DOCKER PRODUCTION DEPLOYMENT SECURITY CONSIDERATIONS

Part of our series on Docker. Check out How to Introduce Docker Containers in Enterprise, Docker 101, and Top 4 Docker Management Tips

Docker has been garnering a lot of attention in the enterprise world, and rightfully so. Companies are looking at it and adopting (at least in development environments) at a rate not often seen in in the enterprise world.

Docker has developers, cloud providers, and operating system (OS) vendors excited for various reasons. For developers, Docker makes it easy to package their software once to be deployed and run anywhere. Cloud providers are interested as it gives them a way to increase their tenants and application density per server, potentially reducing costs. OS vendors need to be excited as use of Docker potentially reduces the number of operating system licenses that companies need in their environments.

Docker images (containers) are packages of all dependencies and their binaries. Developers like this as containers will start and run on any machine the way they were intended to. This helps developers focus on the app, instead of environment issues like dependencies and their versions. Starting and stopping a container is fast and trivial. All of this means greater agility in setting up apps (even their different versions) on any Docker host. Because of this agility, Docker has been integrated heavily into continuous integration (CI) development workflows. However, using Docker for continuous deployment or for deploying applications into production can actually cause delays

because of the many security considerations that exist for production.

The security considerations can be split into two major components:

1. **Host Machine/OS:**

Even though Docker containers are isolated from one another using *cgroups* and *namespaces*, they still interact with the host OS kernel through the Docker daemon. Not properly secured, this creates the potential for a hacker to use one container to gain access to all of the other containers on the same Docker host.

Docker has a great write-up on security that describes how Docker interacts with the host OS kernel and the related security best practices. This covers how to ensure users are set up in containers, why it's a bad idea to expose the Docker daemon REST API over http, and how to use kernel security features like Security-Enhanced Linux (SELinux), etc.

Because the Docker host is an operating system, it must be updated regularly with patches and configurations to ensure that it contains no security vulnerabilities. BMC's BladeLogic Server Automation can help set up Compliance and patching to ensure that the host OS is compliant.

The Center for Internet Security (CIS) has also published the CIS Docker benchmark that can be used to run compliance and to correct issues with the host OS. You can create a *component template* in BMC BladeLogic Server Automation to check and correct the host OS and Docker daemon configurations.

2. **Docker Images:**

One of the key values of Docker is a layered approach that makes it very easy to bundle the application along with required dependencies (packages) into a Docker image, which can then be run on a Docker host. Once a developer creates a Docker image, it can then be published to a Docker registry for download and reuse. These registries can be private or public, much like the Docker Hub registry which contains thousands of images which have been downloaded millions of times.

Developers and IT operations need to ensure that packages in these images do not have any security vulnerabilities, and if found, are quickly updated or patched. In a virtualized environment, live virtual machines can be updated through patching, configuration changes or application updates. In contrast, when a Docker image needs to be updated, the base image and/or the Dockerfile must be corrected, and then the image must be rebuilt and redeployed.

Conclusion:

Docker is a very exciting virtualization technology that has the potential to change the way that data centers deploy software and continuously update applications. However, before Docker can move into production environments, companies need to develop appropriate policies and procedures to identify and remediate security vulnerabilities. Unlike traditional virtual machine environments, where each VM contains an isolated operating system, Docker containers utilize the same host operating system and are thus not isolated from one another.

In addition, for many companies, changing their update process from simply patching a virtual machine to patching a Docker image, redeploying the image, and then updating the registry can be challenging. How do they ensure that Docker containers are scanned for vulnerabilities? How do they automate the updates across their environment? How do they ensure that only patched and

approved Docker images are deployed in the future? And finally, how do they ensure that changes are properly updated in service desk records and the CMDB?

BMC solutions can help by scanning Docker images and containers, automating remediation of vulnerabilities and providing self-service access to approved applications and containers. Learn more.