snyk.io

What is a Security Vulnerability? | Types & Remediation | Snyk

9-12 minutes

The average cost of a data breach in 2020 was $3.86 million and global cybercrime costs in 2021 are expected to reach $6 trillion. While 82% of known vulnerabilities are in application code, with 90% of web applications vulnerable to hacking and 68% of those vulnerable to the breach of sensitive data.

This article provides insights and tools to help keep your company on the winning side of cybercrime statistics. We discuss types of security vulnerabilities, vulnerability versus exploit, website security vulnerabilities, and security and vulnerability management.

Vulnerability vs. Exploit vs. Threat

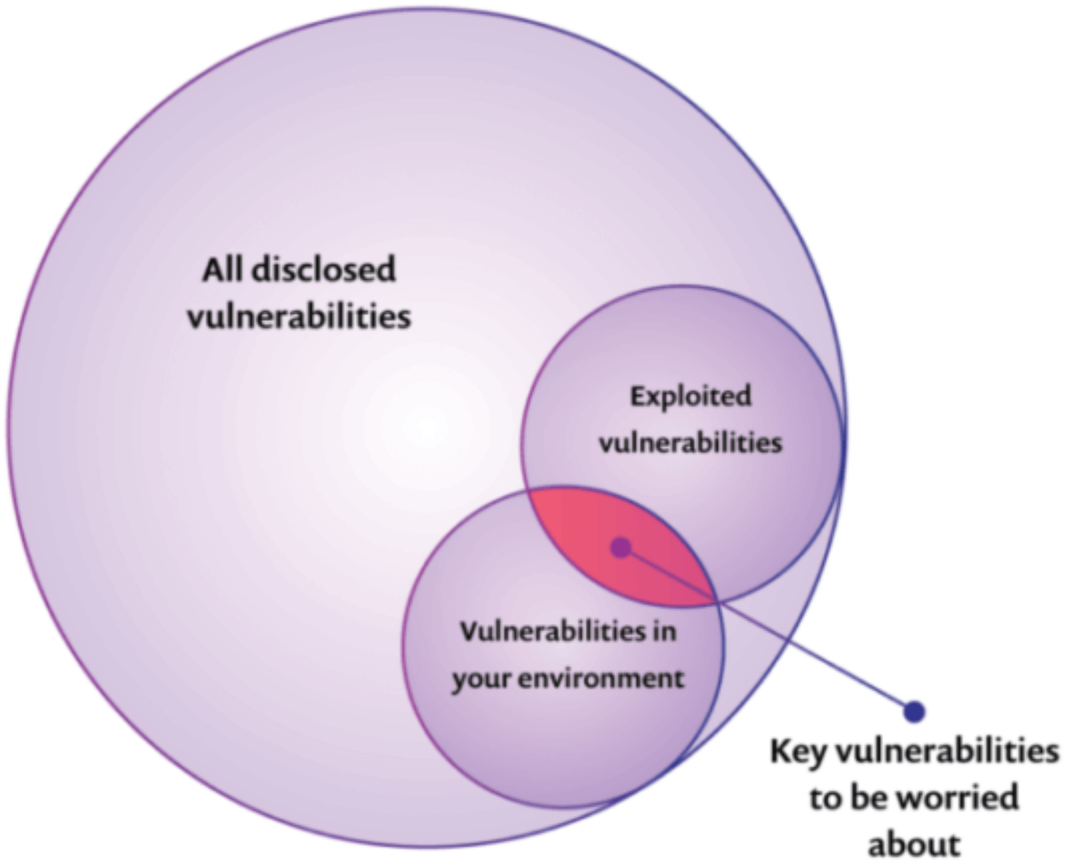
In order to effectively manage cybersecurity risk, it is important to understand the difference between a vulnerability, an exploit and a threat.

A **security vulnerability** is a software code flaw or a system misconfiguration such as Log4Shell through which attackers can directly gain unauthorized access to a system or network. Once inside, the attacker can leverage authorizations and privileges to compromise systems and assets.

An **exploit** is the *means* through which hackers use a vulnerability to mount an attack. An exploit is typically a piece of specially crafted software or a sequence of commands. For example, vulnerabilities in Microsoft IIS (Internet Information Services) and MS-SQL server have been exploited over the years by network worms such as CodeRed, Spida, and Slammer.

