Glossary

Help

We use many technical terms in this course, and they might sometimes be hard to keep straight. As such, we will accumulate definitions of terms here each week.

Next to each term, in parentheses, is the week during which the term was first used, in case you'd like to revisit lecture material to learn more about it.

A

- **Address space** (1) - The range of possible (physical or virtual) addresses available to a process or system.
- **Architecture** (1) - Short for "computer architecture." Refers to the hardware computing platform that runs programs, which notably consists of a central processing unit (CPU), memory, and peripherals.

B

- **Buffer overflow** (1) - A vulnerability in a program in which an attacker can cause a pointer to access outside of the intended bounds of the buffer it points to.
- **Bug** (0) - A bug is a coding mistake in a program that causes it to behave incorrectly and/or insecurely.

C

- **Compiler** (1) - A compiler is a program that translates a program written in one language into an equivalent program in another language. Typically, a compiler takes a program in a high-level language, like C or FORTRAN, and compiles it to a program into the language of a particular processor, which can then run that program.

D

- **Dangling pointer** (1) - A dangling pointer is a pointer to memory that has been deallocated. A *dangling pointer dereference* is a bug that occurs when the program tries to access the memory pointed to by a dangling pointer.
- **Defect** (0) - A defect is a problem in a software system. The problem could be either in the design of the system, in which case we call it a *flaw*, or in the implementation, in which case we call it a *bug*

E

- **Environment variables** (1) - These are variables whose (string) values are tracked by command

shell. Environment variables are passed to programs that are spawned from the shell, along with the command-line arguments.
- **Exploit** (1) - An exploit is a series of steps by which an adversary interacts with a system to turn a vulnerability, or vulnerabilities, in the system into an attack whose outcome is to his advantage.

F

- **Flaw** (0) - A flaw is a (possibly security-relevant) defect in the design of a system.
- **Format string** (1) - A format string is a descriptor used by the C printf famliy of functions to describe how the provided arguments should be formatted.
- **Format specifier** (1) - A format specifier is one element of a format string that describes how particular arguments should be interpreted, when printed.
- **Frame pointer** (1) - It is typical on the x86 to for the compiler to dedicate the %ebp register as the frame pointer. It contains the address of the start of the area in a stack frame at which the local variables are stored.

G

H

- **Heap** (1) - An area of memory in a process responsible for storing dynamically allocated data, which the size and lifetime of that data is determined by information that is not known until run time. (Not to be confused with the heap tree-based data structure.)

I

- **Instruction pointer** (1) - A register that contains the address of the currently executing instruction. On the x86, the instruction pointer is stored in the %eip register.
- **Instruction set** (1) - The set of instruction types that can be executed by a particular computer architecture. There are different styles of instruction set; two common ones are CISC (Complex Instruction Set Computer) and RISC (Reduced Instruction Set Computing).

J

K

L

M

N

- **Nop sled** (1) - A nop sled is a sequence of "no-op" instructions. It is an element of an exploit that is

useful when the exact address of injected shellcode is not known. The nop sled precedes the shellcode, so landing anywhere in the sled will drive the program toward the shellcode.
- **NUL terminator** (1) - The character, a zero (or *NUL*) used to terminate a C string.
- **Vulnerability** (1) - A vulnerability is a bug in a program that can be exploited by an attacker to compromise security.

O

- **Operating system** (0) - A program that supports a computer's basic functions, which include starting, scheduling, and managing processes, and mediating access by those processes to input/output devices, like the stable storage (disk) and the network.

P

- **Physical address** (1) - An address to a location in physical memory.
- **Process** (1) - A process is an instance of a program in execution. Starting, running, and handling the termination of a process is the responsbility of the operating system.
- **Program** (1) - A program is a series of instructions for carrying out some task.
- **Program counter** (1) - Another name for instruction pointer

Q

R

- **Return address** (1) - The address to which to restore the instruction pointer when the current function returns. This address is stored on the stack when using the standard x86 calling convention.

S

- **Shell** (1) - The shell is the command prompt on Unix systems. In actuality it is an interpreter for a small "scripting" language whose aim is start, stop, compose, and otherwise work with processes.
- **Shellcode** (1) - Shellcode is code that the attacker would like to inject when exploiting a vulnerability, such as a buffer overflow.
- **Stack** (1) - Short for "call stack", which is an area of memory in a process responsible for storing information pertaining to the active functions. The stack stores a series of stack frames, one per active function call. These are pushed onto the stack when a function is called, and popped when the function returns.
- **Stack frame** (1) - A group of data on the stack that is associated with a particular function call. In the x86 standard calling conventino, the stack frame stores the arguments passed to the function, the function's local variables, and other metadata about the call, such as the frame pointer and return address.
- **Stack pointer** (1) - This is the address of the logical top of the stack. On the x86 architecture it is typically stored in the %esp register.
- **Static data area** (1) - The area of memory in a process that stores the program's global variables.

T

- **Text segment** (1) - The area of memory in a process that stores the program code.

U

V

- **Virtual address** (1) - An address used by a process to access memory; this address is translated by the hardware, with help from the operating system, to an actual physical memory address. Virtual memory is useful for allowing processes to always have the same address space no matter what physical memory they actual use when running. They also give the iillusion that the process has more memory than it actually does.
- **Vulnerability** (1) - A bug in a program that could potentially be exploited by an attacker to compromise security in some fashion.

W

X

Y

Z

Created Wed 15 Oct 2014 5:41 PM PDT
Last Modified Sat 18 Oct 2014 6:12 AM PDT