Usando "strings tag" achamos:

.encriptador - provavel nome de um hidden file

chmod u+x .encriptador && ./.encriptador - comando para rodar .encriptador

mkdir -p $USER && cp ~/* $USER 2> /dev/null - comando

http://ix.io/2c6V - link

encripta_arquivos - provavel nome de função

write_data - provavel nome de função

Rodando com GDB:

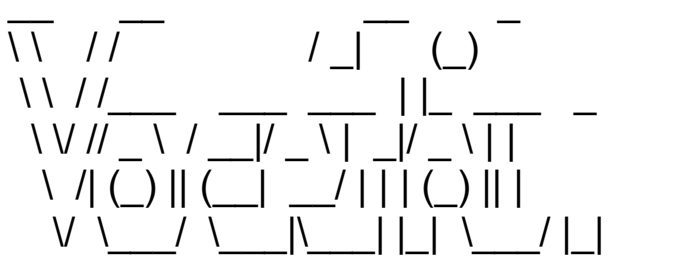
- Printa string: "Olá"
- Faz syscall: "mkdir -p $USER && cp ~/* $USER 2> /dev/null"

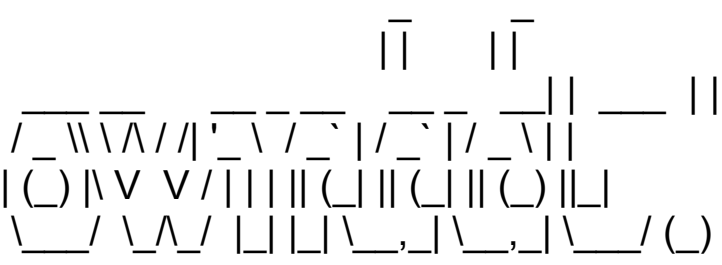
Tradução: "mkdir -p $USER": cria pasta com nome do usuário

"cp ~/* $USER 2> /dev/null": copia tudo (menos pastas e hidden files) para a pasta criada e redireciona 2 (stderr) para /dev/null, ou seja, joga fora

- perco controle total, printa e encerra o programa:

"Procure por uma forma de descodificá-los
OBS: Não desligue sua máquina, se não não será mais possível recuperar os
dados!!!





brincadeira, fiz uma cópia da sua home no diretório atual e encriptei seus
arquivos lá, rs"

No diretório atual existe um ELF chamado ".encriptador" e uma pasta vazia chamada "help" (meu nome de usuario), no home não tem nada

de estranho, só existem pastas e hidden files na minha home.

O GDB confirmou que existe as funções "encripta_arquivos" e "write_data", porém não consegui acessá-las. O GDB mostrou que usa 'libthread_db', então imagino que use pthreads.

Analizando '.encriptador"

Com "strings .encriptador":
	u3UH - suspeito
	[]A\A]A^A_ - suspeito
	usage: ./%s <argument> - provavelmente como o programa funciona
	Error : Failed to open input directory - %s - saida de erro
	%s/%s - alguma formatação de pastas, provavelmente para acessar arquivos
	Error : Failed to open %s - %s - saida de erro
	%s.leo - será que ele transforma os arquivos em [arq].leo ?
	find $USER -type f ! -name '*.leo' -delete - bash script
	;*3$" - suspeito

Com GDB:
 - getenv("USER"): pega o nome do usuário (no caso "help")
 - Printa o usage.

Vamos passar um arquivo "tst", cheio de 'a's.
 - getenv("USER"): pega o nome do usuário (no caso "help")
 - chama "atoi" (converte de ascii para int (unsafe)) passando o tst (meu argumento)
 - chama "opendir" (abre um diretório) passando "help" (o USER)
 - chama __errno_location
 - chama strerror passando 0x2
 - chama fprintf: printa para algum arquivo "Error : Failed to open input directory - No such file or directory\n"
 - encerra o programa

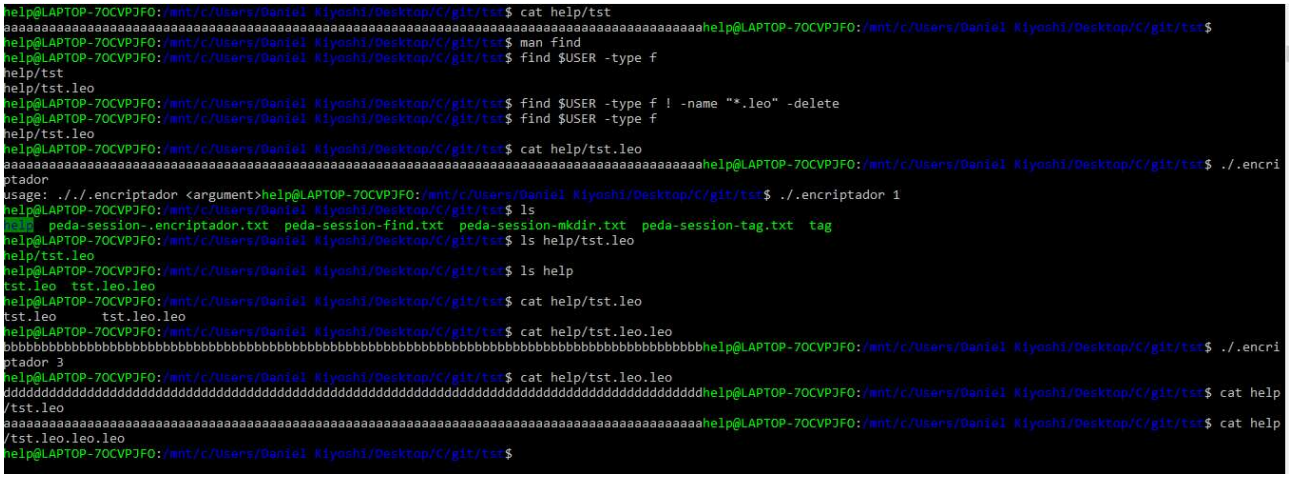
Não existia a pasta "help" (porque eu apaguei) e o programa saiu com erro.

