Exercise 3: Information Flow and Slicing

Deadline for uploading solutions via Ilias:
January 29, 2021, 11:59pm Stuttgart time

Task 1  Information Flow Analysis  [20 points]

This task is about dynamic information flow analysis. Consider the following JavaScript code to analyze:

```javascript
var age = getAge();
var flag = getFlag();
var count = 0;
var mult = 2;
if (age === 20){
  count++;
} else {
  if (age === 25){
    flag = 2;
  } else {
    if (age === 30) {
      flag = 3;
    }
  }
}
if (flag === 2){
  flag = flag * mult;
  print_public(flag);
} else {
  print_public(mult);
}
```

There are three security classes: secret, confidential, and public, which are ordered into a lattice such that:

$$secret > confidential > public$$

By default, all values are labeled as public. Values returned by `getAge()` are labeled as secret.
and values returned by \texttt{getFlag()} are labeled as \textit{confidential}. The function \texttt{print\_public()} is an untrusted sink, which should only be reached by \textit{public} information. Note that passing an argument to function should be handled like an assignment to the formal parameter of the function.

**Subtask 1.1 Execution 1**

Consider a dynamic information flow analysis that considers both explicit and implicit flows. Suppose an execution where \texttt{getAge()} returns 20 and \texttt{getFlag()} returns 3.

- What are the security labels of variables and expressions during the execution? Use the following template to provide your answer.

<table>
<thead>
<tr>
<th>Line</th>
<th>Variable or expression</th>
<th>Security label of variable or expression (after executing the line)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>age</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>flag</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>count</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>mult</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>age === 20</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>count</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>flag === 2</td>
<td></td>
</tr>
</tbody>
</table>

- Does the execution violate the information flow policy? Explain your answer.
Subtask 1.2  Execution 2

Consider a dynamic information flow analysis that considers both explicit and implicit flows. Suppose an execution where \texttt{getAge()} returns 25 and \texttt{getFlag()} returns 1.

- What are the security labels of variables and expressions during the execution? Use the following template to provide your answer.

<table>
<thead>
<tr>
<th>Line</th>
<th>Variable or expression</th>
<th>Security label of variable or expression (after executing the line)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>age</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>flag</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>count</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>mult</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>age === 20</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>age === 25</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>flag</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>flag === 2</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>flag</td>
<td></td>
</tr>
</tbody>
</table>

- Does the execution violate the information flow policy? Explain your answer.
Task 2  Universally Bounded Lattice  [20 points]

In this task we will analyze the characteristics and properties of universally bounded lattices.

Subtask 2.1  Recognize a Universally Bounded Lattice  [10 points]

Figure 1 shows structures that could represent universally bounded lattices.

![Figure 1: Two universally bounded lattice candidates.](image)

- Are the two candidates universally bounded lattices? Explain why (not).
Subtask 2.2  Characteristics  [10 points]

Consider the following structure that represents a universally bounded lattice:

Answer the following questions:

• Give the set $S$ of security classes.

  $$S =$$

• What is the lower bound $\bot$?

  $$\bot =$$

• What is the upper bound $\top$?

  $$\top =$$

• Let $\oplus$ be the least upper bound operator. What is the result of the following operations?

  $$B \oplus BC =$$

  $$ABCD \oplus CD =$$

  $$ABC \oplus \emptyset =$$
Let $\otimes$ be the greatest lower bound operator. What is the result of the following operations?

\[
D \otimes DC = \\
ACD \otimes AD = \\
ABCD \otimes B =
\]
Consider the following JavaScript program:

```javascript
var year = getYear();
var flag = getFlag();
var count = 0;
var mult = 3;
if (year === 2020) {
  count++;
  mult = mult * count;
} else {
  if (year === 2021) {
    flag = 2 + flag;
  }
}
var sliceHere = count;
```

Compute the static backward slice for variable `sliceHere` at line 13. Use the slicing approach of Weiser (IEEE TSE, 1984) and its formulation as a graph reachability problem, as it has been introduced in the lecture. To describe your solution, follow the steps outlined below.

**Subtask 3.1 Data Flow Dependences**

Provide the data flow dependences between statements in the program. Use the following table to summarize the dependences. Each table cell represents a pair of statements. Mark all pairs of statements that have a definition-use relationship.

<table>
<thead>
<tr>
<th>Def</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
</tbody>
</table>
Subtask 3.2 Control Flow Dependences

Provide the control flow dependences between statements in the program. Describe your solutions as a sequence of “Statement .. is control-flow dependent on statement ..” sentences.

Subtask 3.3 Program Dependence Graph

Summarize the data flow dependences and the control flow dependences into a program dependence graph. Use the following template to draw your solution.

\[ \text{.. destination is data-dependent on source} \]
\[ \text{.. destination is control-dependent on source} \]
Subtask 3.4 Slice

What is the slice for the slicing criterion (variable sliceHere at line 13)? Write down the source code of the sliced program as a syntactically correct program.
Task 4  Dynamic Slicing (Revised Approach)  [30 points]

Consider the following JavaScript program:

```javascript
var year = getInput();
var actual_year = 2021;
var count = 0;
var flag = false;
var diff = actual_year - year
var i = 0;
while (i < diff) {
  i++;
  year++;
  if (year > 2000) {
    count++;
  }
}
if (count > 10) {
  flag = true
}
console.log(i)
```

Subtask 4.1  Execution History  [5 points]

Give the execution history with `getInput() = 2018`. You can use the line numbers to refer to statements. Do not include the lines with the closing curly brackets ( `{}` ) into the history.
Subtask 4.2 Dynamic Dependence Graph [20 points]

Suppose we want to compute the dynamic backward slice with the last statement (`console.log(i)`) as the slicing criterion. Use `getInput() = 2019` (note that this input is different from above). Provide the dynamic dependence graph, using the revised approach presented in the lecture.
Subtask 4.3  Slice  
Write the sliced program.