Analysis of Phishing Activities That Deliver Qbot Banking Trojan Using XLL Files

Time of first release: 26 April, 2023

The original report is in Chinese, and this version is an AI-translated edition.

1 Overview

Recently, the Antiy CERT discovered a malicious activity that utilized malicious Microsoft Excel add-in (XLL) files to deliver the Qbot banking Trojan. The attackers sent spam emails to induce users to open the XLL files within the attachments. Once users installed and activated the Microsoft Excel add-in, the malicious code would be executed. Subsequently, the malware would decrypt layer by layer on the user's host and finally release the Qbot banking Trojan.

Since Microsoft announced in February 2023 that it would by default block macros in Office documents, attackers have attempted to use other types of files as new media for spreading malware. The phishing activities that use XLL files to spread malicious files began to increase at the end of 2021. Currently, multiple malware families such as Dridex, Qbot, Formbook, and AgentTesla use XLL files for dissemination. When users open the XLL file, Excel is launched and the XLL file is loaded and executed as an Excel add-in, bypassing the restrictions of Office macro documents.

The Qbot banking Trojan was first discovered in 2008 and has been active since April 2020, mainly spreading through spam emails. In February 2021, the Antiy CERT released the "Analysis Report on the 2020 Activities of the Qbot Banking Trojan" [1]. During its execution, the banking Trojan decrypts multiple times, uses a loader to load and execute the malicious function to evade static detection by anti-virus software, uses scheduled tasks to achieve self-starting, and can obtain screenshots of the victim's host, collect information about the target system, and obtain browser cookie information, etc. The attackers can also use the data stolen from the user to carry out subsequent attack activities.

After verification, the Antiy IEP can effectively detect and eliminate this banking Trojan.

2 ATT&CK Mapping graph of the event

For the complete process of the attackers' delivery of the bank Trojan, Antiy has compiled the corresponding ATT&CK mapping graph for this attack incident as shown in the following figure.

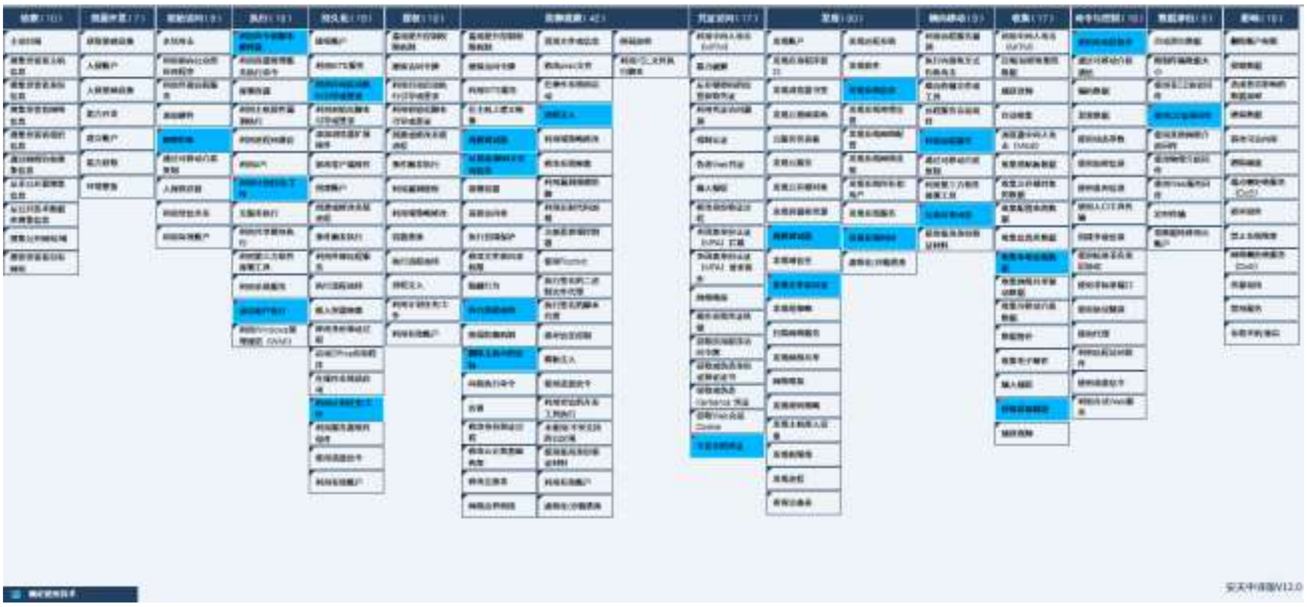


Figure 2-1 Graph of the technical features to ATT&CK 21

The technology points used by the attacker are shown in the table below.

Table 2-1 Description of ATT&CK technical behavior corresponding to the event 21

ATT&CK stages / categories	Specific behavior	Notes
Initial access	Phishing	Spread by phishing mail
Execution	Using command and script interpreters	Execute an XLL file with a command
	Inducing the user to execute	Inducing a user to execute a malicious file
Persistence	Use automatic startup to perform booting or logging	Add a registry startup key
	Utilization of planned tasks / jobs	Create a scheduled task for persistence
Defensive evasion	Anti-obfuscate / decode files or information	Decoding multi-layer payload information
	Remove the beacon from the host	Delete the samples under the created remote services and shared folders
	Execution process hijacking	Hijacking the system process
	Process injection	Into the ultimate bank Trojan horse

	Circumventing the debugger	Determine if a debugger exists
Credential Access	Insecure credentials	Get unsecure application software, registry of credentials
Findings	Circumventing the debugger	Discover the debugger tool process
	Find files and directories	Application software is found in the specified directory
	Discovery of system information	Discovery of system information
	System discovery time	System discovery time
Lateral movement	Use remote services	Start the service on a remote computer
	Contamination of shared content	Copies itself to a shared folder
Collection	Collect local system data	Collect local system data
	Get a screenshot	Get a screenshot
Command and control	The application layer protocol is used	Use the HTTPS protocol
Data seeps out	The C2 channel is used for backtransmission	The c2 channel is used to return the data

3 Recommendations for protection

In order to effectively prevent such attacks and improve the level of security protection, Antiy suggests the enterprise take the following protection measures:

3.1 Identify phishing mail

1. Check mail senders: Watch out for non-organizational senders who send "business mail";
2. Check the addressee's address: Be alert to group email, and contact the addressee for confirmation;
3. See the delivery time: Watch out for the non-working time sent mail;
4. Read the email title: Watch out for emails with the title of "order," "bill," "wage subsidy," "purchase" and other keywords;
5. See the wording of the text: Alert to "pro," "dear users," "dear colleagues" and other more general greetings of the mail;
6. Purpose of reading the text: Be alert to the emails that ask for the account password in the name of "system upgrade," "system maintenance" and "security setting";

7. Look at the main content: Alert to the attached web links, especially short links;

8. Content of the attachment: Before viewing, virus scanning and monitoring of the attachment shall be performed using anti-virus software.

3.2 Daily Email security usage protection

1. Install terminal protection software: Install terminal protection software, open the function of scanning and detecting email attachments in the protection software, regularly conduct security detection on the system, and repair system vulnerabilities.

2. Email login password: The email login password shall be set with certain complexity (including three character elements), the password shall not be recorded in an obvious place in the office area, and the login password shall be changed regularly.

3. Email account shall be bound with mobile phone: After the email account is bound with mobile phone, the user can not only retrieve the password, but also receive the SMS prompt of "abnormal login" for instant disposal.

4. Important documents shall be protected:

    1) Empty the inbox, outbox and trash of important mails that are no longer in use in time;

    2) Backup important files to prevent files from being lost after being attacked;

    3) Important emails or attachments shall be encrypted and sent, and no decryption password shall be attached to the text.

5. Sensitive information shall be protected: Do not release sensitive information on the Internet, and the information and data released by users on the Internet will be collected by attackers. By analyzing this information and data, attackers can send phishing emails to users in a targeted way.

3.3 Government, enterprise and institutional protection

1. Install the terminal protection software: Install the anti-virus software, and it is recommended to install the Antiy IEP;

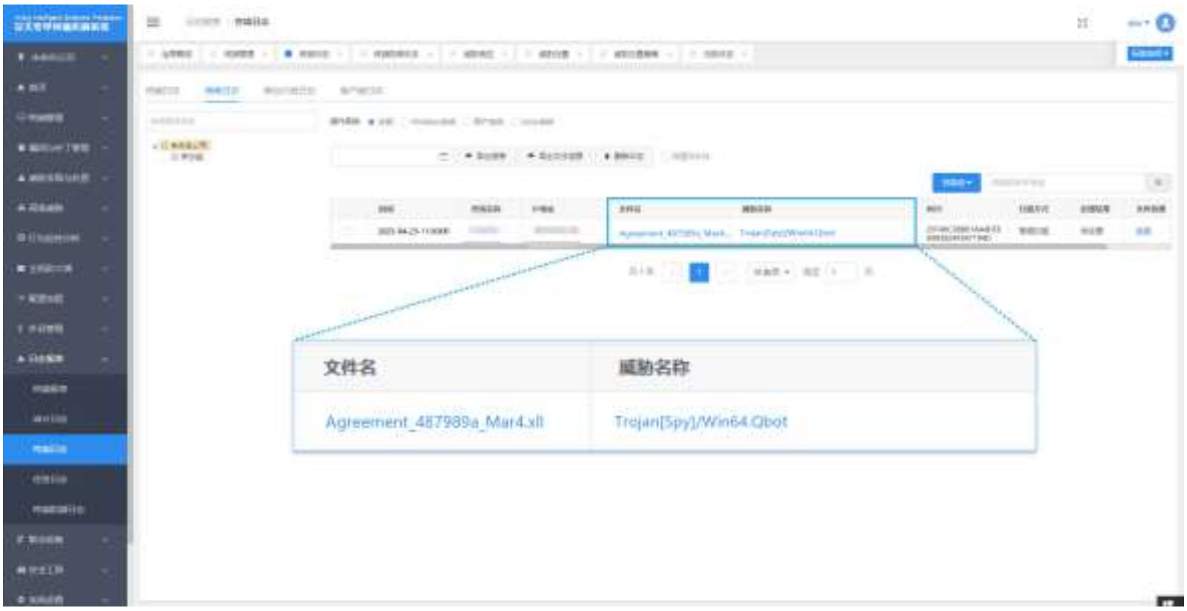
2. Strengthen password strength: Avoid using weak passwords, and recommend using 16-digit or longer passwords, including combinations of upper and lower case letters, numbers and symbols, and avoid using the same password for multiple servers;

3. Close PowerShell: If you do not use PowerShell command line tools within a certain period of time, it is recommended to close them;

4. Deployment of Intrusion Detection System (IDS): Deployment of traffic monitoring software or equipment to facilitate the discovery, tracing and tracing of malware. Taking network traffic as the detection and analysis object, the Antiy PTD can accurately detect a mass of known malware and network attack activities, and effectively detect suspicious behaviors, assets and various unknown threats on the network;

5. Security service: In case of malware attack, it is suggested to isolate the attacked host in time, and protect the site and wait for the security engineer to check the computer; 7 * 24 service hotline: 400-840-9234.

It has been proved that Antiy IEP can effectively kill the bank Trojan.



文件名	威胁名称
Agreement_487989a_Mar4.xll	Trojan[Spy]/Win64.Qbot

Figure 3-1 The effective detection and kill of the user system implemented by Antiy IEP1

4 Attack process

4.1 Attack flowchart

The attacker spreads the junk mail, induces the user to open the XLL file (Agreement _ 487989a _ Mar4.xll) in the attachment, Excel executes the export function xlAutoOpen containing the malicious code, and the malware decrypts and executes the subsequent payload. The creation process executes the cmd command to write the XLL file decrypted from the resource into the target file 3.dat, create the final sample for automatic execution of the scheduled task, and inject itself into the wermgr. exe process. Hijacking execution flow realizes such functions as obtaining system information, obtaining disk drive information, obtaining screen shot, creating pipeline monitoring connection, anti-debugging, judging whether debugging tools and anti-virus software process exist in the environment.

