The reference sheet

Author: Stefan Pettersson, last updated: 2010-07-21, latest version at http://www.bigpointyteeth.se/.

I. Bourne shell

1 Pre-defined variables.

```
$n     positional parameter n (n <= 9)
${n}   positional parameter n
$*     all positional parameters
$@     all positional parameters
"$*"   equivalent to "$1 $2 ..."
"$@"   equivalent to "$1" "$2" … (preferrable)
$#     number of positional parameters
$-     options to shell or by set
$?     return code by last program
$$     pid of current shell
$!     pid of last background program
$_     value of last positional parameter of last
       command

${#var}      length of $var
"${@: -1}"   the last positional parameter
```

2 Shell redirection.

```
n> file   redirect standard output (or n) to file
n>| file  same, but override the -C option
n>> file  append standard output (or n) to file
n< file   redirect standard input (or n) from file
n<&m      duplicate standard input (or n) from
          file descriptor m
n<&-      close standard input (or n)
n>&m      duplicate standard output (or n) to m
n>&-      close standard output (or n)
n<> file  open file for reading and writing on
          standard input (or n)
```

3 Guard against undefined or null variables.

```
$ set -o errexit   # exit on status != 0
$ set -o nounset   # exit when using unset var
$ echo ${X:?"Error: X is undefined"}/tmp
Error: X is undefined/tmp
$ X=muppet
$ echo ${X:?"X is undefined"}/tmp
muppet/tmp
$ echo X is ${X:-default}
X is default
$ X=poop
$ echo X is ${X:-default}
X is poop
$ echo ${X:+changeifdefined}
$ X=spam
$ echo ${X:+changeifdefined}
changeifdefined
$ echo X is ${X:=redefine}
X is redefine
$ echo X is $X
X is redefine
```

4 Boolean operators. Execute if success and execute if failure, respectively.

```
$ true && echo this is executed
$ false && echo this is not executed
$ false || echo this is executed
$ true || echo this is not executed
```

5 Various loop constructs.

```
$ for z in *.zip; do unzip $z; done
$ for ((i=1;i<12;i++)); do echo $i; done
$ for i in $(seq 11); do echo $i; done
$ while read line; do echo $line; done < file
$ while [ $n -lt 5 ]; do n=$[$n+1]; done
$ while test $n -lt 5; do n=$(($n-1)); done
```

6 Evaluation modifiers for **test**(1) and **[**.

```
EXPR1 -a EXPR2  both are true
EXPR1 -o EXPR2  one is true
-n STR          string length is nonzero
STR             string length is nonzero
-z STR          string length is zero
STR1 = STR2     strings are equal
STR1 != STR2    strings are different
INT1 -eq INT2   ==
INT1 -ne INT2   !=
INT1 -gt INT2   >
INT1 -ge INT2   >=
INT1 -lt INT2   <
INT1 -le INT2   <=
-d FILE         is a directory
-e FILE         exists
-f FILE         exists and is a regular file
-L FILE         is a symbolic link
-O FILE         effective UID is owner
-s FILE         size greater than zero
-r FILE         is readable
-w FILE         is writable
-x FILE         is executable
```

7 Expression evaluation examples.

```
$ test -r /etc/passwd && echo readable
readable
$ test -w /etc/passwd && echo writable
$ test -r /bin/ls -a -x /bin/ls && echo both
both
$ if grep -q root /etc/passwd; then echo yeah; fi
yeah
$ if grep -q beer /etc/passwd; then echo yeah; fi
$ (grep -q str file || echo no str found) | sort
no str found
```

II. UNIX tools

Date, crontab and at

8 Convert from epoch time to human-readable date.

```
$ date +%s
1264683793
$ date -d "UTC 1970-01-01 1264683793 secs"
Thu Jan 28 14:03:13 CET 2010
```

9 Convert from human-readable date to epoch time.

```
$ date +%s -d "Fri Apr 24 13:14:39 CDT 2009"
1240596879
$ date +%s -d "2010-01-28 14:28:50"
1264685330
```

10 GNU time string examples (**date -d STRING**).

```
YYYY-MM-DD HH:MM:SS
YYYY-MM-DD
now
tomorrow
yesterday
last-sunday
last-week
last-month
last-year
next-sunday (alias: sunday)
next-week
next-month
next-year
```

