

Cryptography is hard: Breaking the DoNex ransomware

Gijs Rijnders
30-06-2024



whoami

- Malware reverse engineer
- CTI analyst

- Specialized in ransomware
- Finding & exploiting weaknesses to build decryptors/disrupt botnets



 [in/gijs-rijnders/](#)

 [crysearch.nl](#)

 [evolution536](#)

vx-underground

Your chances of being a victim of ransomware increases over 250% if your organization owns a computer.

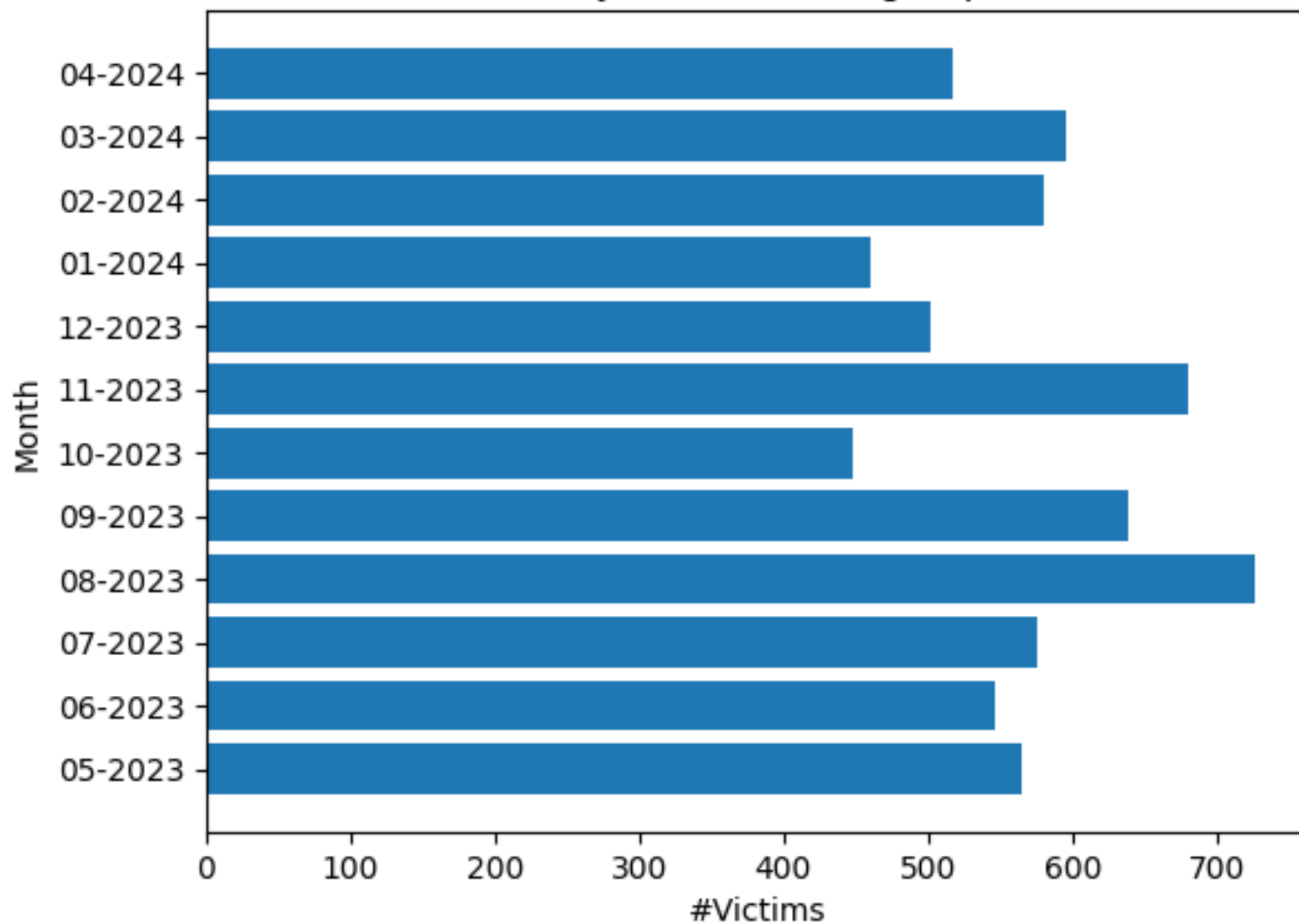
Do not use computers.

[Reposted, apparently people didn't get the joke]

88 7 100 4 3 3
3 2 2 1 3.4K 22:06



#Victims claimed by Ransomware groups (total = 6839)



Donex ransomware leakage

[Home](#) [About](#) [Archives](#)

mirel

Nous sommes votre partenaire en matière de recrutement et de sélection. Nous nous déplaçons sans engagement en entreprise afin de { ... }

2024.02.27

CHOCOTOPIA

Chocotopia is a center of entertainment in the heart of Prague. You can visit here Museum of Chocolate and experience Chocolate { ... }

2024.02.27

elsapsa

Da oltre 50 anni, Elsap è un'impresa dedita alla rappresentanza e alla distribuzione di componenti elettronici ed elettromeccanici { ... }

2024.02.24



Donex ransomware leakage

[Home](#) [About](#) [Archives](#)

