

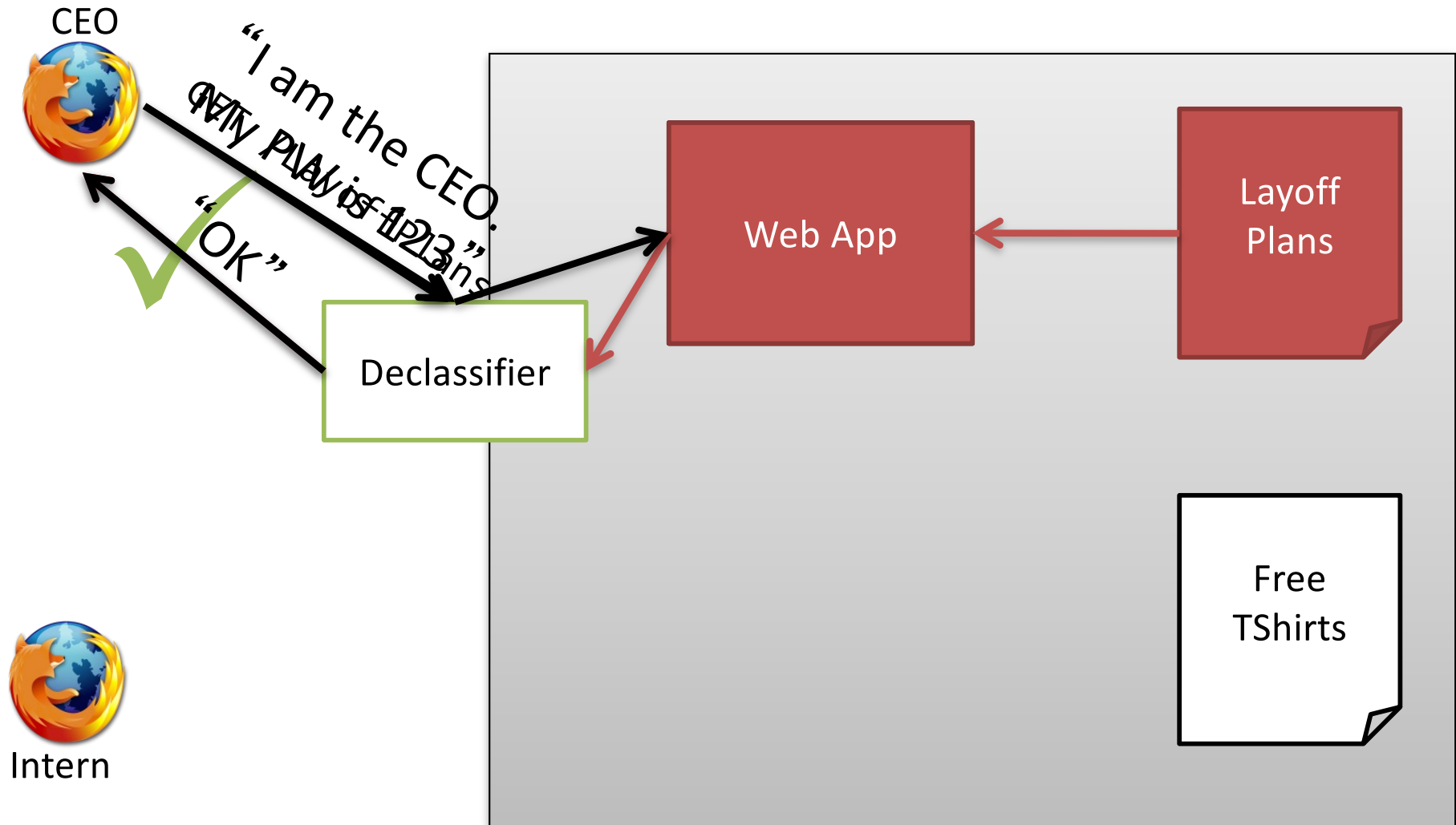
Information Flow Control For Standard OS Abstractions

Max Krohn, Alex Yip, Micah Brodsky,
Natan Cliffer, Frans Kaashoek,
Eddie Kohler, Robert Morris

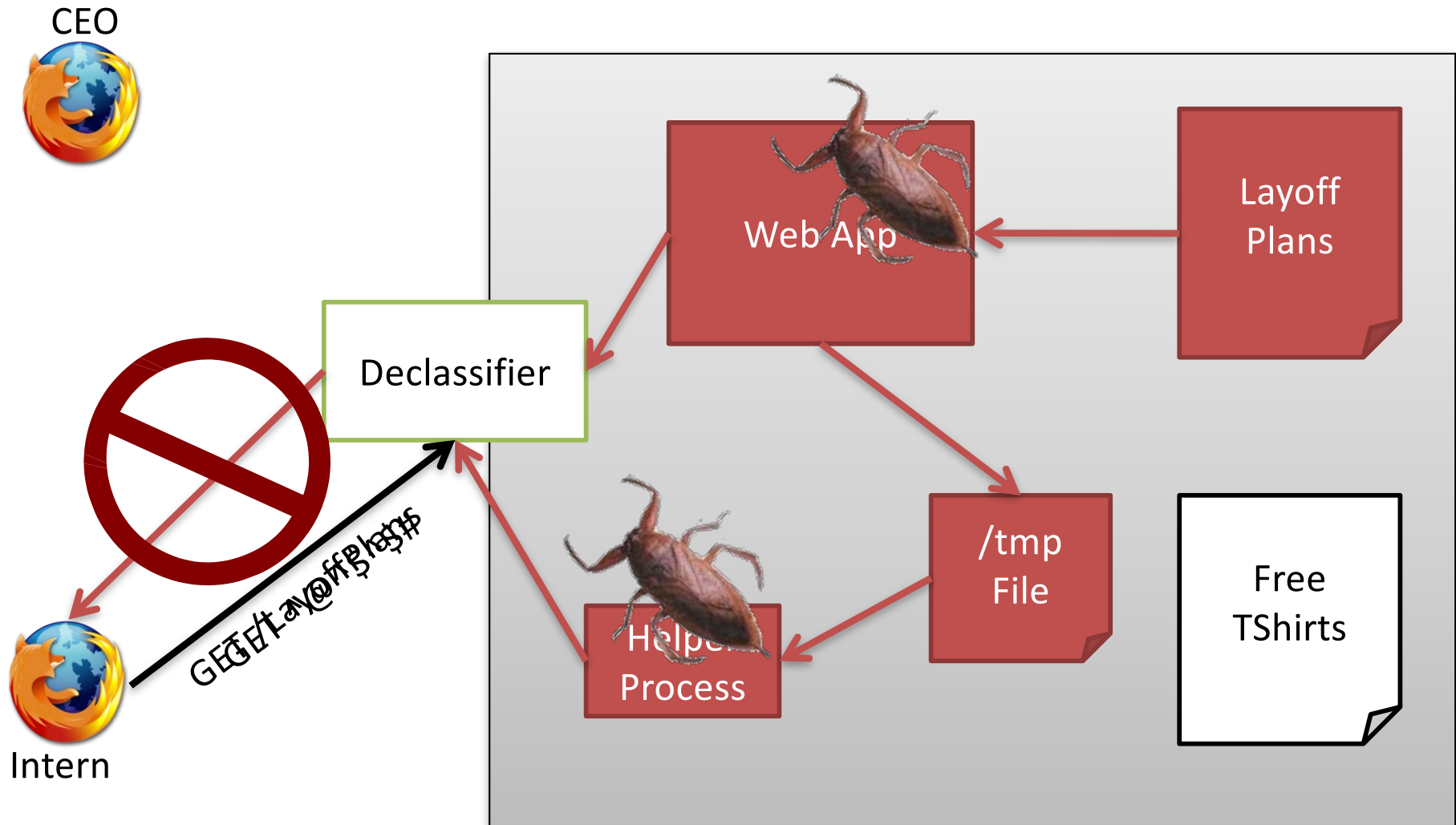
Vulnerabilities in Websites → Exploits

- Web software is buggy
- Attackers find and exploit these bugs
- Data is stolen / Corrupted
 - “USAJobs.gov hit by Monster.com attack, 146,000 people affected”
 - “UN Website is Defaced via SQL Injection”
 - “Payroll Site Closes on Security Worries”
 - “Hacker Accesses Thousands of Personal Data Files at CSU Chico”
 - “FTC Investigates PETCO.com Security Hole”
 - “Major Breach of UCLA’s Computer Files”
 - “Restructured Text Include Directive Does Not Respect ACLs”

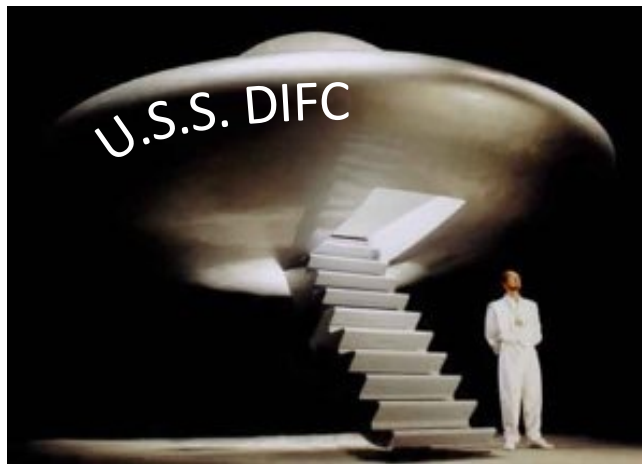
Decentralized Information Flow Control (DIFC)



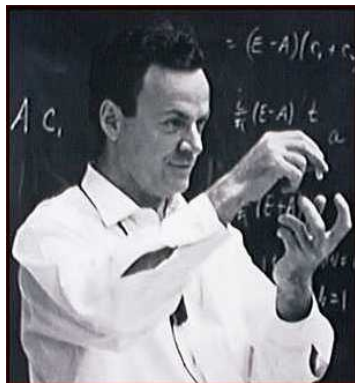
Decentralized Information Flow Control (DIFC)



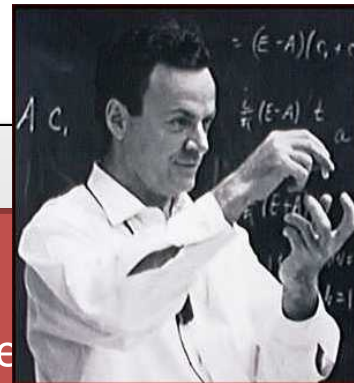
Why is DIFC a cult?



Who Needs to Understand DIFC?



Declassifier



We

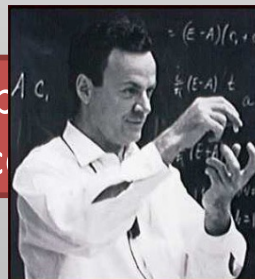
Layoff
Plans

/tmp
File

Free
TShirts



Help
Proc



Why is Today's DIFC **DIFfiCult**?

