

PromptSpy ushers in the era of Android threats using GenAI

 wlivesecurity.com/en/eset-research/promptspy-ushers-in-era-android-threats-using-genai

```
public static void startAutomationLoop(AccessibilityService accessibilityService, String str, String str2) {
    String str3;
    String str4;
    JSONArray jsonArray;
    String str5;
    int i;
    int i2;
    boolean z;
    AccessibilityService accessibilityService2 = accessibilityService;
    String str6 = str;
    if (accessibilityService2 == null) {
        return;
    }
    int i3 = 4;
    String str7 = TAG;
    if (str2 == null || str2.isEmpty()) {
        ServiceInteractionUtil.ToLog(TAG, "未设置 Gemini API Key, 无法执行自动化任务", 4);
        return;
    }
    ServiceInteractionUtil.ToLog(TAG, "开始执行任务: " + str6);
    JSONArray jsonArray2 = new JSONArray();
    String string = accessibilityService2.getString(R.string.atuo_load_msg);
    int i4 = 1;
    boolean z2 = true;
    int i5 = 0;
    while (z2 && i5 < 30) {
        int i6 = i5 + 1;
        ServiceInteractionUtil.ToLog(str7, ">>> 步骤 " + i6);
        AccessibilityNodeInfo accessibilityNodeInfoGetActiveWindowNodeInfo = InputService.GetActiveWindowNodeInfo();
        String aIXml = AccessibilityNodeUtil.toAIXml(accessibilityNodeInfoGetActiveWindowNodeInfo);
        if (accessibilityNodeInfoGetActiveWindowNodeInfo != null) {
            accessibilityNodeInfoGetActiveWindowNodeInfo.recycle();
        }
        if (i6 == i4) {
            str3 = "You are an Android automation assistant. The user will give you the UI XML data of the current screen."
        } else {
            str3 = "The previous action has been executed. This is the new UI XML, please determine if the task is complet
        }
        addToHistory(jsonArray2, "user", str3);
        String strCallGeminiApi = callGeminiApi(str2, jsonArray2);
        if (strCallGeminiApi == null) {
            ServiceInteractionUtil.ToLog(str7, "API 请求失败, 终止任务", i3);
            AutoClicker.bringHomeToForeground();
            return;
        }
    }
}
```

ESET researchers uncovered the first known case of Android malware abusing generative AI for context-aware user interface manipulation. While machine learning has been used to similar ends already – just recently, researchers at Dr.WEB found [Android.Phantom](#), which uses TensorFlow machine learning models to analyze advertisement screenshots and automatically click on detected elements for large scale ad fraud – this is the first time we have seen generative AI deployed in this manner. Because the attackers rely on prompting an AI model (in this instance, Google’s Gemini) to guide malicious UI manipulation, we have named this family PromptSpy. This is the second AI powered malware we have discovered – following [PromptLock](#) in August 2025, the first known case of AI-driven ransomware.

While generative AI is deployed only in a relatively minor part of PromptSpy’s code – that responsible for achieving persistence – it still has a significant impact on the malware’s adaptability. Specifically, Gemini is used to analyze the current screen and provide PromptSpy

with step-by-step instructions on how to ensure the malicious app remains pinned in the recent apps list, thus preventing it from being easily swiped away or killed by the system. The AI model and prompt are predefined in the code and cannot be changed. Since Android malware often relies on UI navigation, leveraging generative AI enables the threat actors to adapt to more or less any device, layout, or OS version, which can greatly expand the pool of potential victims.

The main purpose of PromptSpy is to deploy a built-in VNC module, giving operators remote access to the victim's device. This Android malware also abuses the Accessibility Service to block uninstallation with invisible overlays, captures lockscreen data, records video. It communicates with its C&C server via the VNC protocol, using AES encryption.

Based on language localization clues and the distribution vectors observed during analysis, this campaign appears to be financially motivated and seems to primarily target users in Argentina. Interestingly, analyzed PromptSpy samples suggest that it was developed in a Chinese-speaking environment.

PromptSpy is distributed by a dedicated website and has never been available on Google Play. As an App Defense Alliance partner, we nevertheless shared our findings with Google. Android users are automatically protected against known versions of this malware by [Google Play Protect](#), which is enabled by default on Android devices with Google Play Services.

Key points of this blogpost:

- PromptSpy is the first known Android malware to use generative AI in its execution flow, even though it's only to achieve persistence.
- Google's Gemini is used to interpret on-screen elements on the compromised device and provide PromptSpy with dynamic instructions on how to execute a specific gesture to remain in the recent app list.
- The main (non-generative-AI-assisted) purpose of PromptSpy is to deploy a VNC module on the victim's device, allowing attackers to see the screen and perform actions remotely.
- PromptSpy has not been observed in our telemetry yet, making it a possible proof of concept; however, the discovery of a likely distribution domain suggests the existence of a variant targeting users in Argentina.
- PromptSpy can capture lockscreen data, block uninstallation, gather device info, take screenshots, record screen activity as video, and more.

PromptSpy's AI-powered functionality

Even though PromptSpy uses Gemini in just one of its features, it still demonstrates how incorporating these AI tools can make malware more dynamic, giving threat actors ways to automate actions that would normally be more difficult with traditional scripting.

As was briefly mentioned already, Android malware usually depends on hardcoded screen features such as taps, coordinates, or UI selectors – methods that can break with UI changes across devices, OS versions, or manufacturer skins. PromptSpy aims to achieve persistence by staying embedded in the list of recent apps by executing the “lock app in recent apps” gesture (the full process is described in the section), which varies between devices and manufacturers. This makes it difficult to automate with fixed scripts traditionally used by Android malware.

PromptSpy therefore takes a completely different approach: it sends Gemini a natural-language prompt along with an XML dump of the current screen, giving the AI a detailed view of every UI element: its text, type, and exact position on the display.

Gemini processes this information and responds with JSON instructions that tell the malware what action to perform (for example, a tap) and where to perform it. The malware saves both its previous prompts and Gemini’s responses, allowing Gemini to understand context and to coordinate multistep interactions.

