Linux
Network Security
Course

Linux/Network Security

Table of Contents

1 - System Security

1.1 - System recovery.

1.1.1 - Emergency Recovery (81_Emergency_Recovery)

Boot Protection:
- Lock the PC(server) away from public access

- in BIOS:
  - -Set BIOS to boot only C:
  - -Set a password for BIOS access

- in BOOT LOADER

- System recovery (unknown root password or bad `/etc/fstab`) via:
  - Using another system:
    Boot Diskette, Rescue diskette, Live CD, second installed system
    Protection against it (preventing booting from other than first HD):
    - Locked PC case
    - BIOS Boot device set to C: only & BIOS password

  - Using Boot kernel option (in LILO/GRUB) `init=/bin/bash`
    - Remount the root directory: mount -o remount,rw /
    - `vi /etc/passwd` ---> delete the x in `root:x:0:0:`.....
    - `umount -a` and reboot with <Ctrl-Alt-Del>

  - Protection against it:
    - in `/etc/lilo.conf` (if using LILO)
      `restricted` and `password ="`*xxxxxxx*`"` or
    - in `/boot/grub/menu.lst` (if using GRUB)
      `password ="`*xxxxxxx*`"` in the global section.
      If the booting also needs to be authenticated then insert
      `lock` immediately at the line after each title to limit access.
      GRUB will expect to press 'p' before entering the password.

  - What to change to login as root without password:
    - Delete the 'x' for `root` user in `/etc/passwd` or
    - Delete the encrypted password of root in `/etc/shadow`
    - or enter line: *username hostname*`=NOPASSWD:/bin/su`
      in `/etc/sudoers`. Reboot, login as user and enter:
      `sudo su -` (*username* is a user with known passowrd)

- Disable the `<Ctrl-Alt-Del>` in `/etc/inittab`
- Make boot and emergency disks (boot and rescue)

1.2 - Passwords Security:

1.2.1 - Use the Shadow system
- `/etc/passwd` and `/etc/shadow` concept and format

1.2.2 - Importance of password content

- Recommended composition: 5 char, 2 digits, 2 special Chars.
- Allowed characters in password:
`# * , . ; : _ - + ! $ % & / | ? { [ ( ) ] }`

1.2.4 - Changing the password length: `/etc/login.defs`

```
PASS_MAX_LEN  16
PASS_MIN_LEN   8
```

1.2.5 - Changing password encryption method
From 3DES to MD5 or `blowfish`

- Change the value of `CRYPT_FILES` in `/etc/default/passwd`
  Possible values: `des,md5,blowfish`
  eg. `CRYPT_FILES=md5`

or

- For SuSE, use Yast to change the security options
  `'Security and Users' ---> 'Security Settings'`

or

- Add the extra parameter `md5` or `blowfish` in both of these files:
  ```
  /etc/security/pam_pwcheck.conf
  /etc/security/pam_unix2.conf
  ```
  eg. `password:       md5 nullok` or
- Add the parameter `md5` or `blowfish` to all of the files concerning applications using the modules `pam_pwcheck` and `pam_unix2`

eg. in `/etc/pam.d/login`

```
password required  pam_pwcheck.so  md5 nullok
```

and in `/etc/pam.d/passwd`

```
password required  pam_unix2.so    md5 nullok
```

1.2.6 - Cracking passwords:
Using john:
`john /etc/shadow` or `john -users:root /etc/shadow`

Using other crackers:
http://www.password-crackers.com/

1.3 - File System security

1.3.1 - Access rights settings:

– **SUID**, SGID, StickyBit, acls and attributes (`+i +a`)
The following commands can help to find all the files that have the SUID activated.

```
      find / -perm +u+s
or    ls -laR / | grep "^rws"
```

– Through **`yast`** or **`chkstat`**

```
/etc/permissions
/etc/permissions.easy
/etc/permissions.secure
/etc/permissions.paranoid
```
`/etc/permissions.local` (user entered permissions checks)

File syntax:	Filename	owner.group	permissions
eg.	`/var/tmp`	`root.root`	`1777`

Command syntax:

`chkstat /etc/permissions.easy` checks the permissions only

`chkstat -set /etc/permissions.easy` sets the permissions

1.3.2 - Encrypting files with GPG:

Encrypting a file:

```
gpg -c filename
```

Will ask for a passphrase 2 times

The file `filename.gpg` will be created in the same directory as the original.

Note: Since the original file is NOT erased, it should be erased by hand.

Decrypting the file back to original:

```
gpg filename.gpg
```

Will be asked for the passphrase once.

If successful then the original file will be created.

The `.gpg` file will still be present.

Automatizing Encription/Decryption of a file

using a password(pass) included in the script

This script will not ask for a passphrase.

It will use the word (example: 'pass') as passphrase

NOTE: use the following extra option to encrypt files that are bigger than 1 GB.

```
--cipher-algo twofish
```

IMPORTANT: Since the passphrase is in clear text in the script, make sure that the script is not readable from anybody else than the user for which the script was intended to

Encryption:

Encryption of file '`gpgtest1`' using passphrase '`pass`' into `gpgtest.gpg`.

The original file '`gpgtest1`' will then be erased (`rm gpgtest1`)

```
echo pass | gpg --batch --passphrase-fd 0 --symmetric gpgtest1
rm gpgtest1
```

Decryption:

Decrypting a gpg file using passphrase '`pass`' into `gpgtest`.

The original encrypted file '`gpgtest1.gpg`' will then be erased (`rm gpgtest1`)

```
echo pass | gpg --batch --decrypt --passphrase-fd 0 -o gpgtest1 gpgtest1.gpg
rm gpgtest1.gpg
```

1.3.3 - Limiting logins

- `/etc/securetty` defines in which terminals root is allowed to login.
- `/etc/nologin` if file exists, no login allowed from any users except from root
- `/bin/false` in `/etc/passwd`. If entered in shell field, it prevents a single user to login.

- `Characters'!'` or `'*'` in `/etc/shadow` in the password field either alone or within the encrypted password, prevents a user to login:
  Why? the password becomes illegal.

1.3.3 - IDS - **I**ntrusion **D**etection **S**ystems

1.3.3.1 Tripwire

Up to SuSE 8.1

- Configure the Tripwire configuration file.`/etc/tw.config`
  `man tw.config` Man page for Tripwire configuration entries
  - Syntax: `[!|=]`*`File|Dir SelectionMask`*
    eg.

```
/etc       +pinugsm12-ac3456789
/var/log   L
     /bin       R
     !/etc/
     =/home     R  (= Only one dir level)
```

  In these above examples:
  - `/etc` and its recursive content is checked against: **p**ermissions, **i**nodes, **n**umber of hard-links, **U**ID, **G**ID, **s**ize, (**1**)MD5, **m**odified-timest**a**mp, (**2**)snefru and <u>NOT</u> against **a**ccess-timestamp, **c**hange-timestamp, all other security algorithms(**3456789**) .
  - `/val/log` and its recursive content is checked against: **p**ermissions, **i**nodes, **n**umber of hard-links, **U**ID, **G**ID and <u>NOT</u> the rest of selection masks. Normally meant for LOG files.
  - `/bin` and its recursive content is checked against:
    same as for `/etc`. Normally meant for Read-Only files(programs, etc).
  - `/proc` and its content is excluded form any checks.
  - `/home` and its content is checked against:
    same as for `/etc` BUT its children directories are NOTchecked.
- Building the files status database:
  `tripwire -v -initialize`
  This will create a database called `./databases/tw.db_`*`$HOSTNAME`*
  Now the sytem is used and when a suspicion that the system is been intruded:
- Check the system against the database:
  - `mkdir /etc/tw`
  - `cp ./databases/tw.db_`*`$HOSTNAME`*` /etc/tw/`
  - `tripwire`
    The result will be displayed for every suspicious file that has been altered, deleted or added. This result can be piped to a file for future viewing.

SuSE 8.2 and later.

The package tripwire comes with a basic configuration file `/etc/tripwire/twcfg.txt`, which sets the mandatory variables to the defaults as described in the `twconfig(4)` manual page. This configuration is merely enough to set tripwire to work.
The following steps can serve you as a quick cookbook for setting tripwire to work.

- Generate keys: Choose a convenient *`HOSTNAME`* and generate local and site keys using
  ```
  twadmin --generate-keys -L /etc/tripwire/$(hostname)-local.key
  twadmin --generate-keys -S /etc/tripwire/site.key
  ```
  This creates the files named above as arguments.
- Designate the configuration file with:
  ```
  twadmin --create-cfgfile -S /etc/tripwire/site.key \
  /etc/tripwire/twcfg.txt
  ```
  This creates file `/etc/tripwire/tw.cfg` in a binary format.
- Create a policy file.
  - A complex example can be found in:
    `/usr/share/doc/packages/tripwire/twpol.txt`.
    See the section The Tripwire 'Policy' file on the next page or:
    For test purposes, enter a single rule:
    `/bin -> $(ReadOnly);` <---the ending semicolon is necessary
  - Create the policy file based on this `twpol.txt` file.
    ```
    twadmin --create-polfile -S /etc/tripwire/site.key \
    /etc/tripwire/twpol.txt
    ```
  provided `/etc/tripwire/twpol.txt` is the name of your policy file,
  this command creates the binary encoded policy file `/etc/tripwire/tw.pol`
- Generate a baseline database (snapshot of the objects residing on the system, according to the created policy file) using:
  ```
  tripwire --init
  ```
  This creates database file `/var/lib/tripwire/$(hostname).twd`
  Save this database file and the `/etc/tripwire/*` directory in a secure place for future system checks.
  eg.
  ```
  mv -f /var/lib/tripwire/$(hostname).twd /mnt/secure/
  mv -f /etc/tripwire /mnt/secure/
  ```
- Checking the system:
  - Recover the securely saved database and tripwire files back to their original place:
    ```
            /var/lib/tripwire/$(hostname).twd
    and     /etc/tripwire/*
    eg.     cp /mnt/secure/$(hostname).twd /var/lib/tripwire/
            mkdir /etc/tripwire
            cp /mnt/secure/tripwire/*  /etc/tripwire
    ```
  - And perform the system check.
    ```
    tripwire --check
    ```
    - This prints a report on the standard output and generates the file:
      `/var/lib/tripwire/report/$HOSTNAME-YYYYMMDD-HHMMSS.twr`

- The report can be redisplayed using:
  ```
  twprint --print-report -r \
  /var/lib/tripwire/report/$HOSTNAME-YYYYMMDD-HMMSS.twr
  ```
- Updating of database after regular administration:
  - Define the variable EDITOR to use `vim` text editor to select items to update: `export EDITOR=/usr/bin/vim`
  - Using the last relevant report, edit the excludes by issuing the command:
  ```
  tripwire --update --twrfile \
  /var/lib/tripwire/report/$HOSTNAME-YYYYMMDD-HMMSS.twr
  ```

The Tripwire 'Policy' file(from SuSE 8.2 and up)

`/etc/tripwire/twpol.txt` Tripwire text Policy file
`man twpolicy` Man page for `/etc/tripwire/twpol.txt`

– Syntax: `[!|=]`*`File|Dir`* `->` *`SelectionMask;`* `[#` *`Comments`*`]`

eg.
```
/etc      ->   +pinugtsdbmCM-rlacSH;
/var/log  ->   $(Growing);
/bin      ->   $(ReadOnly);
!/etc/mtab;                          (Do not scan this file)
=/home    ->   $(ReadOnly);          (= One dir level)
```

Selection Masks

-	Ignore the following properties
+	Record and check the following properties
a	Access timestamp
b	Number of blocks allocated
c	Inode timestamp (create/modify)
d	ID of device on which inode resides
g	File owner's group ID
i	Inode number
l	File is increasing in size (a "growing file")
m	Modification timestamp
n	Number of links (inode reference count)
p	Permissions and file mode bits
r	ID of device pointed to by inode (valid only for device objects)
s	File size
t	File type
u	File owner's user ID
C	CRC-32 hash value
H	Haval hash value
M	MD5 hash value
S	SHA hash value

User defined Variables

Syntax: `Variablename = `*`SelectionMask`*

eg.
```
param1 = +SMCH;         # Set variable param1.
dir1   = /etc/inet;     # Set variable dir1.
DIR1   = /etc/init.d;   # Variables are case sensitive.
```

```
$(dir1)   -> +tbamc;     # Left hand substitution.
/etc/inet -> $(param1);  # Right hand substitution.
$(DIR1)   -> $(param1);  # Double substitution.
```

Fixed Mask Variables

`ReadOnly`	ReadOnly is good for files that are widely available but are intended to be read-only. Value: `+pinugtsdbmCM-rlacSH`
`Dynamic`	Dynamic is good for monitoring user directories and files that tend to be dynamic in behavior. Value: `+pinugtd-srlbamcCMSH`
`Growing`	The Growing variable is intended for files that should only get larger. Value: `+pinugtdl-srbamcCMSH`
`Device`	Device is good for devices or other files that Tripwire should not attempt to open. Value: `+pugsdr-intlbamcCMSH`
`IgnoreAll`	IgnoreAll tracks a file's presence or absence, but doesn't check any other properties. Value: `-pinugtsdrlbamcCMSH`
`IgnoreNone`	IgnoreNone turns on all properties and provides a convenient starting point for defining your own property masks. (For example, `mymask = $(IgnoreNone) -ar;`) Value: `+pinugtsdrbamcCMSH-l`

1.3.3.2 - AIDE (**A**dvanced **I**ntrusion **D**etection **E**nvironment)

- **Description:** Aide is composed of only one program and its configuration file. `/usr/bin/aide` and `/etc/aide.conf`
  It creates a database of file and directories attributes that can be used to check the integrity of the system at a later date. Unlike Tripwire, it doesn't use encryption methods or keys. The database needs to be saved in a secure place. eg. usb-stick mounted on `/mnt/secure`
- **Installation:** Package aide from SuSE CD
- **Configuration:**
  Edit the file `/etc/aide.conf` and change the database path:
  eg.
  ```
  database=file:/var/lib/aide/aide.db
  database_out=file:/var/lib/aide/aide.db.new
  ```
  The `.new` database is created when a new `aide --init` is run.
  The rest of the parameters in the configuration file are setting that control which directories and their type of attributes checks.
  SuSE has already entered the normal settings.
  More info on the configuration by running: `man aide.conf`
- **Initializing the file system Database:**
  Issue the command: `aide --init`
- **Checking the system against an original database:**
  - Rename the last created database from `aide.db.new` to `aide.db`
  - Issue the command:
  ```
  aide --verbose=20 --check | tee /mnt/secure/aide.log
  ```
  Providing that we want the log file to appear on the screen and also be saved in a secure place with the databases.
  Verbose level can be from 0 to 255. 20 gives us a workable output.