- Label systems are complex
- Unexpected program behavior
- Cannot reuse existing code
 - Drivers, SMP support, standard libraries

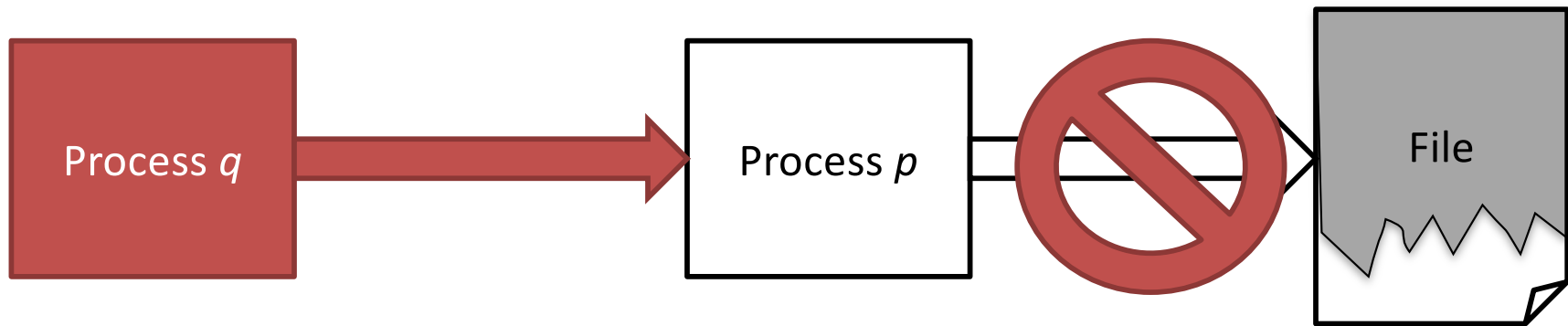
Unexpected Program Behavior (Unreliable Communication)



"Fire Alice, Bob, Charlie, Doug, Eddie, Frank, George, Hilda, Ilya..."



Unexpected Program Behavior (Mysterious Failures)



Solution/Outline

1. Flume: Solves DIFC Problems

- User-level implementation of DIFC on Linux
- Simple label system
- Endpoints: Glue Between Unix API and Labels

2. Application + Evaluation

- Real Web software secured by Flume

Outline

1. Flume: Solves DIFC Problems

- User-level implementation of DIFC on Linux
- Simple label system
- Endpoints: Glue Between Unix API and Labels

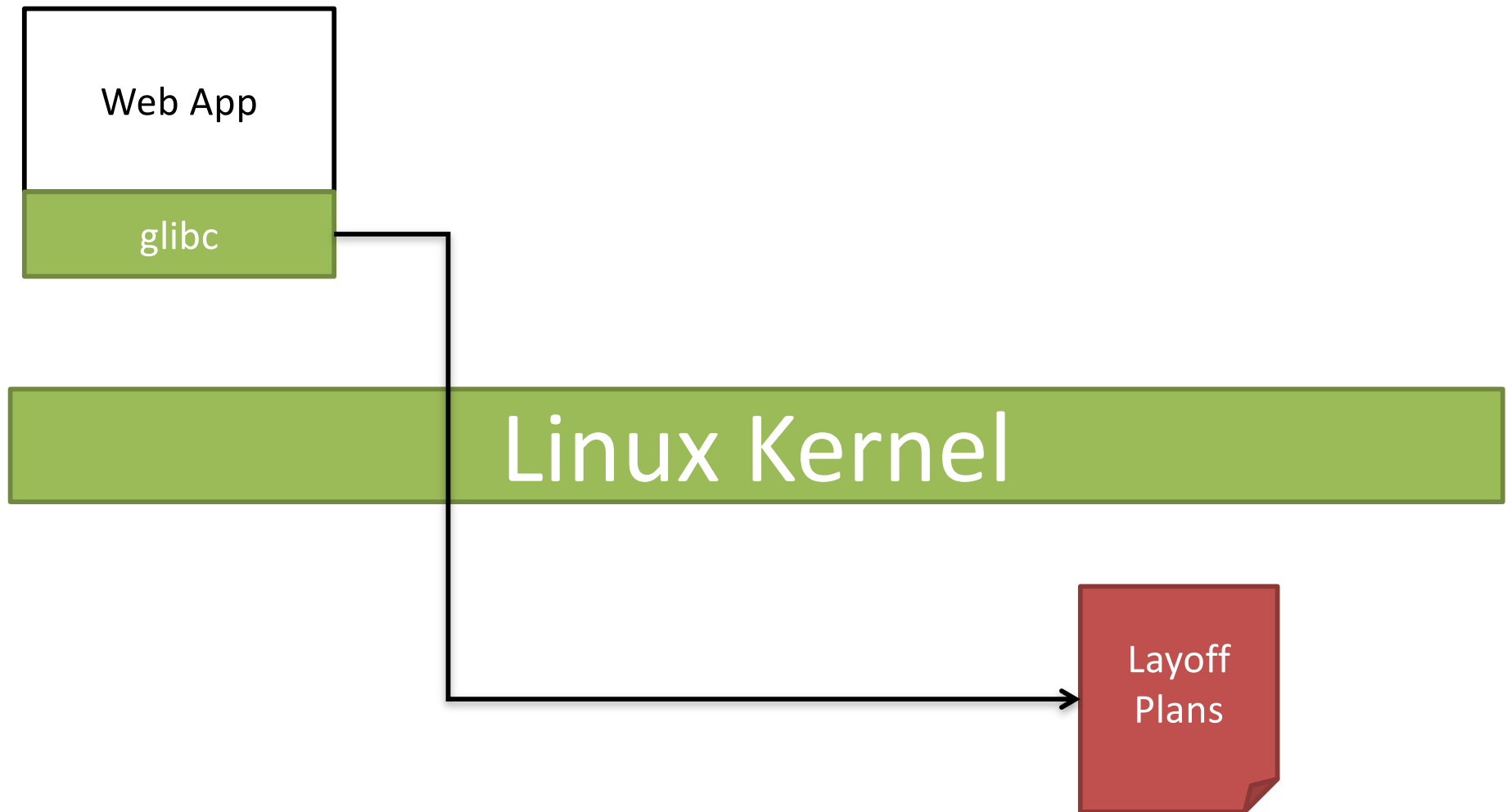
2. Application + Evaluation

Flume Implementation

- Goal: User-level implementation
 - `apt-get install flume`
- Approach:
 - System Call Delegation [*Ostia* by Garfinkel et al, 2003]
 - Use Linux 2.6 (or OpenBSD 3.9)

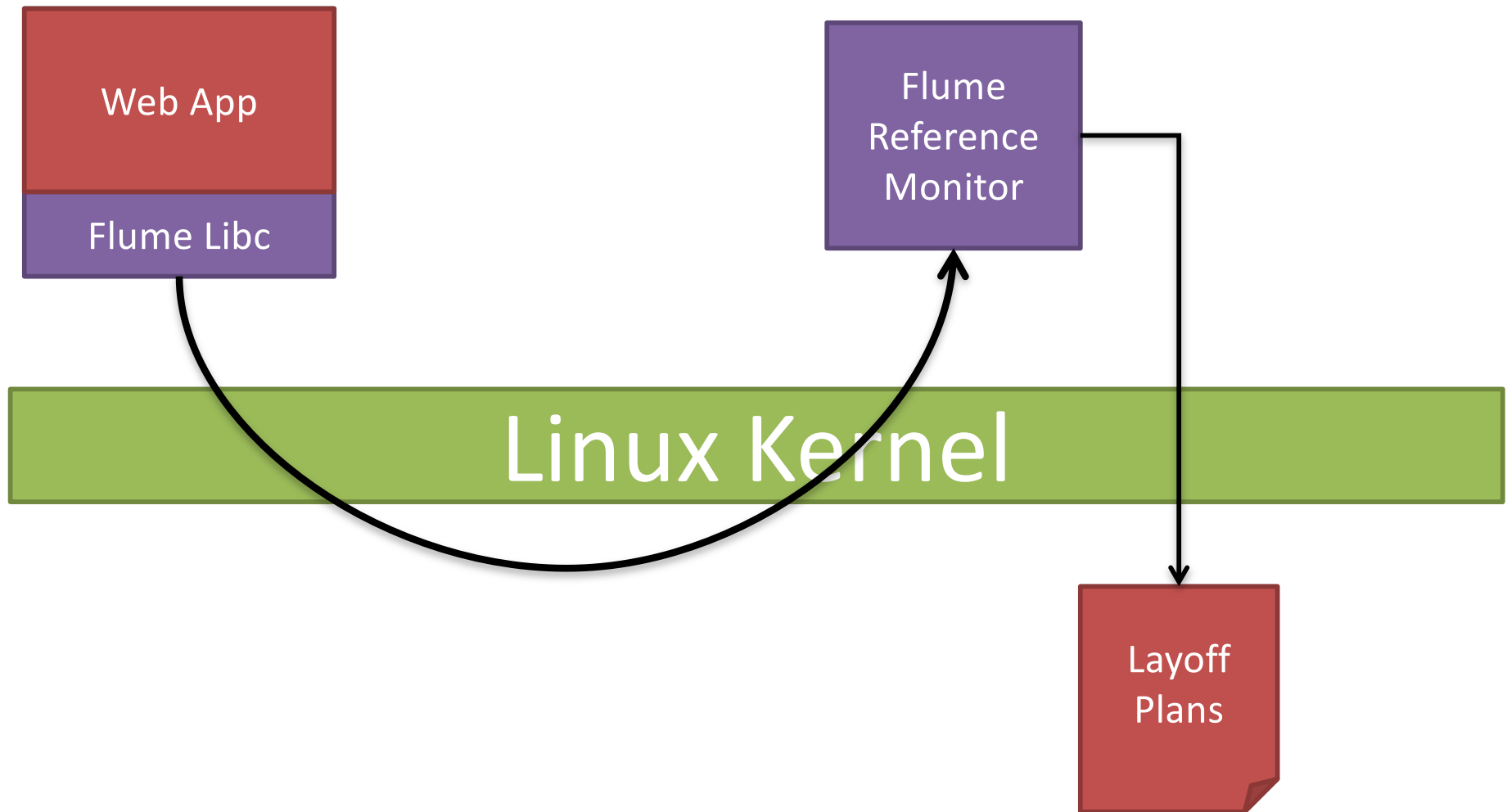
System Call Delegation

```
open("/hr/LayoffPlans", O_RDONLY);
```

