Audit Workbench

Christopher Wee

James Hoagland

Wilson Choi, choiw@lsil.com

Bradford Wetmore, wetmore@bongos.ebay.sun.com

Faculty:

Karl Levitt

Biswanath Mukherjee

Matthew Bishop

University of California, Davis

Sponsored by National Security Agency, University Research Program

Outline

- What is auditing?
- Goals
- Visual audit browser
- Hypertext audit logs
- Scenarios
- Protocol-driven Audit Reduction
- Auditing Clients & Servers
- Audit Reduction
- Policy Enforcement & Security Model
- Further work

What is Auditing?

Logging — recording security relevant behavior by programs and users

Reduction — aggregation of low-level events into high-level, abstract events

Analysis —review logs for intrusions or policy violations

Why Audit?

- ❏ Review access of objects by users,
- ❏ Review the effectiveness of system protection mechanisms,
- ❏ Record attempts to bypass protection mechanisms,
- ❏ Detect uses of privilege greater than, or inappropriate for, the role of the user,
- ❏ Deter perpetrators, and
- ❏ Assess damage and assist in recovery from intrusion.

Goals

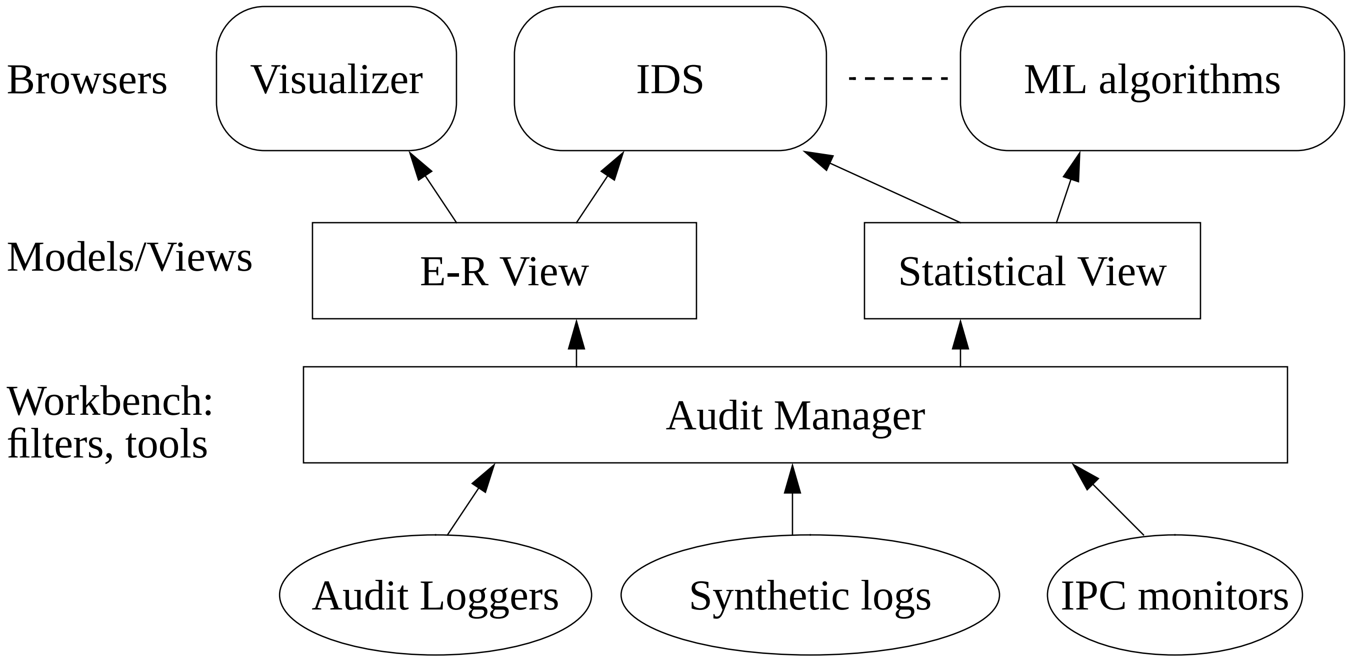
❑ Tools for System Security Officers (SSO)