- More information in: `man aide` or `man aide.conf`
- Other configuration examples can be found in: `/usr/share/doc/packages/aide/examples`

1.4 - Logging System

1.4.1 - Log server (55_Log_Files)

- Client Configuration
  `/etc/syslog.conf` ----> `@`*`logservername`*
- Server Configuration
  `/etc/sysconfig/syslog` ---> `SYSLOGD_PARAMS="-r -s` *`domain`*`"`
  (`domain` is the name of the domain that should be stripped off the client's name for the line to log)
- Restart `syslogd` of Client and Server:
  `rcsyslog restart`

1.4.2 - Logcheck (using SuSE Linux)

- **Description**

  Logcheck is composed of a script (`logcheck.sh`) that calls a binary program (`logtail`) to check for specific patterns based on lists (`logcheck.hacking`, `logcheck.violations ,etc`). The result is sent to the logcheck admin. by email.

- **Installation**
  - Unpack the logcheck-*x.x.x*.tar.gz to `/usr/local/logcheck-`*`x.x.x`*
  - Edit the script: `/usr/local/logcheck-`*`x.x.x`*`/systems/linux/logcheck.sh`
    Enter the `SYSADMIN` email address and under the RedHat section, check the paths for logs. Normally `/var/log/messages` & `/var/log/mail`
  - cd `/usr/local/logcheck-`*`x.x.x`*
  - Compile the `logtail` program with the command: `make linux`
    The program `logtail` and `logcheck.sh` will be created and copied to:
    `/usr/local/bin/logtail` (logcheck binary program)
    `/usr/local/etc/logcheck.sh` (main logcheck shell script)
    The following files will also be created and copied:(see details below)
    ```
    /usr/local/etc/logcheck.hacking
    /usr/local/etc/logcheck.violations
    /usr/local/etc/logcheck.violations.ignore
    /usr/local/etc/logcheck.ignore
    ```
  - Edit the crontab with `crontab -e` and include the regular job (each hour):
    `/usr/local/etc/logcheck.sh`
  - Make sure the syslog logs all messages (`*.info`) to `allmessages` file.
  - Results will be reported by email to the `SYSADMIN` address entered in `logcheck.sh`. This address can be either local user (eg. `root`) or a full e-mail address(`michel@linuxint.com`)

- **Content of keywords files.**

`/usr/local/etc/`**`logcheck.hacking`** (**NOT** Case sentitive)
Contains keywords that triggers the special 'ACTIVE SYSTEM ATTACK' reporting.
`/usr/local/etc/`**`logcheck.violations`** (**NOT** Case sentitive)
Contains system events keywords that triggers the 'Security Violation' reporting.
`/usr/local/etc/`**`logcheck.violations.ignore`** (Case sentitive)
Contains keywords that force ingnoring some of the positive detections provoked from the keywords in `logcheck.violations.`
`/usr/local/etc/`**`logcheck.ignore`** (Case sentitive)
Contains keywords that when detected will force ignore the entry no matter what.

- **The search logic:**

-`logcheck.sh` executes hourly
-`logcheck.sh` executes logtail on log files
-`logtail` parses off any text from the last time it was run **-->**
-extract text for system attack messages**`(logcheck.hacking)`-->**
-extract text for security violations**`(logcheck.violations)`-->**
-scans text for security violations to ignore**`(logcheck.violations.ignore)`-->**
-scans text for all messages to ignore**`(logcheck.ignore)` -->**
any messages found are e-mailed to system administrator **-->** `sendmail`
This process makes sure that all messages that should be ignored are, and the rest is reported with the title: ACTIVE SYSTEM ATTACK or 'Security Violation' or 'Unusual System Events'.

2- Network Security

2.1 - Check services with Port scanners:

2.1.1 - NMAP and XNMAP

Installation:
package `nmap` and `nmap-gtk` (on SuSE CD/DVD)

Introduction examples:

- `nmap -v` *`hostname`*
  Scan all reserved ports on the host with verbose on process(-v)
- `nmap -sS -O 192.168.100.0/24`
  Scan the whole Subnet with half opened connections:
  TCP SYN stealth(`-sS`) with Operating system detection (`-O`)
- `nmap -sP '192.168.70-80.*'`
  Scan with pings (search for alive hosts)
- `nmap -sX -p 1-1024,1080,6666,31337 192.168.100.60`
  Christmas scan of all well known ports(`1-1024`) and some extras of host

Syntax:
nmap [*`Scan_Type(s)`*] [*`Options`*] <*`host_or_net_list`*>

Common Scan Types ('*' options require root privileges)

`-sT`	TCP connect() port scan (default)
* `-sS`	TCP SYN stealth port scan (best all-around TCP scan)
* `-sU`	UDP port scan
`-sP`	ping scan (Find any reachable machines)
* `-sF,-sX,-sN`	Stealth FIN, Xmas, or Null scan (experts only)

`-sR/-I` RPC/Identd scan (use with other scan types)

Common Options (none are required, most can be combined):

```
* -O                    Use TCP/IP fingerprinting to guess remote
                        operating system
  -p <range>            Ports to scan.  Example range: '1-1024,1080,6666,31337'
  -F                    Only scans ports listed in nmap-services(Fast
                        scans)
  -v                    Verbose. Its use is recommended. Use twice for
                        greater effect.
  -P0                   Don't ping hosts (needed to scan microsoft.com and
                        others)
* -Ddecoy_host1,decoy2[,...]
                        Hide scan using many decoys
  -T {Paranoid|Sneaky|Polite|Normal|Aggressive|Insane}
                        General timing policy (Paranoid =slow, Insane =fast)
  -n/-R                 Never do DNS resolution/Always resolve
                        [default: sometimes resolve]
  -oN/-oX/-oG logfile
                        Output normal/XML/grepable scan logs to logfile
  -iL <inputfile>
                        Get targets from file; Use '-' for stdin
* -S your_IP/-e devicename
                        Specify source address or network interface
  --interactive
                        Go into interactive mode (then press h for help)
```

Example:

```
nmap -v -sS -O www.my.com 192.168.0.0/16 '192.88-90.*.*'
```

Examples extracted form nan `nmap`

`nmap -v target.example.com`

This option scans all reserved TCP ports on the machine `target.example.com` . The `-v` means turn on verbose mode.

`nmap -sS -O target.example.com/24`

Launches a stealth SYN(`-sS`) scan against each machine that is up out of the 255 machines on class "C" where `target.example.com` resides. It also tries to determine what operating system(`-O`) is running on each host that is up and running. This requires root privileges because of the SYN scan and the OS detection.

`nmap -sX -p 22,53,110,143,4564 198.116.*.1-127`

Sends an Xmas tree scan(`-sX`) to the first half of each of the 255 possible 8 bitsubnets in the 198.116 class "B" address space. We are testing whether the systems run sshd, DNS, pop3d, imapd, or port 4564. Note that Xmas scan doesn't work on Microsoft boxes due to their deficient TCP stack. Same goes with CISCO, IRIX, HP/UX, and BSDI boxes.

`nmap -v --randomize_hosts -p 80 *.*.2.3-5`

Rather than focus on a specific IP range, it is sometimes interesting to slice up the entire Internet and scan a small sample from each slice. This command finds all web servers on machines with IP addresses ending in .2.3, .2.4, or .2.5. If you are root you might as well add `-sS`. Also you will find more interesting machines starting at 127. so you might want to use "127-222" instead of the first asterisks because that section has a greater density of interesting machines (IMHO).

```
host -l company.com | cut  -d  -f 4 | ./nmap -v -iL -
```

Do a DNS zone transfer to find the hosts in `company.com` and then feed the IP addresses to nmap. The above commands are for my GNU/Linux box. You may need different commands/options on other operating systems.

XNMAP or **NMAPFE**

`xnmap` is a gtk graphic interface for `nmap`.
In SuSE 9.2 this program is in package: `nmap-gtk` the binary is `nmapfe`

If started as normal user, only above NON-ROOT options are allowed. (options without '*****')
- Enter the target host or host range
  eg. `192.168.0.0/16` or `'192.88-90.*.*'`
- Select the desired options and click on SCAN button.
- Note: The `nmap` command that will be run in the background and its parameters are shown under the options after: `Output from:`

2.1.2 - NESSUS (Advanced port scanner/attack simulator)

- Installation Packages: `nessus-core, nessus-libraries, libnasl`

- Composition of Nessus:
  - The daemon `nessusd` (it aslo authenticates users from the client.)
  - The client (`nessus`) X-program which controls the `nessus` daemon.

- Creating Nessus daemon users (must also exist in system):
  `nessus-adduser`<Enter>
  Login: `michel`
  Authentication (pass/cert) [pass]: <*Enter*>
  Login Password: *`xxxxxxx`* (nessus password, not the system password for that user)
  (Enter access rules here)
  eg. `accept 192.168.100.0/24`<*Enter*>
  `default deny`<*Enter*>
  <`Ctrl-D`> This creates 2 files in user's home dir: `.nessusrc .nessusrc.cert`
  It then creates a list of user's properties

NOTE: In SuSE 9.2 it produces an error.

```
trap: usage: trap [-lp] [[arg] signal_spec ...]
rm: cannot remove `/var/tmp/nessus-adduser.12239': Is a directory

INTERRUPT

trap: usage: trap [-lp] [[arg] signal_spec ...]
```

IGNORE THE ERROR

- Creating Certificates:
  NOTE for SUSE 9.2: We must comment out (add a #)the line 394 (`trap 0`) in Script `/usr/sbin/nessus-mkcert`
  - Command to produce an SSL certificate:
    `nessus-mkcert` <Enter>
    - Answer all the questions by <*Enter*> except the Country and Location.
    The certificate files will be created in :
    Certification authority :
    Certificate = `/var/lib/nessus/CA/cacert.pem`
    Private key = `/var/lib/nessus/CA/cakey.pem`
    Nessus Server :
    Certificate = `/var/lib/nessus/CA/servercert.pem`
    Private key = `/var/lib/nessus/CA/serverkey.pem`

- Starting nessus Daemon
  `nessusd -D`

- Starting Nessus Client: Login as the user and start the nessus client:
  <Alt-F2> `nessus`
  - Give the name and password of a nessus user and click on LOGIN
  - Click on the third option and accept the Certificate.

- Starting the port scanning:
  - Select the tab: `Target Selection`
  - Click on button: `Enable all but dangerous plugins`
  - Enter the hostname or IP of the host to scan.
  - Click on: `Start the scan`...this can last a long time....

  - Results: When scan is finished, the window will change:
    - Click on the icon on the left panel to see the report on the right
    - To save the report:
      - Click on: `Save report`
      - Set the `Report file format` to `HTML`
      - Under `Selection` give the *`filename`* and click `OK`
      - Use a Browser to look at the results in this created HTML file
    - Check also the `/var/log/messages` on the scanned host and see ..

- Nessus command options:

`nessusd -D`	Starts the nessusd as daemon
`nessusd -p 1241`	`nessusd` listens on Port 1241 (default)

- Nessus administration:

`nessus-adduser`	Allows to add a nessus user
`nessus-rmuser username`	Allows to remove a nessus user
`nessus-mkcert`	Allows to create a new nessus certificate.

