ECS 235B, Lecture 5

January 16, 2019

Security Properties

- Question: given two models, do they have the same security properties?
  - First comes theory
  - Then comes an example comparison
- Basic idea: view access request as query asking if subject has right to perform action on object

Alternate Definition of "Scheme"

- Σ set of states
- Q set of queries
- $e: \Sigma \times Q \rightarrow \{\mathit{true}, \mathit{false}\}$
  - Called *entailment relation*
- T set of state transition rules
- (Σ, Q, e, T) is an *access control scheme*

Alternate Definition of "Scheme"

- *s* tries to access *o*
  - Corresponds to query $q \in Q$
- If state $\sigma \in \Sigma$ allows access, then $e(\sigma, q) = \textit{true}$; otherwise, $e(\sigma, q) = \textit{false}$
- Write change of state from σ_0 to σ_1 as $\sigma_0 \mapsto \sigma_1$
  - Emphasizing we're looking at *permissions*
  - Multiple transitions are $\sigma_0 \mapsto_\tau^* \sigma_n$
    - Σ_n said to be *τ-reachable from* σ_0

Example: Take-Grant

- Σ set of all possible protection graphs
- Q set of queries

  $\{\ \mathit{can{\bullet}share}(\alpha, \mathbf{v}_1, \mathbf{v}_2, G_0) \mid \alpha \in R, \mathbf{v}_1, \mathbf{v}_2 \in G_0\ \}$
- $e(\sigma_0, q)$ = *true* if q holds; $e(\sigma_0, q)$ = *false* if not
- T set of sequences of take, grant, create, remove rules

Security Analysis Instance

- Let (Σ, Q, e, T) be an *access control scheme*
- Tuple (σ, q, τ, Π) is *security analysis instance*, where:
  - $\sigma \in \Sigma$
  - $q \in Q$
  - $\tau \in T$
  - Π is $\forall$ or $\exists$
- If Π is $\exists$, *existential* security analysis
  - Is there a state σ' such that $\sigma \mapsto_\tau^* \sigma'$, $e(\sigma', q)$ = *true*?
- If Π is $\forall$, *universal* security analysis
  - For all states σ' such that $\sigma \mapsto_\tau^* \sigma'$, is $e(\sigma', q)$ = *true*?

Example: Take-Grant

- $\sigma_0 = G_0$
- q is $can\bullet share(r, \mathbf{v}_1, \mathbf{v}_2, G_0)$
- τ is sequence of take-grant rules
- Π is $\exists$
- Security analysis instance examines whether $\mathbf{v}_1$ has r rights over $\mathbf{v}_2$ in graph with initial state G_0
- So safety question is security analysis instance

Comparing Two Models

- Each query in A corresponds to a query in B
- Each (state, state transition) in A corresponds to (state, state transition) in B

Formally:

- $A = (\Sigma^A, Q^A, e^A, T^A)$ and $B = (\Sigma^B, Q^B, e^B, T^B)$
- *mapping* from A to B is:
  - $f : (\Sigma^A \times T^A) \cup Q^A \rightarrow (\Sigma^B \times T^B) \cup Q^B$

Image of Instance

- f mapping from A to B
- *image of a security analysis instance* $(\sigma^A, q^A, \tau^A, \Pi)$ under f is $(\sigma^B, q^B, \tau^B, \Pi)$, where:
  - $f((\sigma^A, \tau^A)) = (\sigma^B, \tau^B)$
  - $f(q^A) = q^B$
- f is *security-preserving* if every security analysis instance in A is true iff its image is true

Composition of Queries

- Let (Σ, *Q*, *e*, *T*) be an *access control scheme*
- Tuple (σ, *φ*, τ, Π) is compositional *security analysis instance*, where *φ* is propositional logic formula of queries from *Q*
- *image of compositional security analysis instance* defined similarly to previous
- *f* is *strongly security-preserving* if every compositional security analysis instance in *A* is true iff its image is true