- Filters
- Visual graphs
- Graphical, interactive audit browsers

❑ Tools for researchers

- system-independent audit logs
- tools to combine, reduce, splice, etc. logs
- portable analysis algorithms
- portable representation of security relevant behavior and state

Audit Workbench



BSM Audit Log

```
file,Thu Oct 21 16:23:39 1993, + 970501 msec,
header,107,execve(2):,Thu Oct 21 16:23:43 1993, + 160000 msec
path,/,/usr/export/home/heberlei,/usr/export/home/heberlei/loadmodule
process,heberlei,heberlei,heberlei,staff,330
return,No such file or directory,-1
trailer,107
header,53,vfork(2): process creation,Thu Oct 21 16:23:43 1993, + 170000 msec
argument,0,330,child PID
process,heberlei,heberlei,heberlei,staff,319
return,Error 0,330
trailer,53
header,120,execve(2):,Thu Oct 21 16:23:43 1993, + 170000 msec
path,/,/usr/export/home/heberlei,/usr/openwin/bin/./loadmodule
attribute,104755,root,staff,1822,55365,56424
process,heberlei,root,heberlei,staff,330
return,Error 0,0
trailer,120
header,104,open(2): read,Thu Oct 21 16:23:43 1993, + 170000 msec
path,/,/usr/export/home/heberlei,/usr/lib/ld.so
attribute,100555,root,staff,1822,101476,25280
process,heberlei,root,heberlei,staff,330
return,Error 0,3
trailer,104
...
hheader,35,exit(2): process termination,Thu Oct 21 16:23:49 1993, + 100000 msec
process,heberlei,root,root,daemon,334
return,Error 0,0
trailer,35
header,141,stat(2):,Thu Oct 21 16:23:49 1993, + 610000 msec
path,/,/usr/export/home/heberlei/.wastebasket,/usr/export/home/heberlei/.wastebasket
attribute,42755,heberlei,staff,1822,59984,4414
process,heberlei,heberlei,heberlei,staff,174
return,Error 0,0
trailer,141
file,Thu Oct 21 16:23:51 1993, + 447661 msec,
```

Figure 1. Excerpt from BSM audit log

Visual Audit “Browser”

❏ Initial audit browser prototype “ab” [Wetmore92][1]

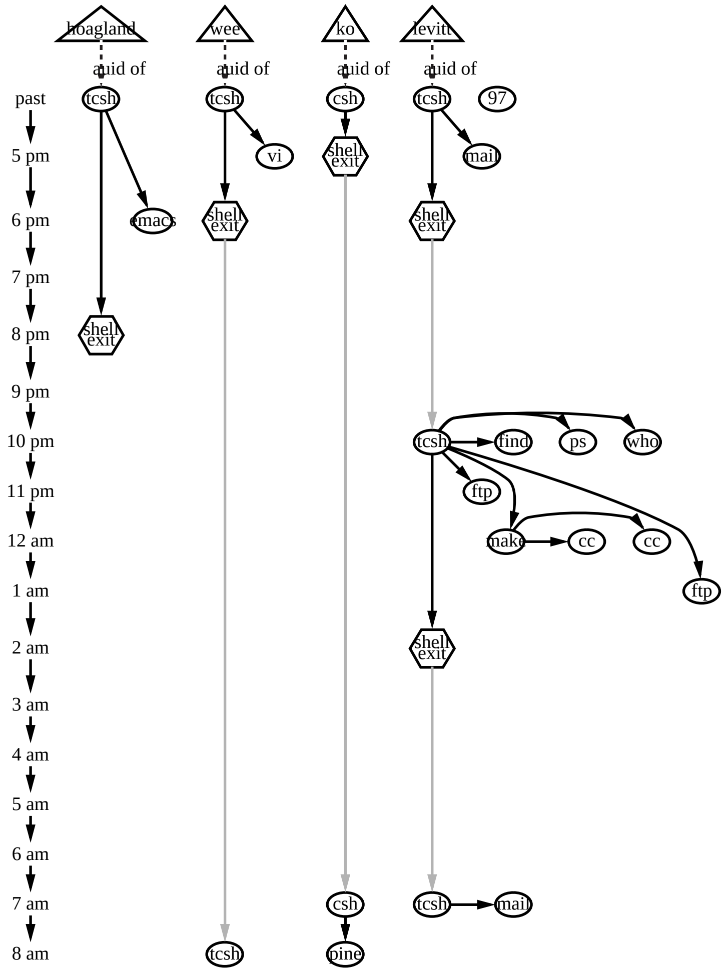
- text based
- written in C, cumbersome to modify
- Sun BSM audit logs

