Decrypted: Akira Ransomware

Researchers for Avast have developed a decryptor for the Akira ransomware and released it for public download. The Akira ransomware appeared in March 2023 and since then, the gang claims successful attacks on various organizations in the education, finance and real estate industries, amongst others.

Note that this ransomware is not related to the Akira ransomware discovered by Karsten Hahn in 2017 and our decryptor cannot be used to decrypt files from this old variant.

The Akira ransomware comes as a 64-bit Windows binary written for Windows operating system. It is written in C++ with heavy support from C++ libraries. Additionally, Boost library was used to implement the asynchronous encryption code. The binary is linked by Microsoft Linker version 14.35.

In June 2023, a security researcher rivitna published a sample that is compiled for Linux. The Linux version is 64-bit and uses the Boost library.

Akira Encryption Schema

During the run, the ransomware generates a symmetric encryption key using CryptGenRandom(), which is the random number generator implemented by Windows CryptoAPI. Files are encrypted by Chacha 2008 (D. J. Bernstein's implementation).

The symmetric key is encrypted by the RSA-4096 cipher and appended to the end of the encrypted file. Public key is hardcoded in the ransomware binary and differs per sample.

Exclusion List

When searching files for encryption, Akira is not especially fussy. Whilst ransomware strains usually have a list of file types to encrypt, Akira has a list of files not to encrypt:

.exe, .dll, .lnk, .sys, .msi, akira_readme.txt

Furthermore, there are folders that are always ignored by Akira:

Winnt, temp, thumb, $Recycle.Bin, $RECYCLE.BIN, System Volume Information, Boot, Windows, Trend Micro

There is even the legacy winnt folder, which was used as default folder for installation of Windows 2000.

Encryption Schema for Small Files

Files are encrypted depending on their size. For files of 2,000,000 bytes and smaller, the ransomware encrypts the first half of the file. The structure of such an encrypted file is as follows:

Block type	Size
Encrypted Block	File Size / 2
Plain Text Block	File Size / 2
File Footer	534 bytes

Encryption Schema for Large Files

For files sizes greater than 2,000,000 bytes, Akira encrypts four blocks. First, the size of a full block is calculated (see Figure 1).

```
HalfOfFileSize = FileSize / 2;
BlockLength = (FileSize - HalfOfFileSize / 5 * 4) / 5;
```

Figure 1: Akira's calculation of full encryption block size.

The size of the encrypted part of the block is then calculated (see Figure 2).

```
EncryptedLength = FileSize * 50 / 500;
```

Figure 2: Akira's calculation of the size of encryption portion of block.

The layout of an encrypted file is then created (see Figure 3).

Block type	Size
Encrypted Block #1	EncryptedLength
Plain Text Block	BlockLength – EncryptedLength
Encrypted Block #2	EncryptedLength
Plain Text Block	BlockLength – EncryptedLength
Encrypted Block #3	EncryptedLength
Plain Text Block	BlockLength – EncryptedLength
Encrypted Block #4	EncryptedLength
Plain Text Block	Rest of the file
File Footer	534 bytes

Figure 3: Layout of encrypted file.

The structure of the file footer can be described by the following structure in C language:

```
typedef struct _FILE_TAIL_AKIRA
{
    BYTE RsaEncrypted[0x200];          // File key, encrypted by RSA-4096
    BYTE Zeroed[0x0C];                 // Only contains zeros
    BYTE EncryptionType;               // 0x01 = half of the file are encrypted
                                       // 0x02 = 4 blocks of the file are encrypted
    BYTE Version;                      // 0x32 ('2') - ransomware version (?)
    BYTE OriginalSize[0x08];           // Original file size, in bytes (little endian)

} FILE_TAIL_AKIRA, *PFILE_TAIL_AKIRA;
```

Figure 4: File footer structure.

Encrypted files can be recognized by the extension .akira. A file named akira_readme.txt – the ransom note – is dropped in each folder (see Figure 5).

