RAIN: Refinable Attack Investigation with On-demand Inter-process Information Flow Tracking

Y. Ji, S. Lee, E. Downing, et.al.

CCS'17

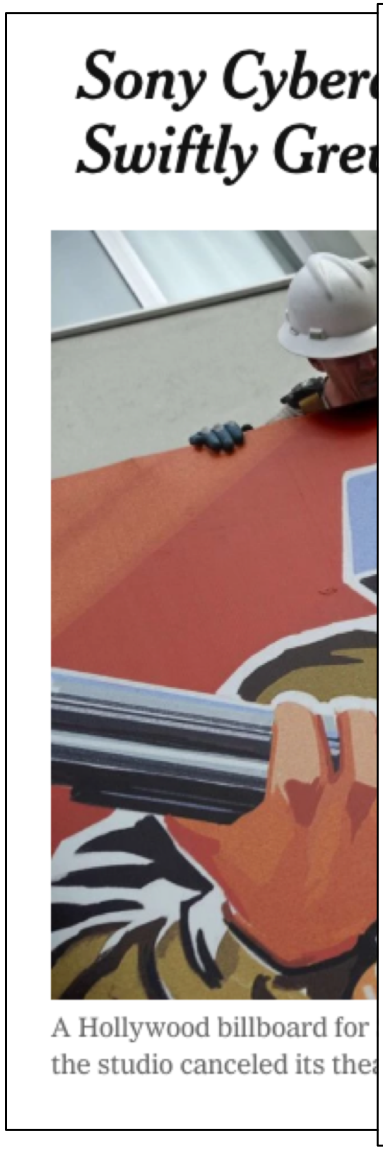
Presented by: Ruimin Sun

University of Florida

Nov 20, 2018

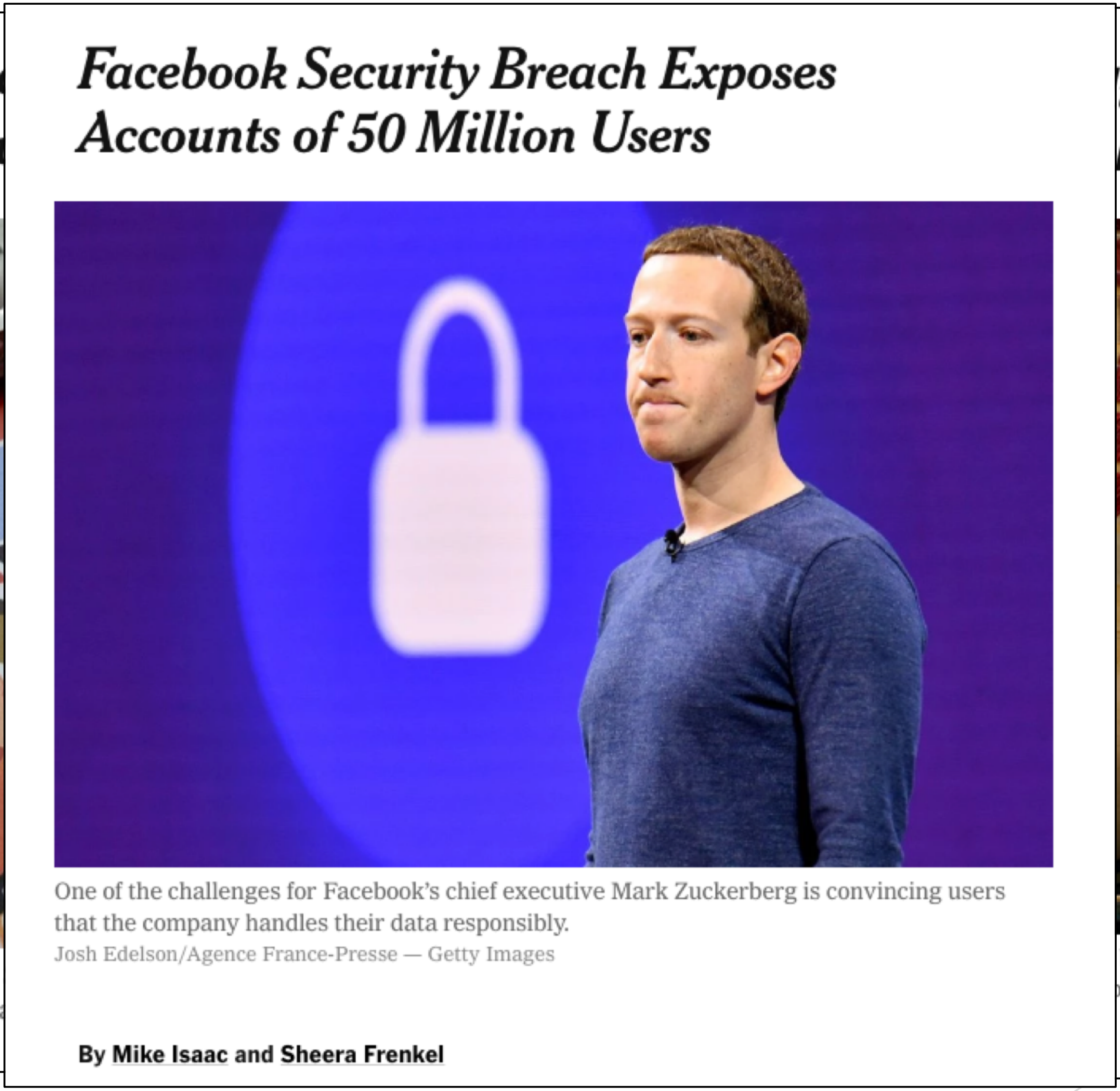
Special thanks to Yang Ji and Mohammad Noureddine for sharing slides.

Severe data breaches



Sony Cyber
Swiftly Gre

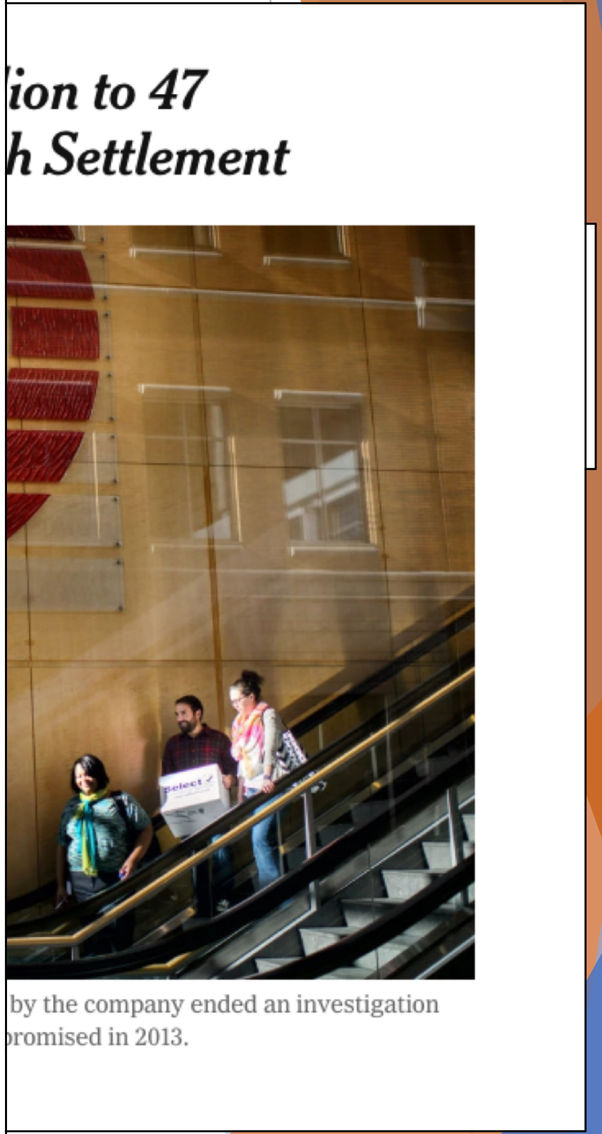
A Hollywood billboard for
the studio canceled its thea



Facebook Security Breach Exposes Accounts of 50 Million Users

One of the challenges for Facebook's chief executive Mark Zuckerberg is convincing users that the company handles their data responsibly.
Josh Edelson/Agence France-Presse — Getty Images

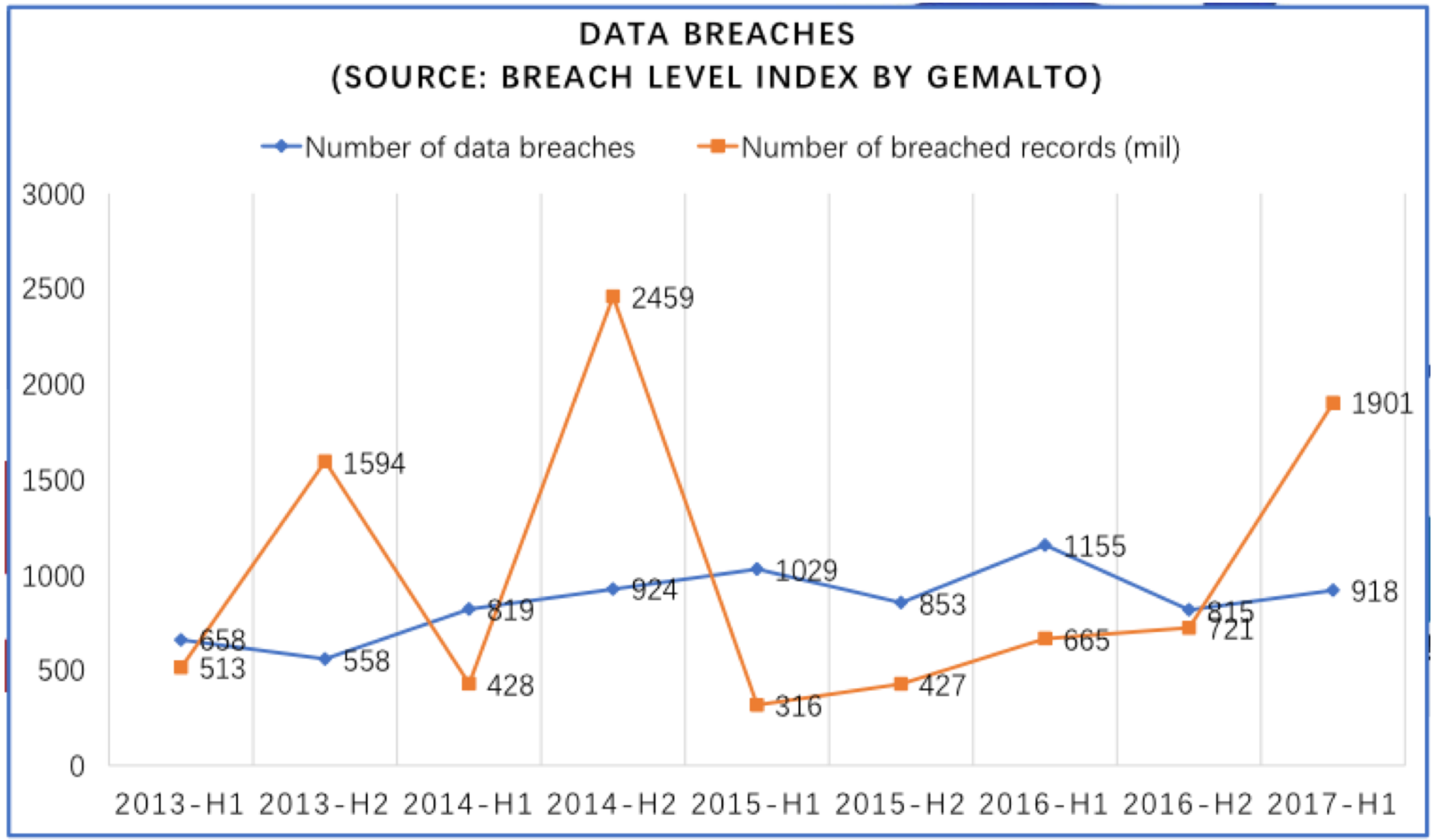
By Mike Isaac and Sheera Frenkel



ion to 47
h Settlement

by the company ended an investigation
promised in 2013.

Consistent data breaches



Solutions ?

- Determine the root cause of a breach?
- Determine the impacts of an exploit on the system?

