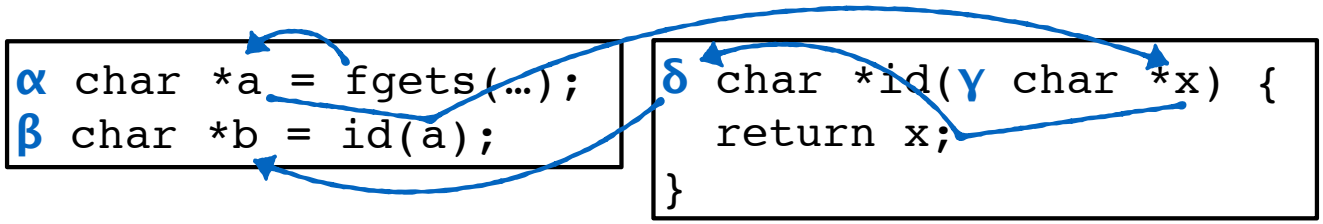
Handling Function Calls

```
α char *a = fgets(…);
β char *b = id(a);
```

```
δ char *id(γ char *x) {
  return x;
}
```

- Names for arguments and return value
- Calls create flows
  - from **caller's data** to **callee's arguments**,
  - from **callee's result** to **caller's returned value**

Handling Function Calls



tainted ≤ **α**

α ≤ **γ**

γ ≤ **δ**

δ ≤ **β**

Function Call Example

```
α char *a = fgets(…);
β char *b = id(a);
ω char *c = “hi”;
printf(c);
```

```
δ char *id(γ char *x) {
  return x;
}
```

tainted ≤ α

α ≤ γ

γ ≤ δ

δ ≤ β

untainted ≤ ω

ω ≤ untainted

No Alarm

Good solution exists:

ω = untainted

α = β = γ = δ = tainted

Two Calls to Same Function

```
α char *a = fgets(…);
β char *b = id(a);
ω char *c = id("hi");
printf(c);
```

```
δ char *id(γ char *x) {
  return x;
}
```

tainted ≤ **α**

α ≤ **γ**

γ ≤ **δ**

δ ≤ **β**

untainted ≤ **γ**

δ ≤ **ω**

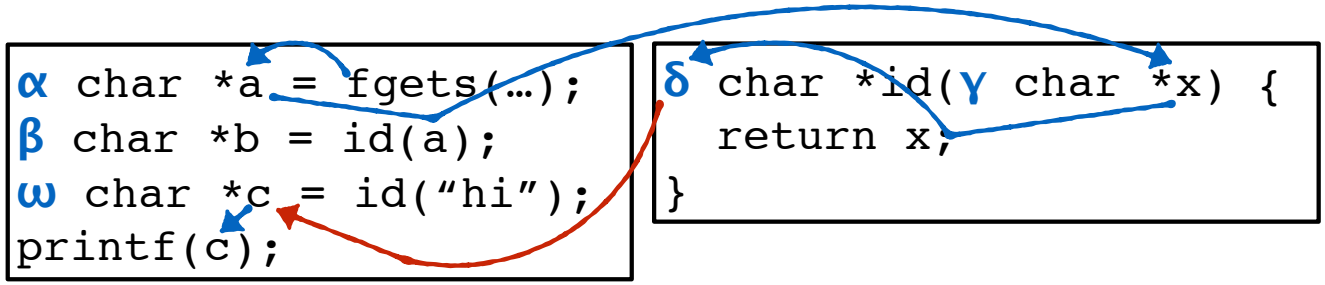
ω ≤ **untainted**

False Alarm!

No solution, and yet

no true tainted flow

Two Calls to Same Function



tainted ≤ **α** ≤ **γ** ≤ **δ** ≤ **ω** ≤ **untainted**

Problematic constraints represent an infeasible path

False Alarm!

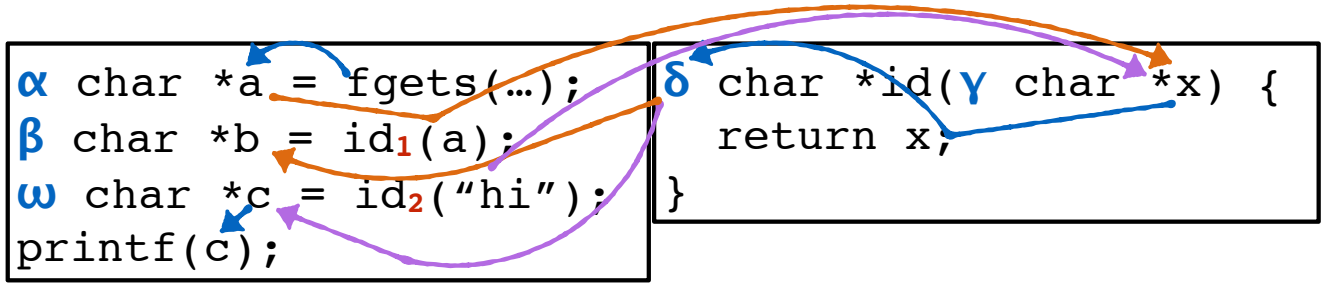
No solution, and yet

no true tainted flow

Context (In)sensitivity

- This is a problem of **context insensitivity**
  - All call sites are “conflated” in the graph
- **Context sensitivity** solves this problem by
  - **distinguishing call sites** in some way
    - We can give them a label ***i***, e.g., the line number in the program
  - **matching** up **calls** with the corresponding **returns**
    - Label call and return edges
    - Allow flows if the labels and **polarities** match
    - Use index **-i** for **argument passing**, i.e., $q_1 \leq_{-i} q_2$
    - Use index **+i** for **returned values**, i.e., $q_1 \leq_{+i} q_2$

Two Calls to Same Function



tainted ≤ α

α ≤-1 γ

γ ≤ δ

δ ≤+1 β

untainted ≤-2 γ

δ ≤+2 ω

ω ≤ untainted

Indexes don't match up

Infeasible flow not allowed

No Alarm

Discussion

- **Context sensitivity** is a **tradeoff** again
  - *Precision* vs. *scalability*
    - *O(n)* insensitive algorithm becomes $O(n^3)$ sensitive algorithm
  - But: sometimes *higher precision improves performance*
    - Eliminates infeasible paths from consideration (makes *n* smaller)
- Compromises possible
  - Only ***some*** **call sites treated sensitively**
    - Rest conflated
  - **Conflate *groups* of call sites**
    - Give them the same index
  - **Sensitivity only *up to a certain call depth***
    - Don't do exact matching of edges beyond that depth