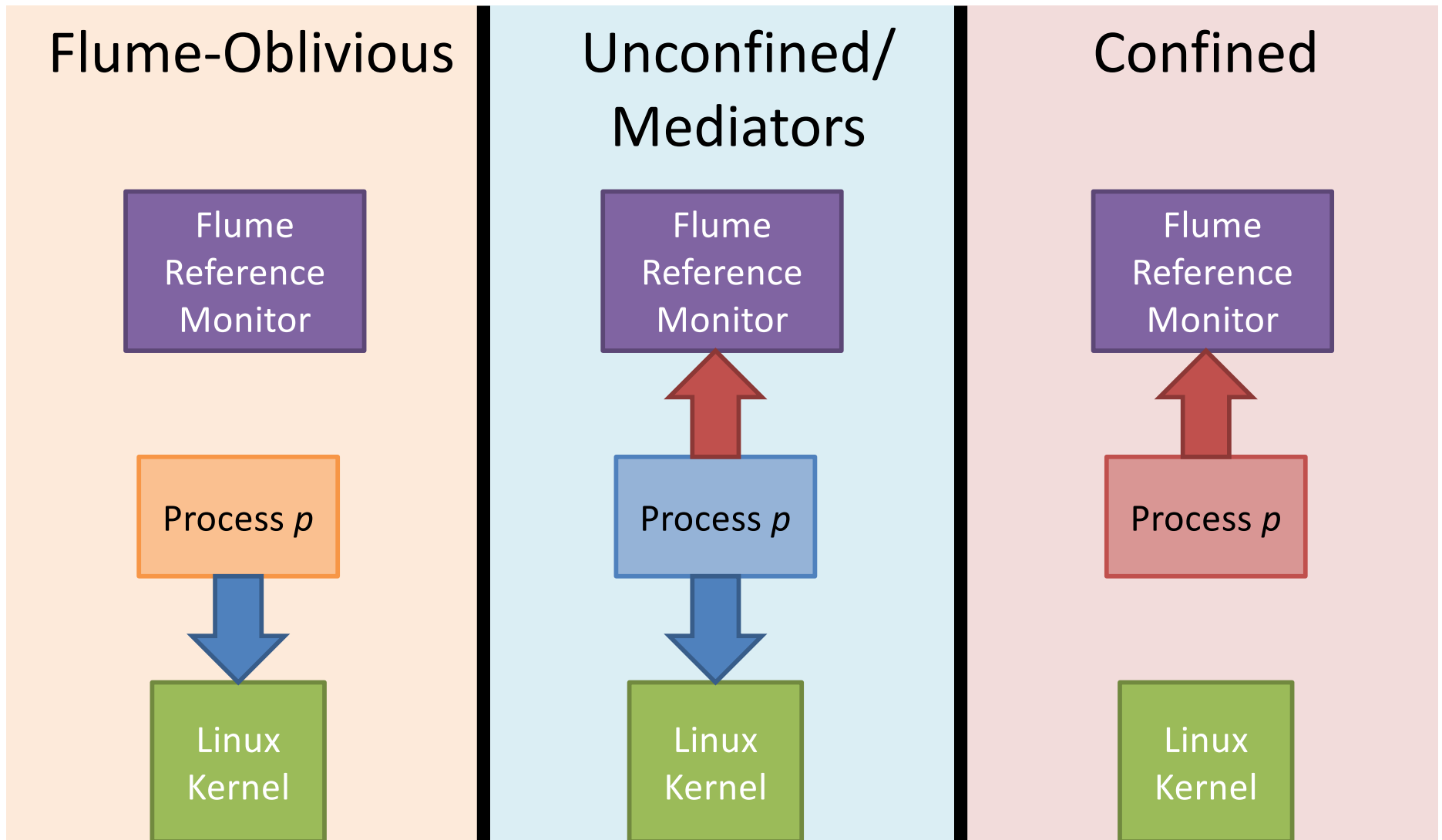


System Call Delegation

```
open("/hr/LayoffPlans", O_RDONLY);
```



Three Classes of Processes



Outline

1. Flume: Solves DIFC Problems

- User-level implementation of DIFC on Linux
- Simple label system
- Endpoints: Glue Between Unix API and Labels

2. Application + Evaluation

Information Flow Control (IFC)

- Goal: track which secrets a process has seen
- Mechanism: each process gets a *secrecy label*
 - Label summarizes which categories of data a process is assumed to have seen

– Examples:

- { “Financial Reports” }
- { “HR Documents” }
- { “Financial Reports” and “HR Documents” }



Tags + Labels

Process p

$S_p \models \{ \text{Finance}, \text{HR} \}$
 $DD_p = \{ \text{HR} \}$

```
change_label({Finance});
tag_get_label(create_tag());
change_label({HR});
```

Any process can add R);

change_label({HR});

DIFC Rule: A process can create a new tag; gets ability to declassify it.

in action.

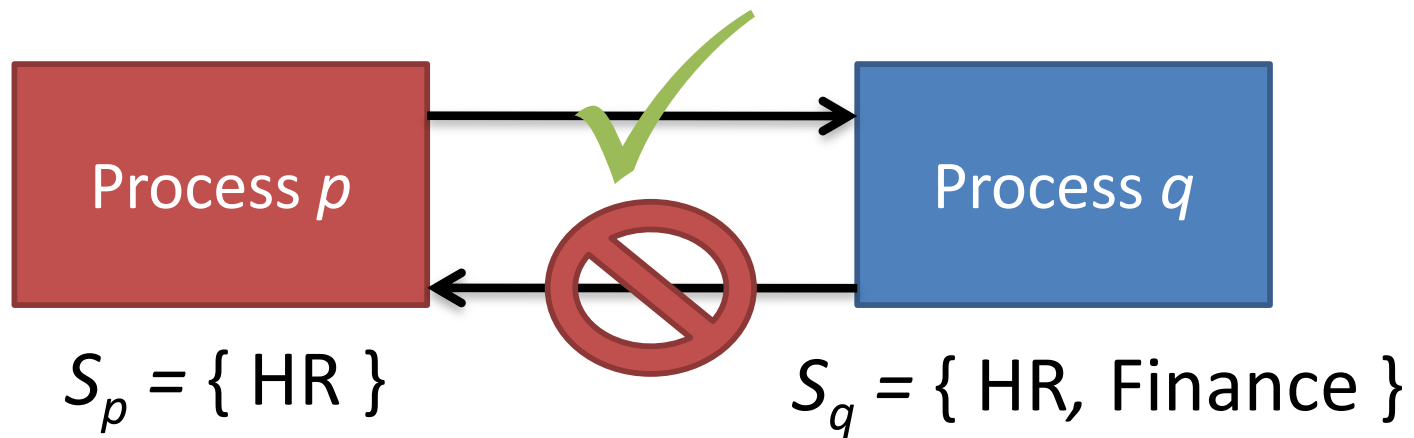
Universe of Tags:

HR

Finance

SecretProjects

Communication Rule



p can send to q iff $S_p \subseteq S_q$

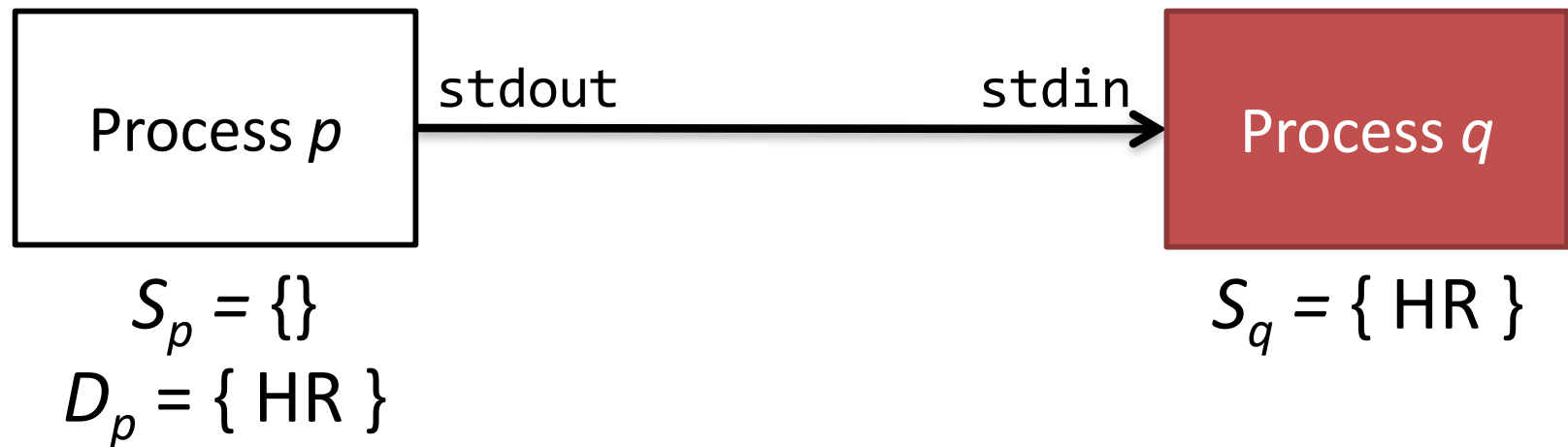
Outline

1. Flume: Solves DIFC Problems

- User-level implementation of DIFC on Linux
- Simple label system
- Endpoints: Glue Between Unix API and Labels

2. Application + Evaluation

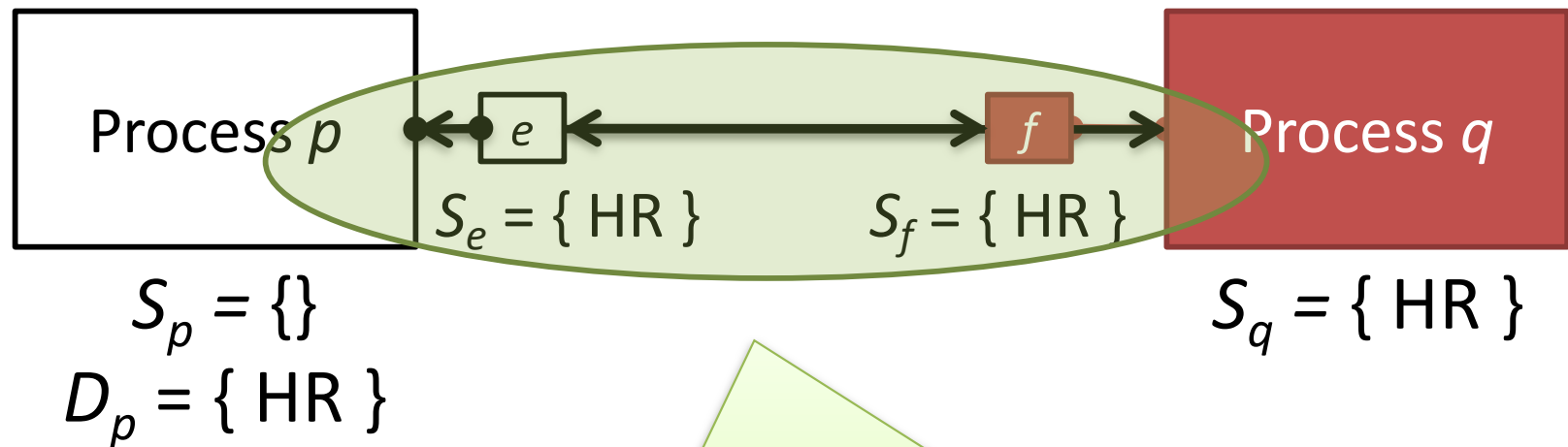
Recall: Communication Problem



“Fire Alice, Bob, Charlie, Doug, Eddie, Frank, George, Hilda, Ilya...”

