GUIDE TO MODERNIZING TRADITIONAL SECURITY: The Advantages of Moving a Legacy Application to Containers

Twistlock

The Leading Cloud Native Cybersecurity Platform

UNDERSTANDING “LIFT AND SHIFT”

As containers become *the* way to ship and run applications, one trend we’re seeing more of is customers taking a “lift and shift” approach to moving their existing apps into containers. This allows security and devops leaders to immediately begin reaping some of the benefits of containers, like greatly simplified testing and deployment, without having to rebuild the entire app. This approach isn’t surprising - major waves of computing disruption depend on a combination of new capabilities and backwards compatibility to become mainstream. So, it comes as no surprise that we see our own customers taking the J2EE and .Net apps that they’ve invested tens of thousands of man hours creating and maintaining and putting them directly into containers with minimal changes to the apps themselves.

This initial step doesn’t mean that they’ll never be rearchitected and decomposed into microservices, but there’s immediate benefits that can be gained by just moving them over as-is. This is the same pattern we saw with public cloud, in which organizations often migrated entire VMs from their own datacenters to AWS without change, and virtualization before that, when organizations would convert physical servers into VMs. In each of those cases, the initial migration enables organizations to build experience and momentum while preserving their existing investments. This in turn accelerates the adoption of the larger scale changes virtualization and cloud enable because the ‘legacy’ environments become easier to manage, freeing time and focus to adoption of the new stack. Seeing this trend repeat itself with containers is a good indicator of the growing momentum of the container ecosystem.

BENEFITS IN PERFORMANCE AND SECURITY

As organizations modernize their traditional apps and run them in containers, they can immediately gain meaningful security advantages before they ever change a single line of code within the app itself.

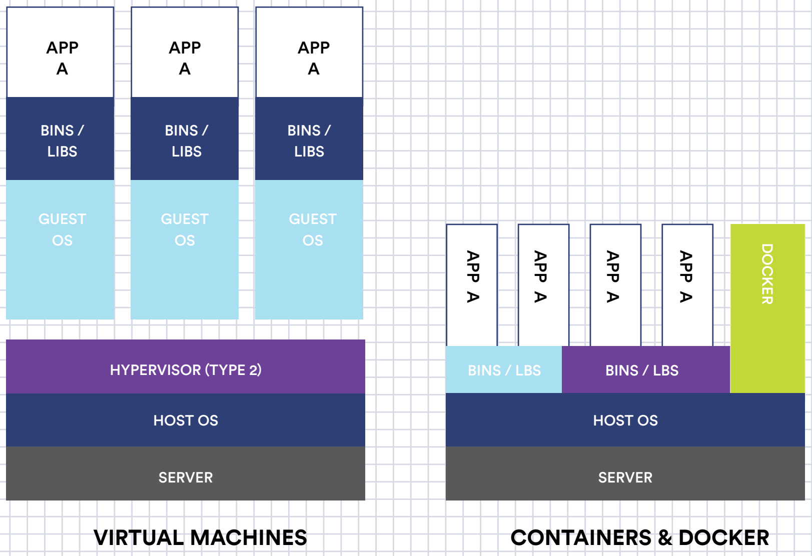
Customers that move traditional apps to containers gain a number of benefits around test automation, deployment, density, and efficiency - but one of the most powerful benefits is security. We’ve often talked about how the fundamental attributes of containers - their minimal, declarative, and predictable nature - enables platforms like Twistlock to automatically learn expected application behavior, hunt for anomalies, and defend from threats with minimal human involvement. As organizations modernize their traditional apps and run them in containers, they can immediately gain meaningful security advantages before they ever change a single line of code within the app itself. Let’s take a look at a real world customer example to illustrate how this modernization delivers pervasive security benefits across their app.

A CUSTOMER EXAMPLE

One of our customers provides environmental science and engineering consulting to some of the largest civil waterworks projects in the world. They operate many different apps to support data collection, modeling, and other core line of business scenarios but have a particularly critical one used to model storm surge. This app was initially built many years ago and has been regularly updated over time to incorporate new data sources and scientific knowledge. It's built on Red Hat, Oracle, and Java, and runs a significant amount of proprietary business logic. It's also a seminal example of the evolution described earlier: 12 years ago, this app ran on bare metal IBM xSeries servers in a datacenter and was then virtualized to run in a VMware ESX VM. Later, that VM was converted to run in Azure. Throughout these transitions, the app itself has stayed largely the same, indeed it was this compatibility that made these moves feasible.

How are Containers different than VMs?

Virtual Machines (VMs) virtualize the hardware; every VM carries its own OS on top of the hypervisor and host OS. Containers vitualize the OS—every container has its ownCPU, memory, block I/O, networkstack, but shares the same kernel as other containiers on the same host.



CHALLENGES

Running any 14 year old app is going to have its challenges and this one is no different. Due to the myriad of dependencies and versioning of different components, having a consistent dev and test environment was a constant struggle. Similarly, it was hard to deploy new builds of the app because each component was installed and configured separately and required specific versions that often conflicted with each other. Finally, the disaster recovery flow was quite complicated because of these attributes and required a detailed runbook operations teams needed to manually follow. Moving to containers delivered immediate benefits for this customer - their dev team immediately became more productive because of the consistency that containers provided between dev, test, and prod. The deployment process went from being an error-prone set of manual steps that often required hours to a simple docker run. Disaster recovery benefitted from this same change and basically became the same flow as initial deployment. Some of the most important benefits, though, came in security.

