DZone
A DEVADA MEDIA PROPERTY

AppSec Trend Report

Defending the End-to-End Lifecycle

BROUGHT TO YOU IN PARTNERSHIP WITH

NS1. SECURE CODE WARRIOR TREND MICRO wallarm

Table of Contents

BUSINESS & PRODUCT
Matt Tormollen CEO
Terry Waters Interim General Manager
Jesse Davis EVP, Technology
Kellet Atkinson Media Product Manager

SALES
Kendra Williams Sr. Dir. of Media Sales
Tevano Green Sr. Account Executive
Brett Sayre Account Executive
Alex Crafts Key Account Manager
Craig London Key Account Manager
Jordan Scales Sales Development Rep.

EDITORIAL
Mike Gates Content Team Lead
Kara Phelps Editorial Project Manager
Tom Smith Research Analyst
Jordan Baker Publications Associate
Andre Lee-Moye Content Coordinator
Lauren Ferrell Content Coordinator
Lindsay Smith Content Coordinator
Sarah Sinning Staff Writer

PRODUCTION
Chris Smith Dir. of Production
Billy Davis Production Coordinator
Naomi Kromer Sr. Campaign Specialist
Michaela Licari Campaign Specialist

MARKETING
Susan Wall CMO
Aaron Tull Dir. of Demand Generation
Waynette Tubbs Dir. of Marketing Comm.
Colin Bish Member Marketing Spec.
Suha Shim Acquisition Marketing Mgr.
Cathy Traugot Content Marketing Mgr.
Melissa Habit Content Marketing Mgr.
Kate Lloyd Customer Marketing Mgr.

To sponsor a Trend Report:
Call: **(919) 678-0300**
Email: sales@devada.com

ORIGINAL RESEARCH

Highlights & Introduction

By Kara Phelps, Editorial Project Manager at DZone

Welcome to DZone's first-ever Trend Report! Trend Reports will release approximately every two weeks. The goal is to expand on the content from DZone Research Guides that our readers have told us is most useful.

How are Trend Reports different from Research Guides? While DZone Research Guides covered broad topics such as Databases or IoT, we designed Trend Reports to reflect the ways our readers define their careers and interests. For instance, rather than a Security Guide, we're offering you this Trend Report on Application Security — with a DevSecOps Trend Report also scheduled for publication this fall. Trend Reports provide predictions based on user research, opinions from leaders at enterprise companies, and more focused thought leadership.

TREND PREDICTIONS

- ► Current security measures don't go far enough, but statistically based static analysis and AI-driven tools will provide stronger defenses against bad actors
- ► Most software will soon be instrumented for security
- ► Organizations who enable a positive security culture and engage developers authentically in thorough security training will ensure they are the ones building the safest software

What You Can Expect

DZone Trend Reports will update you on new advancements, outline how related tools or methodologies have evolved, reveal challenges leaders encounter when executing on the promises of these technologies, and provide current best practices.

In every Trend Report, you'll find:

- An overview of the major developments in the space;
- Key research findings that go in-depth across multiple years of collected insights from software developers — and predictions based on that analysis;
- Expert opinion pieces that explain how and why the covered technology is crucial for the enterprise and what gaps still exist;
- Detailed case studies from sponsoring companies;

- Interviews with executives across several industries;
- Resources for further reading.

Research Methodology

To get as complete a picture as possible of how real software developers are tackling the challenges of their profession, DZone will distribute surveys corresponding to the Trend Report topics to readers who have already expressed an interest in the technology.

We will synthesize the survey results into the Key Research Findings. The Key Research Findings will explore what trends exist in software development, why these trends move in a certain way, and hypotheses on how these trends will move next year. Future Trend Reports will examine and revise previous predictions based on new data and expert opinions.

Interest in AppSec has picked up in the last few years, but release schedules and performance still outweigh security concerns at many organizations.

The surveys will include questions that revolve around what pieces of software are being used for various tasks, what challenges exist when implementing these tools and methodologies, who in the organization is accountable for change, whether they're planning to adopt new tools or strategies in the future, and their thoughts on the major issues pertaining to the topic. These surveys will be further customized based on the best practices, challenges, and types of tools unique to each topic.

Our First Trend Report: Application Security

We chose Application Security as our first Trend Report topic because it's often overlooked — yet so crucial to the future of software development. Interest in AppSec has picked up in the last few years, but release schedules and performance still outweigh security concerns at many organizations. In this Trend Report, we'll highlight the views of three experts in the AppSec space. Jeff Williams, Justin Albano, and Pieter Danhieux share their thoughts on the keys to modern application security automation, the future of secure programming paradigms, and the benefits of "starting left" vs. "shifting left" when it comes to security culture. We're also sharing a few interesting observations we gathered from the results of our AppSec reader survey, which collected 1,870 responses over the course of two weeks in the field. This Trend Report is the product of months of in-depth planning and preparation — on behalf of everyone at DZone, I'd like to thank you for downloading it. I hope you enjoy reading it. Please stay tuned for more Trend Reports (on many more topics!) to come.

OPINION

The Future of Secure Programming

By Justin Albano, Software Engineer at Catalogic Software

Recently, cybersecurity and Application Security (AppSec) have become two of the most publicized topics in news reports — and for a good reason. Since 2013, hackers have breached 3 billion Yahoo and 500 million Marriott user accounts and stolen the sensitive information of 21.5 million security clearance applicants from the US Office of Personnel Management (OPM). These astronomical numbers do not include the disclosure of credit card and other financial information of hundreds of millions of shoppers at Home Depot, eBay, and Target.

TREND PREDICTIONS

- Current secure programming techniques will continue to fall short
- Research is promising on statistical methods for targeted static analysis
- AI for intelligence gathering and attack prediction will continue to level the playing field between hackers and developers

Astonishingly, most of these compromises could have been prevented with simple preemptive measures. As with most software defects, the cost of removal grows exponentially the longer a defect persists after implementation. In practice, many of the security defects that lead to compromises originate during implementation, which means that developers are in a precarious situation: they have the power to stop security defects — which can lead to compromises — at the source. Failure to do so leaves developers responsible for substantial damage to businesses and their customers.

In this article, we will examine the current secure programming practices used by many developers and why these techniques continue to fall short. Leveraging this knowledge, we will then explore some of the future research and technologies that will provide developers with the tools necessary to ensure that applications are secure at inception.

Current Practices

Programming secure applications means that developers are responsible for being knowledgeable about basic best practices and enforcing them when creating code. According to both the Software Engineering Institute (SEI) at Carnegie Mellon University (CMU) and the Open Web Application Security Project (OWASP), basic secure programming can be distilled into a few fundamental tenants, including:

- Validate input from all untrusted data sources.
- Bulkhead different parts of the system.
- Deny all access by default.

- Use mature, well tested security libraries.
- Keep the design of the system simple.

The first step in effectively applying these principles is to understand them from both a theoretical and pragmatic viewpoint; the second step is using them consistently. These steps sound easy, but in practice, we often fail to apply them at the nuts and bolts level.

For example, how often do we check that input data is formatted in a standard character set and does not contain a nefarious command? How often do we use outdated packages or libraries because the cost of upgrading them is too high? How often do we push less-than-stellar code for the sake of meeting a deadline? Even when we try to apply these practices consistently, our aspirations are not enough to ensure that our applications are entirely secure.

Secure Programming Practices Are Not Enough

These practices are useful for ensuring a secure system, but the practical limitations of time and cost — as well as our fallibility as developers — ensure that even with the best intentions, we do not strictly adhere to best practices and we inevitably introduce vulnerabilities into our code.

The first step in effectively applying these principles is to understand them from both a theoretical and pragmatic viewpoint.

Just as with functional correctness, we need help ensuring that the code we write meets our desired specifications. When creating new functionality, we write automated test cases that confirm the code does what we intend it to do. In much the same way, when we write secure applications, we need scaffolding to ensure that we have securely implemented the code. The absence of these tools is the equivalent of coding a feature without tests — hoping that we implemented a new feature correctly on the first attempt.

While automated tests can significantly aid in ensuring we implement some of the best practices — such as creating a test case that assesses whether a module can reject malicious input — more specific tools are needed. In most cases, this means the use of static analysis tools.

Static Analysis

Static analysis tools inspect the code of an application, looking for obvious problems, such as passing unverified input as a parameter to a SQL query or writing characters to a buffer that may cause an overflow. Although valuable, static analysis has two drawbacks: it often reports false positives and is only useful at preventing known vulnerabilities.