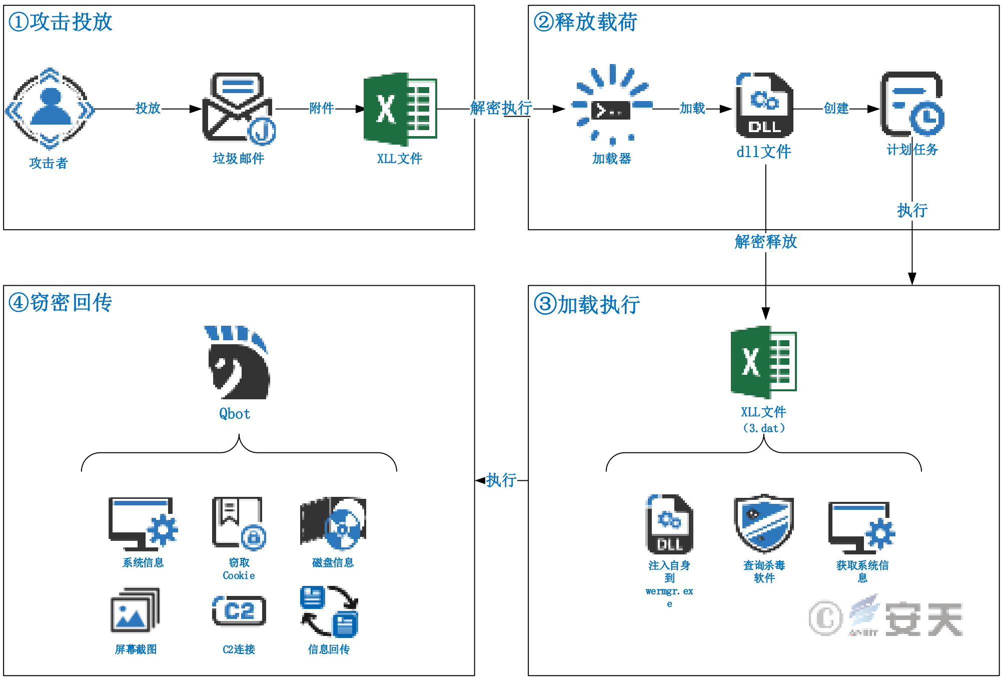
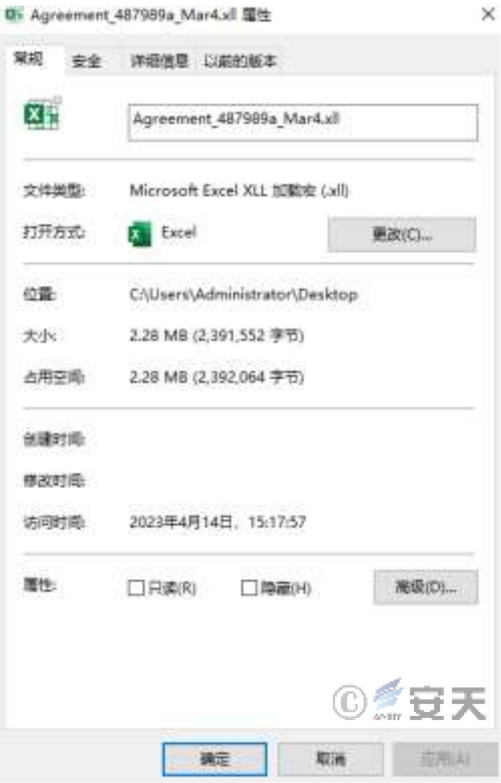


Figure 4-1 The attack flowchart1

4.2 Using XLL files to spread malicious files

The attacker sends spam to the user, inducing the user to open the XLL file in the attachment.



Agreement_487989a_Mar4.xll 属性

Agreement_487989a_Mar4.xll

文件类型: Microsoft Excel XLL 加载宏 (.xll)

打开方式: Excel

位置: C:\Users\Administrator\Desktop

大小: 2.28 MB (2,391,552 字节)

占用空间: 2.28 MB (2,392,064 字节)

创建时间:

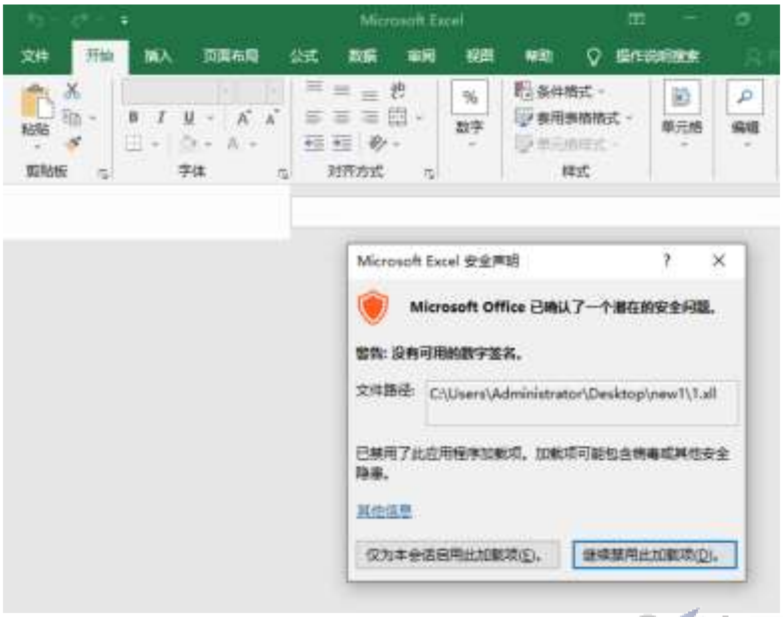
修改时间:

访问时间: 2023年4月14日，15:17:57

属性: □只读(R) □隐藏(H)

Figure 4-2 XLL file carrying malware2

After the user opens the XLL file, Windows Explorer automatically starts Excel to open the XLL file, and before loading the XLL file, Excel displays a warning that it may contain malicious code and prompts the user to install and activate the add-in.



Microsoft Excel 安全声明

Microsoft Office 已确认了一个潜在的安全问题。

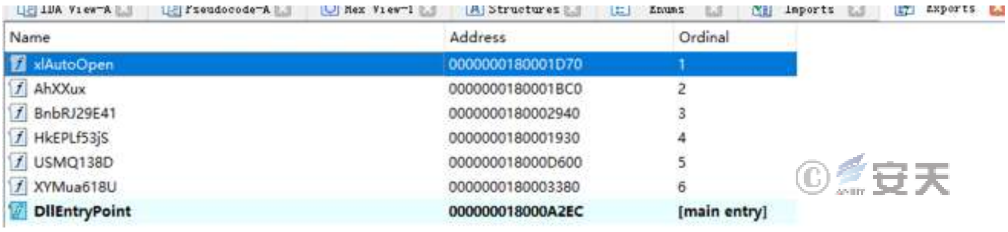
警告: 没有可用的数字签名。

文件路径: C:\Users\Administrator\Desktop\new1\1.xll

已禁用了此应用程序加载项。加载项可能包含病毒或其他安全隐患。

Figure 4-3 Open an XLL document

Xll files are standard Windows dynamic load libraries (dlls) in terms of file types. In order for that Excel add-in manager to successfully load the XLL file, the XLL file must implement at least one export function (called xlAutoOpen) to call the code when Excel loads the XLL file. An attacker typically places malware in the xlAutoOpen function, which triggers execution as soon as the add-on is activated. This means that, unlike VBA macros, which require users to enable macros, the victim will execute malicious code simply by opening the XLL file [2].[2]



Name	Address	Ordinal
xlAutoOpen	0000000180001D70	1
AhXXux	0000000180001BC0	2
BnbRJ29E41	0000000180002940	3
HkEPLf53jS	0000000180001930	4
USMQ138D	000000018000D600	5
XYMua618U	0000000180003380	6
DllEntryPoint	000000018000A2EC	[main entry]

Figure 4-4 The xlAutoOpen function3

5 Sample analysis

The attacker induces users to open the XLL file in the attachment by sending spam, and once the user installs and activates the Microsoft Excel add-in, the malicious code is executed. Subsequently, the malicious code will be in the user host for layers of decryption, the final release of the Qbot bank Trojan.

Qbot bank Trojan has the function of obtaining screen shot, obtaining target system information and browser cookie information on the victim host. In that execution process, the bank Trojan uses multi-layer decryption, load the decrypted file to avoid static detection and killing of the anti-virus software, uses the planned task to realize self-startup, and finally realizes malicious behaviors such as collection and monitoring. The sensitive data will be transmitted according to the needs of the attacker, the attacker can also use the data stolen from the user to carry out the subsequent attack activities.

5.1 Sample labels

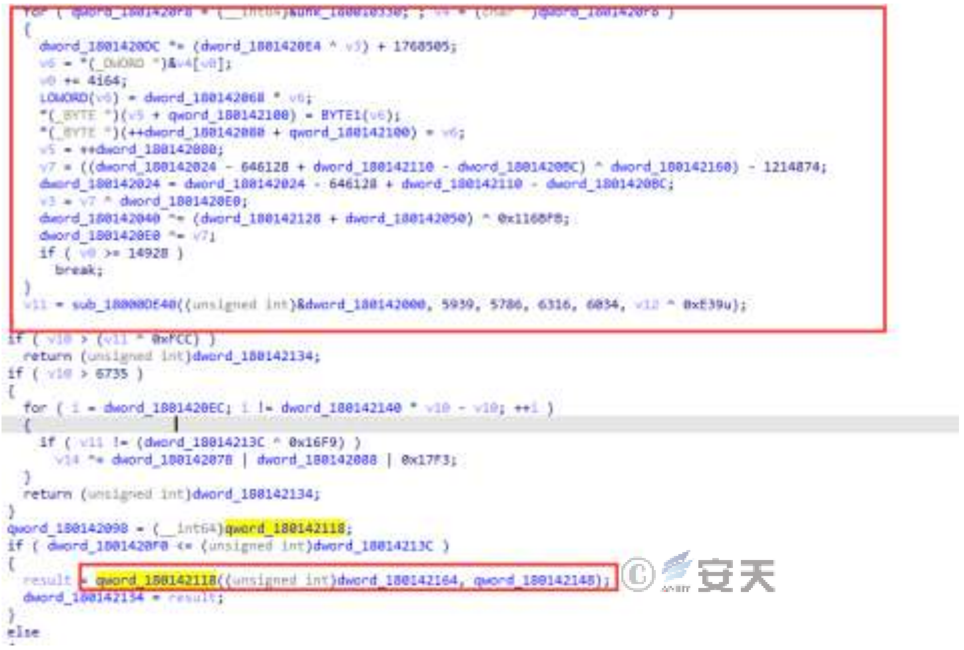
Table 5-1 Sample labels 1

Name of malware	Trojan [Spy] / Win64.Qbot
Original file name	Agreement _ 487989a _ Mar4.xll
Md5	20746c3bb01aa4deeea993824f947194d
Processor architecture	Advanced Micro Devices X86-64

File size	2.28 MB (239,1552 bytes)
File format	Binexecute / Microsoft.EXE [: X64]
Time stamp	2023-03-15 00: 10: 15
Digital signature	None
Shell type	None
Compiled Language	Microsoft Visual C + +
Vt First Upload Time	2023-03-14 20: 20: 02
Vt test result	47 / 69

5.2 First layer of code - decryption Shellcode

After the execution of the Agreement _ 487989a _ Mar4.xll file, the loader and a dll file will be cyclically decrypted from the memory. the decryption algorithm is shown as follows.



```
for ( qword_1801420F8 = (__int64)&unk_180010330; ; v4 = (char *)qword_1801420F8 )
{
  dword_18014200C *= (dword_1801420E4 ^ v5) + 1768505;
  v6 = *(_DWORD *)&v4[v0];
  v0 += 4164;
  LOWORD(v5) = dword_180142068 * v6;
  *(_BYTE *)(v5 + qword_180142100) = BYTE1(v6);
  *(_BYTE *)(++dword_180142088 + qword_180142100) = v6;
  v5 = ++dword_180142088;
  v7 = ((dword_180142024 - 646128 + dword_180142110 - dword_18014208C) ^ dword_180142160) - 1214874;
  dword_180142024 = dword_180142024 - 646128 + dword_180142110 - dword_18014208C;
  v3 = v7 ^ dword_1801420E0;
  dword_180142040 ^= (dword_180142128 + dword_180142050) ^ 0x1168F8;
  dword_1801420E0 ^= v7;
  if ( v0 >= 14928 )
    break;
}
v11 = sub_18000DE40((unsigned int)&dword_180142000, 5939, 5786, 6316, 6034, v10 ^ 0xE39u);
if ( v10 > (v11 ^ 0xFCC) )
  return (unsigned int)dword_180142134;
if ( v10 > 6735 )
{
  for ( i = dword_1801420EC; i != dword_180142140 * v10 - v10; ++i )
  {
    if ( v11 != (dword_18014213C ^ 0x16F9) )
      v14 ^= dword_180142078 | dword_180142088 | 0x17F3;
  }
  return (unsigned int)dword_180142134;
}
qword_180142098 = (__int64)qword_180142118;
if ( dword_1801420F8 <= (unsigned int)dword_18014213C )
{
  result = qword_180142118((unsigned int)dword_180142164, qword_180142148);
  dword_180142134 = result;
}
else
```

Figure 5-1 xlAutoOpen function decrypts the execution loader

The loader loads the execution dll file in memory.