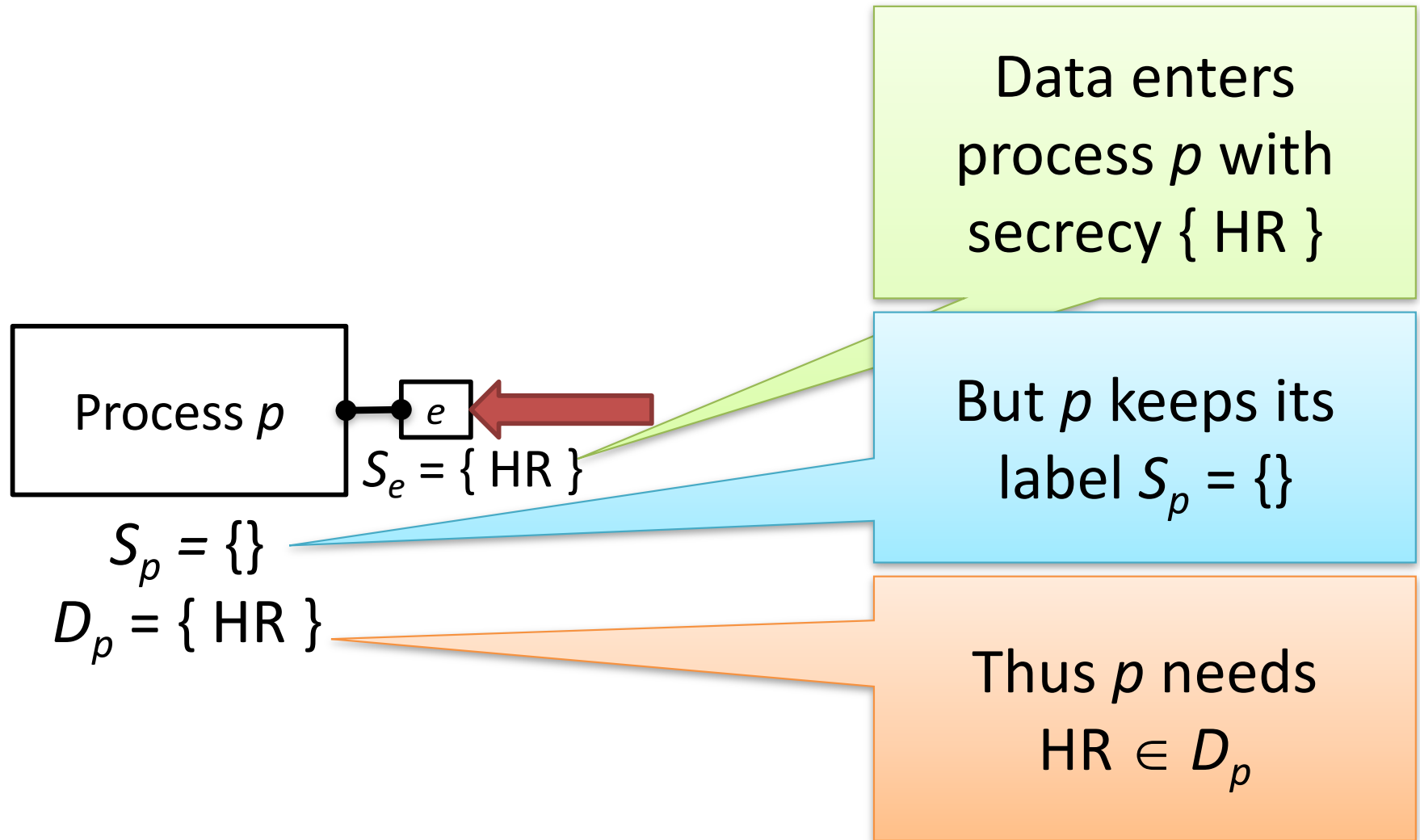
“SLOW DOWN!!”
“I crashed”

New Abstraction: Endpoints



- “Fire Alice, Bob, Charlie, Doug, Eddie, Frank, George, Hilda, Ilya...”
- If $S_e \subseteq S_f$, then allow e to send to f
 - If $S_f \subseteq S_e$, then allow f to send to e
 - If $S_f = S_e$, then allow bidirectional flow
- “SLOW DOWN!!”
- “I crashed”

Endpoints Declassify Data

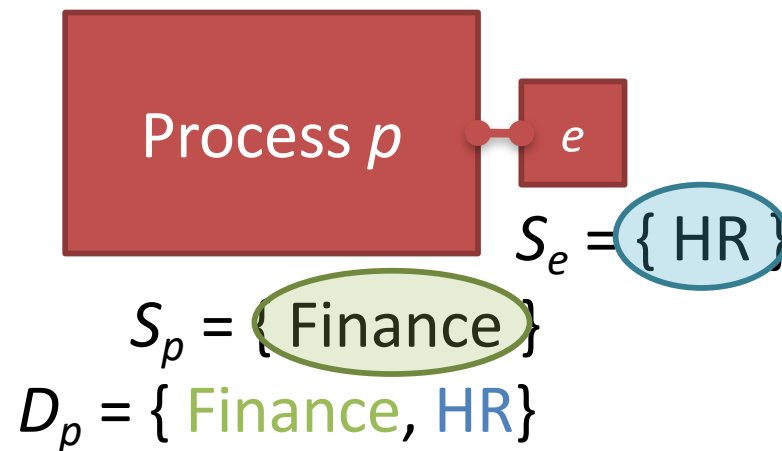


Endpoint Invariant

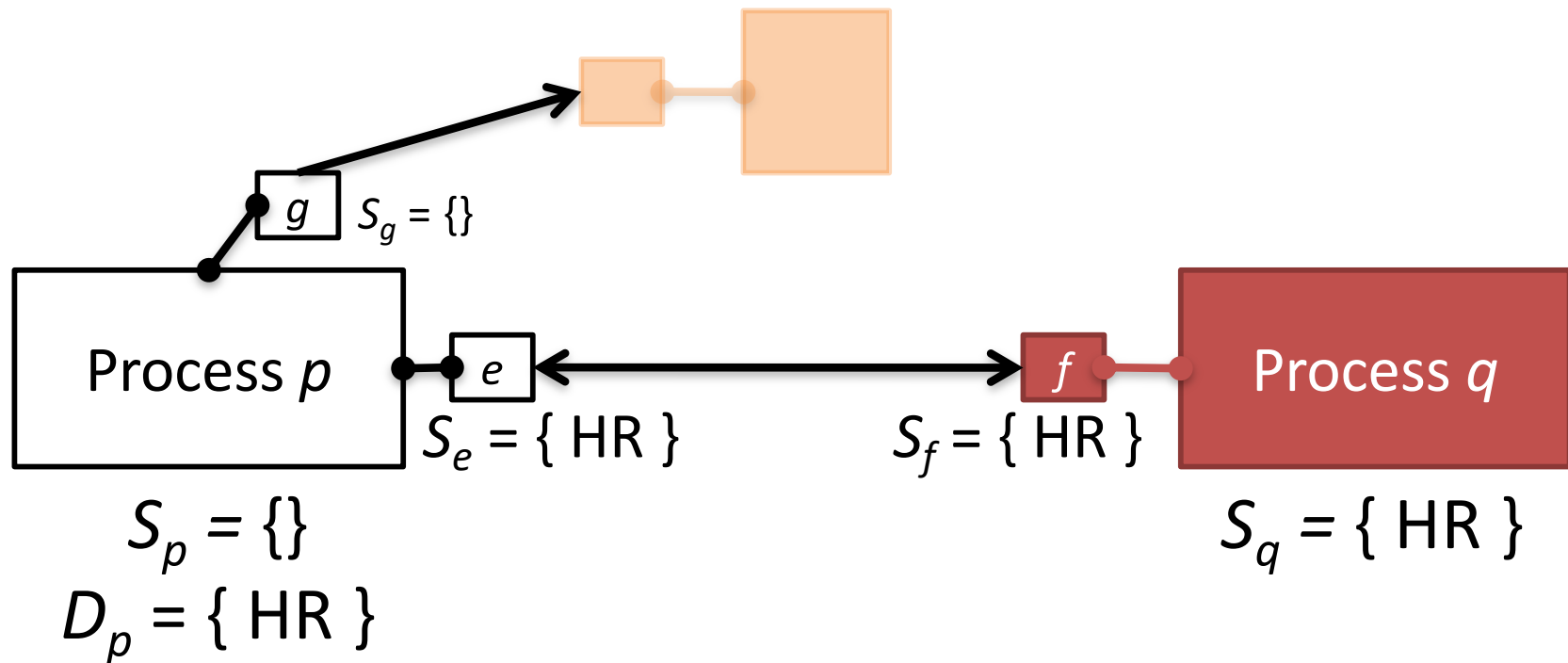
- For any tag $t \in S_p$ and $t \notin S_e$
- Or any tag $t \in S_e$ and $t \notin S_p$
- It must be that $t \in D_p$

