paloalto NETWORKS

Container Security 101

Understanding the Basics of Securing Containers

PRISMA BY PALO ALTO NETWORKS

MediaOps

By now, it's apparent to cybersecurity teams everywhere that the proverbial container genie is out of the bottle. Developers have widely embraced containers because they make building and deploying so-called cloud native applications simpler than ever. Not only do containers eliminate much of the friction typically associated with moving application code from testing through to production, but application code packaged up as containers can also run anywhere. All the dependencies associated with any application are included within the containerized application. That makes a containerized application highly portable across virtual machines or bare metal servers running in a local data center or on a public cloud.

That level of flexibility enables developers to make huge gains in productivity that are too great to ignore. However, as is the case with the rise of any new IT architecture, cloud native applications still need to be secured. Container environments bring with them a range of cybersecurity issues involving images, containers, hosts, runtimes, registries, and orchestration platforms, which all need to be secured.

The challenge organizations will face is first understanding how the many layers of a cloud native computing environment interact with one another, and then finding the right tools to build a repeatable set of processes to secure each layer. Cybersecurity issues specific to containers include:

Images: Vulnerabilities can impact container images just like any other piece of code. Building a bill of materials, identifying any embedded secrets, classifying all the layers of an image, are all still fundamental cybersecurity tasks that still need to be addressed. Where things get complex is in the sheer number of containers running in an application environment and how frequently those containers are updated. Thanks to the rise of DevOps practices, it's not uncommon for organizations to now update containerized applications multiple times a week. Each update to what can quickly become thousands of containers running in an IT environment represents an opportunity for vulnerabilities to be introduced in that environment.

Container registries: A container registry provides a convenient, centralized source for storing and distributing application images. Today's organizations can easily have tens of thousands of images stored in their registries. Because the registry is central to the way a containerized environment operates, it's essential to secure it. A registry is critical for bringing order to potential container chaos, but it also can provide a path through which cybercriminals can easily compromise the entire environment. Continuously monitoring registries for any change in vulnerability status is a core security requirement that needs to include locking down the server that hosts the registry.

Container runtimes: The container runtime is one of the most difficult parts of a container stack to secure because traditional security tools were not designed to monitor running containers. Legacy tools typically can't see inside containers, much less establish a baseline for what a secure container environment looks like. Container runtime security issues require cybersecurity teams to focus on application security concerns not addressed by legacy firewalls.

Container orchestration: Access control to container orchestration platforms such as Kubernetes® to prevent risks from over-privileged accounts, attacks over the network, and unwanted lateral movement, need to be addressed using allow list techniques in much the same way access to legacy IT environments is handled. Where things become different within a container orchestration platform is in the need to also secure communications between pods on a Kubernetes cluster shared by multiple applications.

Host operating systems: The OS that hosts your container environment is perhaps the most important and often overlooked aspect of securing a container environment. Any compromise to the host environment provides cybercriminals with access to the entire application environment. Each host needs to have its own set of security access controls in place as well as be continuously monitored for any new vulnerabilities that might have been discovered since that host was deployed.

Given all the challenges associated with securing containerized applications, it's understandable why so many cybersecurity professionals are a little reticent when it comes to deploying containers in a production environment. While there are some clear advantages in terms of developer productivity, most organizations are only just now beginning to appreciate the tools and processes that must be adopted to secure containerized applications.

As daunting a challenge that may seem, however, containers provide one invaluable cybersecurity benefit that cybersecurity teams don't initially appreciate as much as they should: because containers wind up being ripped and replaced so frequently, the processes associated with remediating vulnerabilities becomes much simpler. Instead of having to wait sometimes months for a patch to be applied to an entire monolithic application, new functionality is introduced into an application environment by ripping and replacing containers. That process is limited to a subset of the application, known as a microservice, and can typically be accomplished within minutes as part of the application lifecycle management process enabled by a continuous integration/continuous deployment (CI/CD) platform such as Jenkins. That ability means the amount of time an application will run with known vulnerabilities in a production environment can now be sharply reduced.

That capability is arguably the driving force behind the rise of DevSecOps processes through which developers are now taking on more responsibility for implementing cybersecurity controls. The cybersecurity team still needs to define those controls, and then validate they have been implemented. However, because developers are now being held accountable for implementing those controls, the number of applications that can pass a cybersecurity audit steadily increases alongside the maturity of the adopted DevSecOps processes.

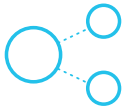
Tools of the Container Security Trade

In the last year alone, the tools organizations can rely on to secure containers have grown in terms of both capabilities and sophistication. Regardless of what level of DevSecOps maturity has been attained, container security tools are now more accessible than ever. The container cybersecurity tools any organization will be required to adopt and master include:

Container monitoring: Core to any ability to apply and maintain container security, container monitoring tools are needed to track what are among the most ephemeral atomic units of computing ever devised. Because developers are continually ripping and replacing containers, monitoring tools that enable cybersecurity and IT operations teams to apply time-series stamps to containers are critical when trying to determine precisely what happened, and when, in a containerized environment.

