THE TASER INTRUSION RECOVERY SYSTEM

Ashvin Goel, Kenneth Po, Kamran Farhadi, Zheng Li,
Eyal de Lara – SOSP 2005

Intrusion Recovery

- Intrusion detection is only part of problem
- Also need:
  - To determine exploit and close it
  - Recovery data to a known good state
- Recovery is hard
  - Need to identify tainted data
  - Preserve legitimate date while removing tainted data
  - Legitimate actions and tainted actions intermingled
    - Even inter-dependent

Taser System

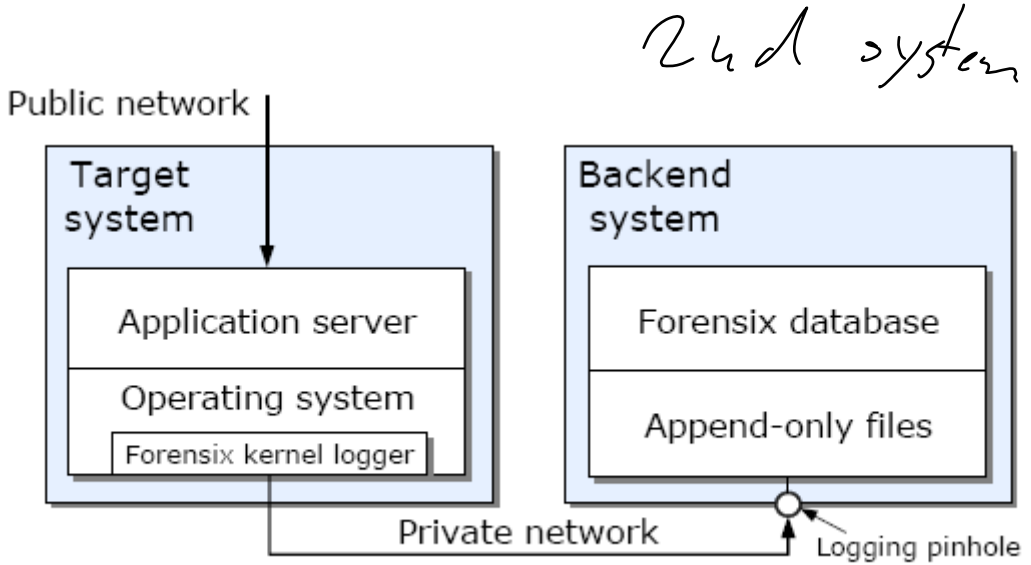
- Auditor
  - Responsible for logging actions
  - Online component
- Analyser
  - Responsible for identifying intrusion point and resulting tainted operations
  - Offline component
- Resolver
  - Takes log and analysis and rolls back file system to remove tainted actions

Forensix

Auditor

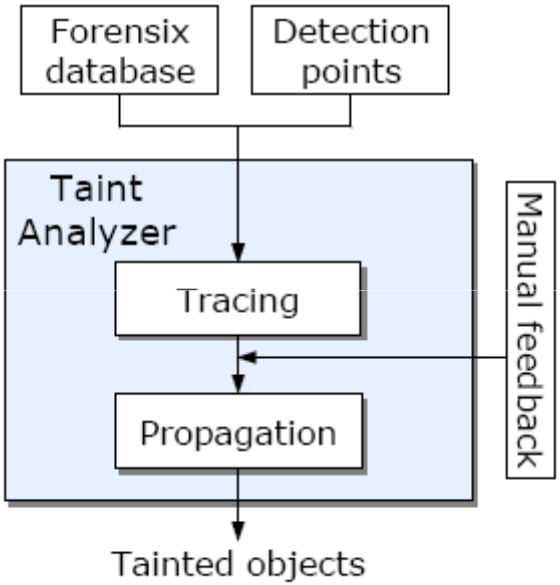
- Divides system up into objects
  - name, files, and attributes
- Records log in backend system

name op	: name id	→ directory name id, name
content op	: object id	→ content
attribute op	: object id	→ attribute



Analyser

- Two phases
  - Tracing phase
    - Like backtracker, help determine source of tainted operations
  - Propagation phase
    - Identifies tainted objects



