Week 2

Help

Defenses against low-level attacks

We will continue our discussion of low-level software security by understanding ways to defend against memory-based attacks like buffer overflows and format string attacks, introduced last week.

Defenses fall into two categories: *automatic*, and *manual* (based on disciplined programming styles). We will also look at a sophisticated attack, called *return oriented programming*, that aims to overcome some of the automatic defenses, as well as an experimental defense against it. In the end, the most sure defense against low level attacks is to program with a *memory-safe* (or better yet, a *type-safe*) programming language in the situations that's possible.

Learning Objectives

After the completion of this week's material, you will be able to:

- Comprehend the meaning of the properties *memory safety*, and *type safety* and why programs enjoying these properties are immune from memory based attacks
- Understand several common automatic defenses against memory-based attacks, including *stack canaries*, *data execution protection (DEP)*, and *address space layout randomization (ASLR)*
- Understand how attacks based on *return-oriented programming (ROP)* work
- Understand the concept of *control-flow integrity (CFI)* and how it can defeat ROP-based attacks
- Understand a series of rules of thumb for programming in C so as to avoid memory-based attacks

Video Lectures

- Defenses Against Low-Level Attacks: Introduction (2:58)
- Memory Safety (16:56)
- Type Safety (4:39)
- Avoiding Exploitation (9:38)
- Return-Oriented Programming (11:13)
- Control-Flow Integrity (14:53)
- Secure Coding (18:29)

Required Readings

The following two blog posts cover the topics of memory safety and type safety in somewhat greater depth

- What is memory safety?
- What is type safety?

Quiz

The quiz for this week covers all of the material for this week. You must submit the quiz no later than the start of week 4. You will have three attempts to complete the quiz, at two hours per attempt. It consists of 15 questions, and if you are well versed in the material it should take about 30 minutes (but longer if you have to go back and look things up, obviously).

Supplemental readings and links

The following readings are optional: Check them out if you are interested in learning more about material we've covered in lecture (many were explicitly linked in the lecture slides).

Attacks and modern defenses, generally

- On the effectiveness of Address Space Randomization, by Shacham, Page, Pfaff, Goh, Modadugu, and Boneh - showed how ASLR implementations on 32-bit systems can be defeated relatively easily
- Smashing the Stack in 2011 - Paul Makowski revisits the 1996 Aleph One article (on the supplemental reading list from last week), considering modern defenses
- Low-level software security by example, by Erlingsson, Younan, Piessens, describes several low-level attacks and modern defenses.

Return-oriented Programming (ROP)

- Geometry of Innocent Flesh on the Bone: Return to libc without Function Calls (on the x86), by Hovav Shacham - introduced the idea, and the term, *return oriented programming*
- Q: Exploit Hardening Made Easy, by Schwartz, Avgerinos, and Brumley - explains how to automatically generate ROP exploits
- Blind ROP - return-oriented programming without source code, automatically

Control-flow integrity (CFI)

- Control Flow Integrity, by Abadi, Budiu, Erlingsson, and Ligatti - paper that introduced CFI
- MoCFI, by Niu and Tan - modular CFI that has low overhead
- See also the paper *Low-level software security by example*, above

Secure coding

These are a few references linked in the lecture slides. We will cover secure coding and design in more depth during week 4.

- CERT C coding standard
- Secure Programming HOWTO by David Wheeler
- Robust Programming by Matt Bishop
- DieHard project - drop-in replacement for `malloc` that uses randomization to defend against heap-based exploits

Project

There is **no new project** this week. Don't forget that Project 1 on exploiting buffer overflows, issued last week, is *due at the end of this week*. Take the on-line assessment when you have completed that project.

Created Wed 9 Apr 2014 6:15 AM PDT

Last Modified Sun 2 Nov 2014 1:15 PM PST