State-Matching Reduction

- $A = (\Sigma^A, Q^A, e^A, T^A)$, $B = (\Sigma^B, Q^B, e^B, T^B)$, f mapping from A to B
- σ^A, σ^B equivalent under the mapping f when
  - $e^A(\sigma^A, q^A) = e^B(\sigma^B, q^B)$
- f state-matching reduction if for all $\sigma^A \in S^A$, $\tau^A \in T^A$,

  $(\sigma^B, \tau^B) = f((\sigma^A, \tau^A))$ has the following properties:

Property 1

- For every state σ'^A in scheme *A* such that $\sigma^A \mapsto_\tau^* \sigma'^A$, there is a state σ'^B in scheme *B* such that $\sigma^B \mapsto_\tau^* \sigma'^B$, and σ'^A and σ'^B are equivalent under the mapping *f*
  - That is, for every reachable state in *A*, a matching state in *B* gives the same answer for every query

Property 2

- For every state σ'^B in scheme *B* such that $\sigma^B \mapsto_\tau^* \sigma'^B$, there is a state σ'^A in scheme *A* such that $\sigma^A \mapsto_\tau^* \sigma'^A$, and σ'^A and σ'^B are equivalent under the mapping *f*
  - That is, for every reachable state in *B*, a matching state in *A* gives the same answer for every query

Theorem

Mapping *f* from scheme *A* to *B* is strongly security-preserving iff *f* is a state-matching reduction

Proof ($\Rightarrow$)

- Must show (σ^A, φ^A, τ^A, Π) true iff (σ^B, φ^B, τ^B, Π) true
- Π is $\exists$: assume τ^A-reachable state σ'^A from σ^A in which φ^A true
  - By property 1, there is a state σ'^B corresponding to σ'^A in which φ^B holds
- Π is $\forall$: assume τ^A-reachable state σ'^A from σ^A in which φ^A false
  - By property 1, there is a state σ'^B corresponding to σ'^A in which φ^B false
- Same for φ^B with τ^B-reachable state σ'^B from σ^B
- So (σ^A, φ^A, τ^A, Π) true iff (σ^B, φ^B, τ^B, Π) true

Proof ($\Longleftarrow$)

- Let f be map from A to B but not state-matching reduction. Then there are $\sigma^A \in S^A$, $\tau^A \in T^A$, $(\sigma^B, \tau^B) = f((\sigma^A, \tau^A))$ violating at least one of the properties
- Assume it's property 1; σ^A, σ^B corresponding states. There is a τ^A-reachable state σ'^A from σ^A such that no τ^B-reachable state from σ^B is equivalent to σ'^B
- Generate φ^A and φ^B such that the existential compositional security analysis in A is true but in B is false
  - To do this, look at each $q^A \in Q^A$
  - If $e(\sigma'^A, q^A) = true$, conjoin q^A to φ^A; otherwise, conjoin $\neg q^A$ to φ^A
  - Then $e(\sigma'^A, q^A) = true$ but for $\varphi^B = f(\varphi^A)$ and all states σ'^B that are τ^B-reachable from σ^B, $e(\sigma'^B, q^B) = false$
- Thus, f is not strongly security-preserving
- Argument for property 2 is similar

Expressive Power

If access control model *MA* has a scheme that cannot be mapped into a scheme in access control model *MB* using a state-matching reduction, then model *MB* is *less expressive than* model *MA*.

If every scheme in model *MA* can be mapped into a scheme in model *MB* using a state-matching reduction, then model *MB* is *as expressive as* model *MA*.

If *MA* is as expressive as *MB*, and MB is as expressive as *MA*, the models are *equivalent*

- Note this does not assume monotonicity, unlike earlier definition

Augmented Typed Access Control Matrix

- Add a test for the *absence* of rights to TAM

```
command add•right(s:u, o:v)
     if own in a[s,o] and r not in a[s,o]
     then
          enter r into a[s,o]
end
```

- How does this affect the answer to the safety question?

