StemJail: Dynamic Role Compartmentalization

Mickaël Salaün, Marion Daubignard, Hervé Debar

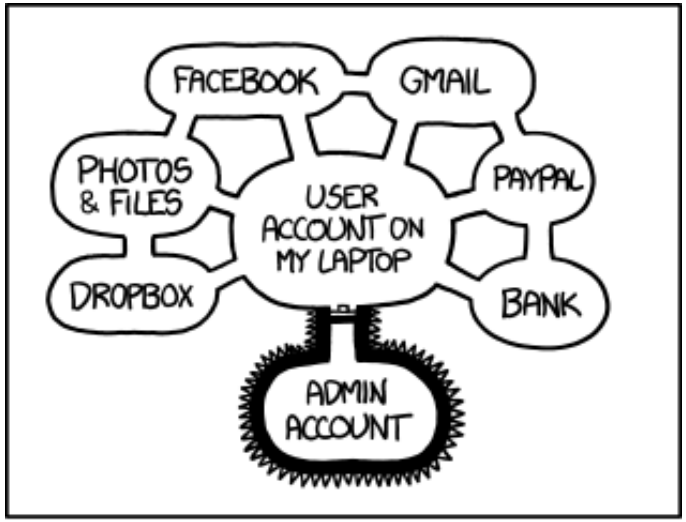
ANSSI & Télécom SudParis

June 3, 2016

Goal: User Data Protection

Threat

Malicious or compromised application: data breach



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/`

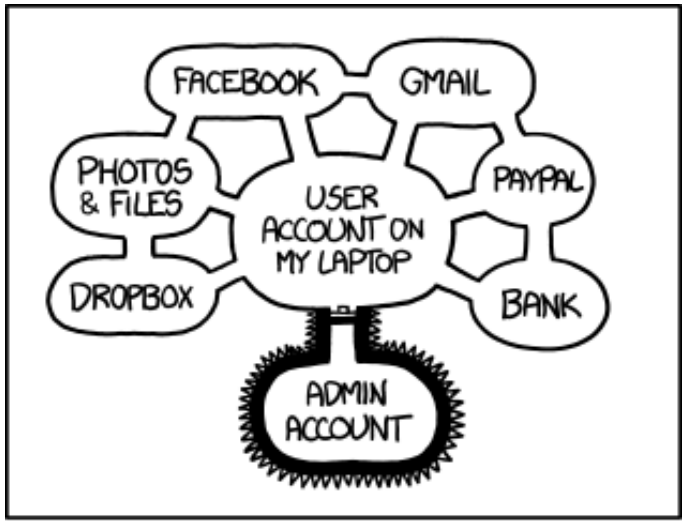
Goal: User Data Protection

Threat

Malicious or compromised application: data breach

How to Contain Breaches?

- new layer over traditional system-wide access control
- dedicated to any (unprivileged) end-user
- jail per user activity/role



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/`

Goal: User Data Protection

Threat

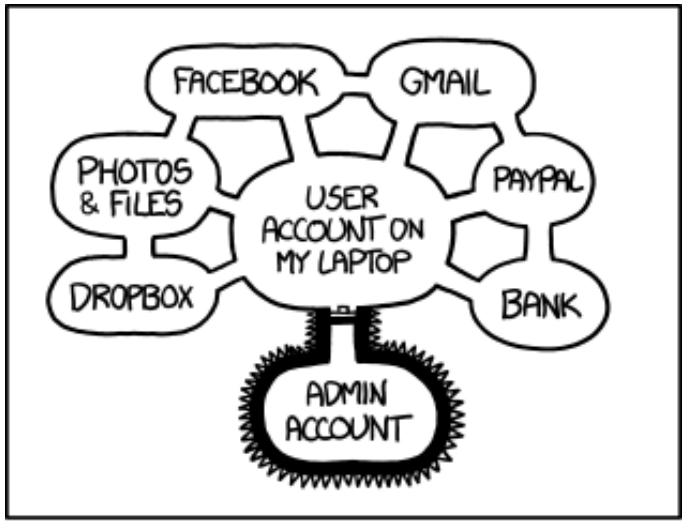
Malicious or compromised application: data breach

How to Contain Breaches?

