

BackTracker

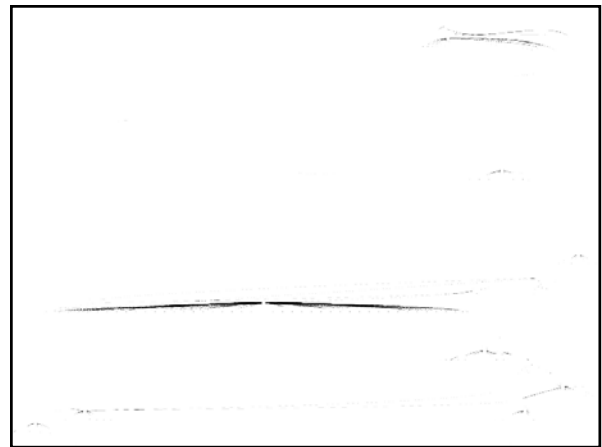
- Online component, log objects and events
- Offline component to generate graphs

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

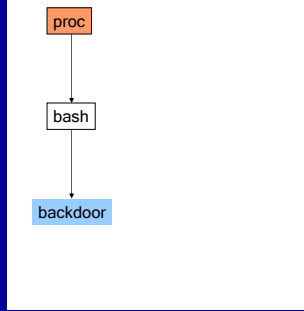
- Hide read-only files
- Eliminate helper processes
- Filter "low-control" events

Prioritizing Dependency Graphs

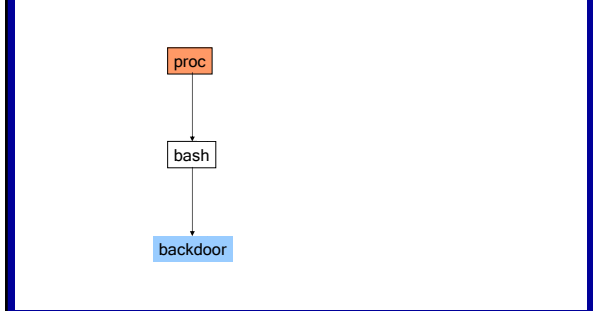
- Hide read-only files
- Eliminate helper processes
- Filter "low-control" events

Prioritizing Dependency Graphs

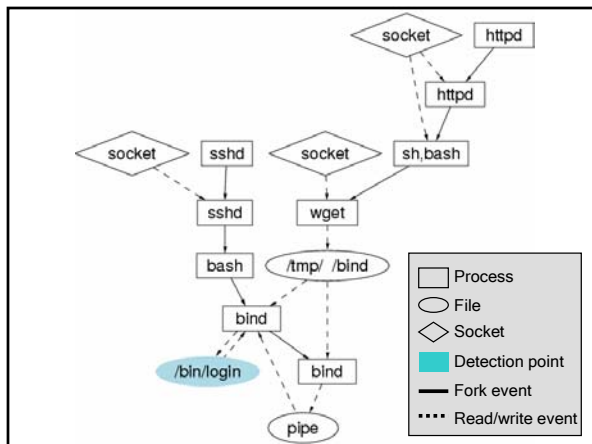
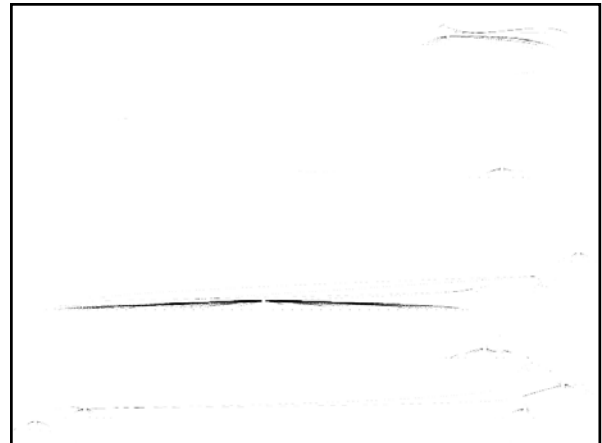
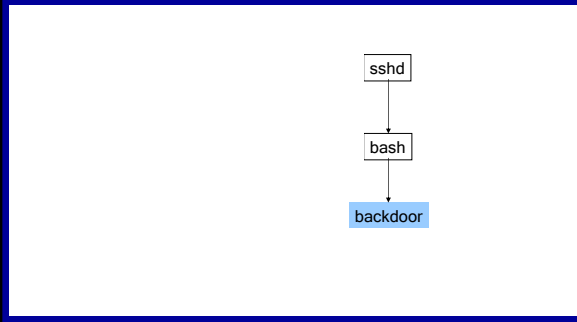
- Hide read-only files
- Eliminate helper processes
- Filter "low-control" events



Filtering "Low-Control" Events

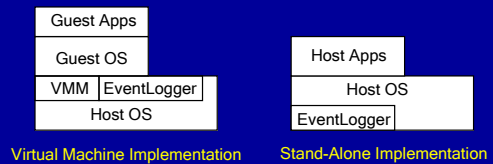


Filtering "Low-Control" Events



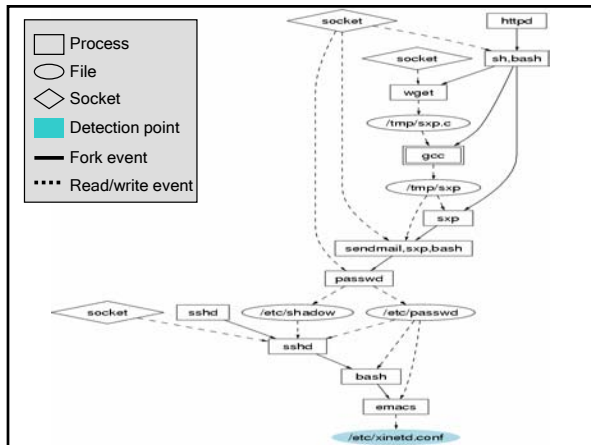
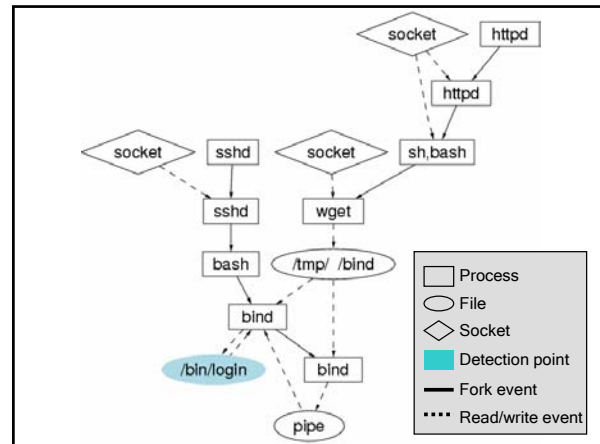
Implementation

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



Evaluation

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



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