www.phparch.com

October 2021
Volume 20 - Issue 10

php[architect]

Decrypting Cryptography

Cryptography 101

What I Wish Someone Told Me About SQL Database Design

ALSO INSIDE

The Workshop:
Local Development with CraftCMS Nitro

Community Corner:
PHPUnit Creator Sebastian Bergmann Pt2

Education Station:
Windows 11 for PHP Development

PHP Puzzles:
Time Value of Money

Security Corner:
Updating the OWASP Top Ten

Design Patterns by Moonlight:
When There Be Dragons

finally{}:
Rubber Ducky, You're the One!

Updating the OWASP Top Ten

Eric Mann

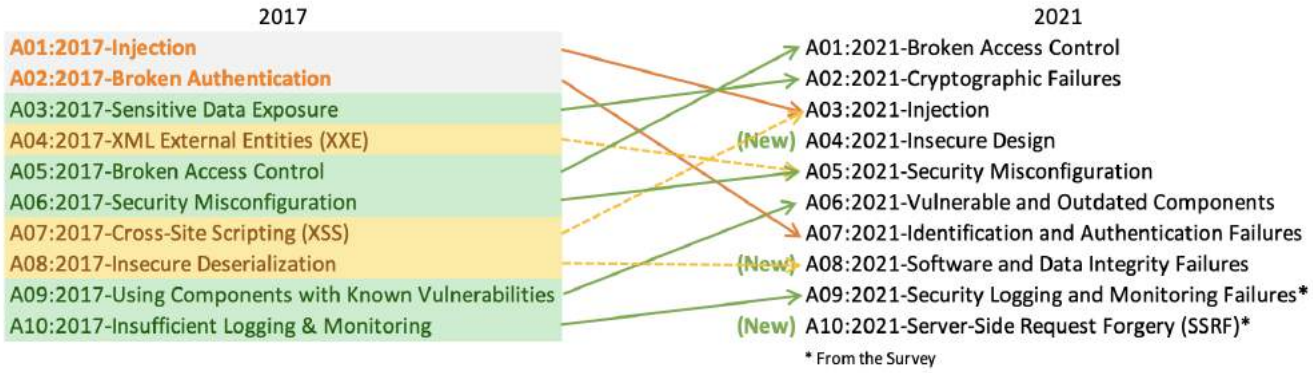
The Open Web Application Security Project (OWASP) is a non-profit that focuses on web security research, training, and documentation to help developers make the world a safer place. They regularly collate application security risks seen in the wild and publish a list of the most frequently encountered issues. This list, the OWASP Top Ten, is a standard tool used by developers and security auditors alike to gauge the level of security maturity of a project or the team maintaining it.

OWASP solicits feedback from the development community and updates its list of the top 10 application security risks (ASRs) roughly every three years. The last edition was published in 2017, and the organization has now published a draft of its latest update, OWASP Top 10 team 2021[1], as shown in Figure 1. In this latest version, few items from 2017 remain the same. OWASP has added three new risks, re-scoped existing risks to cover a broader definition of the risk they represent and reprioritized the existing ones based on market data regarding the frequency with which each risk is seen in the wild.

For context and background, the previous edition of the OWASP Top Ten defined the following application security risks in order of frequency:

> *Remember, the order of the OWASP Top Ten is defined by how prevalent the risk has been in the software community. It is not a representation of the severity of each risk. Cross-site request forgery, for example, is an incredibly severe risk to many applications but was seen rarely enough in 2017 that it fell off that edition of the list entirely.*

Figure 1. Draft Edition 2021



- ASR1—Injection
- ASR2—Broken Authentication
- ASR3—Sensitive Data Exposure
- ASR4—XML External Entities (XEE)
- ASR5—Broken Access Control
- ASR6—Security Misconfiguration
- ASR7—Cross-Site Scripting (XSS)
- ASR8—Insecure Deserialization
- ASR9—Using Components with Known Vulnerabilities
- ASR10—Insufficient Logging and Monitoring

Revised Priorities

In the list's latest draft edition, *Broken Access Control* has been elevated to first place. This isn't a reflection of a change in the impact of the risk but does reflect that this risk has been seen more frequently in the wild—either through internal audits or through threat actors testing systems tested for broken access control. The scope of its documentation hasn't really changed, either, just the prioritization placed on its awareness by OWASP.

"Sensitive Data Exposure" has also moved up on the list to second place, but with a tightened scope of "Cryptographic

1 *OWASP Top 10 team 2021: https://owasp.org/Top10/*

Failures." This move addresses the root cause of engineering mistakes that lead to sensitive data exposure. A solid example is a high potential for cryptographic misconfigurations in primitive data structures like JSON web tokens like we discussed last month[2].

As of 2021, "Injection" moves down the list by one spot to third place. Its scope has also expanded to include cross-site scripting, which used to be a risk category in its own right but is a type of injection.

The previous list's "XML External Entities" risk has been merged into the *Security Misconfiguration* category, and together, they rise to fifth place in the current list.

