The State of Security of Laravel Apps in 2021

A whitepaper detailing self-reported security from Laravel programmers during the second half of 2021

LARAVEL HACKER

laravelhacker.com

Table of Contents

The State of Security of Laravel Apps in 2021

There are many elements that contribute to the security of a web application. These range from the programming techniques you use to the authentication and authorization mechanisms, whether your server and vendor software packages are up to date, rate limiting and web application firewalls, and proactive error and performance reporting.

Your Laravel application is no exception. So, in 2021, Laravel Hacker launched a 7 Question Quiz to help programmers measure the security of their application. The quiz answers were gathered anonymously and aggregated using Cloudflare workers. This means Laravel Hacker was able to gather location information and answers, but not personal-identifiable information. (And you know I wouldn't have it any other way!). This report shares the findings with the community.

This is the State of Laravel App Security in 2021.

How was data captured?

Visitors of LaravelHacker.com directly - or those who landed on /quiz were offered a quiz to take. These questions were a mix of multiple choice and single answers. Multiple choices built a stronger score based on the items that were checked. Single questions had a single score for each answer. The scoring mechanism is a proprietary system developed by Laravel Hacker based on over two decades of programming and security experience.

Data was captured by a Cloudflare worker which was augmented with CDN edge location information. No client-browser information such as user agents, cookies, or other identifying information was gathered. Information was then deposited into a Google Form. This information was later processed offline with business analytics tools. You can find more details at the Laravel Hacker privacy policy.

What can I expect to learn from this report?

Part of the point of this report is to be open and honest with the community. We need to know as a developer community how secure our applications are in aggregate. I believe this will help us understand if we need to create more security-based training and products to help continue to support and bolster the Laravel framework as a framework of choice for applications.

Besides aggregate information, I'm hoping this will help inspire each reader to investigate and improve their own app security, too.

With that, let's get on to the data.

Quiz Answer Demographics

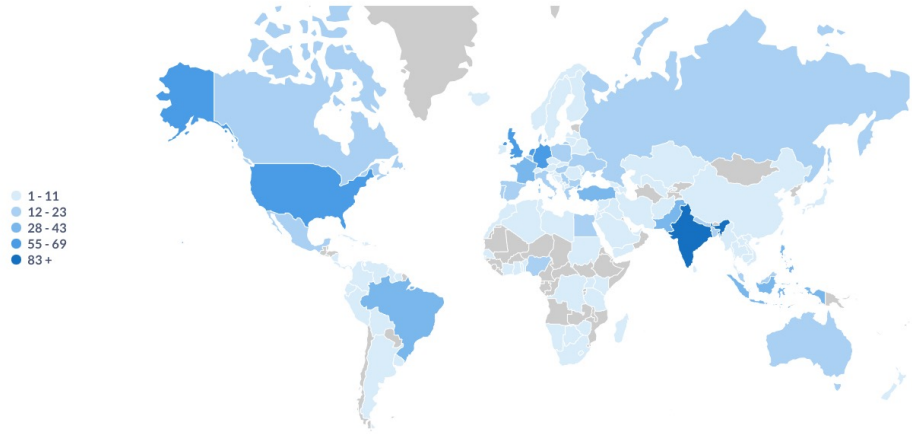
Let's talk about how many quiz submissions we received and where they're from.

The 7 Question Laravel App Security quiz was launched in August of 2021. The information was aggregated Dec 20th.

How many?

Submission count	Average submissions
1,180 total	18 per day

Where do the submissions come from?

Countries


Top Countries		Country Distribution
IN	83	122 unique countries
NL	69	
US	68	
GB	61	
DE	55	

So, we have a nice mix of submissions! Let's find out more about the actual answers now.

Quiz Answers Breakdown

Let's check out the details per question now.

Remember, these are from a total of 1180 responses. Percentages are calculated independently and rounded, so some totals may not meet - or may exceed - 100%.

1: Monitoring / Visibility

This first question focused on performance, uptime and error monitoring.

Question	# Yes / Agree	Percent
Have PHP error reporting tool in production	539	46%
Have JS error reporting tool in production	213	18%
Uptime tool reporting on homepage	289	24%
Uptime tool reporting mission-critical endpoints	114	10%
Review performance locally	331	28%
Review performance against production	108	9%

2: Code Management

Question 2 was about code management and process.

Question	# Yes / Agree	Percent
Project in source control	1010	86%
Code review process	480	41%
Signed off by another programmer	326	28%
Uses security and standard scanning tools	222	19%

3: Config and Environment

This question was a single answer ranging from naive to best practice. Where are your secrets and environment-specific configuration stored?

Answer	Chose	Percent
All are in a local .env file	902	76%
Mix between hard-coded Laravel config and env	142	12%
Hard-coded values in controllers and services	33	3%
Did not answer	103	

At first, I thought the amount of non-answers might mean this question was confusing. Let's see if there's a trend as we go on.

4: Rate Limiting

Question 4 was another sliding scale addressing best practices. Do you use rate limiting - and if so - beyond the authentication kit version?

Answer	Chose	Percent
No rate limiting	384	33%
Rate limiting as part of authentication kit	354	33%
Rate limit authentication and other crucial/expensive endpoints	295	25%
Did not answer	147	

I'm not sure that all developers know that the authentication kits contain built-in rate limiting.

