DZone

THE 2018 DZONE GUIDE TO

Security

DEFENDING YOUR CODE

VOLUME IV

RESEARCH PARTNER SPOTLIGHT

edgescan™

Key Research Findings

BY JORDAN BAKER - CONTENT COORDINATOR, DZONE

DEMOGRAPHICS

For this year's DZone Security Survey, we received 683 responses with a 62% completion percentage. Here's how some of the demographics for our respondents break down:

- 42% work for companies that are headquartered in the USA, 18% work for companies in South Central Asia, and 13% work for companies in Europe.
- 21% work for software vendors, 18% in finance/banking, and 10% in e-commerce.
- 24% work for organizations sized 10,000+ employees, 23% for organizations sized 100-999 employees, and 20% for organizations with 1,000-9,999 employees.
- 35% work as developers/engineers, 25% as developer team leads, and 17% as architects.
- 84% develop web applications/services (SaaS), 48% develop enterprise business applications, and 28% develop native mobile applications.
- 84% work for companies that use the Java ecosystem, 67% use the JavaScript (client-side) ecosystem, and 37% use the Node.js (server-side) ecosystem.

DEVELOPERS AS SECURITY PROFESSIONALS

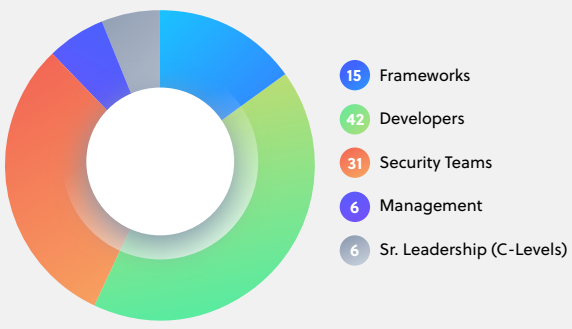
Over the last several years, we've witnessed a large push among the developer community for security to shift left in the SDLC. The statistics from this year's DZone Security Survey show the effectiveness of this trend. When asked who should take primary responsibility for security, 42% of respondents said developers, 31% said security teams, and 15% said the frameworks themselves. Of the respondents who answered this question, 35% currently work as developers/engineers. Of those 35% currently employed in developer roles, almost half (41%) told us they believe developers should be primarily responsible for security. Additionally, 53% of developer team leads reported that developers should be primarily responsible for security. These are both promising trends in the field of application security.

A core part of the shift left movement in application security is not only increasing concern for security among developers, but also providing developers with the necessary training and resources to learn secure coding practices. Taking a historical look at security training data, we can see some positive signs. In the 2017 and 2018 DZone Security Surveys, we asked how frequently developers at our respondents' organizations received security training. Here's how their answers broke down:

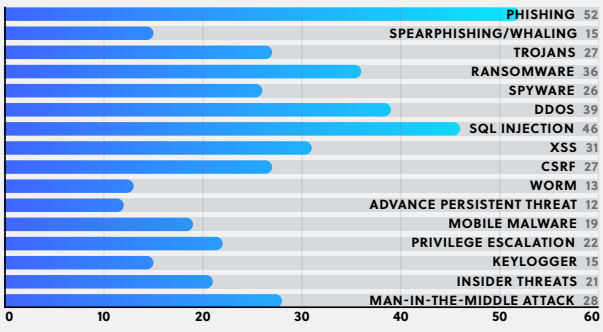
- Ad-hoc:
  - 2017: 37%
  - 2018: 33%
- Never:
  - 2017: 27%
  - 2018: 25%
- Yearly:
  - 2017: 16%
  - 2018: 15%
- Semi-annually:
  - 2017: 12%
  - 2018: 13%
- Quarterly:
  - 2017: 9%
  - 2018: 15%

The increase in quarterly security training proved quite substantial and is

WHO SHOULD BE PRIMARILY RESPONSIBLE FOR SECURITY?



WHICH OF THESE TRAITS HAVE YOU BEEN MOST CONCERNED ABOUT IN YOUR ORGANIZATION IN THE PAST YEAR?



inversely proportional to the percentage of developers reporting that they receive security training on an ad-hoc basis or no training at all. While ad-hoc or no security training remain the two largest categories reported by our respondents, the decrease in their instances over a year, and the marked jump in quarterly trainings, is a positive sign for the industry.

