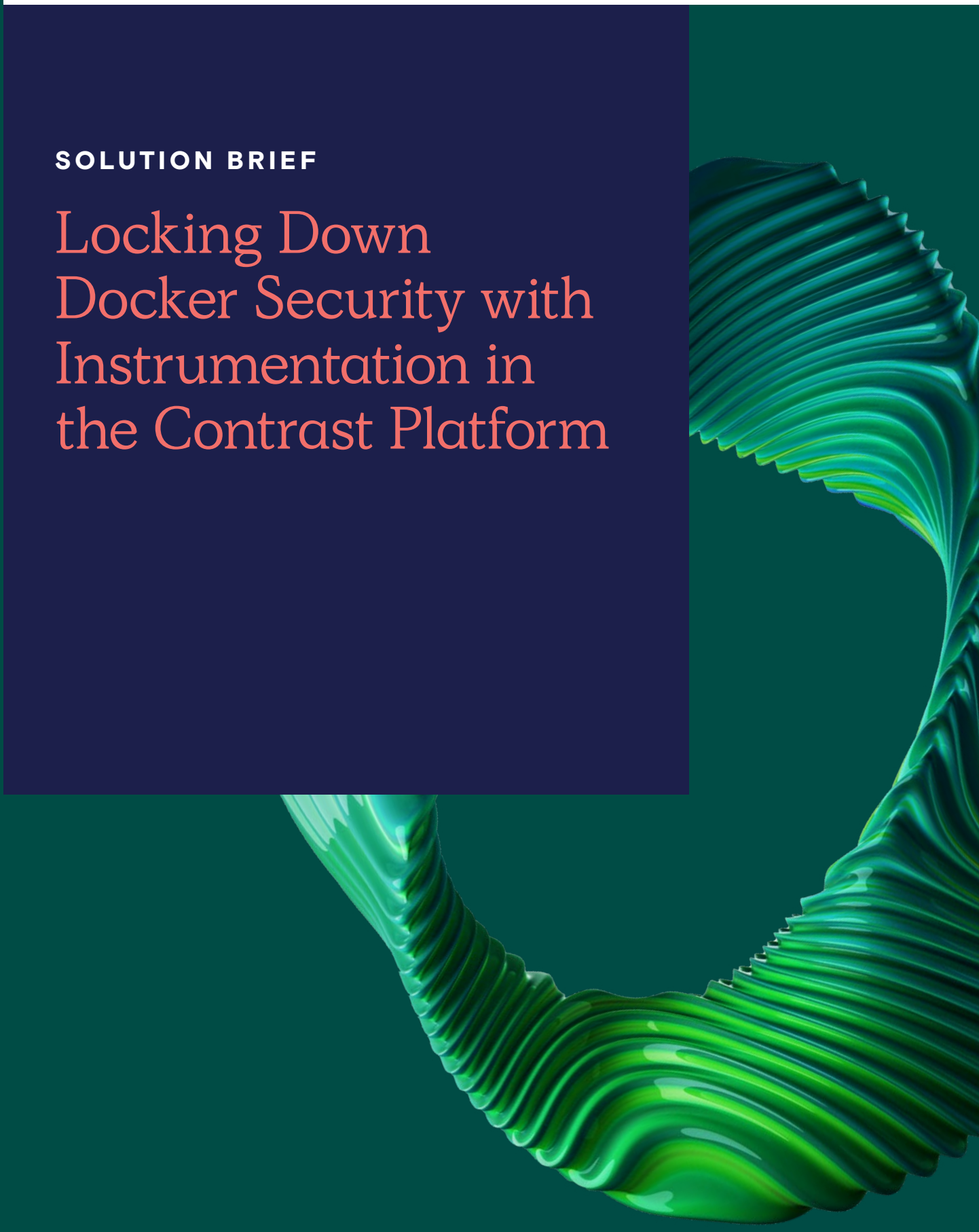
Contrast SECURITY

SOLUTION BRIEF

Locking Down Docker Security with Instrumentation in the Contrast Platform



Executive Overview

Use of containers within DevOps environments continues to grow in popularity. And while containerization solutions like Docker offer a great deal of convenience to DevOps teams, they do not provide any inherent application security and can even introduce their own particular vulnerabilities that cannot be protected by traditional means. An instrumentation-based approach to application security—such as the Contrast Application Security Platform—can address the specific problems associated with containerization in general and Docker in particular.

