CONFIGURATION RECOMMENDATIONS OF A GNU/LINUX SYSTEM

ANSSI GUIDELINES

ANSSI-BP-028-EN
03/10/2022

TARGETED AUDIENCE:

Developers **Administrators** **IT security managers** IT managers Users

Information

Warning

This document, written by ANSSI, the French National Information Security Agency, presents the **"Configuration recommendations of a GNU/LINUX system"**. It is freely available at `www.ssi.gouv.fr/en/`.

It is an original creation from ANSSI and it is placed under the "Open Licence v2.0" published by the Etalab mission [15].

According to the Open Licence v2.0, this guide can be freely reused, subject to mentioning its paternity (source and date of last update). Reuse means the right to communicate, distribute, redistribute, publish, transmit, reproduce, copy, adapt, modify, extract, transform and use, including for commercial purposes.

These recommendations are provided as is and are related to threats known at the publication time. Considering the information systems diversity, ANSSI cannot guarantee direct application of these recommendations on targeted information systems. Applying the following recommendations shall be, at first, validated by IT administrators and/or IT security managers.

This document is a courtesy translation of the initial French document **"Recommandations de configuration d'un système GNU/Linux"**, available at `www.ssi.gouv.fr`. In case of conflicts between these two documents, the latter is considered as the only reference.

Document changelog:

VERSION	DATE	CHANGELOG
1.0	08/10/2015	Initial version
1.1	12/08/2016	Minor revision
1.2	22/02/2019	Minor revision and transition to the new ANSSI model. English version based on a courtesy translation provided by Red Hat
2.0	03/10/2022	Integration of the hardening elements of the Linux Kernel and reorganization of the chapters. English version based on a courtesy translation provided by Red Hat

Contents

1
Preamble

1.1 Foreword

Today Unix operating systems and derivatives, including GNU/Linux, are playing an important role in the ecosystem of equipments, systems, networks and telecommunications. They are widely deployed in several equipments (switches, routers, TV systems, vehicles…).

Their diversity and their composition are such that they are used according to a large number of possible combinations. Addressing in detail each possible use case is not the point of this guide. However some configuration rules allow to get a reasonably safe system if certain fundamental principles are respected, and to methodologically verify that they are properly applied, by using (for example) a checklist.

This guide targets system administrators and designer of products based on Linux who want to reinforce the security of their systems. It focuses mainly on generic system configuration guidelines and on common sense principles that need to be applied during the deployment of services on a GNU/Linux system. The configuration of these services themselves can be subject to a dedicated technical note, for example the *Recommandations pour la sécurisation des sites `Web`* [5] or *(Open)SSH secure use recommendations* [3] on the ANSSI website. The reader is invited to refer to the document *Recommandations pour la mise en place de cloisonnement système* [8] for a more specific and detailed approach of this subject.

It is advisable to study the applicability and the maintainability of each recommendation in the context of the considered use case. It is also strongly recommended to have recourse to the skills of an expert in GNU/Linux systems for the implementation of these good practices.

1.2 Hardening Level

GNU/Linux distributions are highly heterogeneous, the control of the system platform is a complex task; expertise becomes really necessary as the number of services and servers increases. However, some hardening measures can be implemented based on the expected security level, which will depend on the sensitivity of the data handled or hosted by the system and the robustness of access controls realized in order to access resources.

The recommendations of this guide are given based on an estimated level of hardening. This level does not necessarily depend on the difficulty and time required to deploy a given recommendation, elements that can only be appreciated after an audit in a given context. These levels should be seen as guiding principles to help in the system administration.

Except corner cases, all systems should implement the recommendations of the minimal level. The application of intermediary level recommendations should then be targeted. Enhanced and high level recommendations are not to be applied systematically. These may only be cost-effective for systems in a greater need of security. Additionally, these may require qualified personnel and a major effort for their correct implementation and maintenance over time, otherwise they could become counterproductive. On the other hand, the application of the enhanced and high level recommendations can bring a significant security gain and will be necessary in some contexts.

For example, hardening the kernel will be very beneficial to systems hosting containers because these rely on kernel functionalities as their partitioning primitives. However, a kernel not updated regularly due to a lack of human resources will degrade considerably the level of security of the system, exposing it to the many vulnerabilities found and disclosed each day.

Level	Description
	**Minimal** level recommendations. To be implemented systematically on every system.
	**Intermediary** level recommendations. To be implemented as soon as possible on most systems once the minimal level recommendations are applied.
	**Enhanced** level recommendations. To be implemented on systems in need of stronger security or that have multiple applications that must be isolated from each other.
	**High** level recommendations. To be implemented only if the internal resources have enough skills and time to maintain them, otherwise the security of the system may be degraded. However, these recommendations can bring huge security improvements.

Table 1 – Hardening Level Reading Table

Based on the targeted hardening level, the recommendations apply from the minimum level up to the targeted level.

2
Threats and attackers' objectives

At first glance, an operating system such as Linux does not appear to be directly exposed to threats in the same way as, for example, a web application can be. However, all services, no matter how exposed, depend on the underlying system to perform some operations such as reading and writing to the disk or using network devices. Therefore services indirectly expose the operating system to a attacker via their usual mode of operation or via a vulnerability.

Operating systems have therefore developed different mechanisms to protect themselves against malicious applications or users, but also to separate the resources of the different applications and limit the impact of a vulnerability.

The threats being extremely varied, we will classify them here by objective.

- **The availability**, or rather the fact of making unavailable the targeted system is the easiest objective for an attacker to achieve. The overload of one of the resources (network, file system, memory…) can affect the entire system. The exploitation of some vulnerabilities can also cripple a resource, for instance by forcing an application to constantly restart.

  The objective of an attacker may be to prevent the use of a business-specific application, but also to interfere with the proper functioning of the security functions of an information system. A denial of service on a log aggregation server will for example prevent the reception of logs from other machines, hiding the malicious activity of an attacker and preventing any remediation.

- **The confidentiality** or data theft is another goal of attackers. Business data theft (databases, files…) or identity theft may have a direct profit motive. Finally, machine account theft and privilege escalation can allow an attacker to lateralize and access new resources. The initially corrupted machine is then used as a gateway to reach unexposed and more critical systems.

- **The integrity** of the system is the last goal of attackers. Its corruption allows an attacker to reuse the resources at his own profit. It can thus disfigure exposed applications to send a message, use computing resources for mining cryptocurrencies, or even use it as a gateway to conduct attacks anonymously.

  Changes on the system also often involves setting up means of persistence. These allow the attacker to come back to the system to modify their malware according to their need or to deploy it again if it has been detected and deleted. These means of persistence can be implemented at any level (application, kernel, boot chain…).

Example

Some threats combine several of the objectives described above. For instance, a ransomware can exfiltrate data from machines (data theft), then encrypt the machine's

> disks (availability).

Protection against these threats involves hardening both the applications and the operating system, maintaining them up to date and secure, implementing detection mechanisms and by regularly organizing penetration tests and security audits.

3
General principles of security and hardening

Objective

Guide the installation and configuration of an operating system through different essential principles.

3.1 Principle of minimization

This principle indicates that conceived and deployed systems should avoid as much as possible unnecessary complexity, in order to:

- Reduce the attack surface to a minimum;
- Allow updating and effective system monitoring;
- Permit a more accessible monitoring activity of systems, the number of components to be monitored being reduced.

The implementation of this principle is sometimes delicate because it can quickly become in contradiction with others, equally important. Only a case study with the help of a system and security expertise will allow to make reasonable choices. The following chapters give targeted recommendations according to the considered parts of the system.

3.2 Principle of least privilege

This principle defines that any object or entity managed by a system has only the rights strictly necessary for its execution, and nothing more.

The goal is both a gain in security and safety:

- the consequences of dysfunctions or vulnerabilities are limited by the privileges granted;
- the alteration and / or compromise of the system requires an escalation of privileges, less trivial and unobtrusive to achieve in cases where multiple layers of protection are set up.

3.3 Defense in depth principle

This principle imposes the design of several independent and complementary layers of security to delay an attacker whose mission is to compromise the system.

Each layer of security is therefore a point of resistance that the attacker must cross. Escaping from a layer is accompanied by signals, alarms or log messages that allow to detect a suspicious activity and to be able to react to it. The remediation stage is also facilitated by additional aggregated information on the context of the compromise.

This principle has indeed a real advantage: detection, ease of remediation, and improvement of security.

Under Unix and derivative operating systems, defense in depth must be based on a combination of barriers that must be kept independent of each other.

Example

- authentication required before performing any operation, specifically when they are privileged;
- centralised journaling of system and service events;
- preferential use of services that implement partitioning mechanisms or separation of privileges.

4
Hardware configuration

Objective

Configure certain hardware options in order to harden the base that will be used for the installation. This configuration should preferably be done before installation so that the system is able to take this into account as soon as possible.

4.1 Hardware support

The recommendations of the technical note "*Recommandations de configuration matérielle de postes clients et serveurs x86*" [2] apply to x86 architectures. However, the approach remains applicable to other architectures except that the mechanisms and configuration directives may be different.

R1

Choosing and configuring the hardware

It is recommended to apply the configuration recommendations for hardware support mentioned in the technical note "*Recommandations de configuration matérielle de postes clients et serveurs x86*" [2].

4.2 BIOS/UEFI

The BIOS (`Basic Input/Output System`) or its modern counterpart, the UEFI (`Unified Extensible Firmware Interface`) is the main interface for the hardware configuration of the system. This interface is often only accessible during the first moments of starting the machine by typing a combination of keys. The hardware configuration of the machine depends more on the use to which it will be put than on the operating system installed. However, it should be noted that disabling or enabling features at the BIOS or UEFI level may block the system from using them. The technical note "*Recommandations de configuration matérielle de postes clients et serveurs x86*" [2] discusses the different options that can be found on a contemporary x86 machine.

R2

Configuring the BIOS/UEFI

It is recommended to apply the configuration recommendations for BIOS/UEFI mentioned in the technical note "*Recommandations de configuration matérielle de postes clients et serveurs x86*" [2].

4.3 UEFI secure boot

UEFI Secure Boot is a code verification mechanism loaded by UEFI. It is designed to prevent the execution of unsigned code by the machine.

The codes loaded by the UEFI can be:

- bootloaders,
- UEFI firmware update binaries or
- an image of a Linux kernel in EFI format using EFI Boot Stub [1].

The functioning of UEFI Secure Boot is based on a signature and fingerprint mechanism: a code whose signature or fingerprint is not recognized by the UEFI or one of the signature verification entities of the boot chain cannot be loaded.

Information

The UEFI Secure Boot verification mechanism relies on 4 types of variables (UEFI Specification Version 2.3.1):

- a **db** database (Authorized Signature database) which contains the list of signatures, public keys or fingerprints of codes authorized to be loaded,
- a **dbx** database (Revoked Signature database) which contains the list of signatures, public keys or checksums of revoked codes not authorized,
- one or more **KEK** keys (Key Exchange Key or Key Enrollment Key) which are used to verify the signature of the db and dbx databases,
- a **PK** key (Platform Key) that locks and secures the UEFI firmware against unauthorized modifications. This key manages in particular the updates of the KEK and it is generally provided by the hardware manufacturer.

4.3.1 Preloaded keys

In the context of secure boot, a SHIM is a simple EFI application that, when executed, will load and execute another application. It acts as an upstream of the bootloader on UEFI systems. Its signature is verified with the loaded keys in UEFI. A SHIM makes it possible to avoid having each loaded code individually signed.

Information

Most x86 machines come preloaded with Microsoft's KEK. The UEFI of these machines therefore authorizes the load of codes signed by the UEFI certification service hosted by Microsoft. A SHIM signed by this homologation service is developed by the maintainers of certain distributions, for example the maintainers of distributions derived from Red Hat [2] or Debian [3] who are then in charge of signing each Linux kernel of the distribution and the bootloader used.

1. https://www.kernel.org/doc/html/latest/admin-guide/efi-stub.html

Once loaded, a SHIM verifies the codes loaded no longer with the preloaded keys but with the keys integrated into the SHIM at the time of its compilation, for example, the public key of the CA (Certificate Authority) of keys of the distribution.

R3

Activating the UEFI secure boot

It is recommended to apply the UEFI Secure Boot configuration of the distribution.

4.3.2 New keys

In the absence of an update of the preloaded keys, any vulnerability of a signed version of a SHIM or of a bootloader can allow an attacker to bypass UEFI Secure Boot, for example the BootHole vulnerability, CVE-2020-10713, from a bootloader signed by the maintainers of the distribution.

It is possible to replace the preloaded keys with new keys. In this case, the use of a SHIM is no longer necessary, and it is possible to:

- separately sign the bootloader and the Linux kernel;
- generate and sign an image of the Linux kernel in EFI format which will be verified and loaded directly by UEFI without going through a bootloader.

R4

Replacing of preloaded keys

It is recommended to replace the UEFI preloaded keys with new keys used to sign:
- the bootloader and the Linux kernel, or
- the image of the Linux kernel in EFI format.

Information

To protect the kernel command line parameters detailed in section 5.2 or to load a rebuilt Linux kernel as detailed in section 5.3, it is necessary to replace the preloaded keys with new ones.

It is then important to set up a PKI (Public Key Infrastructure).

4.4 Measured Boot

Measured Boot is a technique complementary to Secure Boot. Indeed, the latter ensures that UEFI binaries of the boot chain are all signed, but lets boot several versions of the system (old version, signed parallel version…). In contrast, Measured Boot relies on more parameters and ensures that the currently booted system is in a specific and expected state.

2. https://github.com/rhboot/shim
3. https://packages.debian.org/source/stable/shim-signed

To do so, Measured Boot relies on the TPM (Trusted Platform Module) and its PCR (Platform Configuration Registers). During startup, the various elements making up the chain (firmware, UEFI binaries, secure boot status…) are measured, and their hashes are stored in the PCR of the TPM. This set of values represents uniquely the state of the currently booted system. A secret can then be sealed in the TPM with an unsealing policy based on a set of PCR and their values. On the next reboot, the secret will be unsealed by the TPM on the sole condition that the values of the PCR of the booted system match the values described in the policy associated with the secret. In other words, the state of the system must be the same as during sealing, ensuring that the boot has not been compromised.

This secret can be used, for example, as a disk encryption key. In the event of a compromised boot, Measured Boot will not interrupt the boot process, but the TPM will not unseal the secret, preventing disk decryption.

Information

Each PCR allows the measurement of one or more components during boot. The first eight PCR (PCR{0..7}) are used during the boot chain. The definition of the measurements carried by each PCR is defined by the Trusted Computing Group[4], and some measurements differ depending on the hardware architecture.

Warning

The update of an element of the boot chain that is subject of a measure alters the value of its hash. On the next boot, the measured values will therefore no longer correspond to the values of the policy to unseal the secret. Therefore, the values in the policy must be updated at the same time to reflect the changes. However, some of these changes can only be observed when the machine reboot and will therefore change the state of the system. For example, a device's firmware update will only be applied on the next reboot, making updating the policy tricky.

4. https://trustedcomputinggroup.org/work-groups/pc-client

5
Linux kernel configuration

The Linux kernel is a rich kernel, in charge of providing both the set of software drivers for the platform [5], a subset of the `POSIX` interface (supplemented by the GNU `libc`) and a whole set of security mechanisms to protect itself and to protect the system.

Objective

Hardening the Linux kernel, is about improving the protection mechanisms of the operating system and validating the presence of the self-protection mechanisms of the kernel, by following the principle of defense in depth. We can provide both protections or countermeasures to potential software or hardware vulnerabilities of the platform and at the same time reduce the attack surface of the kernel, initially very large.

5.1 Bootloader

For reasons equivalent to the BIOS/UEFI setup, the bootloader is an important part of the boot chain. Today's ones (`GNU GRUB 2` [6], `systemd-boot` [7] ...) are functionally rich: they allow access to file systems, possibly modify data, boot from external peripherals or change boot options of the selected kernel.

R5

Configuring a password on the bootloader

A bootloader that protects its boot by a password should be preferred. This password must prevent an ordinary user from changing the bootloader configuration options. When the bootloader does not offer the possibility to set a password, some other technical or organizational measures must be set up to block any user to change the settings.