Safety Question

- ATAM can be mapped onto TAM
- But will the mapping, or any such mapping, preserve security properties?
- Approach: consider TAM as an access control model

TAM as Access Control Model

- S set of subjects; S_σ subjects in state σ
- O set of objects; O_σ objects in state σ
- R set of rights; R_σ rights in state σ
- T set of types; T_σ subjects in state σ
- $t : S_\sigma \cup O_\sigma \longrightarrow T_\sigma$ gives type of any subject or object
- State σ defined as (S_σ, O_σ, R_σ, T_σ, t)
- In TAM, query is of form "is $r \in a[s,o]$", and $e(s, r \in a[s,o])$ true iff $s \in S_\sigma$, $o \in O_\sigma$, $r \in R_\sigma$, $r \in a_\sigma[s,o]$ are true

ATAM as Access Control Model

Same as TAM with one addition:

- ATAM also allows queries of form "is $r \notin a[s,o]$", and $e(s, r \notin a[s,o])$ true iff $s \in S_\sigma$, $o \in O_\sigma$, $r \in R_\sigma$, $r \notin a_\sigma[s,o]$ are true

Theorem

A state-matching reduction from ATAM to Tam does not exist.

Outline of proof: by contradiction

- Consider two state transitions, one that creates subject and one that adds right *r* to an element of the matrix
- Can determine an upper bound on the number of answers to TAM query a command can change; depends on state and commands

Proof

- Assume *f* is state-matching reduction from ATAM to TAM
- Consider simple ATAM scheme:
  - Initial state σ_0 has no subjects, objects
  - All entities have type *t*
  - Only one right *r*
  - Query $q_{ij} = r \in a[s,o]$; query $\underline{q_{ij}} = r \notin a[s,o]$
  - 2 state transition rules
    - *make•subj*(*s* : *t*) creates subject *s* of type *t*
    - *add•right*(*x* : *t*, *y* : *t*) adds right *r* to *a*[*x*, *y*]

Proof

- TAM: superscript T represents components of that system
  - So initial state is $\sigma_0{}^T = f(\sigma_0)$, transitions are $\tau^T = f(\tau)$
- By definition of state-matching reduction, how f maps queries does not depend on initial state or state transitions of a model
- Let p, q be queries in ATAM and p^T, q^T the corresponding queries in TAM; if $p \neq q$, then $p^T \neq q^T$
- As commands in TAM execute, they can change the value (response) of q_{ij}
- Upper bound on the number of values of queries a single command can change is m (number of **enter** or **add•right** operations)

Proof

- Choose $n > m$
- In ATAM, construct state σ_k such that:
  - $\sigma_0 \longrightarrow^* \sigma_k$; and
  - $e(\sigma_k, \neg q_{1,1} \wedge \underline{q_{1,1}} \wedge \ldots \wedge \neg q_{n,n} \wedge \underline{q_{n,n}})$ is true
- So $e(\sigma_k, q_{i,j})$ is false, $e(\sigma_k, \underline{q_{i,j}})$ is true for all $1 \leq i, j \leq n$
- As f is a state-matching reduction, there is a state σ_k^T in TAM that causes the corresponding queries to be answered the same way
- Consider $\sigma_0^T \longrightarrow \sigma_1^T \longrightarrow \ldots \longrightarrow \sigma_k^T$; choose first state σ_C^T such that $e(\sigma_C^T, q_{i,j}^T \vee \underline{q_{i,j}^T})$ is true for all $1 \leq i, j \leq n$

Proof

- In $\sigma_{C\text{-}1}{}^{T}$, $e(\sigma_{\mathrm{C\text{-}1}}{}^{\mathrm{T}}, q_{v,w}{}^{T} \vee \underline{q_{v,w}{}^{T}})$ is false for some $1 \leq v, w \leq n$, so $e(\sigma_{\mathrm{C\text{-}1}}{}^{T}, \neg q_{v,w}{}^{T} \wedge \neg\underline{q_{v,w}{}^{T}})$ is true
- State σ in ATAM for which $e(\sigma, \neg q_{v,w} \wedge \neg\underline{q_{v,w}})$ is true is one in which either s_v or s_w or both does not exist
- Thus in that state, one of the following 2 queries holds:
  - $Q_1 = \neg q_{v,1} \wedge \neg\underline{q_{v,1}} \wedge \ldots \wedge \neg q_{n,v} \wedge \neg\underline{q_{n,v}}$
  - $Q_1 = \neg q_{w,1} \wedge \neg\underline{q_{w,1}} \wedge \ldots \wedge \neg q_{n,w} \wedge \neg\underline{q_{n,w}}$
- So in TAM, $e(\sigma_{\mathrm{C\text{-}1}}{}^{T}, \mathrm{Q_1}^{\mathrm{T}} \wedge \mathrm{Q_2}^{T})$ is true