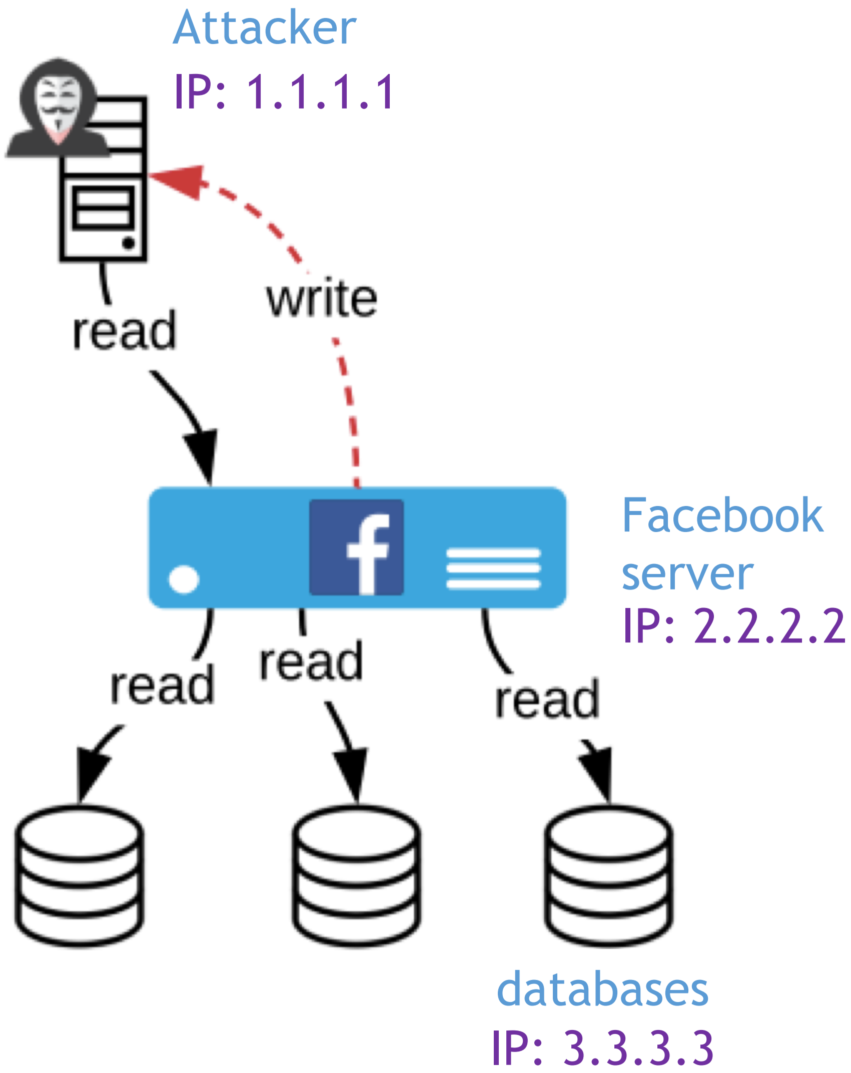
Provenance

“A ***complete description*** of agents (users, groups) controlling activities (processes) ***interacting*** with controlled data types during system execution”

DIFT (Dynamic Information Flow Tracking)

Provenance Examples

Network level



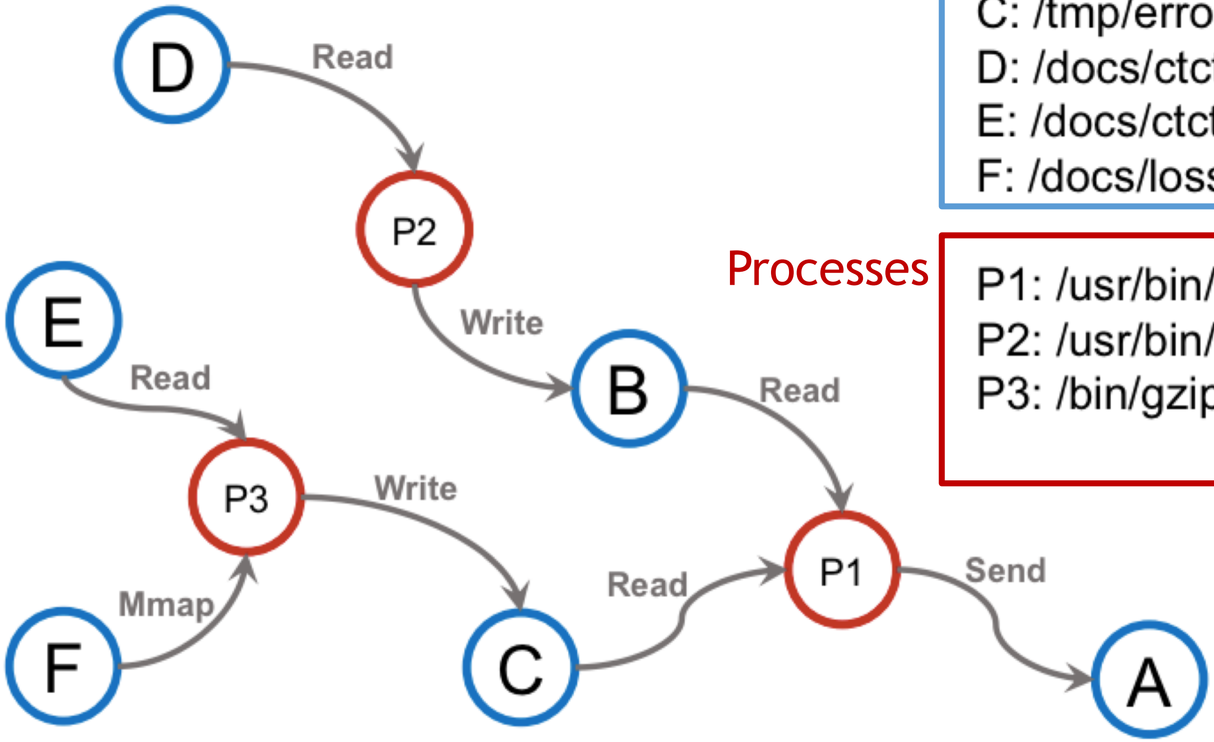
Attacker
IP: 1.1.1.1

Facebook server
IP: 2.2.2.2

databases
IP: 3.3.3.3

Provenance Examples

Operating system level



Files

A: Attacker site
B: /docs/report.doc
C: /tmp/errors.zip
D: /docs/ctct1.csv
E: /docs/ctct2.pdf
F: /docs/loss.csv

Processes

P1: /usr/bin/firefox
P2: /usr/bin/TextEditor
P3: /bin/gzip

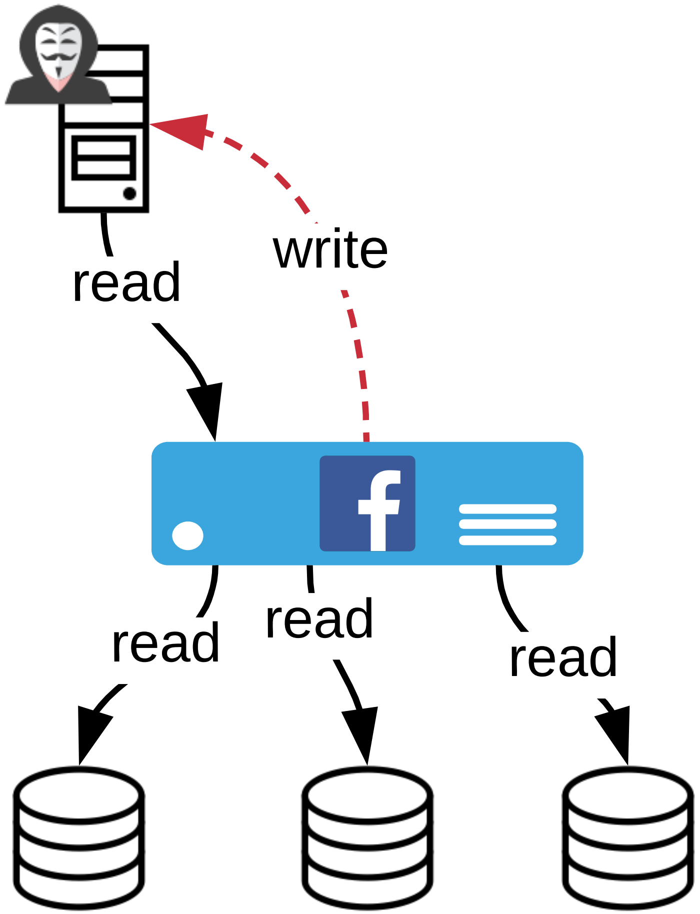
Provenance Graphs

(1) Track and log system information

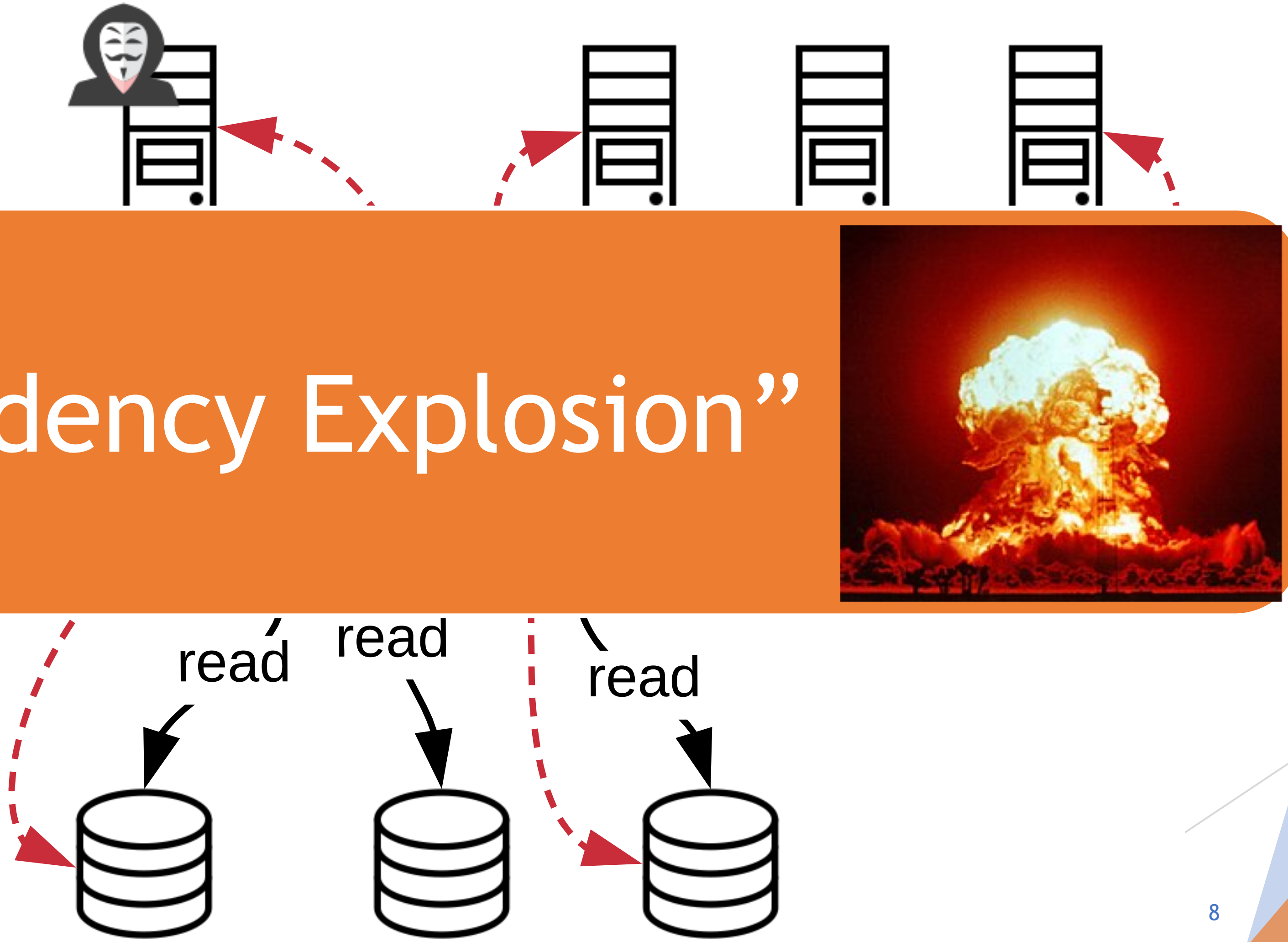
- Through *system calls*
  - e.g. read, write

