Microsoft

Landlock: From a security mechanism idea to a widely available implementation

SSTIC

Mickaël Salaün – Linux kernel maintainer

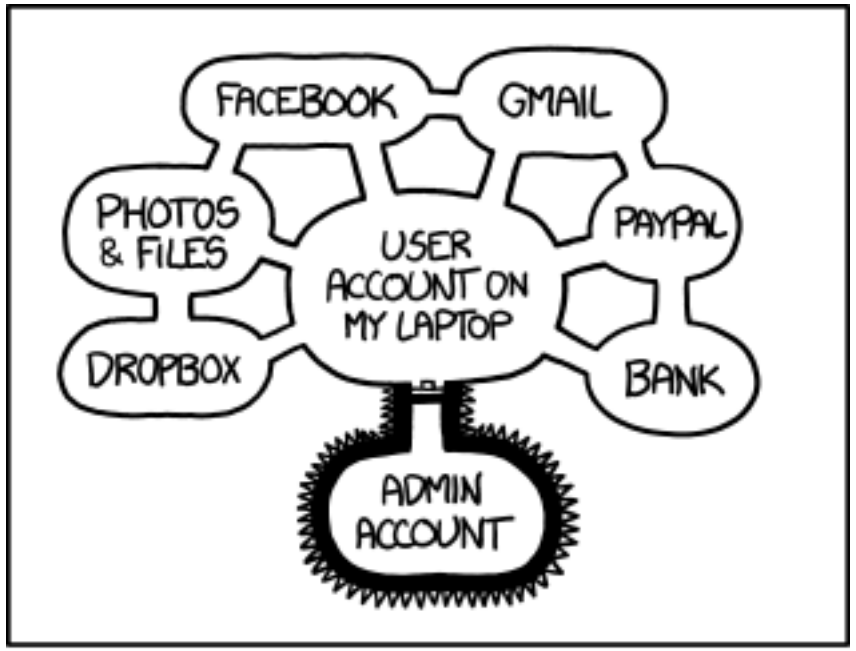
2024-06-06

Agenda

1. Problem statement
2. Security sandboxing
3. Landlock properties
4. Landlock interface
5. Upstreaming and adoption

Problem statement

Goal: protect data



IF SOMEONE STEALS MY LAPTOP WHILE I'M LOGGED IN, THEY CAN READ MY EMAIL, TAKE MY MONEY, AND IMPERSONATE ME TO MY FRIENDS,

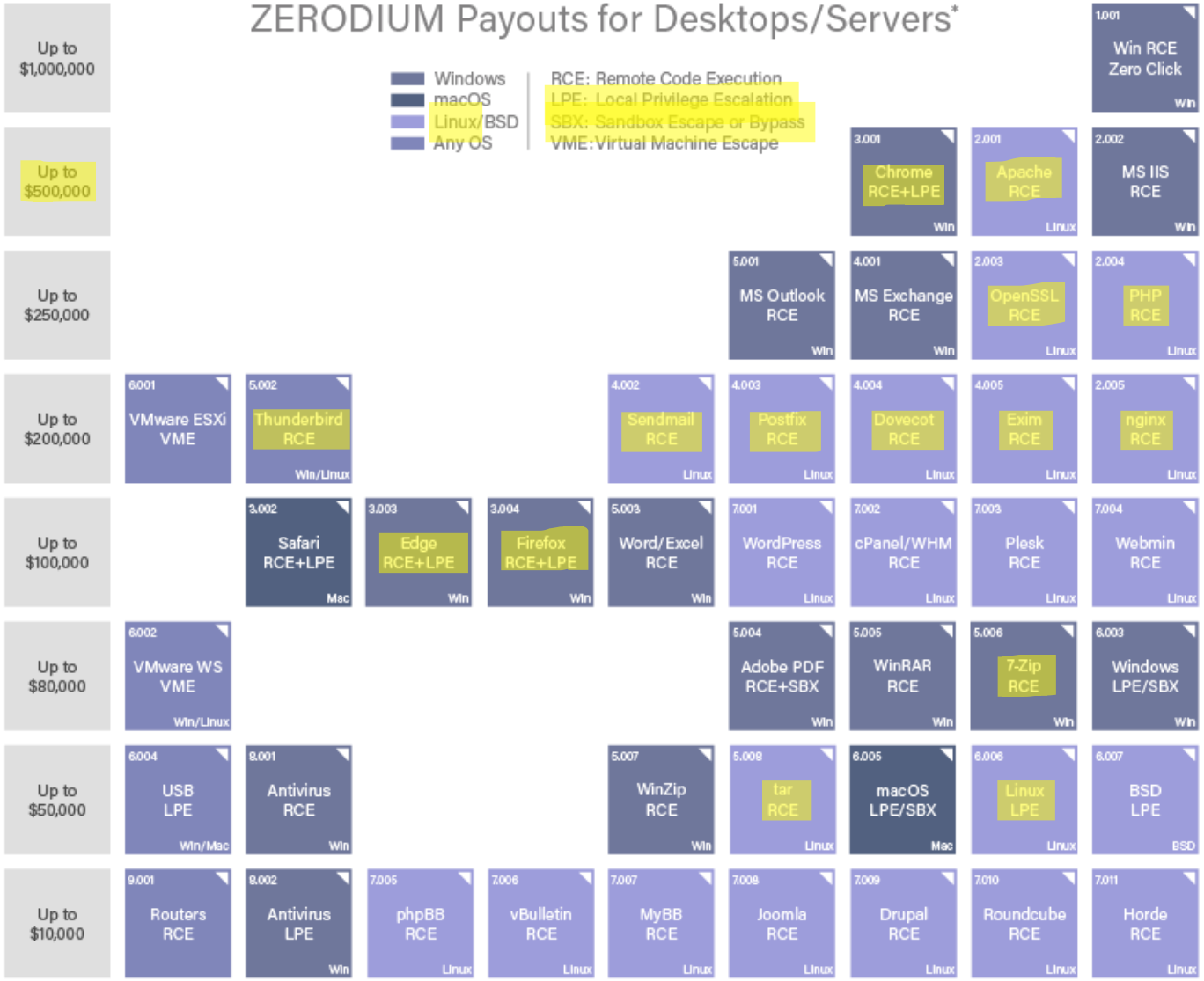
BUT AT LEAST THEY CAN'T INSTALL DRIVERS WITHOUT MY PERMISSION.