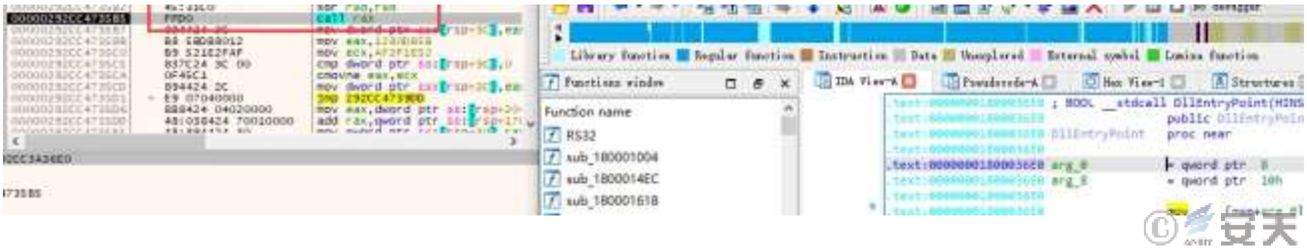
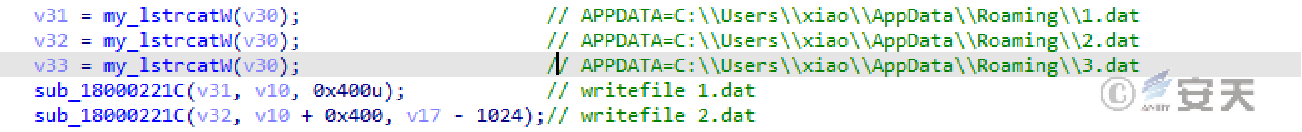


Figure 5-2 The loader executes the dll 1

5.3 Layer 2 code - dll file

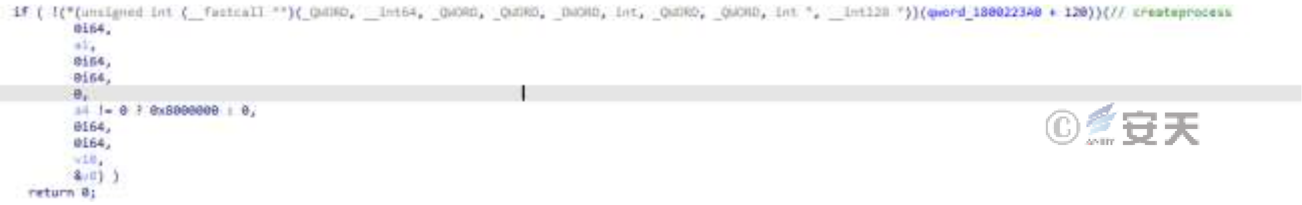
Dll files search resource data and load it into memory to decrypt, create 1.dat, 2.dat files, write the first 400 bytes of decrypted data into 1.dat file, and write the remaining bytes into 2.dat file.



```
v31 = my_lstrcatW(v30);                      // APPDATA=C:\\Users\\xiao\\AppData\\Roaming\\1.dat
v32 = my_lstrcatW(v30);                      // APPDATA=C:\\Users\\xiao\\AppData\\Roaming\\2.dat
v33 = my_lstrcatW(v30);                      // APPDATA=C:\\Users\\xiao\\AppData\\Roaming\\3.dat
sub_18000221C(v31, v10, 0x400u);             // writefile 1.dat
sub_18000221C(v32, v10 + 0x400, v17 - 1024);// writefile 2.dat
```

Figure 5-3 The decrypted data is written to files 2

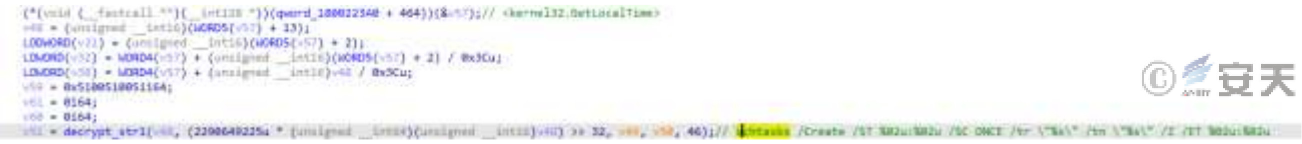
Dll files are decrypted and spliced out of commands, creating process execution commands. Its function is to read the contents of 1.dat, 2.dat files and write them to 3.dat, and use rundll32.exe to export the 3.dat file xlAutoOpen. According to the export function, the 3. dat file is also an XLL file.



```
  if ( !(*(unsigned int (__fastcall **)(_QWORD, __int64, _QWORD, _QWORD, _DWORD, int, _QWORD, _QWORD, int *, __int128 *))(qword_1800223A8 + 120))(// createprocess
          0i64,
          a1,
          0i64,
          0i64,
          0,
          a4 != 0 ? 0x8000000 : 0,
          0i64,
          0i64,
          v18,
          &v8) )
    return 0;
```

Figure 5-4 Release file 3. dat and execute3

The dll file obtains the system time, decrypts and assembles the command, creates the process to execute the command. The function is to add 3. dat to a scheduled task using schtasks. exe with the task name set to QQQ.

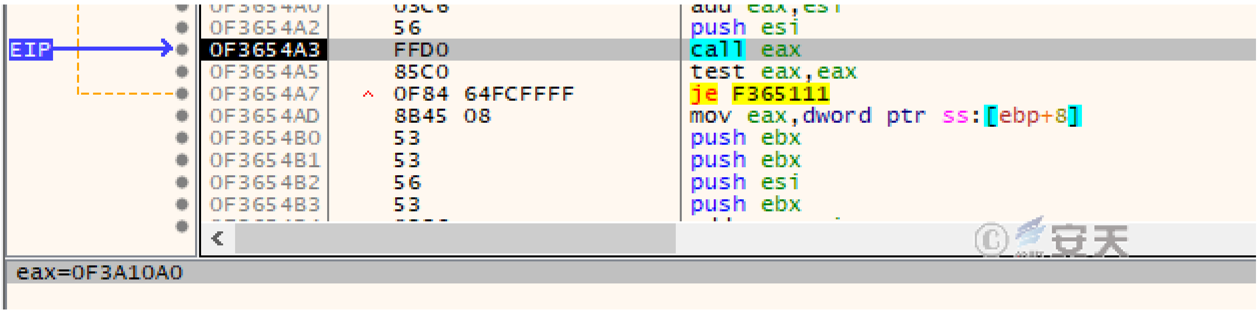


```
(*(void (__fastcall **)(__int128 *))(qword_1800223A8 + 464))(&v57);// <kernel32.GetLocalTime>
v48 = (unsigned __int16)(WORD5(v57) + 13);
LODWORD(v21) = (unsigned __int16)(WORD5(v57) + 2);
LODWORD(v32) = WORD4(v57) + (unsigned __int16)(WORD5(v57) + 2) / 0x3Cu;
LODWORD(v58) = WORD4(v57) + (unsigned __int16)v48 / 0x3Cu;
v59 = 0x51005100511i64;
v61 = 0i64;
v60 = 0i64;
v62 = decrypt_str1(v48, (2290649225u * (unsigned __int64)(unsigned __int16)v48) >> 32, v48, v58, 46);// schtasks /Create /ST %02u:%02u /SC ONCE /tr \"%s\" /tn \"%s\" /I /ET %02u:%02u
```

Figure 55 Create Scheduled Task Implementation Self-Startup 5-4

5.4 Layer 3 code - 3.dat

The dat file is an XLL file, which is executed by calling xlAutoOpen function through rundll32.exe. Similar to Agreement _ 487989a _ Mar4.xll, the xlAutoOpen function decrypts the loader and a dll file by using several rounds of XOR. The loader expands the dll file in memory, modifies the relocation table, modifies the export table, and jumps to the dll export function. different from the original sample, the loader of the original sample has a lot of confusion. Will decrypt the pe file dump under the name Qbot. dll, the follow-up analysis report will use this name.



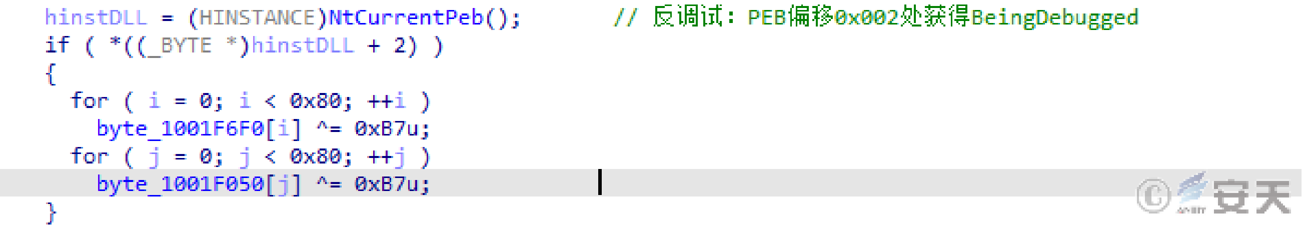
```
EIP  0F3654A2    56              push esi
     0F3654A3    FFD0            call eax
     0F3654A5    85C0            test eax,eax
     0F3654A7  ^ 0F84 64FCFFFF   je F365111
     0F3654AD    8B45 08         mov eax,dword ptr ss:[ebp+8]
     0F3654B0    53              push ebx
     0F3654B1    53              push ebx
     0F3654B2    56              push esi
     0F3654B3    53              push ebx

eax=0F3A10A0
```

Figure 5-6 Executing a decrypted dll file 5

5.5 Layer 4 code - Qbot.dll

The Qbot.dll file mainly performs the following functions: Anti-debugging, obtaining system information, querying anti-virus software process, restarting the process, and injecting its own code into the kernel module of wermgr.exe process execution.

