50.042 FCS Summer 2024
Lecture 19 – Information Flow II

Felix LOH
Singapore University of Technology and Design

SINGAPORE UNIVERSITY OF
TECHNOLOGY AND DESIGN

With selected materials adapted from: *Computer Security: Art and Science, by M. Bishop (2nd Edition)*

A more precise definition of information flow

- From the last lecture, we know how to determine whether a system is secure
- So how do we develop mechanisms to detect and stop flows of information that violate a security policy?

- First, let's <u>precisely</u> define information flow, then we can discuss mechanisms to detect and stop flows of information that violate a security policy

A more precise definition of information flow

- Information flow can be precisely defined by utilizing the concept of *entropy*
  - The notion of *entropy* falls under the field of *information theory*
  - This mathematical field was established by the works of Harry Nyquist and Ralph Hartley in the 1920s, and Claude Shannon in the 1940s

Discrete random variables and expectation

- Before we discuss the definition of entropy, let's talk about some basic probability theory concepts

- Let *X* be a discrete random variable, with *X* having some probability of taking one of the values x_1, x_2, ... , x_n
- Note that $\sum_{i=1}^{n} \mathrm{P}(X = x_i) = 1$

- Then the *expected* value of *X* (a.k.a. the *expectation* of X) is given by
- $\mathrm{E}(X) = \sum_{i=1}^{n} \mathrm{P}(X = x_i) \cdot x_i$

Discrete random variables and expectation

- Let's look at an example…

- Let X denote the outcome of a fair coin toss
- There are two possible outcomes for X:
  - x_1 = 'Heads' = 1, with $P(X = x_1) = 0.5$
  - x_2 = 'Tails' = 0, with $P(X = x_2) = 0.5$

- Then $E(X) = P(X = x_1) \cdot x_1 + P(X = x_2) \cdot x_2 = (0.5) \cdot 1 + (0.5) \cdot 0 = 0.5$
  - Intuitively, we can view the expectation of X as the average value of X after conducting many trials (in this case, many coin tosses)

Information of a particular outcome of a discrete random variable

- The *information* (or *information content*) of a particular outcome x_i of a discrete random variable X is denoted $\mathrm{I}_X(x_i)$, or $\mathrm{I}(X = x_i)$
- It measures the amount of information received/learned in <u>bits</u>, when the outcome of X is x_i, as shown by the following equation:

$$\mathrm{I}(X = x_i) = \log_2 \frac{1}{\mathrm{P}(X = x_i)}$$

- The amount of information received, by learning that that outcome of X is x_i, is <u>inversely</u> proportional to the probability of x_i occurring
  - Recall that you learned this in 50.002 (Computation Structures)

Information of a particular outcome of a discrete random variable – example

- Let's go back to our example for X – outcome of a fair coin toss…
- Two possible outcomes for X:
  - x_1 = 'Heads' = 1, with $P(X = x_1) = 0.5$
  - x_2 = 'Tails' = 0, with $P(X = x_2) = 0.5$

- Then $\mathrm{I}(X = x_1) = \log_2 \frac{1}{\mathrm{P}(X=x_1)} = \log_2 \frac{1}{0.5} = 1$
- Likewise, $\mathrm{I}(X = x_2) = 1$
- We gain 1 bit of information by learning that the outcome of X is 'Heads' (or 'Tails')

Entropy of a discrete random variable

- The *entropy* of a discrete random variable *X* is the <u>un</u>certainty of *X*, measured in bits

- The entropy of *X* is denoted as H(*X*), and we can think of H(*X*) as the *expectation* of the *information* of *X*:

$$\mathrm{H}(X) = \sum_{i=1}^{n} \mathrm{P}(X = x_i) \cdot \mathrm{I}(X = x_i) = \sum_{i=1}^{n} \mathrm{P}(X = x_i) \cdot \log_2 \frac{1}{\mathrm{P}(X=x_i)}$$

- Thus, we have the following formal definition for the entropy of *X*:

$$\mathrm{H}(X) = -\sum_{i=1}^{n} \mathrm{P}(X = x_i) \cdot \log_2 \mathrm{P}(X = x_i)$$

