Статья Анализ шифровальщика Бабук v.3

xss.is/threads/61926

Бабук Ransomware v3

Обзор

Это краткий отчет о последнем образце программы-вымогателя Бабук. Этот образец помечен как версия 3 на основе строки однократного мьютекса.

В этой новой версии автор вредоносного ПО сохраняет большинство старых функций, за исключением схемы шифрования и многопоточности.

Поскольку я рассмотрел здесь старый образец Бабук

(http://chuongdong.com/reverse%20engineering/2021/01/03/БабукRansomware/) , в этом отчете я буду обсуждать только новые изменения.

Для шифрования Бабук использует шифрование ChaCha20, но алгоритм генерации и обмена ключами Диффи-Хеллмана на эллиптической кривой (ECDH) изменен с NIST K-571 на Curve25519 (https://en.wikipedia.org/wiki/Curve25519), одну из самых быстрых кривых ECDH.

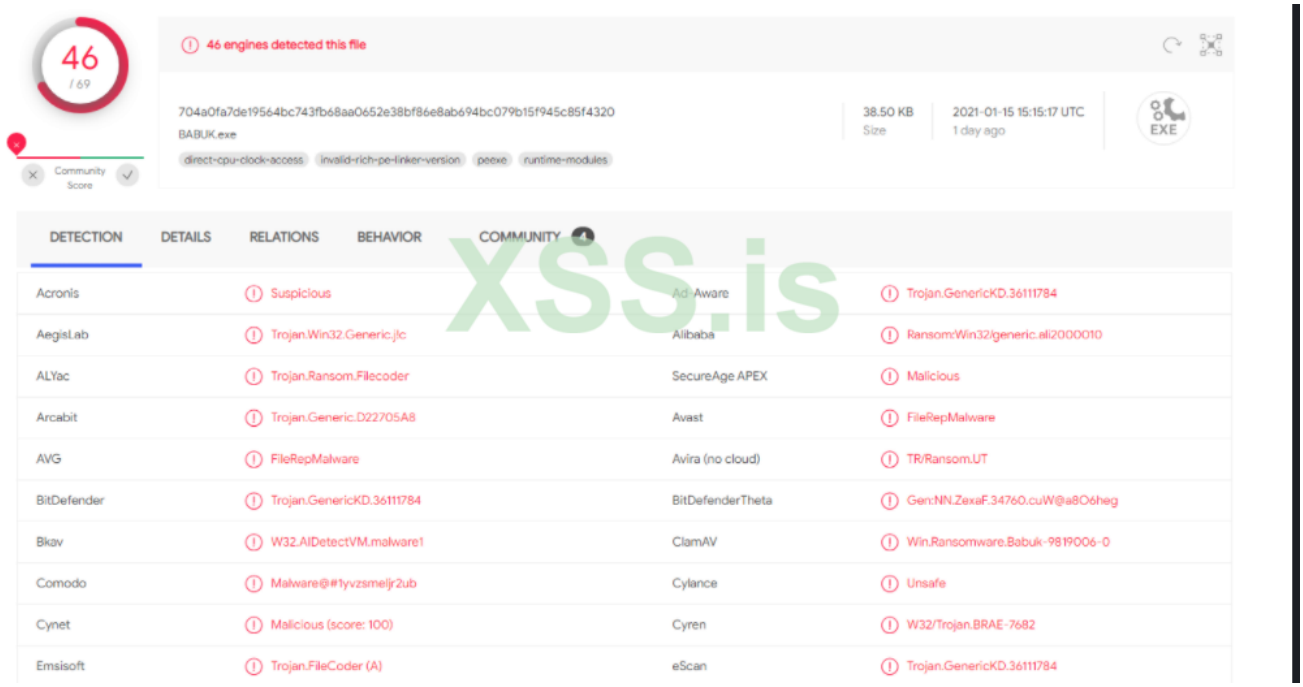
IOCS

Бабук v3 поставляется в виде 32-битного файла .exe.

