Pack it Secretly: Earth Preta's Updated Stealthy Strategies

⋮ 3/23/2023

APT & Targeted Attacks

Earth Preta has actively been changing its tools, tactics, and procedures (TTPs) to bypass security solutions. In this blog entry, we will introduce and analyze the tools and malware used by the threat actor in its most recent campaigns.

By: Vickie Su, Nick Dai, Sunny Lu March 23, 2023 Read time: 20 min (5504 words)

In our previous research, we disclosed and analyzed a new campaign initiated by the threat actor group Earth Preta (aka Mustang Panda). In a more recent campaign we've been tracking, we discovered Earth Preta delivering lure archives via spear-phishing emails and Google Drive links. After months of investigation, we found that several undisclosed malware and interesting tools used for exfiltration purposes were used in this campaign. We also observed that the threat actors were actively changing their tools, tactics, and procedures (TTPs) to bypass security solutions. In this blog entry, we will introduce and analyze the other tools and malware used by Earth Preta.

Infection chain

As we previously mentioned in our past blog entry, the entire attack begins with a spear-phishing email. After a long-term investigation into the attack routine, we've determined that the full infection chain works

as follows:

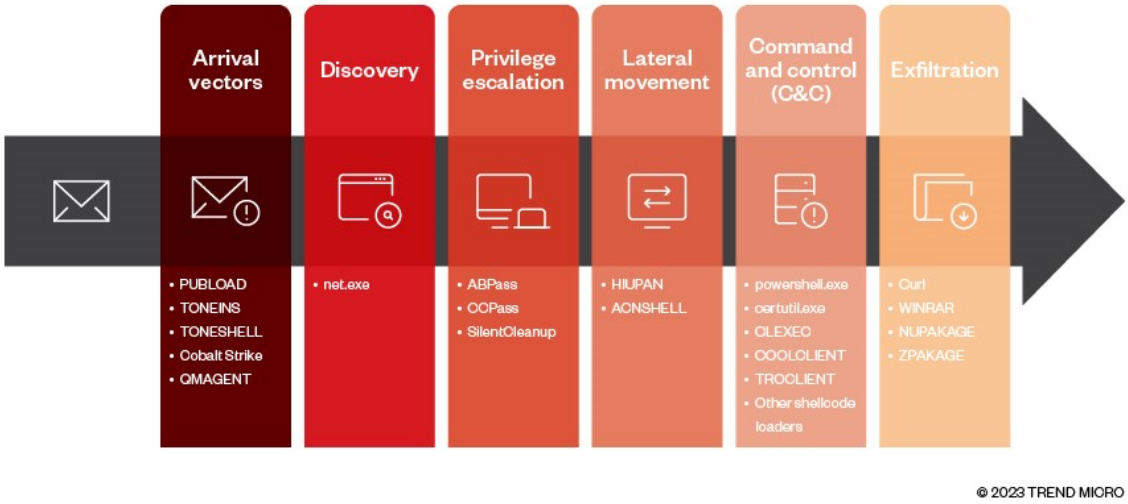


Figure 1. The full infection chain

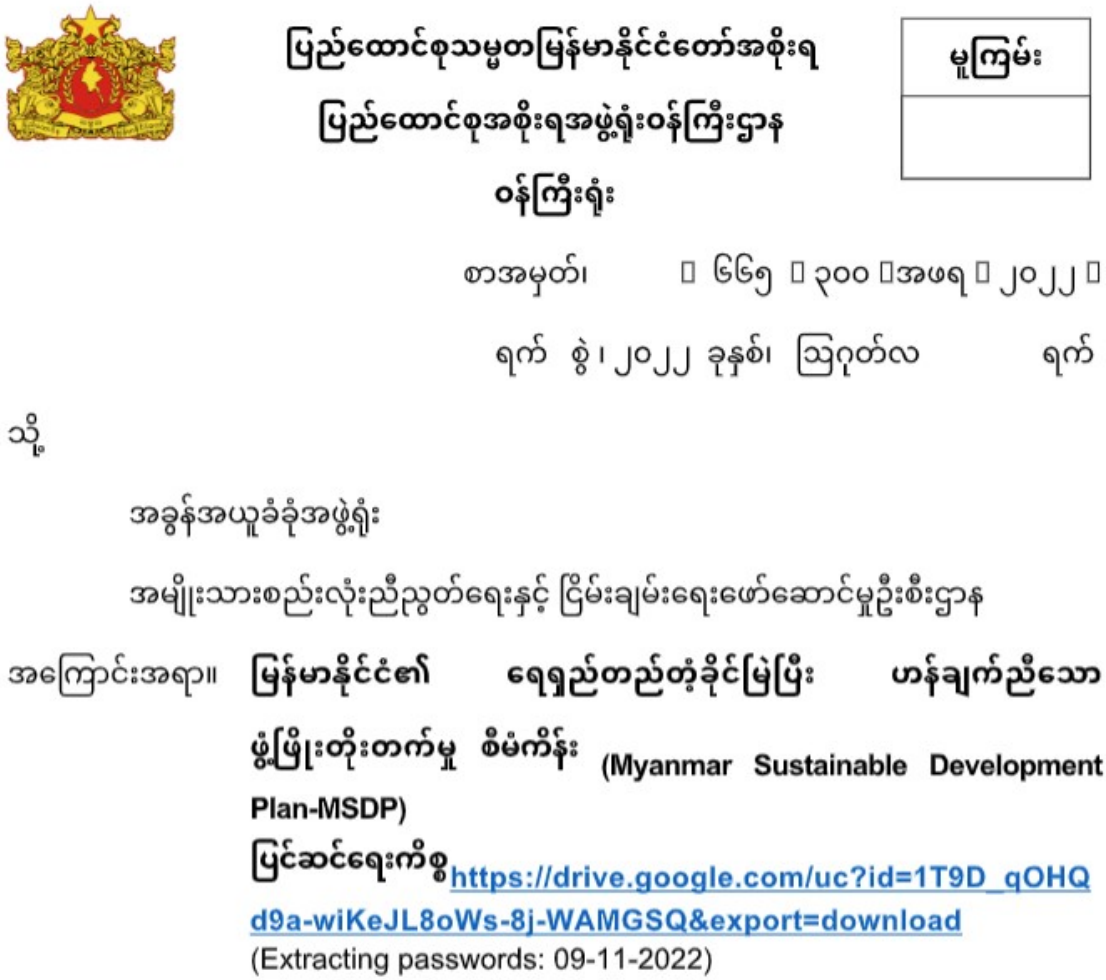
We categorize the different TTPs into six stages: arrival vectors, discovery, privilege escalation, lateral movement, command and control (C&C) and exfiltration, respectively. In our previous research, we covered most of the new TTPs and malware during the first stage, arrival vectors. However, we observed that some of TTPs have been changed. In the following sections, we focus on the updated arrival vectors and their succeeding stages.

Arrival vectors

We previously summarized the arrival vectors used by Earth Preta by categorizing them into three types (DLL sideloading, shortcut links, and fake file extensions). Starting in October and November 2022, we observed that the threat actors began changing their TTPs to deploy the TONEINS, TONESHELL, and PUBLOAD malware. We believe that the threat actors are employing these new techniques to avoid detection.

Trojan.Win32.TONEINS

Based on our earlier observation, the TONEINS and TONESHELL malware were downloaded from the Google Drive link embedded in the body of an email. To bypass email-scanning services and email gateway solutions, the Google Drive link has now been embedded in a lure document. The document lures users into downloading a malicious password-protected archive with the embedded link. The files can then be extracted inside via the password provided in the document. By using this technique, the malicious actor behind the attack can successfully bypass scanning services.


ပြည်ထောင်စုသမ္မတမြန်မာနိုင်ငံတော်အစိုးရ

ပြည်ထောင်စုအစိုးရအဖွဲ့ရုံးဝန်ကြီးဌာန

ဝန်ကြီးရုံး

မူကြမ်း

စာအမှတ်၊ ။ ၆၆၅ ။ ၃၀၀ ။အဖရ ။ ၂၀၂၂ ။

ရက် စွဲ ၊ ၂၀၂၂ ခုနှစ်၊ သြဂုတ်လ ရက်

သို့

အခွန်အယူခံခုံအဖွဲ့ရုံး

အမျိုးသားစည်းလုံးညီညွတ်ရေးနှင့် ငြိမ်းချမ်းရေးဖော်ဆောင်မှုဦးစီးဌာန

အကြောင်းအရာ။ **မြန်မာနိုင်ငံ၏ ရေရှည်တည်တံ့ခိုင်မြဲပြီး ဟန်ချက်ညီသော ဖွံ့ဖြိုးတိုးတက်မှု စီမံကိန်း (Myanmar Sustainable Development Plan-MSDP)**

ပြင်ဆင်ရေးကိစ္စ https://drive.google.com/uc?id=1T9D_qOHQd9a-wiKeJL8oWs-8j-WAMGSQ&export=download

(Extracting passwords: 09-11-2022)

Figure 2. A lure document (allegedly concerning the government-related Myanmar Sustainable Development Plan) embedded with a Google Drive link and a password

For the new arrival vector, the whole infection flow has been changed to the procedure shown in Figure 3.

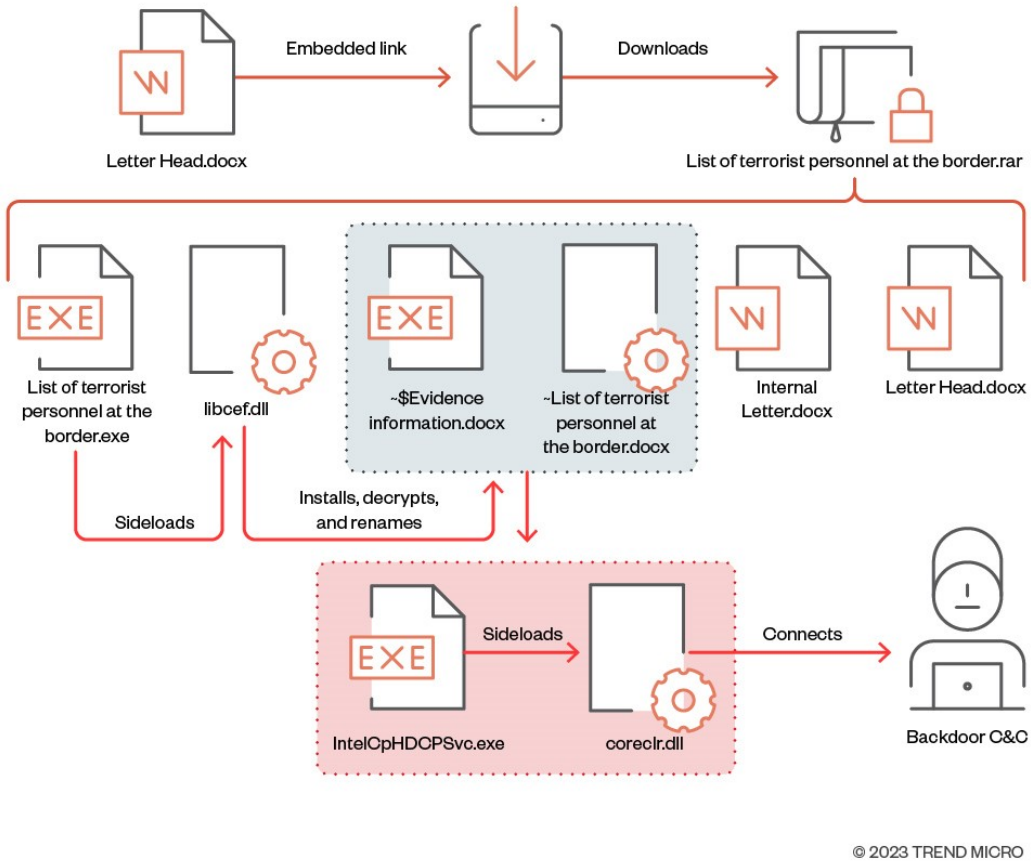


Figure 3. Infection flow for the new arrival vector

File name	Detection name	Description
Letter Head.docx		Decoy document with Google Drive link
List of terrorist personnel at the border.rar (all entries below are part of this archive)		
List of terrorist personnel at the border.exe		First-stage legitimate executable for DLL sideloading
libcef.dll	Trojan.Win32.TONEINS	First-stage malware
~$Evidence information.docx		Second-stage legitimate executable for DLL sideloading
~$List of terrorist personnel at the border.docx	Backdoor.Win32.TONESHELL	Second-stage malware
Internal Letter.docx		Decoy document
Letter Head.docx		Decoy document

Table 1. Files in the new arrival vector

After analyzing the downloaded archive, we discovered it to be a malicious RAR file with the TONEINS malware *libcef.dll* and the TONESHELL malware *~List of terrorist personnel at the border.docx*. The infection flow for these is similar to the arrival vector type C in our previous report, with the only difference being that the fake .docx files have XOR-encrypted content to prevent detection. For example, *~$Evidence information.docx* is a file disguising itself as an Office Open XML document. As such, it seems harmless and can even be opened by using decompression software such as 7-Zip.

We found that the threat actors have hidden a PE file in one of the archive's ZIPFILERECORD structures. The TONEINS malware, *libcef.dll*, will decrypt this file with a single byte in XOR operations, find the PE header, and drop the payload to the specified path.

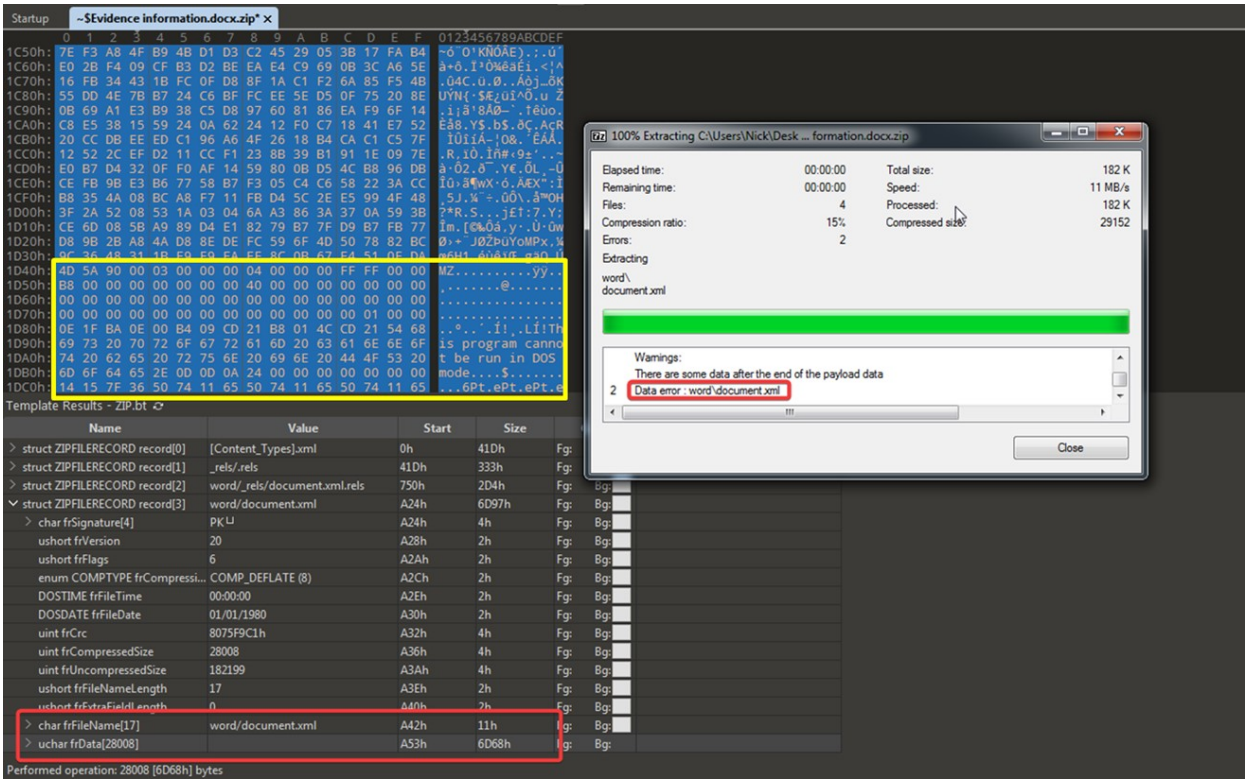


Figure 4. A PE file is revealed after decrypting the frData member in the last ZIPFILECECORD structure.

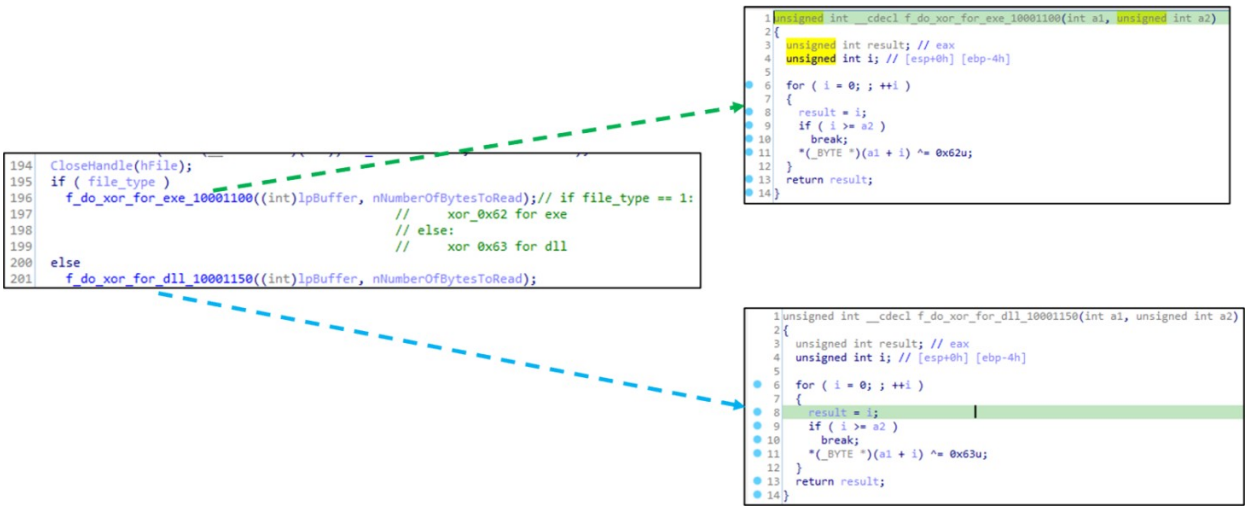


Figure 5. The decryption function of TONEINS

The succeeding behaviors of the infection flow are generally the same as those in our previous analysis, where we provide more details.

In more recent cases, the malware PUBLOAD was also being delivered through Google Drive links embedded in decoy documents.

U.S. EMBASSY Rangoon:
We would like to send the invitation letter and agenda for the 0201-2022 coup meeting which will be held on 11-20-2022 (Thursday).
Please see the attached file and join the meeting via zoom application.
https://drive.google.com/uc?id=1tyBkJ8gkaQXShYZG53jXwygj5TiVMvNK&export=download

Figure 6. The lure document Invitation letter from the US embassy.docx

Since October 2022, we have been observing a new variant of PUBLOAD, which uses the spoofed HTTP header to transfer the data, as LAC's report also discusses. In contrast to the previous PUBLOAD variant, it prepends an HTTP header with a legitimate-looking host name to the packets. We believe that the threat actors are trying to conceal malicious data among normal traffic. The data in the HTTP body is the same as the past variant, which has the same magic bytes *17 03 03* and the encrypted victim information. We were able to successfully retrieve the payload from a live C&C server and were therefore able to continue our analysis.