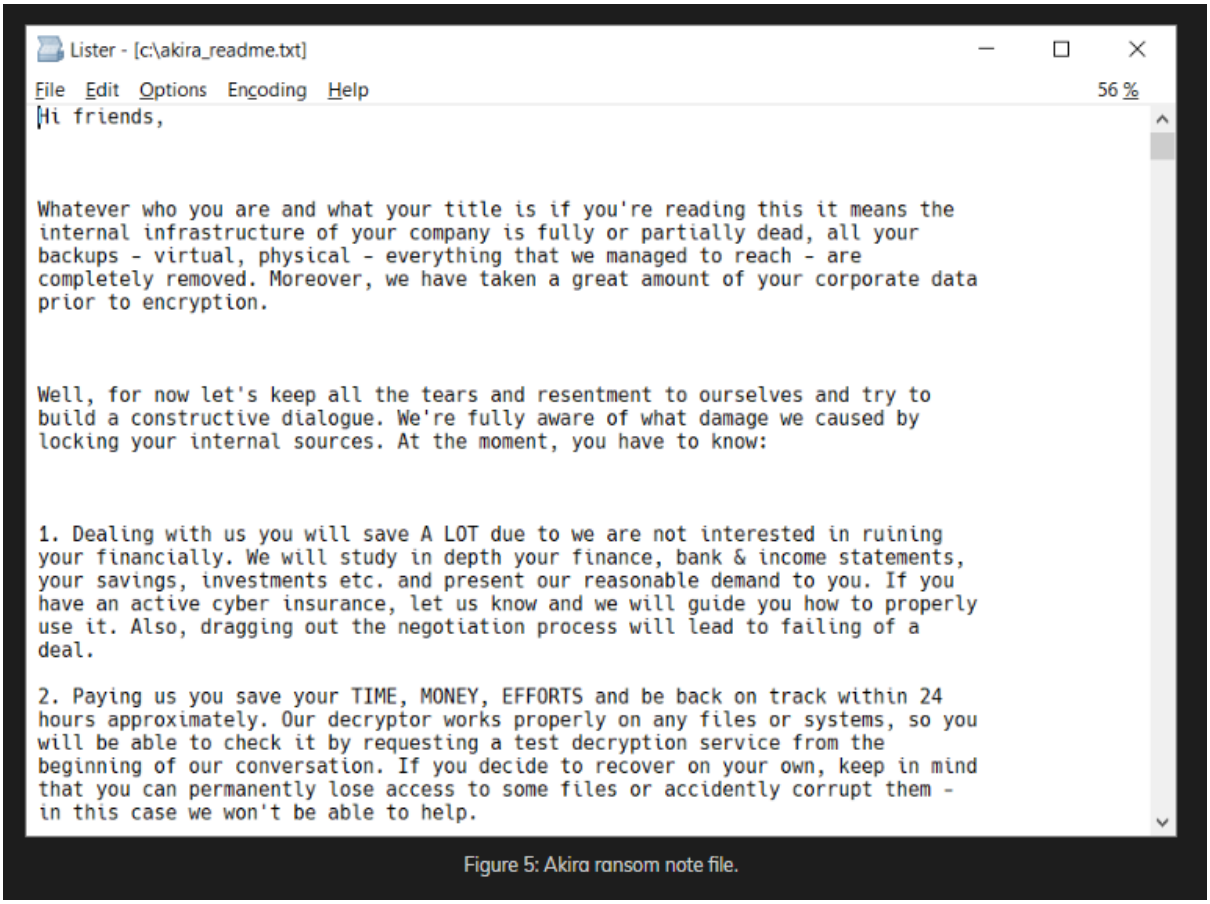


Figure 5: Akira ransom note file.

The ransom note mentions two TOR sites. In the first one (Figure 6), the user can list the hacked companies; in the second, victims are instructed on how to make payment (Figure 7).