Entropy of a discrete random variable – example

- Again, let's go back to our example for X – the outcome of a fair coin toss…
- Two possible outcomes for X:
  - x_1 = 'Heads' = 1, with $P(X = x_1) = 0.5$
  - x_2 = 'Tails' = 0, with $P(X = x_2) = 0.5$

- Then $\mathrm{H}(X) = -\mathrm{P}(X = x_1) \cdot \log_2 \mathrm{P}(X = x_1) - \mathrm{P}(X = x_2) \cdot \log_2 \mathrm{P}(X = x_2) = -(0.5) \cdot \log_2 0.5 - (0.5) \cdot \log_2 0.5 = 1$

Entropy of a discrete random variable

- The entropy of a variable is <u>inversely proportional</u> to its predictability
  - In other words, the lower the entropy of some variable, the more predictable that variable is
  - If $\mathrm{H}(X) = 0$, then *X* is completely predictable (it is always a particular value)
  - If $\mathrm{H}(X) = \infty$, then *X* is completely <u>un</u>predictable

- If the entropy of some variable *X'* is lower than the entropy of some variable *X*, then we can say that *X'* is more predictable than *X*
  - We can also say that we know <u>more</u> about *X'* than *X*

- Let's illustrate this with an example

Entropy of a discrete random variable – another example

- Let X' denote the outcome of a biased coin toss
- There are two possible outcomes for X':
  - x_1' = 'Heads' = 1, with $P(X' = x_1') = 0.75$ → this event is more likely to occur
  - x_2' = 'Tails' = 0, with $P(X' = x_2') = 0.25$

- $E(X') = P(X' = x_1') \cdot x_1' + P(X' = x_2') \cdot x_2' = (0.75) \cdot 1 + (0.25) \cdot 0 = 0.75$
- $\mathrm{I}(X' = x_1') = \log_2 \frac{1}{\mathrm{P}(X'=x_1')} = \log_2 \frac{1}{0.75} = 0.415$
- $\mathrm{I}(X' = x_2') = \log_2 \frac{1}{\mathrm{P}(X'=x_2')} = \log_2 \frac{1}{0.25} = 2$

Entropy of a discrete random variable – another example

- $\mathrm{H}(X') = -\mathrm{P}(X' = x_1{}') \cdot \log_2 \mathrm{P}(X' = x_1{}') - \mathrm{P}(X' = x_2{}') \cdot \log_2 \mathrm{P}(X' = x_2{}') = -(0.75) \cdot \log_2 0.75 - (0.25) \cdot \log_2 0.25 = 0.811$

- Compare this with $\mathrm{H}(X) = 1$
- We can see that $\mathrm{H}(X') < \mathrm{H}(X)$
- This makes sense, because *X'* is <u>more predictable</u> than *X*, as *X'* is more likely to be 'Heads' than 'Tails', whereas *X* has an equal likelihood of being 'Heads' or 'Tails'
- So *X'* has a <u>lower</u> entropy than *X*

Conditional entropy

- Let X and Y be a discrete random variables, with X having some probability of taking one of the values $x_1, x_2, \ldots, x_n$ and Y having some probability of taking one of the values $y_1, y_2, \ldots, y_m$

- $\sum_{i=1}^{n} \mathrm{P}(X = x_i) = 1$
- $\sum_{j=1}^{m} \mathrm{P}(Y = y_j) = 1$

- Then the *conditional* entropy of X given that $Y = y_j$ is:

  $$\mathrm{H}(X \mid Y = y_j) = -\sum_{i=1}^{n} \mathrm{P}(X = x_i \mid Y = y_j) \cdot \log_2 \mathrm{P}(X = x_i \mid Y = y_j)$$

Conditional entropy

- Also, the *conditional* entropy of *X* given *Y* is:

  $\mathrm{H}(X \mid Y) = -\sum_{j=1}^{m} \mathrm{P}\left(Y = y_j\right) \cdot \sum_{i=1}^{n} \mathrm{P}\left(X = x_i \mid Y = y_j\right) \cdot \log_2 \mathrm{P}\left(X = x_i \mid Y = y_j\right)$

- We will use these definitions to develop the notion of information flow in a system, by using *X* and *Y* to model objects in a system
- The basic idea is that information flows from an object *X* to an object *Y*, if the execution of a sequence of commands *c** causes information initially in *X* to affect the information in *Y*

Entropy and information flow

- Let c^* be a sequence of commands that take a system (an FSM) from state a to state b

- Let X and Y be objects in the system, and X_a and Y_a be the values assigned to X and Y respectively at state a
  - Think of X_a and Y_a as outcomes (which can take on particular values) at state a, so treat X_a and Y_a as discrete random variables
  - Same for X_b and Y_b – discrete random variables assigned to X and Y respectively at state b
  - Think of X and Y as more like "containers"

Entropy and information flow

- c^* is a sequence of commands that take a system (an FSM) from state a to state b
- X and Y are objects in the system, and X_a and Y_a are the values assigned to X and Y respectively at state a (same for state b: X_b and Y_b)

$y_a = f(x_a)$ $\quad y_b = f(x_b) \leftarrow$ more predictable

- Then the command sequence c^* causes a <u>flow of information from X to Y</u>, if: $\mathrm{H}(X_a \mid Y_b) < \mathrm{H}(X_a \mid Y_a)$

more predictable

- If Y_a is non-existent in state a, then there is a <u>flow of information from X to Y</u>, if: $\mathrm{H}(X_a \mid Y_b) < \mathrm{H}(X_a)$

more predictable

Entropy and information flow – example

- Suppose the command sequence $c*$ is defined by the following code:

```
if x == 1 then y := 0
          else y := 1;
```

  with **x** equally likely to have been assigned either 0 or 1
- This implies that **y** is equally likely to be assigned either 1 or 0
- a is the state before $c*$ is executed, while b is the state after $c*$ is executed
- Note that Y_a does not exist in state a → y don't exist before code executes.
- We have $\mathrm{H}(X_a) = 1$ (using our earlier example of a fair coin toss)

Entropy and information flow – example

$y_1 = 0 \quad y_2 = 1$

$x_a = 0/1$

- Now, $\mathrm{H}(X_a \mid Y_b) = -\sum_{j=1}^{2} \mathrm{P}\big(Y_b = y_j\big) \cdot \sum_{i=1}^{2} \mathrm{P}\big(X_a = x_i \mid Y_b = y_j\big) \cdot \log_2 \mathrm{P}\big(X_a = x_i \mid Y_b = y_j\big)$

$y_j = 0,$

$$= -\underbrace{\mathrm{P}(Y_b = 0)}^{0.5}[\overbrace{\mathrm{P}(X_a = 0 \mid Y_b = 0)}^{0} \cdot \log_2 \overbrace{\mathrm{P}(X_a = 0 \mid Y_b = 0)}^{0} + \underbrace{\mathrm{P}(X_a = 1 \mid Y_b = 0)}_{1} \cdot \log_2 \underbrace{\mathrm{P}(X_a = 1 \mid Y_b = 0)}_{1}]$$

$0.5[0 \cdot \log_2 0 + 1 \log_2 1]$

$-0.5[1 \log_2 1 + 0 \log_2 0]$

$y_j = 1$

$$-\overbrace{\mathrm{P}(Y_b = 1)}^{0.5}[\overbrace{\mathrm{P}(X_a = 0 \mid Y_b = 1)}^{1} \cdot \log_2 \underbrace{\mathrm{P}(X_a = 0 \mid Y_b = 1)}_{1} + \underbrace{\mathrm{P}(X_a = 1 \mid Y_b = 1)}_{0} \cdot \log_2 \underbrace{\mathrm{P}(X_a = 1 \mid Y_b = 1)}_{0}] = 0$$

$$= -0.5[0 \cdot \log_2 0 + 1 \cdot \log_2 1] - 0.5[1 \cdot \log_2 1 + 0 \cdot \log_2 0]$$

