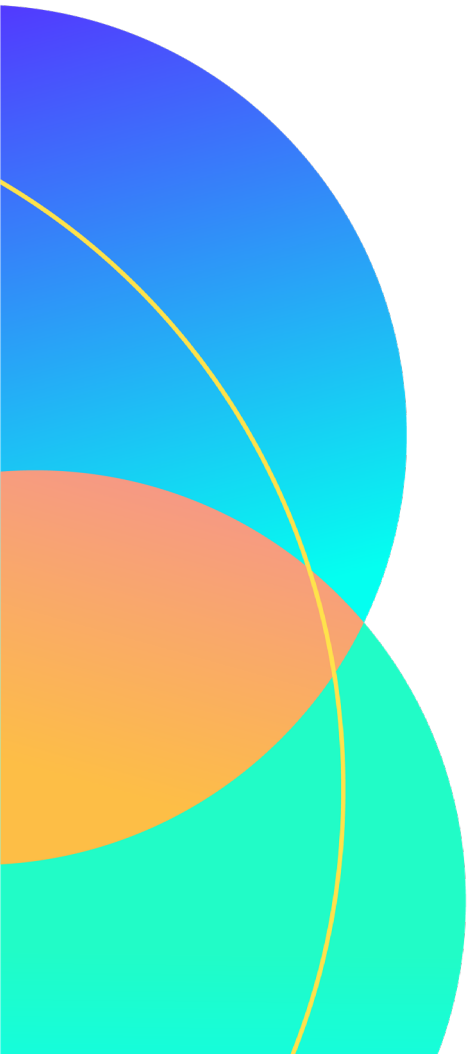


Cybersecurity Strategy Must Include Both Shift-Left and Shield-Right Approaches

Organizations adopting modern technologies such as cloud computing, containers, and infrastructure as code (IaC) are experiencing profound competitive gains and Capex savings. The drive to increase their digital footprint, however, has created some security gaps. Application security can no longer be the responsibility of just one department.

Of course, this is often easier said than done. A cloud native approach can improve developer speed and agility, but 41% of cloud engineering and security professionals surveyed cite agile methodologies as a major impact to their cloud security efforts because they create more complexity. The survey also found that 45% of the responding organizations building cloud native applications suffered from an incident resulting from a known vulnerability.

Agile methodologies used by DevOps teams make traditional security approaches untenable. The rapid pace of development leaves little time for traditional waterfall approaches that place security testing at the end of the software development cycle. Instead, many organizations often start with runtime solutions that look for security vulnerabilities in production. That's because these solutions are perceived to be easier to implement and operate, essentially building the engine while in flight.

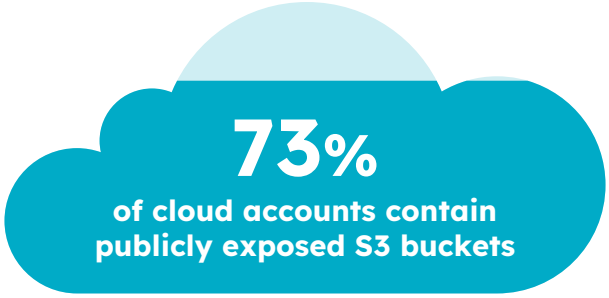
To eliminate security issues pre-deployment, teams have to add specific expertise related to cloud-native security, set up additional training and education, and "shift left" on security, moving the security review processes and tooling to earlier design and development stages. The problem is, developers aren't security experts and must focus on delivering business functionality. They're looking for automated software composition analysis (SCA), especially as open-source software has become ubiquitous in development. Today, between 70% and 90% of modern software applications contain open-source software, according to the Linux Foundation and Snyk.

However, not all issues can be addressed prior to delivery. You can't test for issues that are entirely novel, or the unknown. There's often latency between identifying an issue and when it is ultimately fixed. Code may also be owned by third parties, which raises concerns about who's responsible for fixing found issues. A shield-right security approach, which is preventing or mitigating attacks while the software is running, is equally critical to prevent or mitigate attacks and enable digital forensics and incident response (DFIR). Runtime security underpins all information security and cybersecurity programs.

DevOps teams must bring together people, processes, and technologies to create a perpetual cycle of security and future-proof their digital estate. There has never been a more critical time to revisit cybersecurity strategy. DevOps teams are primed for even more rapid innovation, but this is also while disruptions in the supply chain, economy, and global peace persist. And with faster innovation comes a greater vulnerability backlog. Risk of ransomware attack ranks high for security leadership, and these attacks perpetuate through exploitation of known vulnerabilities. Gartner estimates that by 2025, at least 75% of IT organizations will have faced one or more ransomware attacks.

