Backtracking Intrusions

Sam King
Peter Chen
CoVirt Project, University of Michigan

Motivation

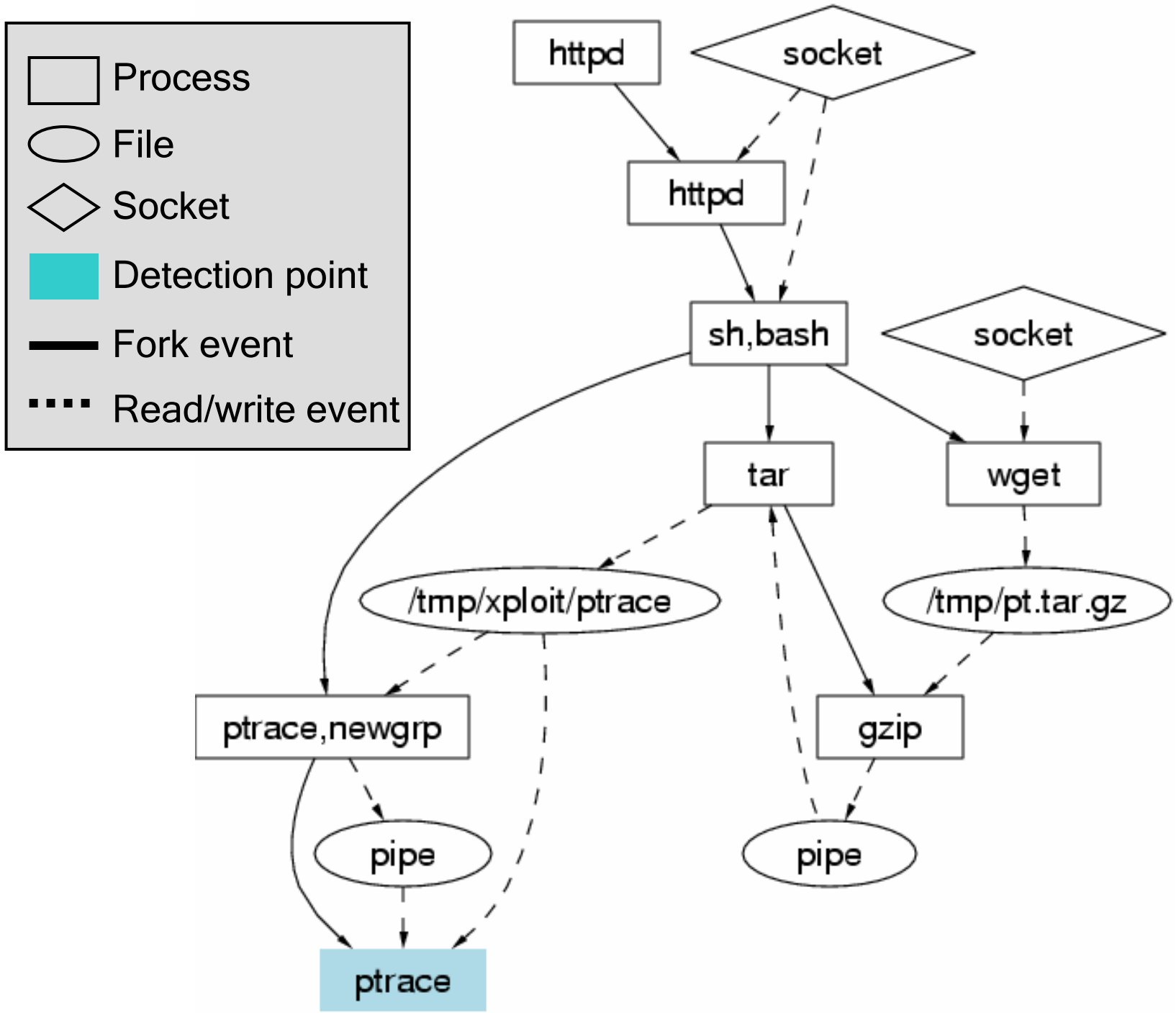
- Computer break-ins increasing
- Computer forensics is important
  - How did they get in

Current Forensic Methods

- Manual inspection of existing logs
- System, application logs
  - Not enough information
- Network log
  - May be encrypted
- Disk image
  - Only shows final state
- Machine level logs
  - No semantic information
- No way to separate out legitimate actions