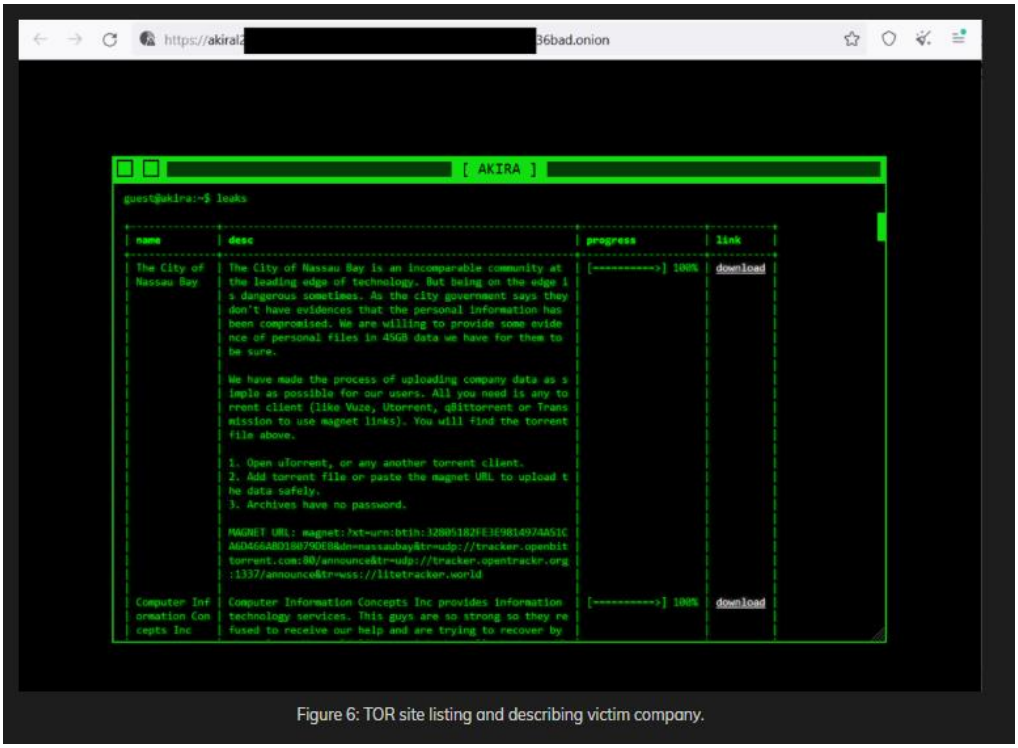


Figure 6: TOR site listing and describing victim company.

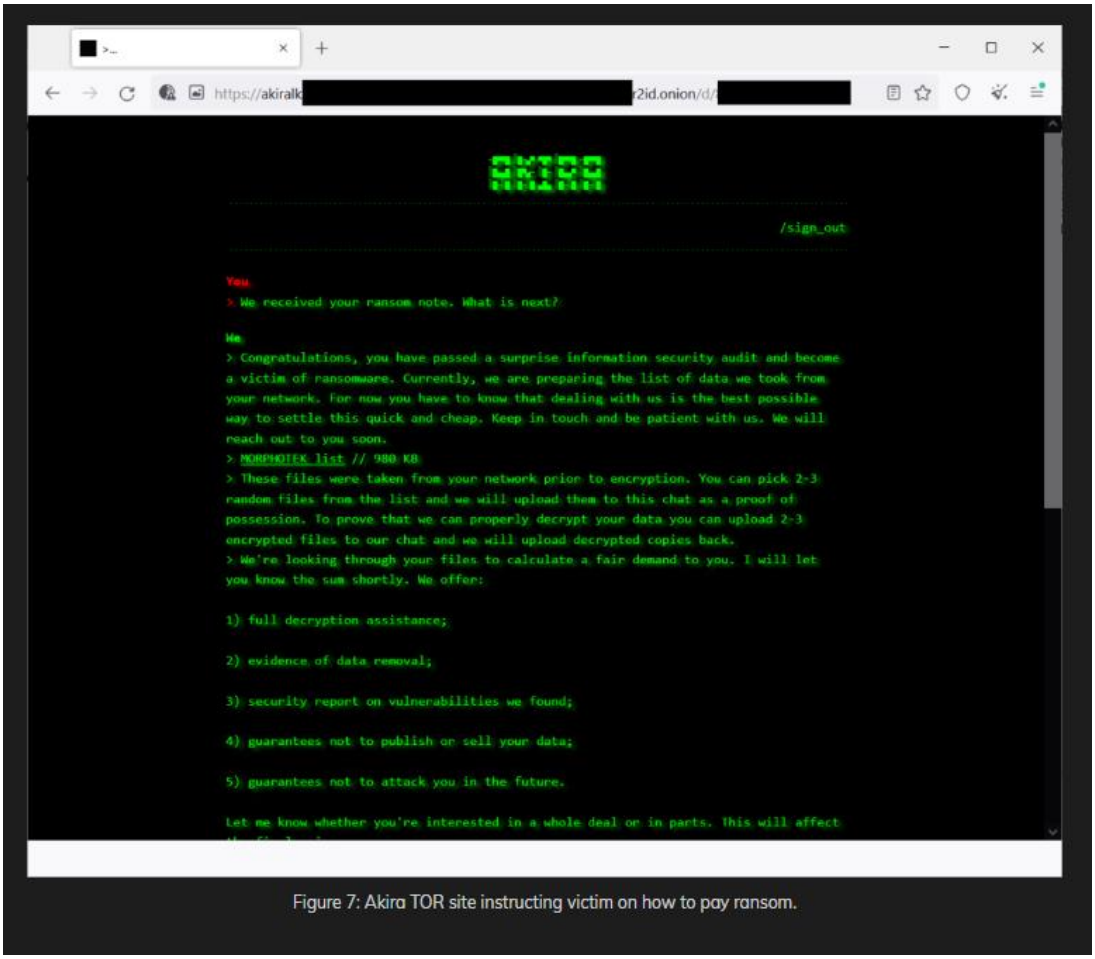


Figure 7: Akira TOR site instructing victim on how to pay ransom.