❏ Visual audit browser [Wee, Hoagland94]

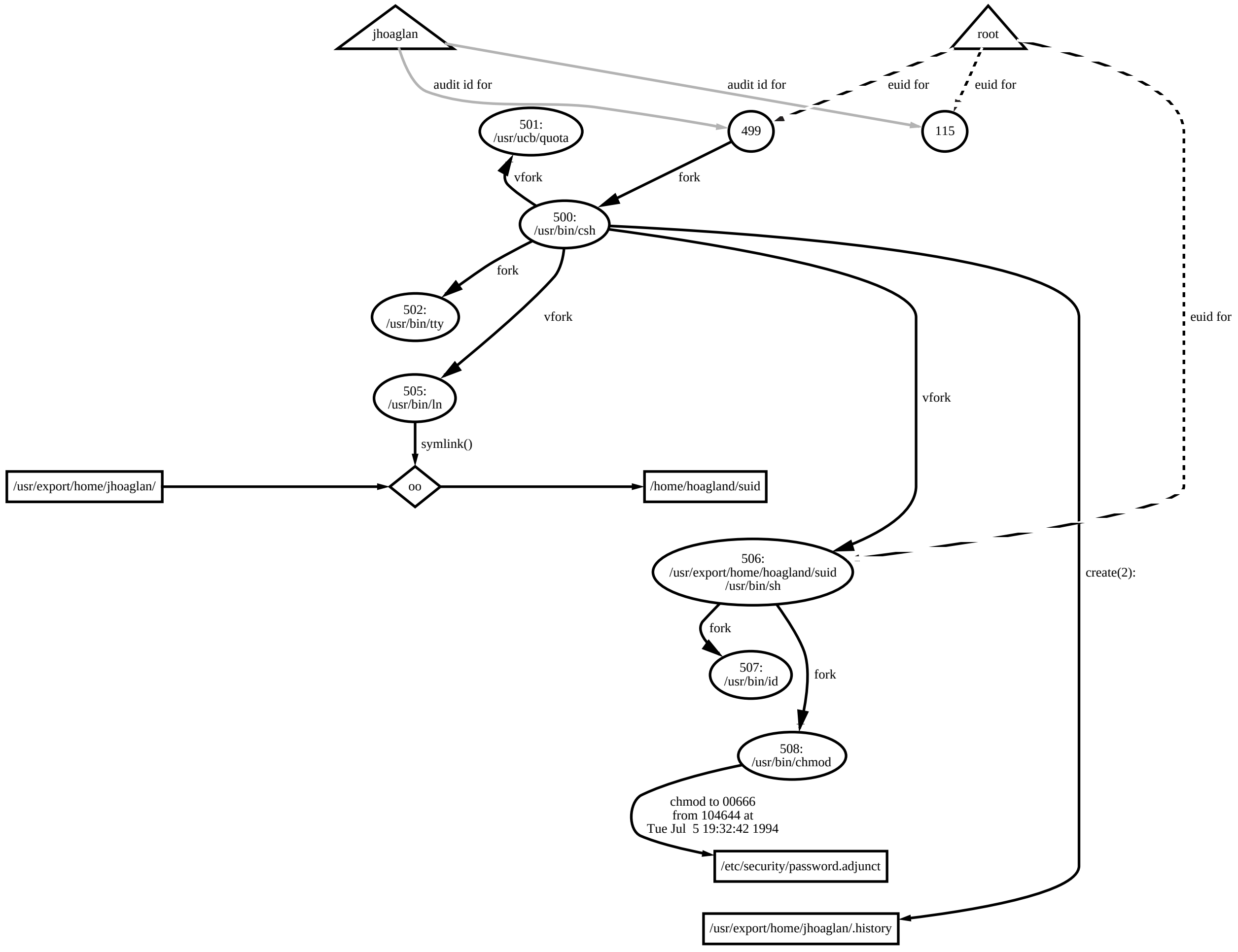
- transformation of audit logs into DAGs with annotations
- graphs produced by AT&T’s dot tool
- written in Perl, easily extensible
- interactive browsing (work-in-progress)

