radware

Understanding the OWASP TOP 10 and How WAFs Can Mitigate Them

Cars require seatbelts. Pill bottles need safety caps. Applications need web application firewalls (WAFs), bot management and API protection – and for good reason. The web application threat landscape is in a constant state of flux. From DevOps to new attack vectors, these changes can leave security professionals scrambling to safeguard their most prized digital assets.

The Open Web Application Security Project (OWASP) Top 10 list is an invaluable tool for accomplishing this. Since 2003, this list has sought to provide security professionals with a starting point for ensuring protection from the most common and virulent threats, application misconfigurations that can lead to vulnerabilities, as well as detection tactics and remediations.

The OWASP Top 10 list for 2021 reflects a significant overhaul, debuting these brand-new categories: Insecure Design, Software and Data Integrity Failures and Server-Side Request Forgery. These point to an increasing focus on architectural vulnerabilities and going beyond surface-level bugs for the benchmark in software security.

Just like the adaptive threat landscape it seeks to define, this list is updated to continue to serve as an industry benchmark for the application security community. This piece provides an overview of the 2021 OWASP Top 10 list and technical capabilities security professionals should consider when evaluating WAFs.

TOP 10

Which OWASP Vulnerabilities Are the Most Popular with Cybercriminals?

Based on data from Radware's Quarterly DDoS and Application Threat Analysis Center, the most common OWASP Top 10 application vulnerabilities that cybercriminals exploit are Broken Access Control and Injection, the two of which typically comprise over half of all violations in any given quarter.

More specifically, predictable resource location attacks (which falls under A1: Broken Access Control) is witnessed nearly twice as often as the second-most-common violation – SQL Injection (which falls under Injection), according to the aforementioned online data. Predictable resource location attacks target hidden content and functionality of web applications. By guessing common names for file directories, an attack may be able to access resources unintended for exposure. Examples of resources that might be uncovered through Brute Force techniques include old backup and configuration files, web application resources yet to be published, and others.

2021 OWASP Top 10 Application Vulnerabilities

A1: Broken Access Control

A2: Cryptographic Failures

A3: Injection

A4: Insecure Design [NEW]

A5: Security Misconfiguration

A6: Vulnerable and Outdated Components

A7: Identification and Authentication Failures

A8: Software and Data Integrity Failures [NEW]

A9: Security Logging and Monitoring Failures

A10: Server-Side Request Forgery [NEW]

