FOSDEM'17

Brussels
4 & 5 February 2017

SECURITY DEVROOM

PTAGS

The module PTAGS allows to manage tags attached to processes.

The module PTAGS is built on top of the Linux Security Module (LSM) infrastructure as it exists since V4.1 with stacking* facility.

*: stacking facility provides ability to activate more than one LSM. A great work of Casey Schaufler, the author of Smack.

Why to tag ?

- Common LSM are tagging processes
    - SELinux
    - Smack
- Files can be tagged
    - Extended attributes
    - Metadata
- Managing process for a framework requires to ta provcesses

Tags are accessed through the ptags files

- /proc/<PID>/attr/ptags
- /proc/<PID>/tasks/<TID>/attr/ptags

Reading ptags files gives the tags, one by line.

Writing ptags files allows, under condition, to change the tags. It also allows querying if tags exist.

Tracking change event is possible using `inotify(7)`.

Tags are text containing fields separated by colons.
A tag can have a value.

Examples:

```
ptags:pcooky:0:set
pcooky:id=0
pcooky:0:factorK=56
```

Tags restrictions:

- no control character, no =, no *, no DEL
- no @ at start
- no : at end

Values restrictions:

- no control character, no DEL

Querying tags of process 567:

```
jo> cat /proc/567/attr/ptags
pcooky:0:factorK=56
pcooky:id=0
jo> grep ^pcooky:id= /proc/567/attr/ptags
pcooky:id=0
jo> _
```

Querying a given tag is possible using write:

```
jo> echo ‘?pcooky:id’ > /proc/567/attr/ptags
jo> echo $?
0
jo> echo ‘?help’ > /proc/567/attr/ptags
jo> echo $?
1
jo> _
```

Note: echo -n wouldn’t work

Tags are structured in fields separated by colons.
The query operation recognize global matching of tags having a given prefix.
The pattern for prefix matching is prefix:*

```
jo> echo ‘?pcooky:*’ > /proc/567/attr/ptags
jo> echo $?
0
jo> echo ‘?pcooky:id:*’ > /proc/567/attr/ptags
jo> echo $?
1
jo> _
```

pcooky:id: is not a prefix for pcooky:id

Adding tags

```
jo> cat /proc/567/attr/ptags
pcooky:0:factorK=56
pcooky:id=0
jo> echo ‘+test’ > /proc/567/attr/ptags
jo> echo ‘+arbiter’ > /proc/567/attr/ptags
jo> cat /proc/567/attr/ptags
arbiter
pcooky:0:factorK=56
pcooky:id=0
test
jo> _
```

Tags are returned sorted.

Setting values of tags

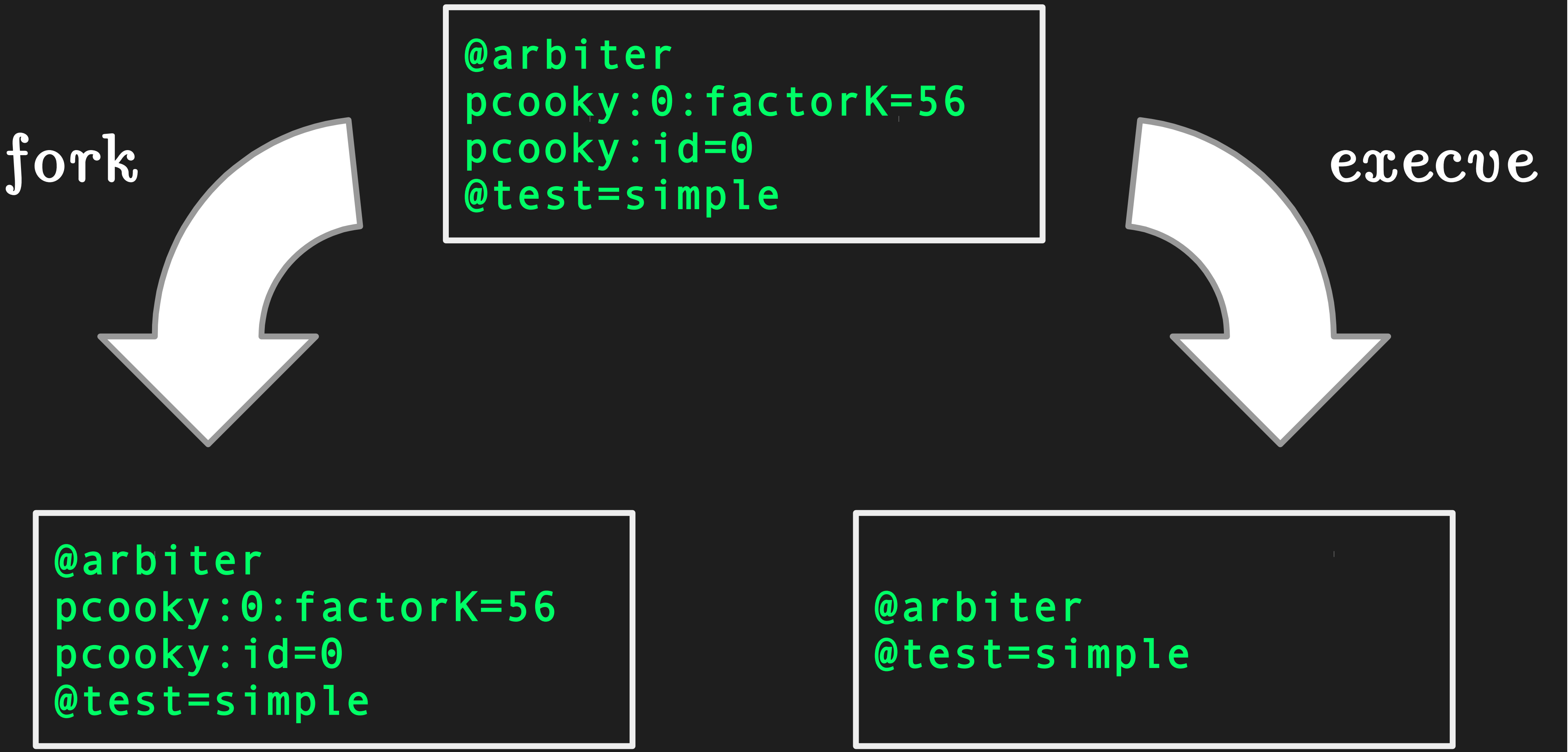
```
jo> echo ‘!test=simple’ > /proc/567/attr/ptags
jo> cat /proc/567/attr/ptags
arbiter
pcooky:0:factorK=56
pcooky:id=0
test=simple
jo> _
```

