Engineering Secure Software Systems

January 26, 2021: Information Flow: Introduction, P-Security

Henning Schnoor

Institut für Informatik, Christian-Albrechts-Universität zu Kiel

Admin: Evaluation, Exam

Lecture Evaluation

running since last week

- everybody got an access code?
- use “free text” fields

Oral Exam

date

- Tuesday, February 23
- Wednesday, February 24
- let me know if you want to cancel!
- each exam: $\approx$ **25** minutes

what to expect?

- questions cover all lecture aspects
- theory lecture: precise formal knowledge of key definitions required for discussion
- sequence: definitions, results, proofs, alternatives, ...

preparation

- use available material: slides, notes, exercises
- **“readiness indicator:” review questions**

organization

- oral exam via BigBlueButton
- registration until Sunday, February 14: `https://www-ps.informatik.uni-kiel.de/pruefungsanmeldung/`, access code: 1O1BIS

Part II: Information Flow

Overview

Part II: Information Flow

Examples

lecture up to now

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

→ 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

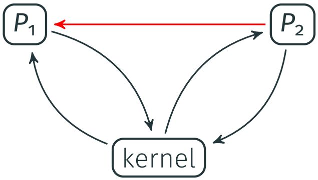
highlights

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

Meltdown as Information Flow Issue

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

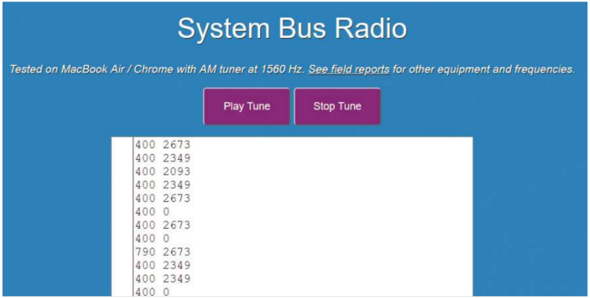
- `https://github.com/fulldecent/system-bus-radio`
- Christof Windeck. PC und Notebook senden auf Mittelwelle - ohne Zusatz-Hardware. 2018. URL: `https://www.heise.de/ct/artikel/PC-und-Notebook-senden-auf-Mittelwelle-ohne-Zusatz-Hardware-3948910.html`

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-

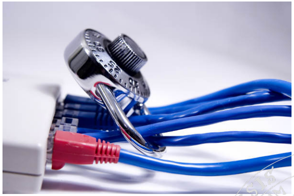
Information Flow Motivation

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

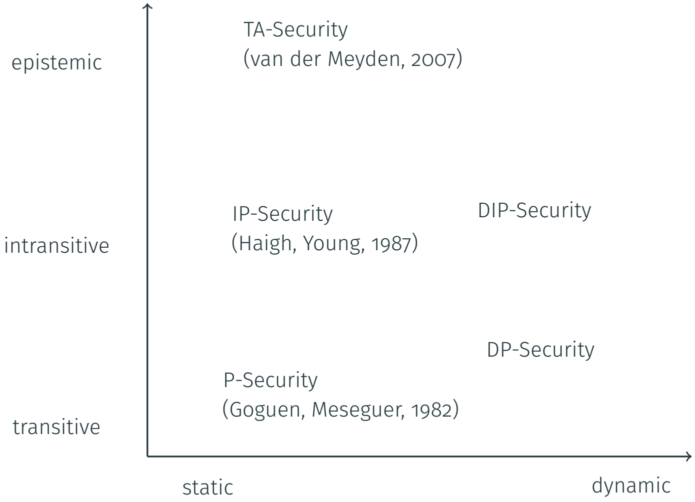
aspects of noninterference

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

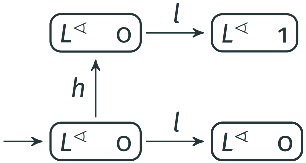
as usual: no “one-size-fits-all” approach

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

Non-Interference Notions



system



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

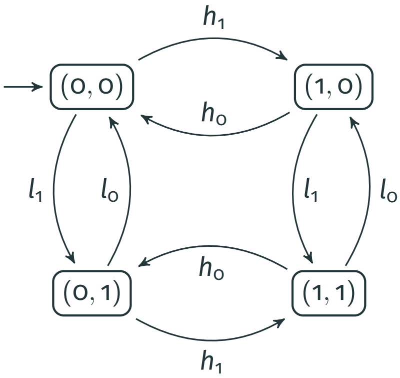
analysis

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].

