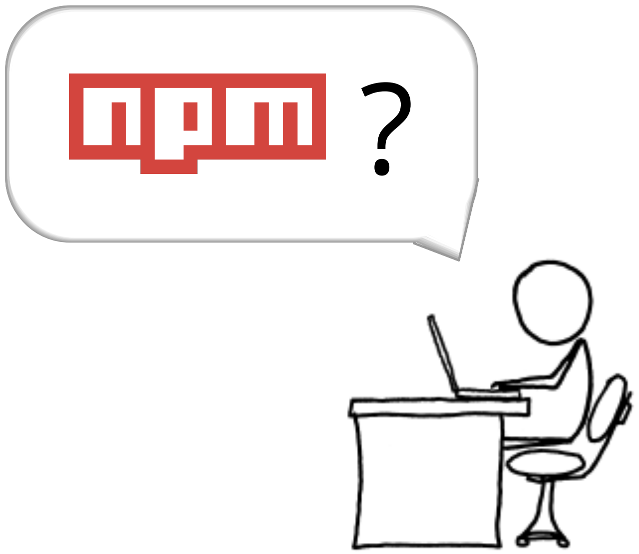
shill

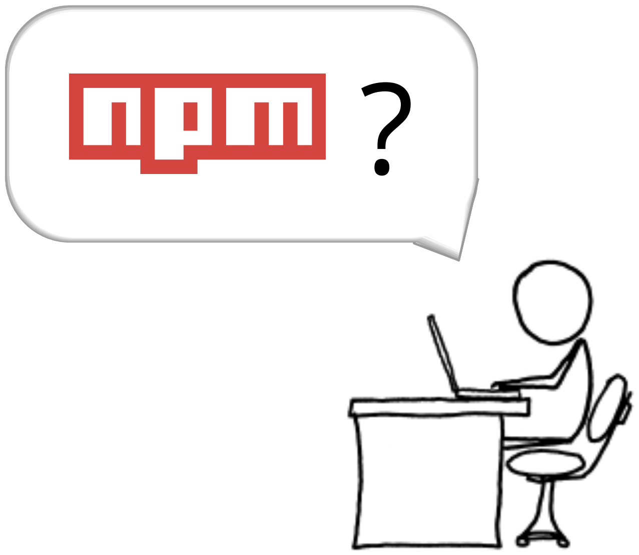
Scripting with Least Privilege

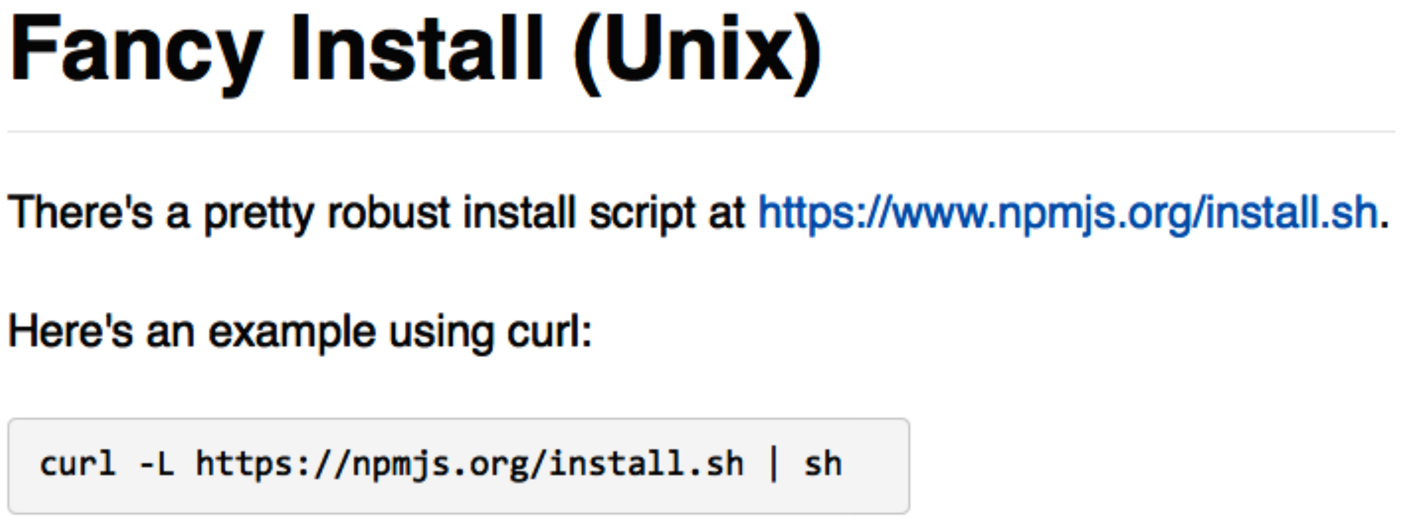
or: Contracts for Security

