Ecole d'ingénieurs et d'architectes de Fribourg
Hochschule für Technik und Architektur Freiburg

Linux Hardening

Youtube: https://youtu.be/rYC3fmq43lA

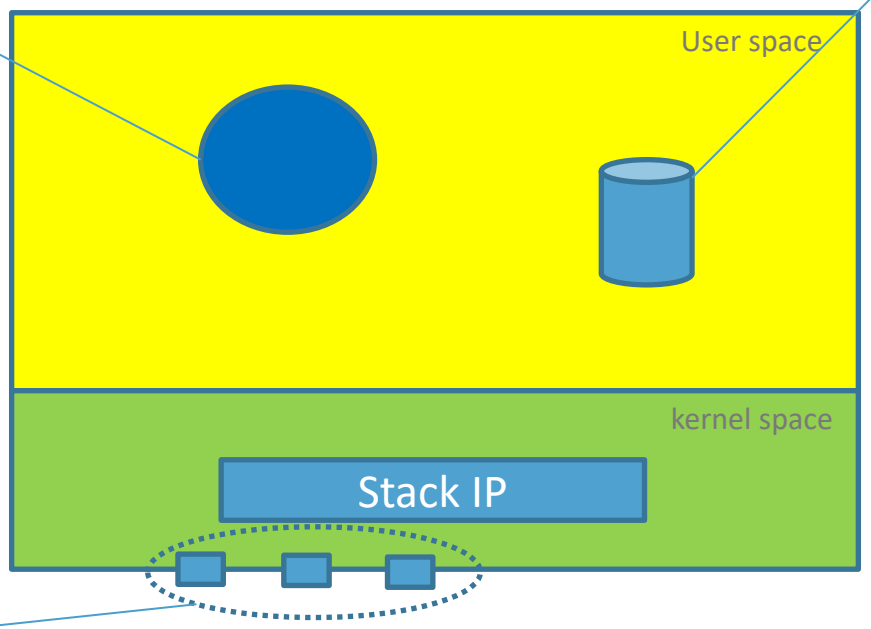
Hes·so FRIBOURG FREIBURG
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz

Secure Embedded System, jean-roland.schuler@hefr.ch

Summary

Applications:

- Authenticated code
- Privileges
- Compilation options
- Configuration

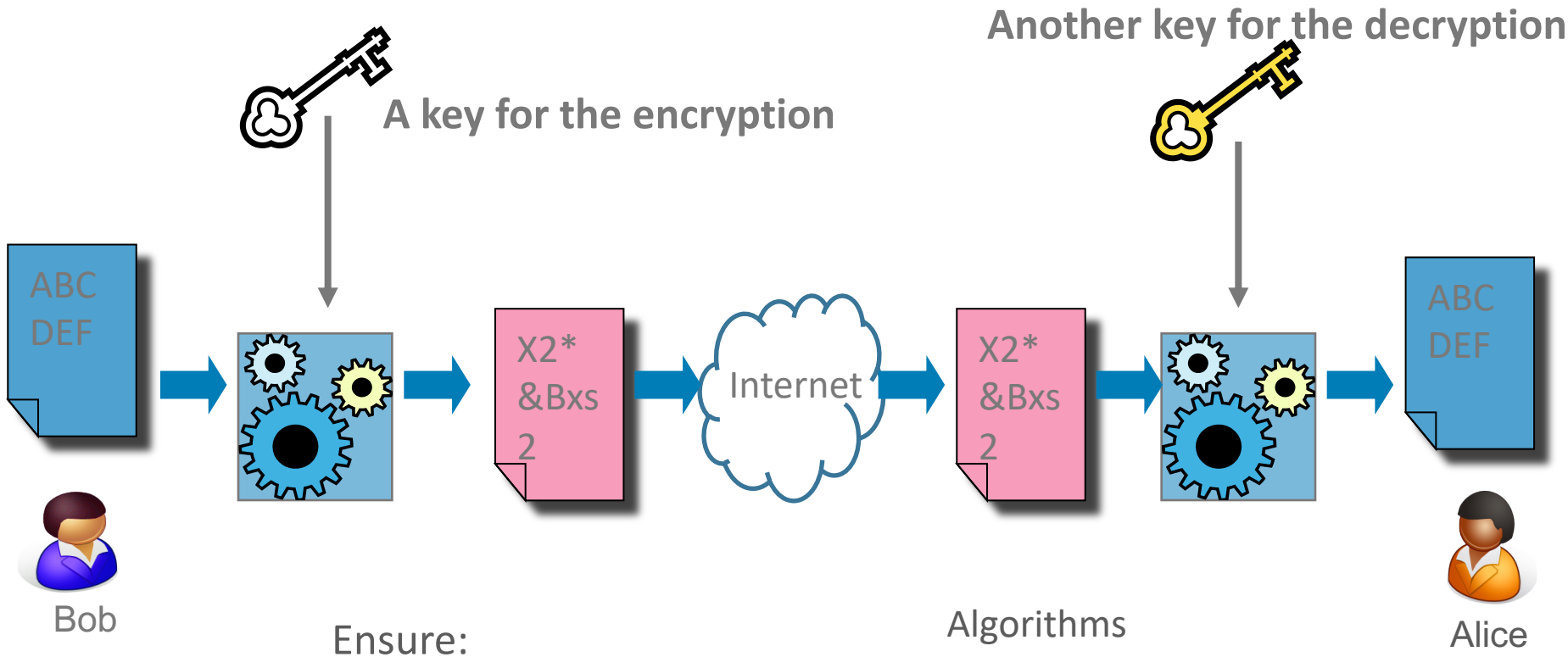


Disk:

- Partition rights
- Files rights

Restrict the opened ports
(TCP-UDP, rs232,…)

