Spot the Difference: Earth Kasha's New LODEINFO Campaign And The Correlation Analysis With The APT10 Umbrella

By By: Trend Micro Nov 19, 2024 Read time: 19 min (5210 words)

Published: 2024-11-19 · Archived: 2026-04-02 11:22:18 UTC

This blog is based on a presentation by the authors at Virus Bulletin 2024open on a new tab.

Introduction

LODEINFO is a malware used in attacks targeting mainly Japan since 2019. Trend Micro has been tracking the group as Earth Kasha. While some vendors suspect that the actor using LODEINFO might be APT10open on a new tab, we don’t have enough evidence to fully support this speculation. Currently, we view APT10 and Earth Kasha as different entities, although they might be related. To avoid confusion caused by names, we use a new term “APT10 Umbrella," which represents a group of intrusion sets related to APT10 (including APT10 itself).

Earth Kasha has been known to have targeted public institutions and academics with spear-phishing emails since their emergence. From early 2023 to early 2024, however, we identified a new campaign with significant updates to their strategy, tactics, and arsenals.

LODEINFO Since 2023

In the new campaign starting in early 2023, Earth Kasha expanded their targets into Japan, Taiwan, and India. Based on the bias of the incident amount, while we believe that Japan is still the main target of Earth Kasha, we observed that a few high-profile organizations in Taiwan and India were targeted. The observed industries under attack are organizations related to advanced technology and government agencies.

Earth Kasha has also employed different Tactics, Techniques, and Procedures (TTPs) in the Initial Access phase, which now exploits public-facing applications such as SSL-VPN and file storage services. We observed that vulnerabilities of enterprise products, such as Array AG (CVE-2023-28461)open on a new tab, Proself (CVE-2023-45727)open on a new tab and FortiOS/FortiProxy (CVE-2023-27997)open on a new tab, were abused in the wild. Earth Kasha was changing these vulnerabilities to abuse from time to time. After gaining access, they deployed several backdoors in the victim's network to achieve persistence. These include Cobalt Strike, LODEINFO, and the newly discovered NOOPDOOR, which we will describe later.

Observed TTPs in Post-Exploitation

Our comprehensive analysis of the activities in the Post-Exploitation phase has revealed that the primary motivation behind the attack was the theft of the victim’s information and data. Earth Kasha first discovered Active Directory configuration and domain user information to achieve this goal using legitimate Microsoft tools, such as *csvde.exe*, *nltest.exe* and *quser.exe*. The following are actual commands used by the adversary.

- csvde.exe -f all.csv –u
- nltest.exe /domain_trusts
- quser.exe

They then accessed the file server and tried to find documents related to the system information of the customer's network by simply using "dir" commands recursively. Interestingly, upon checking on their activity, the operator might check the content of the documents manually. The stolen information may help the adversary find the next valuable target.

Earth Kasha then performs several techniques to acquire credentials. One method uses their custom malware, MirrorStealer, to dump stored credentials in applications. MirrorStealer (originally reported by ESETopen on a new tab) is a credential dumper targeting multiple applications such as browsers (Chrome, Firefox, Edge and Internet Explorer), email clients (Outlook, Thunderbird, Becky, and Live Mail), Group Policy Preferences and SQL Server Management Studio.

Since MirrorStealer may be designed to dump credentials on client machines, Earth Kasha used another way to dump OS credentials. We observed that the adversary abused *vssadmin* to copy registry hives and *ntds.dit* in the Active Directory server from volume shadow copy. The SAM registry hive contains the NTLM hash of local machine users, while *ntds.dit* contains the NTLM hash of all the domain users. The following are commands the adversary uses after creating a volume shadow copy.

- copy \\<AD_SERVER_IP>\c$\windows\temp\ntds.dit .
- copy \\<AD_SERVER_IP>\c$\windows\temp\system .
- copy \\<AD_SERVER_IP>\c$\windows\temp\sam .

While we couldn’t figure out the actual method they abused, we have observed that Earth Kasha successfully compromised domain admin in most cases. After compromising domain admin, they deployed backdoors (LODEINFO or NOOPDOOR) to several machines by copying components over SMB and abusing *schetasks.exe* or *sc.exe* to achieve lateral movement. The following are the adversary's actual commands to deploy malicious components over admin shares.

- copy SfsDllSample.exe \\<IP>\c$\windows\temp\SfsDllSample.exe
- copy SfsDll32.dll \\<IP>\c$\windows\temp\SfsDll32.dll
- copy mssitlb.xml \\<IP>\C$\Windows\system32\UIAnimation.xml
- copy ShiftJIS.dat \\<IP>\C$\Windows\system32\ComputerToastIcon.contrast-white.dat

Once the intrusion progressed, Earth Kasha started to exfiltrate the stolen information. The adversary gathered data, including *ntds.dit*, SYSTEM, SAM registry hives and other interesting files on a single victim machine and compressed these files into a single archive using the *makecab* command. While we couldn’t confirm how these data would be exfiltrated, it might be over the backdoor channel. Earth Kasha also exfiltrated interesting files in the victim network over the RDP session. They copied interesting files to the RDP source host over SMB (“tsclient” is an RDP source host).

- \\tsclient\C\aaa\All PC List.xlsx
- \\tsclient\C\aaa\All IP List.xlsx

- \\tsclient\C\aaa\Network Diagram.xlsx

Malware Analysis

In the previous campaign by Earth Kasha, LODEINFO has been their primary backdoor of choice. In the new campaign, however, we have observed several backdoors, such as Cobalt Strike, LODEINFO and previously undocumented NOOPDOOR. These backdoors were selectively used for each incident.

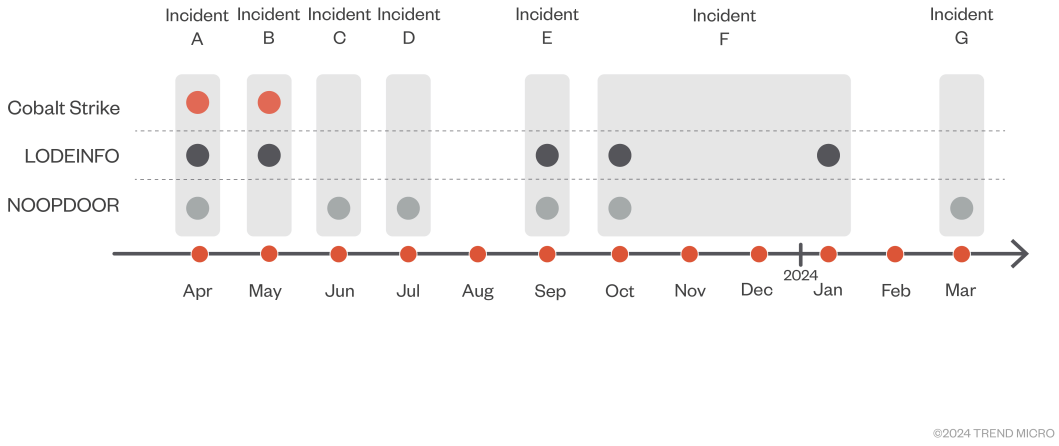


Figure 2. Observed malware in each incident

Possible Cracked Version of Cobalt Strike

In the early incidents above, Earth Kasha also used Cobalt Strike. Like other adversaries, Cobalt Strike is designed to be executed only in memory. In this case, Earth Kasha used a shellcode loader written in Go, which we dubbed GOSICLOADER. GOSICLOADER is intended to be loaded via DLL side-loading and simply decrypts the embedded payload in the data section using Based64+AES.

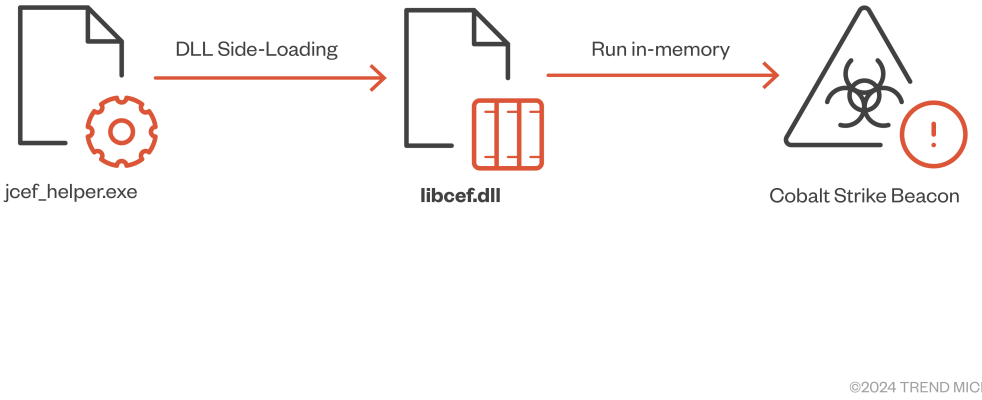


Figure 3. Execution flow of GOSICLOADER

Upon checking the configuration of the Cobalt Strike beacon, we noticed it could be a cracked version of the Cobalt Strike, known as CSAgent, shared among the Chinese-speaking hacking community. According to the developer of Cobalt Strikeopen on a new tab, Cobalt Strike beacon embeds watermark and watermark hash to make it difficult to tamper with authorization. CSAgent modifies the watermark to include "666666" by default and uses a watermark hash that matches the one embedded in the observed Cobalt Strike beacon for this campaign. Since the watermark and its hash can be easily tampered with if the adversary knows the algorithm, this modification could be a false flag, but it is still noteworthy.

```
Watermark_Hash                  - MYhXSMGVvcr7PtOTMdABvA==
Watermark                       - 666666
```

Figure 4. Watermark and watermark hash in configuration

另外，CSAgent这个版本更新了配置方式，现在将配置统一放到CSAgent.properties文件里

```
1 sleeved.decryption.key=f38eb3d1a335b252b58bc2acde81b542
2 beacon.watermark.value=666666
3 beacon.watermark.hash=MYhXSMGVvcr7PtOTMdABvA==
4 # 翻译开关，是否启用翻译功能，值：Y/N
5 need.translation=Y
6 # 是否只使用翻译功能，用于加载自己魔改过的版本，值：Y/N
7 translation.only=N
```

Figure 5. Watermark and its hash in CSAgent

LODEINFO

LODEINFO is a backdoor exclusively used by Earth Kasha since 2019, serving as their primary backdoor. In this new campaign, however, it is just one option among several, showing its adaptability. Since its introduction,

LODEINFO has gone through continuous updates, as indicated by its version numbers. In this campaign, we have observed versions v0.6.9, v0.7.1, v0.7.2, and v0.7.3