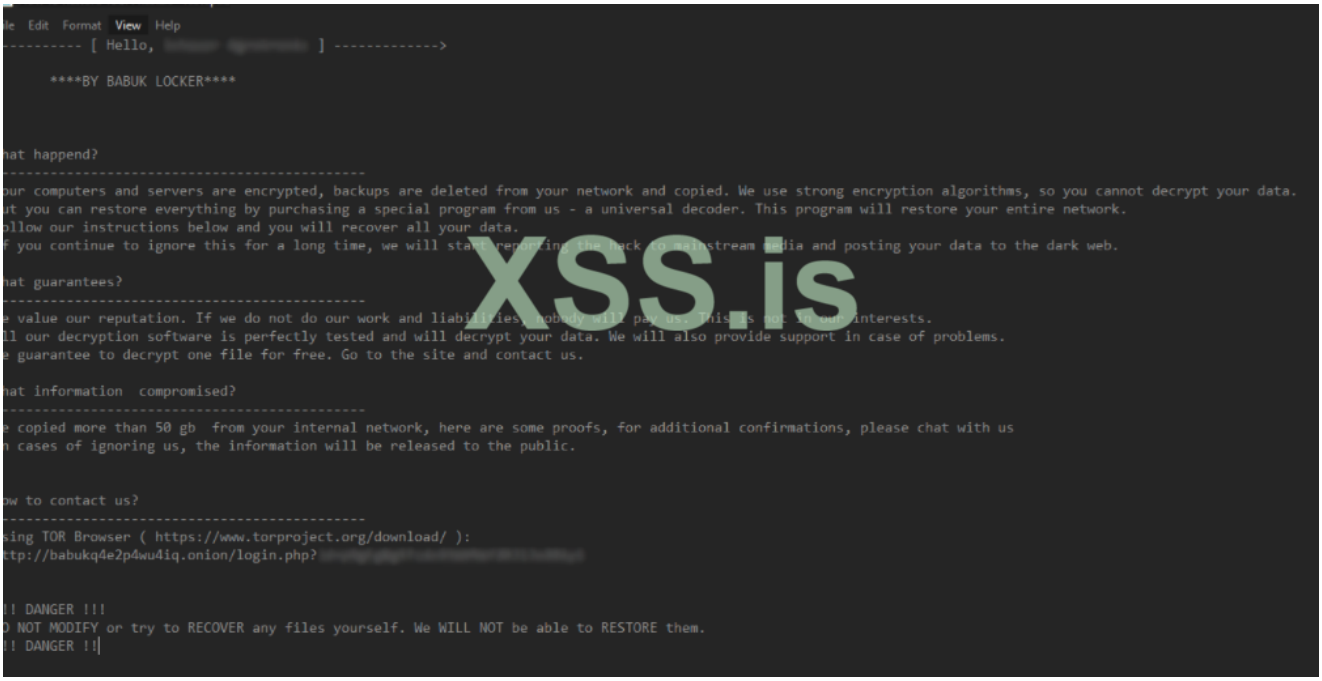
MD5: 8b9a0b44b738c7884e6a14f4cb18afff

SHA256: 704a0fa7de19564bc743fb68aa0652e38bf86e8ab694bc079b15f945c85f4320

Сэмпл:
https://bazaar.abuse.ch/sample/704a0fa7de19564bc743fb68aa0652e38bf86e8ab694bc079b15f945c85f4320/



Записка с требованием выкупа

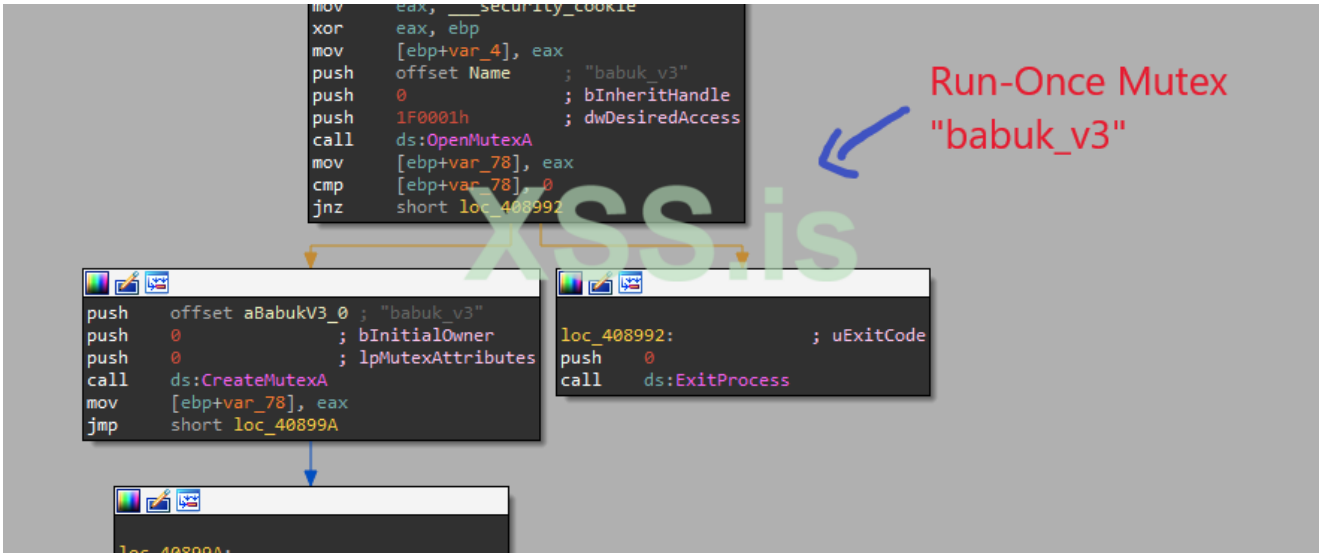


Новые изменения

Однократный мьютекс

В начале Бабук проверяет, существует ли мьютекс с именем bauk_v3, посредством вызова OpenMutexA. Если он уже существует, вредоносное ПО немедленно завершает работу.