Entropy and information flow – example

- In information theory, we define $0 \cdot \log_2 0 = 0$,

  so $\mathrm{H}(X_a \mid Y_b) = -0.5[0+0] - 0.5[0+0] = 0$

- Intuitively, we can see that $\mathrm{H}(X_a \mid Y_b) = 0$, because if Y_b is 1, then X_a **must** be 0, and if Y_b is 0, then X_a **must** be 1; and thus X_a is completely predictable given Y_b; and the conditional entropy of X_a given Y_b is 0

Entropy and information flow – example

- $\mathrm{H}(X_a) = 1$
- $\mathrm{H}(X_a \mid Y_b) = 0$

- So we have $\mathrm{H}(X_a \mid Y_b) < \mathrm{H}(X_a)$

- Therefore, information has flowed from *X* to *Y*
  - Intuitively, this makes sense because the value that was originally assigned to **`x`** directly affects the value that would be assigned to **`y`**

Entropy and information flow – example 2

- Suppose the command sequence c^* is defined by the following code:

```
x := y + z;
```

with **y** <u>equally likely</u> to have been assigned the integer values 0 or 1, (i.e. just like the outcome of a fair coin toss)

and **z** having been assigned:

- integer value 1 – probability of $^1/_2$
- integer value 2 – probability of $^1/_4$
- integer value 3 – probability of $^1/_4$

$P(z=1)=\frac{1}{2}$

$P(z=2)=\frac{1}{4}$

$P(z=3)=\frac{1}{4}$

- a is the state <u>before</u> c^* is executed, while b is the state <u>after</u> c^* is executed

Entropy and information flow – example 2

- X_a does not exist in state a, so $\mathrm{H}(Y_a \mid X_a) = \mathrm{H}(Y_a)$
- We have: $\mathrm{H}(Y_a) = \mathrm{H}(Y_b) = -2 \cdot \left(\frac{1}{2} \cdot \log_2 \frac{1}{2}\right) = 1$

$H(Y_a) = -\sum_{i=1}^{2} P(Y=y_i) \cdot \log_2$

$= -(0.5 \log(0.5) + 0.5 \log 0.5)$

$= -\log 0.5 = 1$

- Now, based on the information in this example, X_b can take on one of **four** different values:
  - integer value 1 – probability of $^1/_4$ $\Rightarrow 0.5 \times 0.5 = \frac{1}{4}$
  - integer value 2 – probability of $^3/_8$ $0+2,\ 1+1,\quad 0.5 \times \frac{1}{4} + 0.5 \times 0.5 = \frac{1}{8} + \frac{2}{8} = \frac{3}{8}$
  - integer value 3 – probability of $^1/_4$ $0+3,\ 1+2,\quad 0.5 \times \frac{1}{4} + 0.5 \times \frac{1}{4} = \frac{1}{4}$
  - integer value 4 – probability of $^1/_8$ $1+3 = 0.5 \times \frac{1}{4} = \frac{1}{8}$

- Now we need to determine: $\mathrm{H}(Y_a \mid X_b) =$

$$-\sum_{j=1}^{4} \mathrm{P}(X_b = x_j) \cdot \sum_{i=1}^{2} \mathrm{P}(Y_a = y_i \mid X_b = x_j) \cdot \log_2 \mathrm{P}(Y_a = y_i \mid X_b = x_j)$$

Entropy and information flow – example 2

- $\mathrm{H}(Y_a \mid X_b) =$
  $-\sum_{j=1}^{4} \mathrm{P}\big(X_b = x_j\big) \cdot \sum_{i=1}^{2} \mathrm{P}\big(Y_a = y_i \mid X_b = x_j\big) \cdot \log_2 \mathrm{P}\big(Y_a = y_i \mid X_b = x_j\big)$

- Evaluating the above expression is going to get ugly, we’ll go through it bit by bit – hang in there…