- new layer over traditional system-wide access control
- dedicated to any (unprivileged) end-user
- jail per user activity/role

Information Diffusion Circles as Domains

- personal/professional (emails)
- private/public (pictures)
- secrets (SSH keys)
- clients X/Y/Z (documents)...



https://xkcd.com/1200/

User-Oriented: Configure then Forget

Create a Security Policy

- domains must be meaningful to the user
- filesystem-based access-control
- deny-by-default: add read-only or read-write access to file hierarchies

Integrated in the User Workflow

1. launch an application
2. use it without bothering about the domain that should be enforced

Running Example

An IT Consultant Managing Data of Multiple Clients

One domain per user activity:

- Client `OpenBar`:
  - read-write: `~/Clients/OpenBar/`
  - read-only: `~/Advertisements/`
- Client `Paranoid`:
  - read-write: `~/Clients/Paranoid/`
  - read-only: `~/Advertisements/`
- `Company` own data:
  - read-write: `~/Company/`

in addition, for every domain, relevant applications and their dependencies

Jail Internals

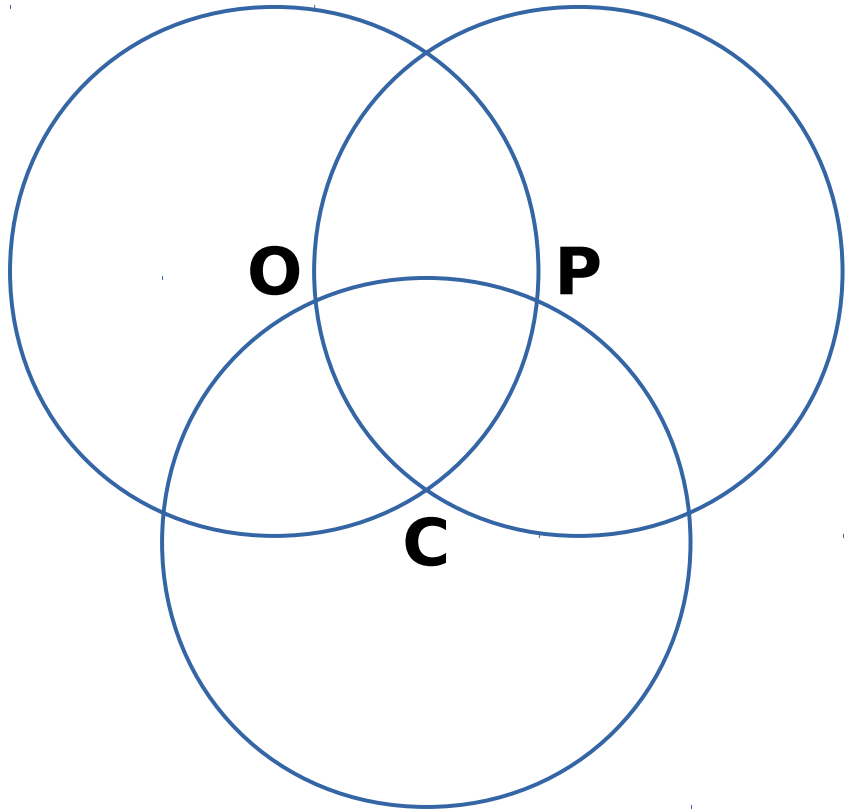
Monitor

- automaton managing a jail
- every jail starts without user data
- receives access requests from application instances

