

Senhas em Sistemas Linux

+ Nas estruturas mais antigas de Linux, o arquivo com usuários e hashes de senhas era o /etc/passwd

```
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
bind:x:105:113::/var/cache/bind:/bin/false
postfix:x:106:115::/var/spool/postfix:/bin/false
ftp:x:107:65534::/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/false
mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false
distccd:x:111:65534:::/bin/false
user:x:1001:1001:just a user,111,,,:/home/user:/bin/bash
service:x:1002:1002::,/home/service:/bin/bash
telnetd:x:112:120::/nonexistent:/bin/false
proftpd:x:113:65534::/var/run/proftpd:/bin/false
statd:x:114:65534::/var/lib/nfs:/bin/false
uftp:x:1003:1003::/home/vsftp:/bin/sh
ricardo:x:1004:1004:,y,Y,:/home/ricardo:/bin/bash
penelope:x:1005:1005::/home/:/bin/bash
hacker:x:1006:1006::,/home/hacker:/bin/bash
```

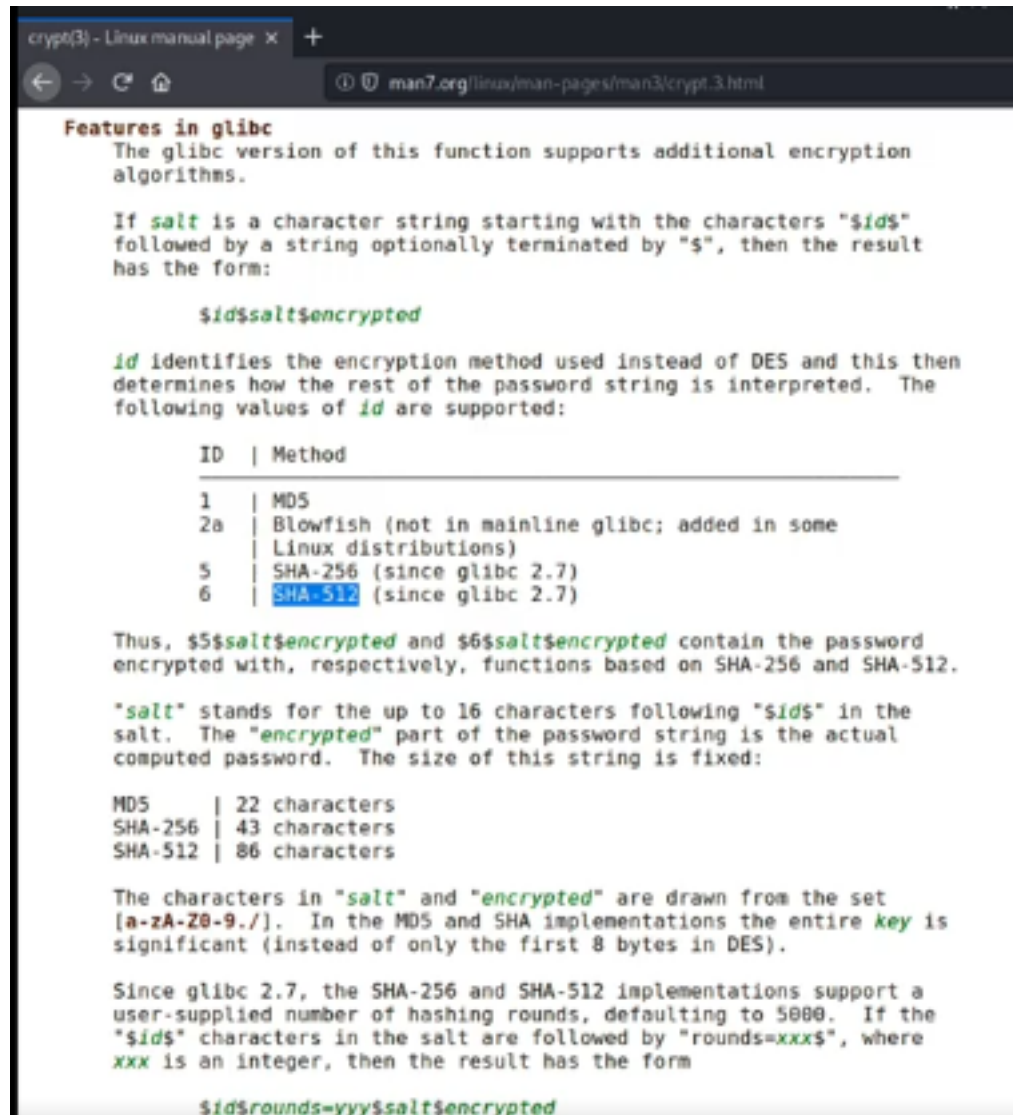
→ Esse "x" depois do usuário ricardo ou penelope (no final)

indica que o acesso aos hashes se dá somente pelo usuário root. Em outras versões, o hash apareceria sem escrúpulos

→ O nome do arquivo que acessa essas informações por meio do

root é o /etc/shadow

+ Uma vez encontrado, veremos que o hash é de um formato diferente.
Isso acontece pelo método encriptográfico do linux



crypt(3) - Linux manual page x +

man7.org/linux/man-pages/man3/crypt.3.html

Features in glibc

The glibc version of this function supports additional encryption algorithms.

If `salt` is a character string starting with the characters `"id"` followed by a string optionally terminated by `"$"`, then the result has the form:

`idsalt$encrypted`

`id` identifies the encryption method used instead of DES and this then determines how the rest of the password string is interpreted. The following values of `id` are supported:

ID	Method
1	MD5
2a	Blowfish (not in mainline glibc; added in some Linux distributions)
5	SHA-256 (since glibc 2.7)
6	SHA-512 (since glibc 2.7)

Thus, `5salt$encrypted` and `$6$salt$encrypted` contain the password encrypted with, respectively, functions based on SHA-256 and SHA-512.

`"salt"` stands for the up to 16 characters following `"id"` in the salt. The `"encrypted"` part of the password string is the actual computed password. The size of this string is fixed:

Method	Length
MD5	22 characters
SHA-256	43 characters
SHA-512	86 characters

The characters in `"salt"` and `"encrypted"` are drawn from the set `[a-zA-Z0-9./]`. In the MD5 and SHA implementations the entire `key` is significant (instead of only the first 8 bytes in DES).

Since glibc 2.7, the SHA-256 and SHA-512 implementations support a user-supplied number of hashing rounds, defaulting to 5000. If the `"id"` characters in the salt are followed by `"rounds=xxx$"`, where `xxx` is an integer, then the result has the form

`idrounds=yyy$salt$encrypted`

→ `idsalt$encrypted`

+ Para gerar o padrão desses hashes, devemos prosseguir com o [openssl](#) como podemos ver no lab05 dessa semana