Linux Version of Akira

The Linux version of the Akira ransomware works identically like its Windows counterpart. Encrypted files have the same extension and the same encryption schema. Obviously, Windows CryptoAPI is not available on Linux, so the ransomware authors used Crypto++ library to cover the parts that are handled by CryptoAPI in Windows.

Our team is currently developing a Linux version of our decryptors. In the meantime, the Windows version of the decryptor can be used to decrypt files encrypted by the Linux version of the ransomware. Please use WINE layer to run the decryptor under Linux.

Similarities to Conti

Akira has a few similarities to the Conti v2 ransomware, which may indicate that the malware authors were at least inspired by the leaked Conti sources. Commonalities include:

1. List of file type exclusions. Akira ignores files with the same extensions as Conti, except that there's **akira_readme.txt** instead of **R3ADM3.txt**.

2. List of directory exclusions. Again, Akira ignores the same folders as Conti, including winnt and Trend Micro, which makes Trend Micro's default installation folder especially resilient against both ransomware strains.
3. The structure of the Akira file tail is equal to the file tail appended by Conti (see Figure 8)

```
#pragma pack(push, 1)
typedef struct _CONTI_FILE_TAIL_V2
{
    BYTE Encrypted[0x200];
    DWORD dwZero1;                          // Zero
    DWORD dwZero2;                          // Zero
    DWORD dwZero3;                          // Zero
    BYTE bEncryptType;                      // CONTI_CRYPT_OLD_XXX
    BYTE bDataPercent;                      // Usually 0
    ULONGLONG FileSize;                     // Original file size
} CONTI_FILE_TAIL_V2, *PCONTI_FILE_TAIL_V2;
#pragma pack(pop)
```

Figure 8: Conti ransomware file tail.

The member variable bEncryptType is set to 0x24, 0x25, 0x26 by Conti version 2, Akira uses 0x32.

4. The implementation of ChaCha 2008 used by Akira ransomware is the same as the one used by Conti ransomware.
5. The code for key generation (two calls to CryptGenRandom followed by CryptEncrypt) resembles Conti's key generation function.

How to use the Avast decryption tool to decrypt files encrypted by the ransomware

Please, read the instructions carefully. The decryption success rate will depend on it. If you don't like reading manuals, at least read the instructions about the file pair.

1. The first step is to download the decryptor binary. Avast provides a 64-bit decryptor, as the ransomware is also a 64-bit and can't run on 32-bit Windows. If you have no choice but to use 32-bit applications, you may download 32-bit decryptor here.

2. Run the executable file, preferably as an administrator. It starts as a wizard, leading you through the configuration of the decryption process.

