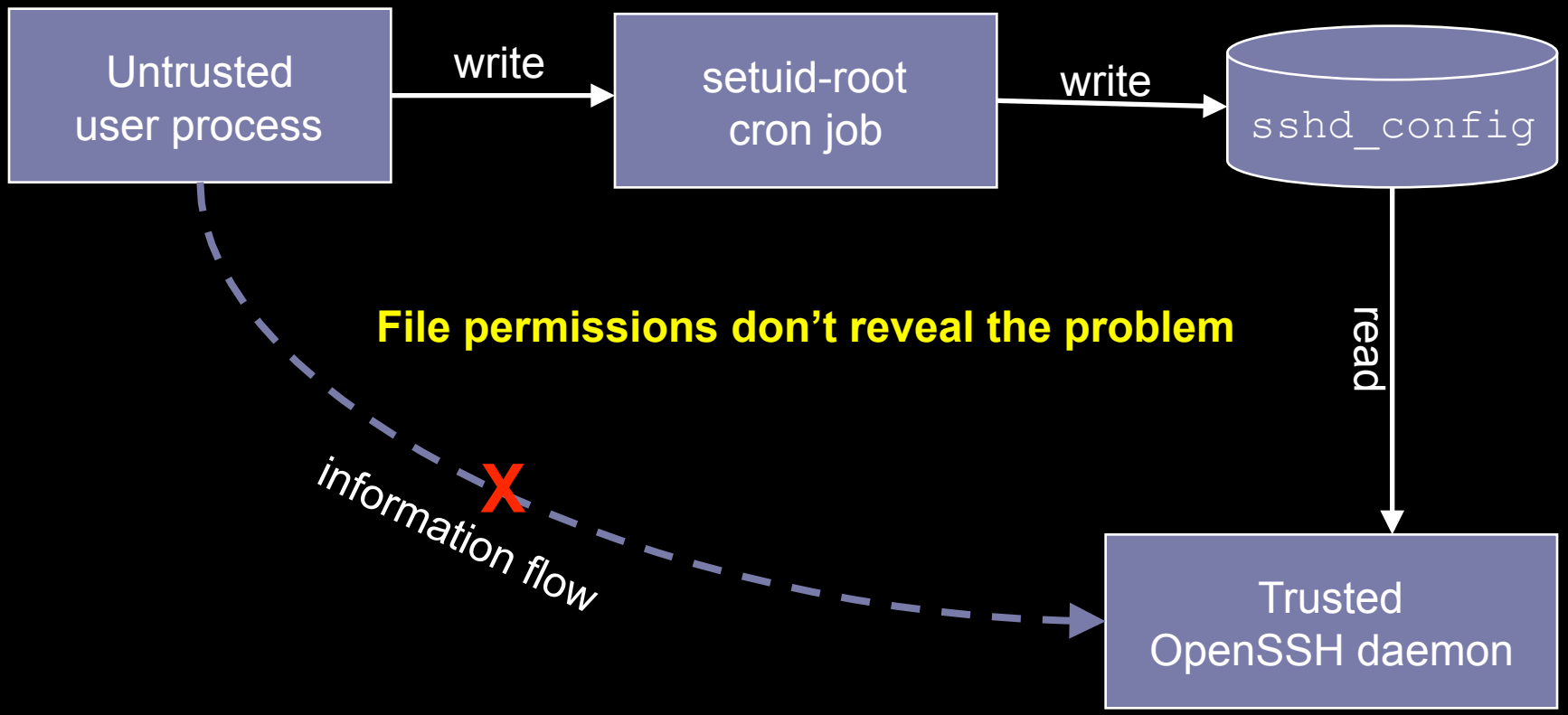
Toward Automated Info-Flow Integrity Verification
(or, Fixing your security policy)

