeface typeface = Typeface.createFromAsset(getAssets(), "iran_sans.ttf");
    Dialog dialog = new Dialog(this, R.style.CustomDialog);
    dialog.requestWindowFeature(1);
    dialog.setContentView(R.layout.final_delete_dialog);
    dialog.setCancelable(false);
    TextView txtDes = (TextView) dialog.findViewById(R.id.txt_des);
    txtDes.setText("و دستگاه شما با موفقت اسکن "
                    + this.virusArray.size()
                    + "فایل مشکوک مشاهده شد. جهت حذف آن ها پاکسازی را لمس کنید");
    TextView txtDelete = (TextView) dialog.findViewById(R.id.txt_delete);
    ProgressBar progressBar = (ProgressBar) dialog.findViewById(R.id.progressBar);
    txtDes.setTypeface(typeface);
    txtDelete.setTypeface(typeface);
    dialog.show();
    txtDelete.setOnClickListener(new C03701());
}
```

شکل ۳۳ نمایش تعداد فایل‌ها به کاربر

۸-۲-۲ اسکن برنامه‌ها پس از نصب

طبق ادعای توسعه دهندگان، با فعال کردن این قسمت، آنتی‌ویروس پس از نصب هر برنامه اندرویدی جدید، به صورت خودکار آن را بررسی خواهد کرد. تحلیل کد مربوط به این قسمت نشان می‌دهد که هیچ‌گونه بررسی پس از نصب انجام نمی‌شود و صرفا پیامی مبنی بر امن بودن برنامه به صورت نوتیفیکیشن نمایش داده می‌شود. کد زیر کد receiver است که پس از نصب برنامه جدید فراخوانی می‌شود و همان‌طور که مشخص است بدون هیچ بررسی فقط پیامی مبنی بر امن بودن برنامه نصب شده داده می‌شود.

```
public class AppInstallReceiver extends BroadcastReceiver {
    private int notifyId = 0;
    private SharedPreferences sharedPreferences;

    public void onReceive(Context context, Intent paramIntent) {
        this.sharedPreferences = C0917G.context.getSharedPreferences("time", 0);
        if (this.sharedPreferences.getBoolean("onTimeFlag", false)) {
            NotificationManager mNotificationManager
                = (NotificationManager) context.getSystemService(NotificationTable.TABLE_NAME);
            Builder mBuilder = new Builder(context);
            mBuilder.setWhen(System.currentTimeMillis())
                .setContentTitle("آنتی ویروس هوشمند")
                .setContentText("برنامه ی نصب شده امن است")
                .setPriority(0).setOngoing(true).setSmallIcon(R.drawable.splash_logo);
            Notification notification = mBuilder.build();
            notification.defaults = 2;
            notification.flags = 16;
            int i = this.notifyId;
            this.notifyId = i + 1;
            mNotificationManager.notify(i, notification);
        }
    }
}
```

شکل ۳۴ بررسی برنامه‌ها پس از نصب

۸-۲-۳ پاکسازی و افزایش سرعت رم

این قسمت حافظه cache مربوط به برنامه‌های اندروید را پاک می‌کند. عملکرد این قسمت از برنامه جعلی نیست و با توجه به کدهایی که بررسی شدند برنامه این کار را انجام می‌دهد. کد زیر مربوط به آزادسازی رم است.

```
private void showRamBoosterDialog() {
    MemoryInfo memInfo = new MemoryInfo();
    ((ActivityManager) getSystemService("activity")).getMemoryInfo(memInfo);
    int percent = (int) ((100.0f * ((float) (getTotalMemory() - memInfo.availMem))) / ((float) getTotalMemory()));
    String totalMem = formatMemSize(getTotalMemory(), 0);
    String availMem = formatMemSize(memInfo.availMem, 0);
    String usedMem = formatMemSize(getTotalMemory() - memInfo.availMem, 0);
    final Dialog dialog = new Dialog(this, R.style.CustomDialog);
    dialog.requestWindowFeature(1);
    dialog.setContentView(R.layout.ram_booster);
    DisplayMetrics dm = new DisplayMetrics();
    getWindowManager().getDefaultDisplay().getMetrics(dm);
    int width = (dm.widthPixels * 90) / 100;
    LayoutParams lp = new LayoutParams();
    lp.copyFrom(dialog.getWindow().getAttributes());
    lp.width = width;
    lp.height = -2;
    lp.gravity = 17;
    dialog.getWindow().setAttributes(lp);
    TextView txtTotalRam = (TextView) dialog.findViewById(R.id.txt_ram_total);
    TextView txtFreeRam = (TextView) dialog.findViewById(R.id.txt_ram_free);
    TextView txtUsedRam = (TextView) dialog.findViewById(R.id.txt_ram_used);
    TextView txtDesc = (TextView) dialog.findViewById(R.id.txt_ram_desc);
    TextView txtRamClean = (TextView) dialog.findViewById(R.id.txt_ram_clean);
    txtRamClean.setTypeface(this.iranBoldTypeFace);
    final ArcProgress arcProgress = (ArcProgress) dialog.findViewById(R.id.arc_progress);
    txtTotalRam.setText(totalMem);
    txtFreeRam.setText(availMem);
    txtUsedRam.setText(usedMem);
    txtDesc.setText(percent + "درصد از فضای رم اشغال شده است، برای افزایش سرعت دستگاه آزادسازی را لمس فرمایید ");
    dialog.show();
```

```
public void onBackPressed() {
    if (isPayed) {
        if (this.back_pressed + 2000 > System.currentTimeMillis()) {
            super.onBackPressed();
        } else {
            Snackbar.make(this.rootLayout, (CharSequence) "برای خروج دوبار ضربه بزن", 0).show();
        }
        this.back_pressed = System.currentTimeMillis();
        return;
    }
    showExitDialog();
}
```

شکل ۳۵ آزادسازی رم

در اینجا تابع memboost برای آزادسازی رم صدا زده می‌شود که قسمتی از کد آن به صورت زیر است و اقدام به متوقف کردن پردازه‌ها و آزادسازی رم می‌کند.