There are even exploit kits out there (e.g., Rig, Magnitude, and Fallout) that can be embedded in compromised web pages where they continuously scan for vulnerabilities. As soon as a weakness is detected, the kit immediately attempts to deploy an exploit, such as injecting malware into the host system.

A **threat** is the actual or hypothetical *event* in which one or more exploits use a vulnerability to mount an attack. For example the CodeRed exploit on the Microsoft IIS vulnerability has been actively used to infect more than 300,000 targets. These threats have caused huge financial losses around the globe.


All disclosed vulnerabilities
Exploited vulnerabilities
Vulnerabilities in your environment
Key vulnerabilities to be worried about

Only a small percentage of known vulnerabilities will be **exploited**, or in other words, used to hack into a system. Vulnerabilities that pose the highest risk are those that have a higher chance of being exploited and therefore should be prioritized and attended to first, as seen in the diagram:

Find and fix security vulnerabilities

Automatically find, prioritize and fix vulnerabilities in the open source dependencies used to build your cloud native applications

Types of Security Vulnerabilities

Security vulnerabilities can be found at all layers, including infrastructure, network, and application. In this section, we focus on weaknesses in application security and website security.

There are two important lists that track the weaknesses that make web applications and websites vulnerable to cybersecurity risk. The first is maintained by the open-community, global Open Web Application Security Project (OWASP). Of the 60 or so application security weaknesses described in OWASP, the OWASP Top 10 Vulnerabilities features those that are most commonly exploited as vulnerabilities. The current list is from 2017 and it is in the process of being updated.

The second list is CWE, or Common Weakness Enumeration, which is a "community-developed list of common software and hardware weakness types that have security ramifications." CWE, like the well-known Common Vulnerabilities and Exposures (CVE) standardized dictionary, is run by the MITRE Corporation, a not-for-profit company that operates federal government funded R&D centers. The CWE-25 is an annually updated listing of the 25 most dangerous software weaknesses.

MITRE CWE-25 enumerates 3 major types of application and website security weaknesses:

1. Porous defenses
2. Risky resource management
3. Insecure interaction between components

1. Porous Defenses

This first weakness type encompasses flaws that could allow users to bypass or spoof authentication and authorization processes.

Authentication verifies the identity of someone trying to access a system while authorization is the set of access and usage permissions assigned to the user.

Porous defense flaw examples include:

- Weak password encoding
- Insufficiently protected credentials
- Missing or single-factor authentication
- Insecurely inherited permissions
- Sessions that don’t expire in a timely manner

All of these porous defense vulnerability types can seriously undermine the organization’s security posture if unauthorized entities successfully access and abuse sensitive resources.

Exploits that leverage porous defense vulnerabilities may include:

- Credential stuffing attacks
- Hijacking of session IDs
- Stealing login credentials

- Man-in-the-middle (MITM) attacks (essentially electronic eavesdropping)

2. Risky Resource Management

Many vulnerabilities fall within the category of risky management of resources such as memory, functions, and open-source frameworks. During the design and development stages of web applications and websites, it is critical that all third-party components to be included in the architecture—such as libraries and functions—are scanned for vulnerabilities.

Flaws in this category include:

- **Out-of-bounds write or read (buffer overflow):** The application can be tricked into writing or reading data past the end or before the beginning of the intended memory buffer.
- **Path traversal:** Allows attackers to craft pathnames that let them access files outside of restricted directories.

Buffer overflow attacks are a classic example of how risky resource management flaws expose web applications and websites to cybersecurity risk. These attacks exploit inadequate memory buffer controls to change execution paths and thus gain control over the application, damage files, or exfiltrate sensitive information.

3. Insecure Interaction Between Components

Today's highly distributed application architectures send and receive data across a wide range of services, threads, and processes. At runtime, web applications and websites must implement a zero-trust approach, whereby every input is suspect until actively verified as coming from a trusted source and addressing its intended purpose.