Writing

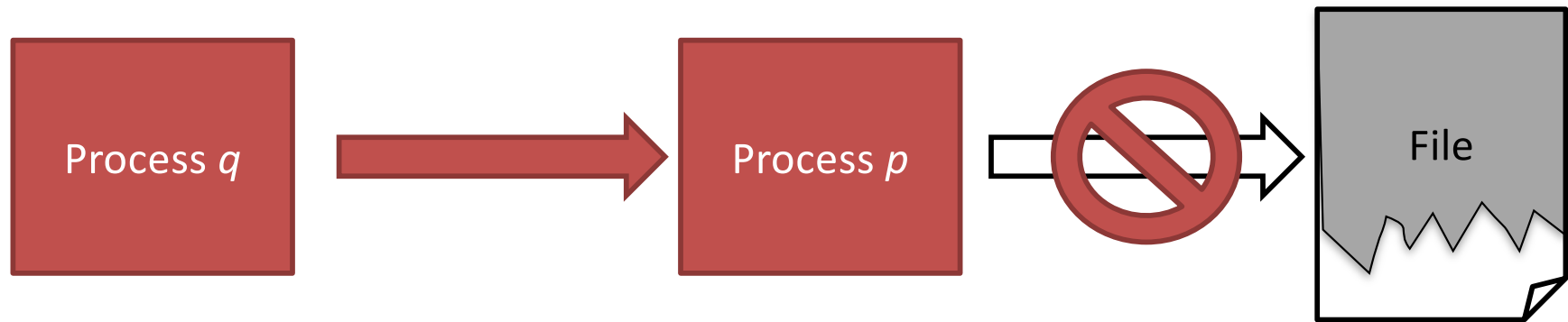
Reading



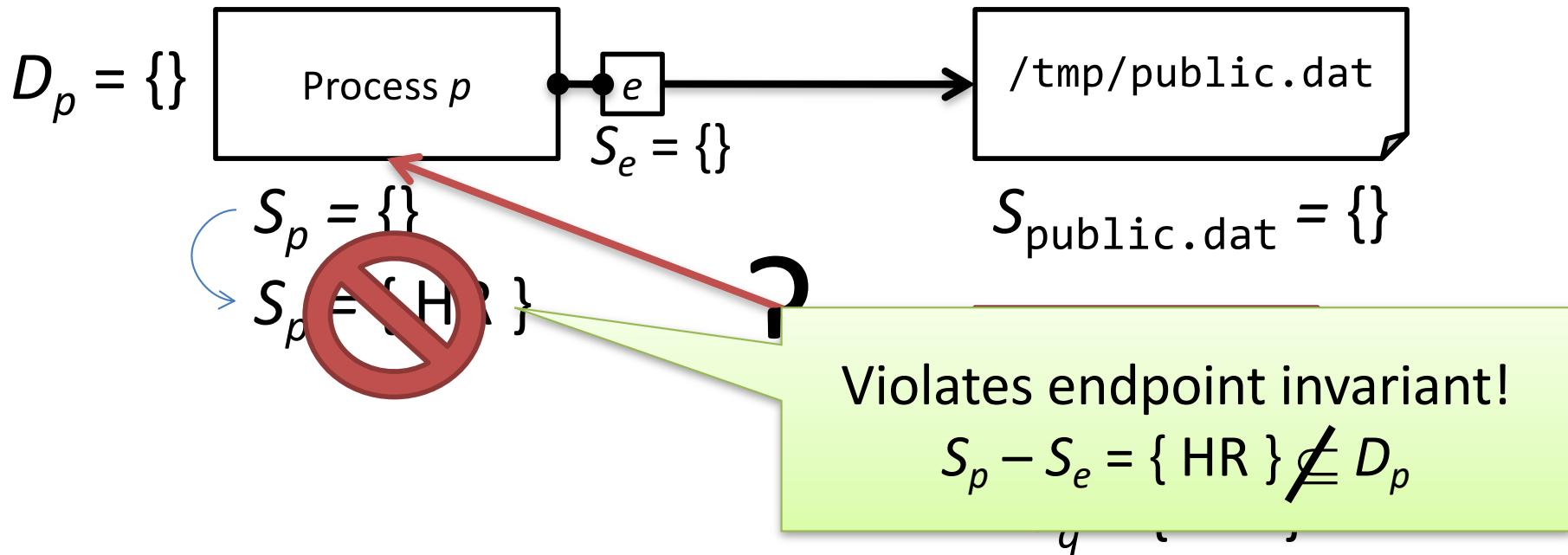
Endpoints Labels Are Independent



Recall: Mysterious Failures

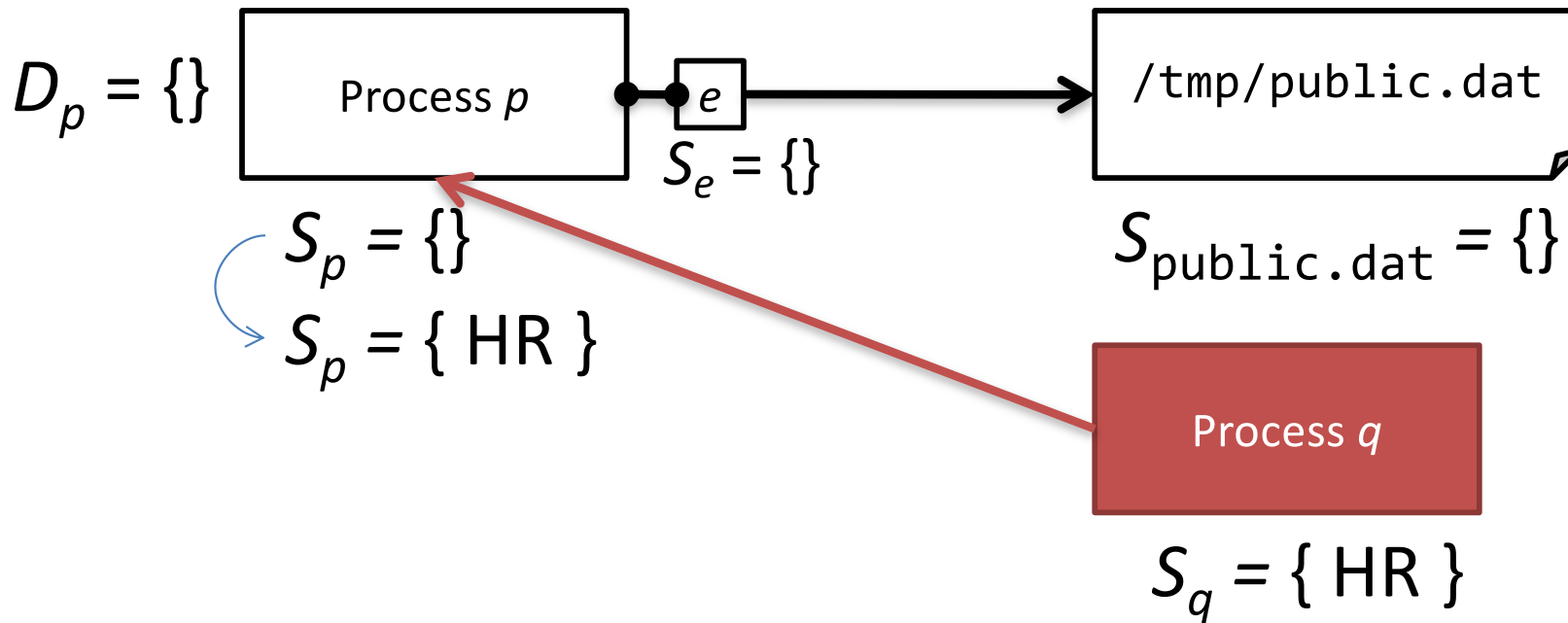


Endpoints Reveal Errors Eagerly



~~$\rightarrow \text{open}("/tmp/public.dat", O_WRONLY);$
 $\rightarrow \text{change_label}(\{HR\})$~~

Endpoints Reveal Errors Eagerly



```
→fd = open("/tmp/public.dat", O_WRONLY);  
→close(fd);  
→change_label({HR})
```

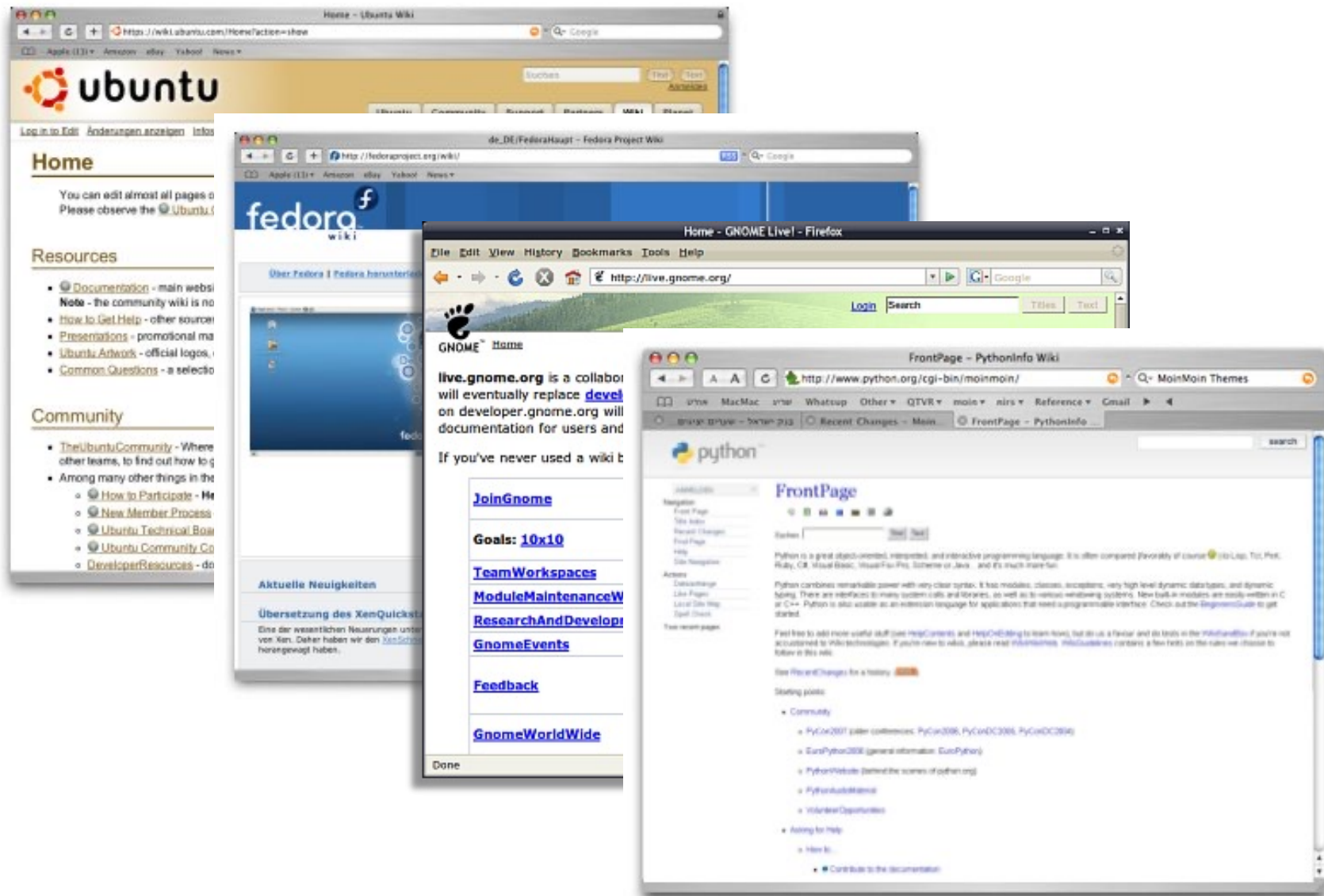
Outline

1. Flume: Solves DIFC Problems
2. Application + Evaluation

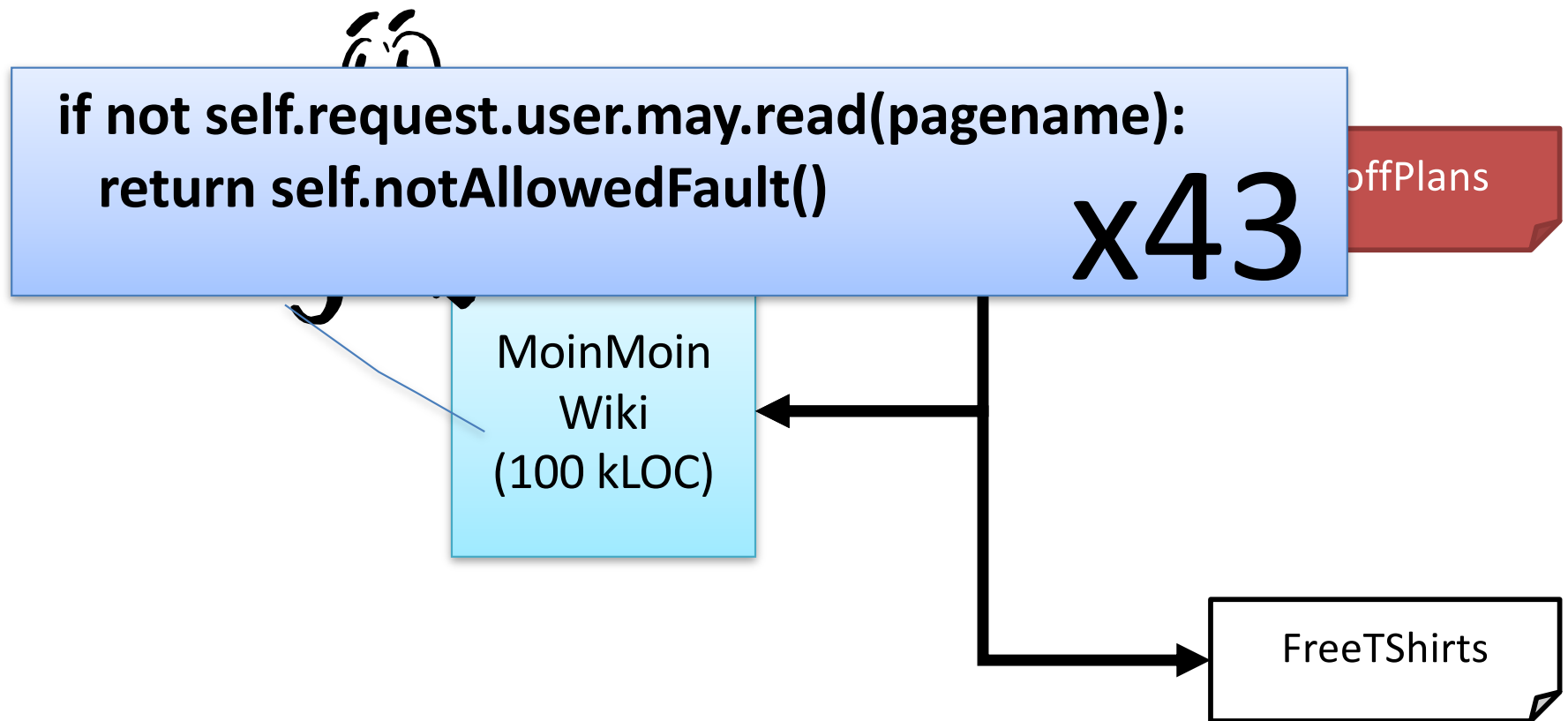
Questions for Evaluation

- Does Flume allow adoption of Unix software?
- Does Flume solve security vulnerabilities?
- Does Flume perform reasonably?