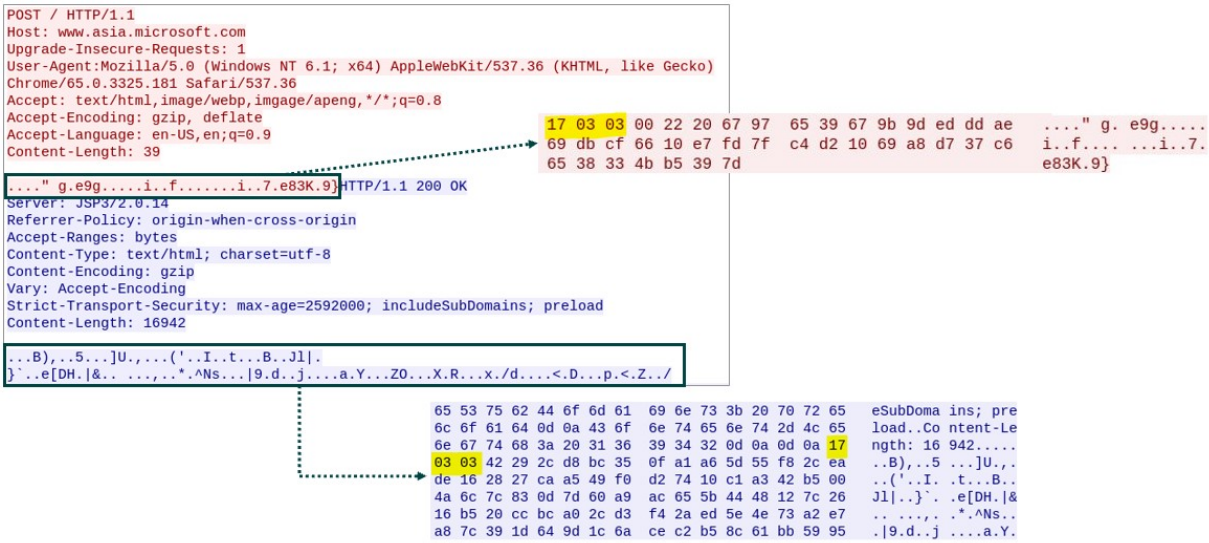


Figure 7. C&C traffic of the PUBLOAD HTTP variant

Once the payload is received, it will check if the first three magic bytes are *17 03 03* and if the following two bytes are the size of payload. It will then decrypt the encrypted payload with the predefined RC4 key *78 5A 12 4D 75 14 14 11 6C 02 71 15 5A 73 05 08 70 14 65 3B 64 42 22 23 20 00 00 00 00 00 00 00*, which is the same as the one used in the PUBLOAD loader.

Figure 8. The first payload retrieved from the PUBLOAD HTTP variant

After decryption, it then checks if the first byte of the decrypted payload is *0x06*. The decrypted payload contains another payload that is XOR-encrypted with the bytes *23 BE 84 E1 6C D6 AE 52 90*.

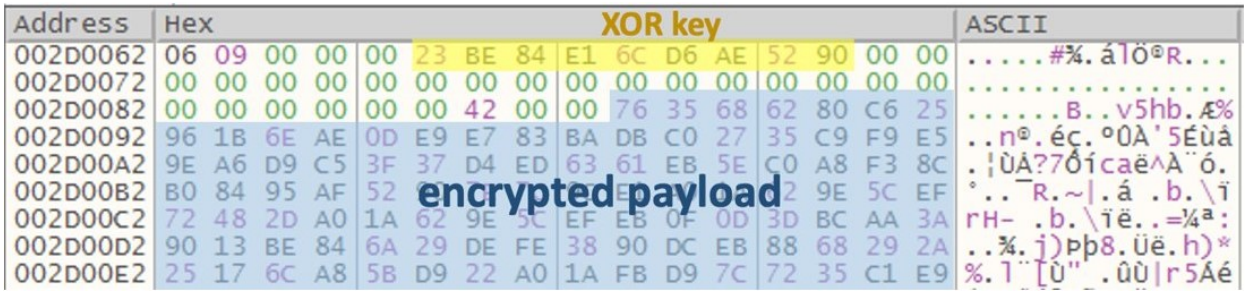


Figure 9. The second payload retrieved from the PUBLOAD HTTP variant

After this is decrypted, there is yet another final backdoor payload that supports data upload and command execution.

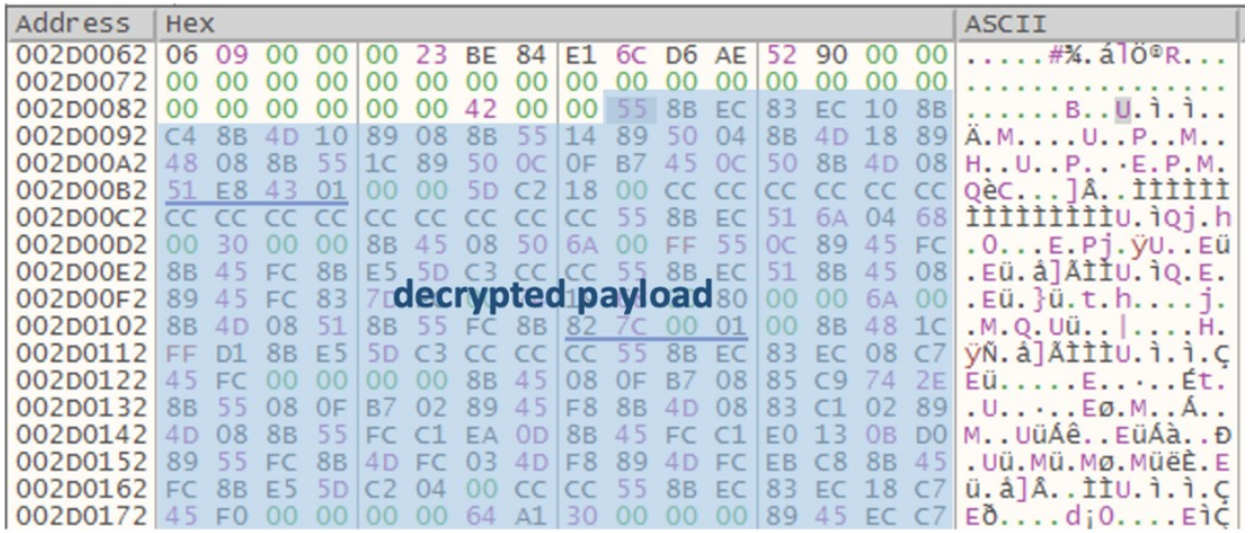


Figure 10. The final payload of the PUBLOAD HTTP variant

Command	Internal string
0x03	-
0x01	-
0x1B	*UploadBegin error : %d!*

0x1D	*UploadData error : %d!*
0x1A	-
0x1E	*CmdStart error : %d!*
0x1F	*CmdWrite error : %d!*
0x30	*CmdWrite error : %d*
0x20	-

Table 2. Command codes in the PUBLOAD HTTP variant

In addition, we found some interesting debug strings and event names among the PUBLOAD samples.

```
BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
  void *v3; // ecx
  int v5; // [esp+0h] [ebp-1Ch]
  int v6; // [esp+4h] [ebp-18h]
  int v7; // [esp+8h] [ebp-14h]
  int v8; // [esp+Ch] [ebp-10h]
  int v9; // [esp+10h] [ebp-Ch]
  int v10; // [esp+14h] [ebp-8h]
  int v11; // [esp+18h] [ebp-4h]

  if ( fdwReason == 1 )
  {
    if ( OpenEventA(0x1F0003u, 0, "ARRxYxelonmuskxxxx") )
      ExitProcess(0);
    CreateEventA(0, 0, 0, "ARRxYxelonmuskxxxx");
    v7 = 0;
    v8 = 0;
    v9 = 0;
    v10 = 0;
    v11 = 0;
    v6 = 0;
    sub_10018F60(&v6);
    GetModuleFileNameW(0, &PathName, 0x104u);
    wcsrchr(&PathName, 0x5Cu)[1] = 0;
    SetCurrentDirectoryW(&PathName);
    sub_100193D0();
    v6 = 0;
    v7 = 0;
    v8 = 0;
    v9 = 0;
    v10 = 0;
    v11 = 0;
    v5 = 0;
    sub_100190E0((int)&v5);
    sub_10019380(v3);                           // main
  }
  return 1;
}
```

Figure 11. Event name in PUBLOAD

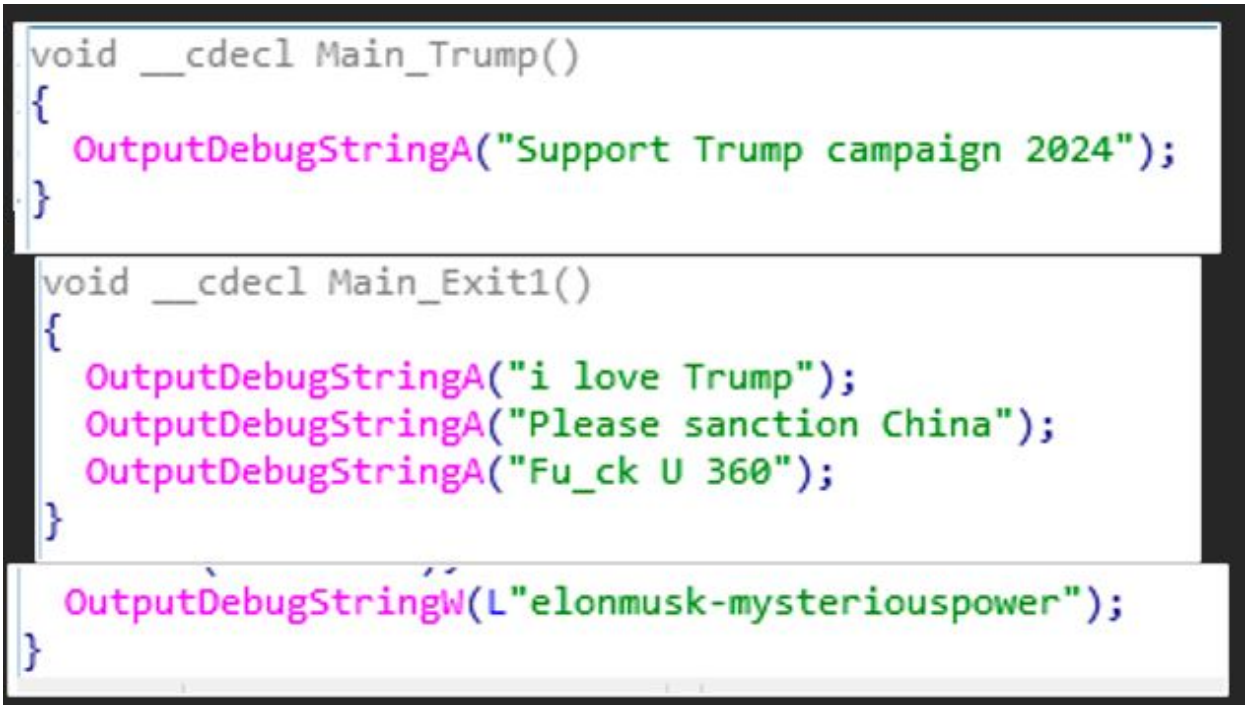


Figure 12. Debug string in PUBLOAD

In summary, we think that the new TONESHELL and PUBLOAD archives have been evolving and now have something in common. For example, both of them are now being placed in decoy documents (such as Google Drive links) in order to bypass antivirus scanning.

Discovery

Once the threat actors obtain access to the victim's environment, they can start inspecting the environment via the following commands:

net user

net user <username>

net user <username> /DOMAIN

Privilege escalation

In this campaign, we discovered several tools used for UAC bypass in Windows 10. We will go into detail for each of them.

HackTool.Win 32.ABPASS

HackTool.Win32.ABPASS is a tool used to bypass UAC in Windows 10. Based on our analysis, it reuses codes from the function ucmShellRegModMethod3, which is from a famous open-source project called UACME. A report from Sophos introduces this tool.

This tool accepts an argument, and the following data is written into registry:

Registry Key	Name	Value
HKEY_USERS\<SID>-1001_Classes\aaabbb32\shell\open\command	*(Default)*	*argv[1]*

HKEY_USERS\<SID>-1001_Classes\ms-settings\CurVer	*(Default)*	*aaabbb32*

Table 3. Registry keys changed by ABPASS

It also changes how Windows handles the *ms-settings* protocol — in this case, the string *ms-settings* is a Programmatic Identifier (ProgID). If the CurVer key is set under a ProgID, it will be used for versioning and mapping the current ProgID (*ms-settings*) to the one specified in the CurVer's default value. In turn, the behavior of *ms-settings* is redirected to the custom defined ProgID *aaabbb32*. It also sets up a new ProgID *aaabbb32* and its shell open command. Finally, *fodhelper.exe* or *computerDefaults.exe* will be executed to trigger the *ms-settings* protocol.

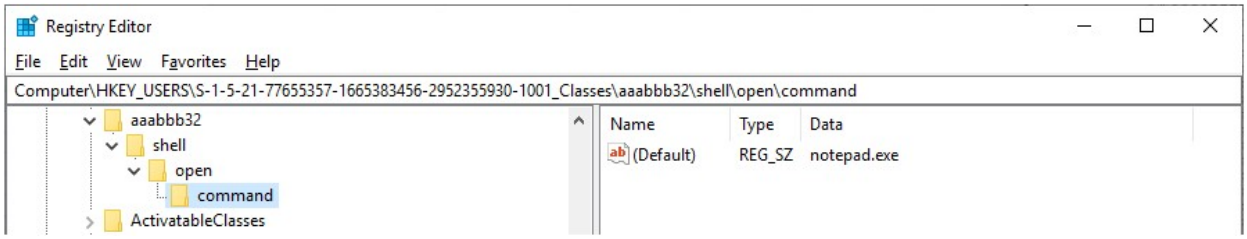


Figure 13. The added ProgID aaabbb32

HackTool.Win 32.CCPASS

HackTool.Win32.CCPASS is another tool that is also used for Windows 10 UAC bypass and similarly reuses codes from the function ucmMsStoreProtocolMethod in the project UACME.

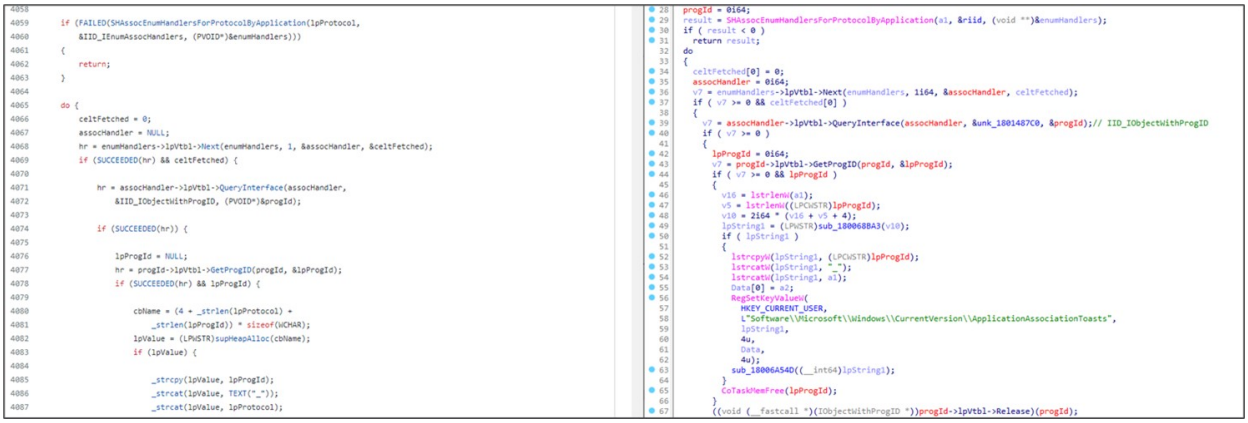


Figure 14. Code similarities in CCPASS and ucmMsStoreProtocolMethod

It works in a similar way to ABPASS. However, unlike ABPASS, it hijacks the *ms-windows-store* protocol. The hack tool CCPASS works as follows:

1. It disables the application association toasts for the protocol *ms-windows-store*.
2. It creates a new Shell in the registry.
3. It invokes the undocumented API UserAssocSet to update the file association.
4. It executes *WSReset.exe* to trigger this protocol.

In Windows 10 and above, the system shows a new toast dialog for selecting the open application for the selected file type. To hide this window, the tool explicitly adds new entries to *HKCU\Software\Microsoft\Windows\CurrentVersion\ApplicationAssociationToasts* to disable all toasts related to the protocol *ms-windows-store*.

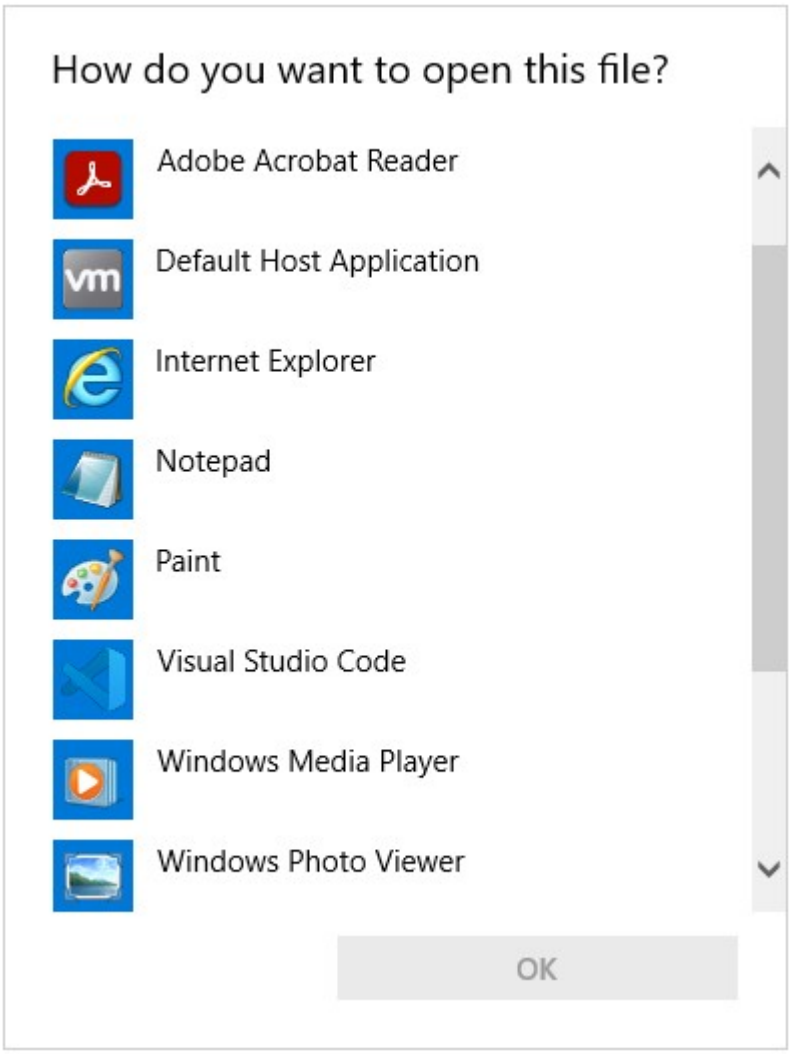


Figure 15. An example of the application association toast

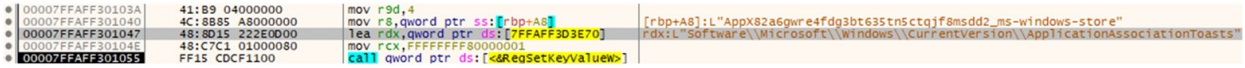


Figure 16. Hiding application association toasts via the registry

Once this is done, the tool starts to alter the shell command of *ms-windows-store* and finally triggers it using *WSReset.exe*.

