Engineering Secure Software Systems

Winter 2020/21, Weeks 10 – 13: Information Flow

Henning Schnoor

Institut für Informatik, Christian-Albrechts-Universität zu Kiel

Part II: Information Flow

Overview

Information-Flow Security

lecture up to now

- attacker model: network attacker
- models attacks on communication
- protection: cryptography

$\rightarrow$ protection at network level

alternative: internal point of view

attacks “inside” one system

- buffer overflows
- format strings
- RPC vulnerabilities
- malware
- covert channels

also need protection at system/application level

Level of Abstraction

cryptographic protocols

- high level of abstraction with respect to cryptography
  - term model
  - idealized security properties
- low level of abstraction with respect to processing
  - structure of messages modeled precisely
  - pattern-matching steps fixed completely

information-flow modeling

- scope: all system components
- basic model: FSMs
- high/low level of abstraction depending on semantics of states
- fewer details in model, more modeling work

Information-Flow in the News

recent high-profile security issues

- Meltdown
- Spectre

(one of the) core issue(s)

information leakage via timing

reference

Richard J. Lipton and Kenneth W. Regan. Timing Leaks Everything. 2018. URL: `https://rjlipton.wordpress.com/2018/01/12/timing-leaks-everything`

Background: Speculative Execution

recall

microprocessor design: pipelining

reasons why code is not executed

- unauthorized memory access
- branching in “unexpected” direction

speculative execution

- execute code anyway
- roll-back if code not to be executed (backtrack)

Attack Outline

attack

- attacker wants to learn value b at location x of memory map K
- creates array A of objects Q with width equal to cache page size
- array only **created**, not read or initialized

$\rightarrow$ content of A not in cache

```
object Q;         //loaded into chip memory
byte b = 0;
while (b == 0) {
  b = K[x];       //violates privilege---so raises an exception
}
Q = A[b];         //should not be executed but usually is

//continue process after subprocess dies or exception is caught:
int T[256];
for (int i = 0; i < 256; i++) {
  T[i] = the time to find A[i];
}
if T has a clear minimum T[i] output i, else output 0.
```

cases

$b \neq 0$ while-loop exits, $A[b]$ cached $\rightarrow$ accessing $A[b]$ faster

$b = 0$ race condition handling (possibly no fetch)

Reference

reference

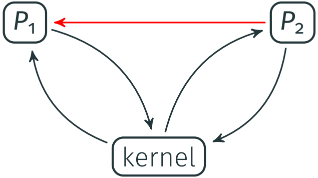
Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. “Meltdown”. In: ArXiv e-prints (Jan. 2018). arXiv: 1801.01207

highlights

- operating systems: Linux, Windows
- Docker
- Intel: read speed 503 KB/sec
- only “toy examples” for AMD, ARM

security policy

process isolation



communicating processes

- two user processes and kernel
- both processes may communicate with kernel
- communication between processes forbidden

meltdown

- allows direct communication between P_1 and P_2: **system does not respect security policy**
- uses covert channel: timing information

Side Channel Attack: Project System Bus Radio

approach

- electronic systems emit electromagnetic radiation
- approach: choose processor workload so that radiation is AM signal (amplitude modulation, "Mittelwelle")

references

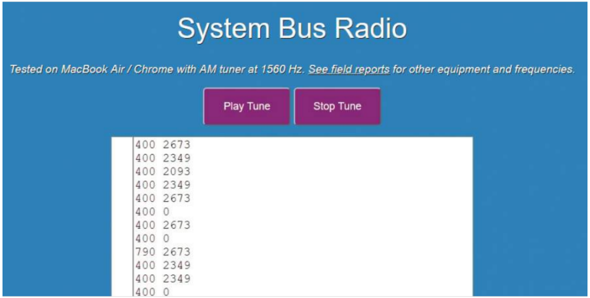
- `https://github.com/fulldecent/system-bus-radio`
- Christof Windeck. PC und Notebook senden auf Mittelwelle - ohne Zusatz-Hardware. 2018. URL: `https://www.heise.de/ct/artikel/PC-und-Notebook-senden-auf-Mittelwelle-ohne-Zusatz-Hardware-3948910.html`

System Bus Radio in Practice

Kurztest | PC sendet per Mittelwelle c't 5/2018 S. 48

PC sendet Mittelwelle

Die freie Software System Bus Radio verwandelt einen PC in einen Mittelwellensender – per JavaScript im Browser, ohne weitere Hardware.



System Bus Radio demonstriert eine Sicherheitslücke, die Hacker nutzen könnten, um Daten aus einem PC völlig ohne Netzwerkverbindungen abzugreifen. Dabei geht es um elektromagnetische Abstrahlungen, wel-

Overview

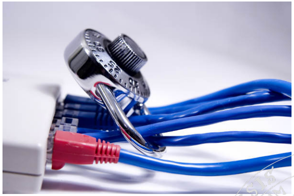
Information Flow Motivation

information security

crucial aspect: protection against unauthorized access or manipulation

noninterference

- introduced by Goguen and Meseguer [GM82]
- general approach to capture security
- information flows and covert channels
- confidentiality and integrity
- goal: detect undesired information flows



Security Policies

scenarios: different "security levels" on single system

- different processes running on the same system,
- different users interacting with the same system,
- different tabs in a browser

security policies

- policies: govern "what may be done" with information
- can be arbitrarily complex (see later)
- suffices for start: H/L policy

  H high-security data (and users), must be protected

  L low-security data (and users), considered public

Information-Flow Security in Lecture

variations of noninterference

- classical: **transitive** noninterference
- policy generalizations: **intransitive** noninterference
- system generalizations: **dynamic** noninterference
- timing assumptions: (a)synchronous systems

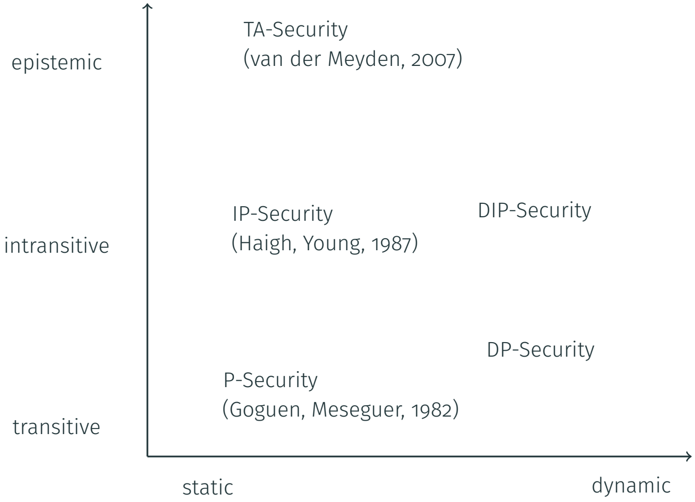
aspects of noninterference

- definitions and relationships
- characterizations
- verification algorithms and complexity results

as usual: no “one-size-fits-all” approach

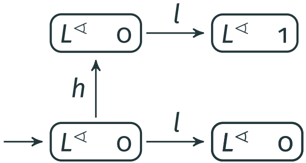
- choice of “correct” definition depends on situation
- covered definitions share basic structure
- similarities lead to common algorithmic approach

Non-Interference Notions



Information-Flow Example

system



- $L^{\lhd}$: output to L in state
- users H and L perform actions h, l
- goal: L must not learn anything about which actions H performs

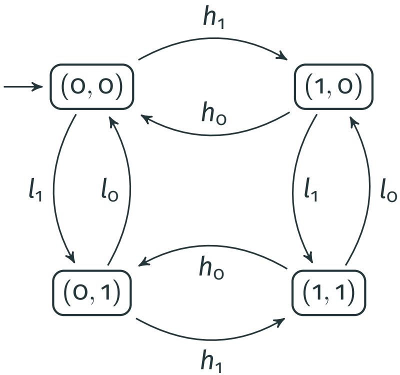
analysis

system secure?

