Linux Security Primer

Piero Calucci

SISSA eLab – Trieste

December 18th 2007

Outline: User Security

1. Security what?
2. Password Selection
3. Storage of Sensitive Data
4. Spam, Phishing, Viruses and other beasts

Outline: Host Security

Outline: Network Security

A Word of Wisdom

«There is a ton of evidence both in computing and outside of it which shows that **poor security** can be very much **worse** than **no security at all**.
In particular stuff which makes users think they are secure but is worthless is very dangerous indeed.»

Alan Cox
Linux Kernel mailing list
October 25th, 2007

Part I

Introduction and User Security

What do we need to protect from?

- automated attacks
- targeted attacks
- insiders
- errors

What do we need to protect from?

- automated attacks
- targeted attacks
- insiders
- errors

What do we need to protect from?

- automated attacks
- targeted attacks
- insiders
- errors

What do we need to protect from?

- automated attacks
- targeted attacks
- insiders
- errors

What do we need to protect?

Short answer: everything.

- machines that operate on valuable data
- machines that are in any way connected and can be used to later attack other machines that operate on valuable data
- any other machine

What do we need to protect?

Short answer: everything.

- machines that operate on valuable data
- machines that are in any way connected and can be used to later attack other machines that operate on valuable data
- any other machine

What do we need to protect?

Short answer: everything.

- machines that operate on valuable data
- machines that are in any way connected and can be used to later attack other machines that operate on valuable data
- any other machine

What do we need to protect?

Short answer: everything.

- machines that operate on valuable data
- machines that are in any way connected and can be used to later attack other machines that operate on valuable data
- any other machine

Have a Strong Password

OK, everybody heard this one more times than there are stars in the sky. . . and everyday machines are compromised using ‘admin’ as a password to login as `root`.

- the name of your cat is not a good password
- the incredibly long and difficult password you use for your free email account is not a good `root` password
- using cracklib is good
- running John the Ripper over your own password file now and then is good
- using different means of authentication is even better

Have a Strong Password

OK, everybody heard this one more times than there are stars in the sky. . . and everyday machines are compromised using ‘admin’ as a password to login as `root`.

- the name of your cat is not a good password
- the incredibly long and difficult password you use for your free email account is not a good `root` password
- using cracklib is good
- running John the Ripper over your own password file now and then is good
- using different means of authentication is even better

Have a Strong Password

OK, everybody heard this one more times than there are stars in the sky. . . and everyday machines are compromised using ‘admin’ as a password to login as `root`.

- the name of your cat is not a good password
- the incredibly long and difficult password you use for your free email account is not a good `root` password
- using cracklib is good
- running John the Ripper over your own password file now and then is good
- using different means of authentication is even better

Have a Strong Password

OK, everybody heard this one more times than there are stars in the sky. . . and everyday machines are compromised using ‘admin’ as a password to login as `root`.

- the name of your cat is not a good password
- the incredibly long and difficult password you use for your free email account is not a good `root` password
- using cracklib is good
- running John the Ripper over your own password file now and then is good
- using different means of authentication is even better

Have a Strong Password

OK, everybody heard this one more times than there are stars in the sky. . . and everyday machines are compromised using ‘admin’ as a password to login as `root`.

- the name of your cat is not a good password
- the incredibly long and difficult password you use for your free email account is not a good `root` password
- using cracklib is good
- running John the Ripper over your own password file now and then is good
- using different means of authentication is even better

Have a Strong Password

OK, everybody heard this one more times than there are stars in the sky. . . and everyday machines are compromised using ‘admin’ as a password to login as `root`.

- the name of your cat is not a good password
- the incredibly long and difficult password you use for your free email account is not a good `root` password
- using cracklib is good
- running John the Ripper over your own password file now and then is good
- using different means of authentication is even better

Have a Strong Password

OK, everybody heard this one more times than there are stars in the sky. . . and everyday machines are compromised using ‘admin’ as a password to login as `root`.

- the name of your cat is not a good password
- the incredibly long and difficult password you use for your free email account is not a good `root` password
- using cracklib is good
- running John the Ripper over your own password file now and then is good
- using different means of authentication is even better

Know How Data Are Saved

- a central shared storage system is easy to backup and easy to protect
- workstation-local storage is much harder to backup and protect
- a mixed solution has the worst parts of both

Backups are as valuable as live data

- backups should be protected at least as much as live data
- backup protection is quite different from live date protection
- keep backups offline and they are safe from online attacks
- don't keep backup media (be it CD, tapes, external disks) where they can be stolen or copied

Online threats

- Unsolicited Bulk Email a.k.a. «spam»
- «phishing» and related malicious activities
  can be both «generic», easier to detect and block, and targeted to you / your organization, much more dangerous
- downloaded viruses and other malware, ranging from half-innocent adware, to keyloggers, to nasty bots, to rootkits

Spam

- more than a half of email messages are spam
- this is at best a huge resource waste

There is no definitive solution, however several partial ones are available:

- blacklist-based rejection at the mail gateway
- pattern matching over message content
- bayesian filters

They can of course be combined together for increased effectiveness.

Spam

- more than a half of email messages are spam
- this is at best a huge resource waste

There is no definitive solution, however several partial ones are available:

- blacklist-based rejection at the mail gateway
- pattern matching over message content
- bayesian filters

They can of course be combined together for increased effectiveness.

Anti-Spam Solutions

Blacklists

- easy to set up
- fast if applied at the first gateway
- lists need to be continuously updated (at least daily, maybe hourly)
- high risk of false positives if you expect to receive mail from Third World countries, or from users connected through sloppy ISPs
- may depend on external resources

Anti-Spam Solutions

Blacklists

- easy to set up
- fast if applied at the first gateway
- lists need to be continuously updated (at least daily, maybe hourly)
- high risk of false positives if you expect to receive mail from Third World countries, or from users connected through sloppy ISPs
- may depend on external resources

Anti-Spam Solutions /2

Pattern Matching over Message Content

- highly effective
- can be tuned to balance risk of false positives against effectiveness
- easy to customize with site-specific rules
- very CPU- and memory-intensive
- rules need to be updated and checked

Anti-Spam Solutions /2

Pattern Matching over Message Content

- highly effective
- can be tuned to balance risk of false positives against effectiveness
- easy to customize with site-specific rules
- very CPU- and memory-intensive
- rules need to be updated and checked

Anti-Spam Solutions /3

Bayesian Filters

- very easy to customize for each user
- automatically updates itself, following evolving spam patterns
- user interaction required
- CPU- and memory-intensive, requires access to storage for token database
- long start-up time, requires several thousands known good and bad messages to be effective

Anti-Spam Solutions /3

Bayesian Filters

- very easy to customize for each user
- automatically updates itself, following evolving spam patterns
- user interaction required
- CPU- and memory-intensive, requires access to storage for token database
- long start-up time, requires several thousands known good and bad messages to be effective

Phishing

Mail-based phishing can be treated as spam, and in fact a large fraction of spam is phishing-related.
This however doesn't work for

- targeted phishing, i.e. messages forged especially for you or your organization
- phishing that is not mail-based, e.g. using instant messengers, VoIP, blogs, web-based forums

The only real solution is user education: make people aware of the threat and make them able to recognize forged messages of whatever origin.

Phishing

Mail-based phishing can be treated as spam, and in fact a large fraction of spam is phishing-related.
This however doesn't work for

- targeted phishing, i.e. messages forged especially for you or your organization
- phishing that is not mail-based, e.g. using instant messengers, VoIP, blogs, web-based forums

The only real solution is user education: make people aware of the threat and make them able to recognize forged messages of whatever origin.

Viruses and Downloaded Malware

Once upon a time Linux was considered virus-free.
This is not the case any more.

Sure, ActiveX do not run under Linux, and most malware one can be infected by surfing the web is specific to that other Operating System.
Still vulnerabilities have been found for every single Linux application and (rarely) for the Linux kernel itself.