Это обычно используется вредоносными программами, чтобы предотвратить одновременный запуск нескольких экземпляров.



Аргументы командной строки

Бабук может работать как с параметрами командной строки, так и без них.

Новые параметры командной строки: "lanfirst", "nolan" и "shares".

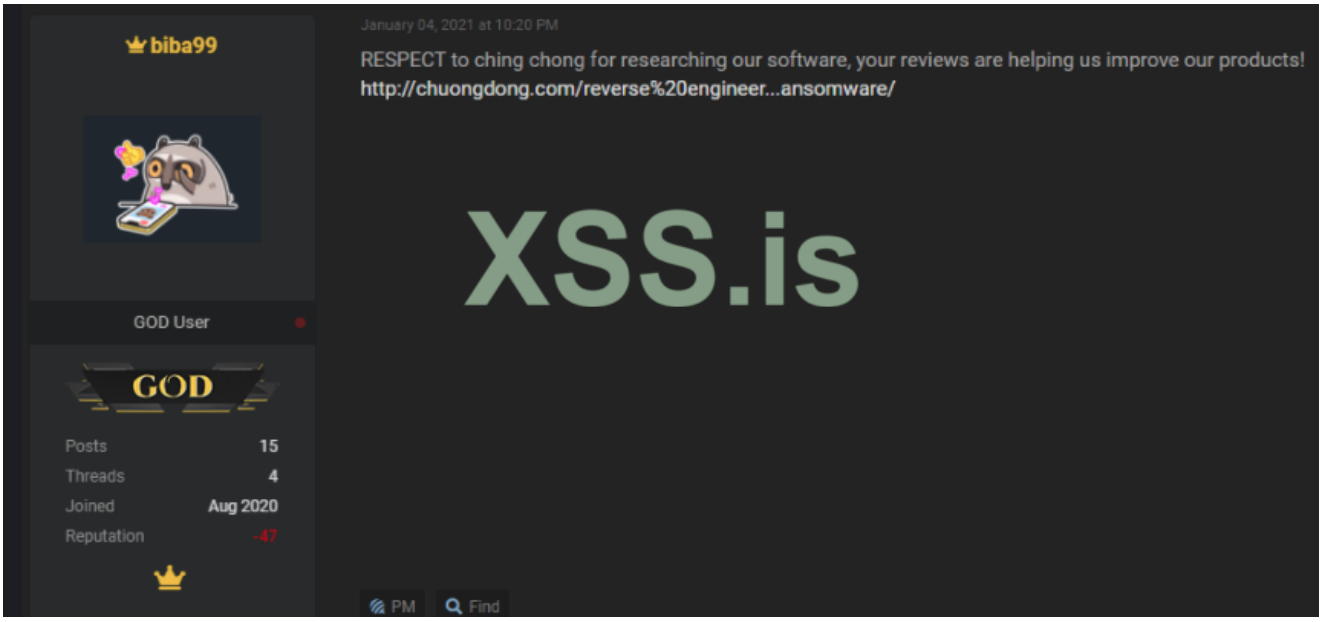
```
argv = 0;
command_line = GetCommandLineA();
v5 = parse_cmd_line(command_line, (int)&argv);
if ( cmd_arg_exist(argv, (int)v5, "lanfirst") )
{
  LAN_FLAG = 1;
}
else if ( cmd_arg_exist(argv, (int)v5, "nolan") )
{
  LAN_FLAG = -1;
}
SetProcessShutdownParameters(0, 0);
w_InitializeCriticalSection();
w_CryptAcquireContextW();
close_services();
terminate_processes();
delete_shadow_copies();
SHEmptyRecycleBinA(0, 0, 7u);
shares_strings = parse_shares(argv, (int)v5, "shares");
if ( shares_strings )
{
  v13 = 1;
  v2 = lstrlenA(shares_strings);
  for ( i = 0; i < v2; ++i )
  {
    if ( shares_strings[i] == ',' )
    {
      shares_strings[i] = 0;
      ++v13;
    }
  }
```


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

CMD Args	Functionality
-lanfirst	Encrypting other drives on LAN and locally
-nolan	Encrypting locally
shares	Encrypting shared drives and locally