```
int j2 = random.nextInt(((il * 15) - il) + 1) + il;
Method[] amethod = pm.getClass().getDeclaredMethods();
if (mLength > 0) {
    Method method = amethod[0];
    if (!method.getName().equals("freeStorage")) {
        try {
            Method method2 = method;
            PackageManager packageManager = pm;
            method2.invoke(packageManager
                , new Object[]{Long.valueOf(0), Integer.valueOf(0)});
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
ActivityManager acm = (ActivityManager) getSystemService("activity");
MemoryInfo memInfo = new MemoryInfo();
acm.getMemoryInfo(memInfo);
long beforeMemory = memInfo.availMem;
List<RunningAppProcessInfo> taskList = acm.getRunningAppProcesses();
int before = taskList.size();
for (i = 0; i < taskList.size(); i++) {
Log.v("process: " + i, ((RunningAppProcessInfo) taskList.get(i))
.processName + " pid: " + ((RunningAppProcessInfo) taskList.get(i)).pid
+ " importance: " + ((RunningAppProcessInfo) taskList.get(i)).importance
+ " reason: " + ((RunningAppProcessInfo) taskList.get(i)).importanceReasonCode);
}
for (i = 0; i < taskList.size(); i++) {
    RunningAppProcessInfo process = (RunningAppProcessInfo) taskList.get(i);
    int importance = process.importance;
    int pid = process.pid;
    String pname = process.processName;
    if (!(pname.equals("com.raihanbd.easyrambooster")
        || pname.equals("android") || pname.equals("com.android.bluetooth")
    || pname.equals("android.process.acore") || pname.equals("system")
    || pname.equals("com.android.phone") || pname.equals("com.android.systemui")
    || pname.equals("com.android.launcher"))) {
        for (int count = 0; count < 3; count++) {
            acm.killBackgroundProcesses(((RunningAppProcessInfo) taskList.get(i)).processName);
        }
    }
}
```

شکل ۳۶ متوقف کردن پردازه‌ها برای آزادسازی رم

۳-۸ نتیجه‌گیری

این آنتی‌ویروس آزادسازی رم را انجام می‌دهد ولی در مورد بررسی برنامه‌های اندرویدی نصب شده عملا هیچ کاری انجام نداده و هیچ بررسی روی آن‌ها انجام نمی‌دهد، عملکرد اسکن برنامه‌های اندرویدی پس از نصب کاملا جعلی است و بدون هیچ بررسی پیام امن بودن برنامه نصب شده نمایش داده می‌شود. این آنتی‌ویروس از این نظر پوچ‌افزار محسوب می‌شود. اسکن دیگری هم که روی فایل‌های موجود روی حافظه دستگاه انجام می‌شود، فقط بر اساس فرمت فایل‌ها است و ارزشی ندارد. متاسفانه تعداد نصب این دسته از آنتی‌ویروس‌ها بسیار زیاد است و حداقل نیم میلیون نفر یکی از برنامه‌های این دسته را نصب کرده‌اند.

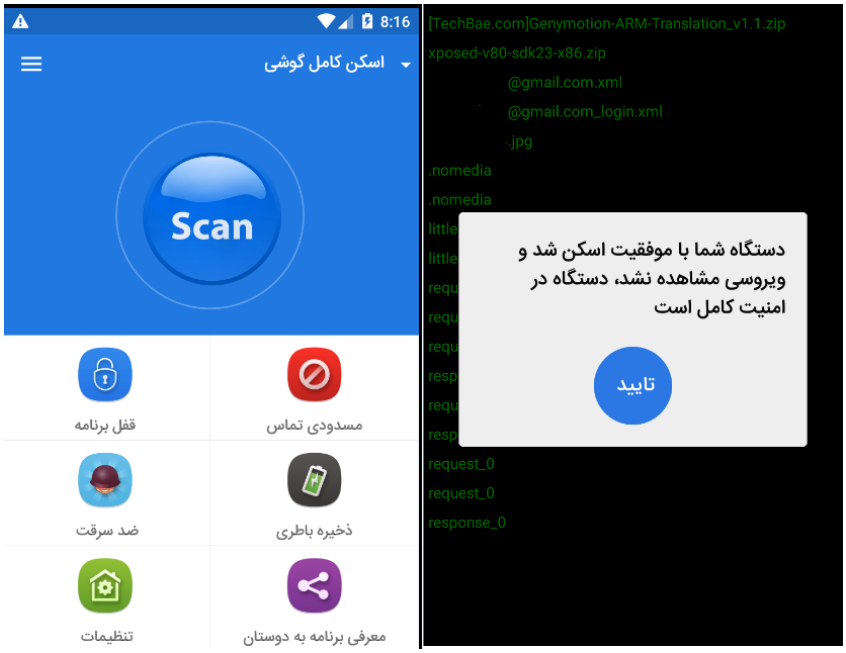
۹ بررسی دسته ششم آنتی‌ویروس‌ها

با بررسی ظاهر، عملکرد و کد برنامه‌ها می‌توان متوجه شباهت کامل برنامه‌های زیر شد:

آیکون	نام برنامه	نام بسته	توسعه‌دهنده	تعداد نصب	صفحه دانلود در مارکت
	آنتی‌ویروس همه‌کاره	com.vahid.anitvirus_hamekare	**********	+۵۰۰۰۰	**********
	آنتی‌ویروس همه‌کاره ۲۰۱۸	com.antivirus.gentleteam	**********	+۱۰۰۰۰	**********
	آنتی‌ویروس پکیج امنیتی حرفه‌ای	com.appline.security_package	**********	-۱۰۰	**********

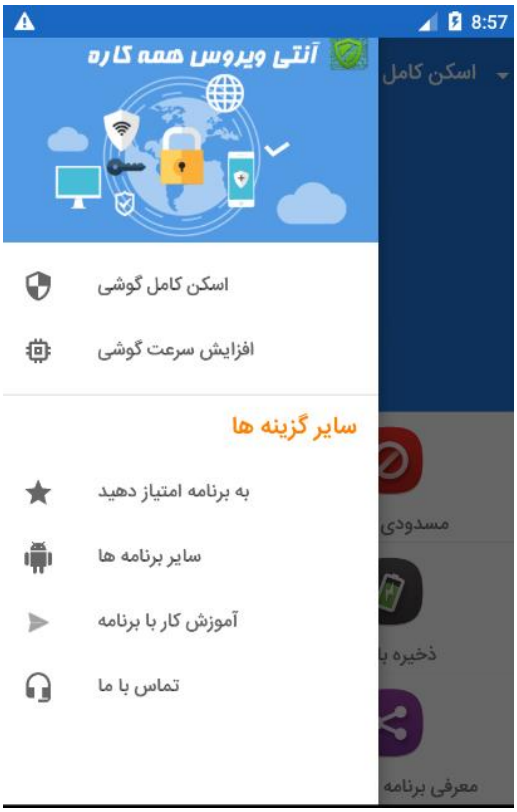
۹-۱-۱ محیط برنامه

با ورود به برنامه، مشاهده می‌شود که آنتی‌ویروس ظاهر ساده‌ای دارد. یک دکمه Scan در بالا که ظاهرا فایل‌های روی دستگاه را اسکن می‌کند (مطابق شکل) و چهار امکان دیگر شامل مسدودی تماس، قفل برنامه، ضد سرقت و ذخیره باطری هم در پایین صفحه وجود دارد.



شکل ۳۷ اسکن کردن فایل‌ها

منو کناری برنامه نیز شامل گزینه‌های اسکن کامل گوشی و افزایش سرعت گوشی است. البته اسکن کامل گوشی همان عملکردی است که در صفحه اصلی برنامه وجود دارد و با فشردن دکمه Scan انجام می‌شود.



شکل ۳۸ منو کناری برنامه

تصویر زیر نیز مربوط به افزایش سرعت گوشی است.

شکل ۳۹ افزایش سرعت گوشی

۹-۲ تحلیل برنامه

در قسمت قبل عملکردهای این آنتی‌ویروس معرفی شد در ادامه به تحلیل هر یک از این امکانات می‌پردازیم

۹-۲-۱ اسکن

اسکن فایل‌ها و برنامه‌ها به منظور یافتن ویروس و برنامه‌های مشکوک انجام می‌شود. با فشردن دکمه scan در صفحه اصلی آنتی‌ویروس این کار انجام می‌شود. تحلیل کد برنامه نشان می‌دهد که این آنتی‌ویروس‌ها صرفا در صورتی که فرمت یک فایل، یکی از انواع زیر باشد آن را به عنوان مورد مشکوک در نظر گرفته (بدون لحاظ کردن محتوا) و با توجه به خواست کاربر حذف می‌کند:

- .exe
- .elf
- .app
- .exp
- .tmp
- .temp
- .cmd
- .ime
- .bin
- .bis
- .ini
- .chm

- .ipa
- .mex
- .msi
- .prg

قطعه کدهای زیر مربوط به تشخیص چنین فایل‌هایی به عنوان فایل مشکوک است و سپس نمایش دادن تعداد آن‌ها به کاربر است.

