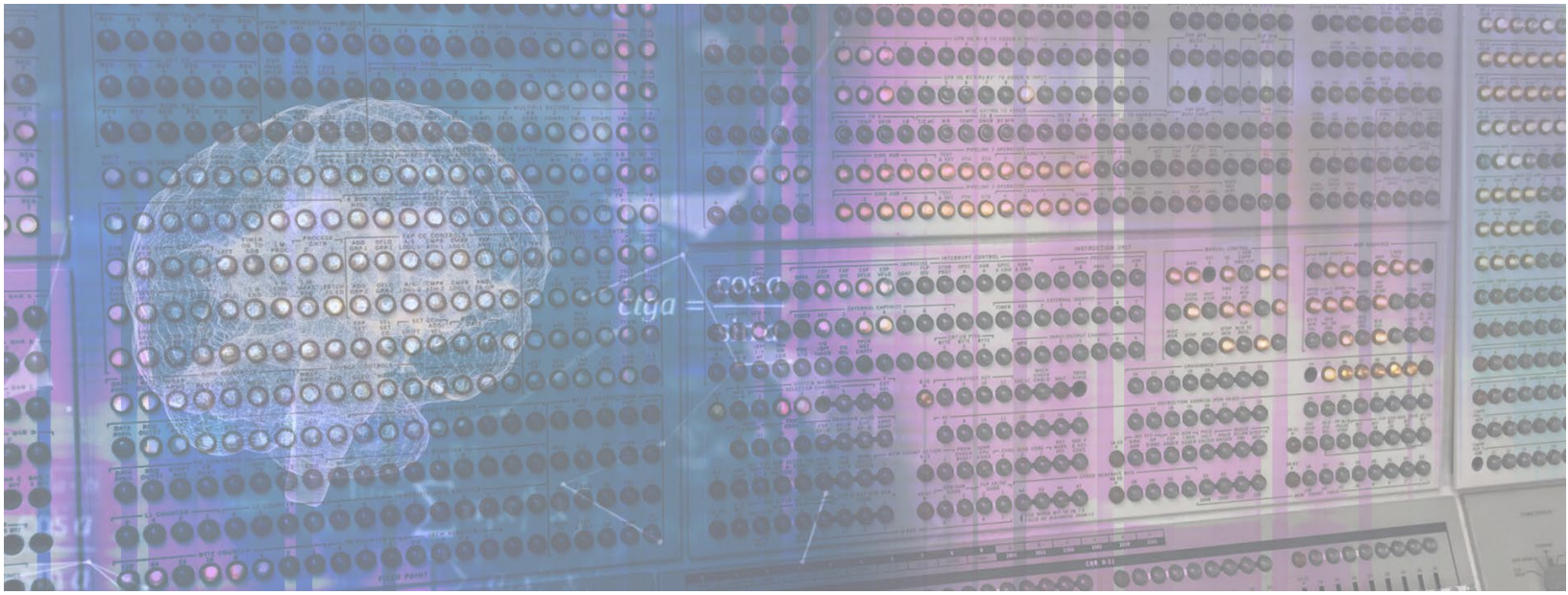


Closing the Zero-Day DevSecOps Disconnect

An Intellyx Whitepaper for Slim.AI
by Jason English, Principal Analyst
May 2023

It's all hands on deck right now for a firefight here in development.

Customer support teams are fielding customer complaints, and generating issue tickets that don't get accepted. Nobody's sure which development team or individual made the change that caused the problem. Ops teams are noticing flagging performance in some production servers. And the SoC is now reporting suspicious lateral network movement across container instances.

Who is responsible? Perhaps these concurrently surfacing incidents are due to developer error, but how can we even know where to start, if the source of a zero-day exploit could be lurking anywhere within this ephemeral, fast-changing application stack?

While fictional, this story unfortunately plays itself out in real-world enterprises every day, as the exact origins of a zero-day are difficult to uncover.

This paper will help IT leaders in development, security and operations gain a strategic understanding of software supply chain security. Getting better collaboration among DevSecOps teams can reduce the risk and damage of zero-day attacks and unexpected failures in today's cloud native and container-based delivery environments.

In the most successful organizations, the ***platform engineering team*** proactively disseminates best practices for streamlining and securing containers, through a 'service bureau' model or ***Center of Excellence (CoE)*** capability, which we will discuss below.

