

Building a Secure Server and Hardening Existing Systems

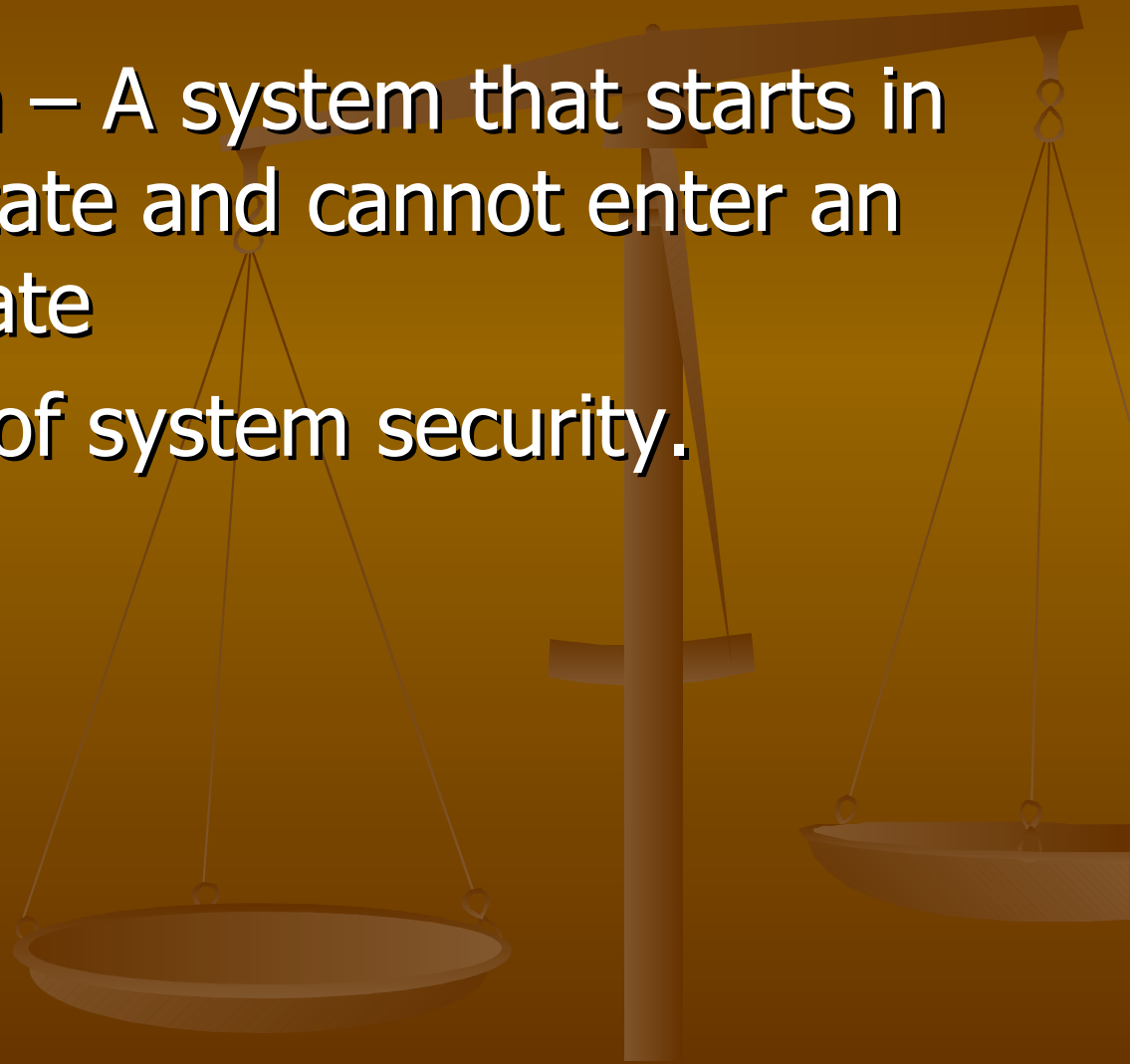


Adir Abraham

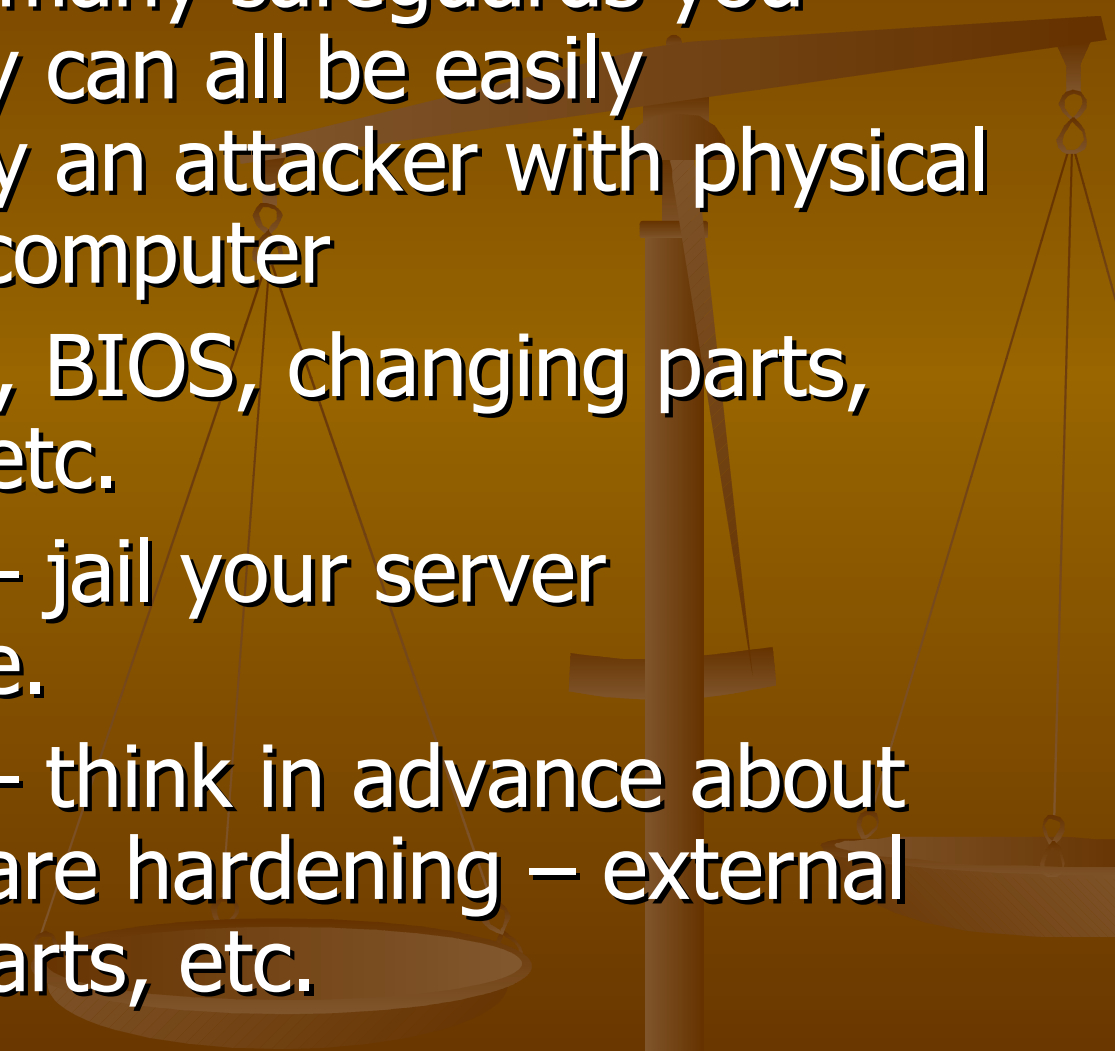
`adir@vipe.technion.ac.il`

System Security

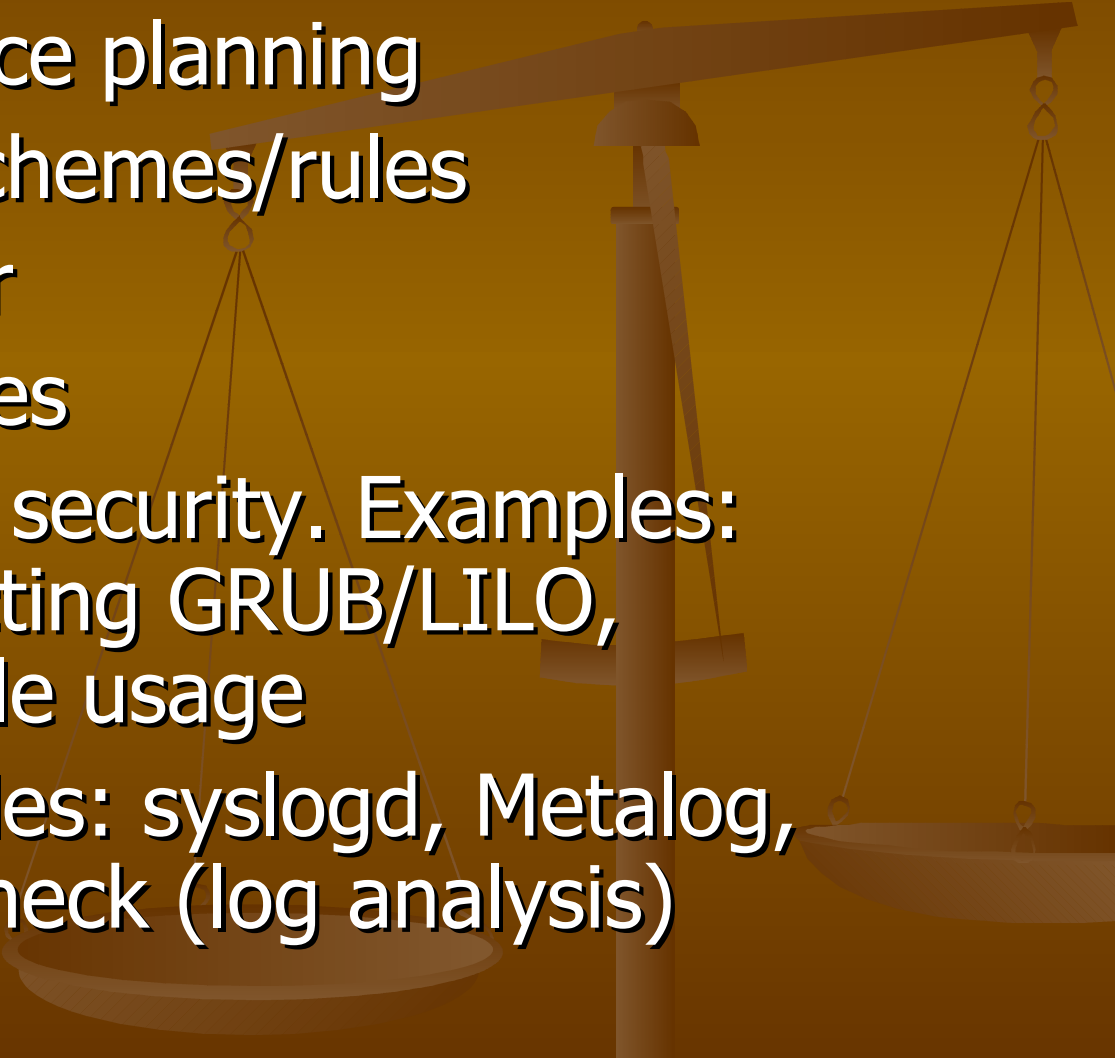
- A secure system – A system that starts in an authorized state and cannot enter an unauthorized state
- This is the goal of system security.



