Intrusion Recovery using Selective Re-execution

Taesoo Kim, Xi Wang,
Nickolai Zeldovich, M. Frans Kaashoek

MIT CSAIL

Attackers routinely compromise system integrity

'Google' Hackers Had...

www.wired.com/threatlevel/2010/03/source-code-hacks/

'Google' Hackers Had Ability to Alter Source Code

By Kim Zetter March 3, 2010 | 11:05 pm | Categories: Cybersecurity, Hacks and Cracks

Hackers who breached Google and other companies in January targeted source-code management systems, security firm McAfee asserted Wednesday. They manipulated a little-known trove of security flaws that would allow easy unauthorized access to the intellectual property the system is meant to

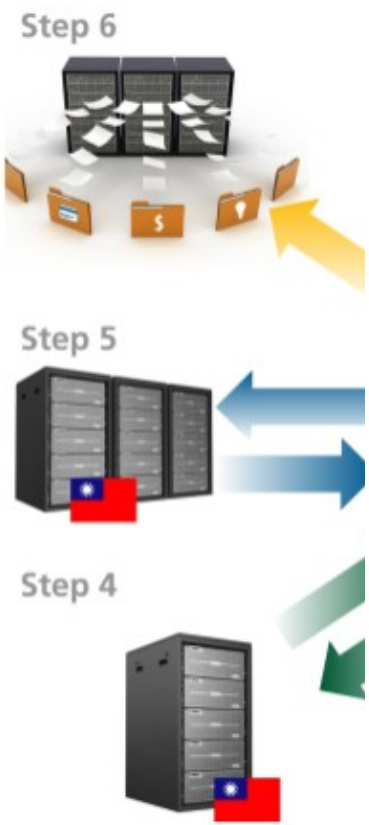
Attackers routinely compromise system integrity

'Google' Hackers Had...

www.wired.com/threatlevel/2010/03/source-code-hacks/

'Google' Hackers Hac

By Kim Zetter March 3, 2010 | 11:05 pm |



Hackers who breached Google and
systems, security firm McAfee asser

The Harvard Crimson | Faculty Council Votes To Dismiss Student - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.thecrimson.com/article/2010/5/14/facult

Google

The Harvard Crimson | Faculty ...

The Harvard Crimson

May 14, 2010

Faculty Council Votes To Dismiss Student

By Eric P. Newcomer and Naveen N. Srivatsa, CRIMSON STAFF WRITERS

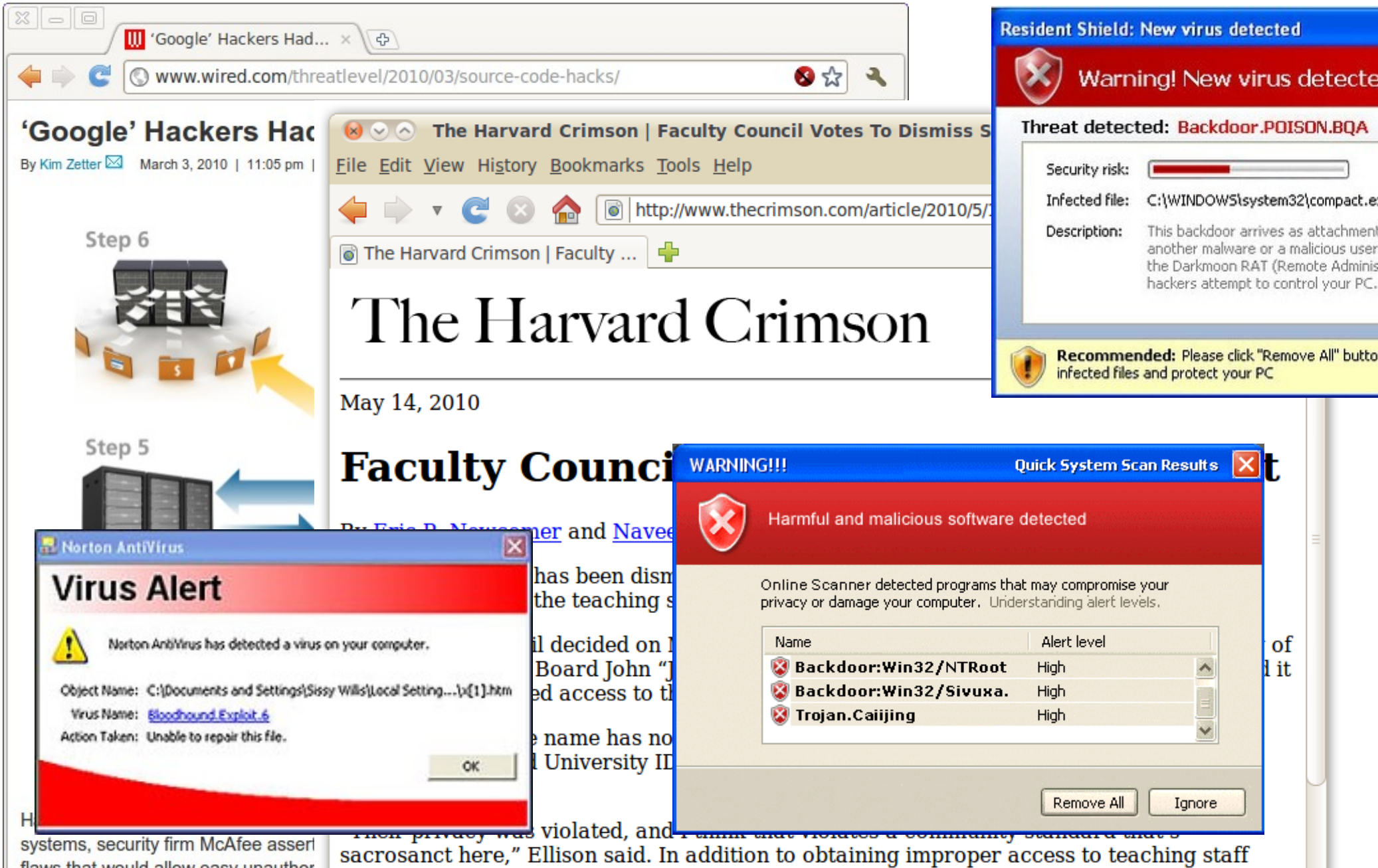
An undergraduate has been dismissed from Harvard College after allegedly hacking into online accounts of the teaching staff in his courses.

The Faculty Council decided on May 5 to dismiss the student, who, according to Secretary of the Administrative Board John "Jay" L. Ellison, collected confidential information and used it to gain unauthorized access to the Registrar's Office's grading portal.

The student, whose name has not been released by administrators, designed a website to obtain the Harvard University IDs and the personal identification numbers (PIN) of his instructors.

"Their privacy was violated, and I think that violates a community standard that's sacrosanct here," Ellison said. In addition to obtaining improper access to teaching staff

Attackers routinely compromise system integrity



Compromises inevitable

- Difficult to write bug-free software
- Administrators mis-configure policies
- Users choose weak, guessable passwords

Compromises inevitable

- Difficult to write bug-free software
- Administrators mis-configure policies
- Users choose weak, guessable passwords

- Need both “proactive” security,
  and “reactive” recovery mechanisms

Limited existing recovery tools

- Anti-virus tools
  - Only repair for predictable attacks
- Backup tools
  - Restoring from backup discards *all* changes

Limited existing recovery tools

- Anti-virus tools
  - Only repair for predictable attacks
- Backup tools
  - Restoring from backup discards *all* changes
- Administrators spend days or weeks manually tracking down all effects of the attack
  - No guarantee if they found everything

Challenge: disentangle changes by *attacker* and *legitimate user*

- Adversary could have modified many files directly
- Legitimate processes may have been affected
  - Users ran trojaned `pdflatex` or `ls`
  - SSH server read a modified `/etc/passwd`
- Those processes are now suspect as well

Our approach: help users disentangle on one machine

- Record history of all computations on machine
- After intrusion found, roll back affected objects
- Re-execute actions that were indirectly affected
- Minimize user input required to disentangle
  - User edited attacker's file with `emacs`
  - External effects outside of our control

Contributions

- New approach to *system-wide* intrusion recovery
  - *Action history graph* tracks computations and repairs
  - Techniques: *re-execution*, *predicates*, and *refinement*
- *Retro*: prototype recovery system for Linux
  - Recovers from 10 real-world and synthetic attacks
  - No user input required in most cases

Contributions

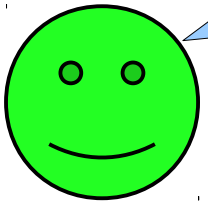
- New approach to *system-wide* intrusion recovery
  - *Action history graph* tracks computations and repairs
  - Techniques: *re-execution*, *predicates*, and *refinement*
- *Retro*: prototype recovery system for Linux
  - Recovers from 10 real-world and synthetic attacks
  - No user input required in most cases

- Instead of spending days on manual recovery, admin can use Retro to automatically recover, and ensure that *all* effects of attack are caught

Example attack scenario

- Attacker modifies `/etc/passwd` to add new account
- Installs trojan `pdflatex`, `ls` to restart, hide botnet

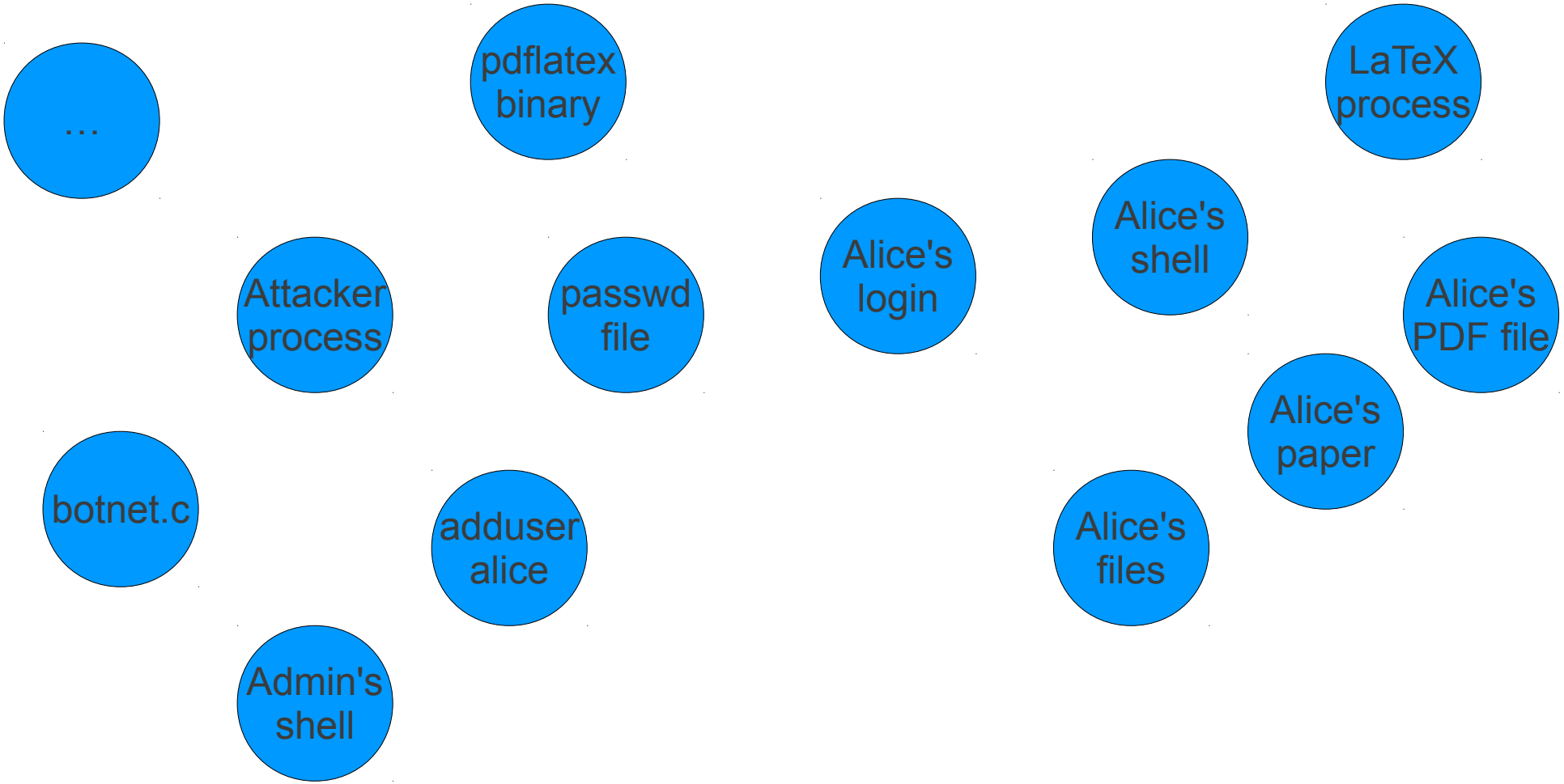
- Admin modifies `/etc/passwd` to add account for Alice



- Alice logs in via SSH
- SSH server reads `/etc/passwd`
- Alice runs trojaned `pdflatex`, `ls`

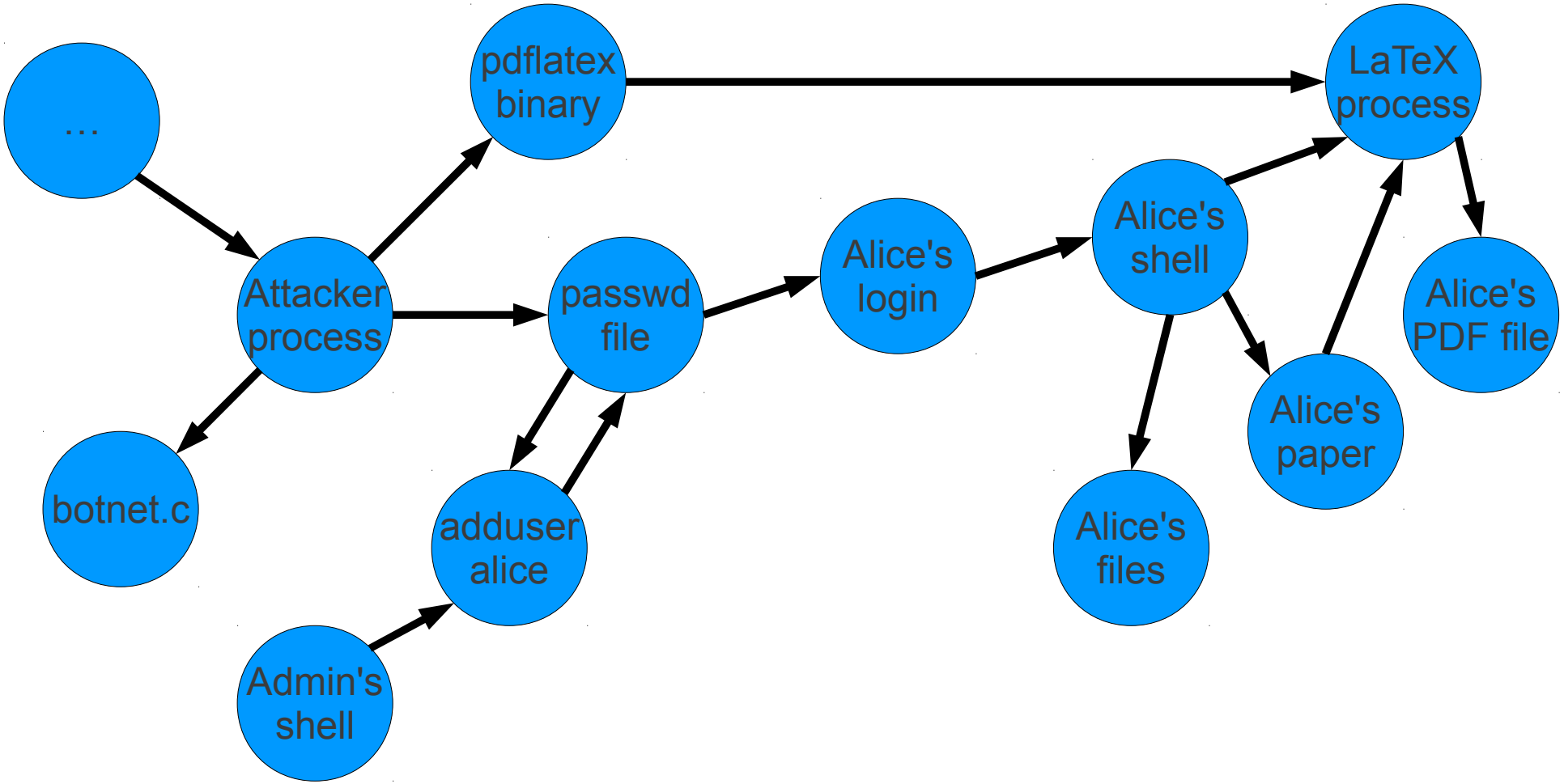
- Attacker not targeting Alice, wants to run botnet