Common Web Application Attacks and Threats

SQL Injection

Cross-Site Scripting

Cross-Site Request Forgery

Server-Side Request Forgery

Protocol

Irregular Expressions

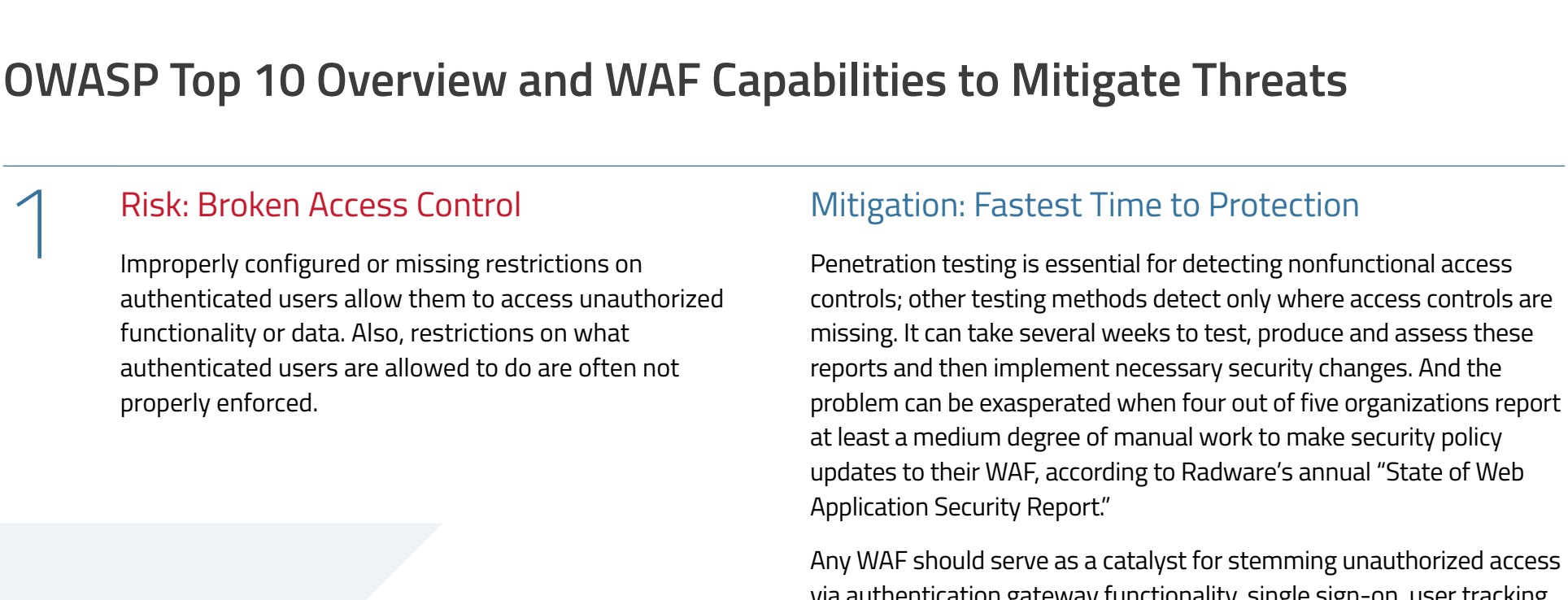
Denial of Service

Cookie Poisoning

Zero Day

Brute Force

Local File Inclusion and Remote File Inclusion



OWASP Top 10 Overview and WAF Capabilities to Mitigate Threats

1 Risk: Broken Access Control

Improperly configured or missing restrictions on authenticated users allow them to access unauthorized functionality or data. Also, restrictions on what authenticated users are allowed to do are often not properly enforced.

Mitigation: Fastest Time to Protection

Penetration testing is essential for detecting nonfunctional access controls; other testing methods detect only where access controls are missing. It can take several weeks to test, produce and assess these reports and then implement necessary security changes. And the problem can be exasperated when four out of five organizations report at least a medium degree of manual work to make security policy updates to their WAF, according to Radware's annual "State of Web Application Security Report."

Any WAF should serve as a catalyst for stemming unauthorized access via authentication gateway functionality, single sign-on, user tracking and access controls to the web application based on user role, profile information and security token validations.

2 Risk: Cryptographic Failures

Cryptographic failures (formerly known as Sensitive Data Exposure) focus on cryptography-related failures, which often lead to sensitive-data exposure or system compromise.

Many web applications and APIs contain vulnerabilities due to coding, thereby exposing sensitive data such as financial, healthcare and personally identifiable information. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.

Mitigation: Encryption

Encryption is key, both for data at rest or in transit. Leading WAFs provide inspection and encryption of data, including SSL inspection and protection capabilities to eliminate security blind spots. This includes, but is not limited to, SSL traffic decryption and encryption, masking server identities and veiling sensitive information. An adaptive WAF that leverages automatic policy generation and machine learning capabilities to automatically create and apply security configurations and policies is also critical. Finally, any enterprise-grade firewall should support the encryption of ingress and egress traffic across both on-premise and cloud-based infrastructures.

3 Risk: Injection

Injection flaws, such as SQL, NoSQL, OS and Lightweight Directory Access Protocol (LDAP) injection, have been a perennial favorite among hackers for some time, so it's no surprise that this threat is still at the top of the list. An injection flaw occurs when suspicious data is inserted into an application as a command or query. This hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

The most common code injection is a SQL injection, which is an attack that is accomplished by sending malformed code to the database server. It's a simple and quick attack type that almost anyone with internet access can accomplish, since SQL injection scripts are available for download and are easily acquirable.

Cross-site scripting (XSS) is now included as part of this category as well. XSS occurs whenever an application includes untrusted data in a new webpage without proper validation or updates an existing webpage with user-supplied data using a browser API that can create HTML or JavaScript. These flaws give attackers the capability to inject client-side scripts into the application to hijack user sessions, deface websites or redirect the user to malicious sites.

Mitigation: Positive Protection

Many web application security solutions leverage a negative security model, which defines what is disallowed while implicitly allowing everything else. Since attack signatures may generate false positives by detecting legitimate traffic as attack traffic, such rules tend to be simplistic, trying to detect the obvious attacks. The result is protection against the lowest common denominator.

