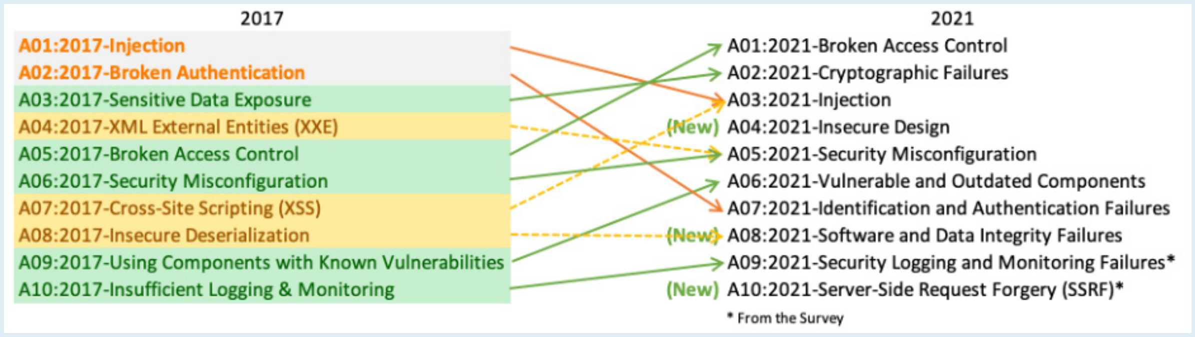
# OWASP TOP 10 FOR 2021

kiuwan

## INDEX

# OWASP TOP 10 FOR 2021

For as long as there has been software, there has also been open-source software. Over the years, different software foundations have worked to uphold the vision of open-source: peer production, transparency, and mass collaboration. The Open Web Application Security Project (OWASP) is one such organization.


2017
A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring
2021
A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*
* From the Survey

OWASP is a nonprofit foundation that works to make software more secure and accessible. For nearly two decades, OWASP has organized a community of tens of thousands of members with chapters worldwide.

You may be wondering how OWASP can help you and your dev teams create more secure software. Here's how: By informing you of and guiding you through the top 10 security vulnerabilities of 2021.

Part of OWASP's mission includes identifying vulnerabilities in software as they arise and helping development teams navigate through them. Ultimately, their goal is to help devs create more secure software through hands-on training and experience.

So stick with us to find out everything you need to know about this year's latest OWASP security vulnerabilities!

## WHAT IS THE OWASP TOP TEN?

Before we jump straight into what's new in the OWASP Top Ten 2021, you should first know about the OWASP Top Ten.

One of OWASP's most important goals is to help developers everywhere create more secure web applications. They achieve this by constantly monitoring software security threats as they arise and evaluating the threats based on their risk levels.

OWASP looks at all the security threats present online and ranks them according to the threat level they pose. The top 10 risks make it to the OWASP Top Ten, a list of the ten most critical web application security risks.

The OWASP Top 10 originally started in 2003 and has since grown over the years based on evolving threats and feedback from the dev community. These days, the OWASP Top 10 serves as a pseudo-standard for web application security worldwide.

OWASP released the latest version of the OWASP Top 10 on September 24, 2021.
The new version includes some significant structural changes from the last (2017) edition.

To summarize, OWASP has made the following three major changes to its Top 10 2021 list:

- ✓ **Added three new categories**
- ✓ **Changed the name and scope of some old categories**
- ✓ **Consolidated old categories into existing or new ones**

Keep on reading if you want a more detailed overview of OWASP Top 10 2021, including a summary of each category and version changes.

Note: Old category names (AXX:2017) are added along with the previous rank where applicable to track category name changes. Categories marked with an asterisk* are categories from the surveys.

## A01 BROKEN ACCESS CONTROL

**PREVIOUS RANK:** 5
**CONSOLIDATED:** NO

Jumping all the way to the top from fifth place back  in 2017 is A01:2021-Broken Access Control.

Broken Access Control refers to a violation of the access control policy.
The access control policy typically enforces permissions that bind users to specific actions at an access level.

However, attackers can sometimes bypass these policies, leading to unauthorized access.

The damage from unauthorized access depends on what an attacker chooses to do with sensitive data. In most cases, attackers will disclose, modify, or even destroy vulnerable information. In other cases, an attacker can also perform unauthorized business transactions outside their limits.

### Common Vulnerabilities

- **Insecure software API:** Attackers can bypass insecure software API using HTTP methods for restful services. These methods include missing access controls for the PUT, POST, and DELETE commands.
- **Unrestricted URL access:** When URLs are not adequately secured, attackers can bypass access control checks through them. For example, an attack can modify URL parameters to authenticate restricted pages as an unauthenticated user.

