

VOLUME 11

State of Software Security

Flaw Frequency
by Language

VERACODE

Volume 11 of Veracode's *State of Software Security* report is forward-thinking, digging into the trajectory of software security trends and backed by data from over:

130,000

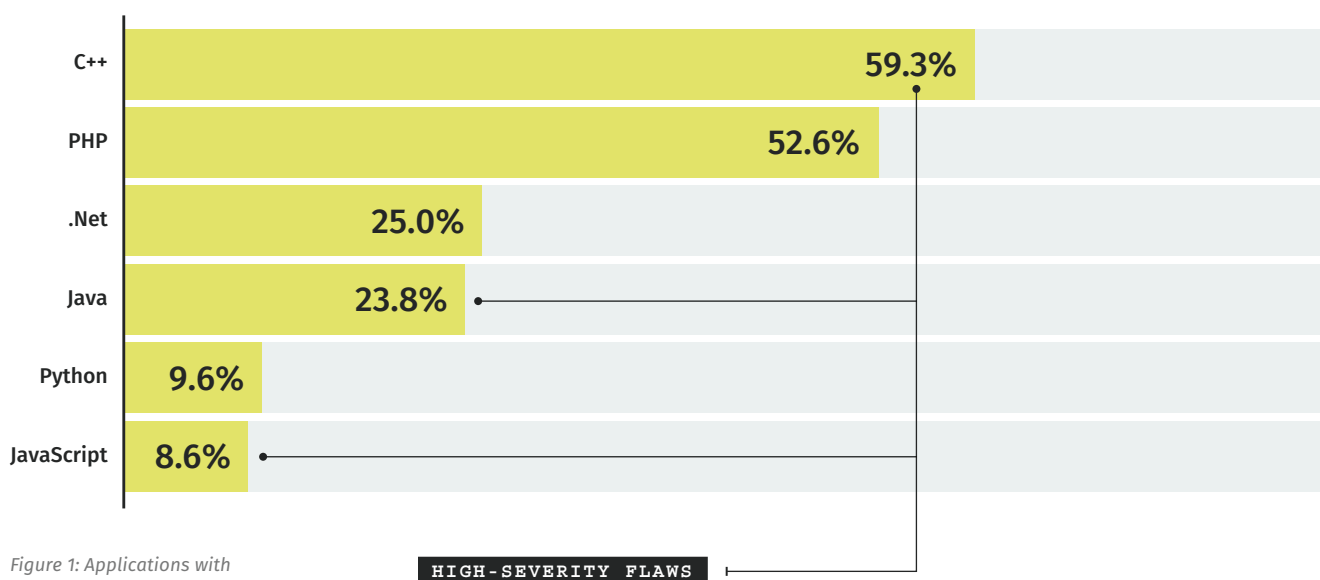
APPLICATION SCANS

These trends not only provide a window into the risks that development and security teams face regularly, but also shed light on the riskiest vulnerabilities and languages that developers should keep an eye on while they work.

Today, it's critical that developers are able to write code faster and more efficiently in order to keep up with the demands of modern software development, and though it's easier than ever to find and remediate flaws with the right application security (AppSec) tools, development teams still face language-specific challenges that can turn into roadblocks. By examining flaw frequency trends in various common languages, developers have a better understanding of the everyday risks they face while coding and can use that knowledge to get ahead of those flaws before they become a problem.

In Volume 10 of *State of Software Security*, one of our focuses was on flaw prevalence among the most common languages. We found that some languages are more susceptible to certain types of flaws than others, which is consistent with what we know about different platforms. Buffer overflow and buffer management errors are common issues in C++, but the built-in buffer management capabilities of higher level languages (like .NET and JavaScript) mean those flaws tend to be rare in those applications.

In this iteration of data from *State of Software Security* Volume 11, we see that the distinctions across languages still hold true: 59 percent of C++ applications have high (and very high) severity flaws, compared to just 9 percent of JavaScript applications. The script is a bit flipped around with Java, as only 24 percent had critical flaws this time around (Volume 10 looked for at least one flaw, so it's not a straight comparison with the below figure, which focuses on just critical flaws).



59 percent of C++ applications have high (and very high) severity flaws, compared to just 9 percent of JavaScript applications. It's flipped with Java, as only 24 percent had critical flaws this time around.

The interesting thing about the language breakdown was the fact that the most common flaw type was different for each language.