Viruses and Downloaded Malware

Once upon a time Linux was considered virus-free.
This is not the case any more.

Sure, ActiveX do not run under Linux, and most malware one can be infected by surfing the web is specific to that other Operating System.
Still vulnerabilities have been found for every single Linux application and (rarely) for the Linux kernel itself.

Viruses and Downloaded Malware /2

A Linux infection path is typically:

- malicious user gets unprivileged shell access through a vulnerable exposed application (e.g. a mail server)
- the attacker operates a local privilege escalation exploiting some other vulnerable application or the kernel itself
- once the attacker has root privileges, he installs a rootkit that somehow patches the running kernel, completely hiding malicious activities (i.e. involved files, processes and network connections become invisible even to the legitimate privileged user)

All of this can be highly automated.

Viruses and Downloaded Malware /2

A Linux infection path is typically:

- malicious user gets unprivileged shell access through a vulnerable exposed application (e.g. a mail server)
- the attacker operates a local privilege escalation exploiting some other vulnerable application or the kernel itself
- once the attacker has root privileges, he installs a rootkit that somehow patches the running kernel, completely hiding malicious activities (i.e. involved files, processes and network connections become invisible even to the legitimate privileged user)

All of this can be highly automated.

Viruses and Downloaded Malware /2

A Linux infection path is typically:

- malicious user gets unprivileged shell access through a vulnerable exposed application (e.g. a mail server)
- the attacker operates a local privilege escalation exploiting some other vulnerable application or the kernel itself
- once the attacker has root privileges, he installs a rootkit that somehow patches the running kernel, completely hiding malicious activities (i.e. involved files, processes and network connections become invisible even to the legitimate privileged user)

All of this can be highly automated.

Viruses and Downloaded Malware /2

A Linux infection path is typically:

- malicious user gets unprivileged shell access through a vulnerable exposed application (e.g. a mail server)
- the attacker operates a local privilege escalation exploiting some other vulnerable application or the kernel itself
- once the attacker has root privileges, he installs a rootkit that somehow patches the running kernel, completely hiding malicious activities (i.e. involved files, processes and network connections become invisible even to the legitimate privileged user)

All of this can be highly automated.

Part II

Host Security

Keep Your Systems Updated

Every day a new vulnerability is discovered.

Not all of these may be critical to your systems, however you should think before deciding not to apply a provided patch – and you should as well think before you do apply it:

- what is the patch about?
- is the vulnerability critical to the affected software?
  is it critical in your environment?
- what could be the consequences of not applying the patch?
  (think worst case)
- can you foresee any collateral damage?

Keep Your Systems Updated

Every day a new vulnerability is discovered.
Not all of these may be critical to your systems, however you should think before deciding not to apply a provided patch – and you should as well think before you do apply it:

- what is the patch about?
- is the vulnerability critical to the affected software?
  is it critical in your environment?
- what could be the consequences of not applying the patch?
  (think worst case)
- can you foresee any collateral damage?

Do You Trust Your Update Servers?

Vulnerability discovered, patch released, system updated. Now you think you are safe... **think better!**

- have you tested patches before applying?
  this takes time and effort
- do you trust update servers? (assuming that you at least trust the vendor)
  update servers probably use the same operating system as yours: what if they had been hacked and rogue patches added?
- do you trust DNS?
  if you pull updates from a rogue server you could install rootkits instead of patches

Do You Trust Your Update Servers?

Vulnerability discovered, patch released, system updated. Now you think you are safe. . . **think better!**

- have you tested patches before applying?
  this takes time and effort
- do you trust update servers? (assuming that you at least trust the vendor)
  update servers probably use the same operating system as yours: what if they had been hacked and rogue patches added?
- do you trust DNS?
  if you pull updates from a rogue server you could install rootkits instead of patches

Do You Trust Your Update Servers?

Vulnerability discovered, patch released, system updated. Now you think you are safe. . . **think better!**

- have you tested patches before applying?
  this takes time and effort
- do you trust update servers? (assuming that you at least trust the vendor)
  update servers probably use the same operating system as yours: what if they had been hacked and rogue patches added?
- do you trust DNS?
  if you pull updates from a rogue server you could install rootkits instead of patches

Do You Trust Your Update Servers?

Vulnerability discovered, patch released, system updated. Now you think you are safe. . . **think better!**

- have you tested patches before applying?
  this takes time and effort
- do you trust update servers? (assuming that you at least trust the vendor)
  update servers probably use the same operating system as yours: what if they had been hacked and rogue patches added?
- do you trust DNS?
  if you pull updates from a rogue server you could install rootkits instead of patches

Do You Trust Your Update Servers?

Vulnerability discovered, patch released, system updated. Now you think you are safe... **think better!**

- have you tested patches before applying?
  this takes time and effort
- do you trust update servers? (assuming that you at least trust the vendor)
  update servers probably use the same operating system as yours: what if they had been hacked and rogue patches added?
- do you trust DNS?
  if you pull updates from a rogue server you could install rootkits instead of patches

Do You Trust Your Update Servers? /2

- any decent developer/package maintainer cryptographically signs packages
- enable signature checking if you use an auto-update service
- manually verify signatures if you manually update
- downloading both a package and the public key needed for signature verification from the same server at the same time is useless
- protect your keyring

Do You Trust Your Update Servers? /2

- any decent developer/package maintainer cryptographically signs packages
- enable signature checking if you use an auto-update service
- manually verify signatures if you manually update
- downloading both a package and the public key needed for signature verification from the same server at the same time is useless
- protect your keyring

Do You Trust Your Update Servers? /2

- any decent developer/package maintainer cryptographically signs packages
- enable signature checking if you use an auto-update service
- manually verify signatures if you manually update
- downloading both a package and the public key needed for signature verification from the same server at the same time is useless
- protect your keyring

Do You Trust Your Update Servers? /2

- any decent developer/package maintainer cryptographically signs packages
- enable signature checking if you use an auto-update service
- manually verify signatures if you manually update
- downloading both a package and the public key needed for signature verification from the same server at the same time is useless
- protect your keyring

Do You Trust Your Update Servers? /2

- any decent developer/package maintainer cryptographically signs packages
- enable signature checking if you use an auto-update service
- manually verify signatures if you manually update
- downloading both a package and the public key needed for signature verification from the same server at the same time is useless
- protect your keyring

What if You Cannot Patch?

- a patch might not be available
- collateral damage might be unacceptable

What if You Cannot Patch?

- a patch might not be available
- collateral damage might be unacceptable

e.g.: HPC cluster with a lot of custom software:

- kernel upgrade would force network drivers recompilation and/or upgrade
- network drivers recompilation would force middleware recompilation and/or upgrade
- middleware recompilation would force recompilation of all user application
- half of them would not work any more. . .

What if You Cannot Patch?

- a patch might not be available
- collateral damage might be unacceptable
- find workarounds if possible
- have multiple layers of security around the vulnerable component
- check often for signs of compromise
- strictly control access

Protect Your Bootloader

- If a machine is not locked in a server room chances are that ordinary, untrusted users can reboot it.
- If they can select an alternate boot device they could just install the OS of their choice.
- If they have access to the boot loader command line they could start the installed operating system in some «maintenance» mode with most security features disabled.
  - Just add `init=/bin/sh` to a linux kernel command line and you get a root shell and no logs.

Protect Your Bootloader

- If a machine is not locked in a server room chances are that ordinary, untrusted users can reboot it.
- If they can select an alternate boot device they could just install the OS of their choice.
- If they have access to the boot loader command line they could start the installed operating system in some «maintenance» mode with most security features disabled.
  - Just add `init=/bin/sh` to a linux kernel command line and you get a root shell and no logs.

Protect Your Bootloader

- If a machine is not locked in a server room chances are that ordinary, untrusted users can reboot it.
- If they can select an alternate boot device they could just install the OS of their choice.
- If they have access to the boot loader command line they could start the installed operating system in some «maintenance» mode with most security features disabled.
  - Just add `init=/bin/sh` to a linux kernel command line and you get a root shell and no logs.

