

Delivering Safer Apps with Docker Enterprise and Windows Server

SEPTEMBER 2018

Table of Contents

Introduction.....	3
Usable Security.....	5
Secure by Default Runtime	5
Application Secrets	7
Trusted Delivery: A Secure Software Supply Chain.....	8
Image Signing and Verification.....	8
Image Security Scanning	9
Windows Updates.....	10
Windows Code Integrity	10
Infrastructure Independence	11
Using Active Directory Service Accounts for Containerized Application Identity	11
Single Sign On with Active Directory.....	11
Role-based Access Control	11
Conclusion	12
Additional Resources	12

Introduction

Security is a journey not a destination. What's been deployed today may be found to have a vulnerability tomorrow. Operating systems like Windows Server 2003 and 2008, which were once trusted building blocks for critical applications, are now potential liabilities as time goes on and maintenance comes to an end and can open organizations to unnecessary security risks that are difficult to fix. Combined with an evolving threatscape with new application and cloud architectures, how can you ensure that your organization is protected from the full gamut of application security pitfalls?



Quickly migrate
and reduce the risk
profile of legacy
Windows applications.

Today, many organizations are turning to containers with the [Docker Enterprise](#) container platform and considering modern operating systems like Windows Server 2016 and future versions of Windows Server as the foundation for their application platform.

This powerful combination enables organizations to both quickly migrate and reduce the risk profile of their legacy Windows Server application portfolio, and at the same time accelerates delivery of safer modern applications. Legacy applications are migrated simply and their attack surface area greatly reduced with containers. Containerized applications have a forensic history: coming from secured sources and are up to date with the latest security patches and automated through deployment to reduce risk and exposure.

When looking at application containers and the security surrounding them, Docker believes there are three key characteristics that directly correlate to safer apps.



**Usable
Security**



**Trusted
Delivery**








**Infrastructure
Independent**

Usable Security: Security features need to be usable by people at all stages of a software supply chain. It's been proven time and time again, people will actively circumvent overly onerous security policies and procedures. Both Docker Enterprise and Windows Server 2016 and later are designed to be secure by default, and are also built with flexible configurations that makes sense for developers and operators -- enabling workflows that work for them.

Trusted Delivery: By their very nature applications move around, whether that be from development to production or from the datacenter to cloud. Because of this, organizations need to ensure that their code can move safely from point A to point B with proof that nobody has tampered with it. Docker Enterprise and Windows Server 2016 and future releases of Windows Server provide integrated features to ensure the integrity of the applications that you deploy into production.

Infrastructure Independent: The promise of Docker is infrastructure agnostic application portability. This portability is not limited to the application code itself. With Docker, security configurations that are intrinsic to the application can also move from a developer's workstation through testing to production deployment - whether that's Windows Servers running in Azure or your datacenter - without any modification and all managed via Docker Enterprise.

THE KEY COMPONENTS OF CONTAINER SECURITY

 Usable Security	Secure defaults with tooling that is native to both dev and ops
 Trusted Delivery	Everything needed for a full functioning app is delivered safely and guaranteed to not be tampered with
 Infrastructure independent	All of these things in your system are in the app platform and can move across infrastructure without disrupting the app
  Safer Apps	



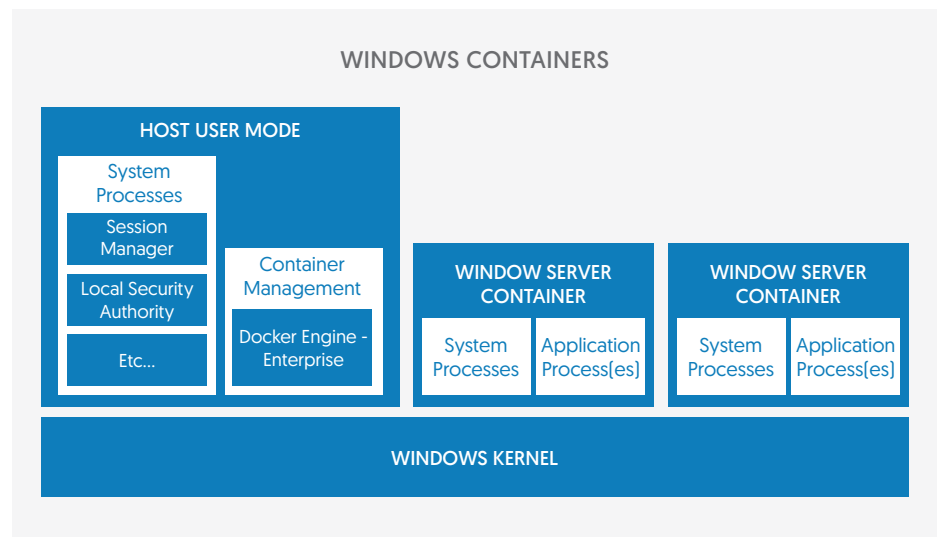
The result of nearly
4 years of joint
engineering effort
between Microsoft
and Docker.

