COMPGS10/M028 Language Based Security

Course Work 2, Due Date: 16 April 2012

By: Eman Alashwali

Question (1): All of the questions except for question 4 refer to the following program:

```
if (a > c)
   f = c + d
else
   while (h > 0)
      b = e + 7
```

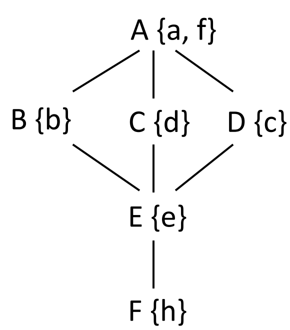
Use Volpano and Smith type inference rules for a while language to establish whether the above program satisfies noninterference with respect to the following security policy.

$\langle Lat, \leq \rangle = \{ F \leq E, E \leq B, E \leq D, E \leq C, B \leq A, D \leq A, C \leq A \}$

and

$\rho = \{ (a,A), (c,D), (f,A), (d,C), (h,F), (b,B), (e,E) \}$

Answer (1):



1. First, we type the true branch ***f = c + d***
2. Since ***f*** has type A var, we try to type the assignment as A cmd. To do this, we must type ***c+d*** as A. We start by typing the variables ***c*** and ***d*** using the (R-VAL), (BASE), (SUBTYPE) rules.

   (VAR) $\gamma \vdash c : D\ var$

(R-VAL) $\frac{\gamma \vdash c: D\ var}{\gamma \vdash c: D}$

3. Since ***f*** is of type A, and the command must agree on type, we will coerce the types of ***c*** and ***d*** to make them type A (the LUB for D).

   (BASE) $\frac{D \leq A}{\vdash D \subseteq A}$

   (SUBTYPE) $\frac{\gamma \vdash c: D \quad D \subseteq A}{\gamma \vdash c: A}$

4. Similar steps for ***d*** in order to agree on type A (the LUB for D).:

   (VAR) $\gamma \vdash d: C\ var$

   (R-VAL) $\frac{\gamma \vdash d: C\ var}{\gamma \vdash d: C}$

   (BASE) $\frac{C \leq A}{\vdash C \subseteq A}$

   (SUBTYPE) $\frac{\gamma \vdash d: C \quad C \subseteq A}{\gamma \vdash d: A}$

5. Type ***c+d***

   (PLUS) $\frac{\gamma \vdash c: A \quad \gamma \vdash d: A}{\gamma \vdash c+d: A}$

6. Now, we can apply the (ASSIGN) rule:

   (ASSIGN) $\frac{\gamma \vdash f: A\ var \quad \gamma \vdash c+d: A}{\gamma \vdash f:=c+d: A\ cmd}$

7. Next, we type the false branch. We type ***b = e + 7.*** Since ***b*** is type B, we will coerce the type of ***e*** + ***7*** to make it type B (the LUB for F).

   (VAR) $\gamma \vdash e: E\ var$

   (R-VAL) $\frac{\gamma \vdash e: E\ var}{\gamma \vdash e: E}$

8. We type ***7*** Using the axiom (INT), then we coerce the type of ***7*** to make it type E. Using (BASE) and (SUBTYPE)

   (INT) $\gamma \vdash 7: F$ (The type of ***7*** is $\perp$ the lowest type in the lattice (bottom) which is F).

   (BASE) $\frac{F \leq E}{\vdash F \subseteq E}$

   (SUBTYPE) $\frac{\gamma \vdash 7: F \quad F \subseteq E}{\gamma \vdash 7: E}$

9. Now, we can apply the rule (PLUS)

   (PLUS) $\frac{\gamma \vdash e: E \quad \gamma \vdash 7: E}{\gamma \vdash e+7: E}$

10. To type ***b = e + 7***. All the types have to agree on type B, so we have to coerce the type of ***e+7*** to make it type B.