Tracing and Propagating

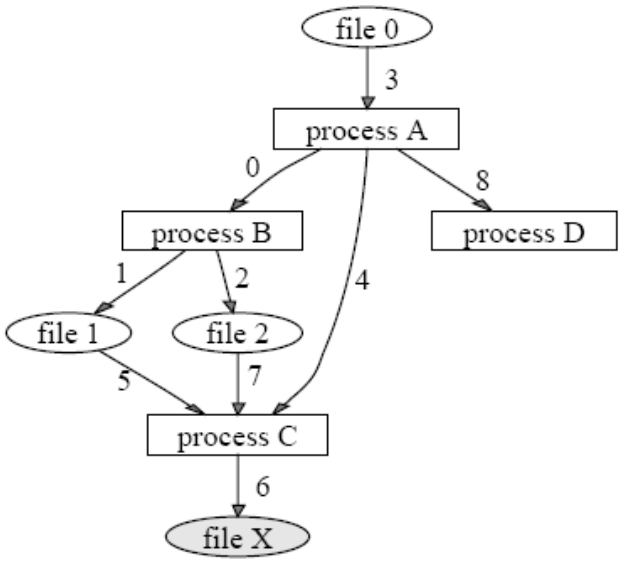
- Note the difference between
  - solely "back tracking" like GraphGen, and
  - The need for accurate dependencies for taint propagation

we want this for propagation

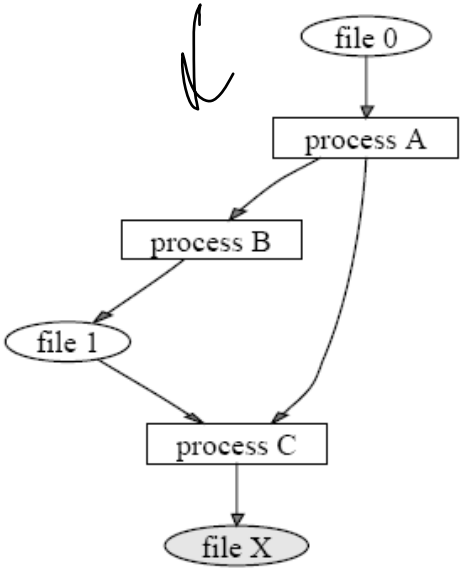
tracing

time 0: process A creates process B
time 1: process B writes file 1
time 2: process B writes file 2
time 3: process A reads file 0
time 4: process A creates process C
time 5: process C reads file 1
time 6: process C writes file X
time 7: process C reads file 2
time 8: process A creates process D

(a) event log



(b) dependency graph for complete event log



(c) dependency graph generated by GraphGen

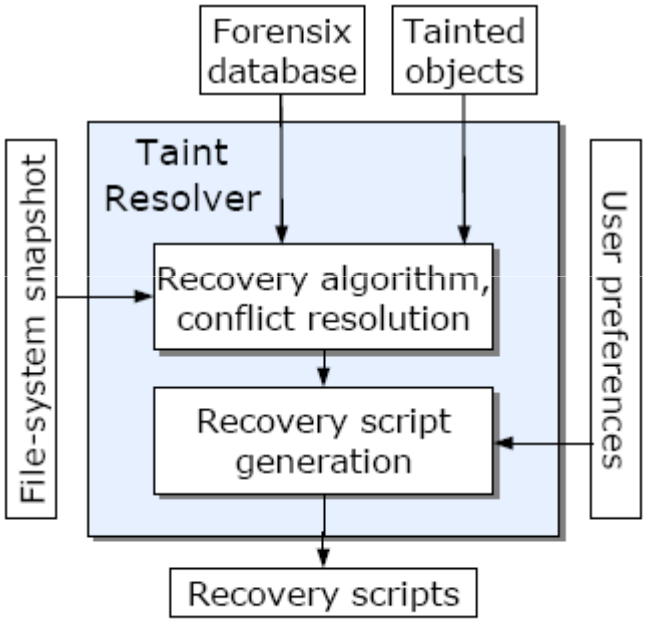
Analyser

- Uses set of dependency rules for tracing and propagation

Dependency Rule	Type of Operation	Operation
Process → Process	Fork	fork, vfork
	IPC, Signals	pipe, kill, mmap
Process → File	Write file content	creat, truncate, unlink, write
	Write file name	creat, link, sym-link, rename, un-link
	Write file attributes	create, unlink, chown, chmod
File → Process	Execute	execve
	Read file content	read
	Read file name	open, truncate, chown, chmod
	Read file attributes	open, truncate, chown, chmod
Process → Socket	Write	write, socketcall, sendfile
Socket → Process	Read	read, socketcall