- intuitively?
- formally?

Information-Flow Example

system



specification

- output to L: second component of pairs
- users H, L perform actions h_0, h_1, l_0, l_1
- goal: L must not learn anything about H's actions

analysis

system secure?

- intuitively?
- formally?

Noninterference: Formal Model

systems

- finite automata, actions change states
- agents, domains: users of system
- $\mathtt{obs}_L(s)$: observation of agent L in state s

policy

$\rightarrow$ indicates allowed information flow:

- $L \rightarrow H$: information may flow from L to H
- $H \not\rightarrow L$: not from H to L

H, L: agents (users) or processes in a system

central question

what does "information flows from H to L" mean?

noninterference

formalize this!

Nointerference: System Model

system: tuple $(S, s_0, A, \texttt{step}, D, O, \texttt{obs}, \texttt{dom})$ with

- S set of states
- $s_0 \in S$ initial state
- A set of actions
- $\texttt{step}\colon S \times A \to S$ **deterministic** step function
- D set of security domains (agents)
- O set of possible observations
- $\texttt{obs}\colon S \times D \to O$ observation function
- $\texttt{dom}\colon A \to D$ domain function

notation

- $s \in S$, $\alpha \in A^*$, then $s \cdot \alpha$: state obtained by "performing α from s"
  - $s \cdot \epsilon = s$
  - $s \cdot \alpha a = \texttt{step}(s \cdot \alpha, a)$ (for $\alpha \in A^*, a \in A$)
- write $\texttt{obs}_u(s)$ for $\texttt{obs}(s, u)$

Security Policies (formal)

definition (security policy)

For a set of domains D, a security policy is a set $\rightarrowtail \subseteq D \times D$.

properties

- $\rightarrowtail$ is usually reflexive
- $\rightarrowtail$ is often transitive (why?)

reference

examples from Sebastian Eggert, Ron van der Meyden, Henning Schnoor, and Thomas Wilke. "Complexity and Unwinding for Intransitive Noninterference". In: CoRR abs/1308.1204 (2013). URL: `http://arxiv.org/abs/1308.1204`

Recall: Indistinguishability

crypto protocols

- secrecy on term level
- indistinguishability: **tests** (operations on terms)
- security: t_1 and t_2 indistinguishable
  - e.g., t_1 and t_2 Alice's messages in voting protocol

information-flow security

- "data:" performed actions
- indistinguishability: from observations ($q_1 \equiv q_2$ iff $\mathtt{obs}_L(q_1) = \mathtt{obs}_L(q_2)$)
- security: q_1 and q_2 indistinguishable
  - if "same public data" in q_1 and q_2

Information-Flow Security Approach

“required” and “achieved” indistinguishability

traces $\alpha_1, \alpha_2 \in A^*$ **should be** indistinguishable if they have same “public data”

states $s \cdot \alpha_1, s \cdot \alpha_2$ **are** indistinguishable, if they have same observations

security: system achieves required indistinguishabilities

- state-equivalence relation “includes” trace-equivalence relation
- traces that should be indistinguishable lead to indistinguishable states

three instances

- P-security [GM82],
- IP-security [HY87],
- TA-security [Mey07].

Overview

Overview

Noninterference: P-Security

overview

- simple notion of security [GM82]
- assumes H/L policy. $L \rightarrowtail H$
- intuition: “low” users may not “see” anything that “high users” do

definition (purge function)

$E \subseteq D$ set of domains, sequence $\alpha \in A^*$, policy $\rightarrowtail$

- $\alpha\upharpoonright E$: subsequence of actions a from α with $\mathtt{dom}(a) \in E$
- $\mathtt{purge}(\alpha, u) = \alpha\upharpoonright \{v \in D \mid v \rightarrowtail u\}$
- often write $\mathtt{purge}_u(\alpha)$, omit $\rightarrowtail$

intuition

$\mathtt{purge}(\alpha, u)$ contains actions from α that u may “learn about”

Noninterference: P-Security

definition (P-security)

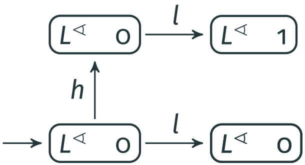
A system $(S, s_0, A, \texttt{step}, D, O, \texttt{obs}, \texttt{dom})$ is P-secure with respect to a policy $\rightarrowtail$, if for all $u \in D$, $s \in S$, $\alpha_1, \alpha_2 \in A^*$ we have that:

$$\text{If } \texttt{purge}_u(\alpha_1) = \texttt{purge}_u(\alpha_2), \text{ then } \texttt{obs}_u(s \cdot \alpha_1) = \texttt{obs}_u(s \cdot \alpha_2).$$

intuition

- α_1 and α_2 should "look the same" to u
- performing α_1 or α_2 from s should make no difference for u
- u should receive the same information from the system for both sequences

system



- $L^{\lhd}$: $\mathtt{obs}_L$
- h, l: actions of H, L
- policy: $L \rightarrowtail H$

analysis

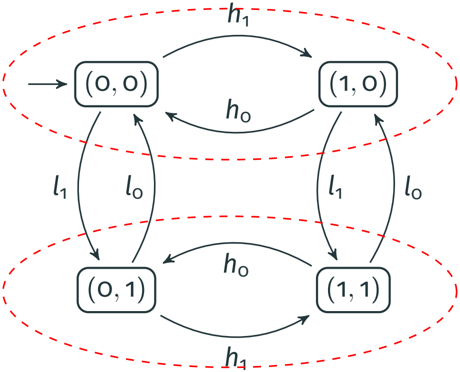
system secure?

- intuitively?
- formally?

insecure

- $\alpha_1 : l$, then $\mathtt{purge}_L(\alpha_1) = l$
- $\alpha_2 : hl$, then $\mathtt{purge}_L(\alpha_2) = l$
- $\mathtt{obs}_L(q_0 \cdot \alpha_1) = 0 \neq 1 = \mathtt{obs}_L(q_0 \cdot \alpha_2)$

system



specification

- L observation: second component of state name
- h_x, l_x: actions of H, L
- policy: $L \rightarrowtail H$

analysis

system secure?

- intuitively?
- formally?

A dual view

policy $H \not\rightarrow L$

intuition: secrecy

- H has access to “secret” data
- L tries to learn secrets
- (H not necessarily honest)
- protect H data from read access

dual perspective: integrity

- L data: must be preserved
- H tries to modify L data
- (H: untrusted process)
- protect L data from write access

dual approach

both cases covered with policy $H \not\rightarrow L$.

Exercise

Task (P-Security Example I)

Is the following system P-secure? Justify your answer.

$$
\begin{tikzcd}
\boxed{0} \arrow[r, "l"] & \boxed{1} \arrow[r, "l"] \arrow[d, "h"] & \boxed{0} \\
\boxed{0} \arrow[r, "l"] \arrow[u, "h"] & \boxed{1} \arrow[r, "{l,h}"] & \boxed{0} \arrow[u, "h"]
\end{tikzcd}
$$

Exercise

Task (P-Security Example II)

Is the following system P-secure? Justify your answer.

$$
\begin{tikzcd}
\boxed{0} \arrow[r, "l"] & \boxed{0} \arrow[r, "l"] \arrow[d, "h"] & \boxed{0} \\
\boxed{0} \arrow[u, "h"] \arrow[r, "l"] & \boxed{0} \arrow[r, "{l, h}"] & \boxed{1} \arrow[u, "h"']
\end{tikzcd}
$$

Exercise

Task (alternative definition of P security I)

Let $M = (S, s_0, A, \texttt{step}, D, O, \texttt{obs}, \texttt{dom})$ be a system and let $\rightarrowtail$ be a policy for M. Prove that the following are equivalent:

1. M is P-secure with respect to $\rightarrowtail$,
2. for all states $s \in S$, all $u \in D$, and all traces $\alpha \in A^*$, we have that

$$\texttt{obs}_u(s \cdot \alpha) = \texttt{obs}_u(s \cdot \texttt{purge}_u(\alpha)).$$

Note: The characterization from this task is in fact the original definition of P-Security, the (equivalent, by the above) definition we work with in the lecture was later user by Ron van der Meyden.

