

Linux security and system hardening

lab 1

Jeffrey Benteux - <https://www.benteux.fr>

September 29, 2026

objective

setup

vanilla kernel

configuration hardening

Module signature

Linux Security Modules
(LSM)

memory protections

BPF

command-line hardening

sysctl hardening

conclusion

objective

setup

vanilla kernel

configuration hardening

command-line hardening

sysctl hardening

conclusion

objective

- build a kernel from source
- harden the configuration, the command line and the runtime

Outline

objective

setup

vanilla kernel

configuration hardening

command-line hardening

sysctl hardening

conclusion

- virtual machine with Qemu
- Debian 13 image¹
- Kernel sources²

¹<https://cloud.debian.org/images/cloud/trixie/20260914-2601/debian-13-nocloud-amd64-20260914-2601.qcow2>

²<https://github.com/torvalds/linux>

install dependencies

```
pacman -S qemu-base qemu-desktop
```

boot base image

```
$ qemu-img info lab1-debian-13-nocloud-amd64-20260914-2601.qcow2
[...]
$ qemu-system-x86_64 -drive file=$PWD/lab1-debian-13-nocloud-amd64-20260914-2601.qcow2 -m 1024M -enable-kvm -cpu host -net user,hostfwd=tcp::10022-:22 -net nic -nographic

root:root
```

check SSH

```
# On host
$ ss -lptn
State      Recv-Q      Send-Q      Local Address:Port      Peer Address:Port      Process
LISTEN     0            1            0.0.0.0:10022 0.0.0.0:*      users:((("qemu-system-x86",pid
           =121096,fd=12))
# user:user
$ ssh user@127.0.0.1 -p 10022
user@localhost:~$ id
uid=1000(user) gid=1000(user) groups=1000(user),100(users)
```

objective

setup

vanilla kernel

configuration hardening

command-line hardening

sysctl hardening

conclusion

- we are first going to compile a vanilla kernel, with a minimal configuration

build configuration

```
# de-tar kernel sources
$ tar xf linux-7.2.tar.gz
$ cd linux-7.2
# making base config, with qemu support
$ make defconfig
$ make kvm_guest.config
# You can check what has been done
$ less .config
```

compile kernel

```
# Compile kernel first
$ make -j8
[...]  
Kernel: arch/x86/boot/bzImage is ready (#1)  
# Compile modules  
$ make -j8 modules
```

boot on kernel

```
$ pwd
/path/to/linux-7.2
$ qemu-system-x86_64 -drive
file=/path/to/lab1-debian-13-nocloud-amd64-20260914-2601.
qcow2 -m
1024M -enable-kvm -cpu host -kernel./arch/x86_64/boot/bzImage -append
'root=/dev/sda1' -net user,hostfwd=tcp::10022-:22 -net nic
$ ssh user@127.0.0.1 -p 10022
user@localhost:~$ uname -a
Linux localhost 7.2.0
#1 SMP PREEMPT_DYNAMIC XXXX x86_64 GNU/Linux
```

Questions?

Outline

objective

setup

vanilla kernel

configuration hardening

- Module signature

- Linux Security Modules (LSM)

- memory protections

- BPF

command-line hardening

sysctl hardening

conclusion

Outline

objective

setup

vanilla kernel

configuration hardening

- Module signature

- Linux Security Modules (LSM)

- memory protections

- BPF

command-line hardening

sysctl hardening

conclusion

- let's start with module signature

module signature

enable module signature

```
$ make menuconfig
# Enable loadable module support ->
# Module signature verification ->
# Require modules to be validly signed AND
# Automatically sign all modules
# save and exit

# Check with
$ grep SIG_ .config
```

- once a change has been made to the configuration, we need to rebuild the kernel

recompile, boot and check

```
$ make -j8
$ make -j8 modules
...
# On guest
$ cat /sys/module/module/parameters/sig_enforce
Y
```

Outline

objective

setup

vanilla kernel

configuration hardening

- Module signature

- Linux Security Modules (LSM)

