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

• **Abstraction** (5) - With reference to a program's semantics, an abstraction is an approximation. All valid behaviors of the program are represented by the abstraction, but invalid ones may be too.

• **Abuse case** (4) - A scenario that details a potential security failure if some defense is not put in place. Used during the design phase to motivate and justify security-related design decisions.

• **Access control** (4) - A kind of security policy that governs what principals have access to resources, and what kind of access (e.g., read vs. write) is allowed. There are different kinds of access control policies, e.g., role-based policies vs. discretionary policies.

• **Address space** (1) - The range of possible (physical or virtual) addresses available to a process or system.

• **Address Space Layout Randomization (ASLR)** (2) - This is a technique by which the starting address of memory areas is chosen randomly when a process is executed. Such randomization makes it harder to perform attacks which might otherwise rely on addresses not changing.

• **Alias** (5) - An alias is another name for the same thing. In programming, this comes up with pointers: a program may have two pointers to the same object, where one pointer is an alias for the other.

• **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.

• **Audit** (4) - The process of determining whether a security breach has occurred, and how. A system must be built to be auditable, e.g., by logging security-relevant data. Can also refer to the process of studying a system for vulnerabilities.

• **Authentication** (4) - The process by which the purported identity of an actor is confirmed. Examples include password checking and biometric scanning.

• **Authorization** (4) - The process by which a principal's proposed action is determined to be acceptable, or not, according to a system's security policy.

• **Availability** (0,4) - Refers to a classic security requirement about a system's operation; a system is *available* if it can be accessed by authorized principals when sought. A violation of availability is often called a *denial of service*.

B

• **Basic block** (2) - A sequence of statements in a program where control always begins at the start of the block, and flows from one statement to the next, until the last statement, which ends with a branch (to a different basic block), or halt instruction.

• **Big endian** (project 1) - Big-endian systems store the most significant byte of a word in the smallest address and the least significant byte is stored in the largest address.

• **Blacklist** (3) - A collection of possibly harmful elements that are checked, removed, or modified during input validation

• **Branch** (2) - A non-sequential control transfer. Branches can be conditional --- they occur only if a condition is met --- or unconditional (sometimes referred to as "jumps").

• **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. Sometimes this term more narrowly refers to the act of writing beyond the bounds of a buffer, but in this course we use it more generally to include reads as well.

• **Bug** (0) - A bug is a coding mistake in a program that causes it to behave incorrectly and/or insecurely.

C

• **Call graph** (2) - a graphical representation of the program where nodes are functions and edges indicate that source node's function could call the target node's function.

• **Capability** (3) - a communicable, unforgeable (or hard to forge) token of authority.

• **Chown** (4) - A system call by which one process can change the owner of a file to a different principal.

• **Chroot** (4) - A system call by which the current process's current working directory is made to look like the root directory of the file system, making inaccessible all the directories that are not descendants of the current directory.

• **Client** (3) - one role in a communicating pair (the other being the server), usually initiating the communication by requesting a service.

• **Complete mediation** (4) - A requirement of a secure computer system: all security-relevant actions must mediated so they can be authorized, or there can be a breach of security.

• **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.

• **Completeness** (5) - A *complete* analysis is one that, if some property *X* is true, then analysis says *X* is true. As applied to static analysis, we interpret completeness to mean: if the analysis says that there is a bug, then there really is one; i.e., there are no false positives.

• **Concolic execution** (5) - A kind of symbolic execution in which we run the program concretely, but keep track of symbolic values and the path condition "on the side". When the program terminates, this information can be used to generate a test that will take the program down a different path.

• **Confidentiality** (0,4) - Refers to a classic security requirement about a system's data; data enjoys confidentiality if it cannot be learned by an unauthorized principal. Also referred to as *secrecy* and *privacy* (where the latter is used when referring to an individual)

• **Constraint graph** (5) - An abstraction of a program that expresses directional constraints on its behavior. For example, when considering flow analysis, constraints indicate whether data can flow form one variable/position to another, and all constraints together can be visualized as a graph.

• **Context sensitivity** (5) - A feature that adds precision to a static analysis whereby a call to a function from one part of the program can be distinguished (i.e., analyzed separately from) another call.

• **Control flow graph (CFG)** (2) - a graphical representation of the program, where nodes in the graph are basic blocks, and edges indicate the possibility of control transferring from the source block to the target block.

• **Control flow integrity (CFI)** (2) - This is a property of a program execution. An execution satisfies control flow integrity if it conforms to a model of the program's control flow, determined in advance. Typically this model is a control-flow graph. CFI is often enforced as an in-line reference monitor.

• **Cookie** (3) - a piece of data associated with a particular web host and stored by the client's browser. Sent along with subsequent requests to that host. Used to implement sessions and personalization, and privacy-threatening tracking.

