HARVARD
School of Engineering
and Applied Sciences

λshill

A Secure Shell Scripting Language

Scott Moore, Christos Dimoulas, Dan King, and Stephen Chong

Can you advise me how to recognize if it’s safe or how to set it’s limited rights in the system?

2

I downloaded this shell script from this site.

It's suspiciously large for a bash script. So I opened it with text editor and noticed that behind the code there is a lot of non-sense characters.

I'm afraid of giving the script execution right with `chmod +x jd.sh`. Can you advise me how to recognize if it's safe or how to set it's limited rights in the system?

thank you

security shell sh chmod access-rights

share | improve this question

asked Nov 25 '11 at 12:25
xralf
2,126 7 31 81

add a comment

Principle of Least Privilege

> “Every program … should operate using the least amount of privilege necessary to complete the job.”
>
> —*Saltzer [CACM ’74]*

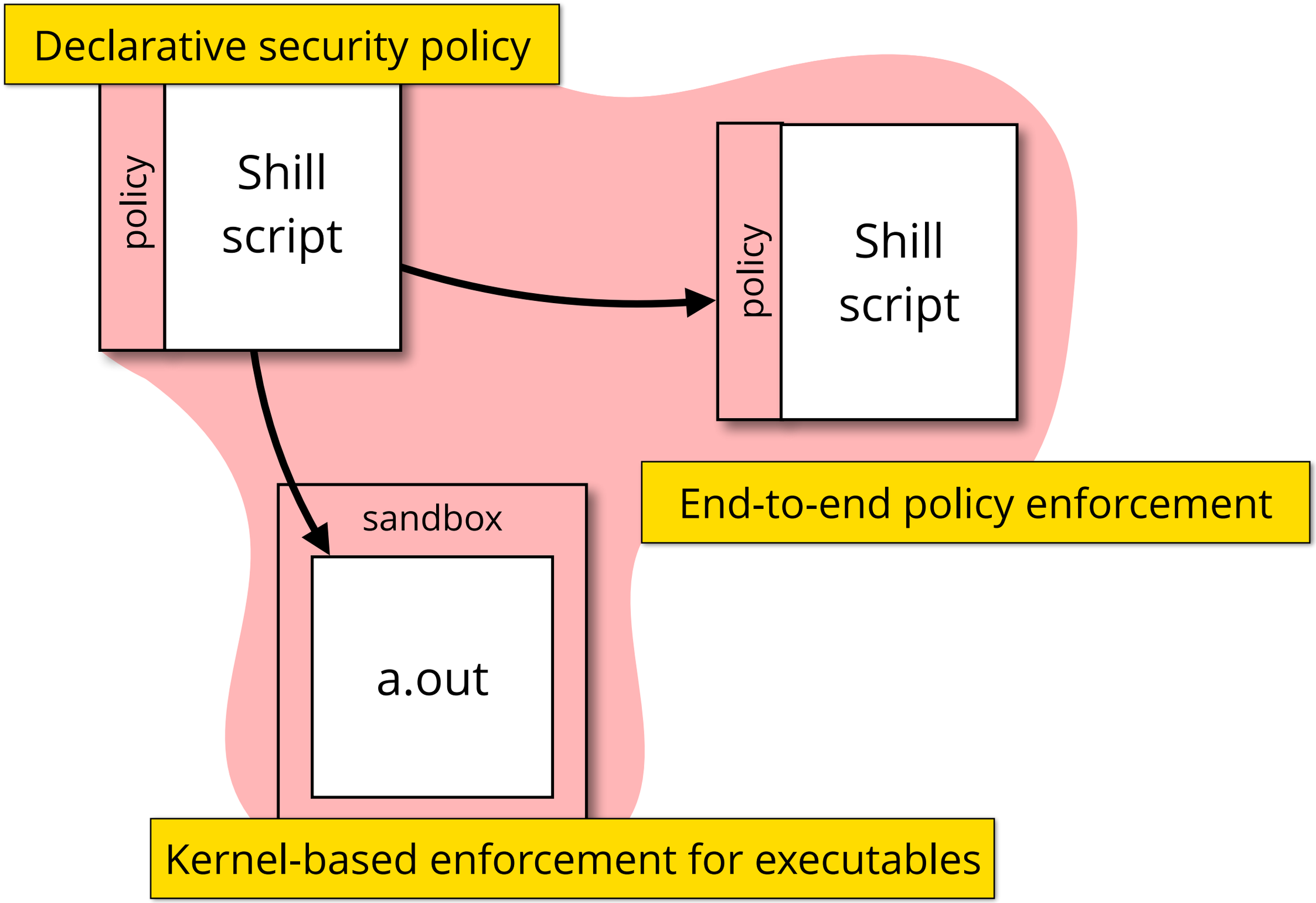
Principle of Least Privilege

> “Every program … should operate using the least amount of privilege necessary to complete the job.”
>
> —*Saltzer [CACM ’74]*

How do we limit the authority of scripts?

How do we determine what authority is necessary?

shill


Declarative security policy
policy
Shill script
policy
Shill script
End-to-end policy enforcement
sandbox
a.out
Kernel-based enforcement for executables

Capabilities to manifest authority

+

Contracts to communicate authority

+

Contracts and *sandboxes* to control authority

=

Scripting with Least Privilege

Capabilities

Capability-based security

A *capability* is an unforgeable token of authority

EROS

Coyotos

PDP-1 Supervisor

KeyKOS

seL4

CAL-TSS

NTLSS

CapDesk

PLASH

Dennis and Van Horn

PSOS

Polaris

Capsicum

1960 1970 1980 1990 2000 2010

Google Caja

W7

Joe-E

Oz-E

CaPerl

Emily

A capability-safe script

```
copy = fun(from_dir,to_dir) {
    for entry in contents(from_dir) do {
        current = lookup(from_dir,entry);
        if is_file(current) then {
            new = create_file(to_dir,name);
            write(new,read(current))
        }
    }
}
```

A cap

capabilities: files, directories, pipes, sockets, ...

```
copy = fun(from_dir,to_dir) {
    for entry in contents(from_dir) do {
        current = lookup(from_dir,entry);
        if is_file(current) then {
            new = create_file(to_dir,name);
            write(new,read(current))
        }
    }
}
```

A capability-safe script

```
copy = fun(from_dir,to_dir) {
    for entry in contents(from_dir) do {
        current = lookup(from_dir,entry);
        if is_file(current) then {
            new = create_file(to_dir,name);
            write(new,read(current))
        }
    }
}
```

operations: require privileges on capabilities

Directories: `+contents, +lookup, +unlink, ...`

Files: `+read, +stat, +path, ...`

Sockets: `+bind, +send, +receive, ...`

A capability-safe script

derived capabilities

```
copy = fun(from_dir,to_dir) {
        in contents(from_dir) do {
          current = lookup(from_dir,entry);
          if is_file(current) then {
            new = create_file(to_dir,name);
            write(new,read(current))
          }
      }
}
```

Directories: `+contents, +lookup, +unlink, ...`

Files: `+read, +stat, +path, ...`

Sockets: `+bind, +send, +receive, ...`

A capability-safe script

```
copy = fun(from_dir,to_dir) {
    for entry in contents(from_dir) do {
        current = lookup(from_dir,entry);
        if is_file(current) then {
            new = create_file(to_dir,name);
            write(new,read(current))
        }
    }
}
```

Capability safety

All resources are represented by capabilities

Scripts have no capabilities by default

Contracts

Software contracts

Applying “Design by Contract”,
Bertrand Meyer [IEEE Computer ’92]

Software contracts in Shill

executable specifications

```
provide find_c :
  { dir : is_dir }
  → listof(is_filename);
```

```
find_c = fun(dir) {
  filter(fun (entry) has_ext("c",entry),
         contents(dir));
}
```

Contract violations stop execution at runtime

Compositional reasoning about scripts

Capability contracts in Shill

More precise contract specifying +contents privilege

```
provide find_c :
  { dir : is_dir(+contents) }
  → listof(is_filename);
```

```
find_c = fun(dir) {
  filter(fun (entry) has_ext("c",entry),
         contents(dir));
}
```

Capability contracts both
require and **restrict** privileges

Sandboxing

Capability-based sandboxing

unknown binary

```
/bash

...
./a.out $test &> $log
...
```

Need to enforce security on executables!

Emulate capabilities with Mandatory Access Control

Capability-based sandboxing

File capability with +exec

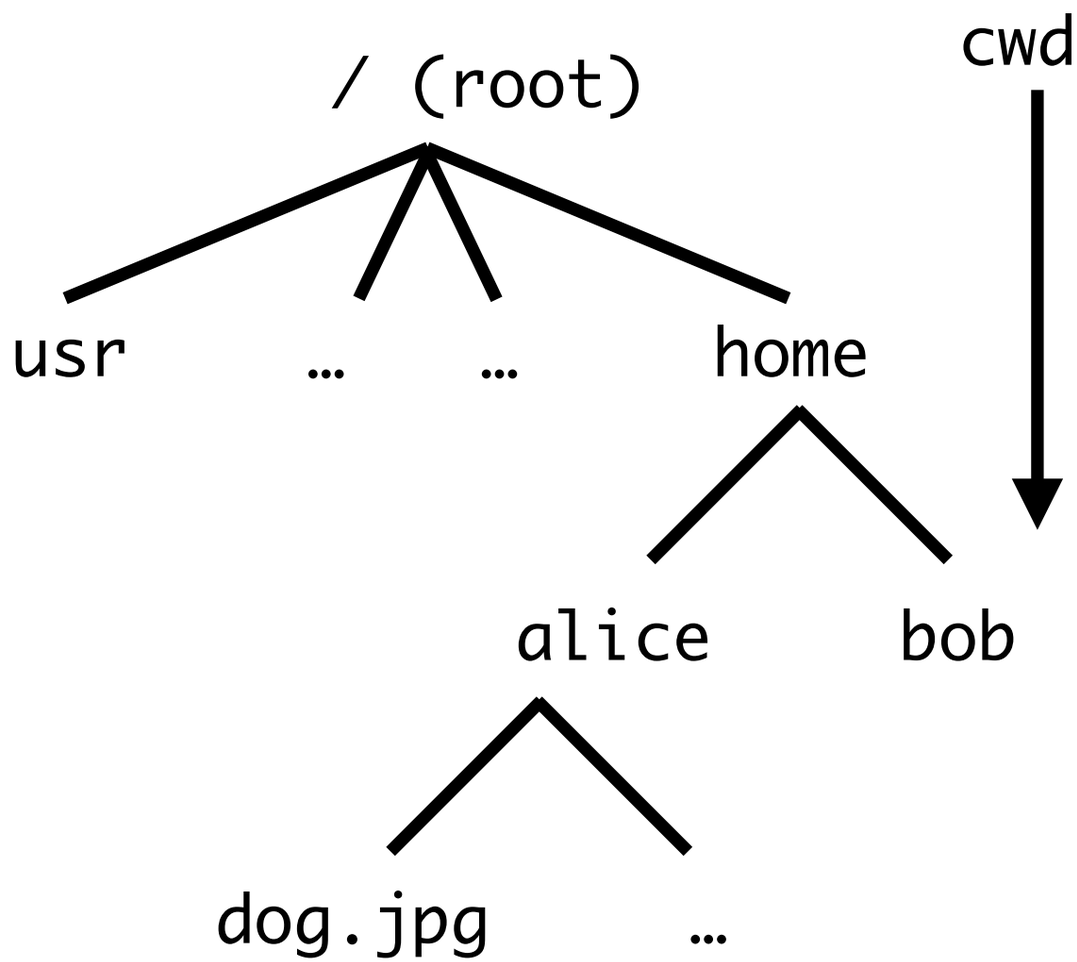
arguments

capabilities for stdin/out/err

```
exec(aout,[“test.txt”],
     stdout = log, stderr = log,
     extra = [libc,cwd,...]);
```