2.1.3 - SAINT (**S**ecurity **A**dministrator's **I**ntegrity **N**etwork **T**oot):

What is SAINT:
Saint is the more evolved follower of the port scanner `satan`

Installation:
Install the `saint` package from CD

Starting Saint:
As root , start the saint with the command: `saint`
If it doesn't want to start, then edit its configuration file:
`/usr/lib/saint/config/saint.cf`
and add in:
The default browser will start with the *saint* page:
`file://localhost/usr/lib/saint/html/saint.html`

Scan with Saint:
- Click on Target Selection Button(on left pane)
- Enter the target Hostname in `Primary target selection`
- Select the scan level in `Scanning level selection`
- If behind a Firewall Select the 'Firewall Support' option in:
  `Firewall Support`
- Click on `Start the scan` button.
- Progress of the scan will be shown.
- When finished, to see the report, click on one of the links:
- Continue with report and analysis or
- View primary target results

2.2 - Monitoring system intrusions:

2.2.1 - NIDS- SNORT - **N**etwork **I**ntrusion **D**etection **S**ystem (**SuSE** 8.2 and up)

- Installation of `snort` package is from SuSE CD
  SuSE 8.2 has the Version 1.9.1
  Debian 'Woody' has the version 1.8.4

- Configuring snort: Edit the `/etc/snort/snort.conf` and:
  1) Set the network variables for your network
     ```
     var HOME_NET $ppp0_ADDRESS
     or $netzkarte_ADDRESS eg. $eth0_ADDRESS
     or any
     ```
  2) Configure preprocessors (change to do is in underlined here)
     The preprocessors are the intelligent modules that detects certain types of attacks. eg.
     ```
     preprocessor rpc_decode: 111  2049 <---- For NFS
     preprocessor bo: -nobrute
     ```
  3) Configure output plugins. It allerts the syslog with a specific message.
     Syntax: `output alert_syslog:LOG_`*`facility`*` LOG_`*`priority`*
     eg. `output alert_syslog: LOG_LOCAL0 LOG_ALERT`
  4)Edit the file `/etc/syslog.conf` and set the proper destination for `local0.alert` messages.
     eg. `local0.alert   -/var/log/snort.log`
     and then restart `syslogd (rcsyslog restart)`
  5) Customize your rule set as needed (may already be prepared by the distribution)
  6) Edit the file `/etc/sysconfig/snort` and set the paramters of the variables as needed.
     ```
     SNORT_INTERFACE="ppp0" or "$netzkarte"
     SNORT_ACTIVATE="no"
     SNORT_AUTO="yes"       Auto IP Number setting of HOME_NET
     SNORT_PROMISC="no"
     SNORT_USER="snort"
     SNORT_GROUP="snort"
     SNORT_EXTRA_OPTIONS="" <----Extra command line options come here
     ```

• To Start/Stop the snort Daemon:
```
rcsnort {start|stop|restart|reload|status|activate|deactivate}
```
`activate|deactivate` is for automatic snort startup on interface startup
In SuSE 8.2 It will start with the following parameters:
```
-dD  -i $SNORT_INTERFACE $PROMISC \
     -l /var/log/snort \
     -u $SNORT_USER \
     -g $SNORT_GROUP \
     -c /etc/snort/snort.conf \
     $SNORT_EXTRA_OPTIONS
```
(The settings for above variables are found in `/etc/sysconfig/snort`)

In Debian 'Woody' it will start with the following parameters:

```
-Dbd -S "HOME_NET=[$DEBIAN_SNORT_HOME_NET]" \
     -h "$DEBIAN_SNORT_HOME_NET" -c /etc/snort/snort.conf
     -l /var/log/snort -u snort -g snort \
      $DEBIAN_SNORT_OPTIONS >/dev/null
```

(The settings of above `Variables` are set in `/etc/snort/snort.debian.conf`)

SNORT Report example (received per email)

```
Subject: [SNORT] dozlinux.linux.local
 daily report
The log begins from: Jan 3 07:55:45
The log ends     at: Jan 6 01:54:00
Total events: 144
Signatures recorded: 10
Source IP recorded: 45
Destination IP recorded: 4


The number of attacks from same host to same
destination using same method
=========================================================================
  # of
 attacks  from             to               method
=========================================================================
   26     217.230.115.132   217.230.248.153   WEB-IIS cmd.exe access : {TCP}
   15     212.21.228.22     217.230.248.153   HTTP Request {TCP}
   5      62.134.73.105     217.5.61.66       ICMP PING NMAP : {ICMP}
   5      62.134.72.1       217.5.61.66       ICMP PING NMAP : {ICMP}
   4      217.8.87.195      217.5.50.16       ICMP PING CyberKit 2.2 Windows : {ICMP}
   4      62.134.77.204     217.5.50.16       ICMP PING NMAP : {ICMP}
   3      217.8.34.20       217.5.50.16       ICMP PING CyberKit 2.2 Windows : {ICMP}
   3      217.227.129.246   217.230.254.214   ICMP PING CyberKit 2.2 Windows : {ICMP}
   3      62.134.73.248     217.5.61.66       ICMP PING NMAP : {ICMP}
   2      210.224.186.4     217.230.254.214   ICMP PING speedera : {ICMP}
   2      63.166.13.66      217.230.254.214   ICMP PING speedera : {ICMP}
   2      217.8.80.7        217.5.50.16       ICMP PING CyberKit 2.2 Windows : {ICMP}
   2      202.160.241.130   217.230.254.214   ICMP PING speedera : {ICMP}
   2      62.134.73.3       217.5.61.66       ICMP PING NMAP : {ICMP}
   2      64.0.96.12        217.230.254.214   ICMP PING speedera : {ICMP}
   2      63.209.221.226    217.230.254.214   ICMP PING speedera : {ICMP}
   2      217.230.115.132   217.230.248.153   WEB-IIS CodeRed v2 root.exe access : {TCP}
   2      208.185.54.14     217.230.254.214   ICMP PING speedera : {ICMP}
   2      217.229.69.159    217.230.248.153   ICMP PING CyberKit 2.2 Windows : {ICMP}
   2      62.134.73.147     217.5.61.66       ICMP PING NMAP : {ICMP}
   2      64.14.117.10      217.230.254.214   ICMP PING speedera : {ICMP}
   2      212.162.1.194     217.230.254.214   ICMP PING speedera : {ICMP}
   2      217.230.115.132   217.230.248.153   HTTP Request {TCP}
   2      205.158.108.194   217.230.254.214   ICMP PING speedera : {ICMP}
   2      62.134.76.41      217.5.50.16       ICMP PING NMAP : {ICMP}
   2      208.185.219.166   217.230.254.214   ICMP PING speedera : {ICMP}
   2      62.134.78.101     217.5.61.66       ICMP PING NMAP : {ICMP}
   2      217.230.115.132   217.230.248.153   WEB-FRONTPAGE /_vti_bin/ access : {TCP}
   2      63.219.179.130    217.230.254.214   ICMP PING speedera : {ICMP}


Percentage and number of attacks from a host to a
destination
============================================================
        #  of
  %    attacks   from             to
============================================================
10.42    15      212.21.228.22     217.230.248.153
 3.47    5       62.134.72.1       217.5.61.66
 2.08    3       62.134.73.248     217.5.61.66
 2.08    3       217.227.129.246   217.230.254.214
 2.08    3       217.7.65.77       217.5.50.16
 1.39    2       205.158.108.194   217.230.254.214
 1.39    2       62.134.78.101     217.5.61.66
 1.39    2       208.185.219.166   217.230.254.214
 1.39    2       63.166.13.66      217.230.254.214
 1.39    2       210.224.186.4     217.230.254.214
```

```
Percentage and number of attacks from one host to any
with same method
==============================================================
        #  of
  %    attacks   from              method
==============================================================
18.06    26      217.230.115.132   WEB-IIS cmd.exe access : {TCP}
10.42    15      212.21.228.22     HTTP Request {TCP}
 4.17    6       62.134.72.1       ICMP PING NMAP : {ICMP}
 4.17    6       62.134.73.105     ICMP PING NMAP : {ICMP}
 4.17    6       217.8.34.20       ICMP PING CyberKit 2.2 Windows : {ICMP}
 2.78    4       62.134.77.204     ICMP PING NMAP : {ICMP}
 2.78    4       217.7.65.77       ICMP PING CyberKit 2.2 Windows : {ICMP}
 2.08    3       217.227.129.246   ICMP PING CyberKit 2.2 Windows : {ICMP}
 2.08    3       62.134.73.248     ICMP PING NMAP : {ICMP}
 2.08    3       62.134.73.3       ICMP PING NMAP : {ICMP}
 2.08    3       217.229.69.159    ICMP PING CyberKit 2.2 Windows : {ICMP}
 2.08    3       62.134.73.147     ICMP PING NMAP : {ICMP}
 1.39    2       217.230.115.132   WEB-IIS CodeRed v2 root.exe access : {TCP}
 1.39    2       208.185.219.166   ICMP PING speedera : {ICMP}

 1.39    2       64.14.117.10      ICMP PING speedera : {ICMP}
 1.39    2       217.230.115.132   HTTP Request {TCP}
 1.39    2       64.41.192.103     ICMP PING speedera : {ICMP}
 1.39    2       63.166.13.66      ICMP PING speedera : {ICMP}
 1.39    2       217.230.115.132   WEB-FRONTPAGE /_vti_bin/ access : {TCP}


Percentage and number of attacks to one certain host
=================================================================
        #  of
  %    attacks   to                method
=================================================================
21.53    31      217.230.254.214  ICMP PING speedera : {ICMP}
18.75    27      217.230.248.153  WEB-IIS cmd.exe access : {TCP}
13.19    19      217.5.61.66      ICMP PING NMAP : {ICMP}
11.81    17      217.230.248.153  HTTP Request {TCP}
 9.03    13      217.5.50.16      ICMP PING NMAP : {ICMP}
 9.03    13      217.5.50.16      ICMP PING CyberKit 2.2 Windows : {ICMP}
 5.56    8       217.5.61.66      ICMP PING CyberKit 2.2 Windows : {ICMP}
 3.47    5       217.230.254.214  ICMP PING CyberKit 2.2 Windows : {ICMP}
 1.39    2       217.230.248.153  WEB-IIS CodeRed v2 root.exe access : {TCP}
 1.39    2       217.230.248.153  ICMP PING CyberKit 2.2 Windows : {ICMP}
 1.39    2       217.5.61.66      SCAN Proxy attempt : {TCP}
 1.39    2       217.230.248.153  WEB-FRONTPAGE /_vti_bin/ access : {TCP}


The distribution of attack methods
===============================================
        #  of
  %    attacks   method
===============================================
22.22    32      ICMP PING NMAP
21.53    31      ICMP PING speedera
19.44    28      ICMP PING CyberKit 2.2 Windows
18.75    27      WEB-IIS cmd.exe access
11.81    17      HTTP Request {TCP}
                 15    212.21.228.22   -> 217.230.248.153
                 2     217.230.115.132 -> 217.230.248.153
 1.39    2       ICMP superscan echo
 1.39    2       WEB-FRONTPAGE /_vti_bin/ access
 1.39    2       WEB-IIS CodeRed v2 root.exe access
 1.39    2       SCAN Proxy attempt
```

2.2.2 - SNID - SCANLOGD

Description:

`scanlogd` is a daemon that watches for port scans. Available on SuSE CDs

Port Scans detected:

It will detect the incoming port scans if:

- 7 different privileged ports (1-1023) or
- 21 different non-privileged ports (1024-65535) or
- a combination of the above is detected.

If the above conditions are detected within less than 3 seconds then it is called a 'scan'. If 5 scans are detected within 20 seconds then it will be logged as: `daemon.alert` (syslog priority).
The the logging will then be stopped temporarily.

Example of an `/etc/syslog.conf` entry:

```
daemon.=alert  /var/log/scanlogd.log
```

The deamon is started as root. It will then switch to chroot() mode in the directory `/var/empty` and run as user `scanlogd`.

Configuration: Create the system user : `scanlogd`

```
useradd scanlogd
```

Start service: as root : `scanlogd`

Results: One line (in log file)per notification in the following format:

```
saddr[:sport]  to  daddr  [and others,] ports port[, port...],
..., flags[, TOS TOS][, TTL TTL] @HH:MM:SS
```

NOTE: `scanlogd` will NOT record the scans from its own host where it runs.

Testing scanlogd: from another host issue the command:

`nmap -sS` *`Hostname`*

(*`Hostname`*=host where the `scanlogd` runs)

To see the log: `tail -f /var/log/messages | grep "scanlogd"`

2.3 - Network Diagnostics Tools (86_Network_Diagnostics)

2.3.1 - Packet sniffers

`ethereal`	Graphic packet sniffer Displays everything about each packet. Capable of setting capture and display filters
`tethereal`	Terminal oriented packet sniffer Displays the packet headers. Capable of setting capture filters Part of the ethereal package.
`ngrep`	Terminal oriented packet sniffer. No more on SuSE CD ...:-(Displays packet addresses and port ans packet content. Capable of text search and capture filters
`tcpdump`	Versatile Terminal oriented packet sniffer Displays packet headers, content etc. Capable of setting capture filters
`iptraf`	Semi graphic (addressed Text screens) packet ' and connection sniffer. Displays Connection statistics, TCP connections, Minimum packet headers.
`tleds`	Flickers the keyboard LEDs on reception and transmission of packets. Left=receiving, Right=transmitting
`telnet`	Allows to test certain TCP services using the port parameter. eg. `telnet www.mywebserver.com 80`
`netcat`	Allows to test certain UDP services using the port parameter. eg. `netcat  www.mynfsserver.com 2049`

2.3.2 - Network checkers

`ping`	Verify if a host is on line.
`fping`	Sends pings to a list of hosts
`sping`	Script that pings a whole Class 'C' subnet.
`pping`	Script that verifies the presence of open ports

sping script

```
#! /bin/sh
#------------------- Super Ping : sping --------------------------
# File:          sping
# Purpose:       ping a full sub-net (netmask 255.255.255.0)
# Date:          07.09.2003
# Author:        Michel Bisson
# Syntax:        sping 71
#                this number(eg. 71) is the third part of an IP subnet.
#                eg. 192.168.71.0)
#-------------------------------------------------------------
# Declare error function using the Variable $errortext:
# Displays error message and exits with error code 1
error () {
      echo "$errortext"
      echo "Syntax: eg. $0 71 "
      exit 1
}
# --------Check the validity of the given parameter (Partial IP)
# Only one parameter
if [ "$#" -ne 1 ] ; then
      errortext="ERROR: Incorrect number of parameters."
      error
fi
# ----------Parameter is a correct Partial IP? Correct it if possible
# Accept if xxx
if ! (echo $1 | egrep '^[0-9]{1,3}$' &>/dev/null) ; then
      errortext="ERROR: Bad subnet Partial IP Syntax"
      error
fi

# Verify validity if all numbers in IP (0-255)
if [ "$1" -gt 255 ] ; then
      errortext="ERROR: Too high values in partial IP."
      error
fi

# Check if subnet IP is valid for local machine in network
if ! (/sbin/ifconfig | grep 192.168.$1 &>/dev/null) ; then
      errortext="ERROR: Partial Subnet IP is not in our local subnet"
      error
fi
allips=""
i=1
while [ "$i" -lt 255 ] ; do
    allips="$allips 192.168.$1.$i"
    let i++
done
#-------- Generate all the IPs from xx.xx.xx.1 to xx.xx.xx.254--
#----- Create or Empty the temporary file and counter variable---------
netnum=1
> /tmp/sping

#------- ping them all almost at the same time (sent as separate jobs)
for ip in $allips; do
    /bin/ping -w1 -c1 $ip 1>>/tmp/sping &
    let netnum++
done
#----------- wait a bit to let some answers back ----------
sleep 2

#----------- Kill all the ping that are still waiting ---
killall ping &> /dev/null
#----------- Show only the pings that received an answer --
grep '64 bytes from' /tmp/sping | cut -d: -f1 | cut -d" " -f4 \
| sort -t. -k4n
exit 0
```