```
class getFileListTask extends AsyncTask {
    getFileListTask() {
    }

    protected String doInBackground(Object[] params) {
        List<File> files = FilesListActivity.this.storage.getNestedFiles("/");
        Log.i("myLog", "doInBackground: " + files.size() + "");
        int i = 0;
        while (i < files.size()) {
            FilesListActivity.fileArray.add(((File) files.get(i)).getName().toString());
            if (((File) files.get(i)).getName().toString().endsWith(".exe")
                || ((File) files.get(i)).getName().toString().endsWith(".elf")
                || ((File) files.get(i)).getName().toString().endsWith(".app")
                || ((File) files.get(i)).getName().toString().endsWith(".exp")
                || ((File) files.get(i)).getName().toString().endsWith(".tmp")
                || ((File) files.get(i)).getName().toString().endsWith(".temp")
                || ((File) files.get(i)).getName().toString().endsWith(".cmd")
                || ((File) files.get(i)).getName().toString().endsWith(".ime")
                || ((File) files.get(i)).getName().toString().endsWith(".bin")
                || ((File) files.get(i)).getName().toString().endsWith(".bis")
                || ((File) files.get(i)).getName().toString().endsWith(".ini")
                || ((File) files.get(i)).getName().toString().endsWith(".chm")
                || ((File) files.get(i)).getName().toString().endsWith(".ipa")
                || ((File) files.get(i)).getName().toString().endsWith(".mex")
                || ((File) files.get(i)).getName().toString().endsWith(".msi")
                || ((File) files.get(i)).getName().toString().endsWith(".prg")) {

                    FilesListActivity.this.virusArray.add(((File) files.get(i)).toString());
            }
            i++;
        }
        return "";
    }
```

شکل ۴۰ تشخیص فایل‌ها با فرمت‌های خاص به عنوان فایل مشکوک

```
private void showDialog() {
    Typeface typeface = Typeface.createFromAsset(getAssets(), "iran_sans.ttf");
    Dialog dialog = new Dialog(this, R.style.CustomDialog);
    dialog.requestWindowFeature(1);
    dialog.setContentView(R.layout.final_delete_dialog);
    dialog.setCancelable(false);
    TextView txtDes = (TextView) dialog.findViewById(R.id.txt_des);
    txtDes.setText("دستگاه شما با موفقت اسکن و "
                + this.virusArray.size()
                + "فایل مشکوک مشاهده شد. جهت حذف آن ها پاکسازی را لمس کنید");
    TextView txtDelete = (TextView) dialog.findViewById(R.id.txt_delete);
    ProgressBar progressBar = (ProgressBar) dialog.findViewById(R.id.progressBar);
    txtDes.setTypeface(typeface);
    txtDelete.setTypeface(typeface);
    dialog.show();
    txtDelete.setOnClickListener(new C03701());
}
```

شکل ۴۱ نمایش تعداد فایل‌ها به کاربر

لازم به ذکر است که این عملکرد برای اسکن فایل‌ها کاملا مشابه عملکرد اسکن مربوط به دسته پنجم آنتی‌ویروس‌ها است. ظاهرا این عملکرد بی‌فایده در مورد اسکن فایل‌ها توسط هر دو دسته از آنتی‌ویروس‌ها از یک منبع باز یکسان کپی شده‌اند.

۹-۲-۲ افزایش سرعت گوشی

این قسمت حافظه cache مربوط به برنامه‌های اندروید را پاک می‌کند. عملکرد این قسمت از برنامه جعلی نیست و با توجه به کدهایی که بررسی شدند برنامه این کار را انجام می‌دهد. کد زیر مربوط به آزادسازی رم است.

```
private void showRamBoosterDialog() {
    MemoryInfo memInfo = new MemoryInfo();
    ((ActivityManager) getSystemService("activity")).getMemoryInfo(memInfo);
    int percent = (int) ((100.0f * ((float) (getTotalMemory() - memInfo.availMem))) / ((float) getTotalMemory()));
    String totalMem = formatMemSize(getTotalMemory(), 0);
    String availMem = formatMemSize(memInfo.availMem, 0);
    String usedMem = formatMemSize(getTotalMemory() - memInfo.availMem, 0);
    final Dialog dialog = new Dialog(this, R.style.CustomDialog);
    dialog.requestWindowFeature(1);
    dialog.setContentView(R.layout.ram_booster);
    DisplayMetrics dm = new DisplayMetrics();
    getWindowManager().getDefaultDisplay().getMetrics(dm);
    int width = (dm.widthPixels * 90) / 100;
    LayoutParams lp = new LayoutParams();
    lp.copyFrom(dialog.getWindow().getAttributes());
    lp.width = width;
    lp.height = -2;
    lp.gravity = 17;
    dialog.getWindow().setAttributes(lp);
    TextView txtTotalRam = (TextView) dialog.findViewById(R.id.txt_ram_total);
    TextView txtFreeRam = (TextView) dialog.findViewById(R.id.txt_ram_free);
    TextView txtUsedRam = (TextView) dialog.findViewById(R.id.txt_ram_used);
    TextView txtDesc = (TextView) dialog.findViewById(R.id.txt_ram_desc);
    TextView txtRamClean = (TextView) dialog.findViewById(R.id.txt_ram_clean);
    txtRamClean.setTypeface(this.iranBoldTypeFace);
    final ArcProgress arcProgress = (ArcProgress) dialog.findViewById(R.id.arc_progress);
    txtTotalRam.setText(totalMem);
    txtFreeRam.setText(availMem);
    txtUsedRam.setText(usedMem);
    txtDesc.setText(percent + "درصد از فضای رم اشغال شده است، برای افزایش سرعت دستگاه آزادسازی را لمس فرمایید");
    dialog.show();
```

