Cache Storage Channels **Alias-driven Attacks**

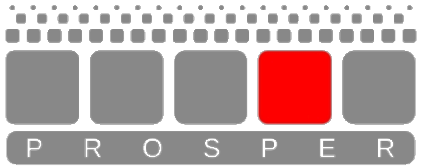
Roberto Guanciale
Mads Dam
Hamed Nemati
Christoph Baumann

IEEE S&P, 2016

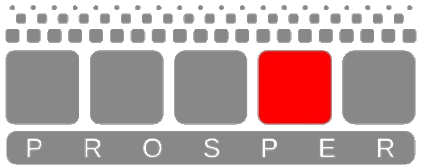
Presented by:
Paul Murley
21 September, 2017

This is a modified version of the original slide deck presented by the authors.

Formally Verified Platforms



Formally Verified Platforms



Caches Excluded from the analysis

Formally Verified Platforms

seL4
Security. Performance. Proof.

Microsoft
Hyper-V

INTEGRITY

PROSPER

Caches Excluded from the analysis

Formally Verified Platforms

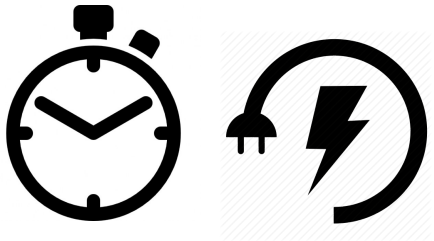
seL4
Security. Performance. Proof.

Microsoft Hyper-V

INTEGRITY

PROSPER

Caches Excluded from the analysis



Formally Verified Platforms

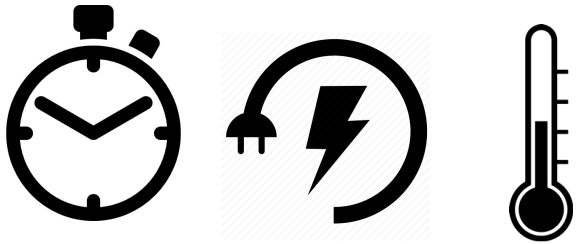
seL4
Security. Performance. Proof.

Microsoft Hyper-V

INTEGRITY

PROSPER

Caches Excluded from the analysis



Formally Verified Platforms

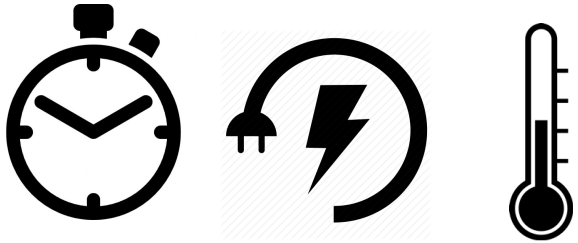
seL4
Security. Performance. Proof.

Microsoft Hyper-V

INTEGRITY

PROSPER

Caches Excluded from the analysis



Models should be Sound

Formally Verified Platforms

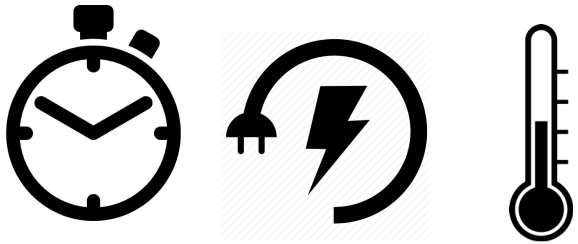
seL4
Security. Performance. Proof.

Microsoft Hyper-V

INTEGRITY

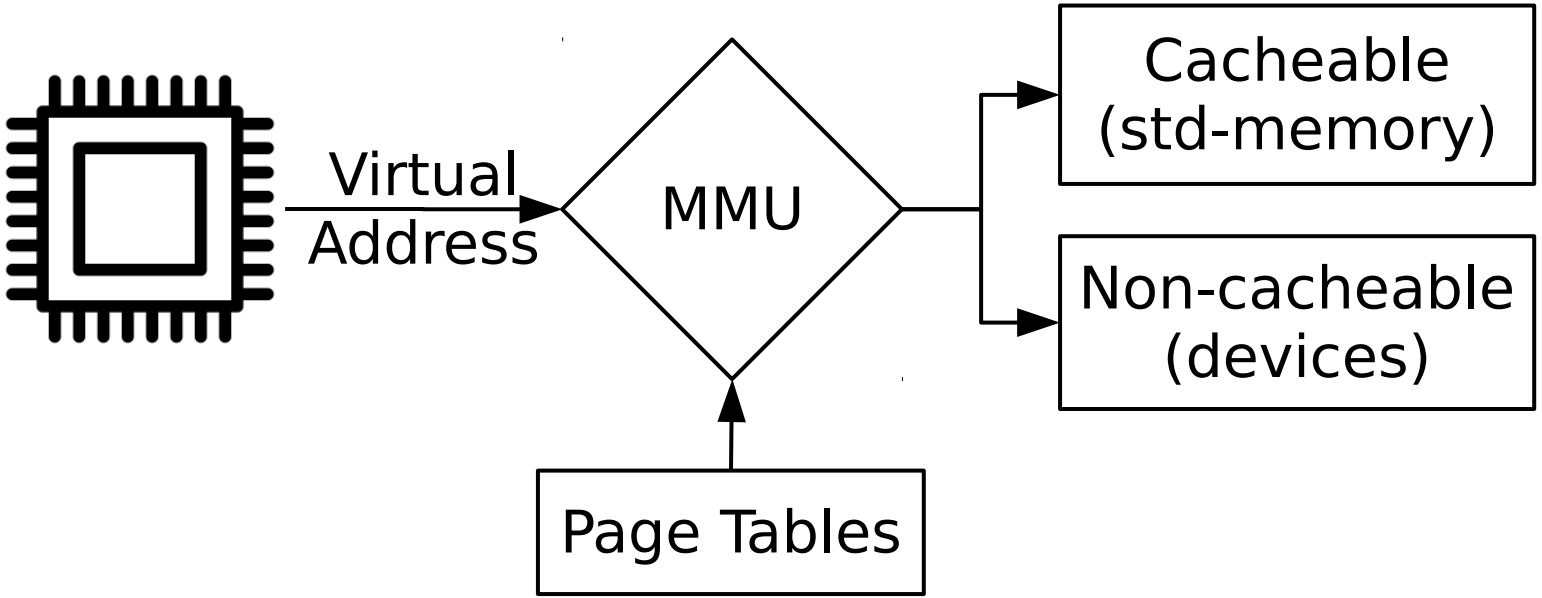
PROSPER

Caches Excluded from the analysis



Models should be Sound

Storage Channels can invalidate results



Incoherent Cache Behaviors

Mismatched cacheability attributes

Mismatched cacheability attributes

Please, do not do this

Mismatched cacheability attributes

Please, do not do this

Incoherent Cache Behaviors

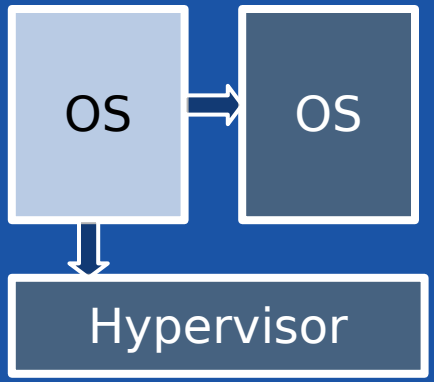
Mismatched cacheability attributes

Please, do not do this

Incoherent Cache Behaviors

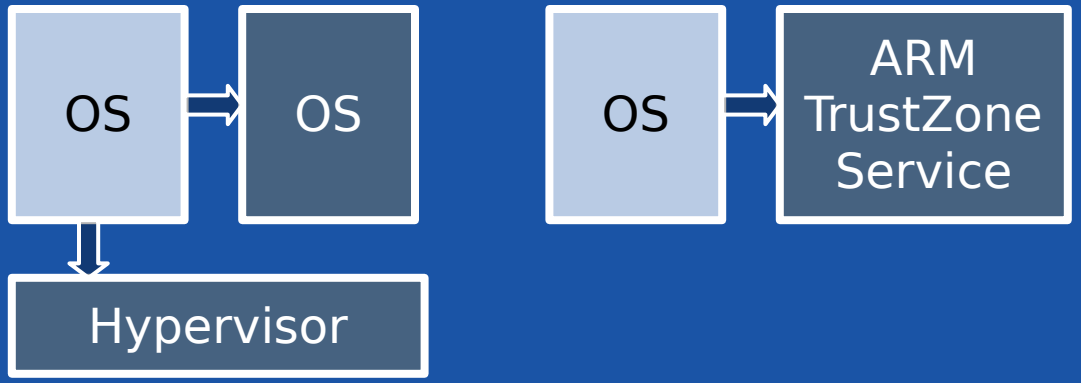
ARM-terminology: **unexpected cache hit**

if the data cache reports a hit on a memory location that is marked as non-cacheable, the cache might access the memory disregarding such hit.