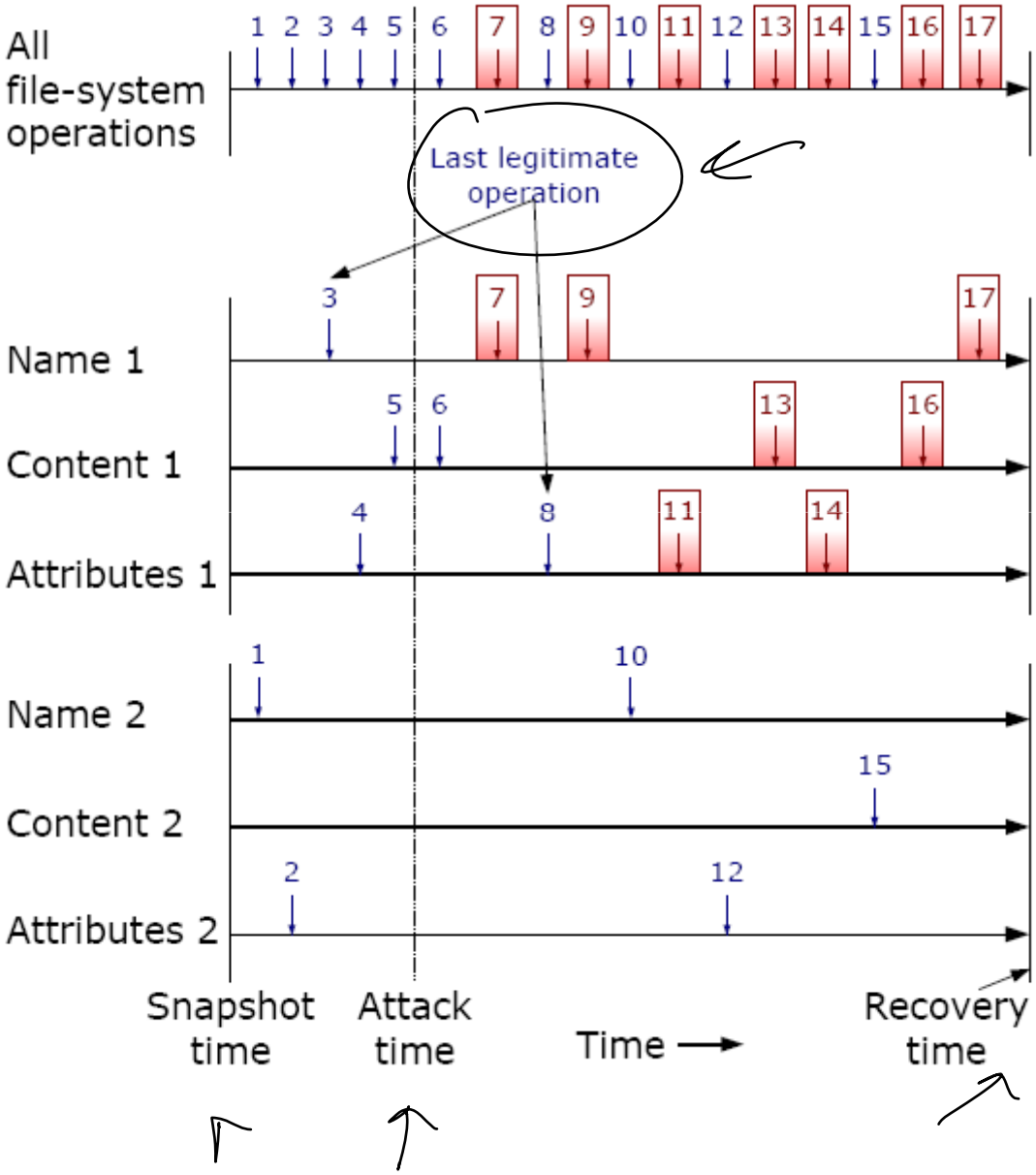
Resolver

- Reverts tainted objects
- Uses a file system snapshot
  - Easiest to revert to last clean snapshot
  - Lose legitimate operations
- Replays legitimate operations from the log



Recovery

- Simple Redo
  - Replay up to last legitimate operation
  - Correct
- Can be slow if a lot of operations have transpired
- Note object 2 needs no recovery, but is still replayed