1. presented NSATechFest93

Visual summary of overnight audit log



Visual summary of suid-shell script attack



Visual Audit Browser (continued)

Benefits

- ❏ Irrelevant details are filtered
- ❏ Enables the user to scan for unusual patterns
- ❏ Useful in studying attacks that exploit system vulnerabilities

Difficulties

- ❏ temporal relationships are hard to discern
  - prototype of a audit log “movie maker” (in progress)
- ❏ Multiple views required
  - Control-flow, process centered view
  - Data object view
  - Information flow view
  - Accountability-flow view (in progress)
- ❏ Filtering is arbitrary
- ❏ No automatic inferences and minimal reduction performed

Hypertext Audit Logs

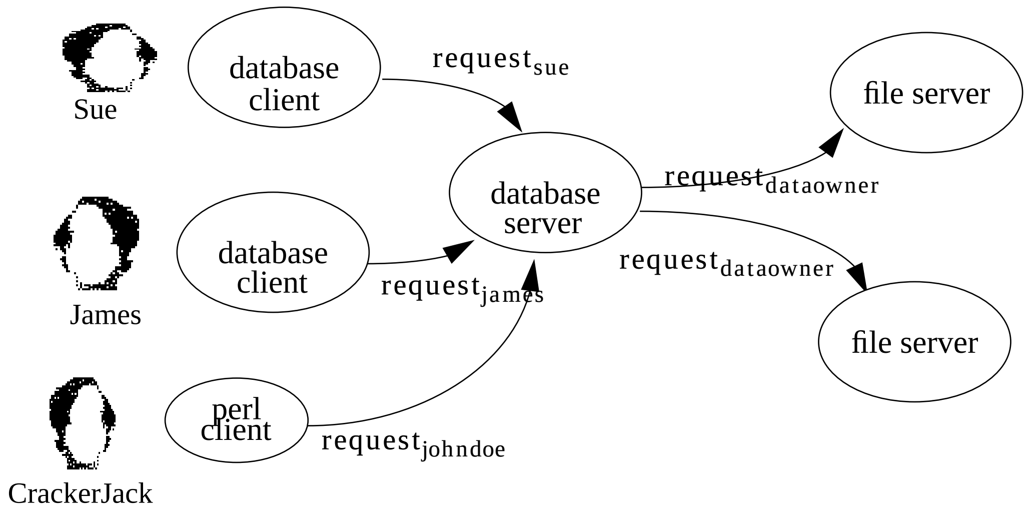
Extending the original text-based audit browser

- ❑ HTML, W^3 http, & NCSA mosaic
- ❑ Hypertext allows rapid investigation of audit logs
- ❑ Full audit log details available
- ❑ WWW permits distributed browsing and annotation
  - coordinated analysis by SSOs at different sites

Disadvantages:

- ❑ No filtering
- ❑ Security of HTTP protocol is weak
- ❑ Few automated inferences
- ❑ Not portable across different audit systems
- ❑ Human must search for malicious activity
  - malicious activity may remain undetected

A client-server database system



Challenges:

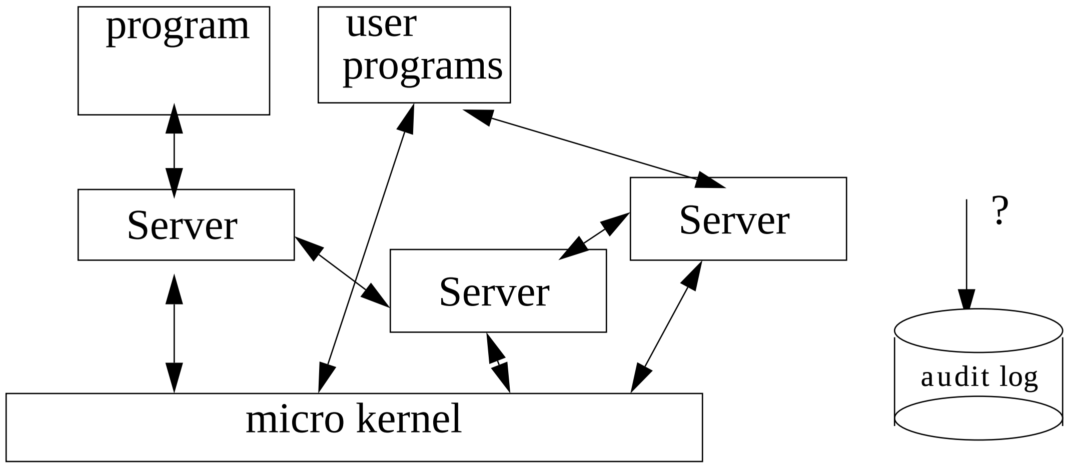
❏ Indirection through database server obscures accountability