Asymmetric cipher principle

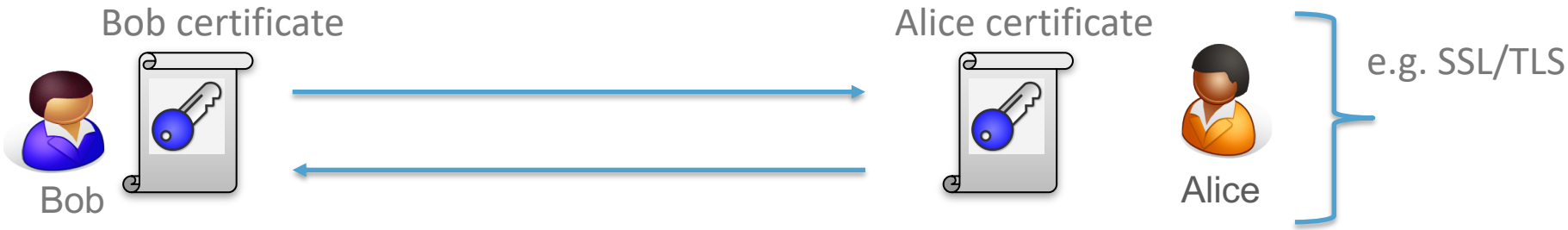


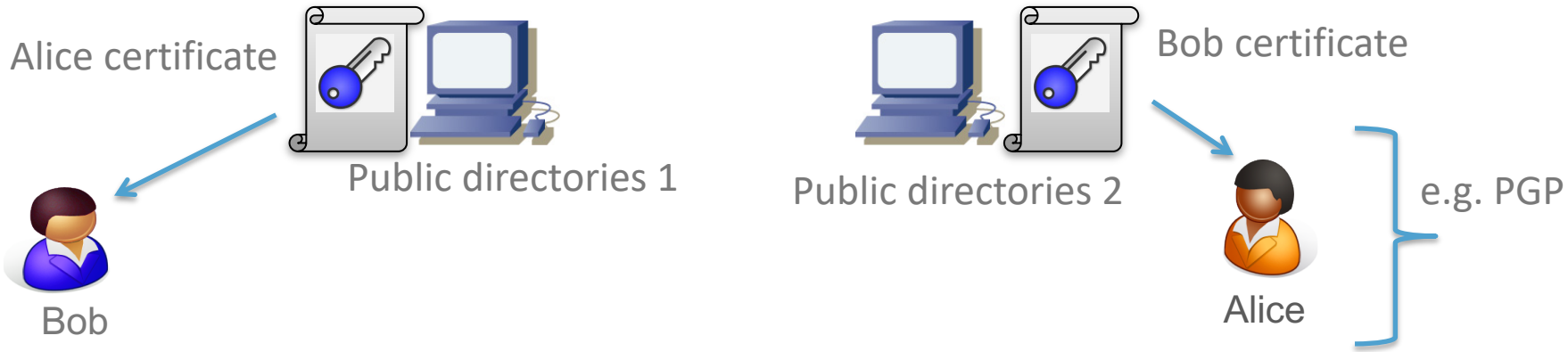
Ensure:
- The confidentiality
- The integrity
- The authentication

Algorithms
- RSA
- DSS or DAS : signatures
- Diffie-Hellmann : key exchange

Asymmetric cipher

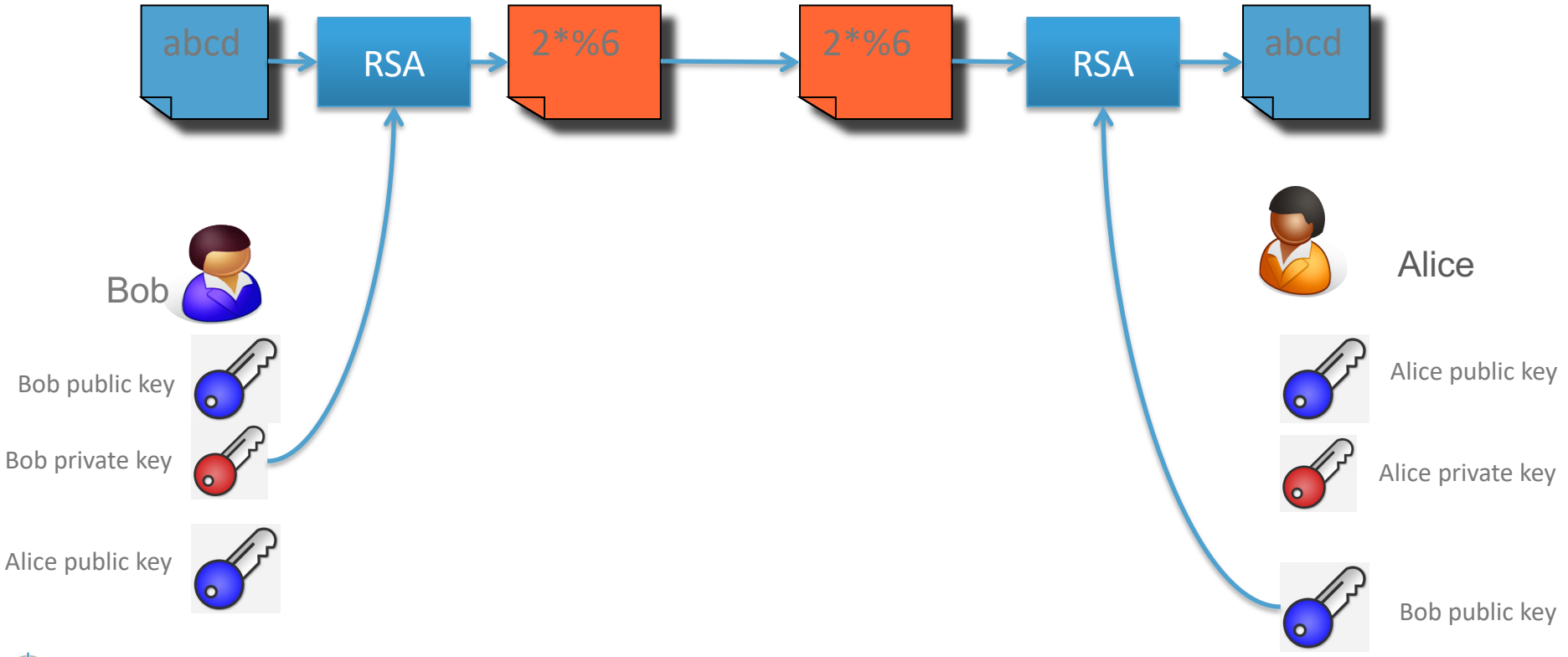
- Bob and Alice must exchange their public keys
- Generally the public keys are stored in a certificate