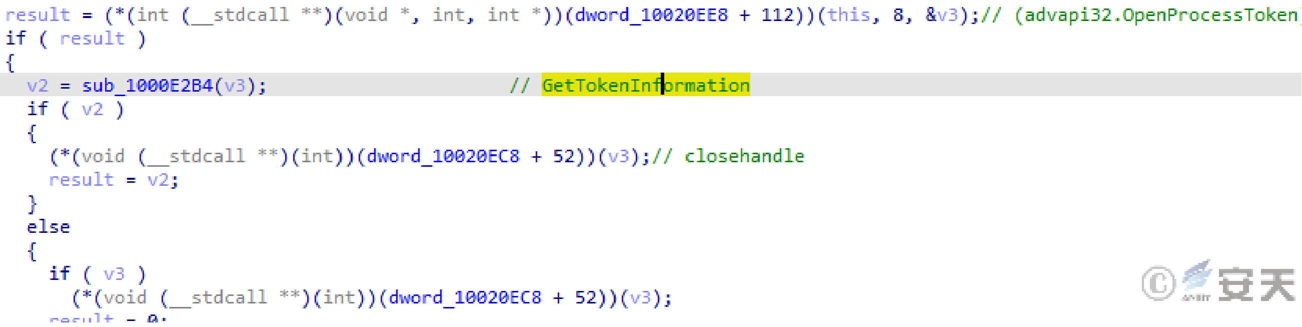
Get the peb through NtCurrentPeb and use the BeingDebugged member for anti-debugging. If there is a debugger, the key is modified to affect the decryption function of the sample.



```
hinstDLL = (HINSTANCE)NtCurrentPeb();       // 反调试：PEB偏移0x002处获得BeingDebugged
if ( *((_BYTE *)hinstDLL + 2) )
{
  for ( i = 0; i < 0x80; ++i )
    byte_1001F6F0[i] ^= 0xB7u;
  for ( j = 0; j < 0x80; ++j )
    byte_1001F050[j] ^= 0xB7u;
}
```

Figure 5-7 Debug with BeingDebugged6

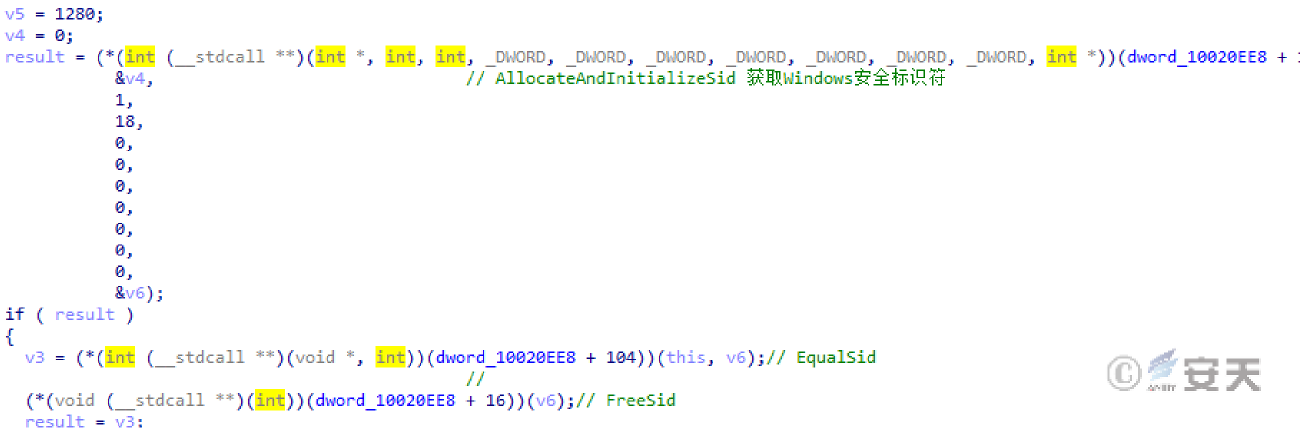
The Qbot bank trojan obtains the group membership in the access token.



```
result = (*(int (__stdcall **)(void *, int, int *))(dword_10020EE8 + 112))(this, 8, &v3);// (advapi32.OpenProcessToken
if ( result )
{
  v2 = sub_1000E2B4(v3);                    // GetTokenInformation
  if ( v2 )
  {
    (*(void (__stdcall **)(int))(dword_10020EC8 + 52))(v3);// closehandle
    result = v2;
  }
  else
  {
    if ( v3 )
      (*(void (__stdcall **)(int))(dword_10020EC8 + 52))(v3);
```

Fig. 58 obtains the group membership in the access token 5-7

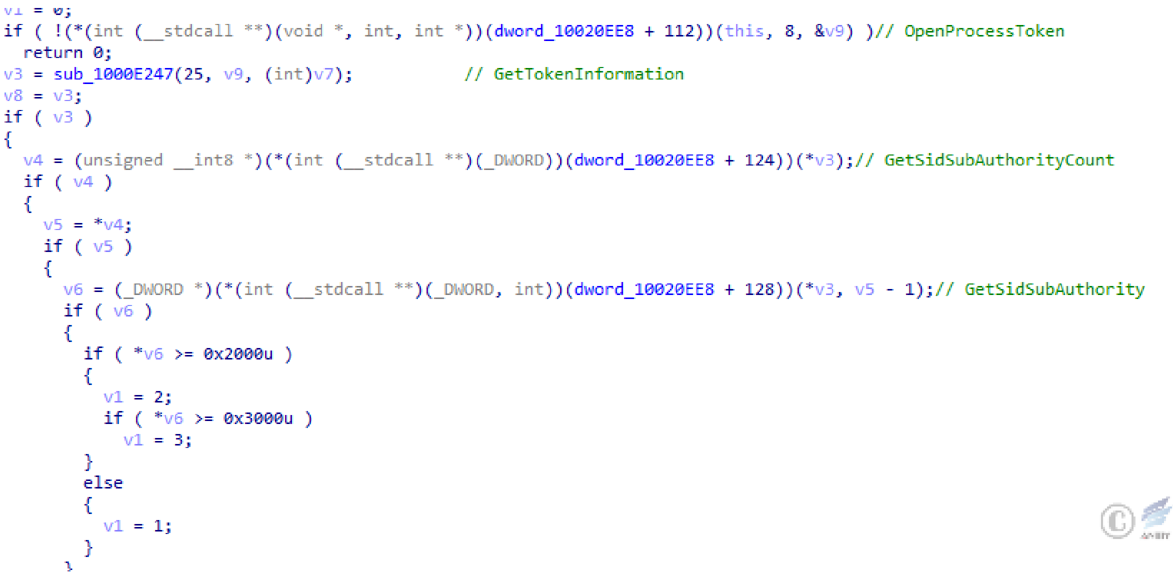
The Qbot bank trojan horse judges whether the current process is the administrator authority.



```
v5 = 1280;
v4 = 0;
result = (*(int (__stdcall **)(int *, int, int, _DWORD, _DWORD, _DWORD, _DWORD, _DWORD, _DWORD, _DWORD, int *))(dword_10020EE8 + :
           &v4,                                  // AllocateAndInitializeSid 获取Windows安全标识符
           1,
           18,
           0,
           0,
           0,
           0,
           0,
           0,
           0,
           &v6);
if ( result )
{
  v3 = (*(int (__stdcall **)(void *, int))(dword_10020EE8 + 104))(this, v6);// EqualSid
                                                 //
  (*(void (__stdcall **)(int))(dword_10020EE8 + 16))(v6);// FreeSid
  result = v3:
```

Figure 5-9 Determine whether the current process is under the administrator authority 8

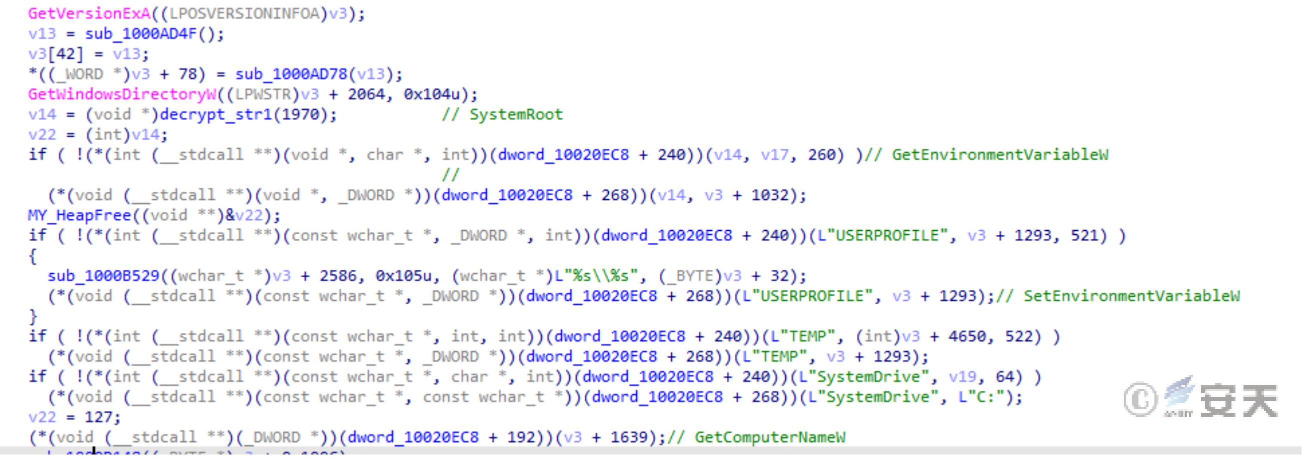
The Qbot bank Trojan horse obtains the RID information of the process, if RID < 0x2000, it means that it is not trusted or has low integrity.



```
v1 = 0;
if ( !(*(int (__stdcall **)(void *, int, int *))(dword_10020EE8 + 112))(this, 8, &v9) )// OpenProcessToken
  return 0;
v3 = sub_1000E247(25, v9, (int)v7);           // GetTokenInformation
v8 = v3;
if ( v3 )
{
  v4 = (unsigned __int8 *)(*(int (__stdcall **)(_DWORD))(dword_10020EE8 + 124))(*v3);// GetSidSubAuthorityCount
  if ( v4 )
  {
    v5 = *v4;
    if ( v5 )
    {
      v6 = (_DWORD *)(*(int (__stdcall **)(_DWORD, int))(dword_10020EE8 + 128))(*v3, v5 - 1);// GetSidSubAuthority
      if ( v6 )
      {
        if ( *v6 >= 0x2000u )
        {
          v1 = 2;
          if ( *v6 >= 0x3000u )
            v1 = 3;
        }
        else
        {
          v1 = 1;
        }
      }
```

Figure 5-10 Acquiring the RID graph 9

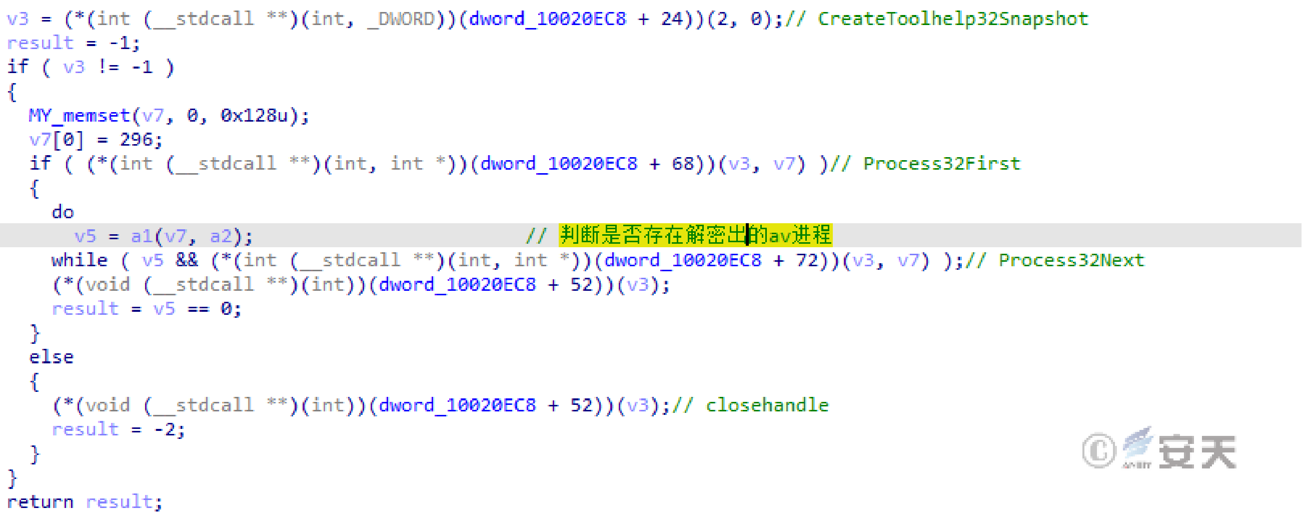
When RID < 0x2000, the Qbot bank Trojan obtains the version information, environment variable information, computer name and other system information.



```
GetVersionExA((LPOSVERSIONINFOA)v3);
v13 = sub_1000AD4F();
v3[42] = v13;
*((_WORD *)v3 + 78) = sub_1000AD78(v13);
GetWindowsDirectoryW((LPWSTR)v3 + 2064, 0x104u);
v14 = (void *)decrypt_str1(1970);           // SystemRoot
v22 = (int)v14;
if ( !(*(int (__stdcall **)(void *, char *, int))(dword_10020EC8 + 240))(v14, v17, 260) )// GetEnvironmentVariableW
                                            //
  (*(void (__stdcall **)(void *, _DWORD *))(dword_10020EC8 + 268))(v14, v3 + 1032);
MY_HeapFree((void **)&v22);
if ( !(*(int (__stdcall **)(const wchar_t *, _DWORD *, int))(dword_10020EC8 + 240))(L"USERPROFILE", v3 + 1293, 521) )
{
  sub_1000B529((wchar_t *)v3 + 2586, 0x105u, (wchar_t *)L"%s\\%s", (_BYTE)v3 + 32);
  (*(void (__stdcall **)(const wchar_t *, _DWORD *))(dword_10020EC8 + 268))(L"USERPROFILE", v3 + 1293);// SetEnvironmentVariableW
}
if ( !(*(int (__stdcall **)(const wchar_t *, int, int))(dword_10020EC8 + 240))(L"TEMP", (int)v3 + 4650, 522) )
  (*(void (__stdcall **)(const wchar_t *, _DWORD *))(dword_10020EC8 + 268))(L"TEMP", v3 + 1293);
if ( !(*(int (__stdcall **)(const wchar_t *, char *, int))(dword_10020EC8 + 240))(L"SystemDrive", v19, 64) )
  (*(void (__stdcall **)(const wchar_t *, const wchar_t *))(dword_10020EC8 + 268))(L"SystemDrive", L"C:");
v22 = 127;
(*(void (__stdcall **)(_DWORD *))(dword_10020EC8 + 192))(v3 + 1639);// GetComputerNameW
```

