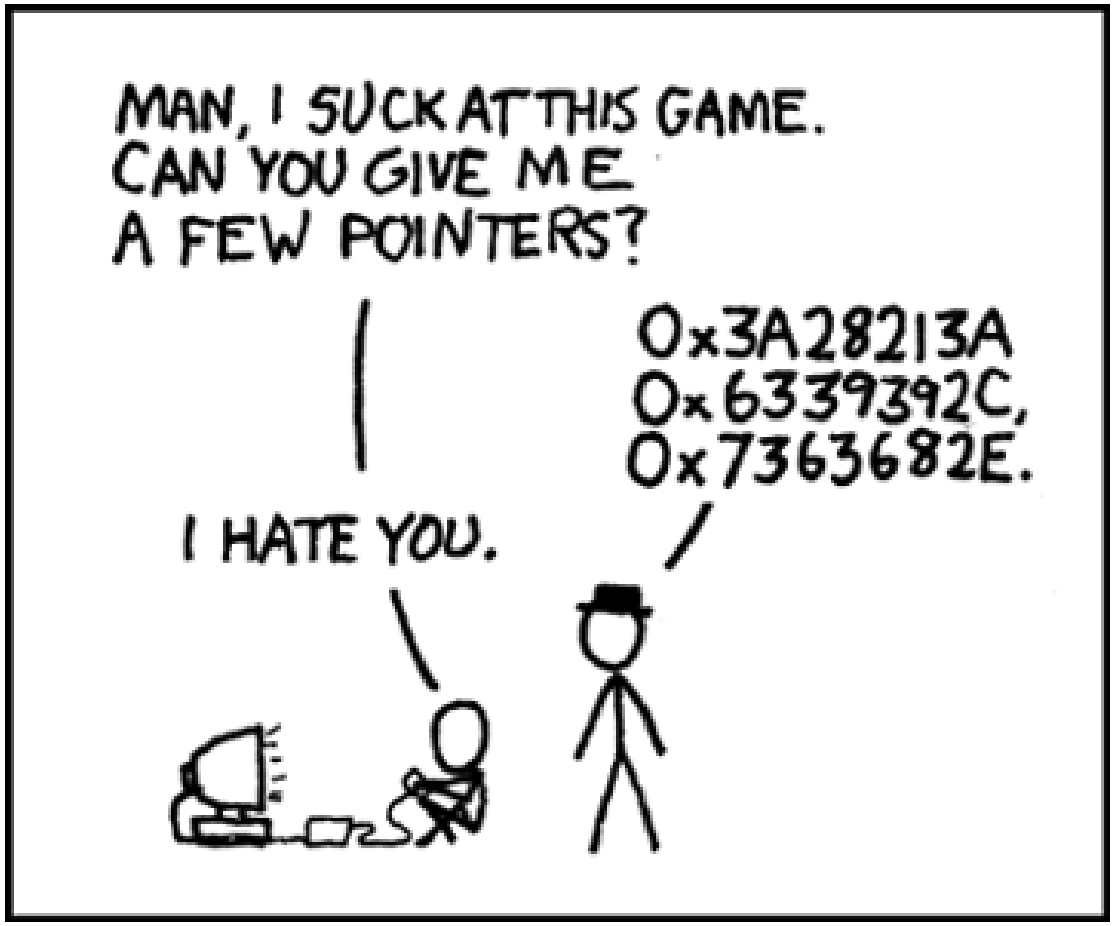
Integer Overflows

Luke Gotszling
lmg@vicecloud.net



http://www.xkcd.com/c138.html

What are they?