Protect Your Bootloader

- If a machine is not locked in a server room chances are that ordinary, untrusted users can reboot it.
- If they can select an alternate boot device they could just install the OS of their choice.
- If they have access to the boot loader command line they could start the installed operating system in some «maintenance» mode with most security features disabled.
  - Just add `init=/bin/sh` to a linux kernel command line and you get a root shell and no logs.

Avoid Dual-Boot

If a machine can boot different OSes there is no practical way you can protect data belonging to one OS only from the other. If one of the installed OSes is compromised, you have to consider the whole machine compromised.

Consider using some virtualization technology if you really need to have multiple OSes available at the same station.

If you have a «main» OS and one or more «secondary» ones that are rarely used, you can go with a in-OS virtual machine, e.g. VMware or Qemu.

If several OSes are used and none can be considered the main one, go for a hypervisor solution, e.g. Xen. Modern x86 hardware can run many unmodified OSes under a hypervisor.

Avoid Dual-Boot

If a machine can boot different OSes there is no practical way you can protect data belonging to one OS only from the other. If one of the installed OSes is compromised, you have to consider the whole machine compromised.

Consider using some virtualization technology if you really need to have multiple OSes available at the same station.

If you have a «main» OS and one or more «secondary» ones that are rarely used, you can go with a in-OS virtual machine, e.g. VMware or Qemu.

If several OSes are used and none can be considered the main one, go for a hypervisor solution, e.g. Xen. Modern x86 hardware can run many unmodified OSes under a hypervisor.

Check Filesystem Permissions

- give everyone permissions they need
- don't give them anything more

So:

- check that permissions are what they should be after initial installation and after patching
- look for setuid/setgid executables
- look for world writable directories
- look for world readable configuration files

  (Wait before you `chmod 600 /etc`! most configuration files have to be world readable. Some of them however really need to be readable only by root.)

Check Filesystem Permissions

- give everyone permissions they need
- don't give them anything more

So:

- check that permissions are what they should be after initial installation and after patching
- look for setuid/setgid executables
- look for world writable directories
- look for world readable configuration files

  (Wait before you `chmod 600 /etc`! most configuration files have to be world readable. Some of them however really need to be readable only by root.)

Use Filesystem Permissions Wisely

- do not create unneeded world writable directories
- set the «sticky bit» (`chmod +t`) on world writable directories (`/tmp` should be already there, and should be enough)
- world writable directories are happier in a filesystem of their own; you will be happier if you do not depend on that filesystem availability — think local DoS
- groups can be used to share files among users; have a different group for each project/office
  (the default behavior of some RedHat-derived distros to have a different group for each user makes groups completely useless)

Use Filesystem Permissions Wisely

- do not create unneeded world writable directories
- set the «sticky bit» (`chmod +t`) on world writable directories (`/tmp` should be already there, and should be enough)
- world writable directories are happier in a filesystem of their own; you will be happier if you do not depend on that filesystem availability — think local DoS
- groups can be used to share files among users; have a different group for each project/office
  (the default behavior of some RedHat-derived distros to have a different group for each user makes groups completely useless)

Mount options

Some filesystem permissions can be completely disabled at mount time:

- `ro` read-only
- `noexec` disables execution of any binary on the filesystem (this actually works only with kernel 2.6.x and 2.4.25 or later)
- `nosuid` makes the setuid and setgid bits ineffective
- `nodev` do not allow character or block devices on the filesystem

Disable Services You don't Need

The average modern Linux distro has somewhere between 50 and 100 available services. Maybe half of them are enabled after a default install.

- Do you really need so many active services?
- Do you really need them installed at all?
- Do you know what all those services are?

Disable Services You don't Need

The average modern Linux distro has somewhere between 50 and 100 available services. Maybe half of them are enabled after a default install.

- Do you really need so many active services?
- Do you really need them installed at all?
- Do you know what all those services are?

Disable Services You don't Need

The average modern Linux distro has somewhere between 50 and 100 available services. Maybe half of them are enabled after a default install.

- Do you really need so many active services?
- Do you really need them installed at all?
- Do you know what all those services are?

Disable Services You don't Need

The average modern Linux distro has somewhere between 50 and 100 available services. Maybe half of them are enabled after a default install.

- Do you really need so many active services?
- Do you really need them installed at all?
- Do you know what all those services are?

Disable Services You don't Need /2

A typical «server» install has

- a web server, e.g. Apache, thttpd
- a mail server, e.g. Postfix, Sendmail
- a print server, e.g. CUPS
- NFS server and/or Samba
- xinetd
- sshd

You probably want SSH, but if you are installing a DHCP server or a caching web proxy you don't need any of the others!

Disable Services You don't Need /2

A typical «server» install has

- a web server, e.g. Apache, thttpd
- a mail server, e.g. Postfix, Sendmail
- a print server, e.g. CUPS
- NFS server and/or Samba
- xinetd
- sshd

You probably want SSH, but if you are installing a DHCP server or a caching web proxy you don't need any of the others!

Disable Services You don't Need /3

A typical «workstation» or «laptop» install has

- DHCP client, PPP tools and other «home» connectivity sw
- wireless connectivity tools
- X11 and some fancy graphical desktop environment
- a truckload of games and desktop gadgets
- multimedia tools
- autoeverything-wonderfilesystem, up to and including the dreadful automatic execution on CD/DVD insertion

Again, you probably need a graphical environment, but if this is going to be a software development workstation, or a central point of control for your network. . . ?

Disable Services You don't Need /3

A typical «workstation» or «laptop» install has

- DHCP client, PPP tools and other «home» connectivity sw
- wireless connectivity tools
- X11 and some fancy graphical desktop environment
- a truckload of games and desktop gadgets
- multimedia tools
- autoeverything-wonderfilesystem, up to and including the dreadful automatic execution on CD/DVD insertion

Again, you probably need a graphical environment, but if this is going to be a software development workstation, or a central point of control for your network. . . ?

Protect Services You Need

- restrict access to services from network
  e.g. a NFS server probably doesn't need to be accessed from outside your LAN
- enable and configure authentication where appropriate
- don't mix internal end external services
  e.g. the database that powers your dynamic web pages should be not the same used by your accounting office
- make servers «know» what is your trusted network and what is not
  e.g. your DNS should answer queries about internal, protected machines only if queries come from internal network

Password Aging

User accounts and passwords should expire automatically

- this reduces risk of having valid accounts for people that are not part of the organization any more
- can be used to force users to change password often
- can help in monitoring which users are active regularly
  a change in this pattern is usually something interesting to spot

Password Aging

User accounts and passwords should expire automatically

- this reduces risk of having valid accounts for people that are not part of the organization any more
- can be used to force users to change password often
- can help in monitoring which users are active regularly
  a change in this pattern is usually something interesting to spot

Password Aging

User accounts and passwords should expire automatically

- this reduces risk of having valid accounts for people that are not part of the organization any more
- can be used to force users to change password often
- can help in monitoring which users are active regularly
  a change in this pattern is usually something interesting to spot

Limit User Access per-System

User access should be limited to systems where it is actually needed.

If you use local accounts (unrecommended...) you simply need to create accounts only where they are needed.

If you have some central account management service, you probably want to limit which accounts are enabled on each and every machine, and where users can login from.

All of this can be done in the `/etc/security/access.conf` configuration file.

Limit User Access per-System

User access should be limited to systems where it is actually needed.

If you use local accounts (unrecommended...) you simply need to create accounts only where they are needed.

If you have some central account management service, you probably want to limit which accounts are enabled on each and every machine, and where users can login from.

All of this can be done in the `/etc/security/access.conf` configuration file.

Limit User Access per-System

User access should be limited to systems where it is actually needed.

If you use local accounts (unrecommended...) you simply need to create accounts only where they are needed.

If you have some central account management service, you probably want to limit which accounts are enabled on each and every machine, and where users can login from.