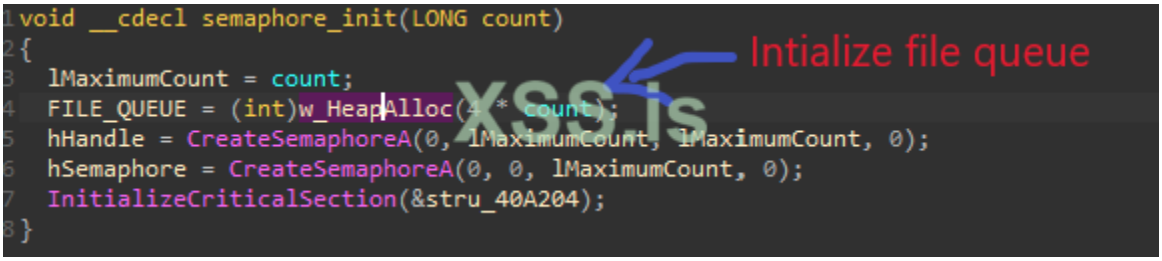
Многопоточность

Реализация многопоточности сильно изменилась по сравнению с первой версией. Я думаю, они действительно пытались улучшить его после прочтения того, что я сказал в последнем сообщении в блоге.



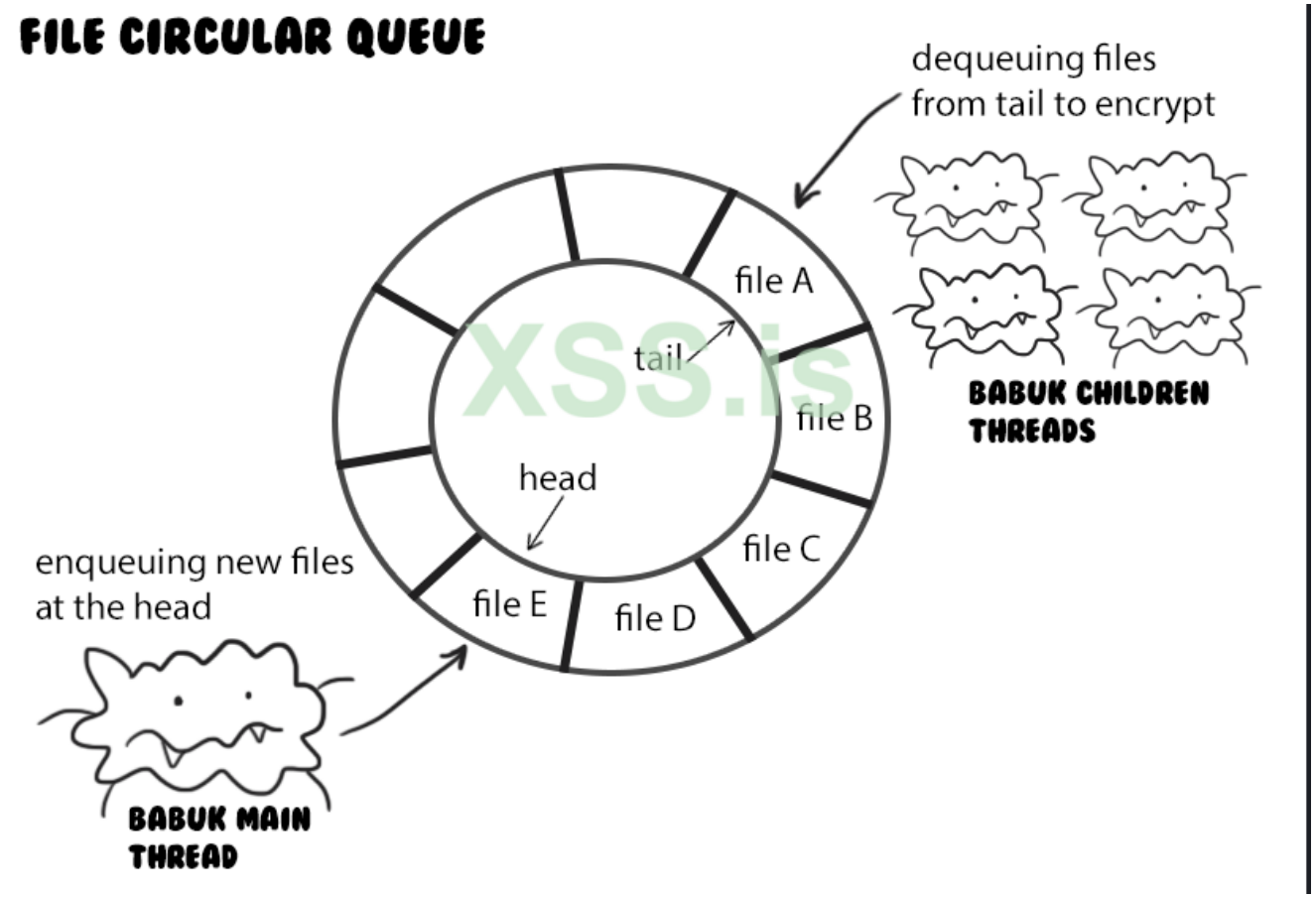
Шаги, предпринятые для улучшения функций многопоточности программы-вымогателя, идут в правильном направлении, поскольку они значительно увеличивают скорость шифрования.