### Sample Attack Scenario: URL Tampering

An attacker can use unauthenticated data in an SQL query to access account information:

```
pstmt.setString(1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery( );
```

Once this command executes, all the attacker needs to do is modify their browser's account parameter "acct." If the web application fails to verify this parameter correctly, an attacker gains unauthorized access to their user account of choice.
https://example.com/app/accountInfo?acct=notmyacct

### Prevention Measures

- **Restrict software API access:** Enforce record ownership on model access control to restrict API access. This will prevent users from creating, updating, or deleting records.
- **Secure URLs:** Keep track of access control policy failures and alert the webmaster/site admins in case of repeated failed attempts.

## A02 CRYPTOGRAPHIC FAILURES

**PREVIOUS RANK:** **3; A03:2017-Sensitive Data Exposure**
**CONSOLIDATED:** **NO**

A02:2021-Cryptographic Failures have increased in frequency, stepping up a spot from third place back in 2017.

Previously referred to as Sensitive Data Exposure, Cryptographic Failures have been renamed to reflect the root cause of this problem rather than a resulting symptom. In other words, Cryptographic Failures are what can expose sensitive data.

Cryptographic systems protect sensitive data both in transit and at rest. Examples include passwords, credit card numbers, and health records, just to name a few. Such data is often subject to data privacy laws, such as the EU General Data Protection Regulation (GDPR).

### Common Vulnerabilities

- **Simple hash functions:** Usually, cryptographic hash functions such as SHA-2 are required for cryptographic systems, as they cannot be reversed. However, attackers can reverse engineer and exploit cryptographic systems that use simple non-cryptographic, unsalted, or outdated hash functions.
- **Weak crypto keys:** A weak key or key policy failure is often vulnerable to exploits. Examples include devs leaving keys in source code repos that are publicly available or reusing previously exposed crypto keys.

### Sample Attack Scenario: Reverse Engineering Simple Hash Functions

Imagine a password database that uses simple hash functions. An attacker can download the password database through a file upload weakness.

Since the hash function is simple, it most likely will not be salted. In that case, an attacker can expose unsalted hashes by looking up pre-calculated hashes from a rainbow table.

Worse still, attackers can also decrypt simple hash functions using powerful Graphical Processing Units (GPUs). This is true even if the simple hash function is salted.

### Prevention Measures

- **Use strong cryptographic hash functions:** Salted functions such as Argon2, bcrypt, and PBKDF2 are preferable, as they also have a work/delay factor. Avoid using deprecated functions like SHA-1.
- **Manage keys properly:** Make sure you randomly generate a cryptographic key and store it as byte arrays in your memory. Don't reuse exposed keys or leave keys out in the open where they can easily be found (e.g., a public source code repository).

## A03 INJECTION

**PREVIOUS RANK:** 1
**CONSOLIDATED:** YES; A07:2017-CROSS-SITE SCRIPTING (XSS)

Next up, we have A03:2021-Injection. Previously in the number one spot, the 2021 category of injection has also consolidated to include A07:2017-Cross-Site Scripting (XSS). XSS is now a special case of injection.

Injection attacks are cyberattacks where attackers inject malicious code into a network. When the network accesses a database, it also executes the injected code. The end result is usually that the attacker gains access to the database. Examples of common injection attacks include SQL injection, Object Relational Mapping (ORM), and Cross-Site Scripting (XSS).

### Common Vulnerabilities

- **Improper data handling:** Any data that users input to the web application is not verified or filtered by the web application. As a result, attackers can inject malicious code since there are no input restrictions.
- **Hostile data:** The application uses hostile data in its codebase. An SQL command with dynamic queries or object-relational mapping (ORM) with specific search parameters can both exploit this hostile data and extract sensitive information.

### Sample Attack Scenario: Hostile Data SQL Injection

If a web application uses hostile or untrusted data in its codebase, attackers can use an SQL injection attack. For this example, we assume an SQL injection attack on the Northwind Database, a sample database by Microsoft.

The following query is an example of SQL injection:

```
String query = "SELECT \* FROM CUSTOMERS WHERE CustomerID='" + request.getParameter("id") + "'";
```

In the query above, attackers will modify the 'id' parameter by sending the following value:

```
http://sample.com/app/accountView?id=' or '1'='1
```

As a result, both queries will return all the entries from the CUSTOMERS table. These entries include sensitive information such as customer name, phone number, and address. Other SQL injection attacks can insert, update, or even delete data entries.

