情 報 保 護 學 會 誌
第 16 卷 第 5 號, 2006. 10

정보흐름보안성 분석기술

신 승 철*

요 약

소프트웨어보안 분야는 정보보호, 소프트웨어공학, 프로그래밍언어 분야 등이 중첩되는 곳에 위치한다. 본 고는 프로그래밍언어 기술을 이용하여 소프트웨어 보안문제를 접근하는 한 예로서 정보흐름 보안성 분석법을 설명한다. 먼저 정보흐름 보안성이 관련되는 보안 문제들을 상기시킨 후에 이를 해결하는 프로그래밍언어 기술의 기본 개념들을 프로그램 분석법 중심으로 설명하고 최신 연구 경향을 소개한다.

Ⅰ. 서 론

소프트웨어보안(software security)이라는 용어는 프로그램 보안(program security)이나 응용프로그램 보안(application security) 등으로도 불린다. 소프트웨어보안은 정보보호 분야와 소프트웨어공학 분야 그리고 프로그래밍언어 분야 등이 비슷하거나 서로 변환 가능한 목적과 수단을 공유하는 와중에 새로이 형성된 틈새 분야라고 할 수 있다. 소프트웨어보안이 다루는 문제는 크게 다음 두 가지 질문으로 요약할 수 있다:

- 보안취약점을 갖지 않는 소프트웨어를 어떻게 하면 만들 것인가?
- 주어진 소프트웨어가 보안취약점을 가지지 않는다고 어떻게 하면 확신할 수 있나?

정보보호의 관점에서는 접근제어시스템과 같은 보안용 소프트웨어이든 아니면 웹브라우저와 같은 일반 소프트웨어이든 보안취약점을 가짐으로써 본래의 보안 목적을 완전하게 달성하지 못하거나 아니면 예기치 않은 공격 방법을 제공하는 경우를 어떻게 예측하고 방지할 것인가하는 문제가 소프트웨어보안이라고 할 수 있다. 소프트웨어공학 관점에서는 소프트웨어 개발 프로세스와 소프트웨어 검증 부분에서 특정 보안취약점에 집중한다면 소프트웨어보안 문제가 된다. 마찬가지로 프로그래밍언어 관점에서도 타입 시스템과 같은 성공적인 프로그램 분석법이나 프로그램 검증 방법을 이용하여 보안취약점의 존재 여부를 증명한다면 이것을 소프트웨어보안 문제라고 할 수 있다.

컴퓨터 시스템에서 정보의 기밀성을 보호하는 것은 매우 오래된 문제이지만 그 중요성은 여전히 점점 증가하고 있다. 그럼에도 불구하고 정보의 기밀성을 완전하게 확보하는 일은 쉽지가 않다. 기존 보안 이론의 틀에서 이러한 보안 성질을 표현하기가 적절하지 않거나 기밀성을 확보하는 메카니즘도 만족스럽지 못한 경우가 많이 있다.

또한 접근제어, 암호, 방화벽 등 일반적인 보안 메카니즘들은 프로그램을 블랙박스로 취급해왔다. 이런 방식으로는 인터넷에서 다운로드된 이동성 코드가 버퍼오버런과 같은 보안취약점을 가지는 지에 대한 조사가 불가능하다. 소프트웨어보안기술 중에서도 언어기반 보안기술(language-based security)은 프로그램 속을 수학적이고 논리적인 방법으로 엄밀하게 들여다봄으로써 프로그램 내부의 보안취약점 존재 여부를 확인하는 것이다. 여기에는 프로그램 시멘틱스와 프로그램 분석법과 같은 프로그래밍언어 기술이 핵심적인 역할을 한다. 본 논문에서는 언어기반 보안기술을 이용하여 정보흐름보안성이라 불리는 정보 기밀성에 대한 보안 명세와 보안 메카니즘을 구현하는 방법을 소개한다. 다음 장에서부터 먼저 정보흐름보안성을 비형식적으로 정의한 후에 정보흐름보안성 명세를 좀 더 형식적으로 제시하고, 정보흐름보안성 메카니즘으로

* 한국기술교육대학교 인터넷미디어공학부 (scshin@kut.ac.kr)
이 논문은 2006년도 한국기술교육대학교 신임교수 연구과제 연구비 지원에 의하여 연구되었음.

사용되는 언어기반 기술인 타입시스템과 요약해석을 중심으로 프로그램 정적 분석법을 설명한다. 또한 정보흐름보안성 문제의 좀 더 복잡한 변형과 함께 최신의 연구 경향에 대하여 소개한다.

II. 정보흐름과 소프트웨어보안

기밀 정보를 보호하는 전통적인 방법은 접근제어시스템이다. 기밀 데이터를 포함하는 파일이나 자원에 접근하기 위해서는 특별한 권한이 요구된다. 이러한 방법으로 접근제어시스템은 기밀 정보의 노출을 제한할 수 있다. 하지만 기밀 정보의 전달과 전파까지 제한할 수는 없다. 예를 들어 기밀 정보의 접근이 허가된 프로그램이 의도적이든 아니면 오류에 의해서든 허가되지 않은 사람이나 프로그램에게 기밀 정보를 전송하는 것을 제어할 수 없다. 또한 어떤 인증시스템이나 바이러스검사기도 기밀 정보의 전파를 제어할 방법을 가지고 있지 않다. 데이터가 기밀성을 항상 만족시키면서 사용된다는 것을 확인하기 위해서는 데이터를 사용중인 프로그램 내에서 정보가 어떻게 흘러가는 지를 분석할 필요가 있다. 그러나 오늘날 컴퓨터시스템에서 사용되는 프로그램들은 매우 복잡하기 때문에 사람이 손수 분석하는 것도 불가능하다.

예를 들어 인터넷에서 작업 중에 [그림 1]과 같은 창을 만나는 일은 이제 일상이 되었다. 어떤 웹 페이지를 보기 위해서 또는 인터넷 결제 등의 작업을 수행하기 위해서 필요한 프로그램을 설치할 것을 요구하고 있다. 사용자가 이 프로그램을 설치하고 사용한다면 이 프로그램은 사용자의 기밀 정보에 접근할 수 있도록 허가된다. 그러나 이 프로그램이 트로이목마처럼 사용자가 알지 못하는 사이에 사용자의 기밀 정보를 인터넷을 통해서 외부로 유출할 가능성을 없을까? 만일 매우 보안성에 민감한 환경에서 작업 중인 사용자라면 기밀 정보의 전파를 제어하는 어떤 방법이 반드시 필요하다.