```
public void onBackPressed() {
    if (isPayed) {
        if (this.back_pressed + 2000 > System.currentTimeMillis()) {
            super.onBackPressed();
        } else {
            Snackbar.make(this.rootLayout, (CharSequence) "برای خروج دوبار ضربه بزن", 0).show();
        }
        this.back_pressed = System.currentTimeMillis();
        return;
    }
    showExitDialog();
}
```

شکل ۴۲ آزادسازی رم

در اینجا تابع memboost برای آزادسازی رم صدا زده می‌شود که قسمتی از کد آن به صورت زیر است و اقدام به متوقف کردن پردازه‌ها و آزادسازی رم می‌کند.

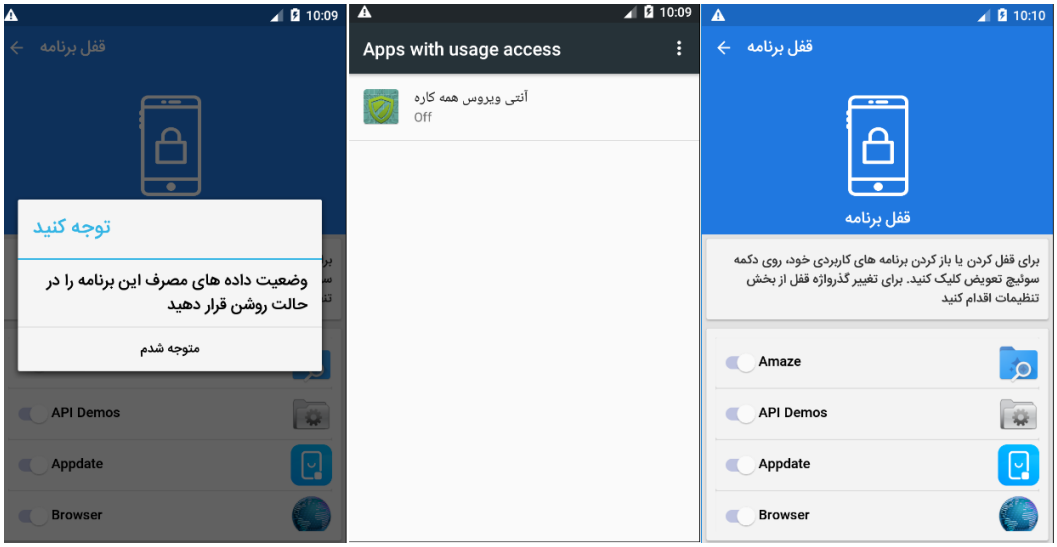
لازم به ذکر است که این ویژگی این دسته از آنتی‌ویروس‌ها نیز کاملا مشابه همین ویژگی در دسته پنجم آنتی‌ویروس‌ها است و در هر دو دسته از منبع یکسانی کپی کرده‌اند.

```
int j2 = random.nextInt(((i1 * 15) - i1) + 1) + i1;
Method[] amethod = pm.getClass().getDeclaredMethods();
if (mLength > 0) {
    Method method = amethod[0];
    if (!method.getName().equals("freeStorage")) {
        try {
            Method method2 = method;
            PackageManager packageManager = pm;
            method2.invoke(packageManager
                , new Object[]{Long.valueOf(0), Integer.valueOf(0)});
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
ActivityManager acm = (ActivityManager) getSystemService("activity");
MemoryInfo memInfo = new MemoryInfo();
acm.getMemoryInfo(memInfo);
long beforeMemory = memInfo.availMem;
List<RunningAppProcessInfo> taskList = acm.getRunningAppProcesses();
int before = taskList.size();
for (i = 0; i < taskList.size(); i++) {
Log.v("process: " + i, ((RunningAppProcessInfo) taskList.get(i))
.processName + " pid: " + ((RunningAppProcessInfo) taskList.get(i)).pid
+ " importance: " + ((RunningAppProcessInfo) taskList.get(i)).importance
+ " reason: " + ((RunningAppProcessInfo) taskList.get(i)).importanceReasonCode);
}
for (i = 0; i < taskList.size(); i++) {
    RunningAppProcessInfo process = (RunningAppProcessInfo) taskList.get(i);
    int importance = process.importance;
    int pid = process.pid;
    String pname = process.processName;
    if (!(pname.equals("com.raihanbd.easyrambooster")
        || pname.equals("android") || pname.equals("com.android.bluetooth")
    || pname.equals("android.process.acore") || pname.equals("system")
    || pname.equals("com.android.phone") || pname.equals("com.android.systemui")
    || pname.equals("com.android.launcher"))) {
        for (int count = 0; count < 3; count++) {
            acm.killBackgroundProcesses(((RunningAppProcessInfo) taskList.get(i)).processName);
        }
    }
}
```

شکل ۴۳ متوقف کردن پردازه‌ها برای آزادسازی رم

۳-۲-۹ قفل برنامه

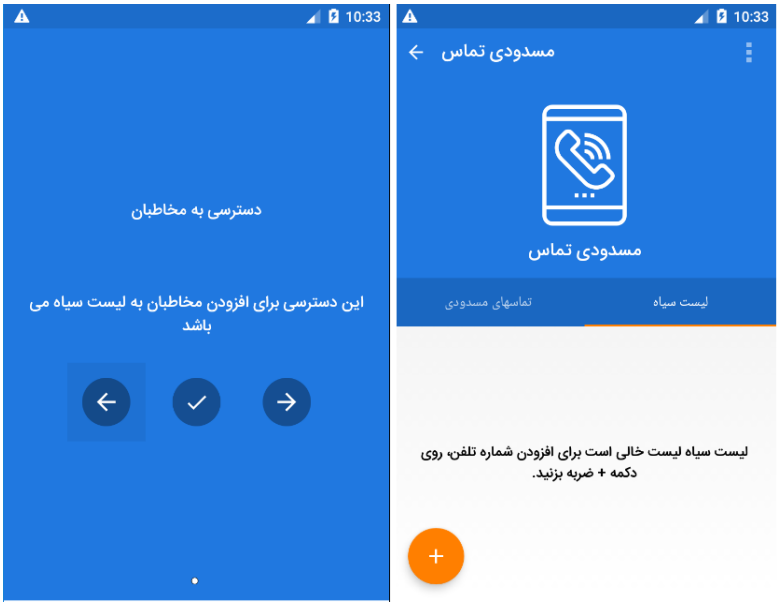
با استفاده از این ویژگی آنتی‌ویروس، می‌توان برای برنامه‌های دیگر قفل گذاشت تا بعدا برای اجرا کردن آن‌ها نیاز به وارد کردن رمز باشد. برای اجرای این ویژگی لازم است که به آنتی‌ویروس دسترسی usage access داده شود. تست این ویژگی نشان از عملکرد صحیح آن داشت.



شکل ۴۴ اعطای مجوز usage access و امکان قفل‌کردن برنامه‌ها

۹-۲-۴ مسدودی تماس

این قسمت برای بلاک کردن تماس‌های ورودی است. کاربر می‌تواند برخی افراد را به لیست سیاه اضافه کند تا تماس‌های آن‌ها توسط آنتی‌ویروس مسدود شود.



شکل ۴۵ مسدود کردن تماس‌ها

تکه کد زیر مربوط به این ویژگی است.