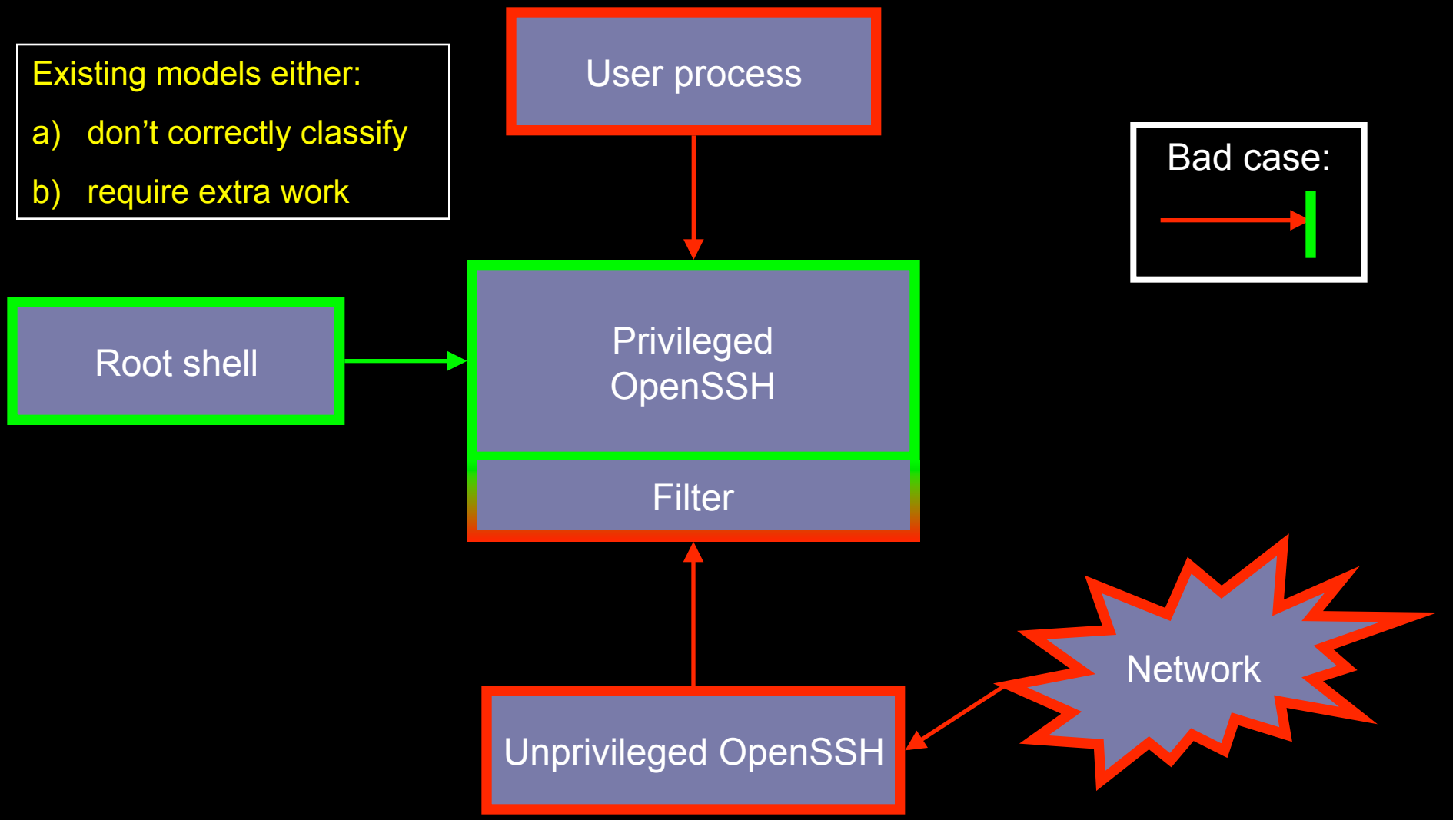
Umesh Shankar (UC Berkeley)
Trent Jaeger (Penn State / IBM)
Reiner Sailer (IBM)