Readers of this paper will learn:

- The nature of zero-day attacks appearing within cloud application infrastructures
- Supply chain security tools and practices for containers within the DevSecOps lifecycle
- What defines a Software Bill of Materials (SBOM)
- How to establish a Center of Excellence (CoE) for container optimization within your organization

The zero-day challenge

There's nothing magical about a zero-day exploit—it's simply a cyberattack that the security community hasn't encountered or documented yet. But there is definitely something mystifying about how zero-days seemingly appear out of nowhere and infiltrate our applications.

There's a lucrative black market out there for zero-day exploits, with reports of buyers paying anywhere from $50K to as much as $2.5M to resell a novel attack that would impact a large target base.

Rich incentives ensure we'll continue to see hackers coming up with innovative new attacks on software supply chains.

Most cyberattacks are motivated by greed—criminals looking to steal or ransom valuable company data to resell or extort cash or crypto out of an established organization. Bad actors in this game can also include state-sanctioned hackers from places like North Korea, corporate or government spies looking for IP and secrets, or even saboteurs and script kiddies who just like breaking things.

One thing for sure: we will never have enough DevSecOps professionals to properly deal with this problem. Staffing woes only increased in 2021 by almost 25 percent to 3.4 million unfilled cybersecurity positions, meaning development teams will have to step in and help stem the threat tide.

Why cybersecurity matters to developers

The resource shortage in security inevitably trickles down to developers, who have their own priorities for delivering new digital capabilities to market.

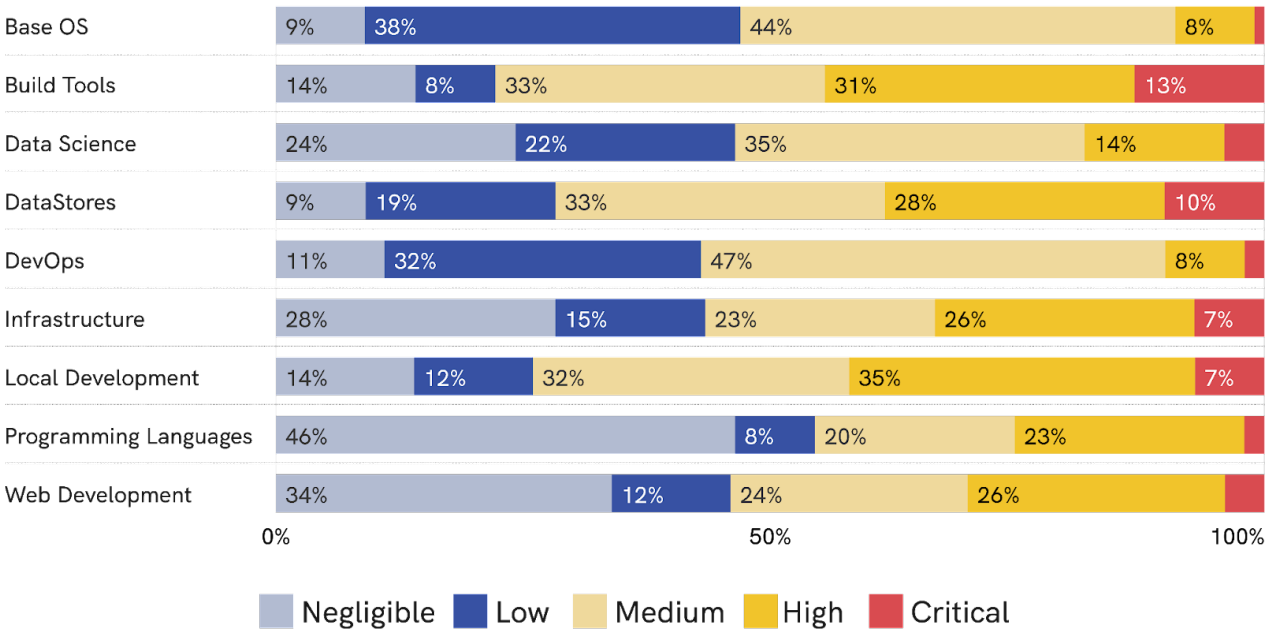
When time is of the essence, developers need to move fast, so when looking for workarounds, they will often turn to existing code and infrastructure examples that are

working for peers. Thousands of downloadable packages and source code examples are ready to go on npm and stackoverflow and DockerHub—many with millions of downloads.

The open source movement and the advent of git-style repositories has created immense value for the software industry as a whole, but by nature it also provides an ideal entry point for software supply chain (or SSC) attacks.

According to 2022 research that scanned more than 900K container images, 60 percent of the most frequently developer-downloaded images contained more publicly known vulnerabilities than the previous year. This trend is only becoming more prevalent as developers grab the resources they need for development and delivery environments.

