프로그래밍언어연구회

LiComR 2003

송광사

Language-based Information-Flow Security

2003. 08. 19

도경구
한양대학교

주제

- 기밀정보보호confidentiality 관련 보안정책 security policies을 명세specify하고 집행enforce 하는데 프로그래밍언어programming language 기술을 적용
  – 프로그램 의미구조 formal semantics
  – 프로그램 분석기법 static analysis

- 프로그래밍언어 분석 기술 + 정보보호 기술 = 차세대 정보보호 기술

기밀성 보장을 위한 대표적인 보안 메커니즘

- 접근제어 Access Control
  - 접근권한을 가진 프로세서에게만 파일 접근 허용
  - 파일에 접근허용 후 기밀정보의 부적절한 전달이나 누출에 대해서는 통제능력이 없음
- 방화벽 Firewalls
  - 외부와의 통신차단을 통하여 기밀정보보호
  - 허용된 일부 통신에 대한 통제능력이 없음
- 암호화 Encryption
  - 정보의 안전한 전달을 보장
  - 전달된 정보가 복호화된 후 기밀성 보호에 대한 통제능력은 없음
- Antivirus Software
  - 알려져 있는 악성코드의 패턴을 감지하여 보호
  - 알려져 있지 않은 새로운 공격에는 무방비

정보흐름의 보안 Information-Flow Security

채택된 정보보호 정책에
시스템이 항상 부합되는지
총체적으로 보장하기 위해서는
정보흐름information flow의 안전성을 보장해야 한다.

"정보흐름이 안전하다"란? Secure Information-Flow

- 침입자가 시스템의 공개된 출력을 보고 비밀 입력데이터를 유추할 없다.
- 불간섭noninterference 성질
  - 비밀 입력데이터만 바꾸어 실행시키는 경우, 공개 출력데이터는 같다.

정보누출 Information Leaks

- 가정
  - *h* : 비밀변수 (보안수준이 높은 변수)
  - *l* : 공개변수 (보안수준이 낮은 변수)
- 명시적 흐름explicit flow을 통한 정보 누출

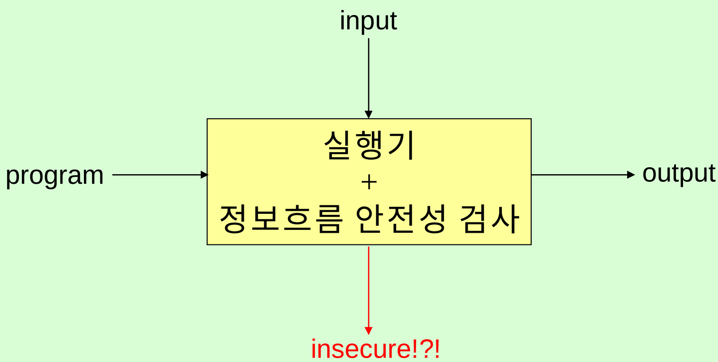
```
l = h
```

- 묵시적 흐름implicit flow을 통한 정보 누출

```
h = h mod 2;
l = 0;
if h == 1 then l = 1
          else skip
```

동적dynamic 정보흐름 제어

- 프로그램을 실행하면서 정보누출 여부 검사



Mandatory Access Control

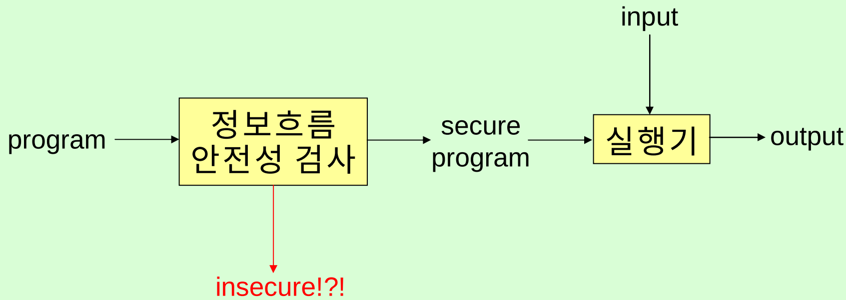
Fenton, Bell-LaPadula

- 어떻게?
  - 프로그램을 실행하면서 (동적으로) 보안수준을 표시한 라벨 process sensitivity label을 전파하여 정보흐름 제어
- 문제점
  - 계산시간 및 메모리 사용량 추가부담
  - 정확한 탐지를 위해서 모든 실행 경로 검사 필요 ⇒ 실행 중에는 근본적으로 불가능
  - 해법: 묵시적인 흐름implicit flow의 영향을 남은 프로그램에 미치게 함 [label creep = monotonically increasing labels]
  - Too restrictive to be practical