Scenarios

Scenarios

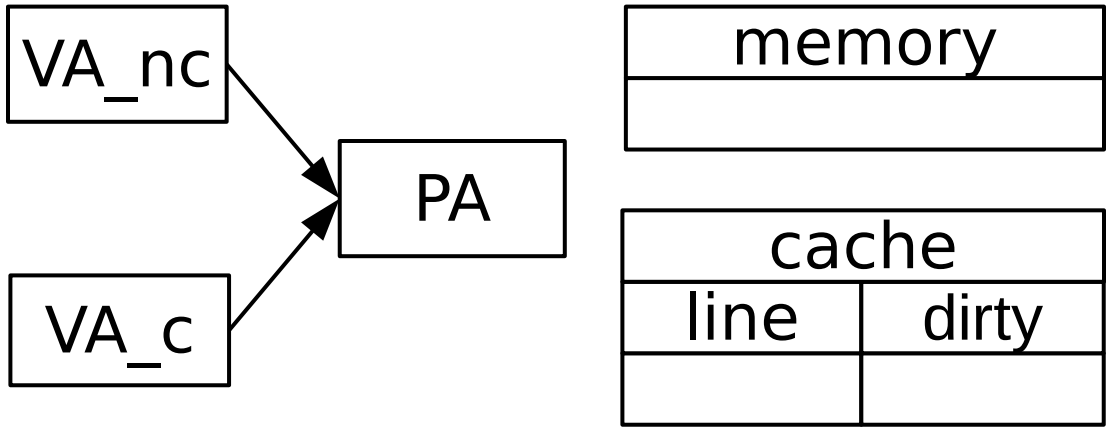


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

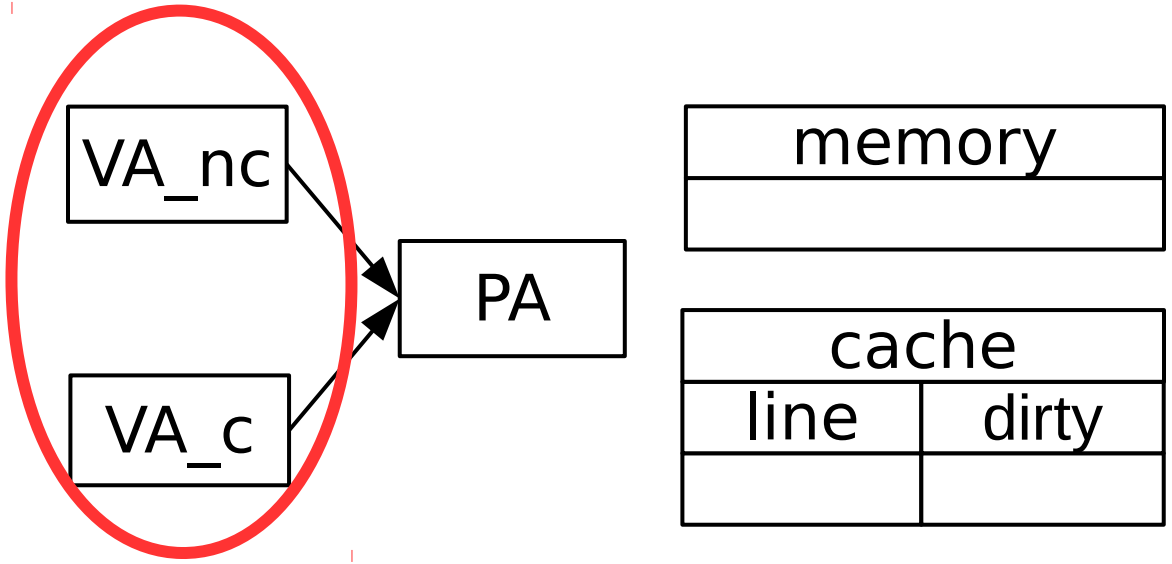


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

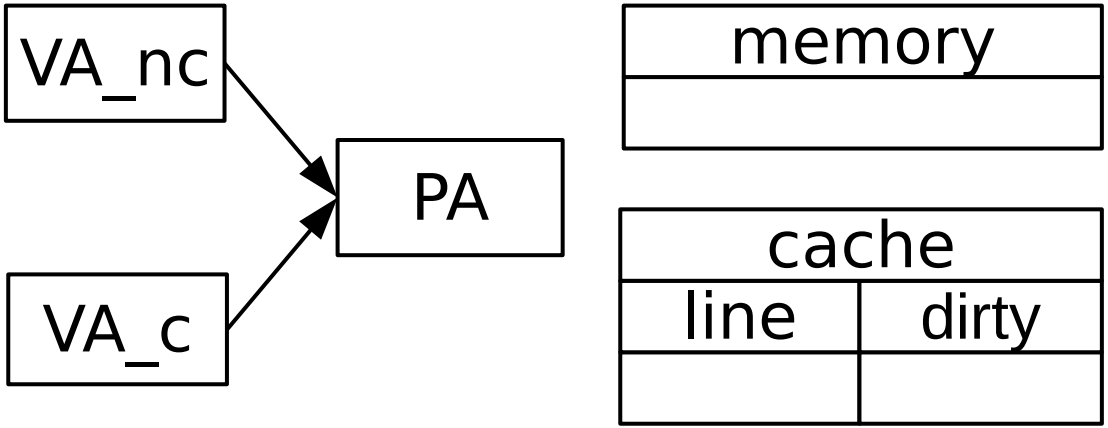


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

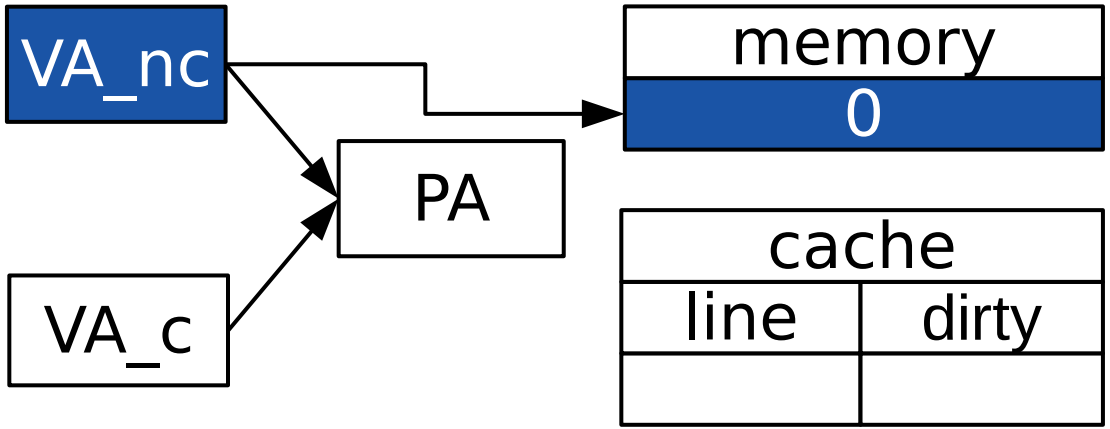


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

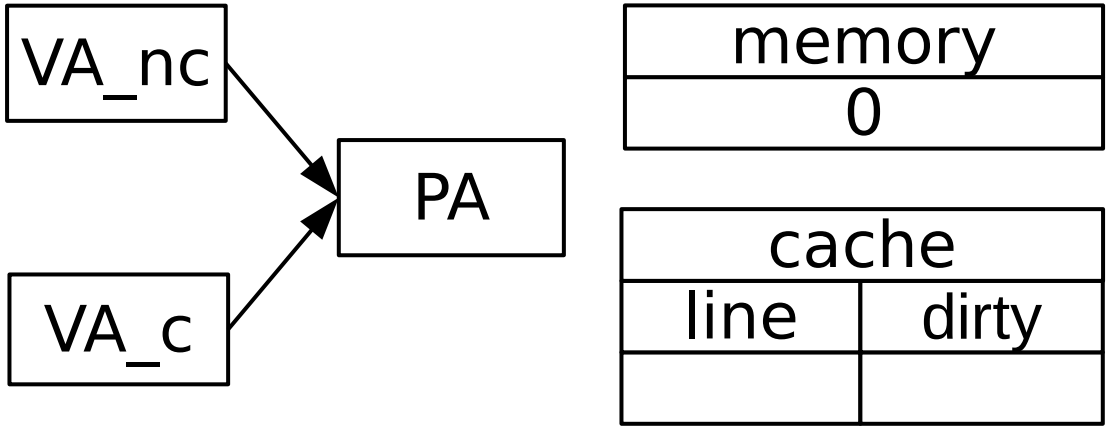


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

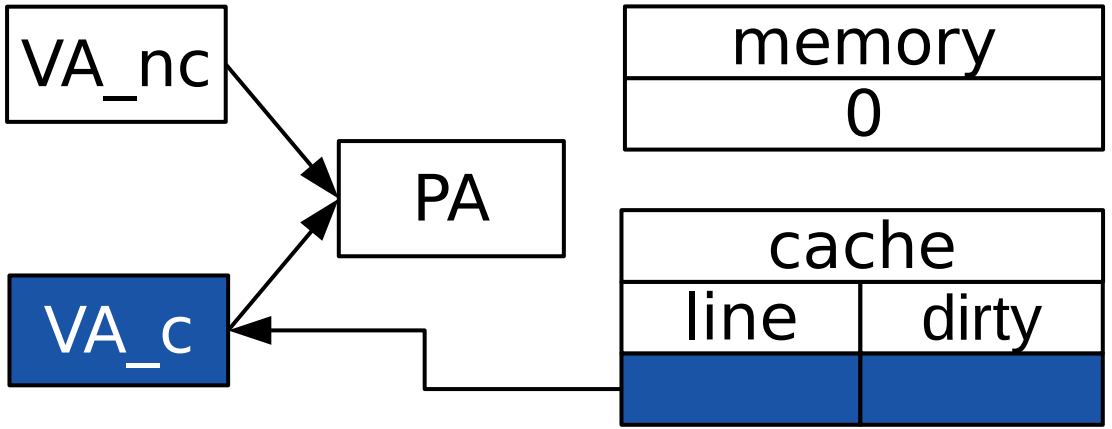


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

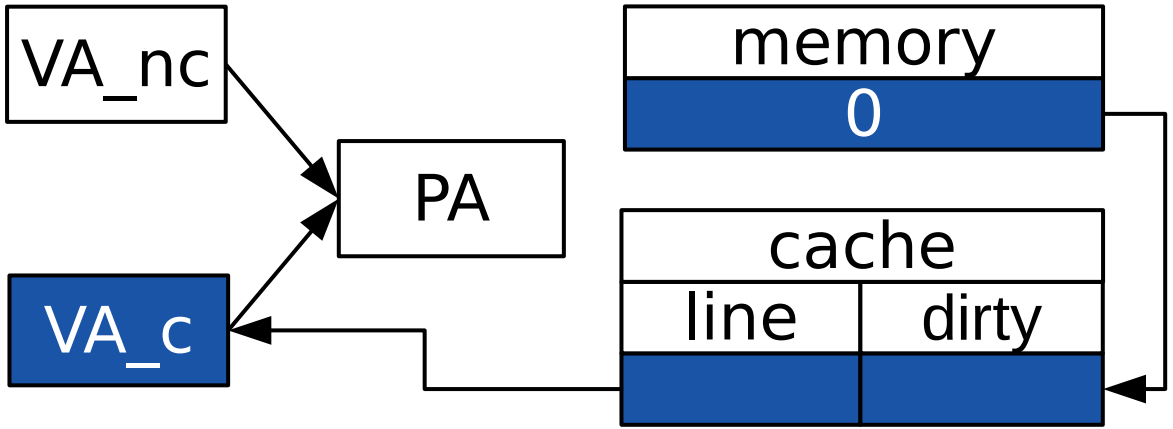


