Sonatype

DEV**SEC**OPS: EARLY, **EVERYWHERE**, AT SCALE

EXPERT COMMENTARY ON THE 2018 DEVSECOPS COMMUNITY SURVEY

eBook

TABLE OF CONTENTS

Deploy Secure Applications with DevOps: DevSecOps

Hasan Yasar, Technical Manager of the Secure Lifecycle Solutions Group, CERT Division of the Software Engineering Institute | Carnegie Mellon University

Developing a secure application is no different than delivering a high-quality application as long as security is addressed throughout the software development lifecycle.

It seems easy to say, “Let’s assess security at every phase of the development lifecycle”. But, in reality, it is very difficult to integrate traditional security practices into the software development lifecycle for DevOps because there is no time for manual security testing, secure code reviews, dependency assessments, and audits. Also, there is no opportunity to put in control gates and perform extensive security reviews because the review take longer than the commit and deploy cycles.

In DevOps practices, the velocity of change increases due to rapid business changes, growing vulnerabilities, software complexity, and dependencies on third-party libraries (commercial or open source). There are also many questions like:

- Where and how to address security?
- Which method is effective?
- How to link security activities?
- What is the most important security check to perform?
- Who is responsible: development, security, or both?

Most often, security folks do not like frequent constant changes on the delivery pipeline with releasing a new version because they have to reassess, retest, and approve new applications builds even though the new builds many have a small change. As a result, most of the organizations who do not practice DevOps principles bolt on secure testing at the end of the SDLC. These organizations also face some challenges like organization culture, operational silos, and toolset complexity when trying to integrate automated security testing into the development lifecycle.

Culture: Organizations as a whole can make a huge impact on how security is implemented throughout the SDLC. This is more than just a mantra handed down from people at the top of the organizational chart. It is the organizational culture and policies along with the organization’s structure.

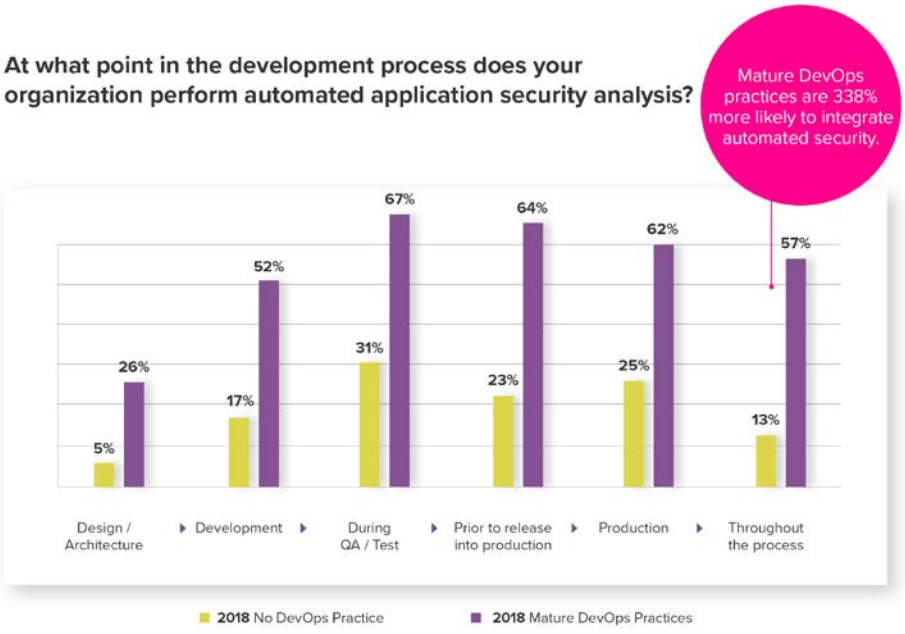
There are several opinions about the correct company structure and how to set up a company with security in mind. However, most companies are do not have green field operations. Their operational practices are already established and this introduces obstacles for them in transitioning to a security-focused development strategy.

There isn't simply one easy way to overhaul your company's culture, policy, or structure. More importantly, this is not a practical approach. The key to successful transformations lie in how development organizations can leverage the knowledge, experience, and tools from within their security teams to develop secure applications from inception to deployment.

Silos: Silos are often brought up when discussing barriers to new methodologies and technologies in the software community. Unfortunately, security, which is often its own silo, is another stakeholder that needs to be part of a project from its inception. Forcing strict guidelines from a security silo into the development realm can create strong backlash as enforcing those practices slows development and release cycles.

Discrete Toolset: Another major barrier to implementing security is the difference in toolsets between security, operational, and even other development teams. Typically, developers have their favorite tool for each development activity. They also have limited knowledge and desire on using security analysis tools on different phases of SLDC. While many security tools allow for easy integration into development tools, that is not always the case. Tools may integrate with each other, but integration can take a lot of time to manually set-up and may require continued maintenance over time.

To overcome these challenges, security integration into application development lifecycle requires practice with DevOps. DevOps is a set of principles and practices emphasizing collaboration, empathy and communication between software development teams and IT operations staff along with acquirers, suppliers, security teams, and other stakeholders in the life cycle of a software system [1].



At the minimum, security analysis can be performed at Design/Architecture, Development, QA/Test, Staging (Prior to release into production), and active monitoring production phases. And as pointed earlier, security analysis should be done throughout the application lifecycle. The 2018 DevSecOps Community survey data shows an average 15-20% improvement of security analysis on each phase in either mature DevOps organizations or others for this year. (Fig -1)

Mature DevOps organizations are able to perform automated security analysis on each phase (Design, Develop, Test) more often than non-DevOps organizations. Since DevOps enables strong collaboration, automation of the process and enforces traceability, mature DevOps organizations are 338% more likely to perform automated security analysis (57%) than non DevOps organizations (13%). Furthermore, when organizations adapt to DevOps processes they will get major benefits with increased team efficiency, traceability, complete visibility, and a quick incident response to cyber incidents.

Implementing security for an application is a difficult problem which is solved easily with a DevOps mindset. In non-DevOps mindsets, integrating security is often avoided because it is seen as an extra effort without getting any new features or improvements. However, this type of thinking is flawed. Security should be seen as a non-functional quality attribute and should carry the same, if not more, importance as the other major functional quality attributes like availability, usability, etc. As such, Security and DevOps can align perfectly with other attributes of application quality assessments and remediation paths. Once a secure SDLC is setup, there are several benefits that arise from inception to operate with constant collaboration.