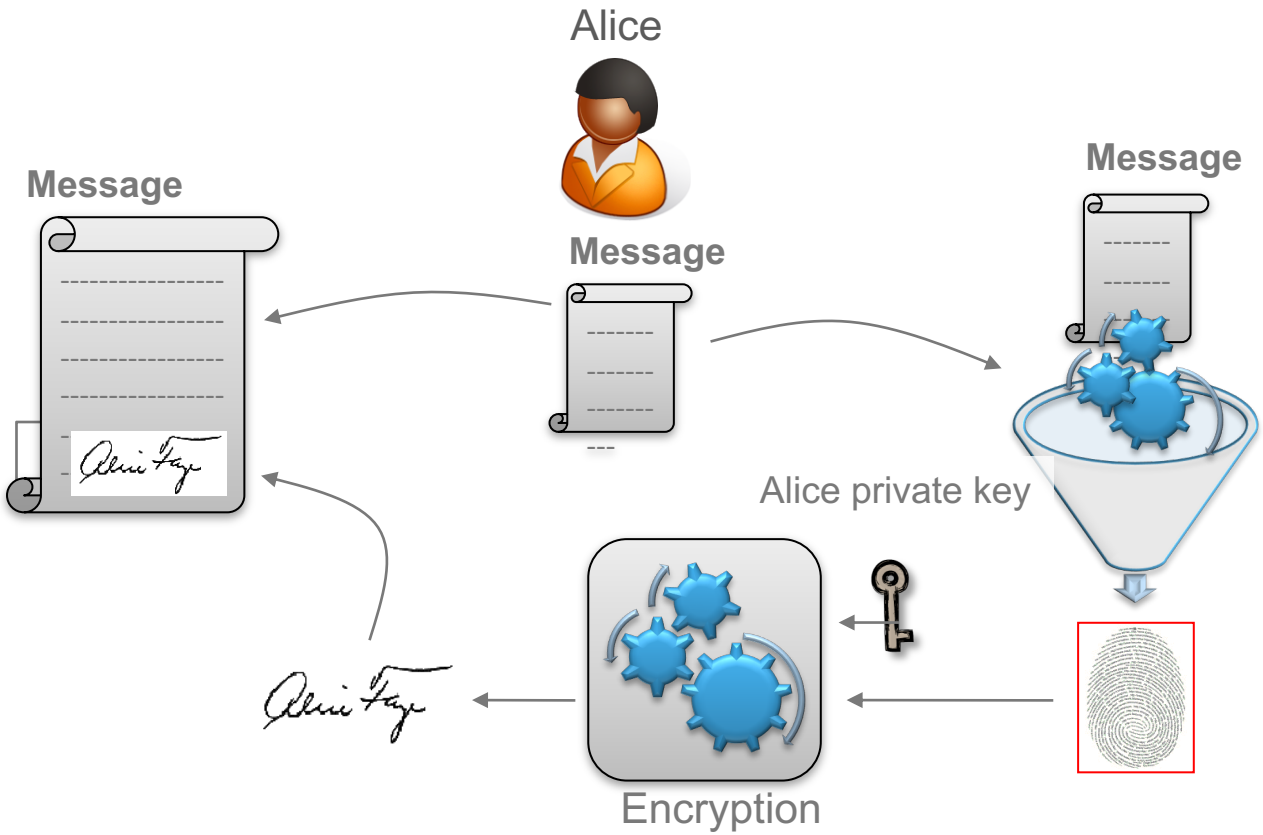


Asymmetric cipher: signature

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

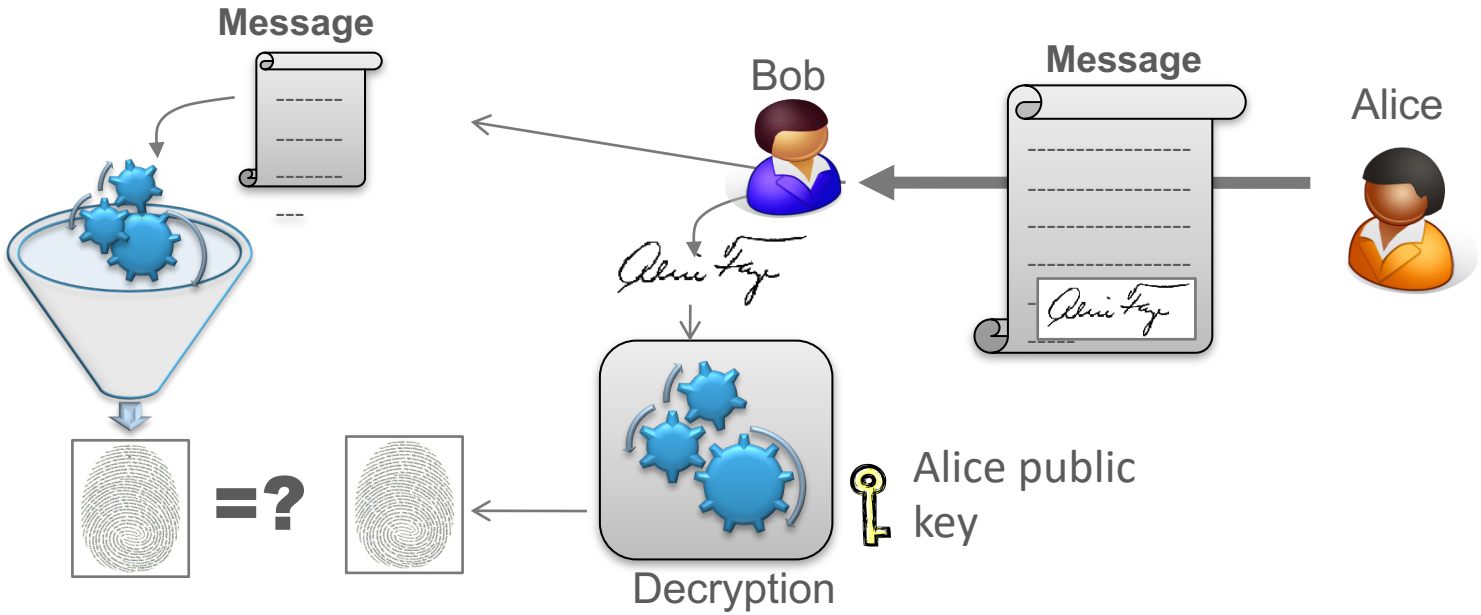


Digital signature



Digital signature

- In order to check a signature:
  - Bob uses the Alice public key
  - Bob computes the message finperprint
  - Bob compares the two results.



Check a new package

When a new package is downloaded, 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
On the PC, these files are downloaded in the same directory

- CentOS-7.tar.gz — CentOS-7 package
- CentOS-7.tar.gz.asc — CentOS-7 package signature

- 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 Fri 21 Aug 2018 07:10:03 AM CEST using RSA key ID 6D920D30
  gpg: using PGP trust model
  gpg: Good signature
```

If gpg don't known public key

- Import public key:

  Command `gpg --verify CentOS-7.tar.gz.asc` indicates a key ID number
  Import the key:
  ```
  gpg --keyserver hkp://keys.gnupg.net --search-keys 6D920D30
  gpg --keyserver hkp://keys.gnupg.net --recv-keys 6D920D30
  ```

- List public keys:

  gpg --list-keys

Configure a new package

When you install a new package, it is necessary to check and configure different options.

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-compile a new package

Configure package1 for cross compilation and change the default install directory

```
./configure --host=aarch64-none-linux-gnu --prefix=/home/lmi/install
                                                (--prefix option is important)
make
make install
```

--host: cross compiler name

-- prefix: change the directory installation of the executable, libraries, …

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

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

Configure a new package

```
OpenSSH has been configured with the following options:
OpenSSH has been configured with the following options:
     User binaries: /home/schuler/work/1_nanopi_labo/master/5_openssh/install_arm/bin
     System binaries: /home/schuler/work/1_nanopi_labo/master/5_openssh/install_arm/sbin
     Configuration files: /home/schuler/work/1_nanopi_labo/master/5_openssh/install_arm/etcAskpass
program: /home/schuler/work/1_nanopi_labo/master/5_openssh/install_arm/libexec/ssh-askpass
Manual pages: /home/schuler/work/1_nanopi_labo/master/5_openssh/install_arm/share/man/manX
PID file: /var/run
Privilege separation chroot path: /var/empty
sshd default user PATH:
/usr/bin:/bin:/usr/sbin:/sbin:/home/schuler/work/1_nanopi_labo/master/5_openssh/install_arm/bin