Firstly, false positives reduce the likelihood that a developer will heed reported vulnerabilities, burying true errors in a sea of useless information. Secondly, past information aids in preventing known vulnerabilities, but it does not help in preventing rumored attacks or likely attack vectors devised by hackers. Although a solution to the second issue is currently infeasible with static analysis alone — we will revisit this topic in a later section — statistical methods can help in resolving the first.

Statistically Based Static Analysis

Statistical methods can be used to prioritize the vulnerabilities in an application to ensure that the most pertinent threats are brought to developers' attention while less threatening vulnerabilities and false positives are hidden. In much the same way as medical triage highlights the most severe, life-threating cases first, statistical methods can incorporate data from actual compromises and research on patterns of vulnerability to target the most critical liabilities.

The SEI, in particular, has done extensive research on this topic and is working to incorporate this research into tools that can identify and automatically correct common vulnerabilities. While more work is needed before this technology becomes mainstream, it has the potential to focus the attention of developers on critical weaknesses and reduce the burden of inconsequential warnings. Filtering of this type will go a long way in solving the first deficiency of static analysis tools, but another sophisticated approach is needed to address the second.

Artificial Intelligence

Cybersecurity, and more locally AppSec, has become a high-stakes game of attack and defense, with hackers inventing more creative strategies and developers devising more resilient and ingenious resistances to thwart these attacks. Following this pseudo-military analogy, one of the best tools for developers is intelligence. In many cases, rumors of attacks or new attack vectors are discussed on the Dark Web or even openly on forums, providing developers with a glimpse into future strategies that will be employed to debilitate applications.

Unfortunately, the amount of information available is overwhelming and differentiating pertinent threats from benign saber-rattling is beyond the capability of most experts and developers. Artificial Intelligence (AI), in the form of machine learning and deep learning, can be an effective solution to this problem.

Using AI and AI-driven secure analysis tools, developers can use the reconnaissance gathered from across the world and analyze systems for the characteristics matching existing and perceived vulnerabilities. Using existing information and rumors, AI can even make educated guesses on what types of attacks may be coming and what kind of solutions developers can implement to stop these attacks before they occur.

The SEI and IBM (Watson) have done extensive research and testing, respectively, on the use of AI for secure programming. Currently, most implementations focus on system-level security, where AI can detect intrusions and abnormal behavior; but researchers are also focusing on how AI can aid static analysis and observe patterns of vulnerable implementations to ensure developers code secure applications.

Conclusion

Considering the numerous data breaches in the past decade and the mindboggling number of users effected, developing secure applications can feel like a futile endeavor. Despite these failures, forthcoming research on statistical methods for targeted static analysis and AI for intelligence gathering and attack prediction is continuously leveling the playing field between hackers and developers.

SPONSOR OPINION

3 Things You Need for Better End-User Application Experience

By Devin Bernosky, Global Director of Solutions Engineering, NS1

We've seen significant progress toward distributing applications on the infrastructure and application side, but the tools engineers have at their disposal to effectively route traffic to their newly distributed applications haven't kept pace. Mobile data growth and the rise of DevOps has increased the number of applications and in turn users. Although, your app is distributed, the question still remains — how do you get your users to the right POPs?

Today, traffic management is typically accomplished through complex and expensive networking techniques like BGP anycasting, capex-heavy hardware appliances with load balancing add-ons, or leveraging a third party Managed DNS platform. As the ingress point to nearly every application and website on the Internet, DNS is a great place to enact traffic management policies, but the capabilities of most managed DNS platforms are severely limited because they were not designed with today's applications in mind. Below are three things you need for better end user application experience.

1. Route users based on their ISP, ASN, IP prefix, or geographical location. Use geofencing to ensure users in the EU are only serviced by your EU datacenters. Use ASN fencing to make sure all users on China Telecom are served by Chinacache.

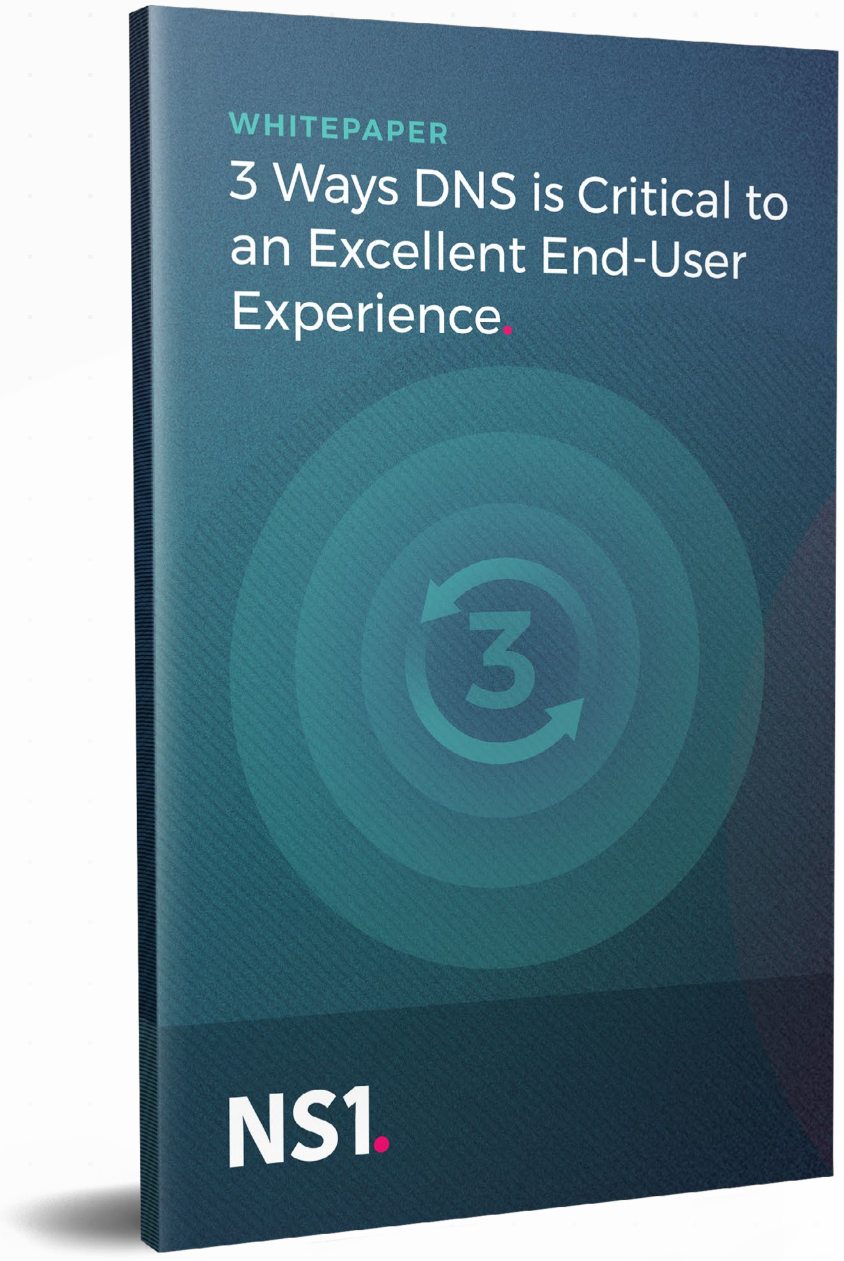
2. Leverage load shedding to prevent meltdowns. Automatically adjust the flow of traffic to your endpoints in real-time based on telemetry coming right from your endpoints or application. Avoid overloading a datacenter without taking it offline entirely, and seamlessly route your users to the next nearest datacenter with capacity.

3. Enact business rules and meet your applications needs with filters that use weights, priorities, and even stickiness. Distribute your traffic in accordance with your commits and capacity. Combine weighted load balancing with sticky sessions — session affinity to adjust the ratio of traffic distributed among a group of servers while ensuring returning users continue to be directed to the same endpoint.

NS1's next-gen DNS platform was built from the ground up with traffic management at its core, and delivers next-gen capabilities and innovative new tools that allow businesses to enact traffic management in ways that were previously impossible.

For businesses that need to deliver Internet-scale performance and reliability for high-volume, mission-critical applications, NS1 is a comprehensive DNS and traffic management platform that uses infrastructure data to make intelligent routing decisions in real time. Unlike traditional DNS technologies that are fractured and rudimentary, NS1's dynamic, data-driven platform and unique Filter Chain routing engine were purpose-built for the most demanding applications on the Internet. Furthermore, NS1's technology closes the loop on modern distributed application delivery by converging dynamic, intelligent, and responsive routing technology with RUM and your application's own unique telemetry. Whether you're building the next big thing or you've already made it to the Fortune 500, NS1's technology is solving previously intractable problems, and improving performance for webscale applications.

