16.4. Execution-Based Mechanisms

The goal of an execution-based mechanism is to prevent an information flow that violates policy. Checking the flow requirements of explicit flows achieves this result for statements involving explicit flows. Before the assignment

$$y = f(x_1, \ldots, x_n)$$

is executed, the execution-based mechanism verifies that

$$\mathbf{lub}(\underline{x}_1, \ldots, \underline{x}_n) \leq \underline{y}$$

If the condition is true, the assignment proceeds. If not, it fails. A naïve approach, then, is to check information flow conditions whenever an explicit flow occurs.

Implicit flows complicate checking.

EXAMPLE: Let **x** and **y** be variables. The requirement for certification for a particular statement **y op x** is that $\underline{x} \leq \underline{y}$. The conditional statement

```
if
x = 1 then
y := a;
```

causes a flow from **x** to **y**. Now, suppose that when $x \neq 1$, $\underline{x}$ = **High** and $\underline{y}$ = **Low**. If flows were verified only when explicit, and $\mathbf{x} \neq 1$, the implicit flow would not be checked. The statement may be incorrectly certified as complying with the information flow policy.

Fenton explored this problem using a special abstract machine.

16.4.1. Fenton's Data Mark Machine

no 16.4.1,16.4.2

Fenton [345] created an abstract machine called the **Data Mark Machine** to study handling of implicit flows at execution time. Each variable in this machine had an associated security class, or tag. Fenton also included a tag for the program counter (PC).

The inclusion of the PC allowed Fenton to treat implicit flows as explicit flows, because branches are merely assignments to the PC. He defined the semantics of the Data Mark Machine. In the following discussion, **skip** means that the instruction is not executed, **push**(**x**, $\underline{x}$) means to push the variable **x** and its security class $\underline{x}$ onto the program