                    Manpage format: doc
                       PAM support: no
                   OSF SIA support: no
                    ….

               Host: aarch64-none-linux-gnu
           Compiler: aarch64-none-linux-gnu-gcc
    Compiler flags: -g -O2 -pipe -Wno-error=format-truncation -Wall -Wextra -Wpointer-arith -
Wuninitialized -Wsign-compare -Wformat-security -Wsizeof-pointer-memaccess -Wno-pointer-sign -Wno-
unused-parameter -Wno-unused-result -Wimplicit-fallthrough -fno-strict-aliasing –

D_FORTIFY_SOURCE=2 -ftrapv -fno-builtin-memset -fstack-protector-strong -fPIE

Preprocessor flags:  -D_XOPEN_SOURCE=600 -D_BSD_SOURCE -D_DEFAULT_SOURCE
      Linker flags:  -Wl,-z,relro -Wl,-z,now -Wl,-z,noexecstack -fstack-protector-strong -pie
         Libraries: -lcrypto -ldl -lutil -lz  -lcrypt -lresolv
```

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
- Delete a script → service will not be started (ex.: `rm S50sshd` → sshd service will not be started

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 -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
```

File system, partitions

- By separating file systems into various partitions, it is possible to fine tune permissions and functionalities.
- A journaling file system must be installed (e.g. ext3, ext4) and activated

- Areas where users have “write privileges” should be kept on their own partition.

- LVM (Logical Volume Manager) can be used when more than four partitions are required.
- Partitions are configured to the file /etc/fstab
- Example:

```
/dev/root       /              ext2    rw,noauto         0       1
proc            /proc          proc    defaults          0       0
devpts          /dev/pts       devpts  defaults,gid=5,mode=620,ptmxmode=0666   0     0
tmpfs           /dev/shm       tmpfs   mode=0777         0       0
tmpfs           /tmp           tmpfs   mode=1777         0       0
tmpfs           /run           tmpfs   mode=0755,nosuid,nodev  0       0
sysfs           /sys           sysfs   defaults          0       0
```

File Permissions

Every file or folder in Linux has access permissions. There are three types of permissions (the file or directory rights):

- read access (r)
- write access (w)
- execute access (x)

Permissions are defined for three types of users:

- the owner of the file
- the group(s) that the owner belongs to
- other users

Example:

```
-rw-rw-r--.  user1   grp1   292 Sep 27 17:55 Config.in
```

File Permissions

- Access to log directories (/var/log …) and syslog files for normal users must be denied.