SilentCleanup

In Windows 10, there is a native Windows service called “SilentCleanup.” This service has the highest privileges that can be abused for Windows 10 UAC bypass. Normally, this service is intended for running *%windir%\system32\cleanmgr.exe*. However, the environment variable *%windir%* can be hijacked and changed to any path to achieve privilege escalation.

Figure 17. Malicious commands abusing the SilentCleanup service

We observed that the threat actors used this technique to execute *c:\users\public\1.exe*.

Lateral movement

In this stage, we observed certain malware such as HIUPAN and ACNSHELL (initially introduced and analyzed by Mandiant and Sophos) being used to install themselves to removable disks and create a reverse shell.

USB Worm: Worm.Win 32.HIUPAN and+ Backdoor.Win 32.ACNSHELL

We found a pair of malware comprised of a USB worm and a reverse shell —includin g a USB worm and a reverse shell (detected as Worm.Win32.HIUPAN and Backdoor.Win32.ACNSHELL, respectively,) — being used to spread themselves over removable drives.

Figure 18 shows the infection chain for both.

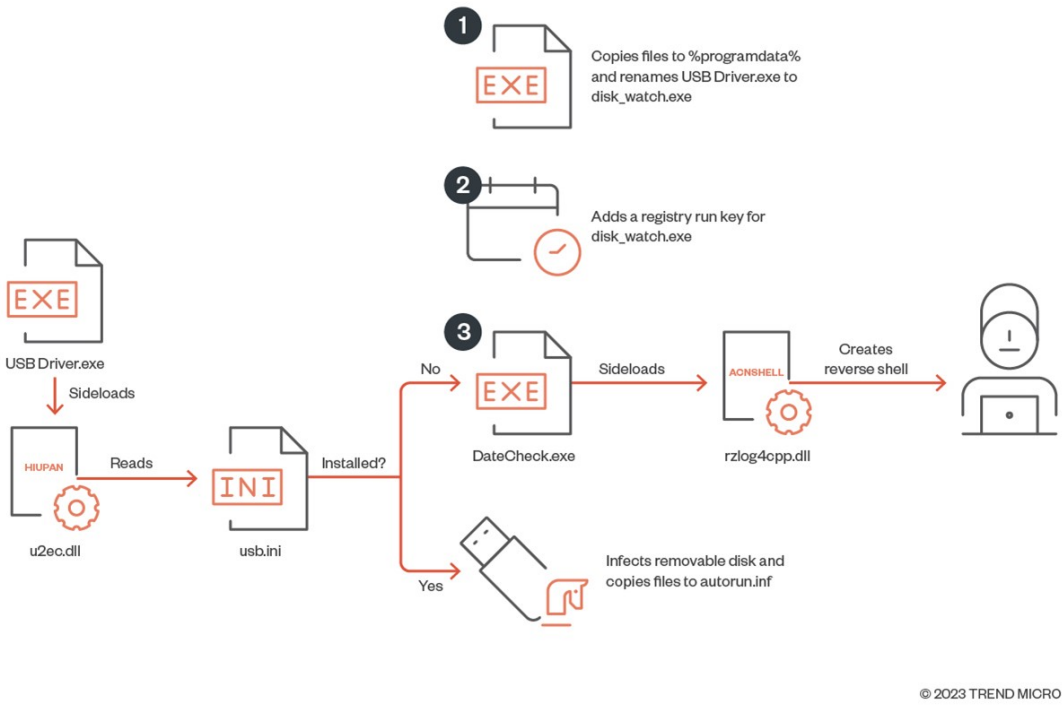


Figure 18. HIUPAN and ACNSHELL infection flow

The *USB Driver.exe* program first sideloads *u2ec.dll*, which then loads the payload file *usb.ini*. They have the following PDB strings, respectively:

- *G:\project\APT\U盘劫持\new\u2ec\Release\u2ec.pdb*
- *G:\project\APT\U盘劫持\new\shellcode\Release\shellcode.pdb*

The string *U盘劫持* means “U disk hijacking,” where “U disk” refers to removable drives.

USB Driver.exe then starts checking whether it is properly installed. If it is installed, it will start to infect more removable disks and copy files to a folder named *autorun.inf*. If it is not installed, it installs itself to *%programdata%* and then sets the registry run key for persistence.

Finally, the ACNSHELL malware *rzlog4cpp.dll* is sideloaded. It will then create a reverse shell via *ncat.exe* to the server *closed[.]theworkpc[.]com*.

Command and Control (C&C) stage

Earth Preta employed several tools and commands for the C&C stage. For example, the group used *certutil.exe* to download the legitimate WinRAR binary as *rar1.exe* from the server 103[.]159[.]132[.]91.

Figure 19. The certutil.exe program downloads the WinRAR binary

We also observed that the threat actors used PowerShell to download multiple malware and archives from the server 103[.]159[.]132[.]181 for future use.

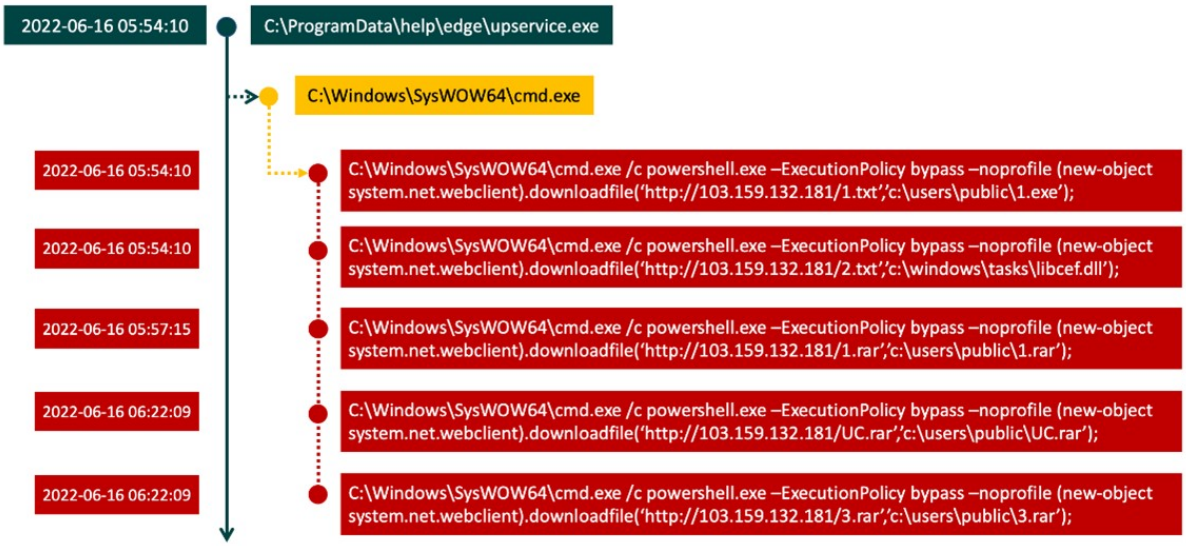


Figure 20. PowerShell downloading malware

In certain instances, they even leveraged the WinRAR binary installed on the victim hosts to decompress all the malware.

Figure 21. Decompressing malware with the installed WinRAR binary

Although we found several logs involving multiple pieces of dropped malware, we only managed to retrieve a few of them. Among all our collected samples, we will introduce the most noteworthy ones.

Backdoor.Win32.CLEXEC

The file name of the backdoor CLEXEC is *SensorAware.dll*. This is a simple backdoor that is capable of executing commands and clearing event logs.

```
LONG __thiscall f_backdoor_commands_10002FA0(int this, _BYTE *a2, int a3)
{
  LONG result; // eax

  result = (unsigned __int8)*a2 - 81;
  switch ( *a2 )
  {
    case 'Q':
      result = InterlockedExchange((volatile LONG *)(this + 40536), 1);
      break;
    case 'T':
      *(_DWORD *)(this + 4 * *(_DWORD *)(this + 40528) + 528) = sub_10003630(
                                                                  0,
                                                                  0,
                                                                  (int)f_WinExec_10002D60,
                                                                  *(_DWORD *)(*(_DWORD *)(this + 4) + 172),
                                                                  0,
                                                                  0,
                                                                  1);
      result = *(_DWORD *)(this + 40528) + 1;
      *(_DWORD *)(this + 40528) = result;
      break;
    case 'V':
      result = MessageBoxA(0, Text, 0, 0);
      break;
    case 'X':
      result = f_clean_event_log_10002E40();
      break;
    default:
      return result;
  }
  return result;
}
```

Figure 22. Command codes of CLEXEC

Backdoor.Win32.COOLCLIENT

The backdoor COOLCLIENT was first introduced in a report from Sophos; the sample mentioned in the report was compiled in 2021. In our case, the COOLCLIENT sample we analyzed had a more recent compilation time in 2022, and while it provides the same functionalities, it has the added capability to open a decoy document (*work.pdf*) when the current process name has ".pdf" or ".jpg" file extensions. It contains less OutputDebugStrings calls. Meanwhile, *loader.ja* is used under two processes: One is under *googleupdate.exe*, which is used for the first sideloading. The second is under *winver.exe*, which is injected to conduct backdoor behaviors. Furthermore, COOLCLIENT applies obfuscation techniques that we discuss in later sections.

```
52
53  memset(Filename, 0, sizeof(Filename));
54  GetModuleFileNameA(0, Filename, 0x104u);
55  strlwr_0(Filename);
56  if ( sub_10003F70((const __m128i *)"pdf", (__m128i *)Filename, a1)
57    || sub_10003F70((const __m128i *)"jpg", (__m128i *)Filename, a1) )
58  {
59    memset(File, 0, 0x104u);
60    GetModuleFileNameA(0, File, 0x104u);
61    v1 = (_BYTE *)sub_10003F60(File, 92);
62    v2 = &v40;
63    *v1 = 0;
64    while ( *++v2 )
65      ;
66    strcpy(v2, "\\work.pdf");
67    ShellExecuteA(0, "open", File, 0, 0, 5);
68  }
69  v4 = GetCommandLineW();
70  v5 = CommandLineToArgvW(v4, pNumArgs);
```

Figure 23. Open decoy document

Figure 24 shows the whole execution flow of COOLCLIENT.

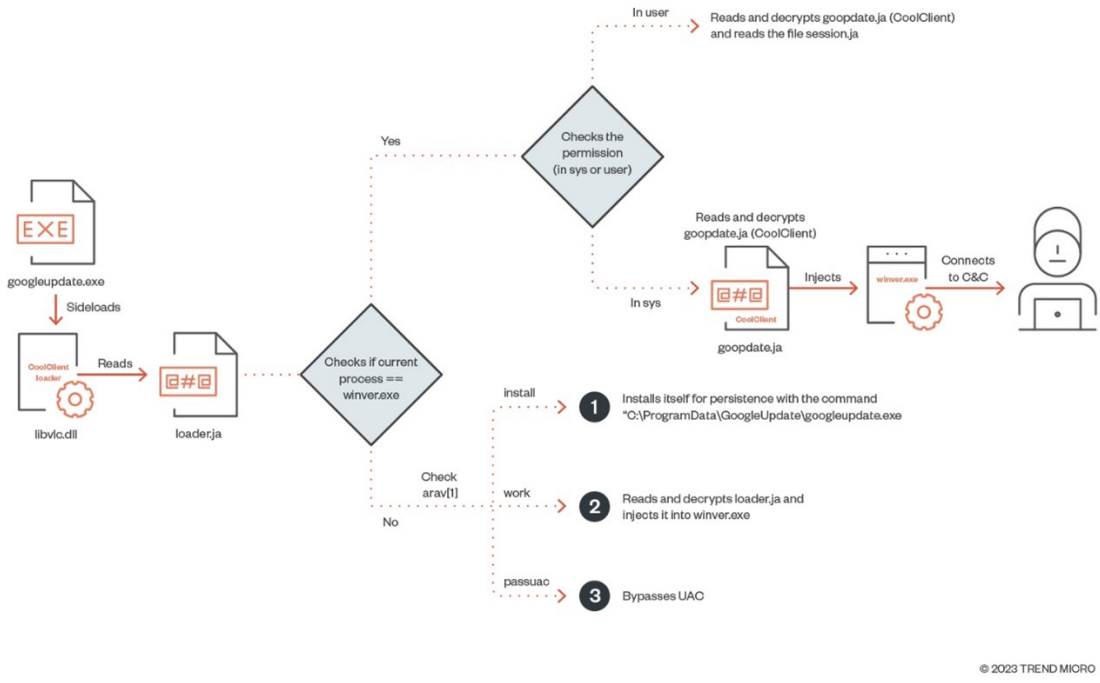


Figure 24. Execution flow of COOLCLIENT

The arguments of COOLCLIENT provide the following capabilities:

install. There are several ways to install COOLCLIENT, detailed here:

1. It installs itself by creating an InstallSvc service called InstallSvc which will trigger "*googleupdate.exe work*"..
2. It sets up a run key for via the command *C:\ProgramData\GoogleUpdate\googleupdate.exe work* for persistence.

work. The malware will continue to read and decrypt *goopdate.ja* and inject it into *winver.exe* for the next-stage payload (COOLCLIENT), which contains malicious behaviors.

passuac. The malware will check if the process *avp.exe* exists. If *avp.exe* doesn't exist, UAC bypass will be executed via the CMSTPLUA COM interface. If *avp.exe* exists, UAC bypass will be executed via the AppInfo RPC service.

```
17  void *ppv; // [esp+264h] [ebp-4h] BYREF
18
19  v15 = this;
20  ppv = 0;
21  pclsid = 0i64;
22  v1 = 0;
23  iid = 0i64;
24  CLSIDFromString(L"{3E5FC7F9-9A51-4367-9063-A120244FBEC7}", &pclsid);
25  IIDFromString(L"{6EDD6D74-C007-4E75-B76A-E5740995E24C}", &iid);
26  memset(pszName, 0, 0x208u);
27  CoInitialize(0);
28  v2 = v7;
29  do
30  {
31    v3 = v2[1];
32    ++v2;
33  }
34  while ( v3 );
35  qmemcpy(v2, L"Elevation:Administrator!new:", 0x3Au);
36  v4 = v7;
37  do
38  {
39    v5 = *((_WORD *)v4 + 1);
40    v4 += 2;
41  }
42  while ( v5 );
43  *(_DWORD *)pBindOptions = 36;
44  v14 = 0;
45  wcscpy(v4, (wchar_t *const)L"{3E5FC7F9-9A51-4367-9063-A120244FBEC7}");
46  v12 = 4;
```

Figure 25. UAC Bypass via the CMSTPLUA COM interface

```
while ( v18 );
v19 = RpcStringBindingComposeW(
        (RPC_WSTR)L"201ef99a-7fa0-444c-9399-19ba84f12a1a",
        (RPC_WSTR)L"ncalrpc",
        0,
        0,
        0,
        &StringBinding);
if ( !v19 )
{
  v19 = RpcBindingFromStringBindingW(StringBinding, &Binding);
  RpcStringFreeW(&StringBinding);
  if ( !v19 )
  {
    v20 = LocalAlloc(0x40u, uBytes);
    if ( v20 )
    {
      if ( CreateWellKnownSid(WinLocalSystemSid, 0, v20, &uBytes) )
      {
        SecurityQOS.Version = 3;
        SecurityQOS.ImpersonationType = 3;
        SecurityQOS.Capabilities = 1;
        SecurityQOS.IdentityTracking = 0;
        v27 = v20;
        v19 = RpcBindingSetAuthInfoExW(Binding, 0, 6u, 0xAu, 0, 0, &SecurityQOS);
        if ( !v19 )
        {
          v31 = Binding;
          Binding = 0;
          LocalFree(v20);
          goto LABEL_20;
        }
      }
      else
      {
        v19 = GetLastError();
```

Figure 26. UAC Bypass via the AppInfo RPC service

According to our analysis, it reads the encrypted configuration file *time.sig*. It is also able to communicate through different network protocols such as UDP (User Datagram Protocol) and TCP (Transmission Control Protocol). Based on some internal strings and the APIs used by Earth Preta, the functionalities of this backdoor can be inferred as follows:

- Send portmap
- Build connection
- Read file
- Delete file
- Keystrokes and windows monitoring

Backdoor.Win32.TROCLIENT

The backdoor TROCLIENT, which was also first disclosed in Sophos's report, is similar to COOLCLIENT. However, this backdoor has an anti-debugging technique, which will check if the running processes have the strings *dbg.exe* or *olly*.

```
  if ( v3 == (HANDLE)-1 )
  {
LABEL_11:
    malicious_main();
    return 1;
  }
  if ( !Process32FirstW(v3, &pe) )
  {
LABEL_9:
    CloseHandle(v3);
    v6 = GetModuleHandleA("ntdll.dll");
    ZwSetInformationObject = GetProcAddress(v6, "ZwSetInformationObject");
    if ( ZwSetInformationObject )
    {
      LOWORD(a2) = 256;
      v8 = GetCurrentProcess();
      DuplicateHandle(v8, v8, v8, &TargetHandle, 0, 0, 0);
      ((void (__stdcall *)(HANDLE, int, int *, int))ZwSetInformationObject)(TargetHandle, 4, &a2, 2);
      DuplicateHandle(v8, TargetHandle, v8, &TargetHandle, 0, 0, 1u);
    }
    goto LABEL_11;
  }
  while ( 1 )
  {
    sub_1001CBE0((__m128i *)v11, 0, 0x208u);
    v4 = 0;
    do
    {
      v5 = pe.szExeFile[v4++];
      pe.szExeFile[v4 + 259] = v5;
    }
    while ( v5 );
    sub_100227D4(v11);
    if ( sub_1001BD24(v11, L"dbg.exe") || sub_1001BD24(v11, L"olly") )
      return 0;
    if ( !Process32NextW(v3, &pe) )
      goto LABEL_9;
  }
}
```

Figure 27. TROCLIENT anti-debugging technique.

Figure 28 shows the whole execution flow of TROCLIENT.

