AKIRA : THE OLD-NEW STYLE CRIME

A THREAT REPORT

AARON JORNET x JOSH PENNY

RexorVc0

VcORExor

Josh penny

2024

CONTENT

1. Executive Summary

The present document compiles the analysis of Tactics, Techniques, and Procedures (TTP) used by the actor as well as several Malware related to Akira, classified as a Ransomware Gang, being one of the most active criminal groups in recent years.

This threat actor, classified as a criminal group, whose main objective is monetary gain through extortion, first appeared in 2022 to start growing their portfolio based on their victims. The attack procedure of Akira Gang has varied over time, where they have been seen to gain access through purchases of credentials in underground markets, as well as exploiting vulnerabilities in VPNs, not exempting the use of Phishing, which continues to be a determining factor in the initial stages of attack by many similar actors.

After gaining access to the infrastructure, attackers conduct phased attacks depending on the target, spacing out their phases to avoid detection. Nevertheless, their movements are often marked by methodologies close to other Ransomware Gangs, where they use different tools and lateral movements to navigate the infrastructure in order to gather information about the victim and perform exfiltration as safely as possible, culminating the attack with the deployment of the Ransomware that gives the group its name, Akira.

The functionality of Akira Ransomware Malware has not varied significantly over the years; however, it remains one of the groups that has had the most impact, where it can be seen managed through commands by the attacker and executed remotely against the infrastructure, affecting files by encrypting them, usually using ChaCha20, in order to demand ransom for the recovery of the affected files.

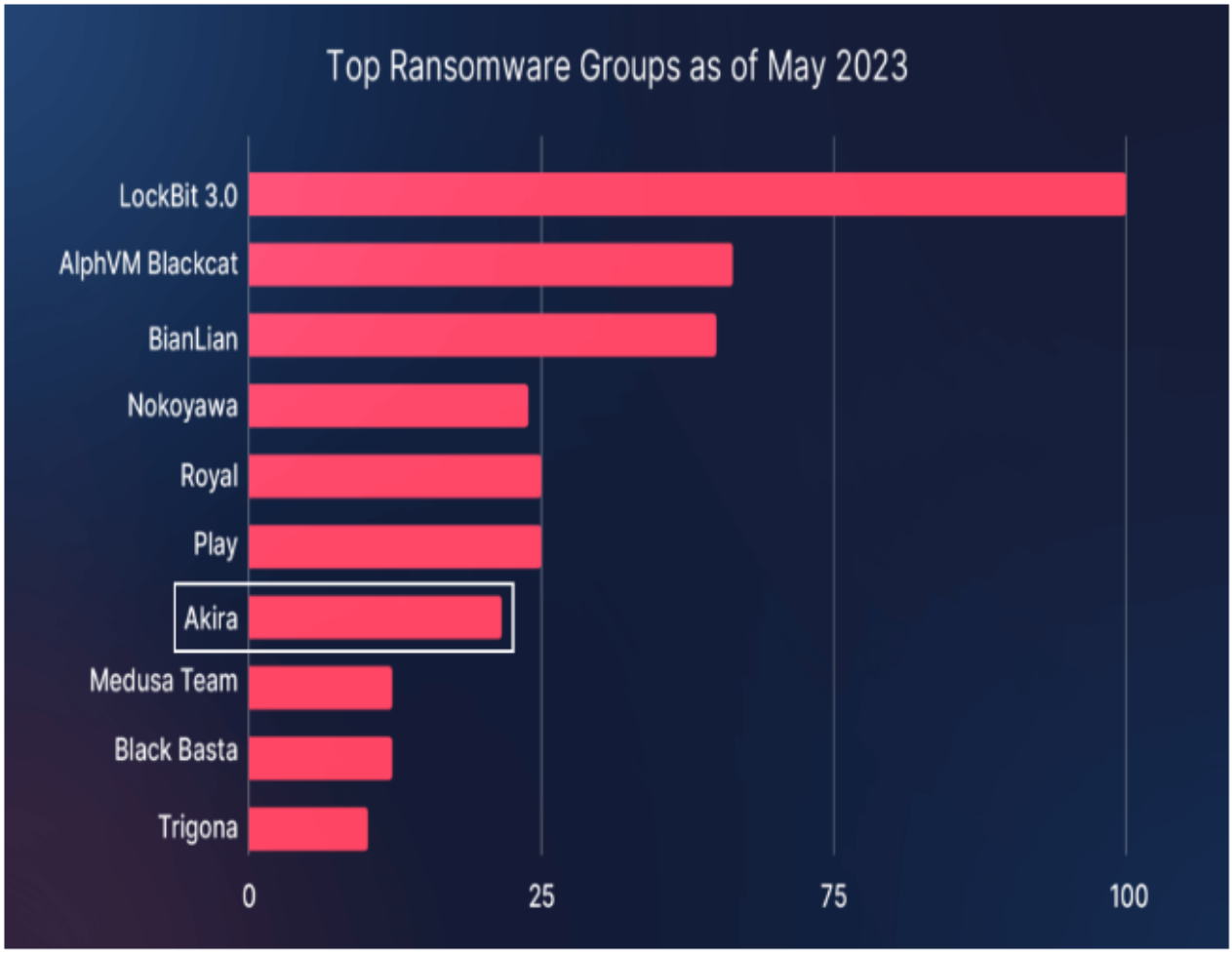
In conclusion, Akira is an organized group whose methodologies are similar to those of its "competitors", who use different tools to carry out their attacks, culminating in their Ransomware, which has not ceased its activity in recent years.

2. Akira History

Akira Gang first emerged in 2022, making various forays into different companies where it began to gain recognition in the few incidents that came to light that same year. However, its notoriety would increase over the years.

During this initial year, there were not many reports of news or companies confirming they had been attacked by Akira. Nevertheless, the popularity it began to gain by late 2022 suggested that many of them had been affected but were not acknowledging the activity of this new actor.

In 2023, it began to enter some ransomware TOPs as its activity increased, and consequently, the number of victims started to grow. From April onwards, Akira rose in the rankings, appearing alongside criminal groups like ALPHV, also known as BlackCat or BianLian.



Top 10 Ransomware Groups (Image Credit: SOCRadar)

In these waves, different types of attacks were observed, including some focused on ESXi. This practice has been seen in many other Ransomware gangs, as targeting these virtualized environments is particularly appealing. They could bypass protections on virtualized devices, potentially affecting the server containing

the machines, which often have lower protections or contain relevant data, being frequently used as data centers or containing virtualized services

Tell me about Akira ransomware's attack on VMware ESXi servers

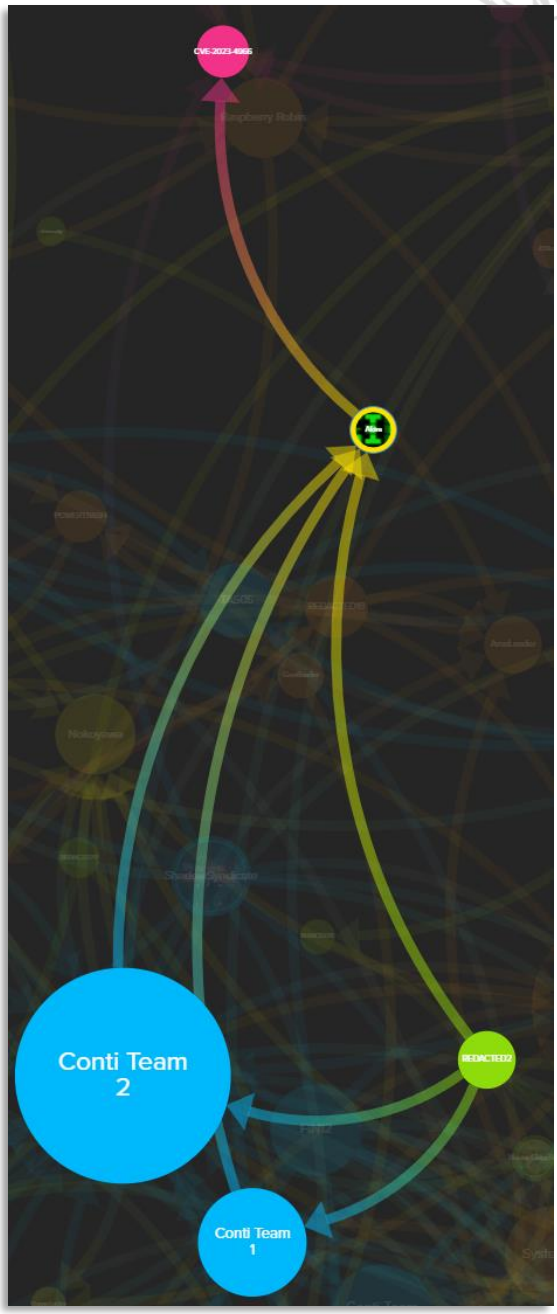
Akira's operation, which has primarily targeted Windows systems across a range of sectors, now includes a project named 'Esxi_Build_Esxi6' specifically designed to attack VMware ESXi servers.

The Linux version of Akira employs command line arguments that allow threat actors to customize their attacks, including designating the percentage of data encrypted on each file. Interestingly, this version of Akira seems to have been ported from the Windows version, given its propensity to skip folders and files typically associated with Windows.

Once activated, the Akira ransomware encrypts a broad range of file extensions, renames files with the .akira extension, and leaves a ransom note named akira_readme.txt in each folder on the encrypted device.

The numbers being mentioned in 2023 were already ranging from ransoms of between $200k and $1 million, so the economic impact they were causing on victim companies was already very high.

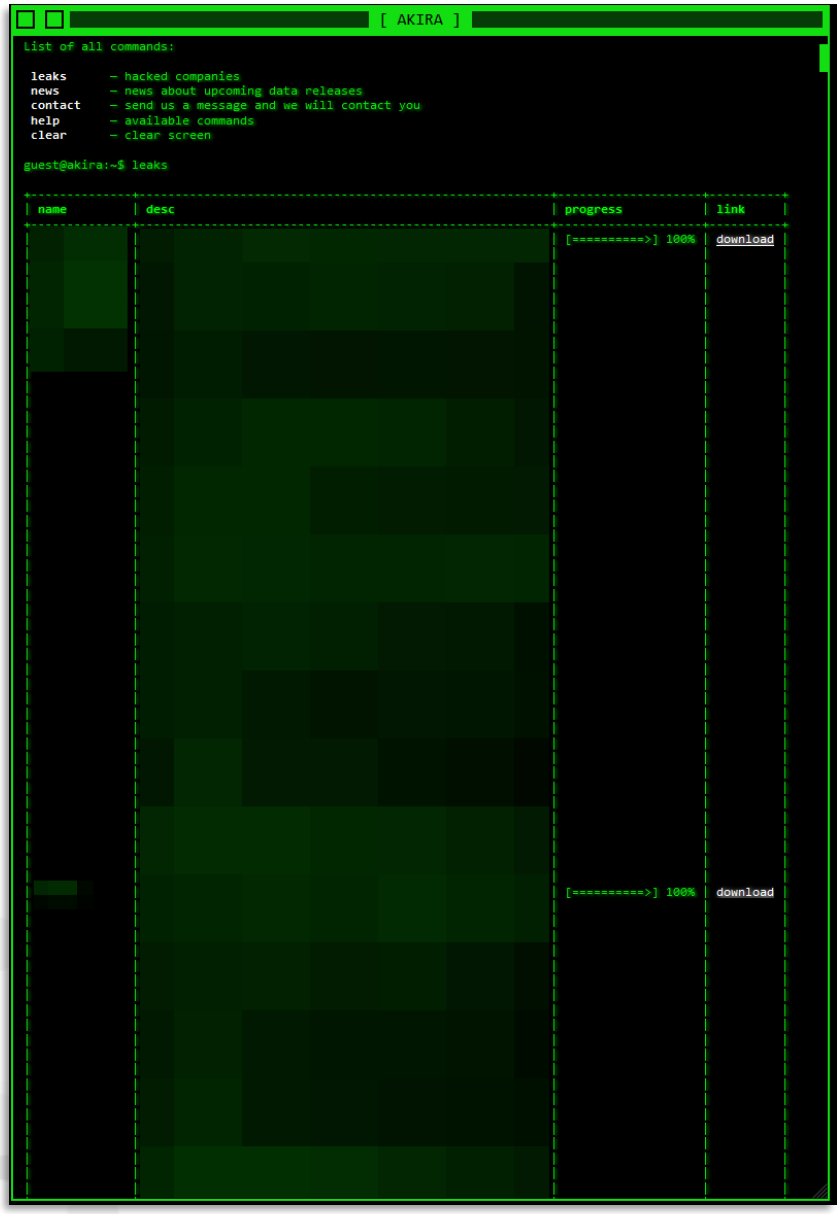
At this point, we had colleagues (@MalGamy12) and companies like Zscaler discussing this ransomware and its evolution compared to Conti, an important point to consider, as ransomware groups often die out and new ones emerge, sometimes using old versions of other gangs as a starting point, or simply being subgroups of these. This reference can also be found in the incredible graph being constructed by pancak3 (@pancak3lullz), where we can see the relationship it has with Conti.



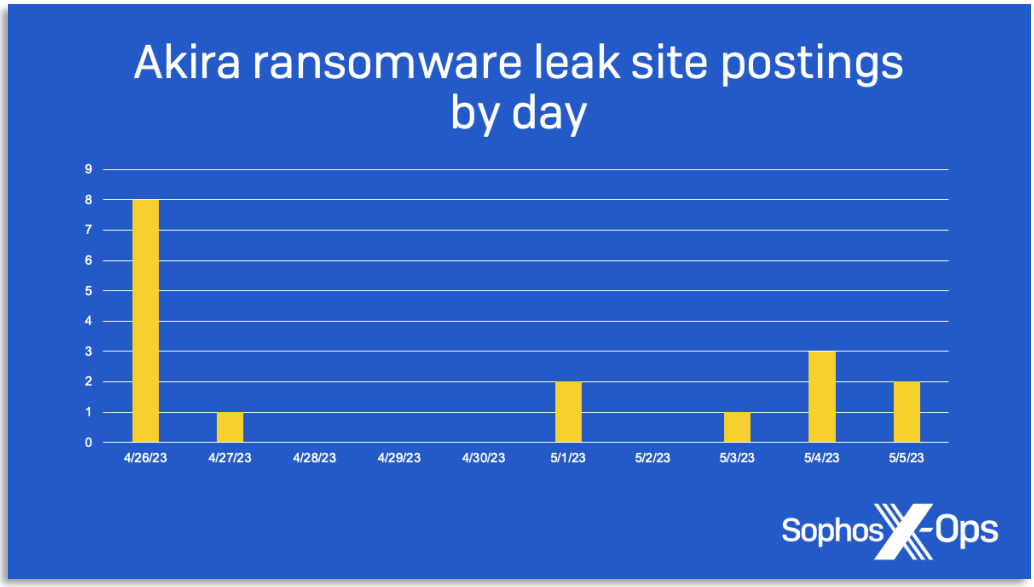
Here it is important to highlight the relationship with Conti, as they are often linked or understood as a subgroup of the Conti matrix, which is always interesting because this group is widely known and has managed to maintain its popularity and business for more than 5 years. When the war between Ukraine and Russia began, there were some very interesting leaks that caused them to disappear (at least for now) from the scene, causing a great stir, as they were usually one of the most active ransomware groups

As usual, when these types of samples and trends start to grow, it is because there are already hundreds of companies that have been affected, but it simply has not leaked to the press. All companies affected by ransomware will try to hide from the public that they have been affected by this type of malware, as it offers great reputational damage in addition to what they are already experiencing in

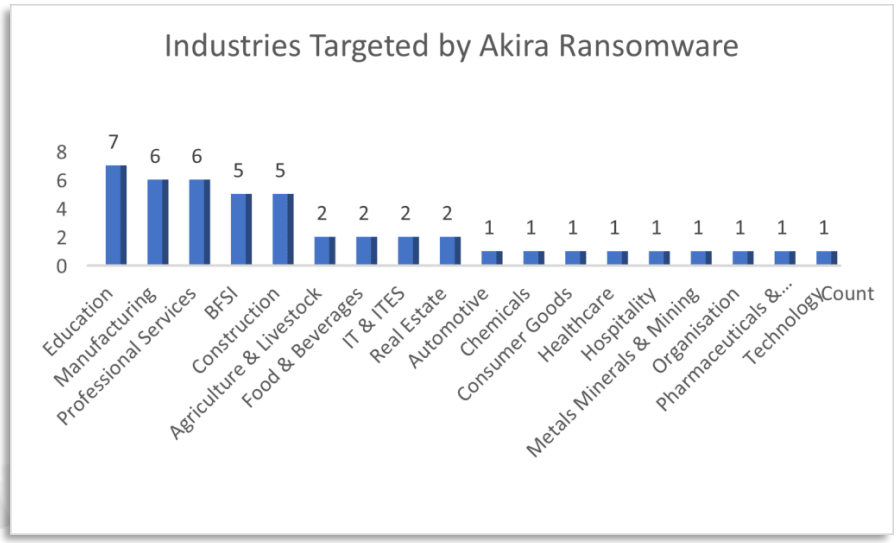
terms of security, data, and services. However, Akira, like many other ransomware groups, would begin to publish information about those from whom a successful exfiltration was achieved. So, as we could see on their old-school portal, each victim would be displayed as a wall of shame, as always, to pressure payment and inflict maximum reputational damage. For this reason (although it sounds obvious), I believe it is important not to display screenshots of companies attacked for that reason, I blur them.



From mid to late 2023, we began to see more reporting activity about Akira, so the popularity of this Gang increased significantly and helped to create statistics to understand the relevance and impact it was having on the world by vendors



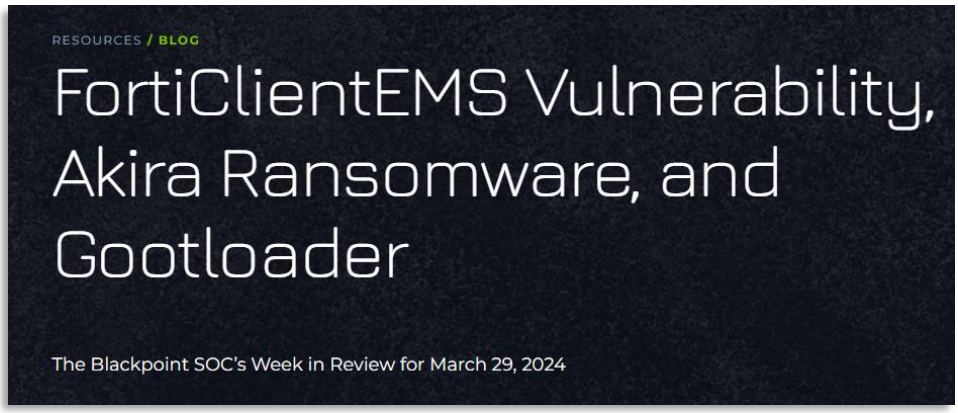
As well as sector estimates by Cyble based on its clients, although these data are never representative because many people cling to these data, but it is important to understand that each company has its clients, in different time zones and in different sectors with respect to other companies, but it is always important to corroborate this information



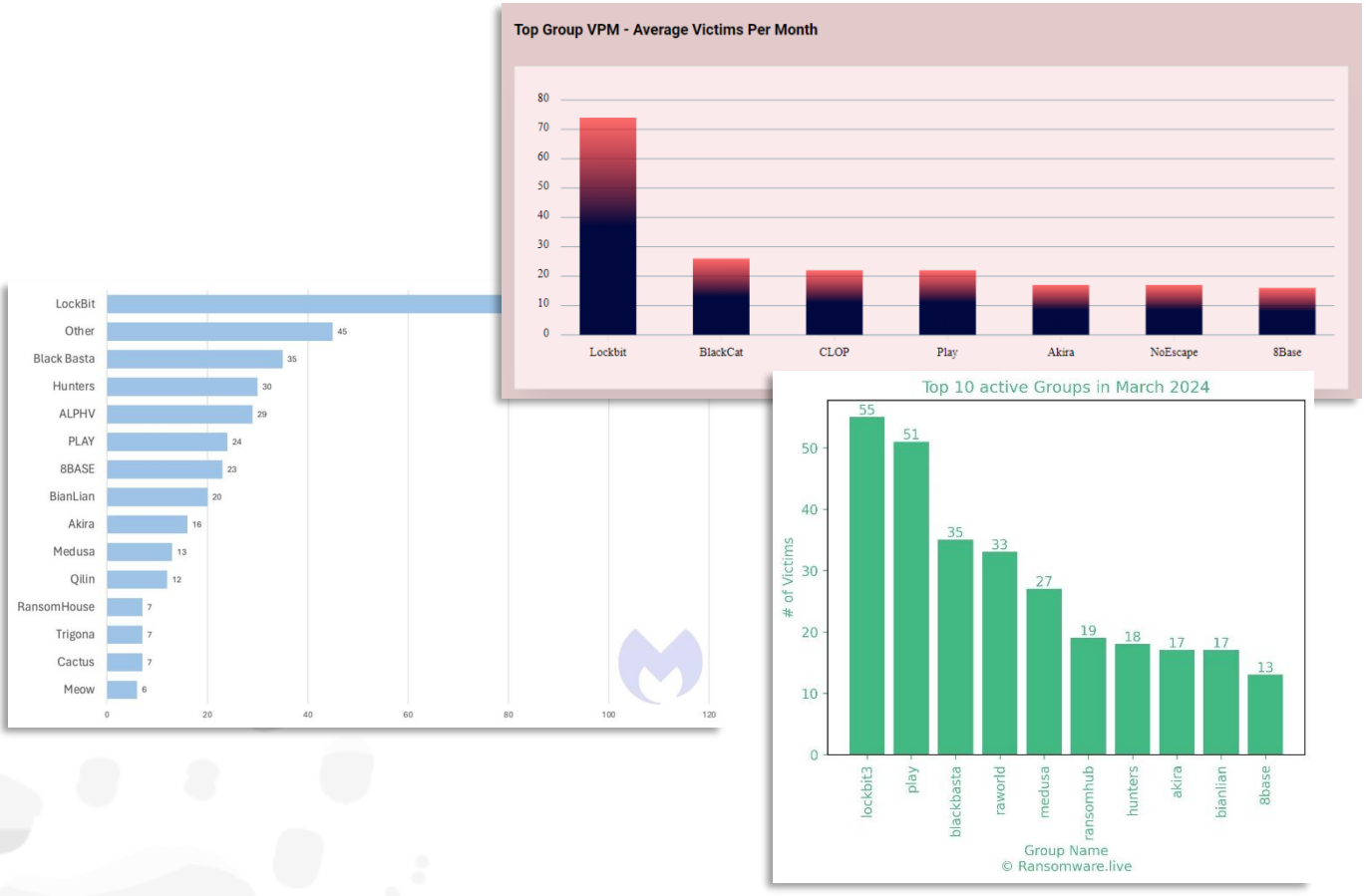
We can see the significant impact on education, which tends to be a common target for many ransomware groups. In their early years of creation, their main targets are often public sectors such as educational systems or administrations, as these state-affiliated entities unfortunately tend to have major security issues. This provides fertile ground for them to practice their skills and refine techniques for more complex objectives.

During this period of the year, we also see that Akira has affected both Windows and Linux environments, which is noteworthy as not all ransomware actors in their early years cause damage to multiple OS. Additionally, the folks at Avast released a decryptor for versions of this ransomware, which managed to help a large number of users.

