جلسه اول

Subject اصول روش های فرمال و امنیت

Date ۱۳۹۴/۱۱/۱۰

Formal Approaches to Information Security

(صوری)

- Formal method is a method for modeling, specification, and verification of systems which is based on mathematics. (Dijnes Bneer)

- به زبان ساده تر فرمال ریاضیات هست. مثل logic, co-algebra, co-induction و ... که از طریق ریاضی (کامپیوتر) به زبان کامپیوتر ترجمه است.

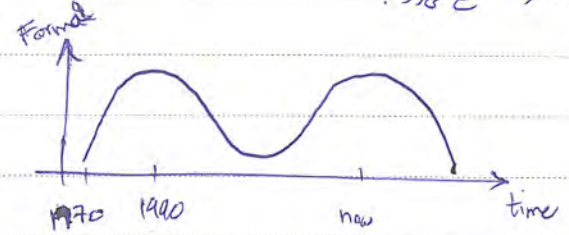
- روش های فرمال یک طیف است از light تا heavy.

model checking, type system ← light

higher-order logic → heavy

- Z, RSL, CSP و ... زبان های توصیف مورد بررسی نرم افزارند برای روش های فرمال هستند. (spi)

- نمودار two-elephant برای رشد علوم و دانش را نشان می دهد:



- شاید کمی استفاده از روش های فرمال دردسر بوده اما راهی جز این روش وجود ندارد.

- علوم دیگر مهندسی مبتنی بر Foundation و تئوری هست و روش Calculus هست اما در مهندسی نرم افزار حضور اینقدر پررنگ methodical نیست اما در راه توصیف و طراحی سیستم ها نقش بسیار مهمی دارد.

Subject
Date

- For a formal approach to Security: (Medows مقاله)

  - A well-defined specification of the system → مثلا: Z یا Pi-Calculus
    اجزاء را کاملا توصیف می‌کنیم.
  - An adversary model
    - who are adversaries?
    - what are their goals?
    - what are their capabilities?

    سوال سخت خواهد است
    مدل Dolev-yao
  - A specification of correct behavior → Security Requirement
  - An effective procedure for determining that the system behaviors correctly in the face of adversaries.

اثبات شود که سیستم دارای خاصیت است که سیستم درخواهد با adversary به درستی رفتار کند.

- تفاوتی که مورد اول این است که توصیف سیستم برای program سخت است. پس این را داریم تقریبا.

- حمله لزوما نیست قوی تر attacker را درنظر گیریم. موردهای رایج: environment و اشتباهات برنامه و...

- در همه مدل‌های خوبی باید این مراحل وجود داشته باشد.

- در قسمت دوم اول سوال این است که خود سیستم را کامل بدانیم یا هنوز non-deterministic دارد؟

Specification با specification شروع می‌شود، بعضی از اولی‌های که موجود نیست. سپس correct behavior را

دقیق تر می‌کنیم (Refinement) و پس از آن با ابزارهایی Verify می‌کنیم که در قسمت دوم اول

مدل adversary را دارد فقط.

مقاله: A Symbiotic Relationship between Formal Methods and Security,
Jeanette Wing, 1998.

Subject Formal
Date 94.11.10

درس را به دو بخش تقسیم می‌کنیم: 1) Information-Flow Security 2) Protocol Analysis

منبع دربخش دوم کتاب Handbook of Model Checking خواهد بود.

البته برای پوشش دوبخش هست به طرح درس

* اما سوال اصلی این است که چه Policy هایی را چگونه enforce کنیم.

روش‌های اعمال:
- Static → قبل از اجرا → ابزار: model checking, type system ⇒ فقط decidable ها را می‌توان enforce کرد.
- Dynamic (Run-time) → در زمان Run برنامه → ابزار: monitor ⇒ co-recursively enumerable ها را می‌توان enforce کرد.

بخش اول برنامه‌های Policy را پذیرفته نمی‌کند و در غیر این صورت آن را اصلاح می‌دهد.

ارزیابی: سیستم، پارامتر، ارائه‌دهنده، تحلیل

مقدمه‌ای بر Security:

ابتدا با این شروع می‌کنند که چطور روش‌های فرمال لازم است.

این axiomatic است که ما 100٪ Secure نداریم. پس سعی داریم با ثابت کنیم 100٪ این را داریم.

که برای این مجموعه‌ای از فرضیات درباره محیط سیستم، سیستم طراحی می‌شود. به همین دلیل دائم فرضیات را می‌توانیم

سعی کنیم که همه چیز هم در مورد functional, non-functional ها درست شده باشد هم داشته باشیم.

پس روش‌های فرمال در محدوده فرضیات اثبات و ارائه می‌دهد.

PdPCO

Subject

Date 04-10-12

همین که چیزی که در ذهن است برای سیستم، باید روی کاغذ اولین Specification. بین این دو gap وجود دارد.

روش‌های فرمال این‌ها را که نمی‌توانند درست کنند!

کارهایی که فرمال می‌تواند انجام دهد:

- Delimit the system's boundary
- Characterize a system's behavior more precisely
- Define the system's desired properties precisely
- Prove a system meets its specification

"اثبات می‌شود" → اطمینان می‌آورد → proof ≠ verification ≠ justification

درستی

اگر یک سناریوی حمله، می‌توان مثال نقضی آورد و نشان داد می‌شود، اثبات کرد، سیستم insecure است.

Formal Specification
↓
مشخصات سیستم
جمع سیستم
کارهای سیستم

Formal Verification
↓
اثبات (درستی)

روشی برای سیستم‌های بزرگ:
که روش‌های فرمال می‌توانند انجام دهند

میزان اطمینانی که روش‌های مختلف دارند: توجیه < تست < اثبات

سال 2000

Past

مرکز روی تعریف و تعریف سیستم امن

وجود سیستم عامل

نمونه: Biba, BLP, CW, System Z

BLP و *-property

1st One must state the property of the system to
2nd prove, as expressed explicitly in a formal specification
3rd prove the property.
The proof.

~~One must model the system so that one can formally~~

→ Semantic (e.g. state machine)
→ Syntactic (e.g. logical or algebraic expression)

اثبات درستی سیستم ← Proof برای سیستم

Subject Formal
Date
94.10.12

در رویکرد Semantic، سیستم را با model می‌سنجیم. در رویکرد Syntactic هم ماهیت دارد بررسی

هم سوال است که روش‌های مختلف دارد. مثلاً از روش‌های Syntactic، logical expression است، از

$$\frac{P \quad P \rightarrow q}{q} \text{ (MP)}$$

deduction برای اثبات استفاده شده.

در چنین از دل ماهیت به دست آمده و این‌ها به ظاهر Syntax با دارایم و استدلال می‌کنیم

System model $\Longrightarrow$ System Property (Theorem Proving)

این رویکرد بیشتر از هر چیزی مثبت دارد.

سردمداران: Zohar Manna, Amir Pnueli

(Temporal Verification of Reactive Systems) (Models of Reactivity کتاب)

(TLA, Lamport)

در این رویکرد، مثلاً Induction است، سیستم را با State Machine مدل می‌کنیم و

ماهیت برای قرار می‌گیرد. البته خود این هم ثابت (Induction به روش اثبات ثابت)

رویکرد Model checking، سیستم State Machine است، property را با logic بیان می‌کنیم و

هم فضای State ها را روی آن بررسی می‌کنیم که هیچ State برای نقض

رویکرد Semantic، Syntactic روش اثبات را هم برای مشخص می‌کند که معنی‌دار است و از

روش‌های دقیق اثبات استفاده کرد.

PgPCO

5/

Subject Formal
Date 99.11.12

Hybrid ، Model checking ، Theorem Proving : Verification

NRL

type system

The Orange Book

Assurance ، (AI)

HDM ، Gypsy ، FDM ، Affirm ... Prover

NRL ، Interrogator ، BAN ، GNY ...

automated proof

Specification

Proof

Z ، RSL ...

Theorem Proving ، model checking :

99.11.17

2000

Past | Present | Future

Relation and Communication between Formal Methods and Security

Part : Orange Book ، BLP ...

Formal Verification

Theorem Prover

Subject Formal
Date

فصل ۱۱، ۱۷ Term-rewriting

- NRL , Interrogator : هم تحت Security

از اولین نمونه های ابزارهای قوی ارائه شده اختصاصی برای کشف و تحلیل پروتکل های امنیتی بود.

Dolev-Yao مدل پرکاربرد

- attacker ابزارهای اختصاصی : طراحی primitive های امنیتی را با توجه به توانایی های در نظر گرفته شده برای تعریف encryption و تحلیل

ابزارهای عام محدوده یا حاظر scope زیادتر، غیر efficient تر. پس ابزار وارد کردن برای کاربردهای خاص

کارآمدتر تر شود. در ابزار مثلا باید تعریف attacker هم توسط کاربر انجام شود.

Backward

- در Interrogator ، یک روش یا حالت به سیستم دادیم ، شروع می کند به استنتاج کردن اینکه آن حالت می تواند

NRL هم به همین صورت است. شروع کردن از حالت نهایی است، یک روش معمول

این دو ابزار پشتیبانی کمتری Security دارند است

- مدل سوم، BAN است. اولین تجزیه و تحلیل است برای Authentication پروتکل ها. Model logic

(1990)
(Belief logic)

قواعدی را مطرح می کند که توسط آن باورهای پروتکل شکل می گیرد. به این ترتیب،

Authentication را تحلیل می کند. تعریف خاصیت Authentication، شرح نحوه Authentication درست است.

یعنی در آن belief ای می شود تصدیق اصالت درست انجام شده. آن تصدیق اصالت به صورت ضمنی تعریف می شود.

مطابق BAN : یک نشانه خوب درست است Authentication وجود دارد به شرطی که بعضی از پروتکل که

اشکال داشته باشد درست نباشد — ابزاری برای تحلیل نیست. (روش های رسمی ابزاری برای تحلیل، کاوش (ابزار)

۷/

Subject
Date

ابزارهم دارد.

روش‌های BAN و ... برای کشف حمله‌ی GNY ، SVO ، AUTLOG

شکل مقدماتی را معرفی کنند. (مثل BAN) اما هنوز شکل جدی نشد.

* هیچ روشی (و البته هنوز امید) برای Semantics است، نه صرف Syntax.

یک تعاریف جدی برای هر شکل تعریف Auth. ارائه و تعریف Woo and Lam برای آن معرفی شد.

مقاله A Semantic Model for Authentication Protocols, Security and Privacy, 1993. IEEE

که تعریف auth. به خوبی بیان شده. (Correspondence تعریف)

↓

چه می‌شود که auth. نقض می‌شود.

یک attacker محتمل طرف در مقابل را به نحوی می‌یابد که این دو جمله‌ی شرط‌های که مراحل به درستی انجام شده‌اند.

(Correspondence assertion) ← نمایش داده شود. و معمولاً از end با begin نوشته و نقض شود.

در زمان present (از 1996 شروع شده) :

man-in-the-middle است و internal attack است! Needham-schroder از Lowe

که خوشبختانه ابزاری دارد.

مشکل این است که BAN این attack را نمی‌تواند تعریف Correspondence آن را نقض می‌داند.

با استفاده از زبان CSP که FDR model checker آن است، توانست این را نشان دهد.

حوزه‌ای شلوغ و کاری Authentication.

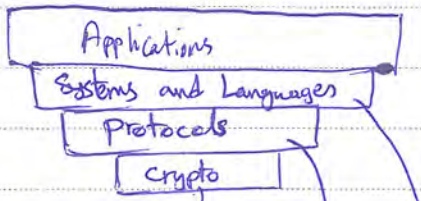
Subject Format
Date 98/11/17

خود Lowe بعدی agreement را برای Correspondence تعریف کرد.

در زمینه Future (از 2000 به بعد): (یعنی بحثی از آینده برای پروتکل‌ها که تا الان در 2016 است)

System Layers:

لایه‌های سیستمی را به این شکل در نظر گرفتیم:



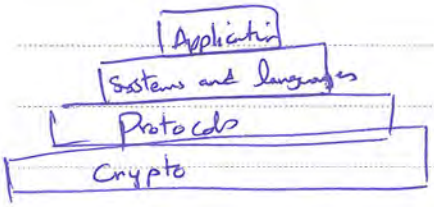
Primitive های پایه در هر کاری نیاز است. مثل enc, hash و ...

برای پیاده‌سازی پروتکل‌ها Crypto Primitive ها را فراخوانی می‌کنیم.

برای پیاده‌سازی پروتکل‌ها

Crypto در پروتکل‌های مختلف استفاده می‌شود، پروتکل‌ها در زبان‌های برنامه‌نویسی مختلف پیاده‌سازی می‌شوند و برای حکم App های مختلف کاربرد دارند.

اما معماری در رابطه امنیت به شکل زیر است:
(تضمین)



پایین‌ترین لایه‌ها Crypto Primitive هستند. مثلا RSA که توسط همه در همه جا استفاده می‌شود.

معماری و اثبات کمی دارد. یعنی چه الان‌های اصلی را تضمین و اثبات خوبی دارد اما در کاربرد آن در دنیای مختلف چه کار می‌کند.

نیازمند ماشین حداکثر است. مثلا چون NP است اما در عمل ... می‌شود، پس معماری خوبی دارد.

Subject
Date

در این درس راه Crypto را از دیدگاه Provable Security می‌بینیم (دیدگاهی که امروزه در آن غالب است).

سوالات تحقیقی را به دو دسته تقسیم می‌کنیم:

N (Near-term)	L (long-term)
Program Analysis Tools	Protocol Composition
Certified Library Components	(پروتکل‌هایی که ترکیب پروتکل‌ها را بررسی می‌کنند)
Benchmark Suite	آیا می‌توان امنیت اجزا را (مثلاً modular)
Programming Language Design → JIF مثل	

(زبان‌هایی که Specify و Policy را هم پشتیبانی می‌کنند و خود کامپایلر آن امنیت و حفاظتی را تضمین می‌کند.)

Case study → Application در مقابل

- در کتاب پروتکل را از ۲ دیدگاه (پروتکل و مطلوبیت) بررسی کرده‌ایم، اما جالب است به این بخش هم بپردازیم.

ابزارهایی مثل Tamarin، Sytter، Cryptoverif، F*، F5، F7 (crypto)، Proverif و ... (پیشرفته‌ترین ابزار)

ساخته شده و ابزارهای بسیار کاربردی هستند. پروتکل‌ها به راحتی تحلیل می‌شوند. (Computational هم مهم است.)

اشکال بعضی از ابزارهای بالا این است که گاه مدل پروتکل را کامل می‌کنند، یا به راحتی از عهده آن برنمی‌آیند.

ممکن است termination تضمین نشود. (البته خود سوال هم undecidable است.)

یعنی هم ممکن است والنا تصور کند که این پیام‌ها را دارد، در حالی که ندارد!

Subject Formal
Date ۹۴/۱۱/۱۷

Language-based Security

- در طراحی زبان‌های برنامه‌نویسی، Jif، TAL، PCML و ... خروجی‌ها هیچ‌وقت وارد وری نمی‌شوند. near-timed

- اکثر ابزارهای موجود در بازار، کاربردشان را ثابت کرده هم نوعی پرسرعت است.

اما Future های ما: ابزارهای ساخته شده کاربردی شوند.

- زبان‌های برنامه‌نویسی هم اسمبلی هم زبان‌های سطح بالای عام منظوره، تضمین شده امنیت را در برنامه‌هایشان دارند.

- حوزه‌های کاربردی جدید مثل Cryptocurrency، Cloud، Distributed Databases، E-voting و ... به روش‌های اثبات‌پذیر

enforcing به هم تضمین خواص دهم

- یک خروجی باید از چه ابزارهایی وجود دارد. اساساً هیچ خروجی را نمی‌توان Enforceable Security Policies نامید.

۹۴/۱۱/۱۹

Information-flow Security:

بیت بدهم چنین داده‌هایی، می‌دهیم flow داشته باشیم. اینه داده‌ها میزان حساسیتشان مشخص شود و اولین کاربردش این است.

داده‌ها را از نظر Sensitive بودن به دسته‌های مربوطه می‌دهد.

داده‌های از جنس A به جنس B: $A \xrightarrow{\text{Flows}} B$

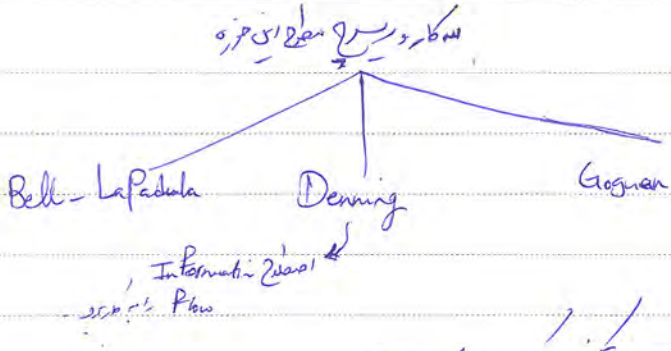
داده‌های از جنس B. مثلاً Secret ↛ classified و classified → Secret

کل داستان اینه که می‌خواهیم جریان‌ها را کنترل کنیم و جلوی حذف خاصی صورت بگیرد.

* یک سیستم این است اگر همگی flow ای مجاز خاصی اتفاق بیفتد. Secure

از سال‌های ۱۹۷۴، ۱۹۷۵ شروع شده

Subject
Date



Formal

Access Control

The Safety Problem

Access Control Matrix

(HRU)
Harrison
Ruzzo
Ullman

Discretionary Access Control

grant

Access Control → The Safety Problem

undecidable

HRU

NP

Grant

BLP

linear

Safety

BLP

Safety Problem is decidable in BLP models.

Subject Formal
Date 9t 11,19

پس BLP هم از نوع کنترل دسترسی و هم از نوع Information Flow است.

BLP:

یک مدل است برای Confidentiality و یک نوع از کنترل دسترسی Mandatory Access Control

* هرچند تعریف رسمی مدل، سه جزء لازم داریم: سیستم، requirement، Verification

سیستم و اجزای آن ← در BLP چه چیزهایی هستند.

requirement ← که امنیت سیستم را، در کدام شرایط، تضمین می‌کند.

Verification ← سیستم را امنیت است.

Definition – Suppose $L = \{l_1, l_2, \ldots, l_p\}$ where $l_i = (c_i, k_i)$, $c_i \in C$, $k_i \in K$. The dominance relation $\gg$ on L is defined as follows:
L → a family of sets

$(l_i, l_j) \in \gg$ or $l_i \gg l_j$ iff

1) $c_i \geq c_j$, and
2) $k_i \supseteq k_j$
→ Superset → Partial ordering

where $\geq$ is a total ordering on C.

- Total order → هر عنصر حتما با هم رابطه دارند.
- partial order → لزومی ندارد همه عضوی با هم رابطه باشند.

هر عنصر L دارای Security Label می‌باشد. مدل Flow کنترل از این نوع است، زیرا dominance

Subject
Date

Proposition – $(L, \preceq)$ is a poset.
(partially order set)

Notation –

$S = \{s_1, s_2, \ldots, s_n\}$ Subjects : processes, Programs in execution (Active Entity)

$O = \{o_1, o_2, \ldots, o_n\}$ Objects : Data, Files, Programs, I/O, ... (Passive Entity)

$O \supseteq S$ → active entity هم می‌تواند object باشد.

$C = \{c_1, c_2, \ldots, c_q\}$ classification — clearance level of a subject
$c_1 > c_2 > \ldots > c_q$ classification level of an object

K is a set.

$A = \{r, e, w, a\}$ Access Attributes

observe	alter	
×	×	e
×	✓	r
✓	×	a
✓	✓	w

Subject Formal
Date 96,11,19

$RA = \{g, r\}$ request elements

g: get, give

r: release, rescind

$S' \subseteq S$ subjects subject to *-property

$R = \bigcup_{1 \le i \le 5} R^{(i)}$ Requests

Request جنس

$R^{(1)} = RA \times S \times O \times A$

e.g. $(g, s, o, r) \rightarrow$ s درخواست گرفتن دسترسی r را روی o

$R^{(2)} = S \times RA \times S \times O \times A$

subject ای درخواست کرده دسترسی را از subject دیگری برای object ای تغییر بدهد.

$\vdots$

$R^{(5)} = S \times L$

subject درخواست تغییر لیبل کرده است.

$D = \{yes, no, error, ?\}$ Decisions تصمیم‌ها

تصمیم

$d_m \in D$

وقتی درخواستی می‌آید یکی از تصمیم‌ها را برمی‌گرداند.

$F \subseteq L^S \times L^O \times L^S$ Security Label Vectors (level)

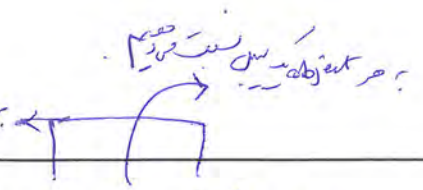
L^S: مجموعه‌ی برچسب‌های S

L^O: مجموعه‌ی برچسب‌های O

L^S: مجموعه‌ی برچسب‌های S

Subject
Date

هر subject به یک سطح امنیتی تعلق دارد.



$f = (f_S, f_O, f_C) \in F$

label function

f_S: max level

f_C: current level — سطح فعلی که این subject روی آن قرار دارد.

$X : R^T$ — مجموعه همه توابع از T به R ⇒ Request Sequences

$T = \{1, 2, \dots, t, \dots\}$ زمان

دنباله‌ای از درخواست‌ها

$x \in X$ یک دنباله از درخواست‌ها

$Y : D^T$ — مجموعه همه توابع از T به D ⇒ Decision Sequences

$y \in Y$

M Access matrices

سطرها subject ها، ستون‌ها object ها

$B = \wp(S \times O \times A)$ Current Access Set

$\wp$: Powerset

مجموعه دسترسی فعلی

$b \in B$ یک مجموعه است که در آن عناصری به این شکل وجود دارد.

در وضعیت فعلی، چه subject‌ای چه دسترسی‌ای روی چه object‌هایی دارد.

در هر لحظه نیاز داریم بدانیم، مجموعه این سه‌تایی‌ها را می‌توانیم داشته باشیم.

Subject Formal
Date 98.11.19

$V = B \times M \times F \times H$ states

state $v \in V$

Configuration ... subject ... object ... H ... hierarchy

... owner ... subject ...

... state machine ... (Semantic) ...

$Z = V^T$ → ... state Sequence ... State Sequences

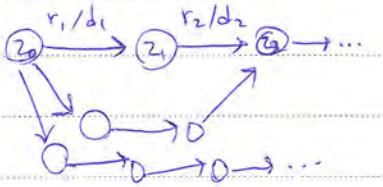
$z \in Z$

z_t : t-th State in Z ⇒ z ... t-th

Definition – Suppose that $W \subseteq R \times D \times V \times V$.

... → System Transition

The System $\Sigma(R, D, W, z_0) \subseteq X \times Y \times Z$,



... (... ...)

is defined by

$(x, y, z) \in \Sigma(R, D, W, z_0)$ iff

$(x_t, y_t, z_t, z_{t-1}) \in W$ for each $t \in T$, where z_0 is an initial state of the system, usually of the form (ϕ, M, f).

Subject
Date

Notation - $b(s:x,y,\dots,z) = \{o \mid (s,o,x) \in b \overset{or}{\vee} (s,o,y) \overset{\in b}{\vee} \dots \vee (s,o,z) \in b\}$

Definition - A state $v = (b, M, f, h)$ satisfies the Simple-Security Property

(SS-Property) iff $\forall s \in S.\ o \in b(s:r,w) \Rightarrow f_S(s) \geq f_O(o)$.

$\forall o \in O$ — max level

Definition - $(s,o,x) \in b$ satisfies the simple security property relative to f (SSC rel f) iff

1) $x \in \{e, a\}$ or

2) $x \in \{r, w\}$ and $f_S(s) \geq f_O(o)$

Subject Formal
Date ۹۴/۱۱/۲۴

Proposition – A state $v=(b,M,f,h)$ satisfies ss-property iff each $(s,o,x)\in b$ satisfies ssc rel f.

Proposition (Prop) → قضیه‌ای که اثبات شده و مهم است
Lemma (lem) → قضیه‌ای که برای اثبات قضیه (theorem) بعدی اثبات می‌شود
Theorem (thm) → قضیه اصلی و مهم
Corollary (cor) → قضیه‌ای که از روی قضیه‌ای دیگر اثبات می‌شود

Definition – Suppose $S'\subseteq S$. A state $v=(b,M,f,h)$ satisfies the *-property relative to S' iff

($S'=S\setminus S_T$ مجموعه سوژه‌های غیر مورد اعتماد)

S_T: Trusted subjects

$$\forall s\in S'.\ \forall o\in O.$$

$$o\in b(s:a)\Rightarrow f_o(o)\ \mathcal{D}\ f_c(s)$$

$$\wedge$$

$$o\in b(s:w)\Rightarrow f_o(o)=f_c(s)$$

($\mathcal{D}$: dominate)

$$\wedge$$

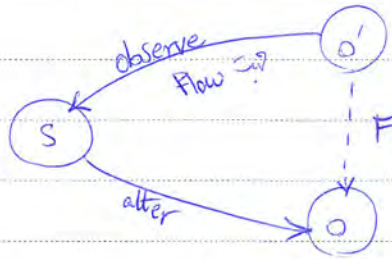
$$o\in b(s:r)\Rightarrow f_c(s)\ \mathcal{D}\ f_o(o).$$

Proposition – If v satisfies *-property rel S' and $s\in S'$, then

$$o\in b(s:a)\wedge o'\in b(s:r)\Rightarrow f_o(o)\ \mathcal{D}\ f_o(o').$$

*-property

Subject
Date



Objects و subjects entities هستند که entity ها می‌توانند properties داشته باشند.

Flow یک subject می‌تواند با observe و alter objects مختلف جریان داشته باشد.

کنترل دسترسی‌ها مختلف است به این معنی که در هنگام بررسی اینکه آیا یک درخواست مجاز است یا خیر، باید به state سیستم نگاه کنیم.

f_c تابع current level یک subject است که به state سیستم بستگی دارد و مشخص می‌کند که آن subject در حال حاضر چه objectهایی را می‌تواند observe کند.

$f_c(s) \geq f_s(s)$

برای اینکه یک subject بتواند یک object را observe کند، باید current level آن subject بزرگتر یا مساوی با current level آن object باشد.

برای هر subject، Max level برابر با Current level است.

در یک untrusted subject، * - property برقرار است، اما SS - property برقرار نیست.

در یک trusted subject، SS - property برقرار است.

Subject Final
Date ۹۲,۱۱,۲۴

Definition - A state $v = (b, M, f, h)$ satisfies the discretionary-Security property (ds-Property) iff

$$\forall s \in S.\ \forall o \in O.\ (s, o, x) \in b \Rightarrow x \in M_{s,o}$$

(Need-To-Know)

↳ ((دسترسی‌ها بر اساس level) برخلاف (دسترسی‌ها بر اساس Subject و object))

با داشتن لازم است که به هر کاربر اجازه دسترسی داده شود.

در discretionary از DAC استفاده می‌شود که بر اساس اینکه چه کاربری چه دسترسی‌هایی دارد، کنترل می‌کند.

برخلاف MAC که بر اساس سطح دسترسی‌ها کنترل می‌کند.

↑ ویژگی خاصیت امنیتی

Definition - A state v is a secure state iff v satisfies SS-property, *-property rel S', and ds-property. A state sequence Z is a secure state sequence iff Z_t is a secure state for each $t \in T$. Call $(x, y, z) \in \Sigma(R, D, W, z_0)$ an appearance (execution) (behavior) of the system. $(x,y,z) \in \Sigma(R,D,W,z_0)$ is a secure appearance iff Σ is a secure state sequence. Finally, $\Sigma(R, D, W, z_0)$ is a secure system iff every appearances of $\Sigma(R, D, W, z_0)$ is a secure appearance.

Subject
Date

در باره تعریف عملی نظارتی را می‌دانیم که

با هم تعاریف خواص ایمنی را دیدیم. با این تعاریف خود، به سادگی تعریف آن را دریافت کردیم یا نه؟

از همین شنیدیم، خوشبختانه هستیم. چون این خواص از جنس Safety هستند و کمی اعمال با هم متفاوت

دقیق است. (البته استدلالی پویا (run-time monitoring

termination: جنس ← Liveness ← می‌تواند چیزی خوب را Prescribe — بعضی از خواص (استثنایی) — انواع Property

توجه کنید

Safety ← می‌تواند چیزی بد را prescribe — بعضی از خواص

این گونه

خواص ایمنی یک prefix دارند و اجرای برای prefix ای که بد است نباید remedy شود. ← سیستم باید همیشه در وضعیت خوب قرار گیرد.

پس است بهتر Safety Property یک BLP، به همین دلیل تعاریف را برای BLP بدهیم.

از خواص هم چیزهایی که اینطور قابل اعمال نیستند، Safety هستند.

طبق این تعریف، اگر تعدادی از اجرای خواص سیستم را نقض می‌کنند، و همان کافی است تا خواص نقض شود.

اجرا، امنیت سیستم را نقض کند. این Property ای است.

اگر همه اجراها (Appearance ها) که سیستم ممکن را دارد سیستم، اگر اجرایی چیزهای بد نباشد این چنین

را تعریف کردیم، به آن نمی‌گوییم Property نیست. حال به سیستم دیگر آن property را داشته باشد،

اگر اجرای سیستم، زیر مجموعه از مجموعه Property ها باشد. البته که هم تعاریف اینطور نیست.

Subject Final
Date ۹۶/۱۱/۲۴

ازطهاشوها

a family of sets of sets

Blp - Secure

hyperproperties (non-property is) non-sense

property

Definition - A rule is a function $\rho : R \times V \longrightarrow D \times V$.

