Prolog to Lecture 9
CS 236
On-Line MS Program
Networks and Systems Security
Peter Reiher

Buffer Overflows

- Buffer overflows are a big problem
- One class of defenses concentrates on not allowing attacker to write
  - Don't let him put his attack code in your memory
- If he can't write his attack code, he can't attack you
  - Can he . . .?

Return Oriented Programming

- Unfortunately, he can
- A technique called *return oriented programming* allows it
- How?
- Attacker doesn't insert new code
- He makes use of code already there

The Basic Idea

- Attacker overwrites the stack
  - Which needs to be writeable
  - But not necessarily executable
- Overwrites correct return addresses with new ones
- Addresses pointing to code in your system that does attacker's job for him

How Likely Is That?

- How likely is it that I have code lying around that does what attackers want?
- How likely is it that they can find it and use it this way?
- Unfortunately, not just likely, but certain

The Return Oriented Technique

- Don't look for one big piece of code that does what you want
- Find lots of little pieces you can stitch together
- In something you know will be there
  - Like the C libraries

Can This Really Work?

- Yes
- This technique has hacked a voting machine and Adobe Acrobat
  - Of course, practically any attack technique seems to work on Acrobat
- Researchers have built "compilers" that create arbitrary programs this way
  - Out of bits of C libraries

The Implications

1. Techniques based on prevention of code injection are insufficient
2. More broadly, proposed solutions to security problems need to be examined very carefully