The goal, with an example

Integrity property: Trusted processes don't depend on untrusted ones



Legal vs. illegal flows



Our new integrity model: CW-Lite

- Motivation: previous models aren't practical
- Preserve info-flow rules of Clark-Wilson
  - Filter untrusted inputs to trusted processes
- But relax two constraints:
  - Don't require all interfaces to perform filtering
  - Check existence of filters, not correctness

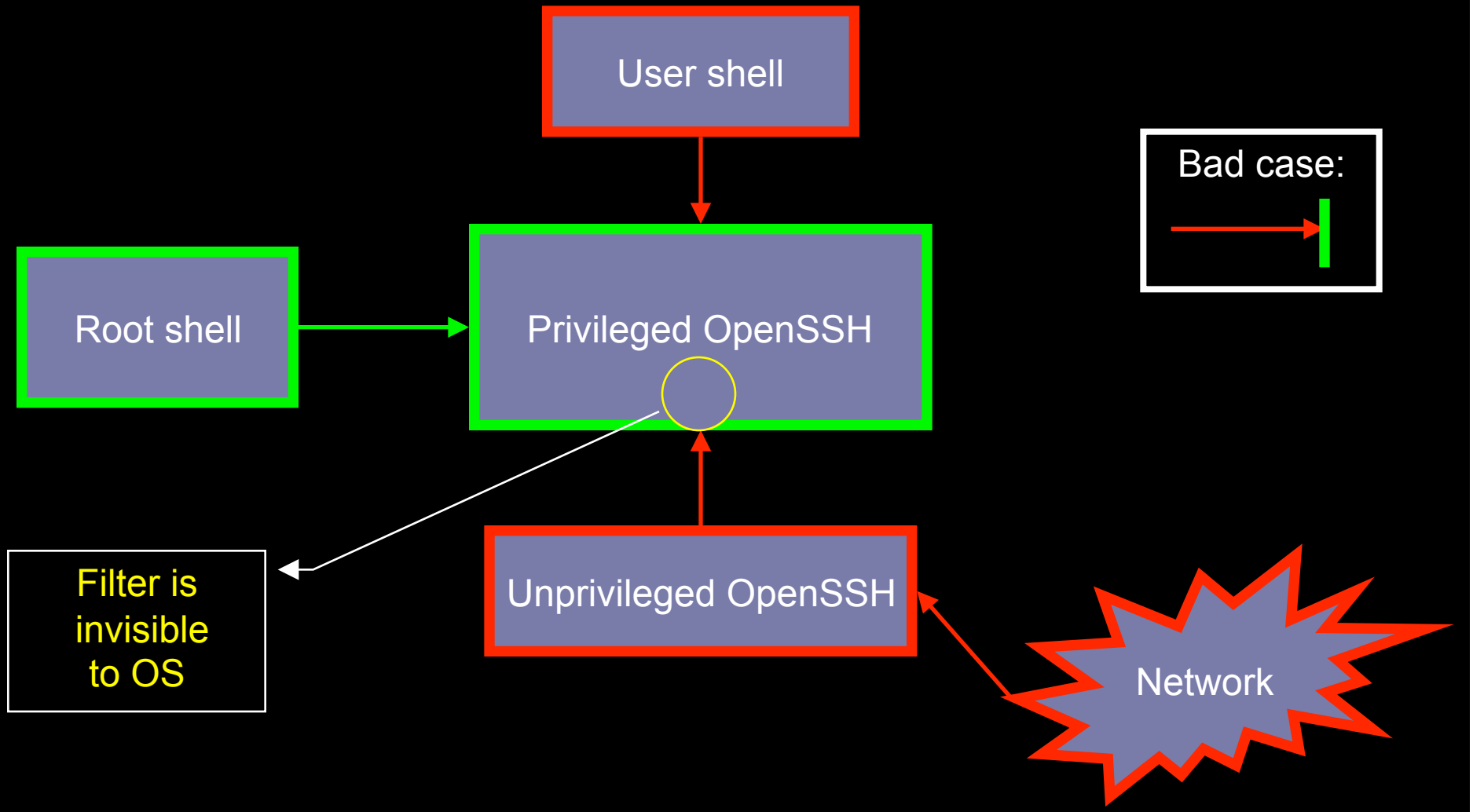
Contributions

- Useful middle ground (C-W vs. nothing)
- Usable with today's apps and OS
- Amenable to automated verification
- Tools to detect and fix integrity violations
- Found several problems with OpenSSH policy in Fedora Linux