### Prevention Measures

- **Secure data checks:** Use a "whitelist" for input validation. The input validation happens on the server-side and filters most data for malicious entries.
- **Query controls:** Certain SQL keywords and controls can prevent retrieving all data records in case of an SQL injection attack. An example is the LIMIT keyword.

## A04 INSECURE DESIGN

**PREVIOUS RANK:** N/A (NEW)
**CONSOLIDATED:** NO

A04:2021-Insecure Design is an entirely new category that did not exist in the 2017 OWASP Top 10 list.

Insecure Design is all about web application weaknesses that result from poor design practices, such as ineffective control design. Note that Insecure Design is not the same as all other categories, as the other categories are more about insecure implementation rather than insecure design.

This category is a fairly broad category that encompasses various vulnerabilities resulting from poor design practices. As an example, consider a dev team that writes software without running a business risk profile. Without a risk profile, the devs would not know how vulnerable the data they're handling is and thus forget to implement proper security checks.

### Common Vulnerabilities

- **Security requirements are poorly defined:** The software engineering team does not properly identify the functional and non-functional security requirements of the software.
- **Poor business logic:** The team does not properly identify the business logic of the software. included here would be who will be using the software, how they will use it, and the nature of the data the software will handle.

### Sample Attack Scenario: Business Logic Exploit

Imagine a restaurant running a special promotion that gives group discounts on reservations of ten or more people.

If the reservation software has no restrictions on the reservations, a single attacker can book all the seats in the restaurant simultaneously. The attacker would only need a few requests to accomplish this.

This exploit would result in a massive loss for a restaurant as all their seats would be reserved with fraudulent diner entries. From this example, it should be clear that hackers don't always need to use code to exploit software weaknesses.

### Prevention Measures

- **Use secure design practices:** The software dev team should employ a secure development lifecycle and have a library of secure design patterns ready at hand. We strongly recommend consulting AppSec professionals for this counter-measure.
- **Restrict the business logic:** Devs can achieve this by placing limits on resource consumption by users. Such restrictions can prevent scenarios like the one highlighted above.

## A05 SECURITY MISCONFIGURATION

**PREVIOUS RANK:** 6
**CONSOLIDATED:** YES; A04:2017-XML External Entities (XXE)

Not only did OWASP bump up A05:2021-Security Misconfiguration to fifth place, but they also consolidated it with A04:2017-XML External Entities (XXE).

Since OWASP also identified XXE as a specific case of Security Misconfiguration, the former is no longer a separate category of its own.

Security misconfigurations are a result of not properly implementing security protocols in the web application. By not following the best practices for security or not remaining vigilant, devs risk exposing their software to cyberattacks.

### Common Vulnerabilities

- **Inappropriate security hardening:** Devs can accidentally reveal stack traces to users or fail to authorize cloud service access properly. Too many unnecessary features for the users provide more room for attackers to find exploits.
- **Improper security settings:** Examples include outdated software, no security headers on the server, and application frameworks using insecure values.

### Sample Attack Scenario: Stack Trace Exploit

When devs don't harden their applications properly, they risk revealing too much to the user. Savvy hackers can use even a small piece of information to engineer an attack.

Consider an application server that has a poor configuration. The server accidentally displays comprehensive error messages, which include information about the stack trace.

Even though the stack trace is not a vulnerability by itself, it can contain potentially vulnerable information. Attackers can use stack traces to engineer future attacks, as they can learn of lower-level system information (memory addresses, pointers, etc) through the stack trace.

### Prevention Measures

- **Minimize user features:** Ensure that your software only does what it needs to and nothing else. Extra features, components, libraries, documents, and error messages are all potential security risks.
- **Properly configure security settings:** This effort should be designated as a task on its own rather than a trivial check. Reviewing the application configuration, checking for the latest security patches, and updating the software should all be a part of the process.

## A06 VULNERABLE AND OUTDATED COMPONENTS*

| | |
|---|---|
| **PREVIOUS RANK:** | **9; A09:2017-Using Components With Known Vulnerabilities** |
| **CONSOLIDATED:** | **NO** |

Previously known as Using Components With Known Vulnerabilities, OWASP has renamed this category to A06:2021-Vulnerable and Outdated Components* in the latest Top 10 list. The name change reflects how vulnerable components are a security risk on their own, even when they're present and not used.

The community survey identified Vulnerable and Outdated Components as #2 on their list of vulnerabilities. The response alone should indicate how seriously devs everywhere take the threat of using vulnerable software components.