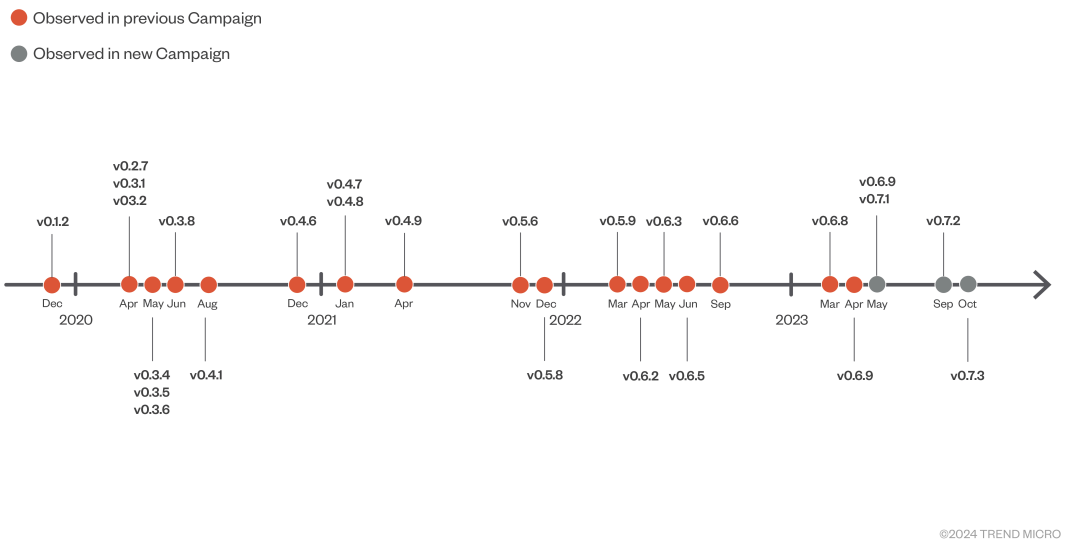


Figure 6. Version number history of LODEINFO

With the incrementing version number, Earth Kasha has also been updating a procedure to execute LODEINFO. In this new campaign, they deployed three components in the victim machine. They registered the legitimate application (SfsDllSample.exe in Figure 7) as a scheduled task, which will trigger DLL Side-Loading of malicious DLL (SfsDll32.dll in Figure 7).

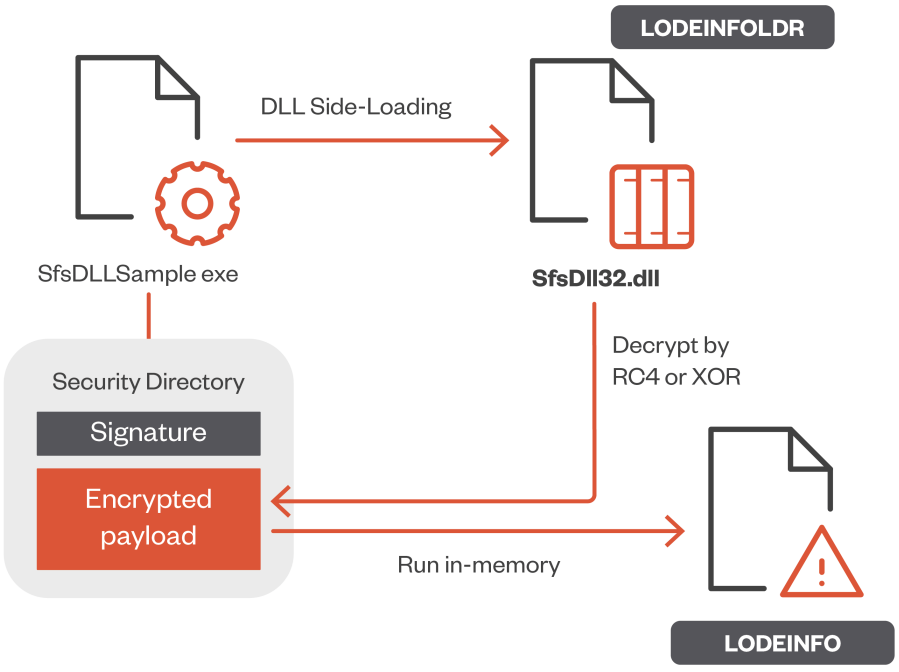


Figure 7. Execution sequence of LODEINFO

This malicious DLL, which we dubbed LODEINFOLDR (aka FaceLoader by ESET), extracts an encrypted payload embedded in the digital signature of the loaded process and decrypts it by RC4 or XOR. The encrypted payload is embedded in the legitimate digital signature by [abusing MS13-098/CVE-2013-3900open on a new tab](#).

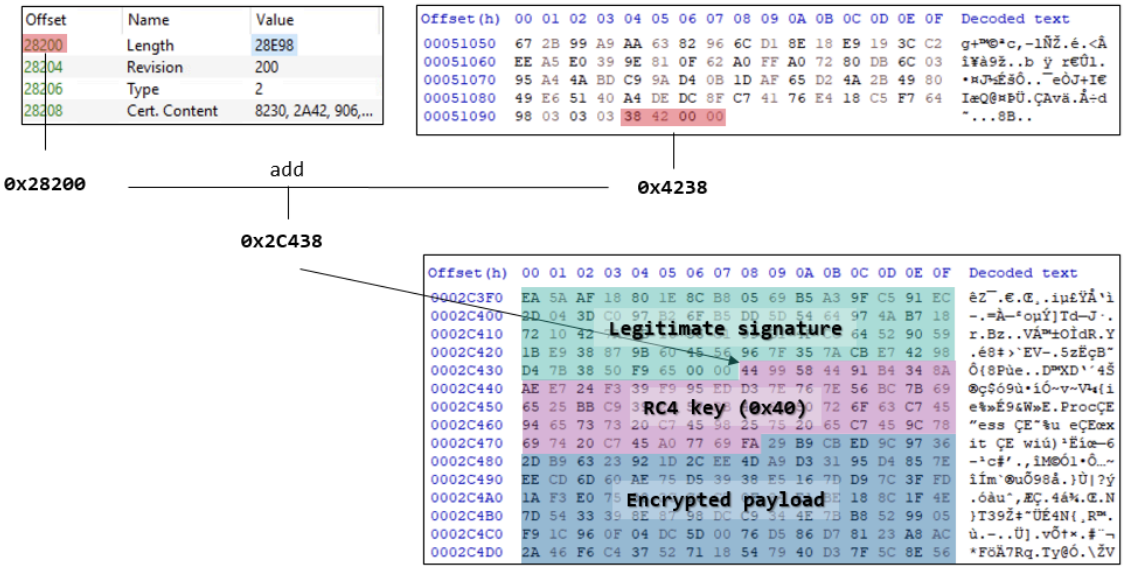


Figure 8. Embedded encrypted payload and RC4 in digital signature

We distinguish this LODEINFOLDR in the new campaign from the ones we had seen in the previous campaign, and we call this new loader LODEINFOLDR Type 2. At first glance, we thought LODEINFOLDR Type 2 was their new loader developed for the new campaign. Still, after further investigation, we identified that LODEINFOLDR Type 2 looks the same as the loader of LODEINFO used in the LiberalFace campaign in 2022, disclosed by ESET3. This may infer that the same entity has used the same malware since the previous campaign.

Regarding LODEINFO, several backdoor commands were newly supported. “pkill”, “ps”, “keylog”, and “autorun” were added in v0.6.9, and “runas” was newly added in v0.7.1. The backdoor commands supported in v0.6.9 differed from the old ones since these commands were initially added in the previous version, removed in v0.6.3 and added again in v0.6.9. On the other hand, “runas” supported in v0.7.1 is a new one that enables running the processes as a specific user. Since v0.7.2, the "config" command, which is just used to display “Not Available.”, has been fully implemented.

v0.6.9	**v0.7.1**	**v0.7.2 and v0.7.3**
command ls rm mv cp cat mkdir send recv memory kill cd ver print ransom (not implemented) comc config *pkill* *ps* *keylog* *autorun*	command ls rm mv cp cat mkdir send recv memory kill cd ver print ransom (not implemented) comc config pkill ps keylog autorun *runas*	command ls rm mv cp cat mkdir send recv memory kill cd ver print ransom (not implemented) comc config pkill ps keylog autorun runas

Table 1. Backdoor commands supported by LODEINFO, newly added commands in *italics*

All the LODEINFO we observed in the new campaign were slightly different in the backdoor command process compared to the LODEINFO in the previous campaign. This LODEINFO type supports running DLL or shellcode in memory without backdoor command processing. After further investigation, we concluded that this type of LODEINF we observed in the new campaign should be the same as the one that ESET calls “The 2nd stage

LODEINFO" observed in the LiberalFace campaign. As Figure 9 and Figure 10 show, the LODEINFO in the new campaign directly supports running DLL or shellcode in memory without processing backdoor commands. This evidence may also infer that the same group has been using the same malware since the previous campaign.

```
if ( a1 )
{
  v1 = (_DWORD *)a1[1];
  v2 = *a1;
  v3 = (_DWORD *)v1[2];
  v5 = v3;
  if ( *v3 )
  {
    init_encstr_table(v6, v2);
    if ( process_backdoor_command(v6, (int)v1) )
      *(_DWORD *)(*v1 + 40) = (*(int (**)(void))(v2 + 160))();
    sub_1DB25(v6);
    v3 = v5;
  }
  (*(void (__cdecl **)(_DWORD *))(v2 + 104))(v3);
  (*(void (__cdecl **)(_DWORD *))(v2 + 104))(v1);
  (*(void (__cdecl **)(int *))(v2 + 104))(a1);
}
```

Figure 9. C&C server response processing of the LODEINFO in the previous campaign

```
if ( v4 == 'M' && *(v15 + 5) == 'Z' )     // run DLL
{
  v13 = v1;
  v6 = sub_6615(v5, v3, v15);
  v7 = v6;
  if ( v6 )
  {
    if ( *(v6 + 20) )
    {
      strcpy(v14, "MyExpFunc");
      v8 = (sub_2485)(v6, v14);
      if ( v8 || (v8 = (sub_2485)(v7, 0)) != 0 )
        v8(0);
    }
    else
    {
      v11 = *(v6 + 36);
      if ( v11 && *(v7 + 24) )
        v11();
    }
  }
}
else
{
  if ( v4 != 0xE9 )                       // process backdoor commands
  {
    init_encstr_table(v16, v1);
    v9 = v12;
    if ( (process_backdoor_command)(v12) )
      *(*v12 + 40) = (*(v1 + 160))();
    sub_6245(v16);
    goto LABEL_11;
  }
  v13 = 0;
  if ( (*(v1 + 52))(v5, v3, 64, &v13) )
  {
    *v5 = 0xE9;
    v5(0);                                // run shellcode
  }
}
```

Figure 10. C&C server response processing of the 2nd stage LODEINFO

NOOPLDR

During our investigation, we encountered two different shellcode loaders; one is XML containing C#, and the other is DLL. These two types of shellcode loaders are completely different in the implementation perspective. However, a payload of both is a previously undocumented backdoor that we call NOOPDOOR, which we will describe later. Both loaders adopt a similar strategy to decrypt and store the encrypted payload using the machine's device ID. Based on these similarities, we categorized both as the same variant, which we dubbed NOOPLDR. We distinguish the former XML/C# one as “NOOPLDR Type 1” and DLL one as “NOOPLDR Type 2," respectively. NOOPLDR Type 1 is designed to be executed by Windows' trusted utility tool, MSBuild, as shown in Figure 11.