11 Scheduling syntax for *crontab*(5).

```
* * * * * <cmd>
| | | | |.day of week (0-7, sunday = {0,7})
| | | |...month (1-12)
| | |.....day of month (1-31)
| |.......hour (0-23)
|.........minute (0-59)
```

12 Examples for *crontab*(5).

```
* * * * *           every minute
50 23 * * *         23:50 every day
*/10 * * * *        every 10 minutes
10 4 1 * *          04:10 the 1st every month
10 4 * * 1          04:10 every monday
0 4 * * 1,2,3,4,5   04:00 every work day
```

13 Examples for **at**(1).

```
$ at 23:50
at> echo "Shutting down in 10 minutes!" | wall
at> ^D
$ tty; whoami
/dev/pts/7
user
$ at now +10 minutes
at> echo spagetti finished | write user pts/7
at> ^D
```

Find

14 Find files of a certain type where *t* is one of *f*ile, *d*irectory, sym*l*ink, *s*ocket, *p*ipe, *b*lock or *c*har.

```
$ find / -type t
```

15 Find files that I can write to.

```
$ find / \( -user $USER -perm +u=w \) -o \
> \( -perm +o=w \)
```

16 Find hidden (dot) files.

```
$ find / -name ".*" -print -xdev
```

17 Find files that are setuid/setgid.

```
$ find / -perm -4000 -print 2> /dev/null
$ find / -perm -2000 -print 2> /dev/null
```

18 Find files that are not owned.

```
$ find / -type f -nouser -print 2> /dev/nul
$ find / -type d -nouser -print 2> /dev/null
$ find / -type f -nogroup -print 2> /dev/null
$ find / -type d -nogroup -print 2> /dev/null
```

19 Find files that are world-writable.

```
$ find / -perm +002 -type f -print 2> /dev/null
$ find / -perm +002 -type d -print 2> /dev/null
```

20 Recursively chmod a directory tree.

```
$ find . -type d -print0 | xargs -0 chmod 755
$ find . -type f -print0 | xargs -0 chmod 644
```

21 Find files larger than, less than and exactly equal to some size. The suffix can be c for byte, k for kb, M for Mb or G for Gb.

```
$ find / -size +100M
$ find / -size -1k
$ find / -size 96583c
```

22 Find all files that have been *a*ccessed, had its content *m*odified or inode *c*hanged (e.g. owner) within the last *n* days.

```
# find / -type f -atime -n
# find / -type f -mtime -n
# find / -type f -ctime -n
```

23 Find all files that have been modified after *file*.

```
$ find / -newer file
```

troubleshooting

24 vmstat

25 iostat

26 mpstat

27 iptraf

28 dmidecode

29 taskset

sar

The examples with **sar**(1) below assumes that CWD is where the log files are placed, */var/log/sysstat/* by default.

30 Install and enable accounting (Ubuntu).

```
# aptitude install sysstat
# ed /etc/default/sysstat <<< \
> $',s/^ENABLED=.*/ENABLED="false"/\n w'
```

31 Sample network usage to stdout every five seconds.

```
# sar -n DEV 5 | egrep "IFACE|eth0"
```

32 Sample full system usage to *file* in the background every ten seconds for one minute (six times) and check memory usage for that period.

```
# sar -o file 10 6  >/dev/null 2>&1 &
# sar -f file -r
```

33 To run sar in the background follow the instructions in the manuals of sa1 and sa2 on how to add them to crontab.

34 Print CPU usage for the 11th around noon.

```
# sar -f sa11 -s 10:00:00 -e 14:00:00
```

35 Print eth0 network usage for today up until now.

```
# sar -f -n DEV | egrep "eth0|IFACE"
```

ISO filesystems

36 Create an ISO file from directory structure.

```
$ mkisofs -f -R -r -l -J -ofoobar.iso isofolder/
```

37 Mount ISO file system.

```
# mount -o loop /mnt foobar.iso
```

38 Burn ISO file with **cdrecord**.

```
# cdrecord --devices
# cdrecord dev=ATAPI:0,0,0 foobar.iso
```

39 Rip a CD image.

```
# dd if=/dev/cdrom of=foobar.iso
```

Miscellaneous

40 Start a typescript and clean the output of ^H, ^M and similar crap.

```
$ script hacklog.0
...
$ exit
$ col -bx < hacklog.0 > hacklog.1
```