(2) A given ***point of interest***

- Determine root cause
  - Backward traversal
- Determine impact on the system
  - Forward traversal



Provenance Graphs: Challenges

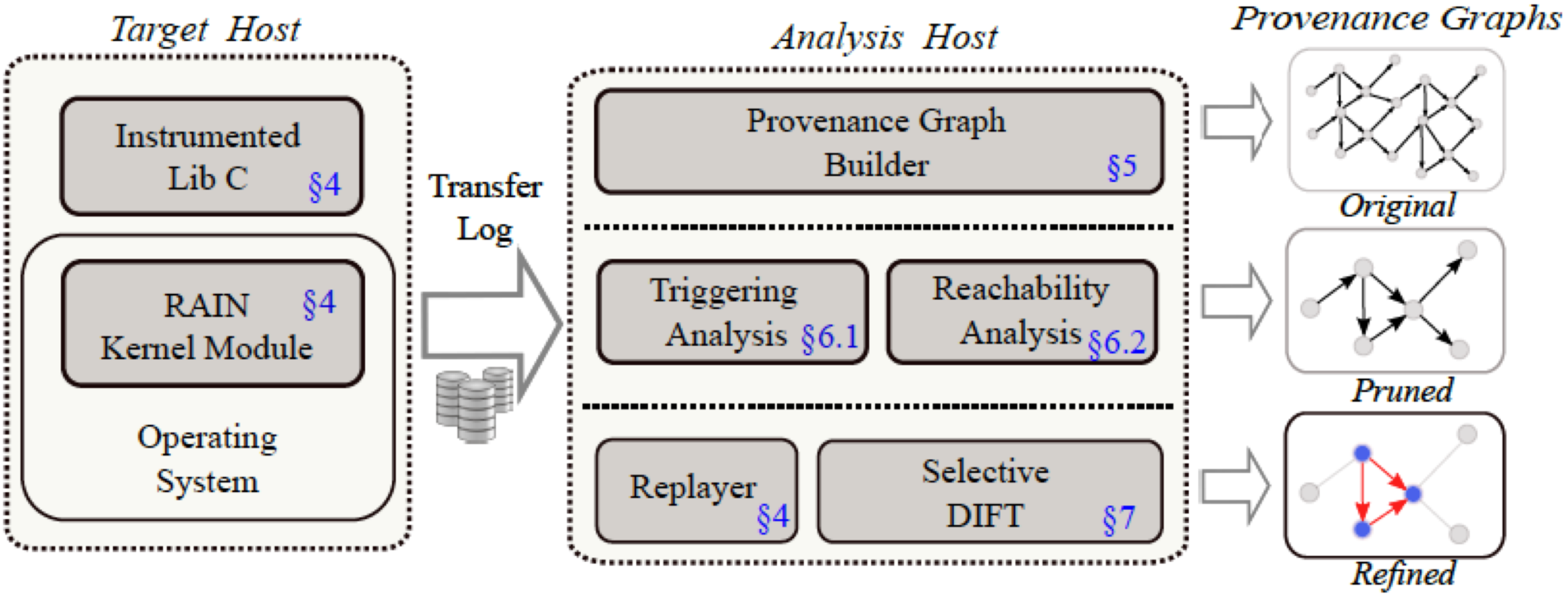


“Dependency Explosion”

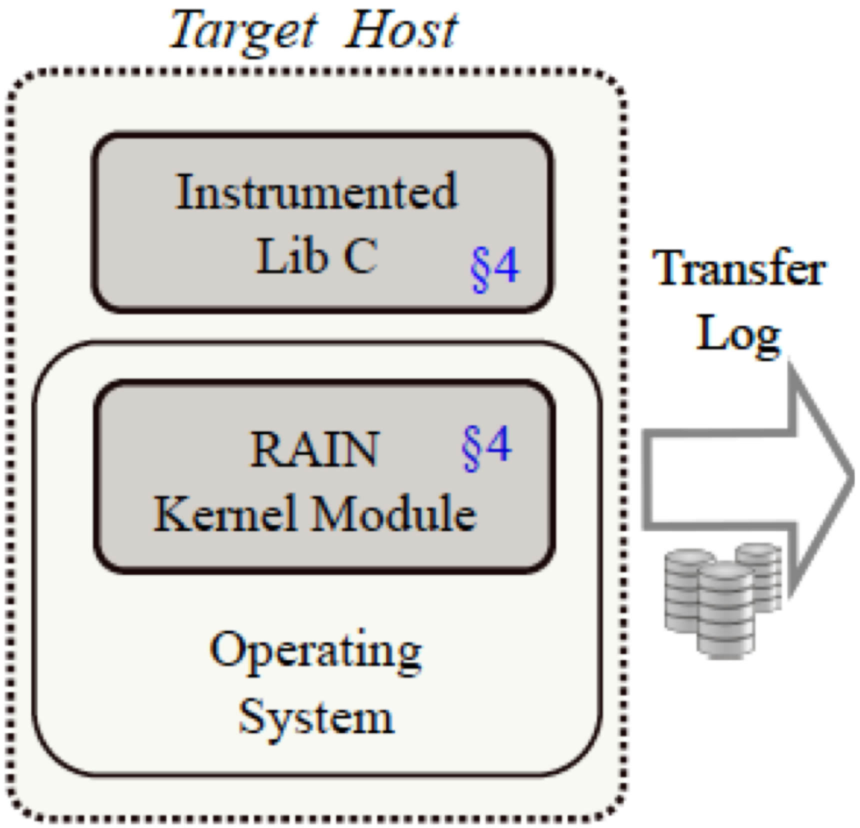
RAIN: Refinable Attack INvestigation

- Good runtime performance
- Reduce performance hit
- Improve granularity

High Level Overview

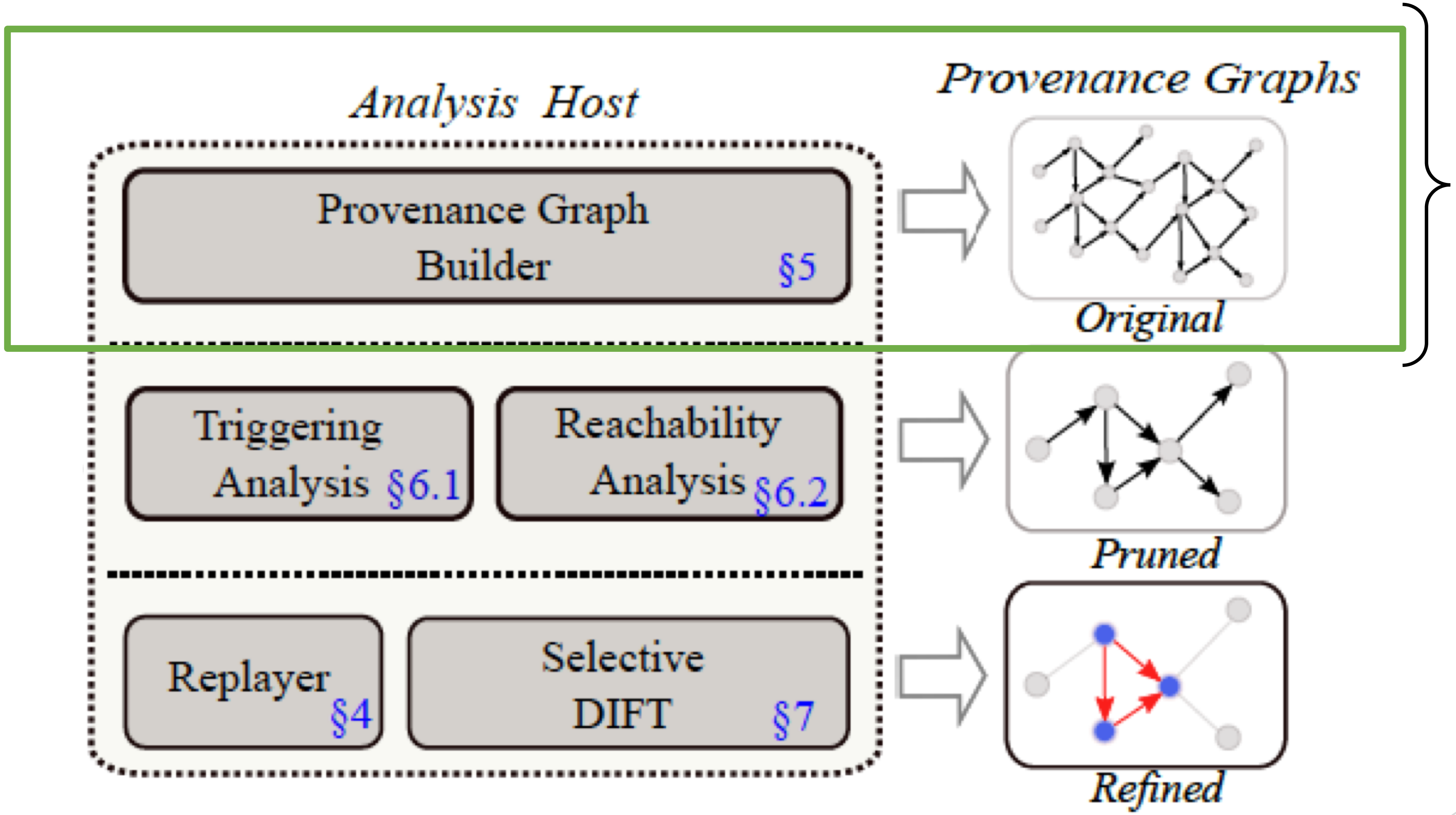


Log File Generation



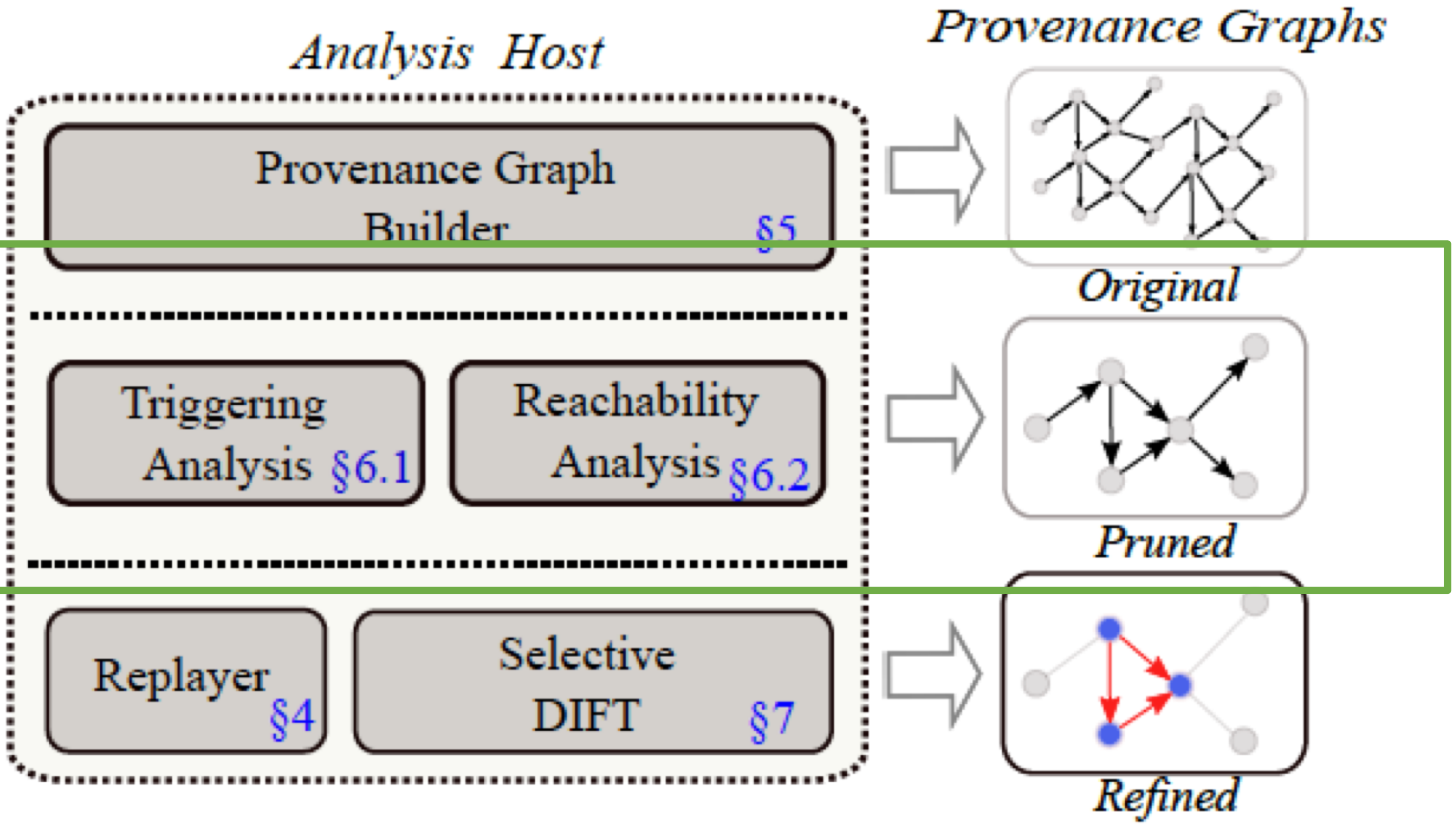
- Capture system calls,
  - read, write, open, send, recv, connect
  - their arguments, and return values
- Record IPC communications
- Cached file and network I/O
- Thread information
  - pthread in libc