Strawman 1: Taint tracking



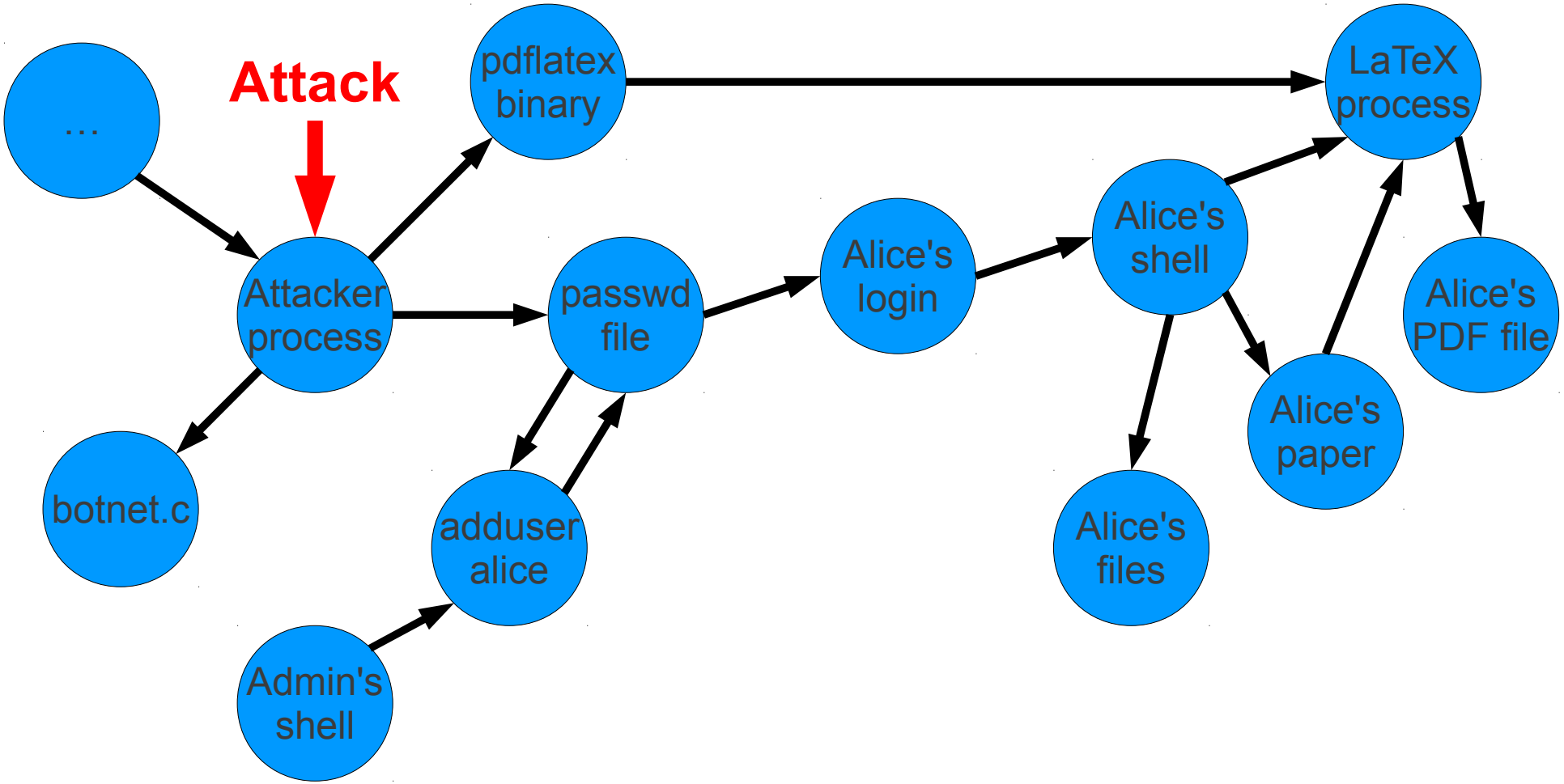
Strawman 1: Taint tracking

- Log all OS-level dependencies in system



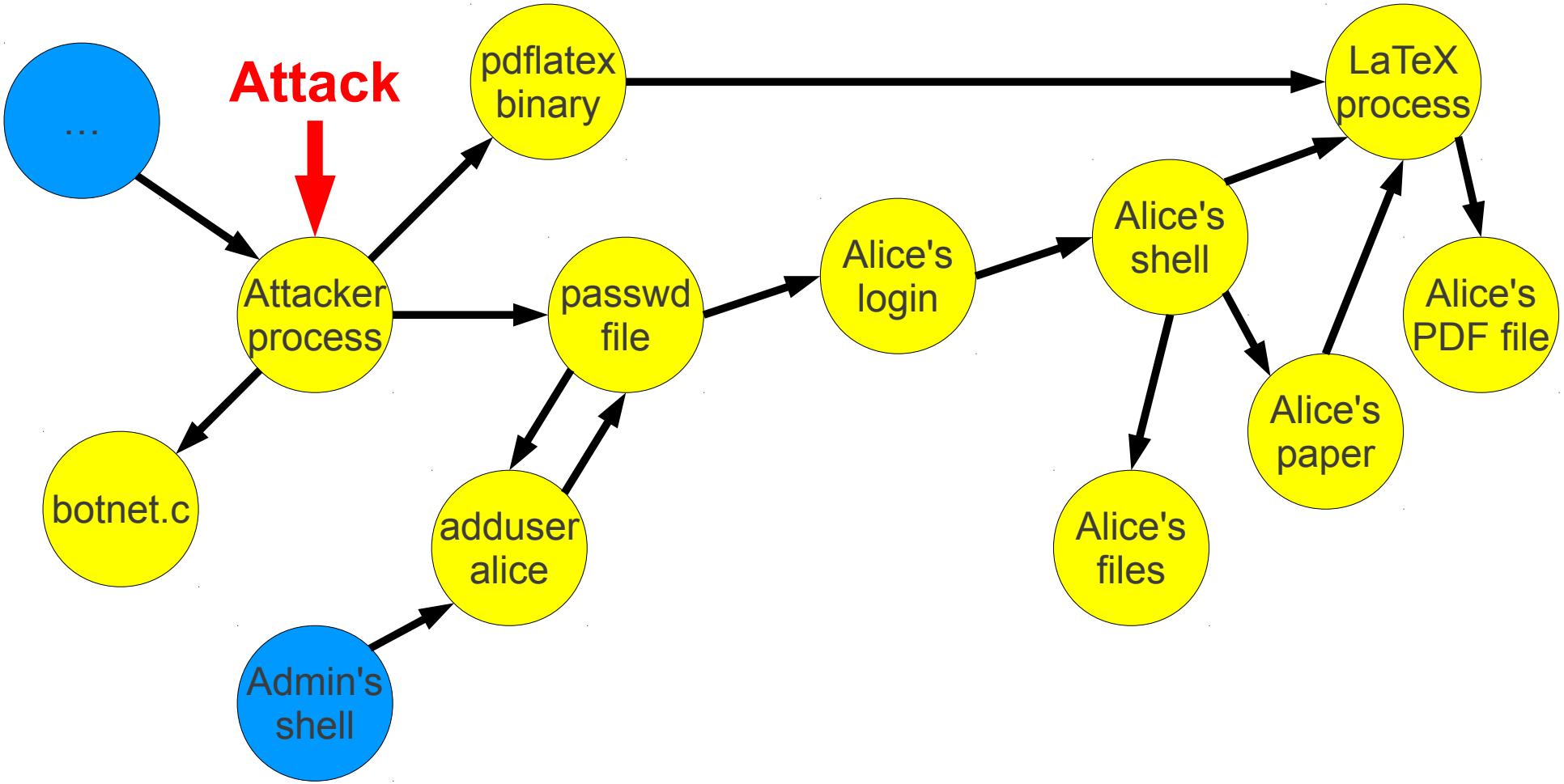
Strawman 1: Taint tracking

- Given attack, track down all affected files, and restore just those files from backup



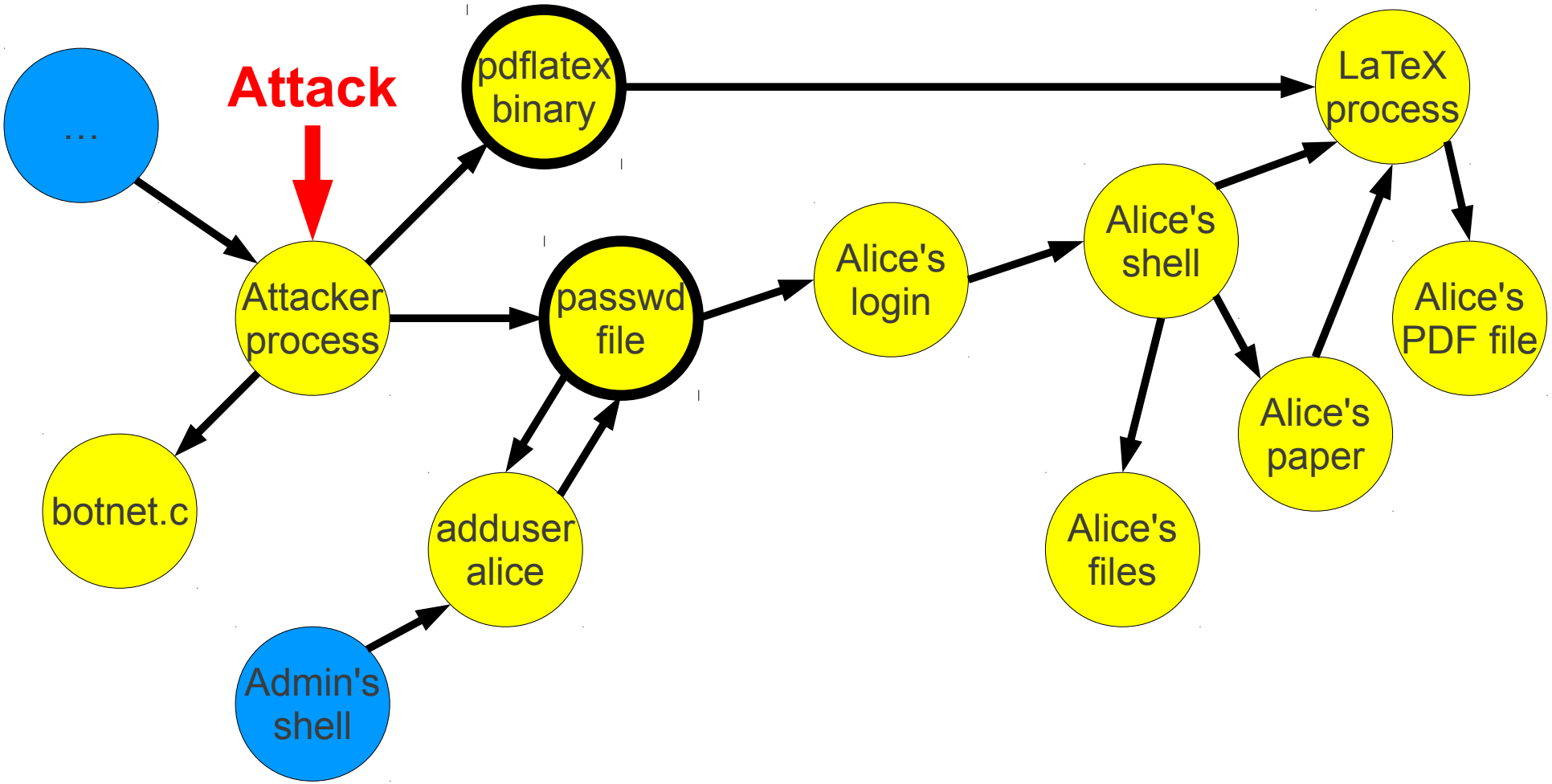
Strawman 1: Taint tracking

- Given attack, track down all affected files, and restore just those files from backup



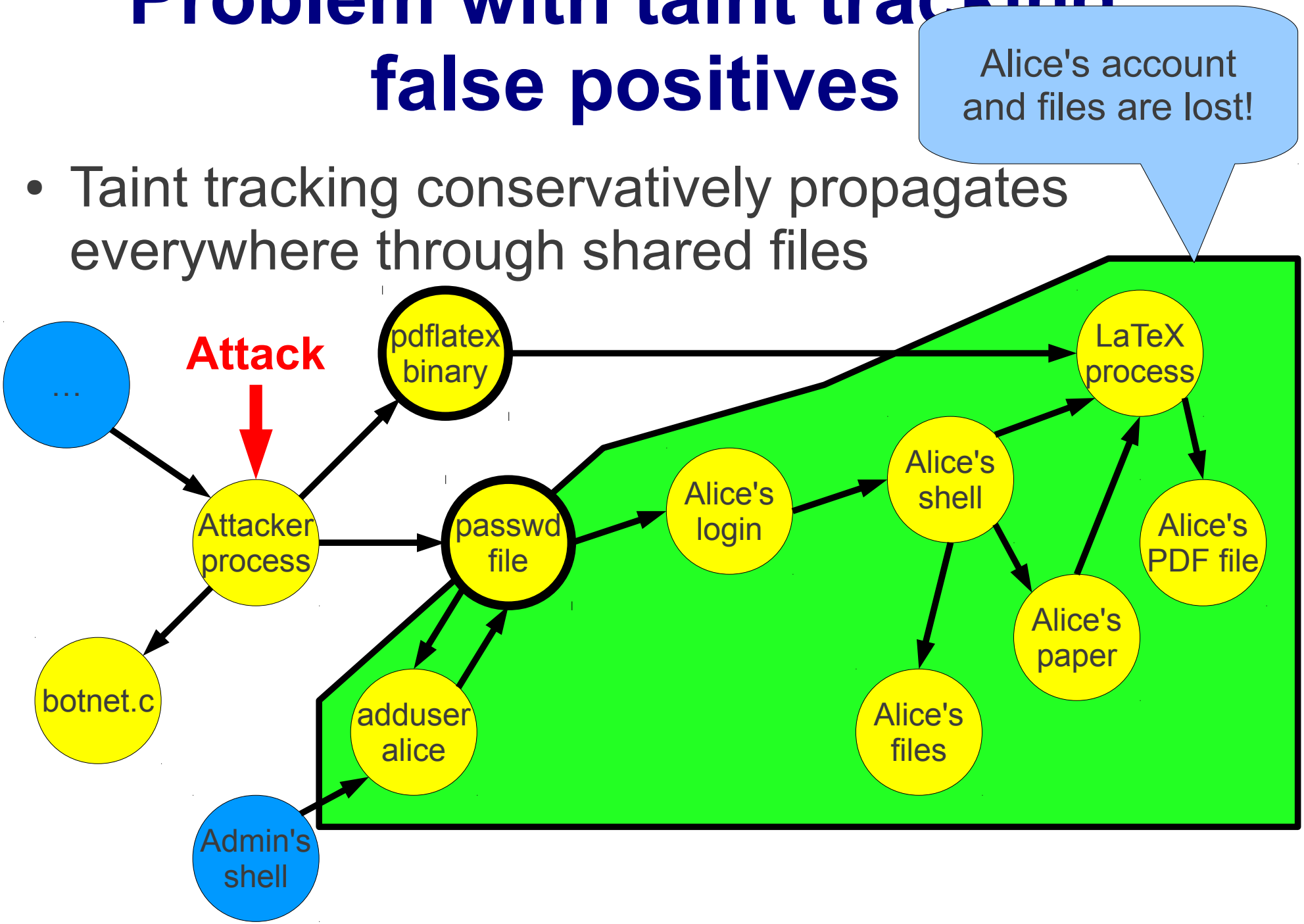
Problem with taint tracking: false positives

- Taint tracking conservatively propagates everywhere through shared files



Problem with taint tracking: false positives

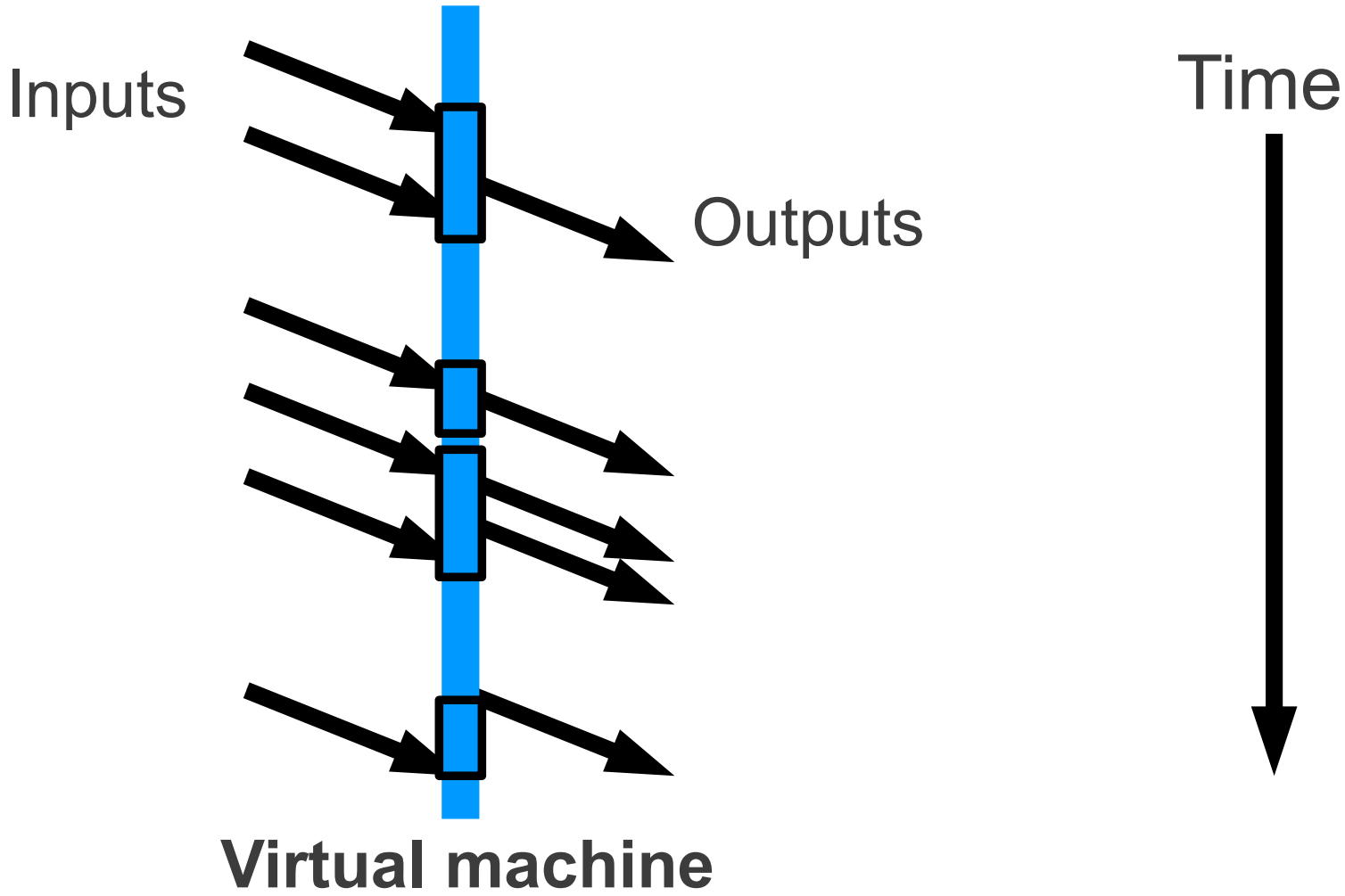
- Taint tracking conservatively propagates everywhere through shared files



