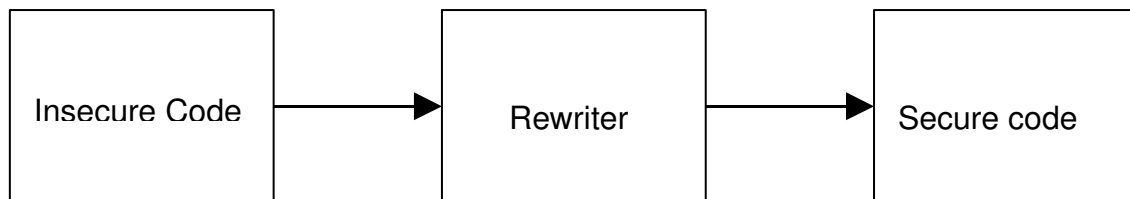


# Program rewriting tools

## Motivation for program rewriting tools

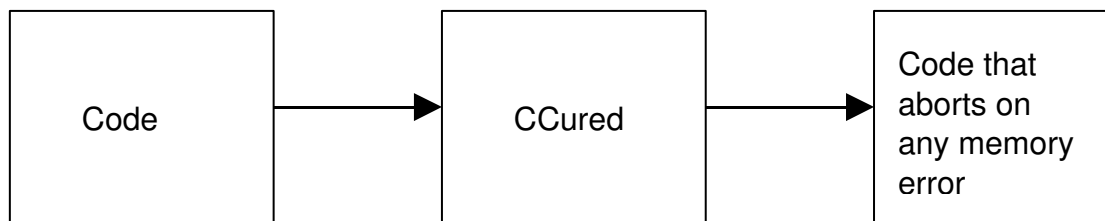
- 1) It is hard to fix all bugs manually.
- 2) It is hard to find all bugs with static analysis tools. Ex: Code that uses complex pointer arithmetic will be difficult to analyze statically.

Program rewriting tools work in the following way:



## CCured

CCured system targets type safety in C programs. CCured works in the following way.



CCured ensures type safety by defining sublanguages of C. The sublanguages differ in the following ways:

1. Types and operations allowed on the types.
2. To ensure memory safety, each language introduces checks on types and operations. Some of these checks are introduced during compile-time and some during runtime. These checks add to the runtime cost of the program. The languages differ in the kind of checks they introduce.

## SAFE language

This language is C with the following restrictions:

1. No casts except pointer to integer.
2. No pointer arithmetic.
3. No arrays.
4. No unions.

```

In a union say,
union foo{
    int a;
    char *b;
}

```

There is only one region of memory allocated for the members in the union. Size of the region is dependent on the size required for the largest member. It is the programmer's responsibility to use the members properly. In the union definition above, if the programmer is not careful, the program could end up using the integer in place of the pointer.

To guard against memory related errors, we need to introduce some checks in SAFE. In particular we need to guard against the following memory bugs:

1. NULL pointer access – To prevent this type of error, SAFE introduces NULL checks on pointer types.
2. Un-initialized pointers – To prevent this type of error, SAFE always initializes pointers to NULL.
3. Dangling pointers – Dangling pointers point to memory that has been de-allocated. Access to data pointed by dangling pointers may cause unpredictable behavior. To prevent this type of error, SAFE never de-allocates memory. In other words, calls to free are replaced by NOPs.

Input program segment	CCured 'SAFE' program
<pre> int x; int *p = &amp;x; char *q = malloc(1); int y = (int)q; free(q); *q = 5; </pre>	<pre> int x; int *p = &amp;x; char *q = malloc(1); int y = (int)q; // free(q); if(q == NULL)     abort(); *q = 5; </pre>

## SEQ Language

This language is same as SAFE with the following restrictions removed.

1. Pointer arithmetic is allowed.
2. Cast from integer to pointer is allowed.
3. Arrays are allowed.

## Making SEQ safe.

In addition to the checks introduced in the SAFE language, we need the following:

1. Bounds check on pointers.

To keep track of bounds, the pointer representation is changed to include the base address and size of the memory it points to.

So essentially, a pointer  $p$  is changed to the triplet  $\langle \text{base}, \text{size}, p \rangle$

When converting pointer to integer, we lose the base and size information. So, when converting the integer back to pointer, CCured prevents dereferencing of this pointer. This is done by representing the pointer as  $\langle 0, 0, p \rangle$ .

<b>Input program fragment</b>	<b>CCured 'SEQ' program</b>
<pre>int *p = malloc(10); int x; int *q = &amp;x; p += 5; *p = 0;</pre>	<pre>//t is a temporary //introduced by SEQ //compiler. int *t = malloc(10); int *p = &lt;t, 10, t&gt;; int *q = &lt;&amp;x, sizeof(x), &amp;x&gt;; p = &lt;t, 10, t + 5&gt;; if(p == NULL        p &lt; t    p &lt; t + 10 -     sizeof(*p))     abort(); *p = 0;</pre>

It is advantageous to compile functions in SAFE whenever possible - since the runtime costs associated with a SAFE program is lesser than SEQ. When this is done we need a way in which a SEQ compiled function can call SAFE compiled function. (Note that the pointer representations are different in SEQ and SAFE). This can be achieved by introducing wrappers at the interface and converting the pointer representations from SEQ to SAFE and vice-versa.

### Converting SEQ pointers to SAFE pointers

Can be done with following code

```
if( Pseq < base || Pseq >= base + size - sizeof(*Pseq))
    abort();
Psafe = Pseq;
```

### Converting SAFE pointers to SEQ pointers

Can be done with the following code fragment.

```
Pseq = < Psafe, sizeof(*Psafe), Psafe>;
```

## **DYNAMIC language**

This is SEQ with the following restrictions removed

1. Casts between pointer types are allowed.
2. Unions are allowed.

To make DYNAMIC safe, we need to have all checks present in SEQ. In addition to that we need a way to check the type of word that a pointer points to at runtime.

Consider the following program fragment

```
int **pp;
int *p;
p = malloc(10);
pp = (int **)p;
```

In the above code pp ends up pointing to the same memory location as p. We need a way to differentiate pointers from integers at runtime. This can be done by having the LSB of every word indicate its type. If LSB is 0 then the word represents a pointer. If LSB is 1 then the rest of the word represents an integer.

## **Results**

As is evident from the discussion so far, CCured programs have a runtime cost associated with them. Experiments have shown that programs run about 50% slower. They may also require substantial modification to the code for proper compilation. If source is not available for all the libraries used by the program, then the libraries are not 'CCured', and can potentially introduce bugs in the program.