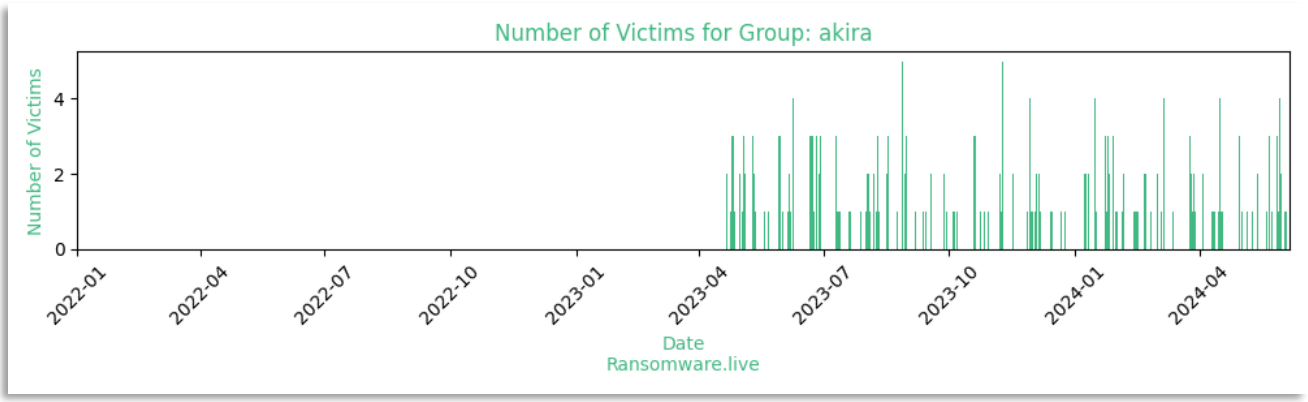
Throughout 2024, the large number of victims continues to increase, exploiting different vulnerabilities such as those we had in Cisco (CVE-2020-3259) or Fortinet (CVE-2023-48788)



Regarding the volume of attacks, we see that it continues to remain within the TOP rankings, so its relevance is maintained or increased depending on the month and who is collecting the information.



In summary, Akira's trend, as we see below, continues during 2024, and the incidents show no signs of stopping, as shown on their website. Like all ransomware groups, they do not come close to showing the number of victims, as there are many companies that pay the ransom, others from which no successful exfiltration has been performed, or similar cases. Therefore, these data must always be taken into account, knowing that attacks and victims are always greater.



So far, no relevant information has been found about those behind Akira, but there is speculation about reaching more than 250 companies, which have made over $40 million with their scope, unfortunately being more profitable than most existing companies.

AKIRA

RANSOMWARE

2022

The first victims of Akira Ransomware appear and we start to talk about the gang that will be decisive for many companies in the following years, which shares a great similarity with the famous Conti

2023

The news exploded and a multitude of companies that have been affected by Akira began to appear, as well as rising in the TOPs, positioning itself as one of the most recurrent in the panorama, carrying out extortions of companies ranging from 200k to a million dollars on average

2024

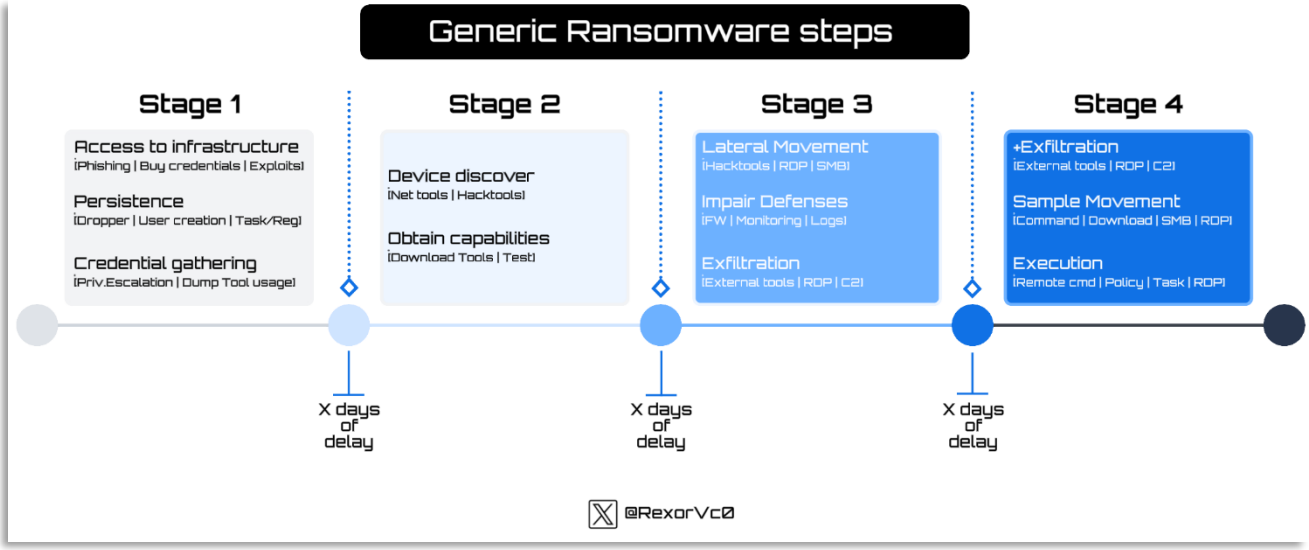
The ransomware gang remains stable in the TOP, having established itself in the global incident landscape rivalling the likes of Play, BianLian or Cl0p, exploiting vulnerabilities and extorting victims without dizzying changes in the way it acts or proceeds with its attacks.

3. ATTACK METHODS

Although ransomware groups are often conservative in their attack methodologies, it's important to understand how they operate, how they divide the phases of their attacks, and, of course, the techniques and malware they employ.

Akira is no exception and tends to work in phases, contrary to what some may think. While they are far from the long dwell time that APTs use to gather information from a company over months and/or years, these organized groups plan in more or less detail depending on how hardened the infrastructure is, steps of attack to avoid raising suspicions or triggering alarms. This way, they can work on obtaining all the necessary information to progress in the following days.

The general phases that almost any ransomware group could use, understanding that it varies by victim and by ransomware group, could be the following:



However, focusing on the activity seen by Akira, although we are going to maintain the number of phases for cohesion, and taking into account that certain activities have occurred in previous or subsequent phases depending on the target, there are time intervals between these phases ranging from days to weeks, completing the attack cycle between 5 and 20 days usually. As you can see, it's nothing like other types of actors. However, it's really critical to have a criminal actor in the infrastructure for a month without triggering alarms, which highlights the technical capability of the group, the lack of hardening of companies, or a combination of both.

Therefore, the phases that Akira Gang has typically used are as follows:

Stage 1: Intrusion

Stage where they focus on accessing the infrastructure, testing credentials, seeing what privileges they have and the handling of the first devices

- Phishing | BruteForce (Weak credentials) | Credentials for sale
  - Sometimes exploit CVE-2023-20269 Cisco
  - Sometimes creation of new accounts for persistence and control
- VPN access to the infrastructure in most cases

Stage 2: Development

Stage in which more credentials are obtained, information is obtained from installed applications and hacktools are downloaded, as well as testing both outside and inside the infrastructure to check access and scope.

- Gathering more credentials (LSASS dumps, Mimikatz, Lazagne, Veeam Backup abuse CVE-2023-27532) or registry abuse
  Downloads/deployment of tools to be used in other phases

Stage 3: Info & Movement

Stage where the network is scanned and the perimeter is checked with hacktools, security systems are disabled and/or the network is prepared to move between computers and out of the infrastructure, tools are also moved.

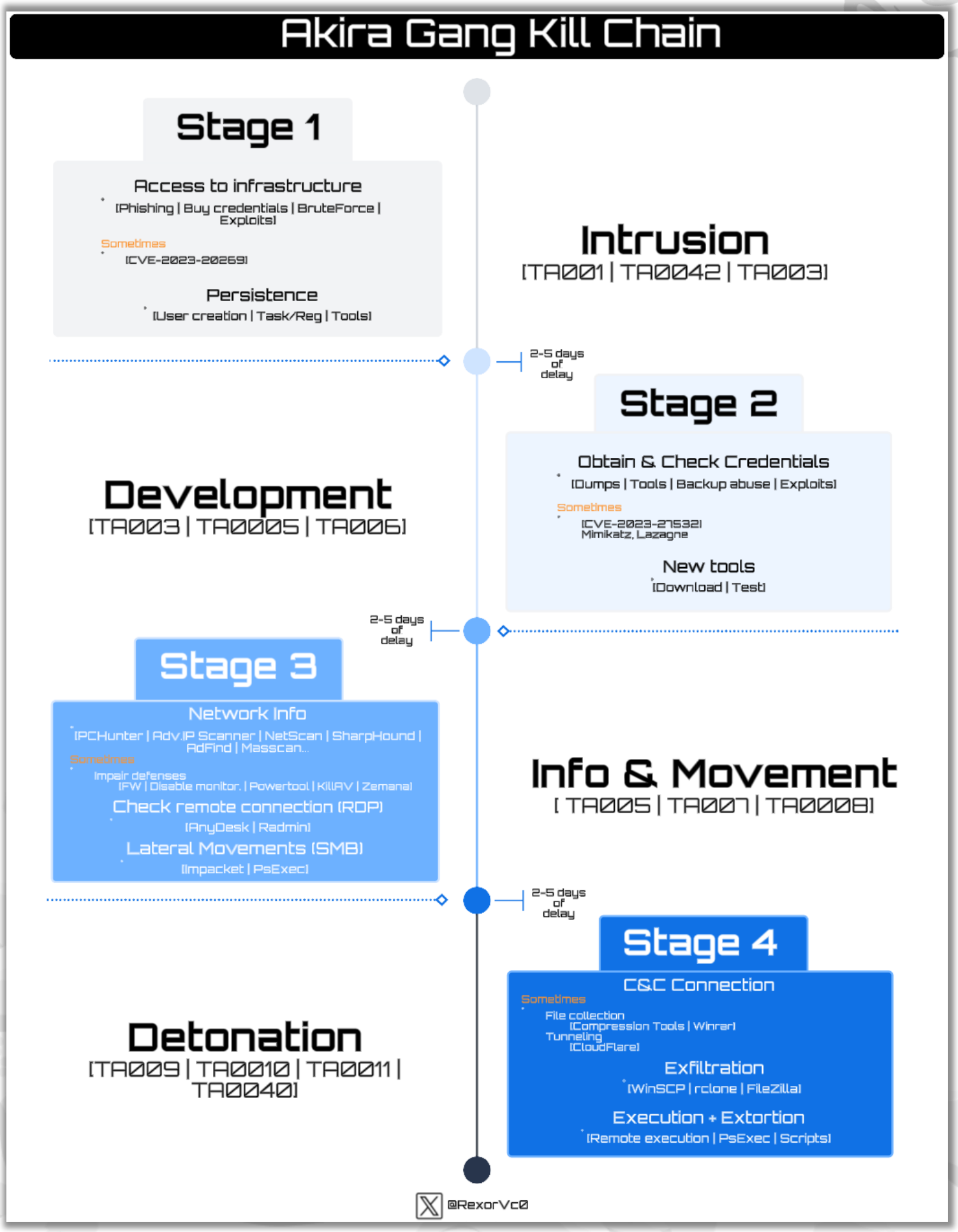
- Obtaining network info (PCHunter, AdvancedIPscanner, Netscan, Sharphound, Adfind, Masscan, SoftPerfect)
  - Sometimes impair defenses (FW, disable realtime, Powertool o KillAV + Zemana Antimalware)
- Connection from outside to other computers, usually via RDP (AnyDesk/Radmin)
- Lateral movements to other computers, usually by SMB (Impacket,Psexec)
- Copy of tools / Copies of Akira or other tools in temporary folders such as ProgramData or AppData

Stage 4: Detonation

Stage in which connections are tested out, data is collected for exfiltration in stages and secure channels are created to share information to subsequently execute the ransomware and demand ransoms

- C&C communication
  - Sometimes Collection of files by compressing them (Winrar)
  - Sometimes creation of tunnels, usually via CloudFlare
- File uploading/Exfiltration, usually via FTP (WinSCP/rclone/FileZilla)
- Execution of ransomware samples (Via NET/Psexec/Scripts)
- Ransom / Post impact extortion

To make it more graphical and understandable at a glance, I think it would be possible with the following chart:


Akira Gang Kill Chain
Stage 1
Access to infrastructure
[Phishing | Buy credentials | BruteForce | Exploits]
Sometimes
[CVE-2023-20269]
Persistence
[User creation | Task/Reg | Tools]
Intrusion
[TA001 | TA0042 | TA003]
2-5 days of delay
Stage 2
Obtain & Check Credentials
[Dumps | Tools | Backup abuse | Exploits]
Sometimes
[CVE-2023-27532]
Mimikatz, Lazagne
New tools
[Download | Test]
Development
[TA003 | TA0005 | TA006]
2-5 days of delay
Stage 3
Network Info
[IPCHunter | Adv.IP Scanner | NetScan | SharpHound | AdFind | Masscan...
Sometimes
Impair defenses
[FW | Disable monitor, | Powertool | KillAV | Zemana]
Check remote connection (RDP)
[AnyDesk | Radmin]
Lateral Movements (SMB)
[Impacket | PsExec]
Info & Movement
[TA005 | TA007 | TA0008]
2-5 days of delay
Stage 4
C&C Connection
Sometimes
File collection
[Compression Tools | Winrar]
Tunneling
[CloudFlare]
Exfiltration
[WinSCP | rclone | FileZilla]
Execution + Extortion
[Remote execution | PsExec | Scripts]
Detonation
[TA009 | TA0010 | TA0011 | TA00401
@RexorVc0

4. Ransomware in depth

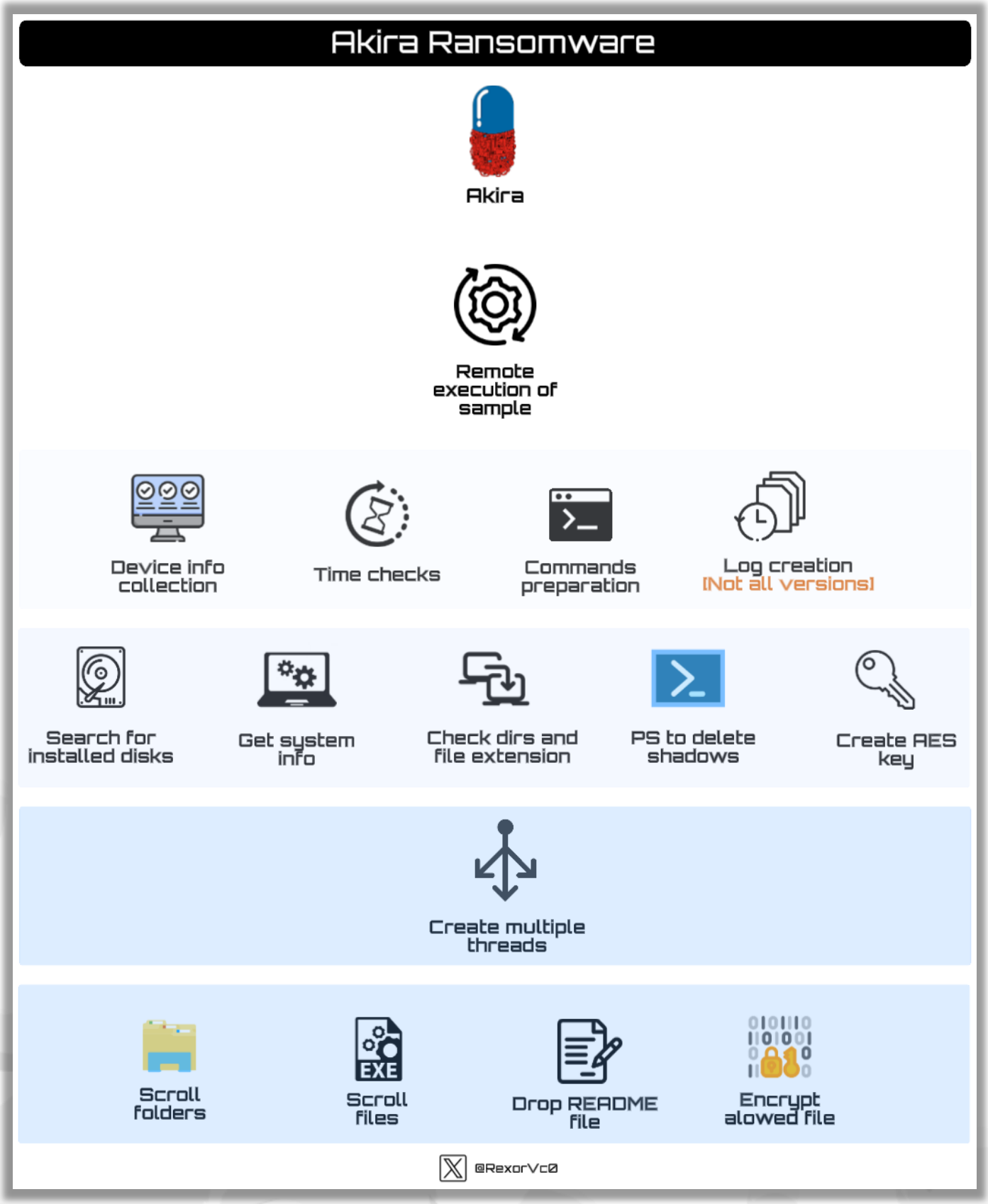
As mentioned in previous sections, the Ransomware used by the Akira group shares its name with the malware itself, and vice-versa. This Ransomware has been active for several years, but its operation has not undergone major changes.

During the analysis of Akira samples, a multitude of samples have been considered in order to locate all available versions, with the intention of better understanding its operation and to exercise a more detailed view of all of them, as well as better detection opportunities based on both the group's behavior and the samples used in their attacks

The general behaviour of the samples, taking into account certain differences, would be given by the following thread:

- After the sample is executed, Akira will obtain information from the affected device, such as machine name, get the timezone, etc.
- Afterwards, it will extract the available commands that can be used to execute the sample and start building a log where it will eventually write based on errors, problems, or useful information about the tasks it is performing. The log will only be present in some versions
- Subsequently, it will start extracting the disks installed on the device and retrieve the internal list of directories and extensions that it will use later. It will also construct a PowerShell command to delete the shadow copies and, depending on the version, will either create the public AES key directly or, alternatively, load AES library and functions in memory and perform the same behaviour at runtime
- After this, Akira will create multiple threads that will simultaneously enter the folders of each disk. It will check, based on its internal list, whether it wants to access each folder, and then check all files against another list to see if it can affect them, while also dropping the ransom note. Meanwhile, it will open the files it wants to work on and encrypt them using ChaCha. Once completed, it will change their extensions

For a better understanding, we show you the behaviour in a graphical way to make it more visual and usable:



4.1. PREPARATION

We begin by positioning ourselves at the start of the execution where we find that the Ransomware starts collecting relevant data such as the time and date on the machine. This can be relevant for different purposes, such as knowing the timezone of the machine

```
000000013F445208   FF15 4AF10200          call qword ptr ds:[<&GetSystemTimeAsFileTime>]
000000013F44520E   48:B9 0080C12A214E62   mov rcx,FE624E212AC18000
000000013F445218   48:B8 008047DD78F083   mov rax,483F078DD478000
000000013F445222   48:034C24 30          add rcx,qword ptr ss:[rsp+30]
000000013F445227   48:3BC8               cmp rcx,rax
```

```
RAX  0000000001DA9ED0
```

In this instance, it obtains the information as it intends to construct a log file with the values obtained earlier

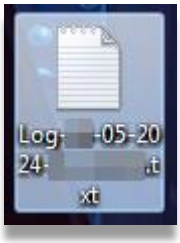
```
lea     r8, aLogDMYHMS  ; "Log-%d-%m-%Y-%H-%M-%S"
mov     edx, 50h
lea     rcx, [rbp+870h+var_80]
call    sub_140094F48
xorps   xmm0, xmm0
movups  [rbp+870h+var_1D0], xmm0
xor     r13d, r13d
mov     [rbp+870h+var_1C0], r13
mov     [rbp+870h+var_1B8], r13
lea     rax, [rbp+870h+var_80]
mov     r8, 0FFFFFFFFFFFFFFFFh
```

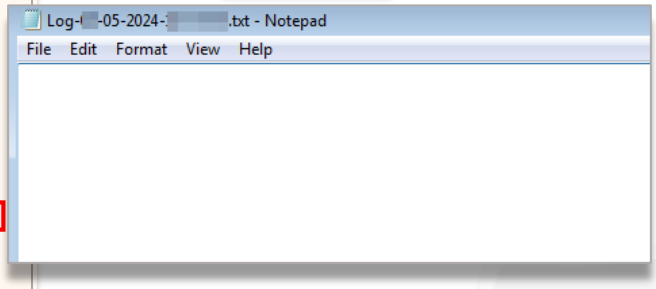
```
000000013F3FCA90   4C:8BC8              mov r9,rax
000000013F3FCA93   4C:8D05 8E5F0800     lea r8,qword ptr ds:[13F482A28]   000000013F482A28:"Log-%d-%m-%Y-%H-%M-%S"
000000013F3FCA9A   BA 50000000          mov edx,50                        50:'P'
000000013F3FCA9F   48:8D8D F0070000     lea rcx,qword ptr ss:[rbp+7F0]
000000013F3FCAA6   E8 9D840400          call kr.13F444F48
```

```
0000000000180D90   4C   L.o.g.-  .-.0.
0000000000180DA0   35   5.-.2.0.2.4.-.1.
0000000000180DB0   31   1.-  ..
0000000000180DC0   0D   .ð.°.ð.°.ð.°.ð.°
```

Later, we find the creation of the log file with an identical structure in the analyzed samples [*Log-DD-MM-YYYY-HH-MM-SS.txt*]

```
000000013F45AA40   C745 E7 18000000   mov dword ptr ss:[rbp-19],18
000000013F45AA47   48:8975 EF          mov qword ptr ss:[rbp-11],rsi
000000013F45AA4B   8945 F7             mov dword ptr ss:[rbp-9],eax
000000013F45AA4E   4C:8975 D7          mov qword ptr ss:[rbp-29],r14
000000013F45AA52   FF15 A0960100       call qword ptr ds:[<&CreateFileW>]
000000013F45AA58   8B5D BB             mov ebx,dword ptr ss:[rbp-45]
000000013F45AA5B   B9 000000C0         mov ecx,C0000000
000000013F45AA60   4C:8BE8             mov r13,rax
000000013F45AA63   48:83F8 FF          cmp rax,FFFFFFFFFFFFFFFF
000000013F45AA67   75 7B               jne kr.13F45AAE4
000000013F45AA69   8BC3                mov eax,ebx
000000013F45AA6B   23C1                and eax,ecx
000000013F45AA6D   3BC1                cmp eax,ecx
000000013F45AA6F   75 40               jne kr.13F45AAB1
000000013F45AA71   41:F6C4 01          test r12b,1
000000013F45AA75   74 3A               je kr.13F45AAB1
000000013F45AA77   8B4D BF             mov ecx,dword ptr ss:[rbp-41]
000000013F45AA7A   4C:8D4D E7          lea r9,qword ptr ss:[rbp-19]
000000013F45AA7E   48:897424 30        mov qword ptr ss:[rsp+30],rsi
000000013F45AA83   0FBAF3 1F           btr ebx,1F
000000013F45AA87   895D BB             mov dword ptr ss:[rbp-45],ebx
000000013F45AA8A   45:8BC7             mov r8d,r15d
000000013F45AA8D   48:8B55 B7          mov rdx,qword ptr ss:[rbp-49]
000000013F45AA91   44:897424 28        mov dword ptr ss:[rsp+28],r14d
000000013F45AA96   894C24 20           mov dword ptr ss:[rsp+20],ecx
000000013F45AA9A   48:8B4D 5F          mov rcx,qword ptr ss:[rbp+5F]
000000013F45AA9E   48:C1EA 20          shr rdx,20
000000013F45AAA2   FF15 50960100       call qword ptr ds:[<&CreateFileW>]
000000013F45AAA8   4C:8BE8             mov r13,rax
000000013F45AAAB   48:83F8 FF          cmp rax,FFFFFFFFFFFFFFFF
000000013F45AAAF   75 33               jne kr.13F45AAE4
000000013F45AAB1   48:630F             movsxd rcx,dword ptr ds:[rdi]
000000013F45AAB4   4C:8D3D 55AA0400    lea r15,qword ptr ds:[13F4A5510]
000000013F45AABB   48:8BC1             mov rax,rcx
```






```
00000000774018E4   894424 20           mov dword ptr ss:[rsp+20],eax
00000000774018E8   E8 9B180100         call <JMP.&CreateFileW>
00000000774018ED   48:8B5C24 60        mov rbx,qword ptr ss:[rsp+60]   [rsp+60]:L"Log-  -05-2024-  .txt"
00000000774018F2   48:8B6C24 68        mov rbp,qword ptr ss:[rsp+68]
00000000774018F7   48:8B7424 70        mov rsi,qword ptr ss:[rsp+70]
00000000774018FC   48:83C4 50          add rsp,50
```