pping script

```
#!/bin/bash
#------------ Port Pinger -----------
# Name:    pping
# Purpose: Pings ports of hosts by sending SYN packets
#          and waiting for an answer
# Syntax1: pping HostAddr PortNr
# Syntax2: pping NetAddr/CIDRNetmask PortNr
# Author:  Michel Bisson (www.linuxint.net)
# Date:         04.10.2004
# Note:         Needs to be ROOT and uses nmap port scanner program
#-----------------------------------
#----------- Allow only 2 arguments. No more no less.
if [ "$#" -ne 2 ] ; then
    echo "ERROR: Bad number of arguments. Must be 2 arguments"
    echo "Syntax: pping HostAddr PortNr or pping NetAddr/Netmask
PortNr"
    exit 1
fi
#----------- Allow only numeric Port Number --------------------

#-- Use the numeric port number or find the port number
in /etc/services
if ! ( echo $2 | egrep "^[0-9]{1,5}$" &>/dev/null ) ; then
    port=$(expand /etc/services | grep "^$2 .*/tcp" | awk '{ print
$2 }' | cut -d/ -f1)
  if [ ! "$port" ] ; then
    echo "ERROR: Bad port number or port name"
    echo "Syntax: pping HostAddr PortNr or pping NetAddr/Netmask
PortNr"
    exit 1
  fi
else
    port=$2
fi

#------
if ( echo $1 | grep "/" &>/dev/null ) ; then
    # -------- Filter the result so that I get only one line per Host
    result=$(nmap -n --host_timeout 210 --max_rtt_timeout 210 -p $port
-P0 $1 | egrep "Interesting|open" | sed 's/$/#/')
    IFS="#"
    for i in $result ; do
     if (echo $i | grep "Interesting" &>/dev/null) ; then
         IP=$(echo -n $i | awk '{ printf $4 }')
     else
         echo  -n $i : $IP | sed -e 's/    open       //' -e 's/:$//'
     fi
    done
else
  # --------- Show the result of the host poking on that port -------
  nmap -n --host_timeout 210 --max_rtt_timeout 210 -p $port -P0 $1 \
     | grep "open" | sed -e 's/    open       //'
fi
echo
```

2.4 - Closing all unused open services (61_TCP_IP_Services)

- Check Open Ports:
  ```
  netstat -tunap
  ```
  Shows all the open `tcp` and `udp` ports(numeric) with Programs PID
  <u>or</u>
  ```
  lsof -Pni4
  ```
  Shows all the server and client ports numbers used in system

Options:	P=Port numbers instead of port names
	n=No reverse resolving of IP Addresses
	`i4`=IP Version 4 only

- Super Daemon `inetd` services
  - Set the values if the tcpwrapper (`tcpd`) in `/etc/hosts.allow` and `/etc/hosts.deny` to disallow some hosts access to certain inetd services.
- **Tcpwrappers**
  The tcpwrappers are programs that uses configuration files to check if the client host is allowed to use the requested service. One commonly used tcpwrapper is `tcpd`. It uses the `/etc/hosts.allow` and `/etc/hosts.deny` files for this purpose. They contain a listing of the hosts concerned for each requested service. Here is the logic: If none of the two files exists, then all of the hosts are allowed to use all watched services.

  The access control software consults two files. The search stops at the first match:
  - Access will be granted when a (daemon,client) pair matches an entry in the `/etc/hosts.allow` file.
  - Otherwise, access will be denied when a (daemon,client) pair matches an entry in the `/etc/hosts.deny` file.

  Otherwise, access will be granted.
  The command:
  ```
  tcpdchk
  ```
  examines your tcp wrapper configuration and reports all potential and real problems it can find. The program examines the tcpd access control files (by default, these are `/etc/hosts.allow` and `/etc/hosts.deny`), and compares the entries in these files against entries in the `inetd` or `tlid` network configuration files.

- **Recommendations**
  - Run this super daemon only if needed or
  - Disable all unused services as described in the following sections:

  - Super Daemon `xinetd` services
    - Run this super daemon only if needed or
    - Disable all unused services by editing the services files in `/etc/xinetd.d/` and making sure: `disable = yes`

  - Runlevels services
    Start YAST2 Run Level Editor and disable all unused services from runlevels.

2.5 - Secure Communications

2.5.1 - SSH, SCP and RSYNC (90 _SSH - SCP)

- `Telnet` and `ssh` differences
- Public and Private keys principle

  Host keys

  ```
  /etc/ssh/ssh_host_rsa_key
  /etc/ssh/ssh_host_rsa_key.pub
  ```

  User Keys

  ```
  ssh-keygen -t rsa
       ~/.ssh/id_rsa
       ~/.ssh/id_rsa.pub
       ~/.ssh/known_hosts
       ~/.ssh/authorized_keys
  ```

- Tunelling one port using `ssh`

  ```
  ssh -2 sun.linux.local -L 7772:sun.linux.local:80
  ```

- **scp** and **rsync**:

  ```
  scp -prv server:/path    localpath
  rsync -e ssh -vazuH server:path localpath
  ```

- Remote Admin using X-Programs and `ssh` tunelling

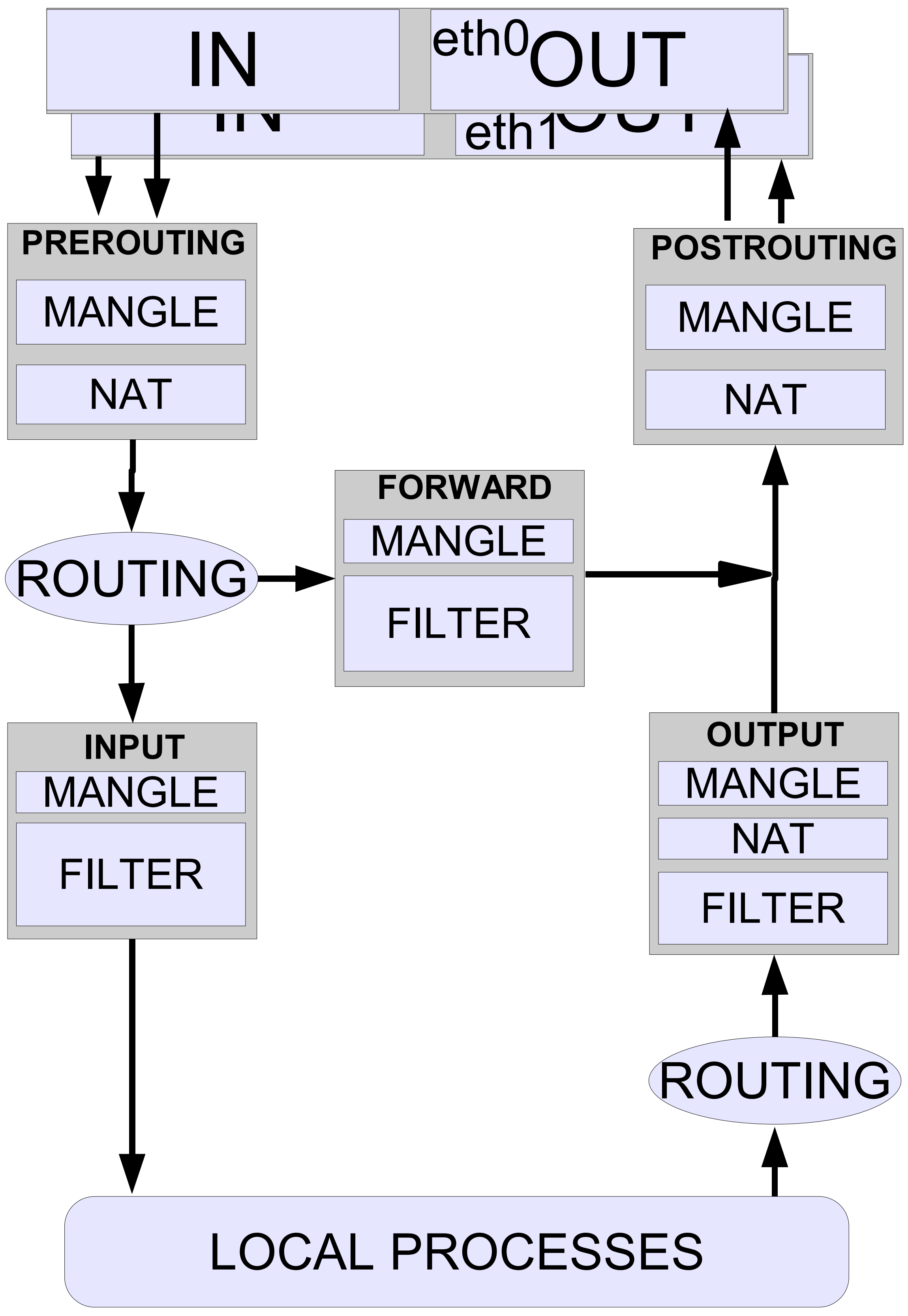
  ```
  xhost +
  ssh -X user@remoteserver
  remote-program
  ```

2.5.2 - VPN - Virtual Private Networking See document `99_VPN.sxw`

2.6 - Firewalls:

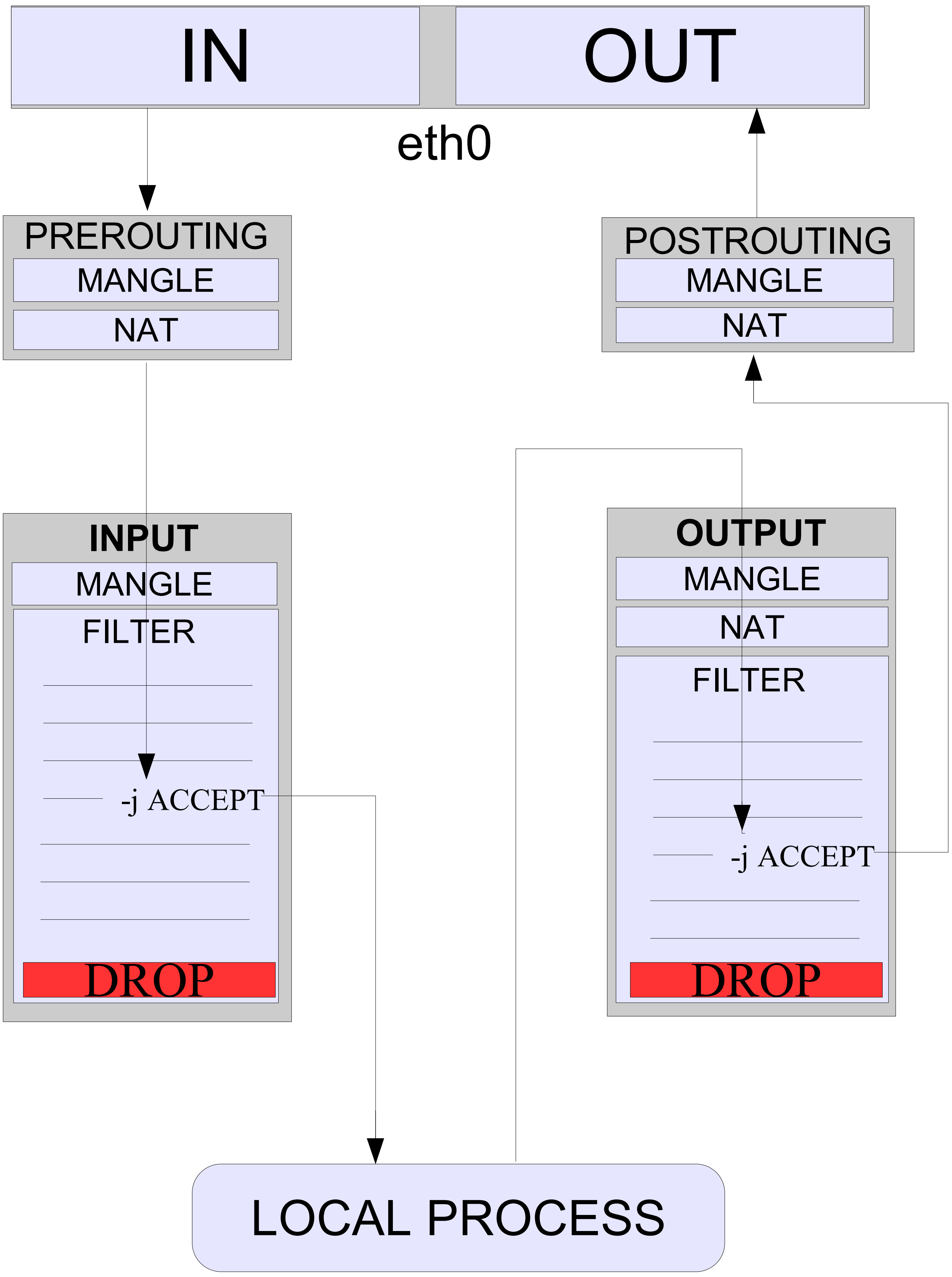
2. **- Firewall basics: IPTABLES Flowchart**