```
public class CallReceiver extends BroadcastReceiver {
    private static final String TAG = CallReceiver.class.getSimpleName();

    @DebugLog
    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equals("android.intent.action.PHONE_STATE")) {
            if (TelephonyManager.EXTRA_STATE_RINGING.equals(intent.getStringExtra("state"))
                && ((AppContext) context.getApplicationContext())
                        .getAppPreferences().isBlockingEnable()) {
                String incomingNumber = intent.getStringExtra("incoming_number");
                Log.i(TAG, "Incoming number: {" + incomingNumber + "}");
                Intent i = new Intent(context, CallBlockingService.class);
                i.putExtra(CallBlockingService.INCOMING_NUMBER, incomingNumber);
                context.startService(i);
            }
        }
    }
}
```

شکل ۴۶ بررسی تماس‌های ورودی برای بلاک کردن

۹-۲-۵ ضد سرقت

با این ویژگی می‌توان یک شماره پشتیبان تعریف کرد که در صورت دریافت برخی پیام‌های خاص از آن شماره برخی کنترل‌ها را می‌توان روی دستگاه داشت، این کنترل‌ها به صورت زیر است:

- **اتصال سیم‌کارت**: اگر این سرویس فعال باشد، هنگام تعویض سیم‌کارت پیامی به شماره پشتیبان ثبت شده ارسال خواهد شد
- **آلارم هشدار**: با ارسال کد #*alarm*# به گوشی، آلارم هشدار روشن خواهد شد
- **قفل گوشی از راه دور**: برای قفل کردن صفحه لازم است کد زیر به گوشی پیامک شود

رمز ورود #*lock*#

- **پاک کردن اطلاعات گوشی از راه دور**: برای پاک کردن اطلاعات کارت حافظه کافی است کد #*del*# به گوشی پیامک شود

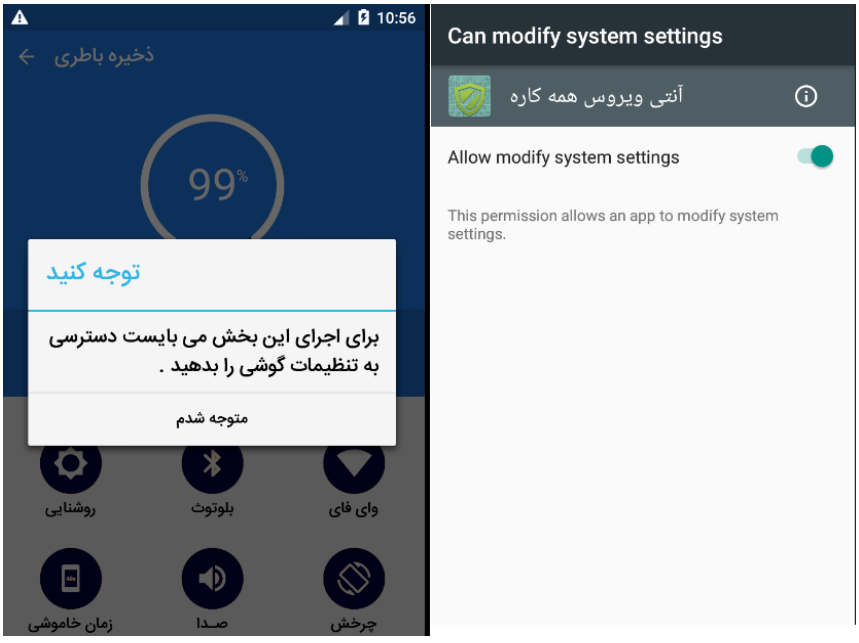
البته برای اجرای دو مورد آخر باید قابلیت administrator برای این آنتی‌ویروس فعال شود.

۹-۲-۶ ذخیره باطری

این ویژگی صرفا امکاناتی که باطری مصرف می‌کنند و خاموش شدن آن‌ها منجر به صرفه‌جویی در مصرف باطری می‌شود را دارد. این موارد در شکل زیر آمده است.

شکل ۴۷ذخیره باطری

البته برای فعال‌سازی این امکان لازم است که آنتی‌ویروس دسترسی تنظیمات گوشی داده شود



شکل ۴۸ اعطای دسترسی به تنظیمات گوشی

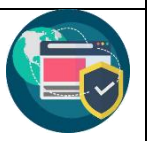
کد مربوط به این قسمت در فایل BatterySaver.java در پکیج com.vahid.anitvirus_hamekare.Battery قرار دارد.

۳-۹ نتیجه‌گیری

این آنتی‌ویروس آزادسازی رم را انجام می‌دهد و ویژگی‌های ضدسرقت، قفل برنامه‌ها، ذخیره باطری و مسدودی تماس نیز در آن چهارچوبی که توضیح داده شده است انجام می‌شود ولی متاسفانه ویژگی اصلی یک آنتی‌ویروس یعنی اسکن کردن برنامه‌های نصب شده را کلاً انجام نمی‌دهد، اسکن آن روی فایل‌های موجود روی حافظه دستگاه انجام می‌شود و فقط بر اساس فرمت فایل‌ها است و ارزشی ندارد. این آنتی‌ویروس از این نظر پوچ‌افزار محسوب می‌شود. قسمت مربوط به بررسی ویروس‌های این دسته از آنتی‌ویروس‌ها کاملاً مشابه دسته پنجم است و از روی یک کد یکسان ساخته شده‌اند ولی این دسته امکانات دیگری هم دارد که در گزارش بررسی شدند.

۱۰ بررسی دسته هفتم آنتی‌ویروس‌ها

با بررسی ظاهر، عملکرد و کد برنامه‌ها می‌توان متوجه شباهت کامل برنامه‌های زیر شد:

آیکون	نام برنامه	نام بسته	توسعه‌دهنده	تعداد نصب	صفحه دانلود در مارکت
	آنتی‌ویروس هوشمند ۲۰۱۷ + (۲۰۱۸)	com.developer.antiscan_v1	**********	+۱۰۰۰۰	**********
	آنتی‌ویروس فوق پیشرفته	com.sabnam.app	**********	+۱۰۰۰۰	**********
	آنتی‌ویروس فوق پیشرفته(نسخه همراه)	com.taher.antivirus.mobilesecurity	**********	+۵۰۰۰	**********
