IMA namespace best for container integrity?

Department name: Dresden Research Center (DRC)
Author’s name: Enrico Bravi, Roberto Sassu
Date: 19/09/2024

HUAWEI

Contents

Background - Containers

- Containers are a flexible and **lightweight** virtualization technique to provide isolation among processes
  - Flexible: can decide the level of isolation
  - Lightweight: hardware and kernel shared among containers

- Containers are built upon Linux **namespaces**
  - Namespaces provide to processes independent instances of the same kernel resource (e.g. mount table)
  - Each process is associated with a set of namespaces and it has access to the resources of those namespaces
  - A process can migrate to new/existing namespaces through system calls
  - 8 upstreamed namespaces: user, network, IPC, PID, cgroup, mount, time, UTS

- A **container** is a set of different namespaces aggregated by userspace (e.g. LXC, Docker, Podman)

Background – Container Runtime

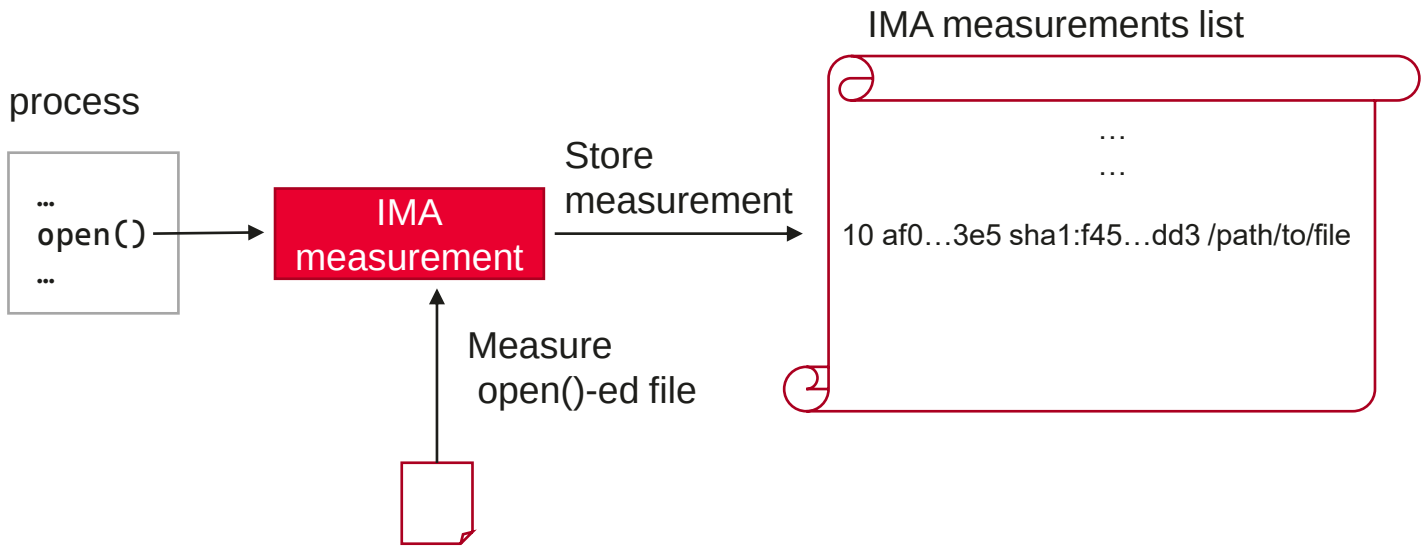
- The kernel is **not aware** of containers
  - The kernel cannot identifies containers (as resources belonging to them)

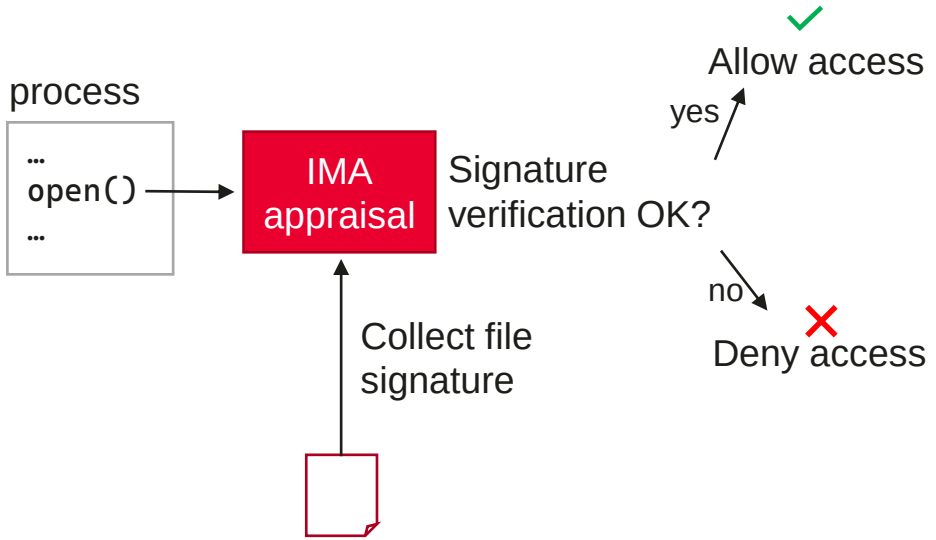
- Containers' isolation is in place only if the container runtime (e.g. Docker) configures everything correctly
  - This cannot be assumed as true with container runtime being a userspace process (outside the TCB)

- The growing adoption of containers has brought interest in allowing integrity verification features per-container
  - Being able to enforce and verify that a container behaves as desired

Background - IMA

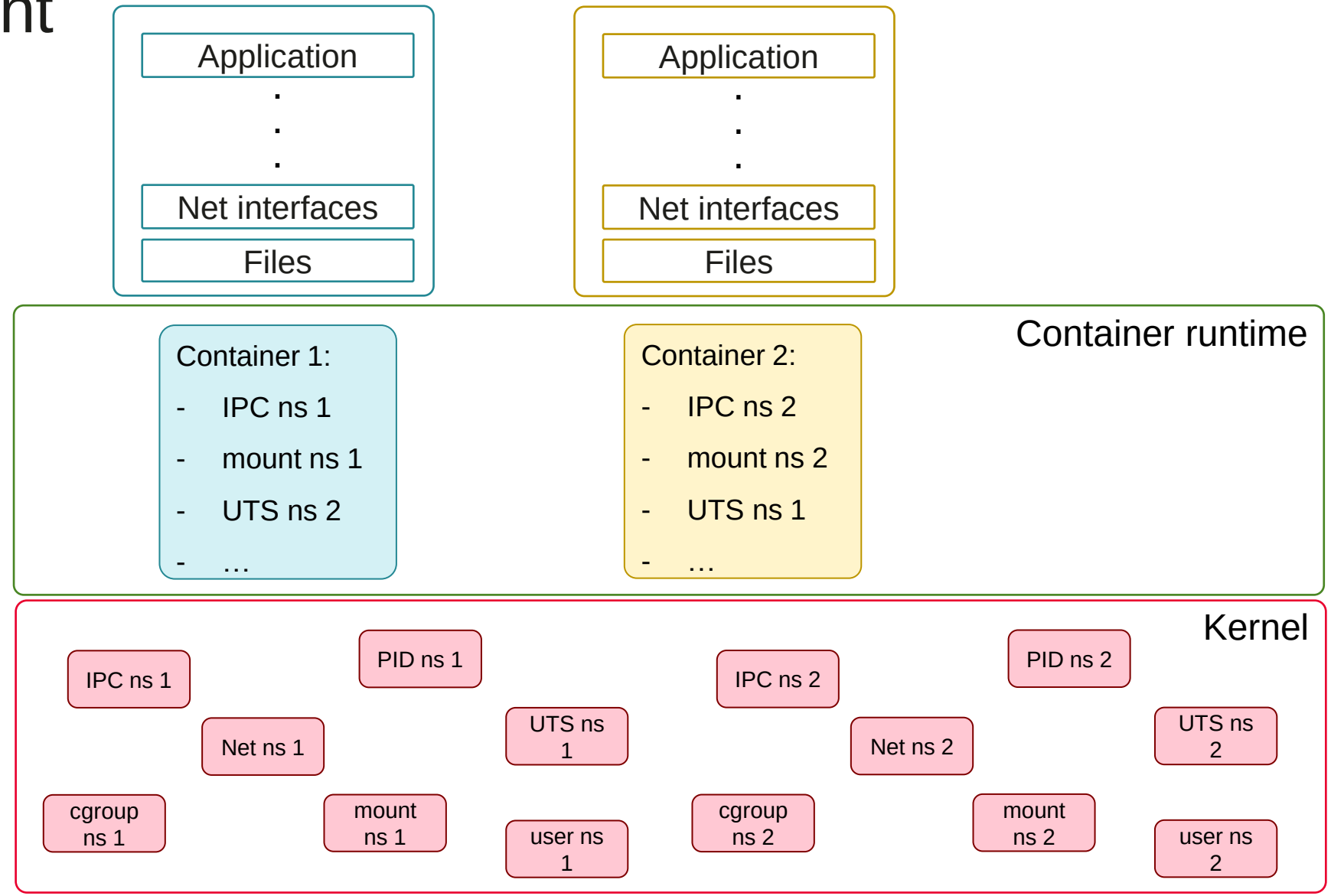
- For integrity verification purposes, Linux provides the Integrity Measurement Architecture (IMA) LSM
  - > IMA is **policy** based and provides three main features:
    - **IMA-measurement**: it measures events for providing evidences of what happened on the host
    - **IMA-appraisal**: it enforces file integrity
    - **IMA-audit**: it augments the system audit log with file hashes