Graph Builder



Still too expensive for analysis

Graph Pruning



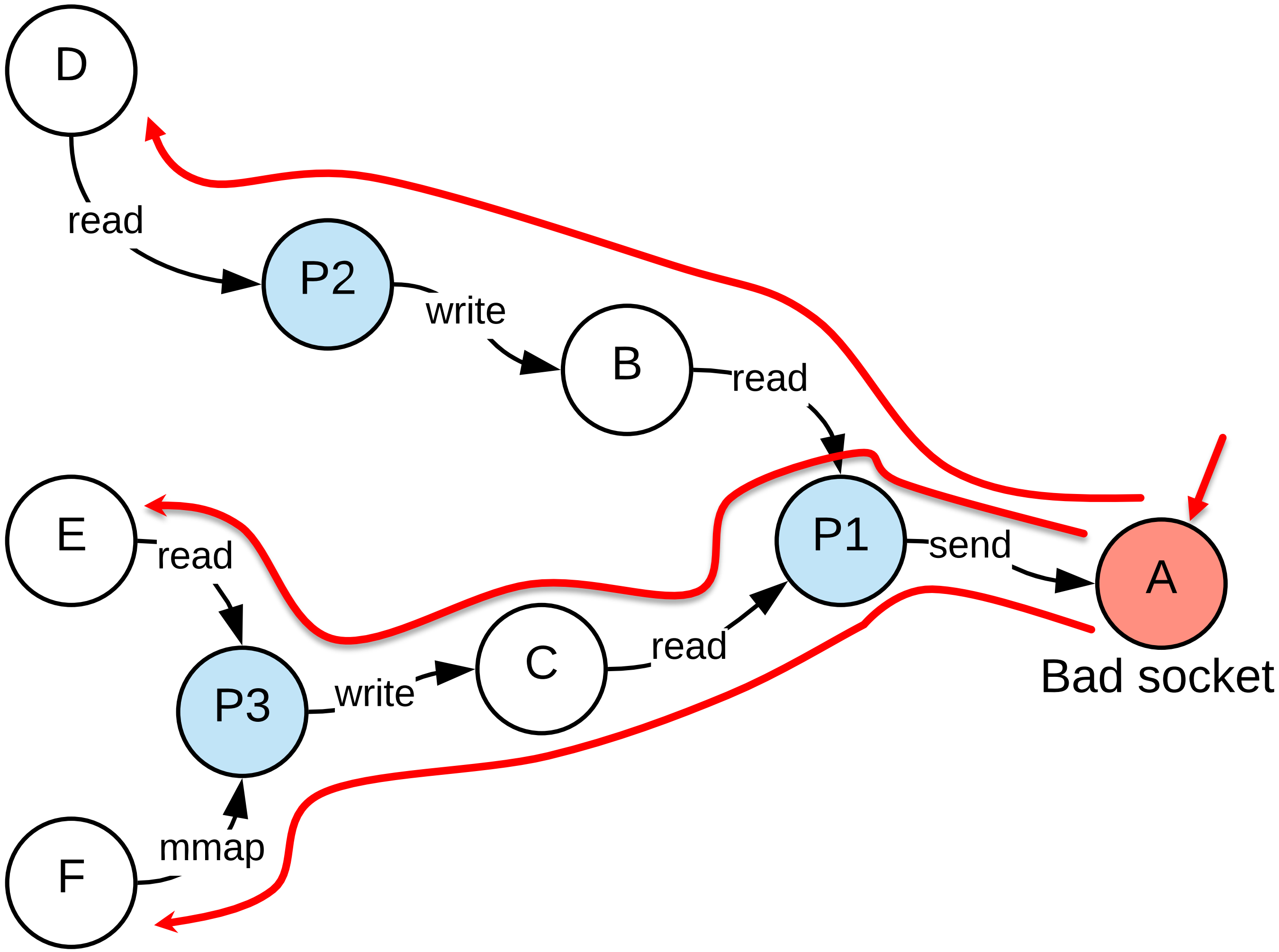
Pruning I: Triggering points

- limit the size of the graph to the most interesting nodes
- Three ***criterion*** for starting the analysis
  - *External signals*: tips from other sources, CVEs, responsible disclosures, etc.
  - *Security policy*: violations to a certain policy are interesting points for looking into
  - *Customized comparisons*: compare hashes of downloaded files

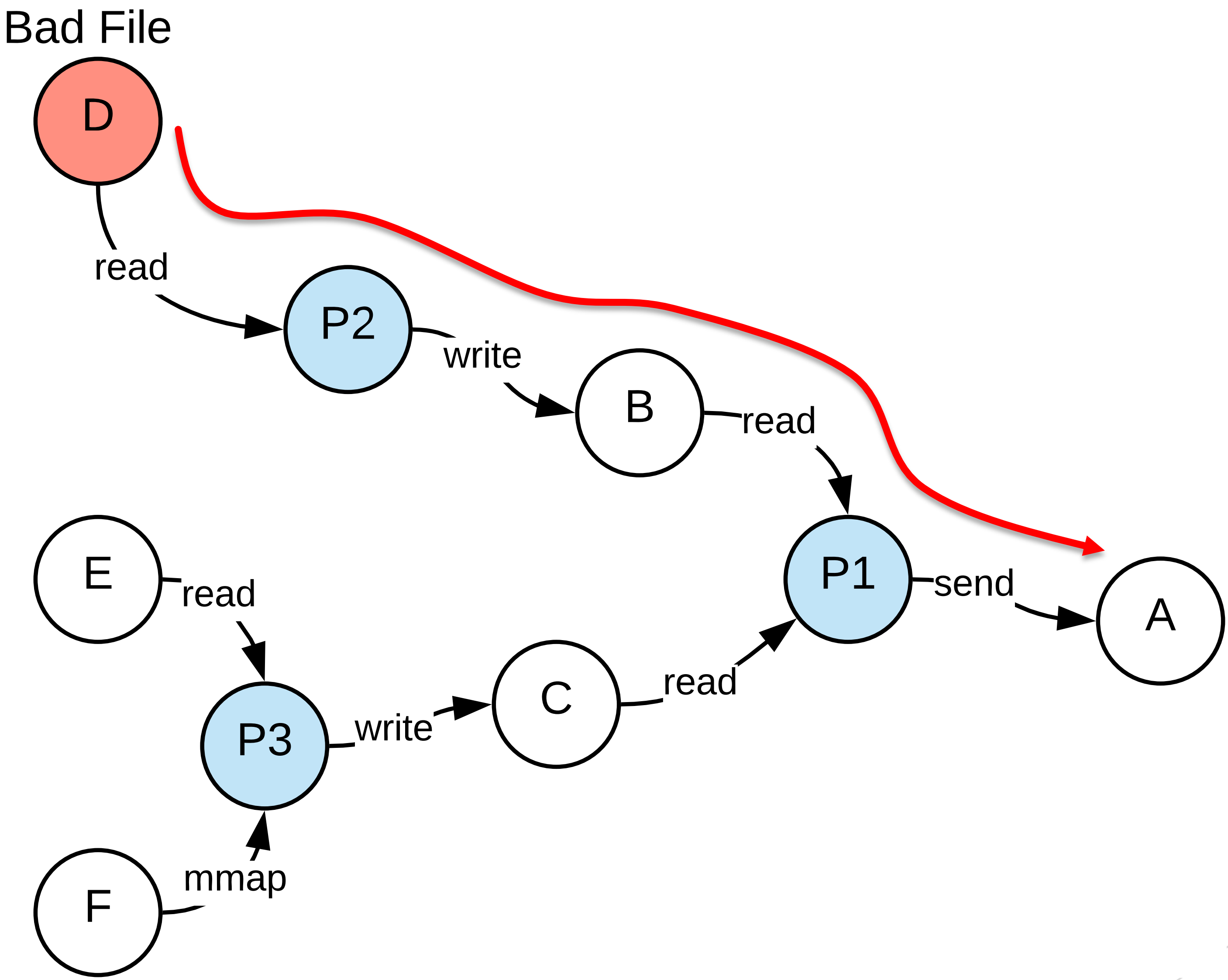
Pruning II: Reachability Analysis

- Starting from trigger points (points of interest)
  - Determine the next set of interesting points
- Forward reachability
- Backward reachability
- Point-to-point: Forward & Backward
- Heuristic *interference* analysis