Verifying CW-Lite (overview)

1. Build information-flow graph
2. Find potentially illegal flows
   - Use Gokyo policy analysis tool
3. If needed, fix security policy and repeat

The OS View: Process info-flow



Terminology

- Subject = process
- Object = file, pipe, shared memory, etc.
- Subject Type = process security label
- Object Type = a label on each object
- Permissions =
  (subject type, operation, object type)

- Example: (sshd, read, sshd_config_file)

Information flow from policy

- SELinux implements complete mediation
- So all information flows are exposed

Inferring information flows:

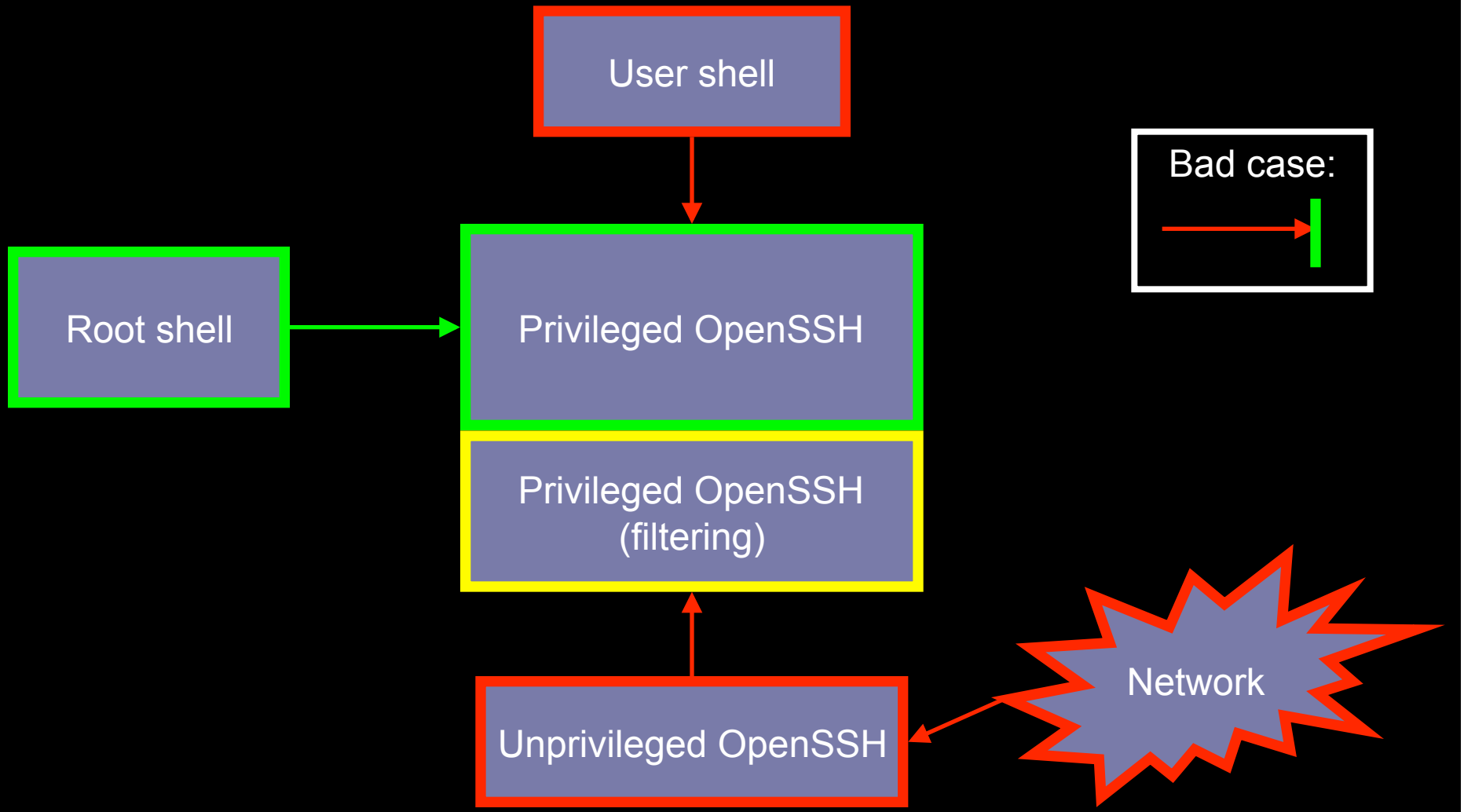
```
( Subject S can write to object O ∧
  Subject T can read from O )
    ⇔ Information flow from S to T
```