A positive security model, which defines the set of allowed types and values, is required to provide proper protection where signature-based protection cannot fill the gap. In the case of a SQL injection, a positive security model screens user input for known patterns of attacks and leverages logic to tell the difference between legitimate user input and injection flaws.

Against XSS attempts, it's important to make sure any WAF can provide signature- and rule-based protection with updated signatures (similar to a blacklist), identify scripting patterns and block malicious requests.

4 Risk: Insecure Design

This category focuses on risks related to design flaws. This means using more threat modeling for secure design patterns and principles in the earlier stages of the application development cycle. It is a broad category representing many different weaknesses. According to OWASP, "Secure design is a culture and methodology that constantly evaluates threats and ensures that code is robustly designed and tested to prevent known attack methods. Secure design requires a secure development lifecycle, some form of secure design pattern or paved road component library or tooling, and threat modeling."

Mitigation: Secure Software Development

First, secure software development lifecycle (SDLC) methodologies must be adapted. Much of this revolves around the continuous integration and continuous development (CI/CD) pipeline. To that end, it's critical to ensure that any web application security solution provides a series of core capabilities that address security assessment at the early stages of application deployment, including API management to configure the web application firewall in such environments, automatic policy generation to automatically create new security policies based on any new application, and integration with dynamic analysis security testing (DAST) tools to create security policies based on a DAST security assessment.

In an unsecure SDLC environment, there is no one-size-fits-all capability to ensure application security. However, the following features can ensure a security policy is properly tailored to safeguard the application: security filters leveraging a positive/negative security model; IP-, geo-, and role-based policies; rate-limiting access to the server resources; use of a multilayered attack correlation detection engine.

5 Risk: Security Misconfiguration

Security misconfiguration remains one of the most commonly seen web application security issues to this day. This risk refers to improper implementation of controls intended to keep application data safe, such as insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and perhaps most important, not patching or upgrading systems, frameworks, libraries, applications and components.

This OWASP category now includes XML external entities as well. Many older or poorly configured XML processors evaluate external entity references within XML documents. Attackers can use external entities for attacks, including remote code execution, and to disclose internal files and Server Message Block (SMB) file shares, conduct internal port scanning and launch denial-of-service attacks.

Mitigation: Ability to Learn

As notable ransomware and malware outbreaks in recent years (for example, WannaCry) have proven, system upgrades are critical. An adaptive WAF will leverage automatic policy generation and machine learning capabilities to automatically create and apply security filters and enforcement rules where security is misconfigured. It will evaluate the structure of a web application, set relevant security filters and analyze traffic properties from a production environment to build a dynamic network profile, thereby maximizing security while minimizing false positives.

A WAF should be able to parse and inspect protocols and structured documents, including HTTP and HTTPS traffic, POST requests and XML JSON schemas. In addition, the aforementioned machine learning algorithms can learn XML and JSON structures and schemas for enforcement as part of the validation phase and create security policies.

6 Risk: Vulnerable and Outdated Components

Formerly referred to as Using Components with Known Vulnerabilities, various components, such as libraries, frameworks, and other software modules, that run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Developers frequently don't know which open source or third-party components are in their applications, making it difficult to update components when new vulnerabilities are discovered. These components can undermine application defenses and enable various attacks and impacts.

Mitigation: Knowledge of Where the Holes Exist

Any WAF that provides integration with programs such as Microsoft's Windows Server Update Services can protect against exploitations of vulnerable and outdated components by screening client requests and server responses. In addition, security updates and threat intelligence feeds are essential to keep security teams in the know and facilitate quicker responses to maximize protection and reduce exposure.

7 Risk: Identification and Authentication Failures (Formerly Broken Authentication)

When an application's functions are not implemented correctly, the door is left open for criminals to break in. Attackers can compromise passwords, keys or session tokens or exploit other implementation flaws to assume other users' identities temporarily or permanently. Sessions should be unique to individual users. Without some session management, an attacker can sneak in disguised as a user to access valuable data.

Mitigation: Challenge and Validate

Securing an application in terms of access control is no easy task. Authenticating users by having them provide their identity and challenging them to verify their identity is a key first step. Single sign-on and multifactor authentication are also key steps that reduce the risk of compromised accounts.

A key second step is to have a WAF that proactively encrypts session parameters between network and client, proactively inspects login attempts and thwarts HTTP sessions via code-encrypting, cryptographic capabilities.