이와 같이 기밀성이 기밀 정보의 접근 뿐 아니라 접근 후의 전파까지 포함하는 경우를 특별히 정보흐름 보안성(information flow security)이라고 한다. 정보흐름 보안성을 제어하는 메카니즘은 오랫동안 실용적으로 개발되지 못했는데 최근에 타입시스템이나 요약해석과 같은 프로그래밍언어 기술의 이용으로 정보흐름 보안성 메카니즘이 다양하게 고안되었다.

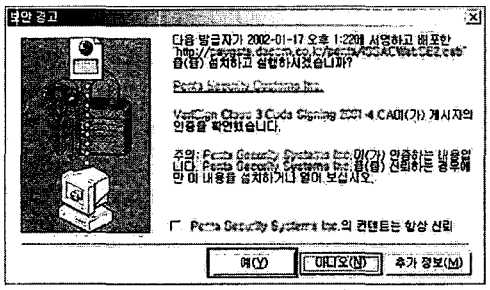
정보흐름 보안성 메카니즘들을 설명하기 전에 먼저 프로그램 내부에서 정보가 흘러가는 형태를 살펴보고 해결해야 할 문제를 명확하게 정의하고자 한다. 어떤 프로그램이 사용자의 데이터 입력으로부터 출력 데이터를 만들어낸다고 하자. 설명을 간단하게 하기 위해서 본 논문에서는 모든 데이터는 두 개의 보안 수준(secret, public)으로 구분된다고 가정한다. [그림 2]와 같이 사용자의 입력 데이터에는 기밀 정보와 공개 정보가 있다. 또한 출력 데이터도 기밀 정보와 공개 정보로 구분할 수 있다. 정보흐름 보안성을 위배하고 입력된 기밀 정보가 직접적이든 간접적이든 공개 출력 정보로 흘러가는 경우에 정보 누출(information leak)이 발생한다고 한다.

다음 프로그램 문장에서 변수 h가 기밀 정보를 가지고 있고 l이 공개 정보를 가지고 있다면 이 프로그램 문장은 명시적 정보 누출(explicit information leak)을 나타낸다.

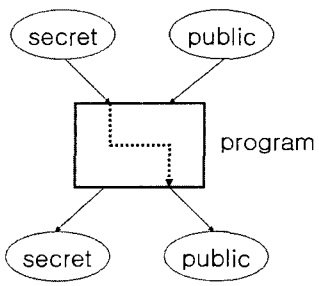
```
l := h;
```

다음의 프로그램 문장에서는 명시적 정보 누출은 없지만 공개 정보인 l을 관찰하여 h에 대한 부분적인 정보를 유추할 수 있으므로 묵시적 정보 누출(implict information leak)이 발생한다.

```
if h > 0 then l := 0 else l := 1;
```



[그림 1] 인터넷 보안 경고



[그림 2] 정보흐름보안성 위배

묵시적 정보 누출은 if와 while 같은 제어문에서 발생하는데 then 또는 else 부분의 문장이 수행되느냐 마느냐에 따라서 조건문에 포함된 기밀 정보가 누출된다. 이것은 은닉 채널(covert channel)의 일종이기도 하다. 프로그램과 관련된 은닉 채널은 묵시적 정보흐름외에도 종료 채널과 시간 채널 등이 있지만 본 논문에서는 논외로 하기로 한다.

Ⅲ. 정보흐름보안성의 명세: 불간섭

사용자가 어떤 데이터의 기밀성을 유지하고자 한다면 다른 사용자에게 공개되는 데이터는 그 기밀 데이터의 정보를 직접적이든 간접적이든 포함하지 않도록 보안 정책을 수립할 필요가 있다. 이런 보안 정책은 사용자가 실행하는 프로그램들이 사용자의 기밀 정보를 조작하고 수정할 수 있게 허가하면서도 [그림 2]와 같은 정보 누출이 없다는 것을 확인하도록 해야 한다. 이러한 종류의 정책을 불간섭 정책(noninterference policy), 불간섭 성질이라고 한다. 이것은 기밀 데이터가 공개 데이터 생성에 영향을 주지 않고 간섭하지 않는다는 것을 의미한다.

여기서 공격자는 기밀 데이터는 접근할 수 없지만 공개 데이터를 접근할 수 있으며 공개 데이터와 프로그램을 이용하여 기밀 데이터를 유추하고자 시도할 것이다. 주어진 프로그램이 불간섭을 만족하는 지를 증명하는 방법은 다음의 시나리오를 확인하는 것이다. 프로그램의 입력 중에서 기밀 데이터 부분만 서로 다른 두 입력에 대하여 프로그램을 각각 실행한다. 이때 얻어지는 두 출력 중에서 공개 데이터 부분들이 동일하므로 공격자에 의해 전혀 구별되지 않는다.

불간섭 성질은 프로그램 실행에 대한 시멘틱스에 의해서 자연스럽게 표현될 수 있다.[21,22,23] 다음과 같은 문법을 가진 간단한 명령형 언어 While을 생각해보자. 이 언어는 배정문, skip, 복합문, 조건문, 반복문으로 이루어진다.

$$
\begin{aligned}
S ::= & \; x := a \mid \text{skip} \mid S_1;S_2 \\
& \mid \text{if } b \text{ then } S_1 \text{ else } S_2 \\
& \mid \text{while } b \text{ do } S
\end{aligned}
$$

여기서 a는 변수($x \in Var$)와 상수($n \in N$)로 이루어진 산술식이고 b는 부울식이다. 다음은 While 프로그램의 동작 시멘틱스를 big-step으로 표현한 것이다. A와 B는 각각 산술식과 부울식의 시멘틱 함수이고 σ: $Var \rightarrow N$:는 변수가 주어지면 변수의 현재 값을 주는 상태 함수이다. 시멘틱스에 대한 표기는 표준적인 기법을 따르고 있으며 더 자세한 사항은 [2]를 참조하기 바란다.

