CS260 – Advanced Systems Security

Integrity

April 23, 2025

Data Integrity

- What is data integrity?
  - What do we need to do to ensure data integrity?

Integrity

- List some items that have integrity
  - What is the source of their integrity?

Integrity

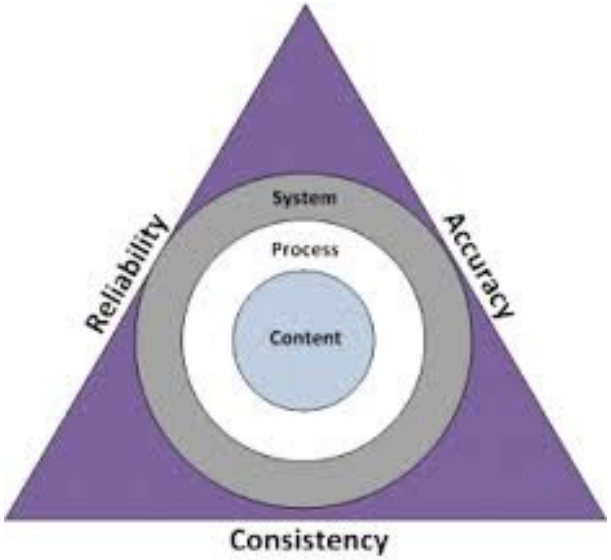
- List some items that have integrity
  - What is the source of their integrity?
- Forbes “Most Trustworthy Companies”
  - “In order to rank companies from the most to the least trustworthy, we look at over 60 different governance and forensic accounting measures…”
  - Not likely to fail, transparent, …
- Academic Integrity
  - Behavior complying with a code of conduct and ethics

Integrity in Software…

- What do expect for integrity of software?

… Impacts Data Integrity

- How does software integrity impact data integrity?



Least Privilege

- *The protection mechanism should force every process to operate with the minimum privileges needed to perform its task.*
- Due to Saltzer and Schroeder (of Multics project)
- One of many "design principles" in their paper "The Protection of Information in Computer Systems" (1975)
- Others
  - Principle of Psychological Acceptability
  - Principle of Fail Safe Defaults

Least Privilege

- How to compute least privilege?
  - Aim: Determines the permissions required for the program to run effectively
- Run the program and see what permissions are used
  - Proposed for a system called Systrace
  - SELinux audit2allow: take denied permissions and add them to policy
  - AppArmor Profile Wizard: Build an approximate profile statically and
    - http://www.novell.com/documentation/apparmor/book_apparmor21_admin/?page=/documentation/apparmor/book_apparmor21_admin/data/sec_apparmor_repo.html

Least Privilege

- Is a good goal because…
- Is a poor goal because…
- Can we use it to verify a policy is secure?

Least Privilege

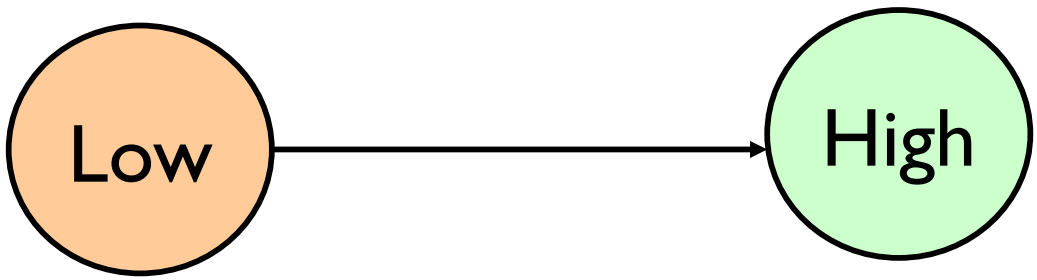
- Is a good goal because…
  - Unnecessary permissions lead to problems (confused deputy)
  - Accounts for function
- Is a poor goal because…
  - Task permissions may conflict with security
  - How do we know when a permission is necessary, but makes the system insecure?
- Can we use it to verify a policy is secure?
  - No. *It defines a policy based on function, not security.*

Information Flow for Integrity

- Another approach looks at the authorized flow of information among processes via objects

Idealized Security

- **Biba Integrity**
  - Integrity requirement: Do not *depend* on data from lower integrity principals
  - Only permit information to flow from high integrity to lower integrity
  - E.g., Can only read a file if your integrity level is dominated by or equal to the file's



Practical vs. Ideal

- Do these idealized approaches based on information flow enable practical realization of OS enforcement?
- Secrecy is possible in some environments
  - Implemented in a paper world, previously
- Integrity has not been realized in practice
  - Many processes provide high integrity services to others
- Result: Depend on many applications to manage information flows

Assured Guards

- What do we do if a system needs an information flow from low integrity to high?
  - E.g., reading from a network socket
- Not authorized by Biba
  - Unless subject is fully assured to upgrade to high integrity or discard low integrity data
  - Called a guard
- What does “fully assured” mean?