	آنتی‌ویروس هوشمند	com.antivirus.jiro.nikparvar2	**********	+۲۰۰۰	**********
	آنتی‌ویروس هوشمند ۲۰۱۸	com.amitis.antivirus.security	**********	+۲۰۰۰	**********
	آنتی‌ویروس همه کاره آوین	com.avin.antivirus.mobilesecurity	**********	+۲۰۰۰	**********
	آنتی‌ویروس فوق حرفه‌ای ۲۰۱۸	com.appdirac.antivirus.mobilesecurity	**********	+۱۰۰۰	**********
	آنتی‌ویروس پیشرفته(ایرانی) محصول ۲۰۱۸	ir.nbnb.antivirus.newmobilesecurity	**********	+۱۰۰۰	**********
	آنتی‌ویروس هوشمند ۲۰۱۷	com.antivirous.app	**********	+۵۰۰	**********
	آنتی‌ویروس اسکای	com.example.ehsans11.merikh_antivirus	**********	+۱۰۰	**********

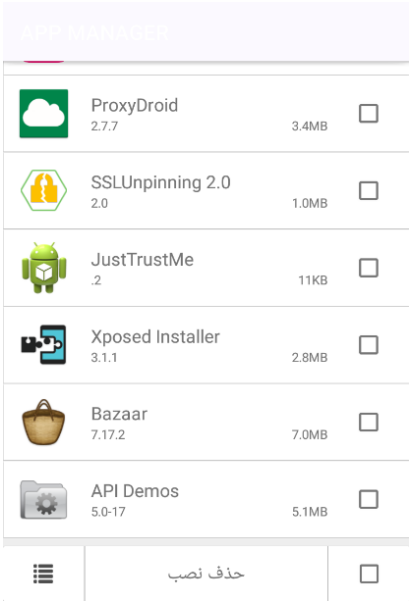
	آنتی‌ویروس کاسپر	com.example.ehsans11.antivirus	**********	+۱۰۰	********** **********
	آنتی‌ویروس هوشمند	ir.zback.antivirus	**********	-۱۰۰	********** **********
	گروه نرم افزاری صدرا	ir.sadra.antivirus	**********	-۱۰۰	********** **********

۱۰-۱ محیط برنامه

محیط اولیه ورود به برنامه مطابق شکل ۴۹ است.

شکل ۴۹ – محیط ورود به برنامه

در قسمت "مدیر برنامه" کاربر می‌تواند لیستی از برنامه‌های نصب‌شده روی گوشی مشاهده کند و اقدام به حذف هرکدام به دلخواه بکند (شکل ۵۰).

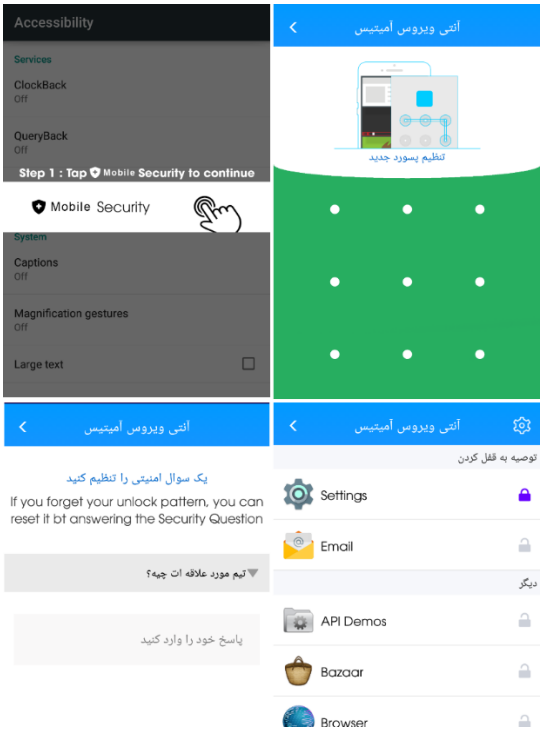


شکل ۵۰ – مدیر برنامه‌ها

در قسمت "مانیتور" نیز کاربر می‌تواند اطلاعات لحظه‌ای در مورد میزان استفاده از پردازنده و رَم گوشی مشاهده کند (شکل ۵۱).

شکل ۵۱ – مانیتور

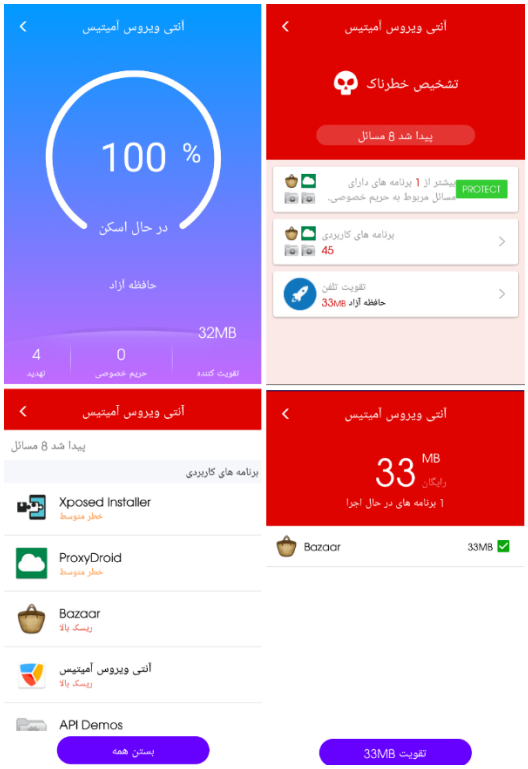
در قسمت "قفل برنامه" نیز کاربر می‌تواند با تنظیم یک الگو به عنوان پسورد، روی برنامه‌های نصب‌شده، قفل تنظیم کند. برای انجام این کار برنامه ابتدا نیاز به دسترسی به سرویس Accessibility دارد که باید توسط کاربر از طریق قسمت تنظیمات به برنامه اجازه داده شود از این رو برنامه قسمت تنظیمات را به کاربر نشان می‌دهد. سپس برنامه از کاربر یک الگو و یک سوال امنیتی (برای بازیابی الگو در صورت فراموشی کاربر) دریافت می‌کند (شکل ۵۲).



شکل ۵۲ – قفل برنامه‌ها

در قسمت "اسکن برنامه‌ها"، برنامه‌های نصب‌شده مورد بررسی قرار می‌گیرند و موارد زیر به کاربر گزارش داده می‌شود:

- برنامه‌هایی که نیاز به محافظت از طریق قفل برنامه‌ها دارند
- برنامه‌هایی که بر اساس دسترسی‌ها، برنامه‌هایی با ریسک بالا محسوب می‌شوند
- بستن برنامه‌های پس‌زمینه‌ای در حال اجرا (به منظور آزادسازی حافظه و صرفه‌جویی در مصرف باتری)



شکل ۵۳

۱۰-۲ تحلیل برنامه

در این قسمت به بررسی سورس‌کد هر یک از بخش‌های برنامه می‌پردازیم.

۱۰-۲-۱ قابلیت مدیر برنامه

کاربر با این قابلیت می‌تواند هر برنامه‌ای را به دلخواه از گوشی حذف کند، کاری که به سادگی از طریق قسمت تنظیمات خود گوشی نیز قابل انجام است پس این قابلیت مزیتی محسوب نمی‌شود.

۱۰-۲-۲ قابلیت مانیتور

کاربر با این استفاده از این قابلیت می‌تواند به صورت لحظه‌ای میزان استفاده از پردازنده و حافظه گوشی را مشاهده کند. این قابلیت نیز مزیتی خاصی برای یک آنتی‌ویروس محسوب نمی‌شود.

۱۰-۲-۳ قابلیت قفل برنامه‌ها

کاربر با استفاده از این قابلیت می‌تواند بر روی برنامه‌های نصب‌شده روی گوشی قفل قرار بدهد. برنامه برای این کار نیاز به دسترسی Accessibility برای رسم روی سایر برنامه‌ها دارد. همچنین برنامه برای بدست‌آوردن اسم برنامه‌ای که در حال حاضر با کاربر در حال تعامل است، از روش‌های مختلفی استفاده می‌کند که یکی از آن‌ها استفاده از دسترسی Usage Stats است (شکل ۵۴).

