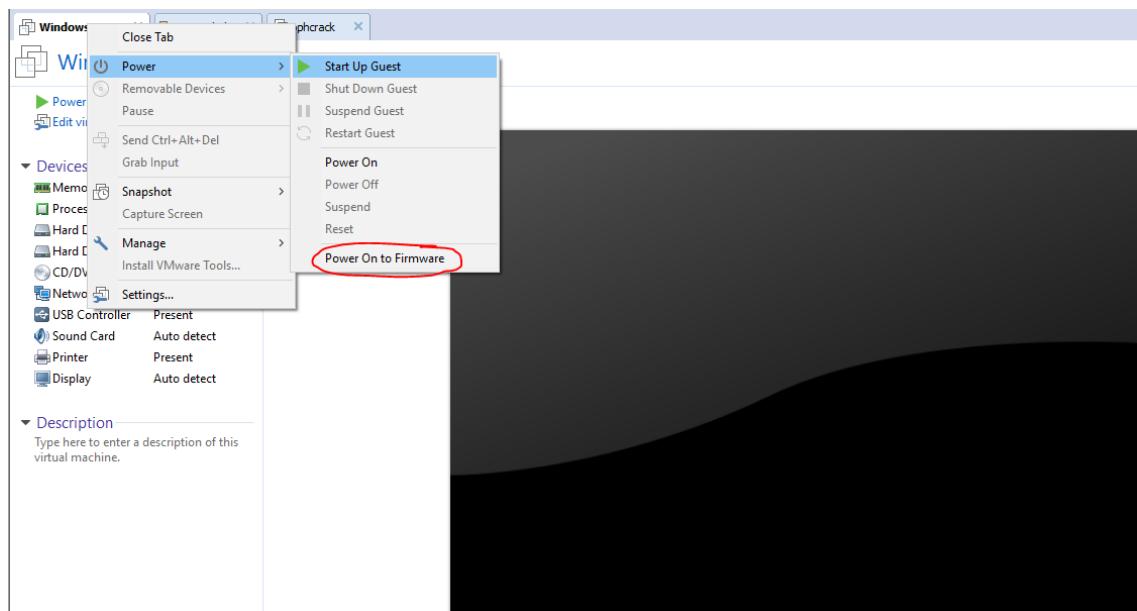


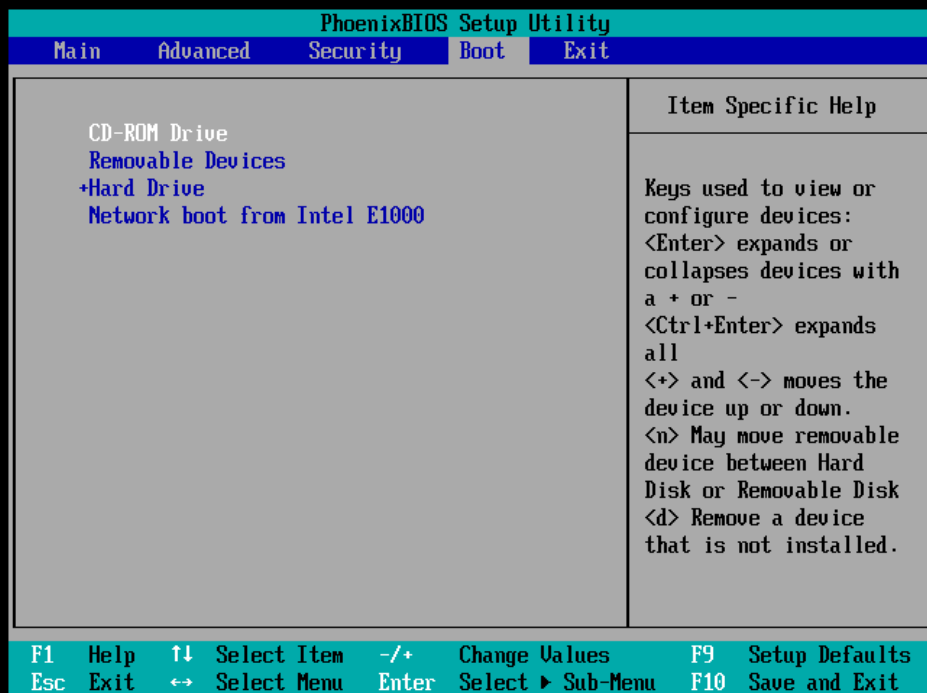
Contenido

Descriptación contraseñas de windows.....	1
Descriptación contraseñas shadow de Linux	5
Compilacion John:	6

