How to Prevent the OWASP Top 10

The Open Web Application Security Project, or OWASP, is a nonprofit that strives to educate the cybersecurity industry (its practitioners, researchers, and developers) about prominent web application bugs and the risks they present. Every three or four years, OWASP reaches out to the companies and organizations with a high-level and wide-sweeping view of the most common and highest risk vulnerabilities for feedback on common and emerging threats. These contributors include pen testing companies, bug bounty organizations, and vendors and consultants that do application testing and code reviews. OWASP collects the data anonymously and then ranks the vulnerabilities based on the level of risk they present and how urgently they believe that developers should be made aware of their threat.

While the OWASP Top 10 aren't the only security issues that impact businesses and developers, they make for a nice rundown of bugs and attacks that teams should keep top of mind as they build new technology or introduce new applications, services, and vendors into their businesses.

The last update from OWASP is from 2017, and though the Top 10 list may be updated later this year, we still believe that the current version is a good reference to guide the foundation and preventive security initiatives that you champion this year. So, to kick off the new year, let's dive into the 2017 OWASP Top 10 list and offer some guidance around how to prevent these bugs and types of attacks from owning you in 2020.

1. Injection

For an injection attack to happen (as defined by OWASP), untrusted data is sent to an interpreter as part of a command or a query. From there, the untrusted data can trick the interpreter into executing unintended commands or accessing unauthorized data.

Injection attacks have been around since nearly the dawn of hacking, but they remain effective. When successful, attackers can leverage injection to access all kinds of lucrative data damaging to businesses and individuals (like credit card information and SSNs). Injection can also enable attackers to control an entire system. The British ISP TalkTalk notoriously fell victim to a SQL injection attack in 2015, which affected approximately 150,000 users.

To Prevent: For client-side injection (e.g., XSS), our recommendation is contextual output encoding. But for server-side injection (e.g., SQLi), prepared statements is a reasonable solution. Essentially, it doesn't hurt to treat all user data as untrusted, either.
Related Reading: XXE Injection in PhpSpreadsheet

2. Broken Authentication

This vuln really tells you what it's about up front - it's a class of vulnerabilities that permit an attacker to "either capture or bypass the authentication methods that are used by a web application" because the authentication in place is broken.

It's often caused by improperly implemented authentication, which can lead to an attacker accessing otherwise restricted resources. This can result in the compromise of sensitive information, like password and keys. Additionally, an attacker could use this to leverage other flaws to assume users' identities.

To Prevent: Implement multi-factor authentication when possible, institute a strong password policy, and always change default credentials (especially for admins).
TL;DR – practice good security hygiene and correctly implement authentication.
Related Reading: SV3C L-Series HD Camera – Multiple Vulnerabilities

3. Sensitive Data Exposure

The term "sensitive data exposure" is maybe a more polite way of saying "breach." Many organizations come up short at doing their due diligence when it comes to protecting the sensitive data (SSNs, financial information, health records, etc.) of their users. And when that sensitive data leaks, that's when we see those noteworthy breaches.

To Prevent: Like the previous security risk, good security hygiene will help you here. For example, ensure that proper authentication and authorization are required to access sensitive data and regularly audit security standards you have in place.
Related Reading: Tegile Intelliflash OS Version 3.7.0.8.180413 (GA) - Password Disclosure

4. XML External Entity Attack (XXE)

XXE refers to an attack vector where an application that parses XML data is the target. When XXE is successful, user-supplied external entities are processed by the XML parser, which can lead to sensitive data exposure and/or server side-request forgery (SSRF).

To Prevent: Ensure that DTD is disabled in your XML parser.
Related Reading: PhpSpreadsheet Versions<=1.5.0 - XXE injection

5. Broken Access Control

The principle of least privilege is an oft-repeated adage in the security world – and for good reason. Users should only have the level of permission their role requires. When a flaw exists that prevents this from happening, users are able to gain unnecessary access and broken access control occurs.