All of this can be done in the `/etc/security/access.conf` configuration file.

Limit User Access per-System /2

Sample `/etc/security/access.conf`

`+:ALL:ALL`

allow (+) everybody (1st `ALL`) from everywhere (2nd `ALL`)

Limit User Access per-System /2

Sample `/etc/security/access.conf`

```
-:ALL EXCEPT root:ALL
```

allow only root

Limit User Access per-System /2

Sample `/etc/security/access.conf`

```
-:ALL EXCEPT root:LOCAL
```

disallow local login to everybody except root

Limit User Access per-System /2

Sample `/etc/security/access.conf`

```
-:ALL EXCEPT root:LOCAL
+:ALL: .mydomain.net
-:ALL:ALL
```

access is granted or denied based on the first matching line

so here we allow only root on the local console, everybody from the `mydomain.net` *domain and nobody else*

Limit User Access per-Service

Limit user access only to services actually needed:

- users who don't have to manage web pages do not need shell access to the web server
  (and even them who do have to manage web pages can probably do so without direct access to the web server machine itself)
- nobody but mail administrators need shell access to mail servers
- the same goes for directory servers, network management boxes, shared storage, proxy servers...

«Test» Accounts

Every now and then you will need some dummy account to perform some test on a pre-production machine or service.
Be extra careful when creating these accounts otherwise...

- you will probably create a test account with username `test`
- you will probably set its password to «test1»
- you will probably use the account for a couple of days, then forget about it
- you will probably complete all tests on the machine, then put it in a production (i.e. hostile) environment with the test account still enabled

THIS IS VERY BAD!

«Test» Accounts

Every now and then you will need some dummy account to perform some test on a pre-production machine or service.
Be extra careful when creating these accounts otherwise...

- you will probably create a test account with username `test`
- you will probably set its password to «test1»
- you will probably use the account for a couple of days, then forget about it
- you will probably complete all tests on the machine, then put it in a production (i.e. hostile) environment with the test account still enabled

THIS IS VERY BAD!

Don't Use `root` When Not Needed

Use the privileged account only when really needed.

- **never** use root account to surf the web or read email
- take extra care when you are root: a `-r` after `rm` can be a typo for the regular user, a disaster for root
- most software can be compiled and installed as a regular user after root has created the destination path; if not, then the installation procedure is trying to do something strange (at best) and needs to be investigated

Don't Give Away `root`

Regular users should not get privileged access.

- you should arrange so that no privileged operations are needed for regular user work (e.g. all needed software is already installed on workstations, no configuration changes are needed, etc.)
- if for some special need a regular user needs to execute some privileged operation, use `sudo`

  `sudo` allows a regular user to perform certain actions with elevated privileges; this must be configured by the superuser and should be done with extreme care
  Keep in mind that allowing a random user to do `sudo bash` is the same as setting the empty password for root! (and allowing arbitrary package installation is more or less the same)

Watch Your Logs

Keeping an eye on logs is one of the most effective ways to detect intrusion attempts and ongoing attacks: if you know your system well enough, and you know typical log patterns, you will easily spot any difference from previous days.

Most Linux distros have some simple built-in log analysis tool (e.g. `logwatch`, `logdigest`, `epylog`) that can produce nicely formatted reports and email them to you. Reading and understanding those reports should be the first morning duty of every security conscious sysadmin (and user who has admin privileges on his own workstation).

Remote Log Storage

Once a machine has been compromised you cannot trust its logs any more: if the attacker has been able to change system files, he could have changed (or simply deleted) local logs as well.
This makes log analysis useless, unless log files are stored on some different machine.

Traditional `syslog` uses the `@hostname` syntax to specify that a log should be sent to some other host instead of written to a local file.

Newer `syslog-ng` uses the `tcp ("address" port(number))` syntax in the `destination` statement.

Beware however that even if remotely-stored logs are safe from deletion and modification, they can still be poisoned with fake messages that make the interesting events more difficult to find and analyze.

Remote Log Storage

Once a machine has been compromised you cannot trust its logs any more: if the attacker has been able to change system files, he could have changed (or simply deleted) local logs as well.
This makes log analysis useless, unless log files are stored on some different machine.

Traditional `syslog` uses the `@hostname` syntax to specify that a log should be sent to some other host instead of written to a local file.

Newer `syslog-ng` uses the `tcp ("address" port(number))` syntax in the `destination` statement.

Beware however that even if remotely-stored logs are safe from deletion and modification, they can still be poisoned with fake messages that make the interesting events more difficult to find and analyze.

AIDE

The Advanced Intrusion Detection Environment is an intrusion detection system that checks file integrity.

- you should start with a known-clean system
- a crypto-safe checksum is computed for every file and saved to a database along with file metadata
- the whole database is cryptographically signed and stored to some safe place, e.g. read-only media
- when checking your system, you recompute checksums and metadata and compare with those saved in the database

AIDE

The Advanced Intrusion Detection Environment is an intrusion detection system that checks file integrity.

- you should start with a known-clean system
- a crypto-safe checksum is computed for every file and saved to a database along with file metadata
- the whole database is cryptographically signed and stored to some safe place, e.g. read-only media
- when checking your system, you recompute checksums and metadata and compare with those saved in the database

ChkRootKit

`chkrootkit` is basically a collection of shell scripts plus a few binaries that locally check for signs of well-known rootkits.
Since it is script-based, it depends on several system utilities, so it is best to provide those from external, read-only media if you suspect that the system has been somehow modified.

`chkrootkit` is likely to report some false positive (many perl packages are detected as suspect), and it is not guaranteed to detect every possible malware piece, but it is very fast and easy to run, so it is a good starting point.

Hardening

System hardening is not about patching specific vulnerabilities or about configuring a system where no compromise can occur.

Rather, it is about making existing vulnerabilities more difficult to exploit, and about making it difficult for the attacker to get any real advantage from an exploited vulnerability.

e.g.: there might be a vulnerability in our web software we don't know about that allows an attacker to execute a shell by sending a carefully crafted HTTP request.

If we do not allow `/bin/sh` to be a child of the web server process, the vulnerability is still there, but cannot be effectively exploited.

Hardening

System hardening is not about patching specific vulnerabilities or about configuring a system where no compromise can occur.

Rather, it is about making existing vulnerabilities more difficult to exploit, and about making it difficult for the attacker to get any real advantage from an exploited vulnerability.

e.g.: there might be a vulnerability in our web software we don't know about that allows an attacker to execute a shell by sending a carefully crafted HTTP request.

If we do not allow `/bin/sh` to be a child of the web server process, the vulnerability is still there, but cannot be effectively exploited.

The grsecurity Kernel

`grsecurity` is a kernel patch that

- allows you to define fine-grained ACL on files and processes
- detects abnormal process behavior (e.g. «too many» segmentation violations) and optionally disables execution of offending processes
- enforces memory protection, disabling execution of any page that has been writable in a process' history
- randomizes address space of processes and PIDs; configures larger than default entropy pools
- hides most processes and large parts of the `/proc` filesystem from unprivileged users

Installing and configuring grsecurity

- only few distros include a grsecurity-enabled kernel; you need to download the patch from `grsecurity.net` and patch a vanilla kernel
- address space protection, `/proc` protection and randomization features must be configured in the kernel itself; most of them can be enabled at runtime via `sysctl`
- ACLs must be stored in a configuration file and loaded at runtime
- a few userspace utilities are provided, but are not strictly needed

SELinux

Security-Enhanced Linux is part of modern vanilla kernel and is included in most distros and is most often part of any default installation.

- basically a large set of ACLs on all filesystem objects
- ACLs are stored in filesystem metadata and can be enabled at runtime
- most filesystem utilities need to be made SELinux-aware

AppArmor

Apparmor is a different implementation of ACLs.