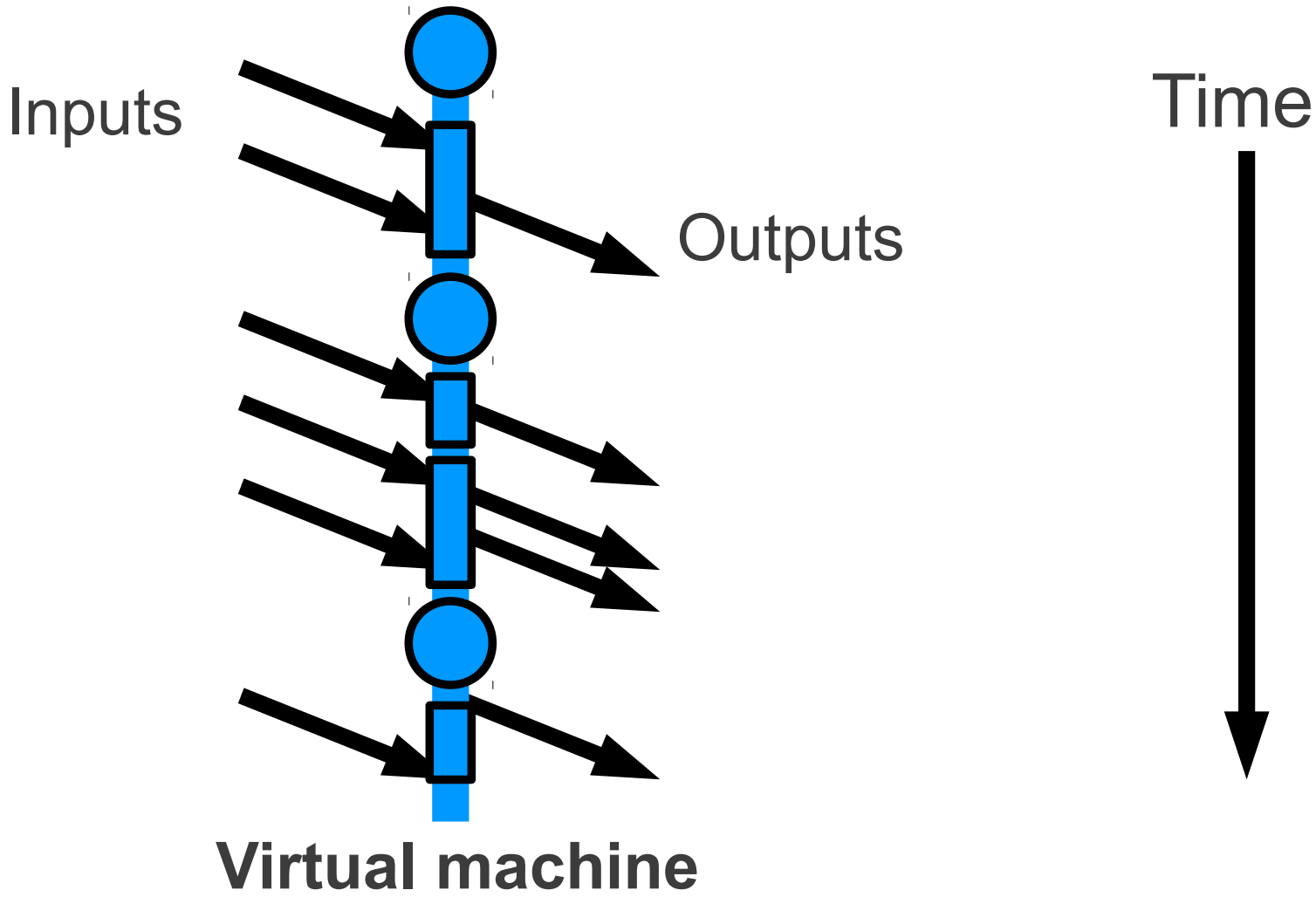
Strawman 2: VM

Virtual machine

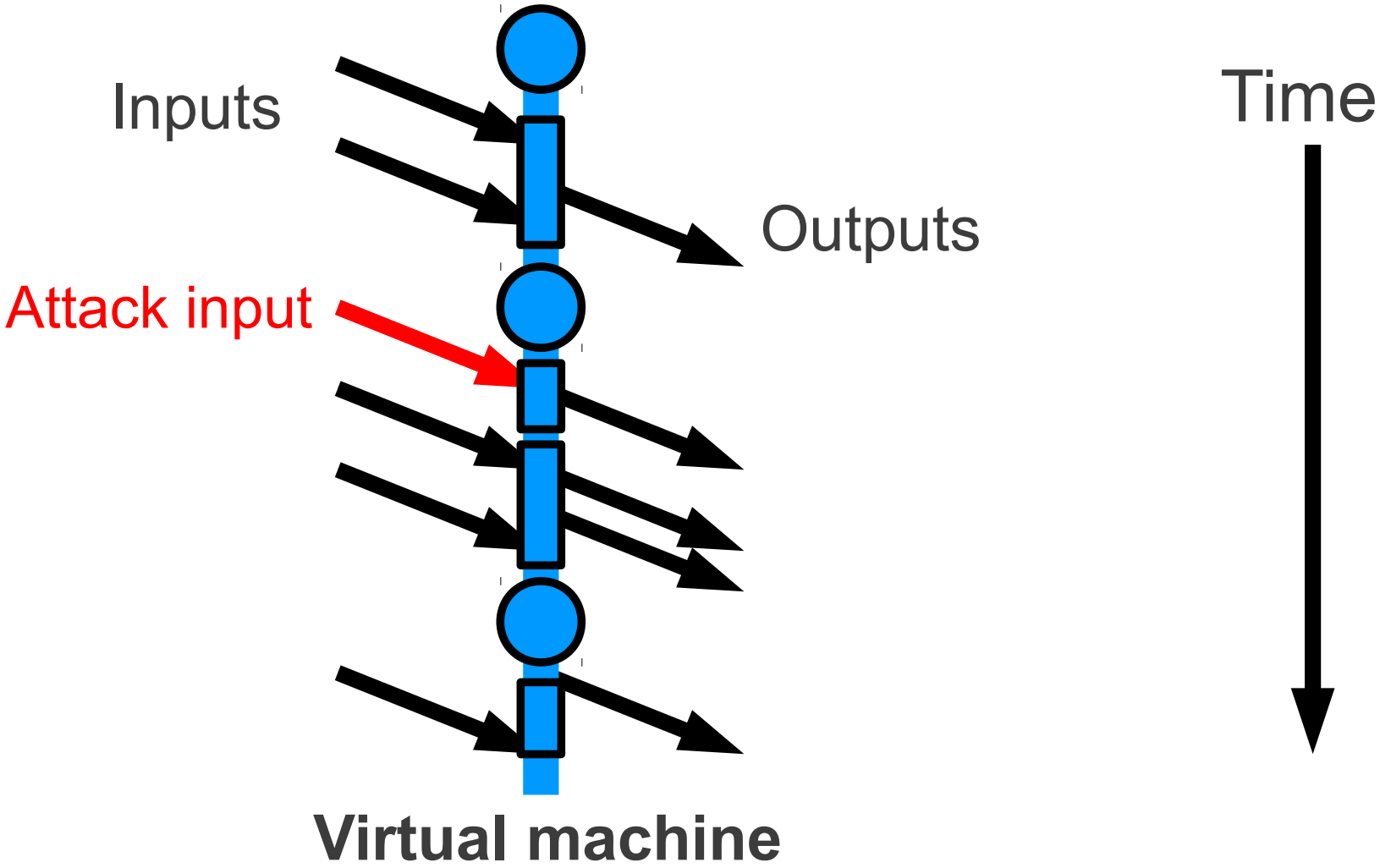
Strawman 2: VM



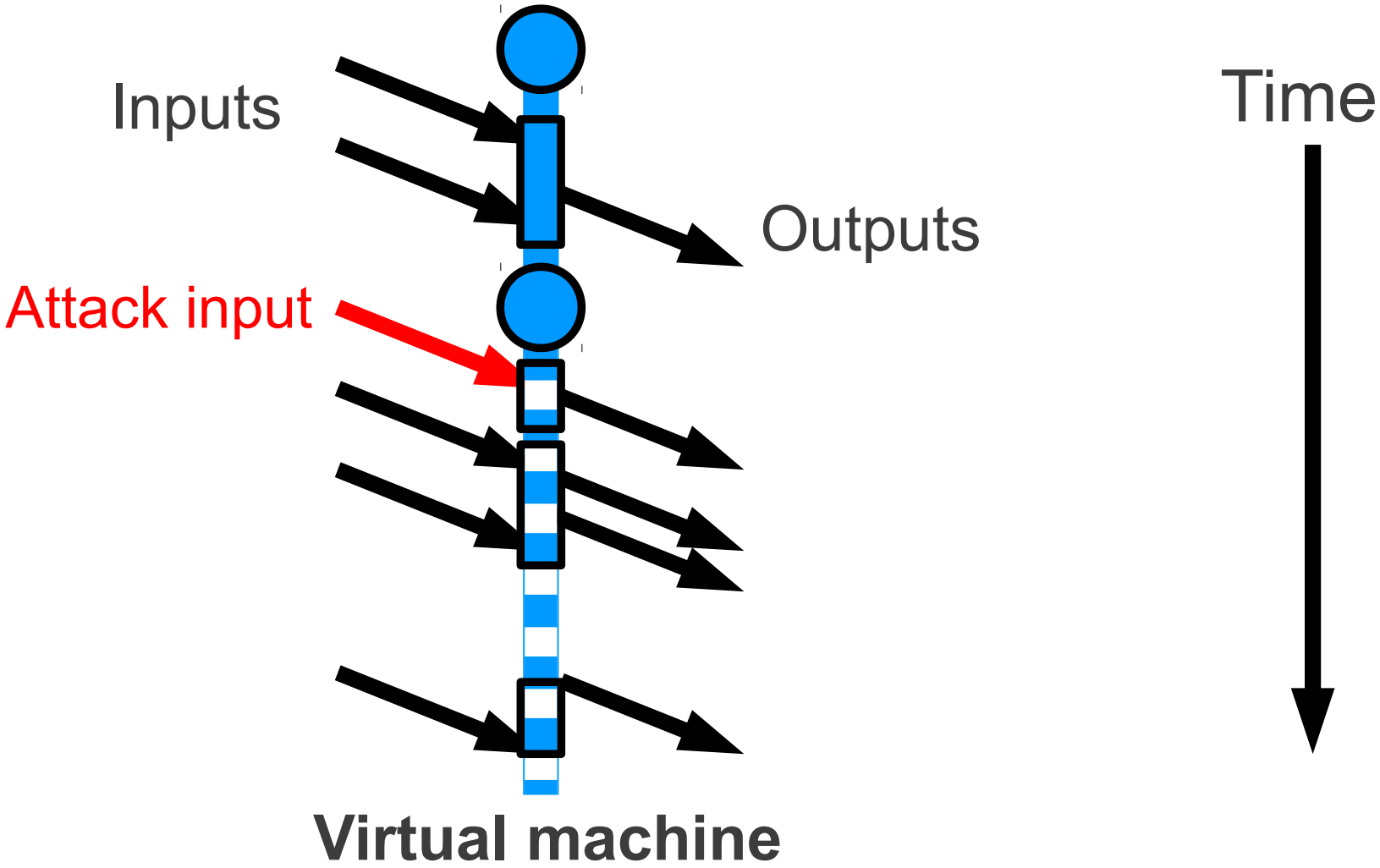
Periodic VM checkpoints



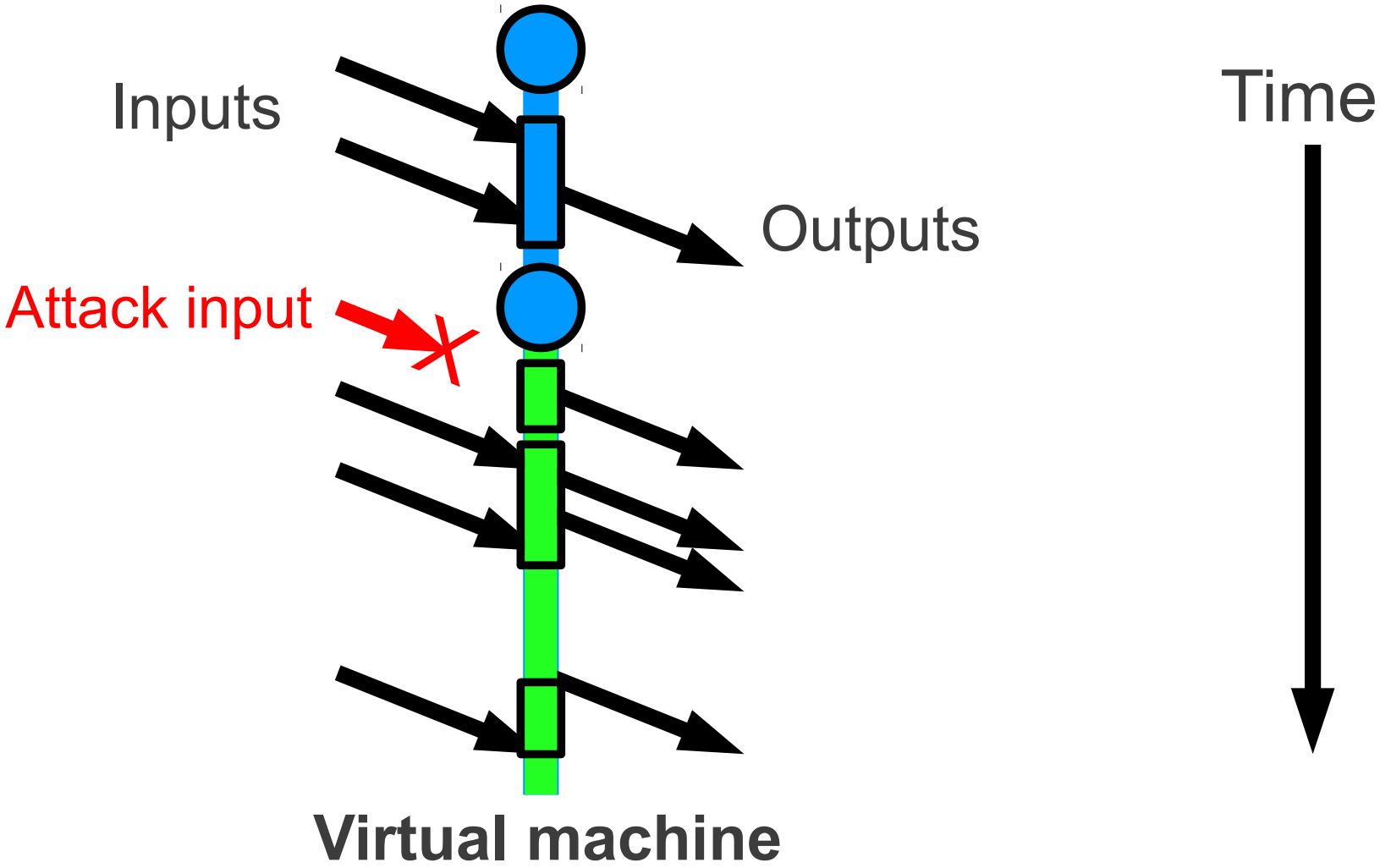
Step 1: identify attack input



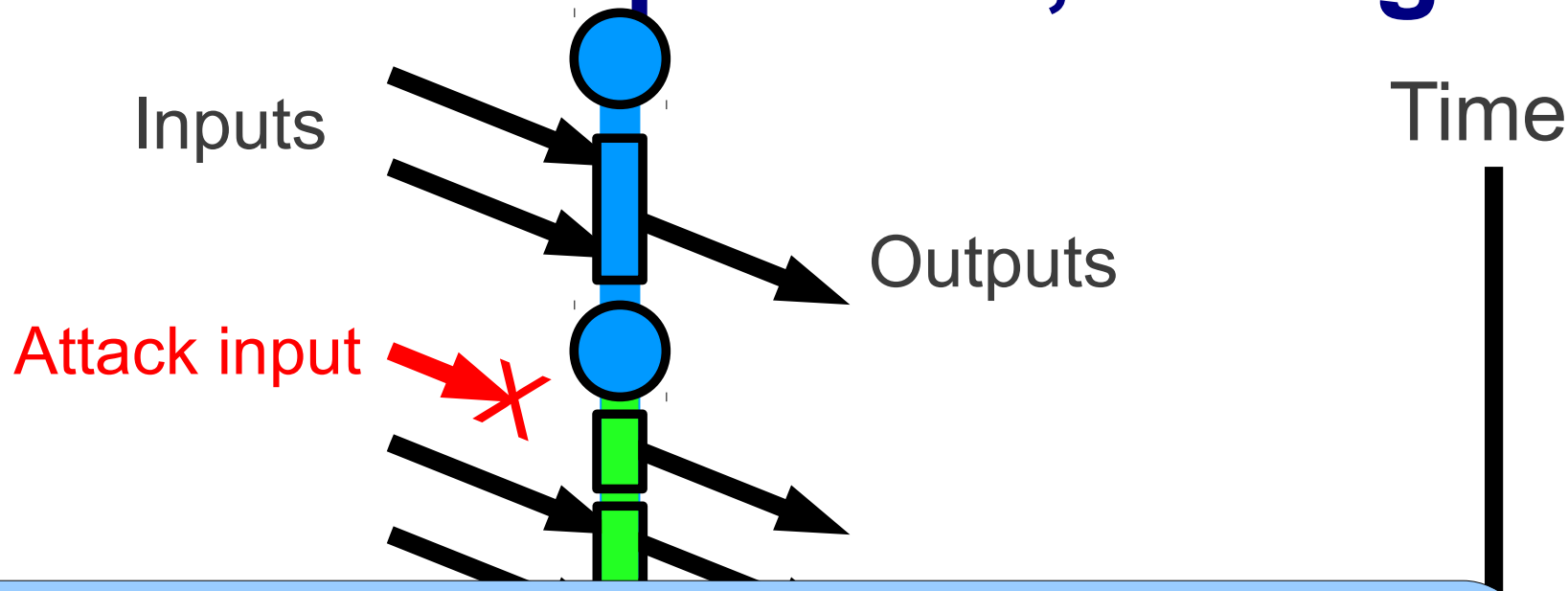
Step 2: roll back to checkpoint



Step 3: replay non-attack inputs



Problem with VM strawman: re-execution is expensive, diverges

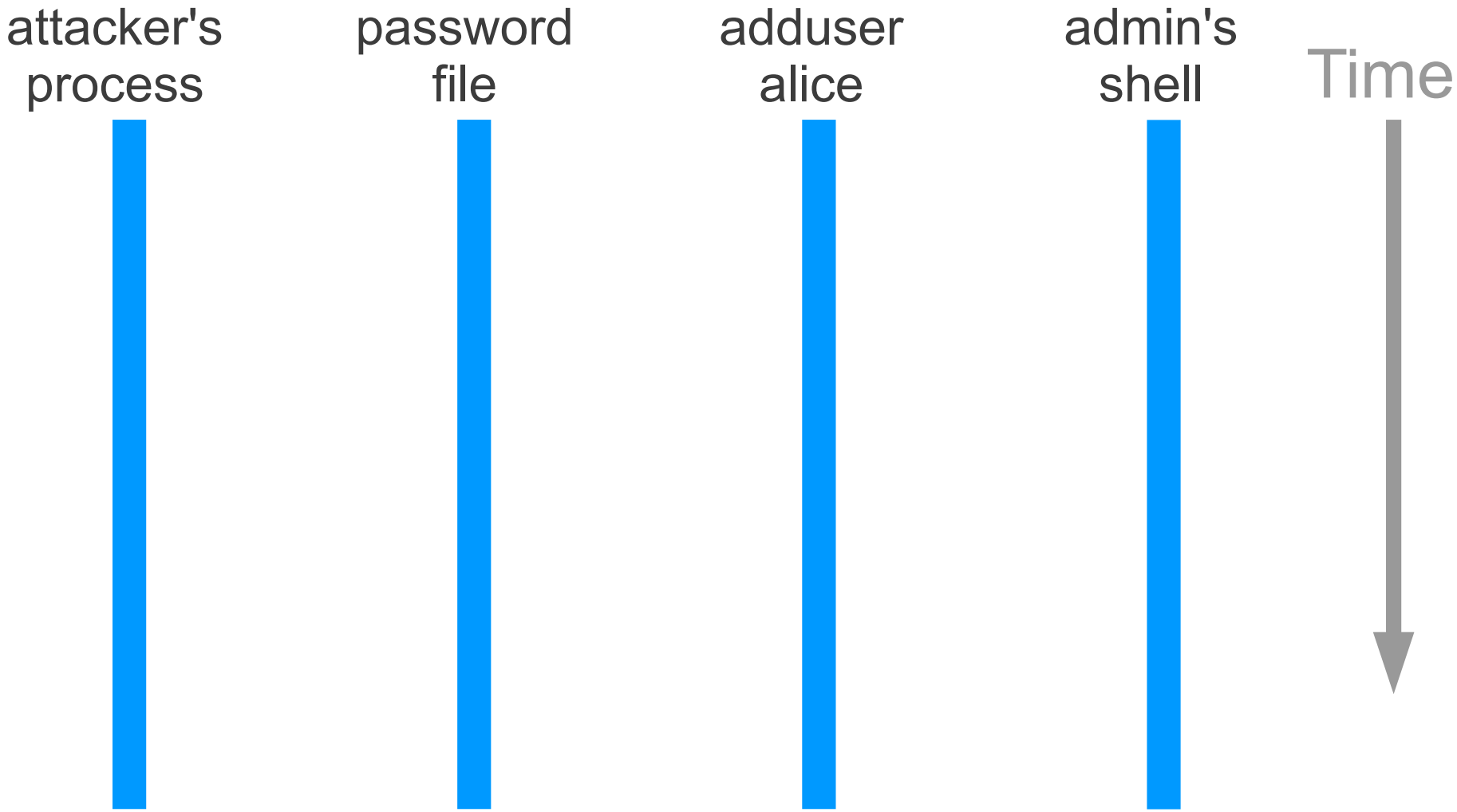


- May take one week to re-execute for a week-old attack
- Original VM inputs may be meaningless for new system
  - Non-determinism: new SSH crypto keys, inode #s, app state, …
  - Can't do deterministic re-execution, since some inputs changed