Information

`GNU GRUB 2` (bootloader for x86 architecture) offers the possibility to setup a password. Consult their respective documentation [16] for more information.

5. with a few exceptions, such as the `libusb` in userspace.
6. https://www.gnu.org/software/grub
7. https://www.freedesktop.org/software/systemd/man/systemd-boot.html

Warning

A change in the configuration of the bootloader may require the execution of a third-party utility to be taken into account in the configuration (`update-grub` for example) as well as a complete reboot of the system.

5.2 Dynamic configuration

This section discusses the configurations for hardening the execution of an already compiled Linux kernel. This applies both to Linux kernels compiled by third parties (distributions…) and to fully controlled and recompiled Linux kernels.

Command line parameters allow a number of Linux kernel configuration options to be changed at boot time. They are often used to override defaults or to specify configuration directives.

Information

With `GNU GRUB 2`, the `GRUB_CMDLINE_LINUX` option of the configuration file `/etc/default/grub` allows you to add or modify the kernel command line parameters.

UEFI secure boot detailed in paragraph 4.3 helps ensure the integrity and authenticity of UEFI binaries during boot (bootloader, Linux kernel…). However, the kernel command line and the `initramfs` (`INITial RAM filesystem`) are not protected as they are not UEFI binaries. To solve this, it is possible to create a `Unified kernel image` which groups together the kernel, the `initramfs` and the kernel command line parameters in a single UEFI binary. The signing of the latter will then be checked at system boot, ensuring the integrity and the authenticity of all these components.

R6

M I E H Protecting the kernel command line parameters and the `initramfs`

It is recommended to protect the kernel command line parameters and the `initramfs`. They should be verified by the UEFI secure boot.

A number of Linux kernel configuration options can be changed dynamically using the `sysctl` command [8]

Information

The config file `sysctl.conf` [9]allows to modify these options of configuration at each system startup using the service `systemd-sysctl.service` [10]of the system and services manager `systemd` [11].

These configuration options can act at any level:

8. https://man7.org/linux/man-pages/man8/sysctl.8.html
9. https://man7.org/linux/man-pages/man5/sysctl.conf.5.html
10. https://man7.org/linux/man-pages/man8/systemd-sysctl.service.8.html
11. https://systemd.io

- **memory:** paging configuration, allocators, mapping properties…
- **kernel:** cache control, `swap`, `scheduling`…
- **process:** allocated resources, execution restrictions, partitioning…
- **network:** sizes and characteristics of buffers, choices of algorithms…
- **file systems:** `setuid dumps`, `hard` and `soft links`, quotas…

These options are enriched with the evolutions and improvements made to the Linux kernel. It is therefore necessary to refer to the documentation (generally under `Documentation/` in the sources or to the official online documentation [1] available for the latest versions of the Linux kernel) for a detailed explanation. Among these options, some have a critical impact in terms of security, and therefore must be carefully considered.

Warning

The scope, use and configuration of the Linux kernel configuration options accessible by command line parameters or dynamically using the `sysctl` command line must be monitored regularly if an administrator wishes to take maximum advantage of their functionality.

5.2.1 Memory configuration

Activating the IOMMU (`Input/Output Memory Management Unit`) service protects the system's memory from being arbitrarily accessed by devices. The effectiveness of this protection depends greatly on how the system is built. Even if its design is imperfect (see [14]), the IOMMU activation on the system contributes to reducing the risk of arbitrary access to memory from devices.

The activation of the functionality depends on the hardware configuration detailed in section 4.1 and the settings of the operating system. Depending on the situation (presence or absence of a functional IOMMU, amount of memory present on the system ...), the Linux kernel may decide not to initialize the IOMMU. It must therefore be instructed to force its use through a configuration directive passed as a parameter during startup.

Warning

Activating the IOMMU on a system can cause hardware instabilities, it is therefore necessary to ensure that it works properly before deploying such a measure in production.

R7

Activating the IOMMU

It is recommended to enable IOMMU by adding the `iommu=force` directive to the list kernel parameters during boot in addition to those already present in the bootloader configuration files.

The memory configuration is relatively generic and should be applicated regardless of the target use case.

R8

Configuring the memory options

The lists below show the recommended memory configuration options.

Lists of recommended memory configuration options

The memory configuration options detailed in this list are to be added to the list of kernel parameters during boot in addition to those already present in the bootloader configuration file:

- *l1tf=full,force* : activate without the possibility of subsequent deactivation all counter measures for the `L1 Terminal Fault` (L1TF) vulnerability present on most Intel processors (at least in 2018). Note that this disables `Symmetric MultiThreading` (SMT) and can therefore have a strong impact on system performance. This option is only necessary, however, when the system is likely to be used as a hypervisor. If the virtual machines are trusted, that is, with a guest operating system that is both trusted and protected against the L1TF vulnerability, this option is not necessary and can even be overridden by l1tf=off to maximize performance;
- *page_poison=on* : activate the `poisoning` of the pages of the page allocator (buddy allocator). This functionality fills the freed pages with a pattern and then checks it on the next reallocation, allowing to reduce the risk of information leaks from the freed data;
- *pti=on* : force the use of `Page Table Isolation` (PTI) including on processors claiming not to be affected by the `Meltdown` vulnerability;
- *slab_nomerge=yes* (equivalent to `CONFIG_SLAB_MERGE_DEFAULT=n`): disables the merging of slab caches (dynamic memory allocations) of identical size. This functionality makes it possible to differentiate the allocations between the different slab caches, and greatly complicates the methodologies of kneading the heap (`heap massaging`) in the event of `heap overflow`;
- *slub_debug=FZP* : activate certain options for checking slabs caches (dynamic memory allocation):
  - F activates the consistency tests of the metadata of the slab caches;
  - Z activates `Red Zoning`; in a slab cache, add a red zone after each object in order to detect writes after it. It is important to note that the value used for the red zone is not random and therefore this is a much lower hardening than using real canaries;
  - P activates the poisoning of objects and padding and causes an error when accessing poisoned areas;
- *spec_store_bypass_disable=seccomp* : force the system to use the default countermeasure (on an x86 system supporting `seccomp`) for the Spectre v4 (`Speculative Store Bypass`) vulnerability;

- *spectre_v2=on* : force the system to use a countermeasure for the Spectre v2 (`Branch Target Injection`) vulnerability. This option activates *spectre_v2_user=on* which avoids `Single Threaded Indirect Branch Predictors` (STIBP) and `Indirect Branch Prediction Barrier`[12](IBPB) attacks;
- `mds=full,nosmt` : force the system to use `Microarchitectural Data Sampling` (MDS) to mitigate the vulnerabilities of Intel processors. The `mds=full` option, which leaves `Symmetric MultiThreading` (SMT) enabled, is therefore not a full mitigation. This mitigation requires an Intel microcode update and also mitigates the Intel processor `TSX Asynchronous Abort` (TAA) vulnerability on systems affected by MDS;
- `mce=0` : force a `kernel panic` on uncorrected errors reported by `Machine Check` support. Otherwise, some of them only cause a `SIGBUS` to be sent, potentially allowing a malicious process to continue trying to exploit a vulnerability such as `Rowhammer`;
- `page_alloc.shuffle=1` (equivalent to `CONFIG_SHUFFLE_PAGE_ALLOCATOR=y`): enables `Page allocator randomization` which improves performance for using `direct-mapped memory-side-cache` but reduces predictability of page allocations and completes thus `SLAB_FREELIST_RANDOM`;
- `rng_core.default_quality=500` : increase confidence in TPM's `HWRNG` for robust and fast Linux `CSPRNG` initialization by crediting half of the entropy it provides.

12. https://www.kernel.org/doc/html/latest/admin-guide/hw-vuln/spectre.html

5.2.2 Kernel configuration

The kernel configuration is relatively generic and should be applicable regardless of the target use case.

R9

Configuring the kernel options

The list below shows the recommended kernel configuration options.

List of recommended kernel configuration options

The detailed kernel configuration options are presented as seen in the `sysctl.conf` configuration file:

```
# Restrict access to the dmesg buffer (equivalent to
# CONFIG_SECURITY_DMESG_RESTRICT=y)
kernel.dmesg_restrict=1
# Hide kernel addresses in /proc and various other interfaces,
# including from privileged users
kernel.kptr_restrict=2
# Explicitly specify the process id space supported by the kernel,
# 65536 being an example value
kernel.pid_max=65536
# Restrict the use of the perf subsystem
kernel.perf_cpu_time_max_percent=1
kernel.perf_event_max_sample_rate=1
# Prohibit unprivileged access to the perf_event_open () system call.
# With a value greater than 2, we impose the possession of
# CAP_SYS_ADMIN, in order to collect the perf events.
kernel.perf_event_paranoid=2
# Activate ASLR
kernel.randomize_va_space=2
# Disable Magic System Request Key combinations
kernel.sysrq=0
# Restrict kernel BPF usage to privileged users
kernel.unprivileged_bpf_disabled=1
# Completely shut down the system if the Linux kernel behaves
# unexpectedly
kernel.panic_on_oops=1
```

It is possible to prevent the loading of new modules (including by `root`) with the Linux kernel configuration option `kernel.modules_disabled`. A kernel module can be loaded explicitly by a privileged user, but also by an unprivileged user using the on-demand loading. This mechanism allows to load a module only when an action needs it. For instance, the creation of a DCCP (`Datagram Congestion Control Protocol`) socket will trigger the loading of the DCCP module by the kernel if it has not been loaded yet. The option `kernel.modules_disabled` protects against the explicit loading of a malicious module, but also against the implicit loading of a vulnerable module (if not loaded yet). An unprivileged user could indeed trigger the loading of such a module to then exploit the vulnerability and increase his privileges.

Setting this option has non-trivial consequences on the system. Thus, it is recommended to test it and ensure that it does not have operational implications.

Especially, the loading of modules being impossible after the option being set, all needed modules should be loaded before setting it. To do so, it is possible to declare all modules to load at boot

time before the option is applied.

R10

Disabling kernel modules loading

It is recommended to block the loading of kernel modules by activating the `kernel.modules_disabled` configuration option.

Configuration option to disable kernel modules loading

The kernel configuration option to block the loading of kernel modules is presented as seen in the `/etc/sysctl.conf` configuration file:

```
# Prohibits loading of kernel modules (except those already loaded at
# this point)
kernel.modules_disabled=1
```

List of the kernel modules loaded by default at boot time

Kernel modules can be listed to be loaded at boot time, before activating the option `kernel.modules_disabled`. The list is declared in `/etc/modules`.
The list of currently loaded modules can be found with the command `lsmod`.

Warning

Once this option is activated, it cannot be deactivated without rebooting the system. Thus, the loading of a new kernel module will require rebooting as well.

5.2.3 Process configuration

The process configuration is relatively generic and should be applicable independently of the target use case.

The security module, `Yama`, allows the addition of the `kernel.yama.ptrace_scope` process configuration option which controls access rights to the `ptrace` system call. This system call is particularly sensitive because it allows to trace the operation of a process. By default, any user can debug all the processes belonging to him, which includes the processes storing sensitive elements in their memory such as keys or passwords (Internet browser, `ssh-agent`…). Compromising a user process would recover these sensitive items.

The `kernel.yama.ptrace_scope` option allows to restrict the use of the `ptrace` system call:

- =0, , all processes can be debugged as long as they share the same UID (`User IDentifier`)
- =1, an unprivileged process (not having `CAP_SYS_PTRACE`) can only debug descendant processes, unless the target process has explicitly authorized it with the command `prctl(PR_SET_PTRACER, pid,…)`
- =2, only a privileged process (owning `CAP_SYS_PTRACE`) can use `ptrace`
- =3, no process can use `ptrace`

R11

Configuration option of the `Yama` LSM

It is recommended to load the `Yama` security module at boot time, for example by passing the `security=yama` directive to the kernel, and to set the kernel configuration option `kernel.yama.ptrace_scope` a value of at least 1.

5.2.4 Network configuration

The network configuration varies depending on the use case (router equipment, terminal equipment, use or not of a network redundancy system). It must therefore be considered in conjunction with the constraints provided by the network specifications of the equipment.

R12

IPv4 configuration options

The list below shows the *IPv4* network configuration options for a typical "server" host that does not perform routing and has a minimalistic *IPv4* configuration (static addressing).

List of recommended IPv4 configuration options

The *IPv4* network configuration options detailed in this list are recommended for a "server" type host that does not perform routing and has a minimalist *IPv4* configuration. They are presented as encountered in the `sysctl.conf` configuration file:

```
# Mitigation of the dispersion effect of the kernel JIT at the cost of a
# compromise on  the associated performance.
net.core.bpf_jit_harden=2
# No routing between interfaces. This option is special and may
# cause modifications to other options. By setting this option early
# we make sure that the configuration of the following options does
# not change.
net.ipv4.ip_forward=0
# Consider as invalid the packets received from outside whose source
# is the 127/8 network.
net.ipv4.conf.all.accept_local=0
# Deny receipt of ICMP redirect packet. The suggested setting of this
# option is to be strongly considered in the case of routers which must not
# depend on an external element to determine the calculation of a route. Even
# for non-router machines, this setting protects against
# traffic diversions with ICMP redirect packets.
net.ipv4.conf.all.accept_redirects=0
net.ipv4.conf.default.accept_redirects=0
net.ipv4.conf.all.secure_redirects=0
net.ipv4.conf.default.secure_redirects=0
net.ipv4.conf.all.shared_media=0
net.ipv4.conf.default.shared_media=0
# Deny the source routing header information supplied by the
# packet to determine its route.
net.ipv4.conf.all.accept_source_route=0
net.ipv4.conf.default.accept_source_route=0
# Prevent the Linux kernel from handling the ARP table globally. By default,
# it can respond to an ARP request from an X interface with information
# from an interface Y. This behavior is problematic for routers and
# equipment of a high availability system (VRRP, etc.).
net.ipv4.conf.all.arp_filter=1
# Respond to ARP requests only if the source and destination addresses are on
# the same network and come from the same interface on which the packet was
    received.
# Note that the configuration of this option is to be studied according to the
```

```
# use case.
net.ipv4.conf.all.arp_ignore=2
# Refuse the routing of packets whose source or destination address is that
# of the local loopback. This prohibits the transmission of packets with
# source network 127/8.
net.ipv4.conf.all.route_localnet=0
# Ignore gratuitous ARP requests. This configuration is
# effective against ARP poisoning attacks but only applicable
# in association with one or more controlled ARP proxies.
# This option can be problematic on networks with devices
# in a high availability setup (VRRP, etc.).
net.ipv4.conf.all.drop_gratuitous_arp=1
# Check that the source address of packets received on a given interface
# would have been contacted via this same interface. Otherwise, the packet
# is ignored. Depending on usage, the value 1 can increase the verification to
# all the interfaces, when the device is a router for which the calculation of
# routes is dynamic. The interested reader is referred to RFC3704 for all
# details for this feature.
net.ipv4.conf.default.rp_filter=1
net.ipv4.conf.all.rp_filter=1
# This option should only be set to 1 in the case of a router because for such
# equipment sending ICMP redirect is normal behaviour. A non-routing
# equipment has no reason to receive a flow for which it is not the recipient
# and therefore to send an ICMP redirect packet.
net.ipv4.conf.default.send_redirects=0
net.ipv4.conf.all.send_redirects=0
# Ignore responses that do not comply with RFC 1122
net.ipv4.icmp_ignore_bogus_error_responses=1
# Increase the range for ephemeral ports
net.ipv4.ip_local_port_range=32768 65535
# RFC 1337
net.ipv4.tcp_rfc1337=1
# Use SYN cookies to prevent SYN flood type attacks.
net.ipv4.tcp_syncookies=1
```

Most of the time, systems only use *IPv4* networks. It is therefore essential, in this case, not to keep *IPv6* unnecessarily active.

R13

Disabling IPv6

When *IPv6* is not used, it is recommended to disable the *IPv6* stack by :

- adding the directive `ipv6.disable=1` to the list of kernel boot parameters in addition to those already present in the configuration files of the bootloader,
- enabling the network configuration options shown in the list below.