A rule is secure_state-Preserving iff v^* is a secure state whenever $\rho(r, v) = (d_m, v^*)$ and v is a secure state.

request

(ds-property, k-property Preserving SS-property preserving)

Subject

Date ۹۴,۱۱,۲۴

Notation - $\omega = \{\rho_1, \rho_2, \ldots, \rho_s\}$ (omega) is a set of rules. The relation $W(\omega)$ is defined by $(r, d, v^*, v) \in W(\omega)$ iff $d \neq ?$ and

$(d, v^*) = \rho_i(r, v)$ for a unique $1 \leq i \leq s$.

Definition - $(r, d, v^*, v) \in R \times D \times V \times V$ is an action of $\Sigma(R, D, W, z_0)$ iff there is an appearance $(x, y, z) \in \Sigma(R, D, W, z_0)$ and some $t \in T$ such that $(r, d, v^*, v) = (x_t, y_t, z_t, z_{t-1})$.

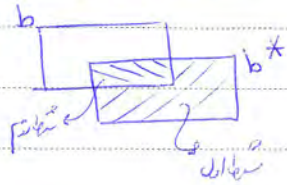
Theorem - $\Sigma(R, D, W, z_0)$ satisfies the SS-property for any initial state z_0 which satisfies SS-property iff W satisfies the following conditions for each action $\left((r, d, (b^*, M^*, f^*, h^*), (b, M, f, h)\right)$:

Subject Final

Date ۹۷/۱۱/۲۴

1) each $(s, o, \underline{x}) \in b^* - b$ satisfies ss-property rel f^*.

2) each $(s, o, \underline{x}) \in b$ which does not satisfy ssc rel f^* is not in b^*.



Computational Induction : z_0 secure است (initial step) appearance اثبات با

(Mana and Preuli)

$(z_{t-1} \to z_t)$ state در کل این است.

Theorem 2 - * - property

Theorem 3 - ds - property

Corollary - $\Sigma(R, D, W, z_0)$ is a secure system iff z_0 is a secure state and W satisfies the conditions of three above theorems.

Subject
Date

Theorem 4 — ... → For ss-property

Theorem 5 — Suppose W is a set of *-property-preserving rules and z_0 is an initial state which satisfies *-property. Then, $\Sigma(R, D, W(w), z_0)$ satisfies *-property.

Theorem 6 — ..... → for ds-property

Corollary 2 — Suppose W is a set of secure-state-preserving rules and z_0 is an initial state which is a secure state. Then, $\Sigma(R, D, W(w), z_0)$ is a secure system.

/

سپس این ها را روی یک rule پیاده می‌کنیم تا ببینیم چه اتفاقی می‌افتد.

A Sample rule:

Rule1 (R1) : get-read

Domain of R_1 : all $r = (g, S_i, O_j, r) \in R^{(1)}$.

(denote domain of R1 by Dom(R1)).

Semantics : Subject S_i requests access to object O_j in read-only mode.

*-property function : $*1(r, x) = TRUE \Leftrightarrow f_c(S_i) \geq f_o(O_j)$.

read-only

Subject Formal
Date ۹۷/۱۱/۲۸

The rule:

$$R_1(r,v) = \begin{cases} (?, v) & \text{if } r \notin dom(R_1) \\ (yes, (b \cup \{s_i, o_j, r\}, M, f, h)) & \text{if } r \in dom(R_1) \wedge r \in M_{s_i,o_j} \wedge f_S(s_i) \geq f_O(o_j) \wedge (s_i \in S_T \vee *1(r,v)) \\ (no, v) & \text{otherwise} \end{cases}$$

ds-property ss-property

dom(R1) ... total function (?, v) ... partial function ... rule ...

... literature ... System Z ... McLean ... Information Flow ... BLP ...

... subject ... object ...

... Tranquility ... PaPCO ...

... BLP ... McLean ...

Subject
Date

- مدل های سیستم Access Control فقط به دارای یا نداشتن دسترسی فرد داده ها propagation اهمیت

بعد از دادن دسترسی های لازم ← سطح سیستم چون دو حق read, write بررسی می شود.

- رویکرد BLP دارای ابهام هایی است که استفاده از آن بعضی از مدل دسترسی ها را در حوزه های general

- تعبیر Extensional سیستم بعضی Intensional بعضی از خواص دارای طرز

Subject Formal
Date ۹۴, ۱۲, ۱

Information Flow Security (Denning ...)

... Info. Flow Security ...

A lattice model of Secure Information Flow 1976

... abstraction ... policy ...

... abstraction ... enforcement ...

... lattice ... language-based Security ...

... lattice ... Security Class ...

An information flow model FM is defined by

$$FM = (N, P, SC, \oplus, \rightarrow)$$

where

- $N = \{a, b, c, \ldots\}$ is a set of logical storage objects or information receptacles
  e.g. file, program variables, ...
- $P = \{p, q, \ldots\}$ is a set of processes.
- $SC = \{A, B, \ldots\}$ is a set of security classes corresponding to disjoint classes of information.

PqPCO

19/

— Need-to-know ، classification ، security clearance ... Security class ...

Subject
Date

- عناصر N و P و عناصری از SC را تشکیل می‌دهند:

$a \in N \quad , \quad \underline{a} \in SC$

$p \in P \quad , \quad \underline{P} \in SC$

- Static binding → کلاس در ابتدای کامپایل تعیین می‌شود
- Dynamic binding → کلاس این object براساس محتوای متغیر تعیین می‌شود

{ Procem در زمان اجرا تعیین می‌شود

- a plus

- $\oplus$: A class combining operator

$\oplus(A, B)$ or $A \oplus B$

$\underline{a} \in A \quad , \quad \underline{b} \in B$ ← عضو $\oplus$ ، کلاسی که $f(a,b)$ متعلق به آن است.

$f(a,b)$

$\underline{a} \oplus \underline{b}$ is the class to which $f(a,b)$ belongs.

PqPCQ

Subject Formal

Date

$$f(a_1 \oplus a_2 \oplus \cdots \oplus a_n) = a_1 \oplus a_2 \oplus \cdots \oplus a_n$$

$\oplus$ is associative and commutative

SC is closed under $\oplus$.

$\longrightarrow$ (right arrow) : A binary relation on SC

$(A, B) \in \longrightarrow$ or $A \longrightarrow B$

اجازه دارد جریان اطلاعات از A به B وجود داشته باشد.

(تعریف انواع flow ها)

$\longrightarrow$ نوعی flow است، نوع flow از نوع

از معایب مدل بل لاپادولا این است که فقط flow

flow تعریف شده وارد و خروجی است به راحتی قابل توسعه نیست. یعنی در واقع امنیت توسعه پذیر نیست.

تعریف رسمی

تعریف policy به شرطی امن است که رابطه $\longrightarrow$ تعریف شود.

A flow model FM is secure iff execution of a sequence of operations cannot give rise to a flow that violates the relation $\longrightarrow$.

FM هیچ‌وقت نمی‌تواند به تعریف امنیت برسد، پس نمی‌تواند امن باشد!

Subject
Date

Flows جریان (یا Flow)

3

Subject Formal — DLM (Barbara Liskov) 2000 — (pierce) 2014-13

Date 96,11,1

SC, $\rightarrow$, $\oplus$ form a universally bounded lattice.

(Common Sentual)

- no generality is lost
- Consistency

```
begin
    b := a;
    c := b;
end
```

Consistency

A universally bounded lattice is a structure consisting of a finite partially ordered set together with least upper and greatest lower bound operators on the set. To show that (SC, $\rightarrow$, $\oplus$) forms such a lattices, we establish that,

1) (SC, $\rightarrow$) is a partially ordered set (poset).
2) SC is finite.
3) SC has a lower bound L such that $L \rightarrow A$ for all $A \in SC$.
4) $\oplus$ is a least upper bound operator on SC. (join)

5 min at first of class!

Subject (Order Theory) → poset, lattice, cpo, Domain Theory, ...
Date (Denotational Semantics)

bounded lattice → ... Bot, Top ... lattice

... Flow ... Top, meet ... Bot, join ...

۱) $A \longrightarrow A$ (reflexivity)

$a := a$ → ... Flow ...

$A \longrightarrow B$ and $B \longrightarrow A \Rightarrow A = B$ ↓

... redundant ... ← practical assumption

$A \longrightarrow B$ and $B \longrightarrow C \Rightarrow A \longrightarrow C$

$b := a$; $b := c$; → $a := c$; ...

... ← ...

... transitive ... transitive ... Flow ...

... Flow ...

۲) $\oplus$ is a least upper bound operator.

$A \longrightarrow A \oplus B$ and $B \longrightarrow A \oplus B$ ...

upper bound

$A \longrightarrow C$ and $B \longrightarrow C \Rightarrow A \oplus B \longrightarrow C$ ...

least upper bound

Subject [illegible]
Date ۹۴/۱۲/۳

(تعریف اولیه)

$c := f(a,b)$

$\downarrow$

$a \oplus b$

[illegible]

(Semantic Absurdity)

[illegible]

(مثال)

$c := a + b$; ⟶ permitted

$c := a$; ⟶ is not permitted

[illegible]

[illegible]

(تعریف ثانویه) [illegible] object [illegible]

a, b, c, c_1, c_2

و قواعد حاکم:

$a \to c$, $b \to c$, $c = c_1 = c_2$

مثال:

$c_1 := a$; $a \to c$

$c_2 := b$; $b \to c$

$c := c_1 * c_2$; $c \oplus c \to c$ ($c = c$)

$\Rightarrow a \oplus b \to c$ (سوری)

[illegible]

[illegible]

Subject
Date

3) There exists $L \in SC$ that $L \to A$ for any $A \in SC$.

Constant

$l := 5$ ... $l := h$ ...

lower bound

meet, glb

$X \subseteq SC$, $X = \{A_1, A_2, \ldots, A_n\}$

$$\bigoplus X = \begin{cases} A_1 \oplus A_2 \oplus \cdots \oplus A_n & \text{if } X \neq \emptyset \\ \bot & \text{o.w} \end{cases}$$

Property: $\forall i \in \{1,2,\ldots,n\} \cdot A_i \to B \iff \bigoplus X \to B$

$\Downarrow$

$lub(X) \to B$

upper bound, least upper bound

PqPCO

Subject Final
Date ۹۶,۱۲,۴

- Information in objects $A_1, \dots, A_n$ can flow separately into an object b

iff $a_1 \oplus a_2 \oplus \dots \oplus a_n \rightarrow b$.

The greatest lower bound operator : $\otimes$ ~> 0 times

$$A \otimes B = \oplus L(A,B) = \oplus \{c \mid c \rightarrow A \wedge c \rightarrow B\}$$

derived form ⊕ است از ⊗

$\otimes$ ... lower bound ... glb

$A \otimes B \rightarrow A$ and $A \otimes B \rightarrow B$

$D \rightarrow A \wedge D \rightarrow B \Rightarrow D \rightarrow A \otimes B$

$c_1 \rightarrow B$, $c_2 \rightarrow B$, ..., $c_n \rightarrow B$ ⇓ $c_1 \oplus c_2 \oplus \dots \oplus c_n \rightarrow B$ ⇒ $A \otimes B$

$c_1 \rightarrow A$, $c_2 \rightarrow A$, $c_3 \rightarrow A$, ... ⇓ $c_1 \oplus c_2 \oplus \dots \oplus c_n \rightarrow A$ ⇒ $A \otimes B$

Subject Formal
Date ۹۴, ۱۲, ۲

$X = \{B_1, B_2, \dots, B_n\} \subseteq SC$ ، فرض کنیم

$$\bigotimes X = \begin{cases} B_1 \otimes B_2 \otimes \cdots \otimes B_n & X \neq \emptyset \\ H & X = \emptyset \end{cases}$$

$H \rightarrow$ top $\rightarrow$ set of upper bounds

یعنی بزرگترین object که بالاتر از آن نیست

$H = \bigoplus SC = \bigotimes \emptyset$

$\hookrightarrow$ عضو SC که بالاتر از همه است

$\forall i \in \{1, \dots, n\} \cdot A \rightarrow B_i \iff A \rightarrow B_1 \otimes B_2 \otimes \cdots \otimes B_n$

$(A \rightarrow \bigotimes X)$

یعنی اگر از A به هر کدام از B_i ها جریان داشته باشد، از A به ترکیب همه آنها هم جریان دارد.

Information in an object a can flow into objects $b_1, \dots, b_n$ iff

$a \rightarrow b_1 \otimes b_2 \otimes \cdots \otimes b_n$

Corollary – $(SC, \rightarrow, \oplus, \otimes, L, H)$ is a bounded lattice.

این نتیجه می‌دهد که هر مجموعه‌ای از کلاس‌های امنیتی یک کران دارد.

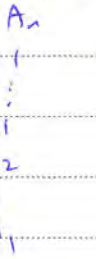
(Example)

Instantiation –

Top secret hierarchical :

SC = {unclassified, confidential, secret, top secret}

Hasse Diagram



top secret
secret
confidential
unclassified

Subject Formal
Date ۹۶/۱۲/۳

$SC = \{A_1, \ldots, A_n\}$:نمی

$A_i \oplus A_j = A_{\max\{i,j\}}$

$A_i \otimes A_j = A_{\min\{i,j\}}$ (hierarchical)

(linear)

$L = A_1$

$H = A_n$

Example -

non-linear ordering

$SC = \mathcal{P}(X)$, $\rightarrow$ = $\subseteq$

power set — arrow — subset

$A \in SC, B \in SC, A \rightarrow B$ iff $A \subseteq B$.

عناصر دارای ویژگی‌های مشترک و ویژگی‌های متمایز

$A \oplus B = A \cup B$

$A \otimes B = A \cap B$

$L = \emptyset$

$H = SC$

مثل) $X = \{med, fin, crim\}$, $SC = \mathcal{P}(X) = \{\emptyset, \{med\}, \{fin\}, \{crim\}, \ldots, \{med, fin, crim\}\}$

$a = \{med, fin\}$

object

$b := a \longrightarrow b \supseteq \{med, fin\}$

superset

Subject
Date

Example -

پس با آمدن یک نقطه خطی دو سیستم، دیگر غیرخطی، مثلاً نقطه سوم ترکیبی از این دو به شکل:

$(SC_1, \rightarrow_1)$, $(SC_2, \rightarrow_2)$ are bounded lattice, so

$(SC_1 \times SC_2, \rightarrow)$ is a bounded lattice.

$$(A, B) \rightarrow (A', B') \Leftrightarrow A \rightarrow_1 A' \text{ and } B \rightarrow_2 B'.$$

(مثل label ها در BLP)

Abstraction به اندازه کافی فراهم است و دست ما باز است. از طرف دیگر کارهای ملموس با آن‌ها تامین پذیر است.

نتیجه‌گیری در سیستم‌ها:

1) $a_1, \dots, a_n$ can flow into b

$$\Leftrightarrow$$

$$a_1 \oplus a_2 \oplus \dots \oplus a_n \rightarrow b.$$

lub تا flow شدن مجاز

2) a can flow into $b_1, b_2, \dots, b_n$

$$\Leftrightarrow$$

$$a \rightarrow b_1 \otimes b_2 \otimes \dots \otimes b_n$$

↓

glb تا

تا اینجا صحبت کردیم denning، اگر یک برنامه به ما بدهند، نمی‌توانیم flow بین متغیرها را مشخص کنیم چون اساساً تعریفی برای flow ارائه نشده.

Subject Formal
Date ۹۴/۱۲/۸

(Enforcement)

if a=0 then b:=0

در این دستور اگر a برابر با صفر باشد مقدار b برابر با صفر می‌شود پس از اجرای دستور

flow وجود دارد. چون اگر b≠0 باشد a≠0 بوده است.

observer can learn nothing ← هیچ اطلاعی از این دستور

flow نیست چون اجرای این دستور باعث نمی‌شود اگر a و b آنی نباشند

flow انواع

۱) Explicit ← statement (assignment) که در آن object ها مستقیم از هم

eg. $b = f(a_1, \ldots, a_n)$ اطلاعات a_1 تا a_n به b منتقل می‌شود.

assignments, I/O در این تعریف Explicit Flow شامل

۲) Implicit ←

Implicit flow to 'b' occurs as the result of executing, or not executing a statement that causes an explicit flow to 'b' when the statement is conditioned on the value of an expression.

هر جا که شامل یک جریان صریح است، مشروط به مقداری خاص است.

PAPCO

Books:

Subject: Concepts of PL

Date

تعریف استقرایی: دو جنس از rules داریم، Axiom و Proper Rule. Axiom ها مقدمه ندارند و Proper Rule ها مقدمه دارند.

if a=0 then b:=c ;

a=0 ← explicit flow ← b:=c

implicit flow ← a ← b

برای اثبات چیزی در مورد برنامه‌ها، از استقرا روی ساختار abstract برنامه‌ها استفاده می‌کنیم.

S (abstract program or statement):

1) S is an elementary statement. (assignment, I/O, ...)
(Every elementary statement is a program)

2) There exists S_1 and S_2 such that $S = S_1 ; S_2$.

3) There exists $S_1, \dots, S_m$ and an m-valued variable c such that $S = c : S_1, \dots, S_m$

switch-case ← S_i: اجرا شود اگر c برابر i باشد

در Sequencing برخلاف I/O و assignment جریان اطلاعات وجود دارد.

implicit flow ← Conditional

abstract ← این‌طور

if c then S_1 $\Rightarrow$ $c : S_1$

while c then S_1

if c then S_1 else S_2 $\Rightarrow$ $c : S_1, S_2$

Subject Final

Date ۹۷/۱۲/۱۸

برای بررسی امنیت یک برنامه (Program) می‌توان از روش‌های مختلفی استفاده کرد. یکی از این روش‌ها استفاده از Syntax-directed برای بررسی امنیت است:

1) An elementary statement S is secure if any explicit flow caused by S is secure.

e.g.: $S \triangleq b := f(a_1, \dots, a_n)$

$a_1 \oplus a_2 \oplus \dots \oplus a_n \rightarrow b$ should hold after the execution of S.

Dynamic Binding و Static Binding در این روش بسته به نوع زبان متفاوت است.

2) $S = S_1 ; S_2$ is secure if both S_1 and S_2 are individually secure.

transitive بودن flow

3) $S = c : S_1, \dots, S_m$ is secure if each S_k is secure and all implicit flows from c are secure.

implicit flow: اگر یک object مانند b_i در یکی از S_i ها explicit flow به یک object مانند b_n داشته باشد آنگاه:

$c \rightarrow b_1 \otimes b_2 \otimes \dots \otimes b_n$ Should hold after the execution of S.

در dynamic binding $b_i := b_i \oplus c$ یعنی هر b_i که وابسته به c است

Subject
Date

این تقسیم از این است، Intensional است (مثل BLP) یعنی دارد عنصر خروجی دارد.

در این کتاب ما، فرض بر این است، attacker, observer, Source code را دارد، فعلاً معماری (Obscurity) را نمی‌گیرد.

ما اجرا را کلی این است که سه مدل دارد.

Enforcement (Mechanisms):

- Static binding
  - Compile-time Mechanism (static Analysis) (static)
  - Hybrid
  - Run-time Mechanism (Runtime Analysis)
- Dynamic binding
  - Compile-time Mechanism (static)
  - Hybrid
  - Run-time Mechanism

* Dynamic ← Run-time یعنی زمانی که برنامه دارد اجرا می‌شود، برای این است که dynamic یعنی برنامه را برای این است امنیت اجرا می‌کنیم.

Compile-time : static ← برنامه قبل از اجرا در فایل می‌شود. این دارد اجرا می‌شود، اجرای آن

Run-time ← برنامه اجرای برنامه، فرض جزء این است که اینش نمی‌شود. قبل از برنامه اجرا.

این که Policy چیزی را نقض از این ها اعمال می‌شود، سوال این است.

Hybrid ← از این است Compile-time استفاده می‌کند، Run-time این است بیشتر کار را.

روش دیگر این است که بیایی قسمتی از برنامه را Parallel اجرا می‌کند و اعمال انجام می‌دهد.

این را در Run-time نمی‌دانیم، پس dynamic نمی‌گوییم.

PgPCO

Subject Formal
Date ۹۶/۱۲/۸

Program Rewriting : Run-time و Static

Static Analysis (Compile-time) :

- Program Analysis (Data-Flow Analysis)
- Type Checking (Type System)
- Abstract Interpretation
- Model checking
- ...

Run-time Mechanism :

- Run-time Monitoring

Hybrid :

Static Binding :

Run-time Mechanisms :

1- Access Control Mechanism : → BLP :

P → a Procss P → Procss clearance

↳ The lowest class p can write into
The highest class p can read from

Subject

Date ۹۴/۱۲/۱۰

read from 'a' : حروف مجاز

verify $\underline{a} \rightarrow \underline{P}$: این را چک می‌کنیم

write into 'b' : حروف مجاز

verify $\underline{P} \rightarrow \underline{b}$

P can read from $a_1, \ldots, a_n$ and write into $b_1, \ldots, b_m$ iff

$$\underline{a_1} \oplus \underline{a_2} \oplus \cdots \oplus \underline{a_n} \rightarrow \underline{P} \rightarrow \underline{b_1} \otimes \underline{b_2} \otimes \cdots \otimes \underline{b_n}$$

تضمین می‌دهد جریان‌های flow غیر مجاز نداشته باشیم.

flow بین object و subject ، process و ... بین چه چیزهایی را تضمین می‌کند.

2 — The Data Mark Machine :

این روش توسط Fenton مطرح شده

— A security class $\underline{P}$ is associated with the program counter P.

$C : S_1, \ldots, S_m$ ← conditional structure

در ابتدای برنامه P را در stack قرار می‌دهیم و این کار را push می‌نامیم :

(push $\underline{P}$)

$\underline{P} := \underline{P} \oplus \underline{C}$

PAPCO

* در انتهای برنامه pop می‌کنیم . (pop $\underline{P}$)

Subject Final
Date ۹۶/۱۲/۱۰

در مورد هر statement به صورت جداگانه بررسی می‌کند که از ورودی‌های کدام سطح و به کدام سطح flow دارد. explicit flow را به این صورت تعریف می‌کند:

$$p = c_1 \oplus c_2 \oplus \cdots \oplus c_k$$

If S specifies an explicit flow from object $a_1, \ldots, a_n$ to an object b (in some S_i), the machine verifies $a_1 \oplus a_2 \oplus \cdots \oplus a_n \oplus p \longrightarrow b$.

که این هم جزئی است که دارد این‌ها را چک می‌کند.

چون Run-time است پس فقط وقتی دارد چیزها را چک می‌کند که کد اجرا شود، یعنی اگر شرطی باشد و اجرا نشود دارد.

Fenton ابتدا این را مطرح کرد که در مورد static binding و flow چه‌جوری این‌ها را چک کنیم. (Exercise!)

implicit flow را هم در نظر گرفته است، چون عدم وقوع آن‌ها را هم باید کنترل کند.

Covert Channels: حالت‌هایی که ما از بررسی کردن دستورات فقط می‌گذریم و اجرا نمی‌کنیم. همین هم خودش مشکل ایجاد می‌کند.

یعنی کدهایی که abort نشدند، اگر abort شدند، نشدند.

Compiler-time Mechanisms:

- این مکانیزم‌ها این مشکلات را ندارند: Covert Channel ندارند (leakage of termination).
- مزایای static: Run-time overhead ندارند ← سیستم‌ها با اجرای کمتر اما با استاتیک بیشتر چک می‌کنند.

Subject
Date

مشکل این است که روی سورس در سطح بالا اجرا می‌شود، زمانی که به اجرا ماشین می‌رسد

مثلاً: Flow هایی که در سورس در سطح بالا چک می‌شوند و ممکن است در حالت ماشینی دیگر Flow نباشند

هر چیزی که در سطح ماشین اجرا شود ← زبان معرفی نیست واقعاً!

Halting Problem: ← Overly Conservative ← زبان‌های واردات!

تعریف خاصیت‌ها و اثبات آن‌ها

مثلاً branch های ممکن در حالت‌های مختلف را بدهد

که تنها راهی است که واقعاً اجرا می‌شود.

در این فضا، میزان محافظه‌کاری را می‌خواهیم کم کنیم ← اینکه چطور این کار را انجام دهیم بحث دیگری است

اینکه چه چیزهایی را رد کنیم و چه چیزهایی را نه.

Permissive ، Restrictiveness ، Completeness ، Soundness : چهار تا term

↓ ↓ ↓ ↓

معیارهای Runtime است — معیارهای Static است — کم کردن دقت تحلیل در این حالت — این معیارها تضمین نمی‌دهند

من که در واقعی را دارد. — نقض کند درباره insecure هم صحبت نمی‌کند. — است است. ← از این چیزها دور باشید!

Schnider ، Hamlen ثابت کردند مسئله‌ای decidable را می‌توان به روشی اجرا کرد.

به صورت Co-recursively ، را می‌توان به monitor اجرا کرد.

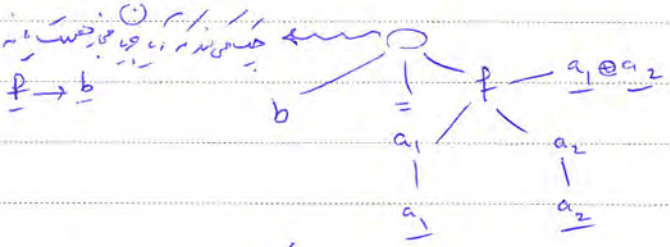
حال به سراغ زبان‌هایی می‌رویم که در آن‌ها (به امروزه در آن‌ها) type System وجود دارد:

Subject Formal
Date ۹۶/۱۲/۱۵

- Certification Mechanism :

S specifies an explicit flow from $f(a_1, \cdots, a_n)$ to b.

- $f(a_1, \cdots, a_n) = a_1 \oplus a_2 \oplus \cdots \oplus a_n$



$S \triangleq b = f(a_1, \cdots, a_n) \rightsquigarrow S = b$

- $S \triangleq S_1 ; S_2$

$\rightarrow S = S_1 \otimes S_2$

- $S \triangleq c : S_1, \cdots, S_m$

$\rightarrow S = S_1 \otimes S_2 \otimes \cdots \otimes S_m$ and verify $c \rightarrow S$

Subject
Date

Example - if c then begin a:=0 ; b:=1 end

$\underline{a}$ $\underline{b}$

$\underline{a} \oplus \underline{b}$

$\underline{a} \oplus \underline{b}$

now, verify that $\underline{c} \rightarrow \underline{a} \oplus \underline{b}$ is allowed or not.

Dynamic binding:

Subject Final
Date ۹۵/۴/۱۰

whenever a statement S specifying a flow from object $a_1, \dots, a_n$ to a dynamically bound object b is executed, the class of b is changed: $\underline{b} := \underline{a_1} \oplus \underline{a_2} \oplus \dots \oplus \underline{a_n} \oplus \underline{P}$

↓

بررسی امنیتی guard های تعریف شده

از) purview) برای جلوگیری از نشت اطلاعات

که وقتی شرط یک حلقه while، ⊕ می‌شود، حرفا نمی‌توان گفت که آن حلقه چی برمی‌گرداند و باید

بیشتر بررسی شود.

مثال حل شده از Fenton:

```
b := c := false;
if ¬a then c := true;
if ¬c then b := true;
```

⇒ `b := c := false;` ⟶ $\underline{b} := \underline{c} := \underline{L}$ (lattice of lower bound)

Fenton الگوریتم

$\underline{P} := \underline{a}$; if ¬a then {c := true; $\underline{c} := \underline{L} \oplus \underline{P} \oplus \underline{a}$; $\underline{P} := \underline{L}$;

↓

static binding از

$\underline{P} := \underline{c}$; if ¬c then {b := true; $\underline{b} := \underline{L} \oplus \underline{P} \oplus \underline{c}$; $\underline{P} := \underline{L}$;

حالا از برنامه بالا که بدون ایمنی اجرا می‌شود:

Subject
Date

if a is (true) → (c, c̲) = (false, L)

(b, b̲) = (true, L)

if a is (false) → (c, c̲) = (true, a)

(b, b̲) = (false, L)

در حوزه‌ی جاوا، b خیلی a است، پس b هم خیلی L است!

(b = L) (b = a)

یعنی از a خیلی L است که خیلی محرمانه است، پس باید b هم این‌طور باشد!

چون branch ای برای عبور از یک سیستم است ← باید Fenton متن بود، پس این flow غیرمجاز است!

از این به بعد تعریف‌ها، همه‌ی حرف‌ها را در همه حوزه‌ها، به‌روز حوزه‌ها

Intensional بود، بحث می‌شود. ← در این فصل، بحث denning

Covert channel

سطوح هم در BLP هم در denning وجود دارند.

Access Control Approach ← (Information Flow Track) که یکی از آن‌ها خواهیم دید.

در denning هم راه‌های مختلفی وجود دارد، تعریف flow این است. اما بحث بعدی Logical در سیستم و سطح است.

Subject Formal
Date ۹۴/۱۲/۱۵

Non-Interference:

در Security Policies and Security Models, Grogeun and Meseguer

- تعریف شد Information Security Policy را به صورت زیر:

- اگر دامنه یک حالت از حالت‌های سیستم، مجموعه‌ای از کاربران و مجموعه‌ای از دستورات باشد

$u \nrightarrow v$

آنگاه کاربر u بر کاربر v تداخل ندارد.

حداقل یکی از این Process ها به اجرای

non interference اصطلاح ← (نداشتن تداخل است)

Has no effect

Definition – A System M consists of

The sets : U, S, SC, Out, Capt, CC

and Functions :

$out : S \times Capt \times U \rightarrow Out$

$do : S \times Capt \times U \times SC \rightarrow S$

$cdo : Capt \times U \times CC \rightarrow Capt$

and Constants t_0 and S_0 as the initial Capability table and the initial machine state.