CHOCOTOPIA

2024-02-27 |

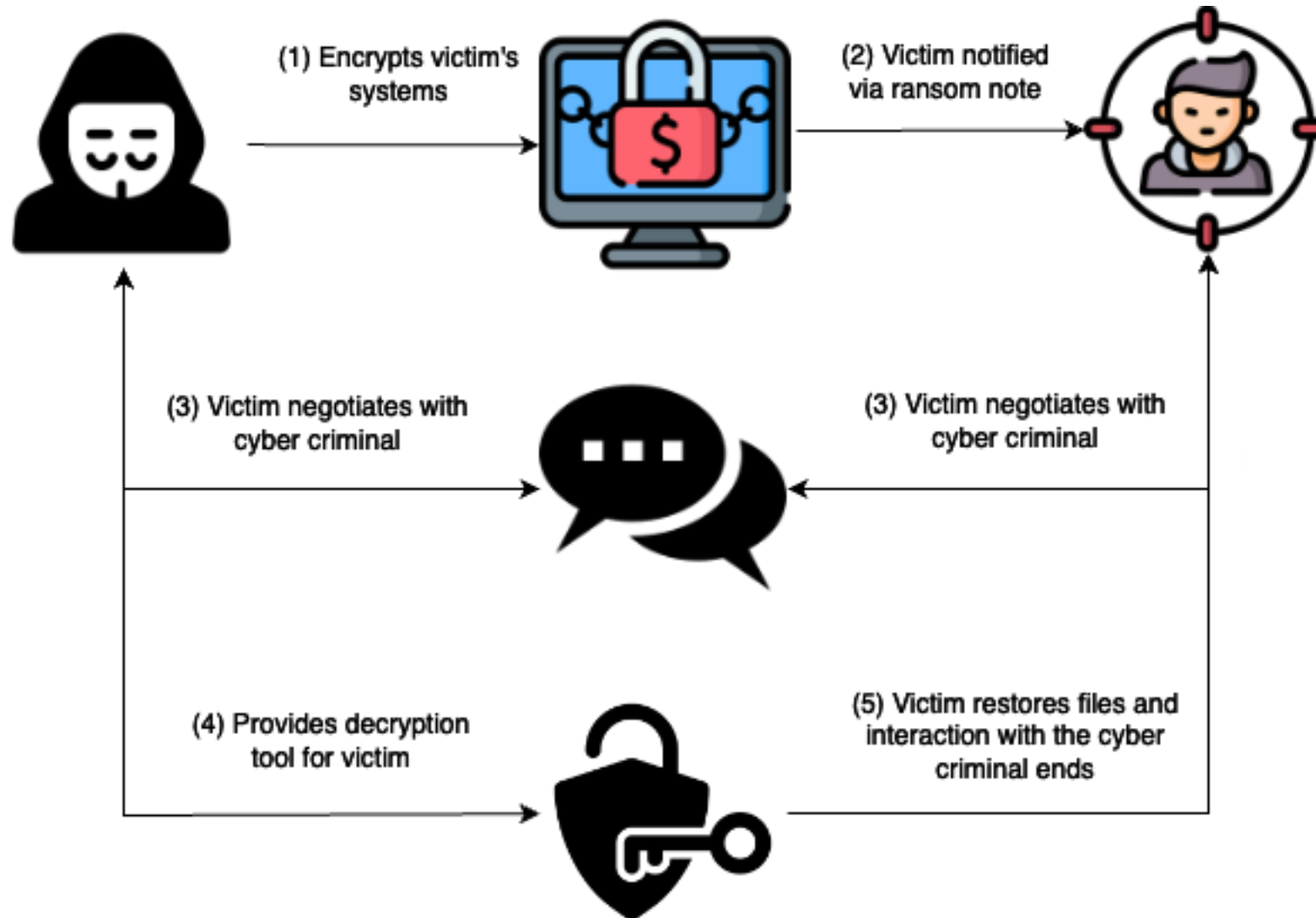
Chocotopia is a center of entertainment in the heart of Prague. You can visit here Museum of Chocolate and experience Chocolate workshops, Wax museum of legends by Grévin, Candy shop, and our Snack @ dessert bar.

Currently, our new Chocotopia Experience center is open and looking forward for visitors, who are looking for unique adventure.

