

# Rules for administration

I have seen a lot of servers in my time, and often I can deduct from a glimpse of the configuration if the administrator has any clue to UNIX.

I have tried to list some of the rules that if implemented rigidly across an enterprise can have immense impact on the stability and security of the infrastructure.

Note: the recommendations are written with a focus on UNIX and infrastructure components such as routers and switches - YMMV.

The purpose of these rules are

- to show beginners the basic functionality that can improve the infrastructure a lot
- to provide some "out of the box" settings which can be implemented with few resource, have great impact and with very small risk of bad influence to the infrastructure

Note: some might consider it bad influence that they are not able to login to root remotely, I consider this a nice side-effect that forces them to consider their working habits and improve them :-)

You won't find any fancy and confusing bells and whistles here, but the rules are a starting point - feel free to do more, but do nothing less!

## Network time - NTP

Use a correct time by configuring NTP on all equipment. Use it well and don't abuse random time servers found on the Internet. First select a time server, which should be close and one you are allowed to use.

You can add the following to your NTP time configuration file, often found as `/etc/ntp.conf`

```
server 0.pool.ntp.org
server 1.pool.ntp.org
server 2.pool.ntp.org
```

or even better a localized version - since I am in Denmark:

```
server 0.dk.pool.ntp.org
server 1.dk.pool.ntp.org
server 2.dk.pool.ntp.org
```

If your ISP provides a time service you can also use that - most serious ISPs do

## Default Publication

that.

So I use the NTP server at Cybercity which is conveniently named:  
ntp.cybercity.dk.

Also you should make sure that time is initialized correctly upon boot by specifying a time server for boot. OpenBSD does this using the /etc/rc.conf.local file:

```
rdate_flags="-n ntp.cybercity.dk"
```

More information about NTP can be found at: <http://www.pool.ntp.org/>

Benefits: all applications running on the server will benefit from correct time and all log files will include timestamps that can be used for correlating events across systems. Further by setting the correct time zone the server will work across daylight savings time etc.

Problems: none - this will have zero negative impact

## Secure Shell - OpenSSH configuration

You need secure shell when working with UNIX and many network equipment vendors also support Secure Shell. Even though Secure Shell is standardized in IETF I only use the OpenSSH version from <http://www.openssh.org>  
Configuration of the OpenSSH daemon sshd is done through the server configuration file sshd\_config often found as /etc/ssh/sshd\_config:

```
Port 12345
Protocol 2
PermitRootLogin no
PubkeyAuthentication yes
PasswordAuthentication no
ChallengeResponseAuthentication no
```

These are my recommended settings allowing only user login and only using keys. Changing the default port is also nice because you wont get as many bruteforce attempts and disabling password use for SSH login also makes bruteforce fruitless. When changing the port I recommend changing your client configuration file \$HOME/.ssh/config:

```
Host *
Port 12345
Protocol 2,1 # protocol 1 still used on Cisco PIX and others
Host aix1
Port 1234 # this server uses a non-non-standard port ;-)
Host solaris1
User demo # this server needs me to login as 'demo'
```

The settings above apply to a UNIX server and probably not possible on network equipment such as routers. More information can be found at the OpenSSH home page <http://www.openssh.org> and also the Snail book "SSH: The Secure Shell *The Definitive Guide*" <http://www.snailbook.com/>.

Benefits: accountability - users will login to the server using their personal user id, nobody logs into the root account directly. Added benefit though is that using ssh-agent people can avoid having to type in passwords, which means you can choose loong, individual and complex passwords on all servers!

Problems: few problems should be expected, but some people will need to learn about keys and use of ssh-agent for storing the key during day to day work.

## Choosing a root password

\*BEEEEPP\* wrong answer. Dont choose root passwords generate them!  
Using Password Safe or other tool you can generate a root password which are better (more random) than most of us can choose.

Especially DONT use the name of anyone, anything, anything resembling names, anything that has meaning - in short generate passwords!

Having unix4ever or P@ssw0rd as the root password is not nearly sufficient to keep honest people honest!

Root passwords need to be strong such as: 5zSP^OQ!J or \xEX0RNMh or 2/g.-f!9Gj or .... (dont use these exact passwords ;-)

Benefits: you seldom actually use the root password so you can make a stronger and longer lasting password. Save the passwords in a password database - special applications such as Password Safe <http://passwordsafe.sourceforge.net/> or Keychain Access on Mac OS X and be confident that bruteforcing them will take a long time.

Problems: people cannot remember them - which can become a problem when they need them. Some people behave irrationally when forced to look up root passwords - I think this is just the way it should be.

Having easy to guess root passwords on business critical servers is evidence of negligence on the administrators part!

## Got mail?

Make sure that important messages from the system is not lost or forgotten by configuring the mailserver and .forward files.

Most UNIX systems include a mailserver that can send mail and most UNIX systems run periodic jobs from cron doing maintenance. Some (mostly) commercial systems can also tell if disks are about to fail, or have failed in case of RAID systems. These mails are typically sent to root on the systems.

My advice is to always make a .forward on systems that you dont frequently log into, the format is just an email address where you want the emails forwarded.

Root mails go to a shared mailbox, personal userids can forward to the same or your personal mail account.

```
root@laura:hlk# cat ~root/.forward
operations@domain.example
root@laura:hlk# ^D
hlk@laura:hlk$ cat .forward
hlk@domain.example
hlk@laura:hlk$
```

Benefits: you will get that mail from the RAID system telling you that a disk failed or you will get that daily security mail telling you that some binaries changed on your system - without you doing an upgrade!

## Default Publication

Problems: people don't like to get too much mail, which is why you should implement filtering and system mail should be sent to a central mailbox, neatly sorted by server :-)

Note: in some cases when a server is behind a firewall preventing it from sending mail directly - a good thing - you need to configure the mailserver to forward mail to a *smart host* this can be done using the DS option in sendmail.cf.

```
# "Smart" relay host (may be null)
DSmymailhost.domain.example
```

More to come

I have planned to add further rules to this page - but you can also suggest rules, send them to [hik@kramse.dk](mailto:hik@kramse.dk)