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

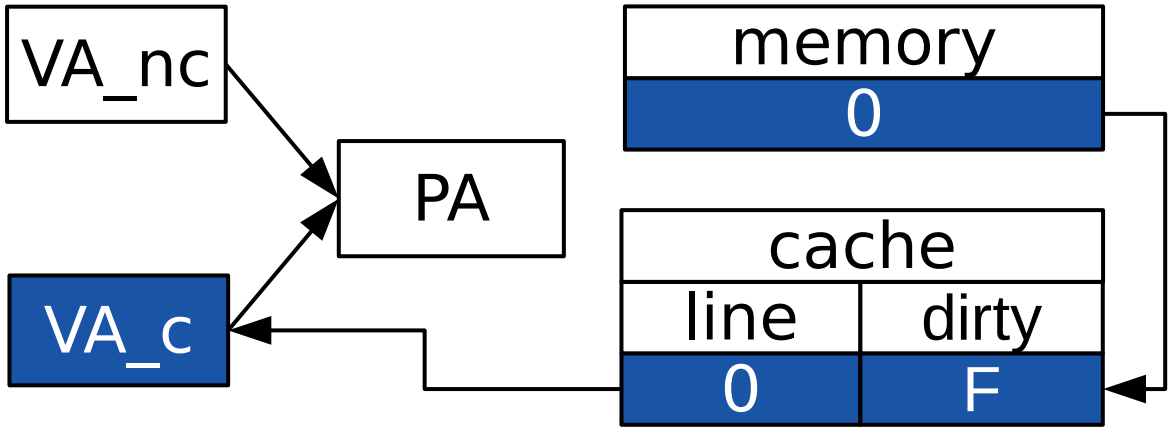


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

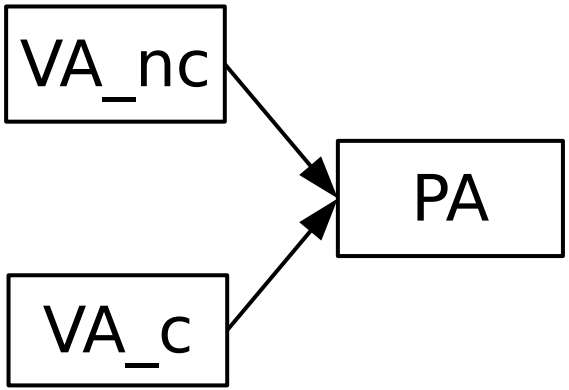


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```



memory
0

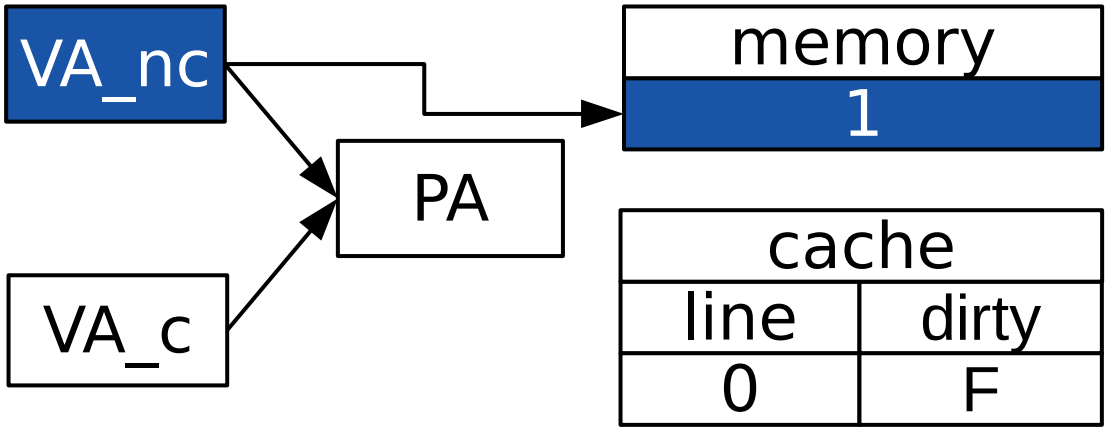
cache	
line	dirty
0	F

Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

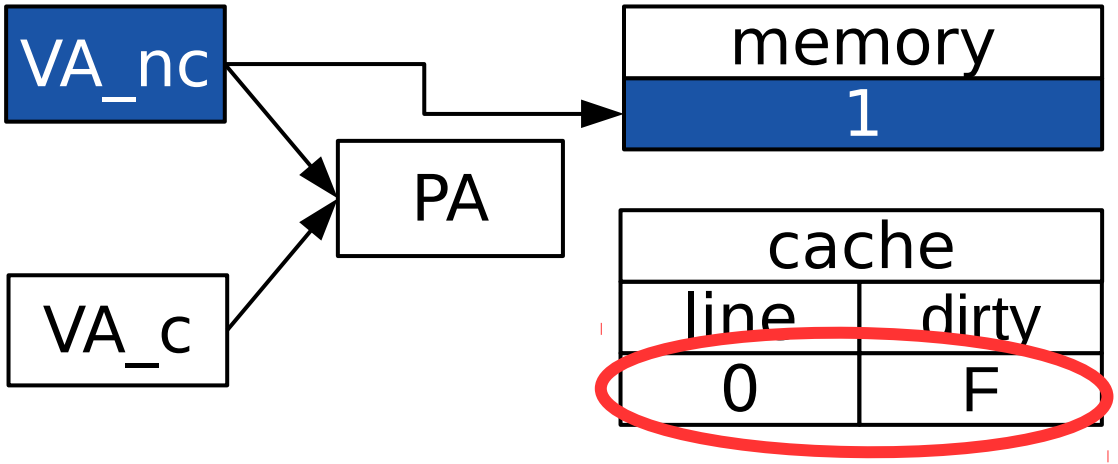


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

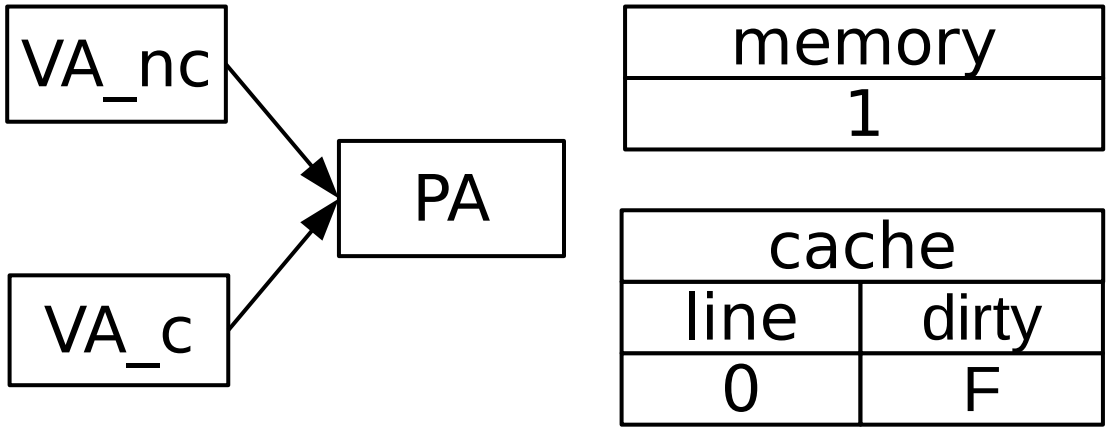


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

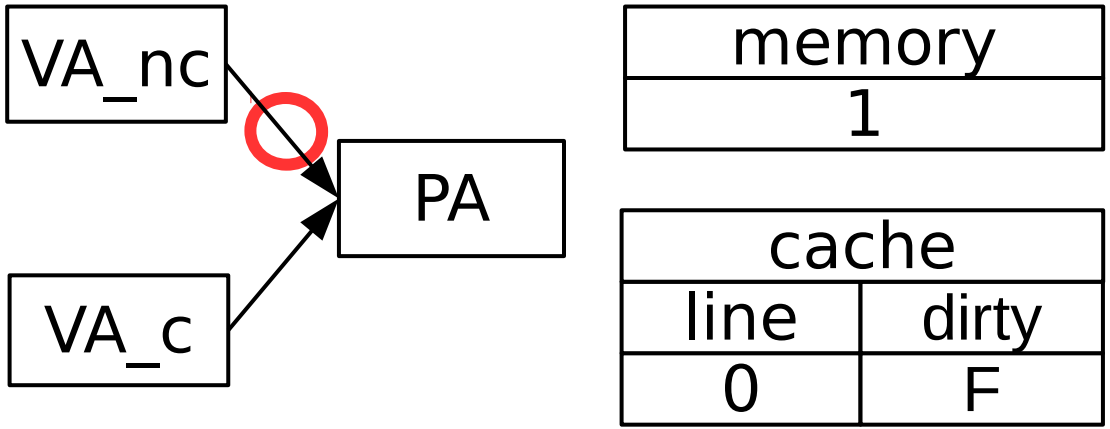


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