Physical Security

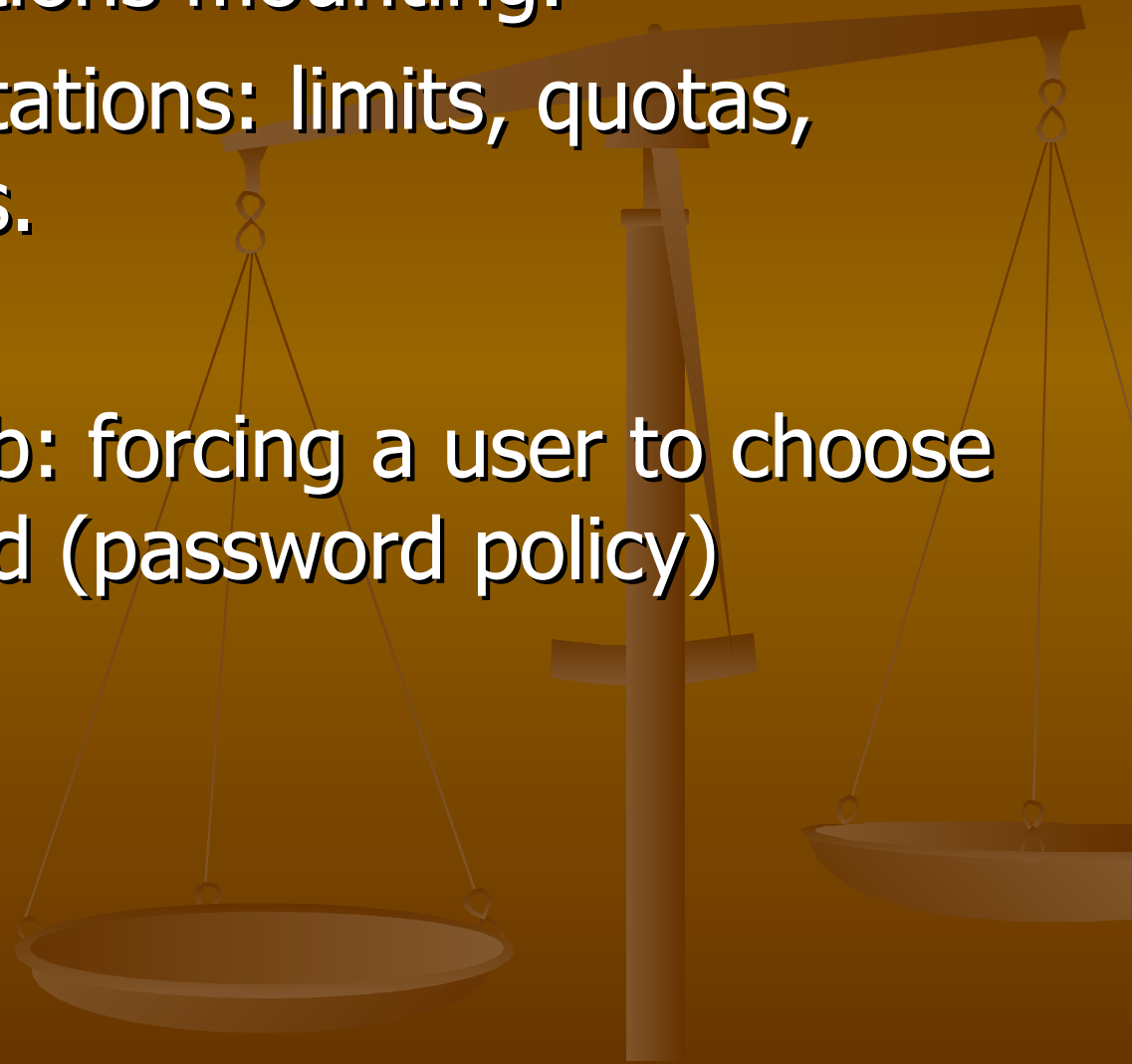
- No matter how many safeguards you implement, they can all be easily circumvented by an attacker with physical access to your computer
 - Examples: Boot, BIOS, changing parts, taking the HD, etc.
 - Conclusion #1 – jail your server somewhere safe.
 - Conclusion #2 – think in advance about levels of hardware hardening – external case, internal parts, etc.
- 

Let's build a secure server

- Plan! Pre-installation concerns:
 - 1)Daemon/Service planning
 - 2)Partitioning Schemes/rules
 - 3)Rooting a user
 - 4)Security policies
 - Tighten Console security. Examples: password protecting GRUB/LILO, restrictive console usage
 - Logging. Examples: syslogd, Metalog, Syslog-ng, Logcheck (log analysis)
- 

Let's build a secure server

- Restrictive partitions mounting.
- User/Group limitations: limits, quotas, login restrictions.
- File permissions
- PAM and cracklib: forcing a user to choose a good password (password policy)
- TCP Wrappers
- Kernel Security

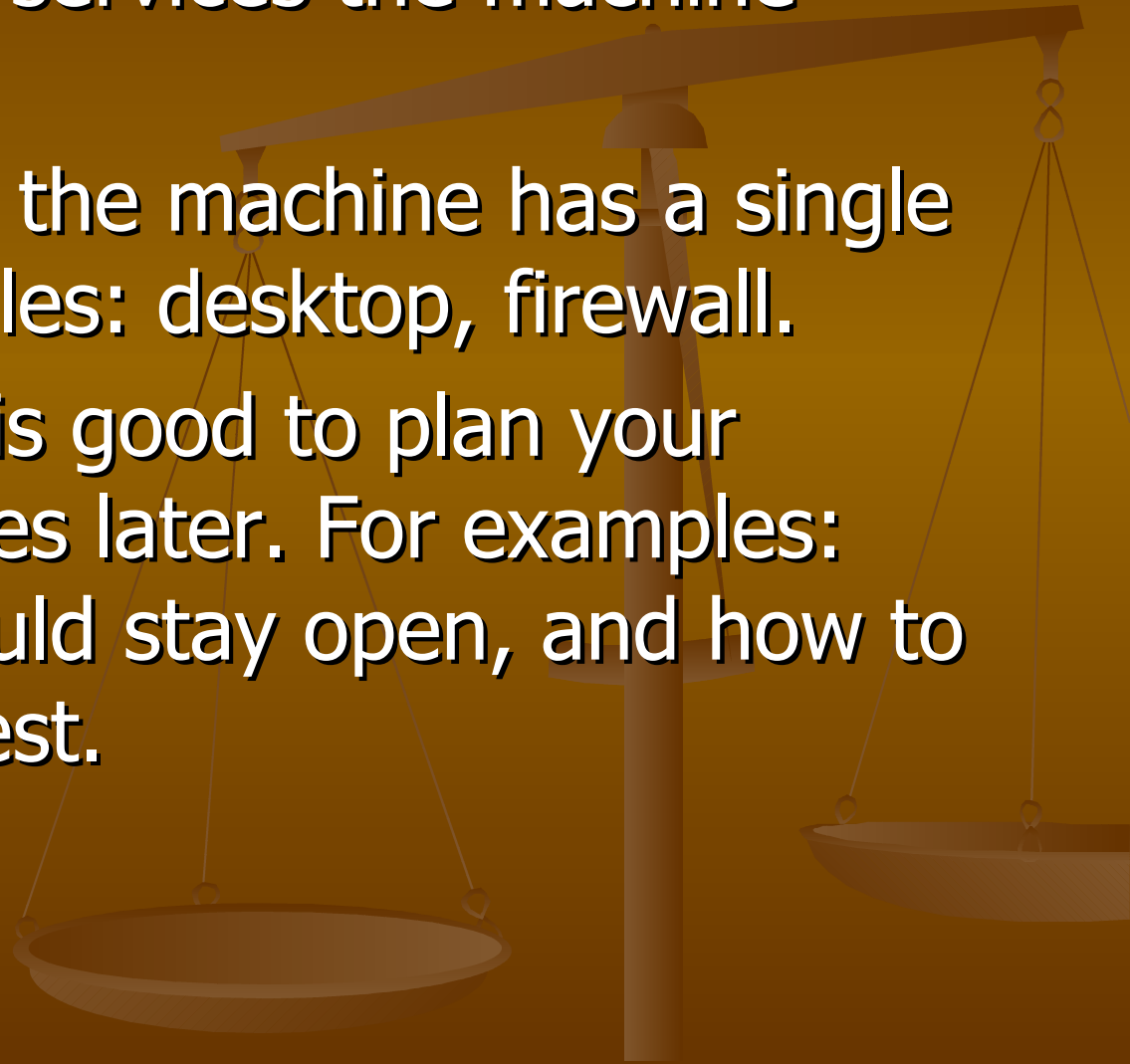


Pre-installation concerns



Daemon/Service Planning

- Document what services the machine should run.
- Not necessary if the machine has a single purpose. Examples: desktop, firewall.
- Documentation is good to plan your security measures later. For examples: which ports should stay open, and how to treat each request.