The most common flaw type in .NET applications was information leakage, while it was Cross-Site Scripting (XSS) for PHP, and CRLF injection for Java applications. The first intersection comes in the second most common flaw, with code quality appearing for both .NET and Java applications. In fact, there are several points of overlap for .NET and Java applications, which makes sense with the similarities between the platforms.

While the language breakdown is useful, there is a significant risk with this kind of analysis as it can artificially elevate certain flaw types. Cross-Site Scripting is also the most common flaw in JavaScript applications, but that applies to less than a third of applications scanned. So it is a little problematic to put it on the same level of severity as PHP, where XSS is found in three-quarters of the scanned applications.

THE HEAT MAP PROVIDES SOME INDICATION OF THE FREQUENCY OF FLAWS:

	.Net	C++	Java	JavaScript	PHP	Python
1	Information Leakage 62.8%	Error Handling 66.5%	CRLF Injection 64.4%	Cross-Site Scripting (XSS) 31.5%	Cross-Site Scripting (XSS) 74.6%	Cryptographic Issues 35.0%
2	Code Quality 53.6%	Buffer Management Errors 46.8%	Code Quality 54.3%	Credentials Management 29.6%	Cryptographic Issues 71.6%	Cross-Site Scripting (XSS) 22.2%
3	Insufficient Input Validation 48.8%	Numeric Errors 45.8%	Information Leakage 51.9%	CRLF Injection 28.4%	Directory Traversal 64.6%	Directory Traversal 20.6%
4	Cryptographic Issues 45.9%	Directory Traversal 41.9%	Cryptographic Issues 43.3%	Insufficient Input Validation 25.7%	Information Leakage 63.3%	CRLF Injection 16.4%
5	Directory Traversal 35.4%	Cryptographic Issues 40.2%	Directory Traversal 30.4%	Information Leakage 22.7%	Untrusted Initialization 61.7%	Insufficient Input Validation 8.3%
6	CRLF Injection 25.3%	Code Quality 36.6%	Credentials Management 26.5%	Cryptographic Issues 20.9%	Code Injection 48.0%	Information Leakage 8.3%
7	Cross-Site Scripting (XSS) 24.0%	Buffer Overflow 35.3%	Cross-Site Scripting (XSS) 25.2%	Authentication Issues 14.9%	Encapsulation 48.0%	Server Configuration 8.1%
8	Credentials Management 19.9%	Race Conditions 30.2%	Insufficient Input Validation 25.2%	Directory Traversal 11.5%	Command or Argument Injection 45.4%	Credentials Management 7.2%
9	SQL Injection 12.7%	Potential Backdoor 25.0%	Encapsulation 18.1%	Code Quality 7.6%	Credentials Management 44.3%	Dangerous Functions 6.9%
10	Encapsulation 12.4%	Untrusted Initialization 22.4%	API Abuse 16.2%	Authorization Issues 4.0%	Code Quality 40.3%	Authorization Issues 6.8%

Figure 2: CWE flaw types in applications in various languages

The frequency of flaw types varies by language, but there is still significant overlap in the flaws that showed up across different languages.

The worm map below helps highlight the prevalence of specific flaws by language. Information leakage was the highest for .NET and PHP applications, but not so much a problem for Python. As mentioned earlier, buffer management errors are big issues only for C++ applications, but not many web applications are written in C++, so web-related flaws such as XSS, are clearly not as common.

The similarity between scripting languages JavaScript and Python is evident in this chart. Untrusted initialization flaws are pretty negligible for most applications but are very common in PHP.