Exercise

Task (alternative definition of P security II)

An alternative definition of P-security is the following[a]:

- for an agent u, a state s, and an action sequence α, define $\mathtt{obs}_u(s \to \alpha)$ as the sequence of observations that u makes when α is performed, starting in state s. Formally:
  - $\mathtt{obs}_u(s \to \epsilon) = \mathtt{obs}_u(s)$,
  - for a sequence α and an action a, $\mathtt{obs}_u(s \to a\alpha) = \mathtt{obs}_u(s) \rtimes \mathtt{obs}_u(s \cdot a \to \alpha)$, (here, $\rtimes$ is string concatenation with elimination of repetitions).
- a system is *secure* if the following holds: For each state s, each agent u and each action sequences α_1, α_2 with $\mathtt{purge}_u(\alpha_1) = \mathtt{purge}_u(\alpha_2)$, we have $\mathtt{obs}_u(s \to \alpha_1) = \mathtt{obs}_u(s \to \alpha_2)$.

Show that this definition is equivalent to P-security, i.e., that any system is secure with respect to the above definition if and only if it is P-secure.

[a] as usual, we fix a noninterference policy $\rightarrowtail$

Exercise

Task (P-security reduction to two domains)

For a system $M = (S, s_0, A, \mathtt{step}, D, O, \mathtt{obs}, \mathtt{dom})$ and a subset of agents $C \subseteq D$, we define the *restriction of M to C* as follows: $M \upharpoonright C = (S, s_0, A', \mathtt{step}', C, O, \mathtt{obs}', \mathtt{dom})$, where

- $A' = \{a \in A \mid \mathtt{dom}(a) \in C\}$,
- $\mathtt{step}'$ is the restriction of $\mathtt{step}$ to S and the actions in A',
- $\mathtt{obs}'$ analogously is the restriction of $\mathtt{obs}$ to S and the agents in C,

For a policy $\rightarrowtail$, the restriction to C is defined as $\rightarrowtail \upharpoonright C = \rightarrowtail \cap (C \times C)$.

Prove or disprove the following statement: A system M is P-secure with respect to a policy $\rightarrowtail$ if and only if $M \upharpoonright C$ is secure with respect to $M \upharpoonright C$ for all $C \subseteq A$ with $|C| = 2$. (Later: Does the corresponding claim hold for IP-security?)

Exercise

Task (P-security and non-transitive policies)

Prove or disprove the following: If $M = (S, s_0, A, \mathtt{step}, D, O, \mathtt{obs}, \mathtt{dom})$ is a system and $\rightarrowtail$ is a policy for M, then the following are equivalent:

- M is P-secure with respect to $\rightarrowtail$,
- M is P-secure with respect to the transitive closure of $\rightarrowtail$.

Overview

Proving (in-)Security

methods

- prove insecurity: counter-example
- prove security: manual proof

comparison to protocols

- approach similar
- model of (realistic) systems: much larger!

consequence

- need proof technique: short “arguments” why we should believe in system’s security
- need automatic security analysis

Information-Flow Security Proofs

verifying P (later: also IP/TA-)security

- if $\mathtt{purge}_u(\alpha_1) = \mathtt{purge}_u(\alpha_2)$, then $\mathtt{obs}_u(s \cdot \alpha_1) = \mathtt{obs}_u(s \cdot \alpha_2)$ (or `ipurge`, or `ta`)
- only finitely many pairs (s_1, s_2) with $\mathtt{obs}_u(s_1) = \mathtt{obs}_u(s_2)$ required
- security proof needs list of all these (s_1, s_2)
- infinitely many α_1, α_2 to consider

algorithmic approach

- complete set of pairs (s_1, s_2) with $\mathtt{obs}_u(s_1) = \mathtt{obs}_u(s_2)$ required
- start with $\{(s, s) \mid s \in S\}$
- add pairs for "suitable" sequences α_1, α_2 until fixpoint reached

Definition

A P-unwinding for a system $(S, s_0, A, \texttt{step}, D, O, \texttt{obs}, \texttt{dom})$ and a policy $\rightarrowtail$ is a family of equivalence relations $(\sim_u)_{u \in D}$ on S such that

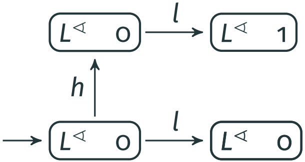
OC^P	if $s \sim_u t$, then $\texttt{obs}_u(s) = \texttt{obs}_u(t)$	output consistency
SC^P	if $s \sim_u t$, then $s \cdot a \sim_u t \cdot a$	step consistency
LR^P	if $\texttt{dom}(a) \not\rightarrowtail u$, then $s \sim_u s \cdot a$	left respect

theorem (Rushby, [Rus92])

A system M is P-secure with respect to $\rightarrowtail$ if and only if there is a P-unwinding for M and $\rightarrowtail$.

corollary

P-Security can be verified in polynomial time. (proof follows) system 1



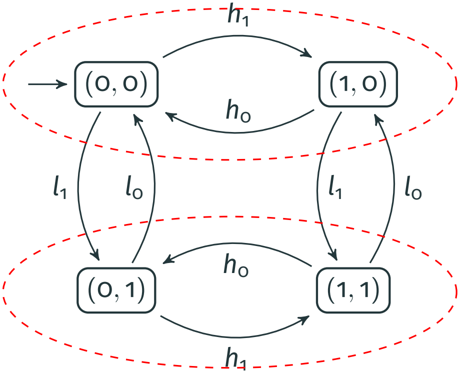
conditions

OC $s \sim_u t$, then $\texttt{obs}_u(s) = \texttt{obs}_u(t)$

SC $s \sim_u t$, then $s \cdot a \sim_u t \cdot a$

LR $\texttt{dom}(a) \not\rightarrowtail u$, then $s \sim_u s \cdot a$

system 2



Exercise

Task (uniqueness of unwindings)

Show that P-unwindings are not unique, but that mininal P-unwindings are, that is:

1. give an example for a system M and a policy $\rightarrowtail$ such that there are (at least) two different P-unwindings for M and $\rightarrowtail$,
2. show that if M is P-secure with respect to a policy $\rightarrowtail$, then there is a P-unwinding for M and $\rightarrowtail$ that is contained (via set inclusion) in all P-unwindings for M and $\rightarrowtail$.

Characterization of P-Security with Unwindings

https://cloud.rz.uni-kiel.de/index.php/s/6k9DT475qW9NcQg

video content

- proof: a system is P-secure if and only if there is an unwinding
- “canonical” choice of unwindings

study

- watch video—feedback welcome!
- video slides contained in slide set (gray background), additional material in lecture notes
- next week: discussion of content (in small groups), bring questions!