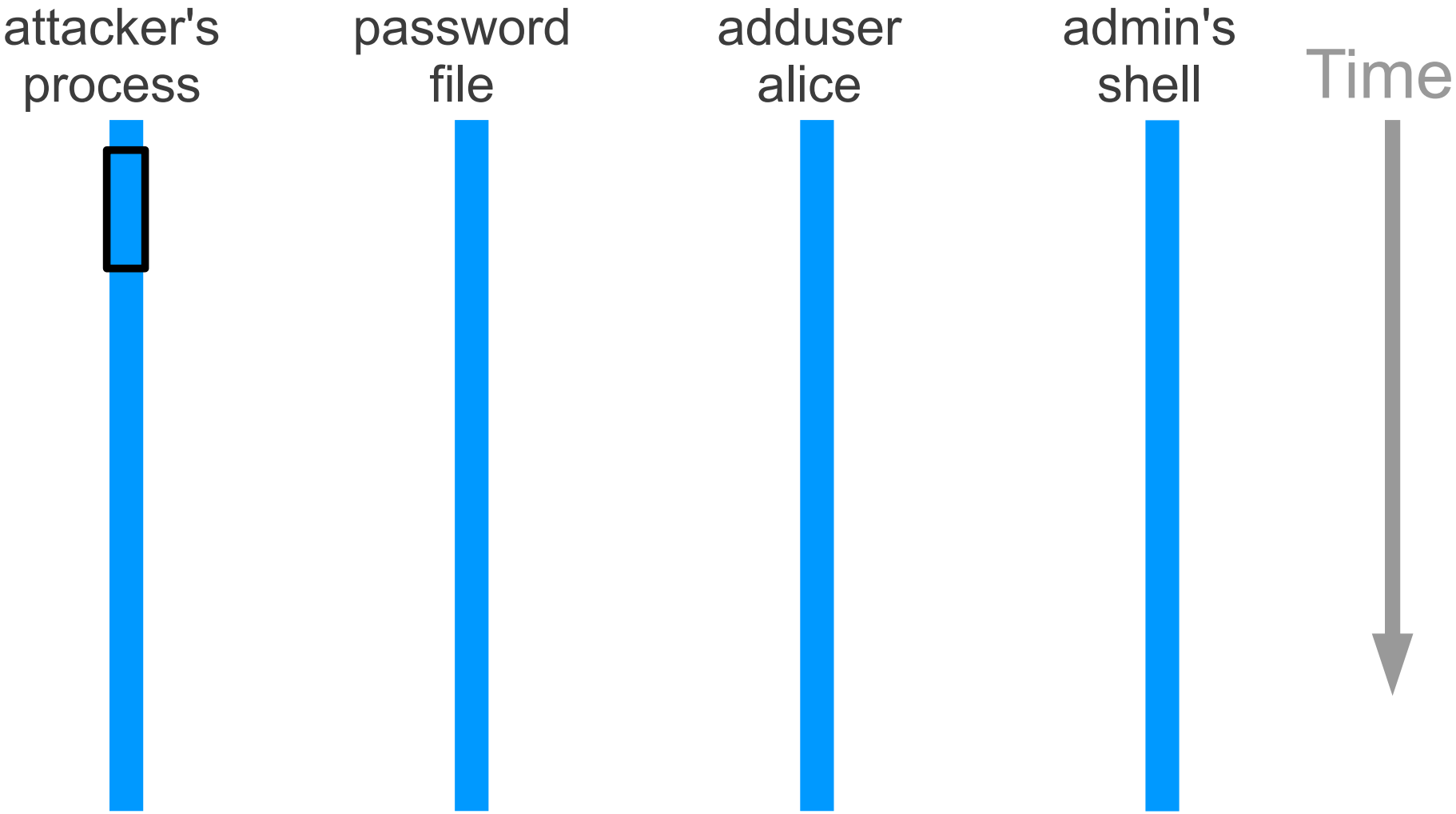
Retro's approach: selective re-execution

- Record fine-grained *action history graph*
  - Includes system call arguments, function calls, …
  - Assume tamper-proof kernel, storage
- *Roll back* objects directly affected by attack
  - Avoid the false positives of taint tracking
- *Re-execute* actions indirectly affected by attack
  - Avoid expense, non-determinism of whole-VM re-exec.

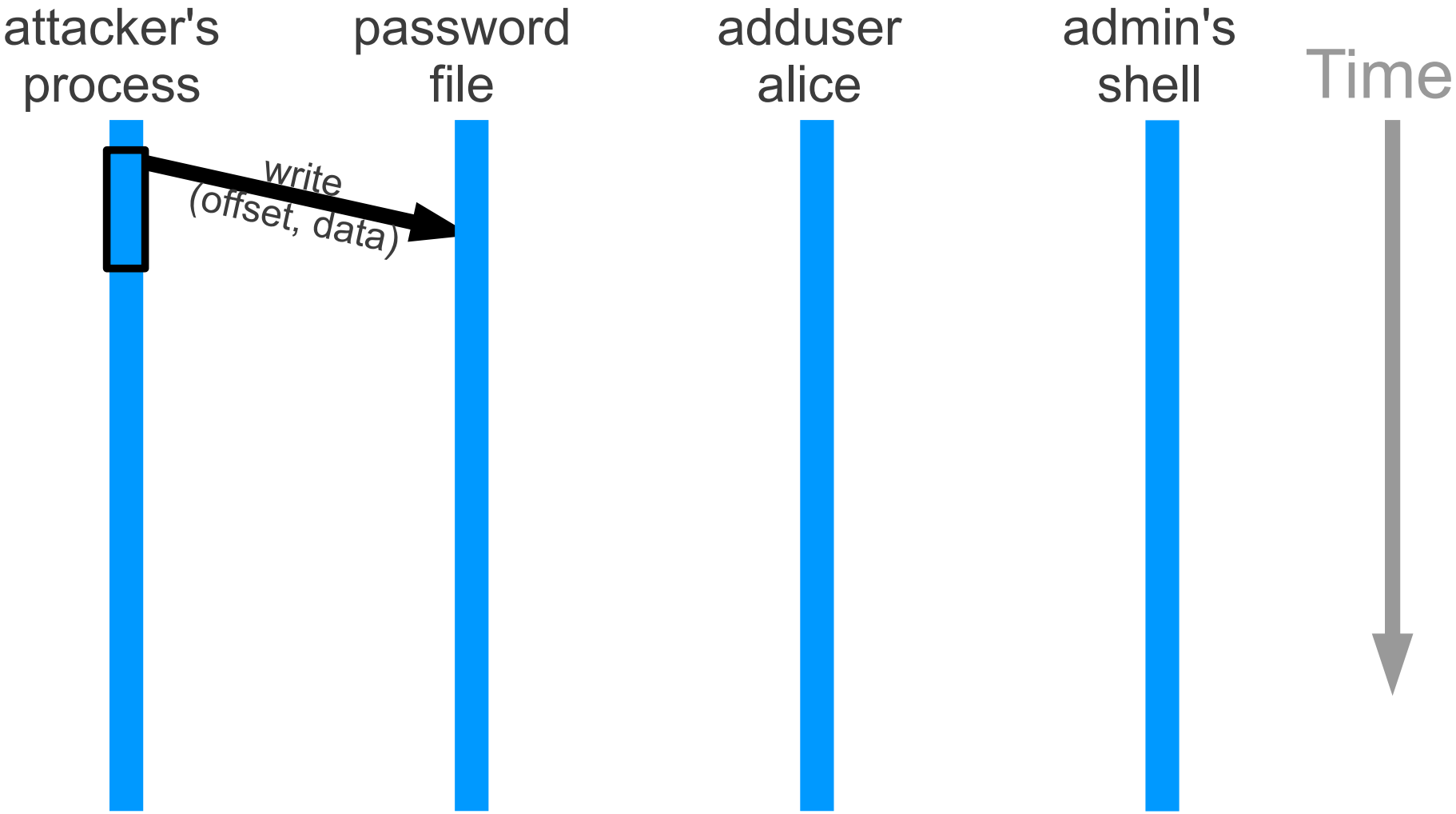
Action history graph: Objects represent files, processes



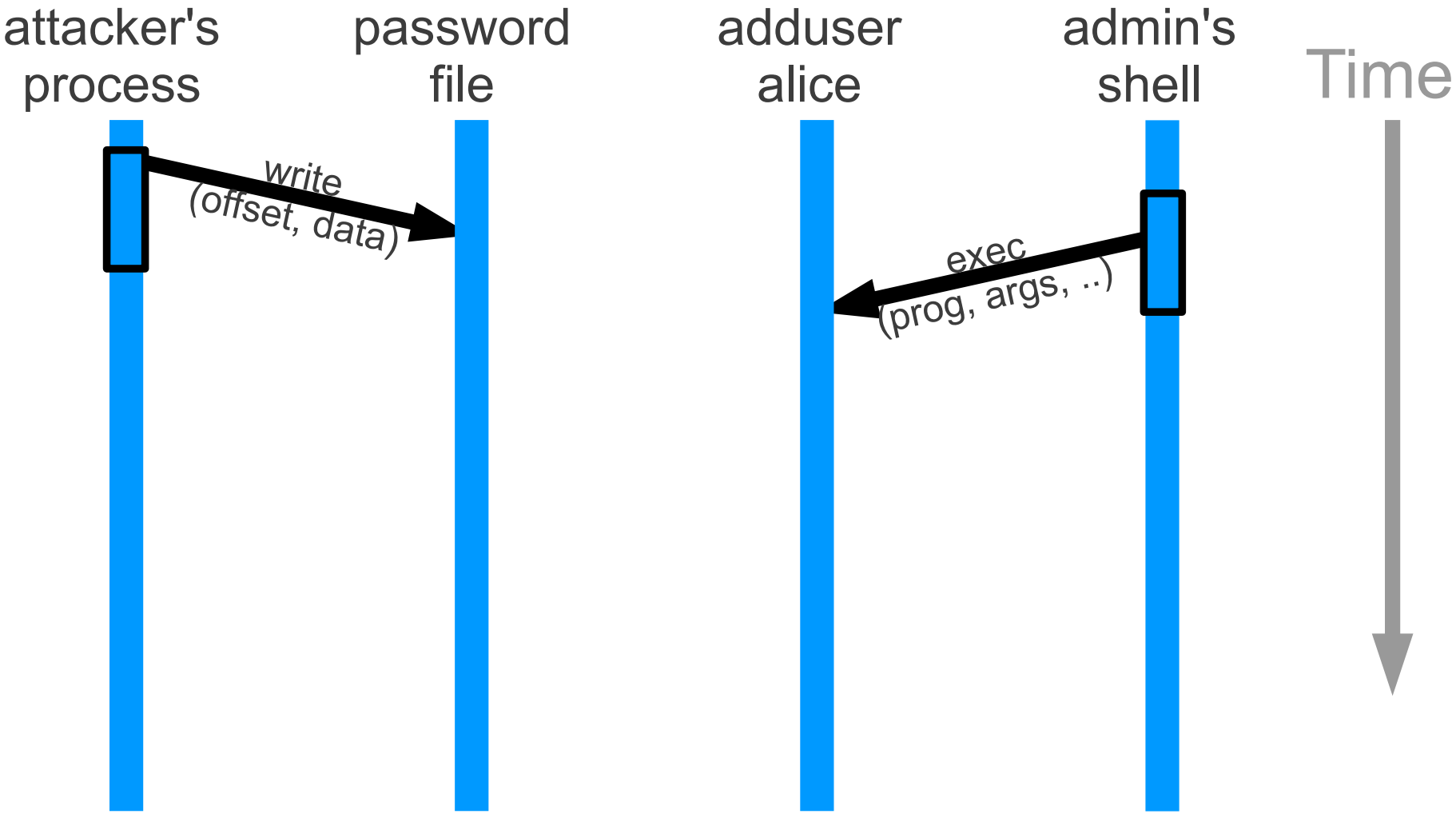
Action history graph: Actions represent execution



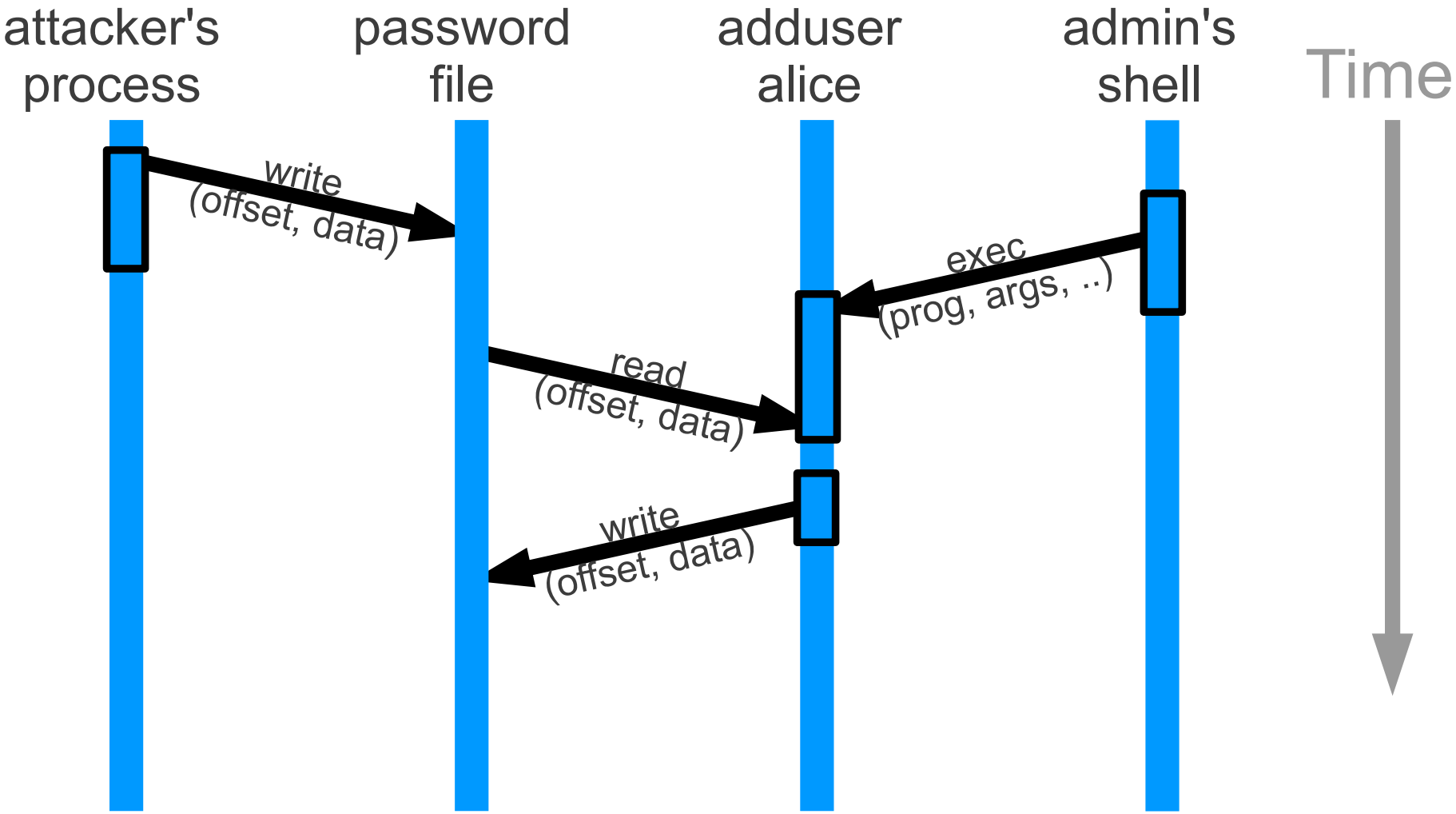
Action history graph: Actions have dependencies



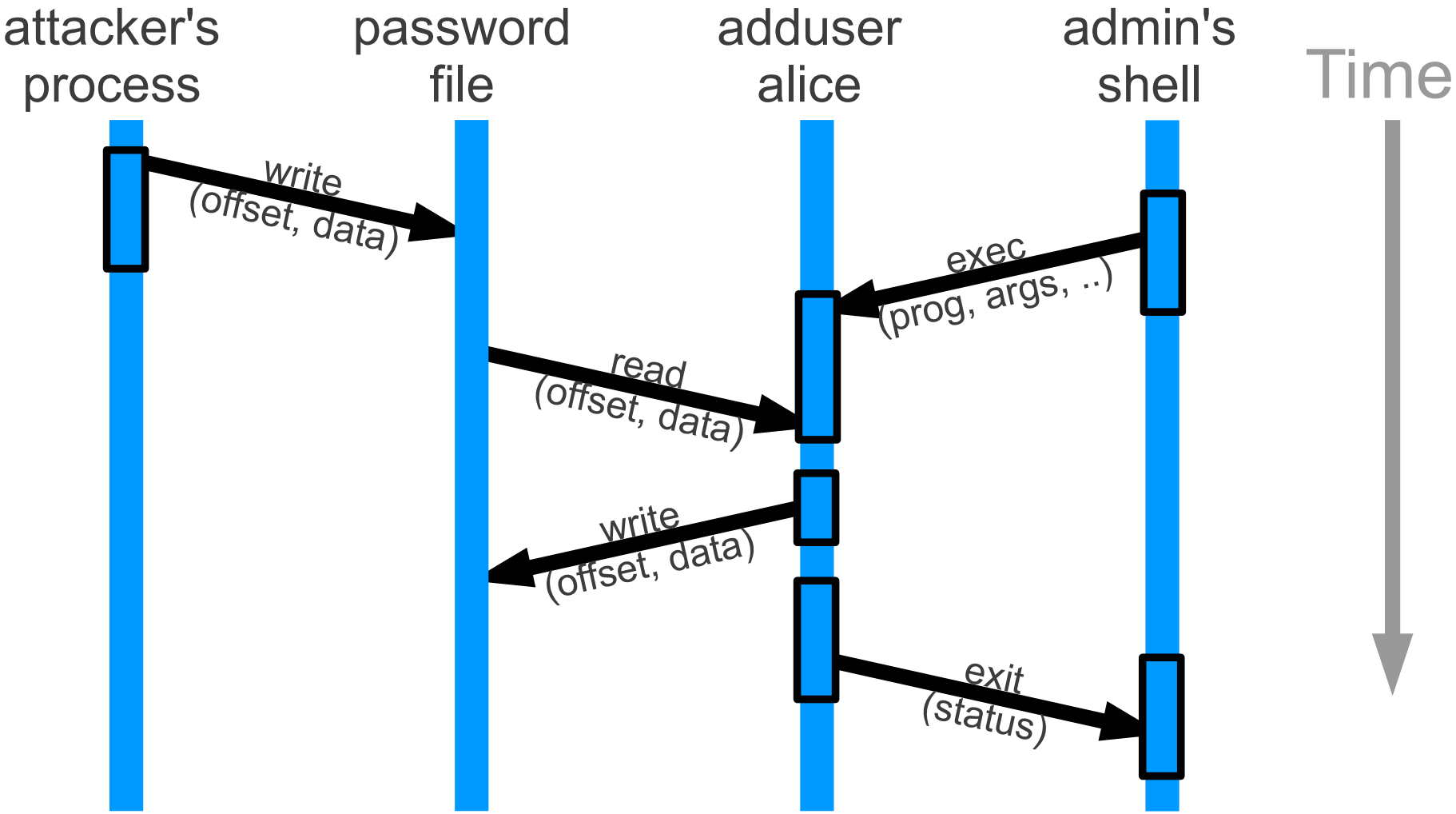
Action history graph: Actions have dependencies