- installed with several distros (openSUSE, Ubuntu)
- ACLs are stored in configuration files
- ACLs apply only to specified «objects», so unconfigured applications have no policy

grsecurity vs SELinux

grsecurity	SELinux
external patch	vanilla kernel, many distros
ACLs and address space protection	ACLs only
small user base, you need to write your own ACLs	large user base, ACLs preconfigured in many distros
configuration stored in plain text files, easy to copy and backup	configuration in FS metadata, SELinux-enabled utilities needed to copy and backup

SELinux vs AppArmor

SELinux	**AppArmor**
vanilla kernel, many distros	external patch, but supported by several distros
ACLs stored in FS metadata	ACLs stored in configuration files
ACLs apply to whole system	ACLs apply only to targeted applications

Virtualization for Increased Security

Virtualization technologies have been developed to allow multiple OS instances to run concurrently on the same hardware. They can as well be used to enforce resource separation where having separate real hardware would be too expensive.

Linux-friendly virtualization technologies include:

- Linux VServer
- Xen
- VMware

Virtualization with VServer

Linux VServer is a very lightweight kernel patch that provides virtual environments at the userspace level.

- a single Linux kernel runs on real hardware
- different userspace environments can be completely different Linux installation (even different distros) or can be clones of a single base
- very easy to detect that you are running in a virtual environment
- good separation at the process level
- no separation at all at the kernel level
- kernel modification needed; only Linux available

Virtualization with Xen

Xen is a hypervisor that can run both unmodified OSes and paravirtualized OSes.

- paravirtualization available only with Linux and certain *BSD
- full virtualization can run almost any OS, but requires hardware support and has higher overhead
- can run completely different OSes concurrently, but multiple kernels means also larger memory footprint

Virtualization with VMware

VMware is actually a family of (commercial) products, some of which are available at no cost.
A VMware virtual machine provides full hardware emulation (you even have a virtual BIOS).

- can run any unmodified OS
- excellent separation among well-behaved OSes
- special instructions are available to guest OS to communicate to the virtualization software itself

Part III

Network Security

Netfilter overview

In modern (2.6) Linux kernel the IP filtering facility is set up, maintained and inspected by means of the `iptables` command. Several different tables may be defined. Each table contains a number of built-in chains and optional user-defined chains.

Each chain is a list of rules which can match a set of packets. Each rule specifies what to do with a packet that matches. This is called a «target», which may be

- a jump to a user-defined chain in the same table
- a terminating target that makes the packet exit the table, being either dropped, forwarded, or allowed to enter or exit the machine
- a non-terminating target that may perform some action while making the packet continue rule traversal

Netfilter overview

In modern (2.6) Linux kernel the IP filtering facility is set up, maintained and inspected by means of the `iptables` command. Several different tables may be defined. Each table contains a number of built-in chains and optional user-defined chains.

Each chain is a list of rules which can match a set of packets. Each rule specifies what to do with a packet that matches. This is called a «target», which may be

- a jump to a user-defined chain in the same table
- a terminating target that makes the packet exit the table, being either dropped, forwarded, or allowed to enter or exit the machine
- a non-terminating target that may perform some action while making the packet continue rule traversal

Netfilter overview /2

The **filter** table contains the built-in chains INPUT (for packets destined to local sockets), FORWARD (for packets being routed through the box), and OUTPUT (for locally-generated packets).

The **nat** table is consulted when a packet that creates a new connection is encountered. It contains the built-ins PREROUTING (for altering packets as soon as they come in), OUTPUT (for altering locally-generated packets before routing), and POSTROUTING (for altering packets as they are about to go out). It can change source and destination addresses and ports for each connection.

The **mangle** table is used for specialized packet alteration.

Netfilter overview /2

The **filter** table contains the built-in chains INPUT (for packets destined to local sockets), FORWARD (for packets being routed through the box), and OUTPUT (for locally-generated packets).

The **nat** table is consulted when a packet that creates a new connection is encountered. It contains the built-ins PREROUTING (for altering packets as soon as they come in), OUTPUT (for altering locally-generated packets before routing), and POSTROUTING (for altering packets as they are about to go out). It can change source and destination addresses and ports for each connection.

The **mangle** table is used for specialized packet alteration.

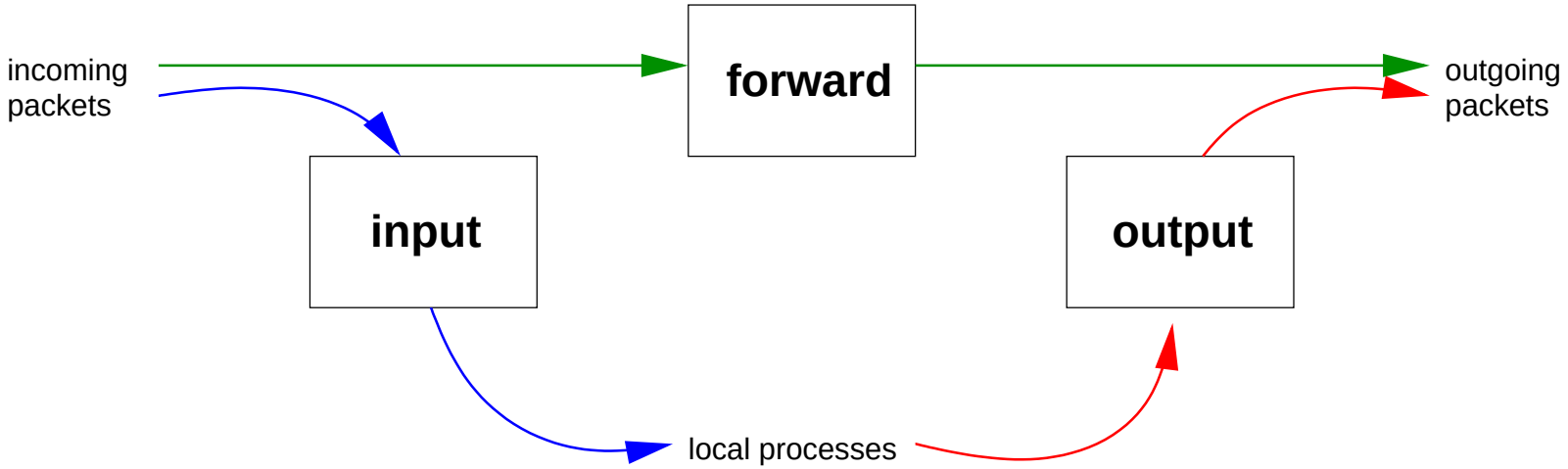
Netfilter overview /2

The **filter** table contains the built-in chains INPUT (for packets destined to local sockets), FORWARD (for packets being routed through the box), and OUTPUT (for locally-generated packets).

The **nat** table is consulted when a packet that creates a new connection is encountered. It contains the built-ins PREROUTING (for altering packets as soon as they come in), OUTPUT (for altering locally-generated packets before routing), and POSTROUTING (for altering packets as they are about to go out). It can change source and destination addresses and ports for each connection.

The **mangle** table is used for specialized packet alteration.

The filter table



Common targets

ACCEPT let the packet through

DROP drop the packet on the floor

REJECT drop the packet (as in DROP) and send back an error packet

RETURN stop processing in this chain and return the packet to calling chain

LOG generate a klog message about the packet
this is a non-terminating target

SNAT modify source address of the packet; only valid in the POSTROUTING chain of the nat table

DNAT modify destination address of the packet; only valid in the PREROUTING and OUTPUT chains of the nat table

(. . . and many more)

Common targets

ACCEPT let the packet through

DROP drop the packet on the floor

REJECT drop the packet (as in DROP) and send back an error packet

RETURN stop processing in this chain and return the packet to calling chain

LOG generate a klog message about the packet
this is a non-terminating target

SNAT modify source address of the packet; only valid in the POSTROUTING chain of the nat table

DNAT modify destination address of the packet; only valid in the PREROUTING and OUTPUT chains of the nat table

(. . . and many more)

Common targets

ACCEPT let the packet through

DROP drop the packet on the floor

REJECT drop the packet (as in DROP) and send back an error packet

RETURN stop processing in this chain and return the packet to calling chain

LOG generate a klog message about the packet
this is a non-terminating target

SNAT modify source address of the packet; only valid in the POSTROUTING chain of the nat table

DNAT modify destination address of the packet; only valid in the PREROUTING and OUTPUT chains of the nat table

(. . . and many more)

Default targets

Built-in chains have a default target (or policy), i.e. a target that is applied if no rule in the chain matches.