Figure 1 shows a code snippet of PromptSpy’s initialization of communication with Gemini, including the first prompt used. By handing the decision-making over to Gemini, the malware can recognize the correct UI element and perform the appropriate gesture, keeping the malware alive even if the user tries to close it.

```

public static void startAutomationLoop(AccessibilityService accessibilityService, String str, String str2) {
    String str3;
    String str4;
    JSONArray jsonArray;
    String str5;
    int i;
    int i2;
    boolean z;
    AccessibilityService accessibilityService2 = accessibilityService;
    String str6 = str;
    if (accessibilityService2 == null) {
        return;
    }
    int i3 = 4;
    String str7 = TAG;
    if (str2 == null || str2.isEmpty()) {
        ServiceInteractionUtil.ToLog(TAG, "未设置 Gemini API Key, 无法执行自动化任务", 4);
        return;
    }
    ServiceInteractionUtil.ToLog(TAG, "开始执行任务: " + str6);
    JSONArray jsonArray2 = new JSONArray();
    String string = accessibilityService2.getString(R.string.atuo_load_msg);
    int i4 = 1;
    boolean z2 = true;
    int i5 = 0;
    while (z2 && i5 < 30) {
        int i6 = i5 + 1;
        ServiceInteractionUtil.ToLog(str7, ">>> 步骤 " + i6);
        AccessibilityNodeInfo accessibilityNodeInfoGetActiveWindowNodeInfo = InputService.GetActiveWindowNodeInfo();
        String aIXml = AccessibilityNodeUtil.toAIXml(accessibilityNodeInfoGetActiveWindowNodeInfo);
        if (accessibilityNodeInfoGetActiveWindowNodeInfo != null) {
            accessibilityNodeInfoGetActiveWindowNodeInfo.recycle();
        }
        if (i6 == i4) {
            str3 = "You are an Android automation assistant. The user will give you the UI XML data of the current screen."
        } else {
            str3 = "The previous action has been executed. This is the new UI XML, please determine if the task is complet
        }
        addToHistory(jsonArray2, "user", str3);
        String strCallGeminiApi = callGeminiApi(str2, jsonArray2);
        if (strCallGeminiApi == null) {
            ServiceInteractionUtil.ToLog(str7, "API 请求失败, 终止任务", i3);
            AutoClicker.bringHomeToForeground();
            return;
        }
    }
}

```

Figure 1. Malware code snippet with hardcoded prompts

PromptSpy continues prompting Gemini until the AI confirms that the app has been successfully locked, showing a feedback loop where the malware waits for validation before moving on.

PromptSpy overview

In February 2026, we uncovered two versions of a previously unknown Android malware family. The first version, which we named VNCSpy, appeared on VirusTotal on January 13th, 2026 and was represented by three samples uploaded from Hong Kong. On February 10th, 2026, four samples of more advanced malware based on VNCSpy were uploaded to VirusTotal from Argentina.

Our analysis of the samples from Argentina revealed multistage malware with a malicious payload that misuses Google's Gemini. Based on these findings, we named the first stage of this malware PromptSpy dropper, and its payload PromptSpy.

It should be noted that we haven't yet seen any samples of the PromptSpy dropper or its payload in our telemetry, which might indicate that both of them are just proofs of concept. However, based on the existence of a possible distribution domain described in the following paragraphs, we cannot discount the possibility of the PromptSpy dropper and PromptSpy existing in the wild.

According to VirusTotal data, all four PromptSpy dropper samples were distributed through the website mgardownload[.]com; it was already offline during our analysis.

After installing and launching PromptSpy dropper, it opened a webpage hosted on m-mgarg[.]com. Although this domain was also offline, Google's cached version revealed that it likely impersonated a Chase Bank (legally, JPMorgan Chase Bank N.A.) site (see Figure 2).

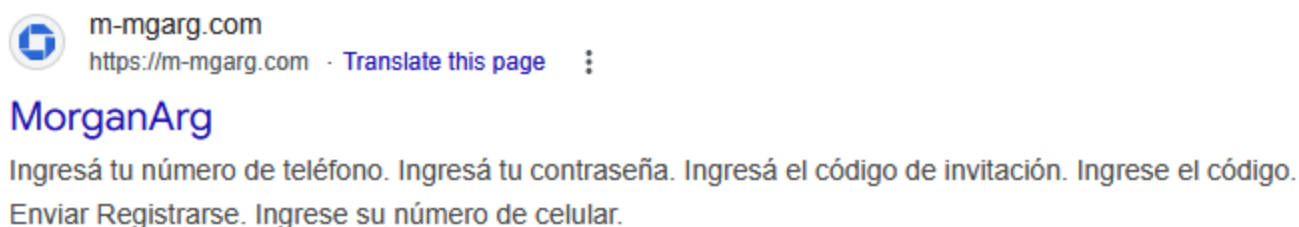


Figure 2. Google's cached data for the fake website

The malware uses similar branding, with the app name MorganArg and the icon inspired by Chase bank (see Figure 3). MorganArg, likely a shorthand for "Morgan Argentina", also appears as the name of the cached website, suggesting a regional targeting focus.



Install unknown apps



MorganArg

7.0

Allow from this source



Your phone and personal data are more vulnerable to attack by unknown apps. By installing apps from this source, you agree that you are responsible for any

damage to your phone or loss of data that may result from their use.



Please enable permission to install
from unknown sources

Figure 3. Dropper requests permission to install unknown apps to proceed with PromptSpy installation

We used the m-mgarg[.]com domain to pivot in VirusTotal, leading us to yet another Android malware sample (Android/Phishing.Agent.M). VirusTotal showed the spoofed website in Spanish, with an Iniciar sesión (Login) button, indicating that the page was probably intended to mimic a website of a bank (see Figure 4).