Containers Introduce Their Own Security Risks

According to Gartner, "By 2023, more than 70% of global organizations will be running more than two containerized applications in production, up from less than 20% in 2019."[2] Whether an application is built in the cloud, on-premises, or in hybrid environments, containerization has clear advantages in terms of scalability, portability, and continuous development and improvement.[3]

Without question, containers accelerate and simplify application deployment. But individual applications still must be assessed and protected within any container environment. The tricky part is that containers have short life spans—which means that monitoring them (especially during runtime) can be extremely difficult. And another key issue comes from a lack of visibility into ever-changing container environments.[5] And on top of complicating security effectiveness, containers introduce their own unique vulnerabilities that can cause problems across the entire life cycle of the application.

> *"Container usage for production deployments in enterprises is still constrained by concerns regarding security, monitoring, data management, and networking.[1]*

Contrast SECURITY

Therefore, in any container-based DevOps environment, organizations must solve three main problems:

1. Securing the custom code running within each container
2. Protecting open-source code and software composition that leverages multiple libraries
3. Ensuring security throughout the software development life cycle (SDLC), including rapid iterations of development and testing

DOCKER IS POPULAR, BUT NO LESS PROBLEMATIC

Docker is widely considered to be the most popular containerization technology in DevOps environments today. Between November 2019 and July 2020, Docker saw a dramatic swell in consumption—almost doubling total pulls in a little over six months (from 130 billion to 242 billion).[6] But with specific regard to Docker, misconfigurations can downgrade the level of overall application security and introduce new vulnerabilities.[7]

A popular feature of the Docker framework—Docker images—has been a main source of critical security issues. Docker images are essentially ready-made gobbets of open-source code that run services or applications, with each image containing the dependencies, libraries, and other periphery required by the code. Stand-alone images become foundational building blocks, requiring a minimum of tweaking to make them fit for purpose, which significantly reduces overall development times.[8] Unfortunately, researchers have found a significant number of security vulnerabilities within Docker images.[9]

For years now, hackers have been able to insert malicious code into Docker images on the Docker Hub.[11] In other cases, such as the core OpenJDK container, well-intentioned groups have accidentally introduced vulnerabilities into their own legitimate containers. But lack of patching is the main reason for ongoing vulnerabilities with Docker images. Many container images simply haven't been updated in the last five to seven years. This is a problem, however, as

> *The number of organizations that have containerized at least half of their applications grew by 22% over the previous six months.*[4]

> *Even in the certified channel (where Docker images receive a great deal of scrutiny), researchers found images with security vulnerabilities described as "high."*[10]

container images require continual updates and auditing to keep pace with the constant flow of new exploits and security benchmarks.[12]

Hackers also find new methods to escalate access and invoke Docker commands. Remote container command execution is especially worrisome. If left open to the internet without the proper configurations, container ecosystems become very vulnerable.[13]

There is also the domino effect to consider. A single compromised Docker container can threaten all other containers as well as the underlying host.[14] Last year, for example, security researchers discovered a cryptojacking worm that propagated using containers in the Docker Engine and spread to more than 2,000 vulnerable Docker hosts.[15]

Traditional Security Cannot Address Container Vulnerabilities

Some organizations use traditional approaches to manage Docker vulnerabilities—such as adding siloed tools to cover individual risk exposures. But securing containers in this way is a non-starter. Existing security methods are unsuitable for addressing container-based risks.[16] And even if the container itself is somehow protected by these means, the container is no longer secure once an insecure application is placed inside it.

Another previously established technique for protecting existing containers is the sidecar approach. The main container processes data while a sidecar is called at the same time to do additional processing. The sidecar makes it possible to move tasks such as logging or perhaps input scanning away from the core processing container, speeding up its time to results. However, this design pattern does not solve the need to observe container-level application flows or to access information at the container level. This would need to occur inside the container's implementing code in a way that is not visible from the outside.

In sum, while the sidecar is able to correctly perform its additional processing, the container itself is compromised while the sidecar simply watches. As a result, the main container would be hacked or tricked to return sensitive data to the attacker based on what was being exploited. And beyond security issues, use of sidecars can also quickly lead to problems with application stability and performance.[17]

QUESTIONABLE TESTING LEADS TO COSTLY REPAIRS LATER

Ineffective security testing in development also creates bigger problems downstream. Organizations are beginning to realize that many production runtime security failures are caused by missed security best practices in development. For that reason, more than half (57%) of organizations report that they are more worried about their build and deploy phases, with misconfigurations and vulnerabilities cited as areas posing the greatest risk.[18]

Cost is also a critical factor to fixing bugs after the design phase. Specifically, it is six times more expensive to fix a bug found during implementation; 15 times more if it is identified in testing; and 100 times more once the code is in production.[19]

Instrumentation-Based Application Security for Docker

To address these problems, organizations need to "shift left" and incorporate effective application security from the very beginning—during the build stage of the SDCL. This requires a rethink of traditional application security where coding must be repeatedly halted and restarted for legacy application security testing.

