Reasoning About a Machine with Local Capabilities

Provably Safe Stack and Return Pointer Management

Lau Skorstengaard[1] Dominique Devriese[2] Lars Birkedal[1]

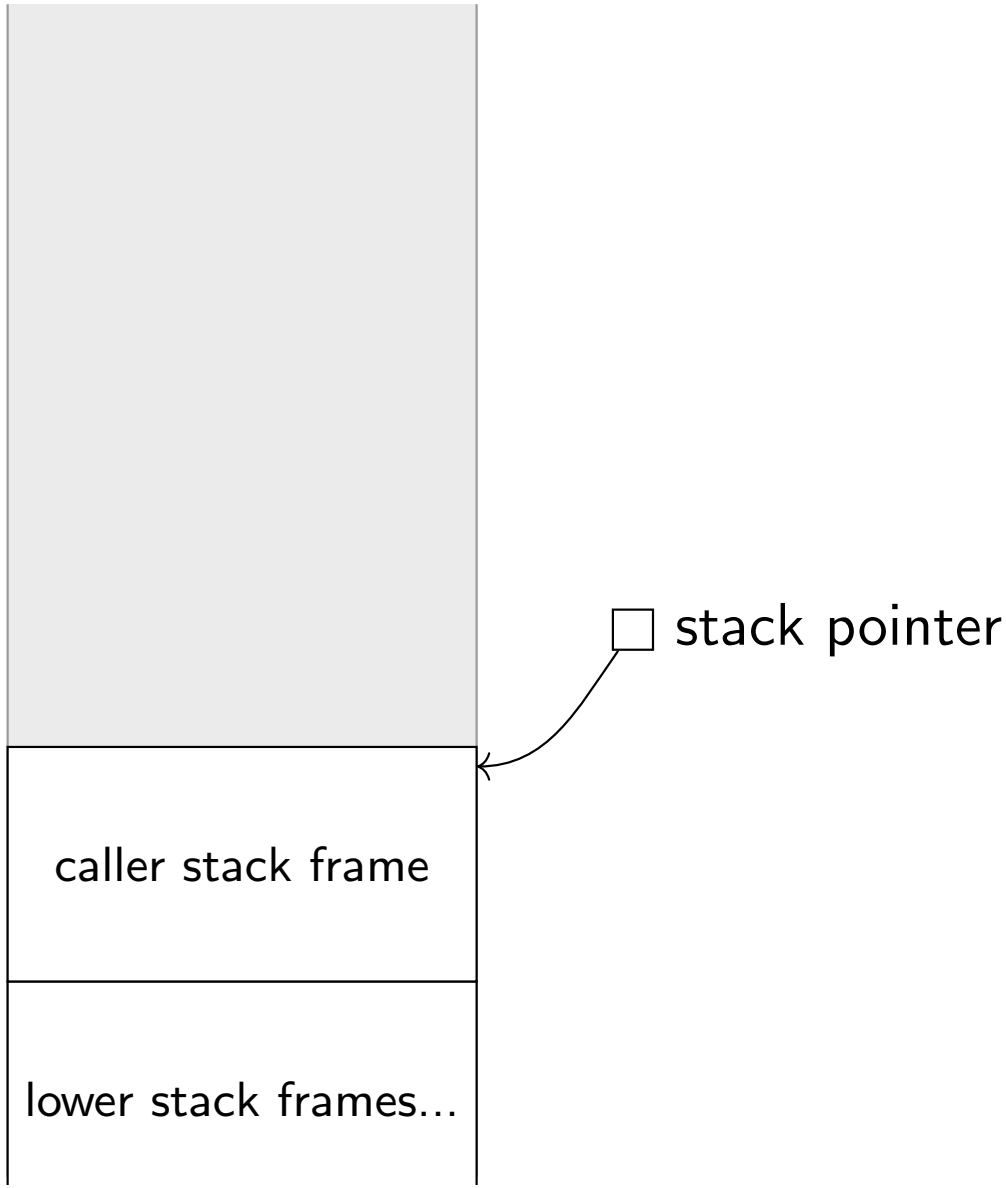
[1]Aarhus University

[2]imec-DistriNet, KU Leuven

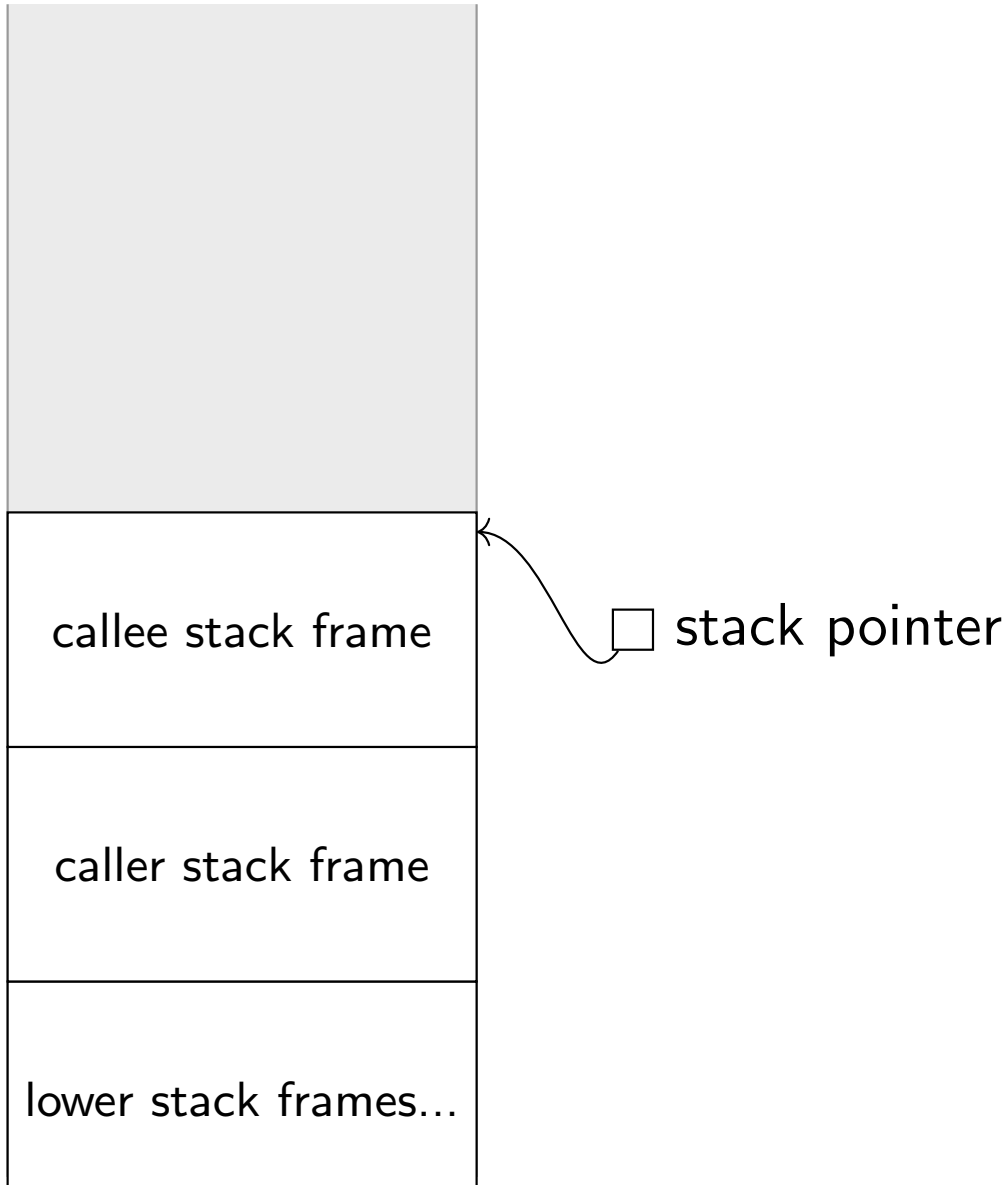
ESOP, April 17, 2018

```
let x = ref 0 in
        λf.(x := 0; f() x := 1; f(); !x)
```

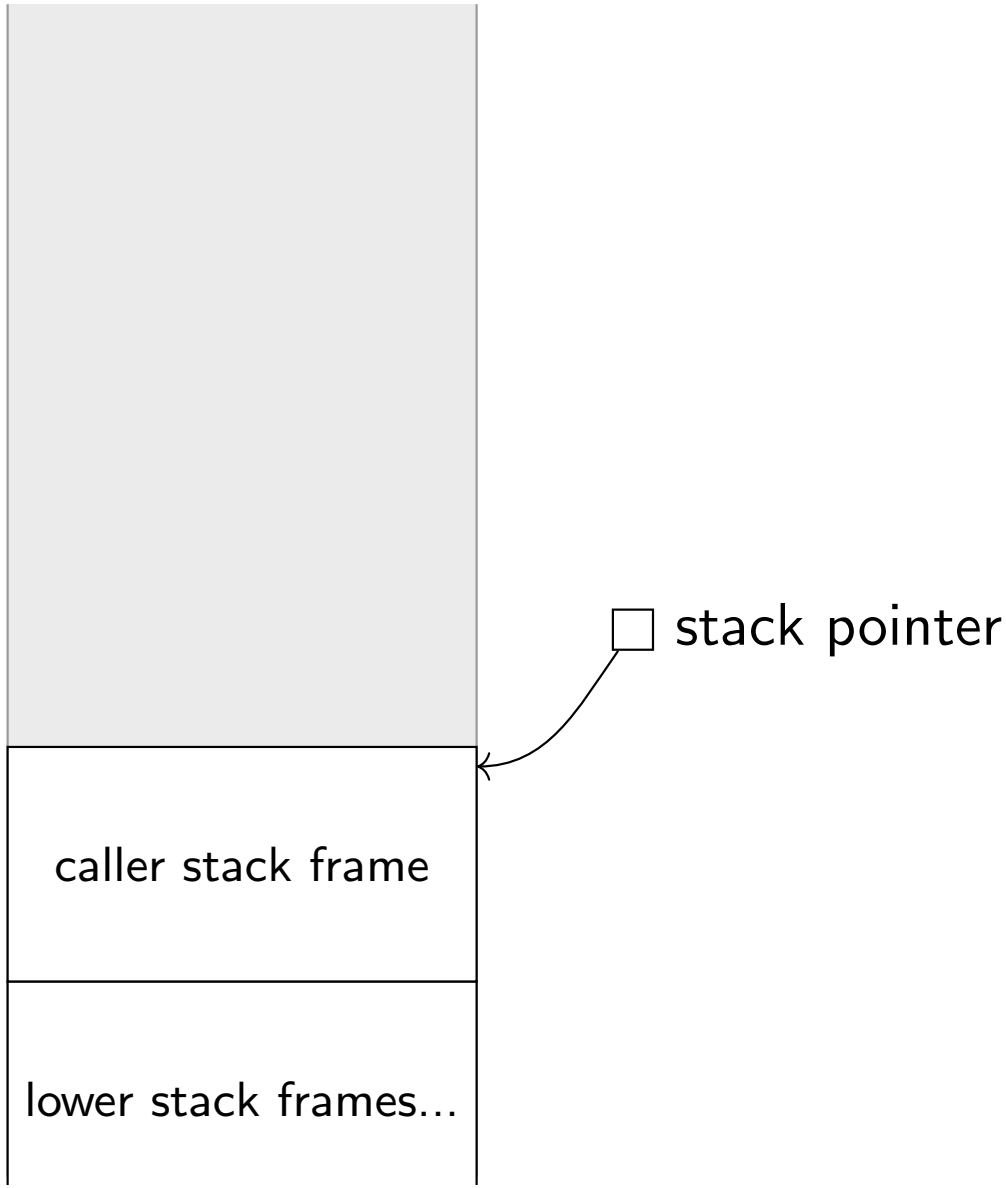
Traditional Stack Pointers



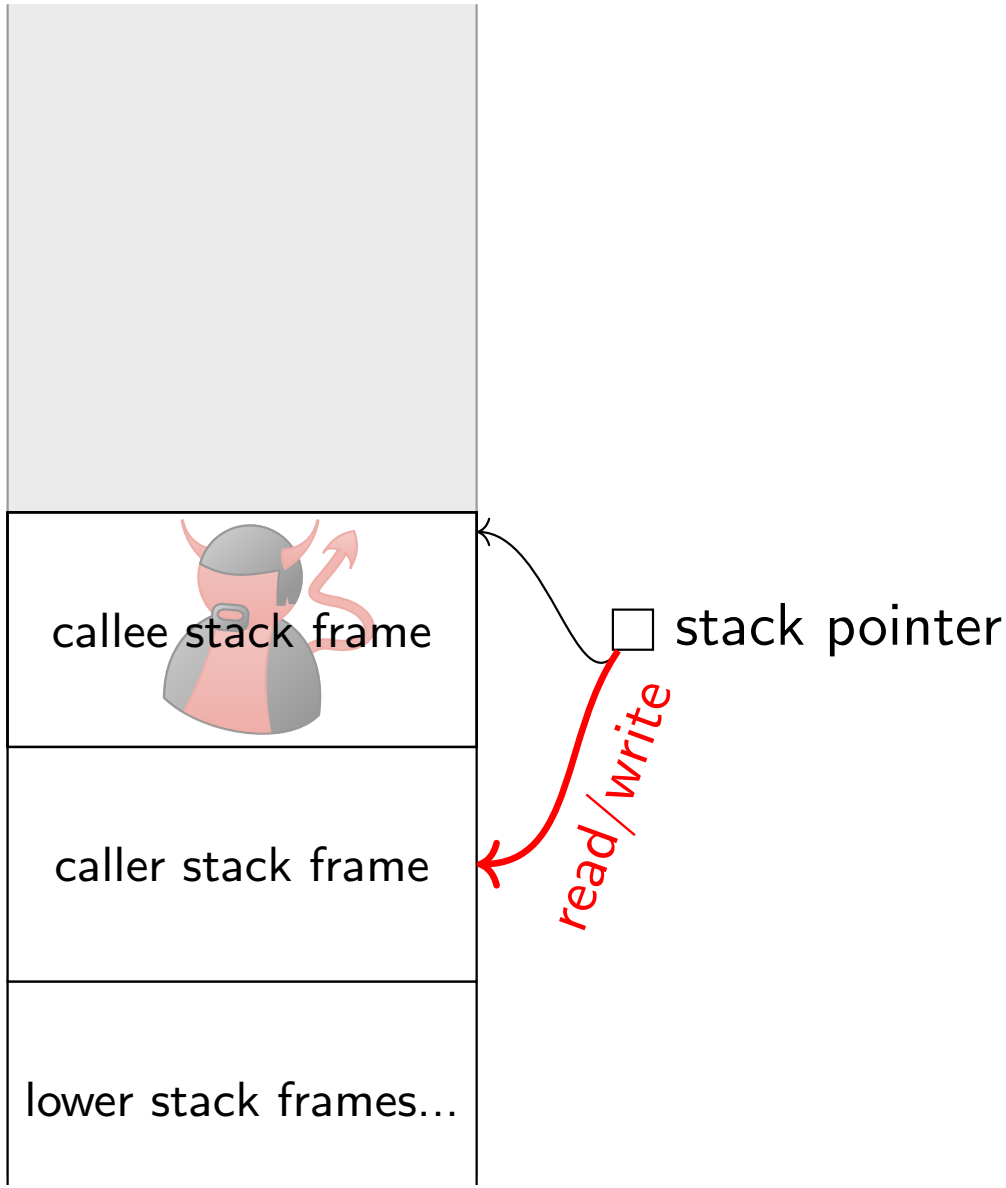
Traditional Stack Pointers



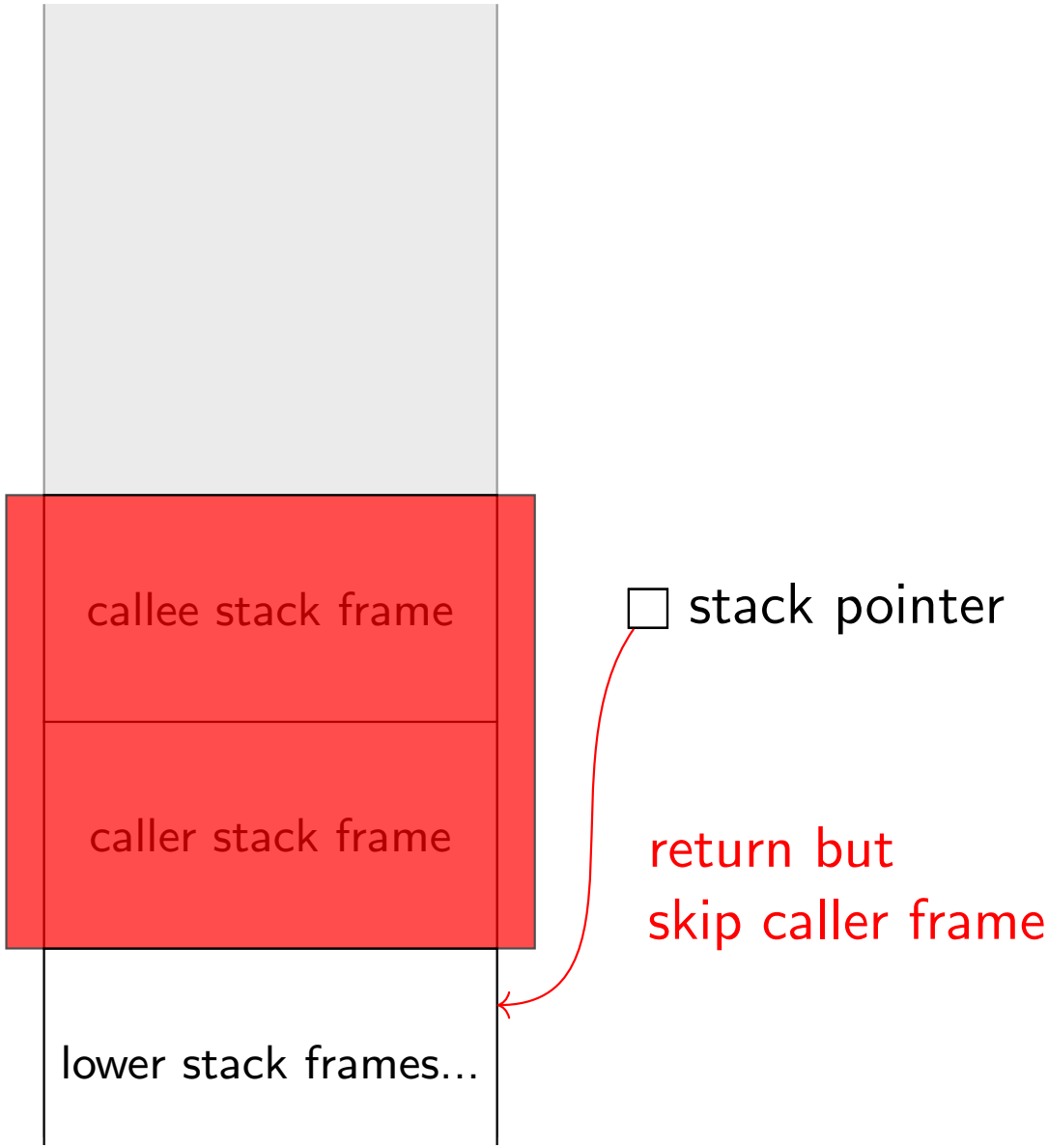
Traditional Stack Pointers



Traditional Stack Pointers



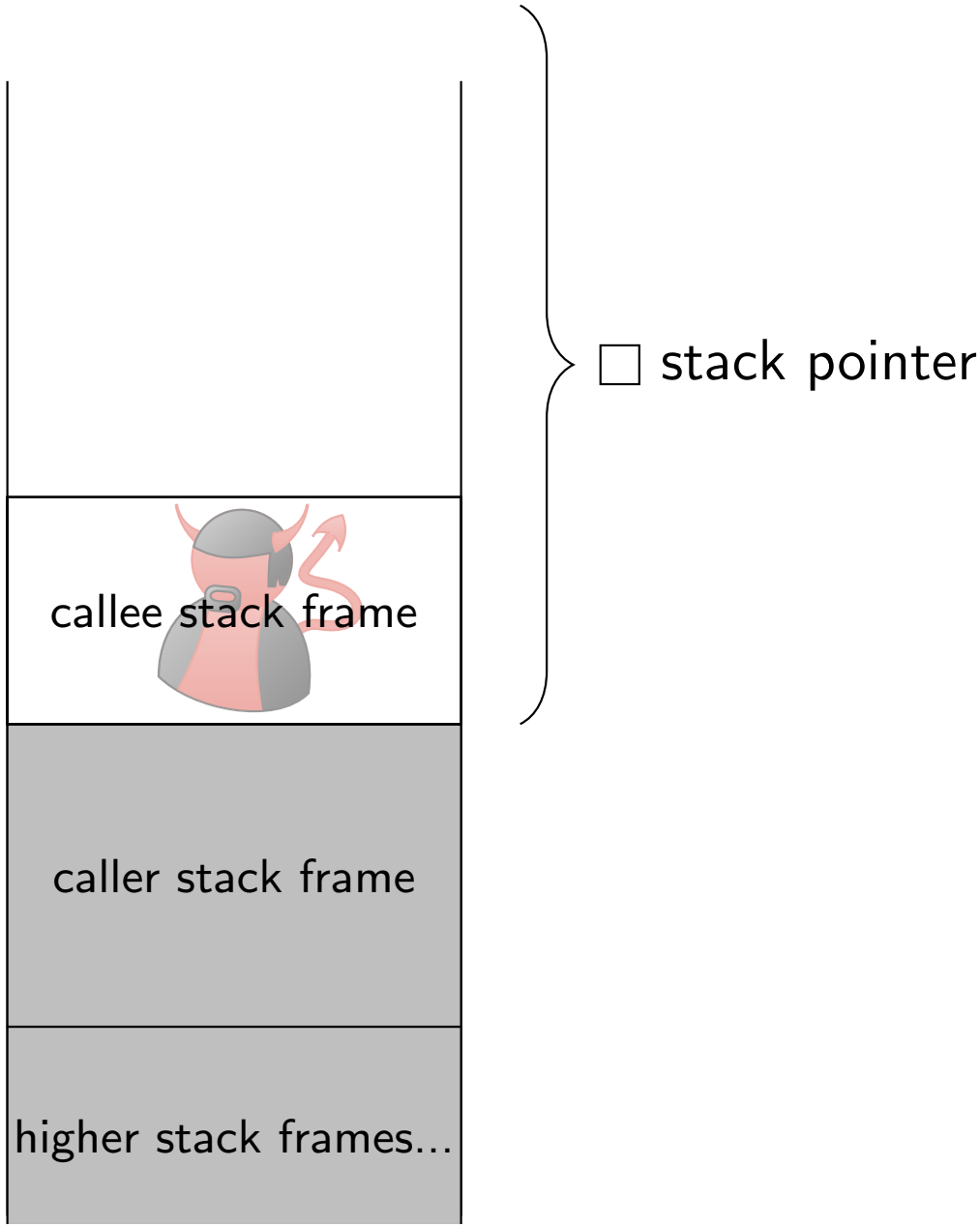
Traditional Stack Pointers



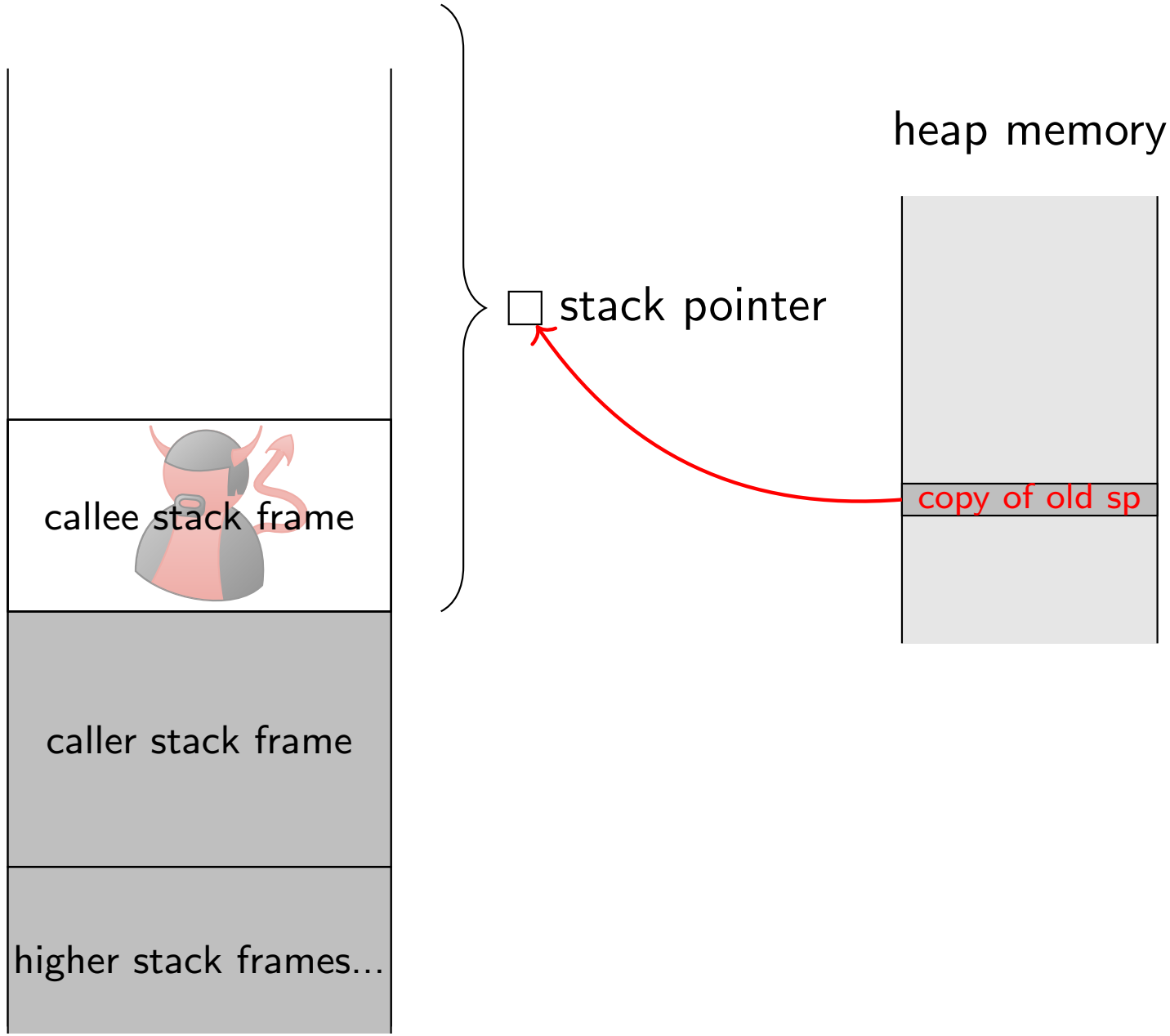
Road map?