https://xkcd.com/1200

Pragmatic statements

1. An innocuous and trusted process can **become malicious during its lifetime** because of bugs exploited by attackers.
2. There are multiple and **different levels of trust** and different **consequences** in case of a breach: system, user, app data…

Attack cost


ZERODIUM Payouts for Desktops/Servers*
Windows
macOS
Linux/BSD
Any OS
RCE: Remote Code Execution
LPE: Local Privilege Escalation
SBX: Sandbox Escape or Bypass
VME: Virtual Machine Escape
Up to $1,000,000
1.001 Win RCE Zero Click Win
Up to $500,000
3.001 Chrome RCE+LPE Win
2.001 Apache RCE Linux
2.002 MS IIS RCE Win
Up to $250,000
5.001 MS Outlook RCE Win
4.001 MS Exchange RCE Win
2.003 OpenSSL RCE Linux
2.004 PHP RCE Linux
Up to $200,000
6.001 VMware ESXi VME
5.002 Thunderbird RCE Win/Linux
4.002 Sendmail RCE Linux
4.003 Postfix RCE Linux
4.004 Dovecot RCE Linux
4.005 Exim RCE Linux
2.005 nginx RCE Linux
Up to $100,000
3.002 Safari RCE+LPE Mac
3.003 Edge RCE+LPE Win
3.004 Firefox RCE+LPE Win
5.003 Word/Excel RCE Win
7.001 WordPress RCE Linux
7.002 cPanel/WHM RCE Linux
7.003 Plesk RCE Linux
7.004 Webmin RCE Linux
Up to $80,000
6.002 VMware WS VME Win/Linux
5.004 Adobe PDF RCE+SBX Win
5.005 WinRAR RCE Win
5.006 7-Zip RCE Win
6.003 Windows LPE/SBX Win
Up to $50,000
6.004 USB LPE Win/Mac
8.001 Antivirus RCE Win
5.007 WinZip RCE Win
5.008 tar RCE Linux
6.005 macOS LPE/SBX Mac
6.006 Linux LPE Linux
6.007 BSD LPE BSD
Up to $10,000
9.001 Routers RCE
8.002 Antivirus LPE Win
7.005 phpBB RCE Linux
7.006 vBulletin RCE Linux
7.007 MyBB RCE Linux
7.008 Joomla RCE Linux
7.009 Drupal RCE Linux
7.010 Roundcube RCE Linux
7.011 Horde RCE Linux
* All payouts are subject to change or cancellation without notice. All trademarks are the property of their respective owners.
2019/01 © zerodium.com

Security sandboxing

What is sandboxing?