41 Output manual as formatted text.

```
$ man pwd | col -b > pwd_manual.txt
```

42 Prepend line numbers on file.

```
$ nl file
$ cat -n file
$ awk '{print NR,$0}' file
```

43 Make a HTML directory index with lynx.

```
$ lynx -source . > index.html
```

44 Set a specific time stamp for *targetfile*.

```
$ touch -t ccyymmddhhmm targetfile
```

45 Use an existing file as reference for *targetfile*'s timestamp.

```
$ touch -r referencefile targetfile
```

46 Retrieving an unlinked opened file

```
# tcpdump -nnw " " &
# ls -i " "
1918711
# unlink " "
# lsof +L1  # list opened files with < 1 link
COMMAND PID  USER FD ... NODE    NAME
tcpdump 2530 root 4w     1918711 /tmp/ (deleted)
# cd /proc/2530/fd
# cat 4 > /tmp/recovered_file
```

47 Download *url* to *file* over HTTP without output.

```
$ wget -q -O file url
$ curl -s -o file url
$ lynx -source url > file  (watch out for header)
$ links -source url > file
$ elinks -source url > file
$ w3m -dump_source url > file
```

48 **patch**(1) and **diff**(1).

```
$ ls
one       one.new   two       two.new
$ diff -C 5 one one.new > patchfile
$ diff -C 5 two two.new >> patchfile
$ cmp one one.new
one one.new differ: char 32, line 2
$ cmp two two.new
two two.new differ: char 11, line 2
$ patch < patchfile
$ cmp one one.new
$ cmp two two.new
$ ls *.orig
one.orig  two.orig
```

49 Ping sweeps.

```
$ x=1; while [ $x -lt "255" ]; do ping -c 1 \
> 10.0.0.$x | grep "bytes from" |\
> awk '{print $4, " up"}'; let x++; done
```

50 Remove boot loader

```
# dd if=/dev/zero of=/dev/sda bs=446 count=1
```

III. Netfilter

51 Simple local firewall.

```
# iptables -P INPUT ACCEPT
# iptables -F
# iptables -A INPUT -m state –state \
> RELATED,ESTABLISHED -j ACCEPT
# iptables -A INPUT -p tcp --dport 22 \
> -s 10.0.0.0/24 -j ACCEPT
# iptables -I INPUT 2 -p tcp --dport 80 \
> -s 10.0.0.0/8 -j ACCEPT
# iptables -A INPUT -p icmp --icmp ping-request \
> -j ACCEPT
# iptables -A INPUT -p icmp --icmp ping-reply \
> -j ACCEPT
# iptables -t filter -P DROP
# iptables-save > /etc/firewall.rules
# iptables -P ACCEPT
# iptables -F
# iptables-restore < /etc/firewall.rules
```

52 Redirect a port locally (10.0.0.1 is the local IP).

```
# iptables -t nat -A PREROUTING -d 10.0.0.1 \
> -p tcp --dport 80 -j REDIRECT --to-ports 8080
```

IV. Windows cmd.exe

53 Loop through files in current directory and lines in a text file.

```
C:\> for %G in (*) do command %G
C:\> for /F %G in (file.txt) do command %G
C:\> for /L %H in (1,1,254) do \
> ping -n 1 10.0.0.%H |\
> find "Reply" >> hostlist.txt
C:\> more hostlist.txt
```

54 Alternate data streams.

```
C:\> echo blah > example.txt
C:\> type nc.exe > example.txt:nc.exe
C:\> del nc.exe
C:\> start .\example.txt:nc.exe
```

55 List of **net** commands.

```
TBD
```

56 The Windows firewall via **netsh**.

```
C:\> netsh firewall show config
C:\> netsh firewall set opmode disable
C:\> netsh firewall add portopening TCP 80
webserver
C:\> netsh firewall delete portopening TCP 80
```

V. Screen

57 Command line options.

```
$ screen -list      list running sessions
$ screen -S name    start a new session name
$ screen -r name    reattach to session name
$ screen -D -R name detach and reattach to name
$ screen -x name    attach to screen w/o
                    deattaching others
$ screen -e^Vv      use V as command character
$ screen -L         start logging in all windows
```

