Access Control in Communication Systems

K. Gopinath
IISc
• Access control model: guards control access to resources.

• Information flow model: classify information, prevent disclosure.
Motivation

- Any machine hosting some services on the net should not get totally compromised if there is a break-in.
- Can we isolate the breach to those services and not to affect rest of the system?
- Possible if we can use “mandatory access policies” (MAC) rather than the std “discretionary access policies” (DAC)
  - MAC: system decides how you share your objects
  - DAC: you decide how you share your objects
- “Play machines” on the net that allow root access but is considered reasonably well-protected! Try cable.coker.com.au as root with password “selinux”
Problems with inappropriate access control in the Netw Stack

- at appl level: sending email should be a privilege, not a right
- at transport level: synflood
- at L3 level: icmp flooding, DNS poisoning, ...
- at L2 level: wireless rogue APs, etc

Also, similar (and many!) problems with IPv6
  eg: Bogus Address Configuration Prefix Attacks

```
``If you think cryptography is the solution to your problem, you don't know what your problem is,'' Roger Needham"
Access Control

• Systems have employed basic access control since timesharing systems began ('60)
  – Multics, (Unix) rwxrwxrwx!

• Complex world-wide information systems, netw/storage subsystems, etc require much more sophisticated models
  – anonymous users/services, delegation, trust mgmt, scalability
  – need to have an integrated model of all authentication/authorization models: rwx, setuid, PAM, SELinux, cryptofs, X11 auth, NFS, ssh, httpd, IPSec, firewalls, iSCSI, ...
  – highly available access control: eg: clusters, SANs

• Need proof that info flow respects some security policies
  – model check?
Security Goals and Information Flow

• Info flow can be used to model security goals
  – confidentiality: to where can passwords flow from /etc/shadow?
  – integrity: from where can information flow to /etc/sshd.conf?
  – assured-pipelines: email spam and virus filtering proxy
  – domain separation: apache virtual hosts

• Assertions state info flow properties to model security properties
  – noflow: flows that should not be present
  – mustflow: flows that should be present
  – onlyflow: only flow that should be present
Brief History

- Simple access control models for “standalone” machines
  - Multics/Unix: universe of users known
    - scale of model “small”: need for automated analysis of policies \textit{small}
  - Theoretical models with “surprising” results (HRU)
    - unlimited subjects and objects
- Covert models of info flow:
  - work on proprietary OSes (eg. DG-UNIX) expensive and too late
  - showing that Trojan horses did not use covert channels to compromise info found to be “ultimately unachievable” (Guttman et al. '03)
- Overt models of info flow: Distributed systems +
  - Recent work with “easily” avlbl OSes expanding the scope
  - scale of model “large”: need for automated analysis of policies \textit{high}
• Once public-key crypto became possible
  – can deal with anonymous users with public keys: authentication/authorization possible
  – SDSI/SPKI, trust mgmt, ...
  – now can combine access control & crypto into a larger framework: logics for auth and access control
  • auth cert (K, S, D, T, V) can be viewed as an ACL entry, where keys or principals represented by the subject S are given permission to access resource K, T. (T: authorizations, D: delegation, V: Validity)
• However, many differing models
  – networking, storage, wireless, booting systems, ...
  – each subsystem itself sometimes has multiple models
    • eg: network: firewalls, IPSec, etc
End2end auth

• large systems need end2end auth
  – span adm domains
  – span netw scales
  – span levels of abstraction
  – span multiple protocols

• need a “certified” netw stack on each node
  – TCPA (TCG) models?

• need to integrate access control and crypto

Pressing need to have a “single model of representation” for analysing access control and info flow policies system-wide
Access Control Models

- **DAC model**
  - each subject decides how its objects interact with others
  - security mgr keeps access control matrix
  - checking safety problem: HRU undecidable
- **However, many decidable models exist:** eg: Take grant model
  - Num of subjects and objects fixed but can have some dynamicity such as conditional auth (based on state)
- **MAC model**
  - security server decides how any object interacts
- **RBAC model**
  - introduces roles
- **ABAC (attr based access control)**
  - rights based on attributes
Highly Available AC

- Graham-Denning Model: If a subject is deleted, the objects owned atomically transferred to its parent.
  - HRU cannot subsume this as it cannot loop over objs of sub
- HA AC: If a subject is deleted, the objects owned atomically transferred to its group
  - how are the objects shared? Need models (sep of duty?)
  - if subject is the active entity or leader, leader election also
  - access control has to be stopped till new leader; auth freeze
  - but does leader election require any AC?
    - yes, if it is netw, storage resources, election processing resources...
    - have to keep these outside purview of the freeze of AC
SSH

- On start, SSH daemon binds a socket to port 22 and waits for new cnxns.
- Every new connection handled by a forked child.