Capability machine

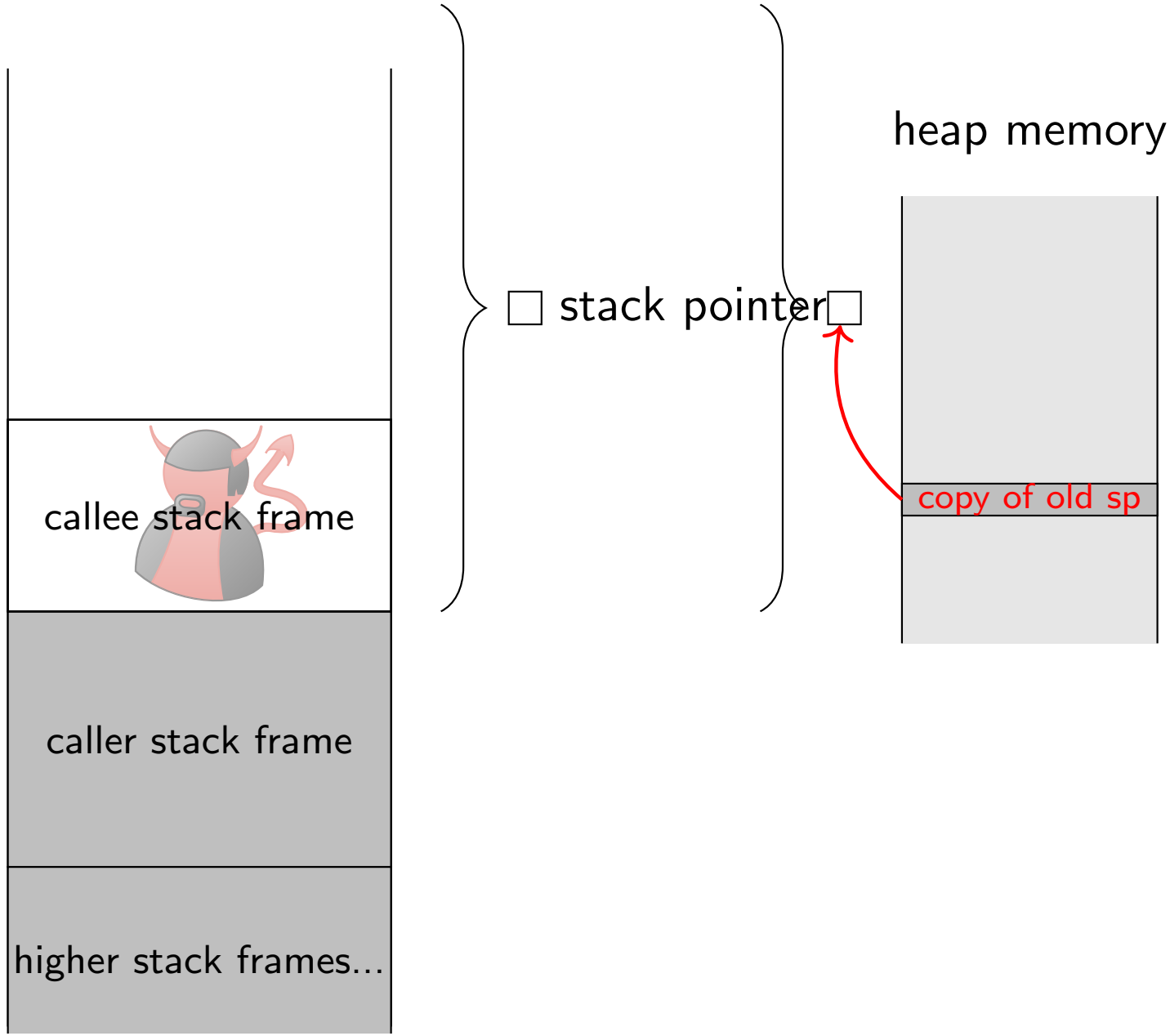
Stack and return capabilities: Attack 1



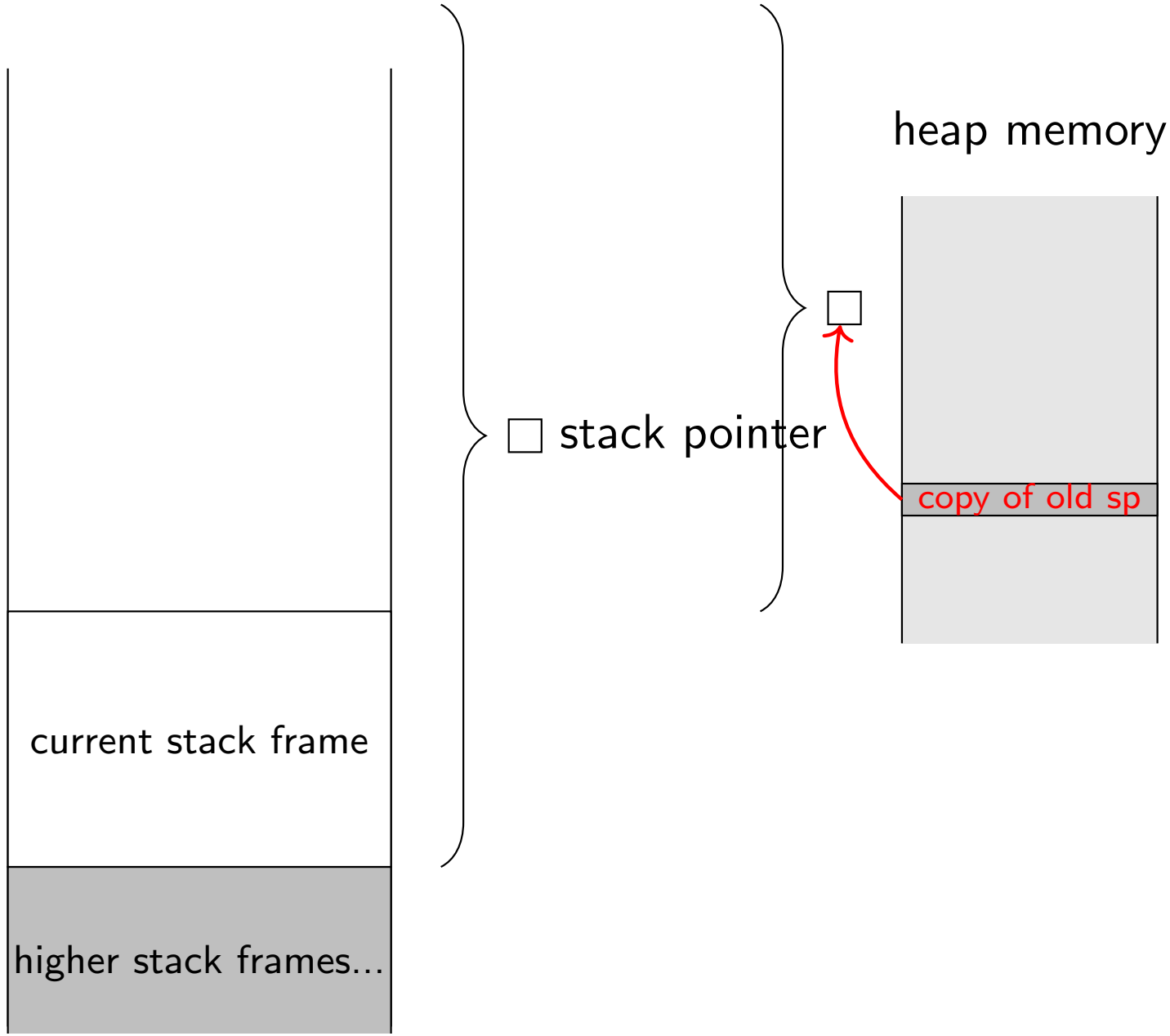
Stack and return capabilities: Attack 1



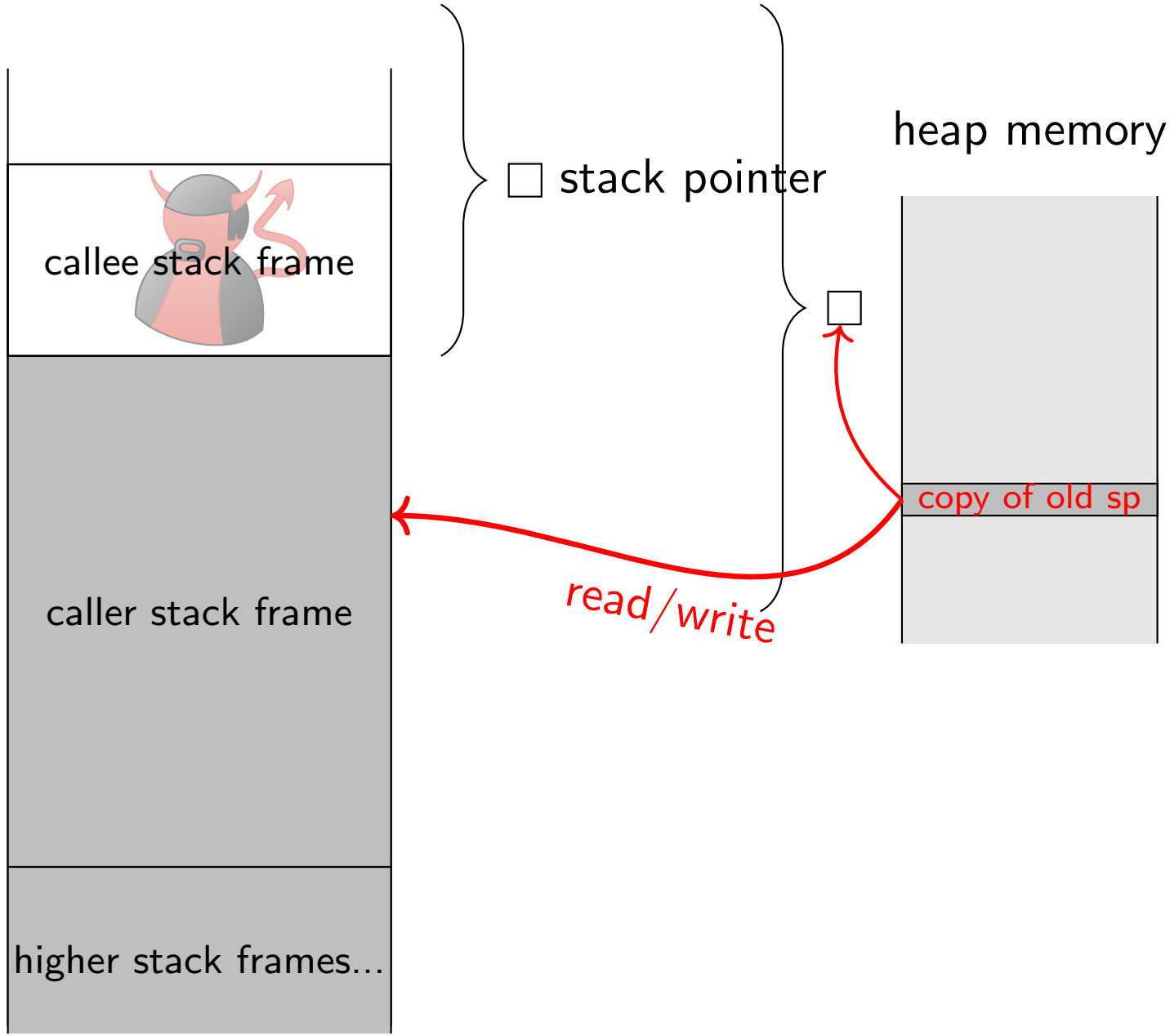
Stack and return capabilities: Attack 1



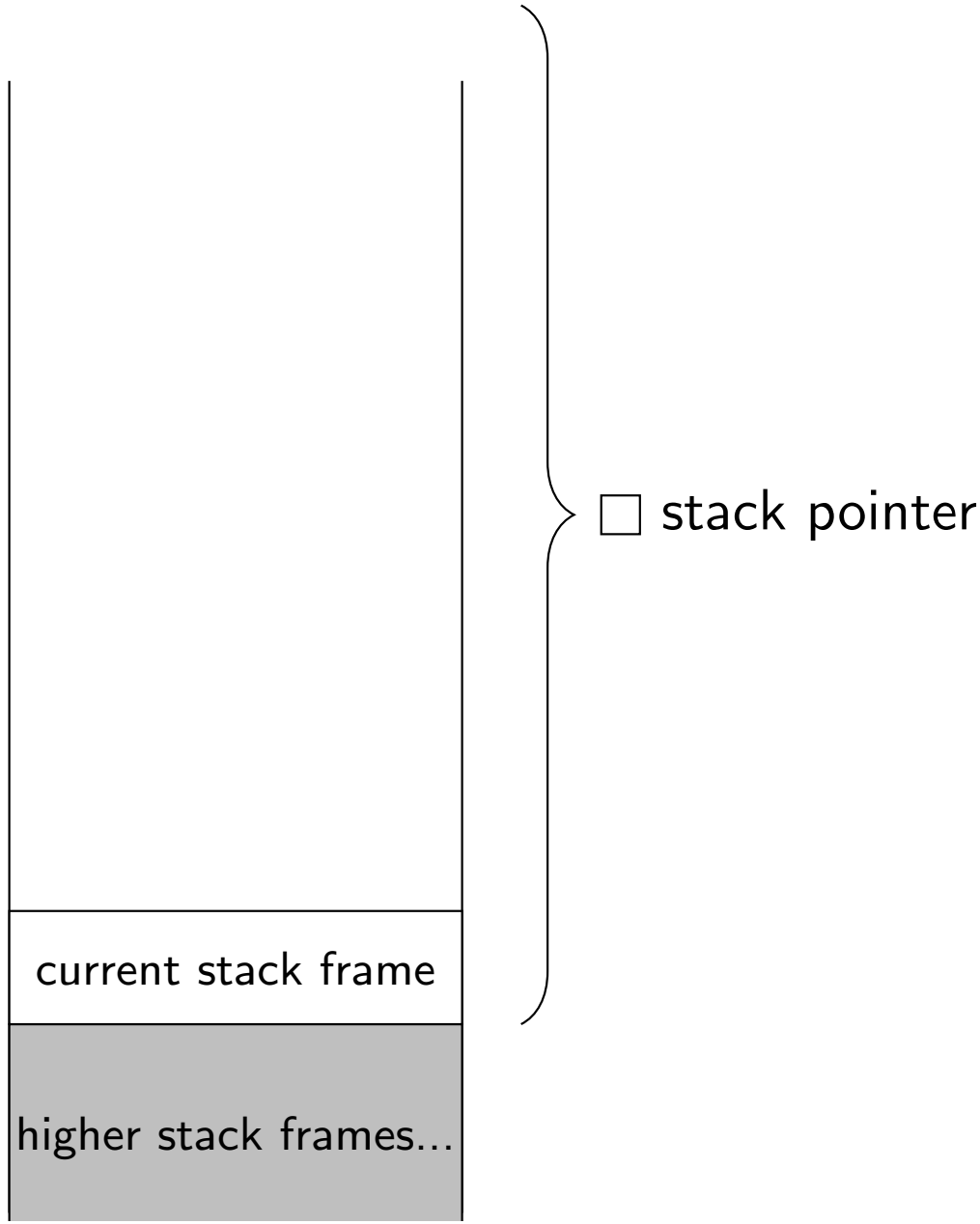
Stack and return capabilities: Attack 1



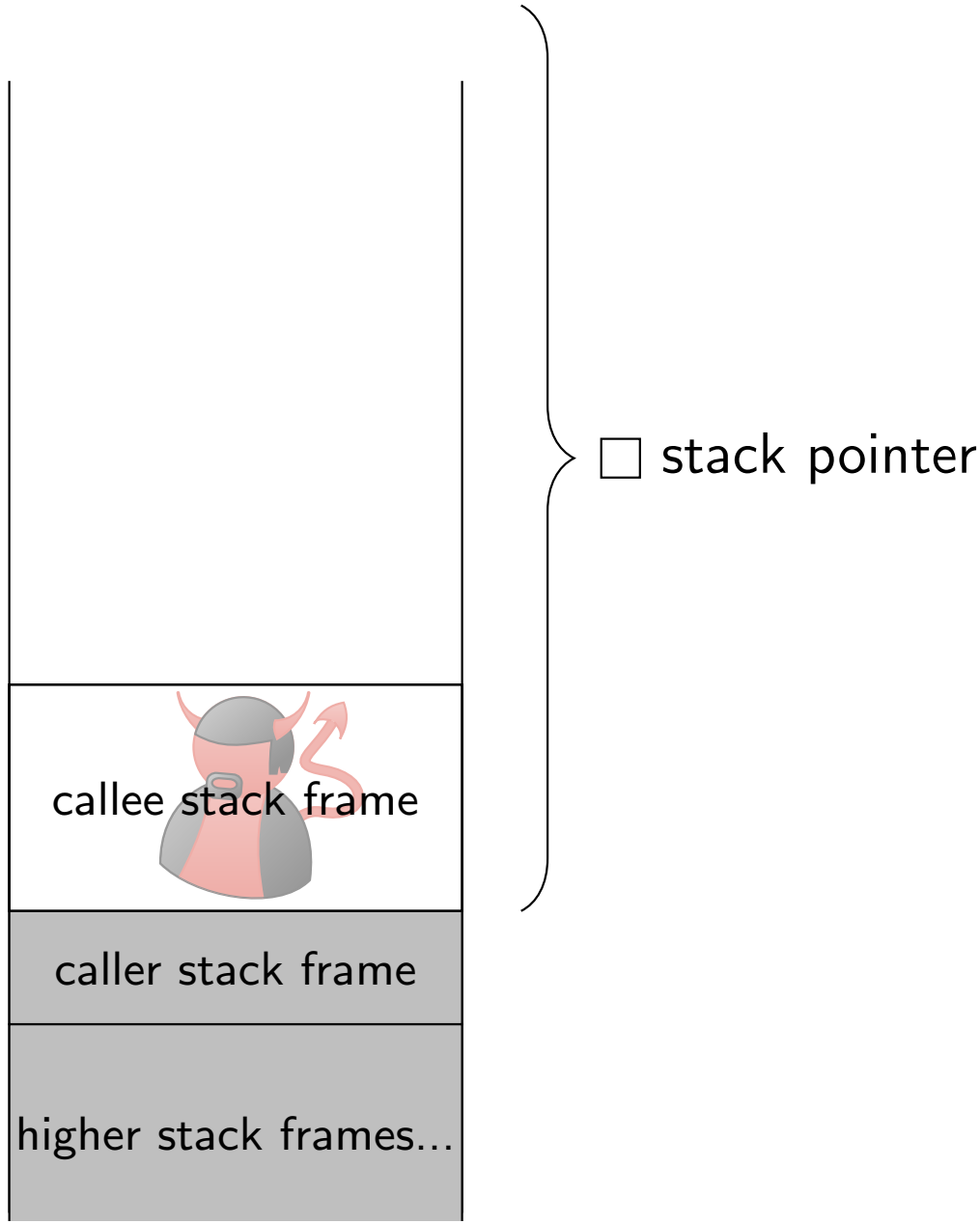
Stack and return capabilities: Attack 1



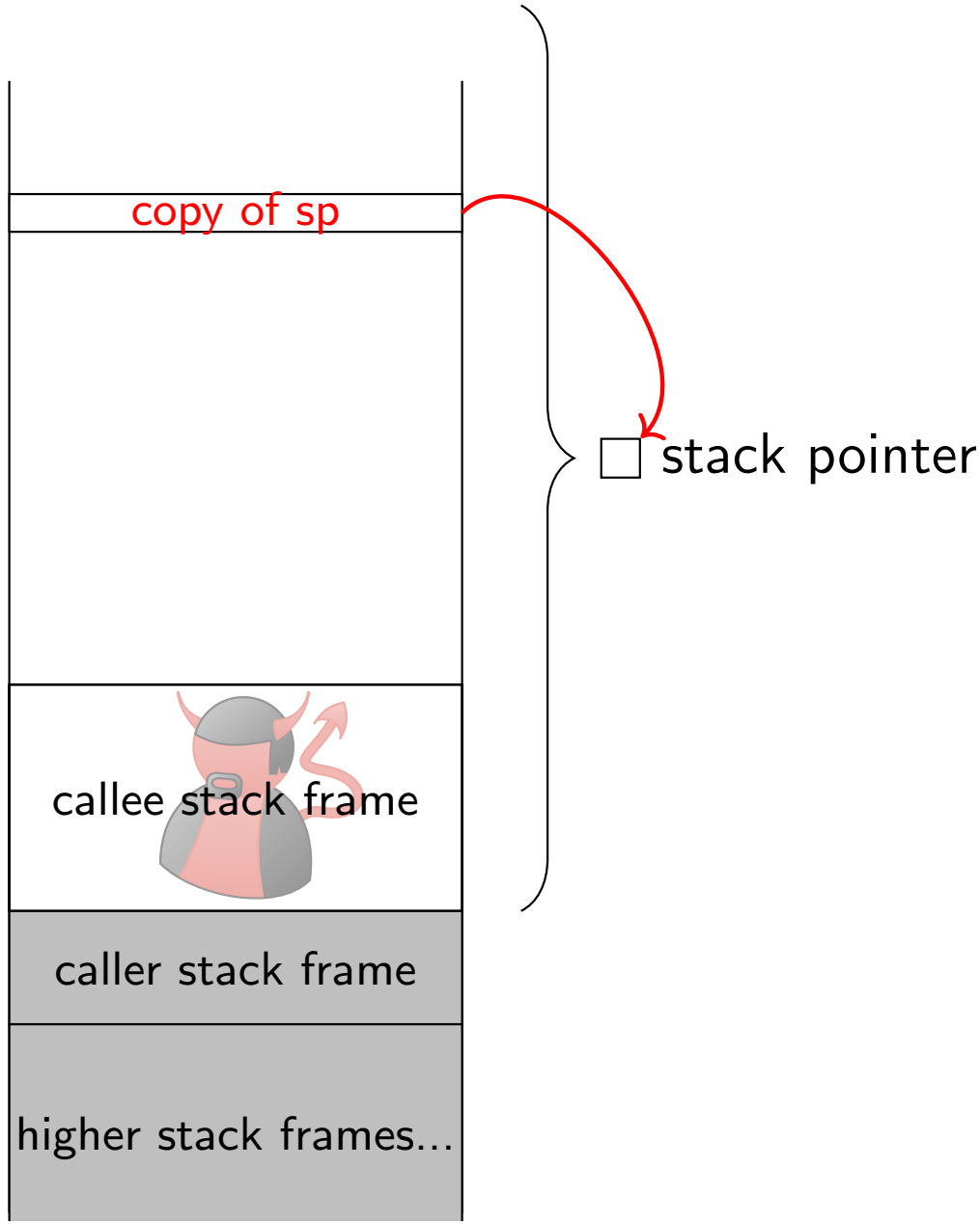
Stack and return capabilities: Attack 2



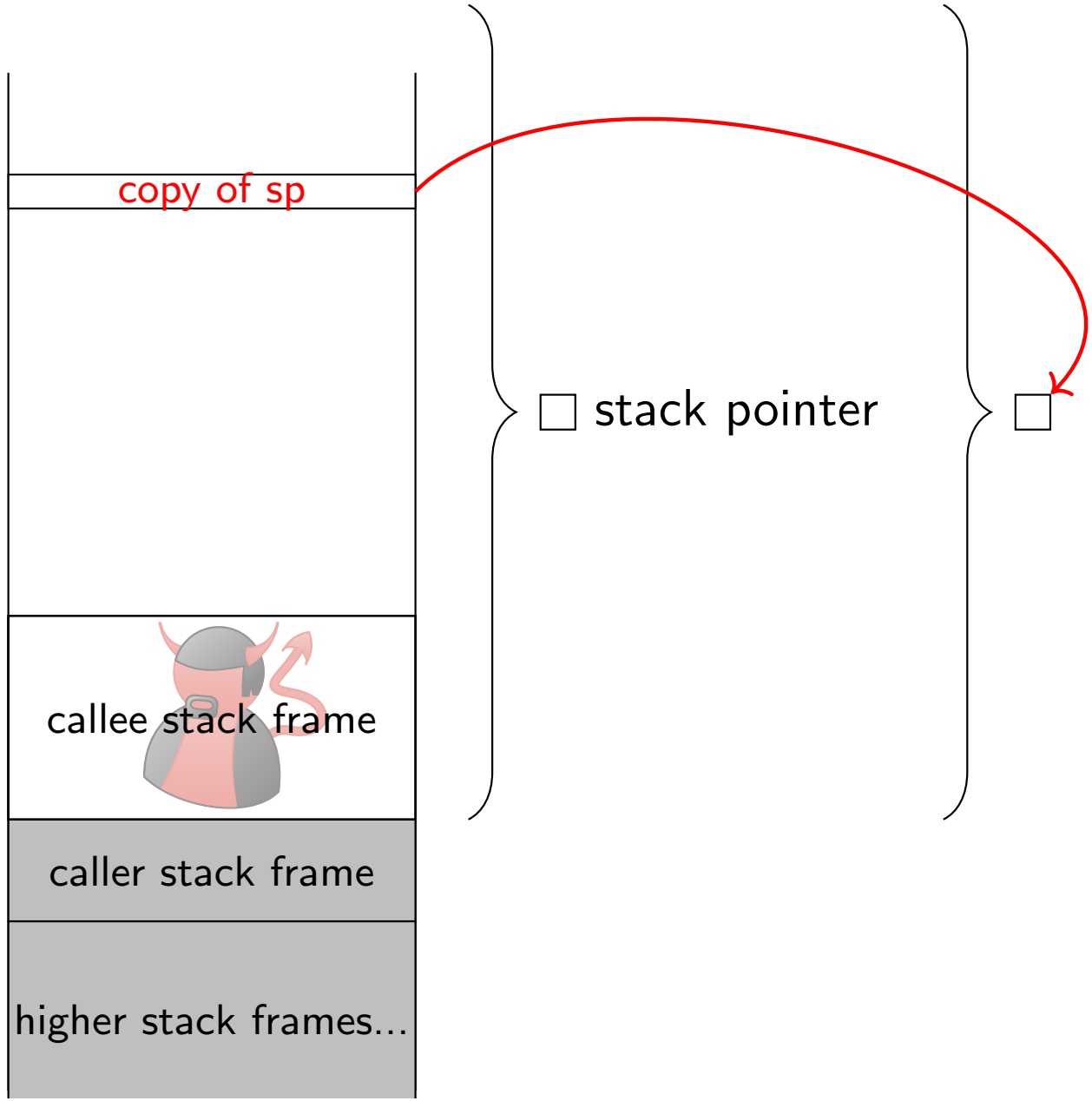
Stack and return capabilities: Attack 2