LOMAC [Fraser 2000]

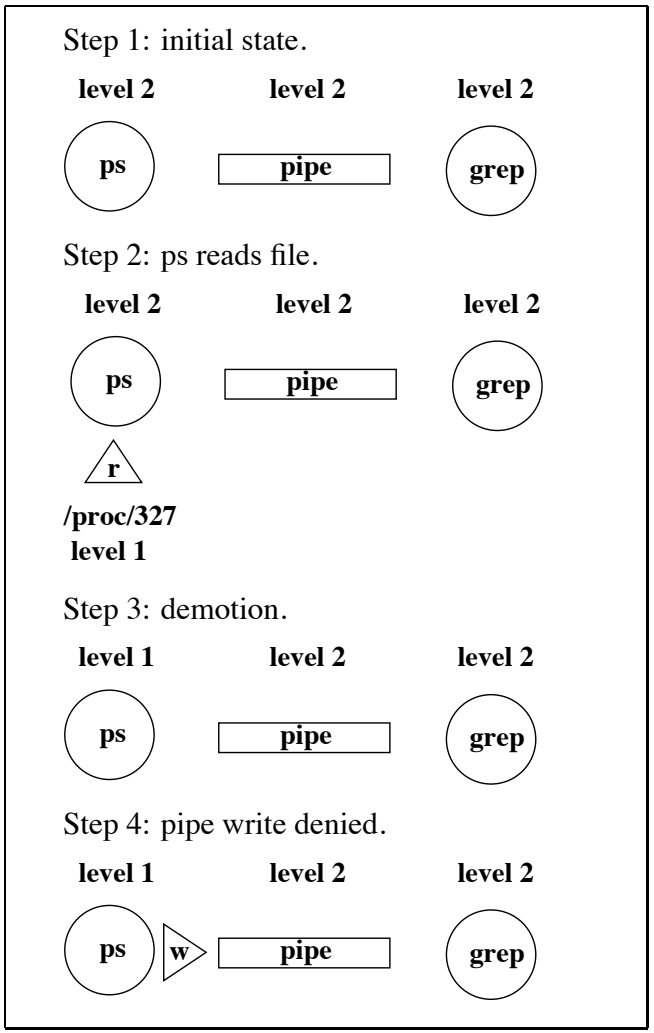
- Subjects and objects have an integrity label
  - Level and category in a lattice policy
- When subject reads an object of a lower integrity label in lattice
  - Subject's label is lowered to that of object
  - Define subject's label in terms of objects accessed
- When subject writes to an object of a higher integrity label in lattice
  - Write is denied
  - Read is still allowed

Biba vs LOMAC

- What is allowed and what is the resultant label?
  - Lattice A → B → C
- Subject at A reads object at C
  - Biba?
  - LOMAC?
- Subject at C writes object at A
  - Biba?
  - LOMAC?
- Subject at C reads from object at A

LOMAC Self-Revocation

- Can cause revocation of own access to objects in LOMAC



Information Flow

- Is a good goal because…
- Is a poor goal because…
- Can we use it to verify a policy is correct?

Information Flow

- Is a good goal because…
  - No false negatives – an attack requires an illegal information flow
  - Can define data and functional security requirements
- Is a poor goal because…
  - Function may conflict with security
  - How do we know when a permission is illegal, but is necessary for functional requirements?
- Can we use it to verify a policy is correct?
  - Yes. *It defines a policy based on security. But what about exceptions?*

Clark-Wilson Integrity Model

- Goal: define integrity in terms of commercial terms rather than military (information flow)
- Insights?

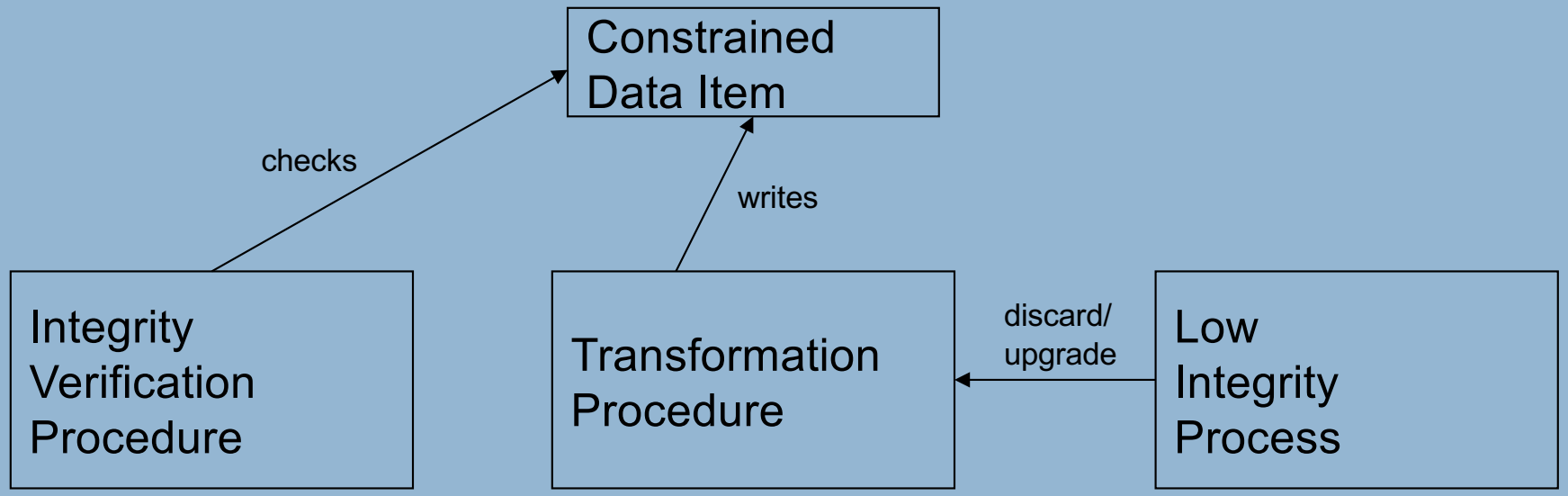
Clark-Wilson Integrity Model

- Goal: define integrity in terms of commercial terms rather than military (MLS/Biba)
- Insights? Based on Double-Blind Accounting
  - Start with high integrity data
    - Validate data integrity (integrity verification procedures)
  - Only apply high integrity processes to change that data
    - Distinguish high integrity code (transformation procedures)
  - Ensure high integrity processes protect themselves
    - When they receive low integrity inputs (convert or reject)
  - Recheck that data still satisfies integrity requirements (IVP)