Backward Reachability Analysis

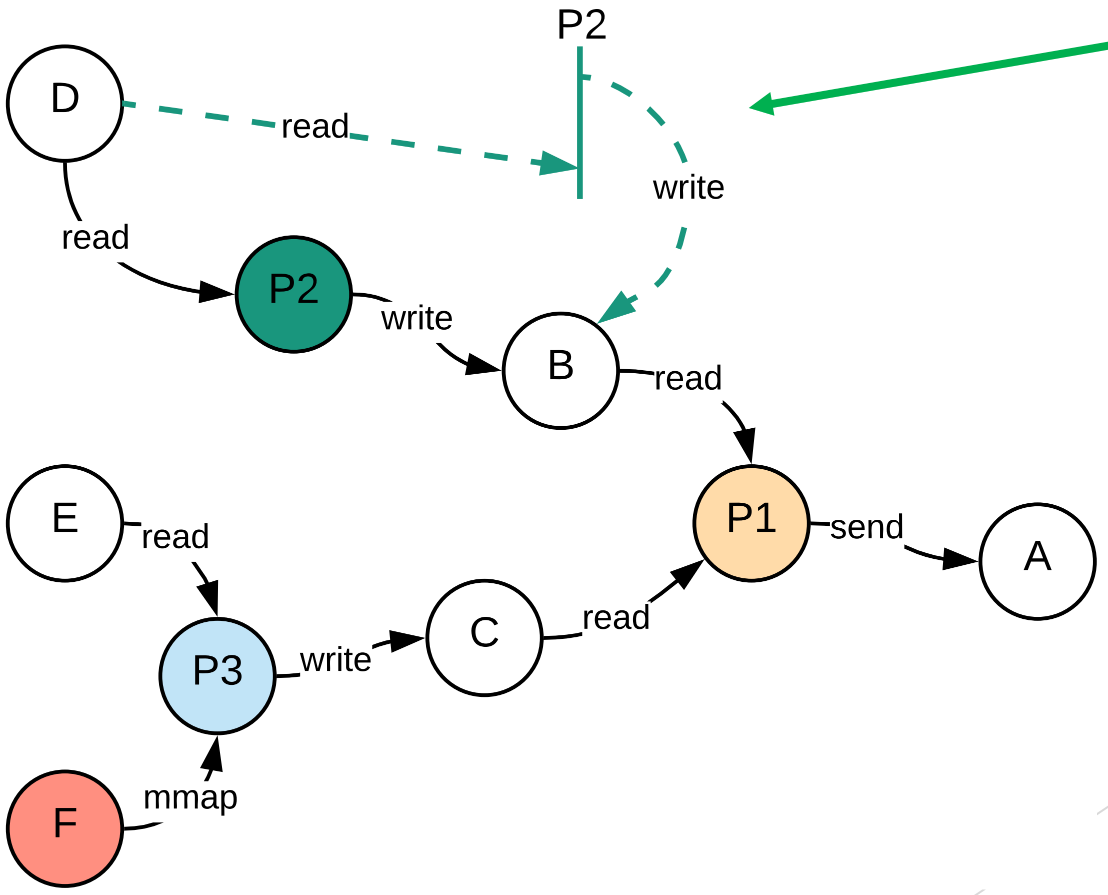


Forward Reachability Analysis



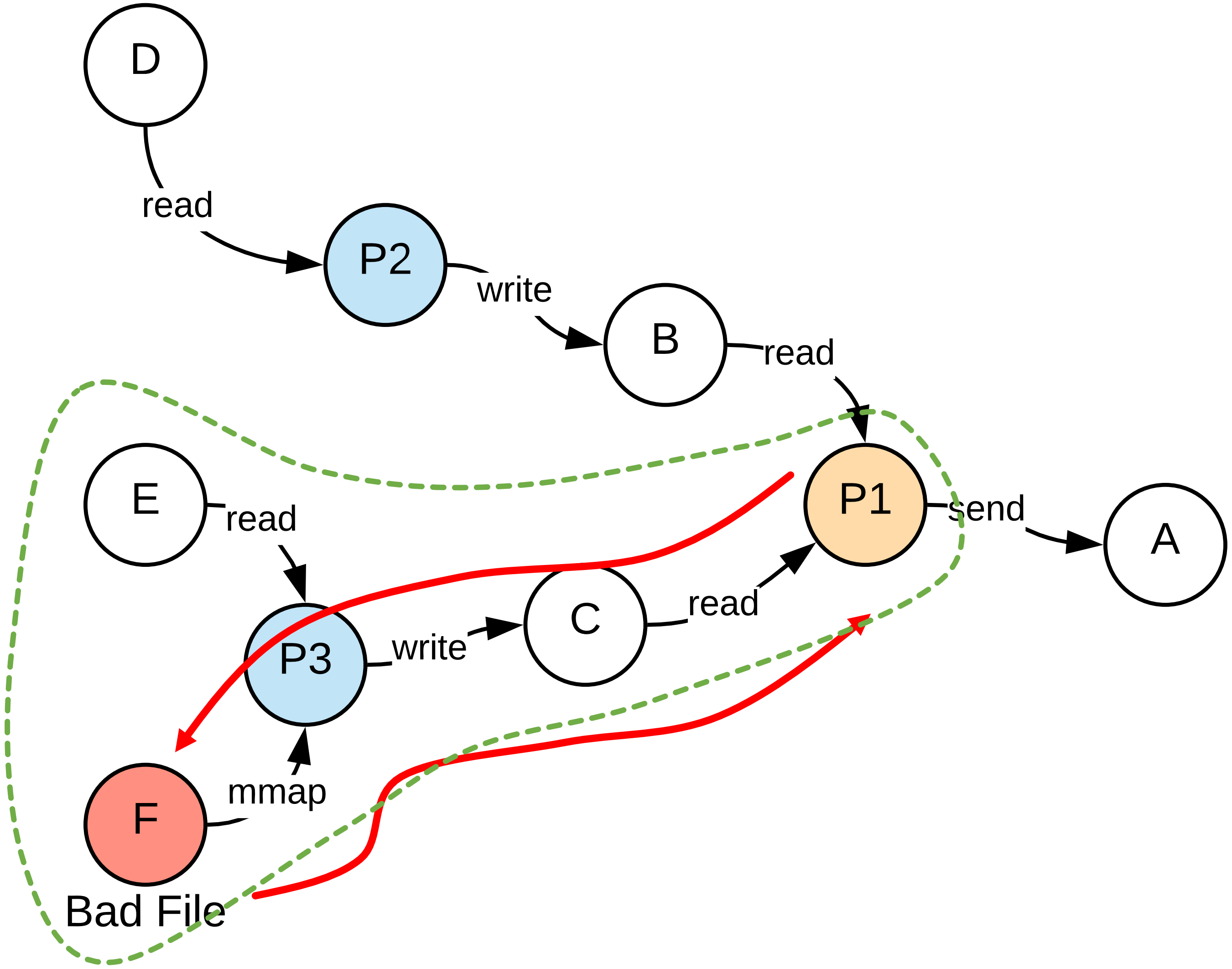
Interference Pruning

- Track *read-after-writes* using syscall timestamps
    - Remove false dependencies

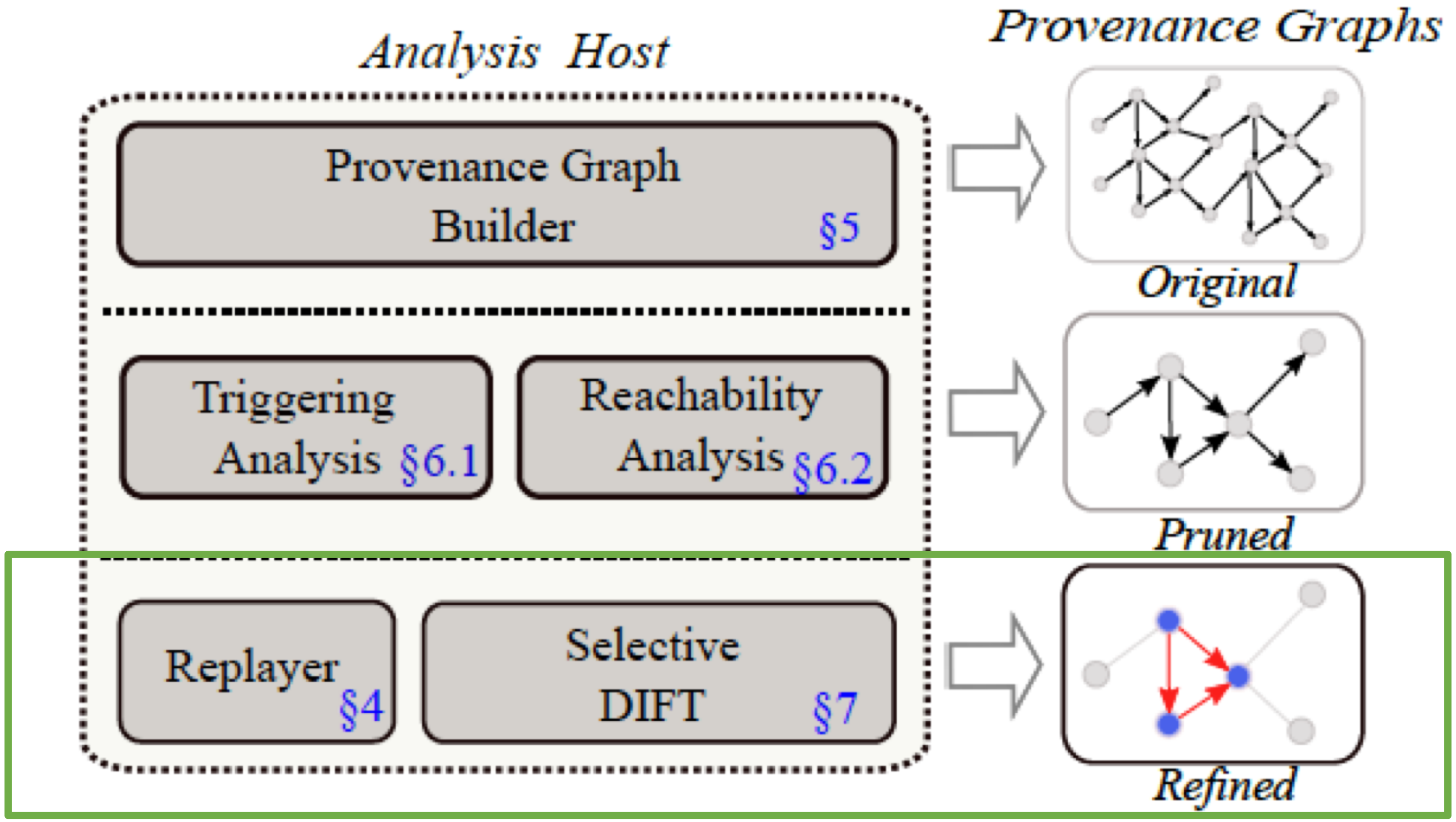


No memory interference

P2P Reachability



Graph Refining

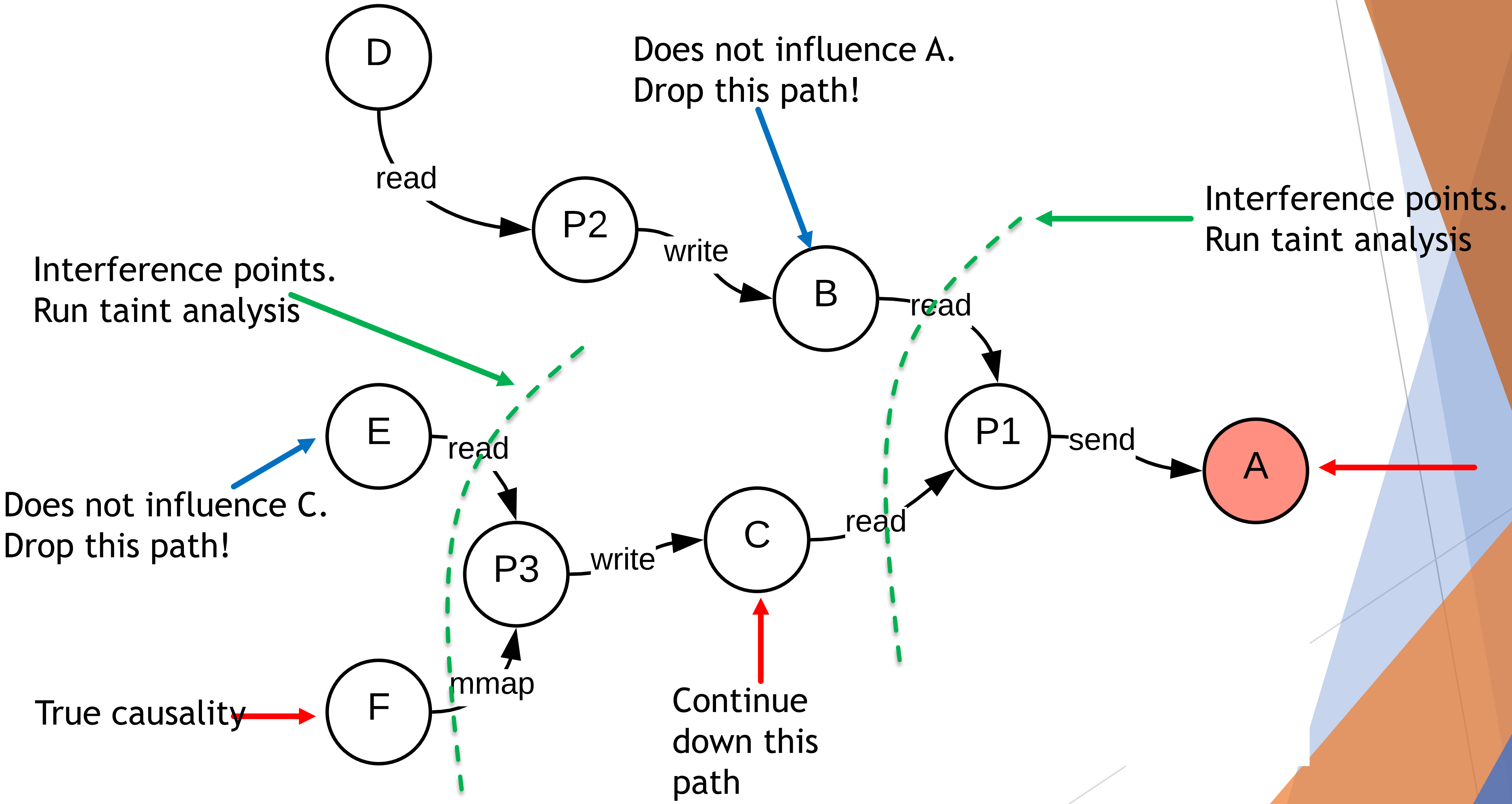


Selective DIFT

- Use the outcomes of the reachability analysis and trigger points
  - Start from interference points
- Refinement for
  - downstream causality,
  - upstream causality,
  - and point to point causality
- Run taint analysis for different processes independently
  - Cache results for improved performance

Q: Any apparent issues here?