U : Users S : states SC : State Commands Out : Outputs

Capt : Capability Tables CC : Capability Commands

Subject
Date

Notation - $Ab = \wp(C)$ $\quad C = SC \cup CC$

ability — Powerset — all commands

$Capt : Ab \xrightarrow{U} Ab$

$States : S \times Capt$

$csdo : S \times Capt \times U \times C \longrightarrow S \times Capt$

$csdo^* : S \times Capt \times (U \times C)^* \longrightarrow S \times Capt$ ⟿ multi-step

sequences of pairs

$csdo(S, t, NIL) = (S, t)$

$csdo^*(S, t, w.(u, c)) = csdo(csdo^*(S, t, w), u, c)$

concat

$w \in (U \times C)^*$

Subject Formal
Date ۹۴, ۱۲, ۱۵

double bracket

Notation - $[[w]] = csdo(s_0, t_0, w)$

↓

state

Notation - $[[w]]_u = out([[w]], u)$ → output on state u

Definition - Let $G \subseteq U$, $A \subseteq C$, and $w \in (U \times C)^*$. Then,

→ Purge

$P_G(w)$ denotes the subsequence of w obtained by eliminating those pairs (u,c) with $u \in G$. Similarly, $P_A(w)$ denotes the subsequence obtained by eliminating those pairs (u,c) with $c \in A$.

Moreover, $P_{G,A}(w)$ denotes the subsequence obtained by eliminating those pairs (u,c) with $u \in G$ and $c \in A$.

Definition - Given a system M and $G, G' \subseteq U$. We say that G does not interfer with (or is noninterfering with) G', written $G \not| G'$, iff $\forall w \in (U \times C)^*. \forall u \in G'. [[w]]_u = [[P_G(w)]]_u$.

Subject
Date

Similarly, $A : |G'$ iff $\forall w \in (U \times C)^*. \forall u \in G'. [[w]]_u = [[P_A(w)]]_u$.

And $G, A : |G'$ iff $\forall w \in (U \times C)^*. \forall u \in G'. [[w]]_u = [[P_{G,A}(w)]]_u$.

Subject Formal
Date ۹۴/۱۲/۱۵

Security ... Multi-level ...

Example - (MLS)

L : A linearly ordered set of security level.
($\leq$) (total ordered)

level : $U \to L$

$U[-\infty, x] = \{u \in U \mid level(u) \leq x\}$ ←

$U[x, +\infty] = \{u \in U \mid level(u) \geq x\}$

A system M is multi-level secure iff $\forall x' < x \cdot U[x, +\infty] :| U[-\infty, x']$.

Example - linearly ordered , partially ordered:

$\forall x \in L \cdot U[x, +\infty] :| \overline{U[x, +\infty]}$

- ACM Transactions ، Myers, Liskov ، (DLM) decentralized label model

ACM TISSEC

I-Formation Flow

Subject ↑ Cohen (?), 1977

Date (CSF, S&P)

... observation ... high ... low ... (high = private, low = public)

Example — A channel is defined to be a set of commands.

Every channel is an ability $A \subseteq C$.

Policy: G and G' can communicate only through channel A.

$$G, \bar{A} :| G' \wedge G', \bar{A} :| G$$

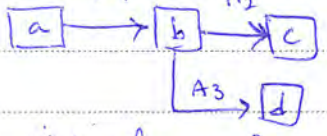
↓

...

Example —

c, b, c, d → process

A_1, A_2, A_3 → channel



... Formation flow diagram ... Policy ... Flow ...

$\{b, c, d\} :| \{a\}$

$\{c, d\} :| \{b\}$

$\{c\} :| \{d\}$

$\{d\} :| \{c\}$

$\{a\}, \bar{A_1} :| \{b, c, d\}$

$\{b\}, \bar{A_2} :| \{c\}$

$\{b\}, \bar{A_3} :| \{d\}$

... (assured pipeline) ... flow c ... b ... flow ... a ... c ... flow ... b ... flow c ... a ... flow c ... b ... flow ...

Subject: Formal
Date 9E/12/17

رابطه ⊲: (noninterfering) intransitive یا transitive

ذات رابطه noninterference، ویژگی‌های مهمی transitive بودن دارد.

همین باعث می‌شود که نمی‌توان با دو a ← c و c ← b، حتماً از a به b نفوذ کرد.

این مشکل سال‌ها بدون راه‌حل بود تا ۲۰۰۸ که حل شد. ایده آن است که actionهای a دارای

(height) و b، follow می‌شوند را حذف می‌کنیم (purge). پس از این تغییر b را نمی‌بیند. اما قضیه‌ای unwinding

را اثبات کرد. خواص Information Flow را بررسی می‌کنیم: خواص trace‌ها. سال Rushby یک proof

آن‌ها چیست و unwinding Theorem را ثابت کرد.

در بررسی، بدون دسترسی، می‌بینید این policyها را بررسی می‌کنیم static policy است.

dynamic policy در دسته‌بندی. یعنی G:G' یعنی گراف G به گراف G' می‌رود. اما نمی‌توان

گفت Capability‌ها می‌توانند برای هر policy ما باشد. Sanitize شده است و به گراف نمی‌رود.

Dynamic Policies

Definition - Let $G, G' \subseteq U$, $A \subseteq C$, and P be a predicate defined over $(U \times C)^*$.

یک محمول برای بررسی گراف، true یا false برمی‌گرداند.

Subject:
Date

Then, G is noninterfering with G' under condition P, written

"$G, A : | G'$ if P", iff

$$\forall w \in (U \times C)^{*}.\ \forall u' \in G'.\ [[w]]_{u'} = [[p(w)]]_{u'}$$

$p(w)$: حذف کردن purge

where p is defined by

$$p(\varepsilon) = \varepsilon \quad (\varepsilon \text{ is empty string})$$

$$p(o_1, o_2, \ldots o_n) = o'_1 \cdots o'_n \quad \text{where}$$

$o'_i = \lambda$ if $P(o'_1 \cdots o'_{i-1})$ and $o_i = (u, a)$ with $u \in G$ and $a \in A$.

and $o'_i = o_i$ otherwise.

شرط‌دار: حذف برای o_i برقرار نیست

این تعریف می‌گوید: unconditional Purge حذف می‌شود، اما این شرط حذف می‌شود که برای (u,a) با $u \in G, a \in A$ برقرار نباشد، آن گاه حذف می‌کنیم که شرط برقرار نباشد.

آن action نباشد، حذف نمی‌شود.

Subject: Formal
Date ۹۶/۱۲/۱۷

Example –

false true

$CHECK : (U \times C)^* \times U \times C \rightarrow \{0,1\}$

$CHECK(\omega, u, c)$

$[[\omega]]$ → حق دسترسی u در state حاصل از اجرای ω

$CHECK(u,c) : (U \times C)^* \rightarrow \{0,1\}$

تابعی از هر رشته‌ای که true یا false برگرداند

$\{u\}, \{c\}$: u ... $\neg(CHECK(u,c))$ for all $u \in U$ and $c \in C$.

حرکت بردی: حرکت سری با حرکت بردی از حق دسترسی، عدم تداوم است مشروط بر اینکه u حق نداشته از p

استفاده کند. چون آن امکان در دو حالت Capability ها تغییر کند. (dynamic)

می‌توان با این Policy حرکتی انجام داد:

A command pass(u,c)

$\omega \in (U \times C)^*$

$\omega = \omega' \cdot o$

$o \in (U \times C)$

Subject: Formal
Date ۹۵۱۲۱۷

Previous(w) = w' last(@w) = 0

CHECK(Previous, (u,c))(w) ≜ CHECK(Previous(w), u, c)

{u, c} ∈ U if ¬CHECK(Previous, u, c) ∧
(CHECK(previous, u', pass(u,c)) →
¬last = (u', pass(u,c)))

در این جمله اگر پس‌گذر نباشد pass(u,c) نمی‌تواند باشد.

که می‌توان شکل بعدی آن را نیز داشت. در مقاله این سه شرط می‌توان

(قرین ۱، ۲، ۳، شکل DLM، شکل HRU ← که می‌توانند باشند ۱۴، ۱، ۹۵)

۹۵/۱/۱۴

که از آن برای اثبات Noninterference استفاده می‌شود. این ابزار GMNI

یک ابزار اثبات خودکار است.

ایرادات GMNI:

- فقط برای سیستم‌های deterministic کاربردی دارد.
- overly restrictive است. (سخت‌گیری بیش از حد)
- از روش‌های statistical استفاده نمی‌کند.
- Composable است.
- Refinement-closed است.
- برای intransitive policy کاربرد دارد.

P4PCO

Subject: Formal
Date
۹۵/۱/۱۴

Nondeducability:

(۱۹۸۶ سال)

(Sutherland مقاله)

این اولین مقاله ای بود که در مورد GNNI ارائه شد.

با این وجود آن مقاله دارای ایرادات زیادی بود.

GNNI مخفف Ideal Cryptography و Secure است.

Nondeducability یعنی low از high نباید چیزی در مورد deduction ها بداند.

(relax نوعی از non-deterministic است که در مورد GNNI)

برای سیستم های Nondeterministic نمی‌توان ماشین حالت مشخصی را در این مورد به کار برد.

از general state machine استفاده می‌شود.

* تابع Information Function: تابعی داخلی است که اطلاعاتی از سیستم دارد.

trace: a sequence of states and events

Information Function : $T \rightarrow V$

(value)

trace ← دنباله‌ای از اجرای سیستم است.

سیستم در هر لحظه می‌تواند اجرای مختلفی داشته باشد.

شده user ها نمی‌توانند با filter کردن اطلاعاتی که دارند، متوجه تفاوت را بفهمند.

(Possible Worlds اصل)

T

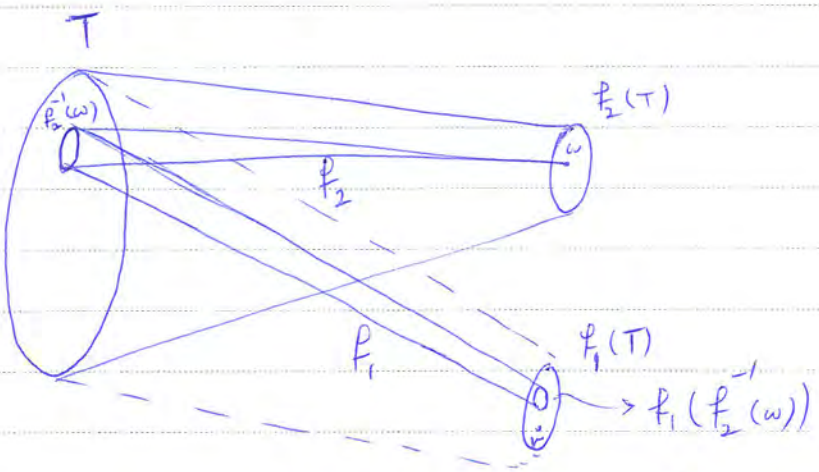
Subject:
Date

Definition – Let T be the set of traces of the system. Let f_1 and f_2 be two functions defined on T, and let $\omega \in Image(f_2)$; i.e., there exists a trace $t \in T$ such that $f_2(t) = \omega$. We say that information flows from f_1 to f_2 on ω if there is a value v of f_1 such that for all traces $t \in T$, $f_1(t) = v \Rightarrow f_2(t) \neq \omega$.

(تعریف: ω در برد تابع f_2 باشد، یعنی $f_2(t) = \omega$ حداقل برای یک t. $f_1(t)$ برابر v نباشد.)

توابع f_1 و f_2 برای همه‌ی ورودی‌ها مشخص هستند، خروجی تابع نامشخص است.

دارنده‌ی f_2 چیزی از مشاهده‌ی دارنده‌ی f_1 deduce یا exclude می‌کند.



زیرا این flow نشان می‌دهد که $f_1(f_2^{-1}(\omega)) = f_1(T)$ یعنی دانستن این که ω در برد f_2 است، هیچ اطلاعاتی درباره‌ی f_1 به ما نمی‌دهد.

Subject: Formal
Date
۹۵/۱/۱۴

Information flows from f_1 to f_2 iff there is some $w \in Image(f_2)$ such that information flows from f_1 to f_2 on w.

[illegible] verify [illegible] trace [illegible] f_1, f_2 [illegible] trace [illegible]

[illegible] pre image [illegible] f_2 [illegible]

[illegible] (moto) [illegible] f_1 [illegible] f_2 [illegible] non-deducability-secure [illegible]

[illegible] exclude [illegible]

Theorem – Given T, f_1 and f_2, information does not flow from f_1 to f_2 iff the joint function (f_1, f_2) from T to the product of image sets of f_1 and f_2 is onto.

$$(f_1, f_2) : T \longrightarrow Image(f_1) \times Image(f_2)$$

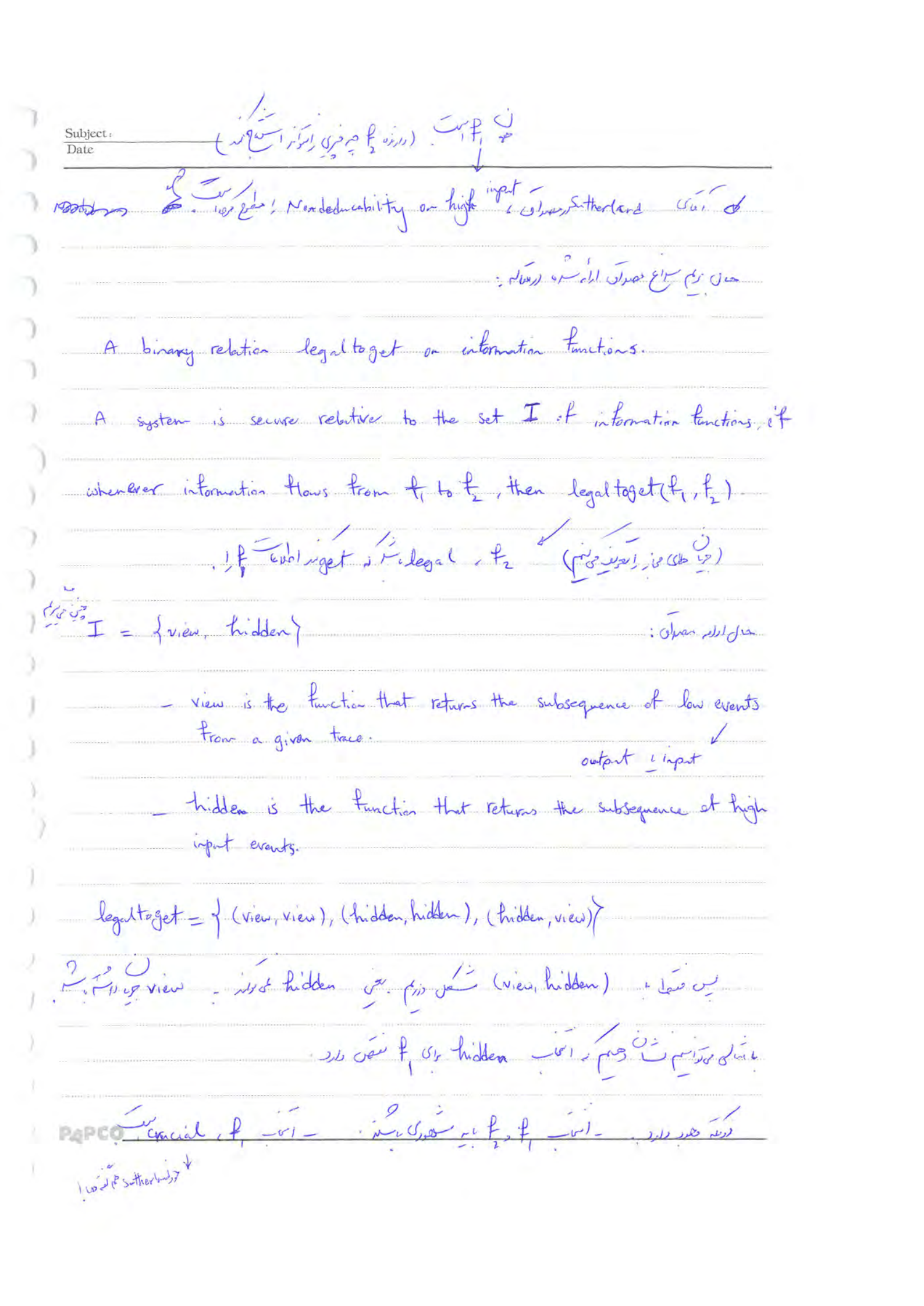
$$(f_1, f_2)(t) = \left(f_1(t), f_2(t)\right)$$

[illegible] f_1 [illegible] f_2 [illegible] f_2 [illegible] f_1 [illegible]

[illegible] abstraction [illegible]

[illegible] instantiation [illegible] Sutherland [illegible]

Subject:
Date



Subject: Formal

Date 92/1/14

Example – A nondeterministic system S, having four states corresponding to possible values of a pair of keys K_1 and K_2 ($K_i \in \{0,1\}$). The (high) transmitter has three possible inputs $\{q, 0_T, 1_T\}$. The (low) receiver has only one input $\{r\}$. Use of the system consists of a sequence of trials. On each trial:

1. Both players submit an input
2. The machine calculates and delivers output for each player. These outputs are function of the current states and the current inputs.
3. The machine makes a nondeterministic state transition based on the current state and the current inputs.

If the current state is (K_1, K_2) and the current transmitter input is $x \in \{q, 0_T, 1_T\}$.

i. when $x \in \{0_T, 1_T\}$

a) the transmitter gets a 0 for output.

b) ~~the receiver gets $x \oplus K_1$ for output.~~

c) both K_1 and K_2 are randomly and independtly updated.

Subject: Information flow in nondeterministic systems, J. Todd Wittbold

Date

A model of information, Sutherland, 9th National Computing, 1986,

2. when $x = q$

a) the transmitter gets K_1 for output.

b) the receiver gets K_2 for output.

c) K_1 remains unchanged, K_2 gets updated randomly.

(initial state is random)

مثال)

$0 \le q \le 1$ $\quad$ q $\quad$ $initial = (0,0)$

transmitter در trial های q ، q فرستاده و در trial های $1-q$ ، 1 فرستاده می‌شود.

در trial های q ، K_1 برابر 0 است و در trial های $1-q$ ، K_1 برابر 1 است. پس leakage داریم. پس سیستم امن نیست.

ولی ثابت می‌شود این سیستم Nondeducability secure است.

برای اثبات این موضوع فرض می‌کنیم X فرستاده شده، $X \oplus K_1$ از روی کانال خوانده می‌شود و receiver $(X \oplus K_1) \oplus K_1 = X$ را می‌خواند.

پس این چنین می‌شود که trial های $1-q$ را نادیده می‌گیریم و trial های q را در نظر می‌گیریم و می‌گوییم که اطلاعات Nondeducibility on high inputs است. پس این سیستم nondeducability secure است.

این سیستم nondeducability secure است. پس این سیستم P_1 می‌باشد.

در این مثال هم این طرف می‌خواهد "استراتژی‌ای" برای transmitter بفرستد که از آن اطلاعی به receiver نرسد.

nondeducability on high strategy

پس صرف high input بودن خروجی، سیستم را ناامن نمی‌کند.

PqPCO

Subject: Formal
Date 90/1/14

نیز به صورت مثال داریم: nondeducability on high strategy است

این تعریف را P_1

The Strongest ... The Weakest

Fully Restrictive ... Fully Permissive

... strategies hidden

پس ما ND on hidden ، آن را weak ، یعنی بعضی از ناامنی‌ها را امن می‌داند.

تعریف مخصوص ما!

نیز ND در framework این تعریف‌ها قرار دارد.

برای آن State Machine را تعریف می‌کنیم: (برای on high strategy)

Synchronized state machine

- S : set of states with $S_0 \subseteq S$ as the set of initial state.
- I_R : Set of inputs of receiver R.
- I_T : Set of inputs of transmitter T.
- O_R : Set of outputs to receiver R.
- O_T : Set of outputs to transmitter T.

Subject:
Date

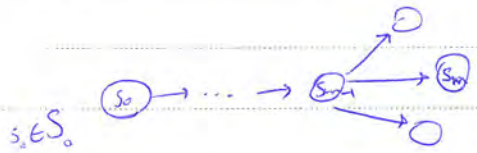
nondeterminism

$- Next : S \times I_R \times I_T \rightarrow \mathcal{P}(S) - \emptyset$

Next state function

$- Out_R : S \times I_R \times I_T \rightarrow O_R$

$- Out_T : S \times I_R \times I_T \rightarrow O_T$



move $\Rightarrow$ $s_{m-1} \; (i_m \; j_m \; k_m \; l_m) \; s_m$

این حرکت از state زیری کوچک.

شرط‌های حرکت:

$- s_m \in Next(s_{m-1}, i_m, j_m)$

$- Out_R(s_{m-1}, i_m, j_m) = k_m$

$- Out_T(s_{m-1}, i_m, j_m) = l_m$

- A trace (or execution) of the state machine is a finite sequence of moves.

$s_0 \; (i_1, j_1, k_1, l_1) \; s_1 \; (i_2, j_2, k_2, l_2) \; s_2 \cdots s_{n-1} \; (i_n, j_n, k_n, l_n) \; s_n$

n حرکت از s_1 تا s_n trace یک

Definition - A strategy of length n for a user process U (either T or R) is a sequence $\pi = (\pi^1, \ldots, \pi^n)$ of n functions, where for each i, $1 \le i \le n$

$\pi^i = (I_U \times O_U)^{i-1} \rightarrow I_U$ ⇐ تابع history ها از ابتدا تا یک مرحله قبل به ورودی بعدی

یک استراتژی از ورودی‌های قبلی و خروجی‌های قبلی به ورودی بعدی تصمیم می‌گیرد.

Subject: Final
Date 95/1/14

extensive ... Game Theory ... strategy ...

... (... : ...

Definition – let $t = s_0(i_1, j_1, k_1, l_1)s_1 \dots s_{n-1}(i_n, j_n, k_n, l_n)s_n$

be a trace of length n, let $\pi = (\pi^1, \dots, \pi^n)$ be a strategy (transmitter) of length n, and let $\lambda = \bar{i}_1 \bar{k}_1 \dots \bar{i}_n \bar{k}_n$ be a low view of length n.

1. λ is compatible with t iff $\bar{i}_r = i_r$ and $\bar{k}_r = k_r$ $(1 \le r \le n)$, i.e. λ is the low projection of t.

2. π is compatible with t iff for each r,

$$\pi^r(j_1, l_1, \dots, j_{r-1}, l_{r-1}) = j_r .$$

... high input ...

3. λ is consistent with π iff there exists a trace t such that

λ is compatible with t and π is compatible with t.

... high strategy ... low view ... high strategy ... low view ...

(... trace

Definition – We say that a synchronized state machine is nondeducible on (high) strategies iff, for any n, any low view λ of length n, λ is consistent with any high transmitter strategy π of length n.

... exclude ... π ... λ ...

Subject: درس
Date 95.1.16

- یک استراتژی های ما، ها یا hidden است، به معنی مستقل از ورودی ها تصمیم می‌گیرد. پس این تعریف ما در واقع تعریفی از یک ND اصلی است.
- ND ما، به صورت های توپولوژیک، Composable است یا Composable, ND on high strategies
- ND on high strategy نوعی از ND on high input, restrictive است.
- مثال: ND on high input را ND on high strategy می‌کند.
- مشکل: flow را از high strategy حذف می‌کند. ← تعریف flow را (تعریف پذیری از سوتر لند Sutherland)

Restrictiveness and Forward Correctability:

- در بدترین حالت برای یک سیستم، ND on high values، پیشنهاد خوبی نیست.
- Restrictiveness یک ویژگی Composable بودن از ND است. بین Composable بودن و Restrictiveness یک رابطه وجود دارد.
- تعریف: Composable است اگر بتوان به آن hook up (وصل کرد) کرد.
- hook up ← بعضی از حالت‌های بد ورودی‌های دیگر را نیز حذف می‌کند.
- Forward Correctability یک زیرمجموعه از Restrictiveness است.

PgPCO

- ⊛ یک مثال از این است که در trade-off بین کدام یک از این دو تصمیم می‌گیریم.

Subject: Formal

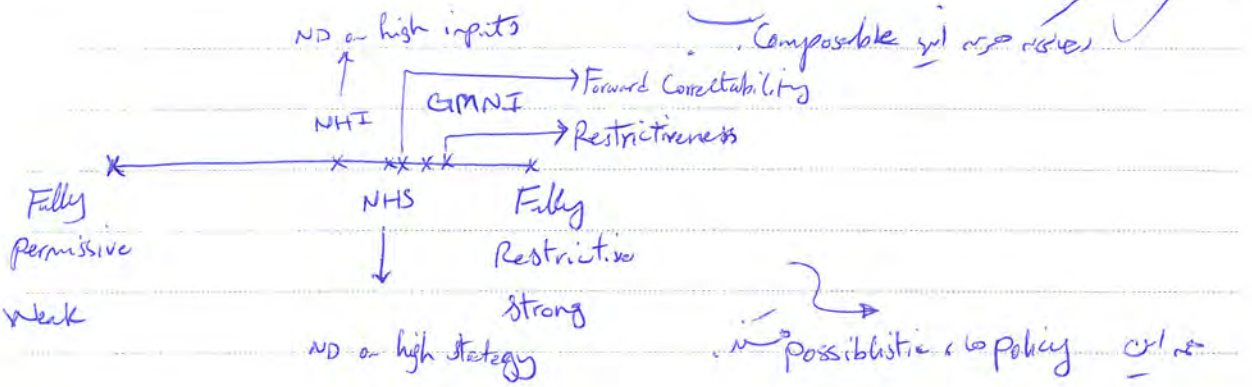
Date ۹۵/۱۲/۱

-, hookup, Composability ... McCullough ... monograph ...

... Taxonomy ← Heike Mantel ...

MASK : An assembly kit

... ND on high Strategy ... Restrictiveness ... Forward Correctability ...



... Composable ... possibilistic ... policy ...

Forward Correctability :

Forwardly correctable ... hook-up ... forwardly Correctable

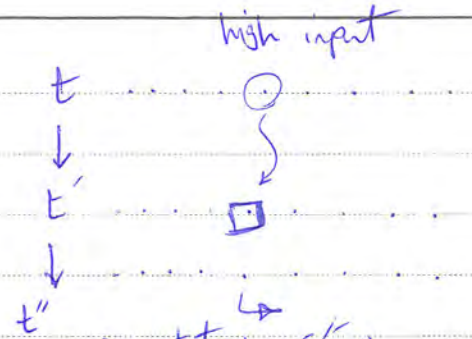
(Composability)

... Composability ... leakage ... high input

Definition — A synchronized state machine is forwardly correctable iff for any trace t of the machine and any perturbation t' given by changing a single high input, there exists a correction of t', i.e., some trace t'' that differs from t at most in its states or in high outputs that come after the changed high input.

... state ... high ...

Subject:
Date



Example – A synchronized state machine M_1,

set of states $S = \{(u,v) \mid u,v \in \{0,1,t\}\}$

$S_0 = S$

$I_R = \{r\}$, $I_T = O_R = \{0,1\}$, $O_T = \{0,1,t\}$

input receiver

Subject: Formal
Date ۹۵/۱/۲۱

state می‌تواند state، non-deterministicly تغییر می‌کند. مرحله دوم، تعداد مراحل اول state

را دنبال می‌کند.

$$S_{m-1} = (u_{m-1}, v_{m-1}) \quad , S_m = (u_m, v_m) \Rightarrow v_m = u_{m-1}$$

if $v_{m-1} = 0$ then $out_R(S_{m-1}, i_m, S_m) = k_m = 0$
if $v_{m-1} = 1$ then $out_R(\dots) = k_m = 1$
if $v_{m-1} = t$ then $out_R(\dots) = k_m = j_m$
↳ high input

$$out_T(S_{m-1}, i_m, S_m) = u_{m-1}$$

Receiver در ابتدا state خود را، همزمان با ارسال داده، به transmitter می‌فرستد.
transmitter با t روبرو شد، Receiver از ابتدا شروع می‌کند. t حرف خاصی است. معنی خاصی ندارد.

R, T توالی‌هایی هستند که معنی خاصی ندارند و فقط برای تعریف trail‌های درست و نادرست به کار می‌روند.

اصطلاحی داریم با نام ND on high states، که FL است.
Formally correctable

مثال)

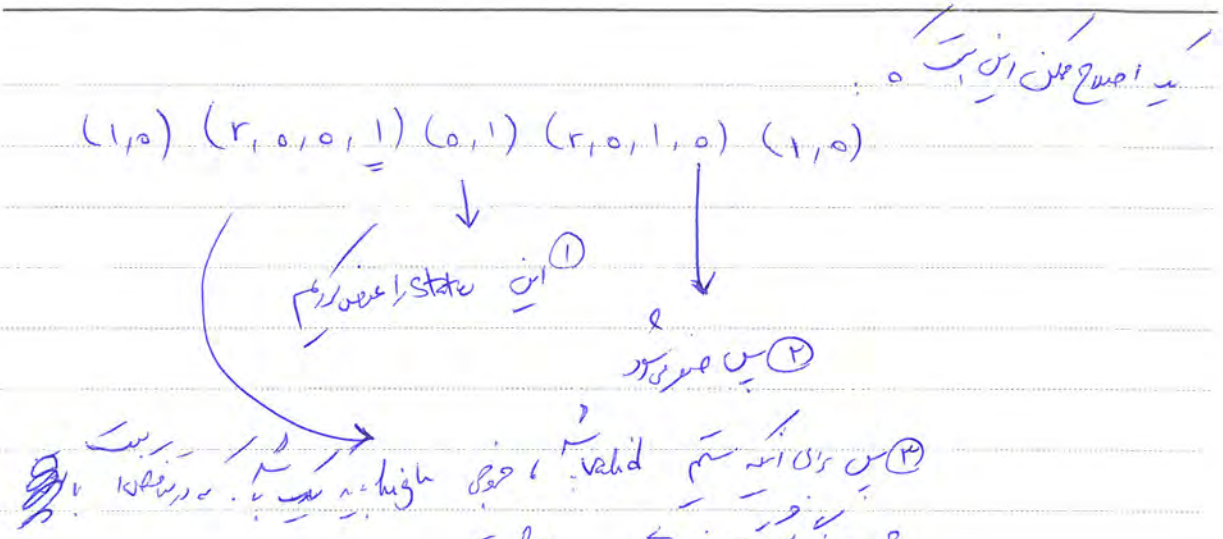
$$(t,0)(r,0,0,t)(0,t)(r,1,1,0)(1,0)$$

↓ perturbation

$$(t,0)(r,0,0,t)(0,t)(r,0,1,0)(1,0)$$

از این به بعد valid trace را بررسی می‌کنیم.