Characterizing P-Security with Unwindings

Theorem

A system M is P-secure with respect to $\rightarrowtail$ if and only if there is a P-unwinding for M and $\rightarrowtail$.

reference

John Rushby. Noninterference, Transitivity, and Channel-Control Security Policies. Tech. rep. CSL-92-02. SRI International, Dec. 1992. URL: `http://www.csl.sri.com/papers/csl-92-2/`

relevance

- classic result, many (more complex) generalizations
- captures "intuitive" reasons for security
- motivation: proof technique, verification

Recall: Definition P-Unwinding

Definition

A P-unwinding for a system $(S, s_0, A, \mathtt{step}, D, O, \mathtt{obs}, \mathtt{dom})$ and a policy $\rightarrowtail$ is a family of equivalence relations $(\sim_u)_{u \in D}$ on S such that

OC^P	if $s \sim_u t$, then $\mathtt{obs}_u(s) = \mathtt{obs}_u(t)$	output consistency
SC^P	if $s \sim_u t$, then $s \cdot a \sim_u t \cdot a$	step consistency
LR^P	if $\mathtt{dom}(a) \not\rightarrowtail u$, then $s \sim_u s \cdot a$	left respect

Simplification

notation

fix user u: write $\texttt{purge}$ instead of $\texttt{purge}_u$, $\sim$ instead of $\sim_u$, $\texttt{obs}$ instead of $\texttt{obs}_u$

possible because P-security "simple:"

- (proof of) unwinding for user u_1 does not depend on unwinding for user u_2
- P-security does not model "interaction" between users
- contrast to IP-security (see later)

Part 1: Unwinding $\to$ P-Security overview

Proof Structure

- assume unwinding exists
- prove key fact:

  for all $\alpha \in A^*$, we have $s \cdot \alpha \sim s \cdot \mathtt{purge}(\alpha)$.
- with key fact and output consistency:

  if $\mathtt{purge}(\alpha_1) = \mathtt{purge}(\alpha_2)$, then $\mathtt{obs}(s\alpha_1) = \mathtt{obs}(s\alpha_2)$.
- this is P-Security.

Proof of Key Fact (Part I)

Claim (Key Fact)

if $\sim$ unwinding, $\alpha \in A^*$, then $s \cdot \alpha \sim s \cdot \texttt{purge}(\alpha)$

recall step consistency

if $s \sim t$, then $s \cdot a \sim t \cdot a$

proof: induction over $|\alpha|$

$\alpha = \epsilon$ $\quad s \cdot \alpha = s \cdot \epsilon = s \sim s = s \cdot \epsilon = s \cdot \texttt{purge}(\epsilon)$, since $\sim$ reflexive

$\alpha \to \alpha a$ $\quad$ induction: $s \cdot \alpha \sim s \cdot \texttt{purge}(\alpha)$, must show: $s \cdot \alpha a \sim s \cdot \texttt{purge}(\alpha a)$

induction step case 1: $\texttt{dom}(a) \to u$

from step consistency and $s \cdot \alpha \sim s \cdot \texttt{purge}(\alpha)$:	$s \cdot \alpha a$	$\sim$	$s \cdot \texttt{purge}(\alpha)a$
since $\texttt{purge}(a) = a$:		$=$	$s \cdot \texttt{purge}(\alpha)\texttt{purge}(a)$
since $\texttt{purge}(\alpha a) = \texttt{purge}(\alpha)\texttt{purge}(a)$:		$=$	$s \cdot \texttt{purge}(\alpha a)$

Proof of Key Fact (Part II)

Claim (Key Fact)

if $\sim$ unwinding, $\alpha \in A^*$, then $s \cdot \alpha \sim s \cdot \mathtt{purge}(\alpha)$

recall left respect

if $\mathtt{dom}(a) \not\rightarrowtail u$, then $s \sim s \cdot a$

proof: induction over $|\alpha|$

$\alpha = \epsilon$ $\quad s \cdot \alpha = s \cdot \epsilon = s \sim s = s \cdot \epsilon = s \cdot \mathtt{purge}(\epsilon)$, since $\sim$ reflexive

$\alpha \to \alpha a$ $\quad$ induction: $s \cdot \alpha \sim s \cdot \mathtt{purge}(\alpha)$, must show: $s \cdot \alpha a \sim s \cdot \mathtt{purge}(\alpha a)$

induction step case 2: $\mathtt{dom}(a) \not\rightarrowtail u$

from left respect	$s \cdot \alpha \sim s \cdot \alpha a$
induction, transitivity	$s \cdot \alpha a \sim s \cdot \mathtt{purge}(\alpha)$
since $\mathtt{dom}(a) \not\rightarrowtail u$	$\mathtt{purge}(\alpha a) = \mathtt{purge}(\alpha)$
so	$s \cdot \alpha a \sim s \cdot \mathtt{purge}(\alpha a)$

proof of key fact complete

next: use this to show security

Proof of Security with Key Fact

Claim

If there is an unwinding, system is P-secure: if $\texttt{purge}(\alpha_1) = \texttt{purge}(\alpha_2)$, then $\texttt{obs}(s \cdot \alpha_1) = \texttt{obs}(s \cdot \alpha_2)$

Key Fact

If $\sim$ unwinding, $\alpha \in A^*$, then $s \cdot \alpha \sim s \cdot \texttt{purge}(\alpha)$

proof

- choose α_1, α_2 with $\texttt{purge}(\alpha_1) = \texttt{purge}(\alpha_2)$
- $s \cdot \alpha_1 \sim s \cdot \texttt{purge}(\alpha_1) = s \cdot \texttt{purge}(\alpha_2) \sim s \cdot \alpha_2$
- output consistency: $\texttt{obs}(s \cdot \alpha_1) = \texttt{obs}(s \cdot \alpha_2)$

recall output consistency

if $s \sim t$, then $\texttt{obs}(s) = \texttt{obs}(t)$

completes proof of first direction

If there is an unwinding, system is P-secure.

Part 2: P-Security $\rightarrow$ Unwinding overview

Proof Structure

- assume system secure: $\texttt{purge}(\alpha_1) = \texttt{purge}(\alpha_2)$ implies $\texttt{obs}(s \cdot \alpha_1) = \texttt{obs}(s \cdot \alpha_2)$
- need to define equivalence relation $\sim$ (for agent u) that satisfies:

OC^P	if $s \sim t$, then $\texttt{obs}(s) = \texttt{obs}(t)$	output consistency
SC^P	if $s \sim t$, then $s \cdot a \sim t \cdot a$	step consistency
LR^P	if $\texttt{dom}(a) \not\rightarrowtail u$, then $s \sim s \cdot a$	left respect

- candidate: $s \sim t$, if "equivalent actions lead to indistinguishably states"

Choice of $\sim$

$s \sim t$ iff for all α_1, α_2 with $\texttt{purge}(\alpha_1) = \texttt{purge}(\alpha_2)$, we have $\texttt{obs}(s \cdot \alpha_1) = \texttt{obs}(t \cdot \alpha_2)$

Proof of Unwinding Properties (Part 1)

Relation

$s \sim t$ iff for all α_1, α_2 with $\mathtt{purge}(\alpha_1) = \mathtt{purge}(\alpha_2)$, we have $\mathtt{obs}(s \cdot \alpha_1) = \mathtt{obs}(t \cdot \alpha_2)$

Claim

if system secure, $\sim$ is an unwinding

proof

- $\sim$ is an equivalence relation
  - $\sim$ reflexive: due to P-security, if $\mathtt{purge}(\alpha_1) = \mathtt{purge}(\alpha_2)$, then $\mathtt{obs}(s \cdot \alpha_1) = \mathtt{obs}(s \cdot \alpha_2)$. So, $s \sim s$.
  - symmetry, transitivity: trivial
- output consistency: let $s \sim t$, choose $\alpha_1 = \alpha_2 = \epsilon$: $\mathtt{obs}(s) = \mathtt{obs}(t)$

