```
void
print_msg(const char *msg)
{
    printf("There is a problem here:\n");
    printf("%s", msg);
}
```

↳ CORRECT way

The lisp syndrom

- Every "call" tends to evolve to do more than its own good
- Beware of abstraction layer violations

"Everything is awesome!" → "Everything is broken!"

Hidden shells

- For instance system(3) and popen(3)
- That's what of the Qualys CVEs
- Both basically run sh -c "string"
- ... so to properly use them you have to quote everything
- What's everything
- #{}()"'`~\$

The best solution tho is to NOT use those function is better off using fork(), pipe() and shit like that

- system(3) is more or less:

```
int R= fork();
if (R==-1)
  err(1, "fork");
if (R==0){
  execl("mycmd", "cmd", "param", ..., NULL);
  err(1, "exec");
}
int status;
R= wait(&status); //XXX
//check status
```

Problem: - won't wait until the program is finished or wait forever
- will fuck up other parts of the code

- popen(3) is more or less:

```
int fd[2];
int R= pipe(fd);
if (R==-1)
  err(1, "pipe");
int R= fork();
if (R==-1)
  err(1, "fork");
if (R==0){
  close(fd[0]);
  dup2(fd[1], 1); //XXX
  close(fd[1]);
  execl("mycmd", "cmd", "param", NULL);
  err(1, "exec");
}
close(fd[1]);
R= wait(&status);
```

```
#! /bin/sh
file=$1
rm $file
```

./s "my file"

```
#! /bin/sh
rm "$@"
```

./s ../myfile
./s -rf/myfile

- quoting is not enough
- use cmd -- args to stop option parsing
- GNU fucked it up...
- if you write your own commands don't allow REORDER!

Fail closed

- Always use | set -e

```
#!/bin/sh
set -e
ERROR=false
if ! [ -f Makefile ] && test -f distinfo && test -d pkg
then
    echo "No ports files ?"
    ERROR=true
fi
fulldir=$(pwd)
importname=$(echo $fulldir | sed -e s,.*/ports/,ports/,)
$ERROR && exit 1
cvs import $importname espie ports
cd ..
rm -Rf $fulldir
```

The Unix security model

You check you can access a file at open/exec

How does it work

- identify who you are: uid/gid
- don't forget supplementary groups
- only check the first entry that applies
- if uid = file owner, check user bits
- else if one group matches file group, check group bits
- else match other bits

Beware of extensions

- Sometimes you've got Mandatory Access control extensions that make this complicated
- The main problem is testing all combinations
- See windows and Active Directory

I am root

- We ignore Rights!
  - ... so first open the file
  - then check (fstat) you could do it

What Rights do I have

- I have the Rights of the process
- ... plus every valid fd I own

Priv Drop

- Start life as root
- do privileged operations yielding fds
- ... then change identity
- I still have the fds!

(application: network server on a privileged port)

Too much

- There is a closefrom(3) on BSD/Solaris. Not on linux though
- There is O_CLOEXE

Dropping privileges, the fine print

- Set supplementary groups using setgroups
- Set your group id using setgid
- Set your user id using setuid
- You can check your code by invoking system("id"))

Beware of Linux: Make sure you verify setuid/setgid did work (capabilities).
(this broke sendmail btw)

What about setuid?

- a program with setuid is run "as the user to whom it belongs"
- you have three concepts of ids
  - effective id
  - real id
  - saved id

Privilege separation

- each process has its own memory space (but see mmap)
- each identity has its own Rights
- ... a bug in a process only affects what it can do

Designing software

- the more complex the code, the less Rights it should have
- sanitize input once thoroughly
- ... then you don't