- memory protections

- BPF

command-line hardening

sysctl hardening

conclusion

- enable LSM and securityfs

- enable LSM and securityfs
- enable wanted LSMs

- enable LSM and securityfs
- enable wanted LSMs
- we will focus on: YAMA, AppArmor, SELinux

kernel configuration for LSMs

```
$ make menuconfig
# type '/', look for and enable
# CONFIG_SECURITY
# CONFIG_SECURITYFS
# CONFIG_SECURITY_YAMA
# CONFIG_SECURITY_APPARMOR
# CONFIG_SECURITY_SELINUX

# Save and recompile
```

kernel command line for LSMs

```
# Append to kernel command line
qemu-system-x86_64 -drive
file=/path/to/lab1/lab1-debian-13-nocloud-amd64
-20260914-2601.qcow2
-m 1024M -enable-kvm -cpu host -kernel ./arch/x86_64/boot/
bzImage
-append 'root=/dev/sda1 lsm=yama,apparmor
      ' -net user ,hostfwd=tcp::10022-:22 -net nic &

# Boot and check
root@localhost:/home/user# cat /sys/kernel/security/lsm
capability ,yama ,apparmor
```

check apparmor status and profiles

```
$ journalctl —grep 'apparmor'
[...]  
localhost kernel: Kernel command line: root=/dev/sda1 lsm=yama,apparmor  
localhost kernel: AppArmor: AppArmor initialized  
localhost kernel: AppArmor: AppArmor Filesystem Enabled  
localhost kernel: AppArmor: AppArmor sha256 policy hashing enabled  
[...]  
  
# /sbin/aa-status  
apparmor module is loaded.  
106 profiles are loaded.  
7 profiles are in enforce mode.  
[...]
```

- because AppArmor and SELinux are mutually exclusive, we need to reboot with `security=selinux`

boot with SELinux

```
qemu-system-x86_64 -drive
file=/path/to/lab1/lab1-debian-13-nocloud-amd64
-20260914-2601.qcow2
-m 1024M -enable-kvm -cpu host -kernel ./arch/x86_64/boot/
bzImage
-append 'root=/dev/sda1 lsm=yama,selinux
        ' -net user,hostfwd=tcp::10022-:22 -net nic
```

```
~$ sestatus
SELinux status:                enabled
SELinuxfs mount:               /sys/fs/selinux
SELinux root directory:        /etc/selinux
[...]
```

check processes contexts

```
$ ps -fauxZ
```

```
# play with seinfo , semanage , sestatus
```

Outline

objective

setup

vanilla kernel

configuration hardening

- Module signature

- Linux Security Modules (LSM)

- memory protections

- BPF

command-line hardening

sysctl hardening

conclusion

- memory protection are usually enable by default
- let's enable them if not

configure memory protections: DEP

```
$ grep RWX .config  
CONFIG_ARCH_HAS_STRICT_KERNEL_RWX=y  
CONFIG_STRICT_KERNEL_RWX=y  
CONFIG_ARCH_HAS_STRICT_MODULE_RWX=y  
CONFIG_STRICT_MODULE_RWX=y
```

configure memory protections: stack

```
grep -E 'STACKPROTECTOR|VMAP_STACK|STACK_END' .config  
CONFIG_HAVE_STACKPROTECTOR=y  
CONFIG_STACKPROTECTOR=y  
CONFIG_STACKPROTECTOR_STRONG=y  
CONFIG_HAVE_ARCH_VMAP_STACK=y  
CONFIG_VMAP_STACK=y  
# CONFIG_SCHED_STACK_END_CHECK is not set
```

configure memory protections: stack

```
$ make menuconfig  
# Type '/' and SCHED_STACK_END_CHECK  
# save, compile and reboot
```

configure memory protections: no DMA

```
$ grep -E 'CONFIG_(DEVMEM|DEVKMEM|ACPI_CUSTOM)' .config  
CONFIG_DEVMEM=y  
  
$ make menuconfig  
# Type '/' and DEVMEM, disable it  
# save, compile and reboot
```

configure memory protections: no legacy features

```
grep -E 'CONFIG_(COMPAT_VDSO|LEGACY_VSYSCALL|  
    X86_VSYSCALL_EMULATION)' .config  
CONFIG_X86_VSYSCALL_EMULATION=y  
# CONFIG_COMPAT_VDSO is not set  
CONFIG_LEGACY_VSYSCALL_XONLY=y  
# CONFIG_LEGACY_VSYSCALL_NONE is not set  
  
$ make menuconfig  
# Look for the enabled features above, disable these, save  
  and recompile
```

exercice

```
# Do so that your kernel config contains:  
CONFIG_SECURITY_DMESG_RESTRICT=y  
# CONFIG_PROC_KCORE is not set  
CONFIG_RANDOMIZE_BASE=y  
CONFIG_RANDOMIZE_MEMORY=y  
CONFIG_RANDOMIZE_KSTACK_OFFSET=y  
  
# Reboot 2 times and check that the following returns  
  different addresses:  
# head -n 3 /proc/kallsyms
```

Outline

objective

setup

vanilla kernel

configuration hardening

- Module signature

- Linux Security Modules (LSM)

