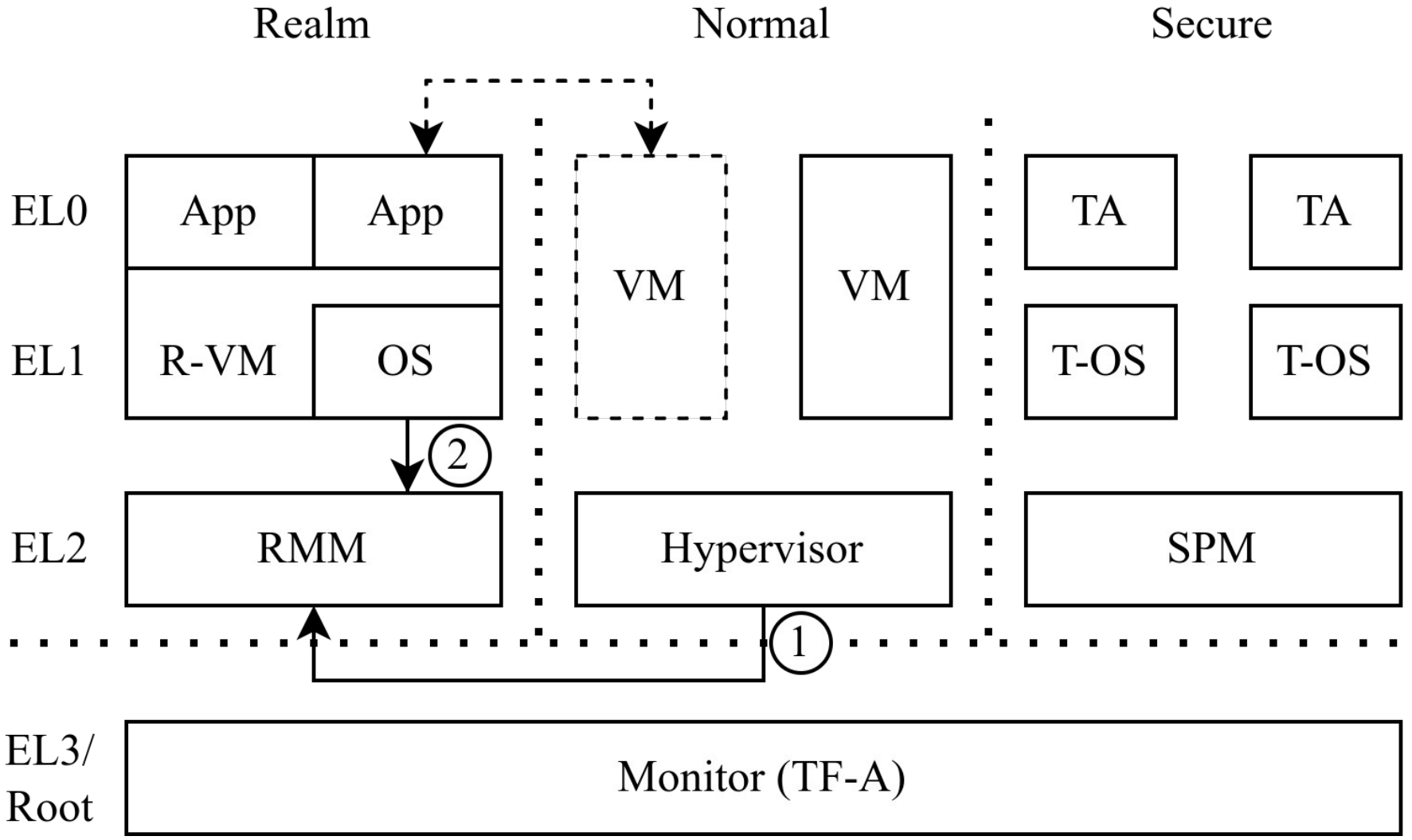
BarriCCAde:
Isolating Closed-Source Drivers with ARM CCA

Matti Schulze, Christian Lindenmeier, Jonas Röckl

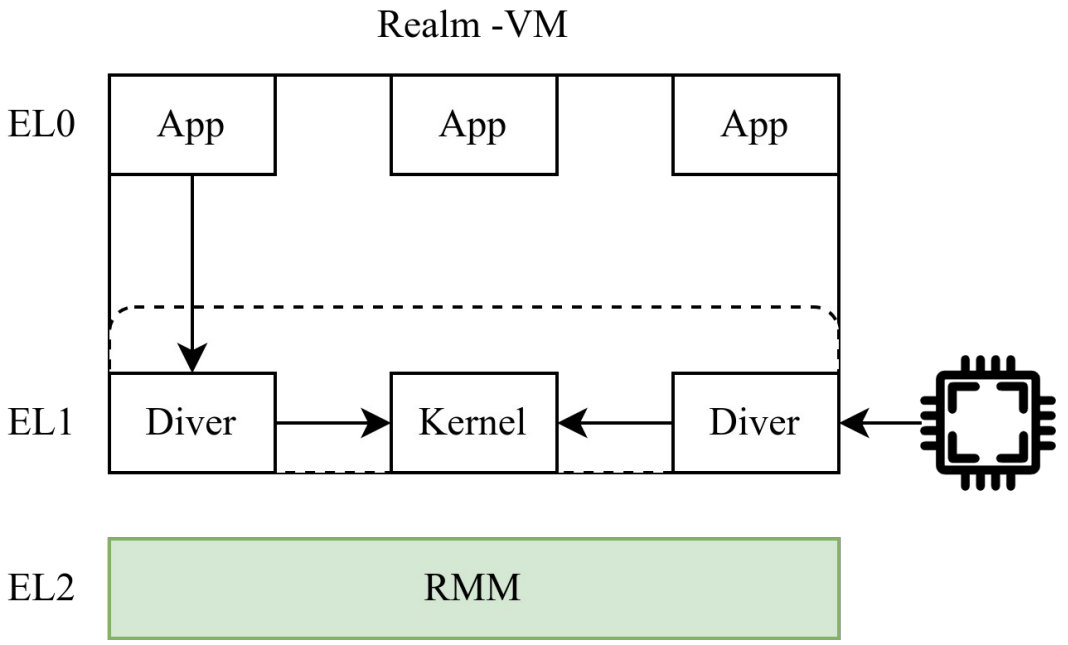
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

ARM CCA


Realm
Normal
Secure
EL0
App
App
VM
VM
TA
TA
EL1
R-VM
OS
T-OS
T-OS
2
EL2
RMM
Hypervisor
SPM
1
EL3/
Root
Monitor (TF-A)

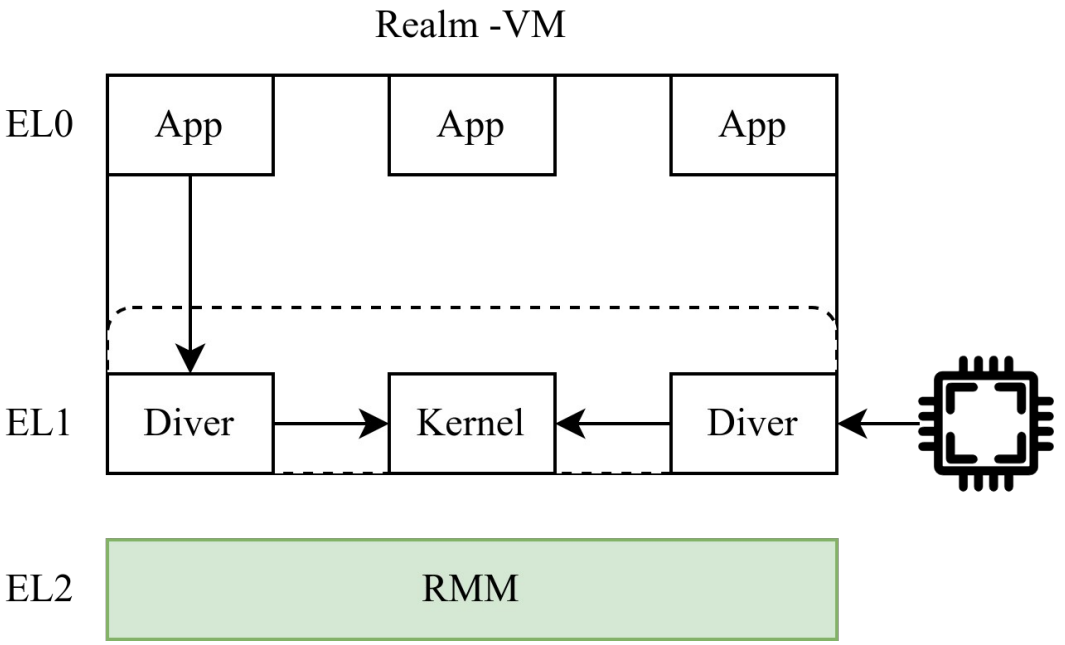
Attacker Model

- CCA offers protection from untrusted software running „outside“ the R-VM
- Attacker „inside“ the R-VM still a siginificant danger



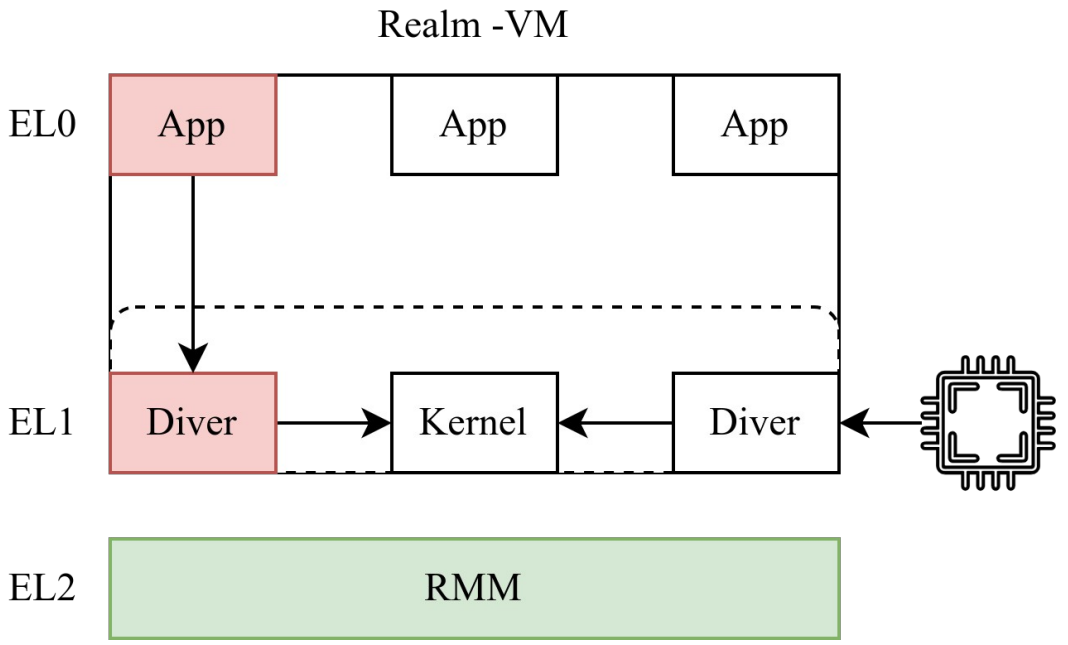
Attacker Model

- CCA offers protection from untrusted software running „outside“ the R-VM
- Attacker „inside“ the R-VM still a siginificant danger
- Most privilege escalations stem from vulnerable drivers