Subject:
Date



Theorem _ On the class of synchronized state machines, forward Correctability implies nondeducability on high strategies.

PqPCO

Subject: Formal
Date 40/1/22

Reading Assignment:

- Possibilistic Definitions of Security. An Assembly kit, Heiko Mantel

- Noninterference and Composability of Security, Daryl McCullough

Generalized noninterference:

Let t_1 be a trace of the system. Let t_2 be formed from t_1 by adding or deleting high-level inputs. Now, if in our system, high-level inputs do not interfere with low-level events, then the changes made in going from t_1 to t_2 should only affect high-level outputs.

Concatenation

$t_1 = \alpha . \beta_1$ → Changed Portion

$t_2 = \alpha . \beta_2$ → Constant Portion

Subject:
Date

GMNI تعریف

Definition – We will say that a system has the (generalized) noninterference property if for all traces t_1 and for all sequences t_2 formed from t_1 (Possibilistic Approach) by adding or deleting high-level inputs, there is a trace t_3 such that t_3 is the same as t_2 in the constant portion, and differs from t_2 in the changed portion only in high-level outputs.

$$t_1 = \alpha . \delta_1 \Rightarrow t_3 = \alpha . \delta_3$$
$$t_2 = \alpha . \delta_2$$

δ_3 and δ_2 are the same if we ignore high-level outputs.

Generalized Noninterference is stronger than Nondeducibility (on high inputs).

Nondeducibility بر ND on high inputs دلالت می‌کند GNI

GMNI برای GNI کافی است برای nondeterministic systems

GNI تعریف‌پذیر است اگر ND on high strategy هم‌ارز باشد

ND دلالت می‌کند GNI تعریف شود برای هر high input به low event

compatible

PqPCO

Subject: Formal
Date 99/1/24

تعریف ویژگی‌های امنیتی trace-based است.

مثال)

System A:

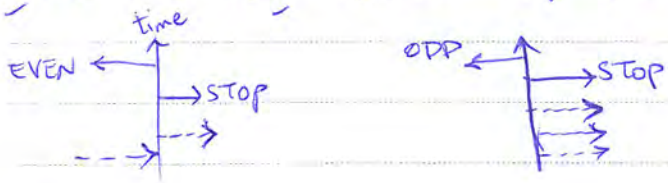
هر trace از A، با تعدادی high input شروع می‌شود و با یک low output و STOP ادامه پیدا می‌کند.

(high level: خط چین)
(low level: solid)

high output ها یا EVEN یا ODD هستند.

Parity ← تعداد high event ها قبل از STOP، زوج یا فرد

EVEN یا ODD، بستگی به زوج یا فرد بودن تعداد high output ها دارد.



از طول خط.

سوال اول: آیا این سیستم IND - Secure است؟ چون تعداد high input ها فرد یا زوج است، high output هم فرد یا زوج است.

چون اگر تعداد high input ها کم یا زیاد شود، high output اضافه شده parity بیت را درست می‌کند.

این سیستم GNI را هم برآورده می‌کند. زیرا این بیت parity، همواره درست اضافه می‌شود.

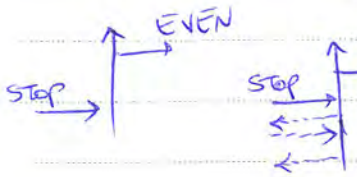
System B:

سیستم B با این تفاوت‌ها:

high output ها از طول خط حذف می‌شوند.

EVEN، ODD از طول بیت حذف می‌شوند.

STOP، در ورودی طول خط است.



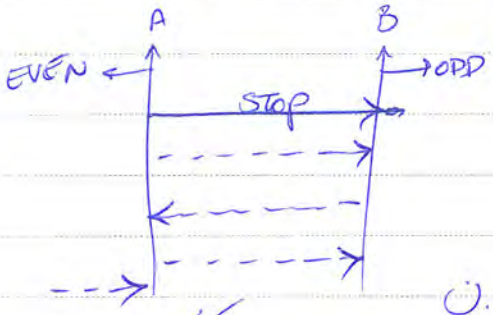
این سیستم هم IND و هم GNI را برآورده می‌کند.

حال سیستم نامنسجم (ترکیب) این سیستم‌ها چه می‌شود؟

Subject: hook-up

Date

Combining A and B:



Definition (Hook-Up Security) – We will say that a system is hook-up secure if for all traces t_1 and for all sequences t_2 formed from t_1 by adding or deleting high-level inputs, there is a trace t_3 such that t_3 is the same as t_2 in the constant portion and differs from t_2 in the changed portion only in high-level outputs, and such that the first changed output of t_3 occurs no sooner than the first output in the changed portion of t_2.

Subject: Formal
Date ۹۵/۱۲۳

۹

t_2 High t_1 High

یعنی این قضیه که اولین تعریف در جزوه t_3 ، بیشتر از اولین خروجی High

و با قضیه سوم این تعریف، جواب ثابت کرده ، Composition هم برقرار است.

مثال) با سه گروه A, B ، hook-up Secure ؟ ← دوباره تعریف

و با تعریف McCullough از Restrictiveness ، State-machine ، ...

Composability را هم دارد.

Restrictiveness :

Possibilistic هم هست.

(سوال: نمی‌دانم که با این تعریف ، enforce کنم؟ برای رفت ← دانشجوی PhD)

A hook-up theorem for multi-level security, McCullough, 1990

State Machine : states, events, transitions, initial state(s)

transition : $\sigma_0 \xrightarrow{e} \sigma_1$ ← accompanying event → input, output, internal signal of the system

states

$\sigma_0 \xrightarrow{[e_1,\dots,e_n]} \sigma_n$: transitive closure

state : event ؛ state از event ای برای ... accept ، ...

Subject:
Date ۹۵/۱/۲۸

Definition – Traces of a state machine are all sequences of events γ such that for some state σ_1, $start \xrightarrow{\gamma} \sigma_1$, where start is the initial state.

Definition – A state machine is said to be input total, if in any state it can accept any input.

این فرض برای این است که از فرستادن ورودی، leakage نداشته باشیم.

Definition – Two states σ_1 and σ_2 are low-equivalent, if they differ only in their high-level information. Two traces γ_1 and γ_2 are low-equivalent if they agree on low-level events. $\sigma_1 \approx \sigma_2$ $\gamma_1 \approx \gamma_2$

Definition – A state machine is defined to be restrictive, if

1) It is input total.

2) $\forall \sigma_1, \sigma_1', \sigma_2 . \forall \beta_1, \beta_2 . (\sigma_1 \xrightarrow{\beta_1} \sigma_1' \wedge \sigma_2 \approx \sigma_1 \wedge \beta_2 \approx \beta_1)$

state ↙ input sequence

آنگاه $\longrightarrow (\exists \sigma_2' . \sigma_2 \xrightarrow{\beta_2} \sigma_2' \wedge \sigma_2' \approx \sigma_1')$.

یعنی possibilistic

از هر دو state معادل، با ورودی‌های معادل، به state های معادل برویم.

این چنین به مدل عدم تداخل، unwinding گفته می‌شود.

تصمیم نهایی nondeterministic است.

Subject Formal
Date ۹۵/۱۰/۲۸

output sequence

3) $\forall \sigma_1, \sigma_1', \sigma_2 . \forall \gamma_1 . (\sigma_1 \xrightarrow{\gamma_1} \sigma_1' \wedge \sigma_1 \simeq \sigma_2)$

$\rightarrow \exists \sigma_2' . \exists \gamma_2 . (\sigma_2 \xrightarrow{\gamma_2} \sigma_2' \wedge \sigma_2' \simeq \sigma_1' \wedge \gamma_1 \simeq \gamma_2)$.

(remain) (produce)

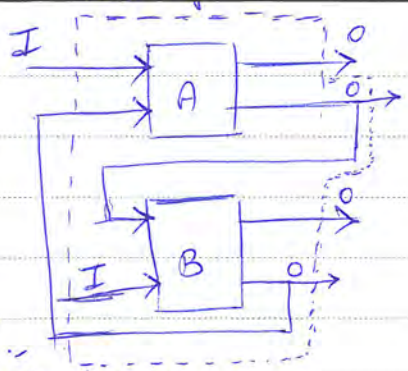
Possibilistic ← exclude

Compositionality ... GNI ، ND ، ...

event ... Composition

Subject
Date

hook-up



$\langle \sigma, \gamma \rangle$: State

State A

State B

Sequence

Communication event E_A , E_B

$\langle \sigma, \gamma \rangle \xrightarrow{\gamma} \langle \sigma', \gamma' \rangle$: transition

γ → trace

iff

$\sigma \xrightarrow{\gamma \uparrow E_A} \sigma' \wedge \gamma \xrightarrow{\gamma \uparrow E_B} \gamma'$

$\langle \sigma, \gamma \rangle \simeq \langle \sigma', \gamma' \rangle$ iff $\sigma \simeq \sigma' \wedge \gamma \simeq \gamma'$: Equivalence

$\gamma \simeq \gamma' \Leftrightarrow \gamma \uparrow E_A \simeq \gamma' \uparrow E_A \wedge \gamma \uparrow E_B \simeq \gamma' \uparrow E_B$

Subject: Formal
Date 95/1/18

Theorem - If state machines A and B are restrictive, then a composite machine forward from hooking them up is restrictive.

proof -

1) $\langle \sigma, \nu \rangle \xrightarrow{\beta} \langle \sigma', \nu' \rangle$ (input total)

$\exists \sigma' . \sigma \xrightarrow{\beta \uparrow E_A} \sigma'$
$\exists \nu' . \nu \xrightarrow{\beta \uparrow E_B} \nu'$
$\Rightarrow \exists (\sigma', \nu') . \langle \sigma, \nu \rangle \xrightarrow{\beta} \langle \sigma', \nu' \rangle$

transition

2) $\langle \sigma_1, \nu_1 \rangle , \langle \sigma_1', \nu_1' \rangle , \langle \sigma_2, \nu_2 \rangle , \beta_1 , \beta_2$

$\langle \sigma_1, \nu_1 \rangle \xrightarrow{\beta_1} \langle \sigma_1', \nu_1' \rangle \wedge \langle \sigma_2, \nu_2 \rangle \simeq \langle \sigma_1, \nu_1 \rangle \wedge \beta_1 \simeq \beta_2$

① $\sigma_1 \xrightarrow{\beta_1 \uparrow E_A} \sigma_1'$
② $\nu_1 \xrightarrow{\beta_1 \uparrow E_B} \nu_1'$

③ $\sigma_1 \simeq \sigma_2$
④ $\nu_1 \simeq \nu_2$

⑤ $\beta_1 \uparrow E_A \simeq \beta_2 \uparrow E_A$
⑥ $\beta_1 \uparrow E_B \simeq \beta_2 \uparrow E_B$

I) $\exists \sigma_2' . \sigma_2 \xrightarrow{\beta_2 \uparrow E_A} \sigma_2' \wedge \sigma_2' \simeq \sigma_1'$

II) $\exists \nu_2' . \nu_2 \xrightarrow{\beta_2 \uparrow E_B} \nu_2' \wedge \nu_2' \simeq \nu_1'$

A: restrictive ①②⑤

B: " " ③④⑥

I II $\rightarrow \exists (\sigma_2', \nu_2') . (\sigma_2, \nu_2) \xrightarrow{\beta_2} (\sigma_2', \nu_2') \wedge (\sigma_1', \nu_2') \simeq (\sigma_1', \nu_1')$.

Subject:
Date

3) $\langle \sigma_1, \gamma_1 \rangle, \langle \sigma_1', \gamma_1' \rangle, \langle \sigma_2, \gamma_2 \rangle, [e]$

$\langle \sigma_1, \gamma_1 \rangle \xrightarrow{[e]} \langle \sigma_1', \gamma_1' \rangle \wedge \langle \sigma_2, \gamma_2 \rangle \approx \langle \sigma_1, \gamma_1 \rangle$

$\sigma_1 \simeq \sigma_2$ (۱)
$\gamma_1 \simeq \gamma_2$ (۲)

(۳) $\sigma_1 \xrightarrow{[e]} \sigma_1'$

(۴) $\gamma_1 \xrightarrow{[e]\uparrow E_B} \gamma_1'$

$\exists \sigma_2' . \exists \gamma . (\sigma_2 \xrightarrow{\gamma} \sigma_2' \wedge \sigma_2' \simeq \sigma_1' \wedge \gamma \simeq [e])$ (۱)(۲) (I)

(۵) $\gamma \uparrow E_B \simeq [e] \uparrow E_B$

$\exists \gamma_2' . \gamma_2 \xrightarrow{\gamma \uparrow E_B} \gamma_2' \wedge \gamma_2' \simeq \gamma_1'$ (II)

$\exists \langle \sigma_2', \gamma_2' \rangle . \exists \gamma . \langle \sigma_2, \gamma_2 \rangle \xrightarrow{\gamma} \langle \sigma_2', \gamma_2' \rangle \wedge \langle \sigma_2', \gamma_2' \rangle \simeq \langle \sigma_1', \gamma_1' \rangle \wedge \gamma \simeq [e]$ (I)(II)

PqPCO

(به Possibilistic معروف است)

(Scheduler-independent noninterference.)

Subject: Formal — Program Analysis, Information Flow — Compositionality of Security
Date ۱۳۹۵/۱/۳۰ — Type-System Enforcement — Monitoring — Unwinding — Cloud

Subject: Computer
Date

کنفرانس CSF – S&P – PoPL – PLDI / ژورنال Journal of Comput & JCS, TISSEC, ... TOPLAS, JFP (Functional prog.)

ACM CCS, Asia CCS

- Elsevier Theoretical computer Sciences
- Information and

1 – Termination – Insensitive Noninterference (TINI)

2 – Termination – Sensitive NI (TSNI)

3 – Progress – Insensitive NI (PINI)

4 – Progress – Sensitive NI (PSNI)

(Interactive) Reactive

Termination does not just one bit of Information

(Month)

{ *nothing serious happens in Iran!
* A professor is someone who can survive himself in the center of a dessert.

#Hardworking ≠ learned

Subject: Formal
Date 98/1/20

<Program, Environment> ⇒ Environment
<expression, Env ⊢> ⇒ Value

environment
Variable → Value
E : X → V

Syntax - $e ::= b \mid n \mid x \mid e_1 * e_2$

$c ::= skip \mid c_1 ; c_2 \mid if\ e\ then\ c_1\ else\ c_2 \mid while\ e\ do\ c\ done \mid x := e$

Semantics -

Big-Step Semantics

$$\frac{}{\langle skip, E\rangle \Rightarrow E} \qquad \frac{\langle c_1, E_1\rangle \Rightarrow E_2 \quad \langle c_2, E_2\rangle \Rightarrow E_3}{\langle c_1 ; c_2, E_1\rangle \Rightarrow E_3}$$

$$\frac{\langle e, E_1\rangle \Rightarrow true \quad \langle c_1, E_1\rangle \Rightarrow E_2}{\langle if\ e\ then\ c_1\ else\ c_2, E_1\rangle \Rightarrow E_2}$$

$$\frac{\langle e, E_1\rangle \Rightarrow false \quad \langle c_2, E_1\rangle \Rightarrow E_2}{\langle if\ e\ then\ c_1\ else\ c_2, E_1\rangle \Rightarrow E_2}$$

$$\frac{\langle e, E_1\rangle \Rightarrow true \quad \langle c, E_1\rangle \Rightarrow E_2 \quad \langle while\ e\ do\ c\ done, E_2\rangle \Rightarrow E_3}{\langle while\ e\ do\ c\ done, E_1\rangle \Rightarrow E_3}$$

$$\frac{\langle e, E\rangle \Rightarrow false}{\langle while\ e\ do\ c\ done, E\rangle \Rightarrow E} \qquad \frac{\langle e, E\rangle \Rightarrow v}{\langle x := e, E\rangle \Rightarrow E[x \mapsto v]}$$

$$\frac{}{\langle n, E\rangle \Rightarrow n} \qquad \frac{}{\langle b, E\rangle \Rightarrow b} \qquad \frac{E(x) = v}{\langle x, E\rangle \Rightarrow v} \qquad \frac{\langle e_1, E\rangle \Rightarrow v_1 \quad \langle e_2, E\rangle \Rightarrow v_2}{\langle e_1 * e_2, E\rangle \Rightarrow v_1 * v_2}$$

γ → Variable Environments

↳ E

Subject:

Date

A Perspective on Information Flow Control, Hedin and Sabelfeld

Termination-insensitive Non-interference leaks more than just a bit. Askarov Sabelfeld

Security policy با lattice:

$(\{H, L\}, \leq)$ with $L \leq H$

$\sigma ::= H \mid L$ Security labels

$\Gamma : X \to \{H, L\}$

Γ : Variable Security map

TINI, TSNI برای NI

Low-Equivalence ($\sim$):

$$\frac{\sigma = L \Rightarrow v_1 = v_2}{v_1 \sim_\sigma v_2}$$

$$\frac{\forall x \in dom(\Gamma).\ \gamma_1(x) \sim_{\Gamma(x)} \gamma_2(x)}{\gamma_1 \sim_\Gamma \gamma_2}$$

environment

observable (behavior)

indistinguishable

$\gamma_1(x)$:

Subject: Formal
Date ۱۳۹۵/۲/۱۴

$\Downarrow$: Evaluation

$$\frac{\langle c, E_1\rangle \Rightarrow E_2}{\langle c, E_1\rangle \Downarrow E_2}$$

↳ observable behavior

$$\frac{\neg \exists E_2 . \langle c, E_1\rangle \Rightarrow E_2}{\langle c, E_1\rangle \Downarrow \diamond}$$

↳ divergence (silent)

$$\diamond \simeq_{TI} O_2 \qquad O_1 \simeq_{TINI} \diamond$$

↳ Termination Insensitive

$$\frac{E_1 \sim_L E_2}{E_1 \simeq_{TI} E_2}$$

$$TINI(c) = \forall E_1, E_2 . E_1 \sim_L E_2 \wedge (c, E_1) \Downarrow O_1 \Rightarrow \exists O_2 . \langle c, E_2\rangle \Downarrow O_2 \wedge O_1 \simeq_{TI} O_2$$

↓ Program

$\checkmark$ termination ... high ... $\times$ ... $\checkmark$

$c =$ if (secret=0) then public=1 else while true do skip done. (...)

PdPCO ... TINI-Secure ...

Subject:
Date

$$\frac{E_1 \sim_\Gamma E_2}{\diamond \simeq_{TSNI} \diamond} \qquad \frac{E_1 \sim_\Gamma E_2}{E_1 \simeq_{TSNI} E_2}$$

$$TSNI(c) = \forall E_1, E_2 .\ E_1 \sim_\Gamma E_2 \wedge (c, E_1) \Downarrow o_1 \Rightarrow \exists o_2 .\ \langle c, E_2 \rangle \Downarrow o_2 \wedge o_1 \simeq_{TSNI} o_2$$

Termination-Sensitive Indistinguishable

closed

Reactive

Environment

Progress

PSNI, PINI

Big-Step ، Small-Step Semantics

PINI and PSNI:

$$c ::= skip \mid c_1 ; c_2 \mid \text{if } e \text{ then } c_1 \text{ else } c_2 \mid \text{while } e \text{ do } c \text{ done} \mid x := e \mid in_\delta\ x \mid out_\delta\ x$$

PgPCO

Subject: Formal
Date ۹۵/۲/۱۴

silent divergence : ...output ... silent div. ... divergence ...

... PINI ... diverge ... PSNI ... high ...

$E : (\gamma, \iota, \omega)$

$\langle c, E \rangle$: evaluation context

Semantics: (Small-step)

$\langle skip, E \rangle \rightarrow E$

$\dfrac{\langle c_1, E_1 \rangle \xrightarrow{\dot{v}} \langle c_1', E_2 \rangle}{\langle c_1 ; c_2, E_1 \rangle \xrightarrow{\dot{v}} \langle c_1' ; c_2, E_2 \rangle}$

$\dot{v}$ = possible observable behavior

$\dfrac{\langle c_1, E_1 \rangle \xrightarrow{\dot{v}} E_2}{\langle c_1 ; c_2, E_1 \rangle \xrightarrow{\dot{v}} \langle c_2, E_2 \rangle}$

$\langle c, E_1 \rangle \xrightarrow{\dot{v}} E_2$: terminating transition

$\langle c, E_1 \rangle \xrightarrow{\dot{v}} \langle c_2, E_2 \rangle$: non-terminating transition

rotation: $\downarrow_L$, $\downarrow_H$ input $\rightsquigarrow (v . \bar{v}_L, \bar{v}_H) \downarrow_L (v, (\bar{v}_L, \bar{v}_H))$ (concat)

$(\bar{v}_L . \bar{v}_H) \downarrow_H (v, (\bar{v}_L, \bar{v}_H))$: ...

۹۴/

Subject:
Date

- Formal Semantics of Programming Languages, Winskel
- ~~Foundations of Programing languages~~, Mitchell, chapter 1,
- Practical Foundation for programing Languages, R. Harper

$\uparrow_L$, $\uparrow_H$: ... output ...

$$(v, (\bar{v}_L, \bar{v}_H)) \uparrow_L (v.\bar{v}_L, \bar{v}_H)$$

$$(v, (\bar{v}_L, \bar{v}_H)) \uparrow_H (\bar{v}_L, v.\bar{v}_H)$$

: Semantics ...

(... Fixed point ... Co-Induction ... Fixed-point ... Induction ... Co-Induction ... Induction ...)

$$\frac{\langle e, E\rangle \Rightarrow \text{true}}{\langle \text{if } e \text{ then } c_1 \text{ else } c_2, E\rangle \longrightarrow \langle c_1, E\rangle} \qquad \frac{\langle e, E\rangle \Rightarrow \text{false}}{\langle \text{if } e \text{ then } c_1 \text{ else } c_2, E\rangle \longrightarrow \langle c_2, E\rangle}$$

$$\frac{\langle e, E\rangle \Rightarrow \text{true}}{\langle \text{while } e \text{ do } c \text{ done}, E\rangle \longrightarrow \langle c; \text{while } e \text{ do } c \text{ done}, E\rangle}$$

$$\frac{\langle e, E\rangle \Rightarrow \text{false}}{\langle \text{while } e \text{ do } c \text{ done}, E\rangle \longrightarrow E} \qquad \frac{\langle e, E\rangle \Rightarrow v}{\langle x := e, E\rangle \longrightarrow E[x \mapsto v]}$$

$$\frac{t_1 \downarrow_\sigma (v, t_2)}{\langle in_\sigma\ x, (\gamma, t_1, w)\rangle \longrightarrow (\gamma[x \mapsto v], t_2, w)}$$

$$\frac{\gamma(x) = v \qquad (v, w_1) \uparrow_L w_2}{\langle out_L\ x, (\gamma, t, w_1)\rangle \xrightarrow{v} (\gamma, t, w_2)}$$

$$\frac{\gamma(x) = v \qquad (v, w_1) \uparrow_H w_2}{\langle out_H\ x, (\gamma, t, w_1)\rangle \longrightarrow (\gamma, t, w_2)}$$

PqPCO

... observable ...

Subject: Formal
Date 98/۲/۶

- A perspective ...
- Termination-insensitive
- Reactive Noninterference

Askarov, ...

$$\frac{\sigma = L \Rightarrow v_1 = v_2}{v_1 \sim_\sigma v_2} \qquad \frac{\forall x \in dom(\Gamma) \cdot \gamma_1(x) \sim_{\Gamma(x)} \gamma_2(x)}{\gamma_1 \sim_\Gamma \gamma_2}$$

$$\frac{}{(\bar{v}_1, \bar{v}_1') \sim (\bar{v}_1, \bar{v}_2')}$$

$$\frac{\gamma_1 \sim_\Gamma \gamma_2 \quad \ell_1 \sim \ell_2 \quad \omega_1 \sim \omega_2}{(\gamma_1, \ell_1, \omega_1) \sim_\Gamma (\gamma_2, \ell_2, \omega_2)}$$

... R : ranges over execution contexts $\langle C, E \rangle$ and environments E.

... $\langle C, E \rangle$ ... R ... E ...

We define $\langle C, E \rangle \overset{v}{\Rightarrow} R$, to capture evaluation until observable output or termination, with or without observable outputs.

$$\frac{\langle C_1, E_1 \rangle \longrightarrow^* \langle C_2, E_2 \rangle \qquad \langle C_2, E_2 \rangle \xrightarrow{v} R}{\langle C_1, E_1 \rangle \overset{v}{\Rightarrow} R}$$

$$\frac{\langle C, E_1 \rangle \longrightarrow^* E_2}{\langle C, E_1 \rangle \Rightarrow E_2}$$

... observable output ... observable behavior

Subject:
Date

Extending $\overset{v}{\Rightarrow}$ to $\overset{vs}{\Rightarrow}$:

$$\frac{\langle C_1, E_1\rangle \overset{v}{\Rightarrow} \langle C_2, E_2\rangle \quad \langle C_2, E_2\rangle \overset{vs}{\Rightarrow} R}{\langle C_1, E_1\rangle \overset{v.vs}{\Rightarrow} R}$$

$$\frac{}{\langle C, E\rangle \Rightarrow \langle C, E\rangle}$$

null

"()"

$\neg \exists R, vs . \langle C, E\rangle \overset{vs}{\Rightarrow} R$ implies that $\langle C, E\rangle$ diverge silently.

The judgment $\langle C, E\rangle \Downarrow vs$:

$$\frac{\langle C_1, E_1\rangle \overset{vs}{\Rightarrow} \langle C_2, E_2\rangle \quad \neg \exists R, vs' . \langle C_2, E_2\rangle \overset{vs'}{\Rightarrow} R}{\langle C_1, E_1\rangle \Downarrow vs . \Diamond}$$

$$\frac{\langle C_1, E_1\rangle \overset{vs}{\Rightarrow} R E_2}{\langle C, E\rangle \Downarrow vs}$$

حال تعریف PINI :

$O_1 \simeq_{PI} O_2$ iff $O_1 = O_2 \vee (\exists o_1, o_1' . O_1 = o_1 . \Diamond \wedge O_2 = o_1 . o_1')$

$\vee (\exists o_2, o_2' . O_1 = o_2 . o_2' \wedge O_2 = o_2 . \Diamond)$.

$\forall E_1, E_2, E_1 \sim E_2 \wedge (C, E_1) \Downarrow O_1 \xRightarrow{\text{Then}} \exists O_2 . \langle C, E_2\rangle \Downarrow O_1 \wedge O_1 \simeq_{PI} O_2$.

تعریف PSNI :

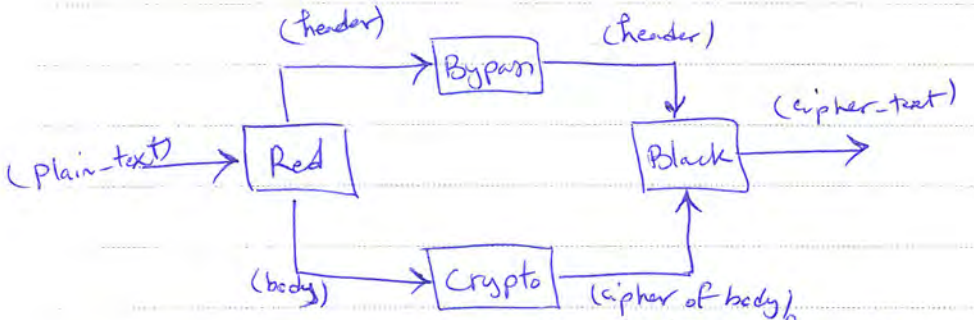
$O_1 \sqsubseteq_{PS} O_2$ iff $O_1 = O_2$.

$\forall E_1, E_2, E_1 \sim E_2 \wedge (C, E_1) \Downarrow O_1 \xRightarrow{\text{Then}} \exists O_2 . \langle C, E_2\rangle \Downarrow O_1 \wedge O_1 \sqsubseteq_{PS} O_2$.