While security has certainly become a greater concern for developers in recent years, it continues to be outweighed by performance concerns. 37% of respondents said that their organization views performance as the largest priority, while 31% reported that security is their organization's most important concern. In addition to performance, releasing software on schedule often overrides security issues. Approximately half of this year's respondents (51%) reported allowing release schedules to interfere with security concerns on an at least semi-regular basis. To take a more granular look, 34% said release schedules "sometimes" interfere with security, 11% reported "often," and 6% reported it happens "all the time." In fact, only 10% of this year's survey takers told us that releasing on schedule never overrides security concerns; and an additional 25% said it rarely happens.

So, what happens when a vulnerability slips through the cracks? 83% of respondents told us they inform customers of potential known vulnerabilities that got shipped in their software. While we'd all like to see this number at 100%, it's still a marked increase from 2017, when only 67% of respondents reported informing customers of potential vulnerabilities in their solutions.

APPSEC

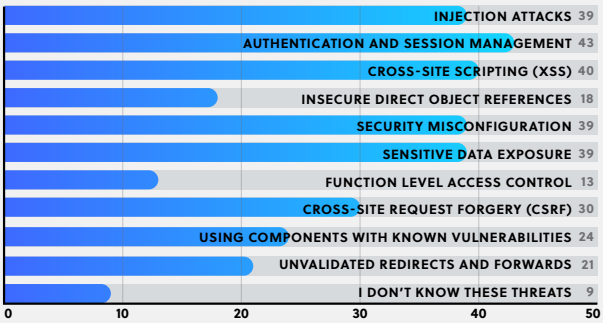
Now that we've established that security is becoming a greater concern among developers, let's explore how developers are approaching application security. When we asked at what stage of the SDLC respondents spend the most time implementing security measures, 33% said design, 28% said implementation, 14% said testing, and 13% said requirements gathering and analysis. The fact that a third of respondents implement security in their code in the design phase provides another indication that security is shifting left in the SDLC. This also means, however, that more effort is being put into securing new code than legacy code. In fact, 31% of respondents told us they spend more time per week securing new code, whereas 20% told us they spend more time securing legacy code. Another 31% reported spending equal amounts of time securing legacy and new code, but even with that demographic included, the security scales still tip toward new code.

When it comes to secure coding techniques, validating inputs proved the most popular option among respondents, with 74% of survey takers reporting to use this tactic. In 2017, however, 83% of our Security Survey respondents told us they validated inputs. Architecting and designing for security policies is a technique that seems to be on the rise, however. In 2017, 59% of respondents reported using this approach; this year, that number rose to 64%. Given the increased popularity of architecting and designing for security policies, let's quickly examine the architectural patterns developers use to secure their code. The two most popular responses were roles and sessions, though these too saw a slight year-over-year decrease. 70% of respondents reported using roles as their architectural pattern, falling from 75% in 2017, and 62% reported using sessions as an architectural pattern, whereas 67% had told us they used sessions in 2017. Secure access layers came in a close third, with 61% of respondents telling us they use this pattern.

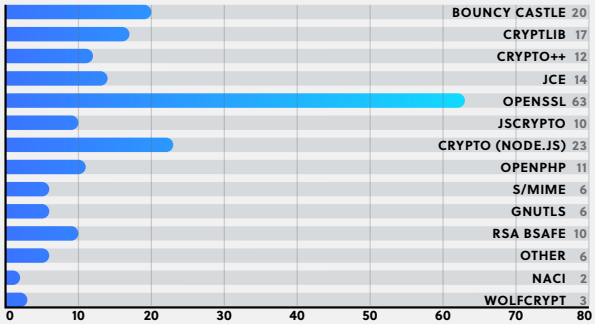
Within these larger architectural patterns, two means of securing code dominate. When we asked what APIs and implementations respondents use for encryption, 63% reported using OpenSSL. Interestingly, OpenSSL proved exceedingly more popular among respondents working as web application developers than those working as enterprise business application developers. Among web application developers, 87% reported using OpenSSL for their encryption purposes, whereas 52% of enterprise business application developers told us they use OpenSSL. Though OpenSSL dominated the responses, the Crypto package for Node.js saw an impressive year-over-year increase. 23% of this year's respondents reported using Crypto, up from 16% in 2017.

The second means of securing code, which dominated responses, was the use of authentication tokens (including digital signatures) for verifying message integrity. 72% of survey-takers said they use authentication tokens in their applications, while the second most popular option among respondents, the use of check sums, received only 34% of the responses. Unlike OpenSSL, however, the use of authentication tokens for securing application code proved relatively equal between web

WHICH OF THE FOLLOWING TYPES OF VULNERABILITIES IN YOUR WEB APPLICATIONS DO YOU ENCOUNTER MOST FREQUENTLY (FROM THE OWASP TOP 10)?