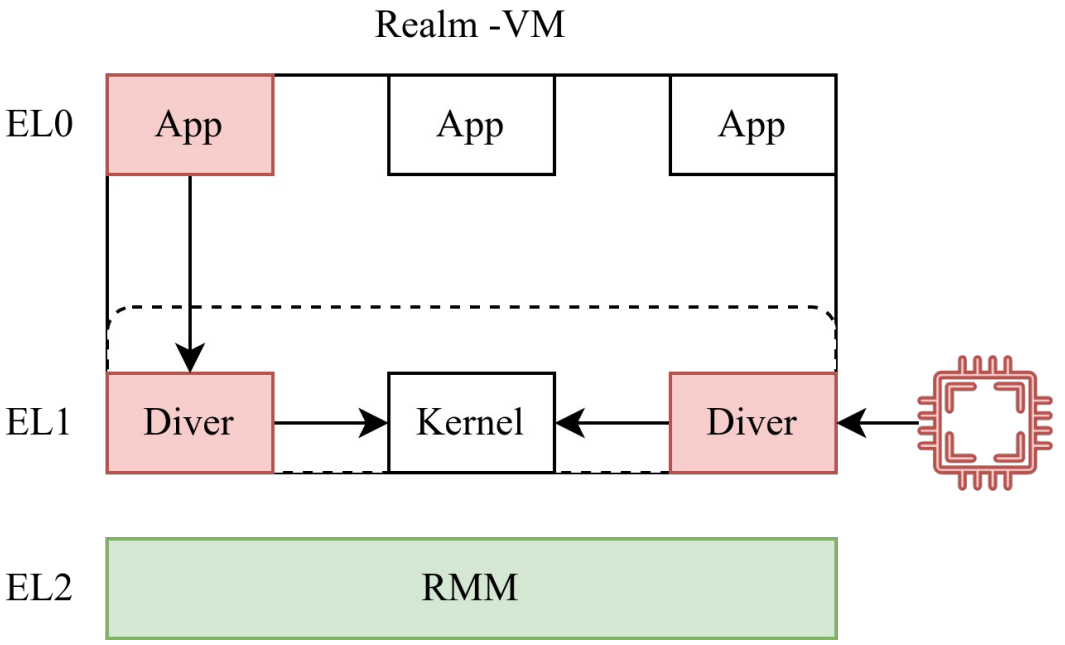
Attacker Model

- CCA offers protection from untrusted software running „outside“ the R-VM
- Attacker „inside“ the R-VM still a siginificant danger
- Most privilege escalations stem from vulnerable drivers
- Apps can exploit these vulnerabilies



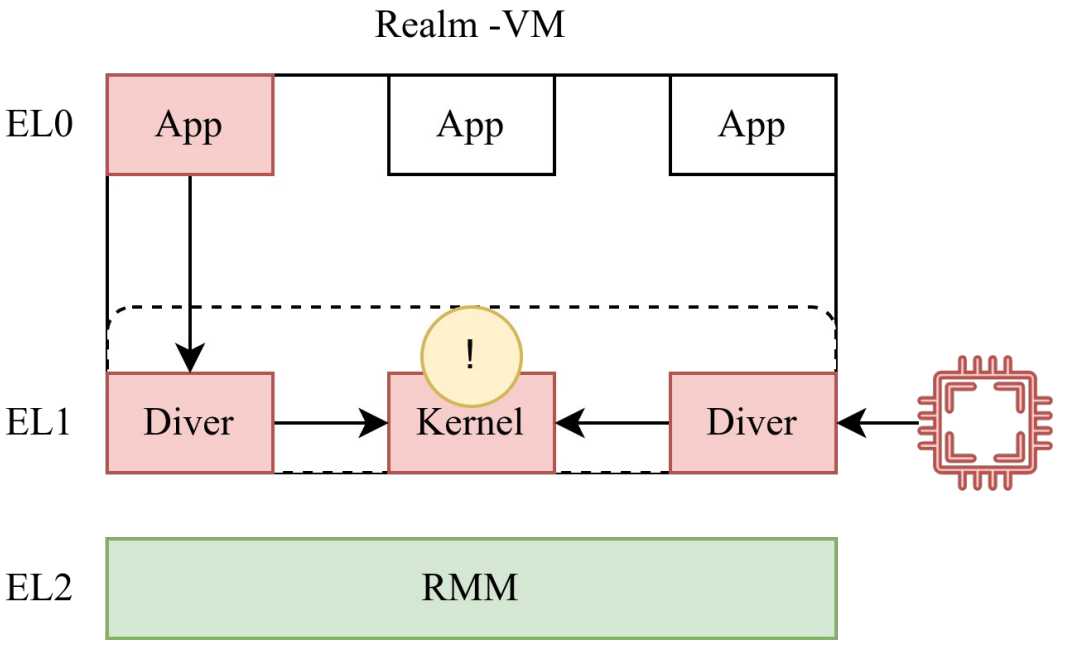
Attacker Model

- CCA offers protection from untrusted software running „outside“ the R-VM
- Attacker „inside“ the R-VM still a siginificant danger
- Most privilege escalations stem from vulnerable drivers
- Apps can exploit these vulnerabilies
- Peripherals can trigger vulnerabilies



Attacker Model

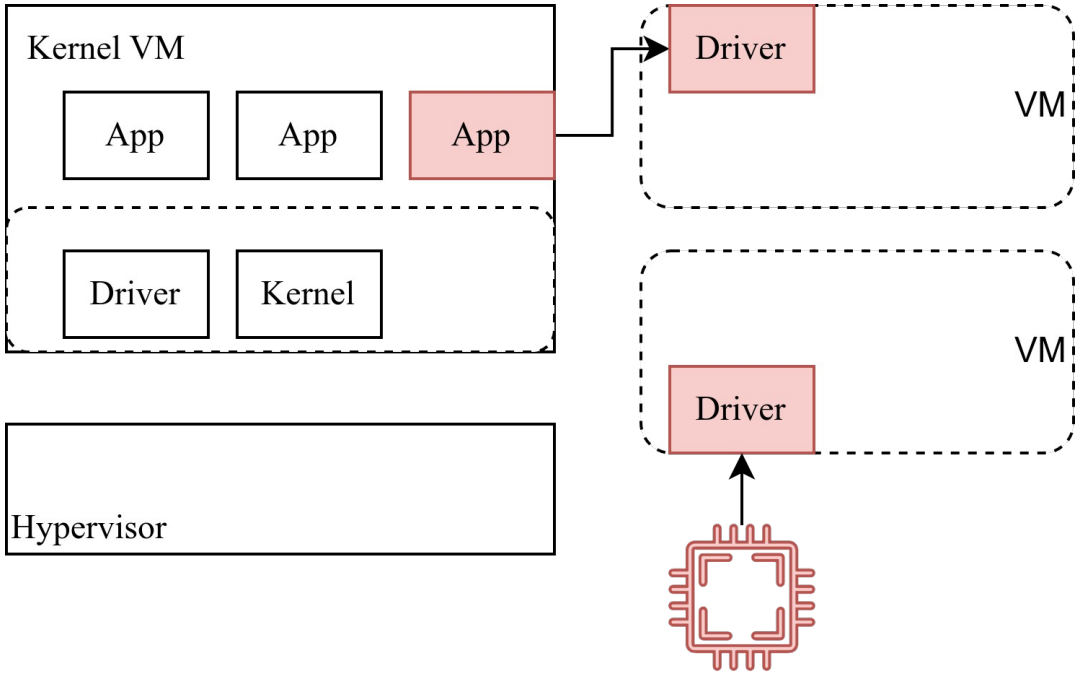
- CCA offers protection from untrusted software running „outside“ the R-VM
- Attacker „inside“ the R-VM still a siginificant danger
- Most privilege escalations stem from vulnerable drivers
- Apps can exploit these vulnerabilies
- Peripherals can trigger vulnerabilies



Monolithic Kernel => System Compromise

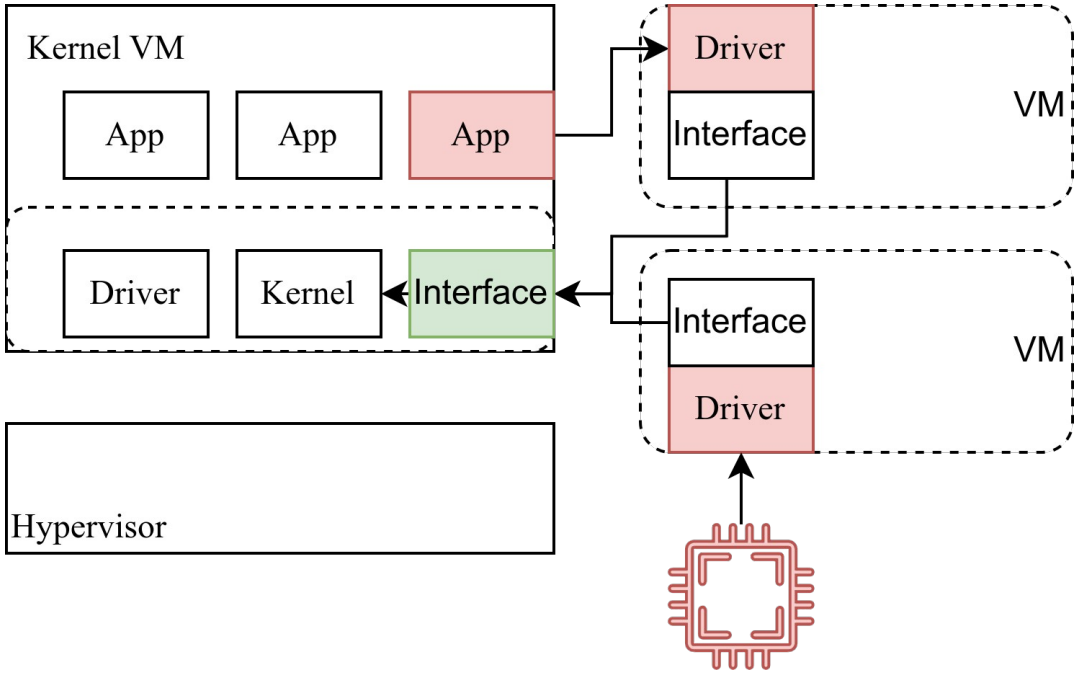
Challenges

- State of the art solutions isolate Drivers into VMs



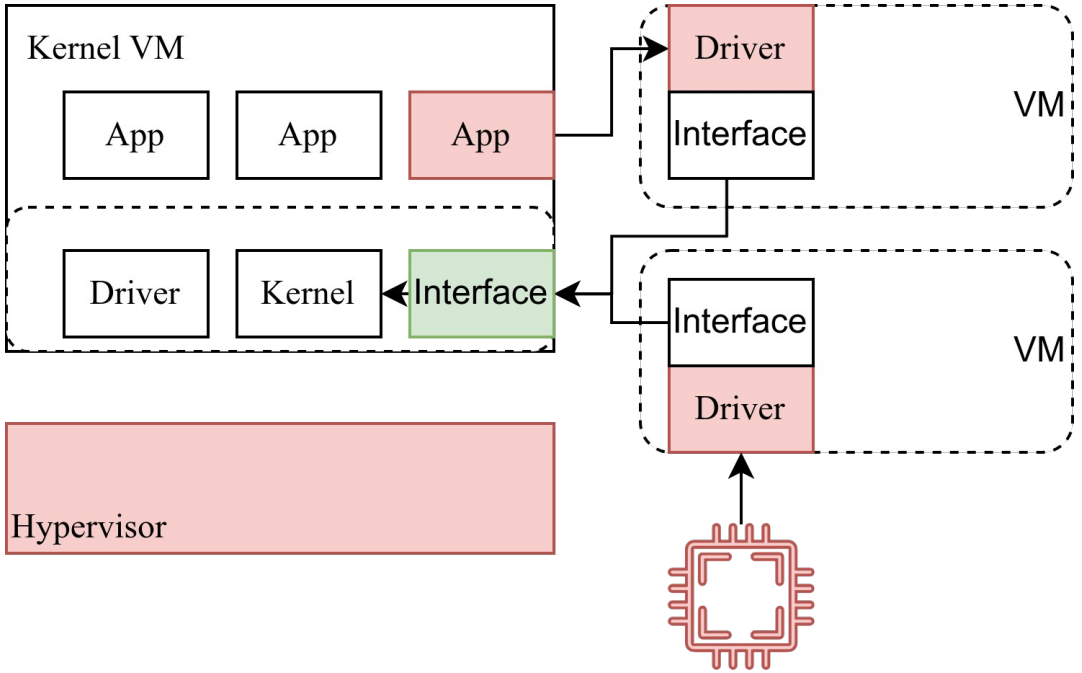
Challenges

- State of the art solutions isolate Drivers into VMs
- Add Interface to Drivers to communicate with the Kernel VM
  - Forward Kernel requests
  - Synchronize Resources



