

Whitepaper

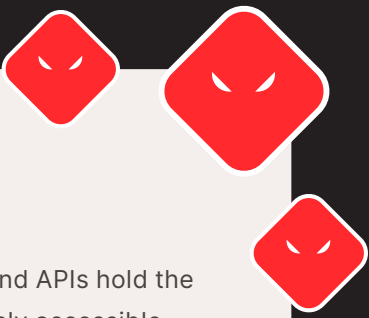
Navigating the OWASP Top 10 with Fastly



Insights, strategies, and real-world
examples for improved web
application security

Table of Contents

- 01 Broken Access Control**
- 02 Cryptographic Failures**
- 03 Injection**
- 04 Insecure Design**
- 05 Security Misconfiguration**
- 06 Vulnerable and Outdated Components**
- 07 Identification and Authentication Failures**
- 08 Software and Data Integrity Failures**
- 09 Security Logging and Monitoring Failures**
- 10 Server-Side Request Forgery**



Introduction

Any organization is a target: attackers know that web apps and APIs hold the keys to the kingdom. Once they've breached an app or publicly accessible API, they can act on their objectives. To better help developers and security practitioners understand the threats and vulnerabilities, the OWASP Top 10 list was developed to help foster a culture of secure software development.

The [OWASP Top 10](#) is a reference standard created by the non-profit OWASP (Open Web Application Security Project) Foundation that provides a ranking of and remediation guidance for the top 10 most critical web application security risks. It's based on a consensus among security experts from around the world and is intended to drive focus and clarity to the complex world of web application security.

OWASP has been updating this list every few years since 2003, based on advancements and changes in the appsec industry. With the major updates in the latest version in 2021, there are three new categories, four categories with naming and scoping changes, and some consolidation in the Top 10.

According to the [2023 Verizon DBIR](#), a staggering 60% of all breaches occurred through web applications. It's tempting to think that addressing the OWASP top ten can be achieved with a single product or process when in reality organizations with heterogeneous applications in multiple locations have to think holistically about application security. Attackers will always look for the weakest link. This often means addressing the OWASP Top 10 is a byproduct of consistent security policies across on-premise and cloud environments and involves products, processes, and improved developer education.

This paper will break down the Top 10, provide examples, and describe how Fastly can help support your organization against these threat categories.

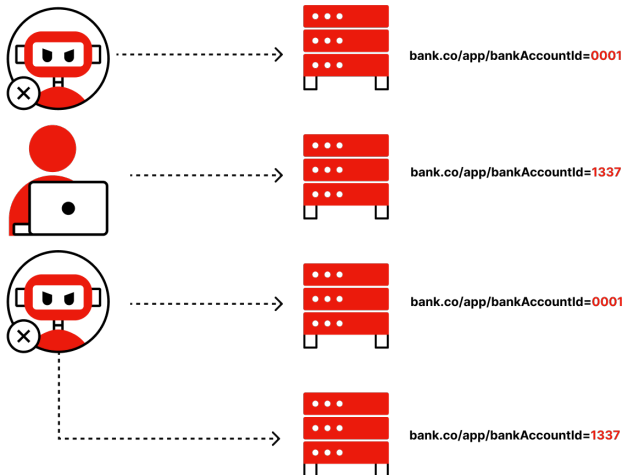
A01:2021

Broken Access Control

This category represents the most serious application security risk: If an attacker can bypass access control with the right privilege level, they can do some serious damage. According to OWASP, 94% of applications were tested for some form of broken access control and it has the most occurrences in the contributed dataset. This category covers the largest number (34) of different [Common Weakness Enumerations](#), or CWEs that can allow attackers to access data, resources, user accounts, or operations that they shouldn't be permitted to access.

Example

An attacker (user id 1337) simply browses to a target URL and tells the application they are another user (user id 1001). The application fails to enforce any access control and allows the request to be seen as coming from this other user.



Prevention

Organizations can protect themselves from Broken Access Control threats by ensuring custom-built applications have been built with a secure development lifecycle in mind and have undergone security penetration testing. In addition to this, deploying a tool such as a WAF can augment and enhance your security posture.