❏ OS access control only mediate direct accesses, not indirect ones

❏ Cannot rely upon server authentication

- inadequate
- buggy or contain trojans, back-doors

Microkernel operating system



Challenges:

- ❏ Auditing is distributed
- ❏ Audit logs likely to be more detailed, less coherent
- ❏ Checking servers introduced to the secured environment for malicious elements

Approach

- ❏ Anomaly detection — compare old traffic with new traffic
- ❏ Audit analysis — compare new traffic against protocol specification

Personal computers & peripherals

Challenges:

❏ Lack adequate identification & authentication

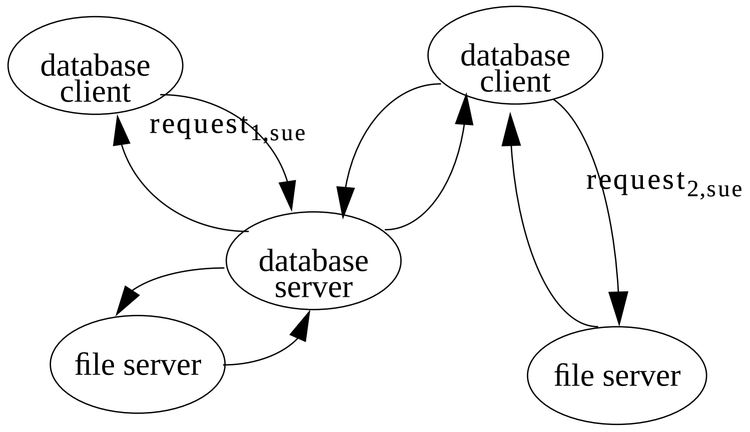
❏ Can serve as storage channels or launchpads for attacks

- Possess network ports, network identifiers, increasing amounts of computation power and “intelligence”

Approach:

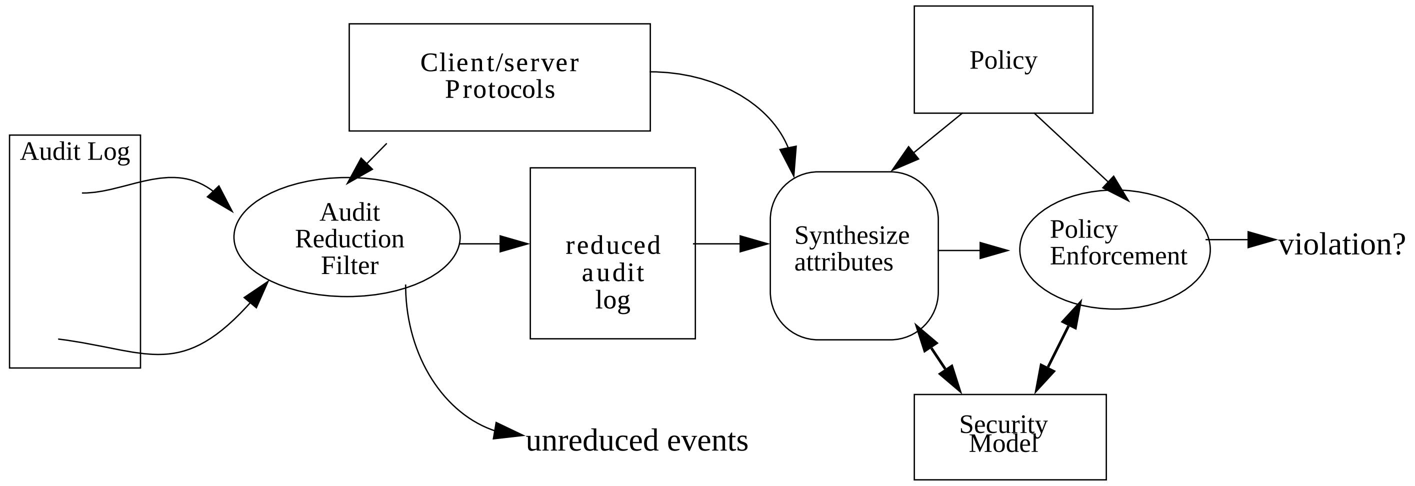
❏ Exploit the fact that even PCs and peripherals use standard protocols

