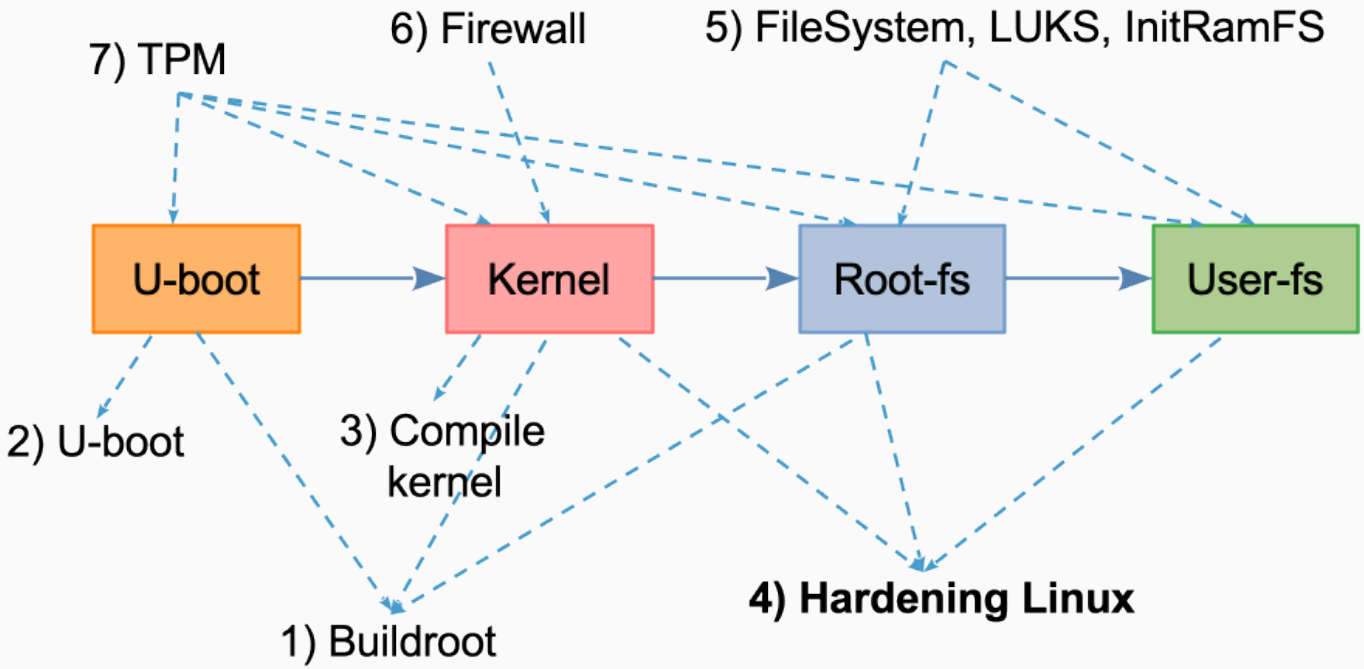
MASTER OF SCIENCE
IN ENGINEERING

MA_SeS - Linux Hardening

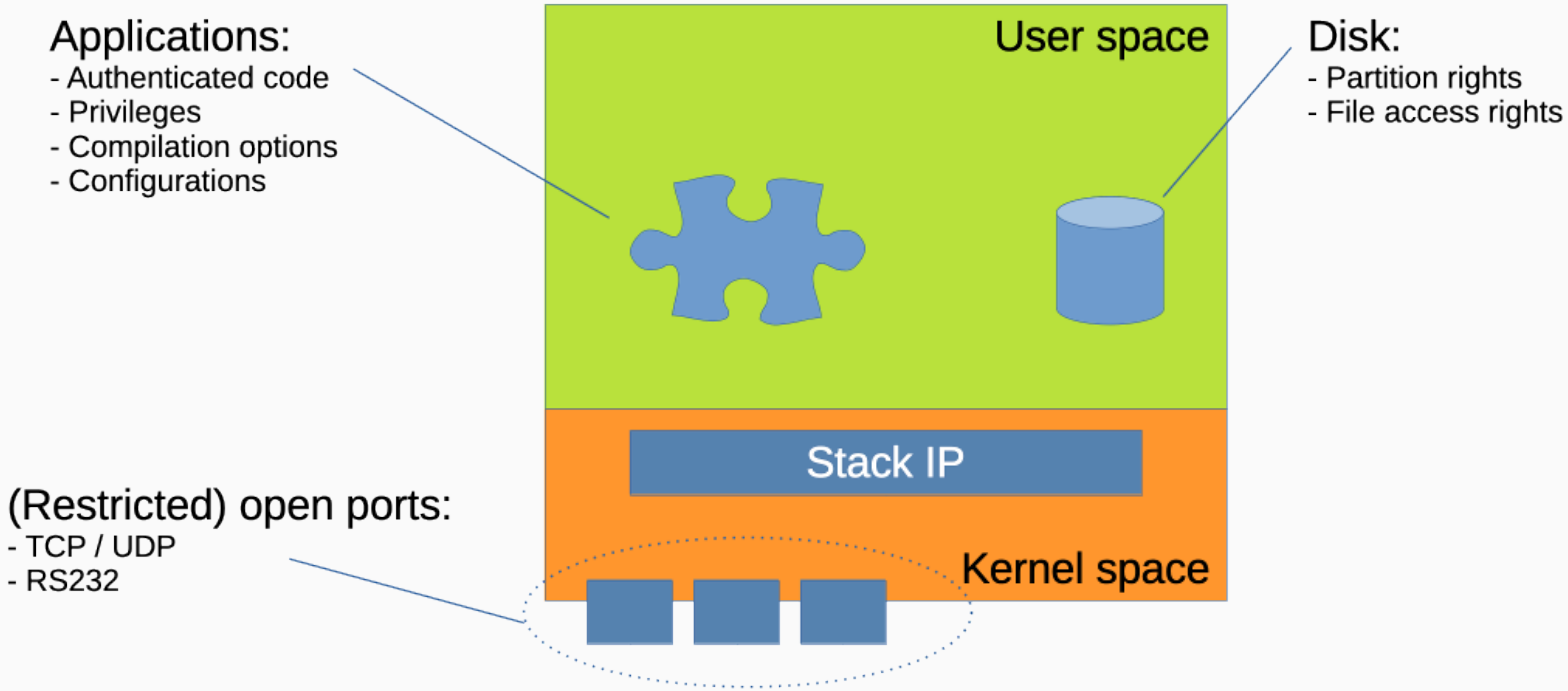
Luca Haab, Michael Mäder, Florent Glück

October 27, 2025

Overview

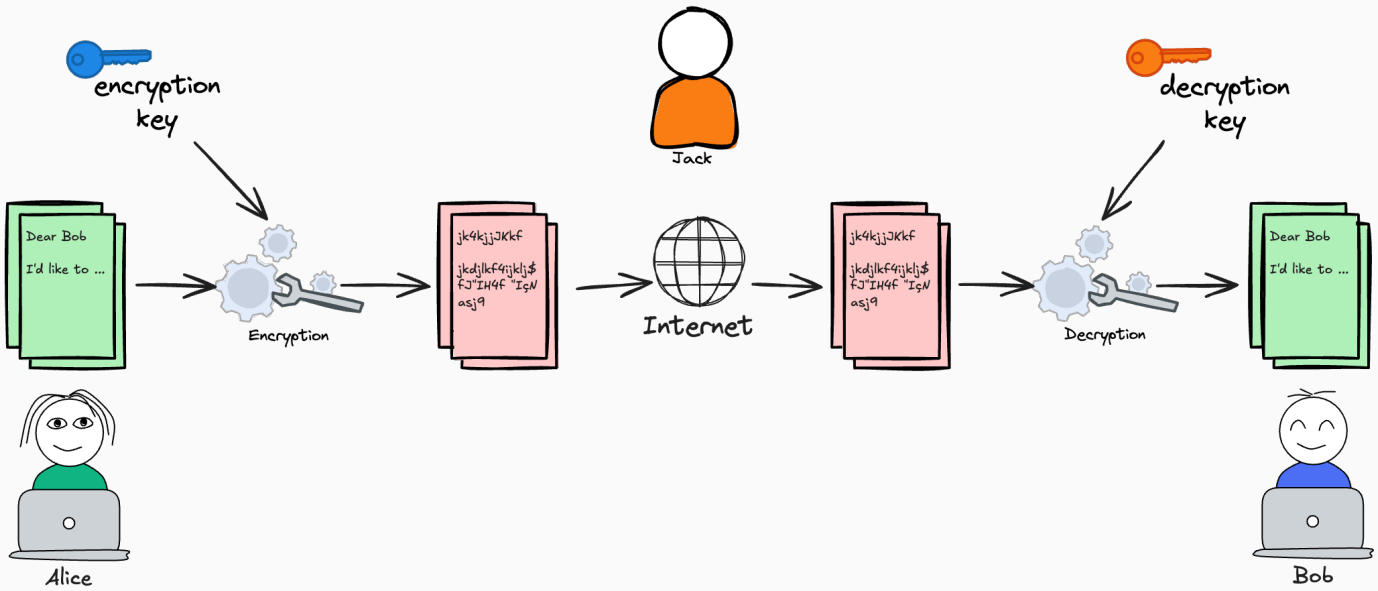

7) TPM
6) Firewall
5) FileSystem, LUKS, InitRamFS
U-boot
Kernel
Root-fs
User-fs
2) U-boot
3) Compile kernel
4) Hardening Linux
1) Buildroot

Summary


Applications:
- Authenticated code
- Privileges
- Compilation options
- Configurations
User space
Disk:
- Partition rights
- File access rights
Stack IP
(Restricted) open ports:
- TCP / UDP
- RS232
Kernel space

Asymmetric Cryptography - Recap

Asymmetric cipher principle



Ensure:

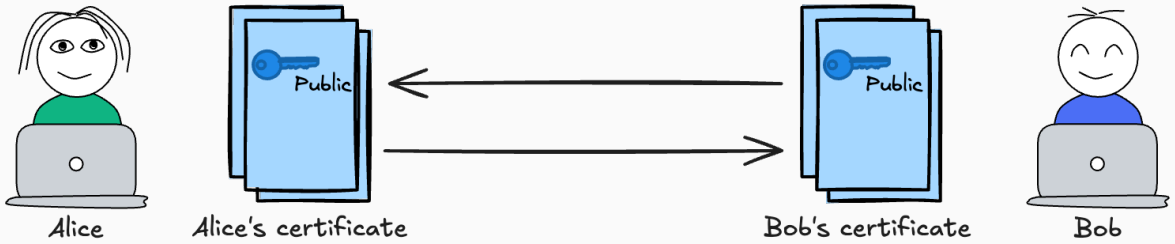
- Confidentiality
- Integrity
- Authentication

Algorithms:

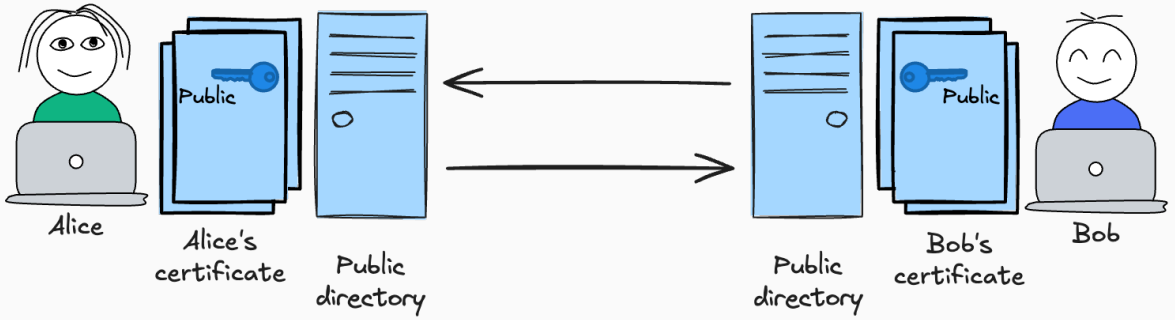
- RSA
- DSA: signature
- ECDH: key exchange

Asymmetric cipher

- Bob and Alice must exchange their public keys in a secure way
- Generally the public keys are stored in certificates



