CSE 484 (Winter 2008)

Software Security: Attacks, Defenses, and Design Principles

Tadayoshi Kohno

Thanks to Dan Boneh, Dieter Gollmann, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Goals for Today

- Defensive Mechanisms
  - Languages
  - System changes
- Software Development Design Principles
  - (Programming practices)

Preventing Buffer Overflow

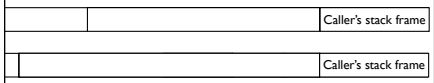
- Use safe programming languages, e.g., Java
  - What about legacy C code?
- Mark stack as non-executable
- Randomize stack location or encrypt return address on stack by XORing with random string
  - Attacker won't know what address to use in his or her string
- Static analysis of source code to find overflows
- Run-time checking of array and buffer bounds
  - StackGuard, libsafe, many other tools
- Black-box testing with long strings

Non-Executable Stack

- NX bit on every Page Table Entry
  - AMD Athlon 64, Intel P4 "Prescott"
  - Code patches marking stack segment as non-executable exist for Linux, Solaris, OpenBSD
- Some applications need executable stack
  - For example, LISP interpreters
- Does not defend against return-to-libc exploits
  - Overwrite return address with the address of an existing library function (can still be harmful)
- …nor against heap and function pointer overflows
- …nor changing stack internal variables (auth flag, ...)

Run-Time Checking: StackGuard

- Embed "canaries" in stack frames and verify their integrity prior to function return
  - Any overflow of local variables will damage the canary



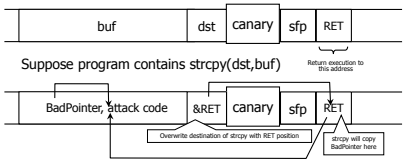
- Choose random canary string on program start
  - Attacker can't guess what the value of canary will be
- Terminator canary: "\0", newline, linefeed, EOF
  - String functions like strcpy won't copy beyond "\0"

StackGuard Implementation

- StackGuard requires code recompilation
- Checking canary integrity prior to every function return causes a performance penalty
  - For example, 8% for Apache Web server
- PointGuard also places canaries next to function pointers and setjmp buffers
  - Worse performance penalty
- StackGuard can be defeated!
  - Phrack article by Bulba and Kil3r

Defeating StackGuard (Sketch)

- Idea: overwrite pointer used by some strcpy and make it point to return address (RET) on stack
  - strcpy will write into RET without touching canary!



Run-Time Checking: Libsafe

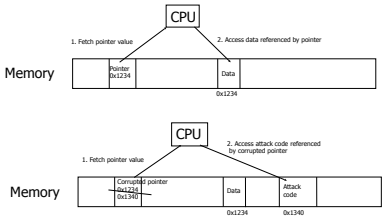
- Dynamically loaded library
- Intercepts calls to strcpy(dest,src)
  - Checks if there is sufficient space in current stack frame

    |frame-pointer – dest| > strlen(src)
  - If yes, does strcpy; else terminates application

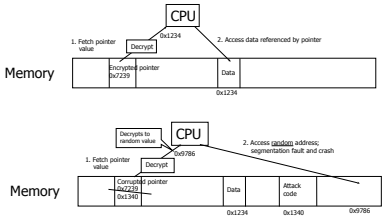
PointGuard

- Attack: overflow a function pointer so that it points to attack code
- Idea: encrypt all pointers while in memory
  - Generate a random key when program is executed
  - Each pointer is XORed with this key when loaded from memory to registers or stored back into memory
    - Pointers cannot be overflown while in registers
- Attacker cannot predict the target program's key
  - Even if pointer is overwritten, after XORing with key it will dereference to a "random" memory address

Normal Pointer Dereference [Cowan]



PointGuard Dereference [Cowan]



Fuzz Testing

- Generate "random" inputs to program
  - Sometimes conforming to input structures (file formats, etc)
- See if program crashes
  - If crashes, found a bug
  - Bug may be exploitable
- Surprisingly effective

- Now standard part of development lifecycle, e.g., for IE

Genetic Diversity

- Problems with Monoculture
- Steps toward diversity
  - Automatic diversification of compiled code
  - Address Space Randomization

Principles

- Check inputs

Principles

- Least privilege

Principles

- Check all return values

Principles

- Securely clear memory (passwords, keys, etc)

Principles

- Failsafe defaults

Principles

- Reduce size of TCB
- Simplicity
- Modularity

Principles

- Open design? Open source?
- Maybe...
- Linux Kernel Backdoor Attempt: http://www.freedom-to-tinker.com/?p=472

Vulnerability Analysis and Disclosure

- What do you do if you've found a security problem in a real system?
- Say
  - IM client?
  - Electronic voting machine?
  - ATM machine?
  - Hospital drug (morphine) pump