Low-level security

or

C and the infamous **buffer overflow**

What is a buffer overflow?

- A buffer overflow is a **bug** that affects low-level code, typically in **C** and **C++**, with **significant security implications**
- **Normally**, a program with this bug will simply **crash**
- But an **attacker** can alter the situations that cause the program to **do much worse**
  - **Steal** private information (e.g., Heartbleed)
  - **Corrupt** valuable information
  - **Run code** of the attacker's choice

Thanks to Dave Levin for many slides in this deck

Why study them?

- Buffer overflows are still **relevant** today
  - C and C++ are still popular
  - Buffer overflows still occur with regularity
- They have a **long history**
  - Many different approaches developed to defend against them, and bugs like them
- They share **common features with other bugs** that we will study
  - In **how the attack works**
  - In **how to defend against it**

C and C++ still very popular

Language Rank	Types	Spectrum Ranking
1. Java		100.0
2. C		99.2
3. C++		95.5
4. Python		93.4
5. C#		92.2
6. PHP		84.6
7. Javascript		84.3
8. Ruby		78.6
9. R		74.0
10. MATLAB		72.6

http://spectrum.ieee.org/static/interactive-the-top-programming-languages

Critical systems in C/C++

- Most **OS kernels** and utilities
  - fingerd, X windows server, shell
- Many **high-performance servers**
  - Microsoft IIS, Apache httpd, nginx
  - Microsoft SQL server, MySQL, redis, memcached
- Many **embedded systems**
  - Mars rover, industrial control systems, automobiles

A successful attack on these systems is particularly dangerous!

History of buffer overflows

The harm has been substantial

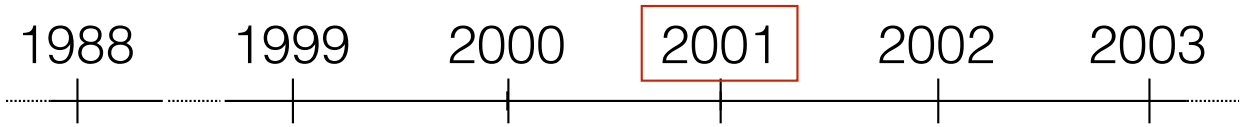
1988 1999 2000 2001 2002 2003

- **Morris worm**
  - Propagated across machines (too aggressively, thanks to a bug)
  - One way it propagated was a **buffer overflow** attack against a vulnerable version of `fingerd` on VAXes
    - Sent a special string to the finger daemon, which caused it to execute code that created a new worm copy
    - Didn't check OS: caused Suns running BSD to crash
  - End result: $10-100M in damages, probation, community service

Morris now a professor at MIT

History of buffer overflows

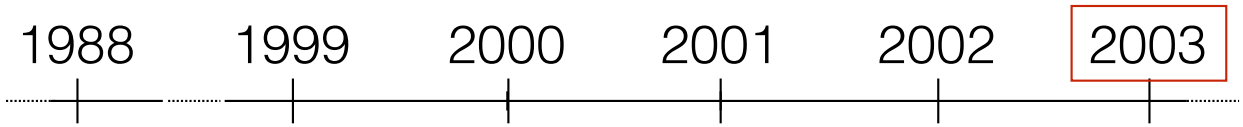
The harm has been substantial



- **CodeRed**
  - Exploited an overflow in the MS-IIS server
  - 300,000 machines infected in 14 hours

History of buffer overflows

The harm has been substantial



- **SQL Slammer**
  - Exploited an overflow in the MS-SQL server
  - 75,000 machines infected in 10 *minutes*