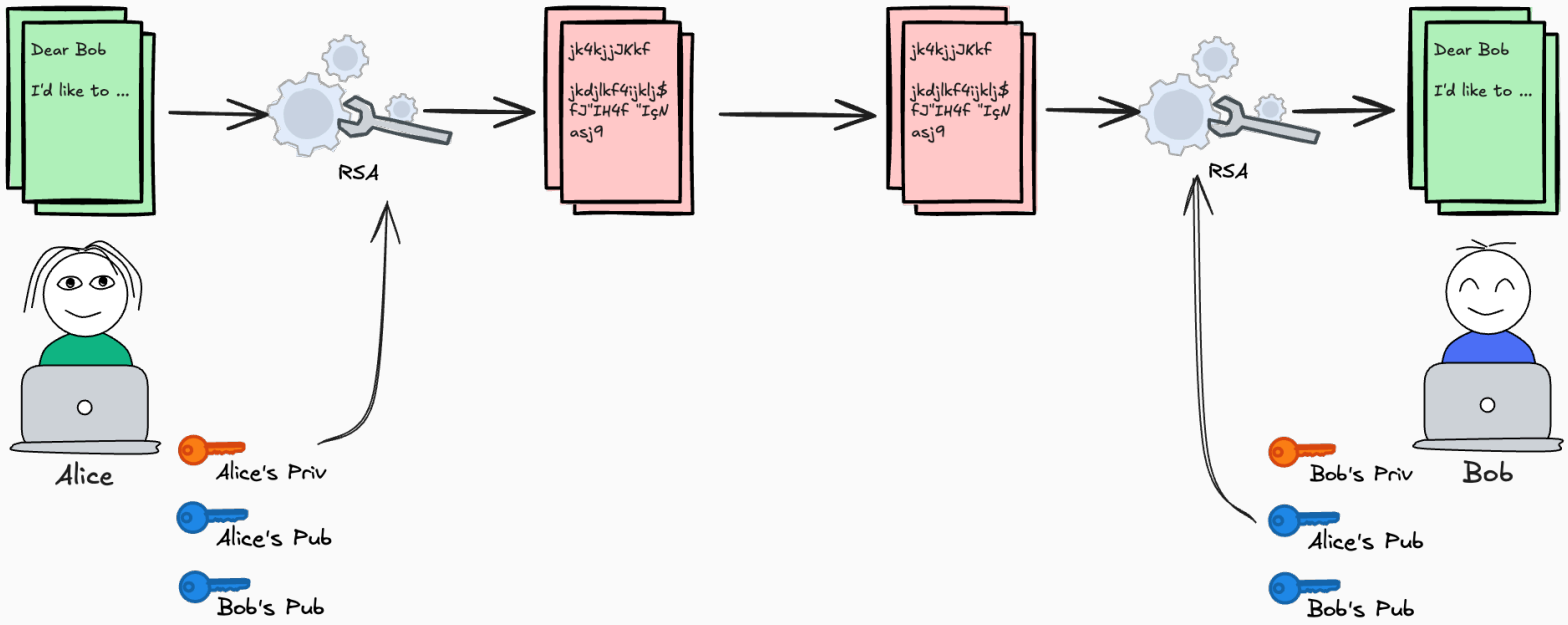
e.g. TLS



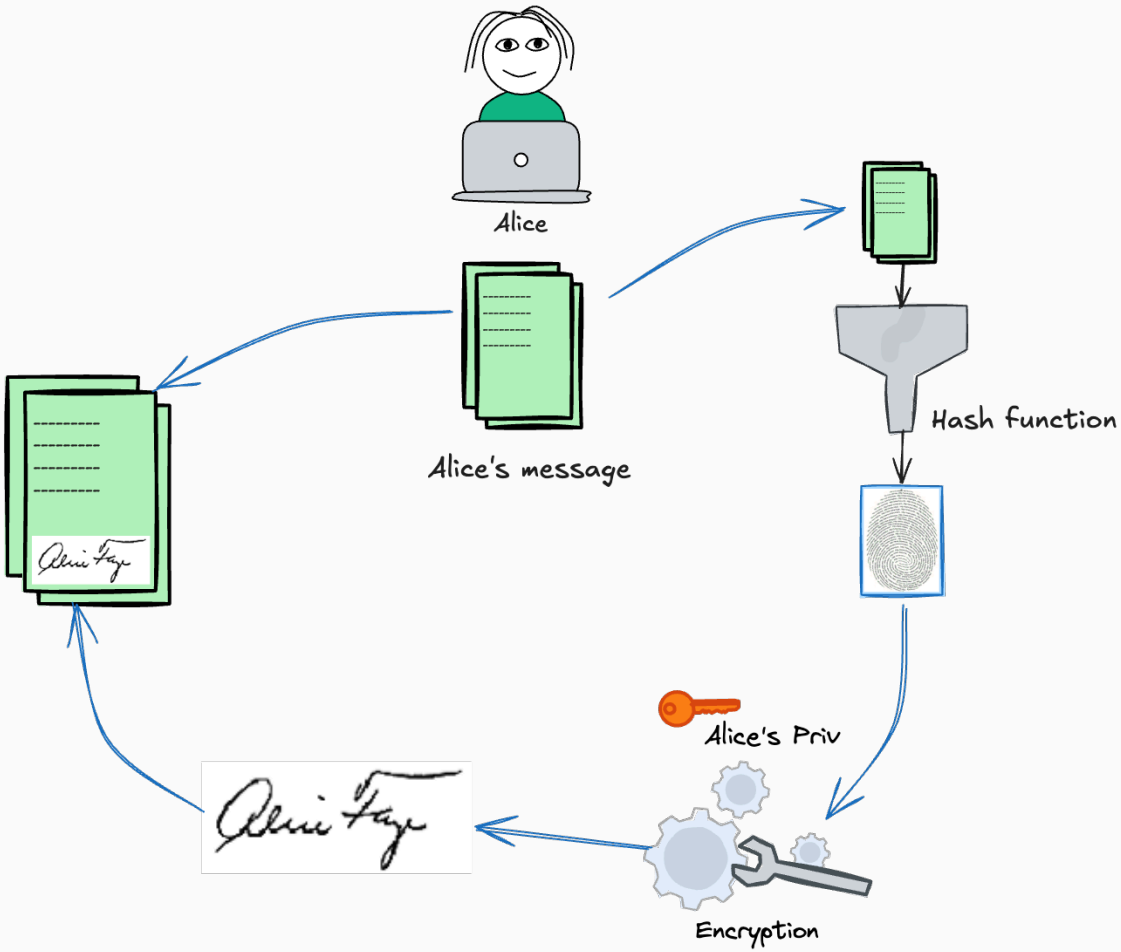
e.g. PGP

Asymmetric cipher - Signature

- Encrypt with the private key and decrypt with the public key
  - Authenticity, integrity