Container scanning tools: Containers need to be continuously scanned for vulnerabilities both before being deployed in a production environment and after they have been replaced. It's too easy for developers to mistakenly include a library in a container that has known vulnerabilities. It's also important to remember that new vulnerabilities are discovered almost daily. That means what seems like a perfectly safe container image today could wind up being the vehicle through which all kinds of malware is being distributed tomorrow.

Container firewalls: A container firewall inspects and protects all traffic entering and exiting containers as well as the traffic moving to and from external networks and legacy applications. Most container firewalls run as "sidecars" that enable them to manage a broad spectrum of traffic moving in and out of microservices made up of multiple containers.



Policy engines: Modern cybersecurity tools make it possible for cybersecurity teams to define policies that essentially determine who and what is allowed to access any given microservice. Organizations need a framework for first defining those policies, and then making sure they are consistently maintained across a highly distributed container application environment.

Defending the Hybrid Attack Surface

Now that containers are making it simpler to port containerized applications across multiple platforms, organizations will also need to be able to first enforce cybersecurity policies, and then remediate any issues that arise across multiple platforms. Most containers today are initially deployed on top of traditional virtual machines to make sure there is a layer of isolation between application workloads sharing the same platform.

However, there is also an emerging use case where organizations don't want to deploy a virtual machine because of all the extra overhead generated, which can adversely affect application performance. In these scenarios, developers will prefer to either deploy their containers on bare metal servers or on top of an emerging class of lighter-weight virtual machines. That's especially true in environments that rely on graphics processing units (GPUs) that don't lend themselves to traditional virtualization techniques other than containers. In other cases, a desire to not have to pay a fee to license commercial virtual machine software is another reason why an organization might opt to deploy containers on a bare metal server.

Whatever the motivation, the one thing cybersecurity teams can count on is that containerized applications will manifest themselves on-premises or in multiple public cloud computing environments. Each environment will consist of multiple types of virtual and physical machines running containers that will all need to be secured via a common framework.

To make matters even more complicated, serverless computing frameworks built using containers represent yet another attack surface that will need to be secured. Based on event-driven architectures, serverless computing frameworks allow developers to invoke child processes from within their applications on demand. This eliminates the need to include code within an application to run a function that is only required on an intermittent basis. The less code in an application, the easier it becomes to secure. However, cybersecurity teams should not overlook the need to secure the serverless computing framework.

The Great Cybersecurity Paradox

There are already millions of cybersecurity jobs going unfilled. As the volume of application code that needs to be secured continues to exponentially increase—thanks primarily to the rise of containers—the only way cybersecurity teams and their application developer colleagues will be able to keep pace is to rely more on automation.

Even if additional cybersecurity professionals were available to fill all those positions, most organizations would continue to find it challenging to retain cybersecurity expertise. The only way to effectively minimize the impact of cybersecurity staff turnover is to automate as many existing manual processes as possible. That approach not only makes it simpler to maintain cybersecurity policies at scale, but also enables the cybersecurity staff to spend more time on tasks such as hunting malware before it becomes active.

Going forward, it's not a question of whether cybersecurity tasks will be automated, but rather to what degree.

The Case for Unification

As cloud native computing enabled by containers continues to become more pervasive, there's a clear need for a cybersecurity framework that can be applied to containers and associated serverless computing frameworks. That argument, however, is not limited to cloud native computing applications. Cloud native computing applications are not going to eliminate all the monolithic application code deployed in enterprises anytime soon. Organizations of all sizes will be running a mix of legacy and emerging cloud native applications well into the end of the next decade. The next big cybersecurity challenge will be finding a way to build and maintain cybersecurity policies across both those environments using the same management framework.

Palo Alto Networks has already invested millions of dollars developing the Prisma™ framework to automate the management of cybersecurity within legacy monolithic application environments. We've further expanded Prisma to support cloud native computing applications based on containers and serverless computing frameworks.

In effect, we've built Prisma to be the most comprehensive cybersecurity lifecycle management platform ever made.

Conclusion

It's the best and worst of times for cybersecurity. As IT environments become increasingly heterogeneous, maintaining security has in many ways never been more challenging. At the same time, however, the rate of cybersecurity innovation has never been higher.

The most important cybersecurity decision any organization is likely to make in the coming days is determining which vendor has the tools and expertise required to secure not just existing environments, but also emerging application environments as developers continue to embrace innovative, potentially risky platforms.

To learn more about how to secure these environments, please visit us online.

paloalto NETWORKS

3000 Tannery Way
Santa Clara, CA 95054

Main: +1.408.753.4000
Sales: +1.866.320.4788
Support: +1.866.898.9087

www.paloaltonetworks.com

container-security-101-ebook-100720