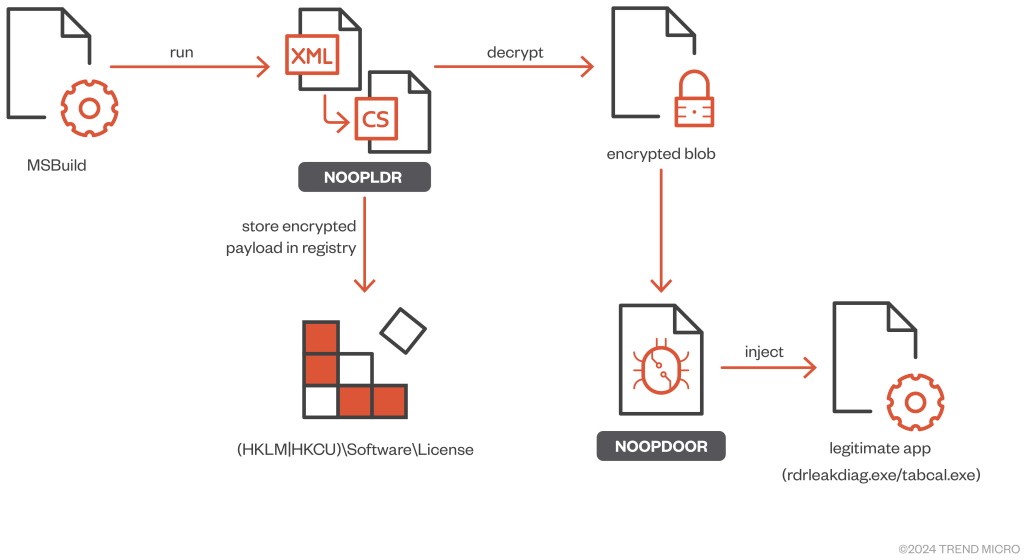


Figure 11. Execution flow of NOOPLDR Type 1 (XML)

In most cases, MSBuild and the target XML file are registered as a Scheduled Task for persistence. MSBuild compiles the inclined C# in XML project on runtime, a key component of NOOPLDR Type 1. The inclined C# code is typically concealed as follows.

```
<Code Type="Class" Language="cs"><![CDATA[using System.Text;using Microsoft.Build.Framework;using
Microsoft.Win32;using System.IO;using System;using System.Threading;using System.Security.Cryptography;
using Microsoft.Build.Utilities;using System.Runtime.InteropServices;public class
EEHXpccyiwg8tw8IMmdOagxHxjmOxZBoy8u9iDF8bsL0nNA6M9zX: Task,ITask{[DllImport("advapi32.dll",
ExactSpelling=true)]static extern bool OpenProcessToken(IntPtr GYxh9AgZybs42Ju,int _9A,ref IntPtr
QfMSFHuTSq);static void xA(string a1j61idziNrs5uef4NJpgi4g1JORxIwN2q2Skfhrgq){DateTime
_9AnMaOMvyYkH17derDvcJrNAOBv4ZGrj0l5PO7OcUn8ZRSbVhaZXUowSXX7I=File.GetCreationTime(Environment.
GetEnvironmentVariable("windir")+"\\system32\\kernel32.dll");File.SetLastWriteTime
(a1j61idziNrs5uef4NJpgi4g1JORxIwN2q2Skfhrgq,
_9AnMaOMvyYkH17derDvcJrNAOBv4ZGrj0l5PO7OcUn8ZRSbVhaZXUowSXX7I);File.SetCreationTime
(a1j61idziNrs5uef4NJpgi4g1JORxIwN2q2Skfhrgq,
_9AnMaOMvyYkH17derDvcJrNAOBv4ZGrj0l5PO7OcUn8ZRSbVhaZXUowSXX7I);File.SetLastAccessTime
(a1j61idziNrs5uef4NJpgi4g1JORxIwN2q2Skfhrgq,
_9AnMaOMvyYkH17derDvcJrNAOBv4ZGrj0l5PO7OcUn8ZRSbVhaZXUowSXX7I);}static byte[]RG1GbRPcyzuZtswsXS6A(byte[]
XYQDyMYm11agTcQgsQbvTTLF8B6az9eahT4Sl,byte[]Mf0,byte[]
GItSKpQlEc5f0enlF7PYX2X99ai75qe5E4nclq1QtoEcfb20F9HO){try{using(AesCryptoServiceProvider
cbOVuTKhSqo89L60k9pg9FJxU37=new AesCryptoServiceProvider()){cbOVuTKhSqo89L60k9pg9FJxU37.KeySize=256;
cbOVuTKhSqo89L60k9pg9FJxU37.BlockSize=128;cbOVuTKhSqo89L60k9pg9FJxU37.Key=Mf0;
```

Figure 12. Example of NOOPLDR

NOOPLDR Type 1 changes its behavior depending on whether it's the first-time execution or otherwise. If it's the first execution, NOOPLDR Type 1 tries to find encrypted data from a hardcoded file path, which differs for each NOOPLDR sample. If it exists, NOOPLDR Type 1 deletes the file after reading the content. The encrypted data consists of a header for checksum, AES key materials and an encrypted body. NOOPLDR Type 1 reads the first 32 bytes, computes the SHA256 hash of the following encrypted body, and then compares the hash with the header to verify if the data is an expected structure. After completing verification, NOOPLDR Type 1 calculates the SHA384 hash of the AES key material following behind the checksum header. The first 32 bytes are used as the AES key, and the later 16 as IV. Finally, NOOPLDR Type 1 decrypts the encrypted payload by AES256-CBC.

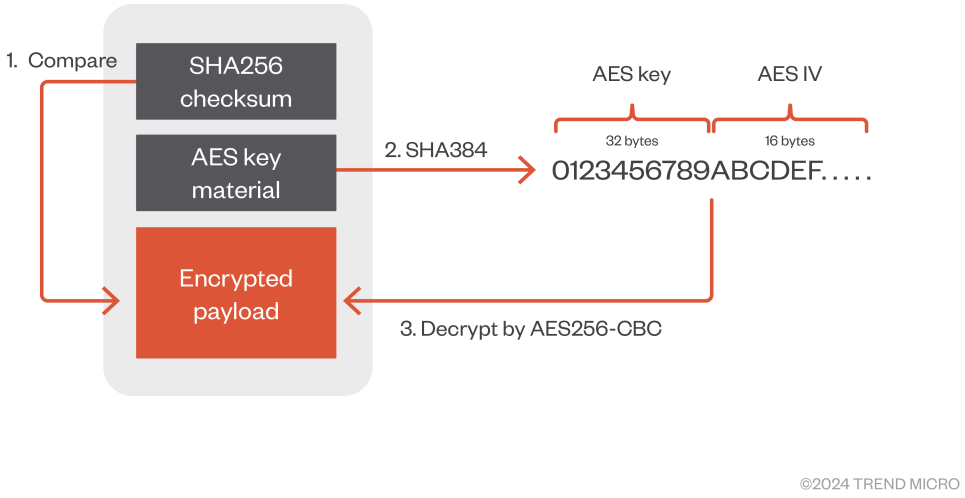


Figure 13. Structure of the encrypted data of NOOPLDR Type 1

The decrypted data has a header containing a 64-bit flag, the payload size, an offset to the payload and the payload data in the following structure.

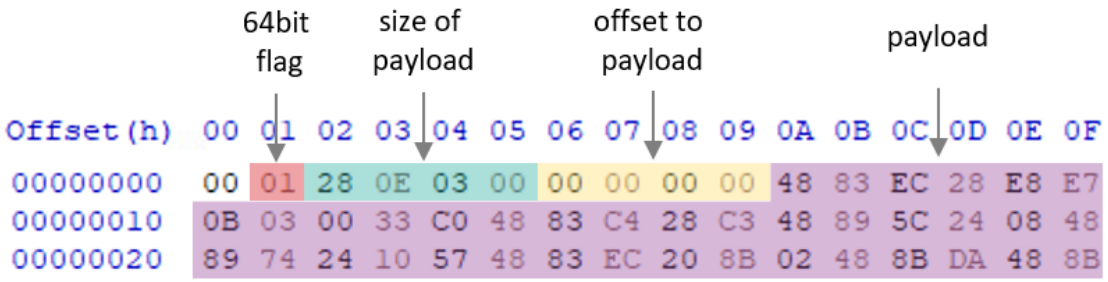


Figure 14. Structure of the decrypted data of NOOPLDR Type 1

Once the decryption succeeds, NOOPLDR Type 1 tries to store the payload in the registry for stealthy persistence. The encryption algorithm is still AES256-CBC, but the AES key and IV are generated based on a machine's Device ID and a hostname. The device ID is retrieved from the registry key "HKLM\Software\Microsoft\SQMClient\MachineId," which contains the machine's unique GUID. NOOPLDR Type 1 calculates the SHA384 hash of the concatenated Device ID and hostname and follows the same procedure in the decryption routine, splitting the hash value into chunks of 32 bytes and 16 bytes for AES key and IV respectively.

NOOPLDR Type 1 then prepends the SHA256 hash of the encrypted payload and stores it in the registry " (HKLM|HKCU)\Software\License\{HEX}", which "HEX" is a hex string of the last 16 bytes of the SHA256 hash of the hostname. Since this encryption procedure uses a unique value for each infected machine, we need to preserve additional info and data, such as registry hive and hostname, to smoothly decrypt the payload. If NOOPLDR Type 1 successfully stores the payload in the registry, it deletes the encrypted file on a disk. Therefore, in the second and subsequent execution time, NOOPLDR Type 1 reads the registry key and decrypts the payload in the same procedure as the encryption routine.