[배정문] $\langle x := a, \sigma \rangle \rightarrow \sigma[x \mapsto A[[a]]\sigma]$

[skip문] $\langle \text{skip}, \sigma \rangle \rightarrow \sigma$

[복합문] $$\frac{\langle S_1, \sigma \rangle \rightarrow \sigma' \; \langle S_2, \sigma' \rangle \rightarrow \sigma''}{\langle S_1; S_2, \sigma \rangle \rightarrow \sigma''}$$

[조건문1] $$\frac{\langle S_1, \sigma \rangle \rightarrow \sigma'}{\langle \text{if b then } S_1 \text{ else } S_2, \sigma \rangle \rightarrow \sigma'} \quad \text{if } B[[b]]\sigma = true$$

[조건문2] $$\frac{\langle S_2, \sigma \rangle \rightarrow \sigma'}{\langle \text{if b then } S_1 \text{ else } S_2, \sigma \rangle \rightarrow \sigma'} \quad \text{if } B[[b]]\sigma = false$$

[반복문1] $$\frac{\langle S, \sigma \rangle \rightarrow \sigma' \; \langle \text{while b do } S, \sigma' \rangle \rightarrow \sigma''}{\langle \text{while b do } S, \sigma \rangle \rightarrow \sigma''} \quad \text{if } B[[b]]\sigma = true$$

[반복문2] $\langle \text{while } b \text{ do } S, \sigma \rangle \rightarrow \sigma$ if $B[[b]]\sigma = false$

프로그램의 모든 입력과 출력에 대하여 보안 수준(secret과 public)이 주어지면 입력 상태도 순서쌍의 형태로 표기된다.

$$\sigma = (\sigma_{secret}, \sigma_{public})$$

σ_{secret}은 기밀 정보를 보관하는 변수, σ_{public}은 공개 정보 변수에 대하여 상태를 나타낸다. 프로그램의 실행은 주어진 프로그램 S와 입력 상태를 나타내는 상태 함수 σ에 대하여 다음과 같이 나타낸다. 여기서 σ'은 프로그램 최종(출력) 상태이다.

$< S, \sigma > \rightarrow \sigma'$

즉,

$< S, (\sigma_{secret}, \sigma_{public}) > \rightarrow (\sigma_{secret}', \sigma_{public}')$

이제 불간섭 성질을 정의하기 위해서 상태 함수에 대한 동치 관계 $=_{public}$을 이용한다. 두 개의 상태 함수가 각 공개 변수들에 대하여 서로 같은 값들을 갖는다고 다음과 같이 표기한다.

$\sigma =_{public} \sigma'$ iff $\sigma_{public} = \sigma_{public}'$

그러면 불간섭 성질은 다음과 같이 정의된다:

주어진 프로그램 S와 두 입력 상태 σ_1와 σ_2에 대하여 $\sigma_1 =_{public} \sigma_2$ 일 때 $<S,\sigma_1> \rightarrow \sigma_1'$ 이고 $<S,\sigma_2> \rightarrow \sigma_2'$ 이면 $\sigma_1' =_{public} \sigma_2'$ 이다.

정의된 불간섭 성질은 프로그램 실행시 공개 변수의 변화가 기밀 변수로부터 영향 받지 않음을 나타낸다. 정보흐름보안성을 확인해주는 언어기반 기법들은 모두 위의 불간섭 성질을 보장하도록 구현되어야 한다.

이제부터는 주어진 프로그램이 불간섭 성질을 항상 만족하는 지를 확인하는 정적 분석법 두 가지(타입시스템과 요약해석)를 보안 메카니즘으로서 설명한다. 정적 분석법은 프로그램 실행 없이 프로그램의 동작을 예측하는 것이므로 100% 완전할 수는 없다. 그러나 분석에 의해 불간섭 성질이 만족된다고 판단된 프로그램이 내부에 정보흐름 보안성을 위배하는 일은 절대 없다. 대신 불간섭 성질을 만족하지 않는다고 판단된 프로그램이 실제로는 불간섭 성질을 만족하는 경우는 있을 수 있는데 이 경우는 분석의 정밀도가 떨어진다고 한다. 이 때문에 정적 분석법은 안전한 근사(safe approximation; conservative approximation)를 한다고 한다.

Ⅳ. 정보흐름보안성 분석: 보안 타입시스템

보안 타입시스템(security-type system)은 표준적인 타입시스템을 변형하여 정보흐름보안성을 확인하도록 고안된 것이다. 타입시스템은 타입규칙들의 모임으로 프로그램에 어떤 보안 타입을 줄 것인지를 부분 프로그램의 타입으로부터 결정한다. 주어진 산술식 a가 타입규칙에 의해서 τ 타입을 갖는다는 것을 $\vdash a:\tau$와 같이 표기한다. 또 주어진 프로그램 S가 보안문맥(security context) pc에서 타입가능하다는 것을 $pc \vdash S$ 와 같이 표기한다. 여기서 타입가능하다는 것은 프로그램이 정보흐름보안성 명세를 따른다는 것을 의미한다. 보안문맥 pc는 프로그램의 실행 여부를 결정하는 프로그램 제어에 대한 보안 수준을 나타낸다. 즉, pc가 secret이라면 프로그램에서 배정문의 좌변에 오는 변수들은 secret 변수의 영향을 받는다는 것을 의미한다. 다음은 앞에서 정의된 While 언어를 위한 보안 타입시스템인데 [8]의 것을 간략화한 것이다.

[식] $\vdash e : secret \quad \dfrac{h \notin Vars(e)}{\vdash e : public}$

[skip문] $pc \vdash \text{skip}$

[배정문-기밀변수] $pc \vdash h := a$

[배정문-공개변수] $\dfrac{\vdash a : public}{public \vdash l := a}$

[복합문] $\dfrac{pc \vdash S_1 \quad pc \vdash S_2}{pc \vdash S_1; S_2}$

[조건문] $\dfrac{\vdash b : pc \quad pc \vdash S_1 \quad pc \vdash S_2}{pc \vdash \text{if } b \text{ then } S_1 \text{ else } S_2}$

[반목문] $\dfrac{\vdash b : pc \quad pc \vdash S}{pc \vdash \text{while } b \text{ do } S}$

[포함] $\dfrac{secret \vdash S}{public \vdash S}$

여기서 h와 l은 각각 보안수준 secret과 public을 가진 변수라고 가정했다. e는 산술식 a 또는 부울식 b를 나타낸다.