(BASE) $\frac{E \leq B}{\vdash E \subseteq B}$

(SUBTYPE) $\frac{\gamma \vdash e+7: E \quad E \subseteq B}{\gamma \vdash e+7: B}$

11. Now, we can apply the (ASSIGN) rule:

(ASSIGN) $\frac{\gamma \vdash b: B\ var \quad \gamma \vdash e+7:B}{\gamma \vdash b := e+7: B\ cmd}$

12. Next, we have to type the while statement. All types must agree. First, we need to type the control expression ***h>0.*** We first type ***h***

(VAR) $\gamma \vdash h: F\ var$

(R-VAL) $\frac{\gamma \vdash h: F\ var}{\gamma \vdash h: F}$

13. We type ***0*** using the axiom (INT), then use the rule (PLUS) to type ***h>0***

(INT) $\gamma \vdash 0: F$

(PLUS) $\frac{\gamma \vdash h: F \quad \gamma \vdash 0: F}{\gamma \vdash h > 0: F}$

14. Now, to type the while statement, all types must agree. We can coerce the type of B cmd to F cmd since the ordering of cmd types is in the opposite direction to the base types.

(CMD) $\frac{\gamma \vdash F \subseteq B}{\gamma \vdash B\ cmd \subseteq F\ cmd}$

(SUBTYPE) $\frac{\gamma \vdash b := e+7: B\ cmd \quad \gamma \vdash B\ cmd \subseteq F\ cmd}{\gamma \vdash b := e+7: F\ cmd}$

(WHILE) $\frac{\gamma \vdash h>0: F \quad \gamma \vdash b := e+7:F\ cmd}{\gamma \vdash while\ h>0\ \mathit{do}\ b := e+7:F\ cmd}$

15. To type the Boolean expression for the if statement ***if (a > c).*** First we type ***a:***

(VAR) $\gamma \vdash a: A\ var$

(R-VAL) $\frac{\gamma \vdash a: A\ var}{\gamma \vdash a: A}$

16. Similar for ***c***:

(VAR) $\gamma \vdash c: D\ var$

(R-VAL) $\frac{\gamma \vdash c: D\ var}{\gamma \vdash c: D}$

17. We have to coerce the type D to make types agrees

(BASE) $\frac{D \leq A}{\vdash D \subseteq A}$

(SUBTYPE) $\dfrac{\gamma \vdash c: D \quad D \subseteq A}{\gamma \vdash c: A}$

18. Now, we can apply the rule (PLUS) to type ***a > c***

(PLUS) $\dfrac{\gamma \vdash a: A \quad \gamma \vdash c:A}{\gamma \vdash a > c: A}$

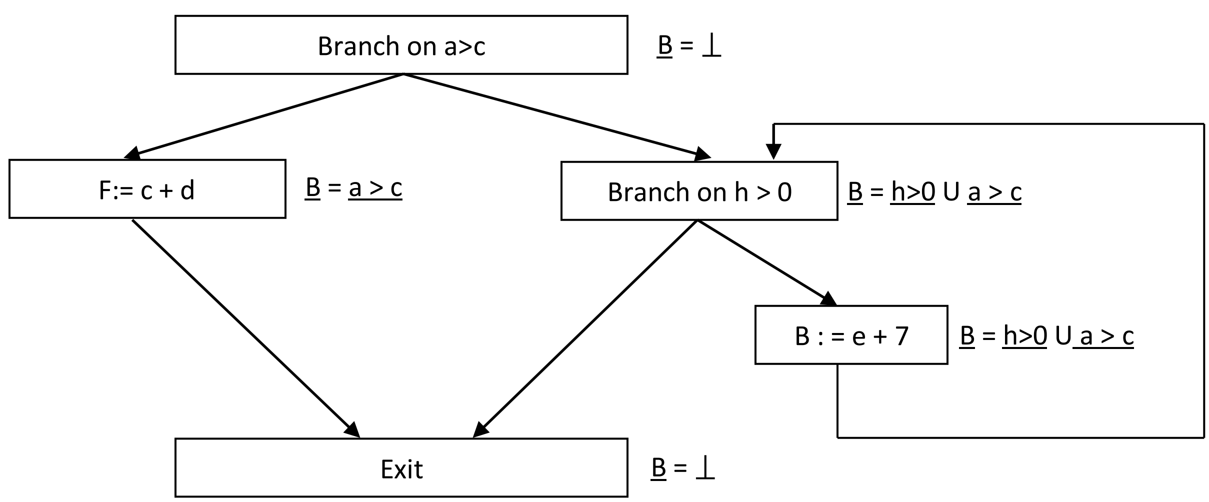
19. To type the if statement, we need all the types to agree. Since A is the highest data type we can not change the type of the statement ***a>c***. The type of the true branch agree with this. **The problem is the type of the false branch**. Since the lattice of the cmd type is inverted, A cmd is on the bottom and so F cmd > A cmd. Since we can not move down the lattice, we can not use the subtype rules to give the ***while (h>0) do b=e+7*** the type A cmd. So, we can not type the if statement, so **it is not flow secure.**
20. End solution