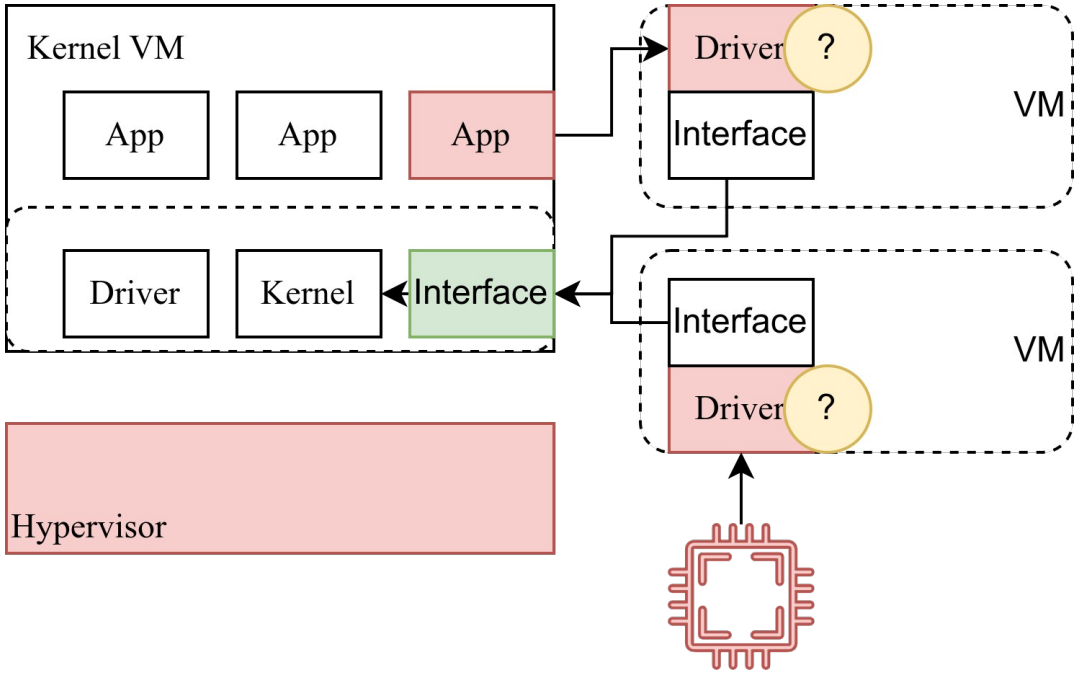
Challenges

- State of the art solutions isolate Drivers into VMs
- Add Interface to Drivers to communicate with the Kernel VM
  - Forward Kernel requests
  - Synchronize Resources
- Problems:
  - Adding Hypervisor to the TCB



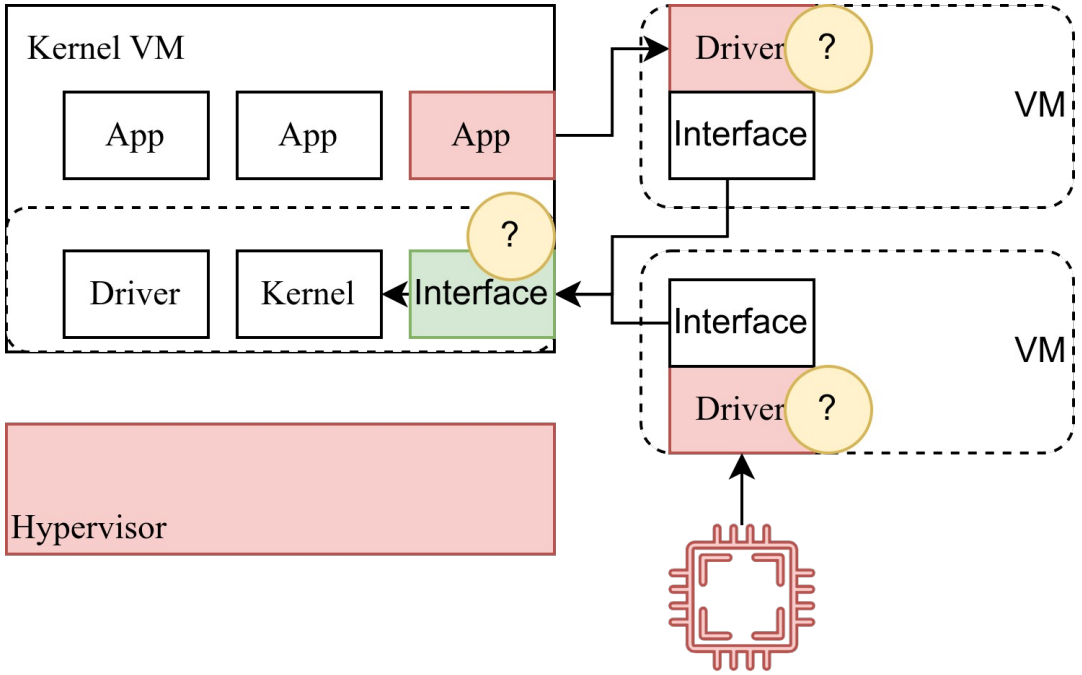
Challenges

- State of the art solutions isolate Drivers into VMs
- Add Interface to Drivers to communicate with the Kernel VM
  - Forward Kernel requests
  - Synchronize Resources
- Problems:
  - Adding Hypervisor to the TCB
  - What about closed-source Drivers?



Challenges

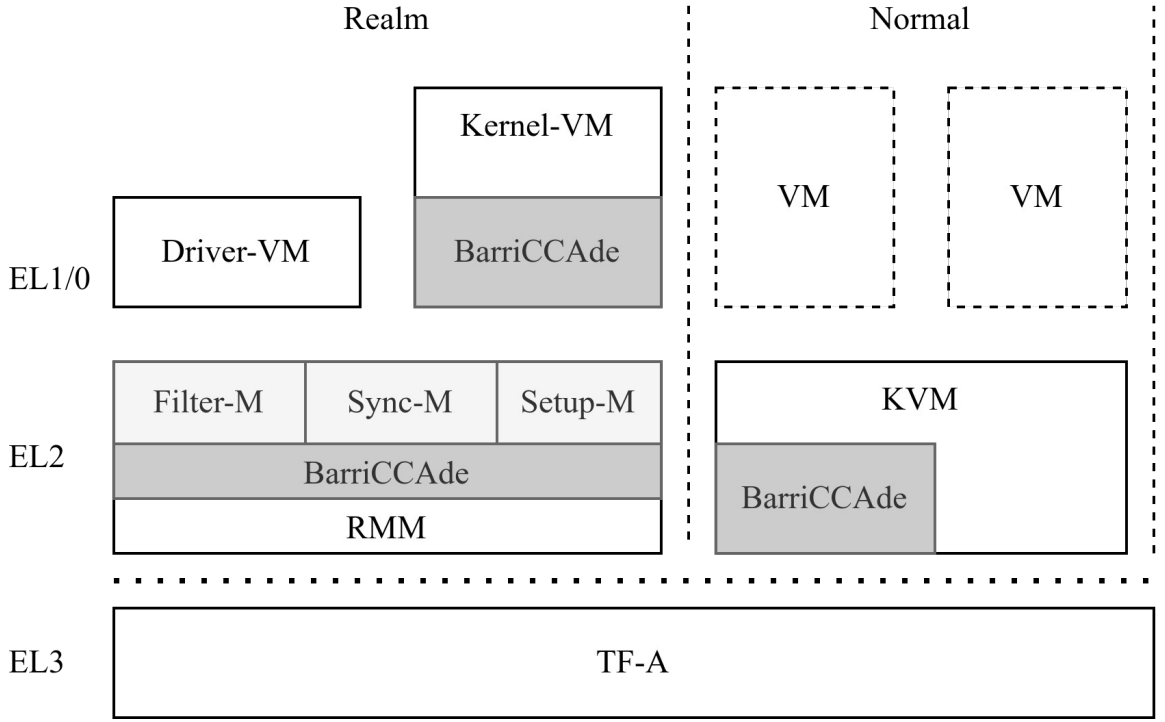
- State of the art solutions isolate Drivers into VMs
- Add Interface to Drivers to communicate with the Kernel VM
  - Forward Kernel requests
  - Synchronize Resources
- Problems:
  - Adding Hypervisor to the TCB
  - What about closed-source drivers?
  - How to actually filter out „evil“ requests?