additional capabilities for sandbox

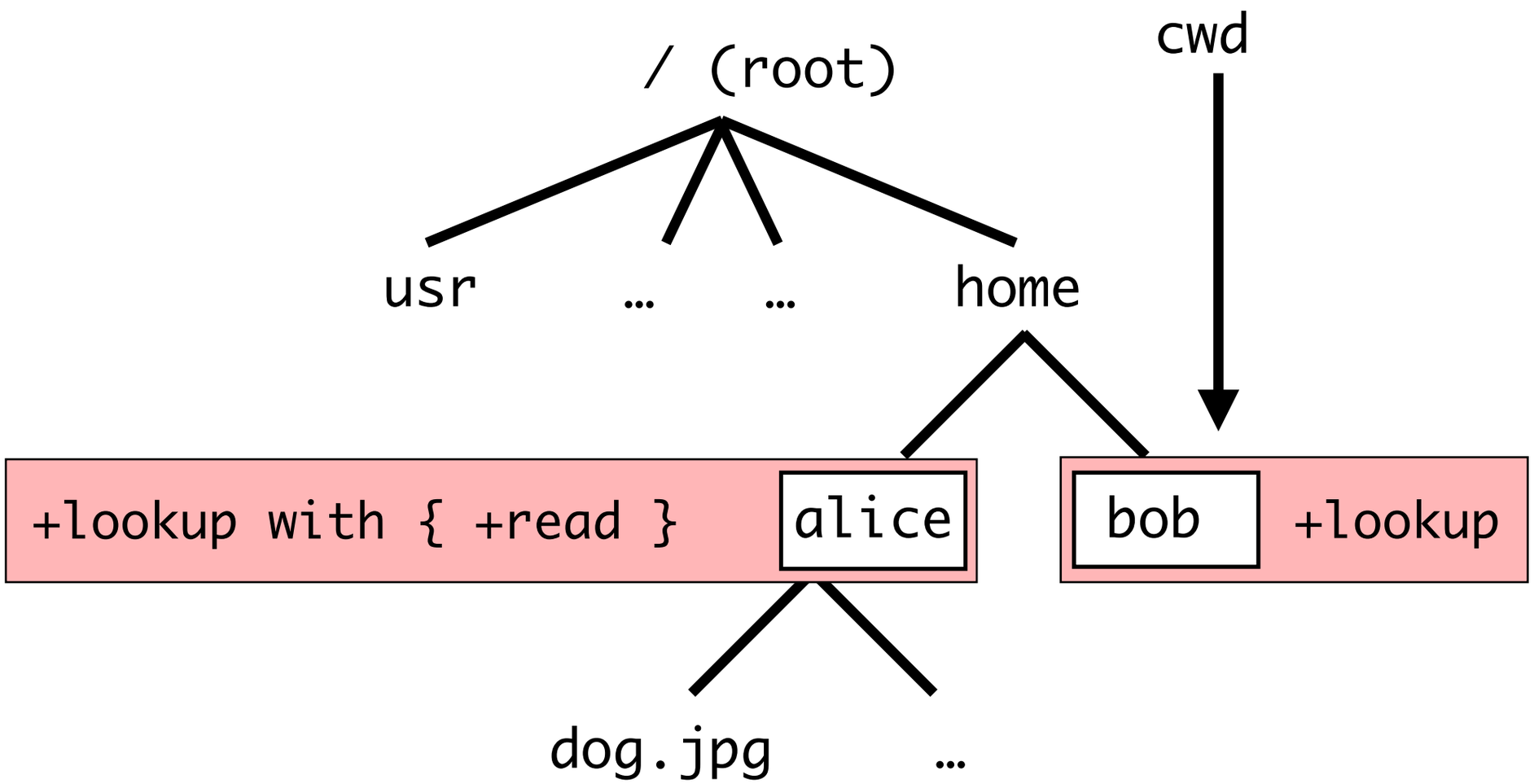
Mandatory Access Control Policy



foo.c

```
open("../alice/dog.jpg", O_RDONLY)
```

Label objects with capabilities

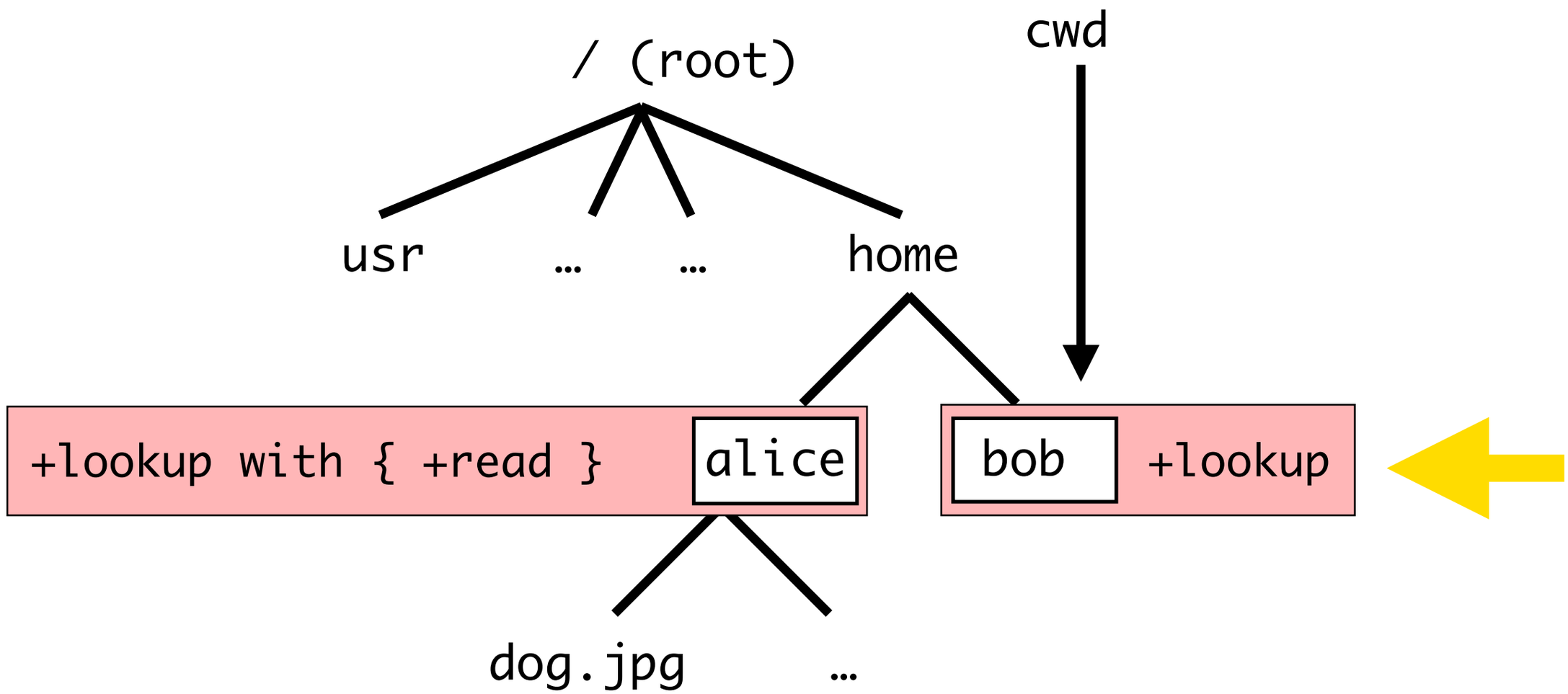


foo.c

```
open("../alice/dog.jpg", O_RDONLY)
```

Label objects with capabilities

Authorize actions against permissions

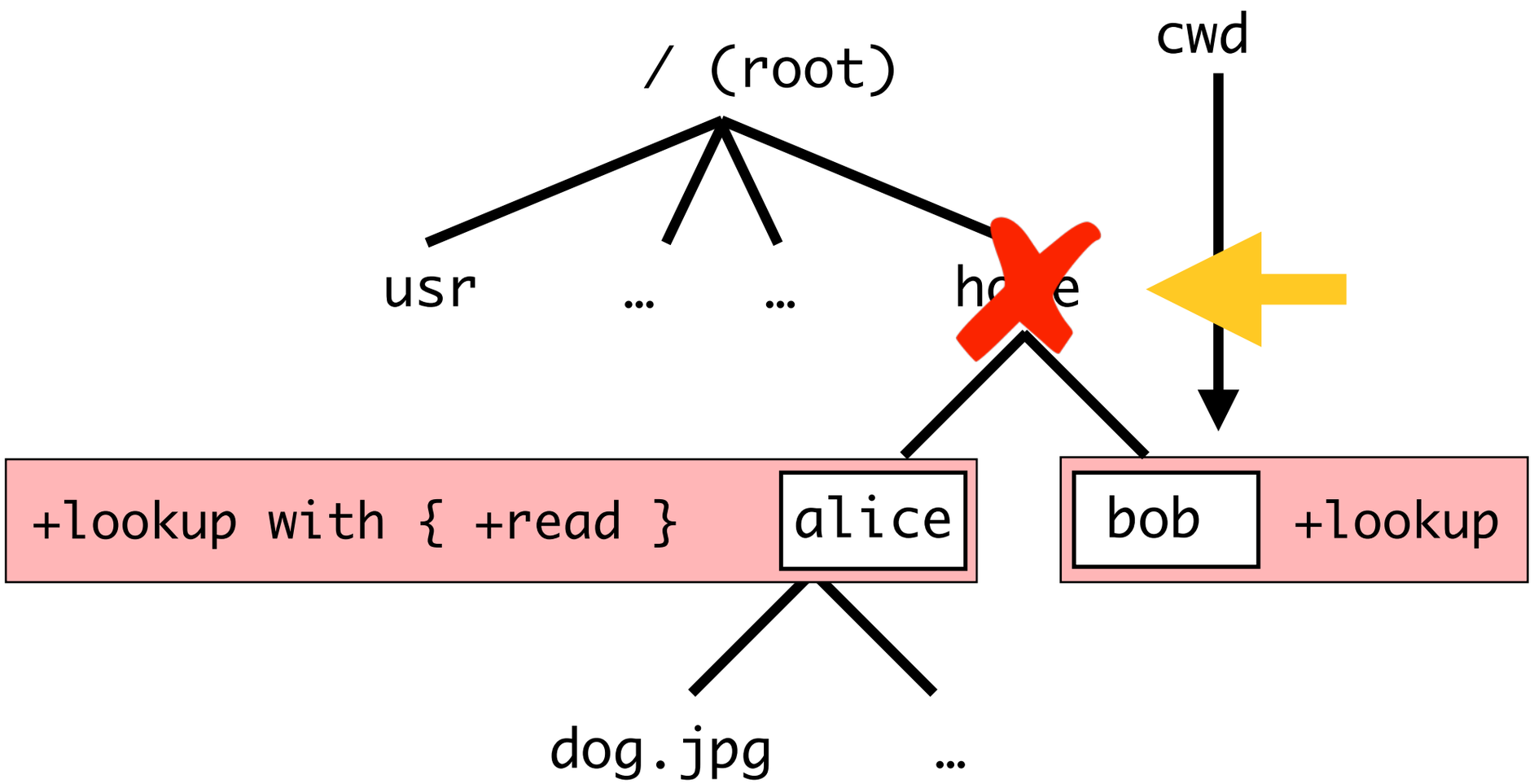


foo.c

```
open("../alice/dog.jpg", O_RDONLY)
```

Label objects with capabilities

Authorize actions against permissions



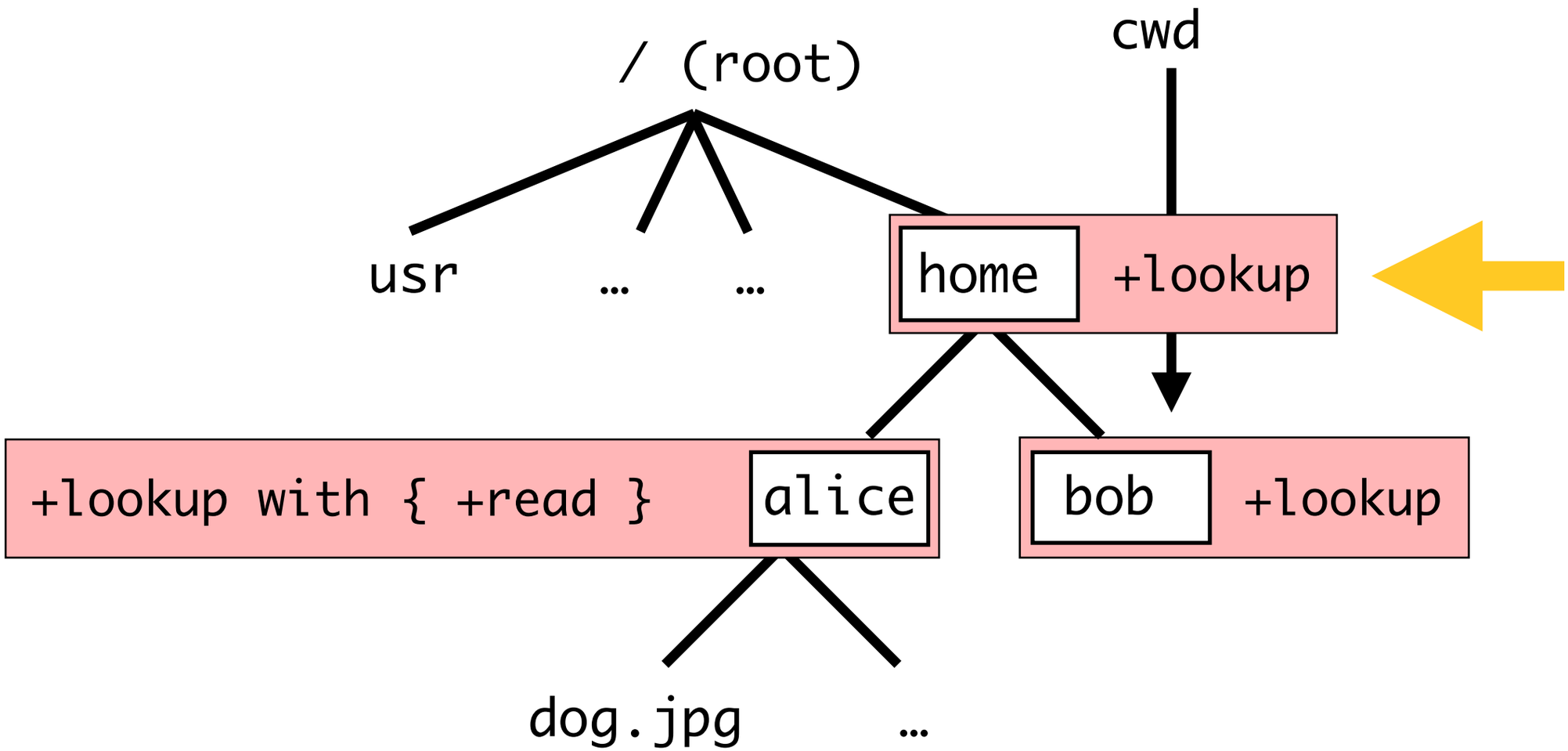
foo.c

```
open("../alice/dog.jpg", O_RDONLY)
```