타입시스템에서 [식]은 두 개의 타입규칙을 한꺼번에 보여주는데 산술식이나 부울식이 어떤 경우에 public이 될 수 있는지를 나타낸다. 즉, 어떤 식 e도 secret 보안수준을 가질 수 있지만 public이 되기 위해서는 식 e의 내부에 secret 변수가 없어야 한다.

타입규칙 [skip문]은 skip은 임의의 보안문맥에서 타입가능함을 나타내고 [배정문-기밀변수]도 임의의 보안문맥에서 secret 변수에 대한 배정문은 타입가능하다는 것을 의미한다. 그러나 [배정문-공개변수] 규칙에 의하면 public 변수에 대한 배정문은 public 보안수준의 보안문맥 하에서 배정문의 우변 식 또한 public 보안 수준을 가지는 경우에만 타입가능하다.

타입규칙 [조건문]과 [반복문]에서 나타나듯이 조건식이 기밀변수를 포함한다면 조건문과 반복문의 then/else 부분과 본체 부분은 secret 보안수준의 보안문맥 하에서 타입가능해야 한다. 조건식의 보안수준은 조건식에 의해 실행 여부가 결정되는 문장들의 보안문맥이 된다. 즉, 묵시적인 정보흐름의 보안성을 강제하고 있다.

타입규칙 [포함]은 프로그램이 secret 보안문맥에서 타입가능하면 public 보안문맥에서도 타입가능함을 의미한다. 이것은 조건문이나 반복문의 실행 후에 보안문맥을 secret에서 public으로 낮추는 것을 가능하게 한다. 그렇지 않으면 보안문맥이 secret으로 한번 정해지면 public으로 돌이킬 방법이 없기 때문에 불필요하게 너무 많은 프로그램이 타입 불가능해지므로 분석의 정밀도가 비현실적으로 떨어지게 된다.

예를 들면, 프로그램 $h := h + l; l := l + 1$는 타입가능하고 $\text{if } h > 0 \text{ then } h := h - l \text{ else } h := h + l$도 타입가능하다. [그림 3]은 이 두 프로그램이 각각 타

입가능함을 보여주는 증명을 보여준다. 반면에 프로그램 if $h>0$ then $l:=1$ else $l:=2$은 타입불가능하다. 조건식으로부터 생성된 보안문맥이 secret이므로 then/else 부분이 불가능하다. 따라서 [그림 4]와 같이 타입 증명을 완성할 수 없다.

$$\frac{\dfrac{h \notin Vars(l+1)}{\vdash l+1 : public}}{\dfrac{public \vdash h := h+l \quad public \vdash l := l+1}{public \vdash h := h+l; l := l+1}}$$

$$\frac{\vdash h>0 : secret \quad secret \vdash h := h-l \quad secret \vdash h := h+l}{\dfrac{secret \vdash \text{ if } h>0 \text{ then } h := h-l \text{ else } h := h+l}{public \vdash \text{ if } h>0 \text{ then } h := h-l \text{ else } h := h+l}}$$

[그림 3] 두 프로그램의 타입 증명

$$\frac{\vdash h>0 : secret \quad secret \vdash l := 1 \quad secret \vdash l := 2}{\dfrac{secret \vdash \text{ if } h>0 \text{ then } l := 1 \text{ else } l := 2}{public \vdash \text{ if } h>0 \text{ then } l := 1 \text{ else } l := 2}}$$

[그림 4] 완성될 수 없는 타입 증명

타입시스템과 같은 정적 분석법의 중요한 장점 중 하나는 분석법이 항상 올바른 답을 준다 - 적어도 틀린 답을 주지 않는다 - 는 것을 증명할 수 있다는 것이다. 여기서는 타입시스템이 정보흐름보안성을 확인하는 보안 타입시스템이므로 "타입시스템을 통과하는 모든 프로그램들이 불간섭 성질을 만족한다"는 것을 증명할 수 있다. 이것을 앞의 불간섭 성질의 정의와 관련하여 형식적으로 나타내면 다음과 같다:

주어진 프로그램 S에 대하여 $public \vdash S$ 이고 임의의 두 입력 상태 σ_1와 σ_2에 대하여 $<S,\sigma_1> \rightarrow \sigma_1'$ 이고 $<S,\sigma_2> \rightarrow \sigma_2'$ 이면 $\sigma_1 =_{public} \sigma_2$ 일 때 $\sigma_1' =_{public} \sigma_2'$ 이다.

사실 이것은 위의 보안 타입시스템에 대하여 증명되었다.[8]

V. 정보흐름보안성 분석: 요약해석

요약해석[6]은 앞서 정의한 While 언어의 동작 시멘틱스를 요약하여 실제 값에 의한 동작 시멘틱스가 아닌 요약 값에 의한 동작 시멘틱스를 나타낸다. 이렇게 함으로써 프로그램을 실제로 실행시키지 않고 관찰하고자하는 성질(이를테면 정보흐름)만을 조작하는 요약된 실행을 보여준다.

요약해석에서는 While 프로그램의 값 정의역 N은 요약값 정의역 $SC=\{secret, public\}$으로 대치된다. 이때 SC는 $public \sqsubset secret$와 같은 부분순서를 갖는 부분순서 집합이다. 즉, $public \sqcup public = public$, $secret \sqcup secret = secret$, $secret \sqcup public = secret$ 이다.

산술식 a의 실제 값을 구하는 시멘틱 함수 A는 요약 시멘틱 함수 $A^{\#}$이 되고 여기서 상태 함수 σ 대신에 요약 상태 함수 $\rho: Var \rightarrow SC$가 사용된다. 다음은 While 언어의 요약 동작 시멘틱스를 나타내는 규칙들이다. 표준적인 요약해석을 이용하고 있으므로 상세한 사항은 [2]를 참조하기 바란다.