Removing value of tags

```
jo> echo ‘!test’ > /proc/567/attr/ptags
jo> cat /proc/567/attr/ptags
arbiter
pcooky:0:factorK=56
pcooky:id=0
test
jo> _
```

Removing tags

```
jo> cat /proc/567/attr/ptags
arbiter
pcooky:0:factorK=56
pcooky:id=0
test=simple
jo> echo ‘-test’ > /proc/567/attr/ptags
jo> echo ‘-arbiter’ > /proc/567/attr/ptags
jo> cat /proc/567/attr/ptags
pcooky:0:factorK=56
pcooky:id=0
jo> _
```



```
@arbiter
pcooky:0:factorK=56
pcooky:id=0
@test=simple
```

fork

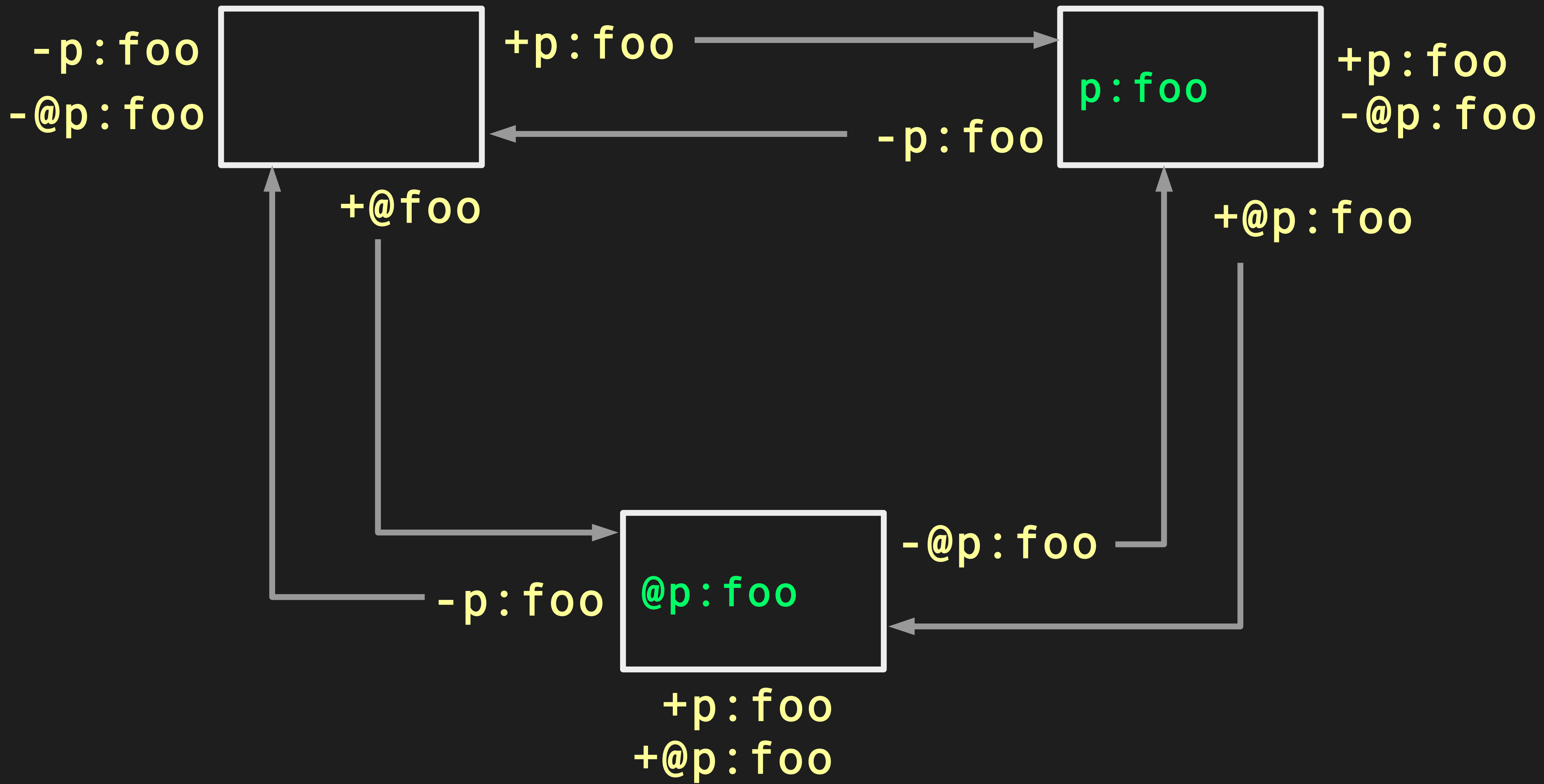
execve

```
@arbiter
pcooky:0:factorK=56
pcooky:id=0
@test=simple
```