Access control is only effective in trusted server-side code or server-less API, where the attacker cannot modify the access control check or metadata. Or in the example listed above, the [route-controller should be checking authorization](#) and thus enforcing access controls of ownership rather than accepting that the user can read, create, update, or delete any record.

Fastly Protection

An attacker (user id 1337) simply browses to a target URL and tells the application they are another user (user id 1001). The application fails to enforce any access control and allows the request to be seen as coming from this other user.

- **Next-Gen WAF** can be configured with allow/block lists to enforce that they are only accessible by authenticated users and/or administrators. This capability works in conjunction with our Edge Cloud Platform. Security decisions made by the Next-Gen WAF can be pushed closer to the attacker using access control lists (ACLs), stopping malicious attackers long before they reach the application.
 - **Next-Gen WAF** by default looks for requests attempting forceful browsing, directory traversal, or access to private files or directories such as `"/web_app/WEB-INF/webapp.properties"`. Simple configurations can be set up to extend this
- detection to other directories or file extensions. This will enforce the OWASP recommendation:
- *Disable web server directory listing and ensure file metadata (e.g. .git) and backup files are not present within web roots.*
- **Next-Gen WAF** protects against brute force attacks such as credential stuffing by detecting such attacks and blocking the attackers' login attempts. All failed login attempts are logged and administrators are immediately alerted when the attacker is detected. This follows the OWASP recommendation to:
 - *Log access control failures, alert admins when appropriate (e.g. repeated failures).*
 - **Next-Gen WAF** can rate limit API requests and alert when thresholds have been crossed. This follows the OWASP recommendation:
 - *Rate limit API and controller access to minimize the harm from automated attack tooling.*

A02:2021

Cryptographic Failures

This category focuses on failures related to cryptography (or lack thereof). A lack of, or poorly implemented, cryptography often leads to exposure of sensitive data (e.g. passwords, credit card numbers, health records, personal information, and business secrets).

Example

This common example scenario often exists in many environments:

The password database uses unsalted or simple hashes to store everyone's passwords. A file upload flaw allows an attacker to retrieve the password database. All the unsalted hashes can be exposed with a rainbow table of pre-calculated hashes. Hashes generated by simple or fast hash functions may be cracked by GPUs, even if they were salted.

Prevention

Organizations can reduce the risk of cryptographic failures by using standardized cryptographic libraries rather than hand-rolled functionality. This in addition to thorough code reviews and penetration testing can provide an initial layer of defense against these threats, but more may be needed.

The OWASP recommendation to prevent this attack scenario is to store passwords using strong adaptive and salted hashing functions with a work factor (delay factor), such as Argon2, scrypt, bcrypt or PBKDF2:

A practical example of good best practice would be using the [bcrypt function](#) (hopefully from a library, from your language of choice), based on the Blowfish cipher. This function incorporates a salt and is an adaptive function, meaning over time the iteration count can be increased making it slower. Which means it will remain resistant to brute force attacks over time.

Fastly Protection

- **Fastly Edge Cloud Platform** handles TLS termination and acceleration at a global scale with secure defaults (such as TLS 1.3 and removal of weak ciphers) for customers. We can ensure that weak encryption ciphers and protocols are not permitted for requests to sensitive areas of the application or for any application requests. This will enforce the OWASP recommendation:
 - *Encrypt all data in transit with secure protocols such as TLS with perfect forward secrecy (PFS) ciphers, cipher prioritization by the server, and secure parameters. Enforce encryption using directives like HTTP Strict Transport Security (HSTS).*
 - *Additionally, our programmable edge network offers developer tools for cryptographic hashing and digital signature verification. Centralizing security with common tools reduces the risks of organizations "rolling their own cryptography" which can lead to increased risks.*
- **Next-Gen WAF** can log requests and alert if required response security headers are ever missing.

A03:2021

Injection

Some of the more common injection attacks are SQL, shell & command line, Object Relational Mapping (ORM), LDAP, and Object Graph Navigation Library (OGNL) injection (think struts - template injections). OWASP also added cross-site scripting (XSS) to this category in 2021. XSS was previously in its own category - putting it under Injection doesn't mean that XSS is less severe now, it's just that other vulnerabilities have become more prevalent in recent years.