Des encriptación contraseñas de windows

Lo primero que vamos a hacer es arrancar cargar la iso de ophcrack en el lector de cd (virtual) de vmware (en la máquina que tiene el windows del cual queremos sacar las contraseñas), tras esto, arrancaremos y nos meteremos en la BIOS para modificar el orden de arranque de los dispositivos disponibles, poniendo como primera opción el lector de cdrom



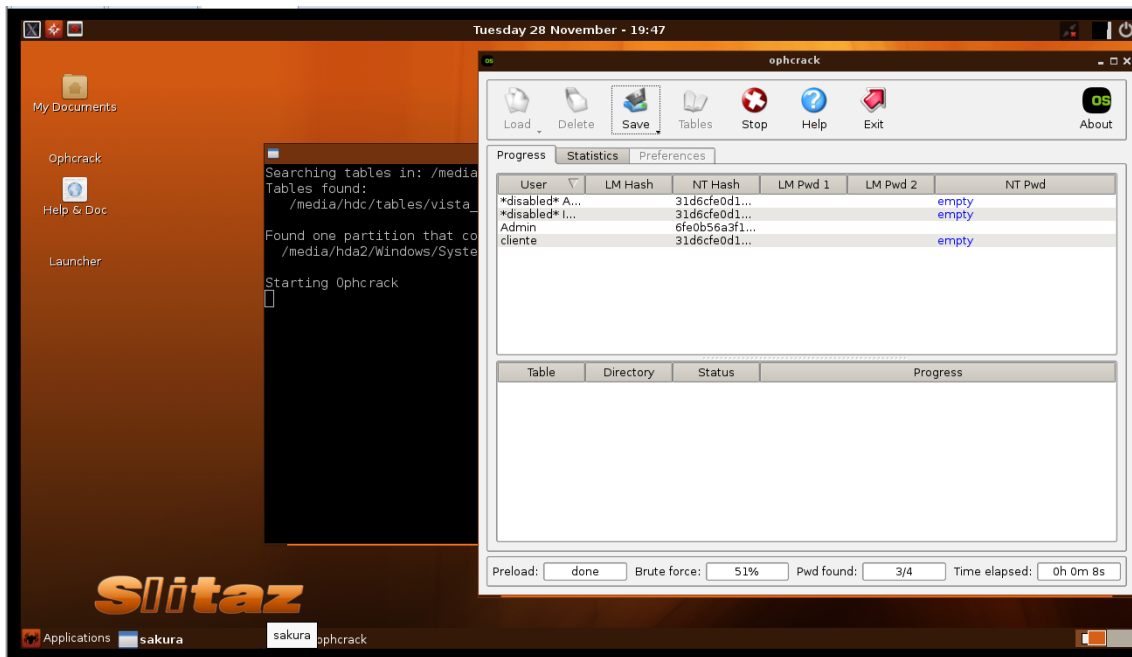


Luego salvaremos y al reiniciar veremos el menú de arranque de ophcrack

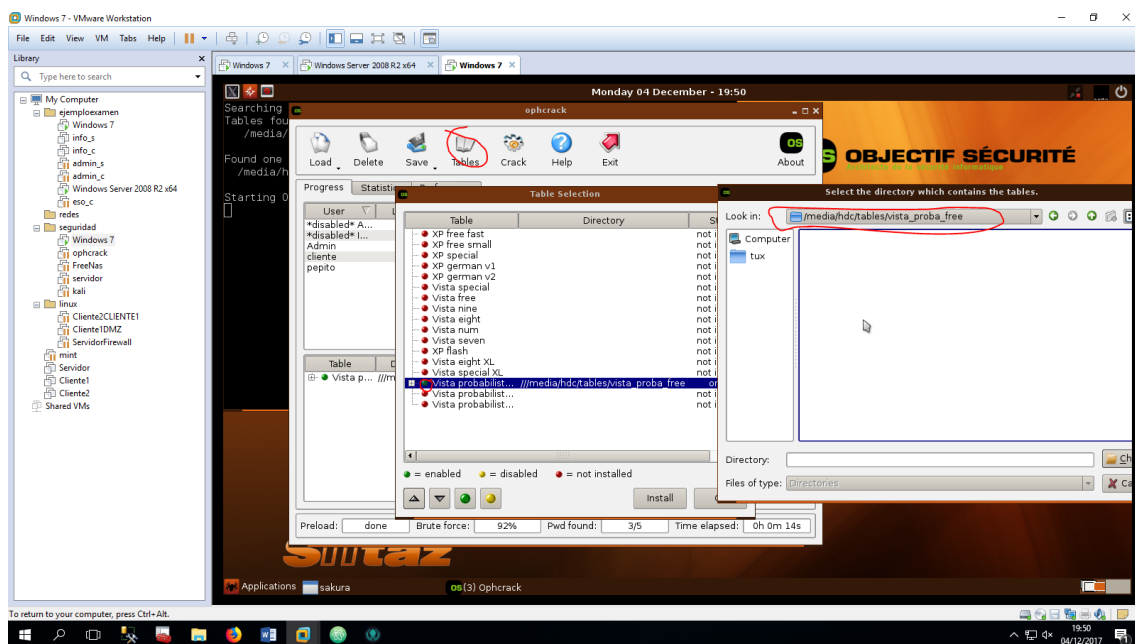


Seleccionaremos la opción por defecto y dejaremos que el sistema arranque.

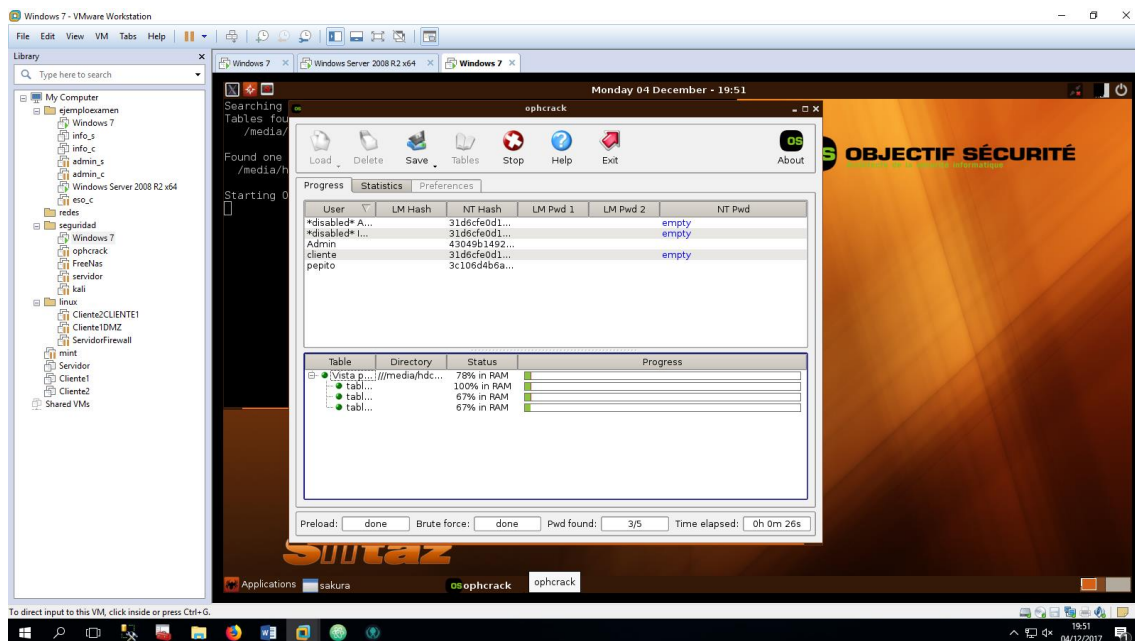
Al arrancar, vemos que automáticamente ophcrack detecta los usuarios del sistema