BarriCCAde

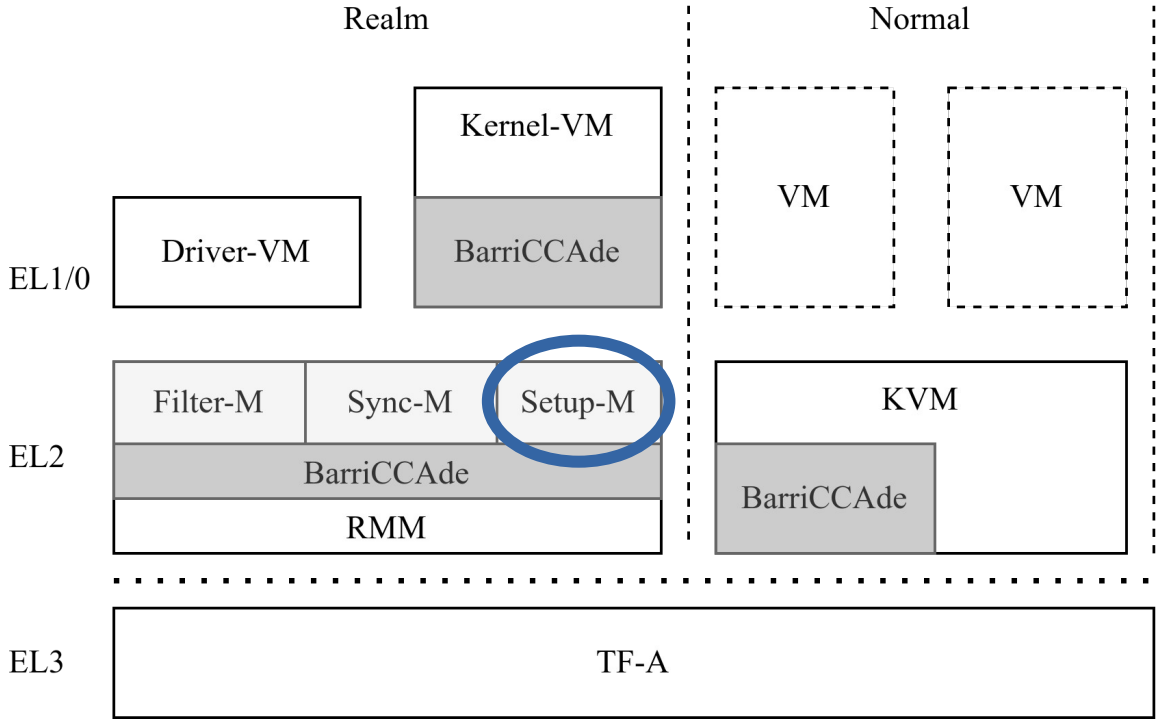
BarriCCAde

- Utilize CCA to not increase the TCB



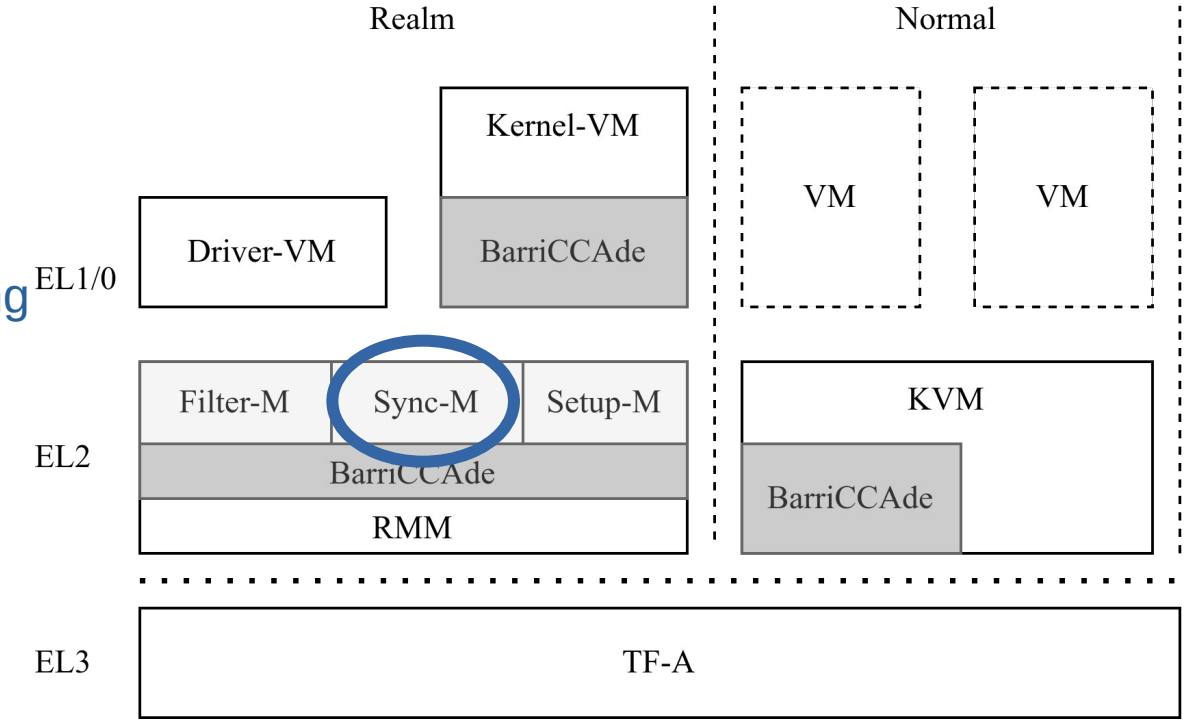
BarriCCAde

- Utilize CCA to not increase the TCB
- Expand the RMM to implement modules handling
  - The setup of Driver-VMs



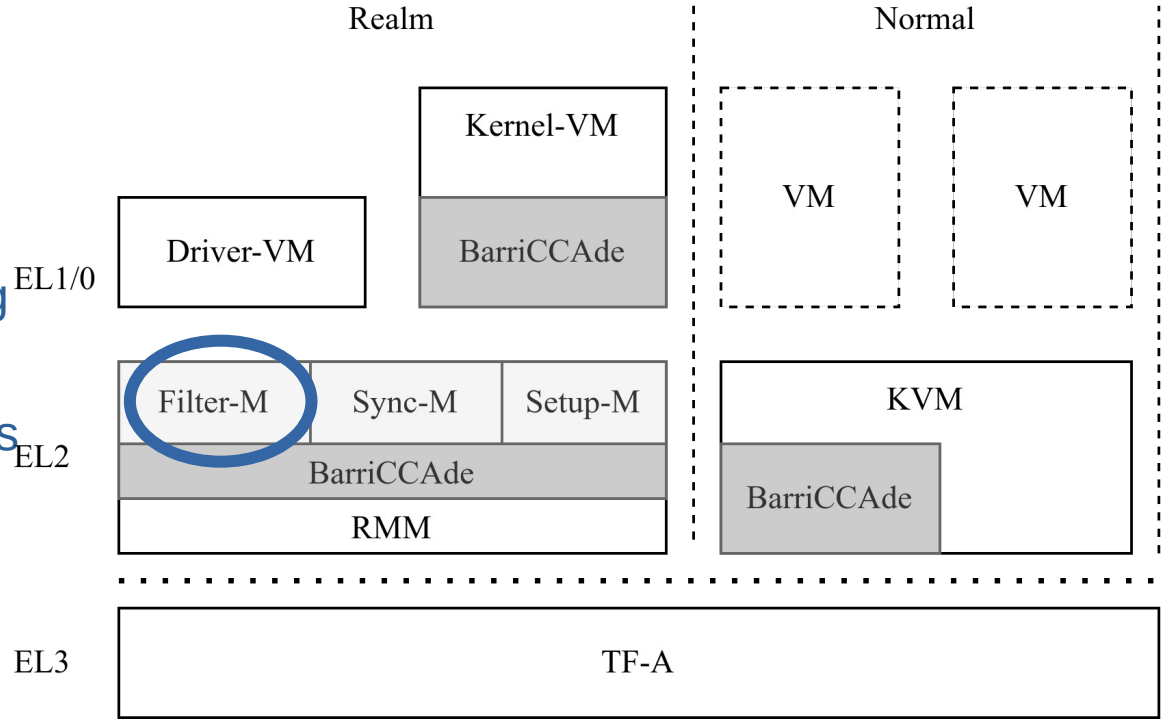
BarriCCAde

- Utilize CCA to not increase the TCB
- Expand the RMM to implement modules handling
  - The setup of Driver-VMs
  - The synchronization of resources including for closed-source drivers



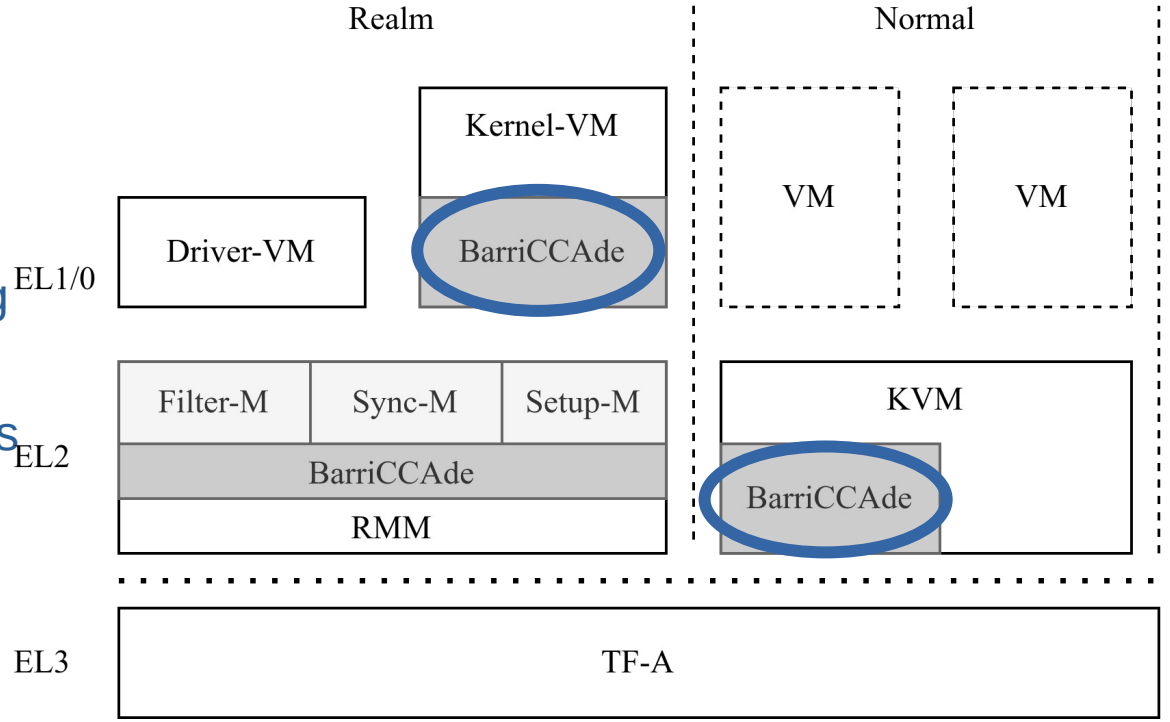
BarriCCAde

- Utilize CCA to not increase the TCB
- Expand the RMM to implement modules handling
  - The setup of Driver-VMs
  - The synchronization of resources including for closed-source drivers
  - Configurable filtering of malicious accesses