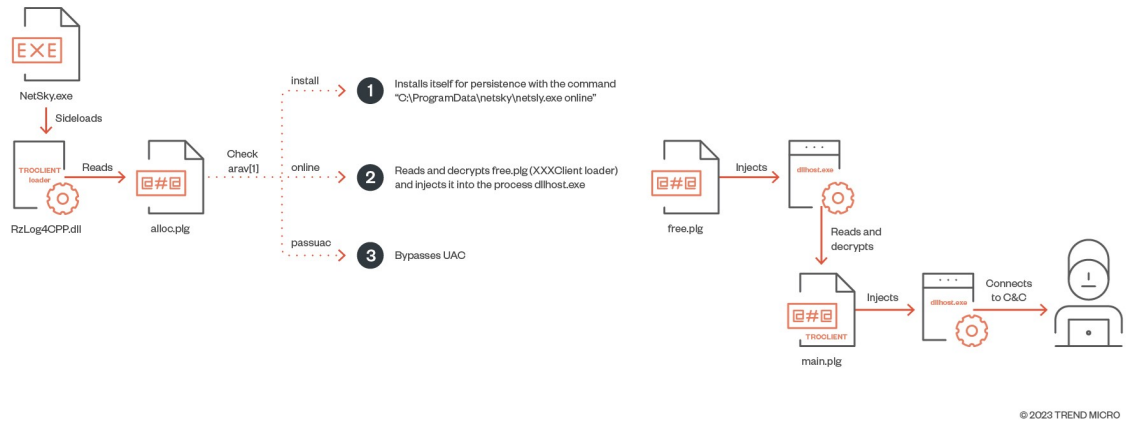


Figure 28. Execution flow of TROCLIENT

The arguments of TROCLIENT provide the following capabilities:

install. There are two waysto determine the method of installation for TROCLIENT, detailed here:

1. It installs itself by creating aservice called InstallSvc which will trigger "*C:\programdata\netsky\netsky.exe online"*.
2. It sets up a run key for the command *C:\programdata\netsky\netsky.exe onlin*e for persistence.

online: It will read the next stage payloads, *free.plg* and *main.plg*, and inject them into *dllhost.exe.*

passuac: The malware will check if the process *avp.exe* exists. If it does not, UAC bypass is executed via the CMSTPLUA COM interface. If *avp.exe* exists, UAC bypass is executed via token manipulation.

```
if ( v2 == 2 )
{
  if ( !sub_10022AF3(v1[1], L"install") )
  {
    if ( sub_1001AF90() )                    // install service
    {
      sub_1001AC80();
    }
    else if ( sub_10002320() )
    {
      sub_1001A4B0();                        // shell execute netsky.exe passuac
    }
    else
    {
      SHSetValueA(
        HKEY_CURRENT_USER,
        "Software\\Microsoft\\Windows\\CurrentVersion\\Run",
        "Appdata",
        1u,
        "C:\\ProgramData\\Netsky\\NetSky.exe online",
        0x27u);
      sub_1001A2E0();
    }
  }
  if ( !sub_10022AF3(v1[1], L"online") )
    sub_1001A2E0();
  if ( !sub_10022AF3(v1[1], L"passuac") )
  {
    if ( sub_1001A0B0() == -1 )
    {
      sub_1001AD90();                        // uacbypass_CMSTPLUA
    }
    else
    {
      v5 = (void (*)(void))VirtualAlloc(0, 0x17BD6u, 0x1000u, 0x40u);
      sub_1001E1A0(v5, dword_100024D0, 97238);// uac is useless
      v5();
    }
  }
```

Figure 29. The capabilities of three different arguments

This backdoor provides the following capabilities:

- Read file
- Delete file
- Monitor keystrokes and windows

There are several similarities and differences between COOLCLIENT and TROCLIENT, as Table 3 shows.

Argument/Behaviors	COOLCLIENT	TROCLIENT
install		
Creates a service named InstallSvc	✓	✓
Executes itself with passuac	✓	✓

Sets Run Key with “work/online”	✓	✓
passuac		
AppInfo RPC	✓	
CMSTPLUA COM	✓	✓
Token manipulation		✓
work/online		
Send portmap	✓	
Connect to C&C	✓	✓
File operations	✓	✓
Keylogging	✓	✓

Table 3. Comparison of COOLCLIENT and TROCLIENT

In addition to the aforementioned malware, we also found several shellcode loaders for PlugX. Since it is a known malware family, we will not expand on its details in this blog entry.

Exfiltration

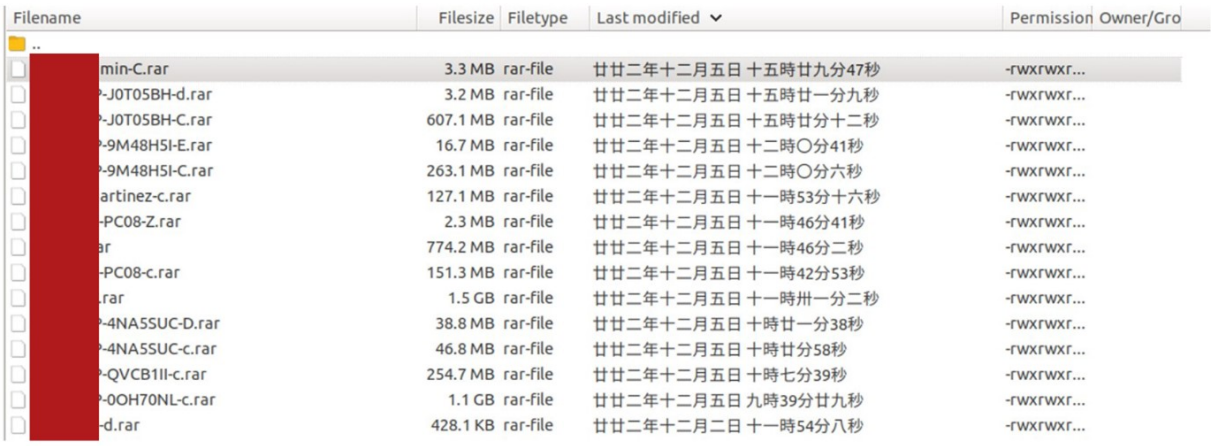
Based on our telemetry, we found that Earth Preta used multiple approaches to exfiltrate sensitive data from the victims. For example, in some cases, we observed that WinRAR and curl (or cURL) were leveraged to collect and transfer data to the threat actor’s server. After further investigation, we even found some previously unseen pieces of malware that were used to collect data in a custom-made file format. In the following sections, we share the details of the unique exfiltration toolsets developed by Earth Preta.

WinRAR and curl

According to some of our monitoring logs, the threat actors abused the installed WinRAR binary and the uploaded curl executable to exfiltrate the files (Figure 30 shows the executed command). Note that the executable *log.log* is a legitimate curl binary. All the exfiltrated data was collected and sent back to the threat actor-controlled FTP (File Transfer Protocol) server.

Figure 30. Exfiltrate data using WinRAR and curl

In some cases, we accidentally stumbled on the account and password of the FTP server. Upon checking the FTP server, we learned that the threat actors focused on sensitive and confidential documents, most of which were compressed and protected with a password. Based on our observations, the documents were organized via the categorization of the victim’s host name and disk drive.



Filename	Filesize	Filetype	Last modified	Permission	Owner/Gro
..					
[redacted]min-C.rar	3.3 MB	rar-file	廿廿二年十二月五日 十五時廿九分47秒	-rwxrwxr...	
[redacted]-J0T05BH-d.rar	3.2 MB	rar-file	廿廿二年十二月五日 十五時廿一分九秒	-rwxrwxr...	
[redacted]-J0T05BH-C.rar	607.1 MB	rar-file	廿廿二年十二月五日 十五時廿分十二秒	-rwxrwxr...	
[redacted]-9M48H5I-E.rar	16.7 MB	rar-file	廿廿二年十二月五日 十二時〇分41秒	-rwxrwxr...	
[redacted]-9M48H5I-C.rar	263.1 MB	rar-file	廿廿二年十二月五日 十二時〇分六秒	-rwxrwxr...	
[redacted]artinez-c.rar	127.1 MB	rar-file	廿廿二年十二月五日 十一時53分十六秒	-rwxrwxr...	
[redacted]-PC08-Z.rar	2.3 MB	rar-file	廿廿二年十二月五日 十一時46分41秒	-rwxrwxr...	
[redacted]ar	774.2 MB	rar-file	廿廿二年十二月五日 十一時46分二秒	-rwxrwxr...	
[redacted]-PC08-c.rar	151.3 MB	rar-file	廿廿二年十二月五日 十一時42分53秒	-rwxrwxr...	
[redacted].rar	1.5 GB	rar-file	廿廿二年十二月五日 十一時卅一分二秒	-rwxrwxr...	
[redacted]-4NA5SUC-D.rar	38.8 MB	rar-file	廿廿二年十二月五日 十時廿一分38秒	-rwxrwxr...	
[redacted]-4NA5SUC-c.rar	46.8 MB	rar-file	廿廿二年十二月五日 十時廿分58秒	-rwxrwxr...	
[redacted]-QVCB1II-c.rar	254.7 MB	rar-file	廿廿二年十二月五日 十時七分39秒	-rwxrwxr...	
[redacted]-0OH70NL-c.rar	1.1 GB	rar-file	廿廿二年十二月五日 九時39分廿九秒	-rwxrwxr...	
[redacted]-d.rar	428.1 KB	rar-file	廿廿二年十二月二日 十一時54分八秒	-rwxrwxr...	

Figure 31. FTP servers with stolen documents

Apart from well-known legitimate tools, the threat actors also crafted highly customized tools used for exfiltration. We named this malware “NUPAKAGE," a name derived from its unique PDB string, *D:\Project\NEW_PACKAGE_FILE\Release\NEW_PACKAGE_FILE.pdb*.

The NUPAKAGE malware needs a unique passcode to be executed, with the exfiltrated data being wrapped in a custom file format. It seems that the threat actors are continuously updating this tool to provide more flexibility and lower the possibility of detection, including adding more command-line arguments and obfuscation mechanisms. By default, it only collects documents, including the files with the following extensions:

- .doc
- .docx
- .xls
- .xlsx
- .ppt
- .pptx
- .pdf

It avoids collecting documents with file names starting with “$” or “~” since these types of documents are usually either temporary files generated by the system or PE files pretending to be decoy documents (as we discussed in the arrival vectors section).

The usage of this tool is as follows:

malware.exe ***passcode start end chunk -s extension_A extension_B …***

Argument Name	Format	Example Value	Description
passcode	String	comeon	A unique code to execute it
start	String	2022-01-01	The start range of the exfiltrated file’s modification timestamp
end	String	2022-12-31	The end range of the exfiltrated file’s modification timestamp
chunk	Integer	4096	Splits the generated data in chunks by the specified size (MB)

-s	String		File extensions to be collected; optional

Table 5. Arguments of the NUPAKAGE malware

Every NUPAKAGE malware needs a unique passcode as its first argument to continue execution. As Figure 32 shows, it first checks if the passcode exists. If not, the malware execution procedure will terminate. In our collection, we observed different passcodes in each malware.

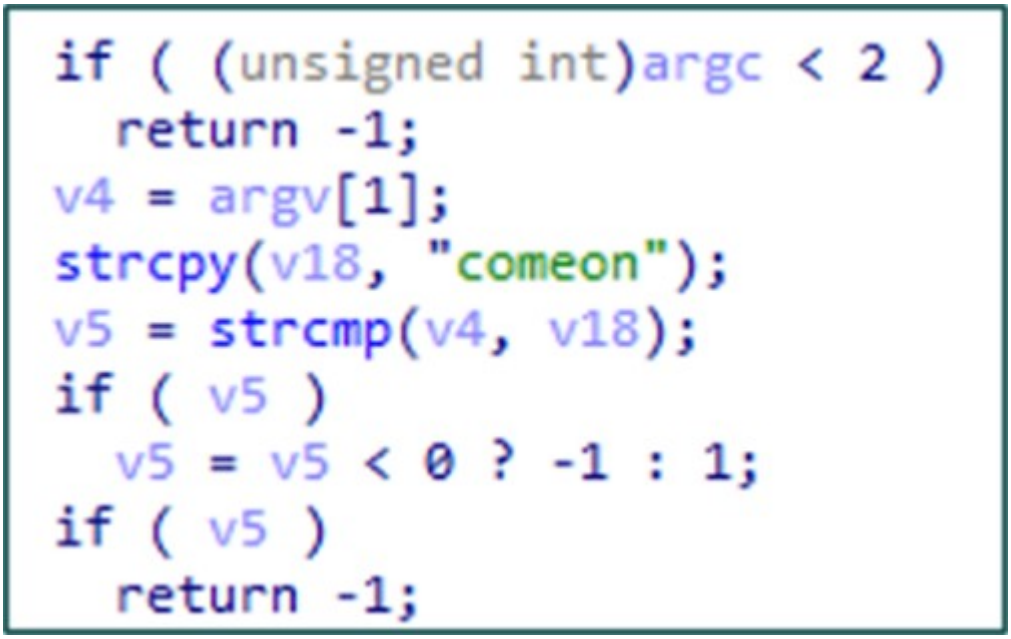
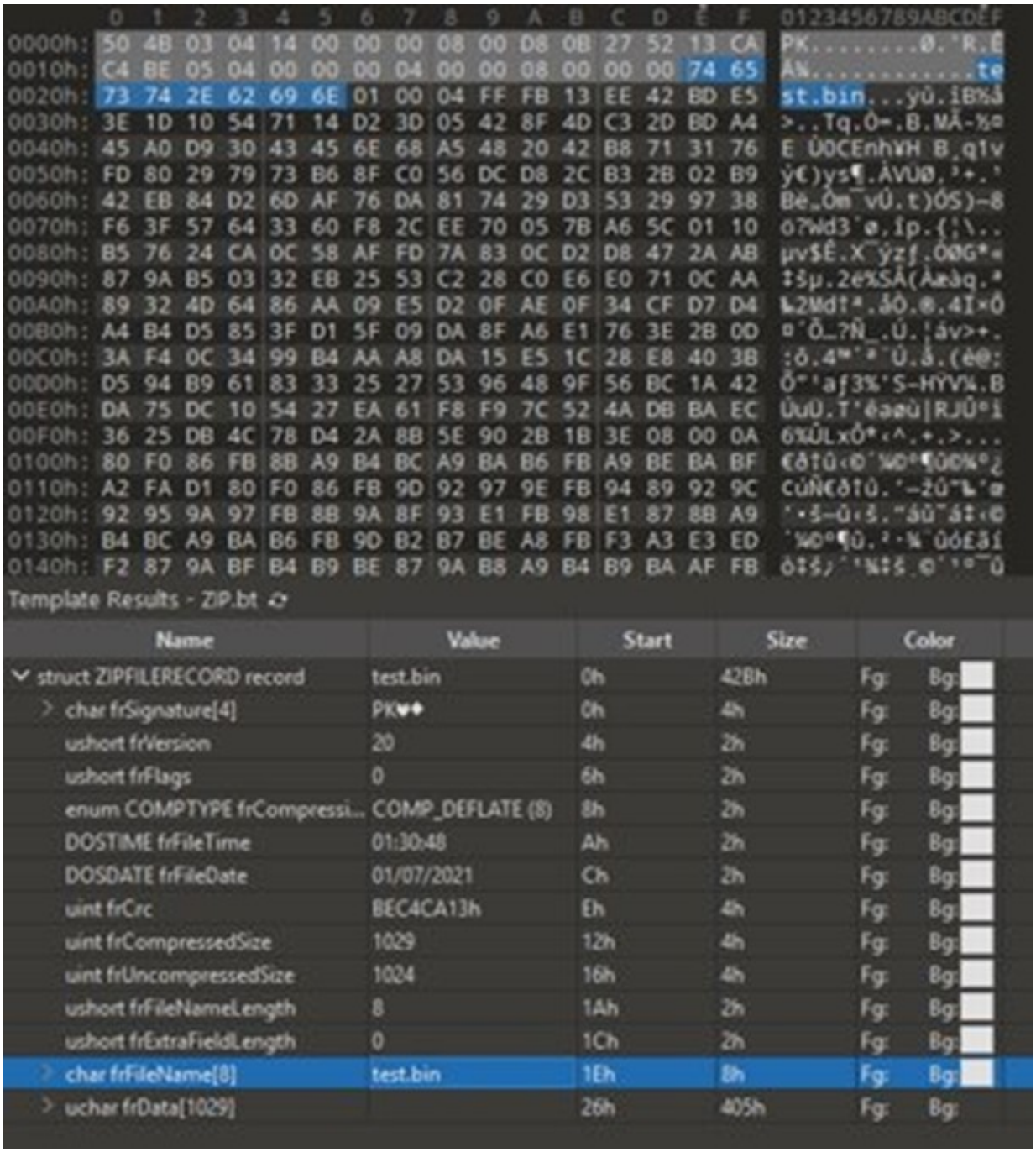


Figure 32. Passcode check routine in NUPAKAGE

SHA256	Passcode
634977a24e8fb2e3e82a0cddfe8d007375d387415eb131cce74ca03e0e93565f	notebook
c835577f1ddf66a957dd0f92599f45cb67e7f3ea4e073a98df962fc3d9a3fbe0	comeon
2937580b16e70f82e27cfbc3524c2661340b8814794cc15cb0d534f5312db0e0	update
c2f5a12ebaeb39d4861e4c3b35253e68e6d5dc78f8598d74bc85db21aeb504e8	comeon

Table 4. Passcodes in NUPAKAGE

After execution, NUPAKAGE will drop two files, *xxx.zip* and *xxx.z*. The file *xxx.zip* is a logging file with a fake ZIP header prepended at offset 0x0 and taking up the first 0x100 bytes. Starting from the offset 0x100, the logging strings are encrypted with a single byte in XOR operations as shown Figure 33.



```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0000h: 50 4B 03 04 14 00 00 00 08 00 D8 0B 27 52 13 CA PK........Ø.'R.Ê
0010h: C4 BE 05 04 00 00 00 04 00 00 08 00 00 00 74 65 Ä¾............te
0020h: 73 74 2E 62 69 6E 01 00 04 FF FB 13 EE 42 BD E5 st.bin...ÿû.îB½å
0030h: 3E 1D 10 54 71 14 D2 3D 05 42 8F 4D C3 2D BD A4 >..Tq.Ò=.B.MÃ-½¤
0040h: 45 A0 D9 30 43 45 6E 68 A5 48 20 42 B8 71 31 76 E Ù0CEnh¥H B¸q1v
0050h: FD 80 29 79 73 B6 8F C0 56 DC D8 2C B3 2B 02 B9 ý€)ys¶.ÀVÜØ,³+.¹
0060h: 42 EB 84 D2 6D AF 76 DA 81 74 29 D3 53 29 97 38 Bë„Òm¯vÚ.t)ÓS)—8
0070h: F6 3F 57 64 33 60 F8 2C EE 70 05 7B A6 5C 01 10 ö?Wd3`ø,îp.{¦\..
0080h: B5 76 24 CA 0C 58 AF FD 7A 83 0C D2 D8 47 2A AB µv$Ê.X¯ýzƒ.ÒØG*«
0090h: 87 9A B5 03 32 EB 25 53 C2 28 C0 E6 E0 71 0C AA ‡šµ.2ë%SÂ(Àæàq.ª
00A0h: 89 32 4D 64 86 AA 09 E5 D2 0F AE 0F 34 CF D7 D4 ‰2Md†ª.åÒ.®.4Ï×Ô
00B0h: A4 B4 D5 85 3F D1 5F 09 DA 8F A6 E1 76 3E 2B 0D ¤´Õ…?Ñ_.Ú.¦áv>+.
00C0h: 3A F4 0C 34 99 B4 AA A8 DA 15 E5 1C 28 E8 40 3B :ô.4™´ª¨Ú.å.(è@;
00D0h: D5 94 B9 61 83 33 25 27 53 96 48 9F 56 BC 1A 42 Õ”¹aƒ3%'S–HŸV¼.B
00E0h: DA 75 DC 10 54 27 EA 61 F8 F9 7C 52 4A DB BA EC ÚuÜ.T'êaøù|RJÛºì
00F0h: 36 25 DB 4C 78 D4 2A 8B 5E 90 2B 1B 3E 08 00 0A 6%ÛLxÔ*‹^.+.>...
0100h: 80 F0 86 FB 8B A9 B4 BC A9 BA B6 FB A9 BE BA BF €ð†û‹©´¼©º¶û©¾º¿
0110h: A2 FA D1 80 F0 86 FB 9D 92 97 9E FB 94 89 92 9C ¢úÑ€ð†û.'—žû”‰'œ
0120h: 92 95 9A 97 FB 8B 9A 8F 93 E1 FB 98 E1 87 8B A9 '•š—û‹š.“áû˜á‡‹©
0130h: B4 BC A9 BA B6 FB 9D B2 B7 BE A8 FB F3 A3 E3 ED ´¼©º¶û.²·¾¨ûó£ãí
0140h: F2 87 9A BF B4 B9 BE 87 9A B8 A9 B4 B9 BA AF FB ò‡š¿´¹¾‡š¸©´¹º¯û
```