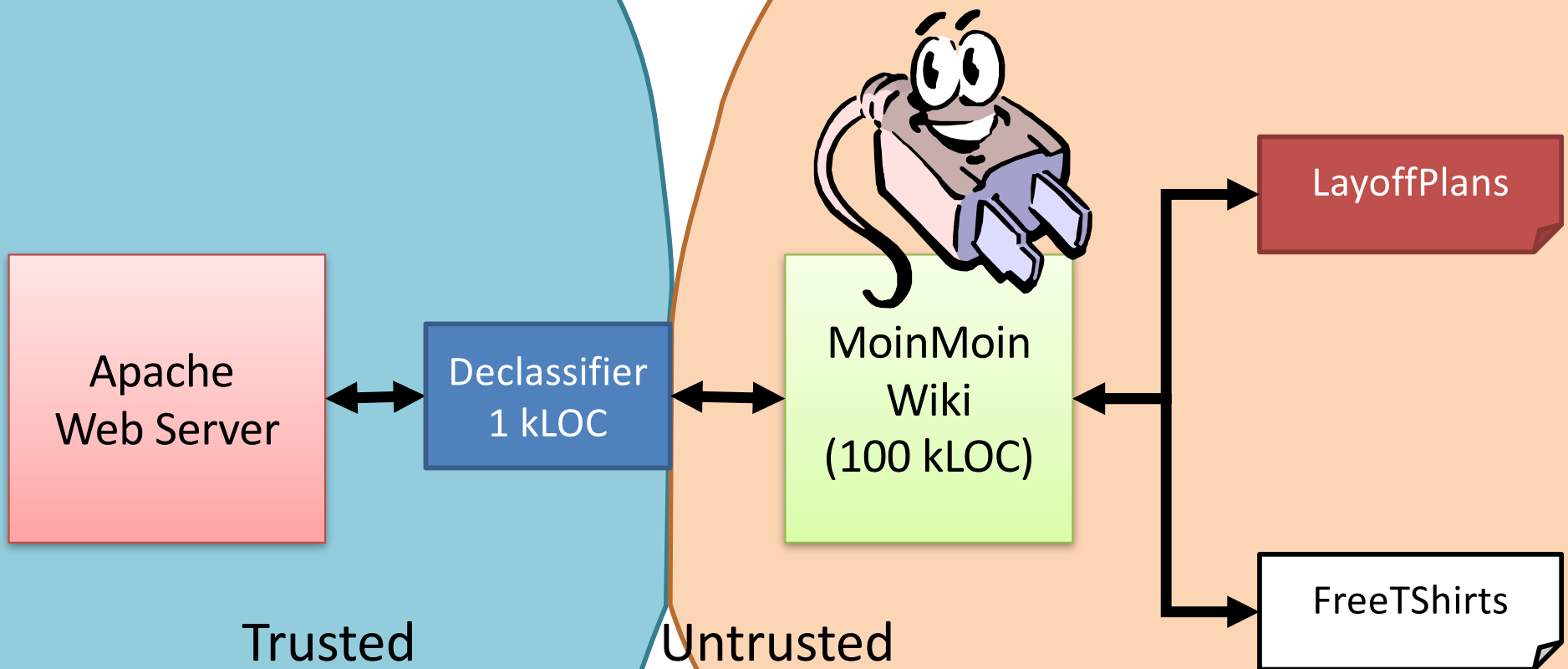
Example App: MoinMoin Wiki



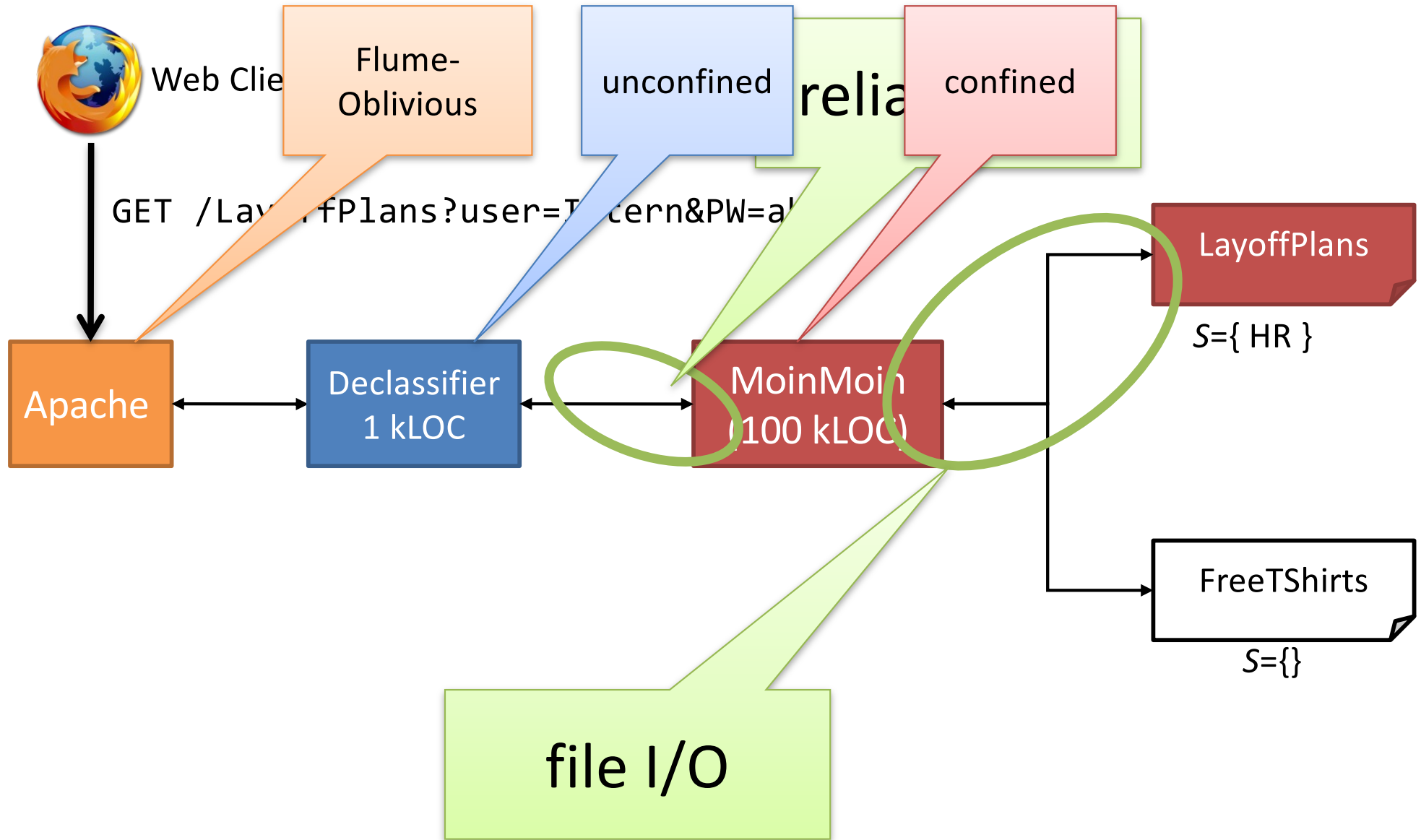
How Problems Arise...



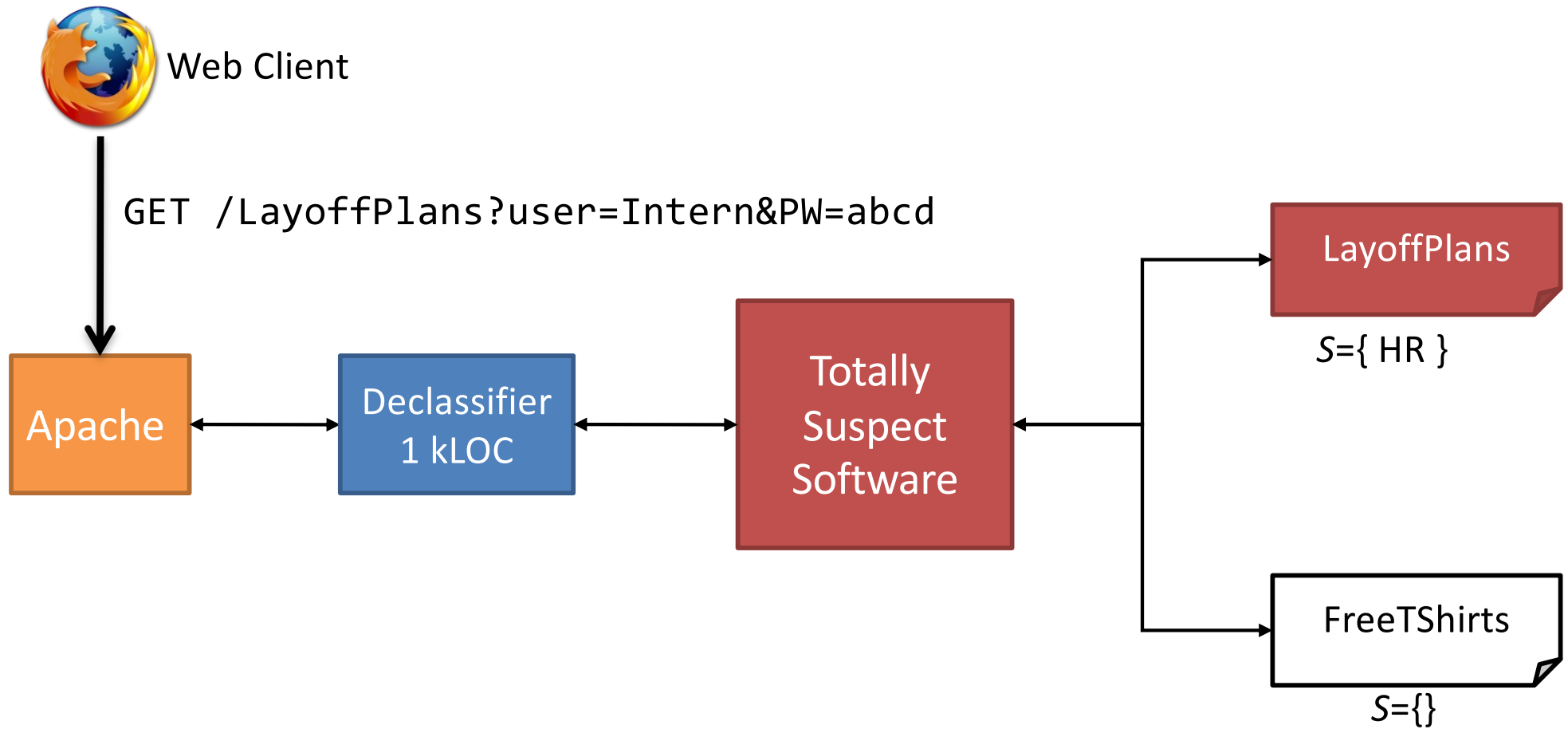
MoinMoin + DIFC



FlumeWiki



Future Work



Results

- Does Flume allow adoption of Unix software?
 - 1,000 LOC launcher/declassifier
 - 1,000 out of 100,000 LOC in MoinMoin changed
 - Python interpreter, Apache, unchanged
- Does Flume solve security vulnerabilities?
 - Without our knowing, we inherited two ACL bypass bugs from MoinMoin
 - Both are not exploitable in Flume's MoinMoin
- Does Flume perform reasonably?
 - Performs within a factor of 2 of the original on read and write benchmarks

Most Related Work

- Asbestos, HiStar: New DIFC OSes
- Jif: DIFC at the language level
- Ostia, Plash: Implementation techniques
- Classical MAC literature (Bell-LaPadula, Biba, Orange Book MAC, Lattice Model, etc.)

