# 10 SCARIEST SOFTWARE SECURITY VULNERABILITIES

Dreadful defects in software are lurking everywhere you look. In fact, more than 85% of all applications have at least one vulnerability in them. These are the 10 most common — and scariest — vulnerabilities plaguing applications today.

## 10 ENCAPSULATION

**20%** OF APPLICATIONS ARE VULNERABLE

**WHAT IT IS**
Encapsulation vulnerabilities don't sufficiently encapsulate critical data or functionality. Examples include trust boundary violations, protection mechanism failures, and deserialization of untrusted data.

**CONSEQUENCE**
Ghoulish code can cross over between components and data can escape.

**REMEDY**
Wrap private data in classes to keep implementation details hidden from the user. Be sure to correctly set security headers — and don't trust serialized inputs from outside the application.

## 9 SQL INJECTION

**27%** OF APPLICATIONS ARE VULNERABLE

**WHAT IT IS**
SQL injection allows an attacker to gain unauthorized access to a back-end database by using maliciously crafted input.

**CONSEQUENCE**
An attacker can access, alter, or delete data in the back-end database without authorization and do other undesirable things.

**REMEDY**
Use parameterized queries so the database treats them as data instead of as part of a SQL command.

## 8 CREDENTIALS MANAGEMENT

**43%** OF APPLICATIONS ARE VULNERABLE

**WHAT IT IS**
Flaws in the handling of user credentials could permit attackers to bypass access controls. Some common errors include hard-coded passwords and plaintext passwords in config files.

**CONSEQUENCE**
Giving you a real scare, attackers can assume privileges of users or administrators.

**REMEDY**
Use custom or off-the-shelf authentication and session management mechanisms to protect passwords and session IDs from abuse.

## 7 INSUFFICIENT INPUT VALIDATION

**47%** OF APPLICATIONS ARE VULNERABLE

**WHAT IT IS**
Insufficient input validation includes a number of flaws that permit malformed input that can cause security issues, including open redirect and unsafe reflection.

**CONSEQUENCE**
Attackers can input creepy code to read and steal data, hijack sessions, and execute malicious code.

**REMEDY**
Treat data entered by users as untrusted. Use whitelists to define valid input data.

## 6 DIRECTORY TRAVERSAL

**48%** OF APPLICATIONS ARE VULNERABLE

**WHAT IT IS**
Directory traversal flaws open up the possibility of attacks that allow cybercriminals to gain unauthorized access to restricted directories and files.

**CONSEQUENCE**
Attackers can access files and directories by sending modified URLs to the web server.

**REMEDY**
Use filters to blacklist commands and escape codes commonly used by attackers.

## 5 CROSS-SITE SCRIPTING

**49%** OF APPLICATIONS ARE VULNERABLE

**WHAT IT IS**
Cross-site scripting (XSS) vulnerabilities give attackers the capability to inject client-side scripts into the application, potentially bypassing security controls in the process.

**CONSEQUENCE**
Attackers can view and steal sensitive information, modify files and content on the affected website, and hijack the user's browsing session or computer.

**REMEDY**
Input sanitization and encoding output are yourbest friends against injection attacks.

## 4 CRLF INJECTION

**60%** OF APPLICATIONS ARE VULNERABLE

**WHAT IT IS**
CRLF injection vulnerabilities enable what is known as Carriage Return Line Feed (CRLF) injection attacks. Examples include improper output neutralization for logs and improper neutralization of CRLF in HTTP headers.

**CONSEQUENCE**
By introducing an unexpected CRLF injection, an attacker can modify application data, deface websites, hijack sessions or browsers, and exploit other vulnerabilities.

**REMEDY**
Never trust user input. Always properly encode output in HTTP headers or log entries that would otherwise be visible to users or administrators.

## 3 CODE QUALITY

**63%** OF APPLICATIONS ARE VULNERABLE

**WHAT IT IS**
Some examples of code quality defects include improper resource shutdown or release, leftover debug code, and using the wrong operator when comparing strings.

**CONSEQUENCE**
Leftover debug code may contain unanticipated functionality that an attacker could use to disclose sensitive data (such as test methods). An attacker could use improper resource shutdown or release to mount a Denial of Service attack by causing the application to use up host resources,like memory.

**REMEDY**
An informed development team is key to secure coding. Development teams with eLearning on secure coding see fix rates improve by 19%.

## 2 CRYPTOGRAPHIC ISSUES

**64%** OF APPLICATIONS ARE VULNERABLE

**WHAT IT IS**
Cryptographic flaws include using broken crypto algorithms, improperly validating certificates, storing sensitive information in cleartext, and employing inadequate encryption strength.

**CONSEQUENCE**
Encryption hides important information like passwords, payment info, personally identifying data, etc. If improperly stored data is leaked, it can turn into your worst nightmare.

**REMEDY**
Don't implement your own encryption — enlist experts in the field to avoid a scream-worthy breach.

## 1 INFORMATION LEAKAGE

**67%** OF APPLICATIONS ARE VULNERABLE

**WHAT IT IS**
Information leakage flaws can reveal sensitive data about the application, environment, or user that could be leveraged by an attacker to hone future attacks against the application.

**CONSEQUENCE**
An attacker can use leaked information about the user or the application to hone successful attacks against the application.

**REMEDY**
Vulnerability scanning tools will cause error messages to be generated and can search for APIs that leak information.

## WONDERING IF THE STATE OF YOUR SOFTWARE IS SECURE OR SCARY?

Get the State of Software Security Report

**VERACODE.COM/SOSS**

SOURCE: Veracode State of Software Security Vol. 9. Vulnerability percentages based on prevalence in initial assessments of applications Veracode scanned between April 1, 2017 to March 31, 2018.

ca technologies | VERACODE

LEARN MORE AT VERACODE.COM, ON THE VERACODE BLOG AND ON TWITTER.