Figure 5-11 Obtain the system information graph10

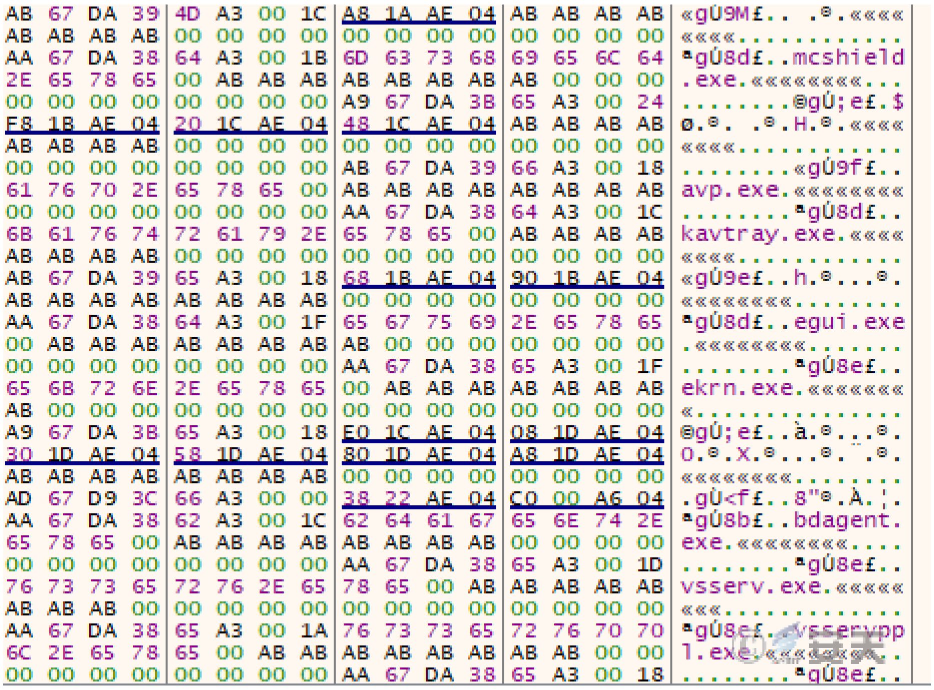
The Qbot bank Trojan enumerates the current system process and inquires whether there are anti-virus software processes such as ccSvcHst.exe, NortonSecurity.exe, nsWscc.exe, avgcsrvx.exe, avgsvcx.exe.



```
v3 = (*(int (__stdcall **)(int, _DWORD))(dword_10020EC8 + 24))(2, 0);// CreateToolhelp32Snapshot
result = -1;
if ( v3 != -1 )
{
  MY_memset(v7, 0, 0x128u);
  v7[0] = 296;
  if ( (*(int (__stdcall **)(int, int *))(dword_10020EC8 + 68))(v3, v7) )// Process32First
  {
    do
      v5 = a1(v7, a2);                        // 判断是否存在解密出的av进程
    while ( v5 && (*(int (__stdcall **)(int, int *))(dword_10020EC8 + 72))(v3, v7) );// Process32Next
    (*(void (__stdcall **)(int))(dword_10020EC8 + 52))(v3);
    result = v5 == 0;
  }
  else
  {
    (*(void (__stdcall **)(int))(dword_10020EC8 + 52))(v3);// closehandle
    result = -2;
  }
}
return result;
```

Figure 5-12 Query antivirus software process11

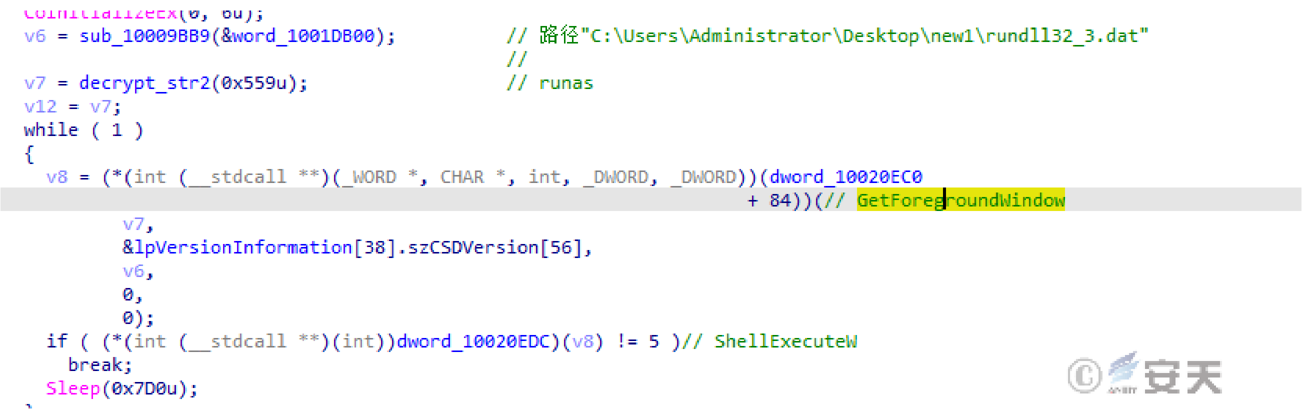
Qbot bank Trojan decrypts the anti-virus software process string.



```
AB 67 DA 39 | 4D A3 00 1C | A8 1A AE 04 | AB AB AB AB | «gÚ9M£.. .®.««««
AB AB AB AB | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | ««««............
AA 67 DA 38 | 64 A3 00 1B | 6D 63 73 68 | 69 65 6C 64 | ªgÚ8d£..mcshield
2E 65 78 65 | 00 AB AB AB | AB AB AB AB | AB 00 00 00 | .exe.««««««««...
00 00 00 00 | 00 00 00 00 | A9 67 DA 3B | 65 A3 00 24 | ........©gÚ;e£.$
F8 1B AE 04 | 20 1C AE 04 | 48 1C AE 04 | AB AB AB AB | ø.®. .®.H.®.««««
AB AB AB AB | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | ««««............
00 00 00 00 | 00 00 00 00 | AB 67 DA 39 | 66 A3 00 18 | ........«gÚ9f£..
61 76 70 2E | 65 78 65 00 | AB AB AB AB | AB AB AB AB | avp.exe.««««««««
00 00 00 00 | 00 00 00 00 | AA 67 DA 38 | 64 A3 00 1C | ........ªgÚ8d£..
6B 61 76 74 | 72 61 79 2E | 65 78 65 00 | AB AB AB AB | kavtray.exe.««««
AB AB AB AB | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | ««««............
AB 67 DA 39 | 65 A3 00 18 | 68 1B AE 04 | 90 1B AE 04 | «gÚ9e£..h.®...®.
AB AB AB AB | AB AB AB AB | 00 00 00 00 | 00 00 00 00 | ««««««««........
AA 67 DA 38 | 64 A3 00 1F | 65 67 75 69 | 2E 65 78 65 | ªgÚ8d£..egui.exe
00 AB AB AB | AB AB AB AB | AB 00 00 00 | 00 00 00 00 | .««««««««.......
00 00 00 00 | 00 00 00 00 | AA 67 DA 38 | 65 A3 00 1F | ........ªgÚ8e£..
65 6B 72 6E | 2E 65 78 65 | 00 AB AB AB | AB AB AB AB | ekrn.exe.«««««««
AB 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | «...............
A9 67 DA 3B | 65 A3 00 18 | E0 1C AE 04 | 08 1D AE 04 | ©gÚ;e£..à.®...®.
30 1D AE 04 | 58 1D AE 04 | 80 1D AE 04 | A8 1D AE 04 | 0.®.X.®...®.¨.®.
AB AB AB AB | AB AB AB AB | 00 00 00 00 | 00 00 00 00 | ««««««««........
AD 67 D9 3C | 66 A3 00 00 | 38 22 AE 04 | C0 00 A6 04 | .gÙ<f£..8"®.À.¦.
AA 67 DA 38 | 62 A3 00 1C | 62 64 61 67 | 65 6E 74 2E | ªgÚ8b£..bdagent.
65 78 65 00 | AB AB AB AB | AB AB AB AB | 00 00 00 00 | exe.««««««««....
00 00 00 00 | 00 00 00 00 | AA 67 DA 38 | 65 A3 00 1D | ........ªgÚ8e£..
76 73 73 65 | 72 76 2E 65 | 78 65 00 AB | AB AB AB AB | vsserv.exe.«««««
AB AB AB 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | «««.............
AA 67 DA 38 | 65 A3 00 1A | 76 73 73 65 | 72 76 70 70 | ªgÚ8e£..vsservpp
6C 2E 65 78 | 65 00 AB AB | AB AB AB AB | AB AB 00 00 | l.exe.««««««««..
00 00 00 00 | 00 00 00 00 | AA 67 DA 38 | 65 A3 00 18 | ........ªgÚ8e£..
```

Figure 5-13 Antivirus software process string

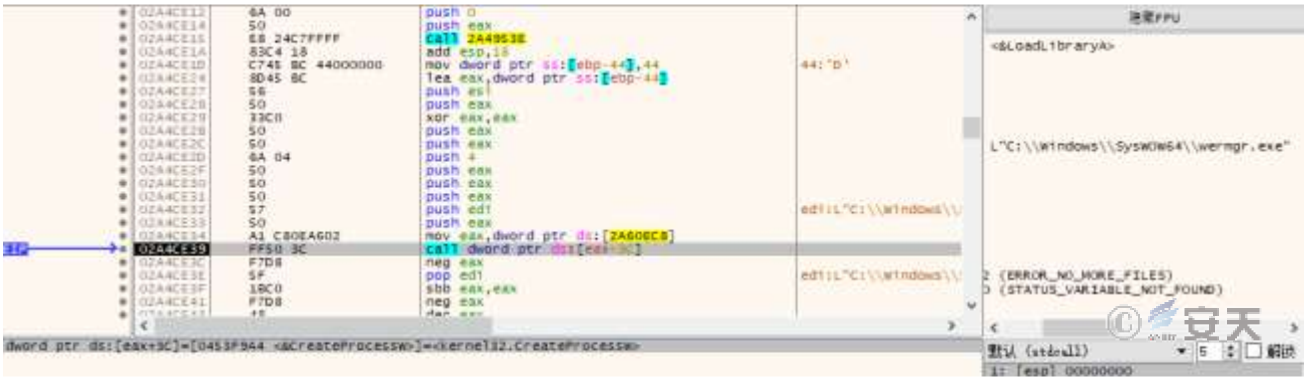
The Qbot bank Trojan detects its own running permission, obtains a handle to the current window of the administrator permission, and restarts the process with the administrator permission.



```
CoInitializeEx(0, 6u);
v6 = sub_10009BB9(&word_1001DB00);          // 路径"C:\Users\Administrator\Desktop\new1\rundll32_3.dat"
                                            //
v7 = decrypt_str2(0x559u);                  // runas
v12 = v7;
while ( 1 )
{
  v8 = (*(int (__stdcall **)(_WORD *, CHAR *, int, _DWORD, _DWORD))(dword_10020EC0
                                                                  + 84))(// GetForegroundWindow
         v7,
         &lpVersionInformation[38].szCSDVersion[56],
         v6,
         0,
         0);
  if ( (*(int (__stdcall **)(int))dword_10020EDC)(v8) != 5 )// ShellExecuteW
    break;
  Sleep(0x7D0u);
}
```

Figure 5-14 Check the self-run permission 12

The Qbot banking trojan horse creates the wermgr. exe process in a suspended manner.