Question (2): State the safety condition for assigning a value to a slot in the decentralized label model. Then draw the Basic Block Graph for the above program and derive the block label for each block using the underline notation

Answer (2):

1. The safety condition for assigning a value to a slot, e.g. f = c + d
   a. Writing a value to a slot: The relabeling must be a restriction, i.e. the slot must have more owners or fewer readers for some owners or both

2. The Basic Block Graph for the above program :



Question (3): Construct a syntax tree and use the inference rules for natural semantics to give the natural semantics for the above program when it starts in a state s =< a → 5, b → 4, c → 3, d → 2, e → 1, f → 0, h → -1 >

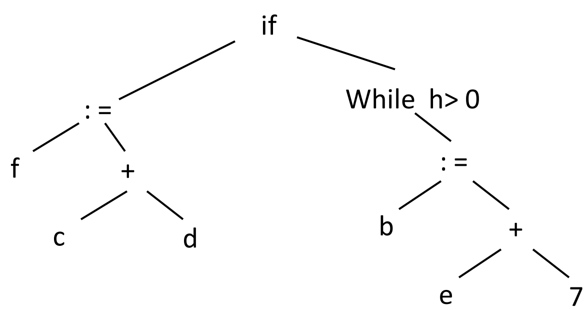
Answer (3):

1. The natural semantics is as follows:

$$(\text{If}_{ns}^{tt})\ \frac{\text{If} < f{:=}\ c + d, s> \to s_1}{< \text{if } a > c \text{ then } f{:=}\ c + d \text{ else while } (h > 0)\ b{:=}\ e + 7,\ s_0 > \to s_1} \quad \text{If B [[a>c]] = tt}$$

$s1 = s[\, f \to 5\,]$

2. The syntax tree is as follows:



Question (4): Consider the program: if (a < 3) then b := 2 else b := b % 2

(a) Both a and b are 2 bit variables with values in the range [0..3]. Assume a uniform probability distribution on the input space. If a is confidential and b is public, calculate the leakage into the final value of b.

(b) State and explain the general definition of leakage. Which definition of leakage is suitable for this program?

Answer (4.a):

Input space has u.p.d of 1/16 each state

The size of the secret space is $4.1/4. \log_2 4$ = 2 bits

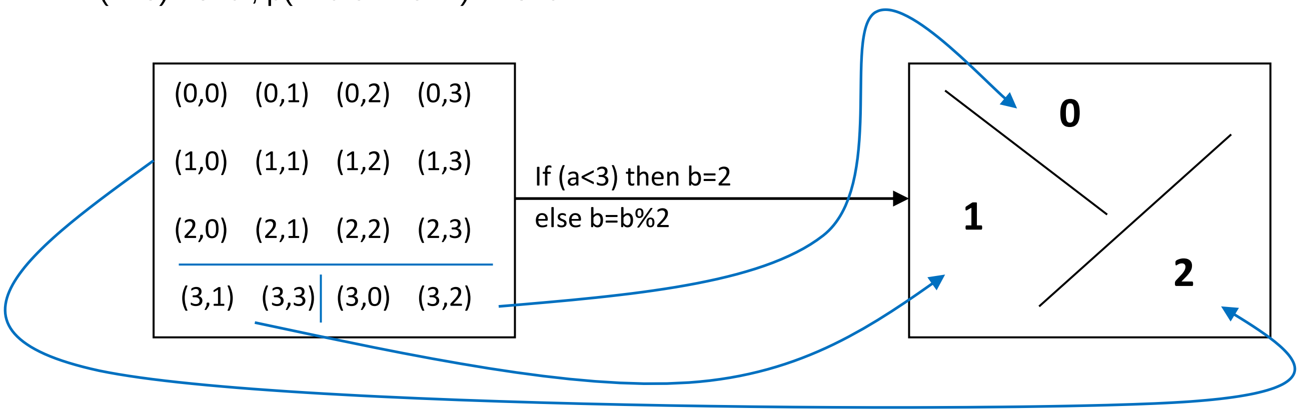
Low can make 3 observations via variable b: 0, 1 and 3.