- No world writable files and directories must exist, except for temporary directories: `/tmp, /usr/tmp, /var/tmp`.

```
# find / -type d -perm -0002 –print
```
// -0002: - is important

- Files in /etc must not have group and other bits writable.

```
# find /etc -type f -perm -0020 –print
# find /etc -type f -perm -0002 –print
```

- SUID or SGID binaries should be disabled

```
# find / -perm -4000 –print
# find / -perm -2000 –print
```

- It must be enforced that /tmp is cleaned during shutdown. If technically possible by the use of /tmp on a ramdisk.

Network stack hardening

- Several network options that are enabled by default 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 > cat /proc/sys/net/ipv4/conf/all/accept_source_route
Or sysctl –w net/ipv4/conf/all/accept_source_route=0
```

Network stack hardening

- 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

- 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 spoofing

```
echo 1 > /proc/sys/net/ipv4/conf/default/rp_filter
Or sysctl –w net/ipv4/conf/default/rp_filter=0
```

Network stack hardening

- 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 by 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

`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)

- 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
```

Account

- 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.

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

- ASLR (Adresse Space Layout Randomization) randomize_va_space: 1: shared libraries will be loaded to random addresses, echo 2 = echo1 plus heap randomization

```
make linux-menuconfig:
            General Setup → [ ] Disable Heap randomization
            Kernel feature → [*] Randomize the address of the kernel image
echo 1 > /proc/sys/kernel/randomize_va_space
echo 2 > /proc/sys/kernel/randomize_va_space
Or
sysctl -w kernel.randomize_va_space=1
sysctl -w kernel.randomize_va_space=2
```

- 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
```

Kernel hardening

- 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
```

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

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

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

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

Application hardening, 1

- It is possible to use options below in order to improve the security of an application during the compilation and link phases

- Makefile:

```
CFLAGS="-fPIE -fstack-protector-all -D\_FORTIFY\_SOURCE=2 -ftrapv"

LDFLAGS="-Wl,-z,now,-z,relro  -z,noexecstack, -pie "
```

- Command line

```
gcc –Wall -Wextra -z noexecstack -pie -fPIE -fstack-protector-all -
Wl,-z,relro,-z,now –O –D_FORTIFY_SOURCE=2 –ftrapv -o test test.c
```

Application hardening, 2: noexecstack

- Modern systems do not allow execution of instructions stored on the stack. These abrevations: NX –PaX- DEP, describe the same thing but for different constructors or processors

- **NX bit** means *No-eXecute*. An operating system and/or hardware which supports the NX bit may mark certain areas of memory as non-executable (stack, heap). The NX bit has different names:
  - Intel: XD bit (eXecute Disable)
  - AMD: Enhanced Virus Protection
  - ARM: XN (eXecute Never)

- The PaX technology can emulate a NX bit (if the NX bit does not exist, ex.: x86 32 bit)
- On Microsoft Windows, the NX bit is called DEP (Data Execution Prevention)

- The `execstack` option can be used with the Linker flags:

```
LDFLAGS: -z execstack : allow executable stack, disable the NX protection
         -z noexecstack : not allow executable stack, enable the NX protection
```

