Defenses Against Buffer Overflow

Guest Lecture by Aseem Rastogi (arastogi@cs.stonybrook.edu)

Fat Pointers
Introduce bounds in the same scope of each pointer

For instance

```c
int sum(int *p, int n, int *p_lo, int *p_hi)
{
    int sum=0;
    for(int i=0; i < n; i++)
    {
        sum += *(p+i);
    }
    return sum;
}
```

Inserted by compiler Automatically

Inserted assertions
`assert(p_lo ≤ p+i < p_hi);
assert(p+i != NULL);`

For locally declared variables
```c
int foo()
{
    int *p = malloc(2*sizeof(int));
    int *p_lo = p;
    int *p_hi = p+2;
    ...
    free(p);
    p_lo = NULL;
    p_hi = NULL;
    p = NULL;
}
```

For pointer assignments
```c
int *p = q;
int *p_lo = q_lo;
int *p_hi = q_hi;
```

For structs
```c
struct foo {
    int *a;
    int n;
    int *a_lo;
    int *a_hi;
};
```
```c
struct foo *p = malloc(sizeof(struct foo));
struct foo *p_lo = p;
struct foo *p_hi = p + 1;
```
```c
p->n = 3;
```

Advantages
- Fool proof (Sound)
- No false positives
  - Only crashes when an out of bounds pointer is dereferenced
- No effort on the part of programmers

Disadvantages
- Memory overhead
  - From storing new pointer bounds
- Performance overhead
  - For checking bounds on deference
- Libraries which use this approach are not compatible with code which doesn’t and vice versa
Deputy Concept

Deputy Concept

```c
int sum(int p, int n)
{
    int sum=0;
    for(int i=0; i < n; i++)
    {
        assert(p \leq p+i < p+n);
        assert(p+i != NULL);
        sum += *(p+i);
    }
    return sum;
}
```

Since \( n \) implicitly contains \( p_{lo} \) and \( p_{hi} \) we don't need them.
The programmer can specify the bounds of a pointer in terms of expressions in the same scope.

Call Site Checking

```c
int * p = malloc(2 * sizeof(int));
sum(p, 1);
```

Structure Checking

```c
struct foo {
    int * [BND(a, a+n)] a;
    int n;
}
```

Advantages

- Foolproof too
- No false positives
- Library compatibility

Disadvantages

- Overhead from assertions
  - Can remove assertions using static analysis
- Programmer overhead
  - Type inferences

Done to reduce load on programmer, the compiler inserts bounds automatically for some

```c
int * [BND(tmp_lo, tmp_hi] tmp = p->a;
int * tmp_lo = p->a;
int * tmp_hi = p->a + n;
```