FogFaas

A faas extension of SecFog

Laboratory for Innovative Software

A. Y. 2019/2020

1 Introduction

2 In class work

3 Extensions

3.1 Communication between services

3.2 Extension of Tau

3.3 Type system

In order to strengthen our type system to prevent information flow leaks from our programs. We considered language-based information-flow security paper, which particularly focusing on work that uses static program analysis to enforce information-flow policies. Extension on type system is based on covert channels definitions that are considered in the paper. We consider implementing implicit flows, by extending if-then-else and termination channels by extending while loop.

The attacker can find out which part of the control flow is executed by checking execution time. We change the type system to enforce both parts of the control flow to have the same execution time therefore the attacker cannot observe to distinguish flow.

In the representation of the type system, we added time units variable to predicator, therefore we can have time that is needed for checking. For the if-then-else we checked the time unit of then and else part to see if they are equal. In both parts of control flow there can be other programs like, sequential, while loop or try-catch there for the time calculation for that program is needed.

- For a single program we just update time unit label with program time units.
- For checking the sequential program, we updated the time of the program with the sum of the time units of two sequential programs.
- For checking try-catch we assumed it is like a control flow therefore first we checked if both try and catch have same time units then update the time units label of the program with one of them.
- For checking while loop we assumed each loop has infinitive time units because we cannot know how many times this loop is going to iterate, therefore, iteration was assumed infinity. To have valid if then else, both parts of the control flow need to have a while loop.

Termination channels signal information through the termination or non-termination of a computation. We need to prevent the use of secret arguments inside guard,

therefore, we add a guard-check predicate inside a while loop which checks if the guard has secret labeled arguments. The operator needs to define the labels that are forbidden to use in guard. In our program top secret, secret, secret to Europe, secret to U.S. are forbidden to use.

3.4 Trigger modelling

4 Putting it all togheter

5 Conclusions and future work