- Child needs to retain superuser privileges throughout its lifetime to create new pseudo terminals for the user, to authenticate key exchanges when cryptographic keys are replaced with new ones, to clean up pseudo terminals when the SSH session ends, to create a process with the privileges of the authenticated user, etc.

- With privilege separation
  - forked child acts as the monitor
  - monitor forks a slave that drops all its privileges and starts accepting data from the established connection.
  - monitor now waits for requests from the slave. If a request not permitted in the pre-authentication phase issued by the child, monitor terminates.

- Can model the monitor as an FSM
Privilege Separation
(from Provos et al., Preventing Privilege Escalation, USENIX Security Symp03)
SELinux Strict Policy

• A system where everything is denied by default. Must specify allow rules to grant privileges

• The policy rules only have allows, no denies.
  – “privacy safe”: removal of rule can only reduce privileges

• Minimal privileges for every daemon

• Separate user domains for programs like GPG, X, ssh, etc

• Default policy provided by NSA

• Difficult to enforce in general purpose OS.
  – Default/Off in Fedora Core 2; available in Fedora Core 3/4
Password mechanism

allow passwd_t shadow_t:file {read, write, append, ...}
domain transitions

1: allow user_t passwd_exec_t:file \{execute, \ldots\}

2: type_transition user_t passwd_exec_t:process passwd_t;
3: allow user_t passwd_t:process transition;
4: allow passwd_t passwd_exec_t:file entrypoint;
What is the problem?

• Very fine level control needed

• every major component such as NFS, X, ... needs extensive work on what permissions needs to be given before it can do its job
  – std assumption is “deny”

• As many as 30,000 rules!

• automated analysis needed
  – check if rules too lax? security problem
  – too strict? failure of program at runtime
    • need code coverage analysis
Extending SELinux across network

• Std SELinux
  - access a function of local subject and local object

• Across network
  - access a function of local subject and (remote object, remote node)
  - need network sid mapping
  - can use any framework
  - can use an attribute based access control framework
Network sid mapping

- each node has a local sid for each network context
- any external object whose security context is unknown:
  - local daemon contacts remote node daemon on a secure channel (such as a ssh cnxn)
  - remote node daemon sends credentials to local daemon
  - local daemon parses credentials and provides a local sid

Can now combine SELinux-type MAC and trust mgmt infrastructure using network sid mapping for an integrated model of authorization!
Trust mgmt

- Access control based on identity
- On Internet, no relationship between requestor and provider prior to request
- When users are unknown, need 3rd party input
  - trust, delegation and public keys
  - issuer authorizes specific permissions to specific principals
  - credentials can be signed by the issuer to avoid tampering
- eg: SDSI/SPKI:
  - credentials with delegation
  - assume locally generated public keys do not collide with other locally generated public keys elsewhere (!)
  - local namespaces
SPKI/SDSI

- name certificates: define the names available in an issuer's local name space
- authorization certificates: grant authorizations, or delegate the ability to grant authorizations
- certificate chain: provides proof that a client's public key is one of the keys that has been authorized to access a given resource either directly or transitively, via one or more name-definition or authorization-delegation steps.
- A set of SPKI/SDSI name and authorization certificates defines a pushdown system [Jha/Reps'03]
  - model checking in poly time
Queries in SPKI/SDSI (Clarke'01)

- **Authorized access**
  - Given resource R and principal K, is K authorized to access R?
  - Given resource R and name N (not necessarily a principal), is N authorized to access R?
  - Given resource R, what names (not necessarily principals) are authorized to access R?

- **Shared access**
  - For two given resources R1 and R2, what principals can access both R1 and R2?
  - For two given principals K1 and K2, what resources can be accessed by both K1 and K2?
• Compromisation assessment: due (solely) to the presence of maliciously or accidentally issued certificate set $C_0 \subseteq C$,
  – What resources could principal K have gained access to?
  – What principals could have gained access to resource R?

• Expiration vulnerability: if certificate set $C_0 \subseteq C$ expires,
  – What resources will principal K be prevented from accessing?
  – What principals will be excluded from accessing resource R?

• Universally guarded access
  – Is it the case that all authorizations that can be issued for a given resource R must involve a cert signed by principal K?
  – Is it the case that all authorizations that grant a given principal $K_0$ access to some resource must involve a cert signed by K?
What is STILL wrong?

• Permission-based trust mgmt cannot authorize principals with a certain property easily
  – eg. give 10% discount to students of a particular institute
    • book store delegates discount permission to the inst key
      – inst has to delegate its key to each student wrt “bookstore” context
      – too much burden on inst
    • inst creates a new group key for students and delegates it to each student key
      – inst has to create a key for each meaningful group; too much burden again!