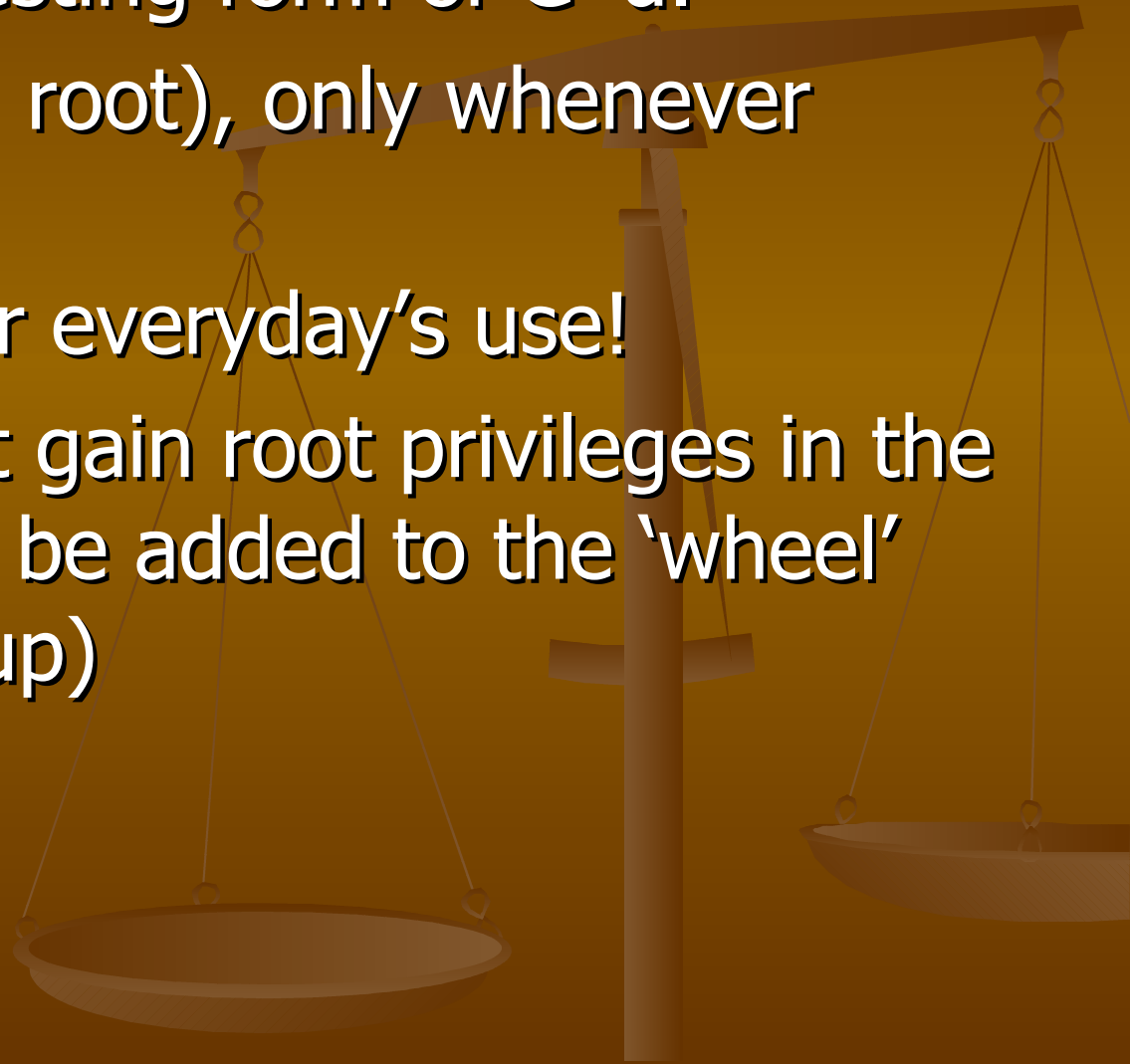


Partitioning Schemes

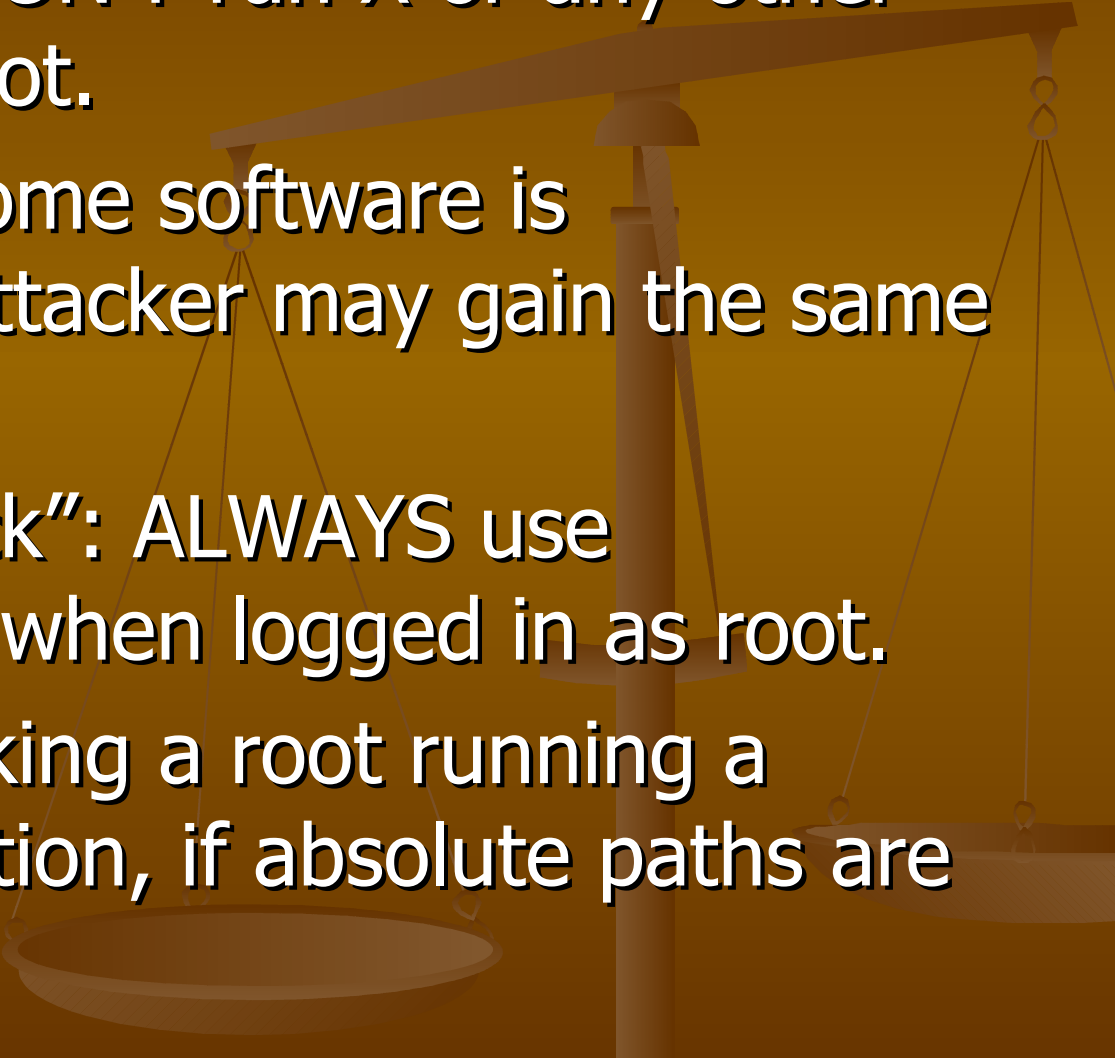
- Define rules for partitioning.
- Common rule #1: Any directory tree a user should be able to write to, should be on a separate partition and use disk quotas.
- Common rule #1 reduces the risk of a user filling up your whole file system.
- Common rule #2: Non-distribution software should be on a separate partition. Usually /opt or /usr/local.
- Common rule #2 promises that those non-distribution packages won't be erased if you reinstall your system

Rooting a user

- Root is an interesting form of G*d.
- Use G*d (ahem, root), only whenever necessary!
- Create a user for everyday's use!
- If this user must gain root privileges in the future, it should be added to the 'wheel' group (/etc/group)

