Towards Practical Application-level Support for Privilege Separation

Nik Sultana
Illinois Institute of Technology

Henry Zhu
UIUC

Ke Zhong
University of Pennsylvania

Zhilei Zheng
University of Pennsylvania

Ruijie Mao
University of Pennsylvania

Digvijaysinh Chauhan
University of Pennsylvania

Stephen Carrasquillo
University of Pennsylvania

Junyong Zhao
University of Pennsylvania

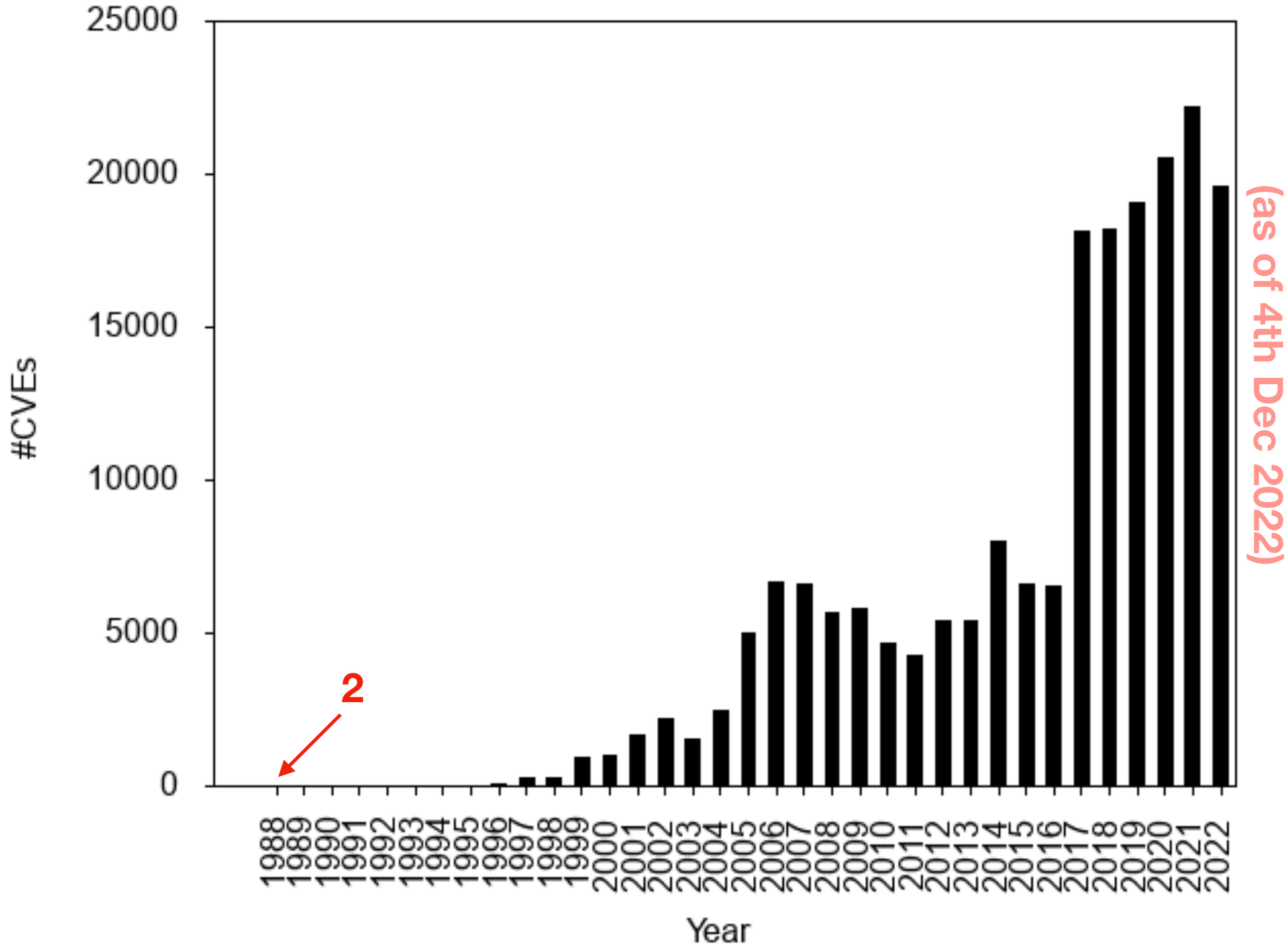
Lei Shi
University of Pennsylvania

Nikos Vasilakis
Brown University & MIT

Boon Thau Loo
University of Pennsylvania

ACSAC'22

Motivation: Software Security



Increased trend in # of CVEs:

Good: we know about problems.

Bad: there are more problems.

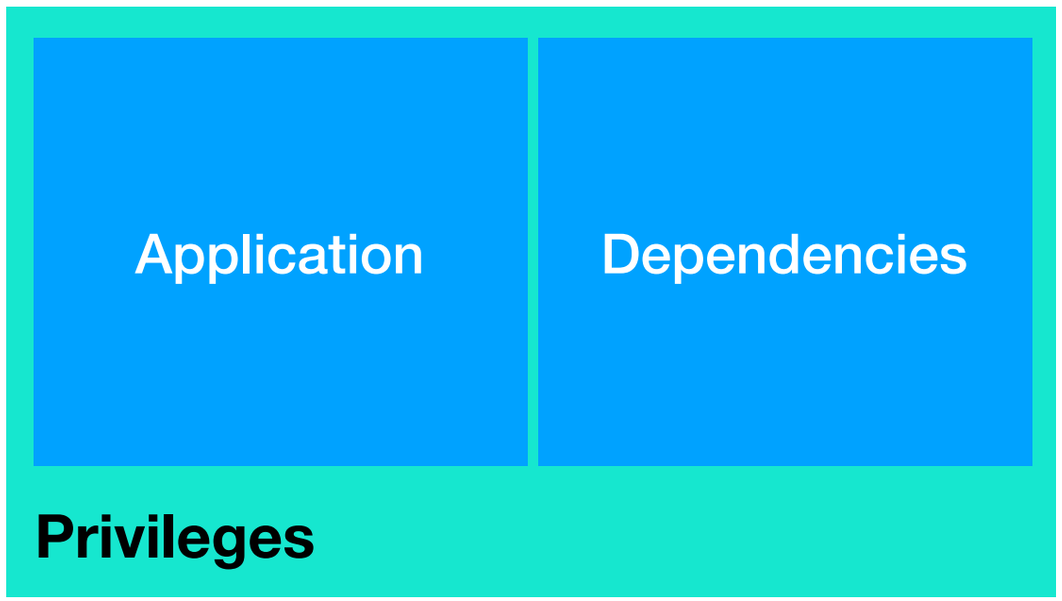
Ack: Graph generated using dataset from https://www.cve-search.org/dataset/