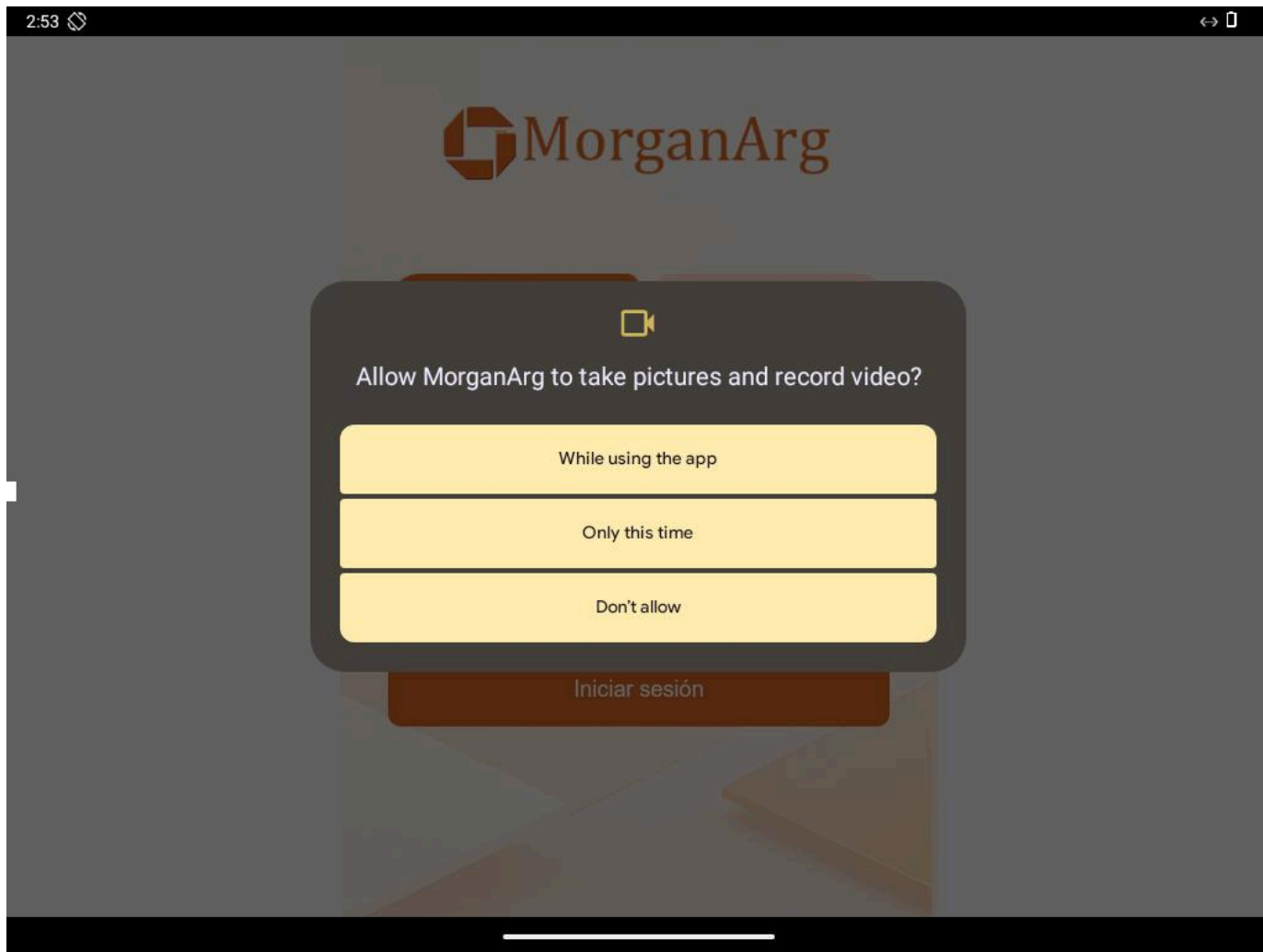


Figure 4. User interface of Android/Phishing.Agent.M displaying the same fake website as PromptSpy dropper (source:

<https://www.virustotal.com/gui/file/4ee3b09dd9a787ebbb02a637f8af192a7e91d4b7af1515d8e5c21e1233f0f1c7>
)

This trojan appears to function as a companion application developed by the same threat actor behind VNCSpy and PromptSpy. In the background, the trojan contacts its server to request a configuration file, which includes a link to download another APK, presented to the victim, in Spanish, as an update. During our research, the configuration server was no longer accessible, so the exact download URL remains unknown. However, given that it uses the same unique bank spoofing website, the same app name, icon, and, most importantly, is signed by the same unique developer certificate as the PromptSpy dropper – we strongly suspect this app may serve as the initial stage designed to lead victims toward installing PromptSpy.

Both VNCSpy and PromptSpy include a VNC component, giving their operators full remote access to compromised devices once victims enable Accessibility Services (see Figure 5). This allows the malware operators to see everything happening on the device, and to perform taps, swipes, gestures, and text input as though they were physically holding the phone.



Allow MorganArgs to have full control of your device?

Full control is appropriate for apps that help you with accessibility needs, but not for most apps.



View and control screen

It can read all content on the screen and display content over other apps.



View and perform actions

It can track your interactions with an app or a hardware sensor, and interact with apps on your behalf.