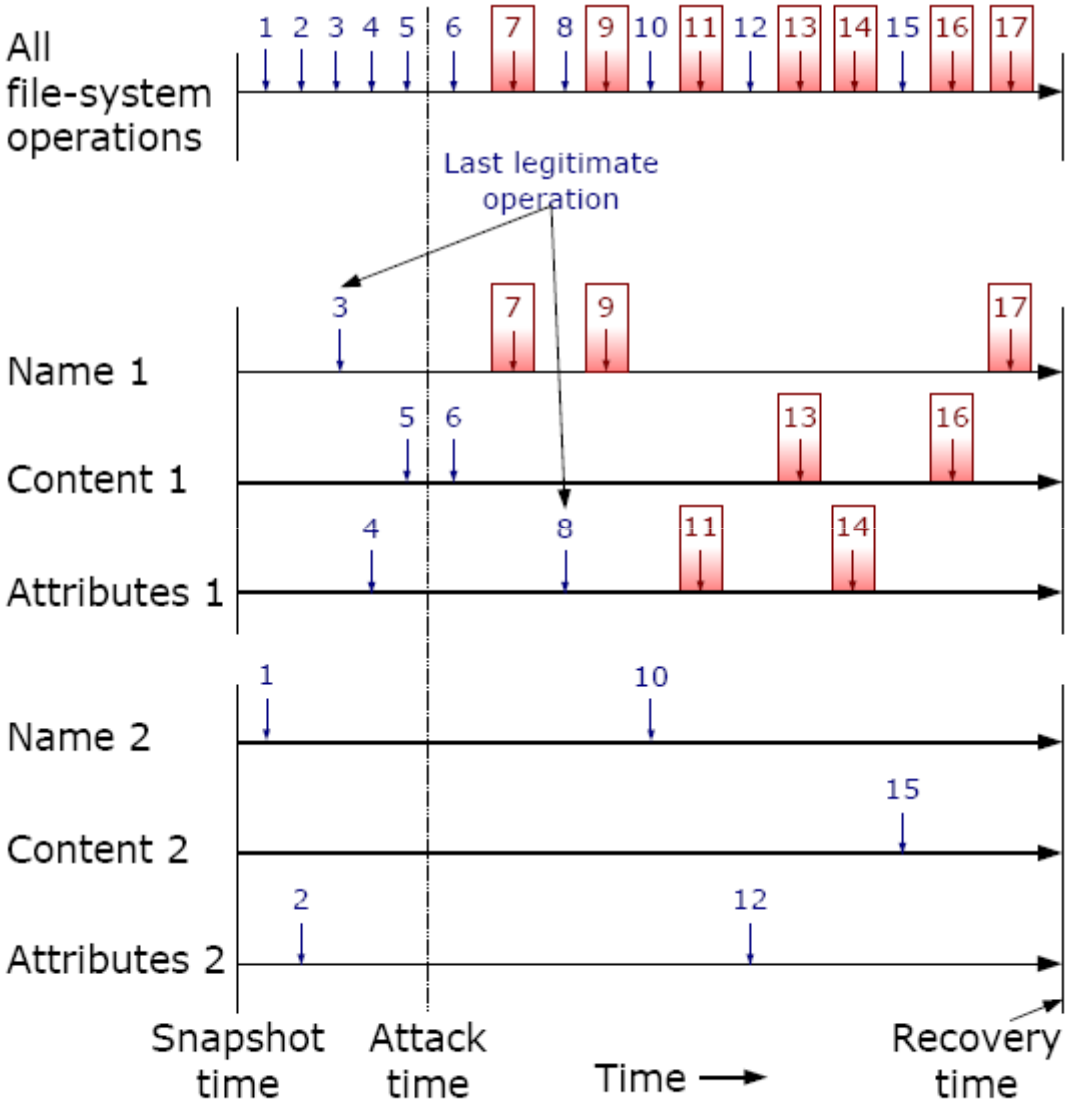


Recovery Observations

- Recovery time file system state correct for untainted object
- Attributes and Name change override previous changes
- Optimisations???