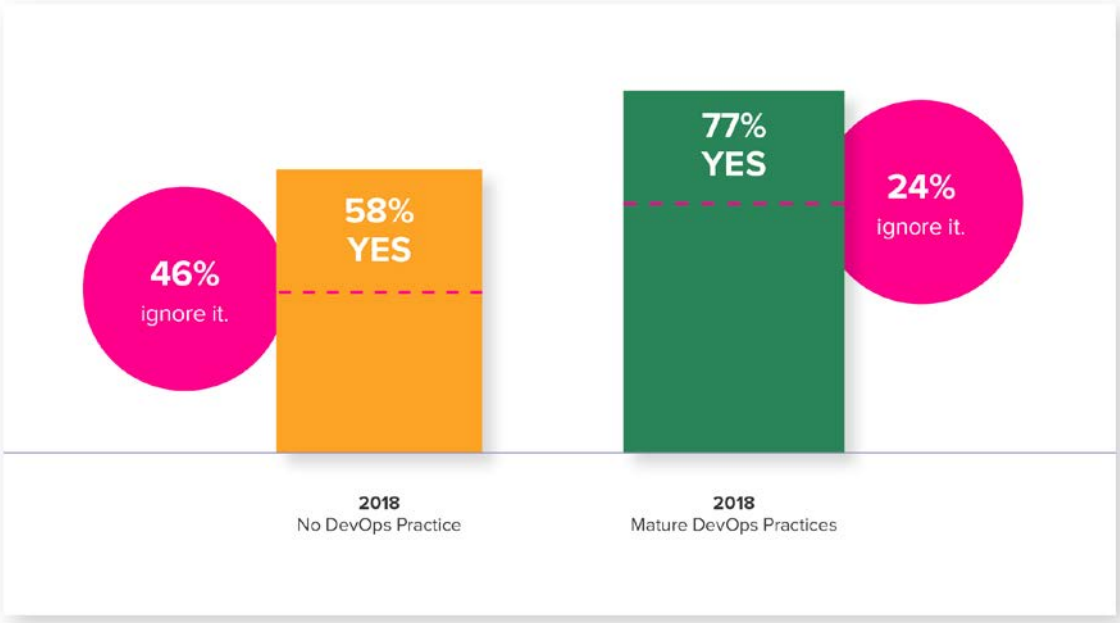
[1] IEEE P2675 DevOps Standard for Building Reliable and Secure Systems Including Application Build, Package and Deployment

The Importance of Having An Open Source Policy

Helen Beal, DevOpsologist | Ranger4

In 2017, 57% of all participants in the DevSecOps Community survey confirmed that, yes, they did have an open source policy. In 2018 this has risen to 64% - but 35% say they ignore it.

Breaking that down further: in 2018, 58% of those with no DevOps practices and 77% of those with mature DevOps practices reported having an open source policy. 46% of the former and 24% of the latter reported ignoring it. Effectively, that's then just 12% of organisations with no DevOps practices actually using an open source policy, while 53% with mature practices are following internal regulations. Having, and using, an open source policy is then an indicator of mature DevOps practices.



The above begs the question: why are people ignoring the open source policies they have? It's worth noting here that nearly half of survey respondents reported being in development or DevOps roles and less than 3% reported having AppSec or security specific roles. That number shouldn't be a surprise alongside the discovery that developers typically outnumber security 100:1 (and, according to the survey, operations 10:1). But, perhaps what we can conclude is that it's the developers who are ignoring the open source policies that are set by the security professionals in an organisation.

A principle of DevSecOps is that security is everyone's job, not just one person's job. Organisations reporting higher levels of maturity also report higher levels of this principle of thinking, phrased in the survey as: "Security is part of everyone's role" - 91% versus 78%. Note though, that both these levels of acceptance can be considered high.

Is it concerning then, that there are low levels of acceptance of an open source policy? Is an open source policy a critical part of security policy and procedures?

First, let's reflect on what constitutes an open source policy. There are several examples available on GitHub from Google, Linux Foundation, Rackspace and Zalando. A key focus of these examples is how these organisations create open source software themselves, rather than how they consume it, although Rackspace's says:

"An open source policy exists to maximize the impact and benefit of using open source, and to ensure that any technical, legal or business risks resulting from that usage are properly mitigated."

The benefits of using open source are obvious - why reinvent the wheel? The ability to scale development activities by consuming pre-built artifacts is clear. The appeal of using one technology that's free rather than buying a licenced, chargeable piece of software is also apparent. But so are the risks - so it is concerning that some developers are simply ignoring the policies crafted and communicated for their organisations, likely for the sake of speed and costs.

IDG's 'Best Practices for Creating an Open Source Policy' provides practical advice and names developers first in the list of stakeholders to include:

- **"Developers - the people who will have to follow the policy**
- **IT staff, as they probably download and use open source software**
- **Managers of teams that use open source software**
- **Attorneys**
- **CIO and staff**
- **Technical architects; many companies have architectural committees, and they should be involved"**

And they also include the qualifier: "the people who will have to follow the policy". But it's eight years since this article is written and this current survey tells us that many of the developers aren't following the policy. "Ignoring" is a strong word - it suggests wilful obstruction. We know though that the vast majority of developers we work with take enormous pride in their work - they don't want to create licensing and security issues for the rest of their organisation through the choices they make for the components in the software supply chain. But it's just too hard for them to find out whether what they are about to download and incorporate into their application comes with what level of risk.

And they are not alone. The IDG article goes on to explain why you might need a background statement for your Open Source Policy, for example, when:

- **“Management doesn't know how much open source software the company uses or depends on**
- **There are widely varying opinions on how much open source software is used**
- **There's an open source software rule or policy that conflicts with reality (e.g., "No open source allowed," but your IT infrastructure is built upon open source software)**
- **There are big disagreements on how the company should use open source software”**

The 2018 DevSecOps survey data also tells us that 38% of organisations have a complete bill of materials for each software application while 62% of organisations report that they do not have meaningful controls over what components are in their software applications.

This doesn’t have to be the case. We can automate all of this with tools like Sonatype’s Nexus Lifecycle. We can automate the bill of materials and make it visible to all where we might have open source components in a software supply chain and highlight the risks of using particular components whilst offering alternatives - and we can do this in the IDE, thereby taking the burden off the developer so they don’t ever have to sacrifice quality of their product for speed of delivery. You can automate your open source policy and break the 100:10:1 security constraint.

What Does DevOps Maturity Tell Us About Security Maturity?

Benjamin Wootton, Co-founder and CTO | Contino

In March 2018 alone, IT Governance counted 20,836,531 records leaked (and that's only in the news!), including hacks at the NHS and the National Lottery.

There's even one incident in which someone accidentally sent information to the wrong *fax number*.

Here's a selection of other major security breaches that shook both enterprises and governments recently:

- **Equifax breach:** affecting 143 million customers
- **Facebook and Google:** defrauded of $100m by a rogue Lithuanian hacker
- **Deloitte:** cybersecurity 'experts' at Deloitte had failed to adopt two-factor authentication allowing hackers access to their entire email system.
- **NSA:** leaked reports of a breach in the main digital defense branch of the US government by North Korean or Russian hackers.
- **WannaCry:** this ransomware infected over 230,000 computers in over 150 countries
- **NotPetya:** hundreds of millions of dollars in losses caused to companies including shipping giant Maersk thanks to this ransomware

The levels of risk in play, as well as the frequency and intensity of attacks, have never been greater.

We often talk about a 'compelling event' that pushes enterprises into action….how about losing a decent chunk of your market cap overnight?!