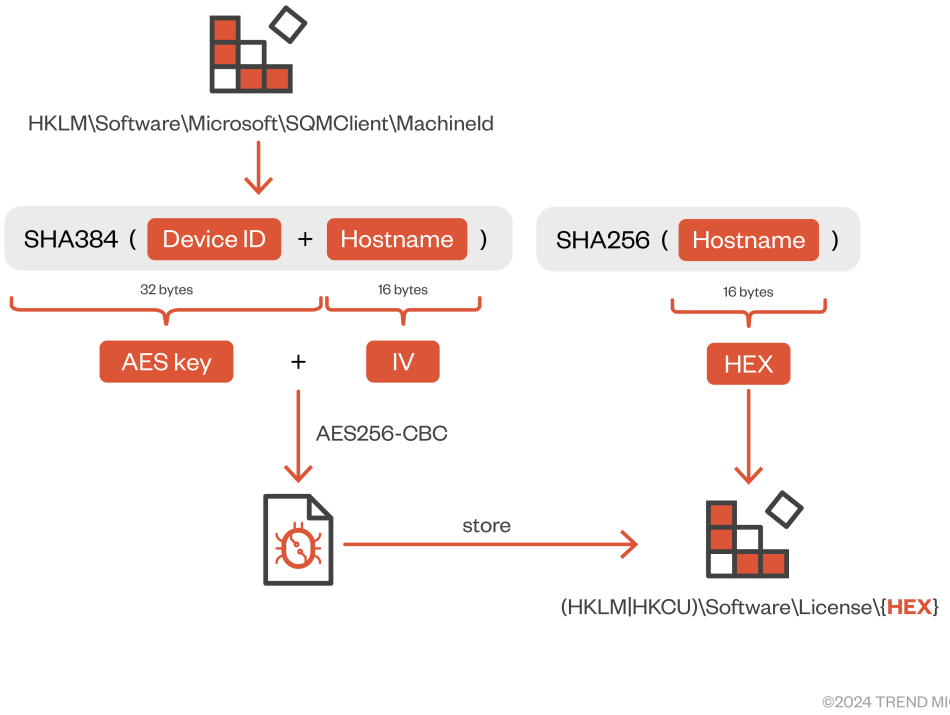


Figure 15. Procedure to store an encrypted payload in the registry by NOOPLDR Type 1

In the final step, NOOPLDR Type 1 injects and runs the decrypted payload into a legitimate application, such as rdrleakdiag.exe and tabcal.exe. If NOOPLDR Type 1 fails to store the payload in the registry, it writes the encrypted payload into a disk again and overwrites it with the same timestamp as the built-in kernel32.dll.

Another type of NOOPLDR in the form of a DLL, which we call NOOPLDR Type 2, adopts a similar strategy to Type 1 but implements more stealthy techniques. As Figure 16 illustrates, during the first execution, NOOPLDR Type 2 also decrypts the encrypted payload from a file and stores the encrypted payload in the registry. It injects the decrypted payload into the legitimate application.

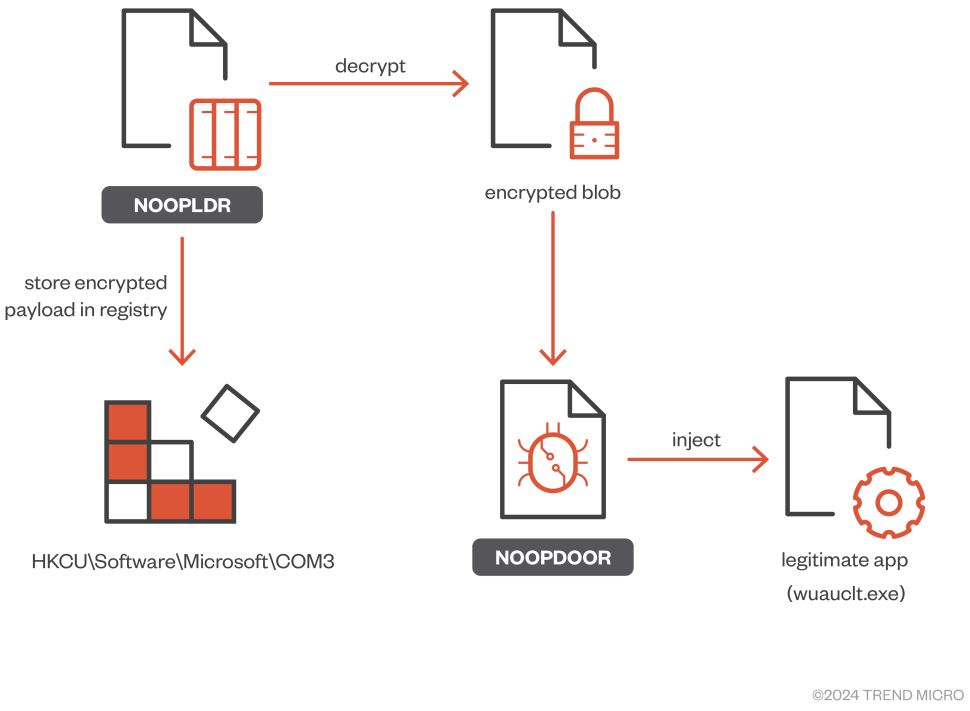
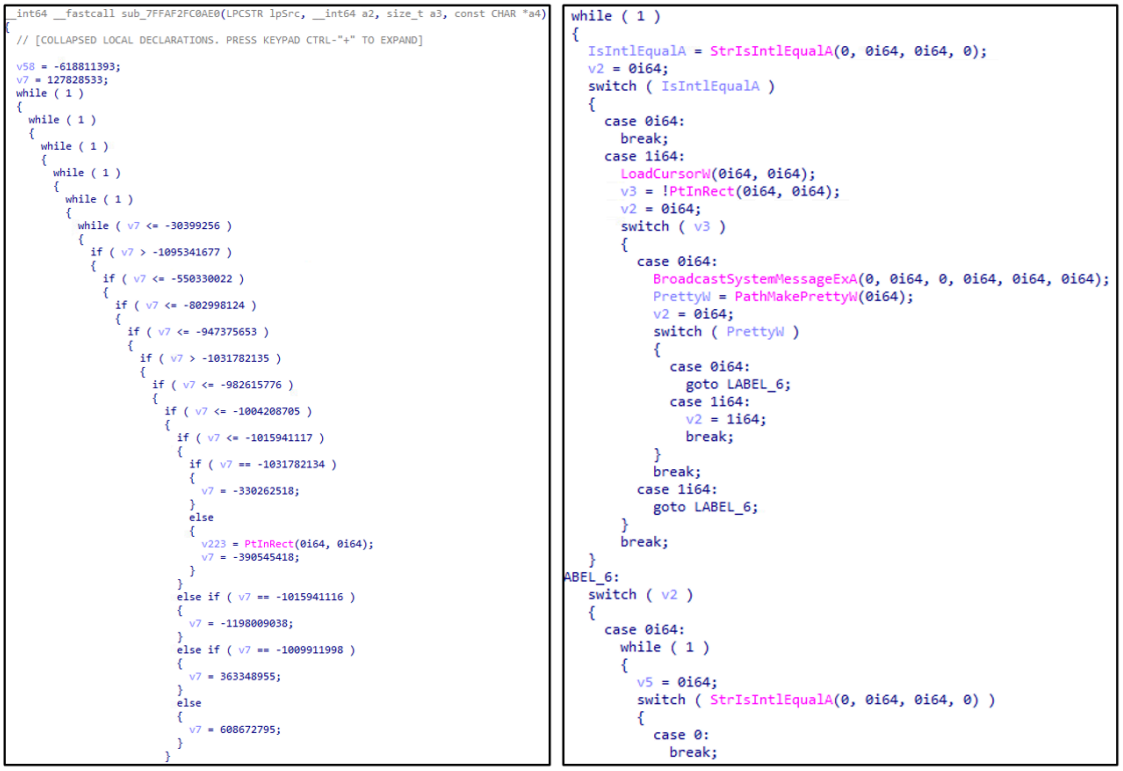


Figure 16. Execution flow of NOOPLDR Type 2 (DLL)

One of the notable features of NOOPLDR Type 2 is the use of multiple anti-analysis techniques. For instance, it is heavily obfuscated by control flow obfuscation and junk codes, as shown in Figure 17. Earth Kasha has already applied this type of obfuscation technique in the previous campaignopen on a new tab, but even before that, it's been popular among China-nexus adversaries, such as APT10open on a new tab and Twisted Pandaopen on a new tab.



```
__int64 __fastcall sub_7FFAF2FC0AE0(LPCSTR lpSrc, __int64 a2, size_t a3, const CHAR *a4)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v58 = -618811393;
  v7 = 127828533;
  while ( 1 )
  {
    while ( 1 )
    {
      while ( 1 )
      {
        while ( 1 )
        {
          while ( 1 )
          {
            while ( v7 <= -30399256 )
            {
              if ( v7 > -1095341677 )
              {
                if ( v7 <= -550330022 )
                {
                  if ( v7 <= -802998124 )
                  {
                    if ( v7 <= -947375653 )
                    {
                      if ( v7 > -1031782135 )
                      {
                        if ( v7 <= -982615776 )
                        {
                          if ( v7 <= -1004208705 )
                          {
                            if ( v7 <= -1015941117 )
                            {
                              if ( v7 == -1031782134 )
                              {
                                v7 = -330262518;
                              }
                              else
                              {
                                v223 = PtInRect(0i64, 0i64);
                                v7 = -390545418;
                              }
                            }
                            else if ( v7 == -1015941116 )
                            {
                              v7 = -1198009038;
                            }
                            else if ( v7 == -1009911998 )
                            {
                              v7 = 363348955;
                            }
                            else
                            {
                              v7 = 608672795;
                            }
                          }
```

```
while ( 1 )
{
  IsIntlEqualA = StrIsIntlEqualA(0, 0i64, 0i64, 0);
  v2 = 0i64;
  switch ( IsIntlEqualA )
  {
    case 0i64:
      break;
    case 1i64:
      LoadCursorW(0i64, 0i64);
      v3 = !PtInRect(0i64, 0i64);
      v2 = 0i64;
      switch ( v3 )
      {
        case 0i64:
          BroadcastSystemMessageExA(0, 0i64, 0, 0i64, 0i64, 0i64);
          PrettyW = PathMakePrettyW(0i64);
          v2 = 0i64;
          switch ( PrettyW )
          {
            case 0i64:
              goto LABEL_6;
            case 1i64:
              v2 = 1i64;
              break;
          }
          break;
        case 1i64:
          goto LABEL_6;
      }
      break;
  }
LABEL_6:
  switch ( v2 )
  {
    case 0i64:
      while ( 1 )
      {
        v5 = 0i64;
        switch ( StrIsIntlEqualA(0, 0i64, 0i64, 0) )
        {
          case 0:
            break;
```