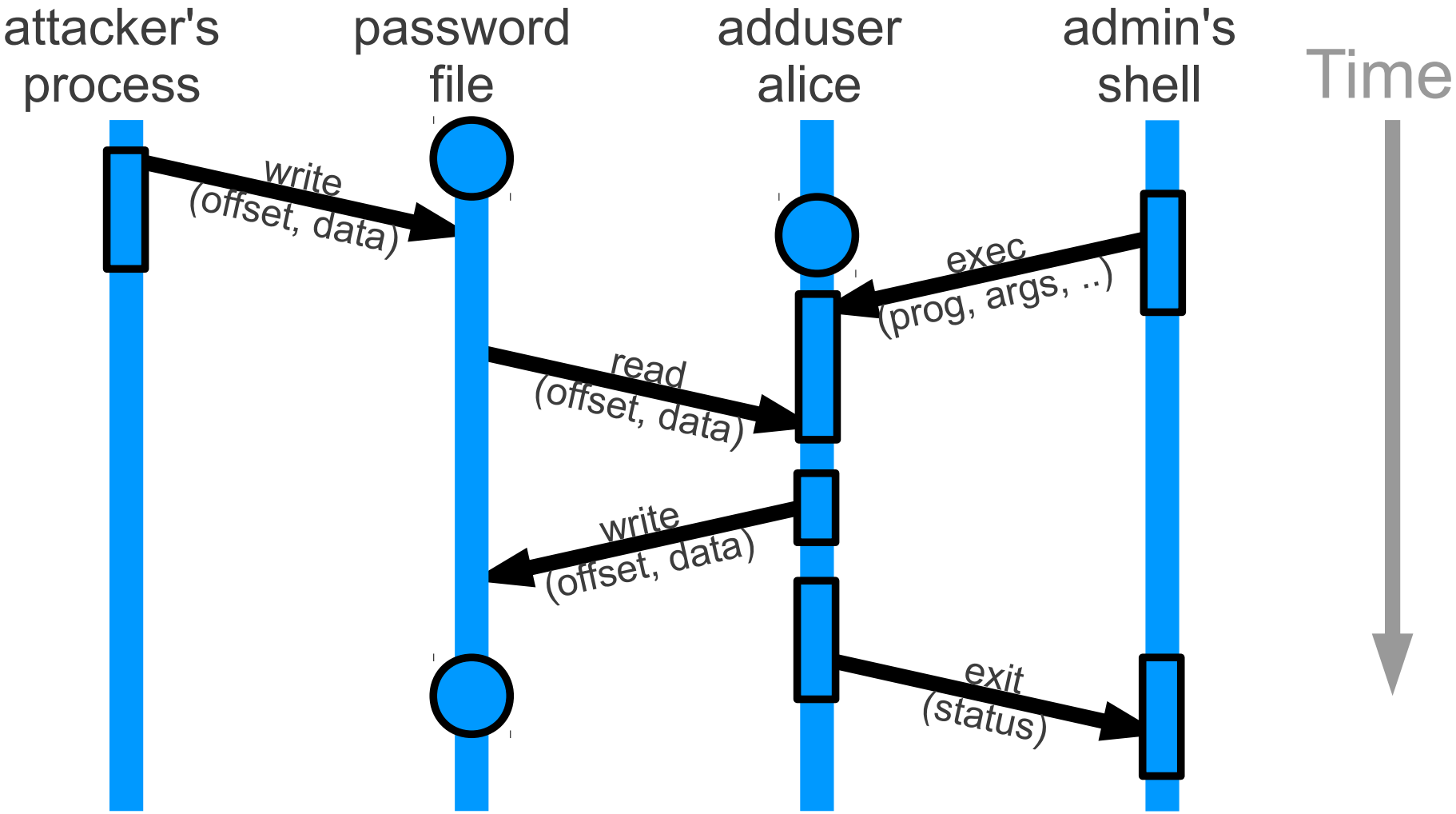
Action history graph: Actions have dependencies



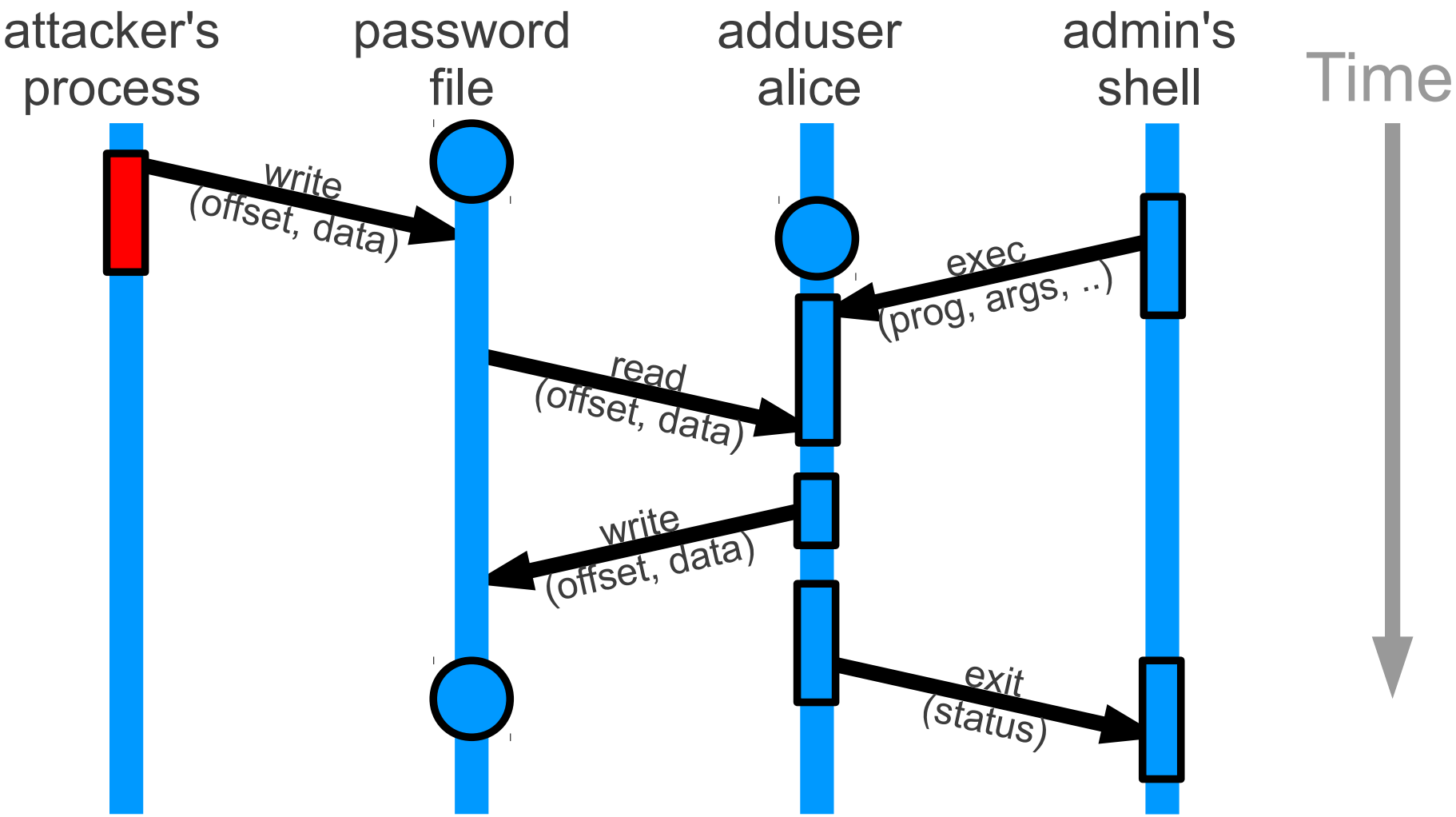
Action history graph: Actions have dependencies



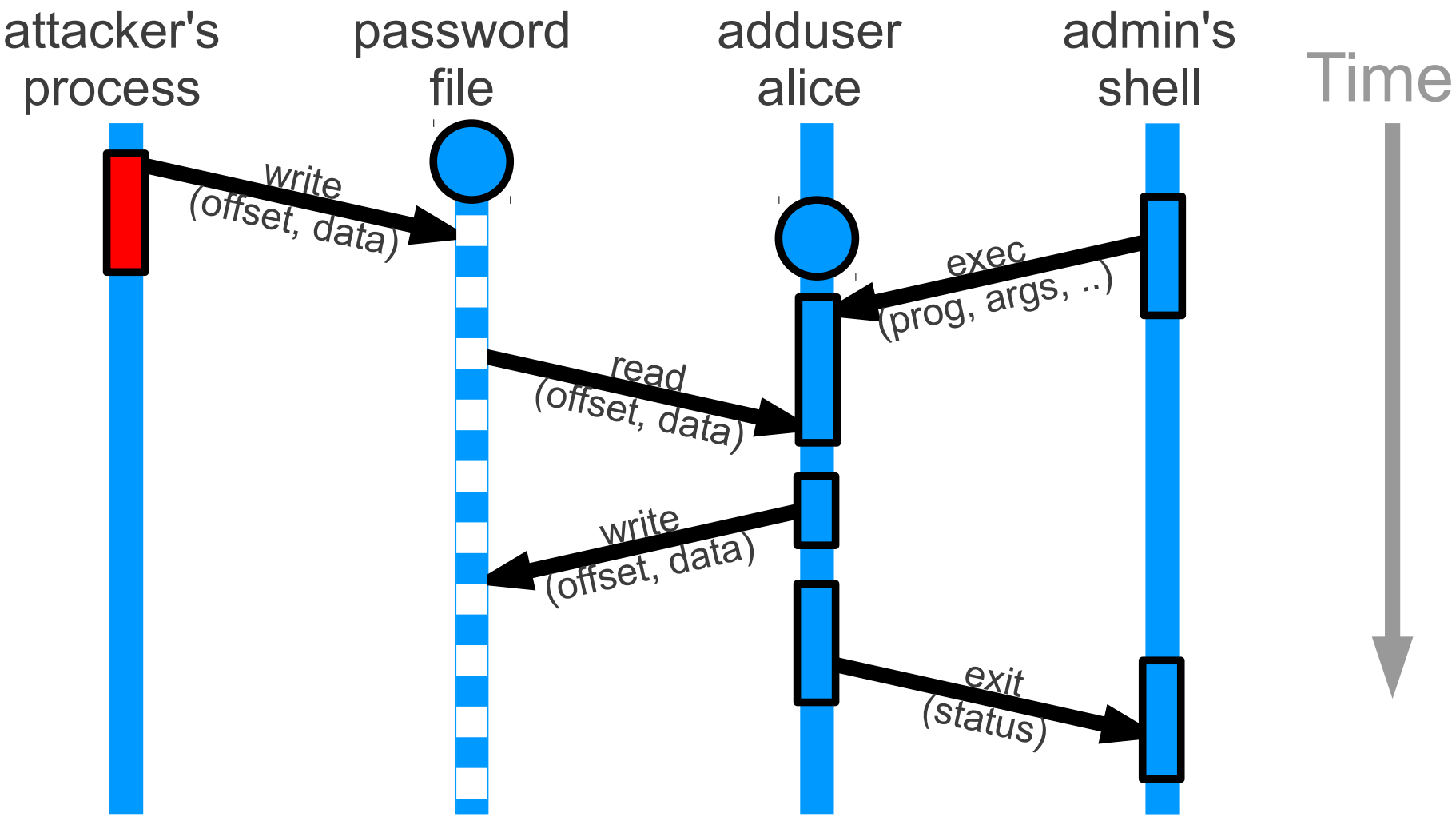
Action history graph: Objects have checkpoints