Template Results - ZIP.bt

Name	Value	Start	Size	Color
struct ZIPFILERECORD record	test.bin	0h	42Bh	Fg: Bg:
char frSignature[4]	PK♥♦	0h	4h	Fg: Bg:
ushort frVersion	20	4h	2h	Fg: Bg:
ushort frFlags	0	6h	2h	Fg: Bg:
enum COMPTYPE frCompressi...	COMP_DEFLATE (8)	8h	2h	Fg: Bg:
DOSTIME frFileTime	01:30:48	Ah	2h	Fg: Bg:
DOSDATE frFileDate	01/07/2021	Ch	2h	Fg: Bg:
uint frCrc	BEC4CA13h	Eh	4h	Fg: Bg:
uint frCompressedSize	1029	12h	4h	Fg: Bg:
uint frUncompressedSize	1024	16h	4h	Fg: Bg:
ushort frFileNameLength	8	1Ah	2h	Fg: Bg:
ushort frExtraFieldLength	0	1Ch	2h	Fg: Bg:
char frFileName[8]	test.bin	1Eh	8h	Fg: Bg:
uchar frData[1029]		26h	405h	Fg: Bg:

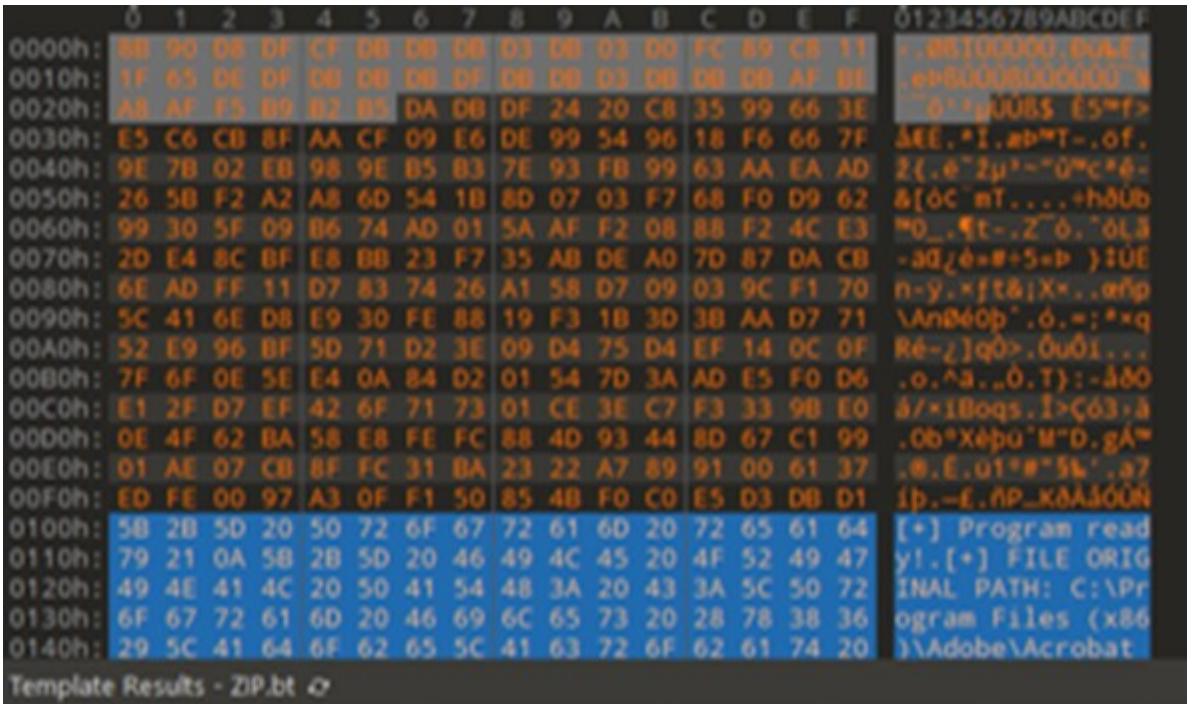


Figure 33. The original logging file (top), with plain text revealed in the decrypted logging file (bottom)

Taking one of the execution results as an example, much of the information of the exfiltrated data is saved, including the original file path, the original file size, and the compressed file size. We believe that the threat actors use it to further track which files have been processed. For security researchers, this logging file also helps reveal how much data is exfiltrated and provides information on the impact scope.

[+] Program ready!
[+] FILE ORIGINAL PATH: C:\Program Files (x86)\Adobe\Acrobat Reader DC\Reader\1494870C-9912-C184-4CC9-B401-A53F4D8DE290.pdf
[+] FILE PATH SIZE: 198
[+] FILE ORIGINAL SIZE: 186837
[+] FILE COMPRESSED SIZE: 183734
[+] FILE ORIGINAL PATH: C:\Program Files (x86)\Adobe\Acrobat Reader DC\Reader\Click on ‘Change’ to select default PDF handler.pdf
[+] FILE PATH SIZE: 210
[+] FILE ORIGINAL SIZE: 186837
[+] FILE COMPRESSED SIZE: 183734
...
<omitted>
...
[*] File or folder access denied!
[*] File or folder access denied!
[+] All completed!

The file with a .z extension is a blob of exfiltrated data within a self-defined file format. The NUPAKAGE malware first generates a key blob randomly, with the key being encrypted in a custom algorithm. After, it stores the encrypted key blob into the first 0x80 bytes of the file with the .z extension. Starting from the offset 0x80, there exists a long array of all the exfiltrated data.

Much of the information from the exfiltrated files are saved, such as the MD5 hash, the length of the file name, the compressed file size, the original file size, the file name, and the file's content. To separate the file blobs, it puts a unique byte sequence at the end of each, *55 55 55 55 AA AA AA AA FF FF FF FF 99 99 99 99*.

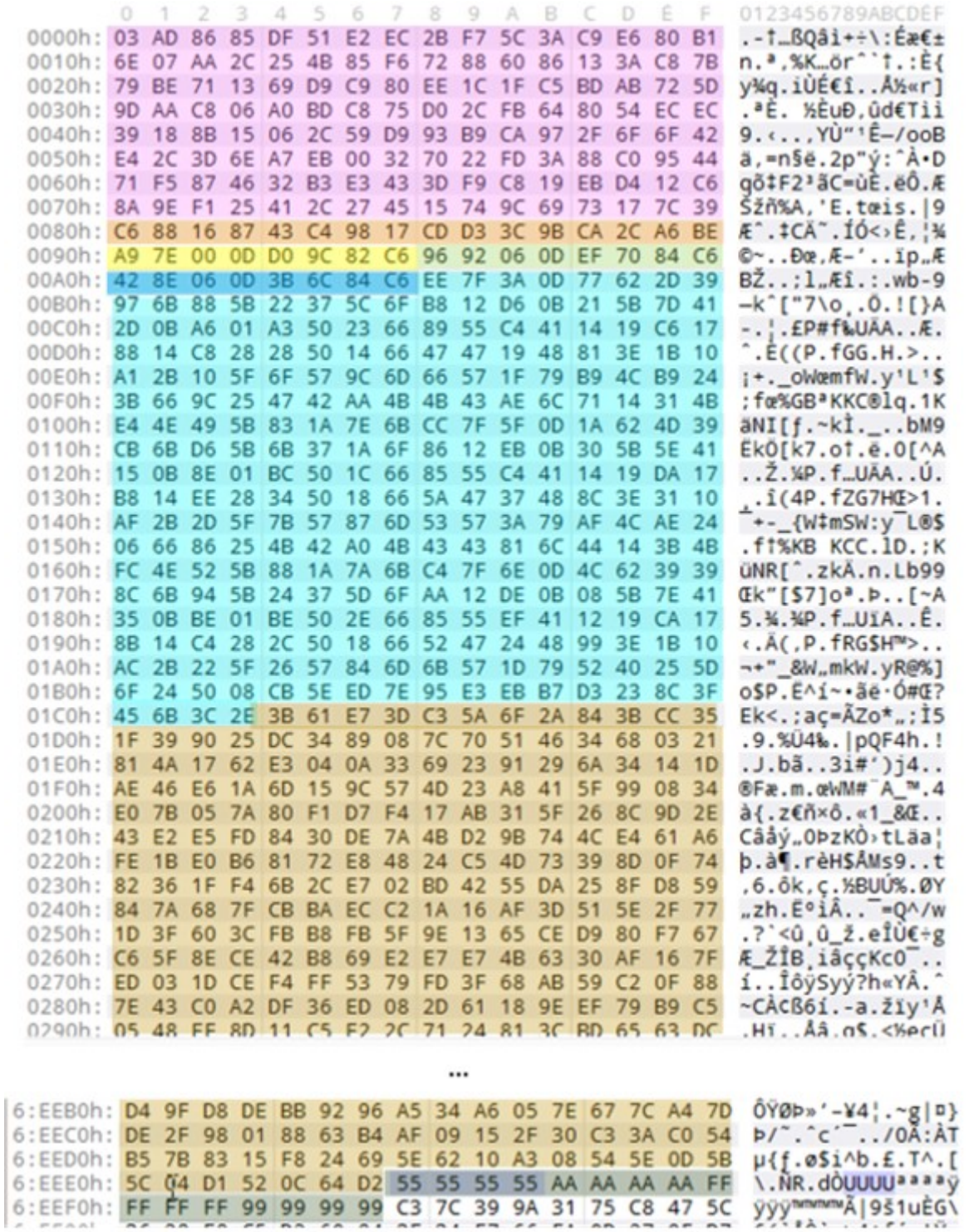


Figure 34. Self-defined format in the file with the .z extension generated by NUPAKAGE

Offset	Field Name	Size	Description
0x0	key	0x80	Encrypted Key
0x80	md5	0x10	MD5 (XORed with Decrypted Key)
0x90	len	0x8	The length of file name (XORed with Decrypted Key)
0x98	size2	0x8	Comepressed file size (XORed with Decrypted Key)
0xA0	size1	0x8	Original file size (XORed with Decrypted Key)
0xA8	file_name	len	File name (XORed with Decrypted KEy)
0xA8 + len	file_content	size2	File content (XORed with Decrypted Key)
...	...	...	...
0xA8 + len + size2	delimiter	0x10	The mark to tell the end of one file object 55 55 55 55 AA AA AA AA FF FF FF FF 99 99 99 99

Table 5. Self-defined format description in the file with the .z extension generated by NUPAKAGE

It's also worth mentioning that in the more recent versions of NUPAKAGE, an increasing number of obfuscations are being adopted to thwart static analysis.

```
memcpy_s(byte_422F50, 0x200u, Source, 0x70u);
v7 = ((((unsigned int)dword_41F300 >> 9) ^ (((unsigned int)dword_41F300 >> 9) - 122958665)) - 1217695470) >> 9;
v8 = (v7 ^ (v7 - 122958665)) - 1217695470;
for ( i = 0; i < 0x70; ++i )
{
  byte_422F50[i] ^= 0xAEu;
  v8 = (((v8 >> 9) - 122958665) ^ (v8 >> 9)) - 1217695470;
}
if ( (unsigned int)argc > 5 )
{
  v10 = 0;
  v64 = -279657982;
  v34 = 0;
  v65 = -15814;
  v8 = (((v8 >> 9) - 122958665) ^ (v8 >> 9)) - 1217695470;
  sub_401000(&v64);
  v11 = strcmp(argv[5], (const char *)&v64 + 3);
  if ( v11 )
    v11 = v11 < 0 ? -1 : 1;
  if ( !v11 )
  {
    v8 = (((v8 >> 9) - 122958665) ^ (v8 >> 9)) - 1217695470;
    memset(&SubStr, 0, 0x1000u);
    if ( (unsigned int)argc > 6 )
```

Figure 35. Junk codes in more recent versions of NUPAKAGE

HackTool.Win32.ZPAKAGE

ZPAKAGE is another example of custom malware used for packing files; it also works similarly to NUPAKAGE. It also needs a passcode to ensure that it is being used as intended. In the example shown in Figure 36, the passcode is “start”.

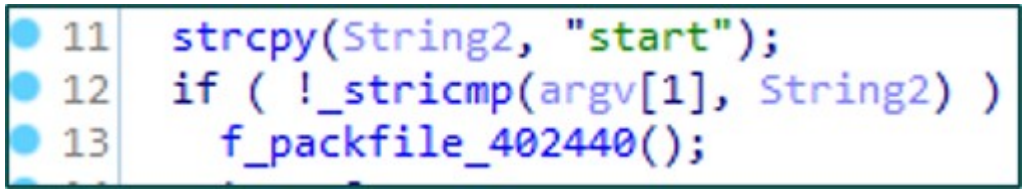


Figure 36. An example of a ZPAKAGE password

ZPAKAGE also supports command-line arguments, but it possesses less functions than NUPAKAGE. The usage of this tool is shown as follows:

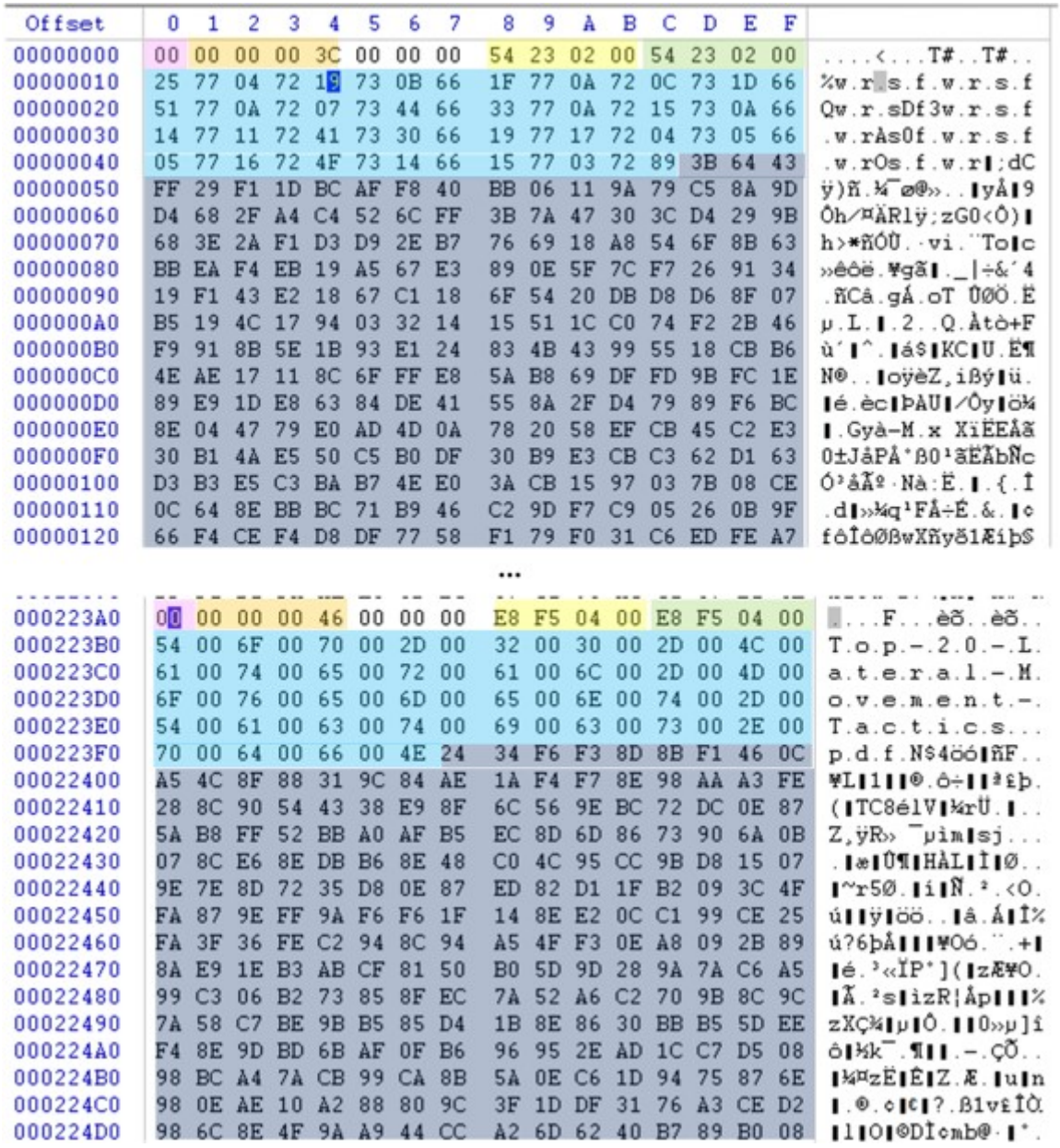
malware.exe ***passcode time***

Argument Name	Format	Example Value	Description
Passcode	String	start	A unique code in order to execute it
Time	String	20221221	The start date

Table 6. Arguments supported by ZPAKAGE

ZPAKAGE also shows similar behaviors to NUPAKAGE. For instance, it also avoids files with names starting with “$” or “~”. In addition, it generates two files, one with a .z extension and another with a .zip extension. The file with a .z extension is the exfiltrated data blob and the file with a .zip extension is the logging file.

In the generated file with a .z extension, the exfiltrated files will be compressed by the zlib algorithm to minimize the file size. It also defines a Boolean field “type” for storage, whether a file is compressed or not. If a file is compressed and its file size is less than the original one, the type will be 1. Otherwise, the type will be set to 0, and the original file content will be chosen instead of the compressed one. Regardless of whether the file content is compressed or not, it will be encrypted in XOR operations with a specific string, *qwerasdf*.