Figure 17. Control Flow Obfuscation (Left) and Junk Code (Right)

For the additional anti-analysis technique, most strings are simply encoded by XOR, which is decoded on runtime.

```
v33 = 0i64;
while ( 1 )
{
  decrypt_string(v47, byte_1009F580[v33] ^ byte_1009F5A0[v33]);// SOFTWARE\\Microsoft\\COM3
  v33 = v33 + 1;
  switch ( v33 < 0x17 )
```

Figure 18. String decoding routine by XOR

NOOPLDR Type 2 is designed to be executed via DLL Side-Loading. NOOPLDR Type 2 supports self-installation as Windows Service by running with the "-install" parameter. During the first execution, it loads an encrypted payload named “<LOADER_PROCESS_NAME>_config” in the current working directory, which will be deleted after installation. For instance, if the loader process name is “symstore.exe," the encrypted file would be "symstore.exe_config." The encrypted blob structure is like the Type 1 but slightly different. It doesn’t have a checksum section; it simply has 32-byte AES key materials followed by an encrypted payload, as Figure 19 shows. The encrypted payload is encrypted by AES256-CBC. The AES key is generated based on the SHA1 of the first 32 bytes, and IV is the first 16 bytes.

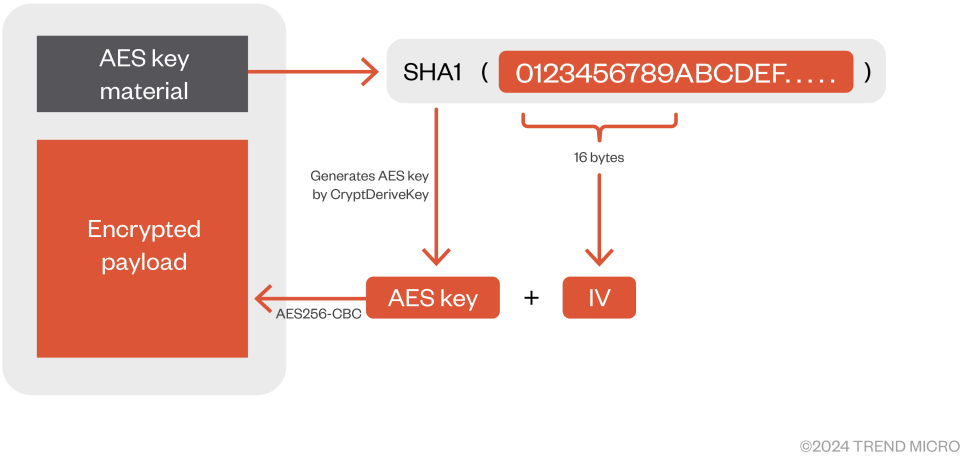


Figure 19. Structure of the encrypted data of NOOPLDR Type 2

Like the NOOPLDR Type 1, the decrypted data has a 0x14 bytes header containing several values used to verify if it's an expected structure, as Figure 20 shows.

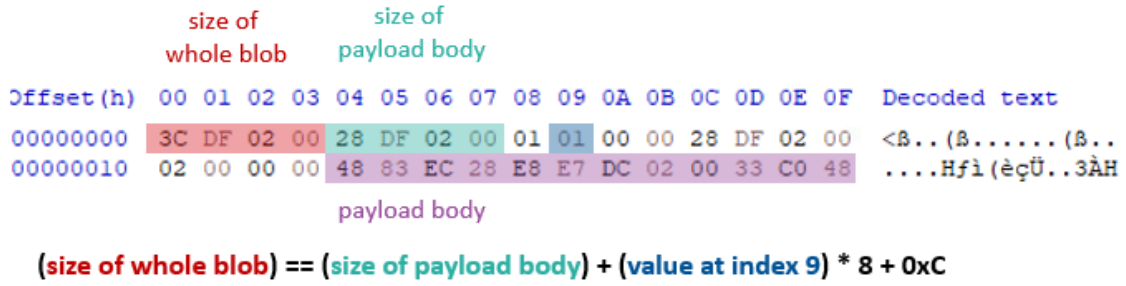


Figure 20. Structure of the decrypted data of NOOPLDR Type 2

After verification, NOOPLDR Type 2 encrypts the decrypted data again with AES256-CBC but with a different key, which consists of a Device ID string, hardcoded key material in the code section and randomly generated 8-byte hex string and stores it in "HKCU\SOFTWARE\Microsoft\COM3\<RANDOM_HEX_STRTING>," as Figure 21 shows.

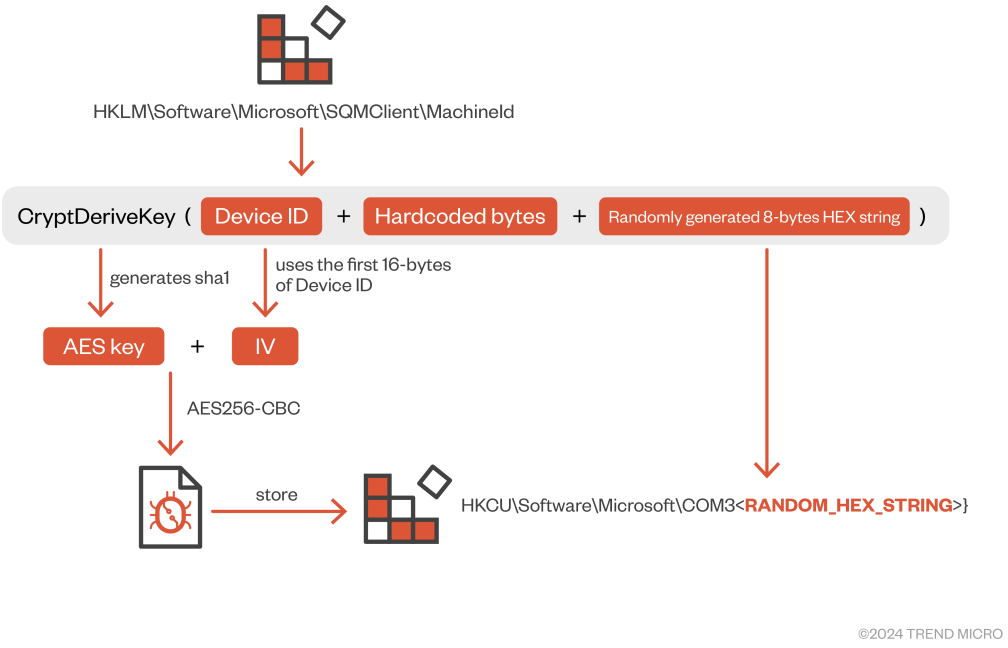


Figure 21. Procedure to store an encrypted payload in the registry by NOOPLDR Type 2

In the second and subsequent execution time, NOOPLDR Type 2 will be executed without the "-install" parameter. Therefore, it skips self-installation and proceeds to the payload decryption routine from the registry. It searches registry data in the registry (HKCU\SOFTWARE\Microsoft\COM3), and if found, it decrypts the encrypted data by the same method in Figure 21 but using the HEX string in the registry key as a part of AES key material.

At last, NOOPLDR Type 2 injects the decrypted payload into legitimate applications, such as wuauclt.exe. This process injection technique is classic, but leverages direct Syscall using NtProtectVirtualMemory, NtWriteVirtualMemory and NtCreateThreadEx. Since Syscall ID can be different on running OS versions, Syscall ID is calculated on runtime.

```
00007FFC790C9D10  48:894C24 08     mov qword ptr ss:[rsp+8],rcx
00007FFC790C9D15  48:895424 10     mov qword ptr ss:[rsp+10],rdx
00007FFC790C9D1A  4C:894424 18     mov qword ptr ss:[rsp+18],r8
00007FFC790C9D1F  4C:894C24 20     mov qword ptr ss:[rsp+20],r9
00007FFC790C9D24  48:83EC 28       sub rsp,28
00007FFC790C9D28  B9 832B5041      mov ecx,41502B83
00007FFC790C9D2D  E8 1C120200      call symsrv.7FFC790EAF4E
00007FFC790C9D32  48:83C4 28       add rsp,28
00007FFC790C9D36  48:8B4C24 08     mov rcx,qword ptr ss:[rsp+8]
00007FFC790C9D3B  48:8B5424 10     mov rdx,qword ptr ss:[rsp+10]
00007FFC790C9D40  4C:8B4424 18     mov r8,qword ptr ss:[rsp+18]
00007FFC790C9D45  4C:8B4C24 20     mov r9,qword ptr ss:[rsp+20]
00007FFC790C9D4A  4C:8BD1          mov r10,rcx
00007FFC790C9D4D  0F05             syscall                          NtWriteVirtualMemory
00007FFC790C9D4F  C3               ret
```

Figure 22. Example of usage of NtWriteVirtualMemory

NOOPDOOR

Now, let’s step into the final payload, NOOPDOOR. NOOPDOOR (aka HiddenFace by ESET) is a sophisticated and complex backdoor with the following characteristics:

- Fully position independent code

- Supporting active and passive mode communication
- C&C domain changed daily by a DGA (by default)
- Proxy-aware TCP communication during working time
- RSA + multiple symmetric cipher to encrypt the entire C&C communication
- Supporting build-in functions + additional modules for backdoor capabilities
- Evading in-memory detection by encrypting/decrypting specific functions on runtime
- Anti-analysis

Due to its complexity, NOOPDOOR should be designed as another backdoor choice, especially for a high-profile target. Based on our records, NOOPDOOR was first observed as a second-stage payload of LODEINFO in 2021, but only in limited cases. And we have not encountered NOOPDOOR until 2023. One of the interesting features of NOOPDOOR is that it supports two channels to communicate with the C&C server, which we call the active and passive modes.