Software Security **Techniques**

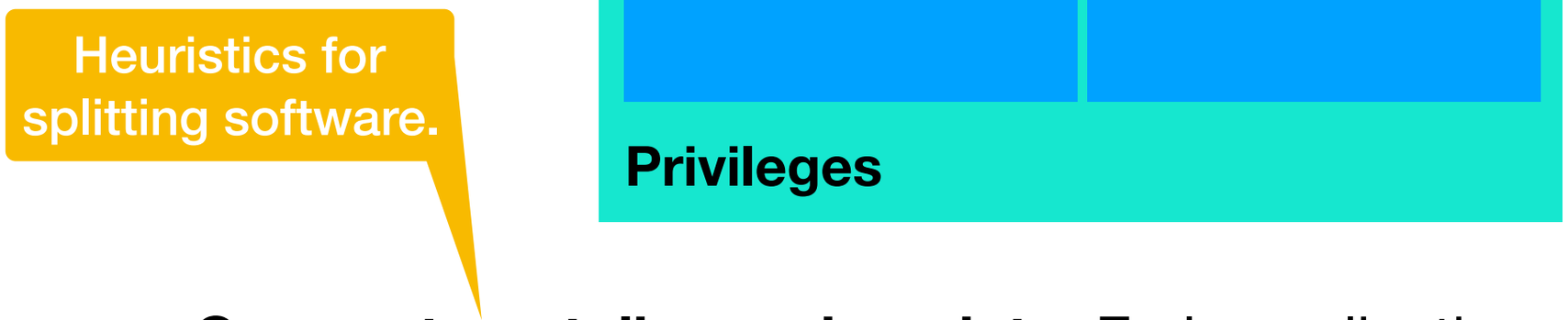
- Range of techniques available: ASLR, Stack canaries, Sandboxing, Soft/hard bounds checking, …
- Combining them is good practice.
  But some techniques are difficult to apply.

  We focus on one such technique: **privilege separation.**

What is Privilege Separation? (privsep)



What is Privilege Separation? (privsep)

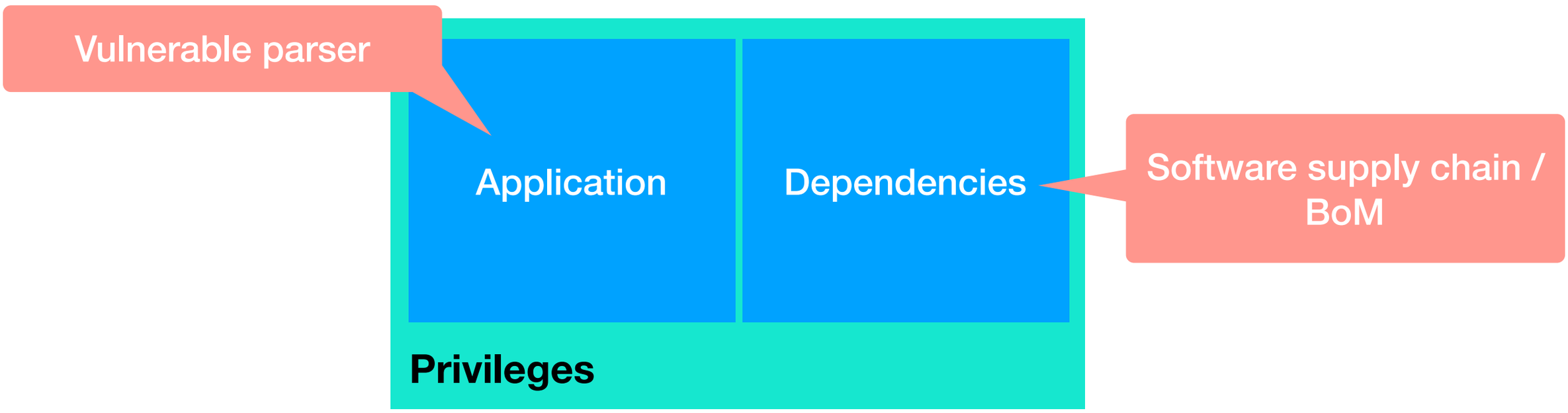


- **Compartmentalize code + data**. Early application: servers: SMTP, SSH.
- Monolithic application ➡ Concurrent set of cooperating programs.
  - Monolithic application: often common privileges throughout.
  - **Distributed system**: granularity of privilege allocation.

Why

~~What is~~ Privilege Separation? (privsep)