58 Key bindings. Assuming "a" to be the command key.

```
^a ?       show help window
^a c       create new window (shell)
^a n       switch to next window
^a p       switch to previous window
^a [0-9]   switch to window number [0-9]
^a k       kill current window
^a A       rename current window
^a d       detach from screen session
^a h       save hardcopy of current window
^a H       start logging output of current window
^a ESC     enter copy mode (scroll)
```

59 Split windows.

```
^a S       split window horizontally
^a TAB     cycle between split windows
^a X       un-split current split window
^a Q       un-split all but current split window
```

VI. Editing

vim/ex

60 Short commands.

```
"Xyy   yank line into register X
"Xp    paste line from register X
%      jump to corresponding paranthesis
*      go to the next occurence of word
#      go to the previous occurence of word
dtX    delete forward to character X
dTX    delete backward to character X
^f     page down (forward)
^b     page up (back)
^e     scroll view down (static cursor)
^y     scroll view up (static cursor)
H      move cursor to top of screen (high)
M      move cursor to middle of screen
L      move cursor to bottom of screen (low)
g^g    count words in file
gqap   format paragraph text margins
guu    lower case whole line
gUU    upper case whole line
ggguG  lower case entire file
gggUG  upper case entire file
^a     increment number under cursor
^x     decrement number under cursor
>>     indent line
<<     un-indent line
K      go to man page of word under cursor
gf     open file name under cursor (goto file)
```

61 Moving around.

```
mX    place bookmark in register X
`X    go to bookmark in register X
``    go to position last edited
'.    go to line last edited
```

62 Binding actions to keys/commands/abbrev.

```
:ab name cmds   commmand mode abbreviation
:una name       remove command mode abbreviation
:iab name cmds  insert mode abbreviation
:iuna name      remove insert mode abbreviation
:iab xdate <c-r>=strftime("%Y-%m-%d")<cr>
:iab xtime <c-r>=strftime("%H:%M:%S")<cr>
```

63 Search and replace.

```
:s/old/new/     replace first old on line
:s/old/new/g    replace all old on line
:%s/old/new/    replace first old in file
:%s/old/new/g   replace all old in file
:%s/old/new/gc  confirm replace all old in file
```

64 Delete.

```
:/str/+d       delete the line below str
:.,$d          delete to final line (like dG)
:/foo/,/bar/d  delete from foo to bar (inclusive)
:/foo/,/bar/-d same but leave bar
:g/[12345]/d   delete all lines matching regex
:g!/[12345]/d  delete all lines not matching regex
```

65 Several substitutions in parallell.

```
$ ex file <<EOF
> %s/Nisse/Janne/g
> %s/Kalle/Gunnar/g
> %s/Fritz/Ludde/g
> w
> EOF
```

66 Multiple buffers.

```
:bn   change to next buffer
:bp   change to previous buffer
:wn   save and change to next buffer
:wp   save and change to previous buffer
```

67 Using macros.

```
qX    start recording a macro to register X
...   commands to be recorded
q     end recording
@X    apply macro in register X
@@    apply last macro again
10@X  apply macro in register X ten times
10@@  apply last macro ten times
```

68 Miscellaneous commands.

```
:r file            insert contents of file
:r !cmd            insert output from cmd
:$r file           insert file at the end
:0r file           insert file at the beginning
:run syntax/2html.vim generate coloured HTML text
:%s= *$==          clear end-of-line whitespace
:%s/\s\+$//        -"-
:v/./d             remove blank lines
:nohl              remove text highlights
:n,ms/^/#/         comment lines n to m
:'<,'>s/^/#/       comment marked lines
80istr<ESC>        repeat string 80 times
```

69 Managing multiple splitted windows.

```
^w s          split current window horizontally
:split file   open file in new horizontal window
^w v          split current window vertically
:vsplit file  open file in new vertical window
^w w          cycle through windows
^w {hjkl}     move to left/down/up/right window
              :map <C-H> <C-W>h
              :map <C-L> <C-L>l
^w _          maximize current split window
:edit file    start editing file in current window
:ls           list open (hidden) files
:q            close current window
^w q          close current window
:only         close all other windows
:qa           exit vim (with multiple windows)
```