Observing 2 can correspond to values {0,1,2} of a.

Observing 1 can correspond to {3}

Observing 0 can correspond to {3}

P(h=3)=4/16 , p(h=0 or 1 or 2)=12/16



Information about a from observation H([12/16, 4/16]) = H([3/4, 1/4]) =

$3/4 \log_2 4/3 + 1/4 \log_2 4 = 0.308 + 0.5 = 0.808 \approx 0.81$

Answer (4. b): The general definition of leakage is: L = I(H; L'|L). That is the mutual information between the random variable in the low output after discounting knowledge of the random variable in the low inputs

---

Question (5): Consider the first program with the following security policy: <Lat, ≤ > = {L ≤ H} with ρ = { (a,H), (c,L), (f,L), (d,L), (h,L), (b,L), (e,L)}, perform a flow logic based analysis for non-interference on the program using this security policy and demonstrate whether the program is flow secure.

Answer (5):

1. First, we need to label the program statements.

(if (a > c) then **(**f := c + d**)**^{l1}1 else **(**while (h > 0) do **(**b := e + 7**)**^{l2} **)**^{l3} **)**^{l4}

2. Then, use the analysis rules to generate the constraints for each label.

$\widehat{D}$ (l_1) ⊇ Id [f ⟶ {c, d}]

$\widehat{D}$ (l_2) ⊇ Id [b ⟶ {e}]

$\widehat{G}$ (l_3) ⊇ {•} ∪ FV(h > 0) ∪ $\widehat{G}$ (l_2) ∪ $\widehat{G}$ (l_3) ; $\widehat{D}$(l_2)

$\widehat{D}(l_3) \supseteq Id \cup \widehat{D}(l_3) ; \widehat{D}(l_2)$

$\widehat{D}(l_3) \supseteq \hat{X}(l_3) \times FV(a > c)$

$\widehat{G}(l_4) \supseteq \widehat{G}(l_1) \cup \widehat{G}(l_3)$

$(\bullet \in \widehat{G}(l_4) \rightarrow \widehat{G}(l_4) \supseteq FV(a > c))$

$\widehat{D}(l_4) \supseteq \widehat{D}(l_1) \cup \widehat{D}(l_3)$

$\widehat{D}(l_4) \supseteq \hat{X}(l_4) \times FV(a > c)$

3. We have $FV(h > 0) = \{ h \}$, $\hat{X}(l_3) = \{ b \}$ and $FV(a > c) = \{ a, c \}$, $\hat{X}(l_4) = \{ f, b \}$
   So, $\hat{X}(l_3) \times FV(h > 0) = \{ b \leftarrow \{h\} \}$ and $\hat{X}(l_4) \times FV(a > c) = \{ f \leftarrow \{a, c\}, b \leftarrow \{a, c\} \}$. We have to substitute these values into the constraints to create a working constraint set:

$\widehat{D}(l_1) \supseteq Id [f \rightarrow \{c, d\}]$

$\widehat{D}(l_2) \supseteq Id [b \rightarrow \{e\}]$

$\widehat{G}(l_3) \supseteq \{\bullet\} \cup \{ h \} \cup \widehat{G}(l_2) \cup \widehat{G}(l_3) ; \widehat{D}(l_2)$

$\widehat{D}(l_3) \supseteq Id \cup \widehat{D}(l_3) ; \widehat{D}(l_2)$

$\widehat{D}(l_3) \supseteq \{ b \rightarrow \{h\} \}$

$\widehat{G}(l_4) \supseteq \widehat{G}(l_1) \cup \widehat{G}(l_3)$

$(\bullet \in \widehat{G}(l_4) \rightarrow \widehat{G}(l_4) \supseteq \{ a, c \})$