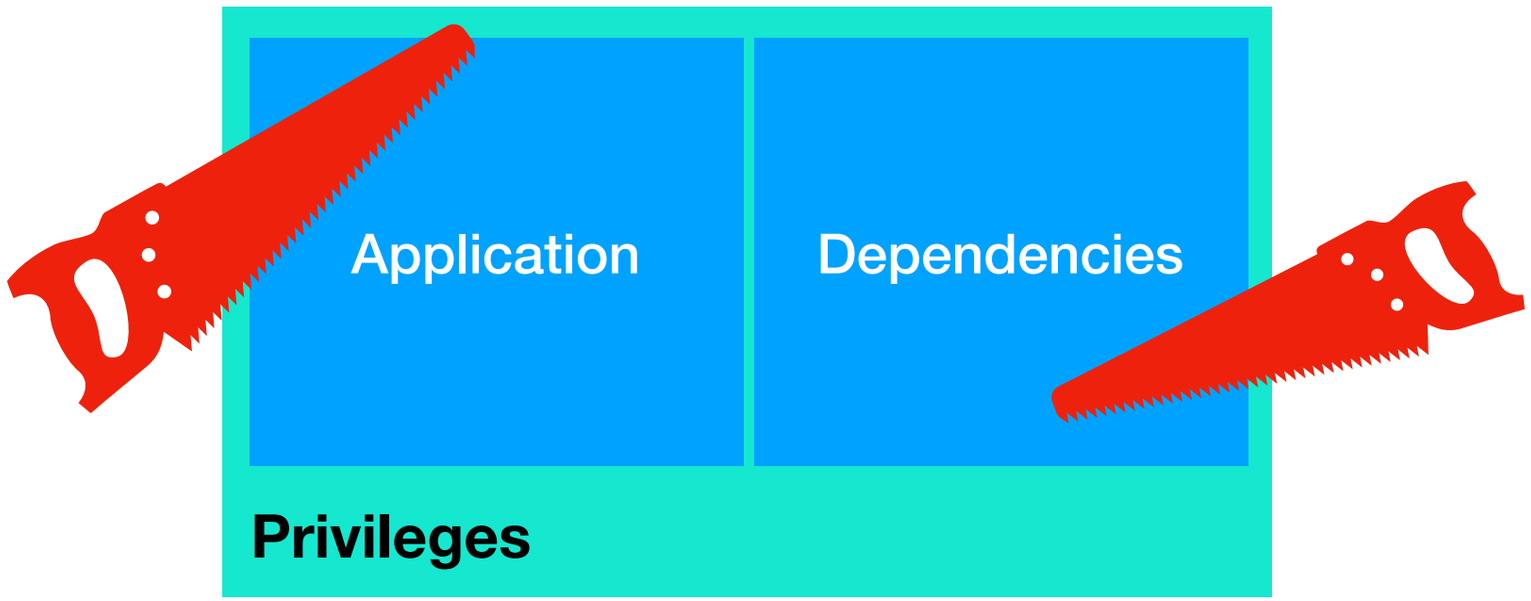
^



- **Compartmentalize code + data**. Early application: servers: SMTP, SSH.
- Monolithic application ➡ Concurrent set of cooperating programs.

Main benefit: **vulnerability containment.**
Best case: if a vulnerability is exploitable, then fewer privileges can be abused.

Implementing Privsep

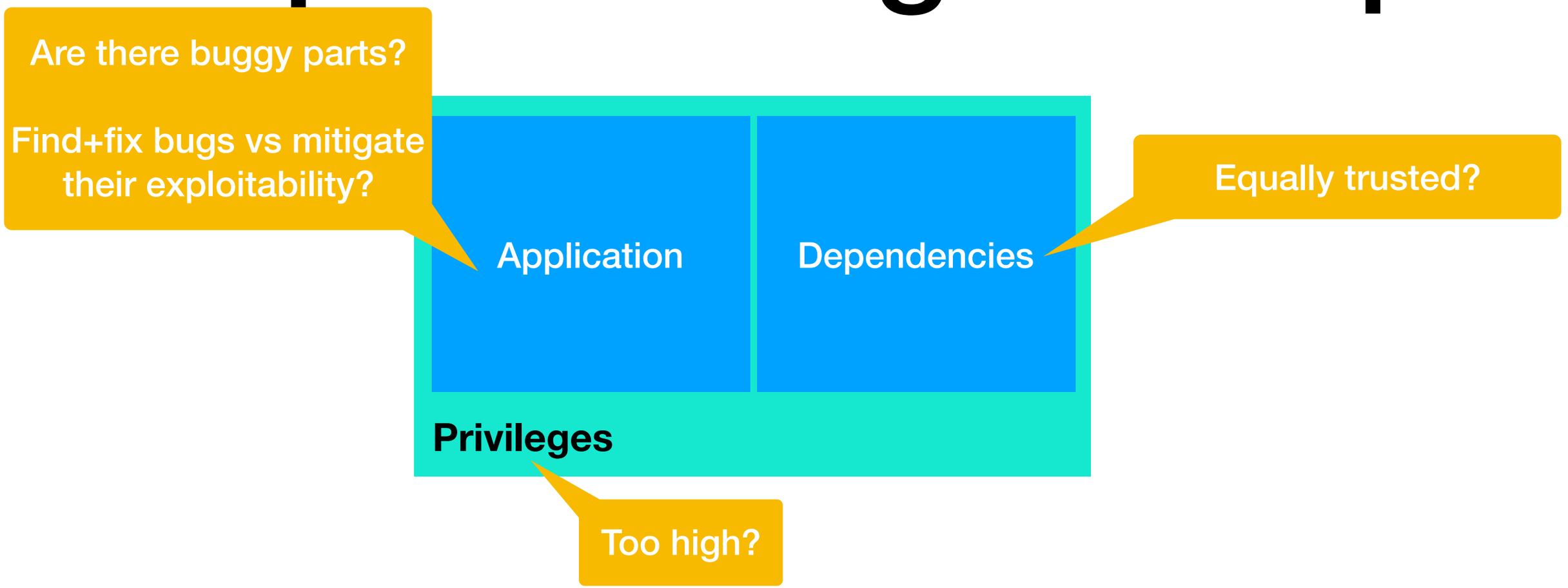


- **Implementing** privsep: usually a lot of work.
  Restructuring logic and code, positive and negative tests.

- Changing software without introducing bugs!

- There are many **decisions** to take (and retake later) wrt what+how to separate.

