10 SCARIEST VULNERABILITIES

Dreadful defects in applications are lurking everywhere you look. In fact, more than **60% of applications fail security policy the first time they're checked.** These are the most common — and scariest — application vulnerabilities today.

10 ENCAPSULATION ERRORS

PREVALENCE 25%

CAUSE
Applications are vulnerable when they don't properly separate or differentiate critical data or functionality, or fail to protect against cross-domain attacks.

CONSEQUENCE
Creepy code can cross over between components and data can escape.

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

9 SQL INJECTION

PREVALENCE 32%

CAUSE
Applications that fail to properly sanitize user input can allow an attacker to use maliciously-crafted SQL queries to trick the application into returning a treat from the server.

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

CURE
Use parameterized queries so the database treats them as data rather than part of a SQL command.

8 CREDENTIALS MIS-MANAGEMENT

PREVALENCE 41%

CAUSE
Hard-coded passwords and plaintext passwords in config files and elsewhere — like leaving the key under the doormat where goblins know to look.

CONSEQUENCE
Like a zombie virus taking over a victim, attackers can assume privileges of users or administrators.

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

7 INSUFFICIENT INPUT VALIDATION

PREVALENCE 44%

CAUSE
Bad things happen when improperly sanitized or authenticated inputs are sent to your server.

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

CURE
Always treat data entered by users as untrusted. Use whitelists to define valid input data.

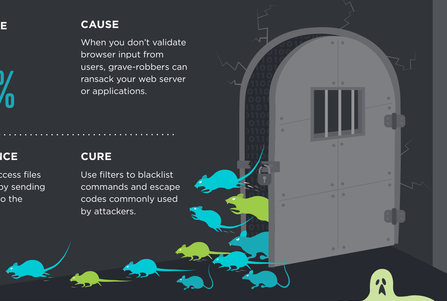
6 DIRECTORY TRAVERSAL

PREVALENCE 49%

CAUSE
When you don't validate browser input from users, grave-robbers can ransack your web server or applications.

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

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



5 CROSS-SITE SCRIPTING

PREVALENCE 50%

CAUSE
When an application does not validate or encode user input, ghoulish attackers can inject malicious scripts into websites that a user's browser executes as trusted code.

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.

CURE
Input sanitization and encoding output are your best friends against injection attacks.

4 CRLF INJECTION

PREVALENCE 53%

CAUSE
CRLF ("Carriage Return" and "Line Feed") characters are embedded in HTTP headers or software log entries to split text streams into discrete elements. CRLF injection vulnerabilities happen when you don't properly sanitize or neutralize data input.

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

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

3 BAD CODE QUALITY

PREVALENCE 62%

CAUSE
Insecure coding practices lead to errors that eventually impact the security of the application, like a ghost in the machine.

CONSEQUENCE
Gruesome code examples include leftover debug code and improper resource shutdown or release.

CURE
An informed development team is key to secure coding. Implementing an eLearning program can improve fix rates by as much as 6x.

2 CRYPTOGRAPHIC ISSUES

PREVALENCE 65%

CAUSE
Cryptographic issues include insecure crypto algorithms and improper key management.

CONSEQUENCE
Encryption hides important stuff: passwords, payment info, personally identifying data and a whole lot more. If improperly stored data gets out, heads may roll.

CURE
Don't play Frankenstein and try implementing your own encryption — you'll end up with a gruesome monster.

1 INFORMATION LEAKAGE

PREVALENCE 72%

CAUSE
Warning! Sensitive information can leak out through error messages and user functions that return different results based on different inputs.

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

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

HOW VULNERABLE ARE YOUR APPLICATIONS?

You can prevent all of these scary vulnerabilities and more by routinely assessing your applications through automated static and dynamic testing.

State of Software Security **GRAB THE REPORT AT VERACODE.COM/SOSS**

VERACODE
SECURING THE SOFTWARE THAT POWERS YOUR WORLD.

SOURCE: *Veracode State of Software Security 2016.* Prevalence based on percentage of applications with vulnerability on initial scan.

Learn more at VERACODE.COM