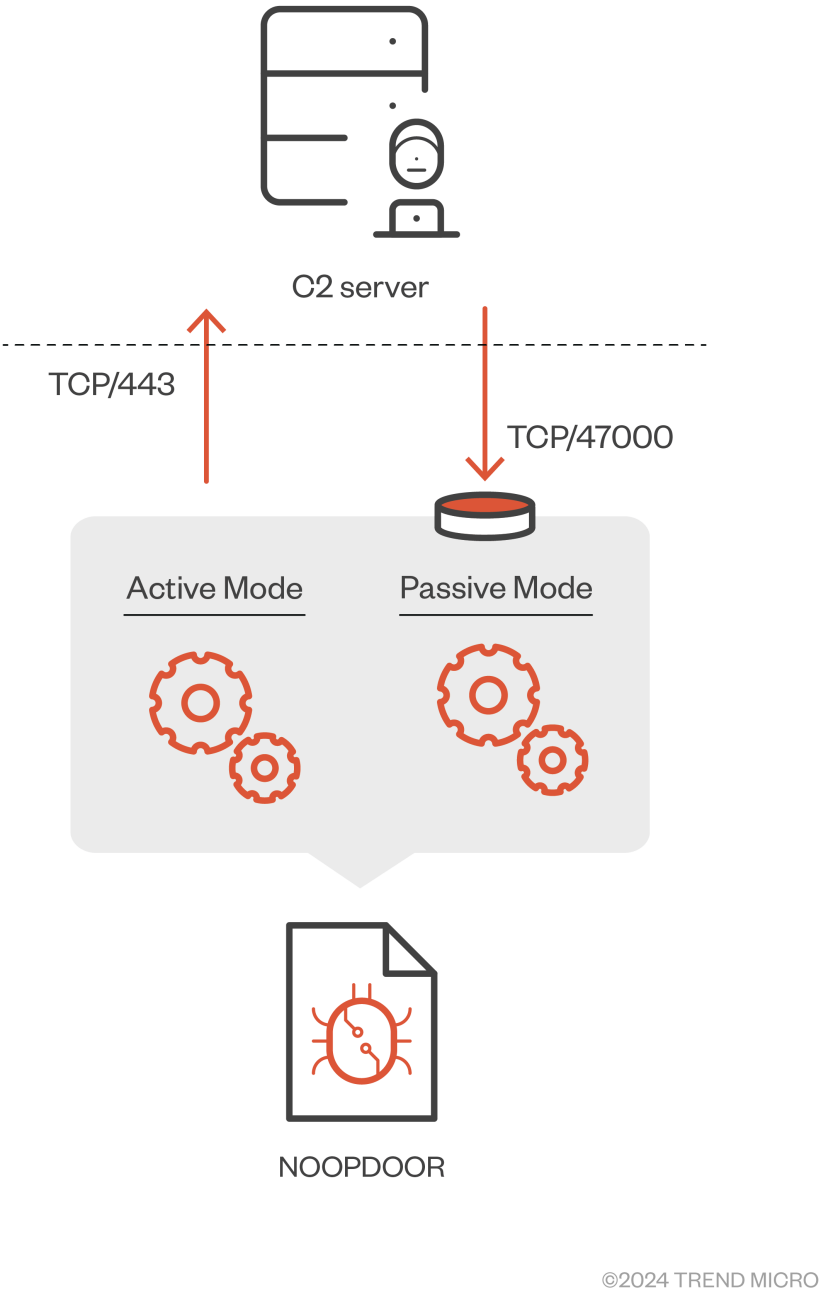


Figure 23. Overview architecture of NOOPDOOR

Figure 23 shows that NOOPDOOR in active mode communicates over TCP/443 by polling the C&C server. NOOPDOOR in passive mode listens on TCP/47000 to receive commands from remote adversaries. Interestingly, the active and passive modes use different encryption algorithms and backdoor commands, respectively, which means that both channels are incompatible and independent methods of communication from each other. The active mode is executed in a primary thread of NOOPDOOR. Before starting communications with the C&C server, NOOPDOOR checks if the specific analysis tools listed in Appendix A are running in the current machine. If any are found, NOOPDOOR will terminate itself. NOOPDOOR then generates the C&C server's domain using a custom Domain Generation Algorithm (DGA). NOOPDOOR has template URLs like

“http://$j[].srmbr\.com/#180” (defanged) that are used to generate the domain, and NOOPDOOR embeds a randomly generated string based on the runtime date into the template URLs. Therefore, a domain can be changed daily (by default, but the lifespan of domains can be changed based on the option). A detailed DGA logic is as follows.

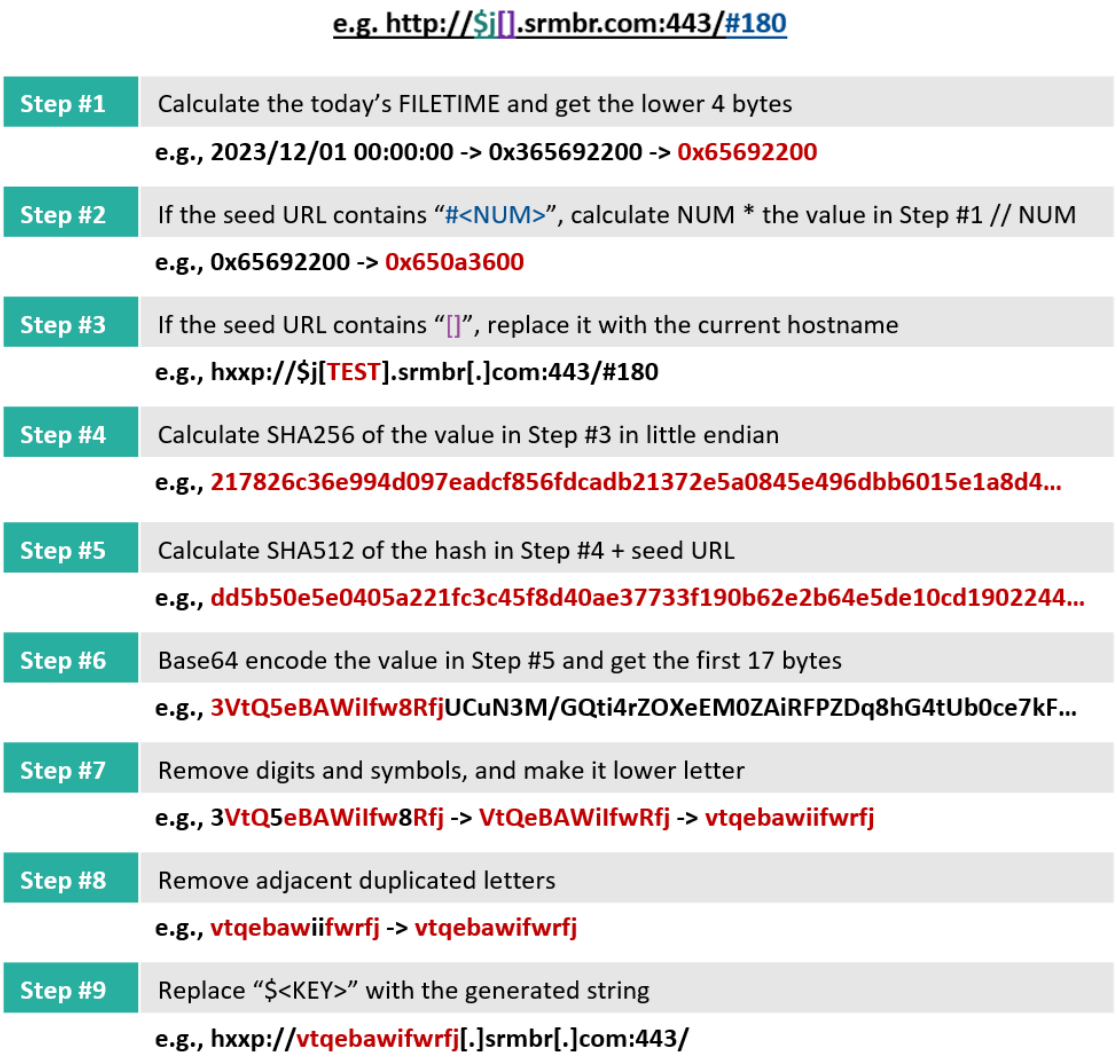


Figure 24. Detailed logic of DGA

We have also observed a few samples of NOOPDOOR that embed slightly different types of URLs. The placeholder “$<KEY>," which is a single letter (such as “j”) in most cases, can be a "word." In the case we observed, the template URL was like “hxxp://$earth[.]hopto[.]org:443/”, in which the "$earth" part is the placeholder. In such a case, the generated domain will be as follows:

Figure 25. DGA generation using “word” as the placeholder

With the generated domain, NOOPDOOR initiates C&C communication. NOOPDOOR supports HTTP proxy in the victim’s environment during business hours (8:30~19:30 from Monday to Friday). C&C communication in the active mode is fully encrypted by a combination of RSA-2048 and symmetric cipher. On initializing a session, NOOPDOOR sends a challenge and randomly selected symmetric cipher ID to the C&C server with encryption by RSA-2048 to negotiate a key for encrypting packets during the following module/command processing. Supported ciphers are DES, 3DES, 2-key 3DES, AES-128-CBC, AES-192-CBC, AES-256-CBC, RC2, and RC4. After key negotiation, it starts to receive commands and sends a result with encryption by the selected cipher.