NS1.



Download Today
ns1.com/dns-end-user-experience

USA TODAY NETWORK

yelp

The Guardian

LinkedIn

salesforce

SQUARESPACE

OPINION

Getting Started With Modern Application Security Automation

By Jeff Williams, CTO at Contrast Security

Every company faces a terrible dilemma — either turn the business into a software business and risk the potential catastrophic downside of getting hacked or refuse to engage in digital transformation and get put out of business by companies that are good at software.

TREND PREDICTIONS

- True automation will become even more crucial to AppSec
- Securing from the inside out with a DevSecOps mindset will continue to be critical
- In the future, all software will be instrumented for security and more

The Dangers of Digital Transformation

Given the choice, most companies choose software. Unfortunately, it's often difficult to see that the cybersecurity risk is exponentially greater than the risk associated with the old real-world process. We are simply not very good at writing secure software.

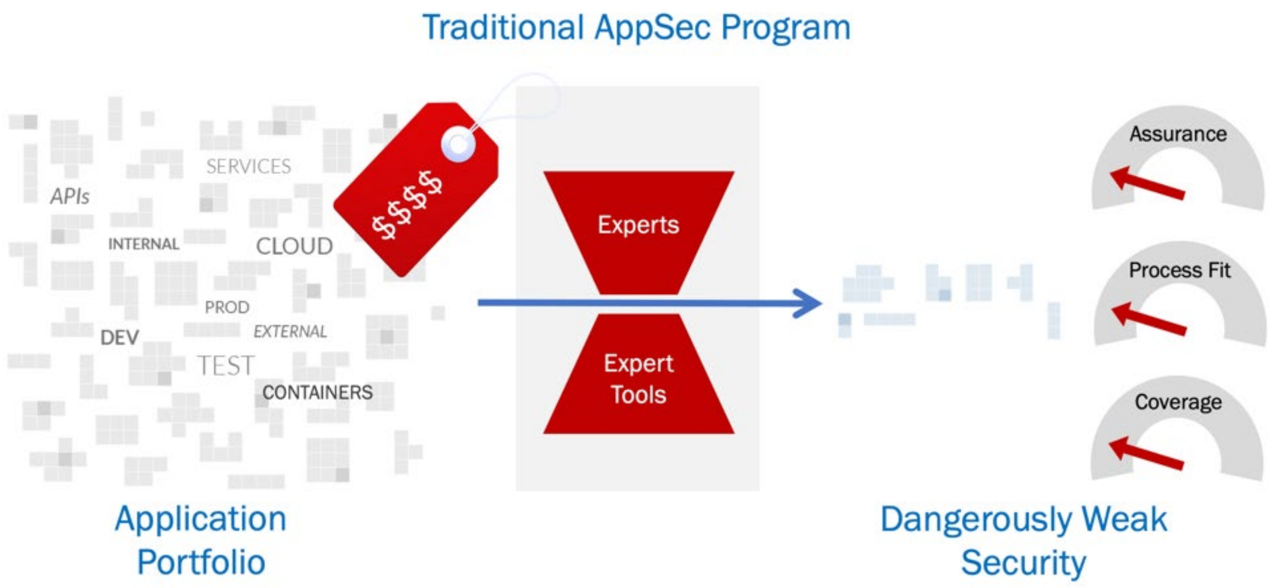
Your web applications and web APIs are the most attractive target for hackers. Over the last 10 years, the hacking game has steadily moved up the stack — from the operating system to your application layer. If you're like most companies, then your applications and APIs are easily accessed by attackers, are full of valuable data and capabilities, and are rife with vulnerabilities.

The average web application is millions of lines of custom code, open source libraries, and configuration files, and it suffers from a staggering 26.7 serious vulnerabilities. These vulnerabilities are primarily well-understood weaknesses, such as SQL injection, path traversal, cross-site scripting, weak access control, and using libraries with known vulnerabilities. The risk isn't limited to your public-facing "external" applications. In an era of cloud, containers, services, and deperimeterization, the distinction between "internal" and "external" is mostly meaningless.

True Automation is the Key to AppSec

In theory, application security is pretty easy. It's just some rules that you have to be careful about when writing your code. Taken individually, they're pretty simple. For example, "don't use libraries with known vulnerabilities" seems pretty obvious. But assessing your entire application portfolio continuously for even this one issue can be complex. And there are thousands of these rules, and many of them are tricky, if not downright counterintuitive. To make matters worse, almost all of these rules are generic and need considerable interpretation to apply to your particular code in your particular environment.

Imagine if you have a portfolio of 100 applications, you have to verify 1,000 rules, you release code several times a day, and these apps (like all apps) are being attacked in production. Many organizations handle this problem by only assessing and protecting their "critical" applications, limiting the rules they check, and only checking periodically. Shockingly, most organizations leave the majority of their applications totally unchecked and unprotected. The more apps you have and the faster you release code, the worse the situation becomes.



The only viable solution is almost complete automation of application security testing and protection. But unless tools are highly accurate, you'll need to involve experts to deal with false positives and false negatives. In fact, if you require experts to install, tailor, run, triage results, or anything else, then your AppSec isn't really automated, and you won't be able to make much progress against securing your entire application portfolio.

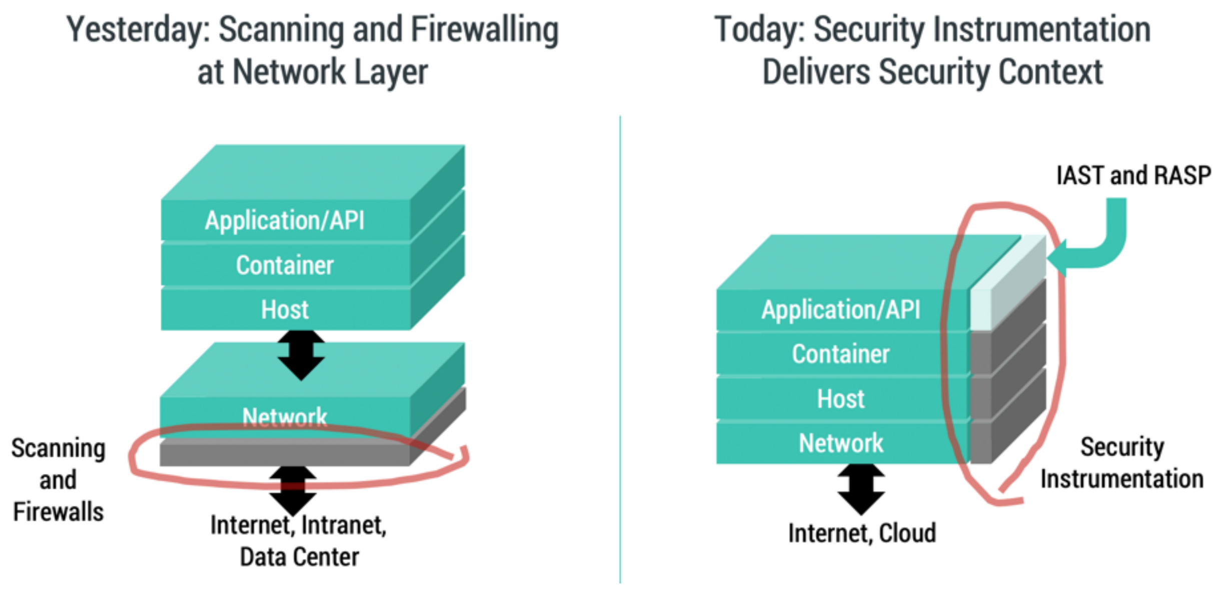
Stop Scanning and Firewalling - Secure From the Inside Out

Historically, static analysis (SAST) code scanners, dynamic analysis (DAST) scanners, open source analysis (SCA) scanners, and web firewalls (WAF) attempt to secure applications from the outside in — parsing code, searching files, trying hacks, and attempting to identify attacks. These tools were designed in a pre-cloud era and haven't kept up with modern software. Like the parable of the blind men investigating an elephant, these tools can only see one "view" of an application, and therefore make a lot of mistakes. These mistakes, both false positives and false negatives, require experts to triage and slow down development dramatically. And automatically starting a scan as part of the CI/CD process does nothing to ameliorate these problems.