• Attribute-based approach: combines RBAC & trust mgmt
Delegation Logic for ABAC

- logic programming + delegation + monotonic/non-monotonic reasoning
- delegation depth and complex principals
  - $k$ out of $n$ threshold (static/dynamic)
Conclusions

• Newer models such as ABAC need systematic investigation
  – currently, focus on RBAC

• Analysis of access control and info flow critical for large comm systems
  – Tool support

• Integrating different models of access control requires work
Bogus Address Configuration Prefix Attacks

- An attacking node can send a Router Advertisement message specifying an invalid subnet prefix to be used by a host for address autoconfiguration.
- A host executing the address autoconfiguration algorithm uses the advertised prefix to construct an address, even though that address is not valid for the subnet.
- As a result, return packets never reach the host because the host’s source address is invalid.
- This is a DoS attack.
- This attack has the potential to propagate beyond the immediate attacked host if the attacked host performs a dynamic update to the DNS based on the bogus constructed address.
Bogus Address Configuration
Prefix Attacks (contd.)

- DNS update causes the bogus address to be added to the host’s address record in the DNS. Should this occur, applications performing name resolution through the DNS obtain the bogus address and an attempt to contact the host fails.
  - However, well-written applications will fall back and try the other addresses registered in DNS, which may be correct.

- A distributed attacker can make the attack more severe by creating a falsified reverse DNS entry that matches with the dynamic DNS entry created by the target.

- Consider an attacker who has legitimate access to a prefix <ATTACK_PRFX>, and a target who has an interface ID <TARGET_IID> The attacker creates a reverse DNS entry for <ATTACK_PRFX>:<TARGET_IID>, pointing to the real domain name of the target, e.g., "secure.target.com".
General Reqs of Access Control

- Conditional auth
- Expressibility of joint action
- Delgation mechanisms
- Support for open and closed systems
- Expressibility of adm policies
- Avoidance of root bottleneck
- Support for fine- and coarse-grained specs
HRU Primitive Operations

- **create subject** \(s\);  
  - Creates new row, column in access control matrix (ACM);
- **create object** \(o\)  
  - creates new column in ACM
- **destroy subject** \(s\);  
  - Deletes row, column from ACM;
- **destroy object** \(o\)  
  - deletes column from ACM
- **enter** \(r\) **into** \(A[s, o]\)  
  - Adds \(r\) rights for subject \(s\) over object \(o\)
- **delete** \(r\) **from** \(A[s, o]\)  
  - Removes \(r\) rights from subject \(s\) over object \(o\)
Safety Question

• Adding a generic right $r$ where there was not one is “leaking”

• If a system $S$, beginning in initial state $s_0$, cannot leak right $r$, it is safe with respect to the right $r$.

• Does there exist an algorithm for determining whether a protection system $S$ with initial state $s_0$ is safe with respect to a generic right $r$?

General Case: leakage of a right undecidable: can reduce halting problem to safety problem (HRU'75):

• Mainly due to arbitrary changes to the sets of subjects and objects
Graham-Denning DAC model

- every object must be owned by at least one subject
- objects are not controlled, only subjects are
- a special subject $u$ exists that is not owned by any subject and is not controlled by any other subject
- a subject other than $u$ is owned by exactly one other subject
- every subject controls itself
- a subject other than $u$ is controlled by at most one other subject
- there exists no set of subjects such that they form a cycle in terms of ownership of each other (and in particular, a subject does not own itself)
State change rules

- grant right r or r* by an initiator that has right r* over obj (* indicates copy flag)
- transfer ownership over obj from i to s
- grant the right r or r* over o by the owner of o
- grant the control right over o by its owner.
- grant the own right over o.
- delete a right r or r* a subject has over o.
- create or destroy an object that is not a subject.
- create or destroy a subject

Safety efficiently decidable (cubic in size of size of start state and the set of rights in the system) [MTNL'05]
Expressive Power of Access Control Models

• HRU cannot subsume Graham-Denning
  – removal of a subject who has the ownership of some objects requires the transfer of ownership to some other subject (often times the owner of the subject being removed)
  – both the removal of the subject and the transfer of ownership of objects it owns occur in a single state-change.
  – a single HRU command cannot capture these features, because it cannot loop over all objects owned by a subject.