"Using Components with Known Vulnerabilities" has repeatedly been a highly prioritized risk in the community but has been relatively low in the OWASP list. The latest edition rebrands this risk category as "Vulnerable and Outdated Components," also shifting it up from ninth to sixth place in the opinion of OWASP contributors.

> *PHP developers can avoid using outdated components by leveraging tools like Composer to manage their dependencies and keeping those libraries up-to-date. Hosted services like GitHub's Dependabot[3] or Snyk.io[4] can scan for and sometimes update outdated components automatically. Developers can also leverage the Local PHP Security Checker[5] to proactively scan their projects for outdated or insecure dependencies during development.*

Similarly, "Broken Authentication" has been renamed "Identification and Authentication Failures" and shifts quite a way down the list to seventh place. While authentication issues are still an ever-present risk to web applications, recent advances in the standardization of web frameworks have made it easier to set secure defaults and "accidentally" do things the right way from the start. This is still a risk that application engineers need to worry about, but it's becoming harder to make mistakes as "the right thing" is made an easier-to-implement standard.

Finally, "Insufficient Logging and Monitoring" have both moved up the list to eighth place and been redefined as "Security Logging and Monitoring Failures." This risk category has been expanded to include additional types of failures and represents risks that aren't well-reflected in existing CVE (Common Vulnerabilities and Exposures) data.

New Risks

There are three entirely new additions to the latest version of the list. Some are due to the ongoing evolution of both web development—namely, that we continue to focus on a "shift left" approach to security responsibility, further making security everyone's job.

> *"Shifting left" is the paradigm where downstream responsibilities for software are progressively shifted upstream towards the development team. The first occurrence of this that many of us are probably familiar with is adding operation responsibilities to development teams who previously relied on separate parties to run their software in production. Today, we call this a DevOps philosophy. The newest occurrence is shifting security responsibilities gradually from a dedicated team into development—we call this DevSecOps.*

Insecure Design

Security is, in fact, the responsibility of the entire engineering team regardless of whether or not their title and job description formally reflect that fact. You are responsible for the security of your own code and, increasingly, for understanding the long-term security impacts of any design decision you make. Secure system architecture does not need to be difficult, but it's too often an afterthought for many development teams. The fact that security in system design can be an afterthought is what slates this new ASR as the fourth item on the list.

Software and Data Integrity Failures

While this is a new category in its own right, it includes the Insecure Deserialization risks from the last edition. It's too easy to make assumptions about the nature of the data—or dynamic code—flowing into or out of a system and fail to properly validate or verify that data's integrity. These failures lead to security weaknesses in our applications and present risks to the stability and security of our web applications. The SolarWinds supply chain attack from 2020[6] is a crucial example of software integrity failure in practice.

Server-Side Request Forgery (SSRF)

Whereas cross-site request forgery attempts to trick a client browser into making malicious requests, server-side request forgery is an attack against a hosted application itself. This might take the form of tricking an application to make requests to other sensitive systems within an otherwise protected network, thus giving an attacker the ability to read or even update resources that would otherwise be protected from public view. It's a relatively rare security risk when compared to the frequency of the other categories. Still, it appears often enough in the wild that every engineer hosting an application should be aware of it.

2 *last month:* https://phpa.me/security-pit-of-success

3 *GitHub's Dependabot:* https://phpa.me/github-dependabot-security

4 *Snyk.io:* https://phpa.me/snyk-composer-vulnerabilities

5 *Security Checker:* https://phpa.me/github-security-checker

6 *attack from 2020:* https://phpa.me/security-supply-chain

Things to Keep in Mind

The OWASP Top Ten is a valuable guide for any developer; security is everyone's responsibility, regardless of whether it's explicitly listed in your job description. Every developer should be intimately familiar with the risks enumerated in this list to ensure their applications and deployments avoid these common pitfalls.

Still, remember this is but one list of risks your application might face, and it is in no way exhaustive[7]. Any security expert can name offhand several risks you and your team might encounter that are not included in this list. This isn't because they're less of a threat—they just haven't been seen frequently enough in the wild to be considered among the top ten. The relative rarity of an application security risk does not afford you any protection whatsoever.

You owe it to yourself and your customers to familiarize yourself with the OWASP Top Ten and fully understand how to avoid each of the risks it categorizes. You also owe it to yourself to think beyond the list about other risks your team, your application, or your customers might face in practice.

Related Reading

- *Security Corner: The Risk of Lists* by Eric Mann, April 2019. https://phpa.me/security-apr-19
- *Security Corner: Updates to the OWASP Top Ten—Logging* by Eric Mann, January 2018. https://phpa.me/security-corner-jan-2018
- *PHP and Database Access* by Erwin Earley, August 2020. https://phpa.me/2020-08-php-db

Eric is a seasoned web developer experienced with multiple languages and platforms. He's been working with PHP for more than a decade and focuses his time on helping developers get started and learn new skills with their tech of choice. You can reach out to him directly via Twitter: @EricMann

7 *it is in no way exhaustive:* https://phpa.me/security-risk-of-lists