To Prevent: Is it a cop-out to say that "proper access control" is the solution? Because it is. Implementing role-based access control and abiding by the principle of least privilege will aid tremendously. It may seem simple, yet it's something that so many organizations continually get wrong.
Related Reading: SolarWinds Log & Event Manager - Improper Access Control

6. Security Misconfiguration

Misconfiguration: a common, often overlooked threat for many businesses. Simply put, configuring anything in your environment incorrectly will put you at risk. And the more complex your environment happens to be, the more opportunities there are for misconfiguration.

To Prevent: Two useful preventative measures are to regularly audit your software and have a strong application architecture. Segmentation can help security misconfiguration from happening, but it’s by no means a panacea.
Related Reading: ConnectWise Control 19.3.25270.7185 - Eight Vulnerabilities, Including Critical

7. Cross-site Scripting (XSS)

XSS occurs when a web application is manipulated to include malicious client-side code to perform actions on behalf of end-users. There are further classifications of XSS attacks: persistent, reflected, and DOM being the main three.

Persistent XSS means attacker-controlled client-side code is stored on the target server - and as a result, this kind of XSS is contained in the application itself and therefore more dangerous than its counterparts. Reflected XSS is when malicious client-side code is “reflected” from a web application executing in a user’s browser; it is usually exploited by enticing users to click a malicious link. And DOM-based XSS happens when the XSS occurs in the document object model and is not apparently visible in the HTML.

To Prevent: Contextual output encoding is the proper fix for all XSS; output encoding prevents most XSS, but taking its environmental context into account prevents nearly all of it. Ensure that wherever user-supplied input is included into the application's interface contextual output, encoding is used. Perform regular security audits of your application to ensure your protections remain effective against XSS, and remediate as soon as an issue is flagged. (Spoiler alert! We will be covering XSS in depth in our next post.)
Related Reading: Big Monitoring Fabric Application

8. Insecure Deserialization

Insecure deserialization happens when the input of a serialization can be controlled by a user and the serialized object is then deserialized in an insecure fashion. At its worst, insecure deserialization can lead to remote code execution, which is almost always a high or critical severity security risk.

To Prevent: To mitigate your susceptibility to insecure deserialization, some of our advice will echo our advice to prevent injection vulnerabilities - don’t trust user input. Or, confirm that your serialization mechanism permits primitive data types alone.
Related Reading: OpenMRS - Insecure Object Deserialization

9. Using Components With Known Vulnerabilities

Insecure components are typically caused by not keeping up with patches or using legacy components in your environment that are affected by known security issues. While this sounds like more of a minor threat, in reality, known vulnerabilities that have gone overlooked in terms of remediation can lead to severe damage.

To Prevent: Keep an eye on the most recent CVEs as they emerge to stay informed of the latest bugs as soon as they are public knowledge. Patch as soon as needed, and if you can remove any known legacy software from your environment without impacting usability, do so.
Related Reading: How 'Small' Security Errors Lead to a Security Breach

10. Insufficient Logging and Monitoring

This isn't a class of vulnerability per se; it's more of an area for improvement - because slacking in this area can have some devastating results (e.g., the Dixons Carphone Breach.)

To Prevent: Adhere to logging and monitoring best practices as closely as possible. Establish processes for reviewing internally that allow logs to be managed efficiently and incidents to be responded to in a reasonable length of time.
Related Reading: There's not a great single resource on this, but we know from experience that if you don't have logging and monitoring in place, you don't have the visibility needed to even know you *were* attacked or breached. These are some of the foundational security measures that businesses need to square away to be confident that their attack surface is secured.

Following basic security hygiene tips and thoughtfully considering how to configure tools and how to handle access control can block many of the most common threats to your environment. Attackers usually go for the tried-and-true methods of attack that are the lowest effort and the easiest way into a business. From that foothold, they'll pivot into new areas with more sensitive data or employ methods of gaining additional access to that sensitive data. Every rough edge you can smooth out will dissuade attackers from continuing to pursue your assets and will further establish your environment as secure.

While you won't stop every emerging threat or zero-day with these basics, you'd be surprised at how often they work to prevent the types of attacks that we see daily in our assessments, including those highlighted in the OWASP Top 10.