BackTracker

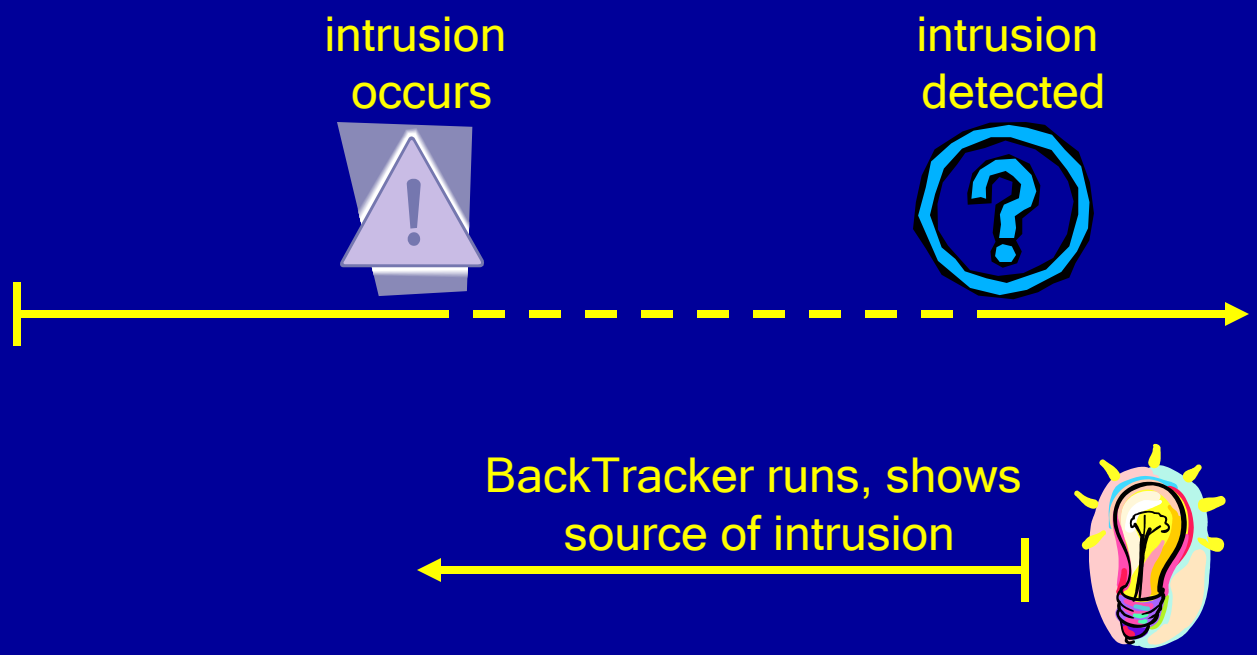
- Can we help figure out what was exploited?
- Track back to exploited application
- Record causal dependencies between objects


Process
File
Socket
Detection point
Fork event
Read/write event
httpd
socket
httpd
sh,bash
socket
tar
wget
/tmp/xploit/ptrace
/tmp/pt.tar.gz
ptrace,newgrp
gzip
pipe
pipe
ptrace

Presentation Outline

- BackTracker design
- Evaluation
- Limitations
- Conclusions

BackTracker



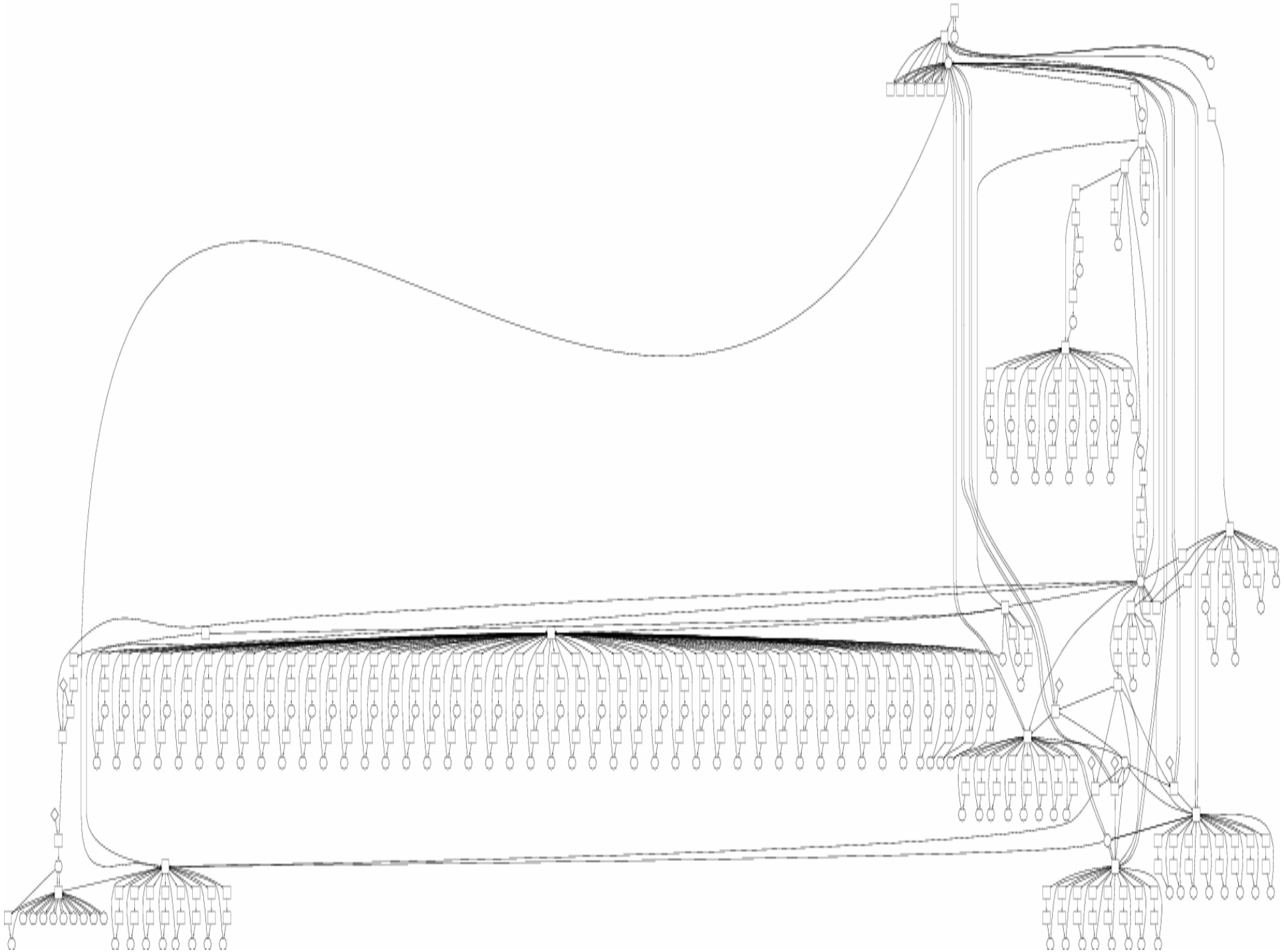
- Online component, log objects and events
- Offline component to generate graphs