Implementing Privsep

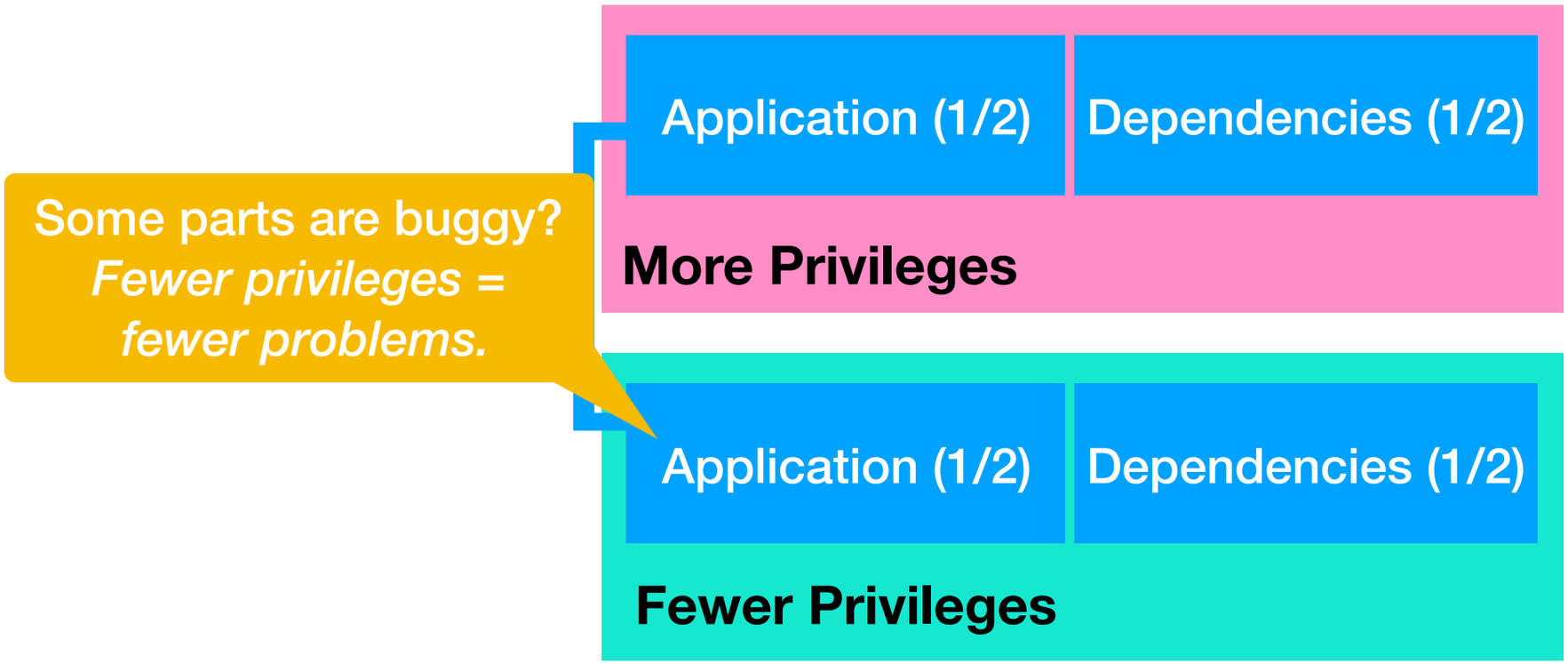


- **Implementing** privsep: usually a lot of work.
  Restructuring logic and code, positive and negative tests.

- Changing software without introducing bugs!

- There are many **decisions** to take (and retake later) wrt what+how to separate. (See yellow bubbles above)

What Privsep looks like

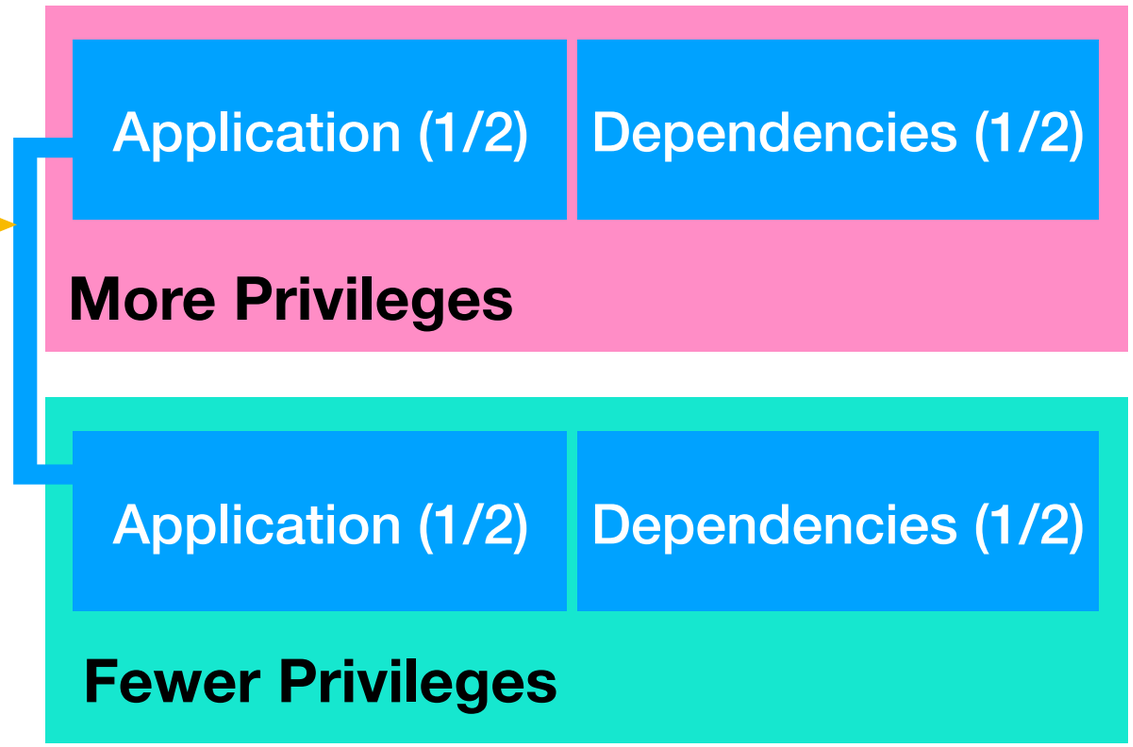


- Distributed system, heterogeneous privileges.