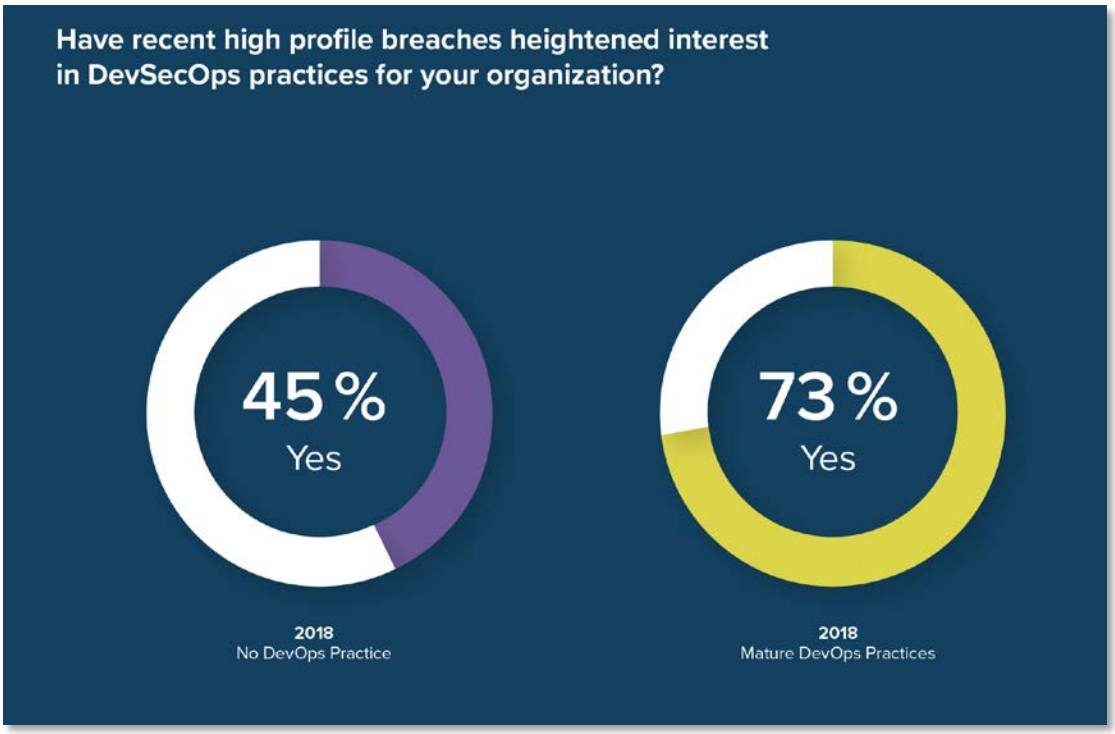
So you'd expect a rise in interest in security best practices, right?

Have all these recent breaches resulted in greater interest in security best practice?

Security at Speed and Scale

While for some this is the case, for others the security penny has not yet dropped, as the results from Sonatype's 2018 DevSecOps Community survey make plain.

Only 45% of those without a mature DevOps practice reported heightened interest in DevSecOps practices as a result of recent high-profile breaches (compared to 73% of those with mature DevOps practices).



This suggests that for more traditional IT teams, security is less of an immediate priority. Speed is the first concern, with security still considered a last-minute, box-ticking exercise that is tacked onto the end of the software delivery lifecycle.

Equally, the survey shows that high-performing IT teams have been paying attention to tech news. Three in four recognise that the rise of security threats requires a serious response from businesses.

This suggests that DevOps practitioners are more aware of the importance of security when it comes to delivering high-quality software at speed and scale (not to mention deploying the underlying infrastructure!). It's understood more as a central principle, rather than a nice-to-have-if-you've-got-time.

Still, I'm surprised the number isn't higher! Given the stakes, it should be approaching 100%.

Cybersecurity Readiness

The survey also reveals a shocking lack of preparedness for cybersecurity attacks.

The traditional teams give themselves a lowly 3.4 out of 10, compared to a 6.3 out of 10 ranking for mature DevOps teams.

Let's imagine the kind of company out there that these stats suggests: they rate themselves about about 3 out of 10 for cybersecurity readiness. *They know they're vulnerable.*

At the same time, enormous breaches at the likes of Talk Talk and Equifax haven't really stirred that much interest in DevSecOps. Would you place your bets on such a company?

Worryingly, even the mature DevOps teams rate themselves only 6.3 out of 10. It seems that DevOps with a security mindset is not enough. Full-blown DevSecOps – in which security is a foundational principle of software delivery and considered from the word 'go' – is needed.

The Relationship Between DevOps and Security

Of course, the other effective way to stay competitive in tough times is to foster digital innovation. DevOps is a proven method for releasing better software, faster.

But when digital innovation at speed and scale becomes the modus operandi of a business, traditional bolt-on security practices can easily fall by the wayside as it otherwise becomes a major roadblock for deployments.

However, the statistics above taken together suggest that engaging in DevOps practices has a positive impact on how you think about security.

Firstly, they reveal the importance of security – you suddenly learn that you can't release at speed without streamlined security and governance controls.

Secondly, they free up time and energy to dedicate to security best practices – you spend less time manually patching machines and finally have enough time to set up configuration management tools that can patch all your machines at once, for example. This, in turn, allows more time for innovation, and so the virtuous cycle continues.

Security Is Evolving

I've seen nearly 100 digital transformation projects at some of the world's most highly-regulated enterprises, like Allianz, HSBC and Barclays. DevSecOps has proven to be immensely valuable in every case.

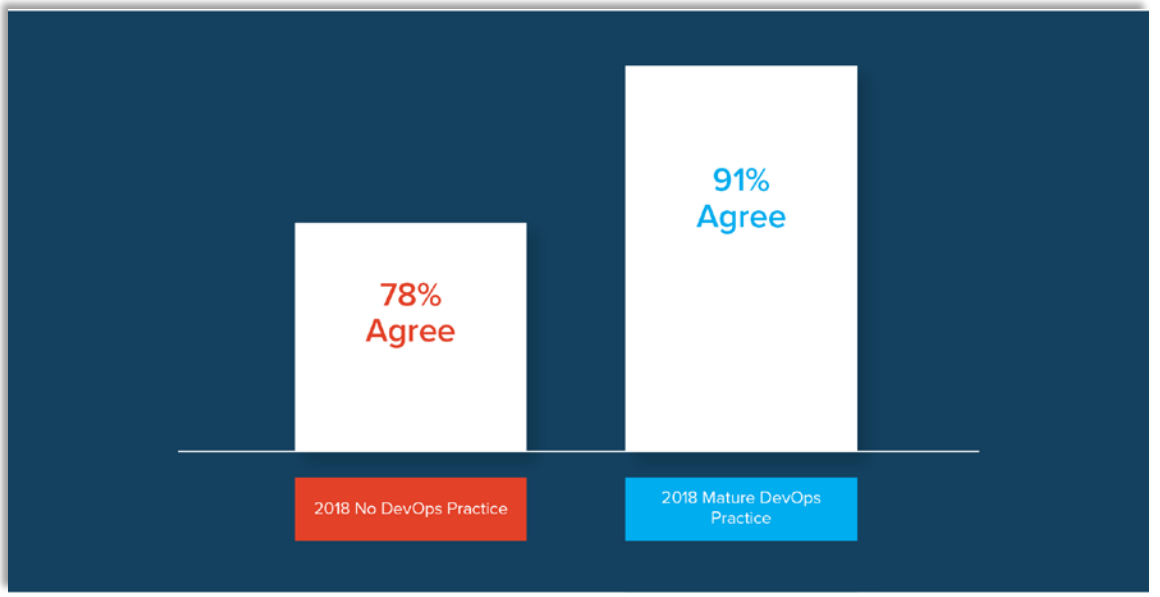
Why?