Auditing Clients & Servers



- ❑ No changes to the clients or server
- ❑ Express audit logs in terms of the abstractions used in security policies
  - e.g., users, information, rights, instead of processes, files, inodes etc.
- ❑ OS services are logged in the system audit logs
- ❑ C-S transactions are logged by monitoring Inter-Process-Communications (IPC)

Protocol-driven Audit Reduction



- ❏ Transactions between clients & servers (C-S) makes analysis tractable
- ❏ Information is retained, clutter is reduced
  - e.g., series of read()s interspersed by NFS_Read IPC replaced by FILE_READ
- ❏ Audit event parameters are matched and checked across multiple audit events
  - consistent values are retained as attributes of the reduced log
  - inconsistent values raise warning flags
- ❏ Events unexplained by the protocol are highlighted for further examination

Protocol-driven Audit Reduction (cont'd)

Benefits:

❑ System independence
- e.g., simple model of Unix processes
- e.g., NFS, DNS, HTTP standards

❑ Systematic reduction

❑ Reduced logs are more abstract and less clutter

❑ Distributed audit aggregation
- most distributed systems interact using standard protocols

Research goals:

❑ Methodology for solving accountability tracing

❑ Retrofit delegation of access rights into Unix protection model
- synthesized delegation credentials

❑ Automatic translation of protocols into audit reduction filters

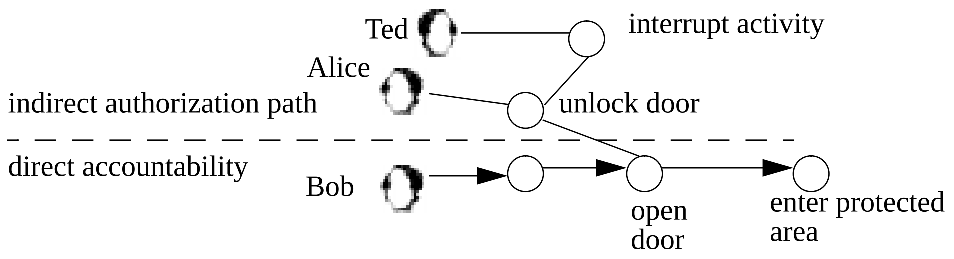
❑ Portable reduction and analysis algorithms

Tracing accountability and authorization

- Accountability — who was responsible?
- Authorization path — how or from whom were permissions/rights obtained?

Accountability is a property of a system that enables actions to be traced to the user responsible for initiating the activity.

The **authorization path** is the set of all principals that delegated rights to the



accountable principal or otherwise “contributed” to the activity.

Reducing C-S Transactions

- ❑ Client-server transactions are represented by a production

NFS_READ	->	[open()] read()+
user X		user Y
cred_list_X		cred_list = { cred_list_X U new($\text{cred}_{X\text{->}Y}$) }
		auth_list = { new($\text{cred}_{\text{root->}Y}$) }

- ❑ based on finite state automata or grammars
- ❑ tailored for each client-server protocol specification
- ❑ attributes synthesized tailored to needs of policy enforcement
- ❑ Initial prototype using attribute grammars on subset of NFS in [Choi93][1]

1. Presented at NSA TechFest93

Example: Reduction using templates

On the client-side

C_Nfs_Read(F)--> read(F) by X@P (NFS_READ(F) from NFS@S to Y@D)+

acct_list = acct_list $\wedge$ (X@P $\rightarrow$ NFS@S)