DIFT: Upstream Refinement



Does not influence A.
Drop this path!

Interference points.
Run taint analysis

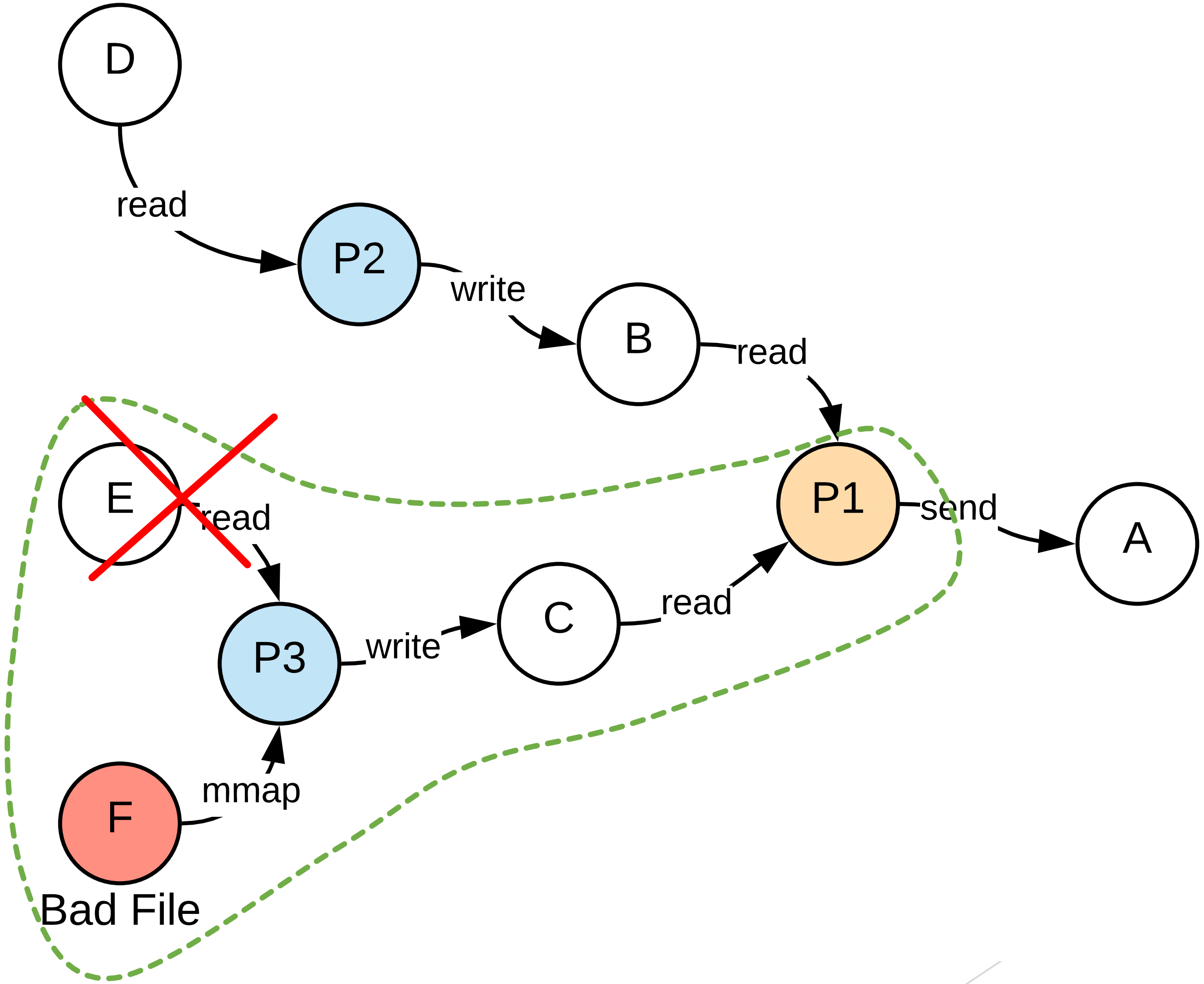
Interference points.
Run taint analysis

Does not influence C.
Drop this path!