Scott Moore

`(sixth RacketCon)`


npm ?



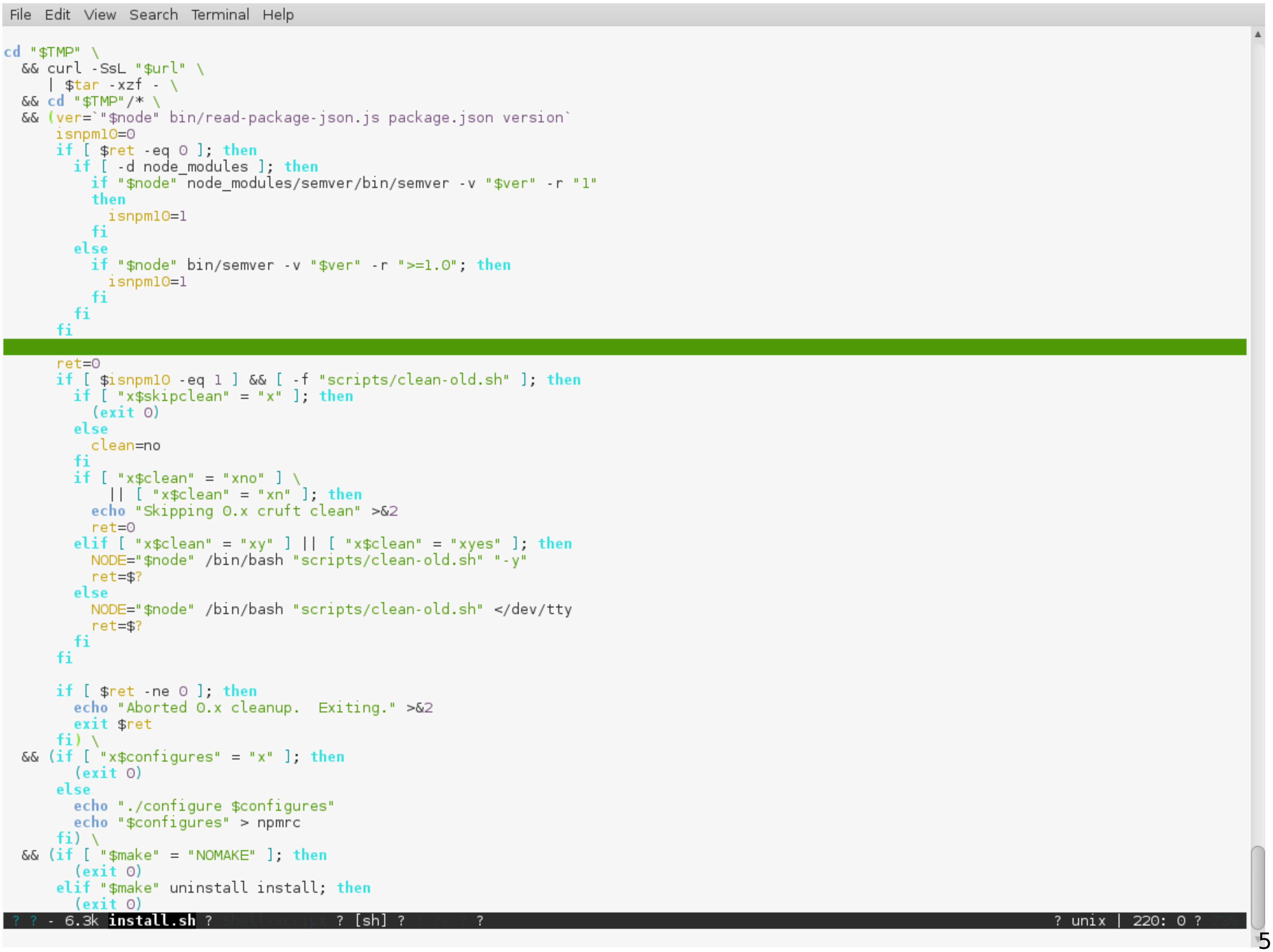


Fancy Install (Unix)

There's a pretty robust install script at https://www.npmjs.org/install.sh.

