APPLICATIONS OF LATTICES TO COMPUTER SECURITY

Catherine Meadows
Code 5543
Center for High Assurance Computer Systems
US Naval Research Laboratory
Washington, DC 20375
meadows@itd.nrl.navy.mil
http://chacs.nrl.navy.mil

OUTLINE OF TALK

- Motivation for use of lattices in access control
- Description of my own work in applying lattices to a sub-case of access control -- dynamic security policies
- Show how Millen applied to survivability
  - In the process, proved some new theorems on lattices and access control

RELATION OF LATTICS TO ACCESS CONTROL

- Access control -- saying who has access to what to do what
  - Closely related to set-theoretic lattices
  - If set A of users has set □ of permissions, and set B of users has set □ of permissions, then
    - A□ B has permissions □ □ □
    - A□ B has permissions □ □ □
  - Both access groups and permissions have lattice structure based on set inclusion
- Of particular interest -- multilevel security
  - Security levels (unclassified, secret, top secret, etc.) form a total order
  - Compartments form an unordered set
  - Cross-product of the two forms a lattice

DYNAMIC ACCESS CONTROL

- Access rights depend on data subject has accessed before
- Examples
  - Chinese Walls -- personnel working at a securities company may not be granted access to data on two companies determined to be in conflict of interest
    - If a subject has had access to data from one company, then is denied access to the other
    - Brewer and Nash formalized this policy in a 1989 paper
  - Aggregation problem -- data that may not be sensitive by itself may become so when combined with other data
    - Subject who has had access to data in an aggregation set may be denied access to other data in the set

