LANGUAGE BASED TECHNOLOGY FOR SECURITY
June 28 2024

Duration 40 minutes – Answers can be written either in English or in Italian.

Exercise 1: Flow Types

Consider the following program under the typing context $\Gamma = \{a = H, b = L, c = \ell\}$ where ℓ is a variable ranging over the lattice $L \leq H$.

```
if (a < b) then {
                if (b < a) then c = 0 else c = 1;
                } else {
                   if (a < b) the c = 0 else c = 1; }
```

- For which value of the variable ℓ is the above program well-typed by considering the rules of the static type system presented during the lectures? Discuss the constraints obtained from the type derivation.
- Although the static type system presented in class may reject some programs that satisfy noninterference, it is sound: it will never accept a program that violates the policy. Discuss an example where tracking information flow dynamically at runtime might mitigate some of the "false" rejections.

Exercise 2: Static Taint Analysis

Consider the following program

```
void printVal(untainted int} {...};

if (a == 5) then {
                if  (b == 1) then {z = 8;} else {z = c;}
                } else  {z = 7;}
 printVal(z);
```

- Assume that variables `a, b, c` are bound to tainted values. Discuss how static taint analysis is applied to discover whether the previous program fragments exhibit taint flow. Specifically define and illustrate the constraints generated by the static taint analysis.

Exercise 3: Security Polices

Consider a piece of mobile code that needs to create or modify files on a local file system for its internal book-keeping operations. It is perfectly reasonable to allow untrusted mobile code to do this, as long as the files created/manipulated are unrelated to other applications. This can be ensured using a policy that enforces the following properties.

- Writes are only allowed to certain directories. The untrusted code should not be allowed to create files in arbitrary directories. The only directories allowed are those under `/tmp`
- Overwriting or deletion of a file is not permitted except for files created by the same mobile code. This ensures that untrusted mobile code does not remove files created by other applications.

1. Design the *Security automaton* specifying the security policy discussed above.
2. Exploit the security automata defined in the previous question to instrument the code below

```
myWriter = new FileWriter(stringVal);
myWriter.write("Writing a file might be tricky");
myWriter.Close()
```

Exercise 4: Liveness analysis

Perform liveness analysis on the following piece of code

```
var  a,b,c;
a = 0;
do  {
b = a + 1;
c = c + b;
a = b + 2;
}  while (a < N);
output  c;
```

- drawing its control flow graph,
- computing the constraints for each block, and
- computing the least solution of the constraints.

[Optional] Do the results of liveness analysis on this code enable any optimization opportunities? If so, describe the optimization.