- Subset of all buffer overflows
- An effect of integers being stored as two's complement

sign bit									
0	1	1	1	1	1	1	1	=	127
0	0	0	0	0	0	1	0	=	2
0	0	0	0	0	0	0	1	=	1
0	0	0	0	0	0	0	0	=	0
1	1	1	1	1	1	1	1	=	-1
1	1	1	1	1	1	1	0	=	-2
1	0	0	0	0	0	0	1	=	-127
1	0	0	0	0	0	0	0	=	-128

8-bit two's complement integers

http://en.wikipedia.org/wiki/2s_compliment

Danger Danger

- Q: What happens when I add 1 to 32767 in a 2 byte signed integer?
- A: -32768

What if I relied on that integer to be greater than 0? (and only checked if it was >0)
What if the integer indicated how many bytes of memory to allocate?
What if the integer is a pointer?

Example 1-Arithmetic overflow

```
/* ex2.c - an integer overflow */
    #include <stdio.h>

    int main(void){
            unsigned int num = 0xffffffff;

            printf("num is %d bits long\n", sizeof(num) * 8);
            printf("num = 0x%x\n", num);
            printf("num + 1 = 0x%x\n", num + 1);

            return 0;
    }
    /* EOF */
```

Phrack #60 0x0a

The output of this program looks like this:

```
num is 32 bits long
num = 0xffffffff
num + 1 = 0x0
```

Example 2-Loss of Precision

```
/* ex1.c - loss of precision */
#include <stdio.h>

int main(void){
        int l;
        short s;
        char c;

        l = 0xdeadbeef;
        s = l;
        c = l;

        printf("l = 0x%x (%d bits)\n", l, sizeof(l) * 8);
        printf("s = 0x%x (%d bits)\n", s, sizeof(s) * 8);
        printf("c = 0x%x (%d bits)\n", c, sizeof(c) * 8);

        return 0;
}
/* EOF */
```


The output looks like this:

```
l = 0xdeadbeef (32 bits)
s = 0xffffbeef (16 bits)
c = 0xffffffef (8 bits)
```

Example 3-Memory allocation

```
// cbSize is an unsigned 32-bit type

bool func(size_t cbSize) {
   if (cbSize < 1024) {
      // we never deal with a string trailing null
      char *buf = new char[cbSize-1];
      memset(buf,0,cbSize-1);

      // do stuff

      delete [] buf;

      return true;
   } else {
      return false;
   }
}
```

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dncode/html/secure04102003.asp

Guess how much memory could potentially be allocated and overwritten and win a prize!

More memory allocation

```
short int bytesRec = 0;
char buf[SOMEBIGNUM];

while(bytesRec < MAXGET) {
  bytesRec += getFromInput(buf+bytesRec);
}
```

http://www.owasp.org/index.php/Integer_overflow

What could happen?

Real life integer overflows

June 2002: Memory corruption vulnerability in Apache
http://www.securityfocus.com/bid/5033

June 2002: Challenge-response buffer overflow in OpenSSH
http://www.securityfocus.com/bid/5093

August 2002: XDR libraries (included libc/glibc)
http://www.cert.org/advisories/CA-2002-25.html

August 2002: FreeBSD signed integer buffer overflow vulnerability
http://www.securityfocus.com/bid/5493

March 2003: XDR libraries again! (likewise included libc/glibc)
http://www.cert.org/advisories/CA-2003-10.html

March 2003: Windows Script Engine for JavaScript remote code execution
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0010

Avoiding the pain and suffering

- **If your code performs any kind of integer manipulation (addition, multiplication, and so on) where the result is used to index into an array or calculate a buffer size, make sure the operands fall into a small and well-understood range.**

- Compile C and C++ code with the highest possible warning level, /W4.

- Be wary of signed arguments to memory allocation functions (new, malloc, GlobalAlloc, and so on) because they are treated as unsigned integers.

- Finally, if you are using managed code, make sure you catch OverflowExceptions, if appropriate.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dncode/html/secure04102003.asp

More code review points:
http://developer.apple.com/documentation/Security/Conceptual/SecureCodingGuide/Articles/BufferOverflows.html
https://buildsecurityin.us-cert.gov/daisy/bsi/articles/knowledge/coding/308.html?branch=1&language=1
http://www.owasp.org/index.php/Integer_overflow

Demo (int_wrap.c)