List of networking configuration options to disable IPv6

The network configuration options detailed in this list and disabling *IPv6* are presented as encountered in the `sysctl.conf` configuration file:

```
net.ipv6.conf.default.disable_ipv6=1
net.ipv6.conf.all.disable_ipv6=1
```

Information

With `GNU GRUB 2`, disabling `IPv6` plan is possible by adding the following option in the configuration file `/etc/default/grub`:

```
GRUB_CMDLINE_LINUX=" ipv6.disable=1"
```

Otherwise, the entire *IPv6* configuration must be secured as well.

5.2.5 File system configuration

The file system configuration is relatively generic and should be applicable regardless of the target use case.

R14

File system configuration options

The list below shows the recommended file system configuration options in the `sysctl.conf` configuration file format.

List of recommended file system configuration options

The file system configuration options detailed in this list are presented as encountered in the `sysctl.conf` configuration file:

```
# Disable coredump creation for setuid executables
# Note that it is possible to disable all coredumps with the
# configuration CONFIG_COREDUMP=n
fs.suid_dumpable = 0
# Available from version 4.19 of the Linux kernel, allows to prohibit
# opening FIFOs and "regular" files that are not owned by the user
# in sticky folders for everyone to write.
fs.protected_fifos=2
fs.protected_regular=2
# Restrict the creation of symbolic links to files that the user
# owns. This option is part of the vulnerability prevention mechanisms
# of the Time of Check - Time of Use (Time of Check -
# Time of Use)
fs.protected_symlinks=1
# Restrict the creation of hard links to files whose user is
# owner. This sysctl is part of the prevention mechanisms against
# Time of Check - Time of Use vulnerabilities, but also against the
# possibility of retaining access to obsolete files
fs.protected_hardlinks=1
```

5.3 Static configuration

In some cases, the Linux kernel is rebuilt from source. In this case, it is strongly recommended to take the security considerations described in this section into account when configuring kernel sources.

These provide a large amount of protection elements for the kernel but also for equipment applications (principle of minimization, principle of defense in depth,…).

The kernel security configuration is separated into two parts:

- Configuration elements independent of the hardware target
- Configuration elements specific to a given hardware target

Warning

Recompiling the Linux kernel can bring a real security gain. However, the complexity and cost brought by the maintenance must be thought first. Deploying custom packages is indeed more complicated. A private repository and a private PKI to sign the packages must both be deployed. Someone will also need to invest time in following the evolutions of the kernel configuration. Lastly, the recompilations must be frequent to ensure that the kernel is always up to date. It would be counterproductive to have a hardened kernel that is outdated and full of vulnerabilities.

5.3.1 Hardware independent configuration

The page tables hold the memory configuration information for each process, which the MMU (`Memory Management Unit`) will use to manage the access rights to its address space. These must be effectively protected. Likewise, kernel memory must be managed in a secure manner. There are various mechanisms for this, such as canaries, KASLR (`Kernel Address Space Layout Randomisation`) ... These protection mechanisms, just like for application processes, complicate the exploit of a kernel vulnerability.

R15

Compile options for memory management

The list below provides the recommended kernel compile options to harden memory management.

Recommended kernel compile options to harden the kernel memory management

```
# This option replaced CONFIG_DEBUG_RODATA in the kernel (> = 4.11)
# which was a dependency of CONFIG_DEBUG_KERNEL
CONFIG_STRICT_KERNEL_RWX=y
# CONFIG_ARCH_OPTIONAL_KERNEL_RWX and
# CONFIG_ARCH_HAS_STRICT_KERNEL_RWX are dependencies of
# CONFIG_STRICT_KERNEL_RWX
CONFIG_ARCH_OPTIONAL_KERNEL_RWX=y
CONFIG_ARCH_HAS_STRICT_KERNEL_RWX=y
# Check and report unsafe memory mapping permissions, for
# example, when kernel pages are writable and executable.
# This option is not available on all architectures
# hardware (x86> = 4.4, arm64> = 4.10, etc.).
CONFIG_DEBUG_WX=y
# Disable the file system used only for debugging. Protecting this file
# system takes additional work.
CONFIG_DEBUG_FS=n
# Starting with kernel version 4.18, these options add
# -fstack-protector-strong at compilation time to strengthen
# the canary stack, it is necessary to have a version of GCC at least
# equal to 4.9.
# Before version 4.18 of the linux kernel, you must use the options
# CONFIG_CC_STACKPROTECTOR and
# CONFIG_CC_STACKPROTECTOR_STRONG
CONFIG_STACKPROTECTOR=y
CONFIG_STACKPROTECTOR_STRONG=y
# Prohibits direct access to physical memory.
# If necessary and only in this case, it is possible to activate a
# strict access to memory, thus limiting its access, with options
# CONFIG_STRICT_DEVMEM=y and CONFIG_IO_STRICT_DEVMEM=y
#CONFIG_DEVMEM is not set
```

```
# Detects stack corruption during the call to the scheduler
CONFIG_SCHED_STACK_END_CHECK=y
# Impose a check of the limits of the memory mapping of the process
# at the time of data copies.
CONFIG_HARDENED_USERCOPY=y
# Forbid the return to a verification of the mapping with the allocator if
# the previous option failed.
#CONFIG_HARDENED_USERCOPY_FALLBACK is not set
# Added cover pages between kernel stacks. This protects against the effects
# of edge of stack overflows (this option is not supported on all
# architectures).
CONFIG_VMAP_STACK=y
# Impose exhaustive checks on kernel reference counters (<=5.4)
CONFIG_REFCOUNT_FULL=y
# Check the memory copy actions that could cause corruption
# of structure in the kernel functions str*() and mem*(). This verification is
# performed at compile time and at runtime.
CONFIG_FORTIFY_SOURCE=y
# Disable the dangerous use of ACPI, which can lead to direct writing
# in physical memory.
#CONFIG_ACPI_CUSTOM_METHOD is not set
# Prohibit direct access to kernel memory from userspace (<=5.12)
#CONFIG_DEVKMEM is not set
# Prohibits provision of kernel image layout
#CONFIG_PROC_KCORE is not set
# Disable VDSO backward compatibility, which makes it impossible
# to use ASLR
#CONFIG_COMPAT_VDSO is not set
# Prevent unprivileged users from retrieving kernel addresses
# with dmesg (8)
CONFIG_SECURITY_DMESG_RESTRICT=y
# Activate retpolines which are necessary to protect yourself from Spectre v2
# GCC> = 7.3.0 is required.
CONFIG_RETPOLINE=y
# Disable the vsyscall table. It is no longer required by libc and is a
# potential source of ROP gadgets.
CONFIG_LEGACY_VSYSCALL_NONE=y
CONFIG_LEGACY_VSYSCALL_EMULATE=n
CONFIG_LEGACY_VSYSCALL_XONLY=n
CONFIG_X86_VSYSCALL_EMULATION=n
```

Providing protection mechanisms (such as KASLR or canaries) may be insufficient. A number of kernel memory structures are often the target of attacks and mechanisms exist to check their integrity. With these mechanisms, it is possible to detect their corruption and take action to respond, usually by stopping the process that caused the corruption or shutting down the system in the event of an unrecoverable error. These measures are useful because an attack leading to corruption of kernel memory is sufficiently advanced to potentially allow the controlled execution of code in supervisor mode on the equipment or the deactivation of generic protection mechanisms.

R16

Compile options for kernel data structures

The list below provides the recommended kernel compilation options to protect kernel data structures.

Recommended kernel compile options to protect kernel data structures

```
# Check the authorisation data structures
CONFIG_DEBUG_CREDENTIALS=y
# Check the notifications data structures
CONFIG_DEBUG_NOTIFIERS=y
# Check kernel lists
```

```
CONFIG_DEBUG_LIST=y
# Check the kernel Scatter-Gather tables.
CONFIG_DEBUG_SG=y
# Generate a call to BUG() if corruption is detected.
CONFIG_BUG_ON_DATA_CORRUPTION=y
```

There are a number of sensitive data structures within the kernel, hosting important data to security. These structures are those that are conventionally targeted when a kernel vulnerability is exploited. It is possible to activate additional validation mechanisms of these structures to help detect their corruption.

R17

Compile options for the memory allocator

The list below provides the recommended kernel compilation options to harden the memory allocator.

Recommended kernel compile options to harden the memory allocator

```
# Randomly position the free block chaining information of the allocator.
CONFIG_SLAB_FREELIST_RANDOM=y
# CONFIG_SLAB is a dependency of CONFIG_SLAB_FREELIST_RANDOM
CONFIG_SLUB=y
# Protects integrity of the allocator's metadata.
CONFIG_SLAB_FREELIST_HARDENED=y
# Starting with kernel version 4.13, this option disables the merge of
# SLAB caches
CONFIG_SLAB_MERGE_DEFAULT=n
# Activates the checking of the memory allocator structures and resets
# to zero the zones allocated when they are released (requires the
# activation of the page_poison=on memory configuration option
# added to the list of kernel parameters during boot). It is better to use
# the slub_debug memory configuration option added to the list of
# kernel parameters during boot as it allows finer management
# from the debug slub.
CONFIG_SLUB_DEBUG=y
# Clean up memory pages when they are freed.
CONFIG_PAGE_POISONING=y
# Deep cleaning disabled. This option comes at a significant cost;
# however if the performance impact is compatible with the need
# operational of the equipment, it is recommended to activate it. (<=5.10)
CONFIG_PAGE_POISONING_NO_SANITY=y
# The cleaning of the memory pages is carried out with a rewrite
# of zeros in place of the data (<=5.10)
CONFIG_PAGE_POISONING_ZERO=y
# Disable backward compatibility with brk () which makes it impossible to
# implementation of brk () with ASLR.
#CONFIG_COMPAT_BRK is not set
```

It may be necessary to use kernel modules. Indeed, the presence of a heterogeneous fleet may require activating certain features and drivers in the form of modules in order to limit the number of kernel images to manage. However, using kernel modules involves additional configuration considerations. These must be signed with a key generated when the kernel is compiled and their signature verified at load time. This mechanism is natively supported in the Linux kernel.

R18

M I E H Compile options for the management of kernel modules

The list below provides the recommended kernel compilation options to harden the management of kernel modules.

Recommended kernel compile options to harden the management of kernel modules

```
# Activation of module support
CONFIG_MODULES=y
# This option replaced CONFIG_DEBUG_SET_MODULE_RONX
# in the kernel (> = 4.11)
CONFIG_STRICT_MODULE_RWX=y
# This option replaced CONFIG_DEBUG_SET_MODULE_RONX
# in the kernel (> = 4.11)
CONFIG_MODULE_SIG=y
# Prevent loading modules that are unsigned or signed with a key that
# does not belong to us.
CONFIG_MODULE_SIG_FORCE=y
# Activate CONFIG_MODULE_SIG_ALL allows signing all modules
# automatically during the "make modules_install" step, without this option
# modules must be signed manually using the script
# scripts/sign-file. The CONFIG_MODULE_SIG_ALL option
# depends on CONFIG_MODULE_SIG and CONFIG_MODULE_SIG_FORCE,
# so they must be enabled.
CONFIG_MODULE_SIG_ALL=y
# Module signing uses SHA512 as hash function
CONFIG_MODULE_SIG_SHA512=y
CONFIG_MODULE_SIG_HASH="sha512"
# Specifies the path to the file containing both the private key and its
# X.509 certificate in PEM format used for signing modules,
# relative to the root of the kernel.
CONFIG_MODULE_SIG_KEY="certs/signing_key.pem"
```

A vulnerability exploit often involves the execution of arbitrary code by a given process. This execution can cause incorrect behavior due to the corruption of certain application data structures. In the event of a critical error, the kernel generates a call to the `BUG()` macro, the purpose of which is to display a logged event and immediately shut down the faulty component of the system. Compiling the Linux kernel with the `CONFIG_PANIC_ON_OOPS` option will result in a complete system halt in the event of unexpected kernel behavior.

R19

M I E H Compile options for abnormal situations

The list below provides the recommended kernel compilation options to handle abnormal situations.

Recommended kernel compile options to handle abnormal situations

```
# Report on the conditions of the call to the macro kernel BUG ()
# and kill the process that initiated the call. Not setting this variable
# may hide a number of critical errors.
CONFIG_BUG=y
# Shut down the system in the event of a critical error to cut short any
# erroneous behavior.
CONFIG_PANIC_ON_OOPS=y
```

```
# Prevents restarting of the machine after a panic which cuts short
# any attempted brute force attack. This obviously has a strong impact
# on production servers.
CONFIG_PANIC_TIMEOUT=-1
```

The Linux kernel provides a large number of security mechanisms aimed at protecting application processes. These are mainly implemented in the form of LSM (Linux Security Module) which can be executed sequentially to authorize access. Some LSM require centralized configurations in the equipment, such as AppArmor detailed in paragraph 6.3.2 or SELinux detailed in paragraph 6.3.3. Others correspond to kernel hardening without specific configuration like Yama. The kernel also provides the possibility to put in place security mechanisms on individual processes such as seccomp or the namespaces.

R20

M I E H

Compile options for kernel security functions

The list below provides the recommended kernel compilation options to configure kernel security functions.

Recommended kernel compile options to configure kernel security functions

```
# Enables the ability to filter system calls made by an application.
CONFIG_SECCOMP=y
# Activate the possibility of using BPF (Berkeley Packet Filter) scripts.
CONFIG_SECCOMP_FILTER=y
# Enables Linux kernel security primitives, required for LSMs.
CONFIG_SECURITY=y
# Active Yama, which allows to simply limit the use of the system call
# ptrace (). Once the security modules used by the system
# have been selected, support for other security modules should be disabled
# in order to reduce the attack surface.
CONFIG_SECURITY_YAMA=y
# Ensure kernel structures associated with LSMs are always mapped
# read-only  after system boot. In the event that SELinux is
# used, make sure that CONFIG_SECURITY_SELINUX_DISABLE is not set,
# for this option to be available. It is then no longer possible to
# disable an LSM after the kernel has booted. It is however still
# possible to do this by modifying the kernel command line. For the
# 4.18 kernel the present LSMs are: AppArmor, LoadPin, SELinux, Smack,
# TOMOYO and Yama.
#CONFIG_SECURITY_WRITABLE_HOOKS is not set
```

Since GCC 4.5, it is possible to integrate plugins into the compiler to provide additional hardening mechanisms when compiling source code. This technique was, for example, used by the PaX [13] patch. There are now a number of plugins present in the Linux kernel sources in order to add security properties during compilation.

R21

M I E H

Compile options for the compiler plugins

The list below provides the recommended kernel compilation options to configure compiler plugins.

13. https://pax.grsecurity.net

Recommended kernel compile options to configure compiler `plugins`

```
# Enable compiler plugins support (implies the use of GCC).
CONFIG_GCC_PLUGINS=y
# Amplify entropy generation at equipment startup for those
# having inappropriate entropy sources
CONFIG_GCC_PLUGIN_LATENT_ENTROPY=y
# Clean up the contents of the stack at the time of the exit system call.
CONFIG_GCC_PLUGIN_STACKLEAK=y
# Force initialization of structures in memory to avoid data leakage by
# superimposition with an old structure.
CONFIG_GCC_PLUGIN_STRUCTLEAK=y
# Globalization of the previous option in the case of the passage
# of structure by reference between functions if they have uninitialized fields
CONFIG_GCC_PLUGIN_STRUCTLEAK_BYREF_ALL=y
# Build a random memory map for the kernel system structures.
# This option has a strong impact on performance. The option
# CONFIG_GCC_PLUGIN_RANDSTRUCT_PERFORMANCE=y should be used
# instead if this impact is not acceptable.
CONFIG_GCC_PLUGIN_RANDSTRUCT=y
```

Most of the time, information systems use only *IPv4* networks. It is possible, in this case, to compile the Linux kernel without the *IPv6* stack.

R22

Compile options of the IP stack

The list below provides the recommended kernel compilation options to configure the IP stack.

Recommended kernel compile options to configure the IP stack.

```
# Disable the IPv6 plan
#CONFIG_IPV6 option is not set.
# Used to prevent SYN flood type attacks.
CONFIG_SYN_COOKIES=y
```

R23

Compile options for various kernel behaviors

The list below provides the recommended kernel compilation options to configure various kernel behaviors.

Certain Linux kernel behaviors may unnecessarily increase the attack surface of the kernel. If these are not absolutely necessary, they should not be activated.