$\widehat{D}(l_4) \supseteq \widehat{D}(l_1) \cup \widehat{D}(l_3)$

$\widehat{D}(l_4) \supseteq \{ f \rightarrow \{a, c\}, b \rightarrow \{a, c\} \}$

4. Next, perform the iterations, the first iteration is for initialization, then in repeat iterations and in each iteration substitute the inclusions from the previous iterations until we reach a fixed point.
   - **Iteration 0:**

   $\widehat{G}(l_1) \supseteq \emptyset$

$\widehat{D}(l_1) \supseteq \emptyset$

$\widehat{G}(l_2) \supseteq \emptyset$

$\widehat{D}(l_2) \supseteq \emptyset$

$\widehat{G}(l_3) \supseteq \emptyset$

$\widehat{D}(l_3) \supseteq \emptyset$

$\widehat{G}(l_4) \supseteq \emptyset$

$\widehat{D}(l_4) \supseteq \emptyset$

- **Iteration 1**

$\widehat{G}(l_1) \supseteq \emptyset$

$\widehat{D}(l_1) \supseteq \text{Id}[\, f \rightarrow \{c, d\}]$

$\widehat{G}(l_2) \supseteq \emptyset$

$\widehat{D}(l_2) \supseteq \text{Id}[\, b \rightarrow \{e\}\,]$

$\widehat{G}(l_3) \supseteq \{\bullet, h\}$

$\widehat{D}(l_3) \supseteq \text{Id}[\, b \rightarrow \{h\}\,]$

$\widehat{G}(l_4) \supseteq \emptyset$

$\widehat{D}(l_4) \supseteq \{ f \rightarrow \{a, c\}, b \rightarrow \{a, c\}\}$

- **Iteration 2**

$\widehat{G}(l_1) \supseteq \emptyset$

$\widehat{D}(l_1) \supseteq \text{Id}[\, f \rightarrow \{c, d\}]$

$\widehat{G}(l_2) \supseteq \emptyset$

$\widehat{D}(l_2) \supseteq \text{Id}[\, b \rightarrow \{e\}\,]$

$\widehat{G}(l_3) \supseteq \{\bullet, h\}$

$\widehat{D}(l_3) \supseteq \text{Id}[\, b \rightarrow \{h, e\}\,]$

$\widehat{G}(l_4) \supseteq \{\bullet, h\}$

$\widehat{D}(l_4) \supseteq \text{Id}[\{ f \rightarrow \{a, c, d\}, b \rightarrow \{a, c, h\}\}]$

- **Iteration 3**

$\widehat{G}(l_1) \supseteq \emptyset$

$\widehat{D}(l_1) \supseteq \text{Id}[\, f \rightarrow \{c, d\}]$

$\widehat{G}(l_2) \supseteq \emptyset$

$\widehat{D}(l_2) \supseteq \text{Id}[\, b \rightarrow \{e\}\,]$

$\widehat{G}(l_3) \supseteq \{\bullet, h\}$

$\widehat{D}$ (l_3) ⊇ Id [b → { h , e }]

$\widehat{G}$ (l_4) ⊇ { •, h, a, c }

$\widehat{D}$ (l_4) ⊇ Id [{ f → { a, c, d}, b → { a, c, h, e } }]

- **Iteration 4**

$\widehat{G}$ (l_1) ⊇ ∅

$\widehat{D}$ (l_1) ⊇ Id [f → { c, d }]

$\widehat{G}$ (l_2) ⊇ ∅

$\widehat{D}$ (l_2) ⊇ Id [b → { e }]

$\widehat{G}$ (l_3) ⊇ { • , h }

$\widehat{D}$ (l_3) ⊇ Id [b → { h , e }]

$\widehat{G}$ (l_4) ⊇ { • , h, a, c }

$\widehat{D}$ (l_4) ⊇ Id [{ f → { a, c, d}, b → { a, c, h, e } }]

Since nothing has been updated, we have reached a fixed point, giving the smallest solution. If we check the $\widehat{G}$ and $\widehat{D}$ for l_4, the label for the whole program, we find that a ∈ $\widehat{G}$ (l_4) and every low security variable depends on the value of a, so the program is not flow secure and will not satisfy non-interference property.