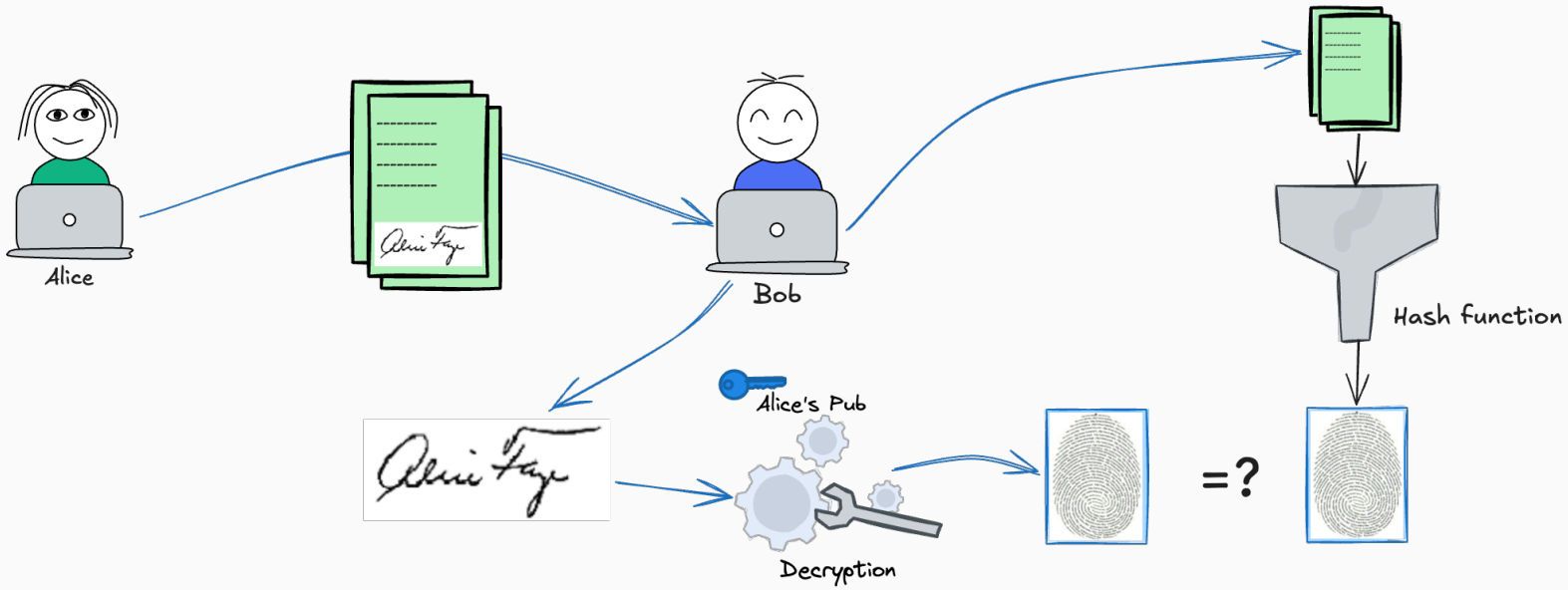
Digital signature



Digital signature

- In order to check a signature:
  - Bob uses Alice's public key
  - Bob computes the message fingerprint
  - Bob compares the two results

Digital signature



Securely install software from source

Securely download source code

Verify authenticity and integrity of the source code

There are mainly two possibilities to download source files

- Download from trusted sources (e.g. official website) and verify the checksum
- Download directly from Git repository

Example of openssh

1. Download *tared* file from official site: https://www.openssh.com → openssh-8.8p1.tar.gz
2. Download the source files with git: https://github.com/openssh/openssh-portable

Download and verify *tared* files (1/2)

- When a new package is downloaded for a site, it is required to check origin and integrity of the package. Often the packages are signed by the program **pgp** (Pretty Good Privacy, Linux command **gpg**)
- Example with CentOS: Download the package, the signature. Then these files should be downloaded (version may vary):
  - `CentOS-7.tar.gz` → CentOS-7 package
  - `CentOS-7.tar.gz.asc` → CentOS-7 package signature

Download and verify *tared* files (2/2)

Use **gpg** in order to check the package integrity

```
gpg --verify CentOS-7.tar.gz.asc    // check the signature
gpg: armor header: Version: GnuPG v2
gpg: assuming signed data in 'CentOS-7.tar.gz'
gpg: Signature made Sun 26 Sep 2021 16:07:27 CEST
gpg:                using RSA key 4898B983815A5EEF59A4ADFD2A3F414E73606945
gpg: using PGP trust model
gpg: Good signature
```

Import public key (if gpg don't know the public key yet):

```
gpg --keyserver keyserver.ubuntu.com --search-keys 4898B983815A5EEF59A4ADFD2A3F414E73606945
gpg --keyserver hkp://keys.openpgp.org --recv-keys 4898B983815A5EEF59A4ADFD2A3F414E73606945
```

List public keys:

```
gpg --list-keys
```

Download with git

```
git clone https://github...     // Download the package with different versions
git log                         // Show the different commits
git tag                         // Show the different versions
git checkout –b TAG_VERSION     // Create a new branch for the used branch
```

Read INSTALL

```
less INSTALL or less README     // Analyze the different options
```

If `configure` file don't exist, execute: `autoreconf`

```
./configure --help              // Get help about configuration options
```

Remark: Sometime commit are signed with pgp, in order to check the signature, you must configure git with the pgp keys, see https://git-scm.com/book/en/v2/Git-Tools-Signing-Your-Work or https://developers.yubico.com/PGP/Git_signing.html for additional help

Configure a new package

- When you install a new package, it is necessary to check and configure different options. And finally create a secure configuration.