BarriCCAde

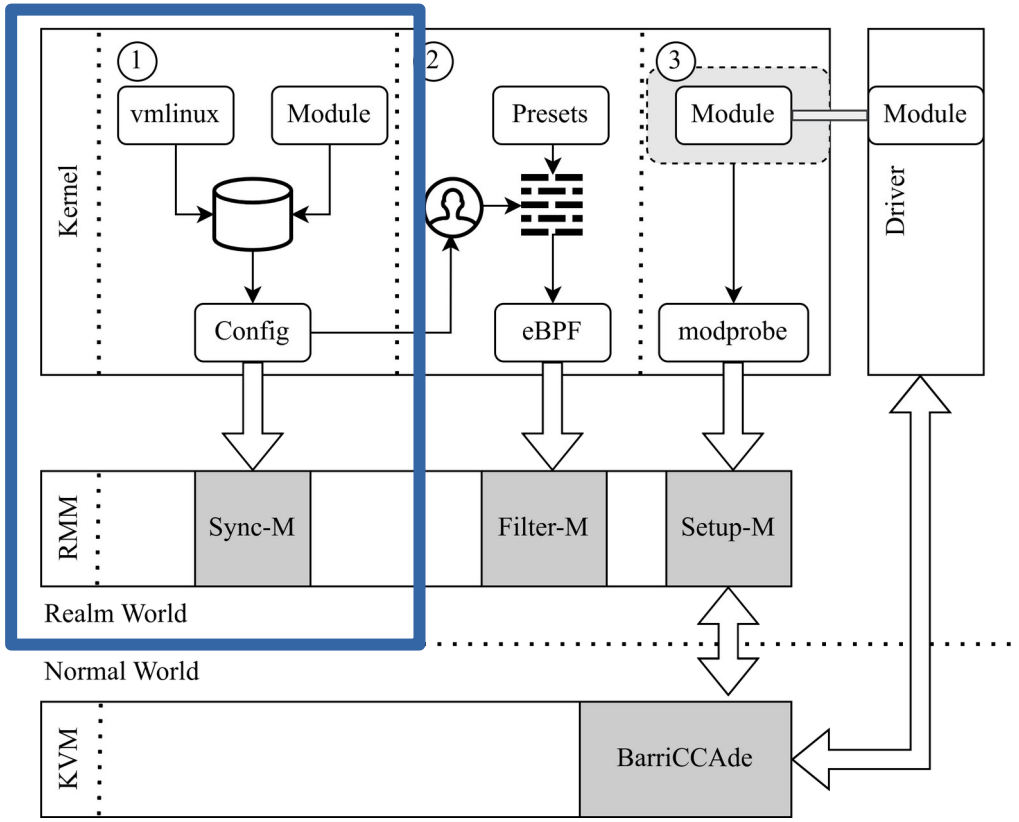
- Utilize CCA to not increase the TCB
- Expand the RMM to implement modules handling
  - The setup of Driver-VMs
  - The synchronization of resources including for closed-source drivers
  - Configurable filtering of malicious accesses
- Expand other untrusted components as needed



BarriCCAde - Setup

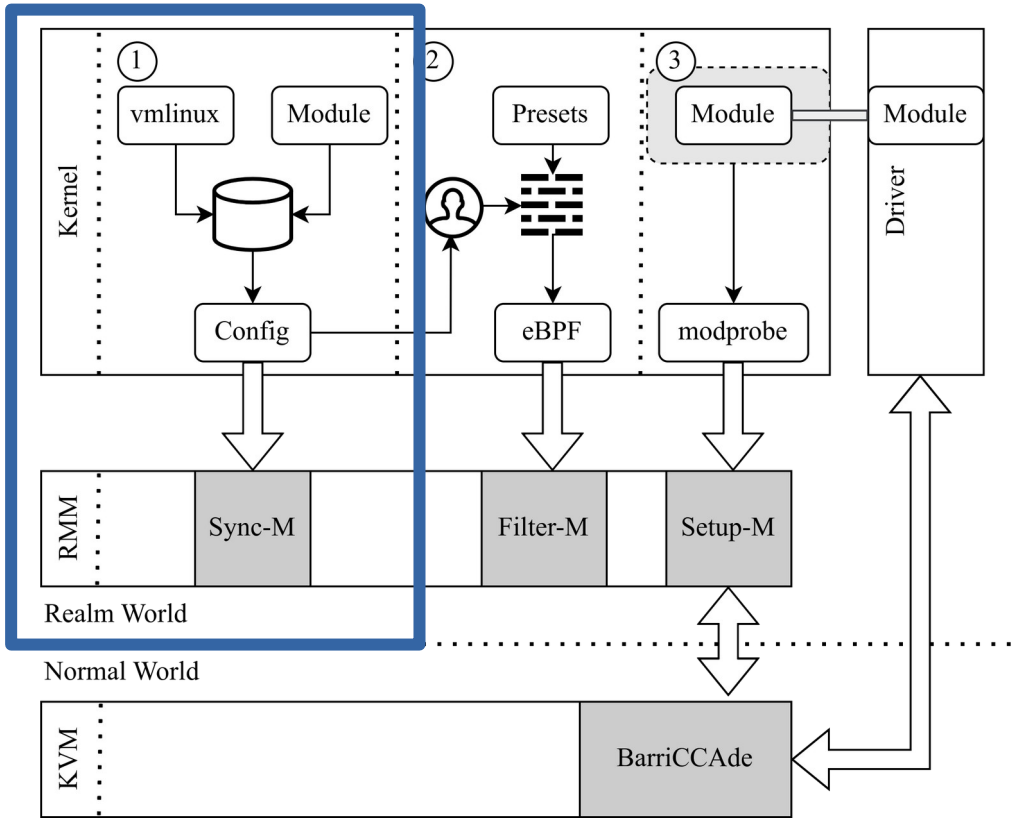
BarriCCAde – Setup

- For resource synchronization memory locations, size and semantics of the resources must be understood



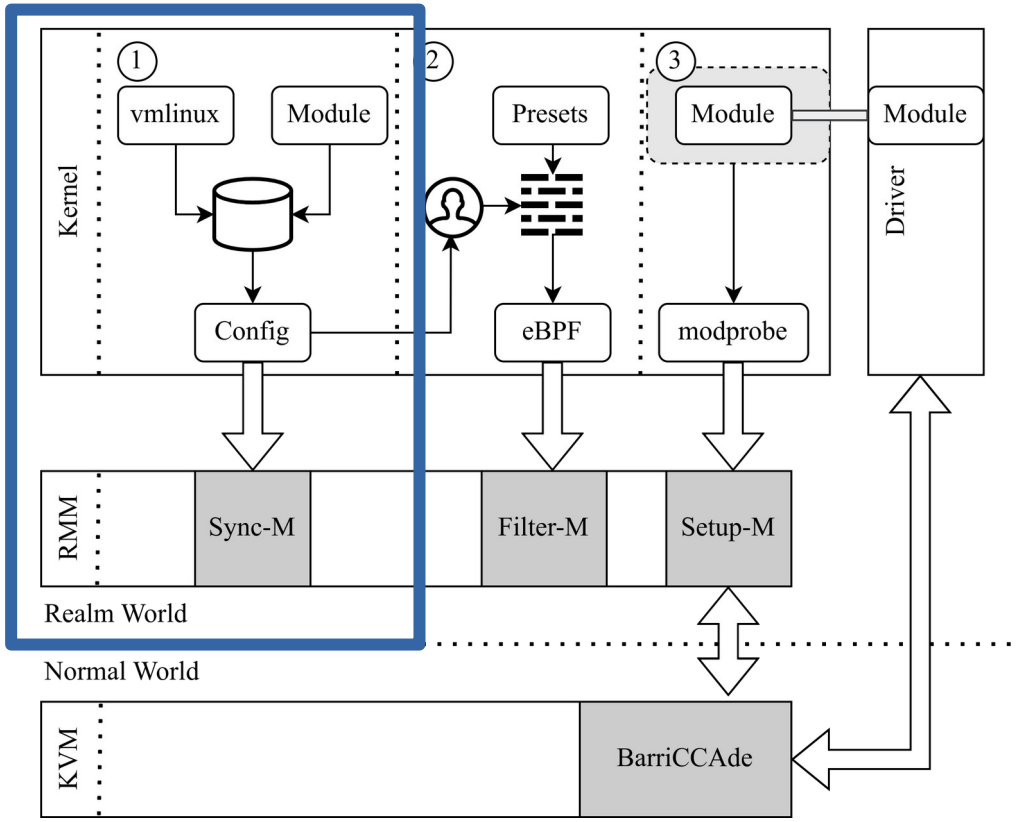
BarriCCAde – Setup

- For resource synchronization memory locations, size and semantics of the resources must be understood
- Kernel creates **Sync-Config** by analyzing the used Kernel resources from the Driver ELF



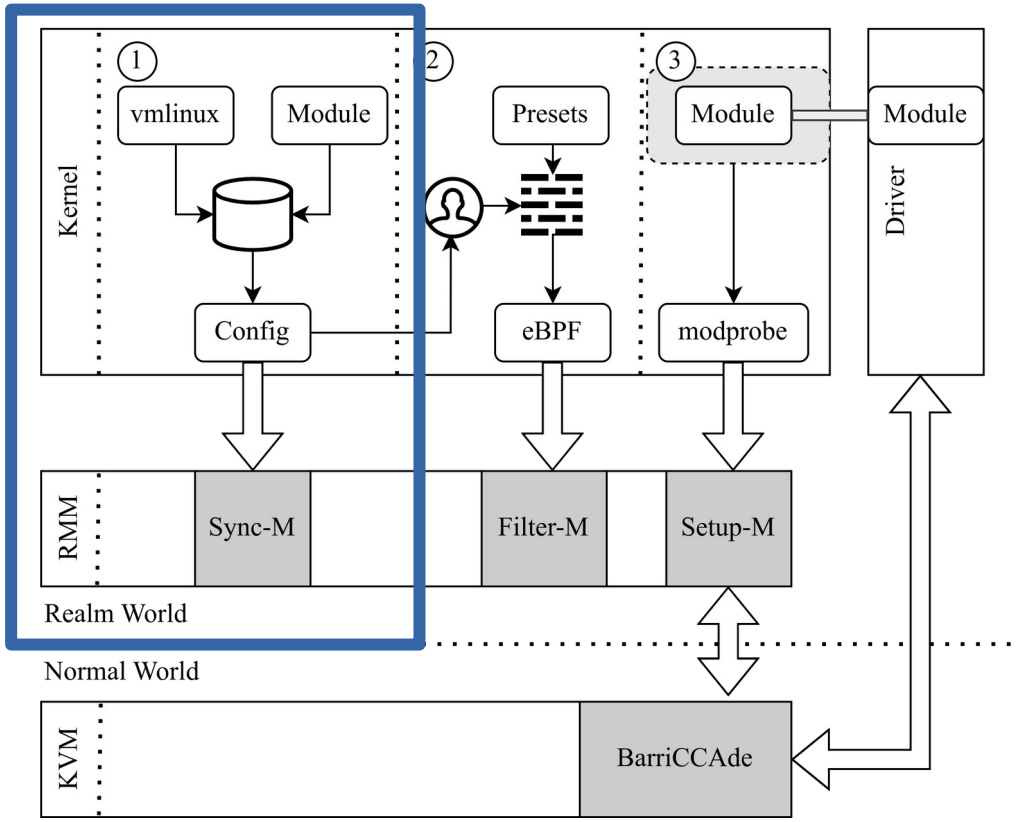
BarriCCAde – Setup

- For resource synchronization memory locations, size and semantics of the resources must be understood
- Kernel creates **Sync-Config** by analyzing the used Kernel resources from the Driver ELF
- Matches resources to the debug information in the Kernel
  - Operands, size of operands, pointers, …