Usable Security

Usable security is defined by two key components: secure defaults and tooling that works for both development and operations teams. Overly complex security policies and technologies are inevitably rendered ineffective as people find ways to work around them rather than with them. With Docker Enterprise and Windows Server 2016 and beyond, security features are designed to provide optimal application security, including protecting sensitive data, while integrating easily into existing workflows for both developers and operations professionals.

Secure by Default Runtime

Docker containers have become a critical component for developers in organizations of any size. Windows Server 2016 and future releases of Windows Server ship with support for Docker Windows Server Containers. Docker Windows Server Containers are powered by [Docker Engine - Enterprise](#), and are the result of over 3 years of joint engineering effort between Microsoft and Docker.



Docker Windows Server Containers provide isolation for key system resources including the Windows namespace, system processes, file system and registry.

Registry and Filesystem

Each container writes changes to its own instance of the registry and file system, isolating it from the host and other containers.

Namespace

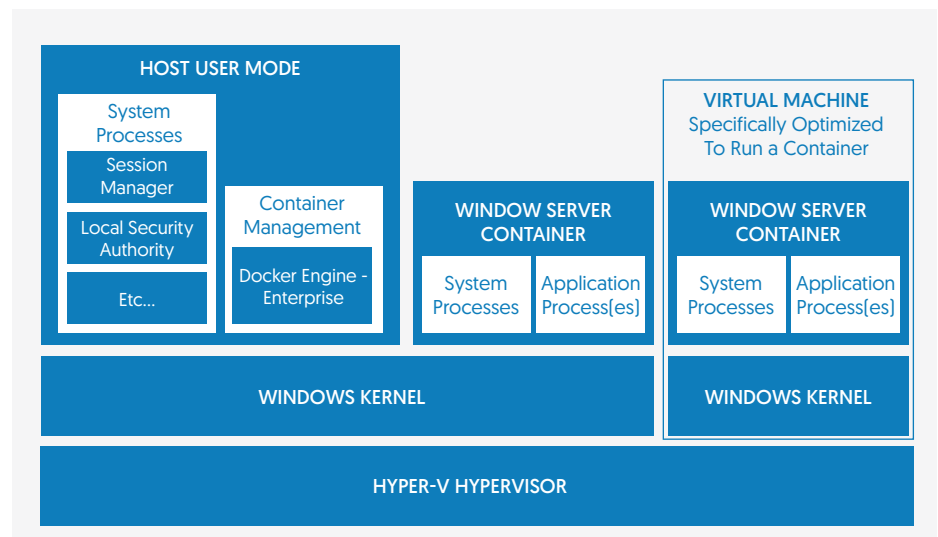
Instead of just allowing changes to the state within the container, namespace isolation on Windows renders most privileged APIs unusable. Drivers cannot be loaded in a container, and therefore cannot change the available attack surface. Containers have their own separate system services such as LSASS, service control manager, and task scheduler to provide common Windows functionality inside the container.

Users

Separation between well-known Windows privilege levels including LocalSystem, Administrators, and Users are preserved in containers. By default, all processes are also created as "ContainerAdministrator" instead of the "LocalSystem" process to separate privilege by default.

Resources

Resource limitations can be applied for CPU, memory, disk usage, and disk throughput for each Docker Windows Server Container too. These tools let you protect the performance of your important applications running in containers while still driving up overall utilization.



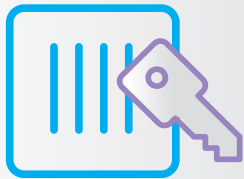
Hyper-V Isolation adds another layer of isolation by putting the container in a specially optimized Hyper-V virtual machine with a completely separate Windows kernel instance. You can start a container with Hyper-V isolation using a normal docker run command and adding the `-- isolation=hyperv` option.

For example:

```
docker run --isolation=hyperv -d -p 8000:8000 --name mysite iis-site
```

will start a copy of an IIS-based web site using Hyper-V isolation.