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000	00	00	00	00	3C	00	00	00	54	23	02	00	54	23	02	00
00000010	25	77	04	72	1F	73	0B	66	1F	77	0A	72	0C	73	1D	66
00000020	51	77	0A	72	07	73	44	66	33	77	0A	72	15	73	0A	66
00000030	14	77	11	72	41	73	30	66	19	77	17	72	04	73	05	66
00000040	05	77	16	72	4F	73	14	66	15	77	03	72	89	3B	64	43
00000050	FF	29	F1	1D	BC	AF	F8	40	BB	06	11	9A	79	C5	8A	9D
00000060	D4	68	2F	A4	C4	52	6C	FF	3B	7A	47	30	3C	D4	29	9B
00000070	68	3E	2A	F1	D3	D9	2E	B7	76	69	18	A8	54	6F	8B	63
00000080	BB	EA	F4	EB	19	A5	67	E3	89	0E	5F	7C	F7	26	91	34
00000090	19	F1	43	E2	18	67	C1	18	6F	54	20	DB	D8	D6	8F	07
000000A0	B5	19	4C	17	94	03	32	14	15	51	1C	C0	74	F2	2B	46
000000B0	F9	91	8B	5E	1B	93	E1	24	83	4B	43	99	55	18	CB	B6
000000C0	4E	AE	17	11	8C	6F	FF	E8	5A	B8	69	DF	FD	9B	FC	1E
000000D0	89	E9	1D	E8	63	84	DE	41	55	8A	2F	D4	79	89	F6	BC
000000E0	8E	04	47	79	E0	AD	4D	0A	78	20	58	EF	CB	45	C2	E3
000000F0	30	B1	4A	E5	50	C5	B0	DF	30	B9	E3	CB	C3	62	D1	63
00000100	D3	B3	E5	C3	BA	B7	4E	E0	3A	CB	15	97	03	7B	08	CE
00000110	0C	64	8E	BB	BC	71	B9	46	C2	9D	F7	C9	05	26	0B	9F
00000120	66	F4	CE	F4	D8	DF	77	58	F1	79	F0	31	C6	ED	FE	A7
...																
000223A0	00	00	00	00	46	00	00	00	E8	F5	04	00	E8	F5	04	00
000223B0	54	00	6F	00	70	00	2D	00	32	00	30	00	2D	00	4C	00
000223C0	61	00	74	00	65	00	72	00	61	00	6C	00	2D	00	4D	00
000223D0	6F	00	76	00	65	00	6D	00	65	00	6E	00	74	00	2D	00
000223E0	54	00	61	00	63	00	74	00	69	00	63	00	73	00	2E	00
000223F0	70	00	64	00	66	00	4E	24	34	F6	F3	8D	8B	F1	46	0C
00022400	A5	4C	8F	88	31	9C	84	AE	1A	F4	F7	8E	98	AA	A3	FE
00022410	28	8C	90	54	43	38	E9	8F	6C	56	9E	BC	72	DC	0E	87
00022420	5A	B8	FF	52	BB	A0	AF	B5	EC	8D	6D	86	73	90	6A	0B
00022430	07	8C	E6	8E	DB	B6	8E	48	C0	4C	95	CC	9B	D8	15	07
00022440	9E	7E	8D	72	35	D8	0E	87	ED	82	D1	1F	B2	09	3C	4F
00022450	FA	87	9E	FF	9A	F6	F6	1F	14	8E	E2	0C	C1	99	CE	25
00022460	FA	3F	36	FE	C2	94	8C	94	A5	4F	F3	0E	A8	09	2B	89
00022470	8A	E9	1E	B3	AB	CF	81	50	B0	5D	9D	28	9A	7A	C6	A5
00022480	99	C3	06	B2	73	85	8F	EC	7A	52	A6	C2	70	9B	8C	9C
00022490	7A	58	C7	BE	9B	B5	85	D4	1B	8E	86	30	BB	B5	5D	EE
000224A0	F4	8E	9D	BD	6B	AF	0F	B6	96	95	2E	AD	1C	C7	D5	08
000224B0	98	BC	A4	7A	CB	99	CA	8B	5A	0E	C6	1D	94	75	87	6E
000224C0	98	0E	AE	10	A2	88	80	9C	3F	1D	DF	31	76	A3	CE	D2
000224D0	98	6C	8E	4F	9A	A9	44	CC	A2	6D	62	40	B7	89	B0	08

Figure 37. Self-defined format in the file with .z extension generated by ZPAKAGE

Offset	Field Name	Size	Description
0x0	type	1	Compression type, 0x0 or 0x1
0x1	len	4	Length of filename
0x5	reserved	3	
0x8	size1	4	Original file size
0xC	size2	4	Compressed file size
0x10	file_name	len	Encoded filename (XOR with “qwerasdf”)
0x10 + len	file_content	size2	Encoded file content (zlib + XOR with “qwerasdf”)
…		…	…

Table 7. Self-defined format description in the file with the .z extension generated by ZPAKAGE

Threat hunting

Since October 2022, the threat actors have changed their TTPs and have started using password-protected archives. For example, we found a TONEINS sample (SHA256: *8b98e8669d1ba49b66c07199638ae6012adf7d5d93c1ca3bf31d6329506da58a*) on VirusTotal that can't be linked to any other file in the "Relations" tab. However, we observed two files that have been opened in the "Behaviors" tab with the file names *~$Evidence information.docx* and *~$List of terrorist personnel at the border.docx*. As mentioned in the arrival vectors section, the next stage payloads are normally embedded in the fake document files.

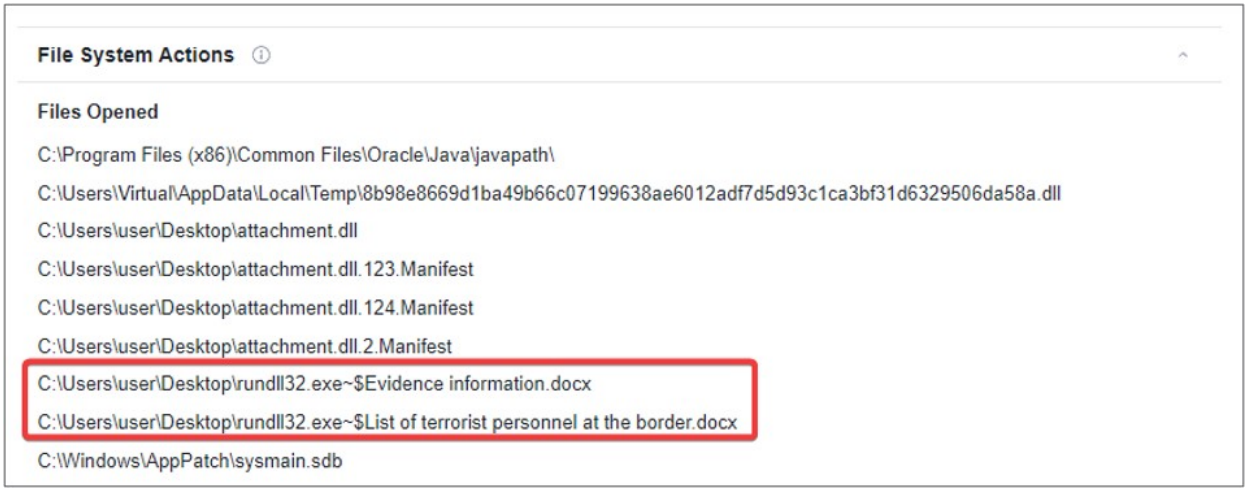


Figure 38. Opened files of TONEINS sample

Figure 39 shows the search results for the query "List of terrorist personnel at the border" on VirusTotal. The first file is the TONEINS DLL sample that we mentioned earlier in this section, while the second file is a benign executable file originally named *adobe_licensing_wf_helper.exe*, which was apparently uploaded to VirusTotal with the file name *List of terrorist personnel at the border.exe*.

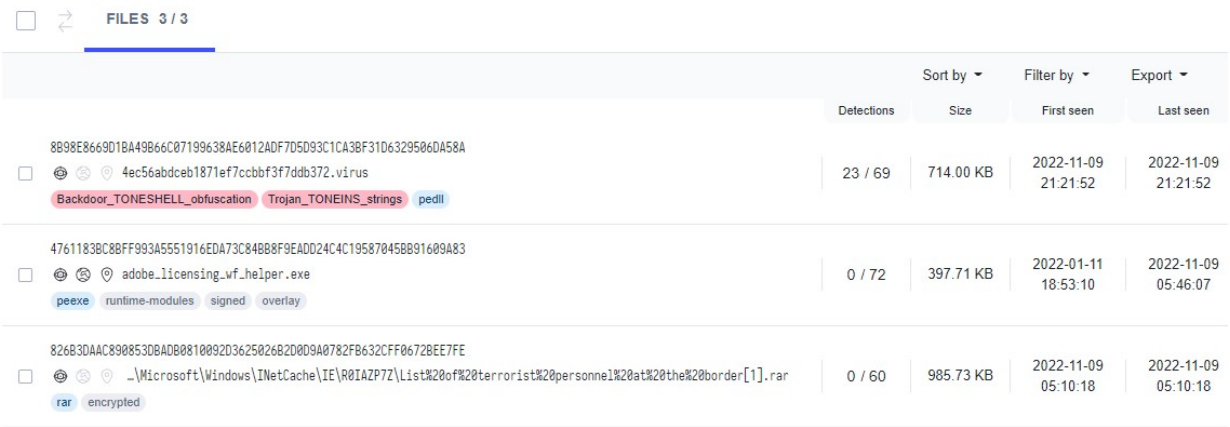


Figure 39. Search result for the string List of terrorist personnel at the border on VirusTotal

Submissions

Date	Name	Source	Country
2022-01-11 18:53:10 UTC	adobe_licensing_wf_helper_acro.exe	63b1639b - api	US
2022-07-14 09:07:27 UTC	Adobe_licensing_wf.exe	5a86d8ac - web	SG
2022-07-17 09:57:50 UTC	4761183bc8bff993a5551916eda73c84bb8f9eadd24c4c19587045bb91609a83	91b0bd83 - api	CN
2022-07-20 06:33:26 UTC	Adobe_licensing_wf.exe	1021f170 - web	EG
2022-08-02 07:58:45 UTC	22-6-2022 Inter(en).exe	8151a3ef - web	SG
2022-08-25 23:18:29 UTC	01. 9th SST Agreed Minutes(English)(2).exe	71612067 - web	AU
2022-10-10 05:33:48 UTC	Notic(20221010)(final).exe	0efa7a0c - web	KR
2022-10-17 04:53:20 UTC	help letter.exe	71612067 - web	AU
2022-10-18 02:36:15 UTC	4761183bc8bff993a5551916eda73c84bb8f9eadd24c4c19587045bb91609a83	a0963584 - web	AU
2022-10-26 06:25:06 UTC	file	0efa7a0c - web	US
2022-10-27 03:45:48 UTC	26-10-2022.exe	71612067 - web	AU
2022-11-09 05:46:07 UTC	List of terrorist personnel at the border.exe	17414873 - web	KR

Figure 40. Submission of adobe_licensing_wf_helper.exe

The third file is a password-protected archive, which has the exact same file name, *List of terrorist personnel at the border[1].rar*. Unfortunately, we didn't have the password, so we were unable to decompress it. But it has an interesting execution parent in the "Relations" tab, which is a document file named *Letter Head.docx*.

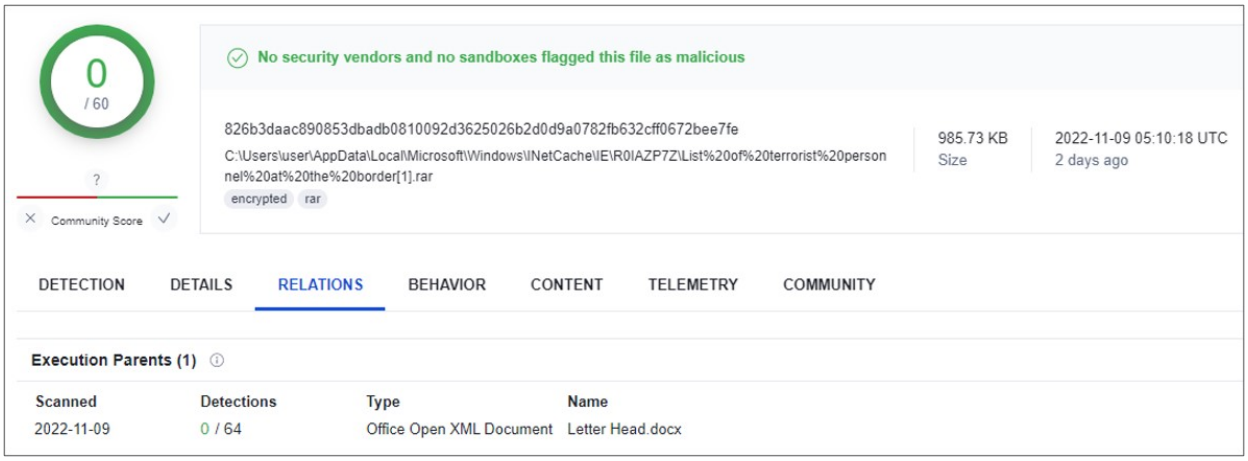


Figure 41. Execution parent of List of terrorist personnel at the border[1].rar

Inside the document *Letter Head.docx*, there is a Google Drive link and a password. The content itself is related to the Government of the Republic of the Union of Myanmar, and is written in Burmese.

Figure 42. Letter Head.docx

Upon checking the download link, we discovered that it was the same password-protected archive file that we found on VirusTotal earlier.

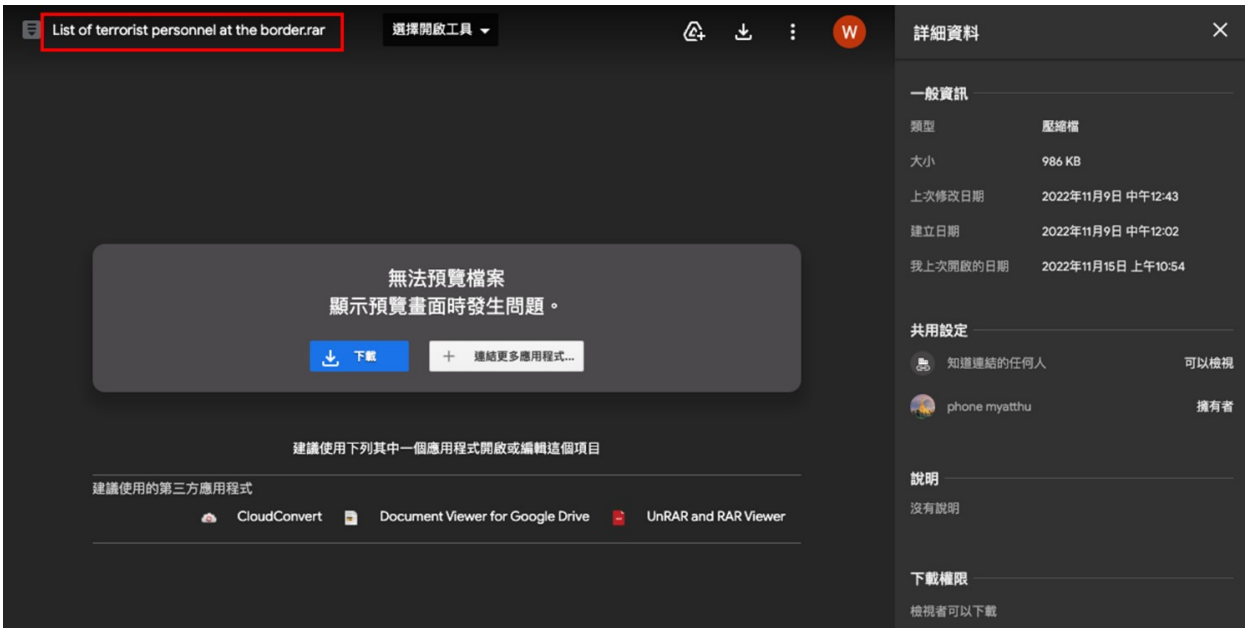


Figure 43. Screenshot of the Google Drive link

The new arrival vector flow is similar to the one we introduced in the arrival vector section: Victims will receive and interact with a decoy document containing a Google Drive link and a corresponding password instead of an archive download link embedded in the email.

As for why the password-protected archive has the execution parent, upon checking the sandbox execution behaviors of *Letter Head.docx* on VirusTotal, we discovered that the VirusTotal sandbox will

select any link embedded in the document. This leads to the opening of an Internet Explorer window with the file download prompt.

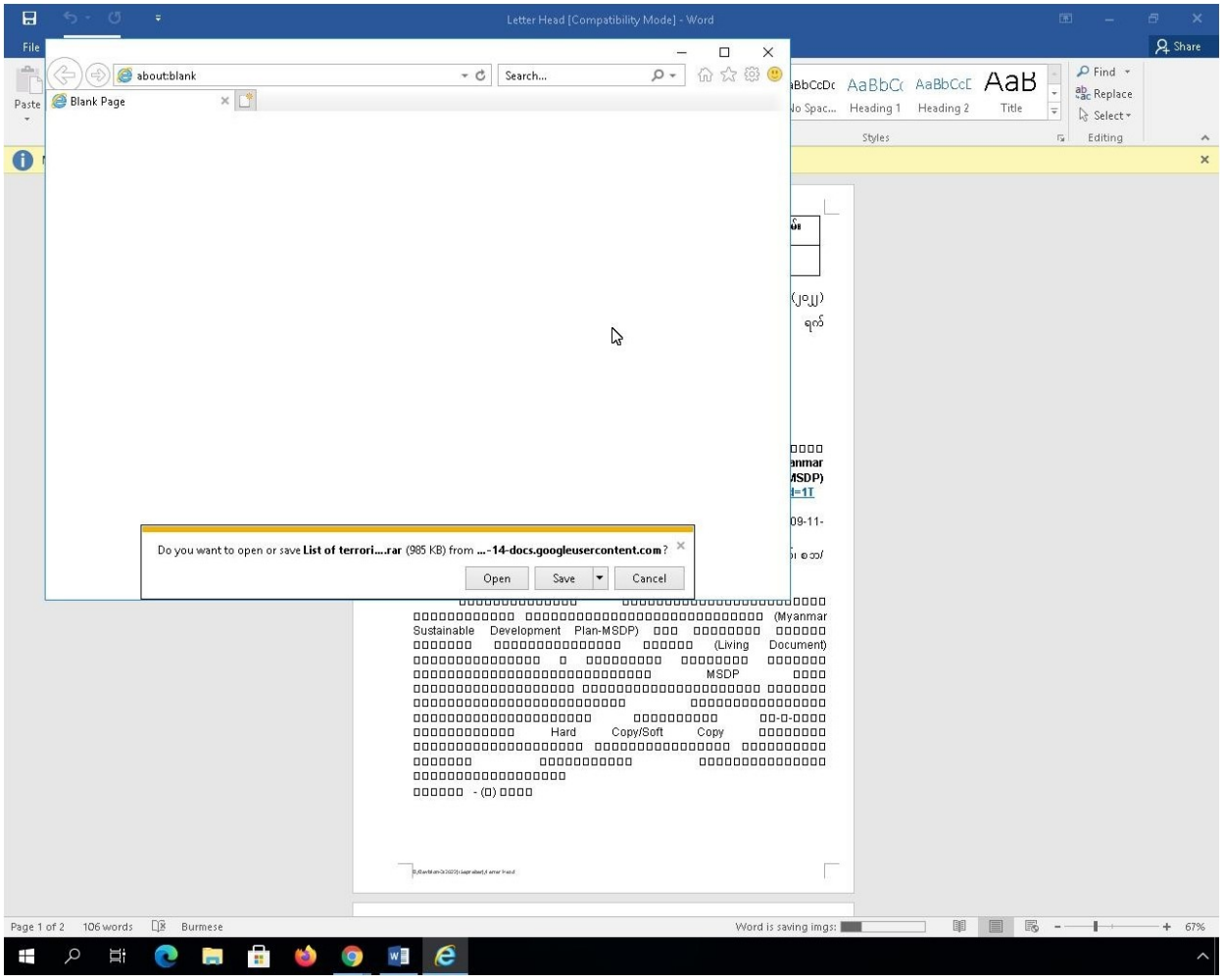


Figure 44. Sandbox screenshot of the file Letter Head.docx on VirusTotal

When the download prompt is shown, Internet Explorer will silently download this file in the background even before the user selects the “Save” button.