Clark-Wilson Integrity Model

- Model consists of a set of certification and enforcement rules governing integrity
- Own terms
  - CDI – Constrained Data Items (High integrity data)
  - UDI – Unconstrained Data Items (Low integrity data)
  - IVP – Integrity Verification Procedures (certify CDIs)
  - TP – Transformation Procedures (High integrity programs)

Clark-Wilson Integrity Model



Effectiveness of IVP and TP are guaranteed based on assurance

Clark-Wilson Integrity Model

- Model consists of a set of certification and enforcement rules governing integrity
  - C1—When an IVP is executed, it must ensure the CDIs are valid.
  - C2—For some associated set of CDIs, a TP must transform those CDIs from one valid state to another.
  - C3—Allowed relations must meet the requirements of "separation of duty."
  - C4—All TPs must append to a log enough information to reconstruct the operation.
  - C5—Any TP that takes a UDI as input may only perform valid transactions for all possible values of the UDI. The TP will either accept (convert to CDI) or reject the UDI.

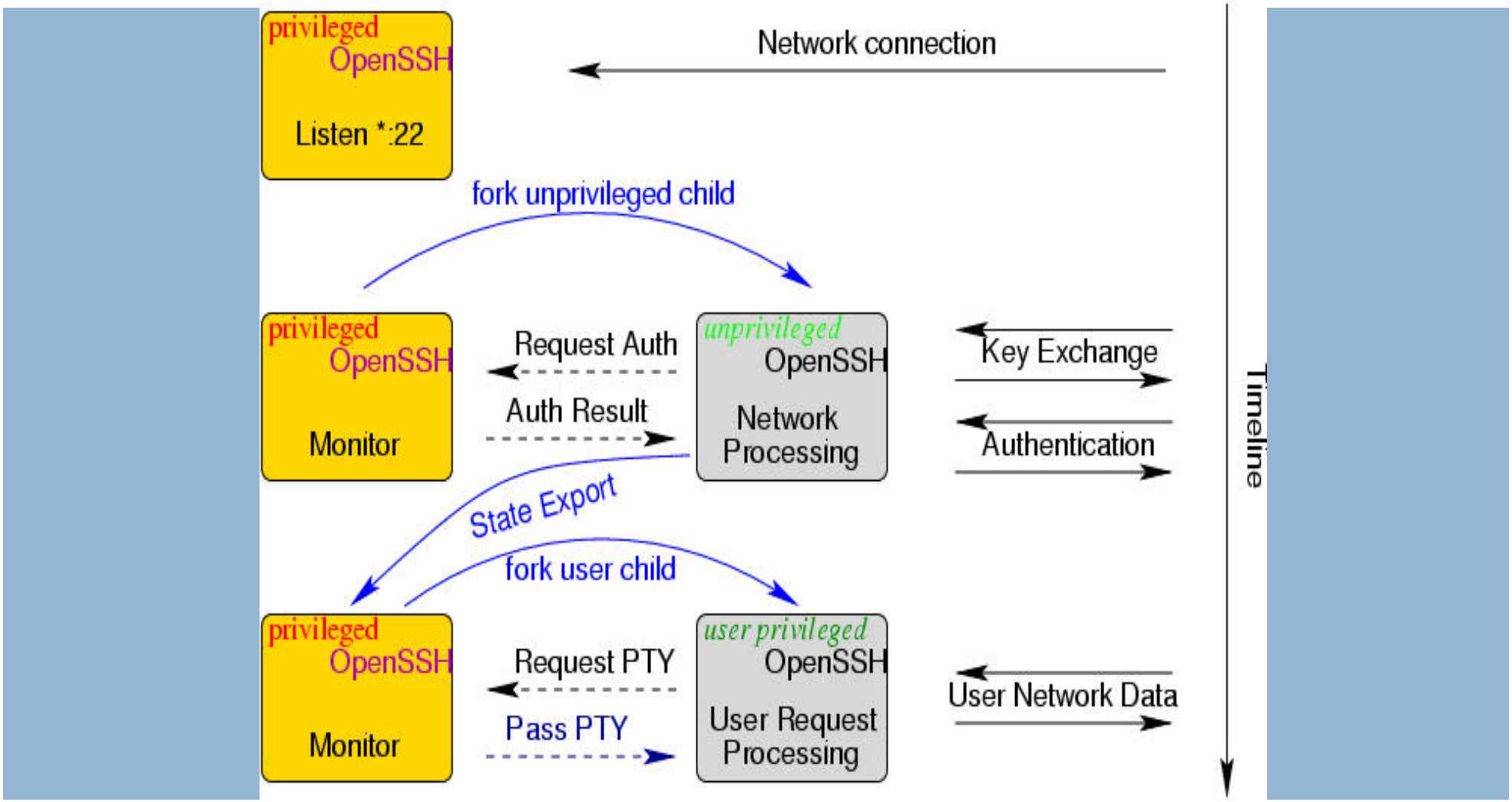
Clark-Wilson Integrity Model

- Model consists of a set of certification and enforcement rules governing integrity
  - E1—System must maintain a list of certified relations and ensure only TPs certified to run on a CDI change that CDI.
  - E2—System must associate a user with each TP and set of CDIs.
  - E3—System must authenticate every user attempting a TP.
  - E4—Only the certifier of a TP may change the list of entities associated with that TP.