Recommended kernel compile options to configure various kernel behaviors

```
# Prohibits the execution of a new kernel image after reboot.
#CONFIG_KEXEC is not set
# Prohibits switching to hibernation mode, which allows you to
# substitute the image kernel without his knowledge.
#CONFIG_HIBERNATION is not set
# Disable arbitrary binary format support.
#CONFIG_BINFMT_MISC is not set
# Impose the use of modern ptys (devpts) which number and use
# can be controlled
```

```
#CONFIG_LEGACY_PTYS is not set
# If module support is not absolutely necessary, it must be
# disabled.
#CONFIG_MODULES is not set
```

5.3.2 Hardware dependent configuration

The Linux kernel supports a multitude of 32 or 64 bit architectures (x86, ARM, MIPS, PowerPC…). There are security configurations specific to these architectures. This section deals with the two main architectures, x86 and ARM, in 32 and 64 bit.

R24

M I E H Compile options for 32 bit architectures

The list below shows the recommended kernel compilation options for 32 bit x86 architectures. They are not all relevant on 64 bit architectures.

Recommended kernel compile options for 32 bit architectures

```
# Enable support for physical address extensions, which is a prerequisite
# for support of the NX permission bit in the page table which allows
# certain pages to be marked as non-executable.
CONFIG_HIGHMEM64G=y
CONFIG_X86_PAE=y
# Prohibits the use of memory addresses below a certain value
# (countermeasure against null pointer dereference).
CONFIG_DEFAULT_MMAP_MIN_ADDR=65536
# Makes the base address of the kernel unpredictable.
# This option complicates the task of an attacker.
CONFIG_RANDOMIZE_BASE=y
```

R25

M I E H Compile options for x86_64 bit architectures

The list below shows the recommended kernel compilation options for x86_64 architectures. They are not all relevant on 32 bit architectures.

Recommended kernel compile options for x86_64 bit architectures

```
# Activate full 64-bit mode.
CONFIG_X86_64=y
# Prohibits the use of memory addresses below a certain value
# (countermeasure against null pointer dereference).
CONFIG_DEFAULT_MMAP_MIN_ADDR=65536
# Makes the base address of the kernel unpredictable, this option complicates
# the task of an attacker.
CONFIG_RANDOMIZE_BASE=y
# Makes the address to which kernel components are exposed in
# the process address space unpredictable.
CONFIG_RANDOMIZE_MEMORY=y
# Countermeasure to the Meltdown attack.
CONFIG_PAGE_TABLE_ISOLATION=y
# Disable 32-bit backwards compatibility, which helps reduce the
# attack surface but prevents 32-bit binaries from being executed.
#CONFIG_IA32_EMULATION is not set
# Prohibits the per-process overloading of the Local Descriptor Table
# (mechanism linked to the use of segmentation).
```

```
#CONFIG_MODIFY_LDT_SYSCALL is not set
```

R26

Compile options for ARM architectures

The list below shows the recommended kernel compilation options for ARM architectures.

Recommended kernel compile options for ARM architectures

```
# Prohibits the use of memory addresses below a certain value
# (countermeasure against null pointer dereference).
CONFIG_DEFAULT_MMAP_MIN_ADDR=32768
# Maximizes the size of the virtual memory of the processes (and of the ASLR
# related).
CONFIG_VMSPLIT_3G=y
# Prohibits RWX memory mappings
CONFIG_STRICT_MEMORY_RWX=y
# Prohibits access by the kernel to user memory (mechanism
# equivalent on ARM to SMAP on x86_64).
CONFIG_CPU_SW_DOMAIN_PAN=y
# This option of compatibility with the old ABI ARM is dangerous and
# paves the way for various attacks.
#CONFIG_OABI_COMPAT is not set
```

R27

Compile options for ARM 64 bit architectures

The list below shows the recommended kernel compilation options for ARM 64 bit architectures

Recommended kernel compile options for ARM 64 bit architectures

```
# Prohibits the use of memory addresses below a certain value
# (countermeasure against null pointer dereference).
CONFIG_DEFAULT_MMAP_MIN_ADDR=32768
# Makes the base address of the kernel unpredictable, this option complicates
# the task of an attacker. The entropy necessary for the generation
# of the hazard must be provided by the UEFI or, failing that, by the
# bootloader.
CONFIG_RANDOMIZE_BASE=y
# Prohibits access by the kernel to user memory (mechanism
# equivalent on ARM to SMAP on 86_64).
CONFIG_ARM64_SW_TTBR0_PAN=y
# Countermeasure to the Meltdown attack.
CONFIG_UNMAP_KERNEL_AT_EL0=y
```

6
System configuration

This chapter provides recommendations to follow when installing the system for the first time. Each GNU / Linux operating system and distribution takes its own unique path during this stage. There are, however, a few configuration items that will apply almost universally.

Objective

Apply security settings when installing a GNU / Linux system for the first time.

6.1 Partitioning

It is usual to reserve dedicated partitions for services that can generate a lot of volume in order to avoid saturating the system partitions. The space to reserve for each partition depends on the use cases: a file server will need a large volume for `/srv` or `/var/ftp`, while a log server will be more concerned with the usable volume for `/var/log`. Partitioning must therefore protect and isolate the various components of the file system. The default is often unsatisfactory: it does not take sufficient account of `nosuid` (ignores `setuid` / `setgid` bits), `nodev` (ignores character or block special files), and `noexec` (ignores execute rights).

R28

Typical partitioning

The recommended typical partitioning is the following:

Mount Point	Options	Description
/	<without option>	Root partition, contains the rest of the tree
/boot	nosuid,nodev,noexec (optional noauto)	Contains the kernel and the bootloader. No access required once the boot finished (except update)
/opt	nosuid,nodev (optional ro)	Additional packages to the system. Mount read-only if not used
/tmp	nosuid,nodev,noexec	Temporary files. Must contain only non-executable elements. Cleaned after reboot or preferably type tmpfs
/srv	nosuid,nodev (optional noexec,ro)	Contains files served by a service type Web, FTP…
/home	nosuid,nodev,noexec	Contains the users' HOME.
/proc	hidepid=2 [14]	Contains information on processes and the system
/usr	nodev	Contains the majority of utilities and system files
/var	nosuid,nodev,noexec	Partition containing files which content varies during the life of the system (mails, PID files, databases of a service)
/var/log [15]	nosuid,nodev,noexec	Contains system logs
/var/tmp	nosuid,nodev,noexec	Temporary files kept after reboot

Information

It should be noted that depending on the systems and distributions, certain mounting options will not be applicable at specific times; for example some utilities, package managers, or products require that files written to /tmp or /var must be executable. In these cases, it is necessary to adapt the partitioning or to put in place alternative measures; for example for distributions derived from Debian [16]where /var/lib/dpkg requires execute rights, an alternative would be to implement a maintenance procedure during which the updates are installed, like what can be found on other operating systems.

The /boot partition contains the boot kernel as well as the System.map file(s) holding the symbol table it uses. This file is often scanned by various malicious software in order to build kernel code "exploits" more easily. An ideal partitioning requires that this partition not be mounted automatically at startup (noauto option), because its content is not needed during normal use of a system. However, its mount is essential to perform certain administrative actions, for example updating the kernel.

14. Need to configure the service systemd-logind from systemd and service manager systemd (cf. https://wiki.debian.org/Hardening#Mounting_.2Fproc_with_hidepid)

15. For distributions under systemd, the service systemd-journald from system and service manager systemd makes it possible to avoid the saturation of the system partitions, in particular with the options prefixed by "System" (cf. https://www.freedesktop.org/software/systemd/man/journald.conf.html#SystemMaxUse=)

16. https://www.debian.org

R29

M I E

Access restrictions on /boot

When possible, it is recommended not to automatically mount the `/boot` partition. In any case, access to the `/boot` folder should only be allowed for the `root` user.

Warning

Changing the `/boot` mount requires adapting the system tools to this specific configuration, in particular for the update of the kernel and the boot loader.

Information

The low-level utility, `dpkg`, which manages packages for `Debian`-derived distributions and which is used by the `apt` utility on which the package managers are based, `aptitude` and `synaptic`, can be configured to perform specific commands before or after the package installation commands.

6.2 Accounts

The Unix and derivative operating systems, and in particular GNU/Linux, have a separation of privileges which is based mainly on the notion of access accounts at two levels: the "regular" user accounts and the system administrator user account (commonly known as `root`).

This separation lacks granularity, in particular today where a user has fairly wide access to system primitives (while being increasingly exposed to attacks: theft of accounts and identifiers, leaks of information…), in-depth expert analysis is often required to highlight residual vulnerabilities in order to minimize their consequences as much as possible.

6.2.1 User accounts

Rigorous management of user accounts contributes to securing the system through essential security principles.

R30

M

Removing the unused user accounts

Unused user accounts must be deleted from the system.

The password for user accounts must be chosen with great care and in accordance with current recommendations.

R31

M

User password strength

It is recommended to apply the recommendations of the technical note *Recommandations relatives à l'authentification multifacteur et aux mots de passe* [9].
This password must be unique and specific to each machine.

R32

Configuring a timeout on local user sessions

Local user sessions (console `TTY`, graphical session) must be locked after a certain period of inactivity.

6.2.2 Administrator accounts

It is common to define different levels of privileges on the system according to the prerogatives of the administrators. Some may have responsibilities on the system, others just on the website or the logging infrastructure, or even on databases.

The system administrator (or `root`) account is not suitable for these purposes. It gives full power to the person who has access to it. The use of such a generic account does not facilitate accountability during an incident, and does not promote a model of fine separation of privileges, for example between different administration teams.

It is essential to dissociate the roles on the system, in particular for an administrator: simple user or administrator with privileged rights. In addition, an administrator can intervene in several technical areas. As a result, separate accounts must be created and used according to the role (user or administrator) as well as separate administrative accounts by technical area as described in the guide *Recommandations relatives à l'administration sécurisée des systèmes d'information* [6].

Logically, and to avoid any replay of a potentially compromised secret, the secrets (PIN, password, private key…) must be different for each account and never be reused.

R33

Ensuring the imputability of administration actions

The administrative actions that need privileges must be logged in order to identify an administrator account that has a malicious activity.
This can be done in multiple ways. One, the most common, is to use dedicated accounts for each administrator and use `sudo` to execute any privileged command. A second one, more powerful, is to identify the user uniquely on the system, then log all new processes with `auditd`.
Multiple ways can be used at the same time to get a better traceability.

Use of dedicated administration accounts and `sudo`

Each administrator must have one or more nominative administration accounts, dedicated (local or remote), and not use the system administration account (or `root`) as an access account for administration. Authentication secrets must be different depending on the account used.
The system administration (or `root`) account must be disabled. Actions requiring privileged rights must therefore be based on tools, for example `sudo`, allowing to monitor the activities carried out.
The deactivation of an account can involve the invalidation of this account at the level of its password (deletion of the `pw_passwd` field in the `shadow` and login shell

at `/bin/false`). However, these actions do not stop an administrator from executing `sudo bash` to pop a `root` shell. These are to be combined with organisational measures to forbid using the `root` account.

```
# Locking an account
usermod -L -e 1 <account>
# Deactivate login shell
usermod -s /bin/false <account>
```

Logging the creation of all processes

Logging the creation of all processes allows to recreate the succession of operations executed on the system. Then, it is possible to attribute the action to an administrator if this one has previously been identified uniquely. This identification can be done, in other means, by the usage of dedicated accounts or means of authentication (SSH key used for connection…). Thus, this information needs to be logged with the identification data and the PID of the initial process.
It is possible to log the creation of all processes on the system with the following `auditd` rules:

```
-a exit,always -F arch=b64 -S execve,execveat
-a exit,always -F arch=b32 -S execve,execveat
```

It is important to anticipate that the log volume generated by this method can be huge depending on the tasks the system is doing. It is recommended to use this method combined with a regular log export as recommended in the guide *Recommandations de sécurité pour l'architecture d'un système de journalisation* [7].

6.2.3 Service accounts

The vast majority of Unix and derivative operating systems isolate applications and services from each other by dedicating each to its own user account. For example a web server, once started, uses the privileges of a "`Web`" user, for example `www` or `www-data`, while the name server runs under a separate account, for example `named`. The majority of these services do not need to open a session to do their work, and should thus be disabled.

R34

Disabling the service accounts

Service accounts must be disabled.

It is important to note that some services can be declared with the `nobody` account. When several of them adopt such a behavior, they find themselves sharing the same identifiers and therefore the same privileges at the level of the operating system. A web server and a directory service using the `nobody` account can therefore control each other and alter their execution: modification of configuration, sending of signals, `ptrace` privileges…

R35

Uniqueness and exclusivity of service accounts

Each business service (nginx, php, mysql, …) must have its own service account which must be dedicated exclusively to it.

6.3 Access control

Access control consists of ensuring that an access account has sufficient rights to access to a given resource. Although access control allows for partitioning, the chosen approach is generally different from that adopted by virtualization mechanisms. Indeed, access control often leaves the reference to a system object visible to the application and returns an error if there is insufficient privilege to access it, while a system based on virtualization partitions the application by the absence of a reference to this object (pointer, path access…).

Access control under Unix and derivative systems has historically been based on the notion of user and group of users. Other mechanisms have appeared, in particular through LSM of which `SELinux` detailed in paragraph 6.3.3 and `AppArmor` detailed in paragraph 6.3.2 are the best known and most used, in order to allow an additional finer control of rights. But the separation of privileges based on user and user groups is still the most commonly used. It can be accompanied by partitioning mechanisms (UNIX compatible mechanisms, container…) or filtering (`seccomp`) in order to limit access to the underlying operating system.

6.3.1 Unix traditional model

The historical access model is based on the notion of user who is known by the system through a unique identifier, called UID. Each resource (process, file, directory…) is associated with a UID (its owner), which, together with rights, will authorize or deny access to a user.

Several UID can be grouped together to which a unique identifier can be associated: a GID (`Group IDentifier`). The principle of access management remains similar to UID.

This access model is a DAC (`Discretionary Access Control`). It is discretionary because the user who owns a resource specifies the access rights, with read, write and execute rights each given for:

- the user owning the resource;
- the group owner of the resource;
- the rest of the users.

When a user makes a file or directory, its access rights are calculated from the UMASK (`User file creation mode MASK`) of the creating process. This UMASK is by default at the value `0022`, which is very permissive. This one will indeed assign the access rights `0644` to the file created (read access for anyone) and the access rights `0755` if it is a directory (read and execution for anyone). Thus, it is recommended to change the default value of UMASK to a more restrictive value.

It is not possible to define this value system-wide, but it is possible to do it for each component. It is important to modify it for the user shells and for some of the system services.

R36

Changing the default value of UMASK

The default value of UMASK for the shells must be set to `0077` in order to allow read and write access to its owner only. This value can be defined in the configuration

file `/etc/profile` that most shells (bash, dash, ksh…) will use.

The default value of UMASK for services must be determined for each service, but in most cases, it should be set to `0027` (or more restrictive). This allows read access to its owner and its group, and a full access to its owner. For services such as `systemd`, this value can be defined directly in the configuration file of the service with the directive `UMask=0027`.

The DAC access model is still the majority today and remains applied by default under the Unix systems and derivatives. However, it has important limitations:

- rights are at the owner's discretion, which may not be suitable for environments constrained by a security policy;
- the owner may not be able to provide adequate rights to his resources, which offers access to confidential data (or even compromises);
- the isolation between users is coarse, the default rights are often too laxist;
- there are only two levels of privileges on the system: "regular" user accounts and the administrator account (or `root`). Changing users requires the use of executables with special rights, for example `setuid root`;
- it does not allow to withdraw special privileges to a process according to the context: the web browser and the word processor from the same user will have as many privileges on the resources;
- the attack surface is large, a standard user has access to a lot of information about the underlying operating system.

Other access models, such as MAC (`Mandatory Access Control`), offer more possibilities, but at the cost of higher investment and complexity.

R37

Using `Mandatory Access Control` features

It is recommended to use the `Mandatory Access Control` (MAC) feature in addition to the traditional Unix user model, `Discretionary Access Control` (DAC), or possibly combine them with partitioning mechanisms.