• **Cross-site Request Forgery (CSRF)** (3) - An attack by which a browser is induced by one site to submit a request to another site to perform an illicit action.

• **Cross-site Scripting (XSS)** (3) - An attack whose goal is to get a browser to execute a Javascript program originating at one site to execute with the privileges (according to the same-origin policy) of another site.

D

• **Dangling pointer** (1) - A dangling pointer is a pointer to memory that has been deallocated. Sometimes also called a stale pointer. A *dangling pointer dereference* is a bug that occurs when the program tries to access the memory pointed to by a dangling pointer.

• **Data execution prevention (DEP)** (2) - DEP is a service provided by the hardware and/or operating system that enforces memory can either be writeable or executable, not both. With DEP enabled, attacker-provided buffers cannot be executed directly, and thus cannot (usefully) contain code. DEP does not prevent return-to-libc or ROP attacks, which use code already in the program.

• **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*

• **Dereference** (1) - To dereference a pointer means to either read or write the memory that the pointer points to

• **Direct call** (2) - A call or jump to a target where the target is a constant. For example, the call printf("hello\n") directly calls the printf function.

• **Dynamic analysis** (5,6) - An analysis of a program's execution, often designed to find bugs. Oftentimes this analysis happens concurrently with the execution, and may terminate that execution if it observes illegal behavior.

E

• **Endiannness** (project 1) - Identifies the order that bytes are stored in memory to make up a multi-byte value. Can be either little endian or big endian.

• **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.

• **Escaping** (3) - The process of transforming a string, altering characters that might be interpreted as control characters in the current context to be inert. For example, a user string containing a < would be escaped in an HTML context to instead be <

• **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

• **File Transfer Protocol (FTP)** (4) - An application layer protocol for transferring files.

• **Flaw** (0) - A flaw is a (possibly security-relevant) defect in the design of a system.

• **Flow analysis** (5) - A kind of static analysis that determines whether values from one position in the program could reach, or *flow* to, another position in the program.

• **Flow sensitivity** (5) - A feature that adds precision to a static analysis whereby the order of the statements in the program is taken into account; a flow-insensitive analysis supposes that program statements could occur in any order.

• **Fork** (4) - The act of a process creating a new process that is a duplicate of itself; the new process is called the *child*, and the forking process is called the *parent*. A typical action of the child process is to immediately use the exec system call to convert itself to run a different program.

• **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.

• **Fuzzing** (or **fuzz testing**) (6) - A kind of random testing whose aim to induce a system to fail, where the expectation is that some of these bugs will be security relevant

G

• **Gadget** (2) - the name of a code block used in return oriented programming. Such blocks end in return instructions, and return-oriented programs string together gadgets to provide complete functionality.

• **GET** (3) - a kind of HTTP request, used to retrieve a resource from a remote site.

H

• **Halting problem** (5) - The halting problem is the problem of determining, for an arbitrary program and input, whether the program will finish running or continue to run forever. This problem is known to be *undecidable.*

• **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.)

• **Hidden Form Field** (3) - A field in an HTML form that is not displayed; it is often used to track hidden state.

• **Hyperlink** (3) - An element of a hypertext document that references, or *links*, a remote document.

• **Hypertext** (3) - Indicates text that contains hyperlinks, which reference other, related documents.

• **Hypertext Transfer Protocol (HTTP)** (3) - The protocol that underlies the World Wide Web (WWW). The protocol consists of *requests* from clients (either GET, or POST, usually), and *responses* from servers.

• **Hypertext Markup Language (HTML)** - The core language of documents served by the WWW. Browsers *render* these documents and permit interacting with them.

I

• **Immutability** (2) - a memory region is immutable if it cannot be changed. For example, the text segment is often immutable.

• **Indirect call** (2) - A call or jump to a target where the target can vary at run-time. For example, calls to function pointers (in C), and virtual method calls (in C++ and Java) are indirect.

• **In-line reference monitor (IRM)** (2) - An IRM checks that a program's behavior corresponds to a security policy, and is implemented as part of the program, i.e., is "in-lined" within it.

• **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).

• **Integrity** (0,4) - Refers to a classic security requirement about a system's data; data enjoys integrity if it cannot be modified or influenced by an unauthorized principal

• **Internet Control Message Protocol (ICMP)** (6) - A messaging protocol used for network diagnostics

J

• **Javascript** (3) - A programming language used to write scripts (small programs) embedded in web pages, which run at the browser. The harbinger of *Web 2.0*.

K

• **Keylogging** (6) - The process of logging keystrokes on a machine, usually carried out surreptitiously by malware, with the goal of stealing information like passwords

L

• **Lattice** (5) - A lattice is a partially ordered set such that each pair of elements in the set has a unique least upper bounds and a unique greatest lower bound. Lattices are often a foundation of static flow analysis.