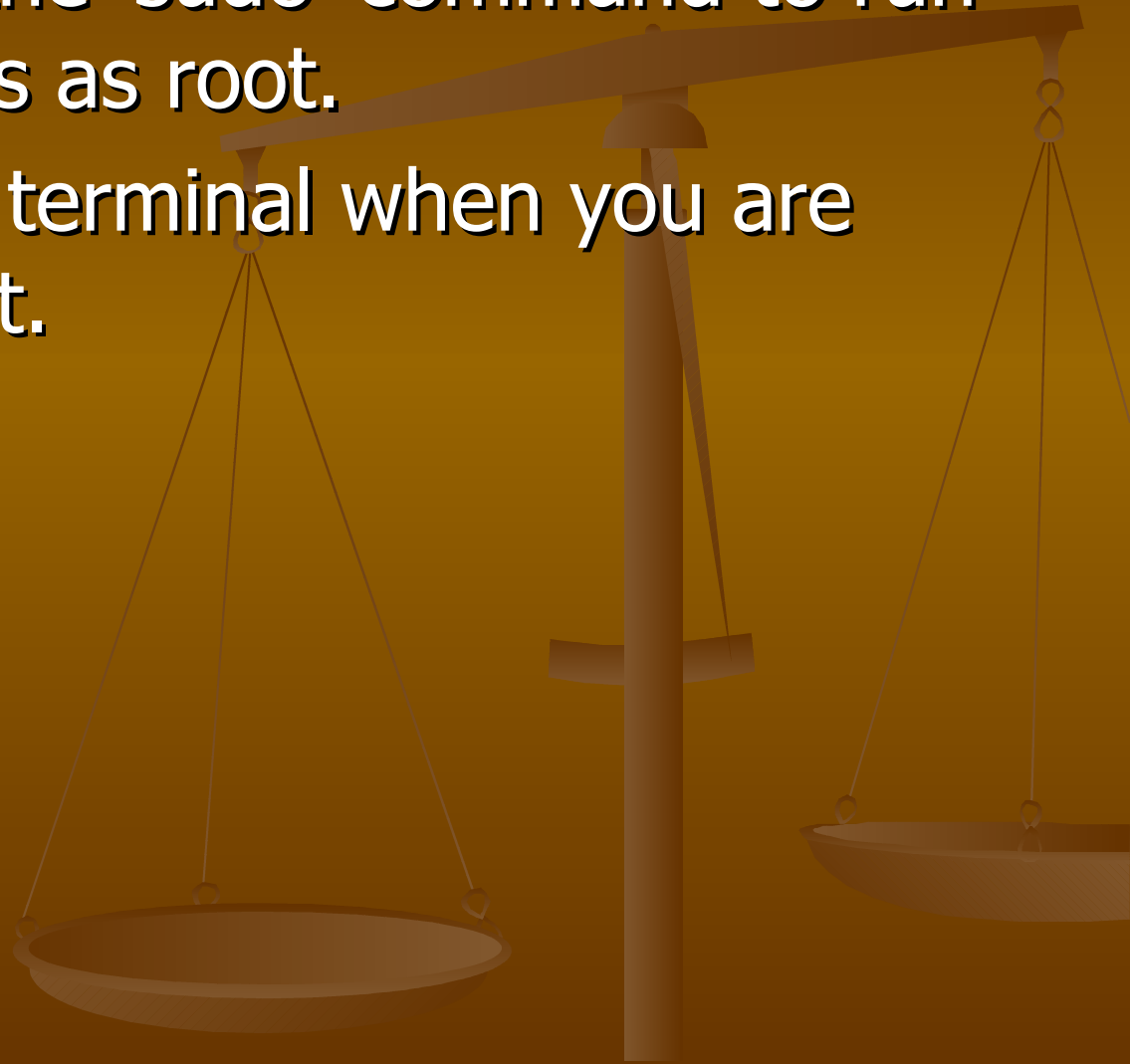


Rooting a user

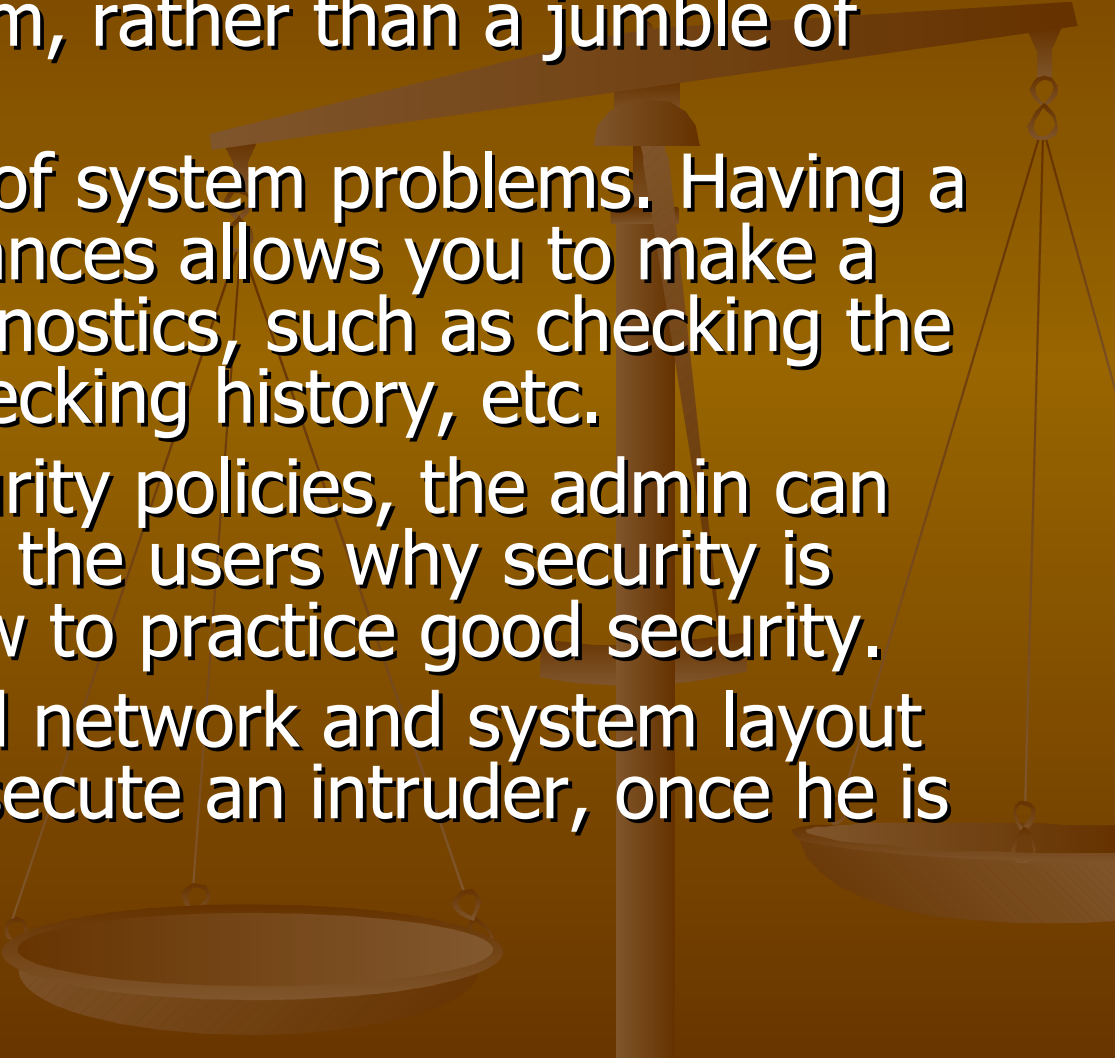
- Tempting, but DON'T run X or any other application as root.
 - The reason: if some software is vulnerable, an attacker may gain the same user's privileges.
 - The "PATH attack": ALWAYS use absolute paths when logged in as root.
 - The reason: tricking a root running a different application, if absolute paths are not used.
- 

Rooting a user

- Consider using the 'sudo' command to run a few commands as root.
- Never leave the terminal when you are logged in as root.



Security policies – Why?

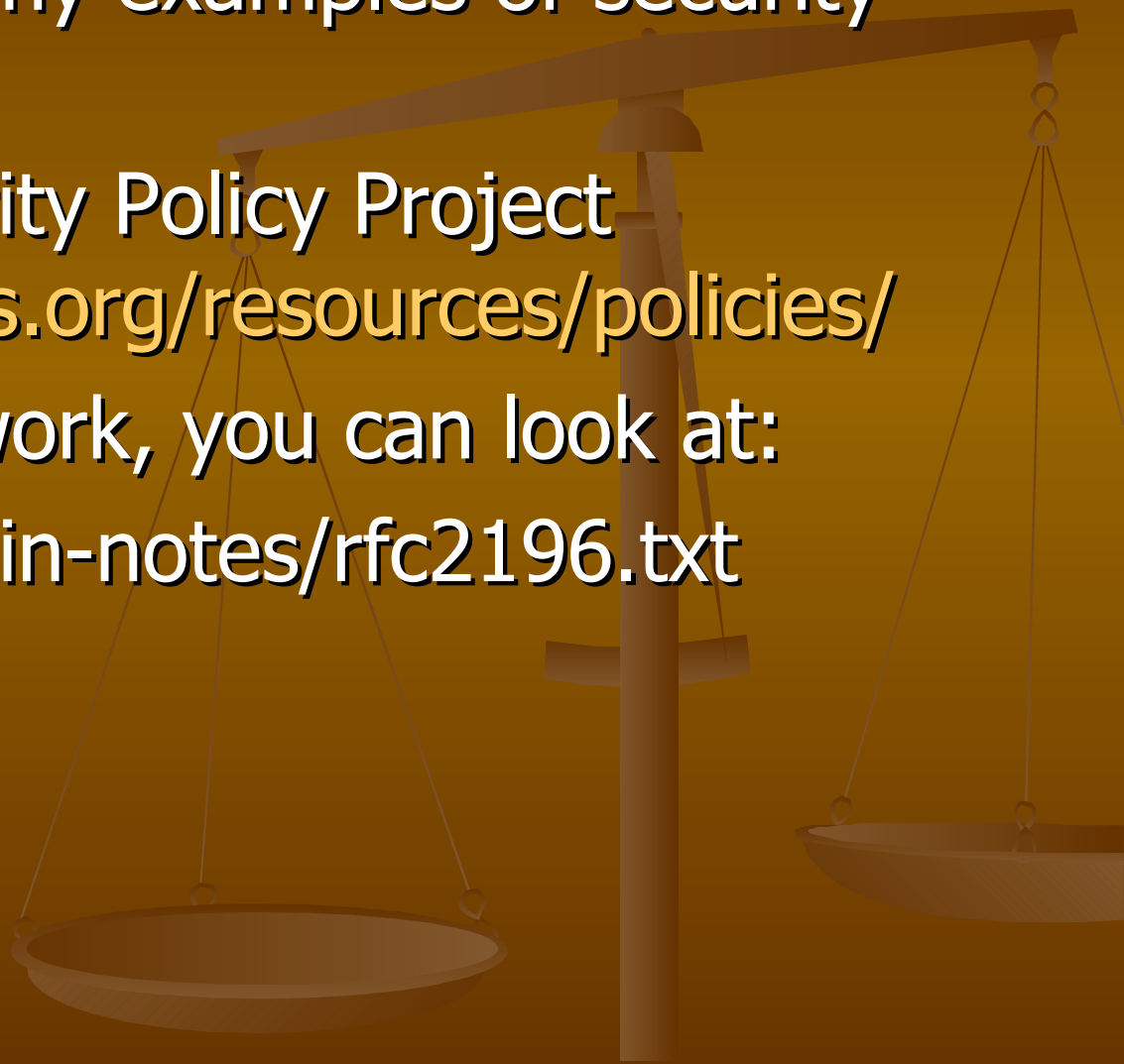
- A good security policy allows you to outline security as a system, rather than a jumble of features.
 - Better diagnostics of system problems. Having a set of rules in advances allows you to make a more efficient diagnostics, such as checking the network traffic, checking history, etc.
 - Once defining security policies, the admin can (should) explain to the users why security is important, and how to practice good security.
 - A well-documented network and system layout will aid you to prosecute an intruder, once he is caught.
- 

Security policy - concerns

- Define an acceptable set of system usage.
- A common example: screen savers, password handling, software download, use of anti-virus.
- Define how to treat sensitive information.
- A common example: clean desk and locked up classified information, PC shutdown before leaving, use of encryption (ssh, etc)
- Define your own set of rules for a laptop.

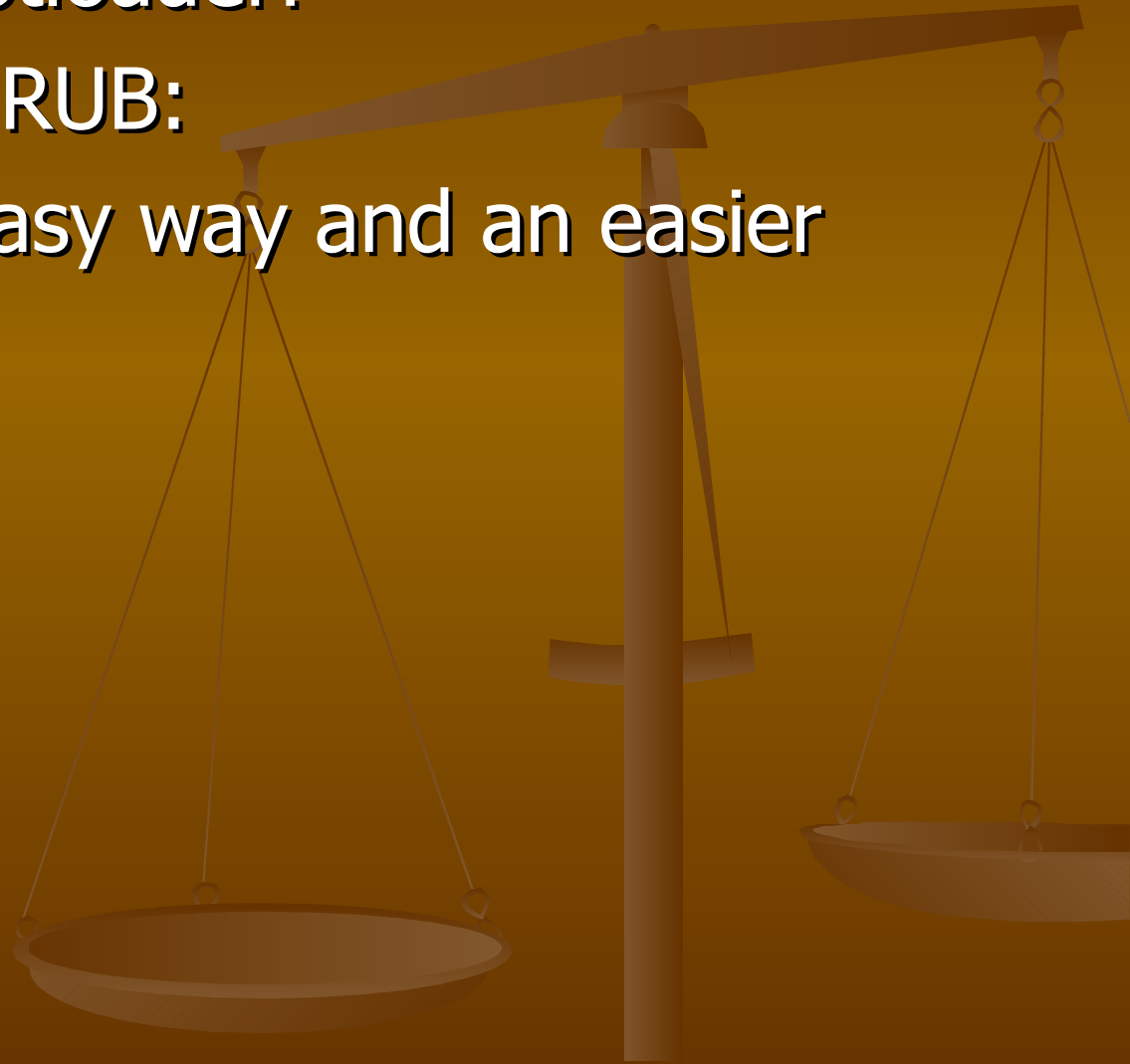
Security policies – more examples

- You can find many examples of security policies in:
- The SANS Security Policy Project
<http://www.sans.org/resources/policies/>
- For a small network, you can look at:
<ftp://ftp.isi.edu/in-notes/rfc2196.txt>



Tightening of Console Security

- Protect your bootloader!
- Example #1 – GRUB:
- Two ways. An easy way and an easier way.



Tightening of Console Security

- The easier way:

- 1) edit your `/boot/grub/grub.conf` file:

```
timeout 5
```

```
password changeme
```

- 2) This will add the password changeme. If no password is entered at boot, GRUB will simply use the **default** boot setting.