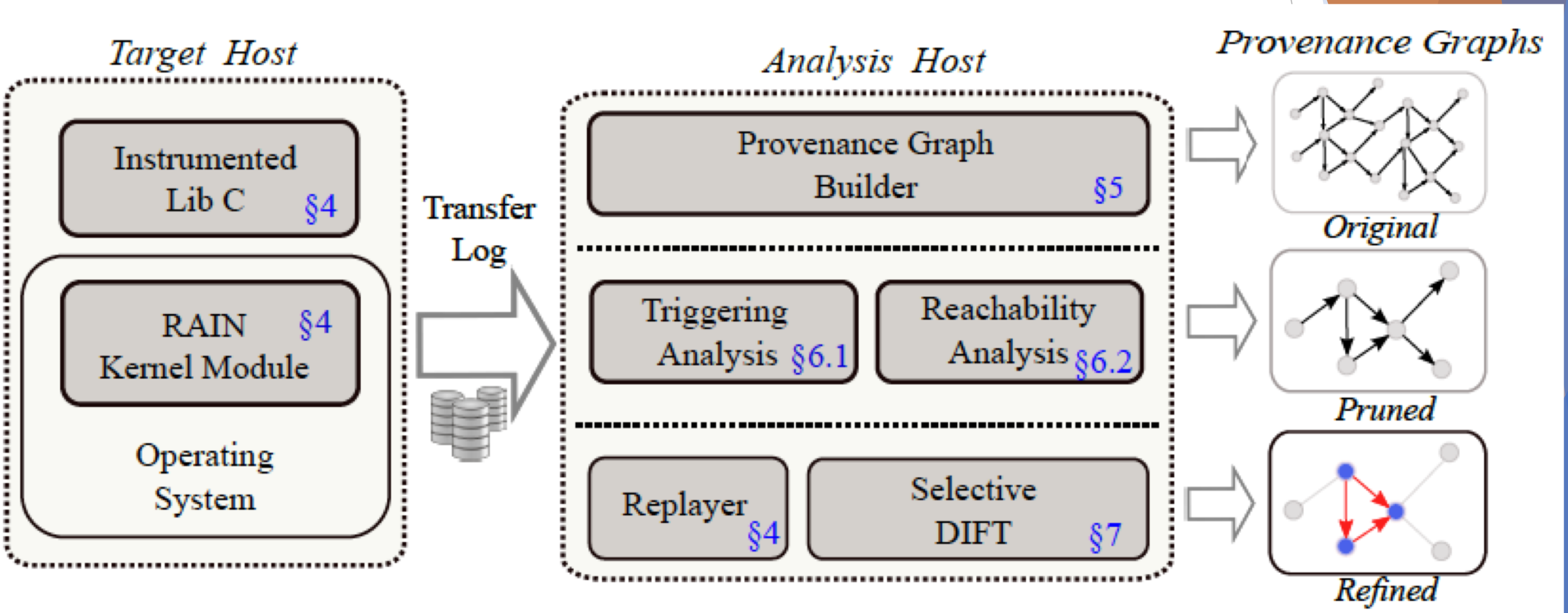
True causality

Continue
down this
path

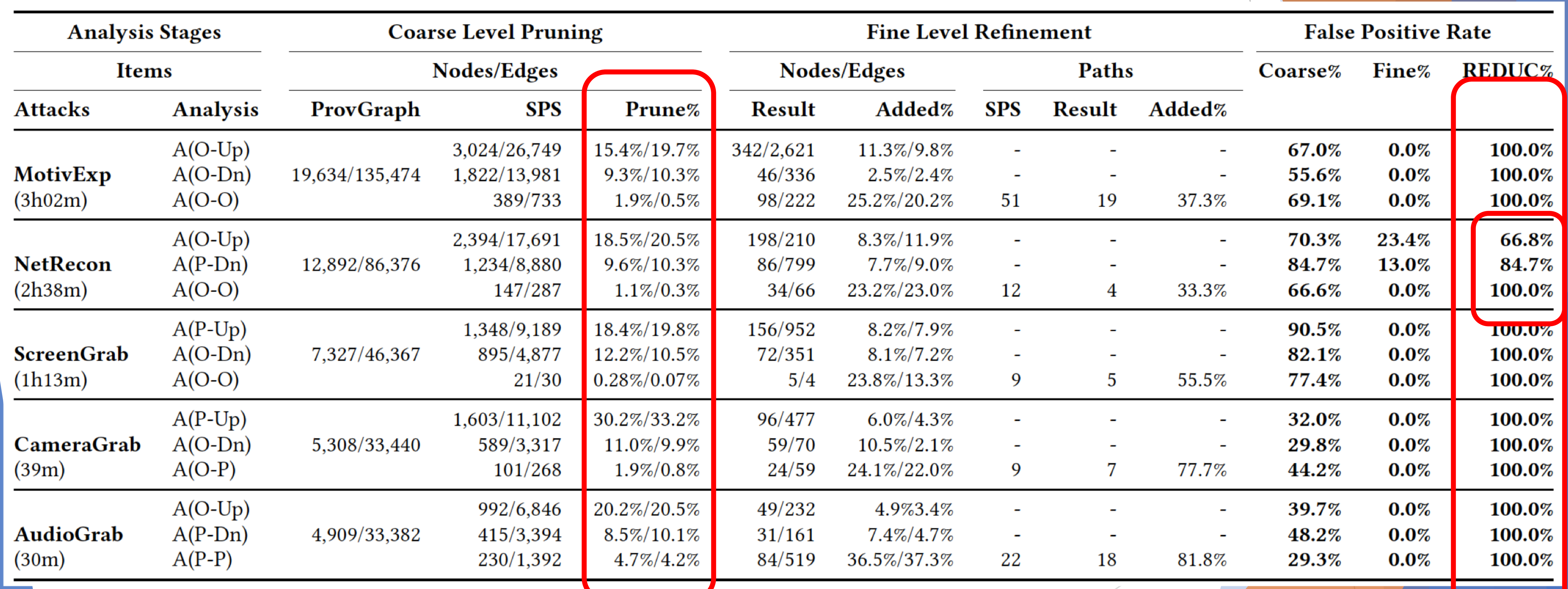
P2P Refinement



High Level Overview

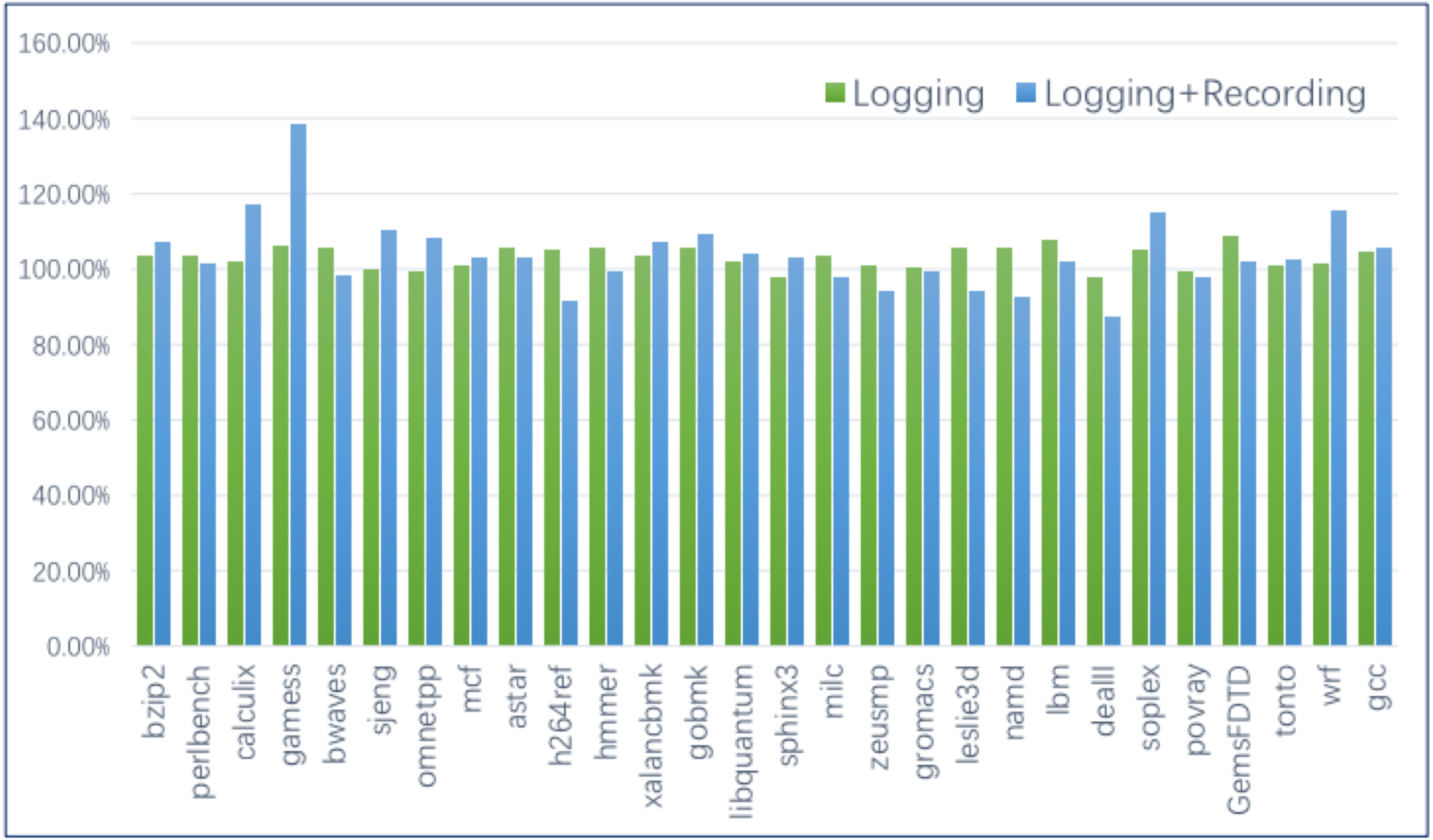


Results

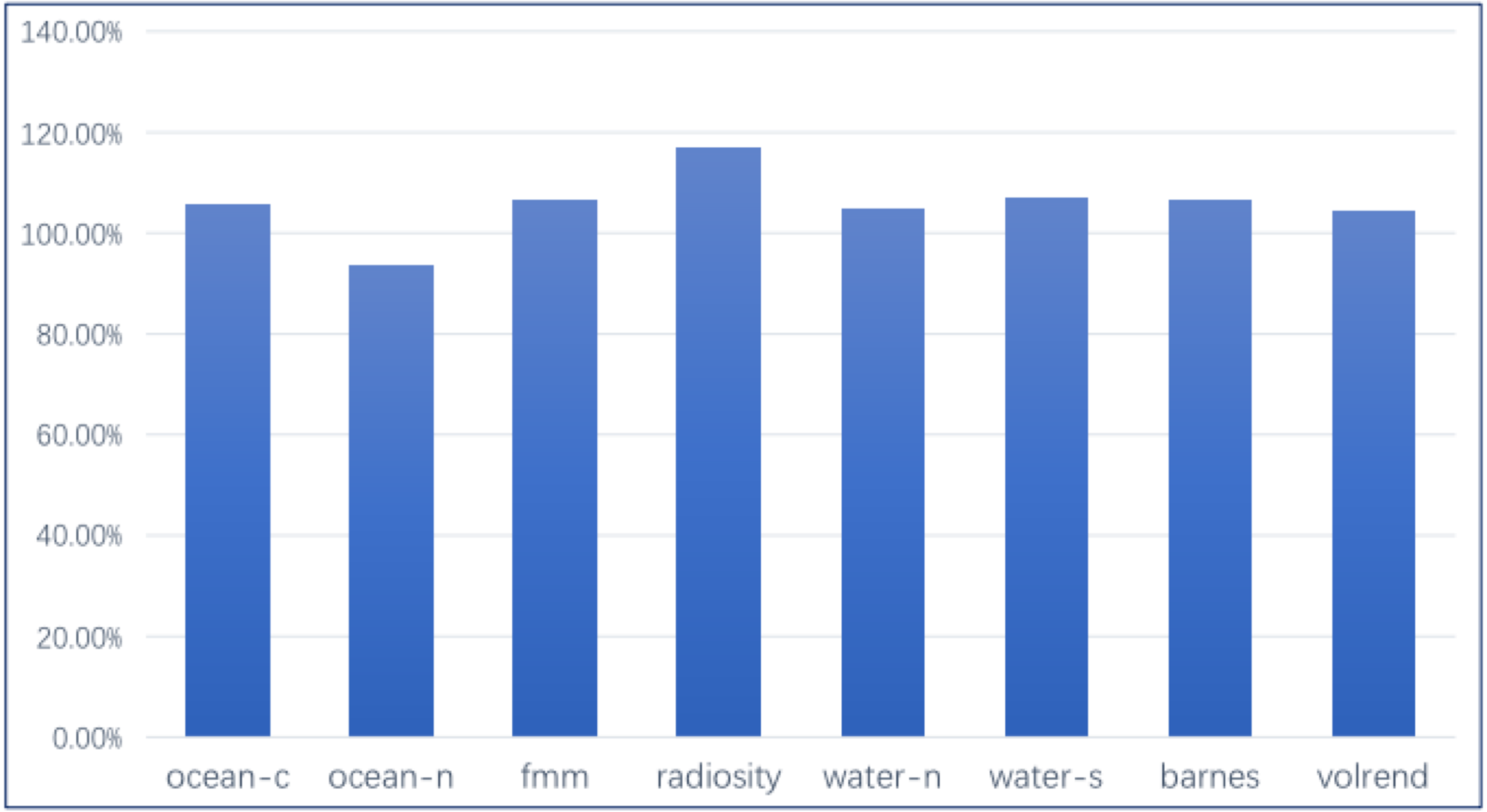


Analysis Stages		Coarse Level Pruning			Fine Level Refinement					False Positive Rate		
Items		Nodes/Edges			Nodes/Edges		Paths			Coarse%	Fine%	REDUC%
Attacks	Analysis	ProvGraph	SPS	Prune%	Result	Added%	SPS	Result	Added%			
MotivExp (3h02m)	A(O-Up)	19,634/135,474	3,024/26,749	15.4%/19.7%	342/2,621	11.3%/9.8%	-	-	-	**67.0%**	**0.0%**	**100.0%**
	A(O-Dn)		1,822/13,981	9.3%/10.3%	46/336	2.5%/2.4%	-	-	-	**55.6%**	**0.0%**	**100.0%**
	A(O-O)		389/733	1.9%/0.5%	98/222	25.2%/20.2%	51	19	37.3%	**69.1%**	**0.0%**	**100.0%**
NetRecon (2h38m)	A(O-Up)	12,892/86,376	2,394/17,691	18.5%/20.5%	198/210	8.3%/11.9%	-	-	-	**70.3%**	**23.4%**	**66.8%**
	A(P-Dn)		1,234/8,880	9.6%/10.3%	86/799	7.7%/9.0%	-	-	-	**84.7%**	**13.0%**	**84.7%**
	A(O-O)		147/287	1.1%/0.3%	34/66	23.2%/23.0%	12	4	33.3%	**66.6%**	**0.0%**	**100.0%**
ScreenGrab (1h13m)	A(P-Up)	7,327/46,367	1,348/9,189	18.4%/19.8%	156/952	8.2%/7.9%	-	-	-	**90.5%**	**0.0%**	**100.0%**
	A(O-Dn)		895/4,877	12.2%/10.5%	72/351	8.1%/7.2%	-	-	-	**82.1%**	**0.0%**	**100.0%**
	A(O-O)		21/30	0.28%/0.07%	5/4	23.8%/13.3%	9	5	55.5%	**77.4%**	**0.0%**	**100.0%**
CameraGrab (39m)	A(P-Up)	5,308/33,440	1,603/11,102	30.2%/33.2%	96/477	6.0%/4.3%	-	-	-	**32.0%**	**0.0%**	**100.0%**
	A(O-Dn)		589/3,317	11.0%/9.9%	59/70	10.5%/2.1%	-	-	-	**29.8%**	**0.0%**	**100.0%**
	A(O-P)		101/268	1.9%/0.8%	24/59	24.1%/22.0%	9	7	77.7%	**44.2%**	**0.0%**	**100.0%**
AudioGrab (30m)	A(O-Up)	4,909/33,382	992/6,846	20.2%/20.5%	49/232	4.9%3.4%	-	-	-	**39.7%**	**0.0%**	**100.0%**
	A(P-Dn)		415/3,394	8.5%/10.1%	31/161	7.4%/4.7%	-	-	-	**48.2%**	**0.0%**	**100.0%**
	A(P-P)		230/1,392	4.7%/4.2%	84/519	36.5%/37.3%	22	18	81.8%	**29.3%**	**0.0%**	**100.0%**