IPTABLES Flowchart



- **IPTABLES - Packet travel for local processes**

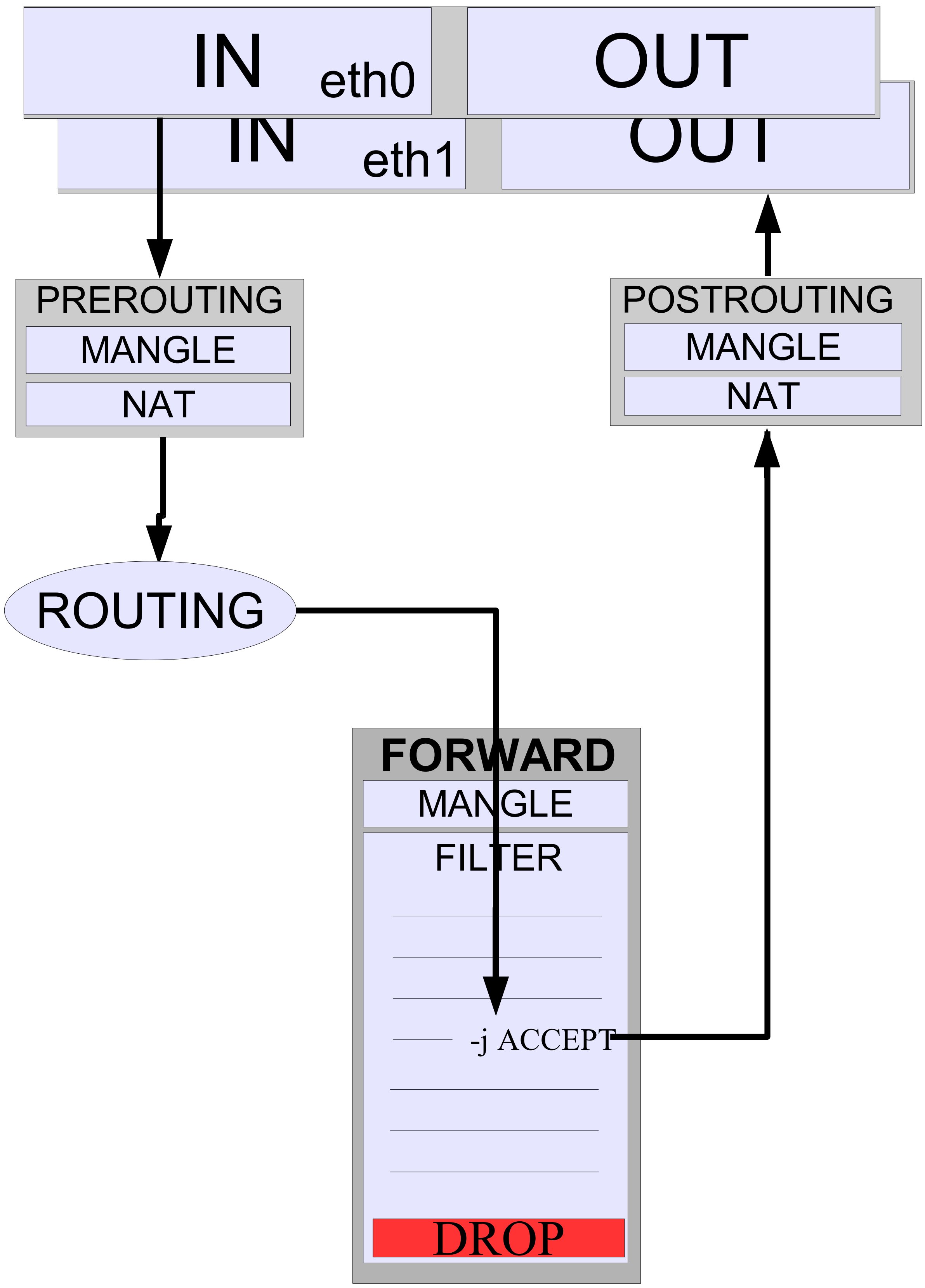
IPTABLES - Packet travel through rule chains for local processes. (Policies set to DROP)



- **IPTABLES - Packet travel for Forwarded packets**

IPTABLES - for **Forwarded** packets.

(Policy set to DROP)



- Flow chart of `ipchains` and `iptables`
- 5 Rule tables**:** Prerouting, Input, Forward, Output,Postrouting,
- Programming the Firewall
  - Copying Firewall basic init script (`Klasse_Firewall`)
  - In `Klasse_Firewall` script:
    - Loading the iptables module (not needed from SuSE 8.2)
    - (1)Local Variables: LAN, proxy address, proxy port, etc
    - (2)Tables Policies to DROP for `start`)
      and ACCEPT for `stop`)
    - (3)Emptying the tables: `iptables -F`
    - (4)Opening the `lo` (INPUT & OUTPUT)
    - 5)Opening SSH Client (with static rules)
    - (6)Stateful inspection for communications(INPUT & OUTPUT)
    - (7)Opening SSH Client (with Stateful Inspection rules)
    - (8)Logging all packets before end DROP Policies
      ((INPUT & OUTPUT)
    - (9)Opening all other Local LAN client services(OUTPUT):
      - DNS, CUPS, CUPS Server
    - (10)Opening all needed Internet client services(OUTPUT):
      - Proxy, HTTP, HTTPS
    - (11)Opening Client FTP(passive) and FTP Server
    - (15)Opening some ICMPs(transfer text from dozent?)
    - (12)Dropping INPUT attacks and unnecessary packets:
      Stealth Scans,INVALID, Full Broadcasts(DHCP).
    - (11)Together with Class(using sniffers) Windows shares
    - (16)**Exercise:** Discovering and Entering:
      HTTP Server,SMTP,POP,NFS.

- **Testing services through the firewall**
  - **ipp Client(cups):**
    ```
    telnet laptop 631 <enter>
    GET / <enter>
    <enter>
    ```
  - **http Client:**
    ```
    telnet laptop 80 <enter>
    GET / <enter>
    ```
  - **smtp Client**
    ```
    telnet laptop 25 <enter>
    HELO laptop <enter>
    QUIT <enter>
    ```
  - **pop3 Client**
    ```
    telnet laptop 110 <enter>
    QUIT <enter>
    ```
  - `http` and `ipp` Servers are tested via clients running the above commands towards the server host to test.
    eg.
    http server:
    ```
    telnet httpServerAddr 80 <enter>
    GET / <enter>
    ```
    ippp server:
    ```
    telnet cupsServerAddr 80 <enter>
    GET / <enter>
    <enter>
    ```

2.6.2 - Masquerading

- Masquerading and Proxy principle (NAT)
- Set `FORWARD` rules
- Turn ON `IP_FORWARDING` in kernel
  ```
  echo 1 > /proc/sys/net/ipv4/ip_forward
  ```
- iptables `POSTROUTING NAT` table rules:
  - `SNAT` for static Internet IPs(faster):
    ```
    iptables -t nat -A POSTROUTING -o eth1 \
    -j SNAT --to-source $ROUTER_INET_IP
    ```
  - `MASQUERADE` for dynamic IPs(clean itself on interface-down):
    ```
    iptables -t nat -A POSTROUTING -o eth1 \
    -j MASQUERADE
    ```
- Testing `MASQUERADING`:

	Ping source addr?
client --(ping)------> router(no masq)------>Client:	Client addr.
client --(ping)------> router(masq)---------->Client:	Router addr.

- **2.6.3 - REDIRECTION**

  - DMZ Principle
  - Set FORWARD rules
  - Turn ON IP_FORWARDING in kernel

– iptables PREROUTING NAT table rule:

Sending all incoming web requests from internet clients to the web server located in DMZ:
```
iptables -t nat -A PREROUTING -p tcp -i $ext_if \
--dport 80 -j DNAT --to-destination $DMZ_HTTP_SERVER
```

Re-routing web server requests from Internet to a web server port 8080 in DMZ:
```
iptables  -t nat -A PREROUTING -p tcp -d $INET_IP \
--dport 80 -j DNAT --to-destination $DMZ_HTTP_SERVER:8080
```

NOTE: the following 2 rules MUST be added to allow the same DMZ web server to be called from the internal network where the web server is,or from the firewall itself:

Rule to allow a local network host to call the DMZ web server:
```
iptables -t nat -A POSTROUTING -d $DMZ_HTTP_SERVER \
          --dport 80 -j SNAT --to-source $FIREWALL_LAN_IP
```

Rule to allow requests direct from the firewall to the DMZ web server:
```
iptables -t nat -A OUTPUT --dst $INET_IP --dport 80 \
          -j DNAT --to-destination $DMZ_HTTP_SERVER
```

2.6.4 - Load balancing for DMZ servers:

Allows to random balance the load of DMZ servers offering all the same service.
Syntax:
Re-routing web server requests from Internet to multiple web servers in DMZ
```
iptables  -t nat -A PREROUTING -p tcp -d $INET_IP --dport 80 \
          -j DNAT --to-destination 192.168.70.1-192.168.70.10
```

2.6.5 - Logging level:

The `--log-level` option allows to change the logging level from `kern.warn` to another level like `kern.debug`:
Syntax:
```
  iptables -A INPUT -j LOG --log-level DEBUG \
                   --log-prefix "END of INPUT CHAIN == "
```

2.6.6 - TTL modification or detection:

The TTL target allows to modify the existing TTL of packets to hide the fact that the clients are residing inside a private network.
Syntax:
```
  iptables -t mangle -A PREROUTING -i eth0 \
                                   -j TTL --ttl-set 64
```
To detect packets with a certain TTL then:
```
  iptables -A INPUT -m ttl --ttl 60 -j DROP
```

2.6.7 - Matching packet owner(user):

This allows to detect which local user has generated the packet and decide what to do with it. This feature is only accessible in the OUTPUT chain
(for local processes)
Syntax:

```
UID:  iptables -A OUTPUT -m owner --uid-owner 500 -j ACCEPT
GID:  iptables -A OUTPUT -m owner --gid-owner 0 -j ACCEPT
PID:  iptables -A OUTPUT -m owner --pid-owner 2056 -j ACCEPT
SID:  iptables -A OUTPUT -m owner --pid-owner 2056 -j ACCEPT
```

(SID is a Session ID based on parent process threads)
see page 51 of `iptables-tutorial.html`

2.6.8 - Matching multiple ports in one rule:

This feature allows us to list up to 15 ports.
Can only be used with `-p tcp` or `-p udp.`
Syntax:

```
iptables -A INPUT -p tcp -m multiport \
          --source-port 22,53,80,110,443

iptables -A INPUT -p tcp -m multiport \
          --destination-port 22,53,80,110,443

iptables -A INPUT -p tcp -m multiport \
          --port 22,53,80,110,443
```

2.6.9 - Limiting frequence of ping or logged packets

This feature allows to limit the throughput of accepted packets

```
 iptables -A INPUT -i eth0 -p icmp --icmp-type echo-request \
                    -m limit --limit 5/minute -j ACCEPT
```

Limit can be: '`/second`', '`/minute`', '`/hour`', or '`/day`' suffix; the default is `3/hour`

2.6.10 - Matching MAC Adresses:

This allows to match Ethernet cards MAC (Hardware) addreses in rules:
Syntax:

```
iptables -A INPUT -m mac --mac-source 00:02:80:2F:56:82
iptables -A INPUT -m mac --mac-destination 00:02:80:2F:56:82
```

2.6.11 - Changing the priority of packets : TOS target

Changing the TOS(Type Of Service) of packets allows to control the different incoming packet queues.
The standard **pfifo_fast** scheduler (classless queue)has 3 different 'bands':
Traffic in band 0 is transmitted first, then the traffic in band 1 then in band 2.
No packets from band 1 are sent until the band 0 is empty, and the same goes for band 2 vs band 1. So it is vital that our interactive traffic be in band 0.

To achieve the band assignment of packets we need to change their TOS. There are four seldom-used bits in the IP header, called the Type of Service (TOS) bits. They effect the way packets are treated; the four bits are "Minimum Delay", "Maximum Throughput", "Maximum Reliability" and "Minimum Cost". Only one of these bits is allowed to be set. The change of the TOS of packets is done via `iptables` rules:
eg.

```
iptables -t mangle -A PREROUTING -p TCP --dport 22 -j TOS --set-tos 0x10
```

Table of TOS codes to Band number for the *pfifo_fast* classless queue:

TOS	Bits	Means	Linux Priority	Band
0x0	**0**	**Normal Service**	**0 Best Effort**	**1**
0x2	**1**	**Minimize Monetary Cost**	**1 Filler**	**2**
0x4	**2**	**Maximize Reliability**	**0 Best Effort**	**1**
0x6	3	mmc+mr	0 Best Effort	1
0x8	**4**	**Maximize Throughput**	**2 Bulk**	**2**
0xa	5	mmc+mt	2 Bulk	2
0xc	6	mr+mt	2 Bulk	2
0xe	7	mmc+mr+mt	2 Bulk	2
0x10	**8**	**Minimize Delay**	**6 Interactive**	**0**
0x12	9	mmc+md	6 Interactive	0
0x14	10	mr+md	6 Interactive	0
0x16	11	mmc+mr+md	6 Interactive	0
0x18	12	mt+md	4 Int. Bulk	1
0x1a	13	mmc+mt+md	4 Int. Bulk	1
0x1c	14	mr+mt+md	4 Int. Bulk	1
0x1e	15	mmc+mr+mt+md	4 Int. Bulk	1

iptables command for TOS control

Option	**`--set-tos`**
Example	**`iptables -t mangle -A PREROUTING -p TCP --dport 22 -j TOS --set-tos 0x10`**
Descrip.	The **--set-tos** option tells the **TOS** mangler what TOS value to set on packets that are matched. The option takes a numeric value, either in hex or in decimal value. As the TOS value consists of 8 bits, the value may be 0-255, or in hex 0x00-0xFF. Note that in the standard TOS target you are limited using the named values available (which should be more or less standardized), as mentioned in the previous warning. These values are: `Normal-Service` (decimal value 0, hex value 0x00) `Minimize-Cost` (decimal value 2, hex 0x02) `Maximize-Reliability` (decimal value 4, hex value 0x04) `Maximize-Throughput` (decimal value 8, hex value 0x08) `Minimize-Delay` (decimal value 16, hex value 0x10) The default value on most packets are `Normal-Service`, or 0. Note that you can, of course, use the actual names instead of the actual hex values to set up the TOS value, and it should generally be recommended since the values behind the names may be changed if you are unlucky. For a complete listing of the "descriptive values", do an **`iptables -j TOS -h`**. This listing is complete as of iptables 1.2.5 and should hopefully be so for another period of time.