Because if security is left out of the conversation, DevOps transformation usually fails to deliver on its promise. Security is now a major pillar of software delivery. It must be a principle that is embedded across your entire software delivery pipeline! Another important reason is that security threats will continue to evolve. Enterprise security measures must follow suit. Ultimately, then, the only proper approach is the ability to be flexible and adaptable.

The DevSecOps Mindset: We Are Not Alone

Oleg Gryb, Chief Security Architect | Financial Services Industry

I'm not going to argue about why "Security is Everyone's Role" - we've already agreed that it is, and there is no point in continuing that discussion. Instead, I'll try to explain how DevSecOps has been influenced by that mindset - and should also be everyone's business



A Long Bit of History in Two Minutes

In the beginning there were mainframe-based batch processing applications created by Dev and Ops supporting big computers, but

Then, light was seen coming from alphanumeric terminals that supported a client server architecture. App specific Ops was born somewhere around this time. Information security -- at least the way how we know it today -- was still nowhere close.

Then God DARPA said, "let there be Internet, be fruitful, and multiply, and replenish the earth with the streams of useful information available to everyone". We all saw that it was all good, until we noticed one bad thing about it. There were bad guys trying to get access to information that they were not supposed to see. This is when information security was born.

At that time security has been completely separated from Dev and Ops. It was not embedded within any SDLC process. Newly created InfoSec companies were coming in to pen test systems and generate long PDF reports with findings and remediation suggestions. Security was a self-sufficient, black box kind of thing that didn't care much about how those PDF reports would be used.

Meanwhile, business wanted software development practices to move faster, and Dev responded with Agile methodologies (but completely forgot about Ops, which was deeply rooted to the way new apps and systems were created at that time). It then became obvious that making Ops the part of this process was a must, and this is how DevOps was born.

All of that was good, but very soon everyone has realised that if security remained isolated, there was not much the DevOps movement could do to achieve the business goals mentioned above. Next, the DevSecOps term was coined. (I could find the first use of it in a tweet published on 03/09/2012).

We Are Not Alone

The major question here is: have we included all necessary parties this time? An obvious answer coming from the title of this article is: of course not.

The thing is, while DevSecOps or DevOps concepts could be sufficient for small companies and startups, where any employee belongs to one Dev, Sec or Ops category -- for bigger, well-established and heavily regulated companies, it's never the case.

There are many other parties involved: e.g. risk, legal, compliance, governance, change control (yes, it still exists), etc. Believe it or not, all of them are part of a software development process. I can recall a case when in the middle of Dev process when the engineers were asking attorneys questions like: "what will happen if we allow users on our system without asking them to agree with our terms", or "how will this new feature impact our risk and compliance posture".

Please note that in situations like these, risk, compliance, and legal pros become a part of the software development process. And if a team runs in agile mode, these three groups don't have the luxury to research an risk, compliance or legal issue for a month or two. They need to provide an answer fast to make sure that Dev, Sec, Ops meet their sprint deadlines.

If you think about all that, you will probably agree that in big organizations DevSecOps is really DevSecOps plus almost everybody else.

We've started with Dev, then realized soon that Ops and Sec should be included as well. Now, let us think about how everybody else could be included to the process.

It's not only about integrating automation throughout development, deployment and security, it's also about big changes in processes. It is also about the way different -- and not necessary technical organizations -- are involved within a software development life cycle and their ability to respond fast to the needs of all other parties involved.

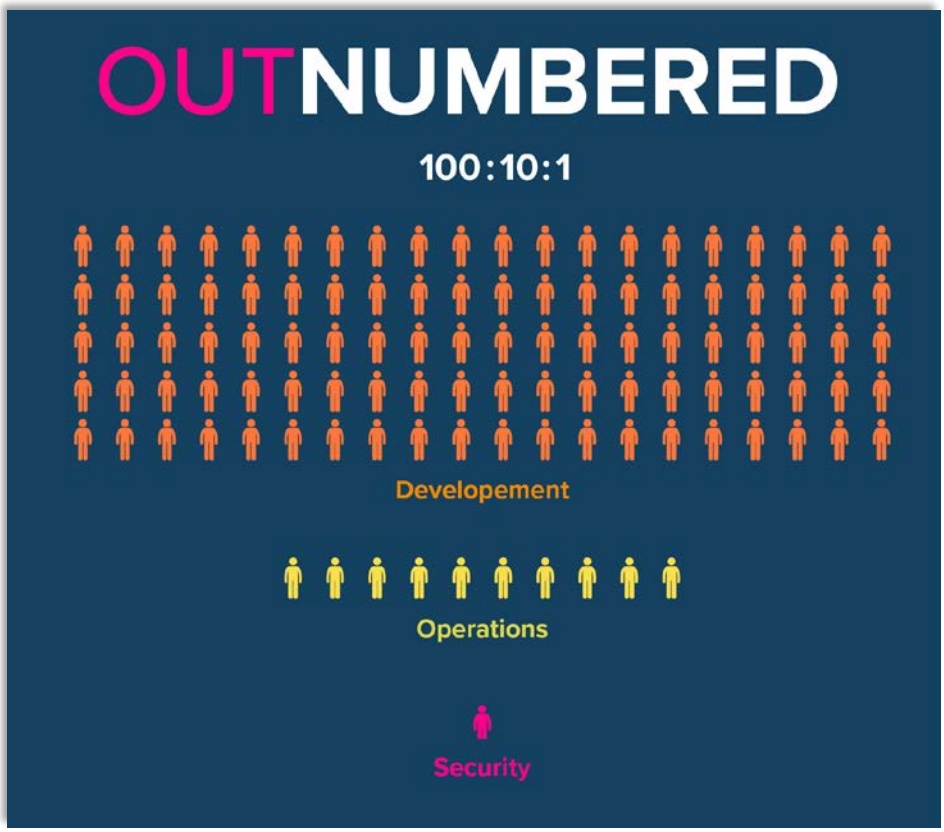
I think, the next big thing, which is coming after DevSecOps is **DevSecOpsAndEverybodyElse.**

Either do that in your organization, or admit that you can't keep pace with the speed required by business (and that you are a show stopper party pooper).