You don't need to take any extra steps to setup a VM, configure it, or remove it later. Everything is automatic and optimized to run your container. You can even run containers with and without Hyper-V isolation on the same machine to deploy each individual container with the level of security it needs. Additionally, there is no change needed to Docker images in order for them to work with Hyper-V isolation. Docker images work the same with both Hyper-V isolation as well as standard Docker Windows Server Containers. Hyper-V isolation make it easy to meet your organization's security and regulatory requirements without adding complexity.



Docker Enterprise Secrets provide the ability to store sensitive information separate from the code and configuration.

Application Secrets

When deploying and orchestrating services, administrators often need to configure those services with sensitive information like passwords, TLS certificates, or private keys.

Docker Enterprise provides IT professionals the ability to store this sensitive information, also known as secrets, in a secure way.

In Docker Enterprise, a secret is a blob of data, such as a password, SSH private key, SSL certificate, or another piece of data that should not be transmitted over a network or stored unencrypted in a Dockerfile or in your application's source code. With Docker Enterprise, you can use Docker secrets to centrally manage this data and securely transmit it to only those containers that need access to it. Secrets are encrypted during transit and at rest. A given secret is only accessible to those services which have been granted explicit access to it, and only while those service tasks are running. Secrets also allow for clearer separation of concerns, as developers and even most operators do not need to explicitly know these sensitive details for the application to function; instead, they are supplied to the running container and can be managed independently.

Trusted Delivery: A Secure Software Supply Chain

Ultimately IT professionals want to know that the software they have deployed in their environments comes from trusted sources, and that their code is free from vulnerabilities. But, as often is the case, this is easier said than done. However, with Docker Enterprise, system administrators can establish a secure software supply chain by setting policies to ensure only trusted code is deployed. In addition, the code and its dependencies can be scanned vulnerabilities and verified throughout the application lifecycle.

Image Signing and Verification

Knowing that what's running in your environment at any given time came from a trusted source is critical to protecting your information assets. But, how can you ensure the images being deployed in your organization are coming from a trusted source? One way is to have those images digitally signed. In the past content signing has been a fairly complex problem, and existing solutions were not well suited to containers; in fact, many failed to meet the "usable security" goals outlined above and thus never made it in to the application supply chain in a meaningful way.

Docker Enterprise solves the challenges associated with image signing and verification of containers via Docker Content Trust. Unlike PGP and other tools, Docker Content Trust has been purpose-built to make the software supply chain secure and auditable. Docker Content Trust uses The Update Framework (TUF), a language-agnostic software update system, and is implemented in Notary, an open source tool built-in to Docker Enterprise which provides trust over any content.

The core principle in Docker Content Trust is that each principal - whether human or automated process - should sign the image when they are done handling it. For example, when testing is complete in a CI pipeline, the CI tool can sign a statement asserting that fact. Before each step in the pipeline, the responsible party will also verify all signatures. A container that has been tampered with will be detected because either its hash will not match the signatures, or it will be missing signatures from each of the necessary parties.

Signatures for Docker Content Trust are stored in separate metadata store to promote separation of concerns. Because the system derives its security from the cryptography that it uses, the metadata store itself does not have to be trusted. Instead, each system and repository has its own keys. These use the trust-on-first-use principle by default, but additional verification is possible.



Automate digital verification of the source and contents of Docker container images as part of a **secure supply chain**.



Docker Enterprise can scan images to verify they are free from known vulnerabilities or exposures.

Image Security Scanning

As previously mentioned, security doesn't stop when an application is deployed. It's important to keep track of any new vulnerabilities that are discovered as they may affect currently running applications. This can be a fairly intensive task without some level of automation.

Thankfully registry services included in Docker Enterprise automate this process. Docker Enterprise registry can scan images in your repositories to verify that they are free from known security vulnerabilities or exposures. The results of these scans are reported for each image version, or tag. Scans run either on demand or automatically on any Docker push to the repository.

The image security scanning process is straightforward, and thorough. First the scanner performs a binary scan on each layer of the image, identifies the software components in each layer, and indexes the SHA of each component. A binary scan evaluates the components on a bit-by-bit level, so vulnerable components are discovered no matter what they're named, even if they're statically-linked, and regardless of if they're included on a distribution manifest.

The scan then compares the SHA of each component against the Common Vulnerabilities and Exposures (CVE®) database installed on your private Docker Enterprise registry instance. The CVE database is a dictionary of known information security vulnerabilities. When the CVE database is updated, the service reviews the indexed components for any that match newly discovered vulnerabilities.

By default, **Docker Security Scanning runs automatically when a Docker image is pushed to an image repository.**

If your registry instance is configured in this way, you do not need to do anything once your image push completes - the scan runs automatically, and the results are reported as part of the image details after the scan finishes.