When building a secure system is extremely useful to have DROP as default policy, so that everything is dropped unless explicitly allowed to pass. Just take care not to lock yourself out. . .

Netfilter configuration

Netfilter rules can be configured with the `iptables` command line utility one rule at a time or by feeding a rule file to `iptables-restore`.

Basic operations include:

- create, destroy or flush (empty) a user-defined chain
- set a default policy for a built-in chain
- appending or inserting in a specified place a new rule into an existing chain
- deleting or replacing an existing rule
- listing rules and counters and optionally zeroing counters

Netfilter configuration /2

Rules can match packets by looking at:

- interface the packet is coming from or heading to
- protocol type
- source and destination IP addresses
- address type (unicast, multicast or broadcast)
- source and destination ports for TCP and UDP, or ICMP type
- TCP connection state and/or any combination of TCP flags
- an arbitrary tag («mark») assigned to a packet by a preceding rule

  this allows state propagation inside the netfilter framework, but do not alter in any way the packet itself, so it is lost as soon as the packet is sent out to the network or passed to user space

Netfilter configuration examples

```
iptables -t filter -P INPUT DROP
```

«DROP everything as default policy for the INPUT chain in the **filter** table».

`-t filter` can be omitted, since filter is the default table.

```
iptables -A INPUT -i lo -j ACCEPT
```

«In the INPUT chain (of the **filter** table) ACCEPT (everything) coming from the loopback interface».

```
iptables -A INPUT -p icmp -m icmp --icmp-type
echo-request -s !  147.122.0.0/16 -j DROP
```

```
iptables -A INPUT -p icmp -j ACCEPT
```

«Load (`-m`) the ICMP matching extension; DROP pings from everywhere but (!) our class B network».

«ACCEPT (any other) ICMP packet».

Netfilter configuration examples

```
iptables -t filter -P INPUT DROP
```

«DROP everything as default policy for the INPUT chain in the **filter** table».

`-t filter` can be omitted, since filter is the default table.

```
iptables -A INPUT -i lo -j ACCEPT
```

«In the INPUT chain (of the **filter** table) ACCEPT (everything) coming from the loopback interface».

```
iptables -A INPUT -p icmp -m icmp --icmp-type
echo-request -s !  147.122.0.0/16 -j DROP
```

```
iptables -A INPUT -p icmp -j ACCEPT
```

«Load (`-m`) the ICMP matching extension; DROP pings from everywhere but (!) our class B network».

«ACCEPT (any other) ICMP packet».

Netfilter configuration examples

```
iptables -t filter -P INPUT DROP
```

«DROP everything as default policy for the INPUT chain in the **filter** table».

`-t filter` can be omitted, since filter is the default table.

```
iptables -A INPUT -i lo -j ACCEPT
```

«In the INPUT chain (of the **filter** table) ACCEPT (everything) coming from the loopback interface».

```
iptables -A INPUT -p icmp -m icmp --icmp-type
echo-request -s !  147.122.0.0/16 -j DROP
```

```
iptables -A INPUT -p icmp -j ACCEPT
```

«Load (`-m`) the ICMP matching extension; DROP pings from everywhere but (!) our class B network».

«ACCEPT (any other) ICMP packet».

Netfilter configuration examples /2

```
iptables -A INPUT -p tcp -m tcp -s 147.122.0.0/16
--dport 22 -syn -j ACCEPT
```

«ACCEPT SYN packets for SSH from our network».

```
iptables -A INPUT -m state --state ESTABLISHED -j
ACCEPT
```

«Load the TCP state matching module; ACCEPT any packet that is part of an already established connection».

This includes SSH packets from connections allowed by the previous rule, but not arbitrary non-SYN packets.

Netfilter is fully stateful.

Netfilter configuration examples /2

```
iptables -A INPUT -p tcp -m tcp -s 147.122.0.0/16
--dport 22 -syn -j ACCEPT
```

«ACCEPT SYN packets for SSH from our network».

```
iptables -A INPUT -m state --state ESTABLISHED -j
ACCEPT
```

«Load the TCP state matching module; ACCEPT any packet that is part of an already established connection».
This includes SSH packets from connections allowed by the previous rule, but not arbitrary non-SYN packets.
Netfilter is fully stateful.

Netfilter configuration examples /3

```
iptables -A inputs -p tcp -m tcp --dport 113 -j
REJECT
iptables -A inputs -p udp -m udp --dport 113 -j
REJECT
```

«REJECT all ident packets».

There are still servers on the net that make ident queries. Rejecting with an error packet these queries instead of just dropping them usually speeds up things, since we do not need to wait for a timeout to expire.

Netfilter configuration examples /4

```
iptables -A INPUT -p tcp -m tcp --syn -m limit
--limit 5/minute -j LOG --log-level debug
--log-prefix "iptables:  "
```

«Load the TCP matching and rate limiting modules; log SYN packets, up to 5 per minute».

This could be the last rule of an INPUT chain with a default DROP policy: we log connection attempts just before dropping them, with an anti-DoS clause.

Packets that have already been ACCEPTed, DROPped or REJECTed never reach this rule.

Netfilter configuration saving and restoring

To save configuration across reboots and for efficient creation of a large number of rules netfilter provides the two commands `iptables-save` and `iptables-restore` that dump the running configuration to a plain text file and restore the configuration from the same (or possibly modified file).

In RedHat and derived distros the configuration file is usually `/etc/sysconfig/iptables`.

Egress Filtering

A firewall is not only about protecting your host or network from what comes from outside, it can be also about filtering what tries to exit from your perimeter.
You may want to do egress filtering for at least two different reasons:

- control what people do with your computing resources, e.g. no peer-to-peer file sharing allowed
- make a whole class of attacks against your network ineffective, e.g. many bots need to connect to an IRC channel to perform their evil deeds

SSH is your Friend

SSH is probably the most useful utility you have when it comes to working on several machines at one time and/or from a remote location.
SSH provides you with

- an encrypted communication channel
- several user authentication methods
- host verification
- secure file transfer
- port redirection, encrypted and authenticated tunnels

SSH for login

```
calucci@yogi $ ssh shannon
calucci@shannon's password:
calucci@shannon-01 $
```

In its simplest form `ssh` is just a replacement for older telnet. You connect to a remote host, give a password, and are presented a shell prompt.

Remote user name can be different from local using the `ssh -l user host` or `ssh user@host` syntax.

SSH for passwordless login

```
calucci@yogi $ ssh shannon
Enter passphrase for key
'/home/calucci/.ssh/id_dsa':
calucci@shannon-01 $
```

While all of this looks quite similar to password authentication there are very important differences:

- your password never leaves your local box so it is safe even if the remote box is compromised
- the same key can be used to authenticate to different remote machines and/or different remote users
- the passphrase you are asked for is used only to encrypt your private key and you can change it locally without any need to connect to remote machines the key is used to authenticate to

SSH for passwordless login

```
calucci@yogi $ ssh shannon
Enter passphrase for key
'/home/calucci/.ssh/id_dsa':
calucci@shannon-01 $
```

While all of this looks quite similar to password authentication there are very important differences:

- your password never leaves your local box so it is safe even if the remote box is compromised
- the same key can be used to authenticate to different remote machines and/or different remote users
- the passphrase you are asked for is used only to encrypt your private key and you can change it locally without any need to connect to remote machines the key is used to authenticate to

Public Key Authentication with SSH

PK authentication works by giving proof to the remote machine that you actually own a private key. The remote machine has been instructed to allow login the user who owns the private key corresponding to a public key that has been remotely installed.