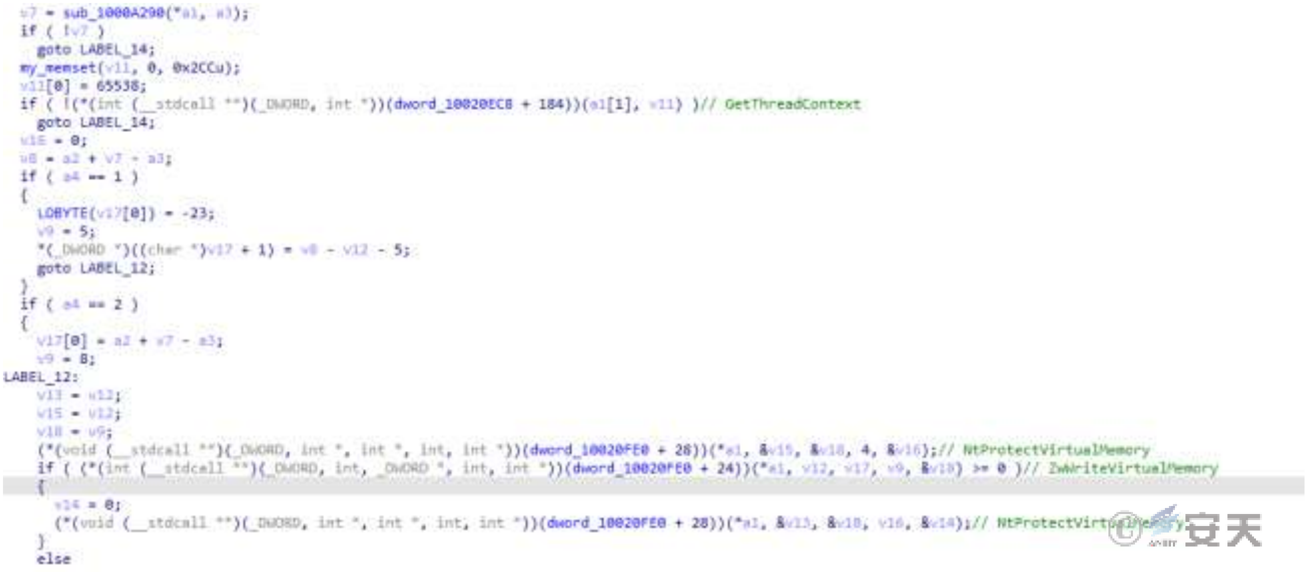


```
02A4CE12   6A 00            push 0
02A4CE14   50               push eax
02A4CE15   E8 24C7FFFF      call 2A4953E
02A4CE1A   83C4 18          add esp,18
02A4CE1D   C745 BC 44000000 mov dword ptr ss:[ebp-44],44          44:'D'
02A4CE24   8D45 BC          lea eax,dword ptr ss:[ebp-44]
02A4CE27   56               push esi
02A4CE28   50               push eax
02A4CE29   33C0             xor eax,eax
02A4CE2B   50               push eax
02A4CE2C   50               push eax
02A4CE2D   6A 04            push 4
02A4CE2F   50               push eax
02A4CE30   50               push eax
02A4CE31   50               push eax
02A4CE32   57               push edi                              edi:L"C:\\Windows\\
02A4CE33   50               push eax
02A4CE34   A1 C80EA602      mov eax,dword ptr ds:[2A60EC8]
02A4CE39   FF50 3C          call dword ptr ds:[eax+3C]
02A4CE3C   F7D8             neg eax
02A4CE3E   5F               pop edi                               edi:L"C:\\Windows\\
02A4CE3F   1BC0             sbb eax,eax
02A4CE41   F7D8             neg eax
           48               dec eax
dword ptr ds:[eax+3C]=[0453F944 <&CreateProcessW>]=<kernel32.CreateProcessW>
```

```
<&LoadLibraryA>
L"C:\\Windows\\SysWOW64\\wermgr.exe"
2 (ERROR_NO_MORE_FILES)
0 (STATUS_VARIABLE_NOT_FOUND)
默认 (stdcall)   5   □ 解锁
1: [esp] 00000000
```

Figure 5-15 Creating the wermgr. exe process in a suspended manner 13

The Qbot bank Trojan horse injects itself into the process of wermgr. exe, modifies the relocation table, and calls the GetThreadContext function to obtain the address of the entry function.

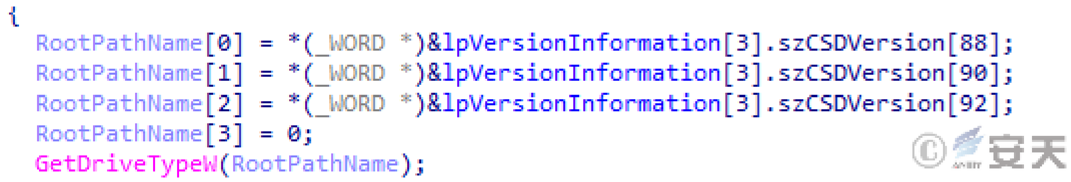


```
  v7 = sub_1000A290(*a1, a3);
  if ( !v7 )
    goto LABEL_14;
  my_memset(v11, 0, 0x2CCu);
  v11[0] = 65538;
  if ( !(*(int (__stdcall **)(_DWORD, int *))(dword_10020EC8 + 184))(a1[1], v11) )// GetThreadContext
    goto LABEL_14;
  v16 = 0;
  v8 = a2 + v7 - a3;
  if ( a4 == 1 )
  {
    LOBYTE(v17[0]) = -23;
    v9 = 5;
    *(_DWORD *)((char *)v17 + 1) = v8 - v12 - 5;
    goto LABEL_12;
  }
  if ( a4 == 2 )
  {
    v17[0] = a2 + v7 - a3;
    v9 = 8;
LABEL_12:
    v13 = v12;
    v15 = v12;
    v18 = v9;
    (*(void (__stdcall **)(_DWORD, int *, int *, int, int *))(dword_10020FE0 + 28))(*a1, &v15, &v18, 4, &v16);// NtProtectVirtualMemory
    if ( (*(int (__stdcall **)(_DWORD, int, _DWORD *, int, int *))(dword_10020FE0 + 24))(*a1, v12, v17, v9, &v18) >= 0 )// ZwWriteVirtualMemory
    {
      v14 = 0;
      (*(void (__stdcall **)(_DWORD, int *, int *, int, int *))(dword_10020FE0 + 28))(*a1, &v13, &v18, v16, &v14);// NtProtectVirtualMemory
    }
    else
```

Figure 5-16 Obtaining the Context structure14

The wemgr.exe process mainly implements the following functions: Obtaining disk drive information, creating multiple sub-threads and setting the priority of these sub-threads to be lower than normal, anti-debugging, dynamically obtaining encryption-related functions, screen shots, Create named pipe and monitor, establish connection with C2 server and return data functions.

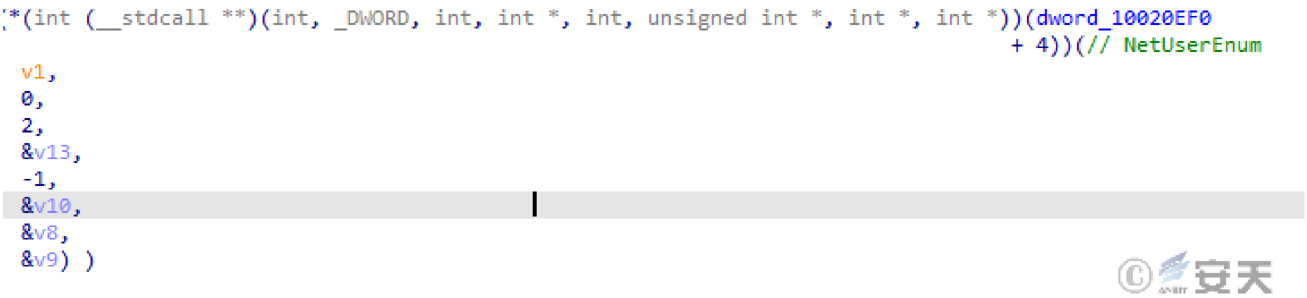
The wermgr. exe process gets the disk drive information.



```
{
  RootPathName[0] = *(_WORD *)&lpVersionInformation[3].szCSDVersion[88];
  RootPathName[1] = *(_WORD *)&lpVersionInformation[3].szCSDVersion[90];
  RootPathName[2] = *(_WORD *)&lpVersionInformation[3].szCSDVersion[92];
  RootPathName[3] = 0;
  GetDriveTypeW(RootPathName);
```

Figure 5-17 Get disk drive information 15

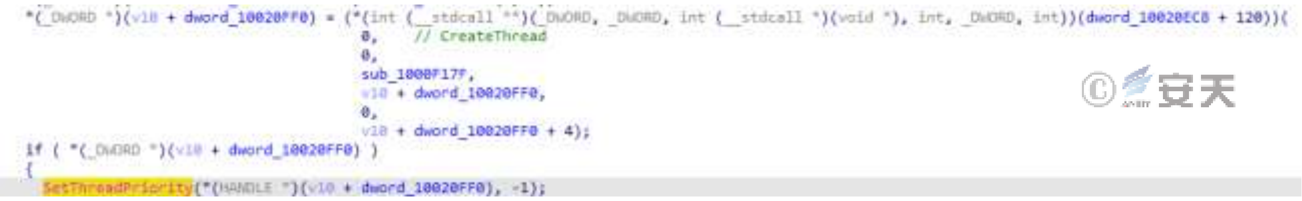
The wermgr. exe process obtains all account names in the system.



```
'*(int (__stdcall **)(int, _DWORD, int, int *, int, unsigned int *, int *, int *))(dword_10020EF0
                                                                              + 4))(// NetUserEnum
  v1,
  0,
  2,
  &v13,
  -1,
  &v10,
  &v8,
  &v9) )
```