The limitations of the DAC have contributed to the emergence of rights delegation management tools, the best known being `sudo` which is a utility installed when there is a need to delegate rights and privileges to different users. This delegation is based on the possibility for a given user to execute a previously defined command with the privileges of another user. In order to achieve this, `sudo` is an executable `setuid root`. It is important to be concerned about its security on two levels:

- at the level of hardening, to prevent a malicious user to exploit the vulnerabilities of the binary;
- at the configuration level, where giving a user the right to perform specific commands may give him more privileges and prerogatives than initially necessary.

The sudo operation relies heavily on the traditional Unix model to work, which is enriched with a finer transition logic than the one originally allowed by su. The items below are intended to give some recommendations and ideas to explore in order to use and to configure sudo so that it offers the least possible circumvention.

These recommendations apply equally for cases where sudo is used for the execution of privileged commands by a "regular" user, or the restriction of privileges when a privileged user (such as root) wants to run a command with lower rights, as a precautionary measure.

sudo is usually installed by default with open rights and allows any user to use it (execution right for everyone). sudo being a privileged executable and complex by its configuration, it is better to restrict its execution rights to a dedicated user group. This reduces the attack surface, especially when the system and the binary itself are victims of exploitable vulnerabilities, for example CVE-2019-14287 (potential bypass of Runas user restrictions [17]) ou CVE-2021-3156 (buffer overflow in command line unescaping [18]), by any user.

R38

Creating a group dedicated to the use of sudo

A group dedicated to the use of sudo must be created. Only the members of this group should have the right to run sudo.

Example of creating a dedicated sudo group

A possible case is to create a dedicated group, for example sudogrp, which is assigned the rights to run sudo. Only the members of this group will be able to call it:

```
# ls -al /usr/bin/sudo
-rwsr-x---. 2 root sudogrp [...] /usr/bin/sudo
```

Warning

This modification can be overwritten by an update of the sudo package. Thus, it is recommended to check the access rights of the file after an update and apply then again if needed. This operation can be automated with most package managers [19].

sudo is configured using the /etc/sudoers file with the visudo command. It contains a set of directives that will allow sudo to know the commands a user is allowed to run or not, this may be based on the machine hostname.

It is particularly difficult to provide a set of recommendations that can cover all situations where sudo can be used. Indeed, the number of available commands, the architecture of the information system, the installed services and their configurations, makes the establishment of a secure standard configuration almost impossible.

However, some good practices must be followed in order to avoid, as much as possible, configuration errors that could lead to circumventions of the security policy. The default values of the

17. https://www.sudo.ws/alerts/minus_1_uid.html
18. https://www.sudo.ws/alerts/unescape_overflow.html
19. https://manpages.debian.org/bullseye/apt/apt.conf.5.en.html

directives of the `sudo` configuration should be set to secure values, and then overridden later if needed on a case-by-case basis. The reading of the `sudoers` man page is highly recommended, especially the part dealing with shell escapes.

R39

M I

Sudo configuration guidelines

The following guidelines must be enabled by default:

noexec	applies the **NOEXEC** tag by default on the commands;
requiretty	requires the user to have a `tty` login;
use_pty	uses a `pseudo-tty` when a command is executed;
umask=0077	forces `umask` to a more restrictive mask;
ignore_dot	ignores the "." in `$PATH`;
env_reset	resets the environment variables;

Guidelines for the `/etc/sudoers` configuration file

The list of directives are presented as encountered in the configuration `/etc/sudoers`:

```
Defaults noexec,requiretty,use_pty,umask=0077
Defaults ignore_dot,env_reset
```

Then, it is possible to override a default value if needed when an entry is added for a user (1) or for a group (2).

```
myuser ALL = EXEC: /usr/bin/mycommand    # (1)
Defaults:%admins !noexec                 # (2)
```

The remaining of the file is then predominantly composed of rules allowing to declare, for a given user or group the list of commands he is allowed to execute, all helped with alias specifications (`User_Alias`, `Runas_Alias`...) when the right management policy becomes complex. The verification of the right to execute or not a command is based on a string comparison between the command desired by the user and the specification present in the `sudoers` configuration file. The simplest model is the following:

```
user    hostname = ( user-target ) command, [...]
```

It is common that the command to be executed does not require an elevated privilege level (editing a file whose owner is not root, sending a signal to an unprivileged process,...). In order to limit any attempt of privilege escalation through a command, it is better to apply regular user rights.

R40

M I

Using unprivileged users as target for `sudo` commands

The targeted users of a rule should be, as much as possible, non privileged users (i.e.: non `root`).

Functionally rich commands can be executed via `sudo`, like text editors (`vi`). They may allow execution of sub-commands in the environment created by `sudo` and thus enable a user to run other

software with privileges not foreseen initially. This may well lead to circumventions of the security policy, or even privilege escalations.

To this end, `sudo` can override the different functions allowing to execute other softwares. This allows for example to block the trivial creation of a sub-shell through `vi`. It is however important to note that this protection is not perfect and only override the functions allowing to create these new processes. The process remains free to execute the system calls it wants (including `execve`) and so bypass this protection mechanism.

R41

M I E Limiting the number of commands requiring the use of the EXEC directive

The commands requiring the execution of sub-processes, `EXEC` directive must be explicitly listed and their use should be reduced to a strict minimum.

Specifying access right using negation, blacklist approach, is inefficient and can be easily circumvented.

Example

It is expected, for example, that a specification of the following type prevents the execution of the `shell` :

```
# To avoid absolutely , this rule can be easily circumvented!
user   ALL = ALL,!/bin/sh
```

That's not the case: just copy the binary `/bin/sh` to a different name to make it executable again through the rule directive `ALL`.

R42

M I Banishing the negations in `sudo` policies

Policies applied by `sudo` through the sudoers file should not involve any negation.

To determine whether a command is executable by a user, `sudo` performs a string comparison between the desired command and the corresponding specifications in the `sudoers` following the rules of the `globbing shell`; this evaluation includes the arguments.

Any argument can modify quite significantly the behavior of a program, whether regarding the realized operation (read, write, delete,...) or accessed resources (path in a file system tree). To avoid any possibility of misuse of a command by a user, the ambiguities must be removed at the level of its specification.

R43

M I Defining the arguments in `sudo` specifications

All commands in the `sudoers` configuration file must strictly specify the arguments allowed to be used for a given user.
The use of `*` (`wildcard`) in the rules should be avoided as much as possible. The absence of arguments in a command must be specified by the presence of an empty

chain (`""`).

Example

On some systems, the kernel events are only accessible by `root`. If a user nevertheless must have the privileges to read them, it is necessary to restrict his arguments in order to prevent the user from reading any arbitrary file through the `--file` option:

```
user   ALL = dmesg ""
```

Warning

It is important to note that allowing an unprivileged user to run under the `root` identity a software susceptible of making an arbitrary write (made possible by a laxist rule) is somehow giving `root` privileges to this user because this one could, by different methods (handling of password files, modification of `setuid root` binary,...) obtain the `root` privileges without the access control and enforcement that `sudo` allows.

It is quite common to allow a user to edit a file through `sudo` by calling directly a text editor. A text editor is a software functionally rich, that can open and edit other files even by internal commands or even run a shell script. Some editors may offer a full privileged environment to a user. The `sudoedit` text editor solves these problems by editing a temporary version of the file with the current user rights, then replacing the original file by this temporary version once the operation is finished. The editor runs thus only with the current user rights and never with the privileges of the target user.

R44

Editing files securely with `sudo`

A file requiring `sudo` to be edited must be edited through the `sudoedit` command.

6.3.2 AppArmor

`AppArmor` is one of the LSM frequently encountered on GNU/Linux, and the one used by default by variants of SUSE [20] (including OpenSUSE [21]) and Ubuntu [22]. It is a MAC access model where an authority decides the access of an application without giving it the capability to alter it.

Its documentation is directly accessible on the project website [11]. Its configuration is based on specifications of rights based on paths in the file system tree, and applied for each executable. Thus, it is possible to implement `AppArmor` profiles for specific applications without having to care about the interactions between the LSM and the other applications or the system. This makes it relatively easy to learn and integrate into a system.

In addition to the access specifications, `AppArmor` allows to block the use of `POSIX` capabilities, and also restrict the network use (`network` directive). These restrictions, however, remain basic and

20. https://www.suse.com
21. https://www.opensuse.org
22. https://ubuntu.com

do not have the richness of expression of `iptables` or `nftables`[23] which are system utilities for configuring firewall rules under GNU/Linux.

When an executable under `AppArmor` calls another executable, its security profile allows to declare how the transition should take place (inheritance of the current profile, activation of another security profile, exit from confinement,…)

Although simple to understand for an administrator accustomed to the traditional Unix access model, `AppArmor` does not allow attaching security labels to a flow or to messages. It is a framework that essentially focuses on the specification of access and privileges for executable, the concept of a security label is absent.

Many of the GNU/Linux distributions that use it provide by default, generally for most sensitive executables (`syslogd`, `ntpd`…), `AppArmor` profiles in the directory `/etc/apparmor.d/`, a file per executable. Operations, including denied access, are recorded by `auditd` detailed in paragraph 7.2.3 Profiles can be enabled in `enforce` or `complain` mode. In the `complain` mode, the actions that would have been blocked are recorded without being really blocked.

R45

Activating `AppArmor` security profiles

Any `AppArmor` security profile present on the system must be enabled in `enforce` mode by default when the distribution supports it and does not use any other security module than `AppArmor`.

Information

On recent GNU/Linux distributions, `AppArmor` is enabled by default. To make sure that `AppArmor` is active, use `aa-status`:

```
# aa-status
apparmor module is loaded.
40 profiles are loaded.
23 profiles are in enforce mode.
...
14 processes have profiles defined.
12 processes are in enforce mode.
...
0 processes are unconfined but have a profile defined.
```

In particular, it is necessary to verify that the processes currently running on the system and for which a profile is declared are confined in `enforce` mode by `AppArmor`.

6.3.3 SELinux

`SELinux` (for `Security-Enhanced Linux`) is a widely used LSM in the distributions derived from `Red Hat`.

`SELinux` relies on the use of “labels” assigned to objects (files, processes, sockets,…) that allow, depending on the domain to which a process is confined, to authorise or deny access to a resource.

23. https://www.netfilter.org

Information

At the file system level, `labels` are stored as extended attributes and can be obtained via the following command:

```
# ls -Z
```

Those associated to processes are visible with the command:

```
# ps -Z
```

The use of "labels" makes it possible to declare "domains" of responsibility and therefore to know at every moment the operations that a process can perform on a given resource. `SELinux` also allows to define transitions from one domain to another when a user requires specific access privileges. The granularity of access is much finer than in `AppArmor`, to the detriment of a greater complexity.

The `SELinux` documentation is extensive and accessible on the project's wiki page [19].

This section is limited to giving some checks so that any administrator wishing to use the profiles provided by its distribution can do it correctly.

`SELinux` offers the following three operating modes:

- the `disabled` mode in which `SELinux` rules are not loaded and no access message is logged;
- the `permissive` mode in which `SELinux` rules are loaded and queried but no access is denied, and access error messages are logged;
- the `enforcing` mode in which accesses are conditioned by `SELinux` rules.

The `SELinux` policies available by default on distributions are:

minimal	(`minimal`) policy which confines only a very limited number of services;
targeted	(`called targeted or default`) policy partitioning each system service into a domain. Interactive users are not partitioned;
multi-level	(`mls`) policy including targeted policy and which additionally proposes the use of hierarchical classification levels (sensitivity).

The use of the `minimum` policy is discouraged because the elements of the system are not confined into separate domains. According to its creators, this policy should be used in constrained environments (little memory) or for tests [18].

The `multi-level` policy was created to handle information of different sensitivities on the same system while ensuring the principle of "need to know". It is important to note that it does not allow processing information of a particular sensitivity level on systems with lower or incompatible sensitivity levels (see document [10] for more information). Its operation will not be discussed here.

The remainder of this section will focus on the default `targeted` policy. Each system service is confined into a separate domain. Multiple instances of the same service running in the same domain can be separated using categories. Interactive users are by default associated with an

unconfined domain (`unconfined_t`) which does not impose additional restrictions. In this case, the access control model remains very close to the DAC.

R46

Activating `SELinux` with the `targeted` policy

It is recommended to enable `SELinux` in `enforcing` mode and to use the `targeted` policy when the distribution supports it and does not use any other security module than `SELinux`.

Information

Activating SELinux with the targeted policy generally involves the following steps:

1. Check the value of the SELINUX and SELINUXTYPE variables in the /etc/selinux/config configuration file:

```
# grep ^SELINUX /etc/selinux/config
SELINUX=enforcing
SELINUXTYPE=targeted
```

2. If the value of SELINUX is not enforcing or if the value of SELINUXTYPE is not targeted (or default), edit the file to correct them.

3. If SELinux was not in enforcing mode, it is necessary to verify that SELinux labels are correctly assigned to files system-wide. This check can be programmed to be executed automatically at the next start with the command:

```
# fixfiles onboot
```

4. If the policy used was not the targeted policy, reload the policy used by the kernel with the command:

```
# semodule -R
```

5. It may be necessary to reboot the system for the policy change or SELinux activation to take effect.

6. Finally, to make sure that SELinux is active and with the correct settings:

```
# sestatus
SELinux status:                 enabled
...
Loaded policy name:             targeted
Current mode:                   enforcing
...
```

By default, the rights of interactive users are not restricted in the targeted policy. But it is possible to selectively confine users who do not need to perform administrative actions on a system.

Example

The following command associates an interactive user with the SELinux user:

```
# usermod -Z user_u <utilisateur>
```

R47

M I E H

Containing the unprivileged interactive users

Unprivileged interactive users of the system should be confined by associating them with a confined SELinux user.

The behavior of a security policy can be modified through boolean variables. Some are useful from a security standpoint.

R48

M I E H

Setting up the SELinux variables

It is recommended to apply the following security policies:

- Prevent processes to make their `heap` executable;
- Prevent processes to have a memory area with rights `w` (write) and `x` (execute);
- Prevent process to make their `stack` executable;
- Prevent dynamic loading of modules by any process;
- Prevent SSH logins to connect directly in sysadmin role.

Example

For distributions derived from `Debian`, modifying these security policies involves modifying the default values of the following variables with the `setsebool` command:

```
#setsebool -P allow_execheap=off
#setsebool -P allow_execmem=off
#setsebool -P allow_execstack=off
#setsebool -P secure_mode_insmod=on
#setsebool -P ssh_sysadm_login=off
```

There are many others. The `getsebool -a` or `semanage boolean --list` commands can be used to obtain information about the boolean variables defined and their current state in the policy used on a system.

Information

A large part of these variables are also documented in the CentOS wiki [17].

Like `AppArmor`, actions are recorded by `auditd` detailed in paragraph 7.2.3.

The use of `SELinux` policy manipulation and debugging tools should be limited to development machines, on which the interpretation of an error reported in the `SElinux` audit logs can be performed.

R49

M I E H

Uninstalling SELinux Policy Debugging Tools

`SELinux` policy manipulation and debugging tools should not be installed on a production machine.

Warning

The `setroubleshootd` service must be disabled and the following packages uninstalled:

- `setroubleshoot`;

- `setroubleshoot-server`;
- `setroubleshoot-plugins`;

6.4 Files and directories

Particular attention must be paid:

- to files and directories containing secret items, such as passwords, password fingerprints, secret or private keys;
- to named IPC (`Inter-Process Communication`) files, such as *sockets* or *pipes*, which allow different processes to establish communication between them;
- to temporary storage directories that everyone has access to;
- executables with special rights, such as `setuid` or `setgid`;

Appropriate rights will have to apply rationally to each of these types.

6.4.1 Sensitive files and directories

Files and directories containing secret or private keys, passwords, fingerprints …(or even logs) are considered sensitive files or directories.

When these files contain system passwords or system password fingerprints, they should only be readable by `root`. On the other hand, public files which contain the list of users are readable by everyone, but can only be modified by `root`.

Fichiers sensibles

Some examples of files containing sensitive elements:

```
-rw-r----- root root  /etc/gshadow
-rw-r----- root root  /etc/shadow
-rw------- foo  users /home/foo/.ssh/id_rsa
...
```

Even when these files have their content protected by additional measures (encryption, fingerprints for passwords…), it is nevertheless necessary to restrict access according to the principle of defense in depth.