It is actually simpler than it sounds. . .

1. you create on the local machine a key pair:
   `ssh-keygen -t dsa -b 1024`
   you are asked for a passphrase to protect the key: enter something reasonably long yet easy to remember;
   do not use any of your passwords!
   the keys are saved in the `.ssh` directory as `id_dsa` (private key) and `id_dsa.pub` (public key)

Public Key Authentication with SSH

PK authentication works by giving proof to the remote machine that you actually own a private key. The remote machine has been instructed to allow login the user who owns the private key corresponding to a public key that has been remotely installed.

It is actually simpler than it sounds. . .

1. you create on the local machine a key pair:
   `ssh-keygen -t dsa -b 1024`
   you are asked for a passphrase to protect the key: enter something reasonably long yet easy to remember;
   do not use any of your passwords!
   the keys are saved in the `.ssh` directory as `id_dsa` (private key) and `id_dsa.pub` (public key)

Public Key Authentication with SSH /2

2. you install your public key to the remote machine:
   `cat .ssh/id_dsa.pub | ssh shannon "cat >> .ssh/authorized_keys"`
   this assumes that your public key is `.ssh/id_dsa.pub` and that a `.ssh` directory exists in your home directory on the remote machine
   (more on this command line later)
3. now when trying to login to the remote host you are asked for the passphrase for your private key instead of your remote password

SSH agent

SSH can be made «more passwordless» than that using the `ssh-agent` to keep you private key available, so that you do not need to write your passphrase for every login, but only once per session.

If you are using a (decently configured) graphical environment, an ssh-agent is most probably already there, waiting for your input. On a text console you probably need to start it manually:

```
eval `ssh-agent`
```

this is some little shell magic: the agent starts and outputs a few lines of shell commands, the shell then evaluates («executes») these commands.

SSH agent

SSH can be made «more passwordless» than that using the `ssh-agent` to keep you private key available, so that you do not need to write your passphrase for every login, but only once per session.

If you are using a (decently configured) graphical environment, an ssh-agent is most probably already there, waiting for your input. On a text console you probably need to start it manually:

```
eval `ssh-agent`
```

this is some little shell magic: the agent starts and outputs a few lines of shell commands, the shell then evaluates («executes») these commands.

SSH agent /2

Once the agent is started, you can start adding private keys:

```
ssh-add -t 4h
Enter passphrase for /home/calucci/.ssh/id_dsa:
Identity added:  /home/calucci/.ssh/id_dsa
```

this adds your default key to the agent keyring, with a lifetime of 4 hours (it is always a good idea to specify a finite lifetime); you are prompted for your passphrase and from that point on you will no longer need it to connect to remote machines (configured to accept your key for authentication)

SSH agent /3

`ssh-add` can list loaded keys:

```
ssh-add -l
1024 21:4b:....ssh/id_dsa (DSA)
```

and remove keys from the keyring:

```
ssh-add -D
All identities removed.
```

SSH agent /3

`ssh-add` can list loaded keys:

```
ssh-add -l
1024 21:4b:....ssh/id_dsa (DSA)
```

and remove keys from the keyring:

```
ssh-add -D
All identities removed.
```

SSH Host Verification

A frequently overlooked feature of SSH is host verification:

- every SSH-enabled host has a unique identification key
- this key is presented to the client at connection request
- the client checks the host key against a database of known hosts

The most common (and quite secure) behavior of an SSH client is

- continue the authentication procedure if the host is known and the key matches
- warn the user and ask for confirmation if the host is unknown
- warn the user and abort authentication if the host is known but the key has changed

SSH Host Verification

A frequently overlooked feature of SSH is host verification:

- every SSH-enabled host has a unique identification key
- this key is presented to the client at connection request
- the client checks the host key against a database of known hosts

The most common (and quite secure) behavior of an SSH client is

- continue the authentication procedure if the host is known and the key matches
- warn the user and ask for confirmation if the host is unknown
- warn the user and abort authentication if the host is known but the key has changed

SSH Host Verification /2

A non-matching host key can be a sign of:

- a reinstalled remote host – nothing to worry about, just delete the offending key from your database (usually `.ssh/known_hosts`) and move on
- a compromised remote host with a trojan SSH
- a man-in-the-middle attack, with your packets being routed through a rogue host

. . . so don't overlook warnings about mismatched host keys.

SSH Host Verification /2

A non-matching host key can be a sign of:

- a reinstalled remote host – nothing to worry about, just delete the offending key from your database (usually `.ssh/known_hosts`) and move on
- a compromised remote host with a trojan SSH
- a man-in-the-middle attack, with your packets being routed through a rogue host

. . . so don't overlook warnings about mismatched host keys.

SSH Host Verification /2

A non-matching host key can be a sign of:

- a reinstalled remote host – nothing to worry about, just delete the offending key from your database (usually `.ssh/known_hosts`) and move on
- a compromised remote host with a trojan SSH
- a man-in-the-middle attack, with your packets being routed through a rogue host

. . . so don't overlook warnings about mismatched host keys.

SSH for Secure File Copy

The SSH-based `scp` utility can do remote file copy over an encrypted channel, with all authentication features of SSH.

Local to remote:

```
scp path1/file user@host:path2/file
```

Remote to local:

```
scp user@host:path1/file path2/file
```

Remote to remote:

```
scp userA@host1:path/file userB@host2:path2/file
```

here the connection goes from `host1` to `host2`, so host verification and user authentication can be a bit tricky

SSH for Secure File Copy

The SSH-based `scp` utility can do remote file copy over an encrypted channel, with all authentication features of SSH.

Local to remote:

```
scp path1/file user@host:path2/file
```

Remote to local:

```
scp user@host:path1/file path2/file
```

Remote to remote:

```
scp userA@host1:path/file userB@host2:path2/file
```

here the connection goes from `host1` to `host2`, so host verification and user authentication can be a bit tricky

SSH for Secure File Copy

The SSH-based `scp` utility can do remote file copy over an encrypted channel, with all authentication features of SSH.

Local to remote:

```
scp path1/file user@host:path2/file
```

Remote to local:

```
scp user@host:path1/file path2/file
```

Remote to remote:

```
scp userA@host1:path/file userB@host2:path2/file
```

here the connection goes from `host1` to `host2`, so host verification and user authentication can be a bit tricky

SSH for Secure File Copy /2

`scp -r` copies recursively entire directories.

`scp -C` (de)compresses data on the fly.
Whether this makes things faster depends on network bandwidth and available CPU power (on both sides).

`scp -l` can be used to limit used bandwidth.
Quite useful if you have several connections open at the same time, or do not want to overload a slower machine.

SSH for secure FTP

The SSH-based `sftp` does FTP over an encrypted channel, with all authentication features of SSH. Most default FTP commands are supported and behave the usual way.

SFTP is much more NAT- and firewall-friendly than plain old FTP, since only the SSH port is involved: no multiple connections, no random ports.

SSH for secure FTP

The SSH-based `sftp` does FTP over an encrypted channel, with all authentication features of SSH. Most default FTP commands are supported and behave the usual way.

SFTP is much more NAT- and firewall-friendly than plain old FTP, since only the SSH port is involved: no multiple connections, no random ports.

SSH for remote batch operation

Once you have arranged for passwordless authentication, SSH allows you to execute arbitrary commands on a remote host non-interactively:

```
ssh root@cerbero df -h /scratch
Filesystem            Size  Used Avail Use% Mounted on
iosrv:/scratch        933G  617G  317G  67% /scratch
```

This is of course extremely useful for remote system administration. In a tightly controlled environment you can even generate a passphrase-less key and use it to perform fully automated remote administration.

SSH for remote batch operation

Once you have arranged for passwordless authentication, SSH allows you to execute arbitrary commands on a remote host non-interactively:

```
ssh root@cerbero df -h /scratch
Filesystem            Size  Used Avail Use% Mounted on
iosrv:/scratch        933G  617G  317G  67% /scratch
```

This is of course extremely useful for remote system administration. In a tightly controlled environment you can even generate a passphrase-less key and use it to perform fully automated remote administration.