Up and down the stack, the modern approach to security works from the inside out. At the application layer, you can add a component to your application that produces continuous, complete, and accurate security without scanning. A single tool can replace all the outside-in tools at once with the following capabilities:

- Application, API, and OSS inventory
- Assess custom code in development with Interactive Application Security Testing (IAST)
- Assess open source with Open Source Security (OSS)
- Prevent exploits in production using Runtime Application Self-Protection (RASP)

Inside-out tools work by dynamically instrumenting applications for security — similar to how tools like New Relic and AppDynamics instrument for performance. This instrumentation can continuously monitor an entire application portfolio for all the AppSec rules at once, in parallel. If the behavior of any of your applications or APIs violates a rule representing a vulnerability, weak library, or attack, you'll get an instant notification with all the relevant details, right down to the exact line of code responsible. Direct measurement of the actual application is the key to accuracy and, ultimately, to a scalable AppSec program.



Achieving DevSecOps at Scale

Working with, not against, development is the key to scaling, as it enables the big machinery of software development to do much of the security work itself.

Organizations seeking to improve their application security should take a hard look at DevSecOps. Unlike heavyweight process models from the past decade, DevSecOps seeks to get security work flowing by breaking it into small pieces, creating tight feedback loops, and creating a security culture. There are many books and articles about DevSecOps, but a few key goals to consider include:

- **Shifting Left** — With easy and accurate automation, we can empower developers to find and fix their own vulnerabilities without experts. But beware of "shifting wrong" by pushing inaccurate scanners onto development teams that are ill equipped to deal with large numbers of false alarms. Instrumentation-based tools can provide instant feedback to development teams through the tools they are already using.

- **Continuous Assurance** — While shifting left is great, we still need to generate assurance that what we push into production has been thoroughly tested for security. Neither static nor dynamic scanners cover the entire application, and they don't provide evidence of what was actually tested. Modern security has to be done continuously across development, test, and production.

- **Runtime Protection** — Don't forget, DevSecOps has to include "Ops." Every company should know exactly who is attacking them, what specific attack vectors they are using, and which applications and APIs they are targeting. Unfortunately, most organizations are almost completely blind — their only attack visibility is based on very noisy WAF data. Modern applications need both automated security testing and automated runtime protection.

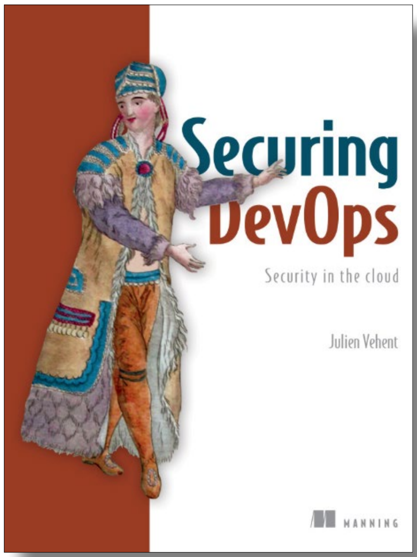
Remember, if you're using the outside-in approach, as your software velocity increases, so does the work for security experts. For every release of every application, you'll have to run SAST, DAST, and SCA scans, correlate the results, triage false positives, create tickets, and retest. This increased work makes achieving DevSecOps virtually impossible.

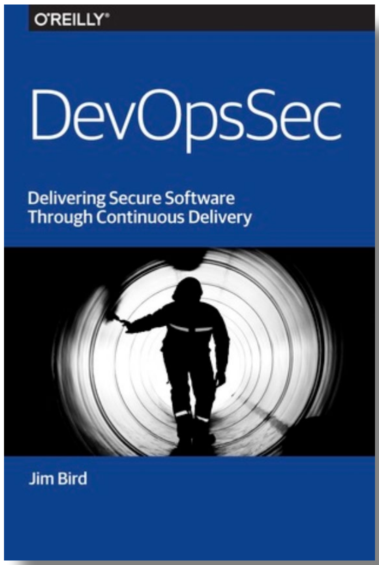
Looking Forward

In the real world, we carefully instrument almost every complex thing: cars, airplanes, factories, spacecraft, etc. We instrument them for temperature, vibration, sound, and much more. But software, arguably the most complex thing that man has ever created, remains opaque. Our best visibility into software is through log files that almost seem to be intentionally designed to obfuscate what's going on. In the future, all software will be instrumented for security and more. There's simply no other way to monitor what's happening inside the complex code we trust with our lives.

Imagine a world where all software is instrumented for security and continuously monitored from within. There's never any scanning. Your security dashboards are always up to date. You can check compliance at any time, not just once a year. You've worked off your vulnerability backlog and new vulnerabilities are fixed as soon as they are introduced. Your pentesters are used to test new risks, and deliver their results as rules, not reports, so their test can be automated and run continuously from that point forward.

You can enable security instrumentation in your applications and APIs today without changing any code. It's the fastest path to taking control of your software inventory, eliminating your vulnerabilities, and preventing your company from being exploited.





Resources for futher learning

SPONSOR OPINION

Developer Tournaments: AppSec's Secret Weapon to Improve Security Culture

By Pieter Danhieux, CEO & Co-Founder, Secure Code Warrior

A developer has a primary responsibility of building software that functional, feature-rich and delivered within strict project deadlines. Security is rarely the priority, and can even be seen as a blocker to rapid delivery and innovation.

AppSec, on the other hand, has the unenviable task of reporting the bad news that security vulnerabilities exist in the code. It's an expensive process in an environment that is often stretched for resources and time, with the setup bound to cause rift between two teams. It's time we changed the conversation about security and made everything a little more positive (not to mention fun!) for both sides, especially the development team. With many developers completing vocational training without learning much on coding securely, it is often the case that their first touch-point with security education is upon entering the workforce.

TREND PREDICTIONS

- Developers will need to skill up and deliver code that is free of security holes
- Application security tools will mature and will have to be built into the developer's environment to accommodate speed and ease of use
- Security flaws will not change drastically, as they haven't done so in the past 20 years

Classroom-based training, video courses, paper-based exams and generic company security policy education are the oft-used solutions, but take away precious time from feature delivery.

We, as an industry, need to tackle viable education from a different angle ... one that harnesses the amazing skills so valued in our developers. They are creative, inquisitive problem-solvers who love a challenge. To gamify security training is to speak their language, to allow them to 'practice by doing' - and who knows, they may just fall in love with security along the way.

Secure Code Warrior's training platform includes a tournament mode in which each developer can validate their skills, see how far they have advanced since training commenced, as well as identify areas that may need improvement. The competition aspect acts as a motivator to engage positively with security, using reward and recognition to support the growth of a robust security culture.

Injecting a little fun into what can be seen as a laborious — if not daunting — task, can go a long way in changing negative mindsets and inspiring continued participation. After all, who doesn't love the glory of scoring more points than their peers in a (healthy) competitive environment? The bottom line is that we must demand better outcomes in security testing. Less common errors, more support for those on the front lines. Why not see how a developer tournament can get you there sooner than you think?

SECURE CODE
WARRIOR

SECURE CODE. IT'S EASY, RIGHT?

Maybe, maybe not...

Test your skills in our secure coding platform.

TRY IT FOR FREE

Do you have what it takes to be a secure code warrior? Your mission, should you choose to accept it, is to test your secure coding skills.

Our secure code training platform has thousands of challenges in scala play, c#, c++, java ee, java spring, c# mvc, ruby on rails, python django, node.js and more, designed to empower developers with the skills to create secure code from the start. Are you ready to play?

securecodewarrior.com

OPINION

Shifting Left is Not Enough

Why Starting Left is Your Key to Software Security Excellence

By Pieter Danhieux, CEO and co-founder of Secure Code Warrior

In a digitally driven world, we are at an ever-increasing risk of data theft. With large organizations acting as the gatekeepers of our precious information, many are recognizing the need to implement stringent security standards.

Much of the initiative around shifting left, that is, introducing security much earlier in the development process, simply doesn't move the needle far enough. There is an implication there that we are still beginning the process the wrong way, ultimately backpedaling to achieve the outcome of more secure software. We must start left, enacting a cultural shift that positively engages development teams and arms them with the knowledge they currently lack. However, all training and tools are not equal. In this article, we explain the ways you can truly empower the development team, transforming them into your defensive front-line against costly cyberattacks.

TREND PREDICTIONS

- Security is transitioning into developers' hands, transforming them into the defensive front-line
- In a positive security culture, developers will have the knowledge and tools to code securely from the beginning
- Proper training that gets everyone involved is the first step toward "starting left" rather than just shifting left

Shift Left vs. Start Left: An Important Distinction

In the age of frequent data breaches affecting some of the world's most trusted organizations, company leaders have looked to the security industry to provide guidance on avoiding the financial, reputational, and showstopping disaster that is a successful attack.