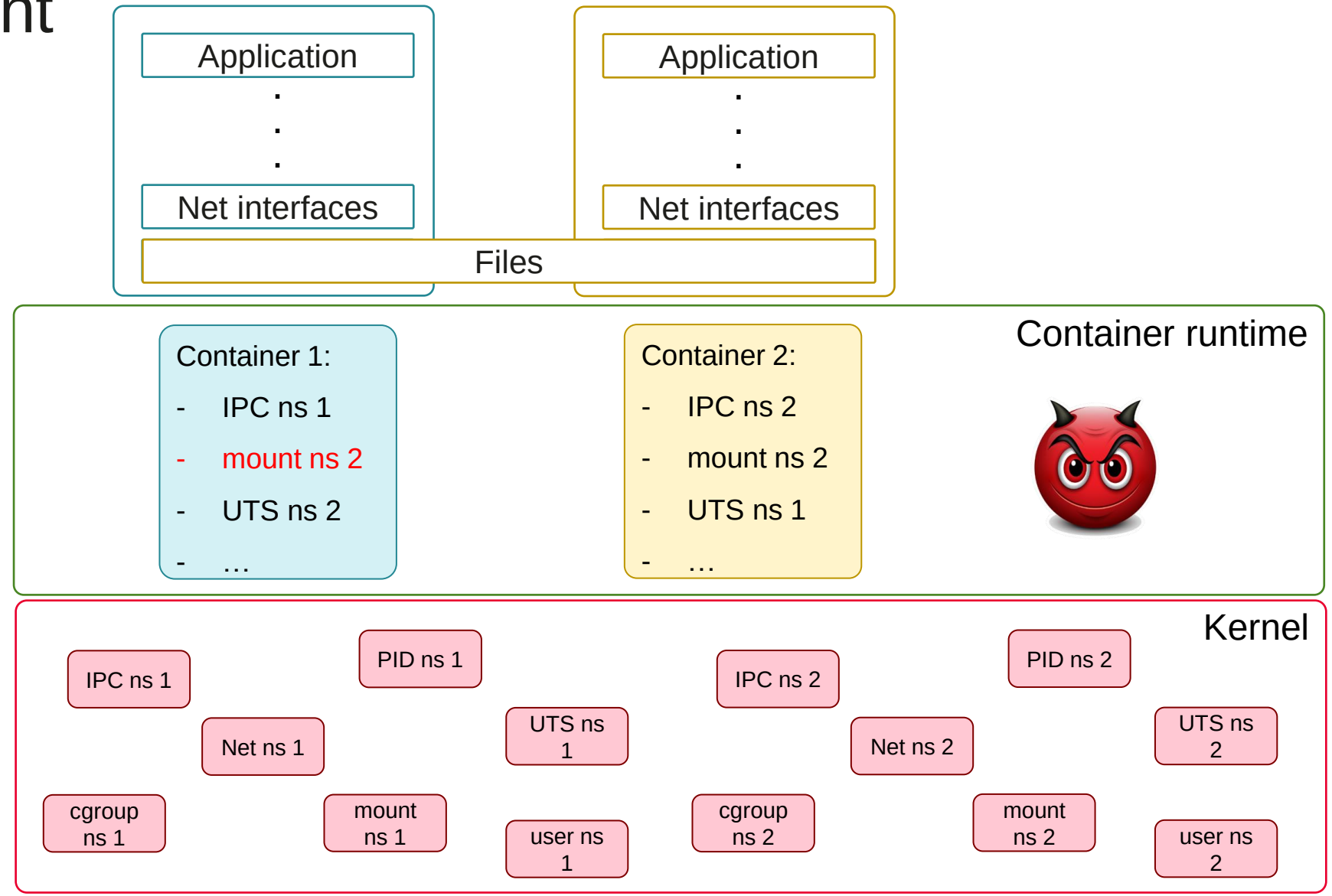
Problem Statement

- Verifying the container integrity as a **whole**
  - > Processes belonging to a container
  - > Files belonging to a container



Problem Statement

- … but containers are configured by userspace
- Kernel cannot detect this misconfiguration
- Kernel would have to verify the container runtime integrity too
- **Hard**: IMA can only give load-time integrity guaranties (runtime attacks cannot be fully detected)



Possible Solution: IMA Namespace

- Not being able to use the container resources association, for integrity verification, we have to define a new one
- This new association must be maintained by the kernel
- This association aggregates processes for which the integrity is verified assuming there is not any other isolation in place
- This association is a new namespace: the IMA namespace
- IMA namespace != userspace container

IMA namespace: a bit of history

- v1: First proposed as a standalone namespace
  - Created only when a mount namespace is created
- v2: Piggy backed into the mount namespace
  - NACK-ed
- v3: Piggy backed into the user namespace
  - Created along CLONE_NEWUSER
- v4: Standalone namespace
  - Created on clone() after write on securityfs
- V5: moved back into user namespace
  - Created after securityfs mount

Contribution

- Define integrity principles IMA must follow when dealing with containers
- Evaluate proposed solutions
- Propose improvements

Principles defined

- **Per-IMA namespace measurements list**

- IMA namespace as a processes' (sub)set
  - File don't belong to IMA namespaces and can be shared among namespaces

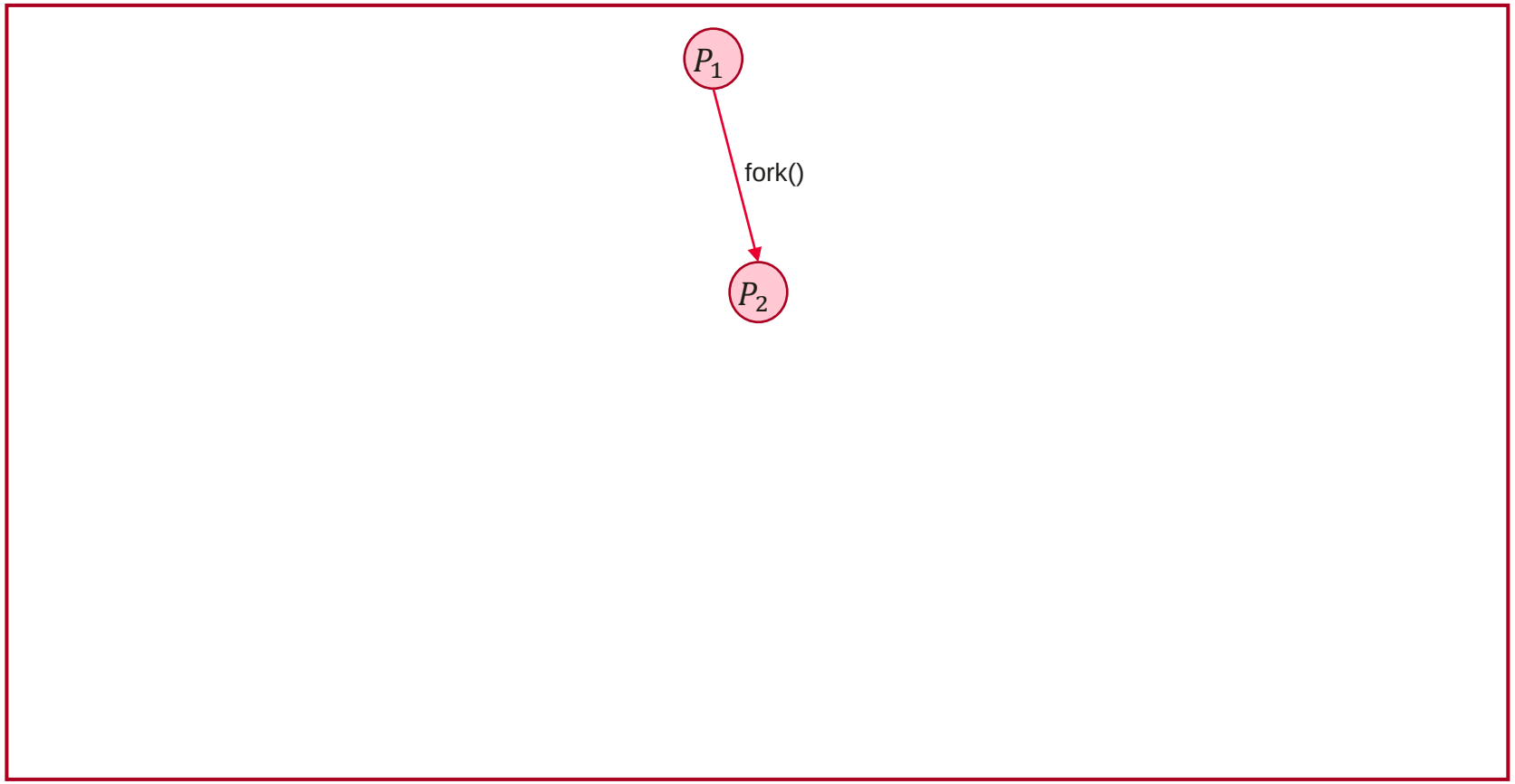
- An IMA namespace is not aware of children IMA namespaces
  - A process cannot escape its current IMA namespace

- The kernel is common for all IMA namespaces
  - For example kernel modules loading affects all IMA namespaces

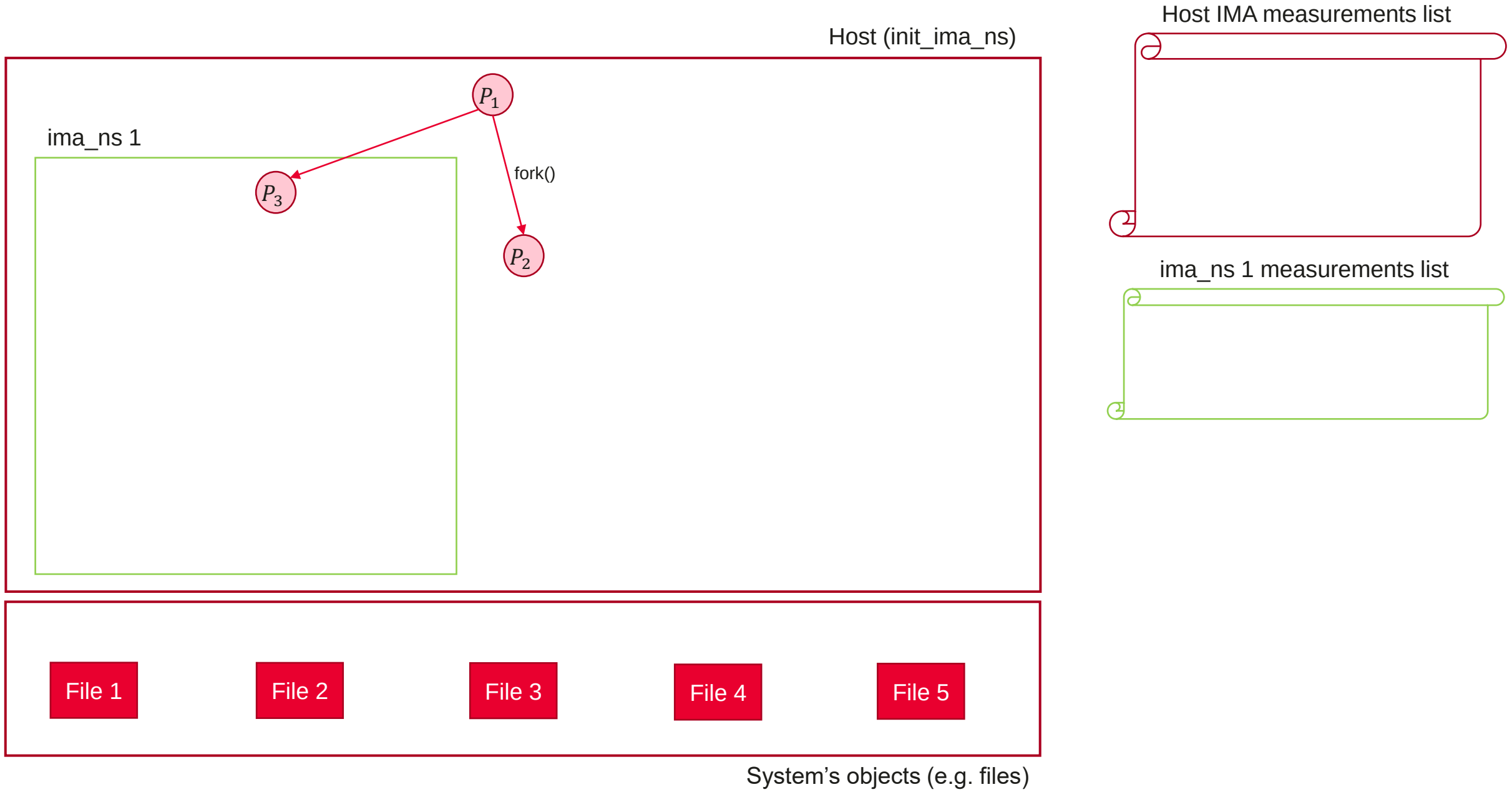
- An IMA namespace cannot miss integrity events related to its associated processes
  - No gap between IMA namespace creation and activation (events in-between would be missed)
  - Join performed during execve() just before the main executable is loaded in the process memory