PL Lab
HYU

정적static 정보흐름 제어 ⇒ Static Certification

- 프로그램을 실행하지 않고 정적분석static analysis을 통하여 정보누출 여부 검사



PL Lab
HYU

정적static 정보흐름 제어 ⇒ Static Certification

- 좋은 점?
  - 분석의 정밀성 보장
    - 가능한 모든 실행 경로 고려
  - 계산시간 부담 감소
    - 실행 전에 정보흐름의 안전성을 검사하므로, 실행 중에 정보흐름을 검사할 필요가 없음
  - 분석의 안전성soundness 보장 가능
- 어떻게?
  - 프로그램 분석 기법 사용
    - Type Systems
    - Control- and Data-Flow Analysis (Flow Logic)
    - Abstract Interpretation
    - Model Checking

Semantics-based Security

- 분석의 안전성soundness이란?
  - 안전한secure 프로그램을 안전하지 않다고 insecure 할 수는 있으나, 안전하지 않은insecure 프로그램을 안전하다secure고 하지는 않는다.
- 안전성soundness은 어떻게 보장하나?
  - 불간섭noninterference 성질을 만족하는지 증명
  - 기밀정보보호와 관련된 보안 정책을 정형적인 의미구조formal semantics 사용하여 정의(명세)
  - 엄밀한 증명 가능

불간섭 Noninterference

state : $s = (s_h, s_l) \in S$

meaning : $[[C]] : S \rightarrow S_\perp$

- low input equivalence

  $s =_L s'$ *iff* $s_l = s_l'$

- low output (behavioral) equivalence

  $s \approx_L s'$

  iff they are indistinguishable to the attacker

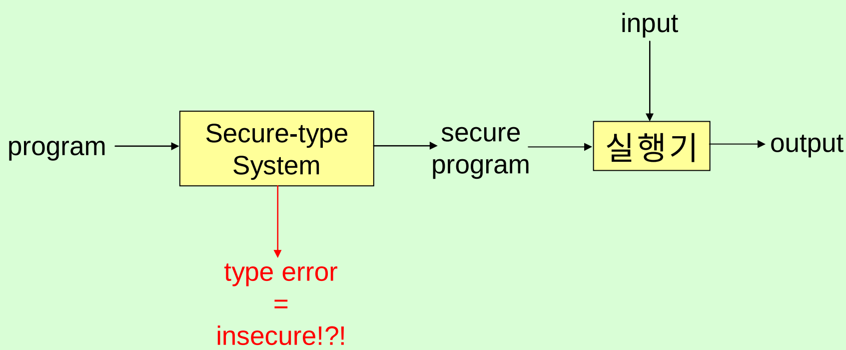
- noninterference

  C is secure iff

  $\forall s_1, s_2 \in S.\ \ s_1 =_L s_2 \Rightarrow [[C]]\, s_1 \approx_L [[C]]\, s_2$

타입시스템을 통한 정보흐름 제어

- 보안수준 = { 비밀*high*, 공개*low* }
- 보안수준을 타입으로 취급
- 정보흐름의 안전을 보장하는 타입시스템 설계



명령형 언어의 핵심부분에 대한 security-type system

Syntax :

C ::= $var := exp$ | `skip` | C_1 ; C_2
| `if` exp `then` C_1 `else` C_2
| `while` exp `do` C

[E1] $\vdash exp : high$

[E2] $$\frac{h \notin Vars(exp)}{\vdash exp : low}$$

[C1] $[pc] \vdash \texttt{skip}$

[C7] $$\frac{[high] \vdash C}{[low] \vdash C}$$

[C2] $[pc] \vdash h = exp$

[C3] $$\frac{\vdash exp : low}{[low] \vdash l = exp}$$

[C4] $$\frac{[pc] \vdash C_1 \qquad [pc] \vdash C_2}{[pc] \vdash C_1 \,;\, C_2}$$

[C5] $$\frac{\vdash exp : pc \qquad [pc] \vdash C}{[pc] \vdash \texttt{while}\ exp\ \texttt{do}\ C}$$

[C6] $$\frac{\vdash exp : pc \qquad [pc] \vdash C_1 \qquad [pc] \vdash C_2}{[pc] \vdash \texttt{if}\ exp\ \texttt{then}\ C_1\ \texttt{else}\ C_2}$$

PL Lab
HYU

타입검사를 통과한 안전한 프로그램

$$\dfrac{\overset{[C2]}{[low] \vdash h = l + 4} \qquad \dfrac{\dfrac{h \notin Vars(l - 5)}{\vdash l - 5 : low}\,[E2]}{[low] \vdash l = l - 5}\,[C3]}{[low] \vdash h = l + 4\,;\ l = l - 5}\,[C4]$$

$$\dfrac{\overset{[E1]}{\vdash h{=}{=}1 : high} \qquad \overset{[C2]}{[high] \vdash h = h + 4} \qquad \overset{[C1]}{[high] \vdash \texttt{skip}}}{[high] \vdash \texttt{if } h{=}{=}1 \texttt{ then } h = h + 4 \texttt{ else skip}}\,[C6]$$

