Jurij Mihelič
Faculty of Computer and Information Science
University of Ljubljana
&
Beyond Semiconductor d.o.o.
Slovenia
Contents

• Cybersecurity
• Information flow
• Confidentiality and integrity
  • Bell-LaPadula and Biba models
• Information flow policy
  • Formalizing tag propagation
• Non-interference
• Language-based IFT
• Information downgrading
• Separation kernel formalization
Cybersecurity

InfoSec – information security
practice of protecting sensitive information and critical systems

CyberSec – cyber security
InfoSec related to computer systems and data

With the goal to prevent/reduce the likeliness of unauthorized/inappropriate access to data such as

unlawful use, disclosure, disruption
deletion, corruption, modification, inspection
recording, devaluation etc.
Cybersecurity

A threat is a potential negative action or event facilitated by a vulnerability that results in an unwanted impact on a computer system or application.

Accidental negative events
natural disasters, fires, tornados, radiation, malfunctioning

Intentional negative events
adversary attacks, criminal, hacking
Cybersecurity

• Certification: Common Criteria, CC
  • ISO/IEC 15408 standard
  • Common Criteria for Information Technology Security Evaluation
  • product evaluation criteria

EAL – Evaluation Assurance Levels

EAL1: Functionality Tested
EAL2: Structurally Tested
EAL3: Methodically Tested and Checked
EAL4: Methodically, Designed, Tested and Reviewed
EAL5: Semiformally Designed and Tested
EAL6: Semiformally Verified Design and Tested
EAL7: Formally Verified Designed and Tested
Cybersecurity

• Formal methods

```plaintext
subsubsection {Interference relation}

abbreviation arc_in :: "'a policy ⇒ 'a ⇒ 'a ⇒ bool" ("(_:_→_)") 70) where
  "arc_in p a b ≡ (a, b) ∈ arcs p"

hide_const (open) arc_in

fun flow_in :: "'a policy ⇒ 'a list ⇒ bool" where
  "flow_in _ [] = False" |
  "flow_in _ [] = False" |
  "flow_in p (a#b#[]) = (p: a → b)" |
  "flow_in p (a#b#w) = ((p: a → b) ∧ flow_in p (b#w))"

definition flow_in' :: "'a policy ⇒ 'a list ⇒ bool" where
  "flow_in' p w ≡ length w ≥ 2 ∧ (∀ i < length w - 1 . (p: w!i → w!(i+1)))"

definition reachable_in :: "'a policy ⇒ 'a ⇒ 'a ⇒ bool" ("(_:→→)_") 70) where
  "reachable_in p a b ≡ (∃ w . a = hd w ∧ b = last w ∧ flow_in p w)"
```

subsection {Non-exfiltration}

text {Non-reachable tags cannot be in outs of the last step}

definition non_exfiltration :: "'a policy ⇒ 'a step list ⇒ bool" where
  "non_exfiltration p w ≡ (w = [] ∨ (∀ a b . a ∈ ins (hd w) ∧ (¬ (p: a → b)) → b ∉ outs (las

lemma preservance_gives_non_exfiltration:
  shows "preservance p w → non_exfiltration p w"
unfolding preservance_def non_exfiltration_def
by blast