70 Settings.

```
:syntax           turn on syntax highlighting
:set tw=n         text width
:set number       show line numbers
:set syntax=X     force syntax to X
:set noswapfile   do not create a .swp file
:set nobackup     do not create ~ backup files
:set scrolloff=3  keep 3 lines visible top/bottom
```

ed

The examples below uses the `$'string'` expansion and here-string of Bash. All three constructs below are equivalent.

```
$ ed file <<< $'/pattern/d\n w'
$ echo -e "/pattern/d\n w" | ed file
$ printf "/pattern/d\n w" | ed file
```

71 Delete first line matching "pattern" in *file*.

```
$ ed file <<< $'/pattern/d\n w'
```

72 Delete all lines matching "pattern" in *file*.

```
$ ed file <<< $'g/pattern/d\n w'
```

73 Replace "foo" with "bar" on all lines of *file*.

```
$ ed file <<< $',s/foo/bar/g\n w'
```

74 Set var to true in *config*.

```
$ ed config <<< $',s/^var=.*/var=true/\n w'
```

75 Replace "apples" with "pears" on all lines of *file* matching "fruit".

```
$ ed file <<< $'g/fruit/s/apples/pears/g\n w'
```

76 Move/copy lines 2-4 to follow line 6 of *file*.

```
$ ed file <<< $'2,4m6\n w'
$ ed file <<< $'2,4t6\n w'
```

77 Move/copy lines 2-4 to the end of *file*.

```
$ ed file <<< $'2,4m$\n w'
$ ed file <<< $'2,4t$\n w'
```

78 Append "string" to end of *file*.

```
$ ed file <<< $'$a\nstring\n.\n w'
```

79 Prepend "string" to beginning of *file*.

```
$ ed file <<< $'1i\nstring\n.\n w'
```

80 Append *file2* to end of *file1*.

```
$ ed file1 <<< $'$r file2\n w'
```

81 Prepend *file2* to beginning of *file1*.

```
$ ed file1 <<< $'0r file2\n w'
```

sed

82 Cut section from file (inclusive).

```
$ sed '/-BEGIN-/,/-END-/p' < file
```

83 awd

awk

84 Print the number of fields in each line and the line.

```
$ awk '{print NF ":" $0}' file
```

85 Print last field of each line.

```
$ awk '{print $NF}' file
```

86 Remove the last *n* fields (if there are more than *n* fields). E.g. remove trailing octets from IP address or extensions from file name.

```
$ awk -F. '{NF=NF-n; OFS="."; print}' file
$ awk -F. '{if (NF>n) NF=NF-n; OFS="."; print}'
```

87 Print every line with more than four fields.

```
$ awk 'NF > 4' file
```

88 Print every line with more than 8 characters. Good for password dictionaries for example.

```
$ awk 'length > 8' file
```

89 Switch first two fields of every line.

```
$ awk '{temp = $1; $1 = $2; $2 = temp}' file
```

90 Print all but second column.

```
$ awk '{ $2 = ""; print }' file
```

VII. Binary analysis tools

Shellcode

91 Disassemble raw shellcode.

```
$ ndisasm -u shellcode.egg
$ objdump -m i386 -b binary -D shellcode.egg
```

92 Create C hexadecimal escape sequence.

```
$ od -An -tx1 shellcode.egg |\
> sed -e 's/ /\\x/g' -e 's/.*/"&"/'
```

Analyzing ELF binaries

93 View ELF file header.

```
$ readelf -h file
```

94 View ELF program header.

```
$ readelf -l file
```

95 View ELF section header.

```
$ readelf -S file
$ objdump -h file
```

96 Disassemble a specific ELF section.

```
$ objdump -d -j .text file
```

97 Hexdump a specific ELF section.

```
$ readelf -x .rodata file
$ objdump -s -j .rodata file
```

HT Editor

98 General keybindings.

```
return           follow link (if applicable)
backspace        go back
space/F6         choose view mode
alt+[1-9]        select window
alt+0            select window list
^left/^right     scroll left/right
cursor keys      move around
pgup/pgdn        next/previous page
alt+S            toggle select
^ins/alt+c       copy
shift+ins/alt+v  insert
^del/alt+d       delete
shift+del/alt+x  cut
```

99 Window key bindings.

```
alt+F3/^w       close window
^F5             toggle resize/move mode
                (in resize/move mode only)
space           toggle resize/move mode
cursor keys     resize/move window
esc/return/^F5  leave resize/move mode
```