Proof of Unwinding Properties (Part 2)

Relation

$s \sim t$ iff for all α_1, α_2 with $\mathtt{purge}(\alpha_1) = \mathtt{purge}(\alpha_2)$, we have $\mathtt{obs}(s \cdot \alpha_1) = \mathtt{obs}(t \cdot \alpha_2)$

proof: if system secure, $\sim$ is an unwinding (here: left respect)

- choose a with $\mathtt{dom}(a) \not\rightarrowtail u$, need to show: $s \sim s \cdot a$
- choose α_1, α_2 with $\mathtt{purge}(\alpha_1) = \mathtt{purge}(\alpha_2)$, need to show: $\mathtt{obs}(s \cdot \alpha_1) = \mathtt{obs}(s \cdot a\alpha_2)$
- since $\mathtt{dom}(a) \not\rightarrowtail u$, we have $\mathtt{purge}(a\alpha_2) = \mathtt{purge}(\alpha_2)$
- so:

$$\begin{array}{lcll} \mathtt{obs}(s \cdot \alpha_1) & = & \mathtt{obs}(s \cdot \alpha_2) & (\text{since } s \sim s \text{ and } \mathtt{purge}(\alpha_1) = \mathtt{purge}(\alpha_2)) \\ & = & \mathtt{obs}(s \cdot a\alpha_2) & (\text{since } s \sim s \text{ and } \mathtt{purge}(\alpha_2) = \mathtt{purge}(a\alpha_2)) \end{array}$$

Proof of Unwinding Properties (Part 3)

Relation

$s \sim t$ iff for all α_1, α_2 with $\mathtt{purge}(\alpha_1) = \mathtt{purge}(\alpha_2)$, we have $\mathtt{obs}(s \cdot \alpha_1) = \mathtt{obs}(t \cdot \alpha_2)$

proof: if system secure, $\sim$ is an unwinding (here: step consistency)

- choose s, t with $s \sim t$, $a \in A$, show: $s \cdot a \sim t \cdot a$.
- choose α_1, α_2 with $\mathtt{purge}(\alpha_1) = \mathtt{purge}(\alpha_2)$, show: $\mathtt{obs}(s \cdot a\mathtt{purge}(\alpha_1)) = \mathtt{obs}(t \cdot a\mathtt{purge}(\alpha_2))$.
- since $s \sim t$ and definition of $\sim$: enough to show that $\mathtt{purge}(a\mathtt{purge}(\alpha_1)) = \mathtt{purge}(a\mathtt{purge}(\alpha_2))$.
- this follows:

$$\begin{aligned} \mathtt{purge}(a\mathtt{purge}(\alpha_1)) &= \mathtt{purge}(a)\mathtt{purge}(\alpha_1) \\ &= \mathtt{purge}(a)\mathtt{purge}(\alpha_2) = \mathtt{purge}(a\mathtt{purge}(\alpha_2)). \end{aligned}$$

completes proof of second direction

If system is secure, there is an unwinding relation.

Conclusion and Outlook

Result

P-Security is completely characterized by unwindings

Consequences

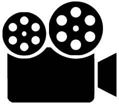
- an unwinding is a formal proof for P-security of a system
- unwindings (or bisimulations) are popular proof techniques for various security notions

Application: Automatic Analysis

How do we determine whether a system has an unwinding?

- "canonical unwinding:"
  $s \sim t$ iff for all α_1, α_2 with $\mathtt{purge}(\alpha_1) = \mathtt{purge}(\alpha_2)$, we have $\mathtt{obs}(s \cdot \alpha_1) = \mathtt{obs}(t \cdot \alpha_2)$
- how is computing this simpler than deciding P-security by the original definition?

Video Lecture: Feedback wanted



questions

- audio/video quality?
- proof presentation as screenshots, or “live writing?”
- better as video or “live Zoom session?”
- any suggestions?

feedback crucial

- your perspective very different from mine!
- constructive criticism always welcome
- review after week 6!

remember

- we’re all still learning this
- new tools, concepts
- big playground :-)

Plan for Review Sessions

purpose, timing

- used after self-study material (videos)
- purpose: discussions / questions about content (usually proofs)
    - mainly: your questions
    - some: review questions
    - **no prepared material**, that's the point!
- length/time: partial next session
    - synchronize schedule with last course iteration

this time: only one group

probably $\approx$ half of next week's session

Algorithm for P-Security

seen

P-security is characterized by unwindings

algorithmic approach

check whether unwinding exists, accept if unwinding found.

issues?

- what are "candidates" for unwindings?
- how many equivalence relations on a set with $|S|$ elements?
- candidate given by proof:

$$s \sim_u t \text{ iff } \forall \alpha_1, \alpha_2 \text{ with } \texttt{purge}(\alpha_1) = \texttt{purge}(\alpha_2) : \texttt{obs}_u(s \cdot \alpha_1) = \texttt{obs}_u(t \cdot \alpha_2)$$

- difficult to construct algorithmically!

Required: “Easier” Unwinding

lemma

If M is P-secure, then this algorithm constructs unwinding:

Input: $(S, A, \mathtt{step}, D, \mathtt{dom})$
  for each $u \in D$ **do**
    $\sim_u := \{(s,s) \mid s \in S\}$
    while elements added to $\sim_u$ **do**
      close $\sim_u$ under transitivity
      close $\sim_u$ under symmetry
      close $\sim_u$ under left respect
      close $\sim_u$ under step consistency
    end while
  end for

corollary

P-Security can be verified in polynomial time.

proof

Algorithm:

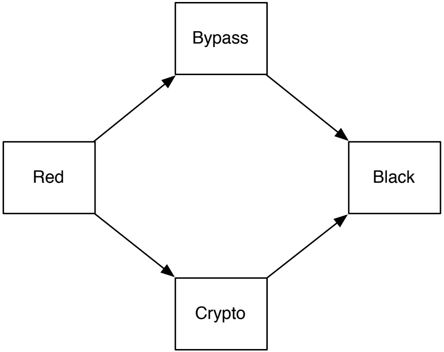
- construct $(\sim_u)_{u \in U}$ as in algorithm
- accept iff each relation satisfies output consistency

Overview

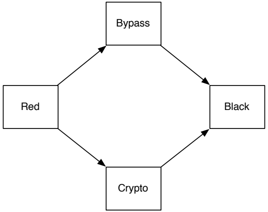
Intransitive Noninterference

P-Security

- reasonable definition of security
- assumes that policies are transitive
- intransitive policies occur in more complex scenarios



Intransitive Noninterference



issue

- information may flow from `Red` to `Black`, but must pass `Crypto` or `Bypass`
- all-or-nothing approach of P-security does not suffice

intransitive policy

$\texttt{Red} \rightarrowtail \texttt{Bypass}$, $\texttt{Red} \rightarrowtail \texttt{Crypto}$, $\texttt{Bypass} \rightarrowtail \texttt{Black}$, $\texttt{Crypto} \rightarrowtail \texttt{Black}$ (and reflexive "arrows")

goals for definition

- `Red`'s actions may have impact on `Black`'s view
- but `Black` may **only** learn of these actions "via `Bypass` or `Crypto`"
- question whether `Black` may learn of action depends on what happens *after* action

Intransitive Noninterference

downgrading

- indirect interference
- trusted "downgrader" D: declassifier, encryption device, ...
  - small enough to be formally verified
- intransitive policies:

$$H \rightarrowtail D \rightarrowtail L$$

- H's actions "transmitted" to L by actions of D
- L must not learn about H's actions directly

intransitive noninterference

meaningful semantics for intransitive policies

Intransitive Noninterference

question

- action sequence: $a\alpha$
- may L "learn" that a was performed?