Subject: Formal
Date 93/12/11

Intransitive Noninterference:

A paradigmatic example:



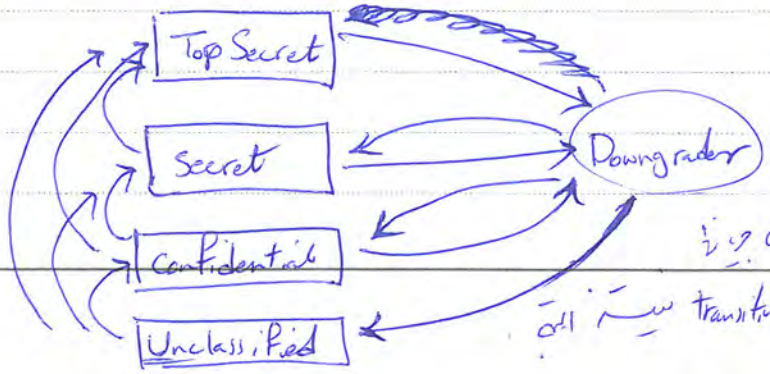
channel control policy

Security domains

routing

Bypass Crypto Red Black

transitive



transitive

MLS

downgrader

transitive

Subject:
Date

Complete Isolation برای ایجاد امنیت خط‌مشی‌ها و رسیدن به Verification.

downgrader را تعریف می‌کند، که می‌تواند داده‌ها را از Top Secret به Confidential تغییر دهد.

این ابزار به Verification اجازه می‌دهد تا برای Specification استفاده کند.

NI به‌صورت گرافی نمایش داده می‌شود. هر arrow نشان‌دهنده non-interfering relation است.

NI relation بین Red و Black برقرار است، زیرا هیچ action از Red نمی‌تواند مستقیماً بر Black تأثیر بگذارد.

NI توسط Bypass می‌شود، زیرا encryption اجازه می‌دهد اطلاعات plaintext به‌صورت ciphertext منتقل شود.

assertion برای Red و Black در NI بررسی می‌شود. اگر این assertion نقض شود، خطا گزارش می‌شود.

Flow برای هر کاربر تعریف می‌شود و فقط Flow‌های مجاز اجازه عبور دارند.

NI به‌صورت همزمان اجرا می‌شود. حالت‌های ممکن این Policy را بررسی می‌کند و نتایج را مقایسه می‌کند.

NI نسبت به Polling‌های مخرب امن است، زیرا همه کاربران به‌صورت همزمان بررسی می‌شوند.

این سیستم برای هسته‌ها مناسب است، اما برای برنامه‌های کاربردی انعطاف‌پذیری ندارد.

PqPCO

Subject: Formal
Date ۹۵/۲/۱۱

Definition - A system (Machine) M is composed of

- a set S of states and the initial state $S_0 \in S$
- a set A of actions
- a set O of outputs
- step: $S \times A \longrightarrow S$
- output: $S \times A \longrightarrow O$

این سه تا را سیستم دارد

خروجی action ای که انجام می‌شود

تابع run که دنباله‌ای از actionها را می‌گیرد و حالت آخر را برمی‌گرداند:

$$run : S \times A^* \longrightarrow S$$

lambda

$$run(s, \Lambda) = s$$

$$run(s, a \circ \alpha) = run(step(s,a), \alpha)$$

(Concat)

A set D of security domains. ⟵ low, high

A partial order بین آن‌ها

A function $dom : A \longrightarrow D$.

A security policy is specified by a reflexive relation $\rightsquigarrow$ on D.

خاصیت reflexive

P4PCO

99/

این رابطه باید transitive, partial order باشد طبق Denning

Subject:
Date

Noninterference Interference

$\not\rightsquigarrow : (D \times D) \setminus \rightsquigarrow$

- A policy is said to be transitive if its interference relation is transitive.

Definition - For $v \in D$ and $\alpha \in A^*$, we define purge (α, v) as follows:

$$purge(\Lambda, v) = \Lambda$$

$$purge(a \circ \alpha, v) = \begin{cases} a \circ purge(\alpha, v) & \text{if } dom(a) \rightsquigarrow v \\ purge(\alpha, v) & \text{o.w.} \end{cases}$$

Security:

$$\forall a. \forall \alpha. output(run(s_0, \alpha), a) = output(run(s_0, purge(\alpha, dom(a))), a)$$

Notation -

$$do(\alpha) = run(s_0, \alpha)$$
$$test(\alpha, a) = output(do(\alpha), a)$$

Thm,

$$test(\alpha, a) = test(purge(\alpha, dom(a)), a).$$

Subject: Formal

Date 98/8/11

Noninterference and Transitivity:

مثال اول این بحث را برای ما آشنا می‌کند:

Red ⇝ Bypass

Red ⇝ Crypto

Bypass ⇝ Black

Crypto ⇝ Black

و همچنین رابطه reflexive نیز هست.

حال Policy ما:

Red ⇏ Black even though

Red ⇝ Bypass and

Bypass ⇝ Black

(نشان می‌دهد که رابطه Intransitive است)

- The assertion Red ⇏ Black means that there must be no way for Black to observe activity by Red. This is not what is required here; Black must certainly be able to observe activity by Red. But, we want all such observations to be immediated by the Bypass or the Crypto.

پس تعریف ما، خواسته‌ای از کاربردهای ما نیست و باید به دنبال راه دیگری باشیم.

Subject:
Date

Definition - Let L be a set of security labels with a partial ordering $\leq$. Let clearance : $D \to L$ be a function that assigns a fixed security label to each domain in D. Then, the multi-level security (MLS) policy is :

$u \rightsquigarrow v$ iff clearance $(u) \leq$ clearance (v). $\circledast$

($\leq$: is dominated by)

- An arbitrary security policy given by a relation $\rightsquigarrow$ on D is said to be an MLS-type policy if a label set L with a partial ordering $\leq$ and a function clearance : $D \to L$ can be found such that $\circledast$ holds.

Theorem - All MLS-type policies are transitive and vice versa.

Proof -

$u \rightsquigarrow v$ and $v \rightsquigarrow w \Rightarrow$ clearance $(u) \leq$ clearance $(v) \wedge$ clearance $(v) \leq$ clearance (w)

$\Rightarrow$ clearance $(u) \leq$ clearance $(w) \Rightarrow u \rightsquigarrow w$ (Partial order)

PqPCO

transitive ; MLS

Subject: Formal
Date ۹۵/۲/۱۱

جلسه‌ی دوازدهم:

Definition - Define the relation $\leftrightarrow$ on D as follows:

$$u \leftrightarrow v \overset{def}{=} u \rightsquigarrow v \wedge v \rightsquigarrow u.$$

$\leftrightarrow$ is symmetric and also reflexive, and of course transitive, because of the hypothesis ($\rightsquigarrow$ is transitive). So $\leftrightarrow$ is an equivalence relation. Thus, we have $[u]$ is the equivalence class of u.

We define $\leq$ on L (L is the set of equivalence classes of $\leftrightarrow$) as follows:

$$[u] \leq [v] \overset{def}{=} \exists x \in [u] . \exists y \in [v] . x \rightsquigarrow y.$$

So, $\leq$ is a partial order on L.

(precedes or equal) "Precq"

$\leq$ is reflexive, asymetric, and transitive.

$[x]=[x']$, $x \rightsquigarrow y$, $y' \rightsquigarrow x'$, $\Rightarrow [x]=[y]$
$[y]=[y']$

Define clearance : D $\longrightarrow$ L as clearance $[x] \overset{def}{=} [x]$.

It is easy to verify that $u \rightsquigarrow v$ iff clearance $[u] \leq$ clearance $[v]$.

$\Rightarrow$ is MLS-type policy

Subject: Non-interference, Transitivity and Channel Control Security / John Rushby.
Date

نکته: GMNI معادل مناسب است MLS مرتبط تر است و در خط‌مشی‌های intransitive غیر MLS می‌شود.

نکته: Unwinding هم در مقاله بالا مطرح شده است. خوانده شود.

اولین بار توسط و در سطح شد.

راستی‌آزمایی مجدد هم اجرا برای Possibilistic NI نکته و کمی پیچیده است.

با توجه به ساختار transition ها و state ها باید برای اجرای کلی به دنبال خواص بود.

برای transition ها، خواص را بیان و اثبات کنیم.

اول: Intransitive

پس از مرحله purge را بگیرد. اسمش را Secret بگذاریم و به viewer ای برگردان اصلح.

Sanitize پس شخص خروجی آن Secret داریم.

Definition - We define the function

$$Sources : A^* \times D \longrightarrow \mathcal{P}(D)$$

(Powerset)

by the equation

$$Sources(\Lambda, u) = \{u\}$$

$$Sources(a \circ \alpha, u) = \begin{cases} Sources(\alpha, u) \cup \{dom(a)\} & \text{if } \exists v \in Sources(\alpha, u).\ dom(a) \rightsquigarrow v \\ Sources(\alpha, u) & \text{otherwise} \end{cases}$$

PgPCO

Subject: Formal
Date ۹۵/۲/۱۴

تعریف متعدی: اگر $A \rightsquigarrow B$ و $B \rightsquigarrow C$ نتیجه می‌دهد $A \rightsquigarrow C$ آنگاه گوییم $\rightsquigarrow$ متعدی است.

در مثال بالا اگر $u = C$ باشد، حال می‌خواهیم ورودهایی بسازیم که خروجی را عوض کنند.

purge برای A: پس Flow C به B و B به A است. پس A باید Follow شود.

(در مقاله هم توضیح این قسمت آمده.)

Definition: The function ipurge (intransitive purge)

$$ipurge : A^* \times D \longrightarrow A^*,$$

is defined by:

$$ipurge(\Lambda, u) = \Lambda$$

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

خروجی در Sources نباشد، حذف می‌شود.

Security: The machine is secure for the policy $\rightsquigarrow$ if

$$test(\alpha, a) = test(ipurge(\alpha, dom(a)), a).$$

Subject:
Date

intransitive ، transitive است ، تعریف جدید transitive

را می‌دهد. این تعریف به ما کمک می‌کند تا بتوانیم ... transitive ، ... Plan های ... را ... کنیم.

(intransitive NI یا ...)

... Multi-Threading ، Concurrency ... :

Noninterference in Concurrent and Multi-threaded Systems:

... Concurrency ... multi-threading ... می‌شود.

... Thread ... fork ، start ... Thread

... Scheduler ... Thread ، ... Sabelfeld ، Mantel ...

... Harvard ، Cornell ، Pennsylvania ، KTH ، ETH ، CMU ...

Morrisett Myers Stankovic Sabelfeld

Observational Determinism:

... Dam Kovic ، Myers ... Roisco ، McLean ...

CCS ... Concurrency ... → Tony Hoare (Observational Determinism For Concurrent Security)

Concurrent Segential process ... Applied-Pi ... Pi ...

Roisco (CSP)

... Process Calculus ... Core ...

Subject: Formal
Date
۹۵/۲/۱۴

Notation - $\mathcal{L}$: A lattice of security labels

$(\preceq)$ $\sqsubseteq$: Partial-order relation

$l_1 \preceq l_2$: l_1 is protected by l_2. l_1 توسط l_2 محافظت می‌شود.

l_2 حساس‌تر از l_1 است.

اگر داده‌های l_1 با l_2 ترکیب شوند، نتیجه حداقل به اندازه l_2 حساس در نظر گرفته می‌شود.

$\approx_\xi$ low-equivalent relation

آن‌ها در سطح H, L تفاوت دارند، به شرطی که از دید سطوح پایین‌تر یکسان به نظر برسند.

$e \approx_\xi e'$ if they differ only in high-security input data.

Program Expressions

high-security data is any whose label l does not satisfy $l \preceq \xi$.

داده‌ای که برچسب آن ξ نباشد.

$\langle h, l \rangle$ نوع داده‌ها

high-Security data

low-Security data

$\langle$attack at dawn, 3$\rangle \not\approx_L \langle$do not attack, 4$\rangle$

$\langle$attack at dawn, 3$\rangle \approx_L \langle$do not attack, 3$\rangle$

PgPCO

۱۰۷

Subject:
Date

Noninterference تعریف:

If $e \underset{\xi}{\simeq} e'$, then $(e \Downarrow v \wedge e' \Downarrow v') \Rightarrow v \underset{\xi}{\simeq} v'$.

در این تعریف ساده: termination را لحاظ کرده؟ فرض کرده است که evaluation های expression ها

ما فعلا فرض می‌کنیم، خواستیم فرق را داریم. از termination صرف نظر می‌کنیم و خیلی سخت افزارها به درستا فرض

علوم رایانه، طراحی زبان، سیستم ها کار می‌کنیم.

از طرف سیستم ها با کدهای بدی تعامل دارند و خیلی سناریوهای خاص را نیاز می‌کنند. به دلیل این موارد خیلی ضروری، علوم مختلف

می‌کنند. پس این تعریف به این دردها در این کار نمی‌خورد.

* Scheduling در Semantics دخیل است. در سیستم عامل ها کار می‌کنیم، Scheduling های مختلف داریم.

$m = \langle M, e \rangle$ is a configuration

Program expression (e) — Memory State (M)

$m \Downarrow T = [M_0, M_1, M_2, \dots]$

evaluate — execution trace

خود سمانتیک ها را انجام می‌کند، وضعیت های مختلف را نگه می‌دارد.

Subject: Formal
Date ۹۵/۴/۱۴

: Semantics

$$\frac{\langle M, e_1\rangle \longrightarrow \langle M', e_1'\rangle}{\langle M, e_1 | e_2\rangle \longrightarrow \langle M', e_1' | e_2\rangle}$$

Concurrency

$$\frac{\langle M, e_2\rangle \longrightarrow \langle M', e_2'\rangle}{\langle M, e_1 | e_2\rangle \longrightarrow \langle M', e_1 | e_2'\rangle}$$

معماری اجرای یک برنامه Concurrent معمولاً است، موازی سازی، عدم قطعیت را نشان می‌دهد. یک Scheduler

است که به طوری تعیین کند که هم می‌خواهیم کدام چی شود، است نرم‌افزار.

If $m_1 \approx_\xi m_2$, then

$$\forall T_1 \cdot m_1 \Downarrow T_1 \Rightarrow \exists T_2 \cdot m_2 \Downarrow T_2 \wedge T_1 \approx_\xi T_2 \text{, and}$$

$$\forall T_2 \cdot m_2 \Downarrow T_2 \Rightarrow \exists T_1 \cdot m_1 \Downarrow T_1 \wedge T_1 \approx_\xi T_2 .$$

این تعریف عدم قطعیت را حل می‌کند اما مناسب نیست. مثالی را ببینیم:

~~[illegible]~~

$$l := true \mid l := false \mid l := h$$

~~[illegible]~~

فضای متغیرهای ما برای ما مهم است. برنامه بالا را هم‌ارز با برنامه زیر است:

$h := true \longrightarrow \quad l := true \mid l := false \mid l := true$

$h := false \longrightarrow \quad l := true \mid l := false \mid l := false$

low-equivalent

P4PCO

(امین سیستم ← ۱۳۹۵/۳/۴)

Subject:
Date:

برای حصر اجرای

اولین از اولی ، اجرای از روی وجود دارد. تعداد ها در حالت پایانی ممکن است چنین نرود. روی نوع ادی

پس حالت تعریف اجرا ، این نیز جایز است . هم خبر possible است .

(تصادفی)

فرض کنیم Scheduler ای داریم ، دستور سوم از چپ است) را اجرا می‌کند. یعنی یک ارجاع است

اجرا می‌شود. پس attacker اگر باشد ، Scheduler اینچنین است ، پس noiseless ثبت داریم!

اگر این چنین می‌دانیم که Scheduler با احتمال خاص دستورات را اجرا می‌کند ، می‌توانیم حدس بزنیم . چون با احتمال هم را متوجه می‌شویم

ن h خواهیم بود . مثال احتمالاتی به چنین داریم. اصطلاحاً Refinement attack داریم . چون ، سعی در refine

Scheduler هم ، عمل می‌کند . اگر احتمالاتی از Scheduler خاص داشته باشیم ، می‌توانیم خبری درباره اینکه high

به دست بیاوریم . حالت اصلی هم ورودی این است .

ن Semantics در Scheduler با Concurrent به کار می‌رود ، Multi-Threaded از این است

نیست و با درست کردن محاسبات خط به خط ، و درست کردن پیمود در کاری ، نمی‌توان نهایی اجرای برنامه چه می‌شود می‌بینیم .

و دیدیم که پیمود Possibilistic مناسب نیست . چون مثال قبلی را ایمن می‌داشت .

برای جلوگیری از این نامنی ، تعریف دیگری نیاز داریم . صحبت از timing های internal, external

نیز در این قسمت اهمیت دارد .

PqPCO

Subject: Formal
Date 95/2/18

Internal and External Timing:

Example — x := true;
(if h then delay(100) else skip; x := false)
| (delay(50); l := x; ...)

thread ها توسط scheduler هم مجدد بندی می‌شوند.

احتمالا ابتدا thread اول اجرا می‌شود و h=false، سپس x برابر false

در Round Robin scheduler پس l=h خواهد شد و این نوع leakage

برنامه ایمن است.

Example — x := true;
(if h then delay(100) else skip; x := false)

در این حالت بسته به اینکه scheduler چه زمانی x را false کند، h می‌تواند متفاوت باشد.

مثال اول Internal Timing channel و مثال دوم External Timing channel است.

Internal Timing channel: زمان اجرا برابر Running Time است.

External Timing channel: زمان اجرا برابر Running Time نیست.

راه‌کار: Padding ⇒ کدهای اضافی برای branch ها قرار می‌گیرد تا زمان اجرای آن‌ها برابر شود.

این راه‌کار برای Recursion و Loop هم قابل استفاده است.

دلیل External است.

Subject:
Date

از بررسی مدل (observational Determinism) OD ، multi-Threaded systems در

آنکه Layers ، External را می‌توان ... Solution کلیه تهدیدهای خارجی

Restrictive خواهد بود.

تعبیری به سطح بالا ، در اینجا ، Mclean است.

If $m \underset{\xi}{\approx} m'$, then $(m \Downarrow T \wedge m' \Downarrow T') \Rightarrow T \underset{\xi}{\approx} T'$.

Configuration

هر جفتی از این دو Configuration ،
trace ایجاد می‌شود با این
trace ها معادل باشند.

حال تعریف معادل بودن دو trace اهمیت دارد. کدام است trace از نظر خروجی یکی باشد.

تعریف possibilistic ، ∃ (حداقل یکی) است ، اما برای هر ∀ داریم پس طبق این تعریف

مثال مثبت داشتن است.

و جهت تعریف Observational Determinism این است که مشاهده‌گر سطح پایین و بالا سیستم غیرمحض

سیستم را قطعی می‌بیند. (Low-Security Observational Determinism)

البته از model checking برای O.D استفاده کرد. enforcement مراحل مختلف

و بررسی مدل از نوع خروجی خود.

PqPCO

Subject: Formal
Date ۹۵/۲/۱۸

فرض داشتیم: $T = [M_0, M_1, \dots]$ (به trace، تاریخچه هم می‌گفتیم)

تعریفی به ازای هر location از trace ها داشتیم این است که هر projection ای از trace روی location های مختلف چیست.

$$T(loc) = [M_0(loc), M_1(loc), \dots]$$

برای هر location یک دید یا observer داشتیم که در واقع trace های قابل مشاهده‌ی آن را می‌بیند.

و تعریف کردیم termination و prefix و up-to-prefix و ... را.

و همچنین up-to-stuttering که $T(loc) = Prefix(T'(loc))$ یعنی از جایی به بعد ثابت شده.

به تمرین سوم که رسیدیم، متغیرهایی داشتیم که state را ذخیره می‌کردند و سطح هم داشتند که low یا high بودند.

سطح متغیر O.D. نیز وجود دارد، و race-freedom برای متغیرهای سطح ثابت.

race به این صورت تعریف شد که دو thread به یک متغیر مشترک دسترسی داشته باشند که حداقل یکی از آن‌ها write باشد.

در ادامه می‌خواهیم یک Type System داشته باشیم که enforce کند.

مثلاً در O.D. یک مثال $l := true \mid l := false$ را هم داشتیم که نامن است!! این هم مثالی بود که ...

Cache اتفاق می‌افتد و یک متغیر high ای در آن استفاده می‌شود، چه مشکلی پیش می‌آید؟! در واقع این یک false positive است که ما نباید این را داشته باشیم.

Subject:
Date

بعضی وقتی race flow محدودیت ۳ ، در جا محسوس اجرا کرد ، حالا همین ایکس

شاید مثل اصلا متغیر high هم داریم!

low observation determinism (۱ ... Approach سه

۲) Extended Timing-based ۳) مدل هایی از scheduler ها میدا میکنند که اجرای (یا اجرایی نیست)

کی بیان کن؟

کتابخانه ها را زیاد ها این یک.

(دیگر روی این صفحه ادامه نمی دهیم، یک کپی است.)

نسخه ی صفحه ی قبل

یک trend ، اما اینجا نیست. حتی عده ای که حضور دارند به دانشجو روی تعبیر اهمیتی از خود ، هر چی بیشتر

تعبیر اهمیتی را میخواهند کردن ، بعد میروند سراغ Enforcement که آن جدی است.

حال این ایده های اجمالی ، در دل زبان اجرا می کند (البته این ایده ها می تواند مرجع abstraction قسمتی از اجرا

قابل پیاده سازی است اما مجموعه در سطح code حرف میزنیم.)

پس خود کلمه اجمال این خاصیت ها و تعبیر هم است.

کتاب را عنوانی از سایت کامپوزیت.

نتیجه را هم بالاتر از سایت قرار داد.

مقاله را از همین سایت پیدا کردن و دانلود کردن.

[1] A Survey of runtime policy enforcement techniques and Implementation, 2011, Terento University.

PqPCO

البته خیلی هم خوب نیست ولی خوب زیاد! technical report است.

Subject: Formal
Date ۹۵/۲/۱۸

Enforcement :

کلیه ابزارهای زیر جزء این دسته هستند

Static : Data-Flow Analysis, Type Systems, Model checking, Abstract Interpretation, ...

Dynamic

هدف از تکنیک‌های Static، برای جلوگیری از بروز خطاهای احتمالی در زمان اجرا است و هدف این است که از این به صورت اجرایی، ...

... objective یعنی برنامه را با مشخصه‌هایی که دارد، با چیزی که باید باشد (Objective) ...

... data-flow است.

... Fleming Nilsen ... Program Analysis ... کلیه ابزارهای ... را دارد.

هدف تکنیک‌های این است که ... representation ... برنامه را قبل از اجرا ... کند.

Model checking ... Non-Standard Semantics ، Abstract Interpretation ...

... برای ... برنامه‌های ... اجرا ... است ... حالت به شکل ... (state machine) ... Type System ... تکنیک‌ها

... است ... Preservation ، Progress را ... و ... این ...

... Data flow ... را ...

... Static ... است. ...

... False Positive ... undecidable ...

Subject:
Date

همه روش‌ها باید Soundness را داشته باشند، یعنی از روش استفاده کردیم این است، برنامه واقعی خطا نشان را دارد نه. این کمتر است که خطا نباشد ولی برنامه از آن گذر کند، اما روش‌های کلی استاتیک برای برنامه‌های اخیر بطور Complete نیستند. مزیت آن این است که overhead بالایی در اجرا ندارند یا خیلی کم دارند.

Dynamic → offline → خطاهایی که استاتیک نمی‌توانستند انجام دهند، در زمان اجرا انجام می‌دهد. سیستم را اندازه‌گیری می‌کند.
Run-time → enforcement می‌کند. dynamic offline و static

برنامه را اجرا می‌کند. البته که پوشش این روش‌ها به اندازه static نیست. یعنی False Negative بالایی دارد. (ثبت قبل از اجرای واقعی)

Run-time یعنی سیستم همواره کار دارد function را انجام می‌دهد و دارد کارش را می‌کند،

Monitor ها code اضافی تغییر می‌دهند. enforcement می‌کنند. هم‌زمان با این دسته، Execution Monitor هم داریم که اگر خطا دیدند متوقف می‌کنند. انواع دیگر هم هست.

Interpreter Monitoring، Inline Monitoring، ... هست.

دسته‌ی دیگر، روش Rewriting است. یعنی کد برنامه را تغییر می‌دهیم و سپس اجرا. روش‌های استاتیک هم این‌گونه‌اند. این روش نیست، فرآیند پیچیده‌ای است. روش‌های دینامیک، Run-time، انجام می‌شوند است. به دنبال اجراست، مانند Inline monitoring (یعنی rewriting است). Rewriting در زمان اجرا

Subject: Formal
Date 98/12/18

استاتیک و به اختصار دینامیک. البته Instramental روش Aspect-Oriented هم داریم. اما برای سیستم ها

مفهوم اعمال یعنی چیزی برنامه باید انجام دهد (البته خود rewriting هم نوعی استاتیک و دینامیک) پس بازرسی برنامه

وقتی اعمال را اجرا می کند در زمان اجراست. پس بازرسی برنامه را نوعی روش بازرسی در زمان اجرا و دینامیک

روش های hybrid (ترکیبی) یعنی از قسمت های استاتیک در زمان اجرا استفاده کنیم. اگر روش دینامیک را با اطلاعات کامل

استاتیک داشته باشیم، روش بهتری است.

سوال اینجاست: شما فعلا این روش ها و رشته ها، هرکدام را کجایی؟

اثبات کرده اند (در تئوری) که Monitor می تواند Non-property را اجرا کند.

از این روش های اعمال که مطرح شد، ما رشته Run-time می گذاریم.

Run-time → اصطلاحی است در حوزه زبان های برنامه نویسی که به زمان اجرا برنامه

Run-time Mechanism, verification, enforcement, ...

وقتی در زمان اجرا سیستم می خواهیم حفاظتی از اعمال کنیم، چطور می توانیم این کار را انجام دهیم

Reference monitoring (Execution Monitoring) → مطرح شده توسط ۴۰-۵۰ سال

دیوید سال 2000، کار فرد اشنایدر (Fred B. Schneider) با عنوان Enforceable Security Policies

سوال: چطوری می توان از سیستم اطمینان حاصل کرد برای ما کار کند؟

چه چیزهایی می توان از سیستم خواست؟

Subject: ورودی
Date ۹۵/۲/۲۵

Run-time Verification ... Run-time mechanism دارد ← مفهوم کلی‌تر

Availability ، Information Flow ، Access Control : امنیتی داریم

Special-purpose ، Application-dependent ... Polling

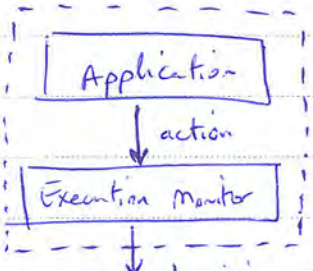
Fine-grain Topology

OS ↔ Reference Monitor (RM) ↔ Application

TCB ... OS ... RM ... mediate

Execution Monitor ... App ...

EM



Transformation ... Yes/No

... enforce ، policy ، EM ، App

Subject: Formal
Date ۱۳۹۵/۲/۲۵

می‌خواهیم Characterize کنیم EM ها را ← یعنی هم policy هایی را که توانایی enforce کردن دارند.

واسطی به EM دارد، که اجرا از برنامه است.

که شامل دنباله‌ای از actionها

ماشورینگا اجرا را می‌بیند، و چیزی نیز از اجراهای آینده خبر دارد و در اجرای در حال App یعنی machinery ی که می‌بیند، فقط اجرای آن را می‌بیند. فرض ماشورینگا این است. حالا تحمین اعمال نیز (که این Traditional EM است).

سه پارامتر داریم: ۱) ماشورچی چی داره؟

۲) ماشورچی چه توانایی برای تغییر اجراها دارد؟

۳) قدرت اعمال چقدر است؟

اعمال نیز Traditional Runtime Monitoring، فقط اجرا را می‌بیند و چیزی را تغییر نمی‌دهد، حالا با این شرایط چی را می‌توان enforce کرد؟

ماشین ماشورینگا عرض کردیم شد، از سه‌تا چیز تشکیل شده، یکی اینکه fact چه اجرایی دارد، این action‌ها (اتفاق‌ها) را می‌بیند.

و می‌داند وقتی!

و این policy ها، Access Control ساده‌ترین است و هم فرم‌های توانایی enforce کردن را دارد، بنابراین بررسی می‌کنیم:

Subject:
Date

Ψ : Universe of all possible finite or infinite sequences.

$\Sigma_S \subseteq \Psi$: The set of execution of S.

Definition of Security Policy:

A security policy is specified by giving a predicate on sets of executions. A target S satisfies security policy P if and only if $P(\Sigma_S)$ equals true.

$$P : \mathcal{P}(\Psi) \longrightarrow \{true, false\}$$

$\mathcal{P}$: powerset

$$\Pi \subseteq \Sigma \wedge P(\Sigma_S) = true \Rightarrow P(\Pi) = true$$

Subject: Formal ۹۵،۲،۲۰ Year: Month: Date:

Execution Monitors:

(Em-enforceable ، منظم) enforce توانند EM ها

چون EM ها فقط می‌توانند اجرای برنامه را مانیتور کنند پس فقط برای خواص منظم کار می‌کند

① $P(\Pi) \Leftrightarrow \forall \sigma \in \Pi . \hat{P}(\sigma)$ به صورت کلی:

↓

روی predicate اجرا

که polices توانند روی چه اجرایی اعمال شوند.

و این polices ها property (خاصیت) را enforce می‌کنند. (که property ها از نوع Information flow)

enforce property توسط EM

Notation $\sigma[..i]$ → prefix of an execution
first i elements

$\tau\sigma$ → یعنی ادامه σ با τ

ادامه σ با τ یک اجرای دیگر است.

$\overline{\Pi}$: The set of all prefixes of executions of Π.

مجموعه prefix های اجراهای موجود در Π

Formal

Subject: Year: Month: Date:

تعریف این است که حداکثر enforce ها را Em

(2) $\forall \tau' \in \Psi^{-} . \neg \hat{P}(\tau') \Rightarrow (\forall \sigma \in \Psi . \neg \hat{P}(\tau' \sigma))$.