Weaknesses that expose a web application or website to insecure and risky interactions include:

- **Cross-site scripting or XSS:** Improper neutralization of user-controllable input during web page generation. The chain of events that leads to a malicious script being served in a generated web page starts with untrusted data entering a web application, usually via a web request.

- **Cross-site request forgery (CSRF):** Improper verification of whether a seemingly legitimate and authentic request was intentionally sent. These attacks are often mounted via social engineering vectors such as bogus emails that trick a user to click a link, which then sends a forged request to a site or server where the user has already been authenticated.

In general, web applications or websites that do not implement zero-trust security controls are vulnerable to backdoor attacks, scripting attacks, worms, trojan horses and other exploits that deploy malicious code to wreak havoc on infrastructure, data, and systems.

What is the most common vulnerability?

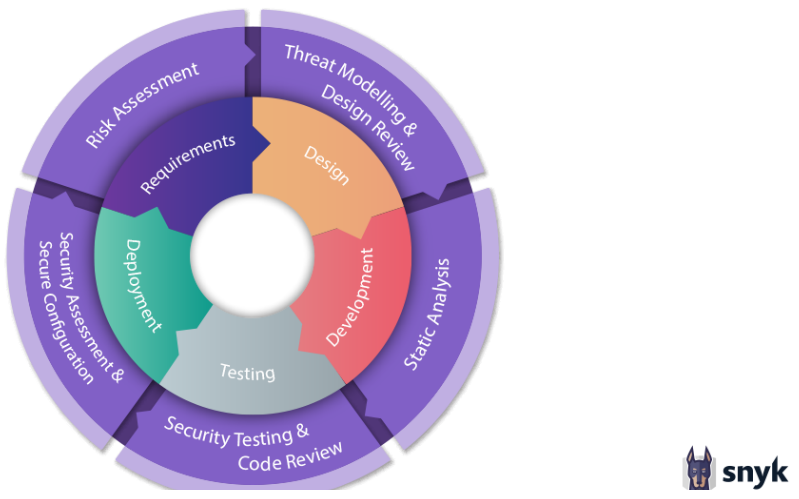
Both OWASP-10 and CWE-25 denote injection flaws as the most common vulnerability. While code injection can take many forms (SQL, NoSQL, OS, LDAP), it always involves hostile data sent to an interpreter via a command or query. The interpreter is tricked into carrying out unauthorized activities such as executing commands or accessing data.

Find and Fix Security Vulnerabilities

Security vulnerabilities are found and fixed through formal vulnerability management programs. Vulnerability management comprises cross-team best practices and procedures for identifying, prioritizing, and remediating vulnerabilities in a timely manner and at scale.

Security vulnerability assessment is an important part of the vulnerability management program. It focuses on gaining insight into the efficacy of the organization’s vulnerability management processes and procedures through penetration testing and other automated vulnerability assessment tests.

Secure Software Development Life Cycle (SSDLC)


Risk Assessment
Threat Modelling & Design Review
Static Analysis
Security Testing & Code Review
Security Assessment & Secure Configuration
Requirements
Design
Development
Testing
Deployment
snyk

Security vulnerability management and assessment should be an integral part of a secure software development life cycle (Secure SDLC) that tests code, libraries, container images, and other components for weaknesses and vulnerabilities throughout all phases of the product life cycle.

Tools for Security Vulnerability Remediation throughout the SDLC

The automated tools that help DevSecOps teams implement Secure SDLC include:

- **Software composition analysis (SCA) tools:** These discover and monitor all components, alerting to and fixing issues.
- **Static application security tools (SAST):** These carry out white-box vulnerability assessment on uncompiled code.
- Dynamic application security tools (DAST)**:** These carry out black-box vulnerability assessment on executable code.
- **Open-source vulnerability scanners:** These

identify vulnerabilities in code libraries for faster response and remediation, while ensuring compliance with open source license requirements.

Find and fix security vulnerabilities

Automatically find, prioritize and fix vulnerabilities in the open source dependencies used to build your cloud native applications

Many vulnerability management tools rely on a database of known vulnerabilities in order to carry out scanning and assessment. The Snyk Intel Vulnerability Database, for example, draws on and enriches public and proprietary vulnerability and threat intelligence sources to provide the industry’s most comprehensive and efficient open-source vulnerability scanner.