For quite a while, AppSec specialists (including myself) have advised that we must indeed shift left. In keeping with DevOps best practice, as well as better software security outcomes, many of us advised that the security part of a software build must come sooner in the software development lifecycle (SDLC). It should not be the final, costly step — rather, it should shift closer to the start of the process, with AppSec teams engaged early as software projects come to life.

This is not *bad* advice, and it's certainly better than the old way of doing things (which, if the amount of stolen data out there and age of the vulnerabilities used to heist it is any indication, isn't working anyway). However, if we actually *started* left, the security outcomes would be far more positive.

Shift left, start left ... what's the difference? The difference lies in how you engage your development team. Truly, they are the key to delivering more secure software, much cheaper than later-cycle toolchains and manual code review can manage. In an ideal world, every single developer writing software would have the knowledge and tools to code securely from the very beginning.

They would spot potential flaws, mitigating them before they're committed (and become far more expensive to weed out and fix). There would be a dramatic reduction in the security bugs we've seen for decades — ones that are *still* responsible for allowing attackers in through the back door. Those windows of opportunity, in the form of SQL injection, cross-site scripting, and broken authentication, would close.

However, right now, there simply isn't enough emphasis on security at the vocational level, and on-the-job secure coding training varies wildly. As a result, developers rarely have what they need to enable an organization to start left.

Developers Don't Love Security (Yet) ... You Can Change the Conversation

Mention "security" to your typical developer, and you're likely to be met with an eye-roll at best, or puzzlement at worst. Generally, the whole security thing is seen as someone else's problem.

A developer has a primary responsibility of building software that is functional, brimming with innovative features, and delivered within a tight project timeline. Security is rarely a priority at the coding level, and can even be seen as a tedious blocker to rapid delivery and getting creative. AppSec teams have the task of meticulously checking code, pentesting, and then reporting the bad news: the presence of security vulnerabilities in code that is often already committed, and functioning quite fine otherwise. It's an expensive process in an environment that is often stretched for resources and time, with the setup bound to cause a rift between two teams that have ultimately the same goal, but speak completely different languages.

A developer has a primary responsibility of building software that is functional, brimming with innovative features, and delivered within a tight project timeline.

Now, in this climate, the reception to mandatory security training is likely to be quite chilly. But, the power to ignite a security mindset in every developer isn't a pipe dream. With the right kind of training and support, they can start to bake security into their software and take responsibility for the security outcomes they are able to control. If developers themselves can take care of the common bugs, that frees up the expensive specialists to iron out the truly complex problems.

All Training Is Not Equal

When was the last time you got really excited about learning something new? Throw in words like mandatory, compliance, or 17 hours of video, and you're probably not going to get too fired up about the prospect of the course ahead.

Developers are no different. They are clever, creative, and love to solve problems. It is quite unlikely that watching endless videos on security vulnerabilities is going to engage them, keep the content memorable, or end up specific to their everyday roles and responsibilities. In my time as a SANS instructor, it became apparent very early that the best training is hands-on, forcing participants to analyze and be challenged intellectually, using real-world examples that test their brain and build on prior learning. Gamification and friendly competition are also powerful tools to get everyone on-board with new concepts, while remaining useful and practical in application.

Another scary factor is that a lot of security training goes unmonitored. Nobody likes to feel as though Big Brother is watching, but what is the point of spending time, money, and effort on education if no one is checking whether it is relevant?

The right solution can make secure coding fun, relevant, engaging, and measurable. Challenge your developers, treat them well, and make it a special event. Gamified training lights up the reward centers in the brain, and offering an incentive to keep learning, pushing knowledge boundaries and, quite simply, building a higher standard of software, is a win-win.

Security Culture Health Check: Is Yours on Life Support?

Creating secure software in an environment with a poor security culture is like trying to win a marathon with a boulder chained to your ankle: virtually impossible, and unnecessarily difficult.

Gamified training, head-to-head tournaments, and a commitment to assisting developers in their security growth help immensely in driving a positive security culture, with AppSec and development teams gaining much more insight into each other's day-to-day work. Better relationships grow and thrive, and the (often limited) security budget isn't burned through on fixing a 'Groundhog Day' scenario of the same small, annoying bugs time and time again.

When you commit to a positive security culture, responsibility is shared and a greater tier of secure software excellence can be achieved.

There's another powerful byproduct, too: the unearthing of the security champions you never knew you had. Proper training that gets everyone involved, while also allowing for a thorough assessment, can uncover those that not only have an aptitude for security, but actively display a passion for it. These champions are vital in keeping the momentum going and acting as a point of contact between teams, overseeing peers, and upholding best practice policies. Implementing a solid champion program, one that includes recognition and executive support, is a feather in the cap of the organization, as well as looking mighty impressive on the individual's CV and bolstering their future career.

When you commit to a positive security culture, responsibility is shared and a greater tier of secure software excellence can be achieved. Ultimately, every person in the software development lifecycle must adhere to a simple mantra: if it's not secure software, it's not good software.

Spread the word.

Key Research Findings

By Jordan Baker, Publications Associate at DZone

Demographics

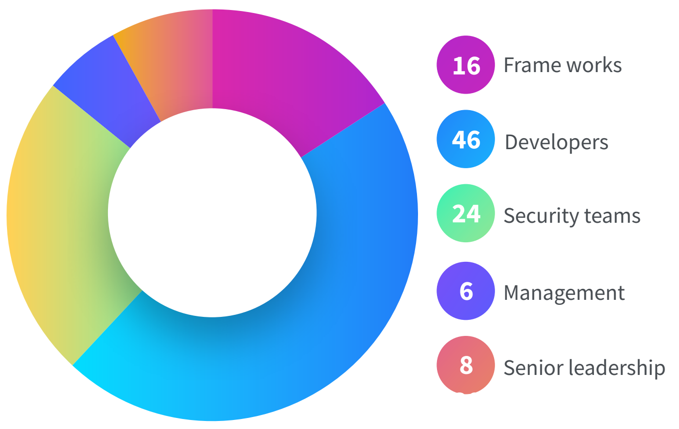
For this first-ever Trend Report, we received 1,870 responses to our community survey on the topic of Application Security (AppSec), with a 51% completion rating. The following is the basic demographic information of this group of respondents.

- Over half (65%) of respondents have 10 or more years of experience in the IT industry.
- Respondents work for companies that develop three main kinds of applications:
  - 87% develop web applications
  - 60% develop enterprise business applications
  - 44% develop native mobile apps
- Within these organizations, respondents fill three main roles:
  - 38% are developers/engineers
  - 22% work as developer team leads
  - 19% are architects
- A majority of respondents work for enterprise-level organizations:
  - 27% work for organizations sized 100-999
  - 22% work for organizations sized 10,000+
  - 19% work for organizations sized 1,000-9,999
- Despite working for such large organizations, most respondents tend to work in immediate teams of under 10 people.

TREND PREDICTIONS

- ► The role of application developers in the fight against vulnerabilities will continue to gain visibility
- ► Source code analysis is growing as an AppSec testing technique
- ► Regulatory requirements will be the main factor shaping AppSec policy over the next 12 months

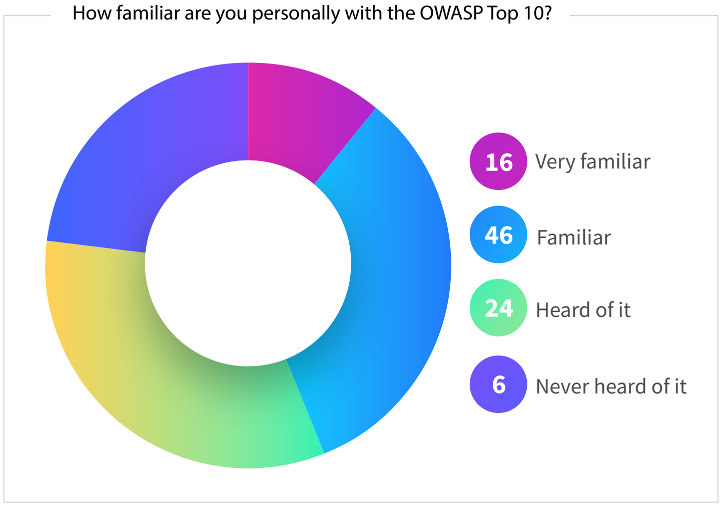
In your personal opinion, who (or what) should be primarily accountable for application security?



