Computer Security - CS 261 (Notes for 09/02)

1 gets - no bounds checking. Can cause stack to be overwritten.

2 Phrack landmark paper on stack overflow - "Smashing the stack for fun and profit" by AlephOne

3 How do you know the absolute address to be places in teh return address on the stack?

- Based on architecturem program version and so on, you can find the approximate range in which the stack is present.
- Add a landing slide in the malicious code, i.e. prepend NOP instructions in the program and start guessing the address.

4 If the stack is growing the other way around, how to do the attack?

- If the return address is placed after the local variables, you can still over write the caller's return address.
- When passing stack allocated buffers to vulnerable functions, the attack is still valid, irrespective of which way the stack grows.

5 Defences against stack smashing

- Can we check the integrity of the stack? - Canaries

6 Heap Smashing

- Malloc maintains 2 seperate buffers
  - A seperate buffer for maintaining the data structures of allocated buffers
  - And a seperate buffer of alloted memory buffers themselves
- If maintenance data structures are stored with the buffer, for e.g., if 3 buffers are maintained as a doubly linked list (next pointer pointing to next allocated buffer and the prev pointer pointing to the previously allocated buffer),
  - Then we could possibly over write the header data structure of the next data buffer by over flowing the current buffer.
  - This gives the felixibility of writing any data you want into any location you want.
    - Code segement, function pointers, GOT, stack, etc can be overwritten
  - Defence: Put a guard page (not mapped) after each buffer, randomise location of each buffer

7 Other attacks

- Return oriented programming - stitch together snippets of existing code segments that end in a return

instruction
    - Typically you can form a turing complete set of sequences from these short code segments
    - Still valid attack even if the code segment is the only executable section and the data segment is the only writable section.
        - NX, W^X defence resistant
    - Does not work with address randomisation, since the address of the stack has to be known deterministically (for filling in the frame pointer value as well knowing where the libraries are laid out in the virtual memory space).
        - ASLR (address space layout randomisation) will defeat the attack
- Double free attack -> If there is a control path that causes a buffer to be freed more than once, the attack will try to enforce that path.
    - If you can use stack smashing to over write the header data structures of a heap buffer, then you can cause a free to actually free arbitrary memory location, i.e. any address that you put into the data structure.
    - We can put the data structures into a seperate location. But at the expense of spatial locality.
    - Switch to a language with garbage collection
    - Singularity, FoxOS were written in higher level language
- Compiler extensions, new data types, String class does buffer checks, new libraries which take buffer length all help defend attacks
- Command Injection (Caused by passing signalling information and data in the same channel)
    - Shell Injection
        - system ("mail " + $emailaddr)
            - ';', '&', '|', etc should be disallowed in email id
            - Escape characters in email id
            - Should not be using system. Should be using fork() and exec()
    - SQL Injection
        - SELECT * FROM users WHERE name=$username AND passwd=$pwd
            - password = "; DROP TABLES *"
            - user name = "administrator –"
                - – is the comment character
            - user name = "foo OR 1=1 –"