Source: Sysdig 2022 Cloud-Native Security and Usage Report

About 75% of containers are running with high or critical severity vulnerabilities that could be patched, and 73% of cloud accounts have publicly exposed S3 buckets, putting sensitive data at unnecessary risk.



Source: Sysdig 2022 Cloud-Native Security and Usage Report

A balance of both shift-left and shield-right approaches should be the goal of every **security program** for full lifecycle security. Here's why.

What is a shift-left approach, and what can happen without it?

A single bad line of code can have a ripple effect throughout an entire project. The same is true for a single security vulnerability. A shift-left approach addresses these security issues early in the development process to be able to identify, manage, and eliminate them at their origination point before deployment. Traditionally, a shift-left approach goes something like this:

Application security testing begins with the code. Code security tools for software composition analysis (SCA) and static application security testing (SAST) help analyze code and dependencies to spot issues early in development. They are the top two tools used to address security concerns, according to a recent study from the Linux Foundation and Snyk. Dynamic code analysis plays an equally important role and involves running code and examining the outcome, including testing possible execution paths of the code.

Once code is in production, there should be a feedback loop from issues discovered in runtime to the underlying code. Interactive analysis tools, informed by runtime security, allow teams to respond to vulnerabilities in real time, make changes, or give instructions. Not surprisingly, DevOps teams prefer scanning that can be fully automated as part of continuous integration/continuous development (CI/CD) builds.

Tools have emerged that deliver a developer-friendly experience and actionable remediation guidance. These solutions are fundamental to DevSecOps practices, executing automated tests early and presenting results to developers in the context of their workflows and pace of development. These comprehensive testing processes within development phases provide a solid foundation for a smooth production release, but cloud-native environments add several extra layers of security complexity.

For starters, security test automation is challenging with respect to processes around test data management and operationalizing open-source test automation tools like Selenium scripting. And rarely do working test environments mirror actual production environments, which can have an unintended side effect of invalidating test results.

Security teams also often lack visibility into all code and potential vulnerabilities due to the sheer number of code sources used, version tracking and management tools introduced, and integration points that must be strategically selected.

DevOps teams often use third-party and open-source software but they don't control the code, so fixes may be out of scope. Additionally, many code sources are dependent on each other, and these dependency chains are often nested and complex. With so many transitive dependencies, there's likely at least one component of code that's vulnerable, though whether it's executed in runtime or exploitable is another challenge.

There are also multiple code artifact types to parse, such as application code, infrastructure-as-code (IaC), and policy-as-code (PaC), each requiring rules to effectively audit for vulnerabilities and compliance. Each code type also leads to multiple CI/CD pipelines. Not all development teams have the skill sets or a formalized software development lifecycle process to review all this code effectively.

Not all security issues originate from the code level. They can result from overall designs, application sources, infrastructure configuration, or mitigating security controls, and those aren't easily identified through scanning. Scanner efficacy varies depending on the source language and artifact type. Most security scanning tools are geared for security personas (and as we'll cover later, developers are not security subject matter experts). Some scanning tools also don't provide actionable details or automate fixes.

Code analysis challenges

Static analysis of code (without executing the application) can lead to a higher number of potential findings, which are sometimes disregarded by teams as false positives or suppressed. Static analysis also can't paint the full picture of the complete, fully-integrated system.

Dynamic analysis of code (while the software is running) is not often a great fit for API-centric architectures or those with multiple front ends, including mobile, because achieving good testing coverage is challenging. Testing functionality fully so that all aspects of code are reached is simple in theory but difficult in practice. Dynamic analyzers are also notoriously bad at detecting logic flaws since it's difficult to precisely locate the conditions in code that result in this category of flaw. Dynamic analysis is also more difficult to use in comparison to static analysis, as you need to authenticate, authorize, and feed enough data to the application to get better results and attain as much code coverage as possible.

Interactive analysis requires some type of instrumenting agent to function properly, such as an application runtime agent, container runtime agent, or web proxy.

It's difficult for any subject matter expert to gauge the relative security risks in a flood of findings, but even more so for developers who are focused on delivering functionality. They are not experts in triaging findings or prioritizing remediation. There's also the impact of figuring out how to even fix things. Many tools don't provide actionable remediation advice, but rather just a "laundry list" of vulnerabilities. Release velocity can take a hit if security testing isn't efficient and thresholds aren't set for scanner output, directly impacting the organization's ability to meet business objectives. Developers need SAST and SCA tools that provide more than just a laundry list of vulnerabilities. The tools have to provide actionable remediation advice on how to fix problems.