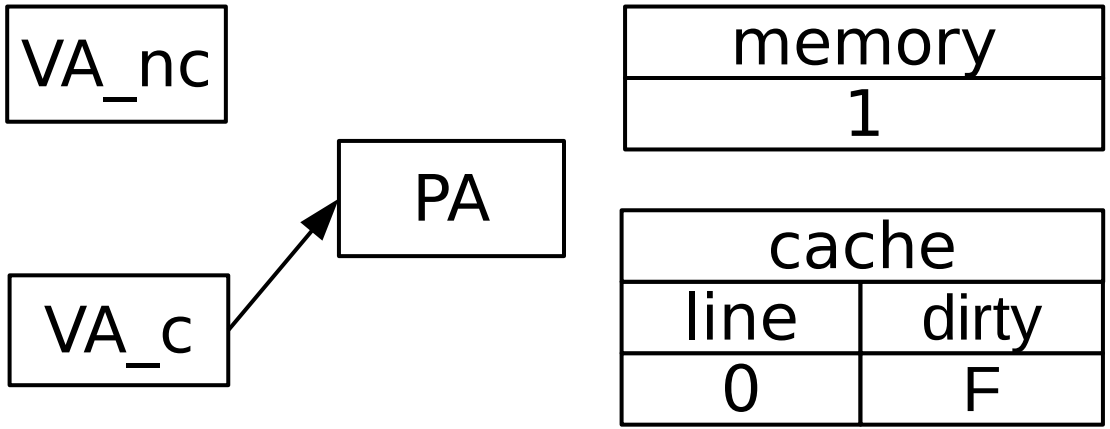


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

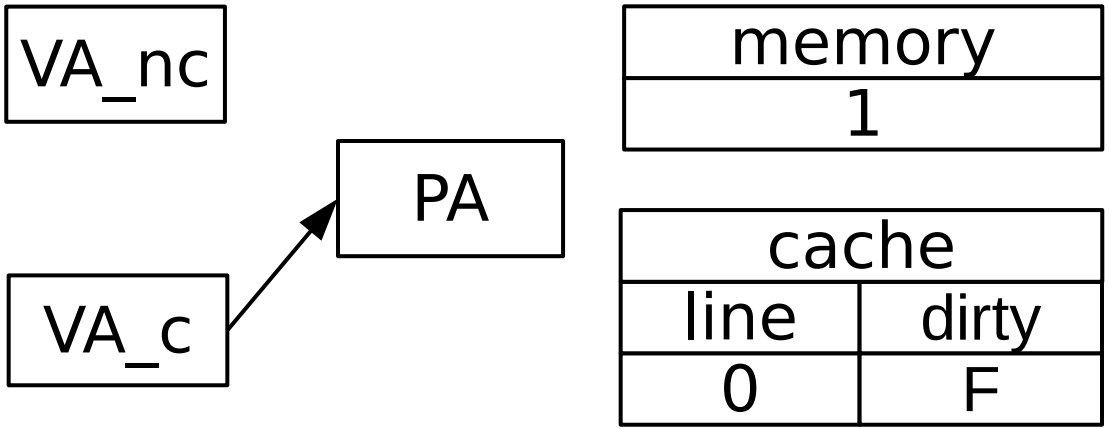


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

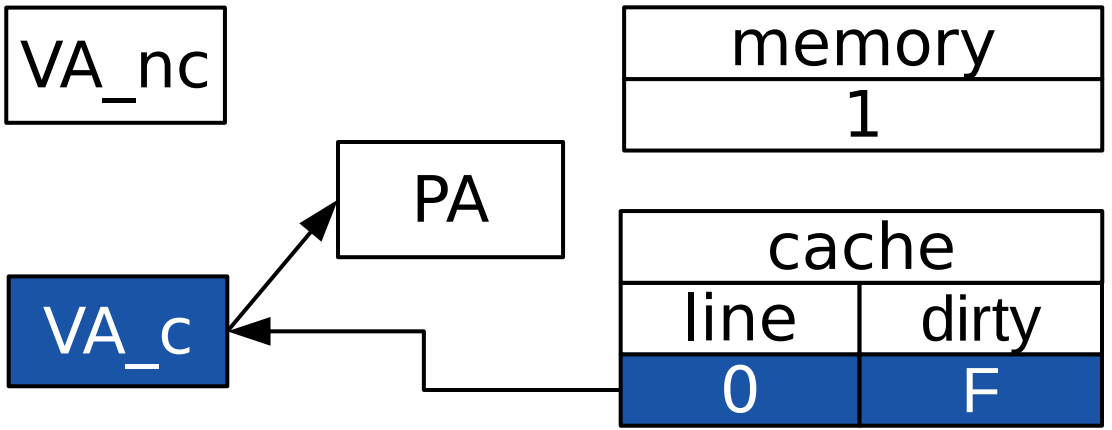


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

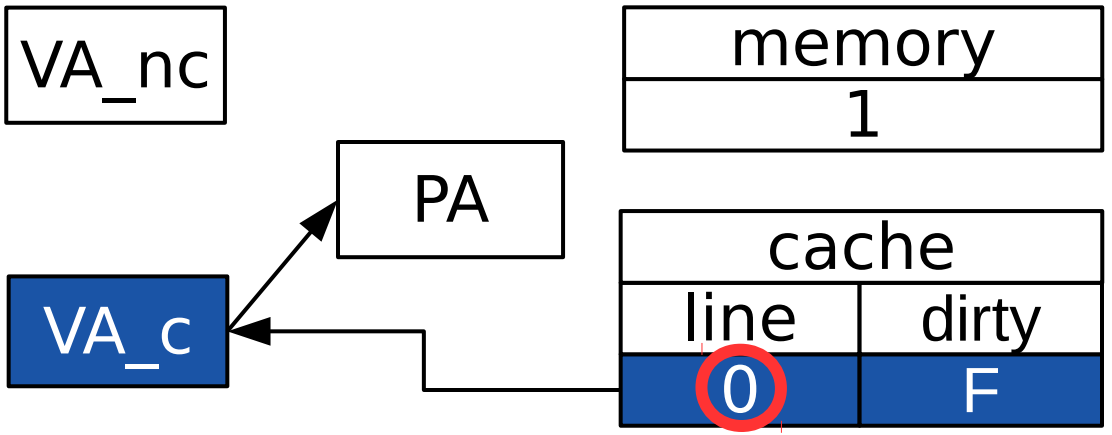


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

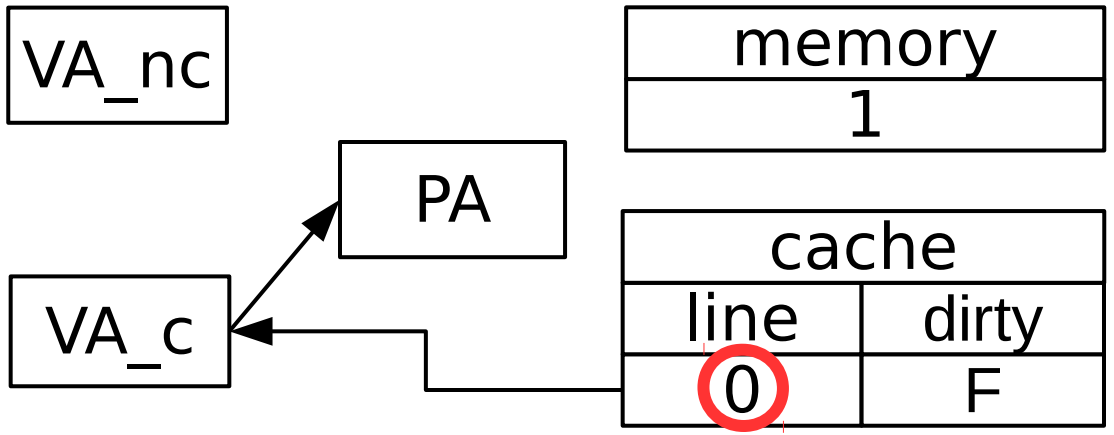


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

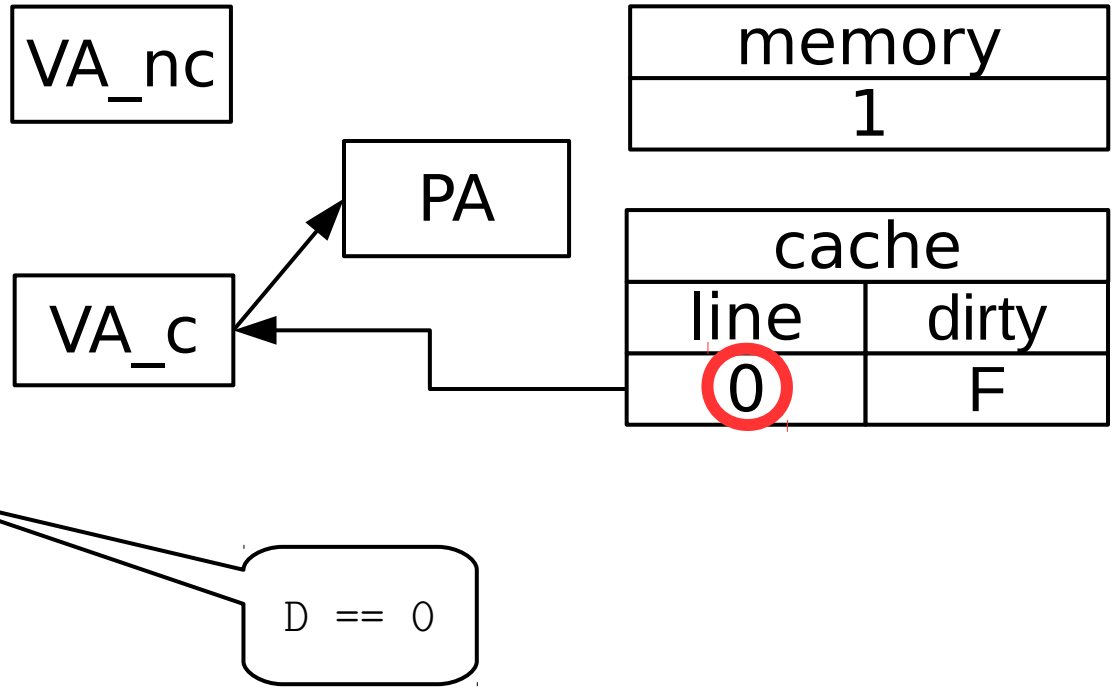


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

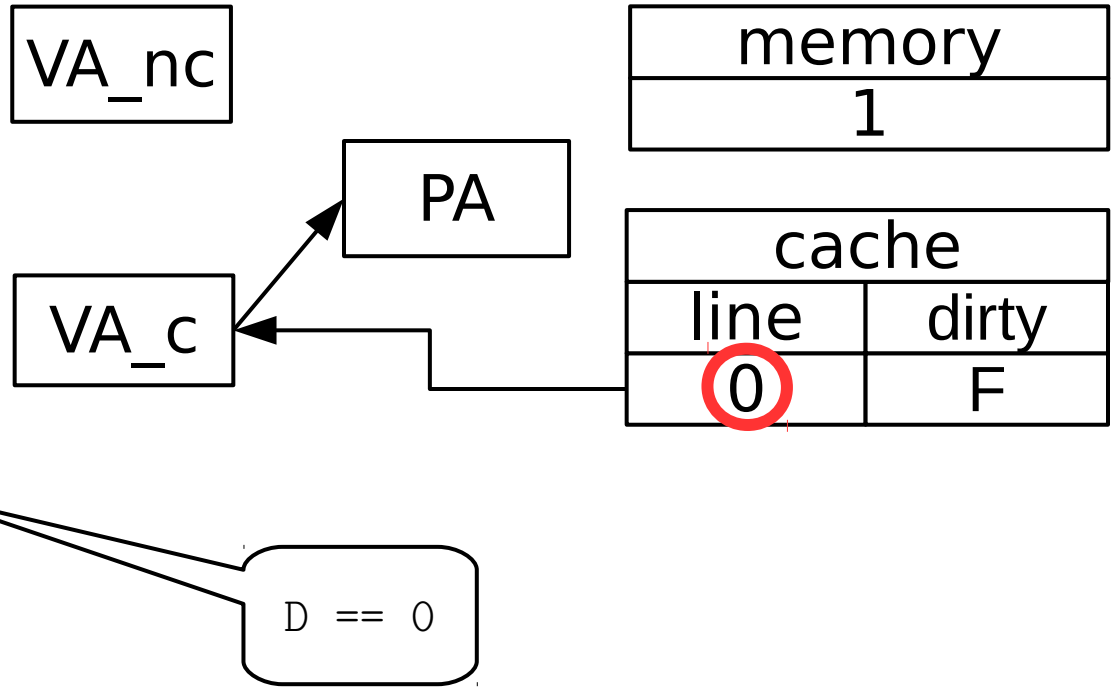


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