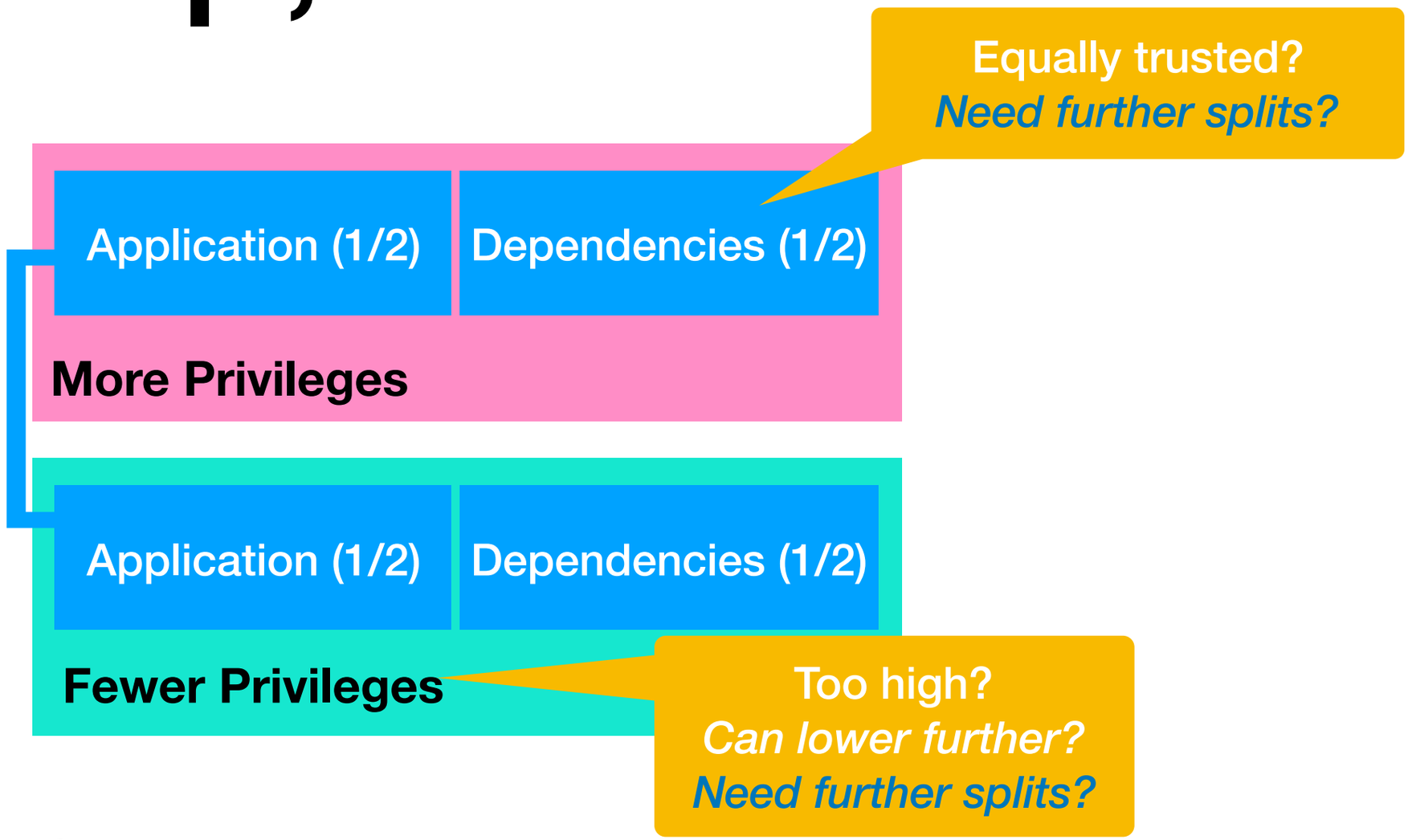
  Sometimes: separating between trusted vs untrusted.

What Privsep looks like

Heuristics:
- Components needing specific access.
- Dependencies incl. libraries.
- Cross-domain interfaces (e.g., parts of network, filesystem)



Privsep, *and then?*



- **Drawbacks** include:
  Inertia wrt **splitting software**, introduction of **new failure modes** (hello distributed systems), performance **overhead**, inertia wrt **maintainability and portability** (e.g., if use hardware enforcement).

(Longstanding) # Research Goal

Widely-applicable tool support for privsep

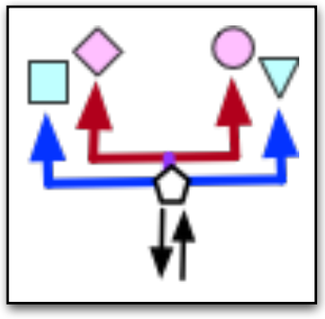
(This paper)

Foundations:
- compartment model
- tool infrastructure
- software-level

(Longstanding) # Research Goal

Widely-applicable tool support for privsep

(This paper)