Entropy and information flow – example 2

- $\mathrm{H}(Y_a \mid X_b) = -\sum_{j=1}^{4} \mathrm{P}\big(X_b = x_j\big) \cdot \sum_{i=1}^{2} \mathrm{P}\big(Y_a = y_i \mid X_b = x_j\big) \cdot \log_2 \mathrm{P}\big(Y_a = y_i \mid X_b = x_j\big)$

  $= -\mathrm{P}(X_b = 1)[\mathrm{P}(Y_a = 0 \mid X_b = 1) \cdot \log_2 \mathrm{P}(Y_a = 0 \mid X_b = 1) + \mathrm{P}(Y_a = 1 \mid X_b = 1) \cdot \log_2 \mathrm{P}(Y_a = 1 \mid X_b = 1)]$

  $-\mathrm{P}(X_b = 2)[\mathrm{P}(Y_a = 0 \mid X_b = 2) \cdot \log_2 \mathrm{P}(Y_a = 0 \mid X_b = 2) + \mathrm{P}(Y_a = 1 \mid X_b = 2) \cdot \log_2 \mathrm{P}(Y_a = 1 \mid X_b = 2)]$

  $-\mathrm{P}(X_b = 3)[\mathrm{P}(Y_a = 0 \mid X_b = 3) \cdot \log_2 \mathrm{P}(Y_a = 0 \mid X_b = 3) + \mathrm{P}(Y_a = 1 \mid X_b = 3) \cdot \log_2 \mathrm{P}(Y_a = 1 \mid X_b = 3)]$

  $-\mathrm{P}(X_b = 4)[\mathrm{P}(Y_a = 0 \mid X_b = 4) \cdot \log_2 \mathrm{P}(Y_a = 0 \mid X_b = 4) + \mathrm{P}(Y_a = 1 \mid X_b = 4) \cdot \log_2 \mathrm{P}(Y_a = 1 \mid X_b = 4)]$

- Now let’s evaluate each conditional probability term before getting back to this…

Entropy and information flow – example 2

- In the expression, notice that we need to evaluate several *conditional* probability terms: $\mathrm{P}\left(Y_a = y_i \mid X_b = x_j\right)$ for some *i* and *j*
  - One major problem: these terms require us to find the probability of *Y*, given that *X* is already some particular outcome/value
  - But the value of *X* is dependent on the value of *Y*!
  - It can be done, but that requires us to reason deductively (not that easy)

- Fortunately, there is a more straightforward way
- We can use Baye's theorem to compute $\mathrm{P}(Y \mid X)$ in terms of $\mathrm{P}(X \mid Y)$:

$$\mathrm{P}(Y \mid X) = \frac{\mathrm{P}(X \mid Y) \cdot \mathrm{P}(Y)}{\mathrm{P}(X)}$$

Entropy and information flow – example 2

- So using Baye's theorem, let's compute the various values of $\mathrm{P}\left(Y_a = y_i \mid X_b = x_j\right)$:

- $\mathrm{P}(Y_a = 0 \mid X_b = 1) = \dfrac{\mathrm{P}(X_b=1 \mid Y_a=0) \cdot \mathrm{P}(Y_a=0)}{\mathrm{P}(X_b=1)} = \dfrac{\left(\frac{1}{2}\right) \cdot \left(\frac{1}{2}\right)}{\left(\frac{1}{4}\right)} = 1$

- $\mathrm{P}(Y_a = 1 \mid X_b = 1) = \dfrac{\mathrm{P}(X_b=1 \mid Y_a=1) \cdot \mathrm{P}(Y_a=1)}{\mathrm{P}(X_b=1)} = \dfrac{(0) \cdot \left(\frac{1}{2}\right)}{\left(\frac{1}{4}\right)} = 0$

Entropy and information flow – example 2

- $\mathrm{P}(Y_a = 0 \mid X_b = 2) = \frac{\mathrm{P}(X_b{=}2 \mid Y_a{=}0) \cdot \mathrm{P}(Y_a{=}0)}{\mathrm{P}(X_b{=}2)} = \frac{\left(\frac{1}{4}\right) \cdot \left(\frac{1}{2}\right)}{\left(\frac{3}{8}\right)} = \frac{1}{3}$