corollary non_exfiltration:
  assumes "valid_policy p"
  and "∀ u . restricted_step p u"
  shows "walk w → non_exfiltration p w"
using assms
using walks_are_restricted preservance_1 preservance_gives_non_exfiltration
by blast
```
Information flow security

- Information flow
  - transfer of **information**
  - from a **source** to a **destination**
    - a passive entity that contains information
    - e.g., variable, record, object, file, memory or storage location
  - by a **subject**
    - an active entity that requests access to an object
    - e.g., user, process
  - during an information processing **activity**
    - ability of a subject to perform a task or interact with an object
    - e.g., operation, program statement, machine instruction

It's different than data flow.
Information flow security

- **Desirable vs. undesirable** information flow
  - depends on the property/application

- **confidentiality**
  - data can be **read by authorized** users and is not disclosed to unauthorized users
  - *secret data does not leak to a public place*
  - read protection

- **integrity**
  - data can be **changed by authorized** users and cannot be altered by unauthorized users
  - *trusted data is not influenced by dubious data*
  - write protection
Information flow security

• Information flow **tracking**
  • analysis and monitoring
    • determine the flow in a given program/process
    • static analysis, dynamic monitoring

• **control**
  • limiting the flow during information processing
  • firewalls, ACLs, secure channels

• **Guarantees and assurances**
  • properties about information propagation
Information flow security

• Perfect security is hard
Confidentiality

• Two-level confidentiality

low level: public data
• insensitive data
• may be publicly observed

high level: private data
• secret data
• may not be publicly observed

• Multiple levels
  • MLS – Multiple Levels of Security
  • EU classified information

<table>
<thead>
<tr>
<th>level</th>
<th>the unauthorised disclosure of this information could</th>
</tr>
</thead>
<tbody>
<tr>
<td>EU Top secret</td>
<td>cause exceptionally grave prejudice to</td>
</tr>
<tr>
<td>EU Secret</td>
<td>seriously harm</td>
</tr>
<tr>
<td>EU Confidential</td>
<td>harm</td>
</tr>
<tr>
<td>EU Restricted</td>
<td>be disadvantageous to</td>
</tr>
<tr>
<td></td>
<td>the essential interests of the EU or one or more of the member states</td>
</tr>
</tbody>
</table>
Confidentiality

• Bell-LaPadula model
  • defined by the US DoD to formalize a MLS policy
  • a state transition model of security policy
  • security labels on objects
  • clearance levels for subjects

• subjects access objects
  • each state transition preserves a secure state
  • two MAC rules
  • one DAC rule (specified with an access matrix)
Confidentiality

- Bell-LaPadula model
- two MAC rules

Simple Security Property
read down / no read up

Star Property
write up / no write-down

<table>
<thead>
<tr>
<th>Top secret</th>
<th>Secret</th>
<th>Confidential</th>
<th>Unclassified</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Top secret

Secret

Confidential

Unclassified
Confidentiality

• Bell-LaPadula model
  • **Strong Star Property**
    • subject can write objects only to the same level
    • motivated by the *integrity* concerns

• **Trusted Subjects**
  • can *downgrade* the information: high to low transfer
  • are not restricted to the Star Property

• **Principle of Tranquility**
  • the security level of an object or subject may never change while it is being referenced
Integrity

• Two-level integrity
  • high level: trusted data
  • low level: dubious data
• information flow policy
  • low to low, high to high, **high to low**
  • but **low to high** is prohibited
Integrity

• Biba model
  • objects and subjects are classified by integrity levels
  • prevent inappropriate modification of data

Simple Integrity Property
read up / no read down

Star Integrity Property
write down / no write up

- Highly trusted
- Trusted
- Slightly trusted
- Untrusted
Integrity

- Bell-LaPadula and Biba models duality

Simple Security Property:
- Read down / no read up

Top secret
- Secret
- Confidential
- Unclassified

Star Property:
- Write up / no write down

Simple Security Property:
- Read up / no read down

Highly trusted
- Trusted
- Slightly trusted
- Untrusted

Star Property:
- Write down / no write up
Information flow policy

A set of rules specifying directions between entities in which the information may flow or must not flow.

- entities
  - subjects: process, person
  - objects: file, memory page, variable
  - tags, labels: data classifications
  - actions: read, write, computation
Information flow policy

• Definition

\[ \mathcal{P} = (T, \rightsquigarrow) \]

- a set \( T \) of entities (labels, tags)
  - specifying security classes
- a binary relation \( \rightsquigarrow \) over \( T \)
  - a set of ordered pairs: \( \rightsquigarrow \subseteq T \times T \)
  - specifying allowed flow between entities
- a negation of \( \rightsquigarrow \)
  - \( x \nsim y \equiv \neg(x \rightsquigarrow y) \)
Information flow policy

• Confidentiality
  • $T = \{ \text{pub, priv} \}$
  • $\sim = \{ \text{pub} \sim \text{pub}, \text{priv} \sim \text{priv}, \text{pub} \sim \text{priv} \}$

<table>
<thead>
<tr>
<th>$x \sim y$</th>