- memory protections

- BPF

command-line hardening

sysctl hardening

conclusion

enable BPF support

```
# Do so that your kernel config contains:  
CONFIG_BPF=y  
CONFIG_BPF_SYSCALL=y  
CONFIG_BPF_JIT=y  
CONFIG_BPF_LSM=y  
CONFIG_BPF_UNPRIV_DEFAULT_OFF=y  
# CONFIG_BPF_JIT_ALWAYS_ON is not set
```

enable BPF support

```
# boot with BPF LSM
qemu-system-x86_64 -drive
file=/path/to/lab1/lab1-debian-13-nocloud-amd64
-20260914-2601.qcow2
-m 1024M -enable-kvm -cpu host -kernel ./arch/x86_64/boot/
bzImage
-append 'root=/dev/sda1 lsm=yama,selinux,bpf
        ' -net user ,hostfwd=tcp::10022-:22 -net nic

# On guest
# cat /sys/kernel/security/lsm
capability ,yama ,selinux ,bpf
```

Outline

objective

setup

vanilla kernel

configuration hardening

command-line hardening

sysctl hardening

conclusion

- we will add numerous parameters to the command-line

- we will add numerous parameters to the command-line
- be sure no-one can edit kernel command-line
 - ie: put a password to your boot manager

Based on ANSSI guide³

- `l1tf=full,force`: defences against L1 Terminal Fault (L1TF) vulnerability

³https://messervices.cyber.gouv.fr/documents-guides/fr_np_linux_configuration-v2.0.pdf

Based on ANSSI guide³

- `l1tf=full,force`: defences against L1 Terminal Fault (L1TF) vulnerability
- `page_poison=on`: activate buddy allocator page poisoning, put patterns in freed memory

³https://messervices.cyber.gouv.fr/documents-guides/fr_np_linux_configuration-v2.0.pdf

command-line parameters

Based on ANSSI guide³

- `l1tf=full,force`: defences against L1 Terminal Fault (L1TF) vulnerability
- `page_poison=on`: activate buddy allocator page poisoning, put patterns in freed memory
- `pti=on`: Page Table Isolation

³https://messervices.cyber.gouv.fr/documents-guides/fr_np_linux_configuration-v2.0.pdf

command-line parameters

Based on ANSSI guide³

- `l1tf=full,force`: defences against L1 Terminal Fault (L1TF) vulnerability
- `page_poison=on`: activate buddy allocator page poisoning, put patterns in freed memory
- `pti=on`: Page Table Isolation
- `slab_nomerge=yes`: disable slabs cache fusioning, decrease heap massaging capacity during heap overflows

³https://messervices.cyber.gouv.fr/documents-guides/fr_np_linux_configuration-v2.0.pdf

command-line parameters

Based on ANSSI guide³

- `l1tf=full, force`: defences against L1 Terminal Fault (L1TF) vulnerability
- `page_poison=on`: activate buddy allocator page poisoning, put patterns in freed memory
- `pti=on`: Page Table Isolation
- `slab_nomerge=yes`: disable slabs cache fusioning, decrease heap massaging capacity during heap overflows
- `slub_debug=FZP`: activate checks in slab caches

³https://messervices.cyber.gouv.fr/documents-guides/fr_np_linux_configuration-v2.0.pdf

Based on ANSSI guide⁴

- `spec_store_bypass_disable=seccomp`: Spectre V4 mitigation
- `spectre_v2=on`: Spectre V2 mitigation

⁴https://messervices.cyber.gouv.fr/documents-guides/fr_np_linux_configuration-v2.0.pdf

Based on ANSSI guide⁴

- `spec_store_bypass_disable =seccomp`: Spectre V4 mitigation
- `spectre_v2=on`: Spectre V2 mitigation
- `mds=full,nosmt`: Microarchitectural Data Sampling, Intel processor vulnerabilities mitigation

⁴https://messervices.cyber.gouv.fr/documents-guides/fr_np_linux_configuration-v2.0.pdf

Based on ANSSI guide⁴

- `spec_store_bypass_disable=seccomp`: Spectre V4 mitigation
- `spectre_v2=on`: Spectre V2 mitigation
- `mds=full,nosmt`: Microarchitectural Data Sampling, Intel processor vulnerabilities mitigation
- `mce=0`: force a kernel panic on Machine Check errors

⁴https://messervices.cyber.gouv.fr/documents-guides/fr_np_linux_configuration-v2.0.pdf

Based on ANSSI guide⁴

- `spec_store_bypass_disable =seccomp`: Spectre V4 mitigation
- `spectre_v2=on`: Spectre V2 mitigation
- `mds=full,nosmt`: Microarchitectural Data Sampling, Intel processor vulnerabilities mitigation
- `mce=0`: force a kernel panic on Machine Check errors
- `page_alloc . shuffle =1`: enable page allocator randomization