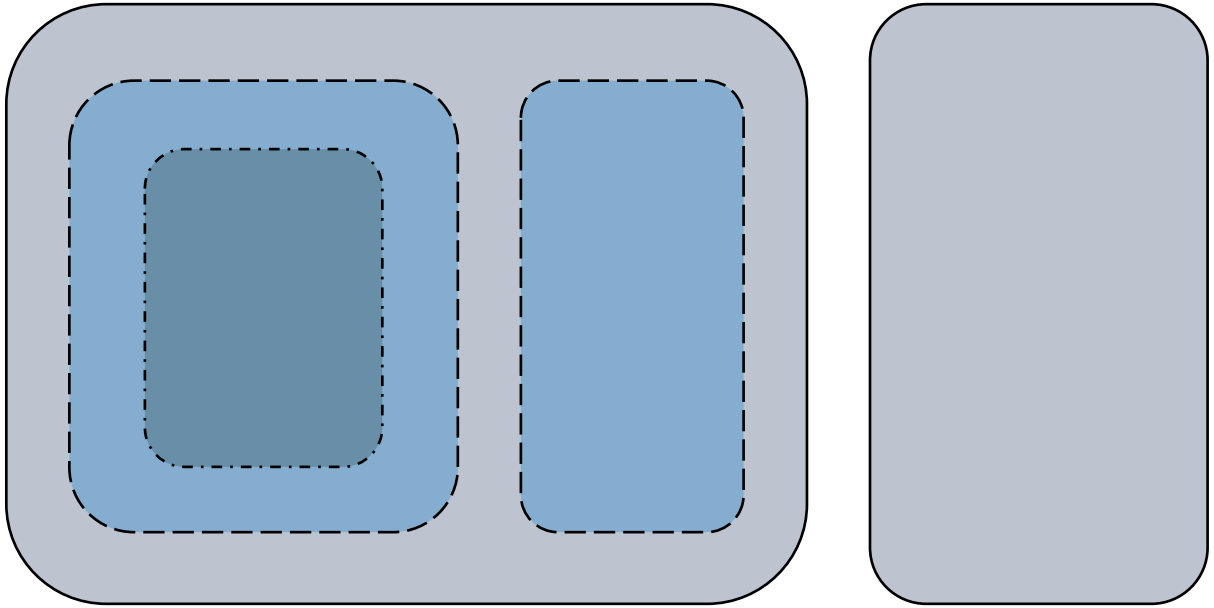
"A **restricted**, controlled **execution environment** that prevents potentially malicious software [...] from accessing any system resources except those for which the software is authorized."

Tailored and embedded security policy

Developers are in the best position to reason about the required **accesses** according to **legitimate** behaviors:

- Application semantics
- Static and dynamic configuration
- Interactions

Dynamic policy composition



Safe security mechanism

Principle of least privilege

- No privileged accounts or services
- No SUID binaries

Innocuous access control

- Only increase restrictions

Protecting against bypasses

- Each process should be protected from less-privileged ones

Non-Linux systems

Main sandbox mechanisms:

- XNU Sandbox (iOS)
- Pledge and Unveil (OpenBSD)
- Capsicum (FreeBSD)
- AppContainer (Windows)

Candidates for a sandboxing mechanism

	Performance	Fine-grained control	Embedded policy	Unprivileged use
Virtual Machine	✗	✗	✗	✗
SELinux	✓	✓	✗	✗
namespaces	✓	✗	✓	!
seccomp	✓	✗	✓	✓
Landlock	✓	✓	✓	✓

✓ Yes, compared to others

✗ No, compared to others

! In some way, but with limitations

Landlock properties

Use case #1

Untrusted applications: protect from potentially malicious third-party code.

Candidates:

- Container runtimes
- Init systems

Use case #2

Exploitable bugs in trusted applications: protect from vulnerable code maintained by developers.

Candidates:

- Parsers: archive tools, file format conversion, renderers…
- Web browsers
- Network and system services

Landlock empowers developers

New unprivileged **security layers**

Lockless concurrent development: avoid policy management bottleneck

Set of **small policies**: easier to maintain and audit

Testable with a **CI** and synchronized with app **semantic**: stable

How Landlock works?

Restrict ambient rights according to the **kernel semantic** (e.g., global filesystem access) for a set of processes, thanks to 3 **dedicated syscalls**.

Security policies are inherited by all new children processes.

A one-way set of restrictions: cannot be disabled once enabled.