Distribution of Vulnerabilities Across Categories



Courtesy 2022 Public Container Report, Slim.AI, p.9

Simply instructing developers to secure their own custom written Infrastructure-as-Code (IaC) against all known and unknown attack vectors won't be the answer to this quandary. Developers may be innovators or problem solvers, but they aren't usually operations or security experts.

Transforming the DevSecOps supply chain at Jit

Open source DevSecOps orchestration platform vendor **Jit** integrated Slim.AI's Automated Container Hardening flow to create a repeatable hardening process within their own CI/CD pipeline and secure customer container delivery process.

The resulting hardened containers are secure by design, both internally (thanks to Jit) and structurally, with built-in sensors and observability to greatly reduce the chance of 'unknown unknowns' appearing in the SBOM. With many containers achieving 100% elimination of critical and high-risk vulnerabilities, developers save hundreds of hours of manual vulnerability remediation per year, increasing dev velocity.

Even better, Jit also achieved much leaner and highly performant containers, with sizes consistently reduced by 30–90% and 50% faster boot times.

Zero day attacks should matter to management

Bad actors have an unfair advantage over overworked developers who are primarily expected to check in new features. Delivery speed is still prioritized over security, when the impact of an as-yet-unencountered zero-day attack is unknown.

Unfortunately, executives usually don't give software supply chain security the attention it deserves. There's not a perceived upside value to mitigating risk, until an exploit such as ransomware occurs.

Then, the picture changes dramatically, with IT management jumping in with both feet on education, compliance and additional security tooling in hopes that a future showstopper can be averted.

Even if developers are brought into the same organizational fold as security professionals into a DevSecOps team and equipped with the best security and operations monitoring tools, hackers will already know the exploits those threat scanners are looking for.

Rethinking the software supply chain

Since the catastrophic Log4j vulnerability was discovered in January 2022, the US federal government has taken serious action on securing our software supply chain—with NSA guidances and even a presidential *"Executive Order on Improving the Nation's Cybersecurity."*

Such mandates are timely: 70% of software producers recently surveyed in the 2022 Public Container Report said their customers are requesting zero vulnerabilities in addition to a software bill of materials (or SBOM) for their applications.

What makes up a software supply chain anyway?

When designing software for real-world supply chains for goods such as computers and cars, there's a combination of raw materials, supplier-furnished components and work processes (a Bill of Materials, or BOM) that have to come together at just the right time to make each finished product.

There's a cool French-origin word used in physical supply chains: ***provenance***. If you were manufacturing cars, you would need to know the provenance—or the origin and material composition of all of the parts actually going into that car, as assuring consistent quality is about more than just specifications.

Who supplied the parts? From what location were they shipped? When were they made, and who handled them in transit? Were they used before? What is the purity of the raw materials inside each part?

The SBOM offers us a 'digital analog' to the real-world supply chain, by looking at the provenance of all of the components, services and infrastructure that are encapsulated in the software supply chain.

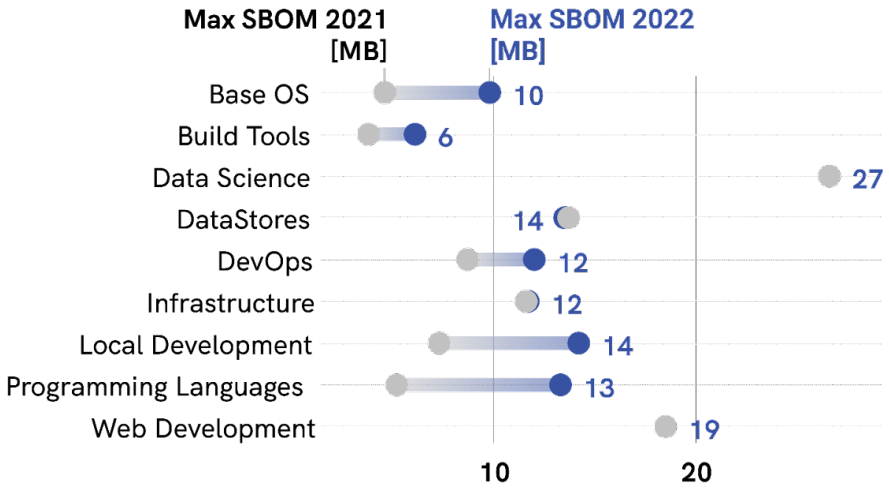
This used to be way simpler before cloud, when the SBOM might have been a couple VMs on a shared server, or even a box of CD-ROMs with an installer app that was written by the dev team. By comparison, the composition of today's distributed cloud applications is exploding with complexity, making provenance far less predictable.