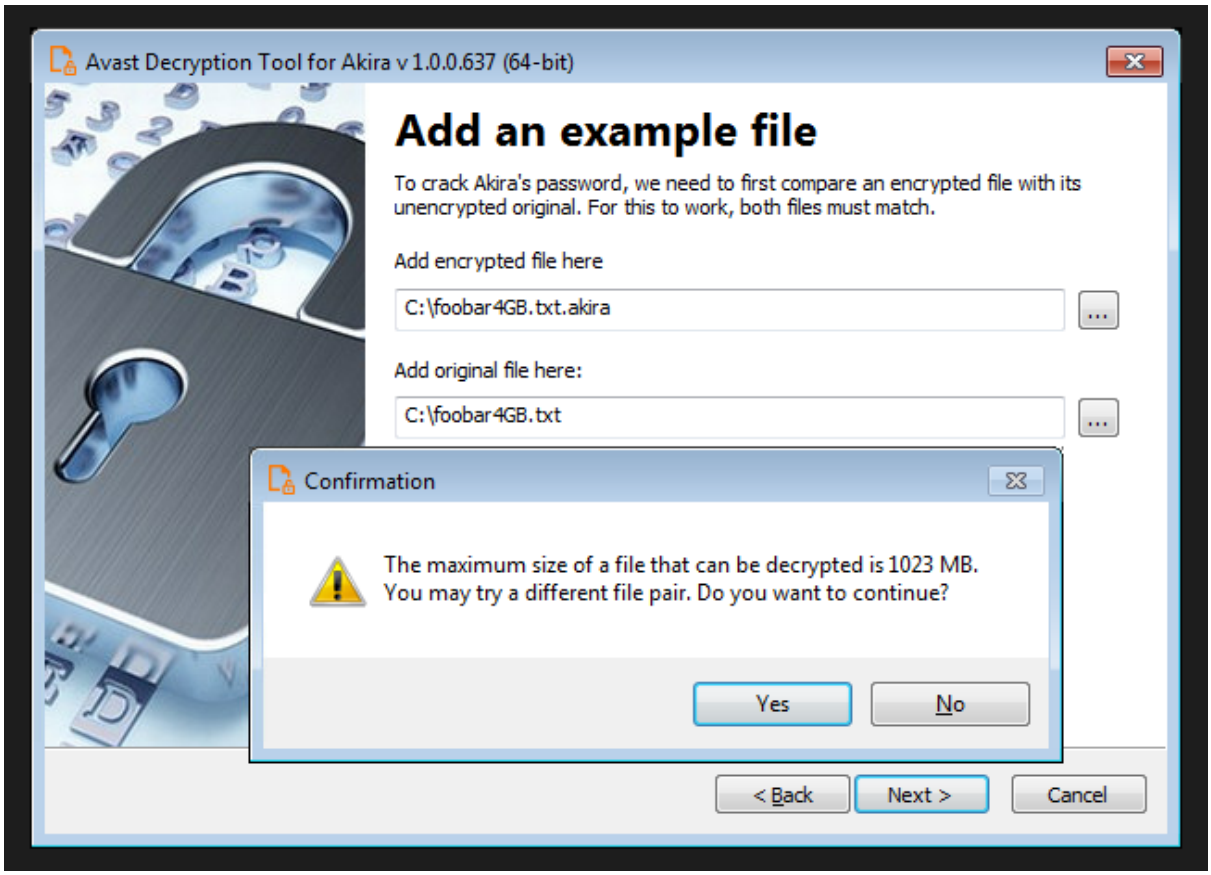
3. On the initial page, we have a link to the license information. Click the Next button when you are ready to start.

4. On the next page, select the list of locations you want to be searched for and decrypted. By default, it has a list of all local drives:

5. On the following page, you need to supply an example of a file in its original form and then one encrypted by Akira ransomware. Type both names of the files. You can also drag & drop files from Windows Explorer to the wizard page.

It is extremely important to pick a pair of files that are as big as you can find. Due to Akira's block size calculation, there may be dramatic difference on the size limit even for files that differ by a size of 1 byte.

When you click Next, the decryption tool will carefully examine the file pair and tells you what the biggest decryptable file is. In general, the size limit should be the same as the size of the original file:



6. The next page is where the password cracking process takes place. Click Start when you are ready to begin. This process usually only takes a few seconds but will require a large amount of system memory. This is why we strongly recommend using the 64-bit version of the decryption tool.

Once the password is found, you can continue to decrypt all the encrypted files on your PC by clicking Next.

7. On the final page, you can opt-in to back up your encrypted files. These backups may help if anything goes wrong during the decryption process. This choice is selected by default, which we recommend. After clicking Decrypt the decryption process begins. Let the decryptor work and wait until it finishes decrypting all of your files.

For questions or comments about the Avast decryptor, email decryptors@avast.com.

IOCs (indicators of compromise)

Windows versions

3c92bfc71004340ebc00146ced294bc94f49f6a5e212016ac05e7d10fcb3312c

5c62626731856fb5e669473b39ac3deb0052b32981863f8cf697ae01c80512e5

678ec8734367c7547794a604cc65e74a0f42320d85a6dce20c214e3b4536bb33

7b295a10d54c870d59fab3a83a8b983282f6250a0be9df581334eb93d53f3488

8631ac37f605daacf47095955837ec5abbd5e98c540ffd58bb9bf873b1685a50

1b6af2fbbc636180dd7bae825486ccc45e42aefbb304d5f83fafca4d637c13cc

9ca333b2e88ab35f608e447b0e3b821a6e04c4b0c76545177890fb16adcab163

d0510e1d89640c9650782e882fe3b9afba00303b126ec38fdc5f1c1484341959

6cadab96185dbe6f3a7b95cf2f97d6ac395785607baa6ed7bf363deeb59cc360

Linux version

1d3b5c650533d13c81e325972a912e3ff8776e36e18bca966dae50735f8ab296