Artefacts:
+ tooling
+ several examples
+ supporting scripts & documentation

Foundations:
- compartment model
- tool infrastructure
- software-level

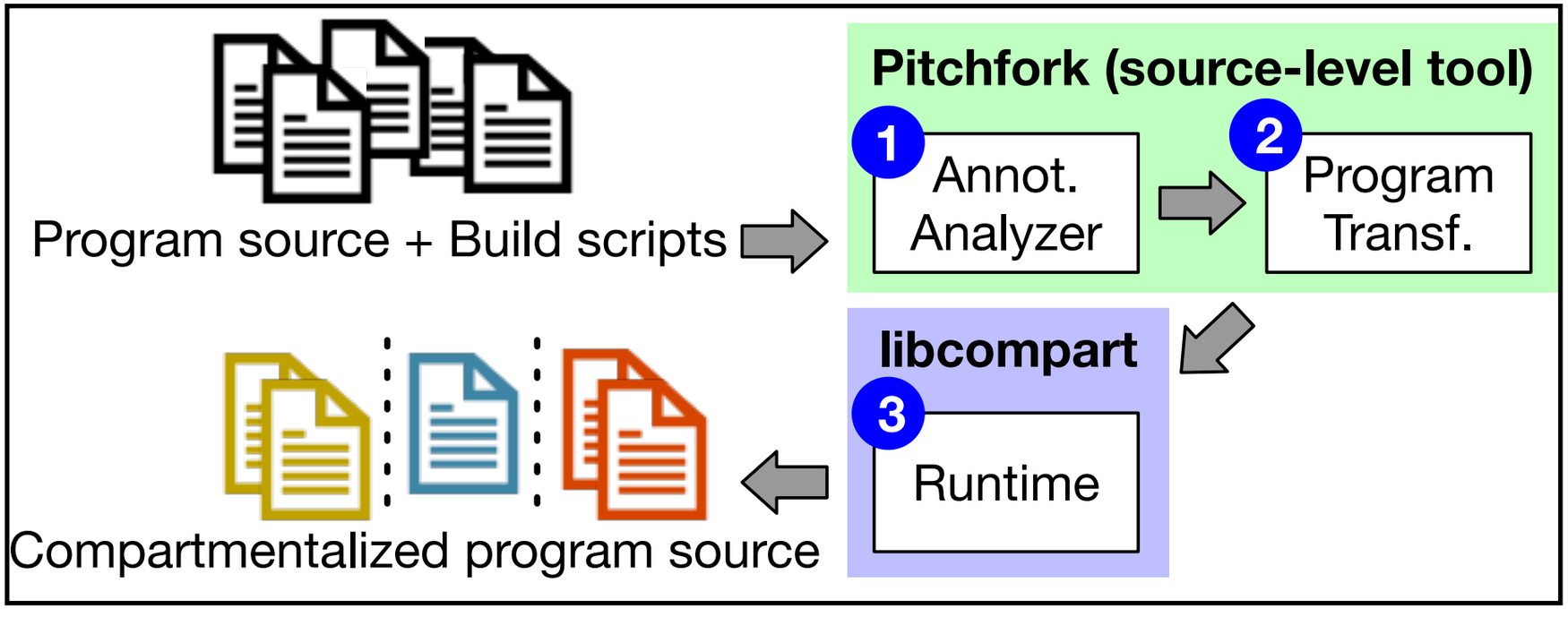
What’s different from prior art?

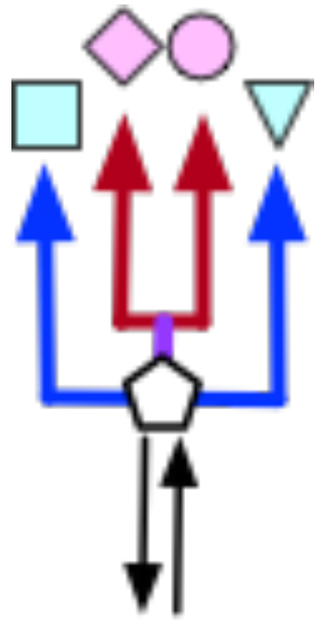
- **Separation “distance” + flexibility.**
  Separate binaries vs separate processes.
  Number of compartments.
  Commodity kernels and hardware.

- **Both tool and library.**
  Either can be used directly.
  Tool adapts code to use library.

- **Model-based approach.**
  Implemented abstractions provided/explained by the model.