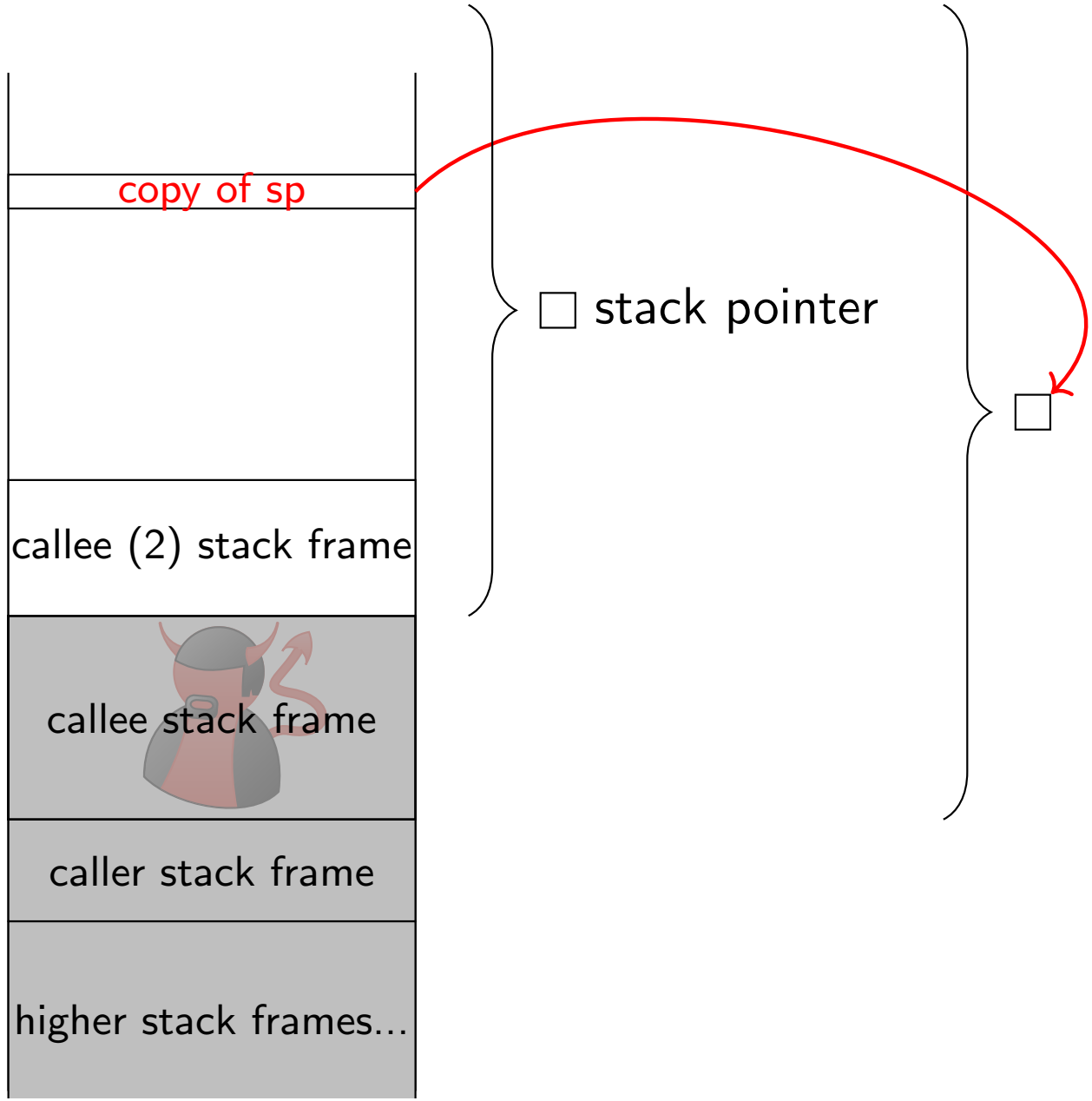
Stack and return capabilities: Attack 2



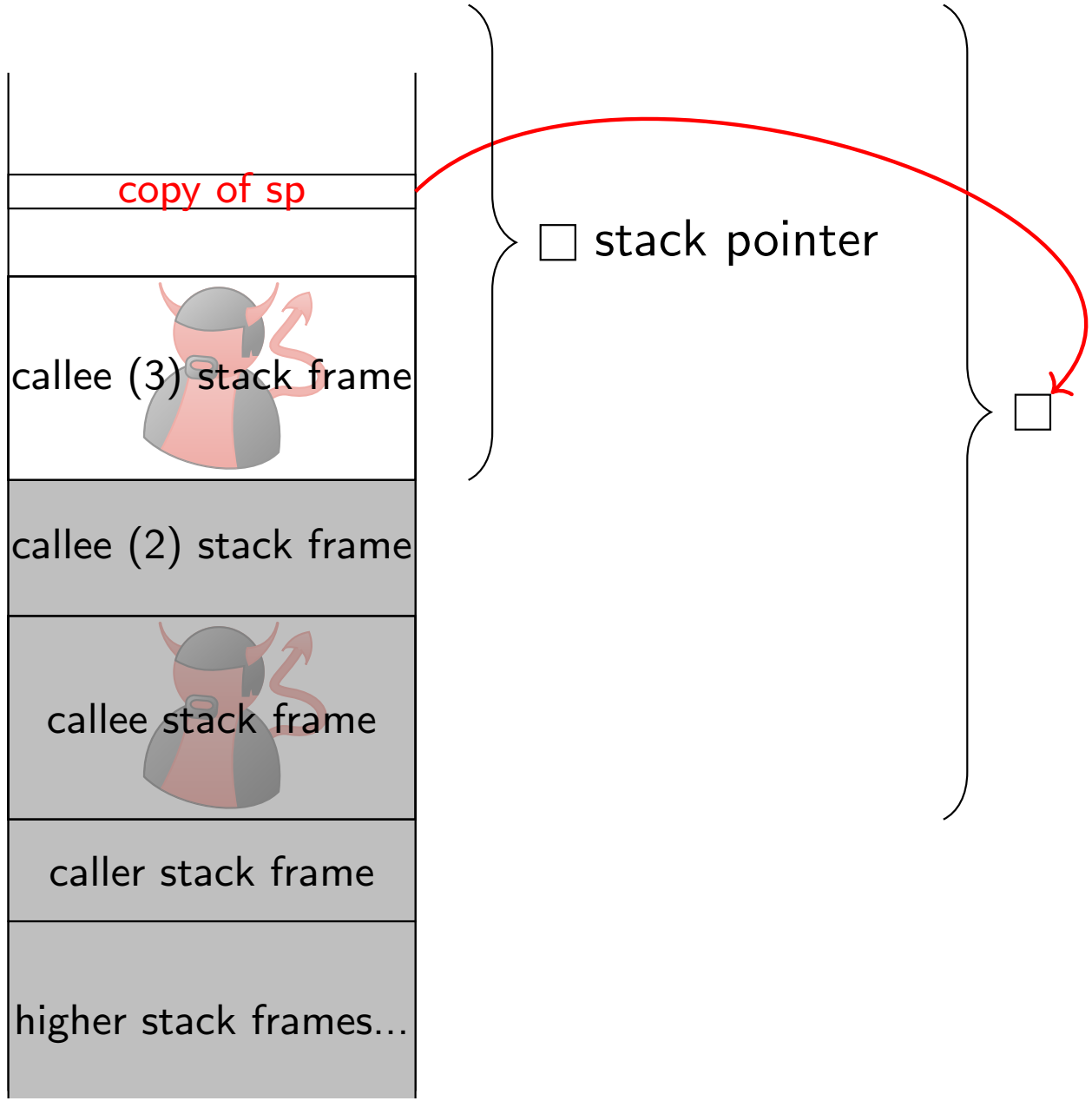
Stack and return capabilities: Attack 2



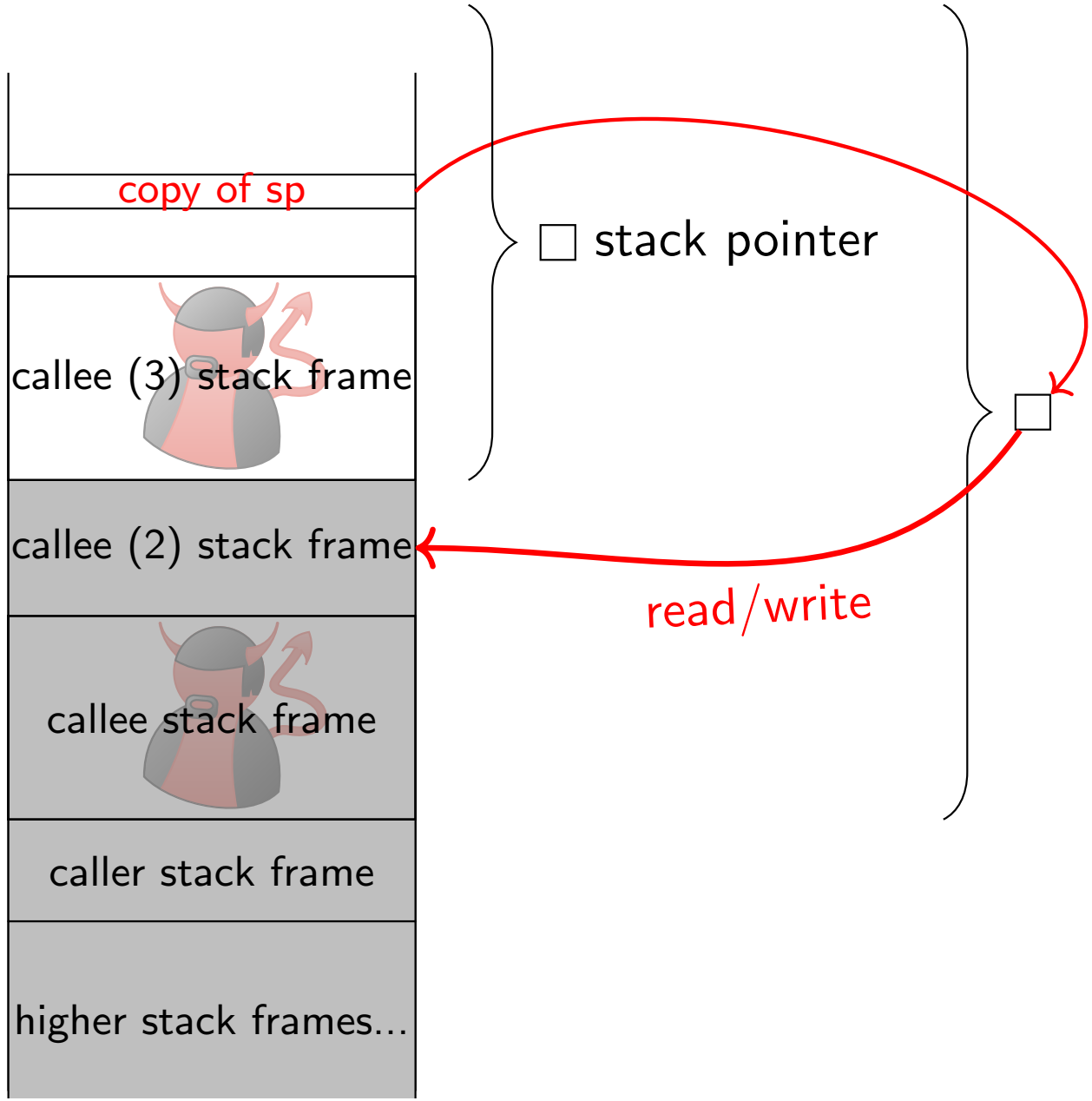
Stack and return capabilities: Attack 2



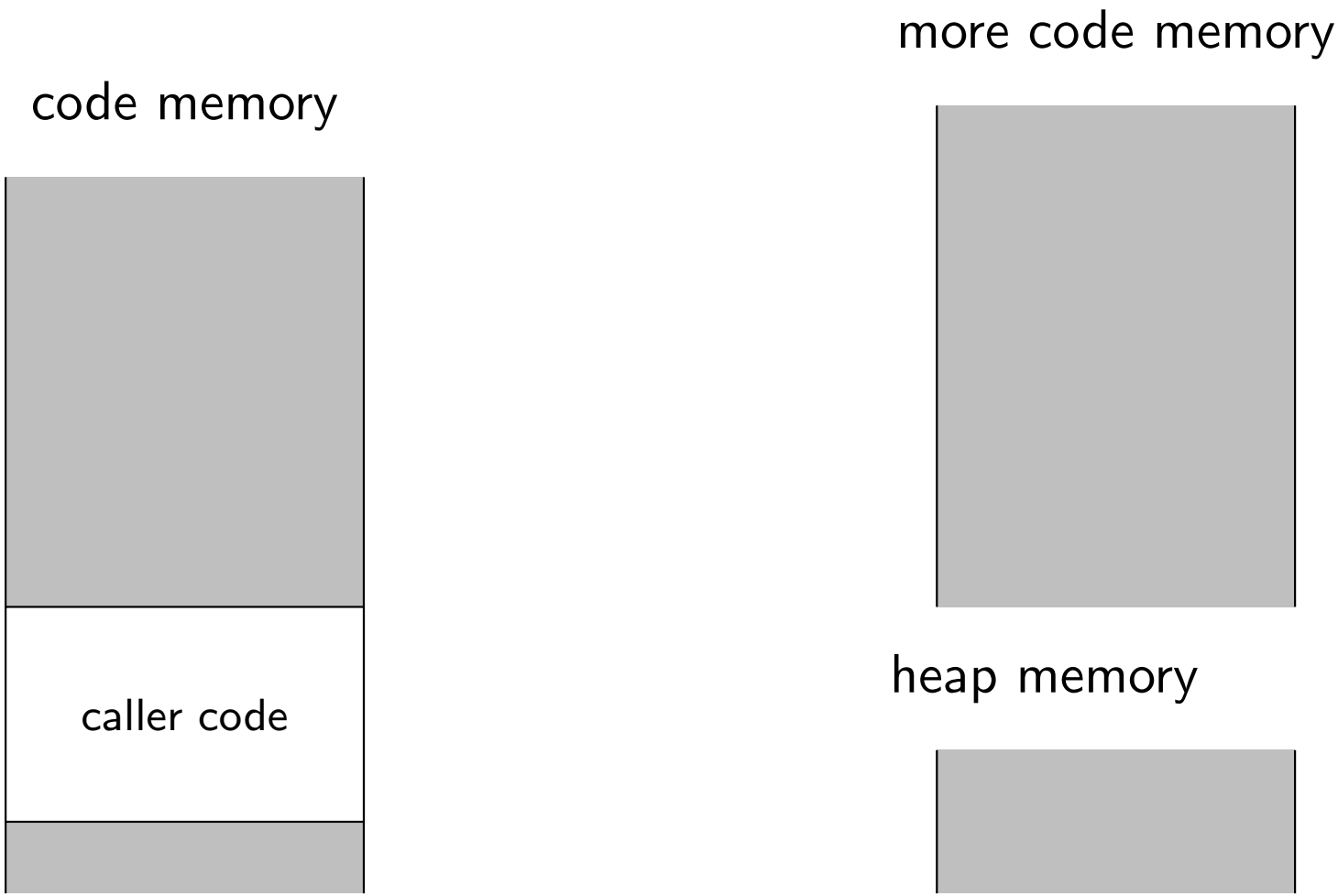
Stack and return capabilities: Attack 2



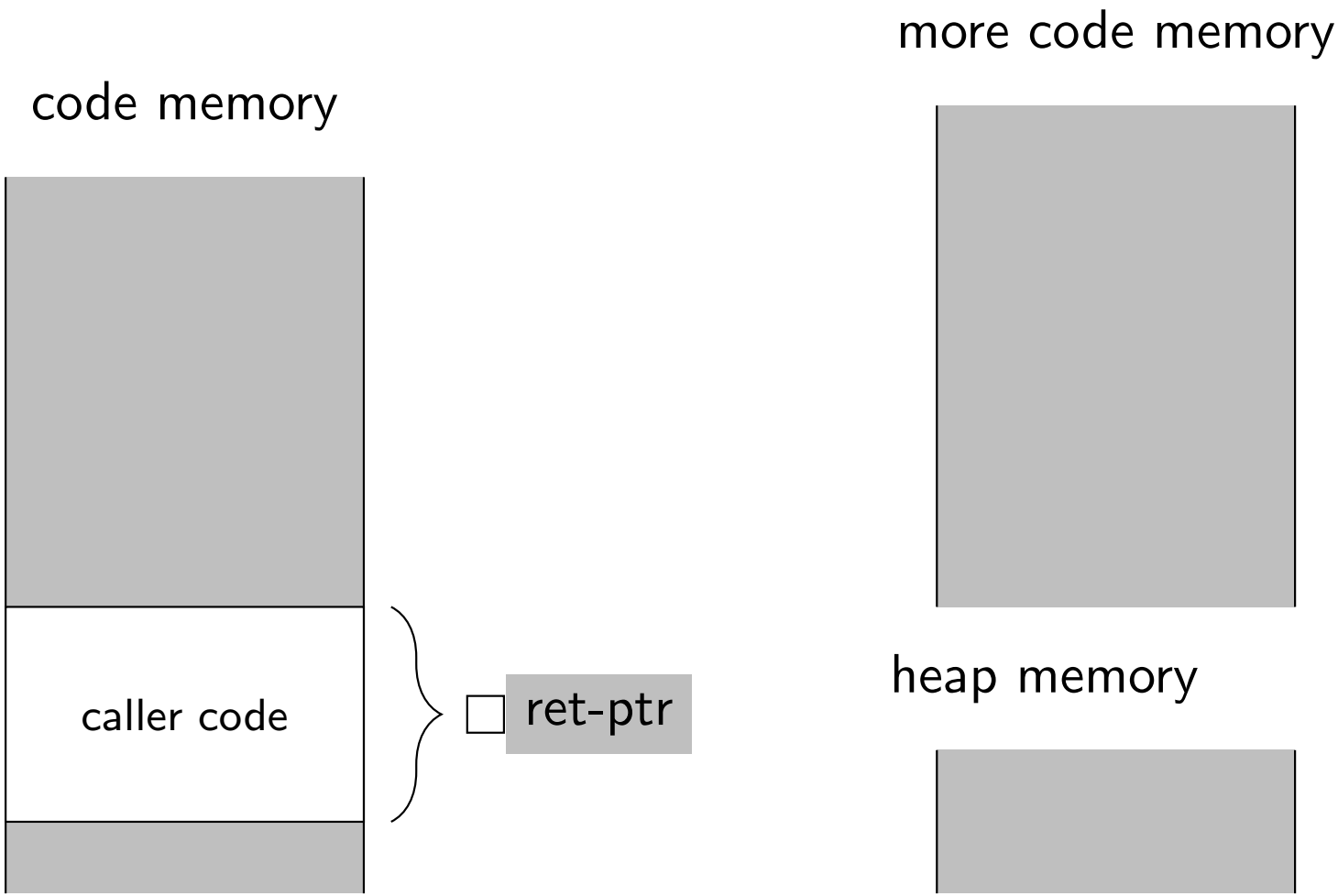
Stack and return capabilities: Attack 2



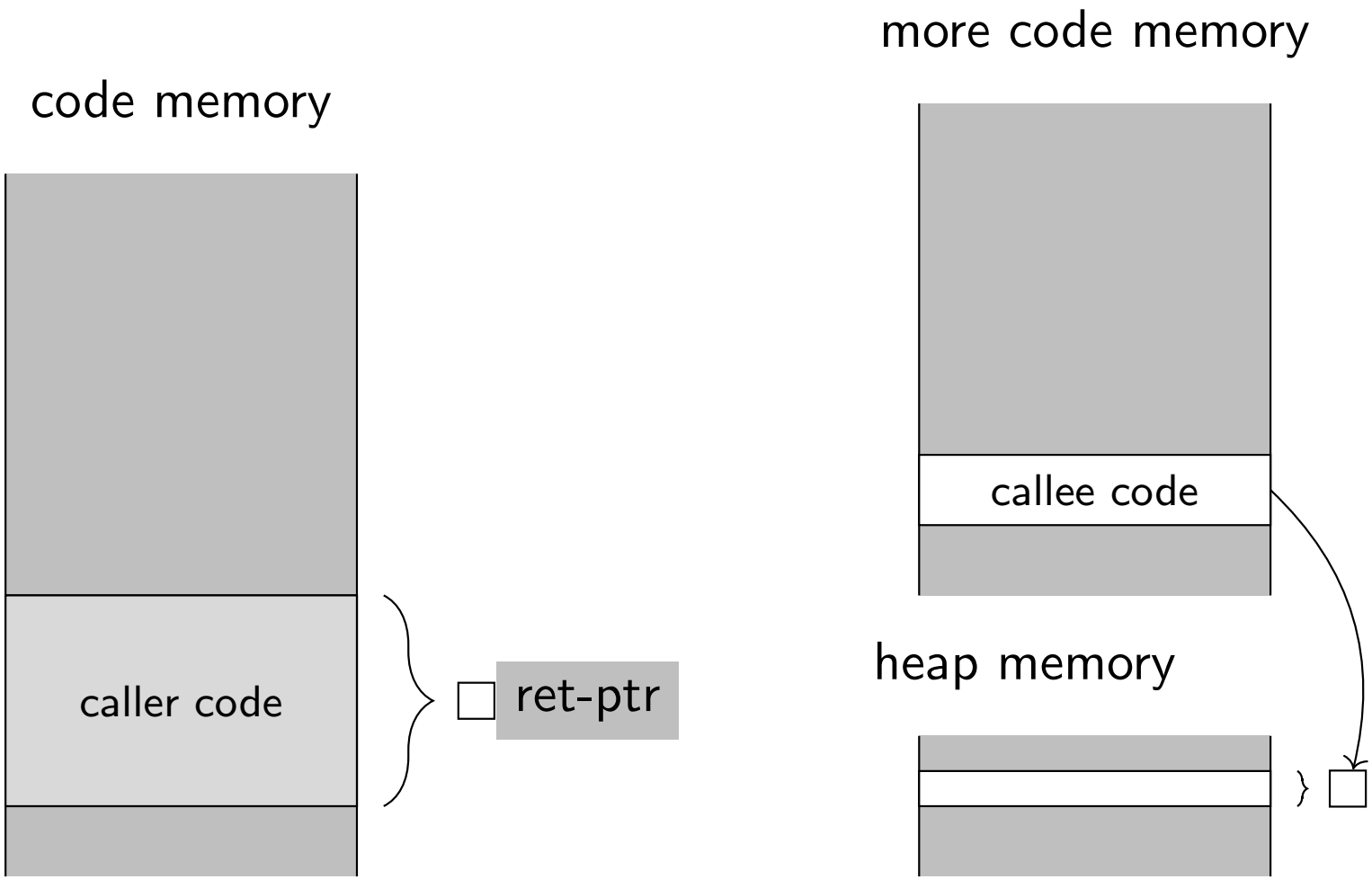
Stack and return capabilities: Attack 3



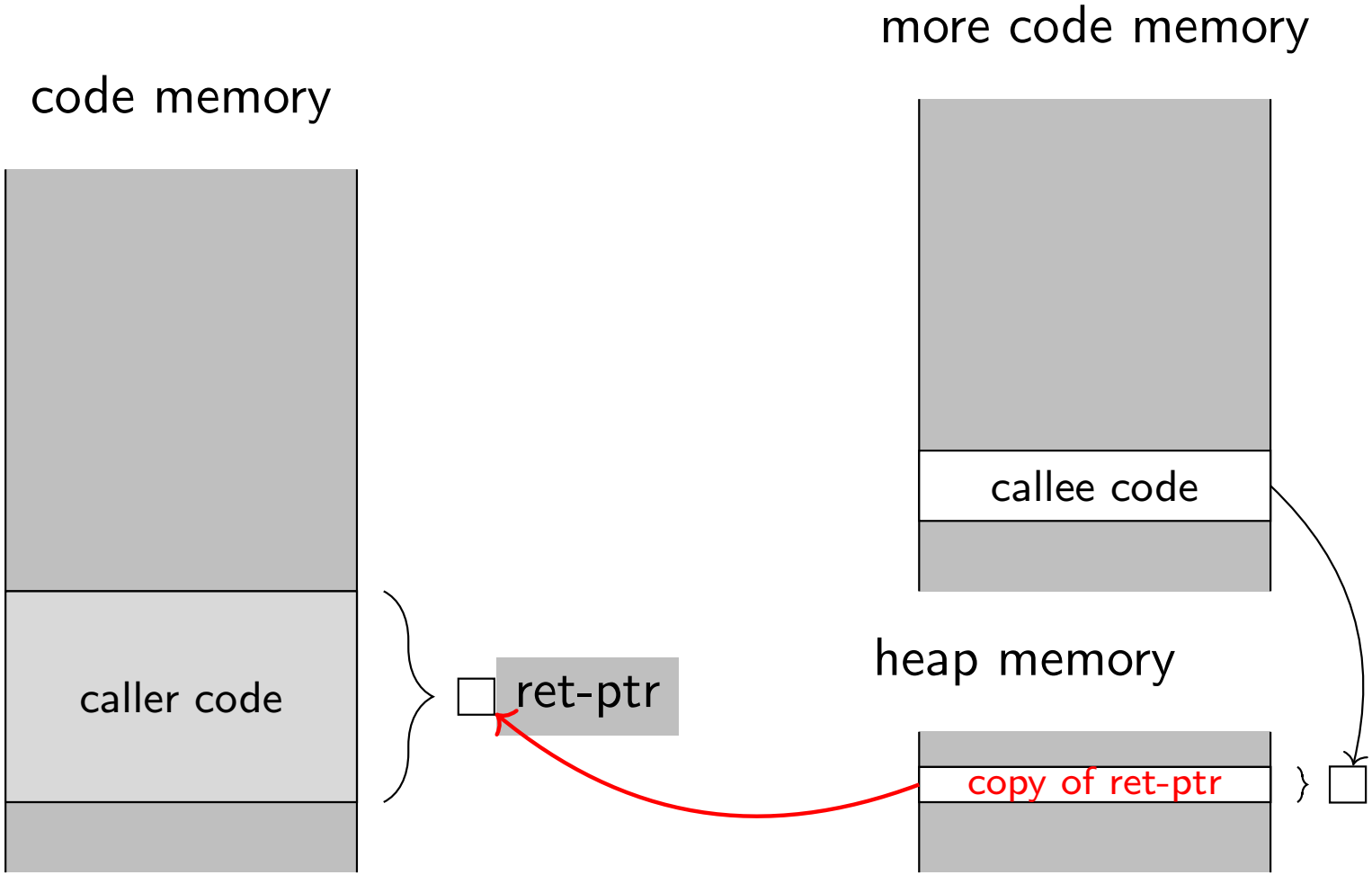
Stack and return capabilities: Attack 3