Automated Role Discovery

Domains

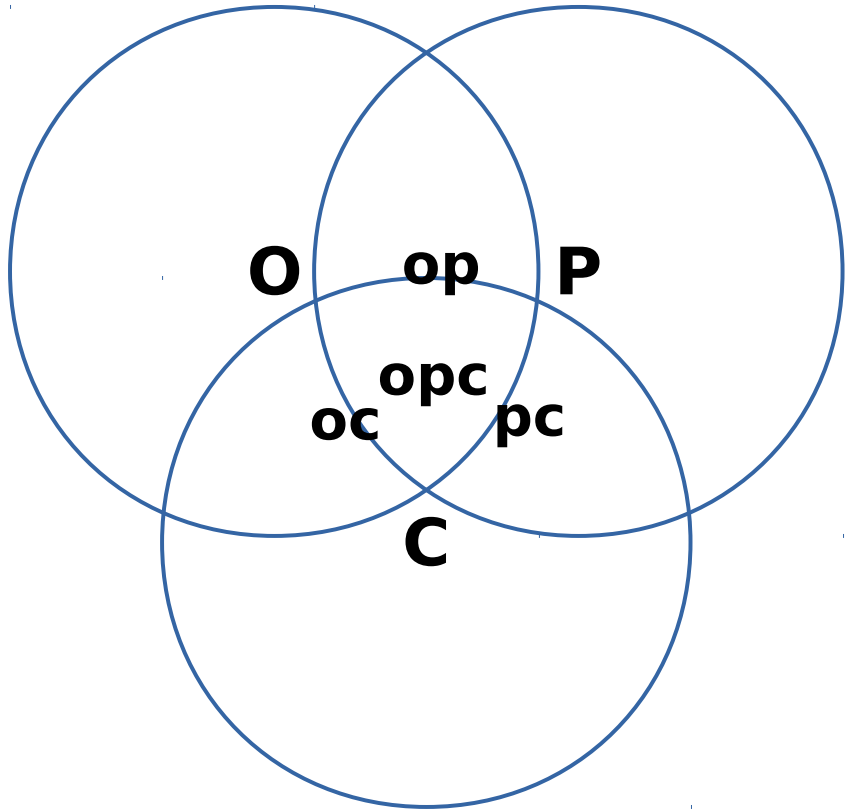
- user-defined:
  - `O`: `OpenBar`
  - `P`: `Paranoid`
  - `C`: `Company`



Automated Role Discovery

Domains

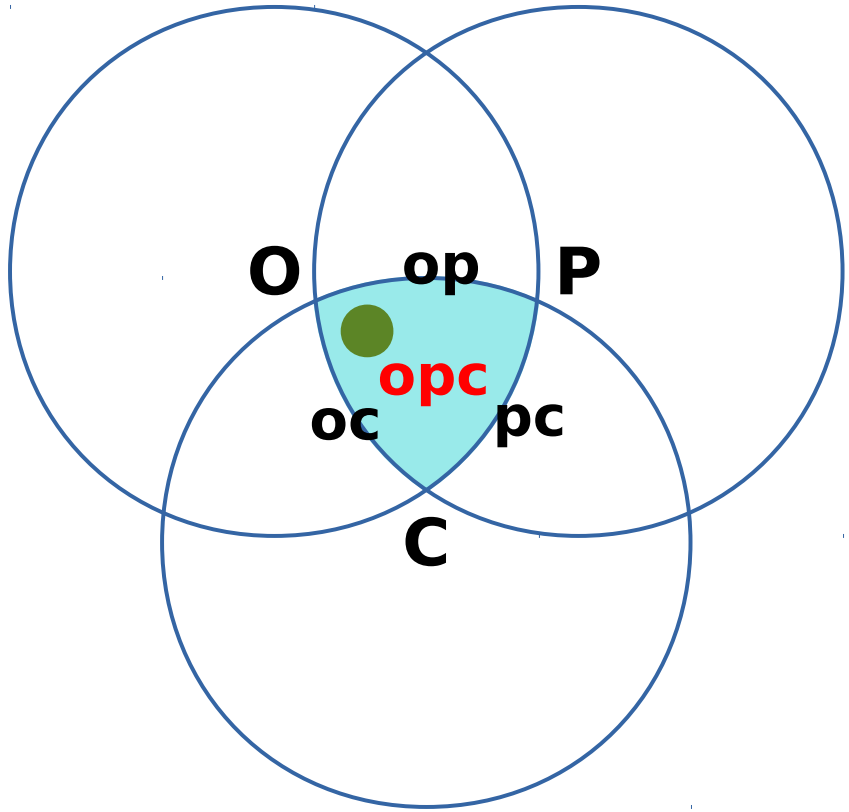
- user-defined:
  - `O`: `OpenBar`
  - `P`: `Paranoid`
  - `C`: `Company`
- automatic intersections:
  - `op`
  - `oc`
  - `pc`
  - `opc`