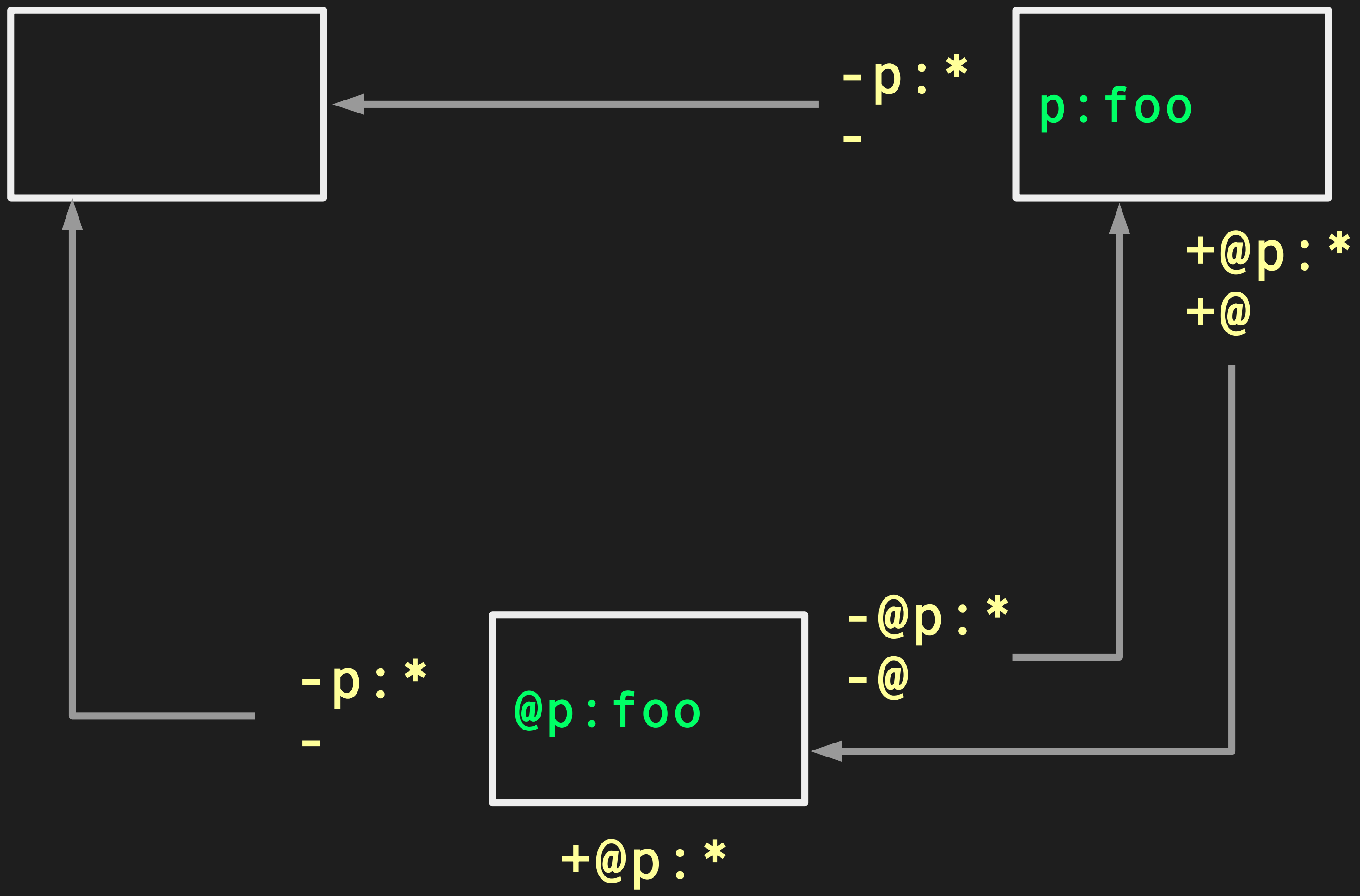
```
@arbiter
@test=simple
```

@ is the keep flag

Managing keep flag

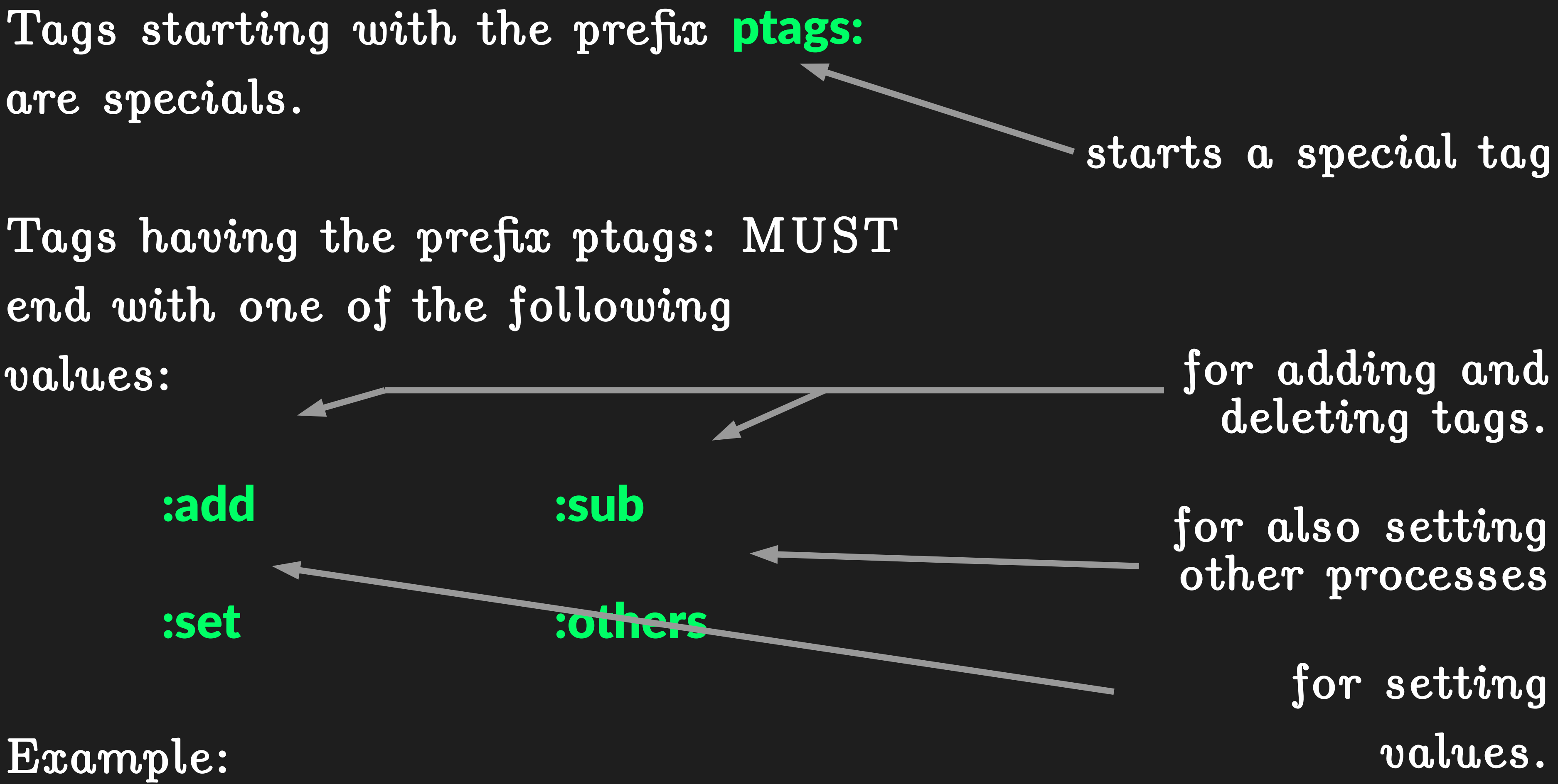


Managing tags globally



SECURITY

- Consulting tags is possible to any process
- To add, remove or set tags and keep flags, a process need rights
  - the capability **CAP_MAC_ADMIN** gives all the rigth and allows a process to perform any action for itself and other processes