- $\mathrm{P}(Y_a = 1 \mid X_b = 2) = \frac{\mathrm{P}(X_b{=}2 \mid Y_a{=}1) \cdot \mathrm{P}(Y_a{=}1)}{\mathrm{P}(X_b{=}2)} = \frac{\left(\frac{1}{2}\right) \cdot \left(\frac{1}{2}\right)}{\left(\frac{3}{8}\right)} = \frac{2}{3}$

- $\mathrm{P}(Y_a = 0 \mid X_b = 3) = \frac{\mathrm{P}(X_b{=}3 \mid Y_a{=}0) \cdot \mathrm{P}(Y_a{=}0)}{\mathrm{P}(X_b{=}3)} = \frac{\left(\frac{1}{4}\right) \cdot \left(\frac{1}{2}\right)}{\left(\frac{1}{4}\right)} = \frac{1}{2}$

- $\mathrm{P}(Y_a = 1 \mid X_b = 3) = \frac{\mathrm{P}(X_b{=}3 \mid Y_a{=}1) \cdot \mathrm{P}(Y_a{=}1)}{\mathrm{P}(X_b{=}3)} = \frac{\left(\frac{1}{4}\right) \cdot \left(\frac{1}{2}\right)}{\left(\frac{1}{4}\right)} = \frac{1}{2}$

Entropy and information flow – example 2

- $\mathrm{P}(Y_a = 0 \mid X_b = 4) = \frac{\mathrm{P}(X_b=4 \mid Y_a=0) \cdot \mathrm{P}(Y_a=0)}{\mathrm{P}(X_b=4)} = \frac{(0) \cdot \left(\frac{1}{2}\right)}{\left(\frac{1}{8}\right)} = 0$

- $\mathrm{P}(Y_a = 1 \mid X_b = 4) = \frac{\mathrm{P}(X_b=4 \mid Y_a=1) \cdot \mathrm{P}(Y_a=1)}{\mathrm{P}(X_b=4)} = \frac{\left(\frac{1}{4}\right) \cdot \left(\frac{1}{2}\right)}{\left(\frac{1}{8}\right)} = 1$

Entropy and information flow – example 2

- So we have $\mathrm{H}(Y_a \mid X_b) =$

$$-\frac{1}{4}\cdot[1\cdot\log_2 1 + 0\cdot\log_2 0]$$
$$-\frac{3}{8}\cdot\left[\frac{1}{3}\cdot\log_2\frac{1}{3} + \frac{2}{3}\cdot\log_2\frac{2}{3}\right]$$
$$-\frac{1}{4}\cdot\left[\frac{1}{2}\cdot\log_2\frac{1}{2} + \frac{1}{2}\cdot\log_2\frac{1}{2}\right]$$
$$-\frac{1}{8}\cdot[0\cdot\log_2 0 + 1\cdot\log_2 1]$$

$$= -\frac{3}{8}\cdot\left[\frac{1}{3}\cdot\log_2\frac{1}{3} + \frac{2}{3}\cdot\left(1 + \log_2\frac{1}{3}\right)\right] - \frac{1}{4}\cdot\left[\log_2\frac{1}{2}\right]$$

Entropy and information flow – example 2

$$= -\frac{3}{8} \cdot \log_2 \frac{1}{3} - \frac{1}{4} + \frac{1}{4}$$
$$= -\frac{3}{8} \cdot \log_2 \frac{1}{3}$$

- Thus, we have $\mathrm{H}(Y_a \mid X_b) = -\frac{3}{8} \cdot \log_2 \frac{1}{3} = 0.594$
- Now, $\mathrm{H}(Y_a) = 1$

more predictable

if we know about X, Y more predictable, Y affects X

- $\mathrm{H}(Y_a \mid X_b) < \mathrm{H}(Y_a)$
- So information has flowed from *Y* to *X*