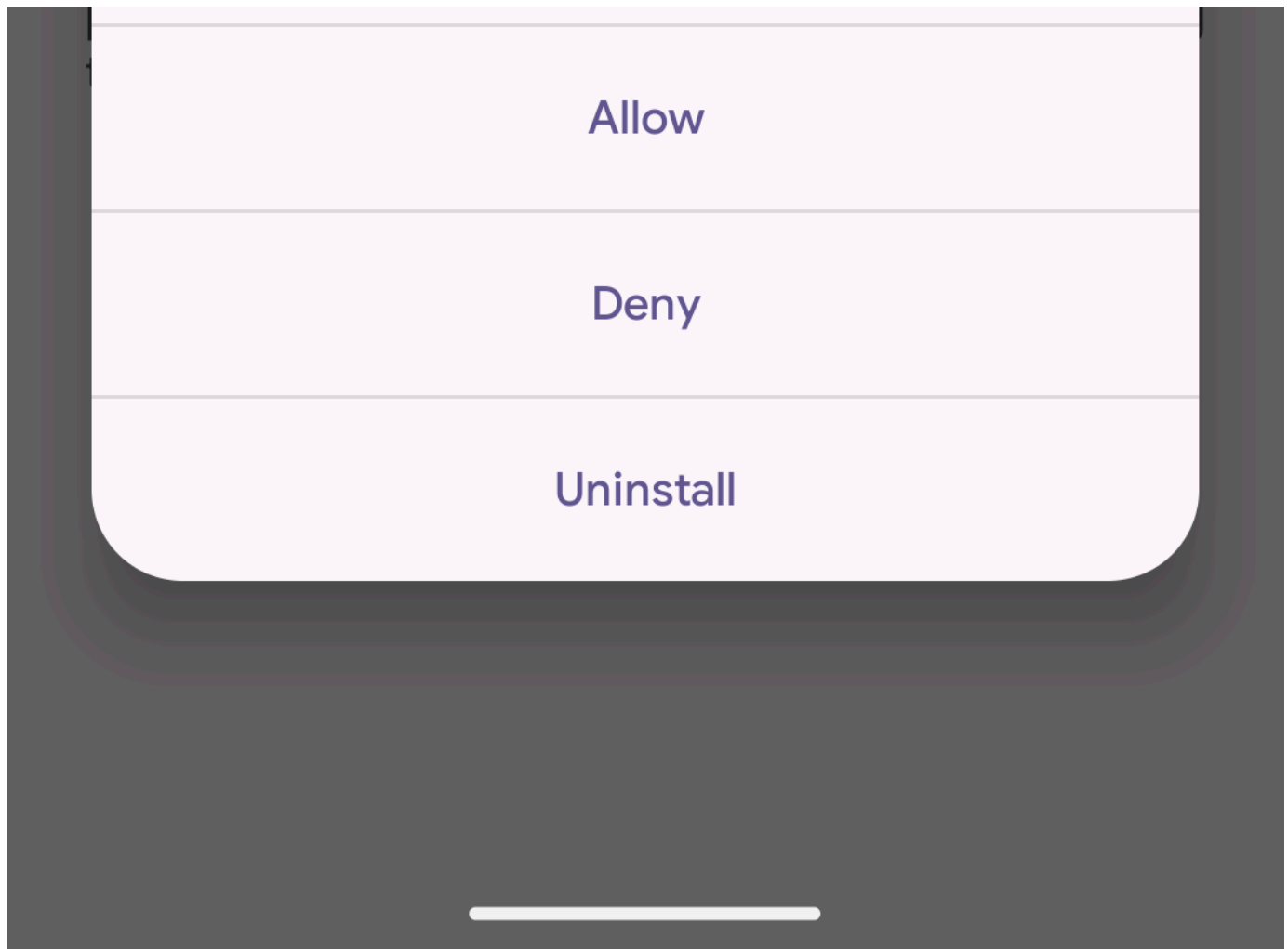


Figure 5. PromptSpy requests the victim to allow Accessibility services

On top of the malicious capabilities already contained in VNCSpy, PromptSpy adds AI-assisted UI manipulation, helping it maintain persistence by keeping the malicious app pinned in the recent apps list (an example of how the lock is indicated in the list can be seen in Figure 6).

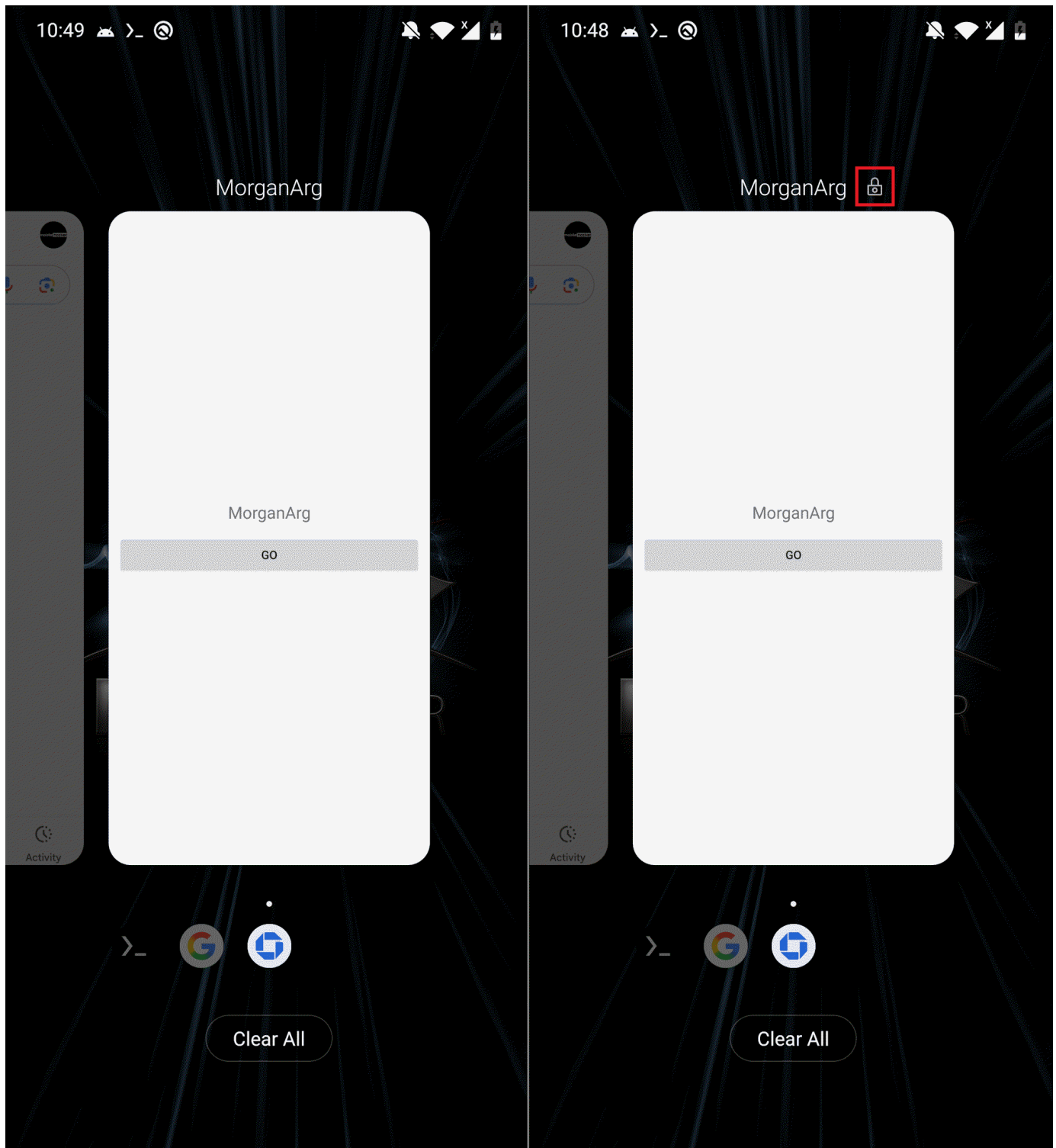


Figure 6. Not locked (left) and locked (right) MorganArg app in the list of recent apps, with the padlock icon representing the lock

We believe this functionality is used before the VNC session is established, so that the user or system will not kill the PromptSpy activity from the list of recent apps. In Figure 7, you can see PromptSpy network communication with Gemini AI.