R50

M I Limiting the rights to access sensitive files and directories

Sensitive files and directories should only be readable by users with strict need to know.

Information

The analysis scheme for sensitive files or directories is as follows:

1. sensitive system files or directories must have as owner `root` in order to prevent any change of rights by an unprivileged user;
2. sensitive files or directories accessible to a user other than `root` (for example, the password base of a Web server) must have as owner that user (for example, the user associated with the Web server) who must be a member of a dedicated group (for example, the `www-group`) and who will have read-only access to that file or directory;
3. the rest of the users should not have any rights to the sensitive files or directories.

All sensitive files, generated or installed during the installation of the system, and those which contribute to authentication mechanisms (root authority certificates, public keys, certificates,...) must be in place as soon as the system is installed.

R51

Changing the secrets and access rights as soon as possible

All sensitive files and those which contribute to the authentication mechanisms must be in place as soon as the system is installed. If default secrets are preconfigured, they must then be replaced during, or immediately after, the installation phase of the system.

6.4.2 Named IPC, sockets or pipes

Applications can exchange information through *sockets*. Most of the time these *sockets* are established at the network level (IP). When it comes to establishing connections between local processes, alternatives exist like *pipes* or local Unix *sockets*.

Care must be taken that the access rights that apply to a local *socket* are those of the directory containing it and not those of the *socket* itself. Although some systems will still honor these permissions, the Portable Operating System Interface (`POSIX`) standard does not impose it.

R52

Securing access for named sockets and pipes

Named *pipes* and *sockets* must be protected by a directory with appropriate rights. The rights on the socket itself must also restrain the access.

Information

Multiple commands allow to get information about *sockets* in use on the system. According to the applied security policy and the loaded modules, the displayed result may not be exhaustive.
The following commands list all *sockets* and give information of associated processes

for local *sockets*:

```
# sockstat
# ss -xp
```

The following command lists the resource usage information for IPC:

```
# ipcs
```

The following command lists all virtual memories shared by processes and their associated access rights:

```
# ls /dev/shm
```

The following command provides detailed information on the `I/O` and IPC for system processes:

```
# lsof
```

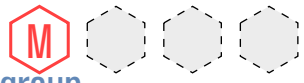
Warning

Local *sockets* should not be created at the root of a temporary directory accessible in writing to everyone.

6.4.3 Access rights

Files or directories without a known user or group may be incorrectly assigned to a user or group when creating a new user account or group. You must therefore make sure that no file or directory is in this situation on the system.

R53



Avoiding files or directories without a known user or group

Files or directories without a user or group identifiable by the system must be analyzed and possibly corrected in order to have a known owner or group of the system.

Information

The following command lists all the files that no longer have an associated user or group:

```
# find / -type f \( -nouser -o -nogroup \) -ls 2>/dev/null
```

The directories accessible to all in writing are in most cases used as temporary storage zones, in which an application saves data for processing. Many applications use them incorrectly. Temporary files can then be diverted from their primary function and be exploited as a rebound, for example for a privilege escalation.

A workaround is then to activate the `sticky bit` on the directory so that only the user or the application that created the file can delete it.

This prevents a user or an application from arbitrarily removing or replacing temporary files for another user or application.

R54

M Setting the `sticky bit` on the writable directories

All directories that are writable by all must have the `sticky bit` applied.

Warning

It is also necessary to ensure that the owner of the directory is `root`, otherwise the owner will be able to modify its content at will, despite the `sticky bit`.

Information

The following command lists all the directories that can be modified by all and without a `sticky bit`:

```
# find / -type d \( -perm -0002 -a \! -perm -1000 \) -ls 2>/dev/null
```

The following command lists all the directories that can be modified by anyone and whose owner is not `root`:

```
# find / -type d -perm -0002 -a \! -uid 0 -ls 2>/dev/null
```

The `sticky bit` does not prevent competitive situations between two applications running simultaneously under the same user account.

R55

M I Dedicating temporary directories to users

Each user or application must have their own temporary directory and dispose of it exclusively.

Information

On recent GNU/Linux distributions, the most direct method to create a temporary directory specific to each user is to use PAM (`Pluggable Authentication Module`) modules, such as `pam_mktemp` and `pam_namespace` detailed in section 7.2.1.

When a file must be editable by several users or applications at the same time, a group must be created and only this group must have write rights on this file.

Information

The following command lists all the files that can be edited by everyone:

```
# find / -type f -perm -0002 -ls 2>/dev/null
```

Executables with the `setuid` or `setgid` special rights are particularly sensitive because they run with the privileges of the owning user or of the group to which the owning user belongs and

not with the privileges of the current user or of the group to which the current user belongs. These executables must be trusted, for example from distribution or official package repositories in accordance with recommendation R59.

The presence of `setuid` or `setgid` special rights on an executable requires a certain number of precautions to guard against vulnerabilities linked to changing users, for example, cleaning up its environment and reinitializing a certain number of elements inherited from the previous context (signaling masks, open file descriptors…). Most executables do not implement such precautions, and adding the `setuid` or `setgid` special permissions to them would therefore introduce the possibility of privileges escalation.

R56

M Avoiding using executables with `setuid` and `setgid` rights

Only trusted software from, for example, distribution or official package repositories and specifically designed for use with `setuid` or `setgid` special permissions can have these permissions set.

Information

The following command lists all the files with the specific rights to the `setuid` and `setgid` present on the system:

```
# find / -type f -perm /6000 -ls 2>/dev/null
```

Information

The following command is used to remove the `setuid` or `setgid` special rights:

```
# chmod u-s <fichier>  # Remove the special setuid right
# chmod g-s <fichier>  # Remove the special setgid right
```

The most common case is the executable whose owner is `root` with `setuid` special rights (`setuid root`) or `setgid` (`setgid root`) which runs with `root` privileges and not those of the calling user. These executables allow an unprivileged user to perform actions for which it has, a priori, no right.

In the presence of vulnerabilities, these executables are exploited to provide a `root` shell to a malicious user, or at least to hijack the legitimate use of the executable. These executables are therefore to be studied on a case-by-case basis.

R57

M I E Avoiding using executables with `setuid root` and `setgid root` rights

The executables with the `setuid root` and `setgid root` special rights should be as few as possible. When only administrators are expected to execute them, these special rights (`setuid` or `setgid`) should be removed and prefer commands like `su` or `sudo`, which can be monitored.

Warning

A check should take place after each update because special rights may have been restored or additional executables may also have been installed during an update.

6.5 Package management

The packages installation is the crucial step that will determine the set of files that will be present on the system, the services it will provide. The installed packages must be maintained over time.

R58

M

Installing only strictly necessary packages

The choice of packages should lead to an installation as small as possible, limiting itself to select only what is required.

Minimalist installation

Sometimes it is easier to achieve a minimalist installation by removing all the pre-selected packages, and to choose only those necessary for the context of use. For example, the implementation of a server does not always require the installation of a local graphical interface (*X server*).

Warning

Some distributions provide pre-configured “roles”. It is not recommended to base an installation on these roles since the choices of the maintainers of the distribution do not match necessarily specific needs, which will require the installation of additional packages.

These packages usually come from servers called repositories which contain a set of packages accessible and installable from a package manager. Each distribution has official repositories configured, by default, in the package manager during installation and which are managed by the maintainers of the distribution. These repositories mainly provide the “base” packages, their update and security updates.

R59

M

Using only official package repositories

Only repositories internal to the organisation, or official (from the distribution or from an editor) should be used.

R60

M I E

Using hardened package repositories

When the distribution provides several types of repositories, preference should be given to those containing packages subject to additional hardening measures. Between two packages providing the same service, those subject to hardening (at com-

pilation, installation, or default configuration) must be preferred.

Warning

In some cases, packages are sourced from servers called repository mirrors and configured in the package manager. These mirrors must be exact copies of official, up-to-date repositories.

6.6 Monitoring and maintenance

Every system must be kept in secure conditions.

Software patches can bring new features to the software environment, thus contributing to increasing its level of security. Others aim to rectify vulnerabilities present on the system. A monitoring activity on the corrective measures to be applied is essential, because it allows to know the vulnerabilities to which it is or has been exposed, and thus to lend special attention until a patch becomes available.

R61

Updating regularly the system

It is recommended to have a regular and responsive security update procedure.

The monitoring activity can be done by subscribing to mailing lists of the security teams of components and applications installed and of their editors, to RSS (`Really Simple Syndication`) feeds of CERT (`Computer Emergency Response Team`), for example CERT-FR website [12].

7
Services configuration

The configuration of an operating system is accompanied by the deployment of services. The following recommendations aim to establish some good administration and configuration practices.

Objective

Identify essential services and appropriate hardening measures to delay as much as possible wide scale compromise of the system.

When installing a distribution, some services are installed by default depending on the choice of the maintainers of the distribution. These services do not necessarily correspond to specific needs.

R62

M ⬡ ⬡ ⬡ Disabling the non-necessary services

Only the components strictly necessary to the service provided by the system should be installed.
Any service, especially in active listening on the network, is a sensitive element. Only those known and required for the operation and the maintenance must be installed. It is recommended to uninstall the services whose presence cannot be justified or to disable them if their uninstallation is not possible.

Information

For distributions under `systemd`, the following command lists all the services installed on the system:

```
# systemctl list-units --type service
```

Information

The services commonly installed on the distributions are:

- autoconfiguration services, such as DHCP [24] or ZeroConf [25], besides the fact that they can disrupt the network on which they are connected, these services can receive illegitimate data. Unless there is an operational need, they should not run on a server;
- RPC services (`portmap`, `rpc.statd`, `rpcbind`...) are used in practice only for a NFS server;

- office services such as `dbus`, `hald`, `ConsoleKit`, `CUPS` or `PolicyKit` should not be running on a server;
- the `avahi` service, used for the publication and automatic discovery of services on the network. Unless there is an operational need, this service should not run on a server;
- the `X server`, rarely useful on a server.

Services are often installed with default configurations that enable features potentially problematic from a security point of view, for example, SSH port forwarding that are often accepted and then used to bypass firewall rules, or an `Apache web` server with directory indexing enabled and allowing to navigate in the directory tree system.

R63

Disabling non-essential features of services

The features configured at the level of launched services should be limited to the strict minimum.

Warning

The default configuration of the following services is often incomplete: `SMTP` (`Exim`, `Postfix`), `NTP` (`ntpd`) and `DNS` (`Bind`).

The Linux kernel subdivides the privileges traditionally associated with the system administrator (or `root`) account in separate units, called `Linux capabilities`[26]. `Linux capabilities` can be enabled or disabled independently and are not limited to processes and are also placed on executable files.

Warning

The kernel currently supports 41 `capabilities`. At least 19 of them allow to elevate trivially your rights to these of the `root` user (`UID=0`) (*full root*)[27]. Most of the others also can help a lot to gain a `root` access on the system.

Information

The following command lists all executable files for which one or more `Linux capabilities` have been enabled

```
# find / -type f -perm /111 -exec getcap {} \; 2>/dev/null
```

As part of the principle of least privilege, it is important to isolate the different services running on the system.

24. Dynamic Host Configuration Protocol
25. Zero-configuration networking
26. https://man7.org/linux/man-pages/man7/capabilities.7.html
27. https://forums.grsecurity.net/viewtopic.php?f=7&t=2522#p10271

R64

Configuring the privileges of the services

All services and executables available on the system must be analyzed in order to know the privileges associated with them, and must then be implemented and configured to use what is strictly necessary.

Information

Some services or executables require to own initially or all the time privileges to start or operate and to access the resources of the system.

7.1 Partitioning

Partitioning is a technique that aims to isolate the processes running on the same system. Historically, it was carried out through the `chroot` utility which allows you to restrict the view of the file system of a process at a given directory which becomes its root. Once confined or `chrooted`, the process can no longer access directories other than the one taken as root as well as its subdirectories. So he only has a partial view of the file system.

`chroot` presents many weaknesses on GNU/Linux, among which:

- inability to forbid a `root` user to escape its jail;
- inability on some operating systems to forbid an unprivileged user to escape its jail with the complicity of an external process;
- limited partitioning to the file system; the access to other processes, to the network, …are not blocked;
- require to have initially `root` privileges in order to be ran.

Other mechanisms have appeared as the kernel evolves and improves Linux to partition the processes according to the following main axes:

- **containment** with the `Linux namespaces` which allow to partition Linux kernel resources in abstract spaces called “namespaces”;
- **filtering** with `seccomp` (`SECure COMPuting mode`) or `seccomp BPF`[28] (`SECure COMPuting with Berkeley Packet Filter`) which filters the system calls of processes;
- **isolation** with `cgroups` (`control groups`) which make it possible to organize processes into hierarchical systems in order to limit, count and isolate the use of resources (sharing of processor, memory limit, disk usage …) of the system by the processes.

These mechanisms can be implemented in a granular fashion from initialization systems (or `init`) of Unix operating systems and derivatives (`systemd`, `OpenRC`[29]…).

28. https://www.kernel.org/doc/html/latest/userspace-api/seccomp_filter.html
29. https://github.com/OpenRC/openrc

R65

Partitioning the services

As part of the principle of minimization, it is recommended to partition the services with the containment, filtering and isolation mechanisms available in the Linux kernel.

Nowadays other partitioning solutions exist and offer different levels of abstraction. They are generally characterized by the partitioning technique:

- with containers, where the kernel is able to handle multiple system instances (`LXC`[30], `Docker`[31], `podman`[32]…);
- with emulation, where an emulator reproduces a full physical machine (`QEMU`[33], `VirtualBox`[34]…);
- with lightweight hypervisor (or `bare-metal`) (`Xen`[35]…), where the hypervisor will manage possibly with the help of a host system different virtual machines;
- with kernel hypervisor (`Linux KVM`[36]…).

These solutions are not exclusive. Some solutions (`LXC`, `Docker`, `podman`…) implement the features of the Linux kernel introduced previously: `Linux namespaces`, `Linux capabilities`, `seccomp` or `cgroups`. Reasonable use should be made keeping in mind that the interfaces they offer are themselves doors through which intrusion is possible, for example a container-based system in which the kernel is compromised will see all the containers themselves compromised. The same goes for virtual machines and their hypervisors.

R66

Hardening the partitioning components

Each component supporting the virtualization must be hardened, especially by applying technical measures to counter the exploit attempts.

7.2 System services

Several tools and services can provide items of information about the system. Many of them are not optimally configured after the installation of the system. The following recommendations aim to fill this gap.

7.2.1 Pluggable Authentication Module

PAM is a set of modules that allow you to “dynamically” configure the different authentication and account management mechanisms on a GNU/Linux system.

30. https://linuxcontainers.org/lxc
31. https://www.docker.com
32. https://podman.io
33. https://www.qemu.org
34. https://www.virtualbox.org
35. https://xenproject.org
36. https://www.linux-kvm.org

PAM's documentation is rich, as are all the features offered by its modules. The purpose of this technical note is not to explain how it works.

PAM will essentially provide the account management service, i.e. allow the authentication of the user, the creation of his session and possibly any operation that must take place when attempting an access:

- environment creation,
- ticket or data recovery,
- access rights,
- password change…

When an application uses PAM to authenticate a user, the operation is directly performed by a PAM module.

In general, the application submits the authentication elements to the PAM modules. Depending on the configuration located in various files present in the `/etc/` and `/etc/pam.d/` directories, PAM returns the result, failure or success, to the application which then grants or denies access to the system.

Two important elements should be noted:

- PAM modules are called by the application. The application manipulates, at least partially, elements that help authenticate the user which includes potentially sensitive data such as passwords;
- depending on the PAM modules used, PAM checks these elements using local databases, for example `shadow`, or remote, for example using LDAP (`Lightweight Directory Access Protocol`), SQL (`Structured Query Language`) or `Kerberos` protocol….

R67

M I

Secure remote authentication with PAM

When authentication takes place through a remote application (network), the authentication protocol used by PAM must be secure (flow encryption, remote server authentication, anti-replay mechanisms,…).