This log is only present in a group of samples, which is the most common in recent months. After this phase, it starts to control the actionable arguments that the threat actor has. These commands are important as they allow us to make decisions based on which one is being used, such as attacking only a specific path, encrypting smaller or larger files, etc

```
call    sub_140053D70
lea     rax, aP_0       ; "-p"
mov     [rbp+870h+var_7F8], rax
lea     rax, aEncryptionPath ; "--encryption_path"
mov     [rbp+870h+var_7F0], rax
lea     rax, [rbp+870h+var_7F8]
mov     qword ptr [rsp+970h+var_900], rax
lea     rax, [rbp+870h+var_7E8]
mov     qword ptr [rsp+970h+var_900+8], rax
movaps  xmm0, [rsp+970h+var_900]
movdqa  [rbp+870h+var_700], xmm0
lea     r8, [rbp+870h+var_700]
lea     rdx, [rbp+870h+var_630]
lea     rcx, [rbp+870h+var_270]
call    sub_14004F250
lea     rcx, [rax+10h]
lea     rdx, [rbp+870h+var_778]
call    sub_1400501D0
mov     rdx, qword ptr [rbp+870h+var_2E0+8]
cmp     rdx, 8
jb      short loc_14004CD1C
```

```
000000013FBC2A34 2D 25 48 2D 25 4D 2D 25 53 00 00 00 2D 00 2D 00 -%H-%M-%S...-.-.
000000013FBC2A44 65 00 6E 00 63 00 72 00 79 00 70 00 74 00 69 00 e.n.c.r.y.p.t.i.
000000013FBC2A54 6F 00 6E 00 5F 00 70 00 61 00 74 00 68 00 00 00 o.n._.p.a.t.h...
000000013FBC2A64 2D 00 70 00 00 00 00 00 2D 00 73 00 00 00 00 00 -.p.....-.s.....
000000013FBC2A74 2D 00 6C 00 00 00 00 00 2D 00 6E 00 00 00 00 00 -.l.....-.n.....
000000013FBC2A84 00 00 00 00 2D 00 2D 00 73 00 68 00 61 00 72 00 ....-.-.s.h.a.r.
000000013FBC2A94 65 00 5F 00 66 00 69 00 6C 00 65 00 00 00 00 00 e._.f.i.l.e.....
000000013FBC2AA4 00 00 00 00 2D 00 6C 00 6F 00 63 00 61 00 6C 00 ....-.l.o.c.a.l.
000000013FBC2AB4 6F 00 6E 00 6C 00 79 00 00 00 00 00 2D 00 2D 00 o.n.l.y.....-.-.
000000013FBC2AC4 65 00 6E 00 63 00 72 00 79 00 70 00 74 00 69 00 e.n.c.r.y.p.t.i.
000000013FBC2AD4 6F 00 6E 00 5F 00 70 00 65 00 72 00 63 00 65 00 o.n._.p.e.r.c.e.
000000013FBC2AE4 6E 00 74 00 00 00 00 00 00 00 00 00 4C 69 73 74 n.t.........List
000000013FBC2AF4 20 6F 66 20 64 72 69 76 65 73 00 00 43 6F 6D 6D  of drives..Comm
000000013FBC2B04 61 6E 64 20 6C 69 6E 65 20 74 6F 20 61 72 67 76 and line to argv
000000013FBC2B14 57 20 66 61 69 6C 65 64 21 00 00 00 49 6E 69 74 W failed!...Init
000000013FBC2B24 20 63 72 79 70 74 6F 20 66 61 69 6C 65 64 21 00  crypto failed!.
000000013FBC2B34 00 00 00 00 4E 6F 20 63 70 75 20 61 76 61 69 6C ....No cpu avail
000000013FBC2B44 61 62 6C 65 21 00 00 00 00 00 00 00 4E 75 6D 62 able!.......Numb
000000013FBC2B54 65 72 20 6F 66 20 74 68 72 65 61 64 20 74 6F 20 er of thread to
000000013FBC2B64 72 6F 6F 74 20 66 6F 6C 64 65 72 20 70 61 72 73 root folder pars
000000013FBC2B74 65 72 73 20 3D 20 00 00 00 00 00 00 4E 75 6D 62 ers = ......Numb
000000013FBC2B84 65 72 20 6F 66 20 74 68 72 65 61 64 20 74 6F 20 er of thread to
000000013FBC2B94 66 6F 6C 64 65 72 20 70 61 72 73 65 72 73 20 3D folder parsers =
000000013FBC2BA4 20 00 00 00 46 61 69 6C 65 64 20 74 6F 20 72 65  ...Failed to re
000000013FBC2BB4 61 64 20 73 68 61 72 65 20 66 69 6C 65 73 21 00 ad share files!.
000000013FBC2BC4 00 00 00 00 4E 75 6D 62 65 72 20 6F 66 20 74 68 ....Number of th
000000013FBC2BD4 72 65 61 64 73 20 74 6F 20 65 6E 63 72 79 70 74 reads to encrypt
```

The list of commands used is as follows:

```
--encryption_path / -p          --share_file / -s          --localonly / -l
                          --encryption_percent / -n
```

I won't provide the definitions for each one, as I noticed that Max (@Libranalysis) had already done so, and who better than Max?

4.2. CONTROL

At this stage, we see how, after obtaining the previously mentioned commands, it focuses on first identifying which drives are available on the affected device, which is very common in this type of malware. To achieve this, it will enumerate and store the drives, and this behavior is similar in most of the analyzed samples. In this case, I only had ***C:*** and ***D:*** available

```
v7 = GetLogicalDriveStringsW(0x104u, lpBuffer[0]);
v4 = lpBuffer[0];
if ( v7 )
{
  v8 = 2i64;
  do
  {
    if ( v4[v8 - 2] )
    {
      v9 = *(_DWORD *)&v4[v8 - 2];
      v10 = v4[v8];
      *(_OWORD *)lpRootPathName = 0i64;
      v26 = 7i64;
      v25 = 3i64;
      LODWORD(lpRootPathName[0]) = v9;
      HIDWORD(lpRootPathName[0]) = v10;
      v27 = 0i64;
      v29 = 7i64;
      LOWORD(v27) = 0;
      v28 = 3i64;
      sub_1400875C0((__m128i *)&v27, (const __m128i *)lpRootPathName, 6ui64);
      WORD3(v27) = 0;
      v11 = (const WCHAR *)lpRootPathName;
      if ( v26 >= 8 )
        v11 = lpRootPathName[0];
      v12 = GetDriveTypeW(v11);
      v30 = v12 == 4;
      v31 = ((v12 - 1) & 0xFFFFFFFB) == 0;
      v15 = *(_QWORD *)(v1 + 8);
      if ( v15 == *(_QWORD *)(v1 + 16) )
      {
        sub_140077530(v1, *(_QWORD *)(v1 + 8), &v27);
      }
      else
      {
        sub_14004FE30(*(_QWORD *)(v1 + 8), &v27, v13, v14);
```

```
loc_14004D3A3:
xorps   xmm0, xmm0
xor     eax, eax
movups  [rbp+870h+var_2D0], xmm0
mov     [rbp+870h+var_2C0], rax
lea     rcx, [rbp+870h+var_2D0]
call    _GetDrives
cmp     rbx, r15
jz      loc_14004D4F8
```

```
00000000004449E0  43 00 3A 00 5C 00 00 00 44 00 3A 00 5C 00 00 00  C.:.\...D.:.\...
00000000004449F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

After this, we see how it obtains data to control temporary elements with *QueryPerformanceFrequency* and *QueryPerformanceCounter*, which can be used for anti-debugging/VM techniques by knowing how long it takes to perform a certain operation or to control timing during encryption

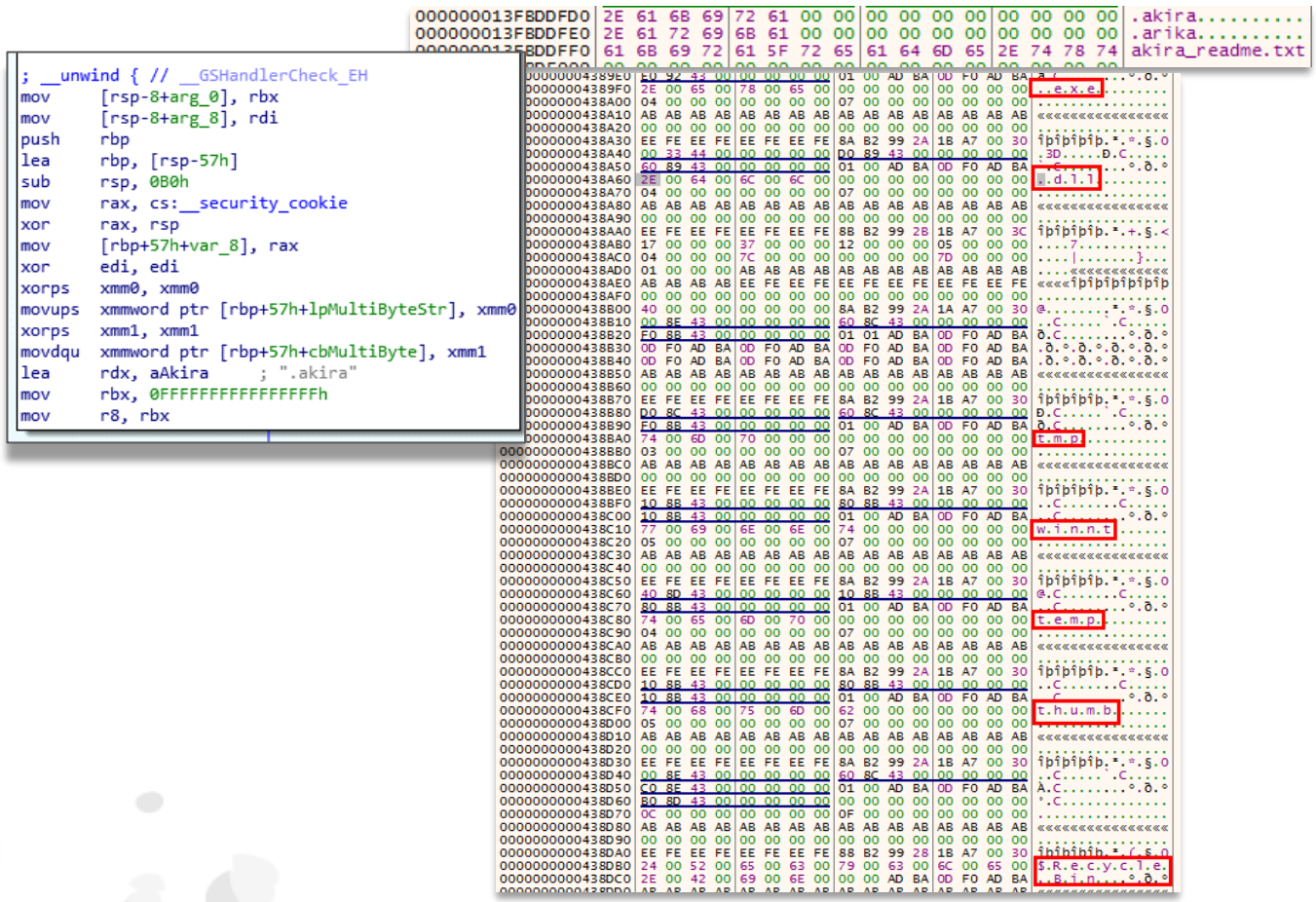
```
loc_14004D4F8:
call    _PerfFreq
mov     rbx, rax
call    _PerfCount
cmp     rbx, 989680h
jnz     short loc_14004D514
```

```
000000013FB68DFE   75 17          jne kr.13FB68E17
000000013FB68E00   48:8D4C24 30   lea rcx,qword ptr ss:[rsp+30]
000000013FB68E05   FF15 EDB40400  call qword ptr ds:[<&QueryPerformanceFrequency>]
000000013FB68E0B   48:8B4424 30   mov rax,qword ptr ss:[rsp+30]
000000013FB68E10   48:8905 59B10700  mov qword ptr ds:[13FBE3F70],rax
000000013FB68DD8   48:8D4C24 30   lea rcx,qword ptr ss:[rsp+30]
000000013FB68DDD   FF15 0DB50400  call qword ptr ds:[<&QueryPerformanceCounter>]
000000013FB68DE3   48:8B4424 30   mov rax,qword ptr ss:[rsp+30]
000000013FB68DE8   48:83C4 28     add rsp,28
```

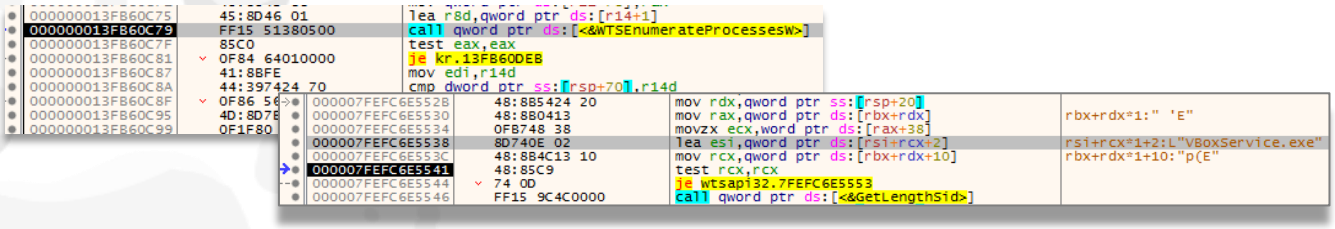
Next, we arrive at very crucial functions, which coincidentally are repeated across the different types of samples

```
loc_14004D52F:
mov     [rbp+870h+var_8F0], rsi
call    sub_14006ECF0
call    sub_140070C30
call    sub_140071D80
lea     rcx, [rbp+870h+SystemInfo] ; lpSystemInfo
call    cs:GetSystemInfo
mov     ebx, [rbp+870h+SystemInfo.dwNumberOfProcessors]
test    ebx, ebx
jnz     loc_14004D5F4
```

The first one is dedicated to preparing both the Akira extension and exceptions for extensions and directories. Most samples more or less match the exceptions, which I will include in an annex later



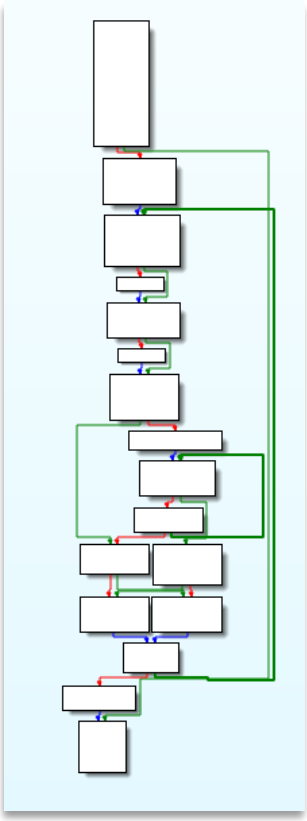
The next function is responsible for obtaining the processes, and this also almost perfectly matches the majority of samples, where instead of taking a snapshot, as is usually done, it performs a *WTSEnumerateProcesses* to control the running processes, which it will later iterate through with a loop



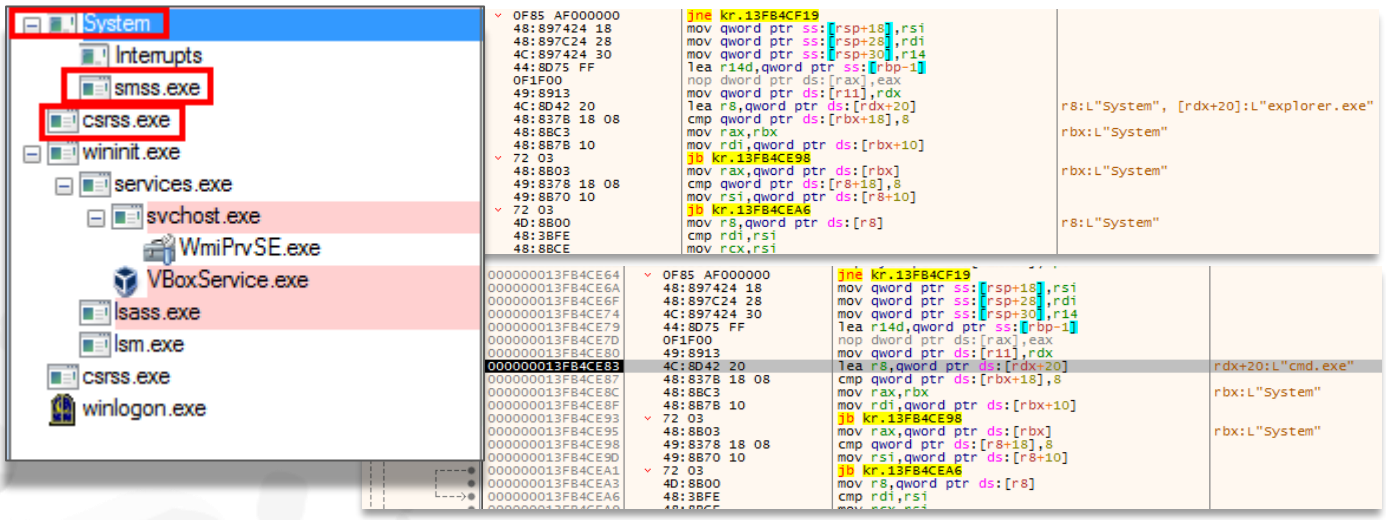
Basically, it will have a function in which it compares active processes collected with the previously obtained list, as it is necessary for Ransomware to control processes and folders that are not of interest, as it would be a mistake to affect the functionality of some executables and potentially alert the user or a

monitoring system that something strange is happening, as well as affect important system files

```
result = WTSEnumerateProcessesW(0i64, 0i64, 1i64, &v19, &v20);
if ( (_DWORD)result )
{
  v1 = 0;
  if ( v20 > 0 )
  {
    while ( 1 )
    {
      v15 = 0i64;
      v16 = 0i64;
      v2 = -1i64;
      v17 = 0i64;
      v3 = 24i64 * v1;
      v4 = *(_QWORD *)(v19 + v3 + 8);
      do
        ++v2;
      while ( *(_WORD *)(v4 + 2 * v2) );
      sub_14003E3A0(&v15, v4, v2);
      v5 = Compare((__int64)&qword_1400F5E28, (__int64)&v18, (__int64)&v15);
      v8 = v15;
      result = *(_QWORD *)(v5 + 16);
      if ( *(_BYTE *)(result + 25) )
        break;
      v9 = *(_QWORD *)(result + 48);
      result += 32i64;
      if ( *(_QWORD *)(result + 24) >= 8ui64 )
        result = *(_QWORD *)result;
      v7 = &v15;
      v6 = v16;
      if ( v17 >= 8 )
        v7 = (__int128 *)v15;
      if ( v9 < v16 )
        v6 = v9;
      if ( v6 )
      {
        v7 = (__int128 *)((char *)v7 - result);
        while ( 1 )
        {
          v10 = *(_WORD *)((char *)v7 + result);
          v11 = v10 < *(_WORD *)result;
          if ( v10 != *(_WORD *)result )
            break;
          result += 2i64;
          if ( !--v6 )
            goto LABEL_16;
        }
        result = 1i64;
```



During this search, we will see how it goes through the active processes with the exclusion list, understanding which ones it can access



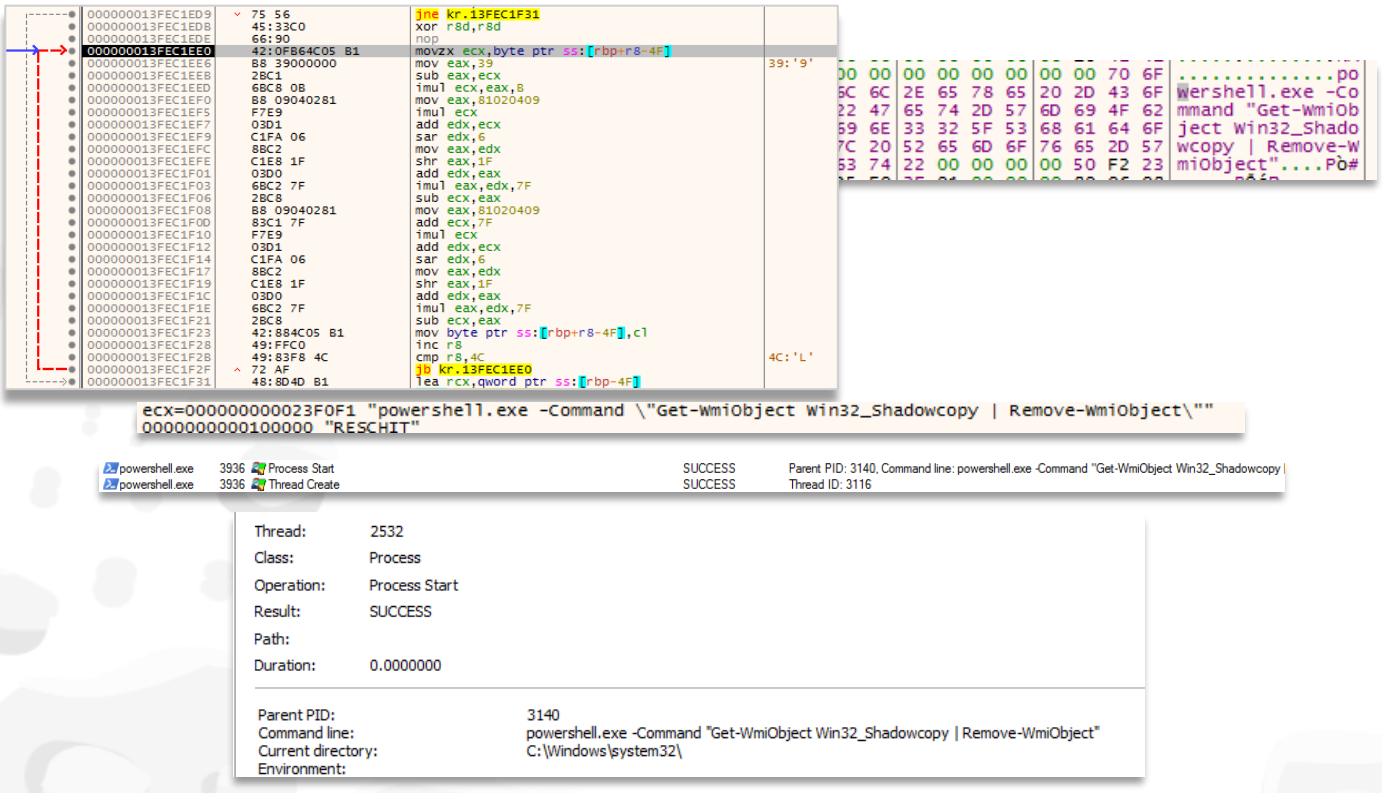
Once it has control over this function, it moves to the third one where the most important element will be the creation of the Powershell command that we will see later. It is interesting because it does not use normal methods of creation, such as *ShellExecute*, but rather something I had seen a few times in Conti (what a surprise), which is the abuse of COM and WMI, both to control privileges in

execution and to create the execution of the command, which is not so common for executing a simple command