Limitations

- Bigger TCB than HiStar / Asbestos
 - Linux stack (Kernel + glibc + linker)
 - Reference monitor (~22 kLOC)
- Covert channels via disk quotas
- Confined processes like MoinMoin don't get full POSIX API.
 - `spawn()` instead of `fork()` & `exec()`
 - `flume_pipe()` instead of `pipe()`

Summary

- DIFC is a challenge to Programmers
- Flume: DIFC in User-Level
 - Preserves legacy software
 - Complements today's programming techniques
- MoinMoin Wiki: Flume works as promised
- Invite you to play around:
<http://flume.csail.mit.edu>

Thanks!

To: ITRI, Nokia, NSF and You

Reasons to Read the Paper

- Generalized security properties
 - Including: Novel integrity policies
- Support for very large labels
- Support for clusters of Flume Machines

Flume's Rule is Fast

- Recall:

p can send to q iff: $S_p - D_p \subseteq S_q \cup D_q$

- To Compute:

- **for each** tag $t \in S_p$:

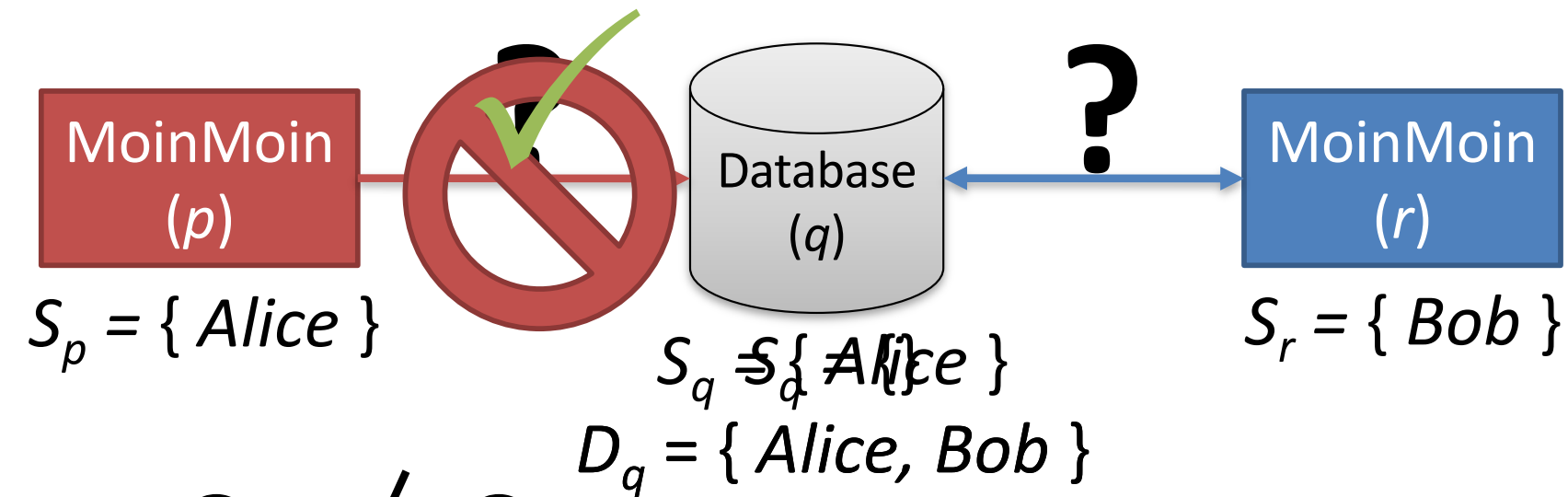
- **If** $t \notin S_q$ **and** $t \notin D_p$ **and** $t \notin D_q$:

- **output** “NO”

- **output** “OK”

- Runs in time proportional to size of S_p .
- *No need to enumerate D_p or D_q !!!*

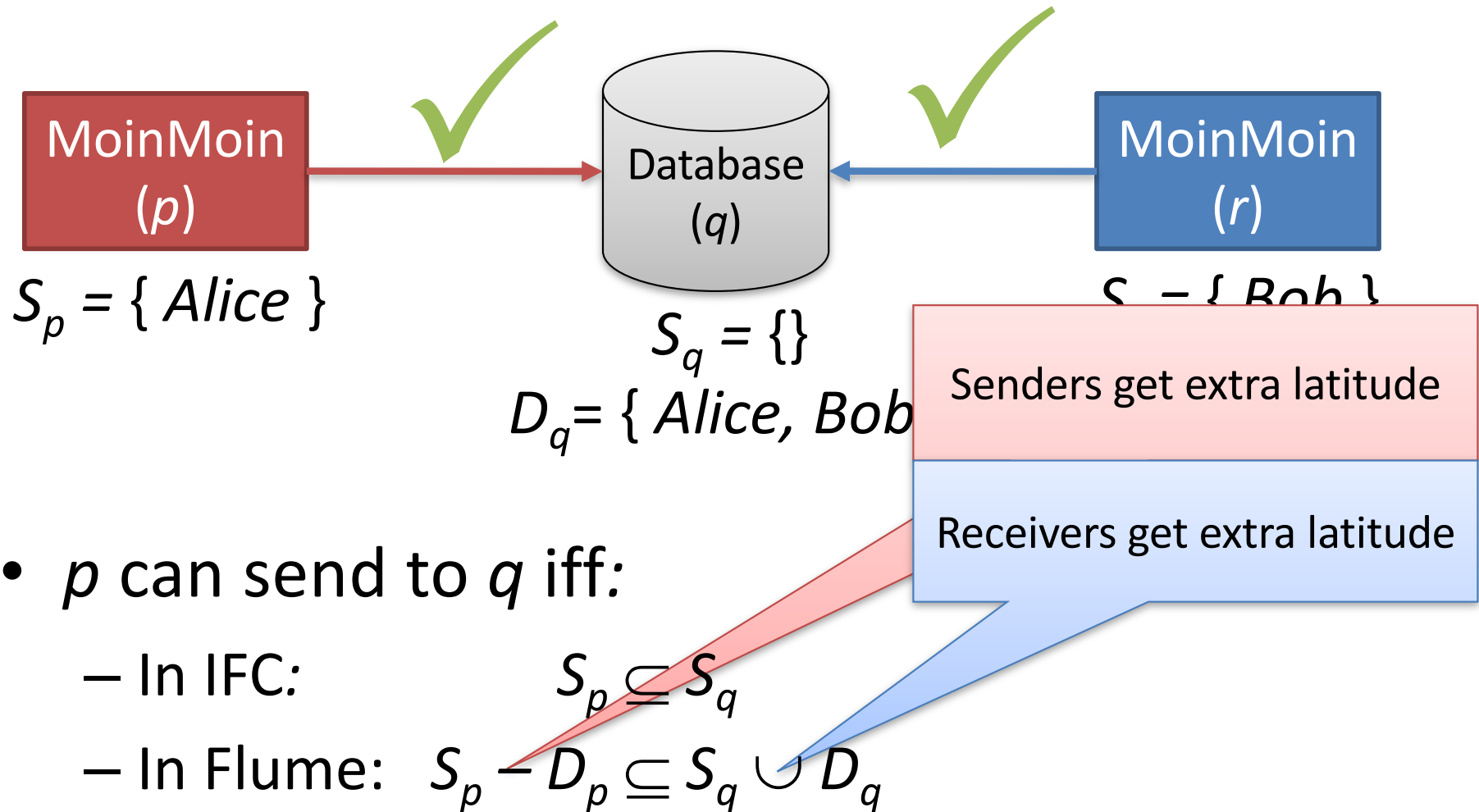
Flume Communication Rule



$$S_p \not\subseteq S_q$$

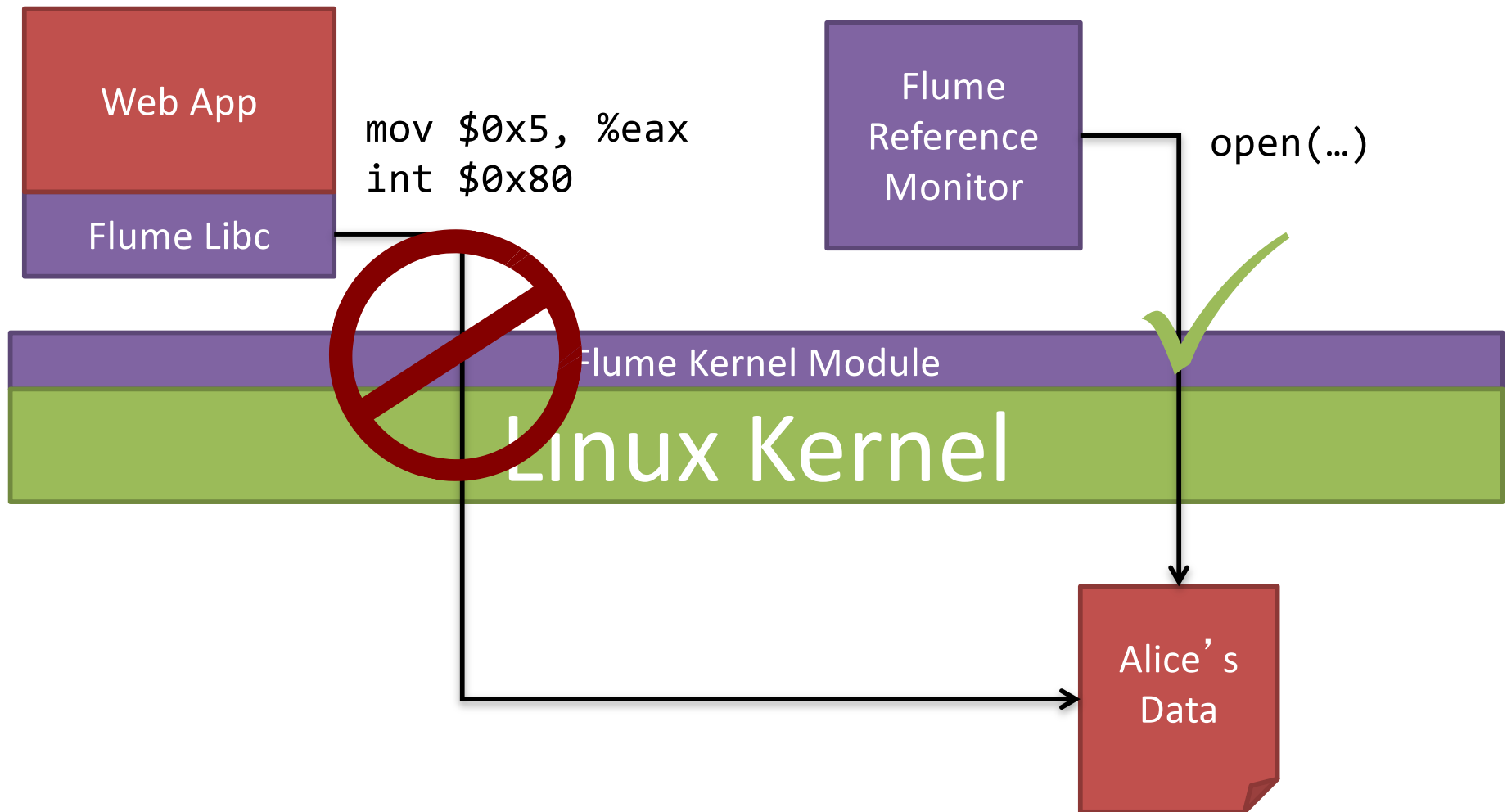
1. q changes to $S_q = \{ Alice \}$
2. p sends to q
3. q changes back to $S_q = \{ \}$