BackTracker Objects

- Process
- File
- Filename

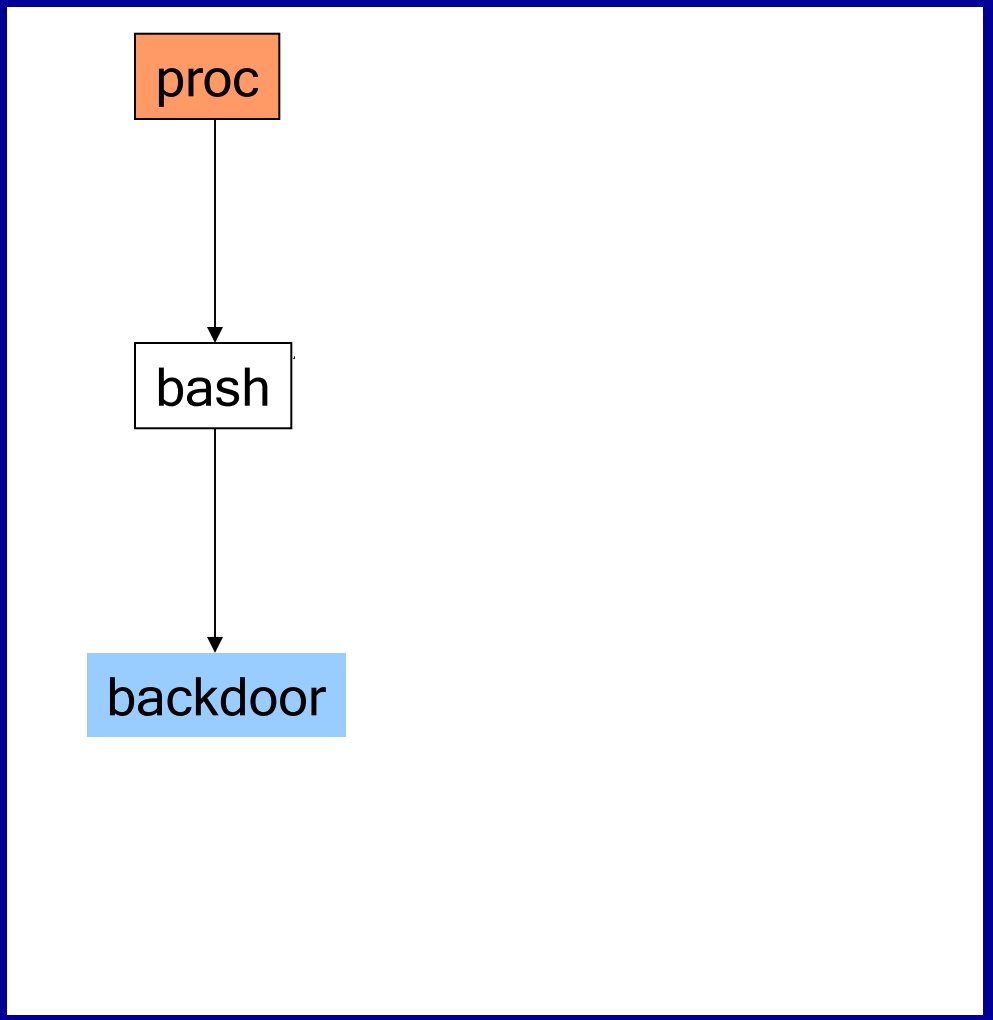
Dependency-Forming Events

- Process / Process
  - fork, clone, vfork
- Process / File
  - read, write, mmap, exec
- Process / Filename
  - open, creat, link, unlink, mkdir, rmdir, stat, chmod, …