Tightening of Console Security

■ The easy way – MD5+SALT:

1) Extract /sbin/grub:

```
#/sbin/grub
GRUB version 0.92 (640K lower / 3072K upper memory)
[ Minimal BASH-like line editing is supported. For the first word,
  TAB lists possible command completions. Anywhere else TAB lists
  the possible completions of a device/filename. ]
grub> md5crypt
Password: *****
(Typed changeme at the prompt)
Encrypted: $1$T7/dgdIJ$dJM.n2wZ8RG.oEiIOwJUs.
grub> quit
```

2) Cut and paste your password to /boot/grub/grub.conf:

```
timeout 5
password --md5 $1$T7/dgdIJ$dJM.n2wZ8RG.oEiIOwJUs.
```

Tightening Console Security

- Example #2 – LILO:
- Lilo supports two ways of handling passwords: global and per-image, but in clear text only. You should edit `/etc/lilo.conf`
- An example of global password:

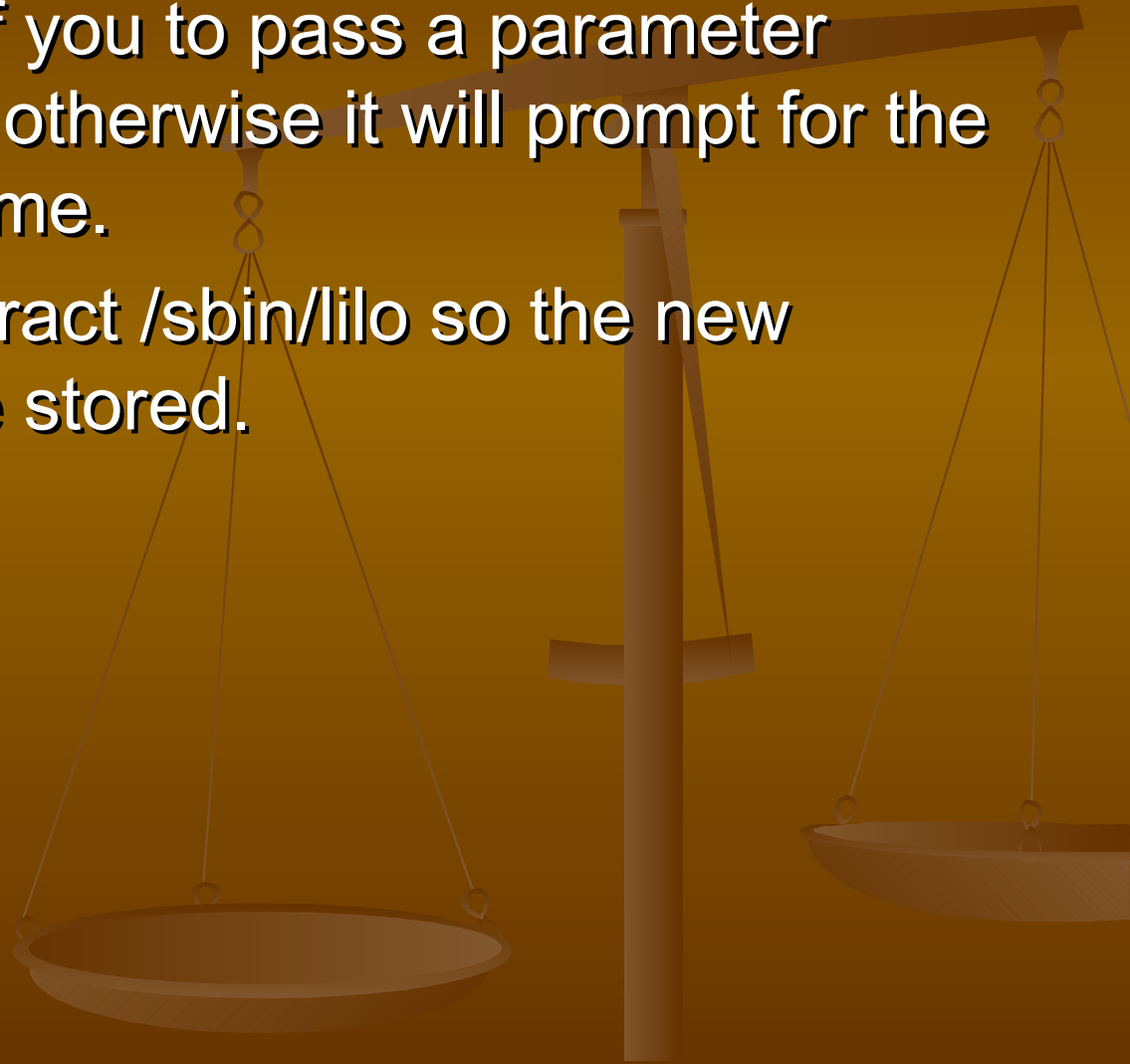
```
password=changeme  
restricted  
delay=3
```

- An example of per-image password:

```
image=/boot/bzImage  
read-only  
password=changeme  
restricted
```

Tightening Console Security

- The 'restricted' option will make LILO prompt for a password only if you to pass a parameter (such as 'single'), otherwise it will prompt for the password every time.
- Don't forget to extract /sbin/lilo so the new information will be stored.

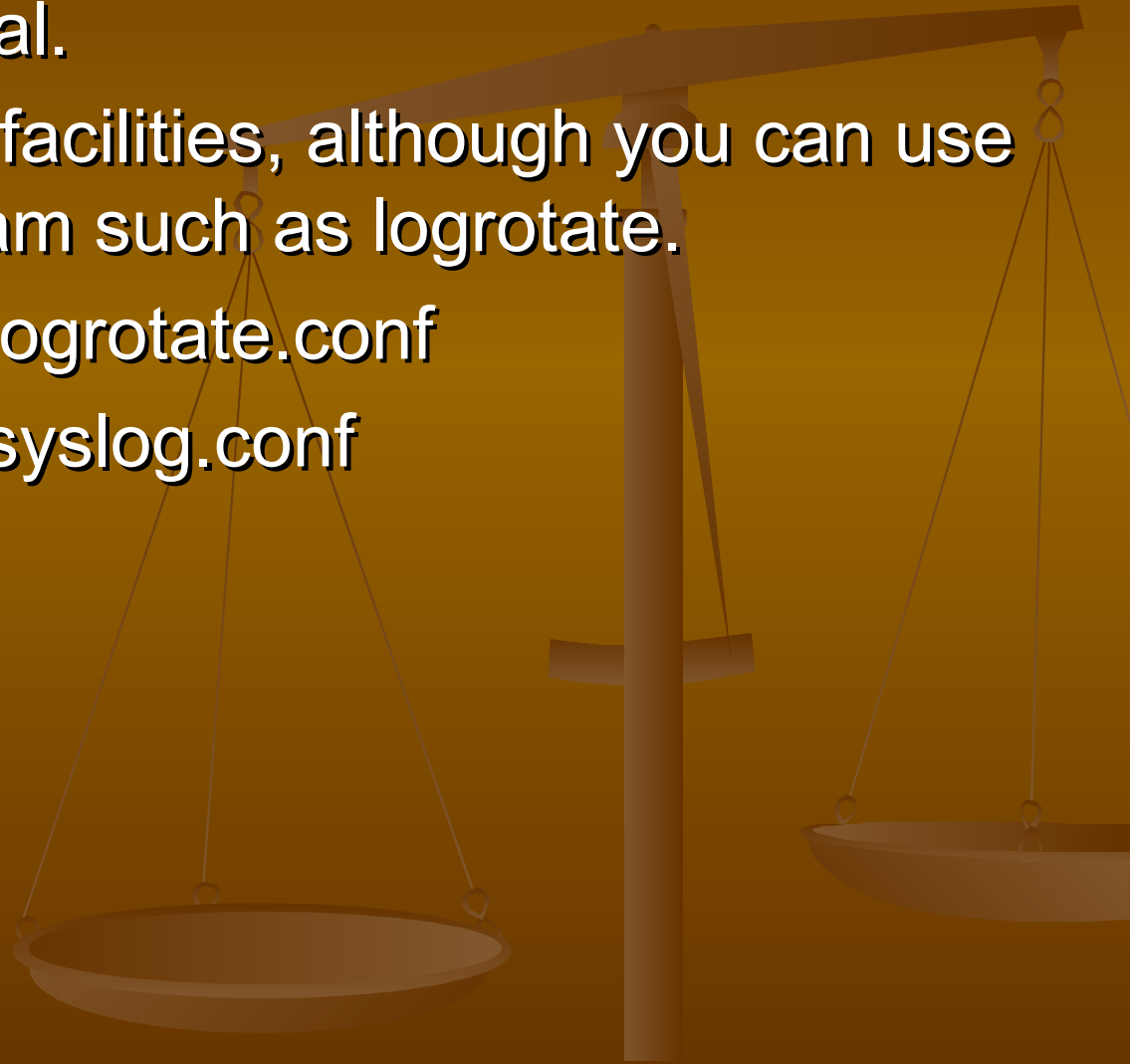


Logging

- Extra logging should be added to catch warnings or errors that might indicate an ongoing attack or a successful compromise.
- Attackers often scan or probe before attacking.
- Protect your log files. Don't let anybody who is not authorized, to touch them. They should be readable only.
- Log files should be easily readable and manageable (ofcourse, by the one who is authorized to do so).