$$
\begin{tikzcd}
 & B \arrow[r, tail] & D \arrow[dr, tail] & \\
A \arrow[ur, tail] \arrow[dr, tail] & & & L \\
 & C \arrow[r, tail] & E \arrow[ur, tail] &
\end{tikzcd}
$$

downgrading

transmission of actions by sequence of actions

With each action

Agent performing action "transmits" knowledge about previous events

step-by-step downgrading

- sequence ***abece***: who may "know" that a occured?
- knowledge "spreads" in each step: *a b e c e*

overview

- adaptation of P-security to intransitive case, defined in [HY87]
- replaces `purge` with `ipurge`: keeping track of “allowed interferences”

definition (`sources`)

- $\texttt{sources}(\alpha, u)$: agents who may interfere with u in sequence α
- $\texttt{sources}\colon A^* \times D \to \mathcal{P}(D)$
    - $\texttt{sources}(\epsilon, u) = \{u\}$
    - $\texttt{sources}(a\alpha, u)$ for $a \in A$, $\alpha \in A^*$: two cases
        1. there is $v \in \texttt{sources}(\alpha, u)$ with $\texttt{dom}(a) \rightarrowtail v$, then
        $$\texttt{sources}(a\alpha, u) = \texttt{sources}(\alpha, u) \cup \{\texttt{dom}(a)\}\ .$$
        2. otherwise: $\texttt{sources}(a\alpha, u) = \texttt{sources}(\alpha, u)$.

Intransitive Noninterference: IP-Security

definition ($\texttt{ipurge}$)

$\texttt{ipurge}\colon A^* \times D \to A^*$ (also: $\texttt{ipurge}_u$) defined inductively

- $\texttt{ipurge}(\epsilon, u) = \epsilon$
- for $a \in A, \alpha \in A^*$:

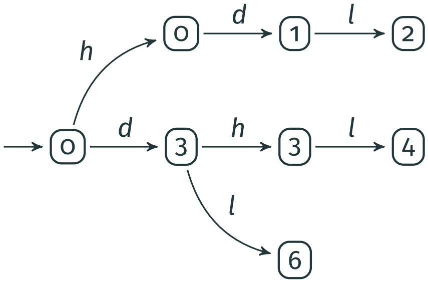
$$\texttt{ipurge}(a\alpha, u) = \begin{cases} a\texttt{ipurge}(\alpha, u), & \text{if } \texttt{dom}(a) \in \texttt{sources}(a\alpha, u), \\ \texttt{ipurge}(\alpha, u), & \text{otherwise} \end{cases}$$

definition (IP-security)

System $(S, s_0, A, \texttt{step}, D, O, \texttt{obs}, \texttt{dom})$ is IP-secure with respect to a policy $\rightarrowtail$, if for all $u \in D$, $s \in S$, $\alpha_1, \alpha_2 \in A^*$:

$$\text{If } \texttt{ipurge}_u(\alpha_1) = \texttt{ipurge}_u(\alpha_2), \text{ then } \texttt{obs}_u(s \cdot \alpha_1) = \texttt{obs}_u(s \cdot \alpha_2).$$

system



specification

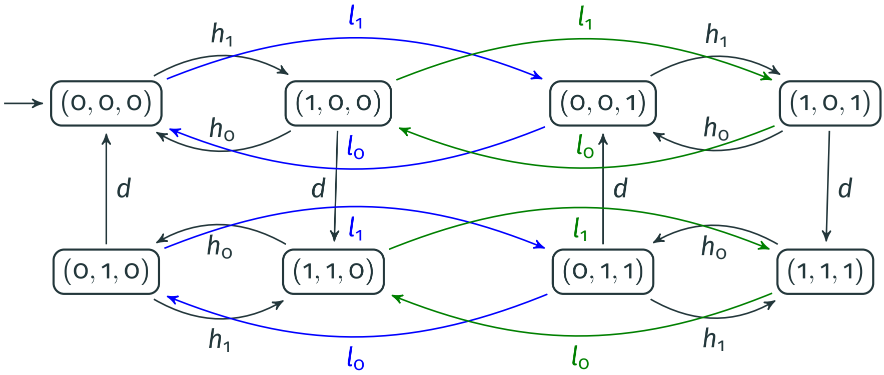
- intransitive policy: $H \rightarrowtail D \rightarrowtail L$
- actions h / d / l of agent H / D / L
- L's observations: indicated numbers

analysis

system secure?

- intuitively?
- formally?

IP-Security Example II



system

- intransitive policy $H \rightarrowtail D \rightarrowtail L$
- actions h_x, d, l_x of agents H / D / L
- $\mathtt{obs}_L(a, b, c) = (b, c)$
- system secure? intuitively, formally?

P-Security and IP-Security

question

- two security properties: P-security, IP-security
- does either implication hold? guesses?

intuition

- IP-security is “relaxation” of P-security
- agents are allowed to have more information
- leads to less-strict security property

fact

If system M is P-secure wrt. $\rightarrowtail$, then also IP-secure wrt. $\rightarrowtail$.

converse?

P-security implies IP-security

fact

If a system M is P-secure with respect to $\rightarrowtail$, then M is IP-secure with respect to $\rightarrowtail$.

proof

- assume M is P-secure
- agent u, state s, traces α_1, α_2 with $\mathtt{ipurge}_u(\alpha_1) = \mathtt{ipurge}_u(\alpha_2)$
- need to show: $\mathtt{obs}_u(s \cdot \alpha_1) = \mathtt{obs}_u(s \cdot \alpha_2)$
- enough to show: $\mathtt{purge}_u(\alpha_1) = \mathtt{purge}_u(\alpha_2)$, since M is P-secure
- general: $\mathtt{purge}_u(\alpha) = \mathtt{purge}_u(\mathtt{ipurge}_u(\alpha))$
- so: $\mathtt{purge}_u(\alpha_1) = \mathtt{purge}_u(\mathtt{ipurge}_u(\alpha_1)) = \mathtt{purge}_u(\mathtt{ipurge}_u(\alpha_2)) = \mathtt{purge}_u(\alpha_2)$

completes proof.

Unwindings for IP- TA- security?

observation

- IP and TA security are more “complex” than P-security
- for deciding security: must keep track of “who-knows-what”
- simple unwinding as in P-security not expected

verification

- IP-security and TA-security can still be decided in polynomial time
- key: unwinding conditions “between several agents”

reference

Sebastian Eggert, Ron van der Meyden, Henning Schnoor, and Thomas Wilke. “The Complexity of Intransitive Noninterference”. In: IEEE Symposium on Security and Privacy. IEEE Computer Society, 2011, pp. 196–211. ISBN: 978-1-4577-0147-4

Unwindings for IP-security

definition

An IP-unwinding for a system $(S, s_0, A, \mathtt{step}, D, O, \mathtt{obs}, \mathtt{dom})$ and a policy $\rightarrowtail$ is a family of equivalence relations $(\sim_u^v)_{u,v \in D}$ on S such that

- **OC^{IP}** if $s \sim_u^v t$, then $\mathtt{obs}_u(s) = \mathtt{obs}_u(t)$
- **SC^{IP}** if $s \sim_u^v t$ and $v \not\rightarrowtail \mathtt{dom}(a)$ then $s \cdot a \sim_u^v t \cdot a$
- **LR^{IP}** if $v \not\rightarrowtail u$ and $a \in A$ with $\mathtt{dom}(a) = v$ then $s \sim_u^v s \cdot a$

intuition

- u: observer (L)
- v: potentially secret actions (H)

theorem [Egg+13]

A system M is IP-secure wrt. $\rightarrowtail$ if and only if there is an IP-unwinding for M and $\rightarrowtail$.