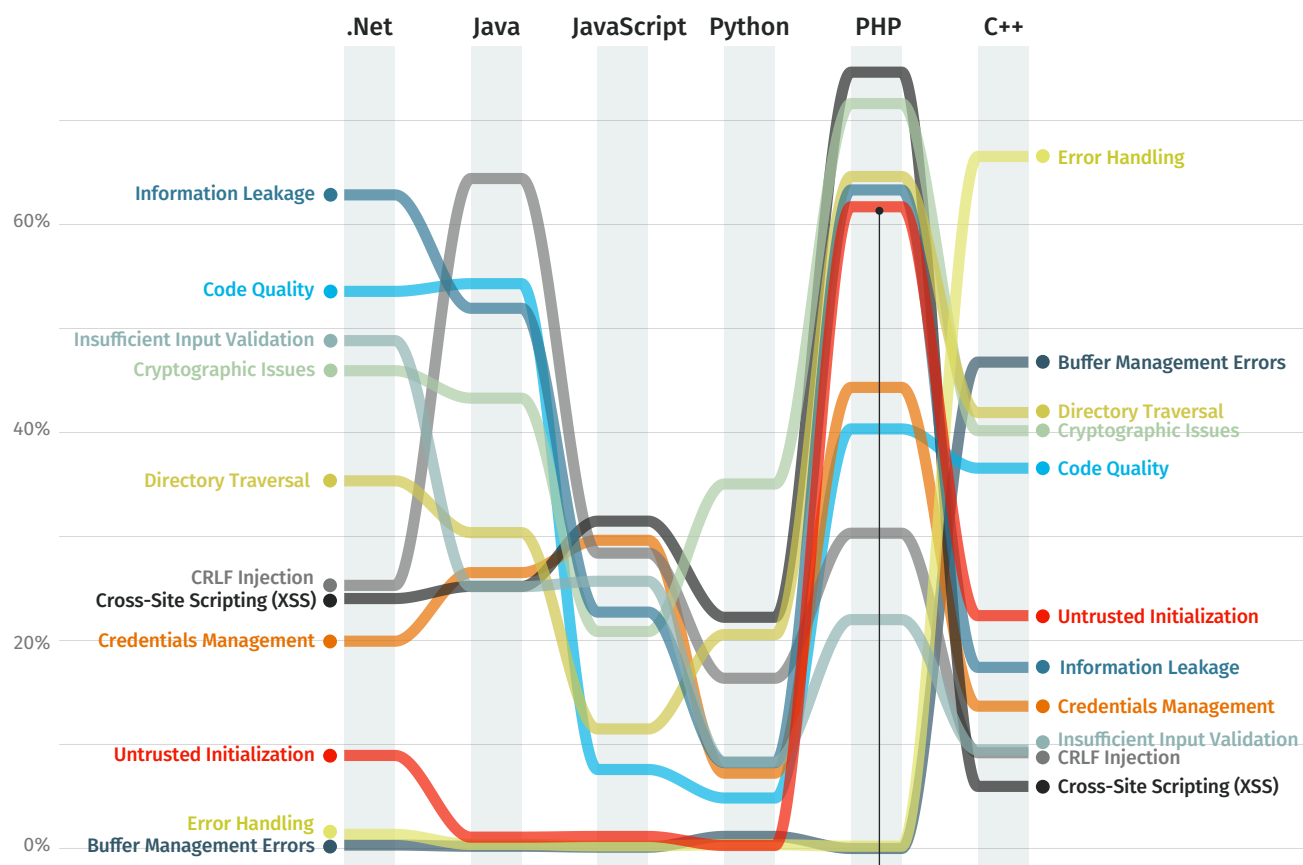



Figure 3: CWE flaw types in applications in various languages

JAVASCRIPT + PYTHON

The similarity between scripting languages JavaScript and Python is evident in this chart.

UNTRUSTED INITIALIZATION

Untrusted initialization flaws are pretty negligible for most applications but are very common in PHP — something for developers to keep an eye on.



It's unrealistic to expect that developers will write perfect code every time they work on an application, but it's critical that they're enabled to find and fix flaws on a schedule that won't create more of a bottleneck.

Implementing secure coding practices and increasing developer know-how for flaws by language can help ensure that the security of your applications (and your sensitive data) is where it needs to be to keep up with modern software development.

For more insight into the trends and challenges in secure application development, download the full *State of Software Security Report*.

READ THE FULL REPORT AT [VERACODE.COM/SOSS](https://veracode.com/sooss)

VERACODE

Veracode is the leading AppSec partner for creating secure software, reducing the risk of security breach and increasing security and development teams' productivity. As a result, companies using Veracode can move their business, and the world, forward. With its combination of automation, integrations, process, and speed, Veracode helps companies get accurate and reliable results to focus their efforts on fixing, not just finding, potential vulnerabilities. Veracode serves more than 2,500 customers worldwide across a wide range of industries. The Veracode cloud platform has assessed more than 14 trillion lines of code and helped companies fix more than 46 million security flaws.

veracode.com [Veracode Blog](#) [Twitter](#)