Automated Role Discovery

User Workflow

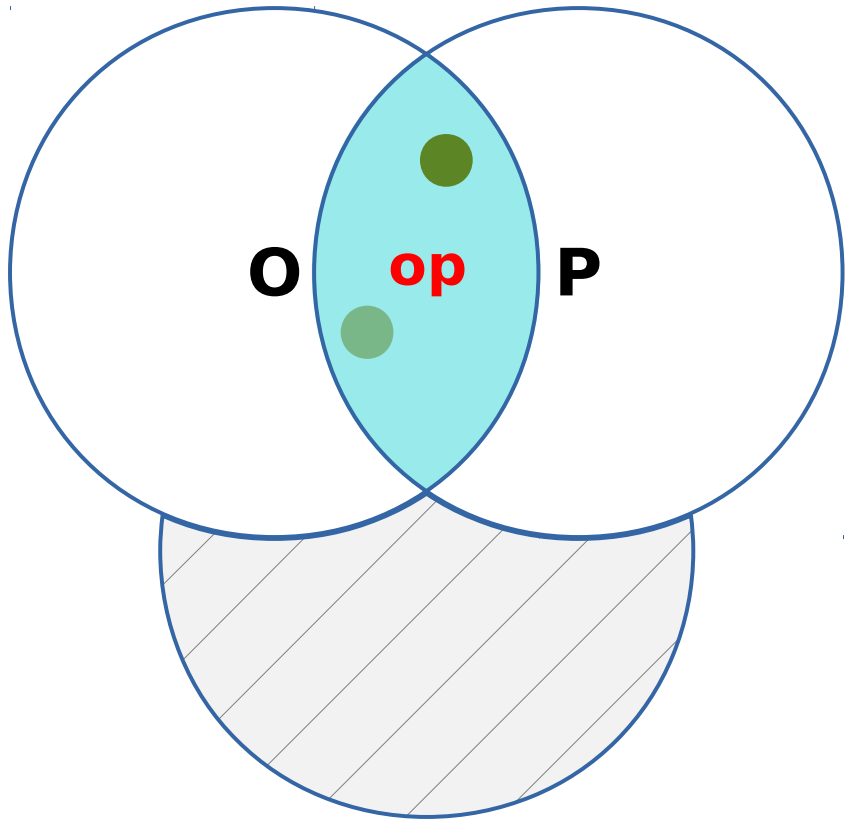
1. access `/usr/bin/viewer`



Automated Role Discovery

User Workflow

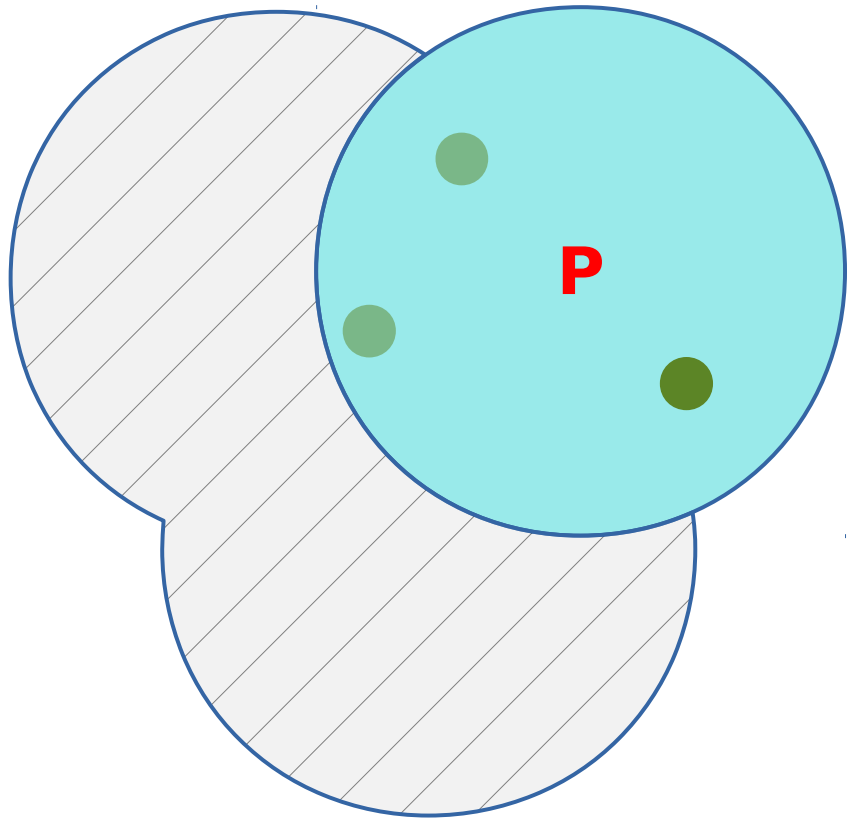
1. access `/usr/bin/viewer`
2. access `~/Advertisements/`