PL Lab
HYU

타입검사를 통과하지 못한 안전하지 않은 프로그램

타입을 정하기가 불가능함

$$\dfrac{\vdash h : \cancel{low}}{[low] \vdash l = h}\,[C3]$$

$$\dfrac{\overset{[E1]}{\vdash h{=}1 : high} \qquad [high] \vdash \cancel{l := 1} \qquad \overset{[C1]}{[high] \vdash \texttt{skip}}}{[high] \vdash \texttt{if } h{=}{=}1 \texttt{ then } l = 1 \texttt{ else skip}}\,[C6]$$

Research Trends

- Enriching Language Expressiveness
- Exploring Concurrency
- Analyzing Covert Channels
- Refining Security Policies

Language Expressiveness

- Security type systems
  – a while language with 1st order procedures
  – a functional language with first-class functions (SLam calculus)
  – a first-class continuation, state and references
  – exceptions
  – objects (JFlow)

Nondeterminism

- the observable behavior of a program is the set of its possible results
- Possibilistic generalizations of noninterference
- Example:

$$h = h \bmod 2;$$
$$(l = h \parallel (l = 0 \parallel l = 1));$$

the final value of *l* reveals the least significant bit of *h* with the probability 0.5 + 0.5 * 0.5 = 0.75

- Solutions:
  - Analysis tracking dependencies between variables
  - Leino-Joshi's approach based on *equational security condition*
  - Sabelfeld-Sands generalizes *it* using PERs

Concurrency

- Multithreaded programs on a single processor

Thread 1	Thread 2
$h = 0;$	$h = h';$
$l = h;$	

- Timing- and probability-sensitive security

$$(\text{ if } h == 1 \text{ then } C_{long} \text{ else skip}); \; l = 1 \;\parallel\; l = 0 \;)$$

needs scheduler-independent security

- Concurrent languages with secure type systems

λ 은밀 채널 Covert Channels

- 채널 : 정보 전달 장치
- 은밀 채널 : 정보전달이 주목적이 아닌 채널
- 종류
  - 묵시적 흐름implicit flow
  - 종료 채널termination channels
  - 시간 채널timing channels
  - 확률 채널probability channels
  - 자원 소진 채널resource exhaustion channels
  - 전력 채널power channels

λ 종료 채널 Termination Channels

- 프로그램의 종료 여부의 관찰

```
while (h == 1) skip ;
```

- Termination-sensitive noninterference

  C is secure iff

  $$\forall s_1, s_2 \in S.\ \ s_1 =_L s_2 \Rightarrow [[C]]\, s_1 \approx_L [[C]]\, s_2$$

  where $s \approx_L s'$ *iff either* $s, s' \in S.\ \ s =_L s'$

  or $s = s' = \bot$

- Solution
  - Disallows high loops
  - Requires high conditionals have no loops in the branches

시간 채널 Timing Channels

- 실행 시간 관찰
  (if h == 1 then C_{long} else skip); $l = 1$ ‖ $l = 0$)
- Timing-sensitive noninterference
  C is secure iff
  $$\forall s_1,s_2 \in S.\ \ s_1 =_L s_2 \Rightarrow [[C]]\, s_1 \approx_L [[C]]\, s_2$$
  where $s \approx_L s'$ *iff both diverge or both terminate in the same number of execution steps in low-equal final states*
- A Solution
  - Requires high conditionals have no loops in the branches
  - Wraps each high conditional in a protect statement whose execution is atomic
- Another Solution
  - Closes timing leaks by program transformation

보안 정책 Security Policies

- 암호화된 데이터는 선택적으로 보안수준을 낮출 필요가 있음
  - 예: 패스워드 검사 프로그램, 암호 프로토콜
- Decentralized Model
  - Selective declassification of security labels is permitted
- Spi Calculus
  - a calculus of cryptographic protocols
  - Type systems that guarantee confidentiality (Abadi)
  - 암호화된 비밀키는 공개가능으로 취급

Future Directions

- system-wide security
  - 언어기반 정보흐름 제어 + 시스템 전반적인 정보흐름 제어
- certifying compilation (in the Trusted Computed Base)
  - Java bytecode verification
  - typed assembly language
  - proof-carrying code
- 동적으로 변하는 보안 정책
- 실용적인 이슈
  - 보안정책을 위반하지 않는 프로그램의 효율성
  - Security-type inference system의 구현
  - 분석의 정밀도precision 증가
- 다양한 정적분석 방법 사용

Discussions