  - being a kernel thread behaves as if capability **CAP_MAC_ADMIN** was set
  - special tags gives capabilities to processes that have it; special tags are prefixed with **ptags:**



Tags starting with the prefix **ptags:** are specials.

starts a special tag

Tags having the prefix ptags: MUST end with one of the following values:

for adding and deleting tags.

:add **:sub**

for also setting other processes

:set **:others**

for setting values.

Example:

ptags:sub
ptags:pcooky:add
ptags:pcooky:others

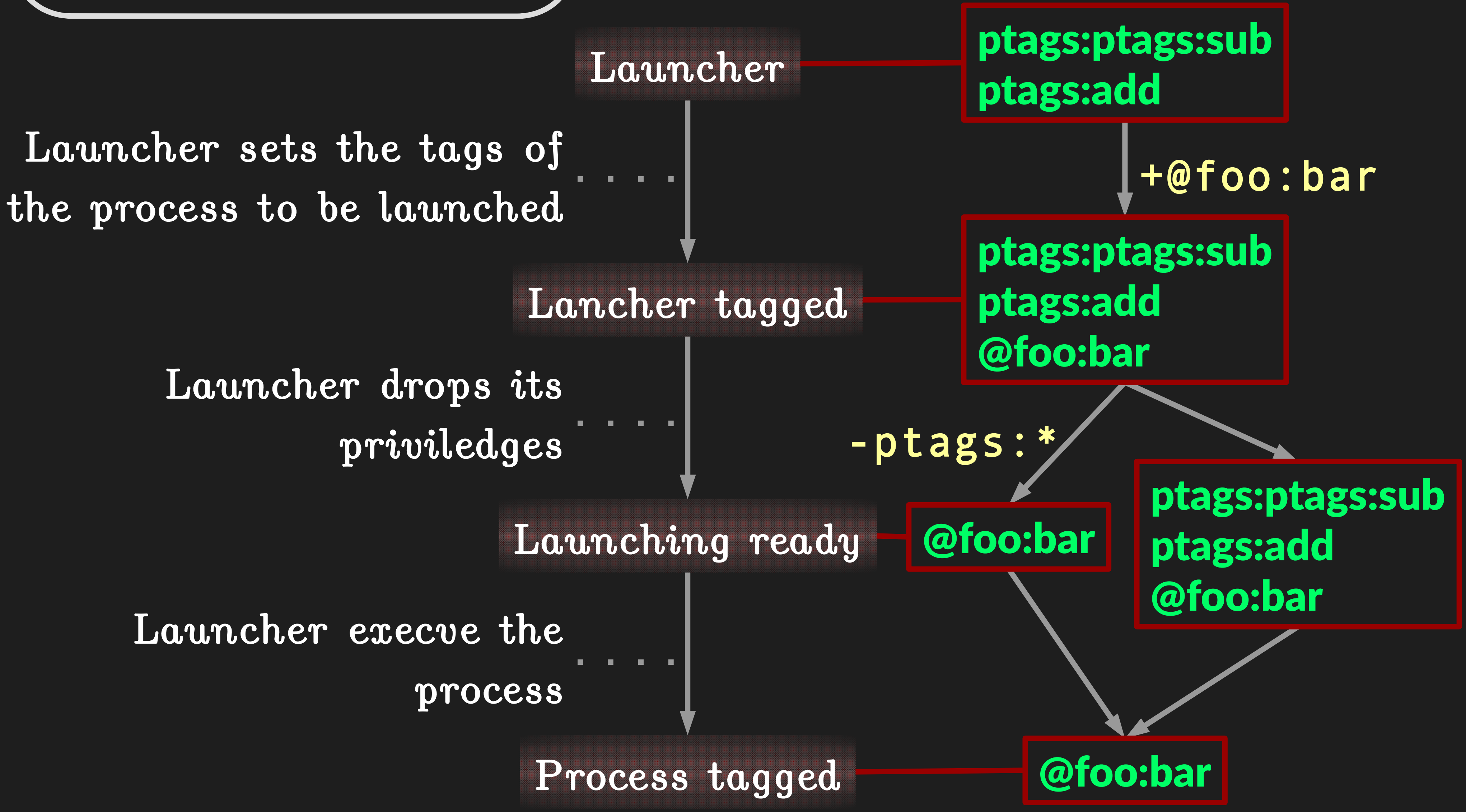
Special tags are of the form: **ptags:ACTION**
ptags:PREFIX:ACTION

Meaning that the action applies only for tags having the given **PREFIX:** or for any tag not prefixed with **ptags:** when no prefix is set.