This category has the second most occurrences in applications. Let's dig in to define what exactly is injection and how it can occur:

Example

An application uses untrusted data extracted from an inbound user query in the construction of the following vulnerable SQL call:

```
String query = "SELECT \* FROM  
accounts WHERE custID=" + request.  
getParameter("id") + "";
```

Prevention

The best way to prevent injection attacks is to keep data separate from commands and queries by using parameterized queries and similar constructs ([ORM tools](#)). Some other defenses are using positive or "allow-list" server-side input validation. This is not a complete defense as many applications require special characters, such as text areas or APIs for mobile applications. Many frameworks use this approach by default, which might explain the lower rank of this category compared to previous editions.

Fastly Protection

- **Next-Gen WAF** by default detects injection attacks without the need for additional rules or tuning. Traditional WAF technologies rely on tens of thousands of regular expression (regex) patterns to detect various injection attacks such as SQLi. The number of patterns needed to detect injection attacks continues to grow over time as does the computing power to inspect every HTTP request against that ever-growing list of regex patterns. The number of false positives typically associated with regex pattern matching has necessitated either dedicated, internal WAF tuning teams or outsourced WAF management at a high cost, if WAF blocking mode is to be maintained. This problem is exacerbated in an environment when new application versions are rapidly released.
- **Next-Gen WAF** utilizes SmartParse, its proprietary detection technology, designed to make instantaneous decisions in line to determine if there are malicious or anomalous payloads present. By evaluating the context of the request and how it would actually execute, SmartParse makes highly accurate decisions. SmartParse is continually updated for efficacy based both on internal research as well as a closed-loop in-product false positive feedback feature. Compared to WAFs that rely on regex matching that are rarely used in blocking mode, 90% of Fastly customers enable full blocking mode across all default attack types without any tuning.

A04:2021

Insecure Design

This broad category represents different weaknesses, expressed as “missing or ineffective control design.” Among the CWEs in the category, you will find quite a few things grouped together including insufficient restrictions, incorrect privilege assignment, cached cookies containing sensitive information, and many other flaws that are not directly testable vulnerabilities. Unfortunately, some of these category groupings will be difficult to think of as a single category for many developers and may potentially lead to some being missed. It might be helpful for developers to look at each component (authentication, authorization, checkoutwvv) of their application and break it down into the simplest pieces. Once broken down it will become easier to test each piece individually and will generally lead to better outcomes both from a security and reliability standpoint. The reason this leads to generally better outcomes is attackers will typically look at a specific attack vector (think broken authorization design) and see how many other targets it applies to.

Example

A cinema chain allows group booking discounts and has a maximum of fifteen attendees before requiring a deposit. Once an attacker determines the attendee limit, they will try to exploit and test if they could book six hundred seats and all cinemas at once in a few requests, causing a massive loss of income.

Prevention

Due to the nature of this category, it is harder to test. The best way to prevent exploitation will require a change in the way that the code is written and tested, really utilizing that “[shift-left](#)” mentality. Below are some examples of ways to prevent these types of vulnerabilities in your code:

- Use threat modeling for critical authentication, access control, business logic, and key flows.
- Write unit and integration tests to validate that all critical flows are resistant to the threat model. Compile use cases and misuse cases for each tier of your application.
- Limit resource consumption/accessibility by user or service.

Fastly Protection

- **Next-Gen WAF** allows for custom signals to be created to monitor activity around susceptible routes or flows in an application. For example, if an application is built around a /checkout flow and this is a sensitive application route, you can attach a custom signal to client requests for this part of the application. The custom signal will allow you to monitor activity and create blocking or rate-limiting rules based on the properties of the request in the face of abusive behavior.

A05:2021

Security Misconfiguration

This category means the application is missing some security hardening or has some configuration settings that have not been applied correctly such as directory listing enabled, default accounts enabled, or missing security headers. This category also contains [XML External Entity \(XXE\) injections](#) which was a separate category in the previous iteration.