100 Analyser key bindings.

```
c    continue code analysis at cursor
f    follow dword ptr at address
n    name current address (empty string to delete)
x    show xrefs (search for xrefs)
#    edit comments
s    define a string
i    define an integer (32bit)
h    define a halfword (16bit)
b    define a byte (8bit)
^a   call assembler
^f   goto start of current function
     (indicated in the 2nd line)
^l   goto previous label
^t   show recursive function references
```

GDB

101 Basic commands.

102 Poor man's hex calculator.

```
$ gdb -q --batch -ex "p 0xbb -0xaa"
```

103 Follow forks made by debugged process.

```
(gdb) set follow-fork-mode child
(gdb) run
```

104 Attach to running process.

```
$ gdb -q --pid=13091
```

105 Compile with debugging information.

```
$ gcc -g file.c
$ gdb -q --symbols=./a.out
```

106 Enable core dumps and debug a dump.

```
# ulimit -c unlimited
# gdb -q -c core
```

xxd

107 blah

Miscellaneous

108 Hexdump in 4-byte lines.

```
$ cat > fmtstr << EOF
"%08.8_ax  " 4/1 "%02x "
"\t" "%_p"
"\n"
EOF
$ hexdump -v -f fmtstr binary
00000000  7f 45 4c 46   .ELF
00000004  01 01 01 00   ....
00000008  00 00 00 00   ....
[...]
```

VIII. OpenSSH

109 Show public key fingerprint and bubblebabble.

```
$ ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key.pub
# ssh-keygen -B -f /root/.ssh/id_rsa
```

110 Connect without allocating a TTY so that **w**(1) and **last**(1) won't log the connection.

```
$ ssh -T -o UserKnownHostsFile=/dev/null remote
```

Port forwarding

111 Issued at local, connections to port 1001 on local will go over SSH tunnel and connect from remote to localhost:25 (itself). Good way to reach a port behind a firewall that the remote host can reach when only SSH is allowed through. In this case, reach remote's local mailserver from local.

```
local$ ssh -L 1001:localhost:25 remote
```

112 Issued at local, connections to 1032 on remote will go over SSH tunnel to localhost:23 (telnet deamon on local). Good way to open up a tunnel to a service inside the firewall (at local) that cannot be reached from remote. Perfect way to prepare for a day working at home while still being able to reach a service inside the corporate network.

```
local$ ssh -R 1032:localhost:23 remote
```

113 Reverse SSH shell.

```
remote$ ssh -NR 3333:localhost:22 user@local
local$ ssh user@localhost -p 3333
```

114 You own a computer on the LAN inside the firewall and have a SSH client. You need access to the firewall's web interface with Firefox. Now you can connect to 8443 on hacker and it will go through the SSH tunnel and owned will connect to the router's (192.168.111.1) web server on port 443.

```
owned$ ssh -R 8443:192.168.111.1:443 hacker
```

IX. OpenSSL tools

115 Retrieve a certificate from an SSL service in PEM format (DER+base64 encoding with header and footer lines.

```
$ echo |\
> openssl s_client -connect host:port 2>&1 |\
> sed '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p
```

116 Message digest algorithms.

```
$ openssl dgst -md5 file1 file2
MD5(file1)= 82c77e1d0ceb6f08d5bb88bc5ab7884d
MD5(file2)= 65d4daeb6dd872b2821182bb306b20a9
$ echo -n string | openssl dgst -sha1
ecb252044b5ea0f679ee78ec1a12904739e2904d
```

117 Password hash generation.

```
$ openssl passwd -crypt passwd1 passwd2
4jyq0gu4DIcnQ
aFc.gGdSVBtpo
$ openssl passwd -1 -stdin
muff
$1$/eepN1BD$gkZqyWGnC0roLrr/VzFBN.
muff
$1$OBY2rPbG$WAUhF0X4Rlx8F/PI9lk6h1
$ openssl passwd -1 -salt OBY2rPbG muff
$1$OBY2rPbG$WAUhF0X4Rlx8F/PI9lk6h1
$ cat > cleartext << EOF
> adam
> bertil
> caesar
> david
> EOF
$ openssl passwd -table -apr1 -in cleartext
adam    $apr1$aB4gCgW.$4fj/v4OEclNeoGpIlBRHk0
bertil  $apr1$u1qLmaiU$2OIN2vHJtdToM75ftXh3A.
caesar  $apr1$s2AXjlP3$kTBkI/BJN3WbElMKW035E0
david   $apr1$Rt.elnCQ$CeppHIhvsFRW0C7RNe7WT1
```