Here's an example using curl:

```
curl -L https://npmjs.org/install.sh | sh
```

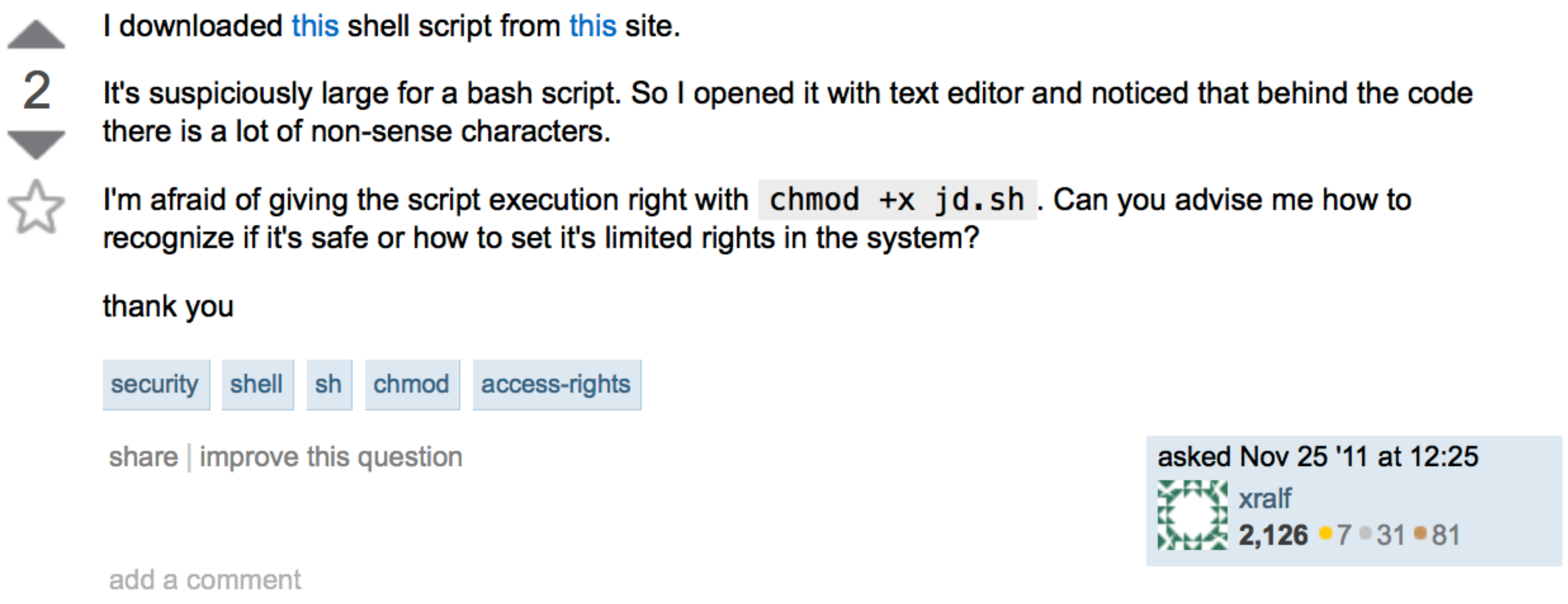


```
cd "$TMP" \
  && curl -SsL "$url" \
     | $tar -xzf - \
  && cd "$TMP"/* \
  && (ver=`"$node" bin/read-package-json.js package.json version`
      isnpm10=0
      if [ $ret -eq 0 ]; then
        if [ -d node_modules ]; then
          if "$node" node_modules/semver/bin/semver -v "$ver" -r "1"
          then
            isnpm10=1
          fi
        else
          if "$node" bin/semver -v "$ver" -r ">=1.0"; then
            isnpm10=1
          fi
        fi
      fi

      ret=0
      if [ $isnpm10 -eq 1 ] && [ -f "scripts/clean-old.sh" ]; then
        if [ "x$skipclean" = "x" ]; then
          (exit 0)
        else
          clean=no
        fi
        if [ "x$clean" = "xno" ] \
            || [ "x$clean" = "xn" ]; then
          echo "Skipping 0.x cruft clean" >&2
          ret=0
        elif [ "x$clean" = "xy" ] || [ "x$clean" = "xyes" ]; then
          NODE="$node" /bin/bash "scripts/clean-old.sh" "-y"
          ret=$?
        else
          NODE="$node" /bin/bash "scripts/clean-old.sh" </dev/tty
          ret=$?
        fi
      fi

      if [ $ret -ne 0 ]; then
        echo "Aborted 0.x cleanup.  Exiting." >&2
        exit $ret
      fi) \
  && (if [ "x$configures" = "x" ]; then
        (exit 0)
      else
        echo "./configure $configures"
        echo "$configures" > npmrc
      fi) \
  && (if [ "$make" = "NOMAKE" ]; then
        (exit 0)
      elif "$make" uninstall install; then
        (exit 0)
? ? - 6.3k install.sh ?  ? [sh] ?  ?                                ? unix | 220: 0 ?
```