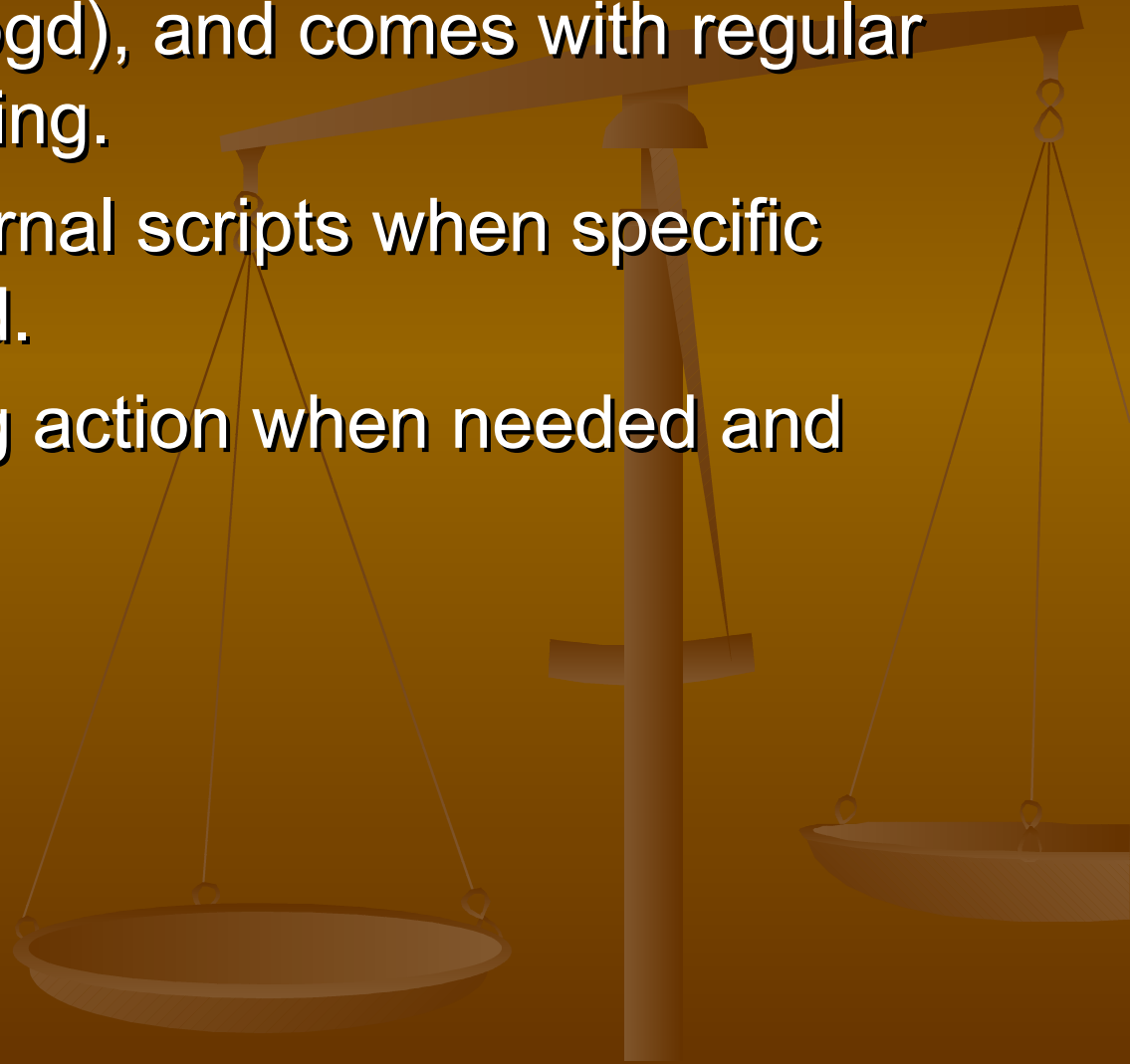
Logging: Syslogd

- Syslogd is the most common logger for Linux and Unix in general.
- It has log rotation facilities, although you can use an external program such as logrotate.
- Example #1: `/etc/logrotate.conf`
- Example #2: `/etc/syslog.conf`



Logging: Metalog

- Metalog logs by program name, urgency, rotation (like syslogd), and comes with regular expression matching.
- It can launch external scripts when specific patterns are found.
- It is good at taking action when needed and defined.



Logging: Metalog

For example:

`/usr/local/sbin/mail_pwd_failures.sh` for postfix

```
#!/bin/sh echo "$3" | mail -s "Warning (program : $2)" root
```

`/usr/local/sbin/mail_pwd_failures.sh` for qmail

```
#!/bin/sh  
echo "To: root  
Subject:Failure (Warning: $2)  
$3  
" | /var/qmail/bin/qmail-inject -f root
```

Remember to make the script executable by issuing
`/bin/chmod +x /usr/local/sbin/mail_pwd_failures.sh`

Then uncomment the command line under "Password failures" in `/etc/metalog/metalog.conf` like:

```
command = "/usr/local/sbin/mail_pwd_failures.sh"
```

Logging: Syslog-ng

- Syslog-ng is a combination of syslog and metalog.
- Features: filtering messages based on level and content (metalog), remoting logging (syslog), handle logs from syslogd, write to TTY, execute programs, and can also act as a logging server.
- The best of both loggers is combined in Syslog-ng, with advanced configuration.
- A disadvantage: the configuration file is huge, so it's easy to miss something in the file.

Logging analysis: Logcheck

- Catching logs is only half of the battle.
- Someone has to do the hard work of analyzing the logs.
- Catch a human, or... use logcheck (a script) and logtail (a binary), found in <http://logcheck.org/>
- Logcheck uses four files to filter important log entries from the unimportant.
- logcheck.hacking for known hacking attack messages, logcheck.violations to indicate security violations, logcheck.violations.ignore contains keywords likely to be matched by a violation file, and logcheck.ignore which matches those entries to be ignored.

Mounting partitions

- “Regular” mounting doesn’t include any security option.
- When mounting an ext2, ext3 or reiserfs partition, you have several options you can apply to the file `/etc/fstab`
- `nosuid`: will ignore the SUID bit and make it just like an ordinary file.
- `noexec`: will prevent execution of files from this partition
- `nodedv`: ignores devices.

Mounting partitions

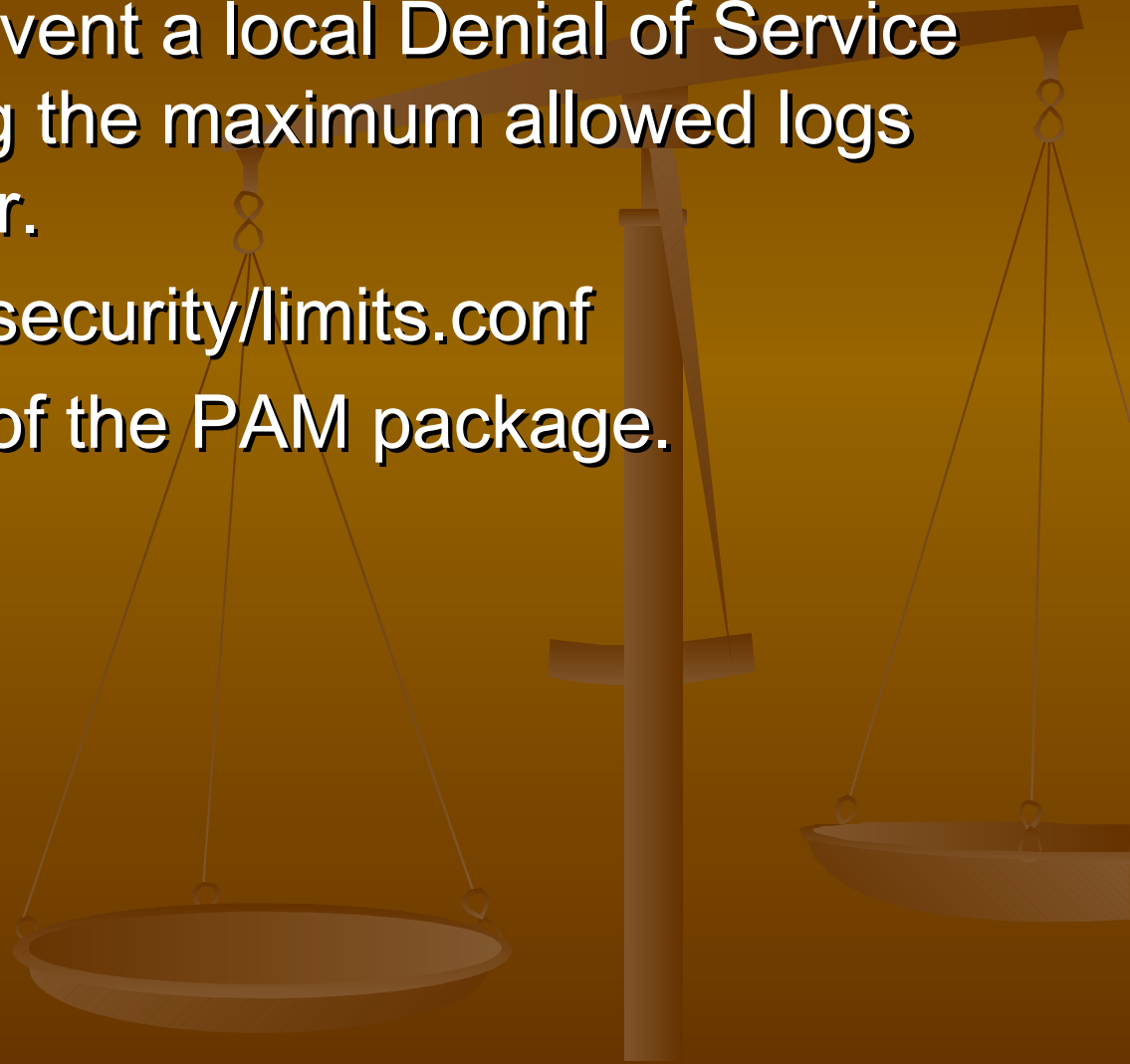
- However... these settings can easily be circumvented by executing a non-direct path.
- However... setting /tmp to noexec will stop the majority of exploits designed to be executed directly from /tmp
- However... placing /tmp in nonexec mode can prevent certain scripts from executing properly.
- Design your mounting with caution.
- an example: /etc/fstab

User/Group Limitations

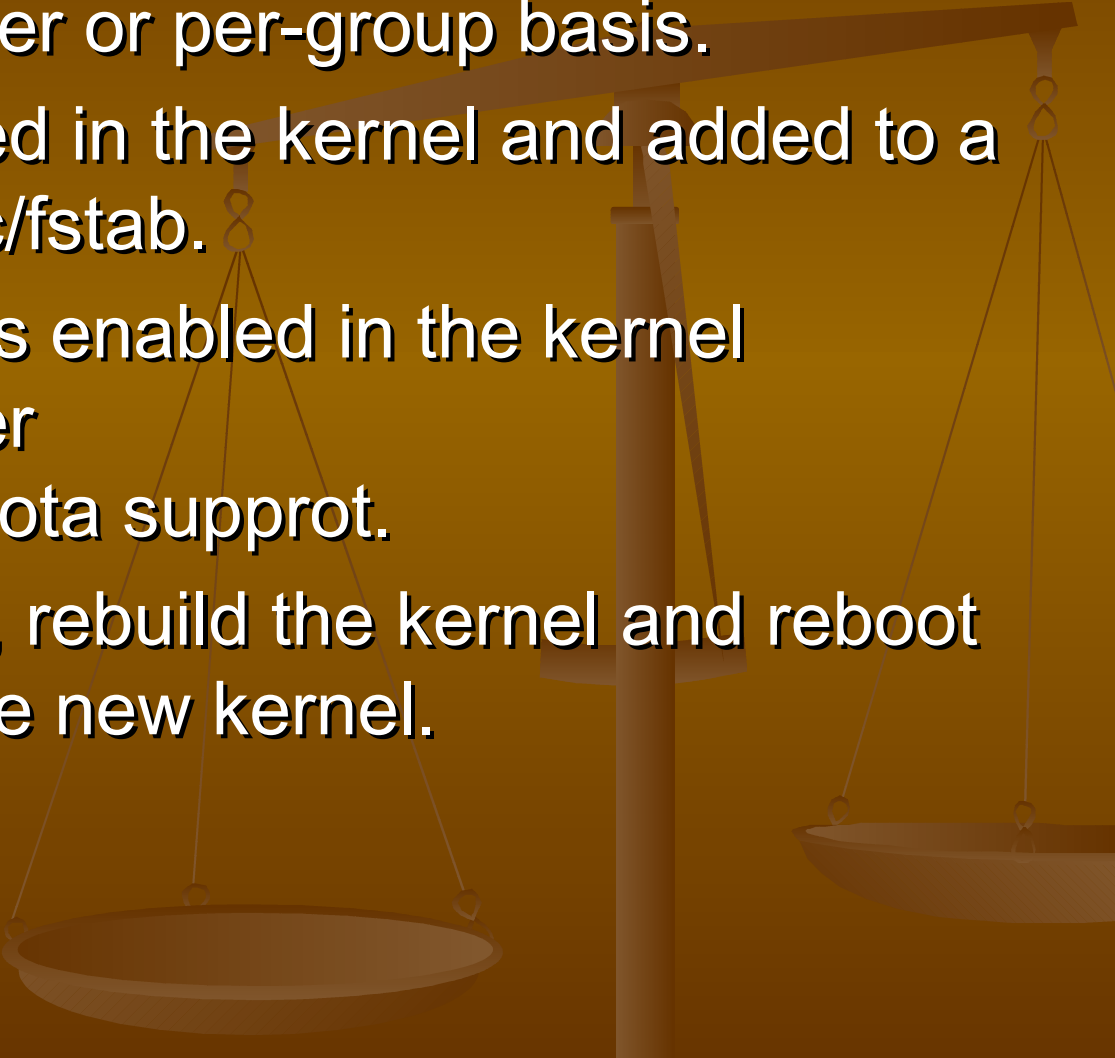


`/etc/security/limits.conf`

- Controlling resource usage can be very effective when trying to prevent a local Denial of Service or when restricting the maximum allowed logs for a group or user.
- An example: `/etc/security/limits.conf`
- `limits.conf` is part of the PAM package.