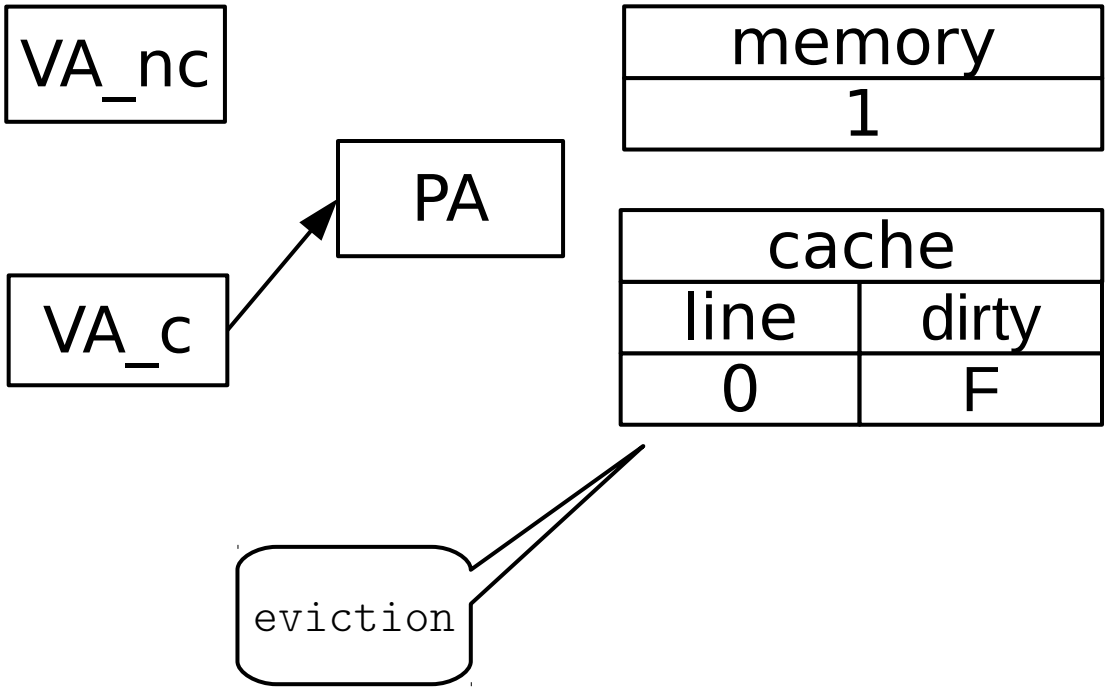


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

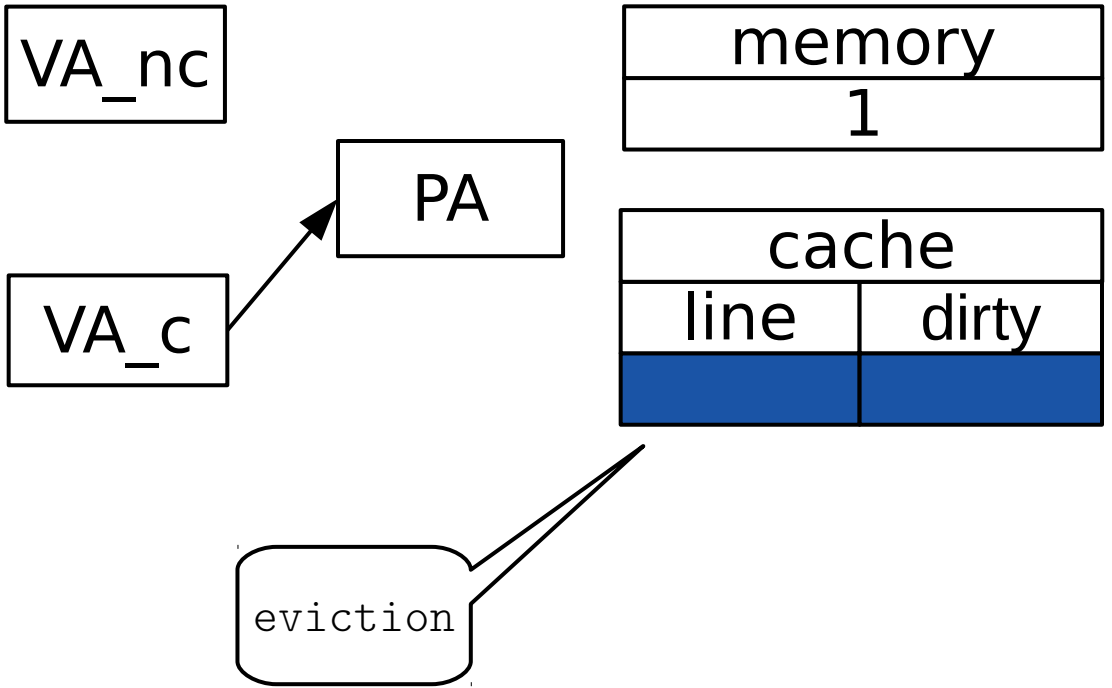


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

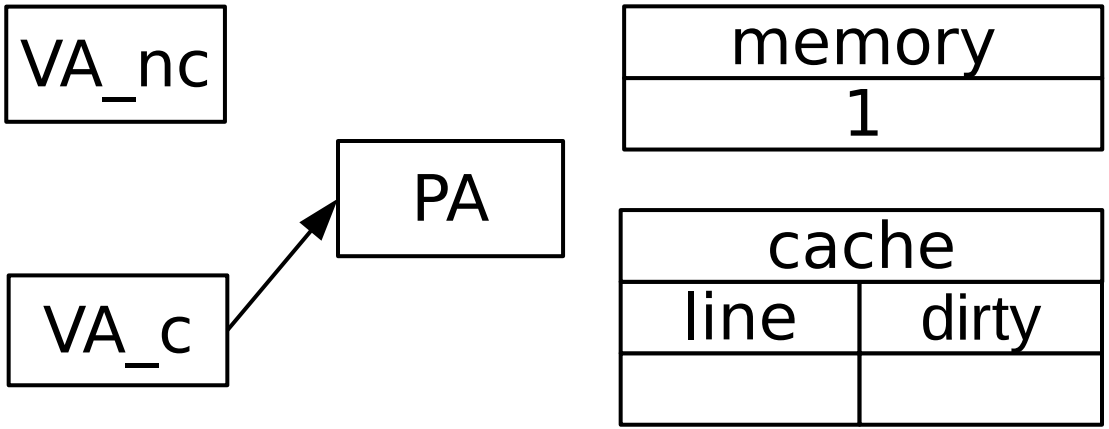


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

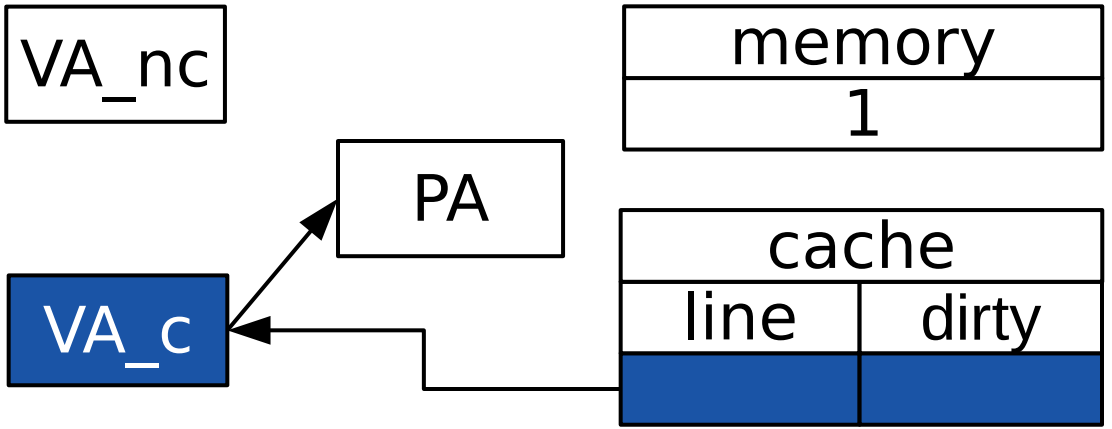


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

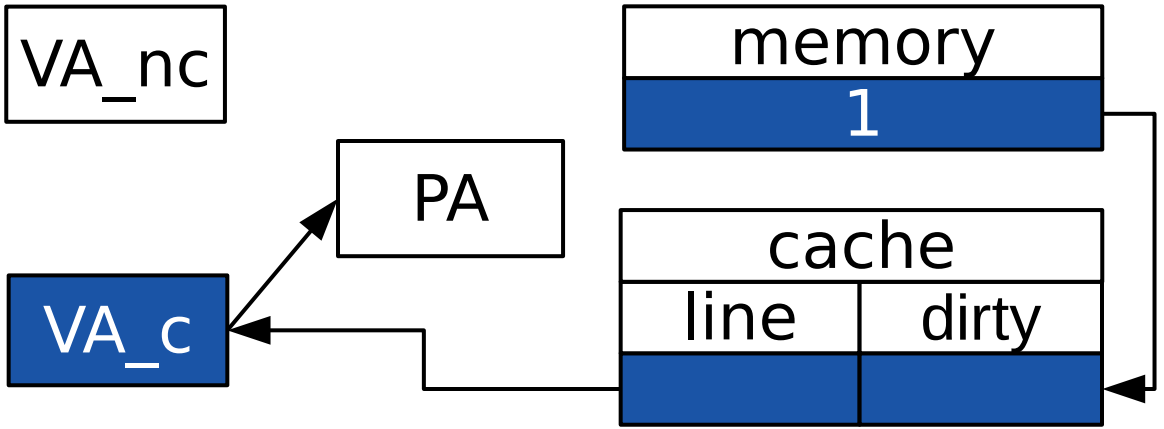


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

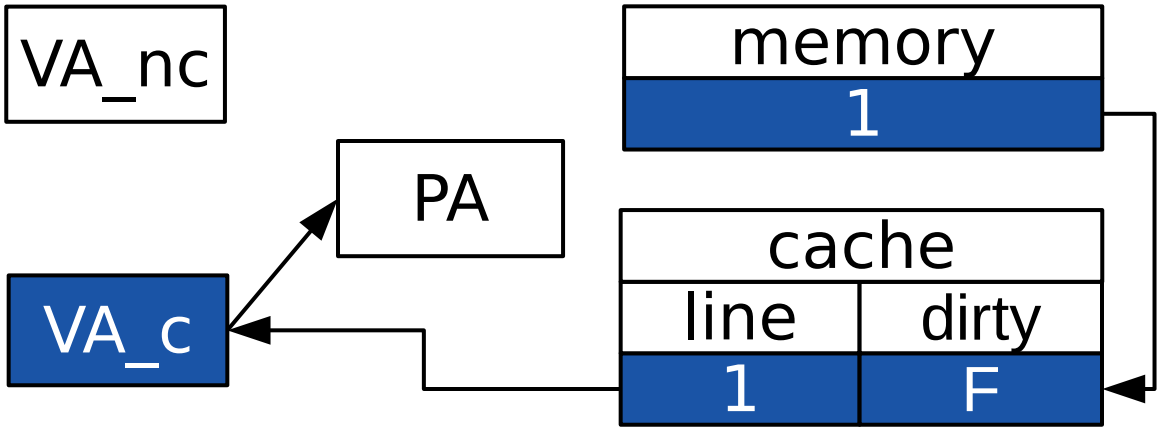


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```

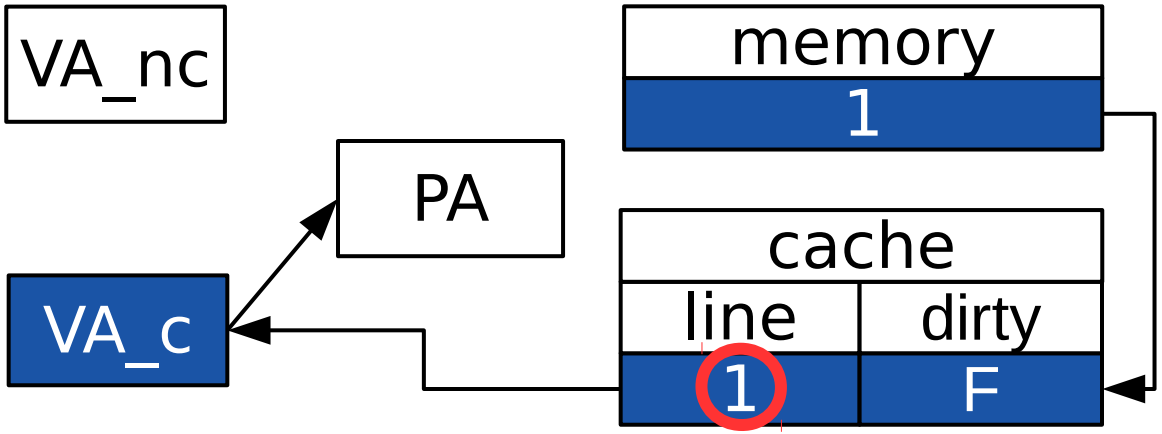


Attacker

```
write(VA_nc, 0)
…
write(VA_nc, 1)