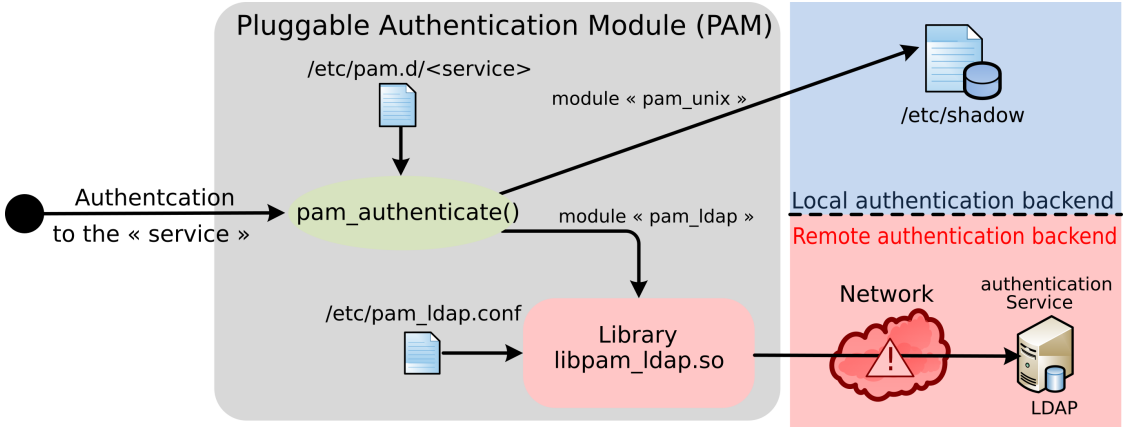


Figure 1 – PAM Unix + LDAP authentication scheme

Information

Some protocols like `Kerberos` offer secure communication functions, for example the `pam_krb5` module if the `keytab host` is specified for the system.
Other modules, such as `pam_ldap` or `pam_mysql`, will not attempt to establish a secure communication between the PAM client and the application if they are not explicitly configured to do so.

In addition to authentication, PAM allows to load other modules whose features can improve the security of the architecture, for example:

pam_faillock allows to temporarily block an account after a certain number of failures;
pam_time allows to restrict access to a time slot;
pam_passwdqc allows to apply constraints following a password complexity policy;
pam_pwquality allows to test the strength of passwords;
pam_wheel allows to restrict access to the `root` account to users who are members of a particular group (default `wheel`);
pam_mktemp allows to provide temporary private directories per user under /tmp;
pam_namespace allows to create a private namespace per user.

Example

The following PAM configuration files are located under `/etc/pam.d/` and are the name of the service to which they are associated. Only the configuration directives concerning are presented.

- `/etc/pam.d/su` and `/etc/pam.d/su-l` to limit the usage of `su` to become `root` to the users part of the group `wheel` only:

```
# Limit the elevation to root via su to the members of the group 'wheel' only
auth    required  pam_wheel.so use_uid root_only
```

- `/etc/pam.d/passwd` to set password complexity rules:

```
# At least 12 characters of 3 different classes among the uppercase letters,
# lowercase letters, numbers and others by prohibiting repetition
# of a character
password  required  pam_pwquality.so minlen=12 minclass=3  \
              dcredit=0 ucredit=0 lcredit=0 \
              ocredit=0 maxrepeat=1
```

- `/etc/pam.d/login` to `/etc/pam.d/sshd` sshd to automatically block accounts:

```
# Account blocking for 5 min after 3 failures
auth    required  pam_faillock.so deny=3 unlock_time=300
```

The storage of passwords in plaintext is forbidden because the compromise of the system allows an attacker to reuse them effortlessly towards other services. Their protection must not only rely on access rights to a password database.

R68

Protecting the stored passwords

Any password must be protected by cryptographic mechanisms to avoid exposing them to an attacker. To do so, see the section *4.6 Stockage de mots de passe* of the guide *Recommandations relatives à l'authentification multifacteur et aux mots de passe* [9].

Information

PAM can be configured to use `yescrypt` by adding in the file `/etc/pam.d/common-password` the following directive:

```
password    required    pam_unix.so obscure yescrypt rounds=11
```

If the system is outdated and does not support `yescrypt`, it is recommended to use `SHA-512crypt` and increase the number of rounds:

```
password    required    pam_unix.so obscure sha512 rounds=65536
```

For more information, refer to `man crypt(5)`.

7.2.2 Name Service Switch

NSS (`Name Service Switch`) is the system layer that handles and queries administrative databases. There are several: `passwd`, `group`, `hosts`, `services`…Only the management databases of the user accounts will be studied below.

When user accounts are stored in an external directory (frequently LDAP), NSS will manage the requests to the directory in order to make the accounts visible from the system. Generally, these requests are anonymous and carried out with an unprotected communication channel. It is therefore possible for an attacker to retrieve a list of valid accounts from the directory or even to spoof the directory server queried by the NSS.

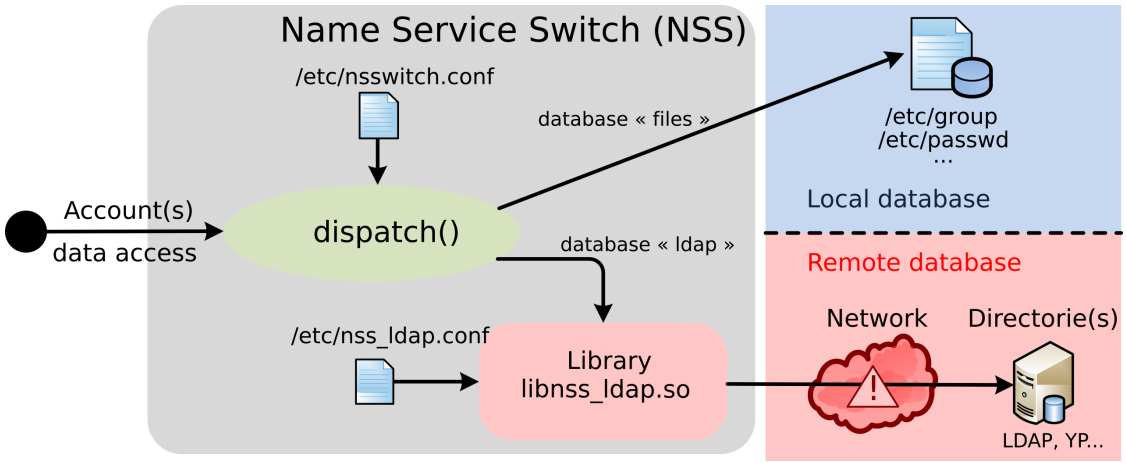


Figure 2 – NSS Administrative Databases Diagram

R69

Securing access to remote user databases

When the user databases are stored on a remote network service, NSS must be configured to establish a secure link that allows, at minimum, to authenticate the server and protect the communication channel.

To access certain administrative databases and more particularly the ones stored on a remote network server, NSS must authenticate using an account. In this case, the account used to access this database must be separate from system user accounts and he should only have the rights strictly necessary to query this database.

R70

Separating the system accounts and directory administrator

It is recommended not to have overlapping accounts between those used by the operating system and those used to administer the directory.
Using directory administrator accounts to perform enumeration queries accounts by NSS must be prohibited.

7.2.3 Logging

As part of the principle of defense in depth, any system must be monitored in order to detect suspicious activity and to be able to react to it.

R71

Implementing a logging system

It is recommended to apply the recommendations of the technical note "*Recommandations de sécurité pour l'architecture d'un système de journalisation*" [7].

The `syslog` service that is the main logging system used under GNU/Linux can be broken down into two parts:

- a server (like `syslog-ng`[37] or `rsyslog`[38]), that collects all the system messages in the `syslog` format;
- several clients that send messages to the server, mostly through the `syslog()` function of the `glibc`.

A `syslog` server is therefore a unifying element of logging and accesses a large amount of data from various system sources. Any service or component may request it in order to record a message, without authentication.

37. https://www.syslog-ng.com
38. https://www.rsyslog.com

Information

It is customary to have a resident `syslog` server that only handles messages submitted `locally` by the components of the system services and which possibly sends these events to a centralized server.

Log security must meet the following two levels of protection:

- between services, so that a service cannot manipulate nor access logs recorded by another service;
- at the service level, so that in the event of a compromise, it cannot read, erase or alter recorded events.

R72

Implementing dedicated service activity journals

Each service must have a dedicated event logging journal on the system. This log must only be accessible by the `syslog` server, and must not be readable, editable or deletable by the service directly.

Information

The use of the function `syslog()` from the `glibc` is a possible solution. Sending messages can also be done through the `logger` [39]command.

The `auditd` service is an advanced logging service that is often present on GNU/Linux distributions. It allows to record specific system operations, even to alert an administrator when unplanned privileged transactions take place.

The recorded events depend on the `auditd` rules written. When a registration is triggered, a second service, `audispd`, will take care of its treatment: `syslog` event, sending mail, writing to a file, …

The `auditd` service is able to monitor a large number of actions:

- system calls made;
- access to a specific tree or files;
- module insertions.

See its documentation for more details.

R73

Logging the system activity with auditd

The logging of the system activity must be done through the `auditd` service.

39. https://man7.org/linux/man-pages/man1/logger.1.html

Example of `auditd` configuration

The following `auditd` configuration file `/etc/audit/audit.rules` records actions that can be of interest:

```
# Run of insmod, rmmod and modprobe
-w /sbin/insmod   -p x
-w /sbin/modprobe -p x
-w /sbin/rmmod    -p x
# On recent GNU / Linux distributions, insmod, rmmod and
# modprobe are symbolic links of kmod
-w /bin/kmod      -p x

# Log changes in /etc/
-w /etc/ -p wa

# Mounting / unmounting monitoring
-a exit,always -S mount -S umount2

# Suspicious x86 syscalls calls
-a exit,always -S ioperm -S modify_ldt
# Syscalls calls that must be rare and closely monitored
-a exit,always -S get_kernel_syms -S ptrace
-a exit,always -S prctl

# Added monitoring for creating or deleting files
# These rules can have important consequences on the
# system performance
-a exit,always -F arch=b64 -S unlink -S rmdir -S rename
-a exit,always -F arch=b64 -S creat -S open -S openat -F exit=-EACCES
-a exit,always -F arch=b64 -S truncate -S ftruncate -F exit=-EACCES

# Added monitoring for loading, changing and unloading kernel modules
-a exit,always -F arch=b64 -S init_module -S delete_module
-a exit,always -F arch=b64 -S finit_module

# Locking the auditd configuration
-e 2
```

Information

The logs thus created by `auditd` can be verbose especially when many activities are reported. The `aureport` tool allows you to select interesting information depending on very specific criteria:

- context on authentication failures;
- event report on specific files or directories;
- abnormal events (software crashes, reconfiguration of network cards…);
- …

7.2.4 Messaging

Messaging is a service commonly used by the system to inform about certain evolutions of its states. This is usually done by sending an email message to a recipient, often a user account.

Information

A frequent case is the `cron` service, which systematically submits an email when the executed task displays data on its error output (`stderr`).

Depending on the distributions, the installed mail service may be configured as an open relay (accepts any mail that is submitted to it), but with the listening *socket* only linked to the local loop (or `loopback` abbreviated `lo`) network interface. If possible, prefer to use a light mail redirection service (such as `ssmtp`).

R74 M I

Hardening the local messaging service

When a mail service is installed on the machine, it must be configured so that it accepts only:

- mails to a local user of the machine;
- connections through the local loop network interface (remote connections to the mail service must be rejected).

Services using messaging are often configured by default to send messages to destination of the system administrator (or `root`) account. In accordance with the recommendation R34, this account must be disabled. Therefore, and in order not to lose any messages addressed to the system administrator, the use of "aliases" is recommended.

R75 M I

Configuring aliases for service accounts

For each service running on the machine, as well as the `root` account, an email alias to an administrator user must be configured to receive notifications and reports sent via email.

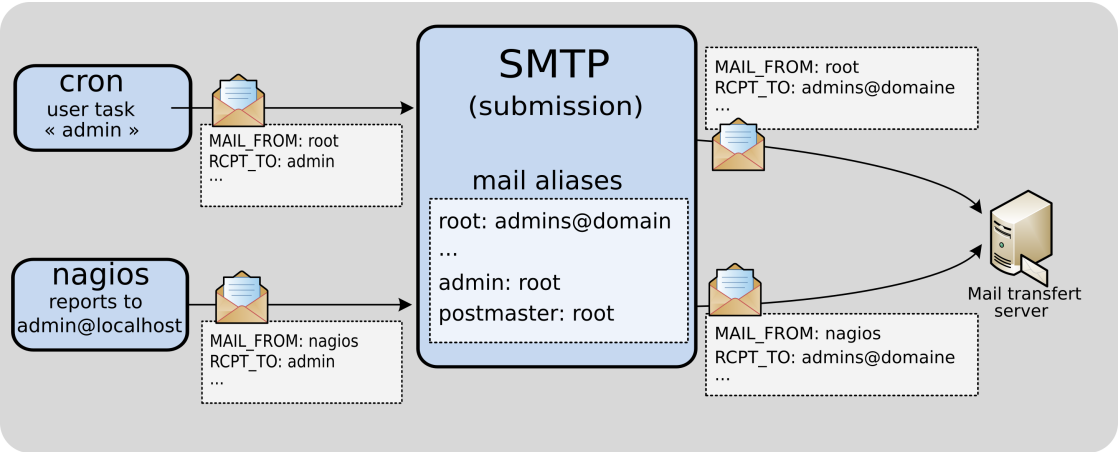


Figure 3 – Sending alert emails to the administration team

7.2.5 File System monitoring

The sealing operation of a file system consists of the installation and then the configuration of a service whose function will be to check, at least periodically, the changes made at the level of a file system tree. This is a feature found within most HIDS (`Host Intrusion Detection System`). Sealing makes it possible to push up information about the changes (discrepancies between the sealed version and the version currently present on the system).

This is a useful function for administrators. In addition to allowing a periodic audit on the system and reports generation, it allows teams to be informed about changes and thus to obtain an evolution follow-up.

R76

Sealing and checking files integrity

Any file that is not transient (such as temporary files, databases, …) must be monitored by a sealing software. This includes among others: directories containing executables, libraries, configuration files, as well as any files that may contain sensitive elements (cryptographic keys, passwords, confidential data…).

The compromise of a machine may be accompanied by a compromise of the sealing database when it is stored locally, technical measures must be implemented to ensure that the integrity of the database content remains preserved.

R77

Protecting the sealing database

The sealing database must be protected from malicious access by cryptographic signature mechanisms (with the key used for the signature not locally stored in plaintext), or possibly stored on a separate machine of the one on which the sealing is done.

There are many solutions, the most deployed on GNU/Linux systems are Tripwire, Samhain [40] et AIDE (`Advanced Intrusion Detection Environment`) [41].

7.3 Network services

Particular attention should be paid to services exposed to uncontrolled flows, that is to say any service communicating with sources that are not trusted (Internet, public `Wi-Fi`, etc.) or not authenticated. These are the assets which are preferentially attacked and compromised because their access is free.

They are all open doors on the system allowing illegitimate access to it when the service is vulnerable or misconfigured: website to execute arbitrary commands, administrative process that does not use a reliable authentication mechanism, obsolete network service with exploitable vulnerability, …

R78

Partitioning the network services

Network services [42]should as much as possible be hosted on isolated environments [43]. This avoids having other potentially affected services if one of them gets compromised under the same environment.

40. `https://www.la-samhna.de/samhain`
41. `https://aide.github.io`

The partitioning of environments can be achieved in different ways (dedicated user, account dedicated container, virtual or physical machine, …), each way has Pros and Cons that need to be studied: a dedicated user account can always access resources on which rights are incorrectly set (a fairly fine audit of access rights must therefore be done), whereas containers and virtual machines offer a more efficient partitioning but can require additional integration efforts.

These services must therefore be hardened and monitored, despite the apparent authentication operation performed by the server. Hardening is a combination of technical measures that aim to delay or even prevent the compromise of the service. This approach applies from the design phase (privilege separation study, unambiguous specifications, …) to the implementation (validation of inputs/outputs, secure configuration, …) and the maintenance.

R79

Hardening and monitoring the exposed services

Services exposed to uncontrolled flows must be hardened and particularly monitored.
Monitoring consists of characterizing the behavior of the service, and reporting any deviation from its nominal operation deduced from the initial expected specifications.

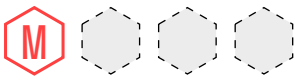
Example

A Web service based on basic `HTTP` authentication may be exploitable on different levels:

1. hardware (firmware, network card drivers…);
2. network (`Ethernet`, `IP`, `TCP`);
3. application (`SSL/TLS` layer, `HTTP` headers issued and received before the authentication).

Network services are often configured by default to listen on all interfaces network, unnecessarily increasing their attack surface.

R80



Minimizing the attack surface of network services

All network services must be listening on the correct network interfaces.

Information

The following command lists all the services listening on the network:

```
# sockstat
```

42. The terminology “service” is to be taken broadly here. Any flaw or exploitable vulnerability before an authentication step is particularly concerned with this recommendation.
43. Environment is to be taken in the broad sense: it represents all the resources and data available and accessible by the service according to the privileges it has.

Recommendation List

Appendix A
Kernel Patches

A number of intrusive patches exist to add security functionality to the Linux kernel. These patches are kept out of the main tree (or upstream) mainly because of their intrusiveness and the strong impact that certain features can have on performance. This appendix does not claim to be exhaustive, but rather aims to present some "examples" of features that can be found in some of them.

The Kernel Self Protection Project [44] (KSPP) focuses on providing functionality directly in the kernel that is acceptable for all machines and all uses.

The grsecurity [45] project, as well as the PaX it integrates, has for many years been the benchmark in terms of kernel hardening. This extremely intrusive and complex patch was never considered for inclusion in the upstream kernel and is no longer publicly available. It is therefore not covered here. Several new projects have emerged to fill this void.

The linux-hardened project is one example. The security features provided are complements to what the KSPP project provides and most of these are taken from PaX:

- improvements to ASLR (Address Space Layout Randomisation);
- post deallocation garbage collection options;
- the declaration of many __ro_after_init variables, which are therefore no longer writable after the kernel initialization phase;
- more restrictive default configurations than the upstream kernel;
- access restrictions to the perf infrastructure;
- some limitations on using user namespaces and connecting new USB devices.

Information

The extra_latent_entropy recommended configuration option for patches provided by the linux-hardened project is to be added to the list of kernel parameters at boot time in addition to those already present in the bootloader configuration file. It allows a very simple form of latent entropy extracted during system startup and added to the entropy obtained with GCC_PLUGIN_LATENT_ENTROPY.

44. https://kernsec.org/wiki/index.php/Kernel_Self_Protection_Project
45. https://grsecurity.net

Information

The recommended configuration options for the patches provided by the `linux-hardened` project detailed in this list are used to complete the kernel configuration in section 5.2.2. They are presented as encountered in the `sysctl.conf` configuration file:

```
# Prohibits the addition of any new USB device. This protection is to
# consider carefully because it is both effective and very impactful, for example,
# when disconnecting a USB keyboard or mouse.
# This option does not protect against all possible attacks on the USB
# as long as the port can continue to be powered.
kernel.deny_new_usb = 0,
# It is important to note that the semantics of this option with the value 3
# is modified by applying the Linux -hardened patch (cf. article LWN -
# https: //lwn.net/Articles/696216/ for a discussion on this)
kernel.perf_event_paranoid=3,
# Restrict the handling of TTYs potentially shared between a
# hierarchy of processes whose privileges are sometimes heterogeneous.
kernel.tiocsti_restrict=1
# Attenuates the risk of attack by time measurement (timing side channel).
kernel.device_sidechannel_restrict=1
```

Information

The list below presents the recommended options for compilation of the patches provided by the `linux-hardened` project detailed in this list allow to complete the static configuration of paragraph 5.3:

```
# Write canaries at the end of SLAB allocations
CONFIG_SLAB_CANARY=y
# Use maximum random bits in mmap base addresses on
# x86_64 processors. This option also affects the number of bits used
# for the base addresses of the stack;
CONFIG_ARCH_MMAP_RND_BITS=32
# Forbid non-privileged users to use TIOCSTI ioctl for
# execute arbitrary commands in processes that share a
# session tty.
CONFIG_SECURITY_TIOCSTI_RESTRICT=y
# Check that the new pages and the allocated slabs are initialized to 0
# in order to detect typical "write -after -free" bugs
CONFIG_PAGE_SANITIZE_VERIFY=y
CONFIG_SLAB_SANITIZE_VERIFY=y
```

In the same way as `linux-hardened`, we can mention `Lockdown Linux` which is a suite of patches whose purpose is to lock down access to kernel data from the userland. This suite of patches was added in Linux 5.4 [46].

Here are some features provided by this patch suite:

- all modules must be signed (it is necessary to activate the kernel compilation option `CONFIG_MODULE_SIG`, otherwise unsigned modules can be loaded);
- no use of `ioperm`, `iopl`, or writing to `/dev/port`;
- no writing to `/dev/mem`, nor to `/dev/kmem`;
- no hibernation;
- restricted access to the configuration registers of PCI devices;

46. https://man7.org/linux/man-pages/man7/kernel_lockdown.7.html

- restricted access to MSR configuration registers for x86 processors;
- no `kexec` ;
- certain restrictions on ACPI;

Information

The list below shows the recommended build options associated with the patches provided by `Lockdown Linux`:

```
# Activate the Lockdown Linux patches from the start of the boot,
# restricting access to features that would allow the user to
# modify the running kernel
CONFIG_SECURITY_LOCKDOWN_LSM=y
CONFIG_SECURITY_LOCKDOWN_LSM_EARLY=y
# Activate privacy mode which extends the above restrictions to
# features that would allow the user to extract information
# confidential contained inside the kernel.
CONFIG_LOCK_DOWN_KERNEL_FORCE_CONFIDENTIALITY=y
```

Finally, the `CLIP OS` [13] project developed by ANSSI offers a set of Linux configurations as well as various hardening patches.

As can be seen from these examples, the security features provided by external patches to the `upstream` kernel are diverse and do not overlap. Cascading multiple patches is a difficult task as they are developed independently of each other and the changes they make to the kernel can conflict with each other.

Warning

The choice of a possible hardening to add to the kernel must be made with regard to the intended use of the machine for which the kernel is compiled.

Appendix B
Configuration Compliance

The SCAP (Security Content Automation Protocol) is a standard maintained by the NIST (National Institute of Standards and Technology) to allow automation of configuration compliance and vulnerability scanning that is interoperable and efficient. Use of SCAP scanners together with SCAP content can lower the barrier and help maintain compliance.

OpenSCAP [47] is the only NIST-certified open source scanner implementation of the SCAP standard. ComplianceAsCode/content [48] is a project that provides security content in various formats, including SCAP, Ansible [49] and Bash. It provides SCAP profiles aligned with the hardening levels presented in this guide and are compatible with distributions derived from Debian and Red Hat. A list of available security policies and documentation on how to use them can be found at https://static.open-scap.org.

Other initiatives have taken place to make recommendations on the configuration of the Linux kernel.

The Recommended Settings [50] page of the KSPP project provides recommendations for achieving a hardened configuration. These recommendations only concern the upstream options of the kernel, they constitute a sound basis without substituting to the recommendations of this guide.

The kconfig-hardened-check [51] Python script verifies the compliance of a kernel configuration with the Recommended Settings of the KSPP project.

Warning

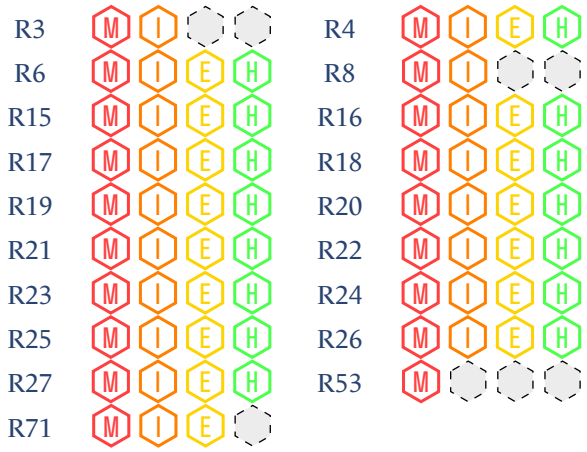
These implementations, while handy, are convenient for checking and are not a substitute for careful and thorough reading with the help of system and security expertise.

47. https://www.open-scap.org
48. https://github.com/ComplianceAsCode/content
49. https://www.ansible.com
50. https://kernsec.org/wiki/index.php/Kernel_Self_Protection_Project/Recommended_Settings
51. https://github.com/a13xp0p0v/kconfig-hardened-check

Appendix C
Changes in the guide

C.1 New recommendations

The following recommendations appear in version 2.0 of the guide:



C.2 Changes between versions 1.2 and 2.0

Besides updates of the style, the updates on the content are the following:

- Redefinition of the hardening levels of the MIEH system in section 1.2 Hardening Level.
- Addition of the chapter 2 Threats and attackers' objectives.
- Redesign of the chapter4 Hardware configuration to add the UEFI secure boot and measured boot.
- Addition of the chapter 5 Linux kernel configuration. It includes the command line options of the kernel and the compilation options of the kernel. The `sysctl` options were also moved to this chapter.
- Addition of the appendix A Kernel Patches.
- Addition of the appendix B Configuration Compliance.
- The MIEH level associated to some recommendations was changed, especially due to the redefinition of the MIEH levels. The table below in section C.3 lists the changes.

C.3 Backward compatibility table

In order to allow readers who have already worked on the basis of the first version of the technical note [4], referred to as v1.2 in the remainder of the text, a backward compatibility matrix is proposed making it possible to find the additions, deletions or equivalences of recommendations.

Warning

This matrix is a tool to facilitate reading but is not intended to establish a strict equivalence between the different versions of the guide. Detailed reading of the updated recommendations is strongly recommended.

v1.2		Current version	
Reference	Level	Reference	Level
R1	M	R62	M
R2	M I	R63	M I
R3	M I E	deletion	
R4	M I E H	R37	M I E
R5	M	deletion	
R6	M I E	R78	M I E
R7	M I E	Included in R71	M I E
R8	M	R61	M
R9	M I	hardware support in R1 Configuration of the BIOS/UEFI in R2	M I E M I
R10	M I	Included in R1	M I E
R11	M I E H	R7	M I E
R12	M I	R28	M I
R13	M I E	R29	M I E
R14	M I	R58	M
R15	M	R59	M
R16	M I E	R60	M I E
R17	M I E	R5	M I
R18	M	Extended to all users in R31	M
R19	M I	R33	M I
R20	M I E	R51	M I E
R21	M I	R79	M I
R22	M I	IPv4 network options in	

v1.2		Current version	
Reference	Level	Reference	Level
		R12 `IPv6` network options in R13	M I
R23	M I	Included in R9 R14	 M I M I
R24	M I E	R10	M I E
R25	M I E	R11	M I
R26	M I E	R30	M
R27	M I	R34	M I
R28	M I E	R35	M I
R29	M I E	R32	M I
R30	M	deletion	
R31	M I	R67	M I
R32	M	R68	M
R33	M I	R69	M I
R34	M I	R70	M I
R35	M I E	R36	M I E
R36	M I	R50	M I
R37	M	R56	M
R38	M I E	R57	M I E
R39	M I	R55	M I
R40	M I	R54	M
R41	M I	R52	M I
R42	M	R80	M
R43	M I	deletion	
R44	M I	deletion	
R45	M I E H	deletion	
R46	M I	R72	M I E
R47	M I	Included in R71	 M I E
R48	M I	R74	M I
R49	M I	R75	M I
R50	M I E	R73	M I E
R51	M I E H	R76	M I E H
R52	M I E H	R77	M I E H
R53	M I E	R64	M I E
R54	M I E	R66	M I E H

v1.2		Current version	
Reference	Level	Reference	Level
R55	M I	R65	M I E
R56	M I	deletion	
R57	M I	R38	M I E
R58	M I	R39	M I
R59	M	deletion	
R60	M I	R40	M I
R61	M I E	R41	M I E
R62	M I	R42	M I
R63	M I	R43	M I
R64	M I	R44	M I
R65	M I E H	R45	M I E
R66	M I E H	R46	M I E H
R67	M I E H	R48	M I E H
R68	M I E H	R49	M I E H
R69	M I E H	R47	M I E H

Bibliography

[1] *Online official documentation of the latest release of the Linux kernel*.
Web page.
https://www.kernel.org/doc/html/latest/index.html.

[2] *Recommandations de configuration matérielle de postes clients et serveurs x86*.
Note technique DAT-NT-024/ANSSI/SDE/NP v1.0, ANSSI, mars 2015.
https://www.ssi.gouv.fr/nt-x86.

[3] *Recommandations pour un usage sécurisé d'(Open)SSH*.
Note technique DAT-NT-007/ANSSI/SDE/NP v1.2, ANSSI, août 2015.
https://www.ssi.gouv.fr/nt-ssh.

[4] *Recommandations de configuration d'un système GNU/Linux*.
Guide DAT-NT-028/ANSSI/SDE/NP v1.2, ANSSI, septembre 2022.
https://www.ssi.gouv.fr/reco-securite-systeme-linux.

[5] *Recommandations pour la mise en œuvre d'un site Web : maîtriser les standards de sécurité côté navigateur*.
Guide ANSSI-PA-009 v2.1, ANSSI, avril 2021.
https://www.ssi.gouv.fr/securisation-sites-web.

[6] *Recommandations relatives à l'administration sécurisée des systèmes d'information*.
Guide ANSSI-PA-022 v3.0, ANSSI, mai 2021.
https://www.ssi.gouv.fr/securisation-admin-si.

[7] *Recommandations de sécurité pour l'architecture d'un système de journalisation*.
Guide DAT-PA-012 v2.0, ANSSI, janvier 2022.
https://www.ssi.gouv.fr/journalisation.

[8] *Recommandations pour la mise en place de cloisonnement système*.
Guide ANSSI-PG-040 v1.0, ANSSI, décembre 2017.
https://www.ssi.gouv.fr/guide-cloisonnement-systeme.

[9] *Authentification multifacteurs et mots de passe*.
Guide ANSSI-PG-078 v1.0, ANSSI, octobre 2021.
https://www.ssi.gouv.fr/mots-de-passe/.

[10] *Instruction générale interministérielle n°1300*.
Référentiel, SGDSN, août 2021.
https://www.ssi.gouv.fr/igi1300.

[11] *AppArmor documentation*.
Web page, GitLab B.V.
https://gitlab.com/apparmor/apparmor/wikis/Documentation.

[12] *French CERT Website (in French)*.
Web page, ANSSI.
http://www.cert.ssi.gouv.fr/.

[13] *Secure multi-level Operating System CLIP OS*.
Web page.
https://www.ssi.gouv.fr/administration/services-securises/clip.

[14] *Analysis of IOMMU service efficiency (in French)*.
Vincent Nicomette Eric Lacombe, Fernand Lone Sang and Yves Deswarte.
Scientific publication, june 2010.
https://www.sstic.org/2010/presentation/Analyse_de_l_efficacite_du_service_fourni_par_une_IOMMU/.

[15] *Licence ouverte / Open Licence v2.0*.
Page web, Mission Etalab, avril 2017.
https://www.etalab.gouv.fr/licence-ouverte-open-licence.

[16] *GRUB Website*.
Web page.
http://www.gnu.org/s/grub/.

[17] *Linux CentOS booleans documentation*.
Web page.
https://wiki.centos.org/fr/TipsAndTricks/SelinuxBooleans.

[18] *Fedora project SELinux policies website*.
Web page.
https://fedoraproject.org/wiki/SELinux/Policies.

[19] *SELinux project Web page*.
Web page.
http://selinuxproject.org/page/Main_Page.

Version 2.0 - 03/10/2022 - ANSSI-BP-028-EN
Licence ouverte / Open Licence (Étalab - v2.0)
ISBN : 978-2-11-167128-7 (papier)
ISBN : 978-2-11-167129-4 (numérique)
Dépôt légal : Juillet 2023

AGENCE NATIONALE DE LA SÉCURITÉ DES SYSTÈMES D'INFORMATION

ANSSI - 51, boulevard de La Tour-Maubourg, 75700 PARIS 07 SP
www.ssi.gouv.fr / conseil.technique@ssi.gouv.fr

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE
Premier ministre
SGDSN