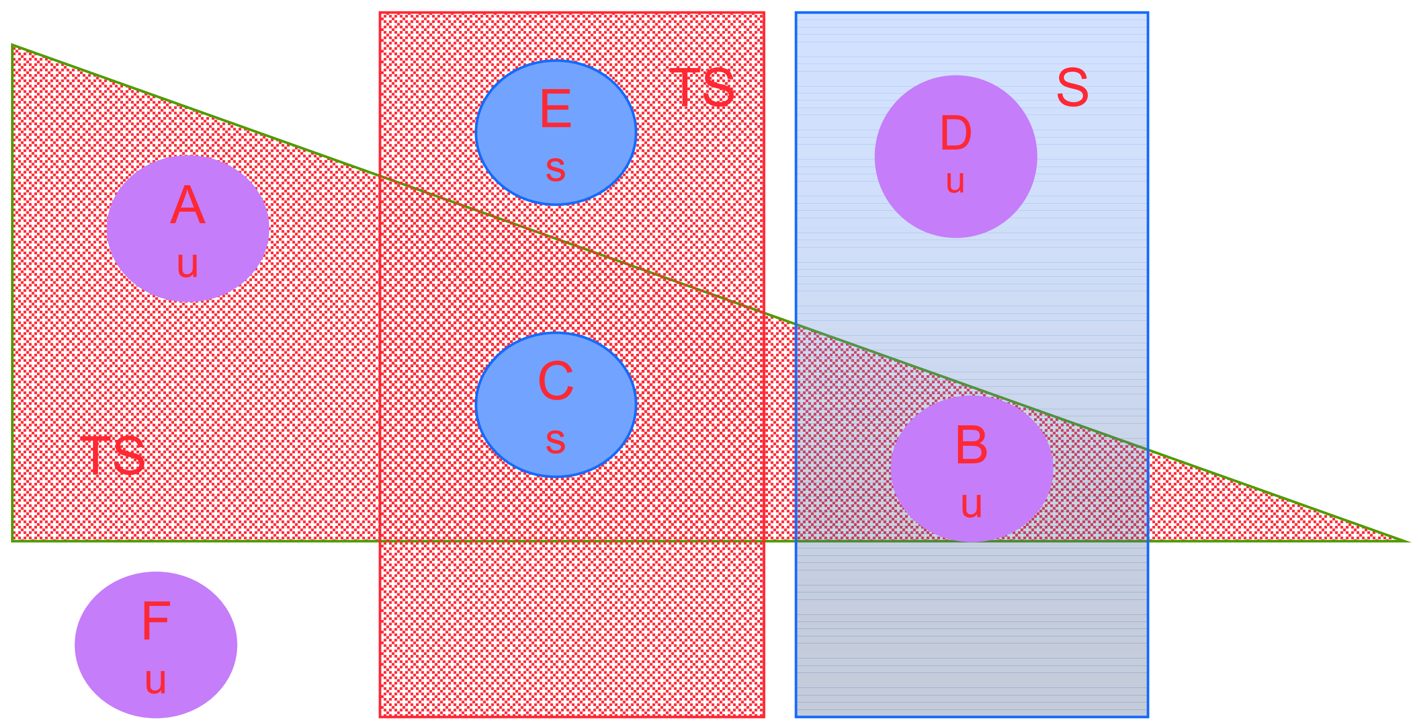
BASIS OF THE POLICY

- A collection of data and subjects, in which datum A and subject S assigned security levels l(A) and l(S)
  - l is a function from data and subjects to a lattice
  - If $l(S) \geq l(A)$ then S can read A
  - If $l(S) \leq l(A)$ then S can modify A
- However, in some cases, classification of a collection of data may be greater than that of any individual item in the collection

DEFINITION OF A DATASET AGGREGATION SYSTEM

- **A triple (D,L,*l*), where D is a set of pairwise disjoint datasets, L is a lattice, and *l* is a function from P(D) to L such that if H □J then *l*(H) ≤ *l*(J)**
  - **If level of H strictly dominates level of all subaggregates, call H an excepted aggregate**
  - **Otherwise, it's an unexcepted aggregate**
- **L is motivated by the lattice of security levels from multilevel security**

EXAMPLE



TS > S > U

DEFINING ACCESS CONTROL POLICIES

- Let (D,L,*l*) be a dataset aggregate system. An information flow policy is a transitive relation R on P(D) such that H□ K implies (H,K) □ R.
- We say that R is safe if
  - for all H and K such that (H,K) □ R, *l*(H) ≤ *l*(K)
  - For all H1, H2, and K such that (H1,K) □ R and (H2,K) □ R, (H1□ H2,K) □ R
- We define the multilevel information flow policy to be the relation R defined by (H,K) □ R if and only if, for each J, *l*(H □ J) ≤ *l*(K □ J)
- Intuitive idea: information flow policy says in what direction information can flow
  - If (H,K) □ R then information can flow from H to K