Example

Directory listing is not disabled on the server. An attacker discovers they can simply list directories. The attacker finds and downloads the compiled Java classes, which they decompile and reverse engineer to view the code. The attacker then finds a severe access control flaw in the application.

Fastly Protection

- **Next-Gen WAF** can alert administrators when server responses do not contain required security headers. This addresses the OWASP application vulnerability concern:
 - *The server does not send security headers or directives or they are not set to secure values.*
- For XXE, the **Next-Gen WAF** inspects XML payloads for injection attacks such as SQLi and command injection. This will help address the following OWASP recommendation:
 - *Implement positive ("whitelisting") server-side input validation, filtering, or sanitization to prevent hostile data within XML documents, headers, or nodes.*
 - If an application should not be accepting XML in HTTP requests, Fastly can look for and alert or block requests containing XML payloads.
- **Next-Gen WAF** can also be configured to look for sensitive data patterns in JSON payloads that can be blocked or logged. Whether blocking or logging, Fastly can immediately alert administrators when sensitive data patterns are found in payloads. This will help address the following OWASP recommendation:
 - *Whenever possible, use less complex data formats such as JSON, and avoid serialization of sensitive data.*
- Virtual patching can help address new data leakage issues until the application is patched.
- The **Fastly Edge Cloud Platform** can also be configured to add security headers such as Content-Security-Policy and Strict-Transport-Security. Automatically setting response headers returned to clients from the edge can provide another layer of protection against improperly configured applications or origin servers.

Prevention

While prevention is difficult for this category some simple procedural items once implemented would likely detect and assist in remediation before the attack can exploit these vulnerabilities. An example of that would be a repeatable hardening process such as a default hardened (think [golden image](#)) apache or nginx configuration file, which will make it fast and easy to deploy another, appropriately locked down, environment.

Development, QA, and production environments should all be configured identically, with different credentials used in each environment. This process should be automated to minimize the effort required to set up a new secure environment. In addition, an automated process is needed to verify the effectiveness of the configurations and settings in all environments. The ideal for these environments is a minimal platform without any unnecessary features, components, documentation, or samples.

A06:2021

Vulnerable and Outdated Components

Vulnerable and outdated software components include libraries and supporting services with known security flaws or out of date security updates. Consistently updating these components is a problem that Enterprises struggle with, especially with large and legacy codebases.

Example

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't be ignored when they are left publicly exposed (NVRs) or provide critical capabilities (biomedical devices).

Prevention

While there are tools and products to assist in this, it is still very difficult to protect against this particular vulnerability class. A good place to start is to ensure an ongoing plan for monitoring, triaging, and applying updates or configuration changes for the lifetime of the application or portfolio. Some steps to do this might include continuously inventorying 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 monitoring sources like Common Vulnerability and Exposures (CVE) and National Vulnerability Database (NVD) for vulnerabilities in the components can make you aware of critical bug patches. Using software composition analysis (SCA) tools can help to automate the process and lessen the work required. Subscribing to email alerts for security vulnerabilities related to components you use is a method that can bring further context to your environment beyond just CVE monitoring.

Fastly Protection

- **Next-Gen WAF** provides virtual patching for multiple CVEs (e.g. CVE-2023-34362 MOVEit SQL Injection Vulnerability). Fastly also allows customers to create virtual patches for vulnerabilities that are unique to their application while an internal patch/fix is developed and released. This will help address the following OWASP recommendations:
- *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.*

A07:2021

Identification & Authentication Failures

Identification and Authentication Failures include poor password management, failure of rate-limiting for credential stuffing attacks, insecure session management, and authentication bypasses.

Example

Credential stuffing is a good example of this category. Using lists of commonly used passwords, or those from data breaches to access accounts with these known passwords is a common attack. An application that does not implement automated threat or credential stuffing protection can be used as a password oracle to determine if the credentials are valid. Even if your application's password and session policies and protection are well-designed and secure, the application can be vulnerable to credential stuffing attacks using credentials stolen from less secure applications.