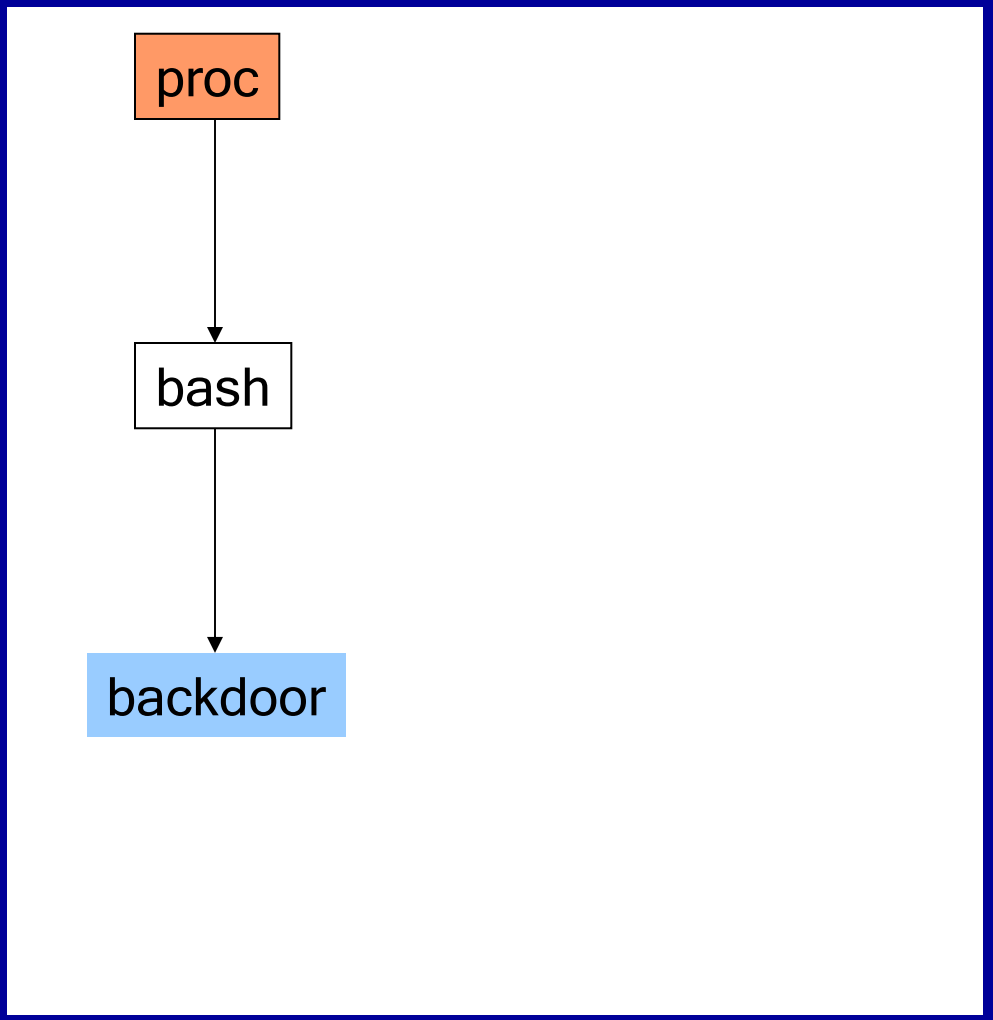
Prioritizing Dependency Graphs

- Hide read-only files
- Eliminate helper processes
- Filter “low-control” events