Бабук использует структуру, похожую на циклическую очередь (кольцевой буфер), поддерживаемую массивом, для хранения имен файлов для шифрования. Размер очереди в два раза превышает количество процессоров в системе, то есть такое же количество порождаемых дочерних потоков.



Эта очередь является общей и используется дочерними потоками.

Родительский поток рекурсивно просматривает каталоги и ставит найденные имена файлов в начало очереди. Дочерние потоки начнут удалять их из хвоста очереди, чтобы начать шифрование.



Во-первых, Бабук создаст дочерние потоки. Количество создаваемых потоков в два раза превышает количество процессоров. Это явно не очень хорошая сумма, поэтому я понятия не имею, почему они все еще используют ее, как и в предыдущей версии. (https://docs.microsoft.com/en-us/wi...api/nf-processthreadsapi-createthread#remarks)

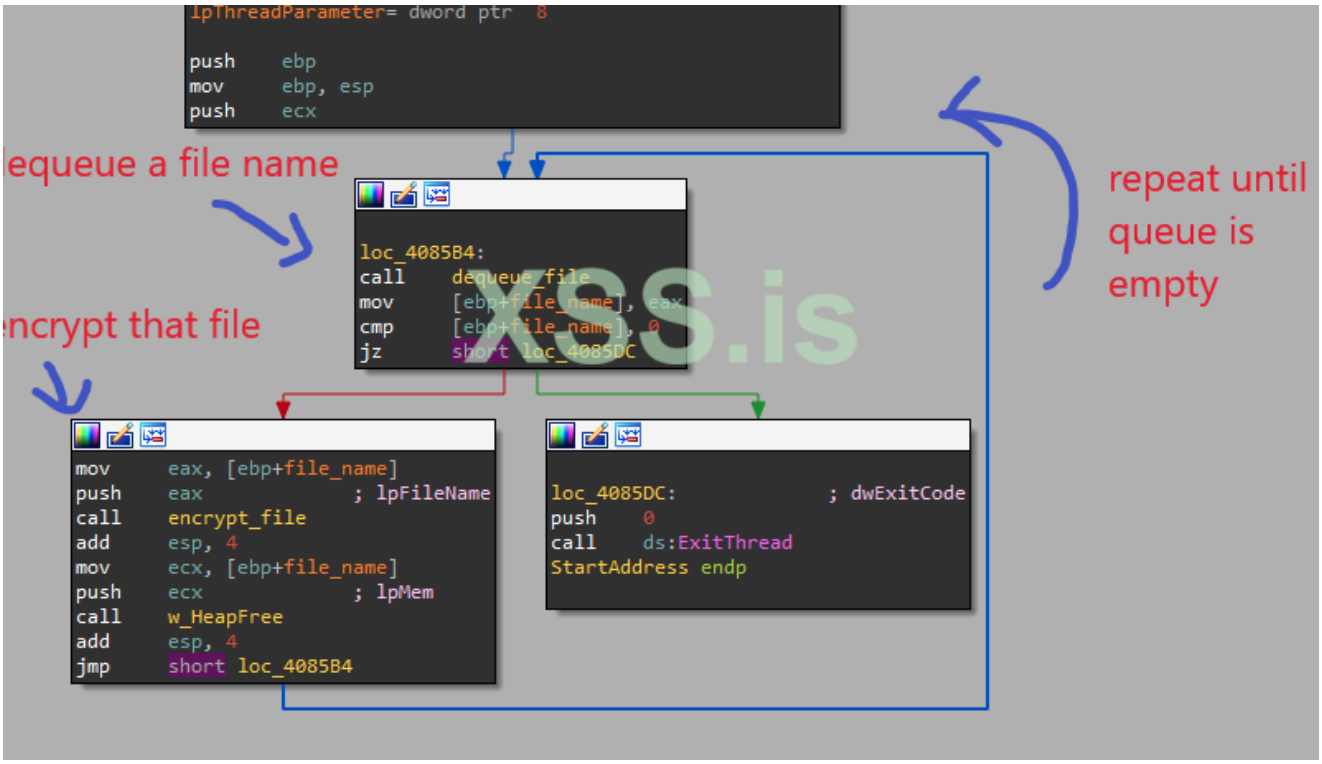
```
GetSystemInfo(&SystemInfo);
nCount = 2 * SystemInfo.dwNumberOfProcessors;
semaphore_init(2 * SystemInfo.dwNumberOfProcessors); // set up threading
lpHandles = (HANDLE *)w_HeapAlloc(4 * nCount);
if ( lpHandles )
{
  for ( k = 0; k < nCount; ++k )
    lpHandles[k] = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)StartAddress, 0, 0, 0);// start encryption
```

Затем родительский поток Бабук переходит по всему диску, проверяя, встретил ли он каталог или файл.