How can I recognize if it is safe?



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

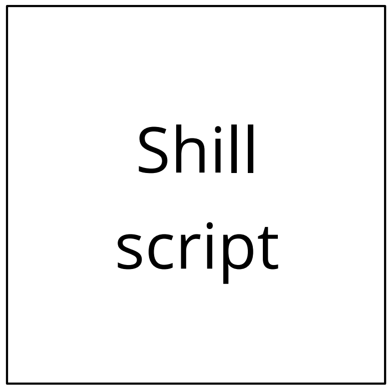
asked Nov 25 '11 at 12:25
xralf
2,126 ● 7 ● 31 ● 81

add a comment

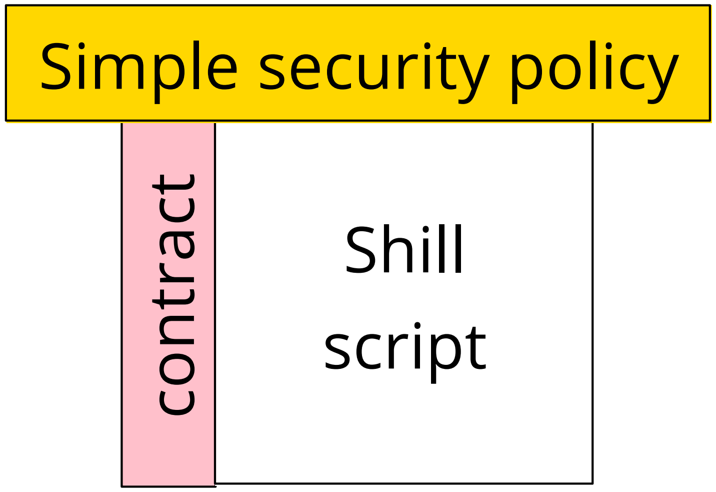
Principle of Least Privilege

> “Every program … should operate using the least amount of privilege necessary to complete the job.”
>
> *—Jerome Saltzer, CACM*