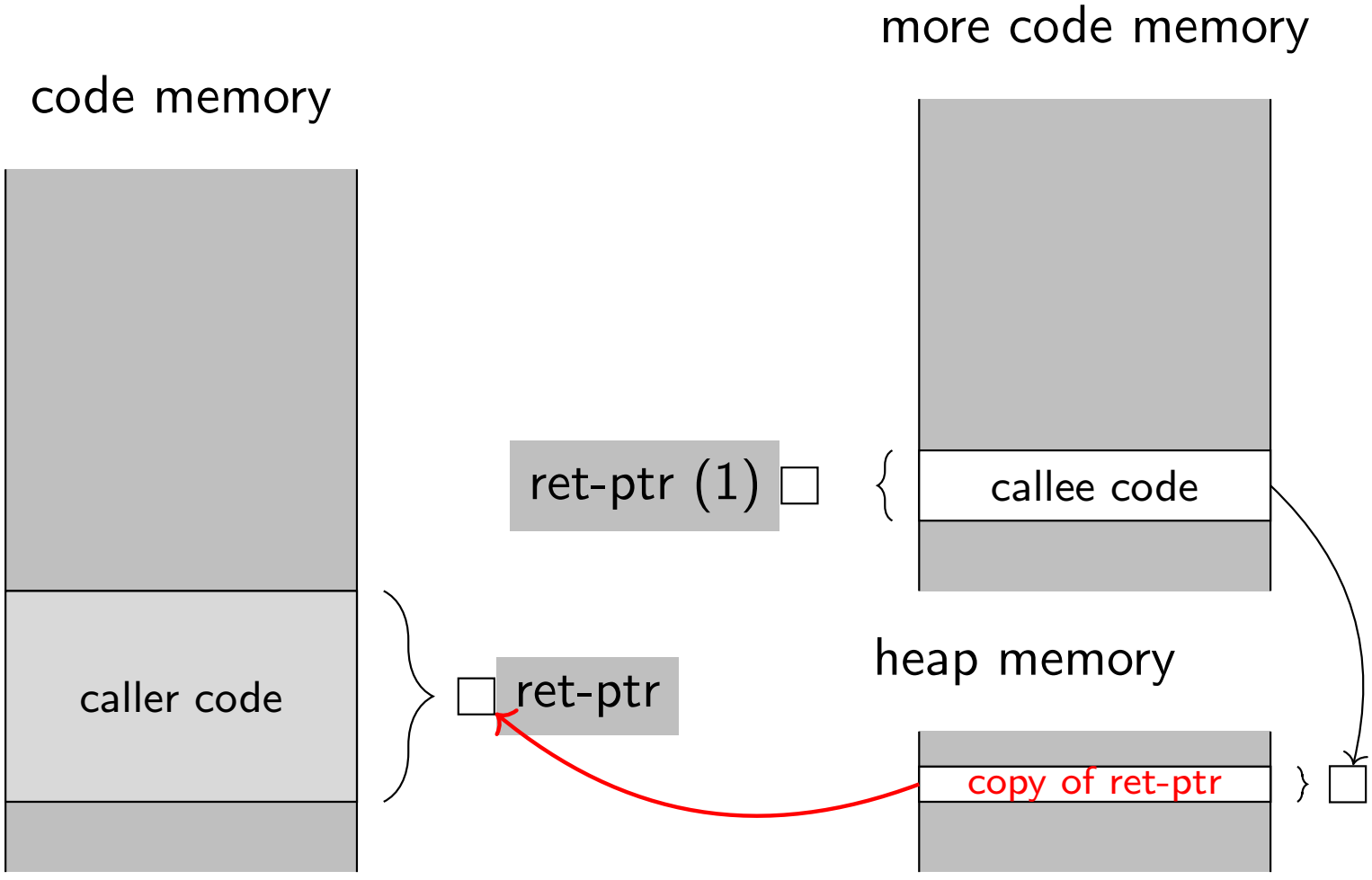
Stack and return capabilities: Attack 3



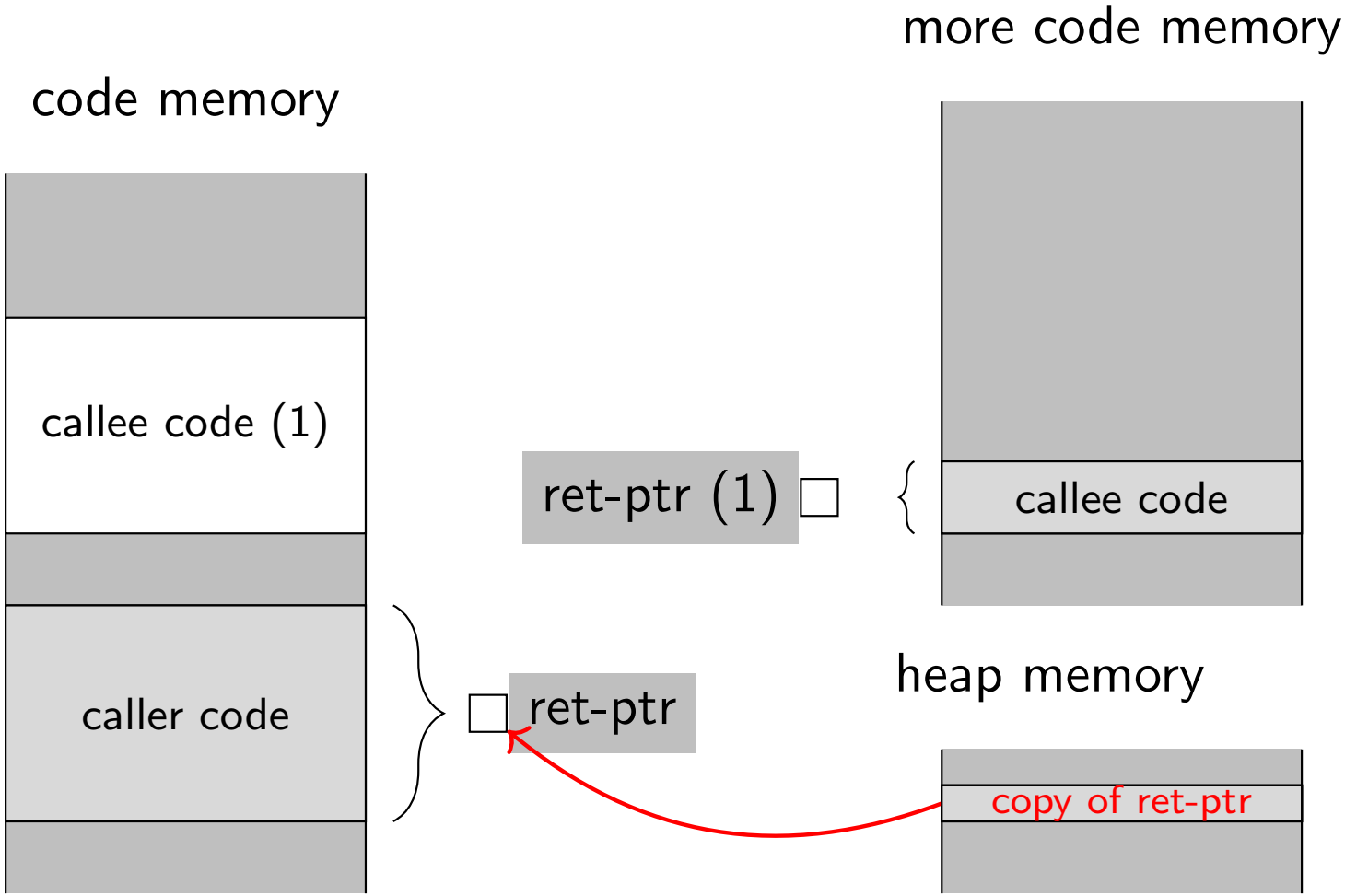
Stack and return capabilities: Attack 3



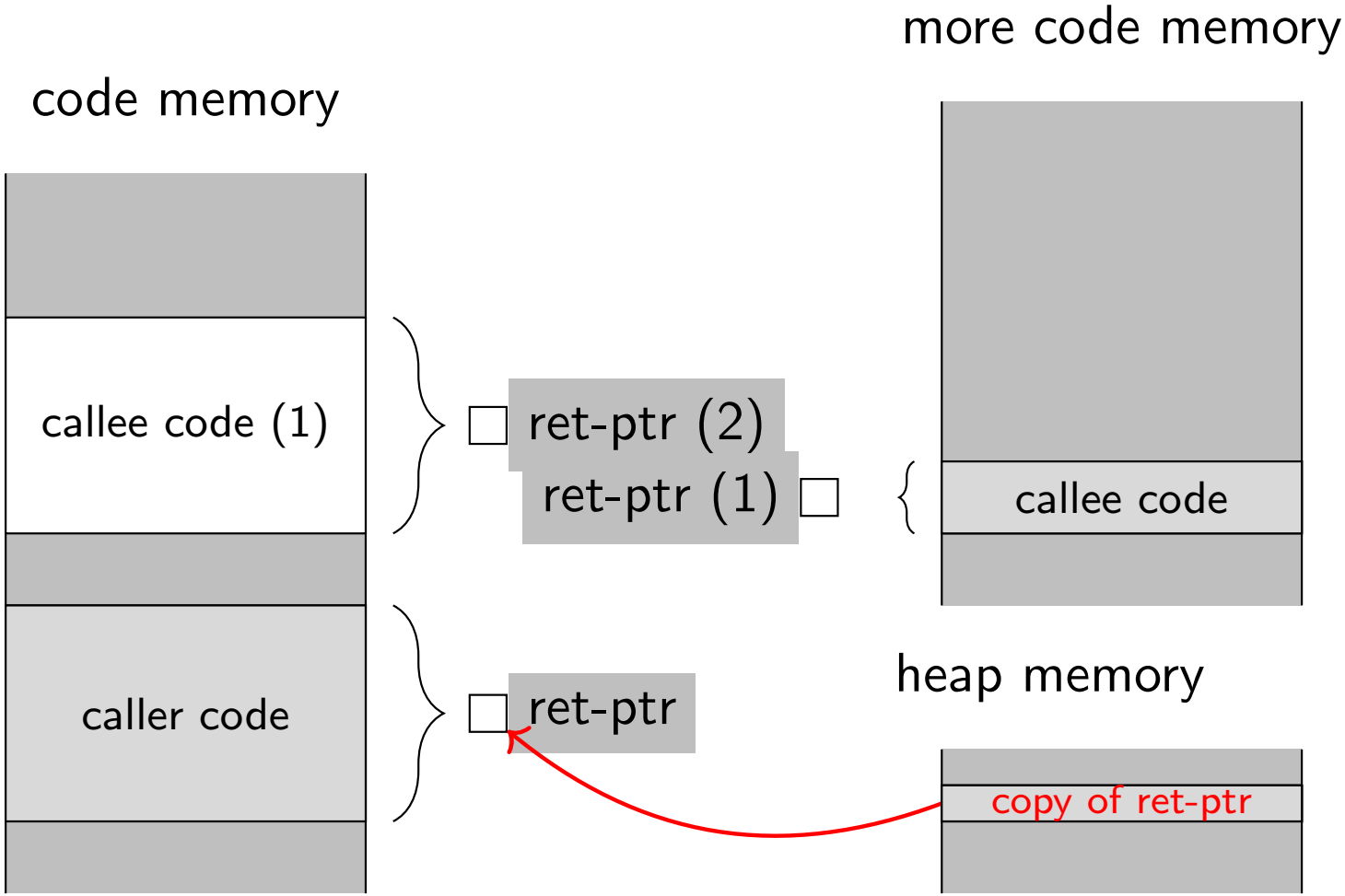
Stack and return capabilities: Attack 3



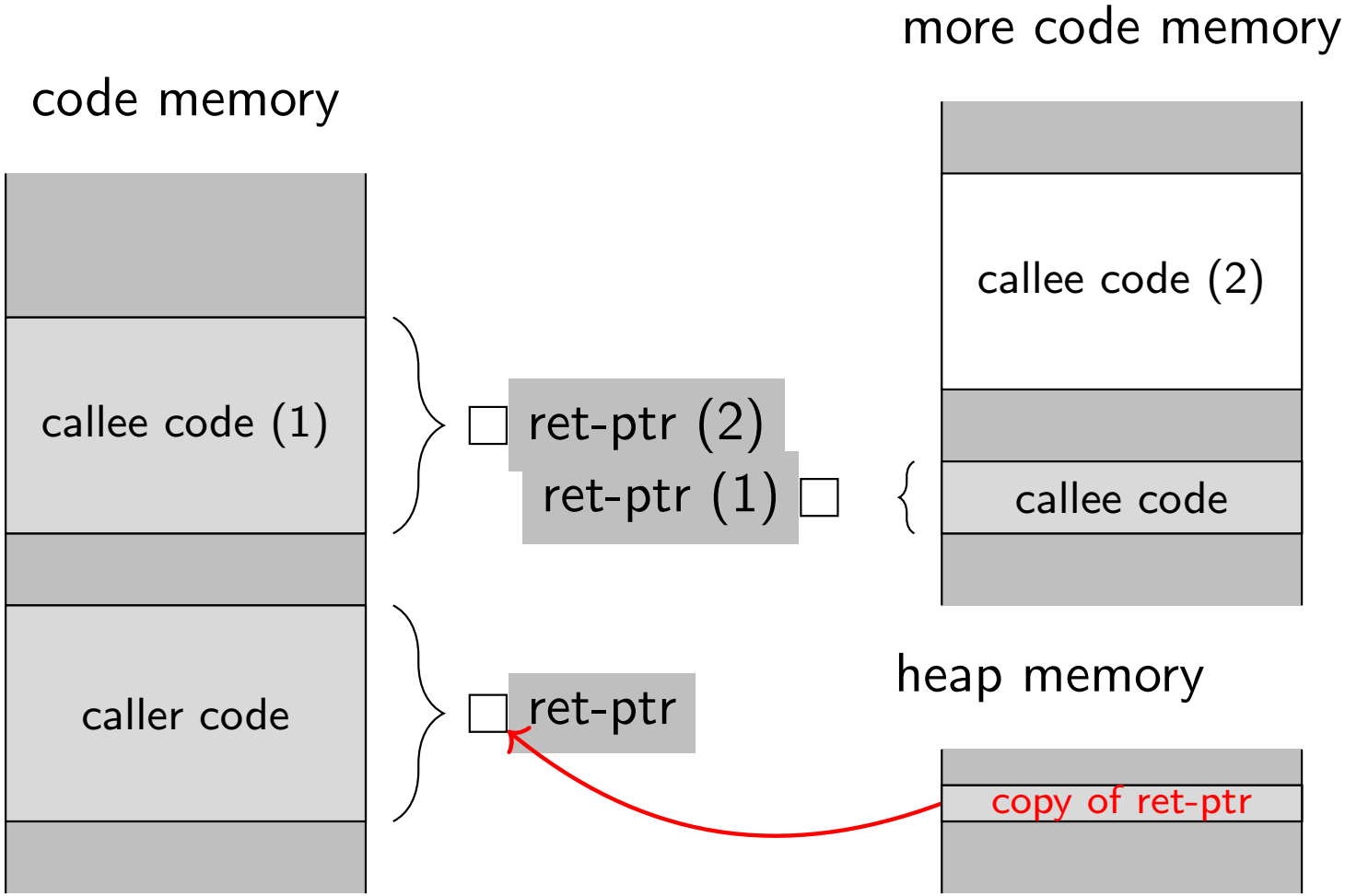
Stack and return capabilities: Attack 3



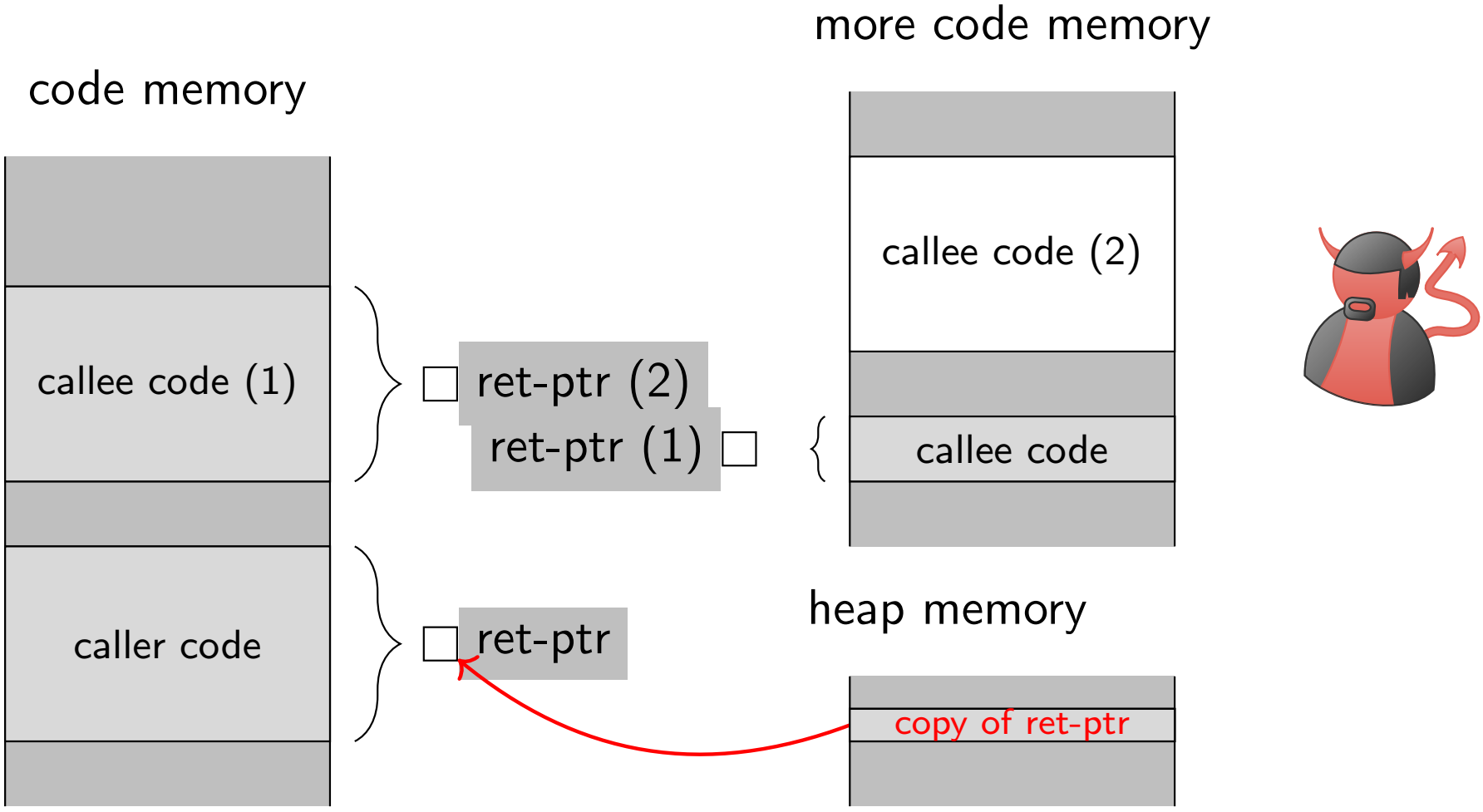
Stack and return capabilities: Attack 3



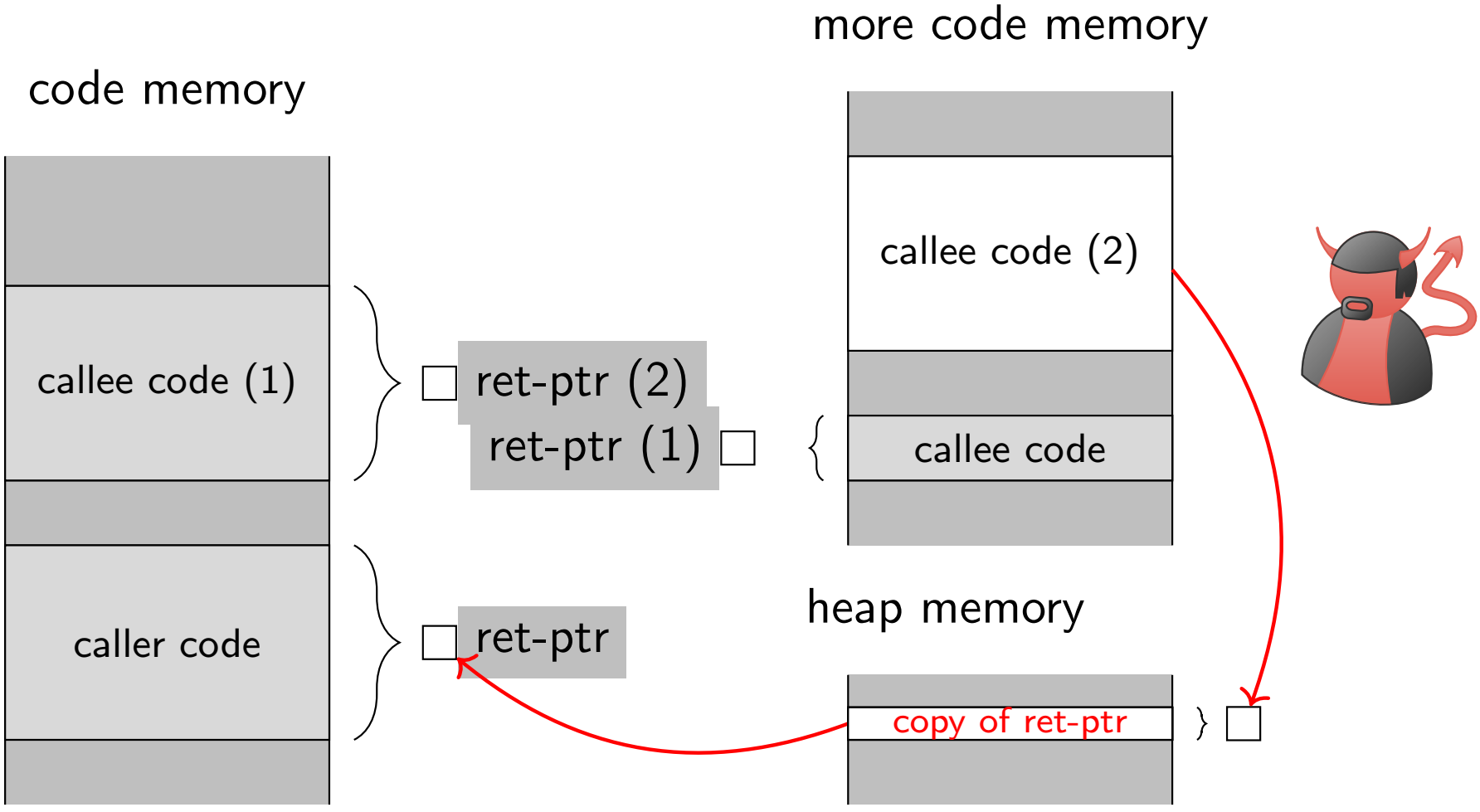
Stack and return capabilities: Attack 3



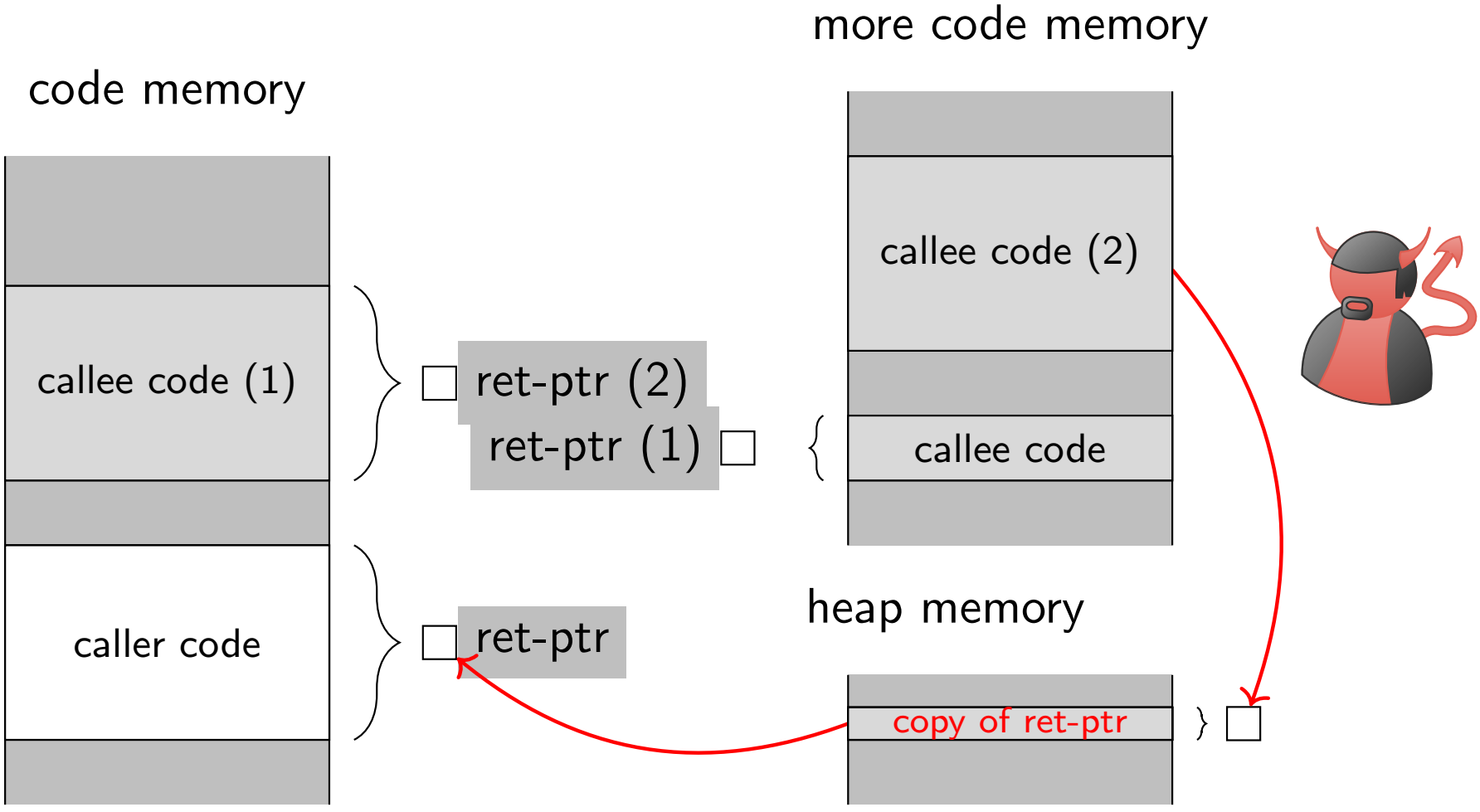
Stack and return capabilities: Attack 3