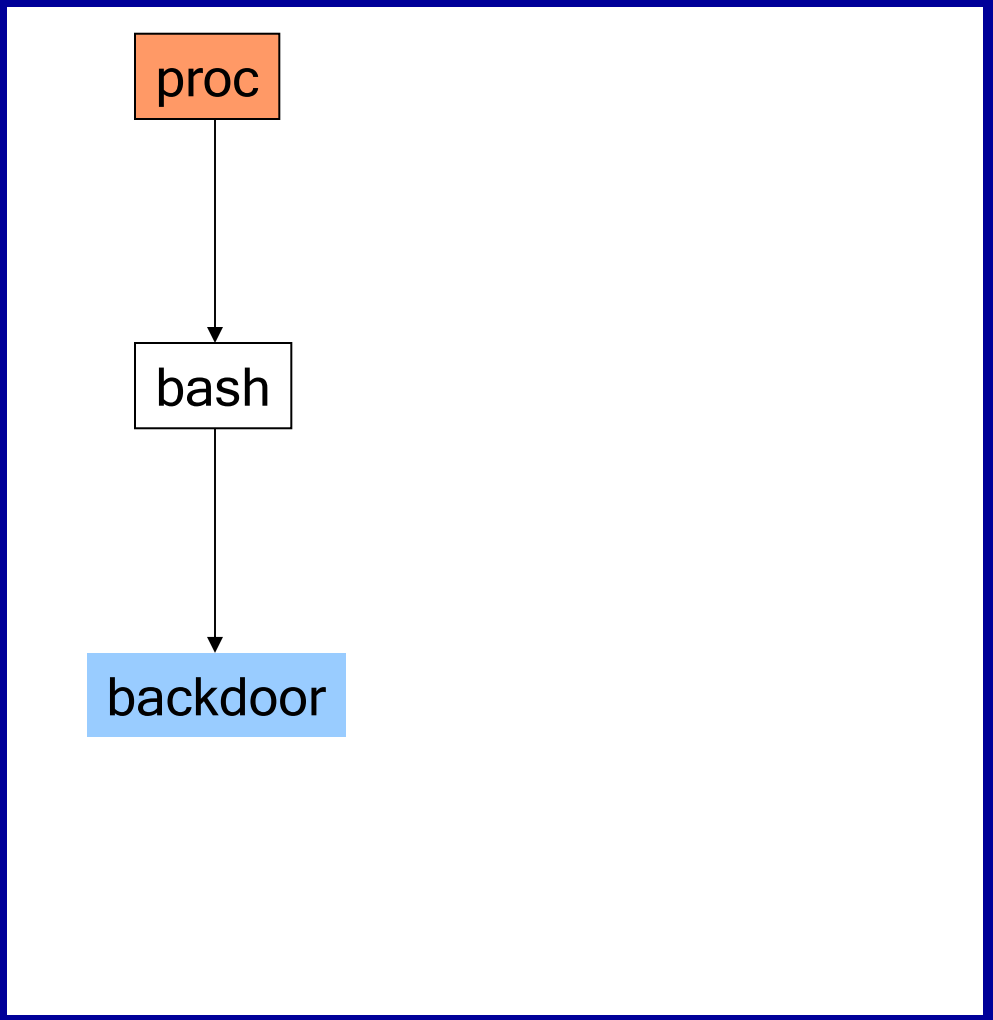
Prioritizing Dependency Graphs

- Hide read-only files
- Eliminate helper processes
- Filter “low-control” events

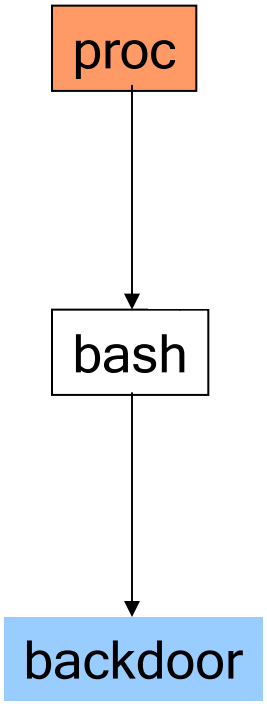


Prioritizing Dependency Graphs

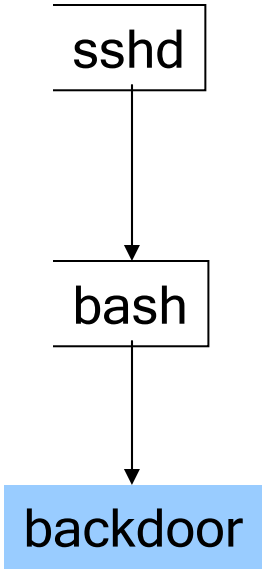
- Hide read-only files
- Eliminate helper processes
- Filter “low-control” events