(prefix-closed)

ماشین های Remediable را الگوریتم هایی در نظر گرفتیم که بتوانند

این است که ماشین خروجی درست بدهد سپس اجرا را ادامه دهد. اگر حداکثر

الگوریتمی نتواند خروجی درست بدهد، ماشین آن است که حداکثر از دنباله خروجی را چاپ نکند.

و در کل تعریف:

(3) $\forall \sigma \in \Psi . \neg \hat{P}(\sigma) \Rightarrow (\exists i . \neg \hat{P}(\sigma[..i]))$.

یک finite prefix برای اجرا (reject (Em) وجود دارد که

reject، finite، Em

prefix ها terminating این الگوریتم

decidable، finite sequences روی $\hat{P}$: implicit

(4) $\hat{P}$ is decidable on finite sequences.

Formal

Subject: Year: Month: Date:

SOBHAN

Subject: Formal Year: Month: Date:

Security Automata

Non-EM-enforceable Security policies:

IF the set of executions for a security policy P is not a safety property, then an enforcement mechanism form EM does not exist for P.

Soundness and Transparency

Subject: Formal
Date 98/2/25

شرط مقاوم اجرای

Soundness : اجرا های ما در سیستم باید این باشد.

Transparency : توانایی ما جهت تعیین اجرای آن است. شرط مقاوم اجرای مجدد

برون تعبیر به درون مجدد. ← سازگاری برون تعبیر

Traditional EM این دو شرط را

Soundness را در حوزه Safety Property بررسی.

Transparency را هم در چنین محیطی تعریف کنیم این است اجرای هیچ متغیری نباشد

اجرا قطع می‌شود و دنیا درون تعبیر به درون می‌دهد. تا در این دنیا اجرا را طی کرده و برگردد

علی در بازی action ... action transparent در حوزه

in lock-step with input actions

این است معنی همین action ها را هیچ کس و هیچ چیز را نقض نمی‌کند.

پس برای دریافت EM برای اجرای سالم را تعریف کنیم.

حالا سیستم حفاظتی ما سطح اعلی می‌تواند شرط EM ایجاد کند.

Access Control Policies ← درون Safety Property است

Information Flow Policies ← درون Property است. چون Safety Property

PqPCO

(hyper liveness, hyper safety) و ...

→ تا چه حد می‌توان اجرا را تضمین کرد.

۱۲۵

Subject:
Date

transparency ← enforce ، NI ،

(How powerful over enforcement mechanism, Ms. Fallah, Iranmehr)

← Availability policies

Safety

Buchi Automaton :

Definition - A Buchi automaton is a tuple $(\Sigma, Q, Q_0, \delta, F)$

Such that :

Subject: Formal
Date ۹۶/۲/۲۵

- Σ is a finite or countably-infinite set of symbols.

- Q is a finite or countably infinite set of states.

- $Q_0 \subseteq Q$ is a set of initial states.

- $\delta : Q \times \Sigma \to 2^Q$ is a transition relation.

  2^Q : $\mathcal{P}(Q)$ — non-determinism

  possibly partial — or (set-valued function)

- $F \subseteq Q$ is an acceptance set.

$e \in \Sigma^\omega$

e is valid if at least one state in acceptance set has seen (observed) infinitely many times.

Subject:
Date

Security Automata:

- a countable set Q of automaton state.
- a countable set $Q_0 \subseteq Q$ of initial automaton state.
- a countable set I of input symbols, and
- a transition function $\delta : (Q \times I) \rightarrow \mathcal{P}(Q)$ (partial; $\mathcal{P}$: power set)

Q_0 : $s_1, s_2, \ldots$

$Q \xrightarrow{\text{transition}} \bigcup_{q \in Q'} \delta(q, s_i)$ (Q': current state)

Security Automata حالت خاصی از Buchi است که همه حالت‌ها acceptance (states) هستند.

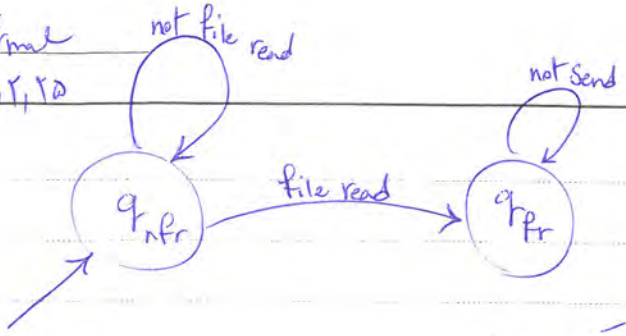
یعنی در polling (یعنی هر transition) همیشه حالت داریم و هیچ‌وقت به حالت خطا نمی‌رسیم. همه‌ی اعمال ممکن را انجام می‌دهیم.

هم به تمامی این موارد دقت داریم. از همین خبر برای ارتباط هم استفاده می‌کنیم. یعنی اگر وارد حالت خطا شدیم، یعنی ورودی حالت آتی به نحوی هیچ‌کدام دیگر از آن‌ها نباید بعد نوشته شود. پس enforcement هم دارد.

مثال) خط‌مشی ورودی در پروتکل Send برای کانال عمومی بعد از خواندن فایل صورت گیرد.

q_{nfr} : no file read $\qquad$ q_{fr} : file read

Subject: Formal
Date ۹۵,۲,۲۵



Predicate ، label ها

Inlined Reference Monitor ←

Safety Property Security Automata

Guarded Command

B → S

guard

Command

true ، guard

Command

state Vars state: {0,1} initial 0

transitions not FileRead ∧ state=0 → Skip

FileRead ∧ state=0 → state:=1

Command

not Send ∧ state=1 → Skip

PdPCO

Subject:
Date

principals

کسی از یک خط می تواند دسترسی به این یک

state vars P : set of PRINS initial ∅

O : set of OBJS initial ∅

A : set of <S: PRINS, O: OBJS, r: RIGHTS> initial ∅

transitions Open(p, o, r) ∧ ⟨p, o, r⟩ ∈ A → Skip

read

AddRight(p, p', r', o') ∧ ⟨p, o', control⟩ ∈ A → A := A ∪ {⟨p', o', r'⟩}

:

bad thing

(BLP)

(Traditional Em)

insert, suppress

terminate

(means)

inter_lock

transparency

Subject: Formal
Date 98/2/27

Survey مقاله Khoury و Tawbi در Computer review 2012 با عنوان

Which security policies are enforceable by runtime monitors?

ابتدا کلیاتی راجع به مانیتورهای امنیتی زمان اجرا گفته شده است:

1- Means put at the disposal of the monitor to react to a possible security policy violation

e.g. - abort the execution of the target program

- Supress a disallowed action and continue the execution.

(Suppress) در صورت اجرا خارج شدن از state یا عوض شدن به یک action یا نداشتن action بعدی در صورت نداشتن action بعدی، action فعلی را نادیده گرفته و اجازه اجرا نمی‌دهد و خطایی هم تولید نمی‌کند.

- insert some action(s) into the control flow.

- insert or supress actions (edit)

2- Information made available to the monitor about the possible executions of the program.

ماهیت این اطلاعات می‌تواند کنترل برنامه باشد (مثل گراف وابستگی برنامه) ← جهت جمع‌آوری اطلاعات کلی

و یا مثل Abstract Interpretation, Type System و ... جهت جمع‌آوری اطلاعات در مورد

کنترل‌های ایستا. (non-uniform)

Subject:
Date

3- Latitude the monitor is given to transform its target's execution.

precise enforcement
effective enforcement
corrective enforcement
} enforcement paradigms

insertion auto. , Supression auto. , Truncation automaton

Subject: Formal
Date

$(q, \sigma) \xrightarrow{\tau} (q', \sigma')$

state — Sequence of actions τ — $(\sigma' \subseteq \sigma)$

(Single-step judgment)

$(q, \sigma) \Longrightarrow (q', \sigma')$ ⟶ multiple step judgment

empty

$(q, \sigma) \xrightarrow{\epsilon} (q, \sigma)$

$(q, \sigma) \xrightarrow{\tau} (q'', \sigma'') \wedge (q'', \sigma'') \xRightarrow{\tau'} (q', \sigma') \Rightarrow$

concat $\tau;\tau'$

$(q, \sigma) \Longrightarrow (q', \sigma')$

Truncation Automaton:

A truncation automaton $\mathcal{A}$ is a tuple (Q, Σ, q_0, δ) where

- Q is a finite or countably infinite set of states.
- Σ is a finite or countably infinite set of atomic actions.
- q_0 is the initial state
- $\delta : (\Sigma \times Q) \longrightarrow Q$ is a (possibly partial) deterministic transition function.

PqPCO

۱۳۴

Subject:
Date

$$\begin{cases} (q,\sigma) \xrightarrow{a} (q',\sigma') & \text{if } \sigma = a;\sigma' \text{ and } \delta(a,q)=q' \\ (q,\sigma) \xrightarrow{\epsilon} (q',\epsilon) & \text{otherwise} \end{cases}$$

Supression Automaton:

A supression automaton $A=(Q,\Sigma,q_0,\delta,\omega)$ where

$\omega : \Sigma \times Q \to \{+,-\}$ is a deterministic function with with the same domain as δ that indicate wether the input action is to be output (+) or supressed (−).

$$\begin{cases} (q,\sigma) \xrightarrow{a} (q',\sigma') & \text{if } \sigma = a;\sigma',\ \delta(a,q)=q',\ \omega(a,q)=+ \\ (q,\sigma) \xrightarrow{\epsilon} (q',\sigma') & \text{if } \sigma = a;\sigma',\ \delta(a,q)=q',\ \omega(a,q)=- \\ (q,\sigma) \xrightarrow{\epsilon} (q',\epsilon) & \text{otherwise} \end{cases}$$

Subject:
Date

Insertion Automaton:

An insertion automaton $A = (Q, \Sigma, q_0, \delta, \gamma)$ where

$\gamma : \Sigma \times Q \longrightarrow \Sigma^* \times Q$ is a function with a disjoint domain from that of δ. This function indicates the finite sequence which must be output for a given input sequence.

$(q, \sigma) \xrightarrow{a} (q', \sigma')$ if $\sigma = a;\sigma'$, $\delta(a, q) = q'$

$(q, \sigma) \xrightarrow{\tau} (q', \sigma')$ if $\sigma = a;\sigma'$, $\gamma(a, q) = (\tau, q')$

$(q, \sigma) \xrightarrow{\epsilon} (q', \epsilon)$ otherwise

Edit Automaton:

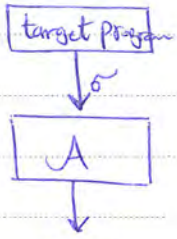
An edit automaton $\mathcal{E}$ is a tuple (Q, Σ, q_0, δ) where

$\delta : (\Sigma \times Q) \longrightarrow (\Sigma^* \times Q)$ is a (possibly partial) function which indicates the output of the automaton when a given action is received as input in a given state.

Subject:
Date

A: one of these automata

A(σ) ← آزمایش



سوال این است، چگونه یک Buchi Automata بسازیم؟

(اثبات سازنده) Constructive Proof

اثبات می‌شود هر Property روی Buchi Automata توسط یک finite sequence، با ثبات پذیر

(معادل‌های Safety) نمی‌توان از چنین چیزی صحبت کرد، زیرا ارائه می‌شود.

یک infinite sequence، چگونه می‌توان اثبات کرد.

این ثبت عدالتی است ماشین در حالت‌های خاص نباید وارد شود و خرابی. اضافه می‌شود action حاوار

ارزیابی است. حد پدیده‌های سیستم این‌گونه ایزنسر. چیزهم اصول است واقعی.

انواع روش enforcement صحبت کنیم، نمی‌کنیم:

Enforcement Paradigms
- Precise Enf.
- Effective Enf.
- Corrective Enf.

- نظریه سیستم‌های برتری در Effective ثبت در precise برای انواع این

نمی‌کنیم اطلاعات اینها، اطلاعات درباره اجراهای ممکن هم مهم است.

Subject: Formal
Date 98,3,1

Uniform Context:

Non-Uniform Context:

Subject:
Date

ثبات‌پذیری و توانایی Edit Automaton برای هر property از finite sequences (over finite sequences).

می‌توانیم اعمال احکام شود. یعنی $A(\sigma)$ با ورودی σ، خروجی قابل قبول تولید کند.

اما محدودیت‌هایی هم وجود دارد:

- قضیه با الزام‌های خروجی باید در مجموعه اجرای valid باشد. (Soundness)

$$\forall \sigma \in \Sigma^{\infty} \cdot \hat{p}(A(\sigma))$$

(S)

Non-Uniform برای

- تفاوت بین الزام‌ها و Transparency

Definition - Let Σ be a set of atomic actions and $S \subseteq \Sigma^{\infty}$ be a subset of sequences. An automaton $A = (Q, \Sigma, q_0, \delta)$ precisely enforces a property $\hat{p}$ iff $\forall \sigma \in S$

1) $(\sigma, q_0) \overset{A}{\Rightarrow} (\epsilon, q')$ → transition

2) $\hat{p}(\sigma')$ → Soundness

3) $\hat{p}(\sigma) \Rightarrow \forall i \in \mathbb{N} . i \leq |\sigma| \rightarrow \exists q'' \in Q . (\sigma, q_0) \overset{\sigma[0..i]}{\Rightarrow} (\sigma[i+1..], q'')$

in lockstep with input actions, repeat the actions.

PAPCO

اولین action ورودی: اگر درست باشد، خروجی مورد نظر را برگردان. یعنی برای اینکه بتوانیم از $i+1$ به i برسیم، باید عملی انجام شود.

Subject: Formal
Date 95,3,1

سخت‌ترین معیار، Precise بودن است که شرط سوم را هم دارد. Transparency

اجرا‌هایی که از اول معتبر بودند ولی در ادامه با action های خود شرط را نقض می‌کنند، Soundness را ندارند.

Edit سلطان‌پور را اجرای عزیز.

Definition – An automaton A effectively $\cong$ enforces a property $\hat{p}$

iff $\forall \sigma \in S$.

1) $(\sigma, q_0) \overset{\sigma'}{\Longrightarrow}_A (\epsilon, q')$

2) $\hat{p}(\sigma')$

3) $\hat{p}(\sigma) \rightarrow A(\sigma) \cong \sigma$ (رابطه هم‌ارزی $\cong$ is an equivalence relation)

اگر به جای effectively بنویسیم precisely، شرط سوم همیشه برقرار است.

شرط سوم همیشه در اجرا Valid است، ولی ممکن است معادل اجرای دیگری نباشد. تعریف معادل بودن دست‌یابی.

معروف‌ترین معادل بودن رابطه هم‌ارزی، ساده‌ترین آن است که از Precise بودن ارث‌بری می‌کند.

چون ساده‌ترین حالت action‌ها هم‌ارز است، مدل اجرا هم‌ارز است. در مدل ساده هم ممکن است یک کد ...

شرطی که برای رابطه معادل بودن داشتیم:

$\forall \sigma, \sigma' \in \Sigma^{\infty}. \ \sigma \cong \sigma' \rightarrow (\hat{p}(\sigma) \leftrightarrow \hat{p}(\sigma'))$

PdPCO

از نقیض شرط را بگوییم، تا زمان بگذرد که بتوانیم رابطه هم‌ارزی که شرط سوم را ندارد، بسازیم.

Subject:
Date

بیشتر روی کدهایی که برای effective enforcement هست تمرکز داریم.

بین تمام کلاس‌های مختلف روی سه تا تمرکز داریم:

- Truncation, Suppression, Insertion, Edit
- Precise, Effective
- Non-Uniform Context, Uniform Context

پس یه policy ضعیف‌تر (یعنی توجیح‌تر به property ای که میخوایم) داریم:

Truncation + Precise + Uniform Context

- Truncation: ضعیف‌تر از
- Precise: کمتر از غیر
- Uniform Context: تغییر نمون اضح

است Safety property ای که

Safety ای که ضعیف‌تر از Truncation + Precise + Non-Uniform می‌تونه

اگر یه policy ای توانایی Violate کردن S رو داره، می‌تونه با action ای غلط کنه

اضافه شدن Liveness هم ویژگی‌ای هست

قدرت policy‌ها:

Trunc. + Precise + Uniform < Trunc. + Precise + Non-Uniform < Trunc. + Effective + Non-Uniform

(Safety) (Safety + Some of Liveness) (more than before!)

Subject: Forward
Date 98,4,1

Suppression + Precise + Uniform = Truncation + Precise + Uniform
Insertion
Edit

Lock step و Precise و Transparency

Edit ... Precise

Suppression + Precise + NonUniform > Truncation + Precise + NonUniform
Insertion
Edit

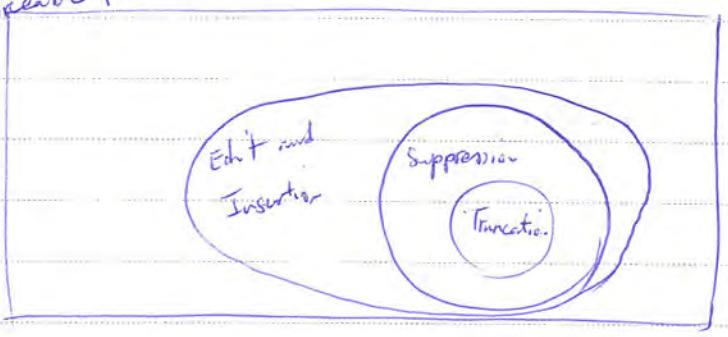
Insertion + Precise + NonUniform = Suppression + Precise + NonUniform

Edit + Precise + NonUniform ≥ Insertion + Precise + NonUniform
Suppression

Edit + Precise + NonUniform = Insertion + Precise + NonUniform

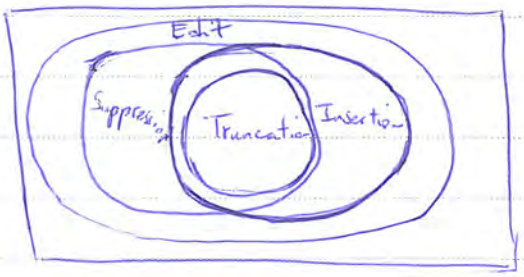
Precise + NonUniform :

Enforceable Policies



Subject:
Date

1) Ligatti, FCS, More enforceable Security Policies, 2002
Walker
Bauer
2) Edit Automata : Enforcement ... , 2005
3) Enforcing Non-Safety Policies with Program Monitors, 2009, 2010. ... Effective + Non-Uniform



Corrective Enforcement :

Corrective ... Soundness ... Effective , precise

3) $A(\sigma) \sqsubseteq \sigma$

Corrective ... Effective , precise

Characterization - (۱

Property (۲

SME (۳

Policies = Property + Hyperproperty

Hyperproperty (Policies) = Property + Non-property

Subject: Formal
Date

(جلسه اول)

Formal Verification of Cryptographic Protocols:

Security Services ⟵ e.g. Confidentiality, Authentication, Authorization, ...

Security Mechanisms ⟵ e.g. public-key, ...

Security Mechanisms ابزاری است برای Security Services

primitive های crypto استفاده می‌شوند، پروتکل برای مقابله با ... طراحی می‌شود.

در ابتدا این پروتکل‌ها ساده هستند اما به مرور به صورت لایه‌ای روی هم ساخته می‌شوند و پیچیده می‌شوند.

پروتکل‌ها دارای باگ هستند. این باگ‌ها می‌توانند امنیتی، غیرامنیتی یا ترکیبی از این دو باشند.

از روش‌های مختلفی برای تحلیل پروتکل‌ها استفاده می‌شود.

Provable Security ← در این روش پروتکل‌ها را به صورت low-level مدل می‌کنند.

از مفاهیمی مانند Reduction، Complexity Theory استفاده می‌شود. مثلاً Kerberos و SSL

هدف اثبات این است که اگر مهاجم بتواند مسئله سختی را حل کند، آنگاه می‌توان نتیجه گرفت که پروتکل امن است.

به اثبات این است. اثبات امنیت پروتکل‌ها معمولاً بر اساس فرضیات امنیتی انجام می‌شود.

این روش‌ها را Computational و Symbolic می‌نامند.

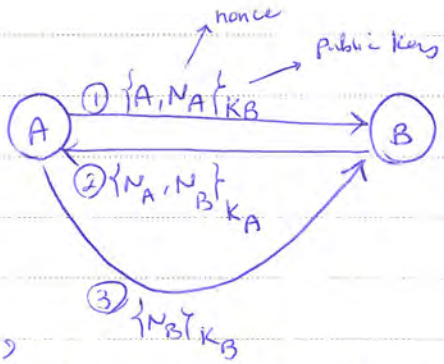
Provable Security این روش‌ها را در جامعه Cryptography Community می‌نامند.

Automated Reasoning (روش‌های خودکار) Symbolic را در جامعه Cryptography Community می‌نامند.

Subject:
Date

فرض در این حالت این است که Crypto System ما perfect باشد. یعنی سیستم‌های رمز
برایج هستند و... ایده‌آل و درست است. حتی وقتی این حرکت می‌کند هیچ‌گونه اتفاقی
هست نمی‌افتد.

Example - Needham-Schroder Key Exchange

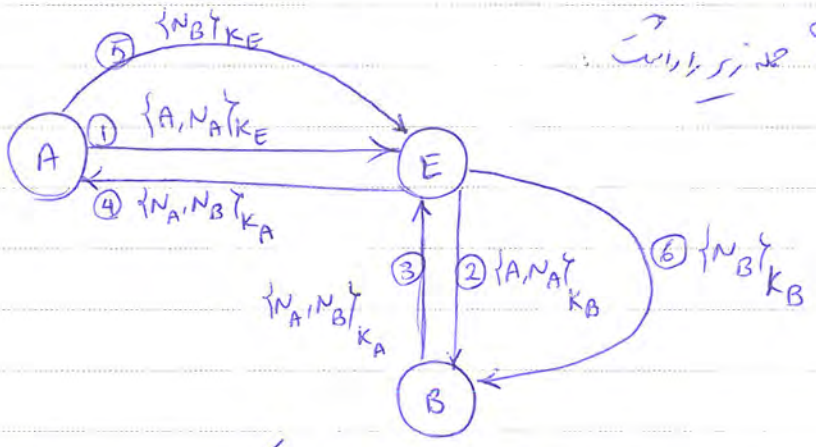


که Session key برای ارتباط بین A و B خواهد بود. N_A , N_B

Principal: طرف درگیر پروتکل

Exchange در این پروتکل Session Key , Mutual Authentication

اما می‌توان حمله‌ای را انجام داد:



حتی اگر این حمله انجام شود Mutual Auth. نقض شده و این که Session key مهم است.

در این حالت B فکر می‌کند با A صحبت می‌کند. این حالت Interleaving است.

E یک internal attacker (Evil) است.

Subject: Formal
Date
۹۵،۳،۸

هی E یه ابزار کاربردی است اما با هارتن خودش روش A و تئوری را ایجاد کرده است.

Prodent Engineering Practice in Cryptography, مقاله Abadi and Needham

که پس از مطالعه کردن این مقاله، اصولی را برای طراحی پروتکل ها ارائه کردند.

در حمله سال بعد، این پروتکل اصلی Needham - Schroeder ، در مرحله دوم، B هم وارد بازی شده و فرا میرود، که خواصه خود. همین مسائل باعث از سال های گروه ۱۰ مورد این سناریو می شود ۱۵ برای

فرمال متد ها راه حل اولیه است. مثال های دیگر: SSL V2 ، Kerberos ، ...

برای کاربری پروتکل های امنیتی:

۱) سیستم مدل شود ← Specification یک مدل مورد نظر که باید رعایت transaction شود

زبان modeling language: یک زبان برای توصیف و ایجاد یک مدل است که باید

بتواند سیستم را به زبان مورد استفاده در مدل کاری قطعی برای توصیف پروتکل تبدیل کند و مدل باید

security rules و Communicating state machine های logical هم مدل شود

تصور شود از زبان های تئوری مثل Applied π: CSP ، Security π-calculus (Spi) ، π-calculus (Pi)

این زبان ها به علت توصیف پروتکل ارائه شده. Concurrency modeling is core

۱۴۵

Subject:
Date

بیشتر به خود مدل پروتکل توجه داشتند و البته دیدگاه‌های اخیر به بازه‌سازی پروتکل نیز اهمیت زیادی داده‌اند.

۲) مدل هاجم ← توانایی‌های attacker چیست؟

می‌تواند پیام‌ها را intercept کند، پیام بسازد، و از داده‌ای که در پیام‌ها هست چه کاری

و چه دسترسی دارد.

وقتی سیستم و مدل هاجم را می‌دانیم باید در نظر بگیریم که اول کدام هاجم

هم کدام از سیستم حرکت می‌کند. هاجم یک process است که ظرفیتی دارد و پروتکل اجرا

می‌شود. سخت‌ترین پروتکل یعنی هاجم می‌تواند پیام‌هایی با ظرف‌های پروتکل‌ها بسازد

باورهای (beliefها) است که می‌خواهد ایجاد کند.

ما خواسته امنیتی مورد نظر برای پروتکل، خود سوال سختی است. ظریف و دقیق باید بیان شود اما

چه خواسته‌ای داریم؟ باید طوری باشد که بتوان Semantics پروتکل را روی آن اجرا کرد؟ پس

باید Security Requirement را formalize (یعنی Semantic Property) کنیم.

۳) حالا مشخص شد سیستم چیست، و چه خواسته مورد نظر را داریم می‌خواهیم

از سروده‌های این حوزه، خانم Katrin Medows است که Survey دارد.

Subject: Formal
Date
۹۸/۳/۸

Methods based on State Machines

Dolev and Yao

On the Security of public key protocols, 1983
IEEE

Intruder:

- read all traffic
- alter or destroy messages
- create messages
- perform any operation, such as encryption, that is available to the legitimate users of the system

Word Problem : N Word Process

Dolev-Yao :

Knowledge state : observation

PqPCO

Subject:
Date

1) System Model → Communicating State machine
2) Attacker Model → Dolev-Yao, state machine
3) Security Requirement → Bad Final State
4) Analysis → Exhaustive Search

(۱) ابتدا باید سه چیز را مشخص کنیم: attacker، سیستم، مدل

سپس با ابزارهایی که بر این اساس ایجاد شده‌اند، مانند Interrogator که از روی Dolev-Yao ساخته شده، برای Confidentiality، Secrecy و ... مناسب است.

استفاده می‌کنیم. این ابزار نرم‌افزاری همه حالت‌های ممکن را ایجاد می‌کند و به صورت Exhaustive Search

در این ابزار، سیستم به صورت مدل Communicating State Machine، یعنی principal ها، ارتباط‌شان مدل می‌شود.

مدل attacker هم مطابق مدل Dolev-Yao

حالت نهایی برای جستجو در نظر گرفته می‌شود که آن حالت‌ها را word می‌نامیم.

Nonce

تعریف حالت بد

سپس Interrogator می‌رود به بقیه حالت‌ها سر می‌زند و آن حالت‌ها را بررسی می‌کند.

این ابزار، حمله‌ای پیدا نکرد اما حالت ساده‌شده را ارائه کرد.

سؤال مرتبط: این ابزارها، آیا حمله‌ای در پروتکل‌ها پیدا کرده‌اند؟

از ابزارهای دیگر می‌توان به NRL Protocol Analyzer اشاره کرد.

مثل مدل Dolev-Yao، Interrogator هم نقض‌ها را بررسی می‌کند.

ورودش برای یافتن حالت insecure state است.

چون Interrogator، گرددی روی State Space آن، پیاده‌سازی شده است.

پس این ابزار عملاً یک interrogator محسوب می‌شود.

PAPCO

Subject: Formal Verification of Cryptographic Protocols: A Survey
Date 90/12/18

Medews

unreachable ... final state ... interrogator ...

infinite loop ... backward ... state ...

initial state ...

Proof Assistant ... prover ...

Theorem Proving ... Model checking ... NRL ...

Syther ... model checker ...

Logic ...

Modal Logics ... Computer scientist ...

(منطق‌های وجهی)

reasoning ... logic ...

e.g. I know that it is cold today. → Propositional ...

I believe that ... → First order ...

know ... believe ... mode (وجه) ...

Epistomic Logic

Belief logic

Belief logic ...

PqPCO

Subject:
Date 90/3/10

ایده اصلی این است که در هنگام اجرای یک پروتکل، با ارسال و دریافت پیام ها، طرف های درگیر در پروتکل از اعتبار فرضیات اولیه، باورهای خود را نسبت به باورهای طرف دیگر به روز می کنند. پس می توان با مدل کردن باورهای درگیر در پروتکل، اعتبار و درستی پروتکل را بررسی کرد.

با حواله اصلی اثبات شود.

لایبنیتز می گوید خدا از میان حوالت های ممکن، بهترین را انتخاب کرده است.

Possible World

I know that
I believe that

BAN: deduction

(Burrows, Abadi, Needham) → A logic of authentication

Prover: First order

Prolog

Theorem Prover, Resolution

Subject: Formal
Date ۹۵/۱۲/۱۵

ابزارهای بررسی صحت پروتکل‌ها: ProVerif، Proof Search، State Space و ... که از راه‌های مختلف (مثلاً ...) به بررسی پروتکل‌ها می‌پردازند.

Applied-π: حوزه‌ی کاری بسیار گسترده‌ای دارد و ...

BAN Logic:

اضافه شد: belief logic از دانشمندان و ... که ...

- What does this protocol achieve?
- ~~Does~~ Does this protocol need more assumptions than another one?
- Does this protocol do anything unnecessary that could be left out without weakening it?

:

اضافه شد: Authentication از ... که ...

دلیل شکست سیستم بررسی صحت این است که ... اثبات نمی‌کند که ...

... از پروتکل‌های مختلف ... که پروتکل‌های ... را بررسی می‌کند.

ایده‌های اولیه برای پیچیده‌سازی پروتکل‌ها: ... Joe ... هر کدام از ... را ...