Figure 5-18 Obtain all account names in the system 16

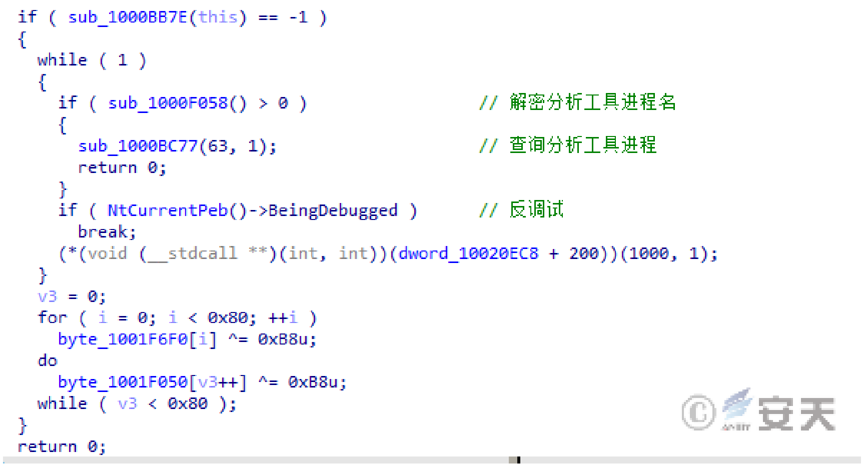
The wermgr. exe process creates a child thread, and when the child thread starts, sets the thread priority to a lower than normal value.



```
*(_DWORD *)(v18 + dword_10020FF0) = (*(int (__stdcall **)(_DWORD, _DWORD, int (__stdcall *)(void *), int, _DWORD, int))(dword_10020EC8 + 120))(
                                      0,     // CreateThread
                                      0,
                                      sub_1000F17F,
                                      v18 + dword_10020FF0,
                                      0,
                                      v18 + dword_10020FF0 + 4);
if ( *(_DWORD *)(v18 + dword_10020FF0) )
{
  SetThreadPriority(*(HANDLE *)(v18 + dword_10020FF0), -1);
```

Figure 5-19 Creating a child thread 17

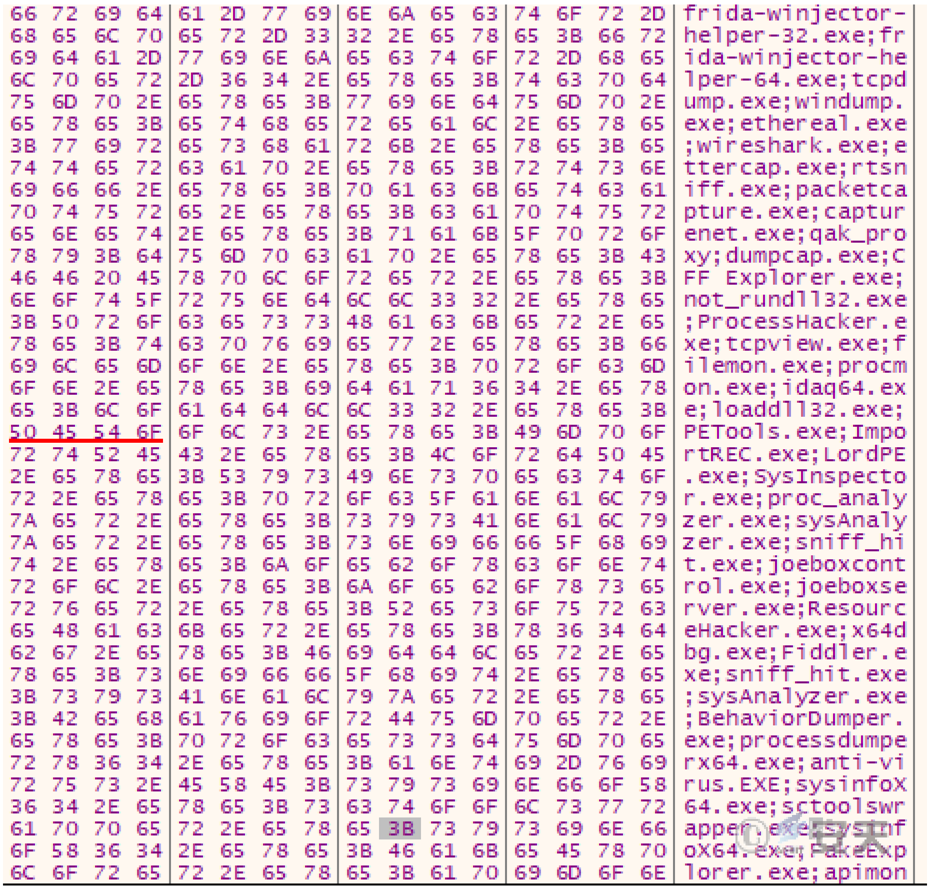
The wermgr. exe process enumerates the current system process and queries whether there are analysis tools such as Fiddler.exe, Autoruns.exe. Determining whether there is a debugger, and performing an exclusive OR operation on the key of the decryption algorithm if there is a debugger.



```
if ( sub_1000BB7E(this) == -1 )
{
  while ( 1 )
  {
    if ( sub_1000F058() > 0 )               // 解密分析工具进程名
    {
      sub_1000BC77(63, 1);                  // 查询分析工具进程
      return 0;
    }
    if ( NtCurrentPeb()->BeingDebugged )    // 反调试
      break;
    (*(void (__stdcall **)(int, int))(dword_10020EC8 + 200))(1000, 1);
  }
  v3 = 0;
  for ( i = 0; i < 0x80; ++i )
    byte_1001F6F0[i] ^= 0xB8u;
  do
    byte_1001F050[v3++] ^= 0xB8u;
  while ( v3 < 0x80 );
}
return 0;
```

Figure 5-20 Query analysis tool process, anti-debug 18

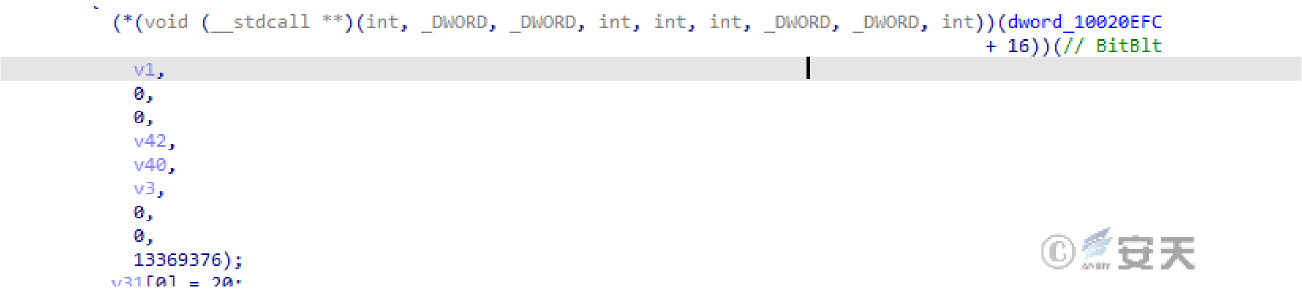
The wemgr.exe process decrypts the parser process string.



```
66 72 69 64 61 2D 77 69 6E 6A 65 63 74 6F 72 2D frida-winjector-
68 65 6C 70 65 72 2D 33 32 2E 65 78 65 3B 66 72 helper-32.exe;fr
69 64 61 2D 77 69 6E 6A 65 63 74 6F 72 2D 68 65 ida-winjector-he
6C 70 65 72 2D 36 34 2E 65 78 65 3B 74 63 70 64 lper-64.exe;tcpd
75 6D 70 2E 65 78 65 3B 77 69 6E 64 75 6D 70 2E ump.exe;windump.
65 78 65 3B 65 74 68 65 72 65 61 6C 2E 65 78 65 exe;ethereal.exe
3B 77 69 72 65 73 68 61 72 6B 2E 65 78 65 3B 65 ;wireshark.exe;e
74 74 65 72 63 61 70 2E 65 78 65 3B 72 74 73 6E ttercap.exe;rtsn
69 66 66 2E 65 78 65 3B 70 61 63 6B 65 74 63 61 iff.exe;packetca
70 74 75 72 65 2E 65 78 65 3B 63 61 70 74 75 72 pture.exe;captur
65 6E 65 74 2E 65 78 65 3B 71 61 6B 5F 70 72 6F enet.exe;qak_pro
78 79 3B 64 75 6D 70 63 61 70 2E 65 78 65 3B 43 xy;dumpcap.exe;C
46 46 20 45 78 70 6C 6F 72 65 72 2E 65 78 65 3B FF Explorer.exe;
6E 6F 74 5F 72 75 6E 64 6C 6C 33 32 2E 65 78 65 not_rundll32.exe
3B 50 72 6F 63 65 73 73 48 61 63 6B 65 72 2E 65 ;ProcessHacker.e
78 65 3B 74 63 70 76 69 65 77 2E 65 78 65 3B 66 xe;tcpview.exe;f
69 6C 65 6D 6F 6E 2E 65 78 65 3B 70 72 6F 63 6D ilemon.exe;procm
6F 6E 2E 65 78 65 3B 69 64 61 71 36 34 2E 65 78 on.exe;idaq64.ex
65 3B 6C 6F 61 64 64 6C 6C 33 32 2E 65 78 65 3B e;loaddll32.exe;
50 45 54 6F 6F 6C 73 2E 65 78 65 3B 49 6D 70 6F PETools.exe;Impo
72 74 52 45 43 2E 65 78 65 3B 4C 6F 72 64 50 45 rtREC.exe;LordPE
2E 65 78 65 3B 53 79 73 49 6E 73 70 65 63 74 6F .exe;SysInspecto
72 2E 65 78 65 3B 70 72 6F 63 5F 61 6E 61 6C 79 r.exe;proc_analy
7A 65 72 2E 65 78 65 3B 73 79 73 41 6E 61 6C 79 zer.exe;sysAnaly
7A 65 72 2E 65 78 65 3B 73 6E 69 66 66 5F 68 69 zer.exe;sniff_hi
74 2E 65 78 65 3B 6A 6F 65 62 6F 78 63 6F 6E 74 t.exe;joeboxcont
72 6F 6C 2E 65 78 65 3B 6A 6F 65 62 6F 78 73 65 rol.exe;joeboxse
72 76 65 72 2E 65 78 65 3B 52 65 73 6F 75 72 63 rver.exe;Resourc
65 48 61 63 6B 65 72 2E 65 78 65 3B 78 36 34 64 eHacker.exe;x64d
62 67 2E 65 78 65 3B 46 69 64 64 6C 65 72 2E 65 bg.exe;Fiddler.e
78 65 3B 73 6E 69 66 66 5F 68 69 74 2E 65 78 65 xe;sniff_hit.exe
3B 73 79 73 41 6E 61 6C 79 7A 65 72 2E 65 78 65 ;sysAnalyzer.exe
3B 42 65 68 61 76 69 6F 72 44 75 6D 70 65 72 2E ;BehaviorDumper.
65 78 65 3B 70 72 6F 63 65 73 73 64 75 6D 70 65 exe;processdumpe
72 78 36 34 2E 65 78 65 3B 61 6E 74 69 2D 76 69 rx64.exe;anti-vi
72 75 73 2E 45 58 45 3B 73 79 73 69 6E 66 6F 58 rus.EXE;sysinfoX
36 34 2E 65 78 65 3B 73 63 74 6F 6F 6C 73 77 72 64.exe;sctoolswr
61 70 70 65 72 2E 65 78 65 3B 73 79 73 69 6E 66 apper.exe;sysinf
6F 58 36 34 2E 65 78 65 3B 46 61 6B 65 45 78 70 oX64.exe;FakeExp
6C 6F 72 65 72 2E 65 78 65 3B 61 70 69 6D 6F 6E lorer.exe;apimon
```

