SE547: Lecture 10

Overview

Information flow

Sequential noninterference

Timing channels

Concurrent noninterference

Declassification

JIF

Information flow

What is *Mandatory Access Control*?

What is a *security level*? Simple security levels: *H* and *L*.

Tag data and variables with security levels. Examples of *information flow* between security levels... Is x_H=y_L (write up) OK? Is y_L=x_H (write down) OK?

(Note: in this world, high-security programs have fewer rights than low-security programs! Spot the military funding...)

What is a covert channel? Examples? Can we completely eliminate covert channels?

Information flow

Note that Java protection doesn’t help!

```
class Example {
  private int xH;
  public int yL;
  public void leak () {
    yL = xH; // Write down is OK according to Java.
  }
}
```

Java doesn’t stop values *escaping* from their static protection.

Sequential noninterference

Informal idea behind noninterference (make this formal later):

> A program P satisfies *noninterference* if the contents of low security variables after running P is independent of the contents of high security variables before running P

Which of the following satisfy noninterference?

1. x_H=y_L;
2. y_L=x_H;
3. x_H=0; if(even(y_L)) { x_H=1; }
4. y_L=0; if(even(x_H)) { y_L=1; }

Sequential noninterference

Formal model of commands...

c, d ::=
  $x = e;$
  $c\ d$
  if (e) { c } else { d }
  while (e) { c }

and expressions...

e, f ::=
  x
  n
  $e + f$
  $e == f$
  ...

This is Smith and Volpano’s model, dressed up in Java syntax.

Sequential noninterference

A *memory* μ is a mapping from variables to integers, for example:

$\{ x = 37, y = 5 \}$

Write $\mu(e)$ for the result of evaluating expression e in memory μ. For example, in the above memory, what are:

$\mu(x + 1)$
$\mu(x == y)$

Write $\mu[x{:=}n]$ for the result of updating memory μ so x now contains n. For example, in the above memory, what are:

$\mu[y := \mu(x + 1)]$
$\mu[y := \mu(x == y)]$

Sequential noninterference

Model of commands $(c_1, \mu_1) \to (c_2, \mu_2)$:

1. If $(c_1, \mu_1) \to \mu_2$ then $(c_1 d, \mu_1) \to (d, \mu_2)$

2. If $(c_1, \mu_1) \to (c_2, \mu_2)$ then $(c_1 d, \mu_1) \to (c_2 d, \mu_2)$

3. If $\mu(e)$ is nonzero then (if (e) $\{c\}$ else $\{d\}$, $\mu) \to (c, \mu)$

4. If $\mu(e)$ is zero then (if (e) $\{c\}$ else $\{d\}$, $\mu) \to (d, \mu)$

5. What about while?

Model of commands final step $(c_1, \mu_1) \to \mu_2$:

6. $(x{=}e, \mu) \to \mu[x{:=}\mu(e)]$

7. What about while?

Example:

x:=1; while(x) { x=x-1; }

Sequential noninterference

Write $\mu \sim \rho$ whenever μ and ρ agree on L variables:

1. $\{ x_H = 37, y_L = 5 \} \sim \{ x_H = 0, y_L = 5 \}$?
2. $\{ x_H = 37, y_L = 5 \} \sim \{ x_H = 37, y_L = 0 \}$?

A command c satisfies *sequential noninterference* when:

If $\mu_1 \sim \rho_1$ and $(c, \mu_1) \rightarrow^* \mu_2$ and $(c, \rho_1) \rightarrow^* \rho_2$ then $\mu_2 \sim \rho_2$.

Which of these satisfy sequential noninterference?

1. x_H=y_L;
2. y_L=x_H;
3. x_H=0; if(*even*(y_L)) { x_H=1; }
4. y_L=0; if(*even*(x_H)) { y_L=1; }

Sequential noninterference

Static analysis for noninterference satisfying:

1. In $x = e$, if e reads from an H variable, then x is an H variable.

2. In if (e) { c } else { d }, if e reads from an H variable, then c and d only write to H variables.

3. In while (e) { c }, if e reads from an H variable, then c only writes to H variables.

Which of these pass static analysis?