Application hardening, 2: noexecstack

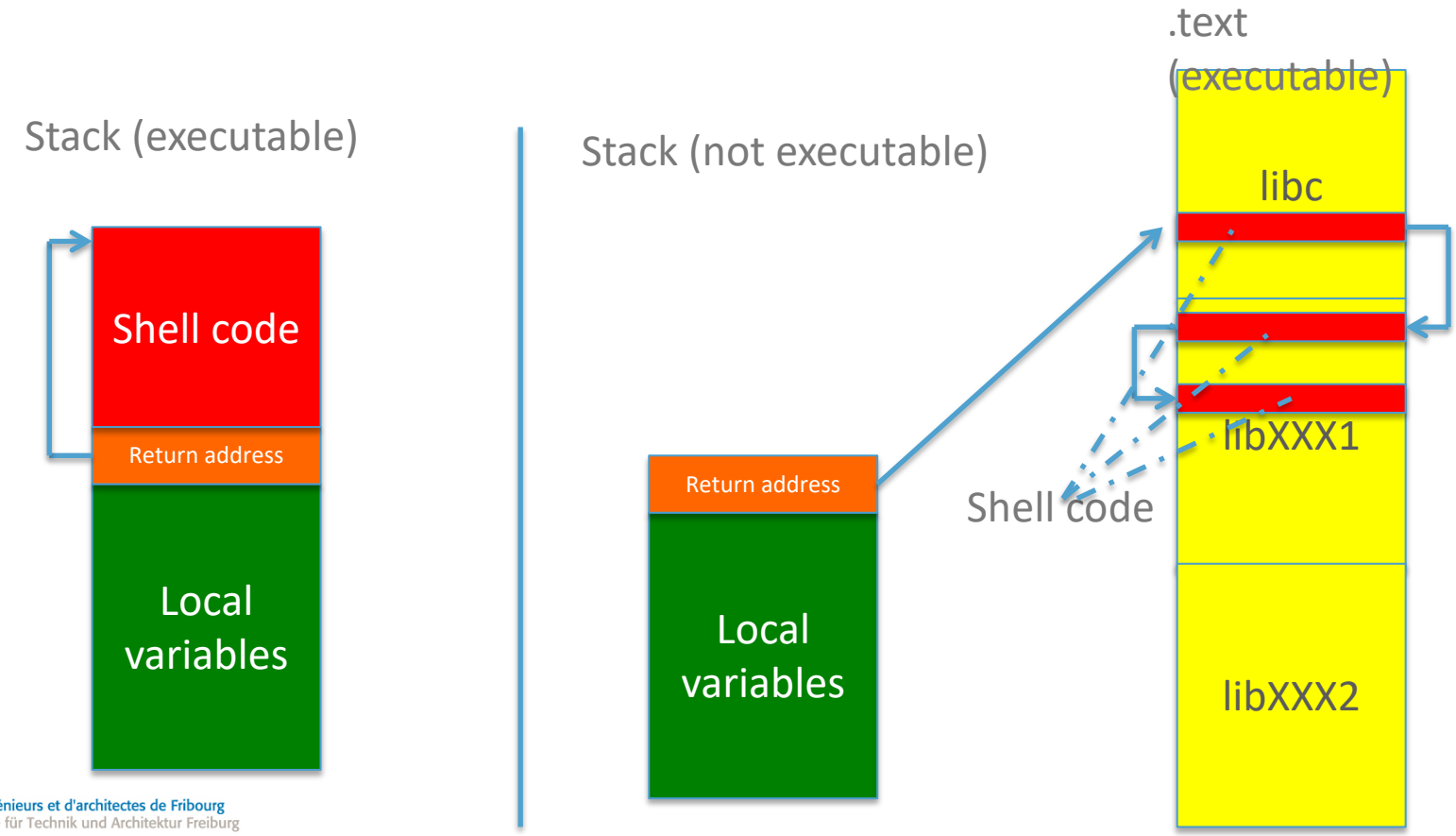
- Example: complile and link the test.c file

```
gcc -Wall -z noexecstack -o test test.c
readelf -l test
    Program Headers:
      Type           Offset             VirtAddr           PhysAddr
                     FileSiz            MemSiz              Flags  Align
      PHDR           0x0000000000000040 0x0000000000400040 0x0000000000400040
                     0x00000000000001f8 0x00000000000001f8  R E    8
      INTERP         0x0000000000000238 0x0000000000400238 0x0000000000400238
                     0x000000000000001c 0x000000000000001c  R      1
          [Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]
      LOAD           0x0000000000000000 0x0000000000400000 0x0000000000400000
                     0x0000000000000714 0x0000000000000714  R E    200000
      LOAD           0x0000000000000e10 0x0000000000600e10 0x0000000000600e10
                     0x000000000000021c 0x0000000000000220  RW     200000
      DYNAMIC        0x0000000000000e28 0x0000000000600e28 0x0000000000600e28
                     0x00000000000001d0 0x00000000000001d0  RW     8
      NOTE           0x0000000000000254 0x0000000000400254 0x0000000000400254
                     0x0000000000000044 0x0000000000000044  R      4
      GNU_EH_FRAME   0x00000000000005c0 0x00000000004005c0 0x00000000004005c0
                     0x000000000000003c 0x000000000000003c  R      4
      GNU_STACK      0x0000000000000000 0x0000000000000000 0x0000000000000000
                     0x0000000000000000 0x0000000000000000  RW     10
      GNU_RELRO      0x0000000000000e10 0x0000000000600e10 0x0000000000600e10
                     0x00000000000001f0 0x00000000000001f0  R      1
```

The stack segment is only RW

Application hardening, 2: noexecstack

- However, a technique called ret2libc could be used to bypass non executable memory. The main idea is to execute code in an executable memory like libc() or other libraries.
- Another technique called ROP, or Return-Oriented Programming allows also to bypass this protection. The main idea is to execute code in the program itself
- ASLR (Address Space Layout Randomization) randomize_va_space is used in order avoid the ret2lib (because stack and head addresses change)
- The PIE (Position Independent Executable) avoids the ROP problem (because code addresses change)



Application hardening, 3: pie

- Pie: Position Independent Executable
- Functions and variables addresses of an executable are not static and are computed during the program execution → this option avoids the ROP problem
- The load of this program is slower that another program with static addresses
- Example: test.c file

```
int b;
int func(int a) {
    b=a;
    return (0);
}
int main (void) {
int a=1;
    a = func (a);
    b = 10;
    return (0);
}
```

```
gcc –Wall –pie –fPIE –o test test.c
// -pie:used by the linker, -fPIE used by the compiler
```

Application hardening, 3: pie

```
readelf -a test
  ELF Header:
    Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
    Class:                             ELF64
    Data:                              2's complement, little endian
    Version:                           1 (current)
    OS/ABI:                            UNIX - System V
    ABI Version:                       0
    Type:                              DYN (Shared object file)
    Machine:                           Advanced Micro Devices X86-64
    Version:                           0x1
    Entry point address:               0x620
```

Not the real excution addresses

```
objdump –d test
   0000000000000738 <main>:
    74c:   e8 17 00 00 00          callq  768 <func>
    751:   89 45 fc                mov    %eax,-0x4(%rbp)
    754:   48 8d 05 dd 08 20 00    lea    0x2008dd(%rip),%rax
    75b:   c7 00 0a 00 00 00       movl   $0xa,(%rax)
```