Clark-Wilson Integrity Model

- How does it work?
- Certify TPs and IVPs
  - IVPs certify CDIs and TPs modify them
  - TPs must also be able to handle an UDIs they receive securely
- Run the system
  - Authenticated users can modify a CDI if and only if:
    - They can access TP and CDI and
    - TP is authorized to change CDI

Target Subject: Privilege Separated OpenSSH

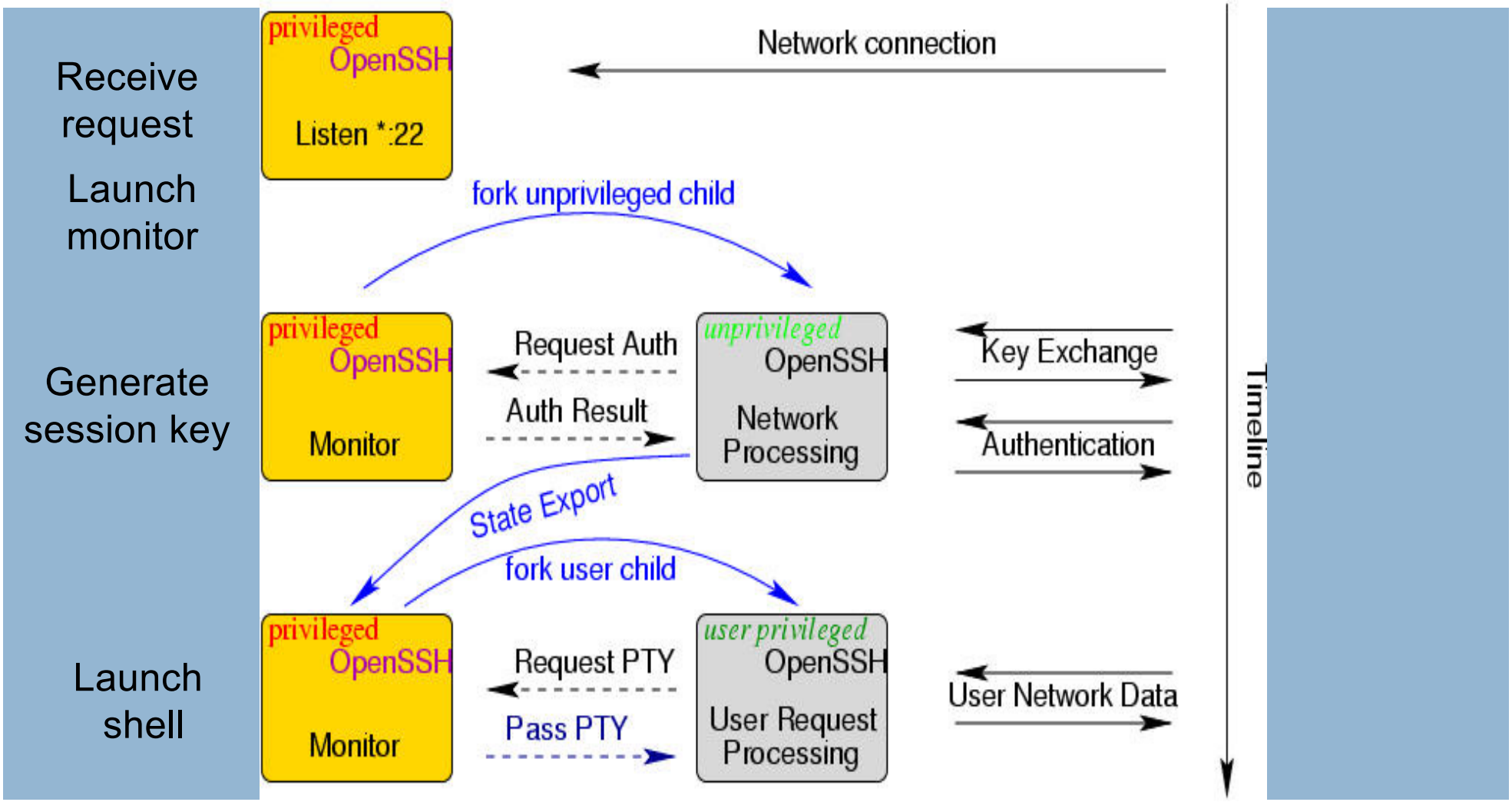


Picture from Niels Provos

Clark-Wilson Integrity Model

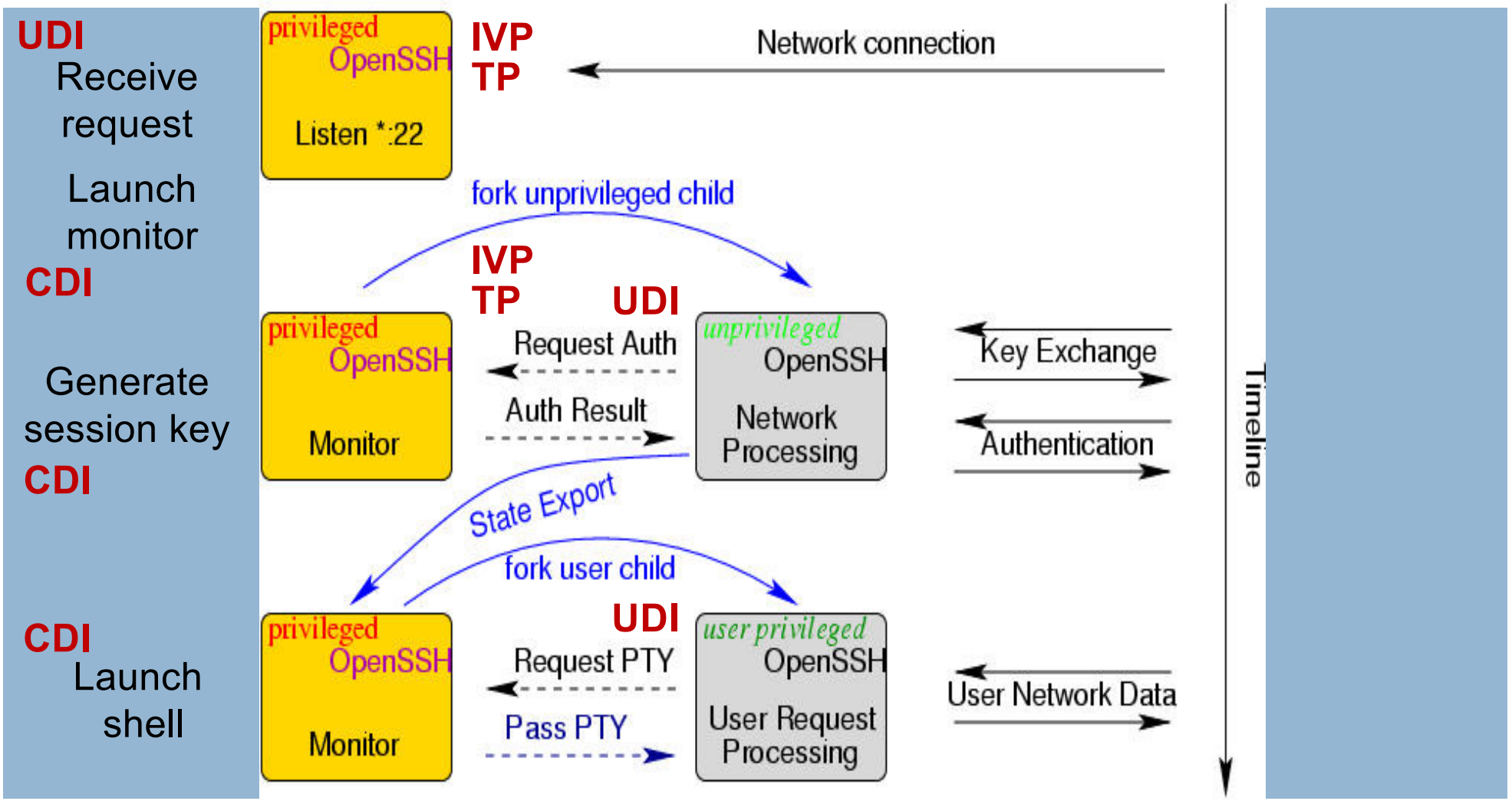
- Which are UDIs and CDIs?
  - Authenticated users can modify a CDI if and only if:
    - They can access TP and CDI and
    - TP is authorized to change CDI
- Which are TPs and IVPs?
  - IVPs certify CDIs and TPs modify them
  - TPs must also be able to handle an UDIs they receive securely (via IVPs)

Target Subject: Privilege Separated OpenSSH



Picture from Niels Provos

Target Subject: Privilege Separated OpenSSH



Picture from Niels Provos

Clark-Wilson Results

- Are the information flows authorized different than information flow?
  - T. M. P. Lee. Using mandatory integrity to enforce “commercial” security. In IEEE Symposium on Security and Privacy, pages 140–146, Oakland, April 1988.
  - W. R. Shockley. Implementing the Clark/Wilson integrity policy using current technology. In 11th National Computer Security Conference, pages 29–37, Baltimore, October 1988.
- Not really

Clark-Wilson Results

- Are the information flows authorized different than information flow?
  - T. M. P. Lee. Using mandatory integrity to enforce “commercial” security. In IEEE Symposium on Security and Privacy, pages 140–146, Oakland, April 1988.
  - W. R. Shockley. Implementing the Clark/Wilson integrity policy using current technology. In 11th National Computer Security Conference, pages 29–37, Baltimore, October 1988.
- Not really, but CW is closer to (ideal) practice
  - Test and analyze code (for integrity), certify code (e.g., signature), check code and data integrity before use (e.g., hash), and deal with untrusted inputs (e.g., filter)

Clark-Wilson Results

- If systems practice is analogous (but not quite) to Clark-Wilson integrity, then where are we going wrong?