```
    do
    {
      *(&v4 + v0) = (11 * (57 - (unsigned __int8)*(&v4 + v0)) % 127 + 127) % 127;
      ++v0;
    }
    while ( v0 < 0x4C );
    v1 = sub_140071620((__int64)&v4);
    if ( v1 )
    {
      v2 = OpenProcess(0x100000u, 0, v1);
      v3 = v2;
      if ( v2 )
      {
        WaitForSingleObject(v2, 0x3A98u);
        CloseHandle(v3);
      }
    }
    CoUninitialize();
  }
}
```

```
v1 = a1;
v2 = 0;
v38 = 0i64;
pProxy = 0i64;
if ( CoCreateInstance(&rclsid, 0i64, 1u, &riid, &v38) < 0 )
  return 0i64;
v3 = SysAllocString(L"ROOT\\CIMV2");
if ( (*(signed int (__fastcall **)(LPVOID, OLECHAR *, _QWORD, _QWORD, _QWORD, _DWORD, _QWORD, _QWORD, IUnknown **))(*(_QWORD *)v38 + 24i64))(
       v38,
       v3,
       0i64,
       0i64,
       0i64,
       0,
       0i64,
       0i64,
       &pProxy) < 0 )
{
  v4 = v38;
  goto LABEL_12;
}
if ( CoSetProxyBlanket(pProxy, 0xAu, 0, 0i64, 3u, 3u, 0i64, 0) < 0 )
  goto LABEL_82;
v5 = SysAllocString(L"Create");
v6 = SysAllocString(L"Win32_Process");
v7 = SysAllocString(L"Win32_ProcessStartup");
v42 = 0i64;
v8 = ((__int64 (__fastcall *)(IUnknown *, OLECHAR *, _QWORD, _QWORD, __int64 *, _QWORD))pProxy->lpVtbl[2].QueryInterface)(
       pProxy,
       v6,
       0i64,
       0i64,
       &v42,
       0i64);
v4 = pProxy;
```

The result we see is Powershell, which, not to break the habit, will be dedicated to deleting Shadows, making it impossible to recover files that will be encrypted later



```
000000013FEC1ED9  75 56        jne kr.13FEC1F31
000000013FEC1EDB  45:33C0      xor r8d,r8d
000000013FEC1EDE  66:90        nop
000000013FEC1EE0  42:0FB64C05 B1  movzx ecx,byte ptr ss:[rbp+r8-4F]
000000013FEC1EE6  B8 39000000  mov eax,39                    39:'9'
000000013FEC1EEB  2BC1         sub eax,ecx
000000013FEC1EED  6BC8 0B      imul ecx,eax,B
000000013FEC1EF0  B8 09040281  mov eax,81020409
000000013FEC1EF5  F7E9         imul ecx
000000013FEC1EF7  03D1         add edx,ecx
000000013FEC1EF9  C1FA 06      sar edx,6
000000013FEC1EFC  8BC2         mov eax,edx
000000013FEC1EFE  C1E8 1F      shr eax,1F
000000013FEC1F01  03D0         add edx,eax
000000013FEC1F03  6BC2 7F      imul eax,edx,7F
000000013FEC1F06  2BC8         sub ecx,eax
000000013FEC1F08  B8 09040281  mov eax,81020409
000000013FEC1F0D  83C1 7F      add ecx,7F
000000013FEC1F10  F7E9         imul ecx
000000013FEC1F12  03D1         add edx,ecx
000000013FEC1F14  C1FA 06      sar edx,6
000000013FEC1F17  8BC2         mov eax,edx
000000013FEC1F19  C1E8 1F      shr eax,1F
000000013FEC1F1C  03D0         add edx,eax
000000013FEC1F1E  6BC2 7F      imul eax,edx,7F
000000013FEC1F21  2BC8         sub ecx,eax
000000013FEC1F23  42:884C05 B1  mov byte ptr ss:[rbp+r8-4F],cl
000000013FEC1F28  49:FFC0      inc r8
000000013FEC1F2B  49:83F8 4C   cmp r8,4C                     4C:'L'
000000013FEC1F2F  72 AF        jb kr.13FEC1EE0
000000013FEC1F31  48:8D4D B1   lea rcx,qword ptr ss:[rbp-4F]
```

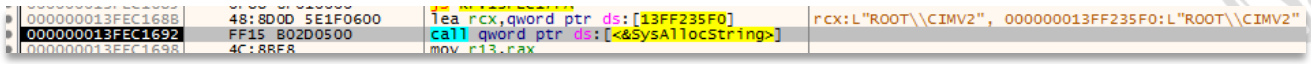
```
00 00 00 00 00 00 00 00 70 6F  ..............po
6C 6C 2E 65 78 65 20 2D 43 6F  wershell.exe -Co
22 47 65 74 2D 57 6D 69 4F 62  mmand "Get-WmiOb
69 6E 33 32 5F 53 68 61 64 6F  ject Win32_Shado
7C 20 52 65 6D 6F 76 65 2D 57  wcopy | Remove-W
53 74 22 00 00 00 00 50 F2 23  miObject"....Pò#
```

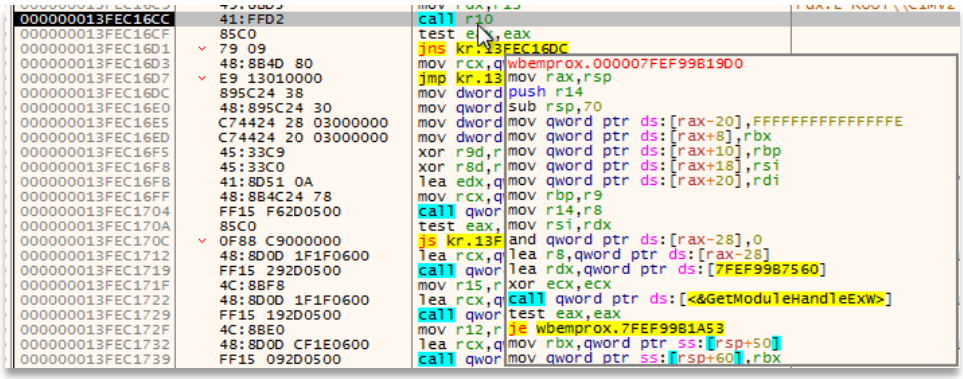
```
ecx=000000000023F0F1 "powershell.exe -Command \"Get-WmiObject Win32_Shadowcopy | Remove-WmiObject\""
0000000000100000 "RESCHIT"
```

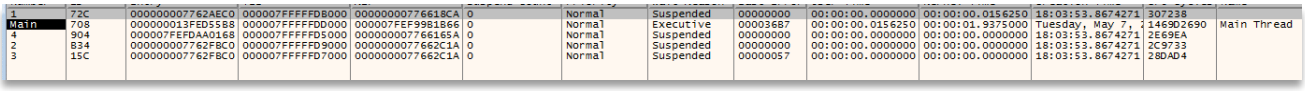
powershell.exe	3936	Process Start	SUCCESS	Parent PID: 3140, Command line: powershell.exe -Command "Get-WmiObject Win32_Shadowcopy |
powershell.exe	3936	Thread Create	SUCCESS	Thread ID: 3116

Thread:	2532
Class:	Process
Operation:	Process Start
Result:	SUCCESS
Path:	
Duration:	0.0000000
Parent PID:	3140
Command line:	powershell.exe -Command "Get-WmiObject Win32_Shadowcopy | Remove-WmiObject"
Current directory:	C:\Windows\system32\
Environment:	

An important element, in the execution prior to the creation of this event, is the creation of multiple threads, as this Ransomware, like others, will work with multiple threads to be more efficient and certainly faster in its execution







An important and very differentiating point between the two main types of Akira existing is when we talk about the creation of the AES key, as the more recent samples are somewhat stealthier. After the creation of the previous points, it starts to load libraries, among which *rsaenh.dll* stands out

```
call    cs:SysAllocString
mov     r13, rax
mov     rcx, [rbp+90h+var_110]
mov     rdx, [rcx]
mov     r10, [rdx+18h]
lea     rax, [rsp+190h+pProxy]
mov     [rsp+190h+var_150], rax
mov     qword ptr [rsp+190h+dwCapabilities], rbx
mov     [rsp+190h+pAuthInfo], rbx
mov     [rsp+190h+dwImpLevel], ebx
mov     [rsp+190h+ppv], rbx
xor     r9d, r9d
xor     r8d, r8d
mov     rdx, r13
call    r10             ; GetModHandl + GetProcAddr + Threads
test    eax, eax
```

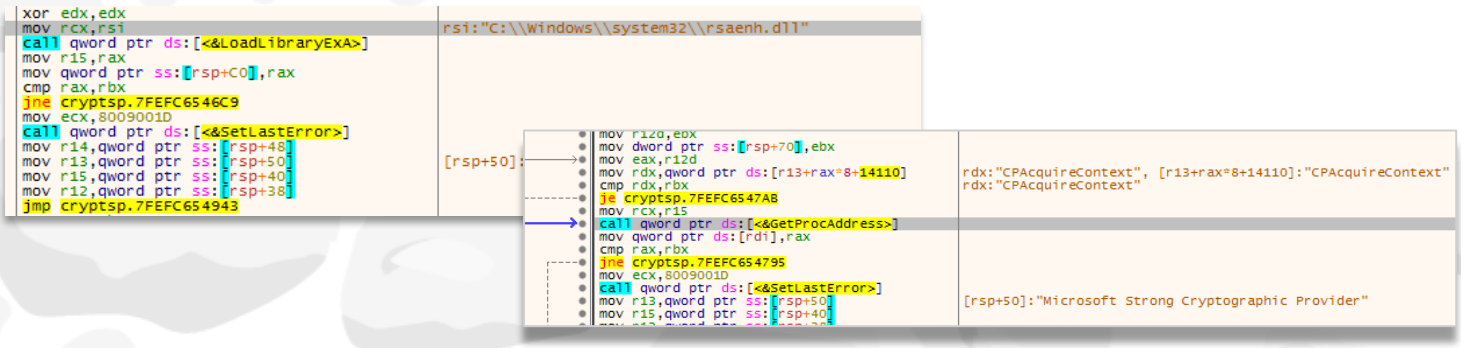
```
push rdi
sub rsp,50
mov ebx,edx
mov rdi,rcx
mov rdx,rcx                                  rcx:L"C:\\Windows\\system32\\rsaenh.dll"
lea rcx,qword ptr ss:[rsp+40]                rcx:L"C:\\Windows\\system32\\rsaenh.dll"
mov rsi,r9
mov ebp,r8d
call qword ptr ds:[<&RtlInitUnicodeStringEx>]
```

The following functions are exported by this dll:

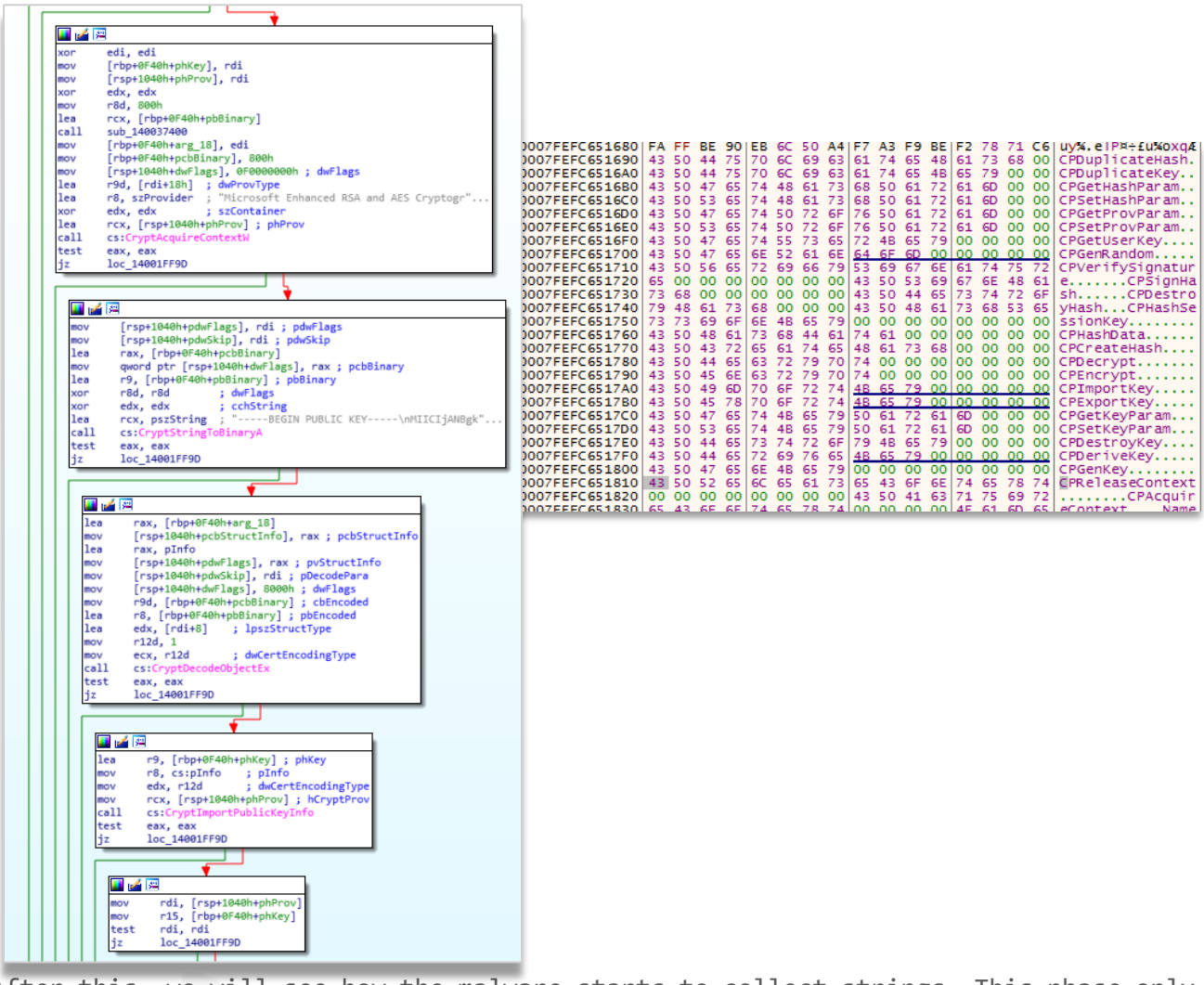
CPAcquireContext	CPCreateHash	CPDecrypt	CPDeriveKey
CPDestroyHash	CPDestroyKey	CPDuplicateHash	CPDuplicateKey
CPEncrypt	CPExportKey	CPGenKey	CPGenRandom
CPGetHashParam	CPGetKeyParam	CPGetProvParam	CPGetUserKey
CPHashData	CPHashSessionKey	CPImportKey	CPReleaseContext
CPSetHashParam	CPSetKeyParam	CPSetProvParam	CPSignHash
CPVerifySignature	DllRegisterServer	DllUnregisterServer	