> *Container security should leverage the same strengths that containers bring to DevOps processes—it needs to be both dynamic and flexible.*[20]

DevOps environments will only become more dynamic and complex as time progresses. To ensure that security can keep pace with the speed of business, organizations need to embrace the need for observability across the distributed systems they build. To protect Docker containerized applications, organizations must integrate security into the application itself using instrumentation. By placing specialized instrumentation sensors throughout the code itself, organizations can gain comprehensive visibility, monitoring, and automation capabilities across all parts of the application and embed security across all phases of the SDLC.

Instrumentation-based application security delivers continuous, automated, real-time identification of vulnerabilities and verification of their remediation. This approach is effective because it operates within the container, protecting the container as well as the application or service hosted inside.[21]

The instrumentation-based Contrast Application Security Platform enables companies to align application security with their container efforts. It supports:

- Extending application security into containers without acquiring/managing more silo-based security tools
- Heterogeneous support for managing containers (e.g., Puppet, Chef, Ansible)
- Multilanguage support for each container
- Coverage of both custom and open-source code
- Fast and easy deployment

The Contrast Application Security Platform Contains Docker Security Risks

To ensure continuous deployment of Docker-containerized applications, organizations need an application security platform that extends continuous security assessment and protection across build, deployment, and runtime environments. Contrast's instrumentation-based platform achieves this by managing security from within the application itself while reducing the workflow burdens on limited staff. It also reduces costs by helping DevOps teams find and fix problems during development, while providing embedded security that scales across all parts of the containerized code—including open-source components. Finally, it helps organizations simplify their infrastructure by eliminating dependency on add-on, siloed approaches to application security.

Contrast SECURITY

[1] Ajmal Kohgadai, "Gartner best practices for Kubernetes & container security," StackRox, June 25, 2019.
[2] Robert Christiansen, "More enterprises are using containers; here's why," CIO, August 26, 2019.
[3] Ali Golshan, "Survey Reveals Rapid Growth in Kubernetes Usage, Security Still a Concern," DZone, August 30, 2019.
[4] Ajmal Kohgadai, "6 Container Adoption Trends of 2020," StackRox, March 4, 2020.
[5] Ajmal Kohgadai, "Docker Container Security 101: Risks and 33 Best Practices," StackRox, September 13, 2019.
[6] John Kreisa, "Docker Index: Dramatic Growth in Docker Usage Affirms the Continued Rising Power of Developers," Docker, July 30, 2020.
[7] "Docker Security Cheat Sheet," OWASP, accessed September 10, 2020.
[8] "Docker Hub harboring harm — research," TechHQ, June 17, 2020.
[9] "Docker Hub harboring harm — research," TechHQ, June 17, 2020.
[10] "Docker Hub harboring harm — research," TechHQ, June 17, 2020.
[11] Bill Doerrfelt, "Common Container and Kubernetes Vulnerabilities," Container Journal, August 3, 2020.
[12] Bill Doerrfelt, "Common Container and Kubernetes Vulnerabilities," Container Journal, August 3, 2020.
[13] Bill Doerrfelt, "Common Container and Kubernetes Vulnerabilities," Container Journal, August 3, 2020.
[14] Ajmal Kohgadai, "Docker Container Security 101: Risks and 33 Best Practices," StackRox, September 13, 2019.
[16] Erik Costlow, "Security Concerns Remain with Containers and Kubernetes Per New Report," Security Boulevard, March 11, 2020.
[17] Apurva Dave, "5 Things We've Learned About Monitoring Containers," DZone, August 14, 2017.
[18] Ali Golshan, "Survey Reveals Rapid Growth in Kubernetes Usage, Security Still a Concern," DZone, August 30, 2019.
[19] Mukesh Soni, "Defect Prevention: Reducing Costs and Enhancing Quality," iSixSigma, accessed April 16, 2020.
[20] Tim Ferrill, "9 container security tools, and why you need them," CSO, August 4, 2020.
[21] Erik Costlow, "Security Concerns Remain with Containers and Kubernetes Per New Report," Security Boulevard, March 11, 2020.

Contrast Security provides the industry's most modern and comprehensive Application Security Platform, removing security roadblocks inefficiencies and empowering enterprises to write and release secure application code faster. Embedding code analysis and attack prevention directly into software with instrumentation, the Contrast platform automatically detects vulnerabilities while developers write code, eliminates false positives, and provides context-specific how-to-fix guidance for easy and fast vulnerability remediation. Doing so enables application and development teams to collaborate more effectively and to innovate faster while accelerating digital transformation initiatives. This is why a growing number of the world's largest private and public sector organizations rely on Contrast to secure their applications in development and extend protection in production.

240 3rd Street
2nd Floor
Los Altos, CA 94022
Phone: 888.371.1333
Fax: 650.397.4133

Contrast SECURITY

contrastsecurity.com