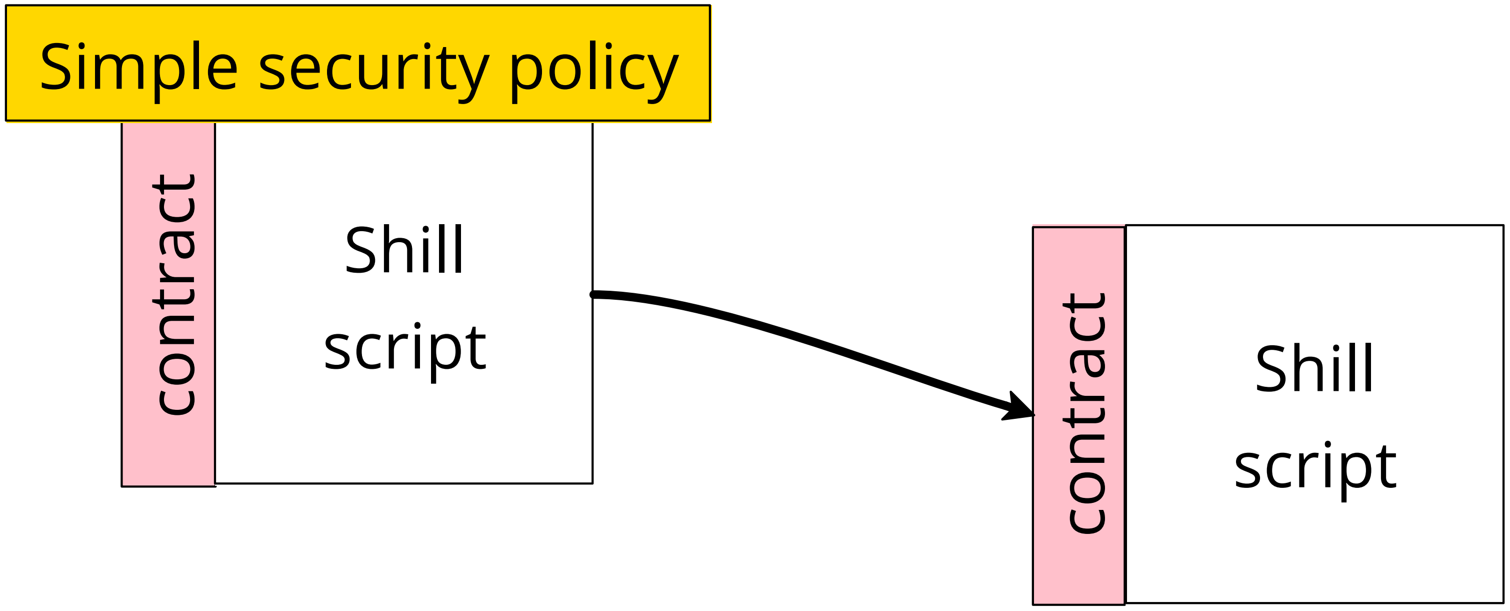
Scripting with Least Privilege



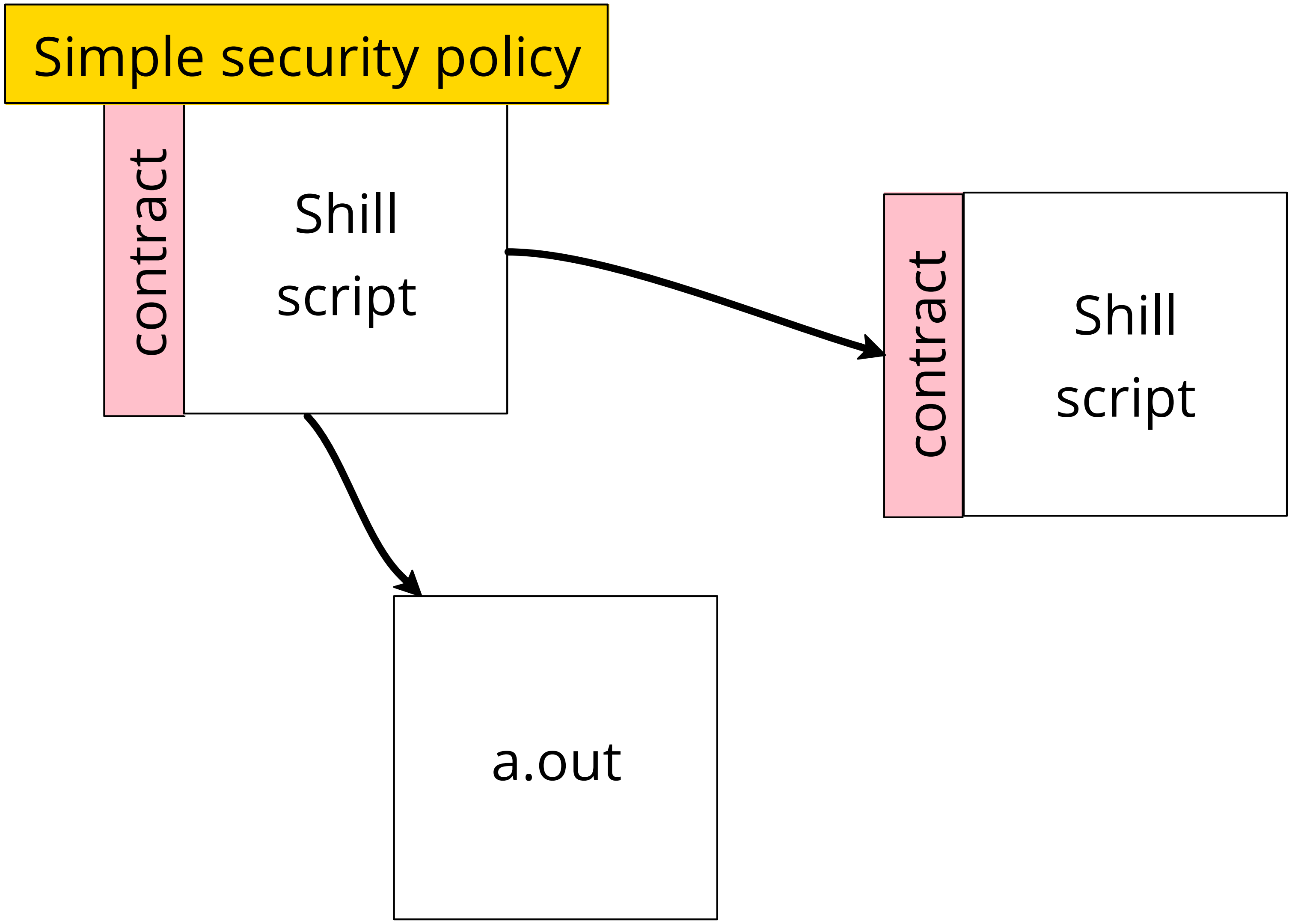
Scripting with Least Privilege



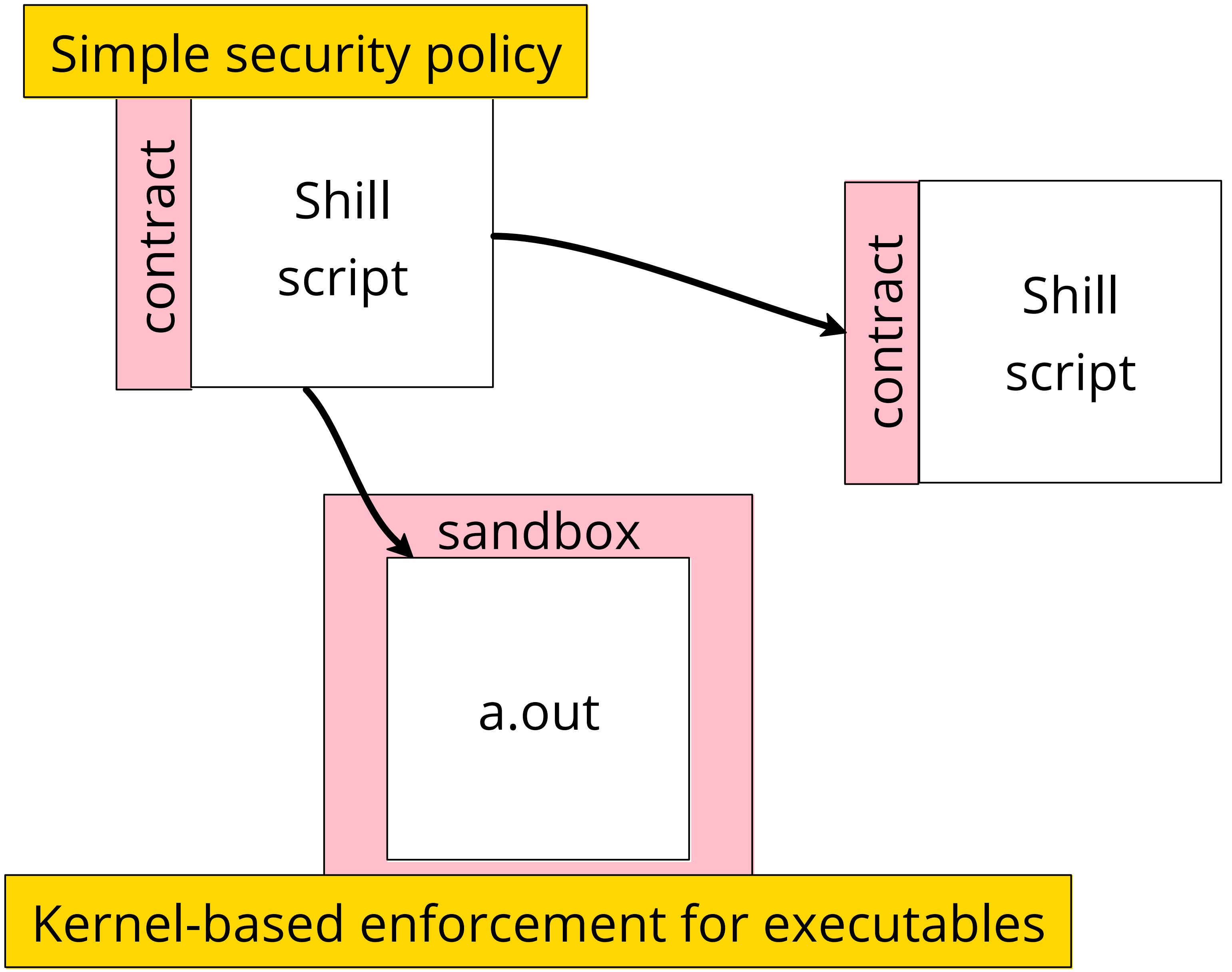
Scripting with Least Privilege



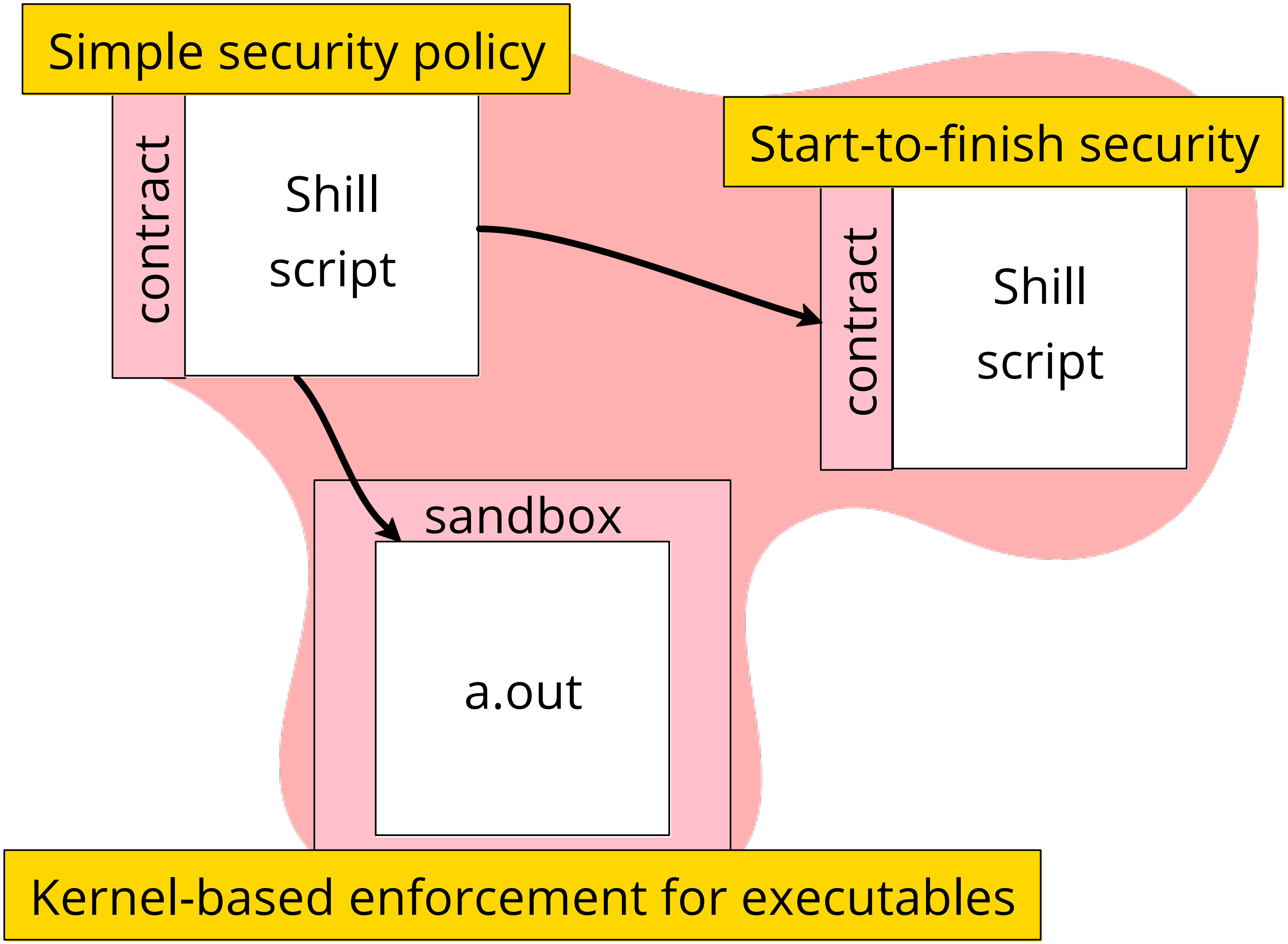
Scripting with Least Privilege



Scripting with Least Privilege



Scripting with Least Privilege



Capability-based security

A capability is an ***unforgeable token of authority***

shill only accesses system resources through operations on capabilities

A Secure Shell Script

copy.cap

```
#lang shill/cap

provide { copy : any/c };

copy = fun(from_dir,to_dir) {
  for entry in contents(from_dir) do {
    current = lookup(from_dir,entry);
    if file?(current) then {
      new = create-file(to_dir,entry);
      write(new,read(current))
    }
  }
}