The modern SBOM is containerized

When Docker appeared on the scene in 2013, containers took enterprise IT by storm. Finally the enterprise software world had a common basic unit of distribution for software and its associated file system and resources that was totally independent of the hardware it ran on. *(Yes, there were other uses of containerization in mainframes before, just not as ubiquitous…)*

Like any rapidly adopted open source project, the plan for a 'standard Docker container' soon became anything but standard over the last decade. Even today a global community of contributors, vendors and end users continue to stretch the definition of how containers are specified as code and deployed, and what goes into a container to meet specific needs.

The SBOM for a single enterprise-class container can be huge—encompassing millions of lines of code and containing an average package count of 387 according to that 2022 survey. So many source code and package variables create a lot of exposure for unknown vulnerabilities within containers, with the additional performance side-effect of enlarging containers so they are less portable and rather cumbersome to launch.



387

The number of packages the average public container has - a 14% increase compared to 2021.

Courtesy 2022 Public Container Report, Slim.AI, p.11

Supply chain observability is key

Developers insert ***application observability*** into their code to transparently reveal the condition of their running software to themselves and other engineers—an extremely valuable step for improving quality and resilience beyond what testing and monitoring can do alone.

There's a hitch though, when 90% or more of a production application's code and its CI/CD pipeline isn't actually written by the developer anymore. Who is checking all of the moving configuration code, packages, and containers from various repositories flowing through the delivery pipeline?

Software ***supply chain observability*** practices start from a 'Zero-Trust' perspective that every container or code snippet in the SBOM, whether introduced by developer or borrowed from elsewhere, could potentially contain unknown and undetected vulnerabilities.

New containers and packages are introduced every day, which should naturally cause other components to be sunset or removed. Continuous scanning across this changing software landscape is critical for detecting any unexpected changes that could portend a zero-day attack.

Visibility and transparency are also vitally important to supply chain observability, so DevSecOps teams can separate suspicious configuration code from all of the other containerized workloads passing through a speedy deployment pipeline. Since resources are limited, any vulnerabilities that have the highest potential blast radius should always displace trivial errors in priority.

Platform engineering with safe self-service

Thanks to the advent of distributed, cloud-based applications and microservices architectures, DevSecOps teams often find themselves struggling to 'roll their own' deployments from a huge inventory of containerized assets and build packages.

There's a lot of self-learning required for application teams to come to grips with the exact right configurations that would meet performance, security and compliance objectives, without wasting productive developer time or cloud costs.

Platform engineering has recently become repopularized as a pattern whereby an expert internal platform team or vendor can readily offer self-service access to tested and compliant containers—ostensible 'golden state' environments for DevSecOps teams to use—with scaling, networking and security settings optimized for the target application and infrastructure.