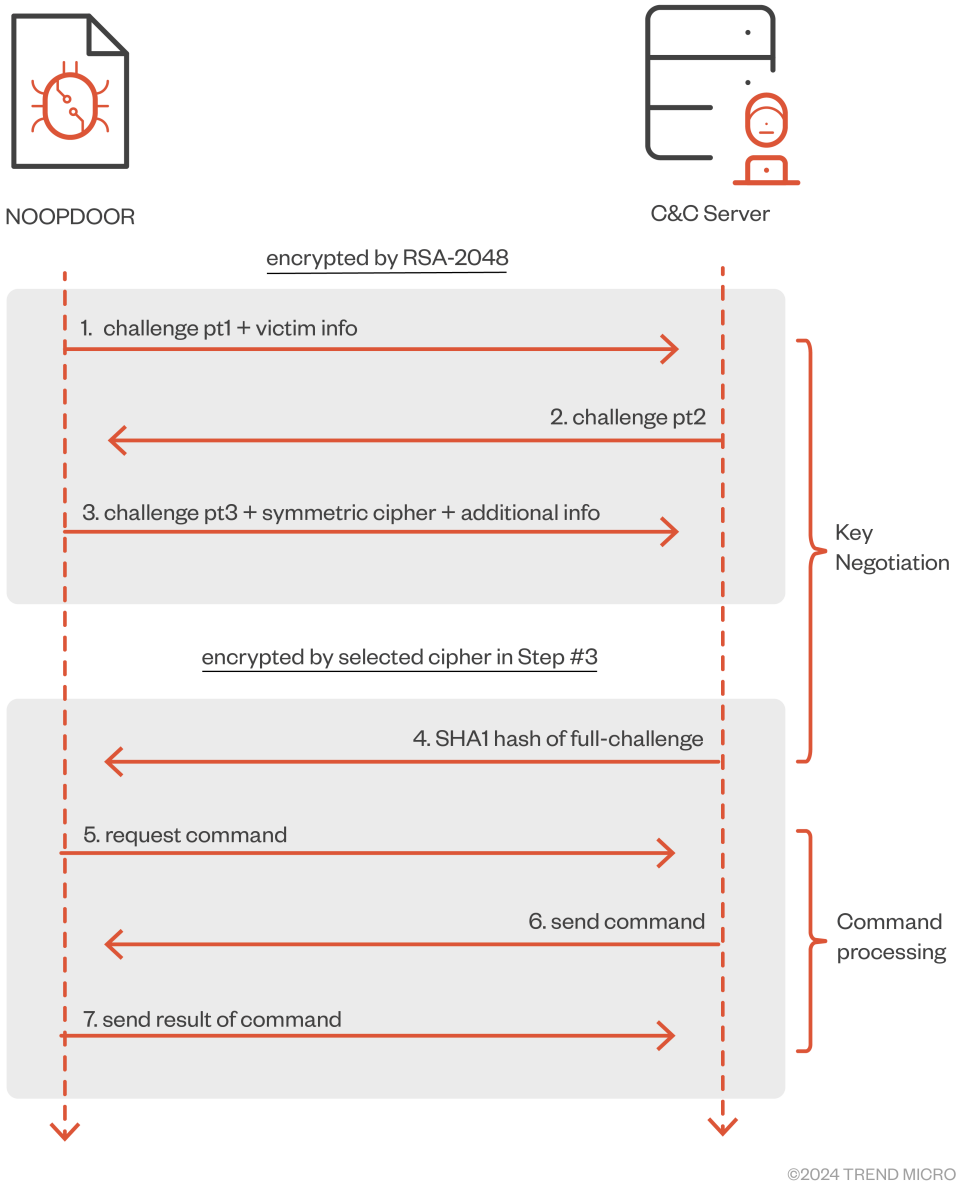


Figure 26. C&C communication flow of NOOPDOOR

The NOOPDOOR operator can execute a loaded module or built-in function through backdoor commands in active mode. The built-in functions that are currently supported are as follows:

ID (active mode)	Action
3B27D4EEFBC6137C23BD612DC7C4A817	Run program
9AA5BB92E9D1CD212EFB0A5E9149B7E5	Download a file (received from the C&C server)
3C7660B04EE979FDC29CD7BBFDD05F23	Upload a file (sending to the C&C server)
12E2FC6C22B38788D8C1CC2768BD2C76	Read specific file (%SystemRoot%\System32\msra.tlb)

2D3D5C19A771A3606019C8ED1CD47FB5	Change the timestamp of the specified file

On the other hand, C&C communication in passive mode is much simpler. NOOPDOOR creates a new thread for passive mode communication and prepares an incoming connection. NOOPDOOR initially tries to add a new Windows Firewall rule named “Cortana” to allow inbound connection to TCP/47000. C&C communication in passive mode is encrypted by AES-128-CBC with key and IV generated based on the current running datetime. Backdoor commands are also different from the ones in active mode as follows.

ID (passive mode)	Action
3049 (0x0BE9)	Keep alive
9049 (0x2359)	Run program
9050 (0x235A)	Upload a file (sending to the C&C server)
9051 (0x235B)	Download a file (received from the C&C server)
9052 (0x235C)	Change working directory
9053 (0x235D)	Run shellcode
else	Returns a message “This function is not supported by server!”

However, it should be noted that the passive mode may be useless in most cases since the operator can’t directly access the listening instance of NOOPDOOR due to a firewall or other network devices in a modern network. The passive mode might be designed for NOOPDOOR being placed in a publicly exposed server (although all the NOOPDOOR have been observed only in a local network so far) or just for testing purposes. In fact, we have observed a few samples of NOOPDOOR that do not implement the passive mode.

As another feature of NOOPDOOR, it supports loading modules from a disk. During initialization, NOOPDOOR looks for a file like "%temp%\{HEX}.tmp," in which the "{HEX}" part is generated from a portion of the SHA256 hash of a combination of the current computer name and username (in UTF-16le). This file contains the modules encrypted by AES-256-CBC. Module blobs consist of metadata, such as information for scheduling, module ID, parameters, and module payload. Due to this feature, NOOPDOOR allows them to execute additional functions at various times (on demand or regularly).

MirrorStealer

MirrorStealer, originally documented by ESET3, is a multi-purpose credential stealer. It is often used in conjunction with NOOPDOOR in cyberattacks. We have observed MirrorStealer in the recent campaign as well. Currently targeted applications are the following.

- Stored credentials in browsers (Chrome, Firefox, Edge, InternetExplorer)
- Stored credentials in email clients (Outlook, Thunderbird, Becky, Live Mail)
- Stored credentials in Group Policy Preferences
- Recently accessed server and stored credentials

- in SQL Server Management Studio (mru.dat, SqlStudio.bin)

All the results of stolen credentials are stored in %temp%\31558.TXT as plain text. We observed that the adversary manually checked the outputs using the "touch" command and deleted them with the "del" command via cmd.exe.

Attribution

As mentioned earlier, we assess the spear-phishing campaign from 2023 to early 2024 to be attributed to Earth Kasha with medium confidence. To explain the reasoning behind our conclusion, we will analyze several campaigns.

LODEINFO Campaign #1 and #2

The following image illustrates the Diamond Model of two campaigns by Earth Kasha. For convenience, we call the campaign being conducted in 2019 to 2023 using spear-phishing as "LODEINFO Campaign #1" and the campaign being conducted since 2023 targeting public-facing applications as "LODEINFO Campaign #2". The Diamond Model highlights the overlaps between the LODEINFO Campaign #1 and #2, leading us to speculate that these campaigns are operated by the same group because exclusive malware was used in both campaigns. There are no major contradictions in victimology and some parts of TTP.

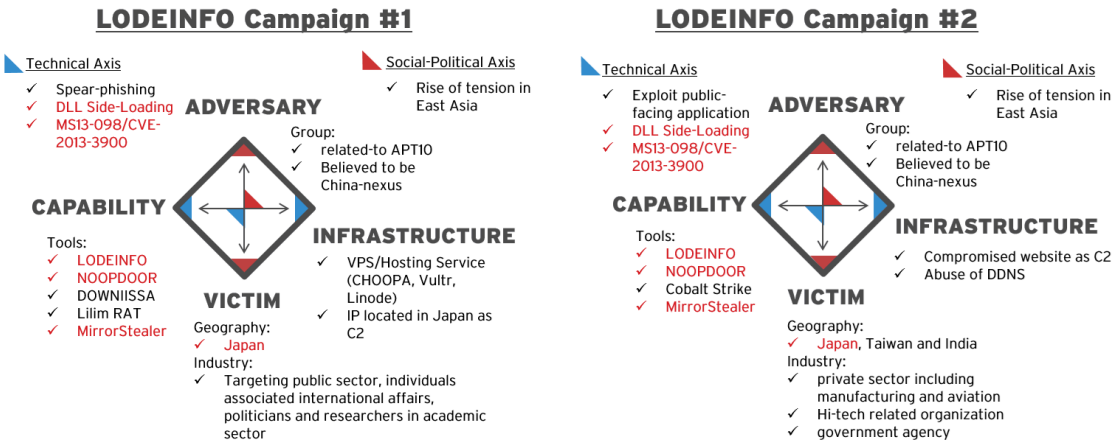


Figure 27. Comparison between the LODEINFO Campaign #1 and #2 by using the Diamond Model

On the other hand, there are several differences between the LODEINFO Campaign #1 and #2, especially in Initial Access methods, which are completely updated. In Campaign #1, they were using spear-phishing for Initial Accessopen on a new tab, but in Campaign #2, they were exploiting public-facing applications for Initial Access. Regarding victimology, there are some differences in the targeted industry. The public sector, individuals associated with international affairs, politicians, and researchers in the academic sector were targeted in Campaign #1. However, the private sector, including manufacturing and aviation, hi-tech-related organizations, and government agencies, were targeted in Campaign #2.

A41APT Campaign and LODEINFO Campaign #2

We analyzed another campaign, known as "A41APT Campaignopen on a new tab" by Earth Tengshe, which is also believed to be related to APT10. This group conducted a campaign targeting several countries, including Japan and Taiwan. The following image uses the Diamond Model to highlight the overlaps between the A41APT Campaign and the LODEINFO Campaign #2.

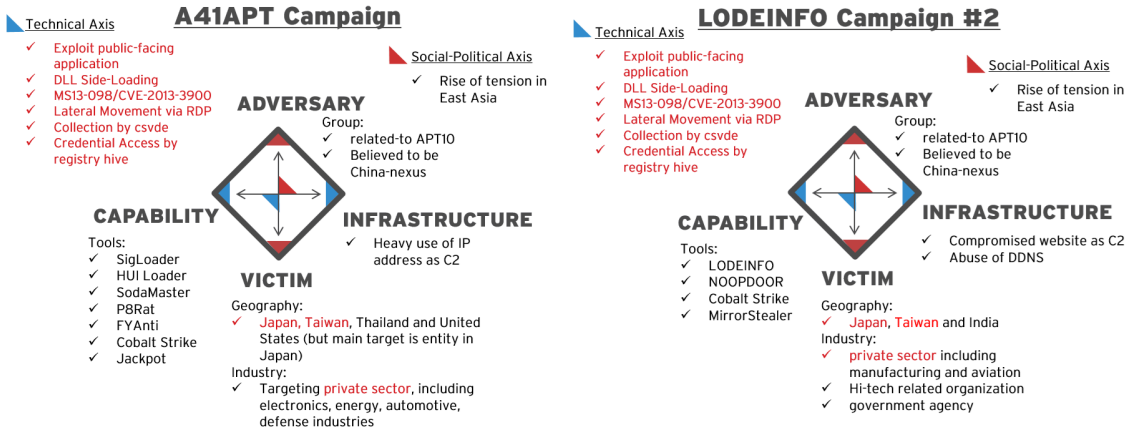


Figure 28 Comparison between the A41APT Campaign and the LODEINFO Campaign #1 by using the Diamond Model

Interestingly, the A41APT Campaign has a lot of overlaps, especially in TTPs of the Post-Exploitation phase. As the presentation on the A41APT Campaign in JSAC2021 shows, there are similar TTPs in both campaigns, such as exploiting SSL-VPN for Initial Access, schedule task abuse for Persistence, RDP by domain admin account for Lateral Movement, abusing csvde.exe to collect Active Directory account information, and dumping registry hives for Credential Access.