Com isso descobrimos que o argumento provavelmente é um número (atol) e o programa precisa de uma pasta com o mesmo nome do usuário.

Vamos criar a pasta e jogar o arquivo "tst" dentro, rodar com os mesmos parâmetros:
 - Segue o mesmo processo ate o opendir, que sai com sucesso
{ começa loop
	{ começa loop
	 - chama readdir
	 - pula para atras (main+225)
	 - faz um strcmp
	 - pula para frente (main+597)
	}
	 - eventualmente sai (main+295)
	 - chama sprintf (formatando: "help/tst") (provavelmente é o arquivo que
está lá e não o argumento que eu passei)
	 - fopen para o arquivo no modo rw
	 - pula para (main+440) (skippa uma secção de erro)
	 - chama sprintf (formatando: help/tst.leo)
	 - chama fopen no modo w
	 - pula incondicionalmente (main+537)
	 - chama fgetc (retornou 'a')
	{ começa loop
	 - pula para (main+505)
	 - chama fputc escreve 'a'
	 - chama fgetc (retornou 'a')
	} // Está copiando o arquivo
	 - fecha os dois arquivos com fclose
}
 - syscall "find $USER -type f ! -name '*.leo' -delete" (deleta os arquivos que estão na pasta e não são .leo)

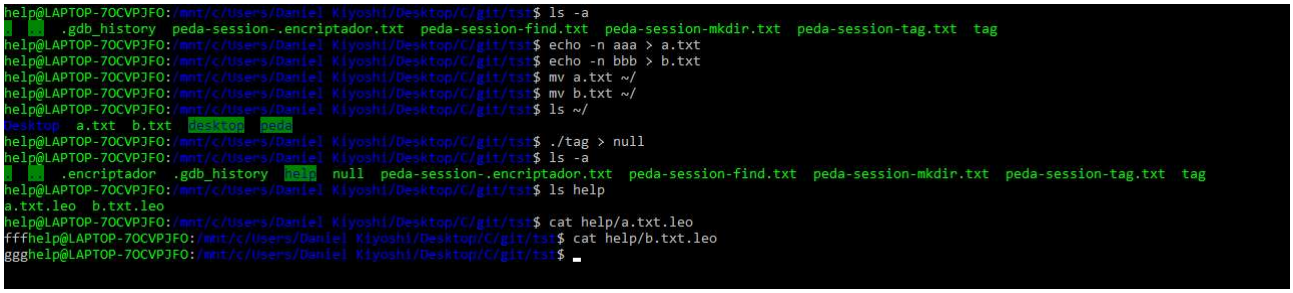
GDB crashou!

Fui testar na mão.



Ele soma o arg em cada byte do arquivo. Temos uma cifra de cesar.

Agora falta descobrir qual chave o 'tag' usa.
Como eu cansei de usar o GDB, vou fazer 2 arquivos testes um com 'a's e outro com 'b's (o segundo é para ter certeza que 'tag' usa a mesma chave).



'f' - 'a' = 5 , então a chave é 5
'g' - 'b' == 'f' – 'a' , então é a mesma chave

Se eu usar "./encriptador -5" e desfazer tudo, eu vou ficar puto e aliviado.

Estou puto e aliviado.

Conclusão:
→ tag cria uma pasta com o nome do usuário, copia a home "toda" para lá, gera o .encriptador e roda ele com o arg 5.
→ .encriptador gera [arq].leo, que são cifras de cesar e a chave é o arg passado, deleta os arquivos que nã são .leo.

Como desfazer:
1) desfaça a encriptação:

./encriptador -5

2) renomeie os arquivos:

```
for i in $(ls $USER | grep .leo.leo); do
        mv $USER/${i} $USER/${i:0:(${#_i}-8)}
done
```

3) delete os lixos (.leo):

```
rm $USER/*.leo
```

Obs: existe um “script”, se deixar ele na mesma pasta em que ‘tag’ foi rodado, e rodar o script, ele conserta.