#	Host	Method	URL	Params	Status code	Length	MIME type	TLS	IP
589	https://generativelanguage.googleapis.com	POST	/v1beta/models/gemini-2.5-flash-lite:generateContent?key=AlzaSy...	✓	200	1231	JSON	✓	142.251.209.10
590	https://generativelanguage.googleapis.com	POST	/v1beta/models/gemini-2.5-flash-lite:generateContent?key=AlzaSy...	✓	200	1600	JSON	✓	142.251.209.10
591	https://generativelanguage.googleapis.com	POST	/v1beta/models/gemini-2.5-flash-lite:generateContent?key=AlzaSy...	✓	200	1798	JSON	✓	142.251.209.10
593	https://generativelanguage.googleapis.com	POST	/v1beta/models/gemini-2.5-flash-lite:generateContent?key=AlzaSy...	✓	200	1599	JSON	✓	142.251.209.10
594	https://generativelanguage.googleapis.com	POST	/v1beta/models/gemini-2.5-flash-lite:generateContent?key=AlzaSy...	✓	200	1392	JSON	✓	142.251.209.10
595	https://generativelanguage.googleapis.com	POST	/v1beta/models/gemini-2.5-flash-lite:generateContent?key=AlzaSy...	✓	200	1470	JSON	✓	142.251.209.10
596	https://generativelanguage.googleapis.com	POST	/v1beta/models/gemini-2.5-flash-lite:generateContent?key=AlzaSy...	✓	503	591	JSON	✓	142.251.209.10

Request

```

1 POST /v1beta/models/gemini-2.5-flash-lite:generateContent?key=
2 Content-Type: application/json
3 User-Agent:
4 Host: generativelanguage.googleapis.com
5 Connection: keep-alive
6 Accept-Encoding: gzip, deflate, br
7 Content-Length: 4871
8
9 {
  "contents": [
    {
      "parts": [
        {
          "text":
            "You are an Android automation assistant. The user will give you the UI XML data of the current screen. You need to analyze the XML and output operation instructions in JSON format to achieve the user's goal. Nodes in the XML contain 'bounds' attributes in the format '[left,top][right,bottom]'. You need to calculate the center coordinates to generate click instructions. Core Judgment Rules: 1. Do NOT guess that the task is completed. Only return 'COMPLETED' when you clearly see visual evidence of success in the current UI XML (e.g., text like 'Saved', 'Success' appears, switch status becomes checked='true', or the screen has navigated to the target state). 2. If you performed the last step but the current XML does not reflect the result yet, return 'IN_PROGRESS' with action 'NONE' (or wait) to check the new UI state in the next cycle. 3. If unsure, remain 'IN_PROGRESS' and attempt to verify. 4. If the current UI XML is empty, use 'RECENTS' to access the recent apps list. You can use SWIPE to scroll/slide to find targets. When action_type = 'SWIPE', you must provide x1,y1,x2,y2,duration_ms. e.g., Scroll UP: slide from bottom to top (y2 < y1). Please strictly follow this JSON output format, do not output any Markdown tags or extra text: {\"status\": \"IN_PROGRESS\" | \"COMPLETED\" | \"IMPOSSIBLE\", \"reasoning\": \"Detailed explanation: what specific text or state I saw on the screen to judge the task is completed or needs navigation\"}
          }
        ]
      }
    ]
  }
}

```

Response

```

1 HTTP/2 200 OK
2 Content-Type: application/json; charset=UTF-8
3 Vary: Origin
4 Vary: X-Origin
5 Vary: Referer
6 Date: Fri, 13 Feb 2026 07:43:26 GMT
7 Server: scaffolding on HTTPServer2
8 Content-Length: 855
9 X-Xss-Protection: 0
10 X-Frame-Options: SAMEORIGIN
11 X-Content-Type-Options: nosniff
12 Server-Timing: gfet4t7; dur=872
13 Alt-Svc: h3=443; ma=2592000,h3-29=443; ma=2592000
14
15 {
16   "candidates": [
17     {
18       "content": {
19         "parts": [
20           {
21             "text":
22               "\n\n\"status\": \"IN_PROGRESS\", \"reasoning\": \"The user wants to lock the 'MorganArgs' app. The current screen shows the recent apps. I need to long-click on the 'MorganArgs' app preview to bring up the options menu, which should include a lock option.\\n\n\"action_type\": \"LONG_CLICK\", \"x\": 540, \"y\": 1126\\n\n\"}
23           ]
24         },
25         "role": "model"
26       },
27       "finishReason": "STOP",
28       "index": 0
29     },
30     "usageMetadata": {
31       "promptTokenCount": 1463,
32       "candidatesTokenCount": 100,
33       "totalTokenCount": 1563,
34       "promptTokensDetails": {

```

Figure 7. Network communication of malware and Gemini with prompt request and response shown in red rectangles

Origins

While analyzing PromptSpy, we noticed that it contains debug strings written in simplified Chinese. It even includes handling for various Chinese Accessibility event types (see Figure 8), a debug method that had been disabled in the code but not removed. The primary purpose of this method is to provide a localized (Chinese) explanation for various accessibility events that occur on an Android device. This makes the event logs more understandable for Chinese-speaking users or developers, rather than just displaying raw integer codes.

```

public static String getChineseEventType(int i) {
    if (i == 1) {
        return "视图被点击";    View clicked
    }
    if (i == 2) {
        return "视图被长按";    View long-pressed
    }
    switch (i) {
        case 4:
            return "视图被选中";    View selected
        case 8:
            return "视图获得焦点";    View gains focus
        case 16:
            return "视图文本改变";    View text changed
        case 32:
            return "窗口状态改变";    Window state changed
        case 64:
            return "通知状态改变";    Notification state changed
        case 128:
            return "视图悬停进入";    View hover enters
        case 256:
            return "视图悬停退出";    View hover exits
        case 512:
            return "触摸浏览手势开始";    Touch browsing gesture begins
        case 1024:
            return "触摸浏览手势结束";    Touch browsing gesture ends
        case 2048:
            return "窗口内容改变";    Window content changes
        case 4096:
            return "视图滚动";    View scrolls
        case 8192:
            return "视图文本选择改变";    View text selection changes
        case 16384:
            return "通知/公告";    Notification/Announcement
    }
}

```

Figure 8. Parsing and logging various event types

With medium confidence, these details suggest that PromptSpy was developed in a Chinese-speaking environment.

Analysis

Our technical analysis focuses on the PromptSpy dropper and its payload, PromptSpy. PromptSpy is embedded (app-release.apk) inside the dropper's asset directory. This APK holds the core malicious functionality. When the dropper is launched, it displays a prompt urging the user to install what appears to be an updated version of the app. This "update" is actually the PromptSpy payload, which the user must install manually (see Figure 9).