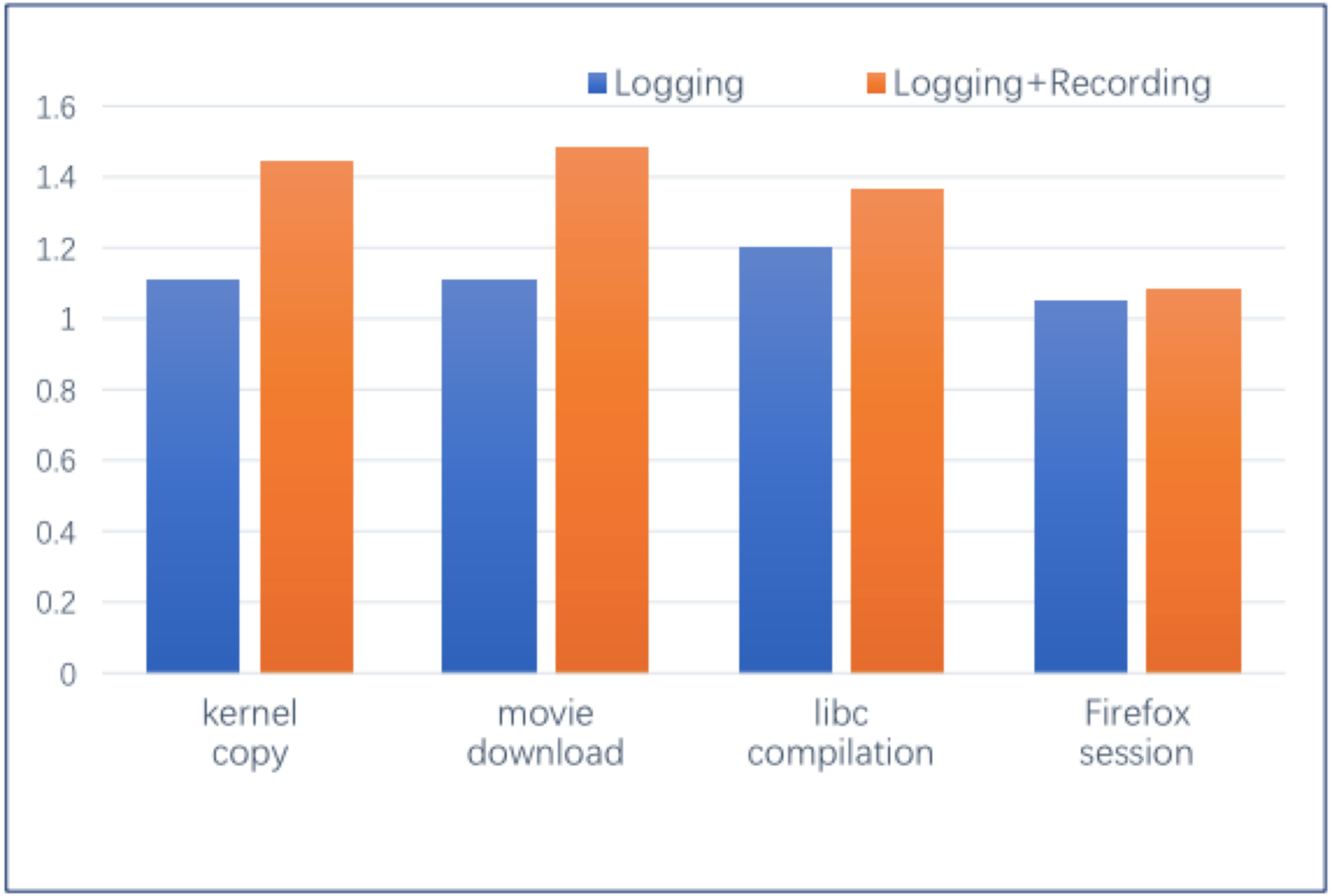
Runtime overhead: 3.22% SPEC CPU2006



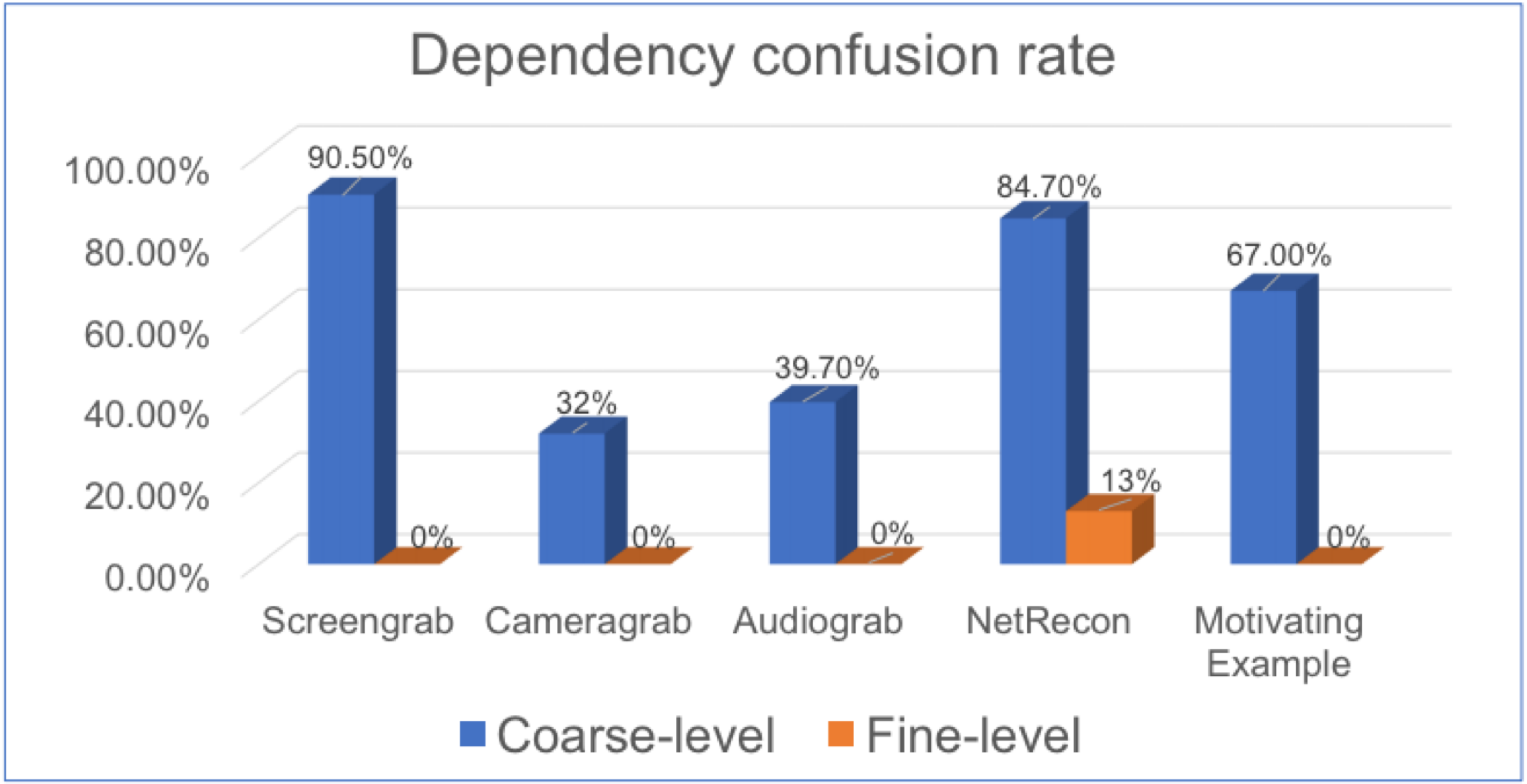
Multi-thread runtime overhead: 5.35% SPLASH-3



IO intensive application: less than 50%

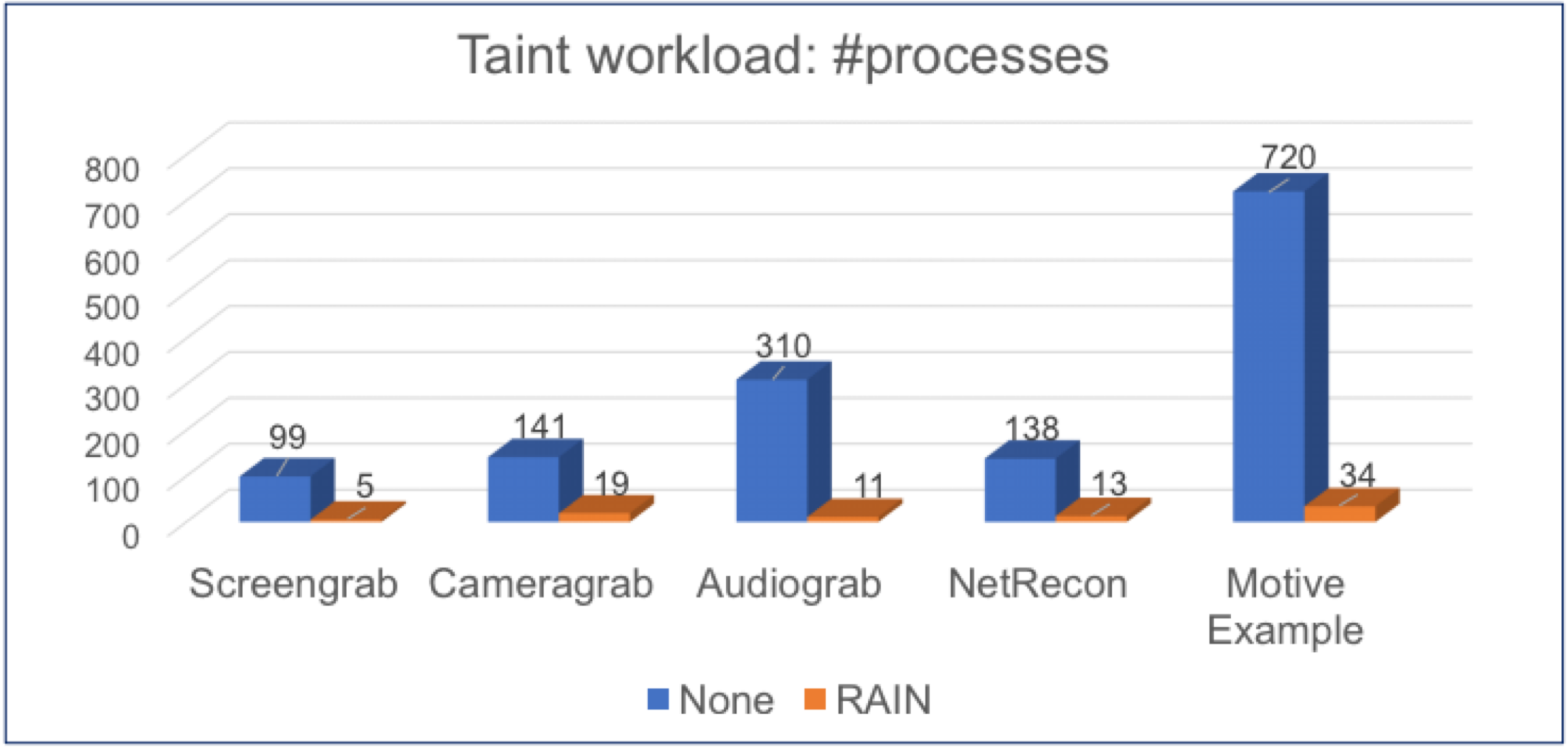


High analysis accuracy

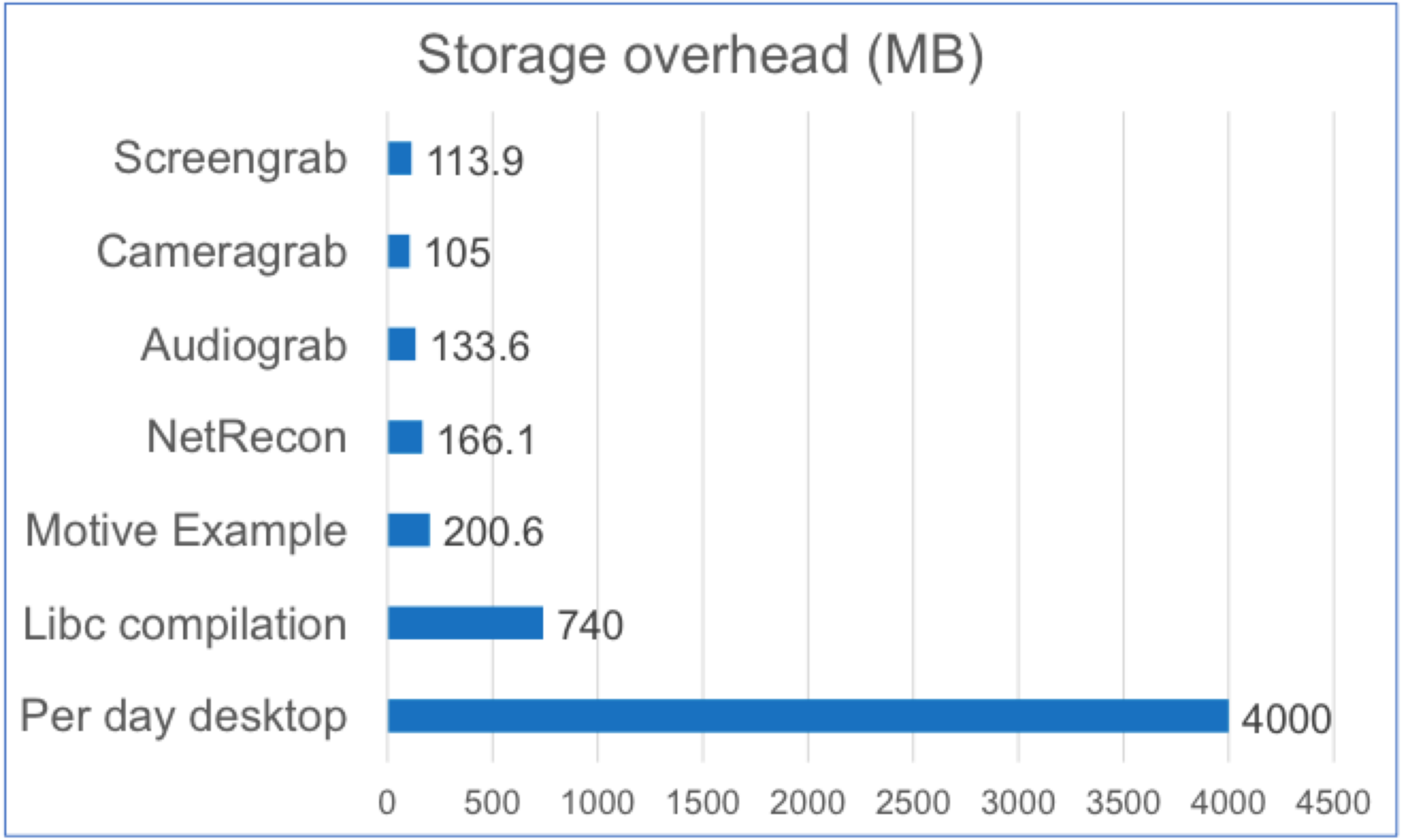


Scenarios from red team exercise of DARPA Transparent Computing program

Pruning effectiveness: ~94.2% reduction



Storage cost: ~4GB per day (1.5TB per year)



Thoughts for AI researchers

- Graph pruning in this work
  - Rule-based
- Can we use machine learning to trim graphs?