WHAT APIS AND IMPLEMENTATIONS DO YOU USE FOR ENCRYPTION?



application developers and enterprise business application developers. 78% of respondents working as web app developers and 74% of enterprise business app developers use authentication tokens to secure their code.

Unlike the means by which developers secure their code, there's no dominant form of testing. Among our respondents, 24% use penetration testing, 17% use security code review, 14% perform source code analysis, 13% do vulnerability assessments, and 12% have no formal security testing. Given that the difference between the most and least popular form of testing is only 12%, let's again use web app developers and enterprise business app developers as a point of comparison to get a more granular look at how respondents are using security testing. Penetration testing proved the most popular option among both web app and enterprise business app developers, receiving 27% of web app dev's responses and 28% of business app dev's responses. Interestingly, the second most popular option among web app developers was performing security code reviews (18%), whereas for enterprise business app developers it was source code analysis (17%).

Despite the use of these testing techniques, 48% of respondents reported that among the applications they test for security, sufficient test coverage is never achieved. While that number is a bit troubling, it dropped from last year, when 54% of survey respondents reported inadequate test coverage. Additionally, 33% reported that the threat model is handled in an acceptable way, up from 27% in last year's survey.

VULNERABILITIES/ATTACKS

Over the past 12 months, the IT industry has witnessed several large-scale attacks, such as the hacking of Equifax and instances of ransomware like NotPetya, and vulnerabilities exploited, like the infamous struts vulnerability that eventually led to the Equifax hack. But do the realities on the ground match the sensational headlines?

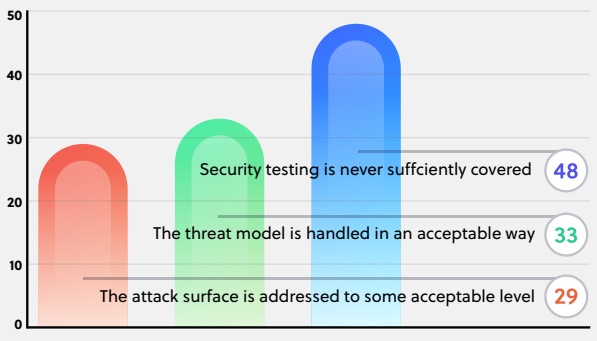
We asked our survey-takers what threats have most concerned their organization over the past year. Despite the flabergastingly bad past year for cybersecurity, our respondents' answers remained virtually identical to those reported in the 2017 DZone Security Survey. 52% reported phishing as their organization's biggest concern, 46% said SQL injection (SQLi), 39% said DDoS, 36% reported ransomware, and 31% said cross-site scripting (XSS) attacks. Even when we compare this data to to our two main developer verticals (web app and enterprise business app developers), the numbers regarding threats that concern their organizations don't undergo any statistically significant changes.

Something interesting does pop out, however, when we compare the threats that most concern organizations and the types of vulnerabilities developers encounter most often. While most vulnerabilites our respondents reported encountering were not that surprising, such as authentication + session mangagment (43%) and cross-site scripting (40%), unvalidated redirects + forwards were selected by a rather small number of respondents. Unvalidated redirects + forwards was the eighth most common vulnerability from the OWASP Top 10 faced by respodents, with 23% of survey-takers reporting to have had issues with this vulnerability. The low position of unvalidated redirects + forwards is surprising given the role this vulnerability plays in the spread of phishing attacks, which was the most prominent organizational security concern among our respondents. Unvalidated redirects and forwards are, in fact, the programmatic mechanism for driving users to a seemingly innocuous, but malicious site (Paul Ionescu, "The 10 Most Common Application Attacks in Action," SecurityIntelligence by IBM). Thus, despite a low instance of phishing attacks over the past year, it seems organizations are bracing for this type of cyberattack to increase in frequency.

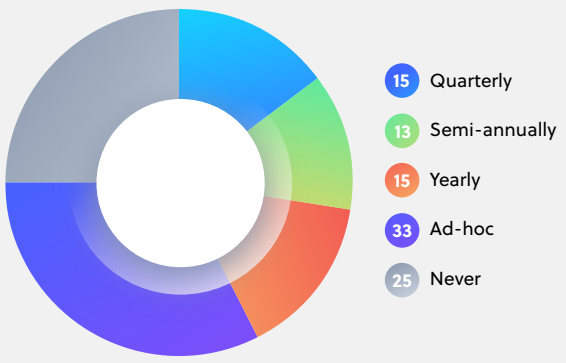
Given that 39% of respondents reported having faced issues with denial-of-service attacks, let's quickly go over the data regarding this common type of attack. Having to deal with many high-resource connections proved by far the most common instance of DDoS attacks faced by survey-takers, with 54% of respondents having faced this issue. The second most common DDoS faced was requests for large files (30%). No other form of DDoS attack registered more than 18% of respondents' votes.