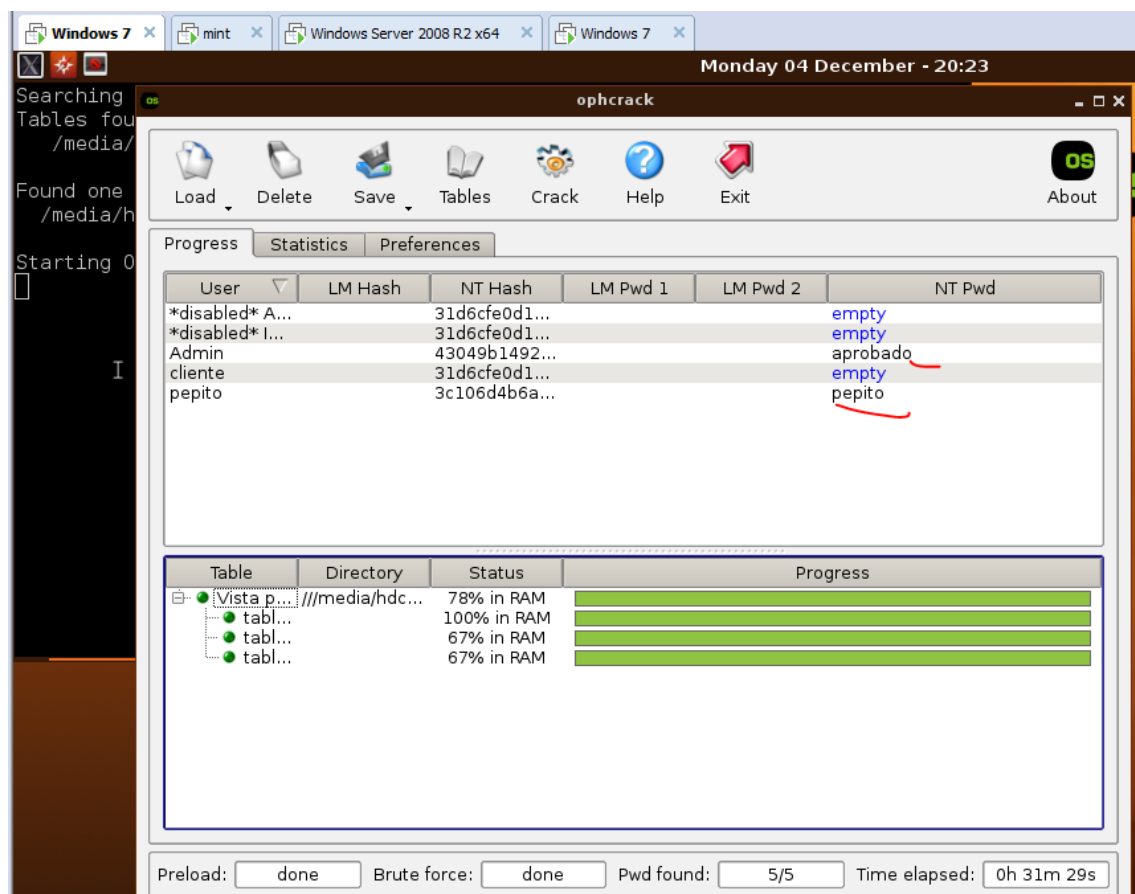
Ahora, tendremos que ir instalando las tablas necesarias para qué ophcrack, para esto, pincharemos en tablas, luego en install y buscaremos las tablas (previamente descargadas)



Ahora pincharemos en crack y esperamos



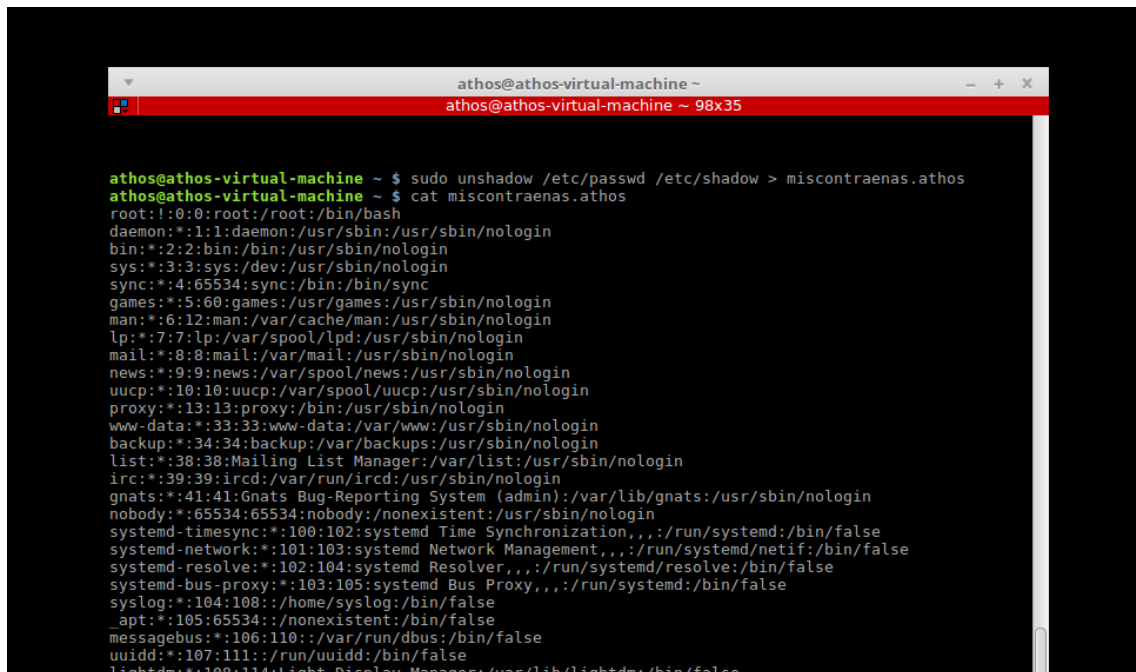
Si todo va bien nos sacara la contraseña.