Examples:

- **ptags:sub**
  - → Allows to remove any tag not prefixed by **ptags:**
- **ptags:pcooky:add**
  - → Allows to add tags prefixed with **pcooky:**
- **ptags:pcooky:others**
  - → Allows to alter tag prefixed with **pcooky:** also for other processes
- **ptags:ptags:add**
  - → Allows to create any tag prefixed by **ptags:**

Launcher use case



Launcher

ptags:ptags:sub
ptags:add

Launcher sets the tags of the process to be launched

+@foo:bar

Lancher tagged

ptags:ptags:sub
ptags:add
@foo:bar

Launcher drops its priviledges

-ptags:*

Launching ready

@foo:bar

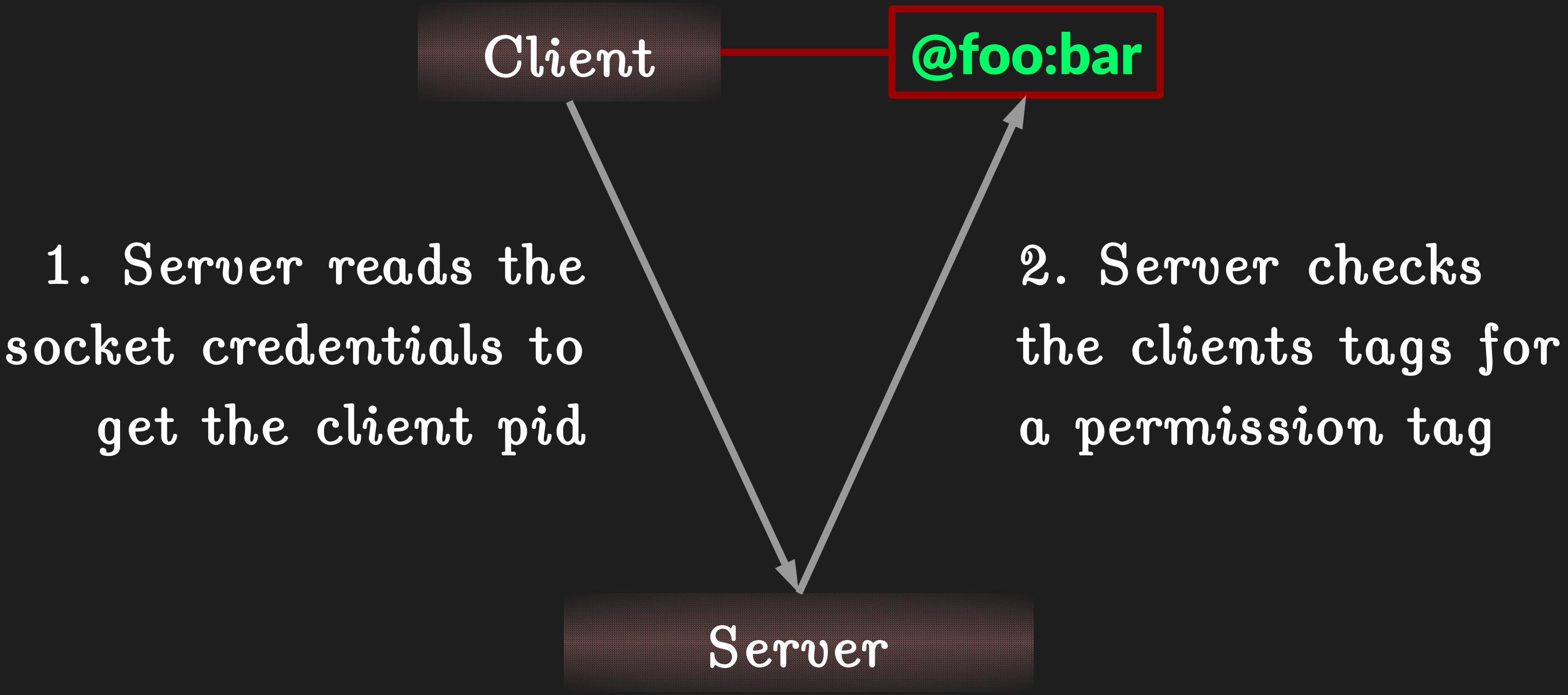
ptags:ptags:sub
ptags:add
@foo:bar

Launcher execve the process

Process tagged

@foo:bar

Permission use case

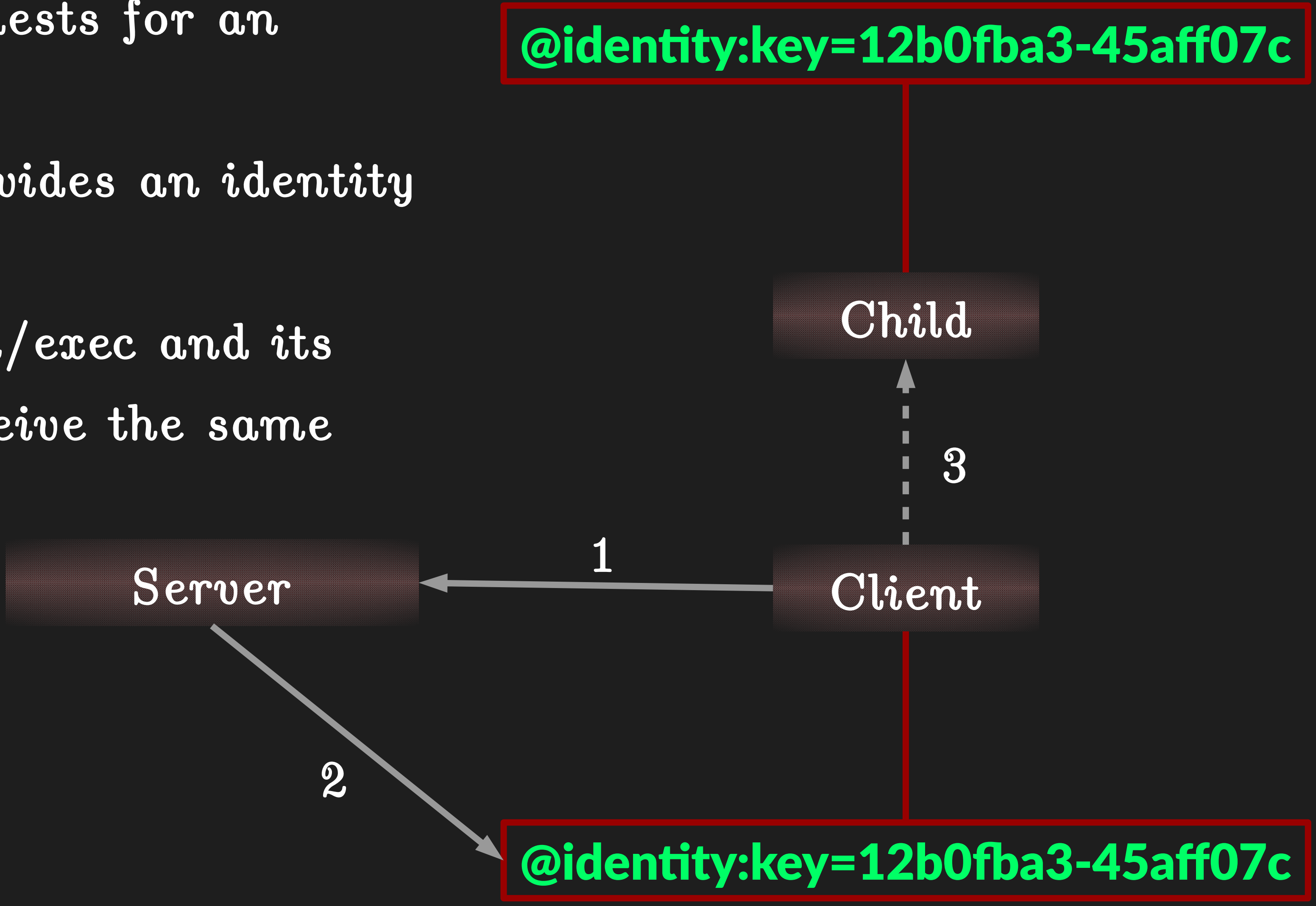


1. Server reads the socket credentials to get the client pid

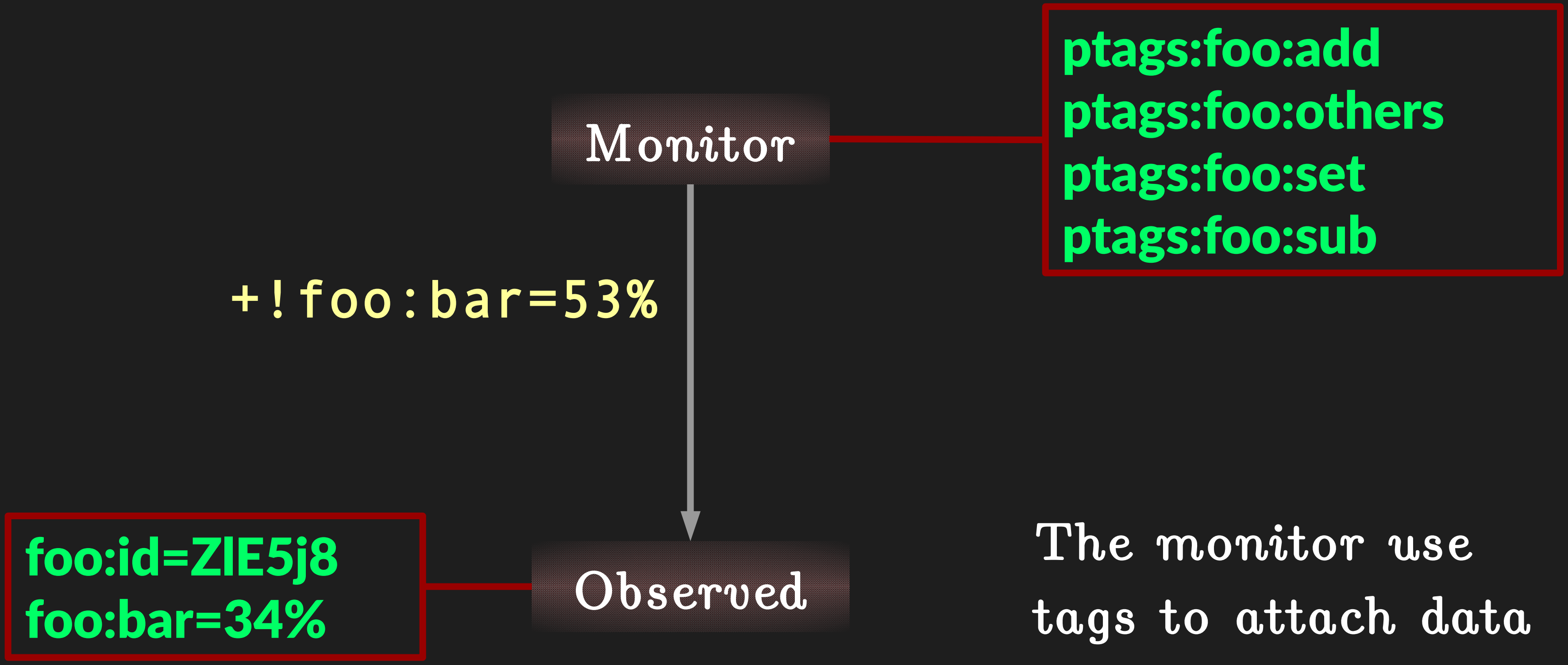
2. Server checks the clients tags for a permission tag

Identity agent use case