licy

Label objects with capabilities

Authorize actions against permissions

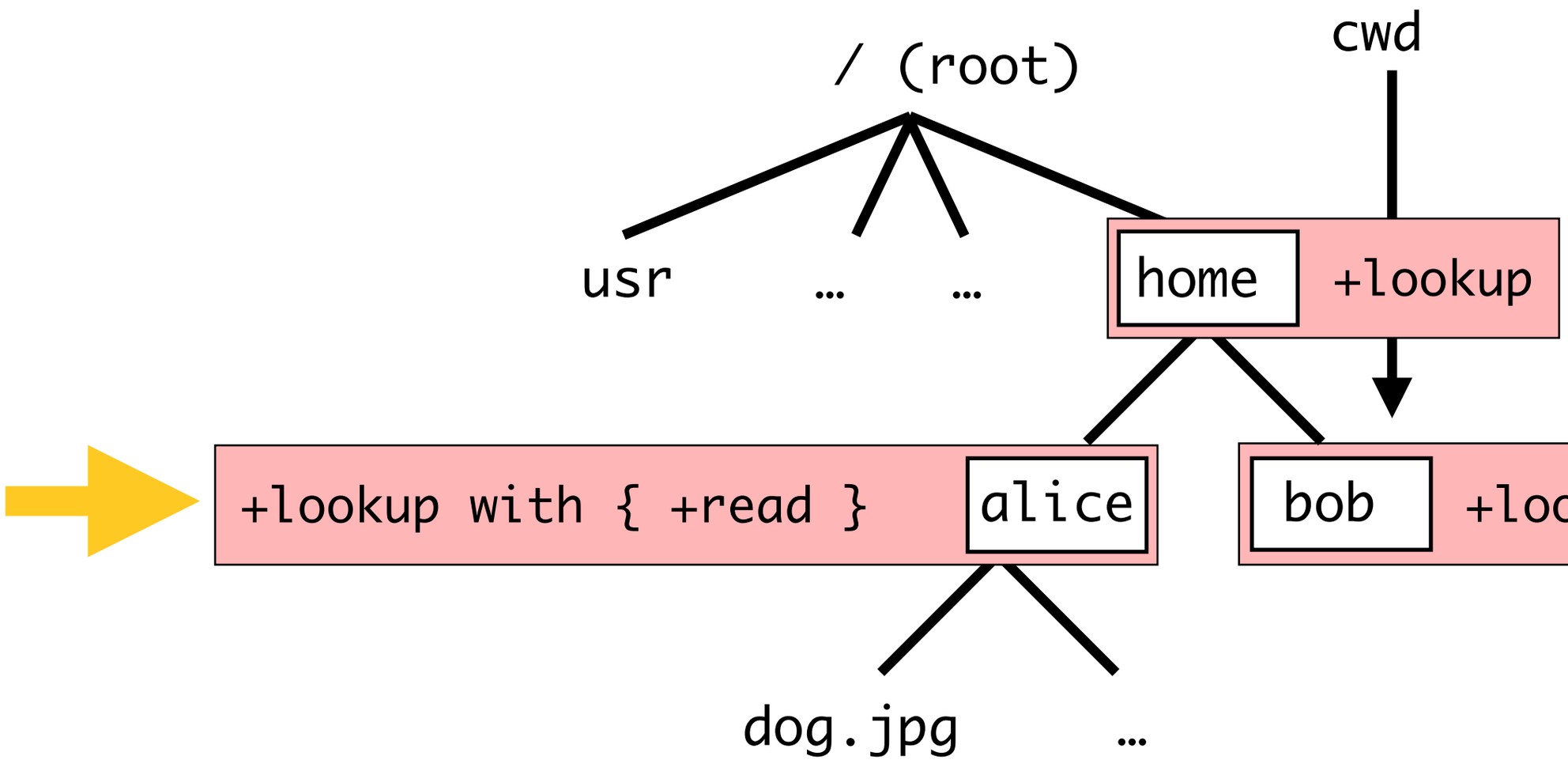


foo.c

```
open("../alice/dog.jpg", O_RDONLY)
```

Label objects with capabilities

Authorize actions against permissions



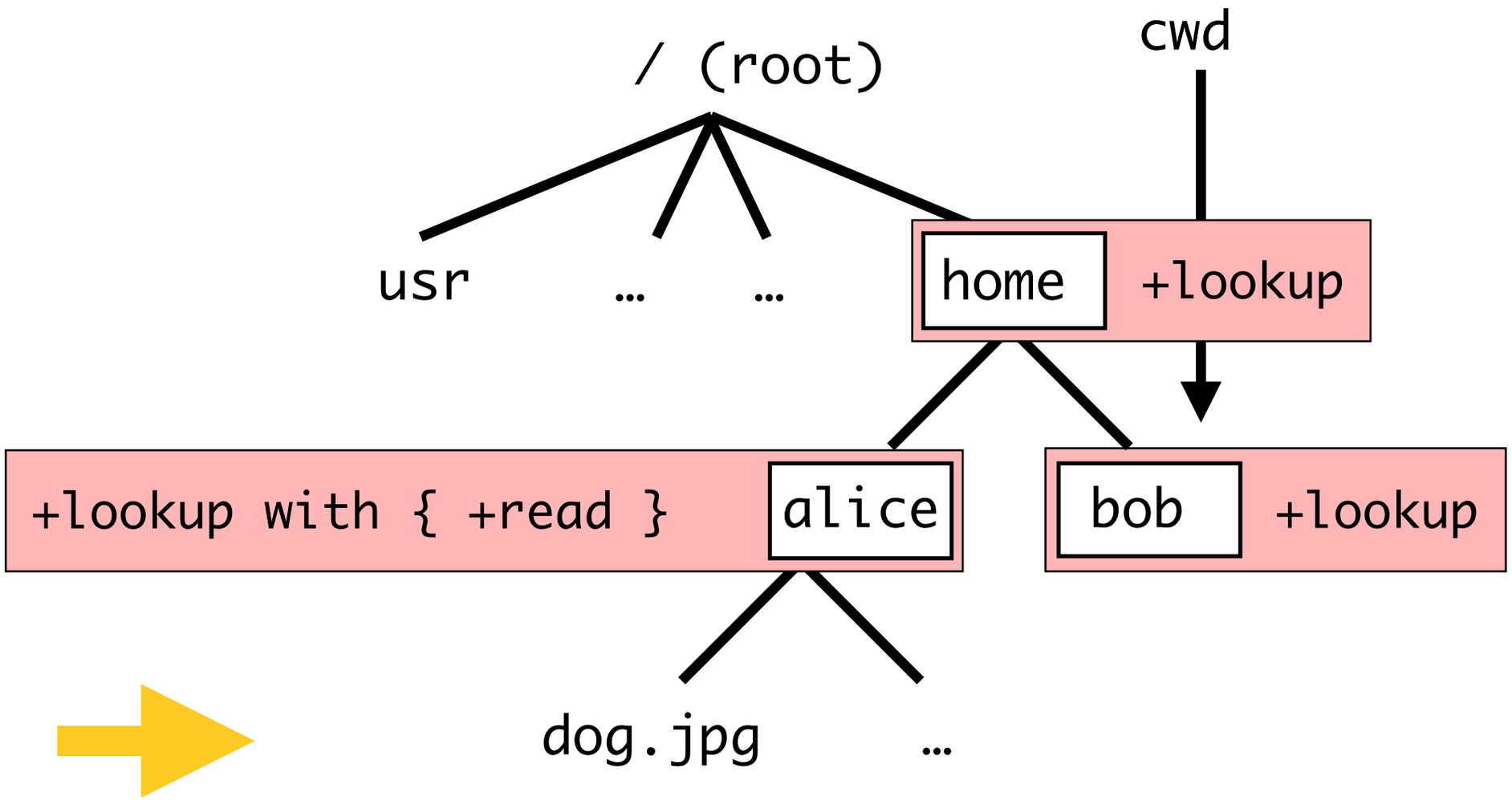
foo.c

```
open("../alice/dog.jpg", O_RDONLY)
```

licy

Label objects with capabilities

Authorize actions against permissions



foo.c

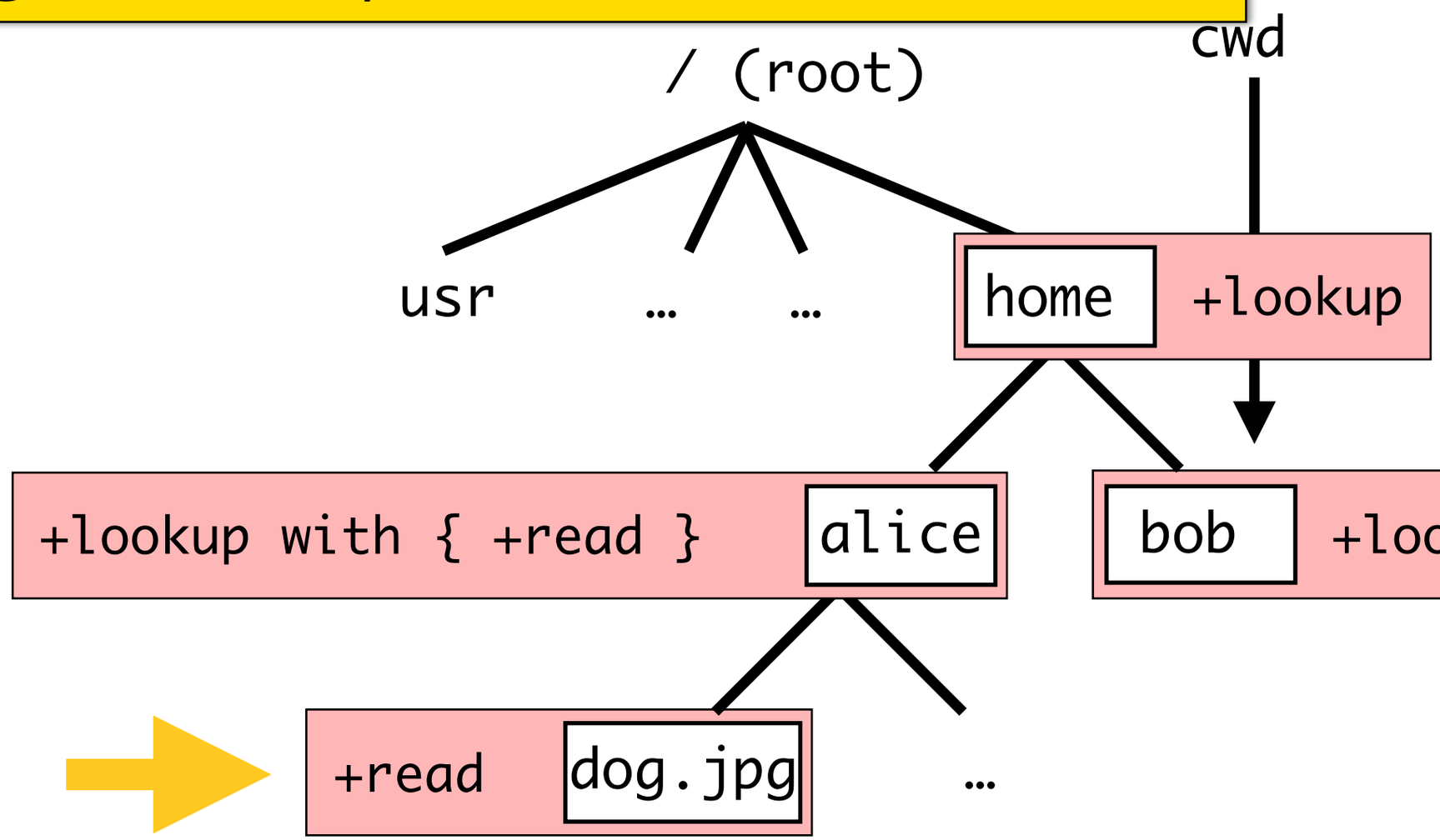
```
open("../alice/dog.jpg", O_RDONLY)
```

licy

Label objects with capabilities

Authorize actions against permissions

Propagate new capabilities