SSH for remote batch operation /2

An extension of the `authorized_keys` file allows you to enable certain keys to execute only defined commands – this is the ultimate SSH-based solution for remote administration automation:

if you have in `authorized_keys`

```
command="/bin/df -h" ssh-dss AAAAB3N...
```

when using the corresponding key to authenticate you are allowed to run only `df -h` (actually this is done for you by the SSH daemon).

Warning: at least some versions of OpenSSH will offer all the available keys for authentication, so if you have multiple keys configured for a single host, and have all of them loaded by the agent (or passphrase-less) you may end up using the «wrong» key. Better have all the keys passphrase-protected and loaded only one at a time, or have them in non-default location so your SSH client cannot find them.

SSH for remote batch operation /2

An extension of the `authorized_keys` file allows you to enable certain keys to execute only defined commands – this is the ultimate SSH-based solution for remote administration automation:

if you have in `authorized_keys`

```
command="/bin/df -h" ssh-dss AAAAB3N...
```

when using the corresponding key to authenticate you are allowed to run only `df -h` (actually this is done for you by the SSH daemon).

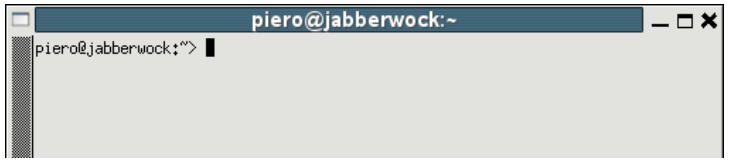
Warning: at least some versions of OpenSSH will offer all the available keys for authentication, so if you have multiple keys configured for a single host, and have all of them loaded by the agent (or passphrase-less) you may end up using the «wrong» key. Better have all the keys passphrase-protected and loaded only one at a time, or have them in non-default location so your SSH client cannot find them.

SSH for X11 forwarding

SSH can forward X11 connections, so that you are not limited to command-line operation on the remote machine:

`ssh -X -f jabberwock xterm`

(`-f` to background `ssh` on command execution)



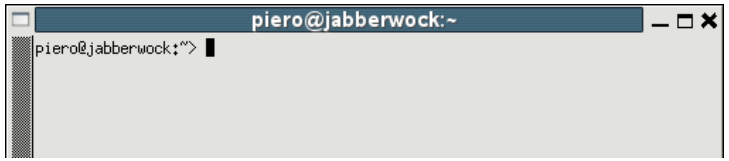
You can even have *trusted X11 forwarding* (`ssh -XY`) i.e. remote application having full access to local display. This is often needed to make applications actually work, but should be enabled with caution and only in controlled environments, since it is a significant security risk.

SSH for X11 forwarding

SSH can forward X11 connections, so that you are not limited to command-line operation on the remote machine:

`ssh -X -f jabberwock xterm`

(`-f` to background `ssh` on command execution)



You can even have *trusted X11 forwarding* (`ssh -XY`) i.e. remote application having full access to local display. This is often needed to make applications actually work, but should be enabled with caution and only in controlled environments, since it is a significant security risk.

SSH for tunneling

An incredibly powerful feature of SSH is port forwarding, that allows to connect local and remote ports through a secure channel, and can be used to connect to arbitrary remote services through a SSH gateway:

```
jabberwock$ ssh -f -L 10022:borg.hpc.sissa.it:22
calucci@shannon.sissa.it sleep 3600
```

```
jabberwock$ ssh -p 10022 root@localhost
The authenticity of host '[localhost]:10022
([127.0.0.1]:10022)' can't be established. Are you
sure you want to continue connecting (yes/no)? yes
root@localhost's password:
root@borg #
```

SSH for tunneling

An incredibly powerful feature of SSH is port forwarding, that allows to connect local and remote ports through a secure channel, and can be used to connect to arbitrary remote services through a SSH gateway:

```
jabberwock$ ssh -f -L 10022:borg.hpc.sissa.it:22
calucci@shannon.sissa.it sleep 3600
```

```
jabberwock$ ssh -p 10022 root@localhost
The authenticity of host '[localhost]:10022
([127.0.0.1]:10022)' can't be established. Are you
sure you want to continue connecting (yes/no)? yes
root@localhost's password:
root@borg #
```

SSH for tunneling

An incredibly powerful feature of SSH is port forwarding, that allows to connect local and remote ports through a secure channel, and can be used to connect to arbitrary remote services through a SSH gateway:

```
jabberwock$ ssh -f -L 10022:borg.hpc.sissa.it:22
calucci@shannon.sissa.it sleep 3600

jabberwock$ ssh -p 10022 root@localhost
The authenticity of host '[localhost]:10022
([127.0.0.1]:10022)' can't be established. Are you
sure you want to continue connecting (yes/no)? yes
root@localhost's password:
root@borg #
```

SSH for tunneling

An incredibly powerful feature of SSH is port forwarding, that allows to connect local and remote ports through a secure channel, and can be used to connect to arbitrary remote services through a SSH gateway:

```
jabberwock$ ssh -f -L 10022:borg.hpc.sissa.it:22
calucci@shannon.sissa.it sleep 3600
```

```
jabberwock$ ssh -p 10022 root@localhost
The authenticity of host '[localhost]:10022
([127.0.0.1]:10022)' can't be established. Are you
sure you want to continue connecting (yes/no)? yes
root@localhost's password:
root@borg #
```

NAT and Private Networks

While not Linux-specific, a powerful network security device is a NAT box that can «decouple» your internal network from the outside world.

- your internal network uses only non-routable addresses e.g. in the `10.0.0.0/8` range
- private addresses are fine for internal communications, but cannot be used on the Internet since packets would be dropped at the first router
- on the network perimeter there is a NAT box that translates private addresses to a (usually smaller) set of routable ones
- the translation is set up for the first outgoing packet, but there is no way an external unsolicited packet can have its destination address translated
- this way you can connect from inside to everywhere, but the reverse connection is not possible

NAT and security

NAT is sometimes presented as the ultimate network security solution. However...

- it offers no protection at all from threats that involve an outgoing connection (e.g. downloaded malware)
- its effectiveness is greatly weakened on IPv6-enabled networks
- if the NAT box itself is compromised, it cannot protect your network any more

...so view it as another layer of security, but do not trust it as the only protection for your network.

NAT and security

NAT is sometimes presented as the ultimate network security solution. However...

- it offers no protection at all from threats that involve an outgoing connection (e.g. downloaded malware)
- its effectiveness is greatly weakened on IPv6-enabled networks
- if the NAT box itself is compromised, it cannot protect your network any more

...so view it as another layer of security, but do not trust it as the only protection for your network.

Online Resources

Security in General

Internet Storm Center `http://isc.sans.org/diary.html`

SecurityFocus `http://www.securityfocus.com/`

Linux Security `http://www.linuxsecurity.com/`

the Honeynet Project
`http://www.honeynet.org/papers/honeynet/`

Online Resources /2

Applications

OpenSSH	http://www.openssh.com/
AIDE	http://sourceforge.net/projects/aide
ChkRootKit	http://www.chkrootkit.org/
CrackLib	http://sourceforge.net/projects/cracklib/
John the Ripper	http://www.openwall.com/john/
SpamAssassin	http://spamassassin.apache.org/
BogoFilter	http://bogofilter.sourceforge.net/

Online Resources /3

Kernel-related

Linux VServer http://www.linux-vserver.org/

VMware http://www.vmware.com/

QEMU http://fabrice.bellard.free.fr/qemu/

Xen http://www.xensource.com/

Security-Enhanced Linux http://www.nsa.gov/selinux/

AppArmor http://www.novell.com/linux/security/apparmor/

grsecurity http://www.grsecurity.net/

Netfilter http://www.netfilter.org/

that's all folks!

<calucci@sissa.it>

made with LaTeX