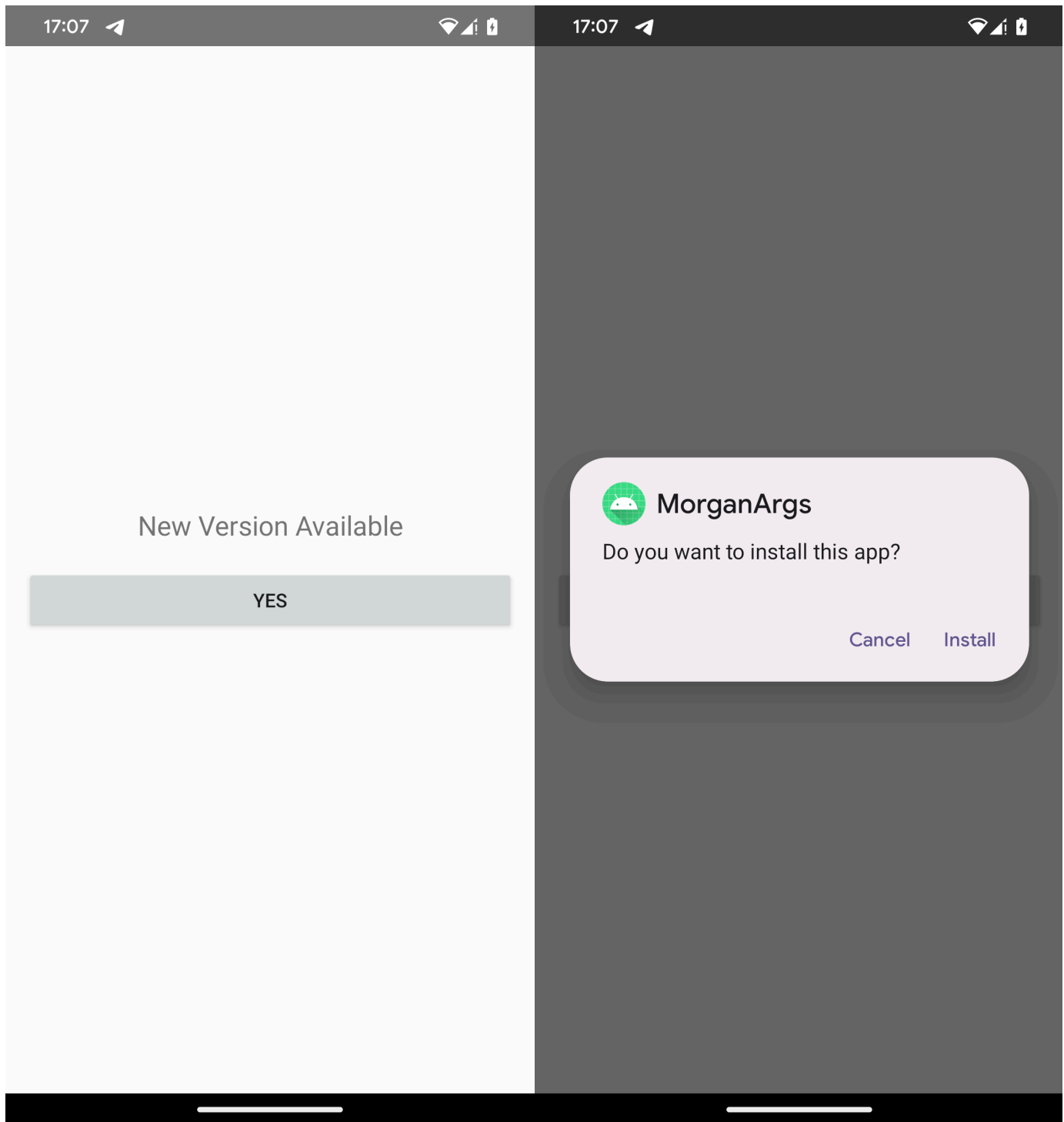


Figure 9. Malware's initial screen that requests to install PromptSpy payload

Once installed and launched, PromptSpy requests [Accessibility Service](#) permissions, giving the malware the ability to read on-screen content and perform automated clicks.

Then PromptSpy shows a simple loading-style decoy screen in the foreground (see Figure 10). Meanwhile, in the background, it begins communicating with Gemini AI to obtain instructions needed to lock its process in the Recent Apps list – a simple persistence technique that allows PromptSpy to remain active and locked in place even after the device is rebooted.

Loading, please wait. 70%

Figure 10. Overlay decoy activity

When the user sees the Loading, please wait activity, PromptSpy uses Accessibility Services to open the Recent Apps screen and collect detailed UI information: visible text, content descriptions, class names, package names, and screen bounds. It serializes this dynamic UI snapshot as XML and includes it in its prompt to Gemini. Gemini then returns step-by-step tap instructions on how to achieve the “app lock” gesture.

This process forms a continuous loop:

- PromptSpy sends updated UI context to Gemini
- Gemini replies with new actions
- PromptSpy executes them and returns the resulting screen state

The loop continues until Gemini confirms that the app is successfully locked in recent apps. Here is an example structure:

- Prompt: You are an Android automation assistant. The user will give you the UI XML data of the current screen. You need to analyze the XML and output operation instructions in JSON format to achieve the user's goal.
Nodes in the XML contain 'bounds' attributes in the format '[left,top][right,bottom]'. You need to calculate the center coordinates to generate click instructions.

*** Core Judgment Rules ***
1. **Do NOT guess that the task is completed**. Only return 'COMPLETED' when you clearly see visual evidence of success in the current UI XML (e.g., text like 'Saved', 'Success' appears, switch status becomes checked=\"true\", or the screen has navigated to the target state).
2. If you performed the last step but the current XML does not reflect the result yet, return 'IN_PROGRESS' with action 'NONE' (or wait) to check the new UI state in the next cycle.
3. If unsure, remain 'IN_PROGRESS' and attempt to verify.
4. If the current UI XML is empty, use 'RECENTS' to access the recent apps list.

You can use SWIPE to scroll\slide to find targets:
When action_type = \"SWIPE\", you must provide x1,y1,x2,y2,duration_ms.
e.g., Scroll UP: slide from bottom to top (y2 < y1).