118 File encryption. The -a option is for automatic base64 encoding of the contents and could be skipped.

```
$ echo this is cleartext > file.txt
$ openssl enc -aes-128-ecb -in file.txt -a \
> -pass pass:qwerty -out file.enc
$ cat file.enc
U2FsdGVkX19Kobq+Tb6FJeRpaoOVfgk6Jqau7xrGUZzx0RNyHm
m5Vgtk51JQ7N9B
$ openssl enc -aes-128-ecb -d -in file.enc -a \
> -pass stdin
qwerty
this is cleartext
```

119 Connect to an SSL-enabled service.

```
$ openssl s_client -connect host:port
```

120 Creating a self-signed certificate.

TBD

X. nmap

121 Timing options.

```
T0   serialized, 5 min between probes
T1   serialized, 15 secs between probes
T2   serialized, 0.4 secs between probes
T3   parallelization, nmap default
T4   rtt 500-1250 ms, max 6 retries
T5   rtt 50-250-300 ms, max 2 retries, 15 m/host
```

122 Top 10 responsive (open or closed) TCP ports.

```
80,25,22,443,21,113,23,53,554,3389
```

123 Top 10 open TCP ports.

```
80,23,22,443,3389,445,139,21,135,25
```

124 Top 10 open UDP ports (might wanna add 69/udp).

```
137,161,1434,123,138,445,135,67,139,53
```

125 This is only used for host discovery. Together with a high-numbered UDP port and a few ICMP packets (echo -PE, timestamp -PP or netmask -PM) we can optimize the host discovery scans of **nmap**(1).

```
# nmap -n -d -vv -T4 -sP -PS80,25,22,443,21,113 \
> -PA80,23,53 -PU53,52819 -PE -PP --source-port 53
```

XI. socat

126 Vanilla netcat-like TCP connection.

```
$ socat STDIN TCP4:remote:80
```

127 Enhanced netcat-like TCP connection with readline functionality, command history and proper network line terminators. Practical for repeated tests against HTTP servers for example.

```
$ socat READLINE,history=$HOME/http_history \
> TCP4:remote:80,crnl
```

128 Transfer file over TCP.

```
remote$ socat -u TCP4-LISTEN:8000 \
> CREATE:recieve.txt
local$ socat -u OPEN:send.txt TCP4:remote:8000
```

129 Transfer file over UDP. This is very unreliable (that's why tftp was created I guess).

```
remote$ socat -u UDP4-LISTEN:8000 \
> CREATE:recieve.txt
local$ socat -u OPEN:send.txt UDP4:remote:8000
```

130 Start the sender before-hand and let it wait for the listener to be set up and exit when finished.

```
local$ while ! socat -u OPEN:send.txt \
> TCP4:localhost:8000; do sleep 10; done
remote$ socat -u TCP4-LISTEN:8000 \
> CREATE:recieve.txt
```

131 Get the banner from a web server. Note the use of "-" instead of "STDIN"...

```
$ printf "HEAD / HTTP/1.0\r\n\r\n" | \
> socat - TCP4:www.dn.se:80 | grep Server
```

132 Tor relay, Privoxy is not needed obviously.

```
$ socat TCP4:LISTEN:8080,fork \
> SOCKS4A:localhost:dn.se:22,socksport=9050
$ nc -vv localhost 8080
```

133 SSL tunnel.

134 Proxy HTTP/SOCKS/CONNECT, etc.

135 Pipe.

136 Proxy chaining

137 Brokering.

XII. The Sleuth Kit

XIII. Package management

Yum

138 List installed packages.

139 Install and remove *package*.

140 Update a specific *package*.

141 Update all packages.

142 Search for a package containing pattern.

143 Show package description.

144 List files in a package.

145 Find out which package *file* belongs to.

RPM

146 List installed packages.

147 Install and remove *package*.

148 Update a specific *package*.

149 Update all packages.

150 Search for a package containing pattern.

151 Show package description.

152 List files in a package.

153 Find out which package *file* belongs to.

Zypper/Yast

154 List installed packages.

155 Install and remove *package*.

```
# zypper install package
```

156 Update a specific *package*.

157 Update all packages.

158 Search for a package containing pattern.

159 Show package description.

160 List files in a package.

161 Find out which package *file* belongs to.

APT

162 List installed packages.

```
$ dpkg --list
```

163 Install and remove *package*.

```
# apt-get install package
# aptitude install package
# apt-get remove package
# aptitude remove package
```