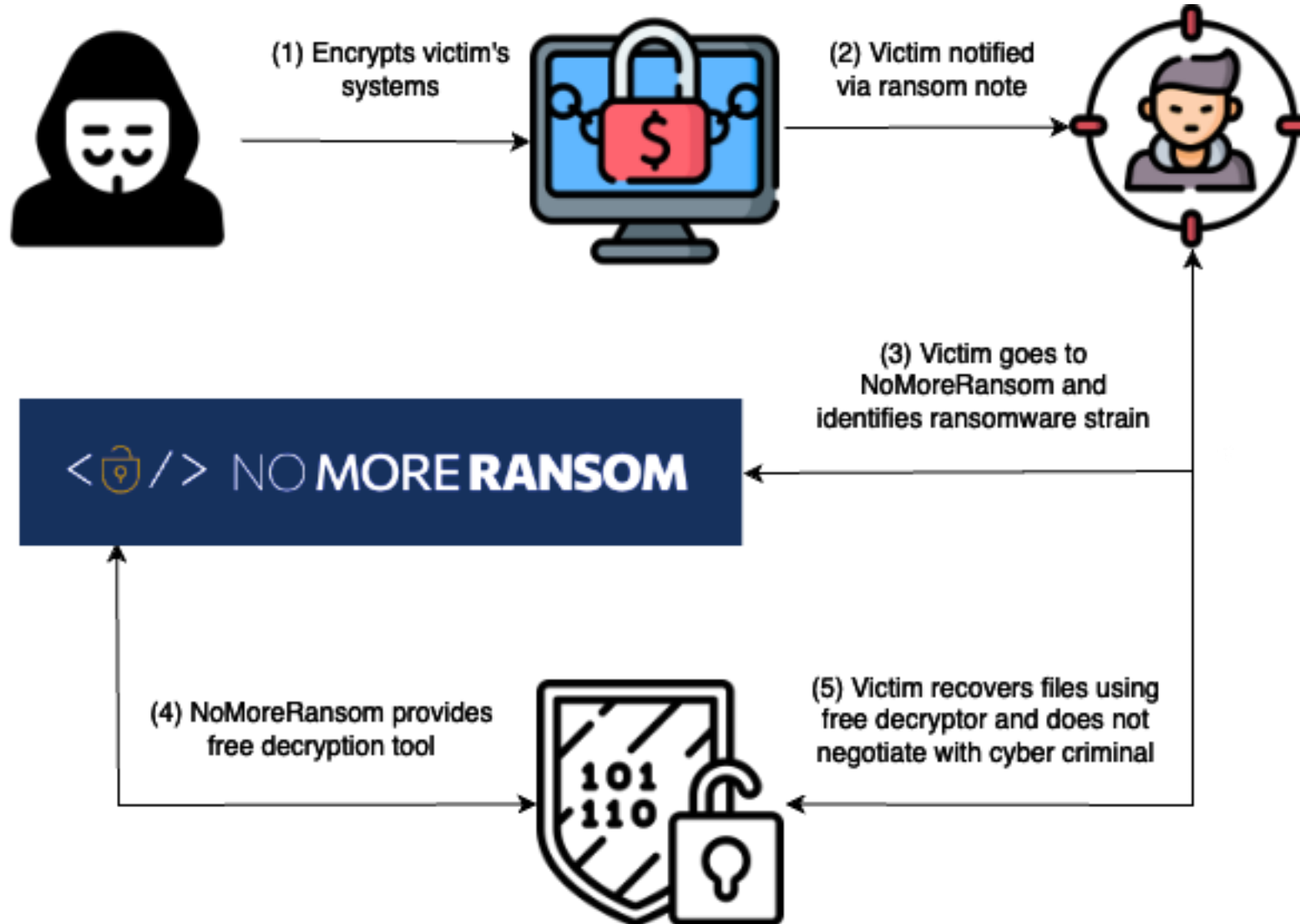
Website: www.chocotopia.cz

Total leaked:33GB

How the cyber criminal plans it

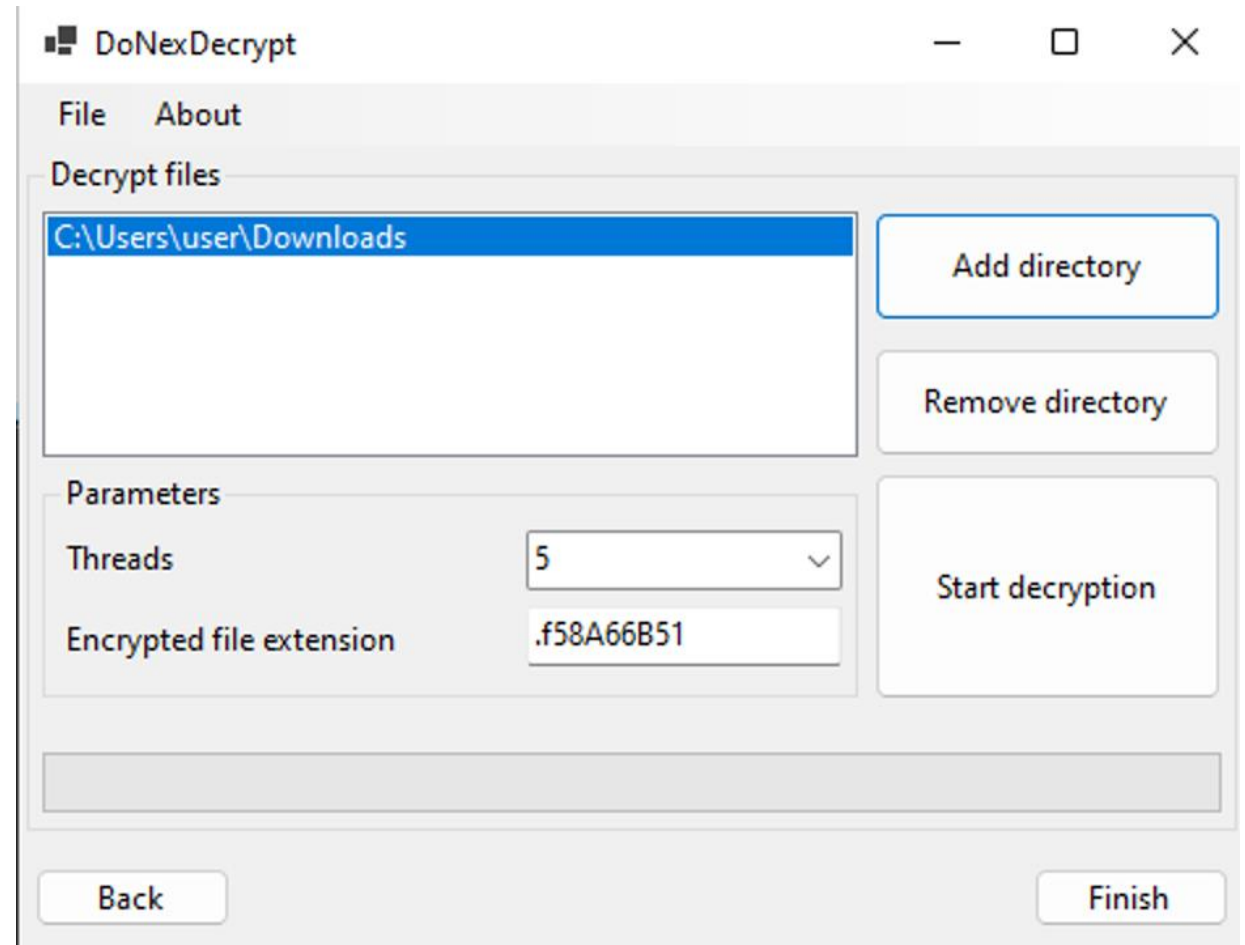


How we plan it



Building a decryptor


- Implements inverse logic of ransomware
- Based on
 - Cryptographic weakness
 - Leaked decryption keys



Distributing decryptors

Upload encrypted files here (size cannot be larger than 1 MB)

 Choose first file from PC

 Choose second file from PC

Type below any email, website URL, onion or/and bitcoin address you see in the RANSOM DEMAND.
Note: Be especially accurate with the spelling.

Or [upload](#) the file (.txt or .html) with the ransom note left by criminals

Go! Find out

TO TOP



The general advice is not to pay the ransom. By sending your money to cybercriminals you'll only confirm that ransomware works, and there's no guarantee you'll get the decryption key you need in return.

