CS590-LBS Language-based Systems Software Security

aka “The Eternal War in Memory”

Fall semester 2014

Mathias Payer
Purdue University

Physical security

- Indoor alarm: leave the building
- Outdoor alarm: seek shelter, look for updates
- Call 911 for any emergency

- Low probability that something happens!

Software is unsafe and insecure

- Low-level languages (C/C++) trade type safety and memory safety for performance
  - Programmer responsible for all checks
- Large set of legacy and new applications written in C / C++ prone to memory bugs
- Too many bugs to find and fix manually
  - Protect integrity through safe runtime system

Course overview

- C/C++ bugs lead to security vulnerabilities
  - Buffer overflow, integer overflow, unchecked format string, improper null termination, double free, …
- Large set of security policies exist, only few adopted in practice
- Goal: understand system security
  - Trade-offs between protection and performance
  - Requirements for strong policies

Logistics

- Course website:
  - http://nebelwelt.net/teaching/14-memory-war.html

- Course has two parts:
  - Discussion of security policies
  - (Small) research project

Logistics: security policies

- Everybody presents 2-3 papers
  - Select papers that you find interesting
  - Goal: 20min presentation, 20min discussion
  - Write a 1 page summary of the discussion

Logistics: research project

- Select a discussed security property, either
  - Design and implement an extension of an existing policy
  - Develop metrics and implement a benchmark that measures effectiveness or overhead of property

Area: Memory Safety

- Language based: CCured, Cyclone
- Compiler based: SoftBounds, BBC, ASAN, CETS
- Runtime system: Cling, DieHard, DieHarder, MemCheck, Valgrind

Area: Data and Pointer Integrity

- Write Integrity Testing (WIT)
- BodyArmor
- PointGuard

Area: Randomization

- Data Space Randomization (DSR)
- ASLR
- Binary stirring
- Instruction Set Randomizatoin (ISR)
- Instruction Layout Randomizatoin (ILR)

Area: Data & Control-Flow Integrity

- Data-Flow Integrity (DFI)
- Control-Flow Integrity (CFI)
- XFI: eXtended Flow Integrity
- HyperSafe
- Code-Pointer Integrity (CPI)

Area: Dynamic Policies

- Safe Loading
- Secure Execution via Program Shepherding
- SysCall policies

Area: Software-based Fault Isolation

- PittSFIeld
- Native Client (NaCL)

Summary

- Security policies exist in many different areas
  - Memory safety
  - Data and Pointer Integrity
  - Randomization
  - Data-Flow and Control-Flow Integrity
  - Dynamic Policies
  - Software-based Fault Isolation
- Different trade-offs and performance characteristics

Your todos:

- Read “Eternal War in Memory” paper
- Look through reading material for policies that are interesting to you
- Think about possible groups/solo projects
- Papers and groups will be allocated on Wednesday!

Questions?