Figure 2. High-level Nfs_Read production (client side)

On the server side,

S_Nfs_Read(F) --> NFS_READ(F) from Y@S to NFS@D [open(F) by X@P] (read(F) by X@P)+

acct_list = acct_list $\wedge$ (Y@S $\rightarrow$ X@P)

Figure 3. High level NFS_READ production (server side)

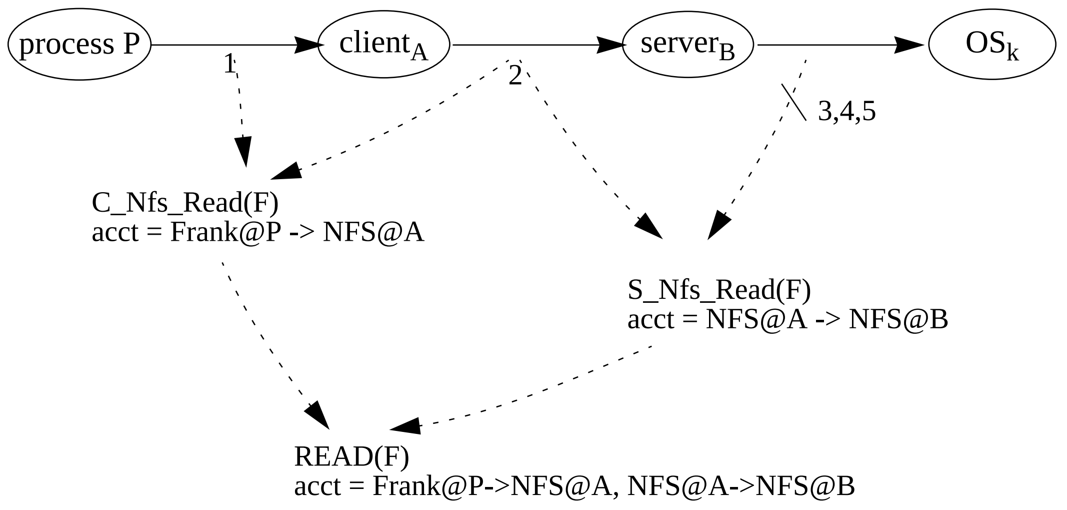
To aggregate client and server sides,

Read(F)--> C_Nfs_Read(F) with Clist S_Nfs_Read(F) with Slist

acct_list = Clist $\wedge$ Slist

Figure 4. Aggregating client and server events

Result of reduction and synthesis



Aggregation[1]

From raw audit events,

open(), NFS_LOOKUP, NFS_GETATTR, read(), NFS_READ, read(),...,
NFS_READ, NFS_GETATTR, close()

Using specification-based audit reduction, audit output might look like,

FILE /net/mailserver/usr/spool/mail/wee read by daemon @ 11:33:04 PST

Adding synthesized accountability, we have

FILE /net/mailserver/usr/spool/mail/wee read by

[wee@client->root@client->root@mailserver->daemon@mailserver]
authorized by [wee@client->root@client->root@mailserver].

1. The prototype required much exception handling and cannot yet trace attributes this concisely.

Policy Enforcement / Intrusion Detection

❑ Security policies do not translate into a well defined set of behavior

Model of security state

❑ Security state of a system is dictated by policy

❑ Initial security state is affected by system’s initial state & system configuration

- exact information about system configuration is hard to obtain due to upgrades, patches, re-configurations etc.

❑ How does specific behavior affect security state?

Alternate approach:

❑ Protocol-based audit reduction defers need for complete model of security

- only requires protocol and minor assumptions about security model

❑ Lower level security policy requirements are easier to define

- e.g., definitions of objects, users, ownership, permission

Further Work

- ❑ Enhancements to visuals
  - data-centered graphs
  - animated displays of audit logs
- ❑ Aggregate Sun BSM and HPUX audit logs
- ❑ Aggregating system logs with other sources of information
- ❑ Protocol-based audit reduction
  - more reductions
  - simple inferences about security state of a Unix system
  - portability
- ❑ Tampered audit logs
  - simple mutations
  - effects of simple mutations