5: Validation

Question 5 involved user input validation.

Question	# Yes / Agree	Percent
Specify required and nullable for all fields	821	70%
Validate types like string, integer	748	63%
Validate bounds	585	50%
Prohibit unwanted data	441	37%
Only retrieve validated data from validator	643	55%

6: Authorization

This question gave a range of options to indicate how authorization was applied.

Answer	Chose	Percent
Security through obscurity and some checks	129	11%
Check IDs in controllers	218	19%
Mix of gates and policies on important stuff	430	36%
All controller methods gated, all models have a policy	238	20%
Did not answer	165	

As the questions progressed, there are more did not answer results.

7: Database Security

The last question was about database security.

Question	# Yes / Agree	Percent
Always use Eloquent or the Query Builder	965	82%

Question	# Yes / Agree	Percent
Rarely raw queries, but never user input if so	512	43%
DB user has least access privileges as necessary	543	46%

Let's move on to some analysis.

Analysis and Findings

What can we learn from the 7 Question Secure Laravel quiz? About the answers / submissions? About inquiring about security configuration?

What the results tell us

With over 120 unique countries and principalities, the reach of Laravel programmers is global. The desire to secure an app transcends borders - it's something we all should - and do - care about. I think it's a good sign that there are so many people interested in securing Laravel apps world-wide.

There needs to be more proactive error monitoring in the Laravel app ecosystem. I think sometimes programmers confuse the security and durability of the framework with their own app. We all make mistakes - there's nothing wrong with that. There is a problem with ignoring them, or being ignorant of them, however. Only 46% of respondents monitor their PHP code for errors in real time. Less than 20% monitor their Javascript. These monitoring systems are your first line of defense in securing your app. If you don't know how someone's breaking your app unsuccessfully, you'll have no idea when they attack you successfully.

I'd like to see more code reviews and sign offs by other programmers. I realize that a lot of developers may be single person shops, but we can do better. Find a trusted colleague and partner with them just for code reviews. Have them be your second set of eyes - and be that for them.

Most developers are using local environment variables to generate their configuration. That is the Laravel paradigm, but sometimes this can be a hard thing to stick with compared to simple version control config. This is a great thing. Still, though, we have about 3% of respondents who are hard-coding their values into controllers and services. Don't forget, you can dependency inject values into constructors, too! Don't hard-code that API end point URL.

More than half of developers are getting validated data directly from the validator and not from the request. Good job! Getting it directly from the request is a recipe

for disaster. You never know when you'll quickly add a new field and promise to validate it later - and that's another hole in the application.

The community could do some work on authorization. There's a pretty big mix between security through obscurity, checking IDs in controllers, and mixing in policies and gates. In a perfect world, all things would be gated and authorized - even when there are guest users. You never know when you need to suddenly lock something down. You can get there, though!

I was pretty happy to see the number of people with limited database user access and those who profess to use Eloquent and Query Builder only. SQL injection attacks should be a thing of the past by now.

Because Laravel gives us a lot of tools out of the box, we can make secure applications. I'm encouraged to see programmers are finally starting to take advantage of these things.

What could have been better with the quiz

Time for some self-reflection. I've noticed a couple of things I could have done better. Sadly, these mistakes could have made my data gathering less accurate.

Require answers No quiz answer was ever required. This was fine for multiple-choice answers, but not so much for single answer questions. I wanted to reduce friction (and to be perfectly transparent, get them to the end with an option to sign up for a free email course). I'm not sure if some people literally had no answer or just wanted to finish the quiz for some reason.

Clearer questions After rereading the questions months later, I can see where some might not have been as clear as I intended. I think my questions that built upon each other (mainly multiple choice ones) were fine for the most part. But the single answer questions sometimes didn't make it clear that they were levels of accurate. I'd have changed up some of these questions.

Ask the Laravel version and size of team I think knowing if the person was on a newer or legacy version of Laravel, as well as if they have multiple projects, or they're on a team could have helped put some of these answers into a better

context. I still think it's possible to get code review when you work alone (in fact, I do this every week), but I could understand single member teams forgoing this easier than multi-member teams.

So how did the community score?

I broke down the scoring mechanism to three sections:

Score	Description
0-7	Emergency! Your app needs help!
7-17	Not bad - but we have some work to do.
18-21	Good job! Ready to get into advanced security.

The average score from 1180 responses:

Ah - not the worst but not the greatest. The community has some work to do. But, there's a light at the end of the tunnel...

How to become Beautifully Secure

Practicing proper application security doesn't mean harming the user experience or destroying your programmer environment. We can enjoy the beauty and ease of the Laravel framework while implementing the most secure best practices.

How do I get started?

If you haven't taken the quiz yourself, you should do it now. After you get your security score, you'll get the option to sign up for the Laravel Hacker 7 Day Secure App email course. Each day is a new email that focuses on the next question. You'll learn why the question is important and what you can do to get a perfect score next time.

Check out the quiz now at https://laravelhacker.com/quiz.

What if I need help now?

To reach out, visit the Work with Me link on Laravel Hacker.

End Notes

Thanks for reading this white paper on the state of Laravel app security in 2021. I appreciate your time and attention. I can't wait to see the progress that both you and the community make adopting more secure programming practices in the following months and years!

Aaron Saray
Laravel Hacker
LaravelHacker.com