Figure 5-21 Analysis of the tool process string19

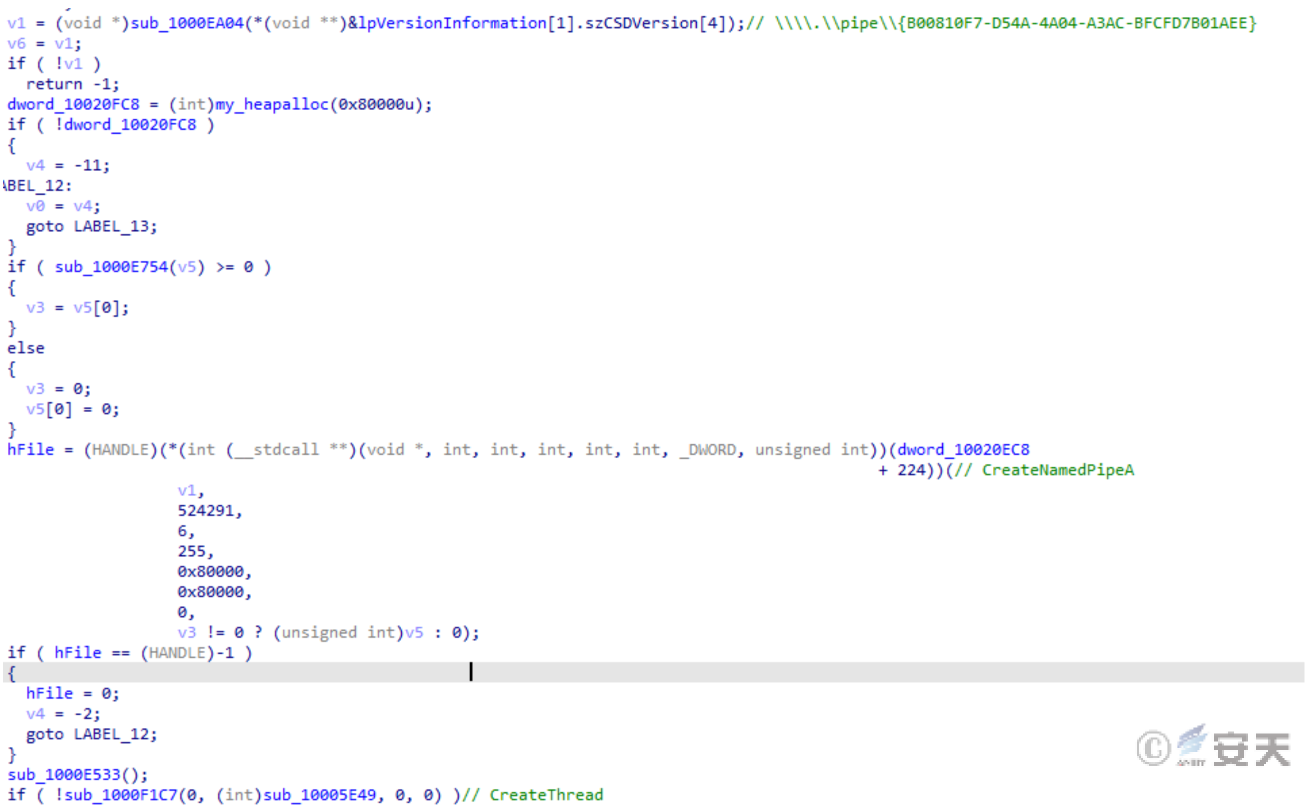
The wermgr. exe process uses APIs such as BitBlt to get screenshots.



```
  (*(void (__stdcall **)(int, _DWORD, _DWORD, int, int, int, _DWORD, _DWORD, int))(dword_10020EFC
                                                                                  + 16))(// BitBlt
    v1,
    0,
    0,
    v42,
    v40,
    v3,
    0,
    0,
    13369376);
  v31[0] = 20;
```

Figure 5-22 A screenshot of the acquisition 20

The wermgr. exe process creates a named pipe\\.\ pipe\\\% ssp that, when connected, creates child threads to monitor and process data.



```
v1 = (void *)sub_1000EA04(*(void **)&lpVersionInformation[1].szCSDVersion[4]);// \\\\.\\pipe\\{B00810F7-D54A-4A04-A3AC-BFCFD7B01AEE}
v6 = v1;
if ( !v1 )
  return -1;
dword_10020FC8 = (int)my_heapalloc(0x80000u);
if ( !dword_10020FC8 )
{
  v4 = -11;
\BEL_12:
  v0 = v4;
  goto LABEL_13;
}
if ( sub_1000E754(v5) >= 0 )
{
  v3 = v5[0];
}
else
{
  v3 = 0;
  v5[0] = 0;
}
hFile = (HANDLE)(*(int (__stdcall **)(void *, int, int, int, int, int, _DWORD, unsigned int))(dword_10020EC8
                                                                                            + 224))(// CreateNamedPipeA

                  v1,
                  524291,
                  6,
                  255,
                  0x80000,
                  0x80000,
                  0,
                  v3 != 0 ? (unsigned int)v5 : 0);
if ( hFile == (HANDLE)-1 )
{
  hFile = 0;
  v4 = -2;
  goto LABEL_12;
}
sub_1000E533();
if ( !sub_1000F1C7(0, (int)sub_10005E49, 0, 0) )// CreateThread
```

Figure 5-23 Creating a named pipe21

The wermgr. exe process decrypts the following IP addresses and ports from the resource to build the communication tunnel.

Table 5-1 IP addresses and ports after decryption 1

Ip address	Ip address	Ip address
92.239.81.124: 443	176.202.46.81: 443	2.49.58.47: 2222
74.66.134.24: 443	213.31.90.183: 2222	12.172.173.82: 50001
70.53.96.223: 995	92.154.45.81: 2222	186.67.54: 443
190.191.35.122: 443	68.173.170.110: 8443	12.172.173.82: 993
12.172.173.82: 22	37.186.55.60: 2222	84.216.198.124: 6881
94.30.98.134: 32,100	78.196.246.32: 443	12.172.173.82: 995
173.18.126.3: 443	201.244.108.183: 995	24.178.201.230: 2222
151.65.134.135: 443	197.14.148.149: 443	197.244.108.123: 443
86.130.9.213: 2222	190.75.139.66: 2222	213.67.255.57: 2222
189.222.53.217: 443	122.184.143.84: 443	92.159.173.52: 2222
91.68.227.219: 443	86.236.114.212: 2222	80.12.88.148: 2222

73.36.196.11: 443	47.196.225.236: 443	65.95.49.237: 2222
184.176.35.223: 2222	186.48.181.17: 995	2.14.105.160: 2222
190.218.125.145: 443	109.11.175.42: 2222	23.251.92.171: 2222
75.156.125.215: 995	184.189.41.80: 443	31.48.18.52: 443
70.51.152.61: 2222	47.203.229.168: 443	104.35.24.154: 443
92.154.17.149: 2222	103.169.83.89: 443	86.169.103.3: 443
92.1.170.110: 995	183.87.163.165: 443	85.241.180.94: 443
92.20.204.198: 2222	103.141.50.102: 995	81.229.117.95: 2222
47.34.30.133: 443	173.178.151.233: 443	47.16.77.194: 2222
76.80.180.154: 995	67.70.23.222: 2222	24.117.237.157: 443
87.202.101.164: 50000	64.237.245.195: 443	103.231.216.238: 443
103.71.21.107: 443	71.65.145.108: 443	12.172.173.82: 465
184.153.132.82: 443	86.178.33.20: 2222	94.200.183.66: 2222
98.159.33.25: 443	136.35.241.159: 443	24.187.145.201: 2222
65.94.87.200: 2222	184.176.110.61: 61202	49.245.82.178: 2222
46.10.198.134: 443	84.35.26.14: 995	103.252.7.231: 443
139.5.239.14: 443	202.142.98.62: 443	27.109.19.90: 2078
75.143.236.149: 443	50.68.204.71: 993	91.169.12.198: 32,100
24.239.69.244: 443	12.172.173.82: 21	174.104.184.149: 443
86.225.214.138: 2222	202.187.87.178: 995	81.158.112.20: 2222
98.145.23.67: 443	73.161.176.218: 443	88.122.133.88: 32,100
76.27.40.189: 443	201.137.185.109: 443	90.104.22.28: 2222
178.175.187.254: 443	12.172.173.82: 2087	208.180.17.32: 2222
196.70.212.80: 443	103.12.133.134: 2222	190.28.116.106: 443
92.27.86.48: 2222	76.170.252.153: 995	50.68.204.71: 995
83.92.85.93: 443	35.143.97.145: 995	74.93.148.97: 995
72.80.7.6: 50003	70.55.187.152: 2222	72.88.245.71: 443
12.172.173.82: 32101	187.199.103.21: 32103	86.190.223.11: 2222
88.126.94.4: 50000	116.72.250.18: 443	

6 Summary

Since Microsoft announced in February 2023 that macros in Office documents have been blocked by default, attackers have turned to XLL files as a new medium for distributing malicious files. The attacker sends spam to the user, induces the user to open the XLL file in the attachment to execute malicious code, thus running bank Trojan, remote control Trojan and other malicious software on the user host.

It is suggested that the user should not easily believe the contents in the unknown mail, confirm the source of the mail, and be alert to the guiding contents in the mail. Antiy CERT will continue to pay attention to the new attack methods of attackers, and conduct in-depth analysis and research on related attack activities.

Appendix I: IoCs

IoCs
20746c3bb01aa4deeea993824f947194d
160d6d1be068c04fcf08553383f1c93a
Ff58f9cf0740aead678d9e36c0782894
84a765f683860eedbb344a9a1aa0c883
5d450b19aa1a0fd9ae4103fa84d5d09b
5ac0d9286d8497c648dfc418218397eb
E09a3bac10565ee80cbdb7a4b1a5d2af

Appendix II: References

[1]. Qbot Bank Trojan 2020 Activity Analysis Report

Https: / / www.antiy.cn / research / notice & report / research _ report / 20210206.html

[2]. Threat Spotlight: Xlling in Excel - threat actors using malicious add - ins

Https: / / blog.talosintelligence.com / xlling-in-excel-malicious-add-ins /

Appendix III: About Antiy

Antiy is committed to enhancing the network security defense capabilities of its customers and effectively responding to security threats. Through more than 20 years of independent research and development, Antiy has developed technological leadership in areas such as threat detection engines, advanced threat countermeasures, and large-scale threat automation analysis.

Antiy has developed IEP (Intelligent Endpoint Protection System) security product family for PC, server and other system environments, as well as UWP (Unified Workload Protect) security products for cloud hosts, container and other system environments, providing system security capabilities including endpoint antivirus, endpoint protection (EPP), endpoint detection and response (EDR), and Cloud Workload Protection Platform (CWPP) , etc. Antiy has established a closed-loop product system of threat countermeasures based on its threat intelligence and threat detection capabilities, achieving perception, retardation, blocking and presentation of the advanced threats through products such as the Persistent Threat Detection System (PTD), Persistent Threat Analysis System (PTA), Attack Capture System (ACS), and TDS. For web and business security scenarios, Antiy has launched the PTF Next-generation Web Application and API Protection System (WAAP) and SCS Code Security Detection System to help customers shift their security capabilities to the left in the DevOps process. At the same time, it has developed four major kinds of security service: network attack and defense logic deduction, in-depth threat hunting, security threat inspection, and regular security operations. Through the Threat Confrontation Operation Platform (XDR), multiple security products and services are integrated to effectively support the upgrade of comprehensive threat confrontation capabilities.

Antiy provides comprehensive security solutions for clients with high security requirements, including network and information authorities, military forces, ministries, confidential industries, and critical information infrastructure. Antiy has participated in the security work of major national political and social events since 2005 and has won honors such as the Outstanding Contribution Award and Advanced Security Group. Since 2015, Antiy's products and services have provided security support for major spaceflight missions including manned spaceflight, lunar exploration, and space station docking, as well as significant missions such as the maiden flight of large aircraft, escort of main force ships, and Antarctic scientific research. We have received several thank-you letters from relevant departments.

Antiy is a core enabler of the global fundamental security supply chain. Nearly a hundred of the world's leading security and IT enterprises have chosen Antiy as their partner of detection capability. At present, Antiy's threat detection engine provides security detection capabilities for over 1.3 million network devices and over 3 billion smart terminal devices worldwide, which has become a "national-level" engine. As of now, Antiy has filed 1,877 patents in the field of cybersecurity and obtained 936 patents. It has been awarded the title of National Intellectual Property Advantage Enterprise and the 17th (2015) China Patent Excellence Award.

Antiy is an important enterprise node in China emergency response system and has provided early warning and comprehensive emergency response in major security threats and virus outbreaks such as “Code Red”, “Dvldr”, “Heartbleed”, “Bash Shellcode” and “WannaCry”. Antiy conducts continuous monitoring and in-depth analysis against dozens of advanced cyberspce threat actors (APT groups) such as “Equation”, “White Elephant”, “Lotus” and “Greenspot” and their attack actions, assisting customers to form effective protection when the enemy situation is accurately predicted.