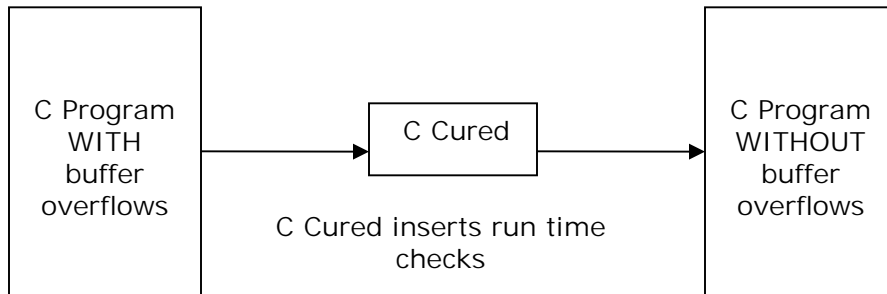


Reading assignment was reading paper on C – Cured

C Cured

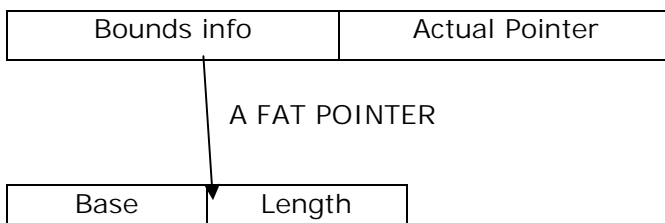


Overview: Some common mistakes that programmers in C make

1. Accessing / referencing outside of an array

Remedy: Bound check the pointer before each dereferencing operation.

How to do this? : FAT POINTER



In this case, the bounds info is the base address of array and length = 'number of bytes of array'

2. malloc / free bugs

e.g. freeing the same location may corrupt the heap

Remedy:

a. Don't use free() at all

b. Use garbage collection instead of free. (Caution: There's a probable security hole since garbage collector runs lazily and attacker may use this temporary extra space to inject his code. But considering its advantages over its disadvantages, garbage collection seems to be a good move.)

3. Casts and Unions

- Use runtime checks with casts.
- Disallow unions

e.g. of unsafe use of unions

```
union u
{ int a;
  char * ptr;
};
```

In this e.g., one can initialize the union with an integer and then dereference it assuming it to be a char * pointer.

e.g. of unsafe operation using cast

```
char * p = (char *) 5;
```

in this case p should not be dereferenceable.

How Fat Pointer helps ?

Casting int to pointer yields an undereferenceable Fat Pointer.

0 (base)	0 (length)	5 (value)
----------	------------	-----------

After type casting '(char *) 5', it will be converted into a Fat Pointer as stated above. This cannot be dereferenced.

RULE: A fat pointer with base = 0, length = 0 cannot be dereferenced.

Thus, C-Cure alters the C structures such that they are still usable as in C but are safer than in C.

Breaking up C in subsets with different levels of associated security:

Language SAFE:

Define SAFE to be subset of C with

- No pointer arithmetic
- No casts
- No union

Therefore, we get rid of security hazards due to

- Bounds checking
- Type checking
- No necessity to keep any kind of bounds information

e.g.

```
int linked_list_search(list * i, int t)
{
    While(i NOT EQUAL NULL && i->data NOT EQUAL t)
        i = i -> next;
    if(i)
        return true;
    return false;
}
```

Language SEQ = SAFE + Pointer Arithmetic:

e.g.

```
int sum(int * vec, int len)
{
    int acc = 0;
    int i;
    for(i = 0 ; i < len ; i++)
        acc += vec[i];
    return acc;
}
```

Define SEQ to be subset of C with

- bounds check
 - bounds information
 - we DON'T need type checks
-

Language C / Dynamic / Wild – Normal C language that we use

The idea is to find the part of the source code which falls in 1 of the 3 different languages and then process parts falling under languages as above stated 1,2,3 differently using secure.

1. C – Cured partitions user programs into

- SAFE
- SEQ
- WILD parts

2. Compiles each of these 3 parts separately

3. Handles the conversions at the boundaries of transition from one type of language to another.

-Partitions are performed using type qualifier inference.

-Analysis is sound (def of sound : Analysis assures on errors due to casting)

-Code should be free of buffer overflows.

Performance:

Overhead – 1.5 x slowdown

Statistically, a source program has:

SAFE pointers – 90 %

SEQ pointers – 9 %

WILD pointers – 1 %

BETTER THAN OTHER SIMILAR TOOLS – (e.g. purify slows 50 X times)

Binary compatibility issues:

CCured is not compatible with other libraries which are not compiled using CCured.

e.g. Non CCured compiled libraries may not have Fat pointers

- Need to convert pointers at interfaces between CCured and NonCCured code.

Proposed advancements:

For the interfaces between CCured and NonCCured code :

- Writing wrapper functions to unFatten the pointers
- Writing wrapper functions to Fatten the pointers back

Privilege Separation

Reading assignment was paper – privTrans

Principle of least Privilege :

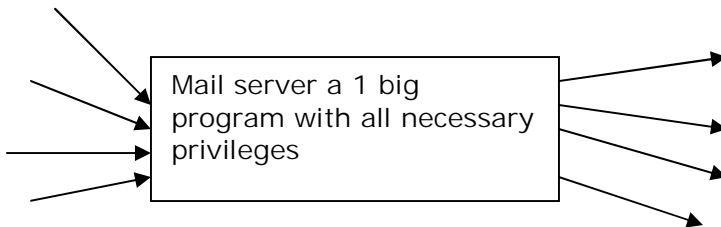
Only assign MINIMUM privilege necessary to do a certain task to the user

A program needs privilege for 3 reasons:

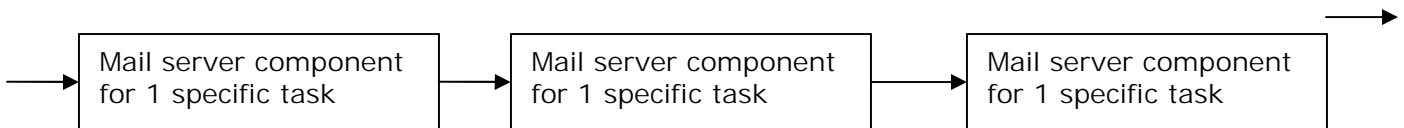
- Access secret data (e.g. read a file)
- perform special operation
- change identity

Ways to do privilege separation:

Divide a big program into smaller programs with different privilege for each.



BAD DESIGN



GOOD DESIGN

----- X X X -----
(END OF DOCUMENT)