Como nota añadiré, que para tener más oportunidades de que ophcrack descifre las contraseñas conviene que instalemos el mayor número de tablas que podamos.

Des encriptación contraseñas shadow de Linux

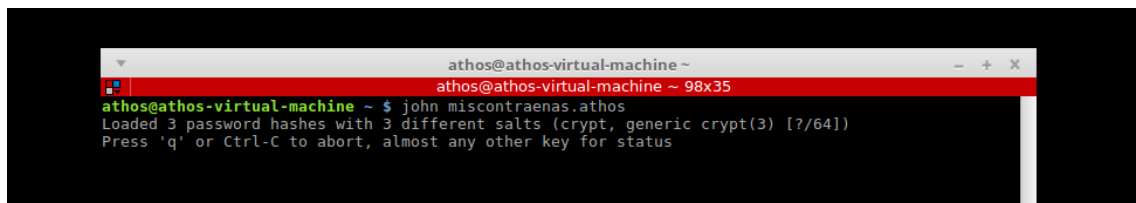
Lo primero vamos a unir los archivos /etc/passwd y /etc/shadow para que cuando nos saque las contraseñas saber de qué usuario es cada una.



```
athos@athos-virtual-machine ~  
athos@athos-virtual-machine ~ 98x35  
  
athos@athos-virtual-machine ~ $ sudo unshadow /etc/passwd /etc/shadow > miscontraenas.athos  
athos@athos-virtual-machine ~ $ cat miscontraenas.athos  
root:!:0:0:root:/root:/bin/bash  
daemon:*:1:1:daemon:/usr/sbin:/usr/sbin/nologin  
bin:*:2:2:bin:/bin:/usr/sbin/nologin  
sys:*:3:3:sys:/dev:/usr/sbin/nologin  
sync:*:4:65534:sync:/bin:/bin/sync  
games:*:5:60:games:/usr/games:/usr/sbin/nologin  
man:*:6:12:man:/var/cache/man:/usr/sbin/nologin  
lp:*:7:7:lp:/var/spool/lpd:/usr/sbin/nologin  
mail:*:8:8:mail:/var/mail:/usr/sbin/nologin  
news:*:9:9:news:/var/spool/news:/usr/sbin/nologin  
uucp:*:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin  
proxy:*:13:13:proxy:/bin:/usr/sbin/nologin  
www-data:*:33:33:www-data:/var/www:/usr/sbin/nologin  
backup:*:34:34:backup:/var/backups:/usr/sbin/nologin  
list:*:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin  
irc:*:39:39:ircd:/var/run/ircd:/usr/sbin/nologin  
gnats:*:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin  
nobody:*:65534:65534:nobody:/nonexistent:/usr/sbin/nologin  
systemd-timesync:*:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false  
systemd-network:*:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false  
systemd-resolve:*:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false  
systemd-bus-proxy:*:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false  
syslog:*:104:108:./home/syslog:/bin/false  
_apt:*:105:65534:./nonexistent:/bin/false  
messagebus:*:106:110:./var/run/dbus:/bin/false  
uidd:*:107:111:./run/uidd:/bin/false  
lightdm:*:108:114:Light Display Manager:/var/lib/lightdm:/bin/false
```

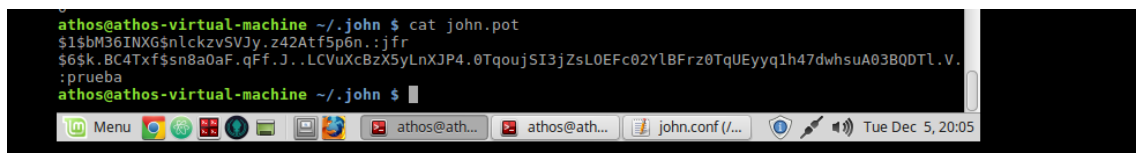
Ahora vamos a probar a ejecutar John pasando como parámetro únicamente el archivo de hashes (miscontraenas.athos)

