Flow Analysis: Adding ***Sensitivity***

Conditionals

```
int printf(untainted char *fmt, …);
tainted char *fgets(…);
```

```
α char *name = fgets(…, network_fd);
β char *x;
if (…)   x = name;
else     x = “hello!”;
printf(x);
```

tainted ≤ **α**

α ≤ **β**

untainted ≤ **β**

β ≤ **untainted**

Constraints still unsolvable

Illegal flow

Dropping the Conditional

```
int printf(untainted char *fmt, …);
tainted char *fgets(…);
```

```
α char *name = fgets(…, network_fd);
β char *x;
x = name;
x = “hello!”;
printf(x);
```

tainted ≤ α

α ≤ β

untainted ≤ β

β ≤ untainted

Same constraints,
different semantics!

False Alarm

Flow Sensitivity

- Our analysis is **flow *in*sensitive**
  - Each variable has **one qualifier** which abstracts the taintedness of all values it ever contains

- A **flow sensitive analysis** would account for variables whose contents change
  - Allow each assigned use of a variable to have a different qualifier
    - E.g., α_1 is x's qualifier at line 1, but α_2 is the qualifier at line 2, where α_1 and α_2 can differ
  - Could implement this by transforming the program to assign to a variable at most once
    - Called **static single assignment (SSA)** form

Reworked Example

```
int printf(untainted char *fmt, …);
tainted char *fgets(…);
```

```
α char *name = fgets(…, network_fd);
β char *x1, γ *x2;
x1 = name;
x2 = "%s";
printf(x2);
```

tainted ≤ α

α ≤ β

untainted ≤ γ

γ ≤ untainted

No Alarm

Good solution exists:

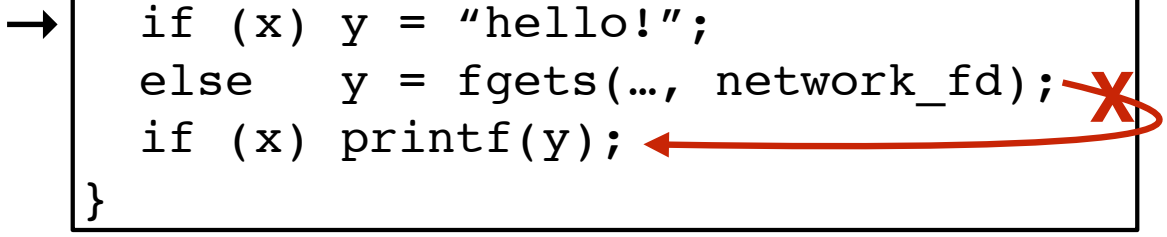
γ = untainted

α = β = tainted

Multiple Conditionals

```
int printf(untainted char *fmt, …);
tainted char *fgets(…);
```

```
void f(int x) {
  α char *y;
  if (x) y = “hello!”;
  else   y = fgets(…, network_fd);
  if (x) printf(y);
}
```



untainted ≤ **α**

tainted ≤ **α**

α ≤ **untainted**

no solution for **α**

False Alarm!

(and flow sensitivity won’t help)

Path Sensitivity

- An analysis may consider *path feasibility*. E.g., `f(x)` can execute path
  - **1-2-4-5-6** when `x` is *not* `0`, or
  - **1-3-4-6** when `x` is `0`. But,
  - path **1-3-4-5-6** *infeasible*

```
void f(int x) {
  char *y;
  [1]if (x) [2]y = "hello!";
   else     [3]y = fgets(…);
  [4]if (x) [5]printf(y);
[6]}
```

- A **path sensitive analysis** checks feasibility, e.g., by qualifying each constraint with a **path condition**
  - $x \neq 0 \Longrightarrow$ **untainted** $\leq \alpha$ (segment 1-2)
  - $x = 0 \Longrightarrow$ **tainted** $\leq \alpha$ (segment 1-3)
  - $x \neq 0 \Longrightarrow \alpha \leq$ **untainted** (segment 4-5)

Why *not* flow/path sensitivity?

- Flow sensitivity **adds precision**, and path sensitivity adds even more, which is *good*
- But both of these **make solving more difficult**
  - Flow sensitivity also *increases the number of nodes* in the constraint graph
  - Path sensitivity *requires more general solving procedures* to handle path conditions
- In short: **precision** (often) **trades off scalability**
  - Ultimately, limits the size of programs we can analyze