[배정문] $\langle x := a, \rho, pc\rangle \rightarrow \rho[x \mapsto A^{\#}[[a]]\rho \sqcup pc]$

[skip문] $\langle \text{skip}, \rho, pc\rangle \rightarrow \rho$

[복합문] $$\frac{\langle S_1, \rho, pc\rangle \rightarrow \rho' \quad \langle S_2, \rho', pc\rangle \rightarrow \rho''}{\langle S_1; S_2, \rho, pc\rangle \rightarrow \rho''}$$

[조건문] $$\frac{\langle S_1, \rho, pc'\rangle \rightarrow \rho' \quad \langle S_2, \rho, pc'\rangle \rightarrow \rho''}{\langle \text{if b then } S_1 \text{ else } S_2, \rho, pc\rangle \rightarrow \rho''' }$$

where $pc' = B^{\#}[[b]]\rho \sqcup pc$ and $\rho''' = \rho' \sqcup \rho''$

[반복문] $$\frac{\langle S, \rho, pc'\rangle \rightarrow \rho'}{\langle \text{while b do S}, \rho, pc\rangle \rightarrow \rho'}$$

where $pc' = B^{\#}[[b]]\rho \sqcup pc$

여기서 pc는 앞의 타입시스템에서 나타난 것과 같은 의미를 가진다. 즉, 해당 프로그램 문장의 실행 여부를 결정하는 보안문맥을 나타낸다.

산술식 a의 요약값은 요약함수를 이용하여 얻어지는데 $A^{\#}[[a]]\rho = \sqcup\{\rho x \mid x \in Vars(a)\}$와 같이 정의된다. 즉, 산술식 내의 변수가 가진 보안 수준 중에서 가장 높은 값이 산술식의 요약값이 된다. 변수 x에 대한 [배정문]은 요약 상태 ρ를 기반하여 산술식 a의 요약값 즉, 보안수준을 계산하고 이것과 보안문맥 pc 중에서 최고값을 변수 x에 배정한다.

[skip문]은 동작 시멘틱스에서와 마찬가지로 요약 시멘틱스에서도 상태 함수가 변하지 않는다. [복합문]의 경우 첫 부분을 실행하고 난 후의 요약 상태 함수가 두 번째 부분의 실행에 사용된다. 그러나 보안문맥 pc는 양쪽에 동일하게 사용된다. 이것은 두 부분이 동일한 범위(scope)에 위치하기 때문에 동일한 보안문맥을 갖는다는 것을 의미한다.

[조건문]은 S_1과 S_2가 조건식 b의 영향 하에 있음을 나타내기 위하여 b의 요약값이 S_1과 S_2의 보안문맥을 구성하는 데에 사용된다. 게다가 S_1과 S_2을 수행한

결과로 각각 얻어진 요약 상태 함수 ρ'과 ρ''을 병합하여 전체 조건문의 결과 상태 ρ'''을 만든다.(만일 같은 변수에 대한 ρ'과 ρ''의 값이 다르다면 높은 값이 ρ'''의 값이 된다.) 이것은 요약해석에 의한 분석법이 모든 가능한 실행 경로를 포함하도록 하기 위함이다.

[반복문]의 경우는 [조건문]과 유사하다. 루프 본체 S의 실행은 전체 반복문의 보안문맥에 조건식 b의 요약값을 추가하여 수정된 보안문맥 하에서 이루어진다.

요약 상태 함수 ρ에서 요약값 public을 갖는 변수들의 집합을 $L(\rho)$로 표시하자. $\rho\, h = secret$ 이고 $\rho\, l = public$ 라고 가정하자. 그러면 프로그램 $h := h + l;\ l := l + 1$은 정보흐름보안성을 만족한다. 왜냐하면 $\langle h := h + l;\ l := l + 1, \rho, public\rangle \rightarrow \rho$ 이기 때문이다. 또 $\text{if } h > 0 \text{ then } h := h - l \text{ else } h := h + l$ 도 정보흐름보안성을 만족한다. 즉, $L(\rho)$의 원소들은 분석 후에도 모두 public을 가진다. 그러나 $\text{if } h > 0 \text{ then } l := 1 \text{ else } l := 2$ 은 $\rho\, l = public$ 이더라도 분석 후에는 $\rho'\, l = secret$ 이다. 여기서도 조건식의 기밀 변수가 then/else 부분에서 공개변수의 배정문에 영향을 주고 있음을 인식할 수 있다. 다시 말해서 요약 해석을 이용한 분석법은 요약 해석이 종료된 후에 초기 요약 상태 함수와 최종 요약 상태 함수를 비교하여 정보흐름이 불간섭 성질을 만족하고 있는지 확인할 수 있다.

타입시스템의 경우와 마찬가지로 요약 해석에 의한 분석 결과 정보흐름보안성이 만족된다고 판단된다면 항상 불간섭 성질을 만족하는지를 증명할 수 있다. 즉, 요약 해석이 항상 안전한 근사를 하는지를 다음의 정리로부터 증명할 수 있다:

프로그램 S가 주어지고 임의의 요약 입력 상태 ρ에 대하여 $\langle S, \rho, public\rangle \rightarrow \rho'$ 이고 $L(\rho) \subseteq L(\rho')$라고 하면, 임의의 두 입력 상태 σ_1와 σ_2에 대하여 $<S, \sigma_1> \rightarrow \sigma_1'$ 이고 $<S, \sigma_2> \rightarrow \sigma_2'$ 이면 $\sigma_1 =_{public} \sigma_2$ 일 때 $\sigma_1' =_{public} \sigma_2'$ 이다.

표준적인 요약 해석의 틀 내에서 이 정리를 증명하는 것은 어렵지 않다.

Ⅵ. 정보흐름보안성의 다양성

요약해석에서 나타난 부분순서를 이용하여 정보흐름보안성을 나타낸 것은 Denning[3]에 의해 처음 등장했다. 또한 프로그램 내에 기밀성과 무결성에 대한 정보흐름 정책을 표기하고 프로그램의 모든 실행 경로가 이것을 따르는 지를 컴파일러가 검사하는 방법을 제시함으로써 정보흐름보안성이 정적 분석법에 의해 확인될 수 있다는 가능성을 보여준 것은 [4]였다.