foo.c

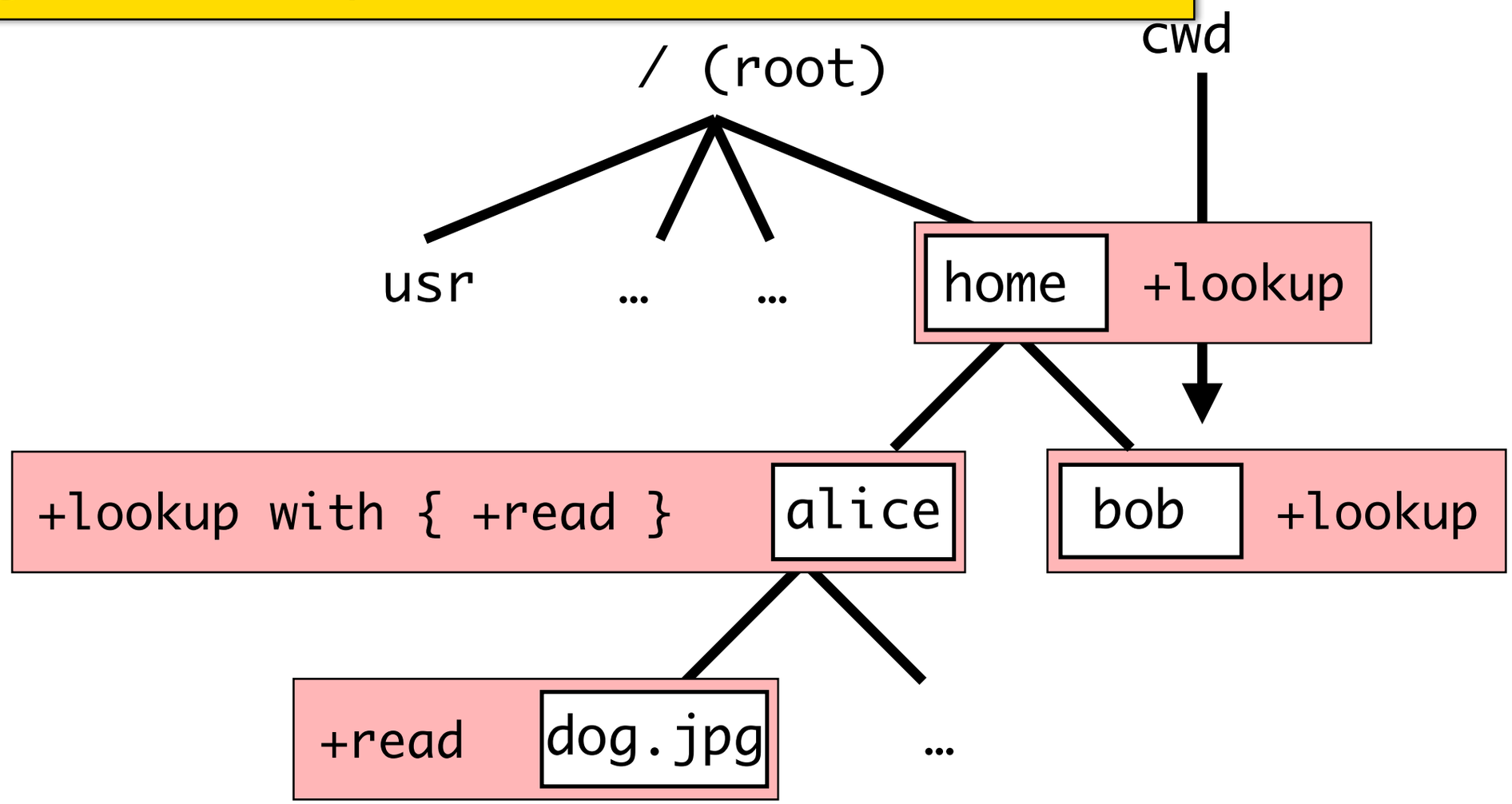
```
open("../alice/dog.jpg", O_RDONLY)
```

licy

Label objects with capabilities

Authorize actions against permissions

Propagate new capabilities



foo.c

```
open("../alice/dog.jpg", O_RDONLY)
```

Managing capabilities

Difficult to gather all the capabilities needed

```
cat foo.txt { /bin/cat
              foo.txt
              &stdout
              /libexec/ld-elf.so.1
              /var/run/ld-elf.so.hints
              /etc/libmap.conf
              /lib/libc.so.7
              /usr/share/locale/en_US.UTF-8
              /usr/share/locale/UTF-8/LC_CTYPE
```