Pitchfork

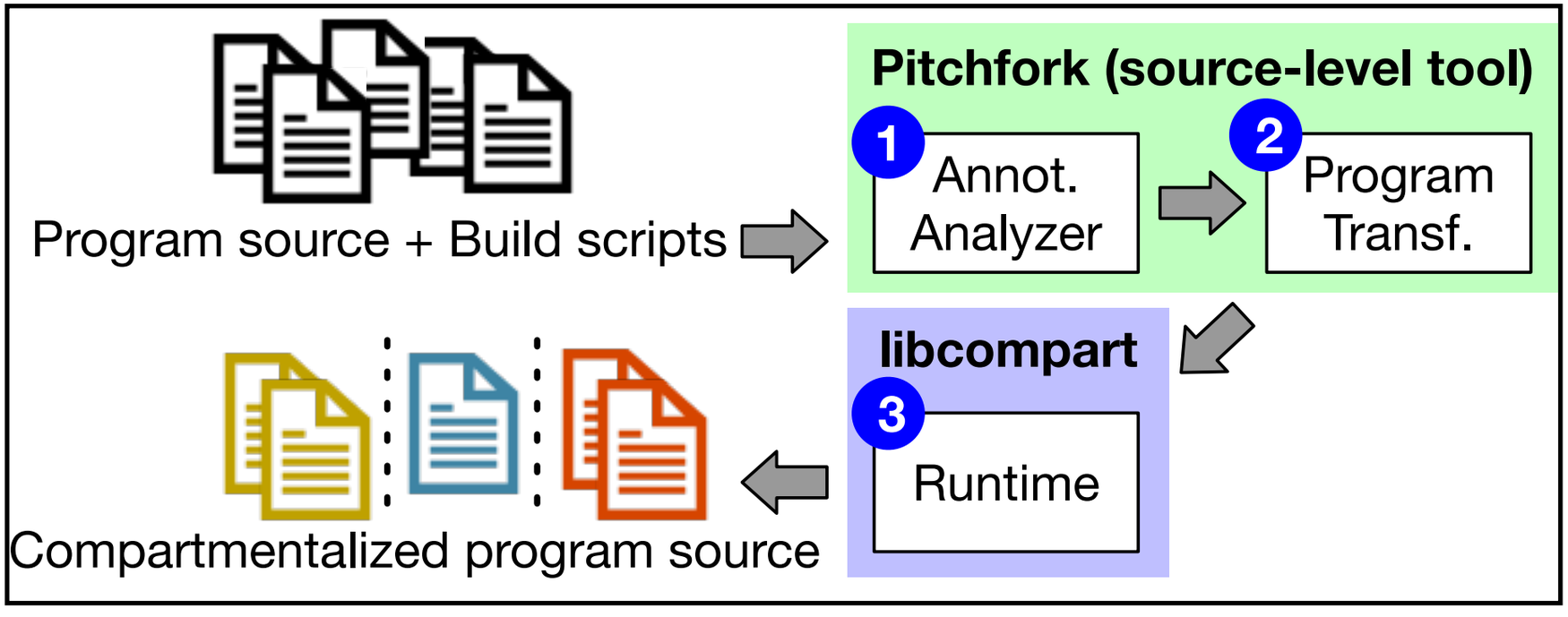


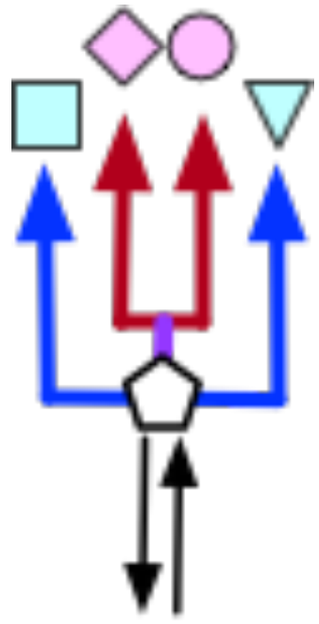


The **system** has two components based on a **model**:

- Pitchfork 1 2
- libcompart 3

Pitchfork





The **system** has two components based on a **model**:

- Pitchfork 1 2
- libcompart 3

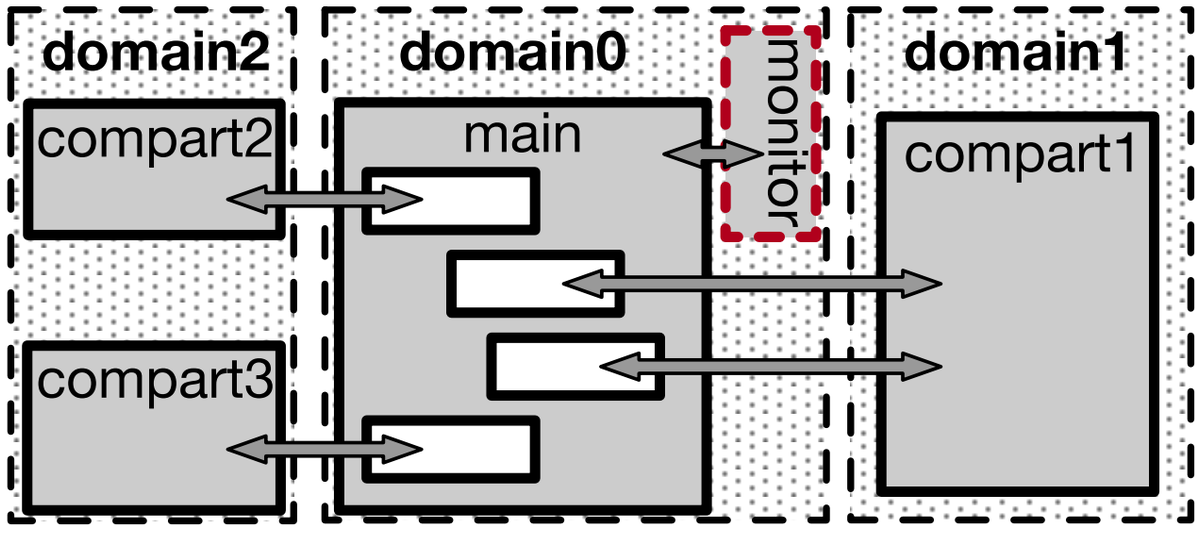
The **model** supports**:**

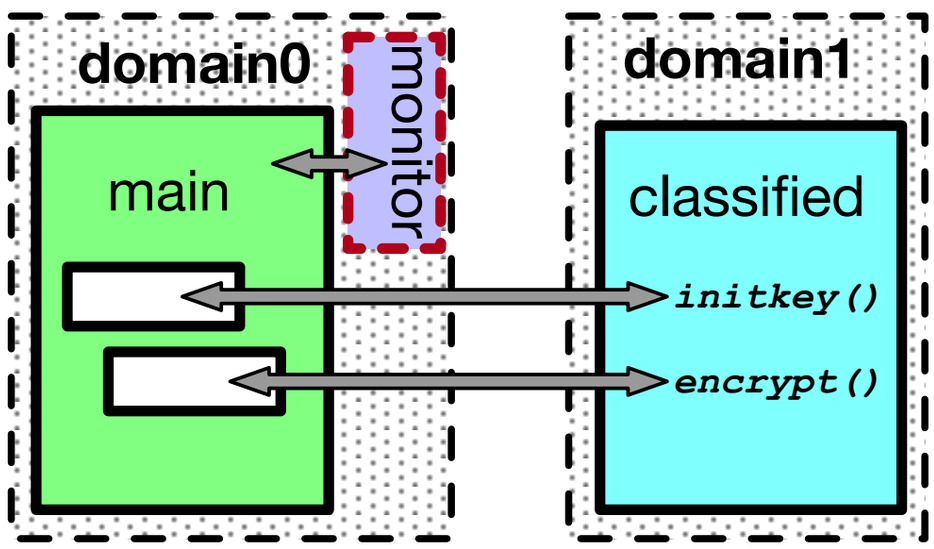
- Multiple compartments (different levels of trust)
- Synchronous communication
- Monitoring and failure-handling