- We use the Gokyo tool (Jaeger+ ’03) to do this step statically

Exposing filtering interfaces

- MAC system can't see filtering interfaces
  - Permissions are per-process, not per-interface
- Solution: Send hint from inside the process
  - Programmer adds annotation to filtered interface
- Use two subject types for each process
  - *Default subject type* allows inputs only from TCB
  - Filtering interfaces use *filtering subject type* which enables additional permissions

Subject type info flow graph

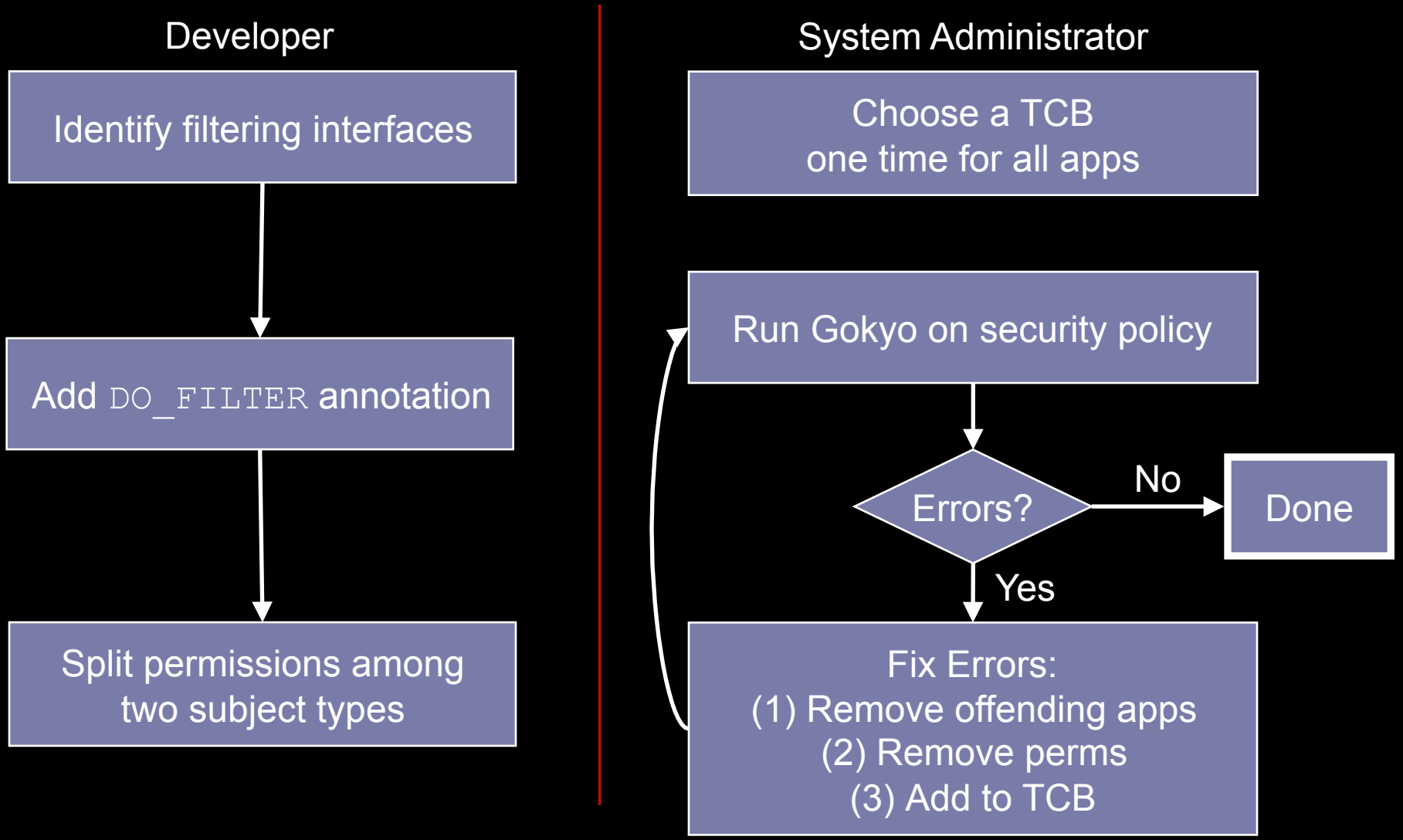


Enabling filtering subject types

- SELinux kernel mod enables two subject types (default & filtering) for each process
- User library extension adds
  - Ability to switch between both subject types
  - DO_FILTER convenience macro

```
DO_FILTER(f()) :=
  Enable filtering subject type
  Call f()
  Disable filtering subject type
```

Who has to do what



Finding filtering interfaces

- Developer analyzes default policy
- Untrusted input permission found
  - Where is it used in the program?
  - Is it really necessary? If so, it should be filtered
- New tracing function to help diagnosis
  - SELinux kernel modification
  - Traps into debugger when that permission used

Filtering Interface Example

BEFORE

Source Code

```
conn = accept()
// accept() fails
get_request_sanitized(conn)
```

Security Policy (default DENY)

```
Apache: ALLOW read httpd.conf
// Problem: network not in TCB
Apache: ALLOW accept
```

AFTER

Source Code

```
DO_FILTER(conn = accept())
// accept() succeeds
get_request_sanitized(conn)