1. x_H=y_L;
2. y_L=x_H;
3. x_H=0; if($even$(y_L)) { x_H=1; }
4. y_L=0; if($even$(x_H)) { y_L=1; }

Sequential noninterference

Payoff from static analysis:

> If *c* passes static analysis then *c* satisfies sequential noninterference.

Timing channels

Adding a clock to the language can break noninterference:

```
finish$_L$ = System.getTimeMillis () + 1000;
while (secret$_H$ && System.getTimeMillis () $<$ finish$_L$) { }
if (System.getTimeMillis () $<$ finish$_L$) { result$_L$ = 0; }
else { result$_L$ = 1; }
```

What happens in initial memory { secret$_H$ = 0, *everything else* = 0 }?

What happens in initial memory { secret$_H$ = 1, *everything else* = 0 }?

Timing channels

State-of-the-art: make the clock high security. Not very practical!

Concurrent noninterference

Noninterference for concurrent programs is much harder. Thread α:

```
while (trigger0_H == 0) {}  result_L = 1;
trigger1_H++
```

Thread β:

```
while (trigger1_H == 0) {} result_L = 0;
trigger0_H++;
```

Thread γ:

```
if (secret_H == 0) { trigger0_H++; } else { trigger1_H++; }
```

What happens in initial memory { $\text{secret}_H = 0$, *everything else* = 0 }?

What happens in initial memory { $\text{secret}_H = 1$, *everything else* = 0 }?

Smith and Volpano scale this up to leak an *n*-bit PIN, not just one bit.

Concurrent noninterference

State-of-the-art: ban high-security while loops. Not very practical!

Also relies on a nondeterministic scheduler, otherwise timing channels are possible.

Declassification

Mandatory Access Control is usually too restrictive in practice: we need some way to *declassify* data.

```
booleanL checkPwd (StringL uid, StringL guess, StringH pwd) {
  return (guess == pwd); // Does not pass static analysis!
}
```

Declassification requires *authority,* either granted directly:

```
booleanL checkPwd (StringL uid, StringL guess, StringH pwd)
where authority(H) {
  return declassify(guess == pwd, L);
}
```

or granted to the calling method and *delegated*:

```
booleanL checkPwd (StringL uid, StringL guess, StringH pwd)
where caller(H) {
  return declassify(guess == pwd, L);
}
```

JIF

Java Information Flow http://www.cs.cornell.edu/jif/ implements static analysis for information flow.

Checks for sequential noninterference, *not* timing channels or concurrent noninterference.

Is this OK (Test1.jif):

```
public int{Alice:} run (int{Alice:} x) {
  return x;
}
```

Is this OK (Test2.jif):

```
public int{Bob:} run (int{Alice:} x) {
  return x;
}
```

JIF

Is this OK (Test3.jif):

```
public int{} run (int{Alice:} x) {
  return x;
}
```

Is this OK (Test4.jif):

```
public int{Alice:} run (int{} x) {
  return x;
}
```

JIF

Is this OK (Test5.jif):

```
public int{Bob:} run (int{Alice:} x)
where authority(Alice) {
  return declassify (x, {Bob:});
}
```

Is this OK (Test6.jif):

```
public int{Bob:} run (int{Alice:} x) {
  return declassify (x, {Bob:});
}
```

Is this OK (Test7.jif):

```
public int{Bob:} run (int{Alice:} x)
where caller(Alice) {
  return declassify (x, {Bob:});
}
```

Summary

Information flow tracks the security levels of data.

Goal of information flow is to ensure noninterference: values in low security variables cannot depend on values in high security variables.

Static analysis of sequential code without timers is possible.

Static analysis of concurrent code or code with timers is much harder.

Practical information flow systems include declassification.

Next week

Final exam.