```
000007FEFC65383C 4D 5A 00 00 11 40 04 03 02 04 03 65 01 48 86 60 MZ...@.....e.H.`
000007FEFC65384C 09 06 0B 30 4F 30 00 00 00 00 00 00 13 40 04 00 ...0O0.......@..
000007FEFC65385C 05 03 02 04 03 65 01 48 86 60 09 06 0D 30 51 30 .....e.H.`...0Q0
000007FEFC65386C 00 00 00 00 11 30 04 02 02 04 03 65 01 48 86 60 .....0.....e.H.`
000007FEFC65387C 09 06 0B 30 3F 30 00 00 00 00 00 00 13 30 04 00 ...0?0.......0..
000007FEFC65388C 05 02 02 04 03 65 01 48 86 60 09 06 0D 30 41 30 .....e.H.`...0A0
000007FEFC65389C 00 00 00 00 11 20 04 01 02 04 03 65 01 48 86 60 ..... .....e.H.`
000007FEFC6538AC 09 06 0B 30 2F 30 00 00 00 00 00 00 13 20 04 00 ...0/0....... ..
000007FEFC6538BC 05 01 02 04 03 65 01 48 86 60 09 06 0D 30 31 30 .....e.H.`...010
000007FEFC6538CC 00 00 00 00 0D 14 04 1A 02 03 0E 2B 05 06 07 30 ...........+...0
000007FEFC6538DC 1F 30 00 00 0F 14 04 00 05 1A 02 03 0E 2B 05 06 .0...........+..
000007FEFC6538EC 09 30 21 30 00 00 00 00 00 00 00 00 10 10 04 05 .0!0............
000007FEFC6538FC 02 0D F7 86 48 86 2A 08 06 0A 30 1E 30 00 00 00 ....H.*...0.0...
000007FEFC65390C 00 00 00 00 12 10 04 00 05 05 02 0D F7 86 48 86 ..............H.
000007FEFC65391C 2A 08 06 0C 30 20 30 00 00 00 00 00 00 00 00 00 *...0 0.........
000007FEFC65392C 00 00 00 00 53 00 4F 00 46 00 54 00 57 00 41 00 ....S.O.F.T.W.A.
000007FEFC65393C 52 00 45 00 5C 00 4D 00 69 00 63 00 72 00 6F 00 R.E.\.M.i.c.r.o.
000007FEFC65394C 73 00 6F 00 66 00 74 00 5C 00 53 00 79 00 73 00 s.o.f.t.\.S.y.s.
000007FEFC65395C 74 00 65 00 6D 00 43 00 65 00 72 00 74 00 69 00 t.e.m.C.e.r.t.i.
000007FEFC65396C 66 00 69 00 63 00 61 00 74 00 65 00 73 00 5C 00 f.i.c.a.t.e.s.\.
000007FEFC65397C 52 00 6F 00 6F 00 74 00 5C 00 43 00 65 00 72 00 R.o.o.t.\.C.e.r.
000007FEFC65398C 74 00 69 00 66 00 69 00 63 00 61 00 74 00 65 00 t.i.f.i.c.a.t.e.
000007FEFC65399C 73 00 5C 00 38 00 41 00 33 00 33 00 34 00 41 00 s.\.8.A.3.3.4.A.
000007FEFC6539AC 41 00 38 00 30 00 35 00 32 00 44 00 44 00 32 00 A.8.0.5.2.D.D.2.
000007FEFC6539BC 34 00 34 00 41 00 36 00 34 00 37 00 33 00 30 00 4.4.A.6.4.7.3.0.
000007FEFC6539CC 36 00 41 00 37 00 36 00 42 00 38 00 31 00 37 00 6.A.7.6.B.8.1.7.
000007FEFC6539DC 38 00 46 00 41 00 32 00 31 00 35 00 46 00 33 00 8.F.A.2.1.5.F.3.
```

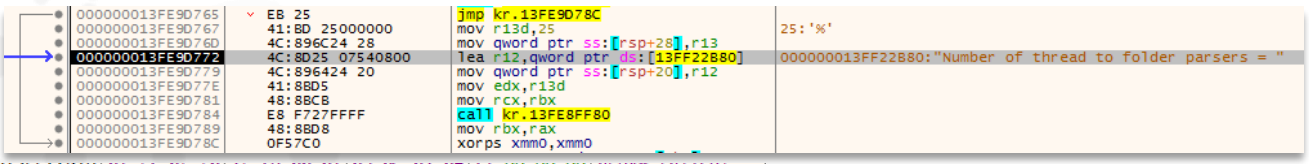
Obviously, the original sample does not include these types of functions, so it loads them at runtime, which is not so unusual, but it does show an evolution compared to previous samples. The way to proceed is as usual with *LoadLibrary+GetProcAddress* to obtain the library and functions

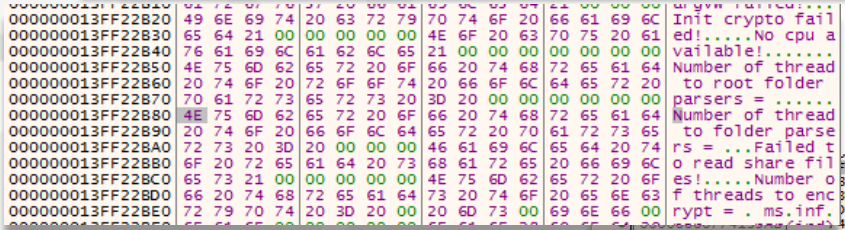


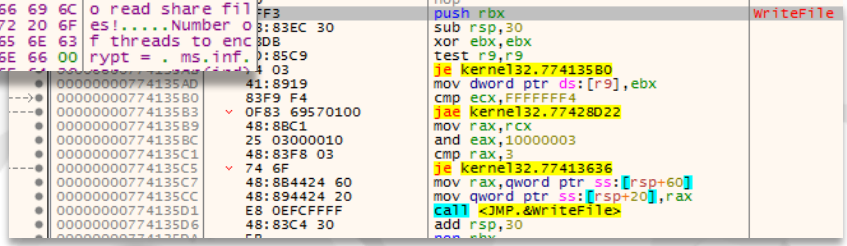
In the end, we obtain the same functionality, but only during execution, which complicates detection because these types of imports are usually very indicative in this type of samples of a Ransomware. Therefore, as we mentioned before, there has been an evolution in certain aspects



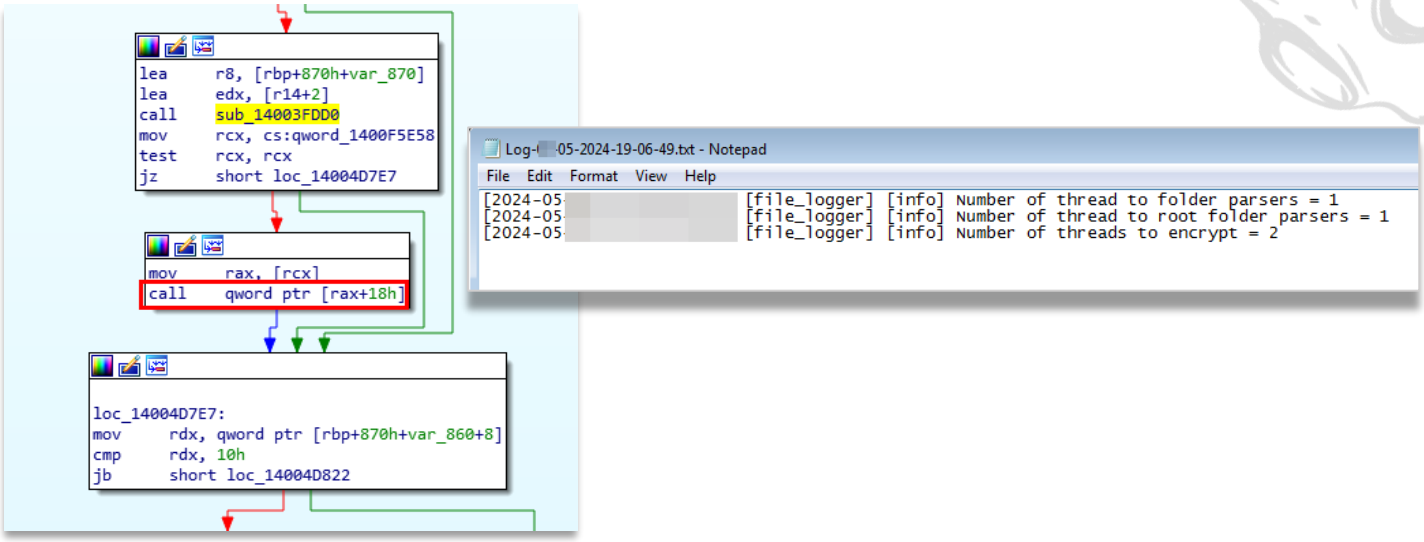
After this, we will see how the malware starts to collect strings. This phase only appears in one type of sample, as I mentioned earlier, so we will see how it retrieves them to subsequently write them. In these strings, even though it is very evident, it clearly documents the threads it will use for each action



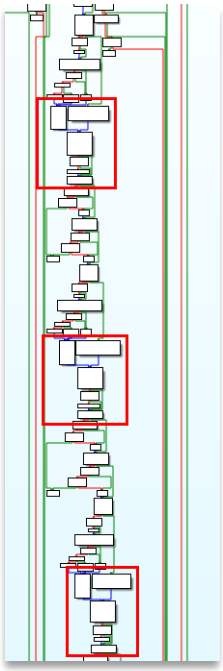




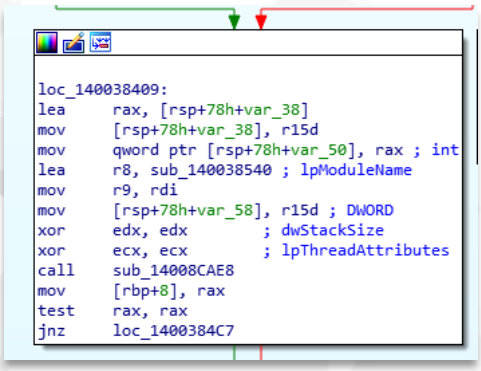
X JOSH_PENNY X REXORVCO

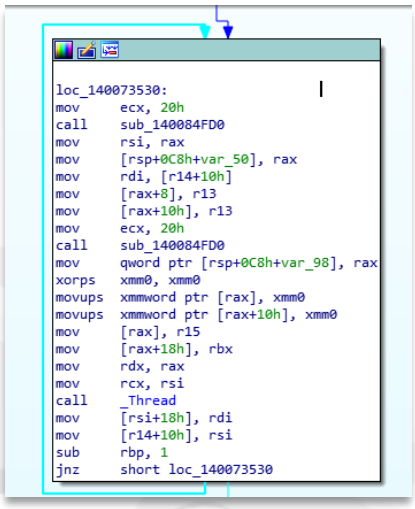


We can clearly see the pattern where it recursively retrieves each of the strings to log them



Subsequently, we start to arrive at more relevant functions, where we find more thread creations through a loop that will generate them. I have seen executions with up to 8 threads

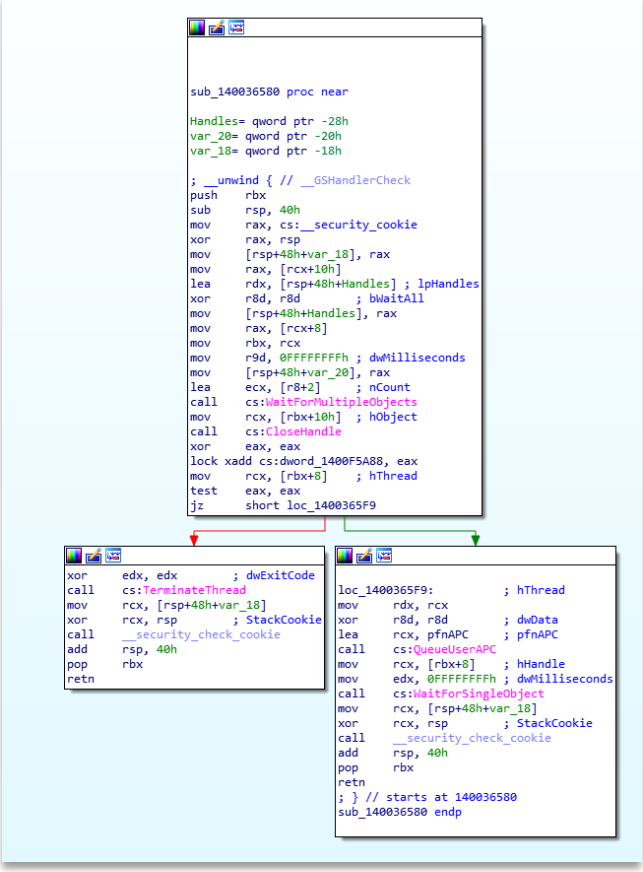




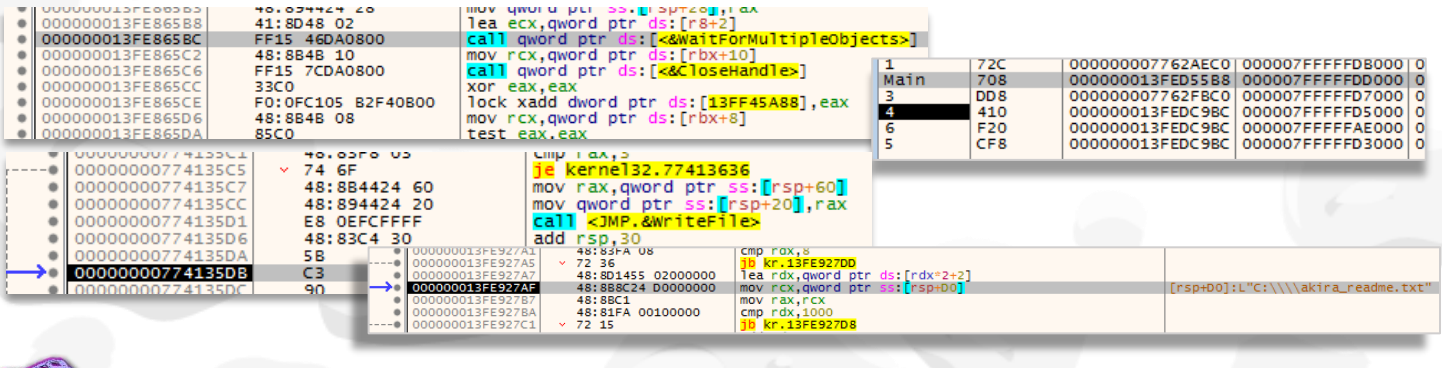
3	FA4	000000013FEDC9BC	000007FFFFFD9000	00000000776618CA	0	Normal	Suspended
1	72C	000000007762AEC0	000007FFFFFDB000	00000000776618CA	0	Normal	Suspended
Main	708	000000013FED55B8	000007FFFFFDD000	000000013FEC3574	0	Normal	Executive
3	DD8	000000007762FBC0	000007FFFFFD7000	00000000077662C1A	0	Normal	Suspended
4	410	000000013FEDC9BC	000007FFFFFD5000	00000000776618CA	0	Normal	Suspended

Number	ID	Entry	TEB	RIP	Suspend Count	Priority	Wait Reason
3	FA4	000000013FEDC9BC	000007FFFFFD9000	00000000776618CA	0	Normal	Suspended
1	72C	000000007762AEC0	000007FFFFFDB000	00000000776618CA	0	Normal	Suspended
Main	708	000000013FED55B8	000007FFFFFDD000	000000013FEC3579	0	Normal	Executive
3	DD8	000000007762FBC0	000007FFFFFD7000	00000000077662C1A	0	Normal	Suspended
4	410	000000013FEDC9BC	000007FFFFFD5000	00000000776618CA	0	Normal	Suspended
5	CF8	000000013FEDC9BC	000007FFFFFD3000	00000000776618CA	0	Normal	Suspended

Just like other ransomware such as Sodinokibi, Akira needs to control multiple threads simultaneously. The malware manages this process by switching from one thread to another as needed. To properly handle this coordination, Akira uses *WaitForMultipleObjects* and *QueueUserAPC* to add procedures to an execution queue and coordinate what will be executed in the threads, waiting for certain conditions to be met. This approach is similar to other ransomware, and as a spoiler, it will later use *PostQueuedCompletionStatus* and *GetQueuedCompletionStatus* to manage the input and output of operations. In this process, *CreateIoCompletionPort* will also be used to handle the completion of tasks. These elements are crucial in the analysis, as they determine the various asynchronous executions that the threads will perform



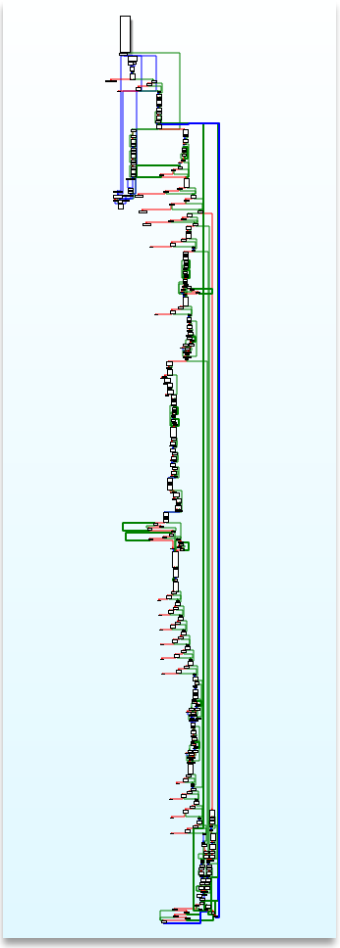
Initially, we see this being used for the creation of the readme file, but that is something for the next section.



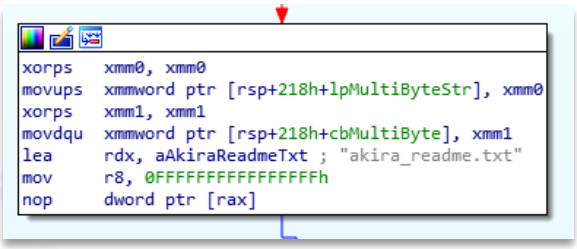
4.3. Impact

In this phase, we will see how it manages the ransom notes, as well as how it affects the files and behaves in all these executions, switching from one thread to another.

For this entire phase, it uses a really interesting (and lengthy) function, whose capability will be to manage which folders it wants or does not want to enter, to leave the ransom notes. Obviously, this is not its only function



First, it will prepare the Readme file, whose content will be visible statically



```
loc_1400426B7:
movdqa  xmm0, cs:xmmword_1400D74F0
movdqu  xmmword ptr [rsp+218h+cbMultiByte], xmm0
mov     byte ptr [rsp+218h+lpMultiByteStr], 0
xor     edx, edx
mov     r8d, 108h
lea     rcx, [rsp+218h+var_128]
call    sub_140087C60
lea     rdx, [rsp+218h+var_148]
lea     rcx, [rsp+218h+var_128]
call    sub_140043610
nop
lea     rdx, aHiFriendsWhate ; "Hi friends,\r\n\r\nWhatever who you are"...
lea     rcx, [rsp+218h+var_128]
call    sub_14003FB10
nop
mov     rax, [rsp+218h+var_128]
movsxd  rcx, dword ptr [rax+4]
lea     rax, ??_7?$basic_ofstream@DU?$char_traits@D@std@@@std@@6B@ ; const st
mov     [rsp+rcx+218h+var_128], rax
mov     rax, [rsp+218h+var_128]
movsxd  rcx, dword ptr [rax+4]
lea     edx, [rcx-0A8h]
mov     dword ptr [rsp+rcx+218h+anonymous_0+4], edx
lea     rcx, [rsp+218h+var_120]
call    sub_140044370
```

```
aHiFriendsWhate db 'Hi friends,',0Dh,0Ah
                                        ; DATA XREF: _ReadmeFile+2D5↑o
                db 0Dh,0Ah
                db 'Whatever who you are and what your title is if you',27h,'re readi'
                db 'ng this it means the internal infrastructure of your company is f'
                db 'ully or partially dead, all your backups - virtual, physical - ev'
                db 'erything that we managed to reach - are completely removed. Moreo'
                db 'ver, we have taken a great amount of your corporate data prior to'
                db ' encryption.',0Dh,0Ah
                db 0Dh,0Ah
                db 'Well, for now let',27h,'s keep all the tears and resentment to ou'
                db 'rselves and try to build a constructive dialogue. We',27h,'re ful'
                db 'ly aware of what damage we caused by locking your internal source'
                db 's. At the moment, you have to know:',0Dh,0Ah
                db 0Dh,0Ah
                db '1. Dealing with us you will save A LOT due to we are not interest'
                db 'ed in ruining your financially. We will study in depth your finan'
                db 'ce, bank & income statements, your savings, investments etc. and '
                db 'present our reasonable demand to you. If you have an active cyber'
                db ' insurance, let us know and we will guide you how to properly use'
                db ' it. Also, dragging out the negotiation process will lead to fail'
                db 'ing of a deal.',0Dh,0Ah
                db '2. Paying us you save your TIME, MONEY, EFFORTS and be back on tr'
                db 'ack within 24 hours approximately. Our decryptor works properly o'
```

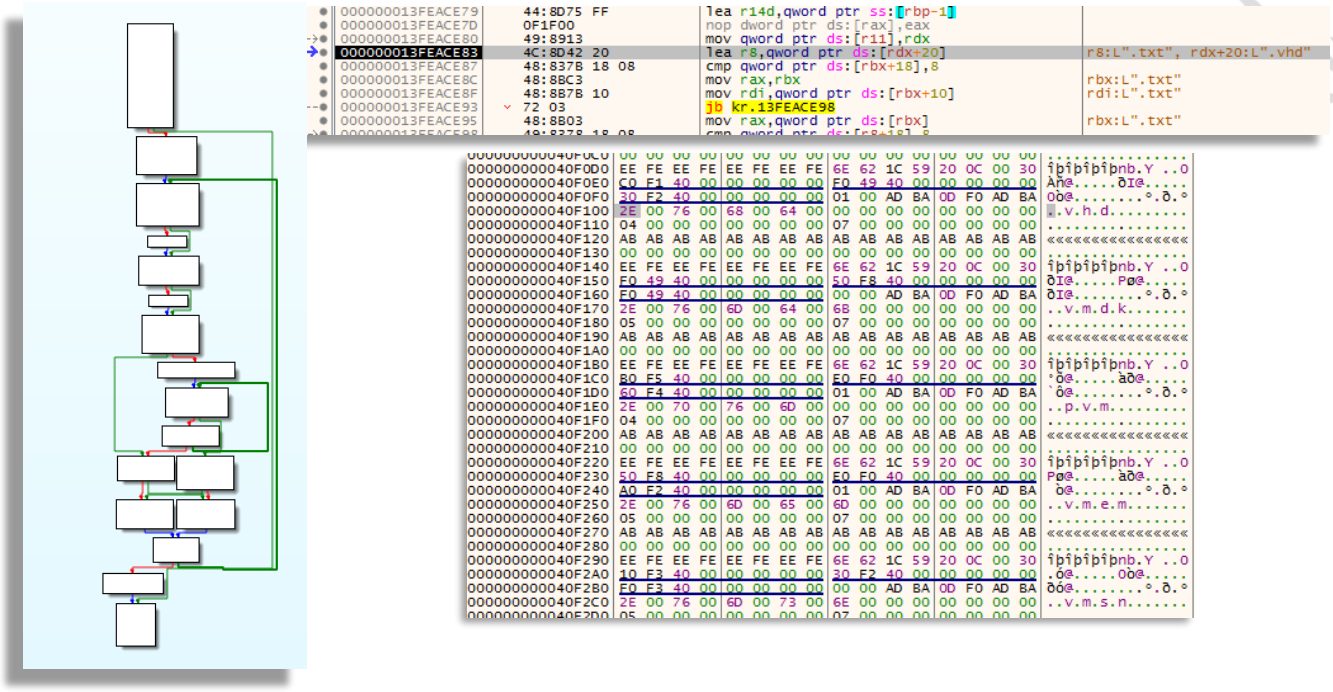
The important part of this phase is the extensive traversal it performs, as to simply write the ransom note, it validates entering the first disk, checking each folder, while simultaneously writing and validating. This work is very fast because it is using multi-threading, but in cases where ransomware does not use these techniques, these validations take enormous amounts of time in comparison, as in this case, it is doing everything at once.

In this way, it checks that it can enter the folder (remember it has a list of processes, but also of folders where it wants or does not want to enter)

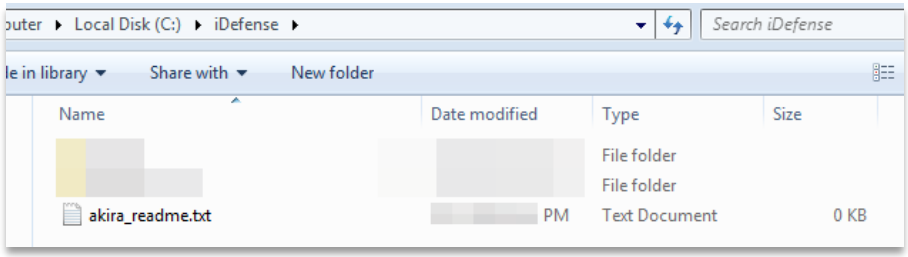
```
000000013FE926CA  33D2                  xor edx,edx
000000013FE926CC  41:B8 08010000        mov r8d,108
000000013FE926D2  48:8D8C24 F0000000    lea rcx,qword ptr ss:[rsp+F0]   [rsp+F0]:&L"C:\\iDefense\\MAP"
000000013FE926DA  E8 81550400           call kr.13FED7C60
000000013FE926DF  48:8D9424 D0000000    lea rdx,qword ptr ss:[rsp+D0]   [rsp+D0]:L"C:\\iDefense\\MAP\\akira_readme.txt"
000000013FE926E7  48:8D8C24 F0000000    lea rcx,qword ptr ss:[rsp+F0]   [rsp+F0]:&L"C:\\iDefense\\MAP"
000000013FE926EF  E8 1C0F0000           call kr.13FE93610
```

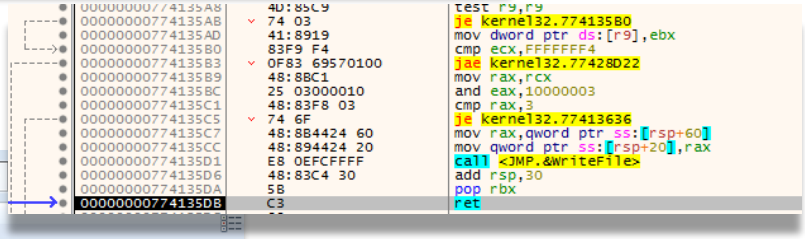
This sequence seems chaotic because, at the same time, the main function in which the writing of the Readme is found, checks with another thread if the file is suitable for encryption or not, so we will see in the same execution, one thread checking that it can enter a folder, while checking that the files can be affected later, plus the writing of the ransom note

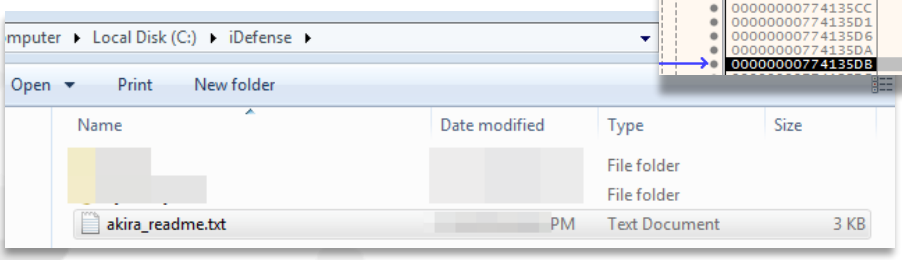
So, ultimately, we will see, on one hand, how it has loaded the ransom note, how it is comparing the file extensions with its internal list. To clarify, Akira not only has a list of extensions that it does not want to affect but also has another one that, depending on the size, it will want to encrypt. This is something that is not so usual, as it is more common to have a more boolean list, where if the extension is not on the list, then it affects it. But this malware goes a step further in its checking and has a long list of "accepted" extensions

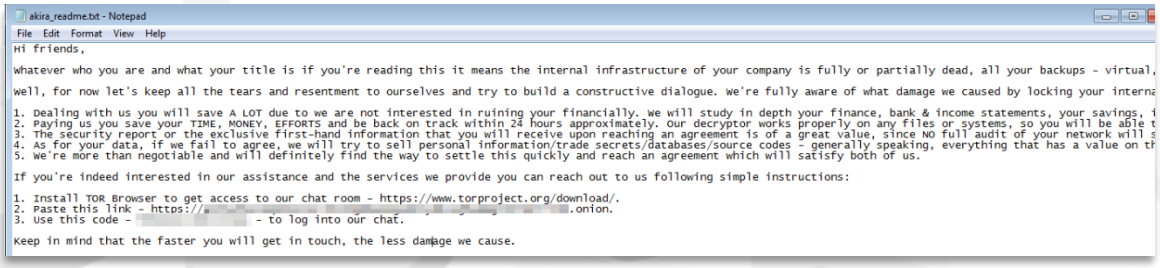


To conclude the topic of the ransom note, there is not much more to it. Once it has checked if it should access the path, since it already has the content loaded and the name of the ransom note file prepared (in the more recent versions of Akira it follows the pattern "Akira_readme.txt" and in the older versions "help-you.txt"), the behavior is similar. Again, it is only worth noting that in the paths it can traverse, it will leave a copy, not just in root paths or on the desktop



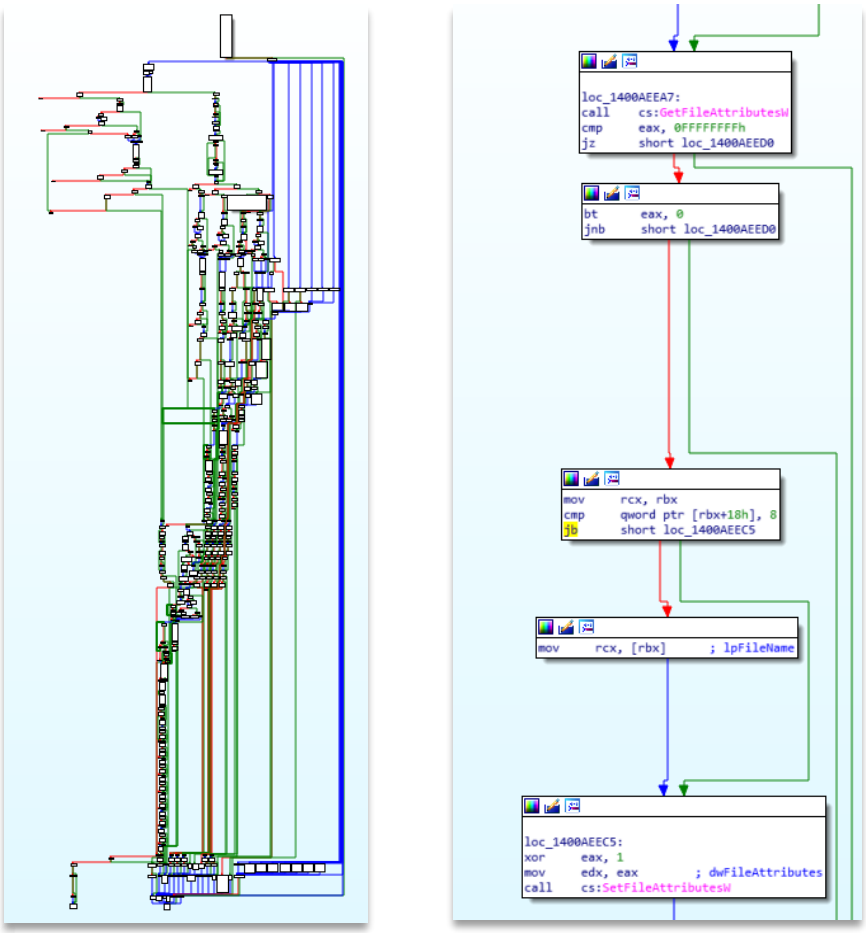






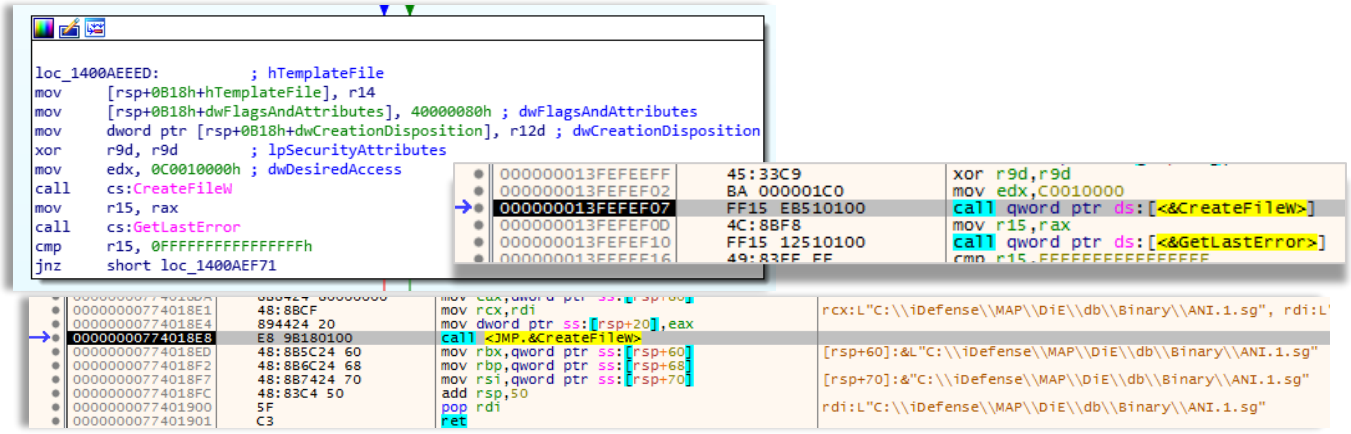
Continuing with the execution of multiple threads, it is quite peculiar to find Akira samples with a large number of functions, many of which are copies of each other or, in other words, there are functions that statically have a series of functions, but dynamically, they do not pass through them, or at least not completely. Instead, there exists another function with the same characteristics that is called from dynamic calls that are not statically traceable.

Continuing with the encryption, we reach the function where all of this is managed. It will be handling files, taking into account that there are several threads working, but there will be one or more traversing this entire function, obtaining the files, and working on them

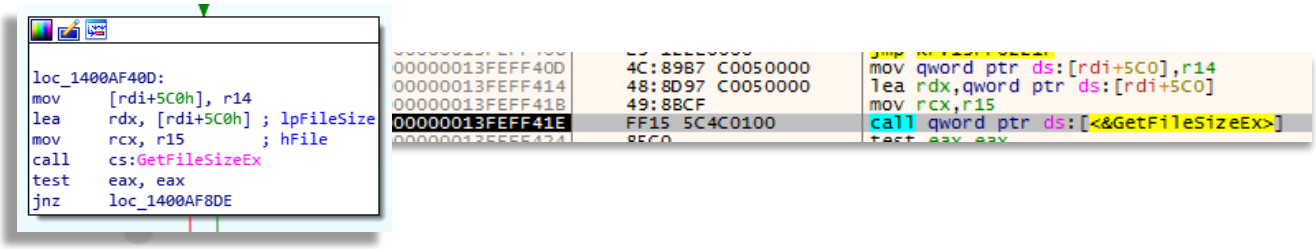