• **Least privilege** (4) - A classic security principal; this principle dictates that a subsystem should be given only the privilege it needs, *and no more*, to carry out its responsibilities. Limiting privilege limits damage if the subsystem is compromised.

• **Little Endian** (project 1) - Little-endian systems store the least significant byte of a word in the smallest address, and the most significant byte in the largest address.

• **LLVM** (2) - The LLVM Project is a collection of modular and reusable compiler and toolchain technologies.

M

• **Memory safety** (2) - A program execution is memory safe it is both spatially safe and temporally safe (these terms are elsewhere in this glossary).

• **Multi-factor authentication (MFA)** (4) - Authentication protocols typically test multiple *factors*, e.g., what a purported principal *knows*, *is*, or *has*. Multi-factor authentication involves testing multiple factors, not just one. For example, it may ask for a password (what you know) and a text message (what you have -- a phone).

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.

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

• **Parse tree** (3) - A representation of how a string is parsed according to a particular grammar, which identifies the structural elements of that string.

• **Path sensitivity** (5) - A feature that adds precision to a static analysis whereby the order of the statements in the program is taken into account, as is the set of conditionals that have been followed. As such, it is strictly more precise than a flow-sensitive analysis.

• **Payload** (6) - The data injected into a vulnerable program to perform an exploit

• **Penetration testing** (6) - A means to assess the security of a complete system (or network of systems) by actively trying to find exploitable vulnerabilities.

• **Phishing** (4) - An attack against a user, whereby the attacker masquerades as a trusted authority in an attempt to convince the user to reveal secret information or otherwise compromise something of value.

• **Physical address** (1) - An address to a location in physical memory.

• **POST** (3) - A HTTP request type, used to submit information to a remote server, e.g., when filling out a form as part of a request.

• **Precision** (5) - In reference to static analysis, precision refers to faithfulness of the analysis's abstraction to the true semantics of a program: the more precise, the less abstract (and more faithful) is the analysis.

• **Principal** (4) - The subject, or actor, in a security policy, which dictates what actions the subject can perform. A principal can be a person, a computer program, a role, etc.

• **Process** (1) - A process is an instance of a program in execution. Starting, running, and handling the termination of a process is the responsibility of the operating system.

• **Program** (1) - A program is a series of instructions for carrying out some task.

• **Program counter** (1) - Another name for the instruction pointer

Q

R

• **Random testing** (6) - A kind of testing that uses randomly generated inputs. Such tests may be fully automatic (as with fuzz testing), or may include pre-conditions on the input and assertions on the output.

• **Replay attack** (4) - A kind of attack that involves replaying a recorded message in an attempt to effect, again, an action carried out previously.

• **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.

• **Return to libc** (2) - This is a style of exploit that aims to get a program to run code already in the program by returning to it (e.g., through an overwritten return address). A typical retun-to-libc attack is to get the program to run the system() library call (e.g., to produce a remote shell controlled by the attacker).

• **Return oriented programming (ROP)** (2) - This is a generalization of return-to-libc attacks, in that the attack may return to arbitrary positions, not just the start of functions, that can be sequenced together by return calls to induced attacker-preferred behavior.

S

• **Same Origin Policy (SOP)** (3) - A key security policy protecting resources (like cookies and web pages) originating from one web site from running Javascript code that originates from another site.

• **Sanitization** (3) - The process of removing or replacing potentially harmful elements from untrusted input.

• **Satisfiability (SAT) solver** (5) - A program that solves satisfiability constraints, which are simply boolean constraints *C*, involving variables (*X*), constants (true and false), conjunctions (*C1* and *C2*), disjunctions (*C1* or *C2*), and negations (not *C*).

• **SAT modulo theories (SMT) solver** (5) - A program that solves satisfiability constraints coupled with constraints drawn from different *theories*, such as the theory of linear arithmetic, arrays, and uninterpreted functions. SMT solvers for the core of the automated reasoning portion of many automated analyses.

• **Scanner** (6) - A tool that scans a network aiming to discover hosts and services they might be running. An example scanner is NMAP.

• **Scalability** (5) - In reference to a static analysis, scalability refers to the analysis's ability to work effectively on large programs, and/or programs with sophisticated features.

• **SecComp** (4) - A Linux system call by which a process's ability to carry out certain system calls is reduced. Implements the principle of least privilege.

• **Semantics** (5) - The semantics of a program is its meaning; i.e., what actions it may perform or outputs it may prod\uce in response to certain inputs.

• **Server** (3) - one role in a communicating pair (the other being the client). The server receives requests and provides service of some sort in response.

• **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.