Current access control

Implicit restrictions

- Process impersonation (e.g., ptrace)
- Filesystem topology changes (e.g., mounts)

Explicit access rights

- Filesystem
- Networking

Current filesystem access rights

- Execute, read or write to a file
- List a directory or remove files
- Create files according to their type
- Rename or link files
- IOCTL commands to devices

Current networking access rights

- Connect to a TCP port
- Bind to a TCP port

Landlock interface

Step 1: Check backward compatibility

```
int abi = landlock_create_ruleset(NULL, 0, LANDLOCK_CREATE_RULESET_VERSION);

if (abi < 0)
    return 0;
```

Step 2: Create a ruleset

```
int ruleset_fd;
struct landlock_ruleset_attr ruleset_attr = {
    .handled_access_fs =
        LANDLOCK_ACCESS_FS_EXECUTE |
        LANDLOCK_ACCESS_FS_WRITE_FILE |
        […]
        LANDLOCK_ACCESS_FS_MAKE_REG,
};

ruleset_fd = landlock_create_ruleset(&ruleset_attr, sizeof(ruleset_attr), 0);
if (ruleset_fd < 0)
    error_exit("Failed to create a ruleset");
```

Step 3: Add rules

```
int err;
struct landlock_path_beneath_attr path_beneath = {
    .allowed_access = LANDLOCK_ACCESS_FS_EXECUTE | […] ,
};

path_beneath.parent_fd = open("/usr", O_PATH | O_CLOEXEC);
if (path_beneath.parent_fd < 0)
    error_exit("Failed to open file");

err = landlock_add_rule(ruleset_fd, LANDLOCK_RULE_PATH_BENEATH, &path_beneath, 0);
close(path_beneath.parent_fd);
if (err)
    error_exit("Failed to update ruleset");
```

Step 4: Enforce the ruleset

```
if (prctl(PR_SET_NO_NEW_PRIVS, 1, 0, 0, 0))
    error_exit("Failed to restrict privileges");

if (landlock_restrict_self(ruleset_fd, 0))
    error_exit("Failed to enforce ruleset");

close(ruleset_fd);
```

Full example: https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/tree/samples/landlock/sandboxer.c

Upstreaming and adoption

History

1. Initial RFC (Mar. 2016)
2. 34 patch series with different designs: seccomp, eBPF (see SSTIC 2017), cgroups…
3. Merged in Linux 5.13 (Apr. 2021)

Why upstream?

- Contribute back
- Make features available to downstream users
- Get reviews improving quality
- Limit maintenance cost
- Get contributions

Linux development

- Largest open-source project: new release every 10 weeks involving ~2k developers, more than ~500k lines of code, with ~17k commits
- Different subsystems, different communities
- Tools: Git, emails, and a lot of scripts

Adoption requirements

Enabled **by default** on multiple distros: Ubuntu, Fedora, Arch Linux, Alpine Linux, Gentoo, Debian, chromeOS, Azure Linux, WSL2.

Working with most container runtimes: Docker, Podman, runc, LXC…

Development tools, libraries for different languages.

Adoption

Some known users: chromeOS, Azure, Cloud Hypervisor, Nomad, Polkadot, Firejail, Suricata…

Soon your applications!

- Secure Open Source Rewards
- Google Patch Rewards

Getting noticed by attackers too!

Landlock support in XZ Utils:

- 5.6.0 (2024-02-24) ✔
- 5.6.1 (2024-03-09) ✘
- 5.6.2 (2024-05-29) ✔

✓ **CMake: Fix sabotaged Landlock sandbox check.**

```
It never enabled it.
```

master

Larhzu committed on Mar 30

Showing **1 changed file** with **1 addition** and **1 deletion**.

2 CMakeLists.txt

```
@@ -1001,7 +1001,7 @@ if(NOT SANDBOX_FOUND AND ENABLE_SANDBOX MATCHES
1001  1001          #include <linux/landlock.h>
1002  1002          #include <sys/syscall.h>
1003  1003          #include <sys/prctl.h>
1004        -  .
```

Try Landlock

```
# WARNING: The "sandboxer" is a demonstration program,
# not a tool with a stable interface.

$ cargo install landlock --examples

$ sandboxer
```

Wrap-up

Roadmap

Ongoing and next steps:

- Add new access-control types: socket, signals, IPCs…
- Add audit support to ease debugging
- Develop a new sandboxer tool
- Improve adoption

See GitHub issues: landlock-lsm/linux

Contribute

- Develop new access types and tests
- Improve libraries: Rust, Go…
- Improve documentation
- Challenge implementations

Questions?

https://docs.kernel.org/userspace-api/landlock.html

GitHub: landlock-lsm/linux/issues

Past talks: https://landlock.io

landlock@lists.linux.dev

Thank you!