Quota

- Putting quotas on a file system restricts disk usage on a per-user or per-group basis.
 - Quotas are enabled in the kernel and added to a mount point in `/etc/fstab`.
 - The quota option is enabled in the kernel configuration under File systems-> Quota support.
 - Apply the settings, rebuild the kernel and reboot the system with the new kernel.
- 

Quota

- On every partition that you have enabled quotas, create the quota files `aquota.user` and `aquota.group`, and place them in the root of each partition.

```
# touch /tmp/aquota.user  
# touch /tmp/aquota.group  
# chmod 600 /tmp/aquota.user  
# chmod 600 /tmp/aquota.group
```

- After adding and configuring the quota files, we need to add the quota script to the boot runlevel.
- For example (in Gentoo) :

```
# rc-update add quota boot
```

Quota

- Now we will configure the system to check the quotas once a week by adding the following line to `/etc/crontab`:

```
0 3 * * 0 /usr/sbin/quotacheck -avug
```

- After rebooting the machine, it is time to setup the quotas for the users and groups.
- `edquota -u someuser` will start your favorite editor (defined in `$EDITOR`) and let you edit the quotas of the user `someuser`.
- `Edquota -g somegroup` will do the same thing for groups.

Quota

Quotas for user someuser:

```
/dev/sda4: blocks in use: 2594, limits (soft = 5000, hard = 6500)  
          inodes in use: 356, limits (soft = 1000, hard = 1500)
```

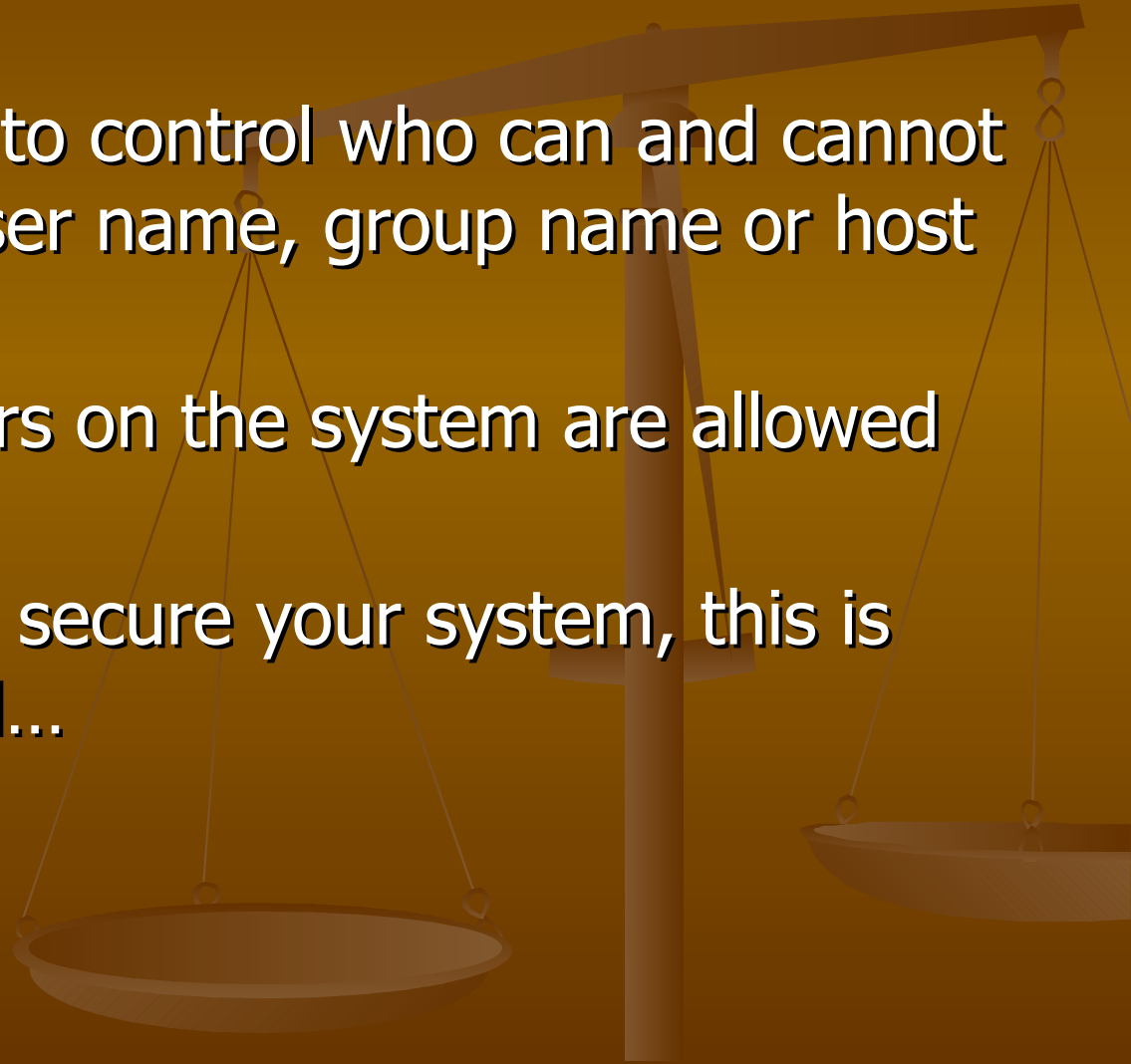


/etc/login.defs

- login.defs is a configuration control definitions file for the login (play with it)
- For example, You can force your users to change their password every X days, while “encouraging” them to replace it after $Y \leq X$ days
 - `PASS_MAX_DAYS 14 # will force changing after 14 days`
 - `PASS_WARN_AGE 7 # will start warning after 7 days`

/etc/login.access

- login.access provides a login access control table.
- This table is used to control who can and cannot login, based on user name, group name or host name.
- By default, all users on the system are allowed to login.
- But if you want to secure your system, this is not recommended...



File permissions



World readable

- Normal users should not have access to configuration files or passwords
- An attacker can steal passwords from databases or websites and use them to deface or even delete data.
- If a file should only be used by root, assign it with the permissions 0600 and assign it to the correct user via 'chown'.

World/Group writable

- World-writeability should be eliminated by default. This script can help:

```
# /usr/bin/find / -type f \( -perm -2 -o -perm -20 \) \ -exec ls  
-lg {} \; 2>/dev/null >writable.txt  
# /usr/bin/find / -type d \( -perm -2 -o -perm -20 \) \ -exec ls  
-ldg {} \; 2>/dev/null >>writable.txt
```

This will create a huge file with permissions of all files having either write permission set to the group or everybody.

You should take the files you found afterwards (found in writable.txt) and eliminate world writeability by executing `/bin/chmod o-w` on the files.

SUID/SGID files

- Files with the SUID or SGID bit execute with privileges of the *owning* users or group and not the user executing the file.
- Normally these bits are used on the files that must run as root in order to do what they do.
- This can easily lead to local root compromises.
- SUID or SGID set bits should be avoided at any cost, unless you know what you do.
- If you are not familiar with a SUIDed file, `chmod -s` him.

SUID/SGID files

- The following script can help you finding SUIDed/SGIDed files.

```
# /usr/bin/find / -type f \( -perm -004000 -o -perm -002000 \) \  
-exec ls -lg {} \; 2>/dev/null >suidfiles.txt
```

- Switch off the SUID bit on programs like ping, mount, umount, chfn, chsh, newgrp, suidperl, pt_chown and traceroute simply by `chmod -s` each, depends on the level of your hardening.
- (Please) do leave the SUID bit on programs like su, passwd, gpasswd, qmail-queue, unix_chkpwd and pwdb_chkpwd, otherwise you will not be able to su and receive mail when needed.

SUID/SGID binaries and Hard links

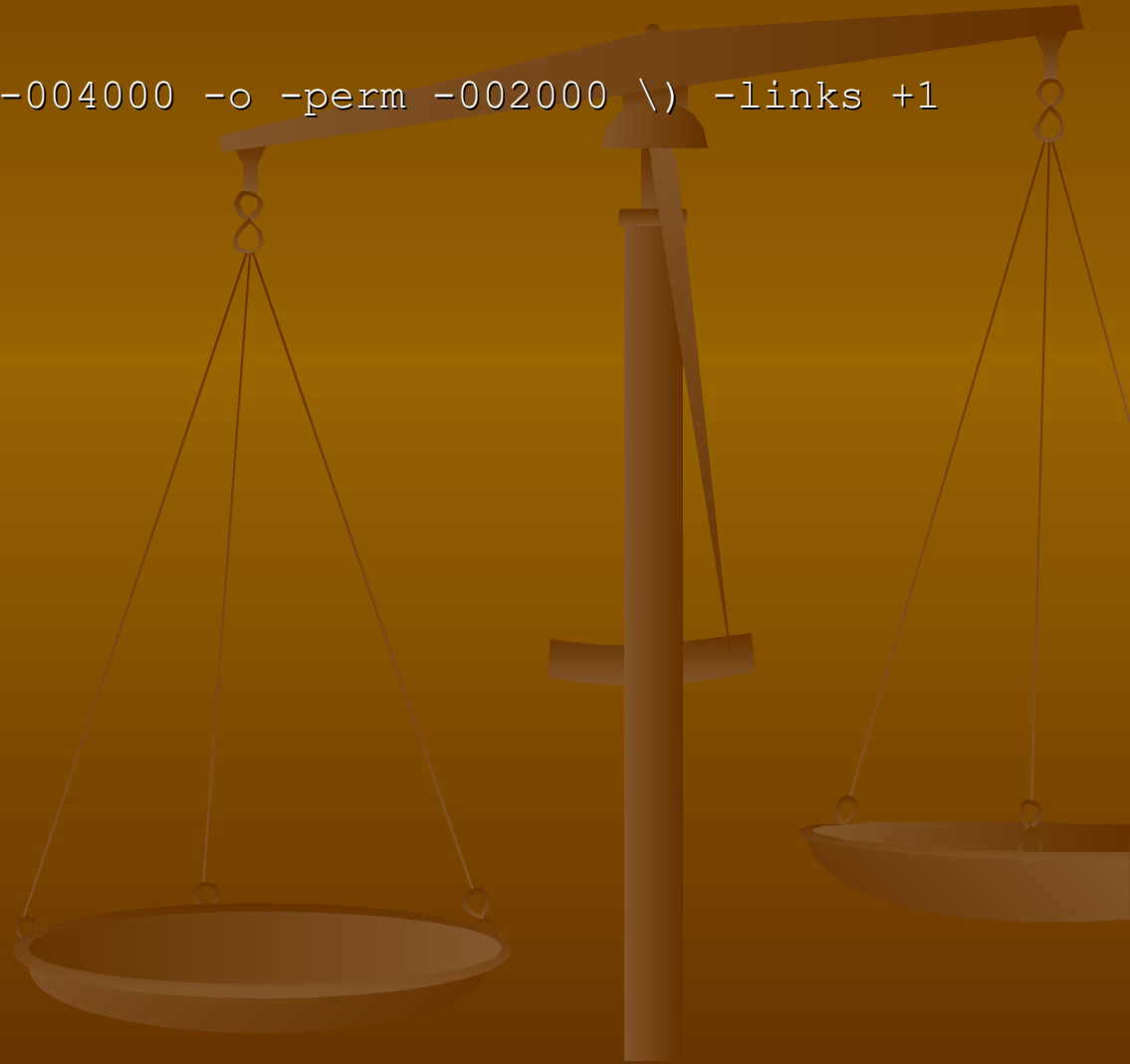
- A file is considered deleted when there are no more links pointing to it.
- (A file does not exist, if there are 0 links to it)
- You might consider a non-deleted file as deleted although you.. deleted it.
- To check how many links a file has, you can use the stat command. For example:

```
$ stat /bin/su
File: `/bin/su'
Size: 29350 Blocks: 64 IO Block: 131072 regular file
Device: 900h/2304d Inode: 2057419 Links: 1
Access: (4711/-rws--x--x) Uid: ( 0/ root) Gid: ( 0/ root)
Access: 2005-02-07 01:59:35.000000000 +0000
Modify: 2004-11-04 01:46:17.000000000 +0000
Change: 2004-11-04 01:46:17.000000000 +0000
```