Clark-Wilson Results

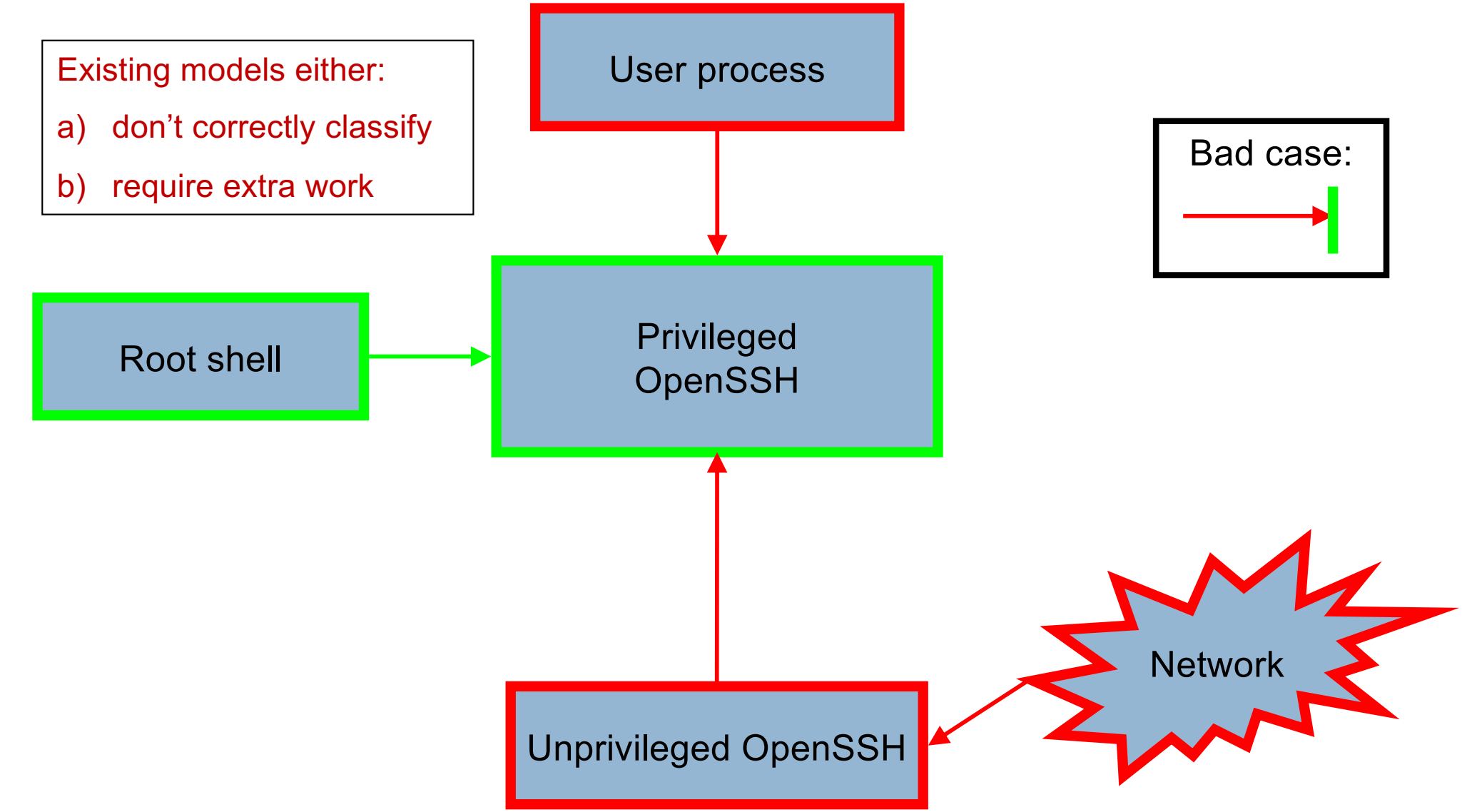
- If systems practice is analogous to Clark-Wilson integrity where are we going wrong?
  - Not certifying TPs or IVPs meet expectations
  - Not distinguishing CDIs from UDIs explicitly
  - Not systematically ensuring programs discard/upgrade UDIs

- But shouldn't programs at least know where they expect to receive UDIs (low integrity data)? And what they do about handling such data?