Scanning functionality is role-based, furthering the concept of separation of concerns. Only users with write access to a repository can manually start a scan. Users with read-only access can view the scan results, but cannot start a new scan.

Windows Updates

Working alongside Docker Security Scanning, Microsoft Windows Updates can ensure that your Windows Server operating system is always up to date. Microsoft publishes Certified Windows Server base operating system images on the [Docker Store](#). The Certified designation indicates that both Microsoft and Docker collaborate to deliver these images, including best practices for providing secure images. These images will be updated the same day as new Windows security updates are released. Rather than going through the hassle of maintaining their own base images, IT practitioners can build their projects on top of the certified Microsoft images and rest assured they have the latest security updates. All of this makes it easy to bring the official Windows Server images into existing continuous delivery and deployment workflows and include Windows updates in the same process as all of your other deployments.

Windows Code Integrity

Windows Code Integrity policies cryptographically validate all Windows system processes in containers with the same policy applied to the host. This ensures the Windows container images are genuine and prevents tampered Windows binaries from running in containers

Anti-malware products, including Windows Defender on Windows Server 2016 and later and Windows 10, scan all containers as they are pulled and extracted onto the host. Containers containing malware can be detected and blocked before they are ever executed.





Use Active Directory to identify and authenticate services running in Docker Windows Server Containers.

Infrastructure Independence

A core tenant of Docker Enterprise is freedom of choice - the choice to move your application seamlessly between environments: from dev to QA to production or from a VM to bare metal to a cloud instance. With Docker Enterprise and Windows Server 2016 and later you can also rest assured that not only does your application move seamlessly, but so do its security requirements: signatures, secrets, and scanning included. Additionally Docker Enterprise integrates into your existing Windows Server environment.

Using Active Directory Service Accounts for Containerized Application Identity

Active Directory is the built-in service used for discovery, search and replication of user, computer, and service account information on Windows. It is commonly used for authentication and authorization between users and services, or between services on different machines. This is simpler to manage and more secure than locally stored usernames & passwords because the Active Directory provides a single place to handle authorization, revocation, and password rollovers. Servers are typically Active Directory domain-joined today, which gives each server a unique identity in the Active Directory domain and enables domain-joined services. Services running as Local System or Network Service can authenticate connections using the server's domain account.

Docker Windows Server Containers can use a similar mechanism to run as an Active Directory Group Managed Service Account. This enables applications running in a container such as a website to authenticate with other services such as a backend SQL Server using an Active Directory identity. There is no password or certificate private key stored in the container image that could be inadvertently exposed, and the container can be redeployed to development, test, and production environments without being rebuilt to change stored passwords or certificates.

Single Sign-on with Active Directory

In addition to enabling new methods for containerized Windows applications to authenticate on the network, Active Directory can also be used to provide user and group synchronization with Docker Enterprise. Docker Enterprise allows for single sign-on between the Docker Enterprise registry and control plane. User accounts can be managed locally, or they can be synchronized with your Active Directory.

Role-based Access Control

Docker Enterprise allows administrators to apply fine-grained role-based access control (RBAC) to the Docker Enterprise platform. Administrators can control which users can run an application, the nodes, networks and storage on which it can run, and even specific actions groups and individuals are permitted to take. Docker Enterprise provides tremendous flexibility in how access control policies are defined in your environment.

Conclusion

As was stated at the beginning of this paper, container security is a journey not a destination. A system that is secure today, may not be secure tomorrow. The critical mass of applications that still run on legacy platforms like Windows 2003 and 2008 is proof that the journey is continuous. That's why IT professionals need a container platform like Docker Enterprise that improve the security of their legacy applications and meet the needs of new, modern applications. The solution must provide not only the highest level of security, but also be designed to ensure the integrity of the applications that run on them from the moment they are deployed until they moment they are retired.

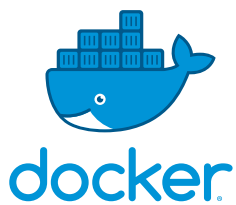
The combination of Windows Server 2016 and beyond for Docker Windows Server Containers with Docker Enterprise offer an unparalleled suite of features to ensure that information resources are protected continuously from the moment they are deployed. Working together with our business partners, Docker and Microsoft can help assess your traditional application landscape and your future requirements, to develop a solution that works for both.



Additional Resources

Learn more about the Docker and Microsoft partnership
www.docker.com/microsoft

Learn more about Docker Enterprise
www.docker.com/enterprise



www.docker.com