As a result, the file will be saved to the cache folder named “INetCache,” after which we see a dropped RAR file:

- *C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\IE\R0IAZP7Z\List%20of%20terrorist%20personnel%20at%20the%20border[1].rar*.

Since the RAR file is downloaded automatically by Internet Explorer, *Letter Head.docx* will be treated as its execution parent. This sample can then be used for hunting this campaign.

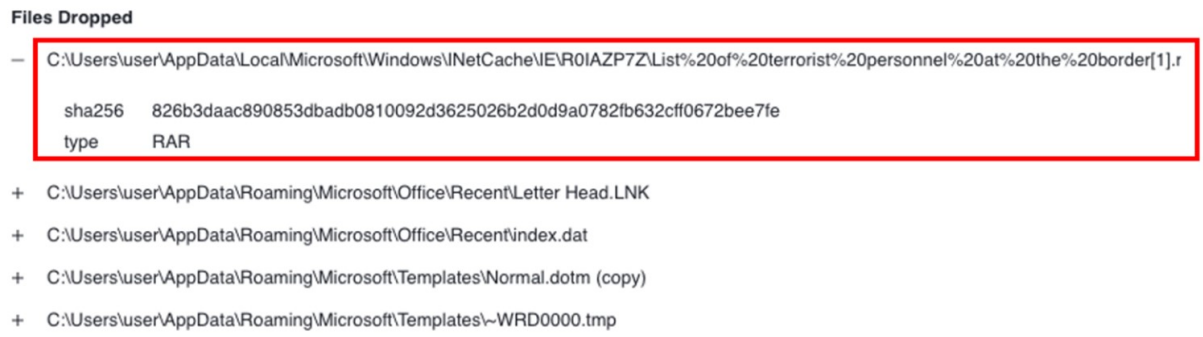


Figure 45. The dropped files of Letter Head.docx on VirusTotal

To find additional password-protected archives and documents embedded with a Google Drive link, we tried to use the following query:

tag:rar tag:encrypted name:INetCache size:500kb+

The query finds any encrypted RAR archive with a large enough file size containing the folder name “INetCache” in its path. Fortunately, we found another RAR file with the document execution parent “*Notic(20221010)(final).docx”* that turned out to be a TONESHELL archive.

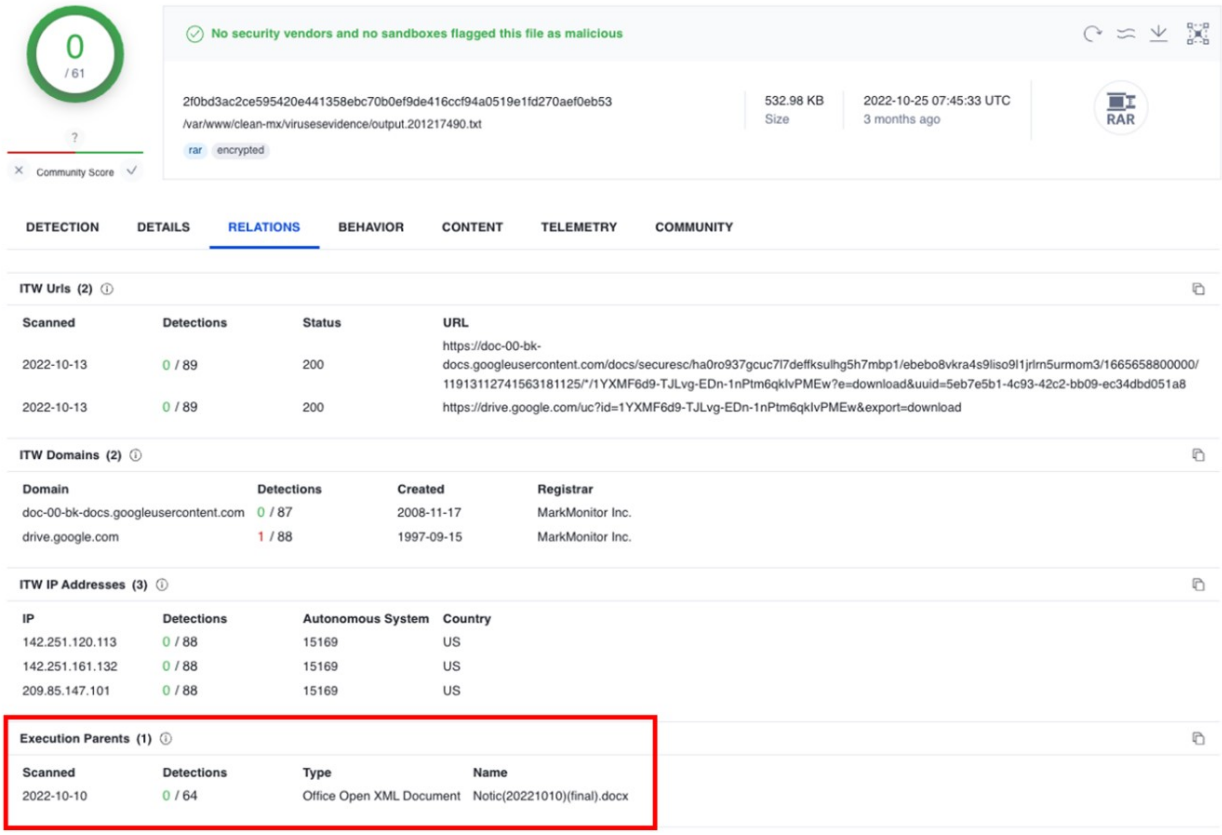


Figure 46. Relations of the archive file

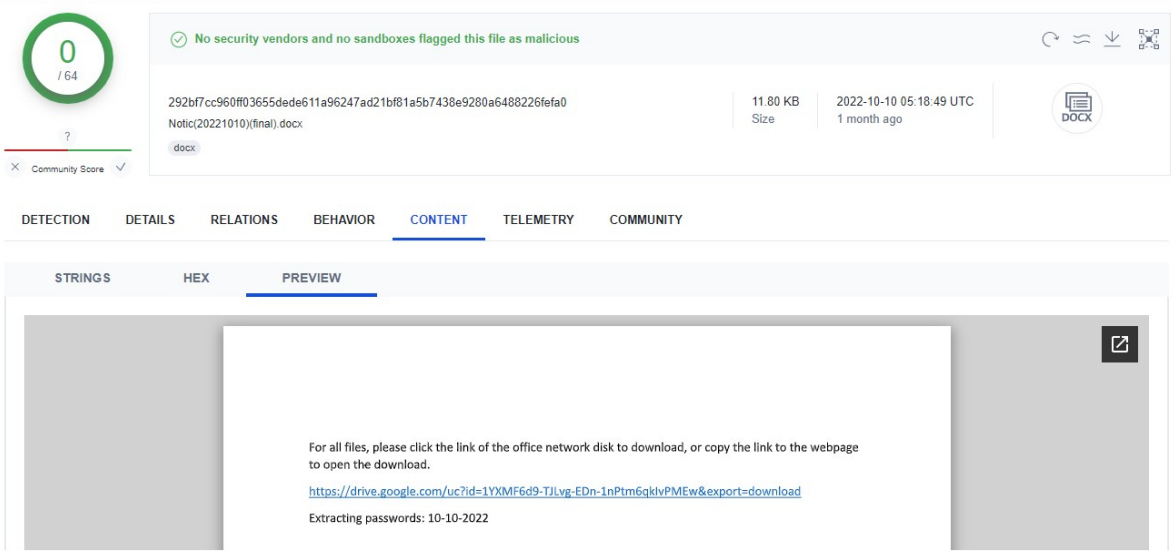


Figure 47. Content of the file Notic(20221010)(final).docx

It's interesting to note that the threat actors use date and time strings written in the same format (DD-MM-YYYY) as the extracting passwords in all the cases we've collected so far.

Connecting the dots

During our investigation, we observed some data points that connect to the same personnel. For example, we found a specific name "TaoZongjie" among the different malware samples we collected. In addition, the GitHub repository named "YanNaingOo0072022," mentioned in Avast's December 2022 report,hosted multiple pieces of malware, including TONESHELL. We also observed that the obfuscation methods have similarities among the different malwares.

User "TaoZongjie"

We found some samples sharing the same special string/name "TaoZongjie," including the Cobalt Strike malware, a Windows user on a TONESHELL C&C server, and the displayed message in the pop-up dialog box of TONESHELL.

Our investigation started with the TONESHELL C&C server 38[.]54[.]33[.]228 that had the remote desktop service enabled. Here, we found that one of the Windows users was called "TaoZongjie."

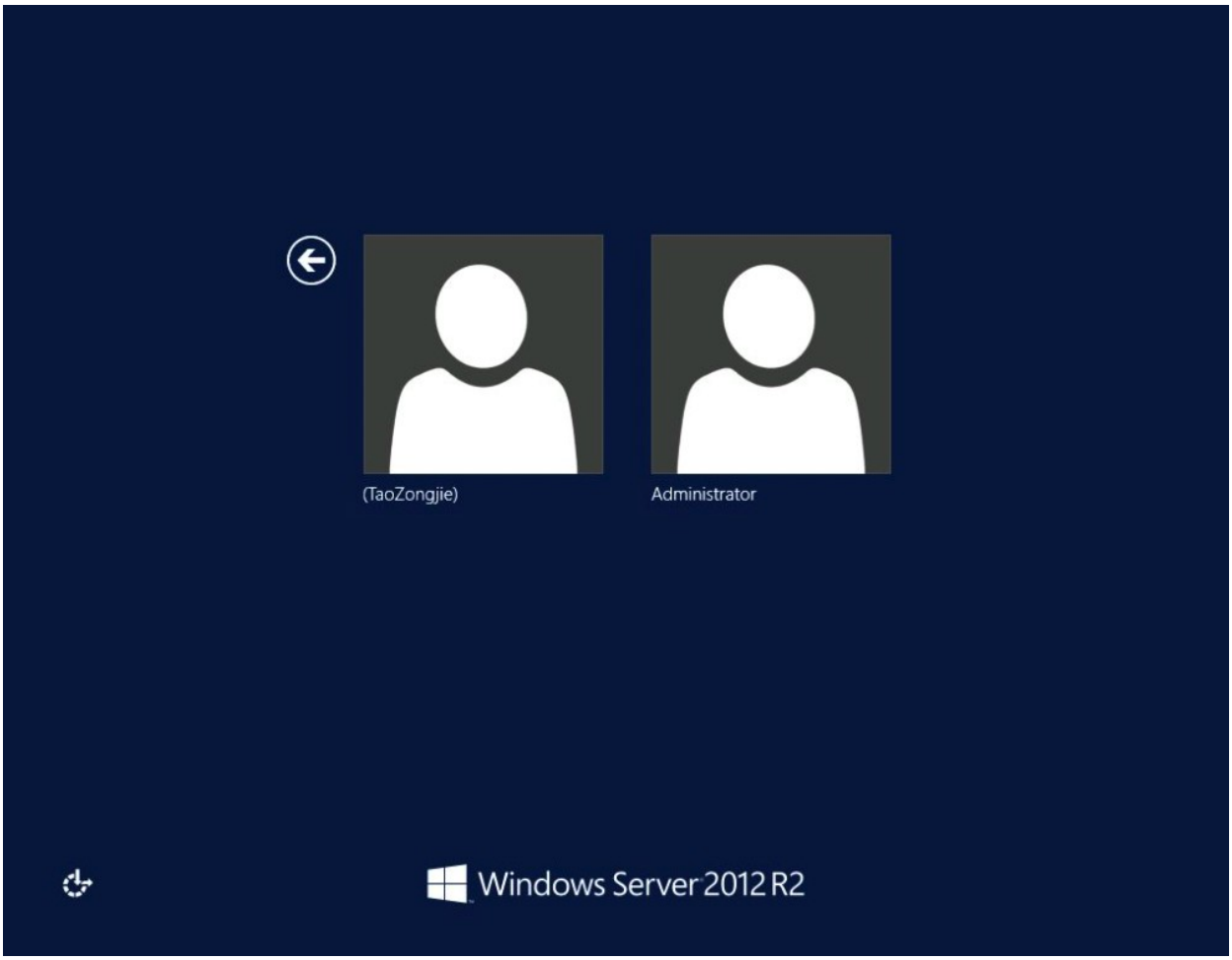


Figure 48. Windows users in 38[.]54[.]33[.]228

While hunting samples related to this campaign, we came across a tweet about Cobalt Strike posted in April 2021. At first glance, Cobalt Strike was used in a manner similar to this campaign, including the use of DLL sideloading, the use of a Google Drive link for delivery, and the creation of a schedule task.

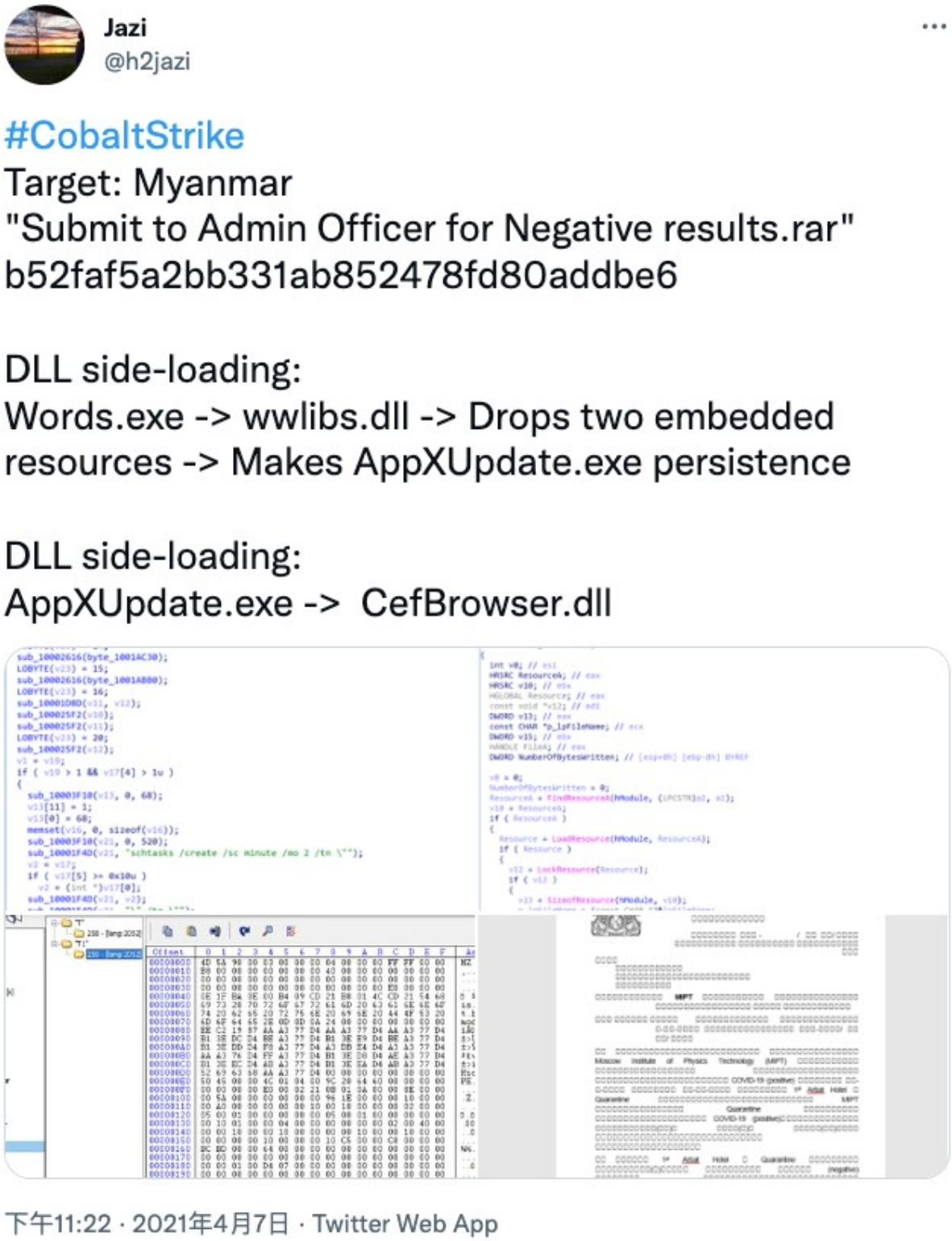


Figure 49. A tweet about Cobalt Strike

The infection flow is as follows: The archive file is delivered through a Google Drive link, which contains a legitimate EXE file, a malicious DLL file, and a decoy document written in Burmese. Once the malicious DLL is sideloaded, it will drop the legitimate EXE file and the malicious DLL file, which are embedded in the resource section of the DLL file. In this sample, the string *By:Taozongjie* is being used as the event name.