<th>pub</th>
<th>priv</th>
</tr>
</thead>
<tbody>
<tr>
<td>pub</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>priv</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

• Integrity
  • $t \sim t, d \sim d, t \sim d$
Information flow policy

- Confidentiality and integrity
Information flow policy

- Confidentiality and integrity combined

<table>
<thead>
<tr>
<th></th>
<th>Dubious</th>
<th>Trusted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private</td>
<td>priv dub</td>
<td>priv trust</td>
</tr>
<tr>
<td>Public</td>
<td>pub dub</td>
<td>pub trust</td>
</tr>
</tbody>
</table>
Information flow policy

• Non-linear policies
  • Cartesian product
  • subset of permissions

• Timing
  • constant/variable time operations

• Tracking different sources
  • keyboard, mouse, GPS, camera
**Information flow policy**

- **Properties of relations**

**x to x, \( \forall x \):**
- reflexive: \( x \sim x \)
- irreflexive: \( \neg(x \sim x) \)

**x to y, \( \forall x, y \):**
- connected: \( x \neq y \Rightarrow x \sim y \lor y \sim x \)
- strongly connected: connected + reflexive

**x to y vs y to x, \( \forall x, y \):**
- symmetric: \( x \sim y \Rightarrow y \sim x \)
- asymmetric: \( x \sim y \Rightarrow \neg(y \sim x) \)
- antisymmetric: \( x \sim y \land y \sim x \Rightarrow x = y \)

**x, y, and z, \( \forall x, y, z \):**
- transitive: \( x \sim y \land y \sim z \Rightarrow x \sim z \)
Information flow policy

• Properties of relations
  • partially ordered set (POS)
    • reflexive, transitive, antisymmetric

• universally bounded lattice \((S, \bowtie, \bot, T, \oplus, \otimes)\)
  • POS + supremum/join and infimum/meet

\(S = \{ABC, AB, AC, BC, A, B, C, \emptyset\}\)
\(\bowtie = \text{see the figure}\)
\(\bot = \emptyset\)
\(T = ABC\)
\(\oplus = U\)
\(\otimes = \cap\)
Information flow policy

• Secure propagation

**Theorem 2.** Given a security policy $\mathcal{P} = (T, \preceq)$ and a walk $(u_1, u_2, \ldots, u_l)$ in a $\mathcal{P}$-restricted step graph we have that the tag $\text{out } u_l$ is reachable from any tag $s \in \text{ins } u_1$.

• Non-exfiltration

**Corollary 4** (Non-exfiltration). Given a security policy $\mathcal{P} = (T, \preceq)$ and a walk $(u_1, u_2, \ldots, u_l)$ in a $\mathcal{P}$-restricted step graph it holds for all $t \in T$ that are not reachable from $s \in \text{ins } u_1$ then $t \neq \text{out } u_l$.

• Non-infiltration

**Corollary 5** (Non-infiltration). Given a security policy $\mathcal{P} = (T, \preceq)$ and a walk $w = (u_1, u_2, \ldots, u_l)$ in a $\mathcal{P}$-restricted step graph it holds for all $s \in T$ from which we cannot reach $\text{out } u_l$ then $s \notin \text{ins } u_1$. 
Noninterference

• Noninterference
  • introduced by Goguen and Meseguer, 1982
  • a property that restricts
    the information flow through a system

\[ \text{\(X\) is noninterfering with \(Y\) across a system \(M\) if \(X\)'s input to \(M\) does not affect \(M\)'s output to \(Y\).} \]
Noninterference implies **confidentiality**

X is noninterfering with Y across a system M if X's input to M does not affect M's output to Y.

Observations of Y are entirely independent of the actions of X.

Expresses X's confidentiality guarantee: X cannot reveal any secrets to Y via M.
Noninterference

- Noninterference implies **integrity**

\[ X \text{ is noninterfering with } Y \text{ across a system } M \text{ if } X'\text{s input to } M \text{ does not affect } M'\text{s output to } Y. \]

No information flows from \(X\) to \(Y\) through \(M\).

Expresses \(Y\)'s **integrity** guarantee: \(Y\) cannot be corrupted by \(X\) via \(M\).
Noninterference

• Interference
  • $pub \sim pub, priv \sim priv, pub \sim priv$

• Noninterference
  • $priv \not\sim pub$
  • private data does not interfere with public data
    • any variation of private data does not cause a variation of public data
  • adversary
    • has access to the public data
    • cannot observe any difference between two executions that differ only in their private data
Language-based IFT

Program analysis
a process of automatic analysis of the behavior of computer programs

Check correctness
• find programming errors (bugs)
• reveal safety errors
• reveal security vulnerabilities

Optimize performance
• improve program performance
• reduce resource usage
Language-based IFT

- Language-based IFT
  - to secure data manipulated by a program
  - enforce a given information flow policy
  - track possible transfers of information occurring throughout program execution
**Language-based IFT**

- **Dynamic IFT**
  - analysis during execution (runtime)
    - data from untrusted sources is labeled (tainted)
    - each data (memory location) has a label
    - label propagation at runtime
    - can cause overhead on execution
  - examines only one possibility
    - the actual input
    - may underapproximate possible behavior
Language-based IFT

• Static IFT
  • analysis without executing the program/code
    • performed before execution (on compilation)
    • major overhead of analysis
  • examines all possibilities
    • considers all inputs and all execution paths
    • can reveal errors that may not manifest themselves for a long time
    • can overapproximate possible behavior
Language-based IFT

• Control flow graph
  • nodes: operations
  • edges: transfer of control