```
public void run() {
    if (VERSION.SDK_INT >= 22) {
        String topPackageName = null;
        UsageStatsManager mUsageStatsManager = (UsageStatsManager) MonitorShieldService.this.getSystemService("usagestats");
        long time = System.currentTimeMillis();
        List<UsageStats> stats = mUsageStatsManager.queryUsageStats(0, time - 10000, time);
        if (stats != null) {
            SortedMap<Long, UsageStats> mySortedMap = new TreeMap();
            for (UsageStats usageStats : stats) {
                mySortedMap.put(Long.valueOf(usageStats.getLastTimeUsed()), usageStats);
            }
            if (!mySortedMap.isEmpty()) {
                topPackageName = ((UsageStats) mySortedMap.get(mySortedMap.lastKey())).getPackageName();
            }
        }
        if (this.listener != null && topPackageName != null && !MonitorShieldService.this.isBoosterRunning) {
            this.listener.onActivityStarting(topPackageName);
            return;
        }
        return;
    }
    ActivityManager am = (ActivityManager) MonitorShieldService.this.getBaseContext().getSystemService("activity");
    PackageManager pm = MonitorShieldService.this.getBaseContext().getPackageManager();
    String mPackageName = null;
    if (VERSION.SDK_INT > 20) {
        List<RunningAppProcessInfo> appProcesses = am.getRunningAppProcesses();
        if (appProcesses.size() >= 0) {
            RunningAppProcessInfo appProcess = (RunningAppProcessInfo) appProcesses.get(0);
            if (appProcess.importance == 100) {
                mPackageName = appProcess.processName;
```

شکل ۵۴

همچنین اگر کاربر سه بار الگوی ورود به برنامه قفل‌شده را اشتباه وارد کند، برنامه با استفاده از دوربین جلویی از کاربر عکس می‌گیرد (شکل ۵۵).

```
this.countFailed++;
if (this.countFailed == 3 && this.sharedPreferences.getBoolean(AppLockSettingsActivity.
    new Selfie(this, getPackageName()).takePhoto();
}
```

شکل ۵۵

اگرچه قابلیت قفل برنامه‌ها، در برنامه‌های مشابه مانند "فضول‌یاب" وجود دارد، اما وجود چنین قابلیتی در یک آنتی‌ویروس می‌تواند به حفظ امنیت کاربران کمک کند.

۱۰-۲-۴ اسکن برنامه‌ها

در این قسمت، برنامه سه مورد را به کاربر گزارش می‌دهد: برنامه‌هایی که نیاز به محافظت دارند، برنامه‌هایی که دسترسی‌های حساس دارند و برنامه‌های پس‌زمینه. در ادامه به بررسی هر یک از این موارد پرداخته خواهد شد.

۱۰-۲-۴-۱ برنامه‌های حساس از نظر حریم شخصی

در این قسمت، برنامه به کاربر پیشنهاد می‌کند که هرکدام از برنامه‌های زیر را (در صورت نصب‌بودن) با استفاده از قفل‌برنامه، محافظت کند (شکل ۵۶):

- com.android.settings
- com.android.email
- com.android.chrome
- com.google.android.youtube

- com.facebook.katana
- com.facebook.orca
- com.google.android.gm
- com.instagram.android
- com.twitter.android
- com.snapchat.android
- com.whatsapp
- com.zing.zalo
- com.facebook.lite
- com.viber.voip
- com.skype.raider
- com.dropbox.android
- com.google.android.apps.docs
- com.google.android.apps.photos

```
public static boolean isRecommendAppLock(String packageName) {
    for (String recommendApp : new String[]{"com.android.settings", "com.android.email", "com.a
        if (recommendApp.equals(packageName)) {
            return true;
        }
    }
    return false;
}
```

شکل ۵۶

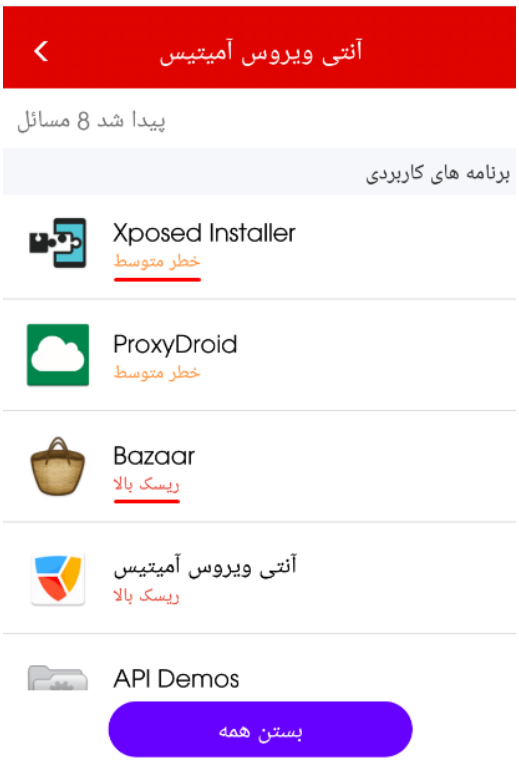
۲-۴-۲-۱۰ برنامه‌هایی با دسترسی‌های حساس

در پوشه assets برنامه، فایلی با نام permissions.json وجود دارد که برنامه بر اساس آن، دسترسی‌های یک برنامه را بررسی می‌کند. در این فایل، دسترسی‌ای با خصیصه dangerous: “1” به منزله خطرناک بودن آن دسترسی است (شکل ۵۷).