Application hardening, 3: pie

Same example but without pie and fPIE options

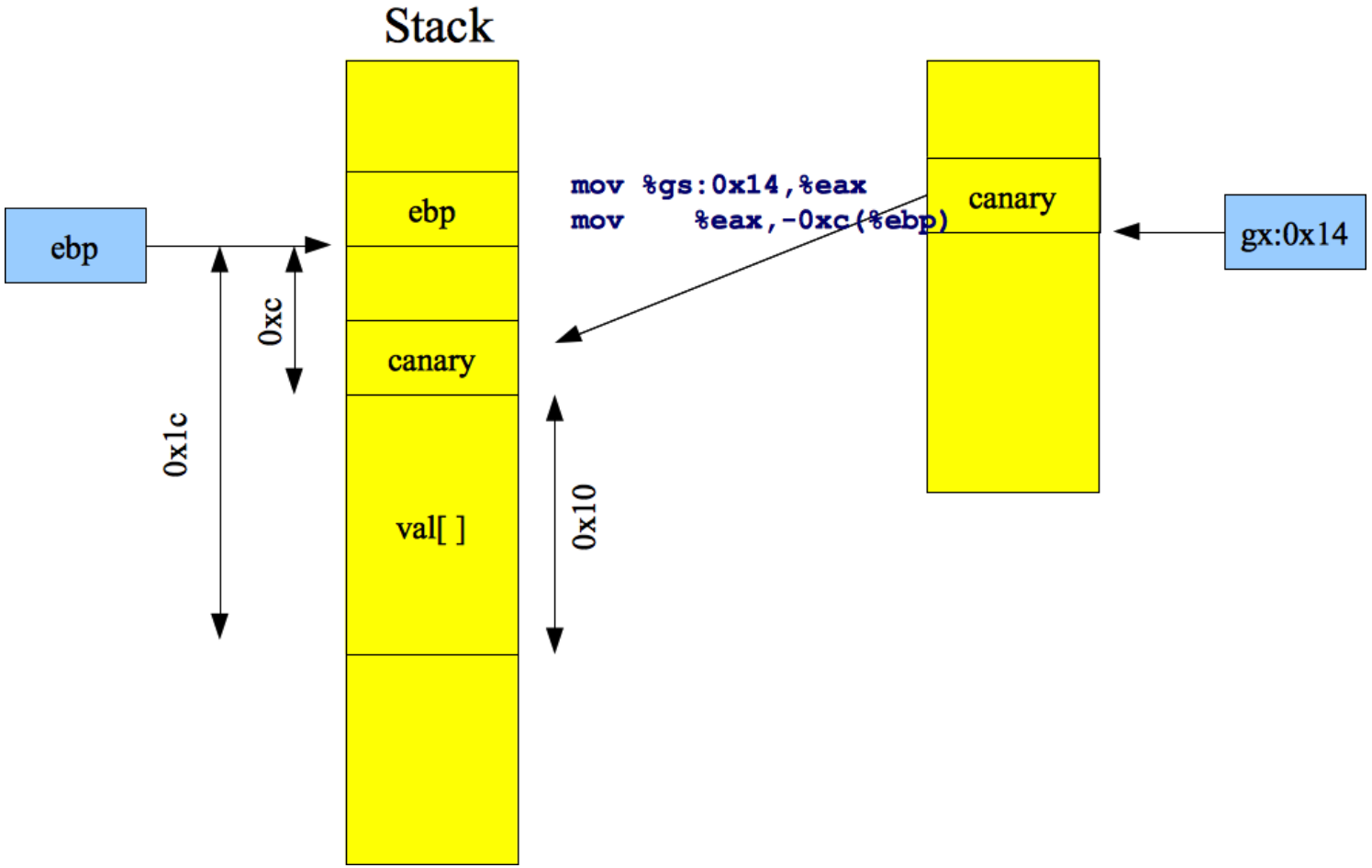
```
readelf -a test
  ELF Header:
    Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
    Class:                             ELF64
    Data:                              2's complement, little endian
    Version:                           1 (current)
    OS/ABI:                            UNIX - System V
    ABI Version:                       0
    Type:                              EXEC (Executable file)
    Machine:                           Advanced Micro Devices X86-64
    Version:                           0x1
    Entry point address:               0x400400
```

Real excution addresses

```
objdump –d test
   00000000004004f0 <main>:
    400504:       e8 14 00 00 00          callq  40051d <func>
    400509:       89 45 fc                mov    %eax,-0x4(%rbp)
    40050c:       c7 05 1a 0b 20 00 0a    movl   $0xa,0x200b1a(%rip) # 601030 <__TMC_END__>
```

Application hardening, 4: stack-protector-all

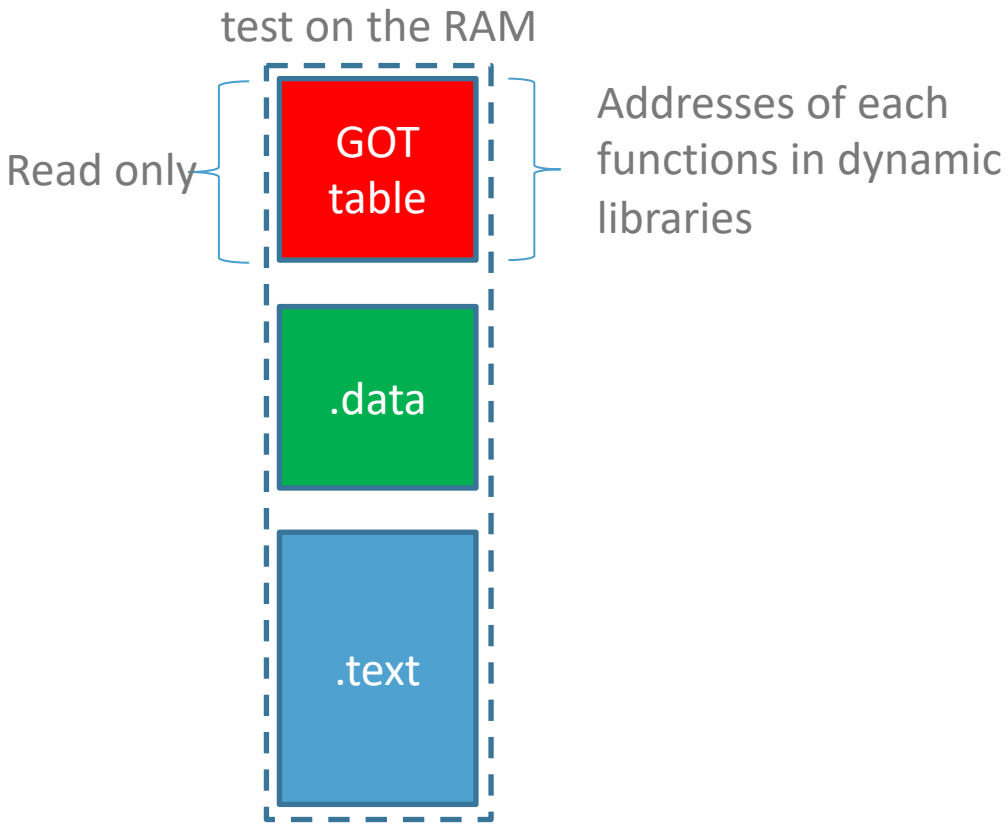
- Stack smashing protection is a way to protect programs from stack buffer overflows by adding random values (canaries) between the function's local variables and the saved instruction pointer.