free(VA_nc)
```

Victim

```
D = access(VA_c)
…
D = access(VA_c)
if not policy(D)
   Reject()
…
use(VA_c)
```



Storage channels

Integrity threat

- Transfer of memory ownership
- Time Of Check To Time Of Use attacks
- No need of
  - simultaneous double mapping
  - concurrency

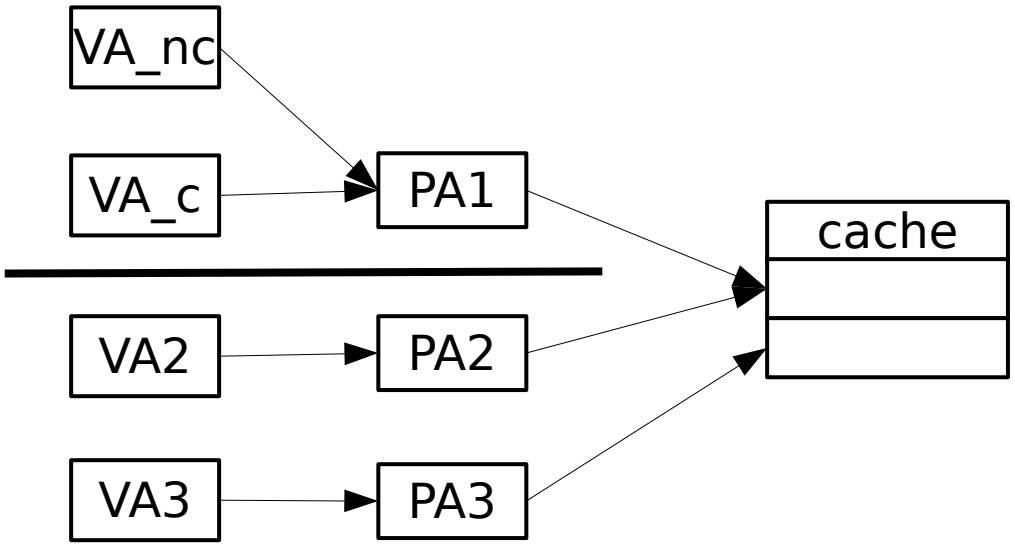
Natural preys: reference monitors

Attacker

```
invalidate(VA_c)
write(VA_nc, 0)
D = read(VA_c)
write(VA_nc, 1)
call victim
D = read(VA_c)
```

Victim

```
if secret
   access(VA2)
else
   access(VA3)
```



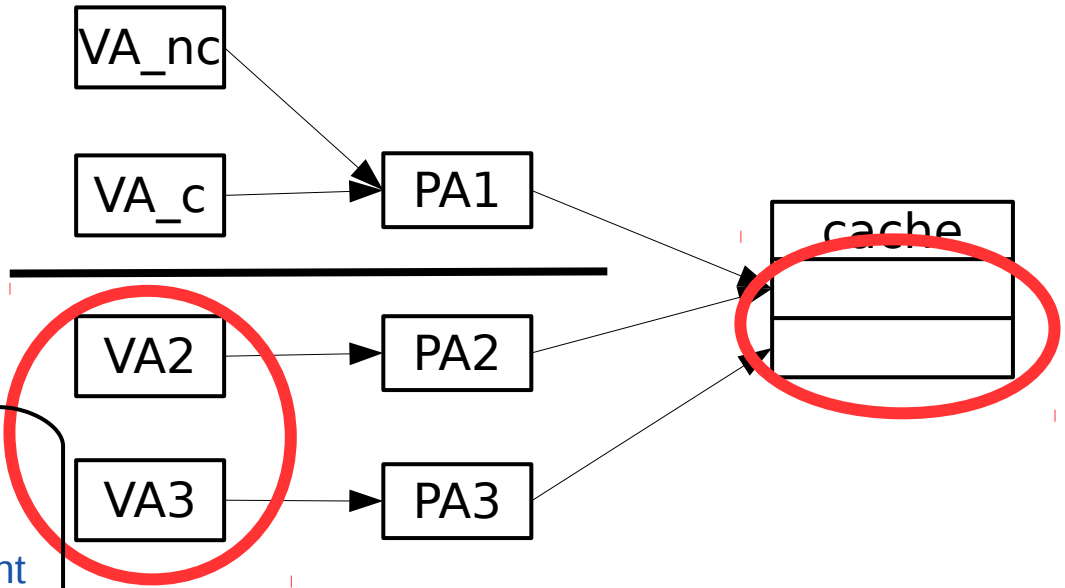
Attacker

```
invalidate(VA_c)
write(VA_nc, 0)
D = read(VA_c)
write(VA_nc, 1)
call victim
D = read(VA_c)
```