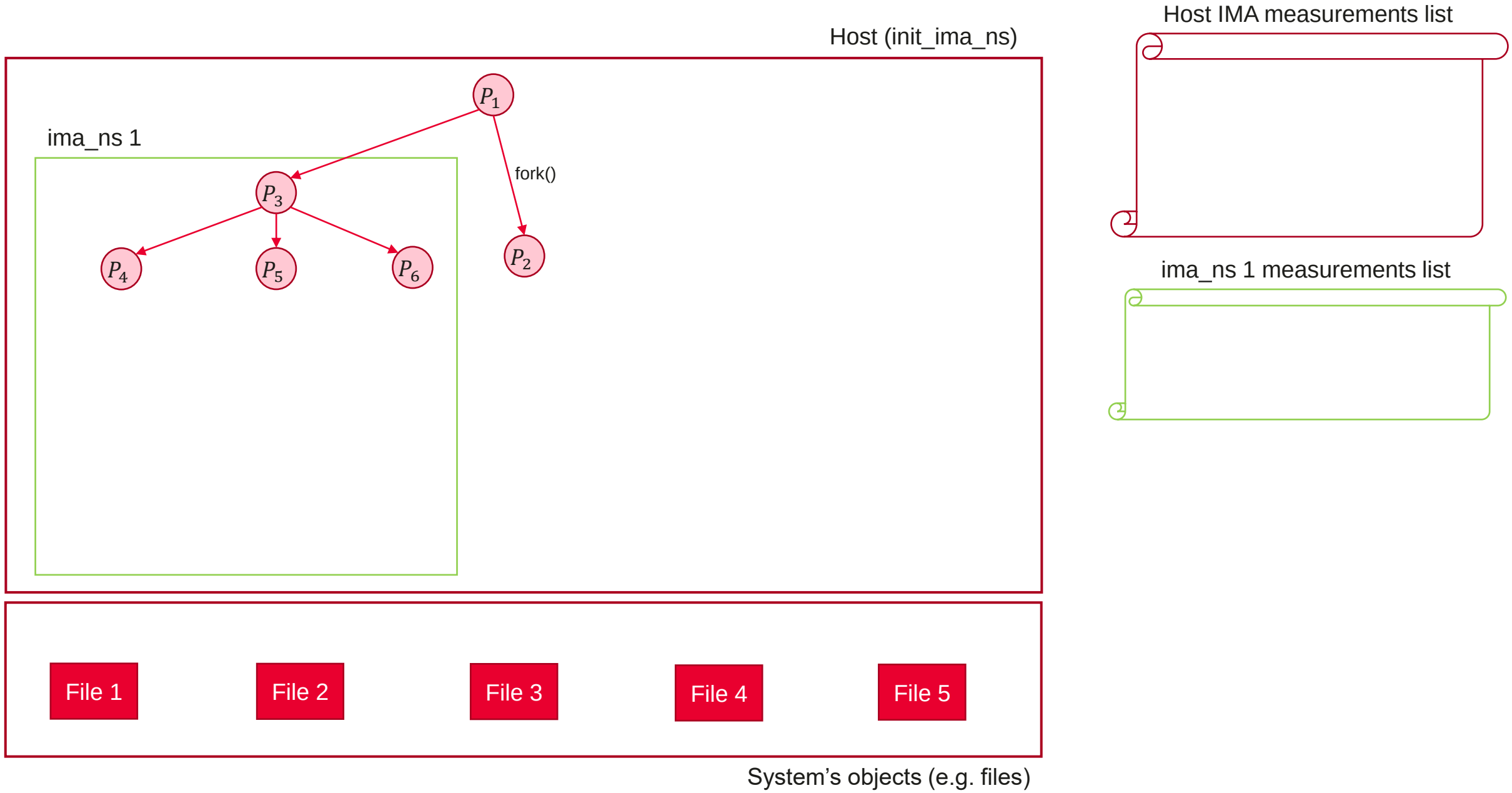
Host IMA measurements list

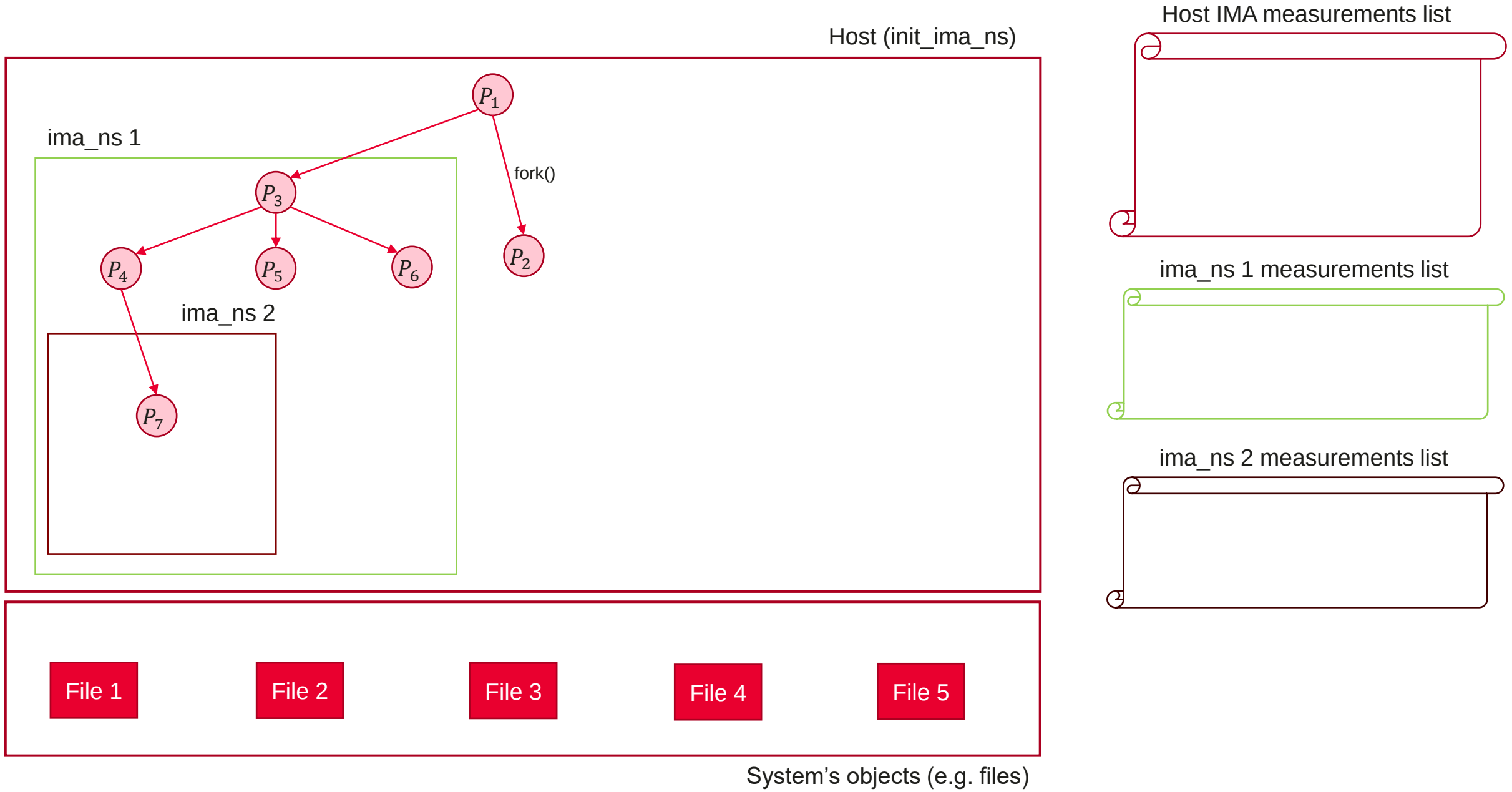
Host (init_ima_ns)

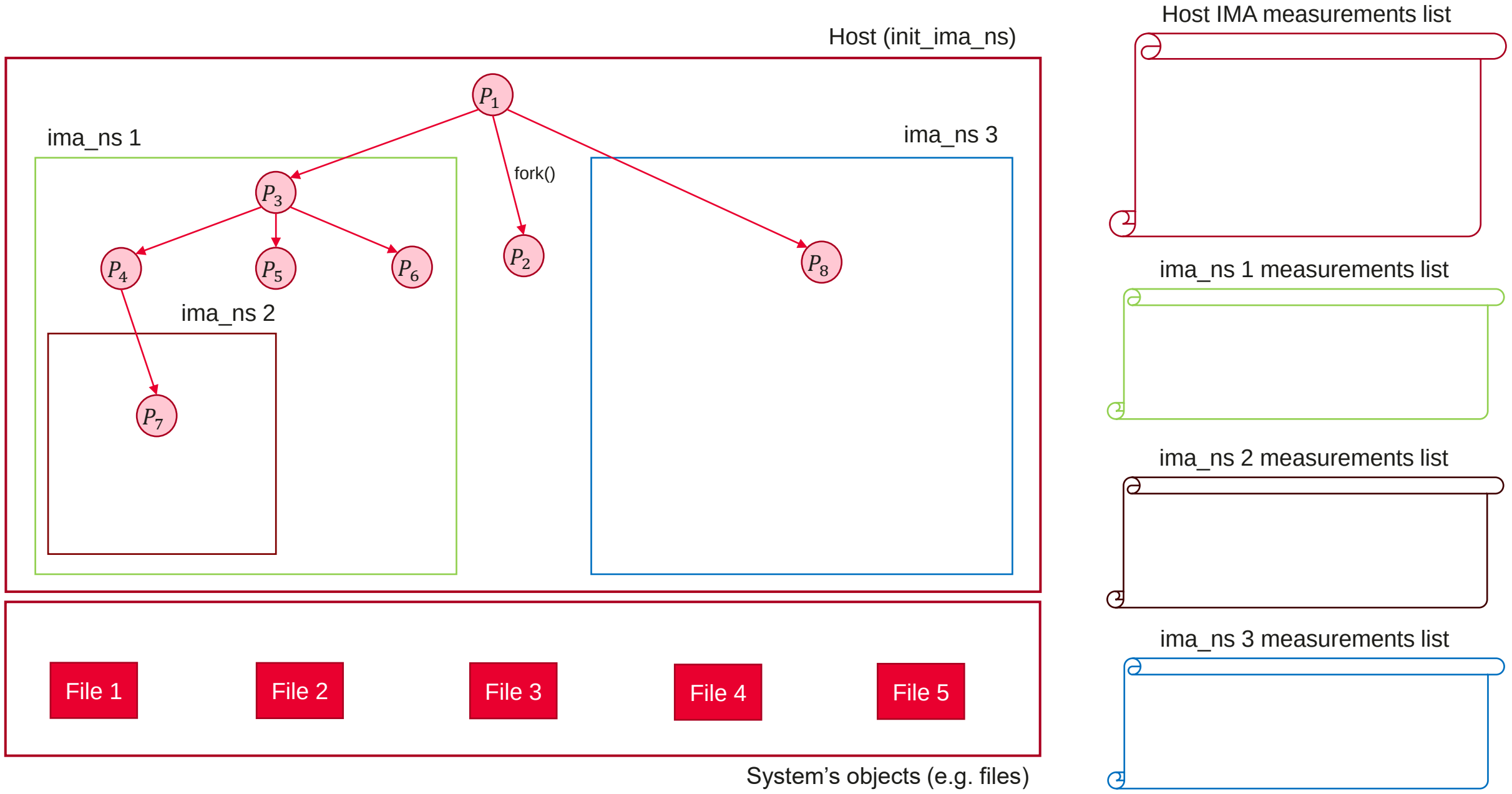


System's objects (e.g. files)