```
000000013FEFEE91  44:0FB66F 5D      movzx r13d,byte ptr ds:[rdi+5D]
000000013FEFEE96  48:8B5F 41        mov rbx,qword ptr ds:[rdi+41]   [rdi+41]:&L"C:\\iDefense\\MAP\\DiE\\db\\Binary\\ANI.1.sg"
000000013FEFEE9A  48:8BCB           mov rcx,rbx
000000013FEFEE9D  48:837B 18 08     cmp qword ptr ds:[rbx+18],8
000000013FEFEEA2  72 03             jb kr.13FEFEEA7
000000013FEFEEA4  48:8B0B           mov rcx,qword ptr ds:[rbx]
000000013FEFEEA7  FF15 53520100     call qword ptr ds:[<&GetFileAttributesW>]
000000013FEFEEAD  83F8 FF           cmp eax,FFFFFFFF
000000013FEFEEB0  74 1E             je kr.13FEFEED0
000000013FEFEEB2  0FBAE0 00         bt eax,0
000000013FEFEEB6  73 18             jae kr.13FEFEED0
000000013FEFEEB8  48:8BCB           mov rcx,rbx
000000013FEFEEBB  48:837B 18 08     cmp qword ptr ds:[rbx+18],8
000000013FEFEEC0  72 03             jb kr.13FEFEEC5
000000013FEFEEC2  48:8B0B           mov rcx,qword ptr ds:[rbx]
000000013FEFEEC5  83F0 01           xor eax,1
000000013FEFEEC8  8BD0              mov edx,eax
000000013FEFEECA  FF15 38520100     call qword ptr ds:[<&SetFileAttributesW>]
000000013FEFEED0  45:8BC6           mov r8d,r14d
000000013FEFEED3  41:BC 03000000    mov r12d,3
```

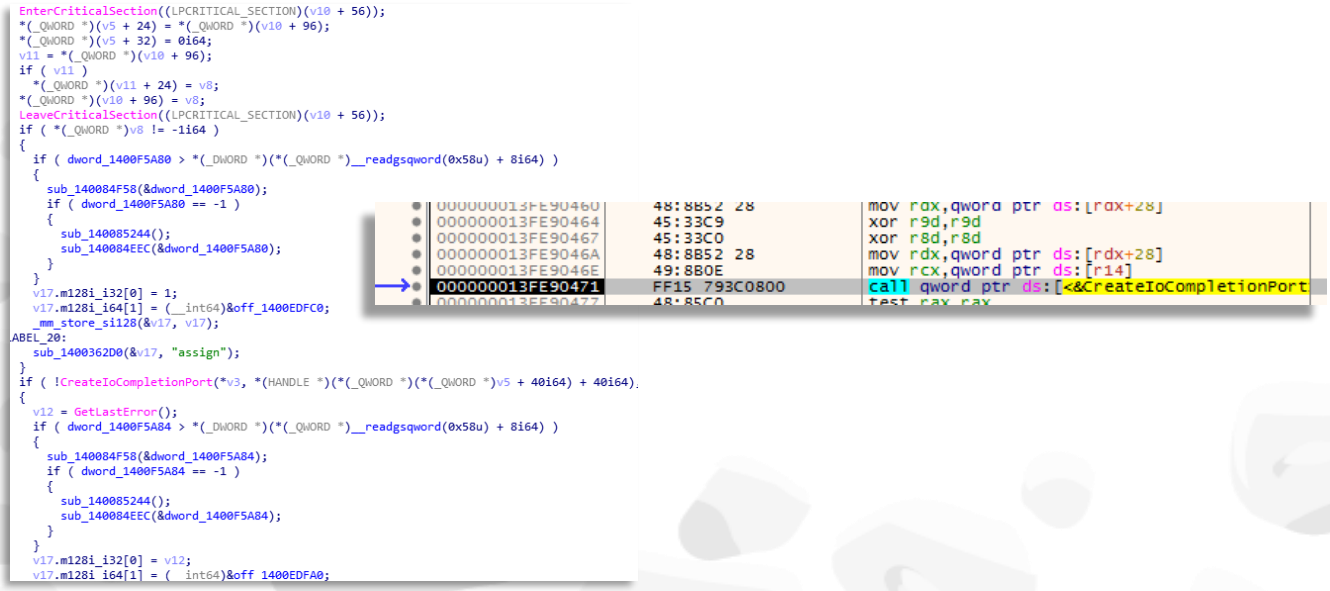
To work with the files, it performs a *CreateFile* and before doing anything, it is very important to check the size



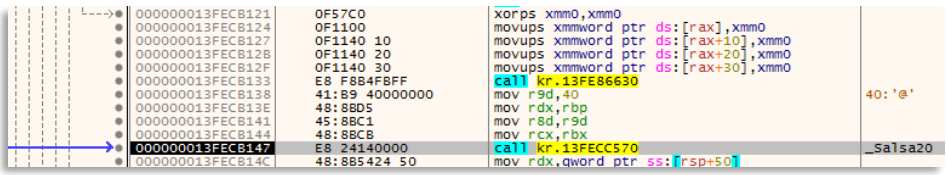
When we talk about Ransomware, we are talking about malware that will be affecting a large number of files simultaneously (in terms of those working with multi-threading) in a matter of seconds, achieving the full impact on the files in less than 2 or 3 minutes. This requires a great amount of processing that needs to be controlled. For this, in addition to working with threads, it is necessary to control the file sizes. In this way, if the file is larger than X megabytes, where in our case it should be less than 5 megabytes, the Ransomware may or may not affect the file or split it into more manageable chunks to be faster, all focused on efficiency



After this, it would lead us to a *CompletionIoPort* mentioned earlier, as a step before reaching the encryption



Subsequently, we reach the Salsa20 function (which, as we will see, is essentially the ChaCha20 algorithm; let's say Salsa20 is the predecessor, so there are functions that are really similar). In this first function, it extracts the keys based on the size




```
loc_14007B121:
xorps   xmm0, xmm0
movups  xmmword ptr [rax], xmm0
movups  xmmword ptr [rax+10h], xmm0
movups  xmmword ptr [rax+20h], xmm0
movups  xmmword ptr [rax+30h], xmm0
call    pfnAPC
mov     r9d, 40h
mov     rdx, rbp
mov     r8d, r9d
mov     rcx, rbx
call    _Salsa20
mov     rdx, [rsp+28h+arg_20]
mov     rcx, rbx
call    sub_14007C550
mov     rbx, [rsp+28h+arg_0]
xor     eax, eax
mov     byte ptr [rdi], 1
mov     rbp, [rsp+28h+arg_8]
add     rsp, 20h
pop     rdi
retn
```

```
char *v3; // r9
_DWORD *v4; // r10
_DWORD *v5; // rcx
__int64 result; // rax

v3 = Salsa20_ChaCha1;
a1[4] = *a2;
v4 = a1;
a1[5] = a2[1];
a1[6] = a2[2];
a1[7] = a2[3];
if ( a3 != 256 )
  v3 = Salsa20_ChaCha2;
v5 = a2 + 4;
if ( a3 != 256 )
  v5 = a2;
v4[8] = *v5;
v4[9] = v5[1];
v4[10] = v5[2];
v4[11] = v5[3];
*v4 = *(_DWORD *)v3;
v4[1] = *((_DWORD *)v3 + 1);
v4[2] = *((_DWORD *)v3 + 2);
result = *((unsigned int *)v3 + 3);
v4[3] = result;
return result;
```

```
Salsa20_ChaCha1 db 65h, 78h, 70h, 61h, 6Eh, 64h, 20h, 33h, 32h, 2Dh, 62h
                                        ; DATA XREF: _Salsa20+2↑o
                db 79h, 74h, 65h, 20h, 6Bh
                public Salsa20_ChaCha2
Salsa20_ChaCha2 db 65h, 78h, 70h, 61h, 6Eh, 64h, 20h, 31h, 36h, 2Dh, 62h
                                        ; DATA XREF: _Salsa20+28↑o
```

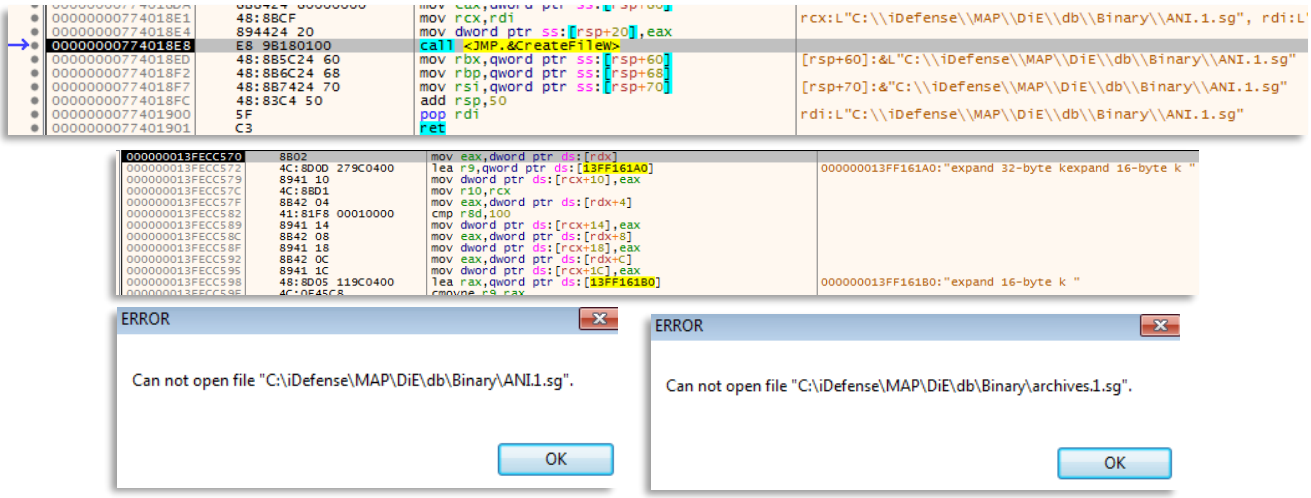
After this, we reach the most interesting function where the action is performed, as the renamed function *_Encrypt* will be the one through which the files that have been checked to be affected will pass

```
v23 = v15 ^ (unsigned __int64)sub_14007CBA0((__int64)a1, (v14 << 8) ^ ((unsigned __int64)v14 >> 24))
v24 = v16 ^ v23;
v31[24] = v23;
v31[25] = v16 ^ v23;
v25 = v17 ^ v16 ^ v23;
v31[26] = v25;
v31[27] = v25 ^ v14;
v26 = sub_14007CBA0((__int64)v31, ((v25 ^ v14) >> 24) ^ ((v25 ^ v14) << 8));
v27 = v31;
v28 = v23 ^ v26 ^ 0x2000000;
v31[2] = v17;
v29 = v25 ^ v24 ^ v28;
v31[28] = v28;
v31[31] = v29 ^ v25 ^ v14;
v31[29] = v24 ^ v28;
v31[30] = v29;
*v31 = v23;
v31[1] = v14;
v31[3] = v22;
v31[9] = v28;
v31[6] = v29 ^ v25 ^ v14;
v31[8] = v18;
v31[11] = v19;
v31[12] = v20;
v31[13] = v25 ^ v14;
v30 = 24i64;
v31[14] = v24;
v31[15] = v25;
v31[4] = v21;
v31[5] = v29;
v31[7] = v12;
v31[10] = v24 ^ v28;
*(_QWORD *)(v31 + 35) = 0i64;
*(_QWORD *)(v31 + 37) = 0i64;
do
{
  Encrypt(v27, 0);
  v27 = v31;
  --v30;
}
while ( v30 );
return 0i64;
}
```

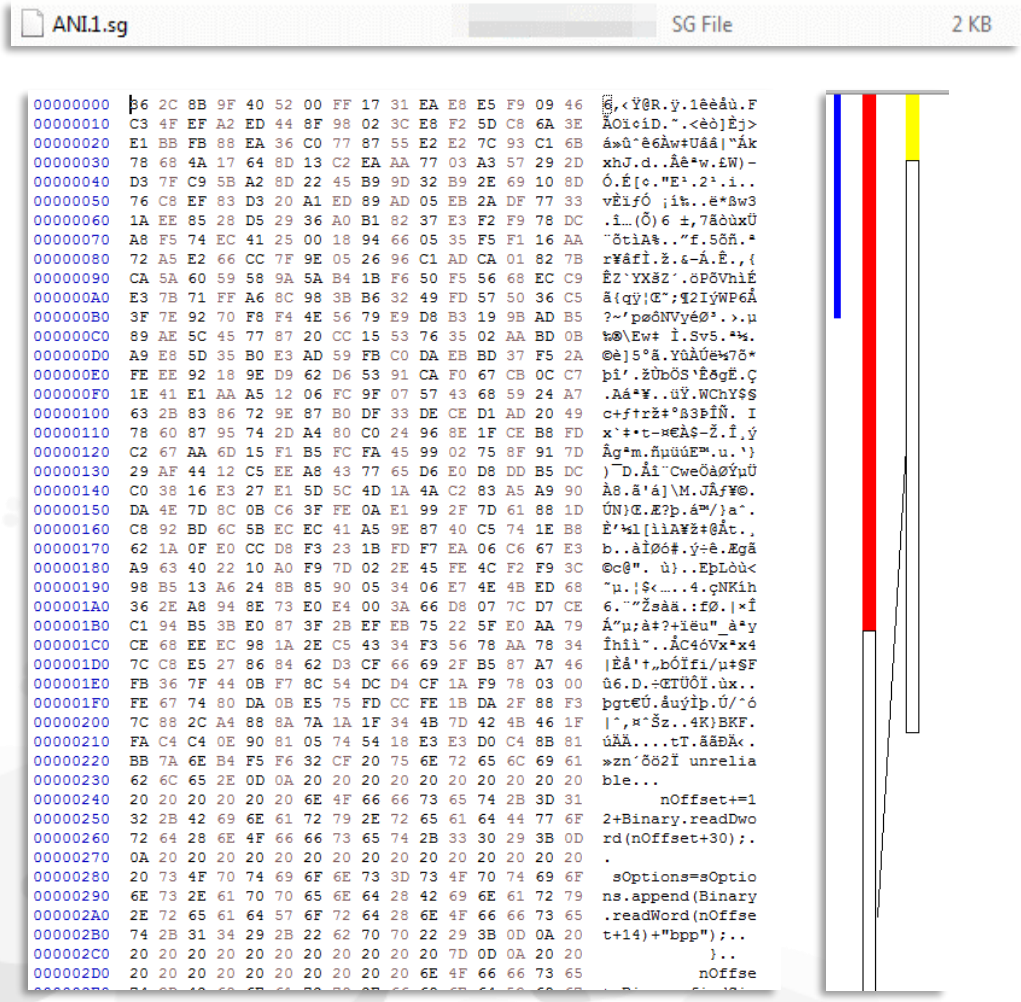
```
loc_14007C870:
xor     edx, edx
mov     rcx, r10
call    _Encrypt
mov     r10, [rsp+68h+var_38]
sub     rbx, 1
jnz     short loc_14007C870
```