Victim

```
if secret
   access(VA2)
else
   access(VA3)
```

Accessed cache line is secret-dependent

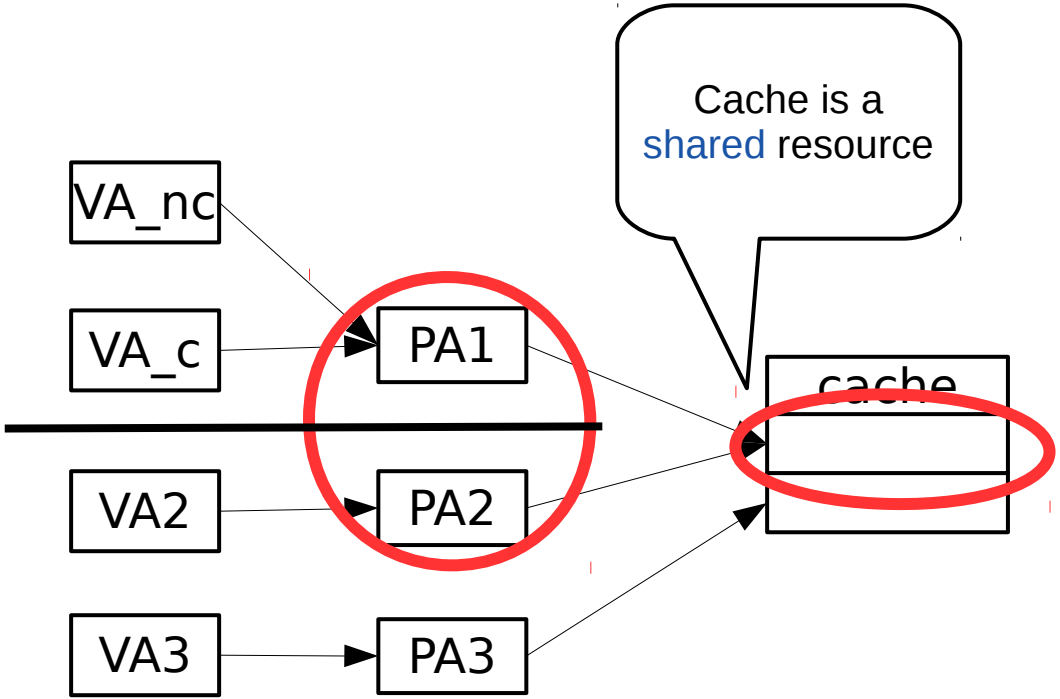


Attacker

```
invalidate(VA_c)
write(VA_nc, 0)
D = read(VA_c)
write(VA_nc, 1)
call victim
D = read(VA_c)
```

Victim

```
if secret
   access(VA2)
else
   access(VA3)
```

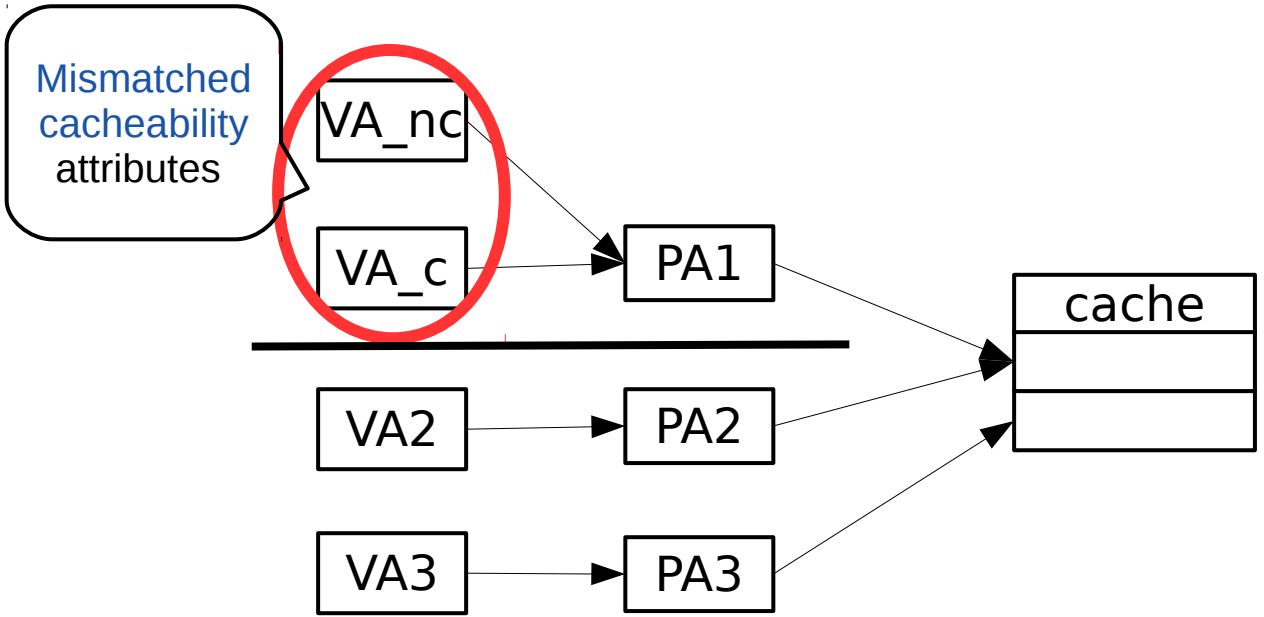


Attacker

```
invalidate(VA_c)
write(VA_nc, 0)
D = read(VA_c)
write(VA_nc, 1)
call victim
D = read(VA_c)
```

Victim

```
if secret
   access(VA2)
else
   access(VA3)
```



Storage channels

Integrity threat

- Transfer of memory ownership
- Time Of Check To Time Of Use
- No need of
  - simultaneous double mapping
  - concurrency

Natural preys: reference monitors

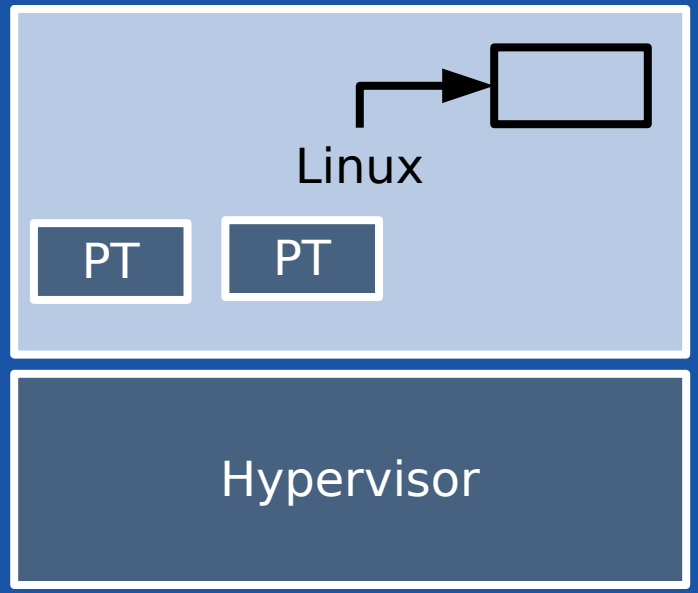
Confidentiality threat

- Access driven attacks
- No external measure needed
- Difficult to counter-measure at probing time