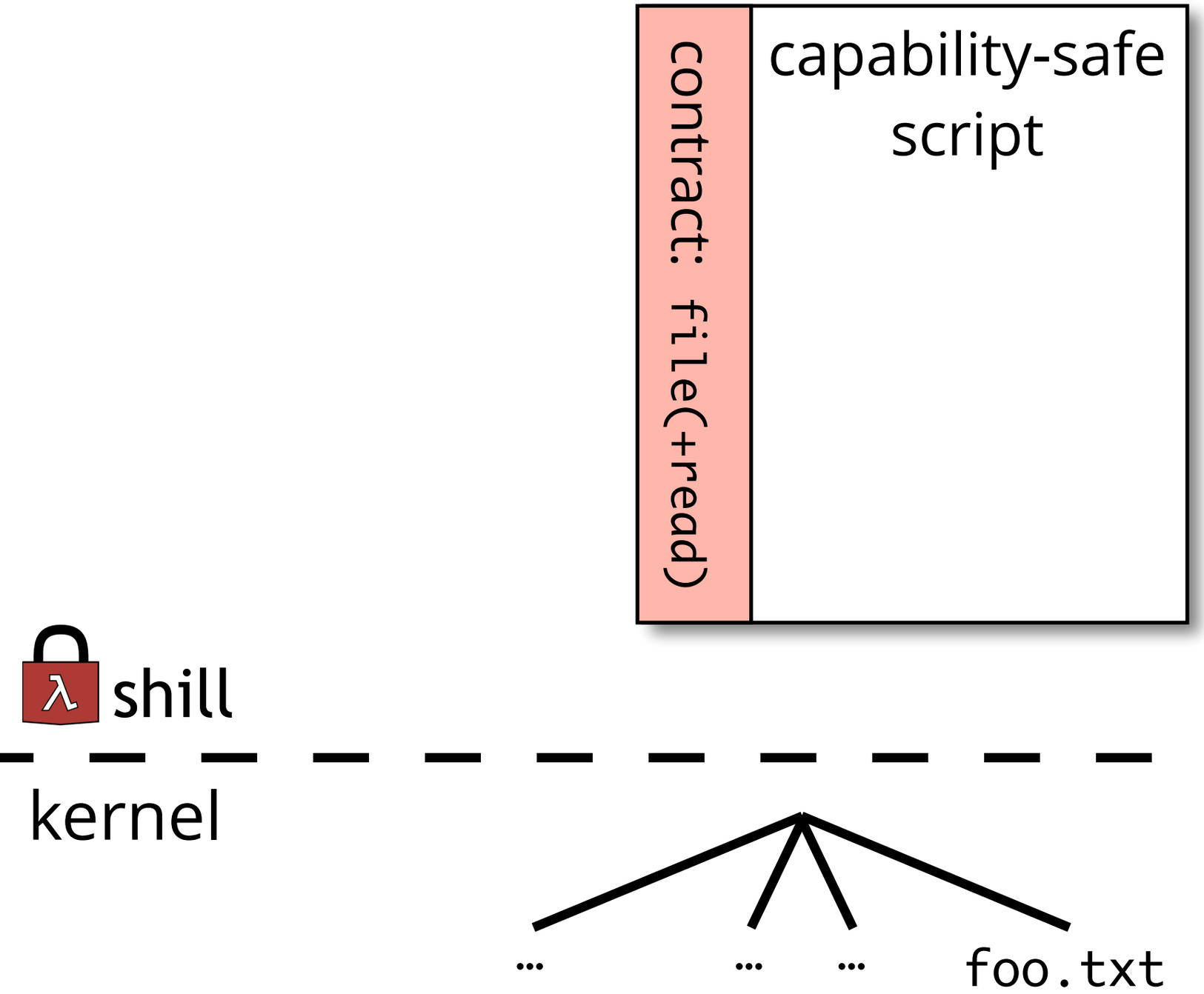
Need abstractions: ***capability wallets***

- package sets of capabilities with contracts
- standard library for running binaries

```
populate-native-wallet(wallet,path,ld_path,...);
...
gcc = pkg-native(wallet,“gcc”);
gcc([source,“-o”,“myprog”]);
```

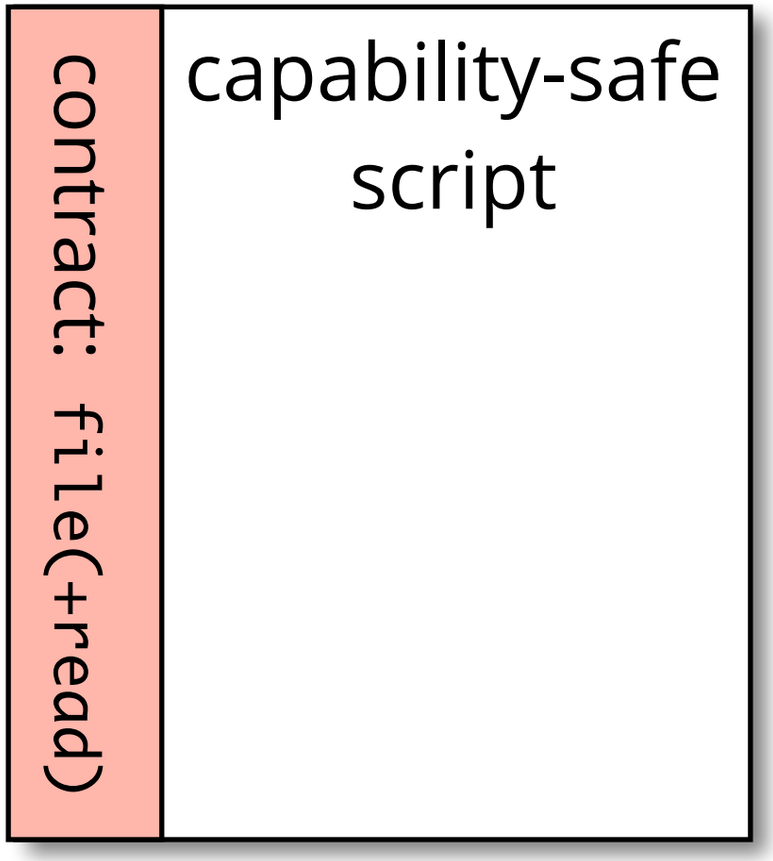
Putting it together

Architecture of shill

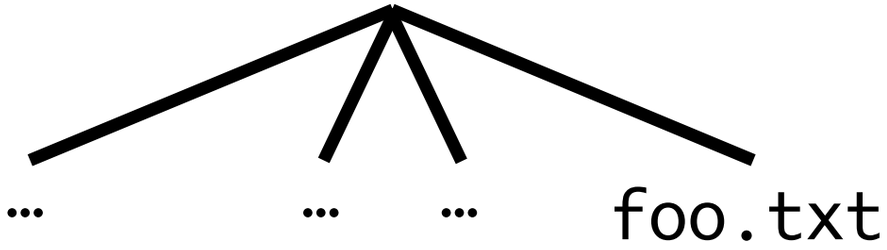


Architecture of shill

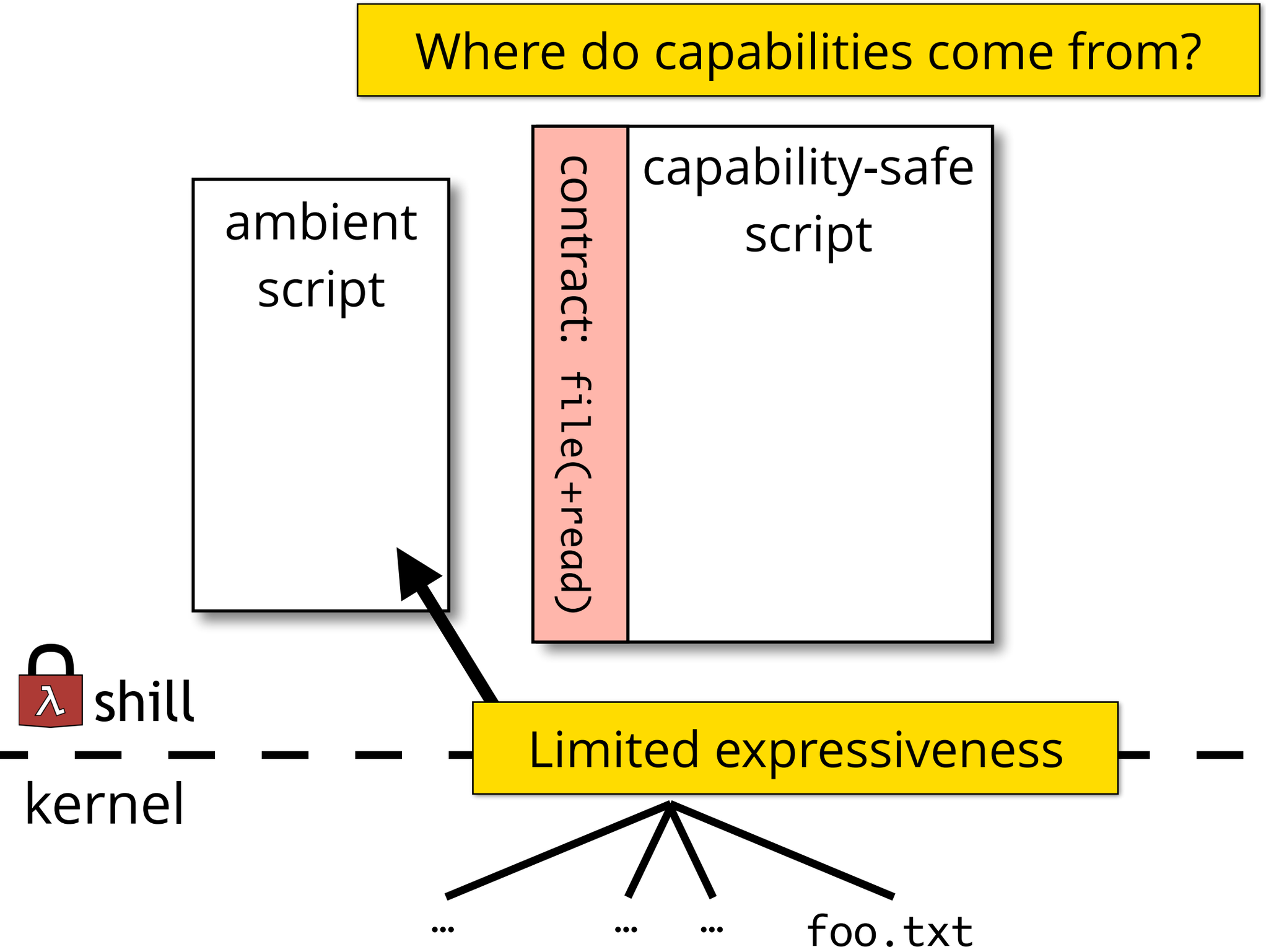
Where do capabilities come from?



