Previously: Stack guard, pointer guard

**ASR - Address Space Randomization**

**Classif Stack Smash:**

```
+------+-----------
| stack | overview  |
|       |           |
+------+
    |     |
    | but |

Assumption: attacker knows approx layout of memory

Idea: Take away this knowledge

Easy to change stack start in memory
- just a pointer

But trampoline attack can beat this!

To prevent trampoline, move around `text` and `mmap` areas, also.

Libs are designed to be loaded anywhere in memory
- have to be aligned on the 4 KB page boundary
- upper 4 bits also fixed
- so effectively, 16 bits.

Heap start can be randomized. [malloc random no of bytes]

Relative position within a segment doesn't change!

Sleep() system call + Fixed offset = System() system call

Paper suggests Brute Force + Ret to Lib C.

16 bits of randomization ≈ 216 (4 mins) of security?! 

Attacker may not need to guess all randomness simultaneously.

Soln: Use 64 bits system ≈ 40 bits of randomization

≈ 2000 years!
Internal randomizations

- Stack
  - Local variable ordering
  - Inter-frame padding
  - Frame ordering (maybe)
  - Parameter ordering (?!??)
  - Intra-frame padding

- Code Segment
  - Function ordering
  - Padding
    - Use many inabs which do the same thing (randomly use these)
  - Polymorphic code? Just set randomization?

- Heap
  - Inter allocation padding
  - Inter allocation ordering
  - Program level data structures

- If attacker is local, he can see the executable to break the compile-time
- Re-randomization doesn't help
- ASR successful because programmer has to do nothing to take adv of it
- OS vendors do the work for programmers. Incentive to vendors - separation