Big-Step Operational Semantics with Security Labels

1 Overview

This document defines a big-step operational semantics for a small expression language extended with a `secret` construct. The goal is to track whether values are *public* or *secret*, and to ensure that secret information cannot be leaked, even through control flow.

The semantics enforces *non-interference*: secret data may influence computation, but it may never influence publicly observable results.

2 Syntax

The expressions of the language are:

$$\begin{array}{rcl} e & ::= & \texttt{true} \mid \texttt{false} \mid i \\ & \mid & \texttt{if } e \texttt{ then } e \texttt{ else } e \\ & \mid & \texttt{succ } e \\ & \mid & \texttt{pred } e \\ & \mid & \texttt{secret } e \end{array}$$

where $i \in \mathbb{Z}$.

3 Values and Security Labels

The set of values is:

$$v ::= \texttt{true} \mid \texttt{false} \mid i$$

Each value is paired with a *security label*:

$$\ell ::= L \mid H$$

- L (low) means public information
- H (high) means secret information

An evaluated expression produces a labeled value of the form v^ℓ.

4 Evaluation Judgement

The big-step evaluation relation is written as:

$$e \Downarrow v^\ell$$

This should be read as:

Expression e evaluates to value v with security label ℓ.

5 Operational Semantics

5.1 Base Values

Constants are always public, since they do not depend on any secret input.

$$\frac{}{\texttt{true} \Downarrow \texttt{true}^L} \qquad \frac{}{\texttt{false} \Downarrow \texttt{false}^L} \qquad \frac{}{i \Downarrow i^L}$$

5.2 Secret

The `secret` construct explicitly marks data as private. It evaluates its subexpression and upgrades the label to H.

$$\frac{e \Downarrow v^\ell}{\texttt{secret}\ e \Downarrow v^H}$$

Intuition. The value itself is unchanged, but from this point onward it is treated as secret and cannot be safely observed.

5.3 Arithmetic Operations

Arithmetic operations propagate the security label of their argument.

$$\frac{e \Downarrow i^\ell}{\texttt{succ}\ e \Downarrow (i+1)^\ell} \qquad \frac{e \Downarrow i^\ell}{\texttt{pred}\ e \Downarrow (i-1)^\ell}$$

Intuition. If an arithmetic result depends on secret data, then the result must also be secret. This captures *explicit information flow*.

5.4 Conditionals

Conditionals are the main source of *implicit information flow*. To prevent leaks, the label of the result must depend on both:

- the condition
- the chosen branch

We define the label join operator $\sqcup$ as:

$\sqcup$	L	H
L	L	H
H	H	H

$$\frac{e_1 \Downarrow \texttt{true}^{\ell_1} \qquad e_2 \Downarrow v^{\ell_2}}{\texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3 \Downarrow v^{\ell_1 \sqcup \ell_2}} \qquad \frac{e_1 \Downarrow \texttt{false}^{\ell_1} \qquad e_3 \Downarrow v^{\ell_3}}{\texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3 \Downarrow v^{\ell_1 \sqcup \ell_3}}$$

Key idea. If the condition is secret, then the observer must not be able to tell which branch was taken. Therefore, the result is labeled H even if the branch itself produces a public value.

6 Security Guarantee

This semantics enforces *non-interference*:

- Secret data may influence computation
- Any value influenced by secret data is labeled H
- There is no rule that converts H back to L

As a result, attackers cannot write code that leaks secret information, even indirectly through control flow.

7 Example

$$\texttt{succ (secret 3)} \Downarrow 4^H \qquad \texttt{succ 3} \Downarrow 4^L$$

The difference in labels reflects whether the computation depends on secret information.