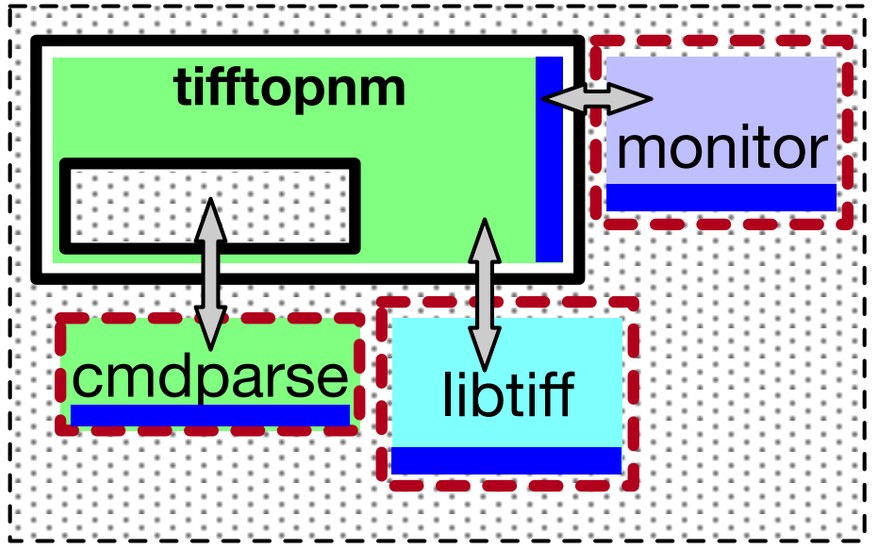
Pitchfork

```
105 if(console_type == BEEP_TYPE_CONSOLE) {
106   pitchfork_start("Privileged");
107   if(ioctl(console_fd, KIOCSOUND, period) < 0) {
108     putchar('\a'); /* Output the only beep we can, in an
                effort to fall back on usefulness */
109     perror("ioctl");
110   }
111   pitchfork_end("Privileged");
112 } else {
113    /* BEEP_TYPE_EVDEV */
114   struct input_event e;
115   e.type = EV_SND;
116   e.code = SND_TONE;
117   e.value = freq;
118   pitchfork_start("Privileged");
119   if(write(console_fd, &e, sizeof(struct input_event)) <
              0) {
120     putchar('\a'); /* See above */
121     perror("write");
122   }
123   pitchfork_end("Privileged");
124 }
```

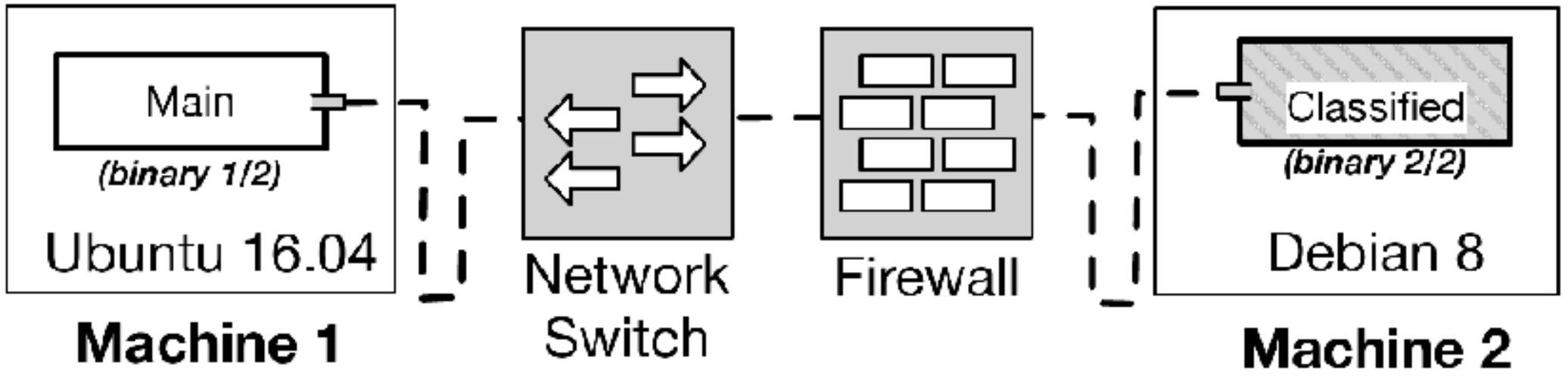
Compartment Model







Example of what's enabled



- Machine and network-level policy+enforcement.

- Communication channel over TCP.

- Organization:
  Domain: one on each machine
  Compartments: one in each domain.
  Segments: 2 in Classified, 1 in Main.

Evaluation

(Many more details in the paper)

- Applicability
  - Examples
  - Maintainability
  - Convenience
- Security
  - Known CVEs
  - Heuristics
- Overhead: running time, memory, binary size.

Evaluation

- Applicability
  - Examples
  - Maintainability
    - Convenience
- Security
  - Known CVEs
  - Heuristics
- Overhead: running time, memory, binary size.

Software	Plat.	Separation Goal
cURL	L	Command invocation, parsing, file transfer.
Evince	L	libspectre dependency—see §2.
gil	L	Historical vulnerability [13].
ioquake3	m	Applying server updates.
tifftopnm	L	Separating parsers—see §C.
nginx	L	HTTP request parsing
redis	L	Isolating low-use commands.
tcpdump, uniq	F	Leveraging Capsicum [68].
Viletris	L	Network-facing code—see §2.

Evaluation

- Applicability
  - Examples
  - Maintainability
  - Convenience
- Security
  - Known CVEs
  - Heuristics
- Overhead: running time, memory, binary size.

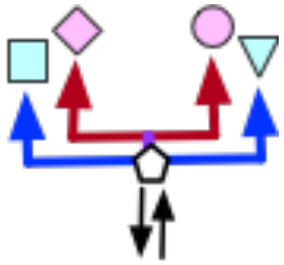
$$\text{SAR} = \frac{\#\text{LOC Synthesized}}{\#\text{Lines of Annotation}}$$

Soft.	#LOC	#Annot.	#LOC Synthesized		SAR
			Compart.	De/marsh.	
beep	372	9	133	245	42
PuTTY	123K	6	52	29	13.5
wget[6]	62.6K	3	65	168	77.7
wget[7]	62.8K	8	57	38	11.9

Evaluation

- Applicability
  - Examples
  - Maintainability
  - Convenience
- Security
  - Known CVEs
  - Heuristics
- Overhead: running time, memory, binary size.

Software	CVE-*-*	Vulnerability
beep	2018-0492	Race condition
PuTTY	2016-2563	Stack buffer overflow
wget	2016-4971	Arbitrary file writing
wget	2017-13089	Stack buffer overflow



System release

- http://pitchfork.cs.iit.edu
- Everything is released except for exploit code:
  - libcompart
  - Pitchfork
  - examples of applying libcompart & Pitchfork
  - FreeBSD ports analysis
- Apache 2.0 license

ILLINOIS TECH

Computer Science Department

Nik Sultana

http://www.cs.iit.edu/~nsultana1