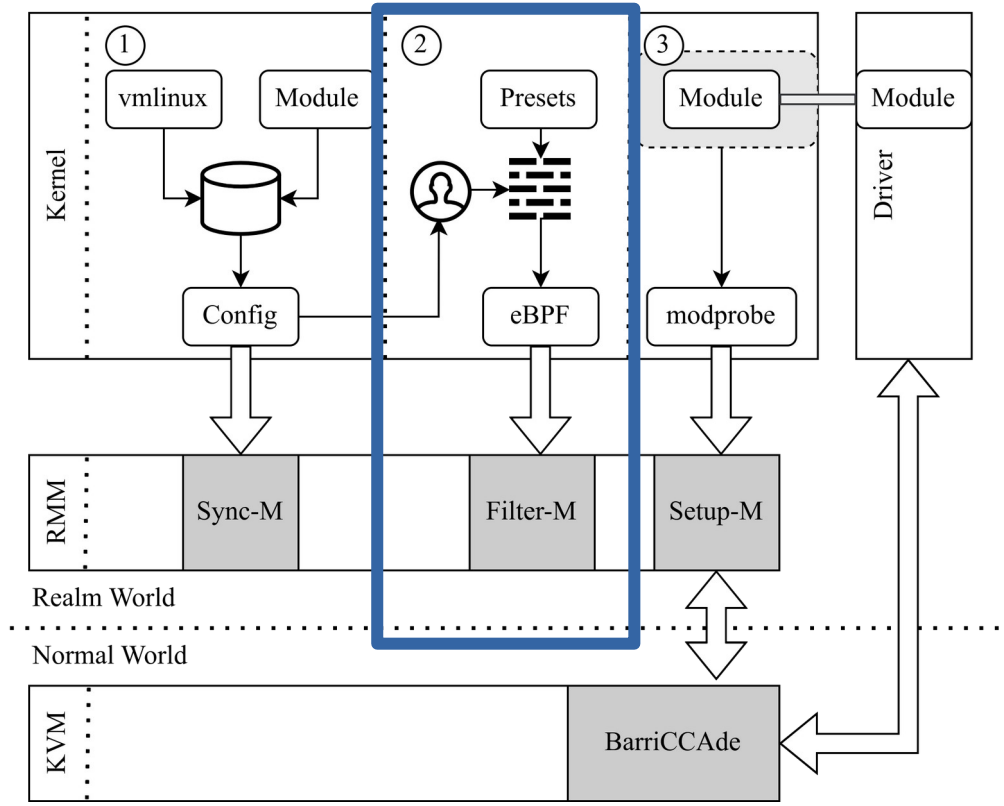
BarriCCAde – Setup

- For resource synchronization memory locations, size and semantics of the resources must be understood
- Kernel creates **Sync-Config** by analyzing the used Kernel resources from the Driver ELF
- Matches resources to the debug information in the Kernel
  - Operands, size of operands, pointers, …
- Evaluate recursively, load into **Sync-M**



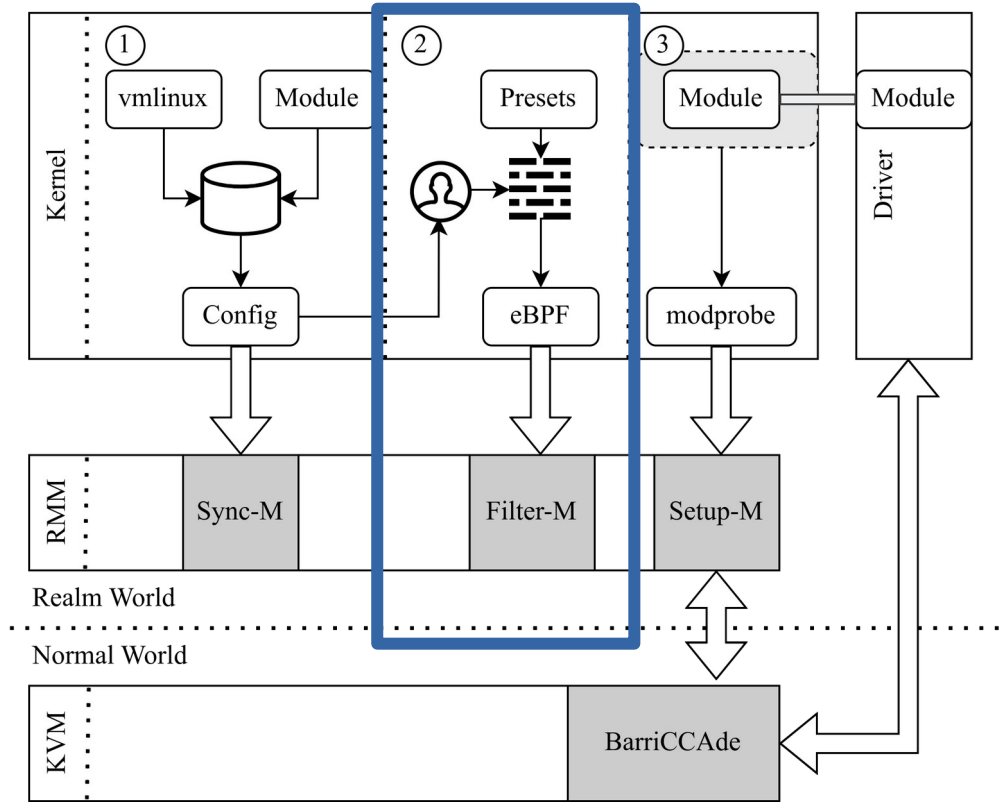
BarriCCAde – Setup

- Config gives hint how to filter accessess
  - e.g., prevent access to task structs
  - e.g., prevent accessess to resources not listed in the config
  - …



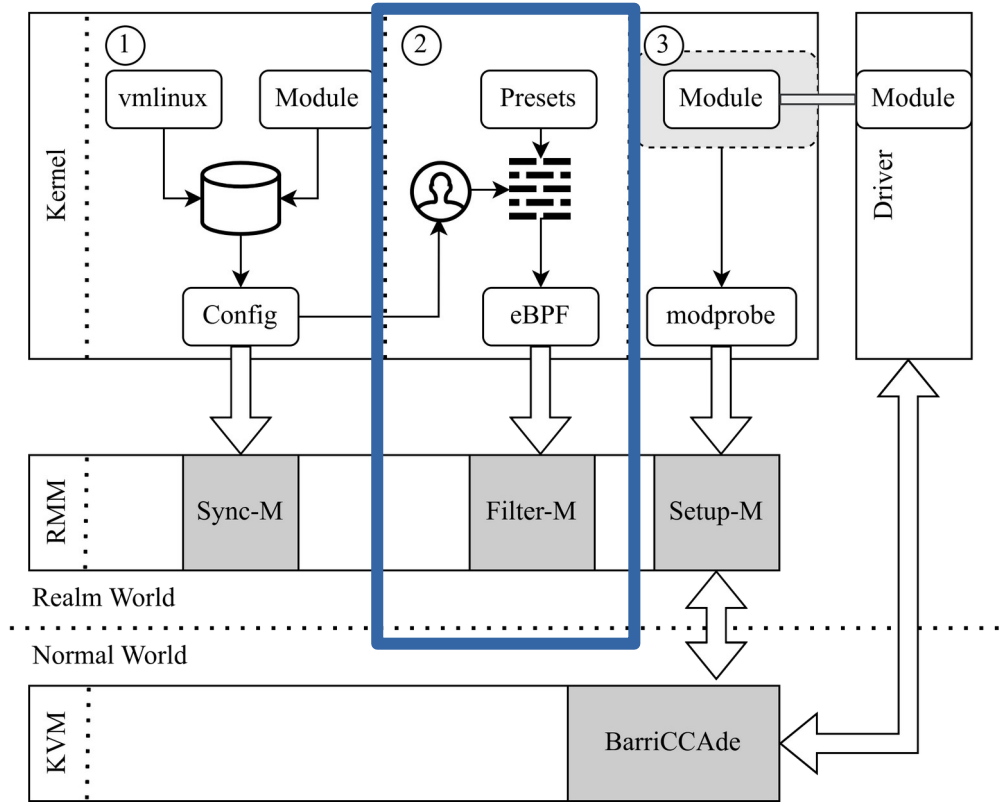
BarriCCAde – Setup

- Config gives hint how to filter accessess
  - e.g., prevent access to task structs
  - e.g., prevent accessess to resources not listed in the config
  - …
- Manual effort



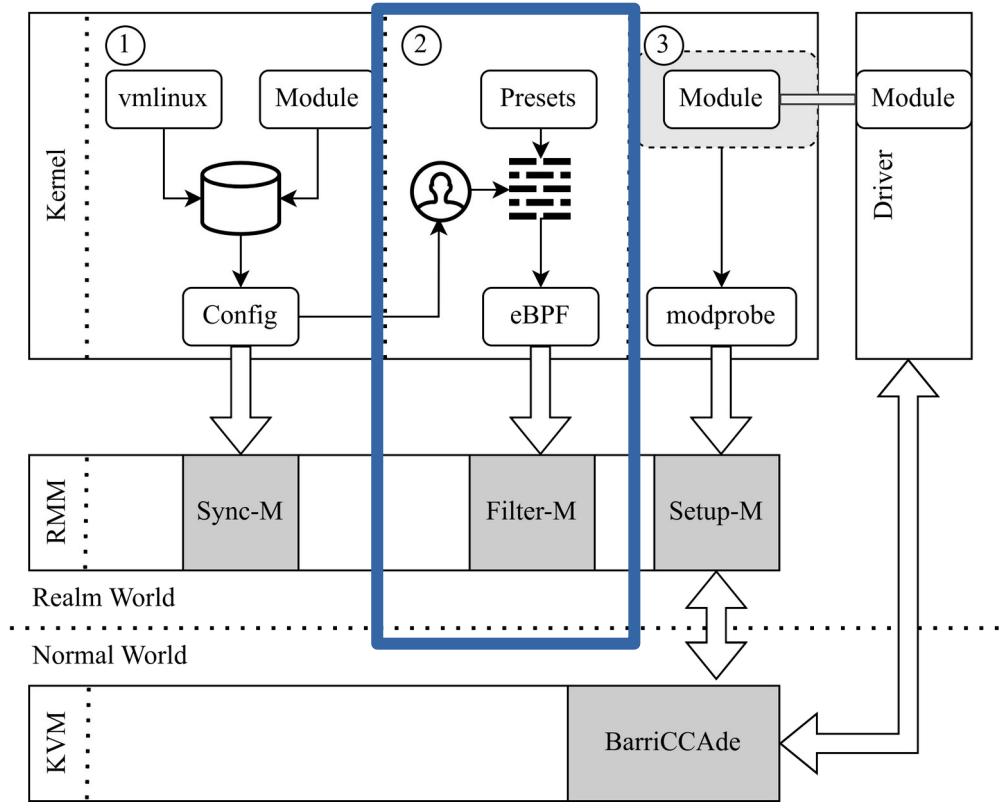
BarriCCAde – Setup

- Config gives hint how to filter accessess
  - e.g., prevent access to task structs
  - e.g., prevent accessess to resources not listed in the config
  - …
- Manual effort
- Vendor provided