 EUROPOL

 POLITIE

Powered by:

powered by  aws

 Barracuda



And now... DoNeX

```
if ( CreateMutexA(0, 1, "CheckMutex") && GetLastError() == ERROR_ALREADY_EXISTS )
{
    _loadll(0);
    JUMPOUT(0x403338);
}
for ( i = 0; i < 0x21C0; i += 64 )
{
    *(__m128i *)&xml_config[i] = _mm_xor_si128((__m128i)xor_key_vector, *(__m128i *)&xml_config[i]);
    *(__m128i *)&xml_config[i + 16] = _mm_xor_si128((__m128i)xor_key_vector, *(__m128i *)&xml_config[i + 16]);
    *(__m128i *)&xml_config[i + 32] = _mm_xor_si128(*(__m128i *)&xml_config[i + 32], (__m128i)xor_key_vector);
    *(__m128i *)&xml_config[i + 48] = _mm_xor_si128((__m128i)xor_key_vector, *(__m128i *)&xml_config[i + 48]);
}
for ( ; i < 0x21E7; ++i )
    xml_config[i] ^= 0xA9u;
config_ptr = maybe_parse_xml(0, (char)xml_config, (int)returns_2);
```

Execute script

Snippet list

Name

 Decrypt config

Line 1 of 1

Please enter script_body

```
1 key = 0xA9
2 ea = 0x435000
3 size = 0x21E7
4 patch = True
5
6 decrypted = bytearray()
7 for i in range(size):
8     dec = get_wide_byte(ea + i) ^ key
9     decrypted.append(dec)
10     if patch:
11         patch_byte(ea + i, dec)
12 print(decrypted)
```

Line:1 Column:1



The configuration

- Ransom note
- Whitelisted files / directories
- Victim-specific options

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <white_extens>386;adv;ani;bat;bin;cab;cmd;com;cpl;cur;
  <white_files>bootmgr;autorun.inf;boot.ini;bootfont.bin
  <white_folders>$recycle.bin;config.msi;$windows.~bt;$w
  <kill_keep>sql;oracle;mysql;chrome;veeam;firefox;excel;
  <services>vss;sql;svc$;mementas;mepocs;msexchange;sophos
  <black_db>ldf;mdf</black_db>
  <encryption_thread>30</encryption_thread>
  <walk_thread>15</walk_thread>
  <local_disks>true</local_disks>
  <network_shares>true</network_shares>
  <kill_processes>true</kill_processes>
  <kill_services>true</kill_services>
  <shutdown_system>true</shutdown_system>
  <delete_eventlogs>true</delete_eventlogs>
  <cmd>wmic shadowcopy delete /nointeractive</cmd>
  <cmd>vssadmin Delete Shadows /All /Quiet</cmd>
  <content>
    !!! DoNex ransomware warning !!!
  </content>
  >>>> Your data are stolen and encrypted
```



The Cryptography: key generation

```
random_encryption_key = generates_secure_random_key((HCRYPTPROV)&savedregs, 16);  
pointer_to_footer = rsa_encrypts_buffer(random_encryption_key, 16u);
```

```
params[1] = hCryptProv;  
params[2] = retaddr;  
if ( CryptAcquireContextA(params, 0, 0, 1u, 0)  
    || GetLastError() != -2146893802  
    || (result = (char *)CryptAcquireContextA(params, 0, 0, 1u, 8u)) != 0 )  
{  
    pbBuffer = (PBYTE)malloc(random_len);  
    memset(pbBuffer, 0, random_len);  
    if ( CryptGenRandom(params[0], random_len, pbBuffer) )  
    {  
        v4 = 0;  
        if ( random_len > 0 )  
        {  
            if ( (unsigned int)random_len >= 8 && unk_439E74 >= 2 )  
            {  
                v5 = _mm_cvtsi32_si128(6u);  
                v6 = _mm_cvtsi32_si128(0x1Fu);  
                do  
                {  
                    v7 = _mm_cvtepu8_epi32(_mm_cvtsi32_si128(*(_DWORD *)&pbBuffer[v4]));  
                    v8 = _mm_sra_epi32(  
                        _mm_add_epi32(  
                            (__m128i)_mm_shuffle_ps(  
                                (__m128)_mm_mul_epi32(_mm_unpacklo_epi32(v7, v7), (__m128i)xmmword_4293F0),  
                                (__m128)_mm_mul_epi32(_mm_unpackhi_epi32(v7, v7), (__m128i)xmmword_4293F0),  
                                221),  
                            v6),  
                        v5);  
                    v4 = (int)v8 + 4;  
                } while (v4 < random_len);  
            }  
        }  
    }  
}
```



Following the trail

Direction	Typ	Address	Text
Up	r	encrypts_file_mode_1+203	push random_encryption_key
Up	r	clears_event_logs_and_s...	mov eax, random_encryption_key
Up	w	clears_event_logs_and_s...	mov random_encryption_key, 0
	w	extract_config_and_prep...	mov random_encryption_key, eax
D...	r	encrypts_local_file+37C	push random_encryption_key

Line 1 of 5

Help Search