To view the number of packets being modified by the TOS:

```
     iptables -t mangle -L PREROUTING -v -x 2> /dev/null
```

dsl_qos_queue

This program implements a user space QUEUE processor which controls outbound traffic over a DSL modem using a pseudo- token-bucket-filter style queue with starvation protection.

Main feature of this outbound rate limiter is it's ability to rate limit based on the calculated raw bandwidth used rather than just the IP bandwidth used. This provides a MUCH more accurate way to prevent packet queuing in the network device (in this case, ADSL modem). Please report any problems you have to me.
This has been tested on Redhat 7.3. You must be using `iptables`.
You must have `libipq` installed to compile. (attempting to avoid this requirement by supplying my compiled `libipq.a` in the distribution package.)

Package now includes a compiled binary for i386 Linux 2.4.x
Version 0.9.2 updated to improve stability. See README for more details.
Download here: `dsl_qos_queue-0.9.2.tar.gz` - v0.9.2

If you're trying to control bandwidth in Windows, try this:
`http://www.bandwidthcontroller.com`

Full implementation of TOS modifications

```
I have left a few sections commented out, as im not sure if I need them.

${IPTABLES} -t mangle -A PREROUTING -p tcp --dport 23 -j TOS --set-tos Minimize-Delay
${IPTABLES} -t mangle -A PREROUTING -p tcp --sport 23 -j TOS --set-tos Minimize-Delay
${IPTABLES} -t mangle -A PREROUTING -p tcp --dport 22 -j TOS --set-tos Minimize-Delay
${IPTABLES} -t mangle -A PREROUTING -p tcp --sport 22 -j TOS --set-tos Minimize-Delay
${IPTABLES} -t mangle -A PREROUTING -p tcp --dport 20 -j TOS --set-tos \
                                                                 Maximize-Throughput
${IPTABLES} -t mangle -A PREROUTING -p tcp --sport 20 -j TOS --set-tos \
                                                                 Maximize-Throughput
${IPTABLES} -t mangle -A PREROUTING -p tcp --dport 21 -j TOS --set-tos Minimize-Delay
${IPTABLES} -t mangle -A PREROUTING -p udp --dport 53 -j TOS --set-tos Minimize-Delay
${IPTABLES} -t mangle -A PREROUTING -p tcp --dport 53 -j TOS --set-tos Minimize-Delay
${IPTABLES} -t mangle -A PREROUTING -p tcp --dport 25 -j TOS --set-tos Minimize-Cost
${IPTABLES} -t mangle -A PREROUTING -p tcp --dport 110 -j TOS --set-tos Normal-Service
${IPTABLES} -t mangle -A PREROUTING -p icmp -j TOS --set-tos Minimize-Delay
${IPTABLES} -t mangle -A PREROUTING -p tcp --dport 5190 -j TOS --set-tos Minimize-Delay
${IPTABLES} -t mangle -A PREROUTING -p tcp --dport 4000 -j TOS --set-tos Minimize-Delay
${IPTABLES} -t mangle -A PREROUTING -p udp --dport 4000 -j TOS --set-tos Minimize-Delay
${IPTABLES} -t mangle -A PREROUTING -p tcp --dport 6666:6667 -j TOS \
                                                               --set-tos Minimize-Delay
${IPTABLES} -t mangle -A PREROUTING -p tcp --dport 1214 -j TOS --set-tos Minimize-Cost
${IPTABLES} -t mangle -A PREROUTING -p tcp --dport 6699 -j TOS --set-tos Minimize-Cost
${IPTABLES} -t mangle -A PREROUTING -p tcp --dport 6700 -j TOS --set-tos Minimize-Cost
${IPTABLES} -t mangle -A PREROUTING -p tcp --dport 1412 -j TOS \
                                                           --set-tos Maximize-Throughput
${IPTABLES} -t mangle -A PREROUTING -p tcp --sport 1412 -j TOS \
                                                           --set-tos Maximize-Throughput
${IPTABLES} -t mangle -A PREROUTING -p udp --dport 4000:7000 -j TOS \
                                                           --set-tos Minimize-Delay
${IPTABLES} -t mangle -A PREROUTING -p tcp --sport 80 -j TOS --set-tos Minimize-Cost

-----------------------------------------------------------------------------------
```

I have also put together a simple but quite effective QOS script which you may want to put into your /etc/ppp/ip-up script

This script was put together from a number of other scripts I found on the internet.
I'm SURE it could be improved on as I learnt about Linux QOS about 3 hours ago and its my first real attempt, but it does seem to work quite well. With a large FTP upload going, real-time traffic is just about real-time and webpages come down with only a small delay.

The basic idea with the firewall and QOS script is to generate 3 queues.
Real-time: For all protocols which transfer small amounts of data and need to be as close to real-time as possible.
Normal: This is for normal traffic www packets, pop etc.
Bulk Traffic: For all protocols which you don't want affecting your normal day to day web browsing. Outgoing mail, ftp uploads, p2p programs like kazza or DC.

I'm not doing any QOS on downloaded packets as I feel its not very effective unless the ISP is doing the QOS. If anyone has any suggestions on how to make these scripts better.. im more then happy to give it a go.

```
#!/bin/bash
#
# Set the following values to somewhat less than your actual download
and uplink speed. In kilobits
# I have a 256k down 64k up ADSL. 55 seem to work quite well..
UPLINK=55
DEV=ppp0

# clean existing down and uplink qdiscs, hide errors
tc qdisc del dev $DEV root 2> /dev/null > /dev/null
tc qdisc del dev $DEV ingress 2> /dev/null > /dev/null

# install root HTB, point default traffic to 1:20:
tc qdisc add dev $DEV root handle 1: htb default 20

# shape everything at $UPLINK speed - this prevents huge queues in your
DSL modem.
tc class add dev $DEV parent 1: classid 1:1 htb rate ${UPLINK}kbit burst 6k
# high prio class 1:10:
tc class add dev $DEV parent 1:1 classid 1:10 htb rate ${UPLINK}kbit
burst 6k prio 1

# Default class 1:20 - gets slightly less traffic, and a lower priority:
tc class add dev $DEV parent 1:1 classid 1:20 htb rate
$[9*$UPLINK/10]kbit burst 6k prio 2

# Bulk - ftp outgoing mail etc.
tc class add dev $DEV parent 1:1 classid 1:30 htb rate
$[9*$UPLINK/10]kbit burst 6k prio 3

# both get Stochastic Fairness:
tc qdisc add dev $DEV parent 1:10 handle 10: sfq perturb 10
tc qdisc add dev $DEV parent 1:20 handle 20: sfq perturb 10
tc qdisc add dev $DEV parent 1:30 handle 30: sfq perturb 10

# TOS Minimum Delay (ssh, NOT scp) in 1:10:
tc filter add dev $DEV parent 1:0 protocol ip prio 10 u32 \
    match ip tos 0x10 0xff flowid 1:10

# ICMP (ip protocol 1) in the interactive class 1:10
tc filter add dev $DEV parent 1:0 protocol ip prio 10 u32 \
    match ip protocol 1 0xff flowid 1:10

# To speed up downloads while an upload is going on, put ACK packets in
the interactive class:
tc filter add dev $DEV parent 1: protocol ip prio 10 u32 \
    match ip protocol 6 0xff \
    match u8 0x05 0x0f at 0 \
    match u16 0x0000 0xffc0 at 2 \
    match u8 0x10 0xff at 33 \
    flowid 1:10

# Bulk Filter
tc filter add dev $DEV parent 1:0 protocol ip prio 10 u32 \
    match ip tos 0x2 0xff \
    flowid 1:30

tc filter add dev $DEV parent 1:0 protocol ip prio 10 u32 \
    match ip tos 0x8 0xff \
    flowid 1:30
```

NOTE: you will need to patch your kernel with the htb patch for the above queues to work. I'm also told its already in the 2.4.20-pre x kernels You can also monitor the queues with the command

```
watch -n 1 tc -s qdisc show dev ppp0
```

2.6.12 - Firewall GUIs

- `webmin` (Very good)
- `fwbuilder`(Object oriented)
- *jay's firewall generator*

2.7 - Proxy Server 'Squid'

2.7.1 - Installation

Installation package '`squid`' is available on SuSE CDs

2.7.2 - squid Configuration

- Access Control:
  - `acl` declarations:
    - Aliases of `src`, `dstdomain` (rest is in squidGuard below)
    - `http_access` directives

```
acl verboten  dstdomain .whitehouse.com .ebay.de .ebay.com
acl fingerweg dstdom_regex porn sex strip hardcore
http_access deny verboten
http_access deny fingerweg
```

- Proxying behind another proxy:(Around line 558)
  (Parent Proxy: `192.168.10.1`)

```
cache_peer 192.168.10.1 parent 3128 3130 proxy-only no-query
```

- Cache control directives:

```
cache_dir ufs /var/cache/squid 100 16 256
```

100 = 100 MB Maximum Cache on hard disk
16 = 16 Subdirectories (level 1)
256 = 256 Subdirectories (level 2) per subdir of level 1

```
cache_mem  16 MB
maximum_object_size 4096
```

- Bandwidth control

How can I assign different bandwidth to different squid clients?
Squid has the possibility to limit the `http/https/ftp` bandwidth via 'delay pools'. The procedure is as follows:
- Assignment of `acl` aliases to groups of hosts.
- Create 'delay pools' and assign a class type to each pool.
- Assign bandwidth parameters to each pool
- Assign each acl group to a 'delay pool'.

- Recreating the squid cache:

```
squid -z
```

Example No1:
We want to assign two bandwidth:
- 512kbit/s(64000 Bytes/s) for the office hosts
- 128kbit/s(16000 Bytes/s) for hosts of rest of the company

Configuration in `/etc/squid.conf`:

```
acl office_hosts src 192.168.71.10 192.168.71.44 192.168.71.242
acl intranet src 192.168.71.0/255.255.255.0

# Create 2 delay pools
delay_pools 2
```

```
# Assign a class type to each delay pool
#   Class 1 has class type 2
delay_class 1 2
#   Class 2 has class type 2
delay_class 2 2
# Assign bandwidth to each pool
# Pool 1 get no overall limit(-1/-1) but each host is limited to max 64KBytes/sec.
delay_parameters 1 -1/-1 64000/64000
# Pool 2 get no overall limit(-1/-1) but each host is limited to max 16KBytes/sec.
delay_parameters 2 -1/-1 16000/16000
# Assign each pool to the hosts groups
delay_access 1 allow office_hosts
delay_access 2 allow intranet
```

Example No2:
We want to assign a single bandwidth limit:
- 8KBytes/s to a all hosts in 2 subnets (192.168.70.0/24 and 192.168.71.0/24)
- No bandwidth limit for some special hosts in those 2 subnets.

Configuration in `/etc/squid.conf`:

```
# Assign acl to host groups
acl intranets src 192.168.70.0/255.255.254.0
acl special_hosts src 192.168.71.10 192.168.71.42 192.168.71.44\
     192.168.71.130 192.168.70.10 192.168.70.130 192.168.71.201
# Create 1 delay pool
delay_pools 1
# Assign the class 2 to pool 1
delay_class 1 2
# Assign bandwidth parameters to pool 1
# Pool 1 gets no overall limit(-1/-1) but each host is limited to 8KBytes/s
delay_parameters 1 -1/-1 8000/8000
# Assign pool 1 to hosts group(acls)
# Notice that special_hosts are simply denied access to the pool1 therefore getting
no bandwidth limit. To achieve this, the deny rule must be declared first.
delay_access 1 deny special_hosts
delay_access 1 allow intranets
```

This above examples are some of the most useful configurations I've found so far.
The following squid configuration file of SuSE 9.x explains in more details the different possibilities.