Proof

- Now consider the transition from σ_{C-1}^{T} to σ_{C}^{T}
- Values of at least *n* queries in Q_1 or Q_2 must change from false to true
- But each command can change at most $m < n$ queries
- This is a contradiction
- So no such *f* can exist, proving the result

Thus, ATAM can express security properties that TAM cannot

Key Points

- Safety problem undecidable
- Limiting scope of systems can make problem decidable
- Types critical to safety problem's analysis

Security Policies

- Policies
- Trust
- Nature of Security Mechanisms
- Policy Expression Languages
- Limits on Secure and Precise Mechanisms

Security Policy

- Policy partitions system states into:
  - Authorized (secure)
    - These are states the system can enter
  - Unauthorized (nonsecure)
    - If the system enters any of these states, it's a security violation
- Secure system
  - Starts in authorized state
  - Never enters unauthorized state

Confidentiality

- X set of entities, I information
- I has the *confidentiality* property with respect to X if no $x \in X$ can obtain information from I
- I can be disclosed to others
- Example:
  - X set of students
  - I final exam answer key
  - I is confidential with respect to X if students cannot obtain final exam answer key

Integrity

- *X* set of entities, *I* information
- *I* has the *integrity* property with respect to *X* if all $x \in X$ trust information in *I*
- Types of integrity:
  - Trust *I*, its conveyance and protection (data integrity)
  - *I* information about origin of something or an identity (origin integrity, authentication)
  - *I* resource: means resource functions as it should (assurance)

Availability

- X set of entities, I resource
- I has the *availability* property with respect to X if all $x \in X$ can access I
- Types of availability:
  - Traditional: x gets access or not
  - Quality of service: promised a level of access (for example, a specific level of bandwidth); x meets it or not, even though some access is achieved

Policy Models

- Abstract description of a policy or class of policies
- Focus on points of interest in policies
  - Security levels in multilevel security models
  - Separation of duty in Clark-Wilson model
  - Conflict of interest in Chinese Wall model

Mechanisms

- Entity or procedure that enforces some part of the security policy
  - Access controls (like bits to prevent someone from reading a homework file)
  - Disallowing people from bringing CDs and floppy disks into a computer facility to control what is placed on systems

Question

- Policy disallows cheating
  - Includes copying homework, with or without permission
- CS class has students do homework on computer
- Anne forgets to read-protect her homework file
- Bill copies it
- Who breached security?
  - Anne, Bill, or both?

Answer Part 1

- Bill clearly breached security
  - Policy forbids copying homework assignment
  - Bill did it
  - System entered unauthorized state (Bill having a copy of Anne's assignment)
- If not explicit in computer security policy, certainly implicit
  - Not credible that a unit of the university allows something that the university as a whole forbids, unless the unit explicitly says so

Answer Part #2

- Anne didn’t protect her homework
  - Not required by security policy
- She didn’t breach security
- If policy said students had to read-protect homework files, then Anne did breach security
  - She didn’t do this

Types of Security Policies

- Military (governmental) security policy
  - Policy primarily protecting confidentiality
- Commercial security policy
  - Policy primarily protecting integrity
- Confidentiality policy
  - Policy protecting only confidentiality
- Integrity policy
  - Policy protecting only integrity

Integrity and Transactions

- Begin in consistent state
  - "Consistent" defined by specification
- Perform series of actions (*transaction*)
  - Actions cannot be interrupted
  - If actions complete, system in consistent state
  - If actions do not complete, system reverts to a consistent state