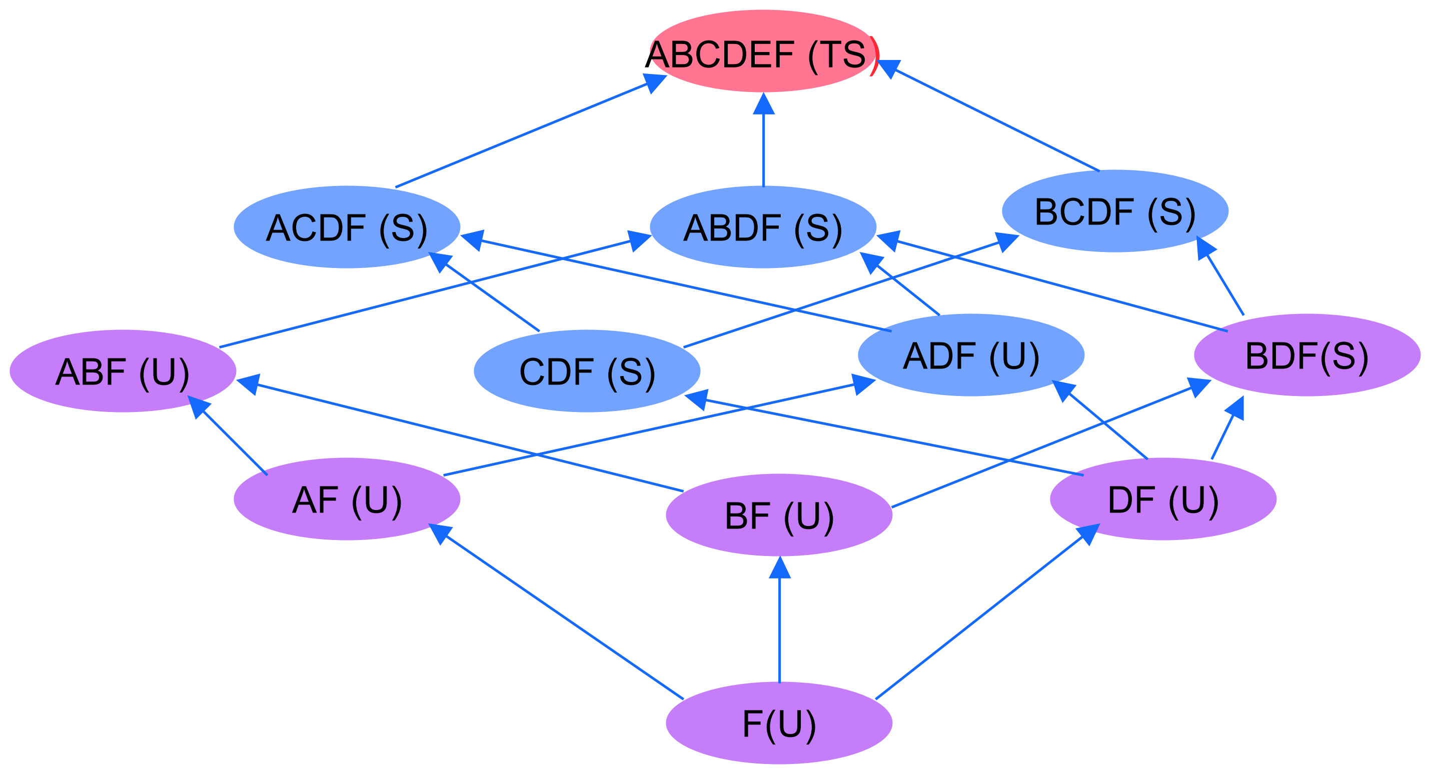
A THEOREM ON INFORMATION FLOW POLICIES

- Let (D,L,*l*) be a data aggregate system. Then the multilevel information flow policy on (D,L,*l*) is the unique maximal safe information flow policy on (D,L,*l*)

MAKING R INTO A LATTICE

- Take advantage of usual technique for transforming quasi-ordered set into a lattice
- Let (D,L,*l*) be a dataset aggregate system. Define g: P(D) -> P(D) by g(H) = {X □ D | ({X},H) □ R}
- Theorem: The collection of sets g(P(D)) together with the subset relation forms a lattice with
  - lub(H,K) = g(H□ K)
  - glb(H,K) = (H□ K)

EXAMPLE


ABCDEF (TS)
ACDF (S)
ABDF (S)
BCDF (S)
ABF (U)
CDF (S)
ADF (U)
BDF(S)
AF (U)
BF (U)
DF (U)
F(U)

MILLEN’S APPLICATION TO SURVIVABILITY

- Consider a system built out of a number of components
- Subsets of components can be configured to provide different sets of essential services
  - Components = datasets
  - Services = security levels

DEFINITION OF A SYSTEM

- A pair $S = (S_1,S_2)$ consisting of a set of services S_2 and a set of components S_1 is a system if there is a basis mapping s -> [s] defined on S_2 such that for all s □ S_2
  1. u □ [s] => u □ S_1, and;
  2. u,v □ [s] and u □ v => u = v
- A composition (subset of S_1) supports a service if and only if it contains a basis element for that service
- Define a survivability preordering
  – $s \preceq t$ means u supports s implies u supports t
  – Reflexive and transitive, but not anti-symmetric
  – However, does define a partial ordering on bases

DEFINITION OF STATE

- A state p of a system S is a pair $p = (p_1,p_2)$ such that
  1. p_2 □ S_2 is a set of services
  2. p_1 □ S_1 is a set of components called the support of p such that p_1 supports every s □ p_2.