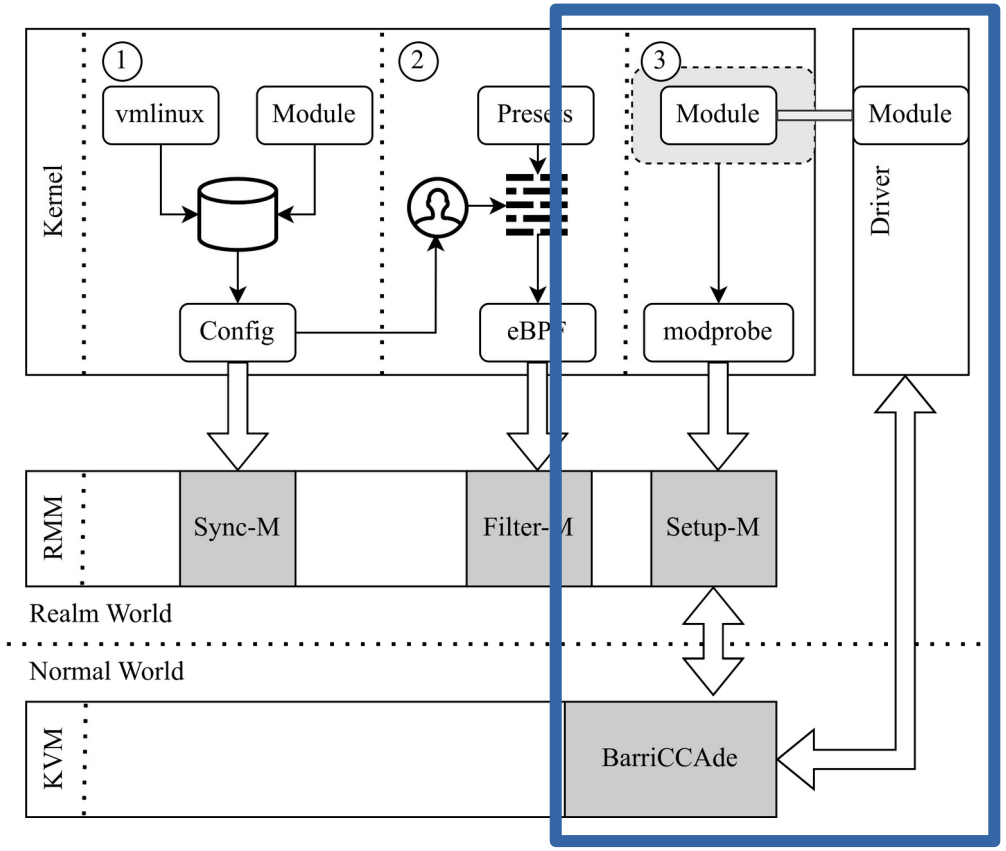
BarriCCAde – Setup

- Config gives hint how to filter accessess
  - e.g., prevent access to task structs
  - e.g., prevent accessess to resources not listed in the config
  - …
- Manual effort
- Vendor provided
- Compile into eBPF Program and load into **Filter-M**



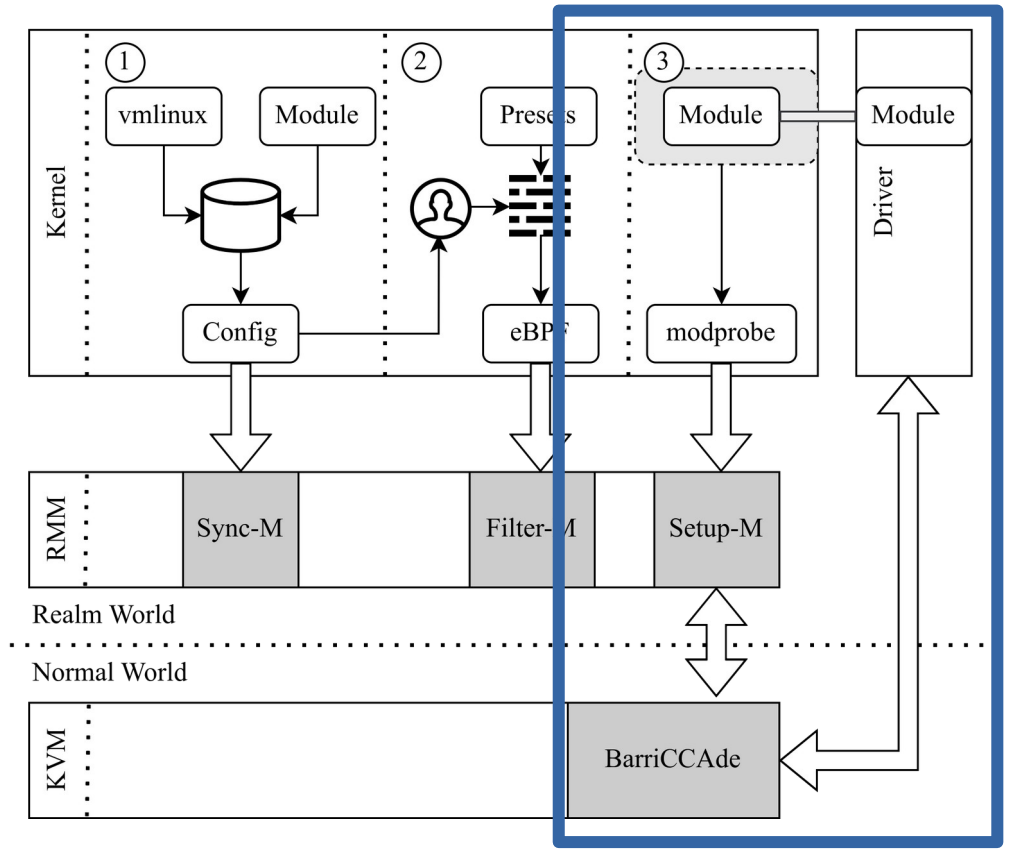
BarriCCAde – Setup

- Load module via modprobe



BarriCCAde – Setup

- Load module via modprobe
- Forward bounds to **Setup-M**
  - Creates Driver-VM



BarriCCAde - Runtime

BarriCCAde – Runtime

- At runtime, requests for Kernel/Driver resources must be forwarded accordingly

BarriCCAde – Runtime

- At runtime, requests for Kernel/Driver resources must be forwarded accordingly
- Usually done via software added to the Driver

  => What about a closed source scenario?

BarriCCAde – Runtime

- At runtime, requests for Kernel/Driver resources must be forwarded accordingly
- Usually done via software added to the Driver

  => What about a closed source scenario?
- Mark module pages in kernel as inacessible via the RMM

  => Accesses trapped to RMM

BarriCCAde – Runtime

- At runtime, requests for Kernel/Driver resources must be forwarded accordingly
- Usually done via software added to the Driver

  => What about a closed source scenario?
- Mark module pages in kernel as inacessible via the RMM

  => Accesses trapped to RMM
- Copy Driver after relocation

  => Kernel resources not mapped

  => Accesses trapped to RMM

BarriCCAde – Runtime

- At runtime, requests for Kernel/Driver resources must be forwarded accordingly
- Usually done via software added to the Driver

  => What about a closed source scenario?
- Mark module pages in kernel as inacessible via the RMM

  => Accesses trapped to RMM
- Copy Driver after relocation

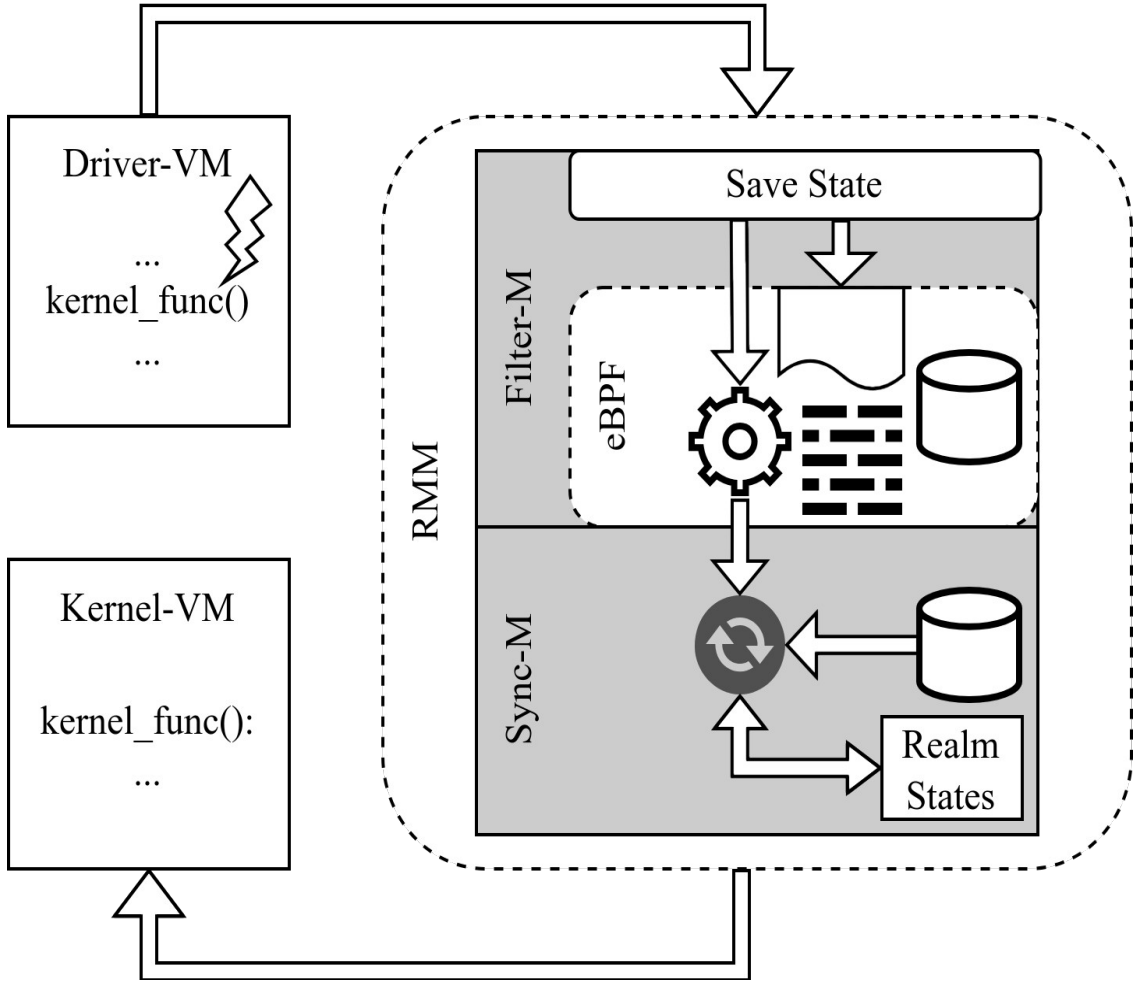
  => Kernel resources not mapped

  => Accesses trapped to RMM
- Trap context contains address accessed and state of the VM (e.g., params in registers)

