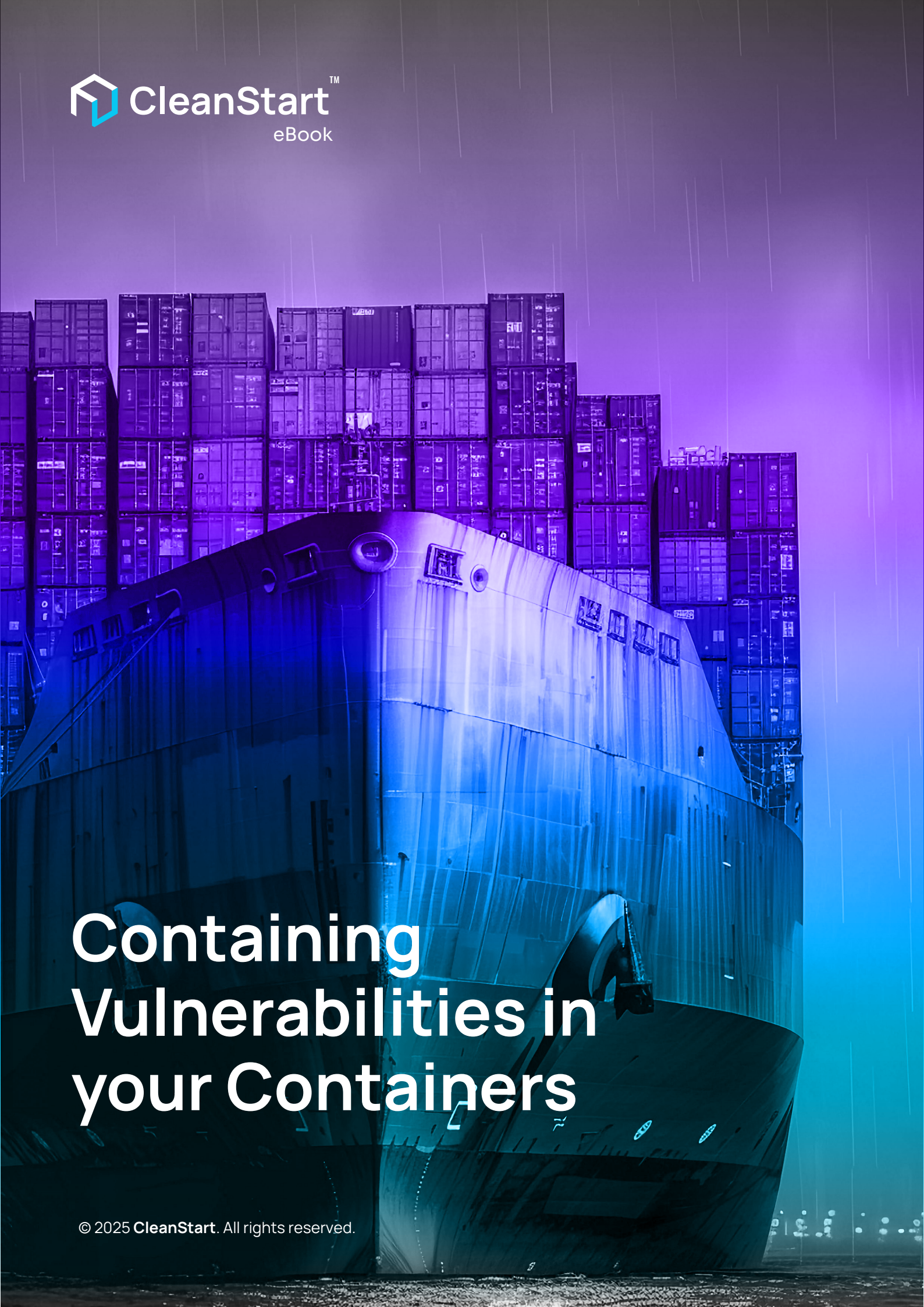
CleanStart™

eBook



Containing Vulnerabilities in your Containers

They likely never worked with container images. Or had to inspect a Kubernetes YAML file at 3 AM. In today's complex, ephemeral IT environments, knowing yourself - i.e. knowing what's inside your containers - can challenge the most ancient philosophical wisdom.

Containing container risk

Let's face it: Containerization is the best thing since sliced bread. There's a certain poetry to containers: immutable infrastructure, predictable environments, and unprecedented control. They deliver agility, portability, and consistency. Deploying and managing them is as easy as, well - eating sliced bread. They make your supply chain lean, your assets self-sufficient, and your mind at peace.

Yet, for all its advantages, the elephant in the room - a nimble elephant, but nonetheless an elephant-sized problem - is security. Few innovations have delivered so much convenience with such brazen disregard for security hygiene. Containerization has delivered new attack surfaces, cluster sprawl, and perhaps worse of all, the perception that "isolated and stand-alone" means "more secure", and that "portable and lightweight" means "less risky".

'Know your enemy and know yourself, and you need not fear the result of a hundred battles,"

Sun Tzu.