- 35% work in 2-5 person teams
- 31% work in 6-10 person teams
- 13% work in 11-15 person teams

- Respondents' organizations tend to use at least one of four main programming language ecosystems.
    - 81% use Java
    - 74% use client-side JavaScript
    - 48% work with the Node.js (server-side JavaScript) ecosystem
    - 42% work in the Python ecosystem

How familiar are you personally with the OWASP Top 10?



- Despite the wide variance in language ecosystems used by respondents' organizations, 57% reported using Java as their primary programming language at work.

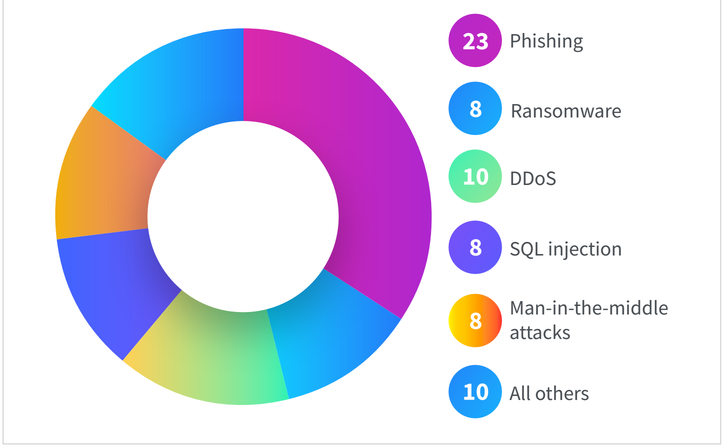
Security and Developers

Developers as the First Line of Defense

With the advent and continuing influence of the shift-left movement among the AppSec community, the role of application developers in the fight against vulnerabilities has increased steadily over the past several years. This year proved no exception. In our 2018 security survey, we found that 42% of respondents thought developers ought to be primarily accountable for application security; in our 2019 survey, this number rose to 46%. Interestingly, the number of respondents who believe security teams should be primarily responsible for AppSec fell from 31% in 2018 to 24% in 2019. Clearly developers have been placed square in the middle of the AppSec conversation.

And, for the most part, developers seem to have taken up this banner rather seamlessly. Of those respondents who currently work as developers, 32% consider themselves to be familiar with, and 7% say they are very familiar with, the OWASP Top 10. There is still work to be done in the developer community in becoming more aware of the top vulnerabilities that exist in the wild, as 40% of respondents who identified as developers told us they have simply heard of the OWASP Top 10, but cannot claim any proficiency in it, while another 20% said they have never heard of the OWASP Top 10. Ultimately, these developer-specific responses matched the overall pattern of familiarity with the OWASP Top 10 among the general survey population. Among all respondents, 33% claim familiarity with the OWASP Top 10, 33% said they have heard of it, 23% told us they have never heard of it (down from 37% in 2018), and 11% said they are very familiar with the OWASP Top 10 (up from 8% in 2018).

Which security threat is your organization planning on allocationg the most resources to in the next 12 months?



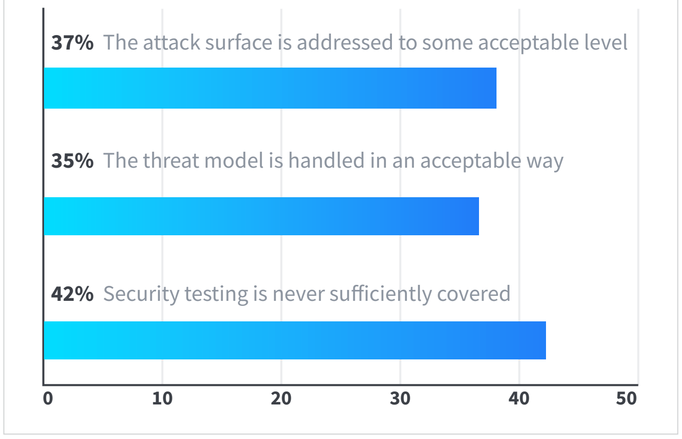
Writing Secure Code

As we've seen, developers are more frequently becoming the de facto security experts for their organization and the first line of defense against cyberattacks. But what techniques are they using in their AppSec efforts? When we asked survey takers what secure coding techniques their organizations use, a variety of answers proved popular. These secure coding techniques included:

- Validating inputs (75%)
- Architecting and designing for security policies (64%)
- Making permission explicit and denial default (47%)

- Using a secure coding standard (44%)
- Executing all processes with the least set of privileges necessary to do the job (41%)
- Sanitizing data before sending it to other systems (41%)

How is sufficient testing determined for applications at your organization?

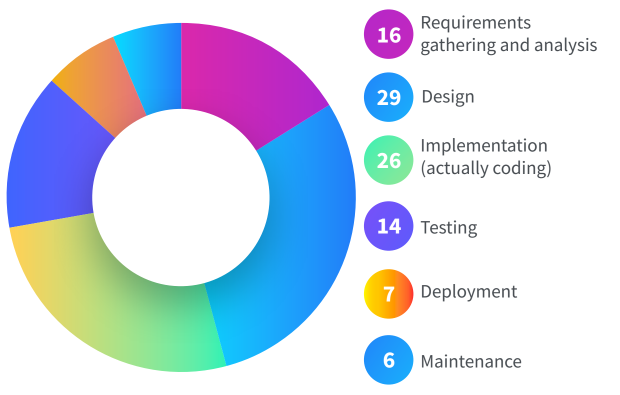


While most of these answers are fairly precise techniques, let's explore the two more vague options, architecting and designing for security policies and using secure coding standards, a little further. Beginning our examination at the architectural level, we find that over half of respondents use the following patterns when enabling security protocols in their applications: authentication/authorization protocols (81%); roles (70%); sessions (54%); secure access layers (53%). As we can see, auth protocols proved the most popular means of building security into an application's architecture. This point was hammered home when we asked respondents how they personally verify message integrity in their application code, and 75% reported that they use authentication tokens (including digital signatures).

Another important secure coding technique reported by respondents is the avoidance of buffer overflows. According to OWASP, a buffer overflow is defined as follows:

"A buffer overflow condition exists when a program attempts to put more data in a buffer than it can hold or when a program attempts to put data in a memory area past a buffer. In this case, a buffer is a sequential section of memory allocated to contain anything from a character string to an array of integers. Writing outside the bounds of a block of allocated memory can corrupt data, crash the program, or cause the execution of malicious code."

At which stage of the application development lifecycle does your organization spend the most time on application security?



As a vast majority of modern applications use, have access to, and produce large amounts of data, this is a good problem to avoid. Among the techniques used to avoid buffer overflows, survey takers reported the following as the most popular: code auditing (35%); bounds checking (31%); the use of compiler tools (29%); only coding in strongly-typed languages with no DMA, including libraries (23%).

To ensure that their applications are safe from malicious code and contain no vulnerabilities, respondents reported using four main testing strategies to determine just how safe their code is. Interestingly, we saw some variance of these practices' adoption rates over last year. Source code analysis proved the most popular AppSec testing technique among survey takers, with a 22% adoption rating; in our 2018 security survey, source code analysis had a 14% adoption rate among respondents. Additionally, in this year's survey we found that 16% of respondents use penetration testing, also known as pentesting; this, however, constituted a rather large drop in the usage of this technique among our survey takers, as pentesting had a 24% adoption rating in our 2018 security survey.

Security and Enterprises

Defending at Scale

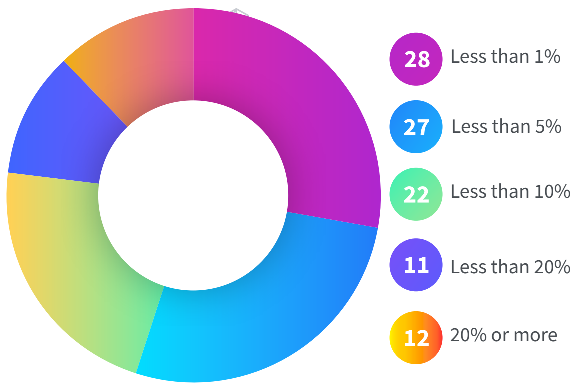
For defending large-scale applications such as the ones our respondents develop, having a well-defined application development lifecycle is crucial. And, for AppSec, knowing when in this lifecycle to implement security protocols can seriously affect the efficacy of your security efforts. For over a quarter of respondents (29%), their organizations spend the most time on application security during the design phase of the SDLC. Another 26% of survey takers told us that their organizations tend to spend the most time on security during the implementation (i.e. actual coding) phase of their software development. This trend of working heavily with AppSec in the first half of the SDLC among respondents' organizations could well be caused by the popularity of the shift left movement discussed earlier in this report.

