Linux Kernel Hardening with the Yocto Project

Embedded World 2026

Michael Opdenacker

Root Commit

Mar. 10, 2026

root commit

Embedded Linux consultant and trainer

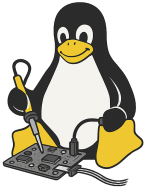
- https://rootcommit.com/about/michael-opdenacker/
- Former founder of Bootlin
- New founder of Root Commit
- Free Software enthusiast and advocate (member of April.org)

April
promouvoir et défendre le logiciel libre

What I Do For a Living — What's Funding This Work

Consulting and engineering work

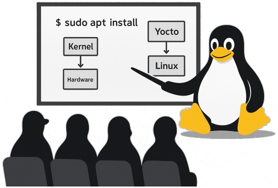
- Yocto Project
  System implementation, porting to new versions, new features
- Linux Kernel
  Driver development, board bring-up, debugging
- Embedded Linux
  Boot time, bug fixing, security and other optimizations



Training — https://rootcommit.com/training

- Yocto Project and OpenEmbedded — Free Materials!
- Linux kernel, board support, driver development
  Free Materials after next course!
- Embedded Linux
- Linux Boot Time Reduction

What's special: focus on practical activities, interactivity and learning techniques. At the heart of the community!



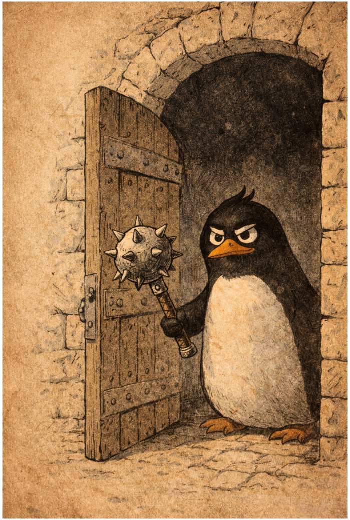
Kernel Hardening

Protecting the Kernel

- The kernel is the cornerstone and stronghold of a Linux based system.
- It provides pretty good security by default, but there are developer friendly settings that are very attacker friendly too!
- If the kernel is compromised, there is almost no limit to what an attacker can do.

There are two main types of techniques to make your kernel harder to compromise:

- Reducing the attack surface
  - Only keep the features and drivers needed in your system
  - Good fit for dedicated embedded systems anyway
  - Also helps to boot faster 😉
- Using kernel hardening features
  - Features making it harder to exploit unknown or future vulnerabilities
  - Developed through the Kernel Self-Protection project.



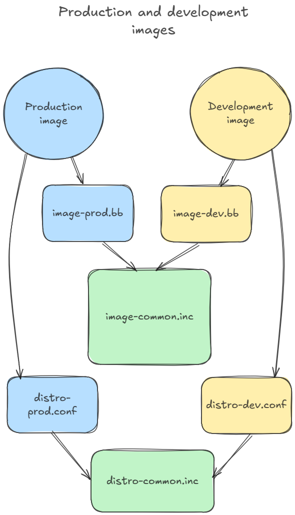
Kernel Hardening

Project Setup

Production and Development Images

Prepare for production early during development

- Maintain a production image alongside the development one
- Development settings isolated from the start from the production one
- Less risk of forgetting debugging features in the production image

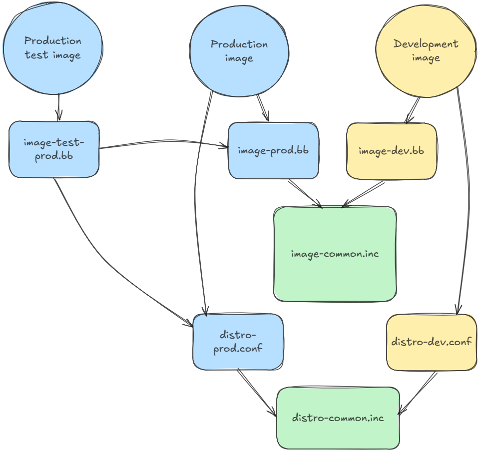


Need for Production Test Image

At this stage, we need some tools to test the production image

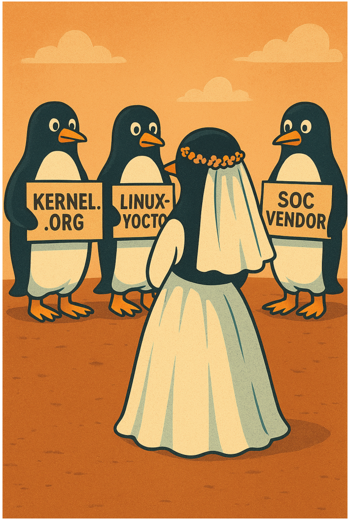
- Don't want to put them in the development image, will get false positives with development tools and features.
- Don't want to leave them in the production image, could be used to reveal weak spots. They may also pull unwanted dependencies.
- Solution: create a third "Production test" image.

Production test, production and development images



Choose Your Kernel Carefully

- An **LTS kernel** from kernel.org is the best solution to get kernel updates quickly
- **linux-yocto** is pretty well supported, if you can wait a few weeks for updates.
- **Vendor kernels** support your hardware well, but they are not meant to include vulnerability fixes in a timely fashion. They can also be very outdated.



Enable vulnerability checking

- Add this to `conf/local.conf`:

  ```
  INHERIT += "cve-check"
  ```

- This will add a CVE task to the recipes you're building
- You may also want to ignore CVEs that are irrelevant to Poky and OE-core:

  ```
  include conf/distro/include/cve-extra-exclusions.inc
  ```

- To speed up NVD database downloads, request a unique key (`NVDCVE_API_KEY`)

  ```
  NVDCVE_API_KEY = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
  ```

- Then, `bitbake` your regular image,
  and the checks will be run (without running the other tasks if not necessary)
- You can also run checks on specific recipes:

  ```
  $ bitbake -c cve_check linux-yocto
  ```

Kernel Hardening

Detect Hardening Opportunities

`kernel-hardening-checker` from Alexander Popov

- https://github.com/a13xp0p0v/kernel-hardening-checker
- Introduced only in Walnascar (`meta-oe`)
- Reports all the kernel configuration and command line settings which could be changed to harden the running kernel.
- My contributions:
  - Updated `meta-oe` to support version 0.6.10.2 (many recent changes), supporting a -a option and the RISC-V architecture (link)
  - Backported to Scarthgap too.

recipes-core/images/image-prod-test.bb

```
IMAGE_INSTALL += "kernel-hardening-checker"
```

kernel-hardening-checker example output

```
root@beaglebone-yocto:~# kernel-hardening-checker -a
[+] Going to autodetect and check the security hardening options of the running kernel
[+] Detected version of the running kernel: (6, 12, 41)
[+] Detected kconfig file of the running kernel: /proc/config.gz
[+] Detected cmdline parameters of the running kernel: /proc/cmdline
ICMPv6: process `sysctl' is using deprecated sysctl (syscall) net.ipv6.neigh.default.base_reachable_time - use net.ipv6.neigh.default.base_reachable_
[!] WARNING: sysctl command returned 1
[+] Saved sysctls to a temporary file /tmp/sysctl-58k0_iyd
[+] Detected architecture: ARM
[+] Detected compiler: GCC 150200
[!] WARNING: can't check CONFIG_ARCH_MMAP_RND_COMPAT_BITS without CONFIG_ARCH_MMAP_RND_COMPAT_BITS_MAX
[!] WARNING: can't check vm.mmap_rnd_compat_bits without CONFIG_ARCH_MMAP_RND_COMPAT_BITS_MAX
=========================================================================================================================
              option_name              | type  |      reason      | decision |desired_val | check_result
=========================================================================================================================
CONFIG_BUG                             |kconfig| self_protection  |defconfig |     y      | OK
CONFIG_SLUB_DEBUG                      |kconfig| self_protection  |defconfig |     y      | OK
CONFIG_THREAD_INFO_IN_TASK             |kconfig| self_protection  |defconfig |     y      | OK
CONFIG_IOMMU_DEFAULT_PASSTHROUGH       |kconfig| self_protection  |defconfig | is not set | OK: is not found
CONFIG_IOMMU_SUPPORT                   |kconfig| self_protection  |defconfig |     y      | OK
CONFIG_STACKPROTECTOR                  |kconfig| self_protection  |defconfig |     y      | OK
CONFIG_STACKPROTECTOR_STRONG           |kconfig| self_protection  |defconfig |     y      | OK
CONFIG_STRICT_KERNEL_RWX               |kconfig| self_protection  |defconfig |     y      | OK
CONFIG_STRICT_MODULE_RWX               |kconfig| self_protection  |defconfig |     y      | OK
CONFIG_REFCOUNT_FULL                   |kconfig| self_protection  |defconfig |     y      | OK: version >= (5, 4, 208)
CONFIG_INIT_STACK_ALL_ZERO             |kconfig| self_protection  |defconfig |     y      | OK
CONFIG_CPU_MITIGATIONS                 |kconfig| self_protection  |defconfig |     y      | OK
CONFIG_VMAP_STACK                      |kconfig| self_protection  |defconfig |     y      | OK
CONFIG_IOMMU_DEFAULT_DMA_STRICT        |kconfig| self_protection  |defconfig |     y      | FAIL: is not found
CONFIG_STACKPROTECTOR_PER_TASK         |kconfig| self_protection  |defconfig |     y      | OK
CONFIG_HW_RANDOM_TPM                   |kconfig| self_protection  |defconfig |     y      | FAIL: is not found
CONFIG_CPU_SW_DOMAIN_PAN               |kconfig| self_protection  |defconfig |     y      | OK
CONFIG_HARDEN_BRANCH_PREDICTOR         |kconfig| self_protection  |defconfig |     y      | OK
CONFIG_HARDEN_BRANCH_HISTORY           |kconfig| self_protection  |defconfig |     y      | OK
CONFIG_DEBUG_ALIGN_RODATA              |kconfig| self_protection  |defconfig |     y      | OK
CONFIG_RANDOM_KMALLOC_CACHES           |kconfig| self_protection  |   kspp   |     y      | FAIL: "is not set"
CONFIG_SLAB_MERGE_DEFAULT              |kconfig| self_protection  |   kspp   | is not set | FAIL: "y"
CONFIG_BUG_ON_DATA_CORRUPTION          |kconfig| self_protection  |   kspp   |     y      | FAIL: "is not set"
CONFIG_SLAB_FREELIST_HARDENED          |kconfig| self_protection  |   kspp   |     y      | FAIL: "is not set"
CONFIG_SLAB_FREELIST_RANDOM            |kconfig| self_protection  |   kspp   |     y      | FAIL: "is not set"
CONFIG_SHUFFLE_PAGE_ALLOCATOR          |kconfig| self_protection  |   kspp   |     y      | FAIL: "is not set"
CONFIG_FORTIFY_SOURCE                  |kconfig| self_protection  |   kspp   |     y      | FAIL: "is not set"
CONFIG_DEBUG_VIRTUAL                   |kconfig| self_protection  |   kspp   |     y      | FAIL: "is not set"
CONFIG_INIT_ON_ALLOC_DEFAULT_ON        |kconfig| self_protection  |   kspp   |     y      | FAIL: "is not set"
CONFIG_STATIC_USERMODEHELPER           |kconfig| self_protection  |   kspp   |     y      | FAIL: "is not set"
CONFIG_SECURITY_LOCKDOWN_LSM           |kconfig| self_protection  |   kspp   |     y      | FAIL: "is not set"
CONFIG_LSM                             |kconfig| self_protection  |   kspp   | *lockdown* | OK: in "landlock,lockdown,yama,loadpin,safesetid,ipe,bpf"
CONFIG_SECURITY_LOCKDOWN_LSM_EARLY     |kconfig| self_protection  |   kspp   |     y      | FAIL: is not found
CONFIG_LOCK_DOWN_KERNEL_FORCE_CONFIDENTIALITY|kconfig| self_protection  |   kspp   |     y      | FAIL: is not found
CONFIG_DEBUG_CREDENTIALS               |kconfig| self_protection  |   kspp   |     y      | OK: version >= (6, 6, 8)
CONFIG_DEBUG_NOTIFIERS                 |kconfig| self_protection  |   kspp   |     y      | FAIL: "is not set"
CONFIG_KFENCE                          |kconfig| self_protection  |   kspp   |     y      | OK
CONFIG_KFENCE_SAMPLE_INTERVAL          |kconfig| self_protection  |   kspp   |    100     | OK
CONFIG_RANDSTRUCT_FULL                 |kconfig| self_protection  |   kspp   |     y      | FAIL: "is not set"
CONFIG_HARDENED_USERCOPY               |kconfig| self_protection  |   kspp   |     y      | FAIL: "is not set"
CONFIG_HARDENED_USERCOPY_FALLBACK      |kconfig| self_protection  |   kspp   | is not set | FAIL: CONFIG_HARDENED_USERCOPY is not "y"
CONFIG_HARDENED_USERCOPY_PAGESPAN      |kconfig| self_protection  |   kspp   | is not set | FAIL: CONFIG_HARDENED_USERCOPY is not "y"
CONFIG_GCC_PLUGIN_LATENT_ENTROPY       |kconfig| self_protection  |   kspp   |     y      | FAIL: "is not set"
CONFIG_MODULE_SIG                      |kconfig| self_protection  |   kspp   |     y      | FAIL: "is not set"
CONFIG_MODULE_SIG_ALL                  |kconfig| self_protection  |   kspp   |     y      | FAIL: is not found
CONFIG_MODULE_SIG_SHA512               |kconfig| self_protection  |   kspp   |     y      | FAIL: is not found
CONFIG_MODULE_SIG_FORCE                |kconfig| self_protection  |   kspp   |     y      | FAIL: is not found
CONFIG_INIT_ON_FREE_DEFAULT_ON         |kconfig| self_protection  |   kspp   |     y      | FAIL: "is not set"
CONFIG_EFI_DISABLE_PCI_DMA             |kconfig| self_protection  |   kspp   |     y      | OK: CONFIG_EFI is "is not set"
CONFIG_RESET_ATTACK_MITIGATION         |kconfig| self_protection  |   kspp   |     y      | OK: CONFIG_EFI is "is not set"
CONFIG_UBSAN_BOUNDS                    |kconfig| self_protection  |   kspp   |     y      | FAIL: is not found
CONFIG_UBSAN_LOCAL_BOUNDS              |kconfig| self_protection  |   kspp   |     y      | FAIL: is not found
CONFIG_UBSAN_TRAP                      |kconfig| self_protection  |   kspp   |     y      | FAIL: CONFIG_UBSAN_BOUNDS is not "y"
CONFIG_UBSAN_SANITIZE_ALL              |kconfig| self_protection  |   kspp   |     y      | FAIL: is not found
CONFIG_SCHED_CORE                      |kconfig| self_protection  |   kspp   |     y      | FAIL: is not found
CONFIG_DEBUG_SG                        |kconfig| self_protection  |   kspp   |     y      | FAIL: "is not set"
CONFIG_LIST_HARDENED                   |kconfig| self_protection  |   kspp   |     y      | FAIL: "is not set"
CONFIG_SCHED_STACK_END_CHECK           |kconfig| self_protection  |   kspp   |     y      | OK: CONFIG_VMAP_STACK is "y"
CONFIG_SYN_COOKIES                     |kconfig| self_protection  |   kspp   |     y      | OK
CONFIG_WERROR                          |kconfig| self_protection  |   kspp   |     y      | FAIL: "is not set"
CONFIG_DEFAULT_MMAP_MIN_ADDR           |kconfig| self_protection  |   kspp   |   32768    | FAIL: "4096"
CONFIG_ARM_DEBUG_WX                    |kconfig| self_protection  |   kspp   |     y      | FAIL: "is not set"
CONFIG_ARM_SMMU                        |kconfig| self_protection  |a13xp0p0v |     y      | FAIL: "is not set"
CONFIG_ARM_SMMU_DISABLE_BYPASS_BY_DEFAULT|kconfig| self_protection  |a13xp0p0v |     y      | FAIL: is not found
```

...

```
CONFIG_MMIOTRACE_TEST                  |kconfig|cut_attack_surface| lockdown | is not set | OK: is not found
CONFIG_KPROBES                         |kconfig|cut_attack_surface| lockdown | is not set | FAIL: "y"
CONFIG_BPF_SYSCALL                     |kconfig|cut_attack_surface| lockdown | is not set | FAIL: "y"
CONFIG_MMIOTRACE                       |kconfig|cut_attack_surface|a13xp0p0v | is not set | OK: is not found
CONFIG_LIVEPATCH                       |kconfig|cut_attack_surface|a13xp0p0v | is not set | OK: is not found
CONFIG_IP_DCCP                         |kconfig|cut_attack_surface|a13xp0p0v | is not set | OK: is not found
CONFIG_FTRACE                          |kconfig|cut_attack_surface|a13xp0p0v | is not set | FAIL: "y"
CONFIG_VIDEO_VIVID                     |kconfig|cut_attack_surface|a13xp0p0v | is not set | OK: is not found
CONFIG_INPUT_EVBUG                     |kconfig|cut_attack_surface|a13xp0p0v | is not set | OK
CONFIG_CORESIGHT                       |kconfig|cut_attack_surface|a13xp0p0v | is not set | OK
CONFIG_XFS_SUPPORT_V4                  |kconfig|cut_attack_surface|a13xp0p0v | is not set | OK: is not found
CONFIG_BLK_DEV_WRITE_MOUNTED           |kconfig|cut_attack_surface|a13xp0p0v | is not set | FAIL: "y"
CONFIG_FAULT_INJECTION                 |kconfig|cut_attack_surface|a13xp0p0v | is not set | OK
CONFIG_ARM_PTDUMP_DEBUGFS              |kconfig|cut_attack_surface|a13xp0p0v | is not set | OK
CONFIG_ARM_PTDUMP                      |kconfig|cut_attack_surface|a13xp0p0v | is not set | OK: is not found
CONFIG_SECCOMP_CACHE_DEBUG             |kconfig|cut_attack_surface|a13xp0p0v | is not set | OK
CONFIG_LKDTM                           |kconfig|cut_attack_surface|a13xp0p0v | is not set | OK
CONFIG_TRIM_UNUSED_KSYMS               |kconfig|cut_attack_surface|a13xp0p0v |     y      | FAIL: "is not set"
CONFIG_VMSPLIT_3G                      |kconfig| harden_userspace |defconfig |     y      | OK
CONFIG_COREDUMP                        |kconfig| harden_userspace |  clipos  | is not set | FAIL: "y"
CONFIG_ARCH_MMAP_RND_BITS              |kconfig| harden_userspace |a13xp0p0v |     16     | FAIL: "8"
nosmep                                 |cmdline| self_protection  |defconfig | is not set | OK: is not found
nosmap                                 |cmdline| self_protection  |defconfig | is not set | OK: is not found
nokaslr                                |cmdline| self_protection  |defconfig | is not set | OK: is not found
nopti                                  |cmdline| self_protection  |defconfig | is not set | OK: is not found
no_hash_pointers                       |cmdline| self_protection  |defconfig | is not set | OK: is not found
nospectre_v1                           |cmdline| self_protection  |defconfig | is not set | OK: is not found
nospectre_v2                           |cmdline| self_protection  |defconfig | is not set | OK: is not found
nospectre_bhb                          |cmdline| self_protection  |defconfig | is not set | OK: is not found
nospec_store_bypass_disable            |cmdline| self_protection  |defconfig | is not set | OK: is not found
dis_ucode_ldr                          |cmdline| self_protection  |defconfig | is not set | OK: is not found
arm64.nobti                            |cmdline| self_protection  |defconfig | is not set | OK: is not found
arm64.nopauth                          |cmdline| self_protection  |defconfig | is not set | OK: is not found
arm64.nomte                            |cmdline| self_protection  |defconfig | is not set | OK: is not found
iommu.passthrough                      |cmdline| self_protection  |defconfig |     0      | OK: CONFIG_IOMMU_DEFAULT_PASSTHROUGH is not found
rodata                                 |cmdline| self_protection  |defconfig |     on     | OK: rodata is not found
iommu.strict                           |cmdline| self_protection  |defconfig |     1      | FAIL: is not found
mitigations                            |cmdline| self_protection  |defconfig |    auto    | OK: mitigations is not found
slab_merge                             |cmdline| self_protection  |   kspp   | is not set | OK: is not found
slub_merge                             |cmdline| self_protection  |   kspp   | is not set | OK: is not found
page_alloc.shuffle                     |cmdline| self_protection  |   kspp   |     1      | FAIL: is not found
slab_nomerge                           |cmdline| self_protection  |   kspp   | is present | FAIL: is not present
init_on_alloc                          |cmdline| self_protection  |   kspp   |     1      | FAIL: is not found
init_on_free                           |cmdline| self_protection  |   kspp   |     1      | FAIL: is not found
hardened_usercopy                      |cmdline| self_protection  |   kspp   |     1      | FAIL: is not found
slab_common.usercopy_fallback          |cmdline| self_protection  |   kspp   | is not set | OK: is not found
kfence.sample_interval                 |cmdline| self_protection  |   kspp   |    100     | OK: CONFIG_KFENCE_SAMPLE_INTERVAL is "100"
nosmt                                  |cmdline|cut_attack_surface|   kspp   | is present | FAIL: is not present
debugfs                                |cmdline|cut_attack_surface|  grsec   |    off     | FAIL: is not found
sysrq_always_enabled                   |cmdline|cut_attack_surface|grapheneos| is not set | OK: is not found
bdev_allow_write_mounted               |cmdline|cut_attack_surface|a13xp0p0v |     0      | FAIL: is not found
norandmaps                             |cmdline| harden_userspace |defconfig | is not set | OK: is not found
net.core.bpf_jit_harden                |sysctl | self_protection  |   kspp   |     2      | FAIL: "0"
kernel.oops_limit                      |sysctl | self_protection  |a13xp0p0v |    100     | FAIL: "10000"
kernel.warn_limit                      |sysctl | self_protection  |a13xp0p0v |    100     | FAIL: "0"
vm.mmap_min_addr                       |sysctl | self_protection  |   kspp   |   32768    | FAIL: "4096"
kernel.dmesg_restrict                  |sysctl |cut_attack_surface|   kspp   |     1      | FAIL: "0"
kernel.perf_event_paranoid             |sysctl |cut_attack_surface|   kspp   |     3      | FAIL: "2"
dev.tty.ldisc_autoload                 |sysctl |cut_attack_surface|   kspp   |     0      | FAIL: "1"
kernel.kptr_restrict                   |sysctl |cut_attack_surface|   kspp   |     2      | FAIL: "0"
dev.tty.legacy_tiocsti                 |sysctl |cut_attack_surface|   kspp   |     0      | FAIL: "1"
user.max_user_namespaces               |sysctl |cut_attack_surface|   kspp   |     0      | FAIL: "3740"
kernel.kexec_load_disabled             |sysctl |cut_attack_surface|   kspp   |     1      | OK: CONFIG_KEXEC_CORE is not found
kernel.unprivileged_bpf_disabled       |sysctl |cut_attack_surface|   kspp   |     1      | FAIL: "2"
vm.unprivileged_userfaultfd            |sysctl |cut_attack_surface|   kspp   |     0      | OK: CONFIG_USERFAULTFD is "is not set"
kernel.modules_disabled                |sysctl |cut_attack_surface|   kspp   |     1      | FAIL: "0"
kernel.io_uring_disabled               |sysctl |cut_attack_surface|  grsec   |     2      | FAIL: "0"
kernel.sysrq                           |sysctl |cut_attack_surface|a13xp0p0v |     0      | FAIL: "16"
fs.protected_symlinks                  |sysctl | harden_userspace |   kspp   |     1      | OK
fs.protected_hardlinks                 |sysctl | harden_userspace |   kspp   |     1      | OK
fs.protected_fifos                     |sysctl | harden_userspace |   kspp   |     2      | FAIL: "1"
fs.protected_regular                   |sysctl | harden_userspace |   kspp   |     2      | FAIL: "1"
fs.suid_dumpable                       |sysctl | harden_userspace |   kspp   |     0      | OK
kernel.randomize_va_space              |sysctl | harden_userspace |   kspp   |     2      | FAIL: "1"
kernel.yama.ptrace_scope               |sysctl | harden_userspace |   kspp   |     3      | FAIL: is not found
vm.mmap_rnd_bits                       |sysctl | harden_userspace |a13xp0p0v |     16     | FAIL: "8"

[+] Config check is finished: 'OK' - 145 / 'FAIL' - 109
root@beaglebone-yocto:~#
```

Using kernel-hardening-checker

Try it on your PC!

- Clone the repository:

```
$ git clone https://github.com/a13xp0p0v/kernel-hardening-checker.git
```

- Run it directly from the repository:

```
$ /path/to/kernel-hardening/checker/bin/kernel-hardening-checker -a
```

Run it on the embedded target:

```
$ kernel-hardening-checker -a
```

Lynis

Lynis (https://cisofy.com/lynis/)

- Very exhaustive checker for a very wide series of system security checks
- This also includes networking and kernel checks
- Primarily for PC distros but gives you tons of relevant ideas for hardening your OS
- Available in the `meta-security` layer
- Add it to your build:

```
IMAGE_INSTALL += "lynis"
```

- Run it on your board:

```
$ lynis audit system
```

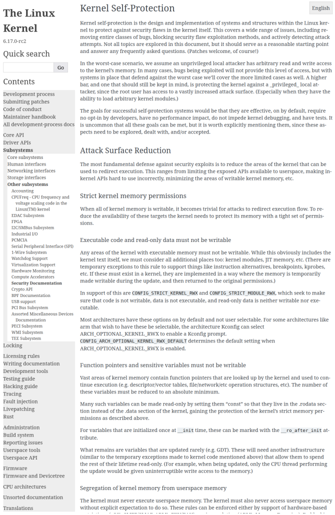
```
  - Checking presence TOMOYO Linux                           [ NOT FOUND ]
  - Checking presence grsecurity                             [ NOT FOUND ]
  - Checking for implemented MAC framework                   [ NONE ]

[+] Software: file integrity
------------------------------------
  - Checking file integrity tools
  - Checking presence integrity tool                         [ NOT FOUND ]

[+] Software: System tooling
------------------------------------
  - Checking automation tooling
  - Automation tooling                                       [ NOT FOUND ]
  - Checking for IDS/IPS tooling                             [ NONE ]

[+] Software: Malware
------------------------------------
  - Malware software components                              [ NOT FOUND ]

[+] File Permissions
------------------------------------
  - Starting file permissions check
    File: /etc/group                                         [ OK ]
    File: /etc/issue                                         [ OK ]
    File: /etc/issue.net                                     [ OK ]
    File: /etc/motd                                          [ OK ]
    File: /etc/passwd                                        [ OK ]
    Directory: /root/.ssh                                    [ OK ]

[+] Home directories
------------------------------------
  - Permissions of home directories                          [ OK ]
  - Ownership of home directories                            [ OK ]
  - Checking shell history files                             [ OK ]

[+] Kernel Hardening
------------------------------------
  - Comparing sysctl key pairs with scan profile
    - dev.tty.ldisc_autoload (exp: 0)                        [ DIFFERENT ]
    - fs.protected_fifos (exp: 2)                            [ DIFFERENT ]
    - fs.protected_hardlinks (exp: 1)                        [ OK ]
    - fs.protected_regular (exp: 2)                          [ DIFFERENT ]
    - fs.protected_symlinks (exp: 1)                         [ OK ]
    - fs.suid_dumpable (exp: 0)                              [ OK ]
    - kernel.core_uses_pid (exp: 1)                          [ OK ]
    - kernel.ctrl-alt-del (exp: 0)                           [ OK ]
    - kernel.dmesg_restrict (exp: 1)                         [ DIFFERENT ]
    - kernel.kptr_restrict (exp: 2)                          [ DIFFERENT ]
    - kernel.modules_disabled (exp: 1)                       [ DIFFERENT ]
    - kernel.perf_event_paranoid (exp: 2 3 4)                [ OK ]
    - kernel.randomize_va_space (exp: 2)                     [ DIFFERENT ]
    - kernel.sysrq (exp: 0)                                  [ DIFFERENT ]
    - kernel.unprivileged_bpf_disabled (exp: 1)              [ DIFFERENT ]
    - net.core.bpf_jit_harden (exp: 2)                       [ DIFFERENT ]
    - net.ipv4.conf.all.accept_redirects (exp: 0)            [ OK ]
    - net.ipv4.conf.all.accept_source_route (exp: 0)         [ OK ]
    - net.ipv4.conf.all.bootp_relay (exp: 0)                 [ OK ]
    - net.ipv4.conf.all.forwarding (exp: 0)                  [ OK ]
    - net.ipv4.conf.all.log_martians (exp: 1)                [ OK ]
    - net.ipv4.conf.all.mc_forwarding (exp: 0)               [ OK ]
    - net.ipv4.conf.all.proxy_arp (exp: 0)                   [ OK ]
    - net.ipv4.conf.all.rp_filter (exp: 1)                   [ DIFFERENT ]
    - net.ipv4.conf.all.send_redirects (exp: 0)              [ DIFFERENT ]
    - net.ipv4.conf.default.accept_redirects (exp: 0)        [ DIFFERENT ]
    - net.ipv4.conf.default.accept_source_route (exp: 0)     [ OK ]
    - net.ipv4.conf.default.log_martians (exp: 1)            [ DIFFERENT ]
    - net.ipv4.icmp_echo_ignore_broadcasts (exp: 1)          [ OK ]
    - net.ipv4.icmp_ignore_bogus_error_responses (exp: 1)    [ OK ]
    - net.ipv4.tcp_syncookies (exp: 1)                       [ OK ]
    - net.ipv4.tcp_timestamps (exp: 0 1)                     [ OK ]
    - net.ipv6.conf.all.accept_redirects (exp: 0)            [ DIFFERENT ]
    - net.ipv6.conf.all.accept_source_route (exp: 0)         [ OK ]
    - net.ipv6.conf.default.accept_redirects (exp: 0)        [ DIFFERENT ]
    - net.ipv6.conf.default.accept_source_route (exp: 0)     [ OK ]

[+] Hardening
------------------------------------

 [WARNING]: Test KRNL-6000 had a long execution: 12.387879 seconds

    - Installed compiler(s)                                  [ NOT FOUND ]
    - Installed malware scanner                              [ NOT FOUND ]
    - Non-native binary formats                              [ NOT FOUND ]

[+] Custom tests
------------------------------------
  - Running custom tests...                                  [ NONE ]
```

Output of `lynis audit system` on BeagleBone Black.

Kernel Hardening Notes

root commit

- Kernel configuration: remove unnecessary features
  - This reduces the attack surface, and boot time too!
  - Yocto stores kernel modules in separate packages. This helps to identify unnecessary ones!
- See `security/self-protection` (kernel documentation) for details about kernel protection techniques.



The Linux Kernel

6.17.0-rc2

Quick search

Go

Contents

Development process
Submitting patches
Code of conduct
Maintainer handbook
All development-process docs
Core API
Driver APIs
Subsystems
Core subsystems
Human interfaces
Networking interfaces
Storage interfaces
Other subsystems
Accounting
CPUFreq - CPU frequency and voltage scaling code in the Linux(TM) kernel
EDAC Subsystem
FPGA
I2C/SMBus Subsystem
Industrial I/O
PCMCIA
Serial Peripheral Interface (SPI)
1-Wire Subsystem
Watchdog Support
Virtualization Support
Hardware Monitoring
Compute Accelerators
Security Documentation
Crypto API
BPF Documentation
USB support
PCI Bus Subsystem
Assorted Miscellaneous Devices Documentation
PECI Subsystem
WMI Subsystem
TEE Subsystem
Locking
Licensing rules
Writing documentation
Development tools
Testing guide
Hacking guide
Tracing
Fault injection
Livepatching
Rust
Administration
Build system
Reporting issues
Userspace tools
Userspace API
Firmware
Firmware and Devicetree
CPU architectures
Unsorted documentation
Translations

English

Kernel Self-Protection

Kernel self-protection is the design and implementation of systems and structures within the Linux kernel to protect against security flaws in the kernel itself. This covers a wide range of issues, including removing entire classes of bugs, blocking security flaw exploitation methods, and actively detecting attack attempts. Not all topics are explored in this document, but it should serve as a reasonable starting point and answer any frequently asked questions. (Patches welcome, of course!)

In the worst-case scenario, we assume an unprivileged local attacker has arbitrary read and write access to the kernel's memory. In many cases, bugs being exploited will not provide this level of access, but with systems in place that defend against the worst case we'll cover the more limited cases as well. A higher bar, and one that should still be kept in mind, is protecting the kernel against a _privileged_ local attacker, since the root user has access to a vastly increased attack surface. (Especially when they have the ability to load arbitrary kernel modules.)

The goals for successful self-protection systems would be that they are effective, on by default, require no opt-in by developers, have no performance impact, do not impede kernel debugging, and have tests. It is uncommon that all these goals can be met, but it is worth explicitly mentioning them, since these aspects need to be explored, dealt with, and/or accepted.

Attack Surface Reduction

The most fundamental defense against security exploits is to reduce the areas of the kernel that can be used to redirect execution. This ranges from limiting the exposed APIs available to userspace, making in-kernel APIs hard to use incorrectly, minimizing the areas of writable kernel memory, etc.

Strict kernel memory permissions

When all of kernel memory is writable, it becomes trivial for attacks to redirect execution flow. To reduce the availability of these targets the kernel needs to protect its memory with a tight set of permissions.

Executable code and read-only data must not be writable

Any areas of the kernel with executable memory must not be writable. While this obviously includes the kernel text itself, we must consider all additional places too: kernel modules, JIT memory, etc. (There are temporary exceptions to this rule to support things like instruction alternatives, breakpoints, kprobes, etc. If these must exist in a kernel, they are implemented in a way where the memory is temporarily made writable during the update, and then returned to the original permissions.)

In support of this are `CONFIG_STRICT_KERNEL_RWX` and `CONFIG_STRICT_MODULE_RWX`, which seek to make sure that code is not writable, data is not executable, and read-only data is neither writable nor executable.

Most architectures have these options on by default and not user selectable. For some architectures like arm that wish to have these be selectable, the architecture Kconfig can select ARCH_OPTIONAL_KERNEL_RWX to enable a Kconfig prompt. `CONFIG_ARCH_OPTIONAL_KERNEL_RWX_DEFAULT` determines the default setting when ARCH_OPTIONAL_KERNEL_RWX is enabled.

Function pointers and sensitive variables must not be writable

Vast areas of kernel memory contain function pointers that are looked up by the kernel and used to continue execution (e.g. descriptor/vector tables, file/network/etc operation structures, etc). The number of these variables must be reduced to an absolute minimum.

Many such variables can be made read-only by setting them "const" so that they live in the .rodata section instead of the .data section of the kernel, gaining the protection of the kernel's strict memory permissions as described above.

For variables that are initialized once at `__init` time, these can be marked with the `__ro_after_init` attribute.

What remains are variables that are updated rarely (e.g. GDT). These will need another infrastructure (similar to the temporary exceptions made to kernel code mentioned above) that allow them to spend the rest of their lifetime read-only. (For example, when being updated, only the CPU thread performing the update would be given uninterruptible write access to the memory.)

Segregation of kernel memory from userspace memory

The kernel must never execute userspace memory. The kernel must also never access userspace memory without explicit expectation to do so. These rules can be enforced either by support of hardware-based

Kernel Hardening

Implement Hardening

Modify Kernel Configuration Settings

Using config fragments

- Possible for recipes inheriting the `linux-yocto` class
- Convenient to manage settings separately
- But adds impredictibility in case of settings

my-linux.bb

```
SRC_URI += " \
        file://kallsyms.cfg \
        file://randstruct-full.cfg \
        file://io-strict-devmem.cfg \
"
```

io-strict-devmem.cfg

```
CONFIG_DEVMEM=y
CONFIG_IO_STRICT_DEVMEM=y
```

Using one complete configuration file

- Guaranteed to work as intended
  No issues because of conflicting parameters
- Can be updated easily:
  `bitbake -c menuconfig linux-yocto`
  `bitbake -c savedefconfig linux-yocto`

my-linux.bb

```
SRC_URI += "file://defconfig"
```

Kernel Hardening — How To Proceed?

- First apply configuration changes
  - First remove unnecessary kernel features (time consuming!)
  - Then follow applicable recommendations of `kernel-hardening-checker`
  - Do it little by little, test at each iteration
  - Keep track of all intermediate configurations (if rewinding is needed)
- Then tune the remaining kernel command line and *sysctl* settings
  Some are no longer relevant after kernel configuration changes

Selected Kernel Hardening Settings (1)

```
                    Harden common str/mem functions against buffer overflows
CONFIG_FORTIFY_SOURCE:

Detect overflows of buffers in common string and memory functions
where the compiler can determine and validate the buffer sizes.

Symbol: FORTIFY_SOURCE [=y]
Type  : bool
Defined at security/Kconfig.hardening:216
  Prompt: Harden common str/mem functions against buffer overflows
  Depends on: ARCH_HAS_FORTIFY_SOURCE [=y] && (!X86_32 || !CC_IS_CLANG [=n] || CLANG_VERSION [=0]>=160000)
  Location:
    -> Security options
      -> Kernel hardening options
        -> Bounds checking
          -> Harden common str/mem functions against buffer overflows (FORTIFY_SOURCE [=y])

                                                                              (100%)
                                  < Exit >
```

```
                                       Warn on W+X mappings at boot
CONFIG_ARM_DEBUG_WX:

Generate a warning if any W+X mappings are found at boot.

This is useful for discovering cases where the kernel is leaving
W+X mappings after applying NX, as such mappings are a security risk.

Look for a message in dmesg output like this:

        arm/mm: Checked W+X mappings: passed, no W+X pages found.

or like this, if the check failed:

        arm/mm: Checked W+X mappings: FAILED, <N> W+X pages found.

Note that even if the check fails, your kernel is possibly
still fine, as W+X mappings are not a security hole in
themselves, what they do is that they make the exploitation
of other unfixed kernel bugs easier.

There is no runtime or memory usage effect of this option
once the kernel has booted up - it's a one time check.

If in doubt, say "Y".

                                                                              ( 73%)
                                  < Exit >
```

```
                 KFENCE: low-overhead sampling-based memory safety error detector
CONFIG_KFENCE:

KFENCE is a low-overhead sampling-based detector of heap out-of-bounds
access, use-after-free, and invalid-free errors. KFENCE is designed
to have negligible cost to permit enabling it in production
environments.

See <file:Documentation/dev-tools/kfence.rst> for more details.

Note that, KFENCE is not a substitute for explicit testing with tools
such as KASAN. KFENCE can detect a subset of bugs that KASAN can
detect, albeit at very different performance profiles. If you can
afford to use KASAN, continue using KASAN, for example in test
environments. If your kernel targets production use, and cannot
enable KASAN due to its cost, consider using KFENCE.

Symbol: KFENCE [=n]
Type  : bool
Defined at lib/Kconfig.kfence:6
  Prompt: KFENCE: low-overhead sampling-based memory safety error detector
  Depends on: HAVE_ARCH_KFENCE [=y]
  Location:
    -> Kernel hacking
      -> Memory Debugging
        -> KFENCE: low-overhead sampling-based memory safety error detector (KFENCE [=n])
                                                                              ( 95%)
                                  < Exit >
```

```
                           Harden memory copies between kernel and userspace
CONFIG_HARDENED_USERCOPY:

This option checks for obviously wrong memory regions when
copying memory to/from the kernel (via copy_to_user() and
copy_from_user() functions) by rejecting memory ranges that
are larger than the specified heap object, span multiple
separately allocated pages, are not on the process stack,
or are part of the kernel text. This prevents entire classes
of heap overflow exploits and similar kernel memory exposures.

Symbol: HARDENED_USERCOPY [=y]
Type  : bool
Defined at security/Kconfig.hardening:225
  Prompt: Harden memory copies between kernel and userspace
  Location:
    -> Security options
      -> Kernel hardening options
        -> Bounds checking
          -> Harden memory copies between kernel and userspace (HARDENED_USERCOPY [=y])
Implies: STRICT_DEVMEM [=n]

                                                                              (100%)
                                  < Exit >
```

Selected Kernel Hardening Settings (2)

```
                                Tracers
CONFIG_FTRACE:

Enable the kernel tracing infrastructure.

Symbol: FTRACE [=y]
Type  : bool
Defined at kernel/trace/Kconfig:200
  Prompt: Tracers
  Depends on: TRACING_SUPPORT [=y]
  Location:
    -> Kernel hacking
      -> Tracers (FTRACE [=y])

                                                        (100%)
                              < Exit >
```

```
                          Load all symbols for debugging/ksymoops
CONFIG_KALLSYMS:

Say Y here to let the kernel print out symbolic crash information and
symbolic stack backtraces. This increases the size of the kernel
somewhat, as all symbols have to be loaded into the kernel image.

Symbol: KALLSYMS [=y]
Type  : bool
Defined at init/Kconfig:1986
  Prompt: Load all symbols for debugging/ksymoops
  Visible if: EXPERT [=y]
  Location:
    -> General setup
      -> Configure standard kernel features (expert users) (EXPERT [=y])
        -> Load all symbols for debugging/ksymoops (KALLSYMS [=y])
Selected by [y]:
  - KPROBES [=y] && HAVE_KPROBES [=y]
Selected by [n]:
  - LATENCYTOP [=n] && DEBUG_KERNEL [=y] && STACKTRACE_SUPPORT [=y] && PROC_FS [=y] && (FRAME_POINTER [=n] || \
MIPS || PPC || S390 || MICROBLAZE || ARM [=y] || ARC || X86)
  - DEBUG_KMEMLEAK [=n] && DEBUG_KERNEL [=y] && HAVE_DEBUG_KMEMLEAK [=y]
  - CODE_TAGGING [=n]
  - LOCKDEP [=n] && DEBUG_KERNEL [=y] && LOCK_DEBUGGING_SUPPORT [=y]
  - FUNCTION_TRACER [=n] && FTRACE [=y] && HAVE_FUNCTION_TRACER [=y]
  - STACK_TRACER [=n] && FTRACE [=y] && HAVE_FUNCTION_TRACER [=y]
                                                        ( 93%)
                              < Exit >
```

```
                                Kprobes
CONFIG_KPROBES:

Kprobes allows you to trap at almost any kernel address and
execute a callback function.  register_kprobe() establishes
a probepoint and specifies the callback.  Kprobes is useful
for kernel debugging, non-intrusive instrumentation and testing.
If in doubt, say "N".

Symbol: KPROBES [=y]
Type  : bool
Defined at arch/Kconfig:117
  Prompt: Kprobes
  Depends on: HAVE_KPROBES [=y]
  Location:
    -> General architecture-dependent options
      -> Kprobes (KPROBES [=y])
Selects: KALLSYMS [=y] && EXECMEM [=y] && NEED_TASKS_RCU [=y]
Selected by [n]:
  - KGDB_HONOUR_BLOCKLIST [=n] && KGDB [=n] && HAVE_KPROBES [=y] && MODULES [=y]

                                                        (100%)
                              < Exit >
```

```
                          Enable kexec system call
CONFIG_KEXEC:

kexec is a system call that implements the ability to shutdown your
current kernel, and to start another kernel. It is like a reboot
but it is independent of the system firmware. And like a reboot
you can start any kernel with it, not just Linux.

The name comes from the similarity to the exec system call.

It is an ongoing process to be certain the hardware in a machine
is properly shutdown, so do not be surprised if this code does not
initially work for you. As of this writing the exact hardware
interface is strongly in flux, so no good recommendation can be
made.

Symbol: KEXEC [=y]
Type  : bool
Defined at kernel/Kconfig.kexec:20
  Prompt: Enable kexec system call
  Depends on: ARCH_SUPPORTS_KEXEC [=y]
  Location:
    -> General setup
      -> Kexec and crash features
        -> Enable kexec system call (KEXEC [=y])
Selects: KEXEC_CORE [=y]
                                                        ( 99%)
                              < Exit >
```

Selected Kernel Hardening Settings (3)

root commit

Poison kernel stack before returning from syscalls

```
CONFIG_KSTACK_ERASE:

This option makes the kernel erase the kernel stack before
returning from system calls. This has the effect of leaving
the stack initialized to the poison value, which both reduces
the lifetime of any sensitive stack contents and reduces
potential for uninitialized stack variable exploits or information
exposures (it does not cover functions reaching the same stack
depth as prior functions during the same syscall). This blocks
most uninitialized stack variable attacks, with the performance
impact being driven by the depth of the stack usage, rather than
the function calling complexity.

The performance impact on a single CPU system kernel compilation
sees a 1% slowdown, other systems and workloads may vary and you
are advised to test this feature on your expected workload before
deploying it.

Symbol: KSTACK_ERASE [=n]
Type  : bool
Defined at security/Kconfig.hardening:88
  Prompt: Poison kernel stack before returning from syscalls
  Depends on: HAVE_ARCH_KSTACK_ERASE [=y] && (GCC_PLUGINS [=y] || CC_HAS_SANCOV_STACK_DEPTH_CALLBACK [=n])
  Location:
    -> Security options
                                                                                    ( 87%)
                                        < Exit >
```

Enable core dump support

```
CONFIG_COREDUMP:

This option enables support for performing core dumps. You almost
certainly want to say Y here. Not necessary on systems that never
need debugging or only ever run flawless code.

Symbol: COREDUMP [=y]
Type  : bool
Defined at fs/Kconfig.binfmt:171
  Prompt: Enable core dump support
  Visible if: EXPERT [=y]
  Location:
    -> Executable file formats
      -> Enable core dump support (COREDUMP [=y])
                                                                                    (100%)
                                        < Exit >
```

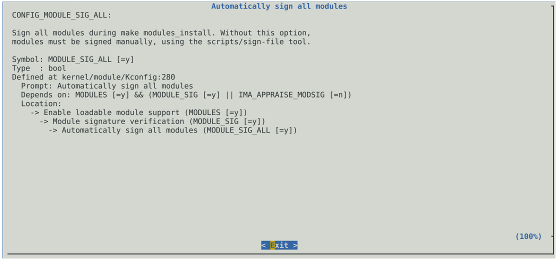
Disable heap randomization

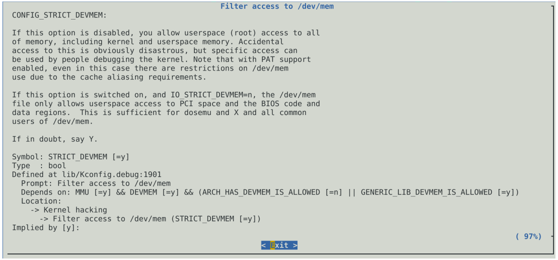
```
CONFIG_COMPAT_BRK:

Randomizing heap placement makes heap exploits harder, but it
also breaks ancient binaries (including anything libc5 based).
This option changes the bootup default to heap randomization
disabled, and can be overridden at runtime by setting
/proc/sys/kernel/randomize_va_space to 2.

On non-ancient distros (post-2000 ones) N is usually a safe choice.

Symbol: COMPAT_BRK [=y]
Type  : bool
Defined at mm/Kconfig:303
  Prompt: Disable heap randomization
  Location:
    -> Memory Management options
      -> Disable heap randomization (COMPAT_BRK [=y])
                                                                                    (100%)
                                        < Exit >
```

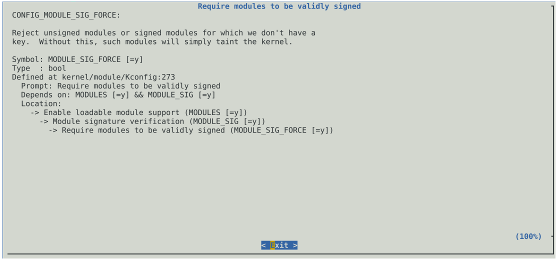
Allow writing to mounted block devices

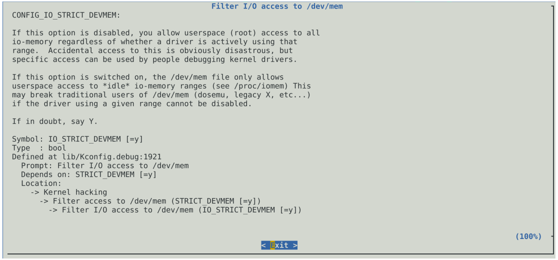
```
CONFIG_BLK_DEV_WRITE_MOUNTED:

When a block device is mounted, writing to its buffer cache is very
likely going to cause filesystem corruption. It is also rather easy to
crash the kernel in this way since the filesystem has no practical way
of detecting these writes to buffer cache and verifying its metadata
integrity. However there are some setups that need this capability
like running fsck on read-only mounted root device, modifying some
features on mounted ext4 filesystem, and similar. If you say N, the
kernel will prevent processes from writing to block devices that are
mounted by filesystems which provides some more protection from runaway
privileged processes and generally makes it much harder to crash
filesystem drivers. Note however that this does not prevent
underlying device(s) from being modified by other means, e.g. by
directly submitting SCSI commands or through access to lower layers of
storage stack. If in doubt, say Y. The configuration can be overridden
with the bdev_allow_write_mounted boot option.

Symbol: BLK_DEV_WRITE_MOUNTED [=y]
Type  : bool
Defined at block/Kconfig:77
  Prompt: Allow writing to mounted block devices
  Depends on: BLOCK [=y]
  Location:
    -> Enable the block layer (BLOCK [=y])
                                                                                    ( 94%)
                                        < Exit >
```

Selected Kernel Hardening Settings (4)









More such settings: https://gitlab.com/rootcommit/yocto-kernel-hardening/-/tree/main/graphics/more-kconfig

Signing Kernel Modules

root commit

- Module signing is automatic with CONFIG_MODULE_SIG_ALL.
  Also turn on CONFIG_MODULE_SIG_FORCE.
- The signing key is generated on the fly and the public key is built into the kernel.
- Not necessary to keep the private key.
- Still possible to use your own keys. Yocto can help you:
  https://ejaaskel.dev/yocto-hardening-kernel-module-signing/

See kernel documentation for details: admin-guide/module-signing

Modifying the Kernel Command Line

Not completely trivial!

- No universal way to change the default command line (check the bootloader recipes).
- However, you can always add parameters to the command line through the kernel *bootconfig* mechanism.

See kernel documentation:
admin-guide/bootconfig

linux-stable_%.bbappend

```
FILESEXTRAPATHS:prepend := "${THISDIR}/${PN}:"
KBUILD_DEFCONFIG = ""

do_configure:prepend() {
    install -m 0644 ${UNPACKDIR}/cmdline.bootconfig ${S}/
}

SRC_URI += " \
    file://defconfig \
    file://cmdline.bootconfig \
"
```

defconfig

```
...
CONFIG_BOOT_CONFIG=y
CONFIG_BOOT_CONFIG_EMBED=y
CONFIG_BOOT_CONFIG_EMBED_FILE="cmdline.bootconfig"
```

cmdline.bootconfig

```
kernel {
        page_alloc.shuffle=1
        hash_pointers=always
        nosmt
}
```

Modifying *sysctl* Parameters

recipes-core/sysctl/sysctl-config.bb

```
SUMMARY = "Custom sysctl configuration for kernel hardening"
LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://${COMMON_LICENSE_DIR}/MIT;md5=0835ade698e0bcf8506ecda..."

SRC_URI = "file://99-kernel-hardening.conf"

do_install() {
    install -Dm 0644 ${UNPACKDIR}/99-kernel-hardening.conf \
        ${D}${sysconfdir}/sysctl.d/99-kernel-hardening.conf
}

FILES:${PN} += "${sysconfdir}/sysctl.d/99-kernel-hardening.conf"
```

files/99-kernel-hardening.conf

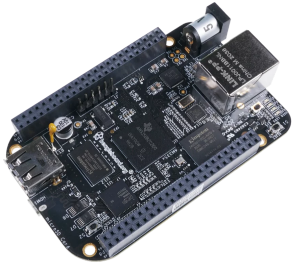
```
# Kernel hardening settings
kernel.oops_limit = 100
kernel.warn_limit = 100
kernel.perf_event_paranoid = 3
kernel.kptr_restrict = 2
user.max_user_namespaces = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv6.conf.all.accept_redirects = 0
net.ipv6.conf.default.accept_redirects = 0
net.ipv6.conf.all.accept_ra = 0
net.ipv6.conf.default.accept_ra = 0
fs.protected_fifos = 2
fs.protected_regular = 2
kernel.yama.ptrace_scope = 3
```

Kernel Hardening: Notes

- Many default kernel options are developer friendly, not security friendy.
- You don't want many such options in a production embedded system.
- Some hardening features are meant to find issues in kernel code,
  but have a significant performance cost.
  You may remove them in production systems after running robustness tests.
- Not all recommended security features are relevant:
  - Some of them may not be available on your hardware
  - You may not need them in your specific system.

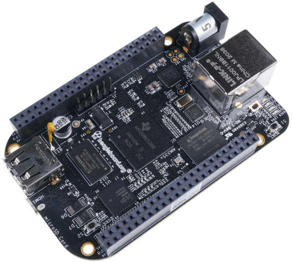
Demo: Kernel Hardening on Beagle Bone Black

- The only hardware board officially supported by Yocto (`meta-yocto-bsp` layer)
- Before hardening:
  - Compressed kernel size: 5.5 MB
  - Number of OK: 159
  - Number of FAIL: 125
- After hardening:
  - Compressed kernel size: 4.7 MB
    Note: minimal effort to reduce kernel size
  - Number of OK: 278
  - Number of FAIL: 6 🎉



Demo: Kernel Hardening on Beagle Bone Black

- The only hardware board officially supported by Yocto (`meta-yocto-bsp` layer)
- Before hardening:
  - Compressed kernel size: 5.5 MB
  - Number of OK: 159
  - Number of FAIL: 125
- After hardening:
  - Compressed kernel size: 4.7 MB
    Note: minimal effort to reduce kernel size
  - Number of OK: 278
  - Number of FAIL: 6 🎉
    which are actually OK for me 😉



Final Report

Can't select these on my platform

```
=========================================================================================================
            option_name               | type  |      reason      | decision |desired_val | check_result
=========================================================================================================
CONFIG_STACKPROTECTOR_PER_TASK       |kconfig| self_protection  |defconfig |     y      | FAIL: is not found
CONFIG_UBSAN_TRAP                    |kconfig| self_protection  |   kspp   |     y      | FAIL: CONFIG_UBSAN_ENUM is not "is not set"
CONFIG_ARM_SMMU                      |kconfig| self_protection  |a13xp0p0v |     y      | FAIL: is not found
CONFIG_ARM_SMMU_DISABLE_BYPASS_BY_DEFAULT|kconfig| self_protection  |a13xp0p0v |     y      | FAIL: is not found
CONFIG_MODULES                       |kconfig|cut_attack_surface|   kspp   | is not set | FAIL: "y"
kernel.modules_disabled              |sysctl |cut_attack_surface|   kspp   |     1      | FAIL: "0"

[+] Config check is finished: 'OK' - 278 (suppressed in output) / 'FAIL' - 6
```

Want to keep modules. Safer with module signing.

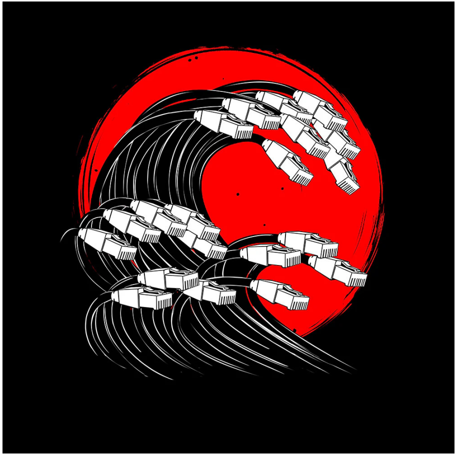
Wrapping Up

What To Remember

- `kernel-hardening-checker` from Alex Popov is your best friend.
  Find it in the `meta-oe` layer (`meta-openembedded` repository)
- Default kernel configurations are developer friendly, not secure enough
- There is lot of functionality you don't need in an embedded system
  This helps to reduce the attack surface
- Pretty easy to increase security without sacrificing functionality
  and performance (not substantially)
- Don't apply recommendations blindly
- This protects against future vulnerabilities.
  But don't forget to fix known ones!

Useful Resources

- Michael Opdenacker — Securing Yocto Built Systems
  https://rootcommit.com/pub/conferences/2025/elce/yocto-security/yocto-security.pdf
- Root Commit Blog — Yocto Security: Kernel Hardening
  https://rootcommit.com/2025/yocto-security-kernel-hardening/
- Root Commit's Yocto Project training course
  https://rootcommit.com/training/yocto
- Esa Jääskelä — Yocto Hardening
  Series of blog posts about many aspects of the topic
  Fun and exhaustive!
  https://ejaaskel.dev/yocto-hardening/



Darknet Diaries podcast shop:
https://shop.darknetdiaries.com

For Further Details... 😉

root commit

Booth 4-648

- Have Yocto goodies: coffee ☕, cookies 🍪
- Our best swag: our smiles 😀
- Come and see how welcoming the Yocto community is.

Questions? Comments?

- mo@rootcommit.com
- https://fosstodon.org/@MichaelOpdenacker
- XMPP: omichael@conversations.im
- Signal: rootcommit.01
- Slides available under the CC-By-SA 4.0 license
  https://rootcommit.com/pub/conferences/2026/embedded-world/yocto-kernel-hardening/
- Sources (LaTeX):
  https://gitlab.com/rootcommit/yocto-kernel-hardening/
- Code used in this demo:
  https://gitlab.com/rootcommit/meta-kernel-hardening-demo