DevSecOps as central to DevOps

From hardcoded credentials to misconfigured clusters silently exposing your backend to the world, container security cannot be treated as a point solution to a point problem. It needs to become a strategic organizational imperative. Security teams need to go beyond merely keeping the lights on. Security can no longer be treated as a side quest - it's the main campaign. Container security needs to be built-in at every layer of your containerized applications from day one. This is a cultural shift as much as a technical one.

Security teams often lack visibility, bandwidth, and tooling that fits into workflows without sabotaging developer velocity. They are expected to secure containers when new CVEs appear faster than they can contain old ones.

This guide outlines 7 common container security pitfalls that keep seasoned DevOps leaders and CISOs up at night. Each comes with its own solution - best suited for its specific conditions, grounded in current best practices, guided by industry data and suited both for the "security breach virgins", as well as the dark humor of seasoned professionals who've been in the trenches too long to cry anymore.

#1 Security Risk
Lack of image visibility

Simplicity is good, but also dangerous. The average enterprise now runs thousands of container instances daily. As container adoption grows, so does the attack surface. Most security incidents start with something as simple as an unscanned image pulled from a public registry. Meanwhile, public registries themselves have become the digital equivalent of a flea market - you might find something useful, but there's an equal chance it's also carrying ancient OpenSSL vulnerabilities and three types of malware.

Security teams often find themselves in the awkward position of discovering new work not via CI/CD dashboards or logs - but from external threat intel feeds.

You can't secure what you can't see

The Solution
Continuous, Contextual, and Comprehensive Image Scanning

A container is like Schrödinger's cat: you don't know whether it hides a vulnerability until you scan it. Image scanning is your first line of defense - and should be capable of catching issues before they reach production. It should be automated, policy-driven, and enforced early in CI/CD.

Modern container security solutions scan every layer of your containers, from base OS packages to application dependencies, using real-time vulnerability databases and intelligent, context-based analysis driven by AI-ML. The best scanners in the market today can generate SBOMs (Software Bill of Materials), integrate with IDEs and registries, and even stop commits and PRs (Pull Requests) when critical CVEs are detected.

And then there are audits. Regulatory frameworks are increasingly demanding better cybersecurity postures.

US Executive Order 14028 signed by President Biden on May 21, 2021 emphasized, among other things, enhancing software supply chain security and transparency, thus making the SBOM generation a bare minimum for any container security tool.

#2 Security Risk Letting Vulnerabilities Foster

Different people have different hobbies. Some read books, some collect stamps, and some look for critical vulnerabilities in your containers. Production security incidents are expensive, stressful, and completely avoidable. Yet, as your Slack channels blow up, between service disruptions and midnight war rooms - you start realizing that "we'll patch it in production" handwaving dismissal was a more ominous decision that you bargained for.

Developers ship fast (they're hardly going to wait for your patch updates), while security teams are often behind the curve and end up trading speed for risk. All of this results in production itself becoming the vulnerability scanner - and a rather expensive one.

You can't patch it if it's already in production

The Solution
Shift-Left Security

You need to strike early and strike hard: shifting security left in your **SDLC** (Software Development Life Cycle). This means integrating security scanning, policy enforcement, and compliance checks directly into your CI/CD pipeline. Vulnerabilities can thus be caught and patched before they even reach production.

The best tools do this seamlessly and silently with minimum fuss - without affecting developer workflows. After all, if controls slow down the pipeline, developers will simply bypass them.

Recent industry reports indicate that over 50-60% of container-related breaches involved known, patchable CVEs.

#3 Security Risk
Runtime Risks

Runtime security is a challenge - but not as challenging as explaining to your CISO why your containers were hiding cryptocurrency miners. Nobody likes getting caught with their pants (or containers) down.

Just because your container started clean doesn't mean it will stay clean. Runtime attacks, privilege escalations, and lateral movement are clear and present risks. Attackers don't need zero-days when they can exploit zero-visibility.

Worse still, runtime alerts are often filtered, ignored, or buried under the noise of 'non-critical' anomalies or routine noise - until someone notices that the Kubernetes bill looks suspiciously like a crypto farm. Cryptomining containers often run in production for days because runtime alerts are deprioritized.

Recent operations such as **SCARLETEEL** and **SILOSCAPE** exploited running containers in real-world cloud environments with embarrassing ease. Most organizations have little idea what their containers are actually doing - until billing anomalies or breach disclosures provide valuable insights. As infrastructures and environments become more scaled and complex, such "Alert Fatigue" is becoming dangerously common.

The Solution
Continuous Runtime Security Monitoring

Container security can go wrong quickly if you're not constantly on your guard. Your incident response needs habitual syscall-level introspection and process behavior logging.

And for that you need the right tools. Runtime security solutions monitor your containers in real-time, detecting anomalous behavior, unauthorized processes, and potential breaches as they occur. Telemetry should be live streamed. Attackers are watching - and so should you.

> ***You can't secure what's already running***

A 2023 Devo SOC Performance Report found that 83% of SOC analysts admitted to ignoring alerts they believed were false positives.

#4 Security Risk
Image vulnerability proliferation

Same vulnerabilities, different containers: If insanity is doing the same thing and expecting different results - container sprawl is an industrial-scale ticking time bomb.