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”

definition (P-security)

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

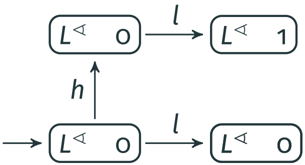
$$\text{If } \mathtt{purge}_u(\alpha_1) = \mathtt{purge}_u(\alpha_2)\text{, then } \mathtt{obs}_u(s \cdot \alpha_1) = \mathtt{obs}_u(s \cdot \alpha_2).$$

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

P-Security Example

system



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

analysis

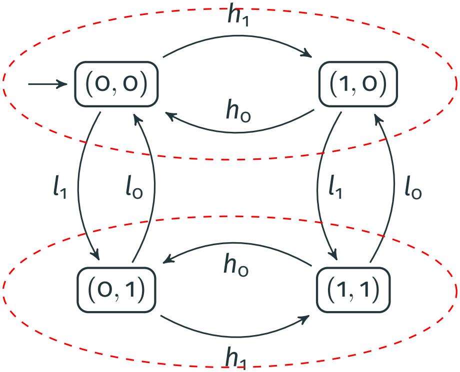
system secure?

- intuitively?
- formally?

insecure

- α_1 : l, then $\texttt{purge}_L(\alpha_1) = l$
- α_2 : hl, then $\texttt{purge}_L(\alpha_2) = l$
- $\texttt{obs}_L(q_0 \cdot \alpha_1) = 0 \neq 1 = \texttt{obs}_L(q_0 \cdot \alpha_2)$

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?

Exercise

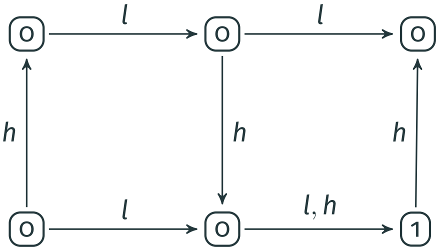
Task (P-Security Example I)

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

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

Task (P-Security Example II)

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



Exercise

Task (alternative definition of P security I)

Let $M = (S, s_0, A, \mathtt{step}, D, O, \mathtt{obs}, \mathtt{dom})$ 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

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

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.

Overview

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 $(S, s_0, A, \mathtt{step}, D, O, \mathtt{obs}, \mathtt{dom})$ 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

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)

Video Lecture

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

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 $\mathtt{purge}$ instead of $\mathtt{purge}_u$, $\sim$ instead of $\sim_u$, $\mathtt{obs}$ instead of $\mathtt{obs}_u$

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 $\rightarrow$ 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 $s \cdot \alpha \sim s \cdot \mathtt{purge}(\alpha)$

recall step consistency

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

proof: induction over $|\alpha|$

$\alpha = \epsilon$ $\quad s \cdot \alpha = s \cdot \epsilon = s \sim s = s \cdot \epsilon = s \cdot \mathtt{purge}(\epsilon)$, 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 1: $\mathtt{dom}(a) \to u$

$$\begin{array}{lcl} \text{from step consistency and } s \cdot \alpha \sim s \cdot \mathtt{purge}(\alpha)\text{:} \quad s \cdot \alpha a & \sim & s \cdot \mathtt{purge}(\alpha) a \\ \text{since } \mathtt{purge}(a) = a\text{:} & = & s \cdot \mathtt{purge}(\alpha)\mathtt{purge}(a) \\ \text{since } \mathtt{purge}(\alpha a) = \mathtt{purge}(\alpha)\mathtt{purge}(a)\text{:} & = & s \cdot \mathtt{purge}(\alpha a) \end{array}$$

Proof of Key Fact (Part II)

Claim (Key Fact)

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

recall left respect

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

proof: induction over $|\alpha|$

$\alpha = \epsilon$ $\quad s \cdot \alpha = s \cdot \epsilon = s \sim s = s \cdot \epsilon = s \cdot \texttt{purge}(\epsilon)$, 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 2: $\texttt{dom}(a) \not\rightarrowtail u$

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

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 $\mathtt{purge}(\alpha_1) = \mathtt{purge}(\alpha_2)$, then $\mathtt{obs}(s \cdot \alpha_1) = \mathtt{obs}(s \cdot \alpha_2)$

Key Fact

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

proof

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

recall output consistency

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

completes proof of first direction

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

Proof Structure

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

OC^P	if $s \sim t$, then $\mathtt{obs}(s) = \mathtt{obs}(t)$	output consistency
SC^P	if $s \sim t$, then $s \cdot a \sim t \cdot a$	step consistency
LR^P	if $\mathtt{dom}(a) \not\rightarrowtail u$, 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 $\mathtt{purge}(\alpha_1) = \mathtt{purge}(\alpha_2)$, we have $\mathtt{obs}(s \cdot \alpha_1) = \mathtt{obs}(t \cdot \alpha_2)$

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 $\texttt{purge}(\alpha_1) = \texttt{purge}(\alpha_2)$, we have $\texttt{obs}(s \cdot \alpha_1) = \texttt{obs}(t \cdot \alpha_2)$

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

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

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

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

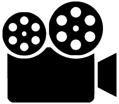
- an unwinding is a formal proof for P-security of a system
- unwindings (or bisimulations) are popular proof techniques for various security notions

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