```plaintext
x := read()
if x > 42
  then y := 4
  else y := 2
z := x + y
while z > 0 do
  z := z - 1
print(z)
```
Language-based IFT

- Variables and security labels
  - the policy specifies security classes
  - but the program uses variables

- Flow relation on variables
  - $x \rightsquigarrow y \equiv \text{tag}(x) \rightsquigarrow \text{tag}(y)$

```plaintext
x := read()
if x > 42
  then y := 4
  else y := 2
z := x + y
while z > 0 do
  z := z - 1
print(z)
```
Language-based IFT

• Explicit flow
  • from inputs of an operation to its outputs
  • tag propagation rule
    • \( \text{tag}(result) = \text{tag}(\text{arg}1) \oplus \text{tag}(\text{arg}2) \ldots \)

int a: public
int b: private

int x, y, z

// private or public?
x := a + a
y := b + b
z := a + b

cased by a data flow dependency
Language-based IFT

• Implicit flow
  • in conditionally executed code
  • from the condition to the code

\textbf{bool a: public}
\textbf{bool b: private}

\textbf{bool x, y, z, w}

// private or public?
if a then x := true else x := false
if b then y := true else y := false

z := w := false
if a then z := true
if b then w := true

\textbf{bool a: trusted}
\textbf{bool b: dubious}

\textbf{string x, y, z, w}
\textbf{string s = user_input()}

// trusted or dubious?
if a then x := "Some string"
if a then y := s
if b then z := "Some string"
if b then w := s
Language-based IFT

- Hidden implicit flow
  - if a branch is not executed
  - How to handle such flows?
    - Add spurious definitions into branches

```
x := false
if cond then x := true
else x := x
```

```
x := y := 0
if cond then
  x := 42
else
  y := 3.14
```

```
x := y := 0
if cond then
  x := 42
  y := y
else
  y := 3.14
  x := x
```
Language-based IFT

• Tag propagation for implicit flow
  • stack $S$ of tags
    • contains tags of values that influence the current flow of control
  • rules
    • when an operation is executed, consider also all tags on $S$ for tag propagation
    • when a value $x$ influences a branch decision push tag$(x)$ on the stack $S$
    • when end-of-branch is reached pop label$(x)$ from the stack $S$
Downgrading

• Challenge: Information upwards drift
  • also called label-creep phenomenon
Downgrading

• Challenge: Noninterference is not practical
  • noninterference is too strict for use in most real-world applications
    • e.g., prevents all information flows from private to public
  • for most applications, the appropriate policy should permit controlled downward flows
Downgrading

• Trusted user/process
  • may perform downgrading
• declassification
  • for confidentiality policies
• endorsement
  • integrity policies

What information is released?
Who is authorized to access it?
Where is the information released?
When is the information released?
Downgrading

• Examples
  • encryption

  pt := "42 is the answer"
  ct := encrypt(pt)

• hashing

  m := "A private message"
  h := hash_sha256(m)

• password check

  pw := read_input()
  ok := pw.length() >= 10

• html escaping

  x := read_input()
  y := html_escape(x)
Downgrading

• Intransitive security policy
  • ensures that downward information flow passes through trusted user
  • cycles in the IF policy

• Intransitive non-interference
  • not accurate description
    • actually, interference relation is not transitive
  • noninterference under an intransitive security policy
Downgrading

• Separating the relation
  • security-oblivious operations
  • security-aware operations

```plaintext
pw := read_input()
ok := pw.length() >= 10
ok := downgrade(ok)
print(ok)
```
Thank you