Configure the package package1:

```
tar xvzf package1.tar.gz
cd package1
less INSTALL or less README     // Analyze the different options
./configure --help              // Analyze the different options
```

- It is very important to check the different **configuration options**, **compilation-link options** in order to improve the security and the robustness code.

Cross-compiling a new package

Configure package1 for cross compilation and change the default install directory

```
./configure --host=aarch64-linux-gnu --prefix=/root/install
                                                    (--prefix option is important)
make
make install
```

- `--host`: cross compiler name
- `--prefix`: change the directory installation of the executable, libraries, etc. For the cross compilation, it is required to **change the default directory in order not to overwrite the original files**.

Cross-compiling a new package

Sometime cross compilation is not easy to configure, and it is necessary to modify directly the `Makefile`

Example - MQTT broker: mosquitto

In order to modify the compilation options, it is necessary to modify the `config.mk` file (which is used by the Makefile)

Buildroot - Where should a new package be installed?

- Known buildroot packages are in the directory `/buildroot/packages`
- Example with a generic `foo` package, the directory `/buildroot/packages/foo` contains mainly these files:
  - **`config.in`** file, written in kconfig language, describing the configuration options for the package.
  - **`foo.mk`** file describing where to fetch the source, how to build and install it, etc.
  - Optional **`foo.hash`** file, providing hashes to check the integrity of the downloaded tarballs and license files.
  - **`Sxx_foo`** file, it is the start script for the foo package

Sources:

[1]: https://buildroot.org/docs.html

[2]: https://bootlin.com/doc/training/buildroot/buildroot-slides.pdf, p 125

Secure system configuration

Minimum services

Example for nanoPi

Startup scripts are in directory `/etc/init.d/`:

```
# ls /etc/init.d/
S01syslogd   S02sysctl    S20urandom   S40network   S50sshd      rcK
S02klogd     S10mdev      S21haveged   S50dropbear  rcS
```

- Script `rcS` starts other scripts (S01.., S02.., not `rcK`) in alphabetical order
- Minimum number of services must be started
- It is necessary to create a new script in order to start automatically a new service

Minimum services

- Delete a script $\longrightarrow$ service will not be started anymore (ex.: `rm S50sshd` $\longrightarrow$ sshd service will not be started at the next startup)

Service privileges

ps command shows the processes

```
# ps -ale        // Show all
process
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME       CMD
4 S     0     1     0  0  80   0 -   492 wait   ?        00:00:05       init
1 S     0     2     0  0  80   0 -     0 kthrea ?        00:00:00       kthreadd
1 S     0     3     2  0  80   0 -     0 smpboo ?        00:00:00       ksoftirqd/0
# ps -axjf      // Show all process
tree
 PPID   PID  PGID   SID TTY      TPGID STAT   UID   TIME                COMMAND
    0     2     0     0 ?           -1 S        0   0:00                [kthreadd]
    2     3     0     0 ?           -1 S        0   0:00                 \_ [ksoftirqd/0]
    2     4     0     0 ?           -1 S        0   0:00                 \_ [kworker/0:0]
    2     5     0     0 ?           -1 S<       0   0:00                 \_ [kworker/0:0H]
# ps aux    // Show privileges processes
root      1514  0.0  0.0   1968   520 ?        S    00:00   0:00        /sbin/syslogd -n
root      1517  0.0  0.0   1968   532 ?        S    00:00   0:00        /sbin/klogd -n
root      1557  0.0  0.0   4232   756 ?        Ss   00:00   0:00        /usr/sbin/sshd
# lsof | grep sshd     // Show open files and privileges
sshd      1557 root    3u     IPv4       7164      0t0    TCP *:ssh (LISTEN)
sshd      1557 root    4u     IPv6       7166      0t0    TCP *:ssh (LISTEN)
```

Check open ports

netstat command shows the open tcp and udp ports

```
# netstat -atun
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 :::22                   :::*                    LISTEN
```

lsof command shows open files

```
# lsof | grep IP
sshd      1557 root    3u     IPv4       7164      0t0   TCP *:ssh (LISTEN)
sshd      1557 root    4u     IPv6       7166      0t0   TCP *:ssh (LISTEN)
```

nmap command scans open ports for another hosts

```
nmap command scans open ports for another hosts
nmap -Pn -p 1-65535 192.168.0.11    // scan all tcp ports for the host   192.168.0.11
nmap -sU -Pn -p 1-65535 192.168.0.11  // scan all udp ports for the host   192.168.0.11
```

Network stack hardening

IPv4 protection (1/2)

- Disable IPv4 Forwarding: if the system is not as a router, `ip_forward` must be disabled:
  - `sysctl -w net.ipv4.ip_forward = 0`
- Enable Source Route Verification: if the system is as a router, `rp_filter` should be activated on all interfaces
  - `sysctl -w net.ipv4.conf.INT.rp_filter = 1`
- Disable IP Source Routing: `accept_source_route` must be disable
  - `sysctl -w net.ipv4.conf.INT.accept_source_route= 0`
    - where `INT=eth0, lo`, ...
- Disable ICMP Redirects: `accept_redirect` must be disable on all interfaces
  - `sysctl -w net.ipv4.conf.INT.accept_redirects = 0`
    - where `INT=eth0, lo`, ...