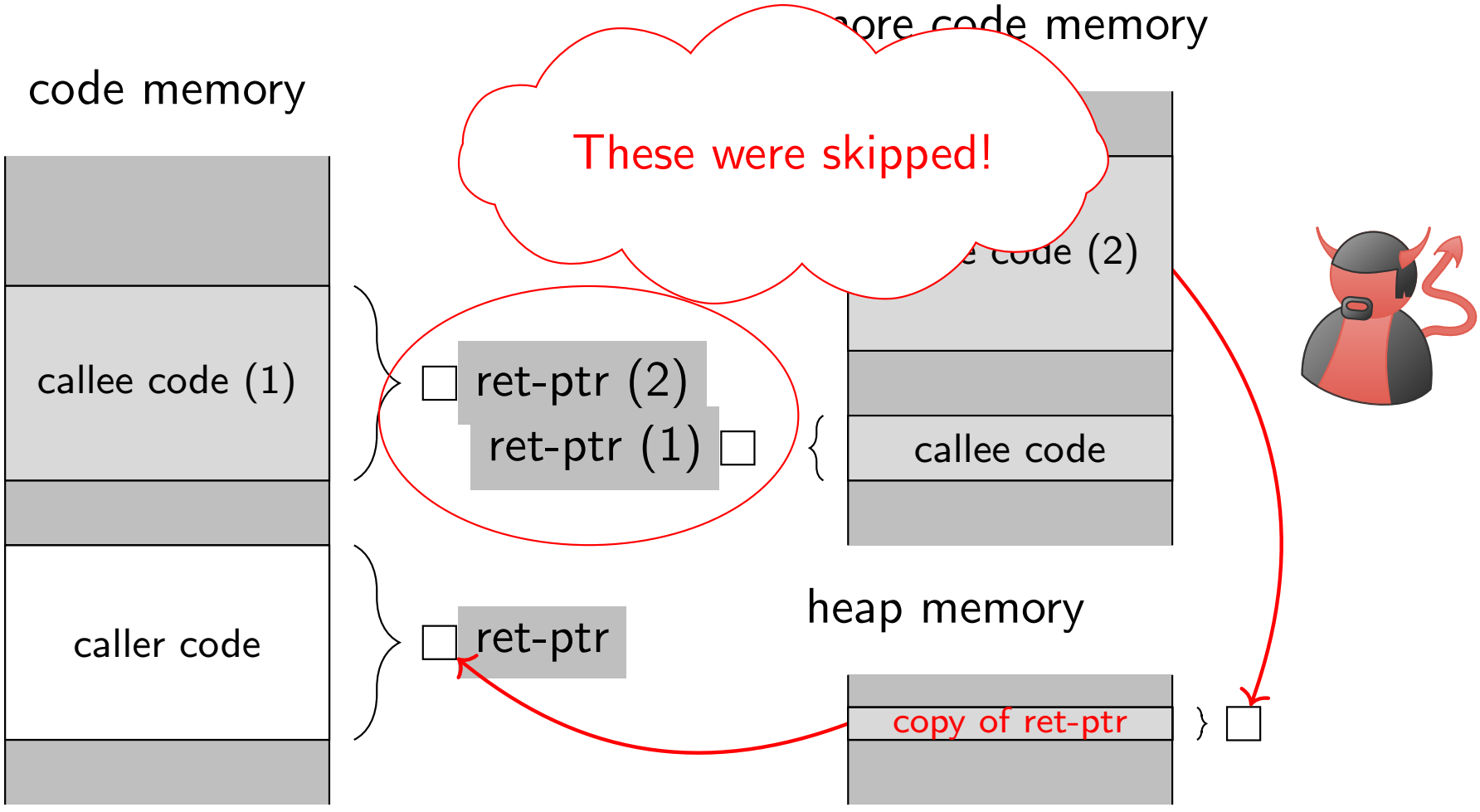
Stack and return capabilities: Attack 3



Stack and return capabilities: Attack 3



Stack and return capabilities: Attack 3



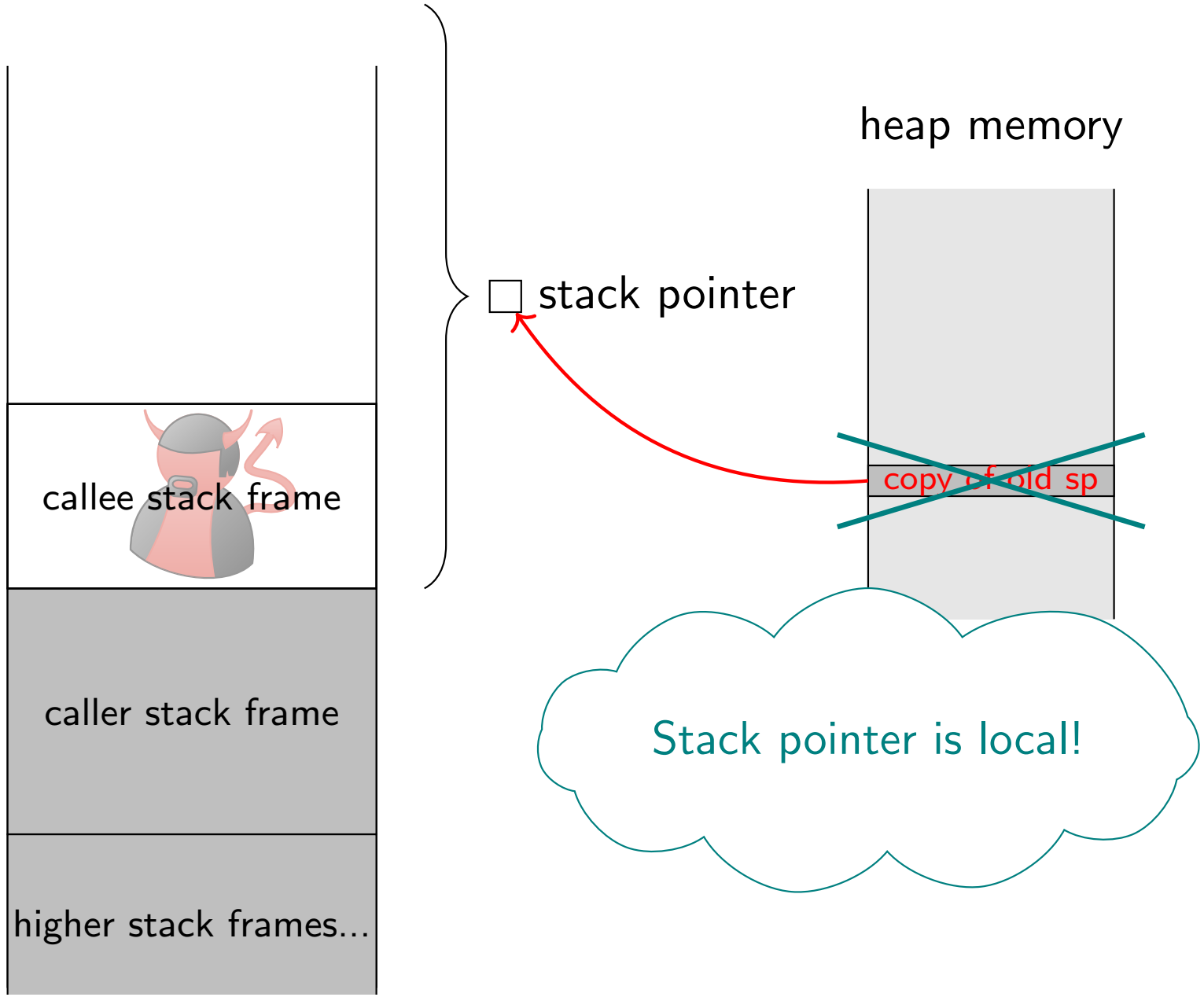
Local capabilities

- Capabilities tagged with locality (local or normal)
- New write-local permission.
- Local capabilities can only be stored by capabilities with write-local permission

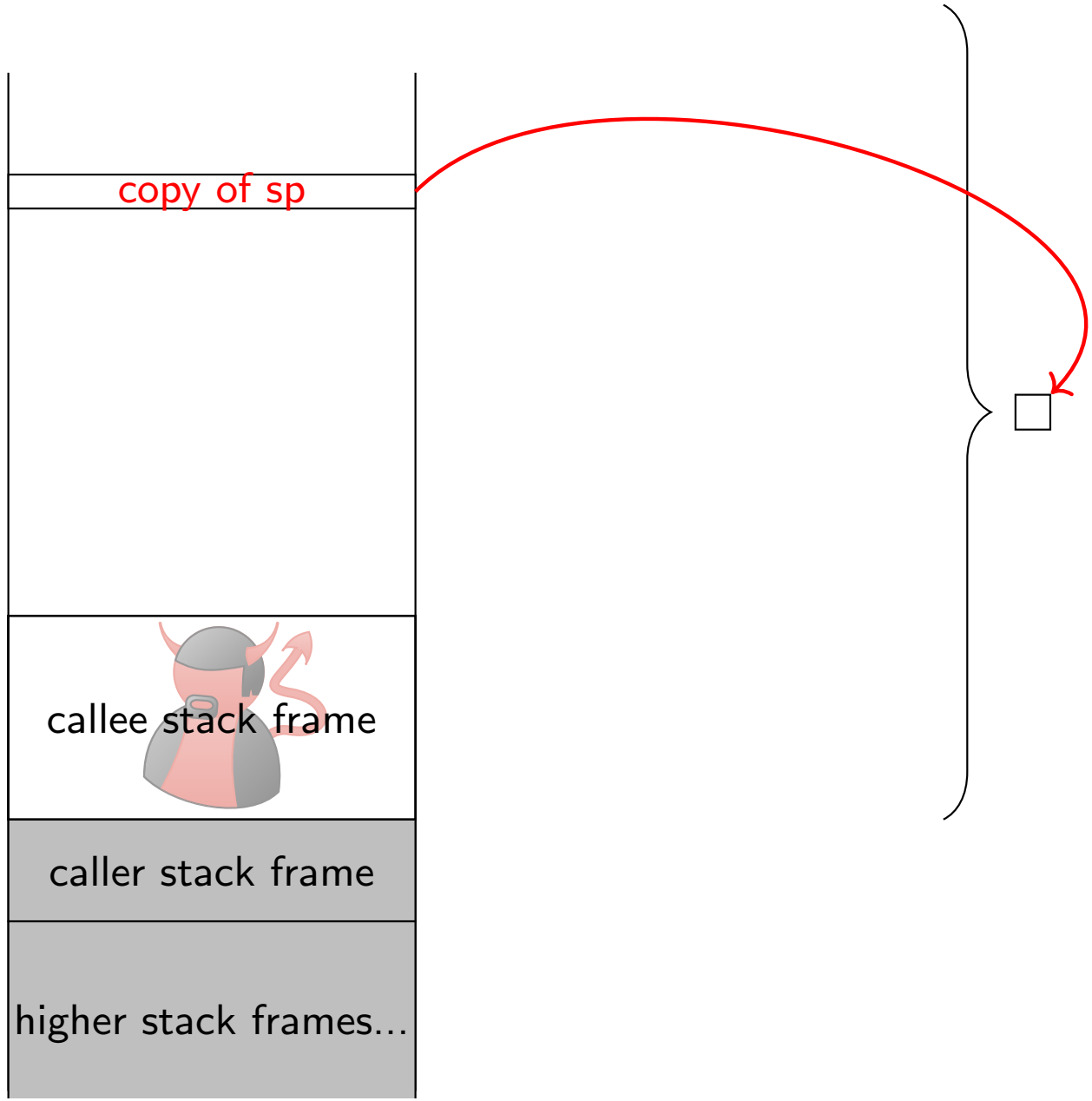
Calling convention highlights

- Stack capability is local with permission read, write-local, and execute.
- Clear stack before passing stack capability to untrusted code.