```

A Secure Shell Script

copy.cap

```
#lang shill/cap

provide { copy : any/c };

copy = fun(from_dir,to_dir) {
  for entry in contents(from_dir) do {
    current = lookup(from_dir,entry);
    if file?(current) then {
      new = create-file(to_dir,entry);
      write(new,read(current))
    }
  }
}
```

Capability safety: scripts have no capabilities by default

Fine-grained security with contracts

copy.cap

```
#lang shill/cap

provide { copy : {
  from : dir/c(+contents, +lookup with { +read }),
  to   : dir/c(+create-file with { +write }) }
  -> void };

require shill/io;

copy = fun(from_dir,to_dir) {
  for entry in contents(from_dir) do {
    current = lookup(from_dir,entry);
    if file?(current) then {
      fwrite(current,"evil");
      new = create-file(to_dir,entry);
      write(new,read(current))
    }
  }
}
```

Contracts describe exactly how a script will use its capabilities

Fine-grained security with contracts

copy.cap

```
#lang shill/cap

provide { copy : {
  from : dir/c(+contents, +lookup with { +read }),
  to   : dir/c(+create-file with { +write }) }
  -> void };

require shill/io;

copy = fun(from_dir,to_dir) {
  for entry in contents(from_dir) do {
    current = lookup(from_dir,entry);
    if file?(current) then {
      fwrite(current,"evil");
      new = create-file(to_dir,entry);
      write(new,read(current))
    }
  }
}
```

Programmable sandboxes

cat.cap

```
#lang shill/cap

provide { cat : {
  cat     : file/c(+exec,+read,+stat),
  file    : file/c(+path,+read),
  lookup  : listof(dir/c(+lookup,+stat)),
  libs    : listof(file/c(+exec,+read,+stat)),
  ro      : listof(file/c(+read,+stat)),
  out     : writeable/c }
  -> integer? };

require shill/contracts;

val cat = fun(cat,file,lookups,libs,ro,out) {
  exec(cat,["cat",file],stdout = out,stderr = out,
       extra = list-append(lookups,libs,ro));
}
```

Under the hood

Black box capability-based sandboxing for executables

+ a few new capability-safe system calls

`#lang shill/cap`:

Capability-safe safe subset of **`racket/base`**

+ a **`set!-transformer`** to control mutation

+ a **`require-transformer`** to only import shill code

+ a capability-based filesystem library using **`ffi/unsafe`**

+ capability contracts using **`racket/contract`**

+ a custom reader

What’s next

Developing commercial version of Shill

Porting to Linux

Plug-in framework for new kinds of capabilities (processes, databases, ...)

shill

http://shill-lang.org