8 Risk: Software and Data Integrity Failures

A new category for 2021, Software and Data Integrity Failures refers to code and infrastructure that fails to protect against integrity violations. This includes software updates, critical data and CI/CD pipelines that are implemented without verification. An example of this includes objects or data encoded or serialized into a structure that an attacker can modify. Another is an application that relies on plugins, libraries or modules from untrusted sources. Insecure CI/CD pipelines that can introduce the potential for unauthorized access, malicious code or system compromise also fit into this category. Lastly, applications with automatic update functionality – in which updates are downloaded without sufficient integrity verification and applied to a previously trusted application – are considered software and data integrity failures because attackers could infiltrate the supply chain to distribute their own malicious updates.

Insecure deserialization is now part of this category. Insecure deserialization often leads to remote code execution to tamper with or delete serialized objects or elevate privileges. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay or injection attacks and privilege escalation.

Mitigation: Best of Both Worlds

To mitigate insecure deserialization vulnerabilities, it's useful to identify WAFs that provide the best of both worlds, combining negative (defining what is forbidden and accepting the rest) and positive (defining what is allowed and rejecting the rest) security models. This winning combination should leverage various WAF access control filters such as cookie encryption, XML and JSON parsing, parameters enforcement and more.

9 Risk: Security Logging and Monitoring Failures

Security Logging and Monitoring Failures (formerly Insufficient Logging and Monitoring), coupled with missing or ineffective integration with incident response systems, is the bedrock for the majority of incidents, allowing attackers to run amok, attacking further systems and tampering with, extracting or destroying data. Many studies show that the time to detect is measured in weeks or months, typically detected by external parties rather than internal processes or monitoring. Typical attacks seeking to exploit these vulnerabilities can include SQL injections, XSS, cross-site request forgery, server-side request forgery, cookie poisoning, and Brute Force attacks.

Attackers rely on the lack of monitoring and timely response to achieve their goals without being detected. Most successful attacks start with vulnerability probing. Allowing such probes to continue can raise the likelihood of successful exploit to nearly 100%.

Mitigation: Suite Solutions Versus Best of Breed

To address the issue of internal processes, it's vital to think like an attacker and internally test and audit to discover if an organization has sufficient monitoring. If it lacks this "white hat hacker" expertise, the cybersecurity vendor chosen as a partner must provide distributed denial-of-service (DDoS) mitigation expertise via a team of security experts.

These same experts should also play a role in the second-biggest concern, which is real-time monitoring and detection. Timely detection of malicious malware or snooping hackers comes down to best-of-breed versus suite offerings. Stopping cyberattacks in near-real time is best accomplished through a single vendor attack mitigation system. Many organizations leverage best-of-breed mitigation tools from different vendors. This hodgepodge collection results in poor communication and detection. Suite WAF and DDoS solutions can more effectively communicate, setting network traffic baselines and comparing data points to quickly detect when something is awry, in addition to providing enterprise-grade monitoring and management dashboards and analytics.

10 Risk: Server-Side Request Forgery

Server-side request forgery (SSRF) occurs when a web application fetches a remote resource without validating the user-supplied URL. An attacker can coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN or other type of network access control list. Though SSRF shows a relatively low incidence rate in the data OWASP reviewed, this category was added based on industry survey results. Users are concerned that SSRF attacks are becoming more prevalent and potentially more severe due to the increased use of cloud services and the complexity of architectures.

Mitigation: Be Positive

A positive security model is critical to successfully mitigate the risks associated with the SSRF vulnerability. To that end, it's important to keep the following capabilities in mind when evaluating a WAF: API security protection, parameter filters, strong input validation, sensitive data exposure and signature creation related to direct file access - all of which require a WAF that can employ a positive security model.

The OWASP Top 10 is not intended as "one list to rule them all," but rather serves as a great starting point for application security programs and WAF evaluation. It serves as a benchmark for empowering improved people, processes and technology.

Successful organizations must establish and use repeatable processes and security controls, testers should establish continuous application security testing, application managers need to take charge of the application lifecycle and the organization needs to have an application security program in place that effectively coordinates across all facets of its infrastructure.

To that end, selecting the right WAF vendor to partner with is a critical step in executing these concepts. Be sure any WAF solution your organization is evaluating not only meets your organization's existing security needs but is flexible enough to adapt to future infrastructure environments, business needs and attack vectors.

Learn More About What Comprehensive Application Security For Any Environment Means ▸

Contact Us ▸