It's not just a case of A05:2021-Security Misconfiguration, but also a separate category of its own. As such, A06 is a combination of both bad software design and poor implementation.

### Common Vulnerabilities

- **Using outdated or vulnerable software:** The Operating System (OS), Database Management System (DBMS), web server, APIs, etc., are not up-to-date or not considered secure.
- **Poor version control:** Devs use server-side or client-side components without knowing what version they're using. Also, devs do not test the compatibility of different software package libraries when they update them.

### Sample Attack Scenario: The Apache Struts Vulnerability

Apache Struts is an open-source web application development framework that is used by devs worldwide. Researchers were able to uncover the CVE-2017-5638 vulnerability in this framework.

As a result of a code vulnerability in the Jakarta Multipart parser, attackers can run remote command injection attacks by parsing an invalid HTTP content header.

### Prevention Measures

- **Discard risky components:** Always discard unused features, libraries, files, or components as soon as you identify them. Additionally, do not use code components that are outdated or no longer supported.
- **Look out for security breaches:** Even software components from a trusted source, such as Apache, can become vulnerable sometimes. Keep your eyes peeled for any vulnerabilities and respond accordingly as soon as you identify them.

## A07 IDENTIFICATION AND AUTHENTICATION FAILURES

**PREVIOUS RANK:** **2; A02:2017-Broken Authentication**
**CONSOLIDATED:** **NO**

There are times when a fall from grace can be a good thing. That's why we're relieved to report that A07:2021-Identification and Authentication Failures are no longer the massive threat they used to be, falling down all the way to seventh place from their previous second spot.

Previously known as Broken Authentication, A07 has expanded its scope to include a wide range of Identification and Authentication Failures.

One of the most important tasks of a web application is to confirm and authenticate the identity of a user logging into the software. That said, the application can sometimes fail to properly authenticate users, opening the possibility of cyberattacks.

### Common Vulnerabilities

- **No login condition checks:** It is easy to exploit software that does not have checks in place for users logging in. For example, if the system has no limitations on the number of failed login attempts, hackers can brute-force their way into an account or use credential stuffing.
- **Poor session handling:** Poorly written software will either expose the session identifier in the URL or reuse a session identifier from a successful login session.

### Sample Attack Scenario: 2014 iCloud Leaks

In 2014, a collection of personal photographs of popular Hollywood celebrities was leaked online.

The attackers managed to obtain the photographs through iCloud. Attackers were able to use a combination of phishing and brute-force guessing to hack into the celebrity accounts.

At the time, Apple did not have lockout mechanisms in place for unsuccessful iCloud login attempts.
As a result, the attackers had an unlimited number of attempts to try and access an account.
They were thus able to brute-force their way into celebrity iCloud accounts and obtain sensitive data.

### Prevention Measures

- **Enforce login checks:** Multi-factor authentication and locking users out after multiple failed login attempts are effective ways to prevent failures.
- **Password checks:** Enforce strict rules for passwords. Forcing special characters, long and complex passwords can reduce the effectiveness of brute-forcing and credential stuffing.

## A08 SOFTWARE AND DATA INTEGRITY FAILURES

**PREVIOUS RANK:** N/A (NEW)
**CONSOLIDATED:** YES; A08:2017-Insecure Deserialization

Both a new category on its own and consolidated with Insecure Deserialization from the 2017 list, A08:2021-Software and Data Integrity Failures are number eight on the latest OWASP Top 10.

Software and data integrity failures arise when the integrity of the software code or infrastructure is violated.

This category targets assumptions related to updating software, CI/CD, or new modules without checking their integrity first.

### Common Vulnerabilities

- **Using untrusted sources:** Getting code from untrusted repositories, libraries, or Content Delivery Networks (CDNs) can lead to vulnerabilities in the software that attackers can exploit.
- **Auto-update functionality:** While developers can benefit from auto-updates most of the time, they are also a potential security risk. Sometimes, the update can include vulnerable code.

### Sample Attack Scenario: SolarWinds Update

The SolarWinds malicious software update was one of the largest data integrity failures in history.

Attackers were able to bypass security checks from SolarWinds, a software development IT company, and distribute malicious updates to over 18,000 organizations that used the company's services.

As a result of this data integrity failure, companies suffered massive losses to their revenue and business operations.

### Prevention Measures

- **Use digital signatures:** Digital signatures can ensure any software or module updates are from verified sources and not imposters.
- **Deploy a security tool:** Sometimes, a software security tool can verify that the software supply chain is legitimate and does not contain vulnerabilities. The OWASP Dependency Check is an easy-to-use tool that can verify software supply chains.