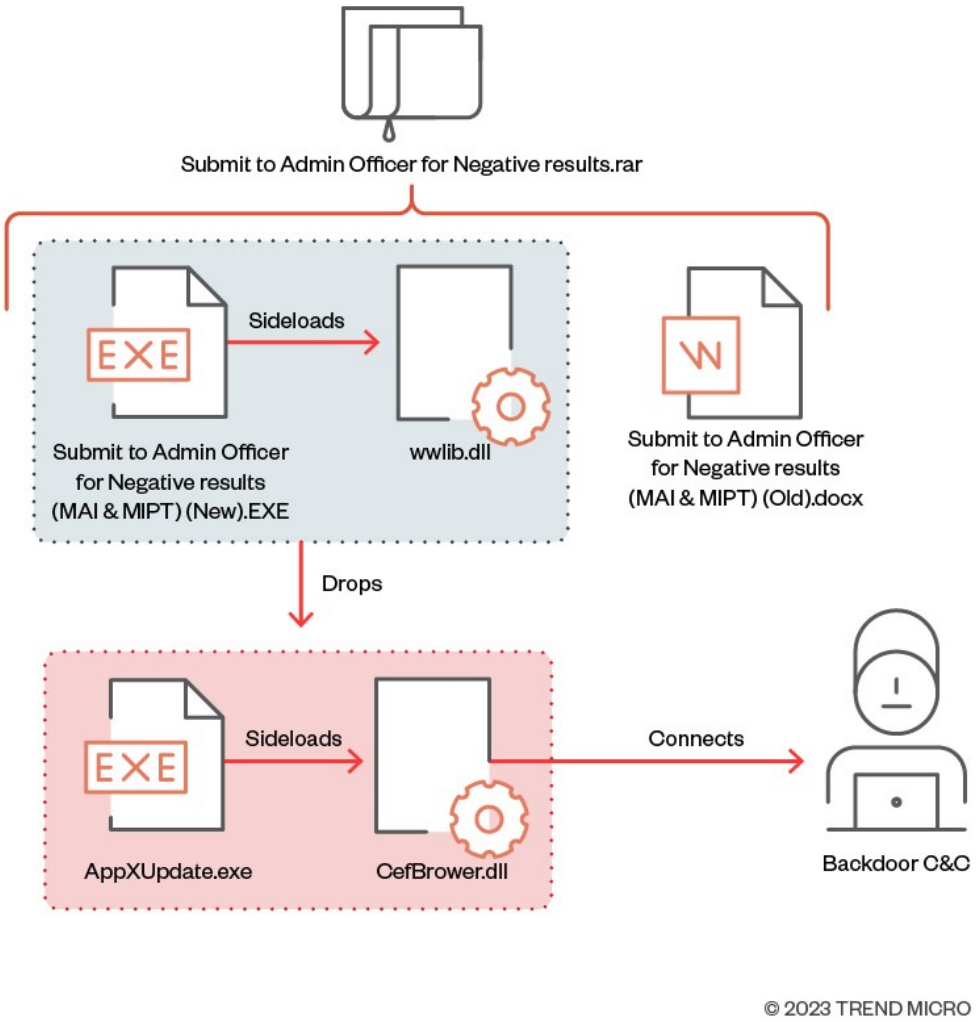


Figure 50. Infection flow of Cobalt Strike

```
void __usercall sub_100014C2(_BYTE *a1@<eax>)
{
  _BYTE *v1; // esi
  unsigned int v2; // eax
  CHAR *v3; // edi
  int v4; // eax
  signed int v5; // [esp+Ch] [ebp-4h]

  v1 = a1;
  v2 = _time64(0);
  srand(v2);
  v3 = (CHAR *)(MultiByteStr - v1);
  v5 = 32;
  do
  {
    CreateEventA(0, 1, 0, "By:Taozongjie");
    SetErrorMode(9u);
    SetConsoleMode(0, 4u);
    v4 = rand();
    v1[(_DWORD)v3] = *(_BYTE *)sub_10001480(v4 % 126 + 1);
    *v1 ^= 0x7Du;                                   // xor 7D
    SetConsoleTitleA("+8618087758761");
    SetLastError(0xEu);
    ++v1;
    --v5;
  }
  while ( v5 );
}
```

Figure 51. Special string in the sample

In one TONEINS sample (SHA256: *7436f75911561434153d899100916d3888500b1737ca6036e41e0f65a8a68707*), we also observed the string *taozongjie*, which was being used for an event name.

```
.text:100057DF
.text:100057DF loc_100057DF:
.text:100057DF mov     ecx, [ebp+var_13DC]
.text:100057E5 mov     word ptr [ecx+32h], 0
.text:100057EB mov     [ebp+var_10], 1
.text:100057F2 lea     eax, aCreateeventa ; "CreateEventA"
.text:100057F8 mov     [esp+1518h+pExceptionObject], ecx
.text:100057FB mov     [esp+1518h+pThrowInfo], eax
.text:100057FF call    sub_1000CB70
.text:10005804 mov     [ebp+var_13EC], eax
.text:1000580A lea     eax, aTaozongjie ; "taozongjie"
.text:10005810 mov     [ebp+var_600], eax
.text:10005816 lea     eax, [ebp+var_F0C]
.text:1000581C mov     [ebp+var_604], eax
.text:10005822 mov     ecx, [ebp+var_604]
```

Figure 52. Create event taozongjie in TONEINS

In another TONESHELL sample (SHA256: *d950d7d9402dcf014d6e77d30ddd81f994b70f7b0c6931ff1e705abe122a481a*), there are some insignificant export functions, which will appear via message boxes, with the strings *Tao* or *zhang!*. Even though the names of these two strings are not spelled exactly same way as *taozongjie*, their spellings are still similar.

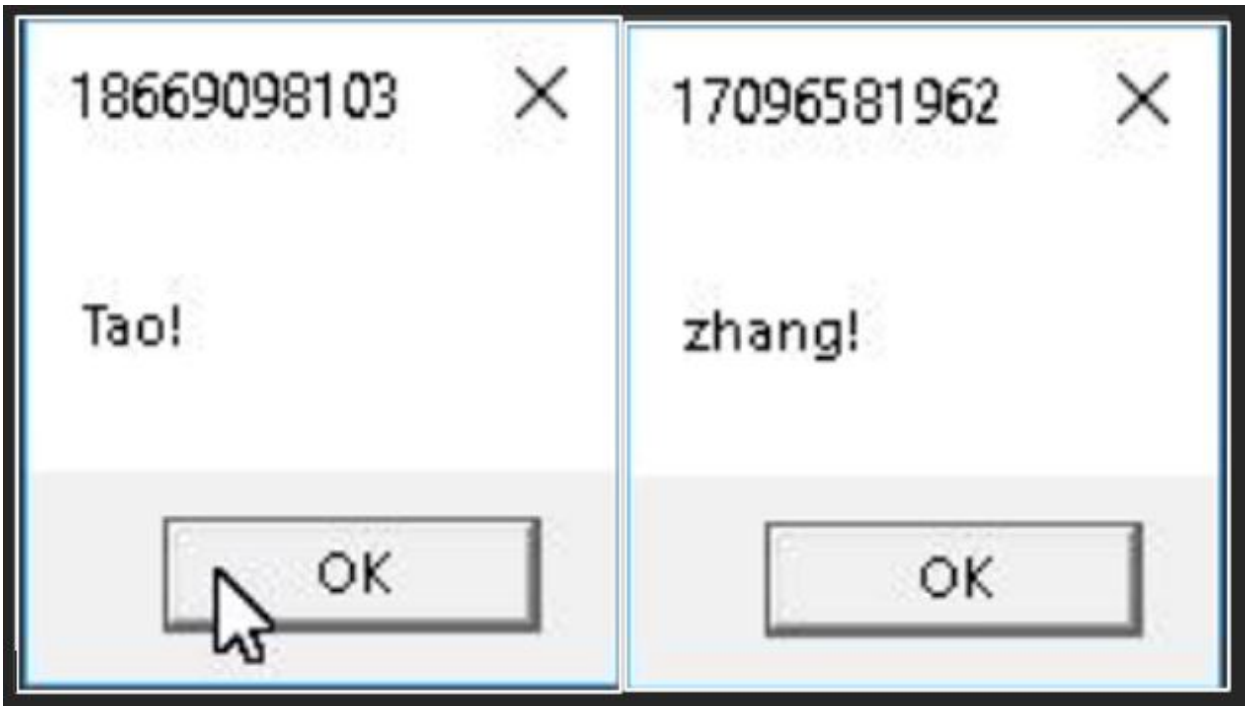


Figure 53. Message boxes of TONESHELL

Based on what we found among the different samples, we assume that *taozongjie* could be one of the flags used by the threat actors.

GitHub user “YanNaingOo0072022”

The GitHub user “YanNaingOo0072022” was mentioned in both an Avast and an ESET report. The user’s repositories host various malware, including the latest versions of TONEINS, TONESHELL, and a new tool, QMAGENT, which is ESET named MQsTTang”. At the time of writing, this GitHub space was still accessible, with five repositories: “View2015,” “View2016,” “1226,” “ee,” and “14.” Among these, “View2015” and “View2016” were empty.

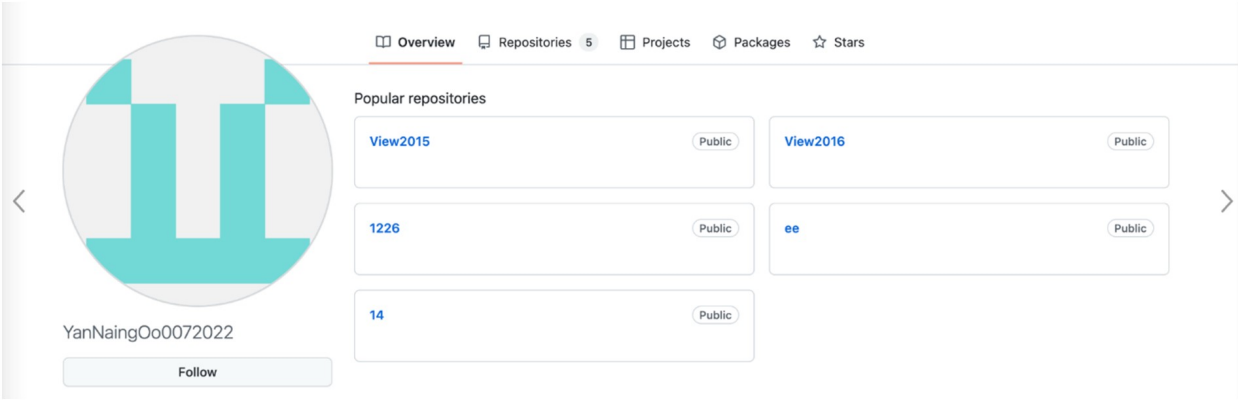


Figure 54. The YanNaingOo0072022 GitHub space

1226

The archive files in this repository are all the same but have different file names. We believe that these files were meant for different victims.

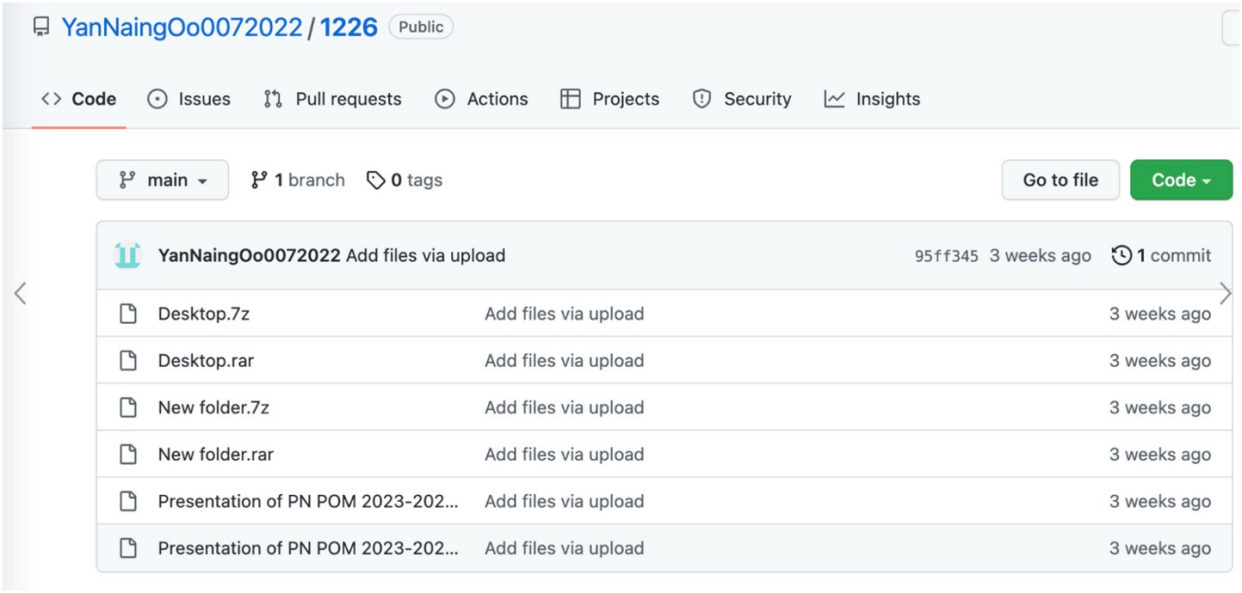


Figure 55. The 1226 repository

Upon unarchiving the compressed file, we found two files with the fake extension ".doc" containing one-byte XOR encrypted sections. Both share the same file structure (a PE payload hidden in a DOCX file) as the one we referred to in the arrival vectors section. These files ended up being the TONEINS and TONESHELL malware.

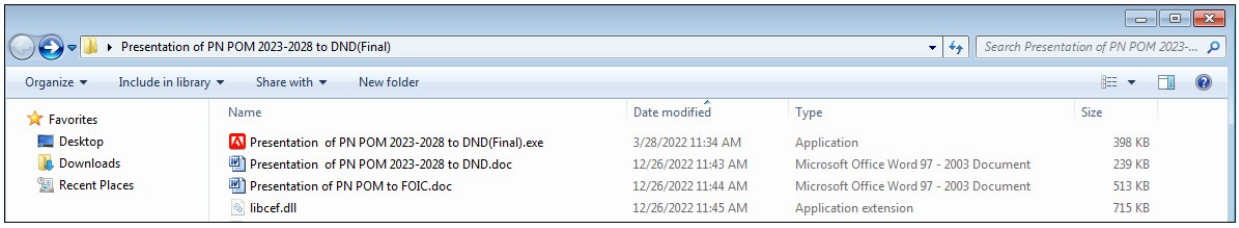


Figure 56. The files inside the archive

14

The file *Documents members of delegation diplomatic from Germany.Exe*, found in the *Documents.rar* archive, is a novel malware that communicates over the MQTT protocol. In March 2023, ESET published a detailed technical report on this backdoor, which it named "MQsTTang."

Beginning in January, we discovered that MQsTTang was being used as the new arrival vector in some of incidents we encountered, specifically in campaigns targeting individuals involved with government entities. This backdoor is unique because it communicates to its C&C servers over the MQTT protocol, which is commonly used in internet-of-things (IoT) devices. Malicious actors using this technique can effectively hide the real C&C server behind the protocol.

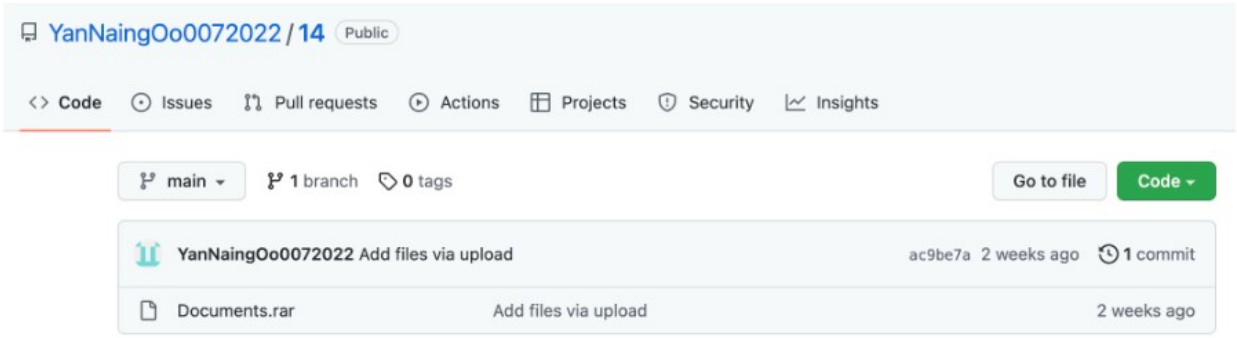


Figure 57. The 14 repository

ee

The file *CVs Amb Office PASSPORT Ministry Of Foreign Affairs.exe*, which is the malware QMAGENT, can be found in the *CVs Amb.rar* archive.

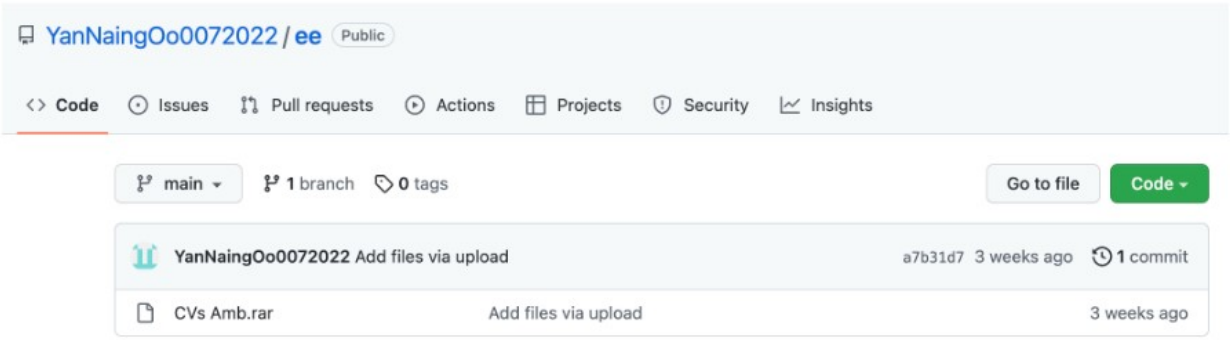


Figure 58. The ee repository

Conclusion

Over the past year, security researchers have been discovering and analyzing Earth Preta's campaigns and toolsets. We were able to attribute some of these to Earth Preta based on similarities among the TTPs, the malware being used, and the timeline of the campaigns. Starting October 2022, the threat actors changed the arrival vector of the TONEINS, TONESHELL, and PUBLOAD malware. Instead of attaching malicious archives or Google Drive links to an email, they now embed the download link in another decoy document and add a password to the archive.

Based on our observations, Earth Preta tends to hide malicious payloads in fake files, disguising them as legitimate ones — a technique that has been proven effective for avoiding detection. As for privilege escalation, the threat actors tend to reuse codes copied from open-source repositories. Meanwhile, they developed customized toolsets designed to collect confidential documents in the exfiltration stage.

Overall, we believe that Earth Preta is a capable and organized threat actor that is continuously honing its TTPs, strengthening its development capabilities, and building a versatile arsenal of tools and malware.

To help prevent potential threats such as the one posed by advanced persistent threat (APT) groups, we suggest that organizations conduct phishing awareness training for their employees and partners to stress the importance of caution when opening emails, particularly those messages from unfamiliar senders or with unknown subjects.

To assist organizations in protecting themselves against sophisticated threats, we recommend adopting a comprehensive security strategy that employs advanced technologies capable of identifying and halting such threats across multiple channels, including endpoints, servers, networks, and email communications.

Indicators of Compromise (IOCs)

The full list of IOCs can be found here.

MITRE ATT&CK

Tactic	ID	Name
Resource Development	T1583.004	Acquire Infrastructure: Server
	T1587.001	Develop Capabilities: Malware
	T1585.002	Establish Accounts: Email Accounts
	T1588.002	Obtain Capabilities: Tool
	T1608.001	Stage Capabilities: Upload Malware
Initial Access	T1566.002	Phishing: Spearphishing Link
Execution	T1204.001	User Execution: Malicious Link
	T1204.002	User Execution: Malicious File
Persistence	T1547.001	Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder
	T1574.002	Hijack Execution Flow: DLL Side-Loading
	T1053.005	Scheduled Task/Job: Scheduled Task
Privilege Escalation	T1068	Exploitation for Privilege Escalation
	T1134	Access Token Manipulation
Defense Evasion	T1140	Deobfuscate/Decode Files or Information
	T1036.005	Masquerading: Match Legitimate Name or Location
Lateral Movement	T1091	Replication Through Removable Media
Command and Control	T1071.001	Application Layer Protocol: Web Protocols
	T1573.001	Encrypted Channel: Symmetric Cryptography
	T1104	Multi-Stage Channels
	T1095	Non-Application Layer Protocol
Exfiltration	T1048	Exfiltration Over Alternative Protocol

Tags