```
if ( !v3 )
  v15 ^= v8 ^ v4[4] ^ (v2 + v4[5]);
v16 = v4[2];
v17 = v4[5];
if ( _bittest(&v16, 0x1Eu) )
  v18 = dword_1400EC6F0[v17 >> 24] ^ (v4[5] << 8);
else
  v18 = dword_1400ECAF0[v17 >> 24] ^ ((_DWORD)v17 << 8);
v19 = v16 < 0;
v20 = v4[13];
if ( v19 )
  v20 = dword_1400ECEF0[v13 >> 24] ^ ((_DWORD)v13 << 8);
v21 = v18 ^ v20 ^ v11 ^ v12;
if ( !v3 )
  v21 ^= *v4 ^ v4[35] ^ (v14 + v6);
v4[6] = v4[7];
v4[7] = v4[8];
v4[8] = v4[9];
v4[9] = v4[10];
v4[11] = v4[12];
v4[13] = v4[14];
v4[35] = v24;
v4[36] = v23;
*v4 = _mm_cvtsi128_si32(v10);
v4[37] = v25;
result = v26;
v4[38] = v26;
v4[1] = _mm_cvtsi128_si32(_mm_srli_si128(v10, 4));
v4[2] = _mm_cvtsi128_si32(_mm_srli_si128(v10, 8));
v4[3] = _mm_cvtsi128_si32(_mm_srli_si128(v10, 12));
v4[4] = v15;
v4[5] = v11;
v4[10] = v12;
v4[12] = v13;
v4[14] = v14;
v4[15] = v21;
return result;
}
```

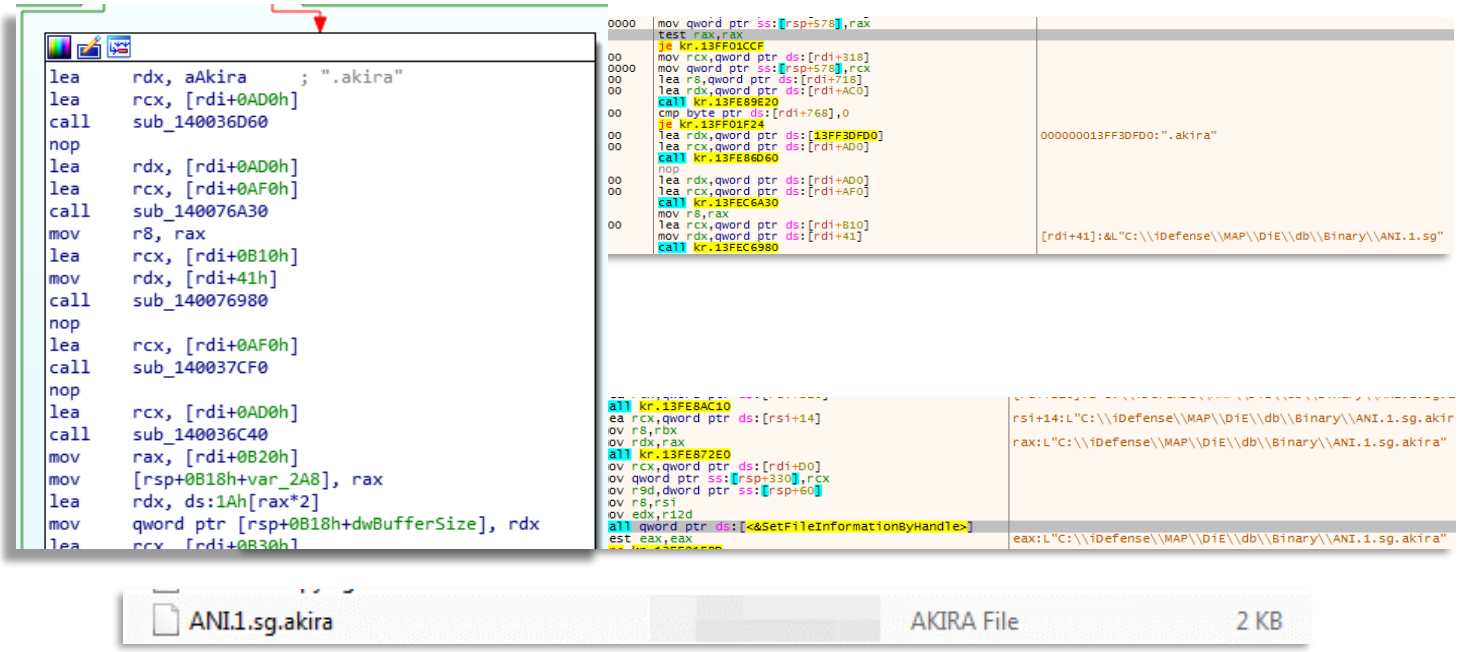
It is interesting to see how it works on several files at once, thus locking different files during the encryption process



Once affected, we can find that the order is usually encryption followed by changing the extension, something that in other Ransomware occurs the other way around. Here we can see one of them before changing the extension, already encrypted. As we see, it adds data at the top and bottom, maintaining part of the original content in the middle of the file



The modification of the extension is done in the same array we saw before, despite having previous moments where it loads the extension. It does everything in the same function and makes the change with *SetFileInformationByHandle*



After this, all the files will be affected, and you will either have to rely on backups if you have them or pay (which is not recommended)

In conclusion, Akira is another Ransomware that has burst strongly into the crime sector, and it is important to know how it works, not only at the sample level but also in terms of techniques to be able to protect ourselves properly. Although it has undergone changes over time, at least the samples, as we will see below, have not undergone significant changes, which is a good point for us.

4.4. Diffing

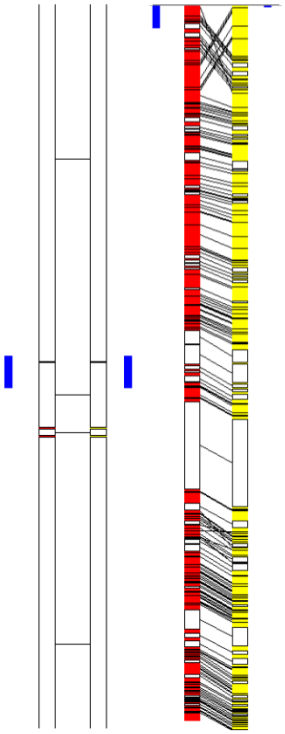
We have been discussing throughout the document how relatively little Akira has changed. This is because a large number of public samples have been analyzed from months ago to now to see how they have varied. The consequence of this has been that among similar samples there were changes, but very insignificant ones, and there was a notable gap between two types of samples that we will see below.

We can see a notable difference in bytes between one group of samples and another. These samples are the majority that have been analyzed and checked for functionality over the months, with those weighing more than 1 MB being the most recent samples, and versions between 540 and 580KB being the older ones, although there are exceptions

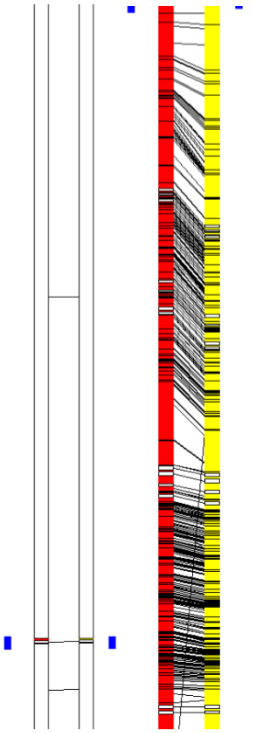
If we compare files with similar sizes of Akira, we see that there is indeed a difference between them, but it is not very large if we look at it from the point of view of files that are from the same group (~500 or ~1000) and they will differ a little more if it is a comparison between these two well-differentiated groups.

So, we performed three comparisons, between files of the same size and among them to determine how they differ from each other

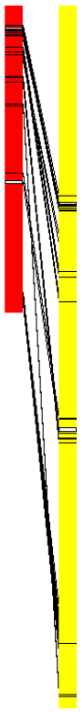
~500KB



~1000KB

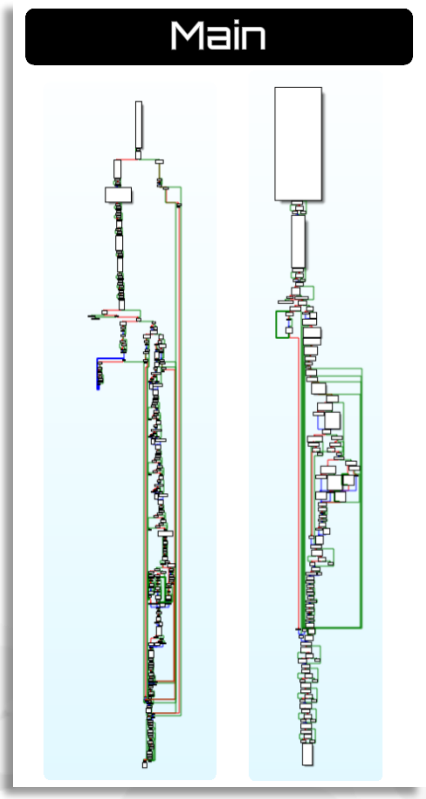


~500KB vs ~1000KB



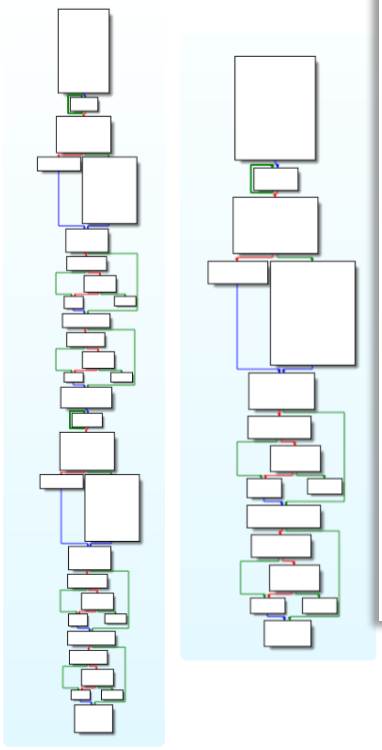
After this, more detailed checks are carried out because it is important to see if the existing versions are very different in functionality. We see that the main function is quite different, but at the functional level, apart from the creation of logs and the creation of the public key, it performs the same actions

Main

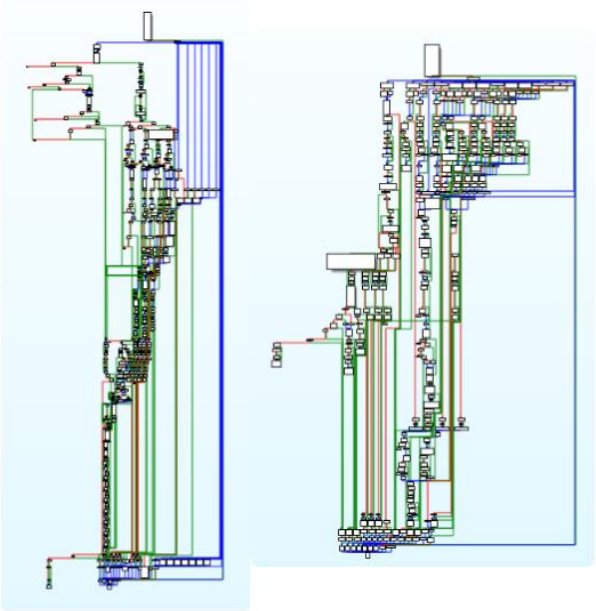


Looking at more specific functions, we see that in others, such as the one that parses the extension, where the processes are obtained to then compare them, the creation of threads, the writing of the extensions, and Readme files, the main functions where comparison occurs, traversal, and even though the functions that handle the encryption framework are similar, and a long etcetera (the names are approximate, the functions are usually part of something more complex or lead to other functions)

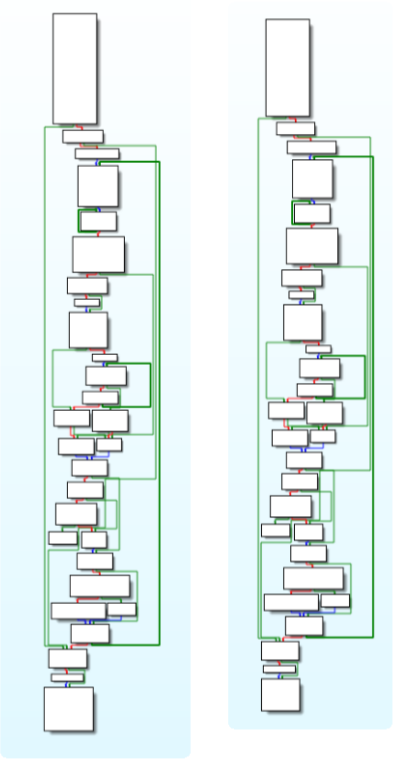
Extension



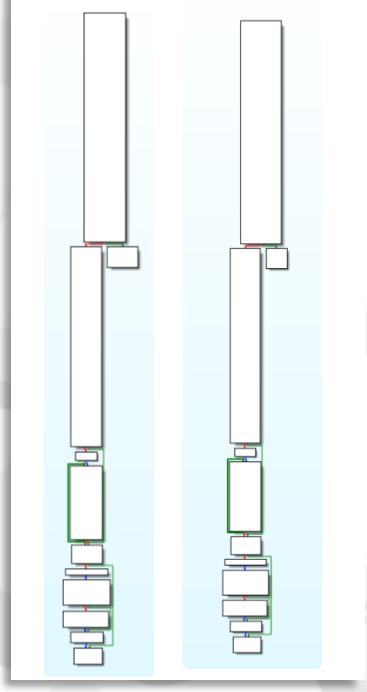
Prev. Encryption



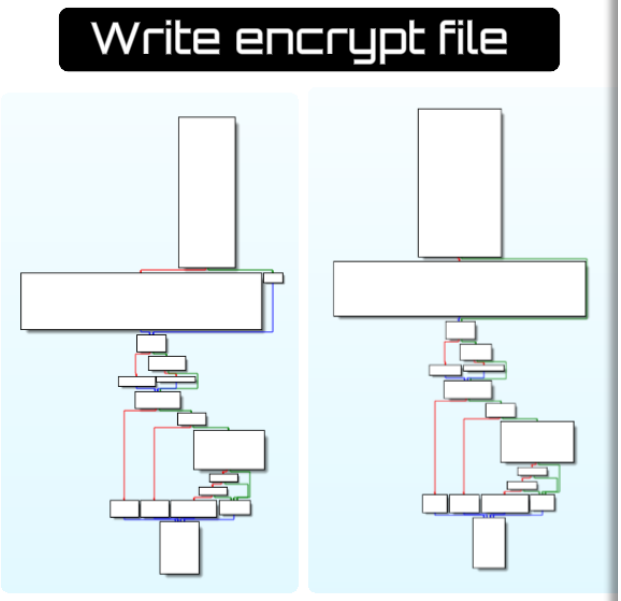
Get Process



Thread creation



Write encrypt file



Compare

We conclude then that, despite having undergone modifications over time, it is not a sample that has varied significantly. It is worth noting that during these variants, MegaZord has also been found, which is a version of Akira, very similar, that practically works the same, although it contains some changes, in essence it is a copy of it, but it is relevant to name it. The less important functions have undergone changes, but those on which an important part of the sample's functionality pivots are quite similar, even identical in operation and form, which opens up a window of possibilities for understanding it and, therefore, for its detection.

4.5. Exclusions

- Extensions avoided

```
.exe
.dll
.lnk
.sys
.msi
```

- Folders avoided

```
tmp
winnt
temp
thumb
Recycle.Bin
RECYCLE.BIN
System Volume Information
Boot
Windows
Trend Micro
ProgramData
```

It has other features based on size, I collected a large number of extensions which did not care about the size, the list is very long, but I think I did not get them all, I do not want to put incomplete data, but all I have match me with Trend Micro, so in this list you have them great

5. INTELLIGENCE

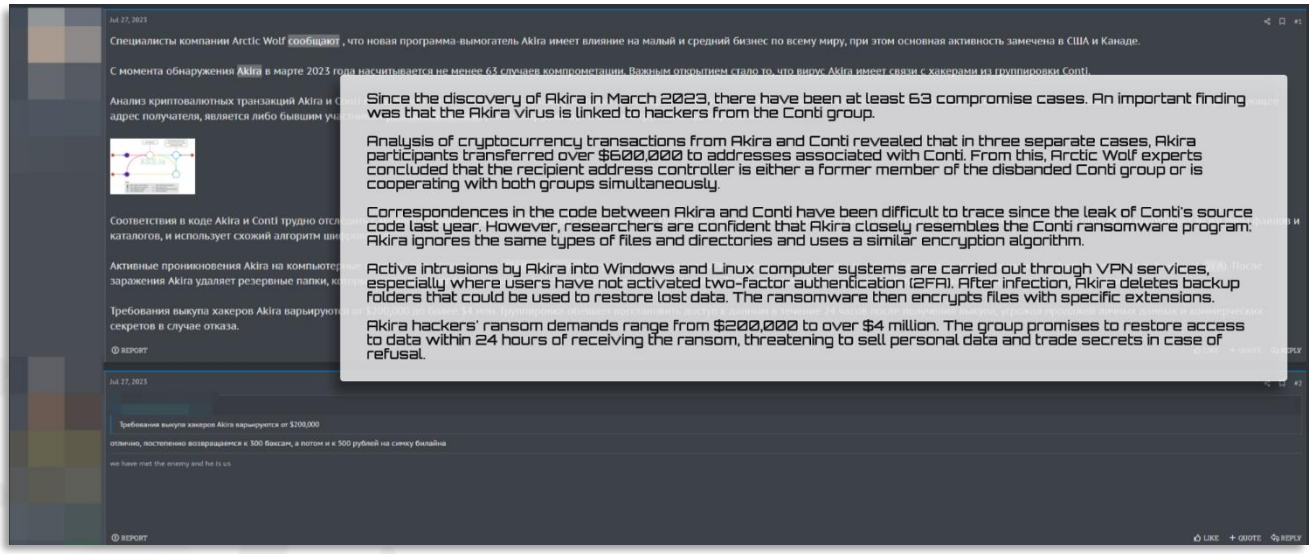
With the information obtained from the analysis of all the samples found, bearing in mind that those shown throughout this document are only selected ones, all the indicators of engagement that could be used in order to discover additional infrastructures that could have been used by the Akira group were collected. In this particular case, it is especially difficult, because ransomware samples are usually tools for impact, among many other things, which means that public information related to IOCs belonging to the network is much smaller, but we nevertheless retrieved as much as possible for this section.

And, of course, it is important in the field of intelligence to maintain good contacts that can give insight or be a point of support to collect or expand certain information, thanks to Michael Koczwara (@MichalKoczwara) and Simone Kraus (@simonekrausora1) we were able to give more depth to the investigations that we will see below.

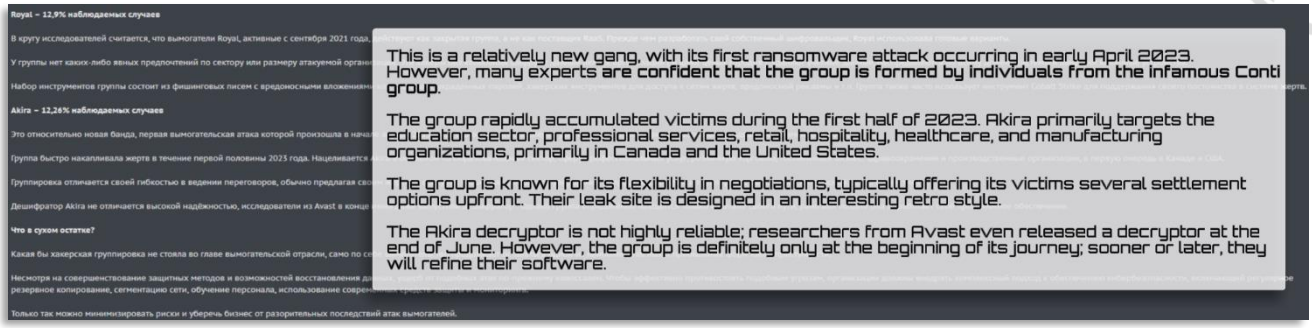
During this section we will also take a look at underground sites for possible interactions of the TA.

Obviously, after much discussion about Anime/Manga, information about the Stealer with the same name, and other topics, we found interesting data that reinforces previous information.

People are discussing topics related to reports from ArticWolf, where the possibility of the student (Akira) surpassing the master (Conti) is raised, and where statistics for 2023 about the famous group are discussed



On another note, there's confirmation that former members of the Conti group (remember it has been inactive for years) are now part of Akira. This is completely logical considering the similarity in operations and the malware they are using



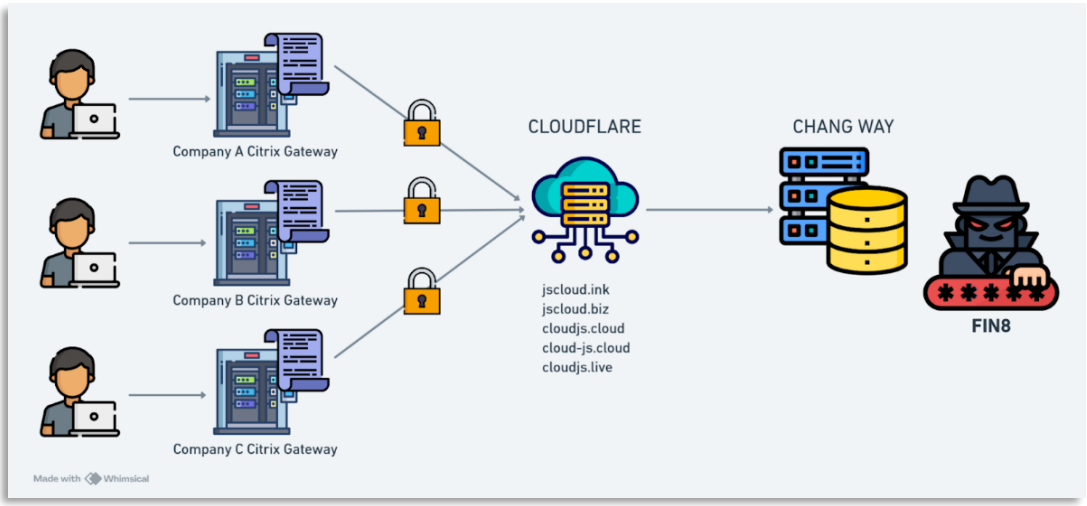
On another note, Akira, as we mentioned earlier, is strongly believed to comprise ex-Conti members. They have previously been linked to ransomware groups such as Snatch and BlackByte, and reports suggest that they may also **collaborate with Initial Access Brokers (IABs) Bassterlord and FIN12.** Akira also operates infrastructure on hosting providers **linked to FIN8 and LockBit**

As seen in the attack methods, they compromises their victims through VPNs, via either brute-forcing single factor credentials or exploiting VPN vulnerabilities. They also purchase credentials from dark web marketplaces and attain access via the aforementioned IABs, and make extensive usage of **Remote Administration tools such as TeamViewer, MobaXterm, Ngrok, Cloudflare Tunnel, AnyDesk and RustDesk. They also utilise FileZilla, WinRAR, WinSCP, and RClone to exfiltrate their data** to either Mega cloud storage or their own infrastructure which, as we have said, being legitimate tools and services, make it very difficult to track their infrastructure

Collaborating with other groups and gaining access via VPNs, means that Akira only needs to stand up servers on VPS providers running common protocols and legitimate tools to get the job done. This leaves minimal footprint in which to track the group as we see little to no reporting of Akira using malware or post-exploitation frameworks

Akira has been reportedly operating from **M247,ITL LLC, Stark Industries, GoDaddy, Reliablesite, Starcrecium, Media Land, Flyservers, xhost Internet Solutions** and **Chang Way Technologies Co**. Limited, to brute force and access VPNs and exfiltrate their data. A number of these hosting providers are regularly associated with criminal activity, taking inaction against shutdown requests. Of these, Starcrecium, Media Land and Chang Way operate ~7000 IP addresses almost exclusively out of Russia. Stark Industries now operates out of over 35 countries and has consistently been utilised for malicious purposes. It is no surprise that these providers have been leveraged by Akira

Chang Way stood out due to previous findings into other groups such as FIN8, Lockbit and BlackByte (the latter being a known connection to Akira). Let's look at past activities connected to Chang Way



In October 2023, Citrix released CVE-2023–4966 and only a month later, CISA issued an advisory that LockBit affiliates were exploiting it in widespread attacks, notably Boeing. A number of the infrastructure utilised belonged to Chang Way, for example, **Citrix credentials** were posted to **62.233.50[.]25** and **data exfiltration via FTP was sent to 193.201.9[.]224**. The former IP was also observed by Rapid7 in brute force attacks against Cisco VPNs by LockBit and Akira, targeting CVE-2023-20269 starting from March 2023. These attacks followed the February post on dark web forums by the Initial Access Broker "Bassterlord" on how to exploit Cisco VPNs. During both of these campaigns, Akira and LockBit intrusions displayed many of the same TTPs such as the utilisation of Remote Administration tools like AnyDesk and TeamViewer.

In a previous blog, **we analysed certificates on RDP, FTP and AnyDesk ports linked to LockBit exploitation of CVE-2023-4966**. We found evidence linking "Bassterlord" not only to LockBit but also their previous connection with Avaddon. Analysis of the FTP port certificates showed a Subject Common Name of "PASCAL" (Notably on Chang Way). Knowing that "Bassterlord" and LockBit had used this infrastructure for CVE-2023-20269 alongside Akira it is therefore possible that some of this infrastructure may also be utilised by the IAB in other VPN exploitation campaigns linked to Akira and Chang Way

We can expand our previous searches to include other famous scientists such as "Bohr", "Tesla" and "Hawking", to identify even more servers running ports for AnyDesk and FileZilla (Hosted on SELECTEL, Russia) although these numbers are small, around 11 servers.

Rapid7 reported that traffic originated from **Chang Way Technologies Co. Limited, Flyservers S.A., Xhost Internet Solutions Lp, NForce Entertainment B.V. and VDSina Hosting** during attacks on CVE-2023-20269. With a large number of Windows clients named "WIN-R84DEUE96RB" associated with the LockBit and Akira infrastructure.

There are currently ~1000 servers on the internet with this client name, hosted on VDSINA, Hosting technology LTD, Cloud assets LLC and SERVERS TECH FZCO, nearly all of which are hosted in Russia:

In a report by Intrinsic, also highlighted a number of client names utilised by Akira targeting CVE-2023-20269:

- DESKTOP-3GCJKGQ
- WIN-KFUMVU06ESH
- WIN-OX9CQTDSEIK
- WIN-MV7S8OJTOIK
- HOST14872171171
- DESKTOP-KT76603

However, our research did not uncover any active servers matching these names at the time of analysis

That was not all of the VPN exploitation activity linked to Chang Way. FIN8 was also targeting and exploiting Citrix VPNs via **CVE-2023-3519** to deploy AlphV/BlackCat ransomware. During our previous research, we identified a Chang Way IP address as the destination node for this campaign, which all Cloudflare proxied domains ultimately pointed to within the injected code

Additionally, we also identified an open server on Chang Way containing over **1 TeraByte of data comprising 37 directories**, with 19 named after organisations posted to the data leak site of the ransomware group **BlackByte** between January and September 2023. In July 2023, a Recorded Future article reported that Yamaha appeared on the DLS **for BlackByte and Akira ransomware**, suggesting that operators from both groups are working together. In August 2023, researchers identified an open-directory of an Akira operator also exploiting Fortinet VPNs and traced activity to a Snatch victim; however, it's unknown whether this directory was hosted in Chang Way.

Finally, a recent investigation uncovered another open directory on the hosting provider that pointed the finger directly at Akira and its operators. Whilst we were unable to analyse the contents, we found historical records that contained filenames of tools known to be utilised by Akira and Conti, such as PCHunter.

It's clear that Chang Way is employed by multiple ransomware groups to conduct their activity, including 3 different campaigns exploiting VPNs, one of which involved Akira ransomware. Open directories also show evidence of connections between Akira and BlackByte.

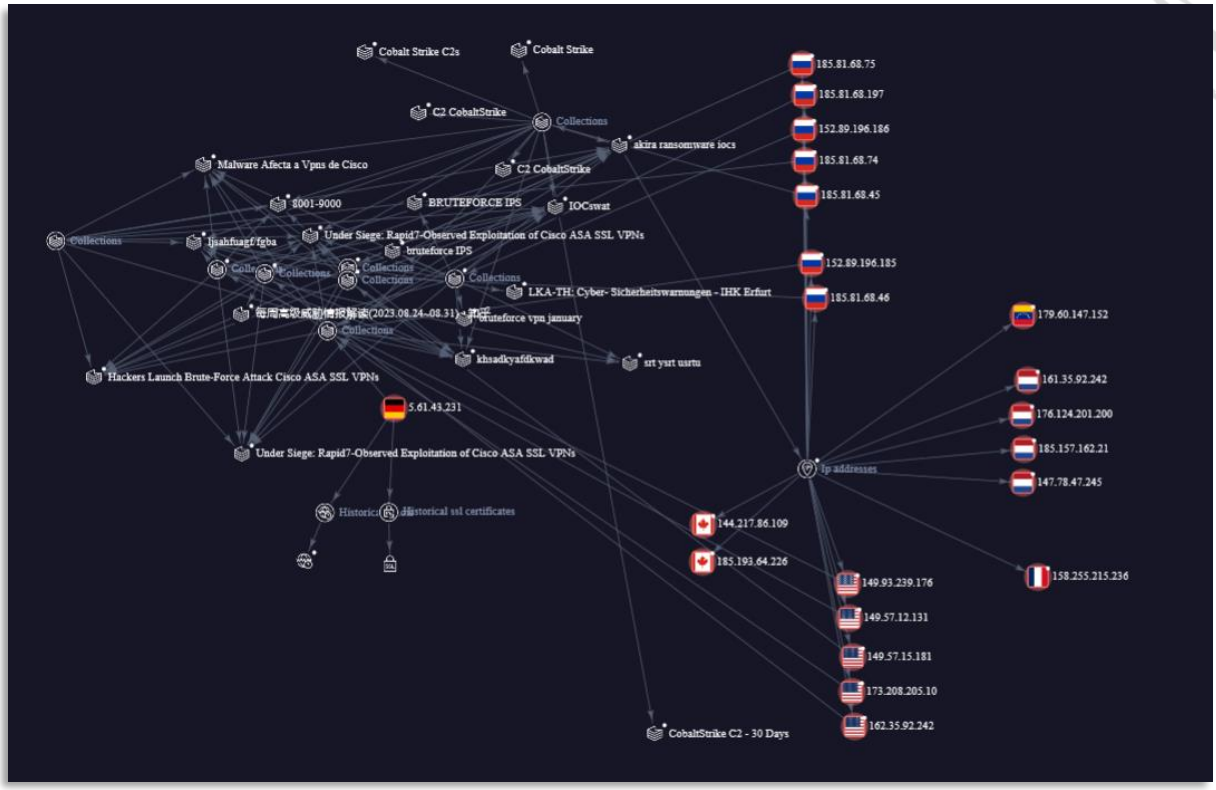
We identified two VirusTotal graphs from 30/01/2024, Linking Akira to Cobalt Strike. This infrastructure formed the basis of our analysis into Cobalt Strike for this report.

We have included IP addresses in this research that we have been able to verify are Cobalt Strike servers through associated file relations and historical community comments in VirusTotal

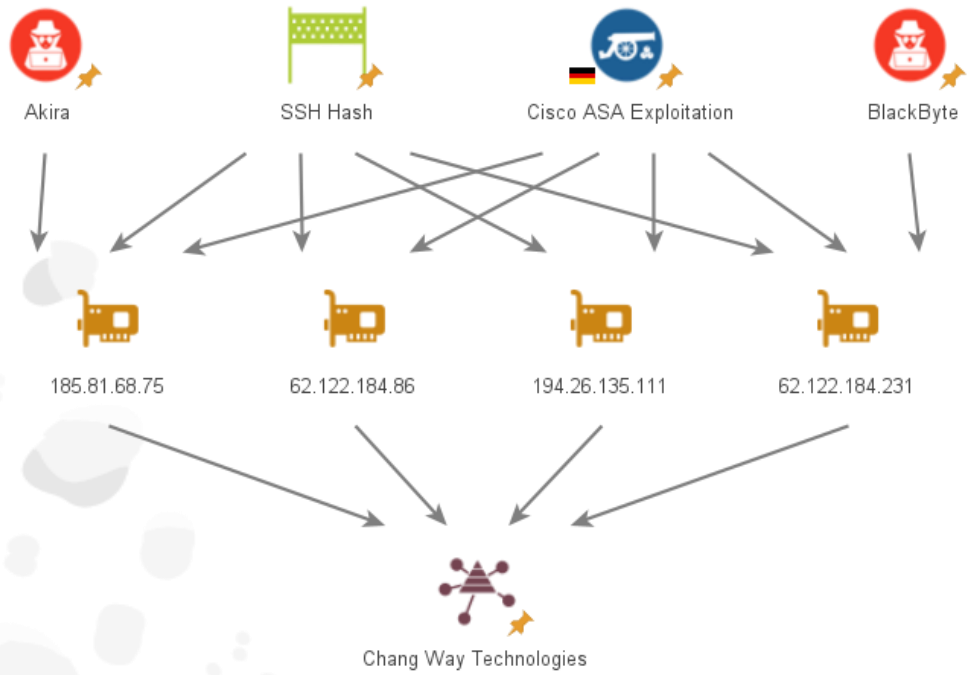
Our analysis of this infrastructure uncovered nearly 60 servers hosted in the US, mostly on LeaseWeb and Nexeon. Whilst Akira have connections to Conti, this cluster was more closely aligned to Black Basta as another ex-Conti group (We identified IP addresses linked to FIN12), historical Conti IOCs from CISa as well as Black Basta. Interestingly, these IP addresses appear highly active and remain so despite a large period of time. We believe this may be due to firewall rules implemented on the servers, limited researchers abilities to uncover and track them. Our assessment is that this cluster is not linked to Akira.

However, we analysed the second graph which lists numerous Akira IOCs relating to Rapid7's report. This graph contained a single Cobalt Strike C2 server however, it was too old to use in our analysis. This single IP was the only Cobalt Strike server we could conclusively link to Akira, confirming that they likely rely almost solely on open-source tools for command and control and persistence

During this analysis, we did identify another connection between Akira and BlackByte

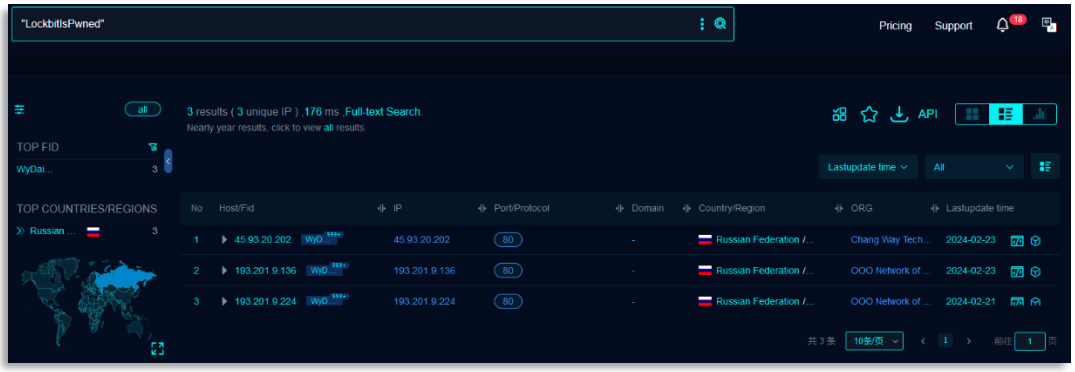


We identified **185.81.68[.]75**, linked to Cisco VPN exploitation, published. This IP address was linked to BlackByte threat actors and a number of other IP addresses that have been active in Cisco VPN exploitation. These IP's all shared the same SSH banner hash, indicating that they were all operated by the same operator of Akira and BlackByte



Chang Way and Selectel infrastructure is certainly drawing the eyes of law enforcement, at least 3 servers were seized by LE as they seized control of Chang Way and Selectel servers when they took down LockBit, with **193.201.9[.]224** used by LockBit to exploit Citrix VPNs.

As Akira draws more attention, we could well see similar interventions across these hosting providers in the future

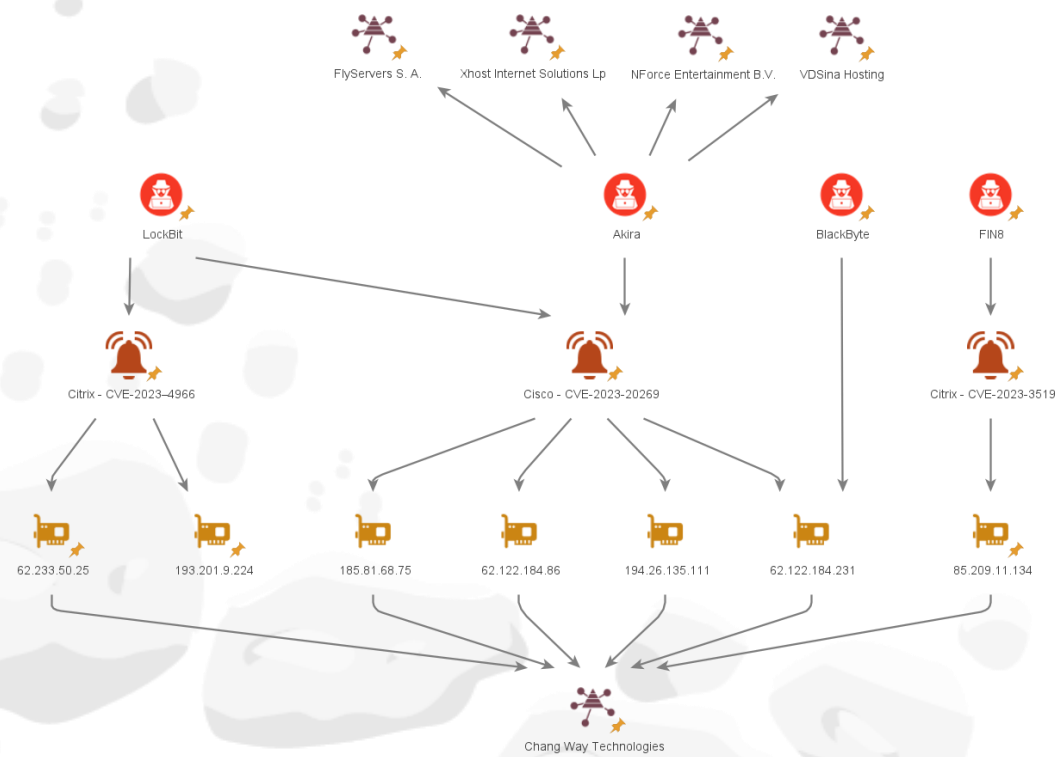


As a group formed of ex-Conti members, our research strongly suggests that the group collaborates with a number of threat groups such as Snatch, LockBit and BlackByte.

Chang Way is a hosting provider that plays a key role in the VPN exploitation ecosystem, being deployed by Akira, LockBit and FIN8 to compromise victims Cisco, Citrix and Fortinet gateways.

Akira focuses on this exploitation and collaboration with other prominent groups, choosing only to deploy open-source or legitimate tools to maintain that persistence. These choices have led to an effective and evasive threat group that continues to compromise victims globally.

To begin our defence against this threat, we should consider removing any ability of these groups to utilise Chang Way and any providers that lease infrastructure from them. We await to see whether future law enforcement action occurs against Akira and its hosting provider ecosystem as they continue to plague victims worldwide



6. DETECTION OPPORTUNITIES

Some terms to keep in mind, taking into account both some of the techniques used by the group and the samples seen and analysed during the paper for the purpose of telemetry detection could be the following:

```
() -> Type of event        * -> Wildcard      {} -> Type of data/Clarification

                           | -> OR (Yes, it is necessary)
```

- **[TA0002][T1059] Execution via commandline of the sample based on AKIRA parameter**

Local or remote execution (T1021 could also fit) of the Akira sample

```
(Process) powershell.exe | cmd.exe > (Command) \-\-encryption\_path|\-p|\-\-share\_file|\-s|\-\-localonly|\-l|\-\-encryption\_percent|\-n

(Process) powershell.exe | cmd.exe > (Command) (\-\-encryption\_path|\-p|\-\-share\_file|\-s|\-\-localonly|\-l|\-\-encryption\_percent|\-n).*\=\\\\\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}.*
```

- **[TA0009][T1074] Log control creation by Akira**

Creation of a control log by Akira

```
(File-Write) [Ll]og\-\d{2}\-\d{2}\-\d{4}\-\d{2}\-\d{2}\-\d{2}\.txt
```

- **[TA0040][T1490] Delete shadows using WMI**

Akira uses powershell to run a WMI command and delete shadow copies

```
(Process) powershell.exe > (Command) powershell.exe -Command "Get-WmiObject Win32_Shadowcopy | Remove-WmiObject"
```

- **[TA0040][T1486] Write readme file by Akira**

Creation of the ransomware file by Akira

```
(File-Write) Akira_Readme.txt | help-you.txt
```

- **[TA0006][T1136] Account creation prior to attack**

Accounts are created for better management of targeted infrastructure prior to impacts

```
(Process) cmd.exe > (Command)
cmd\.exe\s+\/[qQ]\s+\/[cC]\s+net\s+user\s+\/(dom|domain|add)\s+\d\>.*\\\\\d{1,3}\.\d{
1,3}\.\d{1,3}\.\d{1,3}
```

- **[TA0006][T1003] Dumping credentials via LSASS, SAM & NTDS**

Credentials are dumped in different ways to obtain more users

```
(Command) cmd*/c*comsvcs.dll, MiniDump*lsass*full

(Command) cmd*/c*-c -i*\NTDS\*-o*

(Command) cmd*/c*-c -i*\SYSTEM*-o*

(Command) ntdsutil*ac i ntds*ifm*createfull*q q
```

- **[TA0007][T1087] AD account discovery**

Akira queries to obtain the maximum knowledge of the Active Directory and hence the infrastructure

```
(Command) (Get-ADComputer|Get-AdUser)\s+\-Filter.*\-Prop.*Select-Object.*
```

- **[TA0005][T1562] Modify FW and disable defenses**

Firewall is modified as well as native security elements are disabled to avoid complications in execution or Exfiltration

```
(Command) netsh advfirewall firewall add rule
name=*dir=*protocol=TCP*localport=*action=allow

(Command) Set-MpPreference -DisableRealtimeMonitoring $true
```

- **Control over the tools used by Akira**

Akira uses in all its killchain a large number of tools, it is necessary to have them under control (knowing the parameters they use, internal names, etc.) The following are the tools they has used

```
Mimikatz | LaZagne | AnyDesk | Radmin | RustDesk | PCHunter | AdFind | PowerTool |
WinSCP | Rclone | FileZilla | SharpHound | MASSCAN | AdvancedIPScanner

e.g. (Tool-Name) Mimikatz > (Parameters)
(lsadump::|sekurlsa::|sid::|token::|dpapi::|vault::|crypto::|misc::|kerberos::|privi
lege::)
```

- **Yara**

This Yara is a bit generic as I don't like to publish super strict rules so that we don't get caught in the detection, sorry for that

```
rule TA_Ransomware_Akira
{
        meta:
                description = "Akira: The old-new style crime"
                category = "Ransomware"
                author = "vc0rexor"
                reference = ""
                date = "2024-06-01"

        strings:
                $a1 = "expand 32-byte" wide ascii nocase
                $a2 = "akira" ascii nocase
                $a3 = "onion" ascii nocase
                $a4 = "TOR browser" fullword ascii nocase
                $a5 = "--encryption_path" wide ascii nocase
                $a6 = "--encryption_percent" wide ascii nocase
                $a7 = "CreateThread" fullword ascii nocase
                $a8 = "CreateIoCompletionPort" fullword ascii nocase
                $a9 = "AcquireSRWLockExclusive" fullword ascii nocase
                $a10 = "GetCurrentThreadId" fullword ascii nocase
                $a11 = "GetLogicalDriveStrings" fullword ascii nocase
                $a12 = "GetQueuedCompletionStatus" fullword ascii nocase
                $a13 = "encrypt" ascii nocase
                $a14 = "thread pool" fullword ascii nocase
                $a15 = "failed" wide ascii nocase
                $a16 = "System Volume Information" fullword ascii nocase
                $a17 = "Paths Finded" fullword ascii nocase
                $b1 = { 0f 11 45 ?? 0f 57 c9 f3 0f 7f 4d ?? 4c 63 c0 33 d2 48 8d 4d
?? e8 ?? ?? fe ff 48 8d 4d ?? 48 83 7d ?? 08 48 0f 43 4d ?? 4c 8d 45 ?? 48 83 7d ?? 10
4c 0f 43 45 ?? 8b 45 ?? 89 44 24 28 48 89 4c 24 20 44 8b 4d ?? 33 d2 33 c9 ff 15 ?? ??
?? 00 0f 10 45 ?? 0f 11 45 ?? 0f 10 4d ?? 0f 11 4d ?? 66 0f 6f 05 ?? ?? ?? 00 f3 0f 7f
45 ?? 66 89 ?? ?? }
                $b2 = { 8b c7 0f 57 c0 0f 11 44 24 ?? 4c 89 74 24 ?? 4d 8b c7 4c 89
74 24 ?? 48 8d 0c 40 48 8b ?? 24 ?? }
                $b3 = { 48 8d ?? 27 48 83 ?? e0 48 89 ?? f8 48 89 ?? ?? ?? 8d ?? 00
20 00 00 ?? 89 ?? ?? 33 d2 41 b8 00 20 00 00 ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? b9
04 01 00 00 ff 15 ?? ?? ?? 00 ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? 00 00 00 ?? ??
?? ?? ?? 8d ?? ?? 66 66 66 0f 1f 84 00 00 00 00 00 }
                $b4 = { 0f b7 90 d2 03 00 00 48 8b 84 d0 d8 03 00 00 0f b7 90 d2 03
00 00 48 8b 84 d0 d8 03 00 00 0f b7 90 d2 03 00 00 48 8b 84 d0 d8 03 00 00 0f b7 90 d2
03 00 00 48 8b 84 d0 d8 03 00 00 0f b7 90 d2 03 00 00 48 8b 84 d0 d8 03 00 00 0f b7 90
d2 03 00 00 48 8b 84 d0 d8 03 00 00 0f b7 90 d2 03 00 00 48 8b 84 d0 d8 03 00 00 0f b7
90 d2 03 00 00 48 8b 84 d0 d8 03 00 00 48 83 c1 f8 }
                $b5 = { 48 8b 4e ?? e8 ?? ?? ?? 00 48 8d 7c 24 20 48 89 07 48 89 f9
e8 ?? ?? ?? ?? 48 8b 46 28 48 89 47 10 0f 10 46 18 0f 29 07 48 8b 0e e8 ?? ?? ?? 00 48
8d 7c 24 20 48 89 f9 e8 ?? ?? ?? ?? 48 8b 5e 08 48 8d 4b 18 e8 ?? ?? ?? ?? 48 c7 43 18
01 00 00 00 48 83 63 20 00 48 8b 46 08 48 89 07 48 8d 4c 24 20 e8 ?? ?? ?? ?? 90 48 83
c4 40 5b 5f 5e }

        condition:
                filesize > 500KB
                and filesize < 1100KB
                and (8 of ($a*) and 2 of ($b*))
                and uint16(0) == 0x5a4d
}
```

7. MITRE TTP

Tactics:

TA0001: Initial Access
TA0002: Execution
TA0003: Persistence
TA0005: Defense Evasion
TA0006: Credential Access
TA0007: Discovery
TA0008: Lateral Movement
TA0009: Collection
TA0010: Exfiltration
TA0011: Command and Control
TA0040: Impact

Techniques:

T1566.001: Spearphishing Attachment
T1190: Exploit Public-Facing Application
T1105: Ingress Tool Transfer
T1059.001: Command and Scripting Interpreter: PowerShell
T1059.005: Command and Scripting Interpreter: Visual Basic
T1136.001: Create Account: Local Account
T1547: Boot or Logon Autostart Execution
T1547.001: Registry Run Keys / Startup Folder
T1562: Impair Defenses
T1562.001: Disable or Modify Tools
T1562.004: Disable or Modify System Firewall
T1078: Valid Accounts
T1078.003: Valid Accounts: Local Accounts
T1003: OS Credential Dumping
T1003.001: LSASS Memory
T1016: System Network Configuration Discovery
T1046: Network Service Scanning
T1135: Network Share Discovery
T1082: System Information Discovery
T1012: Query Registry
T1083: File and Directory Discovery
T1021: Remote Services
T1021.001: Remote Services: Remote Desktop Protocol
T1021.002: Remote Services: SMB/Windows Admin Shares
T1071.001: Application Layer Protocol: Web Protocols
T1105: Ingress Tool Transfer
T1560.001: Archive Collected Data: Archive via Utility
T1048: Exfiltration Over Alternative Protocol
T1048.002: Exfiltration Over Asymmetric Encrypted Non-C2 Protocol
T1090.004: Proxy: External Proxy
T1490: Inhibit System Recovery
T1486: Data Encrypted for Impact

8. IOC

2CDA932F5A9DAFB0A328D0F9788BD89C
64F8E1B825887AFE3130AF4BF4611C21
A18D79E94229FDF02EF091CF974ED546
3F63951399F8CD578E2A6FAED2C9C0F0
4EDC0EFE1FD24F4F9EA234B83FCAEB6A
C2CBBD6E392A453A47DA69D086756E71
9F801240AF1124B66DEFCD4B4AE63F2A
FD380DB23531BB7BB610A7B32FC2A6D5
BD046164DAF3C30E265D4F9C6647F630
7ca94d84f4a02fb1f608818c1c3ab62d
11a6a4bfa63286feaeaf2c231ce769c3
Adf426e30f8a3383c6696d2f142907d3
F9a3a00b8772103ca109662b32d01934
495afbc7ebae07d50c529c1bd5889f54
491f619c358382872f87e1479c145a5e
0c706908df97857255252837ac1b90c9
D24cd19a50e6d574a0cfdfc07c6d22bb

91[.]132[.]92[.]60
138[.]124[.]184[.]174
148[.]72[.]168[.]13
148[.]72[.]171[.]171
199[.]127[.]60[.]236
45[.]227[.]254[.]26
80[.]66[.]88[.]203
91[.]240[.]118[.]29
152[.]89[.]196[.]111
194[.]26[.]29[.]102
185[.]11[.]61[.]114
23[.]83[.]133[.]104
23[.]108[.]57[.]151
64[.]44[.]102[.]190
20[.]99[.]133[.]109
20[.]99[.]185[.]48
13[.]107[.]4[.]50
192[.]229[.]211[.]108
23[.]216[.]147[.]64
23[.]216[.]147[.]76
64[.]44[.]135[.]135
162[.]159[.]130[.]233
162[.]159[.]134[.]233
162[.]159[.]133[.]233
108[.]177[.]127[.]94
108[.]177[.]119[.]95
108[.]177[.]126[.]132
23[.]106[.]215[.]210
23[.]108[.]57[.]1
157[.]254[.]194[.]99
23[.]106[.]123[.]15
23[.]82[.]140[.]10
23[.]106[.]215[.]64
23[.]108[.]57[.]240
23[.]19[.]58[.]94
23[.]108[.]57[.]94
23[.]81[.]246[.]200
108[.]62[.]118[.]197
23[.]106[.]160[.]141
23[.]106[.]223[.]200
108[.]62[.]118[.]180
23[.]82[.]140[.]122

108[.]177[.]235[.]187
64[.]44[.]102[.]207
45[.]147[.]230[.]83
64[.]44[.]102[.]133
64[.]44[.]102[.]127
108[.]62[.]141[.]243
64[.]44[.]102[.]19
108[.]62[.]118[.]131
64[.]44[.]98[.]232
23[.]108[.]57[.]213

jotuhup[.]com
zuvebeb[.]com
ceyuvigi[.]com
naporiz[.]com
xafehot[.]com
natuzujut[.]com
pucaxejun[.]com
napajep[.]com
nemucefah[.]com
jotuhup[.]com
jugiruturi[.]com
pijixepi[.]com
jahojahi[.]com
hakakebero[.]com
vezawahoy[.]com
sakogabu[.]com
xamayojir[.]com
tevokaxol[.]com
danimos[.]com
vosuxizen[.]com
lugociyah[.]com
duladani[.]com
bukifide[.]com
wijakezada[.]com
yuzowul[.]com
dehelibe[.]com
yavahiyil[.]com
rikukof[.]com
rabihino[.]com
talulime[.]com
http[:]//repairdll[.]net/jHKIOEyC/

Thanks for Reading! Happy Hunting :)