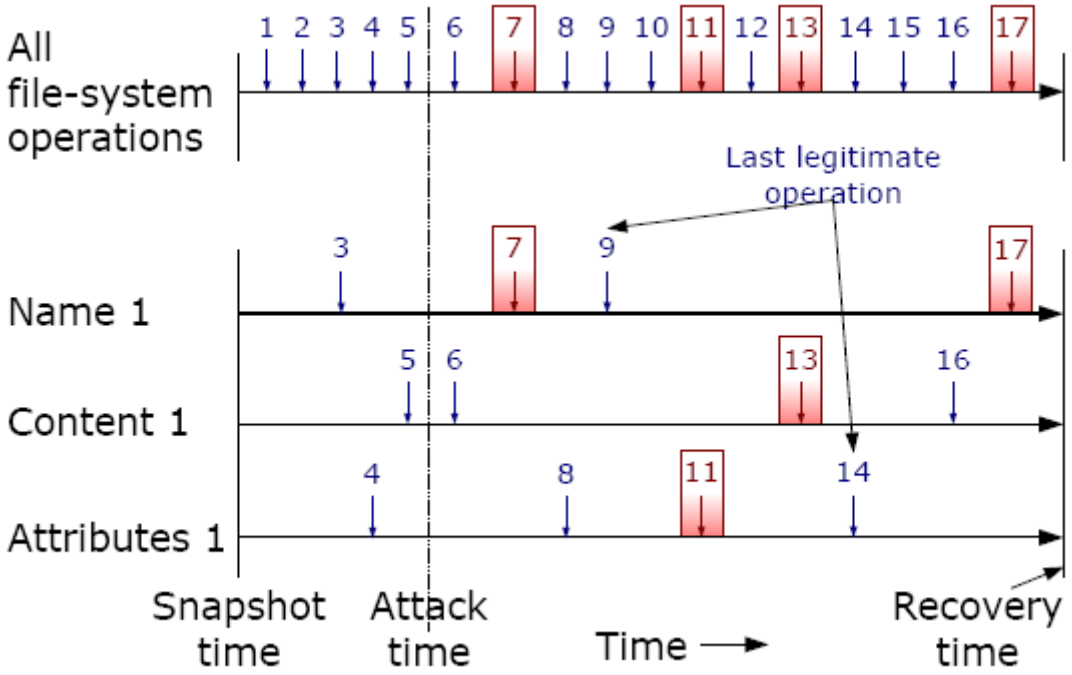
Optimisations

- Use the recovery time versions for untainted objects
- Only replay the last legitimate attribute or name operation



“Tainted” Legitimate Operations

- Normal IPC between tainted client and server
  - server taints all other clients
- Legitimate file stored in tainted directory
- Legitimate data written to file with tainted attributes



Optimistic Analysis

- White lists
- Intervals
- Ignore classes of events in analysis phase
  - Creates the opportunity for conflicts
  - E.g. NoIAN allows legitimate operations in tainted directory
    - Conflict: Remove tainted directory and preserve legitimate file??

Policy	Description	Conflicts
Conservative	All operations shown in Table 2	none
NoI	Ignores IPC, signals	none
NoIA	Ignore reading file attributes	attribute conflicts
NoIAN	Ignore reading file attributes and names	attribute, name conflicts
NoIANC	Ignore reading file attributes, names, content	attribute, name, content conflicts

Accuracy Results

Scenario	Recovery Actions	Recovery one day after attack				Recovery one week after attack			
		Snapshot	NoI	NoIAN	NoIANC	Snapshot	NoI	NoIAN	NoIANC
Illegal storage	507	633, 0	2, 0	0, 0	0, 0	4154, 0	7, 0	0, 2	0, 2
Content destruction	739	1877, 0	0, 0	0, 0	0, 0	5338, 0	0, 0	0, 0	0, 1
Unhappy student	167	1106, 0	2, 0	0, 0	0, 1	4617, 0	4, 0	0, 0	0, 1
Compromised database	3	814, 0	0, 0	0, 0	0, 2	2557, 0	0, 0	0, 0	0, 2
Software installation	350	1542, 0	1, 0	0, 0	0, 0	5006, 0	1, 0	0, 0	0, 0
Inexperienced admin	39	1366, 0	11, 0	11, 0	0, 0	4982, 0	11, 0	11, 0	0, 0
Inexperienced admin (single interval)	39	1366, 0	415, 0	415, 0	125, 0	4982, 0	701, 0	701, 0	126, 0

false positives
tainted legitimate objects

false negatives
objects that should have been tainted

Performance

- Target
  - 7% throughput reduction on webstone
- Backend

Number of operations	13.3 Million
Size of events in flat file	1.9 GB
Size of database	2.3 GB
Database loading time	36.3 min

Table 7: Average daily backend statistics

Conclusions

- Taser can identify and recover legitimate snapshot of the system
- Trade-off in optimistic analysis
  - Misses some intrusion activity
  - Preserves more legitimate data