1. Client requests for an identity
2. Server provides an identity key
3. Client fork/exec and its children receive the same identity key


@identity:key=12b0fba3-45aff07c
Child
3
1
Server
Client
2
@identity:key=12b0fba3-45aff07c

Monitor use case



Monitor

ptags:foo:add
ptags:foo:others
ptags:foo:set
ptags:foo:sub

`+!foo:bar=53%`

Observed

foo:id=ZIE5j8
foo:bar=34%

The monitor use tags to attach data to the processes it monitors

And many more use cases …

Up to you

your imagination

your need

PTAGS module handles user namespaces (thanks to Serge Hallyn remarks).

So PTAGS is “container” ready.

Tags for a process can be different depending on the namespace.

Cool no?

Some issues

- ➔Since V4.10 writing files of /proc/pid/attr for an other process is forbidden by default
- ➔Files in /proc/pid/attr have size restrictions
- ➔Critic was made because pid can be faked! So isn't reliable
- ➔Threads have different tags than the main task
- ➔Cred structure isn't well suited for ptags

Next version

- ➔Add process unic identifier option in kernel with unrestricted files (work in prrogress)
- ➔Get values will need ptags:get tag, a new ptags's key
- ➔Make shared state for all threads of a process
- ➔Use the renewed security field attached to tasks

PTAGS module is available as patches for kernels 4.4, 4.8 and 4.9. It is available as a git repository at the URL:

`https://gitlab.com/jobol/ptags`

It has a precise documentation of the LSM PTAGS:

`https://gitlab.com/jobol/ptags/raw/master/ptags.txt`

The repository also provides a helper library to query and manage ptags. This helper library implement simple API for asynchronous monitoring of changes using inotfy.

`https://gitlab.com/jobol/ptags/blob/master/lib/ptags.h`

A repository provides PTAGS integration for yocto projects

`https://gitlab.com/jobol/meta-ptags.git`

Question? Mail me to ...

jobol@nonadev.net

José Bollo