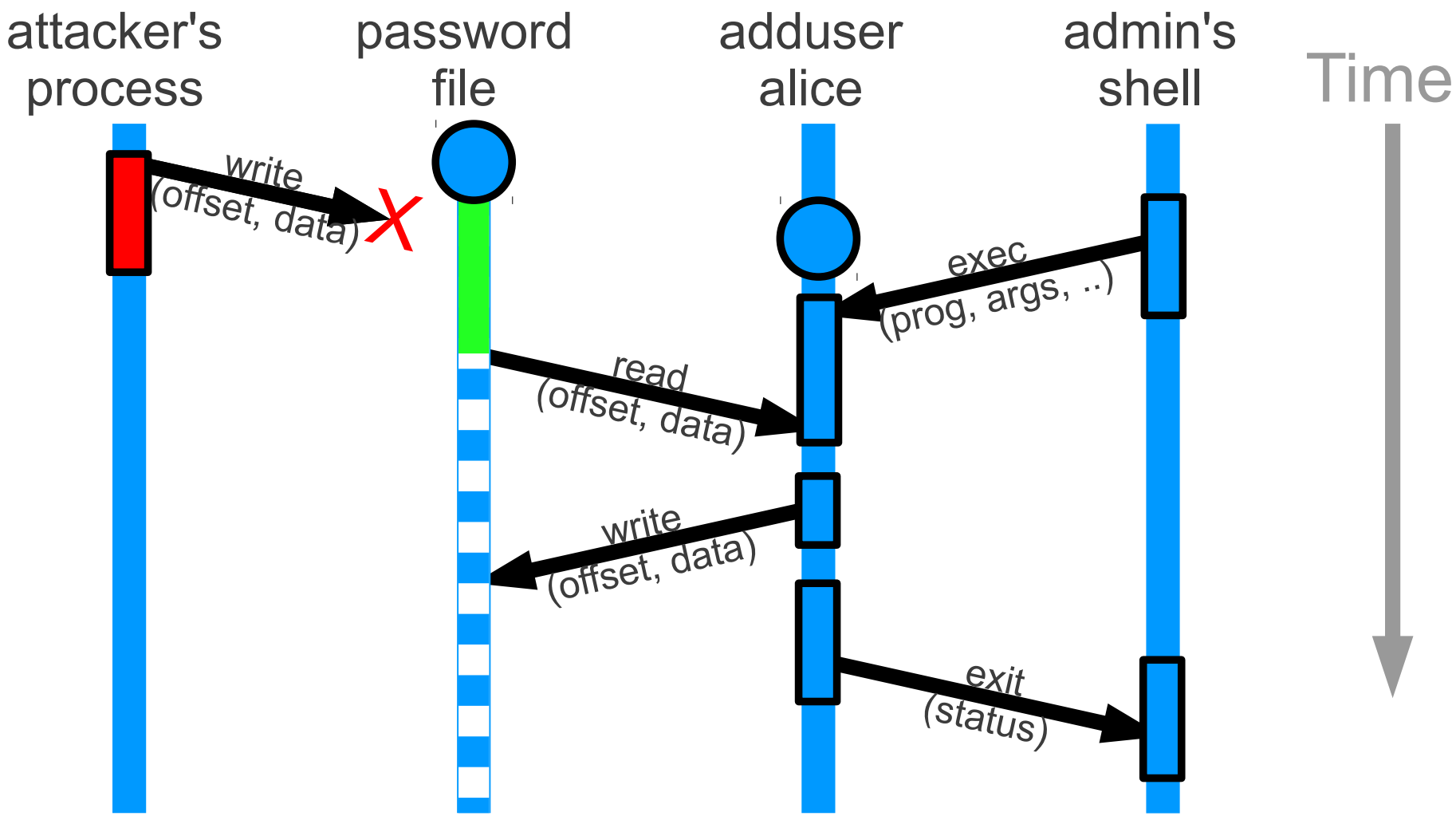
Step 1: find attack action



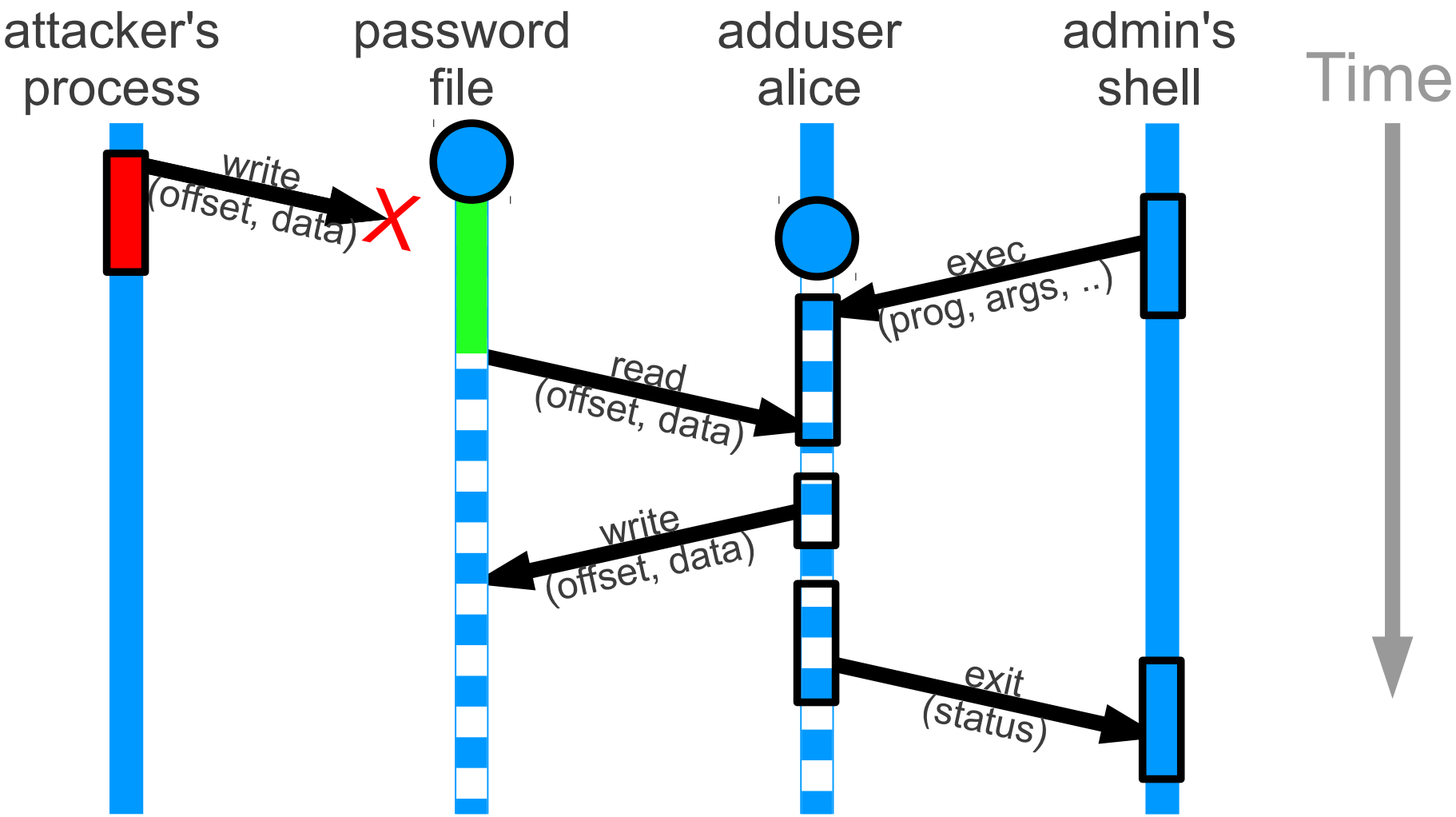
Step 2: roll back affected objects



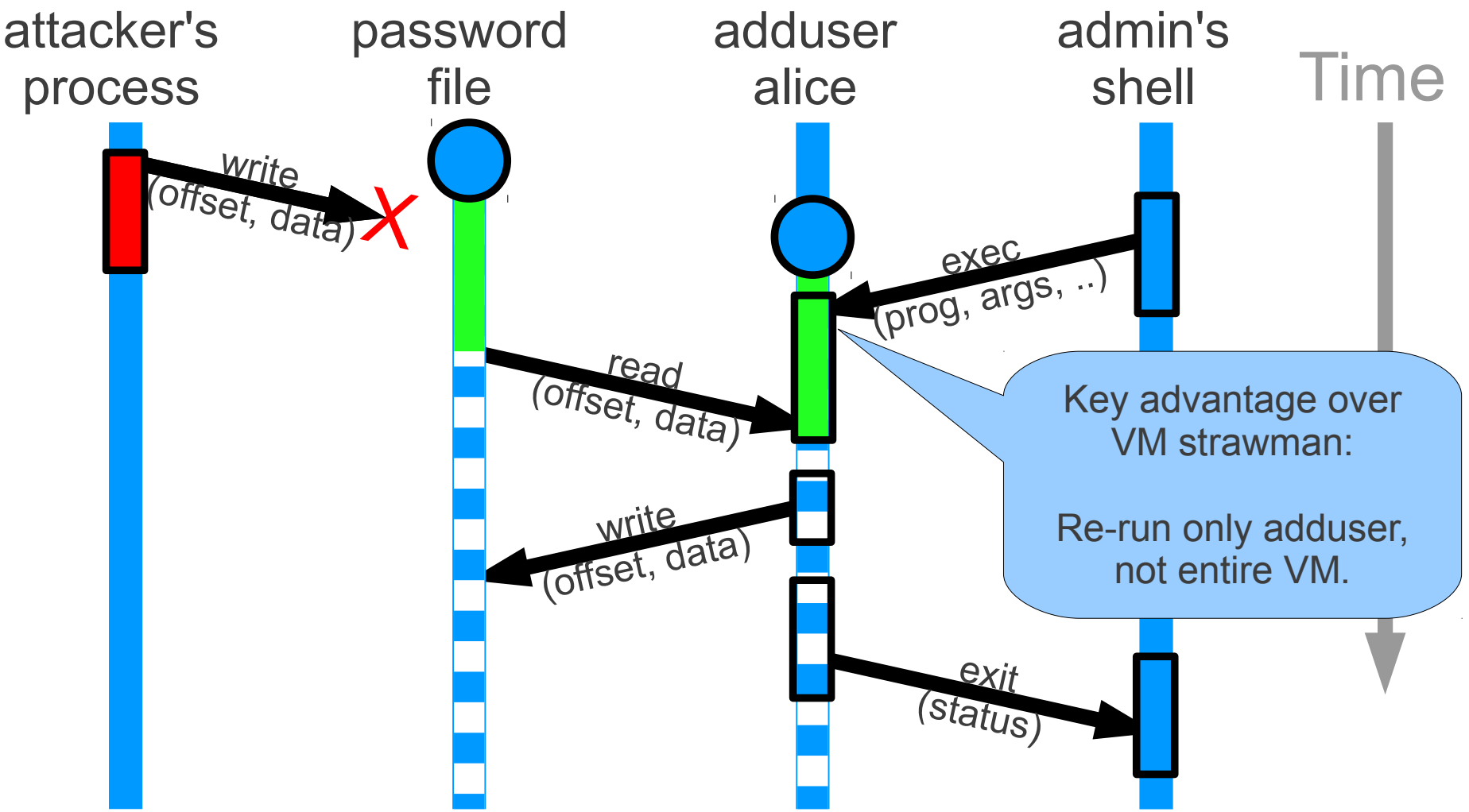
Step 3: redo non-attack actions



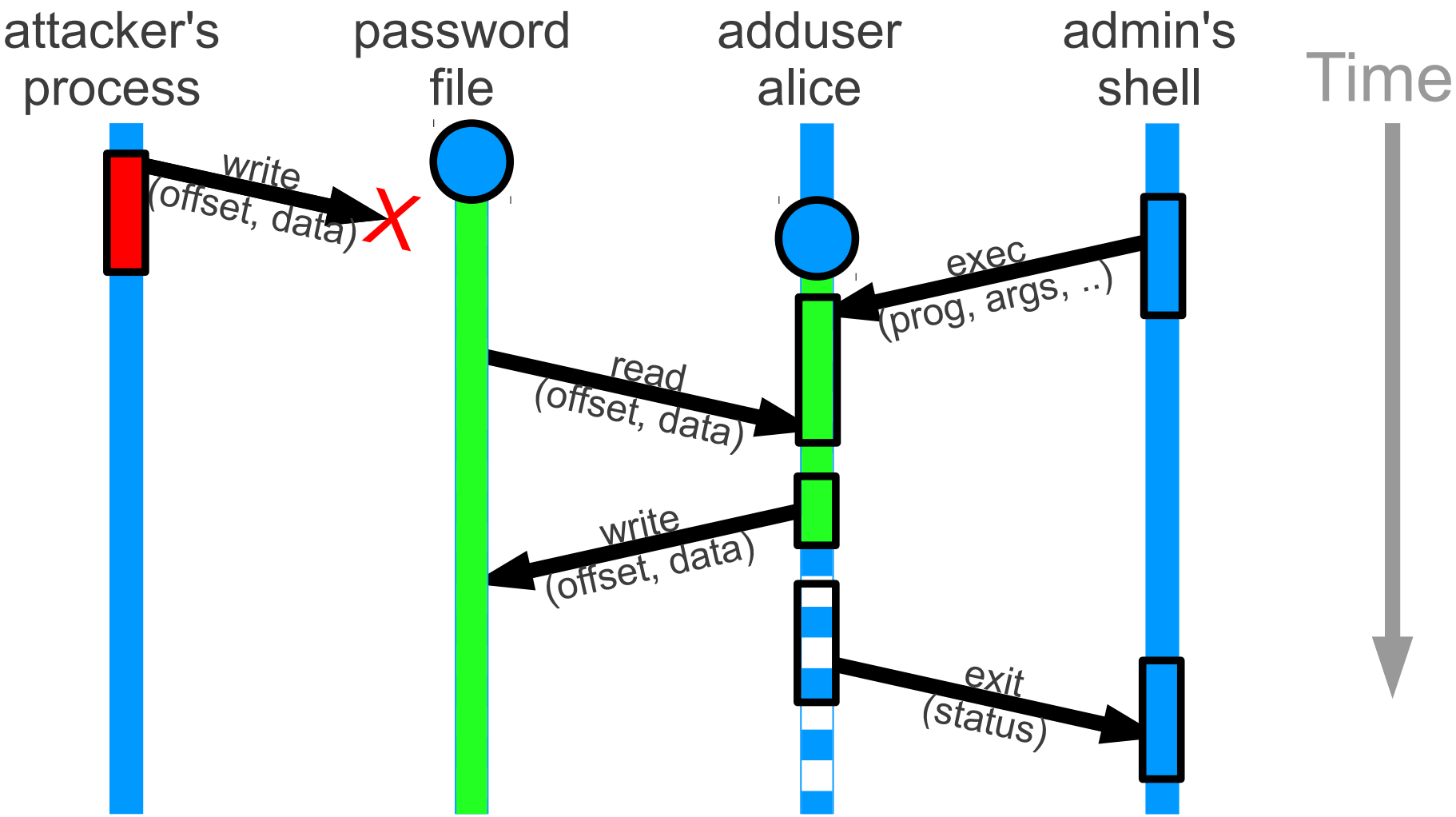
Repeat step 2: roll back objects



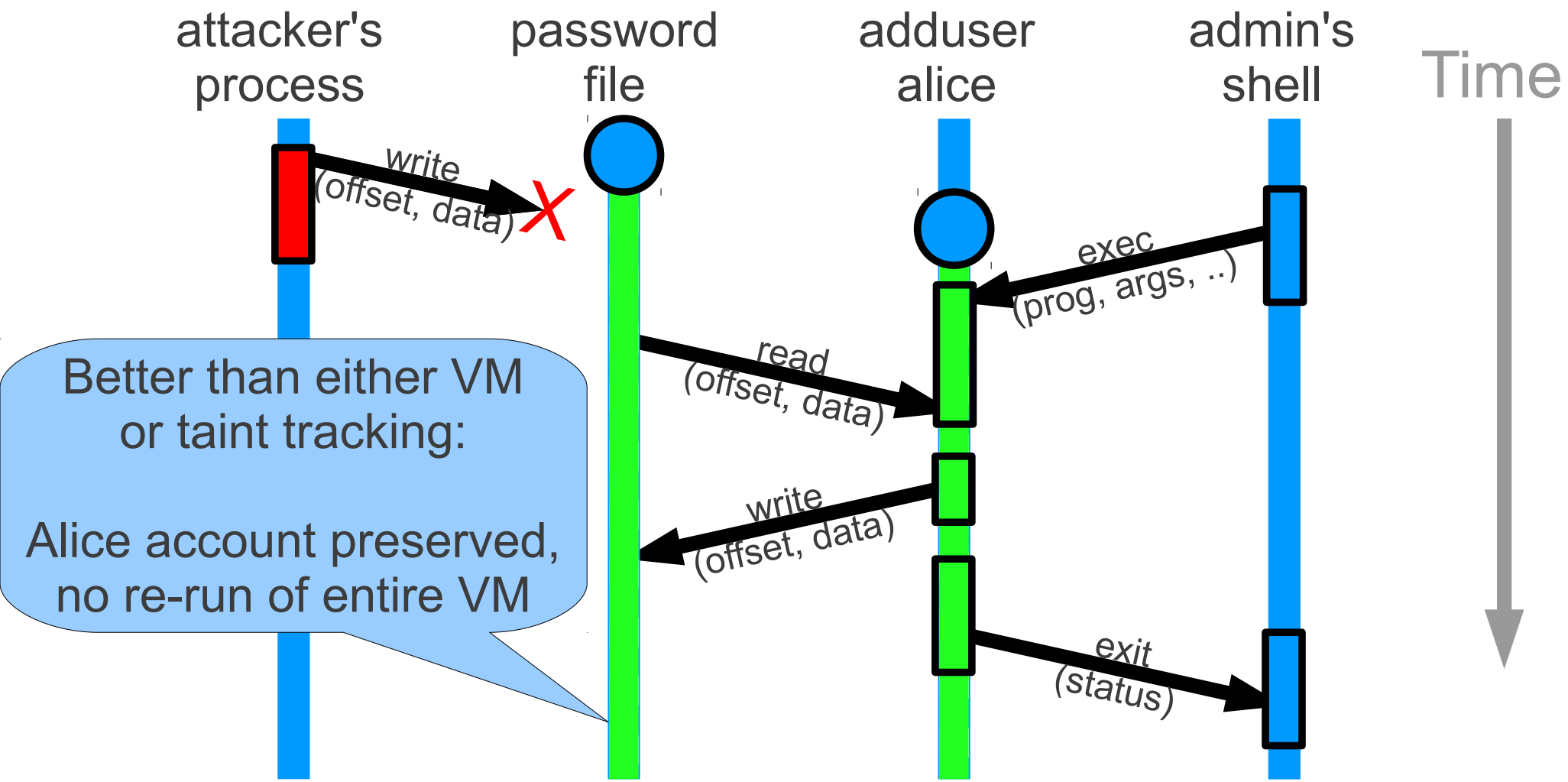
Repeat step 3: redo actions



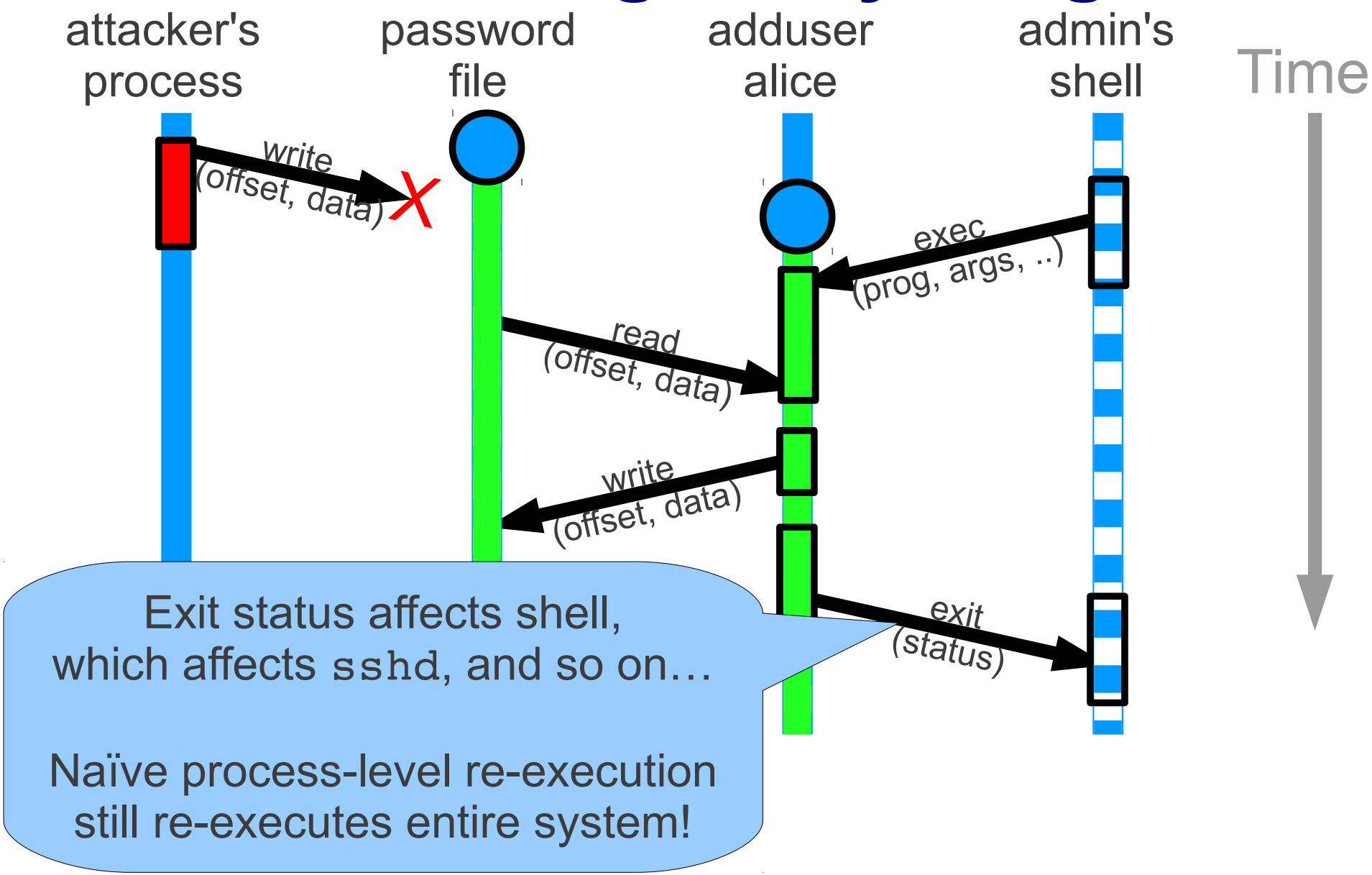
Repeat step 3: redo actions



Repeat step 3: redo actions



Challenge: how to avoid re-executing everything?



Observation: many suspect computations are not affected

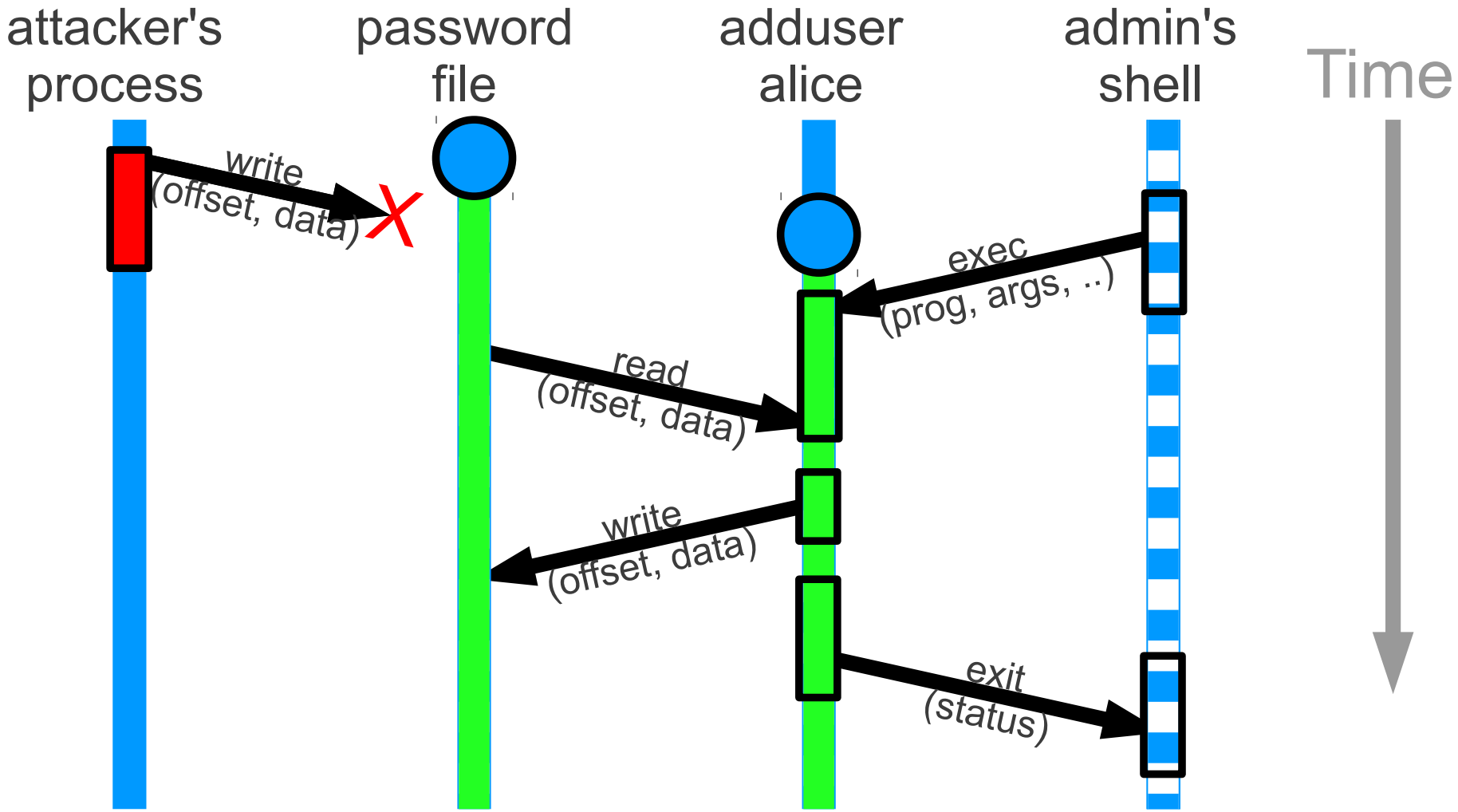
- Attacker adds 1 account to password file
  - Alice's `sshd` reads password file,
    but looks up Alice's account instead of attacker's
- Attacker adds 1 line to `pdflatex` to restart botnet
  - Alice's `pdflatex` process may restart botnet,
    but otherwise does legitimate work
- Significant changes → can detect attack earlier

Approach: minimize re-execution

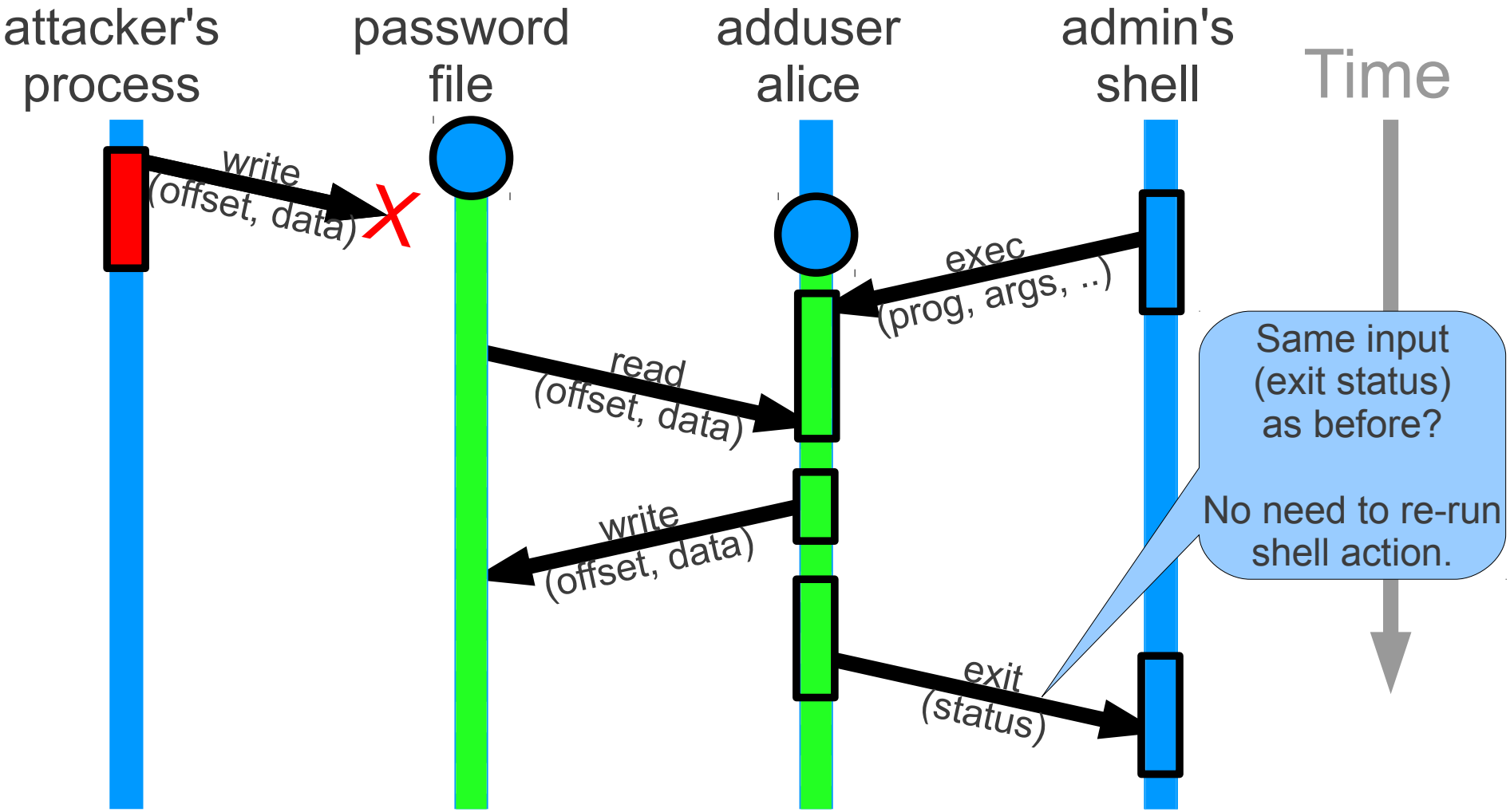
- ***Predicates:*** Retro skips equivalent computations
  - Predicate checks whether inputs are the same
  - If so, assume original result OK, avoid re-execution