IPv4 protection (2/2)

- Ignore ICMP Echo Broadcasts: `icmp_echo_ignore_broadcasts` must be enabled
  - `sysctl -w net.ipv4.icmp_echo_ignore_broadcasts = 1`
- Ignore ICMP Bogus Error Responses: `icmp_ignore_bogus_error_responses` must be enabled
  - `sysctl -w net.ipv4.icmp_ignore_bogus_error_responses = 1`
- Enable Logging of Martians (packets from sources that aren't possible): `log_martians` should be enabled on all interfaces
  - `sysctl -w net.ipv4.conf.INT.log_martians=1`
- Enable TCP SYN Cookie: `tcp_syncookies` must be enabled
  - `sysctl -w net.ipv4.tcp_syncookies=1`

More information: `<kernel source>/Documentation/networking/ip-sysctl.txt`

or

https://dev.gentoo.org/~swift/docs/security_benchmarks/kernel.html#item-gt-sysctl-ipv4forward

TCP SYN cookie protection (1/2)

Normal TCP/IP networking is open to an attack known as a SYN flooding. In a SYN flood attack, an attacker sends a succession of SYN requests to a target's system in an attempt to consume enough server resources to make the system unresponsive to legitimate traffic.

https://www.cyberciti.biz/faq/enable-tcp-syn-cookie-protection/

TCP SYN cookie protection (2/2)

This denial-of-service attack uses half-open tcp connections.

The CONFIG_SYN_COOKIE can avoid tis attack

`make linux-xconfig: Networking support → Networking options → TCP IP Networking → IP TCP Sys Cookies [*]`

Read syn cookies configuration:

```
cat /proc/sys/net/ipv4/tcp_syncookies  // 0= deactivated, 1=activated
sysctl -n net.ipv4.tcp_syncookies]
```

Hardening Linux: Activate sys cookies

```
echo 1 > cat /proc/sys/net/ipv4/tcp_syncookies
sysctl net.ipv4.tcp_syncookies=1
```

Configure kernel parameters

`sysctl -p` command reads the file `/etc/sysctl.conf` and configure kernel parameters

Example:

```
cat /etc/sysctl.conf
kernel.randomize_va_space=2
net.ipv4.conf.lo.rp_filter = 1
net.ipv4.conf.eth0.rp_filter = 1
net.ipv4.conf.lo.accept_source_route= 0
net.ipv4.conf.eth0.accept_source_route= 0
net.ipv4.conf.lo.accept_redirects=0
net.ipv4.conf.eth0.accept_redirects=0
net.ipv4.icmp_echo_ignore_broadcasts = 1
net.ipv4.icmp_ignore_bogus_error_responses = 1
net.ipv4.conf.lo.log_martians=1
net.ipv4.conf.eth0.log_martians=1
net.ipv4.tcp_syncookies=1
```

Network stack hardening (1/5)

- Several network options are enabled by default and offer too much information about the system to an attacker or may lead to system failure under a heavy network load
- It is possible to tune some parameters in order to increase the security of the network stack
- The details of these options are in the file `linux-source-code/Documentation/networking/ip-sysctl.txt`
- Disable IPv6

```
    echo 1 > /proc/sys/net/ipv6/conf/all/disable_ipv6
    echo 1 > /proc/sys/net/ipv6/conf/default/disable_ipv6
or  sysctl -w /net/ipv6/conf/default/disable_ipv6=1
```

- IP source routing must be disabled

```
    echo 0 > /proc/sys/net/ipv4/conf/all/accept_source_route
or  sysctl -w net/ipv4/conf/all/accept_source_route=0
```

Network stack hardening (2/5)

- Forwarding (routing) of normal and multicast packets should also be deactivated unless expressively needed

```
    echo 0 > /proc/sys/net/ipv4/conf/all/forwarding
    echo 0 > /proc/sys/net/ipv6/conf/all/forwarding
    echo 0 > /proc/sys/net/ipv4/conf/all/mc_forwarding
    echo 0 > /proc/sys/net/ipv6/conf/all/mc_forwarding
or
    sysctl -w net/ipv4/conf/all/forwarding=0
    sysctl -w net/ipv6/conf/all/forwarding=0
    sysctl -w net/ipv4/conf/all/mc_forwarding=0
    sysctl -w net/ipv6/conf/all/mc_forwarding=0
```

Network stack hardening (3/5)

- Block ICMP redirect messages

```
    echo 0 > /proc/sys/net/ipv4/conf/all/accept_redirects
    echo 0 > /proc/sys/net/ipv6/conf/all/accept_redirects
    echo 0 > /proc/sys/net/ipv4/conf/all/secure_redirects
    echo 0 > /proc/sys/net/ipv4/conf/all/send_redirects
or
    sysctl -w net/ipv4/conf/all/accept_redirects=0
    sysctl -w net/ipv6/conf/all/accept_redirects=0
    sysctl -w net/ipv4/conf/all/secure_redirects=0
    sysctl -w net/ipv4/conf/all/send_redirects=0
```

