Notes for 11/30/09

Memory exploits in general... storing management information in band with data information:
[buffer] [management information]

Commonly, the buffer will be a stack buffer, and management information will be the return address of the function, but there are other variations

Consider the Code Red worm, which infected Microsoft IIS Web servers...
Overwrote an exception handler on the stack that was triggered when given an incomplete unicode character encoding
The buffer is overflowed when IIS does not reserve enough space for multi-byte characters

Buffer overflows on the heap
Consider the common heap data structure... a doubly linked list...

```
/* from glibc */
struct malloc_chunk {
  INTERNAL_SIZE_T      prev_size;  /* Size of previous chunk (if free).  */
  INTERNAL_SIZE_T      size;       /* Size in bytes, including overhead. */

  struct malloc_chunk* fd;         /* double links -- used only if free. */
  struct malloc_chunk* bk;
. . .
};

#define unlink(P, BK, FD) { /* called on, e.g., free() */
  FD = P->fd
  BK = P->bk
  . . .
  FD->bk = BK
  BK->fd = FD
  . . .
}
```

Double-free vulnerability:
Overflow heap buffer to overwrite fd and bk on heap to write arbitrary data to arbitrary address on free():
So, e.g., set...
fd = write location – 12
bk = what to write

Format string vulnerabilities:

```
printf ("%s", input); /* good */
printf (input); /* bad... what if input has a %foo code? */
```

Suppose input is "%...x%...x%...x%n"... %n says, take given address argument, and then store the number of characters printed thus far to that address
So you can pass an address for %n on the stack that points to where you want to write. Then you can put arbitrary values in between the %...x codes that add up to the value that you want to write there.
Now you can write an arbitrary value to an arbitrary address in memory.