

# INTRODUCTION TO SECURE CODING

# What is secure coding?

- ▣ Secure coding is the practice of writing programs that are resistant to attack by malicious.
- ▣ Secure coding helps protect a user's data from **theft** or **corruption**.
- ▣ An insecure program can provide access for an attacker to take control of a server or a user's computer.
- ▣ **Hacker** refers to an expert programmer — one who enjoys learning about the **intricacies** of code or an operating system.

- ❑ In general, hackers are not malicious. When most hackers find security vulnerabilities in code, they inform the company or organization that's responsible for the code so that they can fix the problem.
- ❑ The malicious individuals who break into programs and systems in order to do damage or to steal something are referred to as **crackers**, **attackers**, or **black hats**.

Black hat is used to describe a hacker (or, if you prefer, cracker) who breaks into a computer system or network with malicious intent. Unlike a white hat hacker, the black hat hacker takes advantage of the break-in, perhaps destroying files or stealing data for some future purpose.

- ▣ White hat describes a hacker who identifies a security weakness in a computer system or network but, instead of taking malicious advantage of it, exposes the weakness in a way that will allow the system's owners to fix the breach before it can be taken advantage by others (such as black hat hackers.)
- ▣ Most attackers are not highly skilled, but take advantage of published exploit code and known techniques to do their damage

# Types of Security Vulnerabilities

- ▣ Most software security vulnerabilities fall into one of a small set of categories:
- ▣ Buffer overflows
- ▣ Race conditions
- ▣ Access-control problems

# Buffer overflows

- ❑ Buffer overflows have been the most common form of security vulnerability.
- ❑ A buffer overflow occurs when more data are written to a buffer than it can hold. The excess data is written to the adjacent memory, overwriting the contents of that location and causing unpredictable results in a program.
- ❑ Buffer overflow is an increasingly common type of security attack on **data integrity** .

- ▣ Attackers can exploit a buffer overflow bug by injecting code that is specifically tailored to cause buffer overflow. The overflow data might contain executable code that allows the attackers to run bigger and more sophisticated programs or grant themselves access to the system.

# Race Conditions

- ▣ The general principle of race conditions is the following : a process wants to access a system resource.
- ▣ It checks the resource is not already used by another process, then it takes over and uses it as it wants.
- ▣ The problem appears when another process tries to benefit from the lapse of time between the check and the true access to take over the same resource



- Because there is a time gap between the check and the use (even though it might be a fraction of a second), an attacker can sometimes use that gap to mount an attack.
- Thus, this is referred to as a time-of-check–time-of-use problem.
- **Ex.** In computer memory or storage, a race condition may occur if commands to read and write a large amount of data are received at almost the same instant, and the machine attempts to overwrite some or all of the old data while that old data is still being read.

- ❑ When an application writes temporary files to publicly accessible directories.
- ❑ If the file already exists before you write to it, you could be overwriting data needed by another program, or you could be using a file prepared by an attacker, in which case it might be a hard link or symbolic link, redirecting your output to a file needed by the system or to a file controlled by the attacker

Victim	Attacker
<pre>if (access("file", W_OK) != 0) {     exit(1); }  fd = open("file", O_WRONLY); // Actually writing over /etc/passwd write(fd, buffer, sizeof(buffer));</pre>	<pre>// // // After the access check symlink("/etc/passwd", "file"); // Before the open, "file" points to the password database // //</pre>

# Access Control

- ▣ Access control, sometimes called **authorization**, is the means by which a web application grants access to specified content and functions to some users and not others.
- ▣ Access control governs what **"authorized"** users are allowed to do.
- ▣ Authentication must precede authorization. Authentication verifies the identity of a user. Authorization is then used to determine what that user is allowed to access.
- ▣ To ensure proper access control, a web application must ensure both that it has proper authorization checks, and that it is using reliable and secure authentication that can distinguish privileged users from others.

- ▣ Of particular interest to attackers is the gaining **of root privileges**, which refers to having the unrestricted permission to perform any operation on the system.
- ▣ An application running with root privileges can access everything and change anything.
- ▣ Many security vulnerabilities involve **programming errors** that allow an attacker to obtain root privileges.
- ▣ Some such exploits involve taking advantage of **buffer overflows or race conditions**, which in some special circumstances allow an attacker to escalate their privileges.
- ▣ Others involve having access to system files that should be restricted or finding a weakness in a program — such as an **application installer** — that is already running with root privileges.
- ▣ For this reason, it's important to always run programs with as few privileges as possible.