```
FileW = CreateFileW(lpNewFileName, 0xC0000000, 0, 0, OPEN_EXISTING, 0x80u, 0); // Open file with GENERIC_READ | GENERIC_WRITE.
if ( FileW != (HANDLE)-1 )
{
    Size = 0;
    v16 = (unsigned __int64)(file_size.QuadPart / (unsigned int)number_of_blocks) >> 32;
    v21 = v16;
    v22 = file_size.QuadPart / (unsigned int)number_of_blocks;
    i = 0;
    do
    {
        lpNewFileName = (LPCWSTR)((PAIR64__(v16, v22) >> 32);
        v19 = i * v22;
        SetFilePointer(FileW, i * v22, (PLONG)&lpNewFileName, 0);
        ReadFile(FileW, lpBuffer, read_block_size, &NumberOfBytesRead, 0);
        salsa20_encrypt((int)random_encryption_key, 1, &v20, 0, (int)lpBuffer, read_block_size); // State reinitialized every time
        SetFilePointer(FileW, v18, (PLONG)&lpNewFileName, 0);
        v20 = lpBuffer;
        WriteFile(FileW, lpBuffer, read_block_size, &NumberOfBytesRead, 0);
        v16 = v21;
        i = Size + 1;
        Size = i;
    }
    while ( i < number_of_blocks );
    SetFilePointerEx(FileW, file_size, 0, 0);
    WriteFile(FileW, pointer_to_footer, 0x200u, &NumberOfBytesRead, 0); // Writes footer, size = 512 bytes
    CloseHandle(FileW);
    writes_random_note(lpFileName);
    if ( v19 )
        j__free_base(v19);
}
}
```



The encryption function

```
    __int64 *nonce_ptr,  
    unsigned int a4,  
    PBYTE buffer,  
    int buffer_length)  
{  
    void *key_schedule_proc; // edx  
    PBYTE v7; // eax  
    unsigned int v8; // ebx  
    int v9; // edi  
    BYTE *v10; // eax  
    int v11; // esi  
    BYTE *v12; // ecx  
    char keystream[64]; // [esp+0h] [ebp-50h] BYREF  
    __int64 nonce_int64; // [esp+40h] [ebp-10h] BYREF  
    __int64 v16; // [esp+48h] [ebp-8h]  
    void *use_128_bita; // [esp+5Ch] [ebp+Ch]  
    PBYTE buffera; // [esp+68h] [ebp+18h]
```

```
    key_schedule_proc = salsa20_schedule_32;  
    if ( ! use_128_bit )  
        key_schedule_proc = 0;  
    v16 = 0LL;  
    if ( use_128_bit == 1 )  
        key_schedule_proc = salsa20_schedule_16;  
    use_128_bita = key_schedule_proc;  
    if ( !key_schedule_proc )  
        return 1;  
    if ( !encryption_key )  
        return 1;  
    if ( !nonce_ptr )  
        return 1;  
    v7 = buffer;  
    if ( !buffer )  
        return 1;  
    v8 = a4;  
    nonce_int64 = *nonce_ptr;  
    if ( (a4 & 0x3F) != 0 )  
    {  
        LOBYTE(v16) = a4 >> 6;  
        BYTE1(v16) = a4 >> 14;  
        BYTE2(v16) = a4 >> 22;  
        BYTE3(v16) = a4 >> 30;  
        ((void (__cdecl *) (PBYTE, __int64 *, char *))key_schedule_proc)(encryption_key, &nonce_int64,  
        v7 = buffer;
```

```
unsigned int __cdecl salsa20_schedule_32(int a1, int a2, int a3)  
{  
    int v3; // esi  
    int v4; // edi  
    unsigned int v5; // kr00_4  
    int v6; // ecx  
    int v8; // edx  
    int v9; // edi  
    char v10; // al  
    int v11; // esi  
    int v12; // edx  
    int v13; // ecx  
    int v14; // eax  
    int v15; // edx  
    int v16; // edx  
    unsigned int v17; // ecx  
    unsigned int result; // eax  
    int v19[16]; // [esp+Ch] [ebp-90h]  
    int v20[16]; // [esp+4Ch] [ebp-50h] BYREF  
    int state[4]; // [esp+8Ch] [ebp-10h] BYREF  
  
    qmemcpy(state, "expand 32-byte k", sizeof(state));  
    v3 = a3 + 1;
```

```
unsigned int __cdecl salsa20_schedule_16(char *a1, int a2, int a3)  
{  
    int v3; // esi  
    int v4; // edi  
    unsigned int v5; // kr00_4  
    _BYTE *v6; // edx  
    char *v7; // ecx  
    int v8; // esi  
    char v9; // al  
    int v10; // esi  
    int v11; // edi  
    int v12; // edx  
    int v13; // ecx  
    int v14; // eax  
    int v15; // edx  
    int v16; // edx  
    unsigned int v17; // ecx  
    unsigned int result; // eax  
    int v19[16]; // [esp+Ch] [ebp-90h]  
    int v20[16]; // [esp+4Ch] [ebp-50h] BYREF  
    int v21[4]; // [esp+8Ch] [ebp-10h] BYREF  
  
    qmemcpy(v21, "expand 16-byte k", sizeof(v21));
```



Salsa20 or ChaCha20?

```
#include <stdint.h>
#define ROTL(a,b) (((a) << (b)) | ((a) >> (32 - (b))))
#define QR(a, b, c, d)(\
    b ^= ROTL(a + d, 7), \
    c ^= ROTL(b + a, 9), \
    d ^= ROTL(c + b, 13), \
    a ^= ROTL(d + c, 18))
#define ROUNDS 20
```

The constant is the same as Salsa20 ("expand 32-byte k"). ChaCha replaces the Salsa20 quarter-round `QR(a, b, c, d)` with:

```
a += b; d ^= a; d <<= 16;
c += d; b ^= c; b <<= 12;
a += b; d ^= a; d <<= 8;
c += d; b ^= c; b <<= 7;
```