Flume Communication Rule



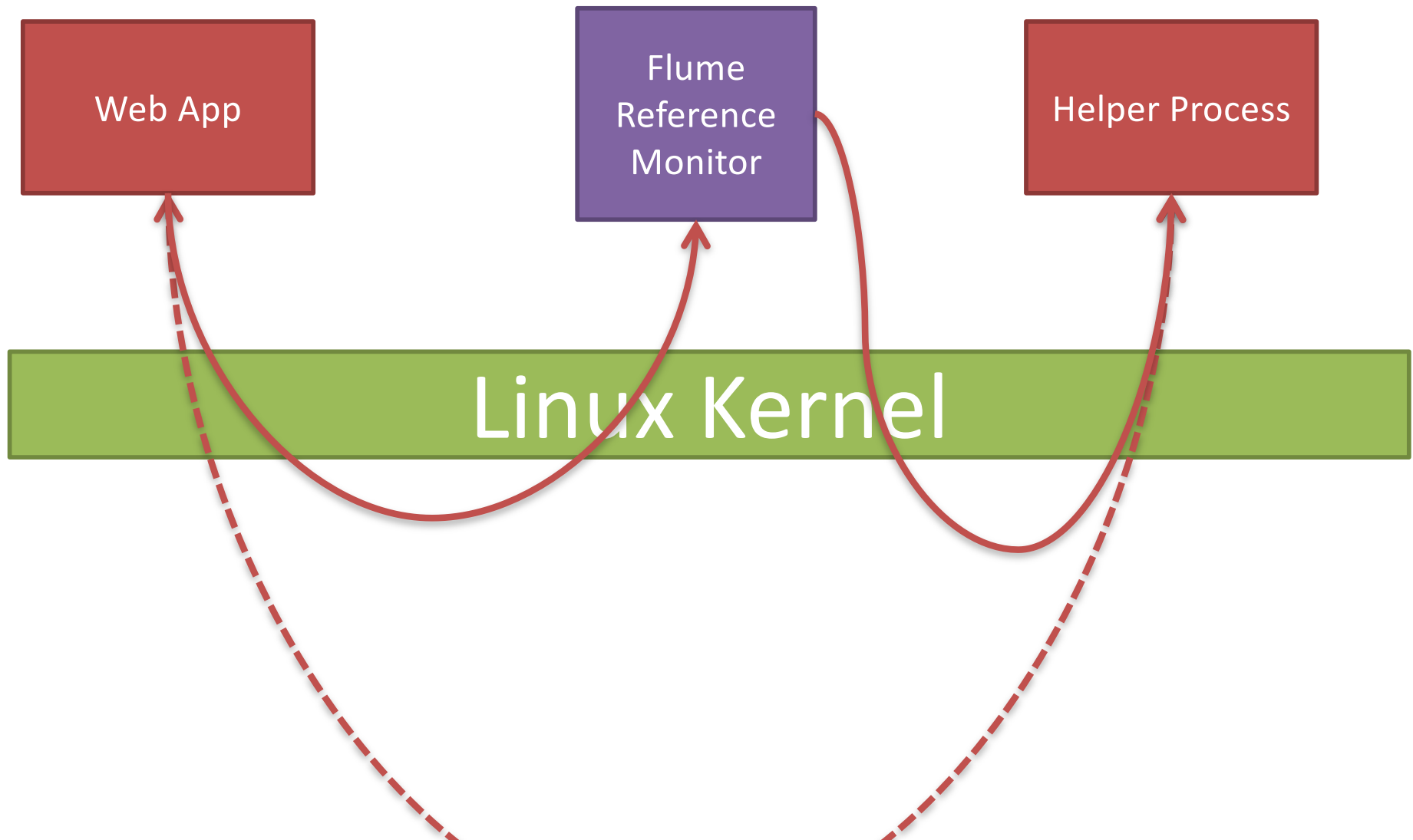
Flume Kernel Module

```
open("/alice/inbox.dat", O_RDONLY);
```

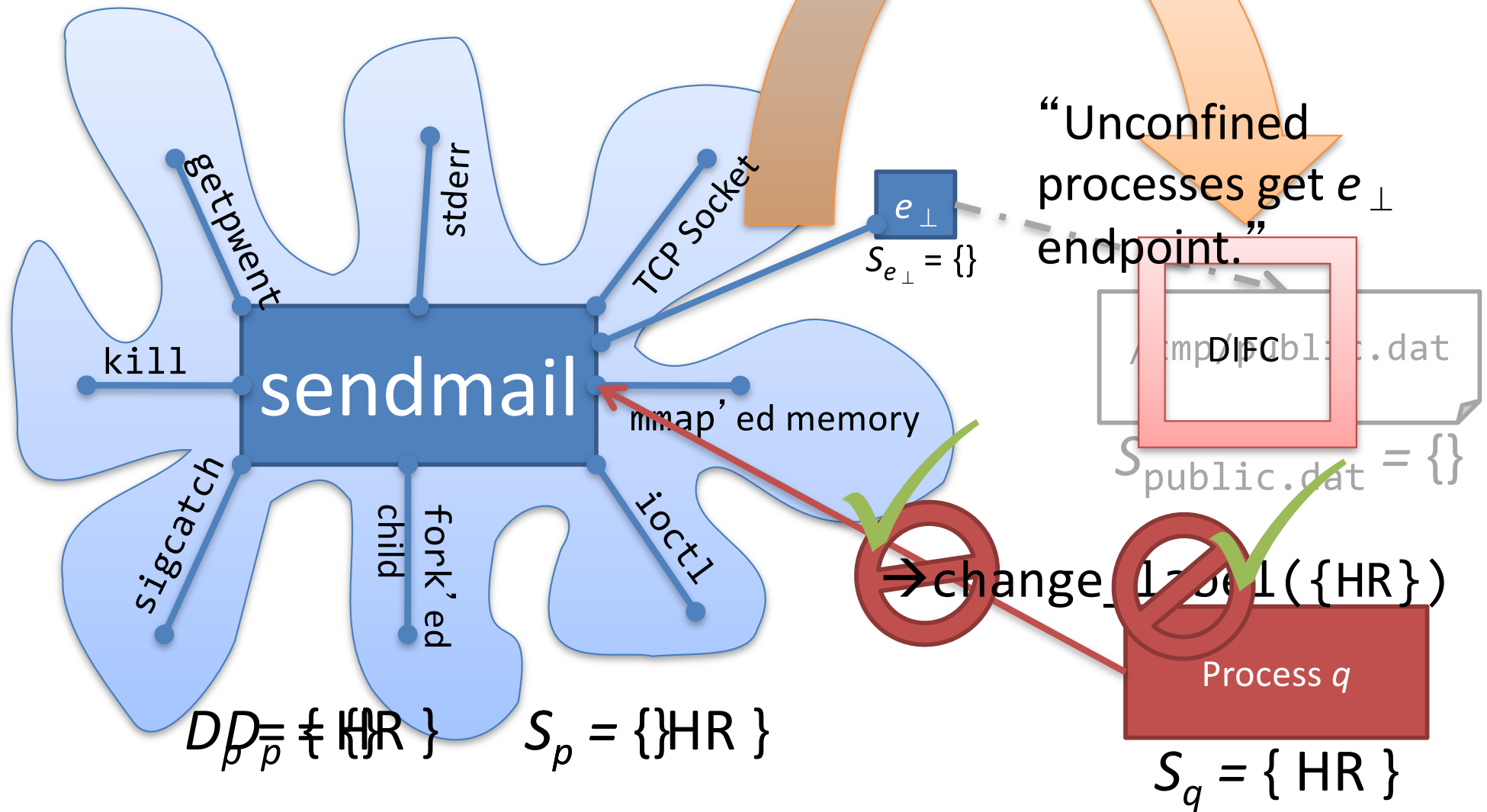


Reference Monitor Proxies Pipes

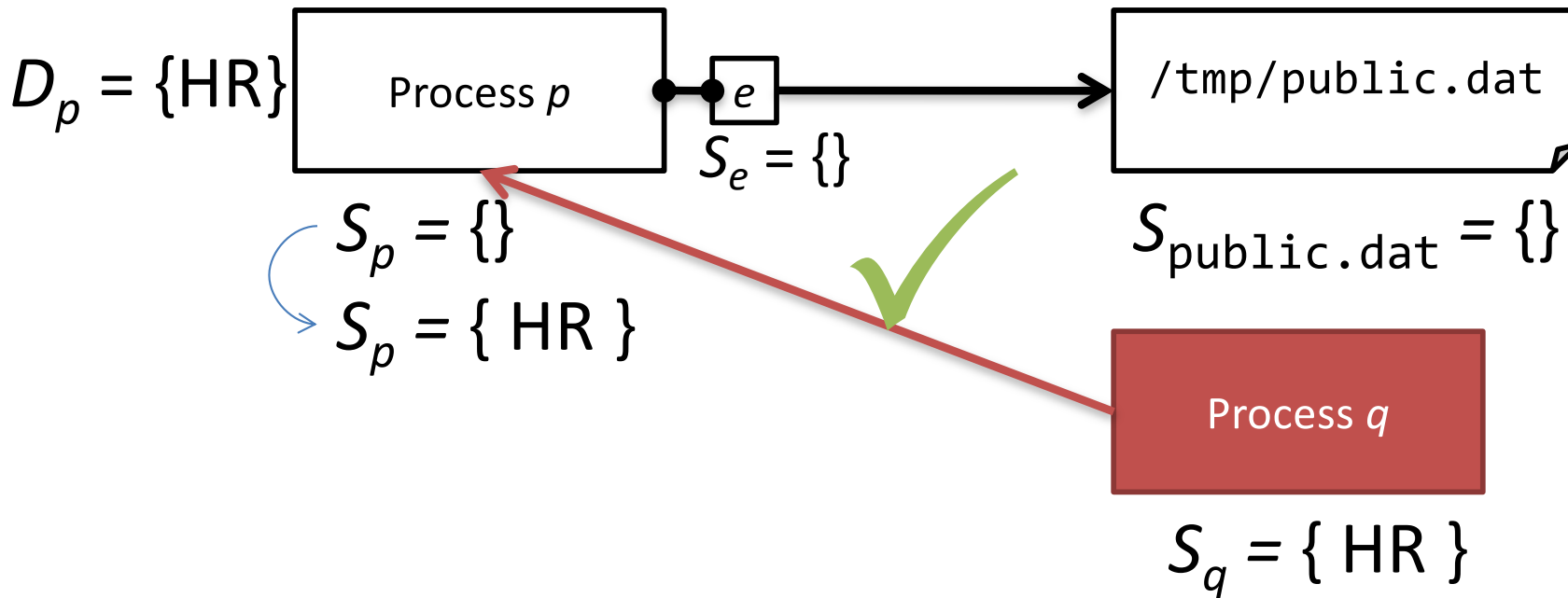
```
write(0, "some data", 10);
```



Unconfined Processes



Endpoints Reveal Errors Eagerly



✓ \rightarrow open("/tmp/public.dat", O_WRONLY);
✓ \rightarrow change_label({HR})

Why Do We Need S_p ?