```
{
  "permissionName": "android.permission.ACCESS_FINE_LOCATION",
  "dangerous": 1
},
{
  "permissionName": "com.android.browser.permission.READ_HISTORY_BOOKMARKS",
  "dangerous": 0
},
{
  "permissionName": "com.android.browser.permission.WRITE_HISTORY_BOOKMARKS",
  "dangerous": 0
},
{
  "permissionName": "android.permission.SEND_SMS",
  "dangerous": 1
},
{
  "permissionName": "android.permission.READ_SMS",
  "dangerous": 0
},
{
  "permissionName": "android.permission.READ_PHONE_STATE",
  "dangerous": 1
},
```


شکل ۵۷ – فایل permissions.json

در صورتی که برنامه‌ای دسترسی‌ای از لیست دسترسی‌های خطرناک داشته باشد، برنامه آن را با عنوان "ریسک بالا" به کاربر معرفی می‌کند و در صورتی که برنامه مورد بررسی دسترسی‌ای با خصیصه dangerous: “0” داشته باشد، آن را به عنوان "خطر متوسط" معرفی می‌کند (شکل ۵۸).



شکل ۵۸

همچنین برنامه، در صورتی که برنامه‌ای از طریق گوگل‌پلی نصب‌نشده باشد، در لیست اخطارها آن را به کاربر اعلام می‌کند (شکل ۵۹).

شکل ۵۹

علاوه بر این در پوشه assets، سه فایل دیگر با نام‌های زیر در برنامه وجود دارد:

- blackListActivities.json : در این فایل، لیستی از نام اکتیویتی‌ها وجود دارد و در صورتی که برنامه ای روی گوشی، حاوی اکتیویتی‌ای با چنین نامی باشد، برنامه به کاربر هشدار می‌دهد
- blackListPackages.json: در این فایل، لیستی از پکیج‌ها وجود دارد و در صورتی که برنامه‌ای با همین نام پکیج‌ها روی گوشی نصب شده باشد، برنامه به کاربر اخطار می‌دهد.
- whiteList.json: در این فایل لیستی از نام پکیج‌های سالم وجود دارد و برنامه چنین پکیج‌هایی را سالم درنظر می‌گیرد.

اگرچه برنامه برای تشخیص بدافزارها از یک لیست از پیش‌تعریف‌شده استفاده می‌کند، اما دو نکته بسیار مهم در مورد این لیست‌ها وجود دارد:

- **لیست‌های کوچک:** این لیست‌ها بسیار کوچک هستند در صورتی که روزانه صدها و یا حتی هزاران بدافزار منتشر می‌شود. با وجود چنین حجم گسترده‌ای از بدافزار، چنین لیست کوچکی قادر به تشخیص همه بدافزارها نیست.
- **عدم به‌روزرسانی لیست‌ها:** در هیچ جایی از برنامه لیست‌های برنامه به‌روزرسانی نمی‌شوند از این رو این آنتی‌ویروس قادر نیست هیچ‌یک از بدافزارهای جدید را تشخیص بدهد.

۳-۴-۲-۱۰ برنامه‌های پس‌زمینه

در این قسمت برنامه لیستی از برنامه‌های در حال اجرا در پس‌زمینه (غیرسیستمی) به کاربر نشان می‌دهد که کاربر می‌تواند با بستن آن‌ها، در حافظه گوشی صرفه‌جویی کند. این قابلیت بیشتر مربوط به برنامه‌های کاربردی می‌شود و وجود آن در یک آنتی‌ویروس مزیت محسوب نمی‌شود.

۵-۲-۱۰ تبلیغات موجود در برنامه

برنامه برای ارسال تبلیغات از سرویس پوشه استفاده می‌کند و اقدامات مختلفی انجام می‌دهد. لیست این اقدامات عبارتند از (شکل ۶۰):

- باز کردن یک دیالوگ
- بازکردن یک لینک در تلگرام
- دانلود یک فایل APK
- نصب یک فایل APK روی گوشی کاربر

```
case 1:
    String image_baner = message.getString("baner");
    String image_logo = message.getString("logo");
    String text_title = message.getString("title");
    String text_desc = message.getString("desc");
    String text_btn = message.getString("textbtn");
    intent = new Intent(getApplicationContext(), Dialog.class).putExtra("image_logo", image_logo).putExtra
    intent.addFlags(268435456);
    startActivity(intent);
    return;
case 2:
    Intent browserIntent = new Intent("android.intent.action.VIEW", Uri.parse(message.getString("link")));
    browserIntent.addFlags(268435456);
    startActivity(browserIntent);
    return;
case 3:
    try {
        link = message.getString("link");
        if (appInstalledOrNot("org.telegram.messenger")) {
            telegram(link);
            return;
        } else {
            popup(link);
            return;
        }
    } catch (JSONException e) {
        Log.e("", "Exception in parsing json", e);
        return;
    }
```

شکل ۶۰

۳-۱۰ نتیجه‌گیری

این دسته‌ها از آنتی‌ویروس‌ها، برنامه‌ها را بر اساس دسترسی‌ها و نام پکیج آن‌ها بررسی می‌کند. مکانیزم بررسی برنامه با استفاده دسترسی‌ها تا حدودی قابل قبول است اگرچه اشکالاتی نیز در این قسمت وجود دارد (به عنوان مثال، دسترسی SYSTEM_ALERT_WINDOW یکی از دسترسی‌های خطرناکی است که در لیست

دسترسی‌های این برنامه گنجانده نشده است). از طرف دیگر مکانیزم بررسی برنامه‌ها با توجه به نام پکیج آن‌ها بسیار ساده و ابتدایی است و به راحتی قابل دور زدن هستند. از جمله مشکلات این آنتی‌ویروس‌ها می‌توان به کوچک‌بودن لیست بدافزارها و عدم به‌روزرسانی آن‌ها اشاره کرد، مشکلاتی که بسیار جدی هستند و عملاً حفاظتی نزدیک به صفر را برای کاربر فراهم می‌کنند.

۱۱ نتیجه‌گیری

تحلیل دسته‌های مختلف از آنتی‌ویروس‌های ایرانی منتشر شده در مارکت های نشان می‌دهد که بسیاری از آن‌ها بارها با اسامی مختلف و توسط افراد مختلف منتشر شده‌اند، از این رو می‌توان نتیجه گرفت که این برنامه‌ها از روی برنامه‌های منبع باز(open source) ساخته شده‌اند و صرفا با تغییر نام و آیکون منتشر شده‌اند. در اغلب موارد هدف از این کار استفاده از سرویس‌های تبلیغاتی داخل این برنامه‌ها و درآمدزایی از طریق نمایش تبلیغ به کاربران است. استفاده از سرویس‌ها تبلیغاتی مثل عدد و سرویس ارسال نوتیفیکیشن پوشه (برای ارسال تبلیغات نوتیفیکیشنی) در این برنامه‌ها بسیار رایج است.

همان‌طور که در این گزارش نشان داده شد برخی از آنتی‌ویروس‌ها (دسته سوم در این گزارش) با کد یکسان بیش از ۱۵ بار روی مارکت های ایرانی منتشر شده‌اند. متاسفانه بسیاری از این آنتی‌ویروس‌ها عملکرد درستی برای تشخیص بدافزارهای اندرویدی ندارند و همه هفت دسته‌ای که در اینجا بررسی شدند یا کاملا بدون هیچ تحلیلی هستند و یا تحلیل بسیار ابتدایی دارند که قابل قبول نیست و نمی‌تواند از دستگاه اندرویدی در برابر تهدیدات دفاع کند.

درمجموع این شصت آنتی‌ویروس بیش از یک میلیون نصب داشتند که نشان از گستردگی کار تبلیغاتی و تجاری این برنامه‌ها است.