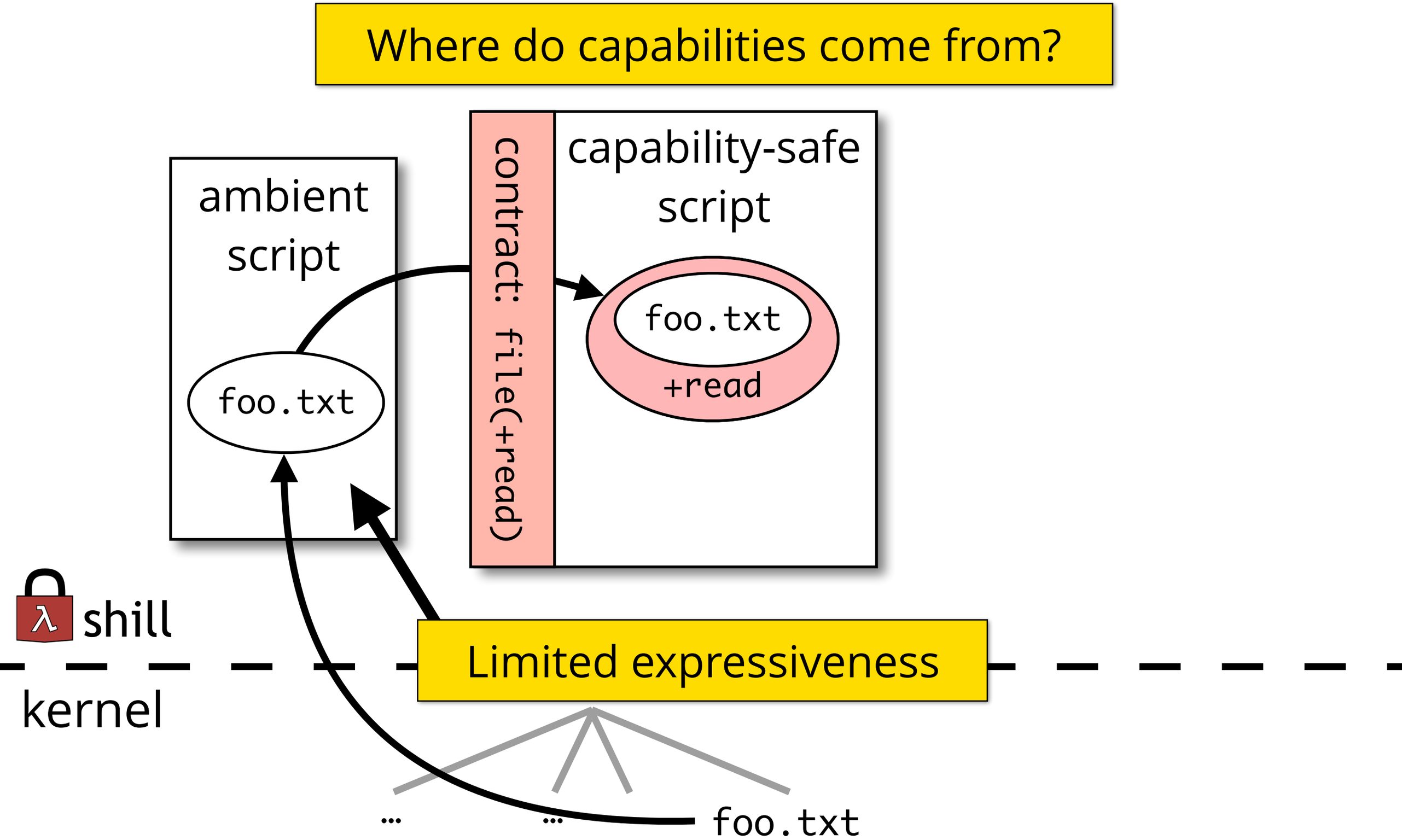
kernel



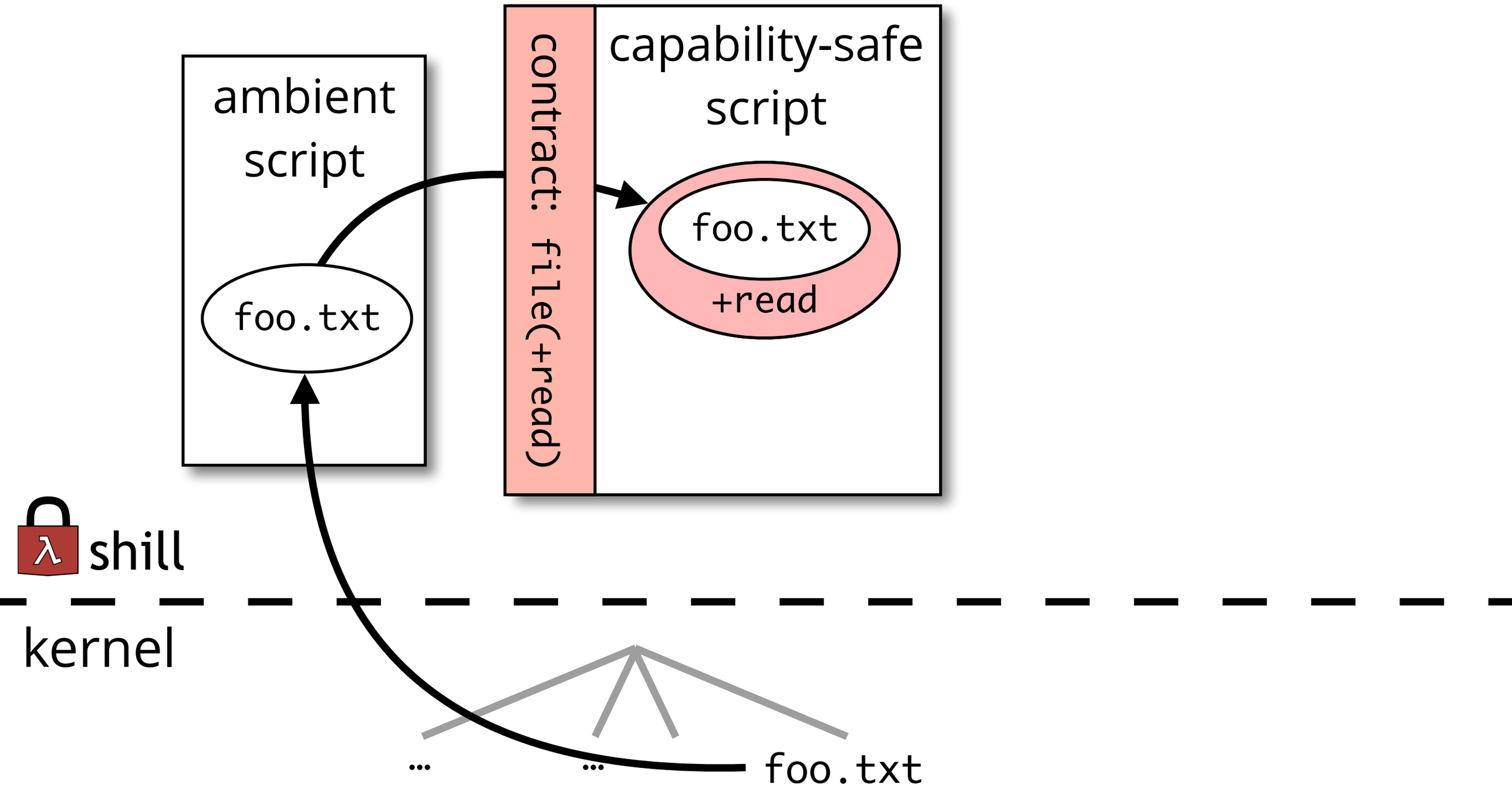
Architecture of shill



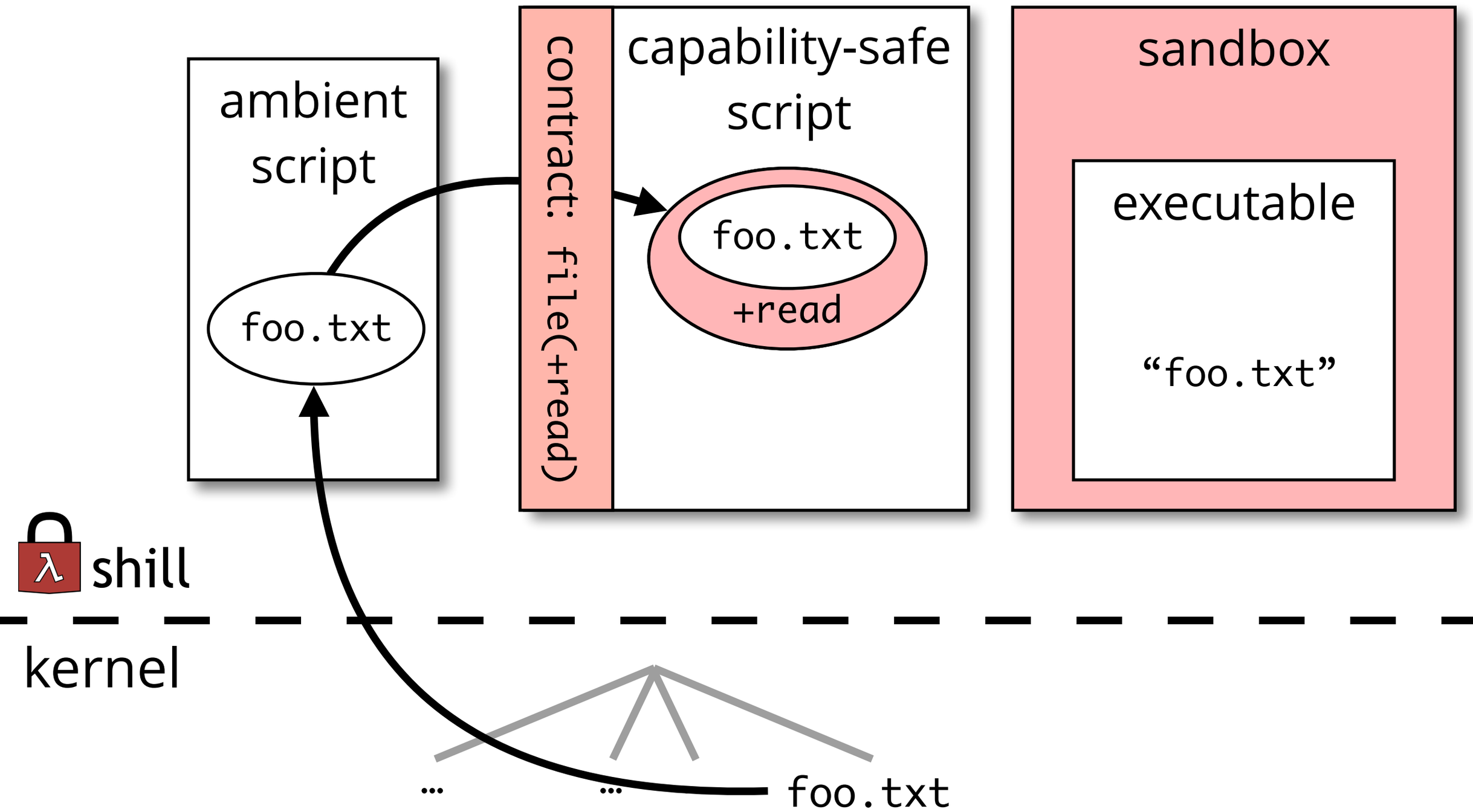
Architecture of shill



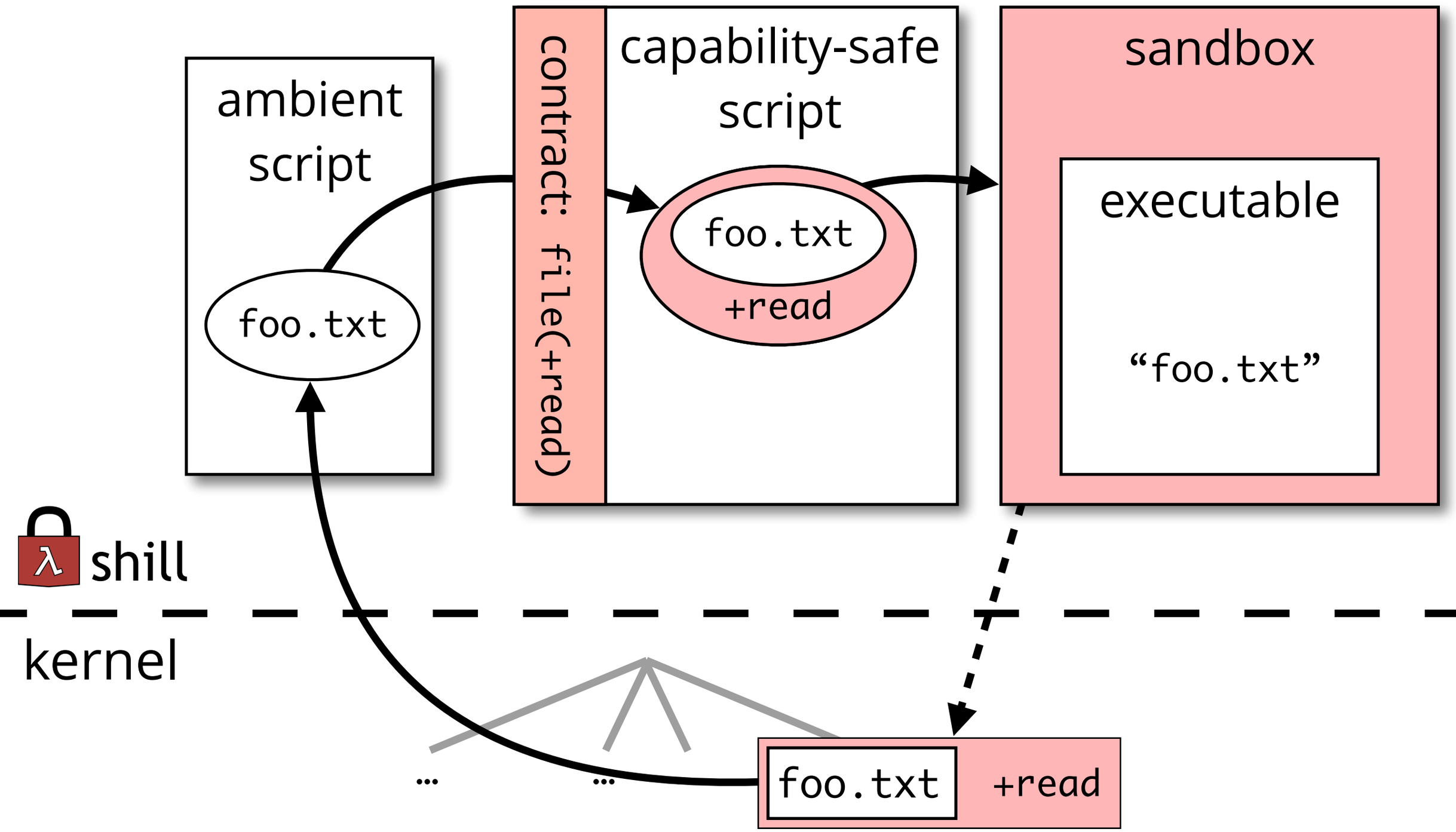
Architecture of shill



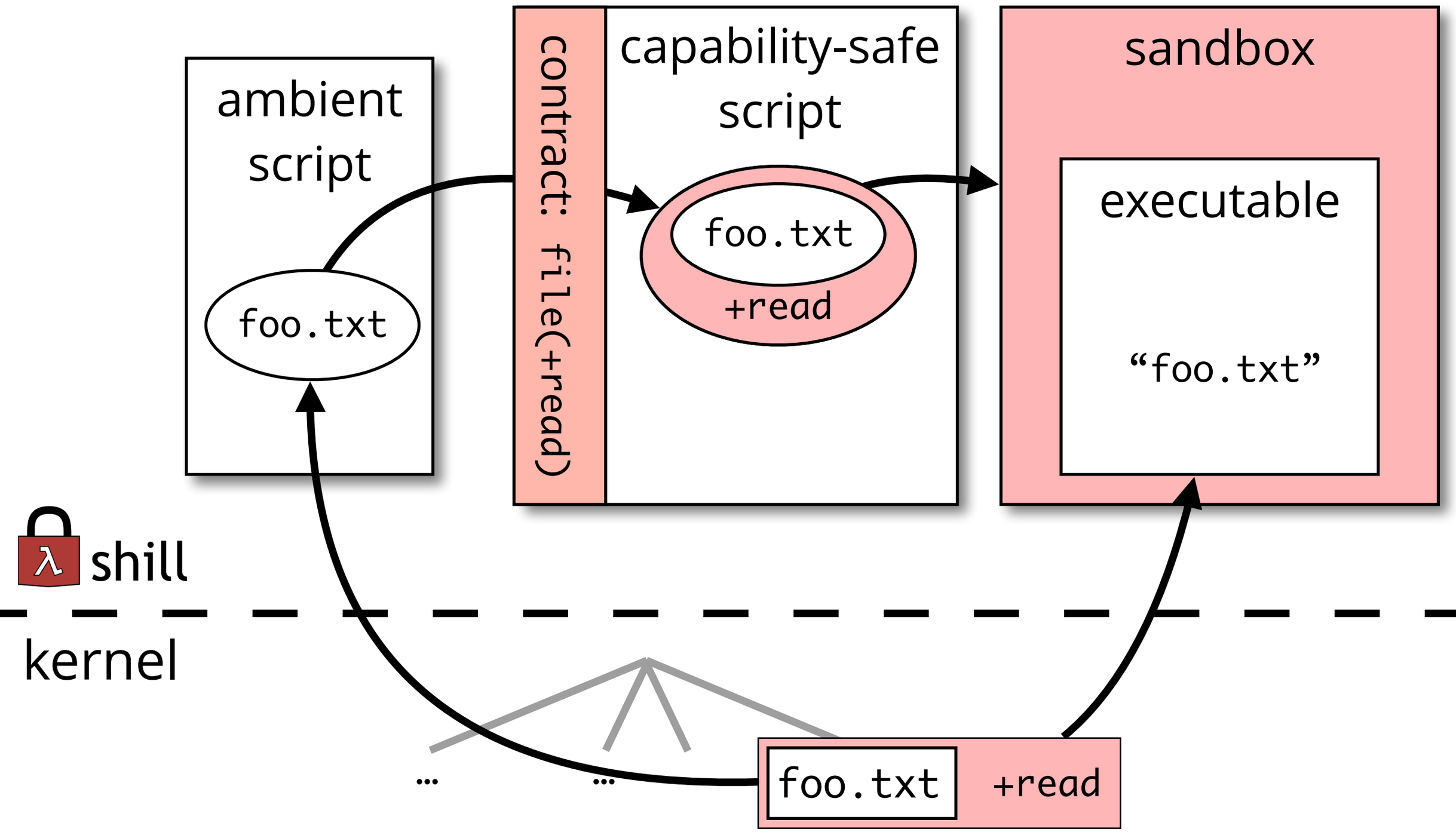
Architecture of shill



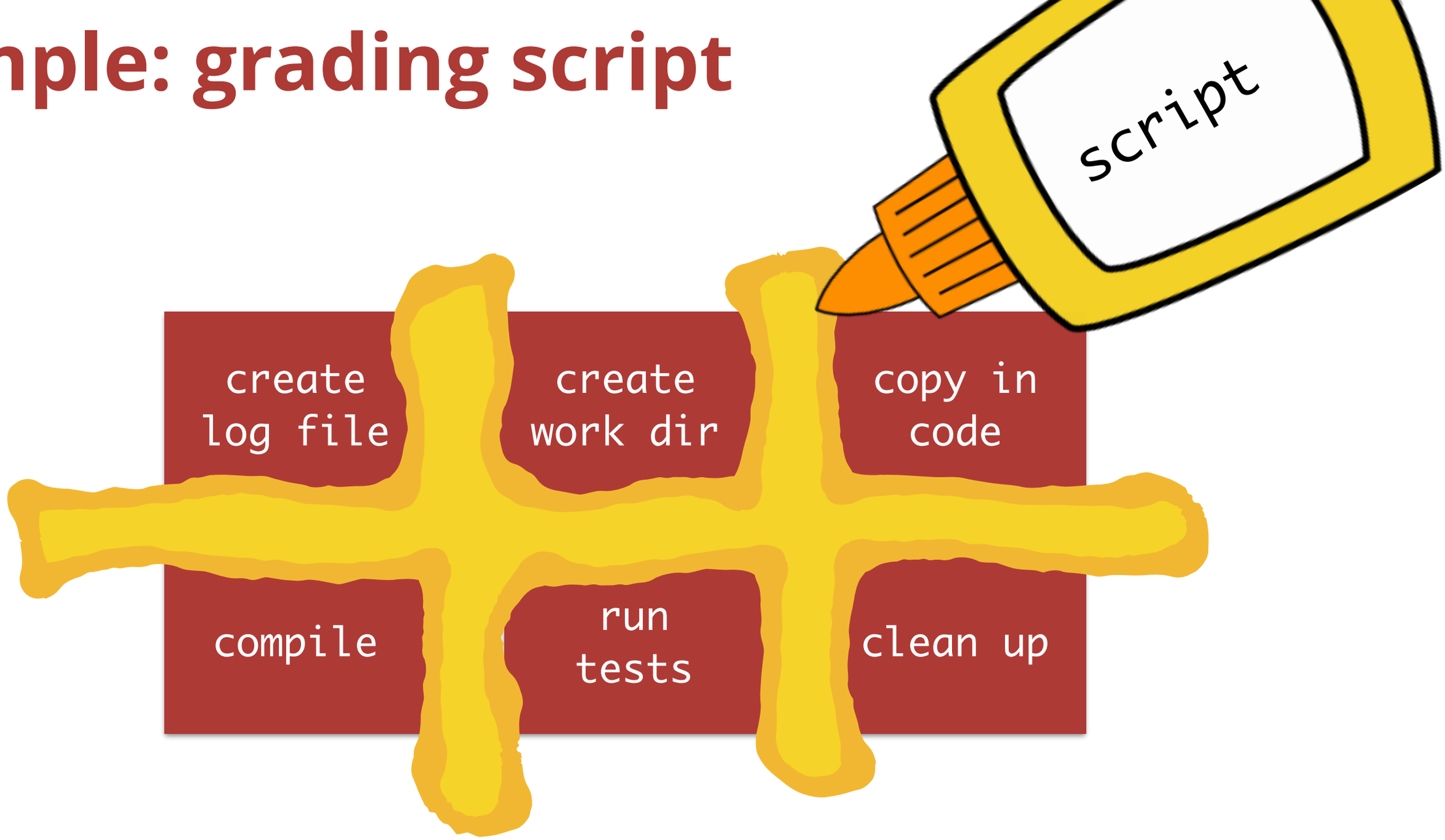
Architecture of shill



Architecture of shill



Example: grading script

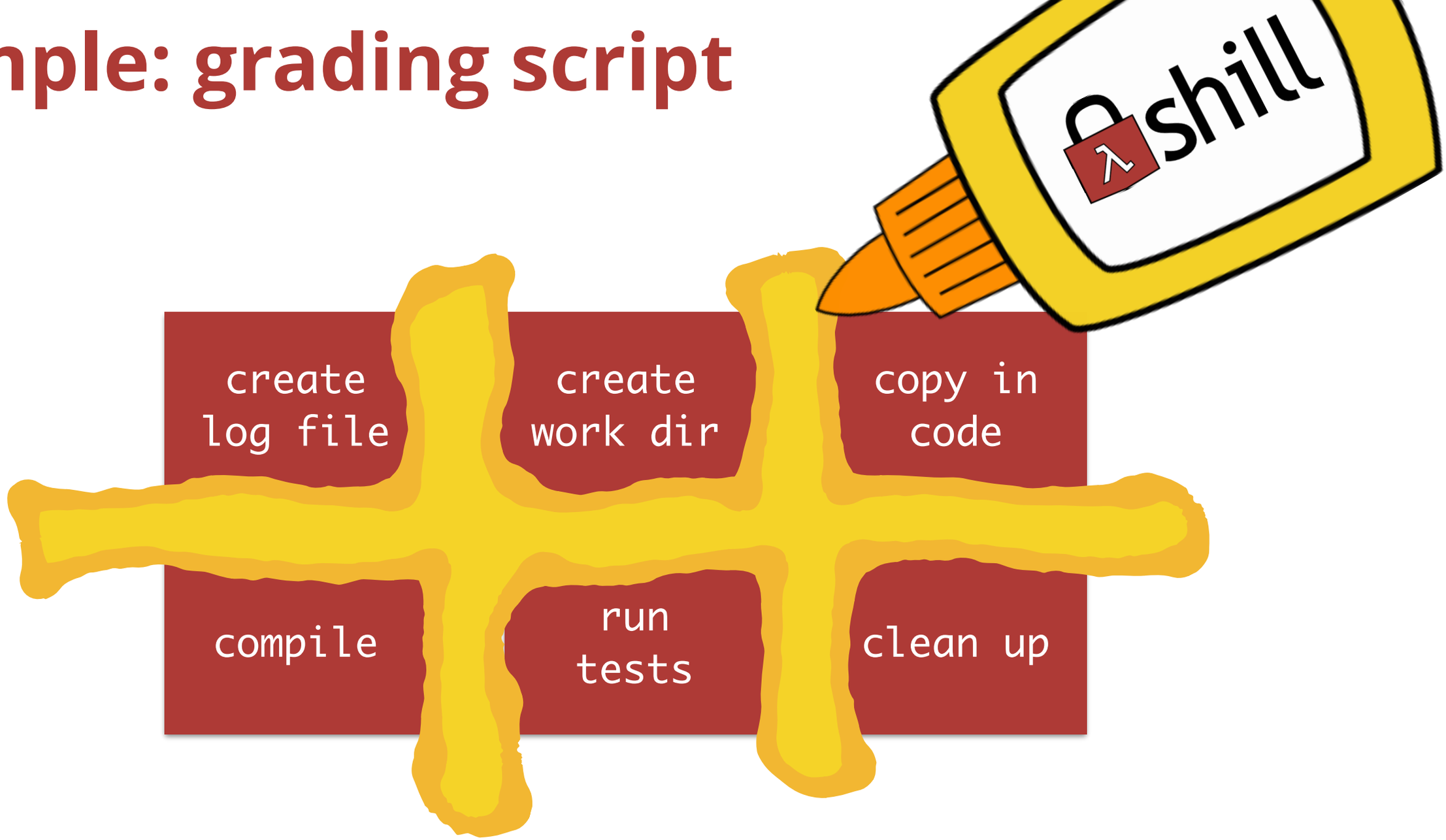


G1. Don’t corrupt my other files

G2. Don’t modify or leak the test suite

G3. Don’t allow submissions to interact

Example: grading script



✓ Reuse the glue for security

Scripting with Least Privilege

```
provide grade :
{ submission : is_file && readonly,
  tests      : is_dir  && readonly,
  working    : is_dir(+create_dir with
                       full_privilege),
  grade_log  : is_file && appendonly,
  extras     : libc_wallet } → void;
```

G1. Don’t corrupt my other files

G2. Don’t modify or leak the test suite

G3. Don’t allow submissions to interact

Scripting with Least Privilege

```