• **Sink** (5) - In reference to *flow analysis*, a sink is a possible destination of a data flow.

• **Soundness** (5) - A *sound* analysis is one that, if the analysis says that *X* is true, then *X* is true. As applied to static analysis, we interpret soundness to mean: if the analysis claims a program is error free, then it really is; i.e., there are no false negatives.

• **Source** (5) - In reference to *flow analysis*, a sink is a possible starting point of a data flow.

• **Spatial safety** (2) - A program execution is spatially safe if it does not use a pointer to access memory not "owned" by that pointer. Roughly speaking, a pointer owns the memory it points to if the pointer was constructed by legal means, and subsequent manipulations (e.g., pointer arithmetic) have not caused it to point outside those bounds. Memory safety implies spatial safety. See this blog post for details.

• **Spear phishing** (4) - A kind of phishing that is targeted at a particular user or group of users, using targeted information to appear more convincing.

• **SQL Injection** (3) - An attack which can effect the execution of SQL code where instead data was expected

• **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 canary** (2) - this is a special value written to a stack frame that is used to detect evidence of a stack smashing attack. When a function goes to return, it checks that the stack canary is still intact; if not, an attack may have overflowed over it.

• **Stack frame** (1) - A group of data on the stack that is associated with a particular function call. In the x86 standard calling convention, 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 anaysis** (5) - An algorithm that analyzes a computer program *without running it* in order to determine some property (or many properties) about its executions.

• **Static data area** (1) - The area of memory in a process that stores the program's global variables.

• **Static single assignment (SSA) form** (5) - Programs in SSA form may not assign to a variable more than once (a "static single assignment"). Programs in such form make them easier to analysis, particularly when one is interested in flow sensitivity.

• **Structured Query Language (SQL)** (3) - A special-purpose programming language designed for managing (querying, updating, inserting, and deleting) data held in a relational database management system (RDBMS).

• **Symbolic execution** (5) - A technique for automated program analysis in which certain variables are treated as symbolic, rather than as having concrete values. Symbolic exectuion falls somewhere between testing (in a sense you are "running" the programs) and static analysis (each run represents many concrete runs).

• **SYN** and **SYN/ACK** (6) - Two particular packets in the TCP connection setup protocol

T

• **Temporal safety** (2) - A program execution is temporally safe it does not use undefined memory. Undefined memory has either never been allocated, or has been allocated but then freed. Memory safety implies temporal safety. See this blog post for details.

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

• **Transmission Control Protocol (TCP)** (3) - TCP is one of the core protocols of the Internet protocol suite (IP), and provides reliable, streaming data delivery (overtop an unreliable medium).

• **Trusted computing base (TCB)** (4) - The portion of a system that must be trusted if the system's operation is to be secure. Ideally, the trusted computing base is small and simple, so that trust placed in it is warranted.

• **Turing completeness** (2) - A Turing complete system or language is one that can simulate a single-taped Turing machine. Such a machine is known to be very computationally expressive, so the Turing completeness of a system shows that it is similarly very expressive.

• **Type inference** (5) - A kind of static analysis whose goal is to assign a type to a program fragment (like a variable or function). Types may be standard language types (like int or float) or enhanced types expressing other program properties (like tainted int or untainted float).

• **Type safety** (2) - Type safety is a consistency property of a programming language: it states that programs accepted by the language's type system are also well-defined. What this means depends on the type system. See this blog post for more detail.

U

• **Undecidability** (5) - An undecidable problem is a decision problem for which it is known to be impossible to construct a single algorithm that always leads to a correct yes-or-no answer. The *halting problem* is an example of an undecidable problem.

• **Universal Resource Locator (URL)** (3) - The name of a document on the WWW. Consists of three parts: the *protocol* by which the document can be retrieved; the *host* at which the document is located; and the *path* at that host that names the document.

• **User Datagram Protocol (UDP)** (6) - A protocol that supports unreliable packet delivery over IP

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 illusion 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

• **Web proxy** (6) - A program that intercepts traffic sent between a web browser and web server, providing the capability to record it, modify it and/or forward it, etc. Examples include the Burp suite, and OWASP ZAP.

• **Web spider** (6) - A program that explores the structure of a web site (or set of sites), following links from one page to the next, to create a model of the overall network of web pages. Burp Suite and OWASP ZAP both implement Spider functionality.

• **Whitelist** (3) - A collection of known-good elements against which untrusted input is validated; non-whitelisted elements are rejected.

• **Worldwide Web (WWW)** (3) - Refers to the collection of servers that communicate HTML documents via HTTP, over the Internet.

• **W xor X** (2) - Stands for "Writable or executable, but not both". Another name for Data Execution Prevention (DEP).

X

• **XSS** (3) - Stands for *cross-site scripting*.

Y

Z