Natural preys: look-up tables

Paravirtualizing Hypervisor

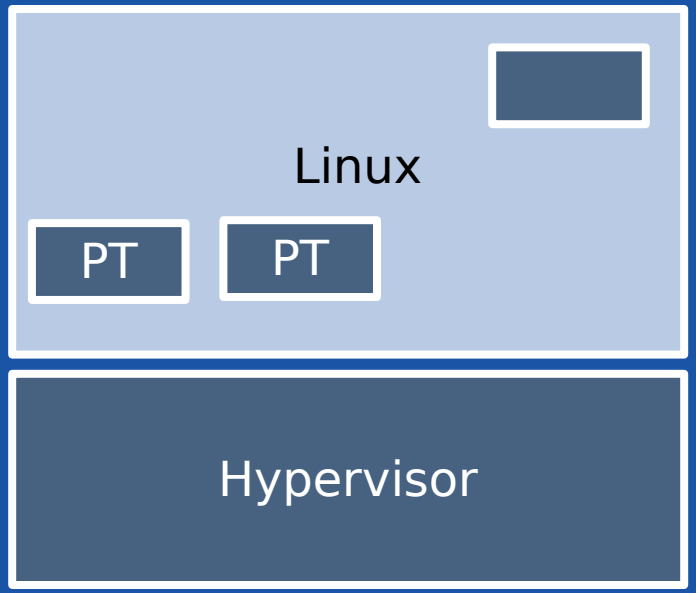
- Beagleboard



Linux prepares a PT

Paravirtualizing Hypervisor

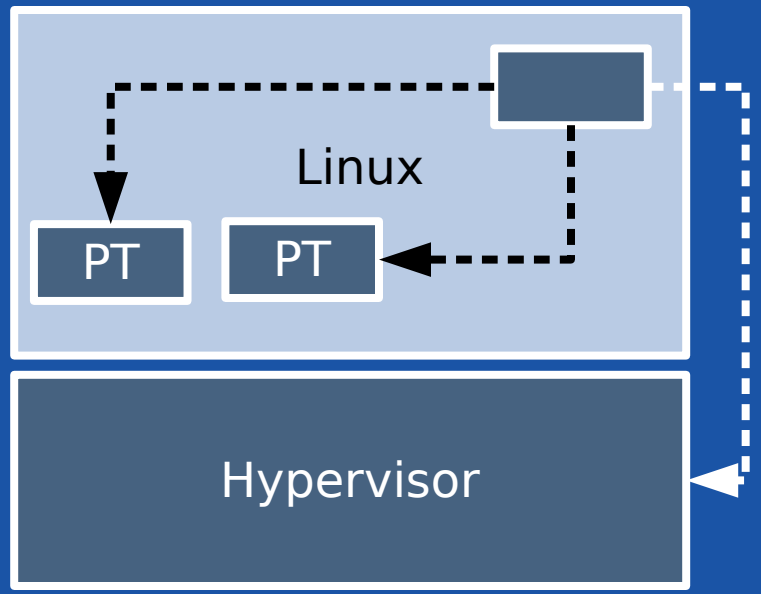
- Beagleboard



Hypervisor makes region read-only

Paravirtualizing Hypervisor

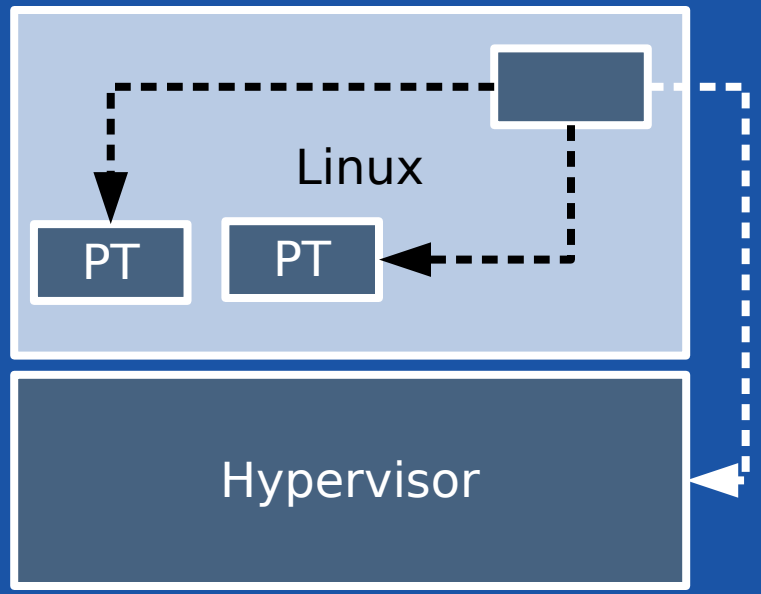
- Beagleboard



Hypervisor validates content

Paravirtualizing Hypervisor

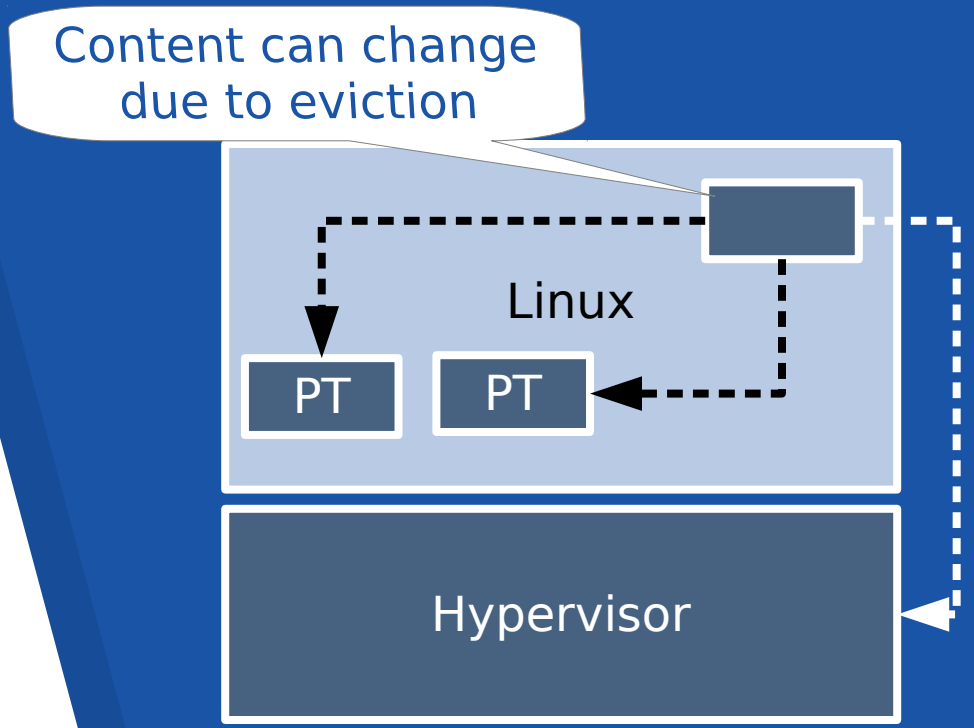
- Beagleboard



Hypervisor activates PT

Paravirtualizing Hypervisor

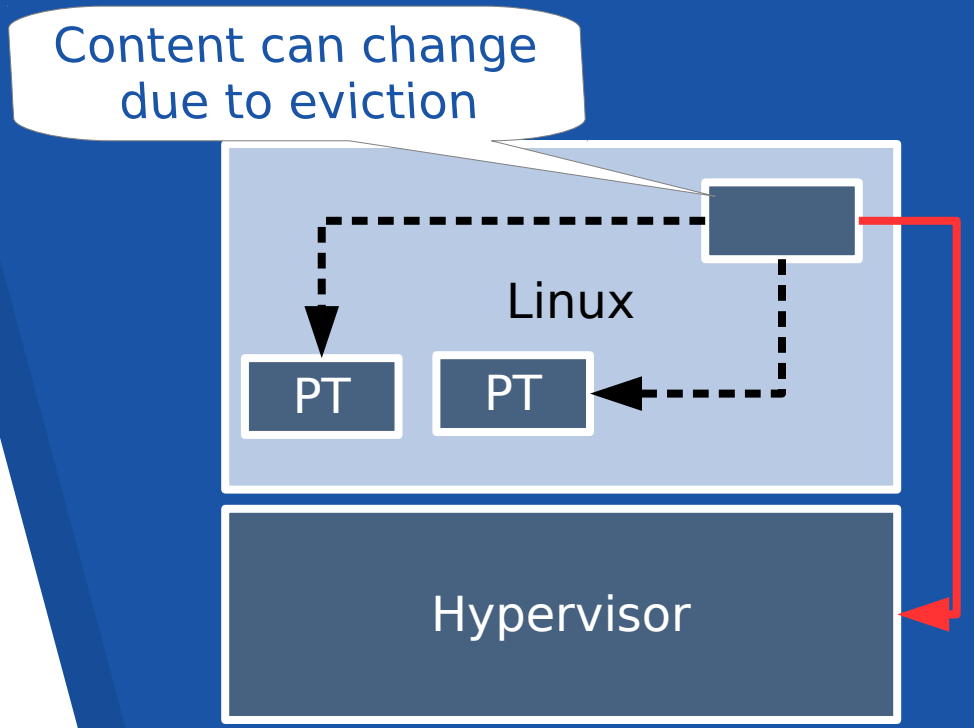
- Beagleboard



Hypervisor activates PT

Paravirtualizing Hypervisor

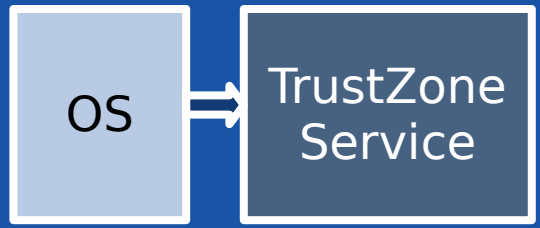
- Beagleboard
- Linux takes complete control



Hypervisor activates PT

AES Cryptoservice

- Raspberry PI 2
- 128-bit key extracted after 850 encryptions



Vulnerability:

c[j]=Kn[j] xor T4[s[j]]

Observable

Inferred

Countermeasures

Integrity threat
guarantee memory coherency

- cache flushes
  (8x overhead)
- selective eviction
  (0.2x overhead)

Countermeasures

Integrity threat
guarantee memory coherency

- cache flushes
  (8x overhead)
- selective eviction
  (0.2x overhead)

Confidentiality threat
standard timing approaches

- secret-independent accesses
  (5x overhead)
- no cache for secret accesses
  (6x overhead)

Countermeasures

Integrity threat
guarantee memory coherency

- cache flushes
  (8x overhead)
- selective eviction
  (0.2x overhead)

Vector specific

- avoid uncacheable aliases
  (0.15x overhead)

Confidentiality threat
standard timing approaches

- secret-independent accesses
  (5x overhead)
- no cache for secret accesses
  (6x overhead)

Countermeasures

Integrity threat
guarantee memory coherency

- cache flushes
  (8x overhead)
- selective eviction
  (0.2x overhead)

Vector specific

- avoid uncacheable aliases
  (0.15x overhead)

Confidentiality threat
standard timing approaches

- secret-independent accesses
  (5x overhead)
- no cache for secret accesses
  (6x overhead)

HW Countermeasures

- do not disregard
  unexpected cache hit

Fixing Formal Verification

- Fix incoherent memory problem
- Verify countermeasures
- Property: Different memory/cache data for same VA means that cache line **must** be dirty

Questions