How to Design Teams to Bridge the Security Gap

Manuel Pais, DevOps and Continuous Delivery Consultant

Even if you work for a large organization, chances are that you could memorize the names of all the security folks working in IT. That's because the ratio of developers to security is 100:1, according to a recent survey (the same study indicates a 10:1 ratio of devs to ops). Previous studies have reported ratios from 100:3 to 100:6, so there's some progress but not fast enough.



That doesn't bode well for these organizations' ability to tackle upcoming data privacy protection regulations like the European Union's GDPR and stop the data spilling that has been on the news on a regular basis over the past few years. There aren't enough skilled security folks out there today to fill in the gap. Successfully adopting secure development and operations practices requires not only deep technical understanding, but also, critically, possessing the communication, persuasion and balancing skills to achieve buy in from development teams and (more often than not) senior management too.

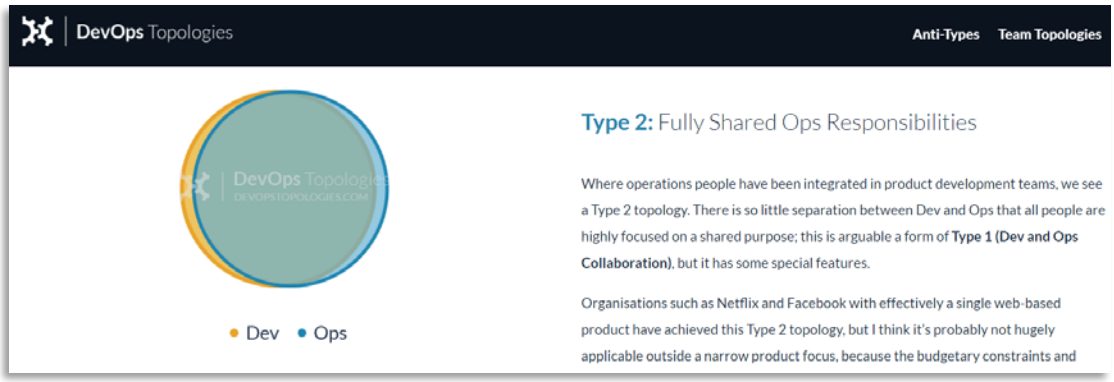
We're still inside a vicious cycle where security specialization is not attractive enough because, historically, companies have not given security enough priority. Those same companies are now facing a growing need to up their security game, but they lack enough people with the right skills.

So what options do we have to bridge this security gap?

Let's first clarify that automating basic security checks (such as third party vulnerabilities) and coding guidelines and integrating security in the delivery pipeline should be a no brainer for any product development team today. The toolset in DevSecOps keeps expanding and both community and enterprise-grade requirements are well served.

So where's the gap? It's in the "hacker mindset", a constant search for new potential attack vectors in our applications (the "unknown unknowns"). And it's also in the hard task of meaningful risk analysis and threat modeling and its translation into actions and priorities. These require deep security thinking and background, it's not something product development teams can be asked to take on as added cognitive load on top of their existing responsibilities.

The work that Matthew Skelton and I have done researching, cataloging and explaining how different DevOps Topologies can play an important role in progressing or blocking DevOps adoption fits nicely with the problem at hand.



Two distinct (and possibly complementary) team topologies come to mind:

A) Fully Shared Security Responsibilities

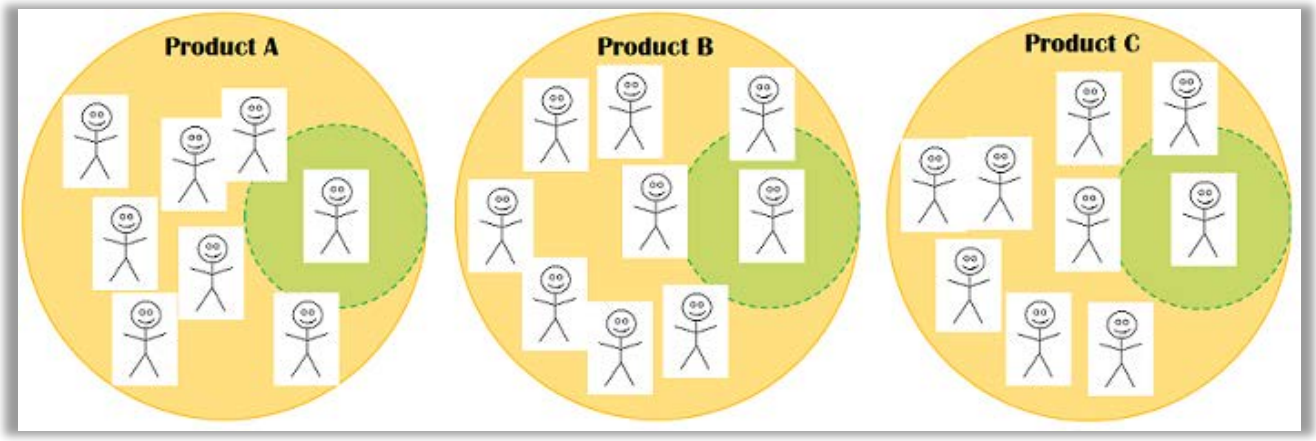
B) Security as an Enabling Team

What are the differences, the pros and the cons of each approach?

Fully Shared Security Responsibilities means that each product development team integrates at least one security-focused T-shape team member. This approach is adequate when the organization strives for cross-functional autonomous product development teams. The security person needs to be able to translate complex security threats into actionable security stories and acceptance criteria that the team can understand and implement from a technical standpoint.

They also must be able to communicate risks and potential costs for the organization (compliance, revenue and reputation) in a way that business owners can understand and prioritize. On the other hand, they should be ready to perform non-security related tasks when needed. Over time, ownership of security analysis and implementation should spread in the team while the T-shaped security person might maintain expertise in terms of staying on top of security best practices, new methods and tools, facilitating security-related workshops and leading product security strategy.

The goal is not for the security person to be allocated all the security work, otherwise we are just moving a silo at a macro level (isolated security team) to a micro-level (isolated team member).



Each product team - depicted as a yellow circle - with 7 to 9 team members, where at least one is a T-shaped security person - depicted inside a green circle - but other people also participate in security work.

Of course, there is still an important place for a centralized view of security across multiple products and business areas in the organization. Sharing experiences, approaches and results is critical. But this can take the form of a community of practice or a guild (in Spotify *parlance*) of motivated individuals gathering on a regular basis, rather than a dedicated team.

Pros

- **Team security ownership will rise**, leading to faster (non-trivial) security incident resolution and, perhaps more importantly, learning from them to avoid repeating the pain in the future.
- As *teams have a natural size limitation* (8-9 people), the **ratio of IT staff to security** (T-shaped) staff should move **much closer to 10:1 over time** with this topology. The diversity in technical background alone will bring benefits in terms of approach to problems and priorities.
- The move to this topology will send out an undeniable **signal to everyone in the organization that security is now a first class citizen** in our products.