## A09 SECURITY LOGGING AND MONITORING FAILURES*

**PREVIOUS RANK:** **10; A10:2017-Insufficient Logging & Monitoring**
**CONSOLIDATED:** **NO**

Previously on number ten, A09:2021-Security Logging and Monitoring Failures also placed third according to the OWASP Top 10 community survey.

It's one thing to prevent security breaches. It's another problem entirely to adequately monitor and contain them when they happen.

A09 is all about detecting and responding to security breaches. When IT and network admins fail to detect breaches, they cannot contain the breach in time and risk damaging the software even more.

A secure logging and monitoring protocol is therefore needed to minimize the damage from security breaches.

### Common Vulnerabilities

- **Improper log handling:** When a company fails to monitor application and API logs frequently, it will fail to detect suspicious activity and breaches. Another problem is when the system only stores logs locally, therefore restricting access.
- **Poor log design:** Even if a company frequently monitors logs, the logs are not helpful if they do not track major network events such as multiple failed login attempts.

### Sample Attack Scenario: Improper Log Handling

A hospital uses a software management system. Attackers were able to successfully breach the system, modify patients' personal health records, and extract their data.

However, a lack of log monitoring resulted in the breach going undetected until it was too late. Without log monitoring, there is no way of knowing how long the breach was in place; the system could have been compromised anywhere from a couple of hours to months.

### Prevention Measures

- **Monitor logs frequently:** Keep a protocol in place for frequently checking and monitoring logs. A DevSecOps team should be in charge of this monitoring.
- **Design better logs:** The logs should contain all the necessary information for monitoring breaches and should be encoded to prevent injection attacks.

## A10 SERVER SIDE REQUEST FORGERY (SSRF)*

**PREVIOUS RANK:** N/A (NEW)
**CONSOLIDATED:** NO

Finally, we have another new category for the latest OWASP Top 10: A10:2021-Server Side Request Forgery (SSRF).

SSRF ranked first in the OWASP community survey. This category deals with web applications that fetch remote resources without first verifying the URL. Attackers can therefore forge user-supplied URLs to launch cyberattacks.

Unfortunately, SSRF attacks are now becoming more prevalent. That said, as long as you know the threat that SSRFs pose, you can protect your software against them.

### Common Vulnerabilities:

- **URL Inconsistency:** Attackers can use inconsistent URLs to launch attacks such as DNS rebinding.
- **Poor firewall policies:** The software's firewall allows most remote traffic through, allowing attackers to break in.

### Sample Attack Scenario: Access Local Files

An attacker can use the following commands to access local files on the software's network:

```
file:///etc/passwd</span>
http://localhost:28017/.
```

### Prevention Measures:

- **Disable HTTP Redirections:** By disabling HTTP redirections, you can prevent SSRF attacks through malicious URL requests.
- **Strengthen firewalls:** Configure your network firewall to block all but essential network traffic.

## FINAL THOUGHTS

As more people start using web applications worldwide, the number of cyberattacks and severity of security risks will only increase. Cloud application security is an ever-increasing critical aspect of application development.

Fortunately, there is a way you can protect your software against the Top 10 vulnerabilities of 2021: with the help of Kiuwan application security solutions.

Kiuwan offers two solutions that can help you build more secure web applications today: Static Application Security Testing (SAST) and SCA insights.

SAST is an automatic code scanning tool that scans your code for security vulnerabilities, design weaknesses, or other potential exploits. The entire process takes minutes on your local machine, and you can share your scan results on the cloud once you're done.

What's more, you can use SAST to build a customized action plan to prepare your web application against different cyberattack scenarios and optimize your response to them.

Additionally, Kiuwan's SCA Insights also helps you manage the open-source risk of your software. The more you use third-party components in your software, the greater your risk of exploits. That's why SCA Insights uses open source code validation to identify code components, detect threats, and avoid obsolescence.

**The best part?** Kiuwan's products are fully compliant with IT security standards, including CERT, NIST and OWASP.

Try out SAST and SCA Insights for your web applications today, and you'll be prepared for even the worst cyber attacks and software vulnerabilities of 2021 in no time!
**DON'T WAIT FOR AN ATTACK** REQUEST A **FREE TRIAL** OR A **DEMO** TODAY

## GET IN TOUCH

**Headquarters**
2950 N Loop Freeway W, Ste 700
Houston, TX 77092, USA

United States: **+1 732 895 9870**
Asia-Paciifc, Europe, Middle-East and Africa: **+ 44 1628 684407**

**contact@kiuwan.com**
Partnerships: **partners@kiuwan.com**