Flow Analysis:
Scaling it up
to a complete language and problem set

Pointers

```
α char *a = "hi";
(β char *)*p = &a;
(γ char *)*q = p;
ω char *b = fgets(…);
*q = b;
printf(*p);
```

Solution exists:

α = β = untainted

ω = γ = tainted

untainted ≤ α

α ≤ β

β ≤ γ

tainted ≤ ω

ω ≤ γ

β ≤ untainted

Misses illegal flow!

- `p` and `q` are aliases
  - -so writing **tainted** data to `q`
  - -makes `p`'s contents **tainted**

Pointers

```
α char *a = “hi”;
(β char *)*p = &a;
(γ char *)*q = p;
ω char *b = fgets(…);
*q = b;
printf(*p);
```

~~Solution exists:~~

α = β = untainted

ω = γ = tainted

untainted ≤ α

α ≤ β

β ≤ γ

γ ≤ β

tainted ≤ ω

ω ≤ γ

β ≤ untainted

Flow and pointers

- An assignment via a pointer “flows both ways”
  - Ensures that **aliasing constraints are sound**
  - But can lead to **false alarms**
- Reducing alarms
  - If pointers are never assigned to (`const`) then backward flow is not needed (sound)
  - Drop backward flow edge anyway
    - Trades false alarms for missed errors (unsoundness)

Implicit flows

```
void copy(tainted char *src,
          untainted char *dst,
          int len) {
  untainted int i;
  for (i = 0; i<len; i++) {
    dst[i] = src[i]: //illegal
untainted char    tainted char
}
```

Illegal flow :

tainted $\not\leq$ untainted

Implicit flows

```
void copy(tainted char *src,
          untainted char *dst,
          int len) {
  untainted int i, j;
  for (i = 0; i<len; i++) {
    for (j = 0; j<sizeof(char)*256; j++) {
      if (src[i] == (char)j)
        dst[i] = (char)j; //legal?
    }
  }
}
```

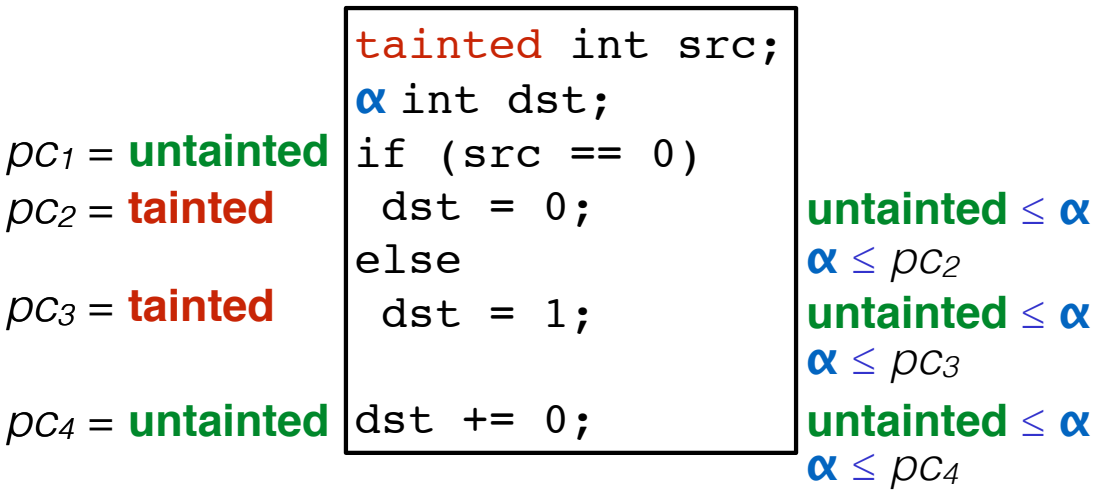
untainted char untainted char

Missed flow

Information flow analysis

- The prior flow is an **implicit flow**, since information in one value *implicitly* influences another
- One way to discover these is to maintain a scoped **program counter (*pc*) label**
  - Represents the maximum taint affecting the current pc
- Assignments generate constraints involving the *pc*
  - `x = y` produces two constraints:
    label(`y`) ≤ *label*(`x`) (as usual)
    pc ≤ *label*(`x`)
- **Generalized analysis** tracks **information flow**

Info flow example



Solution requires **α = tainted**

Discovers implicit flow

Why not information flow?

- Tracking implicit flows with a *pc* label can lead to **false alarms**
  - E.g., ignores values

```
tainted int src;
α int dst;
if (src > 0) dst = 0;
else         dst = 0;
```

- Extra constraints also **hurt performance**
- Our copying example is *pathological*
  - We typically don't write programs like this
  - Implicit flows will have little overall influence
- *So*: **tainting analyses tend to ignore implicit flows**