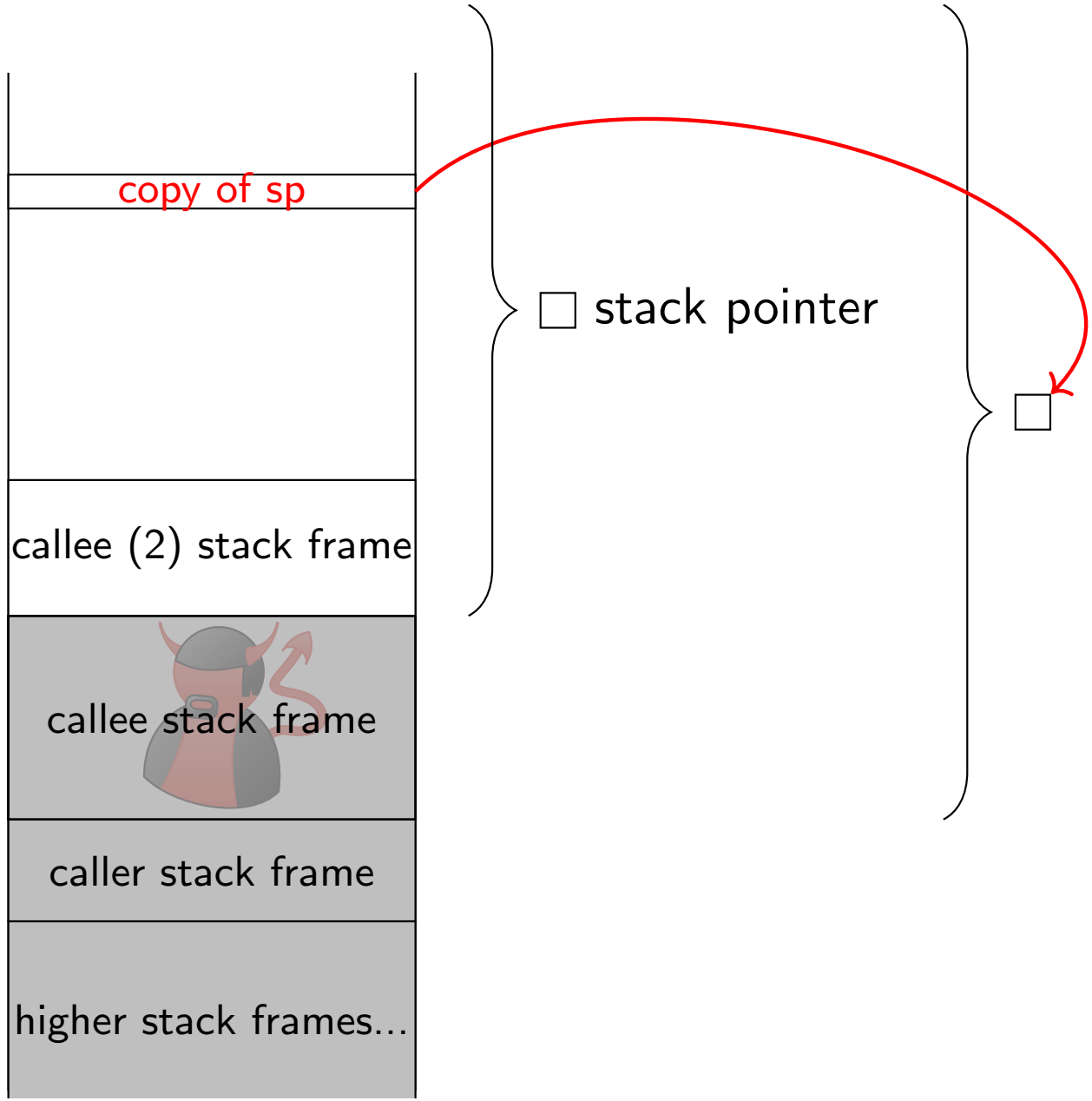
Local Stack Capabilities Prevent Attack 1