Goguen과 Meseguer는 불간섭 정책이 정보흐름보안성을 정형화하는 방법임을 보여주었다[5]. 그러나 타입시스템과 같은 정적 분석법을 고안하고 분석법의 안전성(soundness)을 불간섭 정책을 기반으로 완전히 조화롭게 증명하게 된 것은 한참 뒤에 일이다.[8] 그 후에 함수와 상태를 포함하는 언어로 확장되었고[15,32] ML과 같은 함수 언어를 대상으로 연구가 이루어졌다.[19]

정보흐름보안성 연구의 한 가지 중요한 부분은 최신 프로그래밍언어의 표현력을 수용할 수 있는 분석법을 고안하는 일이다. 여기에는 프로시져, 함수, 예외, 오브젝트 등이 포함된다. [25]는 일계 프로시져에 대한 확장과 불간섭에 의한 타입시스템의 안전성 증명을 보여준다. Heintze와 Riecke는 람다계산법 기반의 함수 언어 SLam calculus에서 정보흐름보안성을 연구했다.[9] Zdancewic와 Myers는 first-order continuation, state, reference를 다루는 언어를 연구하고 불간섭을 보장하는 타입시스템을 고안했다.[15,32] SLam calculus에 reference, exception, polymorphic type을 확장하는 연구를 Pottier와 Simonet가 수행했다.[17,33] 예외 발생을 처리하는 언어에서는 예외 처리로 인하여 묵시적인 정보흐름이 발생한다. 이를 제어하기 위한 분석법이 제안되었다.[11,17,33] 오브젝트는 현대의 프로그래밍언어에서 매우 중요한 요소이다. JFlow 언어와 Jif 컴파일러는 Java를 확장하여 정보흐름분석을 위한 타입시스템을 구현하였다.[31] Banerjee와 Naumann도 Java와 유사한 명령형 객체지향 언어에 대하여 보안 타입시스템을 제시하고 불간섭을 증명하였다.[16]

동시언어를 위한 정보흐름보안성에 대한 연구도 활발하여 비결정성, 쓰레드 프로그램, $\pi-calculus$ 등의 주제들이 다루어졌다. 비결정성을 갖는 언어를 위한 정보흐름보안성의 고찰은 [24,28,34,14,35] 등에서 이루어졌으며 멀티쓰레드 프로그램을 대상으로 다양한 문제에 접근하였다.[11,13,14,36] 또한 동시언어의 핵심언어인 $\pi-calculus$를 대상으로 보안 타입시스템과 불간섭 성질이 연구되었다.[37,38,39]

정보흐름보안성과 다른 프로그램 분석과의 관계에 대해서도 중요한 결과들이 있다. Abadi 등은 정보흐름보안성 분석과 세 가지 종류의 프로그램 분석의 관련성을 밝혀냈다.[18] 이것들은 모두 종속성 분석이라고 할 수 있는데 $\lambda-calculus$의 변형을 위한 부분동

치관계로서 종속성을 표현하였다. 바인딩 시간 분석은 특히 정보흐름보안성과 밀접한 관계를 가진다. 부분연산(partial evaluation) 분야에서 바인딩 시간 분석은 프로그램을 정적인 부분과 동적인 부분으로 나눈다. 여기서 바인딩 시간 분석의 안전성은 정적인 부분이 동적인 부분의 영향을 받지 않는다는 것을 증명함으로써 얻어진다. 정보흐름보안성 분석과 부분연산의 연관성은 [12,14,35,40] 등에서 찾아볼 수 있다.

정보흐름보안성 정책에 대한 다양한 변형들도 연구되고 있다. 불간섭 성질이 명확하게 정보흐름보안성에 대한 정형적인 의미를 나타내기는 하지만 실용적인 측면에서는 다소 지나치게 제한적인 면을 가지고 있다. 특히 불간섭 성질은 보안 수준이 높은 곳에서 낮은 곳으로 흐르는 것을 허락하지 않기 때문에 어떤 정보가 한번 정보 수준이 높아지면 다시는 낮아질 수 없게 된다. 그러나 이러한 경우가 필연적인 상황도 있다. 즉, 기밀 정보라고 해도 암호화된 이후에는 공개적으로 전달 가능해야 한다. 또 패스워드 검사 프로그램의 경우를 살펴보자. 이 프로그램의 결과는 분명 기밀정보인 패스워드에 영향을 받는다. 즉, 패스워드가 공개적으로 입력된 데이터와 일치하는지 여부가 공개정보로서 외부에 노출되어야 한다. 불간섭 성질을 변형하여 이와 같은 보안 수준의 강등(downgrading)을 수용하도록 하는 연구가 활발하게 진행되고 있다.

Myers와 Liskov는 보안 수준의 decentralized model을 제안했는데 여기서는 선택적 강등(selective declassification)이 가능하다.[9,19,41] 이것은 프로세스 권한과 접근 주체들의 관계를 분석한 결과에 기반하여 허용된다. 보안 수준은 구조화 데이터로 표현되고 여기에서 강등을 수행할 수 있는 개체들을 표시한다. 또한 암호 프로토콜은 암호화 결과에 종속되므로 보안 수준의 강등은 필연적이다. Abadi는 암호 프로토콜 전용 언어인 spi-calculus[42]에 대하여 정보흐름보안성을 보장하는 타입시스템을 고안했다.[26]

JVM 애플릿을 위한 허용(admissibility)는 불간섭을 완화한 것이다.[29] 기밀정보의 암호화를 포함하는 온라인 지불 프로토콜을 분석할 때 필요한 허용성을 갖는지를 확인한다. 허용성은 프로그램 내에서 강등이 어떻게 허용되는 지와 같이 어떤 데이터와 어떤 데이터 사이의 종속관계가 허용되는 지를 명시적으로 나타내는 보안 정책이다.