• RBAC (eg. ARBAC97) is not more expressive than DAC

• ATAM (Augmented Typed Access Matrix) is more expressive than TAM (Typed Access Matrix)
Model Checking Access Control

• We will use SELinux as an example

• Overt vs. Covert Information
  – Overt flows transfer data directly
  – often high-bandwidth & easily controllable by the policy

• Covert flows indirect; e.g., file existence or CPU usage
  – often low-bandwidth & difficult to control in SELinux

• Our focus: overt flows as defined by the user in order to understand security properties
  – Direct Information Flow: allow subject_t object_t:file write
  – Transitive: allow subject_a_t object_t : file write;
    allow subject_b_t object_t : file read;
Reachability Analysis

- Treat certs as rewrite rules:
  - name cert \((K, A, S, V)\) rewritten as \(K A \rightarrow S\)
  - auth cert \((K, S, D, T, V)\) rewritten as \(K_T \triangleright \rightarrow S \triangleright\) if delegation bit is on
    else \(K_T \triangleright \rightarrow S\)

- Given a set of certificates \(C\), its closure (denoted by \(C^*\)) is the smallest set of certificates that includes \(C\) and is closed under rule composition; reachable keys = \(C^* \cap \mathcal{K}\)
  - \(C^*\) in general infinite (eg. \(C = \{(K A \rightarrow K A A)\}; C^* = \{(K A \rightarrow K A^i)\}\))
• However, considering only reducing certs: $K \xrightarrow{A} K'$, the name-reduction closure $C#$ of a set of certificates $C$ can be defined as the smallest set of certificates that contains $C$ and is closed under name reduction (one rule is reducing)
  
  - reachable keys = $C# \cap \mathcal{K}$
Pushdown systems

similar to PDAs but do not have an input alphabet
locations, stack (and its alphabet), transitions
not lang recognizers but can specify infinite-state transition sys
Pushdown systems

- configuration $C$ of a PDS: (control loc, stack contents $\Gamma^*$)

- $\text{pre}^*(C)$, $\text{post}^*(C)$: regular
  - configuration automaton for $\text{pre}^*(C)$, $\text{post}^*(C)$ constructible in poly time

- Can map (control loc, v) of a PDS to a set of atomic propositions
  - extend mapping from (control loc, vw) to same
  - model check given config $c$ and LTL formula $\phi$: $c \models \phi$
Mapping SPKI/SDSI to PDSs (Jha/Reps)

- set of locs: each key and resource
- stack alphabet: identifiers and delegation on/off symbol
- transition rules: name certs and authorization certs (viewed as rewriting rules) correspond to transition rules
Certificate set analysis probs

- Given resource R and principal K, is K authorized to access R?
  - \( (R, \triangleright) \in \text{pre}^*\{( (K, \triangleright), (K, \triangleright) \} \)

- Is it the case that all authorizations that can be issued for a given resource R must involve a cert signed by principal K?
  - \( (R, \triangleright) \) satisfies a LTL formula \( \Box (\neg S \cup (Q \lor \Box \neg S) \) where Q in all ground configs that involve loc K and S in all ground configs K', \( \triangleright \) and K', \( \triangleright \) and K' in \( \mathcal{K} \)
combines RBAC and Trust mgmt

RT[]:
- 1: Simple Member: \( A.r \leftarrow D \)
  - A asserts that D is a member of A's r role.
- 2: Simple Inclusion: \( A.r \leftarrow B.r1 \)
  - delegation from A to B

RT[\cap]: RT[] plus
- 3: Intersection Inclusion: \( A.r \leftarrow B1.r1 \) and \( B2.r2 \)
  - partial delegations from A to B1 and to B2

RT[\leftrightarrow]: RT[] plus
- 4: Linking Inclusion: \( A.r \leftarrow A.r1.r2 \)
  - delegation from A to all the members of the role A.r1

RT[\leftrightarrow, \cap]: RT[] plus 3, 4
- Simple safety (Existential): does a principal have access to some resource in some reachable state?
- Simple availability: in every state, does some principal have access to some resource?
- Bounded safety: in every state, is the number of principals that have access to some resource bounded?
- Liveness (Existential): Does there exist a reachable state in which no principal has access to a given resource?
- Mutual Exclusion: In every reachable state, are 2 given properties (or two given resources) mutually exclusive, i.e., no principal has both properties (or access to both resources) at the same time?
- Containment: In every reachable state, does every principal that has one property (e.g., has access to a resource) also have another property (e.g., is an employee)? Containment can express safety or availability (e.g., by interchanging the two example properties in the previous sentence).
Complexity

- Simple safety, simple availability, bounded safety, liveness, and mutual exclusion analysis for RT[\(\leftrightarrow, \cap\)] decidable in poly time in size of state

- Containment analysis:
  - P for RT[]
  - coNP-complete for RT[\(\cap\)],
  - PSPACE-complete for RT[\(\leftrightarrow\)], and
  - decidable in coNEXP for RT[\(\leftrightarrow, \cap\)]
Reqs in an attribute-based system

- decentralized
- delegation of attr authority
- inference
- attribute-based delegation of attribute authority
- conjunction of attr
- attr with fields (expiry, age, ...)
- expressive power, declarative semantics, tractable compliance checking
• Reliability
• Prob CTL?
• Model checking using lang inclusion?
IPv6 problems
SIP

- Java dload
- SIMPLE
-