Polynomial-Time Algorithm for IP-Security

corollary

IP-security can be verified in polynomial time.

proof

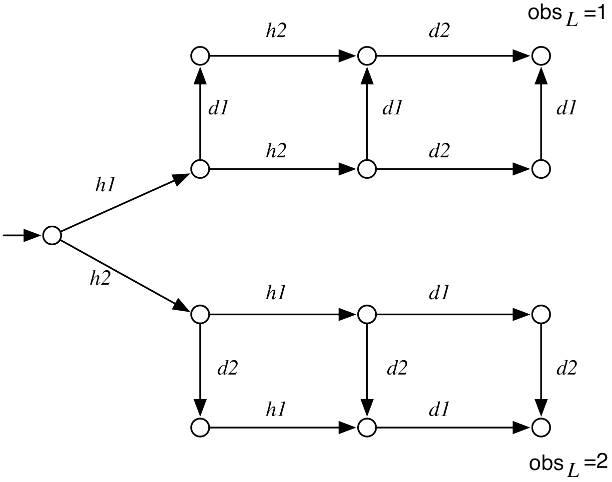
- M IP-secure wrt $\rightarrowtail$ iff all $\sim_u^v$ satisfy output consistency, where: $\sim_u^v$ smallest equivalence relation satisfying SC^{IP} and LR^{IP} with respect to $\rightarrowtail$.
- algorithm: immediately from unwinding, analogous to P-security

Overview

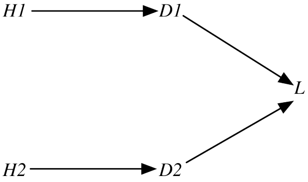
Overview

IP-Security: Is it Enough?

system



policy



analysis

system secure?

- intuitively?
- formally?

From IP to TA security

observation

- P- and IP-security only model *which* actions an agent may “learn”
- not treated: information about order of actions

fixing IP security

- modify definition to add order-information
- are we then sure we captured everything?

TA-Security: Approach

overview

- $\mathtt{ta}$-function: transmission of actions
- defines *maximal information* $\mathtt{ta}_u(\alpha)$ that agent u may have about run α
- requirement: if $\mathtt{ta}_u(\alpha) = \mathtt{ta}_u(\beta)$, then $\mathtt{obs}_u(s \cdot \alpha) = \mathtt{obs}_u(s \cdot \beta)$

reference

Ron van der Meyden. "What, Indeed, Is Intransitive Noninterference?" In: European Symposium On Research In Computer Security (ESORICS). Ed. by Joachim Biskup and Javier Lopez. Vol. 4734. Lecture Notes in Computer Science. Springer, 2007, pp. 235–250. ISBN: 978-3-540-74834-2

TA-Security: The `ta`-function

`ta` function

- models information agents may have about run
- set of actions: same approach as in IP-security
- information about *ordering*: agents have partial order view on action ordering

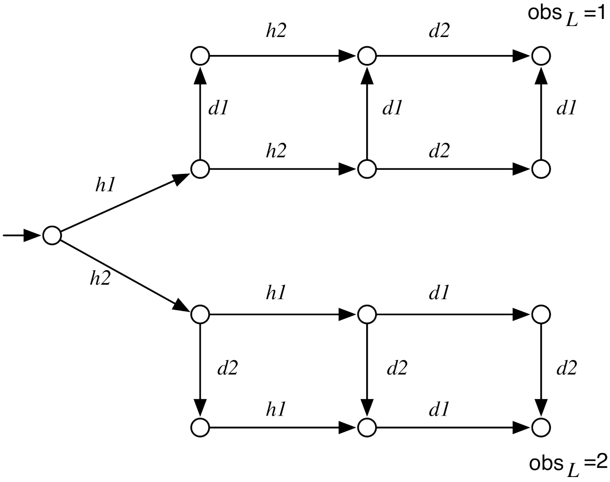
definition

For policy $\rightarrowtail$, agent $u \in D$, define $\mathtt{ta}_u$

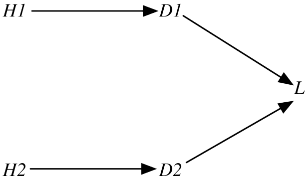
- input: $\alpha \in A^*$, output: tree of actions
- $\mathtt{ta}_u(\epsilon) = \epsilon$
- $\mathtt{ta}_u(\alpha a) =$
$$\begin{cases} \mathtt{ta}_u(\alpha), & \text{if } \mathtt{dom}(a) \not\rightarrowtail u, \\ (\mathtt{ta}_u(\alpha), \mathtt{ta}_{\mathtt{dom}(a)}(\alpha), a), & \text{otherwise.} \end{cases}$$

careful

inductive definitions extending left / right system



policy



properties

- system is IP-secure
- system is not TA-secure

Beyond TA Security?

situation

- IP security: does not cover order
- TA security: takes order into account
- no “good” examples showing we need to go beyond TA security

enough?

- “we do not have a counter-example” is not a good argument
- need to defend against **all** attacks, not just the ones we know
- there could be issues with TA security

question

- can we prove that TA security is enough?
- how would we formalize this?

Exercise

Task (implications between security properties)

In the lecture, some implications between security definitions were stated without proof. Choose and prove one of the following (in the following, M is a system and $\rightarrowtail$ a policy).

1. If M is TA-secure with respect to $\rightarrowtail$, then M is also IP-secure with respect to $\rightarrowtail$.
2. If M is P-secure with respect to $\rightarrowtail$, then M is also TA-secure with respect to $\rightarrowtail$.

Exercise

Task (equivalence for transitive policies)

Show that for transitive policies, P-security, IP-security, and TA-security are equivalent. More formally: Let M be a system, and let $\rightarrowtail$ be a transitive policy. Show that the following are equivalent:

1. M is P-secure with respect to $\rightarrowtail$,
2. M is TA-secure with respect to $\rightarrowtail$,
3. M is IP-secure with respect to $\rightarrowtail$,

Overview

Unwinding for TA

IP-security and TA-security

- TA security: IP security plus **partial** order information
- unwinding must keep track of order

notation

- $u^{\rightarrowtail} = \{v \in D \mid u \rightarrowtail v\}$
- $\mathrm{alph}(\alpha) = \{a \in A \mid \alpha = \alpha' a \alpha''\}$

definition

$\alpha, \alpha' \in A^*$, $a, b \in A$, $u \in D$. Then

$$\alpha a b \alpha' \leftrightarrow_u^{\mathrm{swap}} \alpha b a \alpha' \text{ iff } \mathtt{dom}(a)^{\rightarrowtail} \cap \mathtt{dom}(b)^{\rightarrowtail} \cap \{u, \mathtt{dom}(c) | c \in \mathrm{alph}(ab\alpha')\} = \emptyset.$$

lemma (informal)

For a system M, the following are equivalent:

1. M is TA-secure,
2. M is IP-secure and observations for “swappable” traces are identical.

Unwinding for TA

approach

two requirements:

1. system IP-secure,
2. system “respects swaps”

two unwindings:

1. IP-unwinding (known)
2. “swappable” unwinding

Unwinding for TA

Definition

A TA-unwinding for a system $(S, s_0, A, \mathtt{step}, D, O, \mathtt{obs}, \mathtt{dom})$ and a policy $\rightarrowtail$ is a family of equivalence relations $(\sim_u^{v,w})_{u,v,w \in D, v \neq w}$ on S such that

- OC^{TA} if $s \sim_u^{v,w} t$, then $\mathtt{obs}_u(s) = \mathtt{obs}_u(t)$
- SC^{TA} if $s \sim_u^{v,w} t$ and $a \in A$ with $v \not\rightarrowtail \mathtt{dom}(a)$ or $w \not\rightarrowtail \mathtt{dom}(a)$, then $s \cdot a \sim_u^{v,w} t \cdot a$
- LR^{TA} if $\mathtt{dom}(a) = v$ and $\mathtt{dom}(b) = w$ and $v \not\rightarrowtail w$ and $w \not\rightarrowtail v$, and $(v \not\rightarrowtail u$ or $w \not\rightarrowtail u)$, then $s \cdot ab \sim_u^{v,w} s \cdot ba$.