... Joe ... که ...

Subject:
Date

- اگر بخواهیم بگوییم Joe ، K را فرستاده است باید بگوییم K توسط Joe فرستاده شده است (نه برعکس)

- اگر جمله عوض شود و بگوییم Joe originally ، Joe را فرستاد، یعنی Joe قبلاً این پیام را فرستاده است و حالا دارد replay می‌کند.

:

Formalization برای fact های بالا، belief های بالا را تعریف می‌کنیم.

Basic Notation:

یک زبان منطقی Many-Sorted model است.

Principals, encryption keys, formulas (statements) ⟵ عناصر موجود در این زبان

A, B, S : Specific Principals

K_{ab}, K_{as}, K_{bs} : Specific Shared keys

K_a, K_b, K_s : Specific public keys

$K_a^{-1}, K_b^{-1}, K_s^{-1}$: Corresponding Private keys

N_a, N_b, N_c : Specific statements

PqPCO

statement های خاص را می‌نامیم

Subject: Formal

Date 95/12/10

P, Q, R : Symbols range over principals

X, Y : " " " Statements

K : " " " encryption keys.

Conjunctive

(AND) Conjunction : Propositional بین

عناصر اصلی برای ترکیب های مختلف زیر مهم قرار دارد:

$P \equiv X$: P believes X

برای فرض P صحت عبارت X را باور دارد. Semantics عنوان BAN

$P \triangleleft X$: P sees X

P به نحوی پیام حاوی X را دریافت کرده است.

$P \sim X$: P once said X

P مدتی X را گفته است. P به نحوی پیام را ارسال کرده که حاوی X بوده.

$P \Rightarrow X$: P has jurisdiction over X

P حق اداره X را دارد است.

P is an authority on X and is trusted.

(Certification Authority)

Subject:
Date

$\#(X)$: The formula X is fresh

X در یکی از اجزا فعلی پروتکل ارسال نشده - اولین بار در آن اجرا ارسال می‌شود.

$P \xleftrightarrow{K} Q$: P and Q may use the shared key K to communicate.

K کلید متقارن است و هر یک از P و Q به آن دسترسی دارند.

$\xmapsto{K} P$: P has a public key.

$P \overset{X}{\rightleftharpoons} Q$: The formula X is a secret known only to P and Q.

$\{X\}_K$: The formula X encrypted under the key K.

($\{X\}_K$ from P : فرستاده شده از)

$\langle X \rangle_Y$: X combined with the formula Y.

که در آن X غیر از Y است Secret

دوره (یا زمان) نازل : نسخه دوم (rnn)

اجرای فعلی از پروتکل

هر کدام از اجرای فعلی

Subject: Formal
Date
۹۵/۳/۱۰

(proof system)

Postulates:

message reasoning:

$$\frac{P \ni Q \overset{K}{\leftrightarrow} P \quad P \lhd \{X\}_K}{P \ni Q \sim X}$$

$$\frac{P \ni \overset{K}{\mapsto} Q \quad P \lhd \{X\}_{K^{-1}}}{P \ni Q \sim X}$$

$$\frac{P \ni Q \overset{Y}{\rightleftharpoons} P \quad P \lhd \langle X \rangle_Y}{P \ni Q \sim X}$$

nonce verification:

$$\frac{P \ni \#(X) \quad P \ni Q \sim X}{P \ni Q \ni X}$$

(هر وقت هر پیامی ارسال کرد، باید آن را بداند)

Jurisdiction:

$$\frac{P \ni Q \Rightarrow X \quad P \ni Q \ni X}{P \ni X}$$

Component statements:

$$\frac{P \ni X \quad P \ni Y}{P \ni (X,Y)} \qquad \frac{P \ni (X,Y)}{P \ni X} \qquad \frac{P \ni Q \ni (X,Y)}{P \ni Q \ni X}$$

Subject:
Date

$$\frac{P \Vdash Q \sim (X,Y)}{P \Vdash Q \vdash X} \qquad \frac{P \lhd (X,Y)}{P \lhd X} \qquad \frac{P \lhd \langle X \rangle_Y}{P \lhd X}$$

$$\frac{P \Vdash Q \overset{K}{\leftrightarrow} P \quad P \lhd \{X\}_K}{P \lhd X} \qquad \frac{P \Vdash \overset{K}{\mapsto} P \quad P \lhd \{X\}_K}{P \lhd X}$$

$$\frac{P \Vdash \overset{K}{\mapsto} Q \quad P \lhd \{X\}_K^{-1}}{P \lhd X} \qquad \frac{P \Vdash \#(X)}{P \Vdash \#(X,Y)}$$

$$\frac{P \Vdash R \overset{K}{\leftrightarrow} R'}{P \Vdash R' \overset{K}{\leftrightarrow} R} \qquad \frac{P \Vdash Q \Vdash R \overset{K}{\leftrightarrow} R'}{P \Vdash Q \Vdash R' \overset{K}{\leftrightarrow} R}$$

$\vdots$

A formula X is provable in the logic from a formula Y if there is a sequence $Z_0, \ldots, Z_n$ where $Z_0 = Y$, $Z_n = X$, and each Z_{i+1} can be obtained from previous ones by the application of a rule.

Subject :

Date

Subject: Formal Methods
Year. ۹۴ Month. ۲ Date. ۱ (۱)

Protocols Analysis:

Distributed programs (System)

Secrecy, Authoring

* Security Assurance

Assurance ... formal ; Proof

Cathrin Meadows

Cryptosystem:

Cryptographic primitives

... , Encryption , Hash

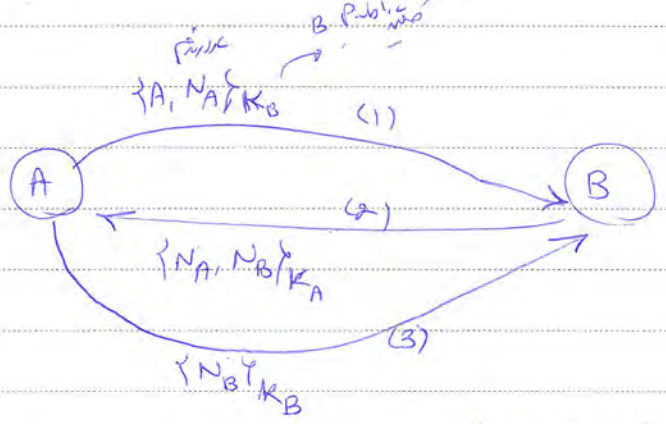
Abadi

Symbolic

Provable Security

Blanchet

Mr. Philipe Rogaway

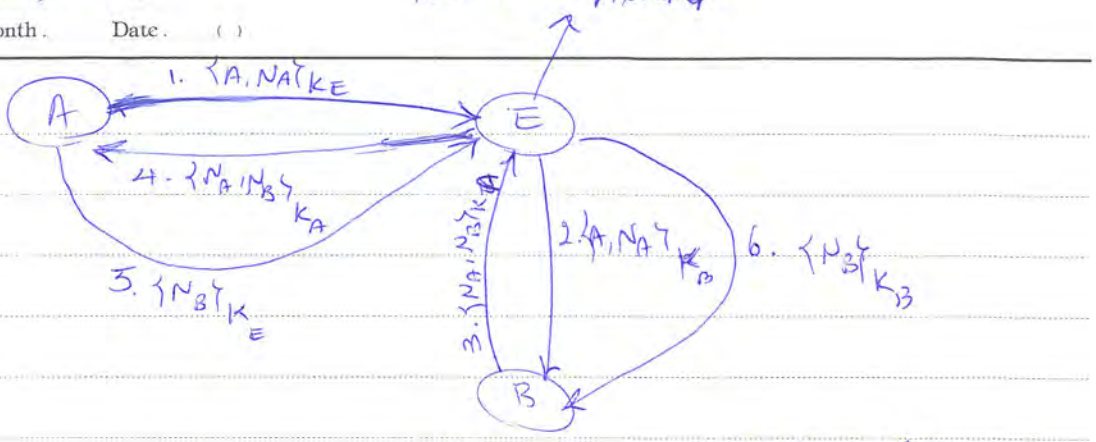


Session key

Session key ... N_B , N_A ... Mutual Authentication

Subject: Private, Public

Year. Month. Date. ( )



آلیس B مقصودش A صحبت میکند. دشمن E صحبت میکند. تصدیق واقعیت قوی است.

ایده (Symmetric Key Session Key) SK تولید N_B ، N_A توسط E

آلیس B در یک کامپیوتر شبکه است و مسیل پوتل میکند.

1) Methods based on state machines روش‌های بر پایه ماشین‌های حالت

2) Systems based on modal logic سیستم‌های بر پایه منطق وجهی (مودال)

3) Using Algebra to reason about knowledge

حوزه اول

از اینجا کار می‌کنیم Dolev and Yao

The network is assumed to be under the control of an intruder who can read all traffic, alter and destroy messages, create messages, perform any operation such as encryption that is available to legitimate users of the system.

insider

Subject: Formal
Year. 95 Month. ۲ Date. ۱۶

The System becomes a machine used by the intruder to generate words.

generating words obeys certain rules.

e.g.:

$$K \underset{E}{} K^{-1}_{E}(M) = M$$

Thus, we can think of the intruder as manipulating a term-rewriting systems. If the goal of the intruder is to find out a word that is meant to be secret, then the problem of proving a protocol secure becomes a word-problem in a term-rewriting system.

Subject:
Year. 98 Month. ۲ Date. 4 ( )

Bounded run

model checking

Polev-Yao ... Intruder ... NRL ... (Meadows)

unlimited round of protocol runs

(infinite state)

Liveness and Safety:

Σ: action

Σ^*: ...

Σ^ω: ...

$\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$

$M \subseteq \Sigma^\infty$

$P \subseteq \Sigma^\infty$

property → ...

$\sigma \in \Sigma^\infty$

$a_0 a_1 a_2 \ldots$

$\tau \in \Sigma^*, \tau \leq \sigma$

prefix

$\tau = a_0 a_1 a_2 \ldots a_n$

M satisfies P iff $M \subseteq P$

$P(\sigma) \Rightarrow \sigma \in P$

Safety properties

$\neg P(\sigma) \Rightarrow \exists \tau \leq \sigma . \left(\neg P(\tau) \wedge \forall \sigma' . \tau \leq \sigma' \rightarrow \neg P(\sigma') \right)$

remediable

liveness property

$\forall \tau \in \Sigma^* . \exists \sigma \in \Sigma^\infty . \tau \leq \sigma \wedge P(\sigma)$

Valid ... remediable

Subject: Formal Verification of Cryptographic Protocols: A survey → Meadows

Year. 94 Month. 07 Date. 4 ( )

روش‌های رسمی برای تحلیل پروتکل‌های رمزنگاری

روش‌های رسمی برای تحلیل پروتکل‌ها Modal logic است.

propositional logic → Modal logic (Intentional logic)

↳ Epistemic logic → Knowledge

↳ Temporal logic

IF I believed I've received a message encrypted with key K, and I believe that only Alice and I know K, then I believe that the message was originated by either Alice or me.

روش‌های رسمی برای تحلیل پروتکل‌ها Algebra است.

هدف این است که برای Participant, Intruder و ... یک مدل ارائه دهیم.

بعد از مدل‌سازی باید خواص امنیتی را بررسی کنیم. از خواص امنیتی می‌توان به Participant اشاره کرد.

تعریف requirement به چه معنی است؟

Tutorial Kremmer

BAN logic

Subject:
Year. ۹۴ Month. ۲ Date. ۸ ( )

BAN Logic:

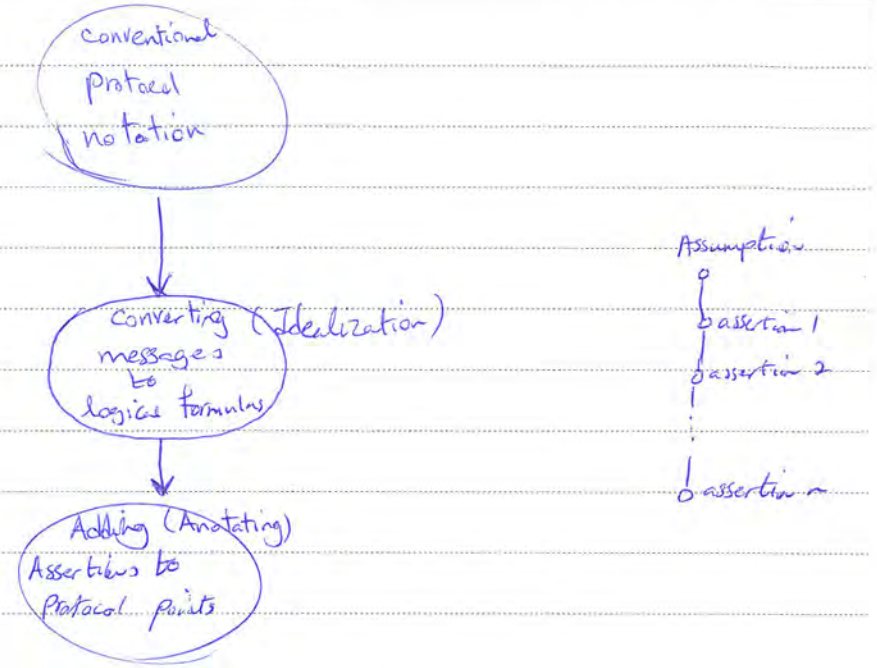
A logic of authentication

AS: Authentication Server
CA: Certification Authority

Freshness

Subject:
Year٢٠٠٤ Month.١ Date.٨ ( )

Formulation:



Notation:

many-sorted modal logic

↓

object الكائن

↳ statements, principals, keys

(formula)

A, B, S, ... (specific)

$K_{AB}, K_{AS}, \ldots$ (Shared keys)

$K_A, K_B, \ldots$ (public keys)

$K_A^{-1}, K_B^{-1}, \ldots$ (private keys)

X, Y, ... (meta-variable)

P, Q, ... (Range) (Variable)

$N_a, N_b, \ldots$ (Specific Statement) (nonces)

nonce: number used once

X, Y, Z ≡ X and Y and Z (Juncion) Conjunction

Subject:
Year. ۹۶ Month. ۷ Date. ۸ ( )

Constructs:

$P \models X$: P believes X

principal P می‌تواند X را بداند

$P \triangleleft X$: P sees X

P، X را می‌بیند.

می‌تواند X را برای P فرستاده باشد.

$P \sim X$: P once said X

P زمانی X را گفته است.

این یعنی در زمانی که پیام را فرستاده، اعتقاد داشته

(P believed X when he sent the message)

$P \Rightarrow X$: P has jurisdictions over X

(P is an authority on X)

P در این رابطه trusted است. نمونه‌ای که CA در آن کاربرد دارد.

$\#(X)$: The formula X is fresh.

X در اجرای فعلی پروتکل تولید شده است یا ارسال شده است.

زمان را در epoch (دوره) های مختلف تقسیم می‌کند.

اجرای فعلی ← اجرای قبلی

$P \overset{K}{\leftrightarrow} Q$: P and Q may use the shared key K to communicate

کلید K کلید مشترکی است که P و Q می‌توانند از آن برای ارتباط استفاده کنند.

$\overset{K}{\mapsto} P$: P has K as a public key (K⁻¹ is the matching secret key)

Subject:

Year. Month. Date. ( )

$P \overset{X}{\rightleftharpoons} Q$: The formula X is a secret known only to P, Q

$\{X\}_K$: The formula X encrypted under the key K.

$\{X\}_K$ From P

$\langle X \rangle_Y$: X combined with the formula Y

Concatenate

Logical Postulates:

- message meaning

$$\frac{P \mid\equiv Q \overset{K}{\leftrightarrow} P, \quad P \triangleleft \{X\}_K}{P \mid\equiv Q \mid\sim X}$$

implicitly from R, $R \neq P$

$$\frac{P \mid\equiv \overset{K}{\mapsto} Q, \quad P \triangleleft \{X\}_{K^{-1}}}{P \mid\equiv Q \mid\sim X}$$

$$\frac{P \mid\equiv Q \overset{K}{\rightleftharpoons} P \quad P \triangleleft \langle X \rangle_Y^S}{P \mid\equiv Q \mid\sim X}$$

Subject: Formal
Year. ۹۴ Month. ۲ Date. ۸ ( )

Formal Semantics برای BAN

- nonce verification

$$\frac{P \equiv \#(X),\ P \equiv Q \sim X}{P \equiv Q \equiv X}$$

- Jurisdiction

$$\frac{P \equiv Q \Rightarrow X,\ P \equiv Q \equiv X}{P \equiv X}$$

$$\frac{P \equiv X,\ P \equiv Y}{P \equiv (X,Y)} \qquad \frac{P \equiv (X,Y)}{P \equiv X} \qquad \frac{P \equiv Q \equiv (X,Y)}{P \equiv Q \equiv X}$$

$$\frac{P \equiv Q \sim (X,Y)}{P \equiv Q \sim X} \qquad \frac{P \equiv Q \sim X \quad P \equiv Q \sim Y}{P \equiv Q \sim (X,Y)}$$

این دو قاعده برای اجرای حمله‌ی جاسازی کاربرد دارند.

$$\frac{P \triangleleft (X,Y)}{P \triangleleft X} \qquad \frac{P \triangleleft \langle X \rangle_Y}{P \triangleleft X} \qquad \frac{P \equiv Q \overset{K}{\leftrightarrow} P,\ P \triangleleft \{X\}_K}{P \triangleleft X}$$

$$\frac{P \equiv \overset{K}{\mapsto} P,\ P \triangleleft \{X\}_K}{P \triangleleft X} \qquad \frac{P \equiv \overset{K}{\mapsto} Q,\ P \triangleleft \{X\}_{K^{-1}}}{P \triangleleft X}$$

$$\frac{P \equiv \#(X)}{P \equiv \#(X,Y)} \qquad \frac{P \equiv R \overset{K}{\leftrightarrow} R'}{P \equiv R' \overset{K}{\leftrightarrow} R}$$

Subject: Formal Methods to Security
Year. ۹۴ Month. ۲ Date. ۱۳ /5/

$$\frac{P \models Q \models R \overset{K}{\leftrightarrow} R'}{P \models Q \models R' \overset{K}{\leftrightarrow} R}$$

این قواعد حتی دارند belief را هم.

توضیحات
Delegation States:

principal A may let the server S generate an arbitrary shared key for A and B.

$$A \models S \Rightarrow A \overset{K}{\leftrightarrow} B$$

(۱ یا K) A دارد، A و B است، S به A می‌گوید K مشترک

$$A \models \forall K. \left(S \Rightarrow A \overset{K}{\leftrightarrow} B\right)$$

∀ for all یا quantifier

$$A \models \forall K. \left(S \Rightarrow B \Rightarrow A \overset{K}{\leftrightarrow} B\right)$$ ← با delegate

$$A \models S \Rightarrow \forall K. \left(B \Rightarrow A \overset{K}{\leftrightarrow} B\right)$$ ← این دو با هم تفاوت دارند

S حتی اگر بعداً هم B را بیاورد (بیاید) ... (این دو با هم تفاوت دارند)

S حتی اگر بعداً هم B را بیاورد (بیاید) ... (این دو با هم تفاوت دارند)

فرض بر این است که quantifier ها را بلدید.

Subject:
Year. Month. Date. ( )

Idealized Protocols:

مقدمه اصلی BAN و معرفی پروتکل‌ها

$P \rightarrow Q : message$ نحوه نوشتن پروتکل:

↓ (تبدیل به)

Symbolic bit string

↓

اضافه کردن فرضیات و Statements مورد نیاز

سپس باید با استفاده از قوانین منطقی اثبات شود. (شکل ایده‌آل)

$A \rightarrow B : \{A, K_{ab}\}_{K_{bs}}$

↓

کلید مشترک B و Server که A دارد

$A \rightarrow B : \{A \overset{K_{ab}}{\longleftrightarrow} B\}_{K_{bs}}$ استفاده می‌کنیم:

این به معنی اینه که

$B \triangleleft \{A \overset{K_{ab}}{\longleftrightarrow} B\}_{K_{bs}}$

سپس با استفاده از قوانین منطقی principal belief تغییر می‌کند.

سپس با استفاده از belief ها plain text را به دست می‌آوریم.

hint به معنی

$\{X_1\}_{K_1}, \dots, \{X_n\}_{K_n}$ رشته پیام

crypted یا رمزنگاری شده

frame

Subject:
Year 99 Month 7 Date 5

اول مسئله این پروتکل و مدل سازی intention آن است که چه ارتباطی بین فرستنده و گیرنده برقرار می‌شود.

این تست‌ها کمک می‌کند که زبان به اندازه کافی قوی باشد تا بتوان خواص را بیان کرد.

Protocol Analysis:

- The idealized protocol is derived from the original one.
  bit string form
- Assumptions about the initial state are written.
  قبل از اجرای پروتکل
- logical formulas are attached to the statements of the protocol, as assertions about the state of the system after each statement.
- The logical postulates applied to the assumptions and assertions in order to discover the beliefs held by the parties in the protocol.

پروتکل را annotate می‌کنیم

$$[\text{assumptions}]\ S_1\ [\text{assertion}_1]\ S_2\ [\text{assertion}_2]\ \ldots\ [\text{assertion}_{n-1}]\ S_n\ [\text{assertion}_n]$$

بعد از هر گام

$$[Y]\ (P \rightarrow Q : X)\ [Y, Q \triangleleft X]$$

assertion ها با postulate ها ثابت می‌شوند.

Subject: Formal Approaches
Year 1394 Month 02 Date 13 5

If X is an assertion (but not the assumptions) in a legal annotation A,
if X' is provable from X, and if A' is the result of substituting X' for X
in A, then A' is a legal annotation.

Concurrent و Sequential ، BAN ... explicit ...

Freshness ... timestamp ...

The Goals of Authentication:

Authentication ... BAN ... Authentication ...

Secrecy ... Knowledge ... attacker ...

Authentication ...

A E A $\xleftrightarrow{K}$ B

B E A $\xleftrightarrow{K}$ B

A E $\xrightarrow{K}$ B

Subject.
Year. ۹۴ Month. ۲ Date. ۱۵،۲

(تکراری)

1987

The Outaway_Rees Protocol:-

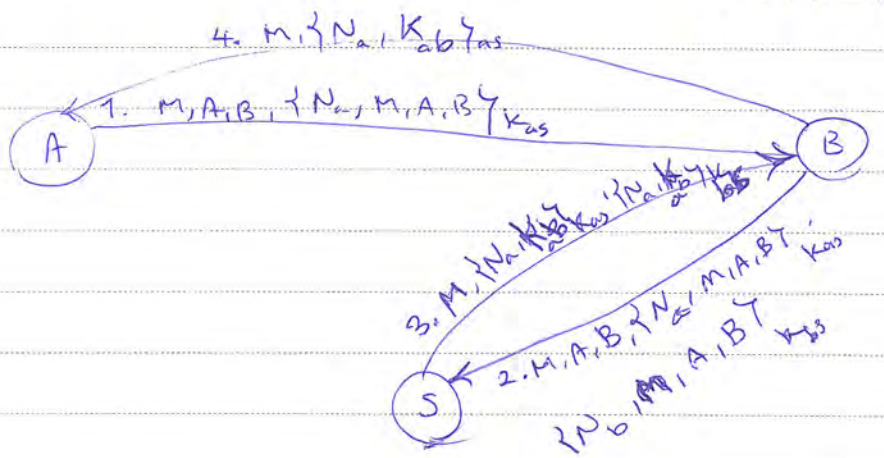
A shared key authentication protocol

$A, B, S \longrightarrow$ Agent

$K_{as}, K_{bs} \longrightarrow$ Shared key (private) (secret)

N_a, N_b, M ... session ... Server

$A \rightarrow B : M, A, B, \{N_a, M, A, B\}_{K_{as}}$



identification:

1. $A \rightarrow B : \{N_a, N_c\}_{K_{as}}$

2. $B \rightarrow S : \{N_a, N_c\}_{K_{as}}, \{N_b, N_c\}_{K_{bs}}$

3. $S \rightarrow B : \{N_a, (A \overset{K_{ab}}{\longleftrightarrow} B), (B \sim N_c)\}_{K_{as}}$,
$\{N_b, (A \overset{K_{ab}}{\longleftrightarrow} B), (A \sim N_c)\}_{K_{bs}}$

hint: ...

Subject: Final
Year 98 Month 7 Date 15

4. $B \to A : \{N_a, (A \overset{K_{ab}}{\leftrightarrow} B), (B \sim N_c)\}_{K_{as}}$

Assumptions: توسعه ایده آل کلیدهای متقارن و فرض اینکه کلید تکمیل شود.

~~A believes that~~ $A \models A \overset{K_{as}}{\leftrightarrow} S$ (از قبل از اجرای پروتکل)

$B \models B \overset{K_{bs}}{\leftrightarrow} S$

$S \models A \overset{K_{as}}{\leftrightarrow} S$

$S \models B \overset{K_{bs}}{\leftrightarrow} S$

$S \models A \overset{K_{ab}}{\leftrightarrow} B$ ← هدف S این است.

$A \models S \Rightarrow A \overset{K}{\leftrightarrow} B$

$B \models S \Rightarrow A \overset{K}{\leftrightarrow} B$

$A \models S \Rightarrow (B \sim X)$

$B \models S \Rightarrow (A \sim X)$

$A \models \#(N_a)$

$B \models \#(N_b)$

$A \models \#(N_c)$

$B \triangleleft \{N_a, N_c\}_{K_{as}}$

$S \triangleleft \{N_a, N_c\}_{K_{as}}$; $S \triangleleft \{N_b, N_c\}_{K_{bs}}$

S دریافت و reply ارسال

$S \models A \overset{K_{as}}{\leftrightarrow} S$ ——— $S \models B \overset{K_{bs}}{\leftrightarrow} S$

$S \models A \sim (N_a, N_c)$ ——— $S \models B \sim (N_b, N_c)$

Subject:
Year. ۹۴ Month. ۲ Date ۱۵ ( )

3.

$B \models S \Rightarrow (A \overset{K_{ab}}{\leftrightarrow} B)$ $\quad B \models S \models A \overset{K_{ab}}{\leftrightarrow} B$

$B \models A \overset{K_{ab}}{\leftrightarrow} B$

$A \models A \overset{K_{ab}}{\leftrightarrow} B$

ثبت حق.

این طی اعتماد است به این دلیل فرض می‌شود.

اتهام این است Authentical God نسبت به کلیدی که

$B \models A \models A \overset{key}{\leftrightarrow} B$

$A \models B \models A \overset{K_{ab}}{\leftrightarrow} B$

قابل derive آن است.

چون از K_{ab} دلیل بعدی دلیل استفاده می‌شود.

و bellet دیگر قابل استنتاج است.

1. $A \models B \models N_c$

$A \models S \models (B \models N_c)$ $\quad S \models \#(N_c)$

$A \models S \models B \models N_c$ $\quad A \models S \Rightarrow (B \models X)$

$A \models B \models N_c$ $\quad A \models \#(N_c)$

$A \models B \models N_c$

↓

B کلیدی liveness

2. $B \models A \models N_c$

قبل از این N_c دیگر مشخص نمی‌شود که فرستاده شود. پس هر زمان که می‌خواهد

به آینده دیگر دوم، N_b را شروع به نشان دادن کند.

۷۵ (کلید اشتراکی) (Shared Key) مشکل دارد. BAN خوانده شود. مشکل‌های دیگر

Subject :
Year . ۲۰۱۴ Month . ۴ Date . ۲۵ ( )

Semantics برای BAN، Formal Trust Theory، Epostimic Logic برای

امروزه modal logic، طراحی و آنالیز شده است.

برای ایجاد ← STEP
(Stanford Temporal Prover)
برای Approach

Running Example –

The Needham-Schroeder Pubic Key Protocol (Mutual Authentication)

$$A \rightarrow B : \{ \langle A, N_a \rangle \}^{a}_{PK(B)}$$

a → asymetric

Concatenation : pairing

$$B \rightarrow A : \{ \langle N_a, N_b \rangle \}^{a}_{PK(A)}$$

$$A \rightarrow B : \{ N_b \}^{a}_{PK(B)}$$

این روش برای حمله challenge-Response

N_a و N_b هم برای ایجاد session است که در حمله

عملکرد حمله :

$$A \xrightarrow{\{A, N_a\}^{a}_{PK(B)}} \text{حمله‌گر} \rightarrow B$$

$$\text{حمله‌گر} \xrightarrow{\{N_a, y\}^{a}_{PK(B)}} B$$

حمله‌گر A

Subject :

Year . ۹۶ Month . ۴ Date ۲۰ ( )

B ← $\{Y_y, N_b\}^a_{PK(x)}$

A ← $\{\langle N_a, Z\rangle\}^a_{PK(A)}$

A → $\{\$Z\}^a_{PK(B)}$

B ← $\{N_b\}^a_{PK(B)}$

یک حمله از روی شکل:

A → $\{KA, N_a\}^a_{PK(C)}$ C

C(A) → $\{KA, N_a\}^a_{PK(B)}$ B

C(A) ← $\{N_a, N_b\}^a_{PK(A)}$ B

A ← $\{\langle N_a, N_b\rangle\}^a_{PK(A)}$ C

A → $\{N_b\}^a_{PK(C)}$ C

C(A) → $\{N_b\}^a_{PK(B)}$ B

در سال ۱۹۹۸ اصرد ... حمله روی پروتکل ... مطرح شد

... decryption oracle ... A ... ک ...

... شکل ... شده ... پروتکل ...