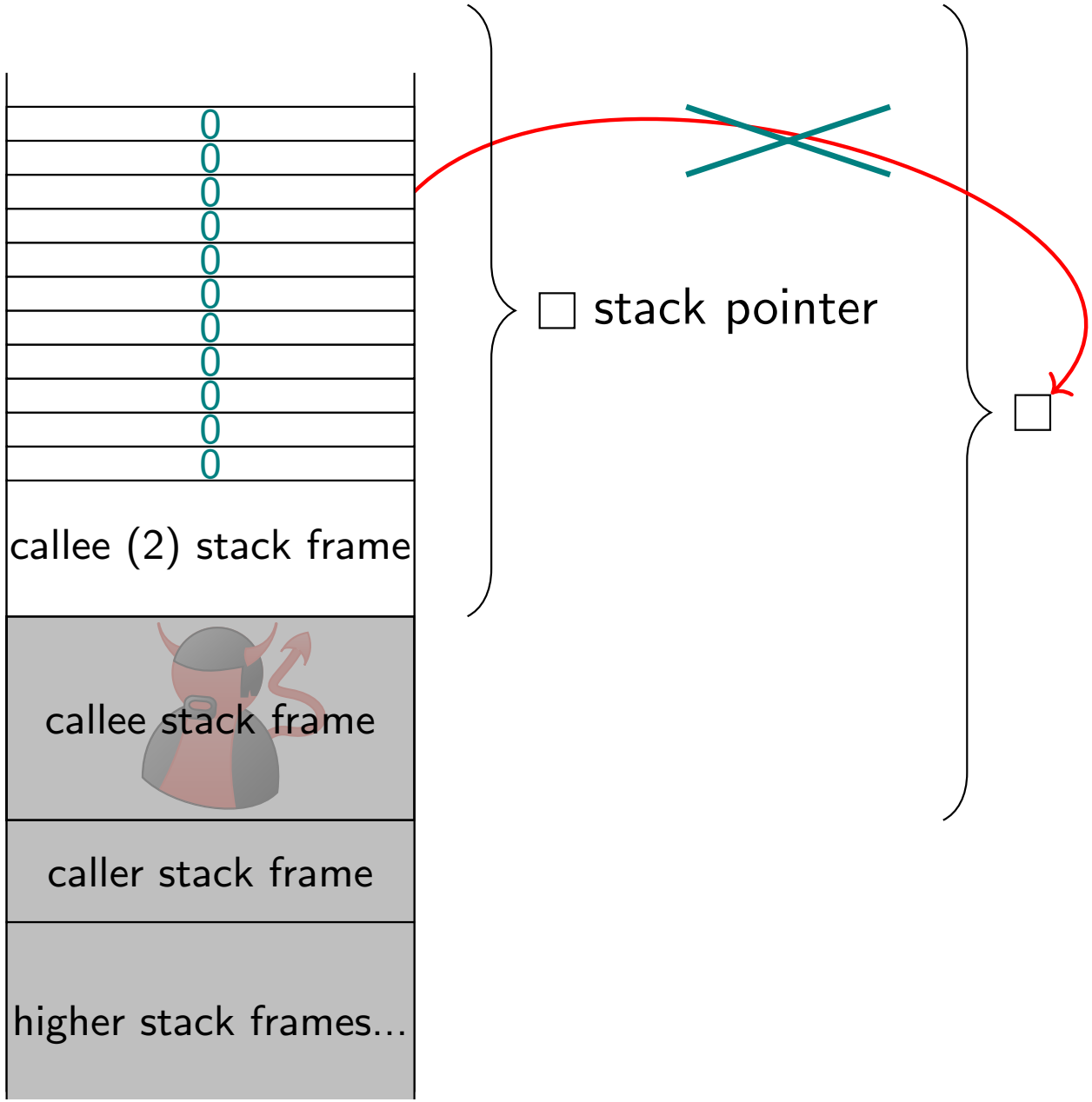
Stack Clearing Prevents Attack 2



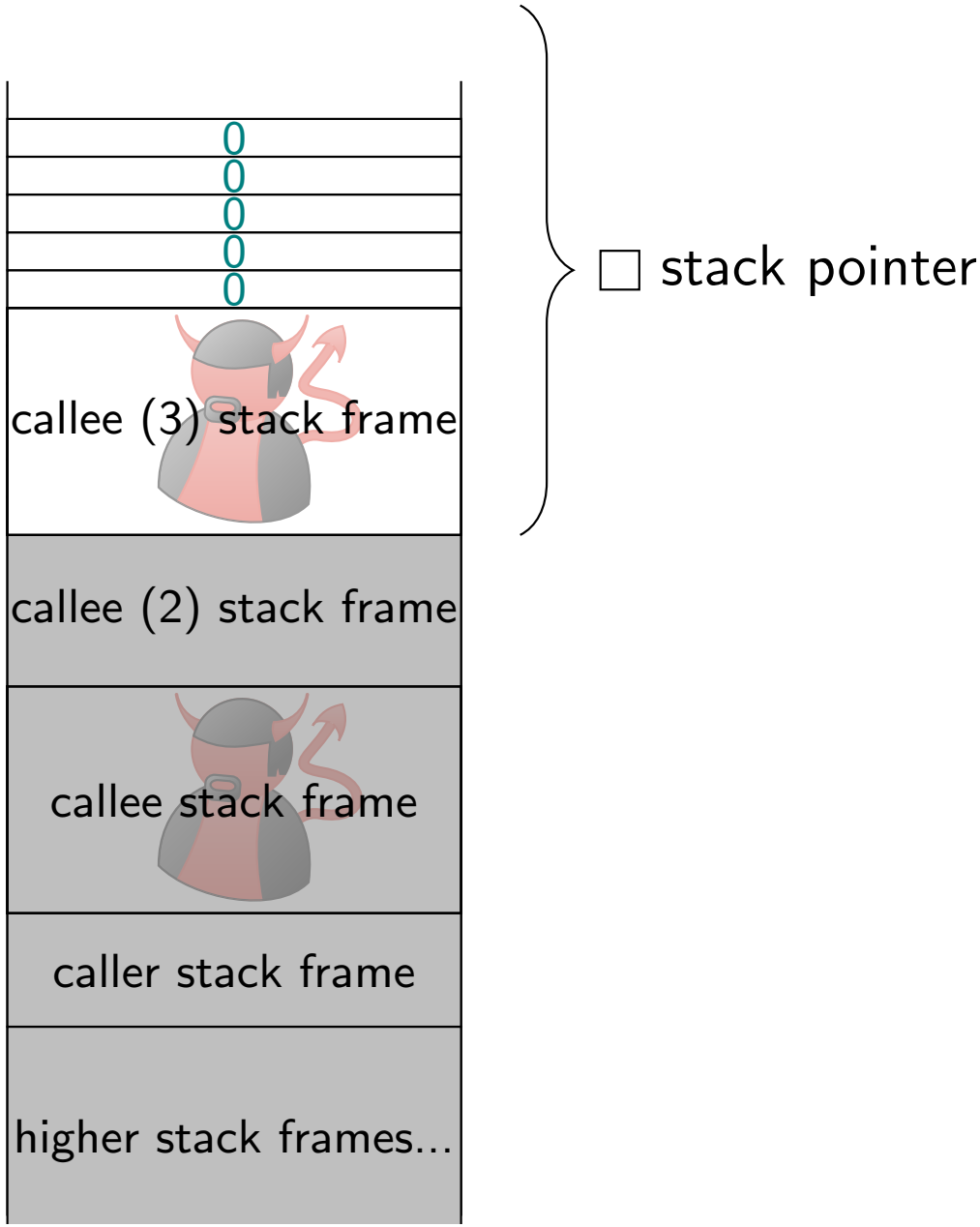
Stack Clearing Prevents Attack 2



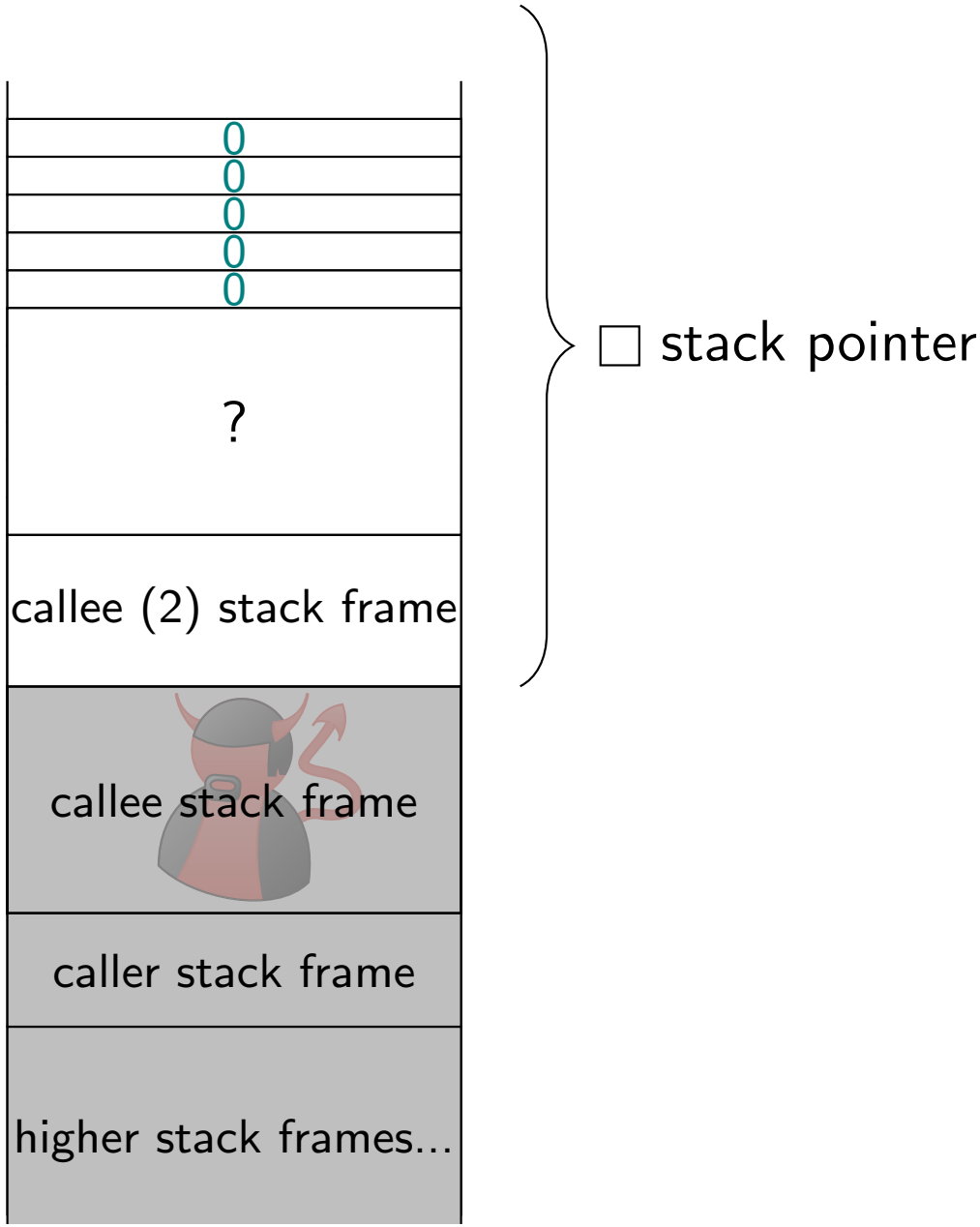
Stack Clearing Prevents Attack 2



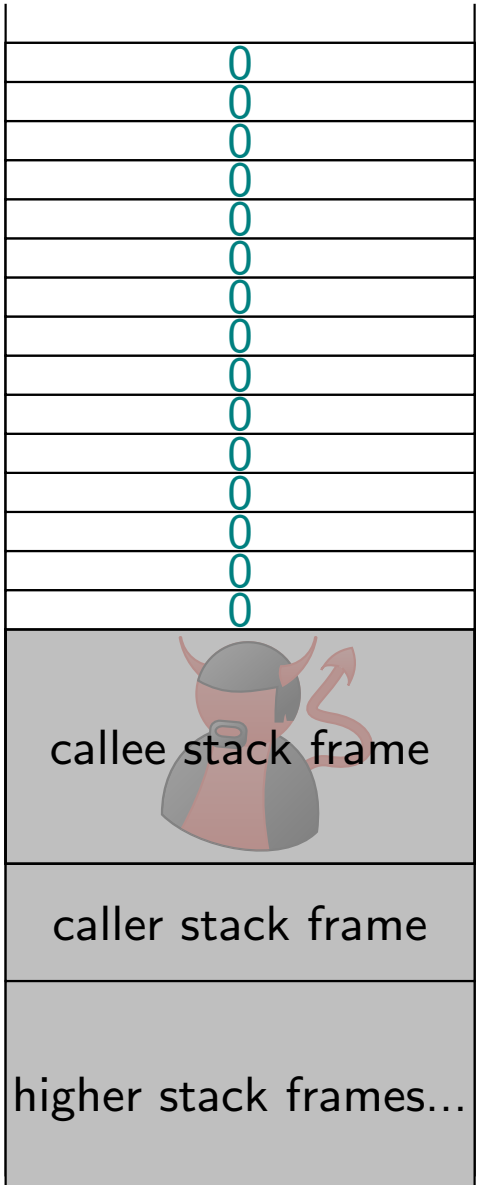
Stack Clearing Prevents Attack 2



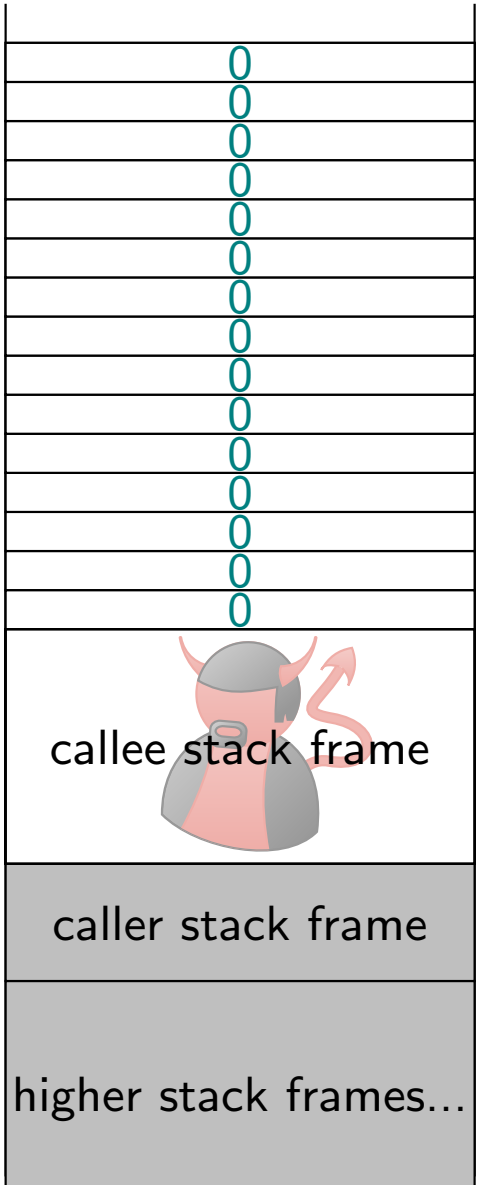
Stack Clearing Prevents Attack 2



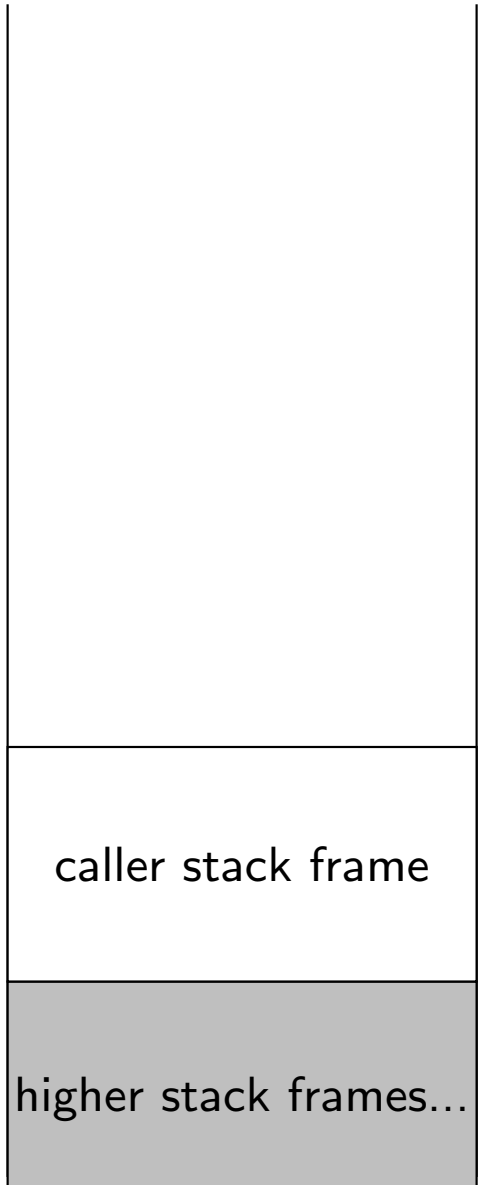
Stack Clearing Prevents Attack 2

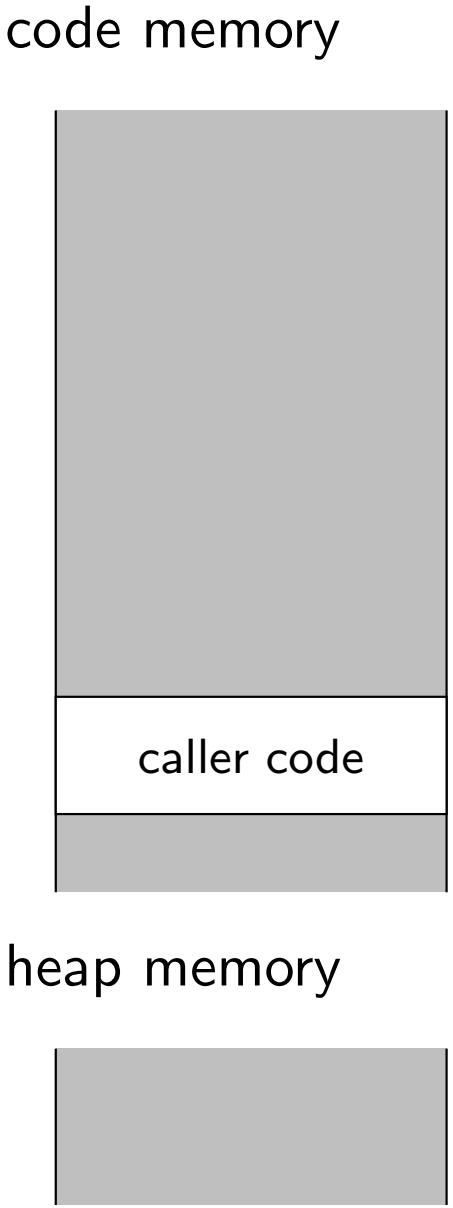


Stack Clearing Prevents Attack 2

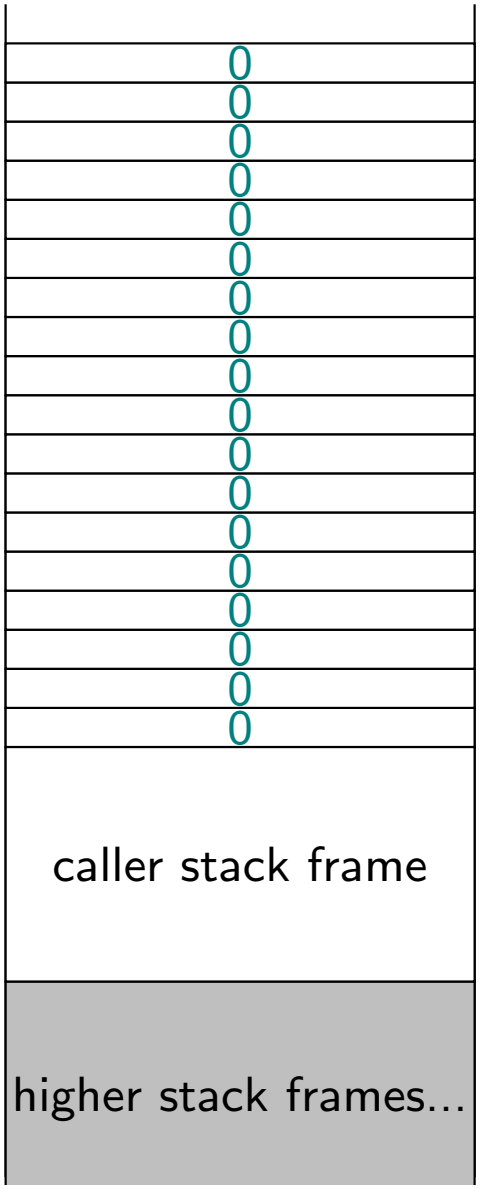


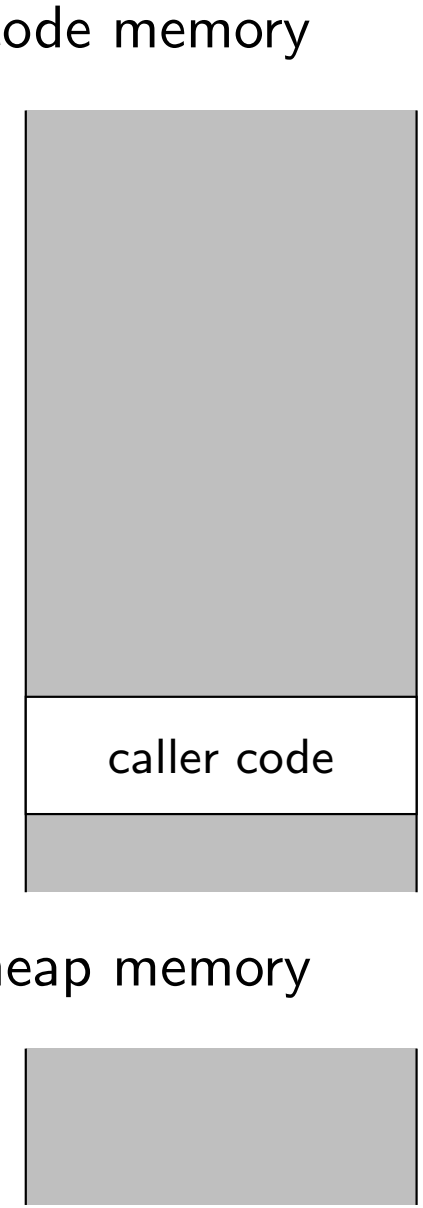
Local Stack Capabilities Prevent Attack 3



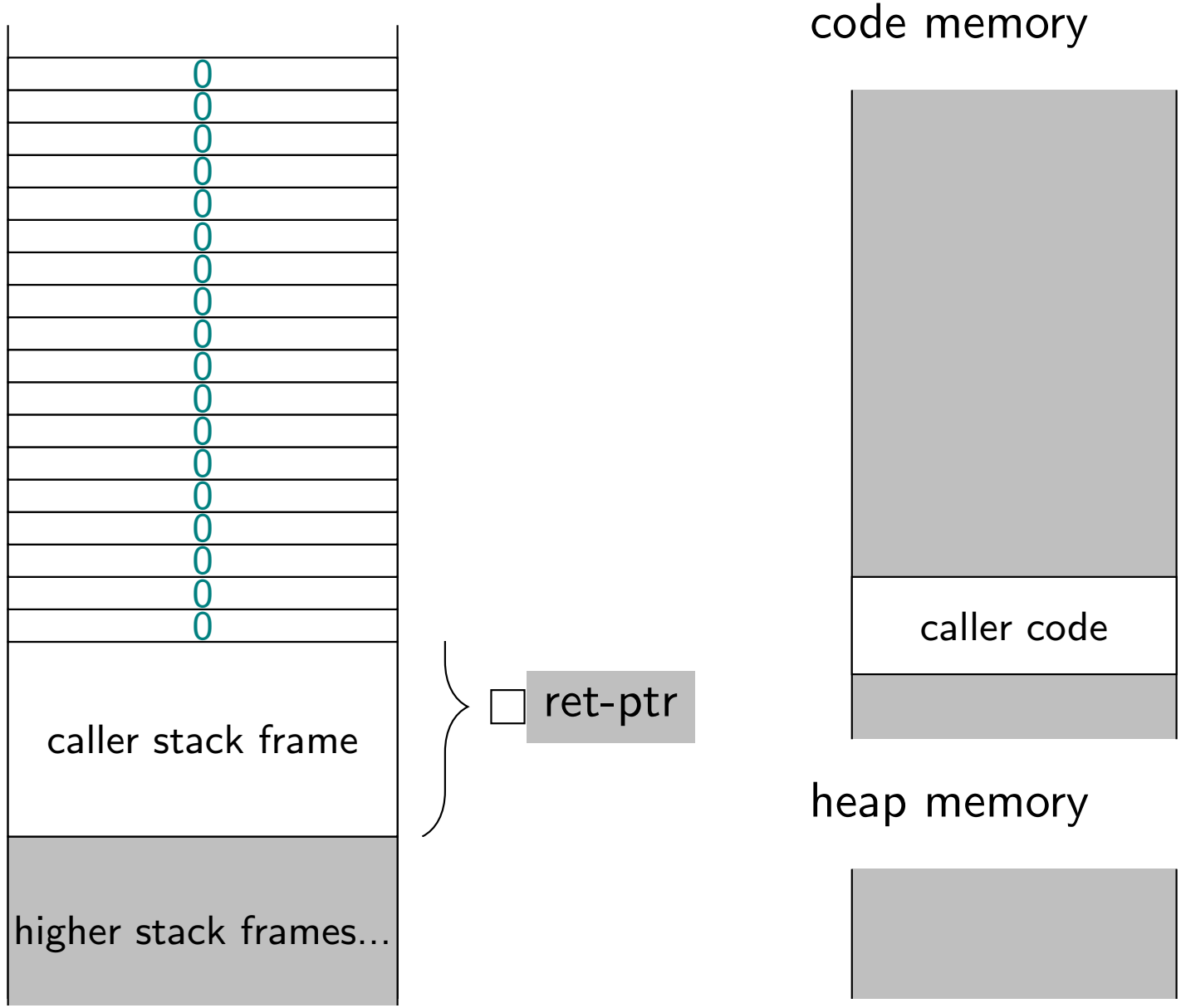


Local Stack Capabilities Prevent Attack 3

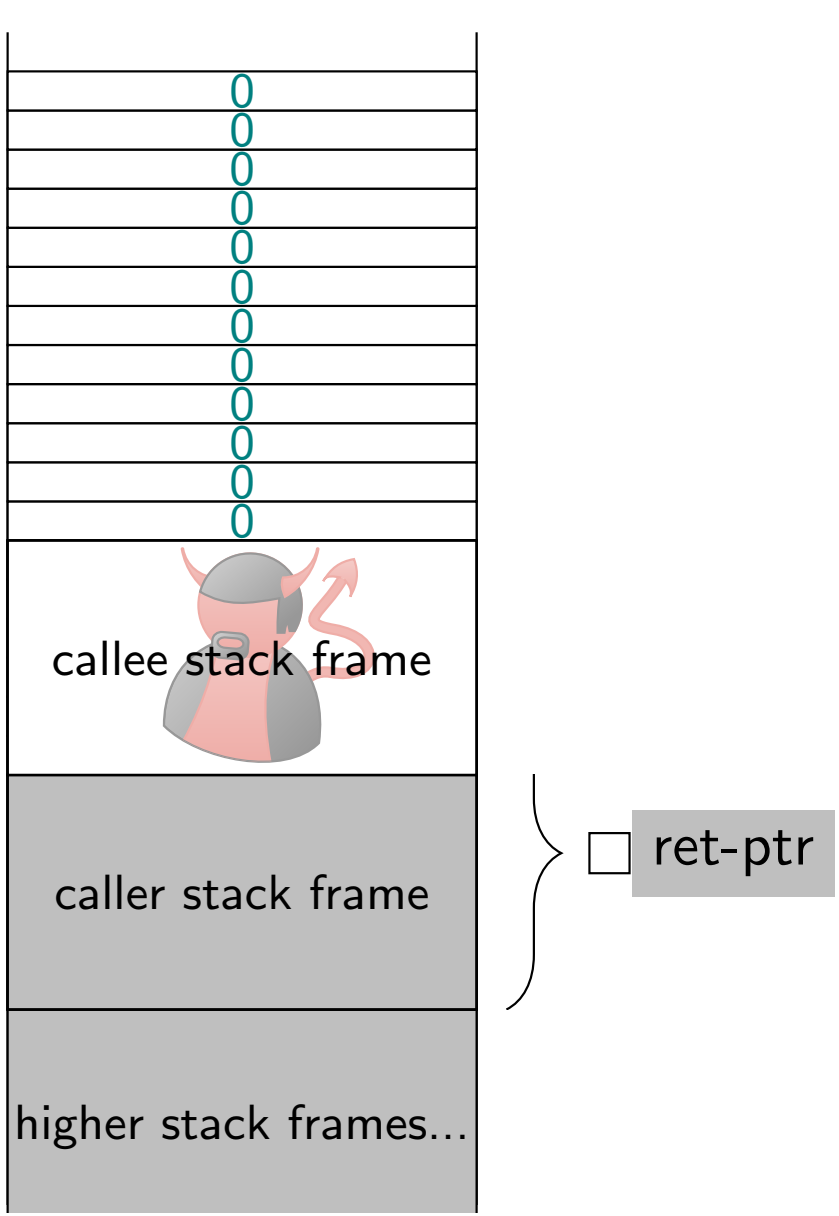




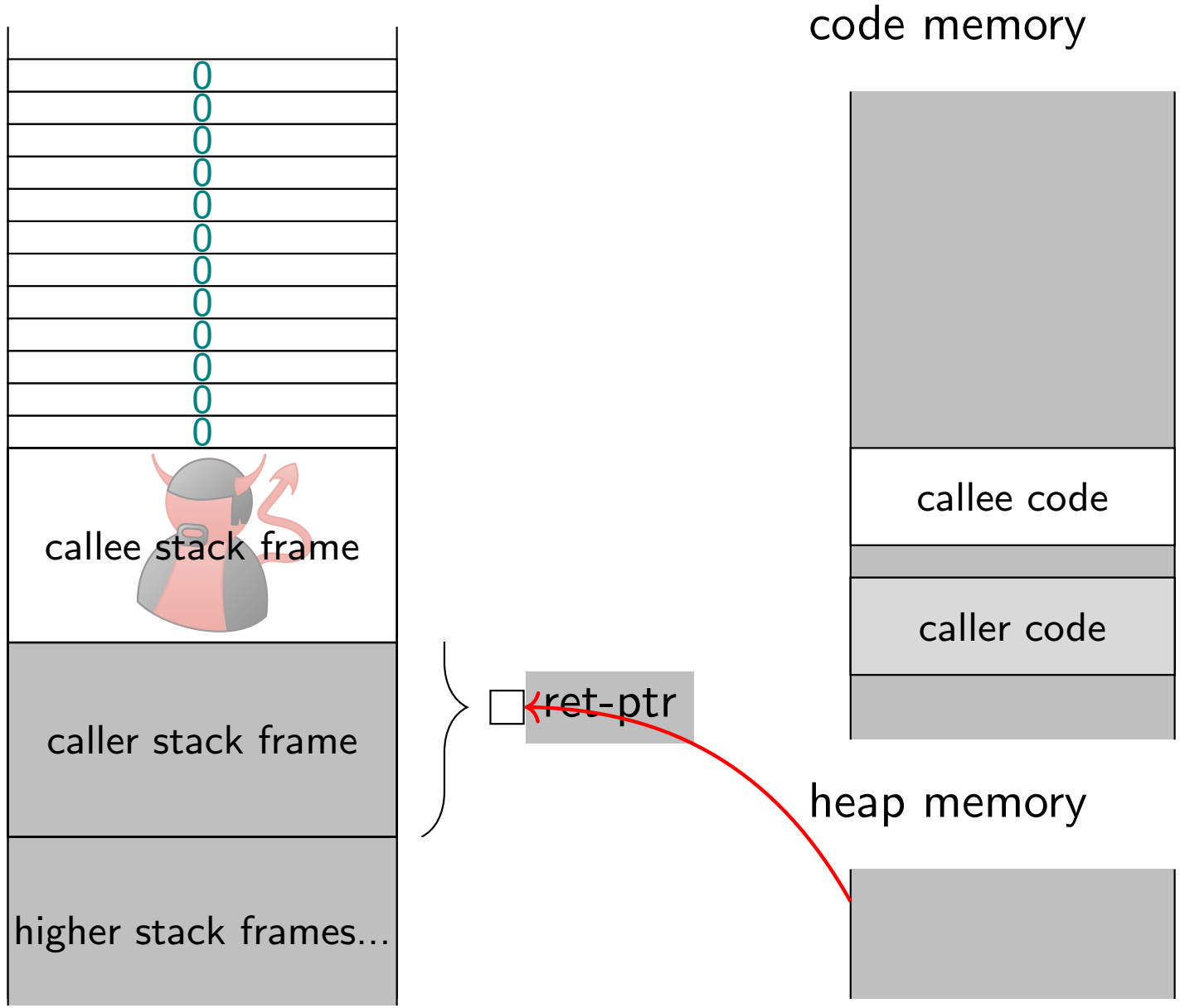
Local Stack Capabilities Prevent Attack 3



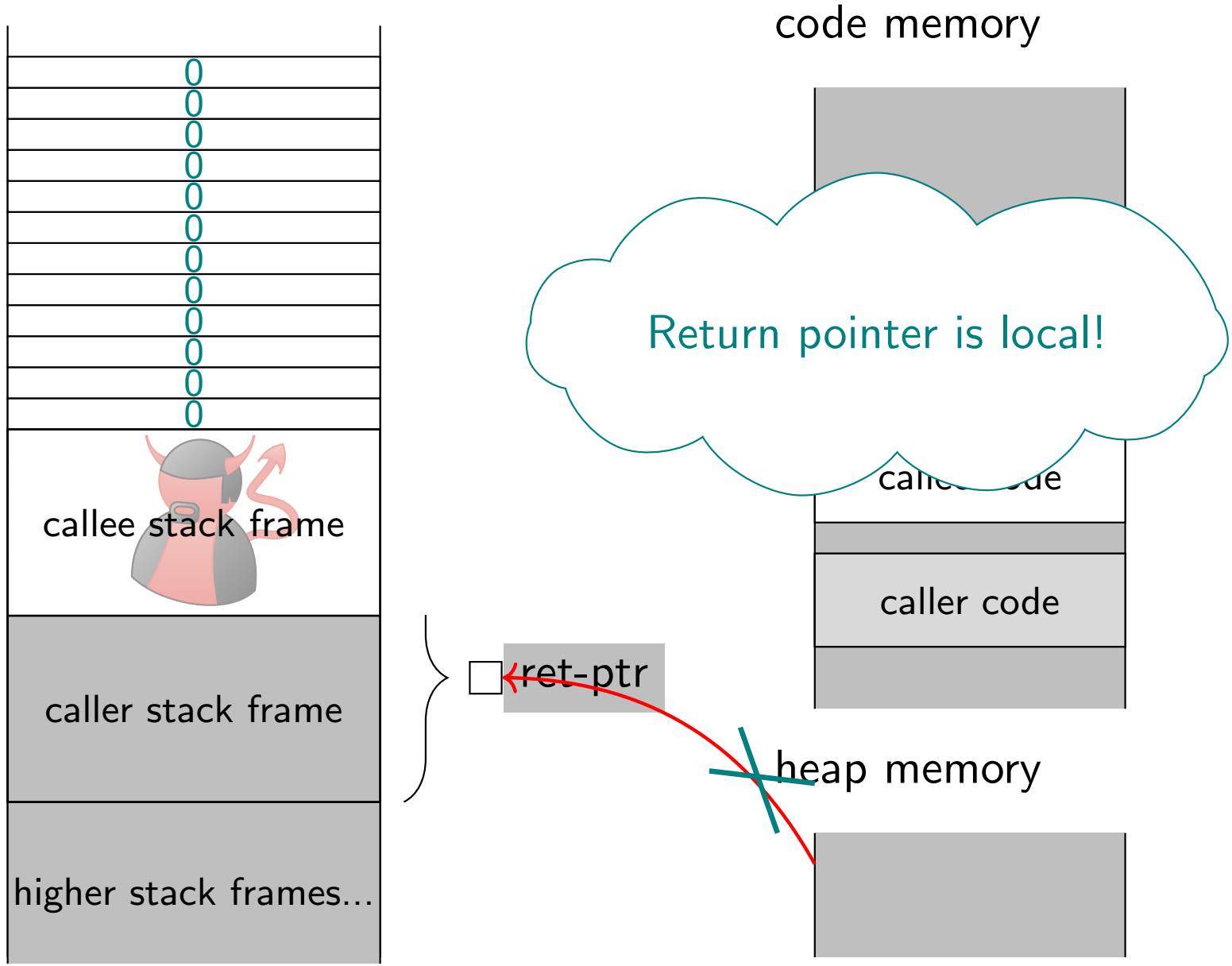
Local Stack Capabilities Prevent Attack 3



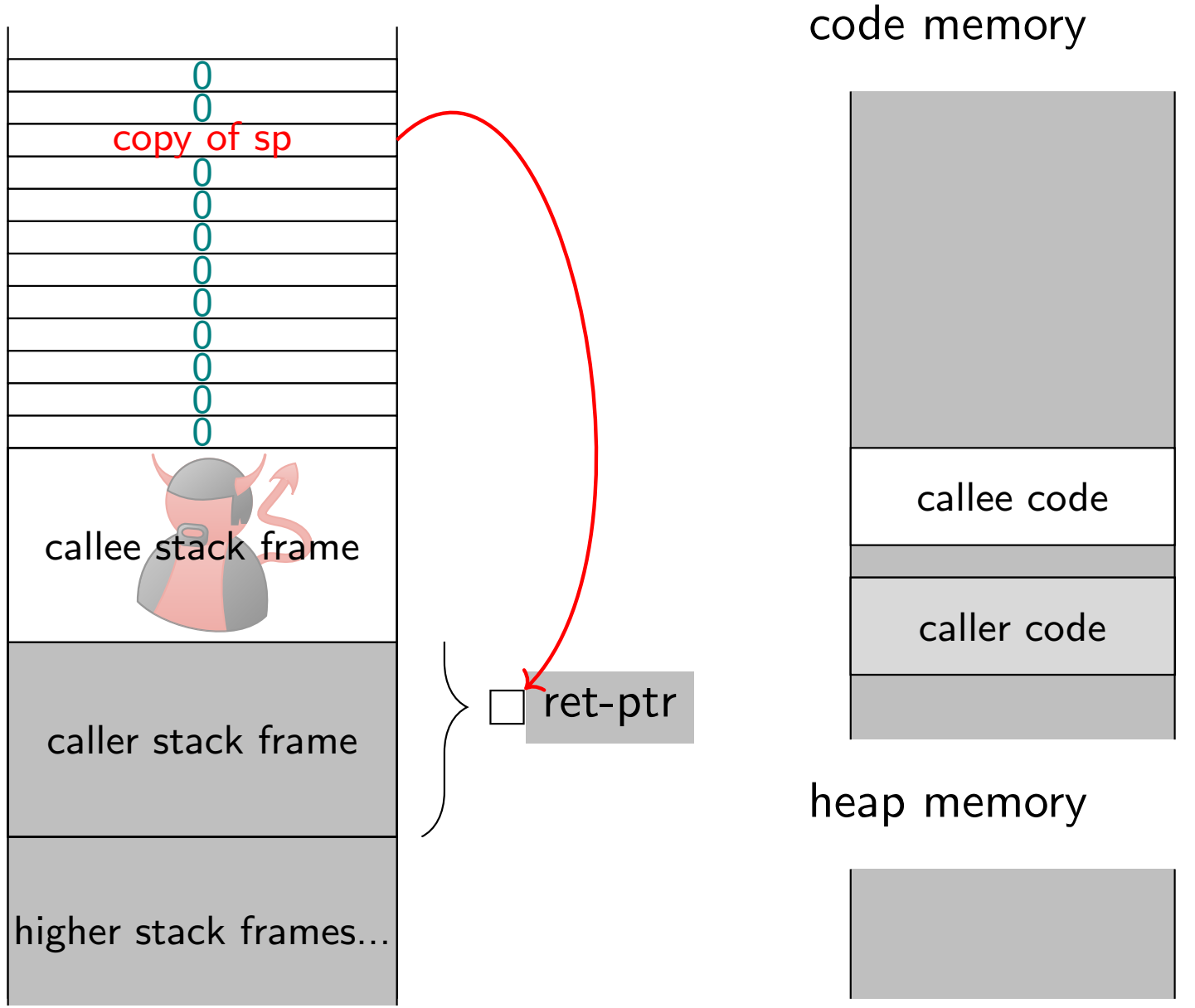
Local Stack Capabilities Prevent Attack 3