시스템이 의도적으로 강등이 가능한 채널을 포함하는 경우에 프로그래머가 의도했던 것보다 많은 정보가 강등되는 데에 이 채널이 이용될 가능성을 배제하기 어렵다. 때문에 Zdancewic과 Myers는 채널의 오용을 막을 수 있는 robust declassification이라는 보안 정책을 제시한다.[27] 이것은 시스템 동작에 영향을 줄 수 있는 공격자(active attacker)가 시스템의 동작을 관찰 밖에 못하는 공격자(passive attacker)보다 더 많은 정보를 얻지 못하게 한다. 즉, 의도적으로 공개된 정보 흐름 이외의 정보 흐름을 보호할 수 있다.

Pierro 등이 제시한 근사 불간섭(approximate noninterference)은 확률적인 프로그램 분석법을 채용한다. 이것은 확률을 이용한 제한 조건을 프로그램하는 계산법에 대하여 정밀한(precise) 불간섭[43]과 근사 불간섭[30]을 보장할 수 있다.

불간섭의 변형은 여기서 소개하지 않은 시도들을 포함하여 모두 불간섭 성질을 좀 더 세밀하게 제어하기 위한 방법을 제시한다. Zdancewic은 변수 x의 정보가 변수 y로 전달되어서는 안된다는 불간섭 성질을 완화하여 변수 x의 정보가 3의 배수로 변환되지 않고서는 변수 y로 전달될 수 없다는 식으로 허용되는 정보흐름의 집합을 확대할 수 있는 타입시스템을 제안하였다.[44]

또한 정보흐름보안성 분석의 이해를 심화하는 노력으로서 Igarashi 등이 제안한 typed λ-calculus $\lambda^{\Box}_{s}$는 다양한 형태의 정보흐름보안성 분석법들을 수용한다.[1] 여기서 제시된 타입시스템은 Curry-Howard Isomorphism으로부터 얻어진 직관 양상 논리(intuitionistic modal logic)의 증명 시스템이다. 이것은 SLam calculus를 포함할 정도로 일반적이고 Church-Rosser나 strong normalization 같은 중요한 정리가 증명되었다. 또한 이것의 modal type은 subject reduction을 만족한다.

Ⅷ. 결 론

기존의 보안 메카니즘들은 기밀성 정책을 완전하게 구현하는 데에 한계를 가지고 있다. 접근제어, 암호화, 방화벽, 디지털서명, 바이러스 검사 등은 컴퓨팅 시스템 내의 정보 흐름을 추적하고 제어하는 데에 도움이 되지 않는다. 실행시간 감시는 특정 실행 경로만을 주시하기 때문에 불완전하다. 정보흐름 보안성은 모든 실행 경로를 고려해야만 한다. 이를 위해서 적합한 방법으로 언어기반 보안기술이 유망하다. 이것은 프로그래밍언어 시멘틱스와 정적 분석법과 같은 프로그래밍 언어 기술을 기반으로 한다.

본 논문에서는 타입시스템이나 요약해석 같은 대표

적인 정적 분석법을 이용하여 정보흐름보안성에 접근하는 기본적인 방법을 소개하였다. 이 외에도 대상 언어의 확장을 위한 방법들이 고안되었으며 방법적인 면에서도 데이터플로우 분석, 프로그램 분할(slicing), 정리 증명(theorem proving), 모델 검사(model checking)[7] 등 다양한 프로그램 분석 및 검증 기법들이 이용될 수 있다.

정보흐름보안성을 정형적으로 표현하는 것은 불간섭 성질이다. 더 복잡하고 실제적인 상황에서는 훨씬 더 다양하고 세밀한 조작이 필요한 정보흐름보안성이 존재한다. 이를 위해서 더 상세한 명세로서의 불간섭 성질도 다양하게 정의되어야 한다. 그리고 이에 따라서 정보흐름보안성 분석기법도 다양하게 고안되고 있으며 여기에 언어기반 기술이 핵심적인 역할을 하고 있다.

참 고 문 헌

[1] K. Miyamoto and A. Igarashi, "A Modal Foundation of Secure Information Flow", Proceedings of the Workshop on Foundations of Computer Security (FCS'04), pp. 187~203, July 2004.

[2] H.R.Nielson, F.Nielson, and C.Hankin, Principles of Program Analysis, Springer-Verlag, 1999

[3] D.E. Denning, "A Lattice Model of Secure Information Flow", Communications of the ACM, 19(5):236~242, 1976.

[4] D.E. Denning and P.J. Denning, Certification of programs for secure information flow, Communications of ACM, 20(7), pp. 504~513, 1977.

[5] J.A.Goguen and J.Meseguer, "Unwinding and inference control", In Proc. IEEE Symp. on Security and Privacy, pp. 75~86, 1984.

[6] P. Cousot and R. Cousot, "Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints", In Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 238~252, 1977

[7] K.-G. Doh and S.C. Shin, "Analysis of secure information flow by model checking", In Proc. of the 2nd Asian Workshop on Programming Languages and Systems, pp. 255~236, 2001.

[8] D. Volpano, G. Smith, and C. Irvine, " A sound type system for secure flow analysis,"J. Computer Security, vol. 4, no. 3, pp. 167~187, 1996.

[9] A. C. Myers and B. Liskov, "A decentralized model for informationflow control,"in Proc. ACM Symp. on Operating System Principles, Oct. 1997, pp. 129~142.

[10] N. Heintze and J. G. Riecke, "The SLam calculus: programming with secrecy and integrity," in Proc. ACM Symp. on Principles of Programming Languages, Jan. 1998, pp. 365~377.

[11] G. Smith and D. Volpano, "Secure information flow in a multithreaded imperative language," in Proc. ACM Symp. on Principles of Programming Languages, Jan. 1998, pp. 355~364.

[12] G. Barthe and B. Serpette, "Partial evaluation and non-interference for object calculi," in Proc. FLOPS. Nov. 1999, vol. 1722 of LNCS, pp. 53~57, Springer-Verlag.

[13] D. Volpano and G. Smith, "Probabilistic noninterference in a concurrent language," J. Computer Security, vol. 7, no. 2-, pp. 231~253, Nov. 1999.

