

Ccure (continued)

- SAFE
- SEQ
- DYNAMIC

SEQ + ptr-to-ptr casts

E.g:

```
Struct A {Int x;}
```

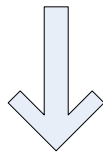
```
Struct B {int *p;}
```

```
Void foo(void)
```

```
{  
    struct A a={5};  
    struct B *b=(struct B*)&a;  
    *(b->p);  
}
```

Solution:

Integers: x represented as 2x
Pointers: p represented as 2ptr



Encodes dynamic type of
memory word into lsb of word

```
Void bar(void)
```

```
{  
    int *px;  
    Int **ppx;  
    Px=malloc(...);  
    Ppx=px;  
    While(1)  
    {  
        T=*px;  
        If(t&1) abort();  
        *px=0;  
    }
```

- SAFE: none/null check (80%)
- SEQ: deref => bcheck, deref
Assign => assign, bounds assign
(10-15%)

```
Int A[5];  
Int *x;  
Int *&x, *&hx;  
X=&A[c]  
&x=&A[0];  
Hx=&A[4];  
For(i=0; i<5; i++){  
    If(x<&x || x>hx)  
        abort();  
    *x=0;  
    X++;  
}
```

- DYNAMIC

Deref => bcheck, dynamic
typecheck, deref

Assign => assign, bounds
assignment

(5%)

```
Int A[80];  
For(i=0; i<0x80; i++)  
A[i]=0;
```

Bug: 0x80 (buffer overflow)

Total overhead: 50-60%

Overhead: pretty good

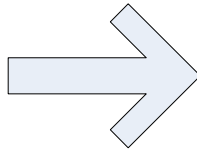
Compatibility: not the best

Effort: not good (need to rewrite)

Soundness: good

```
Struct A{  
    int array[5];  
    intx;  
}
```

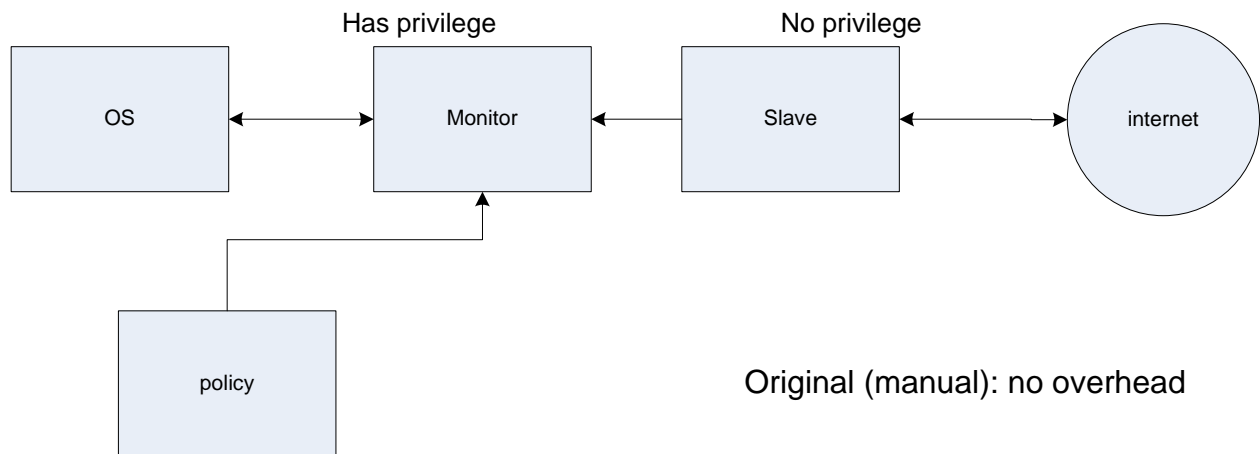
```
Struct A a;  
Int *p=&a.array;  
P[5]=0;
```



Work with gcc but not
CCure

Privilege separation

- Goal: minimize amount of code that runs with privilege
- makes code audit easier
- limit damage from break in
- may make verification possible



Open SSH policy