- ***Refinement:*** Retro re-executes fine-grained actions
  - Avoid re-executing entire process or login session, when only a small part of it was affected

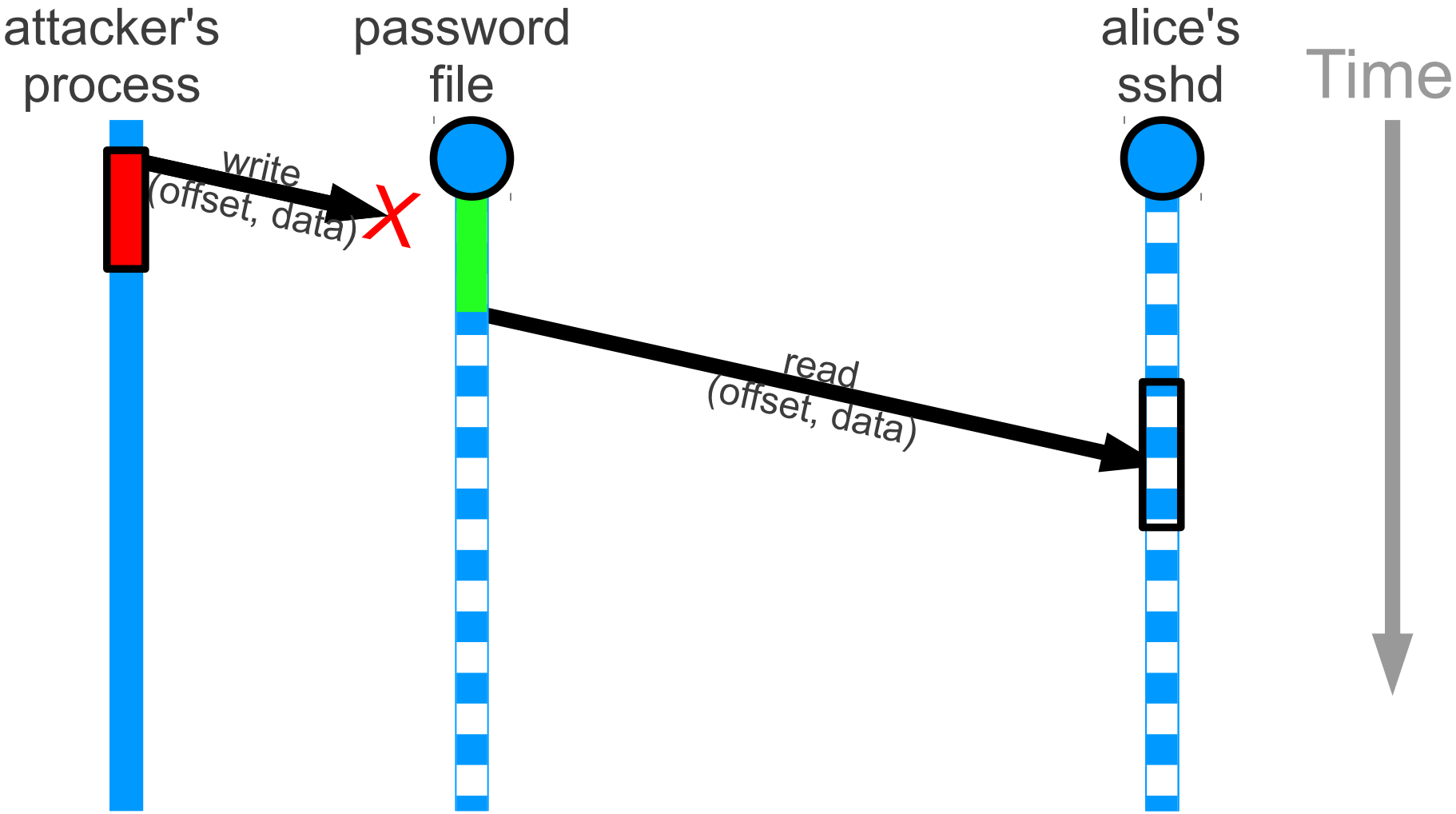
Example 1:
exit status to shell unchanged



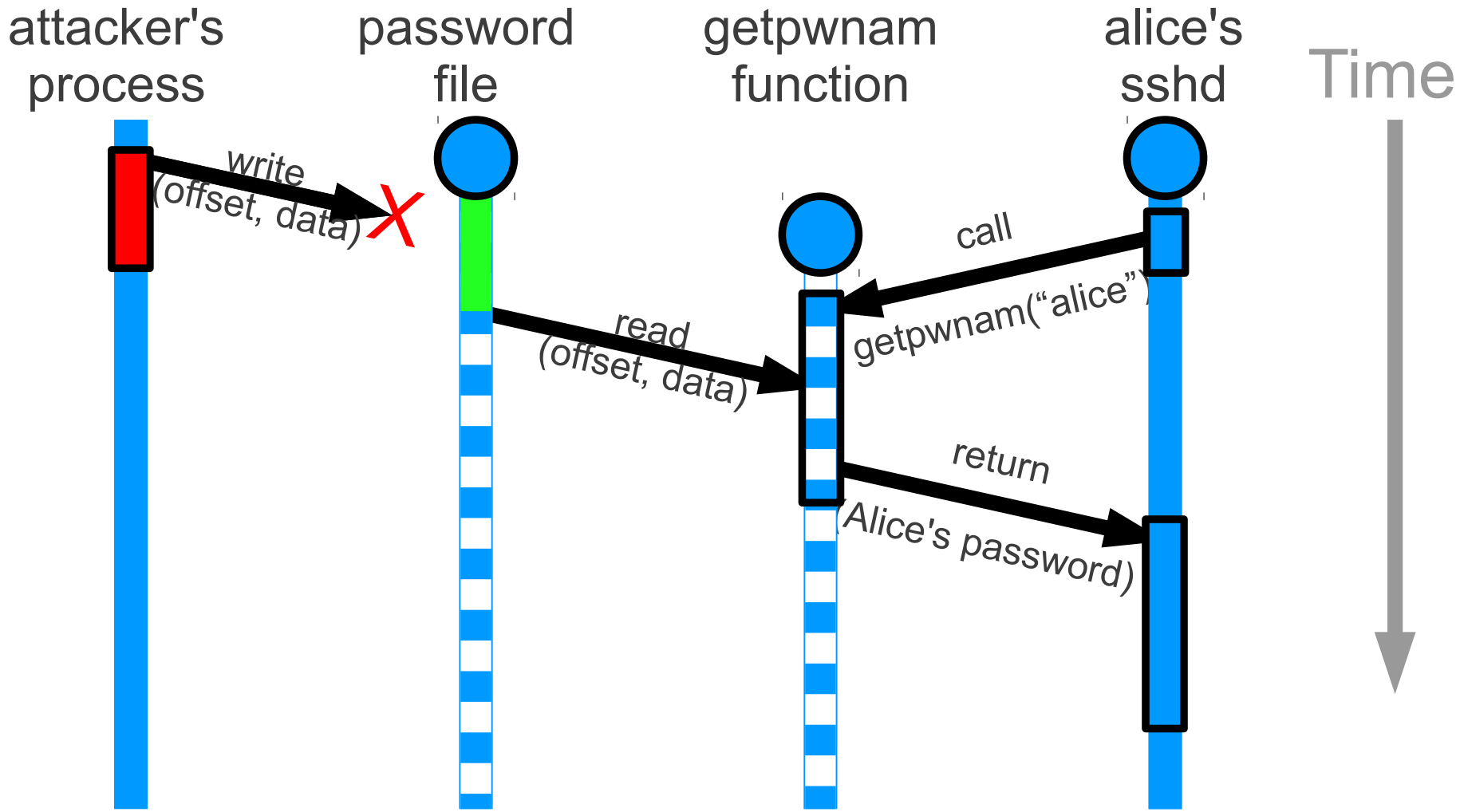
Predicates: avoid equivalent re-execution



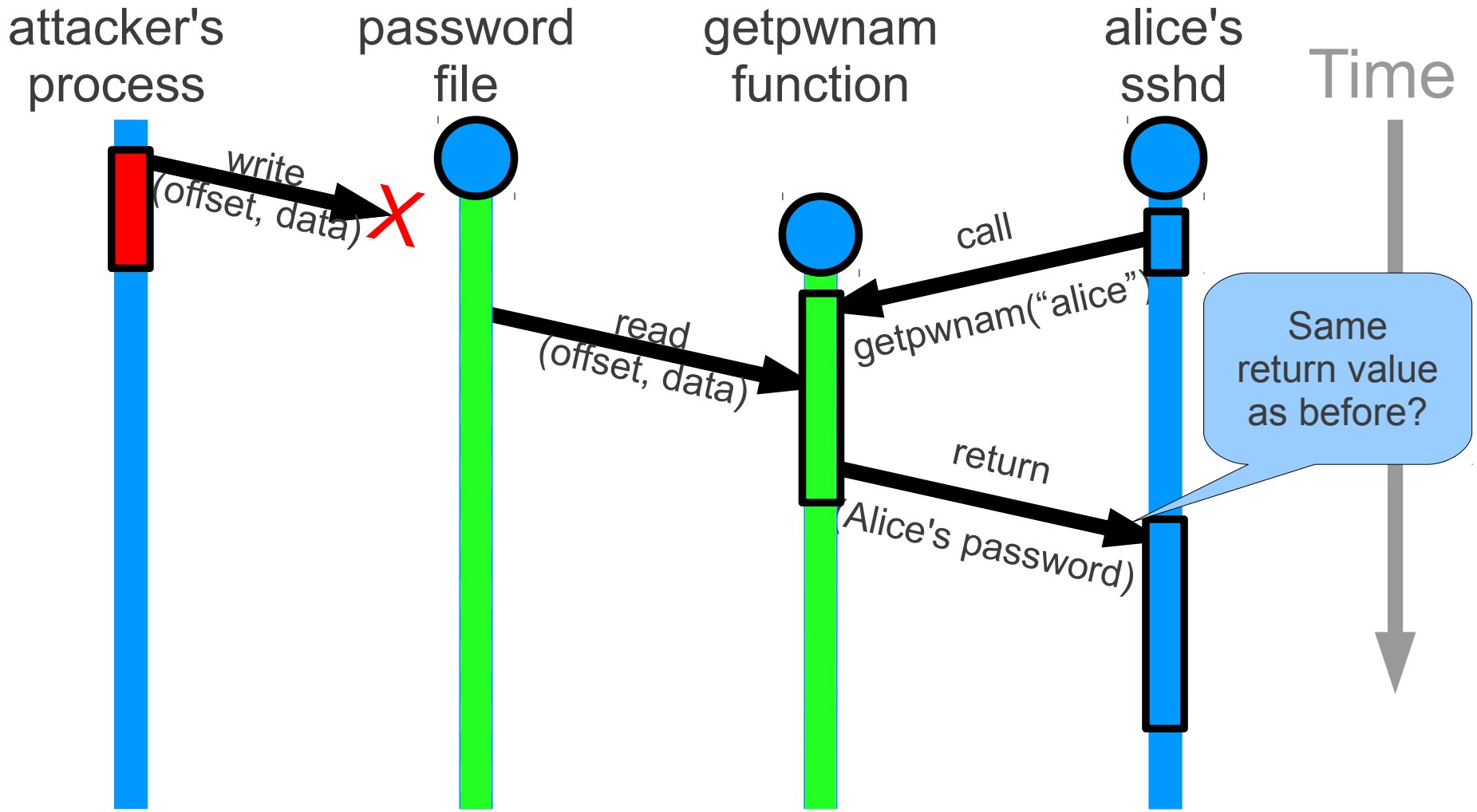
Example 2: user's password unchanged



Refinement: re-execute individual functions



Refinement: re-execute individual functions



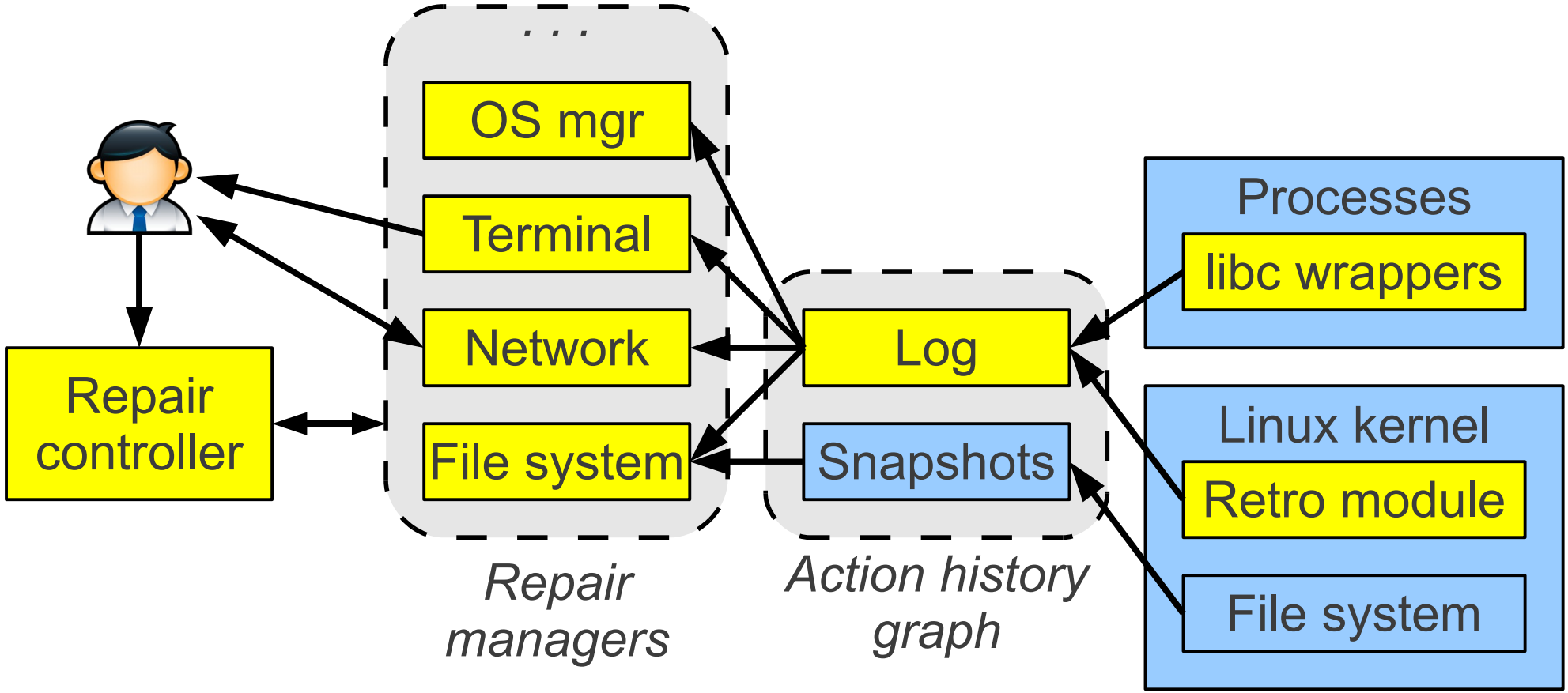
Remaining challenge: external dependencies

- What if the attack was externally-visible?
  - Attacker sent spam, or user saw wrong output from `ls`
- Cannot solve general case (spam already sent)
  - Will need to pause repair and ask for user input
- Can do compensating actions in some cases

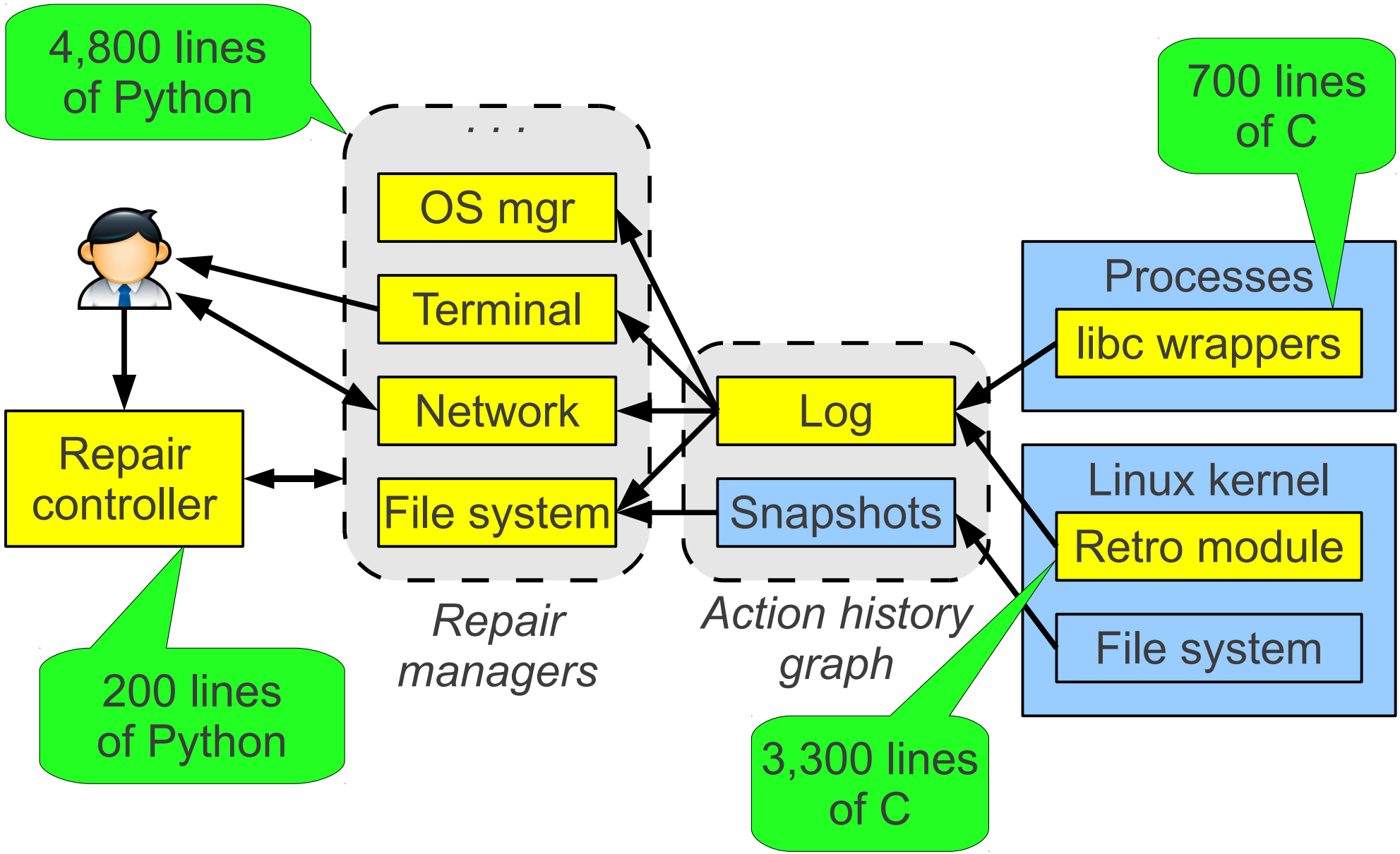
Compensating action for terminals: email diff to user

```
  nickolai@karakum:~$ cd undosys/libundo
  nickolai@karakum:~/undosys/libundo$ ls -l
  -rw-r--r-- 1 nickolai nickolai  493 2010-05-13 09:46 Makefile
- -rw-r--r-- 1 nickolai nickolai 2124 2010-05-13 10:22 attack.c
  drwxr-xr-x 2 nickolai nickolai 4096 2010-05-13 09:46 bdb
  -rwxr-xr-x 1 nickolai nickolai  973 2010-05-13 09:46 mailserver.py
  drwxr-xr-x 2 nickolai nickolai 4096 2010-05-13 09:46 php
  -rw-r--r-- 1 nickolai nickolai 5221 2010-05-13 09:46 pwd.c
  -rw-r--r-- 1 nickolai nickolai 1424 2010-05-13 09:46 undo.py
+ -rw-r--r-- 1 nickolai nickolai  662 2010-05-13 09:46 undocall.c
+ -rw-r--r-- 1 nickolai nickolai 1340 2010-05-13 09:46 undocall.h
+ -rw-r--r-- 1 nickolai nickolai  755 2010-05-13 09:46 undotest.c
+ -rwxr-xr-x 1 nickolai nickolai  360 2010-05-13 09:46 undotest.py
  -rw-r--r-- 1 nickolai nickolai 6603 2010-05-13 09:46 undowrap.c
  nickolai@karakum:~/undosys/libundo$ du -ks .
- 84      .
+ 96      .
  nickolai@karakum:~/undosys/libundo$ cd ..
  nickolai@karakum:~/undosys$
```