Theorem [Egg+13]

For a system M, the following are equivalent:

1. M is TA-secure,
2. M is IP-secure and there is a TA-unwinding for M,
3. there is an IP-unwinding and a TA-unwinding for M.

Corollary

TA-security can be verified in polynomial time.

Overview

TA Security and Knowledge

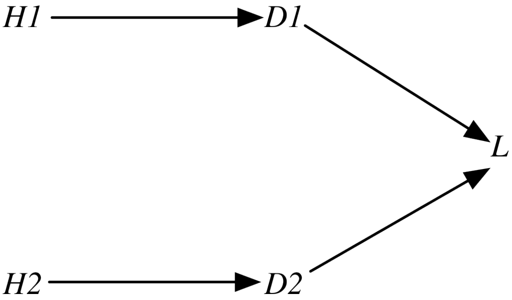
security definition goal

- u should only learn about system input (actions) as allowed by policy
- approach: compare
  - "allowed knowledge" ($\mathtt{purge}/\mathtt{ipurge}/\mathtt{ta}$)
  - "actual knowledge" ($\mathtt{obs}$)

question

- what should "allowed knowledge" be?
  - $\mathtt{purge}$, $\mathtt{ipurge}$, $\mathtt{ta}$, …
- what is a "sound" definition of allowed knowledge?
- what kind of properties do we want?

Approach



basic idea

- policy determines "allowed knowledge"
- L should only have information obtained via flows through policy
- L should only have information that D_1 and D_2 have "together"

recall issue

L has information about order of H_1 and H_2 events that D_1 and D_2 do not have

- individually, or
- "as a team."

Detour: Defining Knowledge

abstract point of view

- Q set of possible situations (states)
- properties: subsets of states
- agents $u_1, \ldots, u_n$: partial view
  - each i: eq. relation $\sim_i$ on Q
  - $q_1 \sim_i q_2$: indistinguishable for u_1
  - (e.g., same observations)

knowledge of u_i in state q

- P property of states
- agent u_i **knows** P holds in q iff:

$$q' \in P \text{ for all } q' \text{ with } q' \sim_{u_i} q.$$

- write $q \models K_{u_i} P$

group knowledge

- agent group $G \subseteq \{u_1, \ldots, u_n\}$
- common information of G?
- candidates: distributed, shared, common knowledge

- $\sim_G^D = \cap_{i \in \{1,\ldots,n\}} \sim_i$
- $\sim_G^S = \cup_{i \in \{1,\ldots,n\}} \sim_i$ (not necc. eq-rel)
- $\sim_G^C$: reflexive, transitive closure of $\sim_G^S$

Application to TA-Security

knowledge and TA policies

- situations/states: action sequences
- $\alpha_1 \sim_u \alpha_2$: $\texttt{ta}_u(\alpha_1) = \texttt{ta}_u(\alpha_2)$
- defines **allowed** knowledge
- contrast: **actual** knowledge

theorem [Mey08]

In a TA-secure system: If $\alpha \models K_u P$, then $\alpha \models K_D P$, where

- D contains all agents v with $v \rightarrowtail u$
- knowledge of D: distributed knowledge $\sim_D = \bigcap_{v \in D} \sim_v$

(holds if $\rightarrowtail$ is acyclic)

informal

TA security: u may not learn anything that does not follow from distributed allowed knowledge of agents allowed to interfere with u.

Exercise

Task (strong secrecy and non-interference)

As suggested by the ProVerif keyword **noninterf**, strong secrecy of cryptographic protocols and non-interference (in the information-flow sense) are related. In this exercise, we will make this relationship more precise. For this, use a simple cryptographic protocol and construct a system M such that strong secrecy of the protocol directly corresponds to P-security of the system M.

Note: Depending on the protocol you choose to model, a finite state system might require a mechanism to limit the possible number of terms and thus the state space. To avoid this, you may use systems with an infinite state space to solve this task.

Unwindings: Summary

overview

- approach: pairs (s_1, s_2) of states that should have same observations
- unwindings give easy fixpoint algorithms
- lead to polynomial-time algorithms in all cases
- stronger result: decidable in non-deterministic logarithmic space

security notions

- works for: P/IP/TA-security
- fails for TO-security (defined in [Mey07])

comparison with protocols

why verification so much easier?

Information-Flow Summary

asynchronous information-flow

- P $\rightarrow$ TA $\rightarrow$ IP
- structurally very similar
- characterization and efficient algorithms with unwindings

synchronous information-flow

- RES $\rightarrow$ NDS $\rightarrow$ NDI
- structurally different: unwindings, views/strategies, views/sequences
- efficient algorithm for RES (unwindings)
- graph exploration algorithm for NDI

  (NDS EXPSPACE-complete)

References i

Sebastian Eggert, Ron van der Meyden, Henning Schnoor, and Thomas Wilke. "The Complexity of Intransitive Noninterference". In: IEEE Symposium on Security and Privacy. IEEE Computer Society, 2011, pp. 196–211. ISBN: 978-1-4577-0147-4.

Sebastian Eggert, Ron van der Meyden, Henning Schnoor, and Thomas Wilke. "Complexity and Unwinding for Intransitive Noninterference". In: CoRR abs/1308.1204 (2013). URL: http://arxiv.org/abs/1308.1204.

Joseph A. Goguen and José Meseguer. "Security Policies and Security Models". In: IEEE Symposium on Security and Privacy. 1982, pp. 11–20.

J. Thomas Haigh and William D. Young. "Extending the Noninterference Version of MLS for SAT". In: IEEE Trans. on Software Engineering SE-13.2 (Feb. 1987), pp. 141–150.

Richard J. Lipton and Kenneth W. Regan. Timing Leaks Everything. 2018. URL: https://rjlipton.wordpress.com/2018/01/12/timing-leaks-everything.

Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. "Meltdown". In: ArXiv e-prints (Jan. 2018). arXiv: 1801.01207.

Ron van der Meyden. "What, Indeed, Is Intransitive Noninterference?" In: European Symposium On Research In Computer Security (ESORICS). Ed. by Joachim Biskup and Javier Lopez. Vol. 4734. Lecture Notes in Computer Science. Springer, 2007, pp. 235–250. ISBN: 978-3-540-74834-2.

Ron van der Meyden. "On Notions of Causality and Distributed Knowledge". In: KR. Ed. by Gerhard Brewka and Jérôme Lang. AAAI Press, 2008, pp. 209–219. ISBN: 978-1-57735-384-3.

John Rushby. Noninterference, Transitivity, and Channel-Control Security Policies. Tech. rep. CSL-92-02. SRI International, Dec. 1992. URL: http://www.csl.sri.com/papers/csl-92-2/.

Christof Windeck. PC und Notebook senden auf Mittelwelle - ohne Zusatz-Hardware. 2018. URL: https://www.heise.de/ct/artikel/PC-und-Notebook-senden-auf-Mittelwelle-ohne-Zusatz-Hardware-3948910.html.