Cons

- The **cost of finding and recruiting** (which might require generous packages) and ramping up those extra 9 security persons (in an org with 100 developers) will be a serious issue (as per the intro to this post). Expect this transition to take a considerable amount of time, during which you need to have a mix of models in place (some autonomous product teams and some still depending on a centralized team), and think about a strategy for selecting which teams go first. For example, start by assigning the more experienced security folks to the teams working on the riskier products in your organization.
- High-performing autonomous teams are based on stability and knowing each other's strong and weak points. **Any change to the team composition**, even a single person, will inevitably **cause some disruption**. The team might feel it is delivering less and becoming less productive while taking in the new security angle. It is crucial that everyone understands this is normal and accepted.
- **Lack of a centralized contact point for security concerns**. Senior managers, in particular, might feel there is "no one at the wheel of security" in a decentralized structure. Making sure there are communication channels and people able to direct requests for security information to the right teams (or to a community of practice) can help here (although that could work as well, if you have the means).

Heuristics

- Does your organization already have **cross-functional teams in place**, integrating business owners and owning responsibility for QA and monitoring and running the applications they develop?
- Do **products** (or services) display **very different risk profiles**?
- Do **products** (or services) run on **heterogenous tech stacks and infrastructure?**
- If the answer to one more more of the above questions is affirmative, then this topology might prove suitable to address the security gap.

Security as an Enabling Team means a dedicated security team provides support and guidance (for example, producing secure development guidelines) to product development teams.

Security as an Enabling Team means a dedicated security team provides support and guidance (for example, producing secure development guidelines) to product development teams.

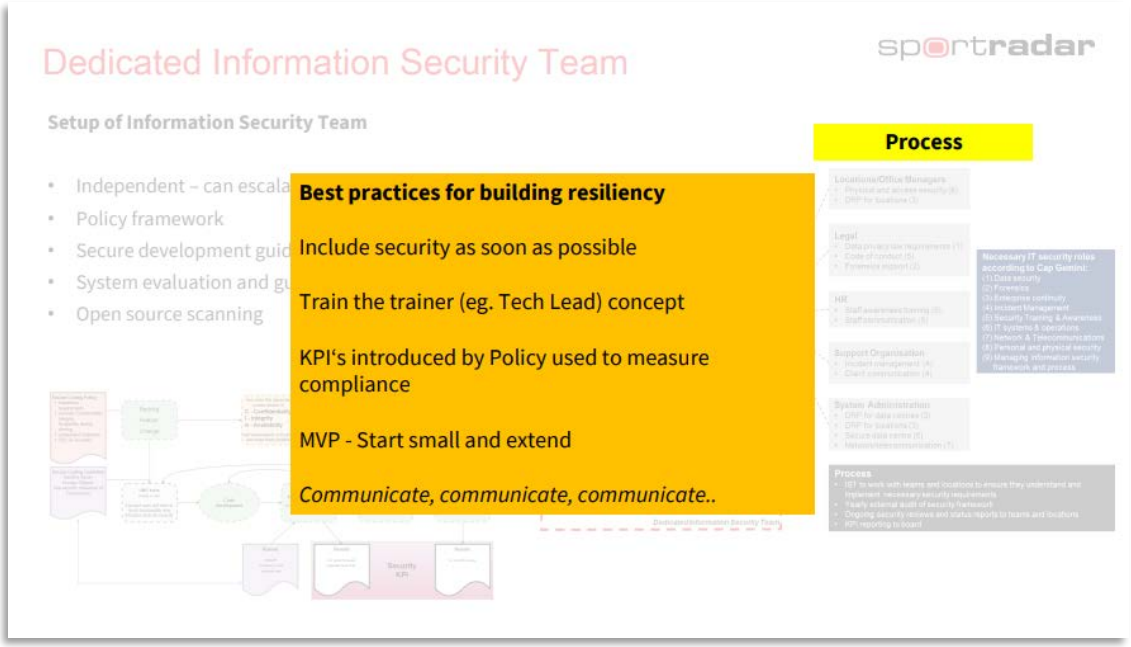
Crucially, this centralized team does not perform the actual security analysis and implementation work, instead promotes and guides it where necessary.

It's equally critical that this team gets involved from the very start of the project or release in order to help the product team consider the security implications, approaches and practices at each stage of the lifecycle.

A transversal security team - depicted vertically in grey - supports three product teams - depicted horizontally in orange - resulting in shared responsibility for security - depicted as a light green circle

Many organizations have taken this approach, including Spotify and Sportradar. It requires a much less drastic structural change from the traditional "security team silo", as we're essentially shifting this team's responsibilities (guidance and support rather than implementation). But that might make it harder for the rest of the organization to fully realize that product teams are expected to take greater ownership of the security of their systems.

Pablo Jensen, CTO at Sportradar, recently talked about their dedicated information security team's role of promoting security guidelines and policies in close collaboration with product teams, listening to their feedback and iterating over time. One of their project lifecycle gates is a "fitness to start development" which requires approval by the security team that enough thought has been given to security implications and work planned accordingly. This team reports directly to CEO and has authority to stop product launches if necessary.



Role of a centralized security team providing guidance rather than doing the product security work Credit: Pablo Jensen, CTO at Sportradar (slide from his QCon London 2018 presentation)

Pros

- **Shorter timeframe to move to an enablement model** from a traditional isolated security team doing the bulk of the security work.
- **Will not require a large number of new hires**, thus avoiding all the associated effort and potencial team disruption. The focus here should be on making sure the enabling team as a whole has all the soft skills required, on top of technical skills.

Cons

- This **team needs to master effective communication**, which in itself is a good thing. This is a skill that needs to be honed over time, thus initially investing in external or internal help to define communication strategies and curate content can be required.
- Moving to this model sends out a **weak signal about the change in security focus**. Signal needs to be amplified, and might require repeating for a long time until it sinks in as part of the org culture.
- **Introducing new responsibilities, practices and tools can be challenging** for product development teams already at the peak of their capacity. The centralized enablement team might get overwhelmed with requests for help and be put under pressure to help with product security implementation, which would defeat the entire purpose of the model.
- Over time the **security focus inside product teams might fade away** without a security person participating in the team's release/sprint planning and daily standups. Some kind of governance will need to be put in place to avoid this effect.

Heuristics

- Is there a small number of product development teams (allowing for closer collaboration with the security team)?
- Do the product team members display interest in security topics and appetite to learn (for example attending technical conferences or meetups)?
- Does the resolution of security related incidents in production naturally involve both security and development people (as opposed to a blame game where each side shies away from responsibility)?

If the answer to one more more of the above questions is affirmative, then this topology might prove suitable to address the security gap.

Conclusion

DevSecOps has raised the profile of security in IT and the regular stream of serious data breaches has exposed the security gap in most organizations. In this post I presented a couple of possible approaches to bridge that gap (fully shared security responsibility vs security as an enabling team), along with their pros and cons, and heuristics based on context.