Host (init_ima_ns)
ima_ns 1
P_1
P_3
P_2
fork()
Host IMA measurements list
ima_ns 1 measurements list
File 1
File 2
File 3
File 4
File 5
System's objects (e.g. files)


Host (init_ima_ns)
ima_ns 1
P_1
P_2
P_3
P_4
P_5
P_6
fork()
Host IMA measurements list
ima_ns 1 measurements list
File 1
File 2
File 3
File 4
File 5
System's objects (e.g. files)


Host (init_ima_ns)
ima_ns 1
P_1
fork()
P_3
P_4
P_5
P_6
P_2
ima_ns 2
P_7
File 1
File 2
File 3
File 4
File 5
System's objects (e.g. files)
Host IMA measurements list
ima_ns 1 measurements list
ima_ns 2 measurements list


Host (init_ima_ns)
ima_ns 1
ima_ns 3
ima_ns 2
fork()
P_1
P_2
P_3
P_4
P_5
P_6
P_7
P_8
File 1
File 2
File 3
File 4
File 5
System's objects (e.g. files)
Host IMA measurements list
ima_ns 1 measurements list
ima_ns 2 measurements list
ima_ns 3 measurements list

Principles defined

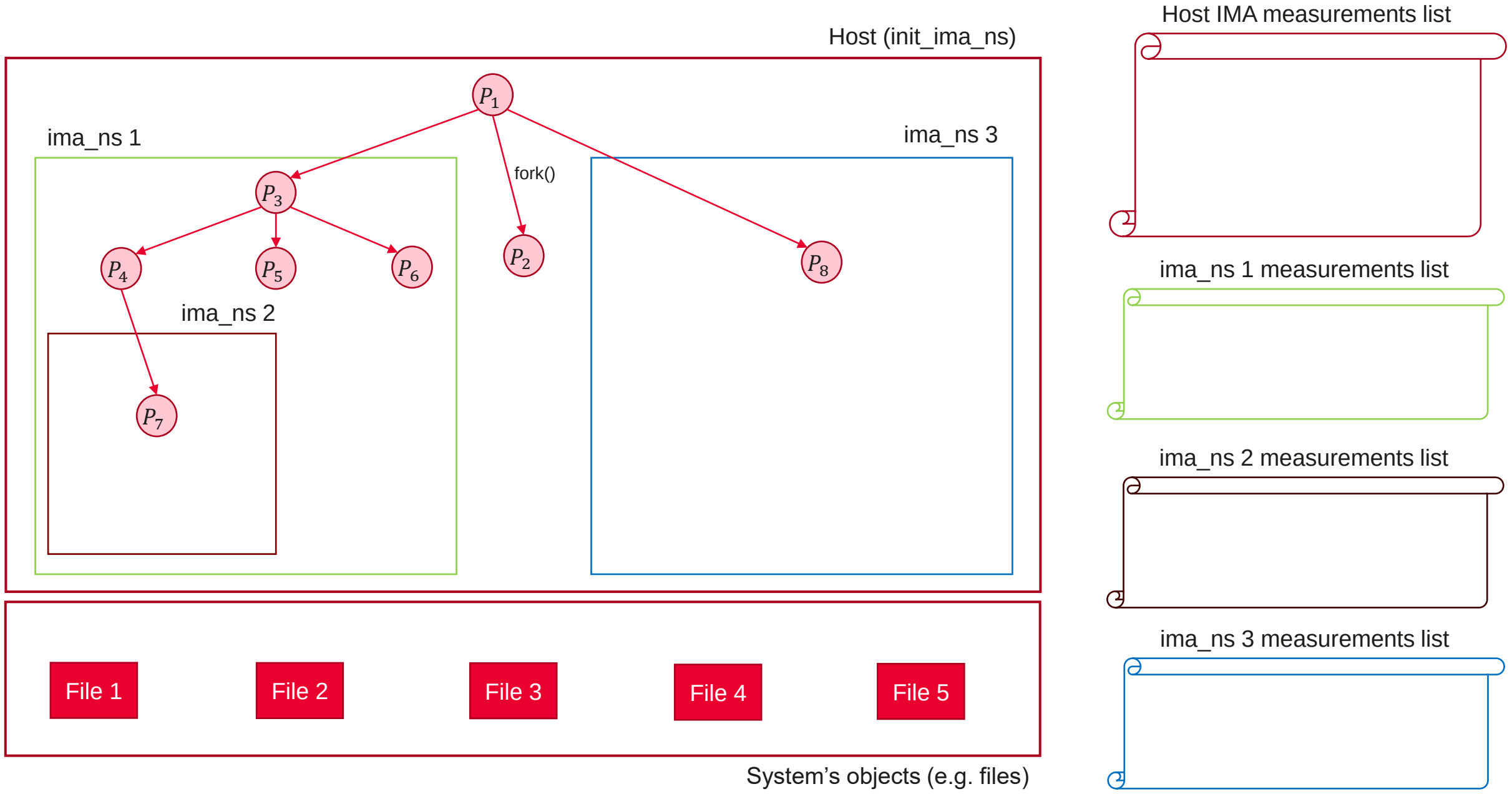
- Per-IMA namespace measurements list

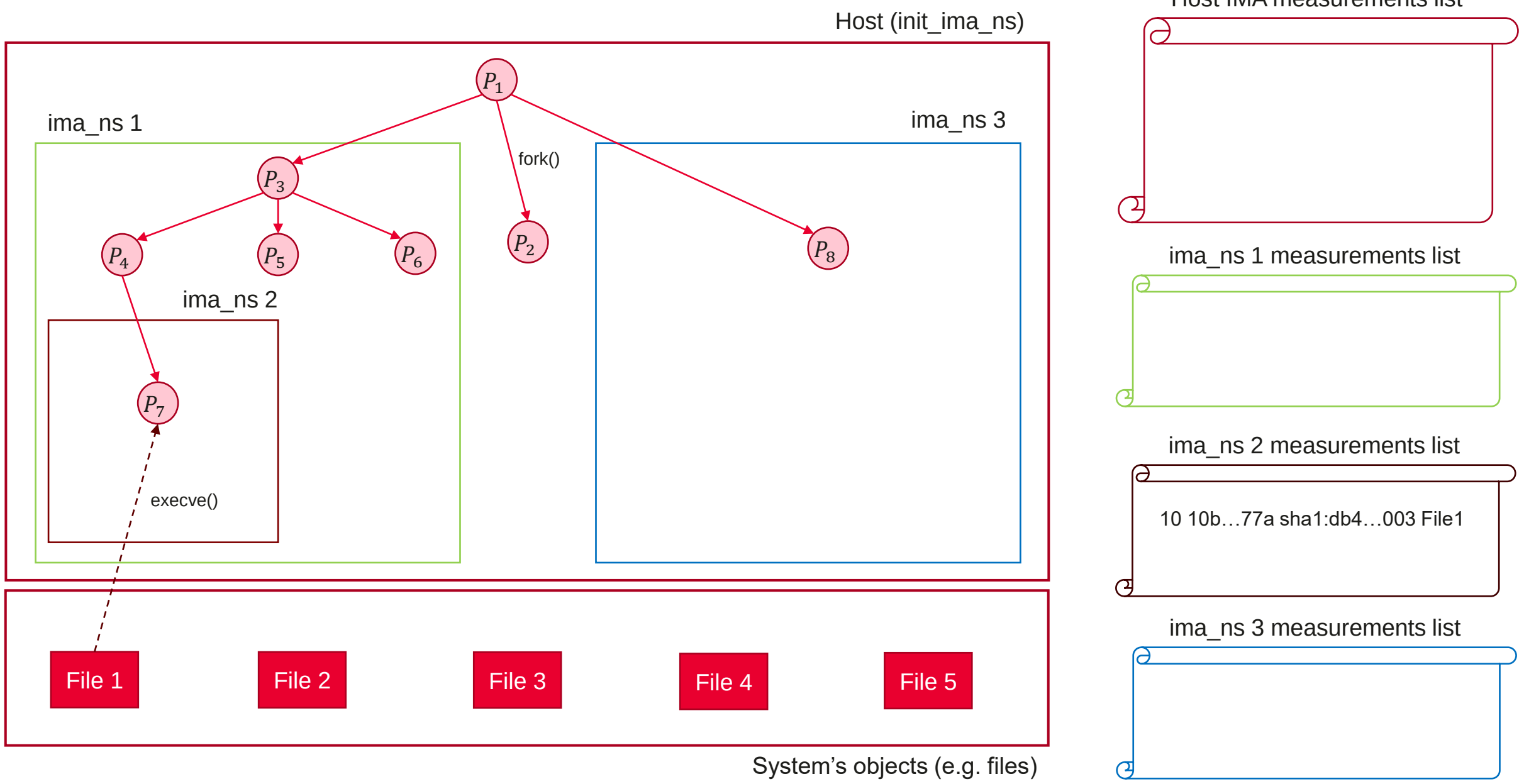
- **IMA namespace as a processes' (sub)set**
  - > **File don't belong to IMA namespaces and can be shared among namespaces**

- An IMA namespace is not aware of children IMA namespaces
  - > A process cannot escape its current IMA namespace