Einstein is often quoted as saying that the definition of insanity is doing the same thing over and over again but expecting different results. Well, building containers from the same vulnerable base images over and over again while expecting each to be secure - is surely the definition of a disaster waiting to blow up.

Runtime environments are not Vegas - what happens there doesn't stay there.

It's not uncommon for organizations to use base images with 200+ known vulnerabilities repeatedly to build and ship containers. It's a systemic (and avoidable) supply chain risk - and your audits can, and will, punish you for the oversight.

> ***You can't secure and ship what's already vulnerable***

The Solution
Golden image Management

Golden Image Management is the practice of maintaining a curated set of pre-approved, hardened, and regularly updated base images that serve as the foundation for all container builds. These images are continuously scanned, patched, and validated before being made available to development teams.

Thus, each container built from a Golden Image starts with a clean security foundation. By adopting such standardization, organizations eliminate recurring vulnerabilities at the source, reduce attack surface, and streamline compliance across development teams.

But it doesn't end there. Organizations also need to make deviation from golden images an auditable offense - or at least something that triggers an awkward conversation with someone higher up. If the problem occurs repeatedly, it's time to roll some heads.

Organizations spend millions automating pipelines - only to recreate the same unpatched base image across every workload. It's like scaling a leaky faucet to 10,000 replicas.

#5 Security Risk Hardcoded credentials

Secrets embedded in container images are less “secret” and more a public service announcement for attackers. Hardcoded API keys, plaintext tokens, open passwords, and forgotten SSH keys are regularly discovered in public images - not by penetration testers or ethical hackers, but by bored teenagers with GitHub dorks. It’s like leaving your keys under the doormat - while having a sign on your door that says “The keys are under the doormat”.

Even today, when security awareness is at an all-time high, hardcoded secrets are disastrously common.

You can’t secure what’s in plain sight

The Solution
Secrets Management Integration

The act (and art) of preventing hardcoded secrets is a rarity in the container security space: it is a low-cost, high-reward activity. While the risks of failure are disproportionately high, the cost of success is gratifyingly low. As in many cases, cultural inertia of developers is often a bigger risk than credential scanning.

A few best practices can dramatically reduce the chances of your secrets being auctioned off on the dark web:

- Inject secrets at runtime, not build time
- Rotate credentials frequently and automatically
- Enforce policies that scan images, environment variables, and configs for embedded secrets
- Modern container platforms can integrate natively with dedicated Secrets Management solutions. These tools store, rotate, and inject sensitive data at runtime - ensuring credentials never become part of the image layer. Secrets can then be managed, audited, and rotated independently of your application code and deployments - as they should be.

In 2024, researchers at Aqua Security found over 21,000 exposed secrets in publicly available containers - including credentials for cloud root accounts, CI pipelines, and internal databases.

#6 Security Risk
Speed versus Security

Adoption is widely considered one of the biggest challenges in the security industry. It's an industry ritual (and often a humiliation ritual) for IT teams: introduce a new security tool, congratulate yourself at a successful rollout, and then watch business users work around it with a mix of cunning and discontent because it slows down their workflows.

Slow scans, false positives, and unhelpful outputs kill adoption. Security that delays deployment is treated less like a safety net and more like an act of passive aggression.

In simple terms, if you make security so slow and painful that developers start skipping it, you've created a bigger problem than the vulnerabilities you were trying to uncover.

> *You can't deploy what slows down your developers*

The Solution
Making security integrated and native, not irritating and intrusive

Security scanning should be fast, automated, and integrated seamlessly into your existing CI/CD pipeline and systems. Modern scanning solutions provide results in minutes, not hours. They offer developer-friendly interfaces, easily understandable logs, parallel scanning, and real-time feedback. The goal is to make security invisible to developers - until there's something they need to fix.

Resentment grows when it takes 45 minutes to scan a single image - with your developers waiting to deploy.

#7 Security Risk
Lack of Automated Compliance

PCI-DSS, ISO 27001, SOC 2 - a veritable alphabet soup of regulatory anxiety is part of the job description these days. Most compliance frameworks weren't written with containers in mind, which means security teams are often left guessing what counts as "evidence" and what to present during audits. Manual compliance checks create deployment bottlenecks that make your DevOps pipeline feel more like a "DevStops" pipeline. Compliance reviews and audits can turn deployments into marathons. And we know whose heads will roll if something blows up.

You can't comply if you're not secure

The Solution
Automatic, Built-in, Embedded Compliance

Compliance needs to be treated not as an optional extra - but as an imperative necessity, and not just for clearing audits. Organizations need to map compliance controls directly into their pipelines. Non-compliant deployments must be subject to strict policy enforcement. If it can't be tested by code, it won't scale. And if it won't scale, it won't work.

Avoiding DevOops

Audit requirements and compliance can't stay in spreadsheets forever. Security teams need to automate compliance scanning with tools that map policies directly to deployment controls. Let the auditors have their evidence - without slowing down the team.

As former US Deputy Attorney General Paul McNulty once helpfully explained: "If you think compliance is expensive, try non-compliance."