Automated Role Discovery

User Workflow

1. `access /usr/bin/viewer`
2. `access ~/Advertisements/`
3. `access ~/Clients/Paranoid/`



Validation

Formalization of the Monitor

Inherent property: destination domain includes source domain

Proven Security Theorem

There always exists at least one user domain granting all accesses performed by a jail.

Properties of StemJail

Relying Only on Linux User Namespaces

- meant to be accessible to any unprivileged user
- can only limit to a subset of the user accesses

Developed in the Rust Language

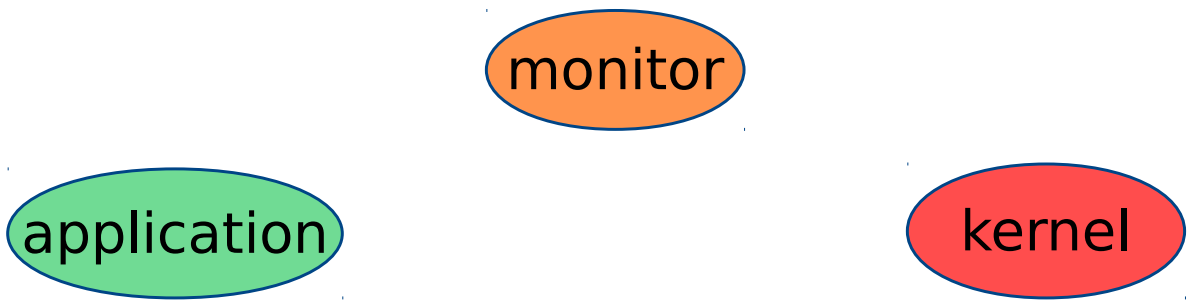
- avoid concurrency and memory-related security vulnerabilities
- can be used to create shared libraries

Application Instances

- compatibility with a preloaded shim library
- use an application local cache to limit number of requests

