Feb 28, 2005 -- Lecture 17

22C:169
Computer Security

Douglas W. Jones

Department of Computer Science

Trusted Systems

Trusted Systems

Definition:

Foundations for trusted applications

Directions in trusted systems work

Military inspired

Accept classification hierarchy as a system requirement

Computer science inspired

Draw on data abstraction and scope rules

These produce different results!

The Military Model, Generalized

Security hierarchy

Unclassified, confidential, secret, top secret, cosmic, ...

Security taxonomy

All information is compartmentalized by topic: cryptography, geography ...

Classification of resources:

`<rank; compartments>`

Clearance of users:

`<rank; compartments>`

The military access rule

Given user `U` with

`U.clearance`

And resource `R` with

`R.classification`

`U` $\geq$ `R` (`U` may access `R`) if both

`U.clearance.rank`

$\geq$ `R.class.rank`

`U.clearance.compartments`

$\supseteq$ `R.class.compartments`

In general, in military models

User `U` may access data `D` if

`U ≥ D`

If user `U` writes `D`

`D.class = <U.class.level,C>`

where `C` $\subseteq$ `U.class.compartments`

In general, for users U_1 and U_2

U_1 *may not speak to* U_2

if U_1`.clearance >` U_2`.clearance`

A DRACONIAN RULE!

Downward information flow:

A central problem with military models

If only upward flow is permitted

Commander can never issue orders!

In sum

Real systems always violate the model

Confusion of trust and accuracy

Trust (Bell LaPadula):

do you trust user U with this data?

The issue is nondisclosure

Accuracy (Biba):

How accurate is data D?

The issue is data integrity

Hierarchic models apply to both

But, are they the same hierarchy?

System Architecture

How do you build a secure system?

- *Lessons from history:*
  - Insecure results are the norm
  - Design by afterthought never works

Security Kernel idea

- *Kernel holds enforcement mechanisms*
- *Kernel size is minimized*
- *Changes to kernel extremely rare*
- *Applications can trust the kernel*
  - Unix Kernel is a really bad example

The Kernel Domain idea

Kernel places all objects in domains

enforces access rights

Kernel does nothing else

File system is outside kernel

Page fault handlers are outside kernel

Device drivers are outside kernel

To the extent possible

Problems

Computer architectures

input/output unprotected

insufficient flexibility in MMU

Programmers and their traditions

programmers expect kernel rights!