SUID/SGID binaries and Hard links

- You can also find multiply linked SUID/GUID binaries

```
$ find / -type f \( -perm -004000 -o -perm -002000 \) -links +1  
-ls
```



PAM

- PAM (Password Authentication Protocol) is a suite of shared libraries that provide user authentication in programs.
- For example: `/etc/pam.d/passwd` and `/etc/pam.d/sshd` will add the cracklib which will ensure that the user passwords are at least 8 characters and contain a minimum of 2 digits, 2 other characters, and more than 3 characters different from the last password.
- This forces a user to choose good password (password policy).

TCP Wrappers

- TCP Wrappers control access to services normally run by inetd/xinetd. Follow the instructions of each to install TCP Wrappers.
- For example, in `/etc/hosts.deny`

```
ALL:PARANOID
```

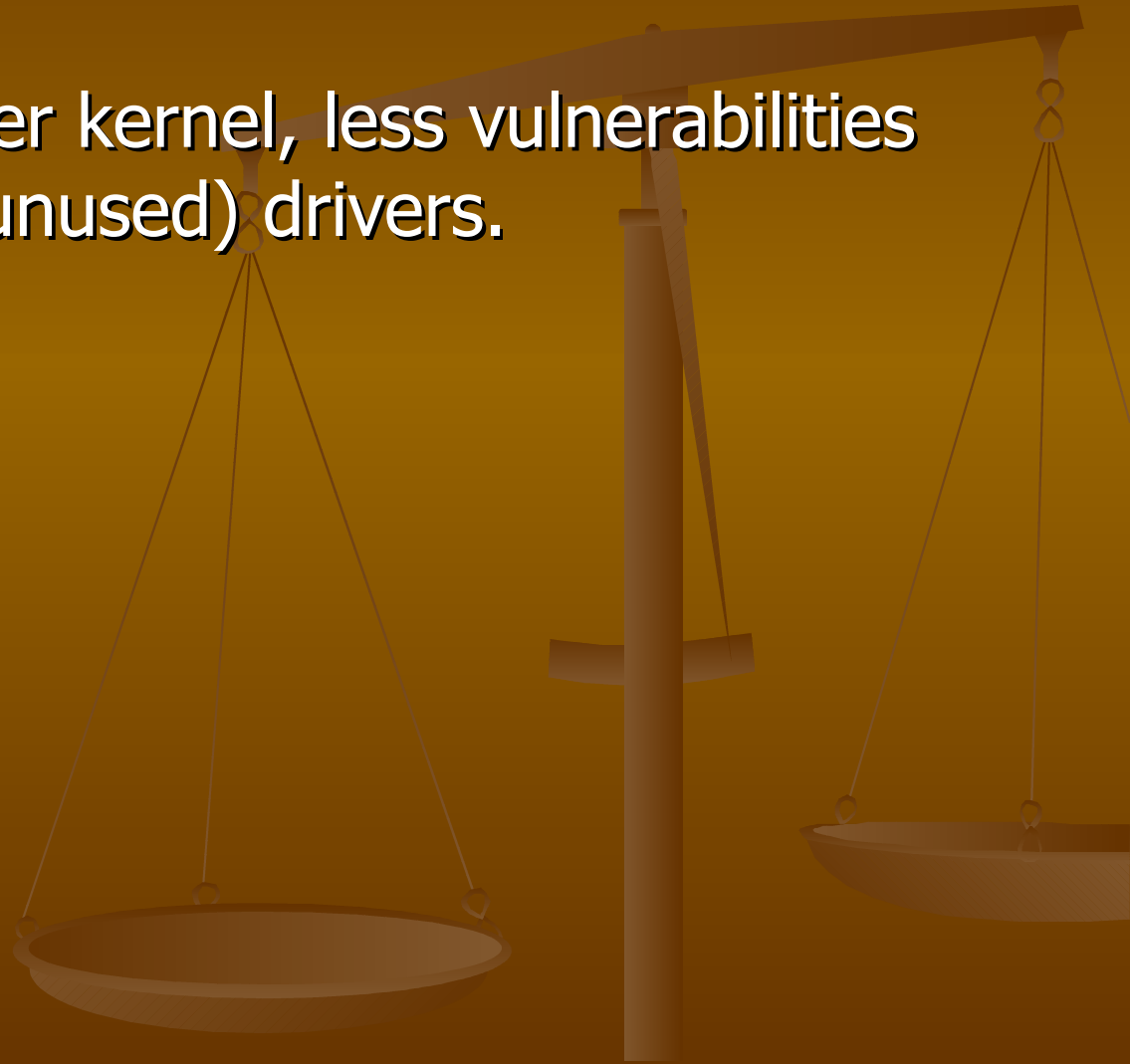
in `/etc/hosts.allow`

```
ALL: LOCAL @wheel
```

- Those settings are applied for services that use TCP Wrappers only (such as `tcpd`)

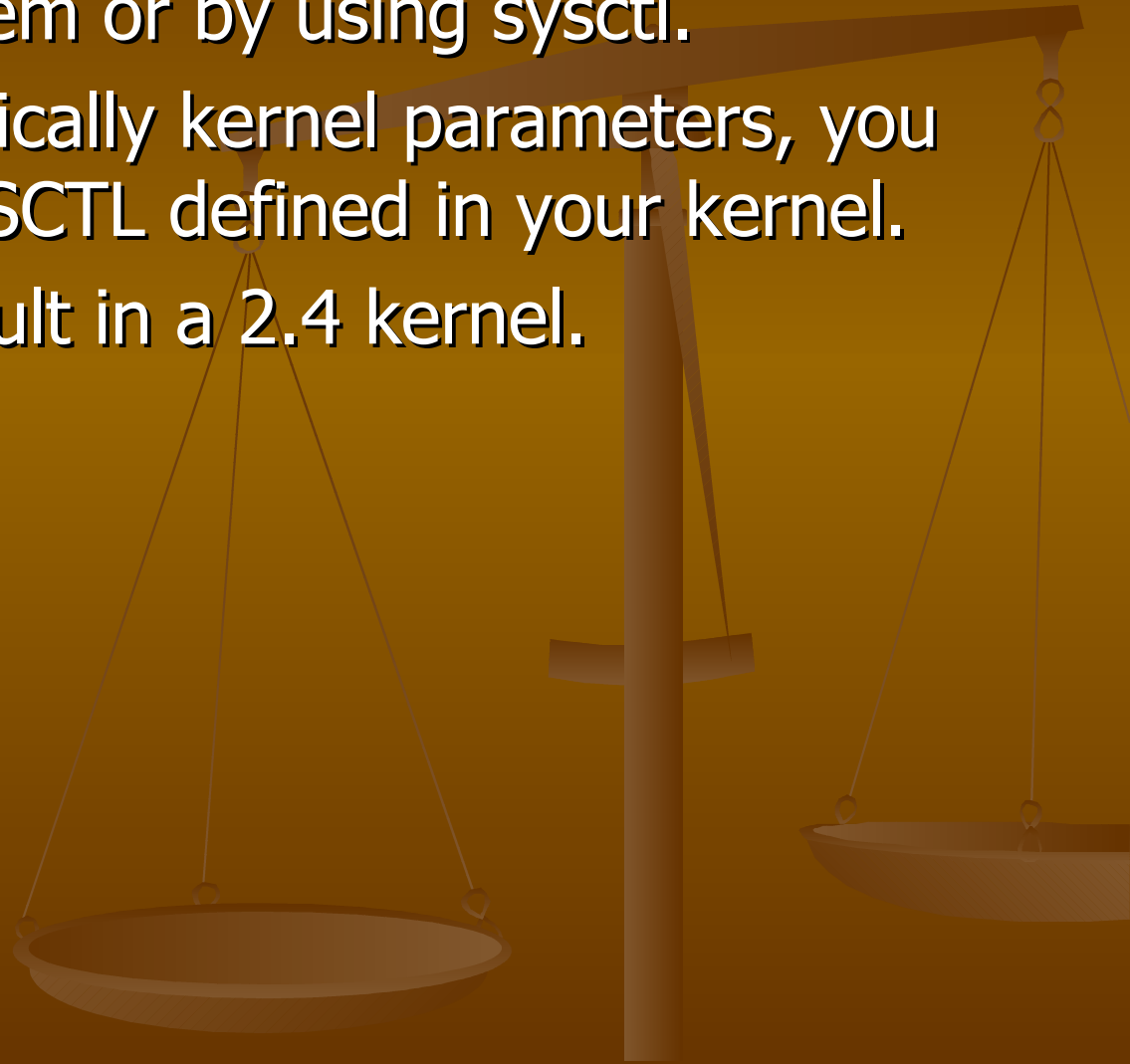
Kernel Security

- Basic rule: Removing everything that you do not need.
- The results: smaller kernel, less vulnerabilities from vulnerable (unused) drivers.



The proc filesystem

- Many kernel parameters can be altered through the /proc file system or by using sysctl.
- To change dynamically kernel parameters, you need CONFIG_SYSCTL defined in your kernel.
- This is on by default in a 2.4 kernel.



The proc filesystem

- Example #1: Deactivate IP forwarding for a multi-homed host:

```
# /bin/echo "0" > /proc/sys/net/ipv4/ip_forward
```

- Example #2: Drop ping packets (to prevent extra information on the net):

```
# /bin/echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_all
```

- Example #3: Ignore broadcast pings (to prevent spoofing and smurt attacks (ICMP type 0 ping)):

```
# /bin/echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

- Example #4: Disable source routed packets (attackers can use source routing to generate traffic pretending to arrive from inside your network):

```
# /bin/echo "0" > /proc/sys/net/ipv4/conf/all/accept_source_route
```

- Example #5: Disable redirect acceptable (ICMP redirects can be used to alter your routing tables, possibly to a malicious end):

```
# /bin/echo "0" > /proc/sys/net/ipv4/conf/all/accept_redirects
```

```
# /bin/echo "0" > /proc/sys/net/ipv4/conf/all/secure_redirects
```

- Example #6: Protect against bad error messages:

```
# /bin/echo "1" >
```

```
/proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
```