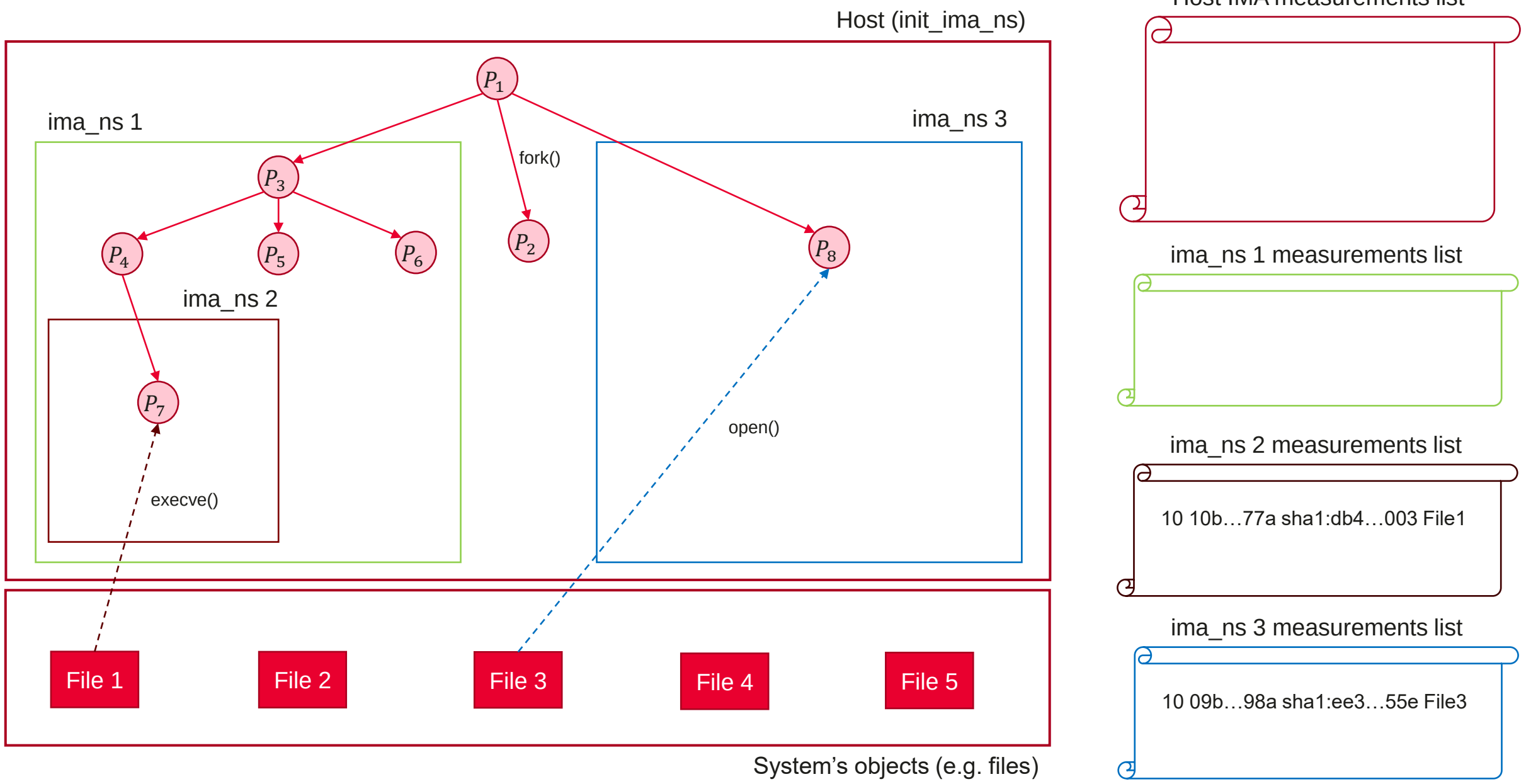
- The kernel is common for all IMA namespaces
  - > For example kernel modules loading affects all IMA namespaces

- An IMA namespace cannot miss integrity events related to its associated processes
  - > No gap between IMA namespace creation and activation (events in-between would be missed)
  - > Join performed during execve() just before the main executable is loaded in the process memory


Host (init_ima_ns)
ima_ns 1
ima_ns 3
ima_ns 2
fork()
P_1
P_2
P_3
P_4
P_5
P_6
P_7
P_8
File 1
File 2
File 3
File 4
File 5
System's objects (e.g. files)
Host IMA measurements list
ima_ns 1 measurements list
ima_ns 2 measurements list
ima_ns 3 measurements list


Host (init_ima_ns)
ima_ns 1
ima_ns 3
ima_ns 2
P_1
P_2
P_3
P_4
P_5
P_6
P_7
P_8
fork()
execve()
File 1
File 2
File 3
File 4
File 5
System's objects (e.g. files)
Host IMA measurements list
ima_ns 1 measurements list
ima_ns 2 measurements list
10 10b…77a sha1:db4…003 File1
ima_ns 3 measurements list

HUAWEI


Host (init_ima_ns)
ima_ns 1
ima_ns 3
ima_ns 2
P_1
P_2
P_3
P_4
P_5
P_6
P_7
P_8
fork()
open()
execve()
File 1
File 2
File 3
File 4
File 5
System's objects (e.g. files)
Host IMA measurements list
ima_ns 1 measurements list
ima_ns 2 measurements list
10 10b…77a sha1:db4…003 File1
ima_ns 3 measurements list
10 09b…98a sha1:ee3…55e File3

HUAWEI

Principles defined

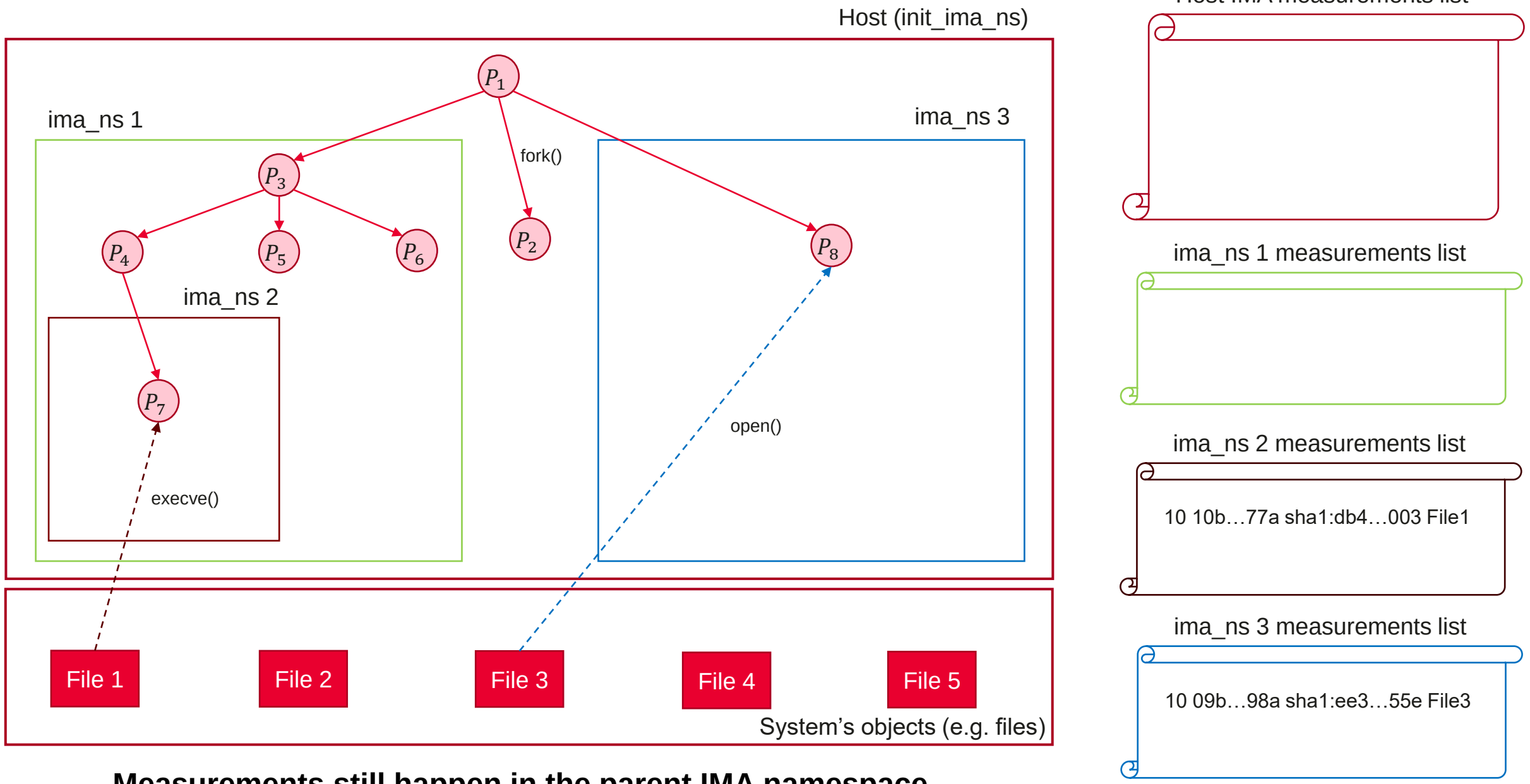
- Per-IMA namespace measurements list

- IMA namespace as a processes' (sub)set
  - File don't belong to IMA namespaces and can be shared among namespaces

- **An IMA namespace is not aware of children IMA namespaces**
  - **A process cannot escape its current IMA namespace**

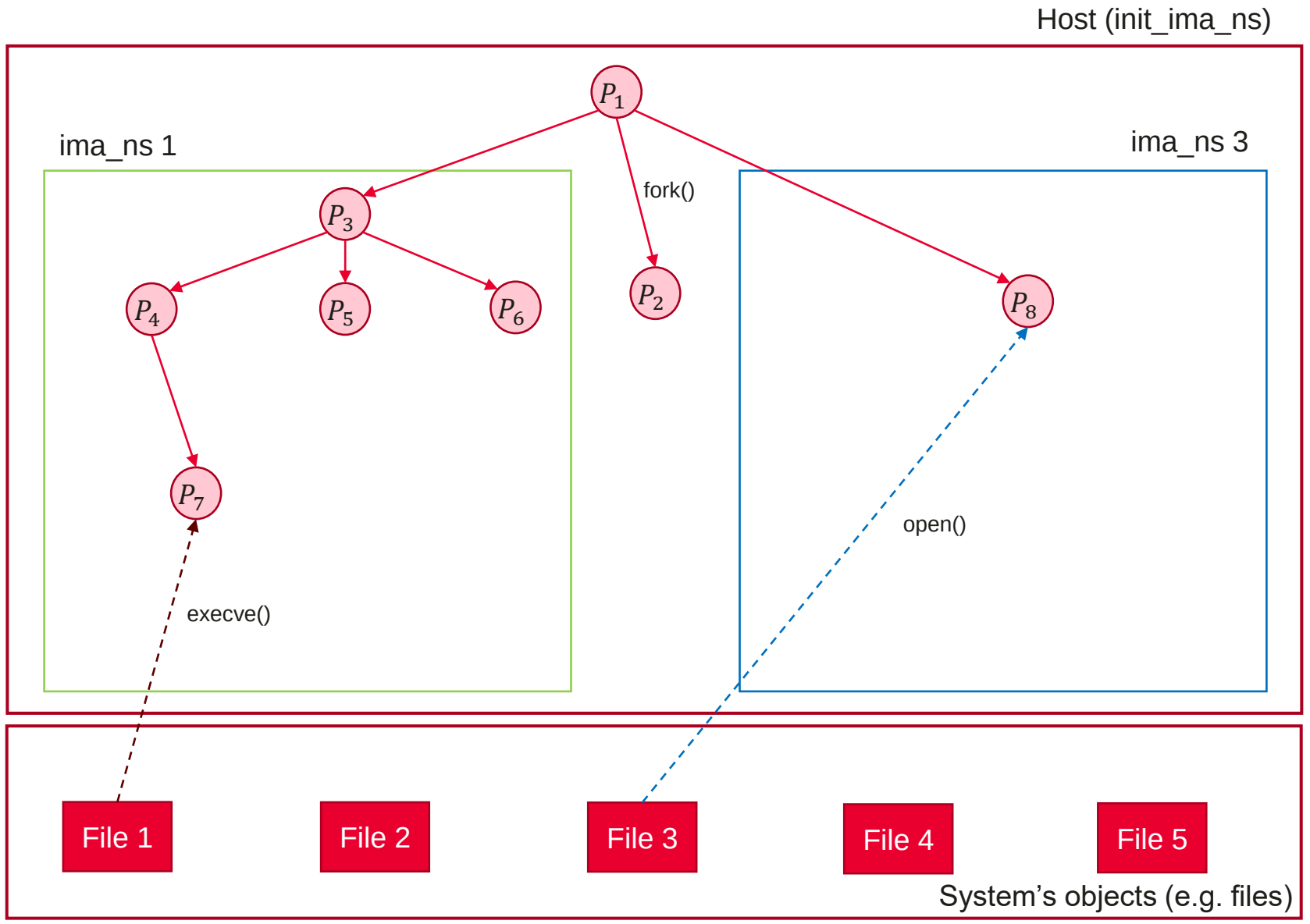
- The kernel is common for all IMA namespaces
  - For example kernel modules loading affects all IMA namespaces