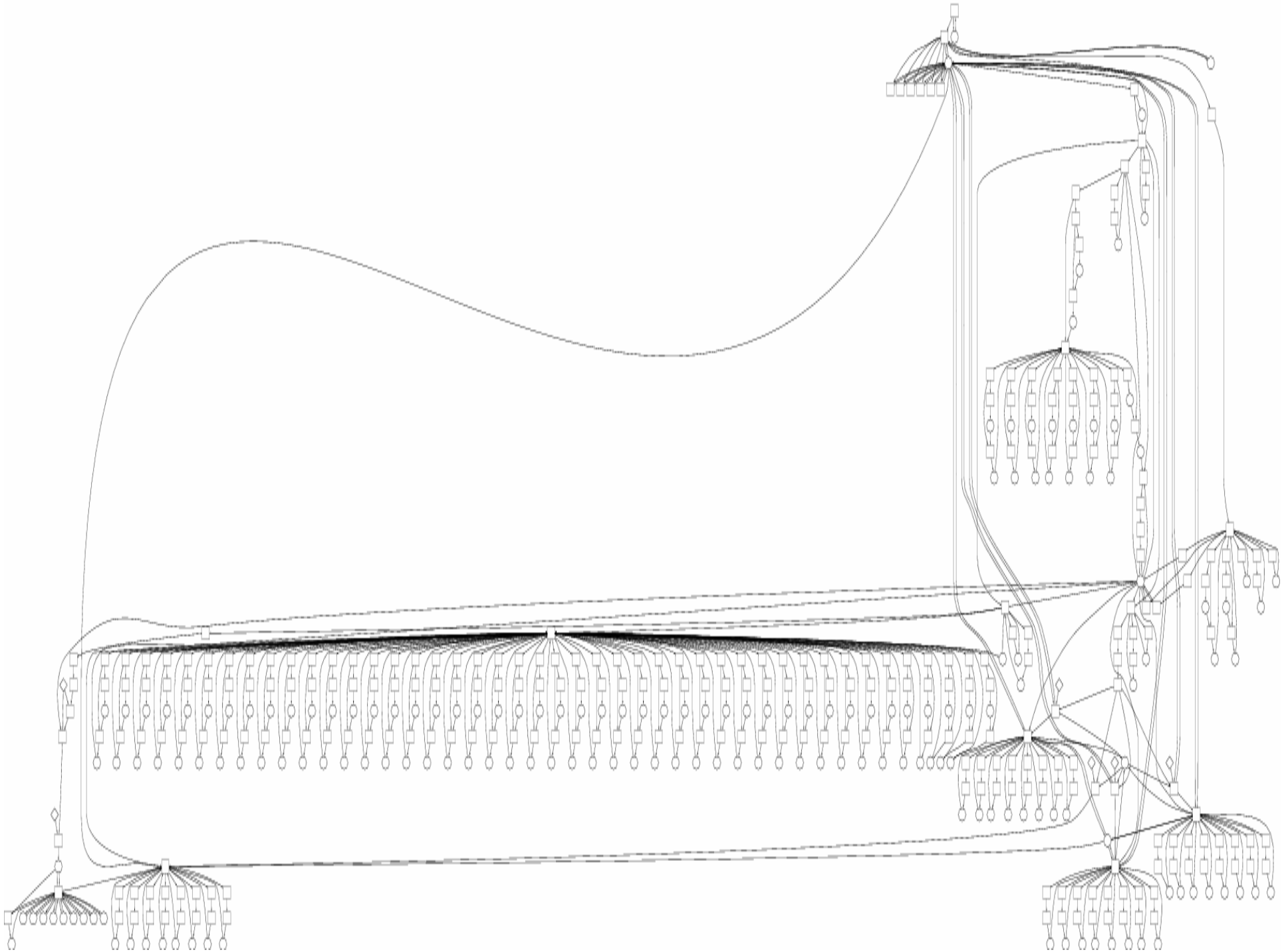


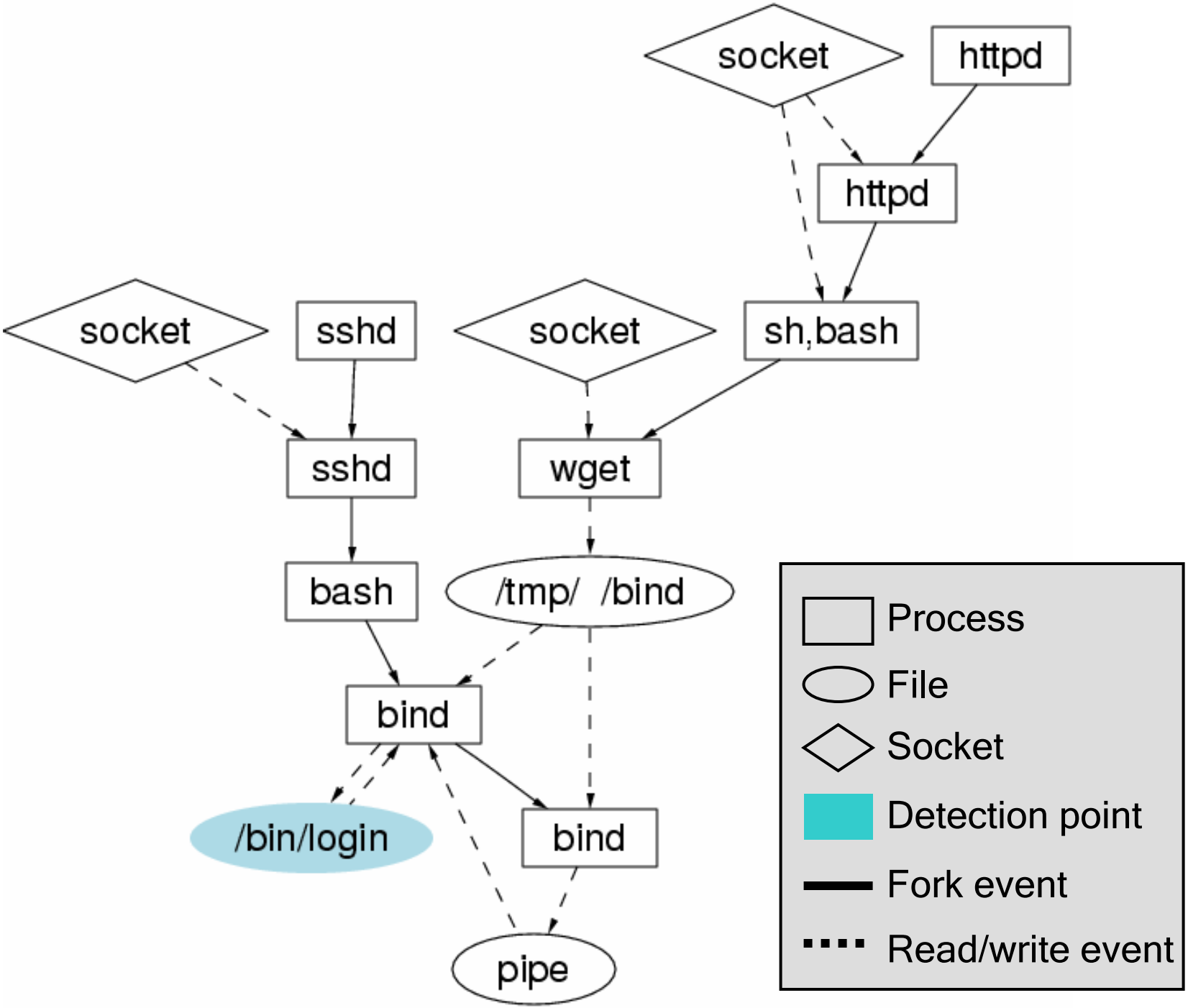
Filtering “Low-Control” Events



Filtering “Low-Control” Events

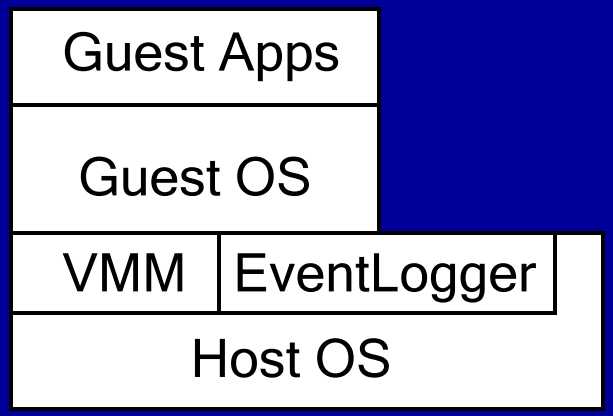




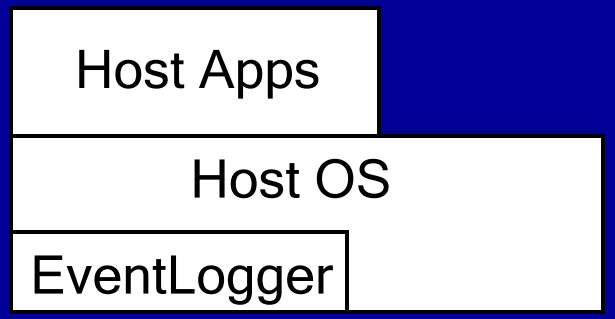

socket
httpd
httpd
socket
sshd
socket
sh,bash
sshd
wget
bash
/tmp/ /bind
bind
/bin/login
bind
pipe
Process
File
Socket
Detection point
Fork event
Read/write event

Implementation

- Prototype built on Linux 2.4.18
- Both stand-alone and virtual machine
- Hook system call handler
- Inspect state of OS directly