- Extract of Delay Pools from /etc/squid.conf

```
# DELAY POOL PARAMETERS (all require DELAY_POOLS compilation option)
#-----------------------------------------------------------------------

#  TAG: delay_pools
#     This represents the number of delay pools to be used.  For example,
#     if you have one class 2 delay pool and one class 3 delays pool, you
#     have a total of 2 delay pools.
#
#     To enable this option, you must use --enable-delay-pools with the
#     configure script.
#
#Default:
```

```
# delay_pools 0

#  TAG: delay_class
#     This defines the class of each delay pool.  There must be exactly one
#     delay_class line for each delay pool.  For example, to define two
#     delay pools, one of class 2 and one of class 3, the settings above
#     and here would be:
#
#Example:
#+++++++++++++++++++++++++++++++++++

# delay_pools 2      # 2 delay pools
# delay_class 1 2    # pool 1 is a class 2 pool
# delay_class 2 3    # pool 2 is a class 3 pool
#
#     The delay pool classes are:
#
#     class 1        Everything is limited by a single aggregate bucket.
#
#     class 2        Everything is limited by a single aggregate  bucket
#                    as well as an "individual" bucket chosen
#                       from bits 25 through 32 of the IP address.
#                       (Individual hosts in a class C network)
#
#     class 3        Everything is limited by a single aggregate bucket
#                    as well as a "network" bucket chosen
#                       from bits 17 through 24 of the IP address
#                       (Overall limit for each Class C subnet)
#                    and a "individual" bucket chosen
#                       from bits 17 through 32 of the IP address.
#                       (Individual limit for each host in a class C network
#
#     NOTE: If an IP address is a.b.c.d
#                           -> bits 25 through 32 are "d"
#                           -> bits 17 through 24 are "c"
#                           -> bits 17 through 32 are "c * 256 + d"
#
#Default:
# none

#  TAG: delay_access
#     This is used to determine which delay pool a request falls into.
#     The first matched delay pool is always used, i.e., if a request falls
#     into delay pool number one, no more delay are checked, otherwise the
#     rest are checked in order of their delay pool number until they have
#     all been checked.  For example, if you want some_big_clients in delay
#     pool 1 and lotsa_little_clients in delay pool 2:
#
#Example:
# delay_access 1 allow some_big_clients
# delay_access 1 deny all
# delay_access 2 allow lotsa_little_clients
# delay_access 2 deny all
#
#Default:
# none
#  TAG: delay_parameters
#     This defines the parameters for a delay pool.  Each delay pool has
#     a number of "buckets" associated with it, as explained in the
#     description of delay_class.  For a class 1 delay pool, the syntax is:
#
```

```
#delay_parameters pool aggregate
#
#     For a class 2 delay pool:
#
#delay_parameters pool aggregate individual
#
#     For a class 3 delay pool:
#
#delay_parameters pool aggregate network individual
#
#     The variables here are:
#
#     pool         a pool number.
#                  ie, a number between 1 and the number specified in
#                  delay_pools as used in delay_class lines.
#
#     aggregate    the "delay parameters" for the aggregate bucket
#                  (Overall bandwidth for the sum all hosts of the pool)
#                  (class 1, 2, 3).
#
#     network      the "delay parameters" for the network buckets
#                  (class 3 only).
#
#     individual   the "delay parameters" for the individual buckets
#                  (Bandwidth for each individual host)
#                  (class 2 and 3 only).
#
#     A pair of delay parameters is written restore/maximum, where
#                  - restore is the number of bytes (not bits - modem and
#                    network speeds are usually quoted in bits)
#                    per second placed into the bucket,
#                  - maximum is the maximum number of bytes which can be in
#                    the bucket at any time.
#
#     For example, if delay pool number 1 is a class 2 delay pool as in the
#     above example, and is being used to strictly limit each host to 64kbps
#     (plus overheads), with no overall limit, the line is:
#
#delay_parameters 1 -1/-1 8000/8000
#
#     Note that the figure -1 is used to represent "unlimited".
#
#     And, if delay pool number 2 is a class 3 delay pool as in the above
#     example, and you want to limit it to a total of 256kbps (strict limit)
#     with each 8-bit network permitted 64kbps (strict limit) and each
#     individual host permitted 4800bps with a bucket maximum size of 64kb
#     to permit a decent web page to be downloaded at a decent speed
#     (if the network is not being limited due to overuse) but slow down
#     large downloads more significantly:
#
#delay_parameters 2 32000/32000 8000/8000 600/64000
#
#     There must be one delay_parameters line for each delay pool.
#
#Default:
# none

#  TAG: delay_initial_bucket_level  (percent, 0-100)
#     The initial bucket percentage is used to determine how much is put
#     in each bucket when squid starts, is reconfigured, or first notices
#     a host accessing it (in class 2 and class 3, individual hosts and
#     networks only have buckets associated with them once they have been
#     "seen" by squid).
#Default:
# delay_initial_bucket_level 50
```

- When Squid starts:

- Squid reads the `/etc/host.conf` to find the sequence of name resolve
- If `oder hosts bind` then it takes a snapshot of the `/etc/hosts`
- it then checks if the directive dns_nameservers exist in `/etc/squid.conf`
  - if yes then it uses these DNS Servers for name resolution
  - otherwise it uses the `/etc/resolv.conf` if it exists.
  - It then starts some DNS-Serving-Daemons called [dnsserver]

- **How can I purge an object from my cache?**

Squid does not allow you to purge objects unless it is configured with access controls in `squid.conf`. First you must add something like

```
acl PURGE method purge
acl localhost src 127.0.0.1
http_access allow purge localhost
http_access deny purge
```

The above only allows purge requests which come from the local host and denies all other purge requests.

To purge an object, you can use the client program:

```
client -m PURGE http://www.miscreant.com/
```

If the purge was successful, you will see a ``200 OK'' response:

```
HTTP/1.0 200 OK
Date: Thu, 17 Jul 1997 16:03:32 GMT
Server: Squid/1.1.14
```

If the object was not found in the cache, you will see a ``404 Not Found'' response:

```
HTTP/1.0 404 Not Found
Date: Thu, 17 Jul 1997 16:03:22 GMT
Server: Squid/1.1.14
```

2.7.3 - squidGuard (Filter & redirector)

- **Installation(SuSE 8.2)**
  - Install squidGuard package from SuSE CD
  - Change the squid configuration file `/etc/squid/squid.conf`
    ```
    redirect_program /usr/sbin/squidGuard
    redirect_children 4
    ```
    (not recommended more than 4 because of memory load)
  - Enter the modifications to the domains urls and expressions:
    ```
    vi /var/lib/squidGuard/db/blacklist/domains
    vi /var/lib/squidGuard/db/blacklist/url
    vi /var/lib/squidGuard/db/blacklist/expressions
    ```
  - Create the Databases from text to fast readable format (`.db`)
    ```
    squidGuard -C /var/lib/squidGuard/db/blacklist/domains
    squidGuard -C /var/lib/squidGuard/db/blacklist/urls
    ```
    (the expressions file does not need a database file. It is read as text)
  - Give the full squidGuard file path to squid user:
    ```
    chown -R squid. /var/lib/squidGuard
    ```
  - Edit the `squidGuard.conf`
    Enter the proper variable declarations and restrictions (see example below)
  - The squidGuard log: `/var/log/squidguard/squidGuard.log`
  - The latest blacklists can be found under:
    `http://www.squidguard.org/blacklist/`

- Example of `/etc/squidguard.conf`:

```
logdir /var/log/squidGuard
dbhome /var/lib/squidGuard/db

src adults {
    ip     127.0.0.0/8
}

src kids {
    ip     192.168.100.0/24
}

dest blacklist {
    domainlist          blacklist/domains
    urllist             blacklist/urls
    expressionlist      blacklist/expressions
}

acl {
    adults {
        pass all
    }

    kids {
        pass !blacklist all
    }
    This following default section is absolutely needed. It's the fallback when refused a site.
    default {
        pass none
        redirect http://www.google.com
    }
}
#---------------------------------------------------
```

- **The *redirect* rule:**

The *redirect* rules declares the altenative URL to be used for blocked destination groups (*`!groups`*) for the actual acl.
Note: Inside an acl, this is a fallback used when there is no special redirect declared for the actual destination group, and the default redirect is the last resort.

squidGuard can do runtime string substitutions in the redirectors. Therefore the character "`%`" has special meaning in the redirector URLs:

`%a` is replaced with IP address of the client.
`%n` is replaced with the domainname of the client or "`unknown`" if not available.
`%i` is replaced with the user ID (RFC931) or "`unknown`" if not available.
`%s` is replaced with the matched source group (client group) or "`unknown`" if no groups were matched.
`%t` is replaced with the matched destination group (target group) or "`unknown`" if no groups were matched.
`%u` is replaced with the requested URL.
`%p` is replaced with the REQUEST_URI, i.e. the path and the optional query string of `%u`, but **note** for convenience without the leading "`/`".
`%%` is replaced with a single "`%`".

Thus you can pass useful information to a more or less intelligent CGI page:

```
http://proxy/cgi/squidGuard?clientaddr=%a&clientname=%n&clientident=
%i&clientgroup=%s&destinationgroup=%t&url=%u
```

For a start, there is a sample of such a script in `samples/squidGuard.cgi` in the source tree.

Domain lists

The domainlist file format is simply domainnames/zonenames separated by a newline. The length of these lists have neglectable influence on the performance.

For instance a start for a financial category:

```
amex.com
asx.com.au
bourse-de-paris.fr
exchange.de
londonstockex.co.uk
nasdaq.com
nyse.com
ose.no
tse.or.jp
xsse.se
```

Note: squidGuard will match any URL with the domainname itself an any subdomains and hosts (i.e.`amex.com`, `www.amex.com`, `whatever.amex.com` and `www.what.ever.amex.com` but not `.*[^.]amex.com` (i.e.`aamex.com` etc.)).

URLlists

The urllist file format is simply URLs separated by newline but with the "`proto://((www|web|ftp)[0-9]*)?`" and "`(:port)?`" parts and normally also the ending "`(/|/[^/]+\.[^/]+)$`" part (i.e. ending "`/`" or "`/`*filename*") choped off.
(i.e. "~~http://www3.~~foo.bar.com~~:8080~~/what/ever~~/index.html~~" =>
"foo.bar.com/what/ever")

For instance a category for banned sites:

```
foo.com/~badguy
bar.com/whatever/suspect
```

Note: The removed parts above are ignored by squidGuard in URL matching.
Thus all these URLs will match the above url list:

```
http://foo.com/~badguy
http://foo.com/~badguy/whatever
ftp://foo.com/~badguy/whatever
wais://foo.com/~badguy/whatever
http://www2.foo.com/~badguy/whatever
http://web56.foo.com/~badguy/whatever
```

but not:

```
http://barfoo.com/~badguy
http://bar.foo.com/~badguy
```

```
http://foo.com/~goodguy
```

- **Expressionlists**

The expressionlist file format is lines with regular expressions as described in regex(5). Of most interest is:

`.`	Matches any single character (use "`\.`" to match a "`.`").
`[abc]`	Matches one of the characters ("`[abc]`" matches a single "`a`" or "`b`" or "`c`").
`[c-g]`	Matches one of the characters in the range ("`[c-g]`" matches a single "`c`" or "`d`" or "`e`" or "`f`" or "`g`". "`[a-z0-9]`" matches any single letter or digit. "`[-/.:?]`" matches any single "`-`" or "`/`" or "`.`" or "`:`" or "`?`".).
`?`	None or one of the preceding ("`words?`" will match "`word`" and "`words`". "`[abc]?`" matches a single "`a`" or "`b`" or "`c`" or nothing (i.e."")).
`*`	None or more of the preceding ("`words*`" will match "`word`", "`words`" and "`wordsssssss`". "`.*`" will match anything including nothing).
`+`	One or more of the preceding ("`xxx+`" will match a sequence of 3 or more "`x`").
`(expr1|expr2)`	One of the expressions, which in turn may contain a similar construction ("`(foo|bar)`" will match "`foo`" or "`bar`". "`(foo|bar)?` will match "`foo`" or "`bar`" or nothing (i.e."")).
`$`	The end of the line ("`(foo|bar)$`" will match "`foo`" or "`bar`"only at the end of a line).
`\x`	Disable the special meaning of *x* where *x* is one of the special regex characters "`.?*+()^$[]{}\`" ("`\.`" will match a single "`.`", "`\\`" a single "`\`" etc.)

Thus a start to block possible sexual material by expression match could look like:

```
(^|[-\?+=/_])(bondage|boobs?|busty?|hardcore|porno?|sex|xxx+)([-\?+=/_]|$)
```

Notes:

- Unless you build your expressions very very carefully there is a high risk you will have annoyed users on your neck. Typically you might accidentally block "Essex", "Sussex", "breastcancer", "www.x.org" etc. in your eagerness for blocking pornographic material. In practice you would probably replace some of the words in the example above with some more clearly pornographic related words that I don't find appropriate to list here.
- While the size of the domain and urllists only has marginal influence on the performance, too many large or complex expressions will quickly degrade the performance of squidGuard. Though it may depend heavily on the performance of the regex library you link with.
- There is a rich set of sample files for a group of supposedly pornographic sites under `samples/dest/adult` in the source tree that you can use as a start if porn blocking is one of your tasks. **Please note:** We recommend that you review these lists before using them. Those domains and urls have been collected *automagically* by a robot. No manual evaluation of the corresponding contents has been performed. Therefor there is a chance some nonpornographic sites have sliped in. Please report such errors but don't blame us if your fine site is on the list. (Blame those who have pointers to appropriate sites mixed in on their heavy porn link pages!)

2.8 - Security web sites:

```
http://www.linux-sec.net
```

2.9 - General Linux references links

```
http://www.icon.co.za/~psheer/book/rute.html.gz
```

2.10 Extra Tips and Tricks

2.10.1- Securely sending Mail from laptop to Internet via Wireless LAN

The saga of "how can I securely send mail from my laptop when it's plugged in to someone else's network" continues. If you run a wireless network, you probably block port 25 to keep people from using your network for spamming. The problem is how do you make an outgoing SMTP connection from your laptop when you travel to other people's wireless networks?
Miek Gieben has an elegant answer based on Postfix and OpenSSH. Set up Postfix on the laptop to use localhost and a high port as its relayhost, then tunnel that high port to port 25 on your mail server. Details and config file entries for the basic method are here: `http://www.miek.nl/linux/postfix.html.`

The problem: you have a laptop and you're not always connected to the Internet. Still you want to sent mail even when you're offline. You cannot use just any mail server out there, 'cause a lot of them don't relay. So you must use your own mail server.

You'll need:

`postfix`	only used for queuing and forwarding the mail
`openSSH`	for setting up a tunnel `OpenSSH config`

You will need to create a ssh tunnel to your mail server. This is the command I use:

```
ssh -2 -N -f -L 10025:elektron.atoom.net:25
miekg@elektron.atoom.net 2>/dev/null
```

Of course you also want to setup ssh so that you can login without typing a password.

<u>postfix config</u>
Next you must tell postfix to use you're tunnel. It is also important to keep postfix from doing MX lookups. In the main.cf of postfix add the following:

```
relayhost = [127.0.0.1]:10025     # use the tunnel, no MX lookups
defer_transports = smtp                     # only send when online
mydestination = localhost.localdomain    # not sure if this is needed
```

<u>system config</u>
Put the startup commands in `/etc/dhclient-exit-hooks`. This way every time you get a IP number from a dhcp-server the tunnel is re-established and the mail is flushed.