Access Request Workflow

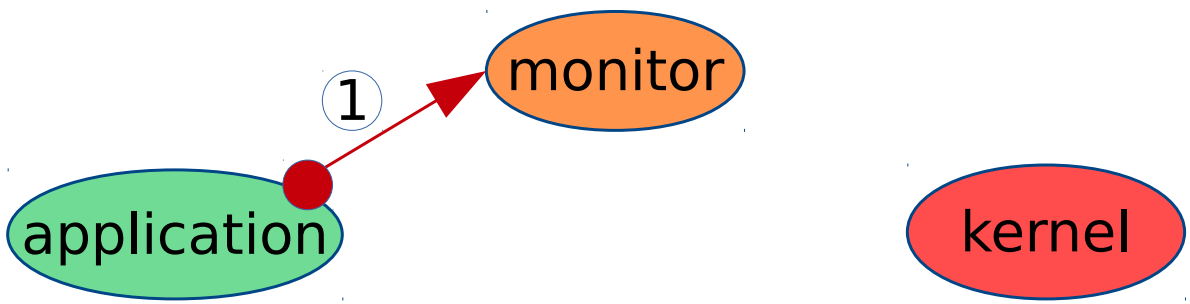
Domain Transition for One Jail



Access Request Workflow

Domain Transition for One Jail

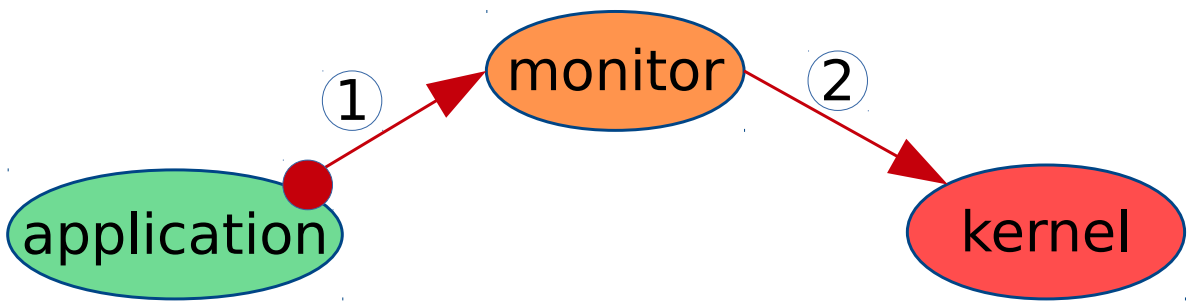
1. request new access (via hook)



Access Request Workflow

Domain Transition for One Jail

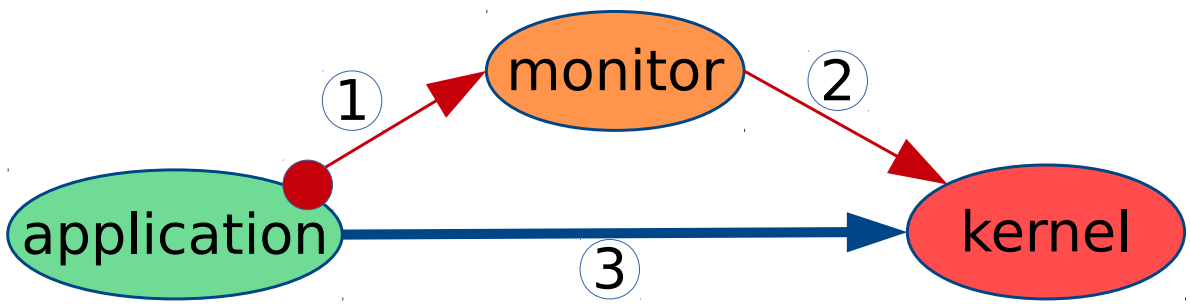
1. request new access (via hook)
2. if allowed by the policy, domain transition to grant the access



Access Request Workflow

Domain Transition for One Jail

1. request new access (via hook)
2. if allowed by the policy, domain transition to grant the access
3. subsequent accesses to the file are direct



Performance Benchmarks

Task	Normal		StemJail			
	HDD	RAM	HDD		RAM	
Gunzip	16.04s	6.13s	16.03s	0.0%	6.13s	0.0%

Table: Benchmarks for StemJail 0.4.0 (with automated role discovery)

- Gunzip: Decompressing the Linux 4.4 archive

Performance Benchmarks

Task	Normal		StemJail			
	HDD	RAM	HDD		RAM	
Gunzip	16.04s	6.13s	16.03s	0.0%	6.13s	0.0%
Untar	68.30s	1.57s	69.95s	2.4%	1.97s	25.4%

Table: Benchmarks for StemJail 0.4.0 (with automated role discovery)

- Gunzip: Decompressing the Linux 4.4 archive
- Untar: Extracting the Linux 4.4 archive

Performance Benchmarks

Task	Normal		StemJail			
	HDD	RAM	HDD		RAM	
Gunzip	16.04s	6.13s	16.03s	0.0%	6.13s	0.0%
Untar	68.30s	1.57s	69.95s	2.4%	1.97s	25.4%
Build	1137.38s	1134.10s	1190.16s	4.6%	1188.02s	4.7%

Table: Benchmarks for StemJail 0.4.0 (with automated role discovery)

- Gunzip: Decompressing the Linux 4.4 archive
- Untar: Extracting the Linux 4.4 archive
- Build: Compiling Linux 4.4 with 1 job

StemJail: Wrap-up

Dynamic Access Control Fully Integrated in the User Workflow

- implemented by automated role discovery
- formalized and validated
- efficient implementation
- usable by unprivileged users

Experimental Project Under the LGPL v3 License

Source code: `https://github.com/stemjail`