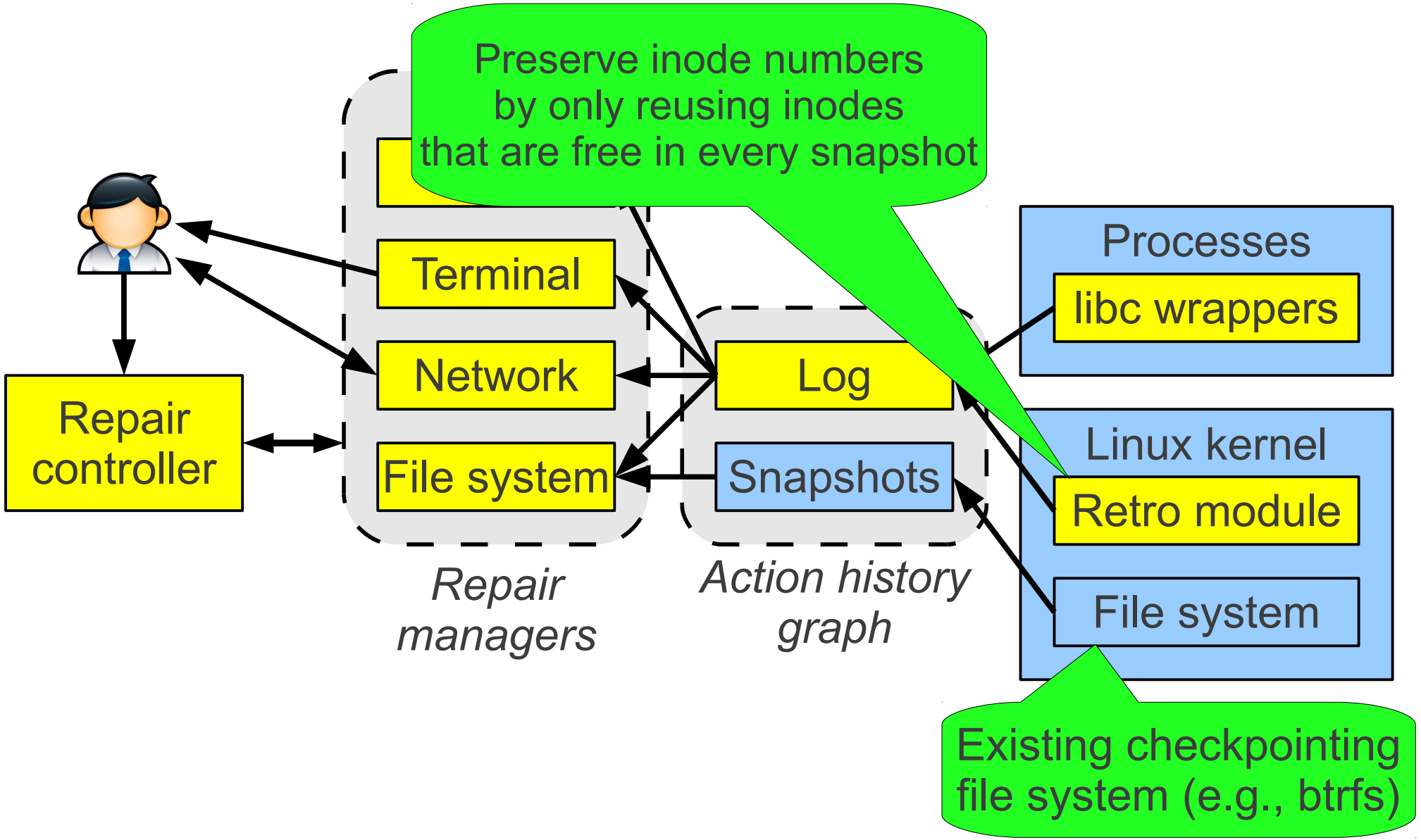
Retro implementation



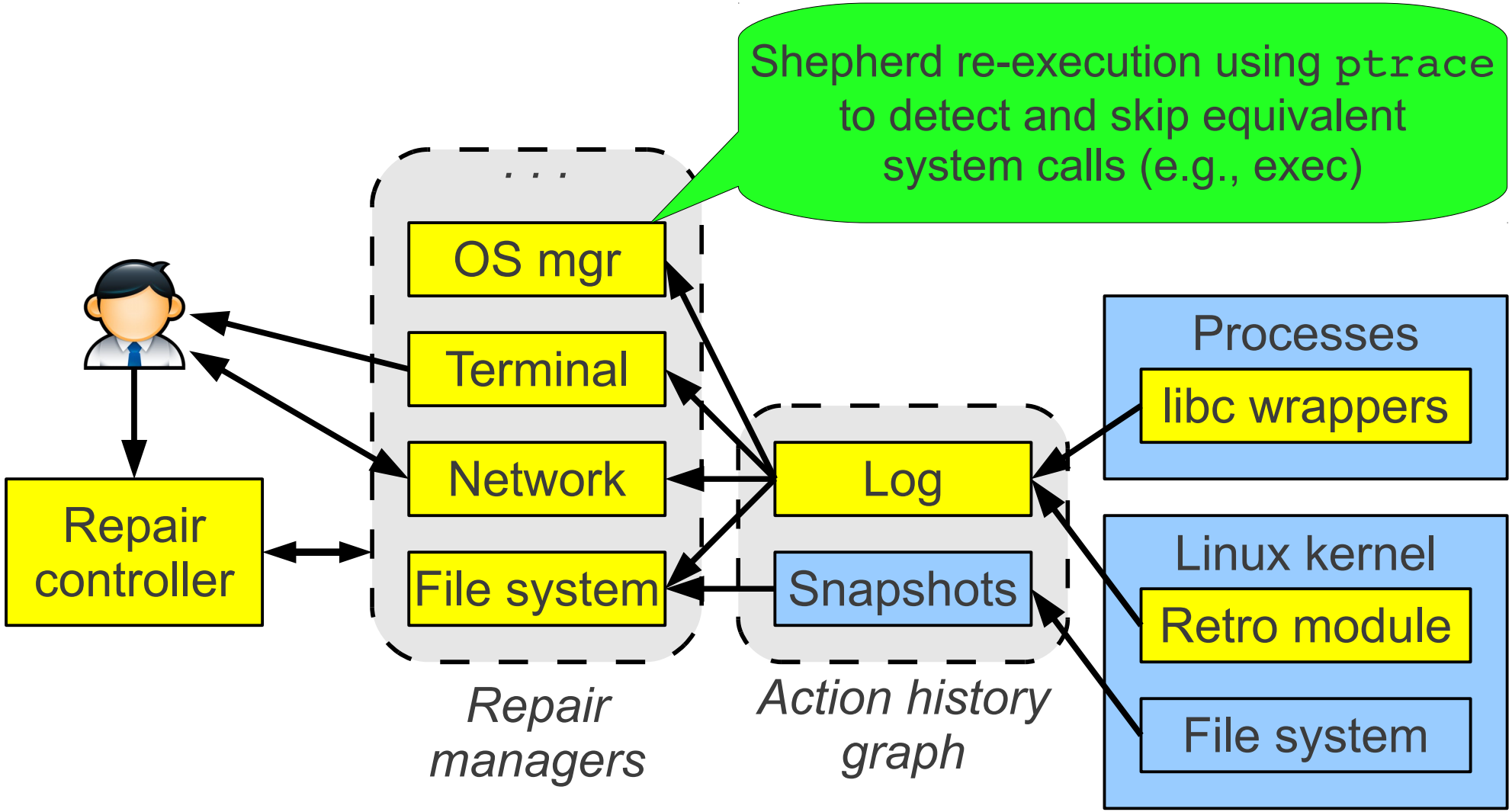
Retro implementation



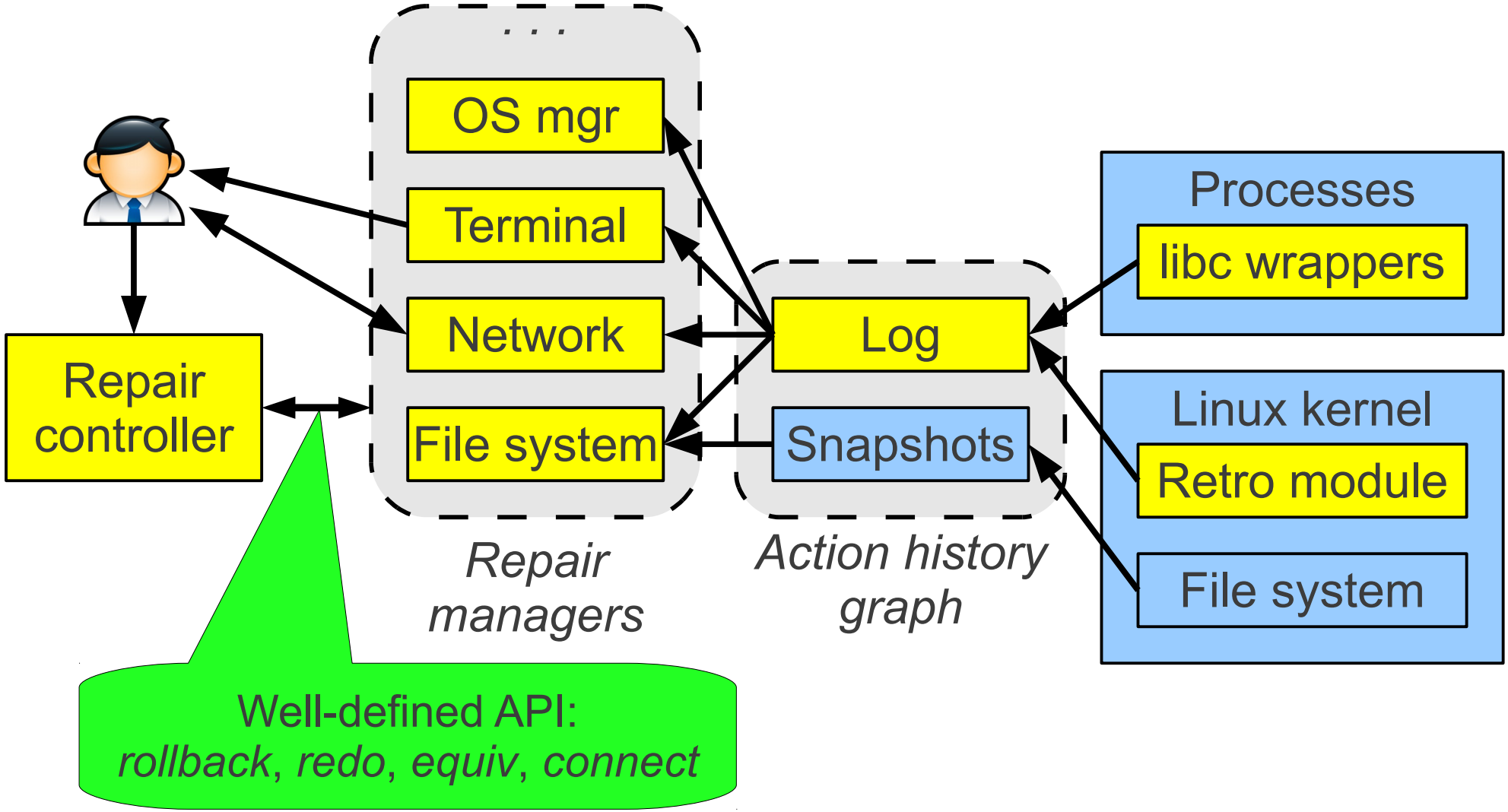
Retro implementation



Retro implementation



Retro implementation



Evaluation questions

- How much better is Retro than manual repair?
- What is Retro's cost during normal execution?

Evaluation setup

- 2 real-world attacks from honeypot
  - Remove log entries, add accounts, run botnet
- 2 synthetic challenge attacks
  - Running example (LaTeX trojan) and `sshd` trojan
- 6 attacks from *Taser* recovery system [Goel'05]
  - File sharing, web servers, databases, desktop apps
  - Website backdoors, trojans in `ls`, new accounts

Retro repairs from all attacks

Attack	Retro	User input required
Root pw change	✓	Skip attacker's login attempt
Log cleaning	✓	–
LaTeX trojan	✓	–
sshd trojan	✓	Packet replay req'd – conflict!
Illegal storage	✓	–
Content destruct.	✓	– (generates terminal diff)
Unhappy student	✓	– (generates terminal diff)
Compromised DB	✓	–
Browser plugin	✓	Skip re-execution of browser
Weak password	✓	Skip attacker's login attempt

Retro repairs from all attacks

Attack	Retro	User input required
Root pw change	✓	Skip attacker's login attempt
Log cleaning	✓	–
LaTeX trojan	✓	–
sshd trojan	✓	Packet replay req'd – conflict!
Illegal storage	✓	–
Content destruct.	✓	– (generates terminal diff)
Unhappy student	✓	– (generates terminal diff)
Compromised DB	✓	–
Browser plugin	✓	Skip re-execution of browser
Weak password	✓	Skip attacker's login attempt

6/10 cases: no user input needed, automatic re-execution suffices

Attack	Retro	User input required
Root pw change	✓	Skip attacker's login attempt
Log cleaning	✓	–
LaTeX trojan	✓	–
sshd trojan	✓	Packet replay req'd – conflict!
Illegal storage	✓	–
Content destruct.	✓	– (generates terminal diff)
Unhappy student	✓	– (generates terminal diff)
Compromised DB	✓	–
Browser plugin	✓	Skip re-execution of browser
Weak password	✓	Skip attacker's login attempt

2/10 cases: user input needed to skip attacker's SSH logins

Attack	Retro	User input required
Root pw change	✓	Skip attacker's login attempt
Log cleaning	✓	–
LaTeX trojan	✓	–
sshd trojan	✓	Packet replay req'd – conflict!
Illegal storage	✓	–
Content destruct.	✓	– (generates terminal diff)
Unhappy student	✓	– (generates terminal diff)
Compromised DB	✓	–
Browser plugin	✓	Skip re-execution of browser
Weak password	✓	Skip attacker's login attempt

2/10 cases: user input needed to handle legitimate network I/O

Attack	Retro	User input required
Root pw change	✓	Skip attacker's login attempt
Log cleaning	✓	–
LaTeX trojan	✓	–
sshd trojan	✓	Packet replay req'd – conflict!
Illegal storage	✓	–
Content destruct.	✓	– (generates terminal diff)
Unhappy student	✓	– (generates terminal diff)
Compromised DB	✓	–
Browser plugin	✓	Skip re-execution of browser
Weak password	✓	Skip attacker's login attempt

Repair cost: *Retro* repairs few objects

Attack	Objects repaired by Retro
Root pw change	**7 (0.5%)**
Log cleaning	**99 (8%)**
LaTeX trojan	**190 (15%)**
sshd trojan	**880 (70%)**

Repair cost: *Retro* repairs few objects

Attack	Objects repaired by Retro
Root pw change	7 (0.5%)
Log cleaning	99 (8%)
LaTeX trojan	190 (15%)
sshd trojan	880 (70%)

- Repair cost proportional to extent of attack

Repair time depends largely on # objects, not log size

Total size of Retro log (action history graph)	Repair time for 136 objects / 399 syscalls
399 system calls	**0.3 seconds**
5,699,149 system calls	**4.7 seconds**

Repair time depends largely on # objects, not log size

Total size of Retro log (action history graph)	Repair time for 136 objects / 399 syscalls
399 system calls	**0.3 seconds**
5,699,149 system calls	**4.7 seconds**

- 10,000X increase in workload leads to 10X increase in repair time
- Much more efficient than whole-VM re-execution

Runtime overheads

Workload	CPU cost	Storage overhead
HotCRP conference web site	35%	4GB / day

Runtime overheads

Workload	CPU cost	Storage overhead
HotCRP conference web site	35%	4GB / day
Apache, small static files	127%	100GB / day
Continuous kernel recompile	89%	150GB / day

- Can store 2 weeks of logs on 2TB disk ($100) even for worst-case extreme workloads

Runtime overheads

Workload	CPU cost	w/ 2nd core	Storage overhead
HotCRP conference web site	35%	2%	4GB / day
Apache, small static files	127%	33%	100GB / day
Continuous kernel recompile	89%	18%	150GB / day

- Can store 2 weeks of logs on 2TB disk ($100) even for worst-case extreme workloads
- Can off-load CPU overhead to extra core

Related work

- Tracking down intrusions
  - **BackTracker** [*King'03*], **IntroVirt** [*Joshi'05*]
- Taint tracking to find, revert affected files
  - **Taser** [*Goel'05*], **Polygraph** [*Mahajan'09*]
- Selective undo and re-execution
  - **Undoable mail store** [*Brown'03*]
    (fixing configuration errors in a single app)

Conclusion

- Hard to recover from attacks *and* preserve legitimate user changes
- **Retro** repairs attacks, keeps legitimate changes
  - Key idea: *re-execution* of legitimate actions
  - *Predicates* and *refinement* minimize re-execution

Additional slides follow

Non-deterministic re-execution

- Goal: an acceptable execution
  - An execution that could have happened in the absence of the attack
- What if program is non-deterministic?
  - Re-run may lead to another acceptable execution
  - Result will not be influenced by attack
  - If significant differences arise (e.g., new crypto keys), might need user input to re-execute