  => Implicitly contains all information needed for access forwarding

BarriCCAde – Runtime

- Driver VM init function is called

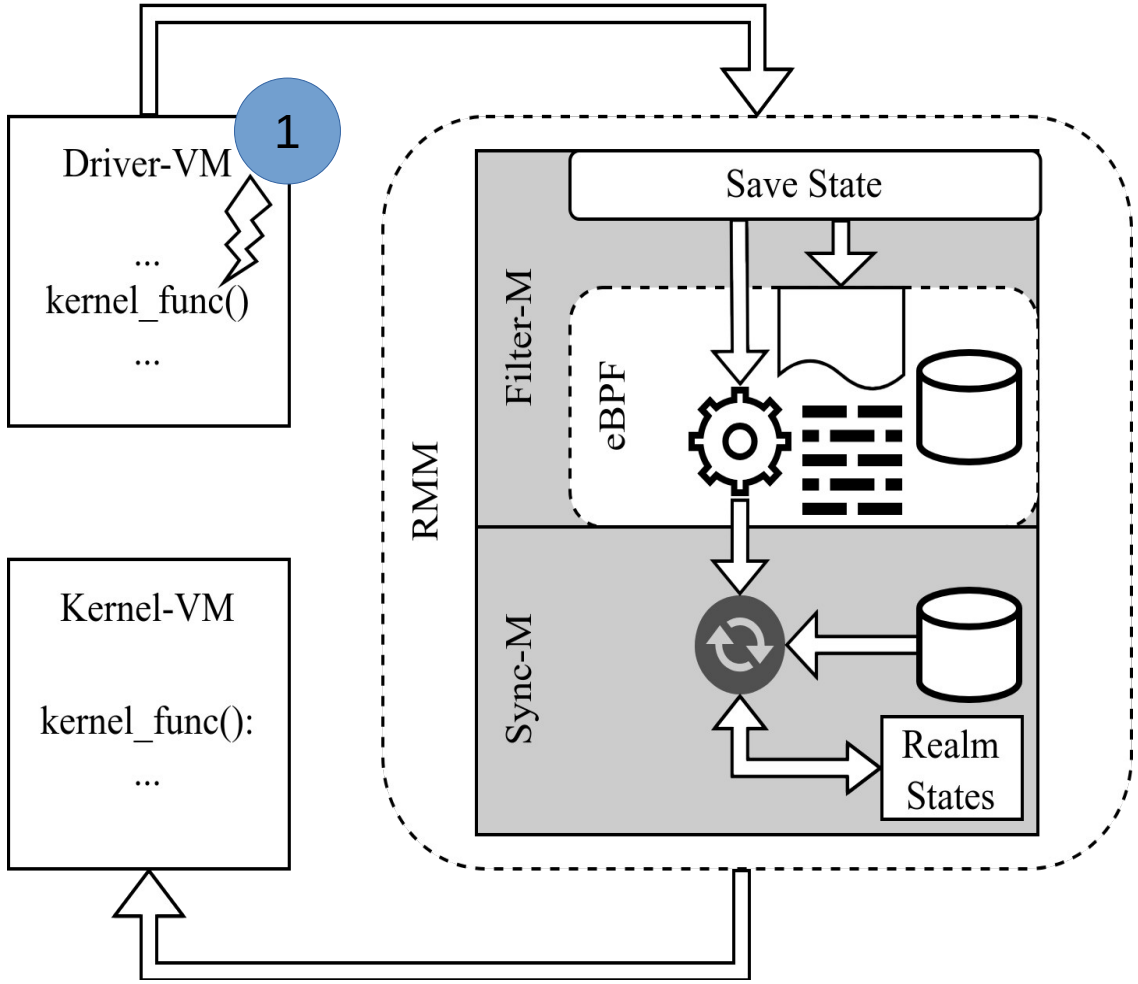


BarriCCAde – Runtime

- Driver VM init function is called
- Accessess kernel function

  => Not mapped

  => Trap to RMM

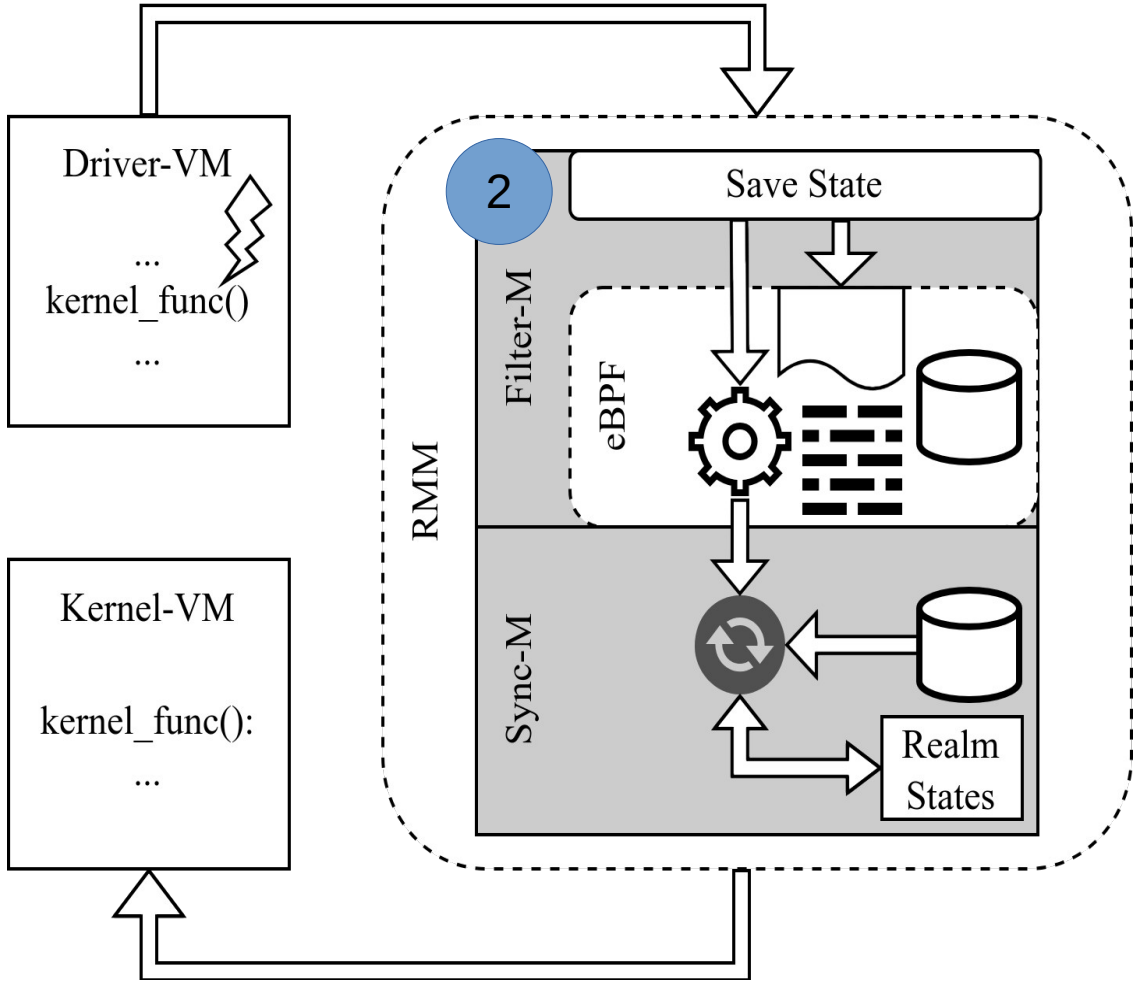


BarriCCAde – Runtime

- Driver VM init function is called
- Accessess kernel function

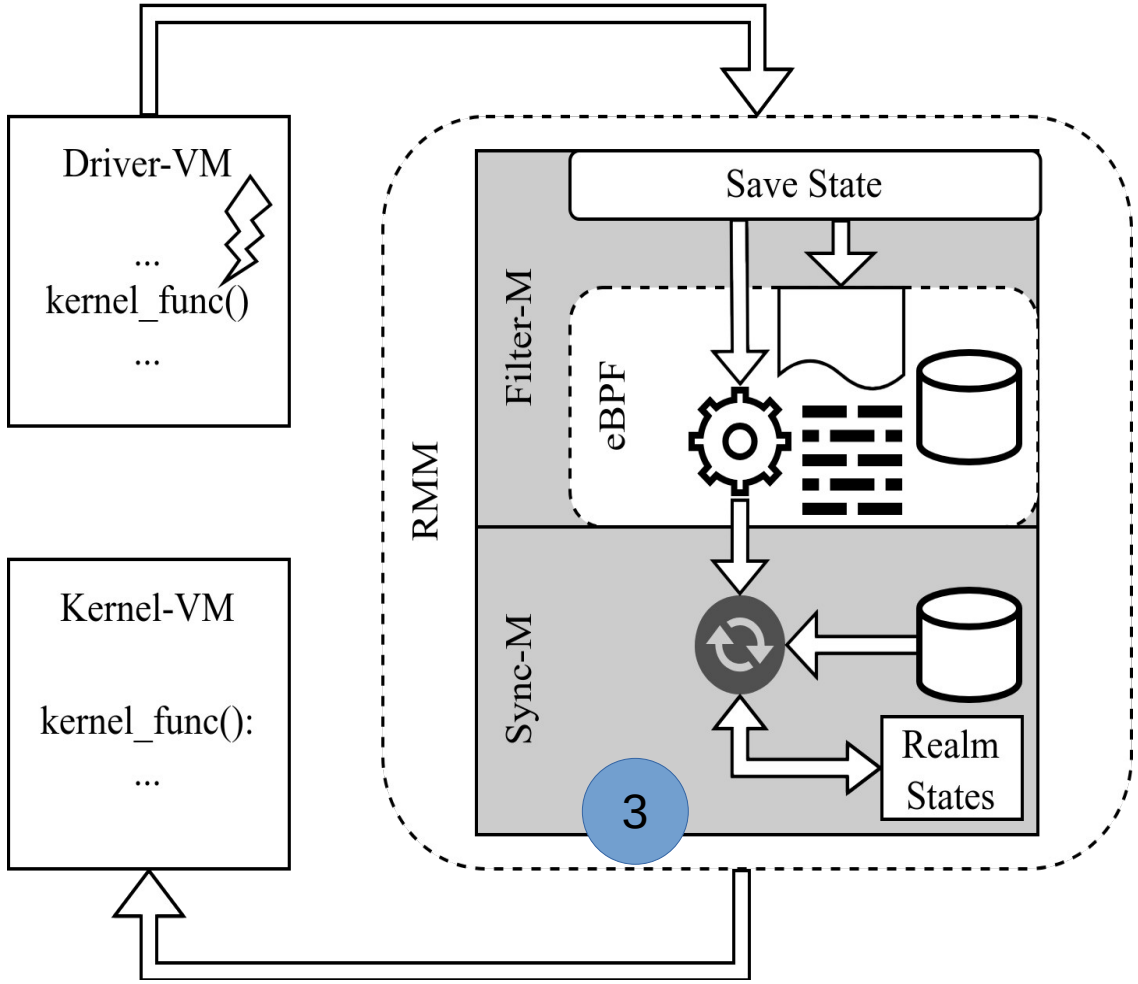
  => Not mapped

  => Trap to RMM
- Forward state, filter rules, config to the eBPF program



BarriCCAde – Runtime

- Extract information for accessed resource from sync config
- Synchronize resources according to config to kernel state



BarriCCAde – Status and Outlook

- Work in Progress
- eBPF Filter fully functional
- Basic Sychronization implemented
- Prototype can load and execute dummy modules
- So far 2195 LOC added to TCB (1722 formally verified eBPF Interpreter, 473 „untrusted“)
- Evaluation on real drivers TBD

BarriCCAde – Status and Outlook

- Work in Progress
- eBPF Filter fully functional
- Basic Sychronization implemented
- Prototype can load and execute dummy modules
- So far 2195 LOC added to TCB (1722 formally verified eBPF Interpreter, 473 „untrusted“)
- Evaluation on real drivers TBD

- Improve (automatic) generation of Sync Configs
- Improve (automatic) generation of Filter Rules

Summary

Thanks for your attention!

BarriCCAde: Isolating Closed-Source Drivers with ARM CCA

Matti Schulze
FAU Erlangen-Nürnberg
matti.schulze@fau.de

Christian Lindenmeier
FAU Erlangen-Nürnberg
christian.lindenmeier@fau.de

Jonas Röckl
FAU Erlangen-Nürnberg
jonas.roeckl@fau.de

- BarriCCAde is a step forward to isolating vulnerable drivers
- Introduces novel resource synchronization technique based on debug information and trap based access forwarding to target closed source drivers
- Allows detection and blocking of malicious behavior via eBPF-based filters
- Minimal TCB increase by relying on novel Confidential Computing Architectures