- An IMA namespace cannot miss integrity events related to its associated processes
  - No gap between IMA namespace creation and activation (events in-between would be missed)
  - Join performed during execve() just before the main executable is loaded in the process memory



Measurements still happen in the parent IMA namespace

HUAWEI



Host IMA measurements list

ima_ns 1 measurements list

10 10b…77a sha1:db4…003 File1

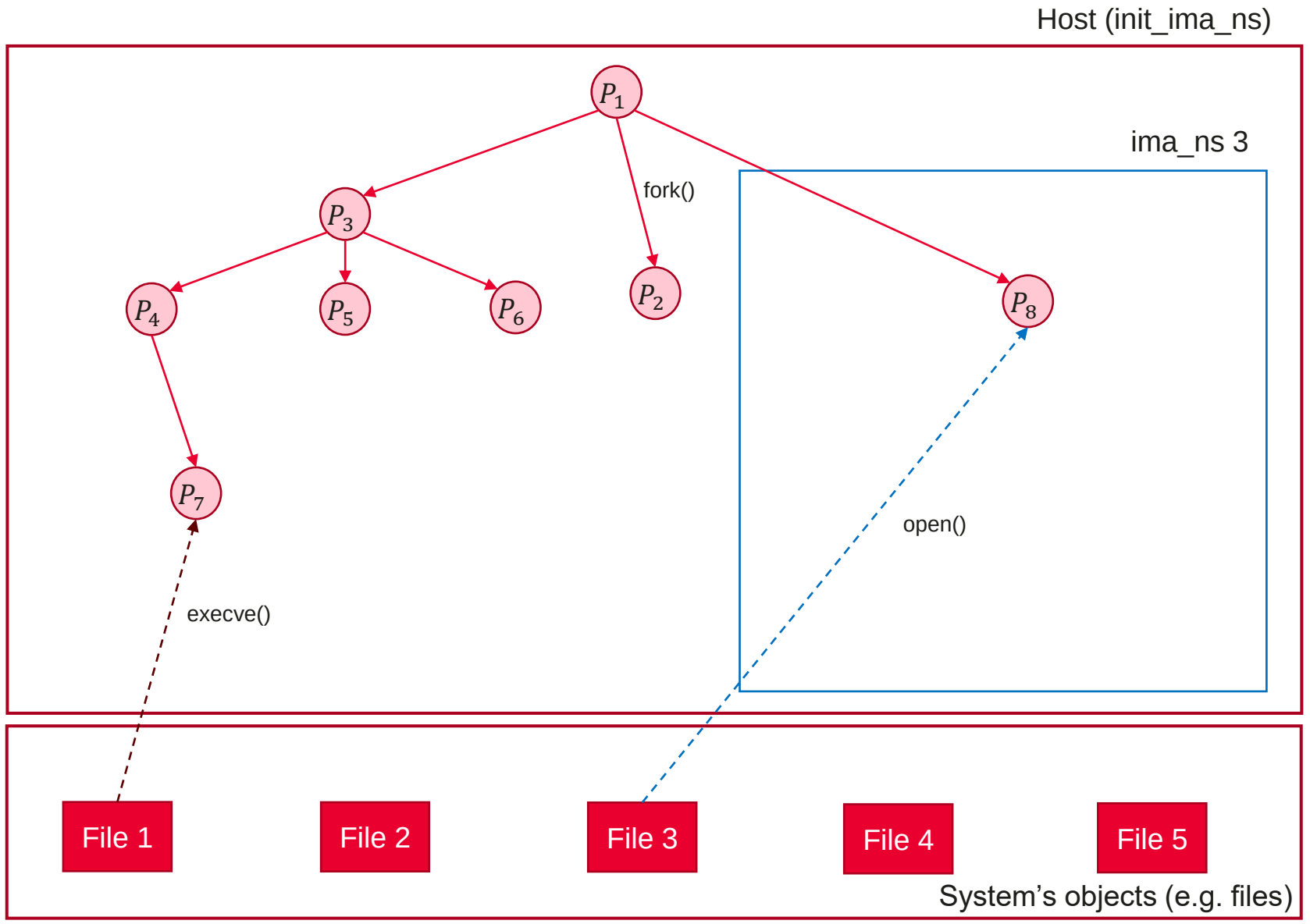
ima_ns 2 measurements list

10 10b…77a sha1:db4…003 File1

ima_ns 3 measurements list

10 09b…98a sha1:ee3…55e File3

Measurements still happens in the parent IMA namespace



Host IMA measurements list

10 10b…77a sha1:db4…003 File1

ima_ns 1 measurements list

10 10b…77a sha1:db4…003 File1

ima_ns 2 measurements list

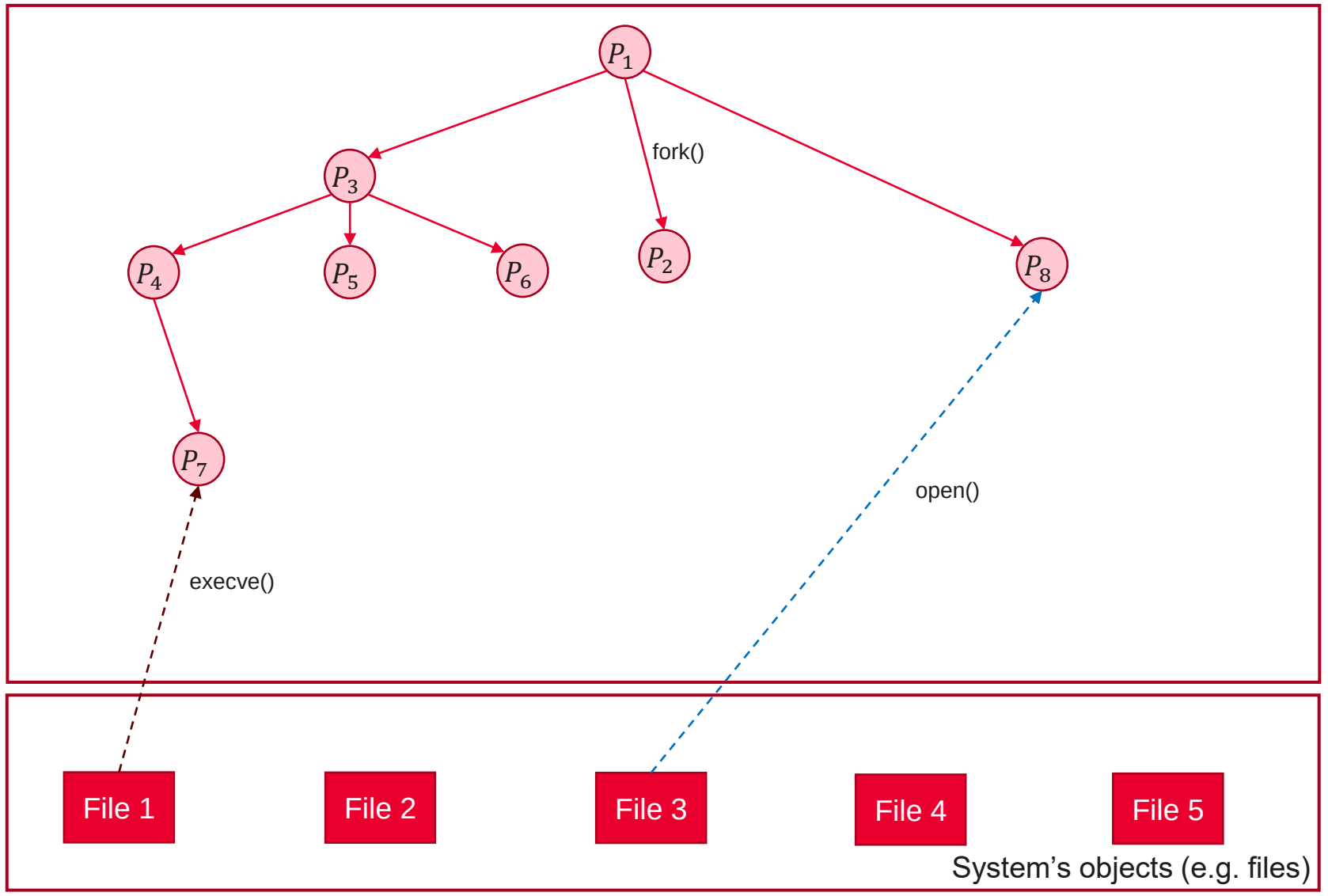
10 10b…77a sha1:db4…003 File1

ima_ns 3 measurements list

10 09b…98a sha1:ee3…55e File3

Measurements still happens in the parent IMA namespace

HUAWEI



Host IMA measurements list

10 10b…77a sha1:db4…003 File1

10 09b…98a sha1:ee3…55e File3

ima_ns 1 measurements list

10 10b…77a sha1:db4…003 File1

ima_ns 2 measurements list

10 10b…77a sha1:db4…003 File1

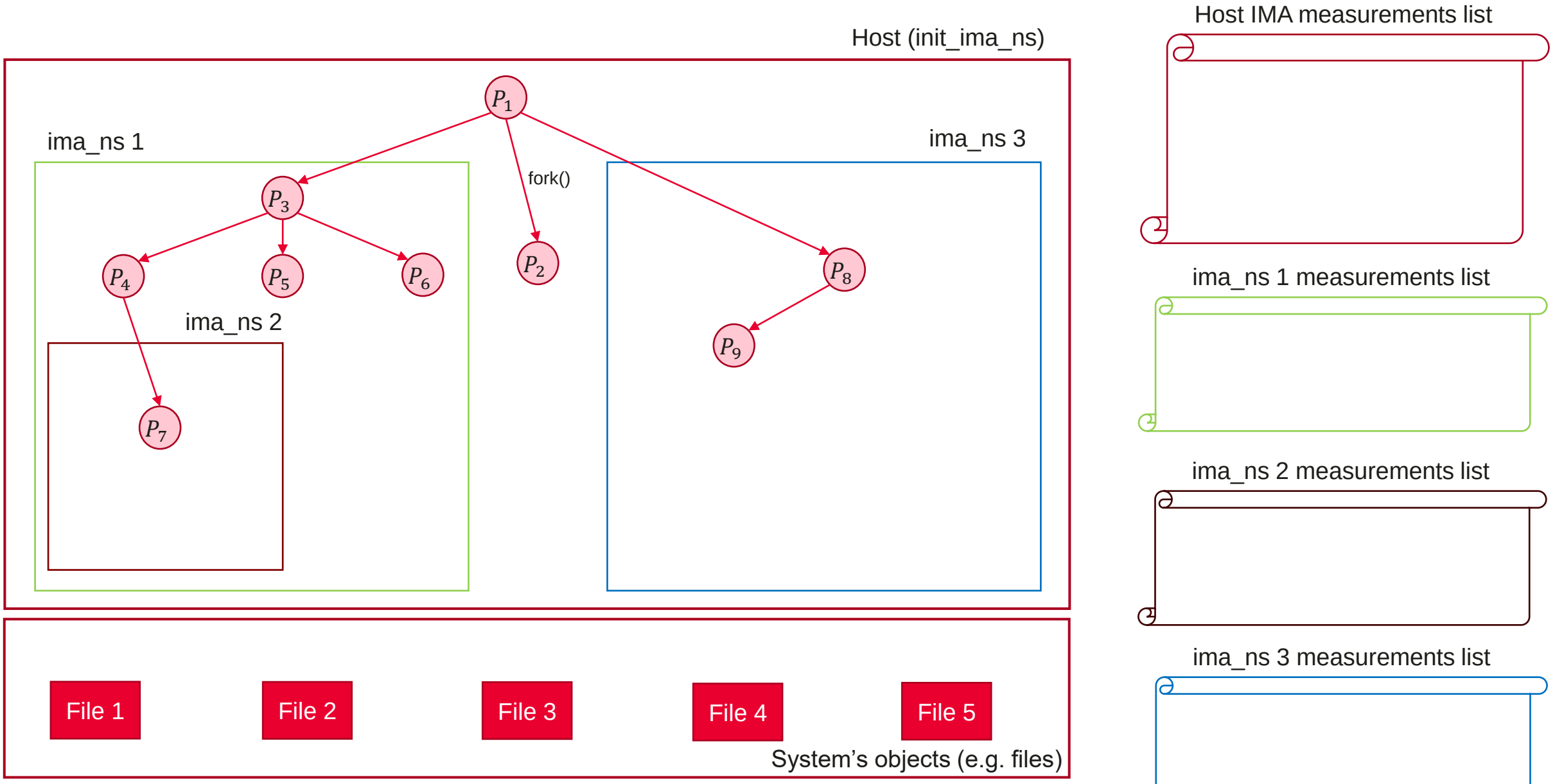
ima_ns 3 measurements list

10 09b…98a sha1:ee3…55e File3

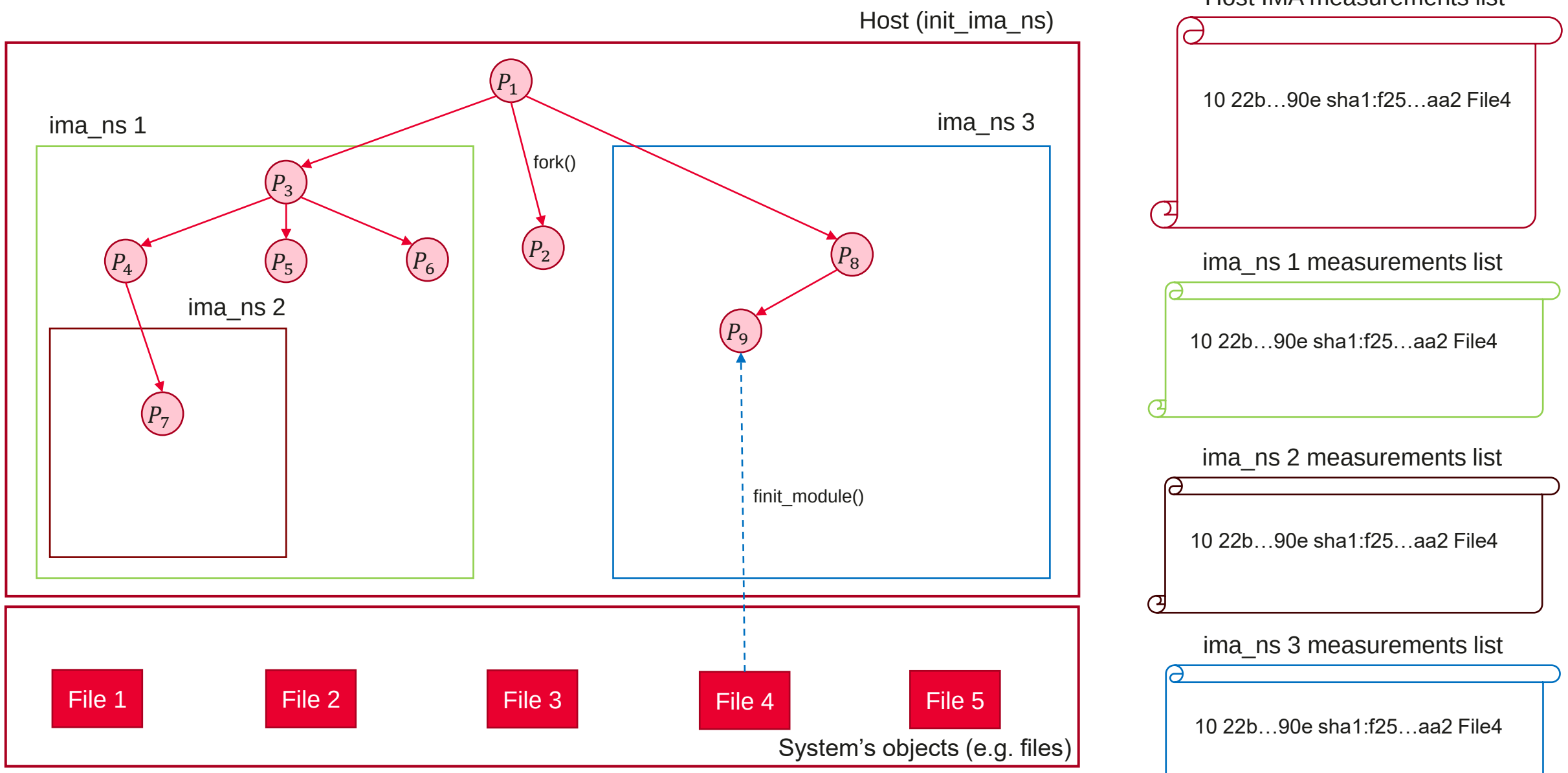
Measurements still happens in the parent IMA namespace

Principles defined

- Per-IMA namespace measurements list
- IMA namespace as a processes' (sub)set
  - File don't belong to IMA namespaces and can be shared among namespaces
- An IMA namespace is not aware of children IMA namespaces
  - A process cannot escape its current IMA namespace
- **The kernel is common for all IMA namespaces**
  - **For example kernel modules loading affects all IMA namespaces**
- An IMA namespace cannot miss integrity events related to its associated processes
  - No gap between IMA namespace creation and activation (events in-between would be missed)
  - Join performed during execve() just before the main executable is loaded in the process memory



Kernel-related events are evaluated by all IMA namespaces



Host IMA measurements list

10 22b…90e sha1:f25…aa2 File4

ima_ns 1 measurements list

10 22b…90e sha1:f25…aa2 File4

ima_ns 2 measurements list

10 22b…90e sha1:f25…aa2 File4

ima_ns 3 measurements list

10 22b…90e sha1:f25…aa2 File4

Kernel-related events are evaluated by all IMA namespaces

HUAWEI

Principles defined