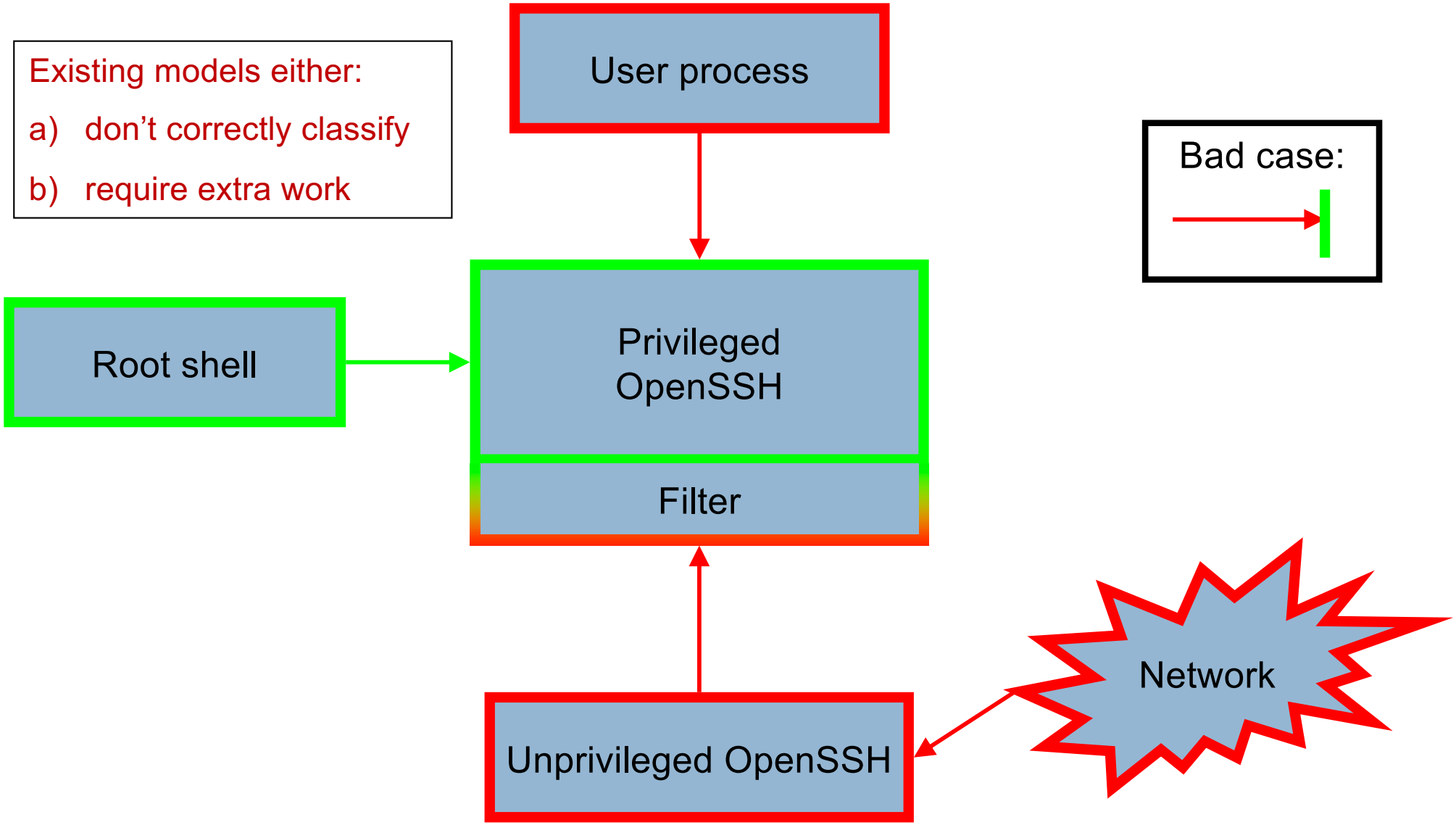
Different integrity model: CW-Lite

- 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

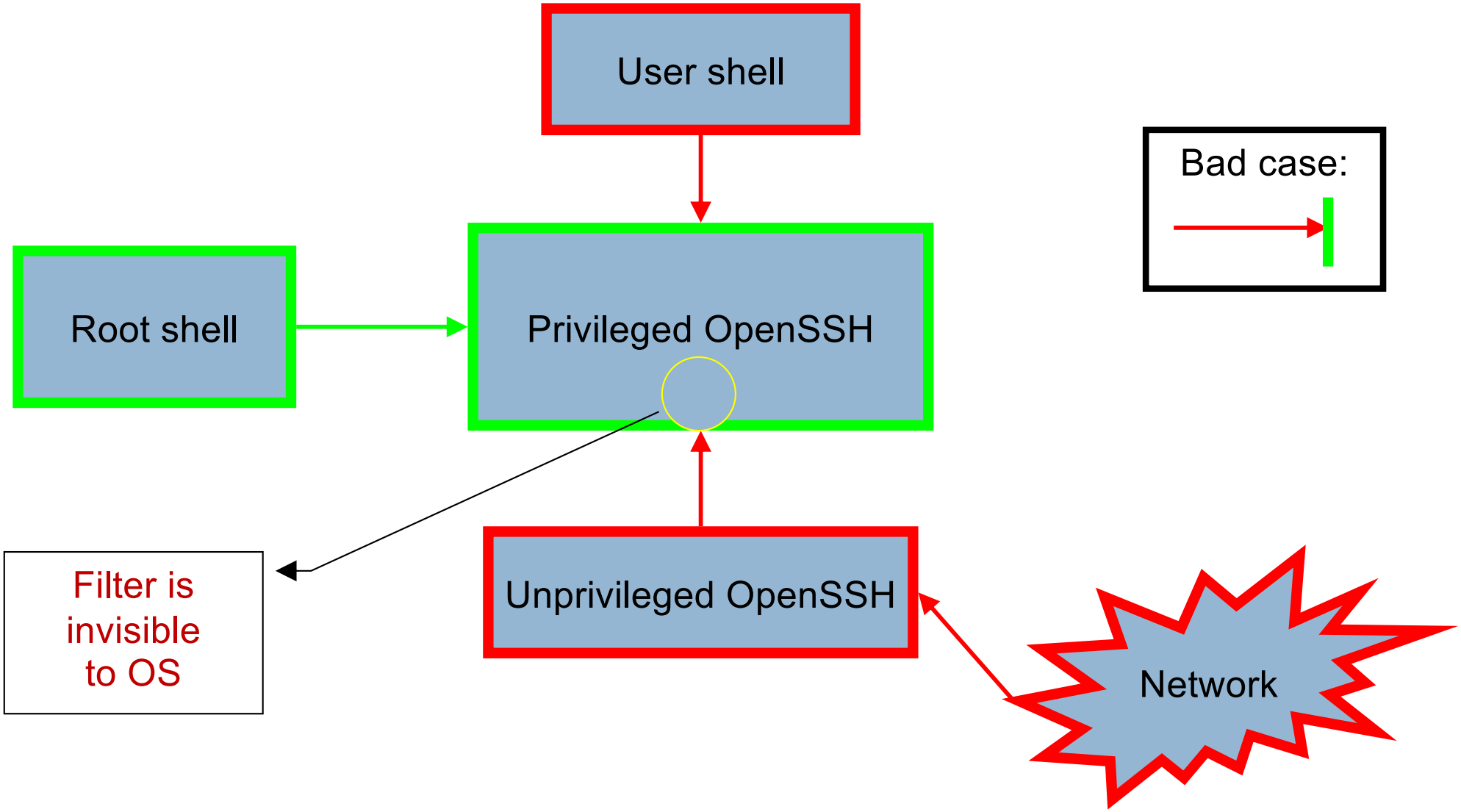
Legal vs. illegal flows



Legal vs. illegal flows



The OS View: Process info-flow

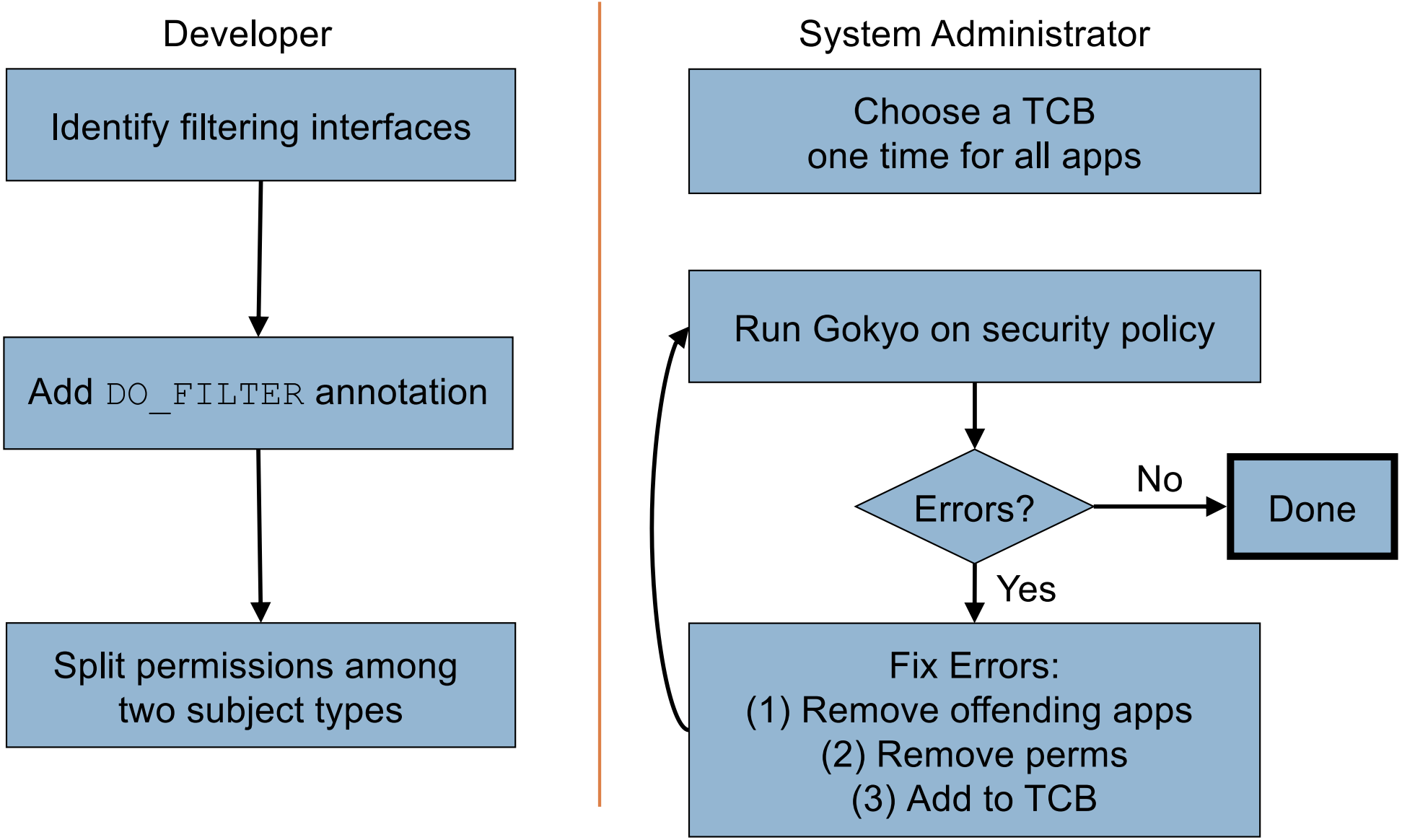


Enabling filtering subject types

- Linux (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



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. Find resources that require filtering
3. Find where programs access such resource
4. Add filters

Take Away

- In a secure system, we must protect data integrity
  - Even a prerequisite to secrecy protection
- Types of integrity – biased toward security or function
  - Functional: least privilege; Security: information flow
- Integrity models
  - Least privilege, Biba, LOMAC, Clark-Wilson, CW-Lite
- *Need to develop approaches to design mandatory protection system for integrity – for function and security*

Questions