Furthermore, there exists at least one function f on p_2 called a configuration of p such that

1. f(s) □ p_1
2. f(s) supports s

The configuration shows how each service is supported by p_1

REALIZABLE CONFIGURATIONS

- A configuration is realizable if it is possible to build a system that implements it
  - For example, it may not be possible to have a configuration in which the same component supports two different services
  - What is considered realizable may vary from system to system
- Let the set of realizable states of a system S be denoted by R
- Axioms
  - Adding components or deleting services does not destroy the realizability of a state
  - Disjoint configurations (in which no component supports more than one service) are always realizable

TRANSLATING INTO AGGREGATION PROBLEM

- Define composition "sensitivity level" as follows

$$\square_s(u) = \{p_2 \mid (u, p_2) \; \square \; R\}$$

- $\square_s(u)$ is monotone
- Theorem: Let $D = P(S_2)$ be the collection of sets of services. Then $(S_1, P(D), \square_s)$ is a dataset aggregate system

THEOREM ON SERVICE-PRESERVING TRANSITIONS

Def. A state transition is service-preserving if the new state supports all the services of the old state.

These two properties are equivalent:

P1. $\square_s(u)\ \square\ \square_s(v)$

P2. For all $p\ \square\ R$ **such that** $p_1 = u$ **there exists** $q\ \square\ R$ **such that** $q_1 = v$ **and** $p_2 = q_2$

P1 is the first of the two properties of a safe flow relation.

P2 says any state supported by u can be reconfigured to a state supported by v with a service-supporting transition

USING FLOW POLICIES TO INDUCE CONFIGURATION POLICIES

- Induced reconfiguration: If $\square_R$ is a flow policy with respect to $\square_S$ (as defined by Meadows), the induced reconfiguration policy $==>_R$ is defined by $p ==>_R q$ if $(p,q) \square R$ and $p_1 \square_R q_1$
- Corollary: Service-Preserving Configuration

Suppose that $\square_R$ is a safe flow policy. Then

1. Any reconfiguration $p ==>_R q$ is service-preserving.
2. If $p_1 \square_R v$ then there exists q such that $p_1 = v$ and $p ==>_R q$.

COMPARISON BETWEEN AGGREGATION AND RECONFIGURATION

AGGREGATION	RECONFIGURATION
DATASETS X	COMPONENTS S_1
AGGREGATES u □ X	COMPOSITIONS u □ S_1
SENSITIVITY LEVEL *l*	$□_s(u) = \{p_1 \vert p$ □ R and $p_2 = u\}$
FLOW POLICY □ $_R$	INDUCED RECONFIGURATION POLICY $==>_R$

MAXIMAL SAFE FLOW POLICY

- Define Maximal Safe Reconfiguration: if $\square_{R}$ is the maximal safe flow policy, then $==>_{R}$ is the maximal safe reconfiguration policy.
- Millen develops techniques for constructing maximal safe reconfiguration
  - Also apply to maximal safe flow policy
  - No complexity results, but best algorithm found is exponential time

CONCLUSION

- Some intriguing connections between aggregation in a secure database and policies for reconstructing survivable systems
- Follows general connection secrecy and integrity
  - Often can get from one to another by turning policy upside down
  - Connection is usually not trivial, need to think about how to apply results from one to problems of another
- Lattices, which have long been the backbone of the multilevel security model, can be applied in similar ways to other security problems

REFERENCES

D.F.C. Brewer and M. J. Nash, “The Chinese Wall Security Policy,” in Proceedings of the 1989 IEEE Symposium on Security and Privacy, pp. 206-214, IEEE Computer Society Press, May 1989.

C. Meadows, “Extending the Brewer-Nash Model to a Multi-Level Context,” in Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy, pp. 95-102, IEEE Computer Society Press, May 1990.

J. Millen, “Local Reconfiguration Policies,” In Proceedings of the 1999 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, May 1999.