```

Security Policy (default DENY)

```
Apache: ALLOW read httpd.conf
// Apache-filter: non-TCB OK
Apache-filter: ALLOW accept
```

Example: OpenSSH — Approach

- Security-critical, privilege-separated
- Handwritten security policy
- 4 processes: `listen`, `priv`, `net`, `user`

Check untrusted flows to `priv`, `listen`

1. Define TCB: `kernel`, `init`, etc.
2. Run Gokyo
3. If conflicts exist: classify, resolve, repeat

Example: OpenSSH — Results

- Analyzed default SELinux policy in Fedora
- Gokyo yielded 20 conflicts
- Three kinds of solutions
  a) Remove offending applications (e.g. `rlogind`)
  b) Disable optional components
  c) Remove unnecessary permissions

Conclusion

- CW-Lite provides a useful information flow guarantee for existing systems
- Trades small developer effort for automated verification by sysadmins
- Helps expose trust relationships
- Using our tools, we found configuration errors in OpenSSH in a real distribution

Thanks!

Related Work

- Integrity Models
  - Biba ’75, Clark-Wilson ’87, LOMAC ’00, Caernarvon ’00
- Information Flow
  - Denning ’76 (Info flow rules as lattice constraints)
  - Li & Zdancewic ’05 (Type checking for info-flow)
  - Chow et al. ’04 (Whole-system information flow)

Related Directions

- The dual problem: secrecy
  - Paper at ICC ’06 (Shankar and Wagner)
- Attestation of the CW-Lite property
  - Useful for distributed systems, corporate LANs
  - Allows checking integrity of relevant processes on a machine being brought in
  - Paper in submission (Jaeger, Sailer, Shankar)