January 29, 2014

- Bell-LaPadula
  - Example Instantiation
- Tranquility
- Controversy
  - System Z

Rules, States, and Conditions

Let ρ be a rule and $\rho(r, v) = (d, v')$, where $v = (b, m, f, h)$ and $v' = (b', m', f', h')$. Then:

1. If $b \subseteq b'$, $f = f'$, and v satisfies the simple security condition, then v' satisfies the simple security condition
2. If $b \subseteq b'$, $f = f'$, and v satisfies the *-property, then v' satisfies the *-property
3. If $b \subseteq b'$, $m[s, o] \subseteq m'[s, o]$ for all $s \in S$ and $o \in O$, and v satisfies the ds-property, then v' satisfies the ds-property

Example Instantiation: Multics

- 11 rules affect rights:
  - set to request, release access
  - set to give, remove access to different subject
  - set to create, reclassify objects
  - set to remove objects
  - set to change subject security level
- Set of "trusted" subjects $S_T \subseteq S$
  - *-property not enforced; subjects trusted not to violate
- $\Delta(\rho)$ domain
  - determines if components of request are valid

get-read Rule

- Request $r = (get, s, o, \underline{\text{r}})$
  - s gets (requests) the right to read o
- Rule is $\rho_1(r, v)$:

if $(r \neq \Delta(\rho_1))$ **then** $\rho_1(r, v) = (\underline{\text{i}}, v)$;

else if $(f_s(s)\ dom\ f_o(o)$ **and** $[s \in S_T$ **or** $f_c(s)\ dom\ f_o(o)]$

and $r \in m[s, o])$

then $\rho_1(r, v) = (y, (b \cup \{\ (s, o, \underline{\text{r}})\ \}, m, f, h))$;

else $\rho_1(r, v) = (\underline{\text{n}}, v)$;

Security of Rule

- The get-read rule preserves the simple security condition, the *-property, and the ds-property
  - Proof
    - Let v satisfy all conditions. Let $\rho_1(r, v) = (d, v')$. If $v' = v$, result is trivial. So let $v' = (b \cup \{ (s_2, o, \underline{r}) \}, m, f, h)$.

Proof

- Consider the simple security condition.
  - From the choice of v', either $b' - b = \varnothing$ or $\{ (s_2, o, \underline{r}) \}$
  - If $b' - b = \varnothing$, then $\{ (s_2, o, \underline{r}) \} \in b$, so $v = v'$, proving that v' satisfies the simple security condition.
  - If $b' - b = \{ (s_2, o, \underline{r}) \}$, because the *get-read* rule requires that $f_c(s)$ *dom* $f_o(o)$, an earlier result says that v' satisfies the simple security condition.

Proof

- Consider the *-property.
  - Either $s_2 \in S_T$ or $f_c(s)$ *dom* $f_o(o)$ from the definition of *get-read*
  - If $s_2 \in S_T$, then s_2 is trusted, so *-property holds by definition of trusted and S_T.
  - If $f_c(s)$ *dom* $f_o(o)$, an earlier result says that v' satisfies the simple security condition.

Proof

- Consider the discretionary security property.
  - Conditions in the *get-read* rule require $\underline{r} \in m[s, o]$ and either $b' - b = \varnothing$ or $\{ (s_2, o, \underline{r}) \}$
  - If $b' - b = \varnothing$, then $\{ (s_2, o, \underline{r}) \} \in b$, so $v = v'$, proving that v' satisfies the simple security condition.
  - If $b' - b = \{ (s_2, o, \underline{r}) \}$, then $\{ (s_2, o, \underline{r}) \} \notin b$, an earlier result says that v' satisfies the ds-property.