- Enable source route verification in order to prevent IP spoofing

```
    echo 1 > /proc/sys/net/ipv4/conf/default/rp_filter
or
    sysctl -w net/ipv4/conf/default/rp_filter=1
```

Network stack hardening (4/5)

- Log all malformed packed and ignore icmp bogus ones

```
    echo 1 > /proc/sys/net/ipv4/conf/all/log_martians
    echo 1 > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
or
    sysctl -w net/ipv4/conf/all/log_martians=1
    sysctl -w net/ipv4/icmp_ignore_bogus_error_responses=1
```

- Disable ICMP echo and timestamp responses sent via broadcast or multicast

```
    echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
or
    sysctl -w net/ipv4/icmp_echo_ignore_broadcasts=1
```

- Increase resilience under heavy TCP load by increasing backlog buffer and enabling syn cookies:

```
    echo 4096 > /proc/sys/net/ipv4/tcp_max_syn_backlog
    echo 1 > /proc/sys/net/ipv4/tcp_syncookies
or
    sysctl -w net/ipv4/tcp_max_syn_backlog=4096
    sysctl -w net/ipv4/tcp_syncookies=1
```

Network stack hardening (5/5)

- `sysctl -p` command reads the file `/etc/sysctl.conf` and configure kernel parameters

Example:

```
cat /etc/sysctl.conf
  net/ipv6/conf/default/disable_ipv6=1
  net/ipv4/conf/all/accept_source_route=0
  net/ipv4/conf/all/forwarding=0
  net/ipv6/conf/all/forwarding=0
  net/ipv4/conf/all/mc_forwarding=0
  net/ipv6/conf/all/mc_forwarding=0
  net/ipv4/conf/all/accept_redirects=0
  net/ipv6/conf/all/accept_redirects=0
  net/ipv4/conf/all/secure_redirects=0
  net/ipv4/conf/all/send_redirects=0
  net/ipv4/conf/default/rp_filter=0
  net/ipv4/conf/all/log_martians=1
  net/ipv4/icmp_ignore_bogus_error_responses=1
  net/ipv4/icmp_echo_ignore_broadcasts=1
  net/ipv4/tcp_max_syn_backlog=4096
  net/ipv4/tcp_syncookies=1
```

Account

- User passwords must be saved in the `/etc/shadow` file (the password is encrypted)
- Insert in the file /etc/issue a warning banner before the login:

WARNING: Accessing a protected computer system without authorization constitutes a criminal of- fence and may be punished by imprisonment (Swiss penal Law StGB Art. 143bis). Unauthorized access to this system is logged and will be reported to the relevant authorities for prosecution under penal law. The right for compensation of damages under civil law is reserved. If you are not authorized to access this system, disconnect now.

Account

- The umask (user file creation right) must be `0027` (and not the default = `0022`)

```
# umask 0027
```

Or insert the `umask` command in the `$HOME/.bash_profile` file or `/etc/profile`

- umask rules:

```
umask = user file-creation rights
[root] #umask              // default umask 0022

Default umask 0022     User Group Other
Creation rights        111   111    111   //example
Umask                  000   010    010   //umask = 0022
Not umask              111   101    101   //= 755
Rights=777 & 755:      111   101    101 = 755

Secure umask 0027
Creation rights        111   111    111   example
Umask                  000   010    111
Not umask              111   101    000 = 750
Rights=777 & 750:      111   101    000 = 750
```

Root login

- Direct root logins must be restricted to the console (emergency situations usually require hands at the console). This can be achieved by modifying the `/etc/securetty` file.
  - Example for NanoPi: `echo ttyS0 > /etc/securetty`
- Only root can access to the root directory: `chmod 700 /root`
- Use `su` or `sudo` commands in order to have the root rights
- The root PATH must not contain the current directory or writable directories (`PATH not equal .:/tmp:/var/tmp:` etc...)

Kernel hardening (1/2)

- ASLR (Adresse Space Layout Randomization) ... already seen in a lecture before
- Write protect kernel text section, kernel configuration:

```
make linux-xconfig:
    Kernel Feature -->  [*] Apply r/o permissions of VM areas also to their linear aliases
    General Setup -->   [*] Set loadable kernel module data as NX and text as RO
```

- Enable -fstack-protector buffer overflow detection , kernel configuration

```
make linux-menuconfig:
   General architecture-dependent options -->
           [*] Stack Protector buffer overflow detection
           [*] Strong Stack Protector
```

Kernel hardening (2/2)

- Strip assembler-generated symbols during link, kernel configuration

```
make linux-menuconfig:
    Kernel Hacking --> Compile time check and compiler options -->
           [ ] Compile the kernel with debug info
           [*] Strip assembler-generated symbols during link
```

- Only root can access to the kernel system logs (through `dmesg`)

```
make linux-menuconfig:
   Security options -->
           [*] Restrict unprivileged access to the kernel syslog
```

end of presentation