Prevention

The prevention methods are fairly straightforward in theory but are harder to implement in reality. This category requires the strategy of defense in depth but here's a starting point:

- Where possible, implement multi-factor authentication to prevent automated credential stuffing, brute force, and stolen credential reuse attacks.
- Implement weak password checks, such as testing new or changed passwords against the top 10,000 worst passwords list.
- Limit or increasingly delay failed login attempts, but be careful not to create a denial of service scenario. Log all failures and alert administrators when credential stuffing, brute force, or other attacks are detected.

Fastly Protection

- **Next-Gen WAF** protects against brute force attacks such as credential stuffing by detecting such attacks and blocking the attacker's login attempts. All failed login attempts are logged and administrators are immediately alerted when the attacker is detected. This follows the OWASP recommendation to:
 - *Limit or increasingly delay failed login attempts. Log all failures and alert administrators when credential stuffing, brute force, or other attacks are detected.*
- **The Next-Gen WAF** can also be configured to enforce the OWASP recommendation to ensure "session IDs should not be in the URL" or simply alert the administrator when a session ID is found in a request URL.

A08:2021

Software and Data Integrity Failures

Software and data integrity failures relate to code and infrastructure that do not protect against integrity violations. An example of this is where an application relies upon plugins, libraries, or modules from untrusted sources, repositories, and content delivery networks ([think about the ua-parser-js takeover](#)). An insecure CI/CD pipeline can introduce the potential for unauthorized access, malicious code, or system compromise (e.g., [Kaseya supply chain attack](#)). Insecure Deserialization from 2017 is now a part of this category.

Example

SolarWinds malicious update: Nation-states have been known to attack update mechanisms, with a recent notable attack being the [SolarWinds Orion attack](#). Solarwinds had secure build and update integrity processes. Still, these were able to be subverted, and for several months, the firm distributed a highly targeted malicious update to more than 18,000 organizations, of which around 100 or so were affected. This is one of the most far-reaching and significant breaches of this nature in history.

Prevention

- Use digital signatures or similar mechanisms to verify the software or data is from the expected source and has not been altered.
- Ensure libraries and dependencies, such as npm or Maven, are consuming trusted repositories. If you have a higher risk profile, consider hosting an internal known-good repository that's vetted.
- Ensure that your CI/CD pipeline has proper segregation, configuration, and access control to ensure the integrity of the code flowing through the build and deploy processes.

Fastly Protection

- **Next-Gen WAF** inspects all requests including payloads for attacks such as injection attacks. This allows the blocking of serialized objects containing attacks such as XSS, SQLi, directory traversal and command execution.
- **Next-Gen WAF** can also assist by looking for excessive deserialization requests from a source. Once a threshold has been crossed, the administrator can be notified and a temporary block of subsequent deserialization requests can be enabled via rate-limiting. This will help address the following OWASP recommendation:
 - *Monitoring deserialization, alerting if a user deserializes constantly*
- **Next-Gen WAF** can also be set up to detect if the request payloads contain serialized payloads that are Base64 encoded. Other serialization patterns can also be specified for detection:
 - "r00" in Base64
 - Content-type = 'application/x-java-serialized-object'
- If Base64 encoded serialized payloads and/or specific content types should never be sent in a request they can simply be blocked. Base64 encoded serialization requests can also be decoded and inspected for blacklisted Java classes. This provides look-ahead class validation and helps address the following OWASP recommendation:
 - *Enforcing strict type constraints during deserialization before object creation as the code typically expects a definable set of classes.*
- Malformed Deserialization Requests - By default, the **Next-Gen WAF** parses XML and JSON payloads and flags the request if the payload is malformed without the need for rules.
- Virtual patching can help address new deserialization flaws until the application is patched.

A09:2021

Security Logging and Monitoring Failures

This category is to help detect, escalate, and respond to active breaches. Without logging and monitoring, breaches cannot be detected and failures in this category can directly impact visibility, incident alerting, and forensics.

Fastly Protection

Example

An external party informed a children's health plan provider that an attacker had accessed and modified thousands of sensitive health records of more than 3.5 million children. A post-incident review found that the health plan provider's website developers had not addressed significant vulnerabilities. As there was no logging or monitoring of the system, the data breach could have been in progress since 2013, a period of more than ten years.