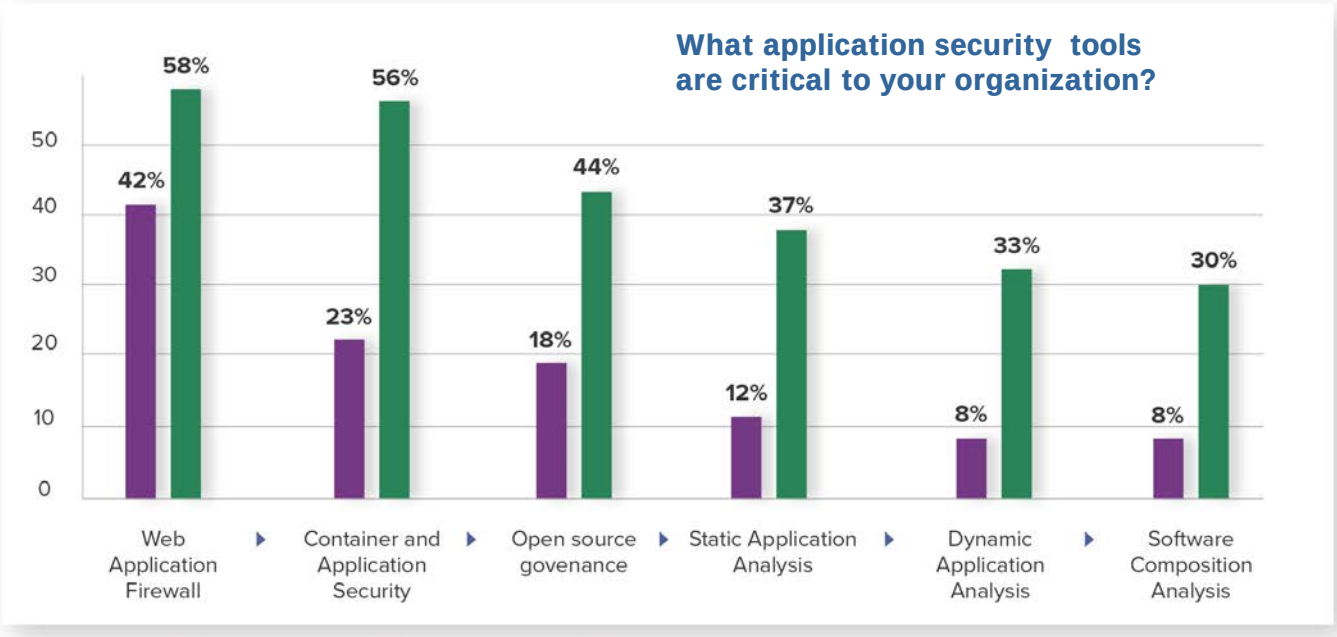
Remember team design should not be static. Organizations, people and processes evolve naturally over time and what today's best fit might be tomorrow's obstacle to faster and safer software. It's quite possibly an organization would first move from a traditional security silo approach to a "security as enabling team" model at first and over time transition to a "fully shared security responsibilities" structure, as security awareness and expertise spreads across the product teams.

What other team topologies and security strategies have you seen in your organizations or clients? Please email me at: me at manuelpais dot com.

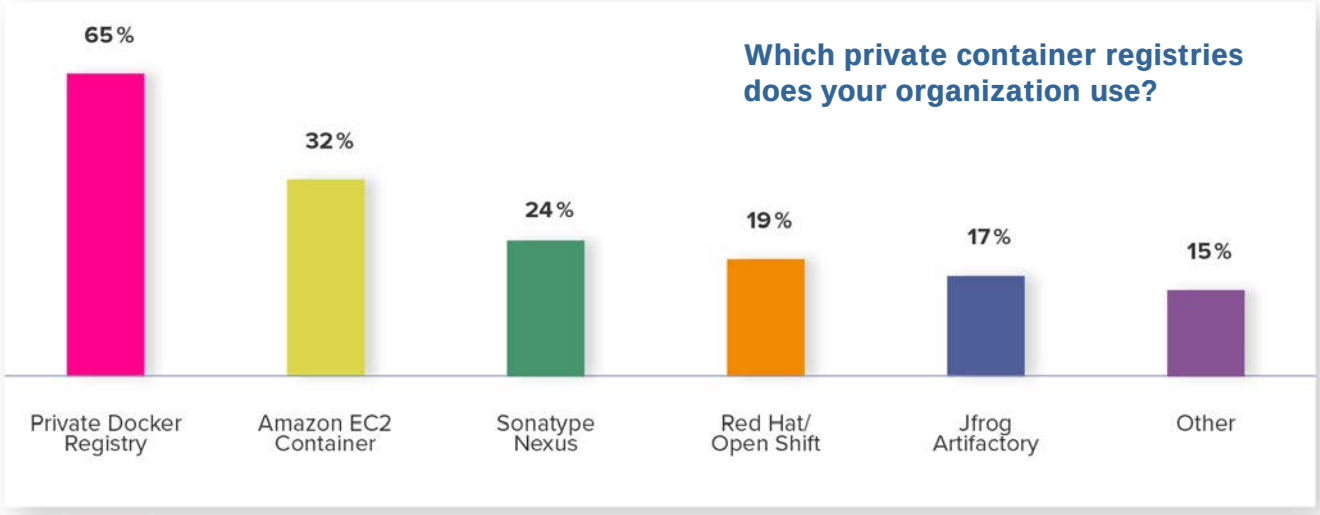
DevSecOps and Containers: The Numbers Don't Lie