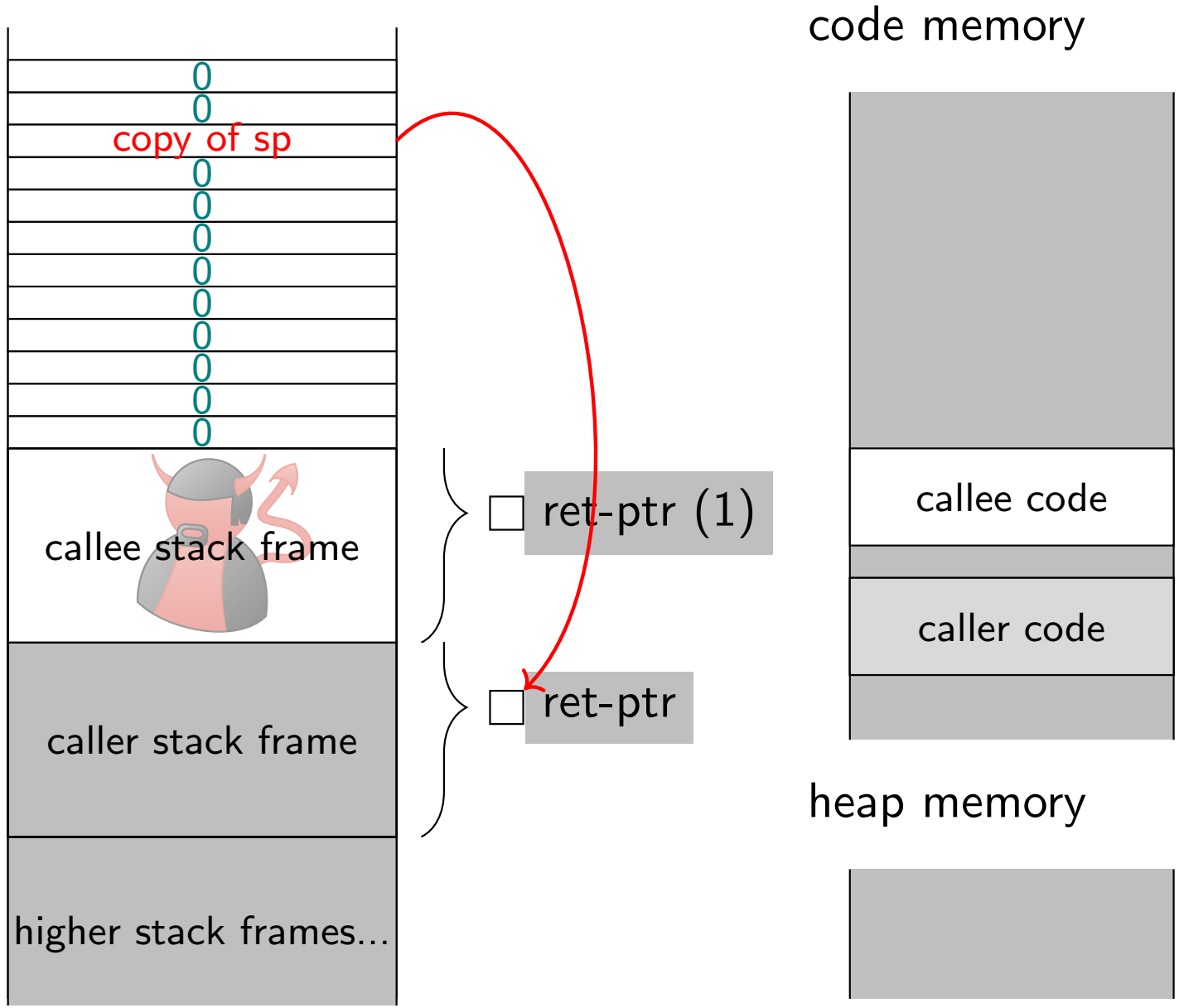
Local Stack Capabilities Prevent Attack 3



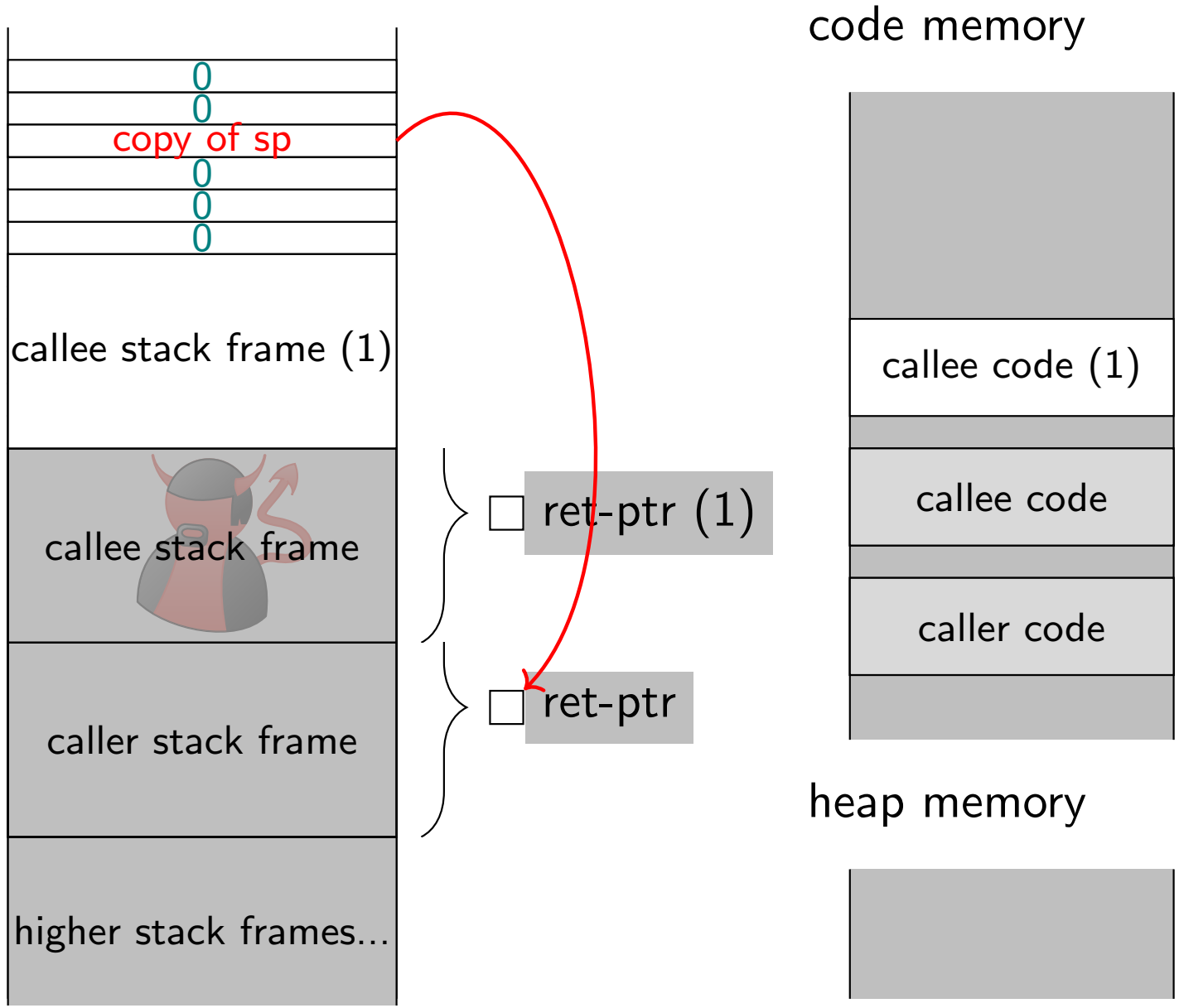
Local Stack Capabilities Prevent Attack 3



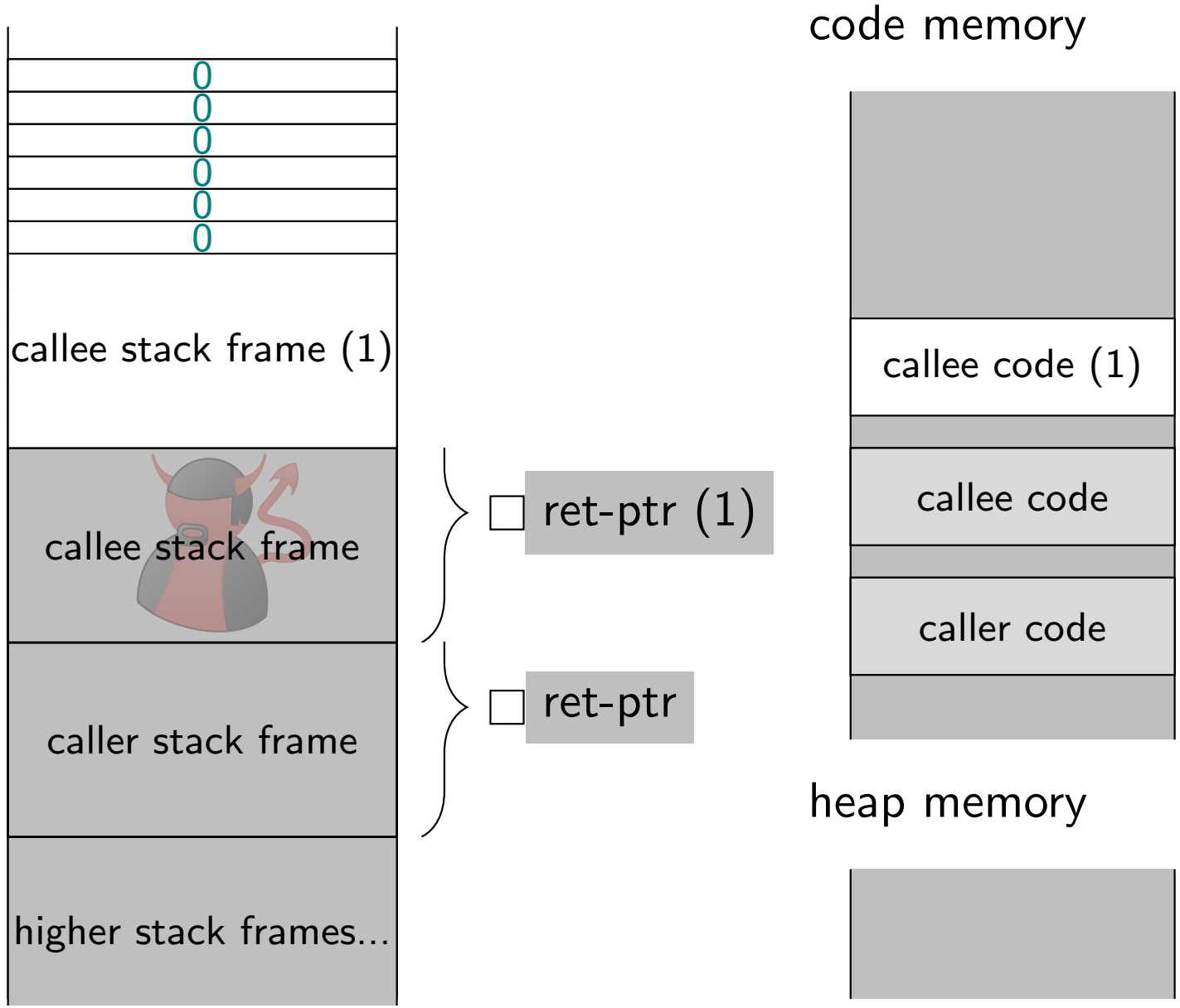
Local Stack Capabilities Prevent Attack 3



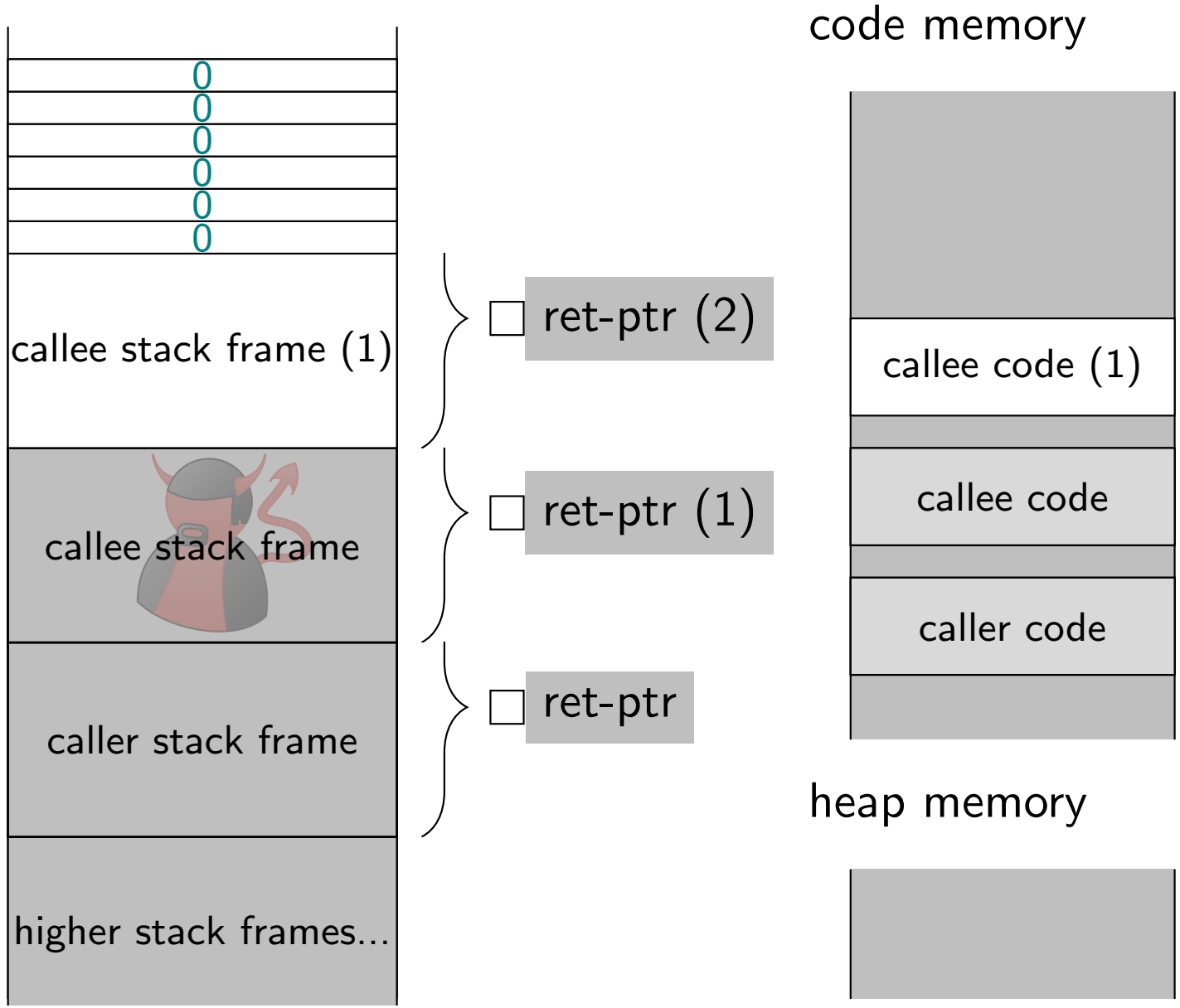
Local Stack Capabilities Prevent Attack 3



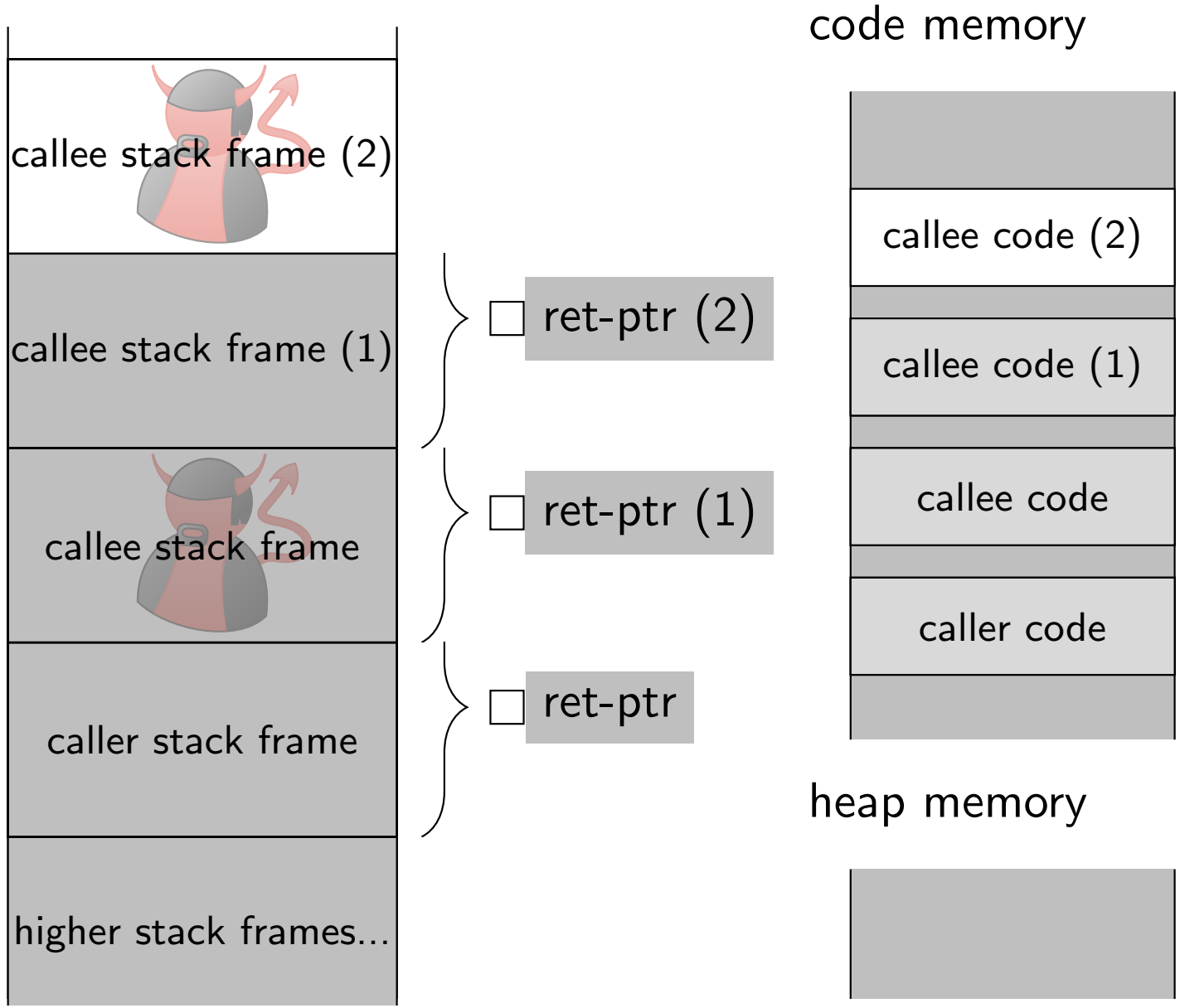
Local Stack Capabilities Prevent Attack 3



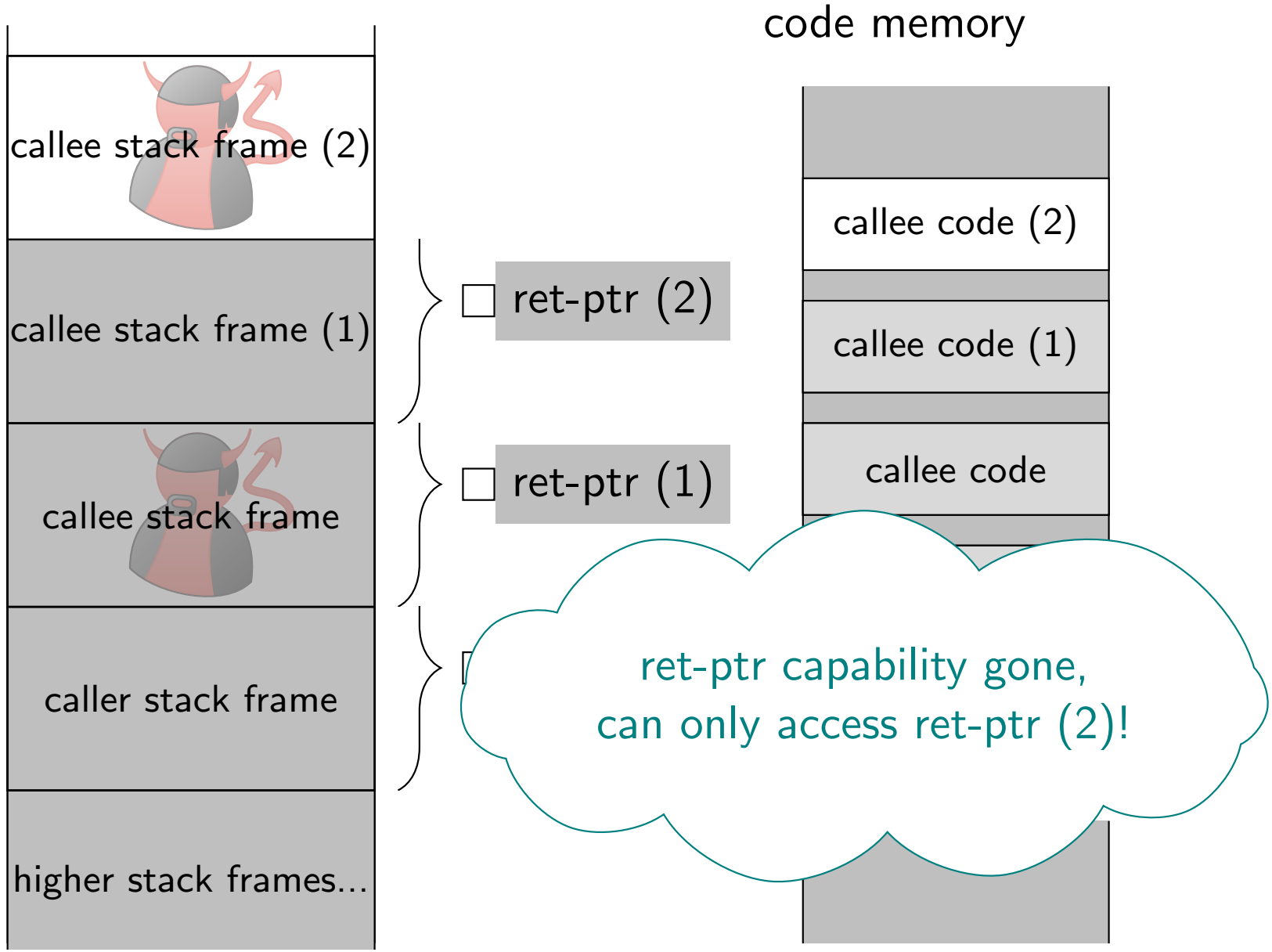
Local Stack Capabilities Prevent Attack 3



Local Stack Capabilities Prevent Attack 3



Local Stack Capabilities Prevent Attack 3



Contributions

- Formalisation of a capability machine with local capabilities
- Novel calling convention for enforcing control-flow correctness and local-state encapsulation
- A calling convention utilizing local capabilities to ensure well-bracketed control-flow and local-state encapsulation
- Step-indexed Kripke Logical Relation to reason about programs on said machine
  - Single orthogonal closure
  - Variant of public/private future world relation to express nature of local capabilities
- FTLR
  - Semantic statement of guarantees offered by capability machine
- Logical relation applied to a number of examples
  - "Awkward example"