Please strictly follow this JSON output format, do not output any Markdown tags or extra text:
{
 \"status\": \"IN_PROGRESS\" |
 \"COMPLETED\" | \"IMPOSSIBLE\",
 \"reasoning\": \"Detailed explanation: what specific text or state I saw on the screen to judge the task is completed or needs next
<REDACTED>
User Goal: Lock the current MorganArgs app in the recent apps list. Device info: <DEVICE MODEL>
Current UI XML: <UI>
Gemini's response (action instructions): status\": \"IN_PROGRESS\",
 \"reasoning\": \"I have accessed the recent apps list. I can see 'MorganArgs' listed in the recent apps. The next step is to long-click on the 'MorganArgs' app card to bring up the options menu, which should include an option to lock it.\"\",
 \"action_type\": \"LONG_CLICK\",
 \"x\": 586,
 \"y\": 1126,
 \"x1\": -1,
 \"y1\": -1,
 \"x2\": -1,
 \"y2\": -1,
 \"duration_ms\": -1
Malware's follow-up with updated screen results: The previous action has been executed. This is the new UI XML, please determine if the task is complete or provide the next instruction: <CURRENTLY VISIBLE ACTIVITY>

All actions suggested by Gemini – taps, swipes, navigation – are executed through Accessibility Services, allowing the malware to interact with the device without user input.

PromptSpy's main malicious capability lies in its built-in VNC service. This allows attackers to remotely view the victim's screen in real time and fully control the device.

The malware communicates with its hardcoded command-and-control (C&C) server at 54.67.2[.]84 using the VNC protocol; the messages are AES-encrypted using a hardcoded key. Through this communication channel, the malware can:

- receive a Gemini API key,
- upload the list of installed apps,
- intercept the lockscreen PIN or password,
- capture the pattern unlock screen as a recording video,
- report whether the screen is on or off,

- report the current foreground app,
- record the screen and user gestures for apps specified by the server, and
- take screenshots on demand.

PromptSpy also misuses Accessibility Services as an anti-removal mechanism. When the user attempts to uninstall the payload or disable Accessibility Services, the malware overlays transparent rectangles on specific screen areas - particularly over buttons containing substrings like stop, end, clear, and Uninstall. These overlays are invisible to the user but intercept interactions, making removal difficult. In Figure 11, we've run PromptSpy with the debug flag enabled (kept there by developers) that would set the color of the transparent rectangle, to visualize where they are specifically displayed. However, on the actual device, they are fully invisible.

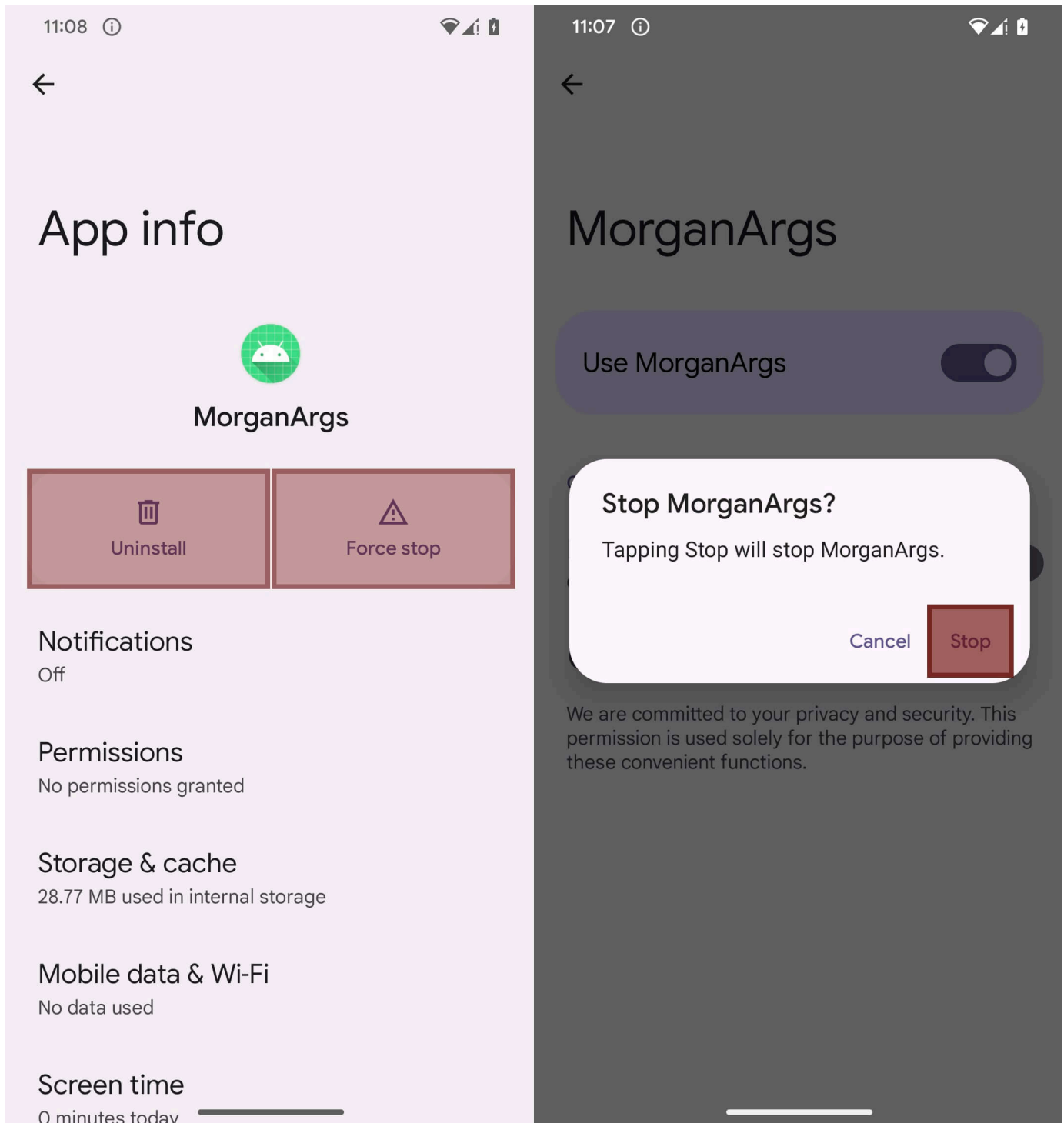


Figure 11. Invisible rectangles (displayed in color for clarity) covering specific buttons

Because PromptSpy blocks uninstallation by overlaying invisible elements on the screen, the only way for a victim to remove it is to reboot the device into Safe Mode, where third-party apps are disabled and can be uninstalled normally.