So, how do these attacks and vulnerabilities affect respondents' ability to deploy their software? 43% reported that security analysis and vulnerability-fixing had a medium impact, 36% reported a low impact, and 13% reported a high impact. These numbers are, again, nearly identical to last year's DZone Security Survey results.

OF THE APPLICATIONS YOU TEST FOR SECURITY, HOW IS SUFFICIENT TESTING DETERMINED?



HOW FREQUENTLY ARE DEVELOPERS AT YOUR ORGANIZATION TRAINED IN SECURITY?



Application Security Blanket

Developers and architects are expected more and more to understand and implement secure code and best practices, but only 25% of developers get any sort of training on the subject — and 33% get "ad-hoc training," which is likely a nice way of saying "searching online for the answer." So, let's start with a few basic security design patterns. These should all look familiar to you, because you've probably used applications in your work and personal lives that follow several of them. So, grab a nice cup of coffee and let's begin.

Roles

Utilizing roles is a fairly simple concept: system access is only granted to people in specific roles at the organization. For example, if you want someone to make changes to the frontend of an application, you may only grant access to developers and architects, and automatically include other roles such as marketing, sales, or support staff.

Sessions

Anyone who logs into a system has begun a "session" of time they're spending with that system, and to make sessions more secure, they are monitored so that any suspicious activity that happens can be traced back to the user or attacker who accessed the system.

Secure Access Layers

Secure access layers, or "layered security," is the practice of combining multiple security protocols and tools to protect a system. For each common attack or vulnerability, there should be at least one protection against it.

Single Access Points

Secure access layers, or "layered security," is the practice of combining multiple security protocols and tools to protect a system. For each common attack or vulnerability, there should be at least one protection against it.

Limited Views

In addition to limiting access to specific roles, developers can introduce limited views to each role so that those roles can only see options that concern them. For example, an admin logged into a web page may see options to edit that page, but a standard user would not.

Checkpoints

Developers can utilize checkpoints to organize security checks and their consequences. A checkpoint is more like a strategy or plan that can be changed to prioritize security threats or alerts. For example, forgetting your password is common, so someone typing in the wrong password a few times should not trigger a major warning, but checkpoints can lock someone out of their account after trying more than a reasonable number of times.

diving deeper

INTO SECURITY

Security dzone.com/security

The Security Zone covers topics related to securing applications and the machines that run them. The goal of the Zone is to help prepare the people who make and support those applications stay up to date on industry best practices, remain aware of new and omnipresent threats, and help them to think "security-first."

Performance dzone.com/performance

Scalability and optimization are constant concerns for the developer and operations manager. The Performance Zone focuses on all things performance, covering everything from database optimization to garbage collection, tool and technique comparisons, and tweaks to keep your code as efficient as possible.

DevOps dzone.com/devops

DevOps is a cultural movement, supported by exciting new tools, that is aimed at encouraging close cooperation within cross-disciplinary teams of dvelopers and IT operations. The DevOps Zone is your hot spot for news and resources about Continuous Delivery, Puppet, Chef, Jenkins, and much more.

twitter

 **@malwareunicorn**

 **@LukasStefanko**

 **@jeremiahg**

 **@aprilwright**

 **@campuscodi**

 **@anton_chuvakin**

 **@Lisa_CyberSN**

 **@window**

 **@dannyjpalmer**

 **@e_kaspersky**

refcardz

Introduction to DevSecOps

This Refcard will show you how to get started with DevSecOps with key themes, crucial steps to begin your journey, and a guide to choosing security tools and technologies to build your DevSecOps pipeline.

Docker Security

This Refcard will lay out the basics of the container security challenge, give you hands-on experience with basic security options, and also spell out some more advanced workflows. We split container security into three sections covering what to do at each step of your container security lifecycle.

Java EE Security Essentials

This newly updated Refcard begins by introducing some common terms and concepts related to Java EE security such as identity stores and authentication mechanisms. We then explore authentication authorization, web module security, EJB module security, and application client security with in-depth examples.

podcasts

Southern-Fried Security

All of your information security needs are addressed in this this podcast, which addresses topics ranging from phishing to building a security strategy to evaluating security product vendors.

Risky Business

This monthly podcast features interviews with security aficionados, along with recent security-related news. It's self-described as a "security podcast without the waffle."