Vemos que ha encontrado tres usuarios de los cuales podemos sacar las contraseñas



```
athos@athos-virtual-machine ~  
athos@athos-virtual-machine ~ 98x35  
  
athos@athos-virtual-machine ~ $ john miscontraenas.athos  
Loaded 3 password hashes with 3 different salts (crypt, generic crypt(3) [?/64])  
Press 'q' or Ctrl-C to abort, almost any other key for status
```

Si vamos al archivo John.pot dentro de la carpeta oculta John veremos que tenemos las contraseñas que nos vaya sacando



```
athos@athos-virtual-machine ~/.john $ cat john.pot  
$1$bM36INXG$nlckzvSV3y.z42Atf5p6n.:jfr  
$6$K.BC4Txf$sn8a0aF.qFf.J..LCVuxCzX5yLnXJP4.0TqoujSI3jZsL0EFc02YlBFRz0TqUEyyq1h47dwhsuA03BQDTL.V.  
:prueba  
athos@athos-virtual-machine ~/.john $
```

Mientras esperamos a que saque las contraseñas, he mirado el uso de procesadores del equipo y me encuentro lo siguiente.

```
athos@athos-virtual-machine ~/john
athos@athos-virtual-machine ~/john 98x35

1 [ | 0.0%] Tasks: 99, 215 thr; 2 running
2 [ | 1.3%] Load average: 1.00 1.34 1.81
3 [ | 1.3%] Uptime: 01:49:51
4 [ | 100.0%]
Mem [ | 668M/2.92G]
Swp [ | 0K/2.00G]

  PID USER      PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
  ---
 9326 athos      39   19 15752  2688  2152  R   100.0  0.1   6:11.54 john miscontraenas.athos
1287 root         20    0 312M 53516 30492  S   0.7   1.7   0:26.20 /usr/lib/xorg/Xorg -core :0 -seat
2260 athos      20    0 548M 48464 34836  S   0.7   1.6   0:15.19 compiz --replace
```

De los cuatro, solo está usando uno. Esto es porque por defecto John viene compilado para usar solo uno de los procesadores. Podríamos compilarlo nosotros para que use los cuatro.

Compilación John:

Lo primero es ir a la página oficial de John y descargarnos la última versión (gratuita ya que también hay versión de pago)

John the Ripper is a fast password cracker, currently available for many flavors of Unix, Windows, DOS, and OpenVMS. Its primary purpose is to detect Unix passwords. Besides several crypt(3) password hash types most commonly found on various Unix systems, supported out of the box are Windows L hashes, plus lots of other hashes and ciphers in the community-enhanced version.

Follow us on Twitter

John the Ripper is free and Open Source software, distributed primarily in source code form. If you would rather use a commercial product tailored for a specific operating system, please consider [John the Ripper Pro](#), which is distributed primarily in the form of "native" packages for the target operating system and in general is meant to be easier to install and use while delivering optimal performance.

Proceed to [John the Ripper Pro](#) homepage for your OS:

- [John the Ripper Pro for Linux](#)
- [John the Ripper Pro for Mac OS X](#)
- On Windows, consider [Hash Suite](#) (developed by a contributor to John the Ripper)
- On Android, consider [Hash Suite Droid](#)

Download one of the latest **official free versions** ([release notes](#)):

- [John the Ripper 1.8.0](#) (sources, tar.xz, 4.3 MB) and its [signature](#)
- [John the Ripper 1.8.x extra charset files archive](#) (tar.xz, 4.5 MB) and its [signature](#)
- [John the Ripper 1.8.0](#) (sources, tar.gz, 5.2 MB) and its [signature](#)
- [John the Ripper 1.7.9](#) (Windows binaries, ZIP, 2029 KB) and its [signature](#)

Download the latest **community-enhanced** version ([release notes](#), [patch to build 1.8.0-jumbo-1 on non-x86](#)):

- Latest development source code in [GitHub repository](#) (tar.gz, ZIP)
- [John the Ripper 1.8.0-jumbo-1](#) (sources, tar.xz, 23 MB) and its [signature](#)
- [John the Ripper 1.8.0-jumbo-1](#) (sources, tar.gz, 30 MB) and its [signature](#)
- [John the Ripper 1.8.0-jumbo-1](#) (Windows binaries, ZIP, 34 MB) and its [signature](#)

This version integrates lots of contributions from the community, including support for a hundred of additional hash and cipher types (including popular ones such as NTLM, bcrypt, etc.)