This app uses many different components from many different vendors and projects. Some of these components had to use specific older versions while other parts used newer frameworks. The app is used in projects involving the US Army Corp of Engineers and other government agencies that require adherence to specific security configuration and compliance standards, which historically required the organization to manually verify each check. Finally, because the app is a legacy core with many new capabilities added over time, it was difficult for the organization to understand how to secure it. Traditional tools like IDS/IPS and host firewalls required significant manual design and debug and often broke when the app was upgraded. This led the organization to have a looser set of security controls around it than they desired.

Containerization presented an opportunity to change all that. Once the app was moved to run in Docker, Twistlock was able to take advantage of the attributes of containers to simplify and automate many of these security challenges.

VULNERABILITY MANAGEMENT

Approaching vulnerability management in a modern, container-based manner means preventing risk throughout the software development lifecycle. This means not only providing visibility into vulnerabilities, but also actively preventing vulnerable images from progressing through the CI pipeline or being deployed in production.

TRADITIONAL APPROACH

Developers have some upstream tools to check for vulnerabilities in some libraries they used but not real visibility throughout lower layers of the stack. Vulnerability scans ware done after the app was built using different tools and a separate workflow and problems then had to be manually remediated. The ops team had to rely on human checks and processes to prevent vulnerable builds from being deployed and had no visibility into the vulnerability posture of what was running until they performed point in time scans using a few different tools.

CONTAINERIZED APPROACH

The dev team uses Twistlock's plugin for Jenkins as part of their build process to see all the vulnerabilities across the entire app in a single UI. The ops team has a threshold defined to prevent deployment of new versions if they contain vulnerabilities with medium or higher CVSS scores. Both dev and ops use Twistlock Vulnerability Explorer for a realtime view and stack ranking of the vulnerabilities across dev, test, and prod so they can identify what problems need to be addressed most urgently.

COMPLIANCE

Compliance encompasses laws or regulations like HIPPA, PCI, NIST, and many other acronyms that auditors live by and enterprise folks would rather not have to live by. Modernizing your security approach can get you audit-ready, real-time and historic compliance reporting without the hassle.

TRADITIONAL APPROACH

Compliance was typically thought of as an operational concern. However, this regularly led to problems being found in production that required dev changes to mitigate. The ops team had to rely on manually created documentation and testing to verify new app builds. There was little versioning control of the actual components being deployed; the ops team had to rely on manual processes to ensure the components being installed were the correct ones. The internal audit team had to run manual tools to check compliance and these checks were purely point in time with no ongoing assurance.

CONTAINERIZED APPROACH

Developers use Twistlock's Jenkins plugin to check the compliance posture of every image on every build so they can find and correct problems early on. Operations team centrally define compliance rules in Twistlock that monitor and enforce all the settings relevant to their specific needs, and automatically block the deployment of non-compliant images. The ops team uses Twistlock to ensure that only specific images, from the official, trusted repo are allowed to run in production. Ops and internal audit teams use Twistlock's Compliance Explorer feature to continuously monitor the state of all the hosts, containers, and images across the environment and ensure they're adhering to the best practices in NIST SP 800-190.

RUNTIME DEFENSE

Modern threats require layered runtime protection. When a new CVE is announced that affects a container image in your runtime environment, you need automated runtime protection that secures your entire environment: network, file system, processes and system calls. Additionally, in order to have visibility into running containers and potential attacks, you'll need feeds with constantly updated Malware and APT data that can stop attacks before they happen.

TRADITIONAL APPROACH

As with many organizations running complex legacy apps, runtime defense was minimal. The organization was simply unable to keep up with the manual effort required to tune IDS and IPS rules for every build or to create a comprehensive, specific network defense policy for the app. Instead, they relied primarily on border firewalls and some limited host based anti-malware capabilities to protect the app at runtime.

CONTAINERIZED APPROACH

Twistlock automatically creates a runtime model of every version of the app, every time it's deployed. This model describes what the app should do across four dimensions - process, network, file system, and system calls - and is correlated to the image digest, so every build has its own unique model, automatically created and tuned exactly for it. For example, if a developer simply makes a small change to one of the binaries, the model for that new image will automatically include the MD5 of the new binary and prevent execution of any abnormal or unknown ones. Similarly, the model automatically learns all the ports the app listens on, the specific versions of each binary bound to each of those sockets, and the characteristics of outbound network behaviors, such as egress IPs and DNS namespaces. This leads to more and significantly stronger layers of defense in depth at runtime - all while removing manual effort.

CONCLUSION

This organization’s initial steps into containers have delivered clear and immediate benefits across all phases of the app’s lifecycle. Development and test is more efficient, deployment and disaster recovery is greatly simplified, and they’re able to run multiple instances of the app without conflict in the same VMs, which has greatly increased density and lowered their Azure spend. Most importantly, though, it’s also immediately improved security in tangible ways. They have greater visibility into the vulnerability posture of the app throughout it’s lifecycle. Compliance has evolved from being an entirely manual and time consuming overhead into an automated part of the build, ship, and run phases. Runtime defense is significantly strengthened from some basic host antimalware to comprehensive controls tailored specifically for every build of the app. Migrating traditional apps to containers isn’t the end state of the journey, but it’s a great first step that delivers real benefits, builds your operational knowledge, and accelerates the larger migration to this next wave of enterprise infrastructure.

ABOUT TWISTLOCK

TWISTLOCK IS CLOUD NATIVE CYBERSECURITY

Twistlock is the leading provider of container and cloud native cybersecurity solutions for the modern enterprise. From precise, actionable vulnerability management to automatically deployed runtime protection and firewalls, Twistlock protects applications across the development lifecycle and into production. Purpose built for containers, serverless, and other leading technologies - Twistlock gives developers the speed they want, and CISOs the control they need.

LEARN MORE AT

Twistlock.com