provide grade :
{ submission : is_file && readonly,
  tests      : is_dir  && readonly,
  working    : is_dir(+create_dir with
                       full_privilege),
  grade_log  : is_file && appendonly,
  extras     : libc_wallet } → void;
```

✓ G1. Don’t corrupt my other files

G2. Don’t modify or leak the test suite

G3. Don’t allow submissions to interact

Scripting with Least Privilege

can't modify test suite

```
provide grade :
{ submission : is_file && readonly,
  tests      : is_dir  && readonly,
  working    : is_dir(+create_dir with
                       full_privilege),
  grade_log  : is_file && appendonly,
  extras     : libc_wallet } → void;
```

no network capability to leak tests

✓ G1. Don't corrupt my other files

✓ G2. Don't modify or leak the test suite

G3. Don't allow submissions to interact

Scripting with Least Privilege

submissions isolated

can't modify test suite

```
provide grade :
{ submission : is_file && readonly,
  tests      : is_dir  && readonly,
  working    : is_dir(+create_dir with
                       full_privilege),
  grade_log  : is_file && appendonly,
  extras     : libc_wallet } → void;
```

no network capability to leak tests

✓ G1. Don't corrupt my other files

✓ G2. Don't modify or leak the test suite

✓ G3. Don't allow submissions to interact

Evaluation

Implementation

Racket

Capability-safe subset of racket
Capability-based systems library
Contracts built with Racket combinators

FreeBSD®

Capability-based sandbox
Policy module for Trusted MAC framework
Few additional capability-safe system calls

Case studies

Grading OCaml assignments
sandboxed execution, isolation between students

GNU Emacs installer
sandboxed execution, install/uninstall locations

Apache webserver
read-only access to config and content directories

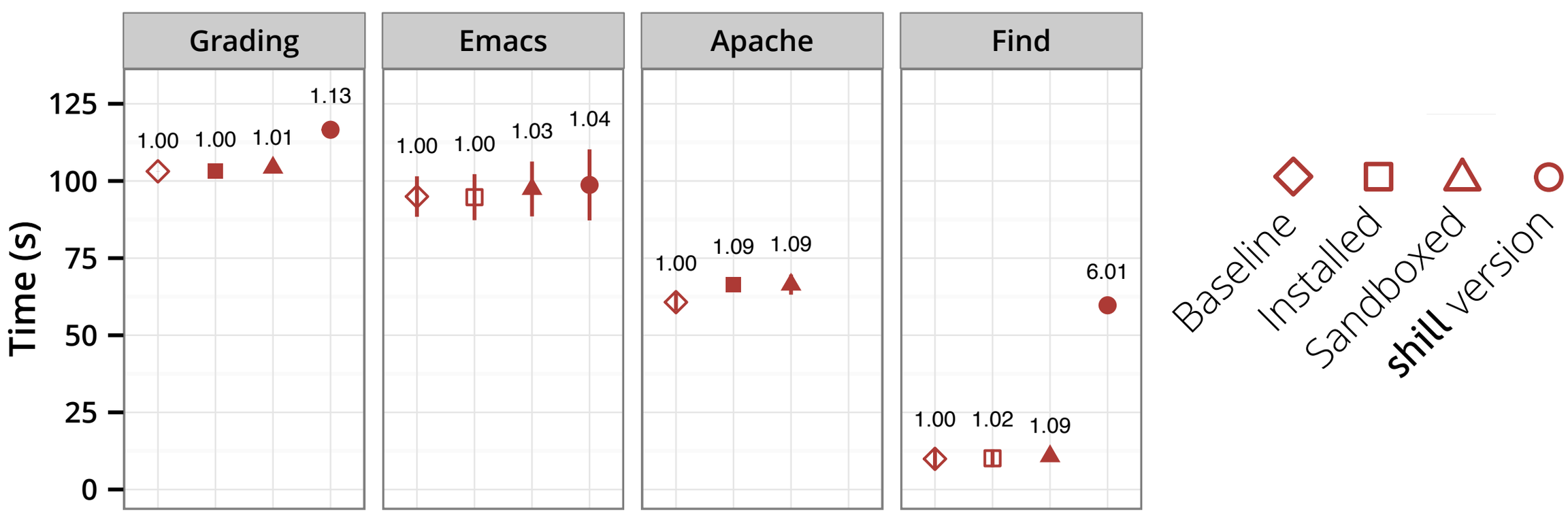
`find`

find and execute
sandboxed execution per-file

Performance

Overhead generally below 20%

Overhead proportional to security guarantees



Capabilities to manifest authority

+

Contracts to communicate authority

+

Contracts and *sandboxes* to control authority

=

Scripting with Least Privilege

www.shill-lang.org