Найдя каталог, он снова вызовет эту функцию и спустится на другой уровень, чтобы рекурсивно пройти по этому каталогу.

Найдя файл, он поставит этот файл в начало очереди и продолжит работу.

Каждый дочерний поток удаляет файл из хвоста очереди и шифрует его.



Вот реализация постановки и удаления файлов из очереди.

```
BOOL __cdecl enQueue_file(LPCWSTR file_name)
{
  int v1; // eax
  WCHAR *file_name_1; // [esp+0h] [ebp-4h]

  file_name_1 = 0;
  if ( file_name )
  {
    v1 = lstrlenW(file_name);
    file_name_1 = w_HeapAlloc(4 * v1 + 4);
    lstrcpyW(file_name_1, file_name);
  }
  WaitForSingleObject(hHandle, 0xFFFFFFFF);
  EnterCriticalSection(&stru_40A204);
  *(FILE_QUEUE + 4 * QUEUE_TAIL_INDEX) = file_name_1;
  QUEUE_TAIL_INDEX = (QUEUE_TAIL_INDEX + 1) % lMaximumCount;
  LeaveCriticalSection(&stru_40A204);
  return ReleaseSemaphore(hSemaphore, 1, 0);
}
```

```
int dequeue_file()
{
  int file_name; // [esp+0h] [ebp-4h]

  WaitForSingleObject(hSemaphore, 0xFFFFFFFF);
  EnterCriticalSection(&stru_40A204);
  file_name = *(FILE_QUEUE + 4 * QUEUE_HEAD_INDEX);
  QUEUE_HEAD_INDEX = (QUEUE_HEAD_INDEX + 1) % lMaximumCount;
  LeaveCriticalSection(&stru_40A204);
  ReleaseSemaphore(hHandle, 1, 0);
  return file_name;
}
```

Как мы видим, Бабук использует файловую очередь, поддерживаемую массивом. Благодаря отслеживанию индексов головы и хвоста добавление и удаление имен файлов из очереди занимает постоянное время и выполняется очень быстро.

Со всеми этими новыми изменениями в реализации эта новая версия Бабук намного быстрее, чем исходная. К сожалению, есть еще много возможностей для улучшения, поскольку с точки зрения скорости и эффективности он не идет ни в какое сравнение с Конти и другими программами-вымогателями.

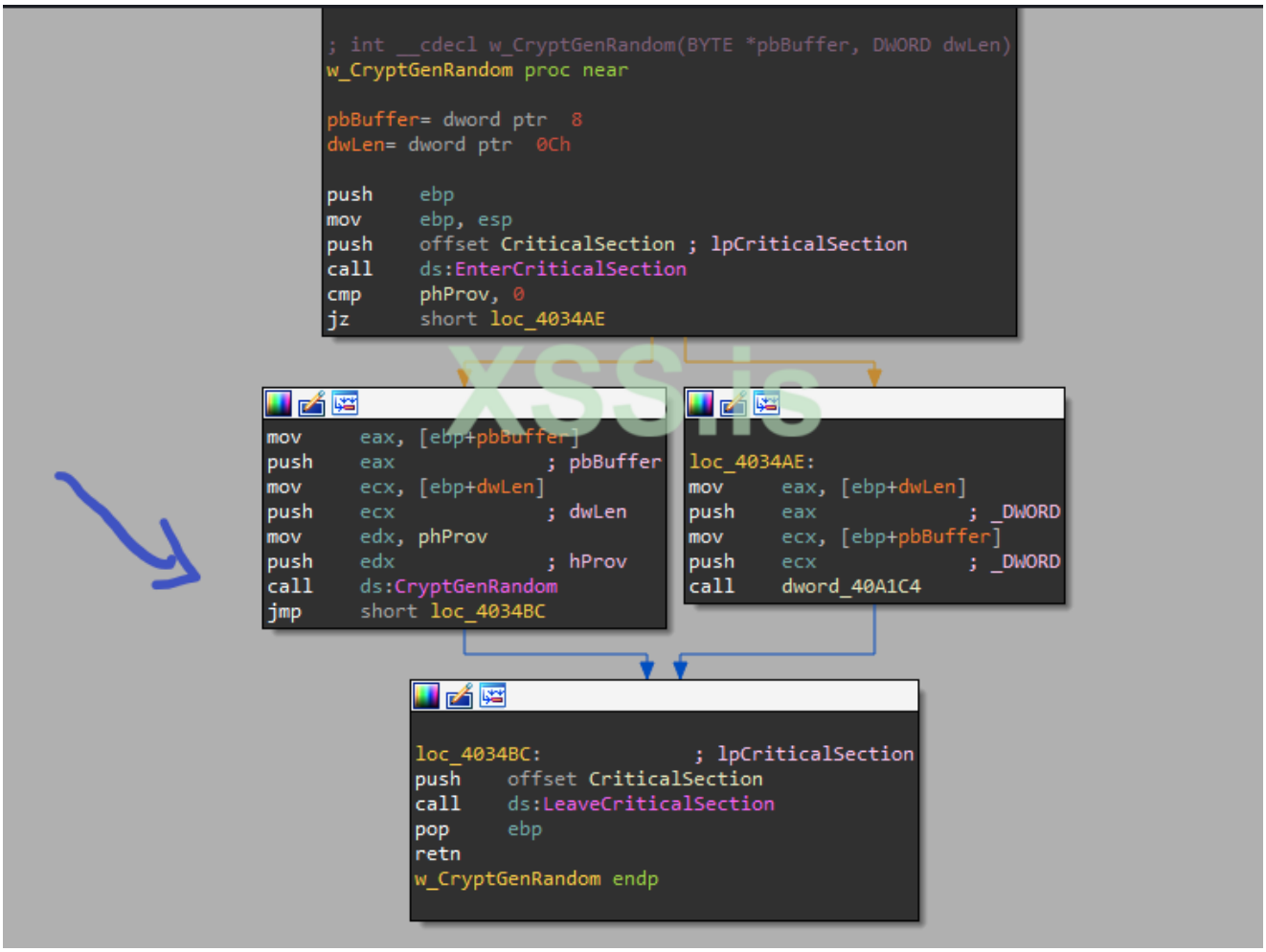
При использовании очереди на основе массива пространство ограничено. Как мы видим в функции постановки в очередь, перед добавлением в нее дополнительных файлов не проверяется, заполнена ли очередь. В теоретическом случае, когда все потоки заняты шифрованием файлов и очередь заполнена, родительский поток продолжит добавлять файлы. Поскольку это циклическая очередь, это приведет к тому, что файлы будут перезаписаны новыми до того, как дочерние потоки смогут их зашифровать, если родительский поток достаточно быстр.