Slashdot

Chann

stories

submissions

popular

blog

ask slashdot

book reviews

games

idle

yro

technology

23-Year-Old X11 Server Security Vulnerability Discovered

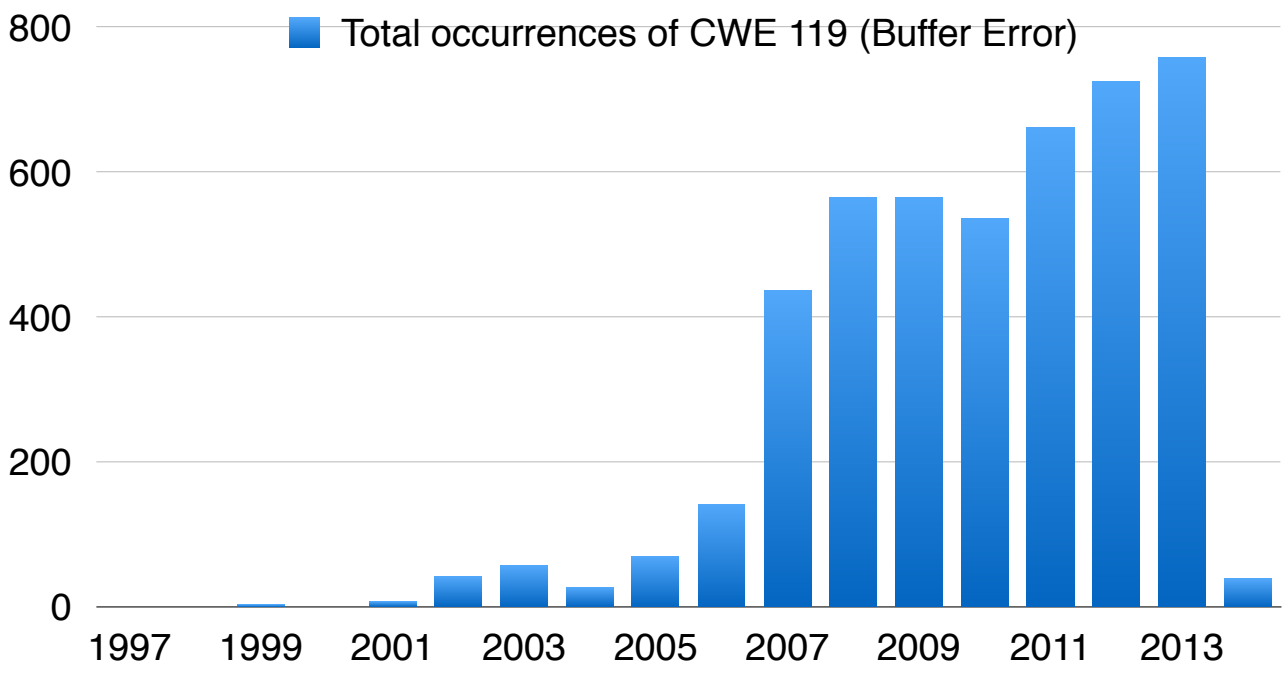
Posted by **Unknown Lamer** on Wednesday January 08, 2014 @10:11
from the stack-smashing-for-fun-and-profit dept.

An anonymous reader writes

> "The recent report of X11/X.Org security in bad shape rings more truth today. The X.Org Foundation announced today that they've found a X11 security issue that dates back to 1991. The issue is a possible stack buffer overflow that could lead to privilege escalation to root and affects all versions of the X Server back to X11R5. After the vulnerability being in the code-base for 23 years, it was finally uncovered via the automated cppcheck static analysis utility."

There's a `scanf` used when loading BDF fonts that can overflow using a carefully crafted font. Watch out for those obsolete early-90s bitmap fonts.

Trends



http://web.nvd.nist.gov/view/vuln/statistics

http://cwe.mitre.org/top25/

This is a brief listing of the Top 25 items, using the general ranking.

NOTE: 16 other weaknesses were considered for inclusion in the Top 25, but their general scores were not high enough. They are listed in a separate "On the Cusp" page.

Rank	Score	ID	Name
[1]	93.8	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[2]	83.3	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[3]	79.0	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[4]	77.7	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[5]	76.9	CWE-306	Missing Authentication for Critical Function
[6]	76.8	CWE-862	Missing Authorization
[7]	75.0	CWE-798	Use of Hard-coded Credentials
[8]	75.0	CWE-311	Missing Encryption of Sensitive Data
[9]	74.0	CWE-434	Unrestricted Upload of File with Dangerous Type
[10]	73.8	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[11]	73.1	CWE-250	Execution with Unnecessary Privileges
[12]	70.1	CWE-352	Cross-Site Request Forgery (CSRF)

What we'll do

- Understand how these attacks work, and how to defend against them
- These require knowledge about:
  - The compiler
  - The OS
  - The architecture

Analyzing security requires a whole-systems view

Note about terminology

- I use the term **buffer overflow** to mean ***any access of a buffer outside of its allotted bounds***
  - Could be an over-*read*, or an over-*write*
  - Could be during *iteration* ("running off the end") or by *direct access* (e.g., by pointer arithmetic)
  - Out-of-bounds access could be to addresses that *precede* or *follow* the buffer
- **Others sometimes use different terms**
  - They might reserve buffer overflow to refer only to actions that write beyond the bounds of a buffer
    - Contrast with terms *buffer underflow* (write prior to the start), *buffer overread* (read past the end), *out-of-bounds access*, etc.