Type Qualifiers, Subtyping, and Type Inference

Jeff Foster

CMSC 631

September 17, 2002

Format String Vulnerabilities

- I/O functions in C use format strings

```
printf("Hello!");              Hello!
printf("Hello, %s!", name);    Hello, name !
```

- Instead of

```
printf("%s", name)
```

Why not

```
printf(name);          ?
```

Format String Attacks

- Adversary-controlled format specifier

```
name := <data-from-network>
printf(name);        /* Oops */
```

  - Attacker sets name = "%s%s%s" to crash program
  - Attacker sets name = "...%n..." to write to memory

- Lots of these bugs in the wild
  - New ones weekly on bugtraq mailing list
  - Too restrictive to forbid variable format strings

Types to the Rescue!

- Observation: two types of strings
  - strings that come from the network
  - strings that are used as format strings
  - ...but both have type char *
- Use qualified types to distinguish trust levels
  - tainted char * = may be from network
  - untainted char * = must not be from network
  - Rule: tainted data never used in untainted positions

Subtyping with Qualifiers

```
void f(tainted int);
untainted int a;
f(a);
```

OK

f accepts tainted or untainted data

untainted ≤ tainted

```
void g(untainted int);
tainted int b;
f(b);
```

Error

g accepts only untainted data

tainted ⋠ untainted

untainted < tainted

Demo

http://www.cs.berkeley.edu/~jfoster

Adding Type Qualifiers

- Work with simply-typed lambda calculus
  - (Talk about moving to C later)

e ::= c | x | λx:t.e | e1 e2 | ...

t ::= char | ptr(t) | t1 → t2

Adding Type Qualifiers

- Work with simply-typed lambda calculus
  - (Talk about moving to C later)

e ::= c | x | λx:t.e | e1 e2 | ...

t ::= Q s
s ::= char | ptr(t) | t1 → t2
Q ::= untainted | tainted | ...

  - Plus a partial order ≤ on Q

Adding Type Qualifiers

- Work with simply-typed lambda calculus
  - (Talk about moving to C later)

$$e ::= c \mid x \mid \lambda x{:}t.e \mid e1\ e2 \mid \ldots$$
$$\mid \text{annot}(e, Q) \mid \text{check}(e, Q)$$
$$t ::= Q\ s$$
$$s ::= \text{char} \mid \text{ptr}(t) \mid t1 \rightarrow t2$$
$$Q ::= \text{untainted} \mid \text{tainted} \mid \ldots$$

  - Plus a partial order $\leq$ on Q

Type Judgments

"In type environment A, expression e has type t"

Type Checking Rules — Characters

$$\frac{}{A \text{ ' } c : \text{char}}$$ No qualifier

$$\frac{A \text{ ' } e : s}{A \text{ ' } \text{annot}(e, Q) : Q\ s}$$

Example: A ' annot('a', tainted) : tainted char

Type Checking Rules — Qualifier Checks

$$\frac{A \text{ ' } e : Q'\ s \qquad Q' \leq Q}{A \text{ ' } \text{check}(e, Q) : Q'\ s}$$

Example: A ' check(annot('a', tainted), untainted) : t