```
int __cdecl salsa20_key_expansion(_DWORD *a1)
{
    int result; // eax

    a1[4] ^= __ROL4__(*a1 + a1[12], 7);
    a1[8] ^= __ROL4__(a1[4] + *a1, 9);
    a1[12] ^= __ROL4__(a1[8] + a1[4], 13);
    *a1 ^= __ROR4__(a1[12] + a1[8], 14);
    a1[9] ^= __ROL4__(a1[5] + a1[1], 7);
    a1[13] ^= __ROL4__(a1[9] + a1[5], 9);
    a1[1] ^= __ROL4__(a1[13] + a1[9], 13);
    a1[5] ^= __ROR4__(a1[1] + a1[13], 14);
    a1[14] ^= __ROL4__(a1[10] + a1[6], 7);
    a1[2] ^= __ROL4__(a1[14] + a1[10], 9);
    a1[6] ^= __ROL4__(a1[2] + a1[14], 13);
    a1[10] ^= __ROR4__(a1[6] + a1[2], 14);
    a1[3] ^= __ROL4__(a1[15] + a1[11], 7);
    a1[7] ^= __ROL4__(a1[3] + a1[15], 9);
    a1[11] ^= __ROL4__(a1[7] + a1[3], 13);
    result = __ROR4__(a1[11] + a1[7], 14);
    a1[15] ^= result;
    return result;
}
```



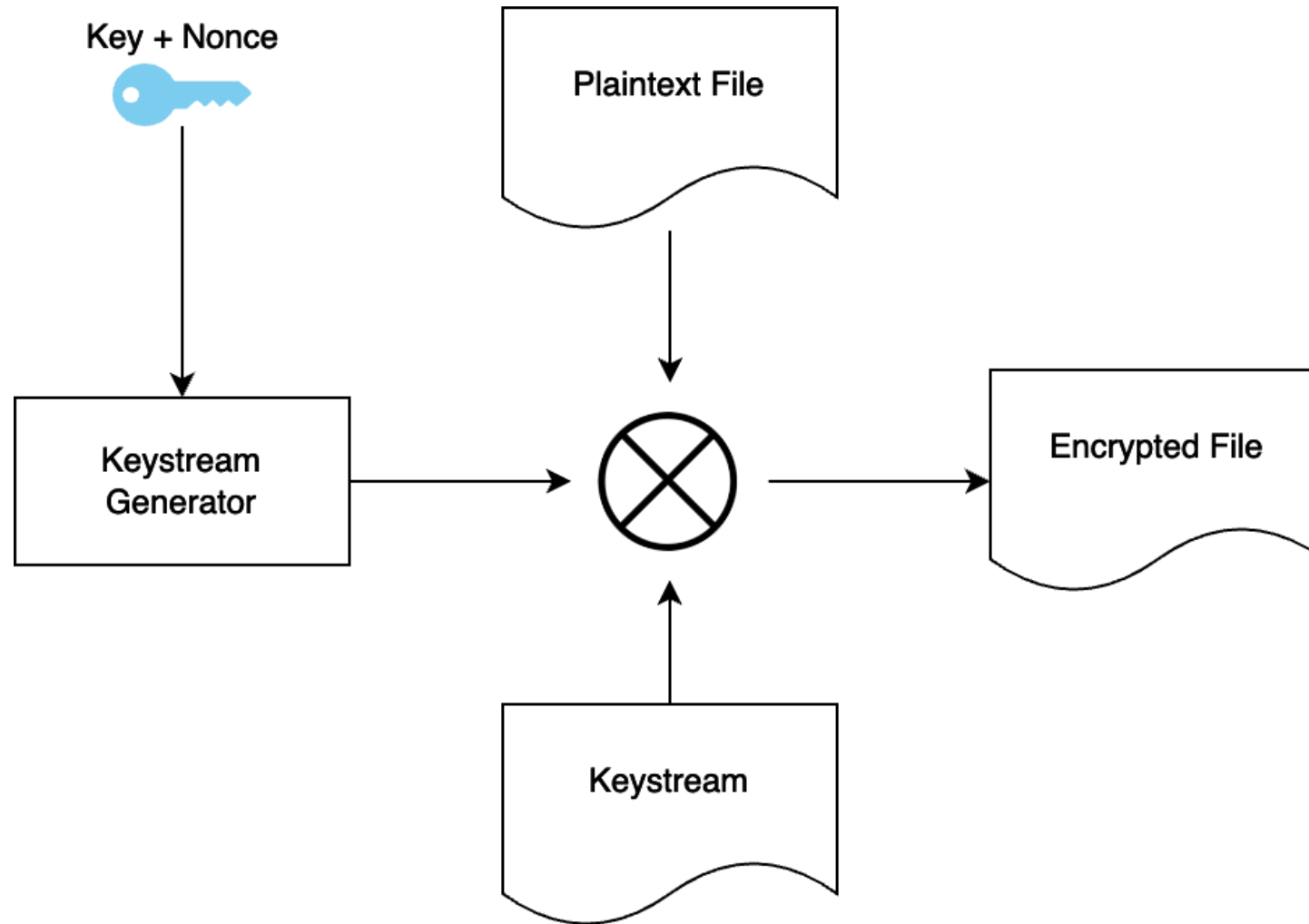
Key in global variable, nonce is zero...

```
xxd -l 128 check2.xlsx f58A66B51
00000000: ed97 e2de 5249 056 2236 5bb4 1791 37e9 ....RI.V"6[...7.
00000010: 8d43 7726 2542 d2af d7a2 b50f 2797 d94c .Cw&%B.....'..L
00000020: 310e 06bb b3a0 c832 fa4b 53ae 83c7 71a5 1.....2.KS...q.
00000030: 0339 a23f 078c 1570 bb9f 6669 db15 d25d .9?...p..fi...]
00000040: 2f65 d38c b8f2 7259 5de5 d23a 9d15 2b2b /e....rY].....++
00000050: 35c8 578f 8fba 20e9 7f30 7d72 7b6c 7e97 5.W... ..0}r{1~.
00000060: ec93 af84 b557 9e46 1d05 19f6 7832 940a .....W.F....x2..
00000070: 3674 70c0 4567 821f e8e8 ae17 a1d3 08d2 6tp.Eg.....
```

