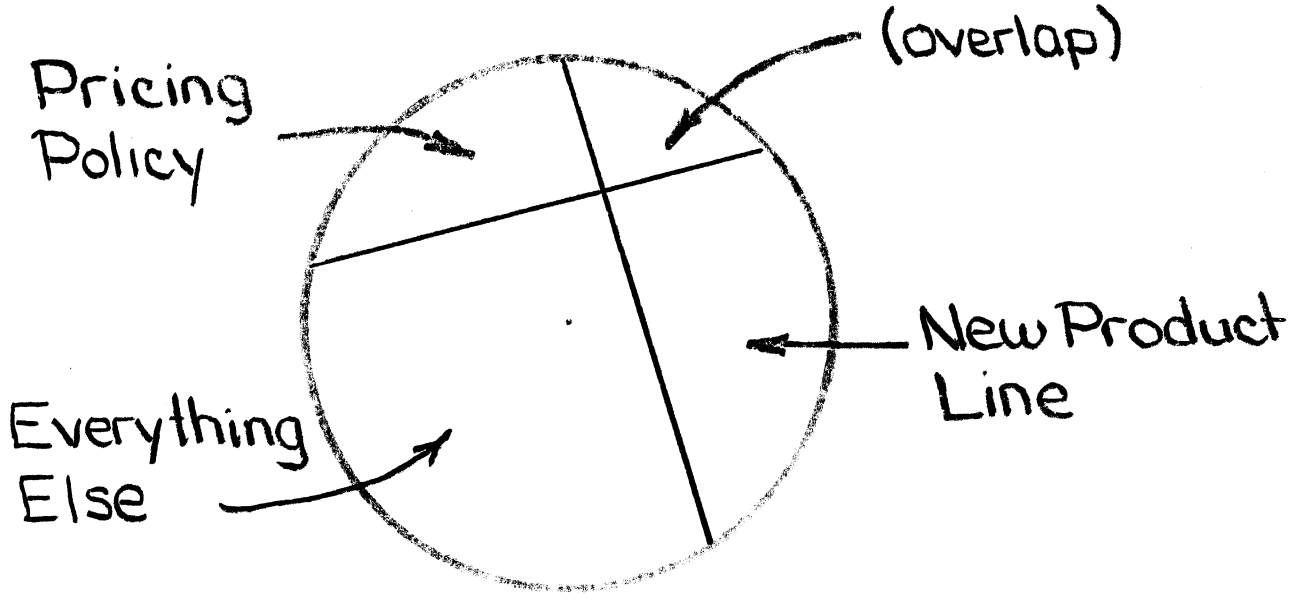
Goals

1. Isolation of Overlapping Compartments

2. Use of Uncertified Programs

Compartments



Authorizations:

Smith: Pricing Policy

Jones: New Product Line

Reilly: Pricing Policy, New Product Line

For Confinement of Uncertified Programs

S_D = { set of compartment labels on file D }

S_C = { set of compartment authorizations of computation C }

for reading:

S_D must be contained in S_C

for writing:

S_D must be identical to S_C

The "System" Problems

1. Enforce the write restriction when uncertified programs are in use

2. Reliably decide when the write restriction may be relaxed

3. Be sure that #1 and #2 work right

Session chair
Prof Robert Fabry
Univ Cal at Berkeley

Intro

Morning session on security suggested there are still some communication troubles as to

- what problems there are
- what problems are being worked on

My position

There are some fairly simple real-world needs which ~~and~~ available systems don't meet very well.

I am focusing on one of these simple problems.

It comes up when one has these two goals.

(Slide #1)

(1)

#1 comes up when a company wants to lay down rules about information transfer, and be sure that they are enforced.

#2 arises because one cannot write all his own software

Let's look more carefully at the first goal to see what it means

Slide 2

(2) Intuitive Interpretation:

Suppose that a company's files are to be organized into watertight compartments, eg.

thus you would what the pix shows

- files on pricing policy are accessible to Smith + Reilly

- files on NPL are accessible to Jones + Reilly

- files in overlap area are accessible ~~also~~ only to Reilly.

Problem: What happens when Reilly talks to Smith? Reilly is contaminated with NPL info, and must use judgement.

If Reilly writes a memo to Smith, it should be labeled PP, and checked to make sure it doesn't accidentally contain NPL info.

Thus: Reilly's writing in PP files ~~should be reviewed~~ must be done only ~~PP files written by Reilly~~ if judgement is exercised.

~~A problem arises when someone who has been working in the overlap, or using the new product line, goes back to talk to someone in the Pricing Policy compartment.~~

~~He must use __judgement__ to avoid unauthorized information release.~~

~~He is __contaminated__ with NPL info, and must be careful not to release it.~~

Unaudited Programs.

They cannot [depended on to] provide judgement
- mistakes
- Trojan Horse planted by espionage specialist.

Conclusions:

Program operating in overlap area

1. Can __read__ data from Pricing Policy __or__ from NPL

Must be restricted → 2. Can __write__ data __only__ in the overlap area

(Writing anywhere else can be done only following the exercise of judgement)

This leads to a generalization

Slide 3

(read slide)

every read or write access made by

the program must be constrained to follow these rules.

that leads to three primary problems:

Slide 4

1. Enforce write restriction
   harder than it appears

   a. May not be possible in a capability system, with unacceptable covert channels.
      (natural with access control lists)

   b. "Borrowing on the walk" problem.
      (example: squirreling in Multics by increasing ~~decreasing~~ presence of a page in core (shared pure library procedure), could transmit 1 bit/second with error rate of one in 50.

2. Hard becomes ~~if~~ you must make sure that all programs which might try to program relocking are certified to execute
   - borrowed program
   - library
   - editors, compilers, etc.

3 is hard because it involves having some complex program that works right.

Conclusion

even a fairly simple real-world need leads to some hard system problems.