Prevention

Developers should implement the following controls, depending on the risk of the application:

- Ensure all login, access control, and server-side input validation failures can be logged with sufficient user context to identify suspicious or malicious accounts and held for enough time to allow delayed forensic analysis.
- Ensure high-value transactions have an audit trail with integrity controls to prevent tampering or deletion, such as append-only database tables or similar.
- Establish or adopt an incident response and recovery plan, such as [NIST 800-61r2](#) or later.

- **Next-Gen WAF** monitors all incoming HTTP requests and responses for attacks and anomalies. This includes monitoring and logging all login attempts, successes, and failures. When an automated logging or blocking decision is made, alert notifications are automatically sent to administrators through configured channels via email or webhook integration. All Next-Gen WAF event and request data can also be imported via API into any system for long-term log archives in JSON format. This will help address the following OWASP recommendations:
 - *Ensure all login, access control failures, and server-side input validation failures can be logged with sufficient user context to identify suspicious or malicious accounts, and held for sufficient time to allow delayed forensic analysis.*
 - *Ensure that logs are generated in a format that can be easily consumed by a centralized log management solutions.*
 - *Ensure high-value transactions have an audit trail with integrity controls to prevent tampering or deletion, such as append-only database tables or similar.*
 - *Establish effective monitoring and alerting such that suspicious activities are detected and responded to in a timely fashion.*
- **Next-Gen WAF** sends out real-time event notifications to a list of default webhook integrations in the dashboard such as Slack, PagerDuty and Datadog in addition to any notification receiver solution that can accept a webhook.
- SIEM integration (e.g Elastic, ELK Stack, Sumo Logic) is straightforward and allows for storage of security events and details as long as needed or required.
- Fastly's [Managed Security Service](#) provides 24/7 proactive monitoring and remediation from our Customer Security Operations Center (CSOC).

A10:2021

Server-Side Request Forgery

SSRF flaws occur when a web application is fetching a remote resource without validating the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network access control list (ACL). You can think of SSRF as enabling a kind of evil reverse proxy for attackers to use in their operations.

As modern web applications provide end-users with convenient features (e.g., talking to multiple backend systems), fetching a URL becomes a common scenario. As a result, the incidence of SSRF is increasing. The severity of SSRF is also increasing due to cloud services and the complexity of architectures.

Example

Accessing metadata storage of cloud services – Most cloud providers have metadata storage such as `http://169.254.169.254/`. An attacker can read this metadata to gain sensitive information.

Prevention

Because of the complexity of the attack surface for SSRF attacks, implementing defense in depth controls is the best way to successfully defend against this particular attack.

From the network layer, you can segment remote resource access functionality in separate networks to reduce the impact of SSRF.

From the application layer, a few of these options include: sanitize and validate all client-supplied input data, disable HTTP redirection, enforce the URL schema, port, and destination with a positive allow list. **Do not mitigate SSRF via the use of a deny list or regular expression. Attackers have payload lists, tools, and skills to bypass deny lists.**

Fastly Protection

- **Next-Gen WAF:** Through [NLX](#) and our [security research team](#) we rapidly roll out mitigations to protect against new, pervasive threats, including SSRF. For example, we include a templated rule that addresses AWS-specific SSRF.
- With the **Next-Gen WAF** you can add a custom rule to block all requests with invalid Host headers (e.g. ones not in your allowed list of domains) to prevent Host Header SSRF.



Conclusion

By using a combination of default detections plus customizable functionalities in the Next-Gen WAF and leveraging the capabilities of our Edge Cloud Platform, Fastly customers can gain strong coverage of the OWASP Top 10. Expand coverage even further through implementing OWASP and Fastly-recommended best practices to shore up your application security skills and capabilities to better protect customer and corporate data.

To learn more and see how Fastly can help you mount a strong defense against OWASP threats, [contact us](#).

Further reading

- [OWASP.org](#)
- [Detection and Blocking with the Next-Gen WAF](#)
- [Ten Capabilities of the Next-Gen WAF](#)