```
xxd -l 128 check.xlsx f58A66B51
00000000: ed97 e2de 5249 056 2236 5bb4 1791 37e9 ....RI.V"6[...7.
00000010: 8d43 7726 2542 d2af d7a2 b50f 2797 d94c .Cw&%B.....'..L
00000020: 310e 06bb b3a0 c832 fa4b 53ae 83c7 71a5 1.....2.KS...q.
00000030: 0339 a23f 078c 1570 bb9f 6669 db15 d25d .9?...p..fi...]
00000040: 2f65 d38c b8f2 7259 5de5 d23a 9d15 2b2b /e....rY].....++
00000050: 35c8 578f 8fba 20e9 7f30 7d72 7b6c 7e97 5.W... ..0}r{1~.
00000060: ec93 af84 b557 9e46 1d05 19f6 7832 940a .....W.F....x2..
00000070: 3674 70c0 4567 821f e8e8 ae17 a1d3 08d2 6tp.Eg.....
```

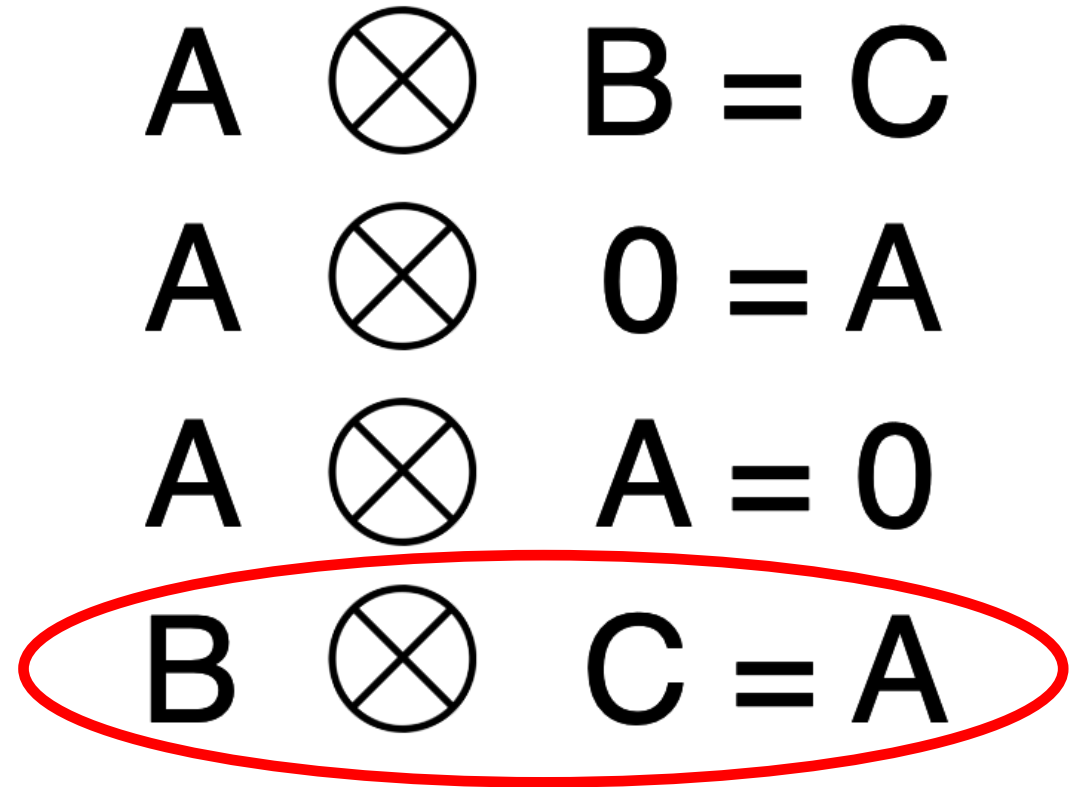


Stream ciphers & re-using key material



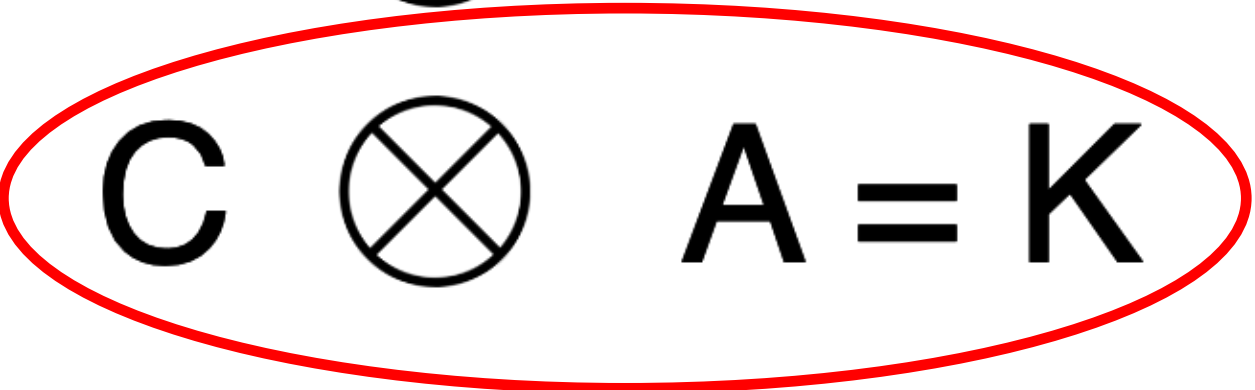
The XOR operation

A	B	A xor B
0	0	0
1	0	1
0	1	1
1	1	0



Recovering the keystream

A = plaintext
C = ciphertext
K = keystream

$$\begin{array}{ccc} A & \otimes & K = C \\ C & \otimes & A = K \end{array}$$




In practice, it's not that easy

File Size	Encrypted
< 1MiB	Entire file
<10MiB	First 1MiB
<100MiB	5 blocks of 1MiB
>100MiB	100 blocks of 1MiB