Application hardening, 5: relro, now

- All dynamic symbols must be resolved during the program startup → the GOT (Global Offset Table) table is initialized and next it is marked read-only.
- This prevents GOT overwrite attacks.

```
gcc –Wall  -Wl,-z,relro,-z,now -o test test.c
```



Application hardening, 6: FORTIFY_SOURCE

- FORTIFY_SOURCE macros provide buffer overflow checks for the following functions:

```
memcpy, mempcpy, memmove, memset, strcpy, stpcpy, strncpy,
strcat,
strncat, sprintf, vsprintf, snprintf, vsnprintf, gets.
```

- Example: test.c file

```
int main(int argc, char **argv) {
char buffer[4];
    strcpy(buffer,argv[1]);
    return (0);
}
```

```
gcc -Wall -O -D_FORTIFY_SOURCE=1 -o test test.c
```

```
./test dddd
   *** buffer overflow detected ***: ./test terminated
   ======= Backtrace: =========
   /lib64/libc.so.6(__fortify_fail+0x37)[0x3dc090d4f7]
   /lib64/libc.so.6[0x3dc090b6b0]
```

Application hardening, 6: FORTIFY_SOURCE

- If _FORTIFY_SOURCE is set to 1, with compiler optimization level 1 (gcc -O1) and above, checks that shouldn't change the behavior of conforming programs are performed.

- With _FORTIFY_SOURCE set to 2 some more checking is added, but some conforming programs might fail.

- Some of the checks can be performed at compile time, and result in compiler warnings; other checks take place at run time, and result in a run-time error if the check fails.

Application hardening, 7: ftrapv

Integer overflow:

- gcc has the option –ftrapv which activates the signal SIGABRT.
- This signal can be catched.
- This option works only with int value (not short or unsigned int)
- Example:

```
// gcc -Wall -ftrapv -o file file.C -lstdc++
int main(void) {
int a, b, c;
    createCatchSignal ();
    a=200000;
    b=200000;
    c=a*b;
    printf ("c=%d\n", c);
    return 0;
}
static void createCatchSignal(void) {
struct sigaction act;
    memset (&act, 0, sizeof (act));
    act.sa_handler = catchSignal;
    sigaction (SIGABRT,  &act, 0);
}
static void catchSignal (int signal) {
    printf ("Integer error, signal = %d\n", signal);
    _exit (0);
}
```

Application hardening, 8: strnXXX functions

- Replace strcat, strcpy by strn.... Functions

- man strnpy:
  char *strncpy(char *dest, const char *src, size_t n)

- The strncpy() function is similar to strcpy(), except that at most n bytes of src are copied. Warning: If there is no null byte among the first n bytes of src, the string placed in dest will not be null-terminated.
- If the length of src is less than n, strncpy() pads the remainder of dest with null bytes.
- It is necessary to initialize the buffer to 0 and n must be = sizeof(buffer)-1

```
int main (void)
{
char  buffer[16];
const char *p="AAAAAAAAAAAAAAAA";  //17bytes

    memset (&buffer[0], 0x0, sizeof(buffer));
    strncpy (&buffer[0], p, sizeof (buffer)-1);
    return (0);
}
```

Application hardening, 8: strnXXX functions

- Some C functions such as gets(), strcpy(), strcat(), printf() are known to be insecure because they don't check the size of the destination buffers.

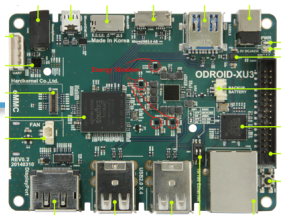
- The following command may be used in order to detect these particular functions in source code.

```
find . -name "*.c" | xargs egrep 'gets|strcpy|strcat|printf|scanf'
```

Nmap commands

192.168.0.4

192.168.0.11



From 192.168.0.4:

nmap –sV –PN –p 1-65535 192.168.0.11 // scan all tcp ports, with software version

nmap –sU –PN –p 1-65535 192.168.0.11 // scan all udp ports, with software version

Nmap scripts

"script man pages" are here: nmap.org/book/man-nse.html

Scripts are listed to: nmap.org/nsedoc/.

On computer, scripts are saved to **/usr/share/nmap/scripts**

Example: ssh

```
ls /usr/share/nmap/scripts/ssh
   ssh2-enum-algos.nse  ssh-hostkey.nse      sshv1.nse

nmap --script-help ssh2-enum-algos
  ssh2-enum-algos
  Categories: safe discovery
  http://nmap.org/nsedoc/scripts/ssh2-enum-algos.html

nmap --script ssh2-enum-algos 192.168.0.11
   PORT   STATE SERVICE
   22/tcp open  ssh
   | ssh2-enum-algos:
   |   kex_algorithms: (6)
   |       curve25519-sha256@libssh.org
   |       ecdh-sha2-nistp521
   |       diffie-hellman-group-exchange-sha256
   |       …
```