[14] A. Sabelfeld and D. Sands, "Probabilistic noninterference for multithreaded programs,"in Proc. IEEE Computer Security Foundations Workshop, pp. 200~214, July 2000

[15] S. Zdancewic and A. C. Myers, "Secure information flow and CPS," in Proc. European Symposium on Programming. Apr. 2001, vol. 2028 of LNCS, pp. 46~61, Springer-Verlag.

[16] A. Banerjee and D. A. Naumann, "A secure information flow and pointer confinement in a Java-like language," in Proc. IEEE Computer Security Foundations Workshop, pp. 253~267, June 2002

[17] F. Pottier and V. Simonet, "Information flow inference for ML," in Proc. ACM Symp. on Principles of Programming Languages, pp. 319~330, Jan. 2002

[18] M. Abadi, A. Banerjee, N. Heintze, and J. Riecke, "A core calculus of dependency,"in Proc. ACM Symp. on Principles of Programming Languages, pp. 147~160, Jan. 1999

[19] F. Pottier and S. Conchon, "Information flow inference for free," in Proc. ACM International Conference on Functional Programming, pp. 46~57, Sept. 2000

[20] A. Sabelfeld and D. Sands, "A per model of secure information flow in sequential programs," Higher Order and Symbolic Computation, vol.14, no. 1, pp. 59~1, Mar. 2001.

[21] E. S. Cohen, "Information transmission in computational systems," ACM SIGOPS Operating Systems Review, vol. 11, no. 5, pp. 133~139, 1977.

[22] E. S. Cohen, "Information transmission in sequential programs," in Foundations of Secure Computation, R. A. DeMillo, D. P. Dobkin, A. K. Jones, and R. J. Lipton, Eds., pp. 297~335. Academic Press, 1978.

[23] J. McLean, "Proving noninterference and functional correctness using traces," J. Computer Security, vol. 1, no. 1, pp. 37~48, 1992.

[24] J.-P. Banˆtre, C. Bryce, and D. Le M´etayer, "Compile-time detection of information flow in sequential programs," in Proc. European Symp. on Research in Computer Security, vol. 875 of LNCS, pp. 55~63, Springer-Verlag, 1994

[25] D. Volpano and G. Smith, "A type-based approach to program security," in Proc. TAPSOFT'7, vol. 1214 of LNCS, pp. 607~621, Apr. 1997

[26] M. Abadi, "Secrecy by typing in security protocols," in Proc. Theoretical Aspects of Computer Software, pp. 611~38, Sept. 1997

[27] S. Zdancewic and A. C. Myers, "Robust declassification," in Proc. IEEE Computer Security Foundations Workshop, pp. 15~23, June 2001

[28] R. Joshi and K. R. M. Leino, "A semantic approach to secure information flow," Science of Computer Programming, vol. 37, no.1, pp. 113~138, 2000.

[29] M. Dam and P. Giambiagi, "Confidentiality for mobile code: The case of a simple payment protocol,"in Proc. IEEE Computer Security Foundations Workshop, pp. 233~44, July 2000

[30] A. Di Pierro, C. Hankin, and H. Wiklicky, "Approximate noninterference," in Proc. IEEE Computer Security Foundations Workshop, pp. 1~7, 2002

[31] A.C.Myers, "Flow: Practical mostly-static information flow control," in Proc. ACM Symp. on Principles of Programming Languages, pp. 228~241, 1999

[32] S. Zdancewic and A. C. Myers, "Secure information flow via linear continuations," Higher Order and Symbolic Computation, vol. 15, no.2, pp. 209~234, Sept. 2002.

[33] F. Pottier and V. Simonet, "Information flow inference for ML,"ACM TOPLAS, Volume 25, Issue 1, pp. 117~158, January 2003

[34] K. R. M. Leino and R. Joshi, "A semantic approach to secure infor-

mation flow," in Proc. Mathematics of Program Construction, vol. 1422 of LNCS, pp. 254~271, June 1998

[35] A. Sabelfeld and D. Sands, "A per model of secure information flow in sequential programs," in Proc. European Symposium on Programming, vol. 1576 of LNCS, pp. 40~58, Springer-Verlag, Mar. 1999

[36] D. Volpano and G. Smith, "Probabilistic noninterference in a concurrent language," in Proc. IEEE Computer Security Foundations Workshop, pp. 34~3, June 1998

[37] K. Honda, V. Vasconcelos, and N. Yoshida, "Secure information flow as typed process behaviour," in Proc. European Symposium on Programming, vol. 1782 of LNCS, pp. 180~99, Springer-Verlag, 2000

[38] K. Honda and N. Yoshida, "An uniform type structure for secure information flow," in Proc. ACM Symp. on Principles of Programming Languages, pp. 81~2, Jan. 2002

[39] F. Pottier, "A simple view of type-secure information flow in the picalculus," in Proc. IEEE Computer Security Foundations Workshop, pp. 320~30, June 2002

[40] P. Thiemann, "Enforcing security properties by type specialization," in Proc. European Symposium on Programming, vol. 2028 of LNCS, Springer-Verlag, Apr. 2001

[41] A. C. Myers and B. Liskov, "Complete, safe information flow with decentralized labels," in Proc. IEEE Symp. on Security and Privacy, pp. 186~97, May 1998

[42] M. Abadi and A. D. Gordon, "A calculus for cryptographic protocols: The Spi calculus," Information and Computation, vol. 148, no. 1, pp. 1~0, Jan. 1999.

[43] A. Di Pierro, C. Hankin, and H. Wiklicky, "Probabilistic confinement in a declarative framework," in Declarative Programming-elected papers from AGP 2000, vol. 48 of Electronic Notes in Theoretical Computer Science, Elsevier, 2001

[44] Peng Li and Steve Zdancewic, "Downgrading Policies and Relaxed Noninterference," In Proc. 32nd ACM Symp. on Principles of Programming Languages (POPL), pages 158~170, January 2005.

〈著者紹介〉

신 승 철 (Seungcheol Shin)
정회원
1996년: 인하대학교 공학박사
1996년~2006년: 동양대 부교수
1999년~2000년: 캔사스 주립대학교 연구원
2006년 3월~현재: 한국기술교육대학교 인터넷미디어공학부 조교수
관심분야 : 프로그래밍언어, 프로그램분석 및 검증, 소프트웨어보안, 수리논리