Chris Short, Sr. DevOps Advocate | SJ Technologies

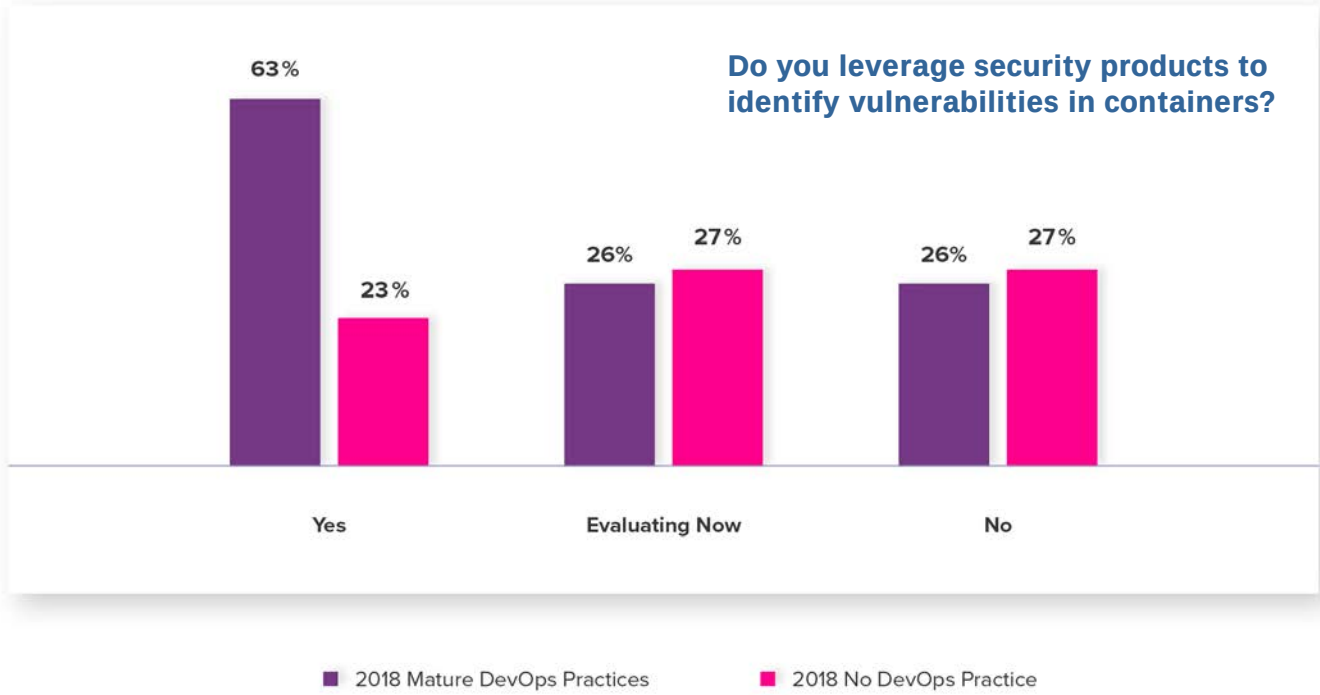
Looking at year over year data for the DevSecOps Community Survey, the percentage of respondents stating Container and Application Security tooling is in use ***doubled***. In 2017, only 23% had tooling in place wherein 2018, 56% responded as having Container and Application Security tooling in place. Container Security is quickly becoming a segment ripe for standardization and simplification. Given the recent explosive growth of Kubernetes and the creation of new container runtimes in the past year, this should not come as a surprise.



What is surprising is the fact that most organizations were using more than one container registry. Most respondents are using a private Docker registry followed by AWS ECR and Sonatype Nexus. Red Hat OpenShift and Jfrog Artifactory were well represented too. One can imagine the many ways container registries could be utilized. But some registries are very different than others. Implementing security tooling with many registries could make for a convoluted pipeline if not thought through. Thankfully, most registries implement common APIs allowing for this.



When asked, "Do you leverage security products to identify vulnerabilities in containers?" almost half of all respondents are using security tooling to identify vulnerabilities in containers. When factoring in only results from respondents considering themselves part of "Mature DevOps Processes" almost two-thirds are scanning containers for vulnerabilities. Another quarter are evaluating container security tooling. Yet a quarter of all respondents stated they have no solution for identifying vulnerabilities in containers.



That's somewhat terrifying in-light of recent breaches. It could be there are two camps here. The first camp could be teams that are using containers tagged "latest" or are continuously pulling from an assumed patched upstream. This isn't a good practice as organizations are trusting upstream to do security for them. The other, and more likely, camp are organizations that don't have a solution for identifying vulnerabilities in containers. Given the adoption of containers, not scanning them for vulnerabilities leaves a dump truck size whole in security processes.

Securing Containers

There are some very common misconceptions about container security. Liz Rice of Aqua Security points out, "I'd say the most common misconception is that containers are more like VMs than they are — and then being disappointed at the level of isolation they give you." Containers are not virtual machines. With VMs, teams worried about the hypervisor and the OSes running underneath it. With containers the model shifts to worrying about the kernel, container runtimes, and every package and framework in use in every container (in addition to the OS and hypervisor). With containers, development becomes drastically simpler and more consistent. But, the number of security vulnerabilities needing to be addressed typically goes up. Luckily, good container security processes, allow for rapid mitigation of vulnerabilities. Containers require a different approach to security than VMs.

Processes

Container lifespans should be much shorter than those of traditional VMs. Containers should be treated as immutable infrastructure. This is a mind shift as much as it is a change in infrastructure. Containers should not be patched; they should be recreated. Updating container images and redeploying all affected containers is how container security is best applied. If the pipeline is built right, this process could be as simple as changing a tag in a configuration file.

Scanning for vulnerabilities in containers that are actively running is critical. But, as DevSecOps practitioners, always be shifting left. Scanning container registries for vulnerable images is a critical function as well. Scanning containers via version control actions should also be practiced. Containers should be lightweight enough to be scanned at every touchpoint. If there is a tool at your disposal to check for vulnerable containers running in a dev environment, use it! Scan early and scan often.

Tooling

The container security tool ecosystem continues to grow. One thing to keep in mind when building out a container security toolset, there is no one right way at the moment. Development pipelines are based off business needs. Implementing container security tooling will need an equal level of thought.

There are a number of open source tools available to help lockdown containers. All of them cannot be covered in this article. But, it wouldn't be a useful article to without discussing a few tools. Using tools like AppArmor and Seccomp is highly encouraged. Both are components of the Linux kernel and provide sane defaults. AppArmor applies Mandatory Access Controls to running programs (like Docker itself). Seccomp restricts the actions (syscalls) available within a container. AppArmor and Seccomp provide the minimum viable protection for systems and containers should a container become compromised. Neither will tell you that a piece of software contains vulnerabilities.

Several container registries offer a scanning tool. But, if those don't cut it there are other options. CoreOS offers a tool called, Clair. Clair is an open source project for the static analysis of vulnerabilities in appc and docker containers. Sonatype's Nexus Lifecycle analyzes the application layer of containers for known open source vulnerabilities. Sysdig Falco is a behavioral activity monitor designed to detect anomalous activity in applications. Dagda (which incorporates Sysdig Falco) is a tool to perform static analysis of known vulnerabilities, trojans, viruses, malware & other malicious threats in Docker images/containers. There are also CIS Benchmarks for Docker and Kubernetes (in addition to operating systems). Needless to say, there are numerous tools available to help secure containers and container environments.

Conclusion

The results of this year's DevSecOps Community Survey shows that container adoption will continue to grow. The need for greater visibility will increase as more containers enter an ecosystem. Container orchestration tools like Kubernetes will be adopted to help manage some of this need. As more software is layered into an ecosystem, more automation will make management less challenging. Automating security tooling into container based workflows will become a critical piece of every major organization's security posture. Remember, always be shifting left.

More than 10 million software developers rely on Sonatype to innovate faster while mitigating security risks inherent in open source. Sonatype's Nexus platform combines in-depth component intelligence with real-time remediation guidance to automate and scale open source governance across every stage of the modern DevOps pipeline. Sonatype is privately held with investments from New Enterprise Associates (NEA), Accel Partners, Hummer Winblad Venture Partners, and Goldman Sachs.

Headquarters
8161 Maple Lawn Blvd, Suite 250 Fulton, MD 20759 United States – 1.877.866.2836

European Office
1 Primrose Street
London EC2A 2EX
United Kingdom

APAC Office
5 Martin Place, Level 14 Sydney 2000, NSW Australia

Sonatype Inc.
www.sonatype.com