Более того, автор вредоносного ПО по-прежнему придерживается старого рекурсивного подхода к обходу файлов. Поскольку только родительский поток проходит через целые диски, фрейм стека будет чрезмерно накладным, поскольку будет слишком много уровней рекурсии. По сути, это делает общее время шифрования зависимым от времени, необходимого одному потоку для обхода всей системы.

Шифрование

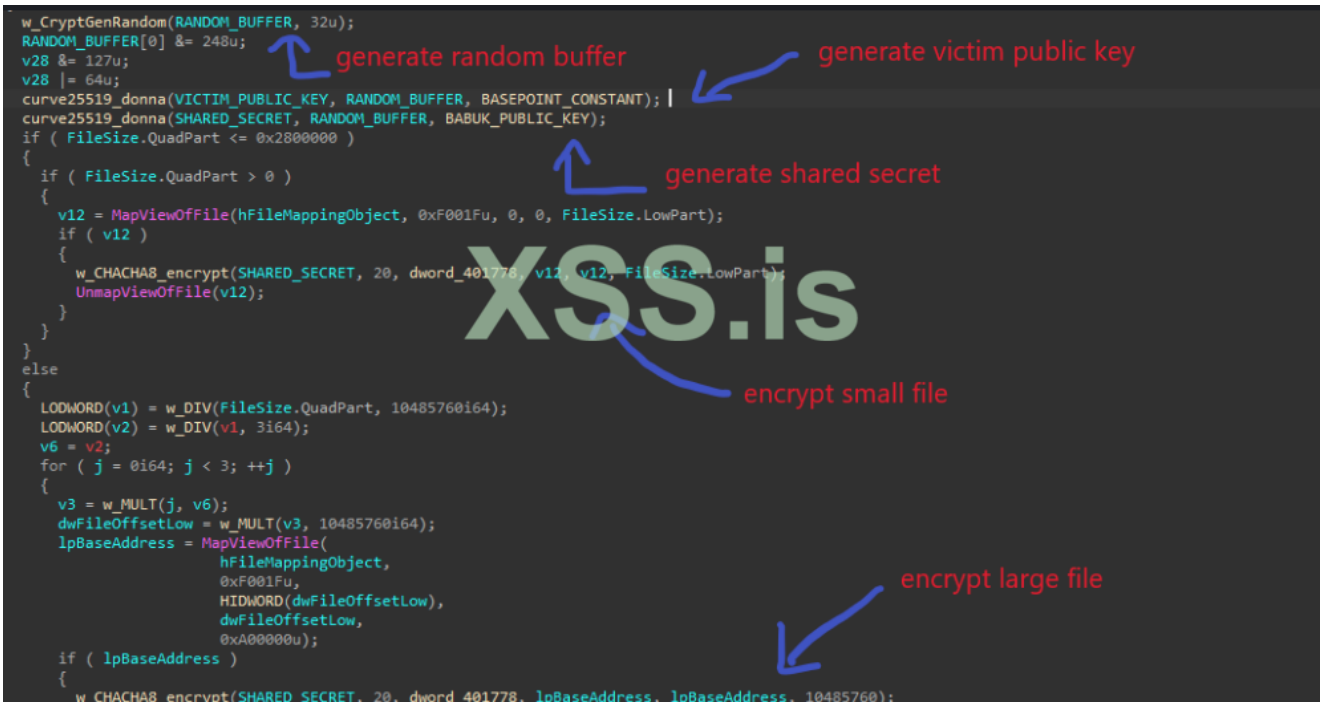
Схема шифрования осталась той же, что и в исходной версии. Однако есть небольшие изменения в генерации ключей ChaCha20.