- Per-IMA namespace measurements list
- IMA namespace as a processes' (sub)set
  - File don't belong to IMA namespaces and can be shared among namespaces
- An IMA namespace is not aware of children IMA namespaces
  - A process cannot escape its current IMA namespace
- The kernel is common for all IMA namespaces
  - For example kernel modules loading affects all IMA namespaces
- **An IMA namespace cannot miss integrity events related to its associated processes**
  - **No gap between IMA namespace creation and activation (events in-between would be missed)**
  - **Join performed during execve() just before the main executable is loaded in the process memory**

No gap between IMA namespace creation and activation

- clone3() and unshare() don’t allow atomic creation and configuration
  - Namespace’s configuration parameters cannot be passed to those system calls
  - Stefan’s proposal is to join an (inactive) IMA namespace, configure and activate it later (not atomically)
  - Issue: the new IMA namespace misses process’ events until it is activated

- Don’t allow creation with direct call to clone3() or unshare()
  - clone3() already disabled (CLONE_NEWIMA overlaps with CSIGNAL)
  - unshare() has to be explicitly denied

- Introduce a new atomic procedure to create and configure the IMA namespace
  - Introduce a new “unshare” file in the securityfs which receives (through a write operation) the configuration and triggers the new IMA namespace creation
  - Forbid joining it until creation and configuration are complete

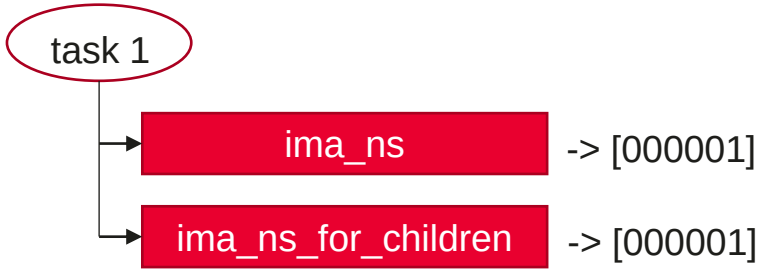
Join performed during execve()

- Stefan’s proposal allows a process, with executable code already loaded, to join a new or existing IMA namespace
  - Issue: target IMA namespace will contain a process with unknown integrity status (loaded code not measured)

- Only chance, to start from a clean integrity state from the IMA perspective, is during execve()
  - It is the only time when joining a new IMA namespace is allowed
  - This guarantees that the new IMA namespace can evaluate the loading of the main executable

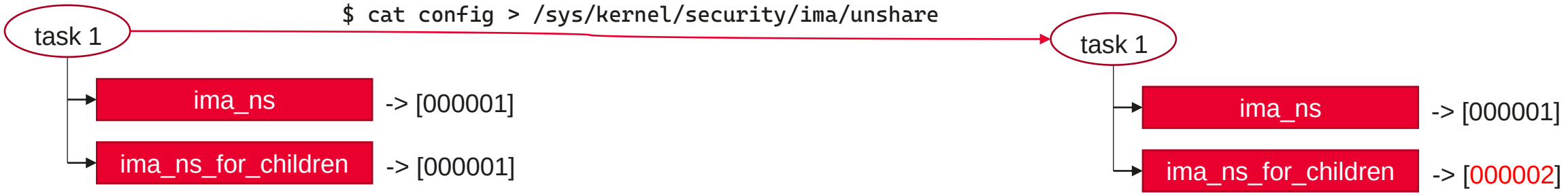
- Since a process cannot join until execve(), the new IMA namespace is temporary referenced by a new nsproxy field called ima_ns_for_children
  - Deferred joining approach is also adopted by PID and time namespace
  - Only time namespace is joined during execve() (later than IMA BPRM_CHECK hook)

Join performed during execve() (2)