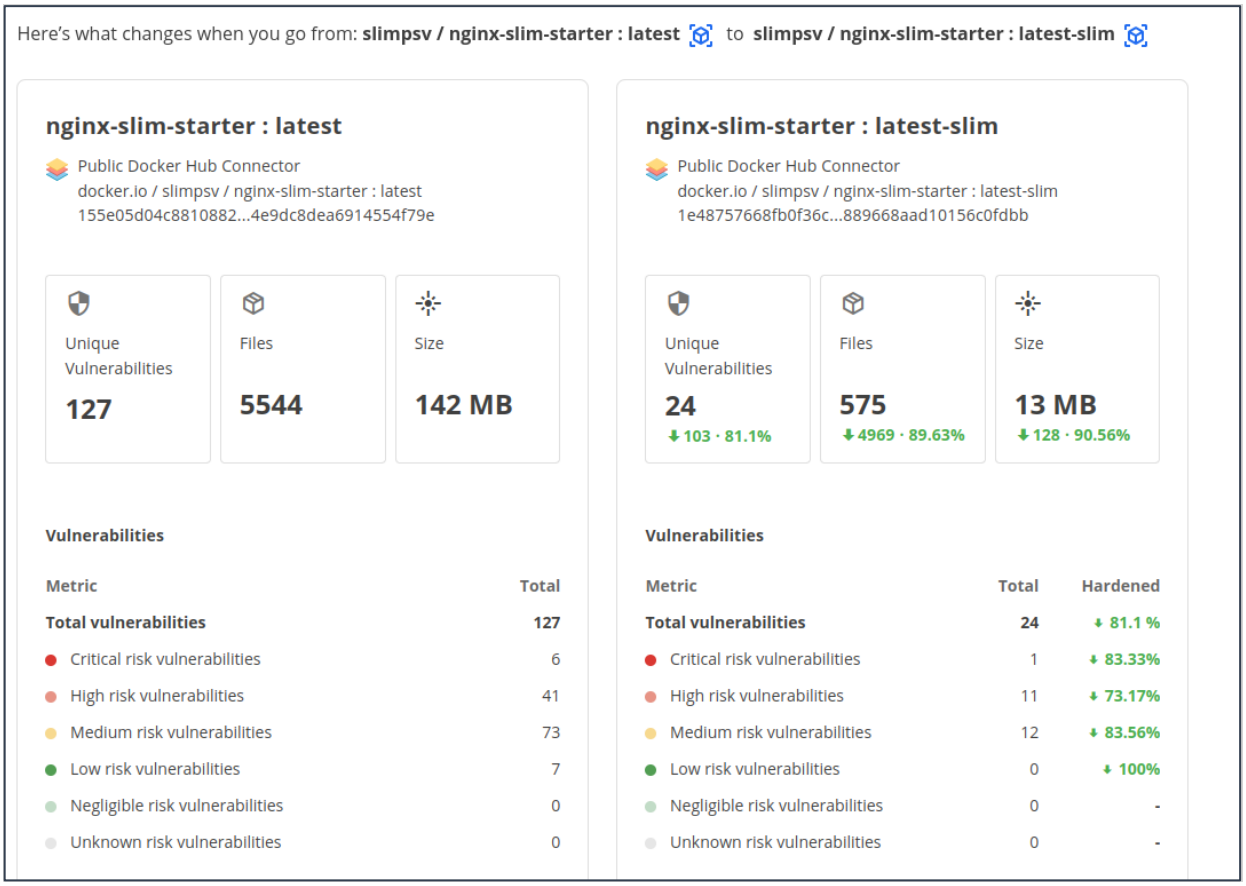
Even if the resulting golden images are not 100% pure, this still sounds like a handy resource for reducing constraints from getting started on greenfield development exercises. However, would this model support the majority of the software supply chain that is being sourced from outside the team or organization?

A CoE for optimizing containers

There's more to a successful platform engineering program than offering up a self-service portal or 'container store' type of image library for devs. The platform engineering team can also promote its own strong governance standards and best practices for streamlining and securing containers, since most of them will be inherited or imported from elsewhere.

This 'service bureau' model or Center of Excellence (CoE) capability would allow the platform team to set service level objectives (SLO) for improving and hardening container images, so they can be safely brought in from previous projects or outside suppliers for production use.

Slim.AI offers a solution for discovering, ingesting, optimizing, onboarding and observing the properties of any container image, for use with whatever GitOps, CI/CD, service and change management tools the platform team may be using.



Shown: A container before and after slimming and hardening with Slim.AI.

The resulting imported images would be enhanced with an auditable track record of their provenance, as well as observable telemetry data about how the containers were hardened for security, performance engineering, compliance or cost control auditing purposes.

Modernizing past vulnerabilities at PaymentWorks

PaymentWorks, a leading supplier management and settlement vendor, was migrating its Python application stack and CI/CD pipeline to a GitOps-based, microservices architectural approach.

They used Slim.AI to locate and reduce potential vulns in their container images, including clearing uncommon non-CVE elements of attack surfaces: shells, package managers, and even devtools like curl that are unnecessary at runtime, and could someday become exploits.

Given the importance of security and compliance to the firm's many B2B customers, having hardened containers not only gives them peace of mind at a component level, it also reduces the effort needed for future software supply chain scans and audit preparation.

The Intellyx Take

Organizations think about hardening containers for various reasons, whether reeling from a costly ransomware attack, looking down the barrel of emerging supply chain exploits such as Log4j, or (hopefully) as a way to embed preventative measures against zero-days into a microservices application modernization initiative.

Rather than thinking of preventing zero-day risks in tactical terms, invite development peers, partners and IT executives to join your team on a longer journey toward supply chain security.

A delivery platform supported by a CoE for container security and compliance can produce much more sustainable improvements in application resiliency – and a reduction in morale-destroying remediation firefights.

About the Author

Jason "JE" English (@bluefug) is a Partner & Principal Analyst at Intellyx, a boutique analyst firm covering digital transformation. His writing is focused on how agile collaboration between customers, partners and employees can accelerate innovation.

In addition to several leadership roles in supply chain, interactive, gaming and cloud computing companies, Jason led marketing efforts for the development, testing and virtualization software company ITKO, from its bootstrap startup days, through a successful acquisition by CA in 2011. JE co-authored the book Service Virtualization: Reality is Overrated to capture the then-novel practice of test environment simulation for Agile development.