To learn how, on an organizational level, enterprises attempt to determine the effectiveness of their AppSec, we asked respondents how their organizations determine sufficient security testing for their applications. Alarmingly, 42% of respondents reported that security testing is never sufficiently covered. But, for those who do perform a high degree of security testing, two options proved the most popular: addressing the attack surface (37%) and handling the threat model (35%). Perhaps as a consequence of the lack of security testing noted above, 12% of respondents reported that one-in-five deployments, or more, contain known security vulnerabilities.

Looking Ahead

When it comes to defending software at scale, enterprises must be forward-looking. As such, we asked respondents what their organizations will be doing and/or prioritizing over the next 12 months. We began by asking survey takers which threats their organizations are planning on allocating the most resources. 23% said phishing attacks, 10% said they'll focus on DDoS, 8% reported ransomware, and another 8% said SQL injection. To understand how they mean to defend against these threats, we asked what APIs and implementations organizations planned on using for encryption. Three main responses prevailed here: 63% reported they plan on using OpenSSL; 24% told us they will be using Bouncy Castle; 19% said they'll use the Crypto package for Node.js.

What percent of your organization's deployments contain known security vulnerablities?



Following these questions about more technical decisions, we asked respondents what will have the highest impact on their organizations' AppSec decisions over the next 12 months. Over half (58%) reported that regulatory requirements will be the main factor shaping AppSec policy moving forward. Other popular responses were customer requirements (49%) and the findings/work of security awareness organizations like OWASP and SANS (43%). To better understand the software factors effected by these decisions, we asked respondents to select one of four factors that affects AppSec that their organization plans to prioritize. The results are as follows: 38% will prioritize security; 25% will prioritize performance; 21% will prioritize scalability; and 16% will prioritize maintainability.

SPONSOR OPINION

Build Security into Application Architectures for Continuous DevOps Protection

By David Clement, Global Product Marketing Manager at Trend Micro

To meet new market opportunities, as well as competitive and regulatory requirements, the software development life cycle has gone through several changes. DevOps methodologies are being adopted to develop and deploy applications at a faster, more sustainable rate.

Microservices architectures, along with cloud deployment technologies, are delivering a new level of business value so companies of all kinds can focus on software development to help drive innovation and success.

With new platforms, tools, and services to manage, integrated security practices are generally not top of mind. Many security solutions and point solutions cannot transition across the broad spectrum of an enterprise's security requirements and end up costing more, delivering less value, and overloading security operations teams with more work.

This is why it's good protocol to bring development, security, and operations together as DevSecOps, for a more unified security posture. Embracing security requirements across the build pipeline breaks down the security silos within DevOps, and allows IT security the stability they require. It also provides developers with the flexibility to administer build pipeline security as code to accelerate security efforts and deliver direct protection. This allows for faster development, helps meet compliance, and builds customer trust with secure applications. Security teams have evolved and improved their ability to keep pace with faster development and release schedules. Developers can help improve traditional security practices by implementing automated security throughout the software delivery pipeline, eliminating the number of human touch points, protecting an ever-increasing attack surface, and improving application uptime.

5 Criteria to Help Secure Your Modern Build Pipeline Development Practices

1. Implement a single security solution that reduces dependencies and integrates with your DevOps tools, pipeline, and hybrid cloud environments.
2. Reduce disruption of development schedules and workflows, with automated protection for images, containers, and your host.
3. Implement early detection best practices via application programming interfaces (APIs) by scanning images at build time and repeatedly for the duration of its life in the registry.
4. Maximize threat detection, at both the software build pipeline and runtime, with industry proven and focused global threat intelligence feeds.
5. Help meet risk and compliance requirements by implementing comprehensive threat and risk detection that covers malware, vulnerabilities, secrets, and policy violations early in the continuous integration/continuous delivery (CI/CD) pipeline.

SPONSOR OPINION

Automated Full Life Cycle, Full Stack Container & Workload Security

By David Clement, Global Product Marketing Manager at Trend Micro

Pivvot is a software company that delivers cloud-based, intelligent asset management systems to infrastructure organizations in industries, such as oil and gas, power, and transport. It needed to ensure security while continuing to empower ongoing and rapid innovation on a microservices architecture.

Working with clients in industries where security is paramount, Pivvot requires build scanning, image registry scanning, and runtime host and Docker® and Kubernetes® platform protection. Pivvot must go from development to deploying applications quickly, and with confidence that solutions are secure when released to production. Trend Micro™ Deep Security™ helps Pivvot build securely so it can focus on customer success and business growth. According to Jason Cradit, Sr. Director, Information and Technology at Pivvot, "We're able to protect a container pre-runtime by understanding what's going on in the environment from a security perspective before it even hits production."

Strengths

1. Prevent exploits within the build pipeline.
2. Provide continuous security for unknown threats.
3. Expedite deployments with image assertion.
4. Secure workloads and container platforms at runtime.
5. Meet compliance needs with trusted security.

Learn how we can help you at trendmicro.com/deepsecurity.

Category
Hybrid cloud security and automation

Release Schedule
Monthly releases

Open Source
No

Notible Users

- Orion Health
- Medhost
- Works Applications
- Dealer Socket

Website: trendmicro.com/deepsecurity
Twitter: @trendmicro
Blog: blog.trendmicro.com

TREND MICRO™

Build Secure, Ship Fast, and Run Anywhere

Implement automated build pipeline and runtime host platform protection to secure applications faster with Trend Micro™ Deep Security™

Deep Security provides comprehensive security in a single solution that is purpose-built for virtual, cloud, and container environments, allowing for consistent and seamless security, regardless of the workload. With its application programming interface (API)-first and developer-friendly tools, your security controls can be baked into DevOps processes—giving you the freedom to secure like you deploy: Fast.

How does Deep Security help eliminate threats and attacks that could impact your build pipeline?

Deep Security:

- Protects against runtime container threats, covering the application layer, Docker® and Kubernetes® platform, and workload host
- Scans for vulnerabilities, malware, and secrets at build time, as well as in your container registries
- Automates security for your continuous integration/continuous delivery (CI/CD) pipeline, reducing manual overhead and improving application uptime
- Provides a full stack of security REST APIs for build pipeline and runtime environments across different languages

“Having a security partner like Trend Micro, that keeps up with modern technologies and advanced threats in real time, gives me confidence that my workloads can be protected at any time, even as architectures shift.”

Jason Cradit,
Senior Director of Technology, Pivvot

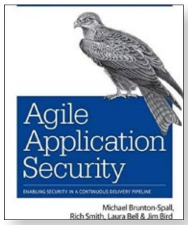
Ready to experience complete protection for your hybrid cloud environments?

Visit **trendmicro.com/deepsecurity** and try Deep Security for yourself.

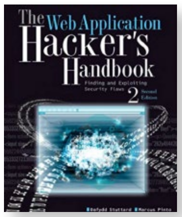
aws   Microsoft Azure   vmware®   Google Cloud Platform

Diving Deeper Into AppSec

Books



Agile Application Security Application security and agile methodology don't inherently play well together, but this book shows how to successfully integrate the two worlds.



The Web Application Hacker's Handbook, Second Edition The second — and most recent — edition of this book was published in 2011, but don't be alarmed. The architecture of the web is still basically the same, and this is one of the best foundational books out there.

Security Automation with Ansible 2 Learn how to leverage the open-source automation tool Ansible 2 to automate tasks like application security, network security, and malware analysis.

Zones

Security Zone 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."

Integration Zone The Integration Zone focuses on communication architectures, message brokers, enterprise applications, ESBs, integration protocols, web services, service-oriented architecture (SOA), message-oriented middleware (MOM), and API management.

Web Dev Zone Web professionals make up one of the largest sections of IT audiences; we are collecting content that helps Web professionals navigate in a world of quickly changing language protocols, trending frameworks, and new standards for UX.

Refcards

IoT Security Best Practices In this Refcard, we'll look to define risk profiles for the most common elements of an IoT system.

Introduction to DNS Security In this Refcard we'll look at how authoritative DNS's ubiquity and critical position in application infrastructure create opportunities to dodge downtime and defend against threats.

Java Application Vulnerabilities Java Applications, like any other, are susceptible to gaps in security. This Refcard focuses on the top vulnerabilities that can affect Java applications and how to combat them.

Podcasts

Security Voices Fascinating people are flying under the radar in the security industry. Listen to conversations with them here.

Application Security Podcast Appsec folks can enjoy topic breakdowns, interviews with appsec thought leaders, and coverage of a wide variety of technologies.