Brakeing Down Security

In this weekly podcast, get the latest in the world of security, privacy, compliance, and regulatory issues in the workplace.

books

Cybersecurity & CyberWar: What Everyone Needs to Know

This easy-to-read, deeply informative resource book explores how cyberspace security works, why it matters, and what we can do.

Threat Modeling: Designing for Security

Get the tools you need for structured thinking about what can go wrong when it comes to security, and learn how to adopt a structured approach to threat modeling.

Iron-Clad Java: Building Secure Web Applications

Learn about developing secure Java applications, eliminating existing security bugs, and defending against cross-site scripting.

Golden Rules for Vulnerability Management

From visibility to API integration, from result validation to developer support, the items below should all be considered when choosing a vulnerability management SaaS.

1. Web application assessment coverage is king. Depth and breadth of assessment is key to ensuring all risks are detected. Assessment technology needs to be "tuned" to ensure production-safe testing without sacrificing rigor. Poor coverage can result in "false negatives," undiscovered vulnerabilities residing in your system just waiting to be exploited.

2. Full stack security is key, as "hackers don't give a s*#t" where your vulnerability resides. Web and infrastructure security combined results in full stack vulnerability intelligence.

3. A SaaS with DevSecOps pipeline integration capabilities is well-suited to early vulnerability detection. Keep pace with development as change occurs.

4. False positives are an evil waste of time. Your chosen SaaS should deliver validated and actionable vulnerability intelligence. Risk rating and prioritization of discovered risks are also integral features.

5. Choosing a SaaS with a continuous assessment model solves the issue of securing a system in continuous flux. As new vulnerabilities are discovered in the industry, continuous assessment helps detect if you're exposed.

6. Situational awareness is required. Alerting and custom events are key to knowing what matters and when.

7. Your chosen SaaS should have strong API capabilities, enabling you to automate, invoke tests, and consume vulnerability intel. "If I can do it on the UX, I should be able to do it via an API."

8. Visibility is key and asset profiling is paramount. What's my attack surface? If it changes, do I get notified about what matters to me? "We can't secure what we don't know about."

9. Developer support is crucial. Vulnerability management and intelligence is as much about mitigation as it is discovery.

10. A SaaS with a strong metrics capability helps measure improvement: Metrics such as time-to-fix, vulnerabilities by type, layer, and risk all help. "We can't improve what we can't measure."

11. Asset categorization, tagging, vulnerability risk acceptance, and retesting on demand are all important features to aid prioritization and realistic metrics.

WRITTEN BY EOIN KEARY
FOUNDER, **EDGESCAN.COM**

PARTNER SPOTLIGHT

edgescan

Continuous Full-stack Vulnerability Management SaaS. Deep cybersecurity testing, on demand, validated including developer support.

edgescan

CATEGORY
Cloud-based SaaS focusing on vulnerability detection and mitigation.

RELEASE SCHEDULE
Monthly releases

OPEN SOURCE?
No

CASE STUDY

The edgescan SaaS delivers vulnerability management, detection, asset profiling, and support to some of the world's largest organizations as well as SMEs.

A leading global pharma organization uses edgescan to protect their global estate. Using edgescan's continuous vulnerability management model, they keep pace with change by constantly assessing web applications, API's, and cloud hosting systems for vulnerabilities.

When systems are newly deployed, edgescan automatically assesses them, which ensures the security team are made aware of any risk related issues.

When vulnerabilities are discovered, they are validated and rated for risk by expert security analysts who also supply developer/technical support.

With our SaaS technology, scale and efficiency are a given. Integration via the API with Icon's risk dashboards and GRC was simple, delivering near-real-time visibility of their security posture.

STRENGTHS

1. Full-stack security: Web apps, infrastructure, cloud, internal, and Internet-facing systems.
2. Continuous assessment: Continuous and on-demand assessments included.
3. Vulnerabilities validated and risk-rated by experts: False-positive-free Intel.
4. Developer support: Upskilling of technical staff with support.
5. Continuous Asset Profiling: Attack surface profiling - detection of exposed services and systems.

NOTABLE USERS

- Paddy Power Betfair
- An Post
- Multinational mass media & entertainment conglomerate
- UK Government
- Illumio

WEBSITE edgescan.com **TWITTER** @edgescan **BLOG** edgescan.com/company/blog

www.edgescan.com

Full-stack Vulnerability Management & Intelligence

"The expertise and delivery of this service has been outstanding…"

— Security and Risk Management, Media Industry 30B + USD

Gartner peerinsights™

4.9 out of 5 Stars