Nos metemos en la carpeta src y editamos el archivo Makefile

```
athos@athos-virtual-machine ~/john-1.8.0/src $ vim Makefile
```

Tendremos que buscar las líneas OMPFLAGS y descomentarlas

```
athos@athos-virtual-machine ~/john-1.8.0/src
athos@athos-virtual-machine ~/john-1.8.0/src 161x45
#
# This file is part of John the Ripper password cracker,
# Copyright (c) 1996-2013 by Solar Designer
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted.
#
# There's ABSOLUTELY NO WARRANTY, express or implied.
#
CC = gcc
AS = $(CC)
LD = $(CC)
CPP = $(CC)
CP = cp
LN = ln -sf
RM = rm -f
TR = tr
SED = sed
NULL = /dev/null
CPPFLAGS = -E
OMPFLAGS =
# gcc with OpenMP
OMPFLAGS = -fopenmp
# gcc with OpenMP on 32-bit x86 with SSE2
OMPFLAGS = -fopenmp -msse2
# Mac OS X (llvm-gcc) with OpenMP
OMPFLAGS = -fopenmp -D FORTIFY_SOURCE=0
# Sun Studio with OpenMP (set the OMP_NUM_THREADS env var at runtime)
OMPFLAGS = -xopenmp
CFLAGS = -c -Wall -Wdeclaration-after-statement -O2 -fomit-frame-pointer $(OMPFLAGS)
# CFLAGS for use on the main john.c file only
CFLAGS_MAIN = $(CFLAGS)
ASFLAGS = -c $(OMPFLAGS)
LDFLAGS = -s $(OMPFLAGS)
OPT_NORMAL = -funroll-loops
# Remove the "-Os" if you're using an ancient version of gcc
OPT_INLINE = -Os -funroll-loops -finline-functions
```

Ahora toca compilar con el comando make linux-x86-64

```
athos@athos-virtual-machine ~/john-1.8.0/src
athos@athos-virtual-machine ~/john-1.8.0/src 161x45
athos@athos-virtual-machine ~/john-1.8.0/src $ vim Makefile
athos@athos-virtual-machine ~/john-1.8.0/src $ make linux-x86-64
ln -sf x86-64.h arch.h
make ../run/john ../run/unshadow ../run/unafs ../run/unique \
JOHN_OBJS="DES_fmt.o DES_std.o DES_bs.o DES_bs_b.o BSDI_fmt.o MD5_fmt.o MD5_std.o BF_fmt.o BF_std.o AFS_fmt.o LM_fmt.o trip_fmt.o dummy.o batch.o bench.o
charset.o common.o compiler.o config.o cracker.o crc32.o external.o formats.o getopt.o idle.o inc.o john.o list.o loader.o logger.o math.o memory.o misc.o optio
ns.o params.o path.o recovery.o rpp.o rules.o signals.o single.o status.o tty.o wordlist.o unshadow.o unafs.o unique.o c3_fmt.o x86-64.o" \
CFLAGS="-c -Wall -Wdeclaration-after-statement -O2 -fomit-frame-pointer -fopenmp -msse2 -DHAVE_CRYPT" \
LDFLAGS="-s -fopenmp -msse2 -lcrypt"
make[1]: Entering directory '/home/athos/john-1.8.0/src'
gcc -c -Wall -Wdeclaration-after-statement -O2 -fomit-frame-pointer -fopenmp -msse2 -DHAVE_CRYPT -funroll-loops DES_fmt.c
gcc -c -Wall -Wdeclaration-after-statement -O2 -fomit-frame-pointer -fopenmp -msse2 -DHAVE_CRYPT -funroll-loops DES_std.c
gcc -c -Wall -Wdeclaration-after-statement -O2 -fomit-frame-pointer -fopenmp -msse2 -DHAVE_CRYPT -funroll-loops DES_bs.c
gcc -c -Wall -Wdeclaration-after-statement -O2 -fomit-frame-pointer -fopenmp -msse2 -DHAVE_CRYPT -Os -funroll-loops -finline-functions DES_bs_b.c
gcc -c -Wall -Wdeclaration-after-statement -O2 -fomit-frame-pointer -fopenmp -msse2 -DHAVE_CRYPT -funroll-loops BSDI_fmt.c
gcc -c -Wall -Wdeclaration-after-statement -O2 -fomit-frame-pointer -fopenmp -msse2 -DHAVE_CRYPT -funroll-loops MD5_fmt.c
gcc -c -Wall -Wdeclaration-after-statement -O2 -fomit-frame-pointer -fopenmp -msse2 -DHAVE_CRYPT -funroll-loops MD5_std.c
gcc -c -Wall -Wdeclaration-after-statement -O2 -fomit-frame-pointer -fopenmp -msse2 -DHAVE_CRYPT -funroll-loops BF_fmt.c
gcc -c -Wall -Wdeclaration-after-statement -O2 -fomit-frame-pointer -fopenmp -msse2 -DHAVE_CRYPT -funroll-loops BF_std.c
```