give-read Rule

- Request $r = (s_1, give, s_2, o, \underline{r})$
  - s_1 gives (request to give) s_2 the (discretionary) right to read o
  - Rule: can be done if giver can alter parent of object
    - If object or parent is root of hierarchy, special authorization required
- Useful definitions
  - $root(o)$: root object of hierarchy h containing o
  - $parent(o)$: parent of o in h (so $o \in h(parent(o))$)
  - $canallow(s, o, v)$: s specially authorized to grant access when object or parent of object is root of hierarchy
  - $m \wedge m[s, o] \leftarrow \underline{r}$: access control matrix m with $\underline{r}$ added to $m[s, o]$

give-read Rule

- Rule is $\rho_6(r, v)$:

  if $(r \neq \Delta(\rho_6))$ **then** $\rho_6(r, v) = (\underline{\mathrm{i}}, v)$;

  else if $([o \neq root(o)$ **and** $parent(o) \neq root(o)$ **and** $parent(o) \in b(s_1{:}\underline{\mathrm{w}})]$ **or**

  $[parent(o) = root(o)$ **and** $canallow(s_1, o, v)\ ]$ **or**

  $[o = root(o)$ and $canallow(s_1, o, v)\ ])$

  then $\rho_6(r, v) = (y, (b, m \wedge m[s_2, o] \leftarrow \underline{\mathrm{r}}, f, h))$;

  else $\rho_1(r, v) = (\underline{\mathrm{n}}, v)$;

Security of Rule

- The *give-read* rule preserves the simple security condition, the *-property, and the ds-property
  - Proof: Let v satisfy all conditions. Let $\rho_1(r, v) = (d, v')$. If $v' = v$, result is trivial. So let $v' = (b, m[s_2, o] \leftarrow \underline{r}, f, h)$. So $b' = b, f' = f, m'[x, y] = m[x, y]$ for all $x \in S$ and $y \in O$ such that $x \neq s$ and $y \neq o$, and $m[s, o] \subseteq m'[s, o]$. Then by earlier result, v' satisfies the simple security condition, the *-property, and the ds-property.

Principle of Tranquility

- Raising object's security level
  - Information once available to some subjects is no longer available
  - Usually assume information has already been accessed, so this does nothing
- Lowering object's security level
  - The *declassification problem*
  - Essentially, a "write down" violating *-property
  - Solution: define set of trusted subjects that *sanitize* or remove sensitive information before security level lowered

Types of Tranquility

- Strong Tranquility
  - The clearances of subjects, and the classifications of objects, do not change during the lifetime of the system
- Weak Tranquility
  - The clearances of subjects, and the classifications of objects, do not change in a way that violates the simple security condition or the *-property during the lifetime of the system

Example of Weak Tranquility

- Only one subject at TOP SECRET
- Document at CONFIDENTIAL
- New CONFIDENTIAL user to be added
  - User should not see document
- Raise document to SECRET
  - Subject still cannot write document
  - All security relationships unchanged

Declassification

- Lowering the security level of a document
  - Direct violation of the "no writes down" rule
  - May be necessary for legal or other purposes
- Declassification policy
  - Part of security policy covering this
  - Here, "secure" means classification changes to a lower level in accordance with declassification policy

Principles

- Principle of Semantic Consistency
- Principle of Occlusion
- Principle of Conservativity
- Principle of Monotonicity of Release

Principle of Semantic Consistency

- As long as the semantics of the parts of the system not involved in the declassification do not change, those parts may be changed without affecting system security
  - No leaking due to semantic incompatibilities
  - *Delimited release*: allow declassification, release of information only through specific channels ("escape hatches")

Principle of Occlusion

- Declassification mechanism cannot conceal *improper* lowering of security levels
  - Robust declassification property: attacker cannot use escape hatches to obtain information unless it is properly declassified

Other Principles

- Principle of Conservativity
  - Absent declassification, system is secure
- Principle of Monotonicity of Release
  - When declassification is performed in an authorized manner by authorized subjects, the system remains secure

Idea: declassifying information in accordance with declassification policy does not affect security

Controversy

- McLean:
  - "value of the BST is much overrated since there is a great deal more to security than it captures. Further, what is captured by the BST is so trivial that it is hard to imagine a realistic security model for which it does not hold."
  - Basis: given assumptions known to be non-secure, BST can prove a non-secure system to be secure