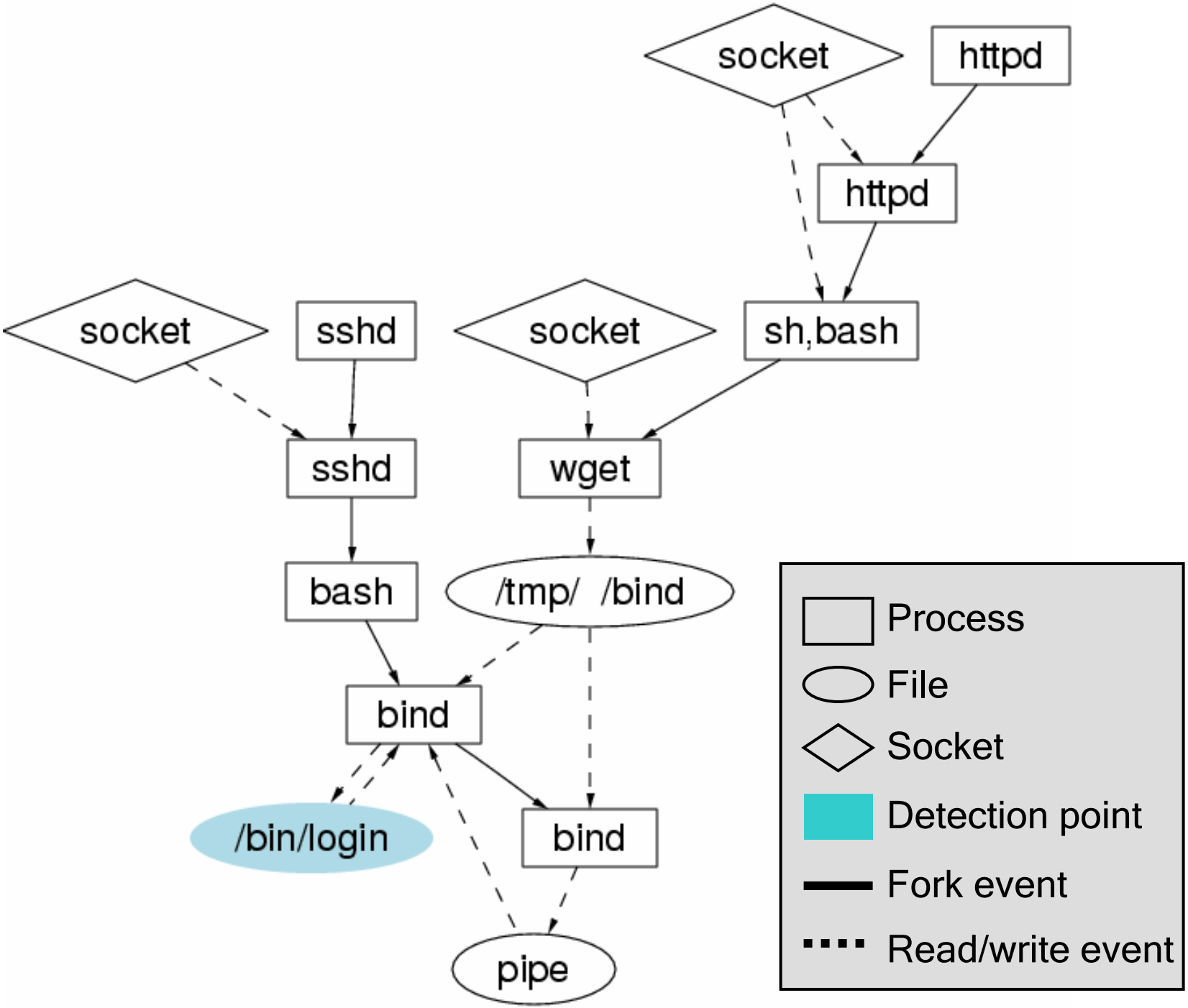
Virtual Machine Implementation

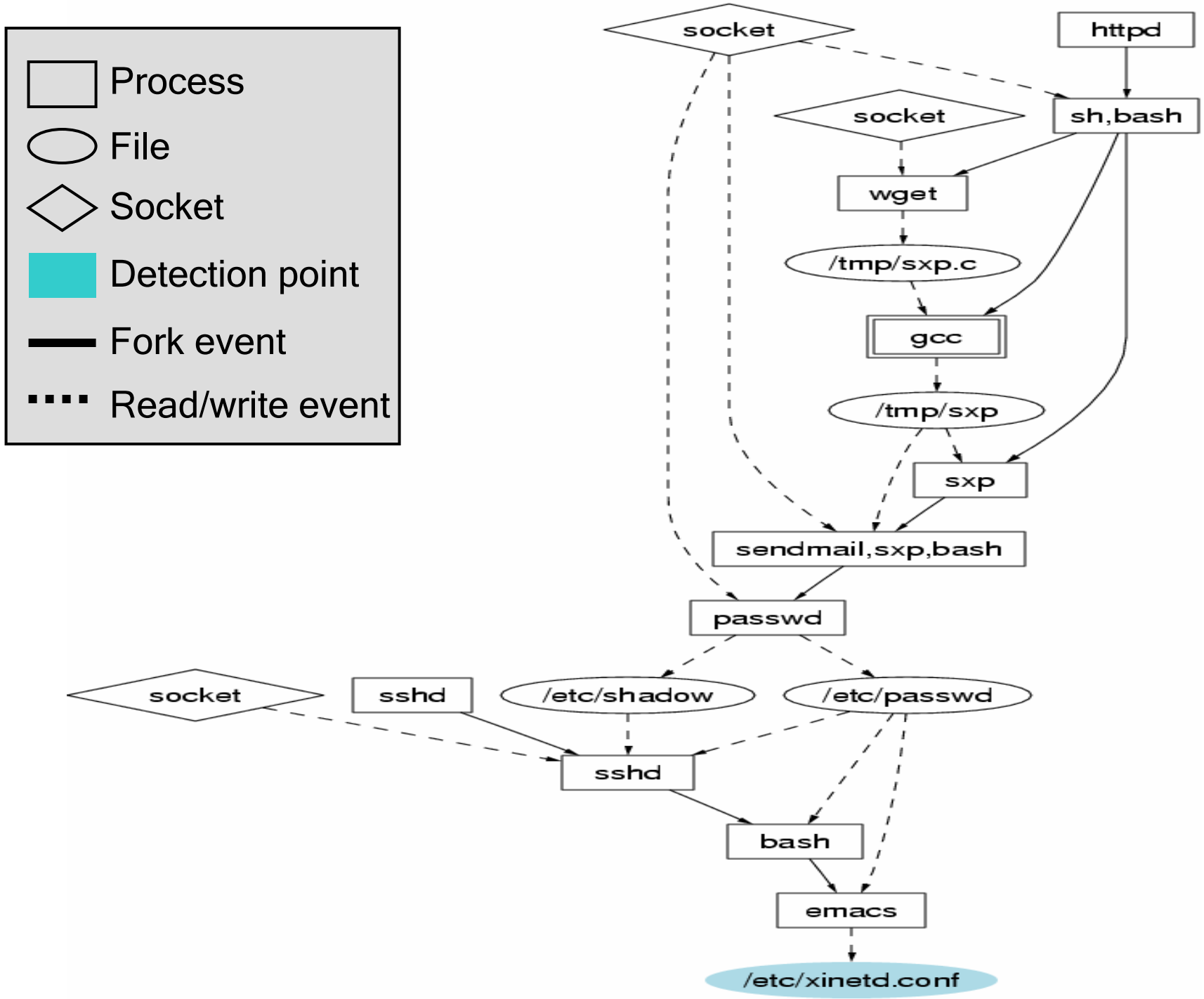


Stand-Alone Implementation

Evaluation

- Determine effectiveness of Backtracker
- Set up Honeypot virtual machine
- Intrusion detection using standard tools
- Attacks evaluated with six default filtering rules


socket
httpd
httpd
socket
sshd
socket
sh,bash
sshd
wget
bash
/tmp/ /bind
bind
/bin/login
bind
pipe
Process
File
Socket
Detection point
Fork event
Read/write event


Process
File
Socket
Detection point
Fork event
Read/write event
socket
httpd
socket
sh,bash
wget
/tmp/sxp.c
gcc
/tmp/sxp
sxp
sendmail,sxp,bash
passwd
socket
sshd
/etc/shadow
/etc/passwd
sshd
bash
emacs
/etc/xinetd.conf

BackTracker Limitations

- Layer-below attack
- Use “low control” events or filtered objects to carry out attack
- Hidden channels
- Create large dependency graph
  - Perform a large number of steps
  - Implicate innocent processes

Future Work

- Department system administrators currently evaluating BackTracker
- Use different methods of dependency tracking
- Forward tracking

Conclusions

- Tracking causality through system calls can backtrack intrusions
- Dependency tracking
  - Reduce events and objects by 100x
  - Still effective even when same application exploited many times
- Filtering
  - Further reduce events and objects