164 Remove unused dependencies.

```
# apt-get autoremove
```

165 Show available upgrades.

```
# apt-get safe-upgrade -s
```

166 Update a specific *package*.

167 Apply security updates.

168 Update package list and upgrade all packages.

```
# apt-get update
# apt-get safe-upgrade
```

169 Search for a package containing *pattern*.

```
# apt-cache search pattern
# aptitude search pattern
```

170 Show *package* description.

```
$ aptitude -v show package
$ dpkg –info package
```

171 List files in a package.

```
$ dpkg-query -L package
```

172 Find out which package *file* belongs to.

```
$ dpkg -S file
```

Slackware packages

173 List installed packages.

174 Install and remove *package*.

175 Update a specific *package*.

176 Update all packages.

177 Search for a package containing pattern.

178 Show package description.

179 List files in a package.

180 Find out which package *file* belongs to.

XIV. MySQL

181 Initial configuration. Add a database administrator and a limited account for the web server.

```
> drop database test;
> use mysql;
> delete from db;
> delete from users where not (user=”root” and
-> host=”localhost”);
> set password for 'root'@'localhost' =
password('nisse');
> flush privileges;
> create user 'dba'@'localhost' identified by
-> 'qwerty';
> grant all privileges on *.* to 'dba'@'localhost'
-> with grant option;
> create database webbapp;
> create user 'webusr'@'localhost' identified by
-> 'qwerty';
> grant select on webbapp.* to
-> 'webusr'@'localhost';
> quit;
# cat /dev/null > ~/.mysql_history
# chmod 000 ~/.mysql_history
```

182 Take a full backup of all databases and restore them.

```
# mysqldump -A -u root -p | gzip > backup.sql.gz
Password:
# echo “drop database webapp;” | mysql -u root -p
Password:
# gunzip backup.sql.gz
# mysql -u root -p < backup.sql
```

183 Create a database with table and fill it.

```
> create database collection;
> use collection;
> create table films
-> (id int not null auto_increment key,
-> title varchar(64) not null,
-> length tinyint comment 'in minutes'.
-> category enum('action','drama','comedy'),
-> purchase_date date);
> alter table films add price decimal(8,2)
-> not null;
> insert into films values
-> (0,'Up',83,'action', 69.90);
> insert into films values
-> (0,'Ned',114.'drama',139,90);
> update films set title='XXX' where id=1;
> delete from films where id>3;
```

184 Dump a table as comma-separated values.

```
$ echo "select * from mysq.user;" |\
> mysql -u root -B | sed 's/\t/,/g' > output.csv
```

185 Some miscellaneous **mysqladmin**(1) commands.

```
ping        exit code 0 if online
version
```

186 TBD

XV. Networking

16 Subnetting chart.

```
     Network Section           Host Section      Net Mask
---+---------+-----------+---------+-----------+-----------
128|1        |   0/510   | 000 0000| 126/32,766|1/9/17/25
192|11       |   2/1,022 |  00 0000|  62/16,382|2/10/18/26
224|111      |   6/2,046 |   0 0000|  30/8,190 |3/11/19/27
240|1111     |  14/4,094 |     0000|  14/4,094 |4/12/20/28
248|1111 1   |  30/8,190 |      000|   6/2,046 |5/13/21/29
252|1111 11  |  62/16,382|       00|   2/1,022 |6/14/22/30
254|1111 111 | 126/32,766|        0|   0/510   |7/15/23/31
255|1111 1111| 254/65,534|         |   x/254   |8/16/24/32
```

187 The 256-rule: Netmask 255.255.255.240 has a block size of 256-240=16. First subnet after the zero subnet is 16, second 32. First subnets broadcast is 31, host ranges 17-30.

XVII. License information