

PCFS: A Proof Carrying File System

Deepak Garg

Carnegie Mellon University

December 08, 2008

Motivation and Goals

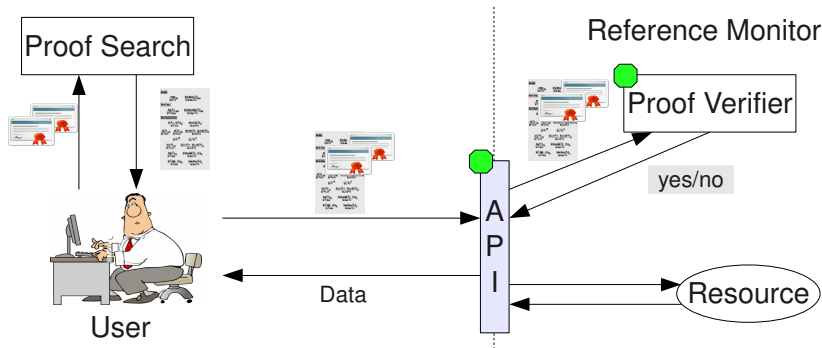
- Objective: investigate issues in PCA at a system level (OS, file system, ...)
- Speed of access checks
 - ▶ Proof verification: $\sim 7\text{ms}$ (Grey)
 - ▶ File system: $< 100\mu\text{s}$ (ext3)
 - ▶ Can we leverage PCA and still have ext3 like speed?
- What method of proof search will work?
 - ▶ Grey investigates some issues
 - ★ Customized method, mixes backward and forward search
 - ★ Small, distributed policies
 - ▶ Policies can be much larger.
 - ★ Classified information: ~ 40 fixed rules + ~ 35 credentials per access

Outline

- Optimize PCA architecture for speed
 - ▶ Idea: Use cryptographic capabilities
 - ▶ Has other benefits too

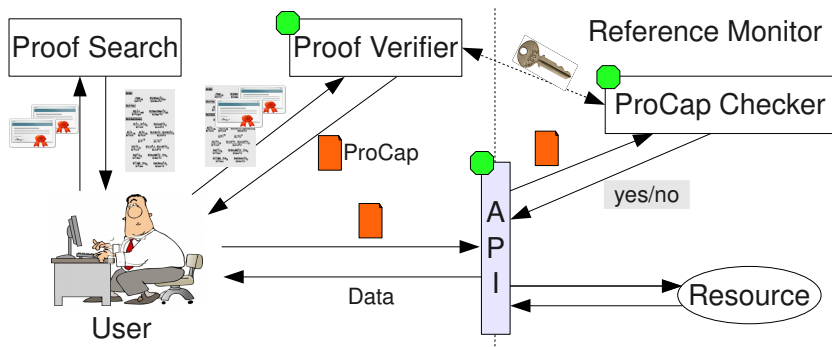
- Generic proof search
 - ▶ Still under exploration
 - ▶ Initial result: goal directed search (*a la* Prolog) works well

PCA (Traditionally)



- For heavily loaded systems, proof verification can be a bottleneck

PCA (With Capabilities)



- ProCap = PROven CAPability
- ProCaps can be *checked fast* ($\sim 100\mu s$)
- ProCaps are *signed* by the verifier
- ProCaps act as a *cache* of proof verifications

Procap Contents

Property: If a ProCap checks at some time T , a valid proof authorizing the corresponding access at time T exists at time T

- Object authorized (file name)
- Principal authorized (uid 1000)
- Permission authorized (read, write, ...)
- Signature (MAC) with a symmetric key shared between verifier and reference monitor
(Symmetric key is the “trust” between verifier and reference monitor)
- *Conditions* that must be checked by reference monitor
 - ▶ Time range (e.g., from 11/01/08 to 11/01/09)
 - ▶ File attributes on which proof depends (owner, extended attributes)
 - ▶ (Optional) List of certificates used in proof for check against CRL

ProCap Benefits

- Reference monitor works faster (~ 100 times)
- Allow for multiple, distributed verifiers, separate from access point
- Allow for multiple logics / policy languages with same reference monitor

Demo of File System Core

Proof search

- Forward search or backward search?
 - ▶ Depends on policy
 - ▶ Number of proofs generated
- Implemented backwards, goal-directed search (*a la* Prolog)
- Logic is called BL; has (K says A) and $A @ [T1, T2]$
- Logic syntax (for proof search):

$$\begin{array}{l} \text{Goals } G ::= P \mid c \mid G_1 \wedge G_2 \mid G_1 \vee G_2 \mid \forall x. G \mid \exists x. G \\ \quad \quad \quad \mid H \supset G \mid \top \mid \perp \mid K \text{ says } G \mid G @ [T1, T2] \\ \text{Clauses } D ::= P \mid D_1 \wedge D_2 \mid G \supset D \mid D @ [T1, T2] \\ \text{Hyps } H ::= D \mid K \text{ says } D \mid c \mid H @ [T1, T2] \end{array}$$

- Current times: 100-300 ms + certificate checking (few ms per certificate)
 - ▶ Realistic policy (classified information in USA)
 - ▶ Conservative time estimates: No optimizations yet

Demo of Proof Search

Summary of Approximate Times

Orders:

Proof Search	~1s	
Proof Verification	~10ms	(Grey)
Procap Checking	~100 μ s	

PCFS Operations:

With ProCaps	1000-2000/s	(Bonnie++)
Direct verification	50-100/s	(Bonnie++)

Other considerations

- Proof theory of logic
 - ▶ Not all modalities are equally amenable to goal directed search (lax logic?)

- What permissions should we use?
 - ▶ Depends on use cases
 - ▶ PCFS uses 5 (read, write, execute, govern, identity)