Entropy and information flow – example 2

- As an extra practice on your own, try calculating $\mathrm{H}(Z_a)$ and $\mathrm{H}(Z_a \mid X_b)$
- Then determine whether information has flowed from *Z* to *X*

Implicit flows of information

- Implicit flows are flows of information from *X* to *Y* without an *explicit* assignment of the form `y := f(x)`
  - `f(x)` is some arithmetic expression involving the variable `x`

- Example of an implicit flow of information:

```
if x == 1 then y := 0
          else y := 1;
```

$f(x) = \begin{cases} 0, & \text{if } x == 1 \\ 1 \end{cases}$

- We'll need to look for both explicit and implicit flows of information when analyzing a program, and then use mechanisms to detect and stop flows of information that violate a security policy

Notation for security classes

- Before we discuss possible mechanisms to detect and stop flows of information that would violate a security policy, we need to define some notation for security policies:

- $\underline{X}$ refers to the security class of *X*, as defined by the security policy (in a Bell-LaPadula based system)
- $\underline{X} \leq \underline{Y}$ means that information is allowed to flow from an element in the security class of *X* to an element in the security class of *Y*
  - Alternatively, it means that information with a label placing it in class *X* is allowed to flow into class *Y*

Compiler-based mechanisms

- A compiler-based mechanism that detects and blocks unauthorized flows of information in a program during compilation, with respect to a given security policy
- The analysis conducted by the mechanism is not precise, but it is secure
  - Not precise: A path of information flow that should have been marked as authorized may instead be marked as unauthorized (i.e. it is a false positive)
  - Secure: **No** unauthorized path of information flow will remain **undetected**

- The following definition is important:

  Certified: A set of statements is *certified* with respect to an information flow security policy if the information flow within that set of statements does not violate the security policy

Compiler-based mechanisms – example

- Suppose we have the following statement:

```
if x == 1 then y := m
           else y := n;
```

- From our understanding of the two examples we discussed earlier (calculation of entropy values), we have information flow from *X* and *M* to *Y*, or information flow from *X* and *N* to *Y*
- Then the above statement is certified only if the security policy states that $\underline{X} \leq \underline{Y}$ and $\underline{M} \leq \underline{Y}$ and $\underline{N} \leq \underline{Y}$
  - Note that the information flow of **both** branches must be accounted for in the security policy, unless the compiler is able to determine that one branch will **never** be taken

Compiler-based mechanisms – security class for array

- Suppose we have the following partial statement (information flowing **out**):

  … `:= m[i];`

- Here, the values of both *I* and *M*[*I*] affect the variable being assigned to, so the security class for the array is *max*($\underline{I}$, $\underline{M[I]}$)

Compiler-based mechanisms – security class for array

- Now suppose we have the following partial statement (information flowing **in**):

  `m[i] :=` …

- Here, only the variable $M[I]$ is affected, so the security class for the array is $\underline{M[I]}$

Compiler-based mechanisms – assignment statements

- Suppose we have the following statement:

```
x := y + z;
```

- There is information flow from Y and Z to X, so the above statement is certified only if the security policy states that $max(\underline{Y}, \underline{Z}) \leq \underline{X}$

- In general, for the statement

$$\mathbf{y} := \mathbf{f}(\mathbf{x}_1, \ldots, \mathbf{x}_n);$$

  to be certified by the compiler-based mechanism, the security policy must state that $max(\underline{X}_1, \ldots, \underline{X}_n) \leq \underline{Y}$

Compiler-based mechanisms – compound statements

- Suppose we have the following code:

```
x := y + z;  m := n * o - x;
```

- The first statement is certified only if the security policy states that $max(\underline{Y}, \underline{Z}) \leq \underline{X}$
- The second statement is certified only if the security policy states that

  $max(\underline{N}, \underline{O}, \underline{X}) \leq \underline{M}$
- For the entire code (both statements) to be certified, the security policy needs to state that $max(\underline{Y}, \underline{Z}) \leq \underline{X}$ and $max(\underline{N}, \underline{O}, \underline{X}) \leq \underline{M}$