En la carpeta run nos habrá creado los nuevos ejecutables

```
athos@athos-virtual-machine ~/john-1.8.0/run
athos@athos-virtual-machine ~/john-1.8.0/run 161x45
athos@athos-virtual-machine ~/john-1.8.0/run $ ls
ascii.chr digits.chr john john.conf lm ascii.chr mailer makechr password.lst relbench unafs unique unshadow
athos@athos-virtual-machine ~/john-1.8.0/run $
```

Ahora ejecutamos John como lo hicimos anteriormente y comprobamos los cpus

```
athos@athos-virtual-machine ~/john-1.8.0/run
athos@athos-virtual-machine ~/john-1.8.0/run 161x45
athos@athos-virtual-machine ~/john-1.8.0/run $ ./john miscontraenas.athos
Loaded 3 password hashes with 3 different salts (crypt, generic crypt(3) [?/64])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
prueba (prueba)
1g 0:00:00:09 38% 1/3 0.1006g/s 1042p/s 1043c/s 1043C/s [jfkdljsjfklafdjakfd..^pjfkdljsjfkla
```

Como vemos, nos marca que está funcionando con cuatro hilos, y podemos ver que el número de contraseñas por segundo C/s ha aumentado X4

Ahora comprobamos los núcleos con htop

```
athos@athos-virtual-machine ~
athos@athos-virtual-machine ~ 161x45
1 [|||||] 98.0% Tasks: 104, 337 thr; 5 running
2 [|||||] 96.7% Load average: 2.60 1.25 1.46
3 [|||||] 98.7% Uptime: 01:59:42
4 [|||||] 100.0%
Mem[|||||] 1.08G/2.92G
Swp[|||||] 0K/2.00G

  PID USER      PRI  NI  VIRT   RES   SHR  S CPU% MEM%   TIME+  Command
10498 athos      20    0  229M  2436  2024  R 99.2  0.1  4:14.93 ./john miscontraenas.athos
10492 athos      20    0  229M  2436  2024  R 99.2  0.1  1:03.73 ./john miscontraenas.athos
10491 athos      20    0  229M  2436  2024  R 97.9  0.1  1:03.74 ./john miscontraenas.athos
10493 athos      20    0  229M  2436  2024  R 97.2  0.1  1:03.72 ./john miscontraenas.athos
1287 root         20    0  374M  74880 46888  S  2.0  2.4  0:34.29 /usr/lib/xorg/Xorg -core :0 -seat seat0 -auth /var/run/lightdm/root/:0 -nolisten
10526 athos      20    0  27916 4532  3228  R  0.7  0.1  0:00.06 htop
2854 athos      20    0  440M  50092 30904  S  0.7  1.6  0:04.09 /usr/bin/python /usr/bin/terminator
1336 mysql        20    0 1217M 1511M 16416  S  0.0  5.1  0:00.18 /usr/sbin/mysqld
2260 athos      20    0  549M 48764 34968  S  0.0  1.6  0:18.85 compiz --replace
9720 athos      20    0 2180M 302M  131M  S  0.0 10.1  0:22.34 /usr/lib/firefox/firefox
1 root         20    0  117M  6052  4024  S  0.0  0.2  0:01.67 /sbin/init splash
580 root         20    0 31028 3104  2688  S  0.0  0.1  0:00.37 /lib/systemd/systemd-journald
594 root         20    0  100M  1652  1424  S  0.0  0.1  0:00.00 /sbin/lvmtool -f
596 root         20    0  154M  268  4  S  0.0  0.0  0:00.00 vmware-vmblock-fuse /run/vmblock-fuse -o rw,subtype=vmware-vmblock,default_perm
597 root         20    0  154M  268  4  S  0.0  0.0  0:00.00 vmware-vmblock-fuse /run/vmblock-fuse -o rw,subtype=vmware-vmblock,default_perm
595 root         20    0  154M  268  4  S  0.0  0.0  0:00.00 vmware-vmblock-fuse /run/vmblock-fuse -o rw,subtype=vmware-vmblock,default_perm
```

De esta manera, hemos aumentado el rendimiento cuatro veces más.