

Buffer Overflows-System Solutions:  
Stack Guard:

```
void getUser(int fd)
{
char username[1024];
read(fd,username,2048);
if(canary!=CANARY_VALUE) abort();
return;
}
```

fd
Return address
Canary
User name

Attacker must guess canary value to succeed. Probability of success=  $2^{-32}$

Good

- very easy
- backwards compatible
- overhead 10-100%

Bad

- can't heal all vulnerabilities, eg: format string bugs
- data corruption attacks
- other function pointers
- heap corruption

Point Guard:

Suppose each word had a bit indicating whether it is a pointer or not:

```
void getUser(int fd)
{
char username[1024];
read(fd,username,2048);
return;
}
```

...	
0	Fd
1	Return address
0	User name
...	

After Overflow:

...	
0	...
0	Over written
0	Over written
...	
...	

When overflow occurs, the extra bit indicating pointer variables gets over written too. Instead point guard encrypts pointers in memory. (eg. retaddr X-OR mask m)

Fd
Retaddr Xor m
username

After overflow:

...
Attacker's return address
Overwritten ...

So when the return address is overwritten by the attacker's return address, system will return to (attacker\_ret Xor m) which is wrong and the system may crash.

### Good

Not much overhead (<20%)

### Bad

-very backwards incompatible (difficult to link point guarded and non point guarded code)

-brute forcible in some cases

-data corruption attacks

-fails unpredictably

### Point guard example:

```
void getUser(int fd)
{
char *p;
char username[1024];
p=malloc();
read(fd,username,2048);
return;
}
```

Would be written as:

```
call malloc;
Xor %r0, %r0, %r31(<- mask)
store %sp+1024, %r0
call read
load %r0
Xor %r0, %r0, %r31
store %r0,0
```

Address Space Randomization:

- Force attacker to guess addresses.
- Simplest: move entire sections
- At most 20 bits of entropy per section
- And some attacks need to guess only one

Stack
.....
mmap
.....
Code
Code
.....
.....

More Randomization:

- randomize each subsection
- location of each function
- stack frame padding
- reorder locals
- insert padding between locals
- reorder arguments( and pad?)
- order of struct fields? NO