```
if ( !lpFileName )
    return;
salsa20_nonce = 0LL;
number_of_blocks = 1;
if ( file_size.HighPart > 0 )
    goto LABEL_10;
if ( file_size.QuadPart > 0x100000 )
{
    if ( file_size.HighPart < 0 || file_size.LowPart <= 0xA00000 )
        goto LABEL_11;
    if ( file_size.LowPart <= 0x6400000 )
    {
        number_of_blocks = 5;
    LABEL_11:
        read_block_size = 0x100000;
        goto LABEL_12;
    }
    LABEL_10:
        number_of_blocks = 100;
        goto LABEL_11;
}
read_block_size = file_size.LowPart;
if ( file_size.LowPart )
{
    LABEL_12:
        lpBuffer = malloc(read_block_size);
        memset(lpBuffer, 0, read_block_size);
}
```

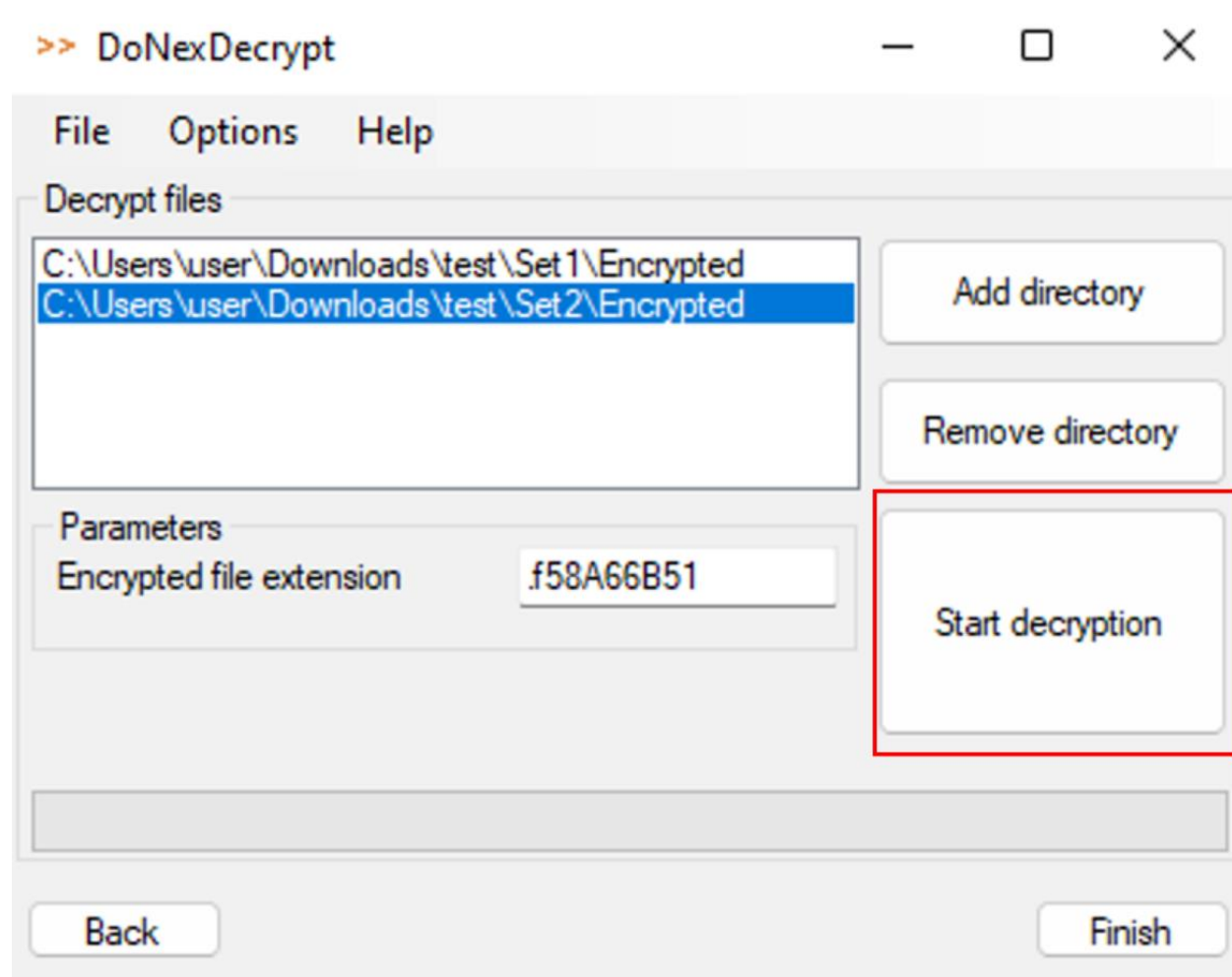


Input file constraints

```
FileW = CreateFileW(lpNewFileName, 0xC0000000, 0, 0, OPEN_EXISTING, 0x80u, 0); // Open file with GENERIC_READ | GENERIC_WRITE.
if ( FileW != (HANDLE)-1 )
{
    Size = 0;
    v16 = (unsigned __int64)(file_size.QuadPart / (unsigned int)number_of_blocks) >> 32;
    v21 = v16;
    v22 = file_size.QuadPart / (unsigned int)number_of_blocks;
    i = 0;
    do
    {
        lpNewFileName = (LPCWSTR)((i * __PAIR64__(v16, v22)) >> 32);
        v18 = i * v22;
        SetFilePointer(FileW, i * v22, (PLONG)&lpNewFileName, 0);
        ReadFile(FileW, lpBuffer, read_block_size, &NumberOfBytesRead, 0);
        salsa20_encrypt((int)random_encryption_key, 1, &v20, 0, (int)lpBuffer, read_block_size); // State reinitialized every time
        SetFilePointer(FileW, v18, (PLONG)&lpNewFileName, 0);
        v19 = lpBuffer;
        WriteFile(FileW, lpBuffer, read_block_size, &NumberOfBytesRead, 0);
        v16 = v21;
        i = Size + 1;
        Size = i;
    }
    while ( i < number_of_blocks );
    SetFilePointerEx(FileW, file_size, 0, 0);
    WriteFile(FileW, pointer_to_footer, 0x200u, &NumberOfBytesRead, 0); // Writes footer, size = 512 bytes
    CloseHandle(FileW);
    writes_ransom_note(lpFileName);
    if ( v19 )
        j__free_base(v19);
}
}
```



Putting it all together



The server has infected by ransomware



Where is backup?



On the server