†-Property

- State (b, m, f, h) satisfies the †-property iff for each $s \in S$ the following hold:
  1. $b(s: \underline{\text{a}}) \neq \varnothing \Rightarrow [\forall o \in b(s: \underline{\text{a}})\ [\ f_c(s)\ dom\ f_o(o)\ ]\ ]$
  2. $b(s: \underline{\text{w}}) \neq \varnothing \Rightarrow [\forall o \in b(s: \underline{\text{w}})\ [\ f_o(o) = f_c(s)\ ]\ ]$
  3. $b(s: \underline{\text{r}}) \neq \varnothing \Rightarrow [\forall o \in b(s: \underline{\text{r}})\ [\ f_c(s)\ dom\ f_o(o)\ ]\ ]$
- Idea: for reading, subject dominates object; for writing, subject also dominates object
- Differs from *-property in that the mandatory condition for writing is reversed
  - For *-property, it's "object dominates subject"

Analogues

The following two theorems can be proved

- $\Sigma(R, D, W, z_0)$ satisfies the †-property relative to $S' \subseteq S$ for any secure state z_0 iff for every action $(r, d, (b, m, f, h), (b', m', f', h'))$, W satisfies the following for every $s \in S'$
  - Every $(s, o, p) \in b' - b$ satisfies the †-property relative to S'
  - Every $(s, o, p) \in b$ that does not satisfy the †-property relative to S' is not in b
- $\Sigma(R, D, W, z_0)$ is a secure system if z_0 is a secure state and W satisfies the conditions for the simple security condition, the †-property, and the ds-property.

Problem

- This system is *clearly* non-secure!
  - Information flows from higher to lower because of the †-property

Discussion

- Role of Basic Security Theorem is to demonstrate that rules preserve security
- Key question: what is security?
  - Bell-LaPadula defines it in terms of 3 properties (simple security condition, *-property, discretionary security property)
  - Theorems are assertions about these properties
  - Rules describe changes to a *particular* system instantiating the model
  - Showing system is secure requires proving rules preserve these 3 properties

Rules and Model

- Nature of rules is irrelevant to model
- Model treats “security” as axiomatic
- Policy defines “security”
  - This instantiates the model
  - Policy reflects the requirements of the systems
- McLean’s definition differs from Bell-LaPadula
  - … and is not suitable for a confidentiality policy
- Analysts cannot prove “security” definition is appropriate through the model

System Z

- System supporting weak tranquility
- On *any* request, system downgrades *all* subjects and objects to lowest level and adds the requested access permission
  - Let initial state satisfy all 3 properties
  - Successive states also satisfy all 3 properties
- Clearly not secure
  - On first request, everyone can read everything

Reformulation of Secure Action

- Given state that satisfies the 3 properties, the action transforms the system into a state that satisfies these properties and eliminates any accesses present in the transformed state that would violate the property in the initial state, then the action is secure
- BST holds with these modified versions of the 3 properties

Reconsider System Z

- Initial state:
  - subject s, object o
  - C = {High, Low}, K = {All}
- Take:
  - $f_c(s)$ = (Low, {All}), f_o(o) = (High, {All})
  - $m[s, o]$ = { w }, and b = { (s, o, w) }.
- s requests r access to o
- Now:
  - $f'_o(o)$ = (Low, {All})
  - $(s, o, \underline{r}) \in b'$, $m'[s, o] = \{\underline{r}, \underline{w}\}$

Non-Secure System Z

- As $(s, o, \underline{r}) \in b' - b$ and $f_o(o)$ *dom* $f_c(s)$, access added that was illegal in previous state
  - Under the new version of the Basic Security Theorem, the current state of System Z is not secure
  - But, as $f'_c(s) = f'_o(o)$ under the old version of the Basic Security Theorem, the current state of System Z is secure

Response: What Is Modeling?

- Two types of models
  1. Abstract physical phenomenon to fundamental properties
  2. Begin with axioms and construct a structure to examine the effects of those axioms
- Bell-LaPadula Model developed as a model in the first sense
  – McLean assumes it was developed as a model in the second sense

Reconciling System Z

- Different definitions of security create different results
  - Under one (original definition in Bell-LaPadula Model), System Z is secure
  - Under other (McLean's definition), System Z is not secure