Down the Security Rabbithole News from the security world, debates on hot topics, and debriefs from major security conferences.

ORIGINAL RESEARCH

Executive Insights on the State of Application Security

By Tom Smith, Research Analyst at DZone

We reached out to our community of IT professionals to get their insights on the current and future state of application security. Here are the IT professionals that shared their insights:

- Erik Costlow, Developer Relations, Contrast Security
- James McClay, Product Manager, Cybera
- Doug Dooley, COO, Data Theorem
- Joseph Feiman, Chief Strategy Officer, WhiteHat
- Vincent Lussenburg, Director of DevOps Strategy, XebiaLabs

TREND PREDICTIONS

- ► While overarching security may be gaining more attention and emphasis, security of applications is not
- ► Keys to application security are visibility and having a holistic security mindset throughout the entire application lifecycle
- ► The most common issues with application security are companies doing nothing or trying to manage security with people rather than automation

Key Findings

Application security is bad and getting worse. According to Security Magazine, more than 1,900 breaches were reported through the first three months of 2019, which is higher than the total breaches reported during all of 2018 (1,244). This is a function of the number of apps being produced and the frequency with which they are being updated.

40% of all applications tested in 2018 were vulnerable to a SQL injection – the same percentage that was seen in 2017, 2016, 2010, and 2005. Each application has an average of 15 vulnerabilities. Over 80% of Java applications have vulnerabilities. These figures suggest that application security is not improving.

Companies require 197 days to identify a breach and 69 days to contain it.

According to IBM, companies require 197 days to identify a breach and 69 days to contain it — about nine months. Remediation time has reached 139 days for the most serious vulnerabilities and 220 days for moderate vulnerabilities.

While DevOps is accelerating the development and deployment of code and applications, the lack of security being integrated into DevOps is resulting in the proliferation of insecure applications with vulnerabilities.

Keys to application security are visibility and having a holistic security mindset throughout the entire application lifecycle. If you can't see what's happening with the application, you don't know when and where action is required. Make it easy for development and operations to inject application security in all phases — programming, build, deploy, and runtime — through decommission. Proceed with a multi-perspective approach, since each perspective will yield different insights.

Automation is key. AppSec needs to automate and scale with the staff they have. They will not be able to find and hire the number of security professionals they will need to keep up with the growth of applications and vulnerabilities.

Applications are changing rapidly, but their security is slow to do so. Apps are no longer behind enterprise firewalls. They're on the web, on public clouds, in containers, and on end-point devices like cell phones and IoT devices. Old security tools are essentially obsolete. Innovative ways to find vulnerabilities faster and handle logistics are necessary to remove vulnerabilities.

Unfortunately, enterprises keep spending money on firewalls and spend neither the time nor money necessary to address application security. Application security should be held in the same regard as network security, given that most applications today traverse networks.

There is no one single set of most effective techniques and tools. A good application security strategy involves looking for vulnerabilities from multiple angles. Runtime Application Self Protection (RASP) can be effective because it protects against vulnerabilities through automatic remediation without code changes. However, RASP is being adopted very slowly, and it requires instrumentation in a runtime environment.

There is no one single set of most effective techniques and tools. A good application security strategy involves looking for vulnerabilities from multiple angles.

Three technologies are good to analyze the application from beginning to end: Static Application Security Analysis (SAST) to analyze for the existence of vulnerabilities, Dynamic Application Security Testing (DAST) analyzes application behavior at runtime, and Static Code Analysis (SCA) can find open-source components (which are in 95% of applications) containing vulnerabilities.

Unfortunately, no more than 50% of enterprises are adopting these technologies. Those that do test only a small percentage of their applications because of the time required to address the vulnerabilities exposed by the tests.

Application security providers are able to detect and differentiate between attacks and vulnerabilities across an entire application portfolio and automate remediation. Newly detected zero-day vulnerabilities can be detected and remediated. Continuous testing of apps in production helps ensure they remain resistant to attack.

The most common issues with application security are companies doing nothing or trying to manage security with people rather than automation. Authenticity, authorization, encryption, and an availability mechanism all need to be done with discipline. This requires automation.

Using one or two tools can create a false sense of security. Cloud services offer a complete set of tools that can be automated. Third-party vendors can help SecOps operate at high scale in the cloud.

Concerns about the current state of DevSecOps revolve around misleading representations, lack of culture adoption, and lack of progress in several areas. Different tools simply claim to do things they cannot and result in clients having a false sense of integration and security. As companies have grown and moved to the cloud, they have failed to embrace the need for automation and security. Internal threats are up; machine identity is an issue with IoT and containers; there is a lack of security vulnerability administration and patching strategy; there is a greater risk of a hacker jumping from a low-risk to a high-risk component due to microservices and containers. Application security is only being practiced against a portion of the network of lifecycle, rather than end-to-end. Application security is not getting better, and neither is mean time to resolution or remediation.

The future of application security is automation and software-defined security whereby applications secure themselves through an integrated and automated software security layer. Two-factor and multi-factor authentication will improve identity management while analyzer engines will analyze apps all of the times.

Visionary business leaders will make security as important as quality as they realize the loss of reputation and the penalties for not protecting data become untenable. Customers will vote with their wallets and will support companies and apps that put security, data protection, and privacy first.

Developers should focus on security that requires human thinking, like abuses in business logic. Take a holistic view of security in the application. Understand how application data flows through a network.

Still, developers don't typically spend time worrying about memory management or garbage collection. Neither should they worry about technical security issues. Developers should cooperate with security and teach their security counterparts about how they move fast and how automated security solutions need to facilitate the speed of development and deployment.

SPONSOR OPINION

Solutions for DevOps Challenges

By Ivan Novikov, CEO at Wallarm

Comprehensive security for modern environments is no longer a fool's paradise. There was a time when businesses had to choose between staying apace with development competition and slowing down for security.

Now, security no longer has to run slow, or slow anyone else down. Modern security tech can solve the problems jeopardizing DevOps.

Today's application security teams face an accelerating pace of hacker attacks, diverse APIs, and rapidly changing code base. APIs, web applications, and microservices help DevOps but complicate application security. Processes—like code freezes for security testing and version control—have unfairly portrayed our protectors as pests mucking up the works.

DevOps presents a seemingly impossible problem. New solutions specialize in areas where security struggles and DevOps blooms.

Goodbye Old Solutions

Outdated security tools can clog up CI/CD pipeline—and overlook vulnerabilities, code anomalies, and attack payloads.

Take legacy WAFs. Today, APIs needs to flow freely through more dynamic environments despite enormous and complex data loads. Traditional WAFs—based on signatures and regular expressions—cannot parse complex modern protocols. Nor can they keep pace with daily new hacker attacks. Similarly, resource-heavy legacy static analysis tools clog CI/CD pipelines. They scan indiscriminately, including previously scanned data.

Traditional solutions simply don't work for DevOps. It's time for security to 'shift left' via automation and machine learning.

Superpower Existing Workflows

Do not reinvent the pipeline. The only way security can work in tandem with the DevOps environment is to work inside it. That's why we have designed solutions that plug into your architecture and workflows.

Designed for continuous coverage, Wallarm works at the application level to provide stronger protection over diverse and scaling payloads. In development, Wallarm plugs into your workflow to transform existing tests into baselines that automatically run security testing. Post-release, we protect at the API level, applying machine learning and expanded intelligence. And all solutions deploy easy and stay that way. They start stronger out of the box and get better as you grow—no duplicative tests, super high false positives, nor manual rule configuration.

We want to make security easy. With automation, there is no more heavy lifting. We run where you run: in public, private, or hybrid clouds.

Secure your entire DevOps environment,
not just parts of it

Automate end-to-end application coverage with an adaptive security platform

Wallarm delivers continuous protection with:

- Focused app, API, and serverless security
- Active threat verification & streamlined analysis
- Automated security testing in your CI/CD pipeline
- Native to any cloud—public, hybrid, or private

wallarm.com/automate-security

Featured integrations:

Azure

Google Cloud Platform

Amazon Web Services

Kubernetes

wallarm

INTRODUCING THE

Security Zone

Security is a perennial hot topic in the developer community. From new threats to securing open-source projects, there is plenty to learn on the security front.

Keep a pulse on the industry with topics such as:

- New threats and defenses
- Educating users on common threats
- Maintaining secure open source projects
- Learning to use OAuth 2.0, TLS, SSL

Visit the Zone

TUTORIALS

CASE STUDIES

BEST PRACTICES

CODE SNIPPETS