- In general, for a series of statements

```
<S_1>; … <S_n>;
```

  to be certified, the compiler-based mechanism must certify each and every statement with the security policy

Compiler-based mechanisms – conditional statements

- Suppose we have the following statement:

```
if x + y < z then m := n
                  else p := n * o - x;
```

- For the above statement to be certified, the security policy must have: $\underline{N} \le \underline{M}$ and $max(\underline{N}, \underline{O}, \underline{X}) \le \underline{P}$
- Now, the parts of the statement that are conditionally executed will reveal information about *X*, *Y* and *Z* (because *X*, *Y* and *Z* are part of the condition), so the security policy must also have:

  $max(\underline{X}, \underline{Y}, \underline{Z}) \le min(\underline{M}, \underline{P})$

Compiler-based mechanisms – conditional statements

- In general, for a statement of the following form

```
if f(x_1, …, x_n) then <S_1>
                  else <S_2>
```

  to be certified, the compiler-based mechanism must certify `<S_1>` and `<S_2>`

- **And** the security policy must also have:

  $max(\underline{X}_1, \ldots, \underline{X}_n) \leq min(\underline{Y} \mid Y$ is target of assignment in `<S_1>`, `<S_2>`$)$

Compiler-based mechanisms – iterative statements

- Suppose we have the following code:

```
    while i < n do
                begin p[i] := q[i];
                      i := i + 1;
                end
```

- For the above code to be certified, the compiler-based mechanism just needs to follow the same certification procedure as that used for a conditional statement, but the compiler-based mechanism must also check that the loop terminates eventually

Compiler-based mechanisms – iterative statements

- In general, for code of the following form

```
while f(x_1, …, x_n) do
        <S_1>
        end
```

- to be certified, the compiler-based mechanism must:
  - check that the loop terminates eventually, and
  - certify the statement **<S_1>**,
- **And** the security policy must also have $\mathit{max}(\underline{X}_1, \ldots, \underline{X}_n) \leq \mathit{min}(\underline{Y} \mid Y$ is target of assignment in **<S_1>**)

Compiler-based mechanisms – infinite loops

- Now suppose we have the following code:

```
y := 0;
while x == 0 do
            begin (* nothing *);
            end
y := 1;
```

- The above code can cause problems for a compiler-based mechanism

Compiler-based mechanisms – infinite loops

- If *X* is 0 initially, then we get an infinite loop
- If *X* is some other value initially, then the code terminates with *Y* set to 1
- There is no explicit flow of information, but there is an implicit flow of information from *X* to *Y*
- It is hard for the compiler-based mechanism to detect whether the loop will terminate at *compile time*

Execution-based mechanisms

- An execution-based mechanism may be able to deal with an infinite loop (the mechanism is *dynamic* in nature)
- An execution-based mechanism checks the flow of information at *run time*, not compile time
- It stops any flow of information that violates the security policy
- Before the statement

  ```
  y := f(x1, …, xn);
  ```

  is executed, the execution-based mechanism first verifies that

  $max(\underline{X}_1, \ldots, \underline{X}_n) \leq \underline{Y}$
- The execution-based mechanism will block the execution of the statement if the verification fails

Execution-based mechanisms

- An execution-based mechanism can check for *explicit* flows of information easily
- However, *implicit* flows of information complicate the checking procedure

- Suppose we have the following statement:

```
if x == 1 then y := m;
```

- There is an explicit flow of information from *M* to *Y*, which can be handled by the execution-based mechanism

Execution-based mechanisms

- However, there is an implicit flow of information from X to M
  - Suppose $X \neq 1$, and $\underline{X}$ = **high** security classification, $\underline{Y}$ = **low** security classification and $\underline{M}$ = **low** security classification
  - There will be an *implicit* flow of information from X to Y, because when $X \neq 1$, the assignment `y := m` is **not** executed
  - The execution-based mechanism is **unable** block this process
  - An observer who is authorized to access only Y and M can infer that $X \neq 1$ by checking the values of Y and M, even though he is not authorized to know the value of X