To enter Safe Mode, users should typically press and hold the power button, long-press Power off, and confirm the Reboot to Safe Mode prompt (though the exact method may differ by device and manufacturer). Once the phone restarts in Safe Mode, the user can go to Settings → Apps

→ MorganArg and uninstall it without interference.

Conclusion

PromptSpy shows that Android malware is beginning to evolve in a sinister way. By relying on generative AI to interpret on-screen elements and decide how to interact with them, the malware can adapt to virtually any device, screen size, or UI layout it encounters. Instead of hardcoded taps, it simply hands AI a snapshot of the screen and receives precise, step-by-step interaction instructions in return, helping it achieve a persistence technique resistant to UI changes.

More broadly, this campaign shows how generative AI can make malware far more dynamic and capable of real-time decision-making. PromptSpy is an early example of generative AI-powered Android malware, and it illustrates how quickly attackers are beginning to misuse AI tools to improve impact.

For any inquiries about our research published on WeLiveSecurity, please contact us at threatintel@eset.com.

ESET Research offers private APT intelligence reports and data feeds. For any inquiries about this service, visit the [ESET Threat Intelligence](#) page.

IoCs

A comprehensive list of indicators of compromise (IoCs) and samples can be found in [our GitHub repository](#).

Files

SHA-1	Filename	Detection	Description
6BBC9AB132BA066F63676E05DA13D108598BC29B	net.ustexas.myavlive.apk	Android/Spy.VNCSpy.A	Android VNCSpy malware.
375D7423E63C8F5F2CC814E8CFE697BA25168AFA	nlll4.un7o6.q38l5.apk	Android/Spy.VNCSpy.A	Android VNCSpy malware.
3978AC5CD14E357320E127D6C87F10CB70A1DCC2	ppyzz.dpk0p.ln441.apk	Android/Spy.VNCSpy.A	Android VNCSpy malware.
E60D12017D2DA579DF87368F5596A0244621AE86	mgappc-1.apk	Android/Spy.PromptSpy.A	Android PromptSpy dropper.

SHA-1	Filename	Detection	Description
9B1723284E3117949879 97CB7E8814EB6014713F	mgappm-1.apk	Android/Spy.PromptSpy.A	Android PromptSpy dropper.
076801BD9C6EB78FC033 1A4C7A22C73199CC3824	mgappn-0.apk	Android/Spy.PromptSpy.A	Android PromptSpy dropper.
8364730E9BB2CF3A4B01 6DE1B34F38341C0EE2FA	mgappn-1.apk	Android/Spy.PromptSpy.A	Android PromptSpy dropper.
F8F4C5BC498BCCE907DC 975DD88BE8D594629909	app-release.apk	Android/Spy.PromptSpy.A	Android PromptSpy.
C14E9B062ED28115EDE0 96788F62B47A6ED841AC	mgapp.apk	Android/Phishing.Agent.M	Android phishing malware.

Network

IP	Domain	Hosting provider	First seen	Details
52.222.205[.]45	m-mgarg[.]com	Amazon.com, Inc.	2026-01-12	Phishing website.
54.67.2[.]84	N/A	Amazon.com, Inc.	N/A	C&C server.
104.21.91[.]170	mgardownload[.]com	Cloudflare, Inc.	2026-01-13	Distribution website.

MITRE ATT&CK techniques

This table was built using [version 18](#) of the MITRE ATT&CK framework.

Tactic	ID	Name	Description
Persistence	T1398	Boot or Logon Initialization Scripts	PromptSpy receives the BOOT_COMPLETED broadcast intent to activate at device startup.
	T1541	Foreground Persistence	PromptSpy uses foreground persistence to keep a service running.

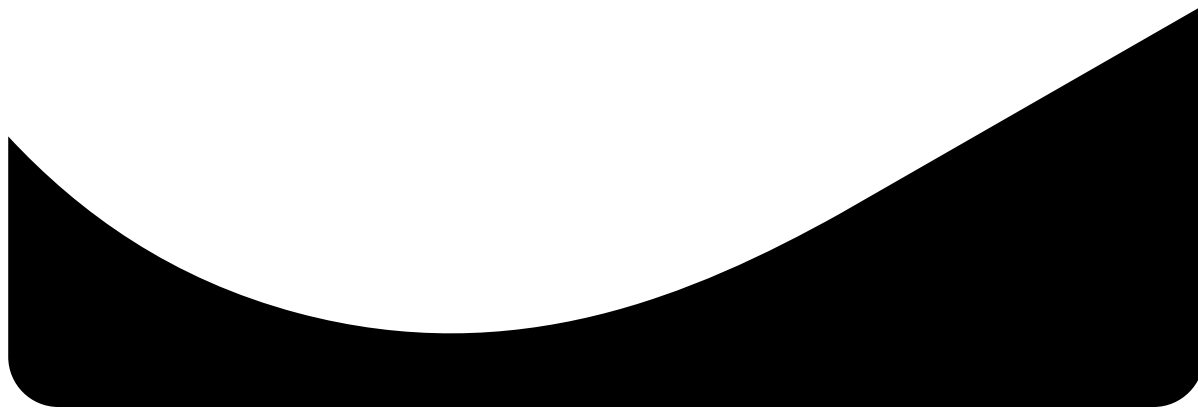
Tactic	ID	Name	Description
Defense Evasion	T1516	Input Injection	PromptSpy abuses the accessibility service to prevent its removal.
Credential Access	T1417.002	Malicious Third Party Keyboard App: GUI Input Capture	PromptSpy can intercept Android lockscreen PIN and password.
Discovery	T1426	System Information Discovery	PromptSpy obtains device name, model, and OS version.
Collection	T1418	Software Discovery	PromptSpy can obtain a list of installed applications.
	T1513	Screen Capture	PromptSpy can record the screen.
Command and Control	T1663	Remote Access Software	PromptSpy can use VNC to remotely control a compromised device.
	T1521.001	Standard Cryptographic Protocol: Symmetric Cryptography	PromptSpy encrypts C&C communication using AES.
Exfiltration	T1646	Exfiltration Over C2 Channel	PromptSpy can exfiltrate collected data to the C&C server.



Let us keep you up to date

Sign up for our newsletters

Ukraine Crisis newsletter Regular weekly newsletter



Related Articles

[ESET Research](#)

[DynoWiper update: Technical analysis and attribution](#)

[ESET Research](#)

[Love? Actually: Fake dating app used as lure in targeted spyware campaign in Pakistan](#)

[ESET Research](#)

[ESET Research: Sandworm behind cyberattack on Poland's power grid in late 2025](#)