Even a perfectly designed, developed, and deployed runtime system is still prone to attack. Organizations face many other threats that are not in scope for any type of early-stage testing, like ransomware, malicious crypto mining, or runtime compromises, which is why it's critical to balance shift-left security with a shield-right approach.

Developers need SAST and SCA tools that provide more than just a laundry list of vulnerabilities. The tools have to provide actionable remediation advice on how to fix problems.

What is a shield-right approach, and what can happen without it?

A shield-right approach emphasizes security mechanisms to protect and monitor running services. Practitioners often describe the approach as runtime security, runtime protection, or runtime threat detection and response, depending on their area of focus. Such runtime capabilities are foundational for modern cybersecurity programs, as can be seen in guidance such as the NIST Cybersecurity Framework (CSF). An organization must fully identify all of its systems and code issues to understand its risk profile, which is where security testing capabilities help. And it must also focus on the other areas of the CSF: protect, detect, respond, and recover.

Runtime security approaches come in many forms. For applications and supporting infrastructure, security teams traditionally rely on intrusion prevention systems (IPS), firewalls, next-generation firewalls (NGFW), and web-application firewalls (WAF). These tools focus on protecting hosts, networks, or applications, and not so much on workload (or container) context.

Organizations leveraging cloud-native designs that include containers or serverless technology need modern security tools that support these abstracted and ephemeral computing patterns, as well as support for newer cloud hosting models like platform-as-a-service (PaaS). Security tools should offer cloud control-plane auditing and monitoring with agentless capabilities that are often classified as cloud security posture

The National Institute of Standards and Technology (NIST) offers a cybersecurity framework to help organizations better understand, manage, and reduce cybersecurity risks.

The framework consists of five concurrent and continuous functions:

- Identify: Map critical business resources and related security risks to focus and prioritize efforts.
- Protect: Implement safeguards to limit the impact of cybersecurity events on critical business services.
- Detect: Enable continuous monitoring and detection to facilitate the timely discovery of anomalies and events.
- Respond: Ensure readiness to take action to contain the impact of cybersecurity incidents.
- Recover: Develop and maintain plans to restore services to reduce the impact of security events.

These five functions empower professionals across disciplines to participate in the security lifecycle.

management (CSPM) and cloud infrastructure entitlement management (CIEM). These capabilities help with verifying cloud misconfigurations and mis-permissioned resources in cloud environments, respectively. Security tooling should also offer workload, container runtime, and orchestration engine instrumentation, often labeled as cloud workload protection platforms (CWPP) or cloud native application protection platforms (CNAPP).

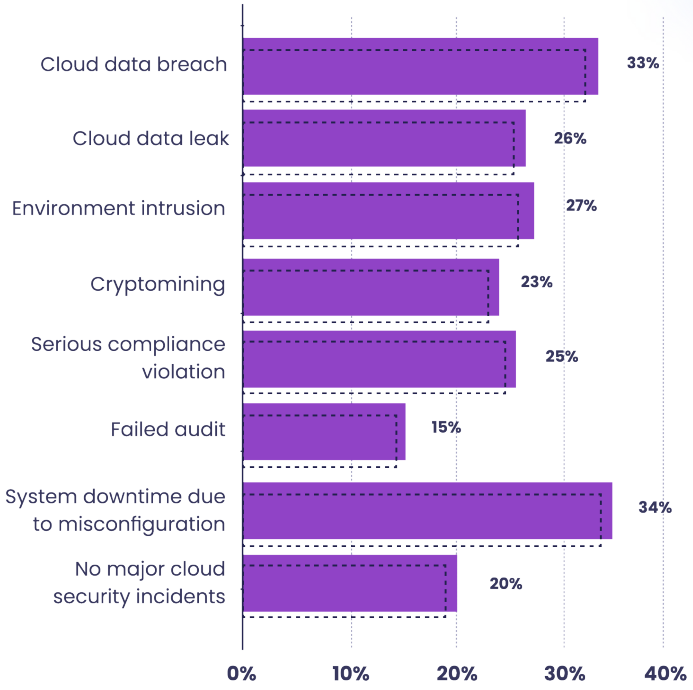
Ideally, these capabilities also provide a unifying engine to gather and correlate telemetry across environments to support modern threat *detection* and *response* in cloud and cloud-native environments. Such runtime threat detection and response is frequently powered by ingestion and real-time analysis of cloud logs, which are correlated with other service or workload activity. The end result is a more accurate picture of the organization's attack surface and how threats are impacting the organization's operating environments. This is a newer breed of capabilities best described as cloud detection and response (CDR).