⁴https://messervices.cyber.gouv.fr/documents-guides/fr_np_linux_configuration-v2.0.pdf

Based on ANSSI guide⁴

- `spec_store_bypass_disable =seccomp`: Spectre V4 mitigation
- `spectre_v2=on`: Spectre V2 mitigation
- `mds=full,nosmt`: Microarchitectural Data Sampling, Intel processor vulnerabilities mitigation
- `mce=0`: force a kernel panic on Machine Check errors
- `page_alloc . shuffle =1`: enable page allocator randomization
- `rng_core . default_quality =500`: increase entropy trust on HWRNG of a TPM for Linux RNG

⁴https://messervices.cyber.gouv.fr/documents-guides/fr_np_linux_configuration-v2.0.pdf

command-line parameters

boot with hardened command-line

```
qemu-system-x86_64 -drive  
file=/path/to/lab1/lab1-debian-13-nocloud-amd64-20260914-2601.qcow2  
-m 1024M -enable-kvm -cpu host -kernel ./arch/x86_64/boot/bzImage  
-append 'root=/dev/sda1 lsm=yama,selinux,bpf l1tf=full,force page.poisson=on  
pti=on,slab_nomerge=yes slub_debug=FZP spec_store_bypass_disable=seccomp_spectre_v2=on  
mds=full,nosmt mce=0 page_alloc.shuffle=1rng_core.default_quality=500  
' -net user,hostfwd=tcp::10022-:22 -net nic
```

Outline

objective

setup

vanilla kernel

configuration hardening

command-line hardening

sysctl hardening

conclusion

- numerous parameters can be changed at runtime with `/proc/sys`

- numerous parameters can be changed at runtime with `/proc/sys`
- we will change some based on ANSSI guide, kernel doc, arch linux wiki and SUSE security guide

- numerous parameters can be changed at runtime with `/proc/sys`
- we will change some based on ANSSI guide, kernel doc, arch linux wiki and SUSE security guide
- get the configuration from <https://github.com/jeffbencteux/sysctlchk>

sysctlchk

```
# Transfer tool to guest
$ scp -r -P 10022 ./sysctlchk user@127.0.0.1:/home/user

# Run
$ ./sysctlchk.sh
[...]
Loading reference file refs/all.conf

[+] net.ipv4.ip_forward
[-] net.ipv4.conf.all.rp_filter
[...]
Statistics:
    19 passed
    32 failed
    2 not found
    0 errors
    53 total
```

sysctlchk

```
# Change parameters with ref file
$ sysctl -p ./refs/all.conf

# Run the tool again
$ ./sysctlchk.sh
[...]
Loading reference file refs/all.conf
[...]
Statistics:
    50 passed
    1 failed
    2 not found
    0 errors
    53 total
```

- experiment with the tool and use `sysctl`

Outline

objective

setup

vanilla kernel

configuration hardening

command-line hardening

sysctl hardening

conclusion

Attacks on our system:

Attacks on our system:

- attackers cannot load malicious modules anymore

Attacks on our system:

- attackers cannot load malicious modules anymore
- kernel is now hardened against several memory bugs, some future exploits will likely fail

Attacks on our system:

- attackers cannot load malicious modules anymore
- kernel is now hardened against several memory bugs, some future exploits will likely fail
- attackers' life to make a reliable exploit is way harder

Attacks on our system:

- attackers cannot load malicious modules anymore
- kernel is now hardened against several memory bugs, some future exploits will likely fail
- attackers' life to make a reliable exploit is way harder
- old vulnerabilities such as Spectre serie will not work

Attacks on our system:

- attackers cannot load malicious modules anymore
- kernel is now hardened against several memory bugs, some future exploits will likely fail
- attackers' life to make a reliable exploit is way harder
- old vulnerabilities such as Spectre serie will not work
- userland bugs will be partially contained thanks to MAC LSMs

Attacks on our system:

- attackers cannot load malicious modules anymore
- kernel is now hardened against several memory bugs, some future exploits will likely fail
- attackers' life to make a reliable exploit is way harder
- old vulnerabilities such as Spectre serie will not work
- userland bugs will be partially contained thanks to MAC LSMs

- kernel hardening is never bulletproof

- kernel hardening is never bulletproof
- it is a continuous job

- `https://docs.kernel.org/security/self-protection.html`
- `https://www.kernel.org/doc/html/v4.14/admin-guide/kernel-parameters.html`

The End. Questions?