Для каждого файла генерируется случайный буфер размером 32 байта с использованием CryptGenRandom.



Затем, используя именно этот фрагмент реализации Curve25519, Бабук сгенерирует открытый ключ для жертвы из случайного буфера, используя ECDH.(https://github.com/agl/curve25519-donna/blob/master/curve25519-donna.c)

Он также сгенерирует общее секретное число, используя свой жестко закодированный открытый ключ и буфер случайных чисел. Этот общий секрет в конечном итоге используется в качестве ключа ChaCha20 для шифрования файла.



Наконец, в конец файла записывается открытый ключ жертвы, который будет использоваться для расшифровки, если жертва решит заплатить.

Не уверен, что это было таак задумано, но я считаю, что группа Бабук испортила фазу генерации открытого ключа.

По словам Дэна Бернштейна, автора этой функции Диффи-Хеллмана, вот процедура генерации открытого ключа с использованием Curve25519. (https://cr.yp.to/djb.html)

Computing public keys. To generate the corresponding 32-byte Curve25519 public key `mypublic[0]`, `mypublic[1]`, ..., `mypublic[31]`, call

```
curve25519(mypublic,mysecret,basepoint);
```

where the constant `basepoint` is 9 followed by all zeros:

```
const unsigned char basepoint[32] = {9};
```

Future library releases will support a more concise `curve25519_public` function.

Вместо того, чтобы использовать 9, за которыми следуют все нули, Бабук использует массив всех 9 значений.

```
                                        ; encrypt_file+2E8↓o
BASEPOINT_CONSTANT db 9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9
                                        ; DATA XREF: encrypt_file+129↓o
```

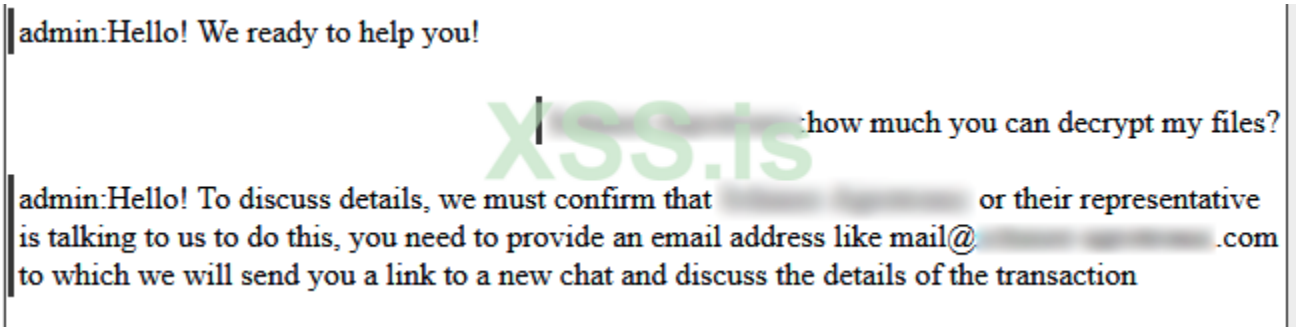

Если автор вредоносного ПО не изменил математику в исходном коде Curve25519, чтобы приспособиться к этому (что маловероятно), эта константа базовой точки может не сгенерировать правильный открытый ключ.

С неправильным открытым ключом автор вредоносного ПО не может сгенерировать правильный общий секрет для расшифровки файлов.

Генерация ключей

Новая версия Бабук была улучшена, чтобы шифровать файлы с гораздо большей скоростью, используя улучшенный многопоточный подход. Несмотря на то, что ему еще многое предстоит улучшить, Бабук действительно эффективно атаковал многие корпорации, используя шифрование ChaCha20, а также алгоритм Диффи-Хеллмана на эллиптических кривых.

Как и предполагалось, команда Бабук, вероятно, использует целевые фишинговые атаки, VPN 0-day и уязвимую настройку RDP для нацеливания на определенные компании.



Сообщение новым жертвам

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

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

Если недавние жертвы Бабук 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%20engineering/2021/01/16/BabukRansomware-v3/
Эксперт