But in reality, challenges arise in production that can elude the best security efforts, which is why balancing runtime security with development-stage security processes is crucial — and for similar reasons. The vast, mixed environments, cloud-hosting models, and workload types all complicate implementation and operation of security controls.

Workload visibility is worsened in containerized applications. Containers often create blind spots that can lead to security control failures. Tools, particularly those not purpose-built for container context, might not alert when security incidents or breaches occur. Or there may be compliance problems or service disruptions, any of which can lead to poor performance or downtime.

Ephemeral workloads and environments are common in modern designs. These resources last just a short amount of time, making event retention and log analysis a problem. This reality also complicates forensic investigations and incident response. Key data for diagnosis can disappear if you didn't plan for it, leaving no trail of what took place during execution.

Serious cloud security incidents experienced



The 2022 State of Cloud Security Report, Snyk

Tracing runtime issues back to the original infrastructure configuration is also problematic. Analysis of IaC can help where an organization has embraced infrastructure automation practices and generates resulting IaC artifacts. But there may be multiple IaC artifacts used to instantiate an entire application and its supporting infrastructure, all of which may be overridden by organization-level cloud configurations and settings.

Appropriate runtime security capabilities must be used for the layers of a tech stack — the application layer, workload layer, runtime layer, and network layer. Open systems interconnection (OSI) may be a well-understood mental model, but it's not an exact fit for modern architectures and security controls.

Similar to their DevOps counterparts, SecOps teams also drown in alerts. Low-hanging fruit, like the use of known vulnerable libraries, poor coding practices, and misconfigurations, leads to wasted cycles for security teams chasing problems that could've been prevented. Data can flood the organization's security information and event management (SIEM), inhibiting effective threat detection and response. And if the SecOps work is farmed out to a managed security services provider (MSSP) or managed detection and response (MDR) vendor, expect increased expense or failure to detect an event quickly, if at all.

Mind the gaps

Gaps in information present another challenge. Log data may not be retained long enough, resulting in an inability to understand issues. Cloud providers may not expose certain telemetry or instrumentation APIs, further expanding the deficiencies.

Many organizations also face a skills gap. Enabling DFIR for modern architectures is complex, and the expertise of a SecOps analyst may be limited to applications, containers, and serverless functions. A security operations center (SOC) may not even exist, or it may be distributed or outsourced to a MSSP.

Shield-right security needs subject matter experts to validate legitimate problems when issues are detected at runtime to address underlying problems in code or to configure an appropriate security mitigation. Traditionally, this requires collaboration among numerous non-security and security roles to determine what happened, how to correct it, and who's responsible for taking action. Security tooling can and should be built to account for different personas in organizations and how they respond to issues. This includes capabilities like remediation tailored to an individual's role and the application environment, prioritizations based on actual risk, and workflow integrations to serve as connective tissue between security and insecurity worlds.

Security tooling can and should be built to account for different personas in organizations and how they respond to issues.

Effective cybersecurity programs need both approaches

Adopting both shift-left and shield-right approaches for processes and tooling, also known as DevSecOps, creates a perpetual cycle of security and empowerment.

Shield-right approaches, or runtime security, help "stop the bleeding" where the organization knows it has security gaps. These gaps often arise as a result of rapidly changing, complex, distributed, and ephemeral environments. Runtime security includes detective, preventative, and responsive capabilities for issues that bypass defenses or creep into the environment even with testing.

Static security testing should be informed by runtime intelligence to help prioritize risks and understand what's truly executed or exploitable. Security tooling must provide appropriate context, including metadata about where in the cloud vulnerabilities are located — what region, cluster, and namespace — and work across workload types whether they exist in cloud or on-premises environments.

Issues found in runtime must also factor into engineering workflows to speed response and remediation. Yes, organizations should raise alerts in their security monitoring tools to notify SecOps teams and track security risks. They should also initiate defect tracking to provide a feedback loop to DevOps. Security shouldn't need to disrupt or impede the engineering workflows in order to deliver value.

Conclusion: A perpetual cycle of security

Cybersecurity programs need both shift-left and shield-right approaches to security, or DevSecOps for full lifecycle security.

Application development, infrastructure engineering, and operations have become closely intertwined as a byproduct of DevOps practices. Likewise, security must be incorporated into DevOps practices and toolchains, starting as early during coding.

With these two approaches working together, organizations can quickly detect and respond to security incidents in cloud and cloud-native architectures. These are the underpinnings of modern cybersecurity programs.

sysdig snyk