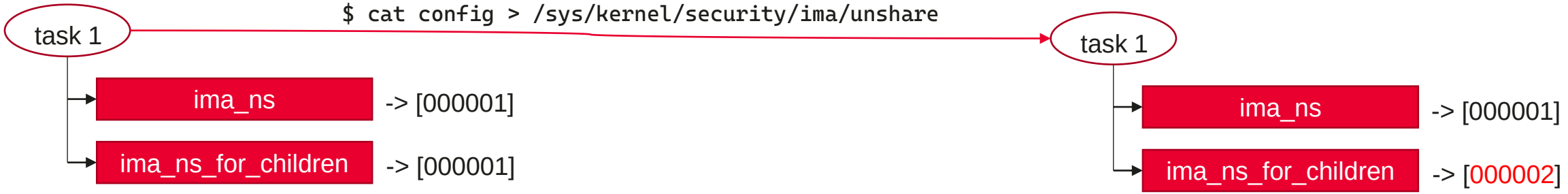
ima_ns_1

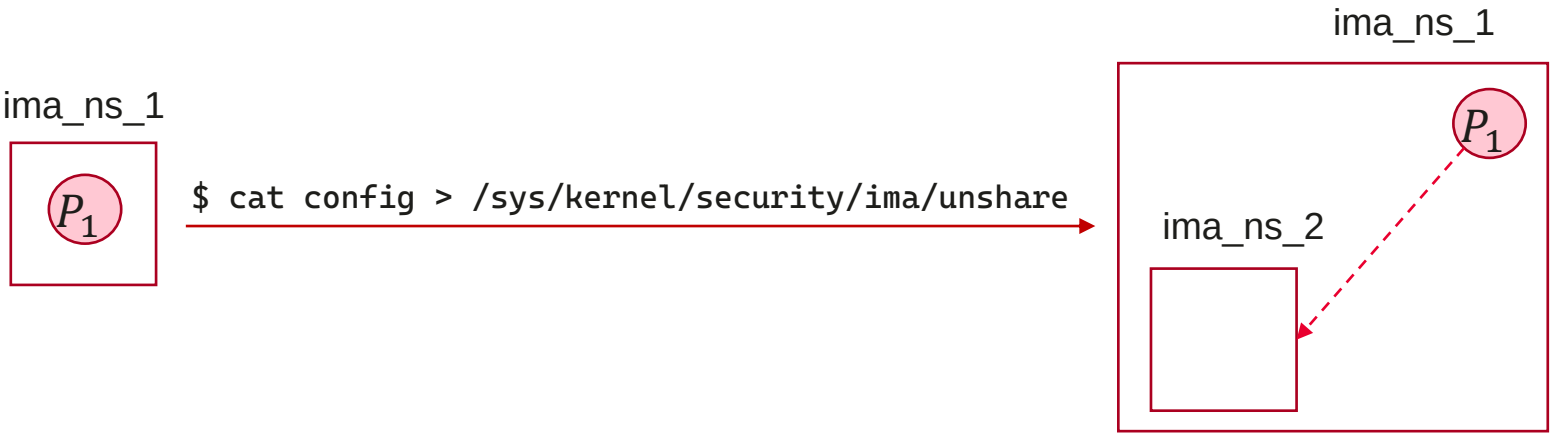
Join performed during execve() (2)



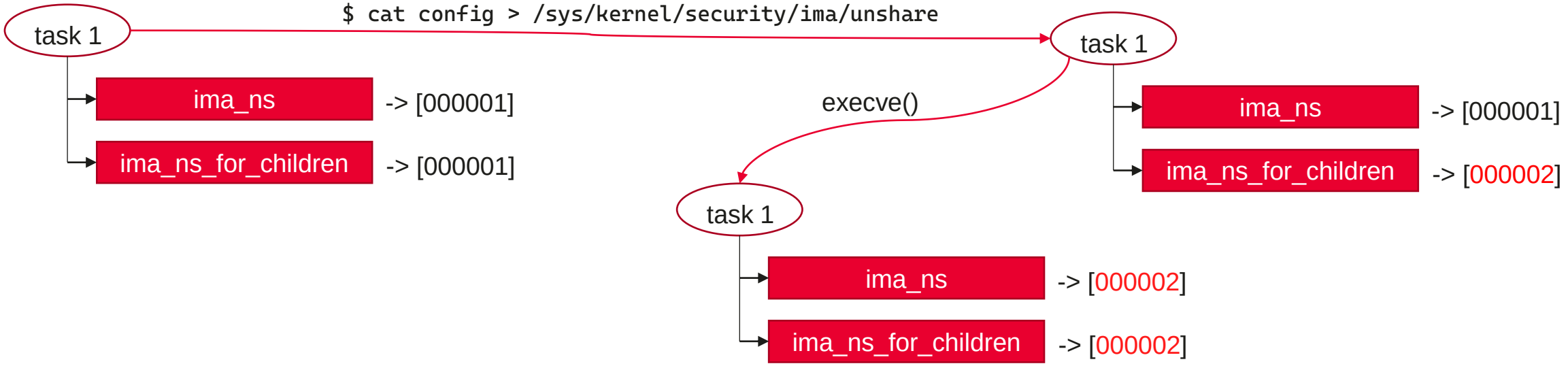
ima_ns_1

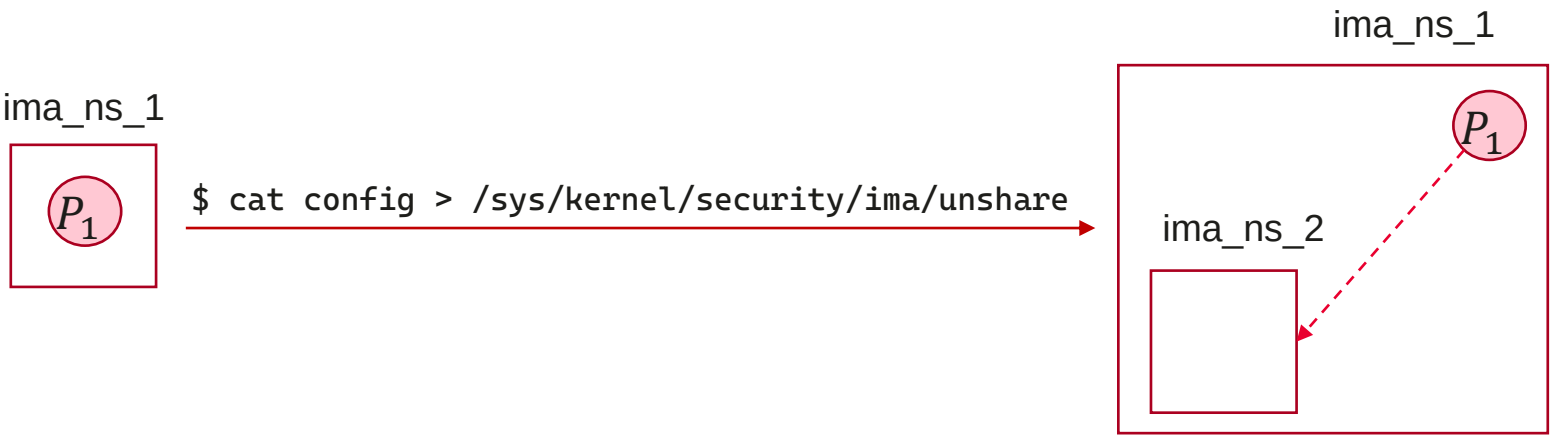
Join performed during execve() (2)



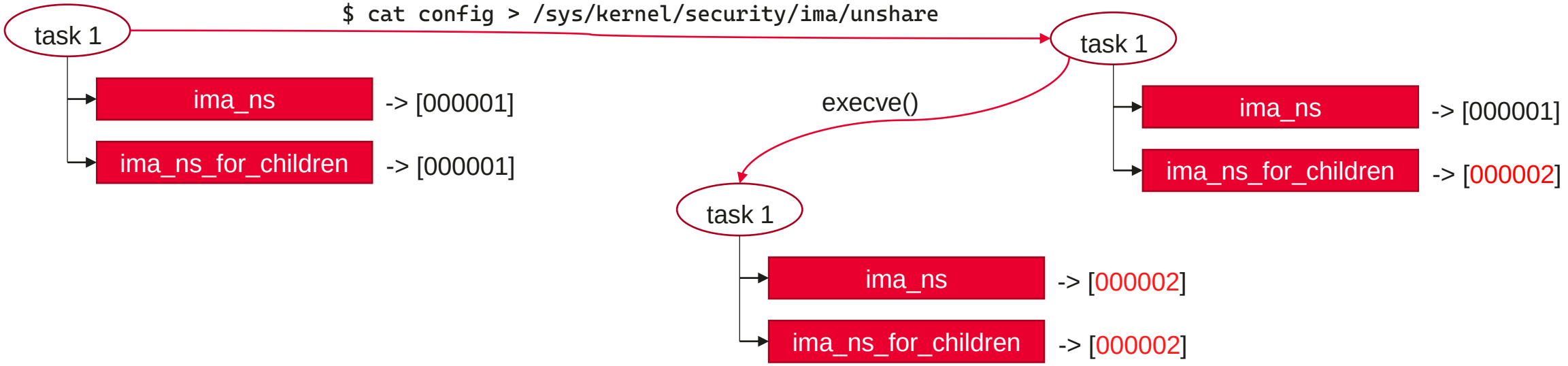


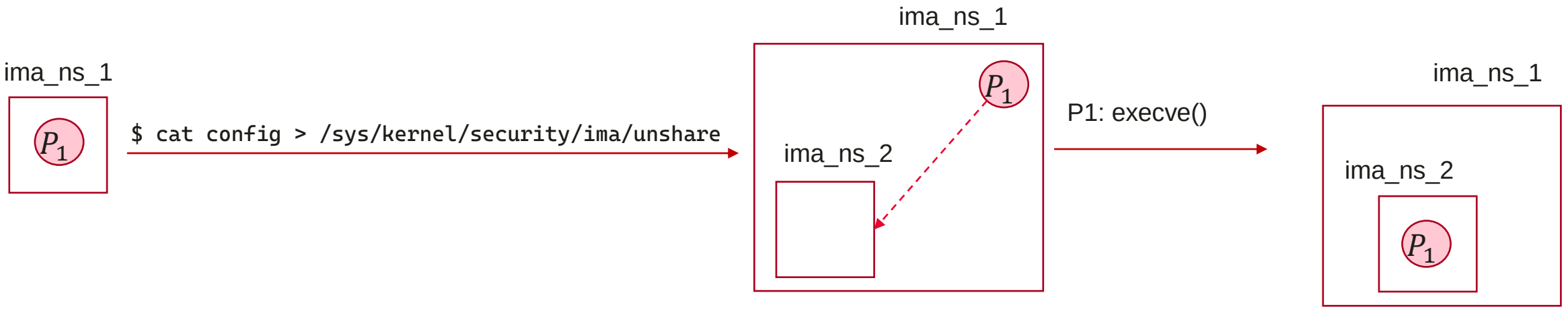
Join performed during execve() (2)





Join performed during execve() (2)





Open Problems

- Protect each IMA namespace measurements list with a RoT
- Ensure the IMA namespace has been configured accordingly to user requirements
- Ensure correct binding between IMA namespace and user application
- Which user namespace is used for IMA namespace policy evaluation (owner?)
- Performance impact (e.g. memory consumption)
- Per-IMA namespace metadata
  - Locking

Conclusion

- IMA namespace is promising for upstream
  - > Already reviewed by maintainers
- Stefan’s proposal very close to our requirements
  - > Issue: a clear design is missing
  - > We addressed this design gap
- Let’s discuss about our design changes and implementation

Thank you.

Bring digital to every person, home and organization for a fully connected, intelligent world.