PqPCO

۷۷

Subject:
Year 94 Month 2 Date 26 ( )

$B \rightarrow A : \{\langle N_a, \langle B, N_b \rangle\}^{a}_{PK(A)}$

Applied - pi calculus CSP ... ; Constraint System , Strand Spaces

Terms:

Definition - Given a set X of variables and a set $\mathcal{N}$ of names (used to represent atomic data such as keys, nonces or identification), the set of terms of the signature F, the variables X and the names $\mathcal{N}$ (a set of function symbols → arity → ...) is denoted $T(F, X, \mathcal{N})$ and is inductively defined as names, variables, and function symbols applied to other terms.

x, y, z n_1, n_2, n_3 f_1, f_2

$Sterm \rightarrow x \; ; \; f_1(x, n_2) \; ; \; f_2(x, f_1(n_1, n_3), n_1) \; ; \; n_1$

Example - $F_{Std} = \{senc, aenc, pair, pk\}$ with arity 2 2 2 1

$t_0 = aenc(pair(a, n_a), pk(k_b))$

$\langle t_1, t_2 \rangle \triangleq pair(t_1, t_2)$

$\{t_1\}^{s}_{t_2} \triangleq senc(t_1, t_2)$ $\{t_1\}^{a}_{t_2} \triangleq aenc(t_1, t_2)$

Subject:
Year 98 Month 2 Date 30 ( )

positions of a term t is written $Pos(t) \subseteq \mathbb{N}^*$.

ε : empty string

$$Pos(f(t_1, \dots, t_n)) = \{\varepsilon\} \cup \bigcup_{i=1}^{n} i \cdot Pos(t_i)$$

(root)

The set of subterms of t is written $st(t)$. The subterm of t at position $p \in Pos(t)$ is written $t|_p$. In particular, $t|_\varepsilon = t$.

The term obtained by replaces $t|_p$ with a term u in t is denoted $t[u]_p$.

$t_0 = aenc(Pair(a, n_a), pk(K_b))$

$Pos(t_0) = \{\varepsilon, 1, 1.1, 1.2, 2, 2.1\}$

$st(t_0) = \{t_0, Pair(a, n_a), a, n_a, pk(K_b), K_b\}$

$t_0[aenc(n_a, pk(K_b))]_1 = aenc(aenc(n_a, pk(K_b)), pk(K_b))$.

position ... subterm ...

Substitution ...

(substitution)

Definition: A substitution is a function σ from a finite subset, called domain and denoted $Dom(\sigma)$ of the set of variables X, to $T(F, X, N)$

Subject:
Year. 2024 Month. 2 Date 28 ( )

when applied to a term, a substitution replaces any variable x by the corresponding term of $\sigma(x)$.

$$\sigma(x) = x \qquad x \notin Dom(\sigma)$$

$$\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$$

Two terms u and v are said unifiable if there exists a substitution σ called unification, such that $\sigma(u) = \sigma(v)$

$$u\sigma = v\sigma$$

Proposition - if two terms u and v are unifiable, then exists a most general unifier $mgu(u, v)$ such that any unifier is actually an instance of $mgu(u, v)$, i.e. for any σ such that

$u\sigma = v\sigma$, $\exists \theta. \ \sigma = mgu(u, v)\theta$.

Message Deduction:

deduce ... Attacker ...

Inference System

Formal Models and techniques for analyzing security protocols.
Steve Kremer, 2014 (Tutorial) 160-170 Pages

Subject:
Year. ۹۴ Month. ۲ Date. ۲۵ ( )

$\frac{u_1 \; u_2 \; \dots \; u_n}{u}$ → Induction definition (rule) — $u_1 \dots u_n$ premises, u conclusion

$\frac{}{u}$ → Axiom

$\frac{Senc(m,k) \quad k}{m}$

Rule Scheme

Definition – An inference rule is a rule of the form

$\frac{u_1 \dots u_n}{u}$

such that $u_1, \dots, u_n, u$ are terms (with variables). An inference system

logic

is a set of inference rules.

$\frac{P \quad P \to q}{q}$

Attacker deduce

I_{DY} : Inference System of Dolev-Yao

$\frac{x \quad y}{\langle x,y \rangle}$ $\frac{\langle x,y \rangle}{x}$ $\frac{\langle x,y \rangle}{y}$

composition / Introduction

$\frac{x \quad y}{Senc(x,y)}$ $\frac{Senc(x,y) \quad y}{x}$ $\frac{x \quad y}{aenc(x,y)}$ $\frac{aenc(x,P_k(y)) \quad y}{x}$

Symmetric encryption — asymmetric encryption

minimal of I_{DY}

Subject:
Year. Month. Date. ( )

Derivability:

$$S_0 = \left\{ \langle k_1, k_2 \rangle, \langle k_3, a \rangle, \{n\}^S_{\langle k_1, k_2 \rangle} \right\}$$

Is $\langle n, a \rangle$ derivable? (deducable) Can $\langle n, a \rangle$ be deduced from S_0?

$$\frac{\dfrac{\langle k_1, k_2 \rangle}{k_1} \quad \dfrac{\langle k_3, a \rangle}{k_3}}{\dfrac{\langle k_1, k_3 \rangle \quad \{n\}^S_{\langle k_1, k_3 \rangle} \to \mathrm{Senc}(n, \langle k_1, k_3 \rangle) \quad \langle k_3, a \rangle}{n} \quad a}{\langle n, a \rangle}$$

Inference ← Scheme / rule

proof tree or derivation tree → Bottom-up (forward)

Definition - Let I be an inference system. A term t is deducible in one step from a set of terms S for the inference system I, denoted $S \vdash^1_I t$ if there exists an inference rule $\frac{u_1 \dots u_n}{u}$ of I, and terms $t_1, \dots, t_n$ in S, and a substitution θ such that

$$t_i = u_i\theta \quad (1 \le i \le n) \text{ and } t = u\theta$$

We say that $\frac{t_1 \dots t_n}{t}$ is an instance of $\frac{u_1 \dots u_n}{u}$.

Subject: Formal
Year 9۴ Month. ۲ Date. ۲۲ ( )

A term t is deducible of a set of terms S, written $S \vdash_I t$, if there exists a proof tree π, that is a tree such that

- the leaves of π are labeled by terms in S.
- if a node of π is labeled by t_{n+1} and has a child nodes labeled by $t_1, \ldots, t_n$ respectively, then $\frac{t_1 \; \ldots \; t_n}{t_{n+1}}$ is an instance of some inference rule of I.
- The root of π is labeled by t.

* The set of terms (π) is the set of all terms that appear in any node of π.

The intruder deduction problem:

Let I be an inference system. The intruder deduction problem consists in deciding the following problem.

→ Decision Problem

Input: A finite set of terms S and a term t

Output: whether $S \vdash_I t$

Subject: Final
Year. 94 Month 02 Date 22 ( )

Abadi 2006 Gaurtee

* برای inference System undecidable دیگر این قضیه نیست. پس همین به راحتی می‌توان

ارائه کرد در این سوال از قضیه استفاده می‌شود. و برای Local Theory نیز همینطور قابل حل است.

Definition - A inference system I is local if for any finite set of terms S and for any term t such that S ⊢ t there exists a proof tree Π of S ⊢ t such that any term labeling Π is in st(S ∪ {t}).

↓ یعنی subterm

Theorem - let I be a local inference system. The intruder deduction problem is decidable in polynomial time (PTime).

proof -

- Let $S_0 = S$
- let $S_{i+1} = S_i \cup \{u \mid S_i \vdash u \wedge st(S \cup \{t\})\}$
- if $S_{i+1} = S_i$ then Stop.
- Check whether $t \in S_i$ (Yes / No)

$|st(S \cup \{t\})| = N \longrightarrow$ حداکثر N بار حلقه تکرار می‌شود تا $S_{i+1} = S_i$ شود.

Correctness: تمام local بودن، تمام الگوریتم زمان چندجمله‌ای بودن را نشان می‌دهد.

Subject: Formal
Year. ۹۴ Month. ۲ Date. ۲۲ ( )

Theorem- The Dolev-Yao inference System I_{DY} is local.

Proof- we say that a proof tree π of $S \vdash t$ is minimal if its number of nodes is minimal. (تعداد رئوس کمتری نسبت به بقیه درخت)

For any S, t such that $S \vdash t$, for any minimal proof π of $S \vdash t$, if holds that $Terms(\pi) \subseteq st(S \cup \{t\})$.

Moreover, if π is reduces to a leaf or ~~node~~ ends with in decomposition rule than $Terms(\pi) \subseteq st(S)$.

Base Case → $|nodes| = 1$ ⇒ $t \in S$

$S \vdash t$

π ← آخرین rule که درخت با آن تمام شده

A composition rule $\frac{t_1 \quad t_2}{t}$ $t = f(t_1, t_2)$

$|\pi_1| < |\pi|$

$|\pi_2| < |\pi|$

Subject: Formal
Year. 94 Month. 02 Date. 22 ( )

$Term_1(\pi_1) \subseteq st(su\{t_1\}) \subseteq st(su\{t\})$

$Term_2(\pi_2) \subseteq st(su\{t_2\}) \subseteq st(su\{t\})$

$\Rightarrow Term_3(\pi) \subseteq st(su\{t\})$.

(قرن ؛ خروجی) decomposition ، رابطه

Equational Theory and static Equivalence:

indistinguishability

Inference System

primitive

Exclusive OR

$m \oplus m = 0$ ⟶ Cancelation property

$\{m \oplus m\}_k = \{0\}_k$

Σ be a signature. An equational theory $=$ is a set of equations $u = v$, where u and v are terms $T(\Sigma, X, N)$.

The equivalence relation $=_E$ is defined by the equation of E closed by

reflexive
symmetric
transitive

Subject:
Year. 94 Month. 2 Date. 2[illegible]

reflexivity, symmetry, transitivity, substitution, and context.

$u =_E v$ for any $u = v \in E$ and any substitution

$u_1 =_E v_1, \ldots, u_k =_E v_k$ implies $F(u_1, \ldots, u_k) =_E F(v_1, \ldots, v_k)$

Context

Holes / terms / placeholder

Examples — XOR $(E_{\oplus})$

↳ Equational Theory Exclusive OR

$x \oplus (y \oplus z) = (x \oplus y) \oplus z$

$x \oplus y = y \oplus x$

$x \oplus x = 0$

$x \oplus 0 = x$ — identity

Fundamental theorem for XOR

Equality

If $u = k_1 \oplus k_2$, $v = k_2 \oplus k_3$, $w = k_1 \oplus k_3$ Then $w =_{E_{\oplus}} u \oplus v$.

$u \oplus v = (k_1 \oplus k_2) \oplus (k_2 \oplus k_3)$

$=_{E_{\oplus}} ((k_1 \oplus k_2) \oplus k_2) \oplus k_3$ Associativity and substitution

$=_{E_{\oplus}} (k_1 \oplus (k_2 \oplus k_2)) \oplus k_3$ Substitution on Associativity rule

Context

Context reflexivity, Symmetry, Associativity, Commutativity, Substitution, transitivity

Subject:
Year. 0[illegible] Month. 2 Date. 27 ( )

Equational Theory

$= E_{\oplus}(K_1 \oplus 0) \oplus K_3$

$= E_{\oplus} K_1 \oplus K_3 = W$

Example — modular exponitiation

E_{exp} (RSA, EL Gamely Diffie–Hellman)

$exp(exp(x,y),z) = exp(exp(x,z),y)$

$x^{y^z} = x^{z^y} \rightarrow = x^{y*z}$

$A \rightarrow B: exp(g, n_a)$

$B \rightarrow A: exp(g, n_b)$

$exp(exp(g, n_a), n_b) =_{Eexp} exp(exp(g, n_b), n_a)$

$exp(g, n_a) \rightarrow n_a$

discrete logarithm

Example — Encryption

E_{enc}

destructor operator $F_{dec} = \{sdec, adec, fst, snd\}$

Subject: Final
Year. 94 Month. 2 Date. 07 ( )

E_{enc} is an equational theory over $T(F_{std} \cup F_{dec} \cup F_{enc}, X)$.

(0 : any, 1 : Function Symbol)

Equational Theory for Encryption:

- $sdec(senc(x, y), y) = x$
- $adec(aenc(x, pk(y)), y) = x$
- $fst(pair(x, y)) = x$
- $snd(pair(x, y)) = y$

Deduction:

$\vdash_E$ is a deduction system defined by

Eq. Theory

1) $\dfrac{t \quad \text{if } t =_E t'}{t'}$

2) $\dfrac{t_1, \ldots, t_n}{F(t_1, \ldots, t_n)}$

Example: $E_{\oplus} \cup E_{enc}$ over $T(F_{std} \cup F_{dec} \cup \{\oplus\}, X)$

Let $S = \{senc(a, a \oplus c),\ a \oplus b,\ b \oplus c\}$

$S \vdash_{E_{\oplus} \cup E_{enc}} a$?

PgPCO

Ng

Subject: Formal
Year. ٢٠٢٢ Month. ٢ Date. ١٧ ( )

$$\frac{a \oplus b \quad b \oplus c}{(a \oplus b) \oplus b \oplus c}$$

$$\frac{a \oplus c \qquad \text{senc}(a, a \oplus c)}{\text{sdec}(\text{senc}(a, a \oplus c), a \oplus c)}$$

Attacker ... deduction ... Equational theory ...

Proposition — Let E be an equational theory. Let S be a set of terms and t be a term. Then, $S \vdash_E t$ iff there exists a context C (a term with holes) such that $n(C) = \emptyset$ and terms $t_1, \ldots, t_n \in S$ such that $t =_E C[t_1 \ldots t_n]$

where $C[t_1, \ldots, t_n]$ denotes C where the holes (variables) are replaced by t_i.

$$\text{sdec}(x, y \oplus z)[\text{senc}(a \oplus c), a \oplus b, b \oplus c] \quad (\text{Dy})$$

$$=_{E_\oplus \cup E_{enc}} a$$

... deduction ...

Proof — ١) $S \vdash_E t \Rightarrow \exists C, t_1, \ldots, t_n \in S.\ n(C) = \emptyset \wedge t =_E C[t_1, \ldots, t_n]$ induction ... only if part

٢) Context ... if part

(Cap Unification Problem)

{ = only if
if

Subject: Formal Approaches to Internet Security
Year. 94 Month. 02 Date. 27

Proposition - Let S be a set of terms and t be a term in the term algebra $T(\mathcal{F}_{std}, \mathcal{X}, \mathcal{N})$. Then,

$S \vdash t$ iff $S \vdash_{Enc} t$.
I_{DY}

Static Equivalence:

Definition - A frame is an expression $\varphi = \nu \tilde{n}.\theta = \nu \tilde{n}.\{M_1/x_1, \ldots, M_n/x_n\}$ where θ is a substitution, $\tilde{n}$ is a set of names that are restricted in φ. $M_1, \ldots, M_n$ → Attacker knowledge
$\tilde{n}$ → initially Attacker doesn't know
$Dom(\varphi) \triangleq Dom(\theta)$

$\nu k.\varphi = \nu(\tilde{n} \cup \{k\}).\theta$

Definition -
$\nu k = \{^0/_{x_1}, {}^1/_{x_2}, \ldots, senc(0,k)/_{x_3}\}$

Attacker

Subject: Formal Approaches to IS
Year. 24 Month. 7 Date. 29 ( )

(Deduction)

Definition — $\varphi \vdash_E t$ if $Dom(\varphi) \cup (\mathcal{N} \setminus \tilde{n}) \vdash_E t$

Example — $\varphi_1 = \nu n_1 k \theta_1$, $\theta_1 = \{ senc(pair(n_1, n), K) / x_1, K/y \}$

$\varphi_1 \vdash_{E_{enc}} n$

$M = fst(sdec(x, y))$

$M\theta_1 =_{E_{enc}} n$

Definition — Let $\varphi = \nu \tilde{n}.\theta$ be a frame and t a term such that $\varphi \vdash_E t$. A term R is said free with respect to φ if $n(R) \cap \tilde{n} = \emptyset$. (w.r.t)

A term R is a recipe of t in φ if R is free w.r.t. φ and if $R\theta =_E t$.

Let φ_1 be a frame, n a recipe, M a term. M is a recipe of n in φ_1.

Proposition — Let $\varphi = \nu \tilde{n}.\theta$ be a frame and t a term. $\varphi \vdash_E t$ iff there exists a recipe R of t in φ.

PdPCO

Subject: Forcing
Year. 94 Month. 02 Date. 29 ( )

$\Phi_1 = \{ {}^{0}/_{x}, {}^{1}/_{y} \}$ ($\to$ function symbol with arity 0) $\quad \Phi_2 = \{ {}^{1}/_{x}, {}^{0}/_{y} \}$

(name $\mathbb{R}, 1, 0$) . derive Φ_1, Φ_2 ...

$$\Phi_1 \vdash_E t \iff \Phi_2 \vdash_E t$$

... deduction ...

Definition – Given Frames Φ_1 and Φ_2, we write $\Phi_1 =_\alpha \Phi_2$ if Φ_1 is equal to Φ_2 up to a Conversion of restricted names. We say that (α-renaming)

the equation $M =_E N$ holds in Φ, written $(M =_E N)_\Phi$, iff there exists $\vec{n}$ and Θ such that $\Phi =_\alpha \nu \vec{n}\Theta$, M and N are free w.r.t. $\nu\vec{n}\Theta$, and $M\Theta =_E N\Theta$

Definition (Static Equivalence) – Two Frames Φ_1 and Φ_2 are statically equivalent w.r.t. an equational theory E, denoted $\Phi_1 \sim_E \Phi_2$, if $Dom(\Phi_1) = Dom(\Phi_2)$ and for any two terms M, N, we have that $(M =_E N)_{\Phi_1} \iff (M =_E N)_{\Phi_2}$. (Abadi and Fournet)

PdPCO

Subject:
Year ۲۰۲۴ Month ۲ Date ۲۹ ( )

Example - E_{enc}

$\Phi_1 = \forall K \{ aenc(0, PK(K))/x, PK(K)/y \}$

$\Phi_2 = \forall K \{ aenc(1, PK(K))/x, PK(K)/y \}$

Frame Φ است PK(K) و ۱ را دارد. encryption : عمومی PK(K) که به ۰ و

$(aenc(0,y) = x)_{\Phi_1}$ $(aenc(0,y) = x)_{\Phi_2}$

درست نیست نادرست است

$\Phi_1 \not\approx \Phi_2$

$\Phi_1 = \forall K, n \{ aenc(pair(0,m), PK(K))/x, PK(K)/y \}$

$\Phi_2 = \forall K, n \{ aenc(pair(1,m), PK(K))/x, PK(K)/y \}$

pair(0,m), pair(1,m) تابع Destructor است، پس نابرابری Restricted نیست.

$\Phi_1 \sim \Phi_2$

یک تابع Procedure که توان نرمیک معادله‌ها را بررسی می‌کند.

Subject: Formal Approaches
Year. 9۴ Month. ۱ Date. ۱۳ ( )

A cryptographic process Calculus:

specification برای کردن process و مدل کردن رفتارهای آن.

CCS, CSP Tony Hore

برای سیستم‌های Concurrent

Pi ← معروف‌ترین Process Calculus

Milner توسط

Concurrency Core Calculus

Extension‌هایی برای کاربردهای امنیتی:

Security Pi ← SPi

Applied-Pi

برای نوشتن specification‌های پروتکل‌های امنیتی

Crypto SPA

Applied Pi مورد استفاده قرار گرفت. در این زبان term‌ها به صورت نمادین مدل می‌شوند.

Syntax:

$$\mathcal{N}, \mathcal{X}, \mathcal{F}, \mathcal{T}(\mathcal{F}, \mathcal{X}, \mathcal{N}), \mathcal{E}$$

$\mathcal{E}$ ← Equational Theory

many-sorted

$\mathcal{N}_{ch} \subset \mathcal{N}$

channel names

مجموعه‌ای از اسم‌های اولیه Pi است.

$\mathcal{X} = \mathcal{X}_b \uplus \mathcal{X}_{ch}$ ← union disjoint

$\mathcal{X}_b$ ← base-sort

$\mathcal{X}_{ch}$ ← channel-Sort

Subject: Famt
Year. 94 Month. 3 Date. 3 ( )

Process : plain and extended

P, Q, R := plain processes

zero → 0

P || Q → parallel

!P → infinity parallel runs of P

$\nu n.P$ → P is restricted in name (name restriction) → local names

if $t_1 = t_2$ then P else Q

in (u, x).P → P is bound to x, input from u

out (u, t).P → output term t on u, then P

A, B, C := extended Processes

$\varsigma_{plain} ::= P$

A || B

$\nu n.A$

$\nu x.A$

variable, names

$\{t/x\}$ → Active Substitution → Term in Process, replaces x with t

let x = t in P ≡ $\nu x.(P \mid \{t/x\})$

→ Local

Subject: Formal
Year 2024 Month 11 Date 11 ( )

$\{t_1/x_1\} \| \{t_2/x_2\} \| \dots \| \{t_n/x_n\} = \{t_1/x_1, \dots, t_n/x_n\}$

$\phi(A)$ = The process obtained by replacing any plain Process with $\overset{\text{zero}}{0}$ in A.

* Extended Processes must not appear under a replication, an input, an output, or a conditional.

if $t_1 = t_2$ then out (c, k)

$\triangleq$ if $t_1 = t_2$ then out $(c, k).0$ else $\overset{2n}{0}$

trailing zero

$n(A)$ → free names
$fv(A)$ → free variables
$bv(A)$ → bound variables → متغيرات مقيدة بـ binder
$bn(A)$ → bound names

Active Substitution متغيرات قابلة للاستبدال

A context C is a process with a "hole" denoted by "$-$".

$C[A]$

Subject:
Year 92 Month 2 Date 2 ( )

$A \rightarrow B \quad aenc(\langle a, n_{ab} \rangle, pk_b)$

$B \rightarrow A \quad aenc(\langle n_a, n_b \rangle, pk_a)$

$A \rightarrow B \quad aenc(n_b, pk_b)$

$E_{enc}, F_{std} \cup F_{dec}$

Roles → Role → Initiator (A) / Responder (B)

Agents → Agent (می‌تونن) — ↳ Role (می‌تونن)

Bob, Alice می‌تونن Role داشته باشن

a, b یعنی چی؟

We parameterize the processes representing the initiator and responder with the keys of the agents who execute the role.

Process یعنی چی؟ اجرا می‌شه. agent یعنی Process

Role of Initiator:

$P_A(sk_i, pk_r) \triangleq \nu n_a . out(c, aenc(\langle pk(sk_i), n_a \rangle, pk_r)).$

A: initiator — sk_i: Secret Key — n_a: عدد تصادفی — $pk(sk_i)$: Public Key — pk_r: responder

$in(c, x).$

if $fst(adec(x, sk_i)) = n_a$ then

let $x_{nb} = snd(adec(x, sk_i))$ in

$out(c, aenc(x_{nb}, pk_r))$

Identity

Subject: Formal
Year. ۹۶ Month. ۴ Date. ۳ ( )

Role of Responder

$P_B(sk_r) \triangleq in(c, y).$

let $pk_i = fst(adec(y, sk_r))$ in

let $y_{na} = snd(adec(y, sk_r))$ in

$\nu n_b.\ out(c, aenc(\langle y_{na}, n_b \rangle, pk_i))$.

in (c, z).

if $adec(z, sk_r) = n_b$ then Q.

$P'_{nspk} \triangleq \nu sk_a.\ sk_b.\ (P_A(sk_a, pk(sk_b)) \,||\, P_B(sk_b) \,||$

$out(c, pk(sk_a)) \,||\, out(c, pk(sk_b)))$.

$P^2_{nspk} \triangleq \nu sk_a.\ sk_b.\ (P_A(sk_a, pk(sk_b)) \,||\, P_A(sk_a, pk(sk_c))$

$||\, P_B(sk_b) \,||\, out \ldots)$

$P^3_{nspk} = \nu sk_a.\ sk_b\ (in(c, x_{pk}).\ P_A(sk_a, x_{pk}) \,||\, P_B(sk_b) \,||$

$out(c, pk(sk_a)) \,||\, out(c, pk(sk_a)))$

این بسته جزو ۴ تا بسته‌ای است که در آن Session داریم و به Session های parallel می‌گویند.

$$P^{4}_{nspk} \triangleq \nu sk_a . sk_b . \Big(!\, in(c, x_{pk}) . P_A(sk_a, x_{pk}) \,\|\, !\, P_B(sk_b) \,\|\, out(c, pk(sk_a)) \,\|\, out(c, pk(sk_b)) \Big)$$

حال تعداد Session های ممکن برای اجرا Initiator می‌تواند با استفاده از سیستم ارسال کند.

نقش: Agent می‌تواند Responder, Initiator کدام است و نقش‌ها Agent اجرا کند.

Initiator یا Responder

$$P^{5}_{nsPK} \triangleq \nu sk_a . sk_b . \Big(!\, in(c, x_{pk}) . P_A(sk_a, x_{pk}) \,\|\, !\, P_B(sk_a) \,\|\, !\, in(c, x_{pk}) . P_A(sk_b, x_{pk}) \,\|\, !\, P_B(sk_b) \,\|\, out(c, pk(sk_a)) \,\|\, out(c, pk(sk_b)) \Big)$$

اگر یک Agent فقط نقش Initiator را اجرا کند به آن Honest Principal می‌گویند.

$$\triangleq !\, \nu sk_a \ldots$$

$$P^{6}_{nspk} \triangleq !\, \nu sk_a . \Big(!\, in(c, x_{pk}) . P_A(sk, x_{pk}) \,\|\, !\, P_B(sk) \,\|\, out(c, pk(sk)) \Big)$$

حداقل یک Agent صادق وجود دارد ← حمله نمی‌شود

Formal Approaches

Formal Semantics: operational semantics

structural equivalence $\equiv$ is the smallest equivalence relation closed under α-conversion of bound names and variables and application of evaluation context and such that:

(par-0) $A \| 0 \equiv A$ (0 is zero) (par-a) $(A \| B) \| C \equiv A \| (B \| C)$

(par-com) $A \| B \equiv B \| A$ (repl) $!P \equiv P \| !P$

(NEW-0) $\nu n.0 \equiv 0$ (NEW-PAR) $A \| \nu u.B \equiv \nu u.(A \| B)$; $u \notin FV(A) \cup n(A)$ (شرط برای این)

(NEW-C) $\nu u.\nu v.A \equiv \nu v.\nu u.A$ (u, v bound ; Free ; capture avoiding)

(ALIAS) $\nu x.\{t/x\} \equiv 0$ (SUBST) $\{t/x\} \| A \equiv \{t/x\} \| A\{t/x\}$

(REWRITE) $\{t_1/x\} \equiv \{t_2/x\}$; $t_1 =_E t_2$

ZERO ← (متغیر x با 0 جایگزین شود) (در همه جا x با t جایگزین شود)

Example - $out(c,t) \equiv out(c,t_2)$; $t_1 =_E t_2$ (برابر)

Internal Reduction: $\longrightarrow$ small-step ($\Longrightarrow$ is the smallest relation on Processes closed under structural equivalence and the application of evaluation context, with FV)

1) $out(c,t).P_1 \| in(c,x).P_2 \longrightarrow P_1 \| P_2\{t/x\}$

2) if $t=t$ then P else $Q \longrightarrow P$

3) if $t_1=t_2$ then P else $Q \longrightarrow Q$ where t_1, t_2 are ground and $t_1 \neq_E t_2$

2_1) if $t_1=t_2$ then P else $Q \longrightarrow P$ where $t_1 =_E t_2$ (قوانین 2 استفاده از تساوی در توصیف راست آشکار)

$\frac{P \to Q \quad P \equiv P'}{P' \to Q}$ $\quad \frac{P \to Q}{C[P] \to C[Q]}$

Example - Needham-Shroeder

حال انتظار داریم که (به راحتی) بتوانیم ویژگی‌های امنیتی را در این سیستم مدل‌سازی کنیم و (خصوصیات) Formal Semantics چگونه بررسی شوند. اولین چیزی که انتظار می‌رود، بررسی خاصیت محرمانگی (secrecy) در این سیستم‌ها است:

Observational Equivalence:

attacker ... process ... observer

A, B ... context ... نقش

A ⇓ a → channel

↳ decom bar → process A is able to send a message on channel a.

process

$\exists C[\_].A \xrightarrow{*} C[out(a,t).P]$ s.t. C does not bind a

(دقیقا مشابه قبل)

Definition - Observational Equivalence denoted $\approx$, is the largest symmetric relation R on closed extended processes, with same domain such that if A R B then:

1) $A \Downarrow a \Rightarrow B \Downarrow a$

2) $A \rightarrow^* A' \Rightarrow \exists B' : B \rightarrow^* B' \wedge A' R B'$

3) $\forall C[\_] . C[A] \; R \; C[B]$

C: closing context (Attacker)

example –

$A = in(c,x).\ \text{if } x=0 \text{ then } out(c,1)$

$B = in(c,x).\ \text{if } x=0 \text{ then } out(c,0)$

$C[\_] = out(c,0).\ in(c,y).\ \text{if } y=1 \text{ then } out(a,1) \parallel \_$

$C[A] \rightarrow^* out(a,1) \Rightarrow C[A] \Downarrow a$

$C[B] \not\rightarrow^* \qquad C[B] \not\Downarrow a$ ✗

example –

$A = in(c,x).\ \nu n.\ out(c, h(n))$

$B = in(c,x).\ \nu n.\ out(c, h(n,n))$

h → Free symbol

Label Semantics :

Semantics: process

$\xrightarrow{\alpha}$ label α

1) labelled operational semantics:

$\alpha = in(a,t)$ (a: channel name, t: term)

$\alpha = \nu x.\ out(a,x)$

$\alpha = out(a,c)$ (a: channel name)

$\alpha = \nu c.\ out(a,c)$