

A06:2021 – Vulnerable and Outdated Components



Factors

CWEs Mapped	Max Incidence Rate	Avg Incidence Rate	Max Coverage	Avg Coverage	Avg Weighted Exploit	Avg Weighted Impact	Total Occurrences	Total CVEs
3	27.96%	8.77%	51.78%	22.47%	5.00	5.00	30,457	0

Overview

It was #2 from the Top 10 community survey but also had enough data to make the Top 10 via data. Vulnerable Components are a known issue that we struggle to test and assess risk and is the only category to not have any Common Vulnerability and Exposures (CVEs) mapped to the included CWEs, so a default exploits/impact weight of 5.0 is used. Notable CWEs included are *CWE-1104: Use of Unmaintained Third-Party Components* and the two CWEs from Top 10 2013 and 2017.

Description

You are likely vulnerable:

- If you do not know the versions of all components you use (both client-side and server-side). This includes components you directly use as well as nested dependencies.
- If the software is vulnerable, unsupported, or out of date. This includes the OS, web/application server, database management system (DBMS), applications, APIs and all components, runtime environments, and libraries.
- If you do not scan for vulnerabilities regularly and subscribe to security bulletins related to the components you use.
- If you do not fix or upgrade the underlying platform, frameworks, and dependencies in a risk-based, timely fashion. This commonly happens in environments when patching is a monthly or quarterly task under change control, leaving organizations open to days or months of unnecessary exposure to fixed vulnerabilities.
- If software developers do not test the compatibility of updated, upgraded, or patched libraries.
- If you do not secure the components' configurations (see [A05:2021-Security Misconfiguration](#)).

How to Prevent

There should be a patch management process in place to:

- Remove unused dependencies, unnecessary features, components, files, and documentation.
- Continuously inventory the versions of both client-side and server-side components (e.g., frameworks, libraries) and their dependencies using tools like versions, OWASP Dependency Check, retire.js, etc. Continuously monitor sources like Common Vulnerability and Exposures (CVE) and National Vulnerability Database (NVD) for vulnerabilities in the components. Use software composition analysis tools to automate the process. Subscribe to email alerts for security vulnerabilities related to components you use.
- Only obtain components from official sources over secure links. Prefer signed packages to reduce the chance of including a modified, malicious component (See [A08:2021-Software and Data Integrity Failures](#)).
- Monitor for libraries and components that are unmaintained or do not create security patches for older versions. If patching is not possible, consider deploying a virtual patch to monitor, detect, or protect against the discovered issue.

Every organization must ensure an ongoing plan for monitoring, triaging, and applying updates or configuration changes for the lifetime of the application or portfolio.

Example Attack Scenarios

Scenario #1: Components typically run with the same privileges as the application itself, so flaws in any component can result in serious impact. Such flaws can be accidental (e.g., coding error) or intentional (e.g., a backdoor in a component). Some example exploitable component vulnerabilities discovered are:

- CVE-2017-5638, a Struts 2 remote code execution vulnerability that enables the execution of arbitrary code on the server, has been blamed for significant breaches.
- While the internet of things (IoT) is frequently difficult or impossible to patch, the importance of patching them can be great (e.g., biomedical devices).

There are automated tools to help attackers find unpatched or misconfigured systems. For example, the Shodan IoT search engine can help you find devices that still suffer from Heartbleed vulnerability patched in April 2014.

References

- [OWASP Application Security Verification Standard: V1 Architecture, design and threat modelling](#)
- [OWASP Dependency Check \(for Java and .NET libraries\)](#)
- [OWASP Testing Guide - Map Application Architecture \(OTG-INFO-010\)](#)
- [OWASP Virtual Patching Best Practices](#)
- [The Unfortunate Reality of Insecure Libraries](#)
- [MITRE Common Vulnerabilities and Exposures \(CVE\) search](#)
- [National Vulnerability Database \(NVD\)](#)
- [Retire.js for detecting known vulnerable JavaScript libraries](#)
- [GitHub Advisory Database](#)
- [Ruby Libraries Security Advisory Database and Tools](#)
- [SAFECode Software Integrity Controls \[PDF\]](#)

List of Mapped CWEs

[CWE-937 OWASP Top 10 2013: Using Components with Known Vulnerabilities](#)

[CWE-1035 2017 Top 10 A9: Using Components with Known Vulnerabilities](#)

[CWE-1104 Use of Unmaintained Third Party Components](#)