<u>`/etc/dhclient-exit-hooks:`</u>

```
----------------------
# start an ssh tunnel to elektron
ssh -2 -N -f -L 10025:elektron.atoom.net:25 \
     miekg@elektron.atoom.net 2>/dev/null
----------------------
```

```
# run the mailqueue
/usr/sbin/sendmail -q
```
Also make a cronjob that runs every now and then to flush the queue:
```
-----------------------
# run queue every 5 minutes
*/5 * * * *     root  test -x /usr/sbin/sendmail # && \
  nice -n10 /usr/sbin/sendmail -q
-----------------------
```
That's it. Mail should now be queued until it can be delivered via your own mail server.
```
-----------------------------------------------------------------
```
For security, though, you don't want to let people log in to your OpenSSH server as you without a passphrase. So, instead of using your main OpenSSH identity for the tunnel, add a user called tunnel to the OpenSSH server and make a tunnel-only SSH key on the laptop with this command:

```
ssh-keygen -t dsa -f tunnel
```

In the file `tunnel.pub`, add the following options to the beginning of the line:
```
command="/bin/false",no-X11-forwarding,no-agent-forwarding,no-pty
```
and then copy the contents of `tunnel.pub` to the end of the tunnel user's `~/.ssh/authorized_keys` file on the OpenSSH server.

If you can run only `/bin/false` and you can't forward X or the SSH agent, what's left? Port forwarding. Add `-i` tunnel to Miek's instructions to use the tunnel, and replace `miekg@` with `tunnel@`.

Class Firewall (for a Desktop Station)

```
#!/bin/bash
# Zweck:    Desktop PC Firewall
# Datum:    11.02.2005
# Syntax:   firewall {start|stop|restart|status}
# Version:  0.01
# Veraenderungen:
#--------------------------------------------------------------------------------
#
# Die folgenden 2 Variablen sind aus /etc/rc.status genommen worden
if [ $TERM = "linux" -o $TERM = "xterm" ]
then
  rc_done="\015\033[80C\033[10D\033[1;32mdone\033[m\017"
  rc_failed="\015\033[80C\033[10D\033[1;31mfailed\033[m\017"
else
  rc_done="done"
  rc_failed="failed"
fi

# Definition der Variablen
#-------------------------------------------------------------------------------
netzkarte="eth0"
mein_host=192.168.71.130
proxy=192.168.71.9
proxy_p=3128
dnsserver=192.168.71.40
druckserver=192.168.71.166
lan1=192.168.71.0/24
lan_broadcast=192.168.71.255
full_broadcast=255.255.255.255
alles=0/0
nfsserver="192.168.71.241"

case $1 in
   start)
      echo "$0 wird gestartet... "
      #---------------- SPEZIELLE FUER NFS CLIENT (nur ein Client ----------
      # Wegen dass mountd Daemon steth auf ein dynamische port
      iptables -P INPUT   ACCEPT
      iptables -P OUTPUT  ACCEPT
      iptables -P FORWARD ACCEPT
       #------------------------ Mountd daemon dynamische Addresse hollen
      mountd_udp_port=$(rpcinfo -p $nfsserver | grep "100005.*udp.*mountd" \
                | sed 's/ \+/ /g' | head -n1 | cut -d" " -f6)
      mountd_tcp_port=$(rpcinfo -p $nfsserver | grep "100005.*tcp.*mountd" \
                | head -n1 | awk '{ print $4}'

      # Definition der Grundpolitik. Grundsaetzlich wird alles verboten.
      iptables -P INPUT   DROP
      iptables -P OUTPUT  DROP
      iptables -P FORWARD DROP

      # Alle Regeln werden geleert (geloescht)
      iptables -F
      iptables -t nat -F

      # -------------LOOPBACK ----------------------------------------
      # Erlaubt alle Pakete vom Loopback Device
      iptables -A OUTPUT -o lo -j ACCEPT
```

```
# Erlaubt alle Pakete zum Loopback Device
iptables -A INPUT -i lo -j ACCEPT


#--------- SSH CLIENT (STATIC Regeln)--------------------------
# Erlaubt die Verbindung zu fremde SSH-Servern.
iptables -A OUTPUT -o $netzkarte -p tcp -s $mein_host --sport 1024: \
                                 -d $lan1 --dport ssh -j ACCEPT

# Erlaubt die Verbindung(antworten) vom fremde SSH-Servern.
iptables -A INPUT -i $netzkarte -p tcp -s $lan1 --sport ssh  \
                                 -d $mein_host --dport 1024: -j ACCEPT

#-- STATEFUL INSPECTION (FOR ALL ESPABLISHED CONNECTIONS)------
# Erlaubt all Pakete von vorhandene Verbindungen
iptables -A OUTPUT -o $netzkarte -m state \
               --state ESTABLISHED,RELATED -j ACCEPT

# Erlaubt all Pakete von vorhandene Verbindungen
iptables -A INPUT -i $netzkarte -m state \
                                    --state ESTABLISHED,RELATED -j ACCEPT
#--------- SSH CLIENT ---------------------------------------
# Erlaubt die Verbindung zu fremde SSH-Servern.
iptables -A OUTPUT -o $netzkarte -p tcp -s $mein_host --sport 1024: \
                                 -d $lan1 --dport ssh  \
                                 -m state --state NEW -j ACCEPT

#--------- SSH SERVER --------------------------------------------
# Erlaubt Verbindungen auf meinem SSH-Server
iptables -A INPUT -i $netzkarte -p tcp -s $lan1 --sport 1024: \
                                -d $mein_host --dport ssh \
                                -m state --state NEW -j ACCEPT

#--------- DNS CLIENT --------------------------------------------
# Erlaubt die Verbindung zum DNS-Server
iptables -A OUTPUT -o $netzkarte -p udp -s $mein_host --sport 1024: \
                                 -d $dnsserver --dport domain \
                                 -m state --state NEW -j ACCEPT

iptables -A OUTPUT -o $netzkarte -p tcp -s $mein_host --sport 1024: \
                                 -d $dnsserver --dport domain \
                                 -m state --state NEW -j ACCEPT

#--------- CUPS DRUCKER CLIENT ------------------------------
# Erlaubt ueber einen LPD-Drucker-Server zu drucken (Port 515).
iptables -A OUTPUT -o $netzkarte -p tcp -s $mein_host --sport :1024 \
                                 -d $druckserver --dport ipp  \
                                 -m state --state NEW -j ACCEPT

#--------- CUPS DRUCKER SERVER ------------------------------
# Erlaubt Verbindungen auf meinem CUPS-Server
iptables -A INPUT -i $netzkarte -p tcp -s $lan1 --sport 1024: \
                                -d $mein_host --dport ipp \
                                -m state --state NEW -j ACCEPT

#--------- PROXY CLIENT(if needed)-----------------------------
# Erlaubt die Verbindung zum Proxy. Dynamisch (Stateful Inspection)
iptables -A OUTPUT -o $netzkarte -p tcp -s $mein_host --sport 1024: \
                                 -d $proxy --dport $proxy_p  \
                                 -m state --state NEW -j ACCEPT
#--------- HTTP/HTTPS CLIENT----------------------------------
```

```
 # Erlaubt die Verbindung zum HTTP. Dynamisch (Stateful Inspection)
 iptables -A OUTPUT -o $netzkarte -p tcp -s $mein_host --sport 1024: \
                                  -d $alles --dport http  \
                                  -m state --state NEW -j ACCEPT

 # Erlaubt die Verbindung zum HTTPS. Dynamisch (Stateful Inspection)
 iptables -A OUTPUT -o $netzkarte -p tcp -s $mein_host --sport 1024: \
                                  -d $alles --dport https  \
                                  -m state --state NEW -j ACCEPT

 #--------- FTP CLIENT -------------------------------------------
 # FTP - Control-Port
 iptables -A OUTPUT -o $netzkarte -p tcp -s $mein_host --sport 1024: \
                                  -d $alles --dport ftp \
                                  -m state --state NEW -j ACCEPT
 # Daten-Port
 iptables -A OUTPUT -o $netzkarte -p tcp -s $mein_host --sport 1024: \
                                  -d $alles --dport 1024: \
                                  -m state --state NEW -j ACCEPT

# Ident (eigentlich einen unoetigen Dienst,
# beschleunigt aber die Antwort vom FTP-Server
iptables -A OUTPUT -o $netzkarte -p tcp -s $mein_host --sport 1024: \
                                  -d $alles --dport ident  \
                                  -m state --state NEW -j ACCEPT

 #------------ FTP SERVER ---------------------------------------
 # Erlaubt Verbindungen auf meinem FTP-Server
 iptables -A INPUT -i $netzkarte -p tcp -s $lan1 --sport 1024: \
                                 -d $mein_host --dport ftp \
                                 -m state --state NEW  -j ACCEPT

 iptables -A INPUT -i $netzkarte -p tcp -s $lan1 --sport 1024: \
                                 -d $mein_host --dport 1024: \
                                 -m state --state NEW -j ACCEPT

 #-------------- ICMP ----------------------------------------------
 # Erlaubt ein paar ICMP-Typen
 iptables -A OUTPUT -o $netzkarte -p icmp \
                        --icmp-type echo-reply              -j ACCEPT
 iptables -A OUTPUT -o $netzkarte -p icmp \
                        --icmp-type echo-request            -j ACCEPT
 iptables -A OUTPUT -o $netzkarte -p icmp \
                        --icmp-type destination-unreachable -j ACCEPT
 iptables -A OUTPUT -o $netzkarte -p icmp \
                        --icmp-type source-quench           -j ACCEPT
 iptables -A OUTPUT -o $netzkarte -p icmp \
                        --icmp-type time-exceeded           -j ACCEPT
 iptables -A OUTPUT -o $netzkarte -p icmp \
                        --icmp-type parameter-problem       -j ACCEPT

 # Erlaubt ein paar ICMP-Typen
 iptables -A INPUT -i $netzkarte -p icmp \
                        --icmp-type echo-reply              -j ACCEPT
 iptables -A INPUT -i $netzkarte -p icmp \
                        --icmp-type echo-request \
                        -m limit --limit 5/minute           -j ACCEPT
 iptables -A INPUT -i $netzkarte -p icmp \
                        --icmp-type destination-unreachable -j ACCEPT
 iptables -A INPUT -i $netzkarte -p icmp \
                        --icmp-type source-quench           -j ACCEPT
 iptables -A INPUT -i $netzkarte -p icmp \
                        --icmp-type time-exceeded           -j ACCEPT
```

```
  iptables -A INPUT -i $netzkarte -p icmp \
                        --icmp-type parameter-problem       -j ACCEPT
```

Übung: Verbindung erlauben (für Teilnegmer zu entwickeln)

```
  #---------- WINDOWS SHARES (CLIENT)(Exercise Together)------
  #---------- WINDOWS SHARES (SERVER)(Exercise Together)------
  #---------- HTTP SERVER (Exercise)----------------------------
  # Erlaubt Verbindungen auf meinem FTP-Server
  iptables -A INPUT -i $netzkarte -p tcp -s $lan1 --sport 1024: \
                                 -d $mein_host --dport http \
                                 -m state --state NEW  -j ACCEPT
  #---------- SMTP CLIENT (Exercise)----------------------------
  #---------- POP CLIENT (Exercise)-----------------------------

 # ############# NFS: Portmapper, Mountd und NFS-Server#######
 # Portmapper
 iptables -A OUTPUT -o $netzkarte -p tcp -s $mein_host --sport :1024      \
                                  -d $lan1 --dport 111                    \
                                  -m state --state NEW -j ACCEPT
 iptables -A OUTPUT -o $netzkarte -p udp -s $mein_host --sport :1024      \
                                  -d $lan1 --dport 111                    \
                                  -m state --state NEW -j ACCEPT
 # mountd-Daemon (Dynamische port) sehe anfang von 'start' case.
 iptables -A OUTPUT -o $netzkarte -p tcp -s $mein_host --sport :1024      \
                                  -d $lan1 --dport $mountd_tcp_port       \
                                  -m state --state NEW -j ACCEPT
 iptables -A OUTPUT -o $netzkarte -p udp -s $mein_host --sport :1024      \
                                  -d $lan1 --dport $mountd_udp_port       \
                                  -m state --state NEW -j ACCEPT
 # erlaubt einen "showmount -e Rechner"
 iptables -A OUTPUT -o $netzkarte -p tcp -s $mein_host --sport :1024      \
                                  -d $lan1 --dport 1024:                  \
                                  -m state --state NEW -j ACCEPT
 # NFS-Server
 iptables -A OUTPUT -o $netzkarte -p udp -s $mein_host --sport :1024      \
                                  -d $lan1 --dport 2049                   \
                                  -m state --state NEW -j ACCEPT
  ################### ENDE von NFS Übung #############################

 #---------------- DROP JUNK BEFORE LOGGING ------------------------
 # Verhindert, dass die Protokolldatei mit Boradcasts gefuellt wird
 iptables -A INPUT -i $netzkarte -p udp -s $lan1 -d $full_broadcast -j DROP

 # Verwirft alle Pakete von ungueltigen neuen Verbindungen(SYN-->DROP)
 iptables -A INPUT -i $netzkarte -m state --state NEW,INVALID -j DROP
 #---------------- LOGGING -----------------------------------------
 # Protokolliert alle restliche Pakete (syslog: kernel.warn)
 iptables -A OUTPUT -j LOG --log-prefix "End-OUTPUT TABELLE - "

 # Protokolliert alle restliche Pakete (syslog: kernel.warn)
 iptables -A INPUT -j LOG --log-prefix "End-INPUT TABELLE - "

 echo -e "$rc_done"
 ;;
stop)
 echo "$0 wird gestoppt. "
 iptables -P INPUT   ACCEPT
 iptables -P OUTPUT  ACCEPT
 iptables -P FORWARD ACCEPT
 iptables -F
 iptables -t nat -F
```

```
        echo -e "$rc_done"
        ;;
```

```
  restart)
     $0 stop && $0 start || echo -e "rc_failed"

     ;;
   status)
     iptables -nvL
     ;;
   *)
      echo -n "Benutzung: $0 {start|stop|restart|status}"
      echo -e "$rc_failed"
      exit 1
esac
exit 0
```