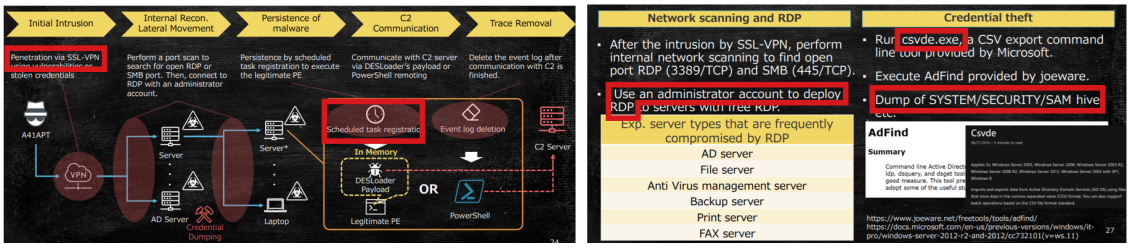


Figure 29. Highlighting the overlapped TTPs from the presentation "A41APT Case" in JSAC2021 10

The major difference in these campaigns is the toolsets. Earth Tengshe used custom malware, such as SigLoader, SodaMaster, P8RAT, FYAnti, and Jackpot, which completely differ from Earth Kasha's use in LODEINFO Campaign #2.

Considering that Earth Tengshe and Earth Kasha are believed to be associated with APT10, both groups may have relationships in TTPs or may share operator resources. Here is a summary of the comparison between the A41APT Campaign, the LODEINFO Campaign #1 and #2.

	A41APT Campaign	**LODEINFO Campaign #1**	**LODEINFO Campaign #2**

Attribution	Earth Tengshe	Earth Kasha	Earth Kasha
Timeline	2020 - 2021	2019 – present	2023 – present
Region	Japan, Taiwan, Thailand, and the United States (but the main target is the entity in Japan)	Japan	Japan, Taiwan, and India
Industry	private sector, including electronics, energy, automotive, and defense industries	public sector, individuals associated with international affairs, politicians and researchers in the academic sector	- private sector, including manufacturing and aviation - Hi-tech related organizations - government agencies
TTPs	- Exploit public-facing application - DLL Side-Loading - MS13-098/CVE-2013-3900 to embed encrypted payload	- Spear-phishing email - DLL Side-Loading - MS13-098/CVE-2013-3900 to embed encrypted payload	- Exploit public-facing application - DLL Side-Loading - MS13-098/CVE-2013-3900 to embed encrypted payload
Tools	- SigLoader - HUI Loader - SodaMaster - P8Rat - FYAnti - Cobalt Strike - Jackpot	- LODEINFO - NOOPDOOR - DOWNIISSA - Lilim RAT - MirrorStealer	- LODEINFO - NOOPDOOR - Cobalt Strike - MirrorStealer

Other Campaigns

Adding to these campaigns, we have observed a few other campaigns that slightly show some overlaps with the LODEINFO Campaign #2.

Our first observation in 2023 shows that the Initial Access and Target methods resemble those of the LODEINFO Campaign #2. This unclustered campaign targeted mainly Japan and abused an exploitation against public-facing applications for Initial Access. Additionally, we confirmed that both campaigns used the same IPs as the origin of exploitation. On the other hand, we didn’t observe any malware or hacking tools during this unclustered campaign. The adversary employed LOLBins in Post-Exploitation, not malware.

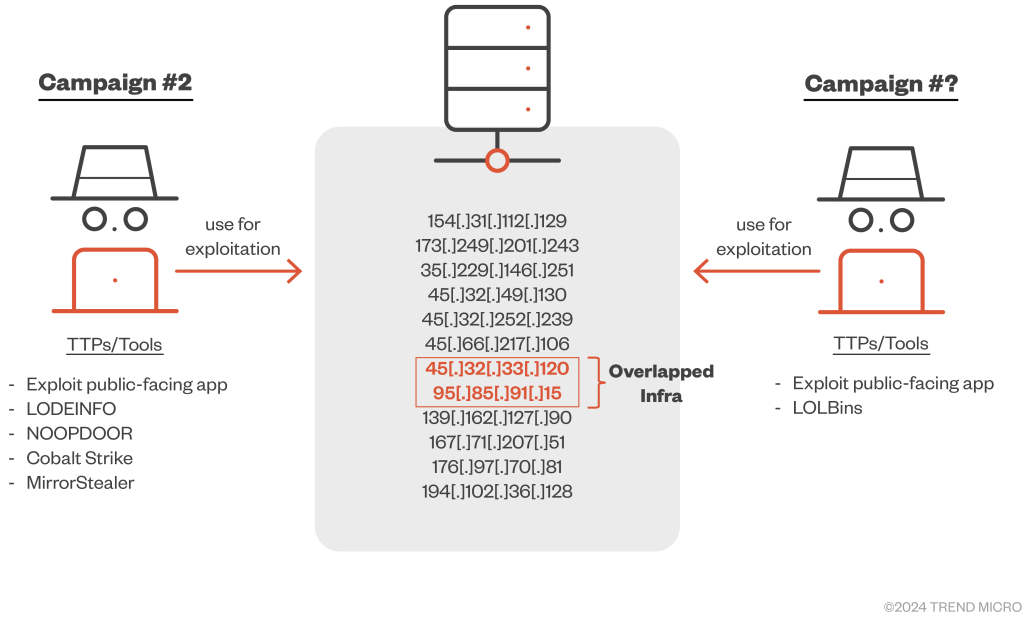


Figure 30. Infrastructure overlap with the unclustered campaign

Furthermore, Volt Typhoon, which is a state-sponsored actor based in China documented by Microsoftopen on a new tab, was reportedly carrying out the exploit against FortiOS/FortiProxy (CVE-2023-27997), which was also used in the LODEINFO Campaign #2 in 2023. However, TTPs and toolsets in Post-Exploitation were totally different between Volt Typhoon and Earth Kasha (instead, the previously mentioned unclustered campaign looks similar, but no commonalities have been confirmed so far). The vulnerability of CVE-2023-27997 was 0-day at the time of usage in both campaigns by Volt Typhoon and Earth Kasha, leading us to the assumption that the 0-day vulnerability was possibly shared or there might be a third-party entity, such as access brokers, specialized in facilitating Initial Access. This is not the only case indicating the possibility of 0-day vulnerability sharing.

LAC reported the multiple campaignsopen on a new tab, abusing Array AG (CVE-2023-28461) and Citrix (CVE-2023-3466, CVE-2023-3467, CVE-2023-3519), which were abused in the LODEINFO Campaign #2 in 2023 as well. Besides the vulnerabilities, however, there are no overlaps in malware and TTPs in Post-Exploitation between the LODEINFO Campaign #2 and these campaigns. This case suggests the possibility of 0-day sharing or the presence of an access broker, indicating that Earth Kasha may be part of such an ecosystem.

Trend Micro Vision One Threat Intelligence

To stay ahead of evolving threats, Trend Micro customers can access a range of Intelligence Reports and Threat Insights within Trend Micro Vision One. Threat Insights helps customers stay ahead of cyber threats before they happen and better prepared for emerging threats. It offers comprehensive information on threat actors, their malicious activities, and the techniques they use. By leveraging this intelligence, customers can proactively protect their environments, mitigate risks, and respond effectively to threats.

Trend Micro Vision One Intelligence Reports App [IOC Sweeping]

- Spot the difference: Earth Kasha's new LODEINFO campaign and the correlation analysis with the APT10 umbrella

Trend Micro Vision One Threat Insights App

- Threat Actors:
  - Earth Kasha
  - Earth Tengshe
- Emerging Threats: Spot the difference: Earth Kasha's new LODEINFO campaign and the correlation analysis with the APT10 umbrella

Hunting Queries

Trend Micro Vision One Search App

Trend Micro Vision One customers can use the Search App to match or hunt the malicious indicators mentioned in this blog post with data in their environment.

Malware Detection Associated with Earth Kasha

eventName:MALWARE_DETECTION AND malName:(*NOOPLDR* OR *NOOPDOOR* OR *LODEINFO*)

More hunting queries are available for Vision One customers with Threat Insights Entitlement enabledproducts.

Conclusion

We have revealed the new campaign by Earth Kasha and provided an in-depth analysis of LODEINFO, NOOPDOOR and other malware. Additionally, we have analyzed several campaigns in the past and present, suggesting a connection with the previous LODEINFO campaign (LODEINFO Campaign #1) and interesting overlaps with the A41APT Campaign by Earth Tengshe, which is also believed to belong to APT10 Umbrella. These findings lead us to conclude that the same group that conducted the previous LODEINFO campaign also conducted the recent LODEINFO campaign (LODEINFO Campaign #2) with significant TTPs updates. The group may be incorporating or sharing TTPs and tools with Earth Tengshe. Furthermore, our correlational analysis of several campaigns, including the ones by the Volt Typhoon and other unclustered groups, suggested that the 0-day vulnerabilities may be shared among China-nexus actors, or there may be third-party access brokers.

Our research on the recent activity by Earth Kasha highlighted the current complex situation and potential cooperative relationships among China-nexus threat actors. Such a situation will likely continue because it’s beneficial for the adversaries on effective operation and hard for threat intelligence analysts on the attribution. We all need to understand the current complex background and carefully work on the attribution process.

- Appendix A: Checked Applications for Anti-Analysis by NOOPDOOR
- x32dbg**.exe
- x64dbg**.exe
- llydbg**.exe
- windbg**.exe

- ida*.exe
- idaq*.exe
- ImmunityDebugger*.exe
- ProcessHacker*.exe
- Stud_PE*.exe
- pexplorer*.exe
- Autoruns*.exe
- procexp*.exe
- Procmon*.exe
- Tcpview*.exe
- 010Editor*.exe
- WinHex*.exe
- Wireshark*.exe
- zenmap*.exe
- ProcessHacker*.exe
- vmmap*.exe
- load_sc*.exe
- HttpAnalyzerStd*.exe
- Fiddler*.exe

Appendix B: Indicators of Compromise (IoCs)

The indicators of compromise can be found here:
