CLIP OS: Building a defense-in-depth OS around Linux kernel security improvements

Timothée Ravier, Mickaël Salaün

Agence nationale de la sécurité des systèmes d'information (ANSSI)

September 26, 2018

About the ANSSI

- *Agence nationale de la sécurité des systèmes d'information*
- French authority in the area of cyberdefence, network and information security
- We are **not** an intelligence agency

Overview

CLIP OS ?

- Linux distribution developed by the ANSSI
- Initially only available internally
- Now open source, mostly under the LGPL v2.1+
- Code and issue tracker hosted on GitHub:
  - Version 4: available as reference and for upstream patch contribution[1]
  - Version 5: currently developed version, alpha status[2]

[1] https://github.com/CLIPOS-Archive
[2] https://github.com/CLIPOS

Hardened OS

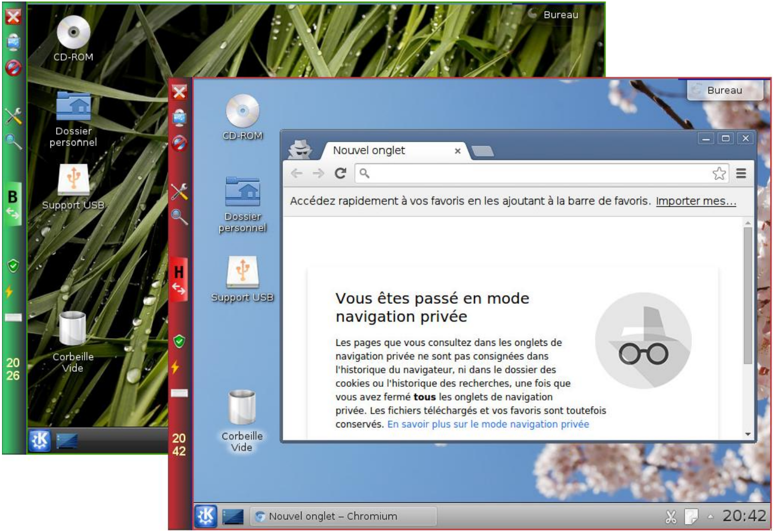
- Hardened Linux kernel and userspace
- Confined services
- "Unprivileged" admin, audit and update roles:

  ⇒ the *root* account is not usable
- Automatic updates using A/B partition model (similar to Android 7+)

Multilevel security OS

- Provide two isolated user environments: *low* and *high*
- Interactions follow the Bell-LaPadula model:
  - Write up: upload documents from *low* to *high*
  - Read down: *high* has read only access to untrusted USB devices
  - Trusted write down: encrypt documents from *high* to write them in *low*
- Level *high* can only access network through a VPN
- Per level user device assignment

Multilevel from the end user point of view

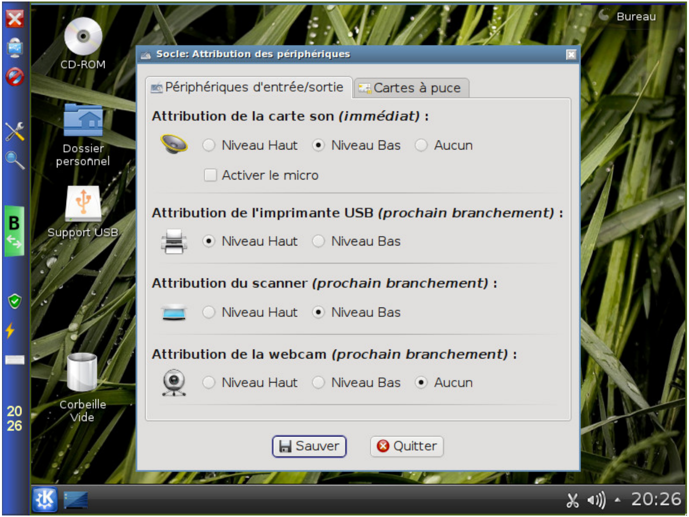


Accédez rapidement à vos favoris en les ajoutant à la barre de favoris. Importer mes...

Vous êtes passé en mode navigation privée

Les pages que vous consultez dans les onglets de navigation privée ne sont pas consignées dans l'historique du navigateur, ni dans le dossier des cookies ou l'historique des recherches, une fois que vous avez fermé **tous** les onglets de navigation privée. Les fichiers téléchargés et vos favoris sont toutefois conservés. En savoir plus sur le mode navigation privée

Admin panel: devices assignment per level



Socle: Attribution des périphériques

Périphériques d'entrée/sortie | Cartes à puce

Attribution de la carte son *(immédiat)* :

○ Niveau Haut ● Niveau Bas ○ Aucun

□ Activer le micro

Attribution de l'imprimante USB *(prochain branchement)* :

● Niveau Haut ○ Niveau Bas

Attribution du scanner *(prochain branchement)* :

○ Niveau Haut ● Niveau Bas

Attribution de la webcam *(prochain branchement)* :

○ Niveau Haut ○ Niveau Bas ● Aucun

Sauver Quitter

Differences with Qubes OS

CLIP OS development began 5 years earlier than Qubes OS

Differences with Qubes OS

CLIP OS development began 5 years earlier than Qubes OS

Goal of CLIP OS

- We target non-expert users
- Bell-LaPadula model with two levels
- We favor a defense-in-depth approach

Differences with Qubes OS

CLIP OS development began 5 years earlier than Qubes OS

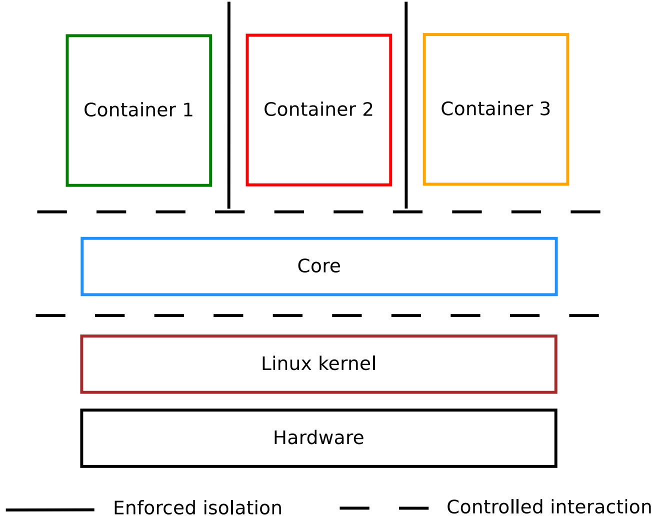
Goal of CLIP OS

- We target non-expert users
- Bell-LaPadula model with two levels
- We favor a defense-in-depth approach

Technical point of view

- Hypervisor vs. supervisor isolation
- Limited access right, even for the administrator

Architecture



CLIP OS 4

Hardening mechanisms

Gentoo Hardened

- Hardened toolchain
- Flexible patching

Hardening mechanisms

Gentoo Hardened

- Hardened toolchain
- Flexible patching

Linux-VServer

- Linux namespaces with additional constraints
- Unique container and network IDs: XIDs and NIDs

Hardening mechanisms

Gentoo Hardened

- Hardened toolchain
- Flexible patching

Linux-VServer

- Linux namespaces with additional constraints
- Unique container and network IDs: XIDs and NIDs

grsecurity/PaX

- Kernel self-protection (e.g., memory protection, CFI)
- Multiple userspace hardening features (e.g., chroot, TPE)

Hardening mechanisms

Gentoo Hardened

- Hardened toolchain
- Flexible patching

Linux-VServer

- Linux namespaces with additional constraints
- Unique container and network IDs: XIDs and NIDs

grsecurity/PaX

- Kernel self-protection (e.g., memory protection, CFI)
- Multiple userspace hardening features (e.g., chroot, TPE)

CLIP LSM

- Complement the Linux permission model
- Leverage Linux-VServer and grsecurity/PaX

Write ⊕ Execute policy

Avoid arbitrary code execution and persistent attacks, improve multilevel isolation

Write $\oplus$ Execute policy

Avoid arbitrary code execution and persistent attacks, improve multilevel isolation

Memory (PaX)

Deny writable memory to be executable, throughout the system lifetime

Write ⊕ Execute policy

Avoid arbitrary code execution and persistent attacks, improve multilevel isolation

Memory (PaX)

Deny writable memory to be executable, throughout the system lifetime

Devctl

Enforce and extend W ⊕ X from devices to mount points

Write ⊕ Execute policy

Avoid arbitrary code execution and persistent attacks, improve multilevel isolation

Memory (PaX)

Deny writable memory to be executable, throughout the system lifetime

Devctl

Enforce and extend W ⊕ X from devices to mount points

Mount points

Enforce W ⊕ X thanks to mount options: $\overline{\texttt{ro}} \oplus \overline{\texttt{noexec}}$

Write ⊕ Execute policy

Avoid arbitrary code execution and persistent attacks, improve multilevel isolation

Memory (PaX)

Deny writable memory to be executable, throughout the system lifetime

Devctl

Enforce and extend W ⊕ X from devices to mount points

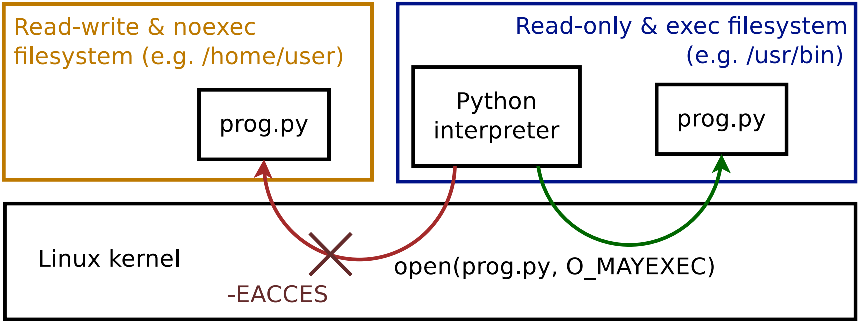
Mount points

Enforce W ⊕ X thanks to mount options: $\overline{\texttt{ro}} \oplus \overline{\texttt{noexec}}$

The O_MAYEXEC flag

Enforce and extend W ⊕ X from mount points to scripts (via interpreters)

O_MAYEXEC


Read-write & noexec filesystem (e.g. /home/user)
prog.py
Read-only & exec filesystem (e.g. /usr/bin)
Python interpreter
prog.py
Linux kernel
-EACCES
open(prog.py, O_MAYEXEC)

Partitioning

Hardened containers

- Leverage Linux-VServer *admin* and *watch* (audit) concepts
- New capability bounding sets: for root and per container
- Hardened chroot

Partitioning

Hardened containers

- Leverage Linux-VServer *admin* and *watch* (audit) concepts
- New capability bounding sets: for root and per container
- Hardened chroot

Container content and interaction

- Tailored filesystem layouts per service
- Container management with `vsctl` and `clip-libvserver` (self-jailing)

Veriexec and permissions (CLIP-LSM)

Goal

- Split Linux capabilities (e.g., Fuse, unshare)
- Add new permissions (e.g., network, XFRM)
- Can be tied to an XID
- Does not use xattr (thus independent from the filesystem)

Veriexec and permissions (CLIP-LSM)

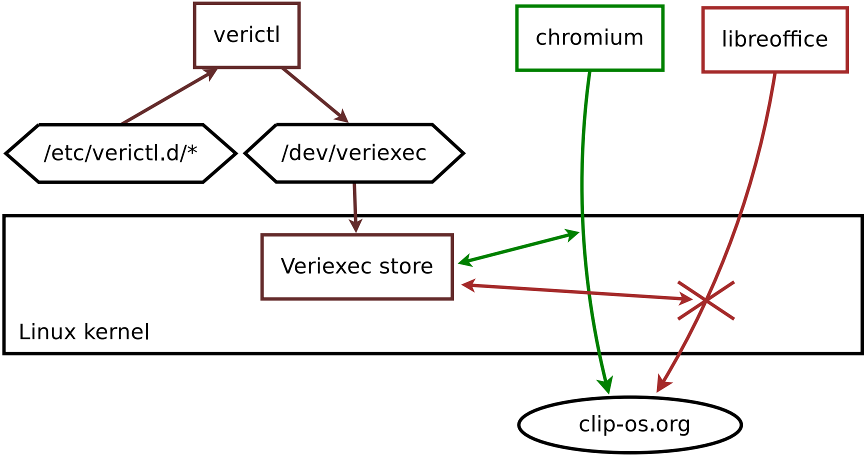
Goal

- Split Linux capabilities (e.g., Fuse, unshare)
- Add new permissions (e.g., network, XFRM)
- Can be tied to an XID
- Does not use xattr (thus independent from the filesystem)

Configuration example: `/etc/verictl.d/chromium`

```
/usr/.../chrome-sandbox 1002 e
  SETUID|SETGID|SYS_CHROOT SETUID|SETGID|SYS_CHROOT -
  cUP sha256 45bcbd1...
```

Veriexec example



CLIP OS 5

General Linux kernel hardening

- Strict whitelist of kernel options, but easily composable sets
- Paranoid command line
  - iommu=force, pti=on, spectre_v2=on, etc.
- Strict sysctl defaults
  - kernel.kptr_restrict, kernel.yama.ptrace_scope, etc.

Enabling Linux kernel hardening

Goals

- Protecting the kernel from itself and from userspace
- Include additional features for userspace
- Being able to test kernel and userspace coordinated changes

Enabling Linux kernel hardening

Goals

- Protecting the kernel from itself and from userspace
- Include additional features for userspace
- Being able to test kernel and userspace coordinated changes

Security may come first

- We can handle minor compatibility breakage in our userspace
- Will accept changes that upstream may reject

Enabling Linux kernel hardening

Goals

- Protecting the kernel from itself and from userspace
- Include additional features for userspace
- Being able to test kernel and userspace coordinated changes

Security may come first

- We can handle minor compatibility breakage in our userspace
- Will accept changes that upstream may reject

Interaction with upstream & KSPP

- Include in-progress or ready-for-upstream patches
- Integrate and validate patches in a single tree
- Maintain hardening patches for latest stable kernel

Patch series: linux-hardened

Features

- Memory hardening improvements, including:
  - better userspace ASLR
  - slab allocators hardening (mostly SLUB)
  - simpler page sanitizing
- Various restrictions: TIOCSTI ioctl, perf subsystem, device timing side channels, etc.
- Miscellaneous additions: more BUG_ONs, more __ro_after_init, etc.

Patch series: linux-hardened

Features

- Memory hardening improvements, including:
  - better userspace ASLR
  - slab allocators hardening (mostly SLUB)
  - simpler page sanitizing
- Various restrictions: TIOCSTI ioctl, perf subsystem, device timing side channels, etc.
- Miscellaneous additions: more BUG_ONs, more __ro_after_init, etc.

- Development status: **In progress**
- CLIP OS status: **Merged**
- Upstream status: **Most changes unlikely to be merged upstream**

Upstream contribution integration: Lockdown

Features

- Reduce options for root to run untrusted code in kernel context

Upstream contribution integration: Lockdown

Features

- Reduce options for root to run untrusted code in kernel context

- Development status: **Feature complete**
- CLIP OS status: **Merged**
- Upstream status: **Ready for upstream integration**

Upstream contribution integration: STACKLEAK

Features

- Reduce information leaks and block attacks using uninitialized kernel stack variables:
  - Erase the stack before returning from system calls
- Improve runtime detection of kernel stack overflows (e.g. Stack Clash):
  - Instrument calls to alloca()

Upstream contribution integration: STACKLEAK

Features

- Reduce information leaks and block attacks using uninitialized kernel stack variables:
  - Erase the stack before returning from system calls
- Improve runtime detection of kernel stack overflows (e.g. Stack Clash):
  - Instrument calls to alloca()

CLIP OS specific changes

- Kept alloca()-related changes (dropped for upstream in v15)

Upstream contribution integration: STACKLEAK

Features

- Reduce information leaks and block attacks using uninitialized kernel stack variables:
  - Erase the stack before returning from system calls
- Improve runtime detection of kernel stack overflows (e.g. Stack Clash):
  - Instrument calls to alloca()

CLIP OS specific changes

- Kept alloca()-related changes (dropped for upstream in v15)

- Development status: **Feature complete**
- CLIP OS status: **Merged**
- Upstream status: **Ready for upstream integration**

Upstream contribution: Landlock

Features

- Enables *seccomp-bpf*-like self-sandboxing for unprivileged processes
- Stackable LSM
- Powered by eBPF
- Dynamic filesystem access control using whitelists & blacklists
- See landlock.io

Upstream contribution: Landlock

Features

- Enables *seccomp-bpf*-like self-sandboxing for unprivileged processes
- Stackable LSM
- Powered by eBPF
- Dynamic filesystem access control using whitelists & blacklists
- See landlock.io

- Development status: **Initial feature set ready**
- CLIP OS status: **Planned**
- Upstream status: **Work in progress**

Upstream contribution: VServer-like LSM

Features

- Adds a single kernel enforced indentifier for confined environments
- Similar in principle to VServer XID or to "Container IDs"
- Inspired by the VServer patch
- Integrated as a stackable LSM

Upstream contribution: VServer-like LSM

Features

- Adds a single kernel enforced indentifier for confined environments
- Similar in principle to VServer XID or to "Container IDs"
- Inspired by the VServer patch
- Integrated as a stackable LSM

- Development status: **Early development stage**
- CLIP OS status: **Planned**

Conclusion

Take away

- Hardened Linux distro and kernel
- Coordinated userspace and kernelspace
- Support multilevel security

Conclusion

Take away

- Hardened Linux distro and kernel
- Coordinated userspace and kernelspace
- Support multilevel security

Ongoing project

- Contributions welcome
- Browse the doc and the sources to find more interesting features: docs.clip-os.org

Thanks!

🌎 clip-os.org ✉ clipos@ssi.gouv.fr

🌎 v4: github.com/CLIPOS-Archive

🌎 v5: github.com/CLIPOS

We're hiring! (but not directly for CLIP OS)

Linux system security expert

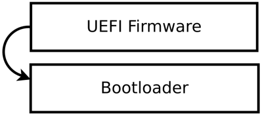
https://www.ssi.gouv.fr/emploi/expert-en-securite-des-systemes-linux/

Boot chain and root partition integrity protection

1. UEFI Secure Boot support:
   - Custom keys (i.e. not signed by Microsoft)
   - Requires enrollment in hardware

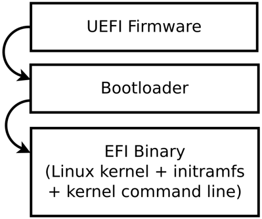
Boot chain and root partition integrity protection

1. UEFI Secure Boot support:
   - Custom keys (i.e. not signed by Microsoft)
   - Requires enrollment in hardware
2. Minimal bootloader (gummiboot/systemd-boot)



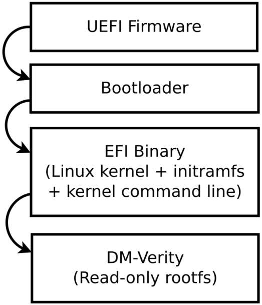
Boot chain and root partition integrity protection

1. UEFI Secure Boot support:
   - Custom keys (i.e. not signed by Microsoft)
   - Requires enrollment in hardware
2. Minimal bootloader (gummiboot/systemd-boot)
3. EFI bundle:
   - Linux kernel
   - initramfs
   - kernel command line



Boot chain and root partition integrity protection

1. UEFI Secure Boot support:
   - Custom keys (i.e. not signed by Microsoft)
   - Requires enrollment in hardware
2. Minimal bootloader (gummiboot/systemd-boot)
3. EFI bundle:
   - Linux kernel
   - initramfs
   - kernel command line
4. DM-Verity partition:
   - DM-Verity root hash set in kernel command line
   - Forward error correction support (FEC)
   - Read only uncompressed SquashFS root filesystem


UEFI Firmware
Bootloader
EFI Binary
(Linux kernel + initramfs
+ kernel command line)
DM-Verity
(Read-only rootfs)